Physically-based Simulation of Tornadoes

by

Xiangyang Ding

A thesis

presented to the University of Waterloo

in fulfilment of the

thesis requirement for the degree of

Master of Mathematics

in

Computer Science

Waterloo, Ontario, Canada, 2004

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.

Xiangyang Ding

Abstract

In this physically-based tornado simulation, the tornado-scale approach techniques are applied to simulate the tornado formation environment. The three-dimensional Navier-Stokes equations for incompressible viscous fluid flows are used to model the tornado dynamics. The boundary conditions applied in this simulation lead to rotating and uplifting flow movement as found in real tornadoes and tornado research literatures. Moreover, a particle system is incorporated with the model equation solutions to model the irregular tornado shapes. Also, together with appropriate boundary conditions, varied particle control schemes produce tornadoes with different shapes. Furthermore, a modified metaball scheme is used to smooth the density distribution. Texture mapping, antialising, animation and volume rendering are applied to produce realistic visual results. The rendering algorithm is implemented in OpenGL.

Acknowledgements

The time as a graduate student at the University of Waterloo has been the best period of my life since I came to Canada, and for that wonderful experience I owe many thanks.

I had one excellent advisor, Dr. Justin Wan. I became to be interested in scientific computing after I took Dr. Wan's $CS370$ class - "Introduction to Scientific Computation." His teaching and advices at that time inspired me choosing this tornado simulation topic which is both interesting and challenging. Dr. Wan have had a profound influence on every process of this project. His guidance and supervision keep my research on the right direction and help me find out the appropriate solutions in modeling, rendering and animation. He also treated me as an equal researcher who is just less experienced and needs to be taught about problem-solving techniques. Dr. Wan also gave me freedom in implementation, choosing OpenGL in the rendering and parameter selection in animation. In addition, he encouraged me to think creatively to find out the most appropriate boundary conditions and improve the visual effect. Dr. Wan's advices are critical to my studies, academic writing and the success of this simulation project. Thanks for all of these, Justin!

I took the course $CS888$ – "Simulation of Natural Phenomena" from Dr. Gladimir Baranoski. The knowledge I learned from him, such as the simulation constraints and trade-offs, is very important to the success of this project. His experience in natural phenomena simulation, especially aurora simulation, is invaluable for me to choose the best parameters in rendering and animation. Also Dr. Baranoski's stamina in aurora simulation encouraged me to work hard and always made me confident. Moreover, I

thank Dr. Baranoski's referral on the cloud simulation paper which enlightened me in the density calculation and rendering steps in this tornado simulation. In addition, Dr. Baranoski's support on my Curupira account was very helpful. Finally, thanks for the nickname – "Mr. Tornado," I like it very much.

I would like to thank Dr. Bruce Simpson for his interests in this project and agreeing to read this thesis. My simulation knowledge and mesh generation techniques learned from him in $CS476$ and $CS870$ improve my background in scientific computing and are very helpful to this simulation project. Moreover, I thank him for his humour and patience in our talks and meetings.

I thank my family, especially my parents and my wife - Chunhui, for their encouragement and support. Thanks for Chunhui's comments on the simulated features. I owe her a lot of time to be together.

Finally, I thank all the friends in Scientific Computing group for their interests in this project, especially Shannon for his help on Latex and Amelie for her technical support on my machine - Scicom8.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As one of the severe phenomena in the world, "A tornado (from the Latin `tornare`, 'to turn') is the most violent storm nature produces" [13]. Tornadoes cause incredible amounts of damage and significant numbers of fatalities every year, thus making this phenomenon a major research topic. In this thesis, a physically-based tornado simulation is presented to produce tornadoes with interesting shapes and colors. Moreover, another research focus in this tornado simulation is to study tornado formation environment, *i.e.*, how boundary conditions are related to tornado shape evolution. The simulation is the combination of the numerical simulation and graphics rendering. The numerical simulation step is to simulate the general tornado flow movement, *i.e.*, rotating on the horizontal planes and uplifting on the vertical planes. A particle system is introduced to define tornado volume and a volume rendering algorithm is used to visualize the flow movement based on the particle density.

The introduction of this thesis is discussed in this chapter. To be specific, Section

1.1 will introduce the motivation of applying numerical simulation and graphics in this tornado simulation, Section 1.2 will talk about the general framework applied, and Section 1.3 will discuss the outline of this thesis.

1.1 Motivation

1.1.1 Why is numerical simulation used in this thesis

Before I discuss the reasons to apply numerical simulation in this thesis, I give a brief discussion about other tornado study methods, including their advantages and drawbacks. The first method to be introduced is tornado-chasing. Tornado-chasing has become an important activity for many scientists and weather enthusiasts to observe the tornado dynamics and collect the real data. The tornado videos recorded by tornado chasers offer us further understandings about its appearance and formation. Unfortunately, due to the difficulties to put the data recording equipments into the tornado path, the real data collection for tornado's internal dynamics study is still not much successful today [4]. Instead of chasing tornadoes, on the other hand, satellite and radar images are also used to understand the flow dynamics of tornadoes [4]. But the cost is high in order to build enough satellites and radars to monitor tornadoes. Moreover, some researchers create their own "twisters" in laboratories [26] to study tornadoes. Tornadoes with different rotation speeds can be simulated by varying the apparatus condition, such as fan speed. The experiments conducted on the man-made tornadoes help scientists study tornadoes and understand their nature, for instance the destructive force and the formation process.

But this method is also cost expensive due to the fact that the simulation apparatuses need to be built or rebuilt.

In addition to the field and laboratory observations [4, 5, 8, 9, 13, 26, 31], mathematical modeling [19, 22, 27, 30, 34] and numerical simulation [12, 23, 28, 33, 36, 38] have been developed for various aspects of tornado studying. For example, applied originally in the simulation of other phenomena, such as turbulence [27, 28], cloud [23] and fluid flow [34, 38], varied models are researched to study tornado's physical properties. Some models [30] are in cylindrical coordinates (2D) and assume that tornadoes are in axisymmetric internal structure while some others [38] are in the cube domain (3D) with the emphasis on asymmetric flow study. At the same time, different numerical methods, such as finite difference method and finite volume method, are applied to solve the model equations. Such modeling and simulation research lead to better understandings of tornado flow structures.

Although numerical simulation[1] has some disadvantages, such as the difficulties to find the accurate model to represent the simulated phenomena, but compared with other tornado research methods discussed earlier, numerical simulation has several apparent advantages. First, numerical simulation has no risks. What the researcher needs to do is to write computer programs to form and solve the mathematical equations instead of going to the field to chase tornadoes as a tornado chasing team does. Second, numerical simulation has lower money expenditure. For example, generally speaking, a computer with high computing performance is enough for numerical simulation; but in order to use satellite and radar images for tornado research, more money needs to be spent in building

[1]Mathematical modeling is included in numerical simulation.

satellites and radars, especially more than one satellites or radars are usually needed. The cost problem is the same in laboratory experiments where simulation apparatuses need to be built, and even rebuilt when a different tornado formation environment is simulated. Third, the tornado formation environment is easily to be studied in numerical simulation. For instance, we can test the model and research the tornado formation environment by applying varied boundary conditions in the simulation. The user just needs to modify the code of the computer programs without worrying about the replacement of the machine parts which maybe happen in laboratory experiments. Finally, numerical simulation can be done within a short period of time, such as several months or one year, but to build satellites, radars and laboratory equipments would be more time expensive. All these advantages make numerical simulation an important research approach in tornado studies.

In addition, compared with other tornado study methods, to use numerical simulation techniques for my tornado flow research is the reasonable choice because of the time limit and cost limitation in my graduate study. The numerical solutions of the physical model is used to produce the general flow movement observed in real tornadoes, *i.e.*, rotating and uplifting. It is assumed that realistic flow modeling can lead to realistic tornado flow movement in the final rendered images.

1.1.2 Why is graphics applied

As discussed in the previous section, numerical simulation is one important research method in tornado studies. But the numerical simulation results in tornado research

literature focused mainly on the velocity, pressure and temperature fields without a simulated realistic tornado image to show its movement, shape and color, thus making the simulation less visually interesting. So in this thesis, in addition to the simulated tornado flow, tornadoes with interesting shapes and colors are also produced for realistic visual results. To do this, graphics techniques are employed to visualize the tornado flow generated in the numerical simulation step.

It needs to be noted that the graphics is combined with physically-based modeling in this thesis, but it is not in this case in the tornado studies. For example, created by using pure computer graphics techniques in entertainment field, visually appealing tornado images are widely used on TV programs and films, such as `TWISTER` [10], but their purposes are for entertainment only without taking into account the tornado physical dynamics.

We can see that the two approaches in natural phenomena simulation, numerical simulation and graphics, emphasize either physics or visual results, same as their applications in tornado simulation. As pointed out by Barr *et al.* [2] below, the ideal approach in natural phenomena simulation is to combine both of them:

> These two approaches have begun to reach the limits of their ability without each other. The realism required in entertainment animation is beyond that obtainable without physically realistic models. Similarly, numerical simulations in the physical sciences are complex to the point of being incomprehensible without visual representations.

In this thesis, a physically-based tornado simulation is presented to combine the ad-

vantages of both numerical simulation and graphics rendering to avoid the either "pure simulation" or "pure animation" problem discussed in [2]. The numerical simulation techniques produce tornado flow simulation results from the physically modeling, and graphics rendering is used to visualize the flow movement. As the result of the combination, the user can see the simulated tornado flow rotating and going up. Also, with the application of animation, we can understand the tornado flow movement further. To the best of my knowledge, the combination of numerical simulation and graphics rendering has never been done in tornado studies. My intention is to provide some understandings about tornado flow generation and visualization to tornado researchers. I would be very happy if my work could contribute to the tornado research at some extent.

1.2 Preview

In this physically-based tornado simulation, the tornado-scale approach techniques [30] are applied to simulate the tornado producing environment. This approach has been widely used in laboratory experiments [6, 24, 33, 41] and numerical simulations [12, 27, 30, 36, 38] for tornado studies. The three-dimensional Navier-Stokes equations for incompressible viscous fluid flows are used in this thesis to model the tornado dynamics. The boundary conditions applied in this simulation lead to rotating and uplifting flow movement as found in real tornadoes and tornado research literatures.

To model the irregular tornado shapes, a particle system is incorporated with the model equation solutions. As will be discussed later in Section 4.1.2, to control the number and initial positions of particles generated at each time step is rather difficult due

to the following facts. First, the number of particles will influence the particle density in the domain, then the transparency and the final rendered image, so a good control scheme on the number of particles generated will give nice color variation effect as seen in real tornadoes. Second, the particle position attribute can change the shape of visualization results. For example, if the flow movements at certain places in the domain are part of tornado shape evolution process, but there are no or not enough particles at those locations, then some parts of the tornado shape will be missing in the final tornado images. As presented in Figure 5.2(a) and (b) in Section 5.2, with the same boundary conditions, different control schemes on the number of particles generated at each step produce tornadoes with different shapes. Moreover, gravity and centrifugal force are applied to some particles in this simulation to simulate the bottom debris swirling effect. In the bottom debris swirling simulation, to produce different particle motion paths from what other particles follow is also very challenging. Therefore, significant amount of time is spent on the parameter selection in the particle system in order to produce the realistic results.

In addition, a modified metaball scheme is applied in particle density calculation to smooth the density distribution. The metaball radius and interpolation functions used in this simulation lead to a continuous density distribution with the efficient computing speed at the same time. Furthermore, the density level discretization strategy applied subdivides the densities into certain levels with the time efficiency consideration as well. As a result, the density differences between levels are retained for color variation purpose in the rendering step.

Finally, the volume rendering algorithm produces realistic tornado images. Some

graphics techniques, such as texture mapping, antialising and animation, are applied for better visual results. Moreover, the blending factors used in OpenGL implementation achieve good blending effects in the combination of the object color with that of the background.

As the results demonstrated in Section 5.2, because the axisymmetric tornado structure is not assumed in the modeling, tornadoes with different shapes are produced. The comparison between a simulated tornado and a real tornado shows that the model and boundary conditions applied in this simulation can generate a tornado with a similar shape as a real tornado has. Moreover, gravity and centrifugal force are applied to simulate the bottom debris swirling effect. The simulated debris movement is qualitatively compared with a real tornado. Furthermore, the simulated dynamic shape change effect in a tornado presented in Figure 5.4 is also consistent with the shape change observation in a real tornado video.

1.3 Outline

In the following chapters, this tornado simulation will be discussed in details. First, Chapter 2 gives a brief overview about tornado phenomenon, especially its physical characteristics closely related to this simulation. The comparisons of tornadoes with hurricanes and typhoons are also discussed, as well as the difficulties of tornado studies. Then in Chapter 3, the details of modeling and simulation are presented. It starts with the literature review and the introduction of simulation approaches generally used in tornado simulation. Then the model and the boundary conditions applied in this simulation are discussed, as

well as the numerical techniques used to solve the equations. Next, the particle system is introduced in Chapter 4. The metaball scheme and rendering algorithm are also covered. After that, validation will be discussed in Chapter 5, as well as the presentation of the simulation results. The comparisons between the simulations and real tornadoes will also be presented. Finally, this physically-based tornado simulation will be summarized in Chapter 6, and the limitations and future work will be discussed.

Chapter 2

Tornado Phenomenon

The introduction of tornado phenomenon is presented in this chapter. Firstly, I discuss the characteristics of a tornado in Section 2.1, as well as the comparisons of a tornado with other two closely related phenomena - hurricane and typhoon. Secondly, in Section 2.2, the difficulties of tornado research are discussed.

