

Objective Validation of Airport Terminal Architecture using Agent-based Simulations

by

Karam Singh Hunjan

A thesis

presented to the Univeristy of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Architecture

Waterloo, Ontario, Canada, 2021

© Karam Singh Hunjan 2021

Author's Declaration

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

This thesis explores how airport terminal architecture is tested before it is built. The purpose of testing is to make sure an architectural layout aligns with the rest of the airport's systems. The design of a terminal is a long and expensive process that must accommodate tens of thousands of passengers every hour, the movement of logistics, and control of security. Evaluating spaces for that many people can be difficult to measure, which can result in architects relying on their intuition and experience to judge the impact of a layout for daily operations without objective validation. It is not practical for designers to build a complete airport to see how it works and make renovations after finding aspects that have poor performance. As a result, testing airports requires using mathematical models and simulations to validate how well different systems work together.

Designers try to validate architectural layouts in airport terminals by using crowd simulations to approximate passenger behaviour. Existing research in civil engineering and computer science has shown how mathematical models can predict patterns of human activity in the built environment on a large scale. However, these simulations have primarily focused on either modelling passengers as a process flow or people in emergency building evacuation. As a result, existing agent navigation does not consider how passengers use the surrounding architecture for decision-making during daily airport interactions. When passengers enter a terminal for the first time, they can be unaware of what they need to do or how to get there. Instead, passengers rely on using their perception of the environment (the architecture) to inform them what to do. However, there currently are no methods that incorporate architectural perception to validate a building layout in these conditions.

This thesis develops an agent-based simulation to validate how well architectural layouts align with the daily operations of an airport terminal. It quantifies the value of a spatial arrangement as a function of people's interactions in a given space. The model approximates human behaviour based on statistics from existing crowd simulations. It uses spatial analysis, like the isovist and graph theory, for agent navigation and measuring architectural conditions. The proposal incorporates agent perception to provide feedback between people's decision-making and the influence of the surrounding space. The thesis calculates architectural value using normalized passenger priorities based on typical processing and non-processing airport domains. The success of a terminal layout is dependent on the agent's ability to complete airport

processing and fulfill their priorities. The final value of an architectural layout is determined using statistical methods to provide a probability distribution of likely values.

The proposed agent simulation and mathematical models are built using Unity software, which is used to perform several simulation tests in this thesis. Basic functional components of the simulation are validated using existing crowd modelling standards. Tests are also performed to illustrate how different agent perception and priorities influence the value of architectural spaces. Monte Carlo simulations are created for simple terminal layouts to illustrate how changing the floor plan of a security area affects the architectural value for departing passengers. Finally, the architectural values of two real airport terminals are compared against an established passenger experience survey in a basic simulation model. The results of the testing shows that the agent simulation can differentiate between different architectural conditions, within reason, depending on the passengers' priorities.

Acknowledgements

I would like to thank my supervisor Jonathan Enns. Thank you for your guidance throughout the thesis process, your enthusiasm, and your patience with me as I worked through my own thoughts.

I would like to thank my committee member John Straube. Thank you for your valuable feedback and insightful discussions.

I would also like to acknowledge Nelson Oliveira. I am grateful for the time I spend working with you at the GTAA, and for you taking the time to discuss my thesis with me when I started my research.

Thank you to my parents for your love and support. Thank you to my mom for your conversations, feedback, and encouragement. Thank you to my dad for your discussions, thoughts, and positivity.

Table of Contents

Author's Declaration	iii
Abstract	v
Acknowledgements	vii
List of Figures	xiii
Part 0 Introduction	1
0.0 Motivation	1
0.1 Problem	3
0.2 Goals	3
0.3 Hypothesis	4
0.4 Expected Results	4
0.5 Thesis Structure	5
Part 1 Context	7
1.0 Architectural Intuition	8
1.1 Airport Terminal Design	14
1.2 Verification and Validation	28
1.3 Probability and Statistics	40
1.4 Simulation Modelling	53
Part 2 Modelling Concepts	85
2.0 Agent-based Modelling	86
2.1 Human Perception	90
2.2 Spatial Analysis	106
2.3 Value Theory	117

Part 3	Simulation Framework	149
3.0	Simulation Components	150
3.1	Agent-related Classes	156
3.2	A* Pathfinding Classes	190
3.3	Airport Architecture Classes	203
3.4	Simulation Utility Classes	227
3.5	Assumptions and Limits	236
Part 4	Simulation Tests	243
4.0	Verification and Validation Tests	244
4.1	Component Tests	264
4.2	Terminal Tests	288
4.3	Airport Tests	317
Part 5	Conclusion	333
5.0	Results and Findings	334
5.1	Ideal Models	337
5.2	Impacts	340
5.3	Future Work	346
5.4	Summary of Conclusion	349
	Letter of Copyright Permission	351
	References	355
	Appendix A	367

List of Figures

Figure	Page	Description Reference
--------	------	--------------------------

Part 0: Introduction

Fig.0.0.a	2	Comparing two different iterations of a floor plan. Drawing by author.
-----------	---	---

Part 1: Context

Fig.1.0.a	10	Floor plan of Eero Saarinen's TWA Flight Centre (1961). Retrieved from: Fiederer, Luke. "AD Classics: TWA Flight Center / Eero Saarinen". ArchDaily. 2016-06-16. https://www.archdaily.com/788012/ad-classics-twa-flight-center-eero-saarinen .
Fig.1.0.b	10	Floor plan of Eero Saarinen's Dulles International Airport main terminal building (1962). Cited from: Yukio Futagawa, ed. <i>Global Architecture: TWA Terminal Building, Kennedy Airport, New York, and Dulles International Airport, Chantilly, Virginia</i> . Tokyo: A.D.A. Edita Tokyo, 1973. plan, p45. Retrieved from: Great Buildings. Dulles Airport. Accessed March 2021. http://www.greatbuildings.com/buildings/Dulles_Airport.html .
Fig.1.0.c	12	Section of TWA Flight Center. Retrieved from: National Park Services. "Trans World Airline Flight Center". National Register of Historic Places. (September 7, 2005): Section 11, page 7.
Fig.1.0.d	12	Section of Dulles terminal building. Cited from: Saarinen, Eero. <i>Eero Saarinen On His Work</i> . New Haven, CT: Yale University Press, 1968. section drawing, p108. Retrieved from: Great Buildings. Dulles Airport. Accessed March 2021. http://www.greatbuildings.com/buildings/Dulles_Airport.html .
Fig.1.1.a	17	Basic airport terminal layout. Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Airport Passenger Terminal Planning and Design, Volume 1: Guidebook". <i>Washington, DC: The National Academies Press</i> . (2010): 173. Sourced from: Daileda, David A. "Considerations for Selecting a Terminal Configuration," FAIA, FAA White Paper.

- Fig.1.1.b 17 **Linear terminal.**
Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Airport Passenger Terminal Planning and Design, Volume 1: Guidebook". *Washington, DC: The National Academies Press. (2010): 173.*
Sourced from: Daileda, David A. "Considerations for Selecting a Terminal Configuration," , FAIA, FAA White Paper.
- Fig.1.1.c 19 **Pier terminal.**
Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Airport Passenger Terminal Planning and Design, Volume 1: Guidebook". *Washington, DC: The National Academies Press. (2010): 173.*
Sourced from: Daileda, David A. "Considerations for Selecting a Terminal Configuration," , FAIA, FAA White Paper.
- Fig.1.1.d 19 **Multi-pier terminal.**
Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Airport Passenger Terminal Planning and Design, Volume 1: Guidebook". *Washington, DC: The National Academies Press. (2010): 173.*
Sourced from: Daileda, David A. "Considerations for Selecting a Terminal Configuration," , FAIA, FAA White Paper.
- Fig.1.1.e 20 **Satellite terminal.**
Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Airport Passenger Terminal Planning and Design, Volume 1: Guidebook". *Washington, DC: The National Academies Press. (2010): 173.*
Sourced from: Daileda, David A. "Considerations for Selecting a Terminal Configuration," , FAIA, FAA White Paper.
- Fig.1.1.f 21 **Satellite terminal with Automated People Mover (APM) system.**
Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Airport Passenger Terminal Planning and Design, Volume 1: Guidebook". *Washington, DC: The National Academies Press. (2010): 173.*
Sourced from: Daileda, David A. "Considerations for Selecting a Terminal Configuration," , FAIA, FAA White Paper.
- Fig.1.1.g 23 **Toronto Pearson Airport's hourly passenger movement forecasts.**
Retrieved from: GTAA. "Toronto Pearson International Airport Master Plan 2017-2037". Greater Toronto Airports Authority. (2017): 38.
- Fig.1.1.h 23 **Toronto Pearson Airport's peak-hour passenger movement forecasts base on existing and projected flight schedules.**
Retrieved from: GTAA. "Toronto Pearson International Airport Master Plan 2017-2037". Greater Toronto Airports Authority. (2017): 36.
- Fig.1.1.i 25 **Experiment showing the level of service (LOS) from A, the least dense, to F, the most crowded.**
Retrieved from: Fruin, John J. *Designing for Pedestrians: A Level of Service Concept.* New York, 1970. 10.

Fig.1.2.a	31	<p>All verification and validation techniques on a spectrum of mathematical formality.</p> <p>Retrieved from: Balci, Osman. "Validation, Verification, and Testing Techniques Throughout the Life Cycle of a Simulation Study." <i>Annals of operations research</i> 53, no. 1 (December 1994): 131.</p>
Fig.1.2.b	33	<p>Simulation processes showing corresponding validation processes.</p> <p>Retrieved from: Robinson, Stewart. <i>Simulation: the Practice of Model Development and Use</i>. Chichester, England: Wiley, (2004): 211.</p>
Fig.1.3.a	43	<p>Graphs illustrating the law of large numbers from random sampling of new-born baby weights.</p> <p>Retrieved from: Watkins, Joseph. <i>An Introduction to the Science of Statistics: From Theory to Implementation Preliminary Edition</i>. University of Arizona: (2016): 180.</p>
Fig.1.3.b	45	<p>Monte Carlo method approaches the value for π based on the fraction of random points that fall inside the circle within a unit square.</p> <p>Retrieved from: Nicoguardo. "File:Pi 30K.gif". Wikimedia Commons. February 16, 2017. https://commons.wikimedia.org/wiki/File:Pi_30K.gif.</p>
Fig.1.3.c	45	<p>Graph showing of 100 random values between 0.0 and 1.0 in Excel.</p> <p>Drawing by author.</p>
Fig.1.3.d	47	<p>Some common probability distributions as a result of a Monte Carlo simulation, which informs statistical behaviour.</p> <p>Drawing by author.</p>
Fig.1.3.e	47	<p>Graphs illustrating the law of large numbers from random sampling of new-born baby weights.</p> <p>Retrieved from: Watkins, Joseph. <i>An Introduction to the Science of Statistics: From Theory to Implementation Preliminary Edition</i>. University of Arizona: (2016): 185.</p>
Fig.1.3.f	49	<p>A Galton board is a physical example of a Monte Carlo simulation, which shows how natural randomness can result in a normal probability distribution.</p> <p>Retrieved from: Argenton, Rodrigo. "File:Galton box.jpg". Wikimedia Commons. December 19, 2016. https://commons.wikimedia.org/wiki/File:Galton_box.jpg.</p>
Fig.1.3.g	49	<p>Galton Board Concept.</p> <p>Drawing by author.</p> <p>Based on drawing from: Galea, Alexander. "Galton's Peg Board and the Central Limit Theorem". WordPress. March 11, 2016. https://galeascience.wordpress.com/2016/03/11/galtons-peg-board-and-the-central-limit-theorem/.</p>

Fig.1.3.h	52	<p>RVLS Central Limit Theorem Simulation. Drawing by author. Based on drawing from: Lane, David et al. "Sampling Distribution". Onlinestatbook.com, Rice Virtual Lab in Statistics (RVLS), Rice University. Accessed February 2021. https://onlinestatbook.com/stat_sim/sampling_dist/index.html.</p>
Fig.1.4.a	55	<p>Graphical representation of a discrete system and a continuous system. Retrieved from: Banks, Jerry; Carson II, John S; Nelson, Barry L; Nicol, David M. <i>Discrete-Event System Simulation 4th ed.</i> Upper Saddle River, N.J: Pearson Prentice Hall, (2005): 11.</p>
Fig.1.4.b	57	<p>AirTop airspace simulation. Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Simulation Options for Airport Planning". <i>Washington, DC: The National Academies Press.</i> (2019): 17.</p>
Fig.1.4.c	57	<p>ARCport terminal simulation. Screen capture from: Proulx, Christian. "Complete Terminal Simulation". Youtube. March 1, 2014. https://www.youtube.com/watch?v=iaXdMO67goE.</p>
Fig.1.4.d	59	<p>A spreadsheet model calculating the area required for security screening. Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Airport Passenger Terminal Planning and Design, Volume 2: Spreadsheet Models and User's Guide". <i>Washington, DC: The National Academies Press.</i> (2010): 40.</p>
Fig.1.4.e	59	<p>Basic queuing node model. Drawing by author. Based on drawing from: Dt-rush-8. "Queuing Node Service Diagram". Wikimedia Commons. 8 December 2018. https://commons.wikimedia.org/wiki/File:Queueing_node_service_digram.png.</p>
Fig.1.4.f	61	<p>The process of a simulation study. Drawing by author. Based on drawing from: Banks, Jerry; Carson II, John S; Nelson, Barry L; Nicol, David M. <i>Discrete-Event System Simulation 4th ed.</i> Upper Saddle River, N.J: Pearson Prentice Hall. (2005): 13.</p>
Fig.1.4.g	65	<p>Dense crowds in a marathon (left) can be approximated as a fluid flow. Retrieved from: Zhou, Suiping, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Low, Feng Tian, Victor Tay, Darren Ong, and Benjamin Hamilton. "Crowd Modeling and Simulation Technologies." <i>ACM Transactions on Modeling and Computer Simulation (TOMACS)</i> 20, no. 4 (October 1, 2010): 6.</p>
Fig.1.4.h	65	<p>Crowd model using particles. Retrieved from: Zhou, Suiping, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Low, Feng Tian, Victor Tay, Darren Ong, and Benjamin Hamilton. "Crowd Modeling and Simulation Technologies." <i>ACM Transactions on Modeling and Computer Simulation (TOMACS)</i> 20, no. 4 (October 1, 2010): 7.</p>

- Fig.1.4.i 67 **The game Planet Coaster uses a fluid model to simulate crowds in an amusement park.**
Retrieved from: McCarthy, Owen. "Game Design Deep Dive: Creating Believable Crowds in Planet Coaster." Gamasutra Article, January 4, 2017.
https://www.gamasutra.com/view/news/288020/Game_Design_Deep_Dive_Creating_believable_crowds_in_Planet_Coaster.php.
- Fig.1.4.j 69 **Evacuation simulation that uses perception.**
Retrieved from: Liu, Z, Liu, T, Ma, M, Hsu, H-H, Ni, Z, Chai, Y. A perception-based emotion contagion model in crowd emergent evacuation simulation. *Comput Anim Virtual Worlds*. 2018; 29:e1817. p.9.
- Fig.1.4.k 69 **Evacuation simulation that uses social forces.**
Abdelhak, Haifa; Ayesha, Aladdin; Olivier, Damien. "Cognitive Emotional Based Architecture for Crowd Simulation". *Journal of Intelligent Computing*, June 2012, 2012. Vol. 3 (2). 64.
- Fig.1.4.l 71 **List of established simulation tools for airport terminal analysis.**
Retrieved from: National Academies of Sciences, Engineering, and Medicine. "Simulation Options for Airport Planning". Washington, DC: The National Academies Press. (2019): 44.
- Fig.1.4.m 72 **A MassMotion simulation in Toronto Union Station.**
Retrieved from: Arup. "MassMotion". Expertise, Services, Digital. Accessed December 2020. <https://www.arup.com/expertise/services/digital/massmotion>.
- Fig.1.4.n 72 **Typical components in a MassMotion environment.**
Retrieved from: Oasys. "MassMotion Help Guide," July 2019. <https://www.oasys-software.com/wp-content/uploads/2019/06/MassMotion-10.0-Help-Guide.pdf>. 15.
- Fig.1.4.o 74 **Agent feelers used to identify other agents and local targets for navigation.**
Retrieved from: Oasys. "MassMotion Help Guide," July 2019. <https://www.oasys-software.com/wp-content/uploads/2019/06/MassMotion-10.0-Help-Guide.pdf>. 290.
- Fig.1.4.p 74 **MassMotion displaying passenger density to show congested areas in the concourse of Toronto Union Station.**
Retrieved from: Hoy, Gregory, Erin Morrow, and Amer Shalaby. "Use of Agent-Based Crowd Simulation to Investigate the Performance of Large-Scale Intermodal Facilities: Case Study of Union Station in Toronto, Ontario, Canada." *Transportation Research Record* 2540, no. 1 (January 2016): 25.
- Fig.1.4.q 76 **Flow chart of a station ticket service counter simulation in Arena.**
Created by author.
- Fig.1.4.r 76 **Flow chart of a plane gating simulation in Arena.**
Created by author.

Fig.1.4.s	78	Flow chart of a healthcare simulation in FlexSim. Recreated by author.
Fig.1.4.t	78	3D model of a healthcare simulation in FlexSim. Recreated by author.
Fig.1.4.u	80	Simulation comparison between Arena, FlexSim, Quelea, and Unity. Drawing by author.
Fig.1.4.v	81	Sample simulations and station test for Arena, FlexSim, Quelea, and Unity. Drawing by author.

Part 2: Modelling Concepts

Fig.2.0.a	89	How agents interact with the environment. Drawing by author. Based on drawing from: Liu, Ao (Leo). “Dynamic Visualizations: Developing a Framework for Crowd-Based Simulations”. M.Arch thesis, University of Waterloo, 2020. 85.
Fig.2.1.a	91	Environments are made up of objects and processes. Drawing by author.
Fig.2.1.b	93	Smith's (2001) ontological marks of an environment and examples in an airport. Drawing by author. Based on drawing from: Raubal, Martin. “Agent-Based Simulation of Human Wayfinding: A Perceptual Model for Unfamiliar Buildings”. PhD diss., Vienna University of Technology, October 2001. 64.
Fig.2.1.c	95	Elements of an airport classified as a substance or medium. Drawing by author. Based on drawing from: Raubal, Martin. “Agent-Based Simulation of Human Wayfinding: A Perceptual Model for Unfamiliar Buildings”. PhD diss., Vienna University of Technology, October 2001. 65.
Fig.2.1.d	95	Further categorization of airport terminal architecture. Drawing by author.
Fig.2.1.e	99	Category of affordances in an airport and architectural elements. Drawing by author. Based on drawing from: Raubal, Martin. “Agent-Based Simulation of Human Wayfinding: A Perceptual Model for Unfamiliar Buildings”. PhD diss., Vienna University of Technology, October 2001. 68.
Fig.2.1.f	101	Gibson's representation of a person's field of view. Retrieved from: Gibson, James J. <i>The Ecological Approach to Visual Perception</i> . New York: Psychology Press, (1986): 196.

- Fig.2.2.a 107 **An isovist is the area that can be seen from a single point.**
Retrieved from: Arabacioglu, Burcin Cem. "Using Fuzzy Inference System for Architectural Space Analysis." *Applied Soft Computing* 10, no. 3 (2010): 927.
- Fig.2.2.b 107 **The three geometries of space syntax.**
Retrieved from: Vaughan, Laura. "The Spatial Syntax of Urban Segregation." *Progress in Planning* 67, no. 3 (2007): 209.
- Fig.2.2.c 109 **Graphs showing the arrangement of connections from different rooms in a house.**
Retrieved from: Vaughan, Laura. "The Spatial Syntax of Urban Segregation." *Progress in Planning* 67, no. 3 (2007): 211.
- Fig.2.2.d 109 **Enclosure defines a value based on the number of surrounding walls on a scale of 1 to 4.**
Retrieved from: Do, Ellen Yi-Luen, and Mark D. Gross. "Tools for Visual and Spatial Analysis of CAD Models." *CAAD Futures* 1997, 1997, 194.
- Fig.2.2.e 111 **Enclosure defines a value based on the number of surrounding walls on a scale of 1 to 4.**
Retrieved from: Turner, Alasdair, Maria Doxa, David Osullivan, and Alan Penn. "From Isovists to Visibility Graphs: A Methodology for the Analysis of Architectural Space." *Environment and Planning B: Planning and Design* 28, no. 1 (2001): 108.
- Fig.2.2.f 111 **A visibility graph, where each vertex is a point in space, and the lines are the visible connections.**
Retrieved from: Arabacioglu, Burcin Cem. "Using Fuzzy Inference System for Architectural Space Analysis." *Applied Soft Computing* 10, no. 3 (2010): 927.
- Fig.2.2.g 113 **An undirected graph with 5 edges and 5 vertices.**
Drawing by author.
- Fig.2.2.h 113 **A directed graph.**
Drawing by author.
- Fig.2.2.i 113 **A path from node v_1 to node v_4 .**
Drawing by author.
- Fig.2.2.j 113 **A weighted graph.**
Drawing by author.
- Fig.2.2.k 113 **Lowest cost path between node A and node B has a cost of 7.**
Drawing by author.
- Fig.2.2.l 115 **Pathfinding solved using Dijkstra's search algorithm.**
Retrieved from: Bhattacharya, Subhrajit. "File:Dijkstras progress animation.gif". Wikimedia Commons. April 13, 2011.
https://commons.wikimedia.org/wiki/File:Dijkstras_progress_animation.gif.

Fig.2.2.m	115	Pathfinding solved using an A* search algorithm. Retrieved from: Bhattacharya, Subhrajit. "File:Astar progress animation.gif". Wikimedia Commons. April 13, 2011. https://en.wikipedia.org/wiki/File:Astar_progress_animation.gif .
Fig.2.3.a	120	First two architect's priorities and school floor plans. Retrieved from: Lera, Sebastian G. "Architectural Designers' Values and the Evaluation of Their Designs." <i>Design studies</i> 2, no. 3 (1981): 135.
Fig.2.3.b	122	Comparing utility model values with overall values for each architect. Retrieved from: Lera, Sebastian G. "Architectural Designers' Values and the Evaluation of Their Designs." <i>Design studies</i> 2, no. 3 (1981): 136.
Fig.2.3.c	124	Simple analytic hierarchy process structure. Drawing by author. Based on drawing from: Sander, Lou. "File:AHPHierarchy1.1.png". Wikimedia Commons. 24 February 2009. https://commons.wikimedia.org/wiki/File:AHPHierarchy1.1.png .
Fig.2.3.d	126	Eigenvectors after a linear transformation. Drawing by author. Based on animation from: 3Blue1Brown. "Eigenvectors and eigenvalues Essence of linear algebra, chapter 14". Youtube, September 15, 2016. https://www.youtube.com/watch?v=PFDu9oVAE-g .
Fig.2.3.e	128	Priority matrix and corresponding eigenvector for each coloured ball. Drawing by author.
Fig.2.3.f	129	Job satisfaction priority matrix. Retrieved from: Saaty, Thomas L. "Modeling Unstructured Decision Problems — the Theory of Analytical Hierarchies." <i>Mathematics and computers in simulation</i> 20, no. 3 (1978): 155.
Fig.2.3.g	129	Company attributes matrices. Retrieved from: Saaty, Thomas L. "Modeling Unstructured Decision Problems — the Theory of Analytical Hierarchies." <i>Mathematics and computers in simulation</i> 20, no. 3 (1978): 155.
Fig.2.3.h	129	Company attributes eigenvectors. Retrieved from: Saaty, Thomas L. "Modeling Unstructured Decision Problems — the Theory of Analytical Hierarchies." <i>Mathematics and computers in simulation</i> 20, no. 3 (1978): 155.
Fig.2.3.i	132	Passenger-centred model hierarchy. Drawing by author.

Fig.2.3.j	132	Airport performance is dependant on processing domains and non-processing domains. Drawing by author. Based on drawing from: Wiredja, Dedy, Vesna Popovic, and Alethea Blackler. “A Passenger-Centred Model in Assessing Airport Service Performance.” Journal of Modelling in Management 14, no. 2 (May 10, 2019): 504.
Fig.2.3.k	134	Airport domains indicating attributes that are influenced by architecture. Drawing by author. Based on drawing from: Wiredja, Dedy, Vesna Popovic, and Alethea Blackler. “A Passenger-Centred Model in Assessing Airport Service Performance.” Journal of Modelling in Management 14, no. 2 (May 10, 2019): 503.
Fig.2.3.l	137	Passengers standing between columns and along the walls of a platform waiting to board a subway. Photo by: Mentatdgt. “Photography of People at Train Station”. Pexels. August 09, 2018. https://www.pexels.com/photo/photography-of-people-at-train-station-1311544/ .
Fig.2.3.m	139	9 factors for scoring architecture, with corresponding mathematical functions. Drawing by author.
Fig.2.3.n	143	General exponential decay function. Drawing by author.
Fig.2.3.o	143	A piece-wise decay function for a typical passenger waiting time. Drawing by author.

Part 3: Simulation Framework

Fig.3.0.a	151	Unity software user interface showing the scene environment models and property toolbars. Drawing by author.
Fig.3.0.b	155	Categories of script classes in Unity for the agent simulation. Drawing by author.
Fig.3.1.a	157	Agent following an A* path (black line) to a local target (white wire sphere). Image by author.
Fig.3.1.b	159	Process logic for the agent class. Drawing by author.
Fig.3.1.c	160	Key variables for the agent class, page 1. Drawing by author.
Fig.3.1.d	161	Key variables for the agent class, page 2. Drawing by author.

Fig.3.1.e	162	Key methods for the agent class, page 3. Drawing by author.
Fig.3.1.f	164	Population distribution, for age and gender. Retrieved from: IMO. "Guidelines for Evacuation Analysis for New and Existing Passenger Ships." International Maritime Organization (IMO). MSC.1/Circ.1238. (October 30, 2007): 6.
Fig.3.1.g	164	Passenger walking speeds. Retrieved from: IMO. "Guidelines for Evacuation Analysis for New and Existing Passenger Ships." International Maritime Organization (IMO). MSC.1/Circ.1238. (October 30, 2007): 8.
Fig.3.1.h	165	Samples of randomly assigned characteristics and priority matrices, page 1. Drawing by author.
Fig.3.1.i	165	Samples of randomly assigned characteristics and priority matrices, page 2. Drawing by author.
Fig.3.1.j	167	Process logic for the characteristics class. Drawing by author.
Fig.3.1.k	168	Key variables for the characteristics class, page 1. Drawing by author.
Fig.3.1.l	169	Key variables and methods for the characteristics class, page 2. Drawing by author.
Fig.3.1.m	170	Priority local class within the characteristics class, page 3. Drawing by author.
Fig.3.1.n	172	Agent perceives the gate sign, as shown by the blue line. The agent state is "read sign", as illustrated by the pink colour. Drawing by author.
Fig.3.1.o	172	Agent perceives check-in counters, as shown by the blue line. The agent state is "go to check-in", as illustrated by the blue colour. Drawing by author.
Fig.3.1.p	174	Process logic for the perception class, page 1. Drawing by author.
Fig.3.1.q	174	Process logic for the perception class, page 2. Drawing by author.
Fig.3.1.r	176	Key variables for the perception class, page 1. Drawing by author.
Fig.3.1.s	177	Key variables for the perception class, page 2. Drawing by author.

Fig.3.1.t	178	Key variables for the perception class, page 3. Drawing by author.
Fig.3.1.u	179	Key methods for the perception class, page 4. Drawing by author.
Fig.3.1.v	180	Key methods for the perception class, page 5. Drawing by author.
Fig.3.1.w	182	Generalized construction of the field of view. Drawing by author. Based on drawing from: Lague, Sebastian. “Field of view visualization (E02)”. Youtube. December 27, 2015. https://youtu.be/73Dc5JTCmKI?t=530 . 8:57.
Fig.3.1.x	182	Agent in front of a wall showing their field of view. Image by author.
Fig.3.1.y	183	Convex corners are refined by selecting a midpoint vector between a max and min viewpoint. Drawing by author. Based on drawing from: Lague, Sebastian. “Field of view visualization (E02)”. Youtube. December 27, 2015. https://youtu.be/73Dc5JTCmKI?t=1070 . 17:50.
Fig.3.1.z	183	A random direction vector for wandering is selected towards the longest visible direction, illustrated by the red line. Image by author.
Fig.3.1.za	185	Process logic for the field of view class. Drawing by author.
Fig.3.1.zb	186	Key variables for the field of view class, page 1. Drawing by author.
Fig.3.1.zc	187	Key variables for the field of view class, page 2. Drawing by author.
Fig.3.1.zd	188	Key methods for the field of view class, page 3. Drawing by author.
Fig.3.1.ze	189	Key methods for the field of view class, page 4. Drawing by author.
Fig.3.2.a	191	Simulated environments are divided into grid tiles for A* navigation. Drawing by author.

Fig.3.2.b	193	An example of walkable and unwalkable areas from a grid environment. Drawing by author.
Fig.3.2.c	192	Process logic for the grid class. Drawing by author.
Fig.3.2.d	194	Key variables for the grid class, page 1. Drawing by author.
Fig.3.2.e	195	Key variables and methods for the grid class, page 2. Drawing by author.
Fig.3.2.f	197	Key variables for the node class, page 1. Drawing by author.
Fig.3.2.g	198	Key methods for the node class, page 2. Drawing by author.
Fig.3.2.h	200	Pathfinding creates a path (black line) between nodes along a tiled grid to a target node (white wire sphere). Drawing by author.
Fig.3.2.i	201	Lague's A* pathfinding process calculates the cost of neighbouring nodes as a sum of its distance to the start and end nodes. Drawing by author. Based on drawing from: Lague, Sebastian. "A* Pathfinding (E01: algorithm explanation)". Youtube. December 16, 2014. https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXlocq5Umv3pMC9SPnKjfp9eGW&index=1 .
Fig.3.2.j	202	Key variables and methods for the pathfinding class. Drawing by author.
Fig.3.3.a	204	Basic geometry used to identify spatial areas. Drawing by author.
Fig.3.3.b	204	Spatial areas are referenced when agents move between them. Drawing by author.
Fig.3.3.c	206	Process logic for the architecture class. Drawing by author.
Fig.3.3.d	207	Key variables for the architecture class, page 1. Drawing by author.
Fig.3.3.e	208	Key methods for the architecture class, page 2. Drawing by author.

Fig.3.3.f	210	Illustration of interaction nodes (green) and exit nodes (red) in security screening. Drawing by author.
Fig.3.3.g	210	Illustration of queue spots (blue) in a queuing line. Drawing by author.
Fig.3.3.h	211	Process logic for the airport objects class. Drawing by author.
Fig.3.3.i	212	Key variables and methods for the airport objects class, page 1. Drawing by author.
Fig.3.3.j	214	A wayfinding sign illustrated with a viewpoint (blue) and two direction nodes (red) for Gate A and Gate B. Drawing by author.
Fig.3.3.k	215	Process logic for the signage class. Drawing by author.
Fig.3.3.l	216	Key variables and methods for the signage class, page 1. Drawing by author.
Fig.3.3.m	218	An example of a simulation schedule assigned in the Unity inspector properties, using the scheduling script, with 3 arrival points and 2 departure points. Screen capture by author.
Fig.3.3.n	220	Key variables for the scheduling class, page 1. Drawing by author.
Fig.3.3.o	221	Key variables for the scheduling class, page 2. Drawing by author.
Fig.3.3.p	223	Process logic for the itinerary class. Drawing by author.
Fig.3.3.q	224	Key variables for the itinerary class, page 1. Drawing by author.
Fig.3.3.r	225	Key variables for the itinerary class, page 2. Drawing by author.
Fig.3.3.s	226	Key methods for the itinerary class, page 3. Drawing by author.
Fig.3.4.a	229	Process logic for the agent spawner class. Drawing by author.

Fig.3.4.b	230	Key variables for the agent spawner class, page 1. Drawing by author.
Fig.3.4.c	231	Key variables for the agent spawner class, page 2. Drawing by author.
Fig.3.4.d	232	Key methods for the agent spawner class, page 3. Drawing by author.
Fig.3.4.e	233	Path request manager process. Drawing by author.
Fig.3.4.f	235	Heap tree process. Drawing by author. Based on drawing from: Lague, Sebastian. "A* Pathfinding (EO1: algorithm explanation)". Youtube. December 16, 2014. https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXlocq5Umv3pMC9SPnKjfp9eGW&index=1 .
Fig.3.4.g	235	Field of view editor displays "handles triangles". Drawing by author.

Part 4: Simulation Tests

Fig.4.0.a	245	IMO and NIST verification tests for evacuation simulations. Drawing by author.
Fig.4.0.b	247	Setup and conditions for test 1. Drawing by author.
Fig.4.0.c	248	Agent walking in corridor from entrance. Screen capture by author.
Fig.4.0.d	248	Screen captures at time intervals during the test of one agent. Screen capture by author.
Fig.4.0.e	249	Travel times for a sample of 50 agents is consistently 40 seconds. Graph by author.
Fig.4.0.f	251	Setup and conditions for test 2. Drawing by author.
Fig.4.0.g	252	Agents in the starting area. Screen capture by author.
Fig.4.0.h	252	Agents walking around the corner. Screen capture by author.

Fig.4.0.i	253	Screen captures at time intervals during the test. Screen capture by author.
Fig.4.0.j	255	Setup and conditions for test 3. Drawing by author.
Fig.4.0.k	256	Agents walking through opening. Screen capture by author.
Fig.4.0.l	256	Agents clumping together causes spikes in flow rate. Screen capture by author.
Fig.4.0.m	257	Screen captures at time intervals during the test. Screen capture by author.
Fig.4.0.n	258	Max flow rate of 1.2 p/s, below 1.33 p/s (redline), but is not maintained. Drawing by author.
Fig.4.0.o	258	Max flow rate spiked to 1.5 p/s, above redline, despite having same conditions as trial 11. Drawing by author.
Fig.4.0.p	260	Setup and conditions for test 4. Drawing by author.
Fig.4.0.q	261	Agent walking to the exit portal. Screen capture by author.
Fig.4.0.r	261	Screen captures at time intervals during the test of one agent. Screen capture by author.
Fig.4.0.s	262	Walking speed follow a uniform distribution, with an average of 1.28 m/s. Graph by author.
Fig.4.1.a	265	Setup and conditions for the wayfinding test. Drawing by author.
Fig.4.1.b	266	Agent's view at the T-junction cannot see where their gate is, and they only have the sign to inform their decisions. Screen capture by author.
Fig.4.1.c	268	Agent with direct navigation goes to the left. Screen capture by author.
Fig.4.1.d	268	Agent with perception navigation follows the sign for Gate B to the right. Screen capture by author.
Fig.4.1.e	269	A* Direct Wayfinding. Drawing by author.

Fig.4.1.f	270	A* Perception Wayfinding. Drawing by author.
Fig.4.1.g	271	Comparing the distance agents travelled to Gate B using direct and perception navigation. Graph by author.
Fig.4.1.h	273	Setup and conditions for the visibility test. Drawing by author.
Fig.4.1.i	275	Agent's view when they see the Narrow Gate. Screen capture by author.
Fig.4.1.j	275	Agent's view when they see the Wide Gate. Screen capture by author.
Fig.4.1.k	276	Screen captures as agent walks to Narrow Gate, showing the change in FOV area. Drawing by author.
Fig.4.1.l	277	Screen captures as agent walks to Wide Gate, showing the change in FOV area. Drawing by author.
Fig.4.1.m	279	Change in Field of View (FOV) Area. Graph by author.
Fig.4.1.n	279	Change in Field of View (FOV) Ratio. Graph by author.
Fig.4.1.o	279	Change in Average Visibility. Graph by author.
Fig.4.1.p	280	Maximum Field of View Area Distribution. Graph by author.
Fig.4.1.q	280	Field of View (FOV) Ratio Distribution When Gate Discovered. Graph by author.
Fig.4.1.r	280	Average Visibility Distribution. Graph by author.
Fig.4.1.s	282	Setup and conditions for the non-processing priority test. Drawing by author.
Fig.4.1.t	284	Agents with low food priorities waiting in the gate seating area (red). Drawing by author.
Fig.4.1.u	284	Agents with high food priorities getting food at the cafe. Drawing by author.

Fig.4.1.v	285	Agent Prioritizing a Non-Processing Domain Drawing by author.
Fig.4.1.w	286	Comparing which agents got food with each agent's priority for food availability. Graph by author.
Fig.4.2.a	289	Setup and conditions for the terminal tests. Drawing by author.
Fig.4.2.b	291	Floor plan of the Centre Security Layout. Drawing by author.
Fig.4.2.c	293	Passengers in the check-in area Screen capture by author.
Fig.4.2.d	293	Passengers in security screening. Screen capture by author.
Fig.4.2.e	294	Passenger entering holdroom concourse before the sign. Screen capture by author.
Fig.4.2.f	294	Passengers linger in holdroom concourse. Screen capture by author.
Fig.4.2.g	296	Centre Security Layout. Drawing by author.
Fig.4.2.h	297	Centre Layout Value Distribution. Graph by author.
Fig.4.2.i	297	Centre Security Screening Scores. Graph by author.
Fig.4.2.j	398	Floor plan of the Asymmetrical Security Layout. Drawing by author.
Fig.4.2.k	300	Passengers wandering (light blue) between check-in isles because they do not see the security area from left side of the check-in processor. Screen capture by author.
Fig.4.2.l	300	Security screening has a bias for passengers checking in on the right side. Screen capture by author.
Fig.4.2.m	302	Asymmetrical Layout Value Distribution. Graph by author.
Fig.4.2.n	302	Centre vs. Asymmetric Comparison. Graph by author.

Fig.4.2.o	303	Asymmetrical Security Screening Scores. Graph by author.
Fig.4.2.p	303	Centre vs. Asymmetrical Security Comparison. Graph by author.
Fig.4.2.q	304	Floor plan of the Perpendicular Security Layout. Drawing by author.
Fig.4.2.r	306	Passengers in the right isle of check-in looking for the security area. Screen capture by author.
Fig.4.2.s	306	Passenger's view walking along the wall from the right cannot see any identifying feature for security at the threshold. Screen capture by author.
Fig.4.2.t	307	Passengers approaching from the right side (purple) recognize the security queue sooner than passengers approaching from the left side (light blue), due to the narrow opening. Screen capture by author.
Fig.4.2.u	307	Perpendicular security screening area, with an exit to the left towards the wayfinding sign. Screen capture by author.
Fig.4.2.v	309	Perpendicular Layout Value Distribution. Graphs by author.
Fig.4.2.w	309	Architectural Value Comparison. Graphs by author.
Fig.4.2.x	310	Perpendicular Security Screening Scores. Graphs by author.
Fig.4.2.y	310	All Security Comparison. Graphs by author.
Fig.4.2.z	311	Assigned agent characteristics and random priority matrix. Drawing by author.
Fig.4.2.za	312	Assigned agent characteristics and high security priority matrix. Drawing by author.
Fig.4.2.zb	313	Assigned agent characteristics and equal priority matrix. Drawing by author.
Fig.4.2.zc	316	Average architectural value for all nine tests. Table by author.

Fig.4.2.zd	315	Centre Layout Range. Graph by author.
Fig.4.2.ze	315	Asymmetrical Layout Range. Graph by author.
Fig.4.2.zf	315	Perpendicular Layout Range. Graph by author.
Fig.4.2.zg	316	Random Priority Effects. Graph by author.
Fig.4.2.zh	316	High Security Priority Effects. Graph by author.
Fig.4.2.zi	316	Equal Priority Effects. Graph by author.
Fig.4.3.a	318	Skytrax Airport Ranking 2020. Drawing by author.
Fig.4.3.b	319	AirHelp Airport Ranking 2019. Drawing by author.
Fig.4.3.c	321	Setup and conditions for the airport tests. Drawing by author.
Fig.4.3.d	322	Simulated floor plan for Changi terminal 1. Drawing by author.
Fig.4.3.e	323	Simulated floor plan for Pearson terminal 1, international departure. Drawing by author.
Fig.4.3.f	325	Passengers in Changi going through the check-in area. Screen capture by author.
Fig.4.3.g	325	Passengers in Changi going through security into the retail courtyard. Screen capture by author.
Fig.4.3.h	326	Passengers in Pearson going through the check-in area. Screen capture by author.
Fig.4.3.i	326	Passengers in Pearson going through security. Screen capture by author.
Fig.4.3.j	328	Changi Population Distribution. Graph by author.

Fig.4.3.k	328	Pearson Population Distribution. Graph by author.
Fig.4.3.l	329	Sample Mean Comparison of Changi and Pearson. Graph by author.
Fig.4.3.m	329	Equivalent normal distributions for Changi (blue), $N(0.802, 0.0166)$, and Pearson (red), $N(0.420, 0.0289)$, as continuous PDFs, given an infinite number of samples. Graph by author.
Fig.4.3.n	331	Average values for each passenger priorities. Chart by author.

Part 0:

Introduction

This introduction describes the thesis's motivation, problems, and intentions. The hypothesis states that, the differences in an architectural layout for an airport terminal can be explained using an agent simulation, if agent decision-making relies on the perception of the surrounding environment. Part 0 also summarizes the expected results and the organization of the thesis structure.

0.0 Motivation

The motivation for this thesis comes from two ideas. Firstly, the thesis investigates how mathematics plays a role in the architectural design process, to model patterns and determine quantifiable outcomes. Design is a creative process, which requires using skills and intuition to develop new ideas. But once these ideas need to translate into the built environment, it can be challenging to demonstrate why one layout is better than another. The interest of using mathematics is to make logical choices during the design process. If a designer makes two different floor plans that are created for the same purpose, is there a way to quantify the differences between the two layouts to determine which design is better than the other (Fig.0.0.a)? Overall, the motivation of this thesis is to propose a world in which architectural decisions are based on mathematical analysis.

Secondly, the thesis is interested in developing a tool that allows designers to quantify differences between architectural spaces. This involves investigating existing research from civil engineering and computer science, which has developed ways of quantifying patterns of human activity in the built environment. These areas of research have demonstrated that human behaviour becomes predictable on a large scale, despite every person acting as an individual. The results from this way of thinking have impacted the way transportation systems are designed, which incorporates tools like crowd and traffic simulations to understand the movement of large volumes of people over time. Could the methods that work for crowd simulations in transport facilities be a starting point for quantifying architectural spaces?

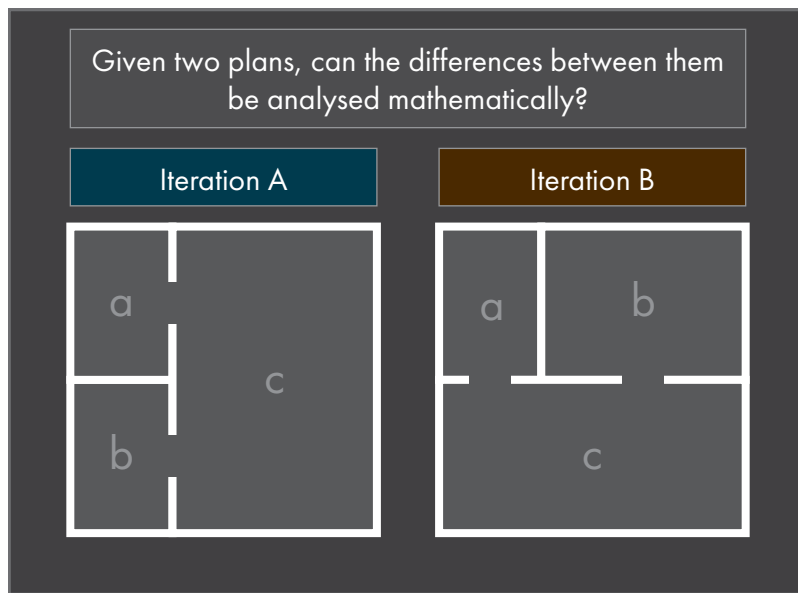


Figure 0.0.a: Comparing two different iteration of a floor plan.

0.1 Problem

Airports are complex facilities that are expensive to build. They need to accommodate over tens of thousands of passengers every hour, the movement of logistics, and control of security. It can be difficult to test a terminal building for that many people and factors. Additionally, it is not practical for designers to build a complete airport to see how it works or fix changes after finding design issues during operations. This complexity can result in architects relying on their intuition to judge the benefits from a design without formal testing or proper validation. Designers try to minimize these issues by using crowd simulations to approximate human behaviour. However, existing crowd simulations typically model passengers as a process flow, or people in emergency evacuations. As a result, these tools do not represent how people use architecture for decision-making for daily interactions. People are not aware of their final destination when they enter a terminal building for the first time. They may not understand what they need to do or how to get there. Instead, people rely on using their surroundings (the architecture) to inform them what to do. However, there are currently no methods to validate a building for this type of decision making. In summary, the thesis looks to address the following problems:

1. The complexity of human behaviour creates uncertainty in design decisions, which causes architects to rely on their intuition without proper validation.
2. Existing crowd simulations do not represent how people interact with architecture for decision making during daily airport operations.

0.2 Goals

The goal of this thesis is to develop a mathematical model to validate an airport terminal layout based on passenger interactions. The model quantifies a spatial arrangement as a function of behaviour of people in that space. It approximates human behaviour based on statistics from existing crowd simulations. It incorporates perception to provide feedback of how agents make decisions. Architecture is quantified using spatial analysis, and it is valued using the method of prioritization within existing airport domains. The thesis proposes a new way of scoring architecture by combining these methods into a single index of architectural value. The success of a floor plan is then dependant on a passenger's ability to fulfill their priorities within a given space. Therefore, to reach this goal, agents must be able to simulate these interactions within an architectural environment. Specifically, the thesis looks to understand the following questions:

1. What are the minimum architectural elements that have an influence on people in a given space?
2. What are the minimum mathematical models needed to quantify an architectural layout as a function of user activity?

0.3 Hypothesis

Consider a simulation of an airport terminal, which contains numerous agents that represent typical passengers. Imagine these agents are given a certain task to complete within an airport domain, which they can only accomplish by relying on the information from their perceivable surroundings. If these agents are given sufficient perception of the architectural environment, then the simulation can quantify how well the terminal's layout influenced passengers' decision making. As a result, this simulation could approximate an equivalent passenger interaction in a real airport environment, based on a statistical probability.

0.4 Expected Results

The thesis attempts to create an agent-based model that can calculate architectural value using agent perception. Firstly, the thesis should identify what aspects of architectural quantification are missing from the design process. Secondly, it should make clear how existing mathematical techniques and simulation tools are already capable of quantifying architectural conditions. Finally, the model should demonstrate a convincing mathematical approach for quantifying architectural spatial conditions.

The agent-based simulation should work as a practical tool that can illustrate basic functions of an airport terminal. In general, it should be able to differentiate two architectural layouts based on their accumulated architectural values. It should show what aspects of a space or airport domain create good or bad passenger interactions. Ideally, it should analyse a given airport and indicate the likelihood of the architecture being successful or a failure within a statistical certainty.

0.5 Thesis Structure

This thesis is organized into five parts, excluding the introduction. Part 1 talks about the context behind the thesis. Chapter 1.0 talks about architectural intuition and its inconsistencies. Chapter 1.1 introduces the basics of airport terminals, including the scope of design for an architect. Chapter 1.2 gives a summary of verification and validation. This includes typical applications in

other industries and its limitations. Chapter 1.3 gives a brief summary of probability and statistics. It shows how statistics approximates real-world patterns and describes the math behind typical probability distributions. Chapter 1.4 introduces simulation modelling, definitions of properties, processes, and different applications for airports. Additionally, chapter 1.4 concludes with a comparison of existing simulations.

Part 2 covers modelling concepts that the thesis identifies to be beneficial for creating an architectural agent simulation. Chapter 2.0 begins by defining what agents are, and their decision-making process. Chapter 2.1 introduces human perception. This talks about categorization, agent knowledge, and visual fields of view. Chapter 2.2 summarizes the concept of spatial analysis. This includes existing techniques and a mathematical description of graph theory. Chapter 2.3 introduces value theory, the method of prioritization, and common airport domains. This concludes with the thesis's proposal of how to calculate architectural value.

Part 3 goes into detail about how the thesis creates its agent simulation. Chapter 3.0 introduces Unity software, which is the program the thesis uses to build the simulation. It also gives a brief description of how its scripting components work. Chapter 3.1 summarizes script classes for the agent's functions, perception, and decision-making. Chapter 3.2 explains how the simulated environment and navigation work. Chapter 3.3 talks about components for airport architecture, scheduling, and value functions. Chapter 3.4 summarizes additional script classes for background functions. Chapter 3.5 concludes by listing the assumptions the thesis made for the agent model, and what the limitations are for the simulation.

Part 4 goes through all the simulation testing. Chapter 4.0 conducts standard tests which follow current simulation practices for verification and validation. Chapter 4.1 illustrates the range of behaviour based on the new components introduced for this thesis. Chapter 4.2 goes through basic airport terminal layouts to illustrate how the agent-model behaves in different floor plans. Chapter 4.3 compares two existing airport terminals to see if architectural value matches a real-world passenger survey ranking.

Part 5 discusses the results of the thesis, its impacts to architecture, and plans for future research. Chapter 5.0 begins by summarizing the results of the simulation testing. Chapter 5.1 talks about the components for an ideal architectural agent simulation. Chapter 5.2 walks through the impacts of simulation testing for the architectural industry. Chapter 5.3 discusses future research topics that could expand from this thesis's research. Finally, Chapter 5.4 summarizes the overall conclusions of the thesis.

Part 1:

Context

Part 1 begins by describing some of the limitations of intuition for architectural design. Chapter 1.1 introduces the complexity of current airport terminals and goes through the contemporary design process. Chapter 1.2 explains about the methods of verification and validation that are typical in other scientific disciplines. It also compares validation to existing practices in architectural design. Chapter 1.3 gives a brief summary of probability and statistics. It describes how statistics uses mathematical models to approximate real-world patterns. Chapter 1.4 introduces different types of simulation models and how they work in different applications. This chapter also describes some existing simulation methods for analyzing airports. It concludes by walking through some of the tools the thesis has considered for an architectural agent simulation.

Chapter 1.0

Architectural Intuition

Architecture is a discipline that incorporates both technical knowledge and artistic sense in the pursuit of creating physical spaces in the built environment. Architects make decisions about how to organize building elements based on their knowledge as a professional. They must make sure that a design meets project requirements set out by clients and developers for the occupants and people involved. Currently, contemporary architectural practice is moving towards using technical analysis to validate certain aspects of a building. Up to now, building validation has an influence on codes, structures, services, construction, and energy usage. However, it is still common that space planning is dependant on the interpretation of the designer. Architects decide if a layout of spaces is functionally validated primarily based on their professional knowledge and experience.

While space planning is an important responsibility of an architect, there is no direct approach to scientifically or mathematically validate if a given arrangement of spaces meet the needs of a project. The layout of spaces is fundamental to the function of a building. All other building elements, like material, structure, and services develop around the framework of space. The risk of unvalidated architecture is that designers who pursue unconventional layouts, or aesthetically unique designs, claim a building is functional for the occupants without objectively checking if that is true or not. It is not wrong for architects to make unconventional designs, or to experiment with the aesthetics of space. However, space planning tends to move from modelling into contract documentation as the primary evidence of design validation, which is a limiting approach to prove how a building operates scientifically.

Architectural spaces are difficult to define because they can have soft outlines that are not limited to conventional boundaries, like social effects. ^[1] A single space may be associated with multiple functions, like multi-use spaces in apartments, or have connections that are not physically related, like sacred spaces in religious architecture. As a result, architects can rely on

1. Arabacioglu, Burcin Cem. "Using Fuzzy Inference System for Architectural Space Analysis." *Applied Soft Computing* 10, no. 3 (2010): 926–37. <https://doi.org/10.1016/j.asoc.2009.10.011>. 926.

their intuition as the primary factor for deciding these types of layouts, since their experience and interaction with these spaces informs how they will be used. [2]

In their research on architectural design ideas and beliefs, Holm states that architects tend to make decisions based on their own experience, skills, and values. [3] This results in information that is based in *personal knowledge*, unlike *shared knowledge* in scientific disciplines. [4] Personal knowledge is difficult to disprove with objective evidence because one architect's knowledge might not share the same fundamental ideas as another architect, which is neither right nor wrong. [5] Holm mentions that this inconsistency shows up in the architectural language, where concepts like space or form do not have clear definitions, in which space can refer to both physical and social boundaries. In addition, it is common for architectural work to start from "scratch", or the need to be inventive for every new project, which can disregard existing practices or proven results. [6] As a result, designers usually try to solve unconventional situations or problems that were not there initially. Holm summarizes that these conditions make architecture value-based rather than evidence or fact-based like other academic disciplines. [7]

Wide Range of Airport Designs

Early examples of prominent airport architecture illustrate how a designer's ideas can be inconsistent. In the 1960's, architect Eero Saarinen created two different terminals that had contrasting design methodologies, which were examples of design flexibility and inflexibility. [8] Both terminals were designed as international airports for two major cities on the east coast of the United States, less than 400 km apart, built within 2 years of each other. With one architect overseeing both designs, one would expect these terminals are built around a common framework. However, it appears that these two terminals are arranged quite differently.

2. Arabacioglu, "Fuzzy Inference System". 926.

3. Holm, Ivar. "Ideas and Beliefs in Architecture and Industrial Design". (PhD thesis, Oslo School of Architecture and Design, 2006). 282.

4. Holm "Ideas and Beliefs". 282.

5. Holm "Ideas and Beliefs". 282.

6. Holm "Ideas and Beliefs". 284.

7. Holm "Ideas and Beliefs". 284.

8. National Academies of Sciences, Engineering, and Medicine. "Airport Passenger Terminal Planning and Design, Volume 1: Guidebook". *Washington, DC: The National Academies Press, (2010): 6-7.*

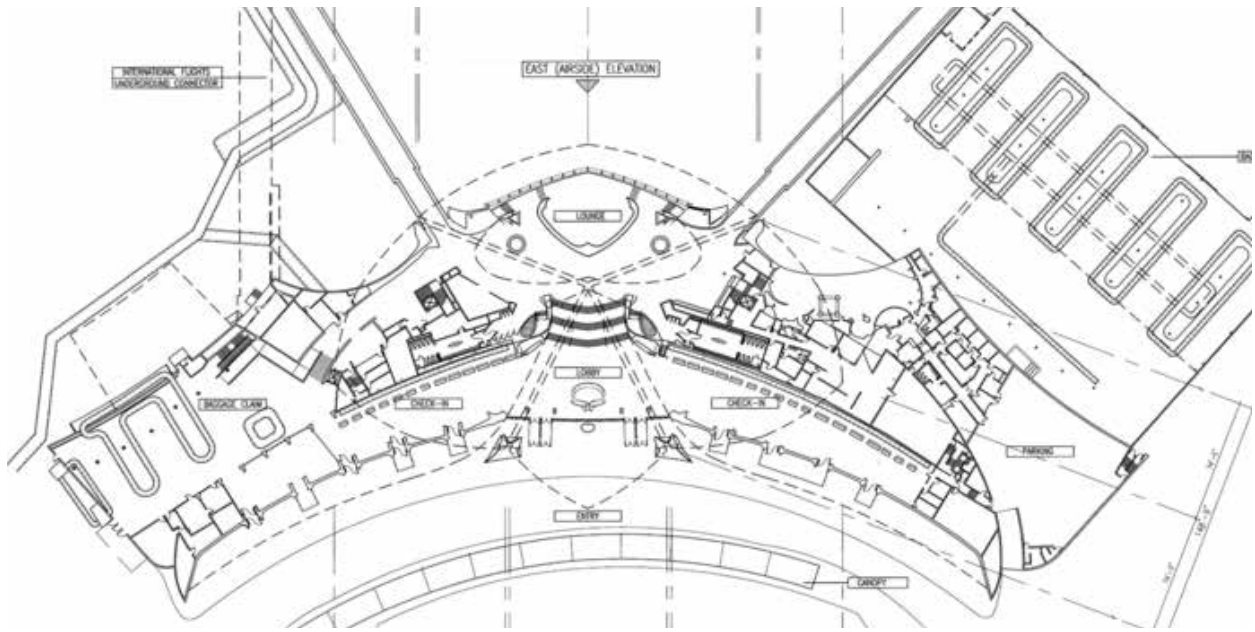


Figure 1.0.a: Floor plan of Eero Saarinen's TWA Flight Centre (1961) has a curved concourse with a centralized lobby, from the National Park Services, as shown by Fiederer (2016).

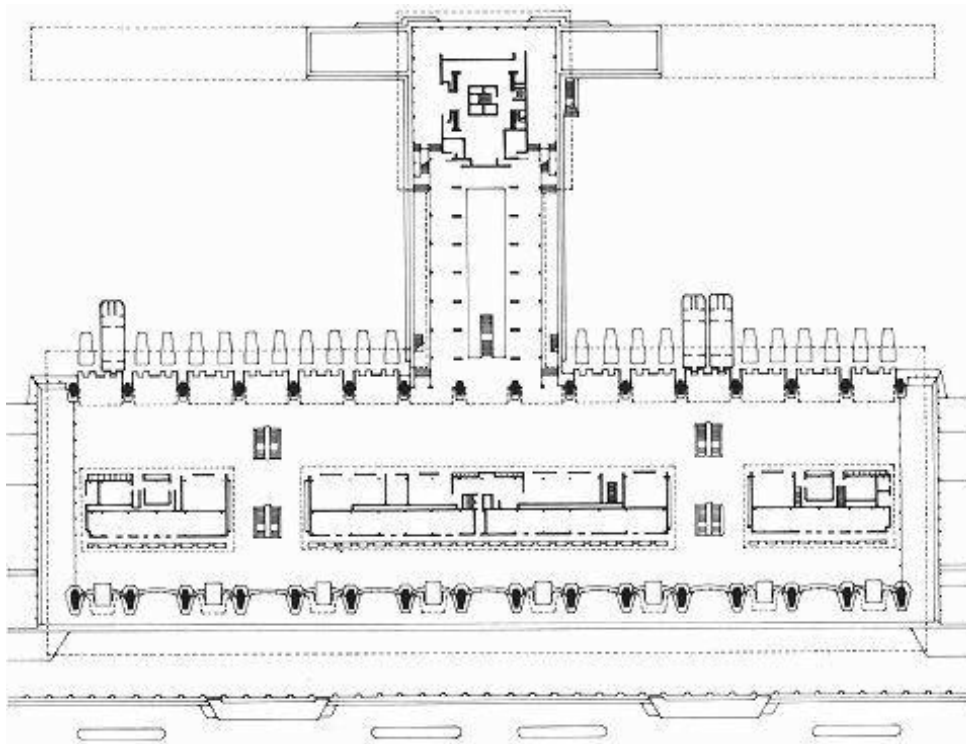


Figure 1.0.b: Floor plan of Eero Saarinen's Dulles International Airport main terminal building (1962) has a linear concourse as a transition to mobile lounges, as shown by Futagawa (1973).

Saarinen's first terminal in 1961 was the TWA Flight Center at John F. Kennedy International Airport (JFK), outside New York City. The design of the terminal differs from the orthogonal geometry of the international style at the time. ^[9] Instead, Saarinen went with curvilinear lines, developing a concrete shell structure that arched over the terminal like a bird's wings. As a work of art, the public saw the building as innovative, beautiful, and a creative design. ^[10] However, as a terminal building, it became "functionally deficient" over time due to the complexity of how the spaces were arranged and the rigid form of the concrete structure. ^[11] ^[12] New piers and concourses needed to be added to accommodate the growing size of planes, like the Boeing 747. Today, JFK has expanded significantly since the 1960's. The TWA building is no longer used as a terminal, and it is instead repurposed as a hotel. ^[13]

Saarinen's second design, completed in 1962, was the main terminal at Dulles International Airport (IAD), outside Washington D.C. He incorporated the building with a modular design which was able to adapt to terminal growth overtime. The concept included the use of "mobile lounges", a bus-like vehicle used to shuttle people between the planes and the main terminal building. ^[14] While initially useful, the maintenance cost of the lounges and the complexity of closing-out flights early to move passengers back to the terminal, resulted in abandoning the lounges altogether. ^[15] Since then, Dulles has converted its terminals to a satellite configuration with an underground people-mover system (automated train). ^[16]

Comparing the floor plans of these airports shows two different design languages. The floor plan of TWA is a single-storey, crescent-shaped building centered around a split-level lobby space (Fig.1.0.a). It has a central lower ticket lobby, with stairs that ascend into an upper lobby concourse on the gates side (Fig.1.0.c). Two piers were later added to this concourse, which extend out to the plane gates. A baggage claim concourse can be seen in the north wing of the terminal and a parking facility in the south wing.

9. National Park Services. "Trans World Airline Flight Center". National Register of Historic Places. September 7, 2005. Section 7: 1.

10. National Academies, "Airport Passenger Terminal Planning and Design". 7.

11. National Academies, "Airport Passenger Terminal Planning and Design". 7.

12. National Park Services. "Trans World Airline Flight Center". 8: 9-10.

13. McFadden, Robbyn. "Up, up and away at the TWA Hotel". CBS News. May 12, 2019. <https://www.cbsnews.com/news/up-up-and-away-at-the-twa-hotel-at-jfk/>.

14. National Academies, "Airport Passenger Terminal Planning and Design". 7.

15. National Academies, "Airport Passenger Terminal Planning and Design". 7.

16. National Academies, "Airport Passenger Terminal Planning and Design". 7.

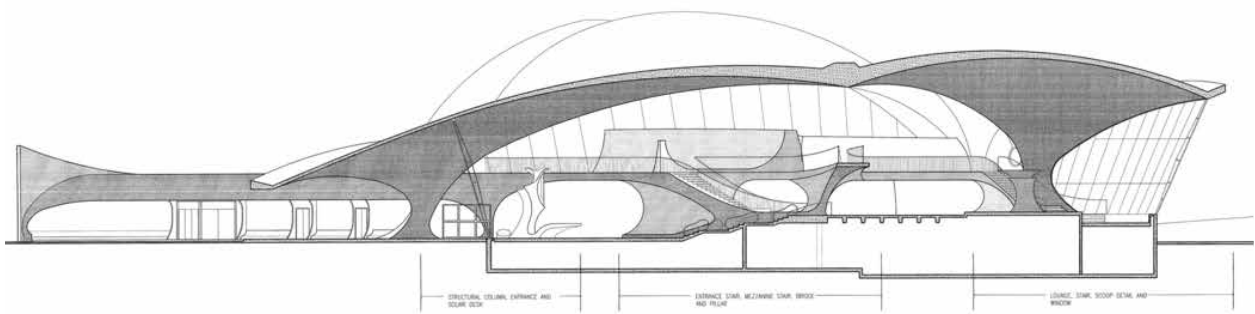


Figure 1.o.c: Section of TWA Flight Center with a split-level lobby and shell structure, from the National Park Services, as shown by Fiederer (2016).

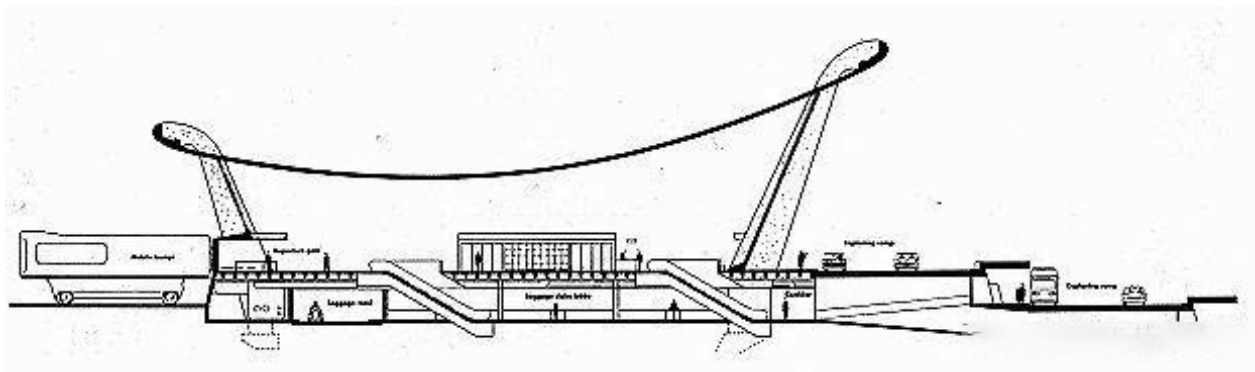


Figure 1.o.d: Section of Dulles terminal building with two levels and connection to the mobile lounges, as shown by Saarinen (1968).

The floor plan of Dulles is a multi-storey rectilinear space with a consistent structure spacing (Fig.1.o.b). The main level has two symmetrical halls connecting the check-in area and a linear holdroom concourse. The holdroom is lined with stalls for the mobile lounges which are parked perpendicularly to the building so passengers can walk straight into them. There is a T-junction in the middle of this concourse, which connects up to the based of a control tower. The baggage claim hall is also on a lower level below the main halls (Fig.1.o.d).

Although both terminals were made by the same architect during the same time, the plans illustrate that they have very different ways of organizing spaces. This does not mean one design is worse than the other; each terminal had to solve their own problems due to the nature of their layouts. Even though they are created under similar circumstances, an architect like Saarinen has a wide range of layout choices that might work. However, terminals still have a logical structure that is inherent with the functions of an airport, like check-in and gate locations. Deciding which layout best suits those functions is the challenge of architectural design.

Chapter 1.1

Airport Terminal Design

The thesis experiments with agent-based simulations in the context of airport terminal architecture. The focus on airports has many benefits for this thesis because of the existing research and available data. Contemporary airports involve many aspects of aviation, logistics, and human wellbeing. Today, understanding the complexity of these areas involve using some form of simulation or modelling to predict the impact to operations. ^[1] There is also well-established research and data that analyzes passenger experience within a terminal building. ^[2] Analysing architecture using agent-based simulations can build from these established practices. Terminals also have a clear purpose for processing passengers and transporting people to where they need to go. Fundamentally, this gives an agent simulation a well-defined goal that is easy to model in a digital environment. The function of passenger processing can also become a clear indicator of architectural performance.

The primary source the thesis uses to understand the airport design process is based on the reports written by the National Academies of Sciences, Engineering, and Medicine, as part of the Airport Cooperative Research Program (ACRP). They present guides for airport planners on the fundamentals of airport design and operations based on standards by the FAA (Federal Aviation Administration). ^[3] The ACRP procedures are considered standard practice in the United States. ^[4] The thesis takes these reports as representative of conditions for North American airports. The research also considers the methods described by the GTAA (Greater Toronto Airports Authority) in their master plan of Toronto Pearson International Airport. ^[5] ^[6]

1. National Academies of Sciences, Engineering, and Medicine. “Simulation Options for Airport Planning”. *Washington, DC: The National Academies Press*. (2019): 3.

2. Wiredja, Dedy, Vesna Popovic, and Alethea Blackler. “A Passenger-Centred Model in Assessing Airport Service Performance.” *Journal of Modelling in Management* 14, no. 2 (May 10, 2019): 492–520.

3. National Academies of Sciences, Engineering, and Medicine. “Airport Passenger Terminal Planning and Design, Volume 1: Guidebook”. *Washington, DC: The National Academies Press*, (2010): 1.

4. National Academies, “Airport Passenger Terminal Planning and Design”. 1.

5. GTAA. “Toronto Pearson International Airport Master Plan 2017-2037”, Greater Toronto Airports Authority, (2017). 5-13.

6. GTAA. “2018 Airport Construction Code, v5.0”, Toronto Pearson International Airport, (2018). xiii.

Planning Process

Airports are built up of many elements like civil infrastructure, maintenance facilities, terminal buildings, and servicing equipment. They can cost in the range of billions of dollars and require the order of magnitude of a million square metres of space. ^[7] It is rare for new international airports to be built from scratch, since it is more likely that a high-populated area is already served by a functioning international airport. Instead, it is more common for new airport developments to be renovations of existing facilities or expanding from existing infrastructure. ^[8] Although, constructing a new airport can occur if an existing airport is over capacity, there is open land available for a new facility, and the local authority is willing to budget the time and resources for a new project. In other words, it is rare for airport projects to occur in the first place, but when they do, it is important to make sure there will be use out of it.

For the terminal building alone, there are countless decisions concerning what the design scope of the facility needs to be, including capacity for the number of flights that the airport expects to handle, and the number of passengers expected to be on those flights. Architecturally, planners need to consider how areas are integrated with the existing terminals, what areas are controlled by security, and how far people need to walk. Additionally, the size of the facility itself is dependant on many factors, which includes the number of service counters, baggage carousels, queue lines, and gate seating.

New projects for major airports usually go through an *airport authority*. An airport authority is, typically, a not-for-profit, government funded, private company that manages airport operations. ^[9] Projects are managed by *airport planners*, who may refer to internal developers from an airport authority, or external consultants in engineering, architecture, or urban planning. ^[10]

In Volume 1 of *Airport Terminal Planning and Design*, the National Academies explains in detail that new terminals must go through several stages of planning before reaching the design

7. Neumann, Peter. “Kosten für Großflughafen steigen um 160 Millionen Euro, weil mehr Passagiere erwartet werden: Noch nicht gebaut und schon teurer”. Berliner-Zeitung, 2008-07-10. <https://www.berliner-zeitung.de/kosten-fuer-grossflughafen-steigen-um-160-millionen-euro-weil-mehr-passagiere-erwartet-werden-noch-nicht-gebaut-und-schon-teurer-li.6277>.

8. National Academies, “Airport Passenger Terminal Planning and Design”. 1.

9. PANYNJ. “Terminal Planning Guidelines”, The Port Authority of New York and New Jersey, August 2013, <https://www.fd.cvut.cz/projects/k621x1ml/dokumenty/panynj-terminal-planning-guidelines.pdf>.

10. Nelson Oliveira (Project Director, Greater Toronto Airports Authority), phone conversation with author, February 3, 2020.

process. ^[11] In the first stages of a new facility, planners identify the function for domestic or international processing, the expected number of flights, and the expected number of passengers. Significant airport projects are planned at least a decade in advance of any actual construction. Planning that far in advance makes it challenging to estimate how an airport will operate decades into the future. However, planners predict how future operations will work using statistics of current airport operations and past growth. During this stage, planners will also communicate with relevant airline companies who may be the primary clients of the new facility.

Terminal Building Layout

There are two main types of terminal concepts that planners can choose from, depending on the capacity of the airport. These are *centralized* or *decentralized* terminal buildings. ^[12] In a centralized terminal, all passengers and logistics are processed in one building. This maximizes the use of shared facilities and amenities, which avoids duplicating services. It also simplifies wayfinding for passengers, since there is only one area for arrivals and departures. ^[13] The downside of a centralized facility is passengers may need to walk long distances between flights, if they all need to pass through a single location. ^[14]

Decentralized facilities are beneficial for separating different types of flights, operations, or airlines. This includes providing security separation between domestic and international travellers. Each building only needs to serve the passenger demand for a given travel, which can distribute an airport's capacity during peak operations. ^[15] The main downside of a decentralized facility is that each building is independent. As a result, every terminal needs their own services and amenities, which can be expensive to maintain.

A basic airport terminal is divided into two areas: landside and airside, which are separated by a security line. Landside interfaces with public areas, whereas airside controls restricted access to the planes. Both areas have core spaces for passenger processing. This typically includes check-in, security, holdroom concourse, gates, immigration, and baggage claim. These areas align with passenger *flows*, which are departure, arrival, or connecting. Departure flow includes check-in, security, (sometimes immigration), holdroom, and gates. Arrival flow includes gates,

11. National Academies, "Airport Passenger Terminal Planning and Design". 23.

12. National Academies, "Airport Passenger Terminal Planning and Design". 171.

13. National Academies, "Airport Passenger Terminal Planning and Design". 171.

14. National Academies, "Airport Passenger Terminal Planning and Design". 172.

15. National Academies, "Airport Passenger Terminal Planning and Design". 172.

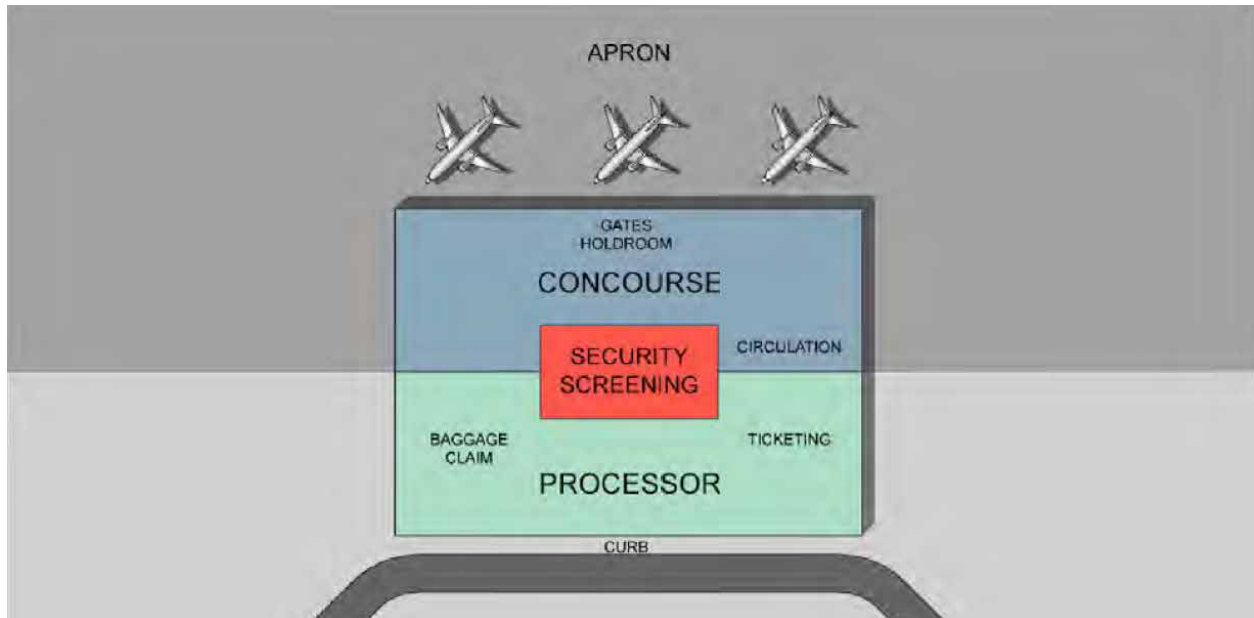


Figure 1.1.a: Basic airport terminal layout, as shown by National Academies (2010), sourced from “Considerations for Selecting a Terminal Configuration,” David A. Daileida, FAIA, FAA White Paper.

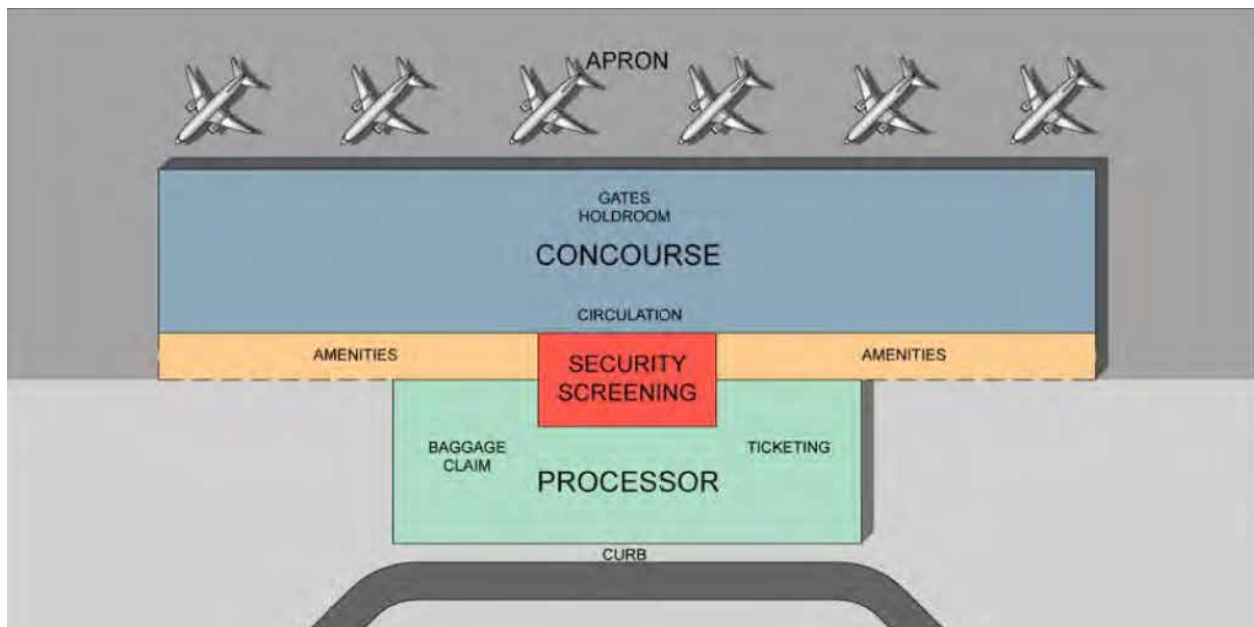


Figure 1.1.b: Linear terminal, as shown by National Academies (2010), sourced from “Considerations for Selecting a Terminal Configuration,” David A. Daileida, FAIA, FAA White Paper.

immigration, and baggage claim. Depending on the airport, connecting passengers may stay in the gate concourse. However, some connecting passengers may need to go through immigration before boarding their next flight.

A terminal can be designed in several different configurations. The most common configurations are *linear*, *pier*, and *satellite*.

Linear: A basic linear configuration has a single passenger processor that is accessed by a road, or curbside. The processor is connected directly to a gate concourse, which passes through a security screening area. The plane gates are then evenly spaced, side-by-side, along the gate concourse on an apron (Fig.1.1.a).^[16] Passengers can access the planes either, directly from the terminal building by a *jet bridge*, or remotely at-grade. For larger airports, these concourses can be elongated to accommodate more planes. This is typically served by a corridor behind the gate holdroom, which contains amenities (Fig.1.1.b).^[17]

Pier: A pier configuration has a similar processor and security area like a linear terminal. However, the gate concourse for a pier extends out perpendicularly, like a boat pier. This allows planes to be served on either side of the concourse, which can extend out further to accommodate more planes (Fig.1.1.c).^[18] To reduce passenger walking distance in larger airports, planners will include multiple piers (Fig.1.1.d).^[19] These piers are either organized parallel to each other, or radially around a centralized processor building. The spacing between the piers is determined by the size of the planes.

Satellite: A satellite configuration separates the gate concourse from the main processor building. The benefit of a satellite building is that planes can be parked around all sides of the terminal. Satellite buildings are always on airside, which contains a gate concourse, holdrooms, and amenities. This can be accessed, either above-grade or underground, by automated trains, shuttlebuses, or walkways (Fig.1.1.e).^[20] For larger airports, it is common to have multiple linear-satellite concourses, which are linked by an automated-people mover (AMP) (train) (Fig.1.1.f).^[21] This takes advantage of the efficiency of a linear configuration and the capacity of a satellite.

16. National Academies, "Airport Passenger Terminal Planning and Design". 173.

17. National Academies, "Airport Passenger Terminal Planning and Design". 174.

18. National Academies, "Airport Passenger Terminal Planning and Design". 175-176.

19. National Academies, "Airport Passenger Terminal Planning and Design". 177.

20. National Academies, "Airport Passenger Terminal Planning and Design". 178.

21. National Academies, "Airport Passenger Terminal Planning and Design". 182.

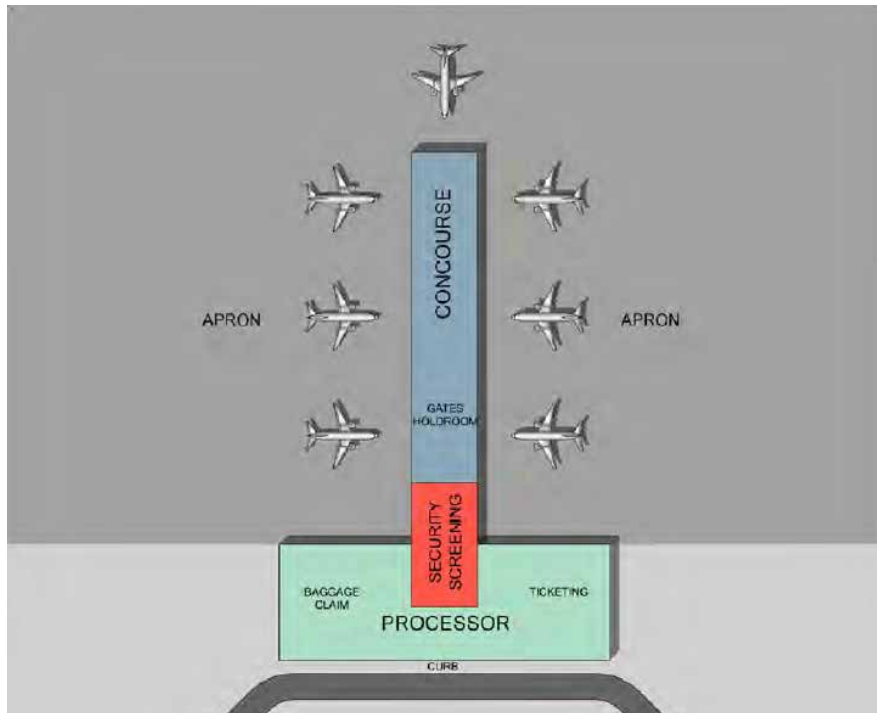


Figure 1.1.c: Pier terminal, as shown by National Academies (2010), sourced from “Considerations for Selecting a Terminal Configuration,” David A. Daileda, FAIA, FAA White Paper.

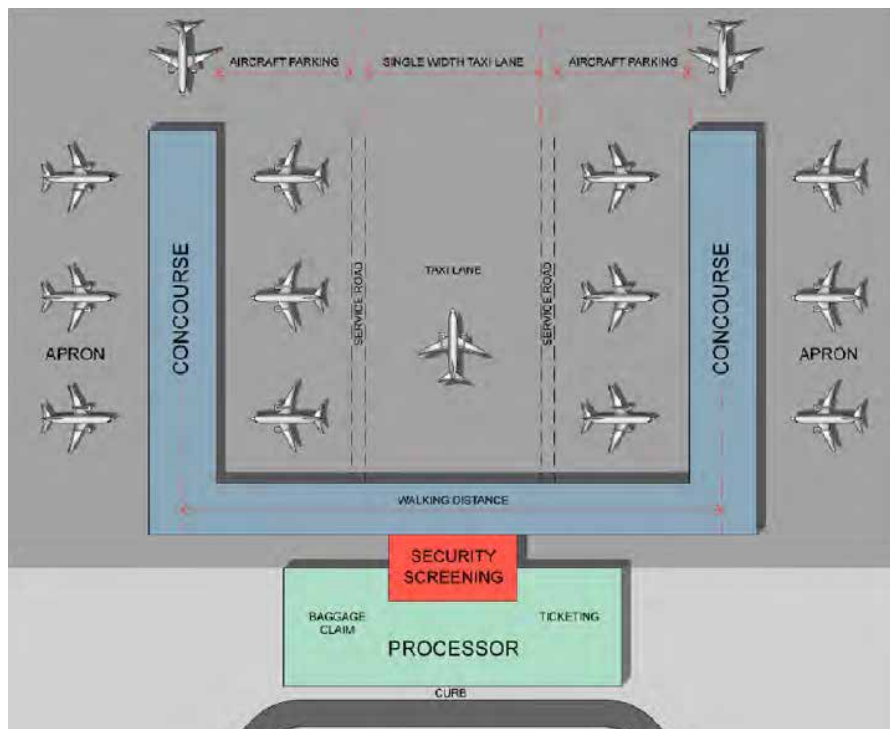


Figure 1.1.d: Multi-pier terminal, as shown by National Academies (2010), sourced from “Considerations for Selecting a Terminal Configuration,” David A. Daileda, FAIA, FAA White Paper.

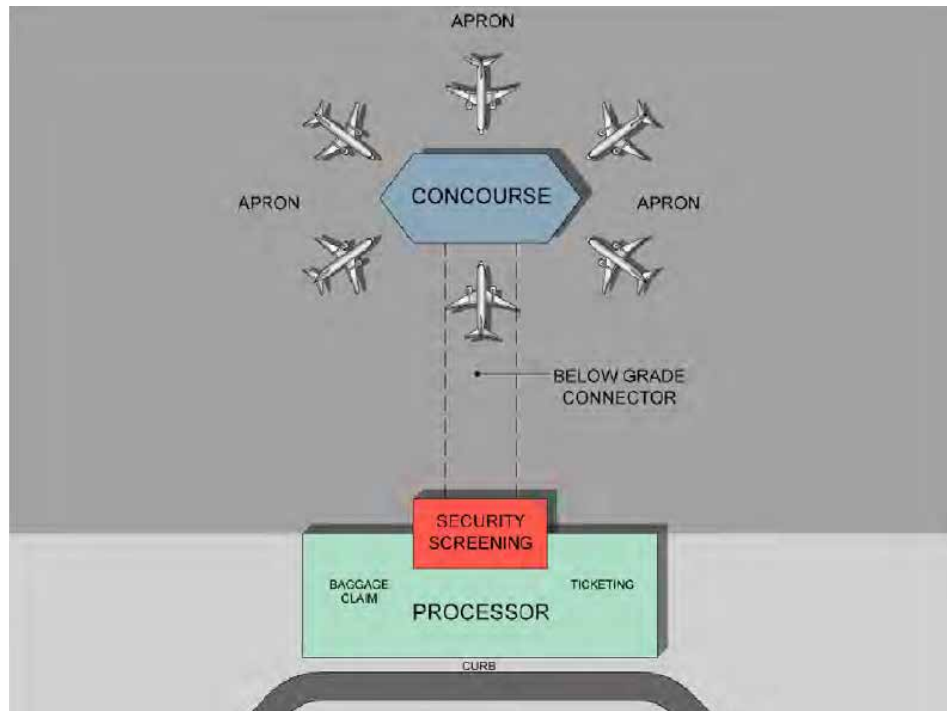


Figure 1.1.e: *Satellite terminal, as shown by National Academies (2010), sourced from “Considerations for Selecting a Terminal Configuration,” David A. Daileda, FAIA, FAA White Paper.*

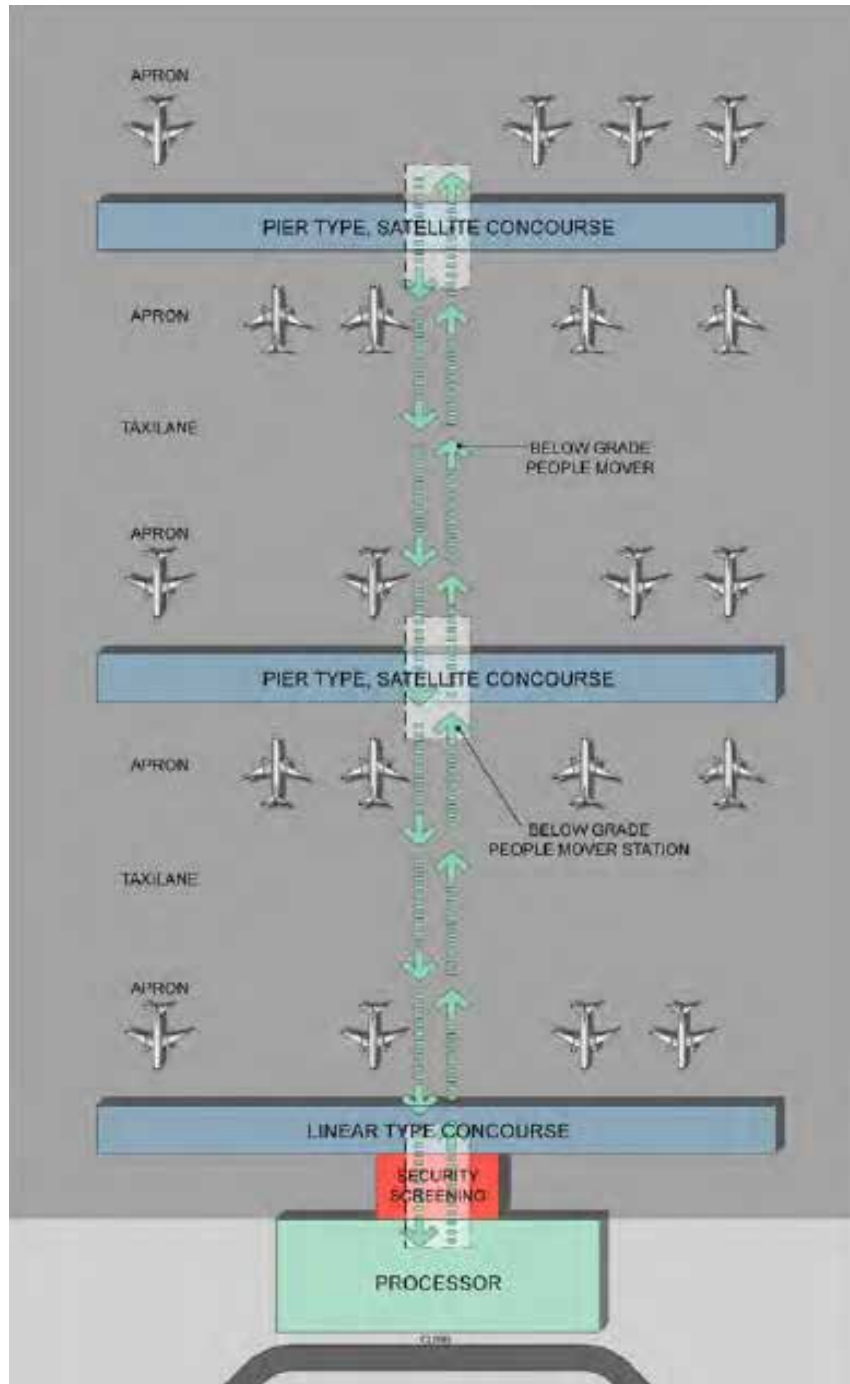


Figure 1.1.f: Satellite terminal with an Automated People Mover (APM) system, as shown by National Academies (2010), sourced from “Considerations for Selecting a Terminal Configuration,” David A. Daileida, FAIA, FAA White Paper.

Demand Forecasting

The size of an airport terminal is based on the future forecast of passengers, cargo, and *aircraft movements*.^[22] These movements refer to how many planes are expected arrive and depart over time. This indicates the number of passengers that *enplane* (depart) and *deplane* (arrive) from those flights. Planners get this information from *demand forecast* data. There are two common approaches for collecting forecast data. One option is to extrapolate past trends from the existing facility into the future. The second option is to use national forecast data and extrapolate based on the latest social and economic factors.^[23] Travel forecasts are broken down into daily and hourly passenger movements. The amount of airport activity fluctuates throughout the year. So, planners account for these changes by considering the number of passengers during the busiest time of the year. However, it is not practical to design airports to accommodate the greatest number of passengers, since that level of activity is not consistent all the time.

Peak Hour

The hour when an airport sees the greatest number of passenger movements is called the *peak hour*. North American airports are designed according to the peak hour of an average day during the busiest month.^[24] The peak hour may not correspond to a clock hour exactly, but instead it can be an interval of time when flights are expected to arrive and depart. Outside of the United States, some airports are designed by considering the 90th to 95th percentile of the busiest hour of the year. However, it is challenging for planners to know precisely what time of year is the busiest. The number of passengers during peak hour determines how large areas in a terminal need to be. It also influences the number of check-in counters, security screening machines, and length of queue lines.

For example, in 2017, Toronto Pearson Airport processed over 12 000 passengers during its peak hour, which was around 18:00 (6:00 PM), on average (Fig.1.1.g).^[25] Passengers are either coming from arriving, departing, or connecting flights. There were about 7 000 passenger movements departing from Pearson over the peak hour (Fig.1.1.h). As an example, planners designing a departure hall for Pearson would want to make sure it could process at least 7 000

22. National Academies, "Airport Passenger Terminal Planning and Design". 9.

23. National Academies, "Airport Passenger Terminal Planning and Design". 21.

24. National Academies, "Airport Passenger Terminal Planning and Design". 89.

25. GTAA. "Toronto Pearson International Airport Master Plan 2017-2037", Greater Toronto Airports Authority, (2017), 38, 56, 85.

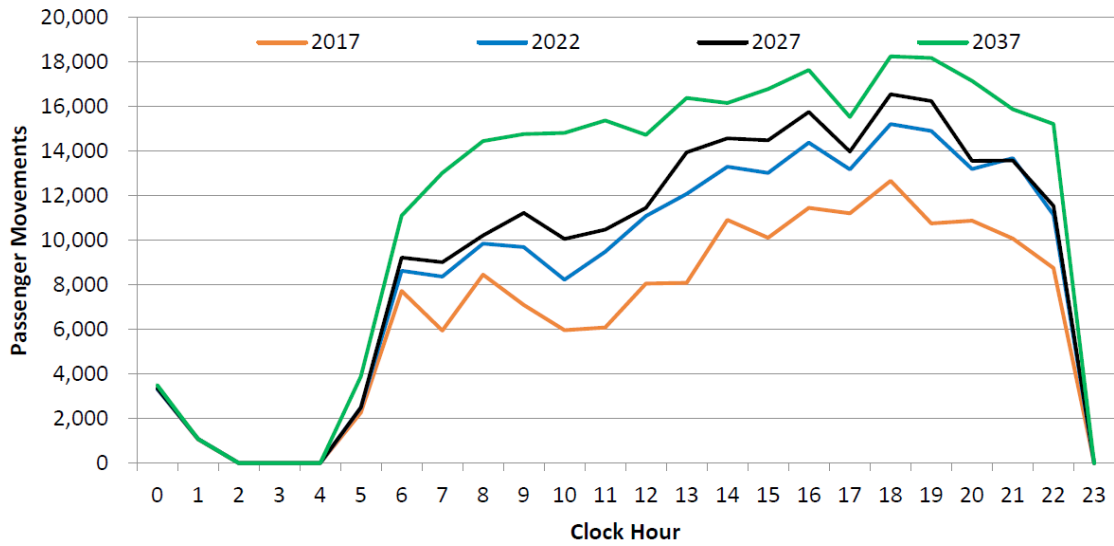


Figure 1.1.g: Toronto Pearson Airport's hourly passenger movement forecasts, GTAA (2017).

Peak-Hour Passenger Movement Forecasts						
	Enplaned/Deplaned		Origin/Destination		Connecting Passengers	
	Arr	Dep	Arr	Dep	Arr	Dep
2017	7,500	6,800	4,800	4,500	2,800	3,300
2022	9,400	8,600	6,400	5,700	3,500	3,700
2027	9,900	9,400	6,600	5,900	3,700	4,200
2037	11,600	11,700	7,700	6,400	4,200	5,600

Figure 1.1.h: Toronto Pearson Airport's peak-hour passenger movement forecasts base on existing and projected flight schedules, GTAA (2017).

passengers during that hour. The terminal design should function normally without having congestion in corridors or queues that overflow into other areas.

Level of Service (LOS)

The required area for the terminal depends on the number of passengers and how many people can comfortably fit in each area. In North America, airport planners use Level of Service (LOS) factors to find the density of people acceptable for public spaces. LOS factors were developed in 1970 by John Fruin based on experiments conducted with crowds. ^[26] It describes how freely people can walk in open paths, stairways, and queues on a scale from A to F (Fig.1.1.i). Level A is when people have free space to walk without any obstructions. Level F is when people are practically squished together and can barely move. To be economical with space and time, most airports are built for a LOS factor of C, which is at least 15sqft (1.4m²) per person. Level C means people can walk uninterrupted in any direction, but there is possibility that people need to adjust their walking speed to avoid obstructions. This provides a balance between the size of a space and passenger comfort levels. Planners may aim for a higher LOS like Level B during conceptual design, knowing over time it tends to drop to Level C for final operations. ^[27] The final size of the final terminal building is dependant on the required square-footage and passenger demand.

Conceptual Planning

Once the number of *aircraft movements* and passengers are established, the next stage involves creating a conceptual site plan. The site plan shows how the new terminal fits together with the entire airport. Planners will explore different arrangements of the building based on the number of gates, overall circulation, and expected construction costs. The best options developed from *conceptual planning* are then further refined with more detail. Areas in the terminal are measured out in CAD as simple geometry based on the space program. At this stage, planners turn to spreadsheet models and simulations to verify demand forecasting data. ^[28] It is common for planners to consult with engineering and architectural teams, who check the terminal using

26. Fruin, John J. *Designing for Pedestrians: A Level of Service Concept*. New York, 1970.

27. Nelson Oliveira (Project Director, Greater Toronto Airports Authority), phone conversation with author, February 3, 2020.

28. National Academies, "Airport Passenger Terminal Planning and Design". 28.

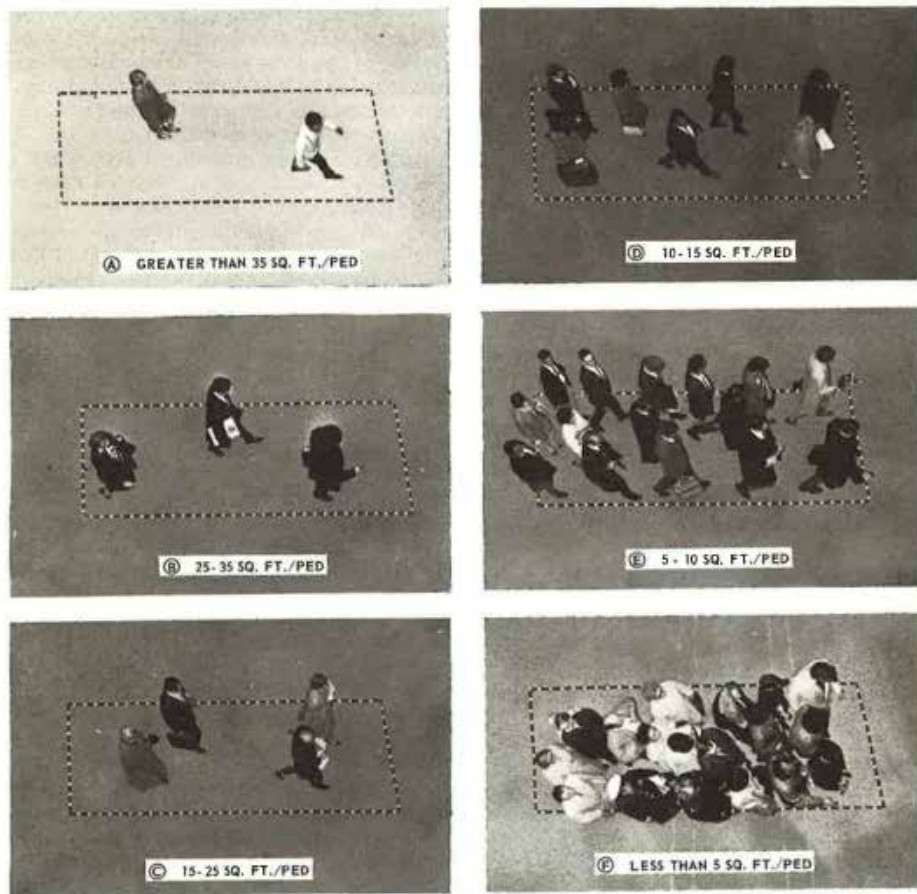


Figure 1.1.i: Experiment showing the level of service (LOS) from A, the least dense, to F, the most crowded, Furin (1970).

traffic simulations and crowd modelling. ^[29] ^[30] This makes sure spaces are acceptable for passenger flow and LOS factors.

Design Process

Once planners confirm a conceptual plan, it is then brought to the architects to begin the design process of the physical building. It is common for architectural design consultants, who were not part of the initial planning stages, to question the form of the building and try to impose new design ideas. ^[31] The focus of airport planners is to maintain the functional integrity of the project. There are critical areas (hard points) in the airport that cannot be changed. This includes the location of airside (aircraft stands, gates, and apron), and landside (curbside and roadways). The location of the security line between secure areas and non-secure areas may also be fixed. The responsibility for architects is to arrange, detail, and design elements between these hard points. This includes arranging the layout for check-in and security, designing the experience for retail and restaurants, and detailing waiting areas and back-of-house staff rooms.

As is common across all architectural projects, this leads into schematic design, design development, and contract documentation. ^[32] It is through schematic design that the conceptual plan is developed into a terminal building. Architects translate the areas of space program into floor plans, verifying that terminal works with simple architectural elements like volume and structure. In design development, the terminal is refined in more detail with form, structure, and building systems. The terminal is also verified at this stage for building codes and safety standards. From this point, only minor changes to the design are made. Then drawings are prepared for construction, which leads into the final contract documentation.

The thesis considers the transition from conceptual planning into design development. Agent-based simulations can help verify that architectural space planning aligns with the functional integrity of the rest of the airport. Using the knowledge established in the early stages of planning, agent modelling can verify architectural conditions align with passenger interactions and decision-making, based on the perspective of the individual.

29. National Academies, “Airport Passenger Terminal Planning and Design”. 28.

30. Nelson Oliveira (Project Director, Greater Toronto Airports Authority), phone conversation with author, February 3, 2020.

31. National Academies, “Airport Passenger Terminal Planning and Design”. 30.

32. National Academies, “Airport Passenger Terminal Planning and Design”. 30.

Summary

The context of this thesis looks at the design of airport terminals. There is established research showing that analysis tools, like simulations, are required to understand complex systems and predict the impact to operations. The design process involves several stages of planning and analysis from airport planners before starting the architectural design of the terminal building. This involves identifying the capacity of the facility and the expected number of flights and passengers, using demand forecasting. Airport terminals are divided into two areas based on security: landside and airside. The passenger-facing areas of a terminal building include several core spaces like check-in concourse, security, holdroom concourse, gates, immigration, and baggage claim hall. The size of an airport terminal is based on the expected number of aircraft movements, during the peak-hour of operations. Aircraft movements influence the expected number of passengers, which then decides how big areas inside the building need to be. The exact square-footage is calculated based on the density of passengers using level-of-service factors. This is verified using crowd modelling and traffic simulations to check for restrictions. A balance is chosen between the economics of available space and expected processing time. After planners put together a suitable site plan, architects begin designing and arranging spaces for the physical building. However, architects are required to design the terminal within the hard points established during planning, like the gates and security line. The scope of the thesis analyzes the arrangement of these spaces using agent-modelling to verify architectural performance, to be consistent with the planning stages.

Chapter 1.2

Verification and Validation

A major focus of this thesis is to increase the credibility and confidence of architectural spatial decisions. Without proper analysis, there is no certainty in the function of a layout, or the accuracy of a designer's choices. ^[1] Verification and validation (V&V) describe an objective process that checks if a product, service, or system meets the requirements of its intended purpose. ^[2] This thesis considers V&V for two conditions: the design process and the agent-based model. Firstly, existing practices of V&V can be used to check if an architectural layout is meeting the scope of a design project. Secondly, V&V can confirm if components in an agent-based model are working correctly, based on simulation standards. Ideally, verification and validation are not meant to be final checks only, but instead should be an iterative process throughout the design cycle. ^[3]

Validation is formally defined by the International Organization for Standardization (ISO) as “confirmation, through the provision of objective evidence, that the requirements for a specific intended use or application have been fulfilled”. ^[4] This definition practically applies to any application, whether it is an object or system. In general, validation ensures that the goals of a project are achieved. It is also meant to give a user confidence that the things they are creating will be useful.

Verification is a process that checks if models, tools, and products are working correctly, according to a given standard or within a defined level of accuracy. ^[5] The process involves comparing properties of one system against the required properties of an ideal system. ^[6] The user of the model must have confidence that their models represent accurate information according to their purpose. For example, a measuring device used to check the distance between

1. Robinson, Stewart. *Simulation: the Practice of Model Development and Use*. Chichester, England: Wiley, (2004). 209.

2. Robinson. *Simulation*. 209.

3. Robinson. *Simulation*. 212.

4. ISO. “Systems and software engineering -- System life cycle processes.” ISO/IEC/IEEE 15288:2015, 2015-05, 4.37 validation.

5. Robinson. *Simulation*. 209.

6. ISO. “Systems and software engineering -- System life cycle processes.” ISO/IEC/IEEE 15288:2015, 2015-05, 4.38 verification.

two objects that are 100 *m* apart, should always give the reading of 100 *m* whenever it is used in that application. In this context, verification is comparable to the process of *calibration*, where tools are set to an accuracy based on a given standard. If a model does not give correct information, then the results of the system cannot be trusted. Verification is considered a subset of the larger concept of validation. ^[7] Therefore, when speaking about validation in general, it refers to both verification and validation, as a total process.

Properties

The thesis follows the information described by Robinson on verification and validation, in their textbook on simulation tools. The definitions apply to both the architectural process and the agent-based model. Robinson states that there are two main concepts in validation: *sufficient accuracy* (tolerance), and *purpose*. ^[8] A model, tool, or system is considered validated if its outcomes have enough accuracy and align with a given purpose.

A model is a representation of a system, and it is a way to simplify a system to help understand how it works. For these reasons, a model should never expect to be completely accurate, or its results taken as 100% correct. ^[9] Due to this uncertainty, verification and validation try to ensure that a model has sufficient accuracy by determining if its results are within a given tolerance. As an example, this idea is like physical tolerances in engineering and construction. A physical dimension on a drawing might require a steel beam to be 1000 *mm* within a tolerance of 5 *mm*. A beam might never be exactly 1000 *mm* due to inaccuracies in manufacturing or material properties. Instead, a tolerance gives a range of sizes that are acceptable for construction. A beam that falls between 995 *mm* and 1005 *mm* is validated based on the drawing.

The level of tolerance for any model is dependant on its purpose. As a result, the specific use of a model needs to be established before it can be validated. For example, the previous steel beam might need to span between two columns. If the beam is too long or too short, then it will not fit the structure. Although, the position where the beam is welded to the column could vary within 10 *mm*. The application of welding gives the constraints for accuracy, and the structural frame gives the condition for validation. Accuracy is described as a range, sometimes as a percentage

7. Robinson. *Simulation*. 210.

8. Robinson. *Simulation*. 210.

9. Robinson. *Simulation*. 210.

from 0% to 100%. Whereas validation is a binary decision; a system is either acceptable or it is not. ^[10]

The thesis applies these ideas to the design of architectural spaces. Tolerances can influence physical constraints (room dimensions), functional constraints (wayfinding visibility), or social constraints (need for amenities). In each situation, there is a range of conditions that are a valid design, but they are limited by the application of the project. For example, an airport holdroom concourse might require access to a concession space (purpose). The concessions are required to be within a 5-minute walking distance from each of the gates (tolerance) and have a variety of 2 or more types of retail spaces (tolerance). A simple validation process for this condition will check if passengers can walk between the gates and the concession spaces within 5 minutes, and if the passengers can engage with more than 2 types of retail space. The architecture is validated if testing shows passengers can access these areas (purpose) and perform according to the conditions (tolerance).

Methods

Testing a system for verification and validation is related to the process of simulation analysis, where each stage of a simulation study requires a method for validation. In their research on verification and validation techniques, Balci explains how V&V methods can be categorized based on mathematical formality. ^[11] There are some systems that can be represented by a pure mathematical relationship, like calculus. Whereas other systems can only be checked philosophically, like if a client is happy with the results. Balci organized V&V methods into six categories, which are, from least to most mathematical: *informal*, *static*, *dynamic*, *symbolic*, *constraint*, and *formal*. ^[12] The complete list of techniques can be seen in Fig.1.2.a. Balci noted that some techniques can fit into more than one category, like structural analysis having static and dynamic testing. Additionally, Balci states that the formalness of the math should increase as the system being validated becomes more complex. ^[13]

While Balci's research was looking at different techniques, Robinson described verification and validation methods within the context of a simulation process. They mention seven methods that include: *conceptual model validation*, *data validation*, *white-box validation*, *black-box*

10. Robinson. *Simulation*. 210.

11. Balci, Osman. "Validation, Verification, and Testing Techniques Throughout the Life Cycle of a Simulation Study." *Annals of operations research* 53, no. 1 (December 1994): 130.

12. Balci, Osman. "Validation". 130-131.

13. Balci, Osman. "Validation". 130.

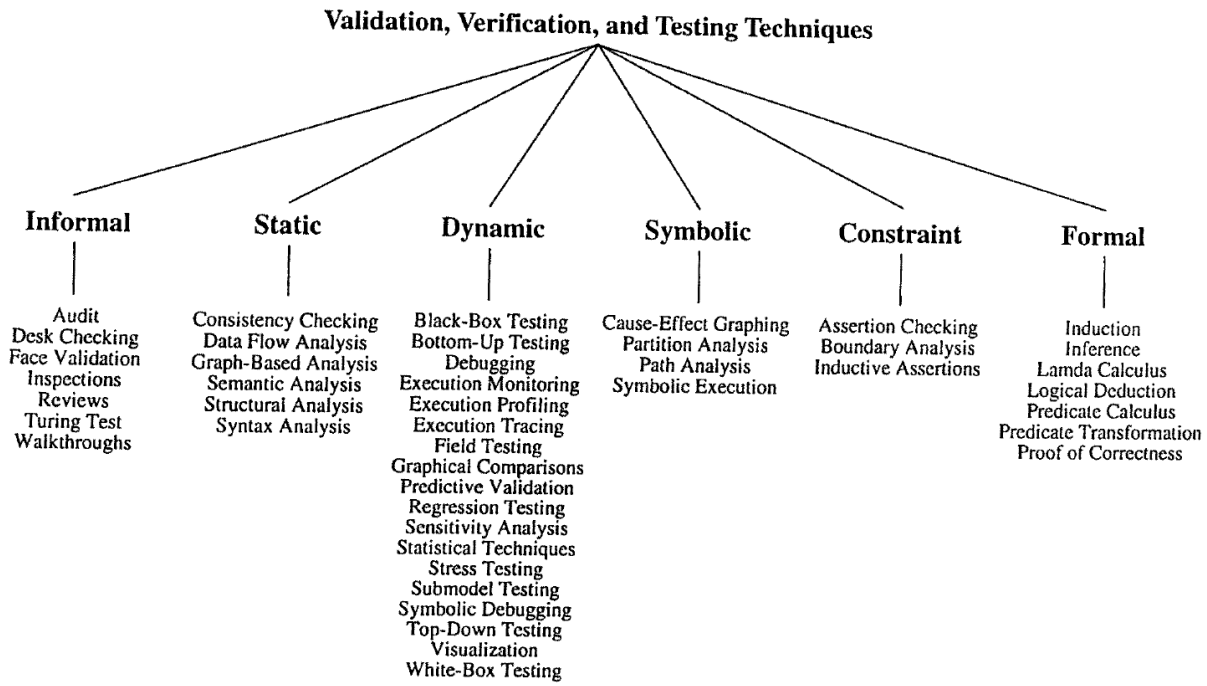


Figure 1.2.a: All verification and validation techniques on a spectrum of mathematical formality, Balci (1994).

validation, experimentation validation, and solution validation. ^[14] Each of these methods may use one or more of the techniques described by Balci. For example, this may involve formally checking if a simulation method outputs the correct values based on mathematics. Similarly, a model can be visually checked to determine if it looks like it is moving correctly. A system is considered validated when at least one of these methods is completed in parallel with a given process, as illustrated in Fig.1.2.b.

Conceptual Model Validation: This determines if all the contexts, assumptions, and simplifications are reasonable enough to meet the goals of the system. The method for validating a conceptual model is dependant on project requirements and specifications. ^[15] A conceptual model can be different from project to project.

Data Validation: This involves checking if both the system and the validation process itself are using relevant information. It also confirms if the data is accurate enough to achieve the given purpose. ^[16] This typically applies to all processes of design and simulation since data is involved at every stage.

White-box Validation: Fundamental parts of a model are checked to see if they correspond to real-world elements under similar conditions, within a given level of accuracy. This involves studying single elements in detail, making sure each part of the model works correctly. ^[17] This is like verification for the system's parts.

Black-box Validation: The overall model is checked to determine if it properly represents the real-world system, under similar conditions, within a given level of accuracy. This involves studying the model's complete operations, to confirm that all parts of the system are working together correctly. ^[18]

Experimental Validation: Any process that uses experimental procedures (non-standard practices) must provide results that are accurate enough to achieve the given purpose. Experimental procedures must also consider the issues of removing biases from initial conditions, controlling the duration of an experiment, replicating the procedures more than once, and analysing the accuracy of the results. ^[19] In other words, when trying a new procedure,

14. Robinson. *Simulation*. 210-211.

15. Robinson. *Simulation*. 214-215.

16. Robinson. *Simulation*. 215.

17. Robinson. *Simulation*. 215.

18. Robinson. *Simulation*. 217.

19. Robinson. *Simulation*. 220.

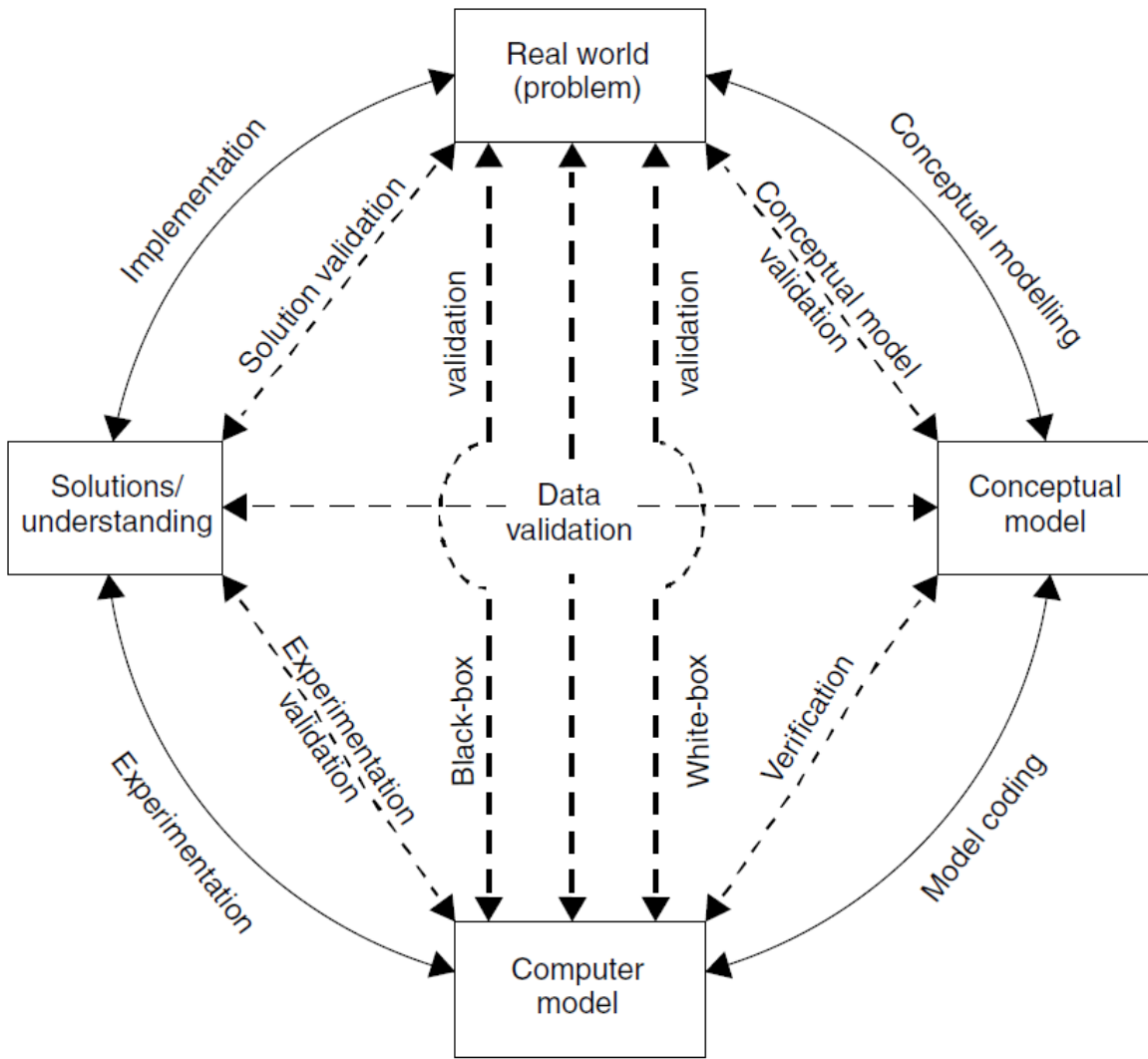


Figure 1.2.b: Simulation processes showing corresponding validation processes, Robinson (2004).

a good process should make it easy to learn from the outcomes of an experiment and help identify a relevant solution to the system.

