Modeling and Animation of Orb Webs

A Thesis Submitted to the

College of Graduate Studies and Research

in Partial Fulfillment of the Requirements

for the degree of Master of Science

in the Departmentof Computer Science

University of Saskatchewan

Saskatoon

By

Anubhav Mehla

Permission to Use

In presenting this thesis in partial fulfilment of the requirements for a Postgraduate degree from the University of Saskatchewan, I agree that the Libraries of this University may make it freely available for inspection. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department or the Dean of the College in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that due recognition shall be given to me and to the University of Saskatchewan in any scholarly use which may be made of any material in my thesis.

Requests for permission to copy or to make other use of material in this thesis in whole or part should be addressed to:

Head of the Department of Computer Science

University of Saskatchewan

Saskatoon, Saskatchewan S7N 5A9

Abstract

Modeling of natural phenomena has been of particular interest in the graphics community in recent years. This thesis will explore a method for creating and animating orb webs using a coupled spring-mass system. Using a spring-mass system for creating the orb web is ideal as we can represent each web strand using coupled spring-mass pairs. This allows the orb web simulator to be physically based, i.e., the simulation follows the laws that act on objects in the real world. This in turn simplifies the process of animating the web, as the animation emerges from the simulator without anyone having to set it up explicitly. Since this model is physically based, it would allow for realistic visualization of effects such as observing an orb web under a wind.

In the children's book "Charlotte's Web", the spider creates orb webs with words inscribed on them. Charlotte's web is used as an inspiration, in this thesis, to create webs which no real world spider could possibly create, while keeping the model physically based. This involves modifying the orb web such that the target text shows up on the orb web while keeping the web looking as natural as possible.

Acknowledgements

I would like to express my most sincere gratitude to Dr David Mould, my supervisor, for his guidance, inspiration and support throughout the last two years. Without him, this dissertation would not have happened. Dr Mould not only provided me with the ideas which form the basis of this dissertation, but also provided numerous clear and concise suggestions, which helped in bringing this work to its conclusion. He was also tireless in helping with the numerous revisions of this thesis document.

I would like to thank Dr Subramanian, my thesis committee member, for his help and support during my MSc research. I would also like to thank Dr Eric Neufeld and Jan Thompson, who facilitated my studies in the Department of Computer Science.

Finally, I would like to express my gratitude to my parents for providing me a sound educational foundation and for their unwavering support during all my academic pursuits.

Contents

Chapter 1

Introduction

Computer Graphics is one the most exciting areas of growth and research in modern technologies. Starting with early applications such as the use of an oscilloscope to display waveforms, this field has grown at an astounding pace. The three main areas of Computer Graphics are *modeling, rendering* and *animation.* Modeling involves creating a description of the objects in a scene, and is the focus of this thesis. While rendering is the process of using the defined model to create an image on the screen, animation involves assembling a sequence of rendered images.

Nature offers us a limitless source of inspiration and challenge in modeling of natural phenomena such as feathers, water, smoke, fire, clouds, trees, animals and countless others. Some key applications of the created models of a natural phenomenon include the following: Computer Animation in movies and multimedia applications, Virtual Environments for flight simulators and war gaming, Rapid Prototyping of scientific models, Computer Games, Medical Simulation and Analysis, Archaeological Reconstruction, Chemistry to create realistic models of molecules and their interaction, and in Meteorology to create accurate representations of quickly developing weather conditions. Further, some of the created models might be useful in conducting scientific research.