2.1 Tornado Characteristics

Before the introduction of tornado phenomenon, a brief overview about other two violent storms in the nature, hurricane and typhoon, will be given. These two phenomena have many similarities with tornadoes, such as the rotation and violence. But the most important reason to discuss them is that the tornado formation process is closely related to hurricane and typhoon. For example, many tornadoes are actually spawned by hurricanes or typhoons [21].

In the `Glossary of Meteorology` [16], hurricane, typhoon and tornado are all classified under the same term - "cyclone". Cyclone is defined as "a large-scale circulation of winds around a central region of low atmospheric pressure, counter-clockwise in the northern hemisphere and clockwise in the southern hemisphere" [44]. One kind of cyclone is the tropical cyclone which "originates over" the tropical oceans [16]. The terms "hurricane" and "typhoon" are *regionally* specific names for a strong tropical cyclone [20]. To be specific, hurricane is defined as a tropical cyclone with wind speed in excess of 32 $m/s(70mph)$ in the Western hemisphere (North Atlantic Ocean, Caribbean Sea, Gulf of Mexico, and in the Eastern and central North Pacific) [16]. And typhoon is an identical phenomenon as a hurricane but it occurs in the Eastern hemisphere [16, 21]. From their definitions, we can see that hurricane and typhoon are regional phenomena which are very different from a tornado. Tornado is defined as an "violently rotating column of air, in contact with the ground, either pendant from a cumuliform[1] cloud or underneath a cumuliform cloud" [16]. The wind rotation speed can be as low as 18 m/s (40 mph) to as high as 135 m/s (300 mph), and the forward motion speed of a tornado averages around 40 mph [13], but can be as high as 68 mph. In addition to the strong vortical motion in a tornado, the "up-rushing currents of lifting strength" [13, 35] is also an important characteristic of it. The typical tornado flow structure is described in [28] as "the flow spirals radially inward into the center of the vortex that is basically a swirling and rising jet". A schematic of such vortical and uplifting flows is shown in Figure 2.1.

As tornado is clearly defined as contacting with the ground [16], they are primarily

[1]Generally descriptive of all clouds, the principal characteristic of which is vertical development in the form of rising mounds, domes, or towers [16].

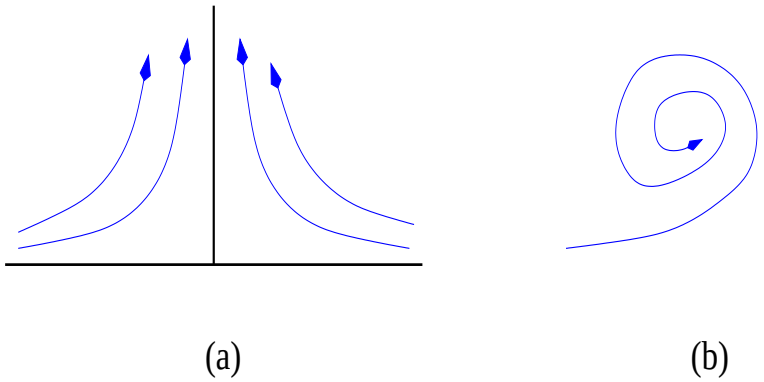


Figure 2.1: The projections of vortex streamlines onto: (a) the vertical plane through the axis of the vortex, (b) the horizontal plane. (Redrawn from [21]).

an "over-land phenomena" while on the other hand, hurricane and typhoon are purely an "oceanic phenomena" because they finally disappear on the land where the "moisture source" does not exist [31]. Moreover, like hurricanes and typhoons, tornadoes are primarily in the summer, but in theory, tornadoes can occur at any time of the day throughout the year. Furthermore, tornadoes can happen on all continents and are only minutes long while hurricanes and typhoons can only happen in some specific regions and can last for days [16]. As a global phenomenon, tornadoes have been reported in many countries in the world, such as Canada, England, France, Holland, Germany, Hungary, Italy, Australia, India, Russia, China, Japan, even Bermuda Islands and Fiji Islands [13]. However, they are most frequent in spring and summer during the late afternoon on the central plains and in the Southeastern states in the United States [13, 16, 36]. Each year an average of a thousand tornadoes are reported in the United States [36]. In Canada, tornadoes are more frequently seen in the Southern provinces and Southwestern Ontario [9]. On average, about 60 tornadoes are reported each year in Canada.

Table 2.1: The Fujita Scales with intensity and wind speed (Redrawn from Glossary of Meteorology [16])

F-Scale Number	Intensity Phrase	Wind Rotaion Speed
F0	Gale Tornado	40-72 mph
F1	Moderate Tornado	73-112 mph
F2	Significant Tornado	113-157 mph
F3	Severe Tornado	158-206 mph
F4	Devastating Tornado	207-260 mph
F5	Incredible Tornado	261-318 mph
F6	Inconceivable Tornado	319-379 mph

Compared with other storms in the nature, a tornado is the most violent one [8, 26]. It is estimated that tornadoes cause about one billion dollars of damage per year in the United States. The destructive rotation in a tornado is the main cause for such violence and damage. Theodore Fujita and Alan Pearson [25] first linked the tornado rotation speed with the amount of damage it produced in 1971. The system they devised, called Fujita-Pearson scale, is widely used as a standard system to classify the intensity of a tornado. As shown in Table 2.1, this system rates tornadoes by 7 classes. Tornadoes with intensity scale F0 - F1 are weak, F2 - F3 are strong, and above F4 are violent. In Canada, no F5 tornadoes have ever been recorded, but a number of F4 events have been [9].

Some important physical characteristics directly related to the simulation of tornado dynamics are summarized in the following categories:

- rotation: the rotation direction of a tornado is counter-clockwise in the Northern hemisphere and clockwise in the Southern hemisphere [13]. The simulation results in Section 5.2 show that the counter-clockwise rotation in the Northern hemisphere

can be successfully produced by using the model and specific boundary conditions applied in this simulation. With the corresponding changes on the boundary conditions, the model used can be easily applied in the simulation of the clockwise rotation in the Southern hemisphere.

- shape: a tornado generally takes the shape of a cone or a cylinder or of a funnel in which tornado is equal or greater in diameter at the top than at the bottom [13]. But a tornado with funnel shape is the most common one [13]; see Figure 2.2(a) for an example of funnel-shape tornadoes. Such funnel shape is the result of the "rotary motion of extremely high speed" [5]. Tornadoes, however, can have some other shapes as well. For example, a tornado can be greater at the bottom [4, 13, 37]. And a tornado can also look like a snake or rope [4, 47] and even makes a "right-angle bend" [13] as it nears the ground. Figure 2.2(b) shows an example of bended rope-like tornadoes. The animation frames demonstrated in Section 5.2 show the simulation results of a funnel-shape tornado and a bended tornado. Moreover, after centrifugal force and gravity are applied in the particle system, the bottom debris swirling effect can be simulated. As presented in Figure 5.4, the simulated arrow shape tornado is qualitively similar to a real tornado.

- color: Laycock [26] describes that a tornado is generally gray or dark. When we see a tornado, what we are really seeing is the water vapor inside the wind or the sucked objects by the swirling strength because the wind itself is invisible [26]. If there is only water vapor in the wind, the color will be gray [26]. But if some objects, such as dirt, leaves and bark, are drawn to the wind, the tornado will

(a) (b)

Figure 2.2: (a) A tornado with a funnel shape on the plains of North Dakota, USA. (http://ww2010.atmos.uiuc.edu/(Gh)/guides/mtr/svr/torn/home.rxml) (Courtesy of National Severe Storms Laboratory (NSSL), USA). (b) A tornado in USA with an unusual shape making a right-angle bend in the middle. (http://www.weatherstock.com/tornadocat3.html) (Courtesy of the Weatherstock, USA).

be dark [26]. It should be noted that not all tornadoes are visible, for example, when a tornado moves through "comparatively dry and clean air" it is invisible [13]. But the visible tornadoes are more interesting to viewers, so to produce a tornado with color is one of our goals in the simulation for the best realistic simulation results. The volume rendering algorithm used in this simulation produces visual response with interesting colors as the observed real tornadoes. As shown in Section 5.2, the rendering algorithm applied can produce gray color and darker tornadoes. Tornadoes with other colors can also be simulated by choosing different backgrounds and wind colors.

2.2 Tornado Research Difficulties

The study on tornado formation and its dynamics is always a challenging topic. Scientific research tells us that tornadoes are from the energy released in a thunderstorm and they usually occur under some specific atmospheric conditions, such as great thermal instability, high humidity, and the convergence of warm, moist air at low levels with cooler, drier air aloft [25]. In general, scientists understand that tornadoes form when a "rotating thunderstorm updraft" and a "gyratory wind" meet at the same place [25]. But latest tornado research finds that tornadoes can also be produced by "non-rotating" thunderstorms [34]. However, the most violent and "long-lived" tornadoes are produced in a special kind of thunderstorm, supercell[2], where the wind rotation is important [34]. Moreover, some other phenomena often provide the conditions for tornado formation. For example, tropical cyclones, an oceanic phenomena, usually spawn tornadoes before they disappear on the land [25]. The physical process involved is that when the tropical cyclones reach the land, they start decaying, the winds at the top decay faster thus a "strong vertical wind shear"[3] is formed which is essential for tornado development [31].

But many other uncertain factors still exist about how a tornado is formed [9]. Also, the physical data in a tornado, which is necessary for tornado dynamics study, is difficult to be recorded [4]. This is because it is hard to put equipment on the right positions

[2] "An often dangerous convective storm that consists primarily of a single, quasi-steady rotating updraft, which persists for a period of time much longer than it takes an air parcel to rise from the base of the updraft to its summit. Most rotating updrafts are characterized by cyclonic vorticity. The supercell typically has a very organized internal structure that enables it to propagate continuously. It may exist for several hours and usually forms in an environment with strong vertical wind shear. Severe weather often accompanies supercells, which are capable of producing high winds, large hail, and strong, long-lived tornadoes" as stated in [16].

[3] The condition is produced by a change in wind velocity (speed and/or direction) with height [16].

since tornadoes are only minutes long and they move with "unpredictable" paths [4]. The film TWISTER was based on the work National Severe Storms Laboratory (NSSL) did in the middle of 1980's [4]. NSSL tried for several years to put the equipment, called TOTO (TOtable Tornado Observatory)[4], into the "coming" path of a tornado, but it was very difficult to succeed. Samaras and National Geogrhphic [39] tried the similiar approach to put the probes, metal disks that can measure a tornado's wind speed, direction, barometric pressure, humidity, and temperature through the embedded sensors, into the path of the tornado funnel. However, only five such instruments have been "deployed" successfully into the tornado paths after more than a decade of attempts.

In addition to the field quantitative observations of tornado chasing teams [4, 46], other tornado research methods have also been used in the understanding of the internal structure and dynamics of tornadoes, such as the development of radars, satellites and laboratory experiments[5], but the improvement in modeling and numerical simulations plays an active role and has made significant advances. The detailed discussion will be given in the next chapter.

[4]This device used "hardened" sensors that had been used to measure wind shear and severe downslope winds. It was designed to be mounted in a pickup truck and deployed in 30 s or less. Wind speed, wind direction, temperature, static pressure, and corona were recorded [4].

[5]The purpose of the laboratory experiments is to create flows that, to the extent that it is possible, are geometrically and dynamically similar to natural flows, in order that meaningful observations and physical measurements can be made [6]. The equipment is called Tornado Vortex Chamber (TVC) [6]. By changing the speed of fan, scientists can control the air swirling speed. The more interesting thing is, when smoke is fed into the swirling air, they can even see the air moving [26].

Chapter 3

Modeling and Numerical Simulation

The use of mathematical model and numerical simulation in this thesis is to model and simulate the tornado flow movement, and then provide physical data at some extent for graphics rendering in order to combine the physically-based modeling and graphics in this tornado simulation. The details of the physically modeling theory, boundary conditions applied and numerical implementation of this simulation are discussed in this chapter. In Section 3.1, the modeling theory of tornado simulation is introduced, as well as the models used by scientists and the model used in this thesis. Then the boundary conditions applied in this simulation are discussed in Section 3.2. Finally, Section 3.3 introduces the staggered grid, the implementation of boundary conditions and the steps to solve the model equations.

3.1 Model Overview

The modeling study on tornado research can generally be divided into two categories: thunderstorm-scale simulations and tornado-scale simulations [30]. As discussed in [23, 30], the former category is pioneered by Klemp and Wilhelmson who present a model originally for cloud simulation known as KW model, and then this model is applied in tornado simulation to numerically simulate the formation and dynamics of the thunderstorms that are responsible for tornado formation. On the other hand, tornado-scale simulation, pioneered by Rotunno [30, 33], assumes that a particular rotating environment caused by the "convergence" of rotating flow along boundaries leads to the tornado-like[1] vortices, and the models applied in this approach are intended to provide the details of the wind field in the tornado and an understanding of the dynamics that lead to the flow rotation structure. Because the domain boundary and initial conditions are easy to control in the tornado-scale simulation, many laboratory researchers [6, 24, 33, 41] produce miniature tornadoes inside the laboratory by simulating the rotating apparatus environment instead of simulating the thunderstorm formation. Also, many tornado scientists [12, 27, 30, 36, 38] apply the tornado-scale approach in their numerical simulations to generate tornado-like rotation and uplifting flow by simulating a rotating environment.