Solution Validation: This determines if the results from the model are within a given level of accuracy, when compared to the results of the real system. This is like black-box validation, since it looks at the system in total. Although, it only compares the value of the solution, instead of the system components. Solution validation is only possible after an experiment finishes, therefore the results do not affect fundamental components of a model. ^[20] For this reason, it is not meant to validate the entire system, but the solutions can give feedback to the designer.

Applications

Verification and validation are part of a scientific practice that is well established across a wide range of industries like healthcare ^[21], engineering ^[22], building science, computer software ^[23], and economics. The reason these industries use V&V is usually concerned with either human safety or product efficiency. These types of validation do not only cover scientific experiments, but also includes design methods that are equivalent to architectural design. The following are some examples of how validation is used in industries today.

Pharmaceuticals:

The healthcare industry has many instances of validation practices to control the impact to human health. Due to strict regulations, there are also instances of validation for design and manufacturing of healthcare products, as seen in the pharmaceutical industry. The Food and Drug Administration (FDA) is an American organization responsible for regulating food, drugs, and medication. Specifically, the FDA established a standard *Process Validation* for testing the design and manufacturing of drugs. ^[24] The purpose of this validation is to check if a drug has been designed for its intended use. ^[25] The process looks at what type of drug it is, how it is being manufactured, and how well the drug performs. Additionally, the process includes clinical

20. Robinson. *Simulation*. 221.

21. FDA. "Guideline on General Principles of Process Validation". U.S. Department of Health and Human Services Food and Drug Administration, FDA-2008-D-0559. Updated 2018-08-24. 1.

22. FAA. "International Validation and Domestic Certification". Federal Aviation Administration. Accessed November 2020.

https://www.faa.gov/aircraft/air_cert/design_approvals/rotorcraft/val_dom_cert/.

23. ISO. "Systems and software engineering -- System life cycle processes." ISO/IEC/IEEE 15288:2015, 2015-05, 4.38 verification.

24. FDA. "Process Validation". 1.

25. FDA. "Process Validation". 3.

trial studies, which evaluates the drug with a sample population, before allowing the medicine to be available for the public. In addition to obvious health and safety concerns, the goal is to make sure that the object being created does what it is supposed to do. Process validation is not just a formal study, but is a layered scientific design process, which is repeatable, and uses analytical data to determine the performance of the medicine, before people use it. ^[26]

Automotive Engineering:

In the automotive industry, validation is an integral part of the design process for vehicles and machinery. *Design validation* is defined as making sure that a design meets the form and functional requirements based on the product needs, analytical methods, or physical testing. ^[27]

Product needs describe the scope of vehicle to be designed, what the capacity is, the engine size, power output, or fuel consumption. Analytical methods can involve testing components using software tools to study stresses, strain, vibrations, temperature, and fluid flow. This may include Finite Element Analysis (FEA) for structures and thermodynamics, Computational Fluid Dynamics (CFD) for fluid flows, and dynamic analysis for moving components. ^[28] Physical testing typically uses standards established by the Society of Automotive Engineers (SAE). ^[29] This may include four-post testing for suspension systems, dyno testing for engine power output, wind tunnel testing for aerodynamics, or on-road testing for full vehicle dynamics.

Designers will validate these systems analytically before moving into physical prototypes. This allows systems to be solved mathematically before spending resources trying to build the full vehicle. Once the mechanics of the theoretical system are understood, then designers have confidence in how it translates into the physical world. Although architectural design does not involve as much physics, mathematics can give an objective understanding of design behind components in buildings.

26. FDA. "Process Validation". 17.

27. EGS India. "Why, How and When do you perform Design Validation for Automotive Systems?". Solidworks Tech Blog. August 30, 2016. <https://blogs.solidworks.com/tech/2016/08/perform-design-validation-automotive-systems.html>.

28. EGS India. "Why, How and When do you perform Design Validation for Automotive Systems?". Solidworks Tech Blog. August 30, 2016. <https://blogs.solidworks.com/tech/2016/08/perform-design-validation-automotive-systems.html>.

29. SAE. "Browse Standards". Society of Automotive Engineers. Accessed February 2021. <https://www.sae.org/standards>.

Building Codes:

In the architectural industry, building codes are a common example of validation. Building codes validate architecture for human safety, fire protection, structural integrity, accessibility, and environmental impact. ^[30] Historically, building codes were created to prevent repeating city-wide destruction, which was seen in the Great London Fire (1666) or the Great Chicago Fire (1871). This established the need for regulations on wall spacing, materials, ventilation, and drainage, which new constructions followed to ensure human safety.

The National Building Code of Canada (NBC) states that its standards are considered the minimum level of performance required to achieve this type of human safety. ^[31] However, the NBC explain that building codes are not textbooks on building design and construction. Instead, a complete building is dependant on numerous factors, which requires professional knowledge and expertise of good design, beyond the requirement of standard building regulations. ^[32] A building can be validated based on codes, but it can still function poorly for the purpose of the project. For example, a well-built library can be validated for safety regulations but would function poorly if it was used as a hospital, hypothetically. Therefore, building codes are a critical piece of the architectural validation process, but there are additional design elements that these codes do not address.

Building Science:

The field of building sciences is concerned with analyzing the physical effects on buildings. For architecture, the focus is typically on the environmental impact of building design, which consists of operational energy consumption, material choices, and building orientation. This requires analyzing a wide range of subjects including thermal control, air quality, material testing, and lighting.

Design validation in building science typically evaluates the performance of a building, based on the energy transfer through its enclosure and mechanical systems. Industry standards for building science in North America are established by the American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE). They are responsible for keeping standards on testing, analyzing, and maintaining mechanical systems or other building

30. NRCC. *National Building Code of Canada 2015 Volume 1*. National Research Council of Canada. (Ottawa: 2018). vi.

31. NRCC. *National Building Code*. vi.

32. NRCC. *National Building Code*. vi.

components. ^[33] For example, ANSI/ASHRAE/IES Standard 100, is a document about how to retrofit existing buildings to achieve better energy efficiency. ^[34] It regulates new material choices, refurbishing mechanical equipment, and explains how energy usage should be calculated.

Building science shows that it is already possible to make objective design decisions in architecture using scientific practices. If these same methods were applied to spatial performance, then designers can validate a floor plan design based on similar metrics. In addition to building performance as a function for energy, the thesis proposes architectural performance in terms of physical geometry and occupant behaviour. In other words, in addition to validating a building for its energy, buildings could also be validated for its geometry.

Limitations of Validation

According to Robinson, in their textbook on simulations, there are several problems that occur when trying to validate a model. The thesis summarizes Robinson’s problems under six categories: *generalization*, *real-world equivalence*, *real-world interpretation*, *data accuracy*, *time*, and *confidence*.

Generalization: “There is no such thing as general validity”. ^[35] If a model is validated for one system, it does not make it validated for another. Models are made to represent a specific system. Therefore, validating a model only applies to the given purpose. For example, a simulation may be validated for scheduling the number of trains to arrive at a station platform. However, this does not mean the simulation is also validated for calculating the passenger flow rate capacity in the station concourse. If the simulation is also being used for passenger capacity, then there would have to be a separate validation process for that situation. It is not practical for a model to simulate every condition in a transit terminal, due to the large amount of data and simulation time. Instead, it is more efficient to model specific situations, or simple processes, which also gives more confidence in the results. ^[36]

33. ASHRAE. “ASHRAE Standards Strategic Plan 2014-15”. American Society of Heating, Refrigerating and Air Conditioning Engineers, Inc. (July 2, 2014): 3.

34. ASHRAE. “Standard 100-2015 -- Energy Efficiency in Existing Buildings (ANSI Approved/IES Co-sponsored)”. American Society of Heating, Refrigerating and Air Conditioning Engineers, Inc. Accessed February 2021. <https://www.ashrae.org/technical-resources/bookstore/standard-100>.

35. Robinson. *Simulation*. 213.

36. Robinson. *Simulation*. 213.

Real-world Equivalence: “There may be no real world to compare against”. [37] It is common to create a model to predict the future behaviour of a system. However, if there is no real-world metric to compare the results of the model to, then the model cannot be validated. For example, an airport simulation can be validated for existing terminal operations. If the same model is then used to simulate a new terminal building, it may not guarantee the same behaviour after the system changes. [38]

Real-world Interpretation: “Which real world?”. [39] Each person has a different view of the world. The expectations of one person may be completely different to someone else. For example, a passenger waiting to pick up their baggage can feel they wait too long at the baggage carousel. Whereas an airport baggage handler removing bags from a plane can feel like they have very little time to fully empty the entire aircraft. Depending on which of these people we ask to judge the efficiency of the baggage system, will result in different interpretations of the airport’s operations. Likewise, when trying to validate a model, information that is accurate for one person may not be representative to someone else. [40] Choosing what perspective to use will depend on what information a designer wants to communicate.

Data Accuracy: “Often the real-world data are inaccurate”. [41] As mention before, validation involves comparing a model system to an equivalent system in the real-world. If a model is conducted under the same conditions as the real-world, then it is validated if the results are the same. However, this assumes that the real-world results are already accurate. If the data is not accurate, then the model is validating conditions that are not correct. Additionally, assuming a designer does get accurate real-world data, these may only be samples, which also has its own inaccuracies and assumptions. For example, if a researcher records check-in times for passengers over a 1-month period, this only represents one time frame. If check-in times were recorded during a different time of the year, then the sample would have produced different results. Statistics and can help estimate the average check-in times. Although this only provides a probability distribution, which may not be precise enough. [42]

37. Robinson. *Simulation*. 213.

38. Robinson. *Simulation*. 213.

39. Robinson. *Simulation*. 213.

40. Robinson. *Simulation*. 213.

41. Robinson. *Simulation*. 213.

42. Robinson. *Simulation*. 214.

Time: “There is not enough time to verify and validate everything”. [43] Projects have a limited time allocated for modelling and analysis. This affects all aspect from building the simulation tool, to validation, and running experiments. The expectation for the designer is to make sure that a model is validated for the simulation’s scope, key components have overall validation, and experiments are conducted thoroughly. [44]

Confidence: “Confidence not validity”. [45] Ideally, validation should be binary; a model is either validated or it is not. However, like the real-world system, it is not possible to prove 100% validation of a model. Instead, it is more practical to consider a level of confidence. The purpose of V&V is to show where a model is incorrect. Therefore, the more tests that a model can complete, the more confidence that people have in the model’s output. Validation is meant to increase the confidence of the model to a point where it can help make decisions in the design process. If a model has proven confidence in its output, then a designer will be confident in using it to communicate information. [46]

Summary

A goal of this thesis is to increase the confidence of architectural design decisions. Verification and validation (V&V) are an established objective practice that checks if a system or object is meeting requirements or expectations. The thesis considers V&V for both architectural design and the agent simulation. Validation ensures that the goals of a project are achieved. Verification confirms that the tools being used are giving the correct values within a given level of accuracy. Any system can be analyzed using the properties of *purpose* (validation) and *tolerance* (verification). Types of V&V can be categorized based on mathematical formality. This can be as informal as a design review, physical testing, or formal mathematical logic. A thorough validation process will involve checking data, model components, system-wide behaviour, and experimental procedures. Validation is commonly used across a wide range of industries, which are similar to architectural design. Other disciplines incorporate validation into the design process using analytical studies to judge performance based on industry standards. Perfect validation is impossible, because it depends on what parts of a system are modelled and the information gathered from real-world data. Validation is most effective for increasing the confidence of the model’s outputs, so that designers can rely on it to communicate information.

43. Robinson. *Simulation*. 214.

44. Robinson. *Simulation*. 214.

45. Robinson. *Simulation*. 214.

46. Robinson. *Simulation*. 214.

Chapter 1.3

Probability and Statistics

Statistics is the science of collecting, analyzing, and interpreting information. ^[1] Probability is the science of measuring uncertainty using mathematical patterns, which is a foundation for statistical analysis. ^[2] These are fundamental concepts for understanding complex systems in scientific, engineering, financial, and social disciplines. Architecture does not commonly use statistics to analyze designs, since design choices are not thought in terms of probability. However, understanding statistics for this thesis is important for modelling human behaviour in an airport, building a simulation tool, and quantifying architectural value.

Uncertainty

The purpose of having statistical methods in architecture is to make objective scientific judgements given the *uncertainty* and *variation* of data. ^[3] Airports need to consider the behaviour of hundreds of thousands of passengers over time, who are made up of a diverse group of people, in terms of age, social, and cultural differences. Likewise, architecture deals with the uncertainty of designing public spaces to accommodate a wide range of people.

Statistics helps approximate of a wide range of characteristics, instead of modelling a generic type of person. This includes the uncertainty of what activities passengers are doing in an airport, and the variation of people's behaviours and characteristics. These properties are difficult to predict, because they appear to occur by chance, or *randomly*. ^[4] Therefore, simulating these systems requires using probabilistic models (*stochastics*) instead of a deterministic one, ^[5] in which random values are analyzed to understand larger patterns.

1. Walpole, Ronald E et al. *Probability & Statistics for Engineers & Scientists 9th ed.* Boston: Prentice Hall, 2012. 1.

2. Watkins, Joseph. *An Introduction to the Science of Statistics: From Theory to Implementation Preliminary Edition.* University of Arizona: 2016. 3.

3. Evans, Michael J; Rosenthal, Jeffery S. *Probability and Statistics the Science of Uncertainty Second Edition.* University of Toronto: 2009. 1.

4. Banks, Jerry; Carson II, John S; Nelson, Barry L; Nicol, David M. *Discrete-Event System Simulation 4th ed.* Upper Saddle River, N.J: Pearson Prentice Hall, (2005). 131.

5. Banks et al. *Discrete-Event.* 131.

Sampling

Statistics allows designers to make a connection between the people they are testing, and the larger community they are a part of. Simulations use statistics to create an approximate model of a human population, in which population data is usually collected through surveys or experiments. Statistical methods only consider a small set, or *random sample*, of information to predict the expected pattern of the entire system, or *population*.^[6] Instead of considering millions of passengers in a terminal, it is more efficient to estimate behaviour in a random sample of thousands of passengers.^[7] Likewise, instead of considering every type of human interaction, it is more reasonable to simulate behaviour that has a higher probability of occurring in an airport environment, like processing or waiting.

As an example, an airport might see two different types of passengers, frequent business travellers and elderly travellers. In general, frequent business travellers can process very quickly and need little information about where to go in the terminal. In contrast, elderly passengers might take longer than normal to process and may require additional guidance. Airport simulations do not need to worry about the behaviour of a single individual. Instead, they are only concerned with the range of behaviours that are likely to occur for each type of passenger. As a result, an airport model only needs to consider the probability of elderly travellers and business travellers.

For instance, using statistics, a passenger survey in the National Academies shows that the average processing time for business travellers is 2.8 to 3.1 minutes, with 95% confidence.^[8] These times are based on recorded domestic passenger data for a specific airline in an existing facility. In this situation, the time between 2.8 and 3.1 minutes is considered the expected *probability distribution* for business traveller's process times. Although, the exact behaviour of these passengers is unknown, the airport knows they will only see behaviour outside this range 5% of the time. Therefore, if a new terminal design could process simulated business travellers within this distribution, then the airport can be confident that the design is suitable for the expected number of people.

6. Walpole, et al. *Probability & Statistics*. 2.

7. Walpole, et al. *Probability & Statistics*. 227.

8. National Academies of Sciences, Engineering, and Medicine. "Airport Passenger-Related Processing Rates Guidebook". *Washington, DC: The National Academies Press*. (2009): 38.

Law of Large Numbers

When dealing with uncertainty for a large population, it is useful to think of many samples over time instead of looking at a single moment or individual. In their textbook on statistics, Watkins shows how *random sampling* from a given population becomes predictable, or *stable*, over time.

For instance, suppose an airport wants to know the average weight of bags passengers check in for their flight. If an airport worker chooses to randomly weigh checked-in bags, they may notice that some bags are lighter, around 20 kg, whereas other bags are heavier, closer to 50 kg. If the airport worker kept a running average of weights they measure, then, because of the differences of each bag, there might be large fluctuations in the average weight at the beginning. ^[9]

However, over time, as the worker continues to weigh more bags, the running average should expect to settle and converge to the true weight of checked-in bags. ^[10] Watkins explains that, in probability theory, this is referred to as the *law of large numbers*. ^[11] If there is a sequence of n random variables X_1, X_2, \dots, X_n with the same distribution, then the average or *sample mean* is,

$$\bar{X} = \frac{1}{n} S_n = \frac{1}{n} (X_1 + X_2 + \dots + X_n),$$

where \bar{X} is a random variable.

Watkin conducted a number of experiments with new-born baby weights to demonstrate this law of large numbers, as shown in Fig.1.3.a. For small values of n , the average changes rapidly since each baby has a different weight. However, eventually, as n reaches 100 random samples, the average converges down to a more stable value. The average weight from all experiments settles to a similar value, despite having different initial values, since the random samples have the same distributions.

If these experiments continued for much larger sample sizes, or as n approaches infinity, then the probability that the difference between the sample mean and the population mean is greater than a positive number is 0% (*weak law of large numbers*). ^[12] In other words, the probability of the sample mean being equal to the population mean will be 100% (*strong law of large*

9. Watkins. *Science of Statistics*. 179.

10. Watkins. *Science of Statistics*. 179.

11. Watkins. *Science of Statistics*. 179.

12. Evans, Michael J; Rosenthal, Jeffery S. *Probability and Statistics the Science of Uncertainty Second Edition*. University of Toronto: 2009. 206.

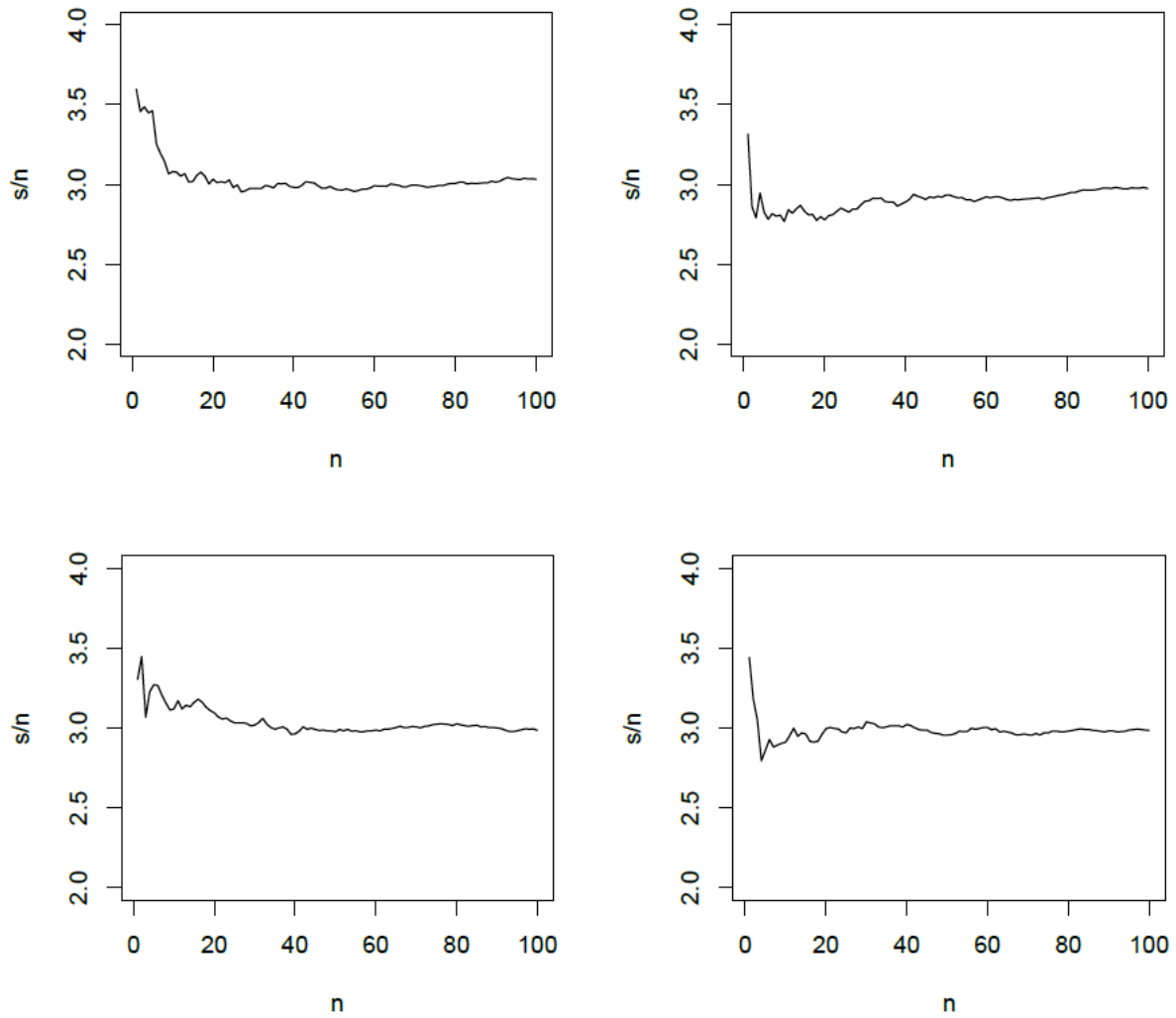


Figure 1.3.a: *Graphs illustrating the law of large numbers from random sampling of new-born baby weights, Watkins (2016).*

numbers).^[13] Therefore, the more information gathered in the system, the closer it becomes to the true value.

Monte Carlo Method

Monte Carlo method is a type of computer algorithm that uses *stochastics*, or *random sampling* to get a numerical result in a system that is very difficult to solve analytically.^[14] Many professional fields use this type of random sampling in computer simulations to solve numerical problems for uncertain situations, whether it is particle physics or material testing.^[15]

The basic idea of Monte Carlo partly works like a “guess-and-check” method. Given a defined domain, input values are randomly generated and solved deterministically within the domain. Once there are enough guesses, or samples, then the results are combined to approximate the true solution. The method works on the principle of the law of large numbers. The more values there are, the closer it becomes to the true value, as the number of samples approaches infinity. A common illustration of a Monte Carlo method is through mathematical integration. Fig.1.3.b shows a Monte Carlo approximation for the value of π . The simulation randomly generates points inside a unit square. The fraction of points that fall inside of the circle approaches $\pi/4$ as n becomes larger.^[16]

The essence of a Monte Carlo method is the random variables, specifically, a *random number generator*.^[17] There is an entire science behind the logic of producing random variables. Fundamentally, random variables in a computer are not truly random. Instead, random number generators are carefully chosen deterministic models that mimic random outputs. These require a user to input a given value, or *seed*.^[18] Since a random number generator is deterministic, the same seed will always produce the same output. Using different seeds produces random results. A simple example of a random seed is using the current date and time (2021/02/23 12:54:24), which provides a unique value every second.

13. Evans et al. *Probability and Statistics*. 211.

14. Watkins. *Science of Statistics*. 182.

15. D.P. Kroese, T. Taimre, Z.I. Botev. *Handbook of Monte Carlo Methods*. Wiley Series in Probability and Statistics, John Wiley & Sons, New York, (2011). xvii.

16. Nicoguardo. “File:Pi 30K.gif”. Wikimedia Commons. February 16, 2017.
https://commons.wikimedia.org/wiki/File:Pi_30K.gif.

17. Kroese, et al. *Handbook of Monte Carlo Methods*. 9.

18. Kroese, et al. *Handbook of Monte Carlo Methods*. 9.

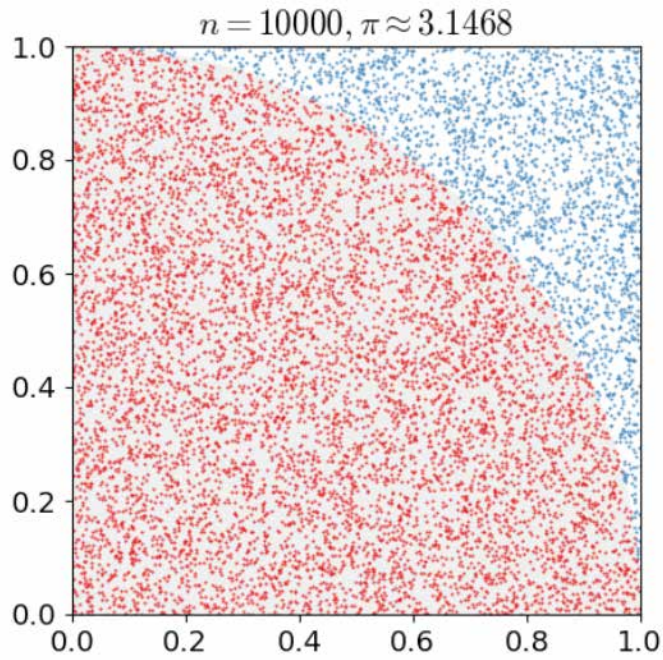


Figure 1.3.b: Monte Carlo method approaches the value for π based on the fraction of random points that fall inside the circle within a unit square, Nicoguardo (2017).

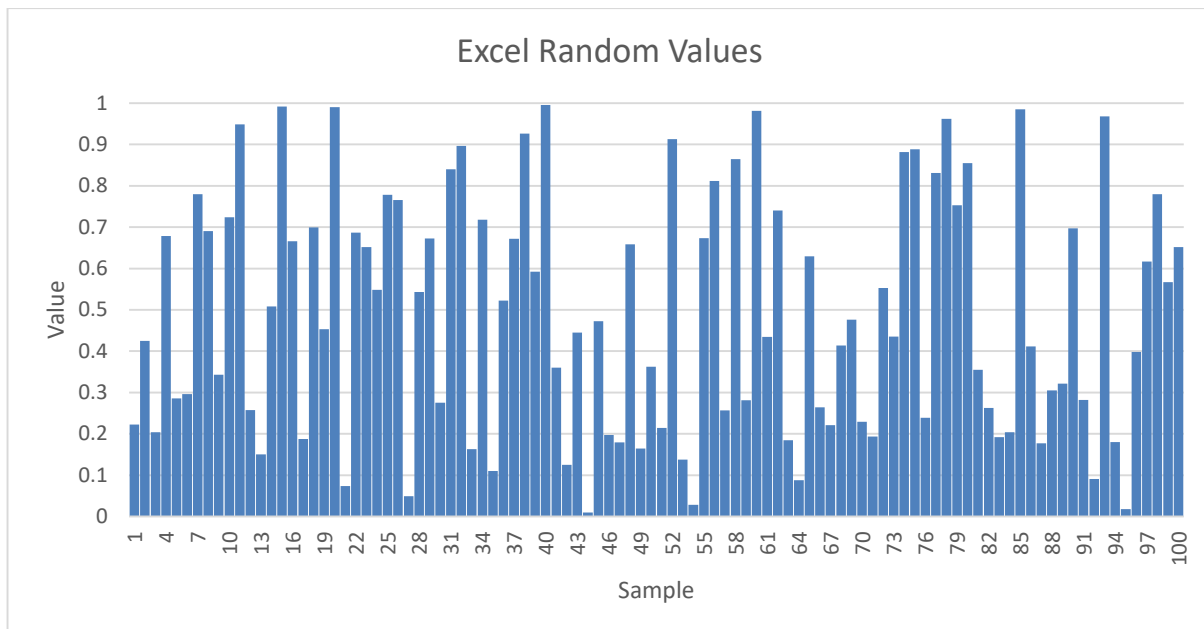


Figure 1.3.c: Graph showing 100 random values between 0.0 and 1.0 in Excel.

The goal is to produce an output that someone could not differentiate from a true random distribution. ^[19] Random distributions do not normally have a recognizable pattern. There may be clumping or voids, but it is not uniform or consistent. An example of a random distribution created using Excel's *RAND* function is illustrated in Fig.1.3.c. Additionally, there are both theoretical and empirical checks to verify if numbers are truly random. ^[20] This usually involves a wide range of statistical tests which compares outputs from a random number generator to some known random variables.

The output from a Monte Carlo simulation will produce some type of *probability distribution*. Probability distributions can either be discrete or continuous, which are also called *probability density functions* (PDF). For discrete distributions, the probability is the value at a given point. However, for a continuous distribution, or PDF, the probability is the area under the curve, or the integral of the function, over a domain.

There are numerous types of distributions, which are classified based on the shape of the graph. The shape also determines how the data is analysed statistically and what application it represents. Some common distributions include *normal*, *logarithmic*, *uniform*, *triangular*, *binomial*, and many others, which are illustrated in Fig.1.3.d. Typically, experimental data will produce a discrete distribution, which is commonly graphed in a *histogram*. Fig.1.3.e shows a probability distribution from a Monte Carlo simulation that approaches a *normal curve*.

Essentially, the true value of a population or system is unknown. These distributions illustrate a range of values that are likely to occur based on the sample of information. Each distribution informs which statistical methods would best match the patterns from the resulting data. This helps narrow down possible behaviours of a system, despite the uncertainty of the larger population.

19. Kroese, et al. *Handbook of Monte Carlo Methods*. 9.

20. Kroese, et al. *Handbook of Monte Carlo Methods*. 18.

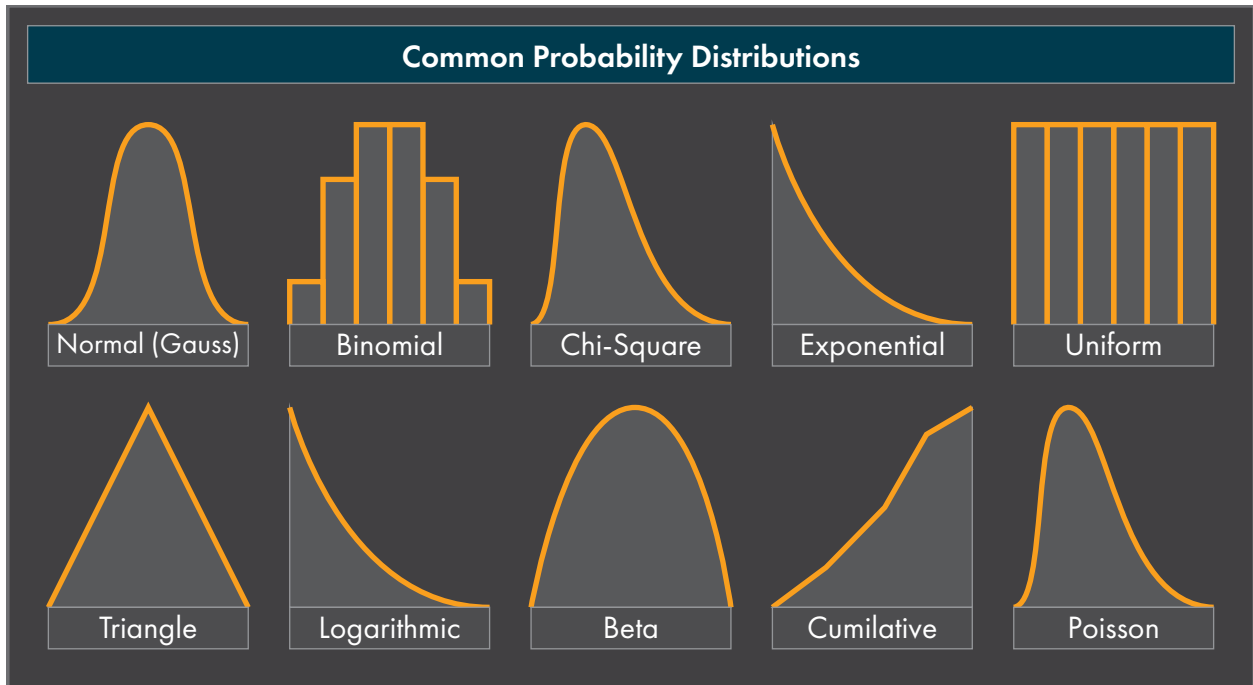


Figure 1.3.d: Some common probability distributions as a result of a Monte Carlo simulation, which informs statistical behaviour.

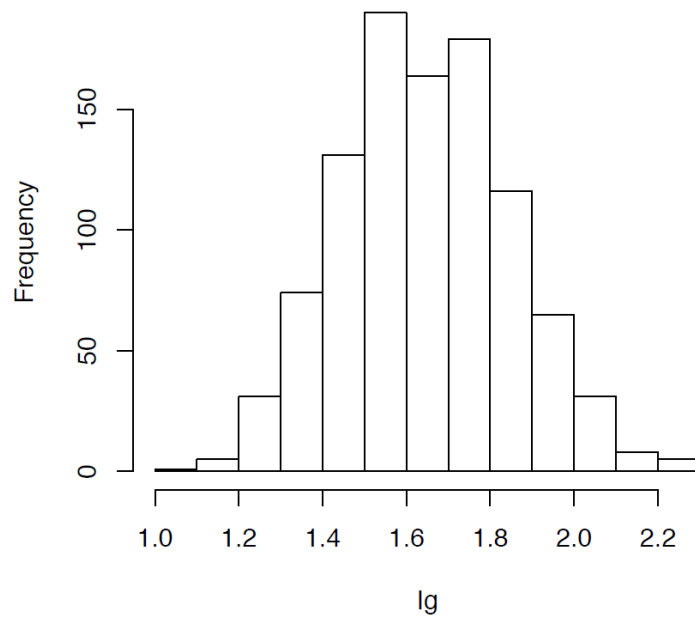


Figure 1.3.e: Histogram of 1000 random Monte Carlo values estimating the area of integral, as a binomial distribution, which approaches a normal curve, Watkins (2016).

Normal Distribution

A *normal distribution* is one of the most fundamental probability distributions in statistics because it shows up in natural patterns and many different applications. It is generally defined by a *mean* (μ) and a *standard deviation* (σ), or *variance* (σ^2), in the equation:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x - \mu)^2}{2\sigma^2}\right),$$

where the mean (average) defines the location of the peak (highest probability), and the standard deviation defines how spread out the curve (or data) is. ^[21]

A physical example demonstrating why a normal distribution is produced from natural phenomenon is seen in a Galton board (Fig.1.3.f). ^[22] This is a board filled with numerous pegs, which small balls are dropped into. The balls bounce off these pegs and land into narrow bins below. Due to the geometry of the pegs, a ball has a 50% chance of falling to the right or left when it hits each peg (Fig.1.3.g). As a result, there is a chance that dropping more than one ball into the board may not land in the same bin. Like a Monte Carlo simulation, if many balls were dropped into the board at once, then the volume of balls in each bin forms a probability distribution, specifically a *binomial distribution*. If the bins were sufficiently narrow and there were infinitely many balls dropped into the board, then the result will become a normal distribution.

Fundamentally, the normal distribution is important because it represents the sum of many independent events in a complex system. Like the balls that have a chance of bouncing off the pegs in the Galton board in different directions, passengers in an airport terminal encounter many situations that also have a chance of going in one direction or another. Events like, if a passenger has one bag or two bags, if a passenger chooses one airline over another, if a passenger chooses to get something to eat, or if a passenger chooses to wait in a concourse. Statistically speaking, passengers are just balls colliding with the pegs of daily experiences in a Galton board airport terminal. Essentially, this can apply to any complex system.

21. Khan Academy. "Deep definition of the normal distribution". Math, Statistics and probability, Modelling data distributions, More on normal distributions. Accessed February 2021. <https://www.khanacademy.org/math/statistics-probability/modeling-distributions-of-data/more-on-normal-distributions/v/introduction-to-the-normal-distribution>. 4:18 - 5:40.

22. Galea, Alexander. "Galton's Peg Board and the Central Limit Theorem". WordPress. March 11, 2016. <https://galeascience.wordpress.com/2016/03/11/galtons-peg-board-and-the-central-limit-theorem/>.

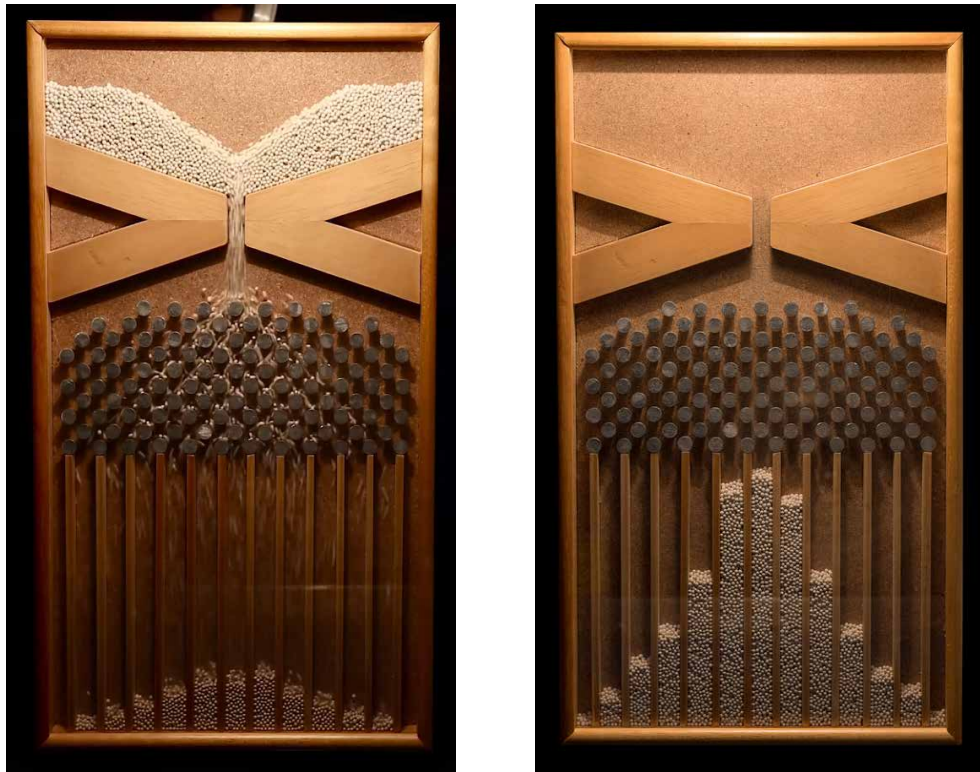


Figure 1.3.f: A Galton board is a physical example of a Monte Carlo simulation, which shows how natural randomness can result in a normal probability distribution, Argenton (2016).

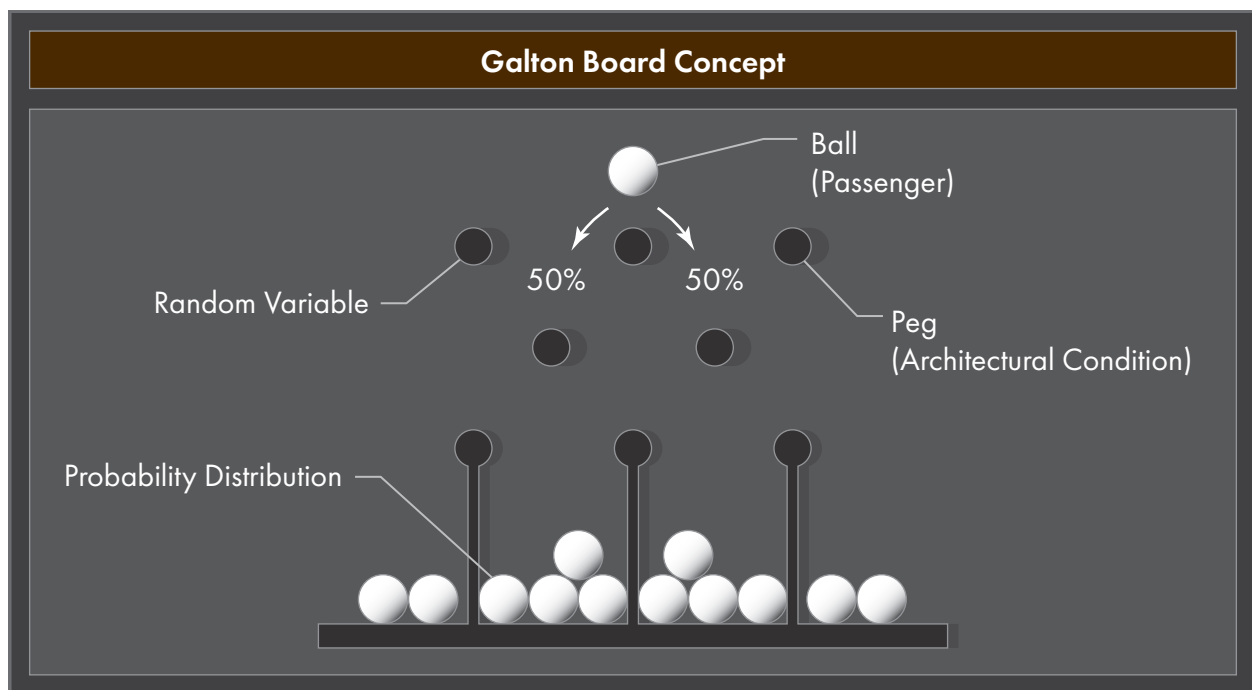


Figure 1.3.g: When a ball hits a peg, it has a 50% chance of going to the right or left, which is comparable to random events in the real world, based on diagram by Galea (2016), redrawn by author.

Central Limit Theorem

Every event in a complex system can be thought of as an independent random event. What statistics shows is that the sum of those events, or many infinite individual experiences people encounter everyday, will always produce a normal distribution. Even if the output from a single event does not happen to be normally distributed, like flipping a coin heads or tails, the sum of all the average outcomes will always approach a normal distribution. ^[23] This is proven mathematically by the *central limit theorem*.

The central limit theorem explains how statistical methods that apply to normal distributions will also work for any probability distribution generated from independent random variables, even if they are not normally distributed already. This is important since any probability distribution with independent variables can use the same mathematics.

The process involves taking sample values from an arbitrary probability distribution and calculating the average of that sample, or the *sample mean*, for many trials. From the law of large numbers, given a large enough sample size, n , and enough trials, the resulting distribution of those sample means will be normal (Fig.1.3.h). ^[24] In general, given a population with a known mean μ and variance σ^2 , if a random sample of size n from that population has a mean of \bar{X} , then the distribution will follow,

$$Z = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}}$$

as $n \rightarrow \infty$, which converges to a standard normal distribution $N(0,1)$, or a normal distribution with a mean of 0 and a variance of 1. ^[25]

Experimentally, the variance of the sample mean distribution $\sigma_{\bar{X}}^2$ can be calculated as the variance of the original population σ^2 and divided by the sample size n ,

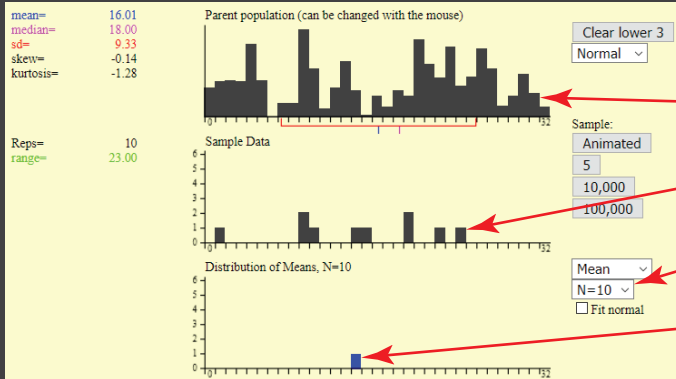
$$\sigma_{\bar{X}}^2 = \frac{\sigma^2}{n}.$$

23. Khan Academy. “Deep definition of the normal distribution”. 4:18 - 5:40.

24. Khan Academy. “Central Limit Theorem”. Math, AP® / College Statistics, Sampling Distributions, Sampling Distributions of a sample mean. Accessed February 2021.
<https://www.khanacademy.org/math/ap-statistics/sampling-distribution-ap/sampling-distribution-mean/v/central-limit-theorem>.

25. Walpole, et al. *Probability & Statistics*. 234.

RVLS Central Limit Theorem Simulation



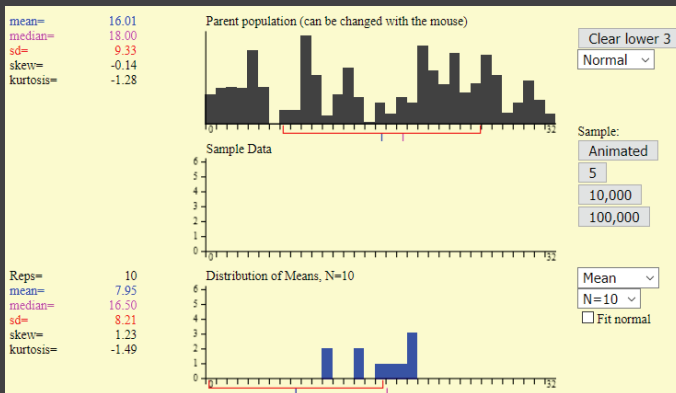
Trials: 1

Population Probability Distribution

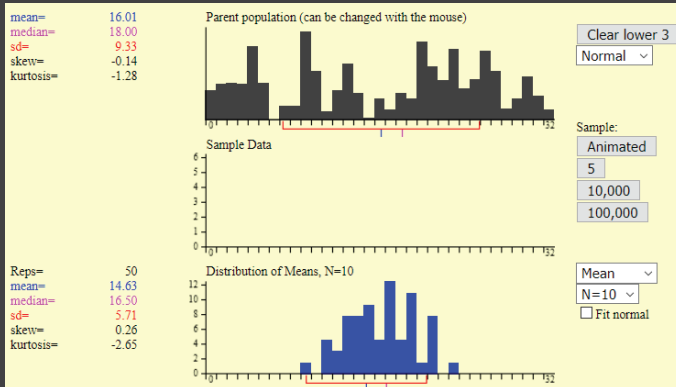
Random Samples from Population

Number of samples; $n = 10$

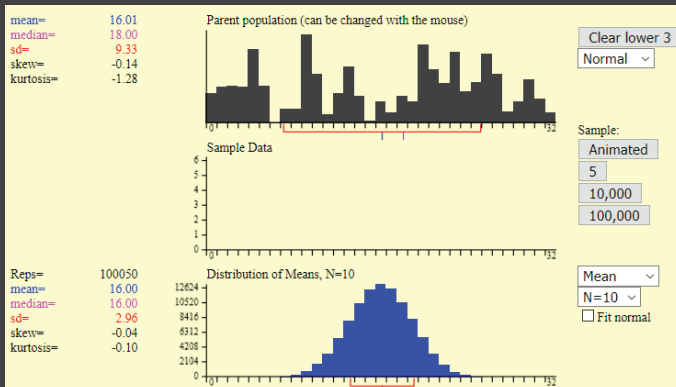
Sample Mean Distribution



Trials: 10



Trials: 50



Trials: 100 000

Figure 1.3.h: Sampling simulation demonstrating the Central Limit Theorem process, built by the Rice Virtual Lab in Statistics (RVLS).

Additionally, the sample mean \bar{X} will always be the same as the population mean μ .^[26] For most applications, a sample size of $n \geq 30$ is enough to get a good normal approximation.^[27] This usually has a variance, or error, within 1% of a true normal curve, which is adequate for the purpose of this thesis.^[28]

Summary

Probability and statistics are the science of analysing data and measuring uncertainty using mathematical patterns. Architectural design is not normally thought in terms of probability. However, statistics is helpful for quantifying uncertainty and variation of data like airport terminals which must accommodate a wide range of people. Simulating these types of systems requires using probabilistic models because most things that are difficult to predict appear to occur randomly. When quantifying complex systems, it is easier to consider a random sample instead of the entire population. For example, differentiating between business and elderly travellers can be simplified to the probability of time each type of passenger spends in an airport. Instead of looking at the behaviour of one individual, which can fluctuate from person to person, characteristics become more stable when considering many samples over time. Based on the law of large numbers, if there are an infinite number of samples, the characteristics of a system will converge to a true value. This applies to Monte Carlo methods, which uses numerous random samples to get a numerical result for a system that is difficult to solve analytically. Monte Carlo simulations use random number generators to provide sample values within a given domain. The result of a Monte Carlo simulation produces some type of probability distribution, which illustrates a range of values that are likely to occur based on the sample. A normal distribution is the most common probability distribution because it frequently shows up in natural patterns, which is defined by a mean and standard deviation or variance. The normal distribution is important because it represents the sum of many independent events in any complex system, which is shown physically in a Galton board. Statistics shows that the sum of independent random events will always produce a normal distribution, even if events' probabilities are not normally distributed. This is proven by the Central Limit Theorem, which explains how the mathematics of normal distributions also apply to any probability distribution with independent variables. Fundamentally, any random events in a complex system, like an airport, can be quantified by probability distributions.

26. Khan Academy. "Central Limit Theorem".

27. Walpole, et al. *Probability & Statistics*. 234.

28. Khan Academy. "Central Limit Theorem".

Chapter 1.4

Simulation Modelling

This chapter introduces what a simulation model is, and the process for using different types of models. The thesis's primary understanding of simulation models is based on the work by Banks et al. in their textbook, *Discrete-Event System Simulation*. They explain when it is appropriate to use simulations, the process for simulation testing, and applications of existing tools. ^[1] This chapter also covers what simulations are used in the design of airports, the limits of existing tools for the purpose of architectural analysis, and what tools the thesis considers for creating an agent-based model.

Modelling Types

A model is a representation of a system, used for the purpose of understanding how the system works. Computer simulations are a specific type of mathematical model. ^[2] They are either based on symbolic algebraic equations or physical relationships. Parabolic and exponential functions are examples of algebraic equations, and properties like time, distance, and mass make up physical relationships. Simulation models have three basic properties: *time*, *randomness*, and *progression*.

Time: The first property of a simulation model is the influence of time. Simulations are either *static* or *dynamic*. A static model represents a system at one point in time, which is similar to solving a single function. An example of a static simulation is calculating how many passengers an airplane can carry. The total number of passengers is constant and is not related to time. A dynamic simulation represents a system over time, or specifically a time dependant function. The number of passengers that have passed through airport security from 09:00 to 17:00 is an example of a dynamic model.

Randomness: The second property of a simulation model is randomness. A simulation that has no random variables is *deterministic*. In a deterministic model, the given inputs have known values and will always produce the same output value. An example of a deterministic simulation

1. Banks, Jerry; Carson II, John S; Nelson, Barry L; Nicol, David M. *Discrete-Event System Simulation 4th ed.* Upper Saddle River, N.J: Pearson Prentice Hall, (2005). 3-9.

2. Banks et al. *Discrete-Event*. 11.

is if a train is scheduled to arrive at 07:30, it will always arrive at 07:30. By contrast, a simulation model that has random variables is *stochastic*. In a stochastic model, the input values are random, which results in random output values. For example, if trains arrive randomly between 07:00 and 08:00, they will produce a random number of passengers in a station over time. Since the outputs are random, the model is only an approximation of a real-world system. Therefore, stochastic outputs use statistics, like the average number of passengers, to estimate the actual system behaviour. ^[3]

Progression: The final property describes how a simulation model progresses, or changes, over time. Simulations are either *discrete* or *continuous* (Fig.1.4.a). A discrete model, or a *discrete-event simulation*, represents a system or process over fixed time steps. ^[4] The *state* of the simulation, or the value of its variables, is static at any given time. This can describe the location of a passenger or the number of bags on a conveyor belt. Variables in a simulation only change when a time step occurs. For example, the location of walking passengers only updates after a few seconds. In contrast, a continuous simulation has variables that are always changing over time. An example of a continuous simulation is water flow in a pipe. However, continuous systems do not always use continuous models. ^[5] The same is true for discrete systems. For example, a model of water flow in a pipe can be discrete, if the value of pressure head only updates after a given time step. Likewise, a model of a crowd can be continuous if each person is represented as a particle in a fluid flow model.

The reason a simulation would prefer to use a discrete model or a continuous model, is dependant on the application. In general, a discrete model can be easier to calculate than a continuous model because a computer can update the state of a discrete model less frequently. Continuous models rely on using differential equations to represent rates of change, whereas discrete models can take larger constant time steps. However, this means discrete models are only an approximation, where the accuracy of the model is depended on how small the time steps are. Due to this approximation, analysing the results of discrete models requires using *numerical* methods rather than *analytical* methods. ^[6] This means that, instead of using deductive reasoning (i.e. solving an equation) to find an exact solution, discrete modelling must use *trial and error* to approach a solution. Increasing a simulation's accuracy can required large

3. Banks et al. *Discrete-Event*. 12.

4. Banks et al. *Discrete-Event*. 9.

5. Banks et al. *Discrete-Event*. 12

6. Banks et al. *Discrete-Event*. 12.

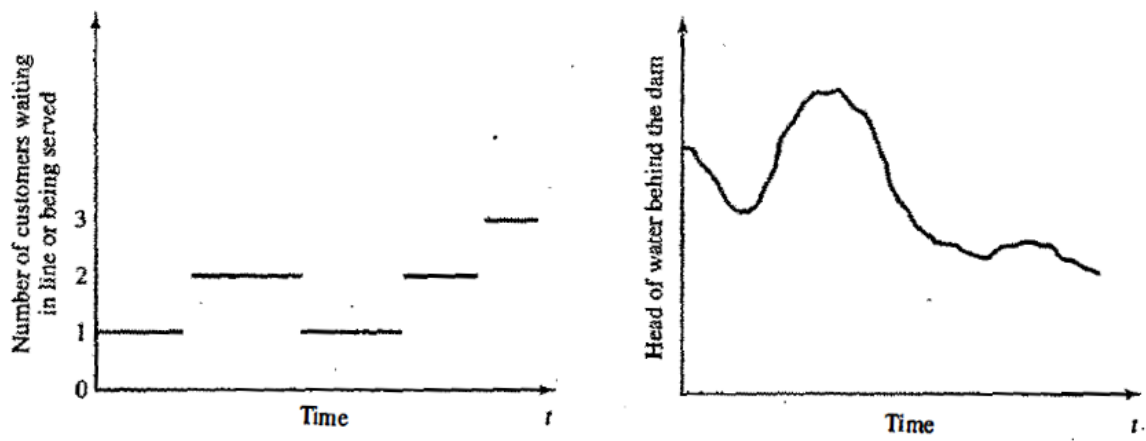


Figure 1.4.a: *Graphs illustrating a discrete system (left) and a continuous system, Banks et al. (2005).*

amounts of data. Instead, simulation tests, or *trial runs*, can be repeated to check if results approach a stable value over long periods of time.

The model considered for the thesis's agent-based simulation can be described as: dynamic, stochastic, discrete, and numerical. It exists over time, it uses random variables, it progresses in constant time-steps, and it is solved by trial and error.

Airport Simulation Types

The design and operation of airports requires understanding the interaction of logistics, security, and passengers in a wide range of scales. It is not practical to build a complete airport to see how it works or fix changes after finding aspects that have poor performance. ^[7] As a result, all critical areas in an airport use some form of simulation during the planning stages to help understand how systems will perform. The architectural design process can use these practices to better understand how existing industries quantify complex systems.

In their report on airport simulation options, the National Academies reviews a wide range of existing tools. They explain that simulations help study airspace, airfield, terminal, and curbside for daily operations. ^[8] Simulations are important during airport development for making sure people are safe, systems work efficiently, and that the airport is profitable. Some simulation studies include airspace traffic modelling (Fig.1.4.b), master planning, capacity-demand forecasting, terminal passenger flow (Fig.1.4.c), curbside traffic capacity, and environmental impact assessments. ^[9]

Airport simulations can be as simple as a spreadsheet stochastic analysis, or more complex dynamic flight data simulations. ^[10] The complexity of a simulation is scaled based on *fidelity*, which describes how closely models match a real-world system. ^[11] A high-fidelity simulation can model small-scale interactions, like the number of processed passengers in a given area over time in a capacity/delay model. Whereas a low-fidelity simulation only gives a broad summary of data, like the total number of passengers in a look-up table. The thesis's agent-based

7. Rittel, Webber. "Dilemmas in a General Theory of Planning." *Policy sciences* 4, no. 2 (June 1973): 163.

8. National Academies of Sciences, Engineering, and Medicine. "Simulation Options for Airport Planning". *Washington, DC: The National Academies Press*. (2019): 3.

9. National Academies. "Simulation Options". 3.

10. National Academies. "Simulation Options". 3.

11. National Academies. "Simulation Options". 4.



Figure 1.4.b: AirTop airspace simulation, National Academies (2005), sourced from AirTopSoft.



Figure 1.4.c: ARCport terminal simulation, Proulx (2014).

simulation expects to approach high-fidelity only if agents' interaction with architecture matches how people would interact in the real-world.

The National Academies mention several mathematical techniques available for planners to model airport systems. The most common techniques include, *spreadsheet models*, *queuing theory*, *optimization techniques*, *Monte-Carlo simulations*, and *discrete-event simulations*.^[12] The inputs for these methods either use recorded data from existing operations or random variables. A complete analysis of an airport system will typically involve all these techniques at some stage of the design process, depending on the fidelity of the simulation.^[13] If the thesis's agent-based simulation wants to replicate passenger processing, it needs to understand the benefits of each technique and when to use them.

Spreadsheet model: A table or chart that records model values and can calculate predictions based on historical airport data. For example, a spreadsheet model can calculate the required size of a security screening area, based on the number of queue lines and expected rate of passengers (Fig.1.4.d). Planners can use these spreadsheets to update calculations as design changes take place in other areas of the terminal.

Queuing theory: Uses a network of resources to illustrate a dynamic change in demand. For example, a resource can be the number of service counters, and the demand comes from passengers using the service counter, or resource, for check-in (Fig.1.4.e).

Optimization techniques: A method in a dynamic environment that tries to find the maximum or minimum use of resources. Queueing models commonly use optimization to maximize the number of active service counters.

Monte Carlo simulations: A method in statistics that selects repeated random variables from a sample population based on a probability distribution (details in Probability and Statistics). For example, a check-in area can simulate passenger processing by selecting a random number of people who arrive within a given range of time.

Discrete event simulations: A process that updates system variables in small time-steps over a given time period (as described earlier in Modelling Types). For example, this can be a simulation of people moving through a queue line, which might update a passenger's position once every second.

12. National Academies. "Simulation Options". 8.

13. National Academies. "Simulation Options". 8.

	A	B	C	D	E	F
1	Security Screening					
2	<input type="button" value="RETURN TO TABLE OF CONTENTS"/>	<input type="button" value="Go To User's Guide"/>	Input Data Values		<input type="text"/>	
3			Calculated Values		<input type="text"/>	
4			Linked or Predetermined Values		<input type="text"/>	
5					<input type="text"/>	
6					<input type="text"/>	
7	DEMAND	INPUTS	OUTPUTS			
8	<i>Use throughput value from Check-In Model</i>	750				
9	Peak 30 min Originating Passengers from Check-In	641	linked to Check In/ Ticketing			
10	% Additional Traffic (non-passenger, employees, crew)	15%				
11	Total Peak Period Security Traffic (passengers)			737		
12	Throughput Rate (Passengers/Hour per lane)	175				
13	# of Passengers Processed/minute per lane			2.9		
14	Maximum Target Wait Time (in Queue)	8				
15	Minimum Required # of Screening Lanes			7	Starting Value	
16	QUEUE MODEL FOR REQUIRED SCREENING LANES				Decision Value	
17	# of Screening Lanes for Queue Model Input	9				
18	Max. Wait Time in Queue (Min.)			8.1	Adequate # of Lanes	
19	EXISTING CONDITIONS					
20	Depth of Security Queue (ft)	20				
21	Width of Scanning Lane Module (2 Lanes) (ft.)	25				
22	Overall Length of Check Point Area (ft)	40				
23	Reconciliation Area Depth (ft)	10				
24	SPACE REQUIREMENTS					
25	Security Queue Area (sq. ft)			2,250		
26	Passengers in Queue based on Queue Wait Time			212		
27	Passenger Space Required for LOS Input (sq. ft/pax)	10.8	(Review the IATA Table)			
28	Required Security Queue Area for LOS Input (sq. ft)			2,294		
29	Passenger Space with Current Dimensions (sq. ft/pax)			10.6	More Queue Needed	
30	Total Checkpoint Area -tables, equipment, search area (sq. ft)			5,625		
31	Total Security Screening Area (sq. ft)			7,875		
32						

Figure 1.4.d: A spreadsheet model calculating the area required for security screening, National Academies (2010).

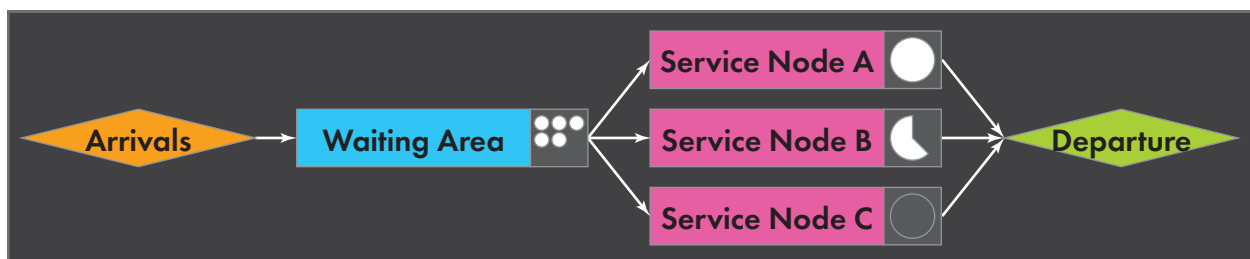


Figure 1.4.e: Basic queuing node model. There are 5 people in the Waiting Area, the Service Node A resource is filled, Node B is busy and still has time to process, and Node C resource is available. Based on diagram by Dt-rush-8 (2018), drawn by author.

Some techniques are better suited for specific applications, like optimization for the airspace, or queuing theory for passenger processing. ^[14] However, each technique can be applied to any system. For example, queuing theory can also be used to model airplanes landing, by assigning planes runways as a resource. The thesis describes its agent-based model as a discrete-event simulation, although a thorough test of a terminal’s architecture will involve these other techniques as well.

An important technique to consider for this thesis is Monte-Carlo simulations. This uses statistical uncertainty to model factors that are difficult to predict, like waiting time. ^[15] It applies random input variables to account for variation in passenger demand, human behaviour, and resource processing time. The benefit of this technique comes from simulating multiple trial runs to approach an expected value. ^[16]

Simulation Process

For any simulation, Banks et al. describe the process necessary for a thorough study, which they divide into four parts: *discovery period*, *model building*, *experimentation*, and *implementation*. These categories are further divided into twelve steps, as illustrated in Fig. 1.4.f. ^[17] The thesis sees these steps as representative of a good architectural design process, which, in addition to the validation process, can objectify design choices.

Discovery Period:

Problem Statement: Identifying what problem the simulation is trying to solve. This is defined by the clients, or the designers, which describes the scope of the simulation or system. The exact nature of the problem might not be known at this stage. Therefore, it is possible to adjust the problem statement as the simulation study progresses. ^[18] For an airport, a problem statement defines the scope, like the airspace, airfield, terminal building, or curbside. ^[19]

Objectives and Plan: This outlines the goals for the study, which the simulation hopes to achieve. Firstly, this decides if simulations are an appropriate tool for the job, and secondly, what type of simulation would be helpful for solving the problem. The plan describes what type

14. National Academies. “Simulation Options”. 8.

15. National Academies. “Simulation Options”. 8.

16. National Academies. “Simulation Options”. 8.

17. Banks et al. *Discrete-Event*. 12

18. Banks et al. *Discrete-Event*. 12

19. National Academies. “Simulation Options”. 14-15.

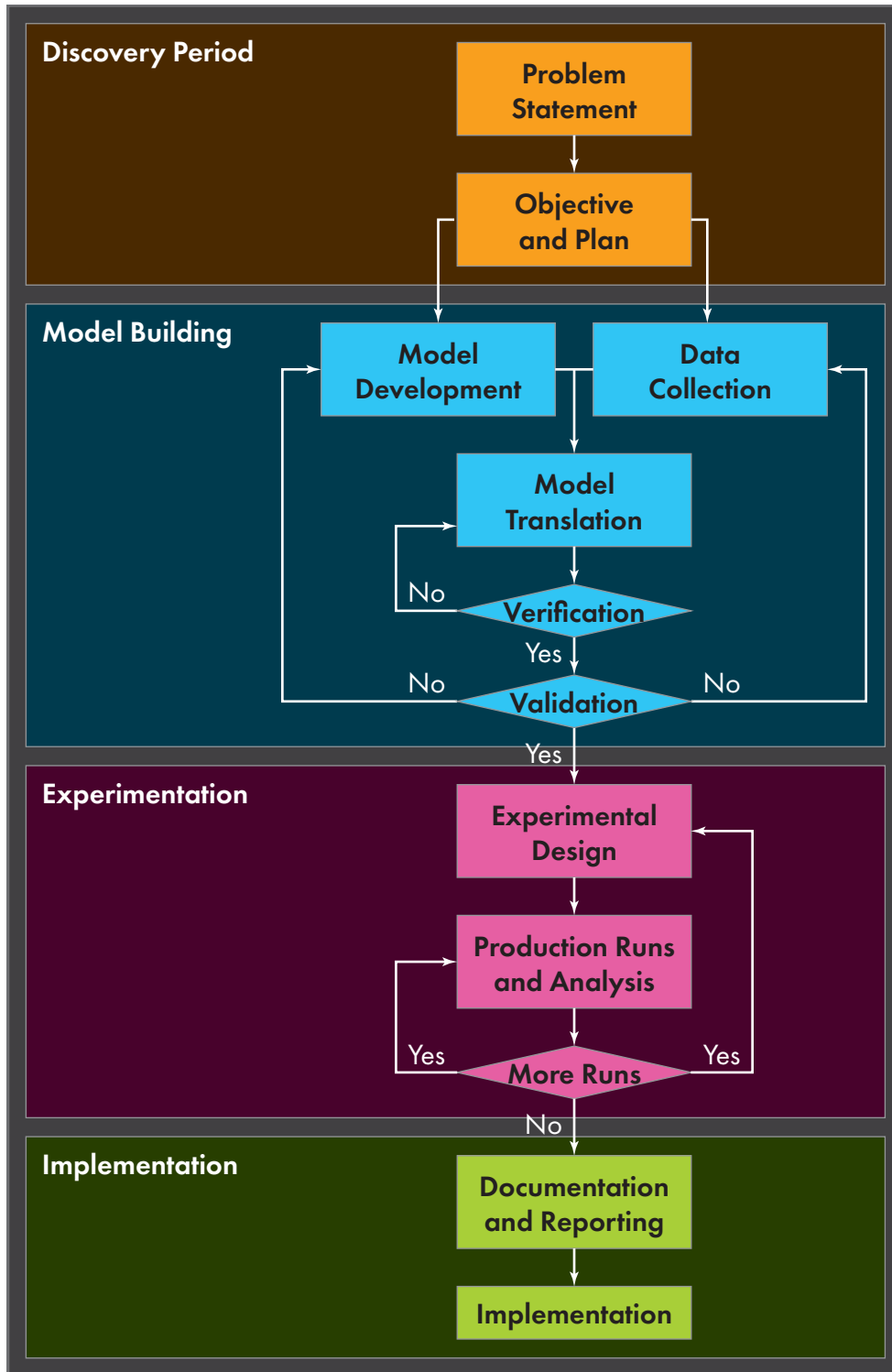


Figure 1.4.f: The process of a simulation study, based on diagram by Banks et al. (2005), redrawn by author.

of simulation is most useful, and any alternative methods that could be considered. Like any good project management, the plan also outlines the required resources, a schedule for building and testing, the estimated simulation-time, the overall cost, and the expected results. [20]

Model Building:

Model Development: The plan for how the simulation model is constructed. Like any design project, there is no instruction for building a simulation, however there are guidelines that designers can follow. The construction process first involves selecting basic elements of the problem to solve in a simple model. Over time, the accuracy of the results can be improved by adding more complexity to the initial model. Designers do not need to create an exact copy of the real system to get good results. Instead, the goal is to improve the quality until there is confidence in the outcomes for the users and context of the problem. [21] For example, a queuing simulation does not need to have correct animation of passengers walking, if it is only looking for the required number of service desks.

Data Collection: Getting information that the simulation model is based on. As the complexity of the simulation changes, the required data also changes. Data collection can take a long time, and therefore needs dedicated time early on while the model building is getting started. The amount of data that is needed for a simulation is dependant on the objectives of the project. For example, a terminal queuing simulation needs to know about the number of passengers, the amount of area for the line, and the average service time. Whereas a runway simulation needs to know the flight times, the wind direction, and the type of aircraft.

Model Translation: The process of turning a conceptual model into a computer-recognizable format. This includes what programs to use, or what code it is written in. Depending on the complexity of the simulation, a model does not necessarily need to be coded. [22] For example, curbside demand simulations can be studied using Excel spreadsheets, by keeping track of resource allocations and random variables as values in a chart. [23]

Verification: Checks if a program is working properly, as discussed in chapter 1.2. Translating a complex system into a model always creates bugs or errors, which the user needs to check and fix, within reason. If the variables and logic structure of a model correctly represent a system,

20. Banks et al. *Discrete-Event*. 13

21. Banks et al. *Discrete-Event*. 14

22. Banks et al. *Discrete-Event*. 14

23. National Academies. "Simulation Options". 8.

then it is considered verified. In some cases, using common-sense judgement is enough for verification. ^[24] For example, if simulated people are not walking through walls, then their obstacle navigation is verified.

Validation: Checks if a model matches the real-world system, through the process of calibration, as discussed in chapter 1.2. This involves repeated testing until a model has reached an acceptable level of accuracy. ^[25] For example, validation of a queuing simulation checks if the line lengths match the data collected from the real-world queue. According to the FAA, there are no specific simulations required to validate airspace, airfield, or terminal planning. ^[26] However, regardless of the tool a designer decides to use, the FAA requires a “simulation tool validation” process step, which involves calibrating the given tool to match its related real-world data. ^[27]

Experimentation:

Experimental Design: The decision about how a simulation experiment is conducted and any alternative approaches. These consider how long a simulation runs for, how many passengers are being considered, or how many trial runs are repeated. ^[28] For example, a check-in simulation can check the system for 500 passengers in one test, or 1000 passengers in another test. These alternatives can produce different information, depending on the system, which, in the case of a crowd simulation, is the result of the emergent behaviour.

Production Runs and Analysis: This is the process of reviewing the results and data of an experiment. The nature of analysis depends on the model, but can involve estimating the system’s performance using statistics, like checking variance, regression, or random sampling.

More Runs: If after an analysis of a simulated experiment, the designer must decide if the results satisfy the objectives of the project. If a simulation has not come to a conclusive result, then the designer must decide to perform more runs, or trials, and determine how the new runs will be conducted to give better results.

24. Banks et al. *Discrete-Event*. 14-15.

25. Banks et al. *Discrete-Event*. 15.

26. National Academies. “Simulation Options”. 10.

27. National Academies. “Simulation Options”. 10.

28. Banks et al. *Discrete-Event*. 15.

Implementation:

Documentation and Reporting: As with any project, good documentation is important for communicating information. With simulations, Banks et al. explain that there are two types of documentation, one for the model, and the other for the process. ^[29] Model documentation is important if designers use the simulation more than once, for multiple experiments or by other design teams. If a user changes any simulation variables, then it will affect the results of the next experiments. Documentation makes sure that users know the relationship between the input variables to the output variables. Likewise, process documentation describes the work that the designers did and the decisions they made during the experiments. Banks et al. mention that it is better practice to have more frequent reporting than one final deadline. ^[30] Finally, if any design decision is questioned further into the project, then documentation provides a history of those choices.

Implementation: Banks et al. state that the success of a simulation project depends on how well the previous steps were followed, during the creation of the model and the analysis of the system. If a designer has been part of this simulation process, and understands the fundamentals of the model, then it is more likely that the final implementation, or operation, gives valuable feedback. ^[31] Otherwise, like any project, poor design communication does not produce good simulation results, regardless of how precise the modelling is.

Crowd Simulations

A common use of these types of models is in the application of *crowd simulations*. A crowd simulation is a virtual model showing the movement, interaction, and dynamics of large numbers of entities or people. ^[32] They are useful for analyzing emergent behaviour of people in crowded areas and are commonly used to create real-time animations of groups of people in a virtual environment.

As described in model types, there is more than one way to model a system. Depending on the application, crowd simulations can model people as *fluid flow*, *particles systems*, or individual *agents*.

29. Banks et al. *Discrete-Event*. 15.

30. Banks et al. *Discrete-Event*. 15-16.

31. Banks et al. *Discrete-Event*. 16.

32. Thalmann, Daniel, and Soraia Raupp Musse. *Crowd Simulation*. Vol. 9781447144502. London: Springer London, 2008. VII-VIII.

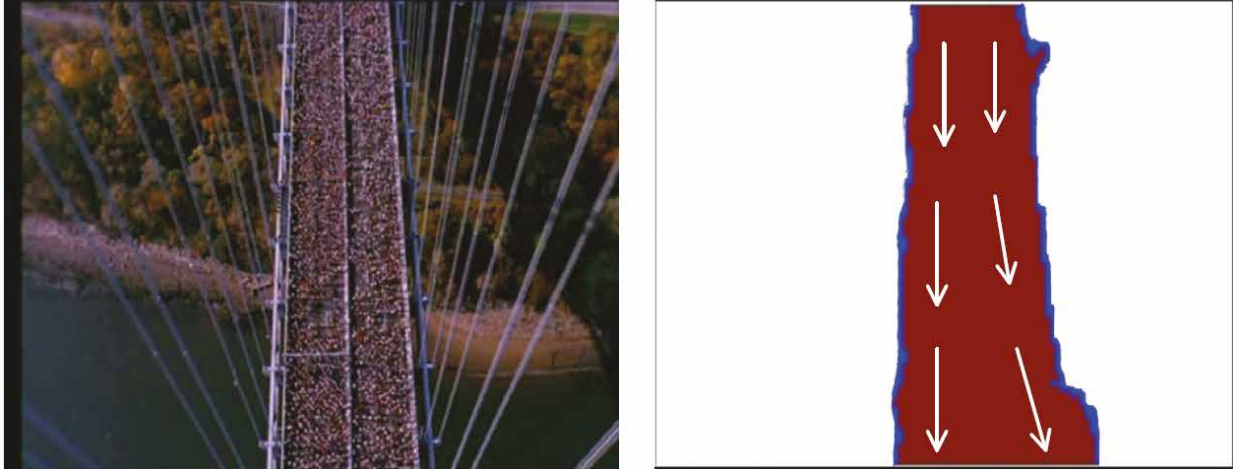


Figure 1.4.g: Dense crowds in a marathon (left) can be approximated as a fluid flow, Zhou (2010).

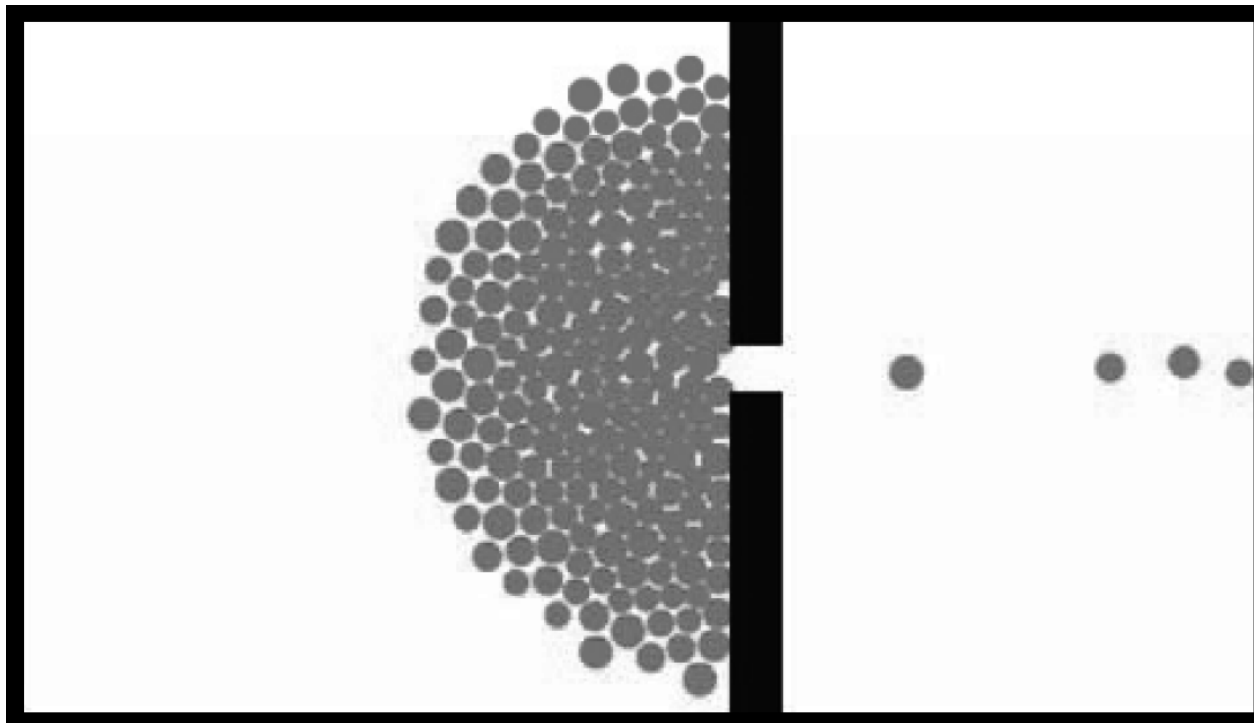


Figure 1.4.h: Crowd model using particles, Zhou (2010), image colours inverted by author for clarity.

Fluids: Fluid models use fluid dynamics to approximate crowds of people as a flow, like water or air. It considers the crowd as one entity instead of the behaviour of individual people. This is best for systems involving a high volume of people in a dense group, like an evacuation or a busy public event (Fig.1.4.g). In this case, knowing where each person is going is less important. Fluid flow simplifies the model by estimating the overall dynamics of the crowd (not that fluid dynamics is simpler to calculate, but that it ignores human behaviour by using an already established physical model). In a fluid model, if you change the density, viscosity, or velocity of the fluid, it affects the behaviour of the crowd. [33]

Particles: Particle systems model individuals as a set of identical entities. Like fluid models, particle systems use physics models to approximate a crowd's movement. However, unlike a fluid, which is continuous, particles are granular, which means they have distinct parts, (not to be confused with discrete and continuous simulations). Each particle can model one person or entity, although each entity is the same (Fig.1.4.h). Like a fluid, changing physical properties of individual particles can affect the behaviour of the crowd, like mass or applied forces. Using this approach, particle systems can realistically model typical crowd behaviour like congestion, herding, or flocking groups. [34]

Agents: Since fluid and particle models ignore the choices of individual people, they limit the ways of analysing human behaviour. Instead, simulations can use *intelligent agents* to model people. The exact nature of an agent is described in chapter 2.0, but the key characteristic of agents is that they can make their own decisions, independently, to achieve a goal. Each agent has unique characteristics and responds to their surroundings or other agents over time. Even with a simple set of rules and constraints, agents can create patterns that emerge on a global scale. [35] Agents might have cognitive, social, and emotional properties, which closely approximates real-world behaviour. For these reasons, crowd simulations are more likely to use agents to model people for transport systems and building evacuation. Whereas fluid and particle models can be more efficient for crowd visualizations, as seen in the video game Planet Coaster (Fig.1.4.i). [36]

33. Zhou, Suiping, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Low, Feng Tian, Victor Tay, Darren Ong, and Benjamin Hamilton. "Crowd Modeling and Simulation Technologies." *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 20, no. 4 (October 1, 2010): 1–35.

34. Zhou et al. "Crowd Modeling". 6.

35. Zhou et al. "Crowd Modeling". 6.

36. McCarthy, Owen. "Game Design Deep Dive: Creating Believable Crowds in Planet Coaster." Gamasutra Article, January 4, 2017.

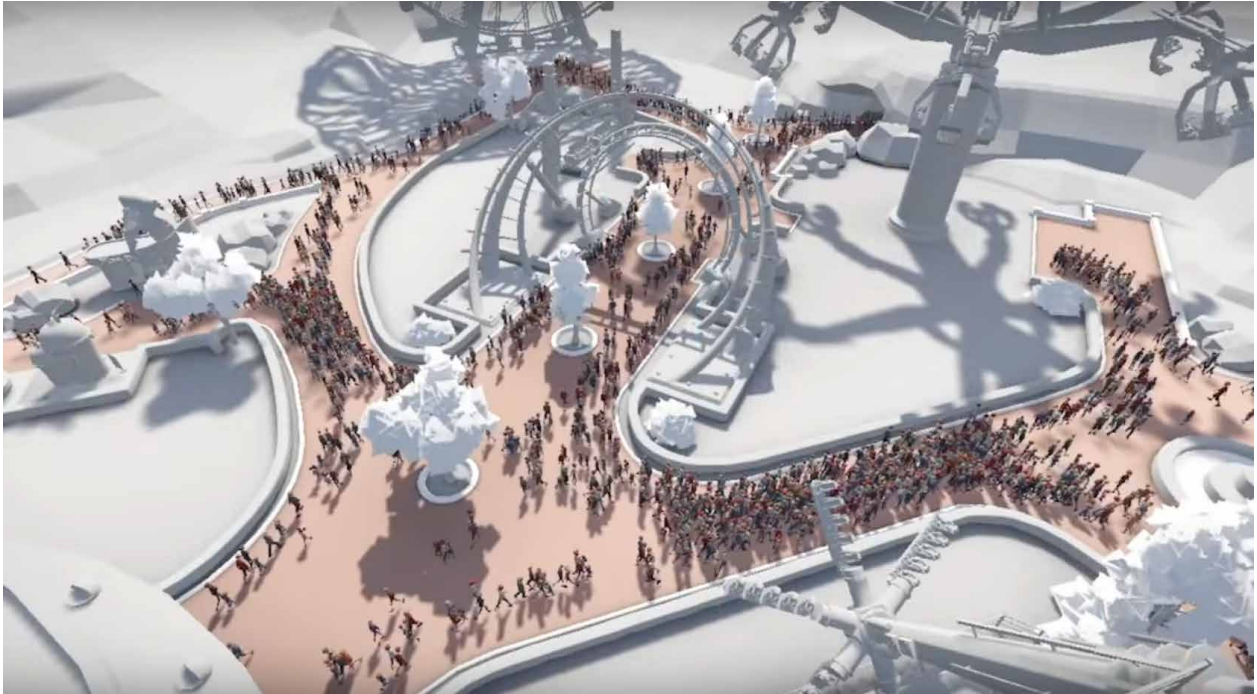


Figure 1.4.i: *The game Planet Coaster uses a fluid model to simulate crowds in an amusement park, Gamasutra (2016).*

The thesis focuses on using agents to model passengers. They provide the ability to control where people are moving, give each person a unique goal, and provide intelligent interaction with architecture, which would not be as easily controlled with fluid dynamics or particle physics.

Evacuation Modelling

Crowd simulations are commonly used in architecture to test buildings for evacuations. ^[37] In their textbook on crowd simulations, Thalmann et al. explain that architecture considers crowd behaviour during forced evacuations, in interior areas or well-defined spaces. Crowd simulations show how people exit a given area, if there is a fixed number of exits, doors, or corridors. The goal is to figure out if people can evacuate an area in a fixed amount of time. It also tries to find locations in a building which cause restrictions in the flow of people, preventing them from escaping.

Over the last decades, research in evacuation simulations have become better at quantifying the impact of human behaviour in a building. Researchers explore ways in which sociological factors influence crowd behaviour like navigation, personality, or emotions. For example, Liu et al. created an agent-based crowd model that uses perception, which demonstrates how emotion changes people's decision-making during an evacuation (Fig.1.4.j). ^[38] Similarly, Abdelhak et al. developed a crowd model that uses emotions to affect agent behaviour, suggesting how people's behaviour changes in a panic situation (Fig.1.4.k). ^[39] Additionally, the simulation created by Aschwanden et al. shows how people take different paths in public spaces because of mental stress in dense crowds. ^[40] Xiu et al. also shows how "local navigation" from people's perspective resulted in more realistic crowd behaviour, instead of using "global navigation typical" of a top-down analysis. ^[41]

https://www.gamasutra.com/view/news/288020/Game_Design_Deep_Dive_Creating_believable_crowds_in_Planet_Coaster.php.

37. Thalmann et al. *Crowd Simulation*. 5.

38. Liu, Z, Liu, T, Ma, M, Hsu, H-H, Ni, Z, Chai, Y. A perception-based emotion contagion model in crowd emergent evacuation simulation. *Comput Anim Virtual Worlds*. (2018); 29:e1817.

39. Abdelhak, Haifa; Ayesha, Aladdin; Olivier, Damien. "Cognitive Emotional Based Architecture for Crowd Simulation". *Journal of Intelligent Computing*, June 2012, 2012. Vol. 3 (2), pp. 55-66.

40. Aschwanden, Gideon, Jan Halatsch, and Gerhard Schmitt. "Crowd Simulation for Urban Planning". *Architecture in Computro* [26th eCAADe Conference Proceedings / ISBN 978-0-9541183-7-2] Antwerpen (Belgium) (17-20 September 2008): pp. 493-500.

41. Xie, Rong, and Yan Zhang. "Agent-Based Crowd Evacuation Modeling in Buildings." *Applied Mechanics and Materials* 411-414 (September 2013): 2639-42.

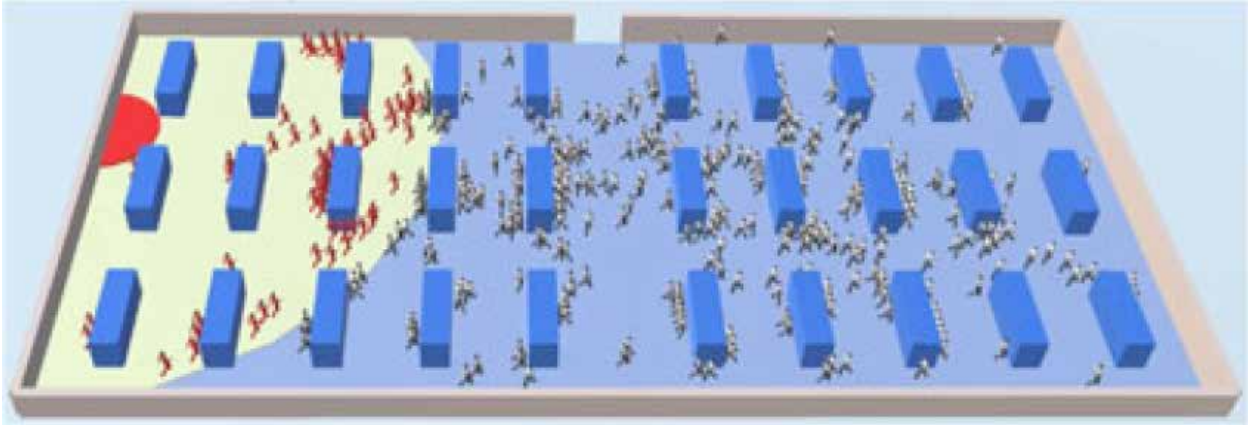


Figure 1.4.j: *Evacuation simulation that uses perception, Liu et al (2018).*

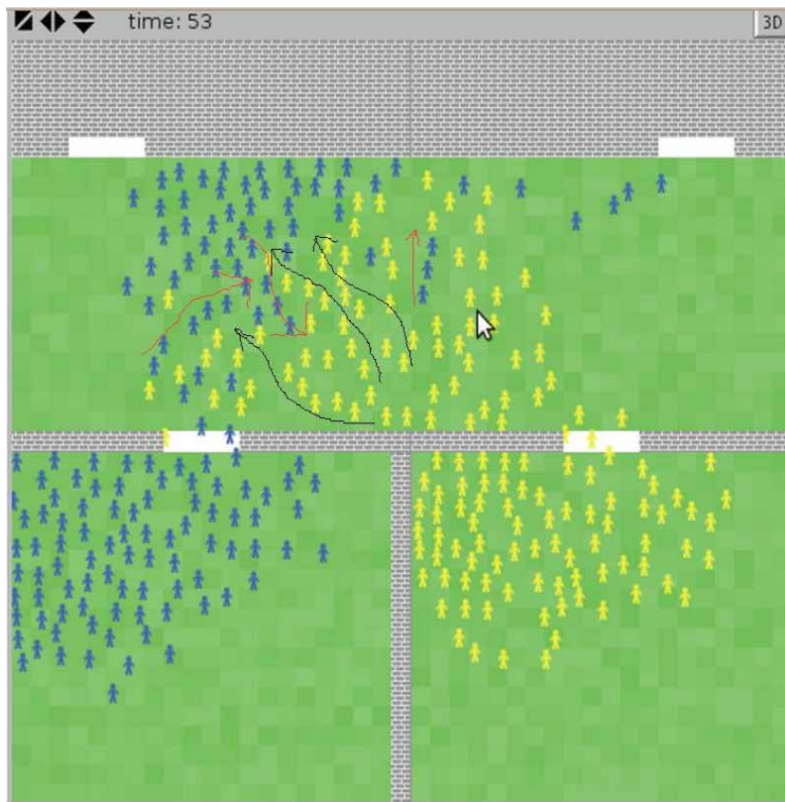


Figure 1.4.k: *Evacuation simulation that uses social forces, Abdelhak et al. (2012).*

Human emotion has a direct impact on crowd behaviour, and research like the ones above shows it is already possible to quantify these properties. Although these simulations primarily focus on building evacuation during an emergency, factors like emotion, stress, and local navigation, can apply to the daily operations of passengers in an airport as well.

Existing Software

As a starting point, the thesis investigates existing discrete-event modelling software to learn about the capabilities of current tools for airport design. The National Academies conducted a review of the current industry standards of airport simulations for the airspace, logistic systems, terminal building, and curbside (Fig.1.4.1).^[42] Out of the list of simulations for terminal modelling, the thesis selected three programs that had a good review for terminal design, and that potentially had a free trial version of the software to experiment with: MassMotion, Arena, and FlexSim. These software are also already used professionally in architectural and engineering disciplines.

MassMotion:

MassMotion is a crowd modelling software developed by Arup, under the company Oasis. It is one of the leading programs used in the architecture and civil industries. Designers use MassMotion for pedestrian modelling in evacuation testing, the design of public spaces, and transportation facilities. MassMotion models real-world environments in 3D by breaking spaces into components that are classified based on function.^[43] Some basic elements include floors, links, stairs, portals, and barriers, which represent architectural features and circulation (Fig.1.4.n). Architectural models of buildings can also be imported into MassMotion from other programs. However, these models need to be converted into MassMotion's native components to be identifiable in the simulation.

People in MassMotion are modelled as agents. Each person is given a character profile, scheduled tasks, behaviours, and goals.^[44] They know how to navigate around components marked as barriers. Agent navigation is based on a cost-system, which assigns a penalty based on deviations from the shortest path. The cost of an agent's path is based on several factors like

42. National Academies of Sciences, Engineering, and Medicine. "Simulation Options for Airport Planning". Washington, DC: The National Academies Press. (2019). <https://doi.org/10.17226/25573>.

43. Oaysis. "MassMotion Help Guide." July 2019. [<https://www.oaysis-software.com/wp-content/uploads/2019/06/MassMotion-10.0-Help-Guide.pdf>]. 15.

44. Oaysis. "MassMotion Help Guide" 19.

Simulation Tool	Scope	Licensor	License Type	Product Support/Last Release
AirTOP	Check-in, security, baggage claim, and customs	Airtopsoft	Commercial	February 2018
ArcPORT	Terminal area	Transoft Solutions	Commercial	Current
AutoMod	Baggage handling system, passenger flow	Brooks Automation/Simul8	Commercial	Current
CAST Terminal	Check-in counter, security control	Airport Research Center	Commercial	July 2017
FlexSim	Baggage handling system, passenger flow	FlexSim	Commercial	August 2017
MassMotion	Passenger terminal area	Oasys Software	Commercial	Current
PAX2SIM	Check-in counter, security control	Hub Performance	Proprietary	Current
PAXSIM	Check-in counter, baggage, security, boarding, customs	Jeppesen/Boeing	Commercial	Unknown
SimWalk Airport	Check-in counter, security control	SIMWALK	Commercial	Current

Simulation Tool	Scope	Licensor	License Type	Product Support/Last Release
Arena	Terminal area	Arena Simulation	Commercial	September 2016
ExtendSim	Terminal area	ExtendSim	Commercial	November 2017
PTV Vissim	Terminal area	PTV Group	Commercial	2017
Simio	Terminal area	Simio	Commercial	July 2017

Figure 1.4.1: List of established simulation tools for airport terminal analysis, simulations selected for this thesis are highlighted in red, National Academies (2019).

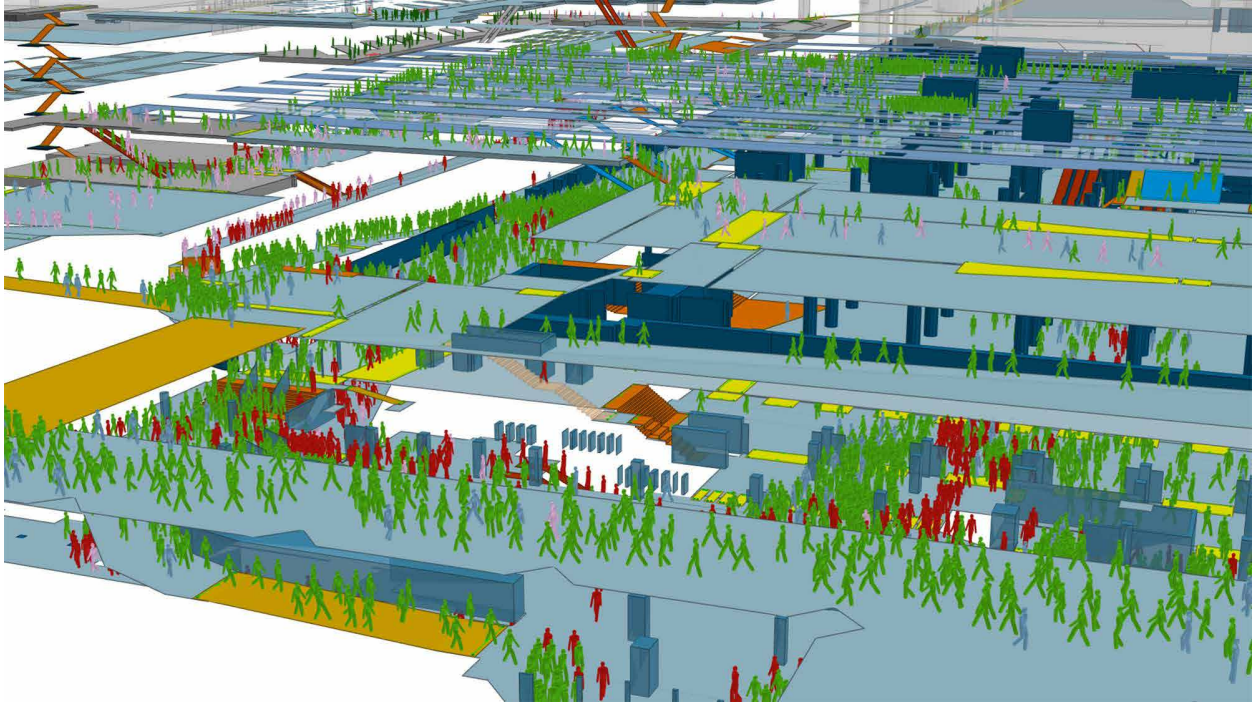


Figure 1.4.m: *MassMotion simulation of Toronto Union Station, Arup (2020).*

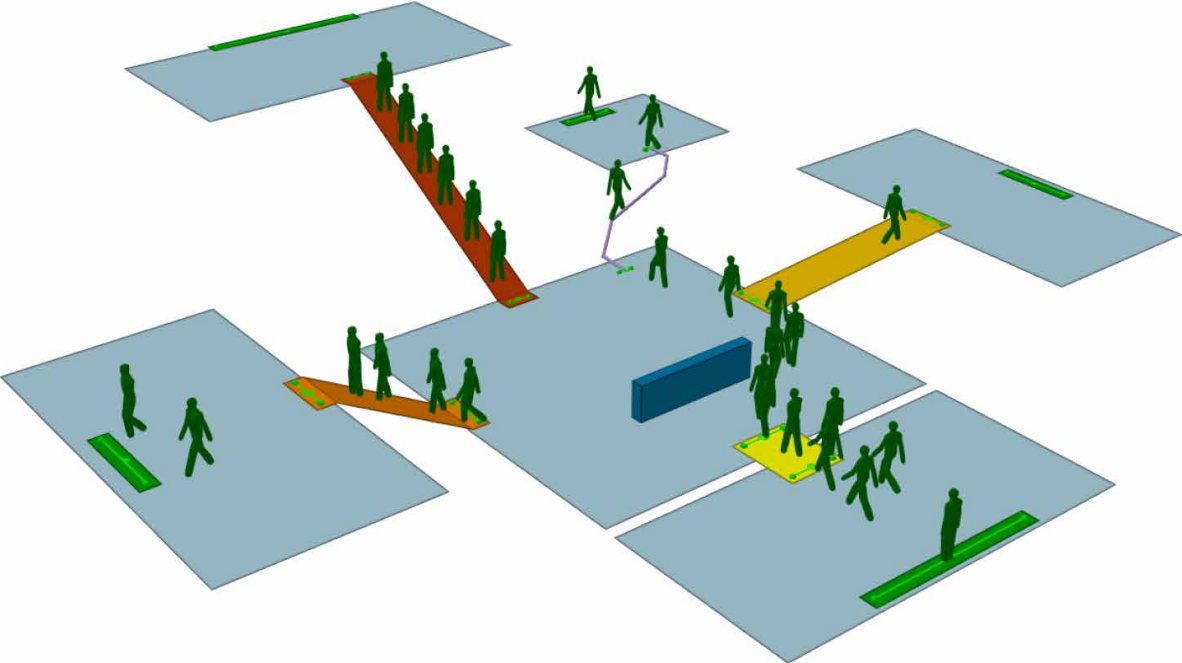


Figure 1.4.n: *Typical components in a MassMotion environment, Oasis (2019).*

the distance to the target, component weights, queues, and other agents. ^[45] While agents are walking, their movement is also influenced by similar environmental forces. Agents move towards local targets using “feelers”, or straight vectors, to judge the distance between their target and surrounding objects (Fig.1.4.o). ^[46] If neighbouring agents are within a given range, they can also adjust their velocity or target to avoid contact in a crowd.

MassMotion can show performance information within a given space, like crowd density, using depth maps and heat maps. Data is displayed on the models as a colour gradient from blue through yellow to red; where blue is low, and red is high. Heat maps, or “vision maps”, are also used to highlight what people are looking at as they walk through a space. ^[47] But there is no influence between what people see and where they walk.

MassMotion software is validated based on the International Maritime Organization (IMO) and the National Institute of Standards (NIST) Technical Note for evacuation simulations. ^[48] These standards provide 19 verification test for basic simulation components, like people walking, crowd dynamics, and emergency situations. MassMotion is also validated based on real-world evacuation tests and public spaces. Some of these tests include daily operations at Toronto Union Station, and evacuation of high-rise office towers in London and New York City. ^[49]

In an independent study, Hoy et al. used MassMotion software to test passenger congestion of Toronto Union Station (Fig.1.4.p). They showed how a 10% higher passenger density than projected capacity causes severe congestion. ^[50] This demonstrates that MassMotion can predict building spatial constraints for daily operations in addition to emergency evacuation behaviour.

Overall, as a tool for architectural validation, the thesis believes MassMotion is more than capable of quantifying architectural conditions. However, one improvement would be making agents aware of building elements during navigation. This would allow agents to use the built environment to inform their decisions, the same way the crowd forces and feelers are doing already.

45. Oaysis. “MassMotion Help Guide” 289.

46. Oaysis. “MassMotion Help Guide” 290.

47. Oaysis. “MassMotion Help Guide” 263-264.

48. Arup. “The Verification and Validation of MassMotion for Evacuation Modelling.” Ove Arup & Partners Ltd. August 10, 2015. <https://www.oaysis-software.com/wp-content/uploads/2017/11/The-Verification-and-Validation-of-MassMotion-for-Evacuation-Modelling-Report.pdf>. 2.

49. Arup. “Verification and Validation of MassMotion”. 22-23.

50. Hoy, Gregory, Erin Morrow, and Amer Shalaby. “Use of Agent-Based Crowd Simulation to Investigate the Performance of Large-Scale Intermodal Facilities: Case Study of Union Station in Toronto, Ontario, Canada.” *Transportation Research Record* 2540, no. 1 (January 2016): 20–29.

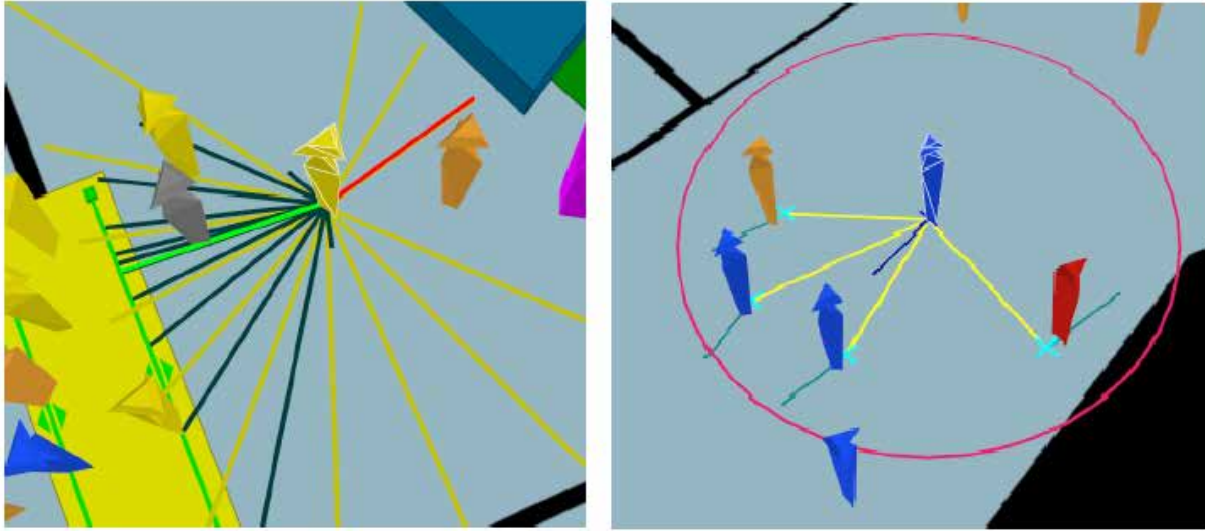


Figure 1.4.o: Agent feelers used to identify other agents and local targets for navigation, Oasys (2019).

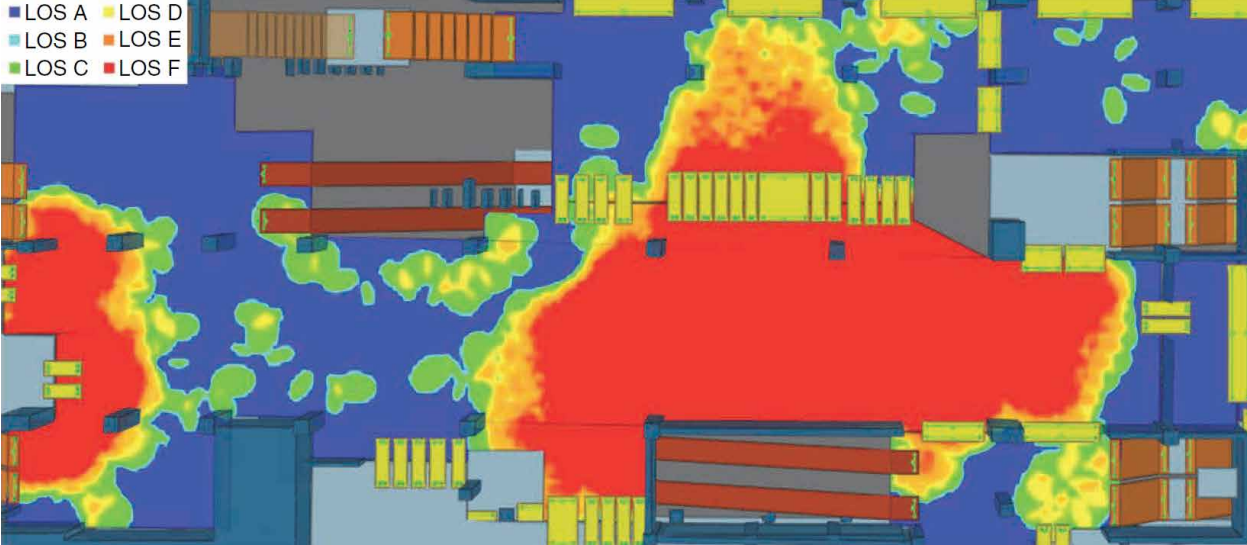


Figure 1.4.p: MassMotion displaying passenger density to show congested areas in the concourse of Toronto Union Station, Hoy et al. (2016).

Arena:

Arena is a simulation software used for modelling discrete and continuous systems. ^[51] It is primarily used in civil and industrial engineering for process analysis and modelling dynamic systems. Designers use Arena for a wide range of applications. This includes airport design, (ground operations, baggage, passenger processing), hospital design, (patient flow, emergency processing), supply-chain (logistics, storage), and manufacturing (assembly lines). Arena has two methods of modelling, first a 2D hierarchical flow-chart model, and second, an object-based 3D modelling environment. Both environments are designed for visual interaction and graphical design. It also includes statistical distributions and scheduling functions for process times.

Simulated environments are built in Arena using *modules*, which are 2D boxes and shapes that represent a process, logic, or physical object. Modules can be assigned properties and statistical data, like resource type, frequency, or process time. ^[52] Some common modules include sources, sinks, decision nodes, stations, processes, and routes. This is ideal for modelling discrete systems, like queue lines, service counters, items on conveyor belts, basic traffic flows, and other resourced-based systems (Fig.1.4.r).

Arena organizes modules based on certain templates, which provide basic features depending on the application. Modules can be joined together to create a sequence using *routes* or *links*. Resources that move between modules along these lines are called *flow items* (Fig.1.4.q). Depending on the application, flow items can be vehicles, people, items, or information.

Passengers in an airport terminal can be simulated in Arena by representing them as flow items in a process network. Arena can accurately represent passenger statistics, by modelling probability distributions based on expected process times. However, Arena cannot model free flowing crowds since people can only move along predefined paths.

Overall, Arena is strong at modelling industrial processes. It can accurately calculate waiting and transfer times for systems or logistics. The built-in process templates make it easy to set up complex networks of operations, which provides a good animation of systems over long time intervals. As a tool for architectural validation, modelling people as flow items makes it difficult to get feedback from individual behaviour in a spatial environment. Although, influences from the built environment can be identified from process time, travel time, and passenger throughput.

51. Banks et al. *Discrete-Event*. 110.

52. Banks et al. *Discrete-Event*. 110.

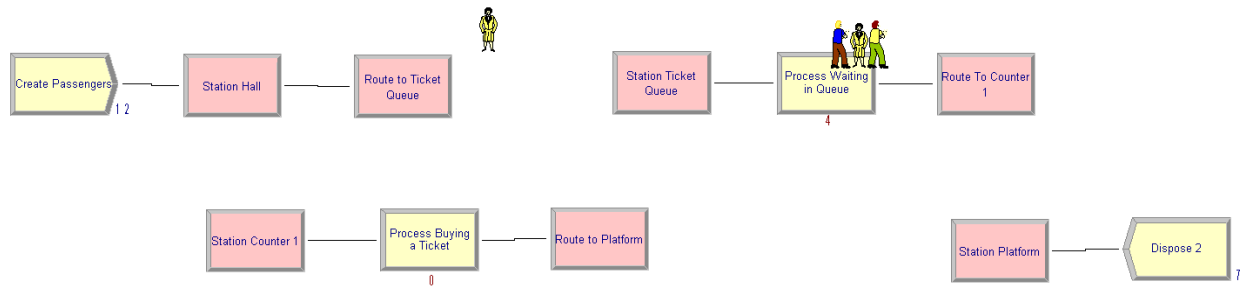


Figure 1.4.q: Flow chart of a station ticket service counter simulation in Arena. Note the passenger icons move along the flow chart during the simulation. Created by author.

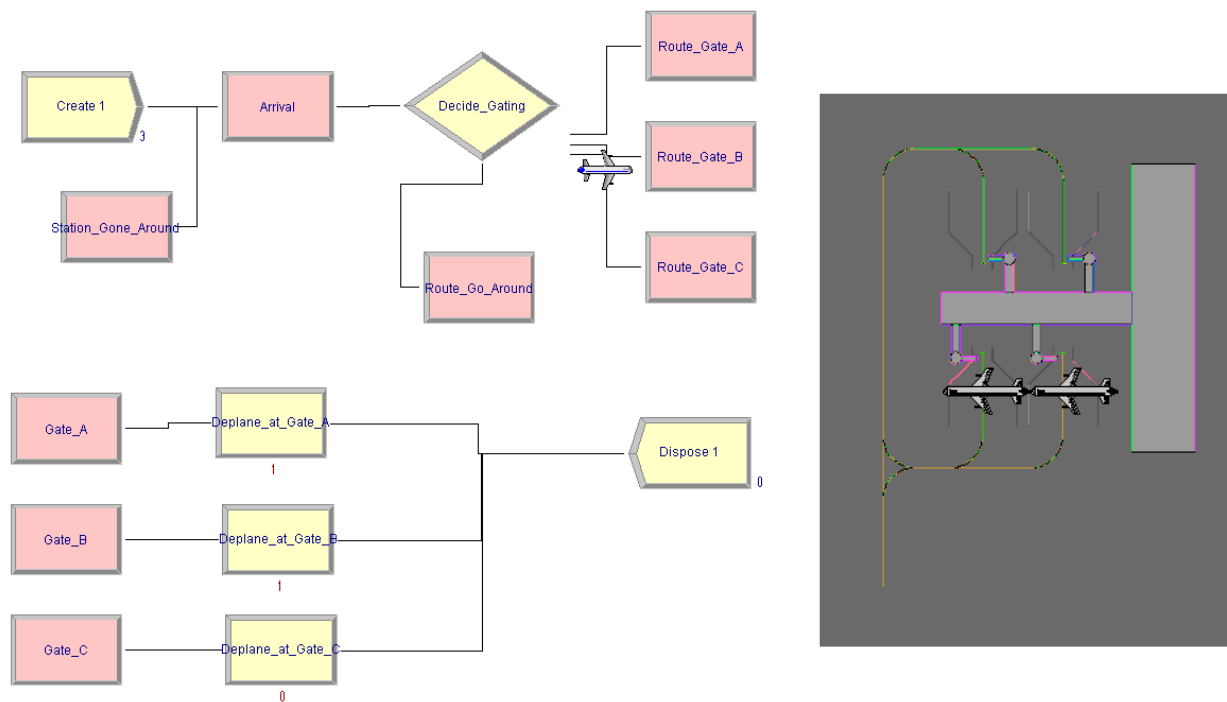


Figure 1.4.r: Flow chart of a plane gating simulation in Arena. Includes a drawing of the gates, which animated planes follow during the simulation. Created by author.

FlexSim:

FlexSim is a simulation software that can model discrete-event and continuous systems. It is primarily used in the civil and industrial industries for simulating a wide range of applications. This includes airports (passenger flows, baggage operations), healthcare operations (patient flows), manufacturing processes (material handling), logistics and transport (shipment and robotic networks). FlexSim models system behaviour using a 2D hierarchical flow-chart structure and animates the simulation with rendered objects in a 3D environment. It also provides common statistical distributions and scheduling functions for processing times.

The flow-chart structure allows users to create simulation behaviour like visual coding. It has numerous components that represent objects, actions, and resources, which can be added and connected together with graphical wires (Fig.1.4.s). FlexSim has numerous prebuilt components which are organized based on the activity or function. Some of these components include system functions, like sources, sinks, and decision nodes. There are also components for human behaviour, like walking, sitting, queuing, or interacting with other people. Most behaviour in FlexSim is resource-based. For example, if a passenger is waiting for a service counter, they must be assigned a resource *token* which is provided based on the number of available counters.

The processes defined in the flow-chart are then assigned to objects in a 3D environment, which are placed by the user (Fig.1.4.t). This includes physical objects like counters, chairs, queues, doors, machines, conveyors, or equipment. Things that interact with these objects, or that can move between them, are called flow items. Flow items can be people, vehicles, items, or logistic materials. Most objects typically follow a defined path or track. However, people can also navigate using an A* algorithm within a predefined space.

Navigation using A* provides people the shortest cost path to their target. The areas people can walk are defined by walls. Areas can also be given higher costs to influence where people walk. If people in FlexSim are using A* navigation, then they can also simulate crowd dynamics. FlexSim can also produce a heat map in a space to show where people have been walking.

Overall, FlexSim can model a wide range of industrial processes, with accurate logistics handling, timing, and movement. It has many prebuilt components that make it easy to create complex processes, both with visual coding and with realistic models in 3D environments. Having people navigate using A* helps simulate human movement in spaces, as a tool for architectural validation. Although, an improvement would be to have people navigate to local targets than to walk directly to their destination.

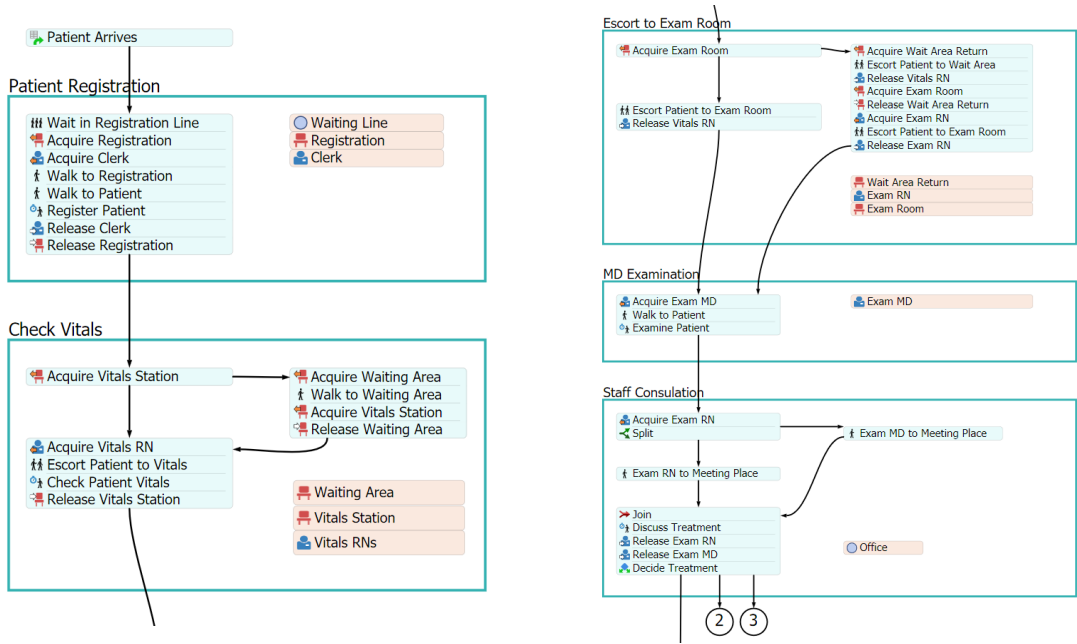


Figure 1.4.s: Flow chart of a healthcare simulation in FlexSim, based on FlexSim healthcare tutorial, recreated by author.