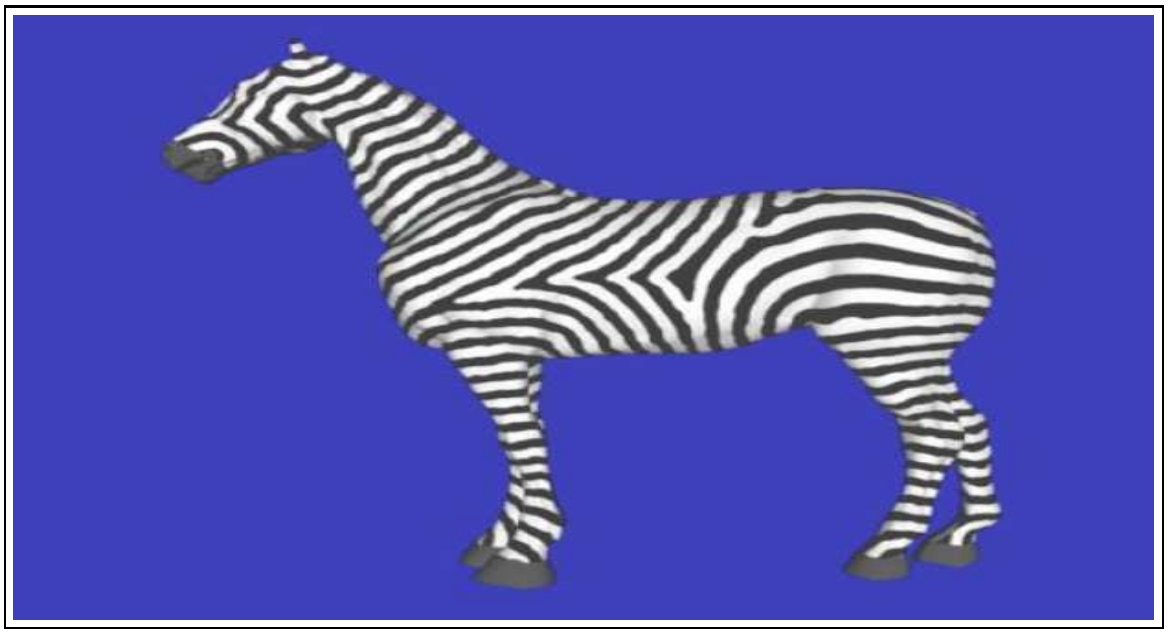

Figure 1.1: Skin patterns in a Zebra[37]

The key focus in modeling is to be able to represent the natural phenomenon under study with enough accuracy to convince most humans of its fidelity. The whole process of creating believable visual representations of natural phenomena incorporates the work of people in different disciplines. As an example, the skin patterns of animals (figure 1.1) can be created by reacting and diffusing morphogens [37]. This requires some understanding of the underlying biological process, some math, computer graphics and programming knowledge. Another example of modeling of a natural phenomenon is given in Figure 1.2, which shows an image created using a model for snow generation, accumulation and stability [8].

Simple modeling techniques using polygons, patches, points and lines are insufficient to represent the complexities of nature. Modeling techniques in Computer Graphics have advanced significantly in the last few years. Most advanced modeling techniques are *procedural modeling* techniques: code segments or algorithms are used to abstract and encode the details of the model, instead of explicitly storing vast

Figure 1.2: A frame from the short animation "Santa's Suitcase"[8]

numbers of low-level primitives. For example, a procedural texture for a marble surface does not use a scanned image to define color values. Instead it uses algorithms and mathematical functions to define color. We can use parameters to control the output of a procedural model. For example, we could have a number that can make the mountains rougher or smoother. Another advantage of procedural models is that the procedural models offer flexibility. We can capture the essentials of the object, phenomenon or motion without being constrained by the complex laws of physics. We can incorporate the desired amount of physical accuracy into the model or we can accurately simulate physical laws or create purely artistic effects. Some procedural techniques include: fractal-based models, grammar-based models, volumetric models, implicit surfaces and particle systems. The work in this thesis is based on the last mentioned technique, the particle systems.

Physically-based modeling attempts to map a natural phenomenon to a computer simulation. Figure 1.3 shows an example of a synthesized gold ring with an in-

Figure 1.3: A physically-based inscription on a synthetic ring[5]

scription. This image is created using a physically-based model which simulates the process of scratch formation on a metallic surface [5]. Figure 1.4 shows the picture of a night sky created using a physically-based model of the night sky [39]. Mathematical modeling and numerical solution to the model are two basic processes in this mapping. Mathematical modeling concerns itself with the description of the natural phenomena by mathematical equations. Differential equations that govern the dynamics and the geometric representation of the objects are the typical ingredients of the mathematical model. The numerical solution of the mathematical equations is required to be both accurate and efficient, and in the implementation of this thesis we have used the fourth order Runge-Kutta method (see chapter 4).

Physically-based techniques facilitate the creation of models capable of automatically synthesizing complex shapes and realistic motions that were, until recently, attainable only by skilled animators, if at all. Physically-based modeling adds new levels of representation to virtual objects. In addition to geometry, other physical quantities such as forces, torques, velocities, accelerations, kinetic and potential en-

Figure 1.4: A physically-based model of the night sky[39]

ergies and heat are used to control the creation and evolution of models. Simulated physical laws govern model behavior, and animators can guide their models using physically-based control systems. Physically-based models are responsive to one another and to the simulated physical worlds that they inhabit. This thesis focuses on creating a physically-based model of a particular type of spider web called an *orb web*.

