ART DIRECTABLE TORNADOES

A Thesis

by

RAVINDRA DWIVEDI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

May 2011

Major Subject: Visualization

Art Directable Tornadoes

ART DIRECTABLE TORNADOES

A Thesis

by

RAVINDRA DWIVEDI

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Approved by:

Chair of Committee,	Vinod Srinivasan
Committee Members,	John Keyser
	Wei Yan
Head of Department,	Tim McLaughlin

May 2011

Major Subject: Visualization

ABSTRACT

Art Directable Tornadoes. (May 2011)

Ravindra Dwivedi, B.E., Rajiv Gandhi Proudyogiki Vishwavidyalaya

Chair of Advisory Committee: Dr. Vinod Srinivasan

Tornado simulations in the visual effects industry have always been an interesting problem. Developing tools to provide more control over such effects is an important and challenging task. Current methods to achieve these effects use either particle systems or fluid simulation. Particle systems give a lot of control over the simulation but do not take into account the fluid characteristics of tornadoes. The other method which involves fluid simulation models the fluid behavior accurately but does not give control over the simulation. In this thesis, a novel method to model tornado behavior is presented. A tool based on this method was also created. The method proposed in this thesis uses a hybrid approach that combines the flexibility of particle systems while producing interesting swirling motions inherent in the fluids. The main focus of the research is on providing easy-to-use controls for art directors to help them achieve the desired look of the simulation effectively. A variety of controls is provided which include the overall shape, path, rotation, debris, surface, swirling motion, and interaction with the environment. The implementation was done in Houdini, which is a 3D animation software whose node based system allows an algorithmic approach to the problem and integrates well with the current tools. The tool allows the user to create

animations that reflect the visual characteristics of real tornadoes. The usefulness of the tool was evaluated among participants who had some experience in 3D animation software. The results from the simulation and evaluation feedback reveal that the tool successfully allowed the users to create tornadoes of their choice efficiently.

DEDICATION

I dedicate this thesis to my family.

ACKNOWLEDGEMENTS

I would like to thank my committee chair, Dr. Vinod Srinivasan and my committee members, Dr. John Keyser and Dr. Wei Yan for their guidance and support throughout the course of this thesis. Special thanks to Dale Mayeda and Rob Dressel for their inputs during the preliminary stages of the research and their feedback during the later stages which helped refine the tool.

Thanks, also, to the Head of the Department, Tim McLaughlin, for his encouragement and to my colleagues, in the Department of Visualization at Texas A&M University, for their support. I also want to extend my gratitude to the participants who helped in the evaluation of the tool.

To my mother and father, I cannot thank you enough for your patience and love and making me a person I am today. I am grateful to my brother, Ajendra and sister, Pooja, for boosting my confidence and being supportive. I am very appreciative to my brother-in-law, C.P. Tiwari, for not only supporting my education but also encouraging me. Thank you to all the family members for their love.

I will always be grateful to the teachers and staff of my boarding school, Rajkumar College, Raipur, where I learned the importance of education. Finally, a special thanks to all my friends for being an important part of my life.

TABLE OF CONTENTS

CHAPTER Page

LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

CHAPTER I

INTRODUCTION

A tornado is considered to be one of the most violent types of storms produced by nature. Broadly speaking, it is a whirlpool of winds swirling at very high speeds with tremendous strength to lift objects along its path. The wind speed has been estimated to be 450-500 miles an hour in many cases. Usually it is formed at great heights in a continuously rotating thunderstorm known as supercell. At the center of this thunderstorm, a vortex begins to form due to wind shears and starts to descend towards the ground. Usually it appears to be a pendant of clouds with more or less a funnel shape as seen in Figure 1, although other forms of tornado have also been noticed [Flora 1954].

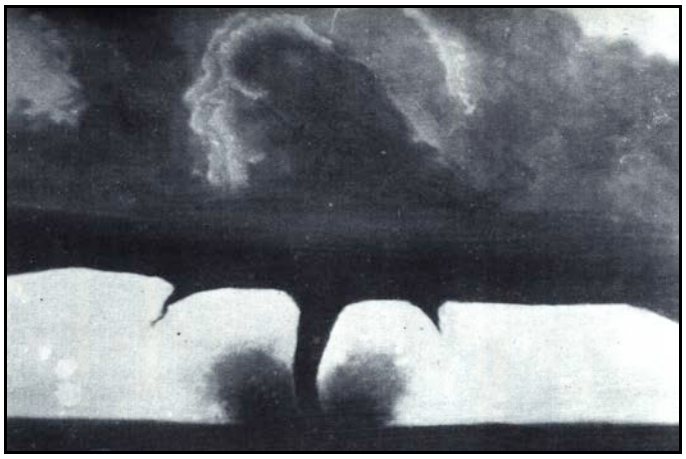


Figure 1: One of the first tornado photographs. August 28, 1884, South Dakota. Photo from [Flora 1954]. Courtesy of U.S. Weather Bureau and F.W. Robinson.

This thesis follows the style of *ACM Transactions on Graphics*.

Tornado simulations have been done in a number of different ways. In early 1950s and 1960s, rotating tanks of water were used to study tornadoes. Later researchers started using vortex chambers with fans to understand its dynamics [Bluestein 2006]. With the advent of computers, numerical simulations became another method for researchers with the advantage that this method was safe and did not require setting up complex physical equipments.

Tornadoes are an important visual effects element in movies. Physically based simulations of tornadoes have been done in the past using fluid simulations but they offer very little control over the behavior and cannot be easily integrated in a production environment. On the other hand, simulations developed for purely visual purposes for movies such as Twister [Bont 1996] were based on particle systems which lack fluid characteristics inherent in tornadoes. Providing animators control over the simulation for art directing purpose while maintaining the fluid behaviors present in real tornadoes to make it more believable is therefore important.

In this thesis, a novel method to model tornado behavior is introduced and an animation tool has been developed to control important aspects of the simulation. The tool was developed as a plug-in in Houdini [SIDE EFFECTS SOFTWARE Inc. 2011] which can be easily incorporated in a production pipeline.

CHAPTER II

BACKGROUND, RELATED WORK

II.1. Motivation

1. Need for Art-directed Simulations

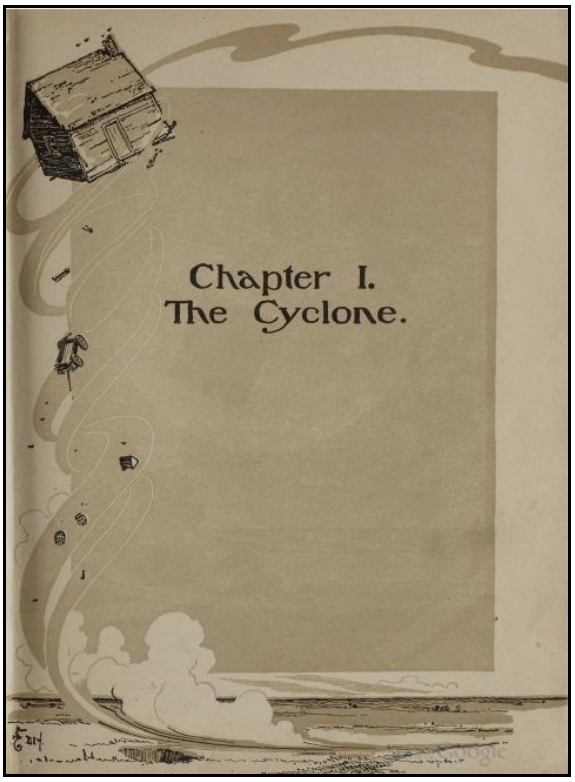


Figure 2: "The Wonderful Wizard of Oz". Illustration by W.W. Denslow [Baum 1900].

"The Wonderful Wizard of Oz" [Baum 1900], a children's novel written by L. Frank Baum (Figure 2) had a reference to a tornado in which a little girl named Dorothy gets caught in the vortex and reaches a strange land. The book became so popular that it

led to its adaptation to other media numerous times, most popularly in the film "The Wizard of Oz" [Fleming 1939]. The film was considered innovative because of its special effects used to create a tornado [Wikipedia 2011]. The tornado in the film was made from muslin cloth which was kept flexible so that it could be bent and twisted as desired [Tausif and Novara 2010].

Since then, tornado simulations in the entertainment industry have appeared in several motion pictures as an important effects shot. Directing the simulation as per the shot requirement has always been a challenging problem. In 1996 computer generated tornadoes were used on a large scale in the movie “Twister” (Figure 3). The tornado simulation in the movie was done by using particle systems and rendering them as fluids [Bont 1996].

Figure 3: Simulated tornadoes in "Twister". Photo from [Bont 1996].

Simulated tornadoes also appeared in movies “Night of the Twisters” [Bond 1996], “Tornado” [Nosseck 1996] and “Hancock” [Berg 2008].

(a) (b)

Figure 4: Simulated tornadoes in "The Day After Tomorrow". (a) Twin tornadoes [Emmerich 2004]. (b) Tornado with a huge funnel [Emmerich 2004].

The film "The Day after Tomorrow" [Emmerich 2004], had a variety of tornadoes with different shapes and sizes and the shots required a lot of art direct-ability to make it visually appealing and believable (Figure 4). In 2009, an animated movie “Cloudy with a Chance of Meatballs” had a stylized tornado of spaghetti sauce (Figure 5) which was simulated using curves attached to the particles [Lord 2009].

Figure 5: Spaghetti Tornado. Scene from "Cloudy with a Chance of Meatballs" [Lord 2009].

For the visual effects to be most effective it is very important that the artists should be able to control the simulation to get the desired look. Lack of flexibility may result in animators spending a lot of time adjusting the simulation to achieve the results which may not be cost effective. It may also not lead to the director's vision. On the other hand, a system that provides easy to use controls over the simulation allows the animators to pay more attention to the aesthetics rather than worrying about technical details.

The design and implementation of my tool is inspired by such requirements in an entertainment industry which is to art-direct a simulation as per the shot requirement in a production environment effectively.

II.2. Tornado and Its Characteristics

1. Description

Tornadoes are rare phenomena's of nature which last for a very short span of time but are capable of leaving behind great calamity. Most of the tornadoes are formed in the chaos of thunderstorms known as mesocylones. A tornado consists of air and water vapor rotating at very high speeds with great uplifting strengths. Dust and debris tend to be picked up in the bottom few hundred feet. Some of the lighter debris is carried to great heights while most of it tends to be thrown around the vortex. On an average a tornado is only about 50 yards wide [Grazulis 2001].

2. Color

Figure 6: The Tracy Minnesota Tornado. June 13, 1968. Photo from Thomas P. Grazulis, *The Tornado: Nature's Ultimate Windstorm* [Grazulis 2001]. Photograph by Eric Lantz.

The color of the funnel when the debris has not been picked up into the vortex is usually milky white as seen in Figure 6. Depending upon the debris being sucked into the whirl, its color gets darkened giving it a frightening appearance.

3. Path

The direction of movement of cold fronts associated with a tornado determines its path. In the northern hemisphere, the fronts extend from north to south causing the tornadoes to move from southwest to northeast. They are also known to change their paths suddenly near the end of their paths [Flora 1954].

4. Rotational Direction

Tornadoes have rotating movement because of the shear of winds blowing in different directions. The opposing currents of air thus determine the direction of the whirl. In the northern hemisphere, the warm moist air to its east moves from southerly direction as opposed to cold air moving from west to northwesterly direction. As a result, the direction of rotation is counterclockwise [Flora 1954]. In contrast, the southern hemisphere has rotational direction clockwise due to opposite directions of winds [Flora 1954].

5. Shapes

A tornado may form a variety of shapes depending upon the immediate conditions of environmental factors such as air, pressure, temperature and moisture

[Grazulis 2001]. The funnel shape of a tornado is very common form which distinguishes it from other types of storms. According to Flora,

> "to some it resembles the trunk of a huge elephant as it weaves back and forth across its path. Often it resembles a long rope or a huge snake dangling in the sky. On occasions it has had the appearance of an almost vertical column. In rare occasions it has been known to be shapes like an hourglass with mid-section almost invisible." [Flora 1954].

Figure 7 shows a tornado in its rope stage before it disappears.

Figure 7: Tornado in its rope stage before disappearing. Photo from Laramie [Chandler]. Courtesy of Stephen Hodanish.

6. Types

For a phenomenon to be classified as a tornado it should extend from base of the cloud to the ground and it should have a swirling vortex.

Vortices that form as a result of forest fire (Figure 8a) do not fall under the category of tornadoes despite the fact that they connect with the cloud overhead [Grazulis 2001]. A waterspout (Figure 8b) is a vortex over a water body. Waterspouts passing from water to land tend to become tornadoes and tornadoes which pass from land to water become waterspouts [Flora 1954].

(a) (b)

Figure 8: Fire vortex and Waterspout. (a) Fire Vortex [Chandler]. (b) Waterspout off the Florida Keys [Chandler]. Courtesy of [National Oceanic and Atmospheric Administration 1999].

A landspout (Figure 9a) is another form of tornado that is formed away from a supercell. A gustnado (Figure 9b) on the other hand, is a short-lived cyclonic cloud that

can form under severe storm conditions. Its vortex does not connect to the clouds overhead and is not classified as a tornado [Wikipedia].

(a) (b)

Figure 9: Landspout and Gustnado. (a) Landspout tornado crossing a farm field [Reed]. Courtesy of Jim Reed. (b) Gustnado in Wisconsin on October 4, 2002 [National Weather Service 2011].

II.3. Tornado Dynamics

Understanding the complete dynamics of a tornado is an ongoing research topic on which many theories have been formulated. In many ways, it can be compared to a whirlpool that is formed when water is draining from a bathtub. Due to gravity, the water in the tub moves downwards while in a tornado the air gets drawn upwards due to rising currents. In both scenarios, the converging current causes the rotation. The law of conservation of angular momentum states that the product of velocity multiplied by the distance from the axis of rotation remains constant, except as reduced by frictional forces. As a result, the speed of the currents is more towards the center and it decreases

as we move away from the axis (Figure 10). The core of the tornado is known be partial vacuum, an effect which is similar to the result of centrifugal force causing a hollow inverted cone formation in a bathtub whirl [Flora 1954].

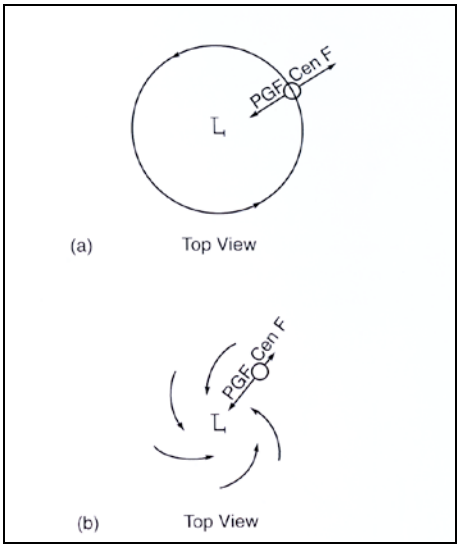


Figure 10: Vortex dynamics. (a) The pressure gradient force (PGF) and centrifugal force (CenF) balance each other above the ground. (b) The reduction in the wind speed due to the drag of the ground reduces the centrifugal force so that it is less than the pressure gradient force. Photo from [Bluestein 2006].

Near the ground, reduction in wind speeds due to drag of the ground causes the inward directed pressure gradient to be more than the outward directed centrifugal force. This results in air converging towards the center of the vortex and unable to flow into the ground it tends to rise near the center (Figure 11). This causes a compensating sinking motion away from the center [Bluestein 2006].

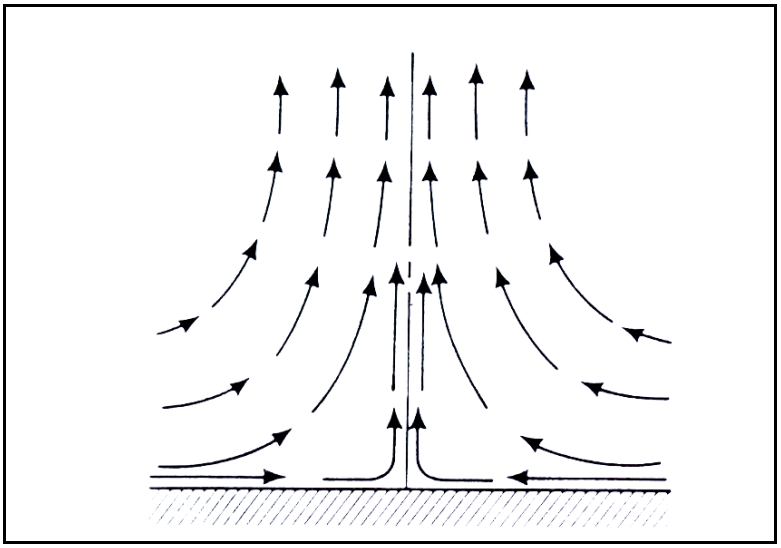


Figure 11: Vertical cross section profile in single vortex tornado. Air rises at the center while a compensating sinking motion happens away from the core. Figure from [Bluestein 2006].

Outside the vortex, wind speeds fall off rapidly at first and then gradually with the distance from the center of the vortex as seen in Figure 12 [Bluestein 2006]. Since the outward directed centrifugal force is proportional to the density, chunks of debris that gets sucked into the vortex causes this force to be greater than the inward directed pressure gradient force, as a result of which an envelope of debris lying around the tornado is seen [Bluestein 2006].

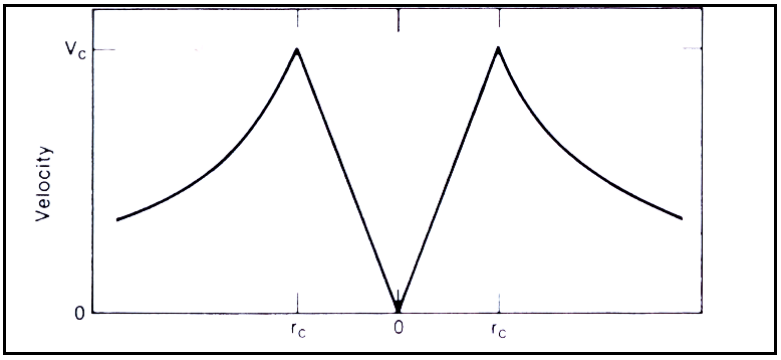


Figure 12: Profile of tangential component of wind in a vortex. The maximum wind velocities (V_c) are found at the "core" radius, r_c. Figure from [Bluestein 2006].

II.4. Existing Simulations

Some of the methods to study the dynamics of a tornado are modeling physical equipments such as vortex chambers, using rotating tanks of water and simulations in computers.

Designing vortex chambers mainly involves using fans which sucks the air in the chamber and rotation is applied along the edges. Dry ice is then used to visualize the flow of air (Figure 13).