But the physical models and boundary conditions applied in their tornado-scale simulations vary due to different considerations on the process of tornado formation. For example, most tornadoes take the funnel shape in the real life, thus a tornado flow structure is generally believed to be axisymmetric [12, 27, 30, 36]. Such assumption leads to

[1]This term could apply to any vortex caused by the convergence of rotating fluid along boundaries [30].

the fact that most of the numerical models are in 2D Polar coordinate system as presented by Sinkevich *et al.* [36] and Fiedler [12]. They all assume the tornado funnel rotating about a vertical axis with a known constant angular frequency. The model of Sinkevich *et al.* [36] involves the two-phase (liquid and gas) turbulent heat and mass transfer while Fiedler [12] uses an incompressible Navier-Stokes equations model applied in a cylinder domain. It is an axisymmetric convection model with a defined thermodynamic speed limit (the rotation speed) in a closed domain [12]. Navier-Stokes equations are also applied in Trapp *et al.*'s [38] simulation, but the model is applied in 3D cube domain and they assume that the flow is compressible. Their goal is to study how tornadoes, which appear "at least locally axisymmetric", are born out of the asymmetric surrounding flow with initial horizontal velocity only [38]. As will be discussed in Section 3.2, their simulation is by simulating a tornado-producing environment, but they do not simulate a rotating environment initially as other tornado-scale simulations generally do. In this tornado simulation, the tornado-scale simulation approach is applied.

The physical model used in the tornado-scale simulation here is the governing equations of fluid mechanics – the Navier-Stokes equations [17]. This model was originally developed to simulate the viscous and unsteady fluid flow dynamics [3, 17, 40]. But recently, it has been widely used in computer graphics for the simulation of physical phenomena, such as water [14], smoke [11], cloud [18], fire [29] and the growth of plant leaf [40]. As described above, this model has also been applied in tornado simulations [12, 28, 30, 38]. In the modeling of this simulation, the equations employed are for incompressible fluid flow, which have also been considered in [12, 28, 30]. However, the axisymmetric tornado internal structure is not assumed here, thus this simulation allows the tornado to form

any shape. Moreover, this model is applied in 3D Cartesian coordinate system instead of the Polar system. The 3D cube model coordinate environment is depicted in Figure 3.1.

The standard Navier-Stokes equations applied in this simulation can be written in dimensionless form as:

$$\begin{aligned} u_t + (u \cdot \bigtriangledown)u &= \frac{1}{Re}\bigtriangleup u - \bigtriangledown p + g \\ \bigtriangledown \cdot u &= 0, \end{aligned}$$

where u is the fluid velocity vector, p is the pressure, g denotes body forces such as gravity, and Re is the Reynolds number which represents the viscosity of the fluids [17, 40]. In our mathematical model, the motion of the liquid is described by the evolution of two dynamic field variables, velocity and pressure. In three dimensions, the velocity vector is composed of the vertical component w along Z and 2 other components u and v on X-Y plane where u is along X and v is along Y. The 3D Navier-Stokes equations in component form can be given as:

momentum equations:

$$\begin{aligned}
\frac{\partial u}{\partial t}+\frac{\partial p}{\partial x} &= \frac{1}{Re}\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}\right)-\frac{\partial u^2}{\partial x}-\frac{\partial uv}{\partial y}-\frac{\partial uw}{\partial z}+g_x \\
\frac{\partial v}{\partial t}+\frac{\partial p}{\partial y} &= \frac{1}{Re}\left(\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}\right)-\frac{\partial uv}{\partial x}-\frac{\partial v^2}{\partial y}-\frac{\partial vw}{\partial z}+g_y \\
\frac{\partial w}{\partial t}+\frac{\partial p}{\partial z} &= \frac{1}{Re}\left(\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}\right)-\frac{\partial uw}{\partial x}-\frac{\partial vw}{\partial y}-\frac{\partial w^2}{\partial z}+g_z
\end{aligned}$$

continuity equation:

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}=0,$$

where

u: horizontal velocity of fluid flow (along the X coordinate). Positive direction is right.

v: horizontal velocity of fluid flow (along the Y coordinate). Positive direction is inward.

w: vertical velocity of fluid flow (along the Z coordinate). Positive direction is up.

g_x, g_y, g_z: gravity component on X, Y and Z coordinates respectively.

Note that since the axisymmetric flow structure is not assumed in modeling, this model can be applied to simulate tornadoes with different shapes, as shown in Section 5.2.

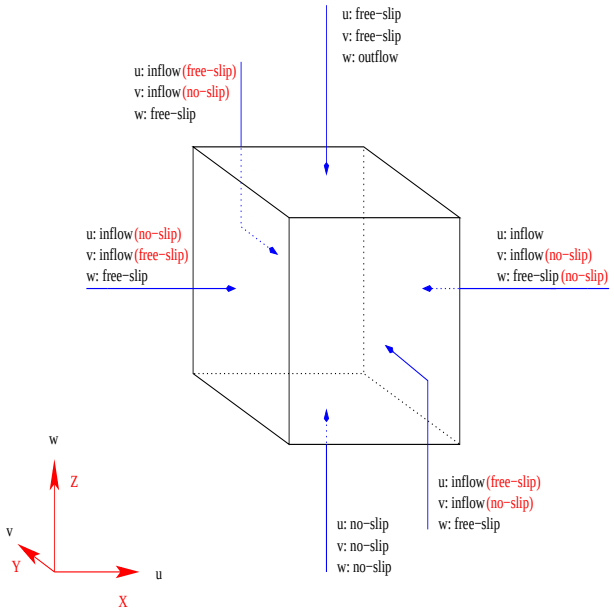


Figure 3.1: Coordinate environment and boundary conditions of the model applied in this tornado simulation. The boundary conditions inside the parentheses are used in Trapp *et al.*'s [38] simulation.

3.2 Boundary Conditions

Before the boundary conditions applied in this simulation are presented, a short introduction on the definitions of different boundary conditions is given first as discussed in [17]. The detailed explainations in mathematical form and how to implement these conditions in this simulation will be given in Section 3.3.

1. no-slip condition: the fluid is at rest there.

2. free-slip condition: no fluid penetrates the boundary. Contrary to the no-slip condition, however, there are no frictional losses on the boundary.

3. inflow condition: the value of the velocity component is given explicitly. The flow can go in or out of the boundary.

4. outflow condition: the velocity does not change in the direction normal to the boundary.

Note that as discussed in Section 3.3, the boundary conditions are applied to the individual velocity component, *i.e.*, velocity u, v and w at the same boundary position can have different boundary conditions. This technique is applied in tornado-scale simulations, such as in [38].

Tornado studies reveal that the boundary conditions are essential in tornado-scale simulation to achieve the rotation and uplifting movement [30, 33]. The general approach applied in tornado-scale simulation is to force the flow rotating along the sides of the domain boundaries and draw the flow out of the top domain with some boundary conditions, such as outflow condition [30].

As introduced in Section 3.1, Trapp *et al.* [38] present a tornado-scale simulation on how a symmetric tornado is born out of the asymmetric surrounding flow. The boundary conditions applied in their simulation are illustrated in Figure 3.1. In their simulation, a fixed vertical shear flow enters the cubic domain through the right boundary plane in Figure 3.1, the vorticity associated with this shear flow is horizontal and "transverse" to the inflow velocity [21, 34, 38]. To simulate the uplifting flow effect, they use outflow condition for w and free-slip for u and v on the top boundary. The extraction flow distribution and how fast it is depend on how outflow condition is applied on w. For example, to simulate the flow exhaust effect that extraction is more violent around the center of the top boundary, they set the values of w on the top based on the exponential function in which the magitude is smaller closed to the edge and bigger approaching the top boundary center.

The boundary condition studies in this thesis is based on Trapp *et al.*'s. In their simulation, the outflow condition on the top is reasonable to simulate the fact, as discussed by Rotunno [34], that the low pressure on the top of tornado causes the air "upward" which finally generates the "updraft". So in this simulation, the same boundary condition is applied on the top (See Figure 3.1). But the boundary conditions on the vertical planes are different from theirs. For instance, on the left boundary plane in Figure 3.1, the no-slip condition for u (along the X direction) used in Trapp *et al.*'s simulation assumes that there is no flow in and out of that plane, which in turn isolates tornado from the natural environment. In fact, it seems awkward to force the flow to go up or down on that plane. The problem is the same to v (along the Y direction) on the back and front boundary planes. In their discussion, they name these boundaries as "stagnation

walls" [38]. Although such boundary condition set up is necessary to their simulation, different boundary conditions are applied in the simulation here in order to simulate a more reasonable tornado formation environment. To do this, inflow condition is used for both u and v on the X-Y planes. It assumes that flow can go in or out of the vertical boundaries at a flow speed explicitly given. Moreover, this simulation is to simulate the rotating environment initially as a tornado-scale simulation generally does. Tornado research has shown that rotation can be generated by applying inflow condition on one boundary only [34, 38], but to obtain the rotation faster, inflow condition is applied on all 4 vertical boundaries (excluding top and bottom) in this thesis. In addition, u and v are assigned at specific directions along the vertical boundaries as shown in Figure 3.2. This rotation implementation is to simulate the counter-clockwise rotation as the tornadoes have in the Northern hemisphere. Note that one problem to this velocity direction assignment is that the direction of the combined velocity changes sharply at the corners. For example, in Figure 3.2, the combined velocity on the left boundary has -45° direction with respect to the positive direction of X axis. On the other hand, the direction is 45° on the bottom boundary. Such sharp direction change at the bottom left corner may cause flow eddy. So a better approach is to assign the flow velocities in continuous changing directions shown in Figure 3.3. But such approach would make the boundary condition implementation very complicated. So to simplify the implementation, the velocity directions are assigned based on Figure 3.2. And to be noted that the expected fluid eddy cannot be seen from the final tornado images presented in Section 5.2. One reason maybe that the flow eddy at the corner is too small compared with the flow in the middle of the domain.

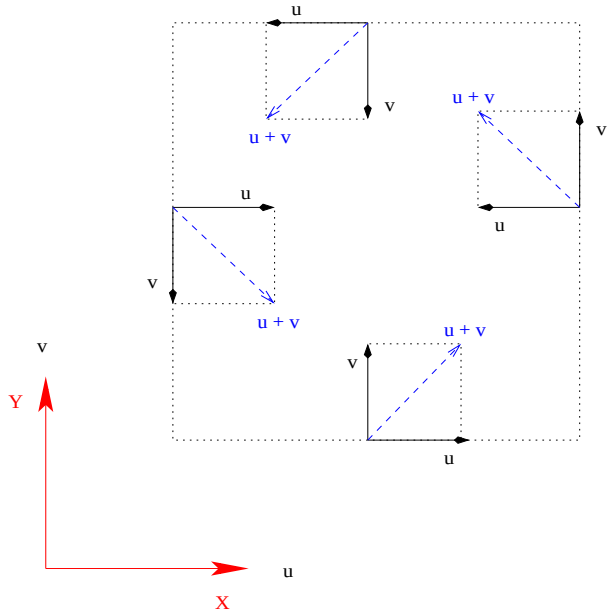


Figure 3.2: Velocity directions on X-Y plane to simulate the counter-clockwise rotation in the Northern hemisphere.

But how to appropriately implement such inflow conditions for u and v on those boundaries leads to more subtle questions. For example, what is the magnitude of u or v at a certain vertical level? And how does the magnitude change along the vertical direction from the bottom to the top, *e.g.*, increasing, decreasing or being a constant? Previous tornado studies [34, 38] show that applying a vertical shear flow with varied velocity magnitude on one of the vertical boundaries is a reasonable choice. For example, in Trapp *et al.*'s [38] simulation, a vertical shear flow, as presented in Figure 3.4(a), enters the domain through the right boundary. Also as pointed out by Rotunno [34] and Houze [21], such vertical shear flow generally leads to the horizontal rotation at the end. At the early step of this simulation, such vertical shear flow approach is applied on one boundary,

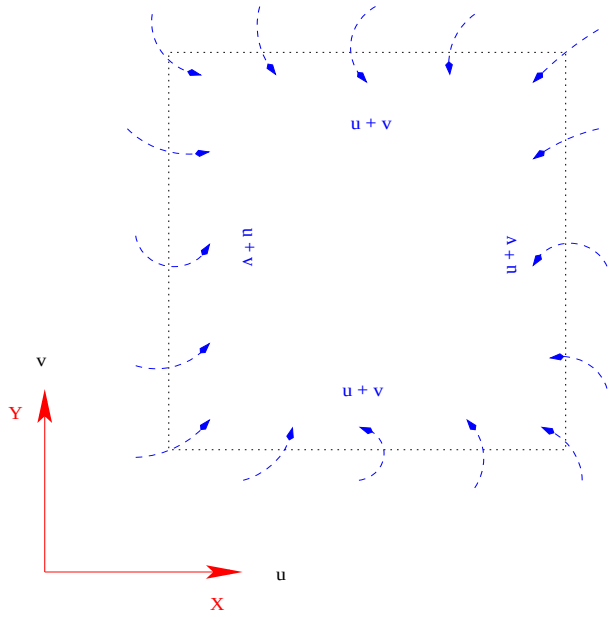


Figure 3.3: A better approach to assign velocity directions on X-Y plane for simulating the counter-clockwise rotation in the Northern hemisphere. The combined velocity direction changes continuously.