Spider webs are beautiful to look at and intricately made. There are more than 35,000 known spider species, and different spider species make webs which vary in their shapes, sizes and construction. Spiders can, very broadly, be divided into two groups: the *web-building spiders* and the *wandering spiders*. Wandering spiders are much more active than web builders, seeking out prey and creating webs for shelters or to hold their eggs. Web-building spiders are usually less active, building their webs and patiently waiting for the prey to come to them. Within this category, there are several types of recognized web patterns. Some of the well known patterns are:

Figure 1.5: Sheet Web[17]

Figure 1.6: Funnel Web[17]

- Orb webs : These webs woven by certain spider species are suspended, sticky and wheel-shaped. They are usually placed in openings between trees and shrubs where insects are likely to fly.

- Sheet webs: These are relatively flat mats of silk, usually with a funnel-shaped retreat at one end (figure 1.5). The spider hides in this funnel until prey stumbles and becomes entangled in the matting.

- Cob webs: A framework of threads supports the trap threads that adhere to a horizontal surface. Prey will trip the trap threads and become entangled.

- Funnel webs: These webs are in the shape of a funnel (figure 1.6). The spider hides in wait at the narrow end and leaps out at passing prey, dragging it back into its lair.

We chose to model orb webs in this thesis as they look the most like the archetypal spider webs, as perceived by the majority of people.

The orb web is modeled and animated using a *spring-mass system*. A spring-mass

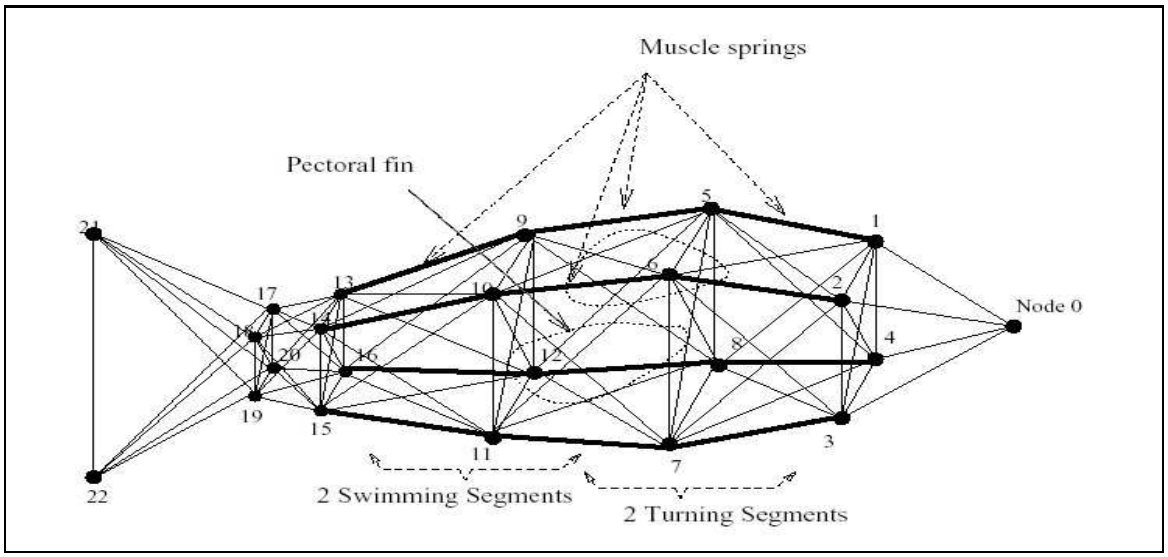

Figure 1.7: A spring-mass fish model[36]

system is an extension to the concept of a particle system in which the particles are inter-connected by springs (figure 1.7 and figure 1.8). The work in this thesis in modeling and animating orb webs draws upon the concepts of procedural modeling and physically-based modeling. The initial model is set up using a procedural approach. Algorithms based on observed biological facts are used to create the orb web model. Once the model is set up it is acted upon by various forces and its behavior observed under an external force (wind).