Figure 13: Vortex chamber. Photo from [Chandler]. Courtesy of Michael Ellestad.

This method can only give some information regarding vortices in a tornado but not its entire dynamics which is much more complex and involves measuring parameters such as wind, pressure and temperature. Another challenging task to the researchers is to measure parameters of these vortices without interfering with the vortex [Bluestein 2006].

On the contrary, modeling tornado simulations in computers is achieved using numerical methods. In order to model a physically based simulation of tornado, parameters such as pressure, temperature and water vapor are initialized similar to conditions present in the weather. Then using basic principles of physics such as the

amount of latent heat released in water molecules and equations of airflow, a digital simulation is created [Grazulis 2001].

Numerical cloud simulation was pioneered by Robert Schlesinger (1975) at the University of Wisconsin followed by Joseph Klemp (National Center for Atmospheric Research) and Robert Wilhemson (1978) of the University of Illinois, who made exceptional computer models to simulate storms [Grazulis 2001]. The results from these simulation helped understand flow of air within a mile of tornado on the ground [Grazulis 2001]. In the year 1993, Lou Wicker and Robert Wilhemson were able to simulate tornado like vortex protruding from a thunderstorm with finer details on a small grid space with the help of computers with fast processing powers [Grazulis 2001]. Although the approaches mentioned above strive for physical accuracy in their simulations, they do not pay much attention to the visual aspects, thus making the simulation visually less interesting.

However, there have been works that focus on the visual features. Shi-guang et al. presented a physically based method for simulating a tornado scene based on the Reynolds average two fluid model [Liu, Wang et al. 2006]. Xiangyang used a particle system with the model equation solution to simulate the irregular tornado shapes. He used three-dimensional Navier-Stokes equations for incompressible viscous fluid flow to model the tornado dynamics [Ding 2004]. Although these methods are based on mathematical modeling and model a realistic tornado environment, they don’t give much control over the simulation and cannot be used in a production environment where focus is on art directing the simulation.

Simulating tornadoes with focus on providing controls to the animators was presented in the Tornado Report [Tausif and Novara 2010]. They implemented tornado simulation using particle systems in Houdini. Their method provided good control over the shape but the debris simulation was not visually realistic and interaction with surrounding objects was not taken into account. The solution provided by them was Houdini specific and was not generic to be implemented using other computer languages or software. The method also did not take into account the fluid behavior noticeable in real tornadoes. Some of the particle based techniques used in this thesis are based on the concepts presented in this report.

CHAPTER III

METHODOLOGY

III.1. Visual Inspiration

Apart from the research in the previous sections, visual references in the form of photographs and videos were also collected to study the visual elements of tornadoes. Below are some of the studies of features which have been implemented in this thesis.

1. Features

Each tornado has different features which are governed by its dynamics and other environmental factors. Figure 14 below shows the more common funnel shaped appearance and comparison of its size with houses and trees. Wall clouds at the upper part of the funnel and debris at the bottom can be seen in Figure 15.

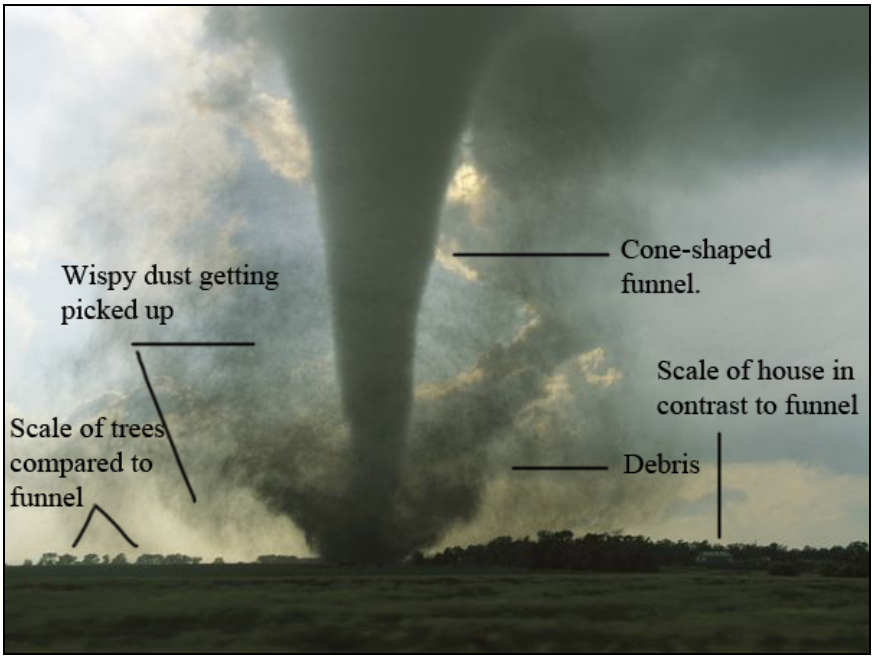


Figure 14: A tornado across South Dakota prairie. It had a rating of F3. An F3 tornado has wind speeds between 158 and 206 miles an hour [Peter].

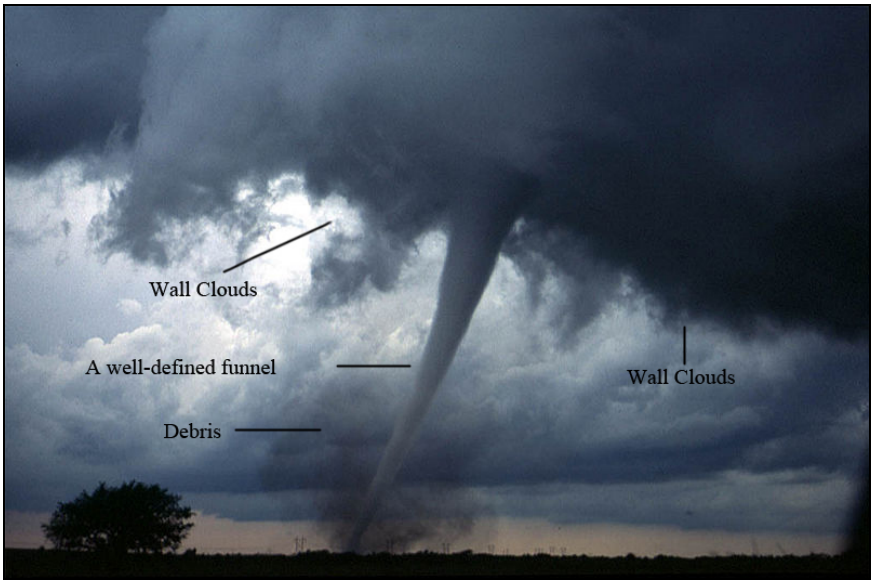


Figure 15: Main sections of a tornado. Location, Anadarko [Chandler]. Courtesy of [National Oceanic and Atmospheric Administration 1999]

Figure 16: Debris sheath at the base. Courtesy of Jim Reed and Katie Bay [Chandler].

Figure 16 shows an envelope of debris forming at the base of a tornado. Figure 17 and 18 show the debris from a house getting caught inside the vortex and swirling around the funnel. Dust on the ground can also be seen being drawn towards the funnel due to strong vortex forces.

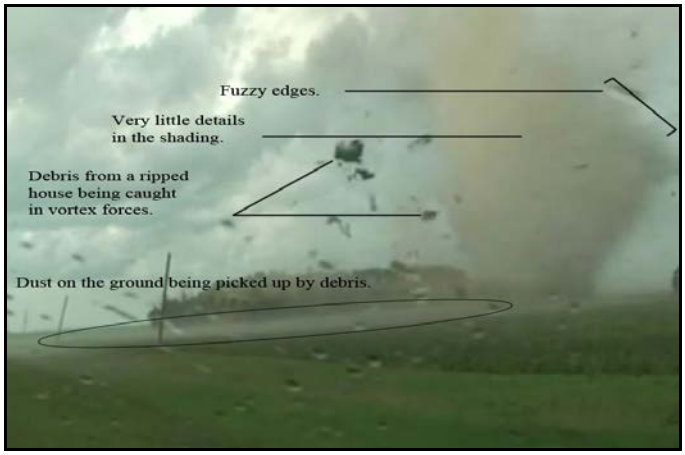


Figure 17: Debris from a house gets caught in the vortex. Photo from [AssociatedPress 2010].

Figure 18: Dust on the ground being sucked in the funnel. Photo from [NationalGeographic].

2. Shapes

A tornado funnel can take a variety of forms depending on the immediate environmental conditions inside and outside the funnel. Below are some of the studies of variety of shapes that a tornado may form. Figures 19, 20, 21, 22 and 23 have been edited to highlight certain aspects which were useful in designing the tool. In these figures, black, red and green lines emphasize what we think of as shape of a tornado. The black lines show the primary axis of the funnel, the red lines highlight the shape of the funnel with varying thickness along the length and the green lines are approximation of the shape of debris at the bottom of the funnel.

Figure 19: Tornado in Lombok, Indonesia. Notice the cylindrical funnel in contrast to the usual funnel shaped structure [Chandler]. Courtesy of Fadil Basymeleh.

(a) (b)

Figure 20: Rope and funnel shapes. (a) Rope out of Cordell, Oklahoma, tornado of May 21, 1981 [Grazulis 2001]. Courtesy of NSSL. (b) The more common funnel shaped tornado on April 12, 1991, east of Pondcreek, Oklahoma [Bluestein 2006].

(a) (b)

Figure 21: Variations in funnel shapes. (a) Laminar-to-turbulent flow in a tornado in southeast Colorado [Chandler]. Credit Linda Lusk. Courtesy of NCAR. (b) Condensation funnel and dust sheath in Arkansas tornado [Chandler]. Courtesy of Arkansas Tech.

(a) (b)

Figure 22: Tornado at its rope and mature stages. (a) Tornado near Lawrence at its Rope stage [Chandler]. Courtesy of [NC911 2004]. (b) Tornado with an F5 rating in Elie, Manitoba, 2007 [Chandler]. Courtesy of Justin Hobon.

Figure 23: A slanting tornado passing near Nebraska. April 23, 1989. Merrille Thomas photographed her daughter, Audra, standing about a mile from the funnel [Grazulis 2001]. Copyright Merrille Thomas.

III.2. Design

The design strategy of the tool was centered on making it easy to use and give as much control to the user to help achieve complex tasks very quickly. In order to meet the above goals, the simulation was divided into two parts, particle simulation and fluid simulation. This approach made the maintenance easier. This section explains both parts

in a step by step manner. For each step, the most commonly used terms are explained below.

1. Metaball

Metaballs are spherical force fields whose fields are determined by a function and a weight. The function defines the gradient density value of the field from 1 at the center to 0 towards outside (Figure 24). The density of the force fields gets added (Figure 25) if the metaballs overlap [SIDE EFFECTS SOFTWARE Inc 2011].

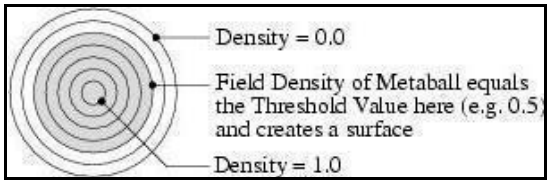


Figure 24: Density field weight distribution in Metaball. Figure from [SIDE EFFECTS SOFTWARE Inc. 2011].

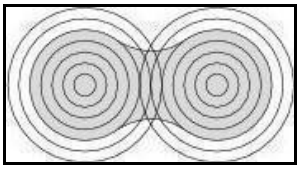


Figure 25: Resulting fields when two Metaballs intersect. Figure from [SIDE EFFECTS SOFTWARE Inc. 2011].

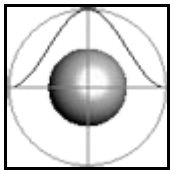


Figure 26: Function used for weight distribution. Figure from [SIDE EFFECTS SOFTWARE Inc. 2011].

The diagram above (Figure 26) shows the weight distribution function within the metaball surface which has been used in the implementation.

2. Directional Force

A directional force can be defined as linear force acting on particles whose direction is determined by a vector. It can be defined as:

$$F = |F|\, d. \tag{1}$$

In this equation, '*F'* is a force vector, $|F|$ is magnitude of vector and '*d'* is the vector that determines the direction in which the force will be acting.

3. Vortex Force

A vortex force acting on particles causes them to orbit about an axis along a circular path. Increasing the force value will increase the amount of twist around the

primary axis. This force is responsible for rotating particles around the profile curve as used in this thesis (Figure 27).

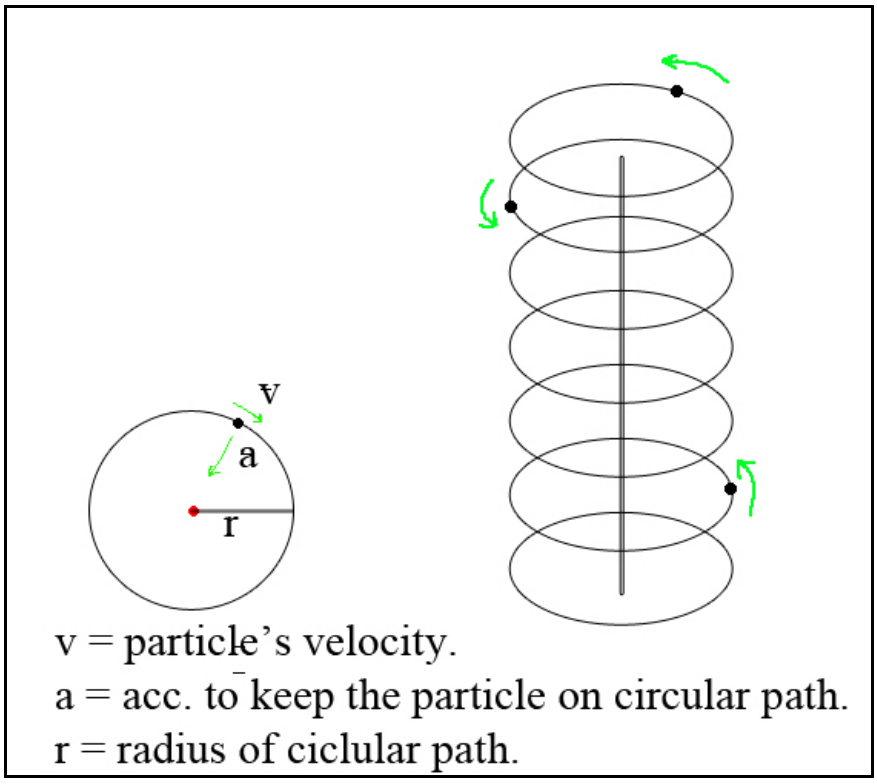


Figure 27: Vortex force causes objects to orbit around an axis.

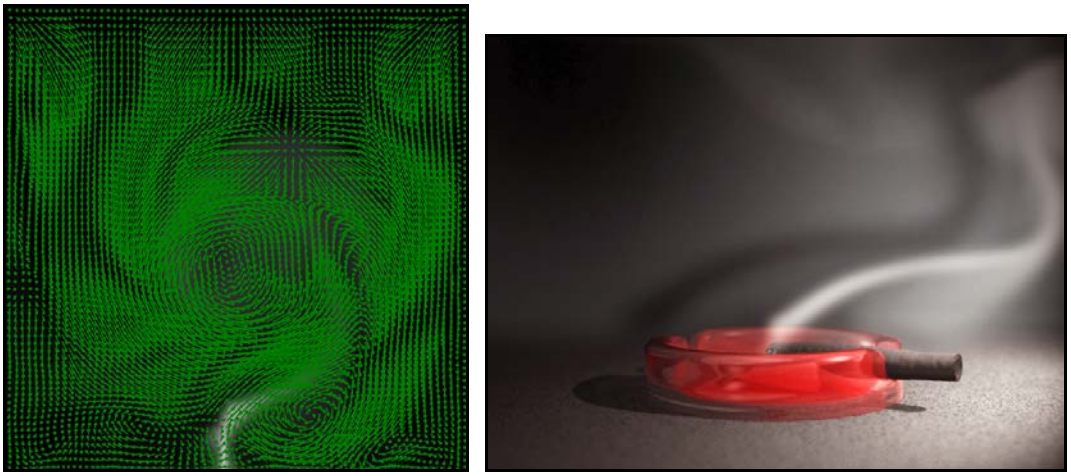
4. Lattice

A lattice is responsible for deforming a target geometry based on the modifications applied to the source geometry. The lattice used in this thesis applies the deformations to the particles based on changes made to the profile curve.

5. Fluid Simulation

Navier-Stokes equations describe the motion of fluids and are widely used in computer graphics to simulate substances that exhibit such behaviors. Smoke is an example of a fluid that exhibits unique rolling turbulent motion as it moves through the air. This swirling motion is known as vorticity in fluid dynamics. Stam [Stam 1999] presents a method which can be used to simulate fluids behaviors using Navier-Stokes equation. Smoke can also be modeled using their approach but the method proposed by Stam suffers from too much numerical dissipation as a result of which the vortices vanish too rapidly. In order to keep this dynamics behavior of smoke, Fedkiw et al. [Fedkiw, Stam et al. 2001] formulated a method to keep this swirling motion in fluids alive by calculating the curl at every voxel of the grid container and enhancing it. Figure 28 shows simulated vorticity on a 100x100 resolution grid.

Since a tornado is basically a fluid turning at a uniform rate, it exhibits very interesting fluid characteristics [Grazulis 2001]. Some of these features can be seen in Figure 29 along the edges of the tornado funnel. These fluid behaviors are modeled using vorticity forces in fluid simulation.



(a) (b)

Figure 28: Smoke simulation. (a) Grid based smoke simulation equation with vorticity confinements. (b) Velocity fields on a two dimensional 100x100 grid based smoke simulation demonstrating flow fields. Note the curls which are responsible for vorticity.