Figure 1.4.t: 3D model of a healthcare simulation in FlexSim, based on FlexSim healthcare tutorial, recreated by author.

Simulation Comparison

Before looking into creating a new agent simulation, the thesis initially explored using an existing discrete simulation. Out of the previous software, only Arena and FlexSim had trial software to experiment with. Therefore, MassMotion was not considered for this thesis. In addition to the existing simulation, the thesis also explored Quelea, a Grasshopper plug-in for Rhino, and Unity, a game engine. These two software have the tools necessary for making an architectural agent simulation. A description of Arena and FlexSim was already covered, so the following is a brief description for Quelea and Unity. A comparison between these four tools is illustrated in Fig.1.4.u-v.

Quelea:

Quelea is an agent-based design simulation created by Alex Fischer. It is a third-party Grasshopper plug-in for Rhino, a parametric 3D modelling software. It is used to model crowd dynamics and flocking behaviour. It works by assigning forces and behaviours to a system of agents to create interactions. ^[53] Some of these forces include following paths, attraction to other agents, obstacle avoidance, and sensing points. It also includes common crowd dynamic behaviour such as cohesion (moving together), separation, aligning (moving in the same direction), and views (having a clear view in front). There are also animal flock-like behaviour and prey-predator behaviour, such as eating and killing.

Unity:

Unity is a game engine that can create video games and other visualization applications. Although it is primarily used in the gaming industry, Unity has applications in the architectural and engineering construction industries for its ability to model and animate buildings during the design process. ^[54] Unity allows users to built custom models and behaviours. However, there are no prebuilt elements for an agent simulation. All behaviour must be coded in Unity using C# (C-Sharp) scripts. Objects are created by assigning these scripts to the models, or *game objects*. Unity can also render and animate custom models in real-time, as the simulation is running, using a built-in physics model.

53. Fischer, Alex. "Quelea - Agent-Based Design for Grasshopper." Grasshopper. Accessed December 14, 2019. <https://www.grasshopper3d.com/groups/group/show?groupUrl=quelea-agent-based-design-for-grasshopper&>.

54. Unity "Architecture, Engineering & Construction." Solutions. Accessed December 2019. <https://unity.com/solutions/architecture-engineering-construction>.





Simulation Comparison				
	 Arena	 FlexSim	 Quelea	 Unity
Scope	Processing, logistics, airports, healthcare, manufacturing	Processing, logistics, airports, healthcare, manufacturing	Parametric design, crowds, swarms	Game development, virtual visualizations
Description	Models discrete and continuous systems, 2D flow chart logic, 3D modelling, statistics, module components, flow items	Models discrete and continuous systems, 2D flow chart logic, 3D modelling, statistics, resource components, flow items, A* pathing	Plugin for Grasshopper, agent modelling, flow digram visual coding, real-time rendering in Rhino 5 model space	Game engine, 3D model space, C# script coding, physics model, real-time rendering
Access	Educational demo	Educational demo	Free complete use	Free personal use
Import Models?	Yes	Yes	Yes	Yes
Model File	3DS Max (.3ds)	3DS Max (.3ds)	Rhino 5 (.3dm)	MotionBuilder(.fbx)
Model Limit?	Yes, between 100 to 1000 agents	Yes, 30 model objects	None	None
Custom Behaviour?	Some conditional behaviours	Yes, but scripting is not in demo	Yes, using C# scripts	Yes, using C# scripts
Pros	<ul style="list-style-type: none"> - Pre-built elements for pedestrian and traffic patterns - Models of planes, vehicles, people - Animates on 2D flow chart - Create custom routes and paths - Default agent logic - Statistic analysis 	<ul style="list-style-type: none"> - Pre-built elements for pedestrian and traffic patterns - Model vehicles and people - Flow chart integrates with 3D models - Agent logic, and A* pathing - Statistic analysis - Passengers behaviours 	<ul style="list-style-type: none"> - Agents avoid obstacles and interact with others - Customize with Grasshopper, Rhino models, and Vray rendering. - No model size limit - Agent perception of space 	<ul style="list-style-type: none"> - Scripting allows custom agent and architecture behaviour - Animates in model space - Import rendered models into scenes - Built-in physics model - No model size limit
Cons	<ul style="list-style-type: none"> - Models follow predefined paths, no free roaming - Agents logic cannot be changed - Poor graphics - Disjointed 3D modelling - Limited demo size - No spatial perception or proximity to other agents 	<ul style="list-style-type: none"> - Models follow predefined paths, no free roaming - Agents logic cannot be changed - Basic graphics - Limited demo size - No spatial perception - No scripts in demo, hard to add custom behaviour 	<ul style="list-style-type: none"> - No passenger types or analysis, must add logic from scratch - Focus is on design and swarms - No moving vehicles or walking people - No statistical analysis - Computationally heavy 	<ul style="list-style-type: none"> - Must create all models and logic from scratch - No passenger behaviours - Requires strong knowledge of C# - No statistical analysis - Cannot extract data easily

Figure 1.4.u: Simulation comparison between Arena, FlexSim, Quelea, and Unity.



Figure 1.4.v: Sample simulations and station test for Arena, FlexSim, Quelea, and Unity.

Summary

A model is a representation of a system, for the purpose of understanding how the system works. Computer simulations are mathematical models, based in algebra or physics. Simulations can be described by three properties, they are either static or dynamic, deterministic or stochastic (random), and discrete or continuous. The reason to choose one property over another depends on the application and scope of a project. There can be more than one way to simulate the same situation. For example, water flow in a pipe could be a continuous dynamic simulation based in physics, or it could be a discrete stochastic simulation of pressure head.

For airports, all critical areas use simulations to understand how systems will perform. This ensures people are safe, systems work efficiently, and that the airport is profitable. The complexity of an airport simulation is defined by fidelity, or how close it matches the real world. The most common simulation techniques are spreadsheet models, queuing models, optimization, Monte Carlo, and discrete event simulations. A thorough understanding of an airport will use all these techniques at some stage of design. A complete simulation study is divided into four stages. This includes identifying the scope and goals, building and validating the model, running experiments and analysing data, and finally, documenting the results.

In architecture, a common type of model is a crowd simulation, which analyzes the movement of many people in public spaces. Crowds of people are usually modelled as a fluid flow, particle systems, individual agents, or some combination of those. Fluid and particle models use physics to approximate crowd dynamics, whereas agents assign individual people characteristics, behaviours, and goals. A common application of crowd simulations in architecture is for evacuation modelling. The purpose is to check how easily people can escape a building in an emergency so that issues with the layout can be identified. Research in crowd modelling studies how social and psychological factors affect human behaviour in buildings. Although these were explored for emergency situations, factors like emotion, stress, and local navigation can also apply to daily passengers in an airport.

Current discrete-event simulation software used in airport design include MassMotion, Arena, and FlexSim. MassMotion is more than capable of modelling human interactions in architectural conditions, with agents adapting to their surroundings. Arena and FlexSim are stronger at modelling industrial applications and are more efficient at handling linear processes. Before exploring a new simulation, the thesis experimented with Arena, FlexSim, Quelea (an

agent simulation in Grasshopper), and Unity (a game engine). For this thesis, FlexSim is the best simulation option if trying to build onto an existing software. Whereas Unity is the best simulation option if trying to build new behaviour from scratch.

Part 2:

Modelling Concepts

Part 2 goes through concepts that the thesis believes are necessary for creating an agent simulation of architectural conditions. Chapter 2.0 begins by describing what an agent is, and briefly introduces a framework for decision making. Chapter 2.1 talks about theories related to human perception, and how people learn information from their surrounding environment. Chapter 2.2 introduces existing theories in architectural spatial analysis. It also gives a brief summary of the mathematics behind graph theory. Chapter 2.3 starts with a discussion of how people value design choices based on value theory. It then introduces the method of prioritization as a way of normalizing different human perspectives. Finally, the chapter concludes with the thesis's new proposed method for quantifying architectural conditions.

Chapter 2.0

Agent-Based Modelling

An *agent-based model* is a type of mathematical model that computes patterns and interactions of individual objects within a larger system. Objects can act independently based on given rules or constraints in an environment. If these objects have a goal and can act towards that goal by interacting with the system, then these objects are intelligent *agents*. Agents differ from a standard function, which takes in an input and produces an output regardless of its surroundings. Instead, agents can conduct a set of operations, with a level of choice, to reach a target state within a set amount of time. ^[1] The purpose of agent-based models is to re-create small-scale interactions to predict how it affects a larger system. The behaviour of the larger system is unknown and typically difficult to model on its own. Instead, giving an agent a simple set of rules can result in patterns that appear across the entire system, which is called *emergence*.

Agent Properties

Agent-based models classify according to the properties of the individual agents. ^[2] Properties describe how an agent acts in an environment. The exact definition of each property depends on the context of the model. In general, agents can have the following properties: *autonomous*, *exist over time*, *reactive*, *goal oriented*, *store memory*, *adaptive*, *characteristic*, and *communicative*.

Autonomous: Agents can exist by themselves without being dependant on outside influence. They can take control over their actions and make their own decisions. Agents can also act differently to other agents in the same system or under the same set of rules.

Exist Over Time: Programs can execute actions over time either as continuous functions or discrete steps. These functions may change the environment or state of a system. However, agents within these systems can remain as a consistent entity even though the environment

1. Franklin, Stan; Graesser, Art. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents". *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, 1996.

2. Franklin et al. "Is it an Agent?". 6.

around them might be changing. While these changes are occurring, agents continue to be active and only stop being active if the program ends, or they reach their target state.

Reactive: Agents can read or see things that surround them in an environment. If an agent discovers information, they can respond to it based on predefined patterns or rules to act in a certain way. If there are any changes in the environment, an agent can respond to those changes within a given amount of time.

Goal-Oriented: By definition, agents have agency, which means they can have a purpose or a goal. Agents do not only respond to outside influences, as described in reactive behaviour. Instead, they can actively work towards a given goal or target. A purposeful agent will make choices that will get them closer to their target. This may include ignoring irrelevant information or changing their surroundings to make their target more accessible.

Store Memory: Agents can take in information given to them from the start or based on things they find in their environment over time. Like a long-term memory, the agent keeps information in a local storage which they can decide to use at any point if it becomes relevant. Agents can store memory about their goals, abilities, or the properties of the environment. Memory is also fundamental to an agent's ability to learn.

Adaptive: Adding onto memory, agents can change their behaviour over time. This is knowledge learned in the system or performing new actions that the model did not defined initially. If an agent confronts a problem, they can record what happened to avoid repeating the same actions again if they find themselves in the same situation.

Characteristic: Agents can have a character. Agents can take on a personality that is unique compared to other agents in the same system. Their behaviour and actions can be dependant on emotional states or inherited beliefs. These attributes can affect how quickly the agent moves over time or how easily they can read information in the environment. As mentioned in adaptive behaviour, their emotions and beliefs can also change as agents learn more information.

Communicative: Communication is the ability to send or receive information between other agents or entities. Agent communication is important for learning information from the environment or other agents. Additionally, communication is fundamental for creating group dynamics. This reveals how agents can join into larger units to reach goals that would not be possible if they were working individually.

Decision Making Process

Decision making in agent-based modelling refers to an agent's behavioural choices from environmental influences. The interpretation of an architectural environment is dependant on how agents process information. The thesis follows the approach used by Raubal in their model for agent-based wayfinding. They created an agent simulation that emulates human decision making for wayfinding in an unfamiliar airport terminal. Raubal explains that agents, who do not have previous knowledge of an environment, understand where they are going based on spatial cognition (the ability to read a space).^[3] Their research shows that an effective model needs to incorporate a decision-making process because agents cannot rely on previously acquired knowledge.^[4] Instead, agents only rely on short term memory of information in these situations.

Raubal discusses various models in artificial intelligence for decision making. His agent-based model developed from a process called *Sense-Plan-Act (SPA)*.^[5] This is a fundamental framework used in robotics to make sure a machine operates effectively.^[6] The general process is made up of three steps: *sensing*, *planning*, and *acting*. In a simulation environment, this approach must go through each step before an agent does anything.

Agents using the SPA framework start by sensing their surroundings for important information, like obstacles, targets, or signs. This involves using sensors or other devices to provide feedback. An agent's ability to sense information is dependant on their level of perception. In planning, agents decide how to respond to the information they just learned. Their choice is based on a given strategy or function defined in the agent's mind. Once the agent knows how they want to respond, the final step is to take action. This step involves using effectors, or other methods, to make the agent move or behave in a certain manner. Once the agent has completed their actions, that completes the decision-making process. This approach repeats as many times as necessary until the agent reaches their goal or is unable to move.

3. Raubal, Martin. "Agent-Based Simulation of Human Wayfinding: A Perceptual Model for Unfamiliar Buildings". (PhD diss., Vienna University of Technology, (October 2001): 17-29.

4. Raubal, Martin. "Agent-Based Simulation". 33.

5. Raubal, Martin. "Agent-Based Simulation". 31.

6. RobotC. "Sense Plan Act (SPA)". Natural Language Resources – VEX Cortex. Accessed November 2020. http://cdn.robotc.net/pdfs/natural-language/hp_spa.pdf.

This framework for decision making is effective because it provides a linear sequence of information that can be replicated in a computer program. [7] An approach like SPA is also well suited for discrete-event simulations because every stage of decision making occurs in finite steps.

SPA may be limiting if the agent-based model is a part of a more complex dynamic environment. An agent’s surroundings may change faster than the agent can process information. This results in agents acting against a condition that is no longer valid. Issues like this can occur in crowd simulations with a high density of people. Quick changes to other nearby people might not register in time for an agent to respond fast enough, causing unwanted collisions or pathfinding blockages. More advanced simulations like Mass Motion solve these issues by creating slow-down forces and predictive awareness. [8] This causes agents to walk slower in crowded areas so they can avoid getting stuck and anticipate where other people are walking to avoid collisions.

Every choice an agent makes using a linear process influences the next steps they come across. The thesis expects that this approach can show unexpected outcomes of poor design choices. If agents encounter an issue with a building’s design, like poor wayfinding as shown in Raubal’s research, then it impacts the agent’s behaviour further down in the simulation.

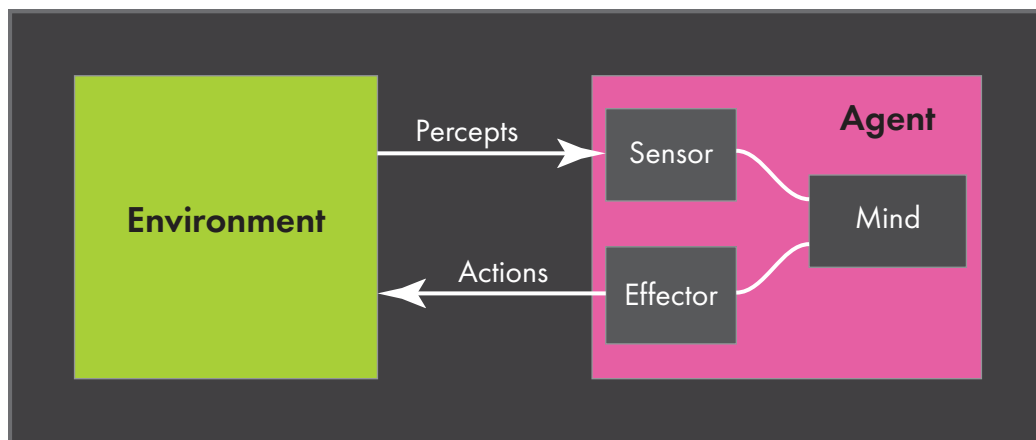


Figure 2.0.a: How agents interact with the environment, based on diagram by Liu (2020), redrawn by author.

7. Raubal, Martin. “Agent-Based Simulation”. 31-32.

8. Oaysis. “MassMotion Help Guide,” July 2019. <https://www.oaysis-software.com/wp-content/uploads/2019/06/MassMotion-10.0-Help-Guide.pdf>.

Chapter 2.1

Human Perception

One of the goals of this thesis is to use people as a function to determine the performance of an architectural layout. The proposal stems from the idea that, architecture is experienced from the perspective of individual people. The effectiveness of this approach is dependant on people being aware of their surroundings. *Perception* is the process of using the senses to take in information from the environment, interpreting it, and trying to understand what it means. Additionally, architecture is concerned with how people view spaces from their perspective. In *The Concise Townscape*, Cullen explains that people's experience of architecture is composed of a collection of *existing views* and *emerging views*.^[1] By deduction, what a person experiences, can indicate the quality of the space around them. Overall, an effective agent must be perceptive of their environment to provide feedback of architecture. They must be able to take in information, interpret it, and decide how to respond.

This thesis follows the theory presented by Raubal in their model for agent-based wayfinding. Their framework for agent perception is based on the concepts of *ontology* and *epistemology*.^[2] Ontology is the science of existence, in the context of categorization. It is concerned with understanding the existence of entities or objects, and how different concepts are grouped together. Epistemology refers to knowledge. It is concerned with the process of understanding what things are, the rationality of beliefs, and the justification of ideas. The behaviour of an agent-based model is built as a collection of decisions. Each decision depends on what is present in a digital environment, and how it is perceived by an agent.

Ontology

The agent-based model uses ontology to understand how to categorize elements in an airport.^[3] Categories can be as broad as the nature of reality, fundamental properties, or relationships. The reason categorization is important for this thesis, is to understand how a digital model can compute information. Categories give models specific definitions for various objects and

1. Cullen, Gordon. *The Concise Townscape*. Abingdon: Routledge, 1971. 9.

2. Raubal, Martin. "Agent-Based Simulation of Human Wayfinding: A Perceptual Model for Unfamiliar Buildings". (PhD diss., Vienna University of Technology, October 2001). 62.

3. Raubal. "Agent-Based Simulation". 63.

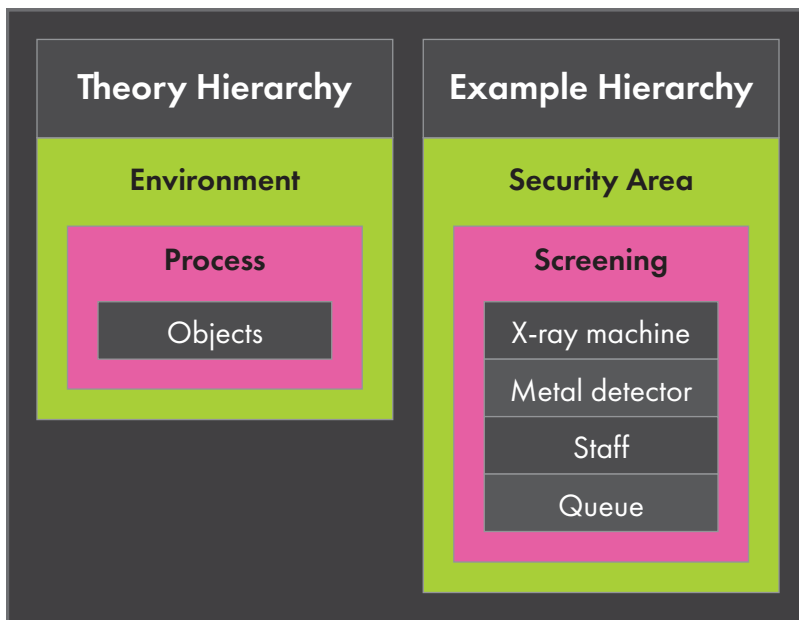


Figure 2.1.a: *Environments are made up of objects and processes.*

conditions. There needs to be defined boundaries so an agent can identify what something is, and how to respond in different situations. For example, agents need to understand where they can check-in their baggage, or how to identify which gate number their plane is boarding from. In further chapters, the thesis will explain how digital perception can model imprecise information, which would not naturally fit into a binary system. However, the first step is to define basic elements in the agent-based model.

Ontology is rooted in theory and philosophy. In *Objects in Their Environment*, Smith introduces how ontology can describe the physical environment of ordinary people. ^[4] They use ontology to organize an environment into Aristotelian *substances* and *accidents*, based on people's behaviour. Substances refer to objects, things, and people, whereas accidents refer to qualities, actions, and processes. ^[5] Using Smith's concepts, Raubal, in his agent-model, explains how the built environment is made from the relationship between substances (objects) and their corresponding accidents (processes). For example, the environment of airport security is made up of objects, (queue lines and x-ray machines), and processes, (waiting and screening), as is illustrated in Fig.2.1.a. Queue lines serve the need for waiting, and x-ray machines serve the need for screening.

Smith defines a list of conditions that describe spatial properties of substances and accidents. These conditions are called *ontological marks*, which describe how human interactions relate to a physical environment. ^[6] For example, the action of checking in baggage requires the time and space for passengers to move bags from one area to another. Smith explains that environments like this, take up physical space, can be divided into parts, and exist over time. ^[7] Raubal's research brings Smith's conditions into an architectural context by relating each of these attributes to wayfinding in an airport. ^[8] For the feedback of airport architecture, the thesis uses the same approach to understand the ontological marks of a terminal environment, which is detailed in Fig.2.1.b. This explains how people exist within an airport terminal based on fundamental conditions. In other words, an environment exists, from a person's perception, if it is defined by relevant objects and processes.

4. Smith, Barry. "Objects and Their Environment". *The Life and Motion of Socio-Economic (GISDATA 8)*, London: Taylor and Francis, 2001, 79–97. As cited in: Raubal. "Agent-Based Simulation". 62.

5. Smith. "Objects and Their Environment". 81.

6. Smith. "Objects and Their Environment". 91.

7. Smith. "Objects and Their Environment". 91-92.

8. Raubal. "Agent-Based Simulation". 64.

Ontological Marks of an Environment	Ontological Examples in Airports
Environments contains substances and accidents, which need a "participant" substance to exist.	Passengers are involved in airport process, like check-in, screening, waiting, boarding.
Environments remain consistent, but can be defined by different participant substances at different times.	Passengers remain the same, but can be checked-in or have security clearance at different times.
Environments are part of a natural process, and are proportional in size with other things.	Crowd dynamics exhibit emergent behaviour based on patterns in an airport environment.
Environments have complete boundaries, where objects can be inside or outside it.	Terminals are defined by architectural components; people occupy inside, and planes park outside.
Environments have parts that are environments, themselves.	The text on a sign is understood in context of wayfinding; the gate number (e.g. C3, F7) refers to a physical location.
Environments are spatially connected, part of a larger entity, but can be physically scattered.	Restaurants exist with in a larger retail space, and are scattered throughout the terminal.
Environments take up space, and can be divided into smaller spatial or temporal segments.	A terminal pier takes up space, which can be divided into individual aircraft stalls, or flight occupancy time.
Environments exist over time, but do need be identical from beginning to end.	A 1 year old terminal is the same terminal when it is 10 years old, but renovations can change areas inside.
Environments exist over time, but do not need to be continuous.	A security area is always present, but may not service people during off-hours.
Environments do not have punctual existance, (distinct beginning and end), but contain punctual events	The act of waiting is expected in terminals, but passenger have a specific depature time.

Figure 2.1.b: *Smith's (2001) ontological marks of an environment and examples in an airport, based on diagram by Raubal (2001), reinterpreted by authour.*

The physical nature of objects in an environment can be further categorized based on how people observe each object. When a passenger reads a departure board, they are seeing coloured lights from a metal box. The nature of that box affects how people perceive flight information from the sign. The influence of these factors is presented by Gibson in *The Ecological Approach to Visual Perception*. This describes how agents and objects in their surroundings can be categorized based on their physical properties.

Gibson defines an environment in terms of three parts: *substances*, *medium*, and *surfaces*.^[9] Like ontological substances, Gibson's substances refer to physical matter and solid objects. These are usually opaque to light and are composed of heterogenous materials.^[10] Things like the earth, minerals, plants, and animal matter, are examples of substances. Medium, in contrast, refers to fluids and light. These are usually transparent to light, primarily homogeneous, and can propagate waves.^[11] Water, air, gases, the atmosphere, and light are examples of mediums. Substances usually exist within a medium and use it to move around: like fish in water, or engines consuming air and fuel. Finally, surfaces are the boundary layers between substances and a medium, which separate them from each other. The interaction of substances and mediums occur through surfaces. All substances have a surface. They can absorb or reflect light, giving substances their appearance and making them identifiable. Surfaces relate to aesthetic properties, like texture and colour. Although, surfaces can also influence friction, durability, or viscosity of a substance or medium.^[12] The grain of wood, the "fluffiness" of clouds, and the coarseness of rocks on a riverbank are examples of surfaces.

For this thesis, airport elements are classified according to substances and mediums, as illustrated in Fig.2.1.c. Substances are divided into living and non-living objects.^[13] In the context of a simulation, these are agents and environmental objects, respectively. Agents in a simulation refer to any object that has agency, awareness, and knowledge. People are substances and they are aware of their surroundings. Therefore, passengers and airport staff would be considered agents.

The environment can also be organized into physical and non-physical elements. The thesis organizes architectural conditions by dividing the environment into objects (substances) and

9. Gibson, James J. *The Ecological Approach to Visual Perception*. New York: Psychology Press, 1986. 16.

10. Gibson. *Visual Perception*. 19-18.

11. Gibson. *Visual Perception*. 16-17.

12. Gibson. *Visual Perception*. 24-26.

13. Raubal. "Agent-Based Simulation". 66.

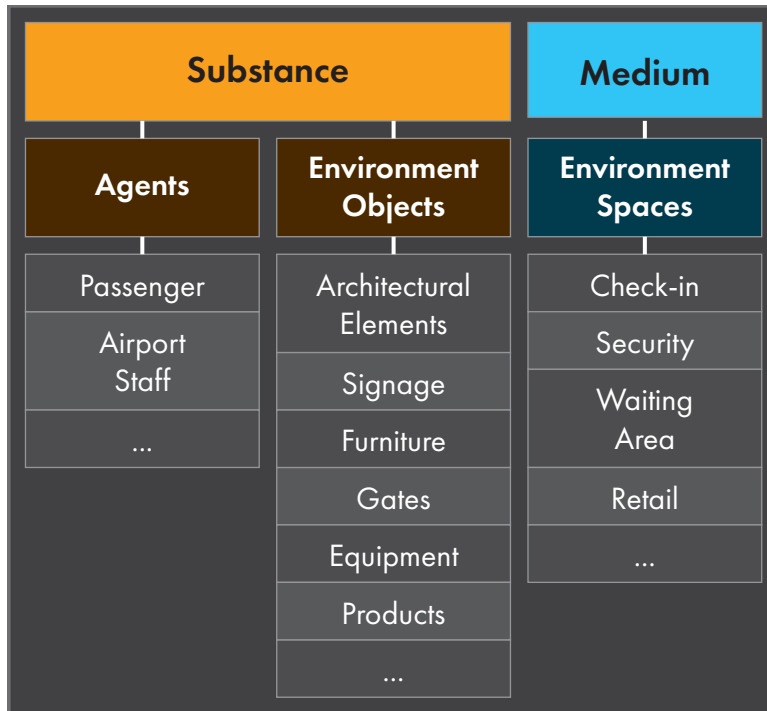


Figure 2.1.c: Elements of an airport classified as a substance or medium, based on diagram by Raubal (2001), redrawn by author.

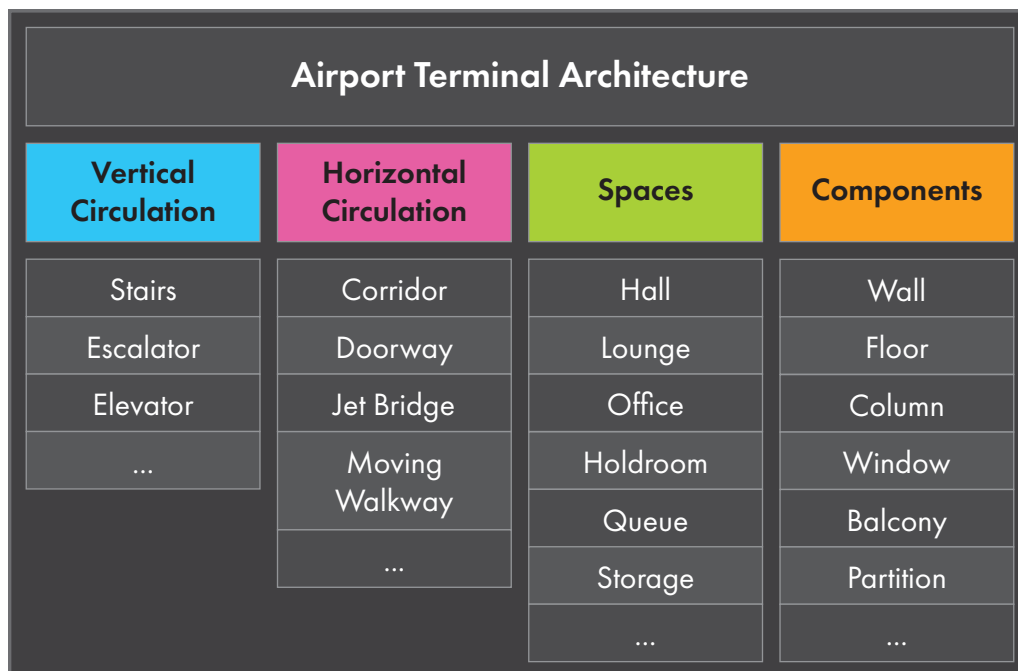


Figure 2.1.d: Further categorization of airport terminal architecture.

spaces (mediums). Environmental objects include architectural components, like walls, doors, columns, or windows. Further investigations might also address surfaces, which describe the texture and colour of these architectural elements. Architectural spaces are categorized as a medium. People occupy spaces and use them to move around the built environment. Spaces can propagate waves of light, or even waves of people in a crowd. In the context of an airport, architectural spaces are simply areas like security, holdroom concourse, or food and retail.

Fundamentally, an architectural space as an ontological environment, can only exist physically, from a person's perception, when defined by architectural objects related to a given process. As mentioned, for example, passengers can perceive the environment of a security area, when it contains objects like x-ray machines or metal detectors that they recognized to be relevant for the security screening process.

Epistemology

Up to this point, the thesis has described how certain objects in an airport correspond to specific processes and behaviours. But how do passengers know which objects correspond to which behaviours? The fact that people might see a security line, and understand that they must wait behind other people, is a result of their existing knowledge of what a queue is and how it works. Knowledge is the understanding of things, objects, or concepts. For agents to give feedback of an architectural environment, they must have knowledge of the areas they are interacting with. They must understand what their goal is, and what elements in an environment will allow them to reach their target. Agents who encounter difficulties in a building can provide feedback of architectural conditions, if they understand what they are perceiving.

The thesis's understanding of knowledge also develops from Raubal's agent-based model for wayfinding. Raubal uses Gibson's ecological approach as a foundation for agent knowledge in combination with *cognition*.^[14] The framework stems from the concept of *affordances* as a way of understanding how people learn information about the world. Gibson defines affordances as the properties of an environment that have use for people, either for good or bad.^[15] In the context of substances, affordances are the layout of surfaces and substances that show properties of use to an observer.^[16] Surfaces absorb and reflect light, which Gibson states, represent what they afford. Affordances are meant to be a relationship between surfaces,

14. Raubal. "Agent-Based Simulation". 39-47.

15. Gibson. *Visual Perception*. 127.

16. Raubal. "Agent-Based Simulation". 67.

substances, and the people who observe them. ^[17] They are not inherent properties of an object, but they are properties relative to a single person.

For example, as Raubal explains, a staircase affords people to climb it because the height of the steps is an affordance for climbing. ^[18] More specifically, the height of each step, relative to the size of a person's leg, is at the right level for someone to lift their feet above each step. If steps on a staircase are too high, relative to the size of a person, then they would not be able to use it. Likewise, a chair affords people to sit because the height of the seat, relative to the size of a person's legs, is the right level to bend down onto it. The chair also affords support, which people know can lift their own body weight. ^[19] The same is true of other objects that people sit on, like ledges or curbs, which are not designed for sitting. Instead, people recognized that the height of the ledge is at the right level to support their weight, which they will use if they are tired.

As Raubal summarizes, Gibson's theory of affordances only considers perception (what people can see), but does not consider the thought process behind choices, or cognition. ^[20] This describes how information goes from what people observe to their actions. It also considers if this process is always consistent or if there are any errors in decision making. Gibson's theory of affordances also does not consider how people perceive things without looking at the environment, such as memory. ^[21]

In *The Design of Everyday Things*, Norman suggests that affordances are the result of mental interpretations of things, which are influenced by people's past experiences and memory. ^[22] He states that when people perceive things, their minds have to process that information before people can take action. ^[23] Additionally, Raubal adds that social context, cultural background, and personal values will influence how people process this information. ^[24] As a result, affordances are not a full representation of the environment, but only represent a subjective view from each person.

17. Raubal. "Agent-Based Simulation". 41.

18. Raubal. "Agent-Based Simulation". 41.

19. Norman, Donald A. *The Design of Everyday Things*. New York: Basic Books, A Member of the Perseus Books Group, 2013. 11.

20. Raubal. "Agent-Based Simulation". 42.

21. Raubal. "Agent-Based Simulation". 42.

22. Raubal. "Agent-Based Simulation". 43.

23. Norman. *The Design of Everyday Things*. 12.

24. Raubal. "Agent-Based Simulation". 43.

According to Norman, the physical properties of an object that communicates affordance information to people are called *signifiers*.^[25] For example, a door has an affordance for people to walk through. However, a signifier of the door's use would be the handles, hinges, or a sign describing which way to push/pull.^[26] In this context, affordances are the possible interactions between people and the door. The actions that people take are then determined by their memory of how a door works and the signifiers that communicate how the door should be used.

In the context of an airport, Raubal developed a list of possible affordances that an adult traveller might do while going to their gate (Fig.2.1.e). Raubal states that affordances can correspond to three different categories, *physical*, *social*, and *mental*.^[27] In addition to Raubal's wayfinding, the thesis considers this list in the context of architectural spaces.

Physical Affordances: This relates to physical properties of an object and how people can physically interact with them. For example, people can place objects on horizontal surfaces (like counters and tables) or hold small objects in their hands (like tickets and drink cups).^[28] This also includes how people can open doors, walk through corridors, climb stairs, and sit in chairs. For architecture, the physical affordance of walls, partitions, and columns can divide spaces, block views, and direct movement.

Social Affordances: This describes how people act according to social contexts, or institutional rules. Even if something is physically possible, there are interactions that are considered socially unacceptable, morally wrong, or illegal.^[29] For example, this includes showing a passport to a customs officer, staying within the stanchions of a queue, not walking into restricted areas, and proper etiquette when communicating with other people. Social affordances can also be triggered based on physical properties. The physical appearance of a customs checkpoint can give a sense of authority (using barriers or signs) and trigger people's memory of how security is handled. In contrast, retail and concourse areas can also signal to people the affordance of socializing, consumption, or relaxing.

Mental Affordances: This represents how people make decisions, which can relate to social affordances.^[30] For example, a flight departure board can trigger people to remember their

25. Norman. *The Design of Everyday Things*. 13.

26. Norman. *The Design of Everyday Things*. 16.

27. Raubal. "Agent-Based Simulation". 67.

28. Raubal. "Agent-Based Simulation". 67.

29. Raubal. "Agent-Based Simulation". 68.

30. Raubal. "Agent-Based Simulation". 68-69.

		Affordances		
Substances		Physical	Social	Mental
Architecture	Space	move through, access, leave, enclose, stand	look around, include, spend time, wander	look for, wait, expect
	Doorway	enter, go through, put through	look through, separate	remember gate
	Column	go around, move towards	block, divide	remember reference
	Corridor	move along, branch, curve, begin, end	direct	remember path, select
	Stairs	go up, go down, stand	wait	pay attention
Airport	Signage	go towards, stand out, eye-catching	advertise, direct, inform, follow, wayfinding	look for, recognize, read, check
	Check-in Counter	go to, stand in front, put ticket, get pass	line up, check-in,	look for, remember flight
	Passport Control	go to or through, enter	block, line up, show passport and pass	look for, remember documents
	Airport Staff	approach	talk to, ask, provide info or documents, behave	look for, remember info
	Decision Point	pass, turn, orient	look around, wait	decide, search, select

Figure 2.1.e: *Category of affordances in an airport and architectural elements, based on diagram by Raubal (2001), redrawn by author.*

flight number, gate, and departure time. Additionally, this accounts for deciding what food to eat in a food court, what things to buy in a store, or where to sit in a waiting area.

When a person interacts with an environment or object, they will typically involve all three types of affordances. For example, a check-in counter requires passengers to walk through the check-in area (physical), wait in line behind others (social), remember their flight information (mental), place documents on the counter (physical) and communicate with airport staff (social).

Memory

Memory is a part of a person's mind that stores, processes, and retrieves information as needed. In a simulated environment, agents need to remember what their goals are, and the actions needed to achieve them. Likewise, passengers in an airport need to remember where they are and the processes they have already completed, like checking in. Memory is also an important part of agent learning, since information gathered from past experiences can inform decisions for future events.

To create a simulated method of how people gather information, the thesis considers Norman's *approximate models* (for memory). Norman explains how parts of human memory can be approximated using simplified models. Although these models are not scientifically accurate, they can still replicate the outcomes of using memory in the real world. ^[31] The thesis's conceptual model of agent memory is built-up of two parts, a *long-term* and a *short-term* memory.

Long-Term Memory:

Long-term memory holds information that stays with the agent from the start to the end of the agent's existence. Agents use long-term memory to store information about their goal or *primary target*, and the agent's properties or *characteristics*.

Primary Target: In an airport simulation, the goal for departing passengers is to board their flight. Since agents must navigate using their surroundings, they should be unaware of their target location until they find it. Therefore, an agent's long-term memory only stores the name of the target, like a gate number (Gate B24). However, once the agent observes the location of their gate, its physical location is also stored in their memory.

31. Norman. *The Design of Everyday Things*. 101.

Characteristics: The characteristics of an agent can describe the agent’s core beliefs and ideas, which are not expected to change over the course of the simulation. Since they represent a person, this can also include their name, gender, age, or other defining properties. As an agent learns new information from the environment, their resulting actions are based on their stored thoughts and beliefs.

Short-Term Memory:

During the agent’s journey, short-term memory only holds information temporarily, or as the agent needs it. In this thesis, short-term memory stores information about *agent state*, *spatial memory*, and *local targets*.

Norman simplifies short-term memory with the idea of having *memory slots*.^[32] Every time a new piece of information is observed, the information fills one of these slots. Once all the slots are full, any new piece of information replaces the oldest memory. When an agent first enters the environment, their short-term memory would be empty. As they experience different parts of the environment throughout the simulation, their short-term memory fills with relevant information.

Agent State: Agent states represent an agent’s current thoughts about how to behave. Raubal explains that, when an agent makes an observation, they recognize some state of the environment at a specific place and time.^[33] For example, if an agent observes the entrance to a security area, then the agent will be more attentive than if they were simply waiting by the gates. These states represent different types of perceived affordances or interactions.^[34] Essentially, the type of state represents what type of behaviour the agent is following. This includes airport behaviour like waiting, queuing, processing, checking in, or security screening. An agent’s state will change every time they observe new information, which, as Raubal states, is comparable to short-term memory.^[35]

Spatial Memory: Like agent state, spatial memory describes the agent’s knowledge of where they are. When an agent recognizes a property in an environment, they can identify the space associated with that property. For example, if an agent recognizes a check-in counter, then the agent learns that the counter affords picking up a boarding pass. As a result, the agent knows

32. Norman. *The Design of Everyday Things*. 102.

33. Raubal. “Agent-Based Simulation”. 75.

34. Raubal. “Agent-Based Simulation”. 75.

35. Raubal. “Agent-Based Simulation”. 75.

they must be in the check-in area. Therefore, their behaviour, or state, should align with the expected behaviour of that space. An agent has a memory of where they are until they observe new spatial information from the environment.

Local Targets: Local targets describe the possible affordances associated with an object or location that an agent can currently observe. When an agent observes an object, they learn information about the object or the associated affordances. If the object has relevant information that is useful for the agent, then they will memorize the given affordances or actions related to the object. ^[36] For example, after noticing a wayfinding sign, an agent might read the sign to understand what it says. If the sign has information on it that matches where the agent is trying to go, then the sign has an affordance for following. ^[37] As a result, the agent memorizes the direction the sign is pointing, so they can follow it. Fundamentally, the direction the sign points to is considered a local target. Once the local target is used, or consumed, then the information may be replaced, or forgotten, from the agent's short-term memory.

Field of View

Gibson defines the field of view for human beings as the solid edge of ambient light that can be registered by a person's optical system, or eyes. ^[38] The volume between these solid edges of light can be represented by a section of a sphere. The sphere is defined geometrically by the angle seen from the perspective of a person's eyes in their head. ^[39] People's view of the world is based on the direction their eyes are looking and the posture of their head. ^[40]

Gibson explains that people see the world in perspective from a single point of view. This is approximated by projecting lines out radially from a person's eye (Fig.2.1.f). ^[41] If light reflected off surfaces reaches someone's eyes, then those surfaces, or objects, are considered visible. Since light travels in straight lines through spacetime, light coming from behind objects will not be detected. As a result, these objects are not visible.

A field of view describes visual perception, which are things that a person is aware of. Things that are not in a person's field of view are not seen, and as a result, are not visually perceived. Philosophically, if something is not observed, then it does not exist from the perspective of a

36. Raubal. "Agent-Based Simulation". 75.

37. Raubal. "Agent-Based Simulation". 75.

38. Gibson. *Visual Perception*. 127.

39. Gibson. *Visual Perception*. 127.

40. Gibson. *Visual Perception*. 196.

41. Gibson. *Visual Perception*. 196.

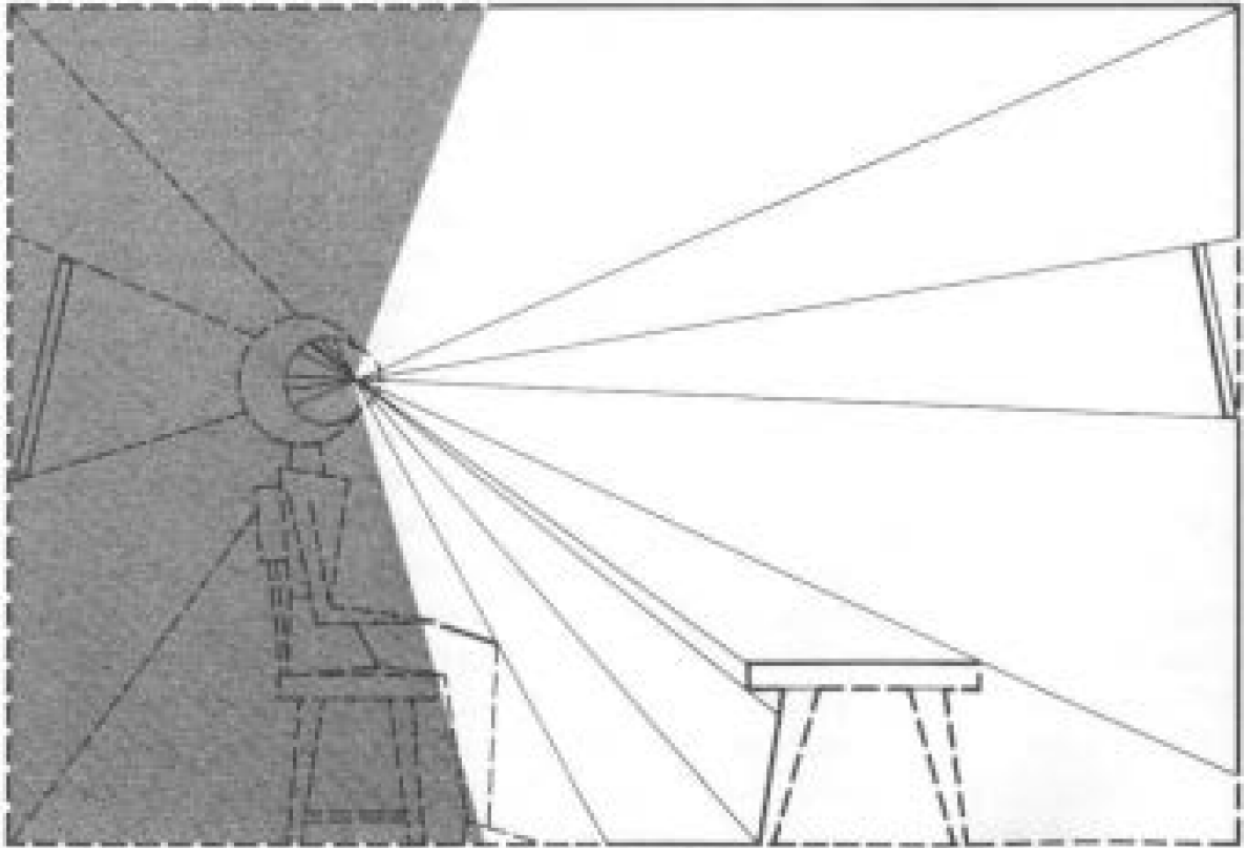


Figure 2.1.f: Gibson's representation of a person's field of view. Every surface that can be visually perceived is a solid line. Otherwise, it is a dashed line if it cannot be perceived, Gibson (1972).

person. For example, students standing in the halls outside of their classroom cannot see their desks behind the walls of the school or a closed door. For this reason, their desks do not technically exist, based on their visual perception. Once students enter the classroom, the desks become visible in their field of view. As a result, they can perceive their physical existence. Although students may expect their desks to be in the classroom before arriving, the physical objects are not in their field of view yet. If the desks are removed from the classroom before the students arrive, then it is no different than if they cannot see the desks behind the walls. Students would also not be aware that there are no desks until they observe an empty space.

Fundamentally, this metaphor also applies to passengers in an airport terminal. If people are navigating through the building and they cannot see their gate, then they are not aware of its physical existence. Only after the location of the gate area becomes visible in a person's field of view, are they able to visually perceive the physical gate. The same condition applies to any architectural feature. For example, if a designer adds windows to view outside the terminal concourse, but no one can see it from their field of view, then the windows do not physically exist from passengers' visual perception of the space.

Additionally, as described by Norman, if there are no signifiers to communicate an object's utility, then the object's affordances do not exist from a given observer. ^[42] Likewise, if an architectural feature is not in a person's field of view, then they cannot understand the affordances of that space. Therefore, the field of view illustrates the elements, surfaces, or objects that an individual can perceive.

Summary

Perception is the process of using the senses to interpret information from the environment. People's experience of architecture is made up of a collection of views. So, what a person experiences can indicate the quality of those spaces. Likewise, an effective agent must be perceptive of their environment to provide feedback of architectural conditions. The thesis follows the work of Raubal in their research of perceptive agent wayfinding. Agent perception is based on understanding how people identify different objects in their mind (categorization or ontology) and understanding how people know what things are (knowledge or epistemology). From ontology, the thesis learns that an environment exists, from a person's perception, if it is defined by relevant objects and processes, which are called ontological marks. Additionally, these objects can be categorized based on how people observe them, which Gibson defines as

42. Norman. *The Design of Everyday Things*. 14.

substances (solids), mediums (fluids and light), and surfaces (layers between substances and medium). Passengers are living substances, and architectural elements are non-living substances. Architectural spaces that people and objects occupy are the medium. Aesthetic and material properties are the surfaces. Fundamentally, an architectural space as an ontological environment can only exist physically, from a person's perception, when objects are related to a given process within those spaces.

For agents to identify what things are, they must have knowledge of the objects they are looking at. Agent-based knowledge develops from affordances, which models how people learn information about the world. Affordances are the relationships between objects (Gibson's surfaces and substances) that have use for people, either for good or bad. Even though affordances are inherent to the object, they are relative to each individual making the observation. The properties of an object that communicates what they can afford to an agent are called signifiers. When people perceive a signifier, they must process the information they see before taking action, which is described by cognition. An agent's social, cultural, and personal values influence how this information is processed. Therefore, affordances are only a subjective view of the environment. Affordances can be physical interaction with the object, social norms associated with the object, or mental processes that are brought about by the object.

Memory is responsible for storing, processing, and retrieving perceived information. This is simulated in an agent using simplified models for both long- and short-term memory. Long-term memory holds an agent's goal (primary target), and their inherent beliefs (characteristics). Short-term memory holds behaviour tied to a given affordance (agent states), the agent's current location (spatial memory), and actions associated with a given affordance (local targets).

Field of view is the optical area that a person can observe, based on the reflected path of light. Objects or elements that fall within someone's field of view can be visually perceived. If an element is not visible, then the person cannot perceive it, which is equivalent to the element not existing from their perspective. If an element is not perceived, then a person cannot understand what it affords.

Chapter 2.2

Spatial Analysis

Spatial analysis is the study of shape, position, size, quantity, or location of geometric features. It describes how information is organized in topology, geography, and architectural spaces. For this thesis, spatial analysis can help connect the behaviour of people and the quality of spaces around them.

Methods

This section covers some of the existing methods in architecture for analysing spatial conditions. These methods include *isovist*, *space syntax*, *axial maps*, *enclosure*, and *visibility graphs*.

Isovist:

One of the foundations of architectural spatial analysis is the isovist. An isovist describes the area that can be seen from a single point, projecting out in every direction (Fig.2.2.a). This can be illustrated in a 2D floor plan, or it can also describe a volume in 3D space. One of the first papers on the isovist was by Benedikt in 1979, who demonstrated how an isovist can analyze architectural and urban spaces. Benedikt explains that spaces are understood as a collection of visible surfaces, which are framed by architectural components like walls or windows. ^[1] They used the isovist to represent visible space from a single point, which made it easy to compare different quality of spaces based on a given area and perimeter. ^[2] An isovist can also be thought of as the volume of space illuminated by light coming from a single point. ^[3]

1. Do, Ellen Yi-Luen, and Mark D. Gross. "Tools for Visual and Spatial Analysis of CAD Models." *CAAD Futures* 1997, 1997, 189–202. https://doi.org/10.1007/978-94-011-5576-2_15. 3.

2. Do et al. "Spatial Analysis of CAD Models." 3.

3. Arabacioglu, Burcin Cem. "Using Fuzzy Inference System for Architectural Space Analysis." *Applied Soft Computing* 10, no. 3 (2010): 927.

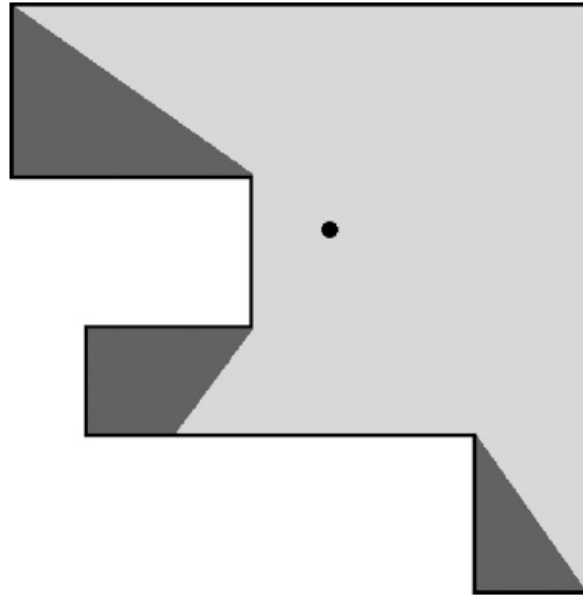


Figure 2.2.a: *An isovist is the area that can be seen from a single point, light grey is visible and dark grey is hidden, Arabacioglu (2010).*

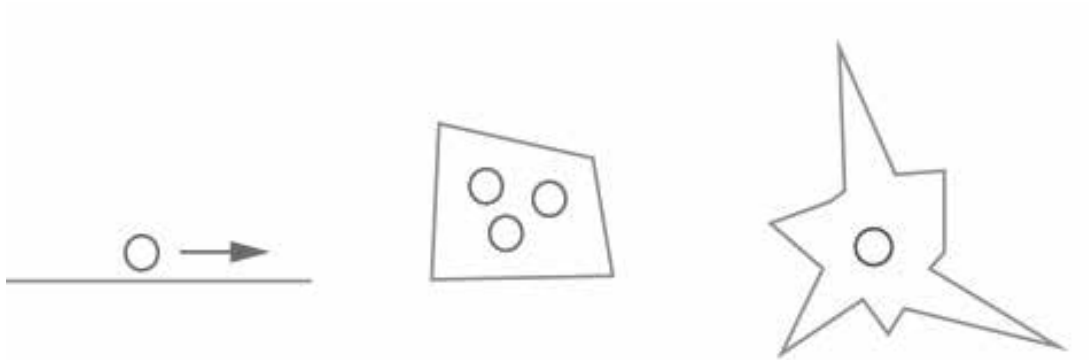


Figure 2.2.b: *The three geometries of space syntax: lines, convex space, and an isovist, Vaughan (2007).*

Space Syntax:

Space syntax is the study of how spaces are organized in urban conditions. This was first published by Hillier and Hanson in their book *The Social Logic of Space* (1984).^[4] They studied configurations of spaces in relation to social structures in an urban context. Space syntax analyses the nature of a built environment based on patterns related to human behaviour, like where people live or how they travel. Although, Vaughan explains how space syntax suggests that spaces have their own formal logic before a social context is applied.^[5] Hillier and Hanson also established methods for describing spatial configurations as physical geometries. The three most fundamental geometries they used were *lines*, *convex space*, and the *isovist* (Fig.2.2.b).^[6] Lines describes the path of people's movements. A convex space is a location that is visible from every other point in that space, which is a property of interactive spaces. Finally, the isovist was used to represent a person's perspective or their field of view.^[7]

Axial Maps:

Space syntax breaks down urban spaces into components to understand how they are connected to each other. One of the ways these connections were studied was using *axial maps*. These maps were a type of graph structure that simplified spaces and their connections into vertices and edges, respectively. Edges can describe how people physically move between spaces, or they can represent concepts like functional, social, or environmental relationships. It simplifies an analysis without having to consider architectural dimensions like walls or doors. In Fig.2.2.c, Vaughan shows how axial maps illustrate different connections between spaces, depending on what room they look from.^[8] In a city fabric, axial maps are useful at describing the paths that people take and the spaces they pass through. In addition to physical movement, axial maps can model social structures based on a network of human behaviour.^[9]

4. Hillier, Bill, and Julienne Hanson. *The Social Logic of Space*. Cambridge: Cambridge University Press, 1984. doi:10.1017/CBO9780511597237.

5. Vaughan, Laura. "The Spatial Syntax of Urban Segregation." *Progress in Planning* 67, no. 3 (2007) 208.

6. Vaughan, Laura. "Urban Segregation." 209.

7. Vaughan, Laura. "Urban Segregation." 209.

8. Vaughan, Laura. "The Spatial Syntax of Urban Segregation." *Progress in Planning* 67, no. 3 (2007). 211.

9. Vaughan, Laura. "Urban Segregation." 208.

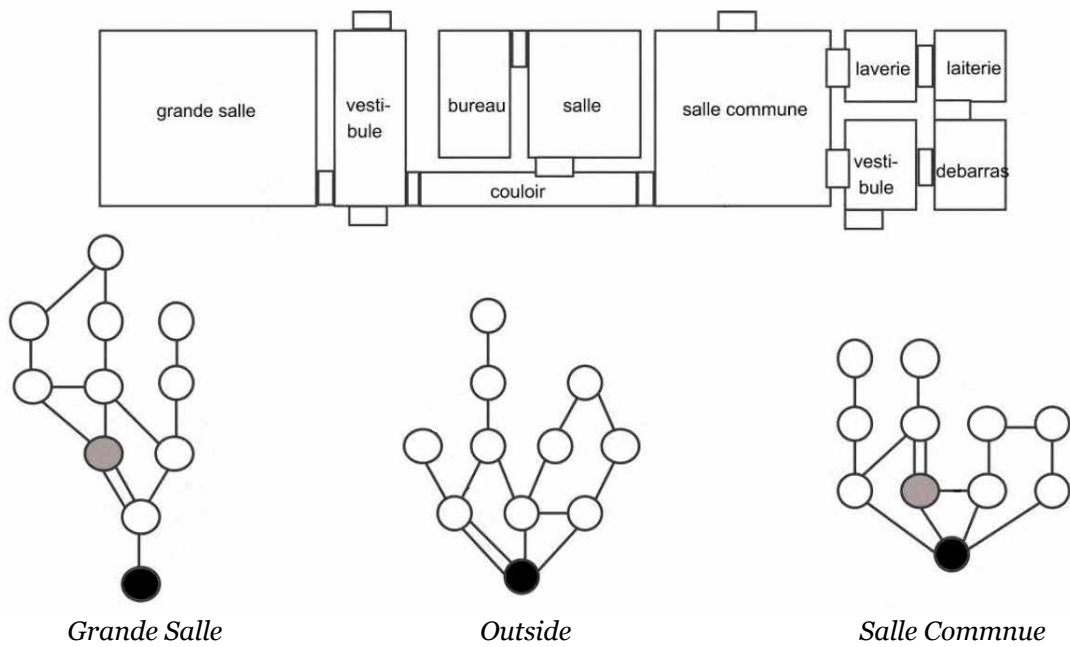


Figure 2.2.c: Graphs showing the arrangement of connections from different rooms in a house, Vaughan (2007), re-highlighted by author.

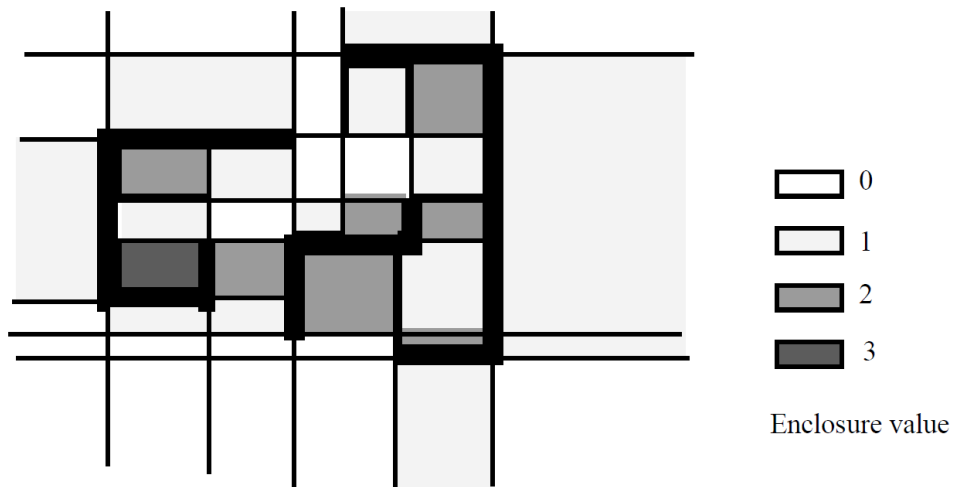


Figure 2.2.d: Enclosure defines a value based on the number of surrounding walls on a scale of 1 to 4, Do et al. (1997).

Enclosure:

Enclosure is a concept published by Gross in 1977 who created a computer program for subdividing spaces in a floor plan. The goal was to create a numerical model of how people feel in different spatial arrangements. ^[10] This was calculated by assigning a score based on the number of walls that surround a given area. For example, a square room that is closed on all four sides would be given a value of 4, whereas a room with no walls would be given a zero (Fig.2.2.d). The value of enclosure does not only apply to a single room but could also be generalized to an arbitrary unit of area. This works by subdividing a floor plan into small enough segments, or increasing the *granularity*, which was like an early concept of *depth mapping*.

Visibility Graphs:

Visibility graphs were popularized by Turner et al. in 1999 for analyzing architectural spaces. It combines the logic of the isovist geometry and the mathematics of graph theory to understand how different space structures affect social function. ^[11] A basic visibility graph works by identifying the locations that can be seen from a single point. If a point in space is visible from more than one location, then that point has higher visibility. This is equivalent to overlapping multiple isovist geometries and adding the areas where they intersect with each other. ^[12] However, Turner et al. demonstrates that, instead of using isovists, a space can be approximated as a network of vertices in a graph, where the edges are the visible connection (Fig.2.2.e). ^[13] The resolution of the analysis depends on the size and frequency of the vertices. In addition to representing visibility of a person at eye level, a visibility graph can be created for any height or dimension, which can also represent volumetric space. A visualization of a visibility graph may also be referred to as a *depth map*, which illustrates visibility information in a pixel grid, typically, on a scale from black to white (Fig.2.2.f).

10. Do et al. "Spatial Analysis of CAD Models." 4.

11. Turner, Alasdair, Maria Doxa, David Osullivan, and Alan Penn. "From Isovists to Visibility Graphs: A Methodology for the Analysis of Architectural Space." *Environment and Planning B: Planning and Design* 28, no. 1 (2001): 104.

12. Turner et al. "From Isovists to Visibility Graphs." 107.

13. Turner et al. "From Isovists to Visibility Graphs." 107.

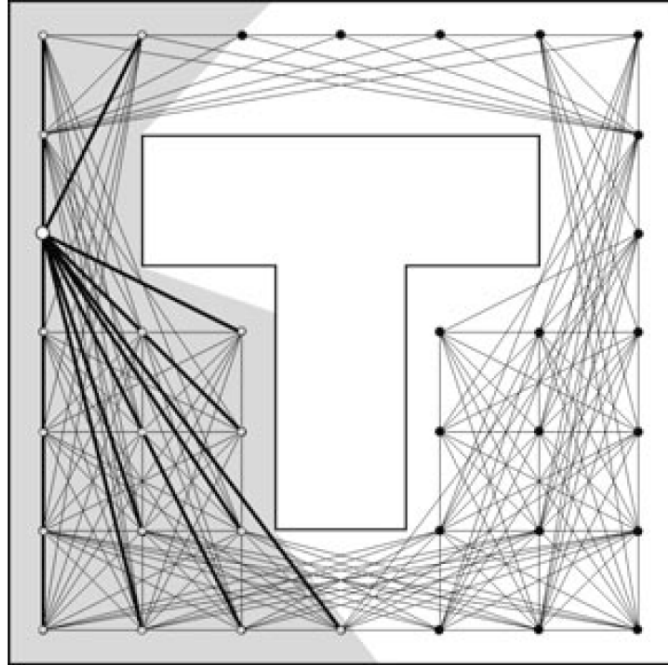


Figure 2.2.e: A visibility graph, where each vertex is a point in space, and the lines are the visible connections, Turner et al. (1999).

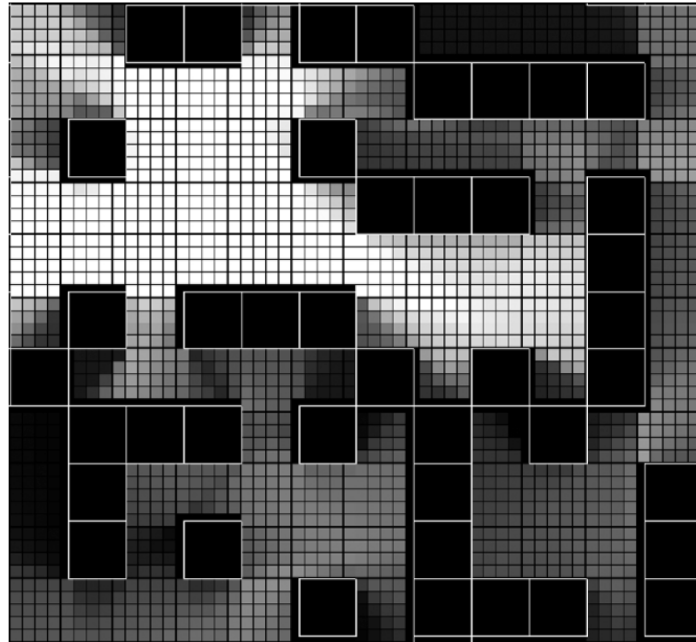


Figure 2.2.f: A depth map illustrating the number of connections in a visibility graph as a pixel grid. White: most connections (high visibility), Black: least connections (low visibility), Arabacioglu (2010).

Graph Theory

Graph theory is the study of mathematical structures called *graphs*. These are models that can represent networks or relationships between items. ^[14] Graph structures are built up of points and connected by lines (Fig.2.2.g), which are referred to as *vertices* and *edges*.

Graph theory is an important concept for two areas in this thesis: understanding architectural arrangements and agent navigation. As mentioned for axial maps, graph theory is useful in architecture because it diagrams how spaces are connected together. Additionally, simulations commonly use graph theory to simplify agent navigation. Graph structures provide a way for simulations to differentiate between accessible or non-accessible areas, and it can be used to calculate the shortest path between two points.

Graphs are not geometrically defined, as visualized in Fig.2.2.g. Instead, graphs are abstractly written as a collection of connected pairs of vertices. ^[15] An edge connected between two points, v_1 and v_2 , can be denoted mathematically as the *ordered pair* $\{v_1, v_2\}$. ^[16] A graph can be written in terms of the number of vertices as a set $V = \{v_1, v_2, v_3, \dots, v_i\}$ or edges as a set $E = \{e_1, e_2, e_3, \dots, e_i\}$. The edges in Fig.2.2.g can be defined as $\{v_1, v_2\}, \{v_2, v_3\}, \{v_2, v_4\}, \{v_3, v_4\}, \{v_3, v_5\}, \{v_4, v_5\}$.

Directed Graphs

In Fig.2.2.g, $\{v_1, v_2\}$ and $\{v_2, v_1\}$ describe the same edge. This means the relationship between v_1 and v_2 is the same in both directions. Since all the edges have this property, the graph is considered *undirected*. ^[17] By contrast, a graph is considered *directed* if the relationship between two vertices does not go in both directions. As seen in Fig.2.2.h, the edge between v_1 and v_2 only has a relationship in the direction $\{v_1, v_2\}$. If this graph was representing roads between cities, then a directed edge would be equivalent to a one-way road, where cars can only travel in one direction.

14. Guichard, David. *An Introduction to Combinatorics and Graph Theory*. Whitman College. January 30, 2020, 7.

15. Guichard. *Combinatorics and Graph Theory*. 7.

16. "Describing Graphs." Computer Science, Algorithms, Graph Representation. Khan Academy. Accessed May 2020. <https://www.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/describing-graphs>.

17. Khan Academy. "Describing Graphs".

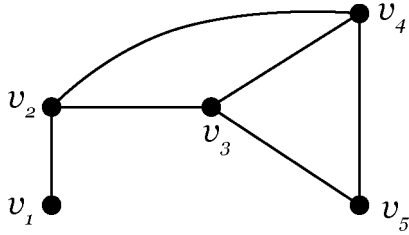


Figure 2.2.g: An undirected graph with 5 edges and 5 vertices.

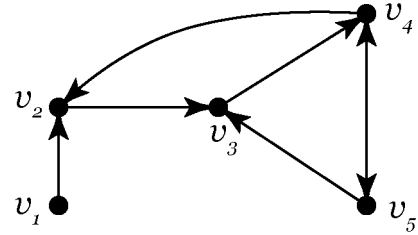


Figure 2.2.h: A directed graph.

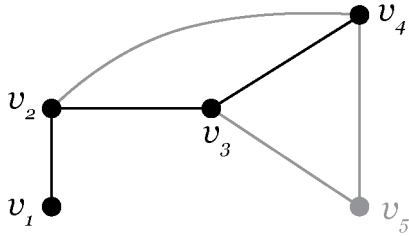


Figure 2.2.i: A path from node v_1 to node v_4 .

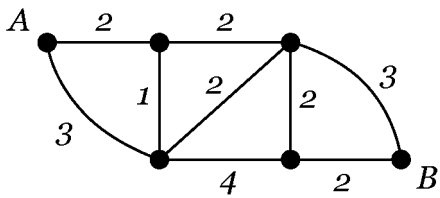


Figure 2.2.j: A weighted graph.

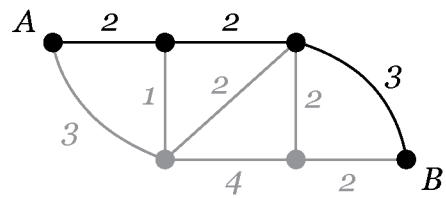


Figure 2.2.k: Lowest cost path between node A and node B has a cost of 7.

Paths

Another attribute of graphs is called a *path*. Paths are made up of a collection of continuous edges and vertices. Any vertices along a path are referred to as *nodes* in that path. The movement from one node to another is called a *walk*.^[18] This is useful for describing how information might travel across a graph. In Fig.2.2.i, there exists a path between v_1 and v_4 . The vertices v_1, v_2, v_3 and v_4 are all nodes in that path, which forms a walk in the sequence $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$.

Weights

In the same way that a directed graph can change the relationship between two vertices, another approach is to introduce *weights*. A graph is said to be *weighted* if its edges or vertices are given a non-negative value.^[19] A weight describes how much *cost* it takes to travel from one vertex to another (Fig.2.2.j). The shortest path in a weighted graph is referred to as the *lowest cost path*. The lowest cost path is the sum of the weights of all the edges (or vertices).^[20] As highlighted in Fig.2.2.k, the lowest cost path from node A to node B follows the top edges. These edges have the weights $w_1 = 2, w_2 = 2, w_3 = 3$, giving it a total cost of $w_T = 7$.

An example of a weighted graph is a model of roadways between cities, where weighting represents the total travel time. A road connecting two cities that are farther apart would be given a higher cost than a road in between two cities that are closer. Likewise, weights can be used to represent traffic. A road with a lot of traffic can be given a higher cost than a road with very little traffic. This concept is useful in crowd simulations for representing crowded spaces. Areas in a building that have a lot of people can be assigned higher cost, compared to areas that have less people. Agents in these simulations would prefer to follow the path with less people, which is represented by the path with the lower cost.

Navigation

In most applications, agents do not have a straight line to their target. Instead, agents need to be able to move around obstacles or other objects. Navigation is the process of searching for a path from one location to another. Graphs can subdivide areas into distinct nodes, which navigation algorithms use to calculate a path.

18. Wilson, Robin J. *Introduction to Graph Theory 4th ed.* Harlow: Longman, 1996. 3-4.

19. Wilson. *Graph Theory.* 39.

20. Guichard. *Combinatorics and Graph Theory.* 106.

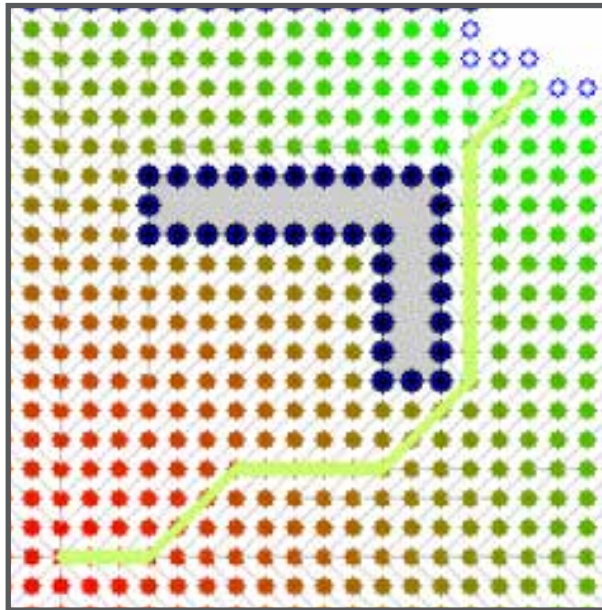


Figure 2.2.l: Pathfinding solved using Dijkstra's search algorithm checks all nodes for the lowest cost, *Bhattacharya (2011).*

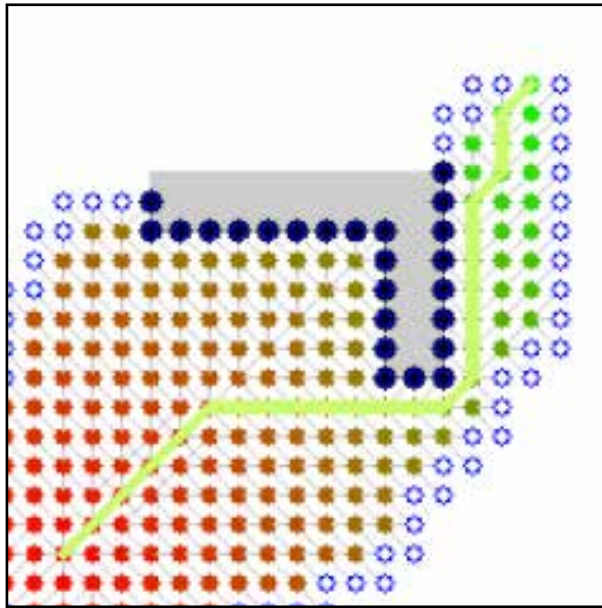


Figure 2.2.m: Pathfinding solved using an A* search algorithm uses heuristics to reduce node search time, which, for this example, takes half as long as Dijkstra's, *Bhattacharya (2011).*

The essence of agent navigation, or *pathfinding*, involves selecting the lowest cost path in a weighted graph. A common pathfinding method is *Dijkstra's search algorithm*. This calculates the shortest path by selecting neighbouring nodes that have the lowest weight cost. However, this can take a long time to calculate if an area is divided into many hundreds of nodes (Fig.2.2.l).

Instead of only considering a node's weight based on environmental conditions, the search algorithm can consider an agent's current node position, as a second weight. This second weight is called a *heuristic*, which is added to a node's overall cost during pathfinding. The idea is used in the method called the *A* (A star) search algorithm*. It works just like Dijkstra's algorithm by selecting the adjacent nodes with the lowest cost. However, instead of looking at all nearby nodes, A* ignores nodes that takes a path further away from the target, which are represented by a higher heuristic cost (Fig.2.2.m). Ultimately, this optimizes the time spent searching in a given space.

Chapter 2.3

Value Theory

The intentions of this thesis are to quantify architectural arrangements in an airport terminal as a function of human performance. To achieve this, the thesis must answer the following questions: what architectural elements influence the value of a space, and how can these elements be modelled mathematically to quantify an architectural performance?

Methods of Valuation

The thesis's understanding of architectural value begins with the philosophy of *instrumental* and *intrinsic* value, as a way of quantifying the physical environment. Instrumental value describes things that are useful for achieving a purpose, or, informally, things that are a *means to an end*.^{[1][2]} For example, a sign has instrumental value for passengers because they can use it to find their gate. This is a common functional description of value, which is like the ideas described for validation and agents.

In contrast, philosophers argue that things are also valuable, or *good* (ethically), not only because of the results they provide.^[3] Instead, things can have intrinsic value, which are inherent qualities that are important by themselves.^{[4][5]} For example, the courtesy shown by airport staff can be intrinsically good behaviour, which passengers can see as valuable by itself. Although, in *Theory of Valuation*, Dewey argues that intrinsic value cannot exist without a *means*, or use.^[6] Additionally, intrinsic value is dependant on the context of what people (society) define as "good". This can be different based on the beliefs of individual people.^[7] For example, the courtesy of airport staff is serving the purpose of getting passengers checked in. If airport staff do not act courteously, then it becomes more difficult to finish checking in due to

1. Hirose, Iwao; Olson, Jonas. *The Oxford Handbook of Value Theory*. New York: Oxford University Press, 2015. 14.

2. Dewey, John. *Theory of Valuation*. Chicago, Ill: University of Chicago Press, 1939. 26.

3. Schroeder, Mark. "Value Theory". *The Stanford Encyclopedia of Philosophy* (Fall 2016 Edition). Edward N. Zalta (ed.). <https://plato.stanford.edu/archives/fall2016/entries/value-theory>.

4. Dewey. *Valuation*. 26.

5. Schroeder. "Value Theory".

6. Dewey. *Valuation*. 27-29.

7. Dewey. *Valuation*. 29-31.

poor communication between staff and passengers. Although courtesy is an intrinsically good behaviour from the view of society, the level of courtesy is dependant on the context and beliefs of individual passengers.

Meanwhile, architectural research is concerned with valuing design in both the design process and the built environment. The thesis considers the research of Holm and their work in design ideas and beliefs. Holm explains that architecture is evaluated on multiple levels and is dependant on a given perspective. They generalize architecture into the following 3 levels: *aesthetics*, *functional*, and *the users*.^[8] These evaluations can be either judged internally within the design process, or externally from critiques.^[9] For example, an airport can be valued on the interior design of a terminal, flight processing, or passenger experience. The value of each of these areas will be different from the perspective of airport developers, airline companies, and passengers. Airport developers will have higher importance on *functional* evaluation levels. Whereas airline companies might focus on the *user* evaluation levels.

Regardless of the perspective, fundamentally, everyone's design judgement is based on an internal expectation or a given standard.^[10] A standard can simply require designs to meet maximum and minimum conditions (like tolerance in validation). Or standards can be based on general criteria like architectural styles, environmental consideration, function, or material and structure.^[11] Holm explains that these general criteria may not result in quantifiable outcomes, or universal agreement.^[12] Absolute valuation is not practical because of these imprecise attributes of architecture. Therefore, architects do not know how closely a design meets these expectations until feedback is given, or, as Farmer et al. writes, "until the judgement is rendered".^[13]

Holms continues to explain that, because there is no agreement on an exact standard for valuing architectural designs, the qualities and elements that make up a design are also variable. One solution to this uncertainty, is to associate design elements with objective conditions or

8. Holm, Ivar. "Ideas and Beliefs in Architecture and Industrial Design". (PhD thesis, Oslo School of Architecture and Design, 2006). 324.

9. Collins, Peter, and William Dendy. *Architectural Judgement*. Montreal: McGill-Queens University Press, 1971. 146. As cited in Holm, "Ideas and Beliefs". 324.

10. Farmer, Ben, H. J. Louw, and Adrian. Napper. *Companion to Contemporary Architectural Thought*. London; Routledge, 1993. 526. As cited in Holm, "Ideas and Beliefs". 324.

11. Farmer, et al. *Architectural Thought*. 526. As cited in Holm, "Ideas and Beliefs". 324.

12. Holm "Ideas and Beliefs". 324.

13. Farmer et al. *Architectural Thought*. 526. As cited in Holm, "Ideas and Beliefs". 324.

phenomena.^[14] Holm discusses the work of Cuff, who explains how these objective conditions are not inherent qualities of a building, but instead are qualities perceived by individual people.^[15] Objective conditions are like physical walls or light from a window, that people can recognize. One interpretation of objective conditions is like Gibson's affordances (as seen in perception). Although, instead of being properties of things that have use for people, objective conditions are things that people may observe, in general, but are not necessarily purposeful.

Additionally, Cuff repeats the idea that design qualities are dependant on the person making that judgement. There are 3 levels of people who can make those evaluations: the *consumer or public*, *design participants*, and *design professionals*.^[16] For example, these could be passengers, airline companies, and airport planners, respectively. Cuff defines design quality as, ideally, having feedback from all three levels of people.^[17] Although, Cuff mentions it is difficult getting feedback from the consumer if there is not one specific group of people representing the public. As Holms states, this is difficult because the evaluation of a design project is dependant on the aspects people have preference for.^[18] Without knowing who the public is, there is no context for feedback. Therefore, to understand design value, it must be known who people are and what they consider important.

From this philosophy, the thesis begins to understand what elements influence the value of architectural space. Holm shows that architectural value is dependant on human beliefs and interaction. Additionally, Dewey states that the value itself is dependant on the observer's context. Therefore, architecture elements must be dependant on a person's perspective, and the context they observe them in.

Judgement-Analysis

The thesis seeks to create a mathematical function of architectural space as a function of human activity. Since every person has their own perspective, each person in the built environment will interpret architecture differently. To understand how to quantify these differences, the thesis looks at the work by Lera and their research in *judgment-analysis* techniques for architecture.

14. Holm "Ideas and Beliefs". 324.

15. Cuff, Dana. *Architecture: the Story of Practice*. Cambridge, Mass: MIT Press, 1991. 196. As cited in Holm "Ideas and Beliefs". 324.

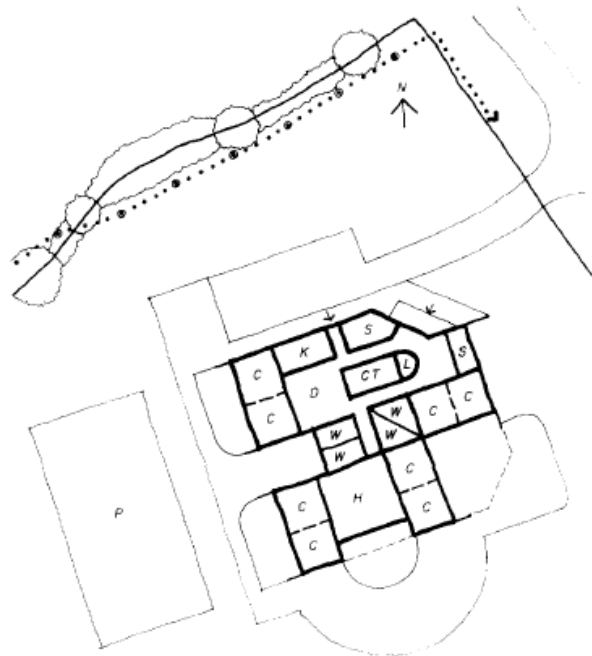
16. Cuff. *Architecture*. 196. As cited in Holm "Ideas and Beliefs". 325.

17. Cuff. *Architecture*. 196. As cited in Holm "Ideas and Beliefs". 325.

18. Holm "Ideas and Beliefs". 325.

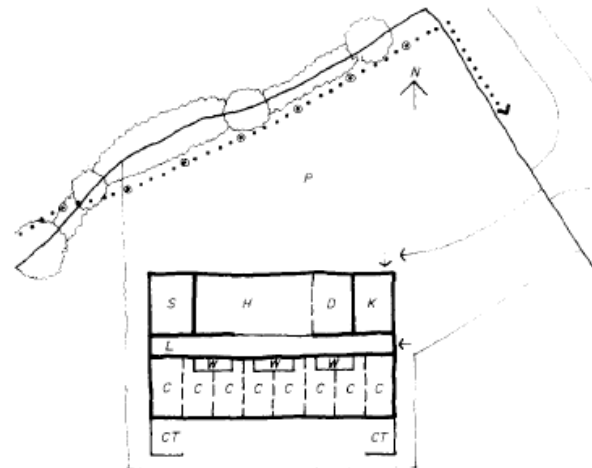
Architect 1

- 0.369 Optimize use of scarce resources
- each space to offer alternative possible uses
 - contiguous spaces jointly offering other alternatives
 - waste eliminated
 - internal circulation
 - external vehicular access
- 0.298 Child's scale/identification
- recognizable/different spaces
 - clarity of space organization
 - group identity
- 0.112 Outside/inside relationships
- teaching extends outside
 - openness
- 0.074 Flexibility
- short term: daily use/activities
 - mid term: changing educational approaches
 - long term: possible other uses (non-school?)
- 0.074 Orientation/aspect/shadowing
- 0.074 Access
- vehicles: car park (staff, visitors), deliveries
 - pedestrians



Architect 2

- 0.465 Similarity of parts
- 0.202 Administration hierarchy not reflected in plan arrangement
- 0.139 Clear, easily understandable circulation
- 0.115 Flexibility
- 0.047 Light and airy (as opposed to cosy)
- 0.033 Simple structure/building system



Key
 C - Classroom W - WCs
 K - Kitchen S - Staff rooms
 D - Dining room CT - Courtyard
 L - Library P - Playground
 H - Assembly hall

Figure 2.3.a: First two architect's priorities and school floor plans, Lera (1981).

Lera conducted an experiment to understand how designers' values influence their own work and their preference between alternative design choices. ^[19] The experiment involved a group of 6 architects (and non-architects for comparison) designing individual floor plans of an elementary school as part of a "judgement-analysis exercise". ^[20] The architects listed out their top 6 attributes, or *priorities*, for a school and ranked them based on their subjective importance. Every architect gave different ideas they thought were important, like having flexibility between inside and outside spaces, having clear circulation, or having good building orientation (Fig.2.3.a). Although some qualities were similar, every architect had 6 unique attributes of a school that were important to them.

Lera's purpose for using a judgment-analysis technique was to assign a value to design attributes both verbally and numerically. ^[21] This gives a comparison between an architect's school proposal and their given subjective values. All 6 school designs were analysed using a distinct *utility model* based on each architect's subjective priorities. ^[22] The architects also valued all the schools based on how the overall designs achieved each attribute. ^[23] Lera's illustrations in Fig.2.3.a show the schools and priorities of the first two architects. The decimal value beside each attribute is the calculated *weight* of importance according to the architect's ranking (see details in Prioritization section). Note that the architects do not list physical components, like walls and windows, but instead properties and conditions.

The results of Lera's experiment compared the design values for each architect's overall judgement with the values from the utility model. As the charts in Fig.2.3.b show, the utility model is able to align closely to the overall judgement for each floor plan. Lera mentions there is no statistical relationship, or *concordance*, between the architects together because they had different design opinions. ^[24] However, for each architect individually, Lera shows that the utility model matches the overall evaluation of the floor plans, given the subjective judgement.

The importance of Lera's research is that subjective design qualities have the potential to be quantified numerically using a method of ranking attributes, solely based on a user's preferences. Like the architects, agents can use a judgement analysis model to quantify the value

19. Lera, Sebastian G. "Architectural Designers' Values and the Evaluation of Their Designs." *Design studies* 2, no. 3 (1981): 131.

20. Lera. "Architectural Designers' Values." 131.

21. Lera. "Architectural Designers' Values." 132.

22. Lera. "Architectural Designers' Values." 133.

23. Lera. "Architectural Designers' Values." 132.

24. Lera. "Architectural Designers' Values." 136.

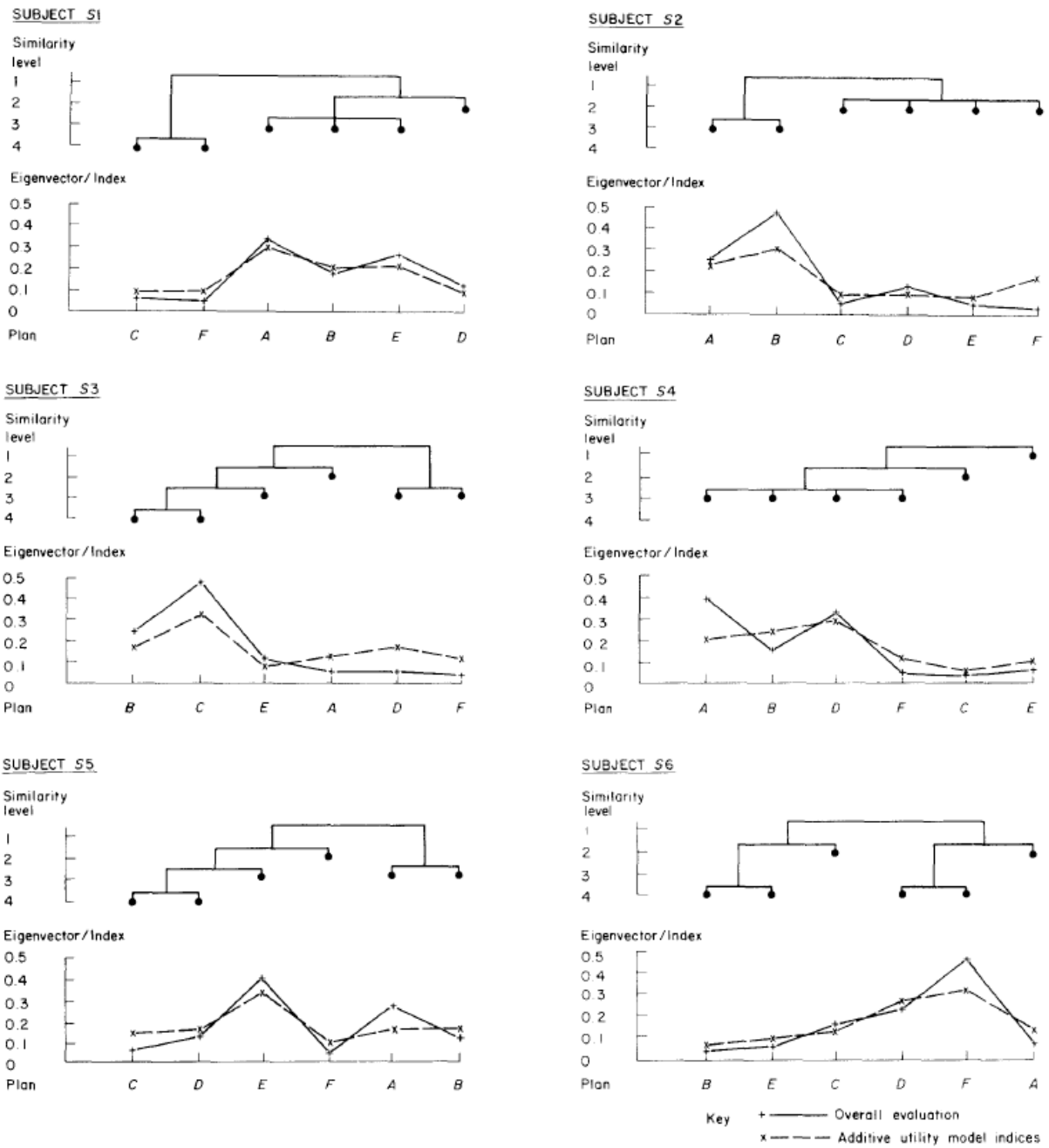


Figure 2.3.b: Comparing utility model values with overall values for each architect, Lera (1981).

of architectural space. Therefore, the thesis must consider how architecture influences passenger preferences in an airport (see details in Airport Domains section).

Prioritization

In Lera's experiment, the utility model uses a method called *prioritization*.^[25] This is an approach established by Saaty for analysing hierarchical structures.^[26] Prioritization is a process of ranking things based on their subjective importance.^[27] It works by comparing different perspectives by scaling, or *normalizing*, subjective values of a given attribute using the property of *eigenvectors*.

Saaty's approach for analysis is like the philosophy of instrumental value. He states that people judge the importance of an object, or activity, based on more than one factor. Each factor is a target that an activity must fulfil, within a larger hierarchy (Fig.2.3.c).^[28] For example, the activity of boarding a plane is dependant on the factors of a boarding pass and security clearance, within the larger hierarchy of passenger processing. The method uses *weights*, which Saaty associates with priorities, to rank activities relative to each other, based on its importance.^[29]

The process involves a user comparing all possible combinations of pairs of attributes, in the form of a matrix.^[30] It requires the user to verbally state which attributes are better than others.^[31] For each pair of attributes, the user records its importance on a scale from 1 to 9. If the attributes have equal importance, then the user records a 1. If one attribute has a higher importance than the other, then the value is recorded from 2 to 9, where 9 has the highest importance.^[32] For the opposite relationship, the user records the reciprocal value (1/2, 1/9, etc.).^[33] In an agent-based model, individual agents cannot verbally state what their preferences are. Instead, priority rankings must be randomly assigned based on statistical data.

25. Lera. "Architectural Designers' Values." 132.

26. Saaty, Thomas L. "A Scaling Method for Priorities in Hierarchical Structures." *Journal of mathematical psychology* 15, no. 3 (1977): 234.

27. Lera. "Architectural Designers' Values." 132.

28. Saaty. "Scaling Method for Priorities." 234.

29. Saaty. "Scaling Method for Priorities." 234.

30. Lera. "Architectural Designers' Values." 132.

31. Lera. "Architectural Designers' Values." 132.

32. Saaty. "Scaling Method for Priorities." 245-246.

33. Lera. "Architectural Designers' Values." 132.



Figure 2.3.c: Simple analytic hierarchy process structure, based on diagram by Sander (2009), redrawn by author.

As Saaty generalizes, if the process compares a set of n objects in pairs based on their given weights, then the objects can be denoted as A_1, A_2, \dots, A_n and the respective weights as w_1, w_2, \dots, w_n .^[34] The *pairwise* comparison can be represented in a matrix as,

$$A = \begin{matrix} & A_1 & A_2 & \cdots & A_n \\ A_1 & \left[\begin{array}{cccc} w_1/w_1 & w_1/w_2 & \cdots & w_1/w_n \\ w_2/w_1 & w_2/w_2 & \cdots & w_2/w_n \\ \vdots & \vdots & & \vdots \\ w_n/w_1 & w_n/w_2 & \cdots & w_n/w_n \end{array} \right] \end{matrix}$$

As can be seen, the matrix has symmetry. The main diagonal will always be equal to one, and the values in the lower triangular matrix are the reciprocal values of the upper triangular matrix.

Saaty compares different priorities by using *eigenvectors*, a property of a matrix. Informally, an eigenvector is a direction that does not change when a 2D space is transformed. It can be thought of as a fixed reference point that subjective ideas can be compared to, since it does not change from different perspectives (Fig.2.3.d). Formally, in linear algebra, eigenvectors are non-zero vectors, whose magnitude is scaled by a factor (*eigenvalue*) during a linear transformation.^{[35][36]} If there is a vector space, A , that undergoes a linear transformation containing a non-zero vector \mathbf{u} , then \mathbf{u} is an eigenvector of A if $A(\mathbf{u})$ is a scalar multiple of \mathbf{u} , which is written in the form,

$$A(\mathbf{u}) = \lambda \mathbf{u},$$

where λ is a scalar in A , called the eigenvalue, or characteristic root, of \mathbf{u} .^{[37][38]} In a finite dimensional space, then the above equation can be simplified to,

$$A\mathbf{u} = \lambda \mathbf{u},$$

which can be rearranged into the form,^[39]

$$(A - \lambda I)\mathbf{u} = \mathbf{0},$$

34. Saaty. "Scaling Method for Priorities." 235.

35. Khan Academy. "Introduction to eigenvalues and eigenvectors". Linear Algebra, Alternative coordinate systems, Eign-everything. <https://www.khanacademy.org/math/linear-algebra/alternate-bases/eigen-everything/v/linear-algebra-introduction-to-eigenvalues-and-eigenvectors>.

36. 3Blue1Brown. "Eigenvectors and eigenvalues | Essence of linear algebra, chapter 14". Youtube, September 15, 2016. <https://www.youtube.com/watch?v=PFDu9oVAE-g>.

37. Khan Academy. "Eigenvalues and eigenvectors".

38. 3Blue1Brown. "Eigenvectors and eigenvalues".

39. Saaty. "Scaling Method for Priorities." 236.

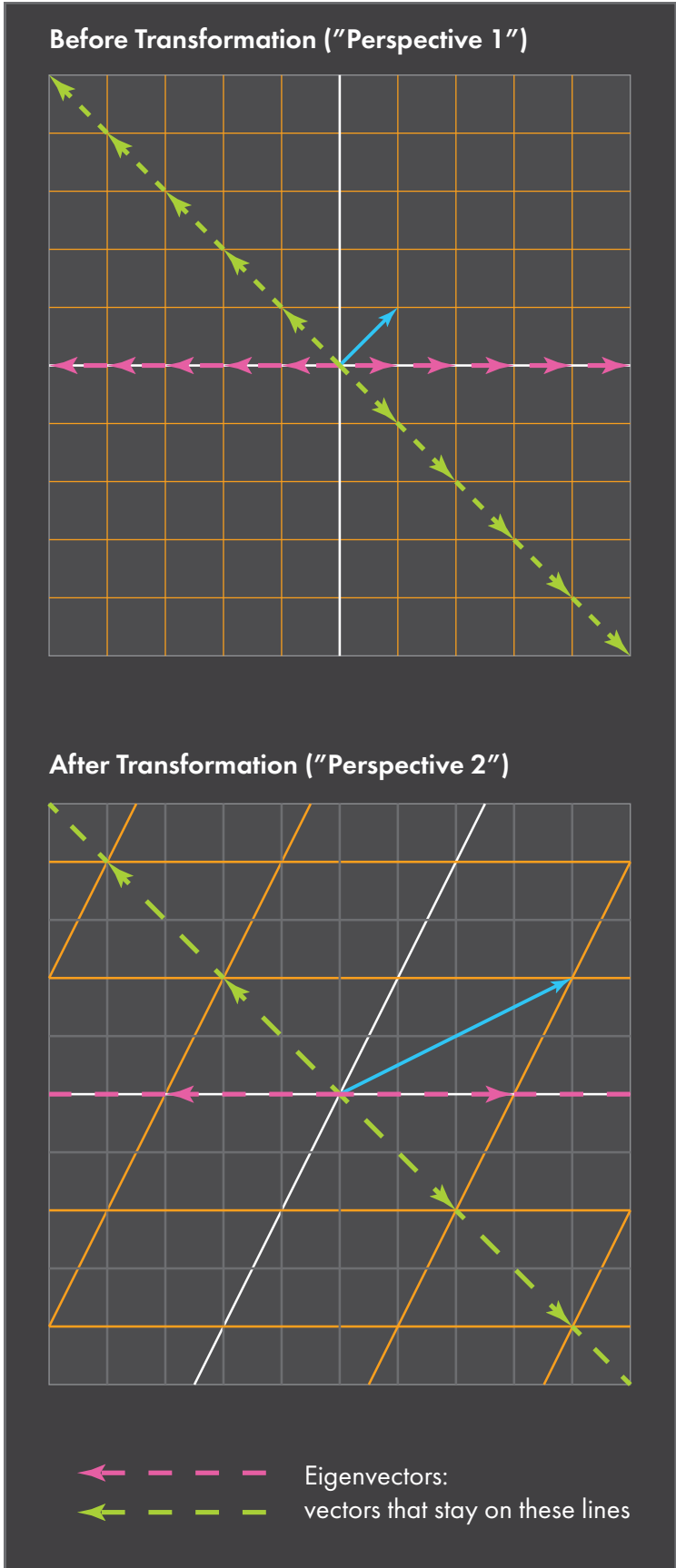


Figure 2.3.d: Eigenvectors after a linear transformation, based on animation by 3Blue1Brown (2016), redrawn by author.

where I is an identity matrix (all ones on the main diagonal, and zero everywhere else). This requires a non-zero vector, \mathbf{u} , that makes the equation equal to a zero vector. [40] [41] Or a vector that stays the same after being transformed into a new perspective.

The maximum eigenvalue, λ , of a priority matrix, A , determines the consistency of the subjective judgement. [42] If there is perfect consistency across the judgment weights, then the eigenvalue should equal the number of priorities, ($\lambda = n$). [43] Therefore, an eigenvalue provides validation for a user’s judgement within the process of prioritization.

In Lera’s experiment, the utility model calculates the final score for each floor plan. A priority’s final weighting, w_n , is calculated as the average of all column weights, a_{ij} , in the matrix, as a comparison between two priorities i and j . The given utility function is the sum of all priorities multiplied by the design’s overall judgement, y_n , in the form, [44]

$$P = \sum_{n=1}^n w_n y_n,$$

where P is a *composite vector*, representing the final value of the floor plan. If a design has a perfect score, the priority weights and final value of P , will equal to one. In the context of an agent-based model, the design judgement value, y_n , must be determined from the agent’s interaction with the environment (see details in Scoring Architecture section).

The best way to understand how this thesis uses the eigenvector process, is to go through some examples. In their research, Saaty walks through several applications of prioritization for economics, politics, and engineering. [45] [46] As an introduction, Saaty explains that eigenvectors are best understood using probability, which also validates how eigenvectors give the correct value. [47] For example, imagine there is a bag that has 6 coloured balls, with 1 blue, 2 red, and 3 green balls. The probability of picking each colour from the bag is 1/6 , 2/6, and 3/6,

40. Khan Academy. “Eigenvalues and eigenvectors”.

41. 3Blue1Brown. “Eigenvectors and eigenvalues”.

42. Lera. “Architectural Designers’ Values.” 132.

43. Saaty. “Scaling Method for Priorities.” 236.

44. Lera. “Architectural Designers’ Values.” 133.

45. Saaty. “Scaling Method for Priorities.” 252-276.

46. Saaty, Thomas L. “Modeling Unstructured Decision Problems — the Theory of Analytical Hierarchies.” *Mathematics and computers in simulation* 20, no. 3 (1978): 153-156.

47. Saaty. “Unstructured Decision Problems.” 153.

respectively. These are like the priorities of every colour. The probabilities can be written in a priority matrix as a ratio to each other, as shown in Figure 2.3.e.

Priority Matrix	● Blue	● Red	● Green	
● Blue	1	1/2	1/3	
● Red	2	1	2/3	
● Green	3	3/2	1	
Column Ratios				Eigenvectors
● Blue	1/6	0.5/3	0.33/2	0.16
● Red	2/6	1/3	0.66/2	0.33
● Green	3/6	1.5/3	1/2	0.50

Figure 2.3.e: Priority matrix and corresponding eigenvectors for each coloured ball.

The process involves taking the priority’s average in every column. For example, the first blue column ratio is equal to the priority matrix weight (1) divided by the sum of the column (1 + 2 + 3), giving a ratio of 1/6 . Since the weighting is consistent, the eigenvalue of this matrix is equal to the total number of colours (3). [48] The consistency also means that the normalized eigenvector is equal to any of the column ratios for all colours. However, if the eigenvalue is not equal to the number of priorities ($\lambda \neq n$), then the eigenvectors must be taken as the average of all columns. The final eigenvector weights (w_n) for blue, red, and green are 0.16, 0.33, and 0.50, respectively, which matches the probability. Note that the sum of all eigenvectors equals one, which is a property of the weighting. As a result, the eigenvector process is the same as normalizing the ratio of each coloured ball in the bag.

Saaty also shows how prioritization works in practical applications, with more than one layer of hierarchy and subjective judgements. [49] For example, imagine a person having to decide between 3 different job offers. Saaty explains that the person’s reasons for choosing a company can be described in a priority matrix as shown in Fig.2.3.f. The attributes of each company can also be ranked using a secondary matrix for each of the person’s 6 priorities (Fig.2.3.g). In the primary priority matrix, Saaty calculates the eigenvalue as $\lambda = 6.35$, and eigenvectors,

48. Saaty. “Unstructured Decision Problems.” 153.

49. Saaty. “Unstructured Decision Problems.” 153.

	Research	Growth	Benefits	Colleagues	Location	Reputation
Research	1	1	1	4	1	$\frac{1}{2}$
Growth	1	1	2	4	1	$\frac{1}{2}$
Benefits	1	$\frac{1}{2}$	1	5	3	$\frac{1}{2}$
Colleagues	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{1}{5}$	1	$\frac{1}{3}$	$\frac{1}{3}$
Location	1	1	$\frac{1}{3}$	3	1	1
Reputation	2	2	2	3	3	1

Figure 2.3.f: Job satisfaction priority matrix, Saaty (1978). The first row says Research is as important as Growth, 4 times more important as Colleagues, and half as important as Reputation.

Research				Growth				Benefits			
	A	B	C		A	B	C		A	B	C
A	1	$\frac{1}{4}$	$\frac{1}{2}$	A	1	$\frac{1}{4}$	$\frac{1}{5}$	A	1	3	$\frac{1}{3}$
B	4	1	3	B	4	1	$\frac{1}{2}$	B	$\frac{1}{3}$	1	1
C	2	$\frac{1}{3}$	1	C	5	2	1	C	3	1	1
Colleagues				Location				Reputation			
	A	B	C		A	B	C		A	B	C
A	1	$\frac{1}{3}$	5	A	1	1	7	A	1	7	9
B	3	1	7	B	1	1	7	B	$\frac{1}{7}$	1	5
C	$\frac{1}{5}$	$\frac{1}{7}$	1	C	$\frac{1}{7}$	$\frac{1}{7}$	1	C	$\frac{1}{9}$	$\frac{1}{5}$	1

Figure 2.3.g: Company attributes matrices, Saaty (1978). The first row of the Research matrix says the Research at Company A is only a quarter as good as Company B and half as good as Company C.

	Research	Growth	Benefits	Colleagues	Location	Reputation
$\lambda_{\max} =$	3.02	3.02	3.56	3.06	3	3.21
Company A	0.14	0.10	0.32	0.28	0.47	0.77
Company B	0.63	0.33	0.22	0.65	0.47	0.17
Company C	0.24	0.57	0.46	0.07	0.07	0.05

Figure 2.3.h: Company attributes eigenvectors, Saaty (1978).

respectively, as 0.16, 0.19, 0.19, 0.05, 0.12, and 0.30. ^[50] Since $\lambda \neq n$, the person ranking the job attributes does not have perfect consistency in judgement. However, Saaty states that the difference of 0.35 is within reason based on statistical analysis. ^[51] This process is repeated to find the eigenvalues and eigenvectors for each attribute (Fig.2.3.h). After normalizing, the composite vectors for each company are $A = 0.40$, $B = 0.34$, and $C = 0.26$. Therefore, Saaty concludes that Company A is the best choice given this person's priorities. ^[52]

With Saaty's research, the thesis begins to answer the question of how to mathematically model architectural performance. Holm shows that architectural value is dependant on the perspective of individual people. Saaty and Lera demonstrate that it is possible to rank any number of conditions, even if people do not share the same expectations. This process works because people's values can be normalized using eigenvectors, to compare different perspectives within an architectural space. Therefore, the thesis concludes that a value function for architecture must be dependant on the collective judgement of many people, like an experience survey or product review. The final analysis of a given architectural value, then must be interpreted using statistical analysis.

Airport Domains

There are numerous things in an airport that might be important to passengers. Saaty shows that, regardless of what people's expectations are, these can be normalized with a certain weight of importance. The next issue to consider for the thesis's agent simulation is, what aspects in an airport are important to people, and how these aspects relate to the physical terminal building. To understand these factors, the thesis looks to the research of Wiredja et al. and their passenger-centred model for evaluating airport performance.

Wiredja et al.'s research begins by stating that an effective model for quantifying passenger experience should use a *weight-based indicator* approach. ^[53] This method defines airport service performance as a function of passenger responses. Based on this approach, Wiredja et al. summarizes existing techniques for analysing passenger experience. Some relevant methods they mention are, importance-performance analysis, regression analysis, common factor approach, fuzzy multi-attribute decision making, and analytical hierarchy process

50. Saaty. "Unstructured Decision Problems." 153.

51. Saaty. "Scaling Method for Priorities." 252.

52. Saaty. "Unstructured Decision Problems." 153.

53. Wiredja, Dedy; Vesna Popovic, and Alethea Blackler. "A Passenger-Centred Model in Assessing Airport Service Performance." *Journal of Modelling in Management* 14, no. 2 (May 10, 2019): 502.

(prioritization).^[54] Overall, these methods are equivalent to the method of prioritization described by Saaty. The basic idea involves passengers ranking every attribute within each airport domain based on their subjective importance.^[55] This is followed by assigning a weight-based metric to compare different passenger perspectives. These methods then quantify passenger experience using statistics to provide an overall rating for an airport. This confirms that prioritization is a reasonable assumption for quantifying architectural value in this thesis.

Wiredja et al. explains that the overall performance of an airport, is dependant on the performance in various sub-areas, where each sub-area has certain attributes.^[56] They explain that existing research normally defines airport performance based on one of two approaches: *service factors* or *airport domains*.^[57] Service factors divide an airport into activities, like screening, staff courtesy, information, comfort, or money value. Whereas airport domains organize an airport into areas, like baggage claim, check-in, security, or retail. In their research, Wiredja et al. mentions that most analysis models tend to focus only on the departure sequence, but fail to represent a complete passenger experience in arrival, transit, and retail domains as well.^[58] For this reason, Wiredja et al. give a more thorough description of attributes across all areas, by also organizing activities into *processing domains* and *non-processing domains*.^[59] These categories better represent passenger experience because people tend to behave differently between *processing* (queuing, checking in, etc.), and *discretionary* (wandering, shopping, etc.).^[60]

Wiredja et al. present a conceptual model for quantifying airport elements, which are comparable to ontological substances as shown in Fig.2.3.i. (as described in perception). The model has a hierarchy with four parts: *airport domains*, *passenger-centred indicators*, *service attributes*, and *passenger travel*.^[61]

Airport Domains: areas in a terminal that allow for passenger activities and interaction, like a baggage claim area.

54. Wiredja et al. "Airport Service Performance." 496-498.

55. Wiredja et al. "Airport Service Performance." 502.

56. Wiredja et al. "Airport Service Performance." 500.

57. Wiredja et al. "Airport Service Performance." 500.

58. Wiredja et al. "Airport Service Performance." 501.

59. Wiredja et al. "Airport Service Performance." 501.

60. Wiredja et al. "Airport Service Performance." 501.

61. Wiredja et al. "Airport Service Performance." 501.

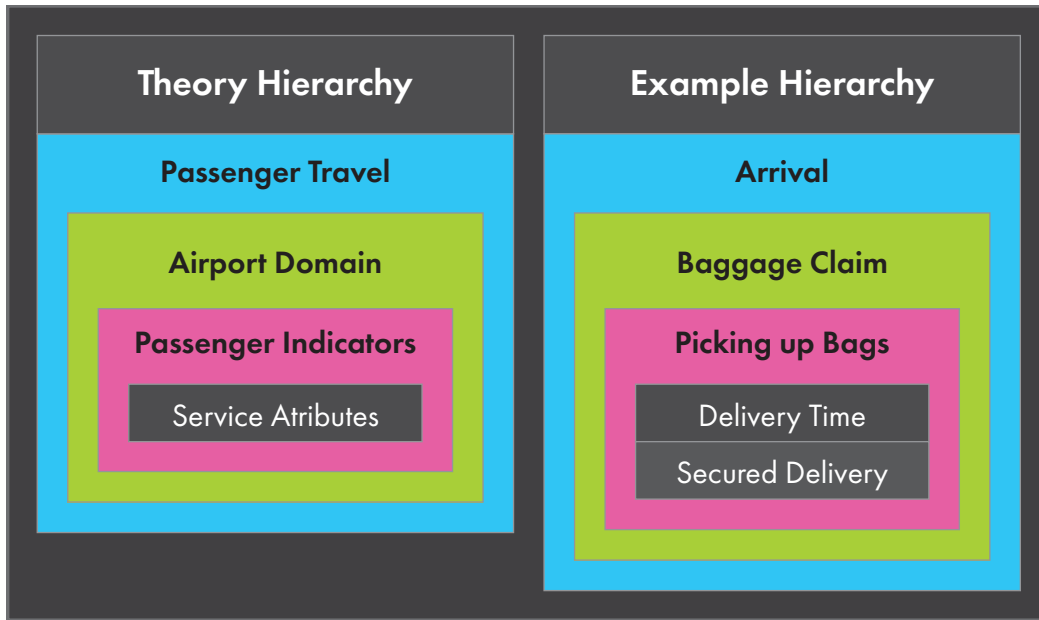


Figure 2.3.i: Passenger-centred model hierarchy, based on theory by Wiredja et al. (2019), drawn by authour.

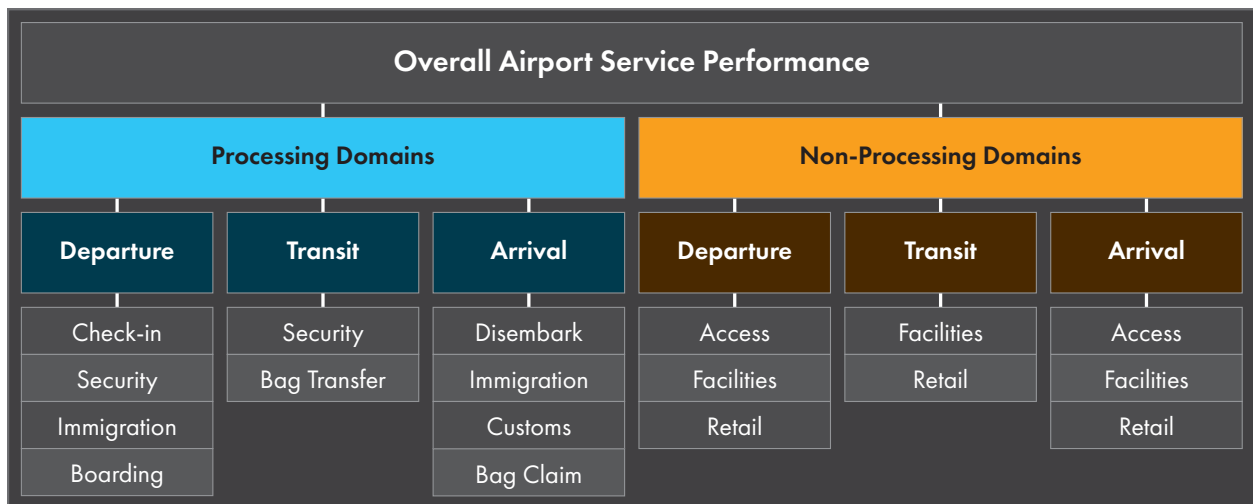


Figure 2.3.j: Airport performance is dependant on processing domains and non-processing domains, based on diagram by Wiredja et al. (2019), redrawn by authour.

Passenger Indicators: service factor groups that people see as important for the airport process, like picking up baggage.

Service Attributes: the properties passengers use to judge performance quality, like baggage delivery time.

Passenger Travel: refers to three types of passenger flows: departure, transit, and arrival; baggage claim would be part of arrival. ^[62]

An airport's overall performance is dependant on these parts within the domains of processing and non-processing (Fig.2.3.j). The full list of airport domains and their corresponding attributes according to Wiredja at al. are shown in Fig.2.3.k. These aspects are important for determining an airport's overall passenger experience, but not every attribute is directly affected by architecture. Out of this list of airport domains, the thesis highlights the attributes that architecture has an influence on, based on the level of impact: *direct*, *indirect*, or *minor* (Fig.2.3.k).

Direct Impact:

Firstly, any attributes that relate to queuing have a direct impact on architecture. The size of a queue is dependant on the number of passengers. The more people an airport expects to process, the more queuing space is required to hold those people. Fundamentally, longer queue lines require more space, which directly affect the layout of architecture. Secondly, the efficiency of any procedure, like boarding or baggage handling, is directly influenced by architecture. The efficiency of a process is concerned with bottlenecks, which are the areas that might restrict operation or movement. The longer it takes to get passengers checked in, or move bags around, the more people accumulate in one area, which takes up space. Walls, corners, and corridors can affect where people walk, or where equipment can move. If people and equipment are unable to move through spaces, due to poor planning, then the efficiency of that process is negatively affected. Thirdly, any attributes related to wayfinding are affected by architecture. The location of signs and the readability of information is dependant on how a terminal is organized and where people need to go.