and with the combination of the outflow condition on the top boundary, the tornado-like rotation and uplifting flow effect are simulated. In addition to these simulation results, note that the tornado studies in this simulation also want to produce simulated tornadoes with interesting shapes. The current tornado numerical simulation results are only limited to the velocity, pressure and temperature fields without a simulated tornado image to show its movement, shape and color, thus making the simulation less interesting. But in this tornado simulation, research on how boundary conditions affect tornado shapes are conducted. To the best of my knowledge, research on how boundary condition leading to a final realistic tornado shape has never been found in tornado numerical simulation literature. Based on the specific boundary conditions applied on the vertical boundaries, the experiments conducted in this simulation show that only applying the vertical shear flow on one boundary can produce rotation, but it cannot generate a nice tornado shape at reasonable simulation time. On the other hand, as Figure 5.2(a) in Section 5.2 demonstrates, by applying shear flow on all 4 vertical boundaries with certain varied velocity magitudes can lead to interesting funnel shape within reasonable time. To simulate such funnel shape, on the 4 vertical boundaries, we can set the velocity magnitude of u and v decreasing from the bottom to the top. Note that this idea focuses on how to generate the expected tornado flow and shape. Since I have not found tornado research literatures about how boundary conditions affect the tornado shapes, unfortunately I cannot refer to tornado scientists' work to verify such magnitude assignment. In my simulation, one implementation of this magitude assignment is to set u and v equal to $2 \times \frac{nz-k}{nz}$, where nz is the number of segments in the numerical discretization along the Z direction, k is the index along the Z coordinate and it is from 0 to nz from the bottom to the top. As the

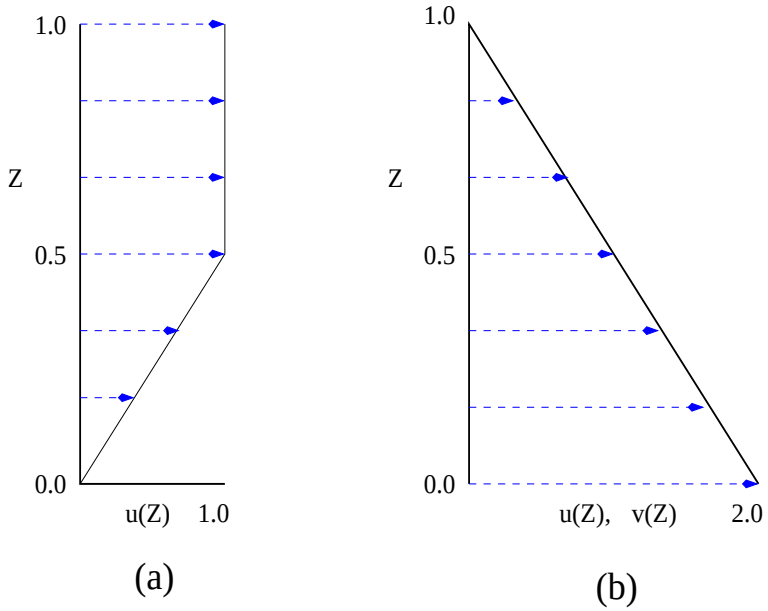


Figure 3.4: The vertical shear flow profile applied on the boundary. (a) Applied only on the right boundary in Trapp *et al.*'s simulation [38] to produce horizontal rotation. (b) Applied on all 4 vertical boundaries in this simulation to generate interesting tornado shape in addition to the horizontal rotation.

velocity profile demonstrated in Figure 3.4(b), for u and v near the bottom, k is small, thus leading to larger magitudes of u and v. When u and v are close to the top, k is bigger, so u and v are smaller on the top boundary.

Moreover, the exponential function used to control the velocity magitude distribution of w on the top is same as Trapp *et al.*'s [38]:

$$w_{i,j,nz} = 2\, e^{-\left[\left(\frac{\frac{i}{nx}-0.5}{r}\right)^2+\left(\frac{\frac{j}{ny}-0.5}{r}\right)^2\right]}, \tag{3.1}$$

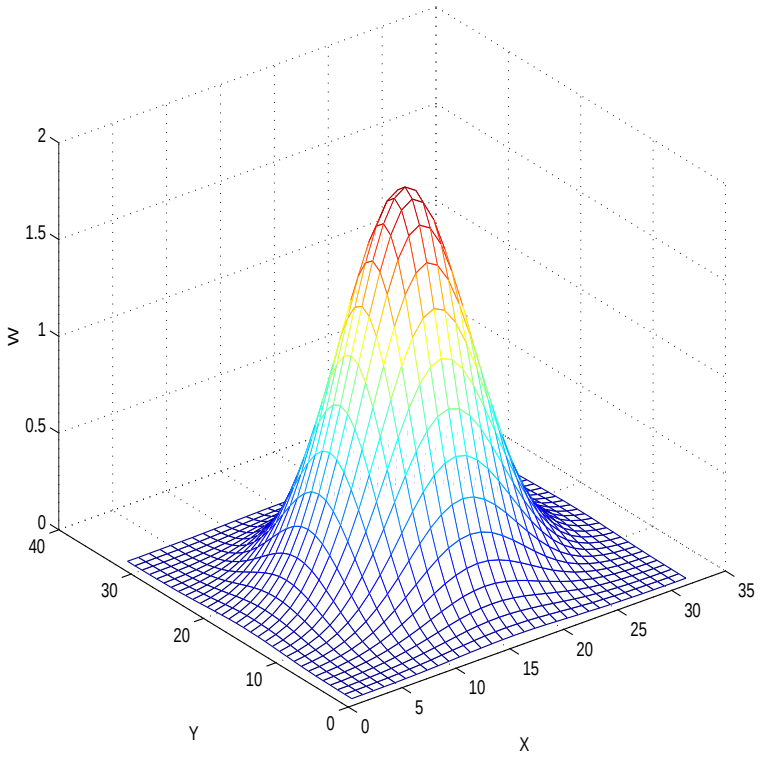


Figure 3.5: The magitude exponential distribution of w on the top boundary.

where nx, ny and nz denote the number of segments in the numerical discretization, i and j are the index along X and Y coordinates, and the value of r is a constant $\sqrt{0.05}$. The magitude distribution of w on the top boundary based on this extraction function is shown in Figure 3.5.

Finally, the boundary condition applied on the bottom boundary is no-slip condition. It is based on the assumption that the ground has no contribution to the environment rotation and upward exhaust flow movement. The complete set of boundary conditions applied in this tornado simulation is given in Figure 3.1. In addition, to help readers understand the similarities and differences between Trapp *et al.*'s model and the boundary

Table 3.1: The comparisons on modeling and boundary conditions between Trapp *et al.*'s simulation [38] and this simulation

Simulation	Model	Boundary Conditions			
		Top	Bottom	Vertical Boundaries	
				Conditions Applied	Velocity Magitude
Trapp *et al.*'s	Navier-Stokes equations	free-slip for u and v, outflow for w	no-slip for u, v and w	inflow for u on the right boundary, all others are free-slip	$u = \begin{cases} 1, & k \geq \frac{nz}{2} \\ 2 \times \frac{k}{nz}, & k < \frac{nz}{2}, \end{cases}$ formula applies to u on the right boundary only
The simulation here	same	same	same	inflow for u and v, free-slip for w	$u, v = 2 \times \frac{nz-k}{nz}$, formula applies to u and v on all 4 boundaries

conditions considered here, the comparisons are summarized in Table 3.1.

3.3 Numerical Solution

The $nx \times ny \times nz$ domain in Figure 3.1 is discretized as a staggered grid, in which the different unknown variables are not located at the same grid points [17]. For example, as demonstrated in Figure 3.6, for a particular cell in 3D, pressure p is located in the cell center while u, v and w are at the center of the right, back and top planes of that cell respectively, *i.e.*, the location of u, v and w all shift by half a cell length along the positive coordinate directions accordingly. This staggered arrangement of the unknowns prevents possible pressure oscillations which could occur if we evaluate all unknown values of u, v, w and p at the same grid points. But it should be noted that not all extremal grid points lie on the domain boundaries. For instance, in Figure 3.6(a), the vertical boundaries have

no v-values, just as the horizontal boundaries have no u-values. For this reason, an extra boundary strip of grid cells, as shown in Figure 3.7, is introduced, so that the boundary conditions may be applied by averaging the nearest grid points on either side.

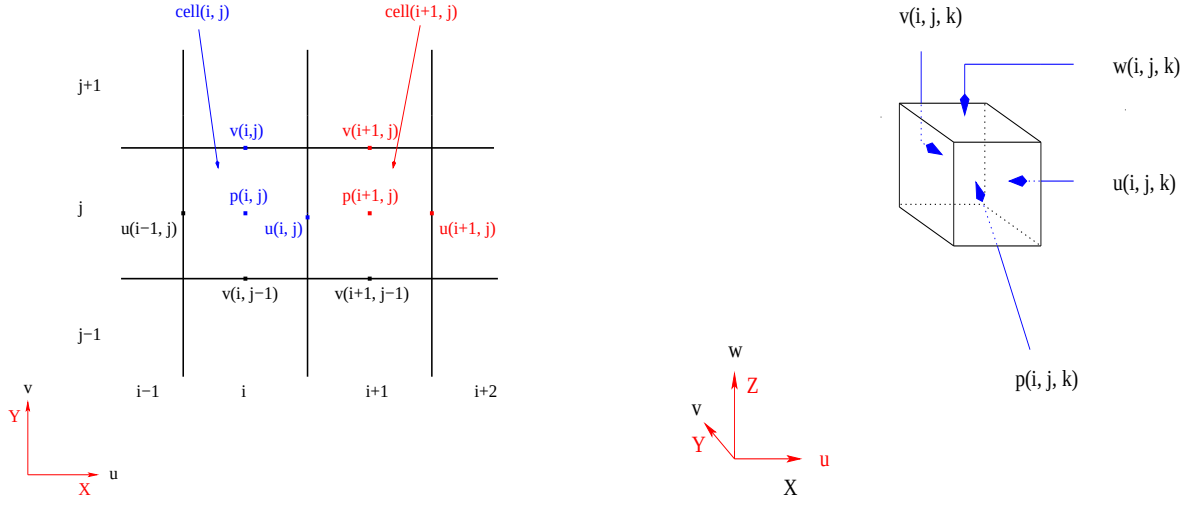


Figure 3.6: Staggered grid applied in 2D and 3D. (a) In 2D, velocities (u, v) and pressure (p) are in different locations on the plane. (Redrawn from Griebel *et al.* [17]). (b) In 3D, velocities (u, v, w) shift by half a cell length along the positive X, Y and Z coordinate directions and they locate at the cell boundary plane centers while pressure (p) is at the cell center.

The numerical treatment on boundary with different boundary condtions is described as follows [17]. To simplify the discussion, the explanation is given in 2D based on the diagrams presented in Figure 3.6(a) and Figure 3.7.

In order to formulate the boundary conditions, we first define the following notations as introduced in [17]:

φ_n: the component of velocity orthogonal to the boundary (in the exterior normal

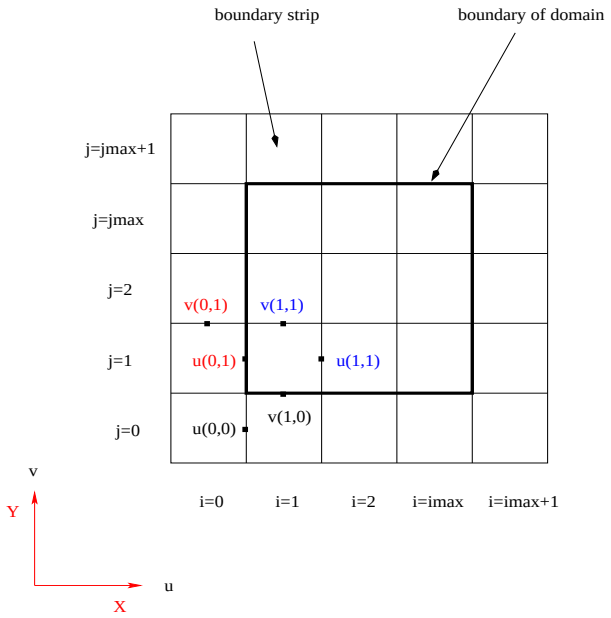


Figure 3.7: Staggered arrangement of the unknowns on a 2D domain after an extra boundary strip of grid cells is introduced (Redrawn from [17]).

direction).

φ_t: the component of velocity parallel to the boundary (in the tangential direction).

$\frac{\partial \varphi_n}{\partial n}$: the first derivative of φ_n (in the normal direction).

$\frac{\partial \varphi_t}{\partial n}$: the first derivative of φ_t (in the normal direction).

Then, along the vertical boundary segments, *i.e.*, along the Y direction in Figure 3.6(a) and Figure 3.7, we have

$$\varphi_n = u, \quad \varphi_t = v, \quad \frac{\partial \varphi_n}{\partial n} = \frac{\partial u}{\partial x}, \quad \frac{\partial \varphi_t}{\partial n} = \frac{\partial v}{\partial x} \tag{3.2}$$

and, along the horizontal segments, *i.e.*, along the X direction in Figure 3.6(a) and Figure 3.7,

$$\varphi_n = v, \quad \varphi_t = u, \quad \frac{\partial \varphi_n}{\partial n} = \frac{\partial v}{\partial y}, \quad \frac{\partial \varphi_t}{\partial n} = \frac{\partial u}{\partial y} \tag{3.3}$$

So for cells on the boundary, different boundary conditions can be implemented as follows [17]:

1. no-slip condition, *i.e.*,

$$\varphi_n(x, y) = 0, \quad \varphi_t(x, y) = 0$$

by

$$\begin{aligned} u_{0,j} = 0, \quad u_{i_{max},j} = 0, \quad j = 1, ..., j_{max} \\ v_{i,0} = 0, \quad u_{i,j_{max}} = 0, \quad i = 1, ..., i_{max} \\ v_{0,j} = -v_{1,j}, \quad v_{i_{max}+1,j} = -v_{i_{max},j}, \quad j = 1, ..., j_{max} \\ u_{i,0} = -u_{i,1}, \quad u_{i,j_{max}+1} = -u_{i,j_{max}}, \quad i = 1, ..., i_{max} \end{aligned}$$

   In this simulation, this condition is applied to u, v and w on the bottom boundary as shown in Figure 3.1.