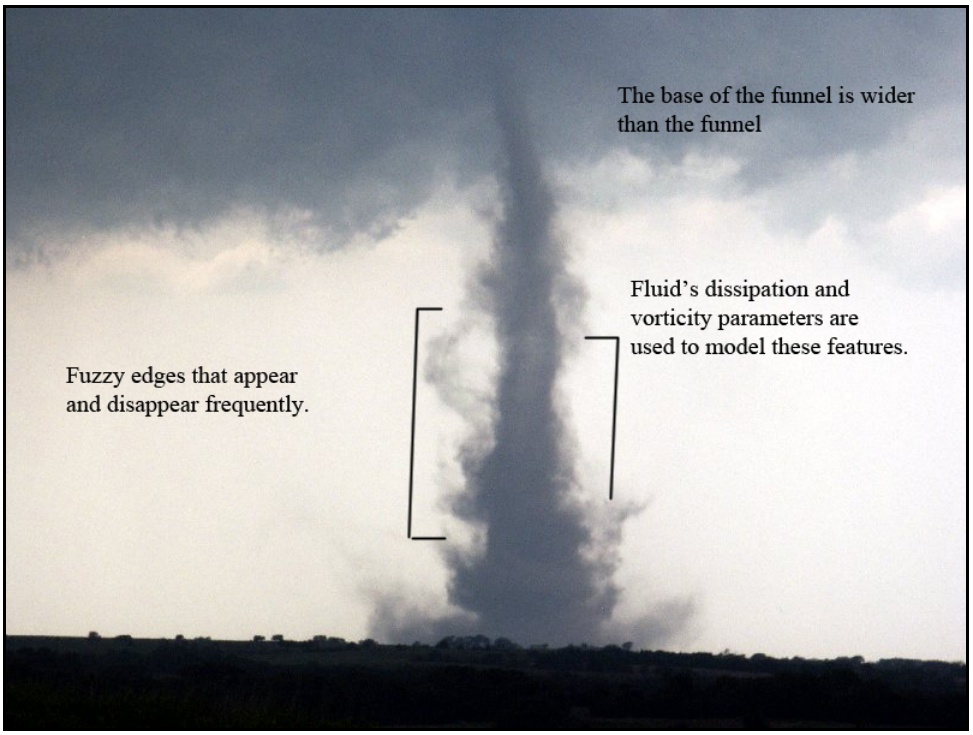


Figure 29: Fuzzy edges are simulated using vorticity parameter. Photo from [NC911 2004].

A detailed explanation of fluid simulation steps is beyond the scope of this thesis. "Realistic Animation of liquids" [Foster and Metaxas 1996], "Stable Fluids" [Stam 1999] and "Visual Simulation of Smoke" [Fedkiw, Stam, et al. 2001] provide additional information about modeling fluid behaviors.

III.3. Particle System

1. Animatable Curve

The modeling of tornado funnel began with creating a profile curve. This curve was drawn as a straight line perpendicular to the XZ plane and it was kept smooth to give the funnel a uniform regular surface. This curve formed the basic rig which was used to control the particle systems which later directed the fluids. It is also used by animators to animate shapes by key framing the control vertices along the X-axis and Z-axis (Figure 30).

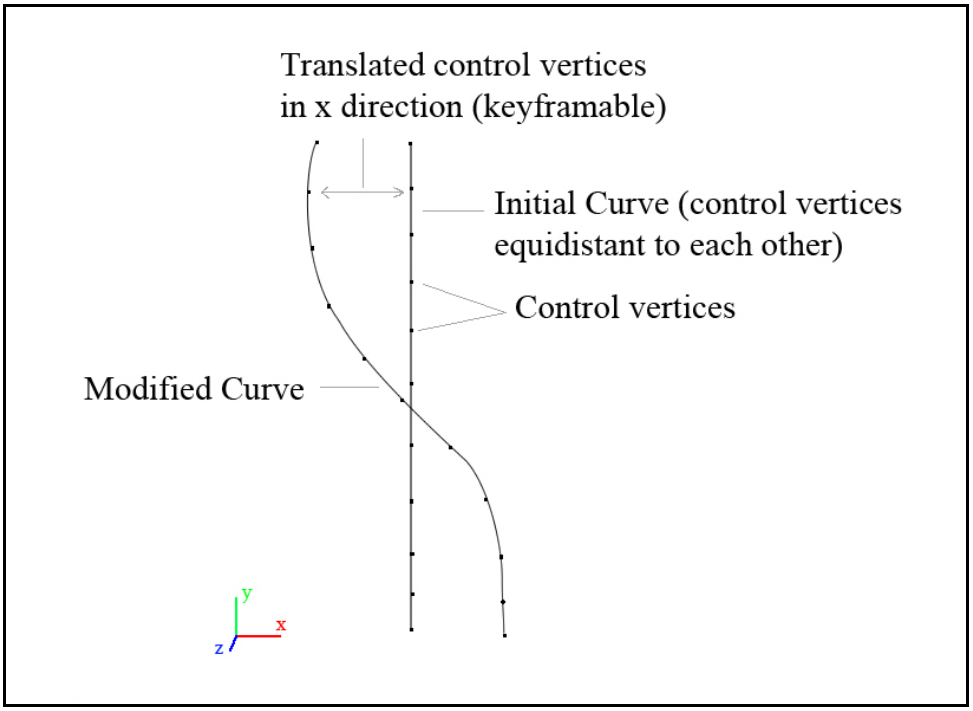


Figure 30: Initial profile curve and modified curve.

Only a few control vertices in the curve are sufficient to shape the funnel. In this implementation, seven control vertices were used. Since the lattice function defines its influence region to attract the particles around each vertex, increasing the number of vertices will provide finer controls over particles (Figure 31). Further, the control vertices in the curve were placed equidistant to each other to provide a uniform distribution of points along the curve.

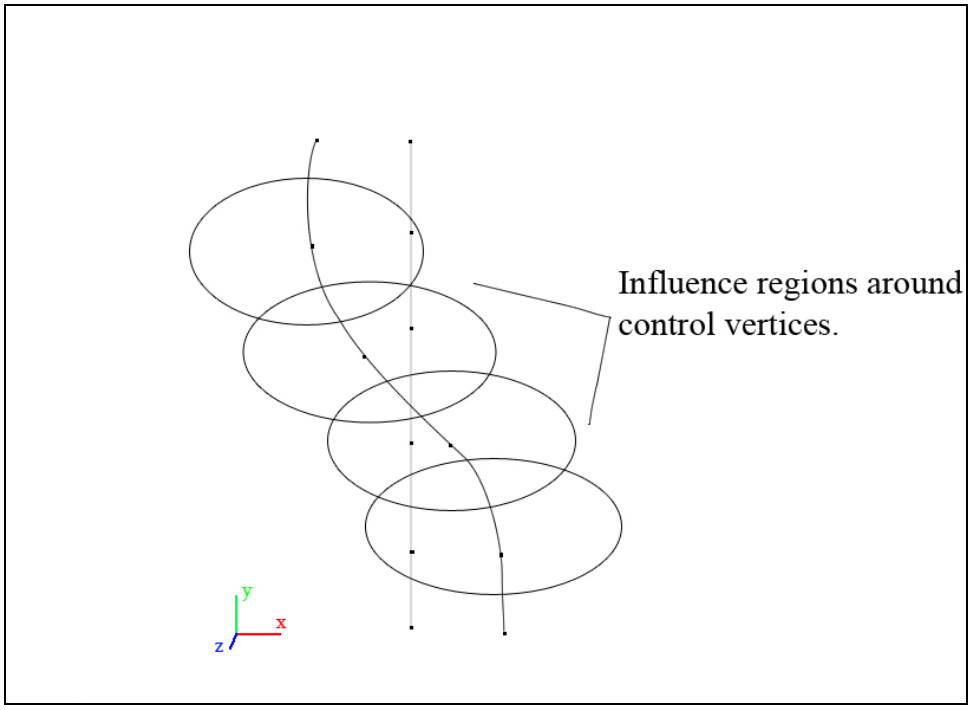


Figure 31: Each vertex has its own influence region. This determines how tightly the particles are attracted to it.

2. Particle Emitter

A circle was then created and constrained to the top of the curve and its vertices were used as emitters of particles. These particles have no mass property hence they are not affected by gravity, unlike debris particles. The diameter of the circle was kept large enough to emit particles with a uniform distribution around the curve. To avoid regular emission of particles from the emitter over time, the vertices of the emitter were jittered with a random function of time and space as seen in Figure 32.

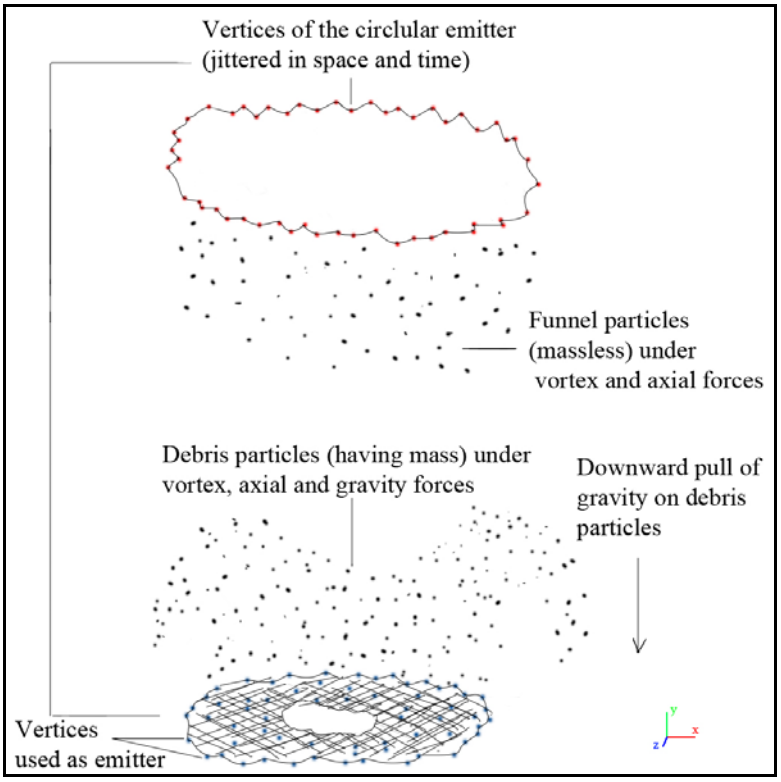


Figure 32: Particles being emitted. Directional and vortex force causes the particles to swirl and move towards the ground.

3. Funnel Force Fields

There are two kinds of forces acting on particles, directional and vortex force. The directional force is responsible for pushing the particles in negative Y-axis and the vortex force gives the particles a swirling movement around the Y-axis (Figure 33).

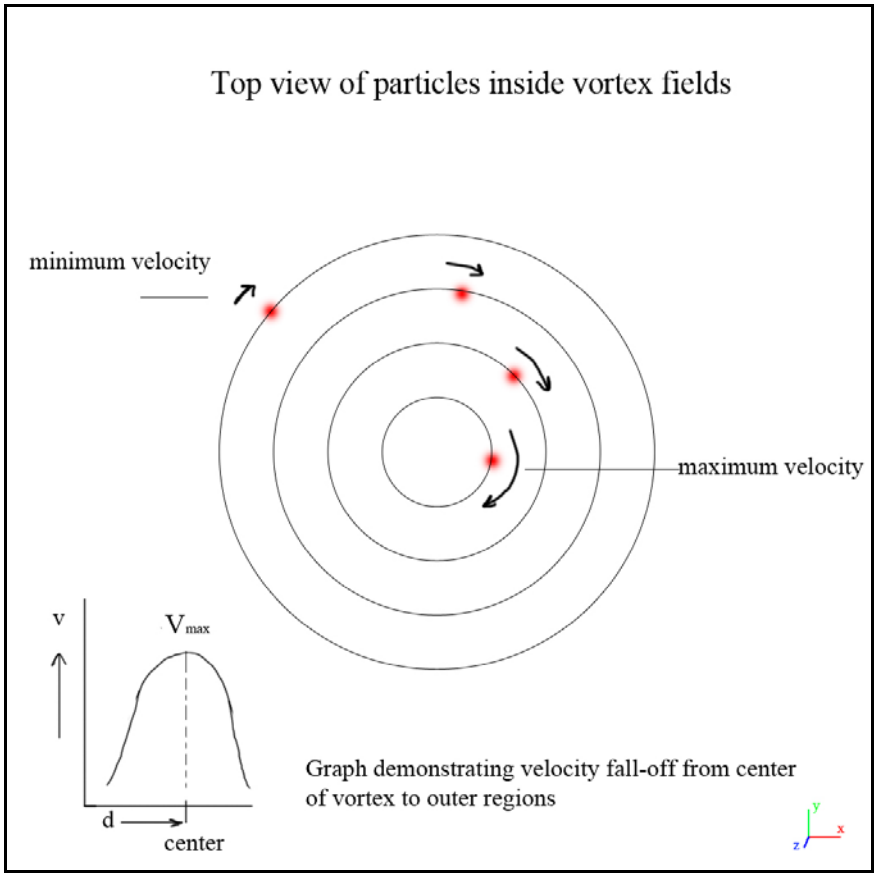


Figure 33: Particle velocities varying with weighted function.

According to the tornado dynamics, the wind speeds outside the core are more and it decreases as we move towards outer regions of the funnel. To create a similar effect, the vortex forces acting on particles were defined within a region with a weighted distribution. Metaballs were used to define the region of influence of forces. The force fields in these regions were then distributed using functions which were maximum at the center and had an exponential fall-off towards outer regions. As a result, particles were forced to swirl with varying velocities based on their distance from the axis as shown in Figure 34.

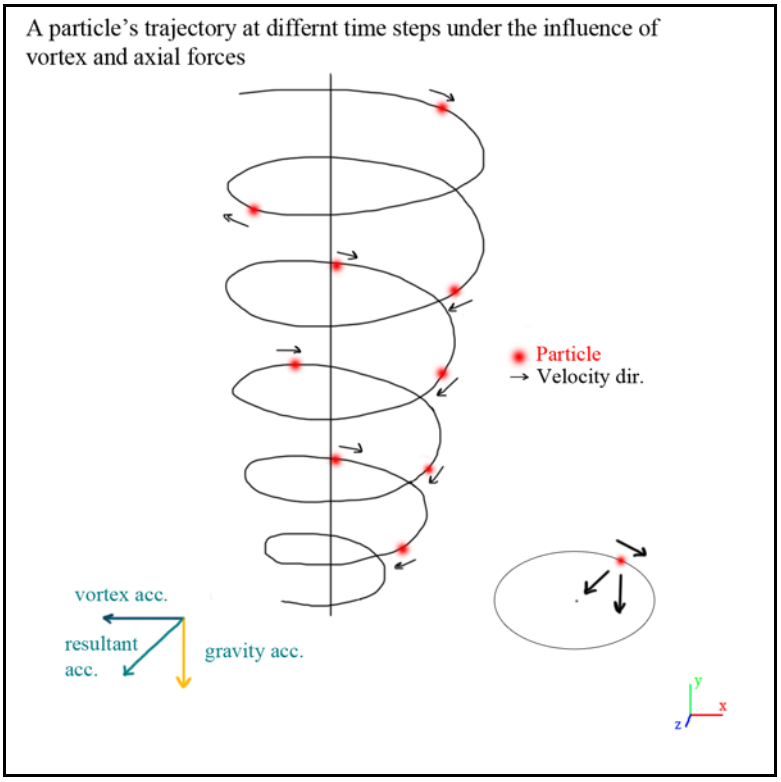


Figure 34: Particles under the influence of vortex and directional force.

In this implementation two metaballs were used, one at the top which represents the wall cloud and one for the primary funnel region. Wherever the regions intersected the forces were added.

4. Deforming Particles

The particles were then given a shape using a lattice. A lattice function deforms the particles (target objects) based on how we reshape the control geometry (profile curve). It calculates the difference between the original curve and the modified curve and determines the deformations to apply to the particles. This results in the particles

being pulled along the modified curve while still having a swirling motion under the influence of forces.

5. Tornado Surface

The surface of the funnel in real tornadoes is rough and its diameter varies along its length. Due to similar forces acting on particles, the surface of the funnel in the simulation is smooth which may not be desired. In order to make the surface irregular, the particles were jittered by function of velocities. Any jitter function can be used as long as it doesn't completely change the trajectory of the particles (Figure 35).

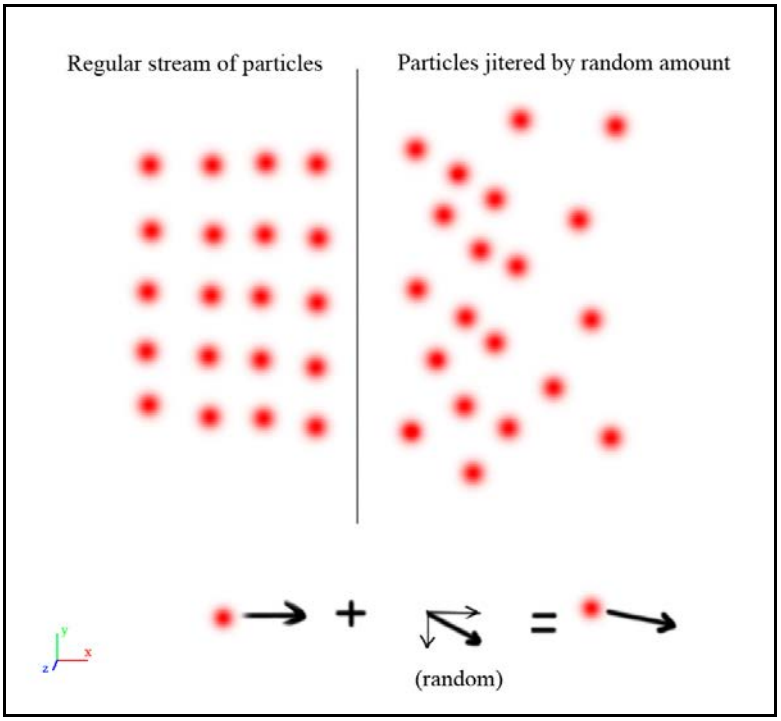


Figure 35: Particles with no jitter and with jitter.

To vary the diameter of the funnel at each control vertex of the curve, another lattice function was used. The initial profile curve was revolved into a cylindrical surface. The radius of the revolved surface used in the implementation was 0.7. This surface was used as the source geometry for the lattice. The modified curve was also revolved. The difference in the surfaces was calculated using the lattice function and deformations were applied to the particles along the modified surface. As a result, the funnel's diameter can be varied along its length.

To give the animators control over how strictly they want the particles to follow the changes defined by lattice functions, a constant was multiplied to the result of this function. The user can modify this constant to achieve different results.

6. Debris Emitter & Forces

To model the debris at the bottom of the funnel, a separate particle system was used. These particles in contrast to the funnel particles have mass so they can be influenced by gravity force. The vertices of torus shaped geometry were used as emitters. The forces acting on these particles were defined within metaball regions. Similar to the funnel, the force fields defined inside the metaball were determined using a function which was maximum at the centre with an exponential fall-off towards the outer regions.

In real tornadoes, the debris at the bottom of the funnel gets pulled inside the core towards the center. While towards the outer edges of the funnel, the debris having

more density gets thrown and a dust sheath lying outside the funnel forms. To achieve this effect, two metaballs were used in the implementation. One of the metaballs defines the influence region for forces which pulls the particles towards the centre. A directional force was used to draw the particles in a positive Y-direction and a vortex force gave the particles a swirling motion as they were being pulled up.

The forces used in regions defined by the second metaball causes the particles to be pushed away from the funnel. There were three kinds of forces defined in this region. A negative vortex force is responsible for pushing the particle away from the axis as they are swirling around the Y-axis, a directional force in positive Y direction pulls the particles upwards and a counteracting gravity force causes the particles to be drawn towards the ground. As a result of these forces, the particles rotate along Y-axis while being pushed away from the funnel and fall to the ground which gives the particles an inverted cone shaped appearance which reflects visual features as seen in real tornadoes. In Figure 36 particles colored red are the debris particles.