For non-processing domains, any attributes related to variety, like retail or transit options, are directly related to architecture. Every retail shop requires different spaces depending on the products their selling, or the services they provide. For example, a souvenir shop might require a

62. Wiredja et al. "Airport Service Performance." 502.

Passenger Travel	Airport Domains	Attributes	Architecture Influence
Processing Domains			
Departure	Check-in	Perception of waiting time or queue length	Direct
		Staff courtesy or helpfulness	Minor
		Check-in efficiency	Direct
	Security screening	Staff courtesy	Minor
		Perception of waiting time or queue length	Direct
		Secure feeling/thoroughness	Indirect
	Immigration and customs	Perception of waiting time	Indirect
		Staff courtesy	Minor
		Efficiency of boarding procedure	Direct
	Boarding	Staff courtesy	Minor
		Avalibility of aerobridge	Direct
		Staff courtesy	Minor
Transit	Transit security screening	Secure feeling/thoroughness	Indirect
		Perception of waiting time or queue length	Direct
		Availability of automatic baggage handling	Direct
	Baggage transfer	Secured baggage	Indirect
		Avalibility of aerobridge	Direct
		The ease of finding a way out	Direct
Arrival	Disembarkation	Perception of waiting time on immigration	Indirect
		Staff courtesy	Minor
		Perception of waiting time on visa on arrival	Indirect
	Baggage claim	Perception of baggage delivery time	Indirect
		Secured baggage delivery	Indirect
	Customs and quarantine	Perception of waiting time or queue length	Direct
		Staff courtesy	Minor
		Clear information of customs declaration	Indirect
Non-Processing Domains			
Departure, Transit, and Arrival	Airport Access	Options of ground transportation	Direct
		Perception of parking or taxi queue length	Direct
		Availability of money exchange or ATM	Minor
	Airport Facilities	Sanitary condition of restrooms	Indirect
		Comfort of waiting area/lounge	Direct
		Availability of information desks	Direct
		Availability of baggage trolleys	Indirect
		Availability of internet or Wi-Fi	Indirect
		Ease of connection among airport terminals	Direct
		Availability of accomadations/hotel	Direct
		Variety of shops	Direct
	Retail Area	Value for money of shops and cafe	Minor
		Variety of food and beverages	Indirect
		Perception of shopping facilities	Indirect

Figure 2.3.k: Airport domains indicating attributes that are influenced by architecture, based on chart by Wiredja (2019), redrawn by author.

storage area, product shelving, and space for people to browse. ^[63] Whereas a café requires a kitchen, a dining hall, and logistics for garbage removal. ^[64] Likewise, the variety of transit options requires space for different vehicle types. A shuttle, or bus, requires road and curbside infrastructure. Whereas a people-mover, or train, requires railway and platform infrastructure. ^[65]

Indirect Impact:

Any attributes related to perception are indirectly impacted by architecture. This includes perception of wait times, secure feelings, or awareness of customs. Observing the number of people in a space indicates potential wait times, but not the space itself. Architecture can give the impression of a secure or safe environment using walls or barriers. Although, it is not that a barrier itself is inherently secure, but the knowledge that it is part of a controlled area that makes it secure.

Attributes related to functional amenities, like the quality of the washrooms, availability of baggage trolleys, or access to Wi-Fi, are considered indirectly impacted by architecture. The location and cleanliness of washrooms can affect where people go to use them. Moving through a terminal with a trolley can present a different dynamic than just pulling a suitcase. ^[66] There can also be areas dedicated for internet access or required infrastructure to make Wi-Fi available in the terminal building. ^[67]

Minor Impact:

Attributes like staff courtesy, money value, or food quality, only have a minor impact from architecture. Airport staff can still be friendly regardless of the design of a space. Although, a poor work environment can negatively affect staff behaviour. The appearance of an expensive retail area can be enhanced by expressive architecture. Although expensive products, or even expensive plane tickets, are not dependant on the layout of space. Likewise, food quality can be enhanced by a nicely designed atmosphere, but architecture does not directly improve food quality.

63. National Academies of Sciences, Engineering, and Medicine. “Airport Passenger Terminal Planning and Design, Volume 1: Guidebook”. *Washington, DC: The National Academies Press, (2010):* 210.

64. National Academies, “Airport Passenger Terminal Planning and Design”. 210.

65. National Academies, “Airport Passenger Terminal Planning and Design”. 281.

66. National Academies, “Airport Passenger Terminal Planning and Design”. 211.

67. National Academies, “Airport Passenger Terminal Planning and Design”. 211.

Ideally, a thorough quantification of an airport's architecture should consider all aspects of passenger experience that Wiredja et al. have listed. Human behaviour can be influenced by many factors, so it is likely all these attributes have an influence on the value of architecture. However, attributes with minor impacts are outside the scope of this thesis's agent model. In the process of making the simulation, the thesis limits the airport domains to six attributes to match the test conducted by Lera, and to simplify the digital model (details in Part 3). Specifically, to experiment with the architectural value functions, the thesis selects six airport domains that focus on the departure sequence. This includes three processing domains, *check-in*, *security screening*, *boarding gate availability*, and three non-processing domains, *waiting area comfort*, *restroom facilities*, and *retail area*.

To answer this chapter's initial question, what architectural elements influence the value of a space, the thesis concludes value is not only dependant on elements like walls, doors, or columns, which was the first assumption, but, in fact, attributes corresponding to a domain. Elements, like a wall, only have impact on architectural value, if that element is perceived by people, while doing an activity, that they feel has importance for that activity. For example, a wall can have positive value for passengers who are waiting for boarding, if passengers use the area framed by that wall to linger around (Fig.2.3.1). Likewise, that same wall can have negative value if it blocks the view of passengers trying to find their gate. Fundamentally, if people perceive architectural elements that are part of the activities they are doing, then a value for that architecture can be determined, whether positive or negative. If an architectural element is not perceived, then it has no value.

Scoring Architecture

The thesis understands the value of an architectural space is dependant on human interaction. Since every person has their own perspective, each person in the built environment will interpret architecture differently. Therefore, a complete evaluation of architectural space must use statistical analysis to approximate judgement from all people. Before this, the thesis must determine how architectural elements are modelled mathematically, from each person's perspective. Overall, the function should calculate how easily spaces allow people to accomplish their activities.

The thesis uses the term *architectural score* to mean the subjective performance of a given activity or element, denoted as the variable y_n . The term *architectural value* means the overall



Figure 2.3.1: *Passengers standing between columns and along the wall of a platform waiting to board a subway, photo by Mentatdgt (2018).*

performance, P , after multiplying the *score* with a priority *weight*, based on Lera's utility function, [68]

$$P = \sum_{n=1}^n w_n y_n,$$

The thesis proposes a function of architectural score (y_n) that depends on the purpose of the space, the perspective of the people, how they interact with the environment, and how things change over time, which can be categorized as *purpose*, *perspective*, *interaction*, and *time*. These categories are divided further into 9 quantifiable factors. For any agent in a simulated domain, the thesis defines their architectural score as a combination of one or more of the following: *priorities*, *task*, *space type*, *field-of-view*, *perception point*, *direction changes*, *accessibility*, *connectivity*, and *time* (Fig.2.3.m).

Purpose:

Priorities: Any attributes or activities that are important to a person, whether specific or broad. In an airport, priorities are attributes of an airport domain. In general, these can be functional, social, or aesthetic priorities, like the quickest path, a sense of community, or a modern style, respectively. If a person's priorities are fulfilled in a given space, then that space has architectural value for that priority.

Task: A person's goal, or intentions, in a given space. This is a subjective goal that is unique to each individual. If people can complete their tasks in a space, then that space has architectural value. However, a task is not always a priority, it can also be serving a need or a chore. It also may be different than the original function of the space. For example, a passenger in a baggage claim hall could be hungry, so their task is to find something to eat. Despite being by the baggage claim, their intentions are not related to picking up bags, which may result in different human behaviour. If a passenger cannot find food near the baggage claim, then it does not have architectural value for that specific task.

Space Type: The space program, or the function of a space. This defines what activities people can do, and what human behaviours are expected. If a space is being used as intended, then that space has architectural value. For example, a check-in area is designed for people to pick up their boarding passes, and passengers are expected to queue in line. If passengers complete

68. Lera. "Architectural Designers' Values." 133.

Priorities	Task	Space Type															
Eigenvector weight <table border="1"> <tr> <td>1.</td> <td>a</td> <td>0.24</td> </tr> <tr> <td>2.</td> <td>b</td> <td>0.07</td> </tr> <tr> <td>3.</td> <td>...</td> <td>...</td> </tr> </table>	1.	a	0.24	2.	b	0.07	3.	Binary function <table border="1"> <tr> <td>×</td> <td>a</td> </tr> <tr> <td>✓</td> <td>b</td> </tr> <tr> <td></td> <td>...</td> </tr> </table>	×	a	✓	b		...	Binary function
1.	a	0.24															
2.	b	0.07															
3.															
×	a																
✓	b																
	...																
Field-of-view	Perception Point	Direction Changes															
Isovist area 	Isovist area 	Decay function 															
Accessibility	Connectivity	Time															
Normalized function 	Normalized function 	Decay function 															

Figure 2.3.m: 9 factors for scoring architecture, with corresponding mathematical functions.

these activities successfully, then the check-in area has architectural value. Additionally, space type can also describe non-programmed spaces, like observation points or open areas for people to linger around.

Perspective:

Field-of-view: Represents visibility, or the area that a person can perceive, through observation or even through other senses. This is defined geometrically using an isovist. For public spaces, the more visibility people have, the better the architectural value. More specifically, the field-of-view is a medium people use to interact with the environment. For example, if there is an announcement on an intercom for flight boarding, this represents the area a passenger can hear that message from. Therefore, the locations where passengers can hear the intercom has architectural value.

Perception Point: The location in a space where a relevant object or feature is perceived, or understood, by a person. People gain knowledge of where they are by recognizing objects related to a certain area. If a person takes notice of something, then it has architectural value. The perception point shows where people learn about new information, which can also indicate where people might change their behaviour. For example, seeing an x-ray machine tells passengers that they are entering security screening. This may change passenger behaviour, knowing they are about to interact with security staff. Therefore, the location where passengers see the x-ray machine has architectural value.

Direction Changes: The number of times in a space a person changes their mind, decisions, or trajectory. This includes either a physical change, or a mental change. The more times a person needs to change their decisions in a space, the less architectural value it has. For example, if a passenger is lost and needs to walk back the way they came, then this is a physical direction change that does not have good architectural value. In contrast, if a passenger suddenly sees a new retail store that they want to go to, then this is considered a mental change that has good architectural value. In airport processing domains, fewer direction changes are better for both architectural value and operations.

Interaction:

Accessibility: The ability to interact or engage with a relevant object or feature in a space. If passengers can interact with elements related to the activity they are doing, then that element has architectural value. As a counterexample, a passenger walking through a concourse might

notice a retail store on the opposite side of a glass partition, which they would like to go to. Although they perceived the retail store through the glass, the partition prevents the passenger from walking over to that store. Therefore, because the store is not accessible, it does not add architectural value for that passenger.

Connectivity: The ability to move, or connect, between two spaces. This can be physical connections or abstract connections. Spaces that are connected have architectural value. For example, if a passenger can walk through security screening into the gate concourse, then the security and gates are physically connected, which has architectural value. The same is true if a passenger can look out from their gate to their plane through a glass window, then the gate and plane are visually connected, which also has architectural value. Additionally, two security check points on opposite ends of a terminal can be abstractly connected to the same security line, or boarder, even though they are physically separated. This means passengers must cross through either one of those check points to get to the gates. Since the check points form a secure boarder, they have architectural value.

Time:

Time: The amount of time a person spends in a space or process. In an airport, this includes flight times for departure and arrival, or processes like searching and queuing. For activities relating to flight time, architecture has value if a passenger is on-time. For processing, time is measured relative to a person's expectations. For example, in their research, Wiredja et al. states the average passenger does not want to wait in line longer than 15 minutes. ^[69] Therefore, if passengers are waiting more than 15 minutes, then architectural value of that space diminishes.

For each of the 9 factors above, the thesis uses the following mathematical tools to quantify architectural score: *eigenvector weights*, *binary function*, *normalized function*, *isovist area*, or *decay function*.

Eigenvectors were described in the prioritization process, and this applies to the Priorities factor. The difficulty of this approach is needing to create a performance score, or value function, for every unique attribute. If a priority is vague, it can simply require the user to score the attribute performance on a scale of 1 to 9, like priority weighting. ^[70] Otherwise, if the priority can be answered by a yes/no question, then the score is calculated with a binary function, which gives a value of either 0 or 1. Similarly, a normalized function gives a decimal

69. Wiredja et al. "Airport Service Performance." 506.

70. Saaty. "Scaling Method for Priorities." 245-246.

value between 0 and 1, like a percentage. This affects Task, Space Type, Accessibility, and Connectivity. If a task is completed or an element was interacted with, it is given a score of one, or some decimal value depending on the level of interaction. Otherwise, the score for that element is zero.

The isovist area is based on the geometry of an isovist, either in 2D or 3D, to show where things are in physical space. This applies to Field-of-view, and Perception Point. The geometry is converted to a score based on a ratio of the area size. For example, when a person takes part in an activity, the average area is calculated based on what a passenger sees. At the *perception point*, or when a person interacts with a relevant object, then the ratio is taken between the area at that point and the average area during the activity,

$$y_n = \begin{cases} \frac{A_0}{A_{avg}}, & A_0 < A_{avg} \\ 1, & A_0 \geq A_{avg} \end{cases}$$

where A_0 is the visible area at the perception point, and A_{avg} is the average area. The ratio measures variation. If there is a significant difference between the areas, like an open space going into a tight space, then the visibility drops, which has a lower architectural value. Otherwise, if the visibility remains consistent, or the area becomes bigger, then the architectural score is one.

Understandably, the thesis's assumption that higher visibility is better than lower visibility may not be true for all people or conditions. A more thorough value function should consider firstly if visibility is a person's priority or not. Additionally, many airport immigration areas are designed to restrict visibility from outside passengers for the security and safety of airport staff. ^[71] Therefore, a better value function for immigration would make sure people cannot see into the secure areas. A simple modification to the visibility ratio could account for all the locations in and around immigration to not have any "perception points" that can see airport staff or their equipment. In essence, if there is a perception point looking into secure areas, the architectural value for that location can be given a score of zero. Likewise, this can apply to any area which restricts visibility or public access.

A decay function is an exponential function that starts at one and decreases in value proportional to its current value. As it tends towards infinity, the value approaches zero. This applies to factors like Time and Direction Changes. The longer passengers are waiting, or the

71. National Academies, "Airport Passenger Terminal Planning and Design". 217, 221, 224.

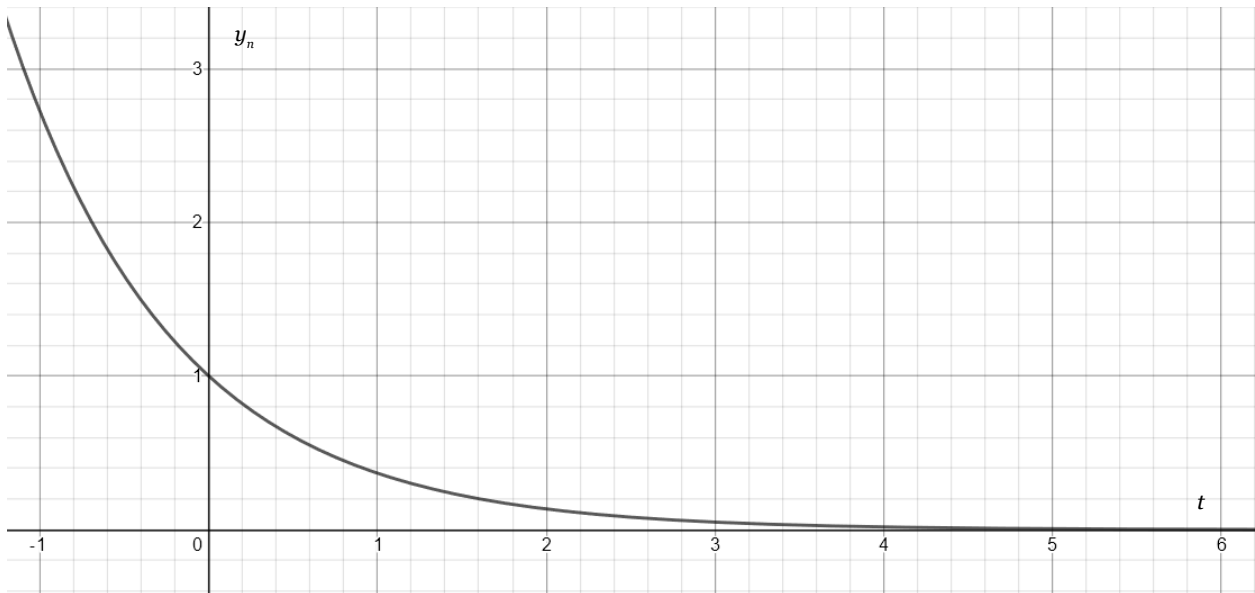


Figure 2.3.n: General exponential decay function.

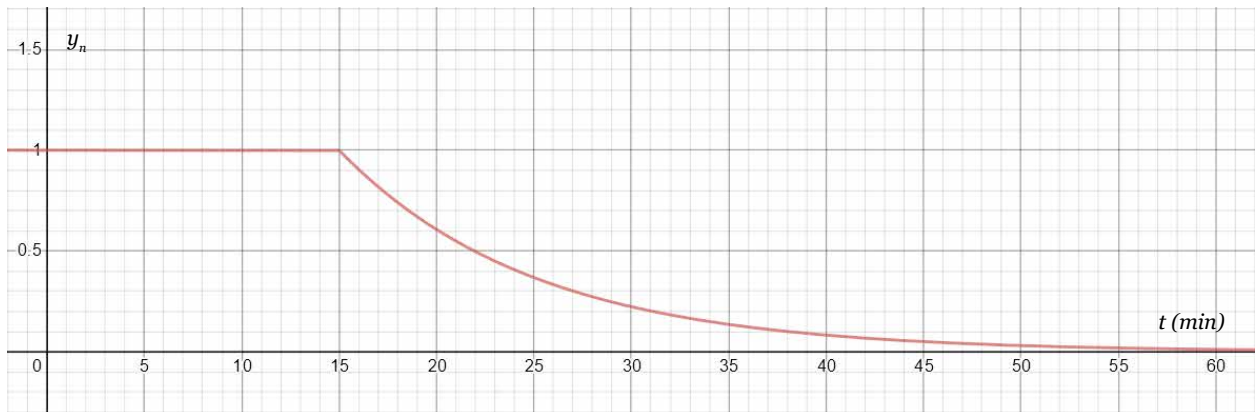


Figure 2.3.o: A piece-wise decay function for a typical passenger waiting time.

more times passengers need to retrace their steps, then the architectural score decays exponentially to zero. The rate of decay is variable and can depend on a passenger's mood or characteristics. The general equation used in this thesis to calculate the score of any activity, y_n , takes the form,

$$y_n(t) = \exp\left(-\frac{(t - k)}{\lambda}\right),$$

where t is the time (in seconds, minutes, or hours) or the number of direction changes, k is a shifting factor (when decay starts), and λ is the rate of decay, which is dependant on a person's characteristics (Fig.2.3.n). For example, a typical passenger waiting in a queue line, who does not want to wait more than 15 minutes, can judge the value of time based on the following,

$$y_n(t) = \begin{cases} \exp\left(-\frac{(t - 15)}{10}\right), & t > 15 \\ 1, & t \leq 15 \end{cases}$$

where t is the time in minutes, $k = 15$ minutes is their level of patience, and $\lambda = 10$ is an arbitrary characteristic decay factor, in which larger values mean higher tolerance (Fig.2.3.o). If this passenger is waiting in queue for 25 minutes, then the final score would be $y_n(15) = 0.37$ or 37%. If the passenger is waiting less than 15 minutes, then the score is simply one.

For any element, if its architectural score is dependant on more than one of the factors above, then the final score is taken as the average score of all factors. If a given factor has more importance than another, like departure time for flight boarding, then they can be multiplied by a higher weight. The architectural value of any element is calculated by multiplying a person's eigenvector weight, w_n , and its relative architectural score, y_n . The total performance value of any architectural space is the sum of all the products for each element or priority, from the perspective of a single person.

Summary

The intention of this chapter was to answer the questions of what architectural elements influence the value of a space, and how these elements can be modelled mathematically to quantify architectural performance. The thesis gives a brief description of the philosophical definition of value, which defines value as instrumental (useful for a purpose, i.e. a wayfinding sign) or intrinsic (good by itself i.e. airport staff courtesy). Philosophers like Dewey argue that things never truly have intrinsic value, since every action or object serves a purpose. They

explain intrinsic value is not consistent because every person has their own belief about what is considered valuable or good, which can be different based on the beliefs of a given society.

Meanwhile, research into design values for architecture shows that, although it is common for designers to organize elements based on function, aesthetics, or the users, there is no universal standard for valuation. Holm states that architects do not know the value of their designs until they are given some form of feedback. Cuff explains that architectural feedback is dependant on the perception of individuals. Although, user feedback is difficult because there is not one type of person representative of the entire public. Holm adds that it is difficult to value architecture since everyone has a different perspective, and because architectural attributes involving aesthetics or society can be imprecise.

However, experiments conducted by Lera show that subjective design values from different architects can be compared directly through a method of ranking, despite the architects having completely different design ideas. Lera's research developed a utility function that can replicate the subjective judgement from different architects. The utility function works using Saaty's method of prioritization, which ranks subjective attributes as a pairwise comparison in the form of a matrix. The process normalizes design values using eigenvectors, a property of a matrix in linear algebra that stays the same after changing perspective. The thesis walks through some of Saaty's examples to demonstrate how this works.

To figure out what things people find important in an airport, the thesis reviews the research of Wiredja et al. who states that these things can be organized into processing and non-processing domains. The thesis summarizes these domains describing which airport attributes have direct, indirect, or minor impact from architectural choices. The attributes with the greatest impact from architecture are part of passenger processing like queuing and wayfinding, or elements that involve logistics like retail storage or transit infrastructure. The attributes with the least impact involve money value, staff courtesy, or food quality. Although all airport attributes can be affected by architectural choices to some degree, the thesis selects six attributes for the agent simulation. Three are from processing domains, which are check-in, security screening, and gate availability, and three from non-processing domains, which are perception of the waiting room, restrooms, and retail areas.

The value of an architectural space depends on the purpose of the space, the perspective of the people, how they interact with the environment, and how things change over time. The thesis develops nine different factors based on the geometry of space and people's interaction to

evaluate an architectural score. The mathematical tools for evaluating a score are unique to each activity, which all involve normalizing factors based on the perception of a person. If a person perceives an architectural element, or interacts with it, then that person scores a one, or some decimal value depending on the level of interaction. Otherwise, the person scores a zero if there was no interaction.

In summary, the process of calculating architectural value involves people ranking the importance of attributes as their priorities, then having them walk through the architecture to complete a task. If people encounter relevant architectural elements, they score those elements based on how well they fulfilled their priorities. The final architectural value is a combination of each person's weighted priorities and their respective scores based on what they perceived.

Part 3:

Simulation Framework

Part 3 walks through the construction of the thesis's agent simulation. Chapter 3.0 begins by introducing the Unity game engine, and software components that are relevant for building the model. The following chapters breakdown the process of specific simulation components. Each chapter includes a process diagram showing how components work together. This is followed by a detailed list that highlights a few of the core variables in each class. Chapter 3.1 talks about how agents are made, their characteristics, and the perception process. Chapter 3.2 details how the environment is modelled for agent navigation using A* pathfinding. Chapter 3.3 describes components for airport architecture and value functions. Chapter 3.4 gives a brief summary of utility components to help control and optimize certain aspects of the simulation. Finally, chapter 3.5 concludes by stating all the assumptions and the limitations of the thesis's agent-based simulation.

Chapter 3.0

Unity Components

The goal of this thesis is to illustrate how architectural layouts can be analysed by perceptive agents. To achieve this, the thesis starts by creating a simplified agent-model that builds on the same basic principles of existing simulations. The idea is to replicate core functions, like agent navigation, so that custom behaviour can be built on top of it. Once a basic agent is established, then functions for architectural spatial analysis can be added.

Unity

This thesis uses Unity software to create the agent-based model. Unity is a game engine that can create video games and other visualization applications. Although it is primarily used in the gaming industry, Unity has applications in the architectural and engineering construction industries for its ability to model and animate buildings during the design process. ^[1] The reason for choosing Unity to create the agent-based model over other programs was because it is accessible and has strong performance for visualizations. Unity allows the ability to create custom behaviour through scripting. Custom models can be added to create 3D environments. From its gaming background, it can easily run animations over time with real-time lighting and rendering. Unity also has an intuitive user interface that the author was already familiar with, therefore time spent learning the software is minimized. There is extensive documentation that helps explain core functions when knowledge is lacking. Finally, the basic version of Unity is free to use, which still includes core components necessary for creating the agent simulation.

In Unity, projects are made up of two main parts, one is the *scene*, and the other is the *scripts*. The scene is a 3D environment where digital models are displayed (Fig.3.0.a). The models act as a physical representation of the agents and the architecture while the simulation is running. The scripts are tools that define the agent's behaviour, animation, and general mathematical relationships. They can be added to control objects or parts of the environment.

1. "Architecture, Engineering & Construction." Unity. Accessed December 2019.
<https://unity.com/solutions/architecture-engineering-construction>.

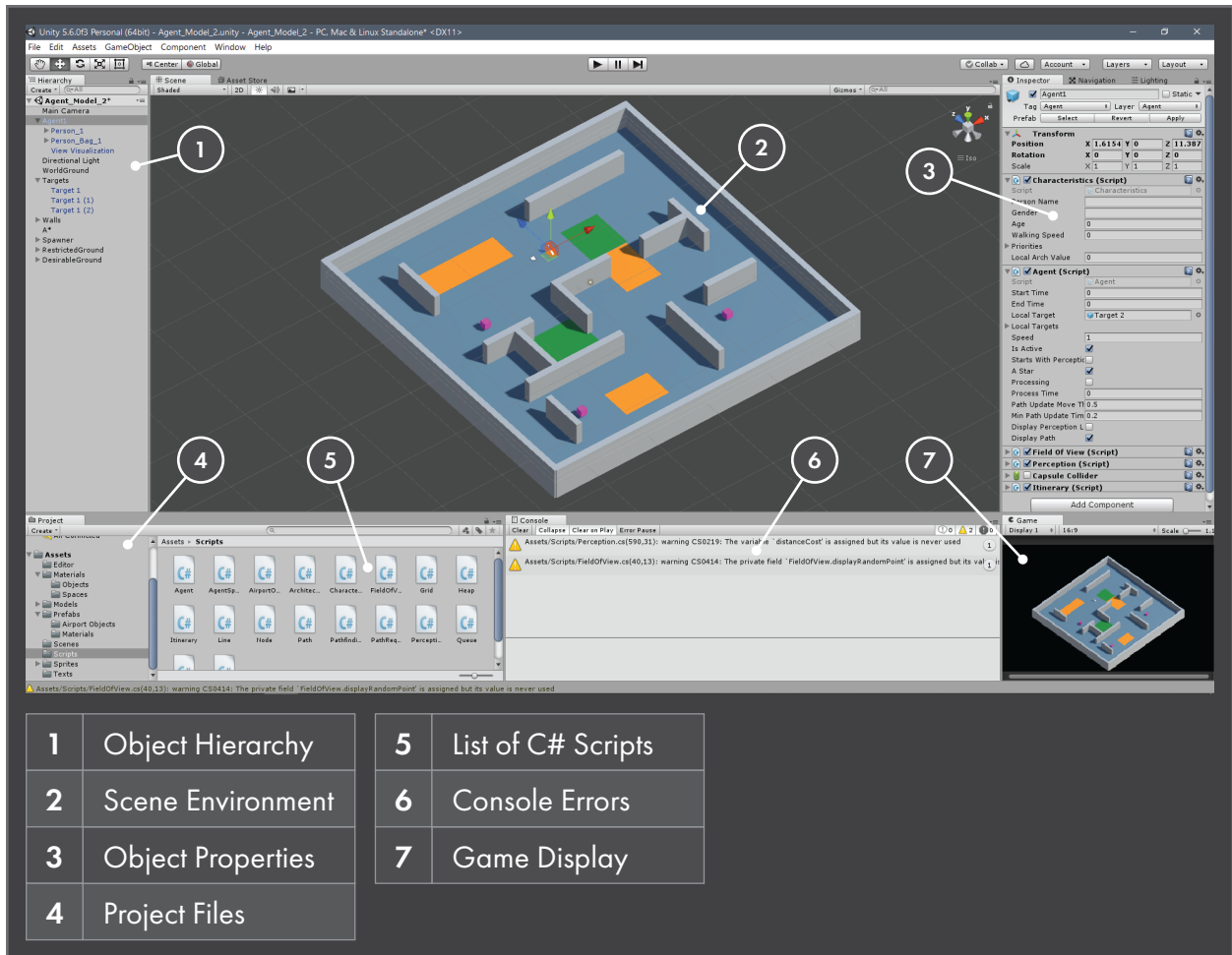


Figure 3.0.a: Unity software user interface showing the scene environment models and property toolbars.

Objects inside of a scene are called *game objects*. They can describe either physical objects or functional components. Physical objects include anything that is visible in the world like walls, flooring, doors, furniture, signs, and agent bodies. The functional components are objects that are not visible in the scene but serve to hold scripts or elements of the environment. In this simulation, this includes lighting, cameras, agent navigation, and utility scripts. If there is no physical model associated with these elements, then they are referred to as *empty game objects*.

Scripts

Scripts in Unity are based in the language C# (C Sharp). This language is described as *strong-typed* and *object-oriented* (class-based). In C#, there are three general concepts for building information: variables, methods, and classes.

A variable is a name that holds information. Every variable can store different types of information or *data types*. Strong-typed means that each type must be defined when making new variables. ^[2] This is because certain types only allow specific operations. Some basic data types are *integers* (int), decimal numbers or *floating points* (float), *strings* of text (string), and conditional *Booleans* (bool). Examples of these variables are shown below:

```
int variable1 = 12;
float variable2 = 34.506f;
string variable3 = "new text";
bool variable4 = true;
```

Methods in C# work like functions. They are a collection of code that can be executed by calling the method and inputting variables. ^[3] Methods allow information to be stored under one function, which can be called multiple times without having to rewrite the same lines of code repeatedly. An example of a method may be written like the following:

2. "Types (C# Programming Guide)." C# Documentation, Microsoft, July 20, 2015. Accessed October 2020. <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/>.

3. "Methods in (C#)." C# Documentation, Microsoft, May 21, 2018. Accessed October 2020. <https://docs.microsoft.com/en-us/dotnet/csharp/methods>.

```
void NewMethod1(int variable5)
{
    // lines of code, given the input of an integer "variable5"
}
```

Classes in C# are a reference type that group together related variables and methods. ^[4] Like methods, they are a way of organizing a collection of code that can be called repeatedly, without having to rewrite information. Classes allow information to be shared through *inheritance* and *composition*. This is an essential characteristic of object-oriented programming. Information is shared by referencing the class under new variables or methods called an *instance*. One class can be used by multiple objects to inherit the same properties. This is useful for agent-based models, where unique agents, who might represent different people, are still based on fundamental properties or classes. Examples of a class and a new class instance are below:

```
class Perception
{
    // methods relating to perception
}

Perception perceptionVariable1 = new Perception();
```

For example, all agents have the property of perception. A new class can be made called Perception, which is inherited in every new instance of an agent. Each agent can have a different level of perception, but the basic code does not need to be repeated every time. Within each agent, a new local variable can be assigned with the Perception class properties.

Unity also has built-in classes that take care of common game operations. Some useful classes include reference to game objects properties (GameObject), vector structures (Vector3), and time dependant functions (Time). These classes make it easier to create agent movement in a 3D environment by referencing pre-built classes instead of redefining basic elements.

4. "Classes (C# Programming Guide)." C# Documentation, Microsoft, August 21, 2018. Accessed October 2020. <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/classes>.

Agent Simulation Scripts

In this agent-based simulation, scripts are organized under four categories: *agent-related*, *A* pathfinding*, *airport architecture*, and *simulation utility*. The categories are not required for the function of the agent-based simulation, but they are helpful for organizing all of the behaviours.

Agent-related: This covers all components use to create the agent. These manage agent logic, properties, characteristics, behaviour, perception, field of view, and movement. It also stores the agent's memory, goals, and targets.

A* Pathfinding: This represents the basics of A* (A star) navigation. It defines components for mathematical graphs like nodes, weights, and paths. This also manages agent pathfinding in the game space.

Airport Architecture: This stores behaviour for all objects and concepts related to airport terminals. It includes objects like check-in counters and security screening. It controls the scheduling for flight arrival and departure times. Additionally, it manages passenger itineraries for boarding passes and security clearances. Finally, it also includes properties of architectural elements and calculations for architectural value.

Simulation Utility: This represents any ancillary tools that manages environmental functions or code optimization. This includes how agents enter and exit the simulation. It covers optimization and management of agent pathfinding. It also controls the way objects are displayed on screen, like the agent's field of view.

Each category contains a collection of classes that perform a specific function, which is listed in Fig 3.0.b. The framework for each category is explained in the following chapters.

Summary

The intension is to replicate core functions of an agent so that architectural analysis can be added to it. The thesis uses Unity, a game engine, to create the agent simulation. Unity is useful for illustrating 3D animations and can incorporated custom behaviour. Unity is composed of two parts: the *scene*, which holds the 3D models, and the *scripts*, which controls the behaviour. Scripts are based in C#, which works by organizing information into three basic layers: *variables*, *methods*, and *classes*. Variables store values based on a certain data type. Methods work like a function, which takes an input variable and produces an output. Classes provide a way to group information based on inheritance. This allows multiple agents to reference the

same properties without having to repeat code. Unity also has pre-built classes, which makes animating objects in a 3D environment easier. The thesis organizes its simulation classes into four categories, based on the agent, pathfinding, airport architecture, and utility functions, which are detailed in the following chapters.



Figure 3.0.b: Categories of script classes in Unity for the agent simulation.

Chapter 3.1

Agent-related Classes

In the following chapters, the thesis gives a brief overview of each class's process. This is followed by a corresponding process flow diagram, and a detailed list showing a small selection of core variables and methods. For this chapter, agent-related classes include *agent*, *characteristics*, *perception*, and *field of view*.

Agent Class

The agent class controls navigation, movement, and basic properties. The thesis's agent class builds on the script class by Sebastian Lague called *Unit*, which was created as part of their Unity game tutorial on *A* Pathfinding (2016)*.^[1] Characters in Lague's game follow a given path to a target using the unit class and a custom A* method. The unit class can request a new path if the target position moves while the game is running. The thesis modifies this mechanism to create a framework for a new *local target* process.

The agent class process is illustrated Fig.3.1.b. The agent is initialized using Unity's Start method. This is activated by the agent spawner class at the beginning of the simulation, and the agent properties, like walking speed, are provided from the characteristics class. There are three types of navigation available for the agent, which are built on top of Lague's unit class framework: *A* direct route*, *A* perception*, and *vector perception*. The reason the thesis has three different methods is to illustrate different navigation behaviour in various simulation conditions. The agent class first checks if perception navigation is enabled. If it is enabled, then the user of the simulation would have selected either A* or vector perception. If perception is not enabled, then the user sets the simulation to direct routing.

Agents using direct routing travel to their targets by taking the cheapest cost path from the A* method. This navigation replicates how agents walk in existing crowd simulations, like FlexSim. Direct routing can avoid high-cost areas using A*, however, it does not consider what agents perceive in the environment. Instead, agents follow the path exactly without deviating or

1. Lague, Sebastian. "Pathfinding/Episode 9 - smooth path 02/Assets/Scripts/Unit.cs". GitHub. December 30, 2016. <https://github.com/SebLague/Pathfinding/blob/master/Episode%209%20-%20smooth%20path%2002/Assets/Scripts/Unit.cs>.

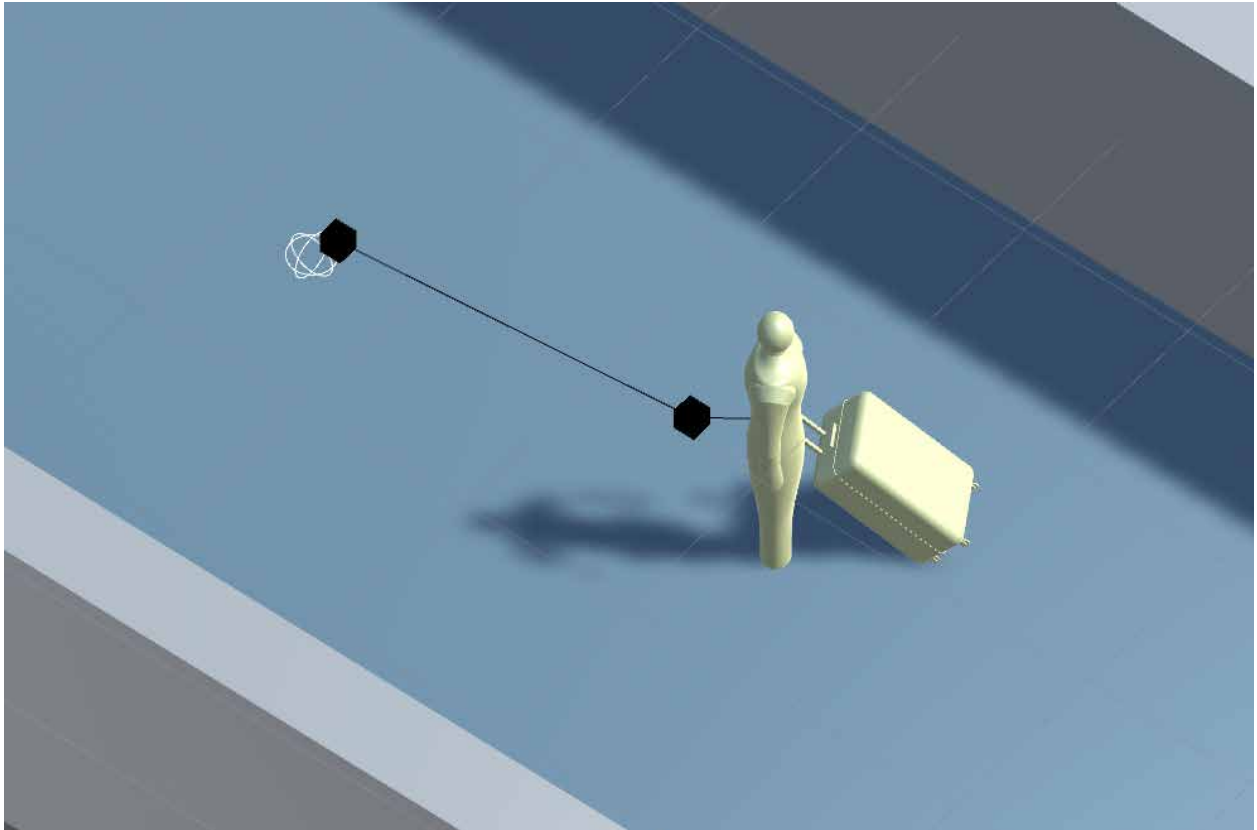


Figure 3.1.a: Agent following an A* path (black line) to a local target (white wire sphere).

updating the pathfinding over time. Once perception is disabled, the agent class bypasses the decision-making process.

If perception is enabled, then the default navigation is set to A* perception, in which agents navigate to a local target using the cheapest cost path (Fig.3.1.a). This navigation process better represents how people navigate an unfamiliar environment. People in real life move relative to objects they observe around them and change their trajectory as they discover new information. The process is approximated by providing agents with short-distance paths that are updated more frequently. It requires the perception class to choose an object the agent can see in their field of view, and then uses A* to walk towards it, avoiding high-cost areas. When the agent gets a new local target, the agent class requests a new path from the path request manager class. If the manager successfully finds a route to the target, then the path points are sent back to the agent, which they begin walking along. Once the agent reaches their local target, the agent spawner class checks if the agent is at their final destination. If the agent is at their final destination, then the agent spawner removes the agent from the world. Otherwise, the agent class requests a new local target from the perception class, and the process repeats.

The last navigation method uses vector perception. The process is similar to the default perception navigation, except that it does not use A* to find the cheapest path. Instead, a straight vector path is created between the agent and the local target to follow. Since the agent is navigating to local targets over short distances, agent behaviour using A* and straight vectors are equivalent. The main reason this navigation was created was to reduced pathfinding computation in very large environments (greater than 100 m long). The precision of the thesis's A* navigation for large environments becomes unreliable when calculating path nodes for intricate architectural areas, (narrow corridors causing agents getting stuck in walls). Therefore, navigating using vector perception is more manageable for experimenting with larger airport terminal layouts, without loosing the perception decision-making process.

In summary, the agent class handles navigation and movement. It is a modified version of Sebastian Lague's unit class from their Pathfinding tutorial. The thesis's agent class has three forms of navigation, which work in different conditions: A* direct route, A* perception, and vector perception. A* direct route is like the navigation in FlexSim simulations, which bypasses the perception process. A* perception, which is the default navigation, uses local targets, which approximates how people navigate unfamiliar environments in real life. The last navigation replaces A* with straight vectors, to reduce computation issues in large environments. However, agent perception is still maintained.

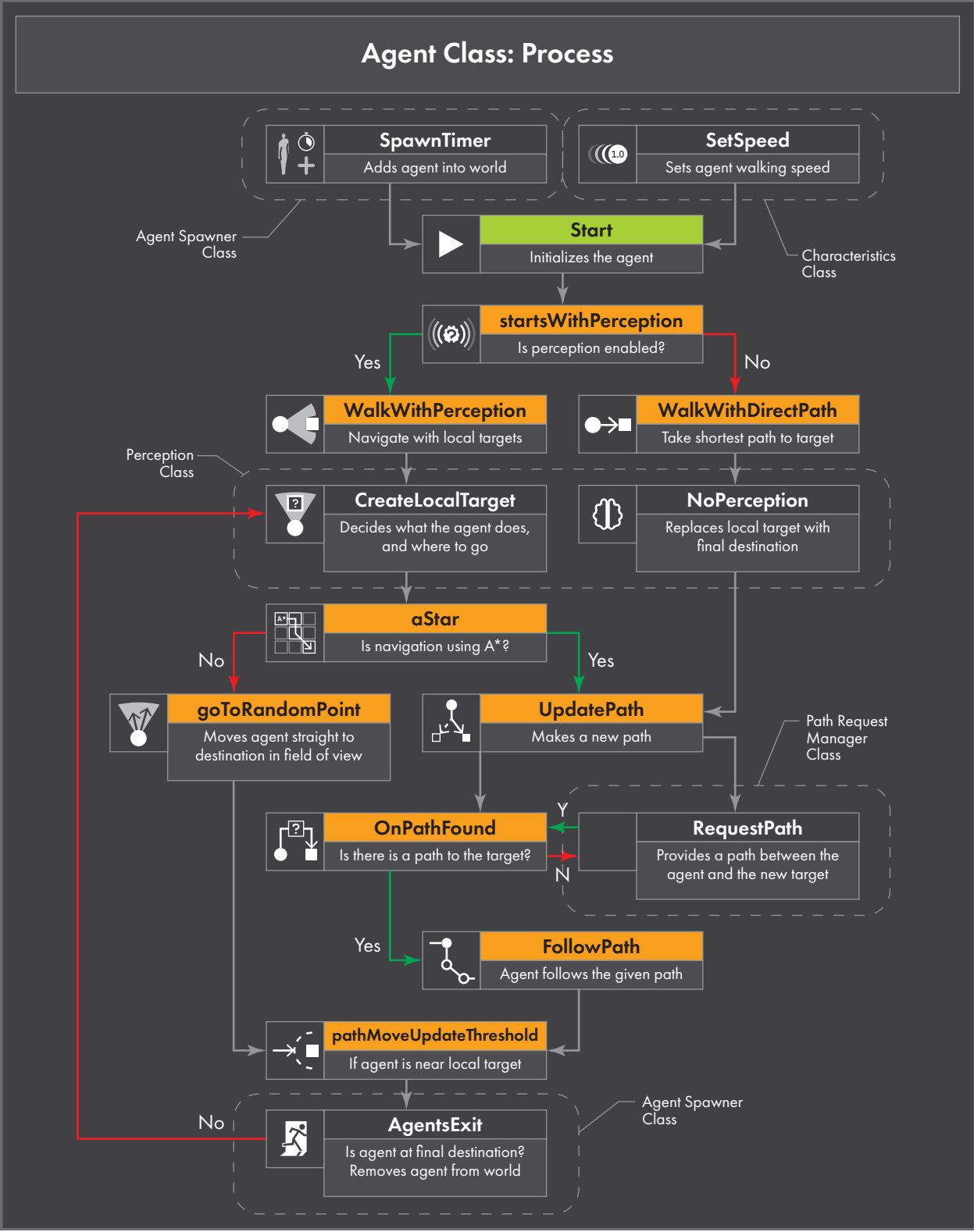


Figure 3.1.b: Process logic for the agent class.

Agent Class: Variables and Methods		
Variables		
	Variable Type Values	Description
	startTime float 0.0 - ∞	World time when the agent enters the simulation (sec)
	endTime float 0.0 - ∞	World time when the agent exits the simulation (sec)
	localTarget GameObject -	The physical object representing agent's target in their field of view
	LocalTargets List of GameObjects -	A list of agent's targets, to be used for pathfinding; holds 2 items, the old and new target
	speed float 1.0, 0.56 - 1.85	How quickly the agent walks (default 1 m/s)
	turnSpeed float 5.0	How quickly the agent can turn to a new direction (default 5)
	isActive bool false, true	Check if the agent is moving, it is not moving by default
	startsWithPerception bool true, false	Check if the agent enters the world with perception enabled
	aStar bool true, false	Check if the agent is navigating with A*

Figure 3.1.c: Key variables for the agent class, page 1.


	processing bool true, false	Check if the agent is in a processing state, or interacting with an object
	processTime float 5.0 - 15.0, 0.0 - ∞	Holds a random amount of time the agent is processing for (sec), (default 5 to 15 sec)
	path Array of Vector3 -	A collection of directions describing the agent's path
	targetIndex int 0 - ∞	The target number stored in the agent's path
	pathUpdateMoveThreshold float 0.5	Agent requests a new path if its within this distance of the final target
	minPathUpdateTime float 0.2	How frequently a path is updated
	newCharacter Characteristics -	Reference to the agent's characteristics class
	fov FeildOfView -	Reference to the agent's field of view class
	perception Perception -	Reference to the agent's perception class
	itinerary Itinerary -	Reference to the agent's Itinerary class

Figure 3.1.d: Key variables for the agent class, page 2.








Methods		
	Method Type Input Output	Description
	WalkWithDirectPath void - NoPerception	Give agent a direct path to the target
	WalkWithPerception void - UpdatePath	Makes the agent perceptive and navigates with local targets, either using A* or random vectors
	UpdatePath IEnumerator LocatTarget LocatTarget	Updates the agent path if the target has moved
	OnPathFound void newPath FollowPath	Determines if an agent can start following a path
	FollowPath IEnumerator path position	Iterates through nodes in a path and moves the agent along the path
	GoToRandomPoint IEnumerator LocalTarget position	Repeating function that makes the agent move without using A* navigation

Figure 3.1.e: Key methods for the agent class, page 3.

Characteristics Class

The characteristics class provides each agent a unique character and manages their airport priorities. The data used to define a passenger's character is based on the International Maritime Organization's (IMO) standard for evacuation simulations. When an agent is spawned into a simulation, they are first randomly assigned a gender, either male or female. The population composition, for age and gender, is randomly assigned based on IMO's distribution listed in Fig.3.1.f. ^[2] The gender of the agent then determines the agent's walking speed, as listed in Fig.3.1.g. ^[3] These charts also determine if the agent has a mobility impairment, which only affects walking speed. The gender of the agent also determines the agent's name, which is assigned from a random list of male or female names. The name is not necessary for the agent to function during the simulation, but the name helps keep track of which passengers scored which values when calculating people's priorities.

Agent priorities are randomly assigned based on six airport domains from the departure sequence. There are three processing domains, *check-in*, *security screening*, *boarding gate availability*, and three non-processing domains, *waiting area comfort*, *restroom facilities*, and *retail/food area*. In the thesis's simulation, every agent has the same six priorities. In each character, priorities are randomly ranked on a scale of 1 to 9. Then the ranking is normalized using the eigenvector process. The name, ranking, and value are stored in a local Priority class. When the agent is walking through the simulation, the architectural score is accumulated in this local class. Once the agent exits the simulation, the final score is sent to the architecture class along with the name of that agent. Sample outputs of the characteristics class are illustrated in Fig.3.1.h and Fig.3.1.i, which lists randomly assigned character attributes and corresponding priority matrices.

In summary, the characteristics class defines the agent's age, gender, walking speed, and airport priorities. The characteristics for this thesis are based on the IMO standard for evacuation simulations. The agent's priorities are randomly assigned when the simulation starts, and the values are stored in a local priority class, which are updated throughout the simulation.

2. IMO. "Guidelines for Evacuation Analysis for New and Existing Passenger Ships." International Maritime Organization (IMO). MSC.1/Circ.1238. October 30, 2007. 6.

3. IMO. "Guidelines for Evacuation Analysis." 8.

Population groups - passengers	Percentage of passengers (%)
Females younger than 30 years	7
Females 30-50 years old	7
Females older than 50 years	16
Females older than 50, mobility impaired (1)	10
Females older than 50, mobility impaired (2)	10
Males younger than 30 years	7
Males 30-50 years old	7
Males older than 50 years	16
Males older than 50, mobility impaired (1)	10
Males older than 50, mobility impaired (2)	10
Population groups – crew	Percentage of crew (%)
Crew females	50
Crew males	50

Figure 3.1.f: Population distribution, for age and gender, from the IMO standard for evacuation simulations (2007), which are used in the characteristics class.

Population groups – passengers	Walking speed on flat terrain (e.g., corridors)	
	Minimum (m/s)	Maximum (m/s)
Females younger than 30 years	0.93	1.55
Females 30-50 years old	0.71	1.19
Females older than 50 years	0.56	0.94
Females older than 50, mobility impaired (1)	0.43	0.71
Females older than 50, mobility impaired (2)	0.37	0.61
Males younger than 30 years	1.11	1.85
Males 30-50 years old	0.97	1.62
Males older than 50 years	0.84	1.4
Males older than 50, mobility impaired (1)	0.64	1.06
Males older than 50, mobility impaired (2)	0.55	0.91
Population groups – crew	Walking speed on flat terrain (e.g., corridors)	
	Minimum (m/s)	Maximum (m/s)
Crew females	0.93	1.55
Crew males	1.11	1.85

Figure 3.1.g: Passenger walking speeds, from the IMO standard for evacuation simulations (2007), which are used in the characteristics class.

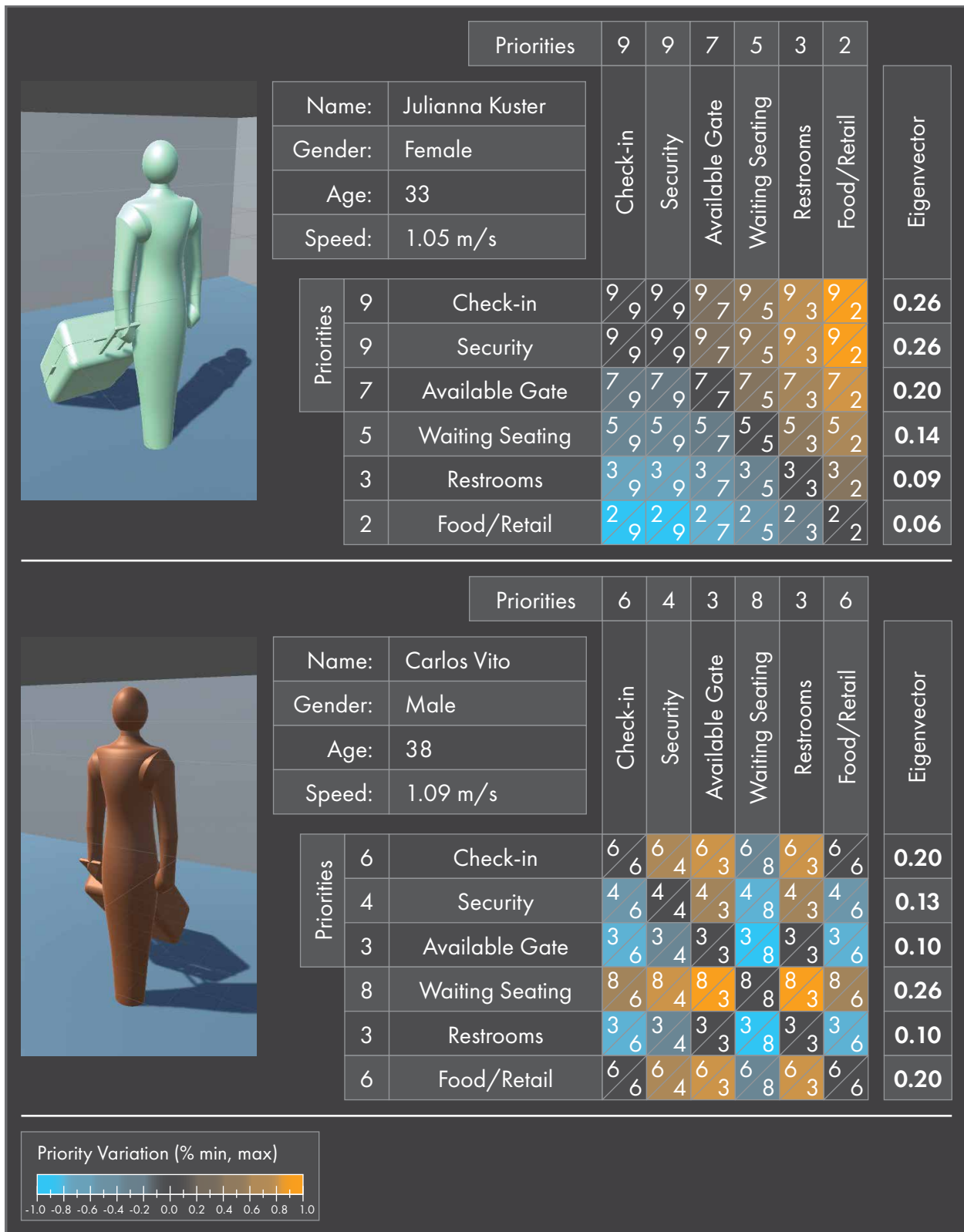


Figure 3.1.h: Samples of randomly assigned characteristics and priority matrices, page 1.

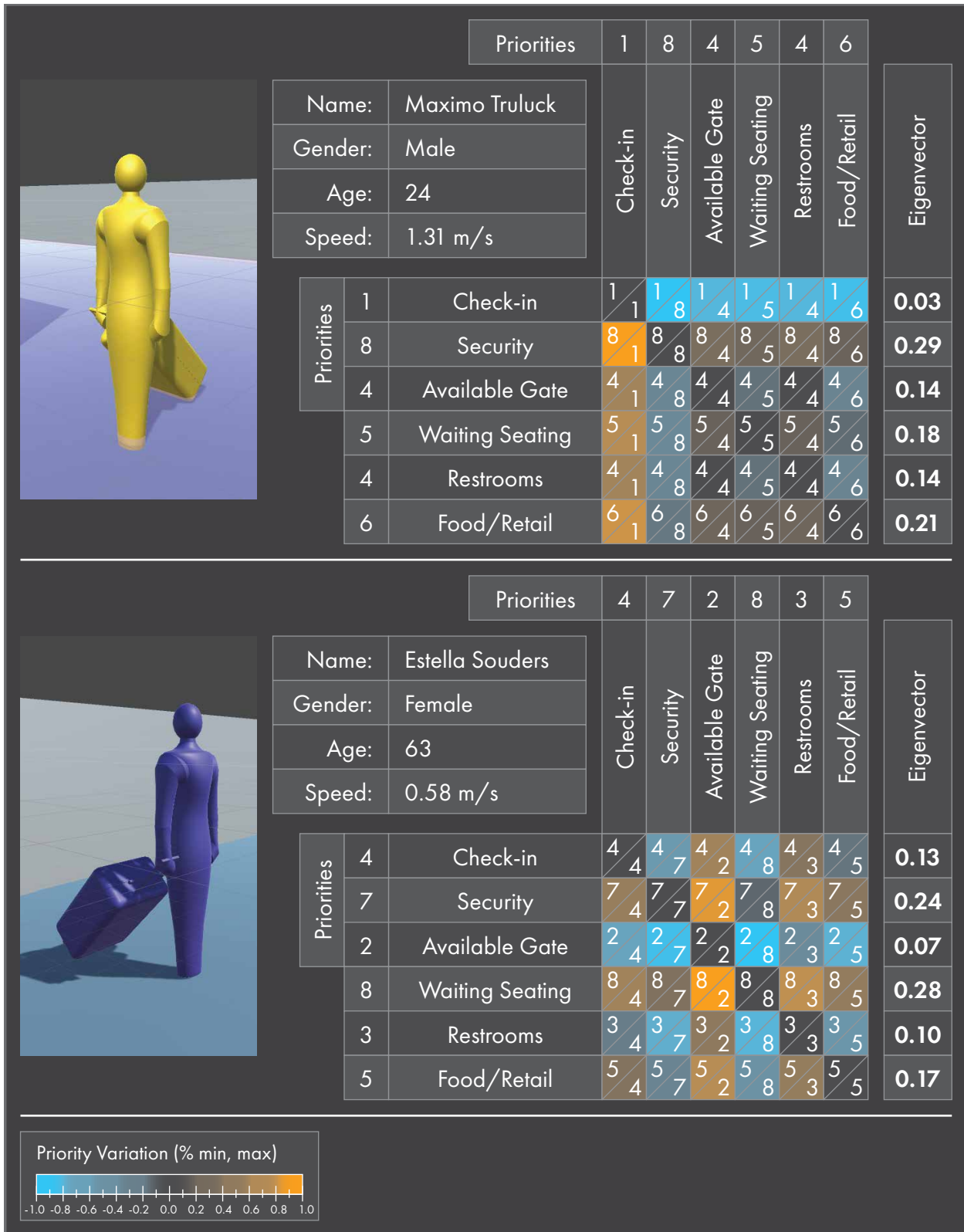


Figure 3.1.i: Samples of randomly assigned characteristics and priority matrices, page 2.

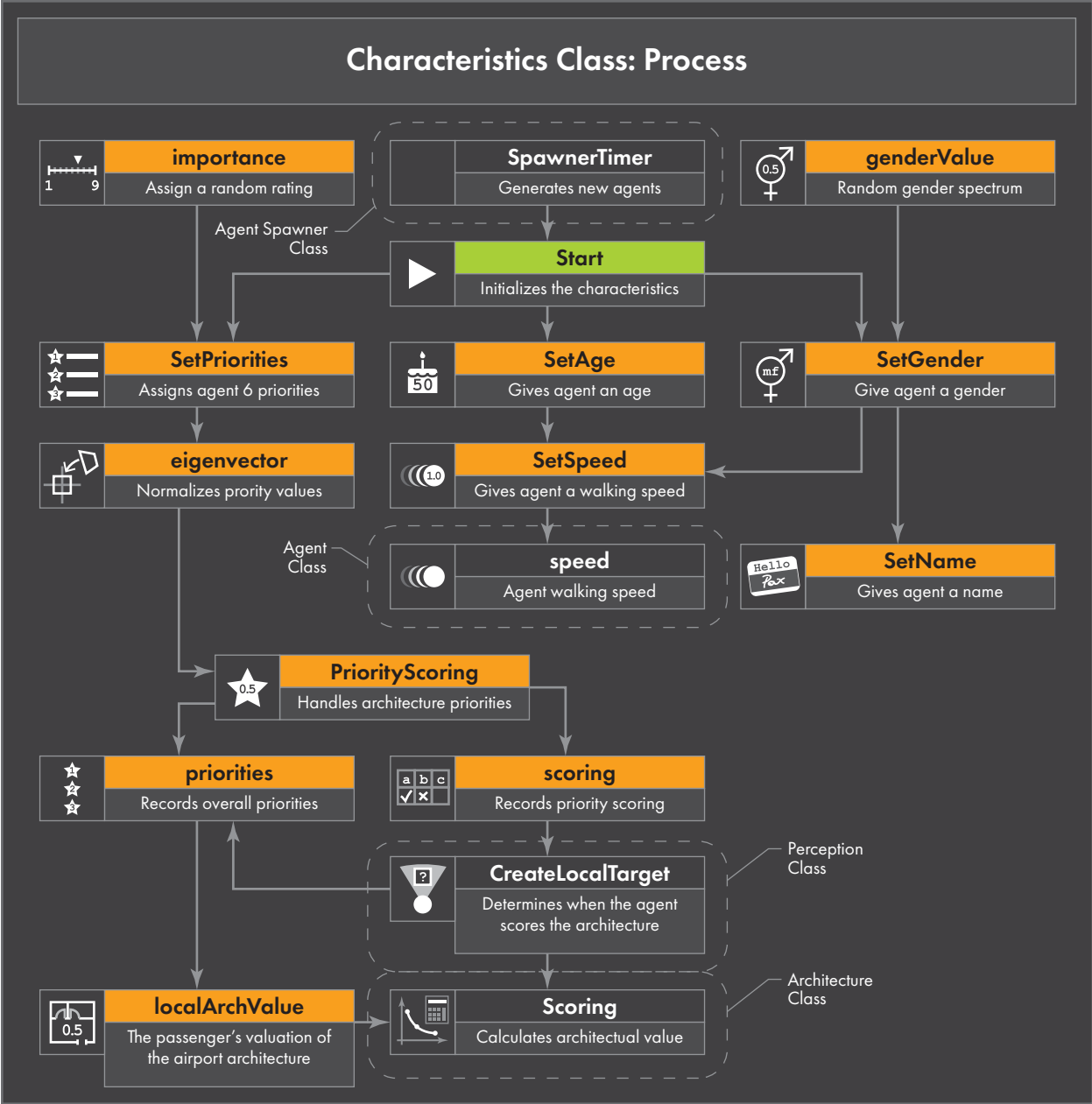


Figure 3.1.j: Process logic for the characteristics class.

Characteristics Class: Variables and Methods

Variables











	Variable	Description
	Type	
	Values	
	maleNames Array of strings -	Reference to random list of male names as text
	femaleNames Array of strings -	Reference to random list of female names as text
	personName string -	Represents the name of the agent or passenger
	genderValue float 0.0 - 1.0	Provides a random value within the gender spectrum
	gender string male, female	Holds the name of the type of gender
	age float 18 - 72	Holds the passenger's age value
	walkingSpeed float 0.56 - 1.85	Represents the passenger's walking speed
	walkingDisability1 bool true, false	Condition for walking disability type 1; based IMO standard for evacuation simulations
	walkingDisability2 bool true, false	Condition for walking disability type 2; based IMO standard for evacuation simulations

Figure 3.1.k: Key variables for the characteristics class, page 1.













	priorityNames Array of string -	Reference to a list of passenger airport priorities
  	priorities Priority (local class) 1 - 9 and 0.0 - 1.0	A collection of priorities important to the passenger in the airport and the corresponding values
	localArchValue float 0.0 - 1.0	Is the passenger's score of the airport or architecture
Methods		
	Method Type Input Output	Description
	SetGender void genderValue gender	Determines the agent's gender
	SetName void gender personName	Determines the name of the agent based on gender
	SetAge void - age, walkingDisability	Determine the agent's age; based on IMO number of passengers table 3.1
	SetSpeed void gender, age walkingSpeed	Determines the walking speed of the agent; based on IMO walking speeds table 3.4
	SetPriorities void - priorities	Calculates the agent's priorities and associated weighting
	PriorityScoring void priorities localArchValue	Calculates the agent's architectural score based on the priorities

Figure 3.1.1: Key variables and methods for the characteristics class, page 2.







Local Class: Priority		
	Variable	Description
	Type	
	Values	
	name string -	The name of the priority as text
	importance int 1 - 9	The importance value the passenger gives the priority (on a scale of 1 to 9)
	eigenvector float 0.0 - 1.0	The normalized eigenvector weighting of the priority (w_n)
	score float 0.0 - 1.0	The recorded score of the perceived priority (y_n)
	Value float 0.0 - 1.0	The product of the eigenvector weight and the priority score ($P = w_n y_n$)

Figure 3.1.m: Priority local class within the characteristics class, page 3.

Perception Class

The perception class is like the agent's mind, it decides what the agent does when they observe outside information. The primary job of the perception class is to create a *local target* for the agent's navigation. Objects that the agent observes are identified and categorized in the field of view class. Then the perception class chooses which of these objects are most relevant to the agent and selects the actions that best suits the situation. Additionally, the perception class manages agent behaviour states over time and controls architectural valuation.

The perception class responds based on several types of objects. The first condition is if the agent sees their final destination. The perception class sets the local target at the destination so the agent can walk there. If perception navigation is disabled in the agent class, then this decision process is bypassed, and a local target is generated at the agent's final destination from the beginning. Once the final destination is observed, the perception class stores this location in the agent's memory.

The second condition is if the agent sees signage (Fig.3.1.n). Firstly, if the agent is not in front of the sign already, the perception class makes the agent walk up to the sign so they can read it. The act of reading the sign works by referencing the signage class that is inherent in every wayfinding object. The perception class checks if the information from the signage class matches the agent's knowledge of their final destination, like an assigned gate number. If this information is the same, the perception class sets the local target based on the vector where the sign is pointing to. After reading the sign, agents keep a short-term memory of the direction the sign was pointing. However, after reaching their next local target, the agent's memory of the signage is reset so they can learn about new information or re-read the same signage again if the agent is lost.

The third condition is if the agent sees *airport objects*, or any significant elements that are part of the terminal. This includes service counters, kiosks, devices, or seating (Fig.3.1.o). This can also include architectural features like walls, doors, or circulation. When an agent sees an airport object, the perception class performs two actions. The first action uses the object to identify what type of architectural space the agent is currently standing in. In Unity, game objects can be assigned a tag. All airport objects are tagged with the architectural space they are a part of. For example, the screening machines are tagged with *Security*, and the waiting area chairs are tagged with *Gate*. If the agent observes any of these objects, then the perception class records what the tag says, which represents the current architectural space. This is helpful for

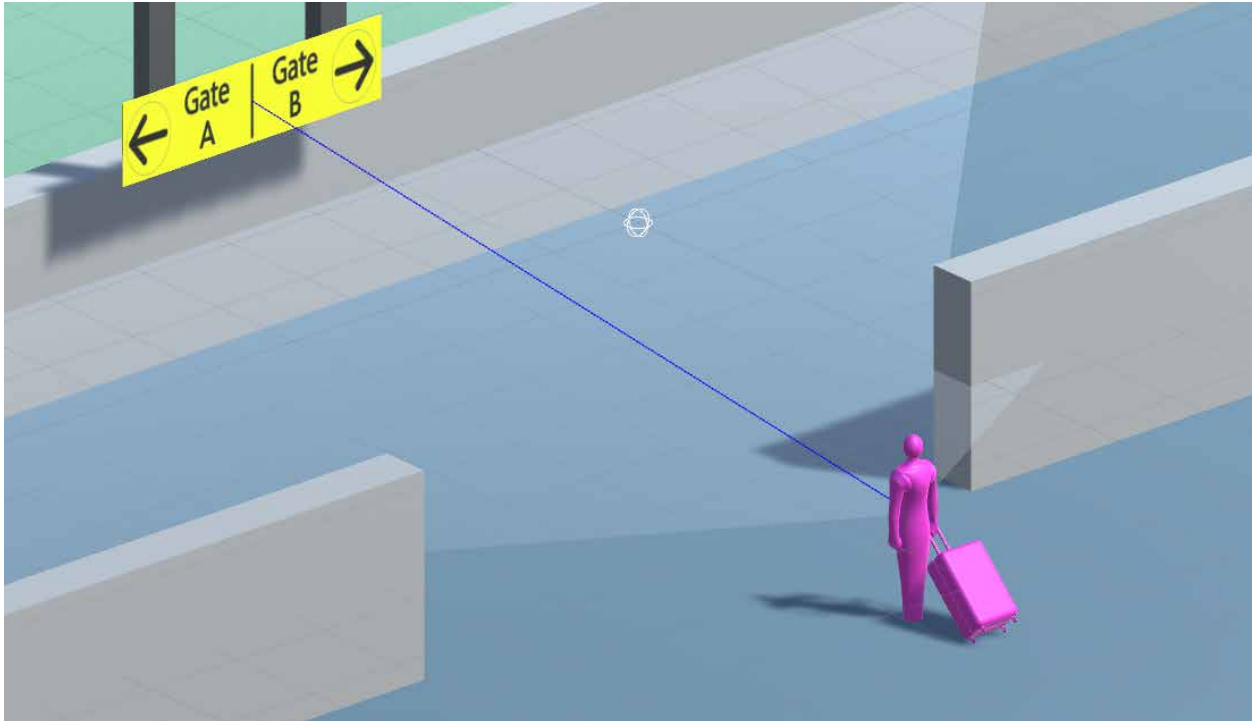


Figure 3.1.n: Agent perceives the gate sign, as shown by the blue line. The agent state is "read sign", as illustrated by the pink colour.

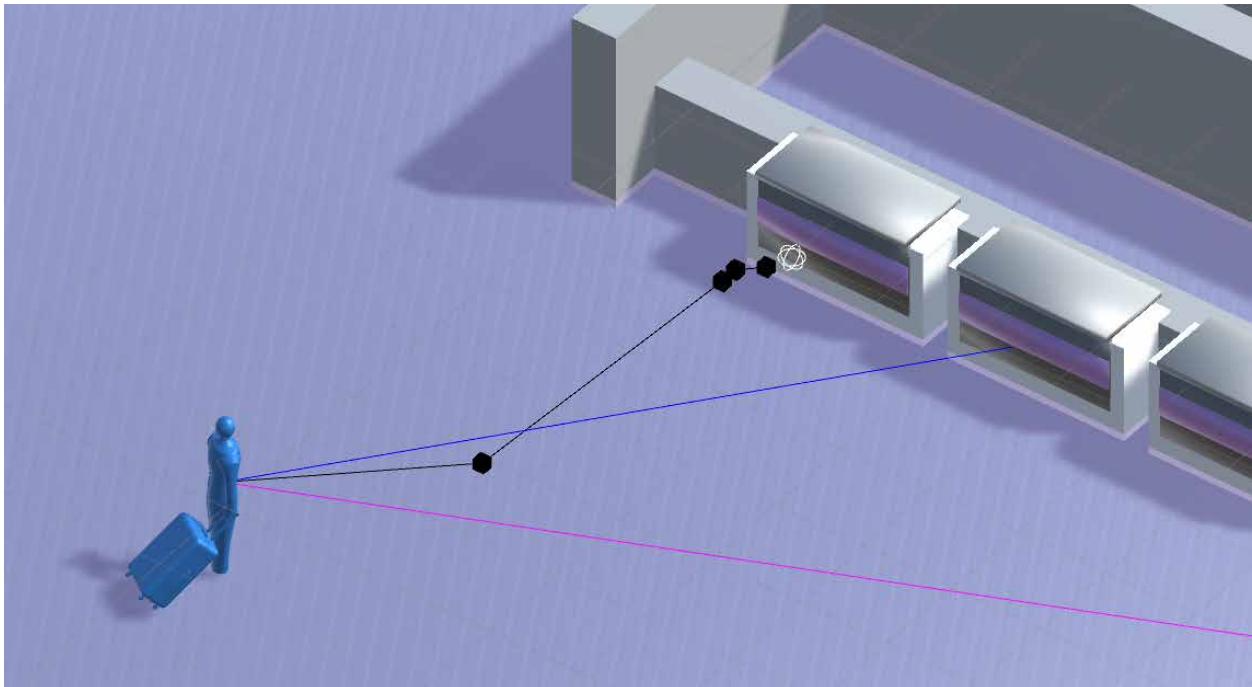


Figure 3.1.o: Agent perceives check-in counters, as shown by the blue line. The agent state is "go to check-in", as illustrated by the blue colour.

deciding what type of behaviour the agent is expected to do in these areas. It also acts as a way for the agent to remember where they are. The simulation assumes that if the agent can see the object, then the agent has full knowledge of what it is and what space it represents. The second action the perception class performs is checking the agent's itinerary class to decide what they need to do next. Each agent has an itinerary that provides a checklist for the tasks they need to complete before boarding their flights. This includes actions like getting a boarding pass or getting cleared through security. If the agent observes objects that are tied to a required task in their itinerary, then the perception class makes the agent interact with those objects, like walking through the screening devices for security clearance. The perception class can only make the agents interact with these objects if it is required in their itinerary. Otherwise, the agent will wander around them. In each situation, the perception will set the agent behaviour state according to the action, like waiting, queuing, or processing.

Once the perception class decides which condition to interact with, the perception class then calculates the agent's corresponding architectural score. For example, if the score involves a processing time, then the perception class will control when to start and stop the timer for that action. Likewise, the perception class also controls how architectural visibility is recorded during the process. Once the agent gets to their final destination, the perception class sends the agent's final architectural score to the global architecture class, which compiles all agent values.

If the perception class does not recognize a suitable local target, like signs or airport objects, then the agent resorts to wandering around. The act of wandering works by giving the agent a random direction to walk in, which is typically a couple metres in front of their current position. The perception class calls the field of view class to provide a random vector within the agent's current perspective. It then creates a new local target at the randomly chosen location. After the perception class makes these choices, the local target information is sent to the agent class for navigation. This process repeats every time the agent reaches their local target, which requires the perception class to make new choices as the surroundings change over time.

In summary, the perception class decides what the agent does. It chooses a response based on the type of object the agent can see and records this information in memory. The perception class updates agent states and architectural values accordingly. If the agent does not perceive anything important, then it makes the agent wander around until it sees something new.

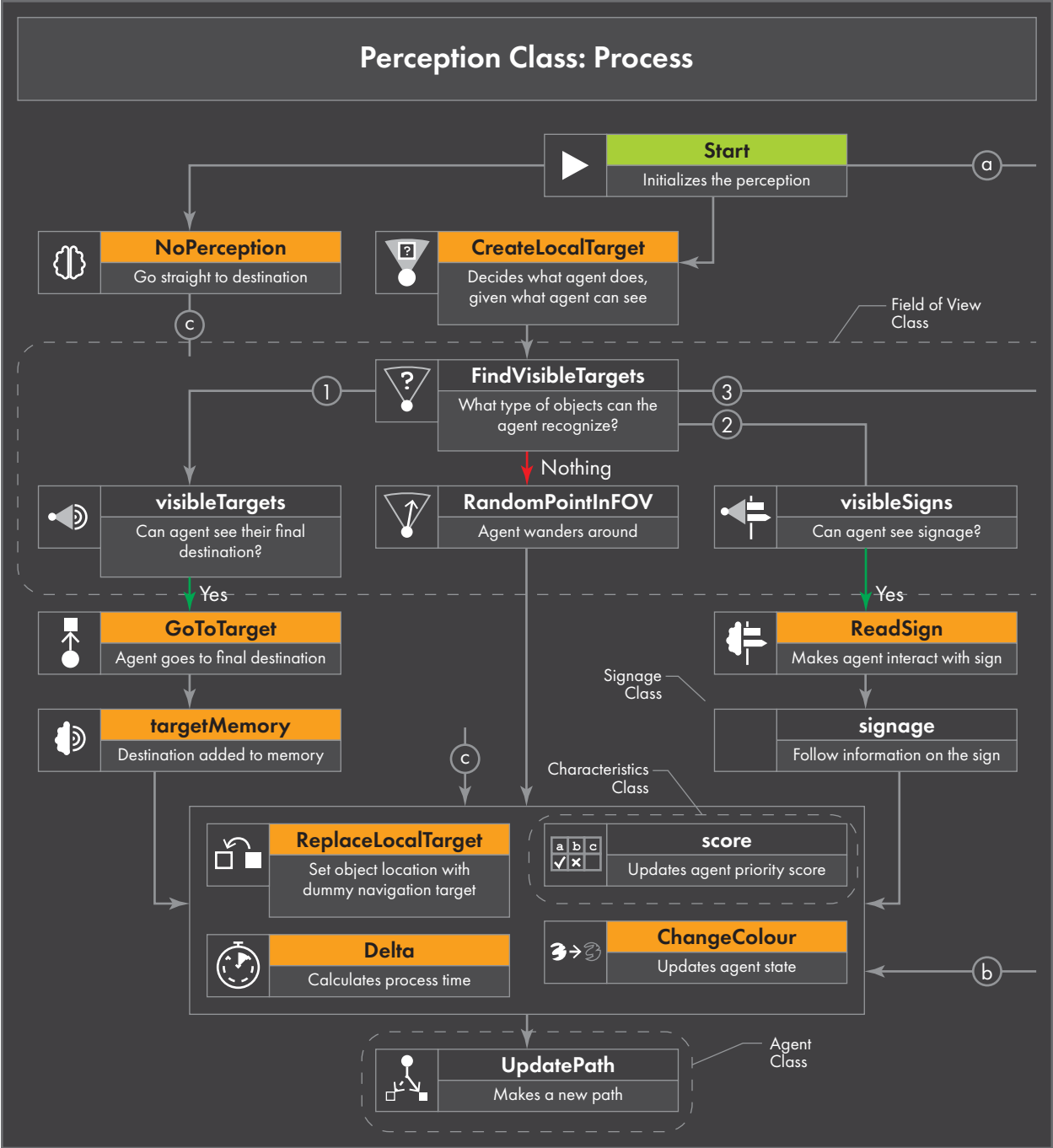


Figure 3.1.p: Process logic for the perception class, page 1.

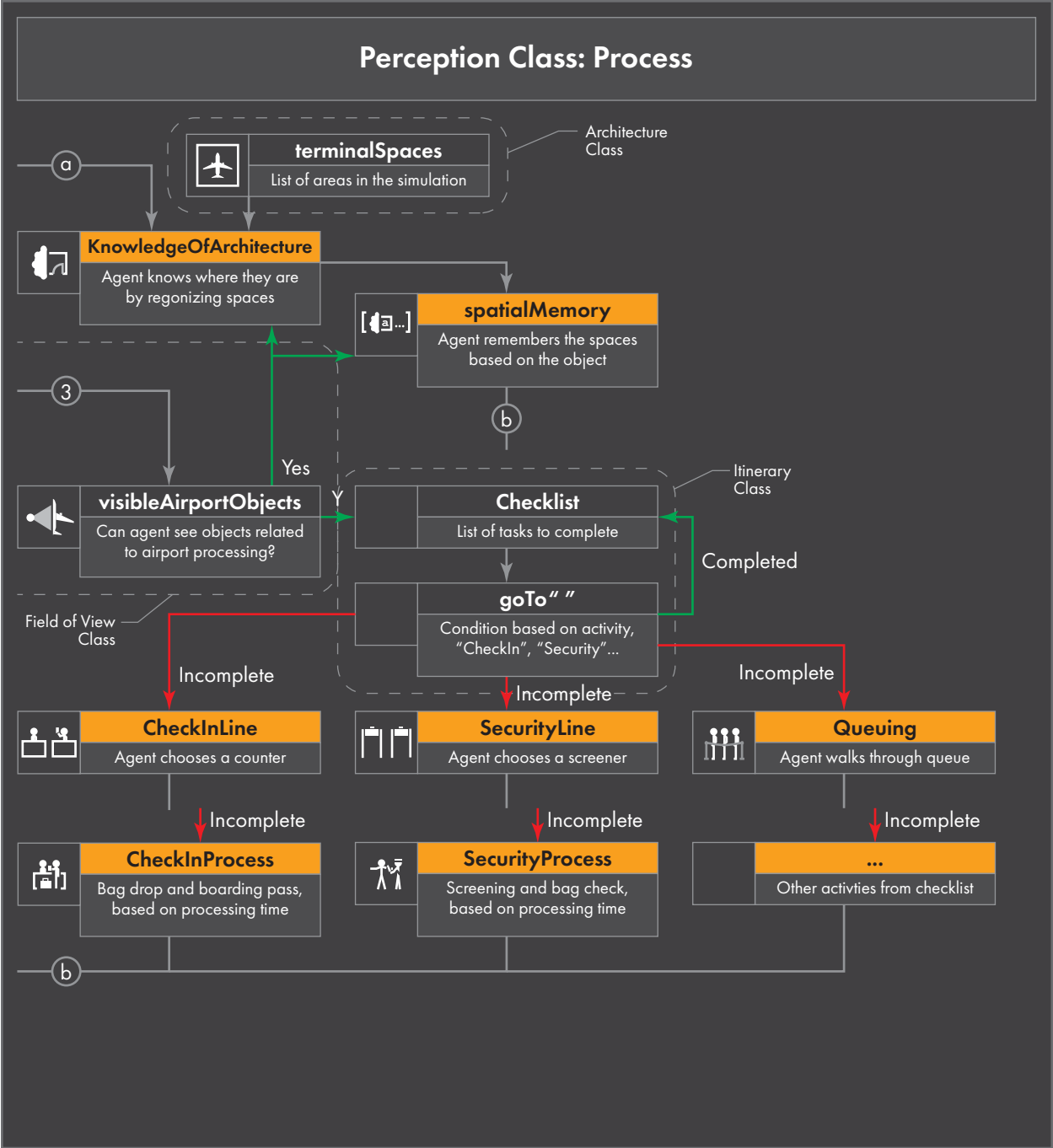


Figure 3.1.q: Process logic for the perception class, page 2.











Perception Class: Variables and Methods		
Variables		
	Variable Type Values	Description
	agent Agent -	Reference to the agent class
	fov FieldOfView -	Reference to the agent's field of view class
	itinerary Itinerary -	Reference to the agent's itinerary class
	arch Architecture -	Reference to the architecture class
	character Characteristics -	Reference to the agent's characteristics class
	objectRender Array of Renderers -	Reference to the agent's rendering for display colouring
	currentSpaces GameObject -	Reference to the architectural spaces in the world
	targetMemory bool true, false	Memory reference to the location of the agent's target
	exitMemory bool true, false	Memory reference to the location of the agent's gate or exit

Figure 3.1.r: Key variables for the perception class, page 1.













	<pre>" "SearchStartTime float 0.0 - ∞</pre>	Time starting search, with respect to any variable: "check-in", "security"
	<pre>" "SearchEndTime float 0.0 - ∞</pre>	Time ending search, with respect to any variable: "check-in", "security"
	<pre>" "StartTime float 0.0 - ∞</pre>	Time when the agent starts a process, with respect to any variable: "check-in", "security"
	<pre>" "EndTime float 0.0 - ∞</pre>	Time when the agent ends a process, with respect to any variable: "check-in", "security"
	<pre>" "StartDirChange float 0.0 - ∞</pre>	Starting number of direction changes, with respect to any variable: "check-in", "security", "gate"
	<pre>" "EndDirChange float 0.0 - ∞</pre>	Ending number of direction changes, with respect to any variable: "check-in", "security", "gate"
	<pre>" "AwareStartTime float 0.0 - ∞</pre>	Time when the agent knows where to find their target, with respect to any variable: "gate"
	<pre>" "AwareEndTime float 0.0 - ∞</pre>	How long since the agent got information about their target, with respect to any variable: "gate"
	<pre>directionChanges float -</pre>	Stores the number of times the agent makes a move in a new direction
	<pre>spatialMemory List of strings -</pre>	Empty list to store agent's memory of where they are
	<pre>ratioList List of floats -</pre>	Empty list to store agent's visibility
	<pre>isPerceptive bool true, false</pre>	Whether or not the agent navigates by perception or direct path; set to perceptive by default

Figure 3.1.s: Key variables for the perception class, page 2.













	<p>possibleStates</p> <p>List of strings</p> <p>-</p>	List of possible agent states as text
	<p>agentState</p> <p>string</p> <p>-</p>	The current state of the agent as text
	<p>displayColourState</p> <p>bool</p> <p>true, false</p>	Toggle to display the agent in colour states or random colours
	<p>colours</p> <p>List of Colors</p> <p>-</p>	Stores agent states as colours
	<p>wanderingColour</p> <p>Color</p> <p>RGB: 153, 217, 255</p>	State display colour while agent is wandering around or searching
	<p>toCheckInColour</p> <p>Color</p> <p>RGB: 0, 77, 166</p>	State display colour while agent is going to the check in area
	<p>toSecurityColour</p> <p>Color</p> <p>RGB: 166, 0, 255</p>	State display colour while agent is going to the security area
	<p>queuingColour</p> <p>Color</p> <p>RGB: 255, 217, 0</p>	State display colour while agent is queuing or in a line
	<p>processingColour</p> <p>Color</p> <p>RGB: 255, 128, 0</p>	State display colour while agent is processing or interacting with an object
	<p>readSignColour</p> <p>Color</p> <p>RGB: 255, 102, 255</p>	State display colour after agent has read a sign
	<p>waitingColour</p> <p>Color</p> <p>RGB: 255, 0, 0</p>	State display colour while agent is waiting
	<p>boardingColour</p> <p>Color</p> <p>RGB: 20, 115, 0</p>	State display colour while agent is boarding

Figure 3.1.t: Key variables for the perception class, page 3.












Methods		
	Method Type Input Output	Description
	AddColours void colours colours	Function adds colours states to a list
	ChangeColour void objectRender newColour	Function changes the colour of the agent for a given state and colour name
	NoPerception void isPerceptive -	Navigates agent directly to destination, if agent is not navigating by perception
	CreateLocalTarget void FindVisibleTargets ReplaceLocalTarget	Controls where the agent decides to go, given what objects are visible to the agent and what the agent is doing
	ReplaceLocalTarget void targetPosition localDummy	Replaces a local target position with a dummy target in the agent's field of view
	GoToTarget void visibleTargets score	Sends agent to final destination, changes state, and records a score
	ReadSign void signage ReplaceLocalTarget	Controls interaction with signage, and determines how agent respond to given information
	Queuing void newLine ReplaceLocatTarget	Function handles interaction with queue lines
	CheckInLine void newLine ReplaceLocatTarget	Handles the selection of a check-in counter
	CheckInProcess void itinerary score	Handles the check-in processing, agent states, and scoring

Figure 3.1.u: Key methods for the perception class, page 4.

	SecurityLine void newLine ReplaceLocatTarget	Handles the selection of a security screener
	SecurityProcess void itinerary score	Handles the security screening, agent states, and scoring
	KnowledgeOfArchitecture void currentSpaces spatialMemory	Controls what the agent does when it sees architectural elements, and adds the architecture to memory
	Delta float _start, _end finalTime	Calculates how much simulation time the agent spent doing something given a start and end time in seconds
	LookDirection Vector3 _pos, _target _direction	Function that makes the agent look in the direction of an object
	AddVisibilityToList IEnumerator ratioList ratioList	Adds the current visibility value to a list (updated once per second)
	AverageVisibility float ratioList average	Determine the average visibility values from the visibility ratio list
	DirectionChangeRatio float directionChanges ratio	Calculates the ratio of direction changes

Figure 3.1.v: Key methods for the perception class, page 5.

Field of View

The field of view class is responsible for creating the isovist geometry, identifying objects that are visible to the agent, and providing vector locations for local targets. The thesis's field of view class is a modified version of Lague's class for their Unity game tutorial on *Field of View Visualizations (2015)*.^[4] Lague's work provides the logic of the isovist geometry, the in-game display of the field of view, and the method for identifying visible objects. The thesis's contribution to the field of view class includes categorizing objects based on airport architecture, the method for agent wandering, and the calculation of visibility area.

The field of view is based on the geometry of an isovist. Lague built the isovist from a collection of numerous *raycasts*, or vectors, projected from the agent's position (Fig.3.1.w). The location where the vectors intersect with an object are called the *viewpoints*. The viewpoint vectors are combined into thin triangles, whose vertices are defined in an array based on the triangle number. This can be generalized for any number of raycasts. The more raycasts there are, the more viewpoint vectors, and therefore, the more refined the agent's field of view can be. In some cases, convex corners of an object may not be captured properly if there are not enough viewpoints. Lague solves this issue by performing an iterative search around an object's corners. The search repeatedly calculates midpoint vectors between max and min viewpoints, until the difference between the corner and midpoint vector is within a small enough tolerance (Fig.3.1.y).

Objects within an agent's field of view are detected based on the interaction between object layers. In Unity, all objects can be assigned a layer, which is defined manually in the model properties. Each type of object in the airport environment is given a unique layer. This includes layers for targets, obstacles (walls, partitions), signage, other agents, and airport objects (kiosks, counters, gates). The field of view class projects out raycast vectors from the agent within the radius of the given field of view. When the simulation begins, these raycasts are sent out repeatedly over time (around 5 raycasts per second). There are raycasts specifically looking for each of the defined layers. If a given raycast collides with an object that has the same layer, then that object's information (name, position) is stored into a corresponding list for that object type. These lists are then sent to the perception class for the agent to interpret.

A key function of this thesis's agent model is the ability for agents to wander around if they are lost. The act of wandering, or searching, involves repeatedly selecting random direction vectors

4. Lague, Sebastian. "Field-of-View/Episode 02/Scripts/FieldOfView.cs". GitHub. December 28, 2015. <https://github.com/SebLague/Field-of-View/blob/master/Episode%2002/Scripts/FieldOfView.cs>.

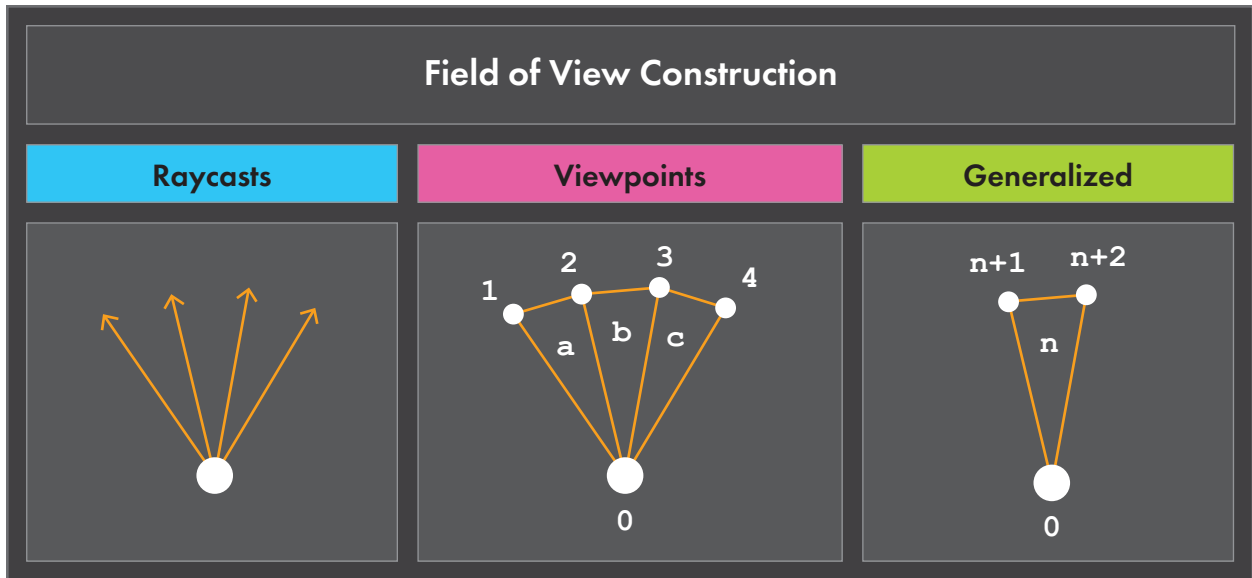


Figure 3.1.w: Generalized construction of the field of view, based on diagram by Laque (2015), redrawn by author.



Figure 3.1.x: Agent in front of a wall showing their field of view.

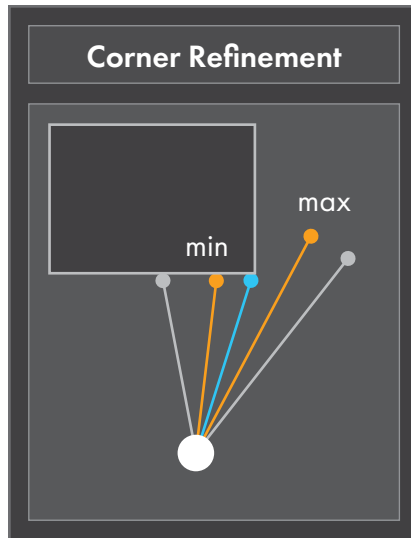


Figure 3.1.y: Convex corners are refined by selecting a midpoint vector between a max and min viewpoint, based on diagram by Lague (2015), redrawn by author.

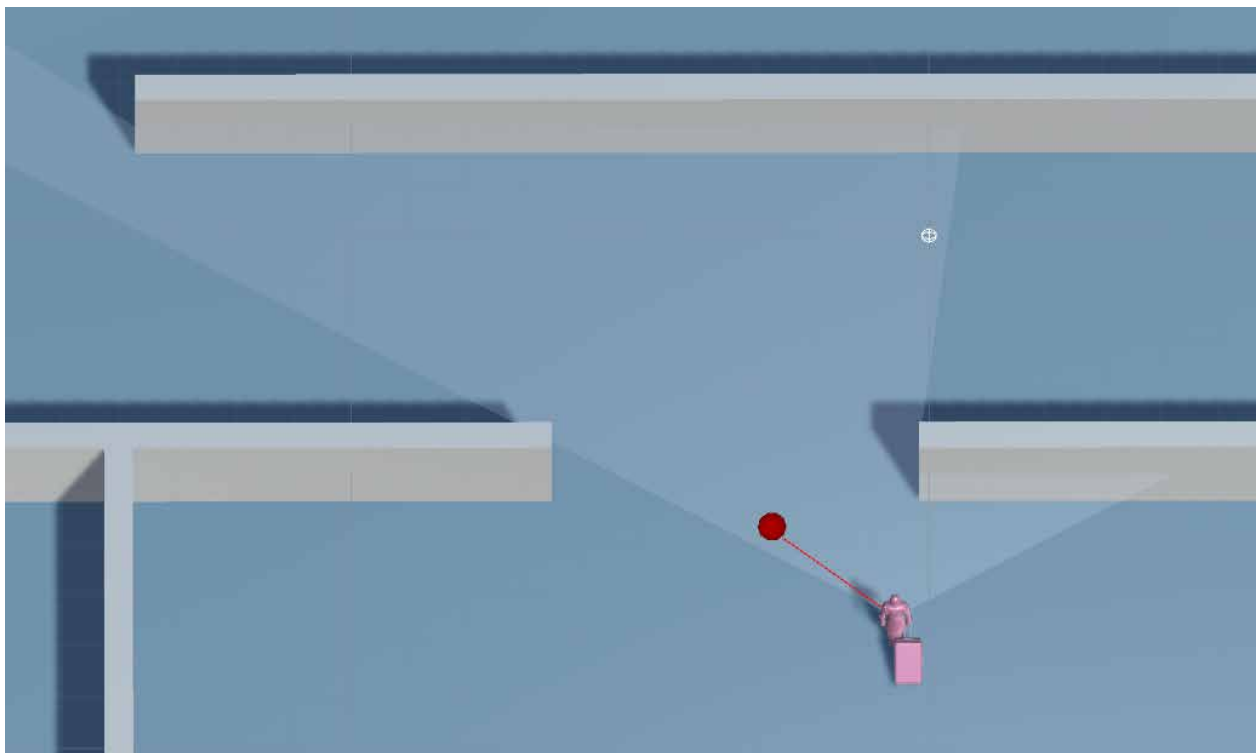


Figure 3.1.z: A random direction vector for wandering is selected towards the longest visible direction, illustrated by the red line and sphere.

within the field of view for the agent to walk to. The function imitates wandering by firstly selecting the longest vector in the agent's field of view. This makes sure the agent keeps walking to the end of a given corridor or moving towards open areas if they encounter confined spaces or split paths. The wandering function searches for the longest vector starting from directly in front of the agent, then alternating searching viewpoints to the right and left of straight ahead. Once the longest vector is found, another viewpoint is randomly selected within a range greater than or less than the longest vector. This allows agents, who may be lost, to not directly follow one path, which is more natural of wandering. The function also naturalizes wandering by selecting random distances to walk, ranging anywhere from one to four metres ahead (Fig.3.1.z). If an agent encounters a dead end, then the function provides a direction vector in the opposite direction to the wall or obstacle. Once a random direction vector is established, this is sent back to the perception and agent classes for navigation.

The final responsibility of the field of view class is to calculate the average visibility of a given space. The area of an isovist is approximated as a sum of viewpoint triangles. As the agent moves throughout the simulation, the field of view class constantly updates this calculation. The average visibility is based on the current area the agent can see divided by the maximum area the agent observes over time. For example, if the agent moves from an open courtyard to a small corridor, then the visibility ratio will be low. By contrast, moving from a confined area into an open area would give a high visibility ratio, which is more desirable. As the agent encounters different sized spaces, the field of view class updates the maximum area. The final visibility ratio is then sent to the perception class for calculating the average visibility of a particular space, as needed.

In summary, the field of view class is responsible for creating the isovist geometry, identifying visible objects, and calculating the visibility. The thesis's field of view class is based on the class created by Lague for his Unity tutorial on field of view visualizations. The field of view geometry is created from raycast vectors projected from the agent by combining viewpoints into a collection of thin triangles. Objects are identified using layers, which is assigned based on the object type, and they are detected using raycasts that are constantly projected from the agent over time. The class also determines a random direction vector within the agent's field of view, to imitate wandering behaviour, if the agent does not observe any relevant objects. Finally, the visibility of a given space is calculated based on the change in field of view area over time, which is then sent to the perception class for final calculation.

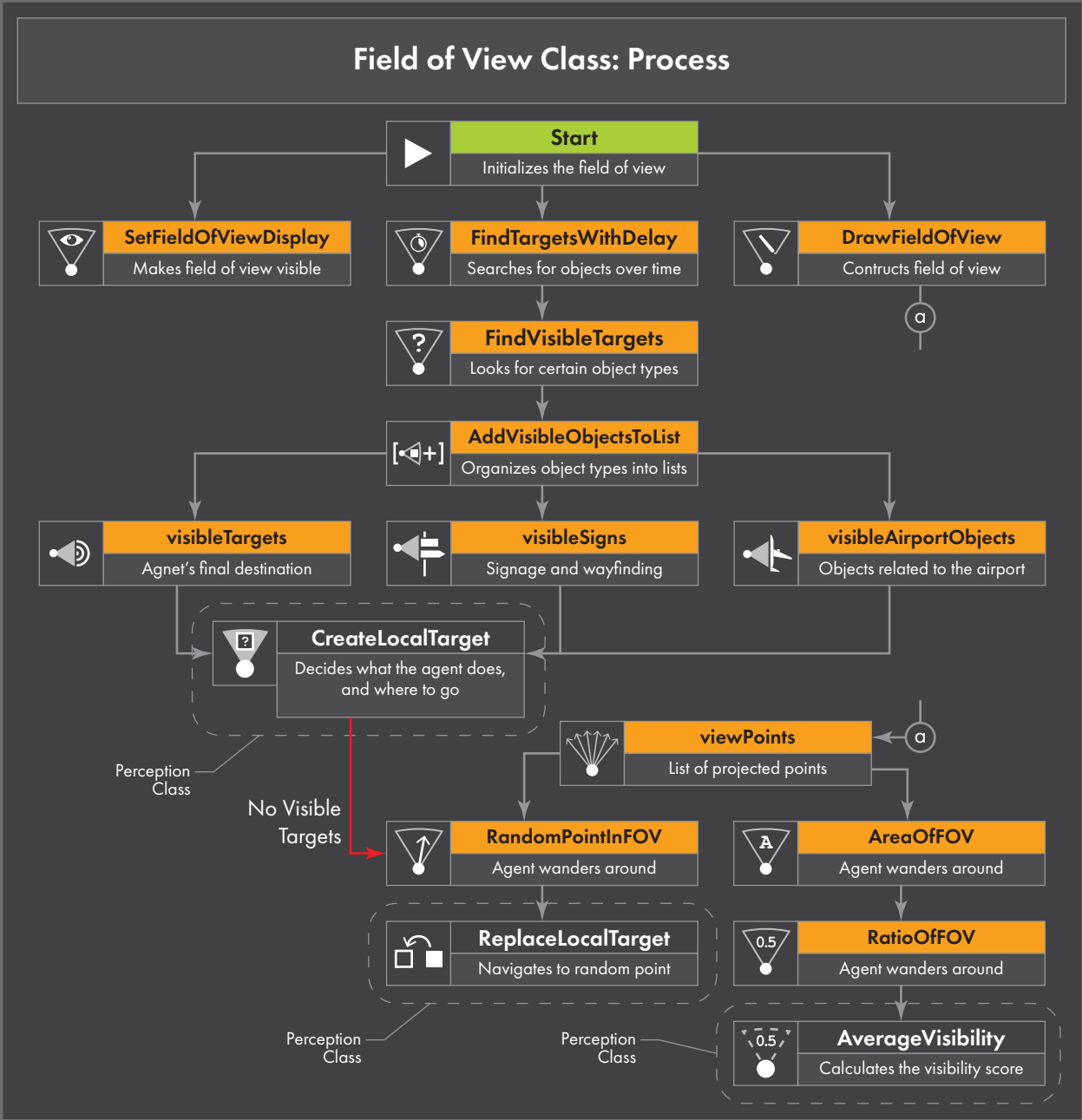


Figure 3.1.za: Process logic for the field of view class.

Field of View Class: Variables and Methods

Variables






	Variable	Description
	Type	
	Values	
	viewRadius float 1.0 - ∞	Radius of the agent's field of view
	viewAngle float 0 - 360	Visible angle range of the agent's field of view
	targetMask LayerMask -	Game layer assigned to targets that the agent is looking for
	obstacleMask LayerMask -	Game layer assigned to obstacles that the agent cannot walk through
	signMask LayerMask -	Game layer assigned to signs that the agent can read for information
	agentMask LayerMask -	Game layer assigned to other nearby agents
	airportObjectsMask LayerMask -	Game layer assigned to airport objects (counters, benches, kiosks, etc.)
	visibleTargets List of Transforms -	List containing targets the agent can see
	visibleObstacles List of Transforms -	List containing obstacles the agent can see (anything that prevents the agent from walking)

Figure 3.1.zb: Key variables for the field of view class, page 1.


	visibleSigns List of Transforms -	List containing signage the agent can see
	visibleAgents List of Transforms -	List containing other nearby agents that the agent can see
	visibleAirportObjects List of Transforms -	List containing relevant airport objects that the agent can see
	meshResolution float 10	The ratio of view angle to the number of view point lines (i.e. 160 degrees gives 1600 view points)
	edgeResolveIterations int 4	The number of times the location of the edge of an object is refined
	edgeDstThreshold float 0.5	The distance tolerance when finding the edge of an object
	currentMaxAreaOfFOV float 1	The initial field of view area when calculating visibility ratio (1 m ²)
	viewMeshFilter ViewCastInfo -	Stores an in-game mesh filter for rendering
	viewMesh Mesh -	Reference to the mesh class
	viewPoints List of Vector3 -	List of projected points from the agent in a field of view
	displayFieldOfView bool true, false	Toggle for displaying the field of view during the simulation

Figure 3.1.zc: Key variables for the field of view class, page 2.










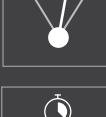

Methods		
	Method Type Input Output	Description
	FindTargetsWithDelay IEnumerator delay FindVisibleTargets	Reoccurring function determines when the agent searches for objects in their field of view
	SetFieldOfViewDisplay void displayFieldOfView -	Controls if the field of view is displayed during the simulation
	FindVisibleTargets void - AddVisibleObjectsToList	Manages lists for collecting certain visible object types
	AddVisibleObjectsToList void visibleObjects, objectMask -	Adds any type of object that is visible to the agent into a new list, given the list name and object layer
	DrawFieldOfView void viewPoints viewMesh	Constructs the field of view as a collection of connected triangles between view points
	MaxAreaOfFOV float viewRadius, viewAngle area	Determine the theoretical maximum field of view area
	AreaOfFOV float viewPoints area	Calculates the area of the agent's current field of view
	RatioOfFOV void AreaOfFOV ratio	Determines the ratio of the current field of view area to the maximum area observed by the agent
	RandomPointInFOV void newLine ReplaceLocatTarget	Calculates a random point vector in the agent's field of view; for the purpose of wandering
	ProximityDelay IEnumerator _obstacle, _distance _close	Function delays finding a new point in the field of view if the agent is standing too close to another agent

Figure 3.1.zd: Key methods for the field of view class, page 3.

	FindEdge EdgeInfo (local struct) minViewCast, maxViewCast minPoint, maxPoint	Function determines the edge of an object in the field of view
	ViewCast ViewCastInfo (local struct) globalAngle ViewCastInfo	Determines the location of a view point, if it hits an object, given a view angle
	DirFromAngle Vector3 angleInDegress Vector3	Determines the global vector location given an angle
	ViewCastInfo struct hit, point, dst, angle	Local structure properties that defines a direction vector, angle, and if it intesects an object
	EdgeInfo struct pointA, pointB	Local structure properties that defines the start and end of a line or edge

Figure 3.1.zc: Key methods for the field of view class, page 4.

Chapter 3.2

A* Pathfinding Classes

The thesis uses an A* search algorithm to determine agent pathfinding and navigation in the simulation environment. All of the thesis's A* classes are based on Sebastian Lague's Unity game tutorial on *A* Pathfinding (2016)*.^{[1][2]} The key classes for A* Pathfinding are *grid*, *node*, and *pathfinding*. Lague gives a thorough description of how the A* pathfinding is made. Therefore, the following chapter gives a summary of how it applies to the thesis's simulation process.

Grid Class

The simulated environment is built as a grid of tiles, which the agent uses to navigate. The grid class defines these tiles as a node structure, or graph, which establishes the network for the A* algorithm. In the Unity scene, the grid class is assigned to an empty game object. The user of the simulation sets the node (tile) size, as a radius, and the world size as the number of tiles in the x and y directions. For the thesis's simulations, the node size is set to 0.1 m, which means each tile is 20 cm wide or 25 tiles per square metre. This provides a balance between mapping intricate architectural spaces and computation time (Fig.3.2.a).

Before the simulation starts, the user must assign environment objects to either walkable or unwalkable layers and corresponding penalties. In most cases, walkable layers are the floor levels and unwalkable layers are walls, columns, and partitions. Depending on the type of equipment, airport objects are also treated as an unwalkable area, like service counters. Most walkable regions have a penalty of zero, whereas unwalkable regions are undefined. Walkable areas can also be assigned a higher cost penalty (magnitude of 10 to 50), which can influence agent navigation, like restricted areas.

When the simulation is initialized, the grid class iterates through all the tiles in the world. It determines which tiles intersect with each walkable region in the scene environment. The grid

1. Lague, Sebastian. "Pathfinding/Episode 7-smooth weights/Assets/Scripts". GitHub. December 30, 2016. <https://github.com/SebLague/Pathfinding/tree/master/Episode%209%20-%20smooth%20path%2002/Assets/Scripts>.

2. Lague, Sebastian. "A* Pathfinding (E01: algorithm explanation)". Youtube. December 16, 2014. https://www.youtube.com/watch?v=-L-WgKMFuhE&list=PLFt_AvWsXlocq5Umv3pMC9SPnKjfp9eGW&index=1.

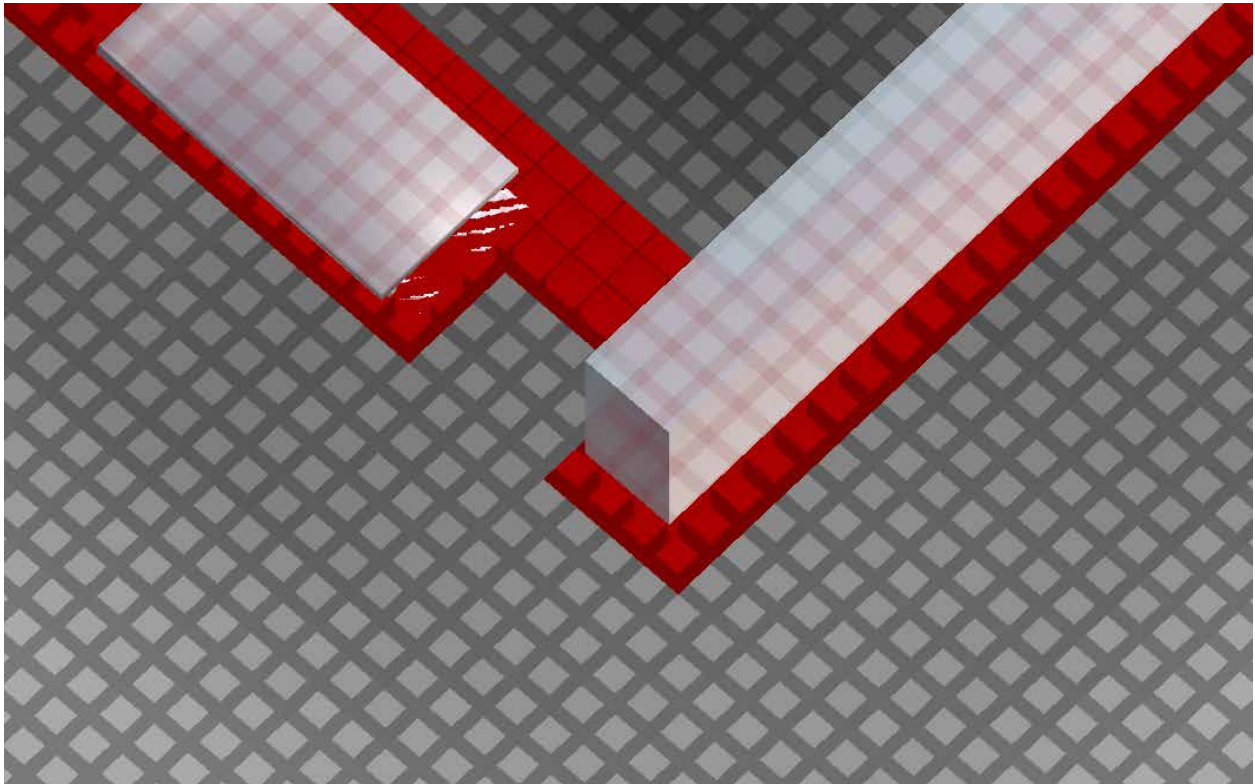


Figure 3.2.a: Simulated environments are divided into grid tiles for A* navigation. Every tile an object touches is considered part of the object's area, even if the object dimensions are smaller.

class also has a function that blurs the boundary between walkable and unwalkable areas. This results in a more naturalized walking path, which discourages agents from walking too close to unwalkable walls or high-cost areas. Once the grid is calculated, it is then sent to a prebuilt Unity function to be display in the scene. An example of a finished grid output can be seen in Fig.3.2.b. The walkable (low cost) areas are in light gray, the unwalkable areas are in red (walls), and there is a high-cost walkable area represented in black. Also note the blurring effect which creates a dark (high cost) gradient along the perimeter of the walls.

In summary, the simulated environment is built as a collection of grid tiles. This provides a graph node structure for the A* search algorithm. The tile, or node size, must be assigned by the user before the simulation begins. The environment must also be assigned into walkable and unwalkable areas, which determines where agents can navigate. Some walkable areas can be assigned higher cost values, which can influence agent navigation. There is also a blurring effect that naturalizes how close agents walk around walls.

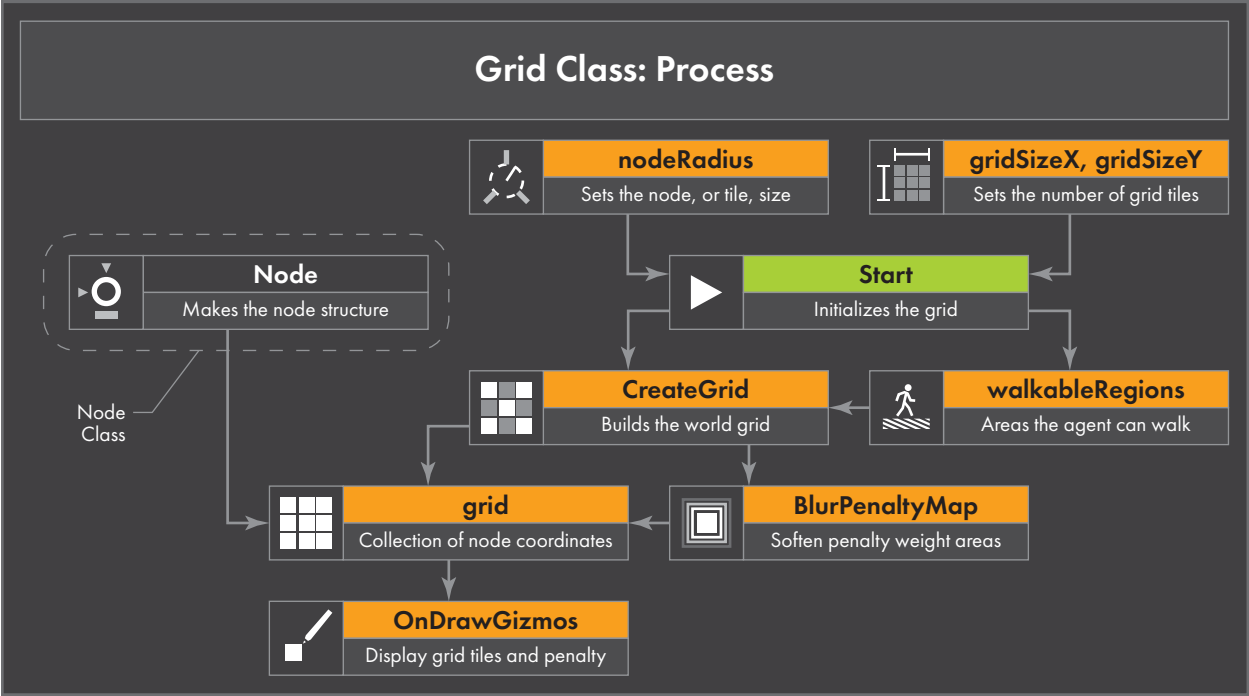


Figure 3.2.c: Process logic for the grid class.

Grid Class: Variables and Methods

Variables










	Variable Type Values	Description
	displayGridGizmos bool true, false	Toggle to display grid during the simulation
	unwalkableMask LayerMask -	Layer for objects that the agent cannot walk through
	gridWorldSize Vector2 (x, y)	Stores the size of the world, or number of grid tiles in two dimensions
	nodeRadius float 0.1	The radius size of the node, or grid tiles
	walkableRegions Array of TerrainTypes -	Area types that the agent can walk on
	obstacleProximityPenalty float 50	The cost of nodes around obstacles
	walkableRegionsDictionary Dictionary <key, value>	Stores a collection of areas that the agent can walk on
	walkableMask LayerMask -	Layer for objects that the agent can walk on
	grid Node [x, y]	Contains coordinates of node that make up the grid world

Figure 3.2.d: Key variables for the grid class, page 1.












	nodeDiameter float 2 x nodeRadius	The diameter of the node, or grid tile
	gridSizeX, gridSizeY int 0 - ∞	The number of grid tiles, or nodes, in the x and y directions
	penaltyMin, penaltyMax int 2147483647, -2147483648	The minimum and maximum values for a walkable penalty when burring neighbouring tiles
Methods		
	Method Type Input Output	Description
	MaxSize int gridSizeX * gridSizeY	Stores the total number of tiles in the world grid
	CreateGrid void gridSizeX,Y grid	Builds the grid based on walkable and non walkable objects intersecting with it
	BlurPenaltyMap void blueSize grid	Softens the transition between high and low penalty areas
	GetNeighbours List of Nodes node neighbours	Determines the neighbouring tiles given a certain node in the grid
	NodeFromWorldPoint Node worldPosition grid[x,y]	Determines a given node in the grid based on an object's vector position in the world
	OnDrawGizmos void grid -	Unity function that displays objects for debugging; to display penalty and grid depth map
	TerrainType class terrainMask, penalty	Local class representing different area types and penalty values

Figure 3.2.e: Key variables and methods for the grid class, page 2.