An animator requires control over physics-based models in order to produce useful animations. We can categorize physics-based control techniques into two approaches: the *constraint-based approach* and the *motion synthesis approach*. The constraint-based approach involves the imposition of kinematic constraints on the motions of an animated object [19]. For example, one may constrain the motion trajectories of certain parts of a model to conform to user specified paths. The animation of an orb web subject to a wind does not require a complex control mechanism. The beauty of the approach used is that once the model is set up and subjected to a wind, the

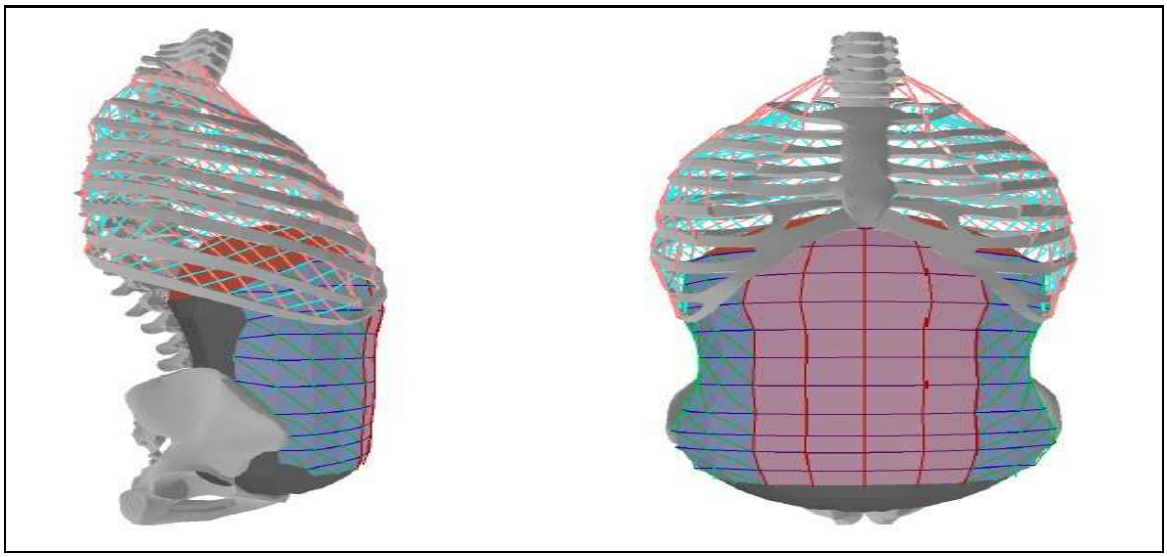

Figure 1.8: A spring-mass chest model[45]

animation flows out of the model.

Considerable success has been obtained in modeling and animating phenomena such as fire and smoke using particle systems [23]. Building on the basic concepts of particle systems, the spring-mass systems have been used to model both *passive objects* such as cloth [22], and *active objects* such as snakes and worms [13]. Passive objects have no internal sources of energy and active objects have an internal energy source . We construct our orb web model using a *deformable spring-mass system*. There are several reasons for this choice:

- A spring-mass model is a simple discrete mechanical structure capable of non-linear, nonrigid dynamics; this is well suited to creating a model of an orb web using a mesh of interconnected spring-mass pairs.

- The spring-mass units with damping in the model are viscoelastic units that serve both as geometric and deformation control primitives.