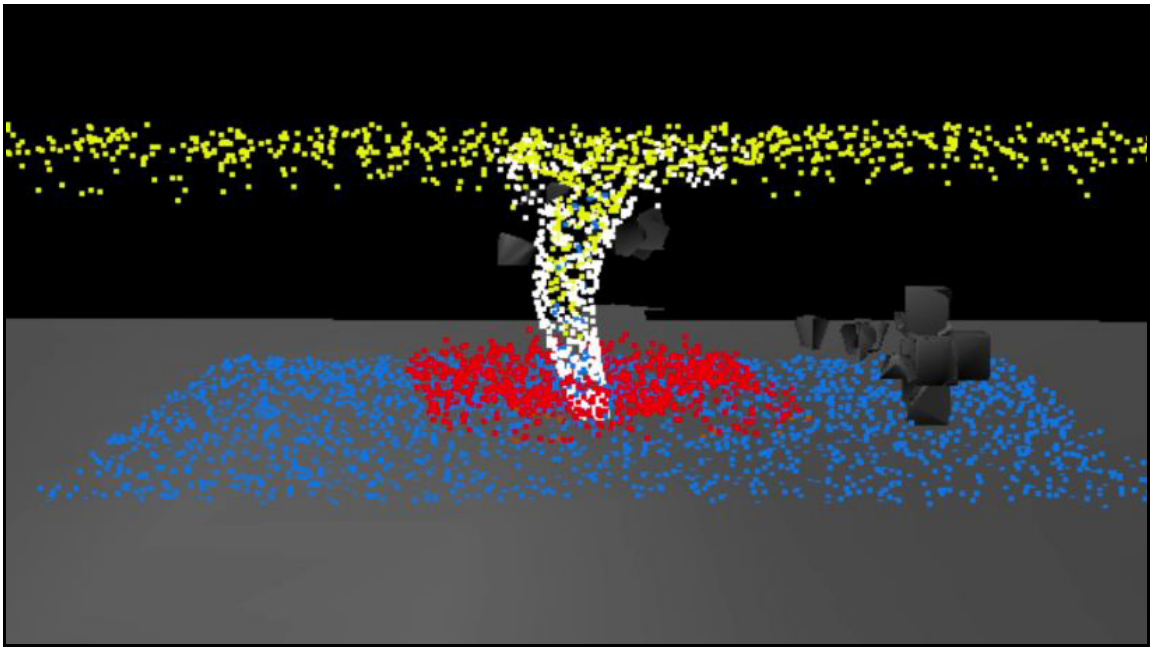


Figure 36: Colors represent different particle systems. Each of the elements, clouds, funnel, debris being picked up and dust on the ground are simulated with different particle systems under different force fields.

As noticed in the tornado references, the debris begins to be picked up by the funnel when the vortex forces are strong enough to influence them. Animators are provided control over this timing to specify when the debris particles should appear. To achieve this, a condition was specified to start emitting debris particles only after a certain frame number had reached. Animators can also key frame this parameter similar to other parameters.

III.4.Fluids

1. Scalar Fields

Particle simulations from previous steps were used to drive the fluids. As a first step to direct the fluids, particles were used as fluid sources. A three dimensional fluid container (3D voxel grid cells) was created large enough to contain the particles. Each particle was given a scalar attribute density and a radius of influence which determined the number of neighboring cells a particle would influence.

2. Particles as Sources

The density attribute of particles were used as sources of smoke. Each particle density value was distributed among the cells located within its radius of influence. The distribution function varies with the distance of the cell from the particle based on an interpolation scheme. A trilinear interpolation was used in this thesis to achieve the results. A weighted distribution is necessary to ensure a smooth fall-out of density with a gradient (Figure 37).

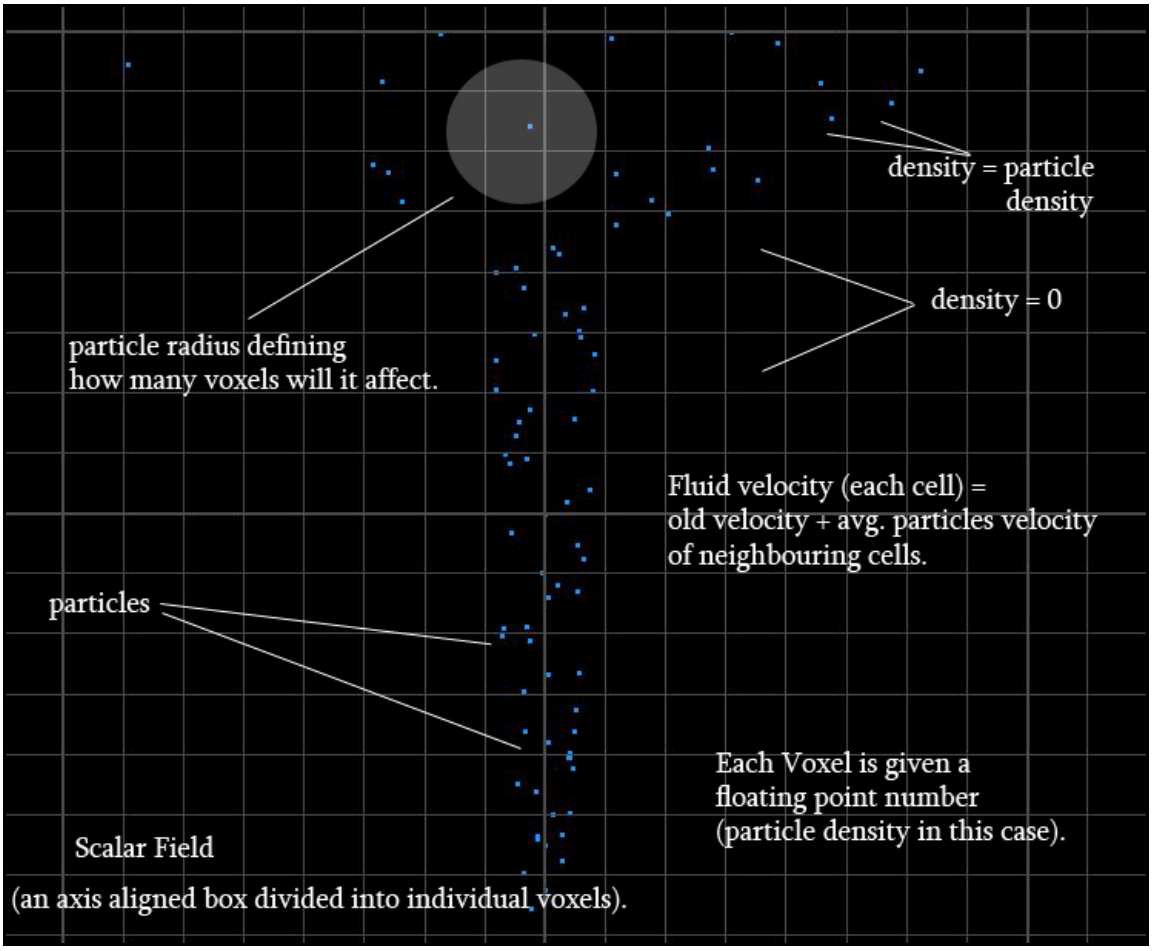


Figure 37: Particles being used as fluid sources. A particles velocity gets transferred to voxel cells.

In order to advect the density of the fluid along with the particles, an interpolation of velocities from neighboring cells was added to each cell. The interpolation ensures a smooth transition of velocities. Visually this gives an appearance of fluid following the particles.

Since the particles are constantly moving while being treated as fluid sources, there can be a case where a voxel has density source in one time step while in the next time-step there is none since the particle is no longer near that voxel. This would lead to

abrupt transition of densities if less number of particles were used in the simulation. To avoid such situation, densities needed to be scaled with time. Scaling of these densities was achieved by using a decaying factor over time. The results achieved in this paper utilize an exponential decay of densities from previous time steps (Figure 38).

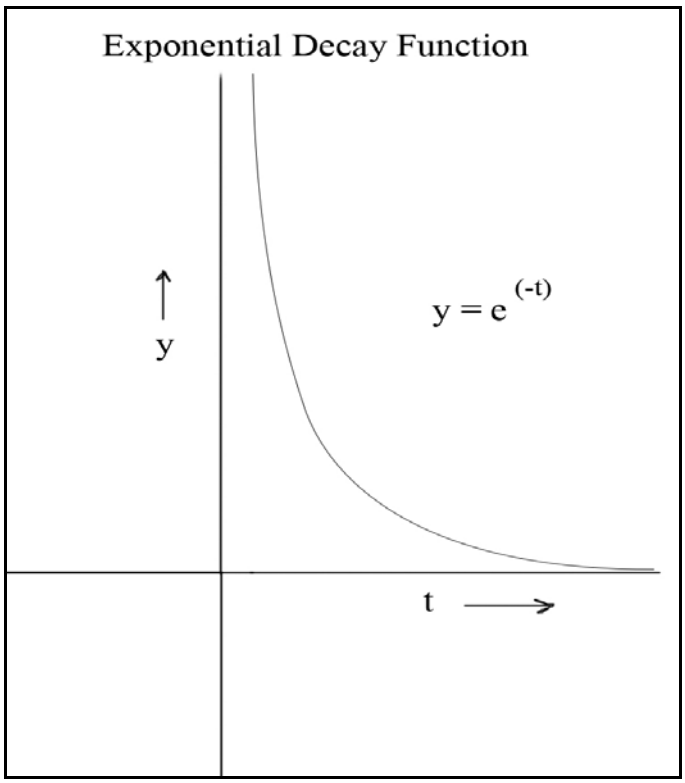


Figure 38: Graph of exponential decay function.

The equation used in the implementation to decay densities is:

$$d_t = d_0 * e^{A(-t)}. \quad (2)$$

where: 't' is the current time-step,

'd_t' is the density at time-step 't'.

'*A*' is a decay constant.

This value is added to the current density calculations which results in particles leaving a trail of smoke as they are moving through the voxel grid. Figure 39 shows a trail left by a single particle as it moves under the influence of forces.

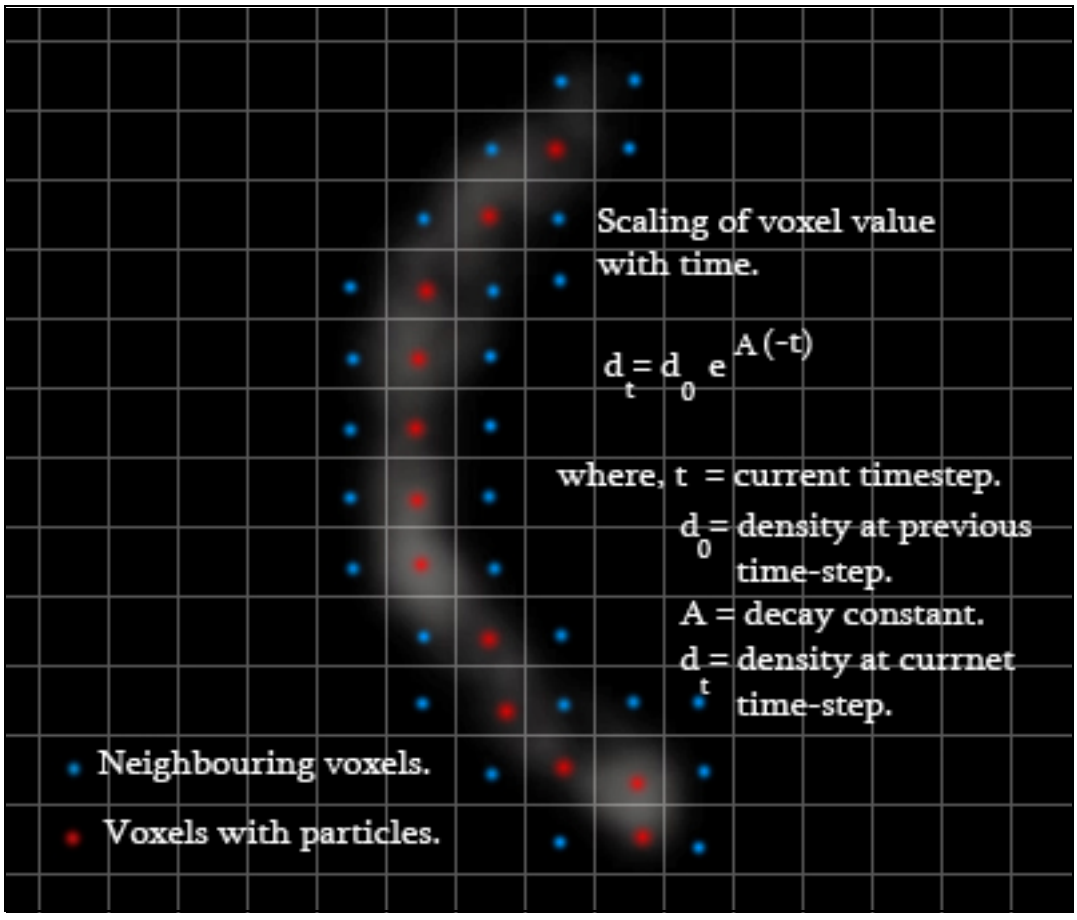


Figure 39: A single particle leaving a trace of fluids. A voxel's density value decays with simulation time.

The general procedure for one time-step is as follows:

Each voxel cell has temporary variables $tempDensity_i$ and $tempVel_i$ to store density and velocities. The variable olD_i stores decay value of density at each cell.

Clear these variables at the start of time-step to zero.

1. Particle Simulation:

Loop through number of particles. For each particle:

i. Compute forces (acceleration).
ii. Integrate acceleration to compute new states.
iii. Update states.
iv. Check which grid cells fall within radius of influence of the particle.
v. Calculate interpolated density value for each cell which falls within particle's influence radius and add this value to the temporary variable $tempDensity_i$ of these cells.
vi. Add particle's current velocity to the $tempVel_i$ variable of the cell in which the particle is located.

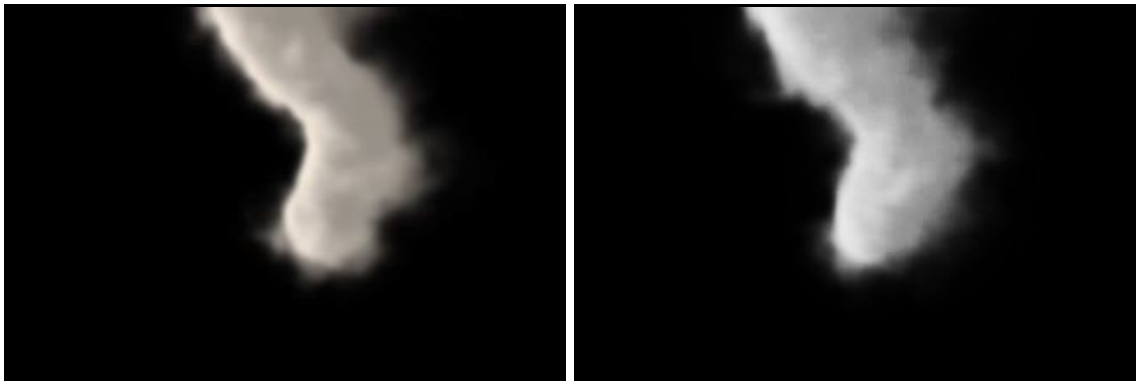
2. Fluid Simulation:

Loop though grid cells. For each cell:

i. Calculate decay value of current density in the cell using equation (2) and store this value in variable $oldD_i$.
ii. Add $oldD_i$ to $tempDensity_i$ from previous step and store this value as the current density.
iii. Calculate a weighted average of $tempVel_i$ from neighboring cells using an interpolation scheme and add this value to the current velocity at the cell.
iv. Calculate vorticity forces and other forces.
v. Perform fluid simulation step: advect - > diffuse -> project.

The results obtained for the funnel part using this technique are shown in Figure 40. The figure also shows comparison of results achieved by varying "vorticity factor". The "vorticity factor" in these figures is a constant multiplied to the vorticity confinement force. Increasing this parameter makes the shape of the funnel fuzzier due

to more turbulence in the smoke, while decreasing this value gives it a well defined shape. Similarly, the "decay factor" is decay constant as defined in equation (2).



(a) (b)

Figure 40: Vorticity factor comparison. (a) Funnel rendered with vorticity factor of 10 and 0.1 decay factor on a 80x80x80 grid. (b) Funnel rendered with vorticity factor of 13 and 0.2 decay factor on a 80x80x80 grid.

Figure 41 below is the outcome of sourcing fluids from debris particles at the bottom of the funnel. This technique was also used to simulate clouds and dust on the ground being picked up in the vortex.

Figure 41: Result obtained for debris particles using fluids. Debris rendered with 20 vorticity factor and 0.5 decay on a 80x80x80 grid.

CHAPTER IV

IMPLEMENTATION

IV.1. Side Effects Houdini

The implementation was done in Houdini [SIDE EFFECTS SOFTWARE Inc. 2011] which is a 3D animation software. Houdini is a complete tightly integrated and fully customizable 3D solution. Its procedural system allows an algorithmic approach to the problem. Other advantages of using Houdini over other available 3D packages are that it integrates well with the current tools and its node based approach helps focus on the creative side without worrying too much about the technology. Digital assets can be easily created which can then be distributed to other artists. The node based system gives a wide range of control and flexibility over the production pipeline. Furthermore, Houdini is a popular package in the visual effects industry for creating effects shots.

IV.2. Houdini Digital Asset

The tool was built using Houdini's Digital Asset. Digital Asset is an encapsulation of a series of operations of nodes in Houdini network into a single node. It serves as a reusable tool with a customized interface which can be easily handed to artists. By reducing several nodes into a single node, the number of clicks to achieve a task is reduced. It is an ideal tool for effects shots which tend to be driven more by

technical directors. It also allows them to create complex systems easily which can be condensed to a single digital asset. This asset is then handed over to the animators who are concerned with adjusting various parameters for timing and placements without worrying about its implementation.

Several nodes were used to implement the method proposed in this thesis. Explanation of commonly used nodes is as follows:

i. SOP (Surface Operators):

This node controls the shape of the geometry. It encapsulates various other nodes including the "Particle Operator" (explained below), "Curve" and "Lattice" nodes (explained below).

ii. POP (Particle Operators):

"Particle Operator" node defines particle system and operations on them. It encapsulates a "Source" node which is used to source particles from any object. When used with an "Attractor" node (used to attract or repel particles towards/away from a force defined by metaballs) and "Drag" node (which applies friction force to the particles), it can be very useful.

iii. DOP (Dynamic Operators):

It encapsulates various nodes which set up the conditions and rules for dynamics simulations such as rigid body, fracturing geometry, smoke simulation and fluid simulation.

iv. Point:

Each geometry has a list of vertices with an x, y and z position. The "Point" node is used to edit the attributes of these vertices including their positions [SIDE EFFECTS SOFTWARE Inc. 2011].