Node Class

The node class is responsible for managing the mathematical nodes of a graph and identifying physical points in space. The simulation uses nodes in a graph structure to build paths for agents to follow. These types of nodes define the areas that an agent can walk. Nodes are also used to identify the location of objects in architectural space. This includes the location of airport objects like service counters or queues, and architectural conditions like walls or thresholds.

Nodes are made up of two parts. The first part is the properties defined in the grid class. These are the node world position as a direction vector (x, y, z) , the co-ordinates on the grid as an index of x and y , the walkability (if the agent can walk on the node), and the movement penalty (to discourage agents walking). When identifying an architectural component, the number of nodes the component intersects with represents the space that it occupies. This information is either translated to agents as a vector location or an unwalkable area.

The second part is the costs associated for the A* pathfinding. Nodes are given three values to determine how useful a node is for navigating to a target. These values are referred to as the G cost, the H cost (heuristic), and the F cost. The G cost is the distance between the current node to the starting node, and the H cost is the distance between the current node and the target node. The F cost is the sum of the G cost and the H cost. This represents the total distance from the start to the target node, if the path goes through the given node. When the pathfinding function is evaluating any node, it considers both the total F cost and the movement penalty.

In summary, the node class creates node structures for mathematical graphs and physical spaces. Nodes can represent areas an agent can walk or the location of objects in space. If nodes are representing architectural features, then the area they occupy is translated into vector locations. A* pathfinding uses nodes structures to evaluate paths based on a cost system. The total cost of a node for travel is based on the distance to the start and target locations and a movement penalty.

Node Class: Variables and Methods

Variables



	Variable	Description
	Type	
	Values	
	walkable bool true, false	Checks if the agent can walk on the node
	worldPosition Vector3 (x, y, z)	The world space vector location of the node
	gridX, gridY int 0 - gridSizeX, Y	The grid position in the x and y directions
	movementPenalty int 0 - ∞	The node's weight, based on the grid penalty
	gCost int 0 - ∞	Distance from the current node to starting node
	hCost int 0 - ∞	The heuristic cost, or the distance from current node to the end node
	parent Node [x, y]	Reference to a neighbouring node where the new node is coming from
	heapIndex int 0 - ∞	Reference number for where the node is in the heap structure

Figure 3.2.f: Key variables for the node class, page 1.






Methods		
	Method Type Input Output	Description
	Node void walkable, position, grid, penalty	Constructs a node based on given properties
	fCost int gCost, hCost fCost	The total node cost as a sum of the gCost and hCost
	HeapIndex int heapIndex heapIndex	Gets and returns the heap index reference number
	CompareTo int nodeToCompare compare	Compares the value or cost difference between two nodes

Figure 3.2.g: Key methods for the node class, page 2.

Pathfinding Class

The pathfinding class decides what route the agent must follow to reach their target.

Fundamentally, this class is the A* search algorithm, which is responsible for calculating the lowest cost path between two points. The pathfinding class is also responsible for converting the list of nodes from the path into vector directions, which represents co-ordinates that the agent class can follow.

The pathfinding class begins by identifying two nodes on the grid network, one as the starting node of the path and the other as the target node. Lague's A* method works by sorting nodes into two categories, *open set* and *closed set*. The open set contains nodes to be evaluated, and the closed set contains nodes that are already evaluated. [3]

The goal of the A* method is to find the cheapest path from the starting node to the target node. Lague explains that the A* method begins its search from the starting node, which is labelled as the *current* node. The A* then checks each of the current node's *neighbours* to determine which one has the lowest total cost (F cost). The neighbour that has the lowest cost becomes the new current node. Once the new current node is selected, then the A* checks its new neighbours to find the next lowest cost node. This process repeats until the A* reaches the target node (Fig.3.2.i). If two nodes have the same cost, then the A* chooses the closest node to the start position. The A* also double checks if the node is walkable before evaluating it. If a node has a neighbour that has already been evaluated, or is in the closed set, then the A* skips it. However, if the A* discovers a shorter path to a node in a closed set, then it will update its F cost value.

Once the A* reaches the target node, and the node's costs are evaluated, the pathfinding class retraces the path back from the target node to the start node. The pathfinding class records each of the node's positions as a direction vector. These direction vectors are compiled into a list of waypoints. The path request manager then sends these waypoints to the agent class as a path structure, which the agent can start following.

In summary, the pathfinding class performs the A* search algorithm to find the lowest cost path between the agent and its target. The A* method involves repeatedly checking neighbouring nodes from the start node, to determine the nodes with the lowest cost path to the target. Once the A* finds the cheapest path, the nodes are converted into vector directions, which are compiled into a path of waypoints that is then sent to the agent to follow.

3. Lague. "E01: algorithm explanation". 7:45.

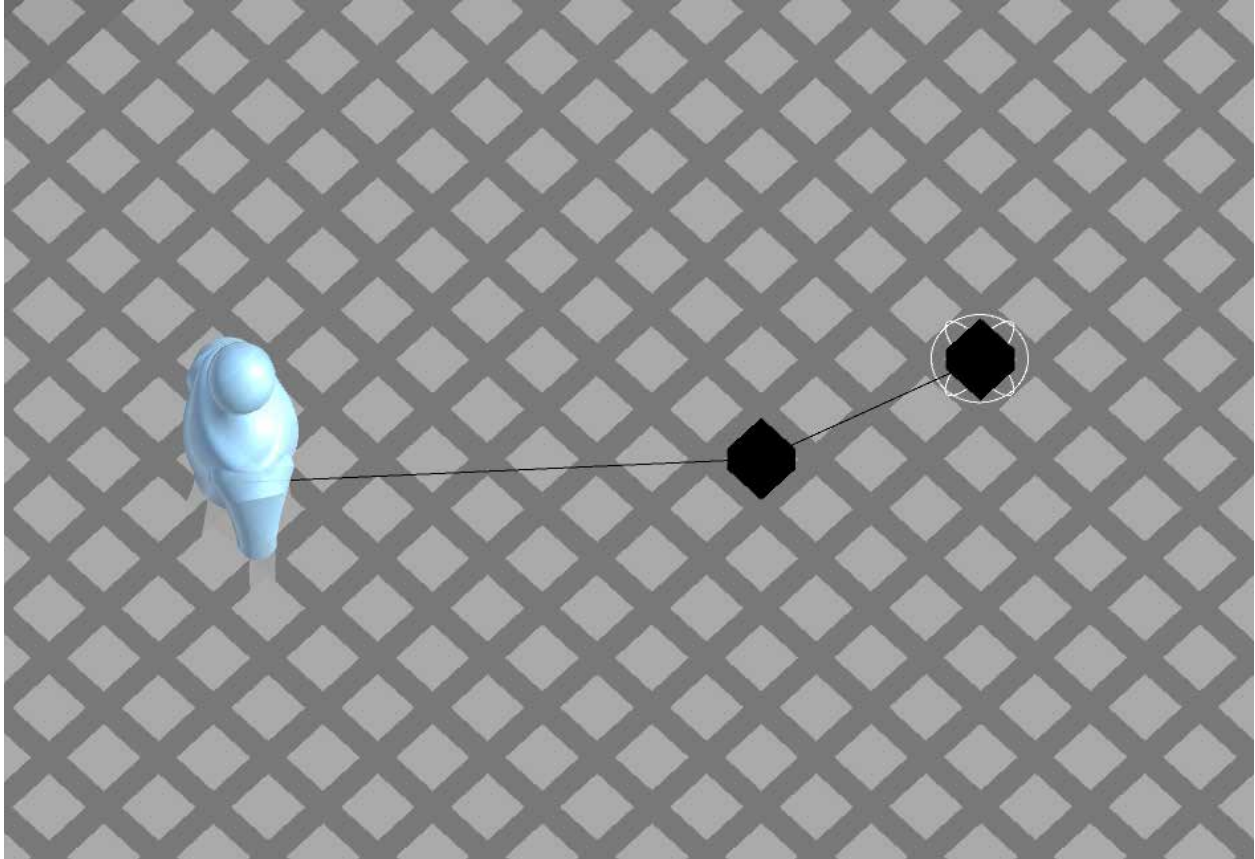


Figure 3.2.h: *Pathfinding creates a path (black line) between nodes along a tiled grid to a target node (white wire sphere).*

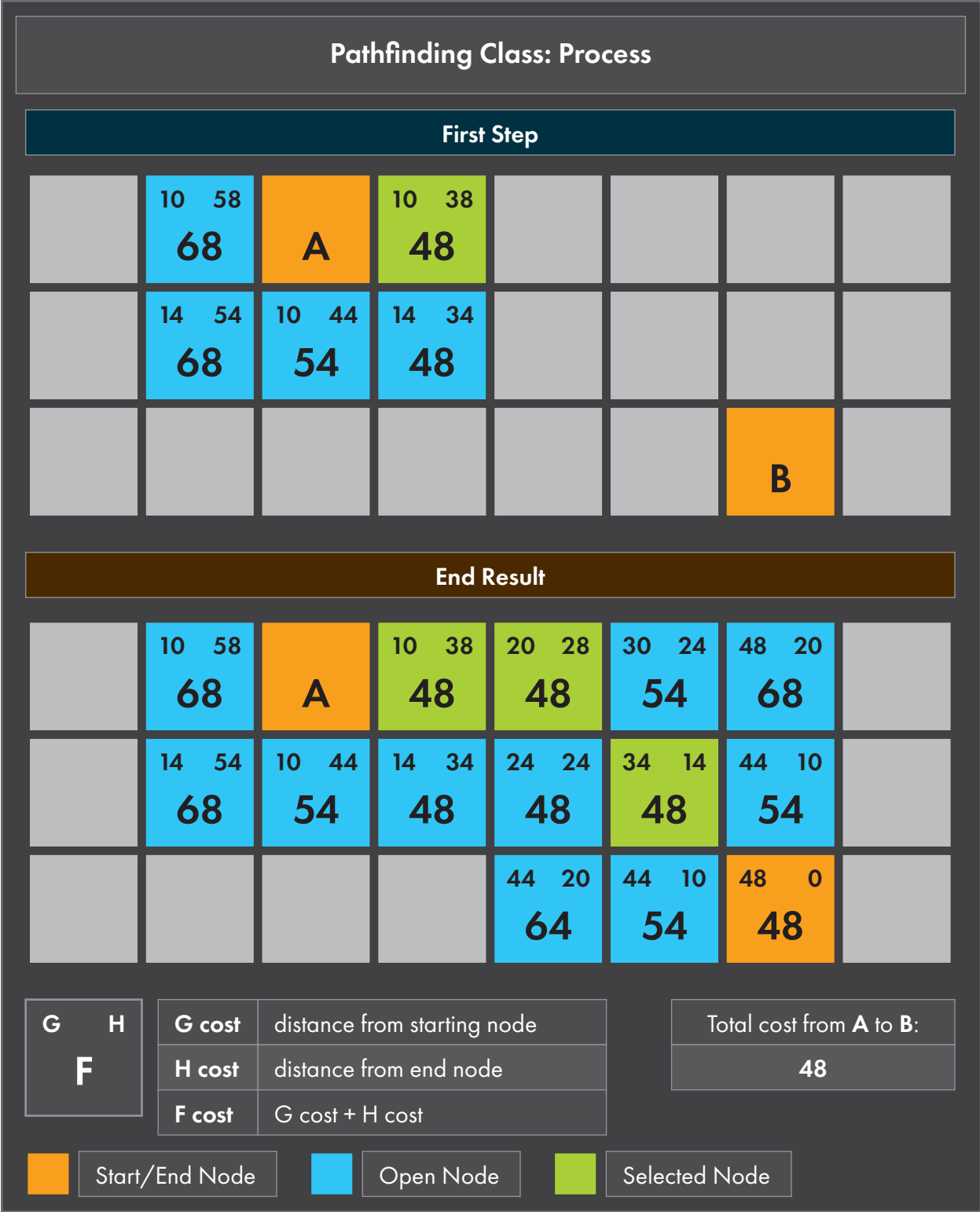


Figure 3.2.i: Lague's A* pathfinding process calculates the cost of neighbouring nodes as a sum of its distance to the start and end nodes. Based on animation by Lague (2014), redrawn by author.










Pathfinding Class: Variables and Methods		
Variables		
	Variable Type Values	Description
	requestManager PathRequestManager -	Reference to the path request manager class
	grid Grid -	Reference to the grid class
Methods		
	Method Type Input Output	Description
	StartFindPath void startPos, targetPos FindPath	Controls when to begin looking for a path, given a start and an end position
	FindPath IEnumerator startPos, targetPos waypoints	The A* search algorithm; reoccurring function that calculates the lowest cost node path between two point
	RetracePath Array of Vector3 startPos, targetPos waypoints	Determines the path vectors between two points
	SimplifyPath Array of Vector3 path waypoints	Resest the order and direction of the path waypoints
	GetDistance int nodeA, nodeB dst	Determines the world distance between two nodes

Figure 3.2.j: Key variables and methods for the pathfinding class.

Chapter 3.3

Airport Architecture Classes

This chapter covers all classes related to airport architecture, which includes *architecture*, *airport objects*, *signage*, *scheduling*, and *itinerary*. Since the thesis is looking at airport terminals, these are the functions specific for the airport processes. If this simulation was used for another building type, then there should be classes related to that building, like healthcare procedures in a hospital. Most of the terminal building elements are created as 3D models for the Unity scene. Therefore, the main responsibility of these classes is to manage object properties, trigger agent processes, and functions for calculating value.

Architecture Class

The architecture class manages value calculations and holds a local class for space properties. In the Unity scene, the architecture script is attached to an empty game object, which allows the user to edit space properties, like program names or areas. Individual architecture elements in the Unity scene, like building components, doors, or walls, do not need to have this script attached to them. Instead, architectural spaces are outlined using basic Unity geometry, like rectangles or boxes, to identify different programmed areas, such as designated security or retail areas (Fig.3.3.a). Each geometry is attached to a given space as a reference, which is stored in the architecture class as a list of spatial areas. This list is then sent to the perception class so that agents can identify where they are in the airport. Whenever an agent walks through a new area, the information about the space is recorded in the agent's memory and updated as the simulation progresses (Fig.3.3.b).

The architecture class is also responsible for scoring agent activities and updating the overall value. When an agent finishes a task associated with a priority, the perception class sends the relevant information to the architectural class to calculate a score. For most processes in this thesis, the score is based on the accumulation of the search time, process time, average visibility, and number of direction changes. These values are normalized using an exponential decay function, which is scaled relative to the simulation rate. For example, Wiredja et al. state that

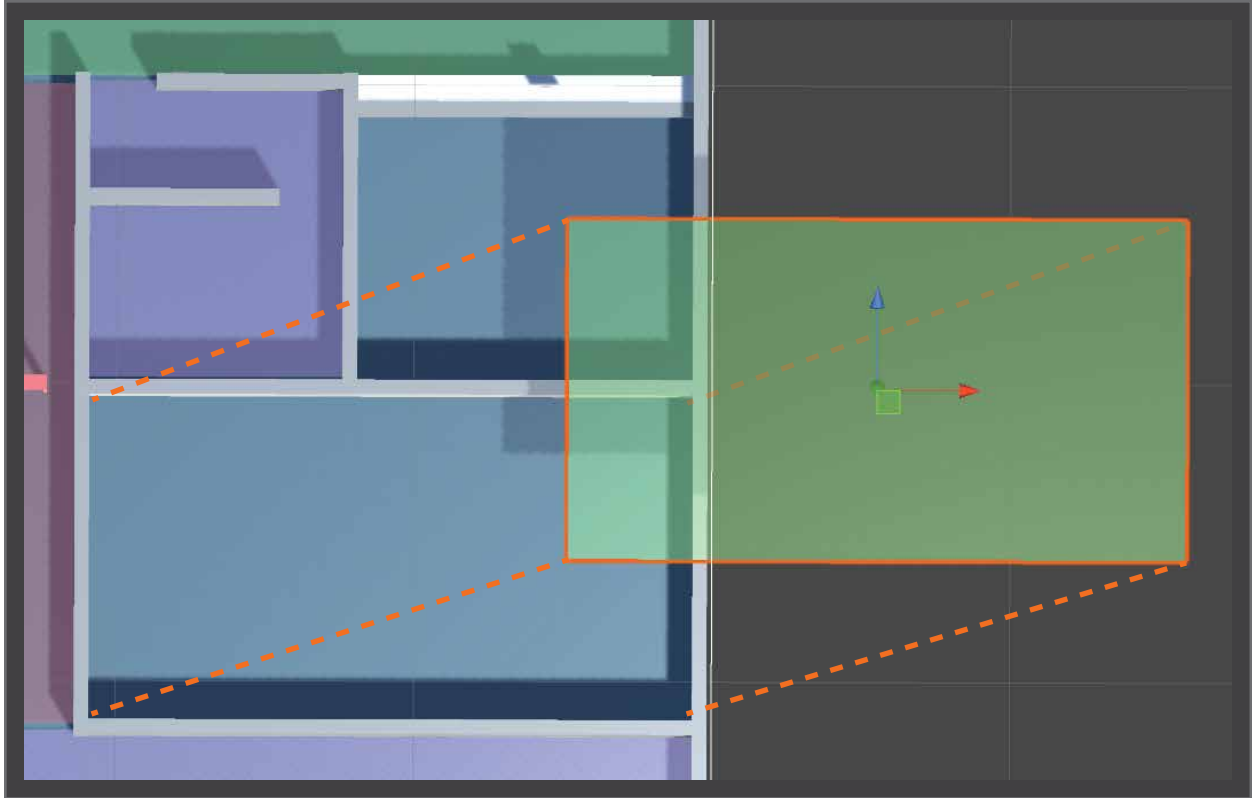


Figure 3.3.a: *Basic geometry used to identify spatial areas.*

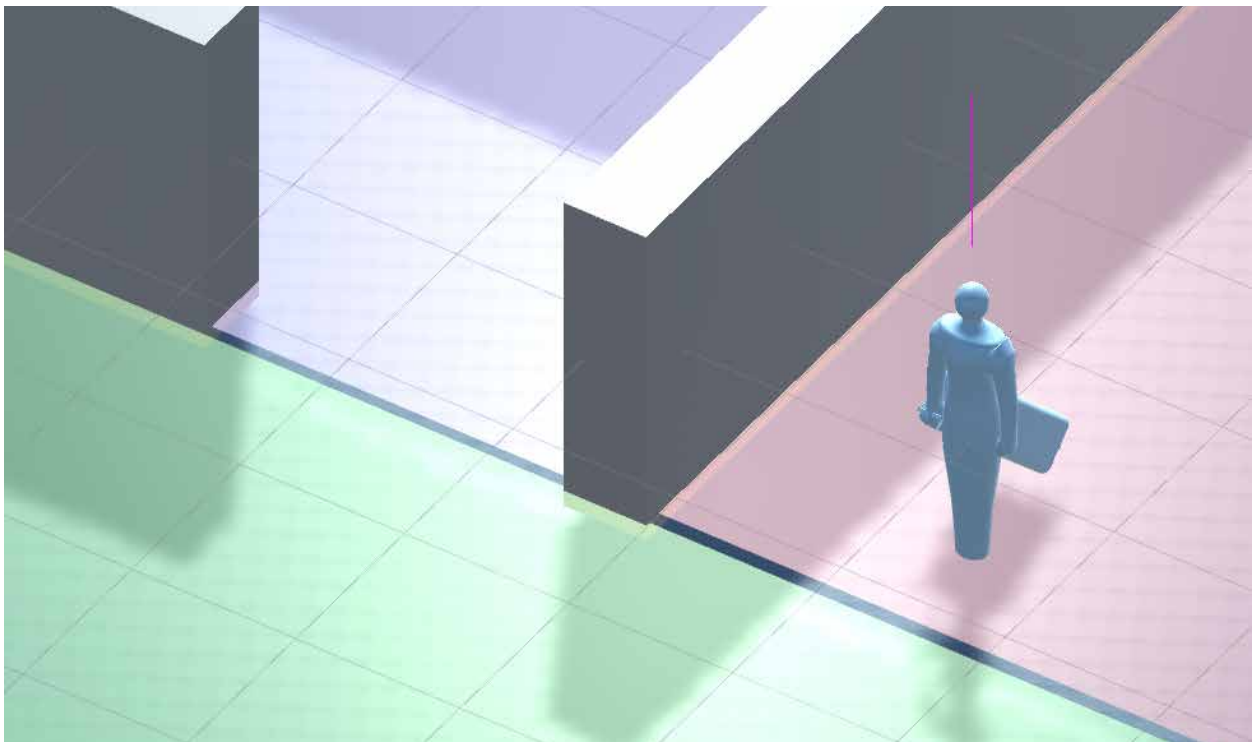


Figure 3.3.b: *Spatial areas are referenced when agents move between them.*

most passengers prefer not to wait in line longer than 15 minutes, on average. ^[1] However, to provide a balance between minimizing simulation run times and modelling passenger behaviour, a scaling factor is applied to increase the rate of decay. In this case, instead of passengers feeling impatient after 15 minutes, this is equivalent to about 15 seconds in the simulation. Likewise, airport processes are also scaled down, so what is normally 10 minutes to check in baggage is only about 10 seconds in the simulation. This makes sure passengers are scoring the same value as if they were waiting in the airport for longer.

Once the scoring is calculated, the values are sent back to the agent's characteristics class. These scores are held by the agent until the end of the simulation. Once the agent reaches their final destination, the agent spawner class, which controls how agents leave the simulation, records the agent's final score and sends this back to the architecture class. All of the agent's scores are then stored in a master list, which keeps updating until all passengers have left the simulation. Once the simulation is complete, the user has the option to print this information from the architecture class to an external text file. The text file then lists each agent's characteristics, priorities, and their respective architectural scores, so they can be analyzed.

In summary, the architecture class is responsible for identifying spatial areas and calculating architectural value. The airport terminal is modelled in the Unity scene and the spaces are highlighted using basic Unity geometry, which is stored in the architecture class as a list of areas. The agent's perception class can reference this list during the simulation so agents can identify where they are in the terminal. Agent priorities are scored in the architecture class using an exponential decay function, based on properties like process time and visibility. The decay functions are scaled down to reduce simulation run time. This means a 10-minute process in real life is equal to a 10-second process in the simulation when calculating a final score. Once agents have completed the simulation, their final scores are sent back to the architecture class in a master list. The list can be exported to a text file for analysis along with agents' respective characteristics.

1. Wiredja, Dedy; Vesna Popovic, and Alethea Blackler. "A Passenger-Centred Model in Assessing Airport Service Performance." *Journal of Modelling in Management* 14, no. 2 (May 10, 2019): 506.

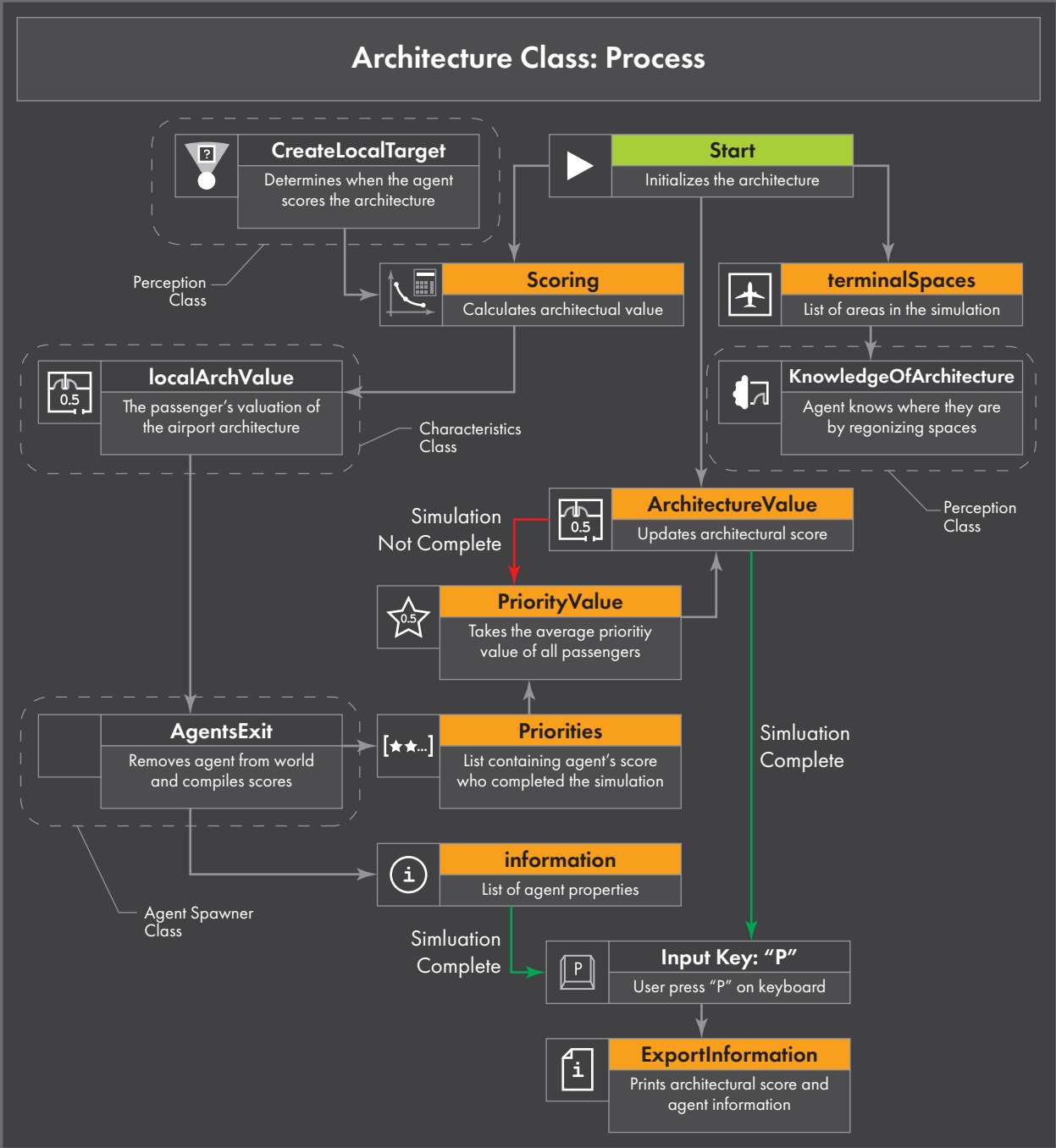


Figure 3.3.c: Process logic for the architecture class.







Architecture Class: Variables and Methods		
Variables		
	Variable Type Values	Description
	Value float 0.0 - ∞	The overall score of the architecture
	Priorities List of floats 0.0 - ∞	List of the overall priority scores of every agent who completed the simulation
	information List of strings -	Stores each agent's properties and characteristics as text
	spacesLayer LayerMask -	Physical object that represents the extents of a spatial area
	terminalSpaces List of Spaces spaceName, area	List of spaces or areas in the airport terminal

Figure 3.3.d: Key variables for the architecture class, page 1.







Methods		
	Method Type Input Output	Description
	ArchitectureValue void Value Value	Updates the current architectural score as the simulation is running
	PriorityValue float Priorities _valueAvg	Calculates the architectural score as the average of all the agent's priorities
	Scoring float time, visibility, dirChanges score	Calculates the score for a completed process
	ExportInformation void information -	Prints the current architectural score and corresponding agent priorities to an external text file
	Space class spaceName, area	Local class defines architectural space based on a given name and physical object representing the area

Figure 3.3.e: Key methods for the architecture class, page 2.

Airport Objects Class

The airport objects class controls how agents interact with objects during a process. As mentioned in the perception class, agents navigate in the environment by replacing the object's location with a local target. However, most objects, like service counters, normally involve standing in front of the object to properly interact with it. Therefore, the main purpose of the airport object class is to provide *interaction nodes*, so agents can approach a given object from a realistic distance. Additionally, objects also have *exit nodes* and *queue spots* to help agents leave the object or navigate through queues, respectively. Fundamentally, the airport object class is necessary for all objects that the agent interacts with (except for signage).

Interaction nodes are represented in the Unity scene as an empty game object. The user of the simulation manually places these nodes in front of objects, depending on the process. For example, security metal detectors have an interaction node in front of the detector and an exit node on the other side, which simulates agents walking through the detector during the screening process (Fig.3.3.f). In the airport objects class, these nodes are assigned to a vector position variable. When an agent perceives a given object, the airport objects class sends the vector position of the interaction node to the agent's perception class, so they can walk towards it. Once the agent completes the corresponding process, the airport objects class then provides the vector location for the exit node, so the agent can leave the object.

The airport objects class also manages how agents interact with queue lines. Queues are created as a collection of waypoints along a given line. This requires the user of the simulation to manually assign a physical line for the corresponding airport objects class. The class uses this physical line to determine the number of available *queue spots* based on a given spacing (Fig.3.3.g). Like interaction nodes, queue spots are used for agent navigation. When an agent is in a queuing state, the airport objects class sends these queue spots to the perception class for the agent to follow.

In summary, the airport objects class makes sure agents interact with objects from a realistic distance. This involves placing interaction nodes and exit nodes in front of objects, for agents to approach and leave them, respectively. The airport objects class also manages queues by providing queue spots along a predefined line. When an agent perceives an object, the airport objects class sends the location of the interaction nodes, exit nodes, or queue spots to the agent's perception class, so the agent can navigate accordingly.

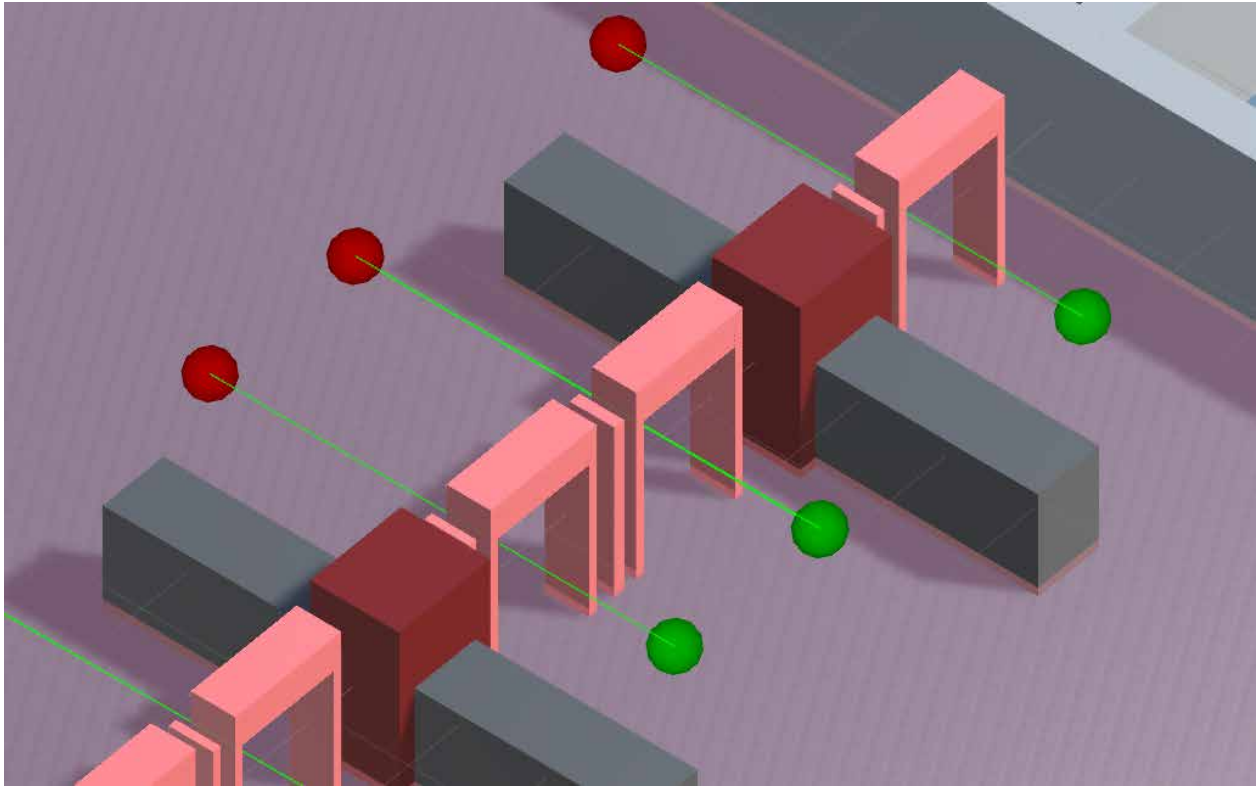


Figure 3.3.f: *Illustration of interaction nodes (green) and exit nodes (red) in security screening.*

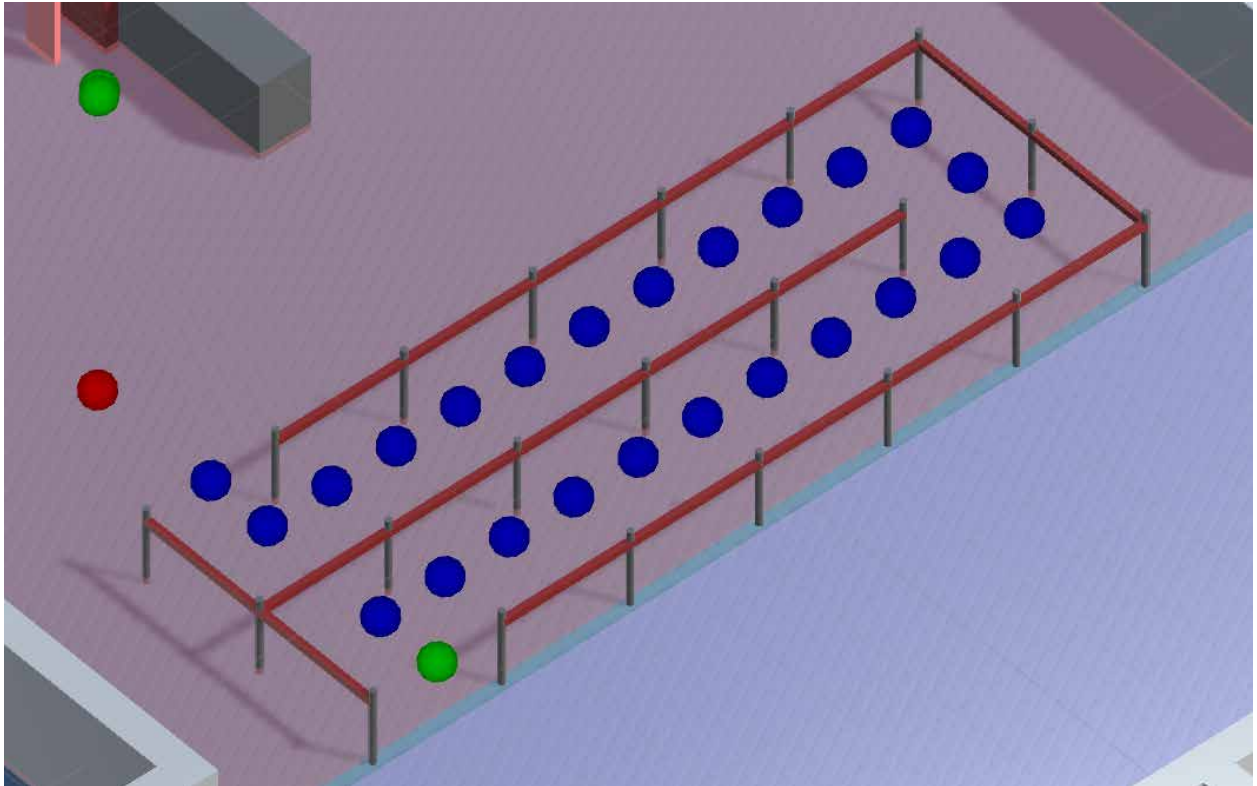


Figure 3.3.g: *Illustration of queue spots (blue) in a queuing line.*

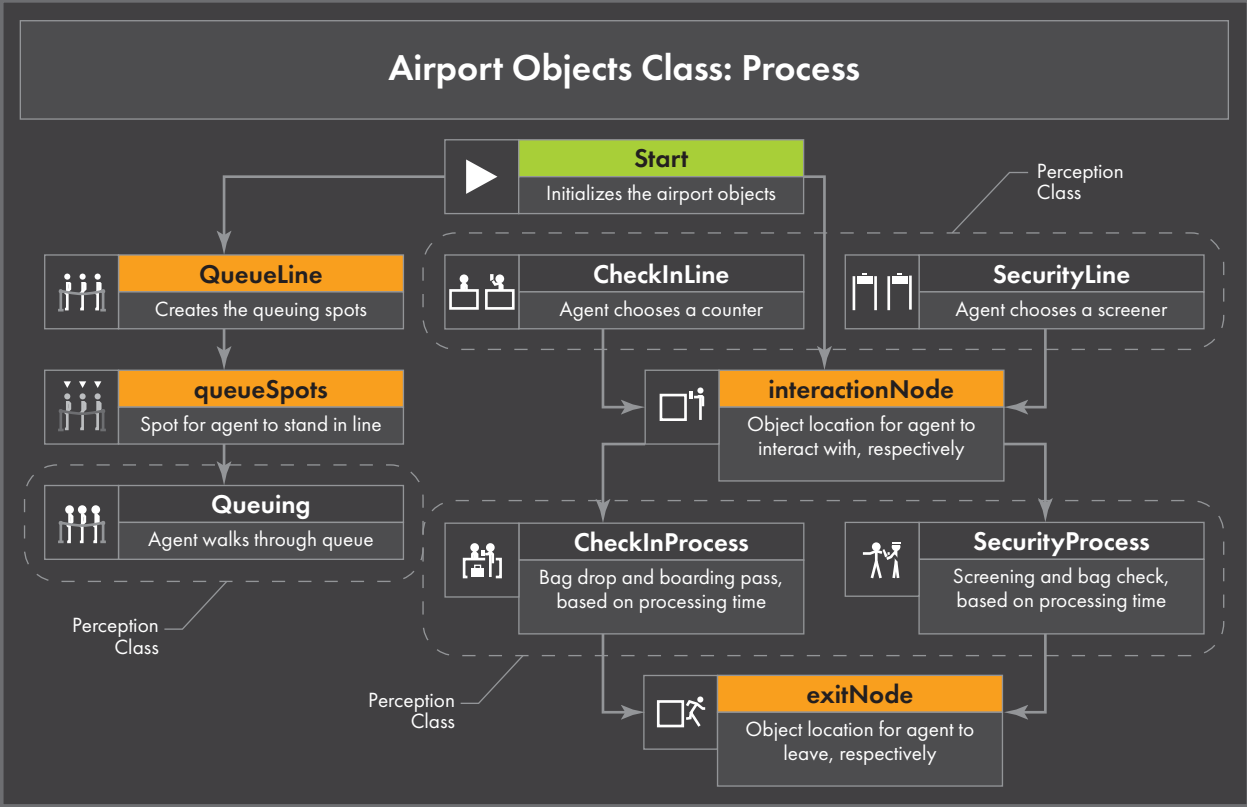


Figure 3.3.h: Process logic for the airport objects class.









Airport Objects Class: Variables and Methods		
Variables		
	Variable Type Values	Description
	interactionNode GameObject -	Reference to the location an agent can interact with the object
	exitNode GameObject -	Reference to the location an agent leaves the object
	lineNode List of Transforms -	List of queue line locations to be referenced in the scene environment
	queueSpots List of Vector3 -	List of queuing spots in a line for the agents to follow
	spacing float 1.0	Distance each agent stands from each other (m)
Methods		
	Method Type Input Output	Description
	QueueLine void lineNode queueSpots	Creates the queue spot locations based on a given line geometry

Figure 3.3.i: Key variables and methods for the airport objects class, page 1.

Signage Class

The signage class is responsible for information written on signs and stores direction vectors for agent navigation. In this thesis's simulation, agents do not read the graphic information shown on the physical sign; what is visible in the scene environment is only for display. Instead, sign objects use the signage class to communicate their information to the agent's perception class.

A signage class is attached to all objects that agents can use for wayfinding. Wayfinding objects are designed to force agents to walk a certain direction if they need to choose between more than one route. Before the simulation starts, the signage class requires the user to provide a *viewpoint*, *info names*, and corresponding *direction nodes* for every item on the sign (Fig.3.3.j). A viewpoint acts as a gathering point in front of the sign. The info name tells the agent what target the sign is referring to, and the direction node is used to calculate a direction vector for the agent's navigation. This vector is only calculated by the signage class when the agent interacts with the sign.

When an agent sees a sign during the simulation, the perception class calls a function to read the sign. The first step is to move the agent to the viewpoint. Since agents may approach the sign from different directions, the viewpoint makes sure the agent is facing the sign correctly before reading it. As described in the perception class, the act of reading involves checking if there is an item name in the signage class that matches the agent's primary target. This is done by referencing the agent's itinerary class for a name like "Gate B". If the sign also says, "Gate B", then the corresponding direction vector for "Gate B" is sent to the agent class for navigation. If the agent is not looking for their gate, then the perception class checks if the names on the sign matches the corresponding task in the itinerary class, like "Security Screening". If none of the items on the sign match the agent's current task, then the agent continues wandering.

In summary, the signage class stores the information written on wayfinding signs and determines the corresponding direction vector for agent navigation. Agents in this simulation cannot read what is visually displayed on wayfinding. Instead, the signage class stores the sign's information, so that the agent's perception class can read it. This information must be assigned by the user before the simulation starts. When an agent sees a sign, they first walk to a viewpoint, so they are in a position to read it properly. If the item names stored in the signage class match the agent's target or task, as defined in the itinerary class, then the corresponding direction vector the sign is pointing to is provided to the agent for navigation.

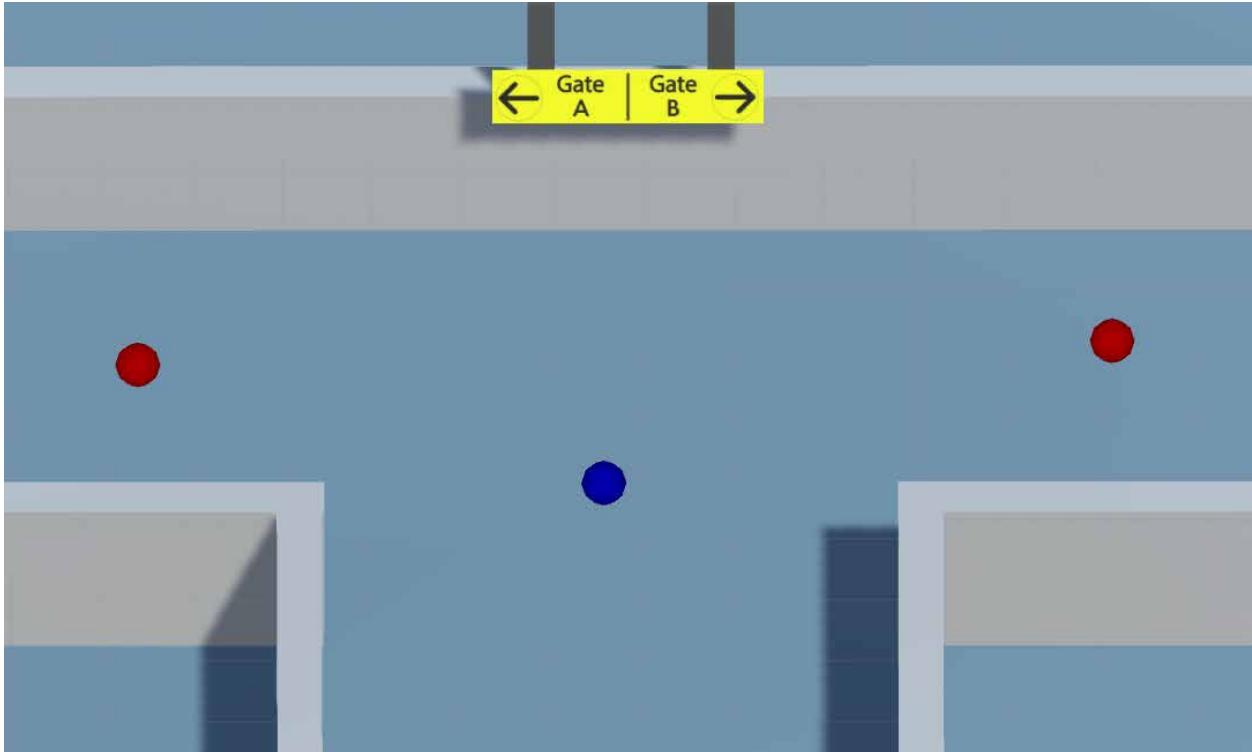


Figure 3.3.j: A wayfinding sign illustrated with a viewpoint (blue) and two direction nodes (red) for Gate A and Gate B.

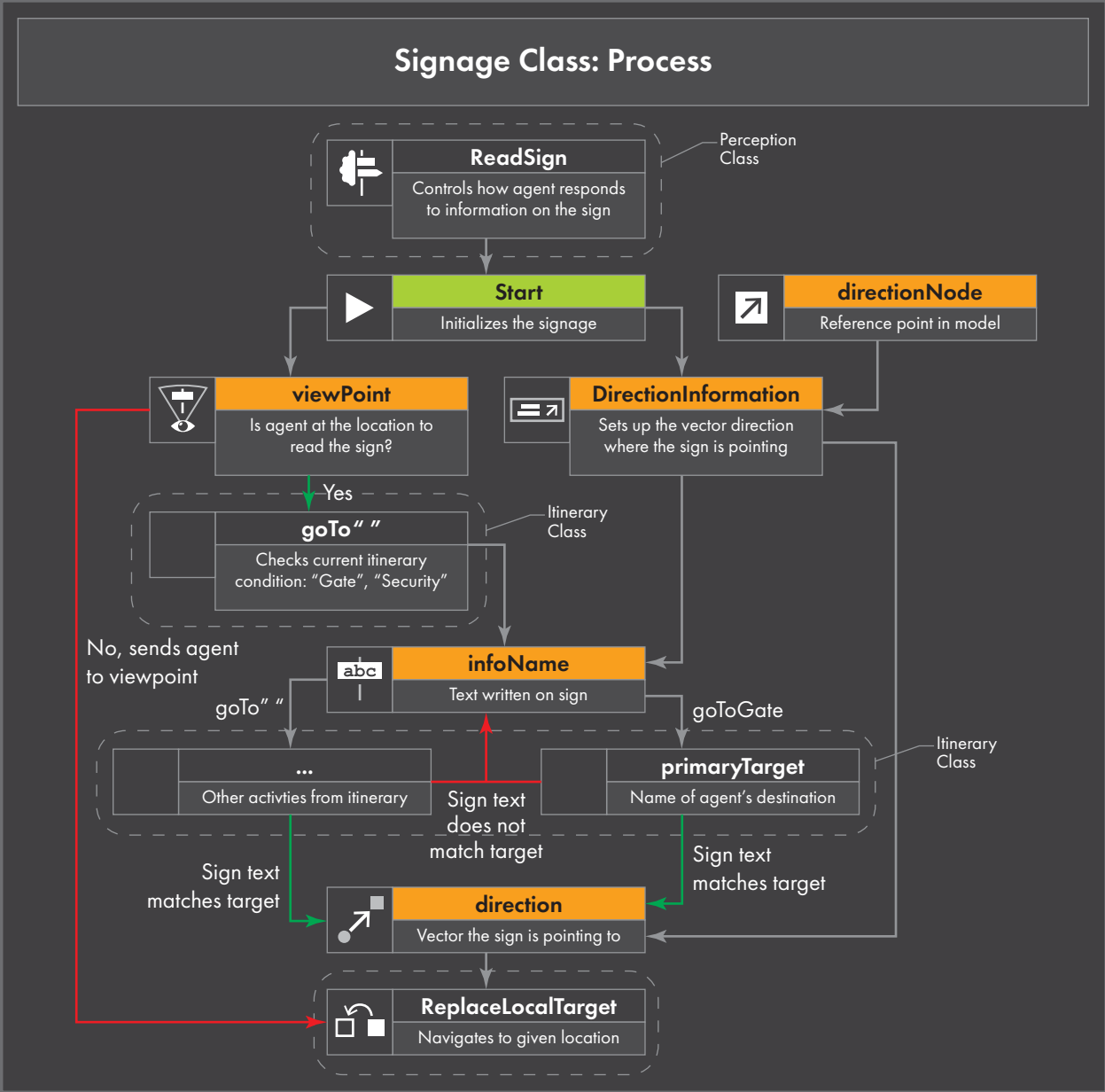


Figure 3.3.k: Process logic for the signage class.










Signage Class: Variables and Methods		
Variables		
	Variable Type Values	Description
	Sign Array of DirectoryInfo -	Array of possible directions displayed on a sign
	viewPoint GameObject -	Is a physical point from where the agent can read the sign
	infoName string -	The name displayed on the sign as text
	directionNode GameObject -	The object representing the location the agent navigates to
	direction Vector3 -	Holds the vector between the view point and possible direction of travel
Methods		
	Method Type Input Output	Description
	DirectionInformation void directionNode direction	Defines the possible directions agents can go from a sign
	DirectionInfo class infoName, directionNode, direction	Local class holds the name written on the sign and the direction the sign is pointing

Figure 3.3.1: Key variables and methods for the signage class, page 1.

Scheduling Class

The scheduling class manages the location where, and frequency of, agents entering and exiting the simulation. The user of the simulation assigns the scheduling class to an empty game object, which requires inputting values into a corresponding property window (Fig.3.3.m). The scheduling class determines the total number of passengers (pax) expected to run through the simulation, either as a statistical distribution or random variables.

The scheduling class controls two types of objects, *arrival points* and *departure points*. Arrival points spawn (generate) agents into the simulation and is the location agents begin their journey. Departure points remove agents from the simulation and are considered the agent's *primary target*. Arrival and departure points can be assigned to any type of object in the environment, but they are commonly represented as doorways. There can also be more than one arrival or departure point in the simulation, but the scheduling class must assign each agent their own arrival and departure point before the simulation begins.

Arrival points are defined by a name, a location, an arrival window, and an agent model. The name and location of the arrival point is used by the agent spawner class to identify which arrival point agents are generated at. The arrival window is a range of time agents can randomly enter the simulation, which is defined by a minimum and maximum time. For example, if the minimum and maximum times are 1 sec. and 5 sec., respectively, then an agent will randomly spawn at the arrival point as quickly as once per second or as slowly as once every five seconds. Finally, the agent model is a reference to the physical agent that will be generate. It is possible for this simulation to have different kinds of agents, but there is currently only one type of agent model.

Additionally, arrival points are typically located at the front of the terminal building. However, if an arrival point is past a check-in area or a security line, then there are also options in the scheduling class to provide agents with the required clearances for that area.

Departure points are similar to arrival points, which are defined by a name, a location, a departure window, and a departure time. The name and location are identified by the agent spawner class to remove agents from a given location. The departure window is a random range of time when a location is accessible, like a gate that is ready for boarding, defined by a minimum and maximum value. A departure time is randomly assigned for each departure point based on this window. If the agent reaches the departure point before the departure time, then

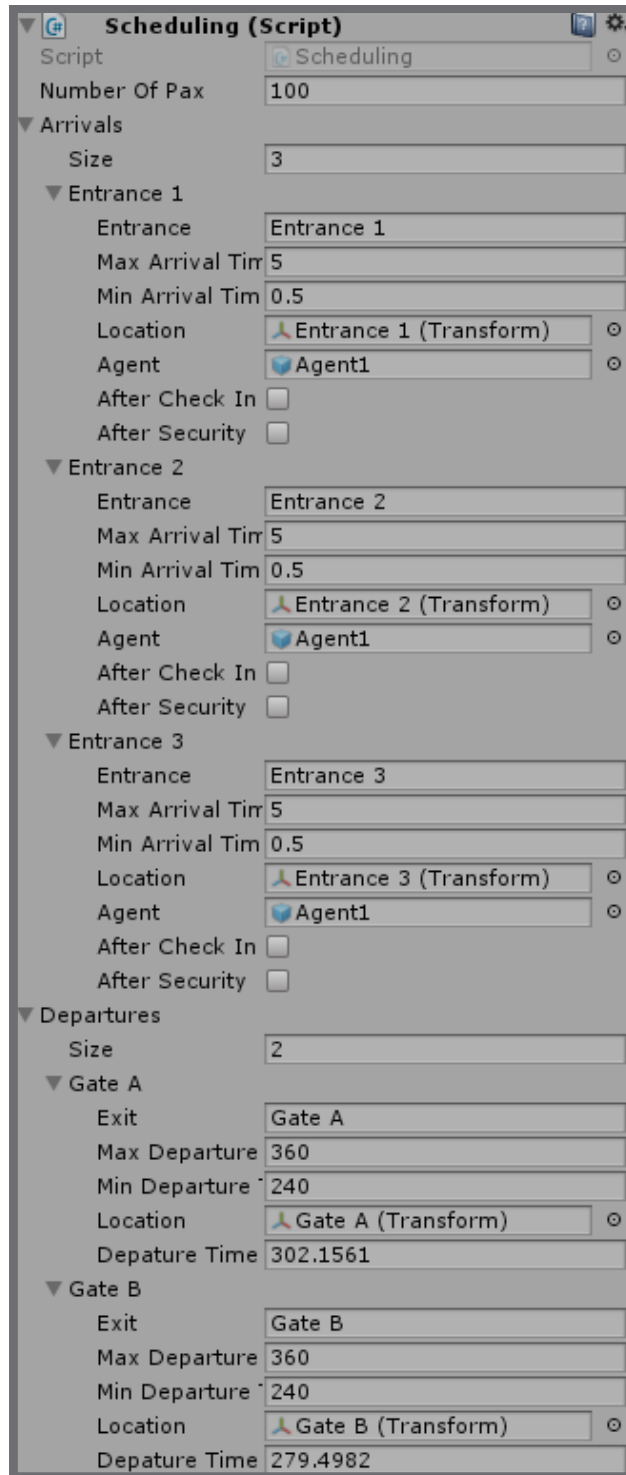






Figure 3.3.m: An example of a simulation schedule assigned in the Unity inspector properties, using the scheduling script, with 3 arrival points and 2 departure points.

they cannot exit and must wait. Only after the simulation time passes the departure time are agents able to exit through their assigned target.

In summary, the scheduling class is responsible for the locations where agents enter and exit the simulation, and the total number of passengers. The scheduling class defines arrival points and departure points, which are locations agent enter and exit the simulation from, respectively. Both arrival and departure points are assigned a time range, which determines a random frequency of agents entering or exiting. There can be more than one arrival or departure point, as long as the scheduling class assigns agents to these locations before the simulation starts. If an agent reaches their departure point, which is considered their primary target, then the agent must wait to exit until the gate's assigned departure time.

Scheduling Class: Variables and Methods

Variables

	Variable	Description
	Type	
	Values	
	numberOfPax int 100, 0 - ∞	Total number of agents expected in the simulation, default is 100 pax (passengers)
	arrivals Array of ArrivalPoints -	List of locations and conditions for agents to enter the simulation
	departures Array of DeparturePoints -	List of locations and conditions for agents to exit the simulation

Local Class: ArrivalPoint

	Variable	Description
	Type	
	Values	
	entrance string "Entrance 1, 2, ..."	the name of the location passengers enter from as text
	MaxArrivalTime float 0.0 - ∞	Maximum time range between passengers entering at the given location
	MinArrivalTime float 0.0 - ∞	Minimum time range between passengers entering at the given location
	location Transform (x, y, z)	Reference to the physical object (door, portal) of the arrival point
	agent GameObject Agent1	The type of agent model that is spawned at the given location

Figure 3.3.n: Key variables for the scheduling class, page 1.

	afterCheckIn bool true, false	Condition if the arrival point is after the check-in area, gives agents relevant clearances at start
	afterSecurity bool true, false	Condition if the arrival point is after the security area, gives agents relevant clearances at start
Local Class: DeparturePoint		
	Variable Type Values	Description
	exit string "Gate A, B, ..."	The name of the location passengers exit from as text
	MaxDepartureTime float 0.0 - ∞	Maximum time range for departure time at the given location
	MinDepartureTime float 0.0 - ∞	Minimum time range for departure time at the given location
	location Transform (x, y, z)	Reference to the physical object (door, portal) of the departure point
	departureTime float 0.0 - ∞	The time when the exit opens and agents can leave the simulation

Figure 3.3.0: Key variables for the scheduling class, page 2.

Itinerary Class

The itinerary class manages a list of tasks in the airport for an agent to complete, like a checklist. This acts as a person's memory, who is keeping track of things they need to do and places they need to go. In general, the itinerary class informs the perception class when to perform certain actions. These actions may include getting checked in for a flight, memorizing a departure gate, or double-checking the time.

The main function in the itinerary class is called the *Checklist*, which repeatedly iterates through all the agent's tasks to see if they are completed. The checklist primarily focuses on the departure process, which includes, check-in, security screening, and gate boarding. Agents must complete each process in order before they can move into the next area.

For example, the first task for the agent is to get checked in. If the agent is not checked in at the start of the simulation, the itinerary class tells the perception class to look for a check-in counter. Once the agent finds the check-in counter, the itinerary class memorizes the location of their *chosen line*, or selected counter, which is used as a reference during the check-in process. When the check-in process is finished, the perception class sets this as complete in the itinerary class, which indicates the agent has checked in. After the agent is checked in, the checklist moves to the next task in the list, which in this case would be security screening.

The same process repeats for each task the agent must complete. This involves the itinerary class informing the perception class of what objects to look for, and the perception class letting the itinerary know when the corresponding process is done. In addition to the departure process, the itinerary class also manages conditions for signage, queuing, and non-processing domains. If the agent is early to the gate, the itinerary tells the perception class to wait at the gate until it becomes the departure time. This may result in the agent going to the retail area, using the washrooms, or sitting in the waiting area. Otherwise, once it becomes the departure time, the itinerary class informs the perception class to exit through the gate.

In summary, the itinerary class functions like a checklist, which informs the agent's perception class of what the agent should be doing. The primary checklist is the departure process, which requires the agent to complete each task before moving into the next area. If a task is incomplete, the itinerary class informs the perception class of what to look for. Otherwise, the perception class will mark off each task as the agent completes it. The checklist also manages conditions for non-processing domains, which may occur before the agent's departure time.

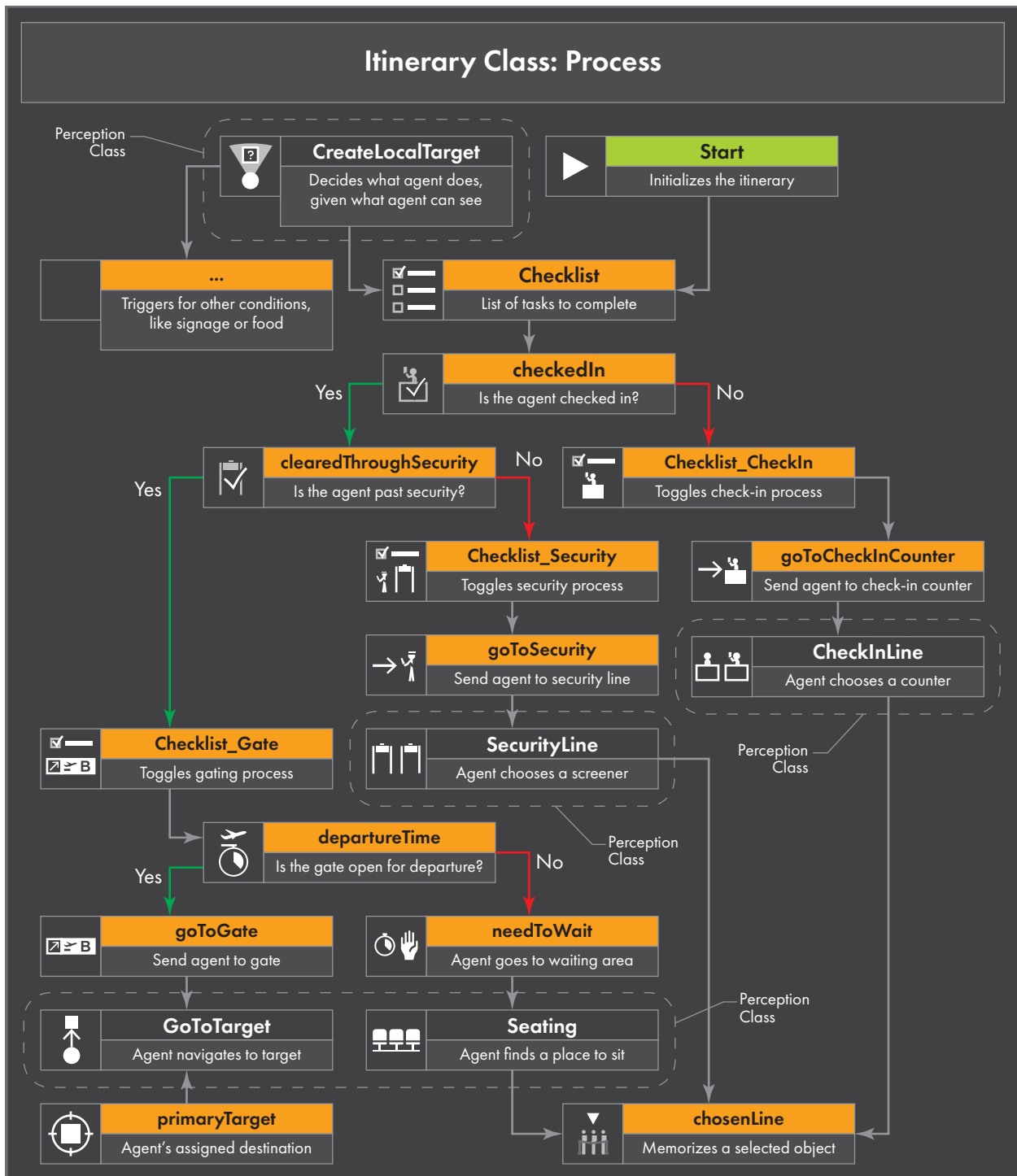


Figure 3.3.p: Process logic for the itinerary class.

Itinerary Class: Variables and Methods

Variables

<input checked="" type="checkbox"/> Variable		Description
<input type="checkbox"/> Type		
<input type="checkbox"/> Values		
	primaryTarget string "Gate B"	Name of the object representing the agent's end destination, default is Gate B
	departureTime float 0.0 - ∞	The agent's flight departure time
	boardingPass bool true, false	Checks if the agent has a boarding pass; agent has no boarding pass by default
	goToCheckInCounter bool true, false	Triggers for the agent to go to the check-in counter
	queueForCheckIn bool true, false	Trigger for check-in queuing
	checkedIn bool true, false	Checks if the agent has been checked into their flight; agent has not checked in by default
	goToSecurity bool true, false	Triggers for the agent to go to security
	queueForSecurity bool true, false	Triggers for security queuing
	clearedThroughSecurity bool true, false	Checks if the agent has been cleared through security; agent has not clear security by default

Figure 3.3.q: Key variables for the itinerary class, page 1.

	goToGate bool true, false	Triggers for the agent to go to their gate
	inQueue bool true, false	Checks if the agent is in a queue
	hasQueued bool true, false	Checks if the agent has finished queuing
	readingSign bool true, false	Checks if the agent is reading a sign
	hasReadSign bool true, false	Checks if the agent has already read a sign
	targetFound bool true, false	Checks if the agent has found their primary target
	needToWait bool true, false	Checks if the agent needs to wait at the gate
	gettingFood bool true, false	Triggers if agent is getting food
	gotFood bool true, false	Checks if the agent has already gotten food
	perception Perception -	Reference to the agent's perception class
	charcter Characteristics -	Reference to the agent's characteristics class
	chosenLine Transform -	Reference for the agent's choice of queue line or other airport objects

Figure 3.3.r: Key variables for the itinerary class, page 2.













Methods		
<input checked="" type="checkbox"/> 	Method	Description
<input type="checkbox"/> 	Type	
<input type="checkbox"/> 	Input Output	
<input checked="" type="checkbox"/> 	Checklist	Agent's list of assigned tasks, which calls other functions based on activity
<input type="checkbox"/> 	void	
<input type="checkbox"/> 	- Checklist_ " "	
<input checked="" type="checkbox"/> 	Checklist_CheckIn	Is the checklist for the airport check-in process
	void	
	boardingPass goToCheckInCounter	
<input checked="" type="checkbox"/> 	Checklist_Security	Is the checklist for the airport security process
	void	
	checkedIn goToSecurity	
<input checked="" type="checkbox"/> 	Checklist_Gate	Is the checklist for the gating process
	void	
	clearedThroughSecurity goToGate	

Figure 3.3.s: Key methods for the itinerary class, page 3.

Chapter 3.4

Simulation Utility Classes

Simulation utility classes deal with user control and help to optimize the code while the simulation is running. This includes *agent spawner*, *path request manager*, *heap*, and *field of view editor*. The agent spawner class is created in this thesis to manage agent generation. Path request manager and heap are based on Lague's *A* Pathfinding* (2016) tutorial ^[1] and the field of view editor is based on Lague's *Field of View Visualizations* (2015) tutorial. ^[2] Lague gives a better description of how these classes work than this thesis can explain. Therefore, this chapter gives a brief summary of how they are used.

Agent Spawner Class

The agent spawner class controls how agents are added and removed from the simulation. A spawner is another name for a generator or creator, which acts like a control panel for the user. It can select different types of agent navigation and modify how fast the simulation is run. Most options in this class are controlled by a keyboard hotkey, which the user can press at any point during the simulation. The spawner will then automatically assign agent characteristics and properties.

For this thesis, the user can select from three types of starting conditions. The first starting condition spawns (generates) agents with direct navigation, which is initiated by pressing the "D" key. This makes agents navigate straight to their target using the shortest A* path, as described in the perception class. The second starting condition spawns agents with perceptive navigation, which is initiated by the "E" key. The third condition starts agents walking if there are already agent models placed in the environment, which is initiated by the "W" key. The first two conditions assume there are no agents in the world from the beginning, and spawn new agents at scheduled arrival points. Whereas the third condition occurs if agents are already placed in environment and the user wants to start them moving from there. This is commonly

1. Lague, Sebastian. "Pathfinding/Episode 9 - smooth path 02/Assets/Scripts". GitHub. December 30, 2016. <https://github.com/SebLague/Pathfinding/blob/master/Episode%209%20-%20smooth%20path%2002/Assets/Scripts>.

2. Lague, Sebastian. "Field-of-View/Episode 02/Editor". GitHub. December 28, 2015. <https://github.com/SebLague/Field-of-View/tree/master/Episode%2002/Editor>.

used for validation tests based on the IMO standard for evacuation simulation, which requires a predefined number of agents in a given space, like 20 passengers starting in one room.

While the simulation is running, the user may press the “Q” key to remove all the agents from the world. However, this does not reset the simulation time, like the arrival and departure times. Instead, resetting the time requires restarting the simulation.

The agent spawner class can also control the simulation time scale, or how quickly the discrete simulation is updated. By default, the time scale is set to one, or real-time. Although the user may press the plus (+) or minus (-) keys to increase or decrease the time scale by increments of one, respectively. This will scale all properties of the simulation, including agent walking speeds, processing times, and scheduled departure times. This is useful for creating time-lapses or decreasing the simulation time of long airport processes. However, running too fast of a time scale causes the thesis’s agents to encounter pathfinding issues, like missing waypoints.

The agent spawner class continually works in the background of the simulation. After the user makes their selection, the spawner repeatedly adds agents at arrival points, based on the conditions defined in the scheduling class. When an agent is spawned, the class randomly assigns them a primary target and random characteristics. As mentioned in the scheduling class, a primary target is assigned to each agent from the available departure points. Likewise, the spawner calls the characteristic class to generate new agent properties based on distributions provided by the IMO standard for evacuation simulations. Additionally, the agent spawner class randomly assigns airport priorities within the thesis’s selected six airport domains. New agents have an equal probability of receiving any given property, which represents a random population. Although, the agent spawner class can also incorporate statistical data to determine how the population is generated.

When an agent reaches their primary target, the agent spawner class is responsible for recording their corresponding information and removing them from the simulation, like an exit checklist. Before the agent is removed, the information stored in their characteristics class is compiled together into one string of text. This string of text contains the agent’s name, age, gender, primary target, simulation time, overall architectural score, all priorities (in alphabetical order), their corresponding importance, and weighting. Once this is recorded, the string is sent to the architectural class to be stored in a master list. The process of removing the agent itself requires making sure the physical agent model, local targets, and components are deleted from the environment.

In summary, the agent spawner class is responsible for adding and removing agents from the simulation, which is controlled by the user using keyboard hotkeys while the simulation is running. The user can decide to spawn (generate) agents with or without perception, and can control the simulation time scale, either a faster or slower rate, within reason. The agent spawner class works in the background adding and removing agents at assigned locations based on the information defined in the scheduling class. This includes assigning agent characteristics and priorities based on a random population distribution. Once an agent reaches their target, the spawner deletes the agent model and compiles the agent's properties into a string of text, which is sent to a master list in the architecture class.

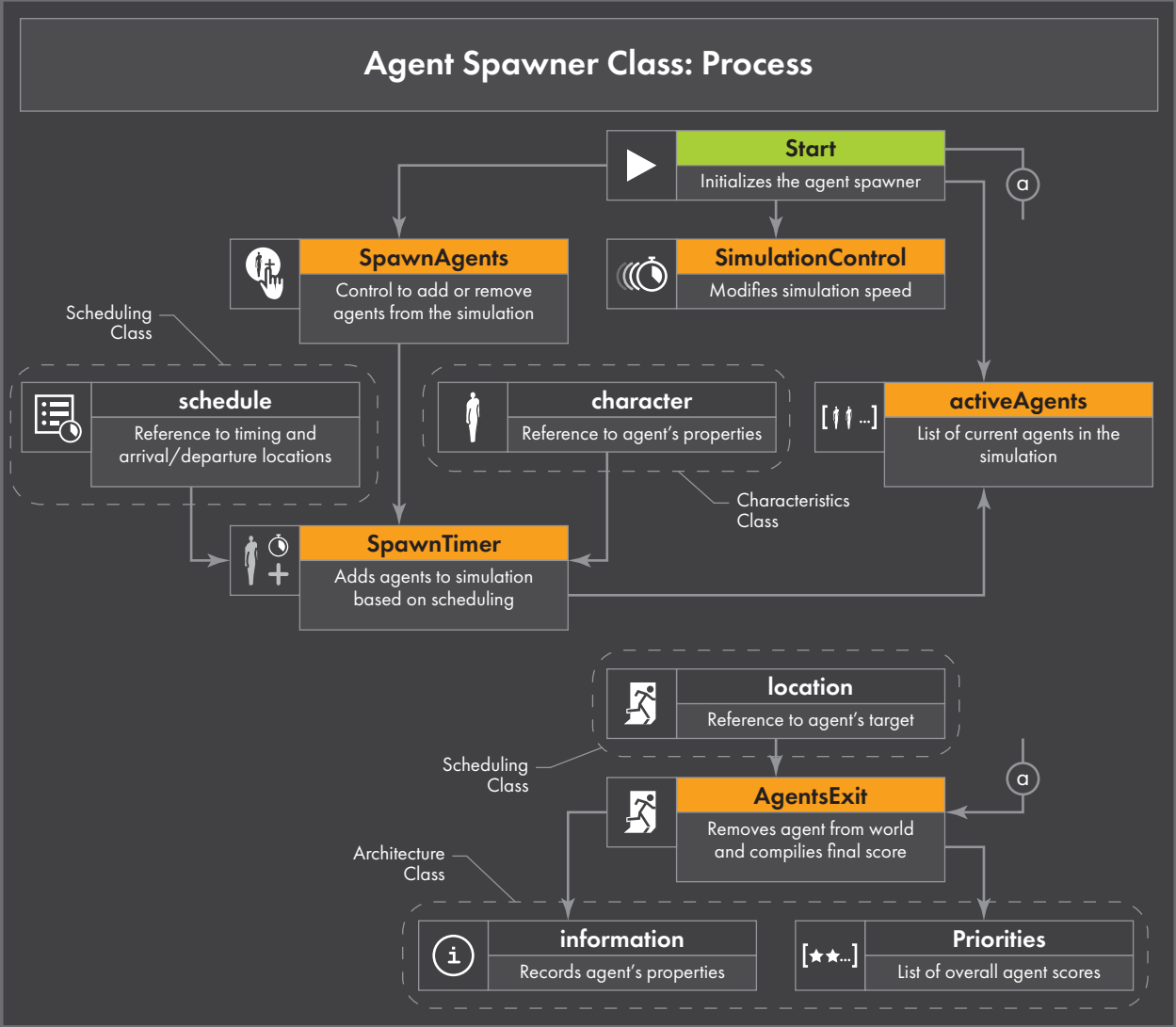


Figure 3.4.a: Process logic for the agent spawner class.

Agent Spawner Class: Variables and Methods

Variables

	Variable Type Values	Description
	activeAgents List of GameObjects -	List of agents currently in the world
	perceptive bool true, false	Determines if agents navigates with perception or direct path
	schedule Scheduling -	Reference to the scheduling class
	arch Architecture -	Reference to the architecture class
	character Characteristics -	Reference to agents' characteristics classes
	currentSpaces GameObject -	Reference to the architectural spaces currently in the world
	spawnedAgentCount int 0 - ∞	Keeps track of how many agent have entered the simulation
	agentCapacity int 30	Maximum number of agents allowed in the simulation at any given time (default 30)
	currentActiveAgents int true, false	Records the number of agents currently running in the simulation

Figure 3.4.b: Key variables for the agent spawner class, page 1.








	startPerceptiveAgentsKey KeyCode W	Keyboard shortcut to start agents moving with perception who are already in the world
	spawnPerceptiveAgentsKey KeyCode E	Keyboard shortcut to begin adding agents to the world with perception
	spawnDirectAgentsKey KeyCode D	Keyboard shortcut to begin adding agents to the world with direction routing
	removeAgentsKey KeyCode Q	Keyboard shortcut to remove all agents currently in the world
	resetTimeScaleKey KeyCode 1	Keyboard shortcut to reset the time scale to one (default speed)
	fasterTimeKey KeyCode +	Keyboard shortcut to increase the time scale by one
	slowerTimeKey KeyCode -	Keyboard shortcut to decrease the time scale by one

Figure 3.4.c: *Key variables for the agent spawner class, page 2.*







Methods		
	Method Type Input Output	Description
	SimulationControl void "1", "+", "-" Time	Controls how fast the simulation is running
	SpawnAgents void "W", "E", "D", "Q" SpawnTimer	Allow user to control adding or removing agents from the simulation
	AgentsExit void activeAgents activeAgents	Removes agents from the simulation and compiles their corresponding architectural score
	SpawnTimer IEnumerator schedule agents	Reoccurring function controlling how agents enter the simulation; assigns random characteristics and targets based on scheduling
	AgentTime float _startTime, endTime finalTime	Determines how much simulation time the agent spent doing something given a start and end time in seconds

Figure 3.4.d: Key methods for the agent spawner class, page 3.

Path Request Manager Class

The path request manager class regulates how often agents use the A* algorithm, when there are a lot of agents trying to navigate at the same time.

This class works like a lending library, in which the A* algorithm is a book that agents are trying to check out from the manager. The path request manager is responsible for controlling which agent gets to read the A* algorithm. Only one agent can check out the A* at a time. If there is more than one agent trying to access the A* at the same time, then they must wait their turn in a virtual queue line.

Before entering this virtual queue, agents must submit a *path request* to the manager class. A path request describes where the agent is standing and where the agent wants to navigate to. The path request manager takes these requests, one at a time, and feeds them into the A* algorithm. If the A* was successful at finding a path, the path request is sent back to the agent as a list of waypoints, which the agent can follow.

In summary, the A* algorithm can only be used by one agent at a time, which is a problem if there are too many agents in the simulation. So, the path request manager class controls which agent gets to use the algorithm. Agents must submit *path requests* to the manager and wait in a virtual queue until it is their turn to use the A*.

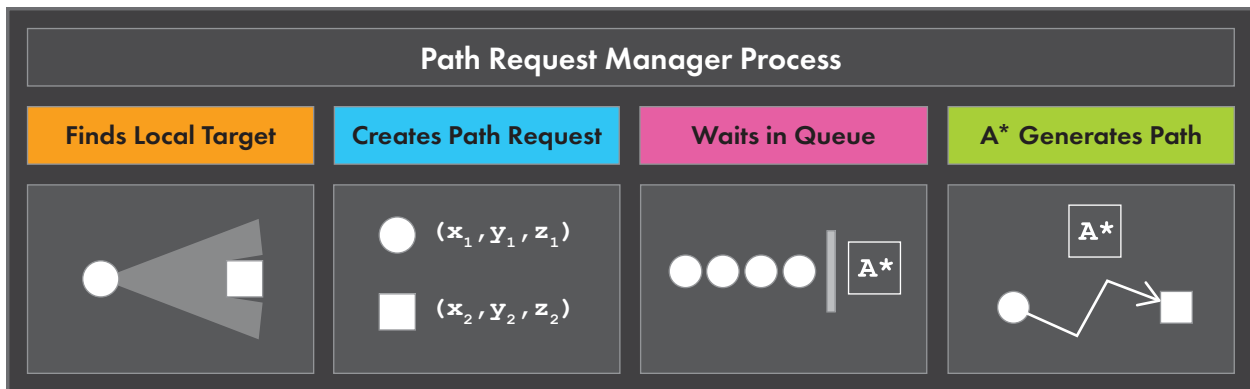


Figure 3.4.e: Path request manager process.

Heap Class

The heap class optimizes the way nodes are sorted during pathfinding. An architectural space that is 10 x 10 metres square can require the A* algorithm to calculate up to 250 nodes, which can be slow. So, the heap class is meant to organize the nodes in a way that makes it easier to compare node costs.

Heap is an abstract tree-based data structure that orders information based on its priority (cost; not airport priorities). Instead of comparing every node in a set, heap only compares nodes that are part of the same tree branch. Like a family tree, heap works by comparing *child* nodes to *parent* nodes. The rule of Lague's heap tree is that a parent node should always have a lower cost than a child node.

When the heap function comes across a new node, it adds the node to the last tree branch as a new child (Fig.3.4.f). Heap then checks if the child node has a lower cost than its parent on the same branch. If the cost is higher, the child node stays where it is. If it has a lower cost, then the child swaps with the parent's position in the tree. Heap then continues to check further up the tree branch to see if the child's cost is lower than its grandparents. If the child's cost is lower than all its grandparents, then it becomes the top position on the tree, or the first position.

When the A* pathfinding algorithm is looking for the next lowest cost node, it only needs to take the value of the node in this first position, which will always be the lowest cost. Therefore, heap saves the pathfinding time from having to check all the other child nodes in the rest of the tree.

In summary, heap organizes pathfinding nodes in a tree-based structure to reduce the time it takes to compare node costs in large architectural spaces. Heap adds new nodes to the tree as a child node to a parent. If the child has a lower cost than its parent, then it moves up the tree. The node at the top of the tree, or first position, is then given to the A* algorithm as the lowest cost node in that set.

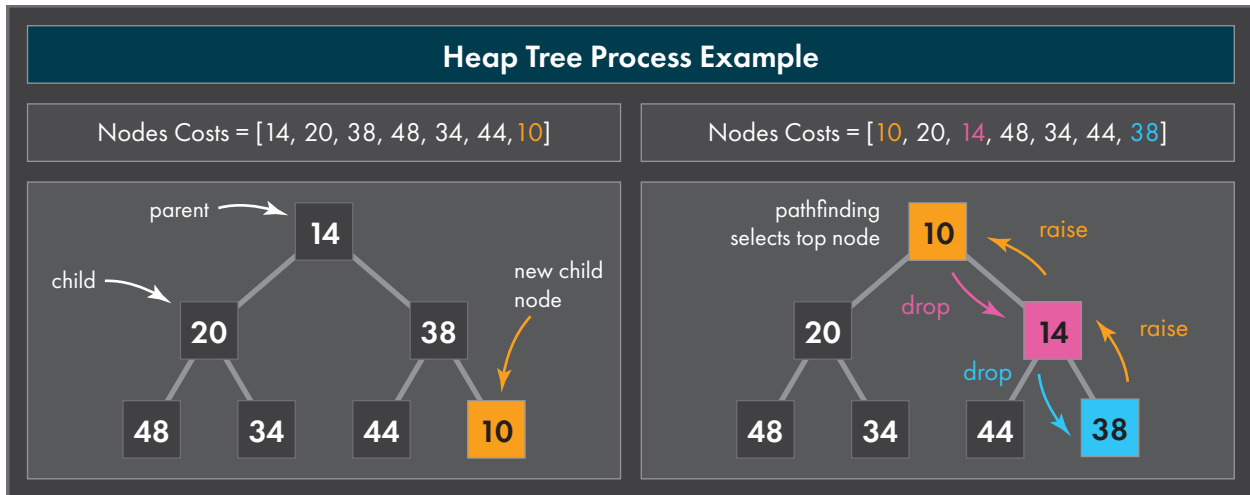


Figure 3.4.f: *Heap tree process, based on diagram by Lague (2014), drawn by authour.*

Field of View Editor Class

The field of view editor class allows the agent's field of view to be displayed during the simulation. The editor works by using Unity's built-in editor components called *handles*. As described in the field of view class, the agent's field of view is made up of thin triangles. Each triangle is constructed by the agent's *viewpoints*, or the locations projected rays intercept with an object. The editor uses the handles to trace each of the triangles along every viewpoint, based on the given angle, radius, and direction the agent is looking. The editor then assigns these triangles a handle colour for rendering, which is then displayed by the editor while the simulation is running.

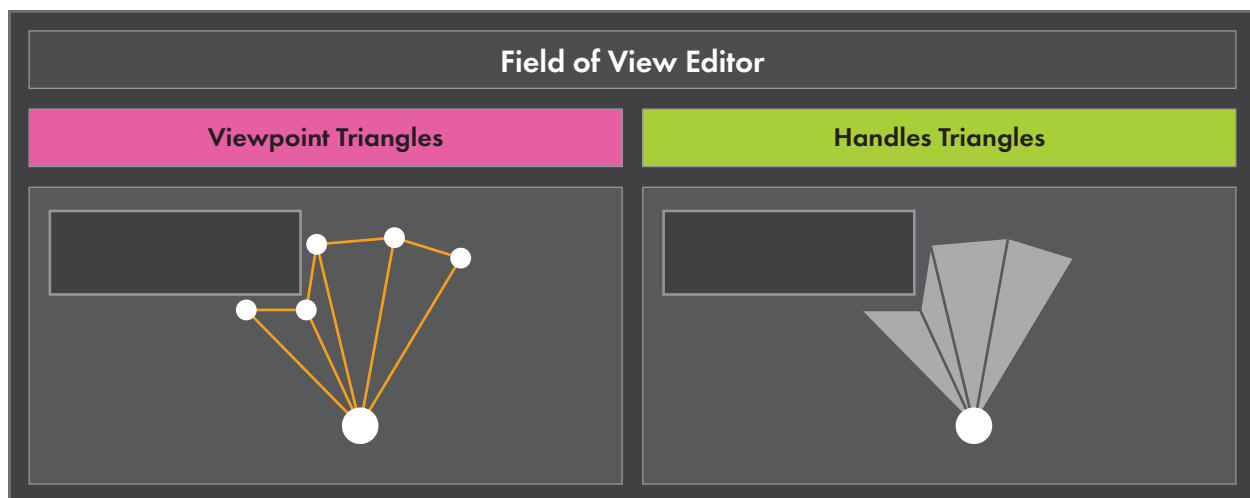


Figure 3.4.g: *Field of view editor displays "handles triangles".*

Chapter 3.5

Simulation Assumptions and Limits

While creating this agent simulation, the thesis made numerous assumptions and simplifications to get a working model. This includes limiting the function of the agents, passenger behaviour, airport processes, architectural conditions, and simulation capacity. A model of a system is only as accurate as the information the designer chooses to add to it. Therefore, although the current state of the simulation has its limitations, it should still be representative of basic people interacting with architectural conditions.

Agent Model

Perception:

Agents are perceptive of their environment. Any objects that fall within their field of view can be perceived. The simulation illustrates agent's line of sight as a 2D isovist. However, agents can perceive elements within a 3D sphere of space around them, including wayfinding signs above their heads.

The way agents read information assumes they have ideal vision. This means they can read objects at the far end of a room as clearly as they can read objects right in front of them. Agents will recognize an object the instant it falls within their line of sight. It is assumed agents can see the entire length of the building. The simulation allows the agent's field of view to have different sizes of view radius and angle. So, changing the field of view can influence the location where they perceive their surroundings.

Navigation:

Agents need to be assigned a goal and are always working towards an end target. There are moments in the simulation when agents are wandering or waiting, but these are meant to control agents' movement on their way to the gates. Navigation decisions are only updated after agents reach a local target. They cannot adjust their path after they started walking. This issue is minimized by shortening how far agents choose to walk (less than 2 m ahead), and as a result, increases the frequency of direction choice updates.

Agents can avoid walking through obstacles and other physical objects. They are aware of other agents and slow down in crowded areas, but they can walk through other people if they become stuck. The simulation does have functions to avoid other people, but it is disabled because of navigation issues and simulation bugs. A better model for avoiding people can replicate Mass Motion's agent "feelers" (proximity detection), which forces agents to navigate around crowded spaces.

Crowd Behaviour:

The thesis cannot model the behaviour of groups of people to the level of a proper crowd simulation. Agents are aware of other agents in the simulation, but they can only adjust their walking speed to avoid colliding with others. It is not capable of modelling social structures or group dynamics. For this reason, the simulation does not consider social interaction between agents, like a family or a group of friends who may walk together.

The simulation can only hold about 30 passengers at anytime. More people than this causes the model to slow down considerably and lag. The simulation can continue to add more people over time as passengers exit the terminal, as long as the number of active agents does not exceed the given capacity.

Memory:

When agents start the simulation, it is assumed they have no knowledge of the building. they gain knowledge of where things are from walking through the terminal. Agents can memorize airport domains, their priorities, and architectural experience. This includes if they checked into their flights, what their departure gate is, and how satisfied they are with conditions in the terminal.

Agents do not remember where they have already walked, or areas they have already been through. This allows agents to walk back the way they came if they get lost or stuck in dead-end corridors. Passengers are restricted to walk through security before checking in and have memory of being cleared through security after passing the checkpoint.

Agents have short term memory when reading a sign to know which direction to walk. When passengers see a sign, they are always inclined to read it, even if they have seen it before. This ensures agents who are lost can read the same sign again to relearn where to go.

Passenger Model

Characteristics:

Agents in these simulations approximate human behaviour that is typical of passengers at an airport terminal. Agents are considered passengers when they are given characteristics. In every simulation test, the characteristics are based on the International Maritime Organization (IMO) 1238 guidelines for evacuation simulations, which is the same standard used to validate MassMotion crowd simulations. ^[1]

Agents are given random age, gender, and walking speeds based on the probability distributions from IMO. The simulation assumes that passengers are fully able adults, who can function independently. Age is assigned from 18 to 75 years old. Although it is possible to assign ages outside this range, there is no behaviour for children. Passenger walking speeds are dependant on age and gender, but only walking speed affects crowd behaviour. The simulations only consider male and female genders to align with the IMO guidelines. It is possible to define non-binary genders with the existing functions, but IMO has no walking speed data associate with this. Passengers are also given random names to help identify them, but the names do not affect their behaviour.

Passenger Types:

All passengers are considered part of general boarding. There are no business travellers, “frequent flyers”, or “preferred” airline customers. The primary demographics for this thesis are passengers who do not travel often. They are more likely to rely on using their surroundings to inform decisions, rather than relying on past experience or memory.

The differences between passengers are defined by their airport priorities. As explained in earlier chapters, the thesis selected six common priorities in the processing and non-processing airport domains, as described by Wiredja et al. All passengers have the same set of priorities, but their importance is randomly assigned. The thesis assumes that a random distribution of priorities is representative of an airport population for testing. However, further research is needed to determine if different cultures or regions place higher importance on certain airport domains than others.

1. Arup. “The Verification and Validation of MassMotion for Evacuation Modelling.” Ove Arup & Partners Ltd. (August 10, 2015): 2.

Disabilities:

Disabilities are represented in the agent model with limited scope. The simulations do consider passengers with walking disabilities, based on the distribution listed in IMO. However, walking disabilities only makes agents speed slower. Disabled people in this simulation do not include equipment like walkers and wheelchairs, and they cannot be guided by social support workers.

There are no visually impaired passengers represented in these simulation tests. However, it is possible to model a passenger with visual disabilities by restricting a passenger's field of view to a smaller radius. This may approximate people who have restricted vision or different levels of perception. But further research is needed to better represent these conditions.

Non-Processing Behaviour:

Passengers can wander around the gate areas, but there are no functions for shopping or eating in the retail and concession spaces. People can interact with food and retail counters to simulate purchasing something. However, once the process is done, the agent returns to wandering. Most of the time, if agents are wandering around, they are searching for their gate.

Passengers have awareness of food and washroom areas, but they do not have hunger or bladder needs. Instead, the simulation randomly assigns passengers with a high priority for food or washroom areas, which approximates people who are hungry or need to use the washroom. Essentially, people who prioritize the washroom are more likely to spend time searching for a washroom in the airport. Although, a more developed simulation should consider passengers with changing hunger and bladder needs over time.

Airport Model

The agent simulation only considers passenger flows for the departure sequence, which includes check-in, security screening, departure gates. Passenger flows like arrival, connecting flights, curbside, and transit are not modelled for this thesis. Additionally, there is no modelling of the baggage systems, service spaces, or aircraft logistics.

Processing:

All passengers follow the same procedures in check-in and security for general boarding. There is no priority check-in lanes or priority boarding. The only variation is the time spent at service counters or screening machines, which the simulation randomly assigns. All passengers check in a single bag, which they are pulling with them when they enter the terminal. There is no carry-on

luggage or personal items. Additionally, passengers move and behave the same whether or not they are pulling a suitcase.

Passengers can walk through queue lines to approximate waiting in line. However, the thesis does not use resources to allocate selected counters, which is more typical of a queuing simulation like Arena. Passengers will walk through queue lines for the security area, but do not stop to wait in line. To account for this behaviour, a time delay factor is applied when calculating architectural value at the end of the process.

Additionally, the simulations do not consider customer service, since there are no airport staff. As explained in earlier chapters, customer service is related to passenger experience, but it does not impact architectural space directly.

Flight Times:

Gates are assigned a random departure time, relative to the length of a given test. It is assumed there is one flight departing from each gate. Flight delays are not considered in these simulations. When the simulation reaches the assigned departure time, it is assumed every passenger begins boarding their flights when they reach the gates. Passengers can wait at the gate if they are early to their flight. These wait times are accounted for in the calculation of architectural value as a random variable.

Architectural Model

The scope of the architectural environment is limited to an airport terminal or similar transit facility. The thesis covers basic architectural features like walls, doors, and thresholds. Transparent materials like windows are not included in these simulations. Although, the thesis expects that transparent materials like glass would allow passengers to see into other areas but restrict them from walking to the other side. The terminal building is limited to a single storey. There is no vertical circulation like stairs, escalators, or elevators. Multiple floors can only be added if navigation functions, like A*, were rewritten to consider movement in vertical direction (y-axis variable in Unity).

The size of the terminal building is limited by the agent's navigation. Terminal buildings bigger than about $100 \times 100 \text{ m}$, or $10\,000 \text{ m}^2$, becomes too laggy (slow) to simulate because of the high number of navigation nodes to compute for the A* pathfinding. Larger terminal models can instead disable A* pathfinding and use vector navigation. Vectors can produce equivalent behaviour to the A* pathfinding if agents are navigating over short distances.

Part 4:

Simulation Tests

Part 4 goes through the tests and experiments to show how the agent simulation works. Chapter 4.0 goes through the verification and validation tests based on existing standards. Chapter 4.1 demonstrates specific components introduced by this thesis, like wayfinding, field of view, and priorities. Chapter 4.2, experiments with a hypothetical terminal layout to show how changing spaces and agent priorities affects architectural value. Finally, chapter 4.3 tests how the agent-model works in an existing airport. The thesis model compares an airport with good passenger experience to an airport that has an objectively worse passenger experience, to see if it can replicate the same results.

Chapter 4.0

Verification and Validation Tests

The first set of tests concerns verification and validation of the simulation. The intention is to use an established standard to check if basic components of the agent and environment are working correctly. Some of these components include:

- Agents move at realistic speeds
- Agents can navigate around walls
- Agents are assigned correct characteristics

The thesis uses standardized tests from the International Maritime Organization (IMO) and the National Institute of Standards and Technology (NIST), which are used to validate MassMotion.

^[1] These standards include eleven tests from IMO 1238 ^[2] and another eight tests from NIST 1822. ^[3] Their primary purpose is to evaluate if a simulation can correctly represent human behaviour during an emergency evacuation. Since the thesis only needs to verify basic agent movement, not all of these tests are considered. Each test covers a specific condition like walking speed, emergency response time, crowd flow rates, group behaviours, and social influences. To help determine which tests are relevant, the thesis organizes the IMO and NIST tests into four categories: *Architectural Conditions*, *Crowd Dynamics*, *Social Behaviour*, and *Emergency Situations* (4.0.a). Out of these categories, the thesis determines it has the conditions necessary to recreate the following tests:

- IMO Test 1: Corridor Walking Speeds (*Architectural*)
- IMO Test 6: Rounding Corners (*Architectural*)
- IMO Test 4: Flow Rates (*Crowds*)
- IMO Test 7: Demographics (*Social*)

¹. Arup. “The Verification and Validation of MassMotion for Evacuation Modelling.” Ove Arup & Partners Ltd. (August 10, 2015): 2.

². IMO. “Guidelines for Evacuation Analysis for New and Existing Passenger Ships.” International Maritime Organization (IMO). MSC.1/Circ.1238. October 30, 2007. <https://nsof.no/media/1129/imo-msc-guidelines-for-evacuation-etc.pdf>.

³. Ronchi, Enrico; Kuligowski, Erica D; Reneke, Paul A; Peacock, Richard D; Nilsson, Daniel. “The Process of Verification and Validation of Building Fire Evacuation Models.” Technical Note (NIST TN) - 1822, (November 2013), <http://dx.doi.org/10.6028/NIST.TN.1822>.

IMO and NIST Verification Tests				
	Test Name	IMO	NIST	Tested
Architectural Condition	Speed in a corridor	Test 1	Verf.2.1	<input checked="" type="checkbox"/> Yes
	Speed up a staircase	Test 2	Verf.2.2	<input type="checkbox"/> No
	Speed down a staircase	Test 3	Verf.2.2	<input type="checkbox"/> No
	Movement around a corner	Test 6	Verf.2.3	<input checked="" type="checkbox"/> Yes
	Elevator usage	---	Verf.2.7	<input type="checkbox"/> No
Crowd Dynamics	Horizontal counter-flows (rooms)	Test 8	Verf.2.8	<input type="checkbox"/> No
	Group Behaviour	---	Verf.2.9	<input type="checkbox"/> No
	Exit route allocation	Test 10	Verf.3.1	<input type="checkbox"/> No
	Congestion	Test 11	Verf.5.1	<input type="checkbox"/> No
	Maximum flow rates	Test 4	Verf.5.2	<input checked="" type="checkbox"/> Yes
	Exit flow for a large room	Test 9	---	<input type="checkbox"/> No
Social Behaviour	Assigned demographics	Test 7	Verf.2.4	<input checked="" type="checkbox"/> Yes
	People with movement disabilities	---	Verf.2.10	<input type="checkbox"/> No
	Social Influence	---	Verf.3.2	<input type="checkbox"/> No
	Affiliation	---	Verf.3.3	<input type="checkbox"/> No
Emergency Situation	Pre-evacuation time distributions	Test 5	Verf.1.1	<input type="checkbox"/> No
	Reduced visibility vs walking speed	---	Verf.2.5	<input type="checkbox"/> No
	Occupant incapacitation	---	Verf.2.6	<input type="checkbox"/> No
	Dynamic availability of exit	---	Verf.4.1	<input type="checkbox"/> No

Figure 4.o.a: IMO and NIST verification tests for evacuation simulations, based on diagram from the NIST 1822 (2013), redrawn by author.

Test 1: Corridor Walking Speeds

The first test is based on IMO 1238 Test 1 and NIST 1822 Verf.2.1. The setup and conditions are illustrated in Fig.4.o.b. This test verifies if an agent can walk down a straight corridor at a constant speed over a given amount of time. The test assigns the agent a walking speed of 1 *m/s*, which is representative of a typical adult. ^[4] The geometry of the corridor is 2 *m* wide by 40 *m* long, to the inside dimensions of the walls. For this thesis's setup, it makes sure that the entrance and exit portals are just before and after the zero and 40 *m* marks, respectively, so that agents are travelling the full 40 *m* distance. The test also expects to see a deterministic result, since the walking speed and distance are a constant value. ^[5] This means, if the agent is travelling at 1 *m/s* over a distance of 40 *m*, they should always complete this in 40 seconds.

The images in Fig.4.o.d illustrate the test in the agent simulation. The procedure begins with an empty corridor. The user of the simulation starts the test by spawning (generating) an agent at the start portal. The agent immediately begins walking down the corridor to the other end using A* navigation (Fig.4.o.c). When the agent reaches the end of the exit portal, they are removed from the simulation and their final time is recorded. The simulation is setup so that multiple agents are continuously spawned every 40 seconds, to test multiple agents at once. Agents were spaced out far enough so that they do not interfere with each other.

The final results of Test 1 are illustrated in Fig.4.o.e, which demonstrates a consistent time of 40 seconds. Therefore, this test is passed.

It was observed that agents walk directly to the target using A* navigation, since they have full view of the exit at the other end. Out of curiosity, the thesis also experimented with other conditions like different perception levels and walking speed. It was also observed that if agents had a restricted field of view shorter than the 40 *m* corridor, then they navigated to local targets, which they followed until the exit was in their field of view. But this had no significant change to the agent's travel time. The data from all trials conducted for this test are listed in Appendix A.

4. Ronchi et al (NIST). "Verification and Validation". 8.

5. Ronchi et al (NIST). "Verification and Validation". 8.

Test 1: Corridor Walking Speeds

Standard

IMO 1238 Test 1 / NIST 1822 Verif.2.1

Status:



Passed

Purpose

Determine if agents can move down a straight corridor and maintain a constant speed over time.

Conditions

- Corridor is 2 m wide and 40 m long
- Agent walking speed is 1 m/s
- Agents must walk from one end of the corridor to the other in 40 seconds

Floor Plan

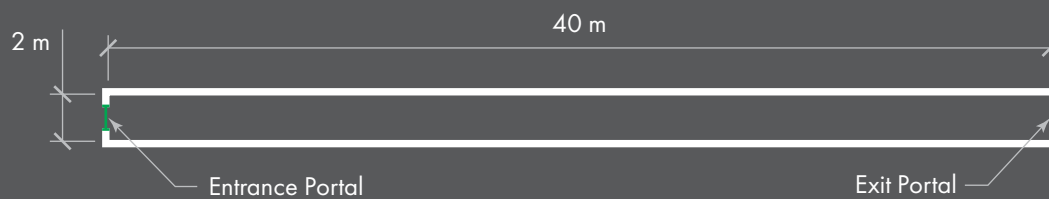


Figure 4.o.b: Setup and conditions for test 1; the test was successful.



Figure 4.o.c: *Agent walking in corridor from entrance.*



Figures 4.o.d: *Screen captures at time intervals during the test of one agent.*

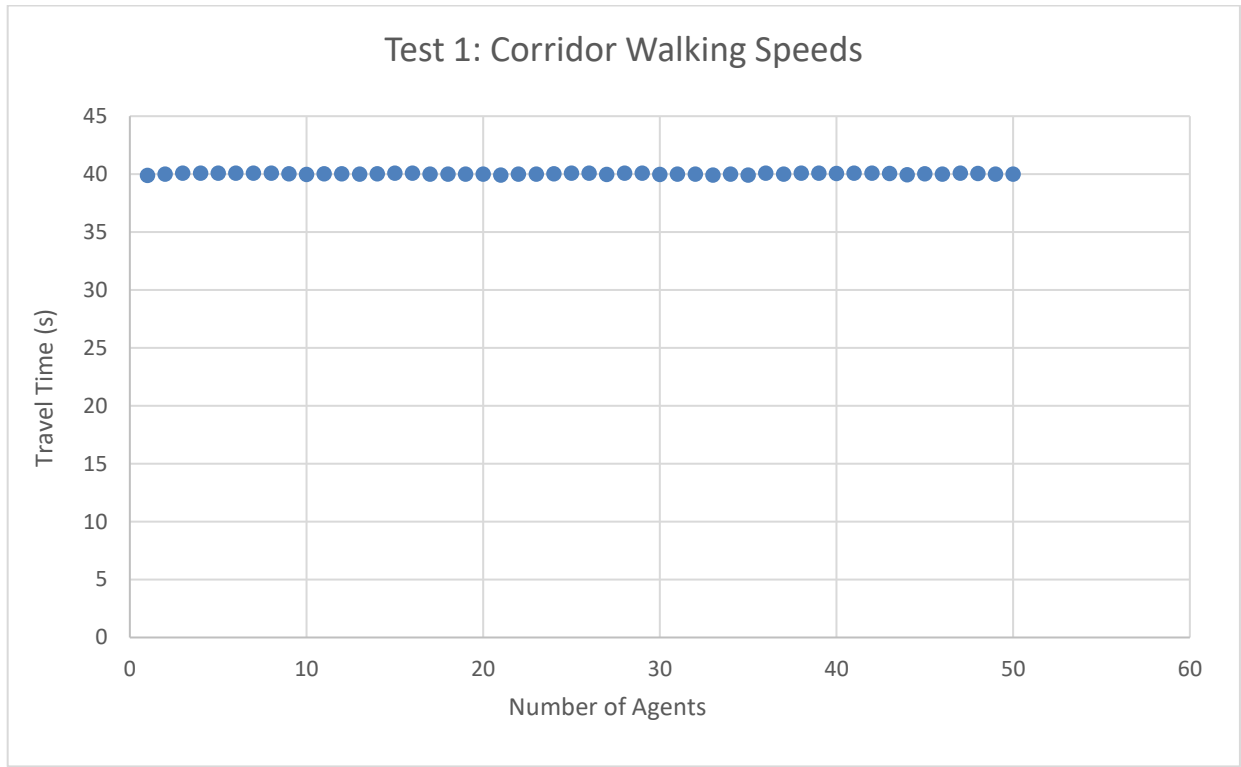


Figure 4.0.e: *Travel times for a sample of 50 agents is consistently 40 seconds.*

Test 2: Round Corners

The second test is based on IMO 1238 Test 6 and NIST 1822 Verf.2.3. This test verifies if agents can navigate around a corner without walking through walls or clipping through the environment boundaries. The geometry of the test is a left-hand 90° corner, which consists of a 2 m wide corridor and 10 m long leg segments. The setup requires 20 people to be uniformly distributed within a starting area that is 2 x 4 m (Fig.4.o.f). The direction of travel goes from the starting area, around the corner, to the exit portal at the end of the corridor.

In this test, agents use A* navigation and have a walking speed of 1 m/s. Agents are aware of walls and boundaries. However, due to debugging issues with the proximity function, collisions between agents are disabled for this test. Some of the debugging issues cause agents to walk backwards, which is not representative of basic navigation. The thesis believes turning off agent proximity is reasonable because the purpose of this test is only to show how agents navigate around walls.

The test begins with agents walking from the starting area towards the corner (4.o.g). Since they are navigating using local targets, most agents navigate to a spot in front of the inside corner. The agents appear to bunch up together at this point, since they are walking to similar locations (Fig.4.o.h). As they reach the corner, it appears some agents can see further down the corridor, whereas other agents cannot. Those that can see further, start walking towards the exit. Agents who cannot see further then take a wider path towards the outside corner. Taking the wider path gets them in a position to see the exit portal in their periphery, which they start walking to. As all the agents get closer to the exit, they begin to line up together since they are all navigating to the same point. Once they reach the exit portal, the agents are removed from the simulation (Fig.o.i).

Since the agents reached the exit portal without cutting the corner, this test is passed. The agents can navigate through the physical environment, despite the agents clipping through each other. The issues with agent collisions will require a better study of agent proximity to fix how the simulation is coded.

Test 2: Rounding Corners

Standard

IMO 1238 Test 6 / NIST 1822 Verif.2.3

Status:



Passed

Purpose

Determine if agents can move around a corner without walking through walls or boundaries

Conditions

- Left-hand 90° corner, 2 m wide corridor with 10 m long legs
- Starting area is 2 m x 4 m and has 20 people, uniformly distributed
- Agents must walk around corner to exit without going through walls

Floor Plan

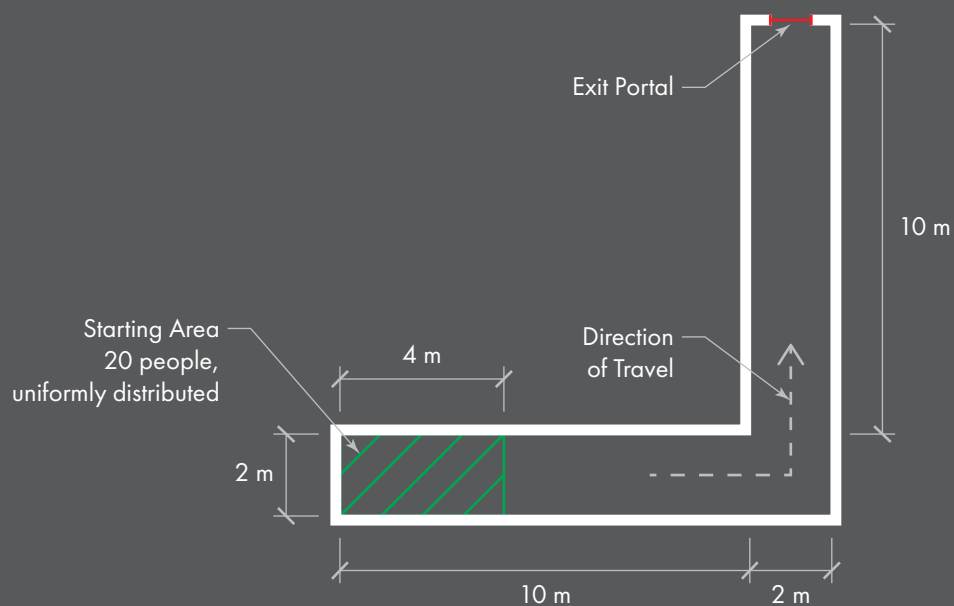


Figure 4.0.f: Setup and conditions for test 2; the test was successful.

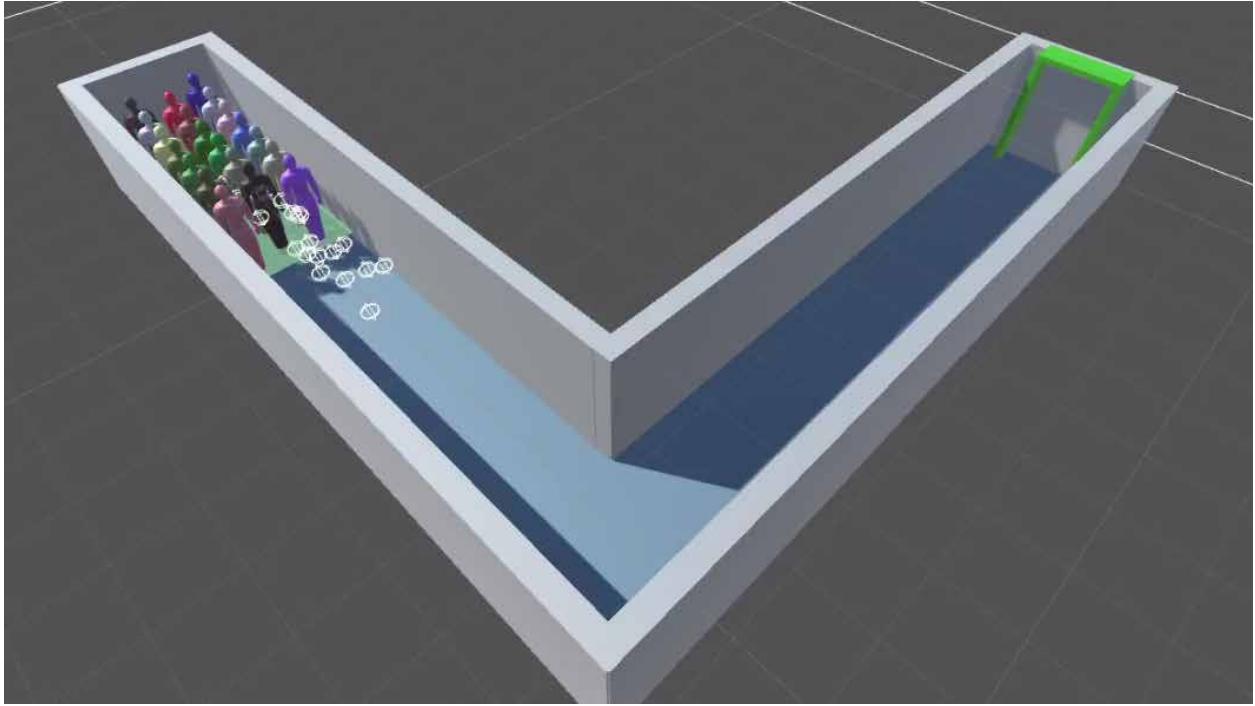


Figure 4.0.g: *Agents in the starting area.*

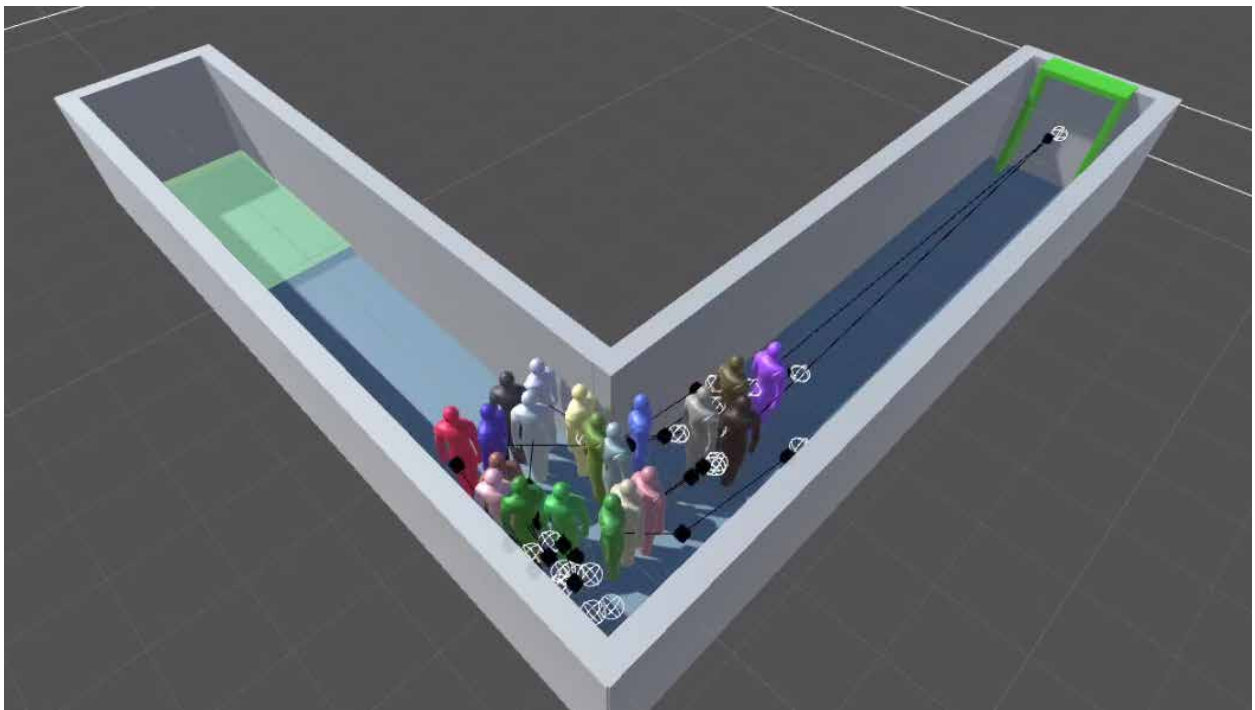
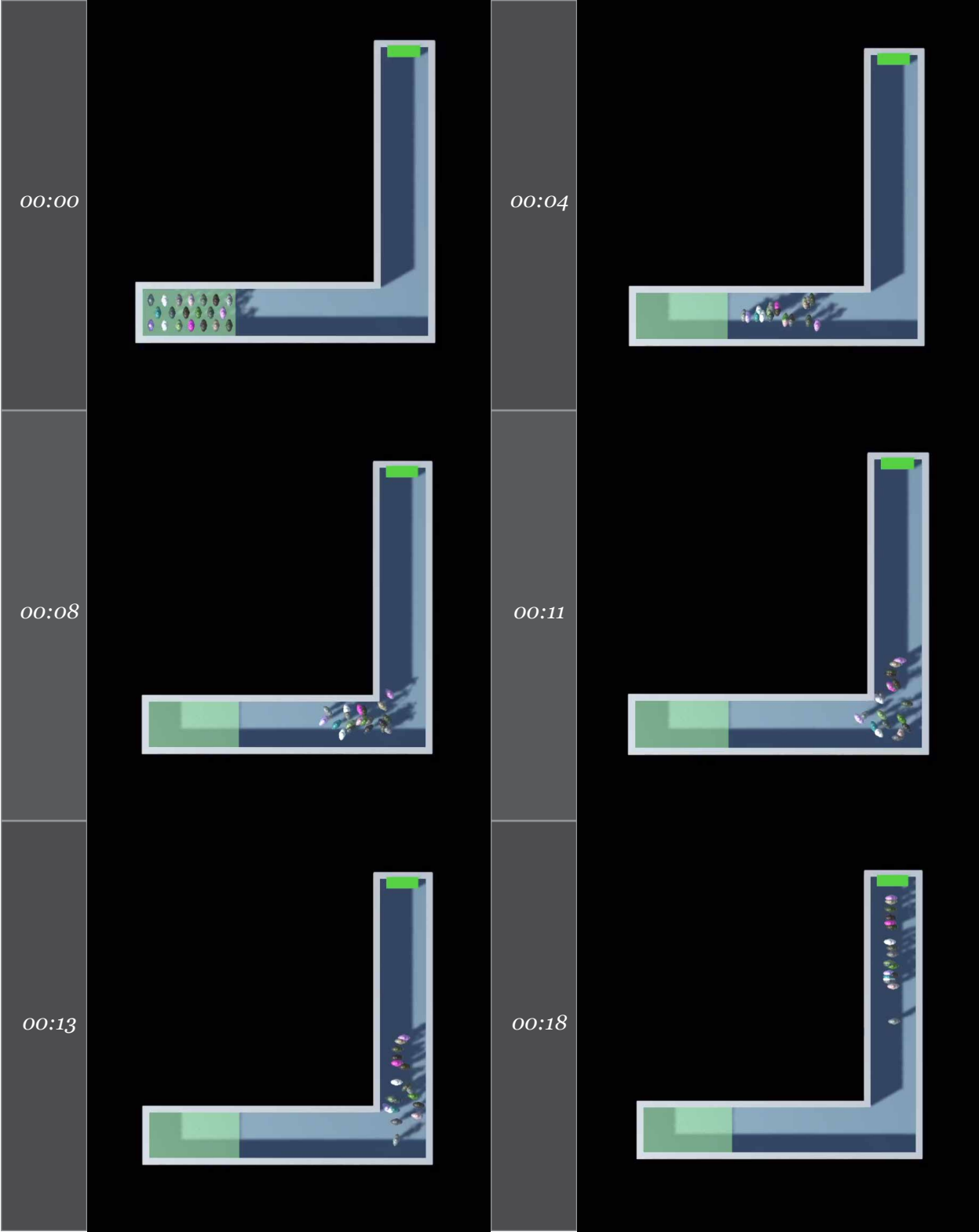


Figure 4.0.h: *Agents walking around the corner.*



Figures 4.o.i: Screen captures at time intervals during the test.