2. free-slip condition, *i.e.*,

$$\varphi_n(x, y) = 0, \quad \frac{\partial \varphi_t(x, y)}{\partial n} = 0$$

by

$$u_{0,j} = 0, \quad u_{i_{max},j} = 0, \quad j = 1, ..., j_{max}$$

$$v_{i,0} = 0, \quad u_{i,j_{max}} = 0, \quad i = 1, ..., i_{max}$$

$$v_{0,j} = v_{1,j}, \quad v_{i_{max}+1,j} = v_{i_{max},j}, \quad j = 1, ..., j_{max}$$

$$u_{i,0} = u_{i,1}, \quad u_{i,j_{max}+1} = u_{i,j_{max}}, \quad i = 1, ..., i_{max}$$

In this simulation, this condition is applied to w on the 4 vertical boundary planes, as well as u and v on the top boundary.

3. inflow condition, *i.e.*,

$$\varphi_n(x, y) = {\varphi_n}^0, \quad \varphi_t(x, y) = (\varphi_t)^0, \quad {\varphi_n}^0, \quad (\varphi_t)^0 \quad given$$

As the definition of inflow condition is that the velocities are explicitly given on the inflow boundary, for the velocities normal to the boundary (*e.g.*, u on the left boundary), we can set the values on the boundary; for the velocity components tangential to the boundary (*e.g.*, v on the left boundary), we assign the values by averaging the values on either side of the boundary. In the simulation here, this condition is applied to u and v on the 4 vertical boundary planes, and the values assigned on the boundary are based on the shear flow profile shown in Figure 3.4(b).

4. outflow condition, *i.e.*,

$$\frac{\partial \varphi_n(x,y)}{\partial n} = 0, \quad \frac{\partial \varphi_t(x,y)}{\partial n} = 0$$

by

$$u_{0,j} = u_{1,j}, \quad u_{i_{max},j} = u_{i_{max}-1,j}, \quad j = 1, ..., j_{max}$$

$$v_{0,j} = v_{1,j}, \quad v_{i_{max}+1,j} = u_{i_{max},j}, \quad j = 1, ..., j_{max}$$

$$u_{i,0} = u_{i,1}, \quad u_{i,j_{max}+1} = u_{i,j_{max}}, \quad i = 1, ..., i_{max}$$

$$v_{i,0} = v_{i,1}, \quad v_{i,j_{max}} = v_{i,j_{max}-1}, \quad i = 1, ..., i_{max}$$

In the simulation, this condition is only applied to w on the top boundary. As the staggered arrangement shown in Figure 3.6(b), w lies on the top boundary. So for a particular boundary cell on the top, its w value is first assigned based on the velocity magitude function (3.1) presented in Section 3.1, then its neighbor inside the domain is set having the same w value.

Finally, the equations are solved by using finite difference method. Interested readers can find detailed explanations in Chapter 3 of [17]. The outline of the equation solving algorithm is that Conjugate-gradient method is used to solve the Possion equation for pressure p at time t given the velocities u, v and w at time $t-1$. Then, the new u, v and w at time t are updated based on p obtained earlier.

Chapter 4

Rendering Algorithm

How to visualize the simulated tornado flow is discussed in this chapter. In Section 4.1, the particle system theory and the system used in this simulation are introduced first. The purpose of using a particle system in this thesis is to define the tornado volume which is difficult to be represented by using the classical geometric elements, such as polygons. Then in Section 4.2, the particle density calculation and density smoothing techniques applied in this simulation are discussed, with the volume rendering algorithm being introduced at the end.

4.1 Particle System

4.1.1 Overview

In computer graphics, the generation of objects or structures procedurally[1] is an important area, and functionally based modeling[2] methods are generally applied in 3D computer graphics to achieve this goal. Among them, three most important techniques applied are discussed in [42]:

- non-deterministic or stochastic functions. They are used to represent a particular subject with randomness. In computer graphics, terrain, fire and turbulence have been generated by using such random functions.
- deterministic functions. Randomness is not involved, so such functions can produce the same output given the fixed initial conditions.
- the functions which involve both randomness and non-randomness, *i.e.*, the combination of the above two.

In 3D computer graphics, these techniques are widely used for modeling purpose to produce objects. For example, in natural phenomena simulation, they have been used to generate particles to model fuzzy objects, such as clouds and smoke [32]. These objects do not have smooth and "well-defined" shapes and are dynamic, so it is difficult to represent them using the classical "surface-based" elements such as polygons or patches. In

[1] Graphical image is generated from programming code but not picture files [42].

[2] To generate objects or structures procedurally, the computer program code is composed of several functions to produce the rendered image step by step. This is called functionally based modeling in computer graphics [42].

these cases, particle systems are applied to model such fuzzy ojects. A particle system models these objects as "a cloud of primitive particles that define their volume", and particles can move over time. Therefore, a particle system is able to represent the object movement and shape changes, thus images with irregular or "ill-defined" object shapes can be generated without applying a complicated geometry system.

In a particle system, depending on the object being modeled, a particle may have varied attributes. But in general, a particle has the following characteristics [15, 32, 42]: position, mass, velocity (both speed and direction), size, color, transparency, shape and lifetime. Over the time, new particles are generated into the system and their attributes are initialized, then these attributes are modified at later time steps to represent the shape and dynamic movement of the modeled object. Finally, when particles reach their lifetime they die and are out of the system.

The general framework of a particle system can be decribed as [15, 32]:

1. new particles are generated into the system,
2. each new particle is assigned initial attribute values,
3. all particles that reach their lifetime are out of the system,
4. all remaining particles update their attributes at each time step, and
5. an image of the existing particles is rendered.

As discussed in [32], such system can be implemented using a computer language to execute several functions. In computer graphics, simple stochastic (non-deterministic)

processes are generally used to generate particles. To control the shape, appearance and dynamics of particles, the model designer needs to assign a set of parameters to control the range of randomness. On the other hand, because such modeling process is procedural, the modeling steps can also incorporate other computational model solutions which describe the dynamics of the object. For example, the solution of a partial differential equations could be tied to a particle system to provide the particle attribute information. This is the approach applied in this tornado simulation which will be discussed next.

Of course, a static object can be also represented using the modeling process of a particle system, but such modeling process is more interesting when it is used to model the "time-varying" phenomena for animation purpose [42]. This technique has been used in computer graphics to simulate natural phenomena. For example, in addition to clouds and smoke introduced earlier, fire and explosions have also been modeled using particle systems, as presented in the film `Star Trek II: The Wrath of Khan` in June, 1982 and `Return of the Jedi` in May, 1983 [32]. Moreover, aurora [1] has been simulated by using a particle-mesh method and grass [32] has been simulated as well by using a partially stochastic technique. One important aspect of particle systems is that particles can move. Thus many other natural phenomena can also be simulated by using particle systems to represent their dynamics, such as the tornado simulation here.

4.1.2 The Particle System in this Simulation

The shape diversity and flow dynamics of tornadoes make their geometry description difficult, thus it is hard to define their volume using the classical geometric elements,

but a particle system makes it possible to model their volume and dynamics over time. Moreover, it is believed that particle density directly relates to transparency, so a coupled particle system is applicable to the realistic visual result. For example, in the film TWISTER [10] released in 1996, particle systems are used to simulate the tornado movement and shape changes. In their particle systems, a particle has position, lifetime, velocity, acceleration, color and transparency attributes. Randomness based functions are used to update such attributes. So their simulation is for pure visual purpose without taking into account the physical characteristics in the tornado evolution process, but in this simulation, the partical attributes are updated based on the solutions of physical model equations which is physically-based at some extent.

In the particle system of this simulation, a particle has position, lifetime, velocity and mass attributes. The initial position of a particle depends on the purpose of such particle. For example, at the beginning of the simulation, certain number of particles are added into the system at random positions to describe how the cube shape converges to a particular tornado shape at the end. But some other particles are added at later steps at specific locations to simulate some other effects, such as color variation and bottom debris swirling effect. The lifetime attribute is defined by the position instead of time as a particle system usually does. For example, during the simulation, a particle dies if it is out of the system boundaries, otherwise, its position is updated based on its velocity (speed and direction). As presented earlier, this particle system is coupled with the model's numerical solutions. So at each time step, the position attribute is updated based on the velocity solutions instead of using random process as generally applied in computer graphics. Therefore, to simulate a particular tornado shape, the particle positions can

be controlled accordingly by applying appropriate boundary conditions. One example is given here to show how this process works in this simulation, such as how the attributes are initialized and how they are updated. In Figure 5.2(a) of Chapter 5, a simulated funnel shape tornado is presented. In order to produce such particular shape, the boundary conditions and velocity magitude control functions decribed in Section 3.2 are applied to compute velocity solutions. Then a particle system is integrated with the equation solver. To simulate the tornado evolving process, 500000 particles are randomly generated at the beginning to fill the whole domain. After that, at each time step, 1200 new particles are generated into the system only at the bottom[3] in order to fill the *hole*, which is caused by the upward particles generated at earlier steps. So depending on the purpose, particles have different initial positions. Then all the living particles move based on their current positions and the velocity information computed from the numerical solutions. Thus after rendering, an interesting tornado shape is produced, and finally the wind flow movement, such as rotation and going up, is observed in the animation.

Moreover, to simulate the debris swirling effect at tornado bottom, some special particles are generated at each time step. In the simulation presented in Figure 5.4(b), 40 special particles are generated into the system at each step. All these particles have mass attribute. Note that the particles discussed above do not have mass attributes due to the assumption that only the water vapor, which does not have mass, is in the wind. But this assumption cannot be applied to simulate the debris swirling effect at bottom because clearly debris has mass. In the simulation, the mass value is randomly initialized when the particle enters the system. Then, by applying gravity and centrifugal force, the

[3]On the plane of $z = 0.05$ where $0.25 < x < 0.75$ and $0.25 < y < 0.75$. The domain is $0 < x, y, z < 1$.

velocity attribute of these particles is modified differently from the numerical solutions. The details are discussed next.

At time step n, for a particular particle for debris swirling effect simulation, it has the following attribute denotions:

1. position: (x^n, y^n),
2. velocity: (u^n, v^n), and
3. mass: m

Moreover, as the simulated rotation is expected to be counter-clockwise, the velocity of this particle is illustrated in component form accordingly in Figure 4.1(a). We also know that, by Newton's first and third law of motion, the magitude of centrifugal force of this particle can be computed by

$$F = m\,\frac{v^2}{r},$$

where v is this particle's velocity, r is the rotation radius, *i.e.*, the distance from point (x_c, y_c) to (x^n, y^n) in Figure 4.1. In this simulation, this formula is rewritten as

$$F^n = m\,\frac{(u^n)^2 + (v^n)^2}{\sqrt{(x^n - x_c)^2 + (y^n - y_c)^2}} \tag{4.1}$$

The angle of β shown in Figure 4.1 can be calculated based on the formula $\arctan(\frac{|v^n|}{|u^n|})$. Thus, the X and Y components of F^n can be given as $F_x^n = F^n \cdot \sin(\beta)$ and $F_y^n = F^n \cdot \cos(\beta)$, respectively. Their geometry relationship is shown in Figure 4.1(b).

Because the acceleration $a^n = \frac{F^n}{m}$, we have the value of its X component $a_x^n = \frac{F_x^n}{m}$

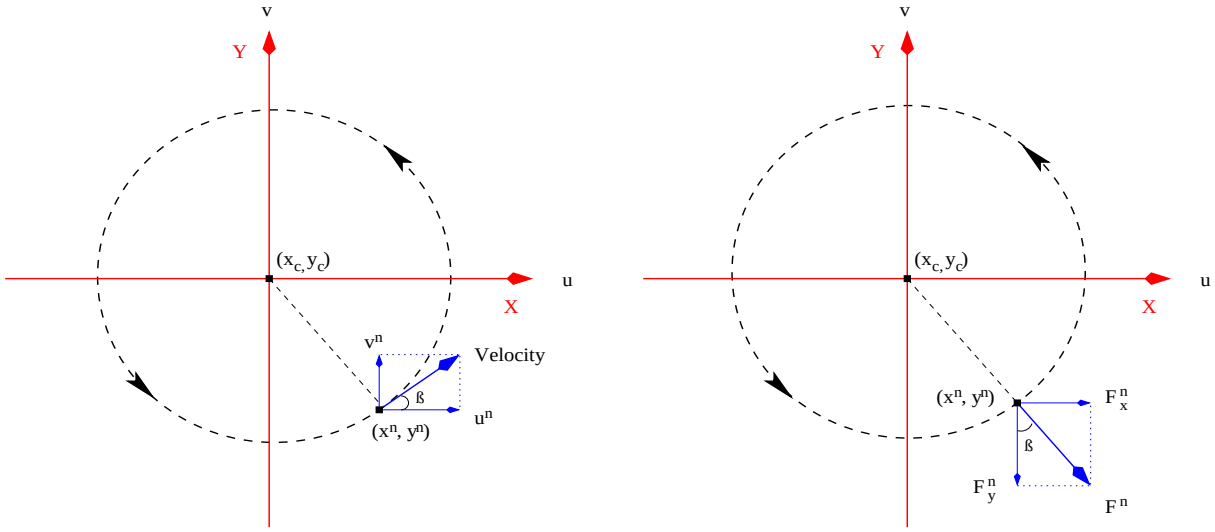


Figure 4.1: The decomposition of velocity and centrifugal force for a particular particle in the bottom debris swirling simulation. (a) Velocity decomposition. (b) Centrifugal force decomposition.

and its Y component $a_y^n = \frac{F_y^n}{m}$. Then the updated velocity components of this particular particle at position (x_n, y_n) can be written as

$$u^{new} = u^n + \triangle t\ \tilde{a}_x^n \tag{4.2}$$

$$v^{new} = v^n + \triangle t\ \tilde{a}_y^n, \tag{4.3}$$

where $\tilde{a}_x^n = a_x^n\ \nu$, $\tilde{a}_y^n = a_y^n\ \nu$ and ν is a control parameter. The reason to introduce such control parameter is because that m appears in F^n, F_x^n and F_y^n but is cancelled in a^n, a_x^n and a_y^n, so this parameter is used to simulate the acceleration effect which originates from the mass attribute. Also, since F^n is positive in Equation (4.1), and $0° \leq \beta \leq 90°$ because $\frac{|v^n|}{|u^n|}$ is always positive, so F_x^n and F_y^n are all positive in the above calculations. Therefore, a_x^n and a_y^n are positive as well in Equation (4.2) and (4.3). But in fact, a_x^n and a_y^n can be negative during the circulation motion. For example, in Figure 4.1, v^n and F_y^n are in opposite directions, thus a_y^n should be negative while a_x^n is positive since u^n and F_x^n have the same direction.