IV.3. Simulation

1. Particles

Implementation of the method started with a "Curve" node with seven vertex points. This curve serves as a skeleton which controls the shape and diameter of tornado at different vertices (Figure 42).

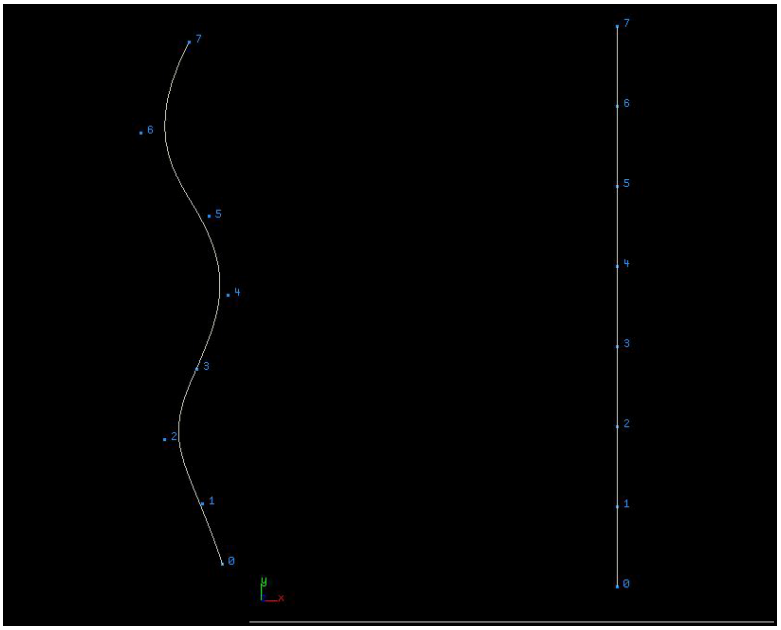


Figure 42: Profile curve in Houdini.

A "Circle" node was then created and its vertices were specified as sources of particles. To emit particles non-uniformly, a "Point" node was appended to this "Circle" node and using a random function of space & time, the vertices of the circle were jittered (Figure 43). A "Particle Operator" (POP) node was used to emit particles. Two "Metaball" nodes were then used to define influence region for forces to act on these particles. Metaball at the top region of the curve modeled the wall cloud while the other was used to simulate the funnel (Figure 44). A "Force" node was added to these "Metaball" nodes to add swirling motion to the particles and to apply a directional force towards negative Y-axis.

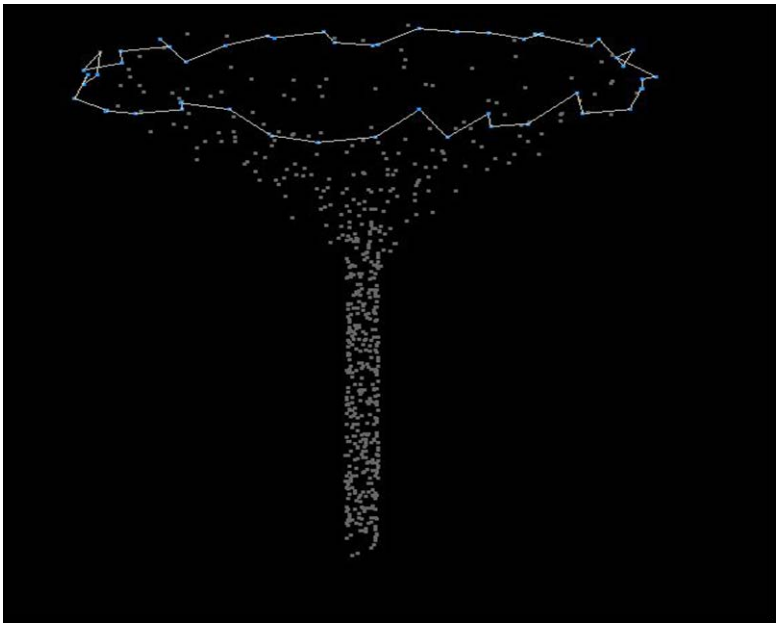


Figure 43: Particles being emitted from a circular emitter. The emitter is randomized in time and space to get a non-uniform emission of particles.

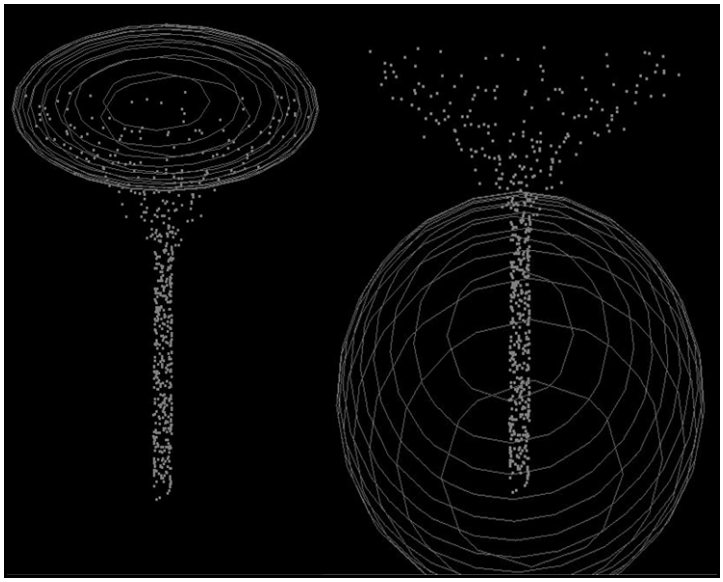


Figure 44: Metaball force fields. (Left) Metaball field at the top region to model wall clouds. (Right) Metaball placed at a lower height to shape the particles in a funnel shape.

To draw the particles towards the curve, a "Lattice" node was used with curve as the source object and particles as the target geometry (Figure 45).

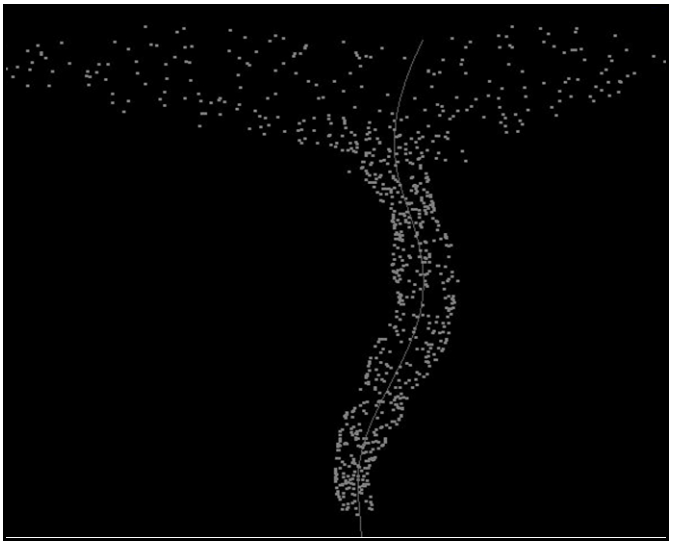
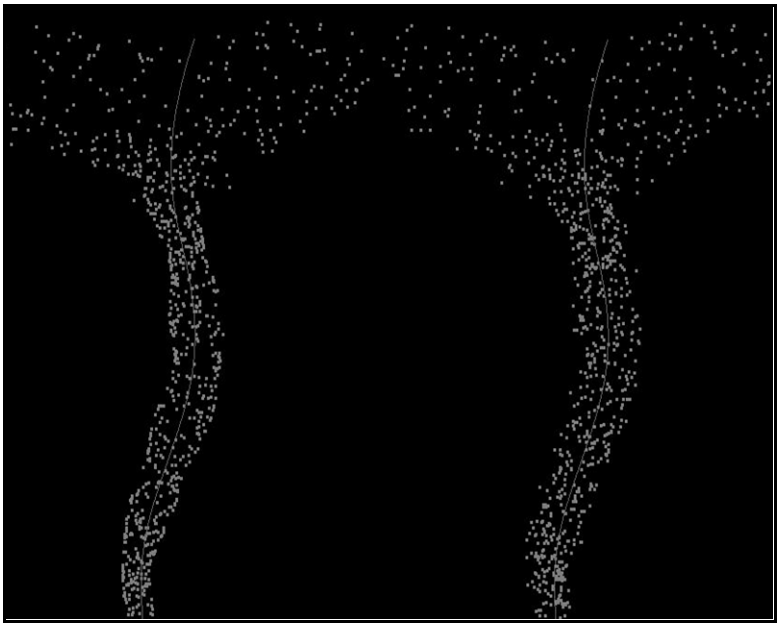


Figure 45: Particles following the profile curve using "Lattice" node.

In order to make the surface of the funnel rough, a "Velocity" node was added to the POP node to jitter the particles based on a function of current velocity (Figure 46).



(a) (b)

Figure 46: Particles jittered using "Velocity" node. (a) Particles forming a regular surface under the influence of similar forces. (b) Particles are jittered from their paths to form an irregular and more turbulent surface.

The give the funnel varying thickness along its length, another "Lattice" node was used. The initial curve and modified curve were revolved with "Revolve" nodes and then fed to the "Lattice" node. The output of this "Lattice" node gives a shape to the particles based on the deformations applied to the revolved surface (Figure 47 and Figure 48).

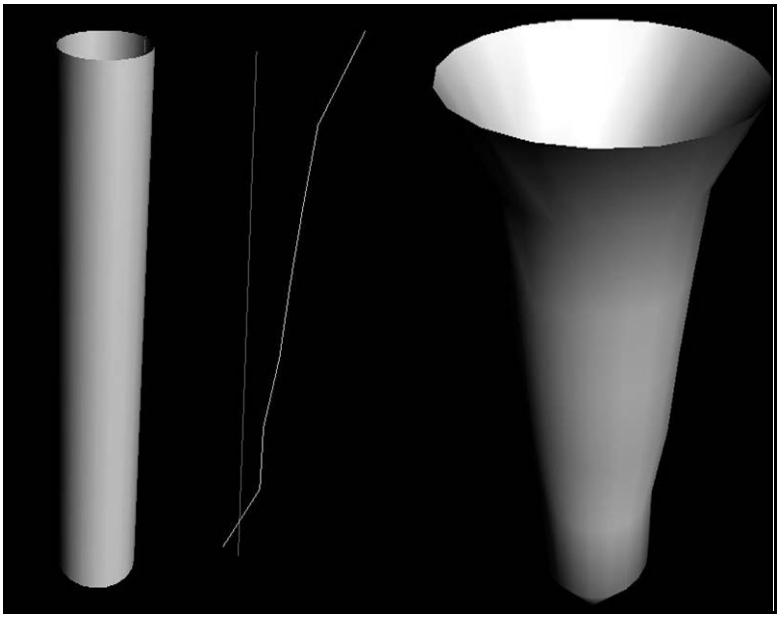


Figure 47: Revolved surfaces. (Left) Revolved surface from original curve. (Right) Revolved surface from modified profile curve.

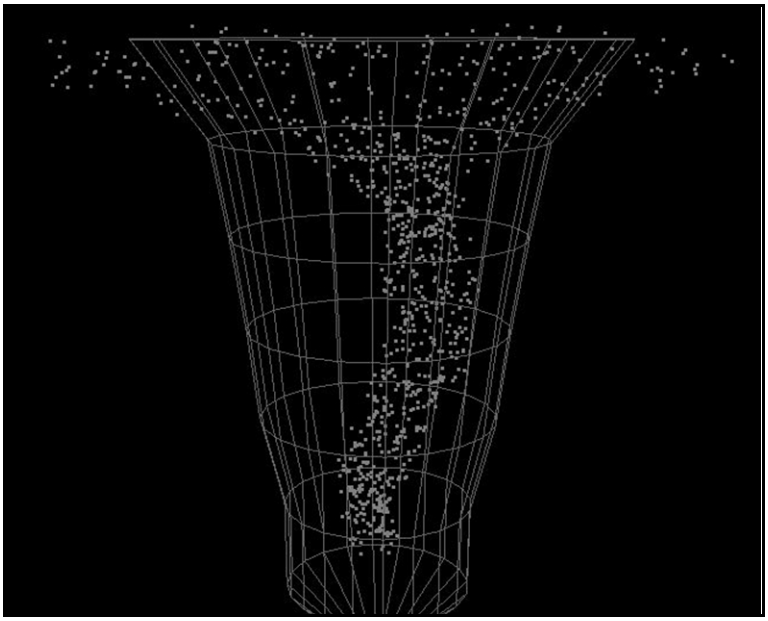


Figure 48: Particles following the modified surface.

The debris particles were created using a "Particle Operator" (POP) node and torus geometry was specified as emitter in the "Source" node inside this POP node. To simulate the particles being drawn towards the center, a "Metaball" node with "Force" node appended to it was used. In this "Force" node, a directional force in positive Y axis and a vortex force along Y-axis were specified. To model debris being thrown around the funnel, a second "Metaball" node was used. A negative vortex force and directional force in positive Y direction was specified in "Force" node attached to this "Metaball" node. To make the particles affect by the gravity force, a "mass" property was attached to these particles using a "Property" node. The particles were then pulled towards the ground using a "Gravity" node (Figure 49). Both the funnel and debris particles were assigned a density scalar attribute using an "Attribute" node attached to their POP networks. This attribute was later used as source of smoke in fluid simulation step. The entire particle network was contained inside a single "Surface Operator" node.

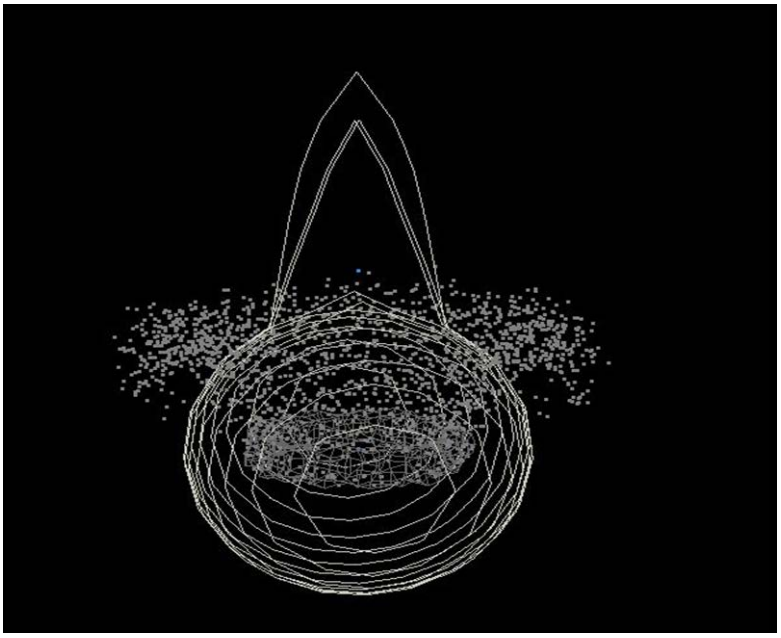


Figure 49: Debris simulation at the bottom of the tornado. Particles are acted upon by vortex and directional forces within metaball regions.

2. Smoke

For the smoke simulation, a network of nodes was built inside the "Dynamic Operator" node. Important nodes and their functions are explained as follows:

i. Smoke Object:

This node creates a smoke container object where voxel grid divisions and positions can be defined [SIDE EFFECTS SOFTWARE Inc. 2011].

ii. Gas Particle To Field:

It is a micro-solver used to copy particle systems attributes into a field with modes to set how many voxels will each particle influence. It uses a trilinear

interpolation to distribute particles values to each voxel depending upon how close they are to the center of the voxel [SIDE EFFECTS SOFTWARE Inc. 2011].

iii. Gas Calculate:

The function of this solver is to perform post-operation or pre-operation on source or destination fields [SIDE EFFECTS SOFTWARE Inc. 2011].

iv. Smoke Solver:

This node is used to configure smoke solver and provides access to fluid parameters such as viscosity, buoyancy and vorticity. It can also be used to inherit point velocities of another object [SIDE EFFECTS SOFTWARE Inc. 2011].

v. Source Relationship:

This node is used to specify another simulation object as smoke source [SIDE EFFECTS SOFTWARE Inc. 2011].

vi. Gas Up Res Object:

This is used to create a high resolution container object using low resolution fluid simulation [SIDE EFFECTS SOFTWARE Inc. 2011].

vii. Gas Up Res:

This node is responsible for up scaling smoke simulation to a higher resolution by adding more details in the form of turbulence [SIDE EFFECTS SOFTWARE Inc. 2011].

The fluid simulation part of the network was responsible for using particle's density attribute as fluid source and adding particle's velocity to voxel cells. A fluid container was created using "Smoke Object" node and its size was specified large enough to contain the entire particle system. The "Smoke Solver" node was then attached to this node. "Smoke Solver" is a solver for smoke simulation with various parameters built in such as vorticity, buoyancy and viscosity. This node also has an option to "use point velocity" from smoke source meaning to inherit the particles velocity into each voxel.

To import the particles from "Surface Operators", a "SOP Geometry" node was utilized. To store the density attribute of these particles to each voxel, a scalar field was created using "Scalar Field" node and it was given the same dimensions as the fluid container. Particles attribute was then populated in this field using "Gas Particle To Field" node. This node also has an option to choose how many neighboring voxel cells to use when distributing densities from particles and it uses trilinear interpolation to do so. Particles densities were later specified as sources to be used for smoke container using a "Source Relationship" node.

To scale the density of the smoke with simulation steps or in other words for the decaying of the smoke with time, a "Gas Calculate" node was used. In this node a post-processing operation on the smoke results was done which was responsible for decaying the density of smoke at each voxel over time. A layout of the network of nodes explained above is shown in Figure 50.