Test 3: Flow Rates

The third test is based on IMO 1238 Test 4 and NIST 1822 Verf.5.2. This test verifies how many people can pass through a doorway over time. The concept is like an hour-glass full of sand; only a certain amount of sand can physically pass through the glass over time, like people through a doorway. For this test, the flow rate through the door must not exceed 1.33 people per second (p/s) at any point, which is consistent with current evacuation research. ^[6]

The layout for this test is an 8 x 5 m room, with a 1 m opening on the 5 m wall (Fig.4.0.j). The testing population requires 100 people, who are placed in the room as the starting area. An exit portal is placed in a threshold just beyond the opening, which forces agents to completely pass through the doorway before exiting the simulation.

Agent walking speeds are randomly assigned based on the IMO population distribution. The thesis's agent simulation attempts to model crowd dynamics using an agent proximity function. If agents are too close to each other in a crowd, then the proximity function restricts their walking speed, until there is more space to move. However, as seen during testing, this proximity function did not provide ideal crowd behaviour.

The test starts with 100 agents standing in the room. The first agents that exit through the doorway are the ones standing closest to the opening (4.0.k). However, it becomes apparent that the crowd's movement is inconsistent. Only the agents at the front of the group move forward. The other agents at the back of the room appear to be deadlocked behind each other. The crowd does not fill into open areas near the doorway. Instead, the agents appear to wait until the space directly in front of them is free. There are instances when none of the agents move because they are standing too close to each other (Fig.4.1.m). Sometimes multiple agents move at the same time. This results in agents clumping together, which causes spikes in the flow rate (Fig.4.0.l).

The thesis has experimented with different proximity radii and walking speeds, which is detailed in Appendix A. It was discovered that smaller radii and slower walking speeds give the closest flow rates to 1.33 p/s. However, the inconsistency is still an issue. Under the same conditions, the flow rates in Trial 11 (Fig.4.0.n) and Trial 12 (Fig.4.0.o) do not maintain a constant rate. Instead, there are spikes that still exceed 1.33 p/s. Therefore, this test failed to produce realistic flow rates. To minimize these issues for later tests, the thesis will experiment with low density crowds, unless a better solution for agent proximity can be solved.

6. Ronchi et al (NIST). "Verification and Validation". 9.

Test 3: Flow Rates	
Standard	IMO 1238 Test 4 / NIST 1822 Verif.5.2
Status:	● Failed
Purpose	Determine if the number of people passing through a doorway is below a certain capacity
Conditions	<ul style="list-style-type: none"> - 8 m x 5 m room with a 1 m opening (in the 5 m wall) - 100 people starting in the room - Flow rate should be less than 1.33 p/s
Floor Plan	<p>The floor plan shows a rectangular room with a width of 8 m and a height of 5 m. The left portion of the room is shaded with green diagonal lines and labeled 'Starting Area 100 people'. On the right wall, there is a doorway labeled '1 m Opening' and 'Exit Portal'.</p>

Figure 4.0.j: Setup and conditions for test 3; the test was unsuccessful due to people clumping together causing inconsistent flow rates.

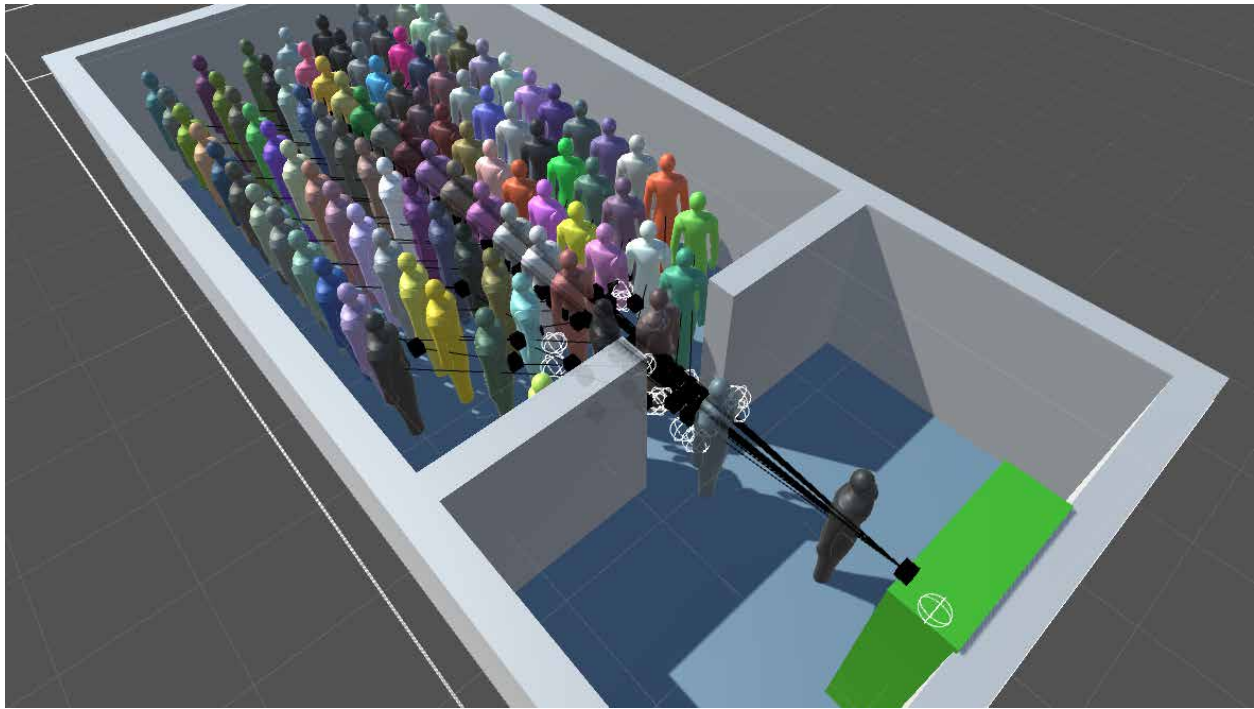


Figure 4.0.k: *Agents walking through opening.*

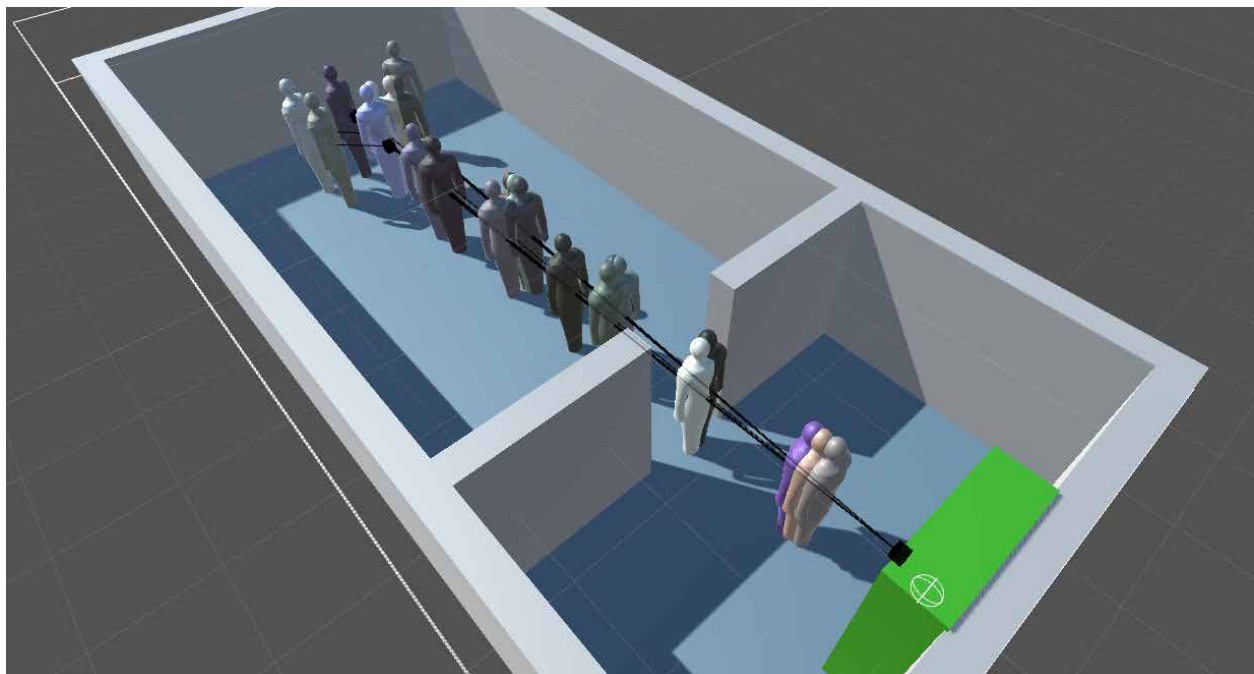


Figure 4.0.l: *Agents clumping together causes spikes in flow rate.*

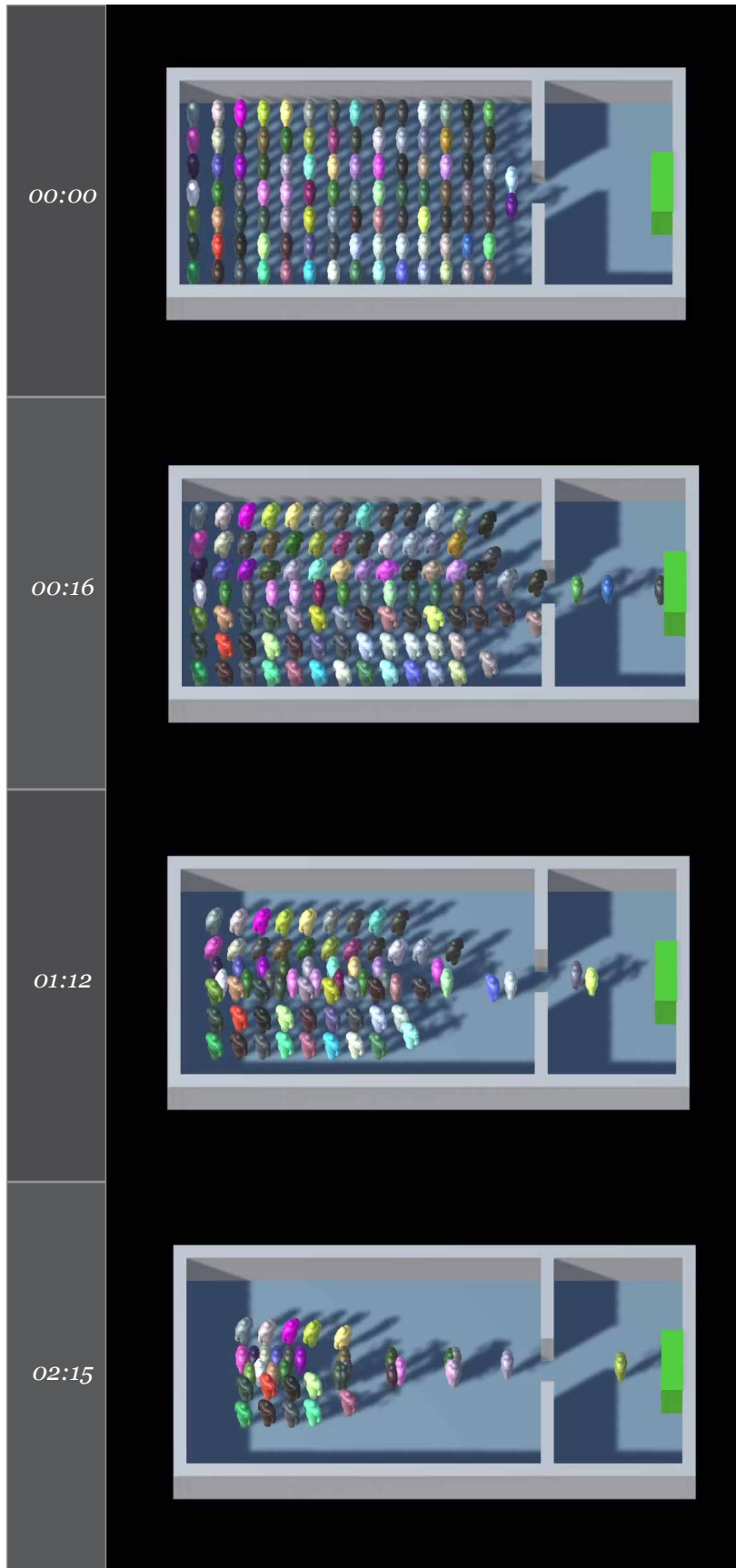


Figure 4.1.m: *Screen captures at time intervals during the test.*

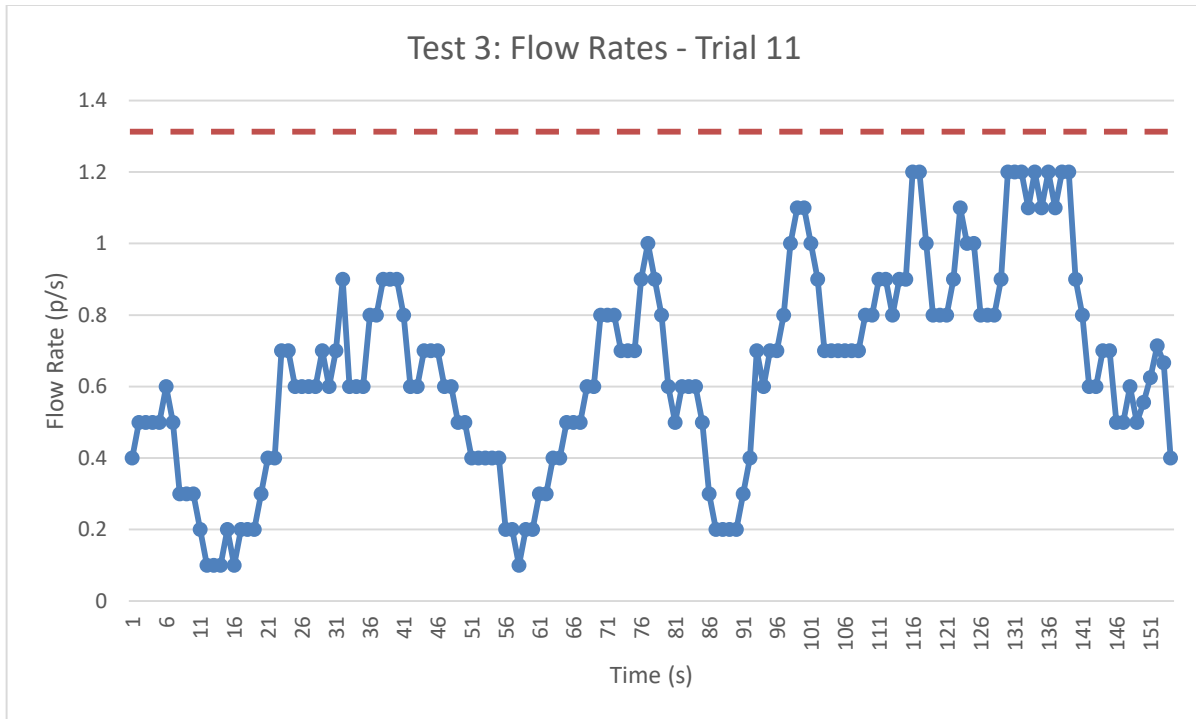


Figure 4.o.n: Max flow rate of 1.2 p/s, below 1.33 p/s (redline), but is not maintained over time.

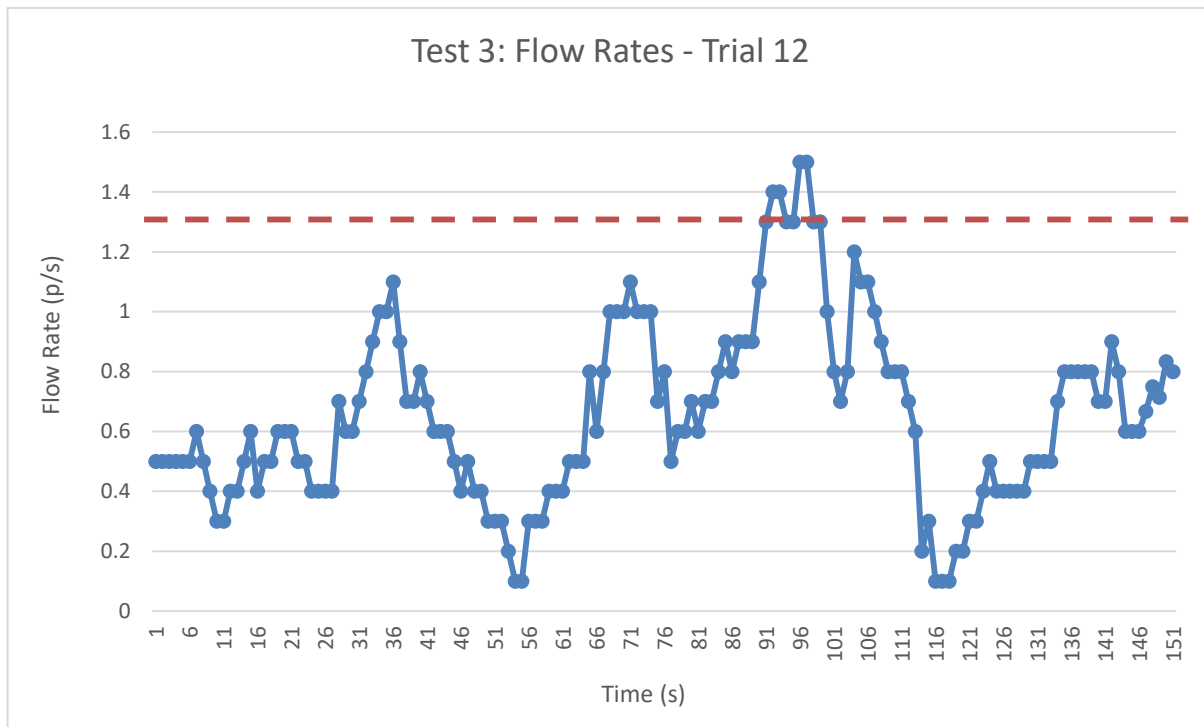


Figure 4.o.o: Max flow rate spiked to 1.5 p/s, above redline, despite having same conditions as trial 11

Test 4: Demographics

The fourth test is based on IMO 1238 Test 7 and NIST 1822 Verf.2.4. This test verifies that the agent characteristics are correctly assigned based on the demographic distribution listed in IMO table 3.4. The test requires a sample population of 50 people, who are males between 30 to 50 years old. Based on table 3.4, agents must be assigned a random walking speed between 0.97 and 1.62 *m/s*. Unlike the first corridor test, which was deterministic, this test is stochastic. This means random input speeds will produce random output times. As a result, the test is judged on the distribution of multiple agents. If the test is successful, then the average agent walking speed must be around 1.295 *m/s*.

There are no layout requirements, so the thesis uses a 20 x 20 m square room, with a starting portal and exit portal on opposite walls (Fig.4.0.p). The average walking speed is calculated as the time it takes agents to travel a distance of 20 *m*.

The procedure for this test is similar to the first corridor test. The simulation starts with an empty room, and an agent is spawned one at a time (Fig4.0.r). Agents immediately begin walking once they spawn into the simulation using A* navigation. Since it is an open room, agents can see the exit portal (Fig.4.0.q). Therefore, agents follow a straight path to the door. Once the agent reaches the exit their time is recorded, and they are removed from the simulation. This process is repeated for the population of 50 people, which is controlled by the spawner utility. It automatically generates a new agent every 25 seconds, making sure they are added after the other agents cleared the room to avoid interference.

The final result for Test 4 is illustrated in Fig.4.0.s, which illustrates the distribution of walking speeds. As it shows, the walking speeds follow a uniform distribution and with an average of 1.28 *m/s*, which is within 1% of the expected value. Therefore, the agents are assigned the correct demographics and the test is passed. The full list of trials performed for this test is recorded in Appendix A.

Test 4: Demographics

Standard

IMO 1238 Test 7 / NIST 1822 Verif.2.4

Status:

 Passed

Purpose

Determine if agent characteristics are assigned correctly based on the population distribution from IMO Table 3.4

Conditions

- Test a population of 50 people, male, aged 30 to 50
- Randomly distributed walking speeds between 0.97 and 1.62 m/s
- Average walking speeds need to be equal to 1.295 m/s

Floor Plan

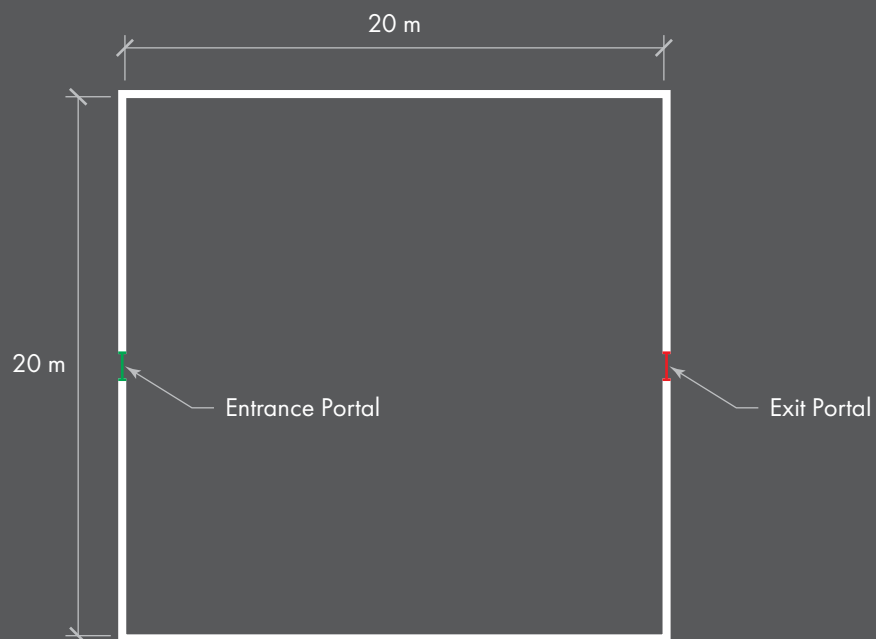


Figure 4.0.p: Setup and conditions for test 4; the test was successful.

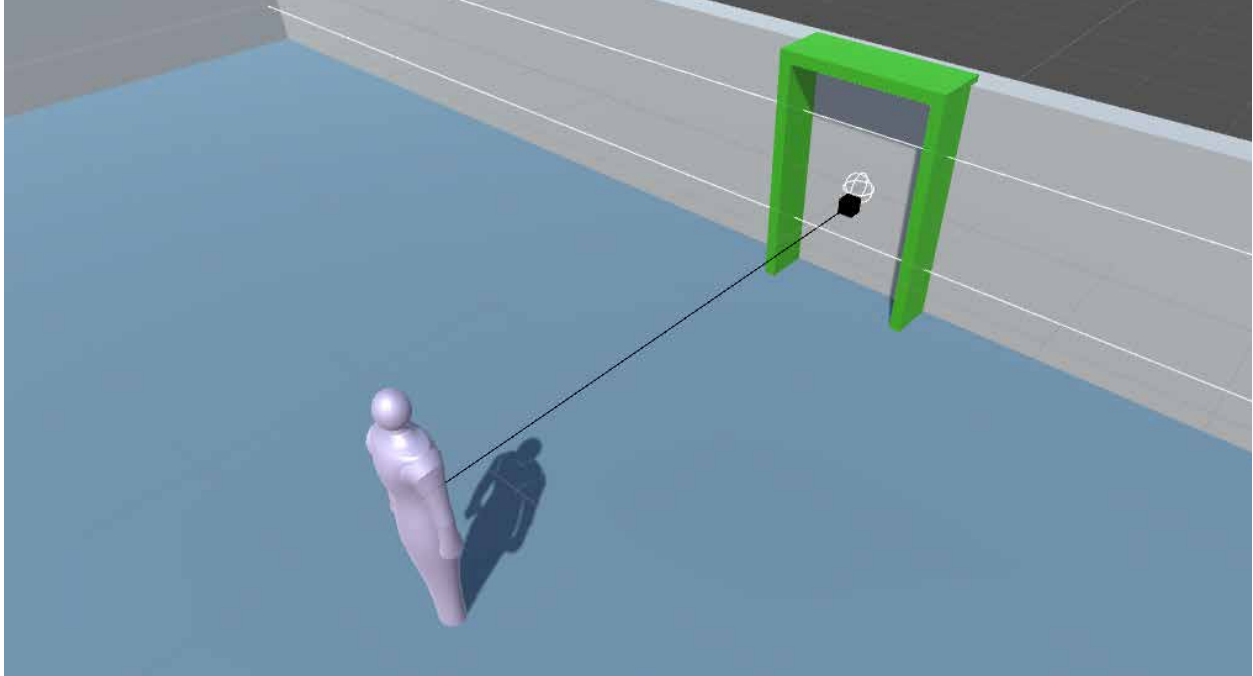


Figure 4.o.q: Agent walking to the exit portal.

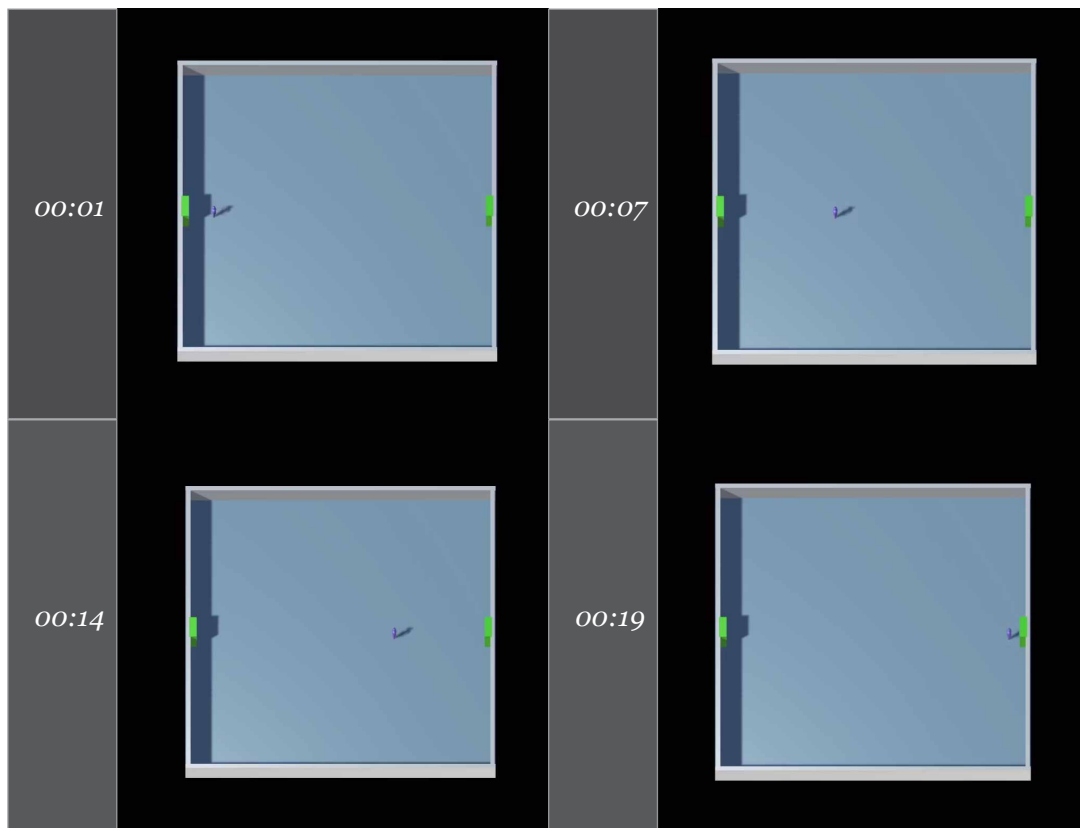


Figure 4.o.r: Screen captures at time intervals during the test of one agent.

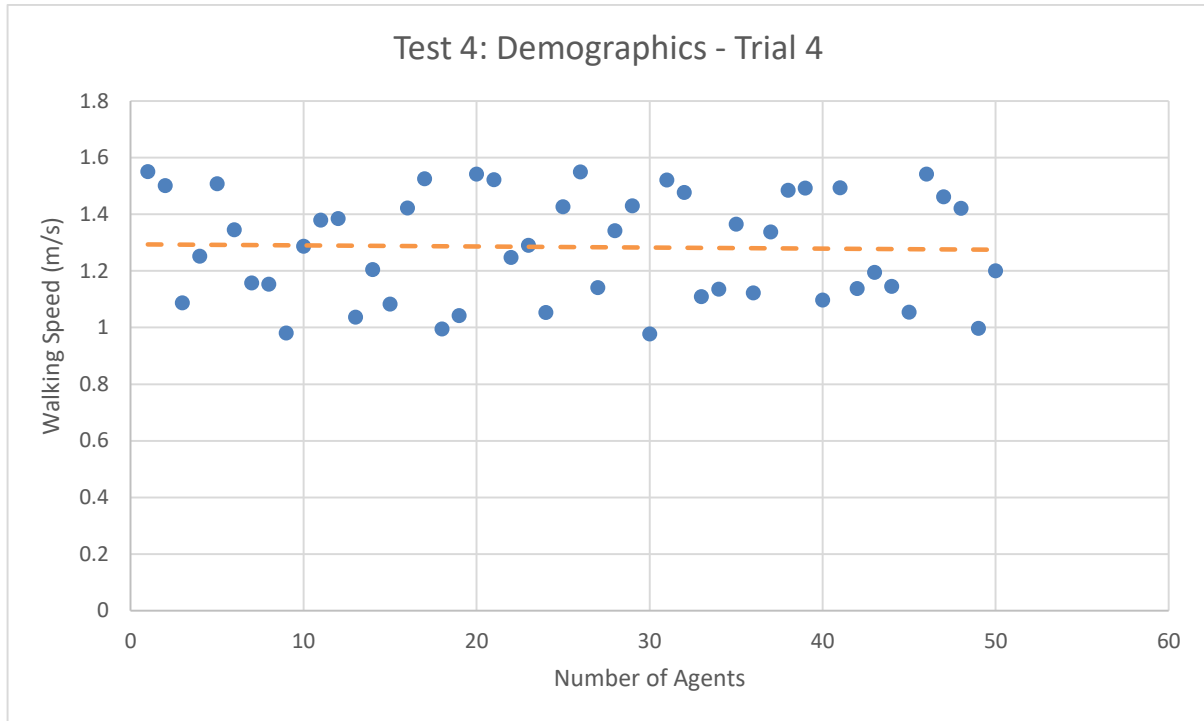


Figure 4.0.s: *Walking speed follow a uniform distribution, with an average of 1.28 m/s.*

Summary

In summary, this chapter explored verification and validation tests to determine if basic components in this agent simulation are working correctly, such as speeds, boundaries, and characteristics. The thesis used tests based on the IMO 1238 and NIST 1822 standards, which are used to validate evacuation simulations, like MassMotion. Since these standards evaluate emergency evacuation, the thesis selected four tests that were relevant to basic agent behaviour. These were: 1. Corridor Walking Speeds, 2. Rounding Corners, 3. Flow Rates, and 4. Demographics. The thesis managed to verify all conditions except for flow rates due to agent proximity issues.

Corridor Walking Speeds verifies if agents can walk down a 40 m corridor at a constant speed of 1 m/s. The test is passed if agents can traverse the corridor in 40 seconds. The thesis ran a sample of 50 agents with perception and A* navigation. All agents walked directly to the end of the corridor and completed the simulation in 40 seconds. Therefore, the first test is passed.

Rounding Corners verifies if agents can navigate around a corner without walking through walls or boundaries. The test involves 20 agents to walk from one end of the corridor, around a 90° corner to the other end. The thesis disabled agent proximity due to navigation issues, since this test is only concerned with agents walking through walls. During the test, agents walked closer to the corner if they saw the exit early. Otherwise, agents took a wider path. All 20 agents walked around the corner without going through the boundaries. Therefore, the second test is passed.

Flow Rates verifies that the number of agents walking through a 1 m doorway is limited to 1.33 m/s. The thesis's agent proximity function models crowd behaviour by forcing agents to slow down if they are too close to each other. Unfortunately, this caused agents to deadlock and move only if the space directly in front was clear. Proximity issues made agent movement inconsistent. Agents would occasionally clump together resulting in flow rate spikes. As an effect, the flow rate through the door was not constant and exceeded 1.33 p/s after running multiple trials. Therefore, the third test failed.

Demographics verifies if agents are assigned the correct walking speeds based on the IMO distribution. The test required matching the average walking speed of 1.295 m/s in a population of 50 middle-aged males. The thesis managed to match the average walking speed within 1% of the true value. Therefore, agent demographics were correct, and the fourth test is passed.

Chapter 4.1

Component Tests

These tests explore new features introduced by this thesis, which existing simulations do not typically address. The intention is to illustrate how these components affect agent behaviour in different architectural conditions. This chapter goes through the behaviour of three key components:

- Agent wayfinding using perception
- Change in visibility using field of view
- Influence of airport priorities in non-processing domains

Wayfinding Test

This test demonstrates the difference between A* direct navigation, like FlexSim simulations, and A* perception navigation. The hypothesis is that agents navigating with perception do not always take the shortest path like direct navigation, because they can be influenced by information from the environment. This test is motivated by the experiments from Raubal in their research of perceptive wayfinding in an unfamiliar environment. Raubal shows that people who have no previous knowledge of a space must use their surroundings to inform their decisions, which does not always follow the shortest path. ^[1]

The scenario simulates passengers trying to find their gate by navigating using the information provided by wayfinding. The layout of this test is a square room that is 40 x 40 m, with a T-junction in the middle of the layout (Fig.4.1.a). Agents enter the simulation at an entrance before the T-junction and exit at one of two gates, A or B, located on the other side. The T-junction is designed to block the agents' view so they cannot see the gates (Fig.4.1.b). There is also a restricted area, marked by an amber rectangle, which emulates high-cost areas in FlexSim, that agents should avoid.

1. Raubal, Martin. "Agent-Based Simulation of Human Wayfinding: A Perceptual Model for Unfamiliar Buildings". (PhD diss., Vienna University of Technology, (October 2001): 17-29.

Wayfinding Test

Purpose

Determine if the environment can influence agents using perception navigation to take a longer path than agents using direction navigation

Conditions

- 40 x 40 m space with a T-junction that has a short and long corridor
- Entrance is before T-junction and two Gates A and B are on other side
- Wayfinding sign at T-junction points left for Gate A and right for Gate B
- High-cost area (5 x 12 m) placed at end of shorter corridor
- Test 1: Agents walk to Gate B through T-junction using direct navigation
- Test 2: Agents walk to Gate B through T-junction using perception navigation

Result:

● Agents with perception took a longer path after reading sign

Floor Plan

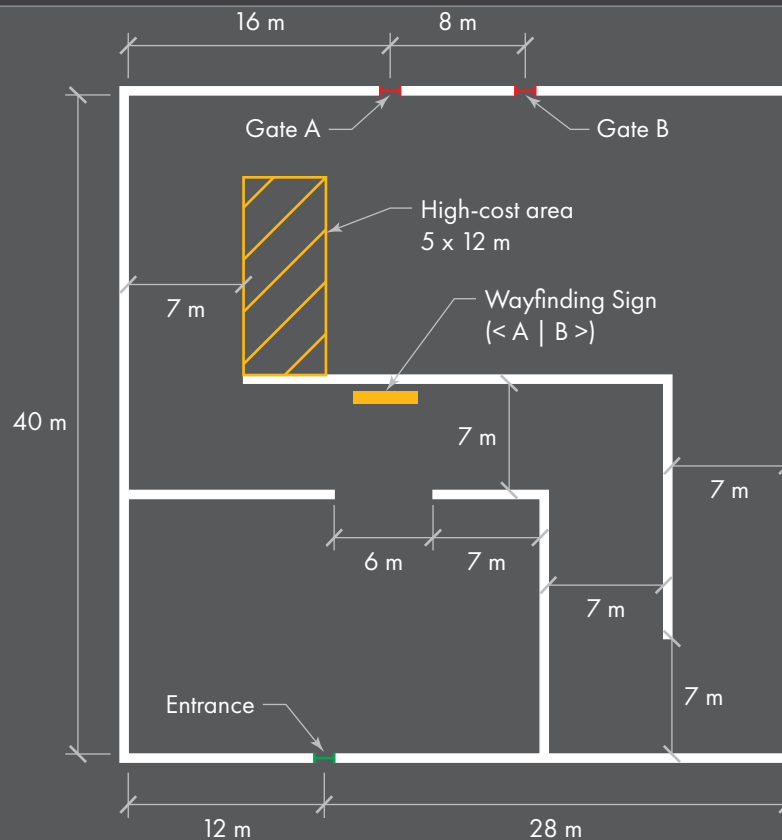


Figure 4.1.a: Setup and conditions for the wayfinding test.

Agents must pass through the T-junction to get to their gates. There is a wayfinding sign at the T-junction which points to the left for Gate A and to the right for Gate B. The left path of the T-junction is a shorter distance to both gates than the right path, which doubles-back on itself. The hypothesis is, agents with direct navigation will always follow the shorter path to the left, regardless of which gate they are assigned and what they observe. Whereas agents with perception navigation will follow the sign for Gate A to the left and for Gate B to the right. Essentially, if agents are assigned Gate B, then agents with perception navigation are more likely to follow the sign to the right, even though it is a longer path.

The thesis demonstrates this behaviour over two experiments. The first experiment is a baseline, which involves agents searching for Gate B using direct navigation. In the second experiment, agents search for gate B using perception navigation. Agents are assigned the same walking speed of 1 m/s for both experiments.

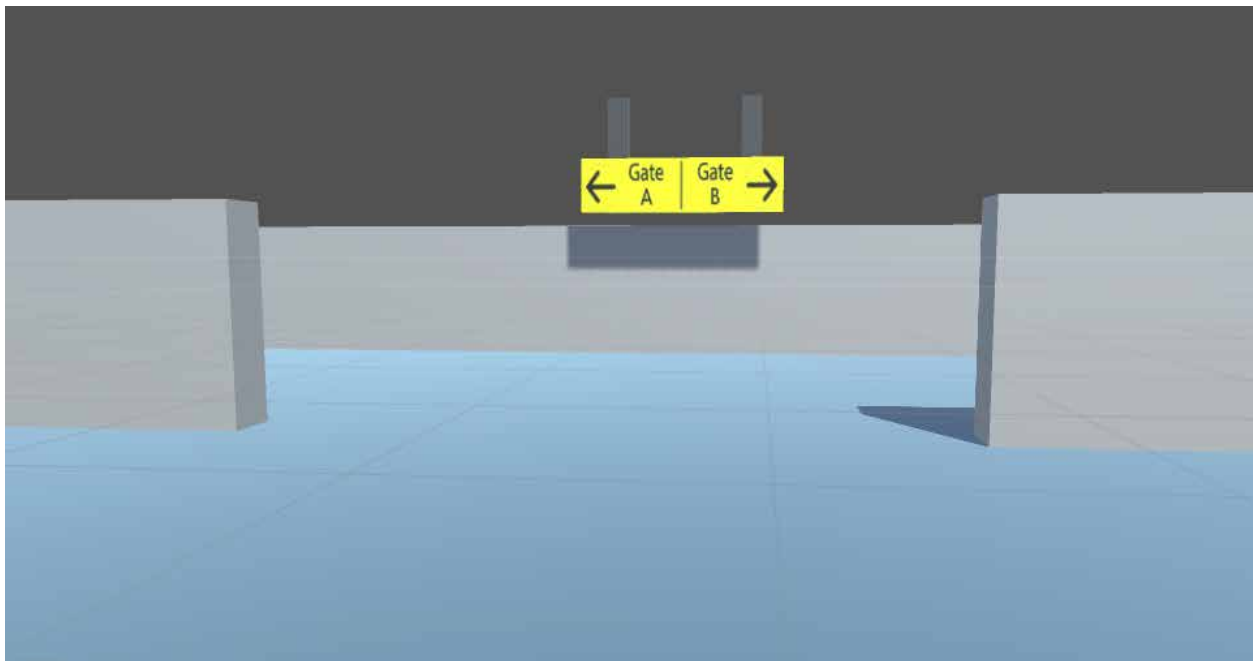


Figure 4.1.b: *Agent's view at the T-junction cannot see where their gate is, and they only have the sign to inform their decisions.*

During the first experiment, an agent is spawned at the entrance with direct navigation. The moment they appear at the entrance, the agent generates a complete path for itself all the way to Gate B. The path first traces a line from the agent's position to the T-junction. At the T-junction, the path turns down the left corridor. Then, the path wraps around the centre wall but stays outside of the restricted area. Finally, it traces a path around the top of the high-cost area and goes straight to Gate B. Once the path is established, the agent begins following the path exactly, without being influenced by the information on the sign at the T-junction (Fig.4.1.c). The agent continues like this until it reaches Gate B, where they are removed from the simulation.

During the second experiment, an agent is spawned at the entrance with perception navigation. When they appear at the entrance, the agent notices the T-junction and the sign above it, which they begin walking towards. Once in front of the sign, the agent then selects a path down the right corridor, which is the same direction the sign is pointing to for Gate B (Fig.4.1.d). The agent follows this path down the corridor. Since the corridor doubles-back on itself, the agent continues to walk down and around the wall, which eventually winds back towards the gates. Once the agent reaches the end of the corridor, Gate B becomes visible. After the agent recognizes Gate B, they walk straight towards the gate and is then removed from the simulation.

The thesis ran multiple trials of the same experiments with a population of 50 agents for both direct and perception navigation (Fig.4.1.g). For all attempts, agents with direct navigation took the left path, whereas agents with perception navigation took the right path. On average, the total distance travelled for direct agents was 61.83 *m*. The total distance travelled for perception agents was 102.18 *m*, which is over 40 *m* longer, on average. The data for these trials are list in Appendix A. Therefore, agents with perception navigation can be influenced by the environment to take a longer path than agents with direct navigation. As a result, this demonstrates agents with perception have the ability to interact with architectural features to influence their behaviour in a space.

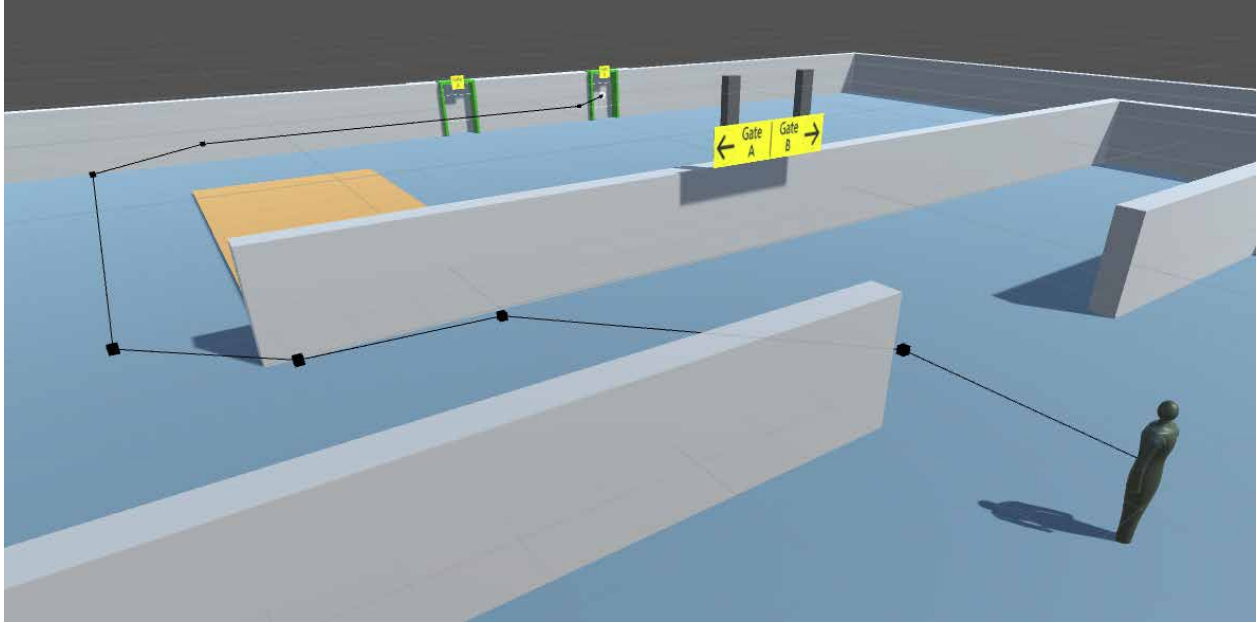


Figure 4.1.c: Agent with direct navigation goes to the left.

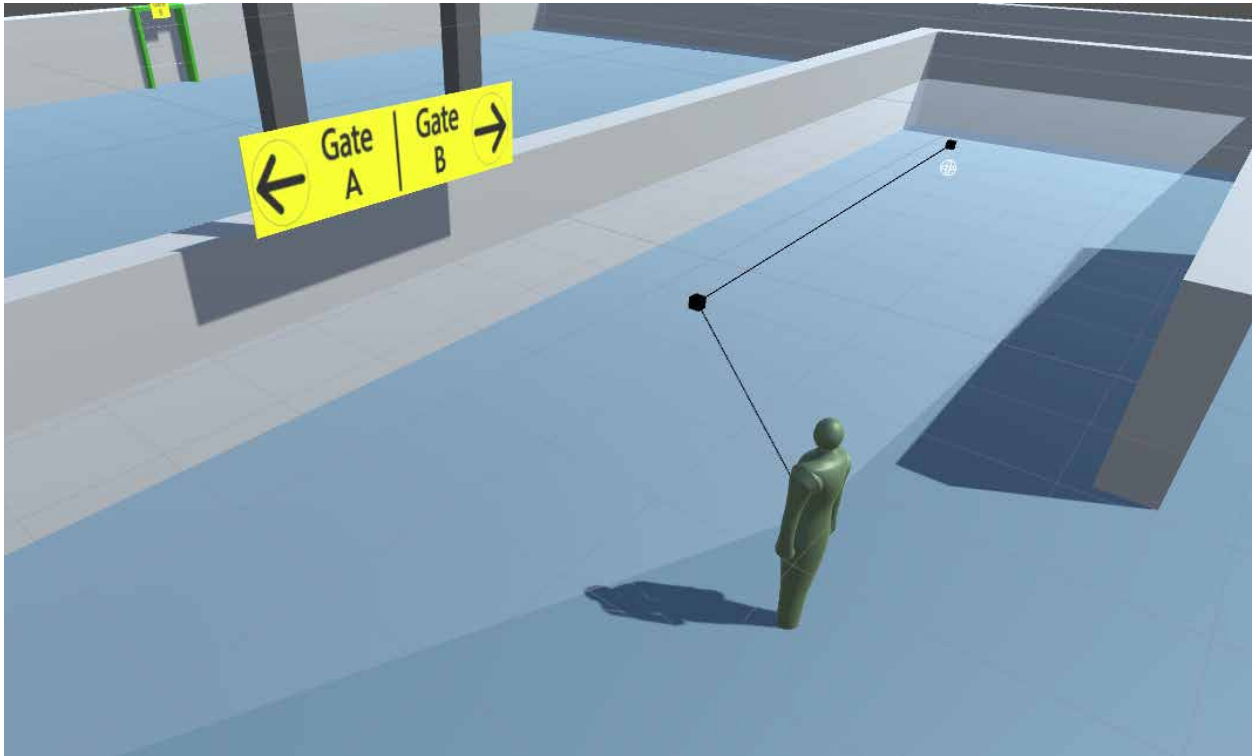


Figure 4.1.d: Agent with perception navigation follows the sign for Gate B to the right.

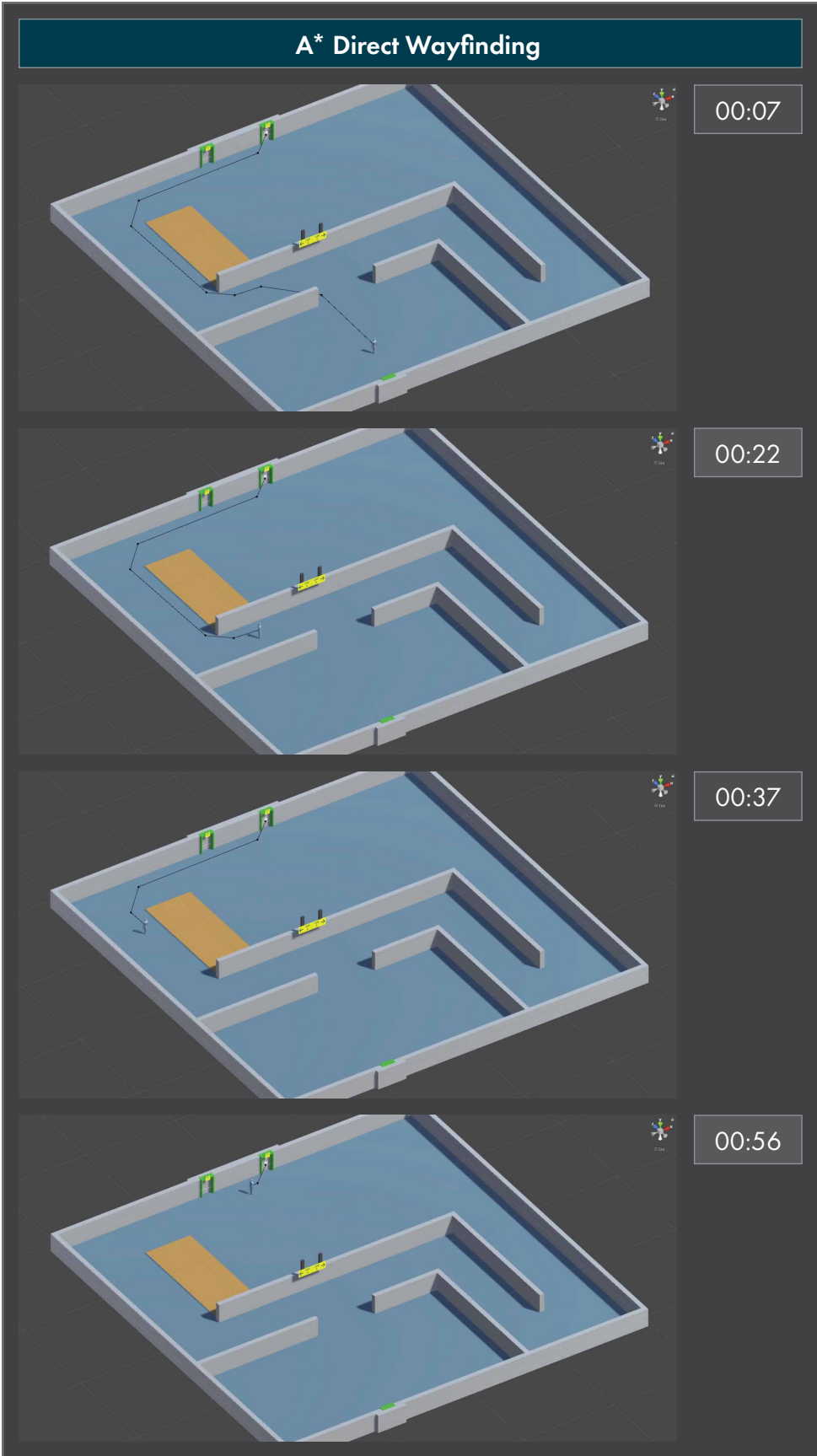


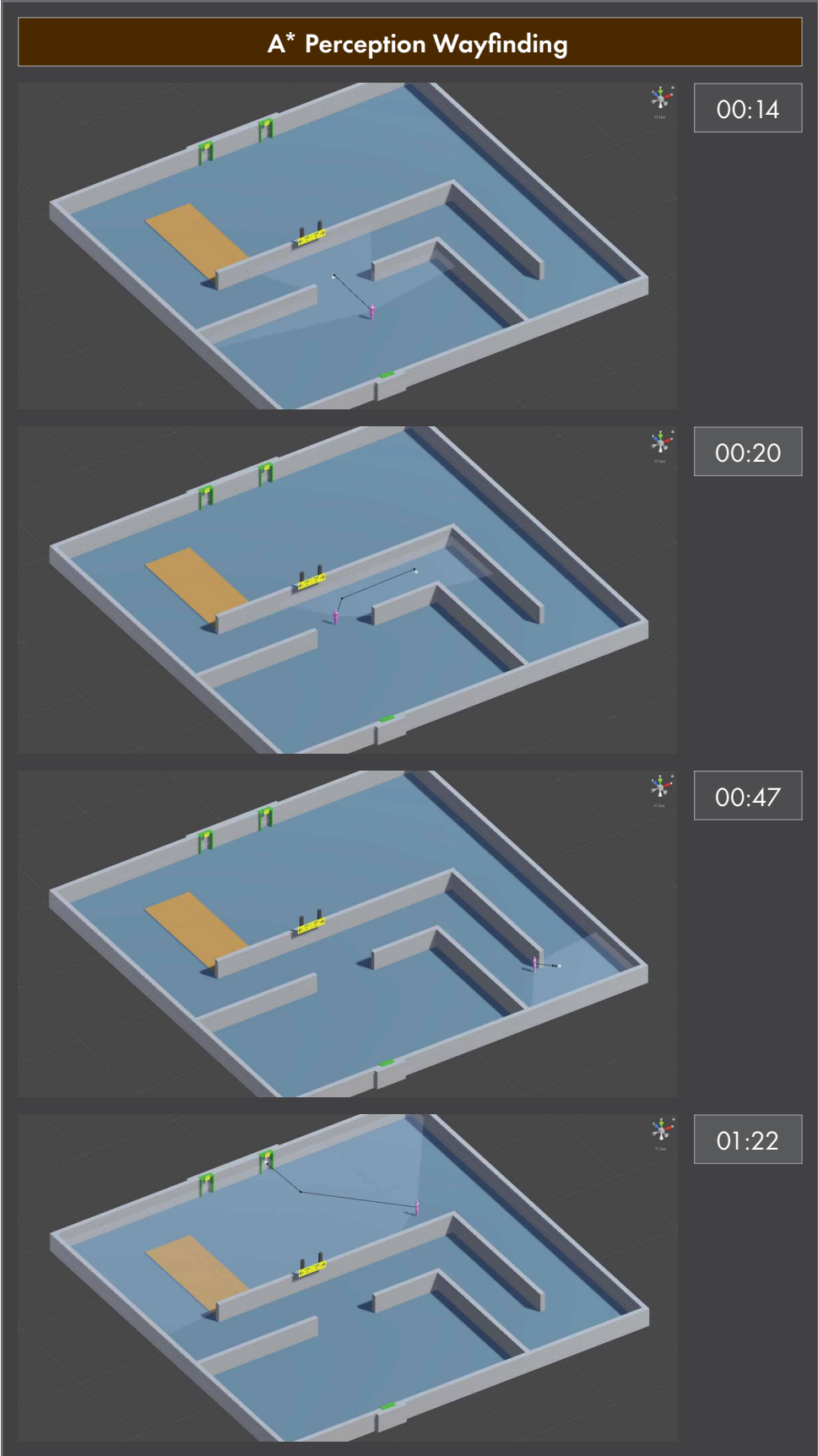
Figure 4.1.e:

Agent finds lowest cost path to Gate B.

Agent follows path to the left, indifferent of the sign.

Agent follows path around high-cost restricted area.

Agent reaches Gate B.



A* Perception Wayfinding

00:14

Figure 4.1.f:
Agent walks to view sign.

00:20

Agent follows the sign to the right.

00:47

Agent navigates around walls.

01:22

Agent sees Gate B and walks towards it.

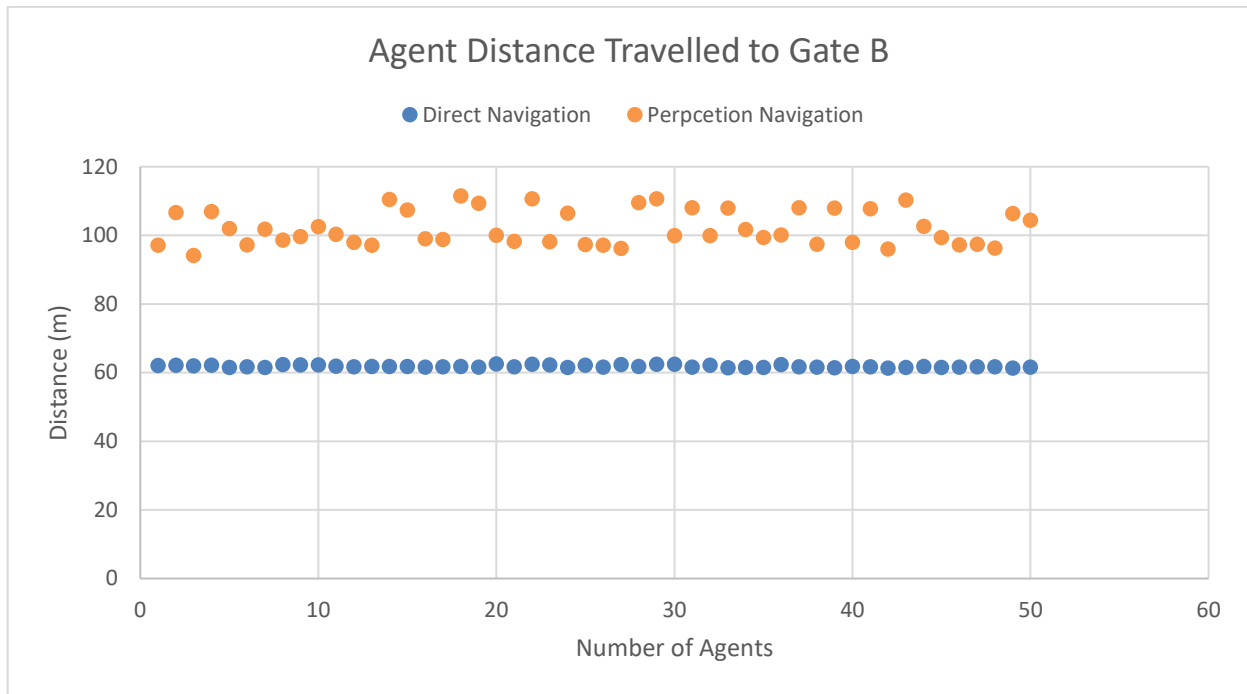


Figure 4.1.g: Comparing the distance agents travelled to Gate B using direct and perception navigation.

Visibility Test

This test demonstrates how isovist geometry can quantify the relative visibility differences between smaller and larger architectural spaces. The hypothesis is that moving from a smaller space into a larger one will increase an agent's visibility. Likewise moving from a larger space into a smaller one will reduce an agent's visibility. Since the isovist geometry is a representation of an agent's field of view, then the isovist should also reflect these differences. The thesis expects to quantify an agent's visibility based on the change of the physical area of the isovist.

This thesis also compares relative areas using a custom metric called the *Field of View (FOV) ratio*. This normalizes the total visible area relative to an agent's maximum observed area over time, where the maximum area is given a value of 1.0. For example, if the maximum area an agent observed over time was 100 m^2 and the current visible area where the agent is standing is 60 m^2 , then their FOV ratio for that location would be 0.6. Additionally, if this agent later observes an area of 150 m^2 , then this would be given a ratio of 1.0, the 100 m^2 would be rewritten as 0.67, and the 60 m^2 would become 0.4.

The layout of this test is designed as a corridor that transitions between a wide section and a narrow section (Fig.1.h). It is a $42 \times 24\text{ m}$ U-shaped space, with a 20 m wide leg and a 2 m narrow leg. There is a gate located at the end of each leg that agents can enter and exit from, which are called the Wide Gate and Narrow Gate, respectively. There is also a transition section between the wide and narrow corridors, which steps the width from 20 m to 9 m , and 4 m , before dropping to 2 m . This transition was added because agents in initial tests had difficulty finding the narrow corridor. It also provides a gradual change in visibility instead of an abrupt change.

Visibility Test

Purpose

Demonstrate that narrow spaces result in low visibility and wide spaces result in high visibility, relatively, based on the agent's field of view.

Conditions

- 42 x 24 m U-shaped space, with a 20 m wide leg and a 2 m narrow leg
- 10 m long transition corridors that are 9 m and 4 m wide
- Test 1 (to Narrow): Agents walk from wide gate to narrow gate
- Test 2 (to Wide): Agents walk from narrow gate to wide gate

Result:

● Visibility value was higher towards wide space than to narrow space

Floor Plan

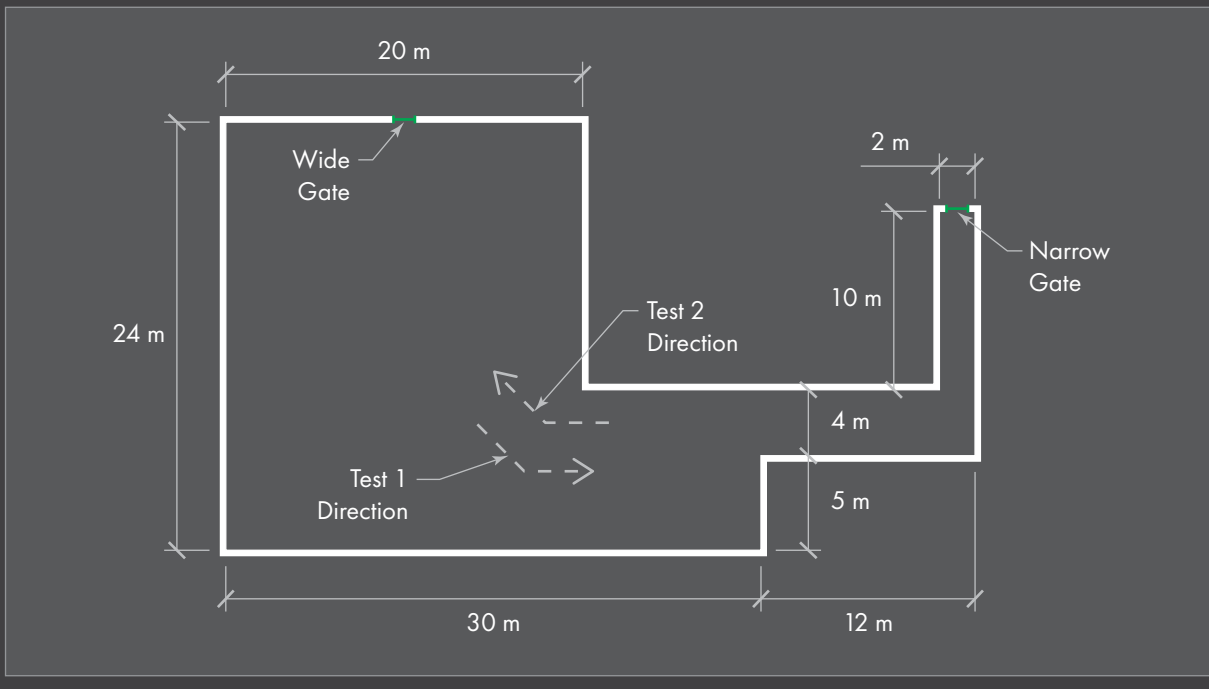


Figure 4.1.h: Setup and conditions for the visibility test.

This test is divided into two experiments. The first experiment has agents walking from the Wide Gate to the Narrow Gate (4.1.k). The second experiment has agents walking in the opposite direction from the Narrow Gate to the Wide Gate (4.1.l). In both experiments, agents are using perception navigation and are assigned a walking speed of 1 *m/s*. The agent's field of view is displayed as the simulation is running. The values of the agent's FOV area, FOV ratio, and average visibility are recorded over time during their journey.

The first experiment spawns an agent at the Wide Gate. The agent walks down the 20 *m* towards the corner of the 9 *m* transition space. As they get closer to the corner, their FOV area slowly becomes smaller. When they reach the corner, the agent is able to see down the transition corridor, which increases their field of view. After rounding the corner, the agent continues to walk to the end of the transition space, which narrows down to 4 *m* wide. When the agent reaches the corner of the 2 *m* narrow corridor, the agent's field of view has become small. As the agent looks around this corner, they can see the Narrow Gate at the end of the corridor, which momentarily increases their field of view (Fig.4.1.i). Once the agent recognizes their destination, they walk straight to the gate, where they are then removed from the simulation.

The second experiment spawns an agent at the Narrow Gate. The agent walks down the 2 *m* corridor to the 4 *m* transition. Their FOV area slowly becomes smaller as they reach the first corner. When the agent turns the corner, their field of view increases as they suddenly have a view all the way down the transition section. The agent continues walking to the end of the transition corridor. As they approach the corner of the wide corridor, their field of view begins to open out into the 20 *m* wide space. When the agent reaches the corner of the wide corridor, they suddenly have view of the Wide Gate (Fig.4.1.j). At this point, the agent recognizes their destination, and walks straight to the Wide Gate, which slowly decreases the size of their field of view as they get closer. Once the agent reaches the Wide Gate, they are removed from the simulation.

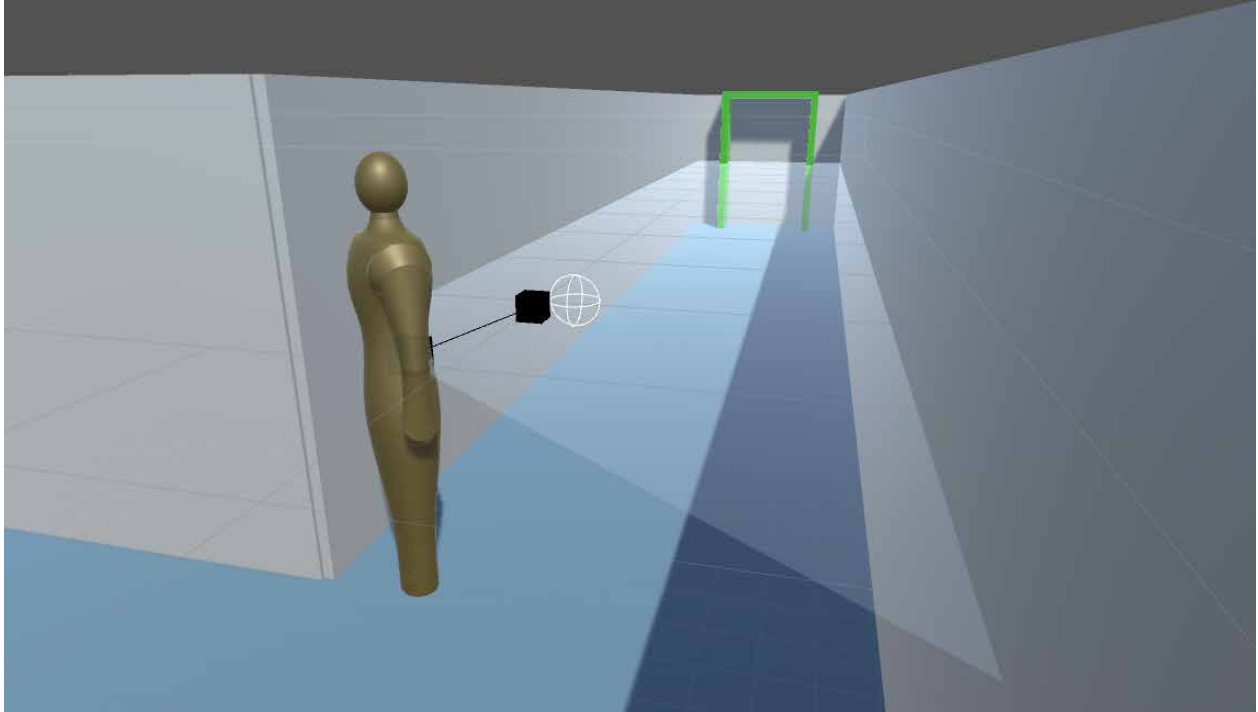


Figure 4.1.i: *Agent's view when they see the Narrow Gate.*

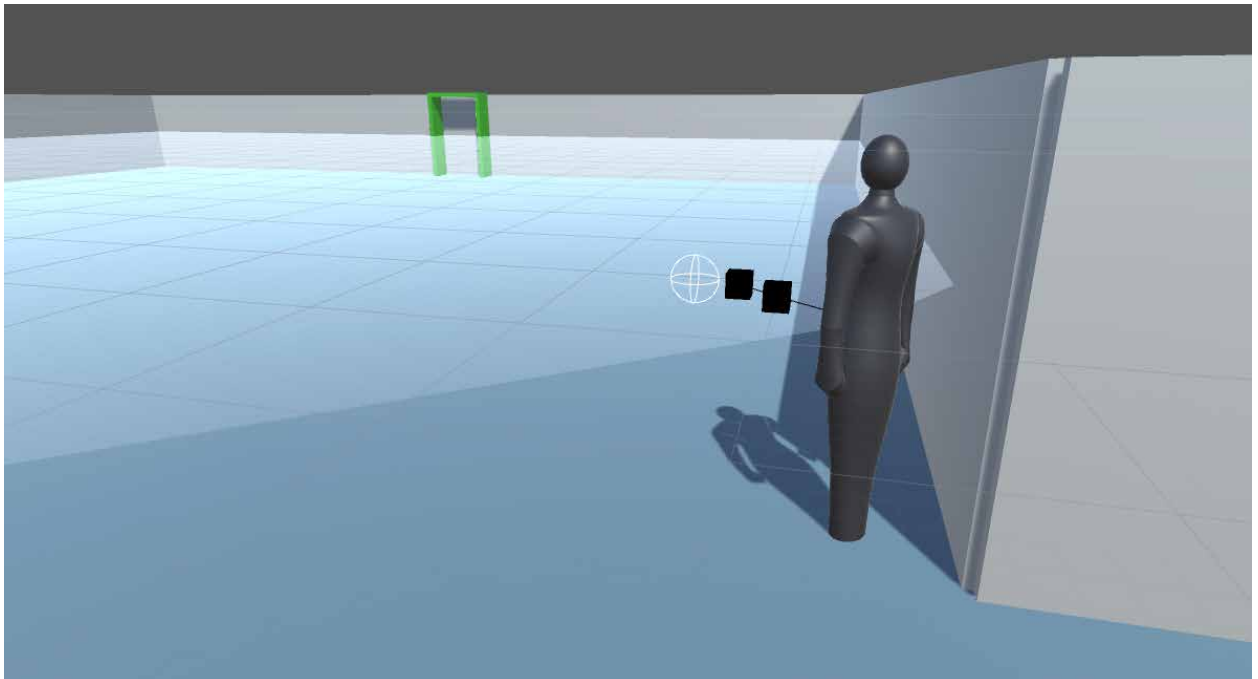


Figure 4.1.j: *Agent's view when they see the Wide Gate.*

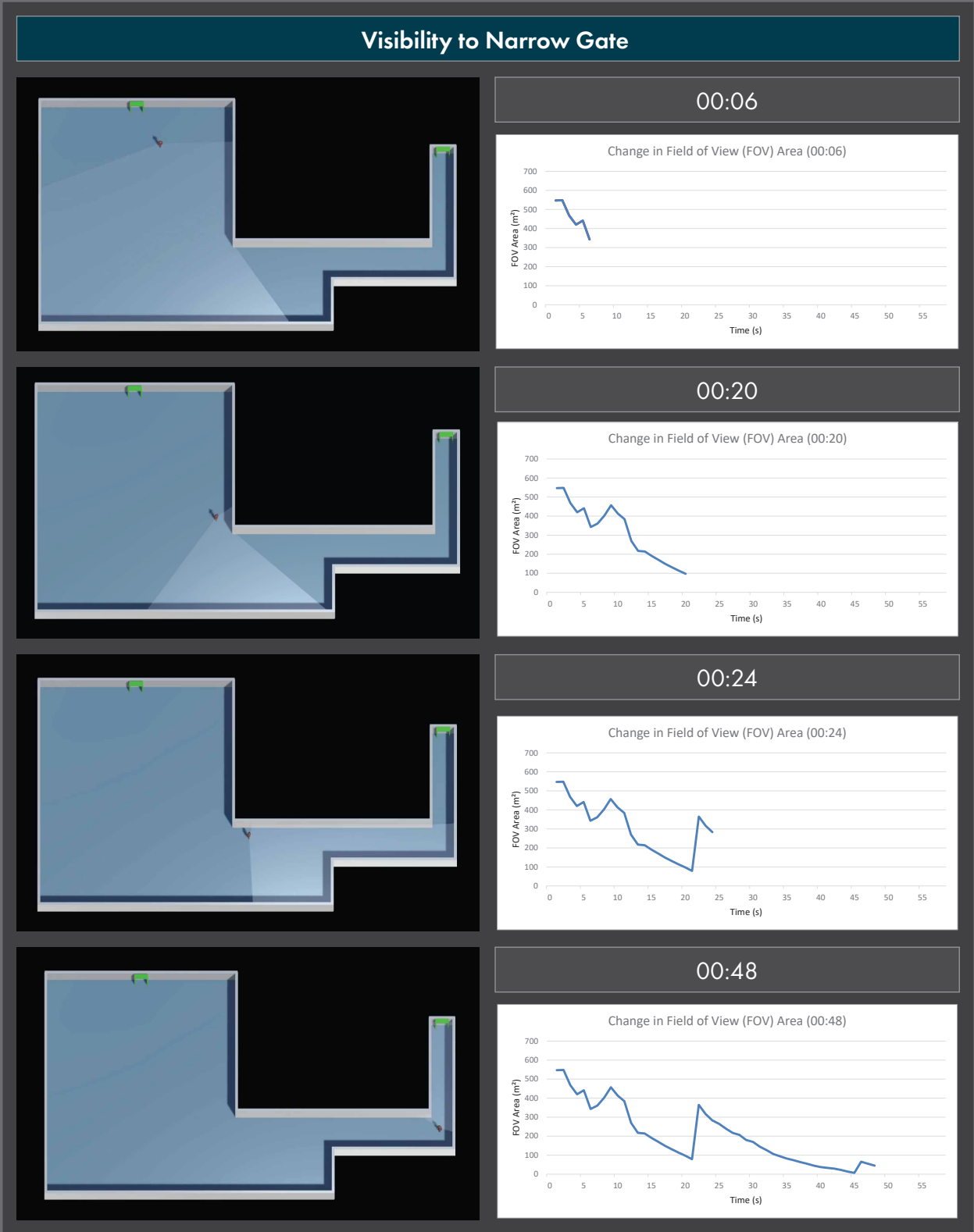


Figure 4.1.k: Screen captures as agent walks to Narrow Gate, showing the change in FOV area.

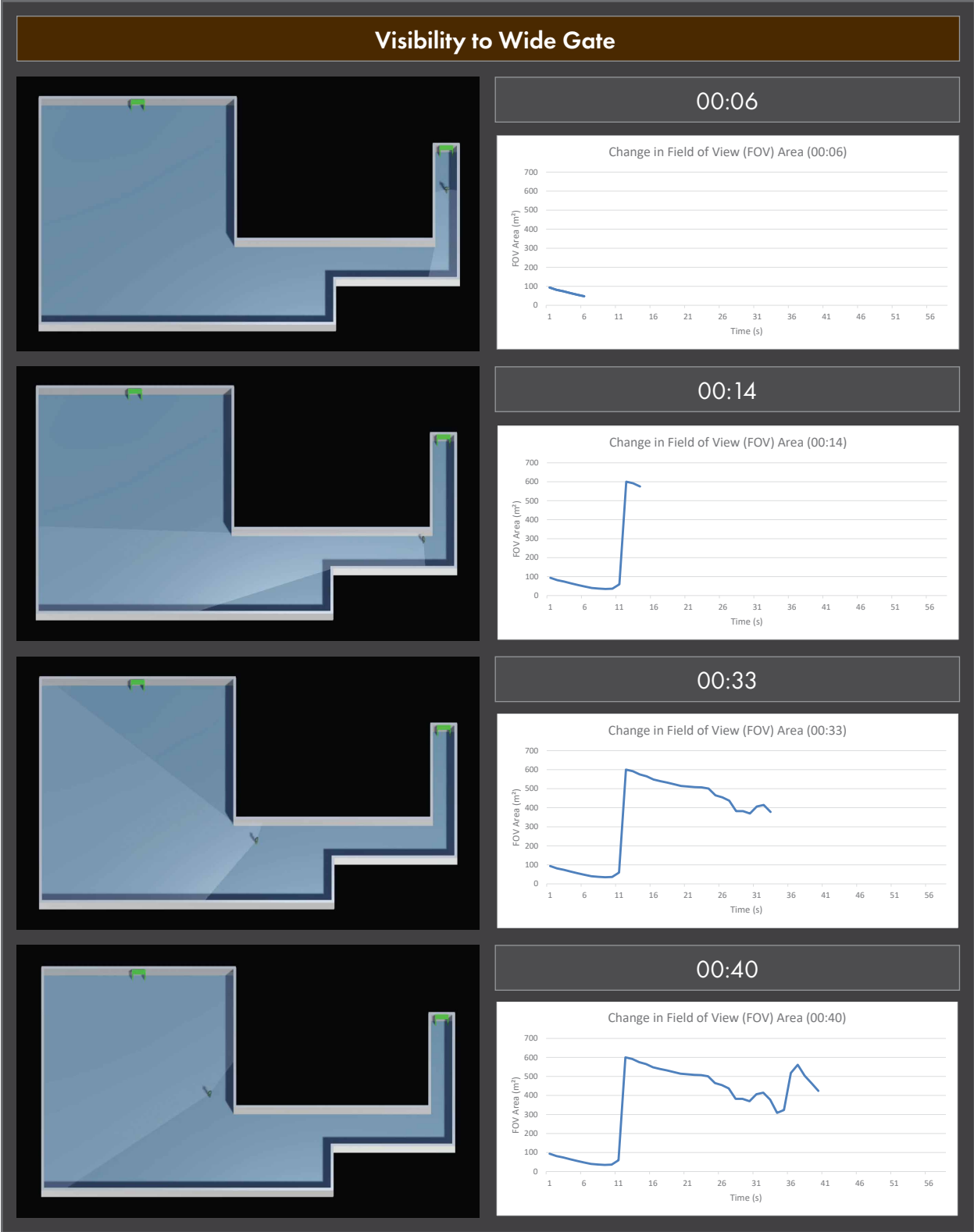


Figure 4.1.1: Screen captures as agent walks to Wide Gate, showing the change in FOV area.

The outputs of agents' visibility in these experiments are compared in Fig.4.1.m-o. When the agent was going to the narrow gate, their initial area was around $550 m^2$, whereas the initial area of the agent going to the wide gate was only around $100 m^2$ (Fig.4.1.m).

Both the FOV area and FOV ratio for the agent going to the Narrow Gate gradually drops towards zero. There is a spike in visibility around 22 seconds into the journey, which is a result of the agent seeing around a corner, but it does not get higher than the initial value (Fig.4.1.n).

The FOV area and FOV ratio for the agent going to the Wide Gate has a big jump up around 10 seconds into the journey. This is due to the agent seeing down the transition space, which was one of the highest visibility values. This slowly drops until the agent turns the second corner to the Wide Gate, which sees another spike. Then the ratio drops again as the agent walks towards the gate, but still maintains a higher value than the agent in the narrow corridor.

As shown in the average visibility graph, the agent starting at the Narrow Gate had a lower visibility than the agent starting at the Wide Gate (Fig.4.1.o). But over the transition space, the agent going to the Wide Gate saw higher visibility over time than the other agent.

The thesis ran these experiments for multiple trials with a population of 50 agents in both directions. The outputs of these trials are shown in Fig.4.1.p-r, which illustrate the differences. The key value to consider is the FOV ratio when agents discovered their gate (Fig.4.1.q). Agents going to the Wide Gate had an average ratio of 0.88 when they saw it, whereas agents going to the Narrow Gate only had an average ratio of 0.10. This trend is similar in the average visibility, with 0.70 for agents going to the Wide Gate and only 0.42 for agents going to the Narrow Gate (Fig.4.1.r).

The thesis notes that the maximum observed areas are similar for both directions, which is around $600 m^2$. Although, some agents walking towards the Narrow Gate saw a maximum area as high as $1200 m^2$ (Fig.4.1.p). This appears to be the result of some agents who doubled-back on themselves trying to find the narrow gate, which resulted in agents seeing down both the wide corridor and transition corridor at the same time. However, this has no significant impact, since $1200 m^2$ is still much greater than the $100 m^2$ observed in the narrow corridor. Data from these trials can be seen in Appendix A.

Therefore, both the FOV ratios and the average visibilities for agents going to the Wide Gate are higher, on average, than agents going to the Narrow Gate. This means the agent's field of view correctly recognizes the difference in visibility between wide and narrow architectural spaces.

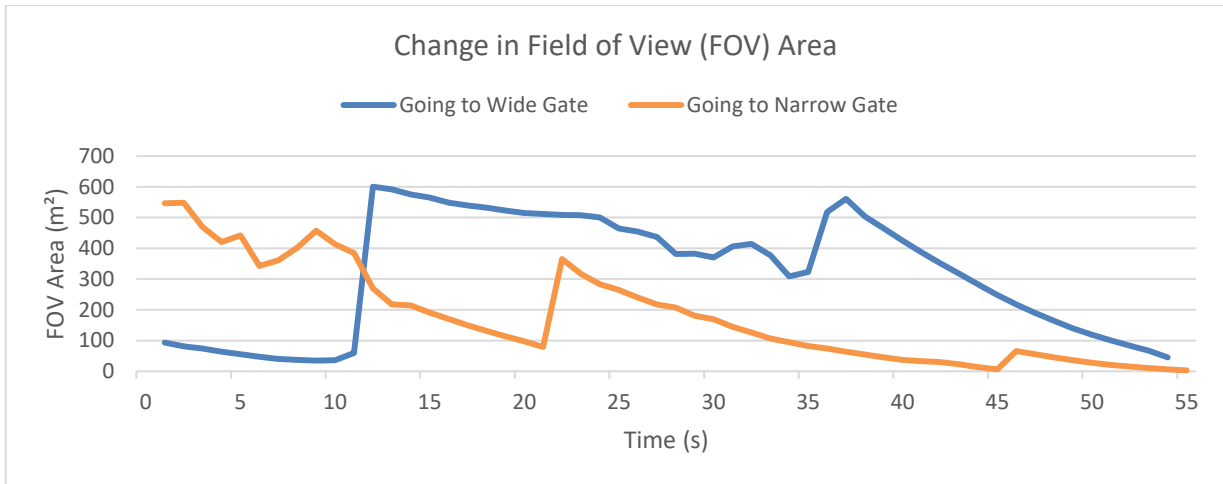


Figure 4.1.m: FOV area gets larger towards wide gate and smaller towards narrow gate.

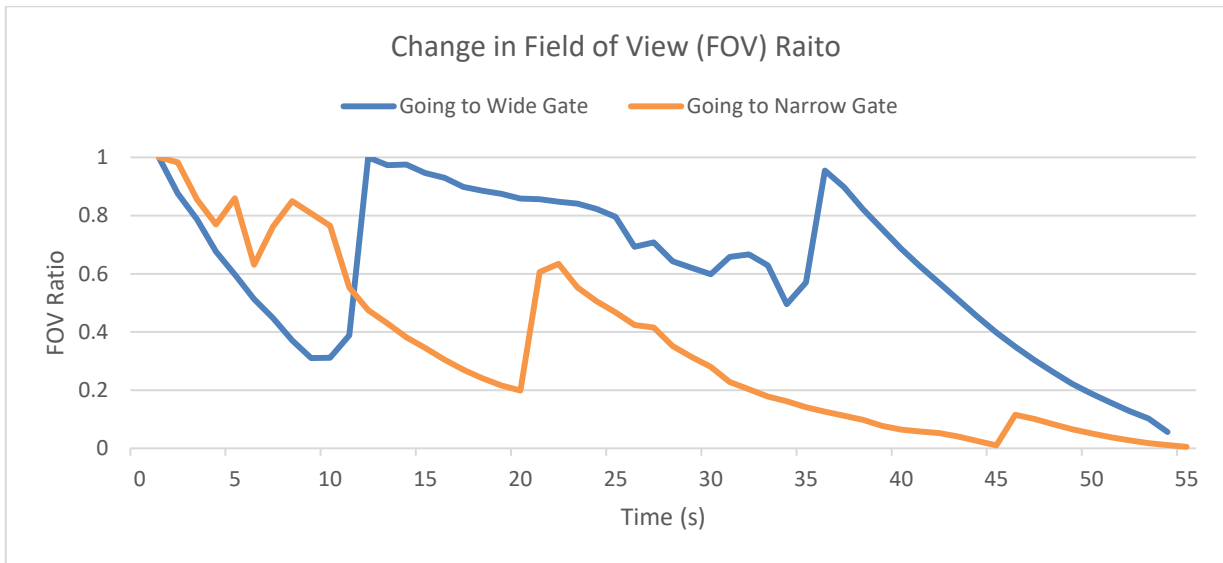


Figure 4.1.n: FOV ratio trends above 0.5 towards wide gate, and below 0.5 towards narrow gate.

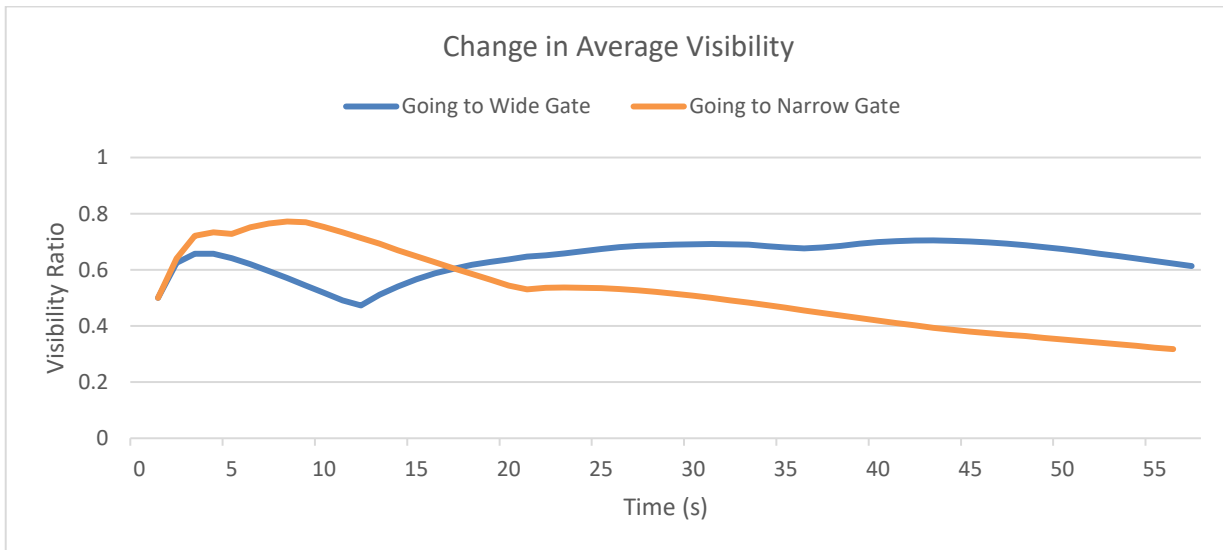


Figure 4.1.o: Over time, average visibility is greater towards wide gate than narrow gate.

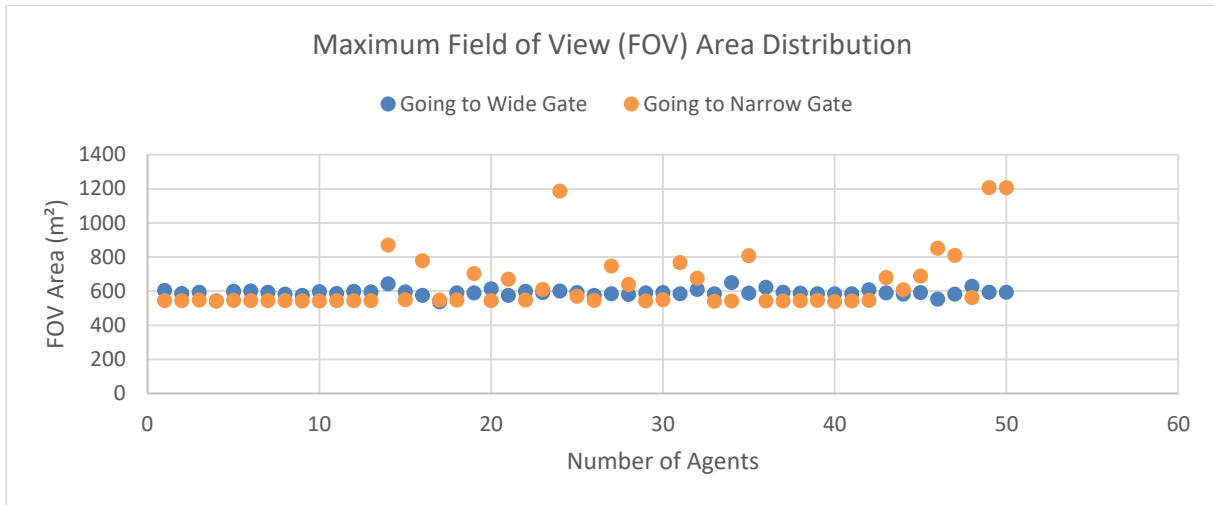


Figure 4.1.p: Max observed FOV area is around 600 m² for both directions, unless agents doubled-back.

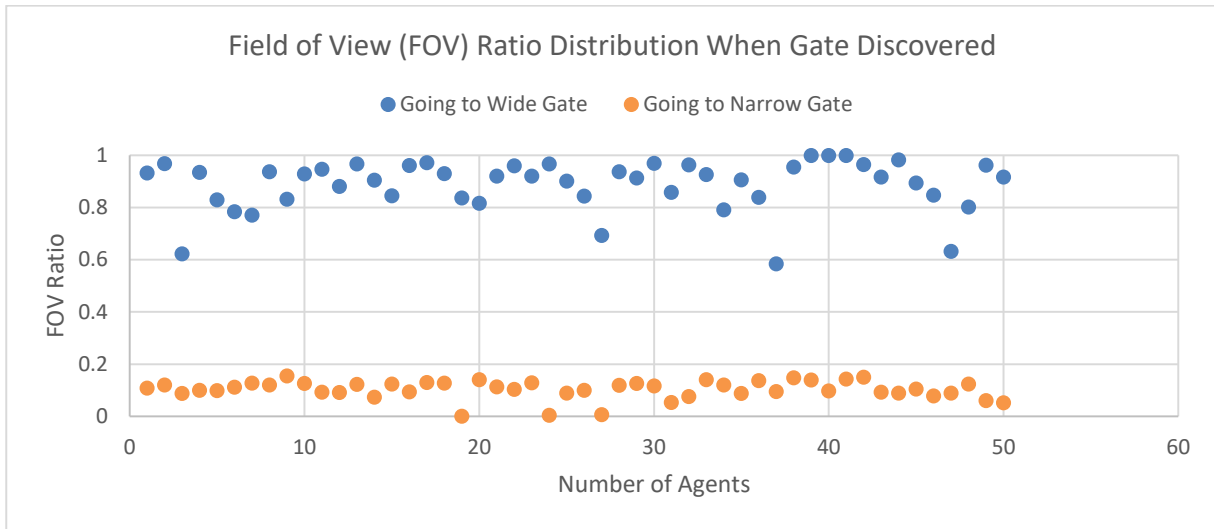


Figure 4.1.q: FOV ratio is higher when agents see the wide gate than the narrow gate.

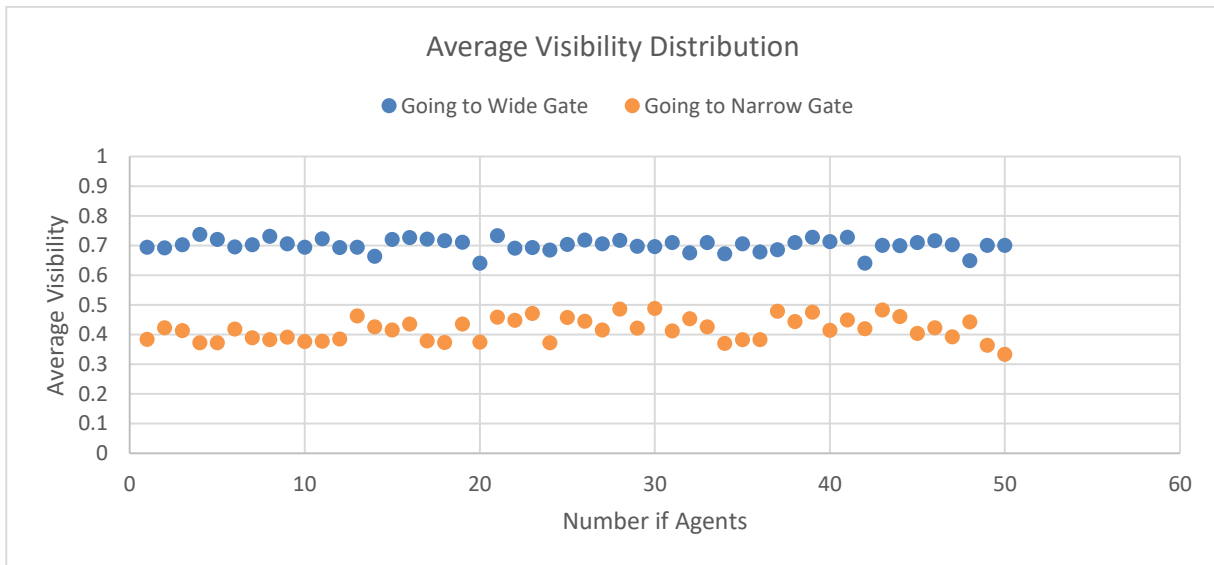


Figure 4.1.r: Average visibility is higher going to the wide gate than the narrow gate.

Non-processing Priorities Test

This test demonstrates how priorities can influence agent behaviour in non-processing airport domains. The hypothesis is that agents who are assigned a high priority for a non-processing domain, like food, are more likely to interact with food and retail areas in an airport than agents who are assigned a low priority.

The scenario replicates conditions of a typical North American terminal. Seating areas are organized linearly along one side of the facility, and food and retail spaces are placed along the opposite side, with circulation running through the middle (Fig.4.1.s).

The layout for the test is a 30 x 30 *m* square space. The west side of the space is designed as a gate holdroom, or waiting area, and the east side is dedicated to food stalls. There is one entrance located in middle of the south wall, and one gate exit on the west wall of the holdroom. The holdroom has three rows of 12 seats for a total of 36 seats. In the food area, there are three equally spaced stalls that each have one service counter and a short queue stanchion. To differentiate the stalls in the model, they are labelled as a pizza shop, a café, and a restaurant bar. However, the food stalls are functionally the same for the test.

The test involves a population of 50 agents, whose goal is to board a flight. Their departure time is set for 2 minutes into the simulation. Before the departure time, agents are free to interact with the environment. In this test, agents can do one of two activities, either wait in the holdroom or get something to eat. When agents enter the simulation, they are randomly assigned a food priority on a scale from 1 to 9. Agents that have a food priority of 5 or higher are expected to get something to eat. Otherwise, agents with a food priority less than 5 should wait in the holdroom, until it is their departure time. For this test, agents are using perception navigation and assigned a walking speed of 1 *m/s*. Their proximity detection is disabled for this test, so agents may walk through each other.

Non-Processing Priorities Test

Purpose

Show that agents with higher priority for a non-processing domain (food) follow different behaviour than agents with a lower priority

Conditions

- 30 x 30 m room, with 3 food stalls, and 3 rows of seating in waiting area
- Agents randomly assigned food priority from 1 to 9.
- Agents with food priority greater or equal to 5 should get food
- Otherwise, agents wait by seating area until a set departure time

Result:

● All agents with a high food priority got food before departure

Floor Plan

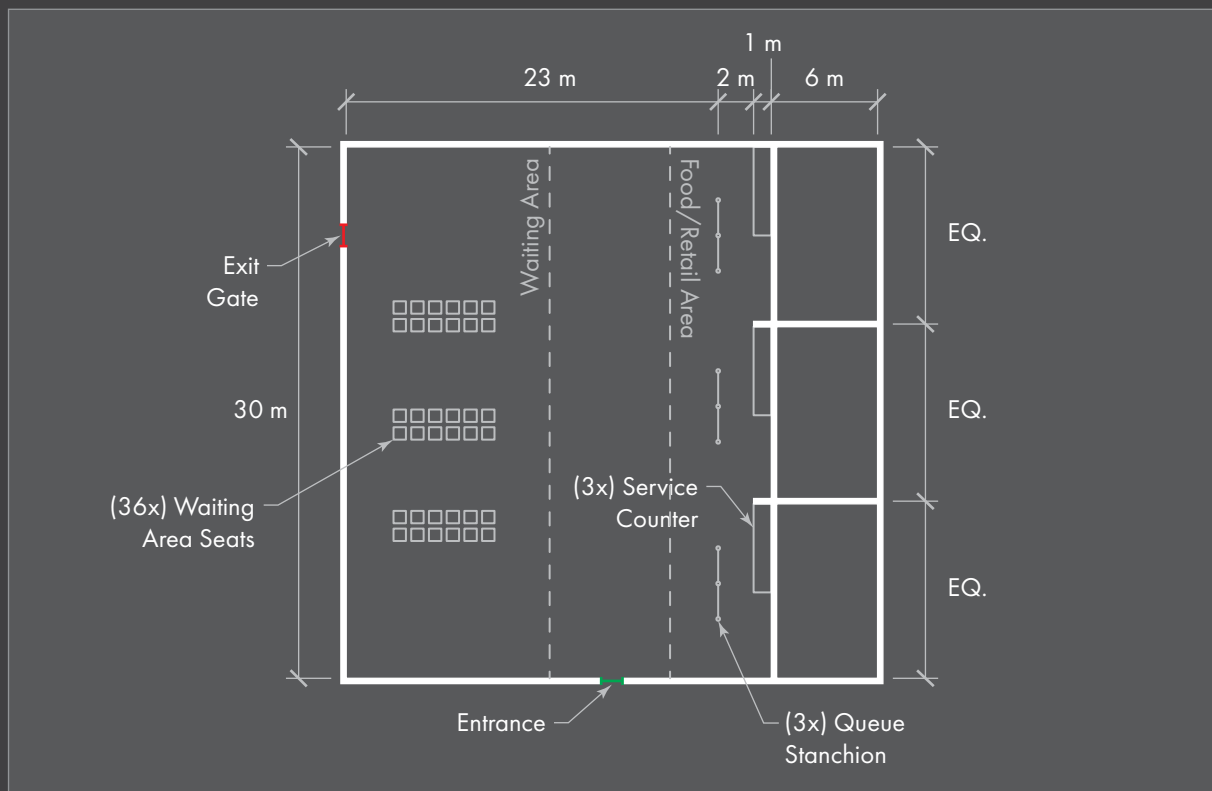


Figure 4.1.s: Setup and conditions for the non-processing priority test.

When the simulation starts, agents randomly enter the space every few seconds (Fig.4.1.v). Agents going to the holdroom are coloured in red and agents going to the food area are coloured in light green. There appears to be a similar number of agents going to each area over time. Agents in the holdroom are waiting, so they walk over to a random seat and come to rest (Fig.4.1.t).

Meanwhile, agents in the food area are looking for a random food stall. If an agent selects a stall, they get in line behind the queue stanchion, which changes their colour state to yellow (Fig.4.1.u). Then, agents walk up to the service counter to order some food, which changes their colour state to orange. The time an agent spends at a counter is based on a random service time, which is about 10 seconds. Once they are done getting food, the agent turns light blue and moves away from the counter, which completes their food interaction.

After agents leave the food area, they check back to the gate area. If it is not their departure time, then they walk over to the holdroom and join the other agents already waiting there, which changes their colour state to red. As soon as the simulation time reaches the departure time, all agents in the holdroom turn a dark green colour, walk over to the exit gate, and leave the simulation. After all agents have exited, the test is complete.

The thesis ran multiple trials of this test with a population of 50 agents under similar conditions. The results of one of these tests are illustrated in Fig.4.1.w. The graph lists each agent's food priority during the simulation and indicates which agents *got food* and which agents *did not get food*. As expected, all the agents that got food have a priority of 5 or higher. This verifies that agents who were assigned a high food priority followed the indented behaviour. This also confirms that changing an agent's priority of a non-processing domain can influence a passenger's behaviour in a simulated environment. Although the priority threshold was set to 5 for this test, this can be changed to account for different domains or passenger types.

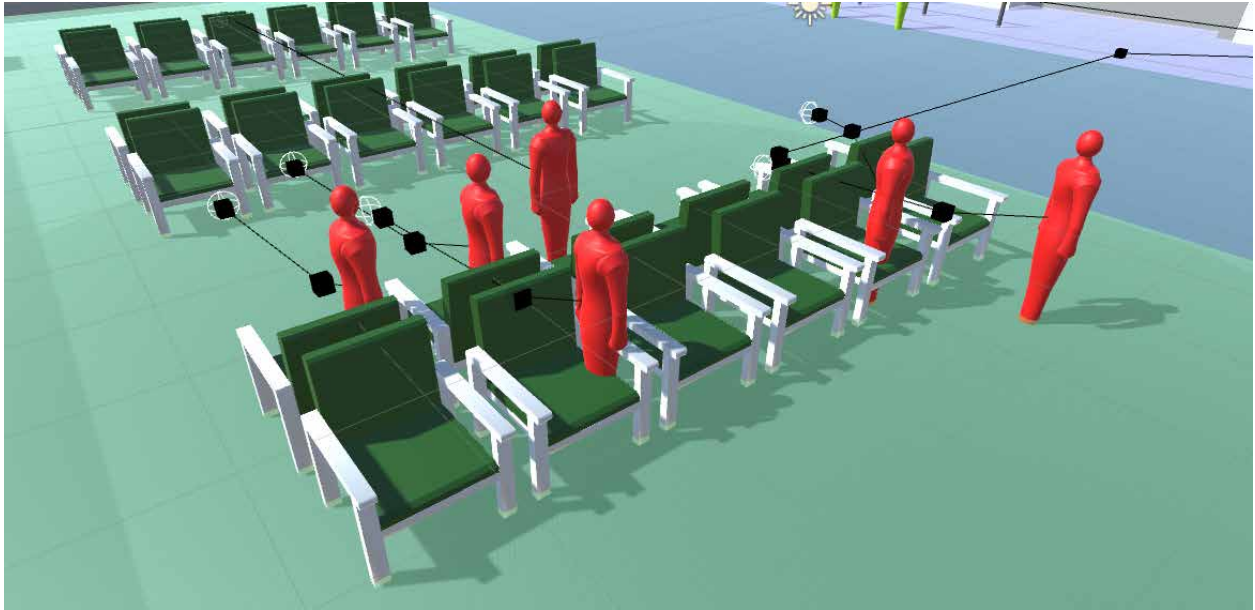


Figure 4.1.t: Agents with low food priorities waiting in the gate seating area (red).

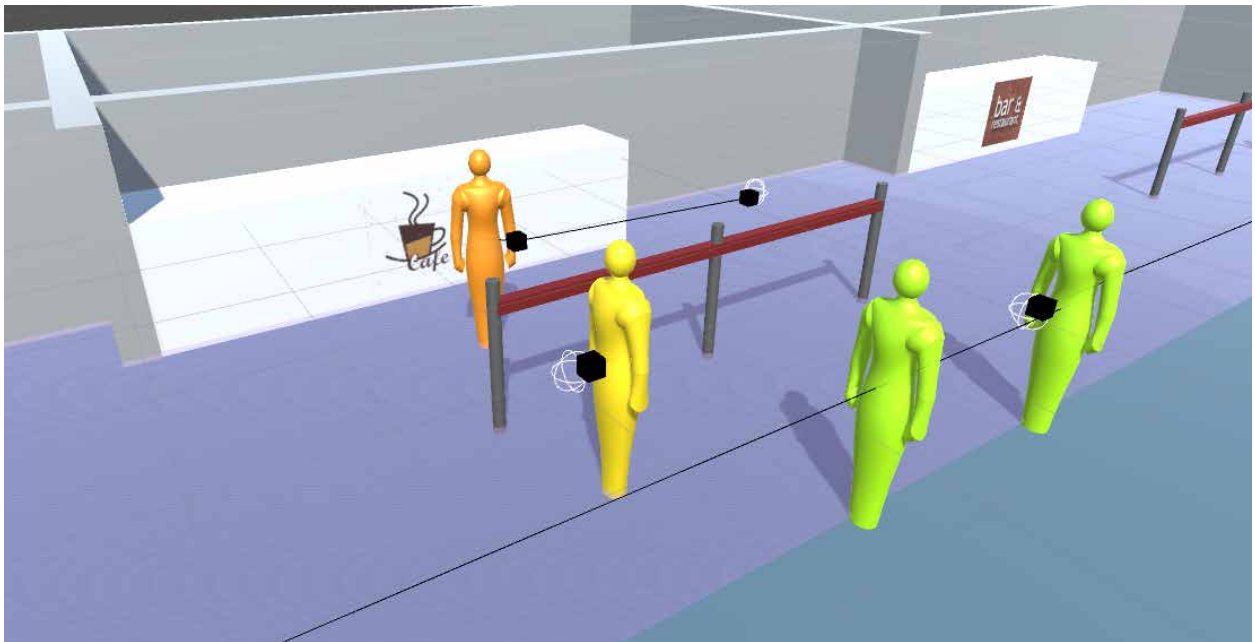
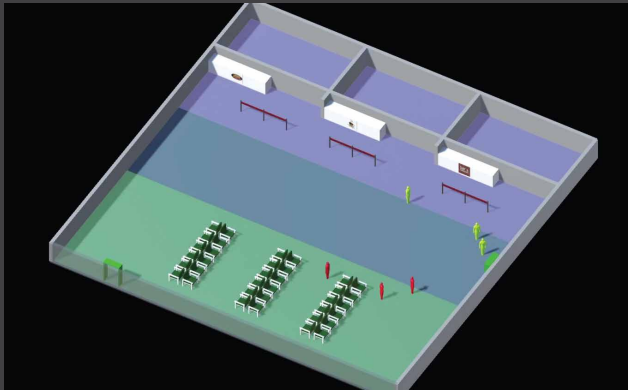


Figure 4.1.u: Agents with high food priorities getting food at the cafe (light green: going to food area, yellow: in line, orange: at counter).

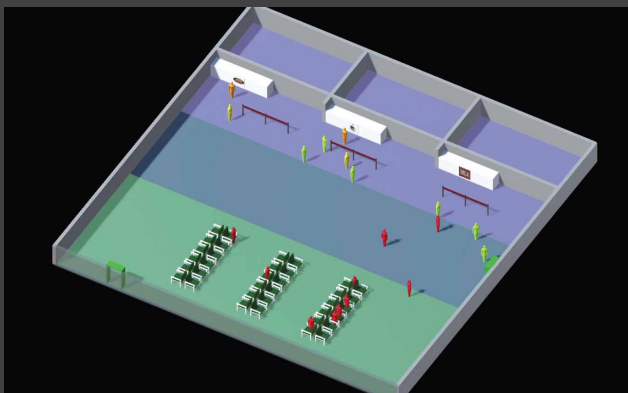
Agents Prioritizing a Non-Processing Domain



00:13

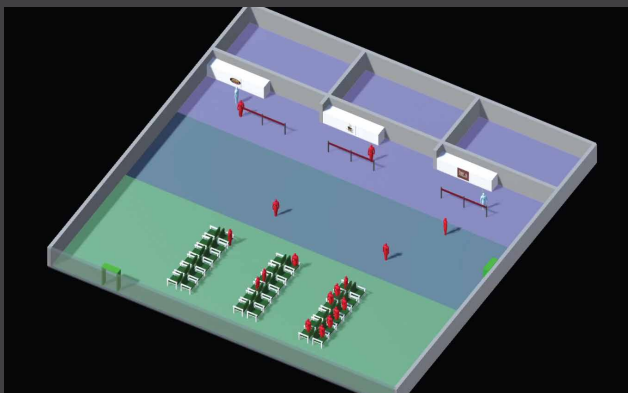
Figure 4.1.v:

Agents either go to food area (light green agents) or waiting area (red agents).



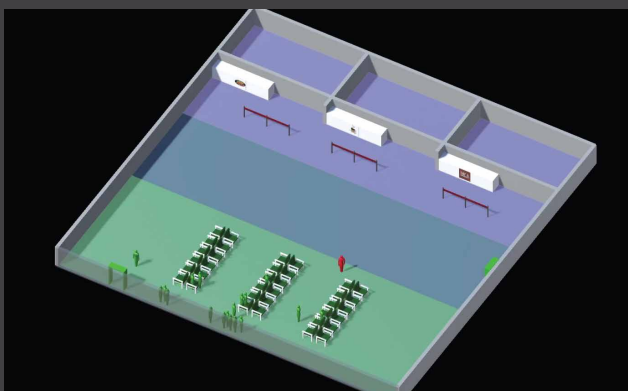
00:45

Agents in food area purchase something at the counters. Agents in seating area wait until departure time.



01:14

Agents finished in the food area, come back to the seating area to wait (red).



01:35

After the scheduled departure time (1:30), agents (dark green) go through gate.

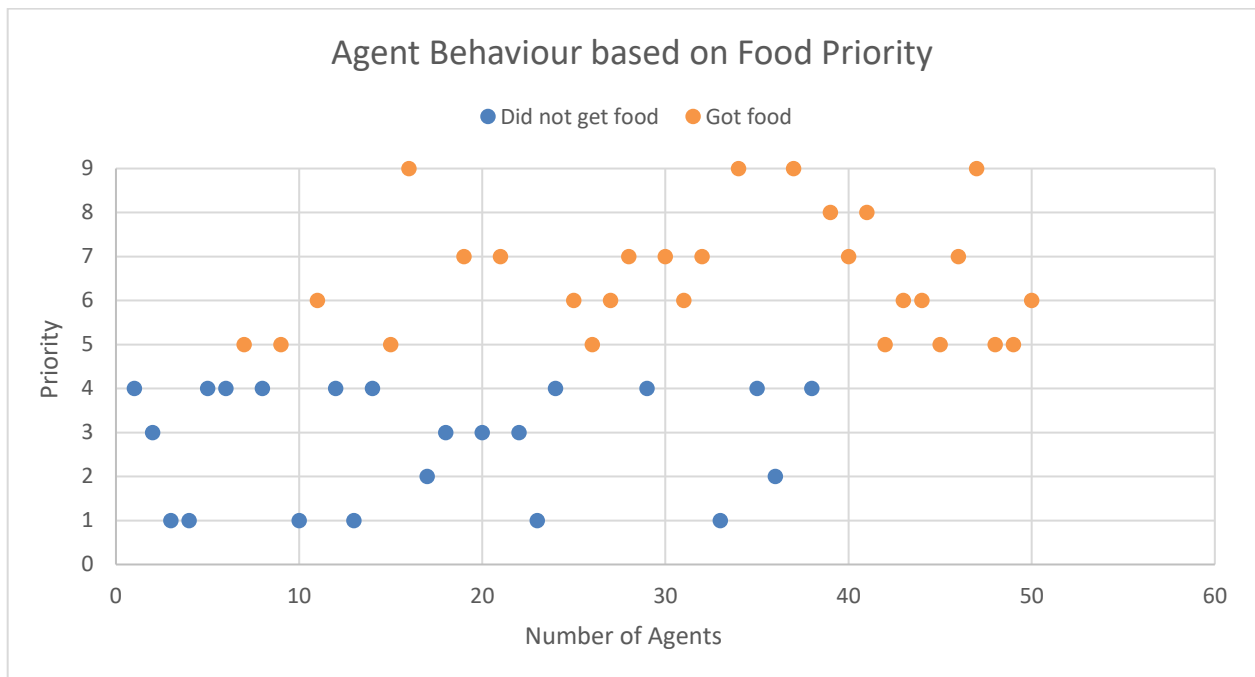


Figure 4.1.w: Comparing which agents got food with each agent's priority for food availability.

Summary

In summary, this chapter explored new simulation components that are proposed by this thesis, such as perception navigation, field of view visibility, and airport priorities. The purpose of these tests was to illustrate how they influence agent behaviour in different architectural conditions. The conditions that were covered include wayfinding, different sized spaces, and non-processing domains. The thesis verifies that these components model the given architectural conditions and corresponding agent behaviours as intended.

The Wayfinding test demonstrates how agents navigating with perception do not always take the shortest path like direct navigation, because they can be influenced by information from the environment. The test makes agents navigate to a gate through a T-junction that had a short and long path. The test shows that, if a wayfinding sign is placed at the T-junction and told agents to take the longer path to the gate, then agents with perception will follow it. This result is true since agents are not aware of the gate location and can only rely on information from the environment to inform their decisions. This is unlike direct navigation, which always takes the shorter path, regardless of the information on the wayfinding sign.

The Visibility test illustrates that isovist geometry, which represents an agent's field of view, can correctly identify the visibility difference between wide and narrow architectural spaces, relative to each other. The test has agents walk back and forth between a wide corridor and a narrow corridor and records the change in field of view area and relative visibility during the journeys. Visibility is calculated as a ratio to the maximum observed area over time, where 1.0 equals the largest area. The field of view correctly shows that moving from the wide corridor to the narrow corridor reduces visibility. Likewise, moving from the narrow corridor to the wide corridor increases visibility.

The Non-Processing Priorities test demonstrates how agents who are assigned a high priority for a non-processing domain, like food, are more likely to interact with food areas in an airport than agents who are assigned a low priority. The test simulates a typical gate with a holdroom and food area, which requires agents to wait around in until their departure time to board their flight. Before boarding, agents are randomly assigned a food priority on a scale of 1 to 9. It is observed that agents with a priority of 5 or higher go get food in the food area, as defined in the simulation model. Whereas agents with lower food priorities wait in the seating area. This verifies that changing an agent's priority of a non-processing domain can influence a passenger's behaviour in a simulated environment.

Chapter 4.2

Terminal Tests

This chapter explores the architectural value of a hypothetical terminal layout. Changing the arrangement of a floor plan can influence passenger behaviour, which corresponds to the overall architectural value. A terminal layout is tested with three different configurations for the security area. If all else is equal, then the difference in architectural value represents the changes from the security layout. The configurations that are explored include (Fig.4.2.a):

- Security area is aligned to the *centre* of check-in
- Security area is aligned *asymmetrically* to one side of check-in
- Security area is *perpendicular* relative to check-in

In addition to the physical layout, the thesis also considers how the overall architectural value is affected by different agent priorities for each design:

- Passengers are assigned *random* priorities for all domains
- Passengers are assigned *high* priority for security
- Passengers are assigned *equal* priority for all domains

The following tests can be described as a Monte-Carlo simulation. Passenger behaviour is given random variables; therefore, the output should produce a binominal distribution. Given a large enough sample size, the distribution should approach a normal curve.

All tests have a sample size of 50 agents. Agents use perception navigation and are assigned characteristics based on the IMO population distribution. For the first set of sets, agents are assigned random priorities for the six pre-defined airport domains. For other priority conditions, see *Priority Range Tests* later in this chapter.