Therefore, for the particle shown in Figure 4.1, the formula to compute the new value of v in Equation (4.3) needs to be changed to[4]

$$v^{new} = v^n - \triangle t\ \tilde{a}_y^n\ \nu$$

As a result, u^{new} is bigger than u^n but v^{new} is smaller than v^n. So from Figure 4.1(a)

[4]Note that such changes are different in other quadrants.

we can see that this particle will move faster on the positive X direction but move slower on the positive Y direction. Finally, as a consequence of applying the centrifugal force, this particle will move away from the circular path. Of course, how far away the particle moves depends on the value of ν, and this value varies for different simulation goals. For example, ν is randomly selected between 0.3 and 0.9 in the simulation presented in Figure 5.4(b). Moreover, on the Z coordinate, w^n is updated after gravity is applied.

After the centrifugal force and gravity are applied to all the particles specially for the debris swirling simulation at the bottom, these particles rotate and go up slower, and move farther away from the wind body. The simulation result, as demonstrated in Figure 5.4(b) of Section 5.2, is consistent with the observed real tornadoes.

4.2 Visualization

4.2.1 Overview

The rendering algorithm applied in this thesis is not the general particle rendering approach in computer graphics. In computer graphics, the general rendering algorithm applied in a particle system is to render the particles directly, *i.e.*, based on the particle attributes, especially position, color and transparency, each particle contribute its color to the pixel it covers [32]. For example, in the film `TWISTER` [10], a particle rendering algorithm is used to visualize particles from multiple light sources. Based on the transparency attribute of a particular particle, the algorithm can calculate the extent that such particle can hide other particles that are behind it, and then the color contribution

of this particle to the pixel it covers can be determined from such calculation [32, 42]. As we noticed in the film `TWISTER` [10], this rendering algorithm leads to realistic tornado movement simulations.

But their rendering process cannot be applied in this simulation due to the fact that the particles in this particle system do not have color and transparency attributes. Although such attributes can be generated based on some random functions, certain amount of work to search parameters by trial has to be done in order to create the realistic visual result [7]. Note that how to control particle generation is already very difficult, so adding more randomness to particle's attributes would be even harder to produce good rendering result. Moreover, doing this involves experience and does not seem to get acceptable results at reasonable time. So to solve this, another rendering approach has to be researched in order to render the image using only the attributes what the particle system in this simulation already has.

The rendering algorithm applied in this thesis is volume rendering. Volume rendering, which means rendering the voxel-based data (in 3D) [42], provides an alternative solution to this problem. This rendering algorithm is most widely applied in medical imaging, but it has also been used in natural phenomena simulation [7]. The basic idea of it is that a viewer look at the volume in a projection view, then after mapping density of each voxel in the dataset to colour and transparency, a color and an opacity to each voxel are assigned. Finally these values are accumulated along the viewing direction and the final color of the covered pixels on the image plane is computed [42].

In volume rendering, since density is only related to position, the density of each voxel can be calculated easily from the 3D data. Therefore, the most important aspect of volume

rendering is how to specify the transfer functions that map density within the volume to colour and transparency [42]. In this simulation, the color to be used is selected by the user, and the density is only related to transparency. The metaball scheme, proposed by Wyvill *et al.* [45] and applied in Dobashi *et al.*'s [7] cloud simulation, is used here with modification to smooth the density distribution for better transparency results.

4.2.2 Density Calculation and Rendering

In this tornado simulation, the cube domain in Figure 3.1 is discretized into cells, and based on the position attribute of particles, each particle is mapped to a particular cell. Then, a modified metaball method is used to calculate and smooth the particle density for each cell. After texture mapping and antialising are applied, each cell color is rendered in OpenGL. The volume rendering algorithm is based on what Dobashi *et al.* apply in their cloud simulation but the algorithm applied here has the different density calculation procedures and visualization properties specifially for tornado simulation.

Metaball Scheme

The metaball scheme applied in this simulation is discussed next. First, the positions of all particles are mapped to a coordinate cube domain for visualization, here $250 \times 250 \times 250$ is used. Then the density for a particular cell in this cube is calculated using the metaball functions proposed by Wyvill *et al.* [7, 45]. In this simulation, the metaball is not a *ball* but a cube. For example, the metaball with radius R for a particular cell C is a cube which is in $2R + 1$ length with cell C in the middle. As a result, the density calculation

function for cell C at time t is given as [7, 45]:

$$
\begin{aligned}
D_{C,t} &= \sum_{i,j,k \in \Omega(C,R)}^{N} K(i,j,k,t) f(r) \\
f(r) &= \begin{cases} -\frac{4}{9}b^6 + \frac{17}{9}b^4 - \frac{22}{9}b^2 + 1, & r \leq R \\ 0, & r > R, \end{cases}
\end{aligned}
$$

where R is the effective metaball radius, $\Omega(C, R)$ is the set of cells whose centers are included in cell C's metaball, $K(i, j, k, t)$ is the number of particles of a particular cell in $\Omega(C, R)$ at time t, r is the distance of a particular cell to cell C, b is defined as $\frac{r}{R}$, and f is the density contribution function which has Gaussian-like shape. The basic idea of this metaball scheme is that, to calculate the density value of a particular cell, say cell C, an influence domain can be specified by choosing a metaball radius. Then the density of cell C is given by the sum of weighted number of particles in the cells surrounding cell C. This metaball method has two apparent advantages. First, the density distribution is smoothed to be continuous after this process. As discussed earlier, the opacity attributes of each cell depend on the density value, so a smoothed density distribution would lead to a smoothed color distribution as well in the rendering step. Second, the influence domain in metaball method is limited by the metaball size, thus the domain is finite instead of infinite as a Gaussian smoothing has. So the metaball size is closely related to the smoothing result and the computing speed. Thus, to obtain a nice smoothing result with satisfying the computing speed requirement at the same time, an effective metaball size has to be chosen appropriately. The experiments show that metaball size equals to 3 is a good choice. In

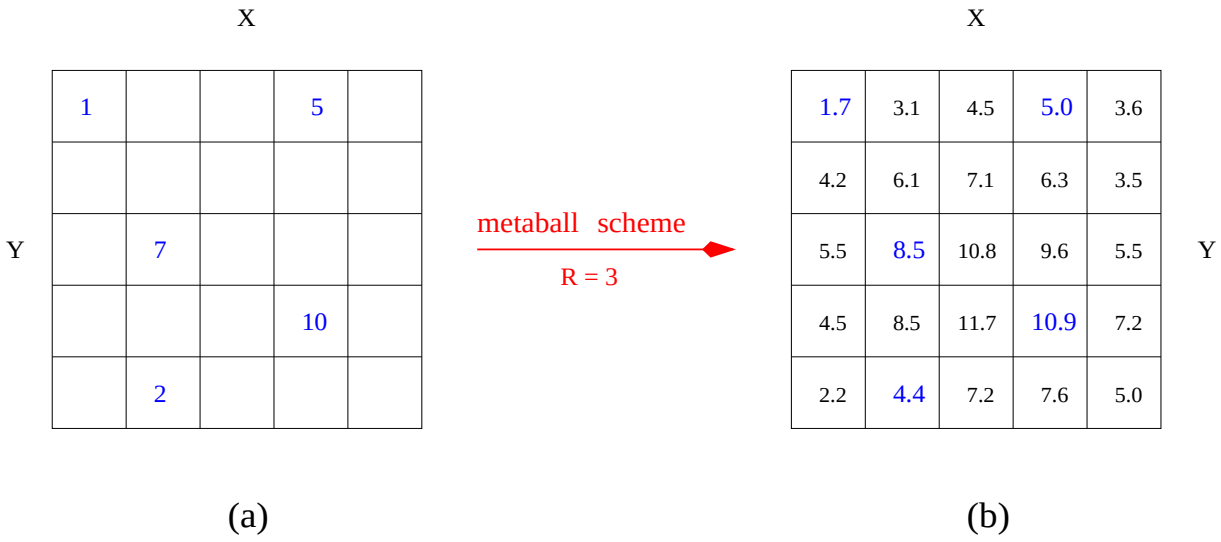
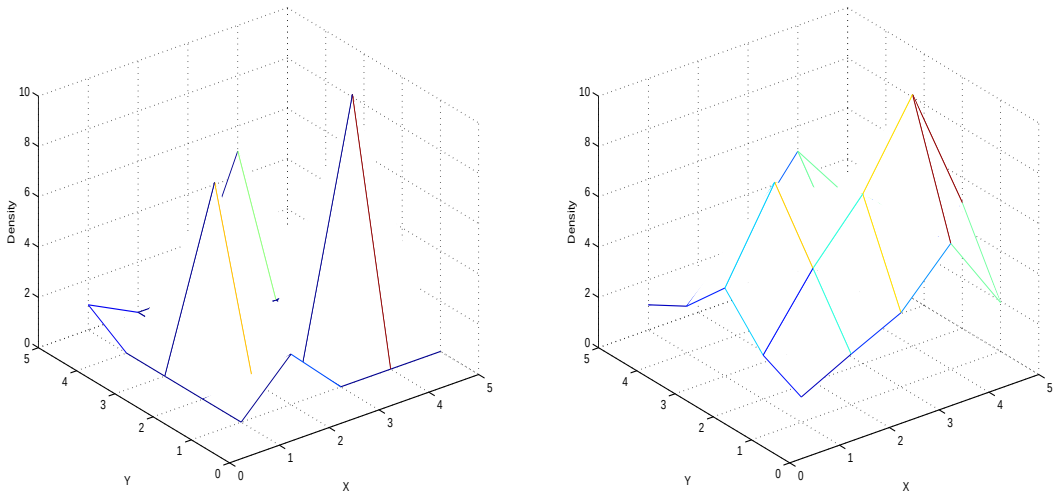


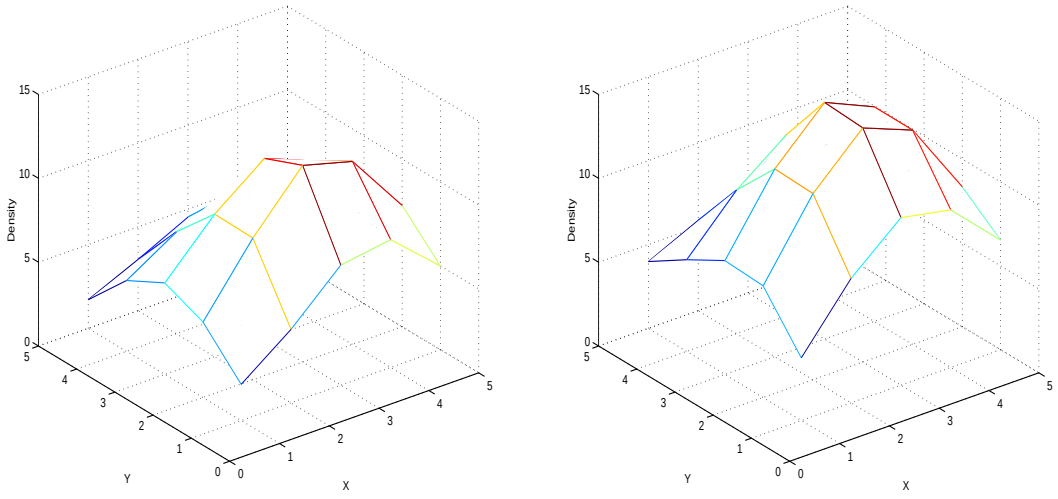
Figure 4.2: A diagram to show how a 2D dataset is modified after a metaball scheme with radius 3 is applied. The density on each grid point is computed based on the density calculation function applied in this tornado simulation.