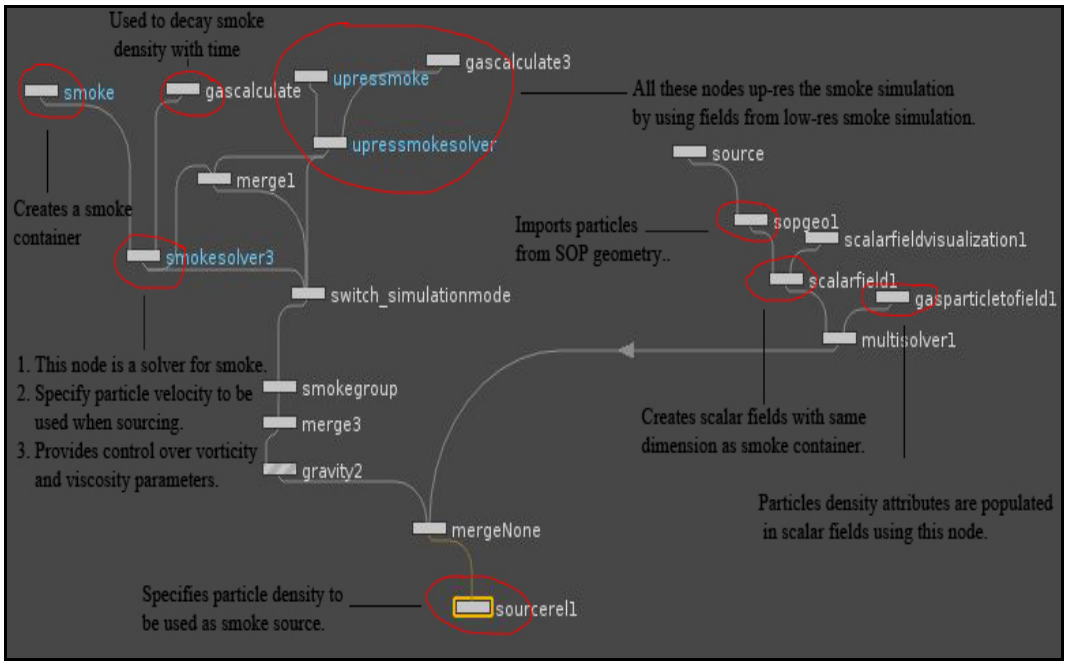


Figure 50: DOP network used to drive smoke using particles.

A simulation in a production environment may require many iterations and it may not be desirable to first wait for the fluid to simulate and later modify the results. In order to prevent such situation, a low resolution fluid container can be used to modify the results and after a particular look is achieved; a high resolution fluid simulation can be run based on the velocity and other fields from this low resolution fluid simulation. Houdini has a very easy solution to this problem and provides nodes such as "Gas Up Res Object" and "Gas Up Res". "Gas Upres Object" uses the low resolution fluid simulation and creates a high resolution version of it. Similarly, "Gas Up Res" node up-scales smoke simulation to a higher resolution by adding details in the form of turbulent

noise. In the implementation, these nodes were utilized to upscale the results from low resolution fluid simulation. This helped in achieving the results comparatively faster.

3. Lighting, Rendering & Shading

Figure 51: Final render of the tornado scene. Houdini's Mantra, version 3.1 [SIDE EFFECTS SOFTWARE Inc 2011] was used to render the scene. A default "Billowy Smoke" shader was used to shade the fluids.

A tornado scene was created using the tool (Figure 51). Lighting, shading and rendering of the tornado scene was done in Houdini. A default "Billowy Smoke" shader was used to shade the fluids. Lighting uses a three point lighting system on each of the elements, funnel, debris and the clouds. Three point lighting system as used in this thesis consists of a key light, a fill light and a rim light. A key light is the main source of light

that illuminates two third of the object being lit, it casts shadows and has specular emission. A fill light is comparatively low intensity light source that fills in the shadow areas from the key light and has no specular contribution. It also casts soft shadows with low intensities on the object. The rim light defines clear edges of the object and is placed opposite to the camera to illuminate the subject from behind. It is a non-shadowing light with no specular emission. The rest of the scene uses one key light which is basically a spot light with an exponential fall off. Houdini's Mantra renderer was used to render the scene. The cloud system and the dust on the ground use the same particle based fluids technique as used for the funnel and debris. They also use a "Billowy Smoke" shader for shading.

4. Lightning

Lightning discharge is very common in tornadoes due to discharge of electricity in thunder clouds. The lightning in the scene was simulated by animating the intensity of light. The animation curve of the light is shown in Figure 52.

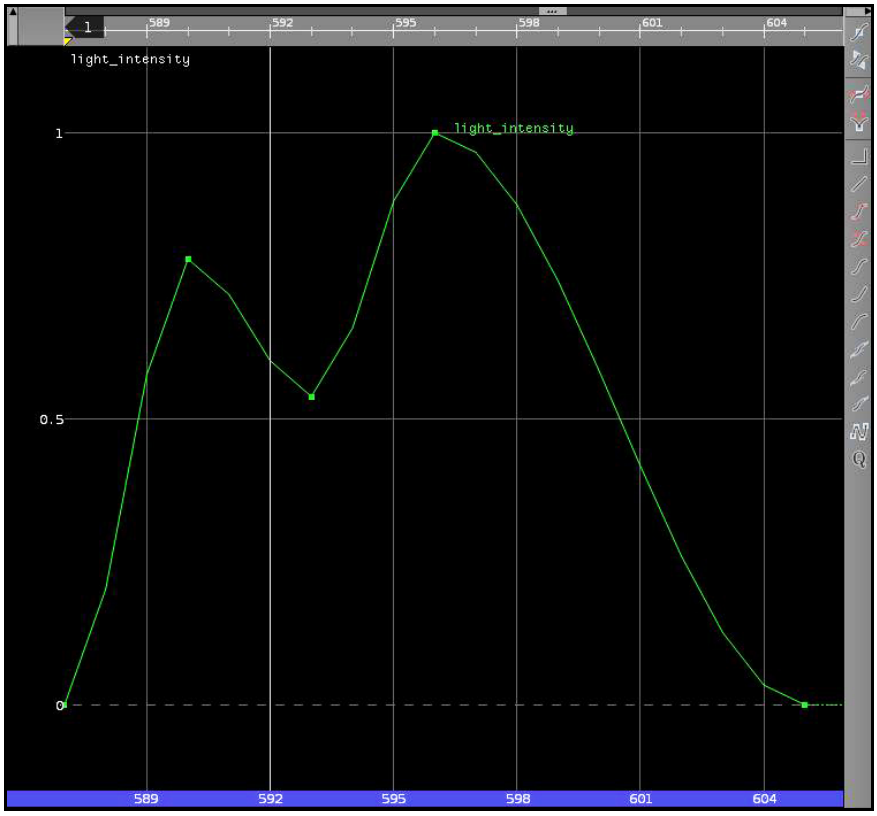


Figure 52: Light intensity curve used to simulate lightning in the scene.

IV.4. Fracturing and Debris

1. Seed Points

To add to the intensity of the tornado scene, fracturing of geometry was also incorporated. This section discusses the method used for fracturing. As observed in reference footages, the debris pieces from destruction gets picked by tornado and starts to swirl around the funnel while some of it gets scattered. Since particle system forms an integral part of the method in this paper, hence they were used to simulate fracturing too. It also provides a lot of control over the movement of fractured pieces.

To simulate the destruction, pre-fractured geometric models were used. Fracturing can be done using a number of methods. For the purposes of the implementation in this paper, fracturing based on Voronoi [SIDE EFFECTS SOFTWARE 2010] seeds worked fine. Voronoi fracturing method takes points within the geometry as inputs and based on the location of points it fractures the geometry.

A "Voronoi Fracture" node was used in Houdini to fracture the input geometry. Apart from the geometry to fracture, it takes number of seed points as its input [SIDE EFFECTS SOFTWARE 2010]. To specify the seed points, an iso-offset volumetric representation of the geometry was created using an "IsoOffset" node (Figure 53).

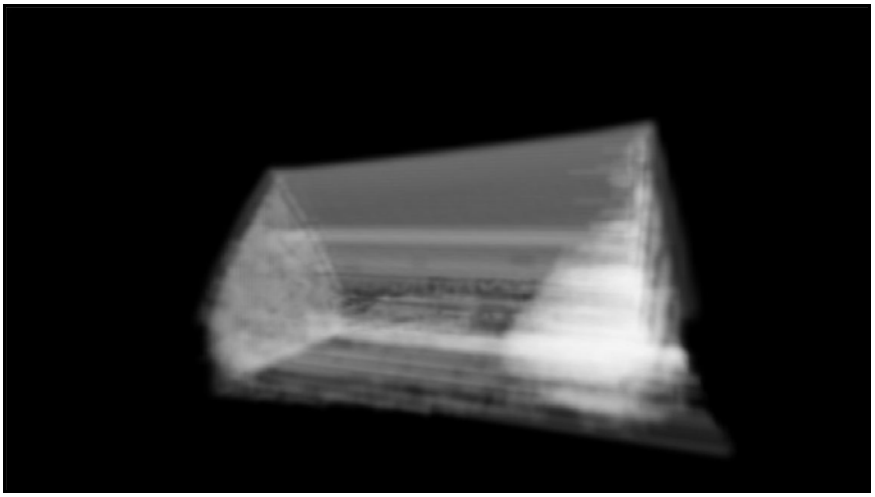


Figure 53: Iso-offset representation of 3D model.

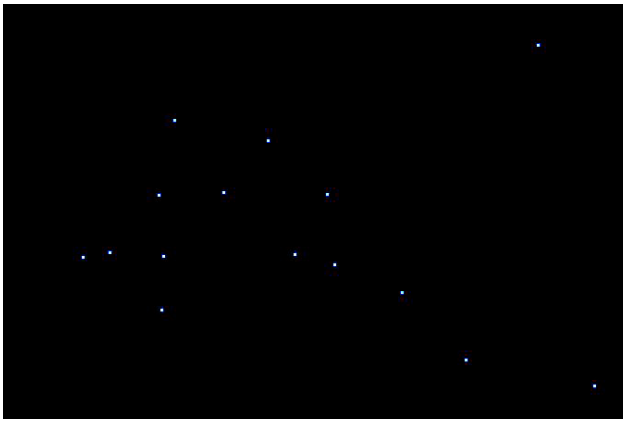


Figure 54: Points scattered uniformly within the iso-offset representation.

Then a number of points were scattered within this volume using a "Scatter" node (Figure 54). The number and distribution of points to scatter depends upon how much detailed the geometry needs to be fractured. If finer details are needed in some parts of the geometry, density of scattered points should be more in those regions (Figure 55).

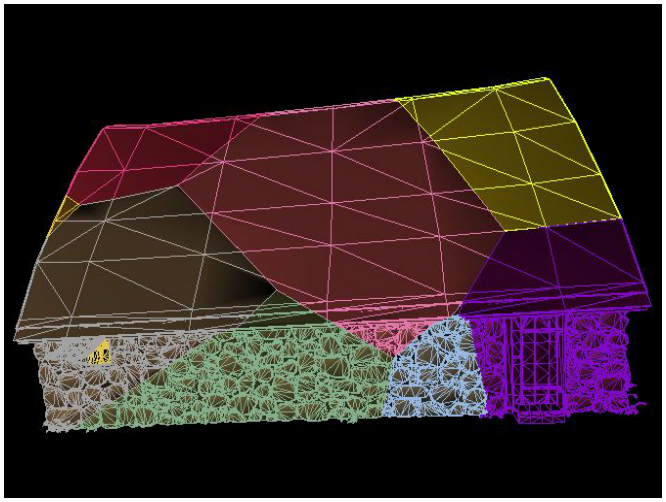


Figure 55: 3D model being fractured by "Voronoi" node.

2. Particle Driven

Once the geometry was fractured into different pieces, centre of mass (centroid) of each piece was determined using a "ForEach" node. This node separates the input geometry into different groups. Inside "ForEach" node, an expression to calculate the centroid of each piece was specified using an "Add" node (Figure 56). An "Add" node creates points based on custom rules that are specified. A "centroid" function was used as a rule inside this "Add" node which returns centroid information for a "Surface Operator" node.

A "Particle Operator" node was then attached to birth particles from the center of mass calculated above (Figure 57). The individual pieces of debris were then attached to these particles.

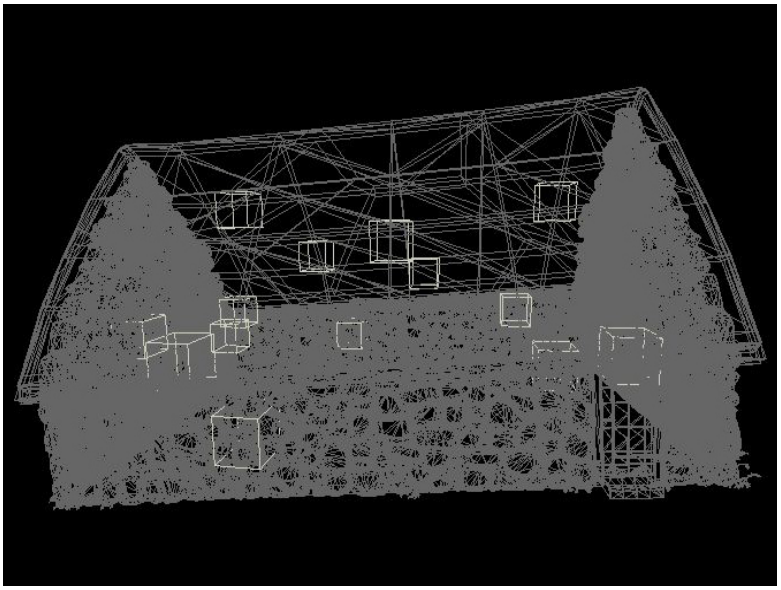


Figure 56: Centre of mass of individual pieces represented by boxes.

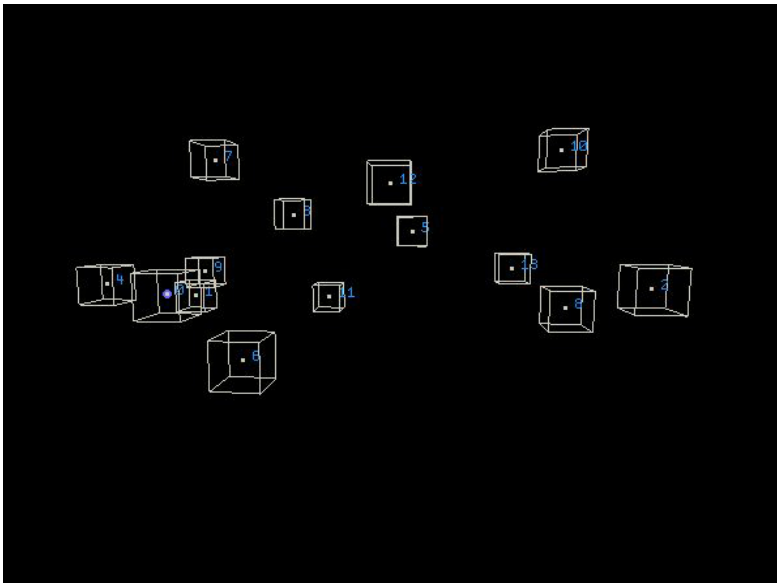


Figure 57: Particles attached to the centre of mass of each piece.

These particles were then acted upon by forces used for the funnel part of the tornado. As a result, the particles were drawn vertically by directional force in positive Y axis and vortex force made them rotate around the funnel. Since pieces of fractured geometry were glued to these particles, the pieces also moved with them. These pieces were also given rotations based on a random function (Figure 58).

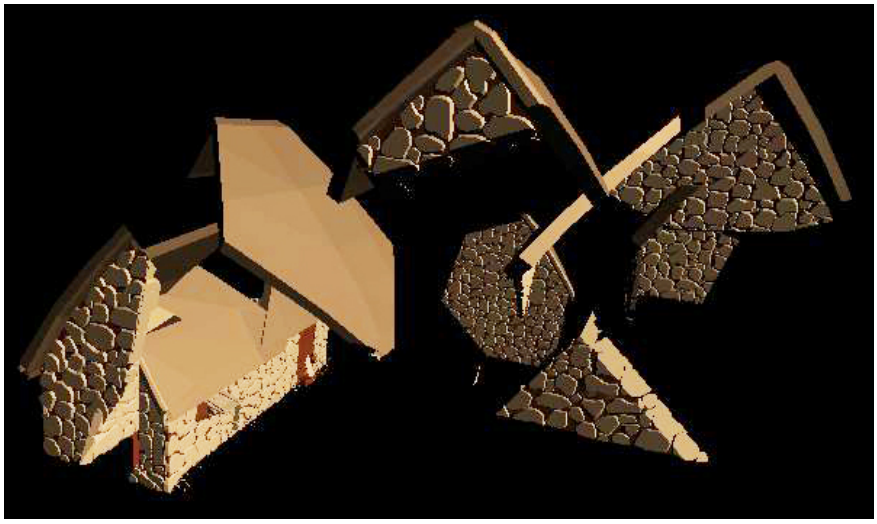


Figure 58: Fractured pieces attached to the particles. These pieces were given random rotations as they were being pulled by tornado forces.

By using this technique, fracturing that happens in a tornado environment was achieved efficiently. The advantage of this technique is that the forces acting on the particles can be changed as desired to achieve different intensities of destruction.

This fracturing model is not physically accurate but it gives believable results when compared to visual references. Although Voronoi fracturing was used for fracturing in the implementation, other methods to fracture geometry will serve the purpose equally.

IV.5. Tool

1. Python Scripting

Houdini provides a very powerful and easy to use object oriented language Python integrated with it. Python can be used to script Houdini using objects and

functions which are automatically imported by an embedded interpreter. Once the tool was created, Python scripting was helpful in tool building. Its robust tool building interface allowed creation and placement of various parameters such as buttons, floats, toggle etc. very easily.

CHAPTER V

EVALUATION

V.1. Results

1. Qualitative Study

For evaluation purposes, the tool was handed over to seven participants resulting in several simulations. The selection process involved picking participants among group of students from the Department of Visualization at Texas A&M University who had some experience in Houdini. The recruitment process involved sending emails to the participants individually. This email elaborated on the purpose and the goals for the study and participation was entirely voluntary. The participants were not given any direct benefits except a chance to explore the tool.

In the evaluation process, a task was given to the user which was to create tornadoes of their choice. A training session was given at the start of the process which was divided into two parts, particle simulation and fluid simulation. Participants were first trained on particle simulation parameters which lasted about 10 minutes. Then they were allowed to explore the tool on their own and create a tornado. After they completed the particle simulation, another 5-10 minutes of training was given on fluid parameters before they could start exploring the fluid part of the tool. Help was available to the participant while working on the tool in the form of tooltips for each parameter.

After the evaluation was completed, the participants were given a questionnaire. The questionnaire was based on the participant's experience with the tool and their suggestions to improve the tool. They were first asked about their experience using 3D animation tools and Houdini in particular. This helped determine their expertise and comfort level in using the tool. When asked about how user friendly they found the tool to be, 6 out of 7 participants responded that they found it "easy" to use while 1 considered it "fairly user friendly". The response of one participant was,

> "I found it quite friendly. It was pretty easy to use and understand, after the brief demo you gave me."

On being asked how helpful the tooltips were, the participants responded by saying that although help was useful they didn't use it much while working on the tool. The time duration that participants took to achieve the desired look of tornado was approximately 20-30 minutes for particle simulation and 30-40 minutes for fluid simulation. This included the training sessions.

Most of the participants found the naming convention of the tool intuitive except for 1-2 parameters that could be clearer. All the 7 participants agreed that they were able to create the tornado they wanted. To test the usefulness of the tool in a production environment, they were asked whether they believed the tool could be used in a professional facility. Most of them thought that after making a few changes in the interface it could be used successfully. All the participants agreed that there were no redundant features in the tool. As far as the features they thought could be added, one of them replied,

> "A short instructional or "Help" button should be added, that would give you the basic steps needed to create the tornado simulation."