The terminals are only designed with the core areas for departure as described by the National Academies. Each layout has a basic check-in processor, security screening, and a holdroom concourse. These spaces are laid out as a linear sequence, which passengers can only follow in one direction. All layouts follow the same general process, as described below.

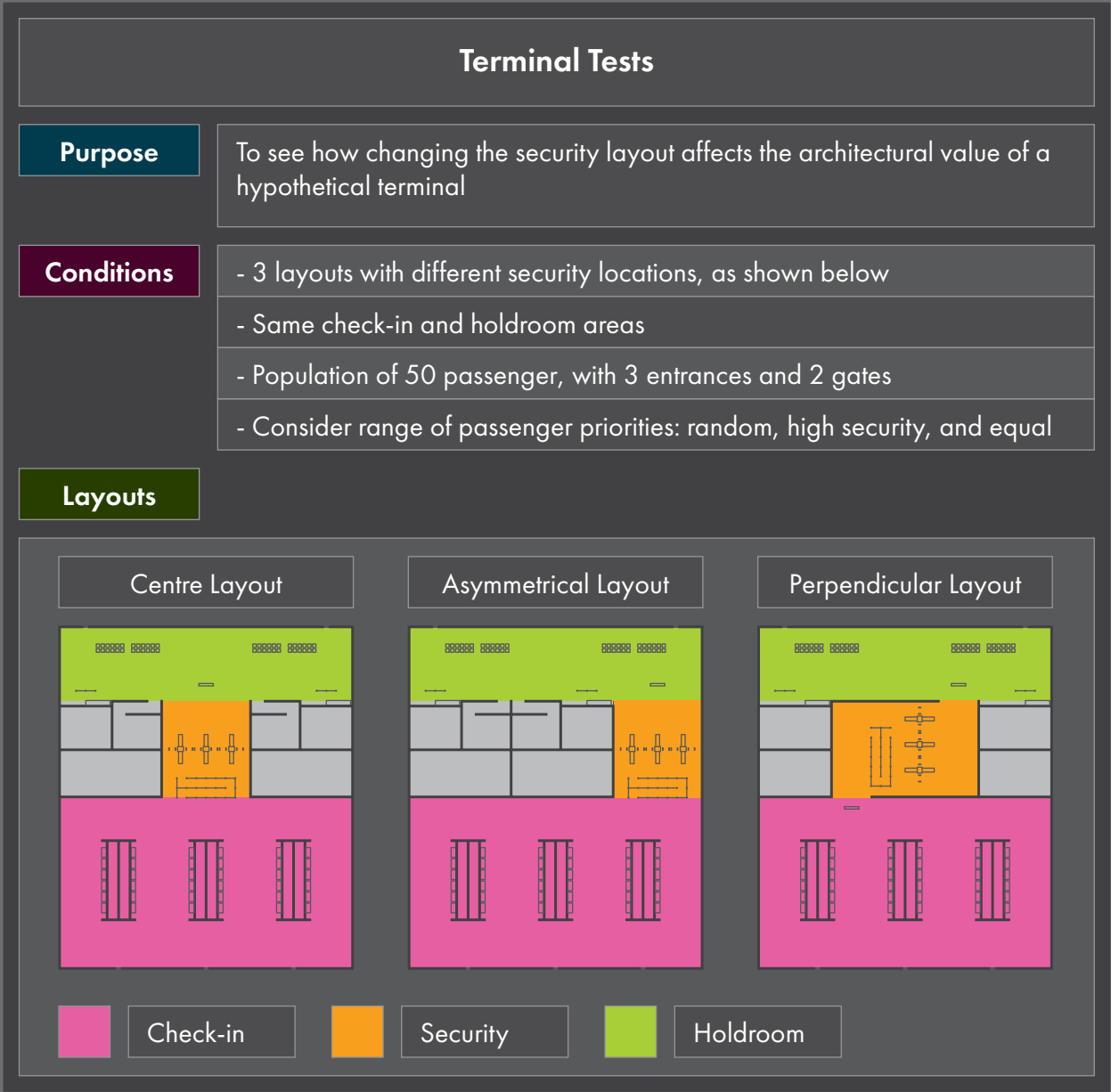


Figure 4.2.a: Setup and conditions for the terminal tests.

Passengers randomly enter the terminal, one at a time, between 1 and 5 seconds, from one of three entrances. The check-in processor has 3 parallel islands of 12 counters each for passengers to drop off their baggage and pick up their boarding pass. Passengers can select any counter at random. Once checked in, passengers can make their way to security screening.

The location of security screening is based on the test. It marks the transition between secure and non-secure areas of the terminal, or *landside* and *airside*, respectively. The security area begins with a queue line, which is marked by stanchions. Passengers must pass through this queue line before getting screened. Inside security screening are 6 metal detectors that flank 3 X-ray machines. Once passengers walk through any one of the metal detectors, they can make their way to the holdroom concourse.

At the end of the security area is a wayfinding sign that marks the threshold to the holdroom concourse. The sign points to the left for Gate A and to the right for Gate B, which are the only two gates in the terminal. Each gate has a waiting area with 24 seats. Across from each waiting area is a food/retail stall and a restroom, representing non-processing domains. Passengers wait in the holdroom until the gate's departure time, which is randomly assigned between 8 and 10 minutes into the simulation. The test ends once all passengers have left through the gates.

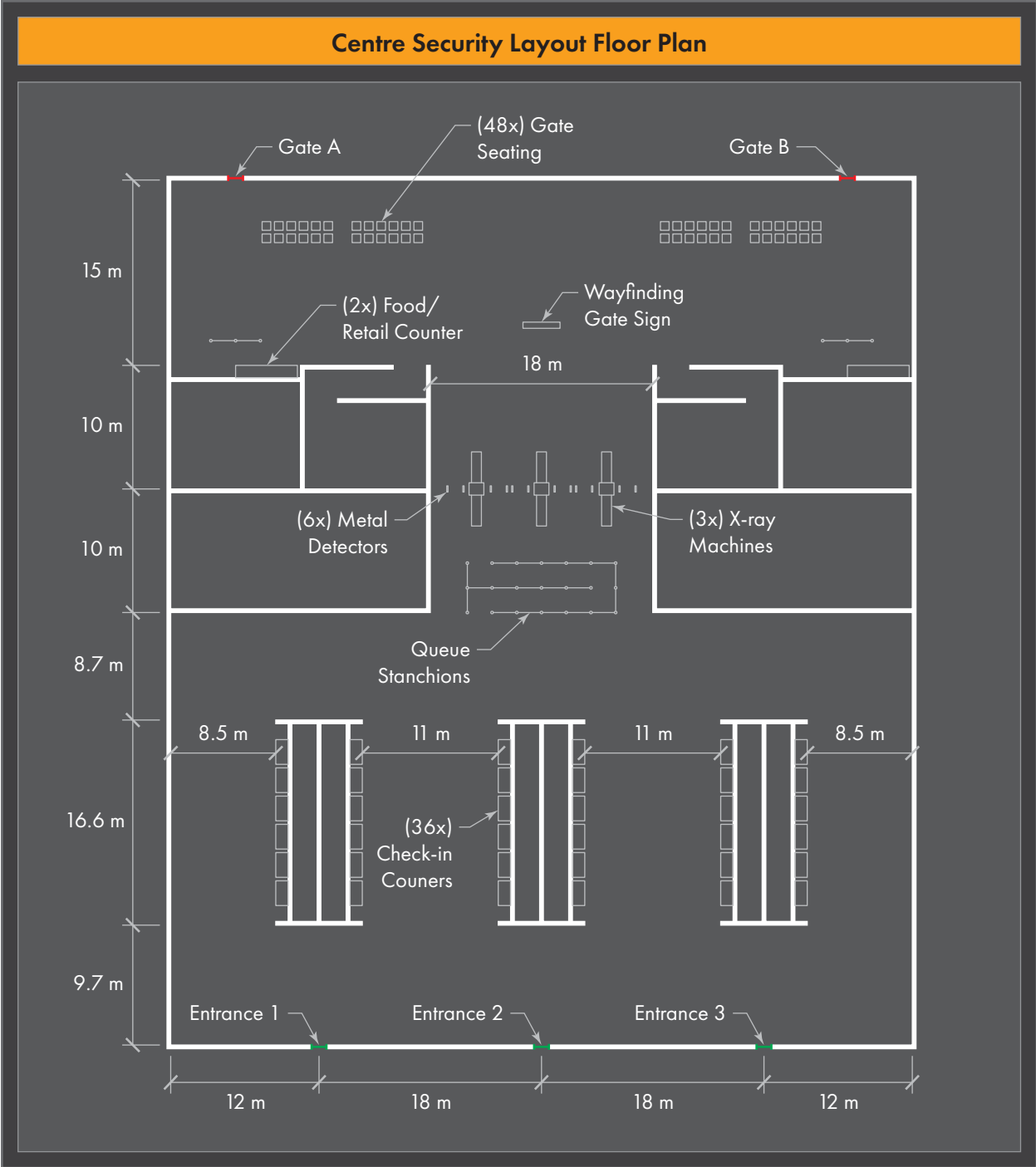


Figure 4.2.b: Floor plan of the Centre Security Layout.

Centre Security Layout Test

The first layout has the security area in the centre of the terminal plan. The results from this first test provide a base value to compare to the other layout conditions. The thesis walks through this test to illustrate the process. However, only the differences will be covered for the following test.

The test begins with passengers entering the check-in area. Passengers tend to walk to check-in counters near the middle of the isle, rather than at the ends, since they have clear view of the middle counters when they first enter. After checking in, it appears most passengers have a direct view to the centre security area from these counters. Nearly all agents walk directly to the security queue line. In some trials, there are one or two passengers who do not notice the security area immediately after checking in. So, these passengers will wander around the check-in isle before noticing the security area from an adjacent isle.

Due to IMO's assigned walking speeds, some passengers walk about three times faster than others. If a fast passenger gets blocked by a slower passenger, then they slow down behind them. During all the tests, it is observed that if there are many faster passengers behind one slow passenger, then they form a train. Although this is intended behaviour, the thesis believes better crowd dynamics should have faster passengers trying to walk around slower moving people.

Passengers enter the security queue one at a time. This triggers their security processing time. Other passengers behind them slow down if they get too close in the line. Although passengers do not stop in the queue line, a time delay factor is applied for security wait times based on Wiredja et al.'s research of passenger tolerance. All passengers are able to process at the x-ray machines and walk through the metal detectors successfully.

All passengers notice the wayfinding sign, immediately after going through the metal detectors. As a result, they all walk towards the sign and follow the direction it is pointing. Since departure gates are randomly assigned, a similar number of passengers walk to the left and right, to Gate A and Gate B, respectively. Since the security area was in the centre, passengers have an equal distance to walk for either gate.

Since the terminal concourse is small, passengers immediately recognize their gates when they enter the holdroom. It appears passengers who have faster walking speed are the first people to arrive at the gate, which is well ahead of the assigned departure time. As mention for the first set of tests, agents are randomly assigned priorities. It is observed that some passengers walk over



Figure 4.2.c: *Passengers in the check-in area (dark blue: walking to counter, orange: processing).*

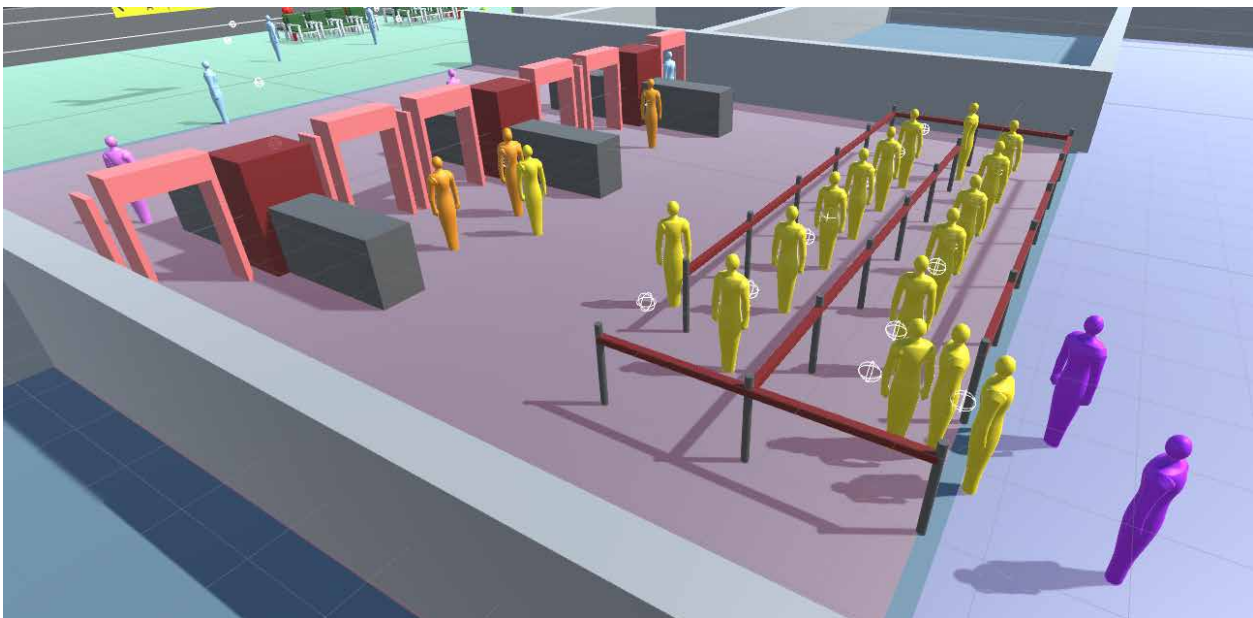


Figure 4.2.d: *Passengers in security screening (purple: walking to security, yellow: in queue line, orange: processing).*

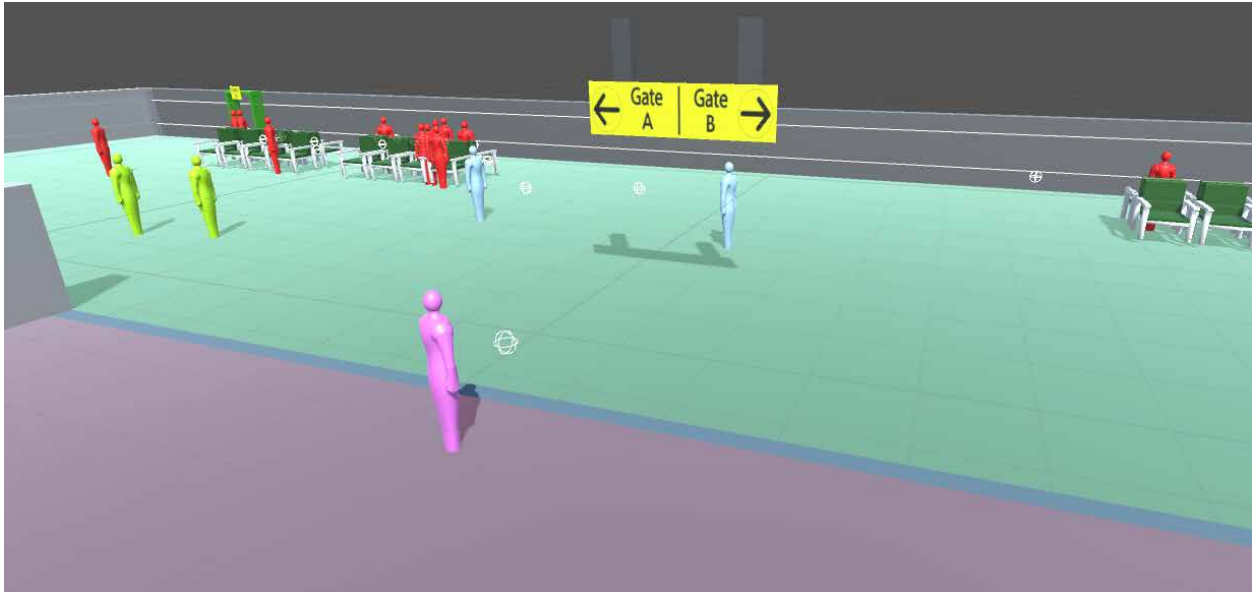


Figure 4.2.e: *Passenger entering holdroom concourse before the sign (pink: reading sign, light blue: wandering, light green: going to food area, red: waiting in seating area).*

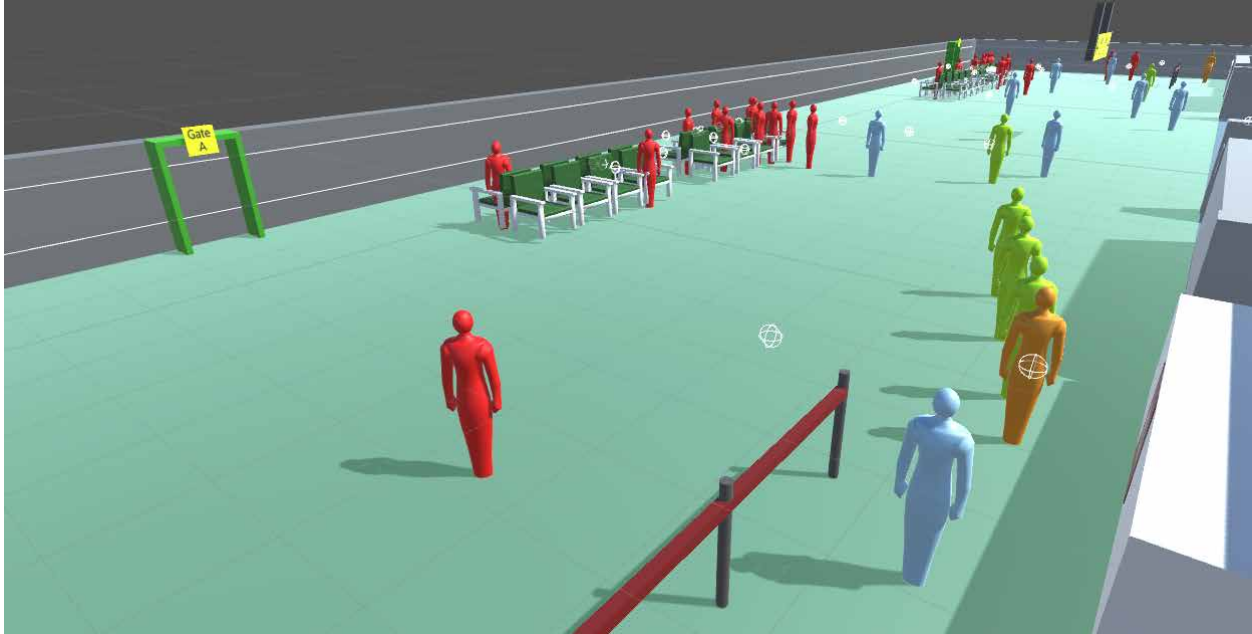


Figure 4.2.f: *Passengers linger in holdroom concourse (light blue: wandering, orange: processing, light green: going to food area, red: waiting in seating area).*

to the seating area to wait, whereas other people make their way to one of the nearby food stalls, which indicates that they have higher food priorities. Once these passengers get their food, they make their way back to the seating area in front of their gate.

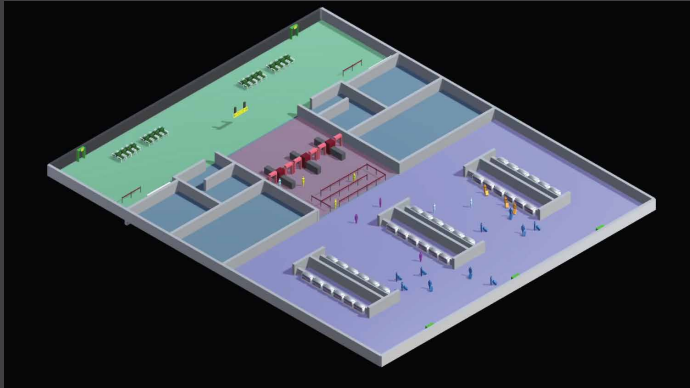
All passengers successfully reach the holdroom concourse before their gate's departure time. It is observed that some passengers sit in seats that are already occupied by other people. But this does not influence architectural value for this simulation. For some trials, it appears a couple of passengers are unable to find seats, possibly due to the number of people. Although, these passengers continue to wander around the holdroom area. This has some influence on architectural value for this simulation since agents have gate seating as a priority.

After about 9 minutes of simulation time, it reaches the first gate's departure time. All passenger departing from that gate get up from the seating and make their way through the portal. Several seconds later, the other gate opens, and the rest of the passengers exit through their portal.

For this test, all 50 passengers completed the simulation successfully. The thesis also performed multiple trials under the same conditions to get a larger sample size. The average architectural value from these trials for the Centre layout is 0.730, and the average security screening value is 0.615. For the moment, these numbers are just a starting point to compare with the other layouts.

The overall binomial distribution for architectural value is illustrated in Fig.4.2.h. The distribution shows that most passengers give an architectural value between 0.75 and 0.80. Additionally, it can be seen that, given enough passengers, the probability distribution should approach a normal curve. The same behaviour is true for passengers' security scores for this layout, which has a highest score between 0.65 and 0.70 (Fig.4.2.i). Therefore, this test produced the expected behaviour of a Monte Carlo simulation.

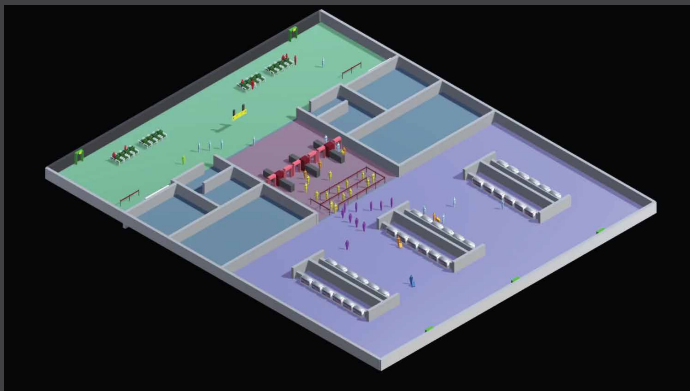
Centre Security Layout



01:04

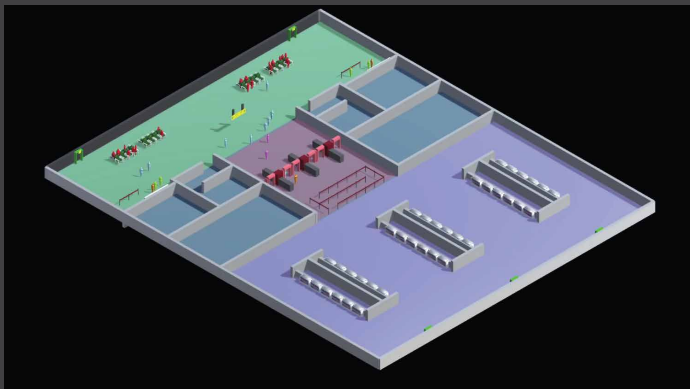
Figure 4.2.g:

Passengers in the check-in area.



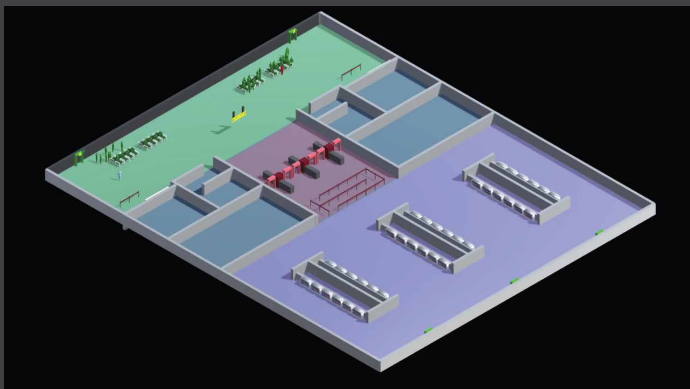
03:18

Passengers making their way through security screening.



05:54

Passengers in the holdroom concourse.



08:34

Passengers going to gate after departure time.

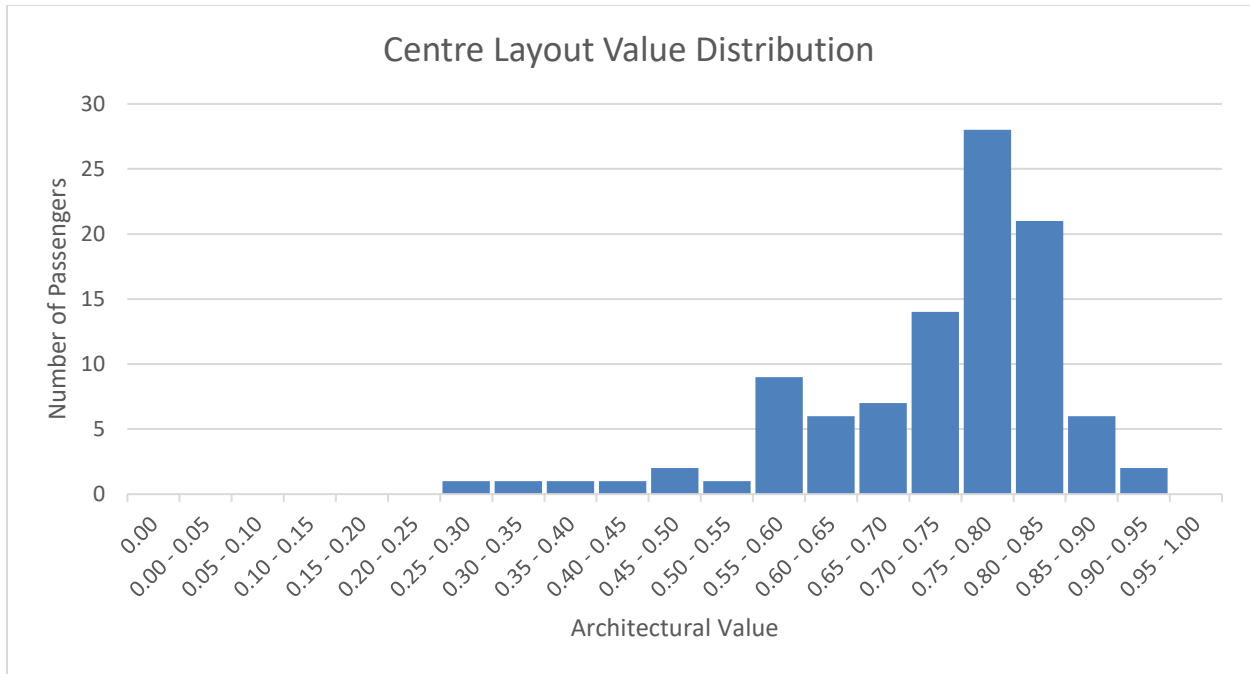


Figure 4.2.h: Binomial distribution for the Centre Layout approaches a normal curve with the highest probability occurring between 0.75 and 0.80.

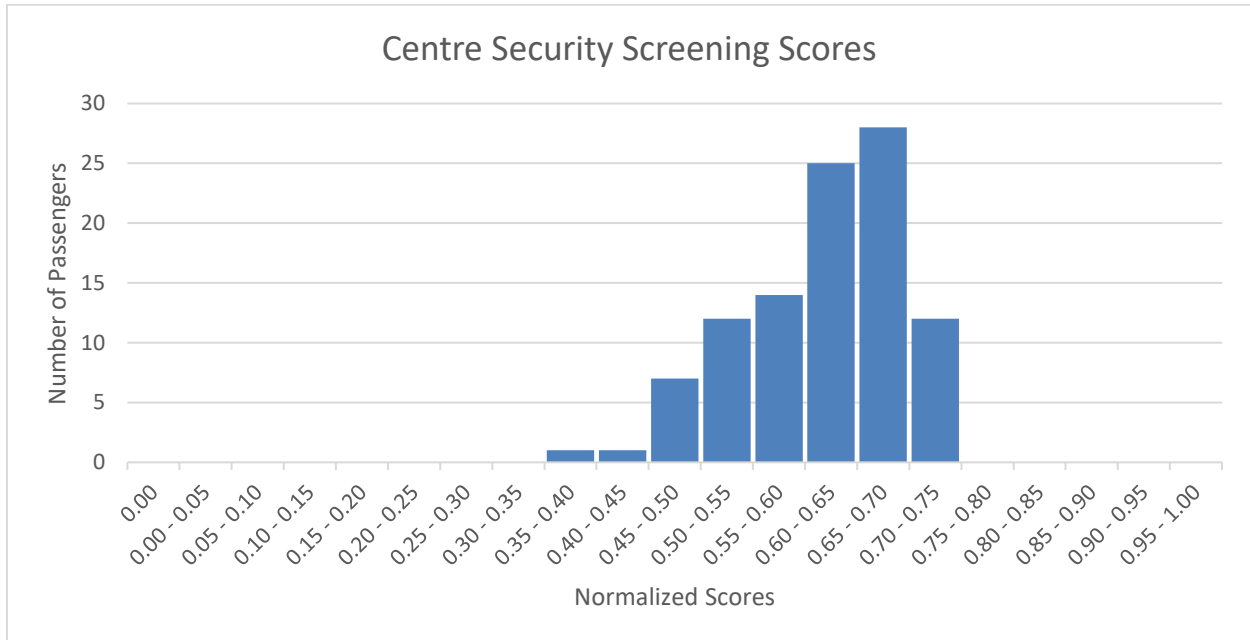


Figure 4.2.i: Likewise, the binomial distribution for the passengers' score for security screening with the highest probability occurring between 0.65 and 0.70.

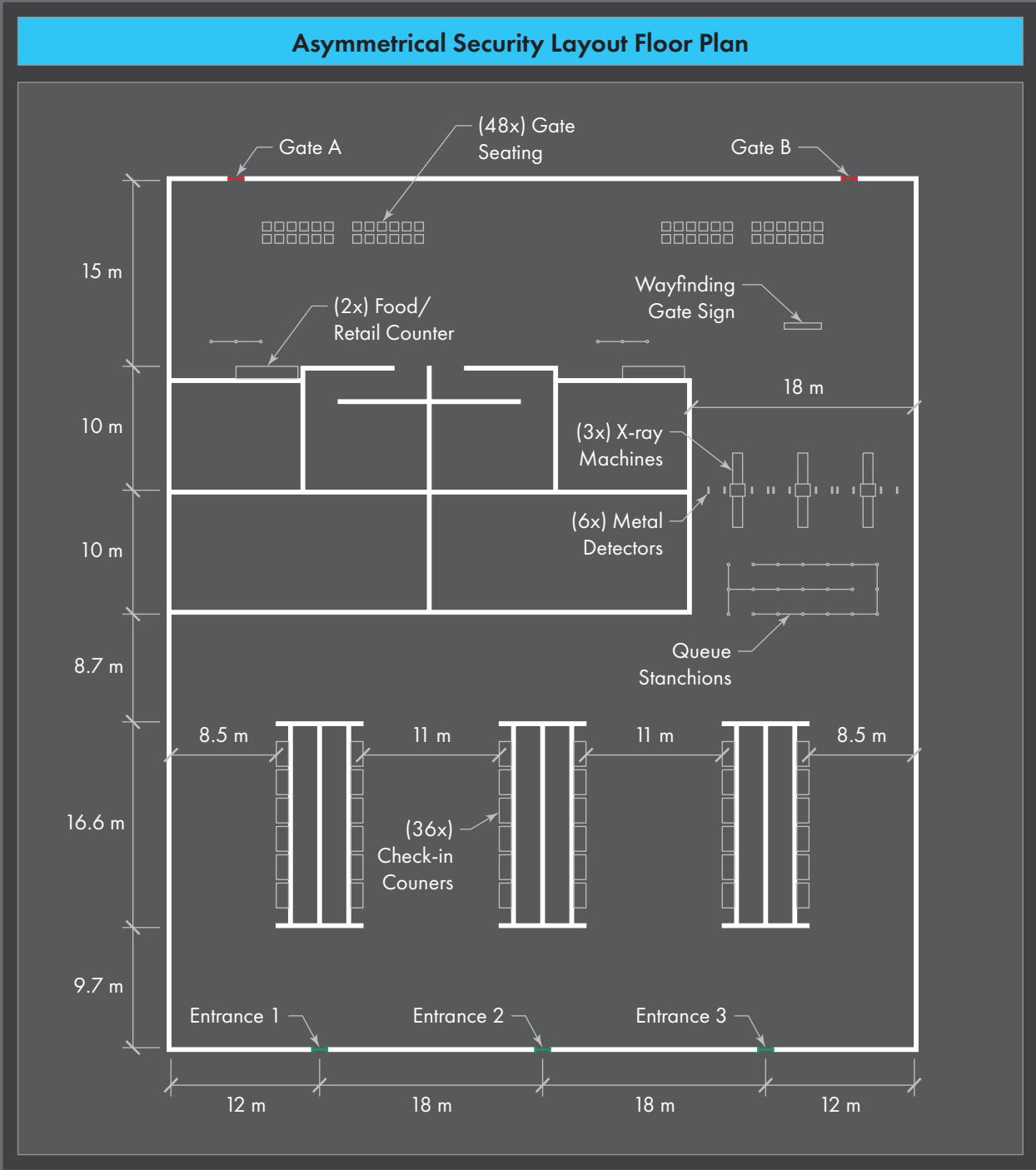


Figure 4.2.j: Floor plan of the Asymmetrical Security Layout.

Asymmetrical Security Layout Test

The second layout has the security screening area aligned to the right side of the terminal, which makes the design asymmetrical. Inside the security area are identical components to the security area in the first layout. It has the same queue stanchions, 3 X-ray machines, and 6 metal detectors. The middle of the terminal is replaced with the same food/retail stall in the holdroom concourse, washrooms, and implied back-of-house areas (Fig.4.2.j).

The thesis conducted this test with the same procedure as the Centre layout, so only the differences are covered here. This test is meant to replicate a new floor plan iteration that an architect might consider during the design phase. The idea for this test is, if the conditions are exactly the same, and only the location of the security area is changed, then the passengers' scores should reflect the differences in the layout. The hypothesis is the Asymmetrical security layout should decrease the security score and reduce the overall architectural value. This is because the security will not be visible to passengers checking in on the left side and it is a further walking distance from Gate A.

Like the first layout, most passengers walk to the middle of the isles in the check-in area when they enter the simulation. Except this time, it is observed that passengers who are at counters on the left side do not have a direct view to the security area. Only passengers on the right side have a direct view to security. When passengers on the left side finish checking in, it is observed that more people return to a wandering state (Fig.4.2.k). These passengers end up walking around the check-in isles searching for security. Once passengers wander around the counters to the right side of the processor, they can recognize the entrance to the security queue. Passengers walk directly to the security area when it becomes visible at this point (Fig.4.2.l).

After passengers make their way through security, the process becomes similar to the first layout again. Some passengers in this layout are further away from Gate A when they leave security. There is a wayfinding sign at the entrance to the holdroom concourse which directs people to the left for Gate A. But it does take longer for passengers to reach Gate A in this layout, as is expected.

One observation the thesis did not consider was the improved visibility for passengers going to Gate B. Since security screening is right across from this gate, passengers had a much better experience finding Gate B than they did in the first layout. Likewise, if these passengers also checked in on the right side of the processor, then their distance travelled is significantly shorter, and their visibility would have been a lot higher than before when they found the gate.

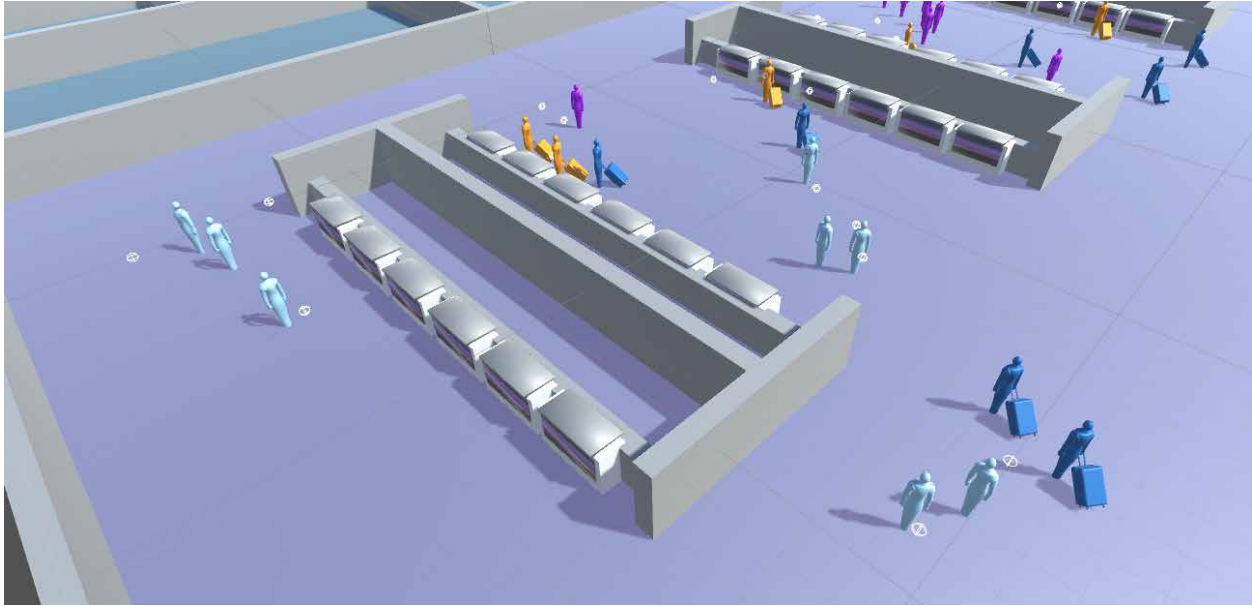


Figure 4.2.k: *Passengers wandering (light blue) between check-in isles because they do not see the security area from left side of the check-in processor.*

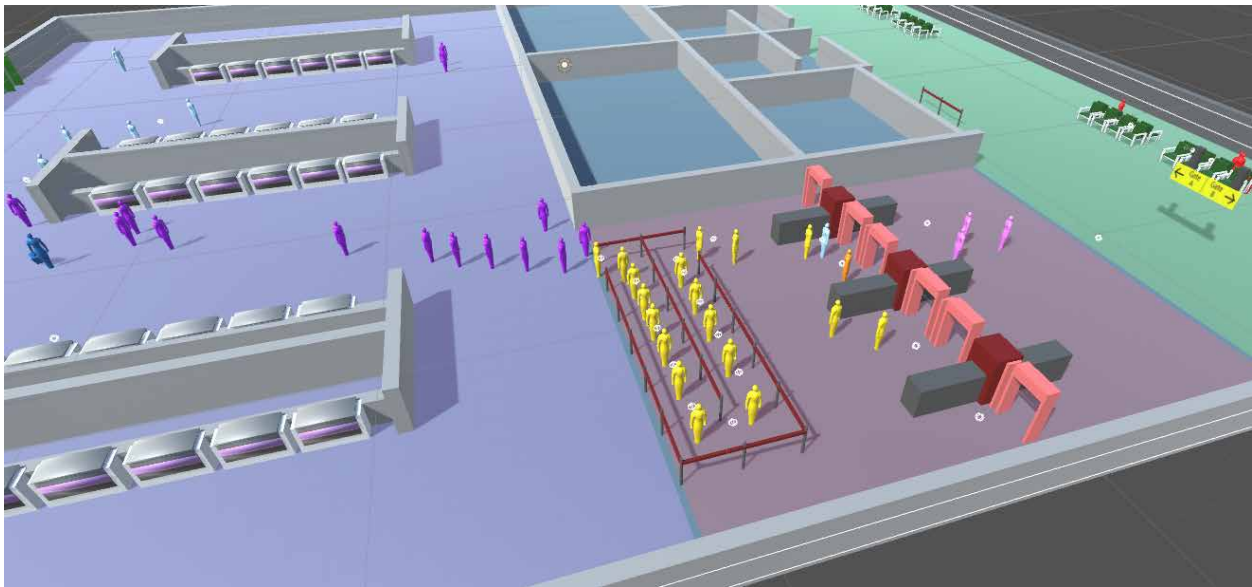


Figure 4.2.l: *Security screening has a bias for passengers checking in on the right side (purple).*

All 50 agents completed this test successfully. Like the first test, multiple trial runs were performed under the same conditions to increase the sample size. The average architectural value for the Asymmetric layout is 0.721, and the average security screening value is 0.505. This security score is about 10% lower than the first layout, mainly as a result of more passengers wandering around. However, the average architectural value is only 1% lower, which is not a significant change.

It is understandable that there might only be a small change in architectural value because security screening is not the only passenger priority. Other airport domains may have had higher priorities for some passengers. For example, there were small improvements to the gate availability score. Passengers finding Gate B immediately was a bigger improvement than passengers walking longer to Gate A. The values from these tests are listed in Appendix A for further information.

The overall binomial distribution for architectural value is illustrated in Fig.4.2.m. Once again, the distribution approaches a normal curve, given enough passengers, which is characteristic of a Monte Carlo simulation. The architectural value has the highest probability between 0.70 and 0.75. A comparison between the architectural value distributions shows that the Asymmetrical layout did cause a 5% decrease in the maximum probability (Fig.4.2.n). Likewise, the peak security score is between 0.40 and 0.45, which is a 25% decrease (Fig.4.2.o). Interestingly, there is a distribution of passengers who scored similar to the Centre layout between 0.60 and 0.70 (Fig.4.2.p). This may be coming from passengers who checked in on the right side and departed from Gate B.

It is not important if the exact differences are measured to the nearest percentage. But what is important is that there is a probability distribution illustrating what range of passenger scores are likely to occur. Fundamentally, the Asymmetrical layout is more likely to give a slightly lower performance than the Centre layout, for a large number of people. It also suggests that the security area is most likely to impact passenger behaviour. Although these differences may be obvious in the floor plan, the simulation provides a quantifiable comparison of this behaviour.

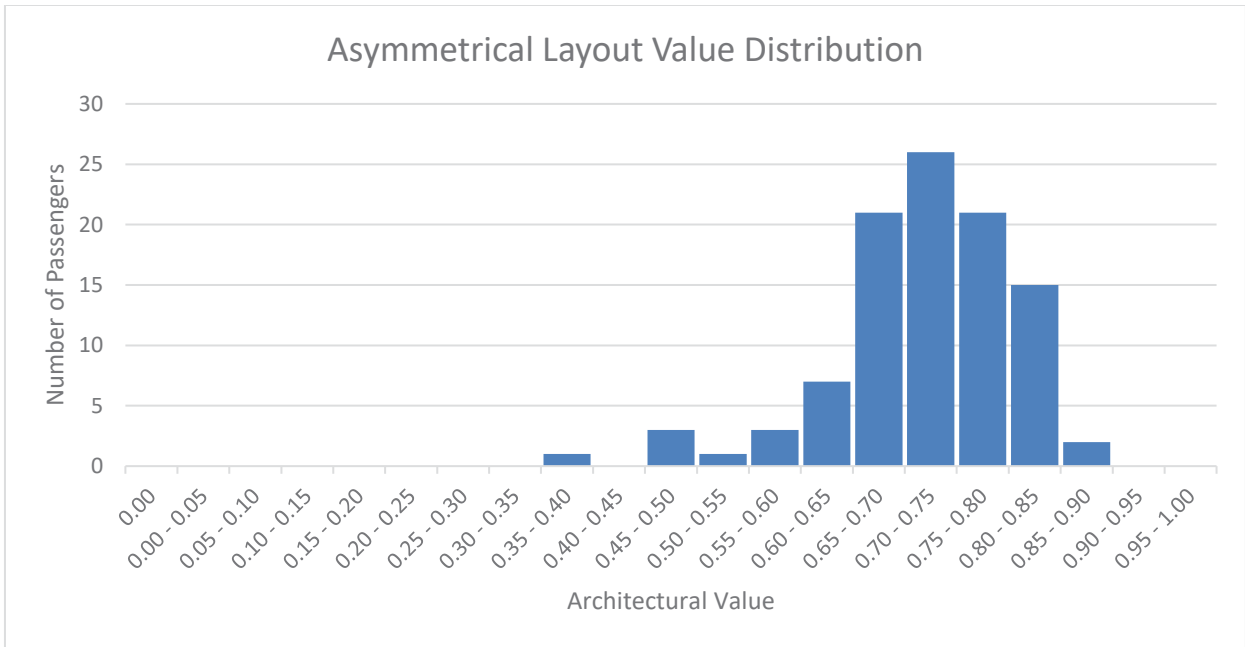


Figure 4.2.m: Binomial distribution for the Asymmetric Layout approaches a normal curve with the highest probability occurring between 0.70 and 0.75.

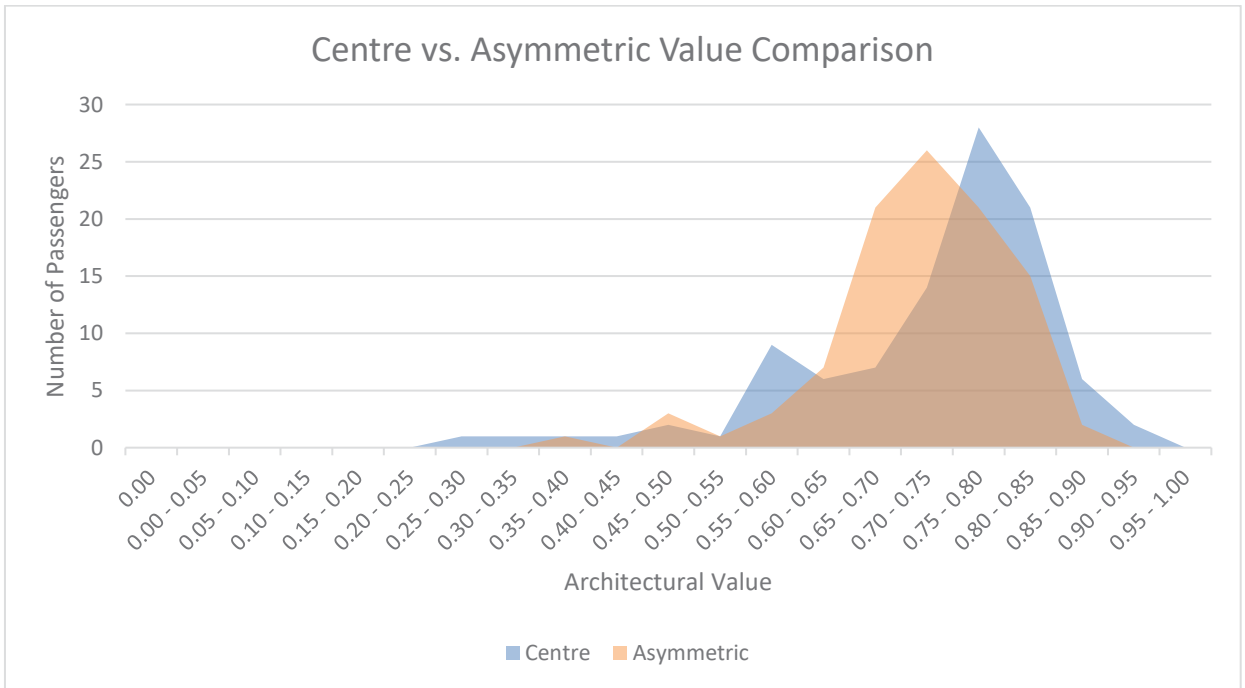


Figure 4.2.n: Comparison shows Asymmetric values (orange) has a lower distribution than Centre values (blue).

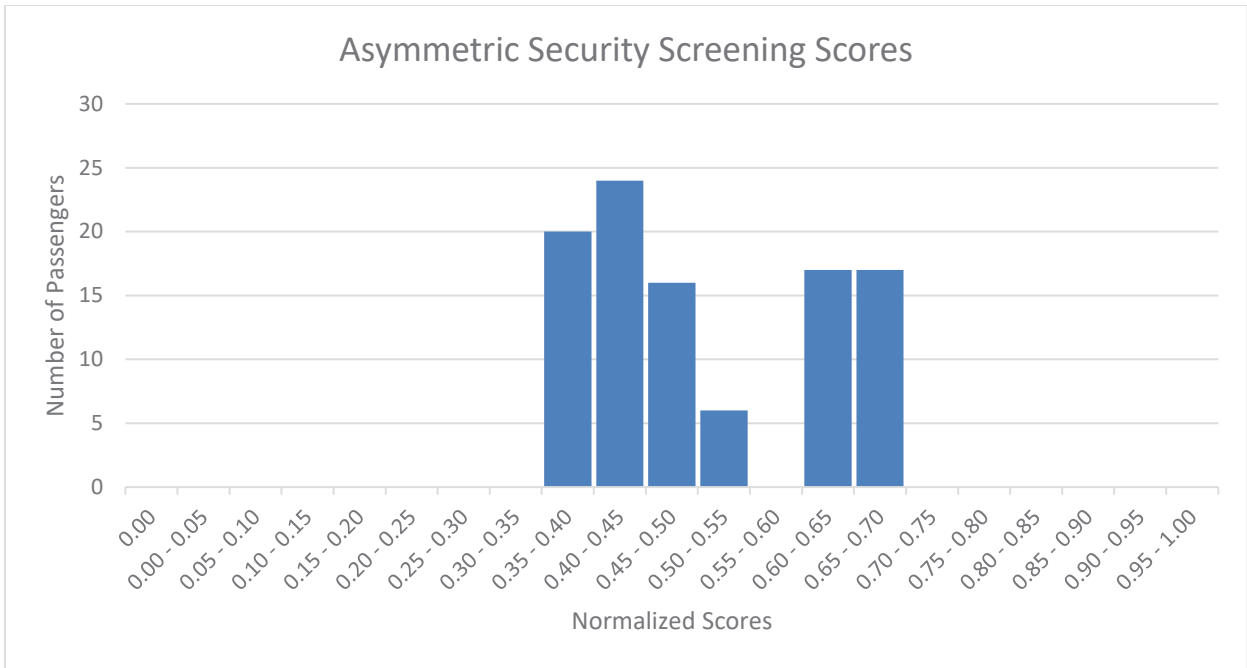


Figure 4.2.o: Binomial distribution for the passengers' security scores has a peak between 0.40 and 0.45. Also note the high distribution between between 0.60 and 0.70.

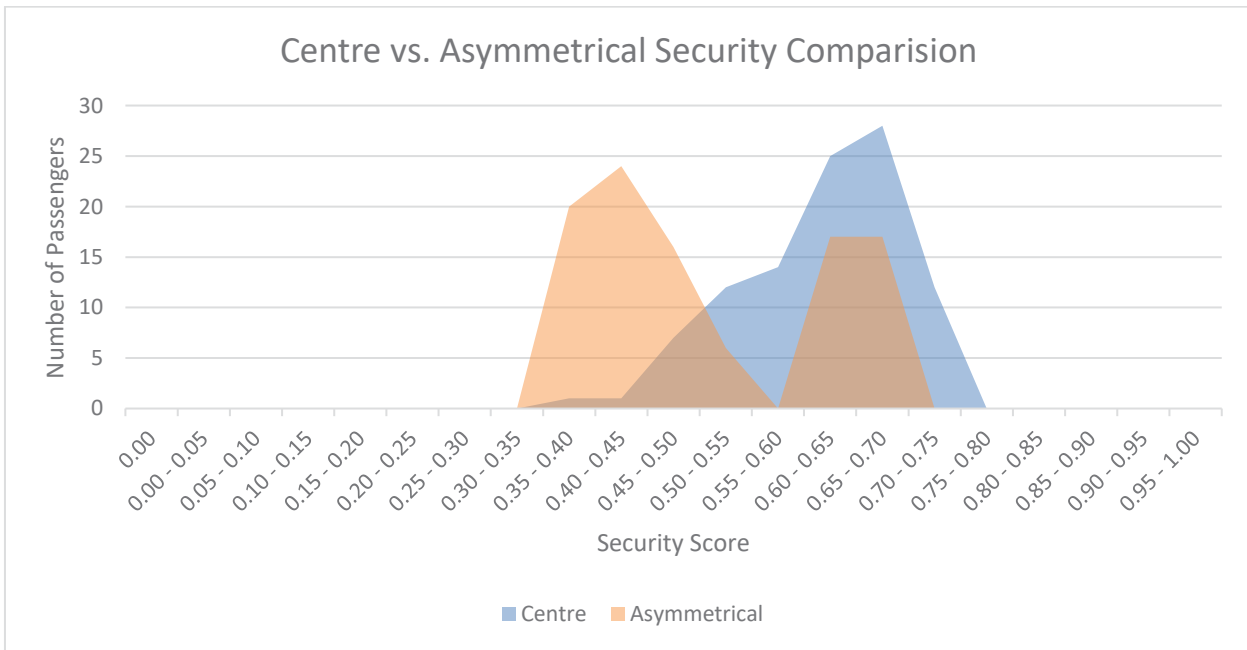


Figure 4.2.p: Comparison shows Asymmetric security (orange) has a lower distribution than Centre security (blue). Some passengers in the Asymmetrical layout did score similar to the Centre layout.

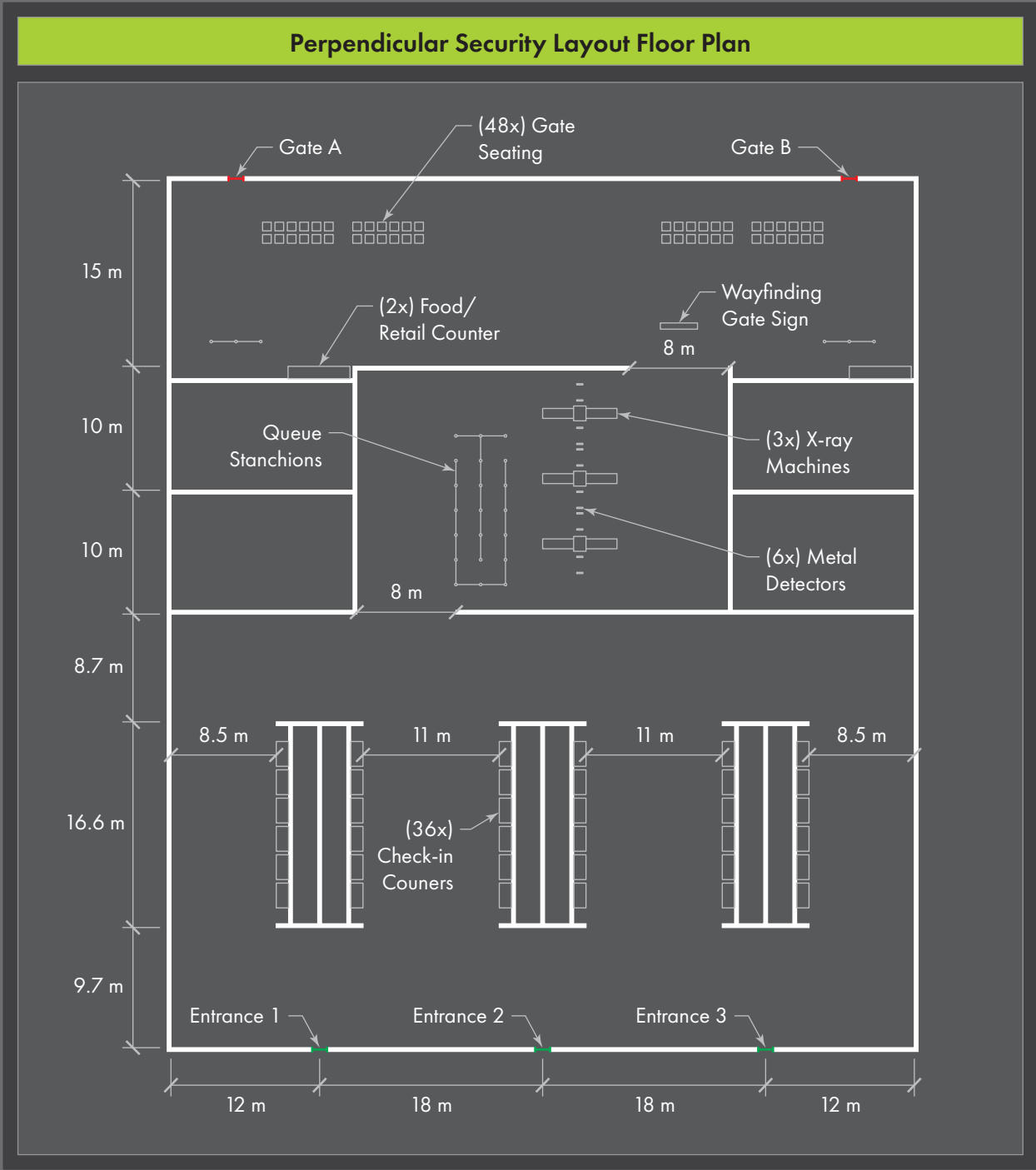


Figure 4.2.q: Floor plan of the Perpendicular Security Layout.

Perpendicular Security Layout Test

The final layout has security screening rotated 90°, in the middle of the terminal, perpendicular to way it was placed for the first test. Like the first two tests, it includes the same queue, X-ray machines, and metal detectors. Since the layout is perpendicular, the security area is wider than the first two tests to provide walk-up space for passengers to approach. Passengers enter and exit into the security area through a 8 m wide threshold. The entrance threshold aligns with the left isle of the check-in processor and the exit threshold is in front of the seating at Gate B. The food/retail stalls fill in the spaces adjacent to the security area, and the rest is implied as washrooms and back-of-house areas (Fig.4.2.q).

This test is performed under the same conditions as the previous two tests. The hypothesis is the Perpendicular layout will produce the lowest security score and architecture value out of all three layouts. The narrow threshold should make the security area difficult for passengers to find and the perpendicular direction may reduce the passenger's overall visibility during screening.

Like the other tests, passengers randomly enter the check-in processor and walk to service counters in the middle isles. Since the entrance to security is aligned to the left isle, passengers on the right side of the check-in area do not have any view of security. As a result, many passengers are observed wandering around trying to find security screening during the test (Fig.4.2.r). From the angle these passengers approach the security threshold, the back wall of the check-in area hides the entire security area (Fig.4.2.s). When passengers approach the threshold from the right, they cannot see any identifiable features of the space, like the queue line. However, when passengers approach the threshold from left, are they able to see the security queue stanchions (Fig.4.2.t).

Once passengers enter security screening, the process is the same as the other layouts. After screening is complete, passengers must leave the area by walking to the left. There is a gate wayfinding sign at the exit threshold, which makes navigating easier than it was for the entrance (Fig.4.2.u).

The simulation continues until all passengers leave through their departure gate. During one trial, there were still a couple passengers lost in the check-in area even after departure time. These people eventually found their way through security, after walking down to the far-left end of the processor, and managed to finish the simulation successfully, despite being late.



Figure 4.2.r: *Passengers in the right isle of check-in looking for the security area (light blue: wandering, dark blue: going to check-in counter).*

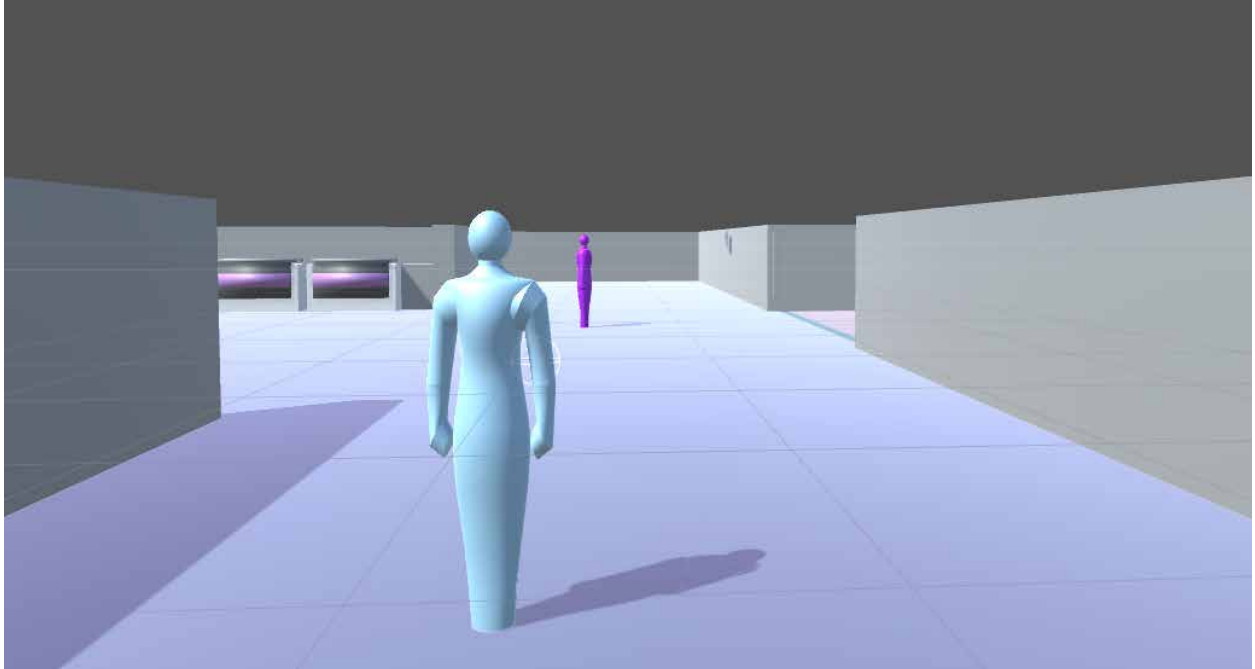


Figure 4.2.s: *Passenger's view walking along the wall from the right cannot see any identifying feature for security at the threshold.*



Figure 4.2.t: *Passengers approaching from the right side (purple) recognize the security queue sooner than passengers approaching from the left side (light blue), due to the narrow opening.*

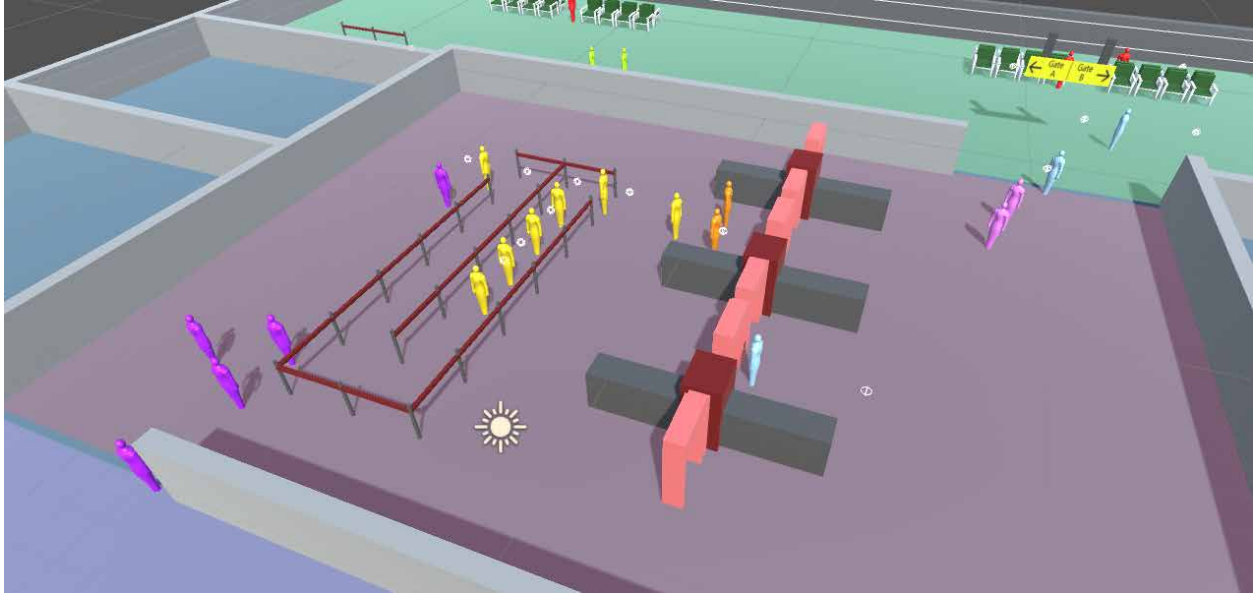


Figure 4.2.u: *Perpendicular security screening area, with an exit to the left towards the wayfinding sign (purple: walking to security, yellow: in queue line, orange: processing, pink: reading sign).*

However, not all agents struggled finding the security area. Like the Asymmetrical layout, some passengers who entered the check-in processor on the left side had a clear view into the security area from the start. As a result, once they got through security, these passenger's experience was practically similar to the Centre layout test.

The results of the Perpendicular layout were as successful as the previous two tests. Once again, multiple trial runs were performed under the same conditions to increase the sample size. The average architectural value for the Perpendicular layout is 0.691, and the average security screening value is 0.483. These are the lowest values of all three layouts.

The overall binomial distribution for architectural value is illustrated in Fig.4.2.v. The architectural value has the highest probability between 0.75 and 0.80. Surprisingly, this is similar to the Centre layout, despite having a lower average. Although this difference might be due to the Perpendicular distribution having a higher variance. When comparing all three layouts, the Centre layout has the highest peak distribution, followed closely by the Perpendicular layout, and then the Asymmetric layout (Fig.4.2.w). However, all three layouts share a similar distribution shape, which shows that most passengers had similar overall experience for every terminal, despite the security screening differences.

The distribution of security scores for the Perpendicular layout are shown in Fig.4.2.x. The peak scores are between 0.35 and 0.40. Although, like the Asymmetrical layout, there is a significant distribution of passengers who scored between 0.65 and 0.70. This may be from passengers who entered the processor on the left, which gave them a clear view into the security area. When comparing all three layouts, the Centre layout had the highest peak distribution, which was well ahead of the Asymmetrical layout, and then the Perpendicular layout (Fig.4.2.y). However, as mentioned before, both the Asymmetrical layout and Perpendicular layout had a significant distribution of people who match the Centre layout security scores. This indicates that not all passengers had the same experience, despite walking through the same terminal design.

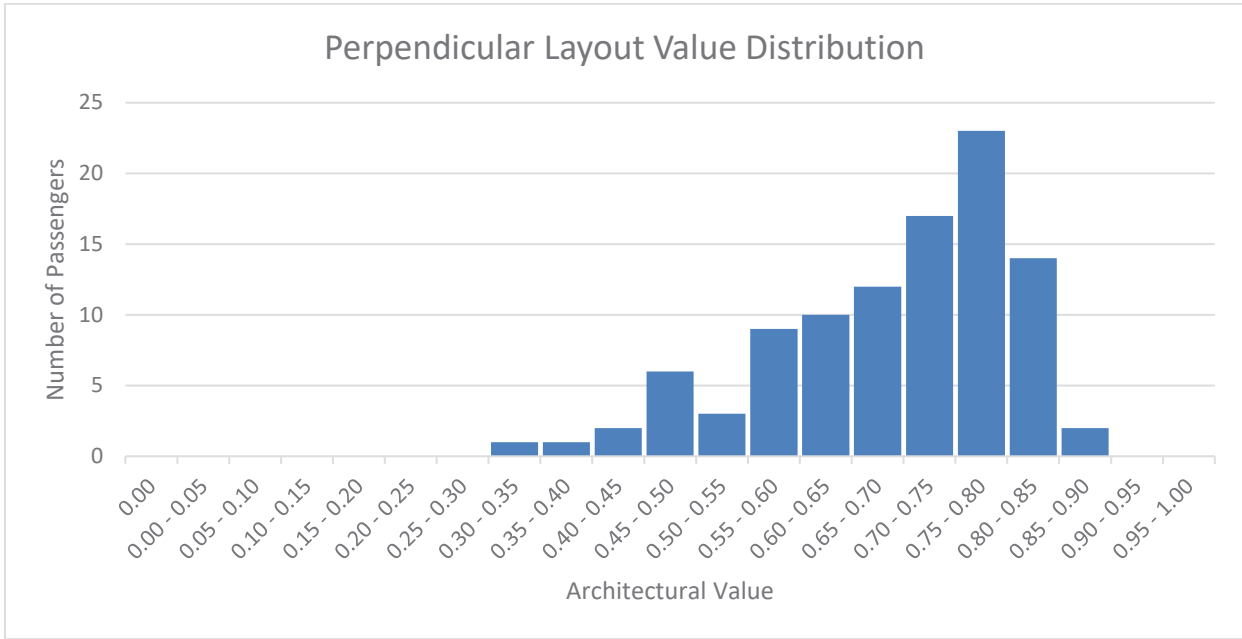


Figure 4.2.v: Binomial distribution for the Perpendicular Layout approaches a normal curve with the highest probability occurring between 0.75 and 0.80.

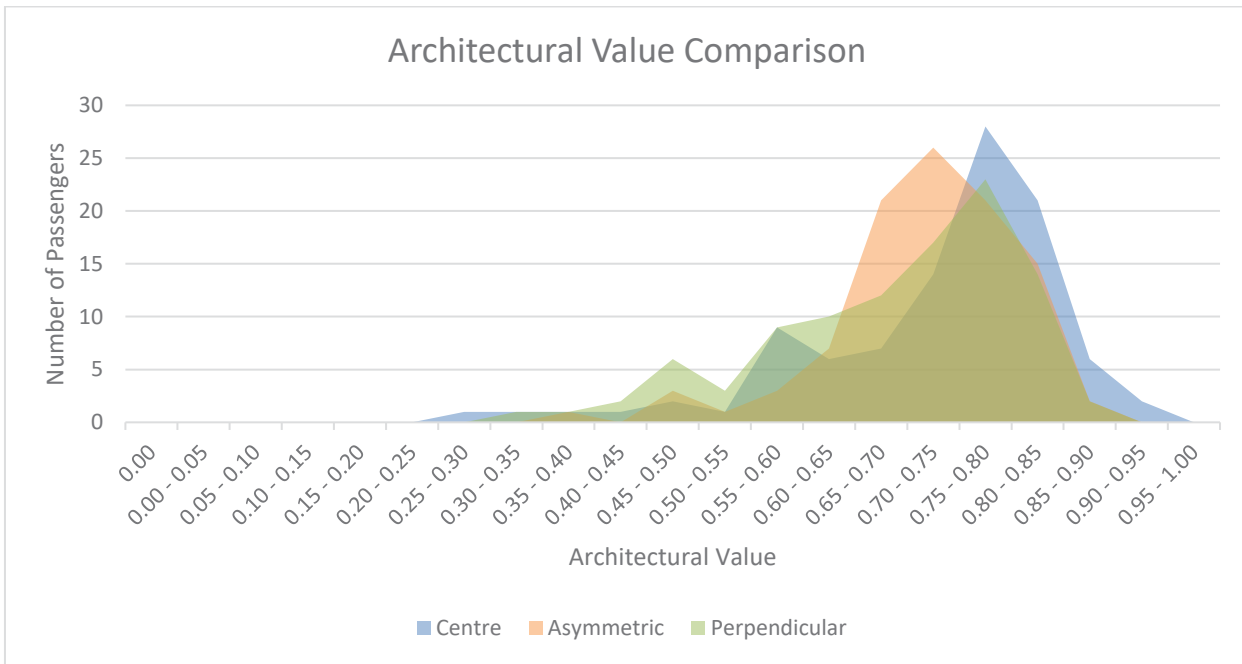


Figure 4.2.w: Comparing the distribution of all layouts' architectural value.

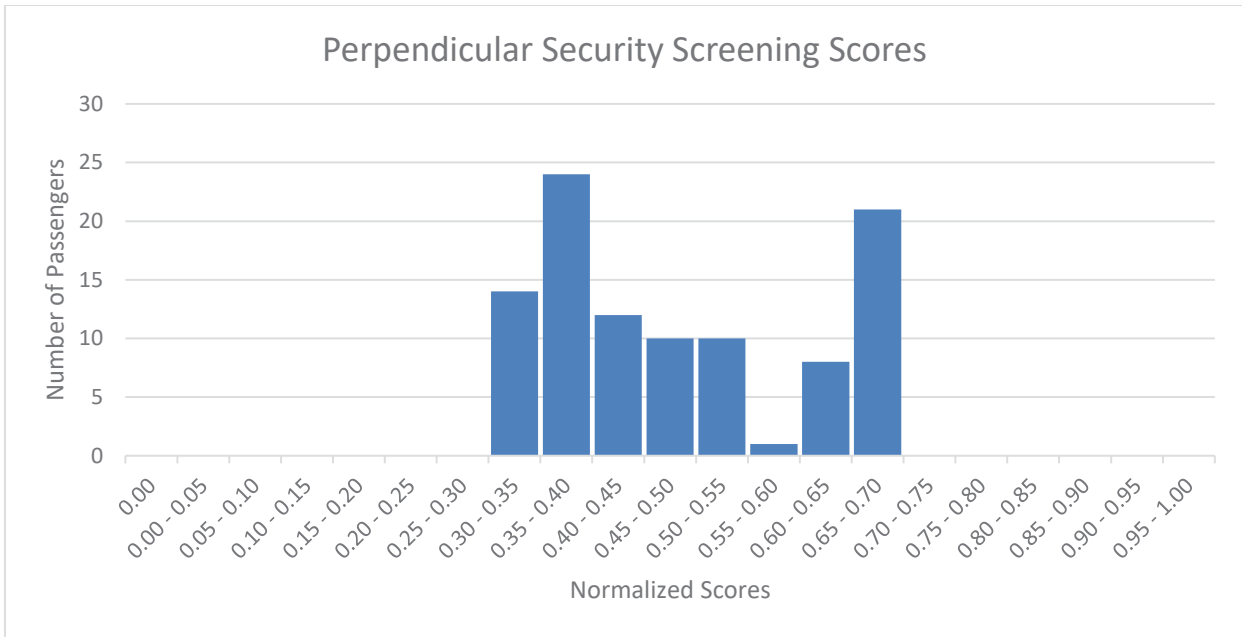


Figure 4.2.x: Binomial distribution for the passengers' security scores has a peak between 0.35 and 0.40. Also note the high distribution between 0.65 and 0.70.

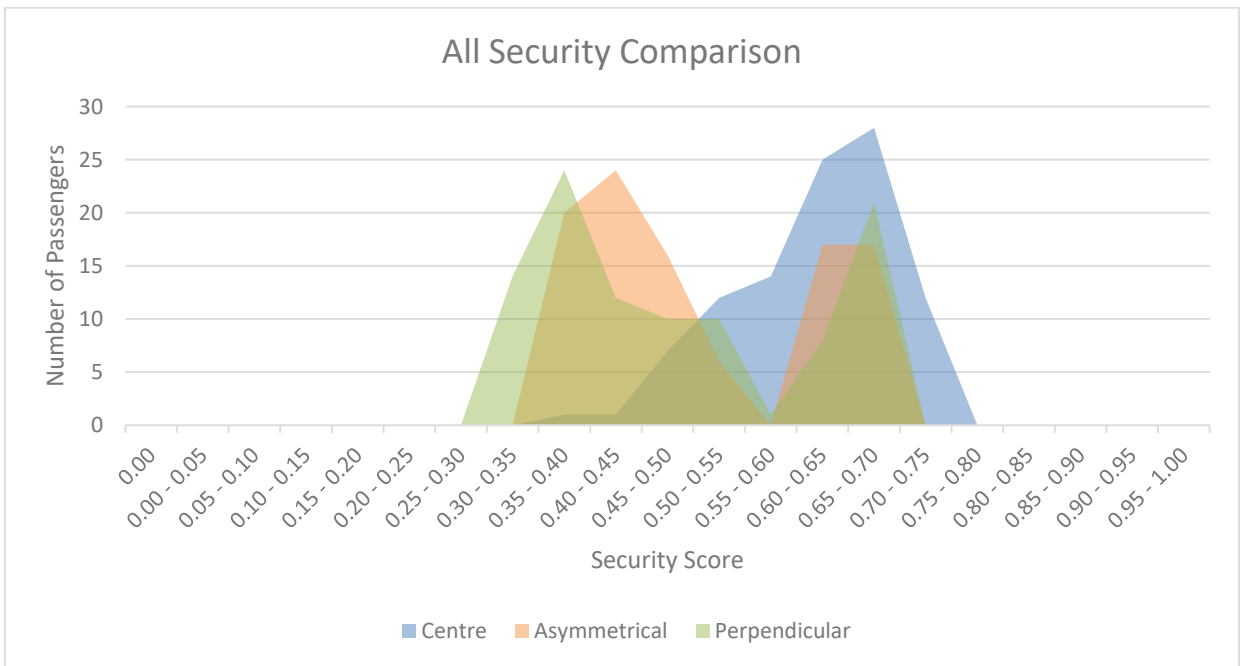


Figure 4.2.y: Comparing all layouts' security score. Passengers in the Perpendicular layout scored the lowest values, but similar to the Asymmetric layout. Centre layout has the highest score distribution.

Priority Range Tests

The previous tests used agents with randomly assigned priorities, like the passenger illustrated in Fig.4.2.z. But what happens to the architectural value if agents are given different priority levels? This test repeats the same experiments as before, except it compares agents for two extreme cases. Firstly, agents who only have a priority for security, and secondly, agents who prioritize all airport domains equally. Since the security area is the primary factor, this should illustrate the maximum range of architectural values for these terminals.

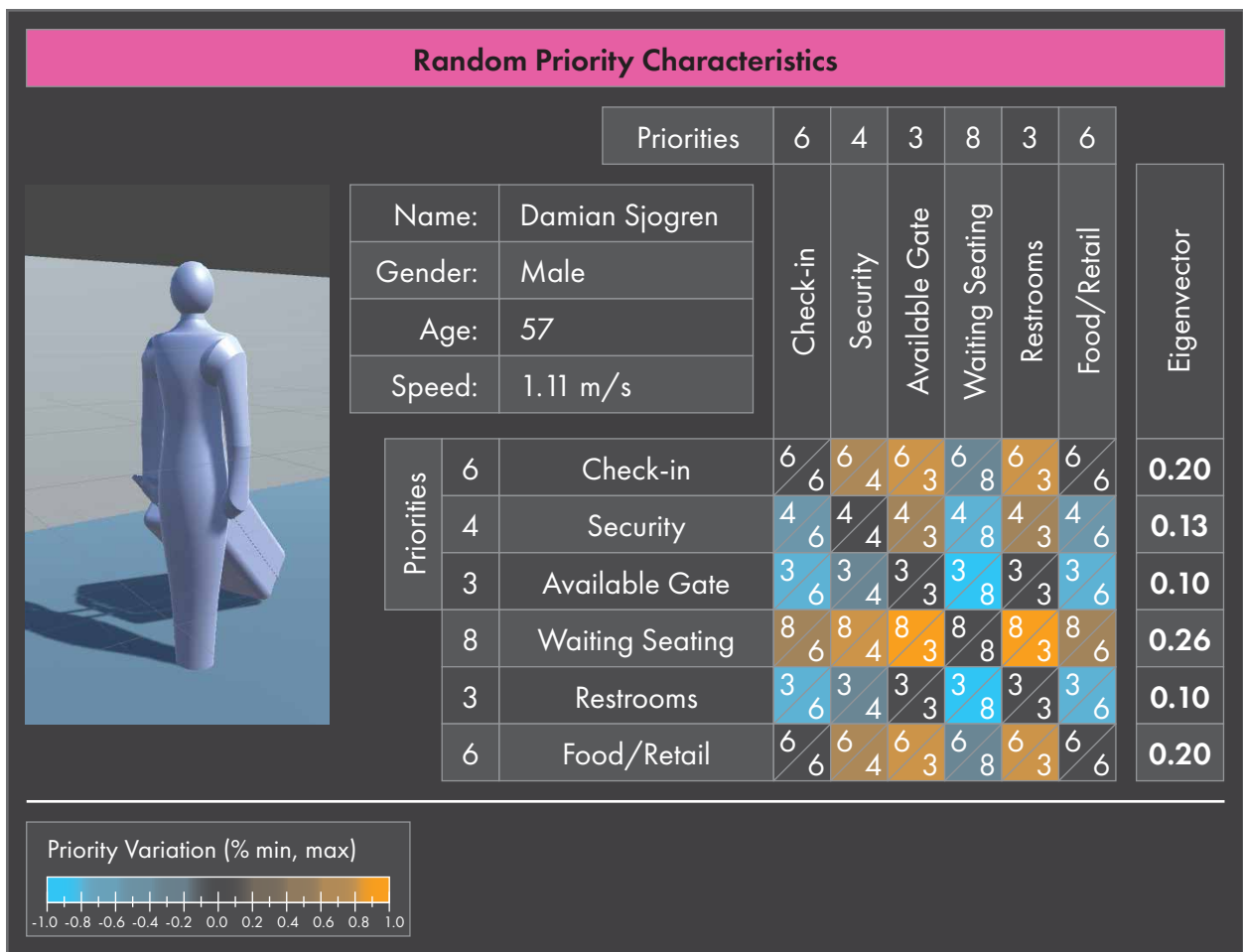


Figure 4.2.z: Assigned agent characteristics and random priority matrix.

High Security Priority Tests:

Agents are given a priority level of 9 for security screening and a priority level of 1 for everything else. An example of a passenger with a high security priority is illustrated in Fig.4.2.za. For all three layouts, passengers follow the same behaviour and patterns as described before.

Note that changing the priority levels in this simulation only affects how agents score their experiences, but this does not change their behaviour. The one exception is passengers with a low food priority will not go get food. Instead, passengers are given a default score for noticing the existence of the food/retail area, like the washrooms.

As a result, all passengers with the high security priority in the holdroom concourse just wait by the seating area. No passengers were observed wandering in the food/retail areas.

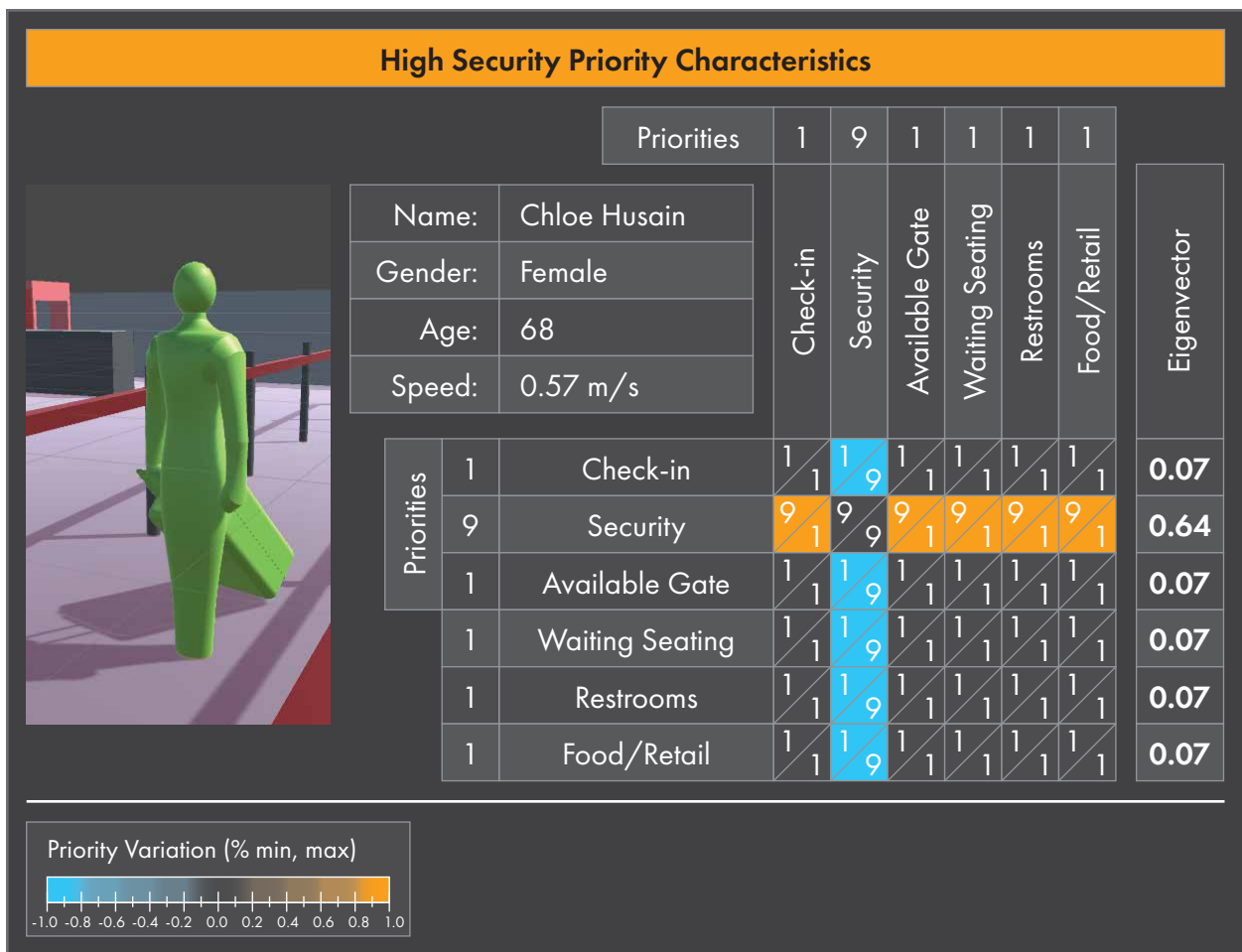


Figure 4.2.za: Assigned agent characteristics and high security priority matrix.

Equal Priorities Tests:

Agents are given an equal priority level of 5 for all domains. An example of a passenger with equal priorities is illustrated in Fig.4.2.zb. Passengers follow the same behaviour as the previous tests.

As mentioned for high security priorities, only food/retail behaviour is affected by priority levels. Since agents have a priority level of 5, all passengers were observed getting food, while waiting for departure in the holdroom concourse. Otherwise, no other significant differences were observed during these tests.

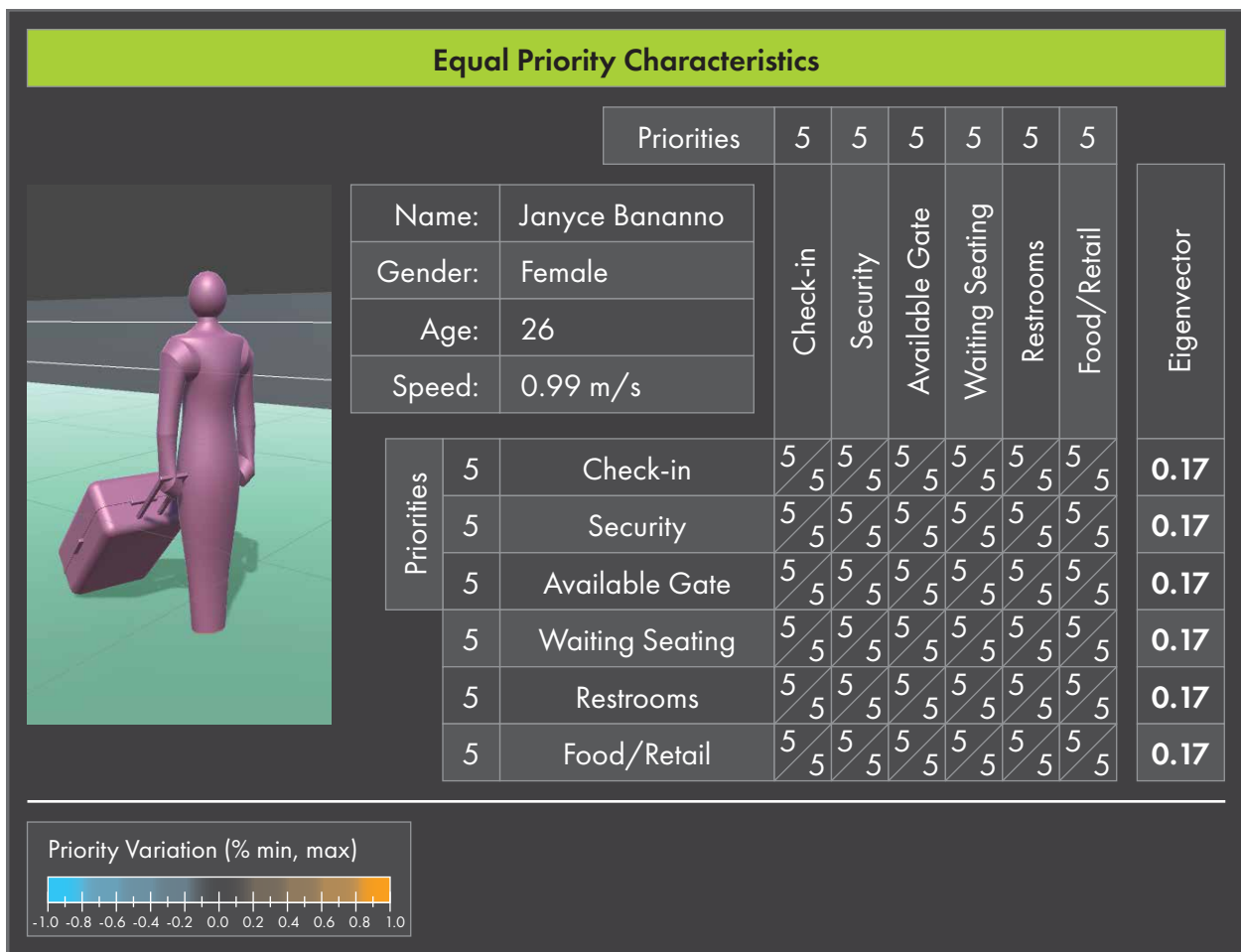


Figure 4.2.zb: Assigned agent characteristics and equal priority matrix.

Results:

The tests were conducted multiple times to get a sample size of 100 passengers for each condition. Together, this produces nine combinations based on the layout and agents' assigned priorities. The average architectural values are listed in Fig.4.2.zc:

Priority	Layout		
	Centre	Asymmetric	Perpendicular
Random	0.727	0.721	0.691
High	0.639	0.572	0.526
Equal	0.770	0.728	0.718

Figure 4.2.zc: *Average architectural value for all nine tests*

The test with the highest value is Centre-Equal, or when passengers had Equal priorities in the Centre layout, which was 0.770. The test with the lowest value is Perpendicular-High, or when passengers had High security priority in the Perpendicular layout, which was 0.526.

The distributions for architectural value are also compared. Firstly, based on layout design (Fig.4.2.zd-zf), and secondly, based on priority type (Fig.4.2.zg-zi).

The highest architectural values occurred when passengers had Equal priorities for all domains (Fig.4.2.zi). By contrast, the lowest architectural values occurred when passengers had High priority for security (Fig4.2.zh). When passengers have equal priorities for all airport domains, their scores are spread out evenly, so value is accumulated from multiple conditions. Whereas, when passengers only have priority for one domain, their scores are dependant on a single condition, which could lose value easily if they had a poor experience.

This is clearly demonstrated in the Perpendicular-High test. Many passengers spent time searching for security, so their value of the architecture is influenced by this experience. Additionally, passengers in the High tests did not prioritize other domains, like food/retail. Therefore, they did not see the value in going to get something to eat while waiting for their departure. If passengers do not interact with these conditions, then the conditions have no value for the people, despite the terminal design having these amenities.

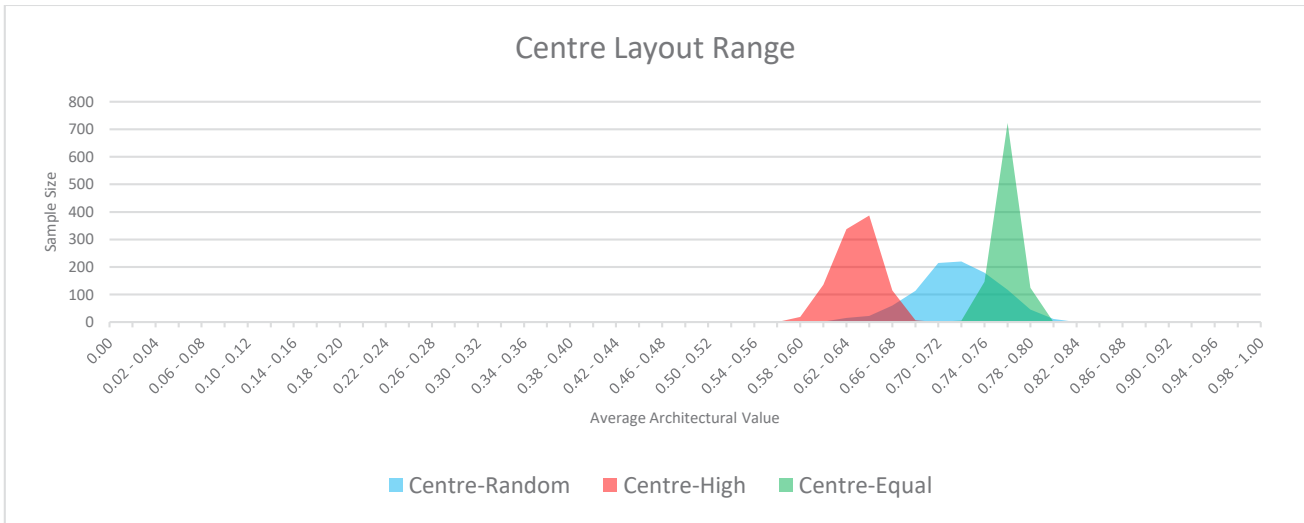


Figure 4.2.zd: Centre has highest distributions overall. Equal priority is the greatest and most concentrated.

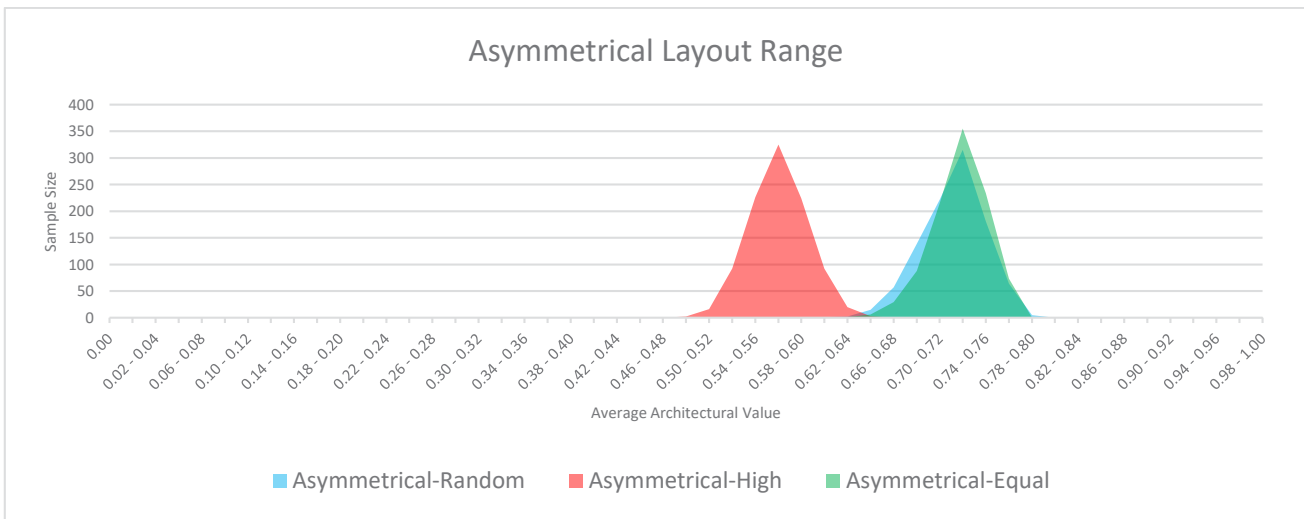


Figure 4.2.ze: Asymmetrical shows random and equal priorities are similar, and high security is lowest.

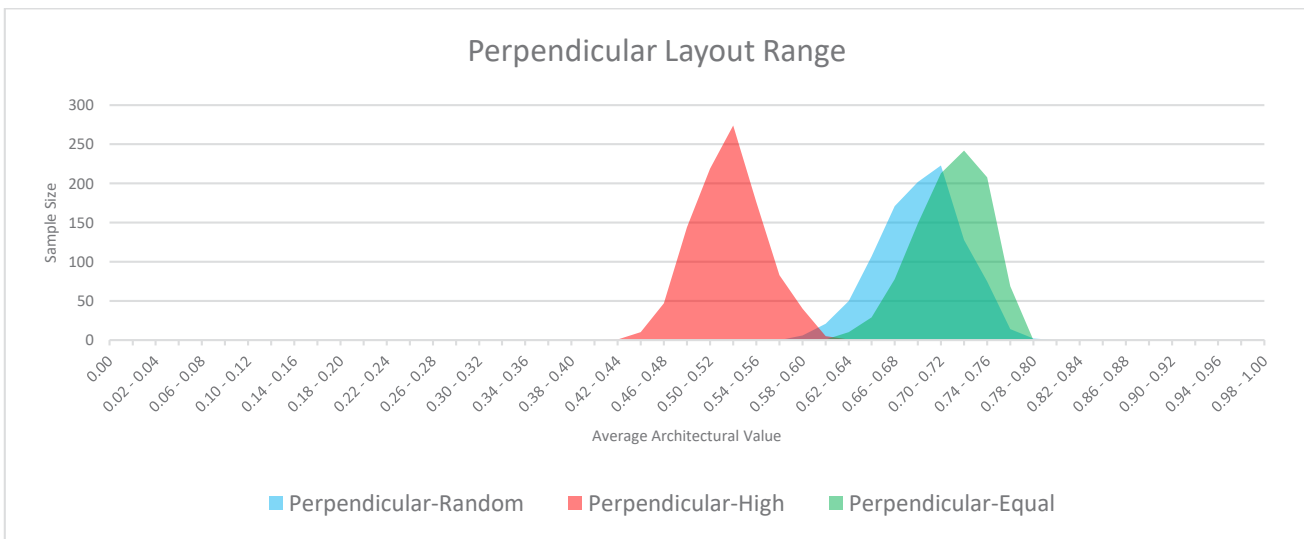


Figure 4.2.zf: Perpendicular has lowest distributions overall. High security priority gives the lowest values.

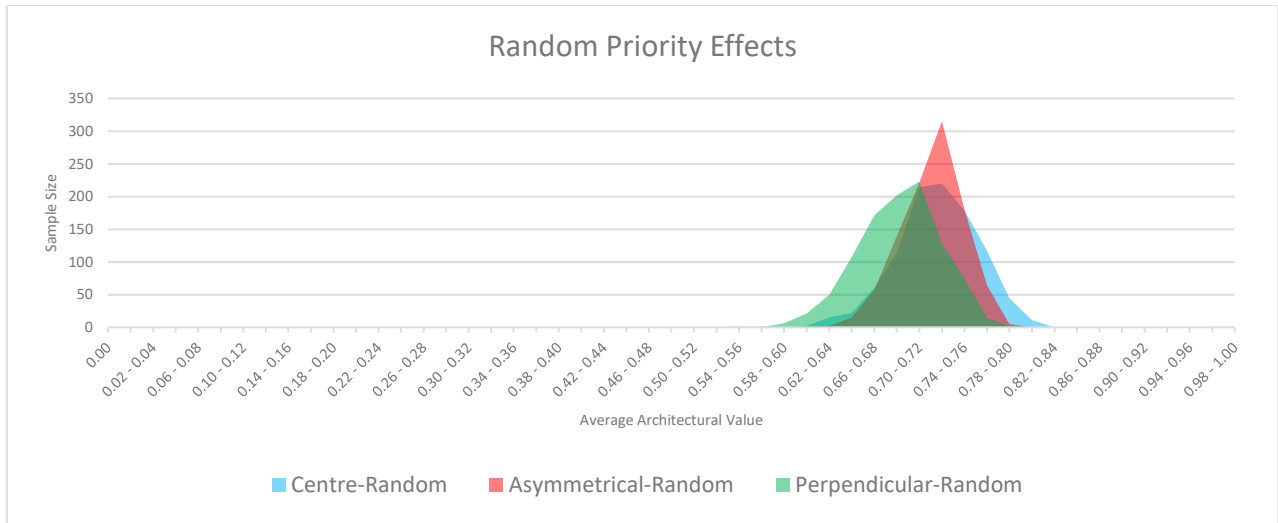


Figure 4.2.zg: Random priority distributions have narrow variance. Perpendicular has the lowest values.

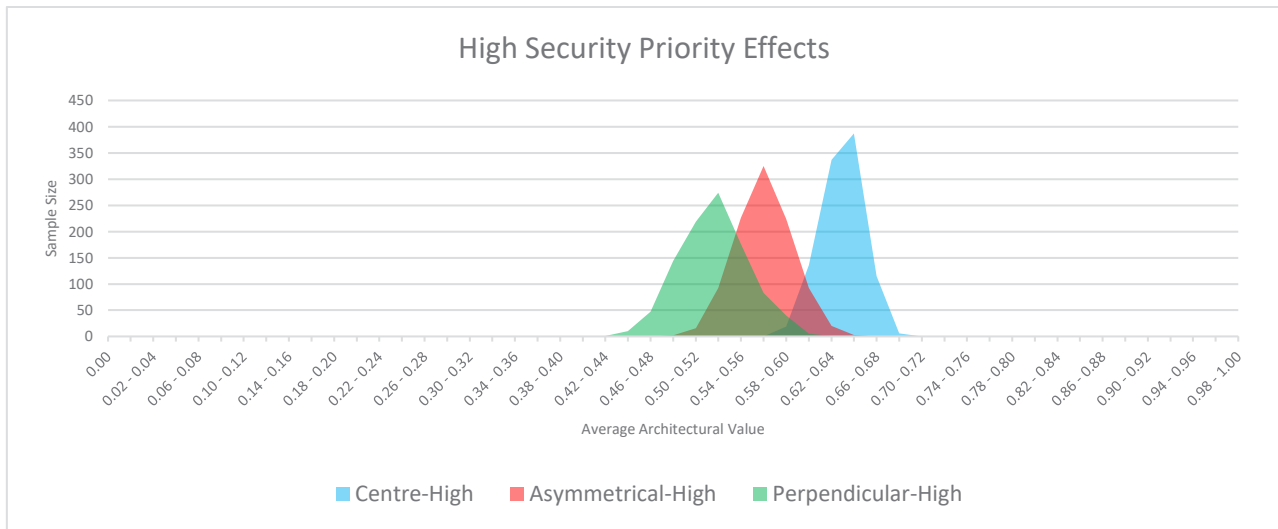


Figure 4.2.zh: High security priorities has wider variance. Centre is the greatest, perpendicular is the lowest.

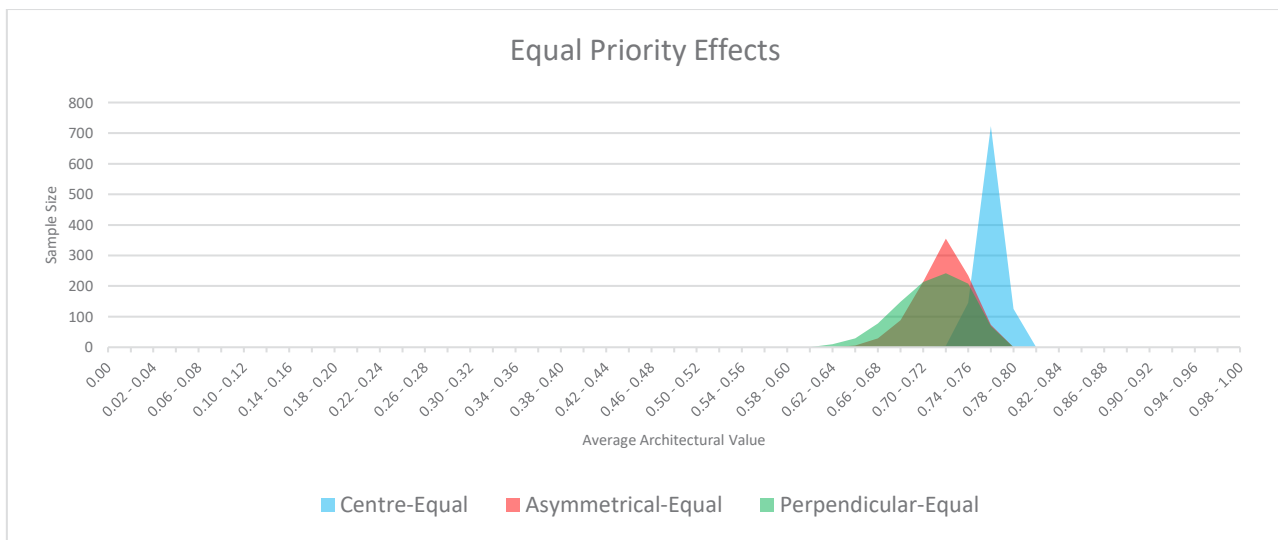


Figure 4.2.zi: Equal priority distributions have little variance. Centre is the greatest and most concentrated.

Chapter 4.3

Airport Tests

The following test compares two existing terminals to demonstrate how the agent's architectural values score in a real-world airport. The intension is to compare a known airport that has a high ranking against another airport which has a lower ranking. If the architectural value in the agent simulation is correct, then the score for each airport should correspond to its known ranking. The higher ranked airport should have a better architectural value than the lower ranked airport.

The two airports that are considered in this test are Singapore Changi and Toronto Pearson, which are the higher and lower ranked airport, respectively. Singapore Changi ranked 1st in Skytrax's World Airport Awards 2020 and 7th in AirHelp's Global Airport Ranking 2019, ^[1] while Toronto Pearson ranked 42rd and 108th, respectively (Fig.4.3.a-b). ^[2] Both Skytrax's and AirHelp's rankings are based on surveys conducted with people on their passenger experiences. The rankings considered a wide range of factors including Accessibility, Public Transit, Wayfinding, Check-in, Security, Immigration, Baggage, Flight Time, Staff Courtesy, Cleanliness, and Passenger Amenities. ^{[3][4]} Although the surveys were conducted differently for each company, they are similar enough to give confidence that Changi Airport has better passenger experience than Pearson Airport. Since architectural value is dependant on agent perception, it is expected that Changi has a higher architectural value than Pearson Airport.

1. "World's Top 100 Airports 2020". World Airport Awards, Skytrax, 2020. Accessed October 2020. <https://www.worldairportawards.com/worlds-top-100-airports-2020/>.

2. "Global Airport Ranking". AirHelp, 2019. Accessed October, 2020. <https://www.airhelp.com/en/airhelp-score/airport-ranking/>.

3. "Awards Methodology". World Airport Awards, Skytrax, 2020. Accessed October 2020. <https://www.worldairportawards.com/awards-methodology/>.

4. "AirHelp Score 2019: Global Airport Rankings". AirHelp, 2019. Accessed October 2020. https://static.airhelp.com/pdf/2019-airport-score/methodology_airhelp_score_2019__global_airport_rankings-en_us.pdf.

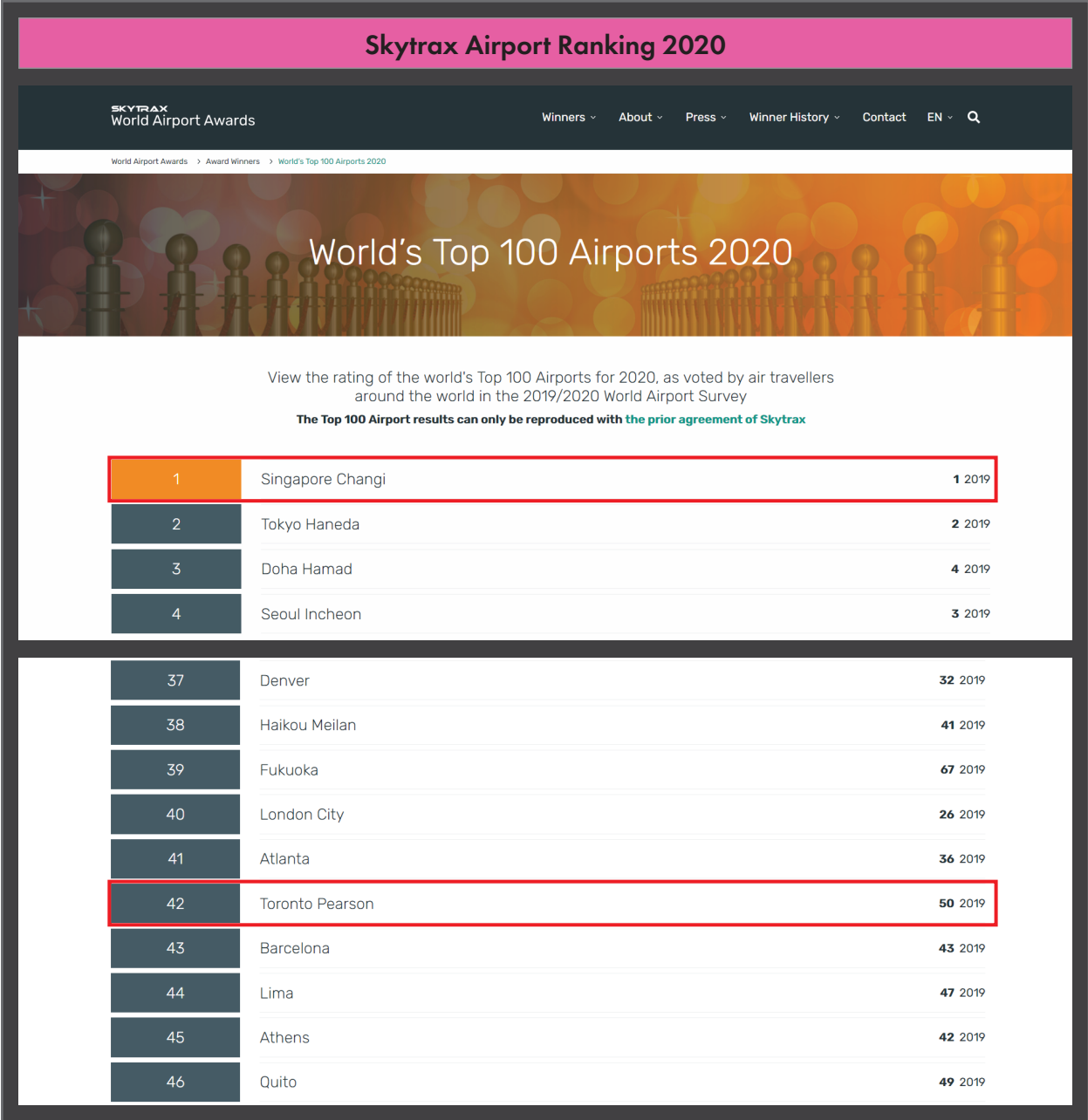


Figure 4.3.a: Skytrax's Airport Ranking from 2020 has Changi as #1 and Pearson as #42, based on a global airport survey of passenger experience, Skytrax (2021), highlighted in red by author.

AirHelp Airport Ranking 2019

Global Airport Ranking

#	Airport	AirHelp Score	On-Time Performance	Service Quality	Food and Shops
1	Hamad International Airport Doha, Qatar	8.39 /10	8.3	8.5	8.5
2	Tokyo International Airport Tokyo, Japan	8.39 /10	8.4	8.4	8.4
3	Athens International Airport Athens, Greece	8.38 /10	8.1	9.0	8.7
4	Afonso Pena International Airport Curitiba, Brazil	8.37 /10	8.4	8.4	8.3
5	Gdańsk Lech Wałęsa Airport Gdansk, Poland	8.35 /10	8.2	8.7	8.5
6	Moscow Sheremetyevo International Airport Moscow, Russia	8.35 /10	8.5	8.1	8.0
7	Singapore Changi Airport Singapore	8.27 /10	7.8	9.2	8.7
8	Hyderabad Rajiv Gandhi International Airport Hyderabad, India	8.27 /10	7.8	9.0	8.8
9	Tenerife North Airport Tenerife, Spain	8.26 /10	8.2	8.4	8.2
10	Viracopos/Campinas International Airport Campinas, Brazil	8.25 /10	8.4	8.2	7.9

101	Montréal-Pierre Elliott Trudeau International Airport Montreal, Canada	7.07 /10	6.7	7.8	7.5
102	Keflavík International Airport Reykjavik, Iceland	7.02 /10	6.2	8.6	8.0
103	Barcelona - El Prat Airport Barcelona, Spain	7.02 /10	6.1	8.3	8.4
104	London City Airport London, United Kingdom	6.99 /10	6.5	7.6	8.0
105	Seoul Incheon International Airport Seoul, South Korea	6.99 /10	5.8	8.8	8.8
106	Geneva Airport Geneva, Switzerland	6.98 /10	6.7	7.6	7.2
107	Palma de Mallorca Airport Palma de Mallorca, Spain	6.98 /10	6.4	7.8	7.9
108	Toronto Pearson International Airport Toronto, Canada	6.97 /10	6.4	7.7	7.8
109	London Stansted Airport London, United Kingdom	6.94 /10	6.4	7.7	7.7
110	Canadian Airport Quebec, Canada	6.94 /10	6.6	7.6	7.3

Figure 4.3.b: AirHelp's Airport Ranking from 2019 has Changi as #7 and Pearson as #108, based on an average of performance factors, AirHelp (2021), highlighted in red by author.

Singapore Changi Airport is the primary international airport in Singapore. It served over 60 million passengers in 2019 and is one of the busiest airports in Asia. It has 4 active terminals (1, 2, 3, and 4) and plans for a 5th terminal over the next decade. It serves as a major hub in Asia for international flights for both passengers and cargo.

Toronto Pearson Airport is the primary international airport in southern Ontario. It served over 40 million passengers in 2019 and is the busiest airport in Canada. It has 2 active terminals (1 and 3) and serves as a major hub for international flights entering the United States providing pre-clearance for all departing and connecting passengers.

Although Changi and Pearson are both prominent international airports, there are differences in the way each airport is organized, how flights are handled, and what amenities they provide. The scope for this test only focuses on the layout of essential terminal spaces relating to the pre-departure process. It tries to minimize differences in operations and culture by only focusing on the following key spaces: check-in, security screening, food/retail, and washrooms. This test simulates international passengers departing from Terminal 1 for both airports. The simulation starts when passengers enter the front doors of the terminal. The simulation ends after passenger exit through the gate portal, which is before the concourse piers, but after security. (Fig.4.3.c).

Due to the size of the terminals and the limited capability of the agent model, the simulation is cut short before the piers to the gates. The length of the piers become too large for the agent model to manage. Large models have caused the simulation to experience time lag and sluggish performance when trying to calculate agent navigation over this distance. Ending the simulation before agents enter this area reduces the geometry the simulation needs to consider. Since both airports are cut off before the piers, they end at a similar point in the departure process. Therefore, both terminal models are compared under similar conditions.

Airport Tests

Purpose

To see if architectural value can differentiate between real-world airports with higher and lower passenger experience ranking, relatively.