Figure 4.2, an example is presented to show how the original dataset is modified after a metaball scheme with radius 3 is applied. Note that the smoothing process is shown in a simplified 2D version. In addition, Figure 4.3 illustrates the smoothing improvement effect on the 2D dataset demonstrated in Figure 4.2(a) after different metaball sizes are applied. Figure 4.3(a) is the mesh plot of the original data while (b), (c) and (d) are the plots after metaball smoothing with metaball size 1, 2 and 3 respectively. As demonstrated in the plots, as the metaball size increases, the density distribution is more continuous, thus the smoothing effect is improved.

After the density of each cell is calculated, we need to relate it to a certain opacity for the rendering step. However the densities after the metaball calculations have very broad distribution. This makes it difficult to ensure opacity consistency of a particular



(a) (b)



(c) (d)

Figure 4.3: The mesh diagrams of an original dataset and the smoothed dataset after metaball schemes are applied: (a) is the density distribution of the data presented in Figure 4.2(a) while (b), (c) and (d) are the density plots after metaball smoothing with metaball size 1, 2 and 3, respectively. The density on each grid point is computed based on the density calculation function applied in this tornado simulation.

density among different time steps. For example, because it is difficult to control the maximum number of particles in a cell in the dynamic simulation, if we simply define the maximum density to be 0 transparency and minimum density to be 100% transparency, a particular density value may have varied opacity values at different time steps because the maximum cell density in the particle system may vary at each step. To solve this inconsistency problem, the upper-bound and lower-bound density values are chosen to determine whether a particular density is high or low. More precisely, the density values around the upper-bound value are defined as high density values while the values close to the lower-bound are treated as low density ones. Then another array, say A, is used to discretize the density into certain levels. In the simulation, 0 and 200 are chosen to be the lower-bound and upper-bound density values and 200 levels' discretization is applied. If a density value is equal to or bigger than 200, *i.e.*, there are 200 or more particles in that cell after metaball scheme is applied, it is treated as high density, and value 1.0 is assigned to the same position in the new array A. On the other hand, if a density value is 0, it means that there are no particles inside a cell's metaball, its new value in A is set to be 0.0. For all other densities between 0 and 200, linear interpolation is used to map to their new values.

Volume Rendering

The rendering step is implementated in OpenGL based on Dobashi *et al.*'s [7] algorithm for cloud simulation. The algorithm, as shown in Figure 4.4, is to generate the image viewed from the viewer's position. The details are discussed below.

The rendering algorithm is based on the standard forward mapping (or splatting)

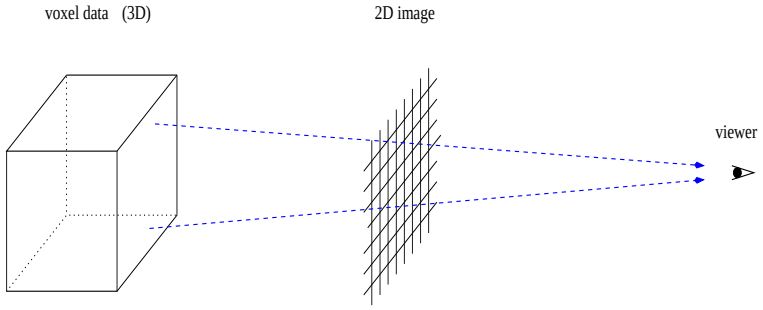


Figure 4.4: The volume rendering algorithm used in Dobashi *et al.*'s cloud simulation [7] is applied in this tornado simulation. Viewed from the viewer's position, the image is generated based on the forward mapping (or splatting) method.

method [1, 7, 43]. As shown in Figure 4.4, the camera is placed at the viewer's position and the perspective view is applied. Then, all the cells are projected onto the image plane. The calculation order is based on the distance of a cell to the viewer from the back to the front. So the cells farthest to the viewer are drawn and blended with the background first, and the closest cells to the viewer are at the end. The color used for each cell is selected by the user and cell's density is used to control the extent of opacity.

The above rendering algorithm is implemented in OpenGL. The biggest advantage of using OpenGL to implement this algorithm is that the blending processes in the algorithm can be done by the blending functions in OpenGL. Another note is that, as introduced earlier, the volume rendering algorithm applied in this tornado simulation is based on what Dobashi *et al.* [7] apply in the cloud simulation, but the blending factors applied in their simulation cannot be simply reapplied in the simulation here due to the fact that blending percentage of the next rendered pixel with the value already rendered depends on the visual representation of the simulated phenomena. The blending factors of $(GL\_SRC\_ALPHA, 1 - GL\_SRC\_ALPHA)$ are used here to obtain the realistic visual

result in tornado simulation, but Dobashi *et al.* use $(1, GL\_SRC\_ALPHA)$ in their simulation. Furthermore, texture mapping is applied to create the simulation background, and antialising is used to reduce the jagged pixel borders between pixels.

The rendering algorithm can be summarized as the following pseudo-code. We define cell's transparency as the values stored in the array A obtained after the metaball scheme is applied. Here T is used to denote that transparency of a particular cell, and the color used for drawing cells is denoted by (SR, SG, SB).

- The rendering algorithm:

  Set perspective view from the viewer position.

  Set blending factors as $glBlendFunc(GL\_SRC\_ALPHA, 1 - GL\_SRC\_ALPHA)$.

  Draw cells in descending distance order from the viewer with color (SR, SG, SB, T).

Note that the values used to represent colors in the pseudo-code are the values of R (red), G (green), B (blue) and *Alpha* in OpenGL respectively. The RGB values are used to respresent colors as the computer graphics generally does, but the value *Alpha* is special in OpenGL which reflects the transparency and is given based on the density values.

Chapter 5

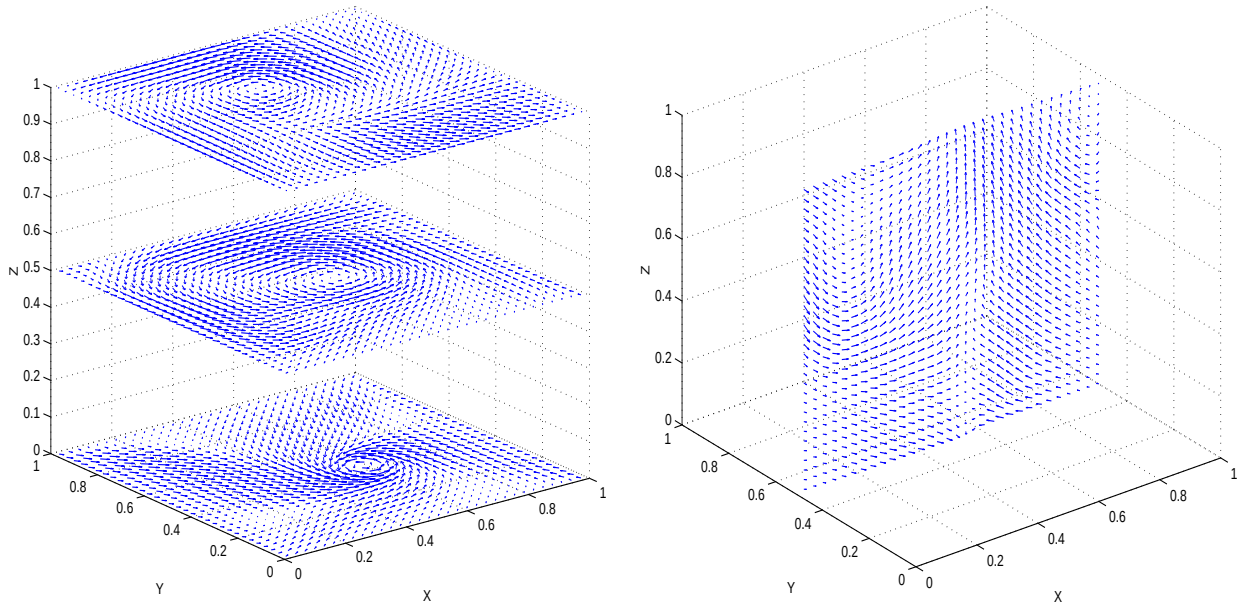
Validation and Visualization Results

The validation of this tornado flow simulation and visualization is first discussed in Section 5.1. I will qualitatively compare the simulated tornado flow in this thesis with a tornado scientist's simulation result. Then in Section 5.2, I will summarize all the results in this simulation and compare them with the real tornadoes. The simulated features presented are tornadoes with different shapes and colors. The bottom swirling effect and dynamic shape change effect of real tornadoes are also simulated in this thesis. Moreover, the parameters used in this thesis are summarized in Table 5.1.

5.1 Validation

As introduced earlier in Section 3.1, Trapp *et al.* [38] apply the tornado-scale simulation to research on how a tornado is born out of the asymmetric surrounding flow with initial horizontal velocity only. Their model is based on Navier-Stokes equations for compressible

flow of constant viscosity. By applying horizontal velocity only on the right boundary in Figure 3.1, rotation and uplifting effect are produced in their simulation. As stated in Section 3.2, there are many similarities between the boundary conditions applied in this simulation and theirs while the difference is also quite apparent, such as velocity directions and the velocity magitude control. By applying the specific boundary conditions presented in Section 3.2, this simulation can produce tornado-like rotation and uplifting effect. The results presented in Figure 5.1 are very similar to the rotation and uplifting effect Trapp *et al.* [38] achieved. In Figure 5.1, velocity vector information is demonstrated after boundary conditions specified in Section 3.2 are applied. In Figure 5.1(a), the velocity vectors of u and v close to the bottom, on the middle, and top X-Y planes in the cube physical domain are presented. As the result of the different boundary velocity directions in Figure 3.2, the counter-clockwise rotations are simulated on the X-Y planes. Moreover, because of the varied magitudes of u and v on the 4 vertical boundaries to simulate funnel-shape tornadoes, the vortex intensity close to the bottom boundary is higher, lower on the middle plane and the lowest on the top boundary. This result is consistent with what Trapp *et al.* [38, Figure 6] obtained in their simulation. Finally, the uplifting effect is illustrated in Figure 5.1(b) where the velocity vectors of u, v and w on the X-Z plane sliced in the middle of the cube domain are demonstrated. As presented in the image, the wind flow converges to the middle and goes up. This result also agrees with Trapp *et al.*'s discussion [38]. So qualitatively speaking, the model and boundary conditions applied in this simulation lead to a similar simulation result to Trapp *et al.*'s [38] but with more realistic boundary conditions.



(a) (b)

Figure 5.1: Velocity vectors on horizontal and vertical planes in the cube domain. (a) Velocity vectors of u and v close to the bottom boundary, on the middle and top X-Y planes. The counter-clockwise rotations are simulated and the rotation is more convergent to the plane center from the top to the bottom. (b) Velocity vectors of u, v and w on the X-Z plane sliced in the middle.

5.2 Visualization Results

Examples of tornado simulations in the Northern hemisphere are presented in this section. Note that the boundary conditions and velocity magitudes on the boundaries applied in the simulated tornado images are all same and described earlier in Section 3.2, but the time step size and particle control scheme may vary between different simulated features.

In Figure 5.2, images of a funnel-shape tornado and a torando with an angle are presented. Compared with the real photograph of tornadoes shown in Figure 2.2(a) and (b),

qualitatively speaking, this tornado simulation can generate tornadoes with interesting and realistic shapes, such as the funnel-shape tornado and a tornado with an angle in the middle. The parameters used in these two simulations are summarized in Table 5.1.

Another shape comparison with a real tornado is given in Figure 5.3. The funnel shape and the angle in the middle appeared in the real tornado in Figure 5.3(a) can be clearly observed in the simulated image. The parameter used in Figure 5.3(b) are same as those in 5.2(b) but with a different wind color for visualization.

Furthermore, the debris swirling effect at the bottom is simulated after centrifugal force and gravity are applied. The simulated tornado is compared with a real tornado in Figure 5.4. The simulated tornado shape is very similar to a real tornado which looks like an arrow, also the simulated debris swirling effect at the bottom observed in the animation is qualitatively close to that of real tornadoes. The parameters used in the simulation of Figure 5.4(b) are also presented in Table 5.1.

Finally, Figure 5.5 demonstrates a simulation of the dynamic shape changes of a tornado. To qualitively compare this shape change effect with that of real tornadoes, frames of a real tornado animation are presented in Figure 5.6. As shown in the frames, this tornado changes its shape dynamically during its lifetime. Qualitatively speaking, the visual result in Figure 5.5 demonstrates the similar effect. The parameters used in the simulation of Figure 5.5 are same as those of Figure 5.2(b), but the frames presented are of the earlier time steps. The details can be found in Table 5.1.

Note that the particles in this simulation do not interact with the environment, so collision detection is not performed in this thesis.

Table 5.1: The parameters used in the simulations

Image	Metaball Radius	Time Step Size	# of Particles ($t = 0$)	# of Particles in ($t > 0$)	Frame(s) order	Total # of Particles
Figure 5.2(a)	3	0.005	500,000	1,200/step	344th	996,800
Figure 5.2(b), 5.3(b)	3	0.00166	450,000	5,000/9 steps	932nd	970,000
Figure 5.4(b)	3	0.005	600,000	1,200/step, 40 of 1200 for bottom simulation	344th	1,096,800
Figure 5.5	4	0.001666	450,000	5,000/9 steps	702nd - 902nd	950,000

(a) (b)

Figure 5.2: Simulated tornado shapes. (a) A funnel-shape tornado. (b) A tornado with an angle.

(a) (b)

Figure 5.3: Shape comparison with a real tornado. (a) A funnel-shape tornado in Neal, Kansas, USA on May 30, 1982 (http://www.chaseday.com/tornado-neal.htm) (Courtesy of G. Moore). (b) A simulated tornado.