- Both *passive* and *active* deformable objects have been successfully animated

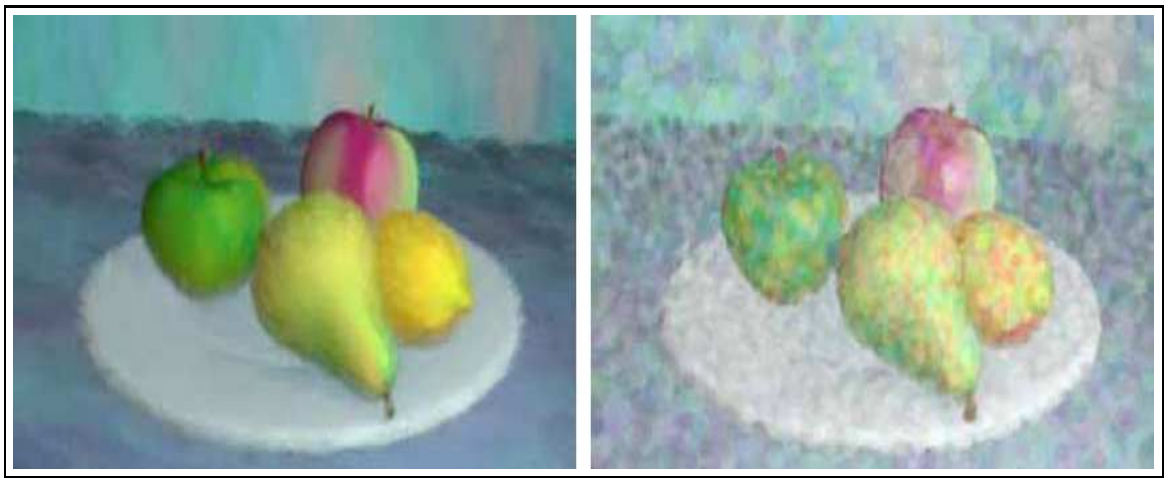

Figure 1.9: Two styles of painterly rendered fruit[45]

using similar spring-mass models.

- As each strand of the artificial orb web can be represented using multiple spring-mass pairs, the spring-mass model appears well suited for modeling orb webs.

Although the automatic creation of the orb web model was a major contribution of this thesis, this model has further been used to create a very interesting non-photorealistic effect. In Computer Graphics *photorealistic rendering* attempts to make artificial images of simulated 3D environments that look like the real world. *Non-photorealistic rendering* (NPR) is any technique that produces images of simulated 3D world, in a style other than realism. NPR techniques combine the expressivity of artistic media with the flexibility of computer graphics. Often these NPR styles are reminiscent of paintings [11]. An example would be the painterly rendering of fruits shown in figure 1.9. *Charlotte's Web* [41] is a children's book in which the spider weaves webs with text inscribed on them to save a pig from being butchered. These spider webs, which are illustrated in the book, are used an an inspiration to modify the physically-based orb webs such that they are inscribed with text or

symbols. This produces a very intriguing non-photorealistic effect of a spider web inscribed with humanly understandable text or symbols.

The work done in this thesis can be best summarized by dividing it into four distinct phases, as is described below.

- First Phase:

  The *first phase* involves the creation of the physical model of the orb web. In this phase the primary goal was to create a model of an orb web taking into account the characteristics of real orb webs. For this purpose the work of Samuel Zschokke [46] is used as a primary source of information. The model which is made of spring-mass pairs is subjected to Newtonian forces and the resulting *Ordinary Differential Equation's* (ODE's) are then solved using the Runge-Kutta method to arrive at the new state of the system. The challenge here is to arrive at a stable state of the system, in a short amount of time (a few seconds), without causing instability. Another challenge is to keep the orb web model as close as possible to a real orb web. Since there are a large number of widely varying orb web structures, orb webs made by two particular species of spider are used as a basis for evaluating the results of the work of this phase.

- Second Phase:

  In the *second phase* we aim to inscribe our webs with text, while striving to keep the same overall appearance of the web and not altering its physically-based nature. In order to accomplish this we first create an artificial orb web using the model developed in phase one. The text to be inscribed is then superimposed on this web (for details see section 6).

- Third Phase:

  In the *third phase*, the orb web model is animated by subjecting it to a wind force. Various wind models have been proposed in literature. These models often create a wind field using complex techniques, but much of the complexity can be avoided in the case of this simulation. The orb web in this simulation can be considered as a two dimensional structure with a maximum diameter of around one meter. The dominant forces on the artificial web are gravity and the spring-forces. Since the effect of the wind force is limited in comparison to other forces in the simulation, the need for developing complex flow fields does not arise. The wind animation is created by collating the images of the orb web swaying in the wind.

- Fourth Phase:

  The *fourth phase* involves placing the orb web in natural surroundings so as to enhance the realism and the aesthetic beauty of the web. The idea here is to place the model in a skybox, with the web being hung outdoors in between the branches of a tree. A digital image is texture mapped onto the screen and the orb web constructed in between the branches of the texture mapped tree.

Although an attempt has been made to use the information available on orb webs to create a visually accurate model of an orb web, the orb web model is not an accurate physical model of a real orb web. The created model can be used in applications such as a computer game where the emphasis is on creating a believable visual representation and not the physical accuracy of the model.