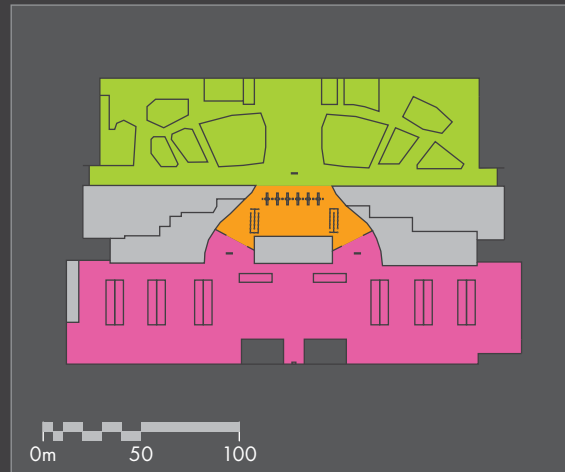
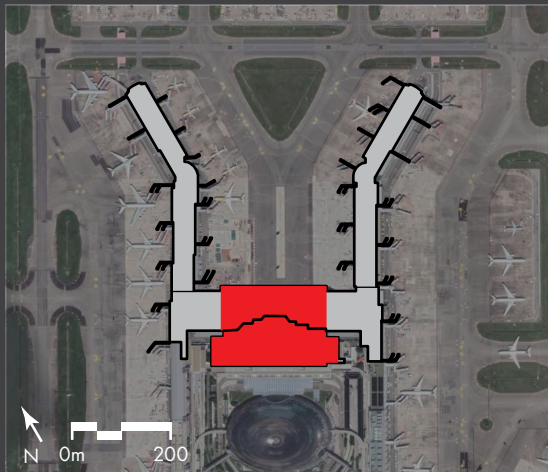
Conditions

- Singapore Changi with a higher rank vs. Toronto Pearson with a lower rank
- International departure (non-USA), terminal 1, check-in and security
- Population of 50 passenger with random priorities
- Cut-off before gate piers and holdroom concourses

Terminal 1 Scope

Simulated Check-in Layout

Singapore Changi



Toronto Pearson

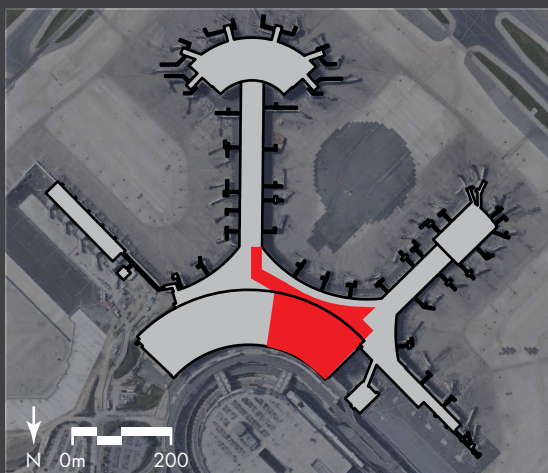


Figure 4.3.c: Setup and conditions for the airport tests.

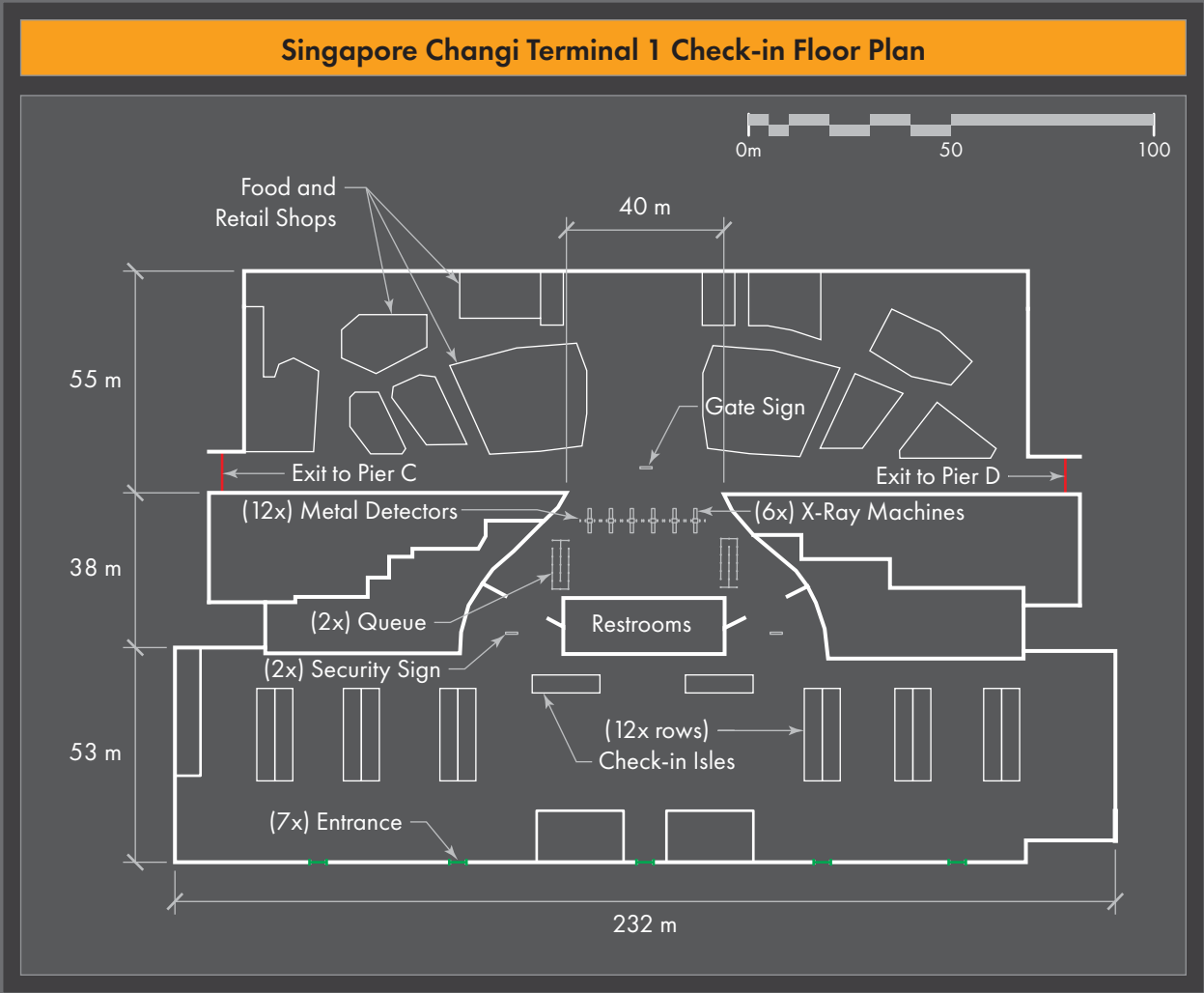


Figure 4.3.d: Simulated floor plan for Changi terminal 1.

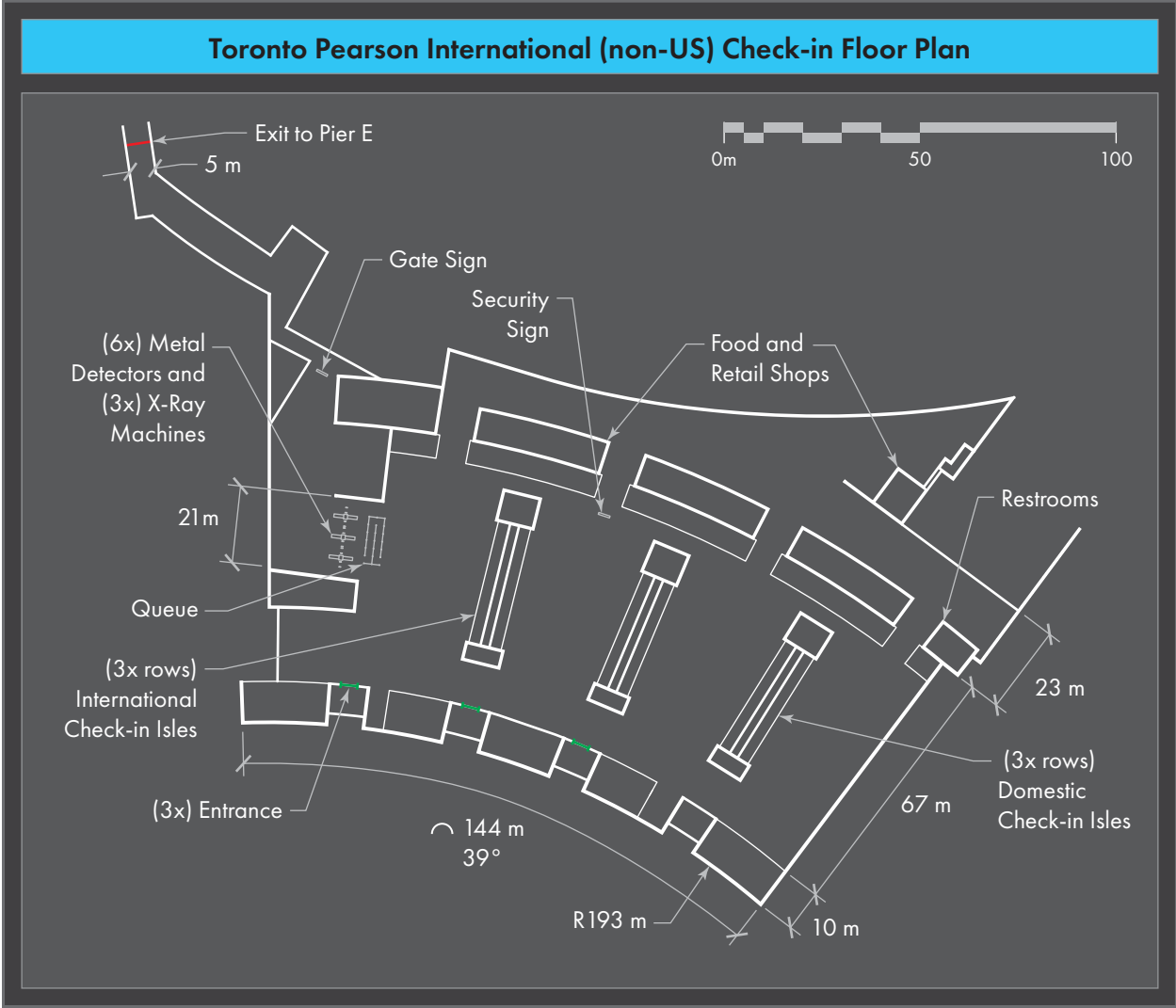


Figure 4.3.e: Simulated floor plan for Pearson terminal 1, international departure.

Singapore Changi Layout

The simulated departure area of Changi terminal 1 is a symmetrical 230 *m* wide hall with a centralized security screening area, which opens into a retail courtyard (Fig.3.d).

There are seven entrances evenly spaced along the south wall of the terminal. In the check-in area, there are 12 parallel isles of counters with two additional rows in the center. Behind these counters is a public washroom facility in the middle of the terminal. The security area is a curved Y-junction that flanks either end of these washroom facilities. Passengers can approach security from either side of the check-in area, since there is dedicated queuing on each side, which are marked by signs.

After screening, the security area opens into a retail courtyard that contains numerous shops, food stalls, and lounge areas. At the front of the courtyard is a wayfinding sign that points passengers to the left for pier C gates and to the right for pier D gates. The simulated area ends to the far right and left sides of the retail courtyard, which is marked by an exit portal.

Toronto Pearson Layout

The simulated departure area of Pearson terminal 1 is a 140 *m* segment of a circular-arc'd space, which focuses on the non-US international check-in area, security screening, and adjacent retail and food court (Fig.4.3.e).

There are 3 main entrances dedicated for international passengers along the inside radius of the terminal. In the check-in area, there are 6 parallel isles of counters, with only the 3 rows on the left dedicated for international flights. Beyond the check-in area is a food court and retail area on the outside radius of the terminal, with washroom facilities along the right end of the food court.

The security screening area sits on the left side perpendicular to the check-in isles, with a queue marking the front. There is also signage placed between the check-in isles, directing passengers to the left for security. After screening, the security area bends up into a narrow corridor, which is indicated by a sign for pier gates E and F “hammerhead” concourse. The simulated area ends at this corridor before it reaches the pier, which is marked by an exit portal.

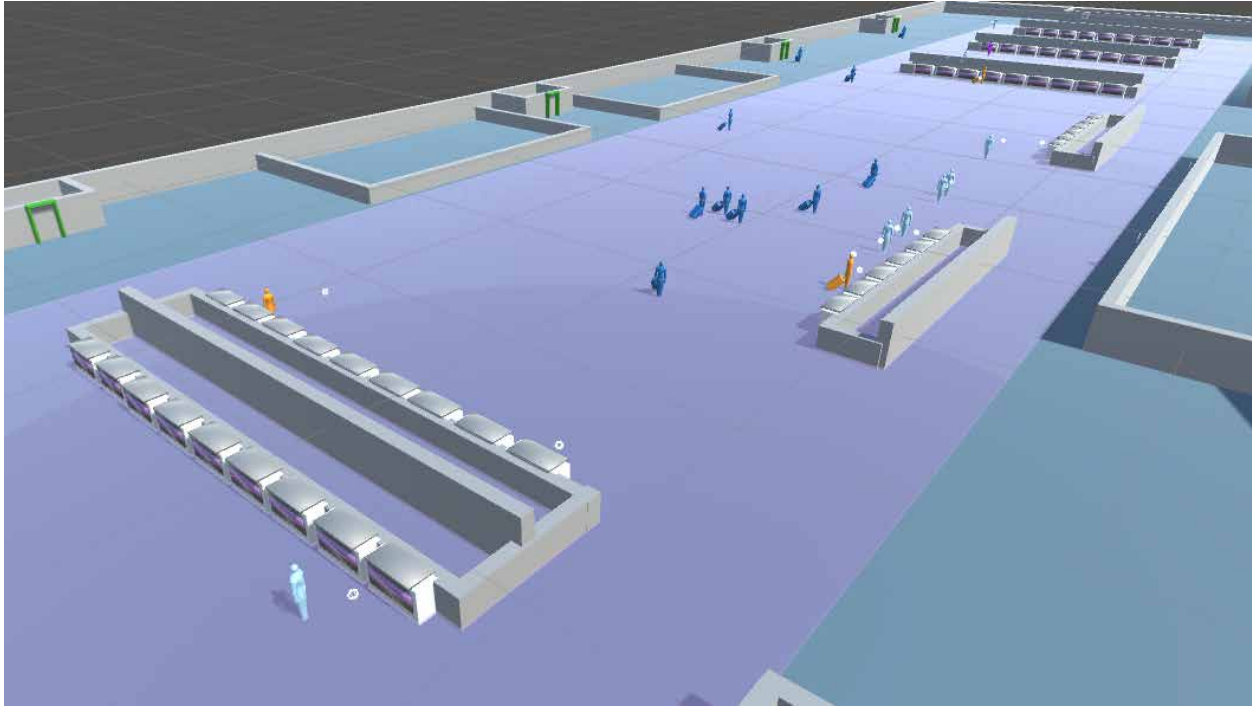


Figure 4.3.f: *Passengers in Changi going through the check-in area.*

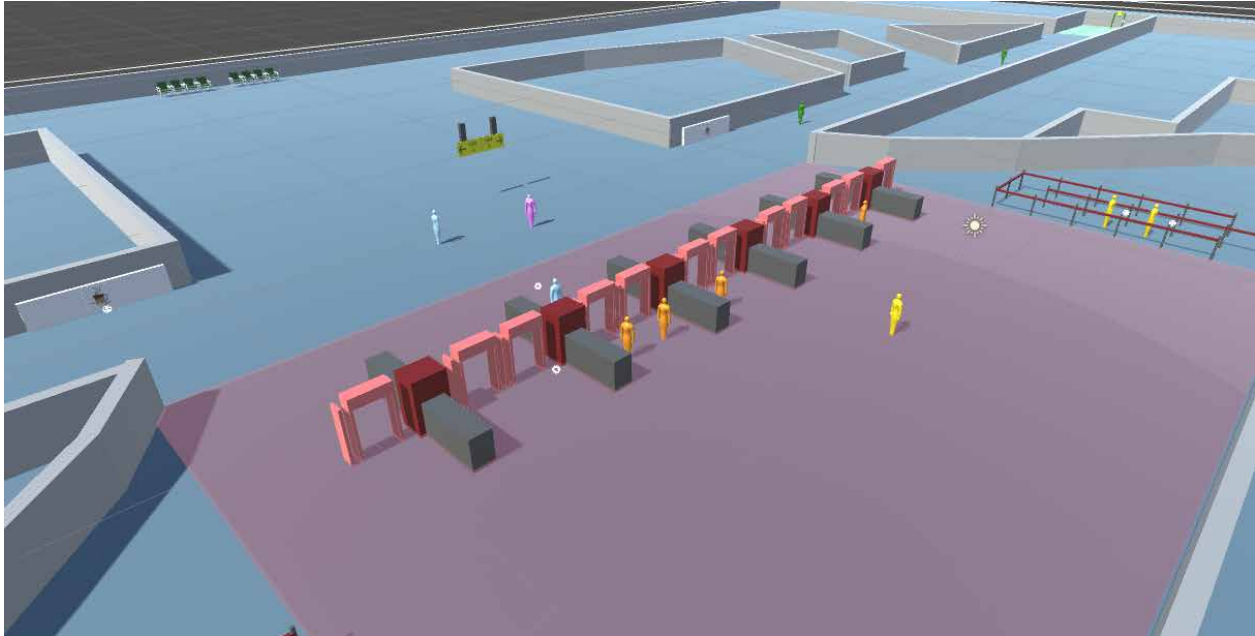


Figure 4.3.g: *Passengers in Changi going through security into the retail courtyard.*



Figure 4.3.h: *Passengers in Pearson going through the check-in area.*

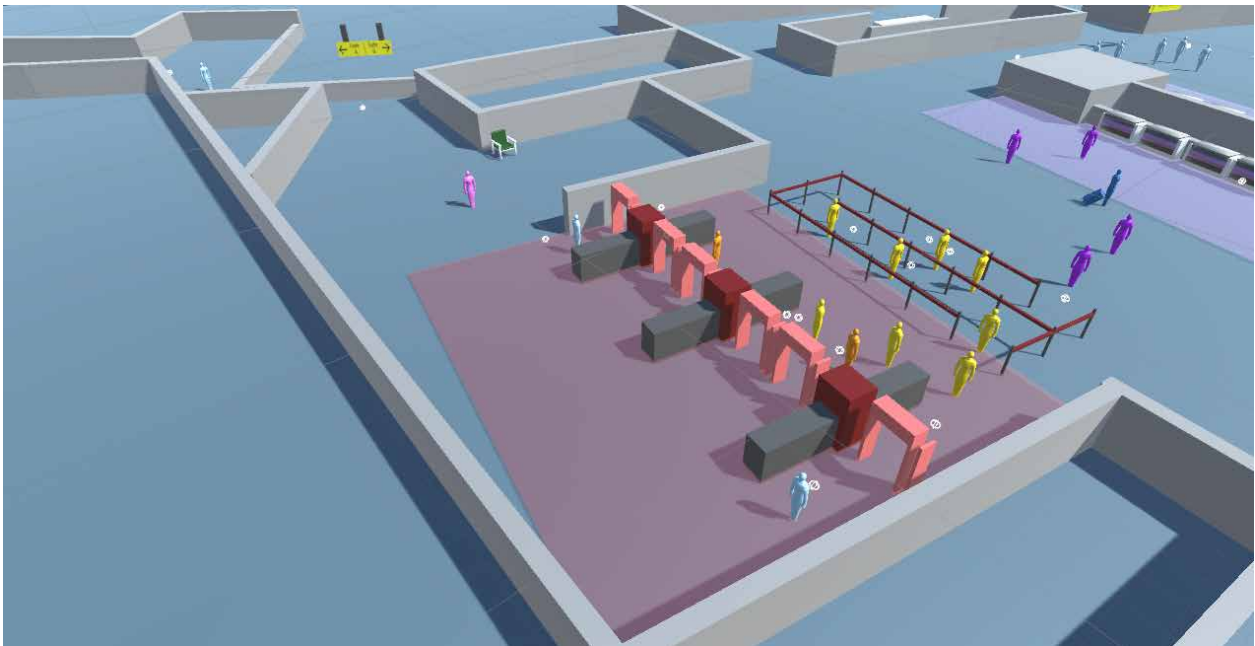


Figure 4.3.i: *Passengers in Pearson going through security.*

Results

The thesis ran multiple trials of the simulation with 100 passengers for both layouts. The average architectural value for all passengers is 0.802 for Singapore Changi and 0.420 for Toronto Pearson. The population distributions from these tests are illustrated in Fig.4.3.j-k. The distribution of passengers in Changi is from 0.64 to 0.92, with most passengers scoring between 0.76 and 0.78. As for Pearson, the distribution of passengers is as low as 0.26 and as high as 0.80. Although, most passengers were between 0.40 and 0.42. This means that passengers in Pearson have a much wider range of experiences than Changi, which gave passengers similar experiences.

Fig.4.3.l compares the sample mean distribution between Changi and Pearson, which uses 1000 trials and $n = 10$ samples for each. This illustrates the range of values that are likely to occur in each airport under the given conditions. The average architectural value for Changi, ranges between 0.74 and 0.88. It approaches a normal distribution with a mean of 0.802 and a standard deviation of 0.0166, as the following,

$$C(x) = \frac{1}{0.0166\sqrt{2\pi}} \exp\left(-0.5\left(\frac{(x - 0.802)}{0.0166}\right)^2\right).$$

Whereas Pearson ranges between 0.32 and 0.54. It approaches a normal distribution with a mean of 0.420 and a standard deviation of 0.0289, in the following form (Fig.4.3.m),

$$P(x) = \frac{1}{0.0289\sqrt{2\pi}} \exp\left(-0.5\left(\frac{(x - 0.420)}{0.0289}\right)^2\right).$$

It appears Pearson has a much lower score than Changi because of the location of the amenity spaces, which are the retail, food, and washrooms. Most passengers in Pearson did not walk beyond the check-in area to retail/food court. Instead, practically all passengers went straight to security screening. Since there were no amenities in the rest of the corridor, passengers in Pearson did not get a second chance to interact with them if it was their priority.

Additionally, passengers in Changi have a clear view of both washrooms in the center and food/retail beyond security. Although Changi does not have any retail between check-in and security, the retail courtyard after security is practically impossible to miss. Whereas passengers in Pearson do not have any amenities until they reach the gate holdroom if they did not see the food court.

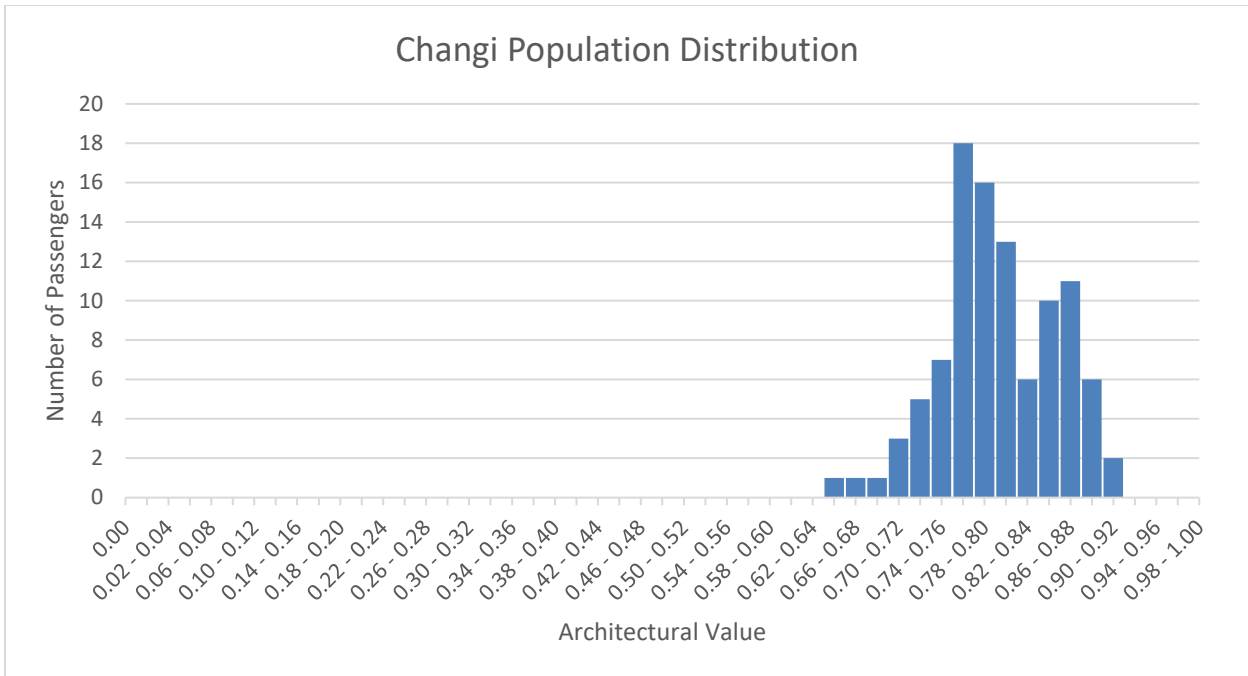


Figure 4.3.j: Population distribution for Changi gives an average value of 0.802.

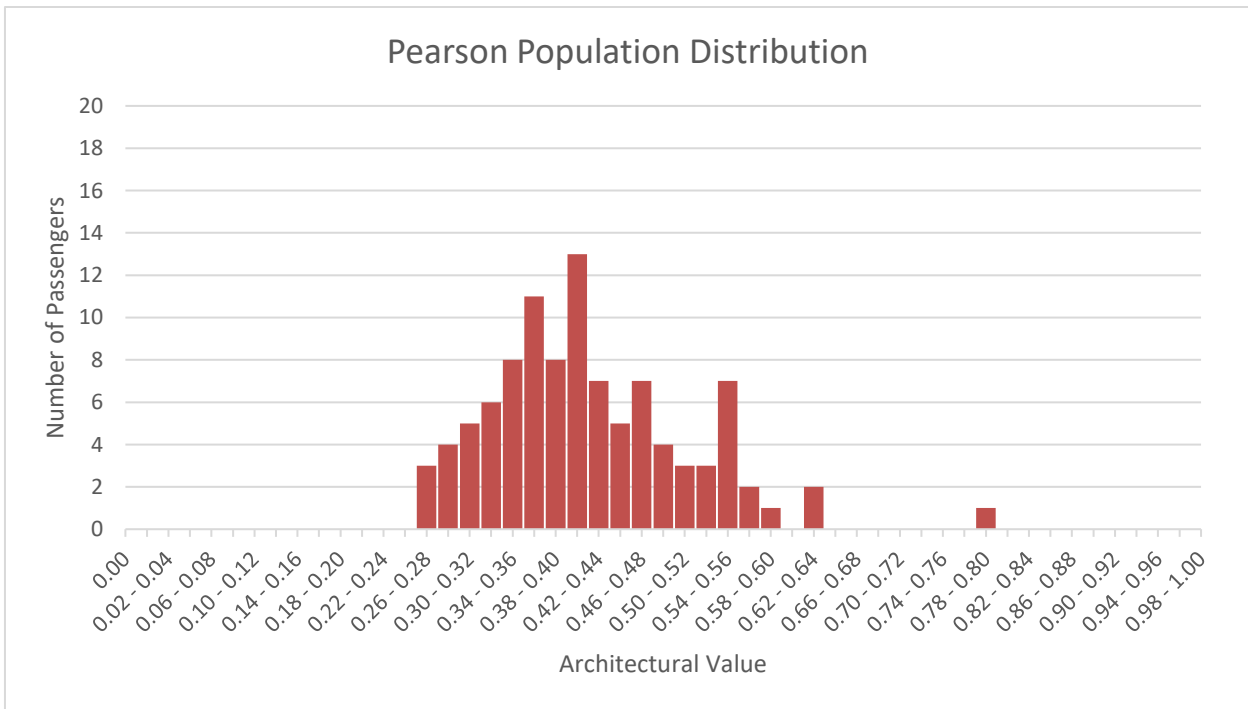


Figure 4.3.k: Population distribution Pearson gives an average value of 0.420, and has a wider spread than Changi.

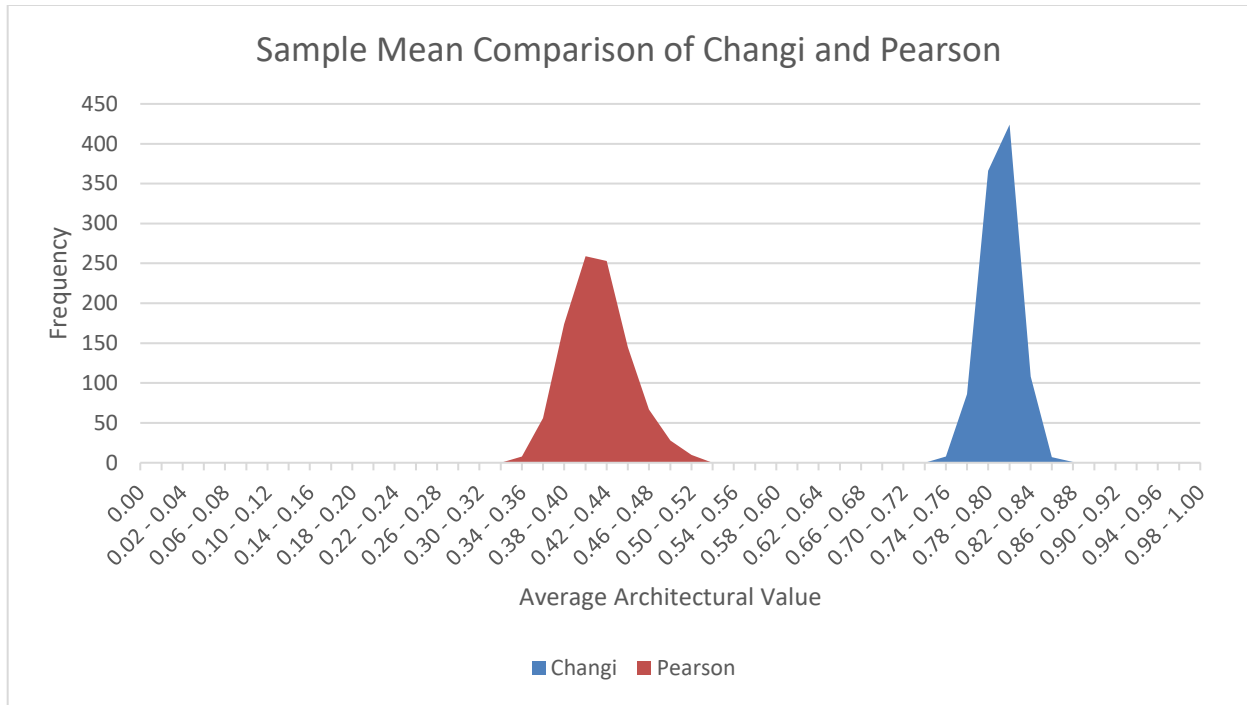


Figure 4.3.l: Sample mean distributions (trials = 1000, $n = 10$ samples) for Changi and Pearson illustrates the range of values each airport is likely to see.

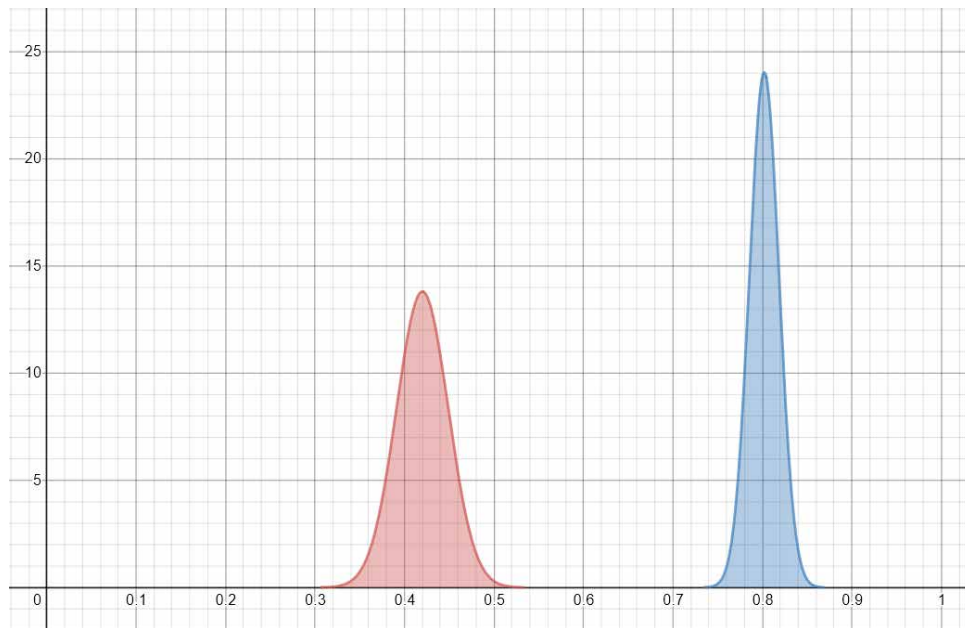


Figure 4.3.m: Equivalent normal distributions for Changi (blue), $N(0.802, 0.0166)$, and Pearson (red), $N(0.420, 0.0289)$, as continuous PDFs, given an infinite number of samples.

The architectural value function is dependent on passengers' priorities. Since food/retail and washrooms make up two of the six priorities, it can have a large impact. The thesis believes if agents have lower food and washroom priorities (or different priorities entirely), then the difference between Changi and Pearson would not be as high. Fig.4.3.n breaks down architectural value into the average values for each passenger priority. It illustrates that passengers in Changi observed every non-processing domain (food/retail, restrooms, and seating area). Whereas only a few passengers in Pearson came across food/retail and restrooms.

On average, Changi has better value for check-in and reaching the gate areas (exit portal). The higher value for check-in shows that passengers maintained more consistent visibility and did not need to wander as much as passengers in Pearson. Likewise, passengers in Changi had an easier time heading towards the gate, since passengers in Pearson had to walk through the narrow corridor. However, Pearson had higher value than Changi in security screening. This may be due to security being a lot closer for passengers in Pearson. Whereas passengers took longer to find the centralized security in Changi.

In summary, the agent simulation finds Singapore Changi has a higher average architectural value than Toronto Pearson. This confirms the agent simulation can correctly differentiate airports with higher and lower passenger experience ranking, respectively.

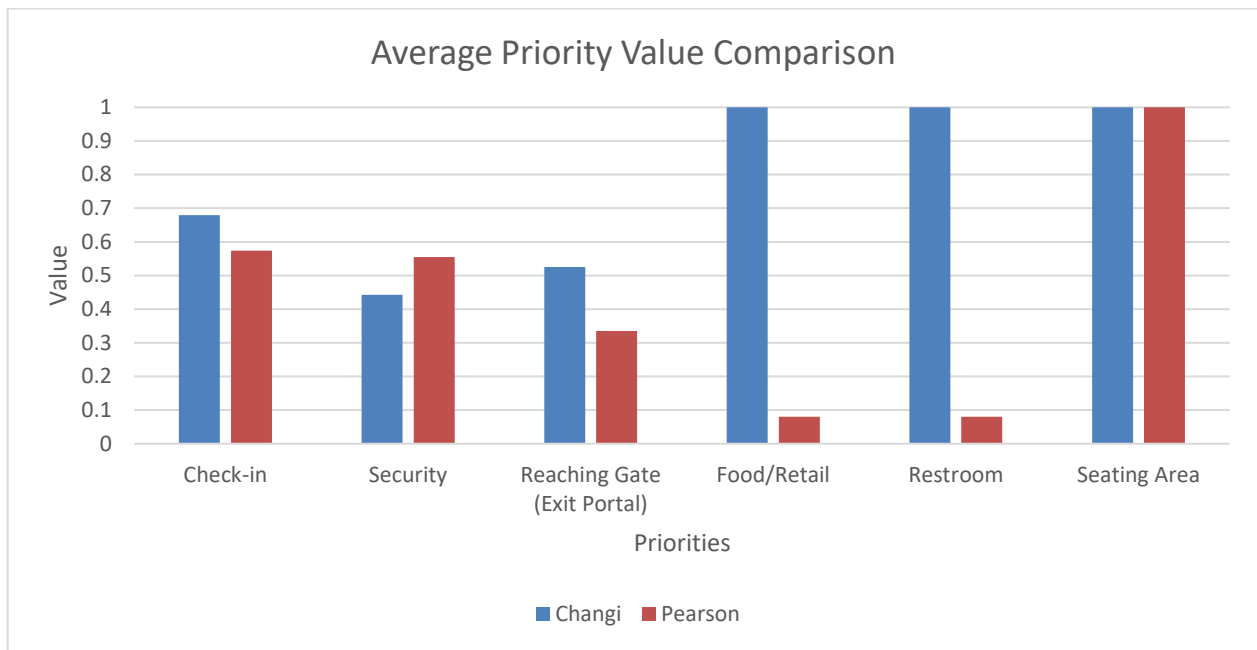


Figure 4.3.n: Average values for each passenger priorities. Note for these tests, processing domains are scored on level of interaction, whereas non-processing domains are scored if observed.

Part 5:

Conclusion

Part 5 discusses the results of this thesis, an ideal version of the agent simulation, impacts to the architectural profession, and plans for future research. Chapter 5.0 begins with a summary of the results and findings from simulation testing. Chapter 5.1 talks about the minimum components for an ideal architectural simulation, and how the agent models could be improved. Chapter 5.2 discusses how simulations could change the architectural design process and the built environment. It also talks about long-term impacts and risks of relying on simulation for design. Chapter 5.3 summarizes topics that would benefit from future research, and ideas that could be the basis for other theses. Finally, chapter 5.4 closes with a summary of the overall conclusions and final remarks from this thesis.

Chapter 5.0

Results and Findings

A major result of this thesis is the creation of a practical agent simulation for testing architectural layouts. The thesis's model builds from existing methods for simulating airport terminals, like discrete-event and statistical modelling. It incorporates agent perception to provide feedback of people's decision making in architectural spaces, and it uses prioritization as a way of quantifying architectural value. This is different from previous crowd simulations that test the built environment, which either focuses on modelling people as a process flow or modelling people's behaviour in emergency evacuations. Agent navigation in these models assumes people already know where they are going, which does not consider how people use the surrounding architecture to inform their decisions. This makes it difficult to test how well an architectural layout aligns with daily airport interactions, before the terminal building is built.

The hypothesis states that differences in a simulated architectural layout of an airport terminal can be quantified based on agent interactions. The thesis demonstrates that this is possible if agents are given sufficient perception of the surrounding environment, and if these agents can only rely on their perception to complete a given task. The thesis concludes that this type of simulation is capable of approximating real-world airport interactions, within a statistical certainty.

The results of testing show that the thesis's agent simulation is capable of basic agent behaviour. It can differentiate between certain architectural layouts based on a statistical distribution. However, the agent simulation is limited when it comes to modelling dense crowd flows, and some airport processes, like queuing. Additionally, the architectural value of a layout can vary significantly depending on agents' assigned priorities and airport domains.

Tests in chapter 4.0 validated basic simulation components against an established standard for evacuation simulations. Three out of the four tests that were conducted were successful. The simulation was able to demonstrate that agents can walk at realistic speeds, and that agents can navigate around walls. The simulation was unsuccessful at demonstrating exit flow rates through a crowded doorway. Densely packed agents were seen clumping together due to proximity issues, which caused spikes in flow rates that exceeded realistic behaviour. These

issues were minimized for further tests by reducing the maximum number of agents. This type of crowd interaction could be better studied to improve agent behaviour for further development.

Tests in chapter 4.1 demonstrated that the thesis's perceptive agent model can be influenced in specific architectural conditions, which is not typically explored in existing simulations. Firstly, the thesis shows that agents who rely on perception navigation can be influenced by the environment to take a different path than agents using direct navigation. As a result, perceptive agents may not always take the shortest path, which can provide different architectural experiences in the same space. The second test shows how agents can correctly differentiate the visibility between walking into a wide space and a narrow space using their visual field of view. This means that perceptive agents can simulate a changing spatial awareness, which provides feedback of different geometric conditions. The third test shows how agents can follow different behaviour in the same space, if they prioritize different things. This includes interacting with non-processing aspects of an airport like food and retail areas. As a result, agents that have different priorities in the same space will read the environment differently, which reflects their interpretation of the architecture.

Tests in chapter 4.2 explored generic terminal layouts with different security screening locations, representing possible design iterations for departing passengers. The thesis suggested that, if only a single space was changed, and all other conditions were the same, then the difference in agents' architectural values would represent those layout changes. A basic terminal was tested based on three security screening alignments: centre, asymmetrical, and perpendicular. The thesis also considered how the terminal's architectural value was affected by agents' assigned priorities: random, high security, and equal. Testing showed that the terminal's architectural value distribution was lower, on average, if passengers had difficulty finding the security area, which occurred in the perpendicular layout. These differences were most pronounced when passengers had a high security priority, since their experience of the terminal was more dependent on that one airport domain. Overall, the range of architectural values has greater variance when agents have random priorities than agents with equal priorities. This indicates that, if passengers prioritize a wide range of aspects in an airport, then architectural changes in a single area do not have a significant effect on the overall passenger interaction. Fundamentally, passengers are more likely to see architectural value in areas that they prioritize.

Tests in chapter 4.3 checked if the thesis's simulation of architectural value could differentiate between a good and bad airport according to an established airport ranking, relatively. The good and bad airports were Singapore Changi and Toronto Pearson, which were ranked 1st and 42rd, respectively. The test was conducted for departing international passengers in similar areas of each terminal building, within basic check-in, security, and retail areas. The results of testing concluded that Singapore Changi has a higher architectural value than Toronto Pearson, on average. Passengers in Pearson experienced a wider distribution of values than Changi, whose passengers had more consistent interactions. The difference in value was primarily caused by passengers in Pearson not interacting with non-processing domains, since these areas were located on the opposite side of the terminal from the security area. Whereas all passengers in Changi were able to interact with these domains, since its security opens directly into a retail courtyard. If agents were assigned different priorities, the thesis believes the difference in architectural value would not be as high. Overall, this testing showed that the thesis's calculation of architectural value is capable of aligning with passenger's experience in a real airport. As a result, this type of analysis could represent how well passengers will interact with the spaces in any terminal design.

Chapter 5.1

Ideal Models

Although the thesis's agent simulation has achieved some success, it is far from ideal and is not sufficient enough to be used in real practice. However, the thesis has investigated several attributes that would be beneficial for any architectural analysis tool.

Minimum Requirements

The thesis believes there are three key elements that an agent-based tool will need to better evaluate architectural spaces. This includes realistic crowd behaviour, perception navigation, and statistical value functions.

Crowd Behaviour: Agents in an architectural simulation need to function at least as well as existing evacuation crowd modelling software. This includes crowd dynamics, social behaviours, interaction with physical conditions, and possibly, behaviour for emergency situations. Crowd dynamics can include how people behave in large groups, how people respond to congestion, and how different flow rates propagate through a space. Social behaviours include how people respond to other people near by, family dynamics, making sure people represent real demographics, and a diverse range of human characteristics. Physical conditions include basic movement in a building, movement between multiple floor levels, and interactions with objects and equipment. Emergency situations include evacuation patterns, stress or panicked behaviour, and dynamic choices.

Perception Navigation: Agents in the simulation must navigate with perception, instead of the shortest path, so that agents can give feedback of architectural conditions, and secondly, to better represent how people interact in unfamiliar buildings. The thesis explored A* pathfinding, however, other options can include vector forces, or similar physics-based models. Agents' navigation needs to have some form of decision-making. People should not be modelled as flow items or passive objects. Ultimately, the decision-making process will inform how people read architectural conditions.

Statistical Value Functions: Objective valuation of architectural space should be based on some form of human interaction, which uses probability and statistics to approximate uncertain

conditions. The thesis used prioritization, a part of the *analytical hierarchy process*, to quantify passenger experiences. However, there are other *multi-criteria decision analysis* that use similar mathematics or logic to make objective decisions within a complex system. These techniques usually involve some form of ranking, weighting system, or vector analysis to evaluate choices. Additionally, since a decision-making process has a level of uncertainty, statistics must be used to estimate the range of values that are likely to occur. This will also help reduce the computation if there are many architectural conditions to calculate.

Agent Model

The function of thesis's agent model could also be improved. As shown during testing, the agent simulation was not able to realistically model dense crowded spaces. An ideal agent should, firstly, match real human walking speeds, and avoid walking through walls, objects, or other agents. They should be able to detect proximity to other agents and slow down in crowded spaces. If agents do encounter crowded areas, they should try to look for open areas to walk. Likewise, agent navigation should be constantly updated so that they can respond appropriately to new information. If an agent's environment changes, their navigation should change too, instead of blindly walking to their original target.

Additionally, not all people would respond to the same information. Agent should model how people respond in different situations. This includes adapting behaviour to different priorities, or human needs like hunger. People who are hungry should be more likely to get food whether it is their priority or not. The agent model should allow people to adapt their behaviour or priorities over time. For example, if people's mood changes, or if they become stressed, then that should reflect their choices. People going through an airport who are impatient may be more likely to become stressed, and as a result, would be more critical of poor wayfinding or the lack of amenities.

The agent model could also consider different groups of people. This includes people who travel in large groups with families or friends, and the dynamics of staying together. This would also affect how people are waiting in lines and occupying amenity spaces. Additionally, children and dependants have entirely different logic in a family group. Although, this may require a completely separate agent model specifically for simulating child behaviour.

Other Architectural Environments

The thesis has been focusing on the architecture of airport terminals for passenger processing. However, there are many areas in an airport that an architect might be responsible for designing. This may also include back-of-house offices, lounges, transit halls, baggage handling areas, or other non-passenger facilities. The intension of this thesis was to provide a basic illustration of the core concepts for one architectural condition, which could be expanded into other domains or situations.

For example, a similar perceptive agent model could be made for baggage workers. This could be used to test how these workers make decisions based on the bags they perceive while they are moving them between aircraft. Likewise, another agent model may be used to test passenger's interaction in airport lounges based on the decisions they make in these areas. This type of testing may be useful for airlines to improve passenger experience.

In addition to airports and transport facilities, the thesis believes this type of agent modelling could be generalized for testing other building types. This would focus on buildings that deal with a lot of people, like hospitals, schools, or community centres.

For example, the design of hospitals could use agent modelling to test patient interactions in waiting rooms, clinics, or operations. As this thesis has shown, agents could be given knowledge of common hospital domains, and asked to complete a healthcare checkup only using their perception. This can indicate how well the layout of the hospital is used to complete that checkup.

Likewise, the design of a university could use an agent model to simulate students going between classes or interacting with university facilities. Students could be assigned different priorities of what aspects of the university they believe is important. The simulation could then provide a distribution of values for each university domain that students were able to engage with.

Fundamentally, the ideal version of this thesis's agent simulation could be reconfigured for any building type. The agents could be reassigned different priorities, or arbitrary value functions that correspond to new environments or tasks they need to complete.

Chapter 5.2

Impacts

The ideas covered in this thesis include objective design testing of architectural spaces and the functional tool of agent simulations. These have the potential to impact two areas of architecture: the design process and the built environment. The design process includes project planning, testing iterations, and client interactions. The built environment describes the construction of airport terminals and similar public facilities. In general, objective layout testing can organize areas in a terminal closer to how passengers naturally act in an airport. Specifically, a terminal that aligns with passengers' intuition allows people to walk through areas without having to question what they are doing. In the long-term, this thesis expects development of scientific design practices, shift in the responsibilities of architects, and automating architectural design. There is also a risk of optimizing for passenger priorities that will change in the future, or that have unintentional effects on human well-being.

Design Process

Gathering Data:

The first impact to the design process is the influence on project planning. Before designs can begin, airport developments will need information about the type of people in the terminal. It is not only enough to design a terminal based on the number of aircraft movements or passenger flow rate. In addition to these aspects, planners will need to understand what passenger preferences are, things that people consider important, and things that people do not care about in their airport experience. These factors will determine what simulations are testing for.

Many of these factors can be influenced by cultural backgrounds. For example, business passengers in Asia may have more tolerance for a noisy environment, while passengers in North America may prefer to have quiet locations to work while they are waiting.^[1] Although, cultural differences cannot be generalized this way, it is important to consider what information a simulation is using to model passenger behaviour. As mentioned, the way people behave in

1. "Can cultural differences impact passenger satisfaction?". Analysis, Airport Technologies. Updated December 7th, 2018. <https://www.airport-technology.com/features/passenger-satisfaction-in-airports/>.

different spaces has an impact on the performance of the architecture. Getting relevant passenger data is a critical first step for objective testing.

Design Testing:

The second impact to the design process is the need for iteration testing. During the design phase, a team might put together a floor plan proposal of an airport terminal. If the architects made good design decisions, then things like the security screening area would be well spaced out, the signage would be in visible locations, and there would be a good distribution of amenities. To check if the layout meets design expectations, it must be validated. In addition to checking crowd flow density and queue times, the design would also be tested for architectural layout performance.

The thesis expects there to be a team of *technical architects* who would be responsible for analyzing designs for layout performance using an agent-based tool. Designers would give their latest terminal model to this team to assess what architectural value it has. After running through simulations, these architects would identify what areas of the terminal have good spatial performance, and areas that have poor spatial performance, based on the agent behaviour. Designers would then take back the terminal plan and adjust the layout to get a better result from the simulation. This process continues with designers making changes to the terminal plan, and analysts would check for validation using agent simulations. At a certain point, designers would reach an arrangement of spaces that gives acceptable passenger performance, at which point the design would be considered validated. Designers then can have confidence that their latest iteration is suitable for passengers. Ultimately, the impact during the iteration process is that designs must go through validation testing to confirm that the layout aligns with passenger activity, before moving on to detailed design.

Client Interaction:

The third impact to the design process is the interaction with clients. Architects can use agent-based simulations to illustrate why they made certain design decisions. When trying to sell ideas to clients during presentations, architects using simulations have more convincing arguments using simulation data than renders or simple animations can show. Firstly, architects can have confidence that they validated their design against standard tests. They can show clients quantitative numbers from testing to explain how their design compares to other concepts. Agent-based simulations can illustrate what areas of an airport terminal are having the most impact on passenger activity. By using an analytic approach to validate the building

performance, architects give more value to their work. Architects may choose to create an unconventional layout or have an aesthetically motivated approach. These simulations can provide confidence for their clients that the unusual design still meets the needs of the project. Clients also have their own project goals, like improving retail spaces or improving security areas. They can benefit from agent-based simulations by verifying how much value their current building has, and what architectural factors can change to improve those aspects. If a simulation of the original building got an architectural score of 40% for retail, due to passengers not spending time in those areas, then the simulation can indicate to clients what architectural changes can improve retail score to 80%. The greatest impact for clients is improvements for airport operations and overall business. If business owners are expecting a certain performance target, agent simulations can validate how the terminal impacts those business decisions.

Built Environment

The main impact to the built environment is providing people with a seamless experience. For example, a passenger walking through a terminal may decide to get a coffee. In a well tested terminal, a café will be there the moment they start to think about it. Layout testing may show, after simulating 10 000 people, that majority of people also wanted to get coffee at that same location. Therefore, a designer using this approach will make sure the terminal has good access to a café at that location.

In general, validating a terminal design with agent-based simulations will show what arrangement of space is most intuitive to walk through. Resulting designs will be better suited for international passengers who are new to an airport. Agent-based validation ensures passengers will be less confused in areas they have never experienced before. For example, signage will be in place in locations that make sure passengers have correct information and the given signs are relevant for where they need to go. Architects will arrange corridors, so people do not need to weave through unintended spaces, like waiting rooms or other gates. Services like information kiosks will be in areas that make it easier to access for people who are likely to get lost. Food and retail spaces will be designed in locations where people are most likely to take a break. Accessible features, like elevators or other aids, will be designed in suitable locations for those with corresponding disabilities. Ideally, architects will design spaces to give passengers better intuition about where things are.

Another impact to the design of terminals is how airports arrange essential areas. It is possible to consider that validation will show passengers have better performance with a different order

of check-in and security. Agent simulations might conclude that if people have a high importance for security screening, it is better to have screening as the first thing when entering a terminal. Instead of laying out a terminal based on existing practices, planners design a terminal that works best with passenger perception. Obviously, this is a simple example, and the design of an airport is dependant on more than just passenger perception. However, spatial validation can start to predict how much the architecture has an effect from these choices, based on quantitative values. Ultimately, architects can be more precise about the value of their design choices on essential areas in an airport.

Long-term Impact

Some long-term impacts of using agent-based simulation for architecture include shifting design to more evidence-based practices and automating design choices. Additionally, the biggest long-term risk of letting simulations control design decisions is optimizing for something that is unintentionally harmful to human health and safety.

Evidence-based Practices:

In the future, architectural design will need to follow scientific practices to be effective for the built environment. The process will include proposing a design hypothesis, and then being required to test the design to check its performance, the same way it is done for material and building sciences. This process will become similar to the engineering design process, which commonly uses a scientific framework and physical analysis to prove the performance of new systems or technology.

For buildings that see many public people, this may result in industry regulations on spatial practices, like regulation on building evacuations and environmental impacts do now. Additionally, this may also result in company design audits to confirm that architectural spaces are meeting a certain spatial standard to prevent bad design practice.

Architects will be responsible for interpreting the outcomes of the simulations or similar tools. This includes checking that building components are validated and deciding what parts of the building need to be tested. However, this does not mean architects will stop making innovative designs. In fact, this will only change how they approach design choices.

In the future, simulations will automate the design iteration process by optimizing layouts based on the performance of passenger behaviour. The agent simulations could iterate through a thousand more concepts that a single designer could explore by intuition. An architect's time on

a project would shift from design exploration to design analysis. They will become responsible for interpreting the data from the simulation trials, judging the accuracy of the outcomes, and deciding on a final design based on the results.

Risk of Optimizing for Unknown Behaviour:

The long-term risk of relying on simulations for architectural design is optimizing for unknown human behaviour. Optimizing for certain passenger priorities today may not be representative of human behaviour in an airport 50 years from now. As a result, relying on purely automated systems can result in something that is unintentionally harmful for society, if no one is there to interpret the results.

For example, until recently, North American airports used to have dedicated smoking lounges in terminal buildings. However, since smoking is widely seen as unhealthy and unsafe, indoor smoking in Canada was restricted by the government. As a result, the need for smoking areas in an airport was removed. If a hypothetical agent simulation optimized a terminal design for smoking lounges, as a passenger priority, decades earlier, then the building would not be well suited for current passengers today.

Fundamentally, planners could be using simulations to optimize for conditions that may be considered unsafe, unhealthy, or socially unacceptable in the future. Some examples may include energy usage (replacing poorly insulated enclosures and combustion building generators), technology changes, (remote check-in, preapproved security clearances) or disease prevention (quarantined areas, seat spacing in waiting rooms, health-check stations).

Unfortunately, there are problems that society is not aware of or has no current way of understanding what they will be. It is difficult for computers today to predict human behaviour far into the future and still have high confidence in the results, like *psychohistory* in Asimov's Foundation series. Additionally, the way a simulation interprets information is based on the biases of the person who created it. Therefore, a computer's understanding of the world can only be as good as the models it is provided.

The risk of relying on simulations that may not be accurate is being able to identify when they give wrong information. If a designer is not already familiar with the background of a project, they will not be able to recognize when the simulation is wrong. The risk is designers might take the results of the simulation as the only truth, without any further analysis or critical evaluation. Simulations should not be relied on as a primary design tool. They are most useful when they

can increase a designer's understanding of a system. The final judgement is still up to the designer to decide how this information is used.

Avoiding these issues will involve continually re-evaluating building systems over time and updating the latest simulation models as our understanding of social behaviour changes. Additionally, this will require more focus on studying public spaces, and scientifically recording information about human interactions for architectural design.

Chapter 5.3

Future Work

The research from this thesis could be taken in two directions. One direction is to move forward with the agent simulation to study the automation of testing and the design process. The other direction is to go back and re-evaluate the relationship between human behaviour and architecture. Automation would be best explored through the creation of another architectural software program. Whereas behavioural relationships would be best studied through real-world experiments with people in the built environment.

Design Automation

The research from this thesis begins the discussion for automation in architectural design. The goal of automation is to reduce the amount of time architects spend exploring design iterations. To get a computer to automate a system, there must be a process in place, or instructions, that a computer can follow. This thesis explores one way to model architectural performance. A computer can use a similar model like this to optimize for a given condition. For example, if a designer wants to optimize a building for high visibility, then they can program the computer to maximize agent field of view in specific locations in the building. More broadly, if a client for an airport wants to increase the passenger interaction in retail areas, then architects can program a simulation to optimize a design for these retail areas.

Current research in automation can involve *machine learning* techniques. This is an artificial system that programs, or *learns*, behaviour without having a user code in specific conditions. For example, instead of an architect telling an airport simulation that an open retail area by departure gates is better, a machine learning program can recognize that designing gates with open retail areas gives better passenger performance.

Machine learning algorithms involve optimizing an *artificial neural network*, which is a graph-like, tree-base, hierarchical structure with weights for specific system attributes. Basically, it is like the computer's brain, which tells the program how to behave under certain conditions. A program learns by updating the weights in this hierarchy, after it studies training data, or performs trial-runs. A computer recognizes what the correct behaviour is by comparing its own output to a correct solution, which may be defined by the user initially. For example, a simple

machine learning program for a driverless car might have weights for steering and obstacle proximity, and a penalty system for hitting a wall. These weights would be changed every time the car hits a wall. A similar score and penalty system would need to be created for architectural conditions.

Depending on the size of its neural network, a machine learning program may need to perform millions of trial-runs. It will also require a lot of training data to optimize for a certain behaviour. This is especially important for architecture because there are dozens of factors that influence design choices, which requires optimizing a weight for each condition. Future agent simulations can incorporate a machine learning algorithm that takes in the performance of the agents (or passengers) as an input and optimizes their architectural value by modifying the walls in a floor plan. Ideally, these simulations can begin to recognize which layout will produce higher architectural value, or as a result, better building performance.

Architectural Influence on Human Behavior

Another critical area of research is studying how architectural choices influence human behaviour. There is no use in optimizing for an agent behaviour if it is not representative of real-world conditions. The thesis chose to focus on airports because there is a lot of research in passenger experience analysis. Future architectural research may also want to continue studying passenger experience, or even explore patient experience in healthcare facilities.

One area of focus in this thesis was understanding people's perception of the built environment. The thesis realizes this is not as simple as asking how a wooden wall affects people's behaviour compared to a concrete wall. Instead, the thesis concludes that people's perception is unique to the individual, and it depends on what people are doing. For example, if people are looking for their platform in a train station, then a solid concrete wall can be a physical obstruction. However, if people are waiting around before their train has arrived, then they may use the solid concrete wall as a place to wait beside or lean on. Therefore, there can be different behaviours for the same architectural feature.

Research may want to identify what behaviours are likely to occur and explore what architectural elements relate to those behaviours. There is opportunity to explore real human activity in actual buildings, like terminals or community centres, as a scientific study. This research can help better understand what activities people normally do in the built environment. It can also help identify where in a building these activities might occur, or how these activities relate to the overall layout of the architecture.

Human behavioural research will also involve exploring ways of assigning a value to architectural conditions. Valuation is important to help perform a scientific study, since research focusing on qualitative properties of an environment can be very diverse. It is also useful to have quantifiable data from these tests so it can be used for analysis or forecasting. Having a ranking system can be beneficial in field-studies when comparing two completely different observations. This thesis explored using a weight-based metric, which normalized people's priorities. However, there are other ranking systems that could provide similar information, as seen in existing passenger experience surveys. The results from this kind of research could then be used to create simulation studies of architectural environments.

Chapter 5.4

Summary of Conclusion

In summary, the thesis's agent simulation can successfully differentiate between certain architectural layouts depending on an agent's airport priorities. The thesis believes any effective architectural simulation must at least use existing practices from crowd modelling, incorporate perception for agent decision making, and analyse architectural values using statistics. If generalized enough, agent simulations can be used to test any building type, in addition to airport terminals.

Simulations can impact architecture by providing a quantifiable value for design iterations. This type of analysis can be effective for architects to communicate design issues or benefits to their clients. Testing will result in more intuitive spaces, which will give people a more seamless experience in the built environment. However, the risk of architects only relying on simulations to create designs is not being able to recognize when they output wrong information. Without critical evaluation, architects risk building something that becomes detrimental to people in the long-term, as society's needs change over time. Ultimately, the most effective use of simulations is for clarifying uncertainty in the existing design process. This will not replace the role of the architect; it will only change how they think about design choices.

Future work from this thesis can take two different directions. One direction is automating agent simulations by exploring machine learning techniques to optimize architectural conditions. The other direction is studying how human behaviour relates to certain architectural elements, by conducting experiments with people in real-world buildings.

In conclusion, the thesis has demonstrated that architectural design can be approached quantitatively. Continuing to learn from these established practices will provide more scientific rigour for architecture in the future.

Letter of Copyright Permission

A* Pathfinding – Collection of Unity Scripts

Lague, Sebastian. "Pathfinding/Episode 10 - threading/Assets/Scripts/". GitHub. December 30, 2016. <https://github.com/SebLague/Pathfinding/tree/master/Episode%2010%20-%20threading/Assets/Scripts>.

MIT License

Copyright (c) 2017 Sebastian Lague

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Field-of-View – Collection of Unity Scripts

Lague, Sebastian. "Field-of-View/Episode 02/Scripts/FieldOfView.cs". GitHub. December 28, 2015. <https://github.com/SebLague/Field-of-View/blob/master/Episode%2002/Scripts/FieldOfView.cs>.

MIT License

Copyright (c) 2016 Sebastian

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

References

- 3Blue1Brown. "Eigenvectors and eigenvalues | Essence of linear algebra, chapter 14". Youtube, September 15, 2016. <https://www.youtube.com/watch?v=PFDu9oVAE-g>.
- 99pi. "The White Elephant of Tel Aviv." 99% Invisible, March 29, 2016. <https://99percentinvisible.org/episode/stop-that-bus/>.
- Abdelhak, Haifa; Ayesh, Aladdin; Olivier, Damien. "Cognitive Emotional Based Architecture for Crowd Simulation". *Journal of Intelligent Computing*, June 2012, 2012. Vol. 3 (2), pp. 55-66.
- ABET. "Criteria for Accrediting Engineering Programs, 2019 – 2020." ABET. Accessed June 2020. <https://www.abet.org/accreditation/accreditation-criteria/criteria-for-accrediting-engineering-programs-2019-2020/#definitions>.
- AirHelp. "AirHelp Score 2019: Global Airport Rankings". AirHelp 2019. Accessed October 2020. https://static.airhelp.com/pdf/2019-airport-score/methodology_airhelp_score_2019__global_airport_rankings-en_us.pdf.
- AirHelp. "Global Airport Ranking". AirHelp, 2019. Accessed October 2020. <https://www.airhelp.com/en/airhelp-score/airport-ranking/>.
- Airport Technologies. "Can cultural differences impact passenger satisfaction?". Analysis. Updated December 7th, 2018. <https://www.airport-technology.com/features/passenger-satisfaction-in-airports/>.
- Alexander, Christopher, Sara Ishikawa, and Murray Silverstein. *A Pattern Language: Towns, Buildings, Construction*. New York: Oxford Univ. Pr., 1977.
- Alexander, Christopher. *Notes on the Synthesis of Form*. Cambridge: Harvard University Press, 1964.
- Arabacioglu, Burcin Cem. "Using Fuzzy Inference System for Architectural Space Analysis." *Applied Soft Computing* 10, no. 3 (2010): 926–37. <https://doi.org/10.1016/j.asoc.2009.10.011>.
- Arbuckle, Alex Q. "1910-1963 The Destruction of Penn Station." Mashable, July 20, 2015. <https://mashable.com/2015/07/20/original-penn-station/#.utV4feGIPq1>.
- Arena. "Discrete Event Simulation Software." Arena Simulation Software. Accessed December 2019. <https://www.arenasimulation.com/what-is-simulation/discrete-event-simulation-software>.
- Argenton, Rodrigo. "File:Galton box.jpg". Wikimedia Commons. December 19, 2016. https://commons.wikimedia.org/wiki/File:Galton_box.jpg.
- Arup. "The Verification and Validation of MassMotion for Evacuation Modelling." Ove Arup & Partners Ltd. August 10, 2015. <https://www.oasys-software.com/wp-content/uploads/2017/11/The-Verification-and-Validation-of-MassMotion-for-Evacuation-Modelling-Report.pdf>.
- Aschwanden, Gideon, Jan Halatsch, and Gerhard Schmitt. "Crowd Simulation for Urban Planning". *Architecture in Computro [26th eCAADe Conference Proceedings / ISBN 978-0-9541183-7-2] Antwerpen (Belgium) (17-20 September 2008)*: pp. 493-500.

- ASHRAE. "ASHRAE Standards Strategic Plan 2014-15". American Society of Heating, Refrigerating and Air Conditioning Engineers, Inc. July 2, 2014.
- ASHRAE. "Standard 100-2015 -- Energy Efficiency in Existing Buildings (ANSI Approved/IES Co-sponsored)". American Society of Heating, Refrigerating and Air Conditioning Engineers, Inc. Accessed February 2021. <https://www.ashrae.org/technical-resources/bookstore/standard-100>.
- Augé Marc. *Non-Places: Introduction to the Anthropology of Supermodernity*. London: Verso, 1995.
- Balci, Osman. "Validation, Verification, and Testing Techniques Throughout the Life Cycle of a Simulation Study." *Annals of operations research* 53, no. 1 (December 1994): 121–173.
- Banks, Jerry; Carson II, John S; Nelson, Barry L; Nicol, David M. *Discrete-Event System Simulation 4th ed.* Upper Saddle River, N.J: Pearson Prentice Hall, (2005).
- Batty, Michael. *Cities and Complexity: Understanding Cities with Cellular Automata, Agent-Based Models, and Fractals*. Cambridge, MA: MIT, 2007.
- Belogolovsky, Vladimir. "Paul Andreu: 'I Would Only Take On a Project If the Ideas Were Mine. Otherwise, I Am Not Interested.'" *ArchDaily*. March 7, 2017. <https://www.archdaily.com/806698/paul-andreu-i-would-only-take-on-a-project-if-the-ideas-were-mine-otherwise-i-am-not-intereste>.
- Bhattacharya, Subhrajit. "File:Astar progress animation.gif". Wikimedia Commons. April 13, 2011. https://en.wikipedia.org/wiki/File:Astar_progress_animation.gif.
- Bhattacharya, Subhrajit. "File:Dijkstras progress animation.gif". Wikimedia Commons. April 13, 2011. https://commons.wikimedia.org/wiki/File:Dijkstras_progress_animation.gif.
- Bouarfa, S, H.A.P Blom, R Curran, and M.H.C Everdij. "Agent-Based Modeling and Simulation of Emergent Behavior in Air Transportation." *Complex adaptive systems modeling* 1, no. 1 (August 15, 2013): 1–26.
- Broadbent, Geoffrey. *Design in Architecture; Architecture and the Human Sciences*. London: John Wiley & Sons, 1973.
- Broto, Carles. *Transport Facilities*. Barcelona: Link Books, 2012.
- Buchgeher, Georg, and Rainer Weinreich. "Continuous Software Architecture Analysis." *Agile Software Architecture*, 2014, 161–88. <https://doi.org/10.1016/b978-0-12-407772-0.00006-x>.
- Canada Alive. "Pearson International Airport, Toronto", Canada Alive, April 8, 2014, Accessed September 2019, <https://canadaalive.wordpress.com/2014/04/08/pearson-international-airport-toronto/>.
- Carmo, Mario. *The Second Digital Turn Design Beyond Intelligence*. Cambridge, MA: The MIT Press, (2017).
- Changi Airport Singapore. "Maps: Changi Airport Singapore." Maps | Changi Airport Singapore. Accessed October 2019. <http://www.changiairport.com/en/maps.html#17.16/1.354975/103.989599/-67>.
- Chatzikonstantinou, Ioannis; Sariyildiz, Sevil; Bittermann, Michael S. "Conceptual Airport Terminal Design Using Evolutionary Computation." *In 2015 IEEE Congress on Evolutionary Computation (CEC)*, 2245–2252. IEEE, 2015.

- Choudhary, Shweta; Pipralia, Satish. "Architectural Perception for Redevelopment of Railway Termini". *International Journal on Emerging Technologies* 8(1), January 26, 2017.
- Chu, Mei Ling, and Kincho Law. "Computational Framework Incorporating Human Behaviors for Egress Simulations." *Journal of Computing in Civil Engineering* 27, no. 6 (November 1, 2013): 699–707.
- Collins, Peter, and William Dendy. *Architectural Judgement*. Montreal: McGill-Queens University Press, 1971.
- Cuff, Dana. *Architecture: the Story of Practice*. Cambridge, Mass: MIT Press, 1991.
- Cullen, Gordon. *The Concise Townscape*. Abingdon: Routledge, 1971.
- Deputy City Manager. "Union Station Revitalization Project (USRP) – Status Report." City of Toronto, June 18, 2020.
- Designworkplan. "Airport Signage." /designworkplan wayfinding design studio. Accessed November 26, 2019. <https://www.designworkplan.com/read/airport-signage-photo-inspiration>.
- Dewey, John. *Theory of Valuation*. Chicago, Ill: University of Chicago Press, 1939.
- Do, Ellen Yi-Luen, and Mark D. Gross. "Tools for Visual and Spatial Analysis of CAD Models." *CAAD Futures* 1997, 1997, 189–202. https://doi.org/10.1007/978-94-011-5576-2_15.
- Dt-rush-8. "File:Queueing node service digram.png". Wikimedia Commons. 8 December 2018. https://commons.wikimedia.org/wiki/File:Queueing_node_service_digram.png.
- Easterling, Keller. *Organization Space: Landscapes, Highways, and Houses in America*. Cambridge, MA: The MIT Press, 2001.
- EGS India. "Why, How and When do you perform Design Validation for Automotive Systems?". Solidworks Tech Blog. August 30, 2016. <https://blogs.solidworks.com/tech/2016/08/performance-design-validation-automotive-systems.html>.
- Evans, Michael J; Rosenthal, Jeffery S. *Probability and Statistics the Science of Uncertainty Second Edition*. University of Toronto: 2009.
- FAA. "Advisory Circular 150/5360-13A - Airport Terminal Planning". APP-400, Office of Airport Planning & Programming, Planning & Environmental Division. U.S. Department of Transportation, Federal Aviation Administration (FAA). July 13, 2018. https://www.faa.gov/documentLibrary/media/Advisory_Circular/AC-150-5360-13A-Airport-Terminal-Planning.pdf.
- FAA. "Airport Design and Engineering Standards." FAA seal, August 2019. https://www.faa.gov/airports/engineering/design_standards/.
- FDA. "Guideline on General Principles of Process Validation". U.S. Department of Health and Human Services Food and Drug Administration, FDA-2008-D-0559. Updated 2018-08-24.
- Fiederer, Luke. "AD Classics: TWA Flight Center / Eero Saarinen". ArchDaily. 2016-06-16. <https://www.archdaily.com/788012/ad-classics-twa-flight-center-eero-saarinen>.

- Fischer, Alex. "Quelea - Agent-Based Design for Grasshopper." Grasshopper. Accessed December 14, 2019. <https://www.grasshopper3d.com/groups/group/show?groupUrl=quelea-agent-based-design-for-grasshopper&>.
- FlexSim. "FlexSim 3D Simulation Modeling Software." FlexSim, September 13, 2017. <https://www.flexsim.com/flexsim/>.
- Frank, Andrew; Raper, Jonathan; Cheylan, Jean-Paul (eds.), *The Life and Motion of Socio-Economic Units*. (GISDATA 8), London: Taylor and Francis, 2001.
- Franklin, Stan; Graesser, Art. "Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents". *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, (1996) Springer-Verlag.
- Fruin, John J. *Designing for Pedestrians: A Level of Service Concept*. New York, 1970. <http://onlinepubs.trb.org/Onlinepubs/hrr/1971/355/355-001.pdf>.
- Fruin, John J. *Pedestrian Planning and Design*. New York: Metropolitan Association of Urban Designers and Environmental Planners, 1971.
- Galea, Alexander. "Galton's Peg Board and the Central Limit Theorem". WordPress. March 11, 2016. <https://galeascience.wordpress.com/2016/03/11/galtons-peg-board-and-the-central-limit-theorem/>.
- Getlein, M. *Living with Art*. New York, NY: McGraw-Hill, (2013): p. 115.
- Gibson, James J. *The Ecological Approach to Visual Perception*. New York: Psychology Press, 1986.
- Globalphotos. "Toronto Pearson Airport Photo Gallery". globalphotos.org. Accessed November 2019. <https://www.globalphotos.org/to-airport.htm>.
- Great Buildings. Dulles Airport. Accessed March 2021. http://www.greatbuildings.com/buildings/Dulles_Airport.html.
- GTAA. "2018 Airport Construction Code, v5.0", Toronto Pearson International Airport, GTAA, 2018, <https://tpprocdnep.azureedge.net/-/media/project/pearson/content/operators-at-pearson/construction/pdfs/airport-construction-code.pdf?modified=20190302002058>.
- GTAA. "Toronto Pearson". Greater Toronto Airports Authority – GTAA, Accessed September 2019, <https://maps.torontopearson.com/>.
- GTAA. "Toronto Pearson International Airport Master Plan 2017-2037", Greater Toronto Airports Authority (GTAA), 2017, 38, 56, 85, <https://tpprocdnep.azureedge.net/-/media/project/pearson/content/corporate/our-future/pdfs/gtaa-master-plan.pdf?la=en&modified=20190228235920&hash=6C155E44692A278979B42F1F976A7456D7F2D53F>.
- Guichard, David. *An Introduction to Combinatorics and Graph Theory*. Whitman College. January 30, 2020.
- Guru99. "Design Verification and Validation". Guru99, accessed May 27, 2020, <https://www.guru99.com/design-verification-process.html>.

- Hagtvedt, Patrick. "The Perception and Evaluation of Visual Art." *Empirical studies of the arts* 26, no. 2 (July 2008): 197–218.
- Hamilton, D. Kirk. "Four Levels of Evidence-Based Practice". *Healthcare Design*. 3,4: 18-26.
- Hanafi, Israa, Moustafa El Araby, Khalid Al Hagla, and Samer El Sayary. "Human Social Behavior in Public Urban Spaces: Towards Higher Quality Cities." *Spaces and Flows: An International Journal of Urban and ExtraUrban Studies* 3, no. 2 (2013): 23–35. <https://doi.org/10.18848/2154-8676/cgp/v03i02/53690>.
- Hillier, Bill, and Julienne Hanson. *The Social Logic of Space*. Cambridge: Cambridge University Press, 1984. doi:10.1017/CBO9780511597237.
- Hirose, Iwao; Olson, Jonas. *The Oxford Handbook of Value Theory*. New York: Oxford University Press, 2015.
- Hirsh, Max. *Airport Urbanism Infrastructure and Mobility in Asia*. Minneapolis: University of Minnesota Press, 2016.
- HOK. "HOK to Design New Transit Hub at Toronto Pearson International Airport." HOK, February 6, 2018. <https://www.hok.com/news/2018-02/hok-to-design-new-transit-hub-at-toronto-pearson-international-airport/>.
- Holm, Ivar. "Ideas and Beliefs in Architecture and Industrial Design". PhD thesis, Oslo School of Architecture and Design, 2006.
- Hoy, Gregory, Erin Morrow, and Amer Shalaby. "Use of Agent-Based Crowd Simulation to Investigate the Performance of Large-Scale Intermodal Facilities: Case Study of Union Station in Toronto, Ontario, Canada." *Transportation Research Record* 2540, no. 1 (January 2016): 20–29. <https://doi.org/10.3141/2540-03>.
- Indraprastha, Aswin, and Michihiko Shinozaki. "Elaboration Model for Mapping Architectural Space." *Journal of Asian Architecture and Building Engineering* 10, no. 2 (November 2011): 351–58. <https://doi.org/10.3130/jaabe.10.351>.
- IMO. "Guidelines for Evacuation Analysis for New and Existing Passenger Ships." International Maritime Organization (IMO). MSC.1/Circ.1238. October 30, 2007. <https://nsf.no/media/1129/imo-msc-guidelines-for-evacuation-etc.pdf>.
- ISO. "Systems and software engineering -- System life cycle processes." ISO/IEC/IEEE 15288:2015, 2015-05, 4.37 validation.
- Joe. "List of Random Names." Accessed August 2020. <http://www.listofrandomnames.com/>.
- Kepes, G. *The Nature and Art of Motion*. London: Studio Vista, 1967.
- Khan Academy. "Central Limit Theorem". Math, AP® / College Statistics, Sampling Distributions, Sampling Distributions of a sample mean. Accessed February 2021. <https://www.khanacademy.org/math/ap-statistics/sampling-distribution-ap/sampling-distribution-mean/v/central-limit-theorem>.
- Khan Academy. "Deep definition of the normal distribution". Math, Statistics and probability, Modelling data distributions, More on normal distributions. Accessed February 2021.

<https://www.khanacademy.org/math/statistics-probability/modeling-distributions-of-data/more-on-normal-distributions/v/introduction-to-the-normal-distribution>.

Khan Academy. "Describing Graphs." Computer Science, Algorithms, Graph Representation, Accessed May 2020. <https://www.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/describing-graphs>.

Khan Academy. "Introduction to eigenvalues and eigenvectors". Linear Algebra, Alternative coordinate systems, Eigen-everything. <https://www.khanacademy.org/math/linear-algebra/alternate-bases/eigen-everything/v/linear-algebra-introduction-to-eigenvalues-and-eigenvectors>.

Khan Academy. "Representing Graphs." Computer Science, Algorithms, Graph Representation, Accessed May 2020. <https://www.khanacademy.org/computing/computer-science/algorithms/graph-representation/a/representing-graphs>.

Kim, Giseop; Kim, Ayoung; Kim, Youngchul. "A New 3D Space Syntax Metric Based on 3D Isovist Capture in Urban Space Using Remote Sensing Technology." *Computers, Environment and Urban Systems* 74 (2019): 74–87.

Kochanski, Jakub. "GoPro Time Lapse – People at O’Hare Airport [Terminal 3]". YouTube. April 8, 2015. <https://www.youtube.com/watch?v=ipHebqQTDlg>.

D.P. Kroese, T. Taimre, Z.I. Botev. *Handbook of Monte Carlo Methods*. Wiley Series in Probability and Statistics, John Wiley & Sons, New York, (2011).

Lague, Sebastian. "Pathfinding/Episode 7-smooth weights/Assets/Scripts". GitHub. December 30, 2016. <https://github.com/SebLague/Pathfinding/tree/master/Episode%209%20-%20smooth%20path%202/Assets/Scripts>.

Lague, Sebastian. "Field-of-View/Episode 02/Scripts/FieldOfView.cs". GitHub. December 28, 2015. <https://github.com/SebLague/Field-of-View/blob/master/Episode%2002/Scripts/FieldOfView.cs>.

Lague, Sebastian. "Field of view visualization (E02)". Youtube. December 27, 2015. <https://www.youtube.com/watch?v=73Dc5JTCmKI>.

Lane, David et al. "Sampling Distribution". *Onlinestatbook.com*, Rice Virtual Lab in Statistics (RVLS), Rice University. Accessed February 2021. https://onlinestatbook.com/stat_sim/sampling_dist/index.html.

Le Corbusier. *Modulor I and II*. Cambridge, Mass: Harvard University Press, 1980.

Lee, Peter M. *Bayesian Statistics an Introduction* 4th ed. Chichester, West Sussex, 2012.

Lera, Sebastian G. "Architectural Designers’ Values and the Evaluation of Their Designs." *Design studies* 2, no. 3 (1981): 131–137.

Lera, Sebastian G. "Empirical and Theoretical Studies of Design Judgement: A Review." *Design studies* 2, no. 1 (1981): 19–26.

Lidwell, William, Kritina Holden, and Jill Butler. *Universal Principles of Design: 125 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach through Design*. Beverly, MA: Rockport, 2010.

- Liu, Ao (Leo). “Dynamic Visualizations: Developing a Framework for Crowd-Based Simulations” MArch thesis, University of Waterloo, 2020.
- Liu, Z, Liu, T, Ma, M, Hsu, H-H, Ni, Z, Chai, Y. A perception-based emotion contagion model in crowd emergent evacuation simulation. *Comput Anim Virtual Worlds*. 2018; 29:e1817. <https://doi.org/10.1002/cav.1817>.
- Markup Validation Service. “Why Validate?” Markup Validation Service. Accessed May 27, 2020, <https://validator.w3.org/docs/why.html>.
- Marmot, Alexi. “Architectural Determinism.” *The British Journal of General Practice* 52, no. 476 (March 2002): 252–53. <https://bjgp.org/content/bjgp/52/476/252.full.pdf>.
- McCarthy, Owen. “Game Design Deep Dive: Creating Believable Crowds in Planet Coaster.” Gamasutra Article, January 4, 2017. https://www.gamasutra.com/view/news/288020/Game_Design_Deep_Dive_Creating_believable_crowds_in_Planet_Coaster.php.
- McElhinney, Sam. “Isovist_2.2: a basic user guide”. v1, 2018. https://isovists.org/user_guide/.
- Mentatdgt. “Photography of People at Train Station”. Pexels. August 09, 2018. <https://www.pexels.com/photo/photography-of-people-at-train-station-1311544/>.
- Miller, G.A. “The magical number seven, plus or minus two: Some limits on our capacity for processing information.” *Psychological Review*, 63(2): 81-97, doi:10.1037/h0043158.
- Microsoft. “Classes (C# Programming Guide).” C# Documentation, August 21, 2018. Accessed October 2020. <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/classes>.
- Microsoft. “Methods in (C#).” C# Documentation, May 21, 2018. Accessed October 2020. <https://docs.microsoft.com/en-us/dotnet/csharp/methods>.
- Microsoft. “Types (C# Programming Guide).” C# Documentation, July 20, 2015. Accessed October 2020. <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/types/>.
- Musse, S. R., and D. Thalmann. “A Model of Human Crowd Behavior: Group Inter-Relationship and Collision Detection Analysis.” *Eurographics Computer Animation and Simulation '97*. (1997). 39–51. https://doi.org/10.1007/978-3-7091-6874-5_3.
- National Academies of Sciences, Engineering, and Medicine. “Airport Passenger-Related Processing Rates Guidebook”. *Washington, DC: The National Academies Press*. (2009). <https://doi.org/10.17226/22990>.
- National Academies of Sciences, Engineering, and Medicine. “Airport Passenger Terminal Planning and Design, Volume 1: Guidebook”. *Washington, DC: The National Academies Press*. (2010). <https://doi.org/10.17226/22964>.
- National Academies of Sciences, Engineering, and Medicine. Airport Passenger Terminal Planning and Design, Volume 2: Spreadsheet Models and User's Guide. *Washington, DC: The National Academies Press*. (2010). <https://doi.org/10.17226/14356>.

- National Academies of Sciences, Engineering, and Medicine. "Innovations for Airport Terminal Facilities". *Washington, DC: The National Academies Press*. (2008). <https://doi.org/10.17226/14219>.
- National Academies of Sciences, Engineering, and Medicine. "Passenger Level of Service and Spatial Planning for Airport Terminals". *Washington, DC: The National Academies Press*. (2011). <https://doi.org/10.17226/14589>.
- National Academies of Sciences, Engineering, and Medicine. "Simulation Options for Airport Planning". *Washington, DC: The National Academies Press*. (2019). <https://doi.org/10.17226/25573>.
- Neumann, Peter. "Kosten für Großflughafen steigen um 160 Millionen Euro, weil mehr Passagiere erwartet werden: Noch nicht gebaut und schon teurer [Costs for major airports rise by 160 million euros because more passengers are expected: not yet built and already more expensive]". *Berliner-Zeitung*, 2008-07-10. <https://www.berliner-zeitung.de/kosten-fuer-grossflughafen-steigen-um-160-millionen-euro-weil-mehr-passagiere-erwartet-werden-noch-nicht-gebaut-und-schon-teurer-li.6277>.
- Nicoguardo. "File:Pi 30K.gif". Wikimedia Commons. February 16, 2017. https://commons.wikimedia.org/wiki/File:Pi_30K.gif.
- Norman, Donald A. *The Design of Everyday Things*. New York: Basic Books, A Member of the Perseus Books Group. (2013).
- NRCC. *National Building Code of Canada 2015 Volume 1*. National Research Council of Canada. Ottawa, September 28, 2018.
- O'Sullivan, Feargus. "Planning the Transit Hubs of the Future." CityLab, July 10, 2017. <https://www.citylab.com/design/2017/07/planning-the-transit-hubs-of-the-future/532905/>.
- Oasys. "HOK Benefits from BIM Integration of Pedestrian Simulation." Oasys. Accessed January 2020. <https://www.oasys-software.com/case-studies/hok-benefits-bim-integration-pedestrian-simulation/>.
- Oasys. "JetBlue – T5 JFK New York." Oasys. Accessed February 2020. <https://www.oasys-software.com/case-studies/jetblue-t5-jfk-new-york/>.
- Oasys. "MassMotion Help Guide," July 2019. <https://www.oasys-software.com/wp-content/uploads/2019/06/MassMotion-10.0-Help-Guide.pdf>.
- Ostwald, Michael. "Le Corbusier (Charles Edouard Jeanneret), The Modulor and Modulor 2 – 2 Volumes. Basel: Birkhäuser, 2000: Reviewed by Michael J. Ostwald". Basel: Birkhäuser-Verlag, April 2001.
- Palisade. "Monte Carlo Simulation". Risk. Accessed February 2021. https://www.palisade.com/risk/monte_carlo_simulation.asp.
- PANYNJ. "Terminal Planning Guidelines", The Port Authority of New York and New Jersey, August 2013, <https://www.fd.cvut.cz/projects/k621x1ml/dokumenty/panynj-terminal-planning-guidelines.pdf>.
- Popovic, Vesna and Kraal, Ben and Kirk, Philip J. "Towards airport passenger experience models." *Proceedings of 7th International Conference on Design & Emotion*, October 2010, 4-7.

- Prabhu, Frischmann. "1. Pedestrian Simulation at a Metro Station (Peak Hour)." YouTube. YouTube, September 28, 2017. <https://www.youtube.com/watch?v=nZ3pE-nIJio>.
- Proulx, Christian. "Complete Terminal Simulation". Youtube. March 1, 2014. <https://www.youtube.com/watch?v=iaXdMO67g0E>.
- Purcell, A.t. "The Relationship between Buildings and Behaviour." *Building and Environment* 22, no. 3 (1987): 215–32. [https://doi.org/10.1016/0360-1323\(87\)90010-2](https://doi.org/10.1016/0360-1323(87)90010-2).
- Raubal, Martin. "Agent-Based Simulation of Human Wayfinding: A Perceptual Model for Unfamiliar Buildings". PhD diss., Vienna University of Technology, October 2001.
- Rittel, Webber. "Dilemmas in a General Theory of Planning." *Policy sciences* 4, no. 2 (June 1973): 155–169.
- Robinson, Stewart. *Simulation: the Practice of Model Development and Use*. Chichester, England: Wiley, (2004).
- RobotC. "Sense Plan Act (SPA)". Natural Language Resources – VEX Cortex. Accessed November 2020. http://cdn.robotc.net/pdfs/natural-language/hp_spa.pdf.
- Ronchi, Enrico; Kuligowski, Erica D; Reneke, Paul A; Peacock, Richard D; Nilsson, Daniel. "The Process of Verification and Validation of Building Fire Evacuation Models." *Technical Note (NIST TN) - 1822*, (November 2013), <http://dx.doi.org/10.6028/NIST.TN.1822>.
- Rosenhahn, Bodo. *Human Motion: Understanding, Modeling, Capture and Animation*. Dordrecht: Springer, 2008.
- Saarinen, Eero. *Eero Saarinen On His Work*. New Haven, CT: Yale University Press, 1968. section drawing, p108.
- Saaty, Thomas L. "A Scaling Method for Priorities in Hierarchical Structures." *Journal of mathematical psychology* 15, no. 3 (1977): 234–281.
- Saaty, Thomas L. "Modeling Unstructured Decision Problems – the Theory of Analytical Hierarchies." *Mathematics and computers in simulation* 20, no. 3 (1978): 147–158.
- SAE. "Browse Standards". Society of Automotive Engineers. Accessed February 2021. <https://www.sae.org/standards>.
- Sander, Lou. "File:AHPHierarchy1.1.png". Wikimedia Commons. 24 February 2009. <https://commons.wikimedia.org/wiki/File:AHPHierarchy1.1.png>.
- Schiphol. "Airport Maps." Schiphol. Accessed October 31, 2019. <https://www.schiphol.nl/en/airport-maps>.
- Schroeder, Mark. "Value Theory". *The Stanford Encyclopedia of Philosophy* (Fall 2016 Edition). Edward N. Zalta (ed.). <https://plato.stanford.edu/archives/fall2016/entries/value-theory>.
- Skytrax. "Awards Methodology". World Airport Awards, Skytrax, 2020. Accessed October 2020. <https://www.worldairportawards.com/awards-methodology/>.

- Skytrax “World’s Top 100 Airports 2020”. World Airport Awards, Skytrax, 2020. Accessed October 2020. <https://www.worldairportawards.com/worlds-top-100-airports-2020/>.
- Smith, Barry. “Objects and Their Environment”. *The Life and Motion of Socio-Economic* (GISDATA 8), London: Taylor and Francis, 2001, 79–97.
- Souza, Carlos Eduardo Gomes. “To cater for a passenger, you have to understand the passenger.” *International Airport Review*. July 23, 2020. <https://www.internationalairportreview.com/article/119756/cater-passenger-understand-passenger/>.
- Straube, J.F. *High Performance Enclosures*. Sommerville: Building Science Press, 2012.
- Sumers, Brian. “Airport Secrets from an Architect Who Designs Them.” Skift, February 6, 2018. <https://skift.com/2018/01/03/airport-secrets-from-an-architect-who-designs-them/>.
- TALUMIS. “Airport; Flexsim Simulation Model.” YouTube. YouTube, January 5, 2009. <https://www.youtube.com/watch?v=Elqx3u658tg>.
- Thalmann, Daniel, and Soraia Raupp Musse. *Crowd Simulation*. Vol. 9781447144502. London: Springer London, 2008.
- Tufte, Edward R. *The Visual Display of Quantitative Information*. Graphics Press USA, 2001.
- Turner, Alasdair, Maria Doxa, David Osullivan, and Alan Penn. “From Isovisists to Visibility Graphs: A Methodology for the Analysis of Architectural Space.” *Environment and Planning B: Planning and Design* 28, no. 1 (2001): 103–21. <https://doi.org/10.1068/b2684>.
- Tutorial Point. “Graph Theory.” Tutorials Point (I) Pvt. Ltd. 2020.
- Ulrich, Roger S., Craig Zimring, Xuemei Zhu, Jennifer DuBose, Hyun-Bo Seo, Young-Seon Choi, Xiaobo Quan, and Anjali Joseph. “A Review of the Research Literature on Evidence-Based Healthcare Design.” *HERD: Health Environments Research & Design Journal* 1, no. 3 (April 2008): 61–125. doi:10.1177/193758670800100306.
- Unity. “Architecture, Engineering & Construction.” Solutions. Accessed December 2019. <https://unity.com/solutions/architecture-engineering-construction>.
- Urban Strategies Inc, “Pearson Connects: A Multi-Modal Platform for Prosperity”, GTAA, February 2016, https://www.urbanstrategies.com/wp-content/uploads/2015/10/PearsonConnects_20160225.pdf
- Vaughan, Laura. “The Spatial Syntax of Urban Segregation.” *Progress in Planning* 67, no. 3 (2007): 205–294.
- Walker, Jarret. “Keys to Great Airport Transit”. *Human Transit*. 2016-03-01. <https://humantransit.org/2016/03/keys-to-great-airport-transit.html>.
- Walpole, Ronald E. *Probability & Statistics for Engineers & Scientists 9th ed*. Boston: Prentice Hall, 2012.
- Watkins, Joseph. *An Introduction to the Science of Statistics: From Theory to Implementation Preliminary Edition*. University of Arizona: 2016.
- WELL. “Concepts and Features.” WELL Certified v2, 2018. <https://v2.wellcertified.com/v/en/concepts>.

- Wilson, Robin J. *Introduction to Graph Theory 4th ed.* Harlow: Longman, 1996.
- Wired. "Airport Expert Creates the Ideal Layout for LaGuardia Airport (New York) | WIRED". Youtube. March 11, 2020. <https://www.youtube.com/watch?v=Kil-slXgVys>.
- Wiredja, Dedy, Vesna Popovic, and Alethea Blackler. "A Passenger-Centred Model in Assessing Airport Service Performance." *Journal of Modelling in Management* 14, no. 2 (May 10, 2019): 492–520.
- Wiredja, Dedy; Popovic, Vesna; Blackler, Alethea. "Questionnaire Design for Airport Passenger Experience Survey." *6th International Association of Societies of Design Research (IASDR) Conference*. (November 2015).
- Xie, Rong, and Yan Zhang. "Agent-Based Crowd Evacuation Modeling in Buildings." *Applied Mechanics and Materials* 411–414. (September 2013): 2639–42.
<https://doi.org/10.4028/www.scientific.net/amm.411-414.2639>.
- Yukio Futagawa, ed. *Global Architecture: TWA Terminal Building, Kennedy Airport, New York, and Dulles International Airport, Chantilly, Virginia*. Tokyo: A.D.A. Edita Tokyo, 1973. plan, p45.
- Zhou, Suiping, Dan Chen, Wentong Cai, Linbo Luo, Malcolm Low, Feng Tian, Victor Tay, Darren Ong, and Benjamin Hamilton. "Crowd Modeling and Simulation Technologies." *ACM Transactions on Modeling and Computer Simulation (TOMACS)* 20, no. 4 (October 1, 2010): 1–35.

Appendix A

The following pages lists the simulation trials from Part 4 Simulation Tests.

4.0 Verification and Validation Tests

IMO MSC - Guidance on Validation/Verification of Evacuation Simulation Tools												
Trial #	Pass/Fail (error)	Room Model	Agent Model	Agent Speed (float)	Navigation	Field of View Radius (m) / Angle (deg)	Time (s)	Time Error	Average Speed (m/s)	Speed Error	Date	Comments
Test 1 - Maintaining set walking speed in a corridor												
One person in a corridor 2 m wide and 40 m long with a walking speed of 1 m/s should be demonstrated to cover this distance in 40 s.												
1	Fail (-50%)	IMO-Test-1 (size scaled to Agent0) (4 m x 80 m)	Agent0 (capsule)	random (3 - 5)	A* Direct Path	---	20.20	50%	1.98	98%	2020-08-11	Scale of room was doubled to fit agent0, need to use properly proportioned person and room size
2	Pass (-2%)	IMO-Test-1 (size equal to dimensions) (2 m x 40 m)	Agent1 (average person)	1	A* Direct Path	---	39.25	2%	1.02	2%	2020-08-11	0.75 s time difference, 0.02 m/s speed difference, error comes from agent exiting when within 1 m of door
3	Pass (-2%)	IMO-Test-1 (2 m x 40 m)	Agent1	1	A* Direct Path	---	39.18	2%	1.02	2%	2020-08-11	
4	Pass (-2%)	IMO-Test-1	Agent1	1	A* Direct Path	---	39.29	2%	1.02	2%	2020-08-11	
5	Pass (+2%)	IMO-Test-1	Agent1	1	A* Perception	20 / 160	40.9	2%	0.98	2%	2020-08-11	agent reach door but did not despawn, started wandering close to the wall near the end of the corridor, causing its path to wandering back to avoid contact
6	Fail (—)	IMO-Test-1	Agent1	1	A* Perception	20 / 160	---	---	---	---	2020-08-11	agent never reached target, was not assigned correct exit, discovered door was embedded too far into wall to be detected
7	Pass (-2%)	IMO-Test-1	Agent1	1	A* Perception	20 / 160	39.07	2%	1.02	2%	2020-08-11	agent reach target, within correct time, there was no wandering motion when agent could see the door
8	Pass (-2%)	IMO-Test-1	Agent1	1	A* Perception	20 / 160	39.4	2%	1.02	2%	2020-08-11	
9	Pass (-1%)	IMO-Test-1	Agent1	1	A* Perception	20 / 160	39.6	1%	1.01	1%	2020-08-11	perception takes slightly longer than direct route, since there is slight pauses between choosing the next local target
10	Pass (-2%)	IMO-Test-1	Agent1	1	A* Perception	20 / 160	39.2	2%	1.02	2%	2020-08-11	reduced field of view radius to 10 m, no
11	Pass (-1%)	IMO-Test-1	Agent1	1	A* Perception	10 / 160	39.5	1%	1.01	1%	2020-08-11	
12	Pass (-1%)	IMO-Test-1	Agent1	1	A* Perception	10 / 160	39.59	1%	1.01	1%	2020-08-11	
13	Fail (-6%)	IMO-Test-1	Agent1	IMO Distributed Speeds	A* Perception	10 / 160	37.87	5%	1.06	6%	2020-08-11	Ran the average of 6 trials, agent speed now dependent on age
14	Pass (-4%)	IMO-Test-1	Agent1	IMO Distributed	A* Perception	10 / 160	38.41	4%	1.04	4%	2020-08-13	Sample size 25 people
15	Pass (+2%)	IMO-Test-1	Agent1	IMO Distributed	A* Perception	20 / 160	40.61	2%	0.99	1%	2020-08-13	Sample size 50 people
16	Fail (+21%)	IMO-Test-1	Agent1	IMO Distributed Speeds (including	A* Perception	20 / 160	48.26	21%	0.83	17%	2020-08-21	Sample size 100 people, now including walking disability
17	Pass	IMO-Test-1	Agent1	1	A* Perception	max range (100) / 120	40.02	0%	1.00	0%	2021-02-01	Re-ran normal test with 50 agent for graph
18												
19												

Trial #	Pass/Fail (error)	Room Model	Agent Model	Agent Speed (float)	Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Proximity Speed (m/s) / Radius (m)	Theoretical Flow (p/s)	Max Flow Rate (p/s)	Date	Comments
Test 4: Exit flow rate												
100 persons (p) in a room of size 8 m by 5 m with a 1 m exit located centrally on the 5 m wall. The flow rate over the entire period should not exceed 1.33 p/s.												
1	Fail	IMO-Test-4	Agent1	IMO Distributed	A* Perception	max range (100) / 160	data	0.2 / 2	1.99203187	4.2	2021-01-29	slower proximity speed took too long
2	Fail	IMO-Test-4	Agent1	IMO Distributed	A* Perception	max range (100) / 160	data	0.01 / 2	0.28135727	0.8	2021-01-29	Slightly narrower FOV angle
3	Fail	IMO-Test-4	Agent1	IMO Distributed	A* Perception	max range (100) / 120	data	0.2 / 2	1.98767641	3.2	2021-01-29	people clumping together near the end
4	Fail	IMO-Test-4	Agent1	IMO Distributed	A* Perception	max range (100) / 120	data	0.1 / 2	1.18301195	2.3	2021-01-29	better distribution, but people clumped together near the end
5	Fail	IMO-Test-4	Agent1	IMO Distributed Speeds	A* Perception	max range (100) / 120	data	0.05 / 2	0.82960013	2.4	2021-01-29	proximity radius smaller, people clumping at the end
6	Fail	IMO-Test-4	Agent1	IMO Distributed Speeds	A* Perception	max range (100) / 120	data	0.05 / 1.5	1.41442716	3.5	2021-01-29	lower flow rate, but jump near the end
7	Fail	IMO-Test-4	Agent1	IMO Distributed	A* Perception	max range (100) / 120	data	0.02 / 1.5	0.92387288	1.9	2021-01-29	better, but clump near end caused a
8	Fail	IMO-Test-4	Agent1	IMO Distributed	A* Perception	max range (100) / 120	data	0.01 / 1.5	0.77899821	1.5	2021-01-29	Tested with vector perception, agent moved faster
9	Fail	IMO-Test-4	Agent1	IMO Distributed Speeds	Vector Perception	max range (100) / 120	data	0.01 / 1.5	1.70910955	2.3	2021-01-30	Agents deadlocked near the beginning, then bunched together near end
10	Fail	IMO-Test-4	Agent1	IMO Distributed Speeds	Vector Perception	max range (100) / 120	data	0.01 / 2.0	0.67444527	2.9	2021-01-30	Max flow less than required 1.33 p/s, but not consistent flow rate
11	Pass	IMO-Test-4	Agent1	IMO Distributed Speeds	A* Perception	max range (100) / 120	data	0.01 / 1.6	0.63799923	1.2	2021-01-30	Same as previous test, one spike was higher than 1.33 p/s
12	Fail	IMO-Test-4	Agent1	IMO Distributed Speeds	A* Perception	max range (100) / 120	data	0.01 / 1.6	0.65338125	1.5	2021-01-30	Got less than 1.33 p/s, but rate was inconsistent
13	Pass	IMO-Test-4	Agent1	IMO Distributed Speeds	A* Perception	max range (100) / 120	data	0.01 / 1.6	0.58819796	1.1	2021-02-01	
14												
15												
16												
Trial #	Pass/Fail (error)	Room Model	Agent Model	Agent Speed (float)	Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Average Time (s)	Average Speed (m/s)	Speed Error	Date	Comments
Test 6: Rounding corners												
Twenty persons approaching a left-hand corner (see figure 1) will successfully navigate around the corner without penetrating the boundaries.												
1	Pass	IMO-Test-6 (Based on figure 1)	Agent1	IMO Distributed Speeds	A* Perception	max range (100) / 160	data	27.66			2020-10-01	Passed, agents walked around corner to the exit without going through boundaries
2	Pass	IMO-Test-6	Agent1	IMO Distributed	A* Perception	max range (100) / 160	data	30.80			2020-10-01	
3	Pass	IMO-Test-6	Agent1	IMO Distributed	A* Perception	max range (100) / 160	data	27.29			2020-10-01	
4	Pass	IMO-Test-6	Agent1	IMO Distributed	A* Perception	max range (100) / 160	data	29.10			2020-10-01	
5												
Trial #	Pass/Fail (error)	Room Model	Agent Model	Agent Speed (float)	Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Average Time (s)	Average Speed (m/s)	Speed Error	Date	Comments
Test 7: Assignment of population demographics parameters												
Choose a panel consisting of males 30-50 years old from table 3.4 in the appendix to the Guidelines for the advanced evacuation analysis of new and existing ships and distribute the walking speeds over a population of 50 people. Show that the distributed walking speeds are consistent with the distribution specified in the table.												
1	Pass (+2%)	IMO-Test-7 (20m x 20m)	Agent1	0.97 to 1.62	A* Perception	max range (100) / 160	data	15.63	1.32	2%	2020-10-08	
2	Pass (-3%)	IMO-Test-7	Agent1	0.97 to 1.62	A* Perception	max range (100) / 160	data	16.43	1.26	-3%	2020-10-08	
3	Pass (0%)	IMO-Test-7	Agent1	0.97 to 1.62	A* Perception	max range (100) / 160	data	16.04	1.29	0%	2020-10-08	
4	Pass (-1%)	IMO-Test-7	Agent1	0.97 to 1.62	A* Perception	max range (100) / 160	data	16.05	1.28	-1%	2020-10-08	
5	Pass (+3%)	IMO-Test-7	Agent1	0.97 to 1.62	A* Perception	max range (100) / 160	data	15.29	1.34	3%	2020-10-08	
6	Pass (-3%)	IMO-Test-7	Agent1	0.97 to 1.62	A* Perception	max range (100) / 160	data	16.26	1.26	-3%	2020-10-08	
Average	Pass (0%)								1.29	0%	2020-10-08	

4.1 Component Tests

Component Testing														
Trial #	Room Model	Agent Model	Agent Walking Speed (m/s)	Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Number of Agents	Assigned Gate	Average Time (s)	Average Distance (m)	Max Distance (m)	Min Distance (m)	Date	Comments
Wayfinding														
Testing the direction 10 agents follow to navigate to Gate B, comparing direction and perception navigation														
1	Wayfinding	Agent1	1	A* Direct	max range (100) / 160	data	5	Gate B	60.8	60.83	60.75	60.75	2021-02-03	Agents followed the same paths
2	Wayfinding	Agent1	1	A* Direct	max range (100) / 160	data	10	Gate B	60.85	60.91	60.82	60.82	2021-02-03	
3	Wayfinding	Agent1	1	A* Direct	max range (100) / 160	data	50	Gate B	61.83	62.5	61.35	61.35	2021-02-03	
4	Wayfinding	Agent1	1	A* Perception	max range (100) / 160	data	50	Gate B	101.06	111.47	95.1	95.1	2021-02-03	Some agents walked into a corner, but eventually managed to find the gate
5	Wayfinding	Agent1	1	A* Perception	max range (100) / 160	data	50	Gate B	101.95	113.69	92.67	92.67	2021-02-03	
6	Wayfinding	Agent1	1	A* Perception	max range (100) / 160	data	50	Gate B	102.18	111.5	94.14	94.14	2021-02-03	
7														
8														
Trial #	Room Model	Agent Model	Agent Walking Speed (m/s)	Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Number of Agents	Direction	Average Visibility	Max FOV Area (m²)	FOV Ratio when gate discovered		Date	Comments
Visibility														
Determining the visibility moving between a narrow space and a wide space														
1	Visibility (ver.1)	Agent1	1	A* Perception	max range (100) / 160	data	1	to narrow	0.3958	284.15			2021-02-03	agents sometimes double back on themselves
2	Visibility (ver.1)	Agent1	1	A* Perception	max range (100) / 160	data	1	to wide	0.6751	301.51			2021-02-03	lost data
3	Visibility (ver.2)	Agent1	1	A* Perception	max range (100) / 160	data	1	to wide	0.6651	364.21	1		2021-02-03	
4	Visibility (ver.2)	Agent1	1	A* Perception	max range (100) / 160	data	1	to narrow	0.3824	544.5	0.1249		2021-02-03	agents sometimes cannot find narrow corridor
5	Visibility (ver.3)	Agent1	1	A* Perception	max range (100) / 160	data	1	to narrow	0.4152	542.68	0.1107		2021-02-04	
6	Visibility (ver.3)	Agent1	1	A* Perception	max range (100) / 160	data	1	to wide	0.6978	598.98	1		2021-02-04	
7	Visibility (ver.3)	Agent1	1	A* Perception	max range (100) / 160	data	50	to wide	0.7016	591.27	0.8878		2021-02-04	Ran at 3x time scale
8	Visibility (ver.3)	Agent1	1	A* Perception	max range (100) / 160	data	50	to narrow	0.42	617.19	0.1206		2021-02-04	some agents saw 1200 m² max area and last ratio of 1.0. (not same agents though)
9	Visibility (ver.3)	Agent1	1	A* Perception	max range (100) / 160	data	50	to narrow	0.4177	631.05	0.0855		2021-02-04	some outliers with max area and one with last ratio
10	Visibility (ver.3)	Agent1	1	A* Perception	max range (100) / 160	data	50	to narrow	0.4176	639.24	0.1031		2021-02-04	1200 m² is not an error, occurs if agent walks all the way to the bottom corner in wide area and look across to narrow side (i.e. they can see all of the wide side and towards narrow corridor 600 + 600 = 1200)
12														
13														
Trial #	Room Model	Agent Model	Agent Walking Speed (m/s)	Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Number of Agents	Non-processing Domain	Priority Threshold	Agents Above Priority Threshold	Agents Below Priority Threshold	Agents Participated in Non-processing Domain	Date	Comments
Non-Processing Priorities														
Checks if agents with high priorities for a non-processing airport domain (food/retail) follow different behaviour than agents with low priorities														
1	Non-Processing Priorities	Agent1	1	A* Perception	max range (100) / 160	data	10	Food Availability	5	6	4	6	2021-02-05	
2	Non-Processing Priorities	Agent1	1	A* Perception	max range (100) / 160	data	50	Food Availability	5	25	25	25	2021-02-05	
3	Non-Processing Priorities	Agent1	1	A* Perception	max range (100) / 160	data	50	Food Availability	5	30	20	23	2021-02-05	Departure time occurred before some agents had time to get food
4	Non-Processing Priorities	Agent1	1	A* Perception	max range (100) / 160	data	50	Food Availability	5	25	25	25	2021-02-05	
5	Non-Processing Priorities	Agent1	1	A* Perception	max range (100) / 160	data	50	Food Availability	5	28	22	28	2021-02-06	
6														

4.2 Terminal Tests

Terminal Tests																			
Trial #	Room Model	Agent Model	Agent Walking Speed	Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Number of Agents	Assigned Priorities	Average Architectural Value	Average Check in Value	Average Food Availability Value	Average Gate Availability Value	Average Restroom Availability Value	Average Security Screening Value	Average Waiting Seating Value	Average Time (s)	Average Visibility	Date	Comments
Centre-Random																			
Centre security layout with randomly assigned priorities																			
1	Terminal Model 2	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Random	0.754	0.729	0.76	0.276	1	0.6245	0.92	479.53	0.3622	2021-02-10	
2	Terminal Model 2	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Random	0.7217	0.7152	0.75	0.2708	1	0.6078	0.9	455.9	0.3645	2021-02-10	
3	Terminal Model 2	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Random	0.7317	0.7384	0.81	0.248	1	0.6217	0.92	448.9	0.398	2021-02-10	
4																			
5																			
6																			
Centre-High																			
Centre security layout with highest priority for security																			
1	Terminal Model 2	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Security: 5; Else: 1	0.6335	0.7206	0.5	0.2465	1	0.5998	1	485.69	0.1421	2021-02-10	
2	Terminal Model 2	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Security: 5; Else: 1	0.6451	0.74	0.5	0.2039	1	0.6318	0.9	478.19	0.1652	2021-02-17	
3																			
4																			
5																			
6																			
Centre-Equal																			
Centre security layout with equal priorities																			
1	Terminal Model 2	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Equal: 1	0.6871	0.7442	0.5	0.2493	1	0.6238	1	487.72	0.1525	2021-02-10	Although priorities were equal, agents did not get food since value was 1
2	Terminal Model 2	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Equal: 5	0.7632	0.7366	1	0.2609	1	0.5964	0.98	457.97	0.4284	2021-02-11	
3	Terminal Model 2	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Equal: 5	0.7762	0.7463	1	0.2762	1	0.6335	1	497.95	0.5597	2021-02-17	
4																			
5																			
6																			
Asymmetrical-Random																			
Asymmetrical security layout with randomly assigned priorities																			
1	Terminal Model 3	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Random	0.7242	0.693	0.79	0.2947	1	0.4979	0.9	503.26	0.4902	2021-02-10	
2	Terminal Model 3	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Random	0.717	0.685	0.74	0.2944	1	0.5119	0.96	541.85	0.4405	2021-02-15	
3																			
4																			
5																			
6																			
Asymmetrical-High																			
Asymmetrical security layout with highest priority for security																			
1	Terminal Model 3	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Security: 5; Else: 1	0.57	0.689	0.5	0.2976	1	0.5058	0.94	476.39	0.4837	2021-02-10	
2	Terminal Model 3	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Security: 5; Else: 1	0.5744	0.7153	0.5	0.3011	1	0.5205	0.84	504.47	0.5731	2021-02-17	
3																			
4																			
5																			
6																			
Asymmetrical-Equal																			
Asymmetrical security layout with equal priorities																			
1	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Equal: 5	0.7277	0.7103	1	0.2693	1	0.5229	0.86	549.99	0.4748	2021-02-11	
2	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Equal: 5	0.7289	0.7148	1	0.2563	1	0.5413	0.86	503.74	0.4491	2021-02-17	
3																			
4																			
5																			
6																			

Trial #	Room Model	Agent Model	Agent Walking Speed	Agent Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Number of Agents	Assigned Priorities	Average Architectural Value	Average Check in Value	Average Food Availability Value	Average Gate Availability Value	Average Restroom Availability Value	Average Security Screening Value	Average Walking Seating Value	Average Time (s)	Average Velocity	Date	Comments
Perpendicular-Random																			
Perpendicular security layout with randomly assigned priorities																			
1	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	47	Random	0.657	0.701	0.766	0.2413	0.9149	0.5293	0.6383	535.64	0.4727	2021-02-10	3 agent got stuck in seating area and failed to reach gates.
2	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	49	Random	0.6367	0.7172	0.6429	0.2171	0.888	0.487	0.6327	514.08	0.4079	2021-02-10	1 agent got stuck in seating area and failed to reach gates.
3	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	49	Random	0.6512	0.714	0.6837	0.2523	1	0.49513	0.6939	457.52	0.3937	2021-02-10	1 agent got stuck in seating area and failed to reach gates.
4	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Random	0.6787	0.7058	0.81	0.2766	0.78	0.4735	0.92	529.34	0.4522	2021-02-16	No security sign from now on; did wash room sign was not visible
5	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Random	0.7033	0.6934	0.78	0.2639	1	0.483	0.9	518.54	0.4212	2021-02-16	
6																			
Perpendicular-High																			
Perpendicular security layout with highest priority for security																			
1	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	47	Security-9; Else: 1	0.55	0.732	0.5	0.2454	0.9149	0.4856	0.982	445.43	0.3887	2021-02-10	3 agent got stuck in seating area and failed to reach gates.
2	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	49	Security-9; Else: 1	0.529	0.734	0.5	0.2245	1	0.4718	0.7551	555.14	0.4232	2021-02-17	No security sign from now on; 1 agent got stuck in seating area and failed to reach gates.
3	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	48	Security-9; Else: 1	0.5211	0.6981	0.5	0.2291	1	0.4783	0.5825	466.36	0.3723	2021-02-17	2 agents got stuck in seating area and failed to reach gates.
4	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	49	Security-9; Else: 1	0.5552	0.7231	0.5	0.264	1	0.4851	0.9184	497.67	0.3396	2021-02-17	1 agent got stuck in seating area and failed to reach gates.
5																			
6																			
Perpendicular-Equal																			
Perpendicular security layout with equal priorities																			
1	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Equal 5	0.7442	0.6988	1	0.3011	1	0.4977	0.96	448.68	0.5552	2021-02-11	
2	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Equal 5	0.7056	0.7212	0.9	0.3094	1	0.4548	0.84	510.9	0.4863	2021-02-17	No security sign from now on
3	Terminal Model 4	Agent1	IMO Distributed	Vector Perception	max area (100) / 160	data	50	Equal 5	0.7307	0.7265	0.98	0.2895	1	0.4923	0.88	502.81	0.4802	2021-02-17	
4																			
5																			
6																			

4.3 Airport Tests

Airport Test																	
Trail #	Room Model	Agent Model	Agent Walking Speed	Navigation	Field of View Radius (m) / Angle (deg)	Link to Data	Number of Agents	Average Architectural Value	Average Check In Value	Average Gate Value	Average Food Value	Average Gate Seating Value	Average Restroom Value	Average Security Value	Average Time (Seconds)	Date	Comments
Test 1: Singapore Changi																	
Terminal 1 check-in and security process for 100 passengers.																	
1	Singapore Changi	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.7704	0.55	0.503	1	1	1	0.3822	313.01	2020-10-05	
2	Singapore Changi	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.7858	0.6056	0.5052	1	1	1	0.3461	299.93	2020-10-06	
3	Singapore Changi	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.7801	0.6001	0.503	1	1	1	0.3603	308.07	2020-10-06	
4	Singapore Changi	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.8018	0.6792	0.5252	1	1	1	0.4431	319.81	2020-10-07	
5																	
6																	
Test 2: Toronto Pearson																	
Terminal 1 check-in and security process for 100 passengers.																	
1	Toronto Pearson	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.4524	0.5106	0.4552	0.16	1	0.15	0.3967	207.86	2020-10-04	Disabled A* navigation because agents were getting stuck at corners of curved walls and simulation was performing slowly due to calculating path in large space. Replaced with walling with vector perception
2	Toronto Pearson	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.4882	0.5137	0.4571	0.24	1	0.24	0.4082	179.83	2020-10-04	
3	Toronto Pearson	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.4251	0.5153	0.4524	0	1	0	0.4809	129.53	2020-10-05	Added sign to security screening, agents never walk over to food and washroom area
4	Toronto Pearson	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.3961	0.5068	0.3147	0.02	1	0.02	0.4897	202.38	2020-10-06	Moved gate further down corridor
5	Toronto Pearson	Agent1	IMO assigned	Vector Perception	max range (100) / 160	data	100	0.4203	0.574	0.3351	0.03	1	0.03	0.5549	248.05	2020-10-06	Last three agents got lost trying to get to gate, had to re-orient them
6																	
7																	