(a) (b)

Figure 5.4: Bottom debris swirling simulation and shape comparison with a real tornado. (a) An arrow-shape tornado with debris swirling effect at the bottom touched down near the town of Woonsocket, USA on June 24, 2003 (http://magma.nationalgeographic.com/ngm/0404/feature1/zoom3.html) (Courtesy of National Geographic). (b) A simulated tornado in an arrow shape with debris swirling effect at the bottom. The simulated debris movement after centrifugal force and gravity applied can be observed in the corresponding animation.

Figure 5.5: Frames of an animation to show the dynamic change of tornado shape.

Figure 5.6: Frames of a real tornado animation which dynamically changes shape during its lifetime (http://iwin.nws.noaa.gov/iwin/videos/tv5.avi) (Courtesy of the National Weather Service (NWS) of National Oceanic and Atmospheric Administration (NOAA), USA).

Chapter 6

Conclusion, Limitation and Future Work

In this thesis, I have proposed a physically-based tornado simulation method which produces realistic animation results. The model applied is based on the 3D Navier-Stokes equations. A particle system is incorporated with the equation solutions to model the unsmooth and ill-defined tornado shape, and a modified metaball scheme is applied in density calculation to smooth the density distribution. Also, the density level discretization strategy retains the necessary information for color variation in the rendering step as well as satisfying the efficient computing speed. Finally, the realistic images are generated using OpenGL based on volume rendering techniques.

The tornado studies in this simulation are useful to understand the tornado flow movement and maybe helpful for the understandings of tornado formation environment. The simulation method used here has the following advantages:

- the combination of physically-based modeling and graphics is used to simulate tornado flow movement.

- using the model in this simulation, appropriate boundary conditions lead to realistic tornado flow generation. Also, different and interesting tornado shapes are simulated by varying the number of particles generated at each time step.

- the density smoothing scheme can be easily modified by choosing different metaball radius and contribution funtion. Also, such changes can lead to different visualization results. For example, a larger metaball radius can be chosen if the user wants to see more details[1].

- image rendering speed can be improved by making use of graphics hardware. With the development of computing facilities, graphical cards with faster computing speed are helpful to improve this real time rendering process.

This simulation also has some limitations which are summarized below, as well as the work needs to be done in the future:

- the physical model used in this thesis is not fully physically-based. In this thesis, I intend to combine the modeling and graphics in order to generate and visualize the realistic tornado flow. But not enough physical factors are considered here due to limited time and research resource. In the future, more physical factors need to be integrated into the model in order to simulate the more realistic tornado formation environment, such as temperature.

[1]In the simulation of Figure 5.5, a larger metaball radius is used in order to see how the objects on the ground are drawn into the wind.

- the particle system applied in this simulation does not take into account the interaction with the environment. The collision detection is not implemented in this simulation. Also, the rendering does not consider the lighting physics, such as scattering. In the future, more physics and graphics need to be applied in this simulation to generate more realistic tornado images.

This simulation gives the framework on how to implement physically-based simulation of tornadoes. I recognize that this is the early step in developing such simulation in tornado studies. I hope my work here is helpful for readers to understand the tornado flow structure and useful for tornado scientists to develop a more physically-based model in the future.

Appendix A

Notation Table

u	velocity component on X axis
v	velocity component on Y axis
w	velocity component on Z axis
p	pressure
Re	reynolds number
g	gravity
φ_n	the component of velocity orthogonal to the boundary
φ_t	the component of velocity parallel to the boundary
F	force

m	mass
F^n	the force of a particle at n^{th} step
F_x^n	the x component of a particle's force at n^{th} step
F_y^n	the y component of a particle's force at n^{th} step
a^n	the acceleration of a particle at n^{th} step
a_x^n	the x component of a particle's acceleration at n^{th} step
a_y^n	the y component of a particle's acceleration at n^{th} step
x^n	the x component of a particle's position at n^{th} time step
y^n	the y component of a particle's position at n^{th} time step
u^n	the u component of a particle's velocity at n^{th} time step
v^n	the v component of a particle's velocity at n^{th} time step
ν	a control parameter used in applying centrifugal force
C	a particular cell in rendering
R	metaball radius in metaball scheme
f	density contribution function in metaball scheme
K	density before metaball scheme applied

D	density after metaball scheme applied
Ω	metaball influence region
t	time

Appendix B

On-line Tornado Resources

1. Torando chasing, pictures and videos
   - Chase Day: http://www.chaseday.com/chaseday5.htm
   - Weather Stock: http://www.weatherstock.com/
   - Storm Chasing Movies of Tornadoes and Thunderstorms: http://www.stormchase.us/stormmovies.html
   - Tornado, Hurricane, Thunderstorm Movies: http://www.mysteries-megasite.com/mysterymovies/tornmovie.html
   - Tornado Chaser: http://www.tornadochaser.com/
   - National Geographic Tornado Chasing: http://magma.nationalgeographic.com/ngm/0404/feature1/index.html

- University of Illinois:

  http://ww2010.atmos.uiuc.edu/(Gh)/guides/mtr/svr/torn/home.rxml

- Public Domain Tornado Images:

  http://www.spc.noaa.gov/faq/tornado/torscans.htm

- Efg's Computer Lab:

  http://www.efg2.com/Lab/ScienceAndEngineering/KCKTornado03/

- Online Tornado Museum: http://members.aol.com/tornadfoto/

- More links: http://www.chaseday.com/links-1.htm

2. Tornado FAQs

- The Tornado Project Online: http://www.tornadoproject.com/index.html

- The Online Tornado FAQs: http://www.spc.noaa.gov/faq/tornado/index.html

- Tornado Questions: http://www.tornadoproject.com/cellar/tttttttt.htm

3. Torando simulation

- PSC scientific visualization:

  http://www.psc.edu/research/graphics/gallery/tornado.html

- Tornado lab experiments: http://www.kilty.com/t_exper.htm

- Tornado lab model: http://www.nws.noaa.gov/om/educ/activit/tornado.htm

4. Research Institute

- National Severe Storms Laboratory (NSSL): http://www.nssl.noaa.gov/

– National Oceanic and Atmospheric Administration (NOAA): http://www.noaa.gov/

– Severe Weather Institute Research Lab:

http://movies.warnerbros.com/twister/cmp/swirl.html

Bibliography

[1] Baranoski, G., Wan, J., Rokne, J., and Bell, I. Simulating the dynamics of auroral phenomena. *ACM Transactions on Graphics 25*, 1 (2005), 37–59.

[2] Barr, A., Reeves, W., and Wolff, R. The physical simulation and visual representation of natural phenomena. *Computer Graphics 8*, 4 (1987), 335–336. SIGGRAPH Panel Discussion.

[3] Batchelor, G. *Introduction to Fluid Dynamics*, first ed. Cambridge University Press, Cambridge, UK, 1967.

[4] Bluestein, H., and Golden, J. A review of tornado observations. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 319–352.

[5] Bradford, S. A study of tornado and tornado producing synoptic systems. Master's thesis, University of Wisconsin, USA, 1969.

[6] Church, C., and Snow, J. Laboratory models of tornadoes. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 277–295.

[7] Dobashi, Y., Kaneda, K., Yamashita, H., Okita, T., and Nishita, T. A simple, efficient method for realistic animation of clouds. In *Proceedings of SIGGRAPH* (2000), pp. 19–28.

[8] Eastlund, B., and Jenkins, L. Taming tornadoes: Storm abatement from space. *IEEE Aerospace Conference Proceedings 1* (2001), 389–395.

[9] Etkin, D., Brun, S., Shabbar, A., and Joe, P. Tornado climatology of canada revisited: Tornado activity during different phases of enso. *International Journal of Climatology 21* (2001), 915–938.

[10] Fangmeier, S. Designing digital tornadoes. *Computer Graphics World 8* (1996), 65–73.

[11] Fedkiw, R., Stam, J., and Jensen, H. Visual simulation of smoke. In *Proceedings of SIGGRAPH* (2001), pp. 15–22.

[12] Fiedler, B. Numerical simulation of axisymmetric tornadogenesis in forced convection. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 41–48.

[13] Flora, S., Ed. *Tornadoes of the United States*. University of Oklahoma Press, Norman, Oklahoma, USA, 1953.

[14] Foster, N., and Fedkiw, R. Practical animation of liquids. In *Proceedings of SIGGRAPH* (2001), pp. 23–30.

[15] Gareau, A., and Vandorpe, D. A simple system for animation of natural phenomena. In *Visualization and Modeling*, R. Earnshaw, J. Vince, and H. Jones, Eds. Academic Press Inc., San Diego, USA, 1997, pp. 340–357.

[16] Glickman, T., Ed. *Glossary of Meteorology*, second ed. American Meteorological Society, Boston, USA, 2000.

[17] Griebel, M., Dornseifer, T., and Neunhoeffer, T. *Numerical simulation in fluid dynamics: A practical introduction*, first ed. Society for Industrial and Applied Mathematics, Philadelphia, USA, 1998.

[18] Harris, M., Baxter, W., Scheuermann, T., and Lastra, A. Simulation of cloud dynamics on graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS Conference on Graphics Hardware* (2003), pp. 92–101.

[19] Hatton, L. Coordinate transformations and two exact solutions of the axisymmetric navier-stokes equations. Computing Laboratory, Univeisity of Kent, UK, 2003.

[20] Holland, G. *Global Guide to Tropical Cyclone Forecasting*, first ed. World Meteorological Organization, Geneva, Switzerland, 1993.

[21] Houze, R. *Cloud Dynamics*, first ed., vol. 53 of *International Geophysics Series*. Academic Press, Inc., San Diego, USA, 1993.

[22] Jischke, M., and Parang, M. On tornado funnels. In *Oklahoma Academy of Science* (1978), pp. 81–87.

[23] Klemp, J., and Wilhelmson, R. The simulation of three-dimensional convective storm dynamics. *Journal of the Atmospheric Sciences 35* (1978), 1070–1095.

[24] Ladue, J. Vortex formation from a helical inflow tornado vortex simulator. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 307–316.

[25] Lagasse, P., Ed. *The Columbia Encyclopedia*, sixth ed. Columbia University Press, New York, USA, 2002.

[26] Laycock, G. *Tornadoes, killer storms*, first ed. David McKay Company, Inc., New York, USA, 1979.

[27] Lewellen, W. Tornado vortex theory. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 19–39.

[28] Lewellen, W., and Lewellen, D. Large-eddy simulation of a tornado's interaction with the surface. *Journal of the Atmospheric Sciences 54*, 5 (1997), 581–605.

[29] Nguyen, D., Fedkiw, R., and Jensen, H. Physically based modeling and animation of fire. In *Proceedings of SIGGRAPH* (2002), pp. 721–728.

[30] Nolan, D., and Farrell, B. The structure and dynamics of tornado-like vortices. *Journal of the Atmospheric Sciences 56* (1999), 2908–2936.

[31] Novlan, D., and Gray, W. Hurricane-spawned tornadoes. *Monthly Weather Review*, 102 (1974), 476–488.

[32] Reeves, W. Particle systems – a technique for modelling a class of fuzzy objets. *ACM Transactions on Graphics 2*, 2 (1983), 91–108.

[33] Rotunno, R. Numerical simulation of a laboratory vortex. *Journal of the Atmospheric Sciences 34* (1977), 1942–1956.

[34] Rotunno, R. Supercell tunderstorm modeling and theory. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 57–73.

[35] Savory, E., Parke, G., Zeinoddini, M., Toy, N., and Disney, P. Modelling of tornado and microburst-induced wind loading and failure of a lattice transmission tower. *Engineering Structures 23* (2001), 365–375.

[36] Sinkevich, O., and Chikunov, S. Numerical simulation of two-phase flow in a tornado funnel. *High Temperature 40*, 4 (2002), 604–612.

[37] Szoke, E., and Rotunno, R. A comparison of surface observations and visual tornado characteristics for the june 15, 1988, denver tornado outbreak. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 353–366.

[38] Trapp, R., and Fiedler, B. Numerical simulation of tornadolike vortices in asymmetric flow. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 49–54.

[39] Vesilind, P. Chasing tornadoes. *National Geographic 4* (2004), 2–38.

[40] Wang, I., Wan, J., and Baranoski, G. Physically-based simulation of plant leaf growth. *Computer Animation and Virtual Worlds 15* (2004), 237–244.

[41] Ward, N. The exploration of certain features of tornado dynamics using a laboratory model. *Journal of the Atmospheric Sciences 29* (1972), 1194–1204.

[42] Watt, A. *3D Computer Graphics*, second ed. Addison-Wesley Longman Publishing Co., Inc., Boston, USA, 1993.

[43] Westover, L. *Splatting: a parallel, feed-forward volume rendering algorithm.* PhD thesis, University of North Carolina at Chapel Hill, NC, USA, 1991.

[44] Whiteman, C. *Mountain Meteorology: Fundamentals and Applications*, first ed. Oxford University Press, New York, USA, 1999.

[45] Wyvill, G., and Trotman, A. Ray-tracing soft objects. In *Proceedings of the eighth international conference of the Computer Graphics Society on CG International '90: computer graphics around the world* (1990), pp. 469–476.

[46] Xia, J. *Large-Eddy Simulation of a Three-Dimensional Compressible Tornado Vortex.* PhD thesis, West Virginia University, USA, 2001.

[47] Xu, Z., Wang, P., and Lin, X. Tornadoes of china. In *The Tornado: its structure, dynamics, prediction, and hazards (Geophysical Monograph 79)* (1993), pp. 435–444.