The work done in this thesis draws on the concepts and principles developed for

modeling phenomena using particle and spring-mass systems; chapter 2 talks about related work using these concepts. This is followed by chapter 3, which describes the features of the orb web which were taken into consideration while creating the web. Chapter 4 contains a discussion on particle and spring-mass systems. Chapter 5 details the algorithms used in the web construction and animation process. It also presents the results of the work done in this thesis and carries out a discussion of the presented results. The process of creating the non-photorealistic *Charlotte's web* is described in chapter 6. Finally, conclusions are drawn from this work and areas for future work are suggested in chapter 7.

1.1 Implementation Note

To provide easy access to computer graphics hardware and to ease the development of computer graphics software, a number of application programmer interfaces are available. The API used in implementing this thesis is OpenGL. Apart from being in widespread use, it has the advantage of being a cross-language, cross-platform API. Efficient implementations of OpenGL, which leverage graphics acceleration hardware to a greater or lesser extent, exist for Windows, many Unix platforms, and Mac OS. Mesa, a 3-D graphics library with an API which is very similar to that of OpenGL, is used in this thesis implementation. The language used is *C* and the development platform is Mandrake Linux.

Chapter 2

Related Work

The work in this thesis draws on the work done in both biology and computer graphics. In this thesis, a system of particles coupled using springs is used to create a model of an orb web. The particles which constitute the orb web are subjected to a gravitational force, air friction and spring forces. This spring-mass system is then solved numerically to arrive at each subsequent state of the system.

Orb webs have been studied in detail by biologists. This thesis draws on this pool of work as a basis for arriving at an orb web model. In particular, the work done by Samuel Zschokke [46] on orb web construction by spiders is used as a guide for creating the orb web model in this thesis. His dissertation focuses on the web construction behavior of the orb web weaving spider *Araneus Diadematus* [48]. His thesis work features descriptive, pictorial accounts of the way Araneus Diadematus starts the construction of a new web, the effect of the environment on the orb web construction, the planarity of the orb webs, the construction speed of spiders and the factors affecting the size of an orb web. It also describes pictorially the web construction behavior of 14 other orb web weavers. Since there exists a large collection of orb web weaving spiders with markedly different web construction patterns (e.g. see figures 2.1 and 2.2), the focus of our work has been restricted to the orb webs

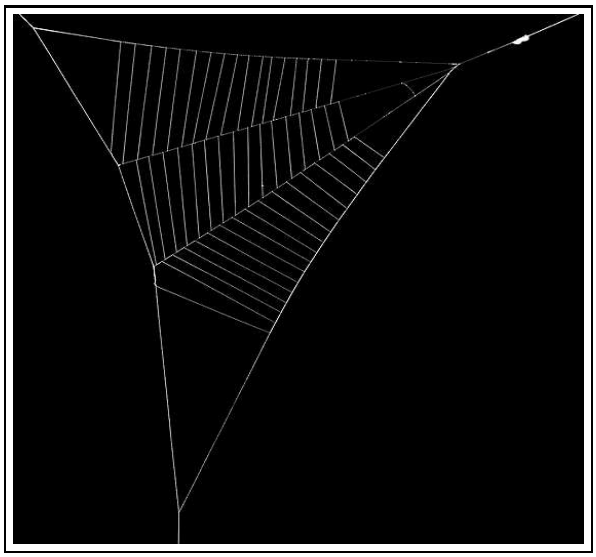

Figure 2.1: Finished Orb-Web of H. Paradoxus[48]

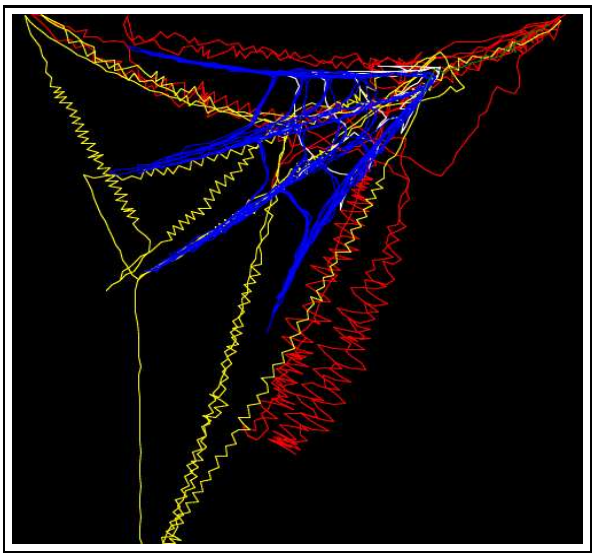

Figure 2.2: Web Construction Tracks[48]

of spiders *Araneus Diadematus* and *Zilla Diodia*. The physical structures of the orb webs of these two types of spiders, along with the other characteristics of these orb webs such as spatial web density, placement of orb web strands and web size are used to guide the construction of our orb web model (see chapter 3 for details).