Another user felt the need to add an automatic rendering feature which would help to pre-visualize the animation frames with basic lighting. 2 participants had trouble locating the buttons to achieve a task which leaves room for improvement in the design of the user interface. Figures 59, 60, 61, 62, 63, 64, 65 and 66 show results from simulations created by participants. Increasing the vorticity parameter makes the shape of the funnel fuzzier due to more turbulence in the smoke, while decreasing this value gives a well defined shape.

Figure 59: Particles: 30; Vorticity: 12; Viscosity: 0.

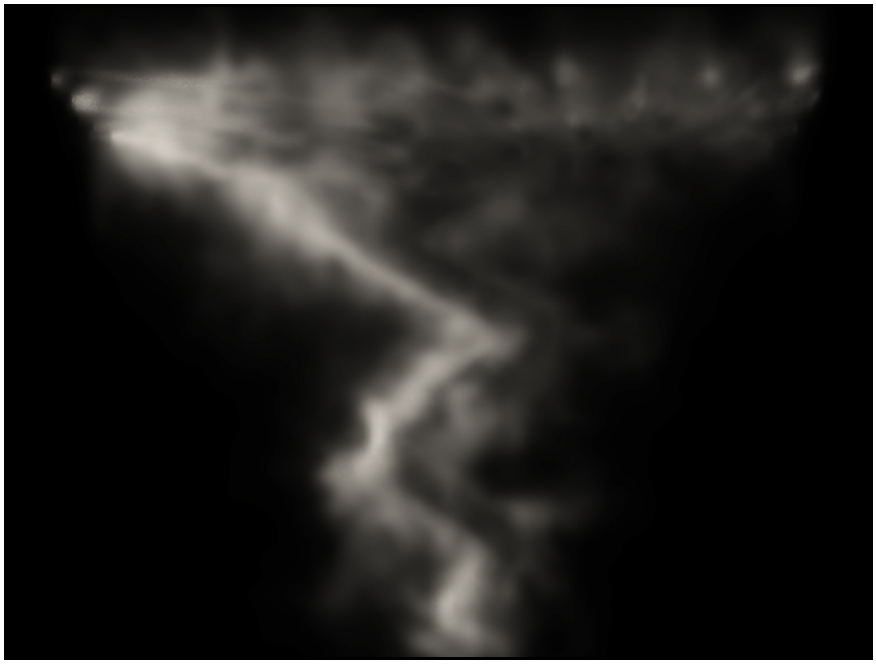


Figure 60: Particles: 30; Vorticity: 13; Viscosity: 0.3.

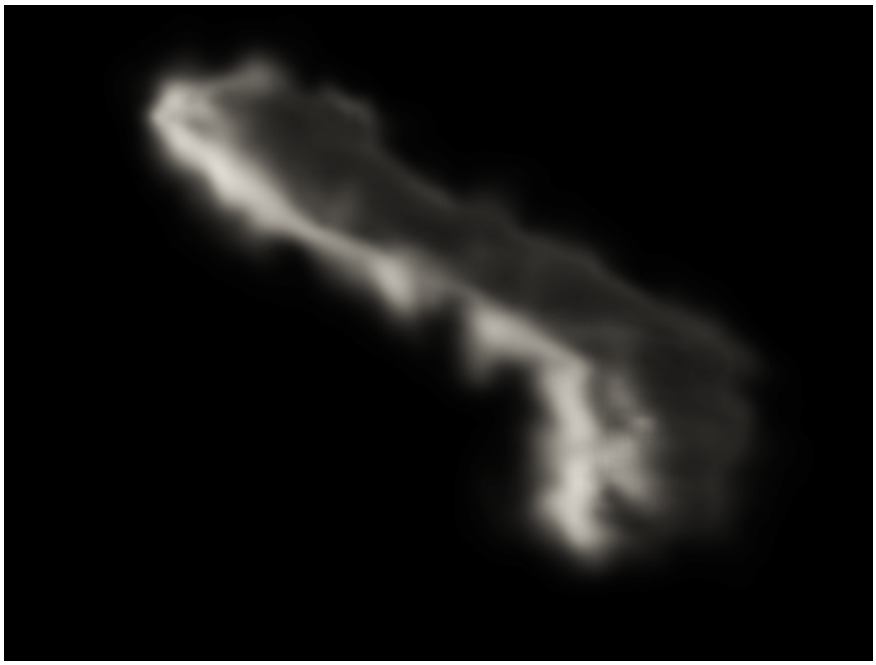


Figure 61: Particles: 30; Vorticity: 5; Viscosity: 3.

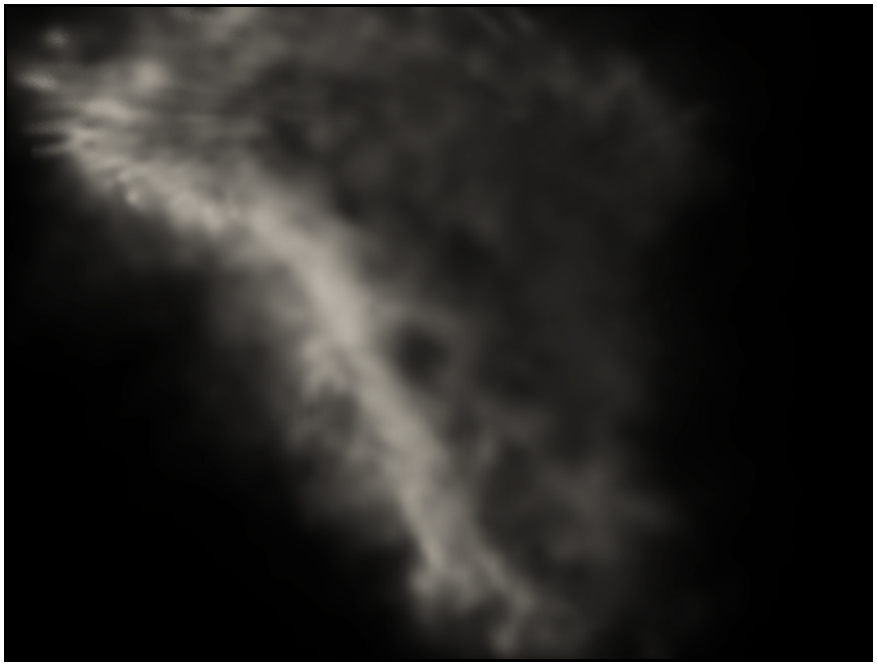


Figure 62: Particles: 60; Vorticity: 30; Viscosity: 0.

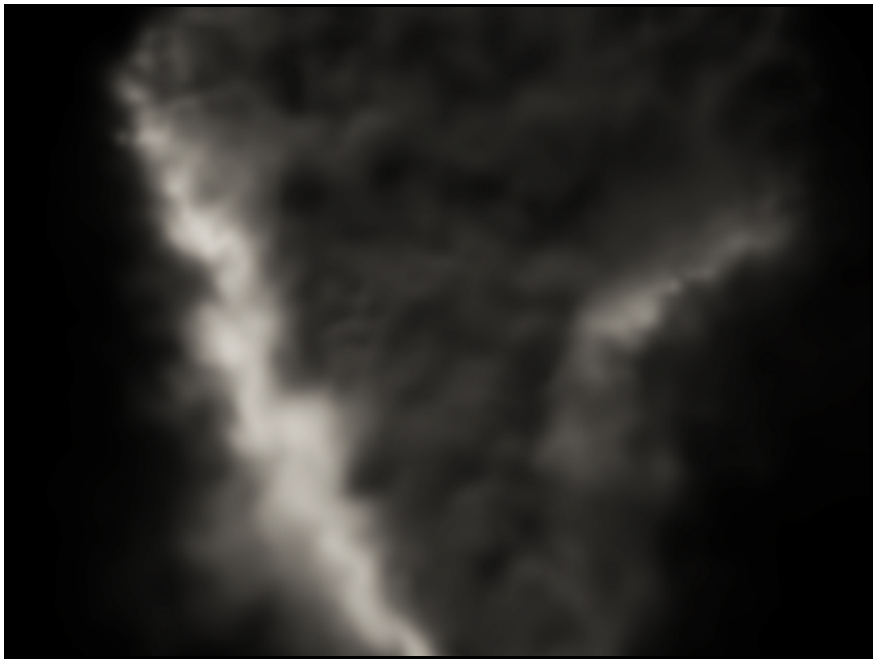


Figure 63: Particles: 200; Vorticity: 15; Viscosity: 0.

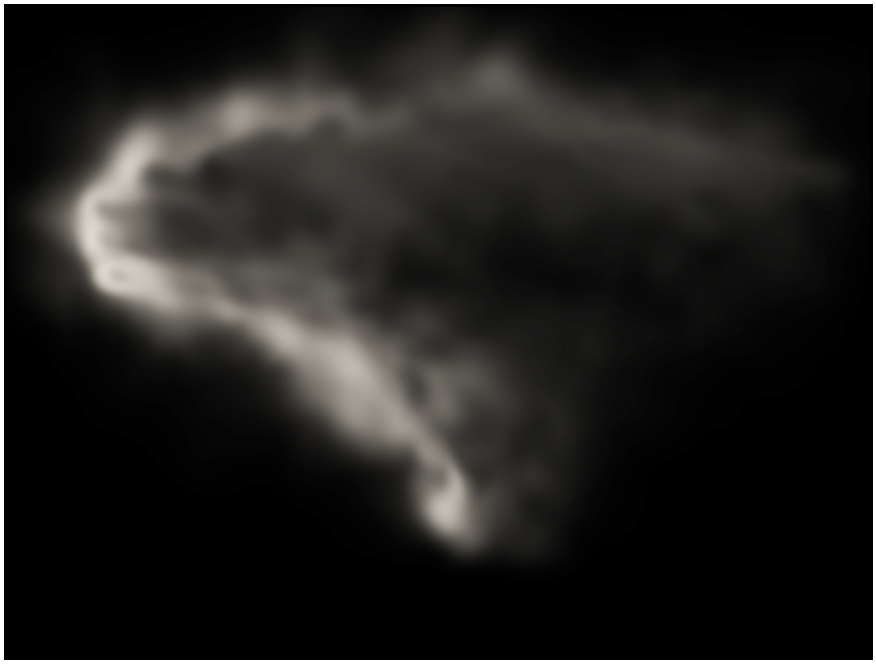


Figure 64: Particles: 100; Vorticity: 8; Viscosity: 0.

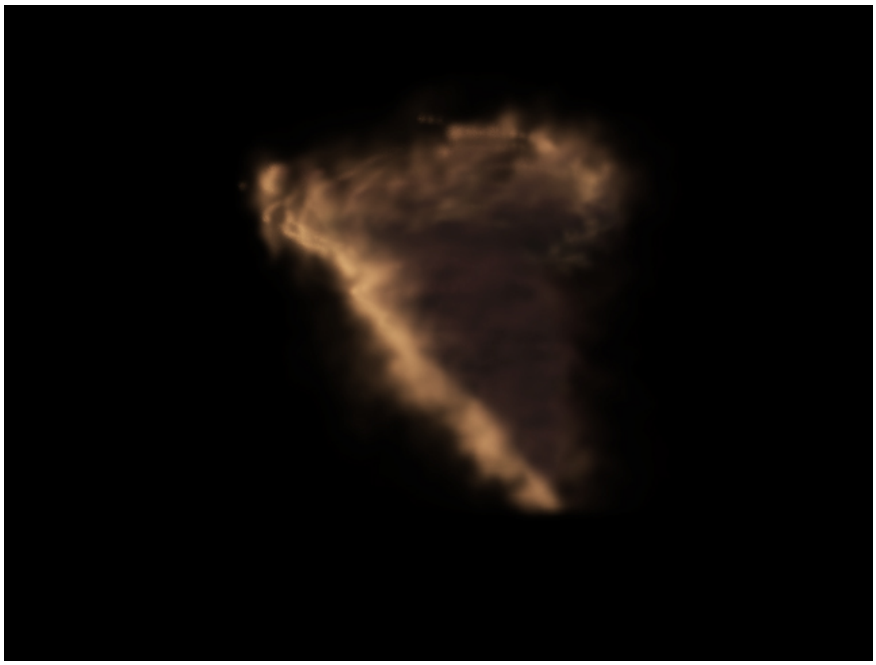


Figure 65: Particles: 30; Vorticity: 15; Viscosity: 2.41.

Figure 66: Particles: 60; Vorticity: 0; Viscosity: 0.

V.2. Limitations of Study

There are limitations in evaluating the results of the questionnaire. The evaluation was performed by a small group of graduate students from the Department of Visualization at Texas A&M University who are skilled at animation and computer graphics applications. It is likely that their expertise enhanced their understanding of the tool. The users might also have different levels of expertise which might have resulted in varied experiences with the tool. In addition, there were no strict deadlines or a production specific shot requirement which makes the usability of the tool for production purposes hard to judge.

The above limitation could have been improved had there been more users to test the tool. Providing production specific needs also would have helped in testing its usefulness in a production facility.

CHAPTER VI

CONCLUSIONS AND FUTURE WORK

VI.1. Summary

The aim of this thesis was to devise a method to simulate tornadoes and create a tool which could be used in a production environment to create an effective tornado shot easily. The goals of this research were met successfully and the results from evaluations are testimony to this. Coupling particles with fluids provided more control over the simulation and the fluid behavior was also retained. Based on the questionnaire, the following conclusions can be derived:

a) Houdini's dynamics operator nodes proved to be very helpful in creating relationship between particles and fluids quickly. There are lots of functionalities within each node that helps create complex systems very easily.

b) There are some limitations in using Houdini nodes, such as the user has to rely on the functionalities embedded in the nodes which puts a limit on a user's flexibility to accomplish a task. To cite an example, Houdini has a limitation that the user can only see the changes made to the fluid container size when they are at the simulation frame which may not be desired and the user may want to change the container size at any frame.

Another limitation in "Surface Operators" when using "Metaball" node is that the user cannot define custom functions which determine the force fields inside them. They are limited to the function models provided by Houdini.

c) The participants found the user-interface of the tool to be less intuitive. The naming conventions of the parameters were not generic enough to help users understand its function.

d) If less number of particles are used in the simulation, chunks of smoke start to appear since fluids are sourced from them. Increasing the number of particles solves this problem but at the same time slows down the simulation. A better approach to this problem could have been to interpolate densities from different particles once the smoke was being emitted from them.

VI.2. Future Work

a) Work can be done on art directing other types of tornadoes, such as land spout, multiple vortex, waterspouts and other vortices that exist in nature like gustnadoes, dust devil and fire whirls.

b) One can also focus on rendering side. Adding light scattering process that occurs in volumetric elements such as clouds and smoke can add more realism to the renders.

c) The method proposed in this thesis was inspired from visual elements and does not take into account pressure and temperature parameters which play an

important role in tornado dynamics. In the future more such factors can be included to simulate physically accurate behaviors while giving the same level of controls on the simulation.

d) There are smaller scale vortex chambers commercially available like tornadoes in a bottle which create rotations and vortex by a simple shake. Inspired by them and with the advent of motion sensor and touch technologies on cellular phones, similar applications can be devised for mobile phones.

e) Integrating image processing in the simulation can allow the animators to be able to input tornado images and create simulations with similar profile curves.

f) The method presented in this thesis accounts for many particles being treated as fluid sources which take a lot of simulation time. In the future, efficient data structures such as octree, quick sort and heap sort can be implemented which would speed up the process considerably.

g) Lightning is a very common phenomenon observed in tornadoes and it would be very useful to have control parameters in the tool which would simulate lightning and give users control over them.

h) The parameters of the tool can be categorized into two groups, physically based parameters and visual driven. This could be useful as a learning tool for educating children for both creative and scientific education. Physically based parameters can help them understand physics driven rule-sets while the artistic side of the tool can help in their creative development.

i) Currently, the user has to wait for fluid simulation and rendering processes before changes made to the parameters can be seen. To avoid waiting time, data visualizations can be incorporated which can make the process faster. Data visualization mainly involves visualizing the velocity and density fields rendered in different colors on a 2D grid, instead of simulating the fluid on a 3D container.

j) The user-interface of the tool can be improved to make it more intuitive and easily accessible to the user. Including sample tutorials on creating tornadoes can help distribute the tool over internet so that users can use the tool without any personal training.

k) Allowing the user to input a curve or specify the number of vertices they want in the profile curve to achieve the desired shape would help overcome the current limitation of being limited by a fixed number of control vertices in the tool.

l) Including a feature that displays the user's actions would add interactivity to the tool.

REFERENCES

ASSOCIATEDPRESS. 2010. YouTube web site. http://www.youtube.com/watch?v=80Q1sU8rD2Q. Accessed on September 2009.

AUTODESK®. *Maya, version* 10.0, February 2010.

BAUM, L.F. 1900. *The Wonderful Wizard of Oz.* George M. Hill Company, Chicago.

BERG, P. 2008. *Hancock*, the movie. 92 minutes.

BLUESTEIN, H.B. 2006. *Tornado Alley: Monster Storms of the Great Plains.* Oxford University Press, New York.

BOND, T. 1996. *Night of the Twisters*, the movie. 92 minutes.

BONT, J.D. 1996. *Twister*, the movie. 113 minutes.

CHANDLER, C. 2010. Electromagnetic Nature of Tornadic Supercell Thunderstorm web site. http://charles-chandler.org/Geophysics/Tornadoes%20Main.php. Accessed on September 2010.

DING, X. 2004. *Physically-Based Simulation of Tornadoes*. Master's Thesis, University of Waterloo, Waterloo, Ontario, Canada.

EMMERICH, R. 2004. *The Day After Tomorrow*, the movie. 124 minutes.

FEDKIW, R., STAM, J. AND JENSEN, H.W. 2001. Visual simulation of smoke. In *Proc. of SIGGRAPH 2001*, 15-22.

FLEMING, V. 1939. *The Wizard of Oz*, the movie. 101 minutes.

FLORA, S.D. 1954. *Tornadoes of the United States*. University of Oklahoma Press, Norman.

FOSTER, N. AND METAXAS, D. 1996. Realistic animation of liquids. *Graphical Models and Image Processing 58*, 204-212.

GRAZULIS, T.P. 2001. *The Tornado: Nature's Ultimate Windstorm.* University of Oklahoma Press, Norman, OK.

LIU, S.-G., WANG, Z.-Y., GONG, Z., CHEN, F.-F. AND PENG, Q.-S. 2006. Physically based modeling and animation of tornado. *Journal of Zhejiang University*, 1099-1106.

LORD, P. 2009. *Cloudy with a Chance of Meatballs*, the movie. 90 minutes.

NATIONALGEOGRAPHIC. 2007. YouTube web site. http://www.youtube.com/watch?v=43VoMesUd2Q. Accessed on December 2010.

NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION, (NOAA). 1999. National Oceanic and Atmospheric Administration Photo Library web site. http://www.photolib.noaa.gov/htmls/nssl0210.htm. Accessed on January 2011.

NATIONAL WEATHER SERVICE. Thunderstorm Wind Event - National Weather Service web site. http://www.crh.noaa.gov/mkx/document/wind/wind_10-04-02.php. Accessed on Januray 2011.

NC911. 2004. Tornado Tour web site. http://www.nc911.com/Misc-Photos/tornado_tour2004.htm. Accessed on December 2010.

NOSSECK, N. 1996. *Tornado*, the movie. 89 minutes.

PETER, C. Natural Disasters Tornado Pictures - National Geographic web site. http://environment.nationalgeographic.com/environment/photos/tornado-general/#/tornado-11_21192_600x450.jpg. Accessed on October 2010.

REED, J. Science Photo Library - The Telegraph web site. http://www.telegraph.co.uk/science/picture-galleries/6514909/Science-Photo-Library-photos-of-the-week.html?image=3. Accessed on January 2011.

SIDE EFFECTS SOFTWARE. 2010. Voronoi Fracture surface node - Side Effects Software web site. http://www.sidefx.com/docs/houdini11.0/nodes/sop/voronoifracture. Accessed on December 2010.

SIDE EFFECTS SOFTWARE Inc. 2011. *Houdini*, *version* 10.0, February 2010.

STAM, J. 1999. Stable fluids. In *Proc. of SIGGRAPH 1999*, 121-128.

TAUSIF, A. AND NOVARA, C. 2010. Tornadoes Plug-in for Houdini. Report. SFX Tricks of the Trade. http://etausif.com/Documents/TornadoReport.pdf. Accessed on September 2010.

WIKIPEDIA. 2010. Wikipedia, the free encyclopedia web site. http://en.wikipedia.org/wiki/Gustnado. Accessed on December 2011.

WIKIPEDIA. 2011. Wikipedia, the free encyclopedia web site. http://en.wikipedia.org/wiki/The_Wonderful_Wizard_of_Oz. Accessed on February 2011.

APPENDIX A

1. Fluid Simulation Parameters

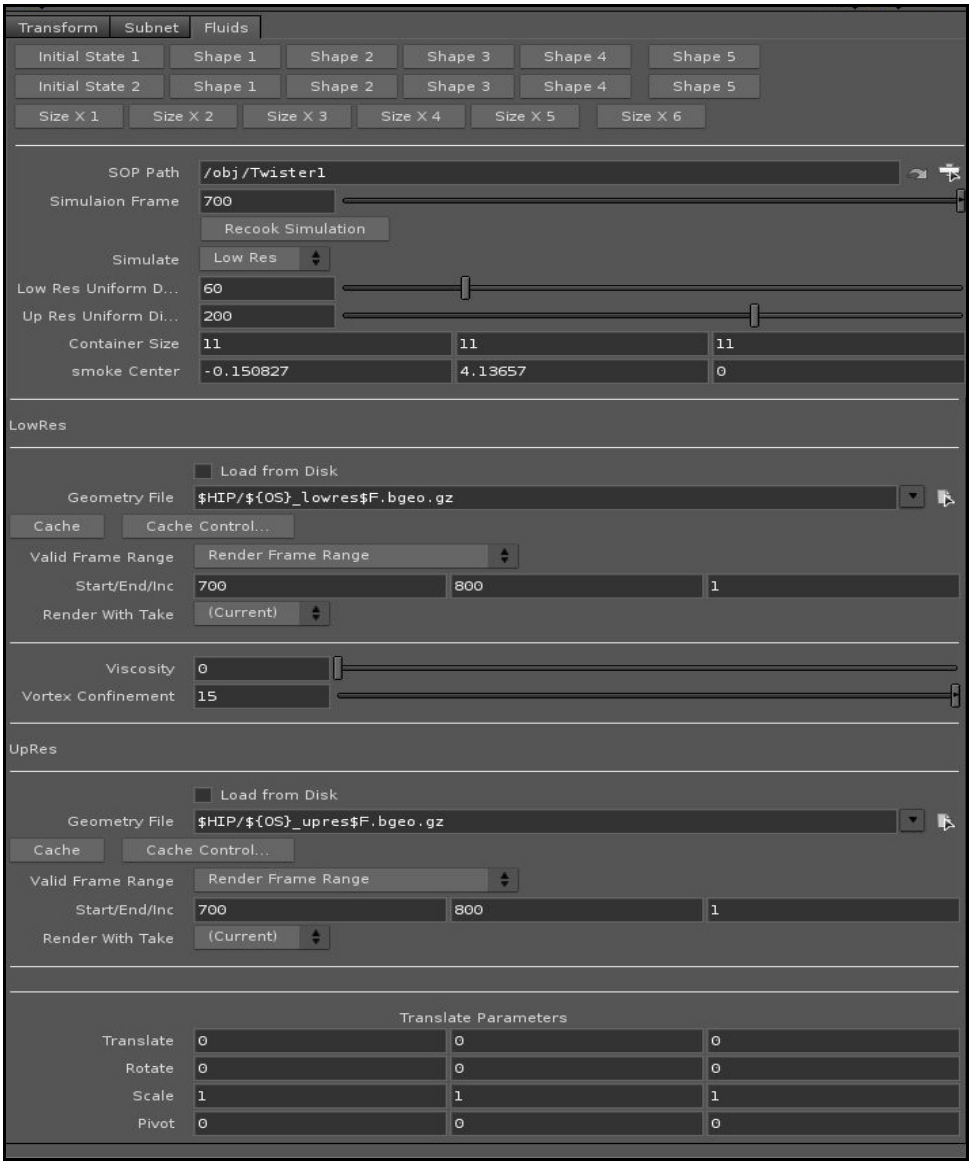


Figure A.1: Fluid Simulation control parameters.

Figure A.1 above shows a snapshot of the tool used to control fluid parameters. These parameters provide control over various aspects of fluid behaviors. Listed below are some of the important parameters and their functions.

Table A.1: Important fluid parameters and their functions.

S.No.	Parameter	Function
1.	Initial State 1 - Initial State 2	Creates default container to fit tornado created using Initial State 1 -2 in particle simulation tool.
2.	Shape 1 - Shape 5	Default container sizes corresponding to funnel shapes in particle simulation tool.
3.	Size X 1 - Size X 6	Default container sizes corresponding to funnel sizes in particle simulation tool.
4.	SOP Path	The SOP path from which the particle data is extracted.
5.	Simulation Frame	Specifies the frame at which the simulation begins.
6.	Recook Simulation	Forces the simulation cache to be cleared and the simulation to be recooked up to the current time.
7.	Low Res Uniform Divisions	This allows one to control the overall resolution with one parameter and still preserve uniform voxels.
8.	Up Res Uniform Divisions	This allows one to control the overall resolution with one parameter and still preserve uniform voxels.
9.	Container Size	The size of the voxel grid. The size of each voxel will be this value divided by the divisions.
10.	smoke Center	The position in world space of the center of the voxel grid.
11.	Load from Disk	Use low res cache data.
12.	Geometry File	Specifies output file for caching low res fluid data.
13.	Viscosity	Viscosity is a force which tries to ensure that neighboring voxels have the same velocity.
14.	Vortex Confinement	This parameter will cause existing vortices to be boosted by this value.

2. Particle Simulation Parameters

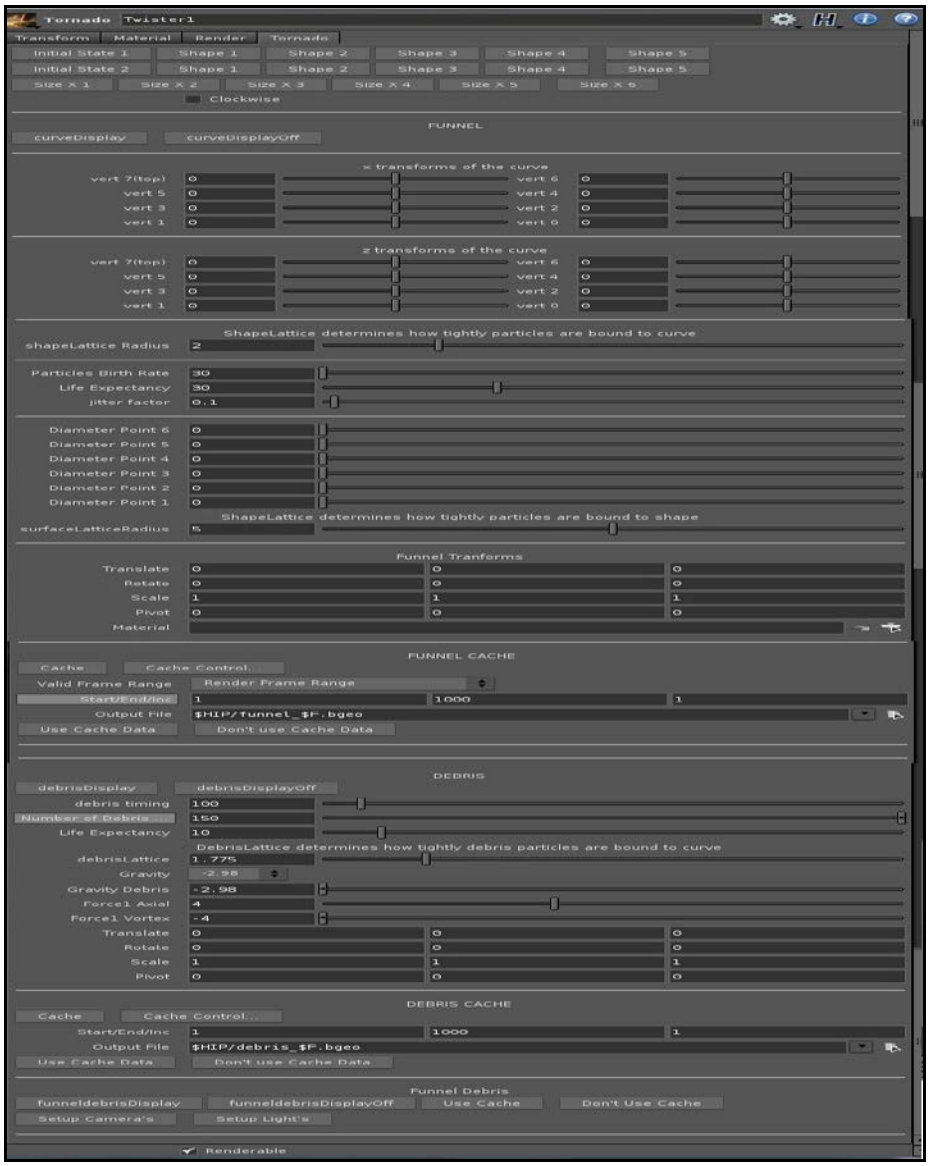


Figure A.2: Particle Simulation control parameters.

Figure A.2 above shows a snapshot of the tool used to control particles. These parameters provide control over various aspects of particle simulation. Listed below are some of the important parameters and their functions.

Table A.2: Important particle parameters and their functions.

S.No.	Parameters	Functions
1.	Initial State 1 - Initial State 2	Default tornado funnel.
2.	Shape 1 - Shape 5	Default funnel shapes.
3.	Size X 1 - Size X 6	Default scales of funnel.
4.	Clockwise	Un-checking this toggle will turn on anti-clock wise rotation of particles.
5.	curveDisplay	Visualizes profile curve.
6.	curveDisplayOff	Turns curve visualization off and displays particles.
7.	vert 7- vert 0	Translates vertex 7 - 0 of the curve in x and z directions.
8.	shapeLattice Radius	Controls deformations of particles along the profile curve.
9.	Particles Birth Rate	Number of particles to emit per second.
10.	Life Expectancy	How long the particle will live (in seconds).
11.	jitter factor	Determines how much the particles will be jittered from their positions.
12.	Diameter Point 6 - Diameter Point 0	Controls the thickness of the funnel at vertex no.7 - 0 of the profile curve.
13.	surfaceLatticeRadius	Controls deformations of particles along the revolved surface.
14.	Output File	Location to save cache files.
15.	Use Cache Data	Uses cached data in output location.
16.	Don't use Cache Data	Cache data is no longer used. Real time simulation data is used.
17.	debrisDisplay	Visualizes debris particles.
18.	debrisDisplayOff	Turns off visualization of debris particles. Only funnel particles are displayed.

Table A.2: Continued.

19.	debris timing	Specifies the frame no. at which the debris particles starts to emit.
20.	Gravity	Specifies gravity acting on debris particles.
21.	Force1 Axial	Magnitude of force which pulls the particle towards positive Y-axis within outer metaball region.
22.	Force1 Vortex	Magnitude of vortex force which makes the particles rotate as they are being pushed away from the center.
23.	Force2 Axial	Magnitude of force which pulls the particle towards positive Y-axis within inner metaball region.
24.	Force2 Vortex	Magnitude of vortex force which makes the particles rotate as they are being pulled towards the center.
25.	funneldebrisDisplay	Visualize funnel and debris particles.
26.	funneldebrisDisplayOff	Only funnel particles are displayed.
27.	Setup Camera's	Creates 2 cameras and 2 render nodes with default settings.
28.	Setup Light's	Creates key, fill and rim lights in the scene.

3. Python Modules

Python scripting was very useful in tool building. It provided easy access to the node parameters. Different types of python classes and functions were used in the implementation. To achieve same task with different nodes, similar code was used. For example, the code to display the node is the same for each node except it references the current node. Similarly, code for setting different parameters of a node with a click of a

button is the same among different nodes. Below are one example each of different types of python modules used with code and their function.

(a) Displaying a node:

Example:

nodedisplayNode : This module is called from "curveDisplay" parameter in particle simulation tool for displaying the curve node. The codes for the function call and python module are:

Callback Script:
```
hou.node('.').hdaModule().displayNode()
```

Module:
```
def displayNode():
    nd = hou.node('./curveParticle')
    nd.setDisplayFlag(True)
    nd.setRenderFlag(True)
```

(b) Creating a node:

Example:

createCam: This module is used to create two cameras in the scene and set values for their parameters. It is called from "Setup Camera's" parameter in particle simulation tool. The code and function call are:

CallBack Script:
```
hou.node('.').hdaModule().createCam(-2.3,-2.3,37,8,0,0,4,5,90,-2,2,0)
```

Module:
```
def createCam(t1x,t1y,t1z,r1x,r1y,r1z,t2x,t2y,t2z,r2x,r2y,r2z):
    hou.node('/obj').createNode('cam')
    hou.node('/obj/cam1').moveToGoodPosition()
```

```
hou.node('/obj/cam1').parm('tx').set(t1x)
hou.node('/obj/cam1').parm('ty').set(t1y)
hou.node('/obj/cam1').parm('tz').set(t1z)
hou.node('/obj/cam1').parm('rx').set(r1x)
hou.node('/obj/cam1').parm('ry').set(r1y)
hou.node('/obj/cam1').parm('rz').set(r1z)

hou.node('/obj').createNode('cam')
hou.node('/obj/cam2').moveToGoodPosition()
hou.node('/obj/cam2').parm('tx').set(t2x)
hou.node('/obj/cam2').parm('ty').set(t2y)
hou.node('/obj/cam2').parm('tz').set(t2z)
hou.node('/obj/cam2').parm('rx').set(r2x)
hou.node('/obj/cam2').parm('ry').set(r2y)
hou.node('/obj/cam2').parm('rz').set(r2z)
hou.node('/out').createNode('ifd')
hou.node('/out/mantra1').moveToGoodPosition()
hou.node('/out/mantra1').parm('camera').set('/obj/cam1')
hou.node('/out/mantra1').parm('vm_volumestepsize').set(1)
hou.node('/out/mantra1').parm('soho_autoheadlight').set(0)
hou.node('/out').createNode('ifd')
hou.node('/out/mantra2').moveToGoodPosition()
hou.node('/out/mantra2').parm('camera').set('/obj/cam2')
hou.node('/out/mantra2').parm('vm_volumestepsize').set(1)
hou.node('/out/mantra2').parm('soho_autoheadlight').set(0)
```

(c) Setting parameters of a node:

Example:

resize: This module is used to set the parameters of a node based on the values passed to the module. This module is called from "Size x 2" parameter in the particle simulation tool. The codes and function call are:

Callback Script:

```
hou.node('.').hdaModule().resize(2,2,2)
```

Module:

```
def resize(x,y,z):
    nd = hou.node('.')
    szx = nd.parm('sx')
```

```
szx.set(x)
szy = nd.parm('sy')
szy.set(y)
szz = nd.parm('sz')
szz.set(z)
```

4. Python Resources

Python is a high level programming language which provides an efficient solution with less lines of code and it can be easily integrated into a system. Many of the 3D animation software's such as Maya [Autodesk®] and Houdini provide support for python as a scripting language which makes it useful as a cross-platform discipline. To learn more about python programming language, "http://python.org/" website is a useful resource.

Other resources for python in Houdini can be found at Side Effects web resource, "http://www.sidefx.com/docs/houdini9.5/hom/". In the implementation in this thesis, "hou" module was used extensively. It contains various sub-modules, classes and functions which provide access to Houdini.

"http://www.sidefx.com/docs/houdini9.5/hom/hou/" is a useful web resource for exploring "hou" module. This location can also be found in Houdini "help" pages. "http://www.sidefx.com/docs/houdini10.0/hom/assetscripts" website was beneficial in explaining how to access python modules inside a digital asset with practical examples.

5. Questionnaire

Questions that were asked to the participants in the evaluation process are listed below.

1. Have you used any 3D animation tools before?

2. If yes which software did you use?

3. Did you use Houdini before? If yes which feature of Houdini have you explored?

4. How user friendly did you find the tool to be?

5. Were you able to control the simulation with the parameters easily?

6. Did you find the layout of the parameters of the tool intuitive and easy to find?

7. Were the help tooltips in the tool clear enough to help understand its features?

8. How long did it take you to complete the simulation?

9. Do you think the naming conventions of the parameters were intuitive?

10. Do you think the tool can be used in a production level to simulate tornadoes? If no, what do you think needs to be changed or improved to make it useful in production level?

11. Were you able to simulate the type of tornado you were aiming for? If no, which feature do you think needs to be implemented to be able to achieve that goal?

12. Do you think the simulation was slow and took longer time than you expected?

13. What other features do you think needs to be added?

14. What features do you think are redundant and can be removed without affecting the usability of the tool?

15. What other suggestions do you have to improve the tool?

VITA

Name:	Ravindra Dwivedi
Address:	Texas A&M University, C108 Langford Center.
	College Station, TX 77843-3137.
Email Address:	ravindra.dwivedi@gmail.com
Education:	B.E., Computer Science & Engineering, Rajiv Gandhi Proudyogiki.
	Vishwavidyalaya, Bhopal, 2006.
	M.S., Visualization Sciences, Texas A&M University, 2011.