In Computer Graphics, particles are considered to be objects that have mass, position, and velocity, and respond to forces, but have no spatial extent [43]. One of the earliest publications about particle systems was by Reeves in 1983 [23], who used a "cloud" of particles. These particles were first generated, then moved and finally at the end of their lifetimes they died. This seminal work has since then been extended and applied to numerous phenomena such as forests and grass [24], ocean waves [25] and waterfalls [30]. Reynolds [27] used coupled particles that interact with each other and he used simple rules to model complex behavior. Miller and Pearce [12], Terzopoulos, Platt and Fleischer [34], and Tonnensen [35], used coupled particle systems to model liquid like and melting behaviour. Miller et al. [15], Szeliski and Tonnensen [33], and van Wijk [38], propose particle interactions that

Figure 2.3: Three painting of one image[11]

are a function of direction, producing deformable sheets and surfaces of particles. Spring-mass systems have been used to model the dynamics of snakes and worms [14], to animate facial expressions [20] and to animate cloth [2]. In this thesis, the orb web model is created using a number of spring-mass pairs connected in a series to model a single strand of the orb web. These web strands are then connected using algorithms inspired by the study of the biological model of an orb web.

This physically based orb web model is later used as a basis to create a *Charlotte's Web* [41]. Although much of the work in Computer Graphics is focused on creating synthetic images that resemble the real-world, non-photorealistic rendering (NPR) is a subset of Computer Graphics that focuses on creating images which do not resemble objects in the real-world. One of the earliest papers in NPR was by Paul Haeberli [11] who used an ordered list of brush strokes to convert a synthetic or natural scene into an impressionist image (see figure 2.3). NPR brings closer together the disciplines of art and science. The illustrations in *Charlotte's Web* serve as an inspiration in this thesis for creating non-photorealistic images of orb webs with text inscribed on

them. The attempt here was not to re-create the Charlotte's Web as described in the book but to use it as an inspiration for creating an intriguing spider web, a web which no real spider could possibly create.

The next phase of the thesis involves subjecting the orb web to a wind. Wejchert et al. [40] used simplified aerodynamic equations to model wind and demonstrated their application to a particle system. Shinya et al. [29] developed a general stochastic model which could be applied most of the existing trees and grass models, including structural models, particle systems, impressionist models, and 3D texture. In 1993 Stam and Fiume [32] applied a similar technique to model winds for gaseous phenomena. Jos Stam [31] used a stochastic method based on modal analysis to animate the motion of trees in a wind. In this thesis a simplified wind model is used as a need for using complicated wind models does not arise.

To the best of our knowledge there is no published work which has used the knowledge gained from the biological studies on the orb web to create a physically based model of an orb web, as is done here.

Chapter 3

Orb Web

Spiders are fascinating creatures. Spiders produce spider silk, which on an equal weight basis is twice as strong as steel. Not only is spider silk strong but it is very elastic and it can be stretched up to 40 percent of its length before it breaks. It may require more than 80g of stress to break a thread of silk only 0.1mm in diameter. Different spiders construct webs whose structures vary vastly [18] [47]. The focus of this thesis is a kind of spider web known as an *Orb Web*, which is a round spider web with a pattern of lines spiraling outward from the center. Such webs are build by a large number of spider species and they differ in their orientation, size and strength. We have chosen this type of web as orb webs look the most like the archetypal spider webs, as perceived by the majority of people. Further, they have geometrical properties that make them easier to model than some of the other types of spider webs. Two particular species of orb web weaving spiders, *Araneus Diadematus* and *Zilla Diodia*, pictures of whose webs are given in figure 3.2 and figure 3.4 respectively, are used as reference webs which guide the orb web construction in this thesis. The figures 3.1 and figure 3.3, obtained from the work of Samuel Zschokke [49], show the show the tracks of the spider during web construction. The tracks have been colored to emphasize the different phases of web construction. These tracks were

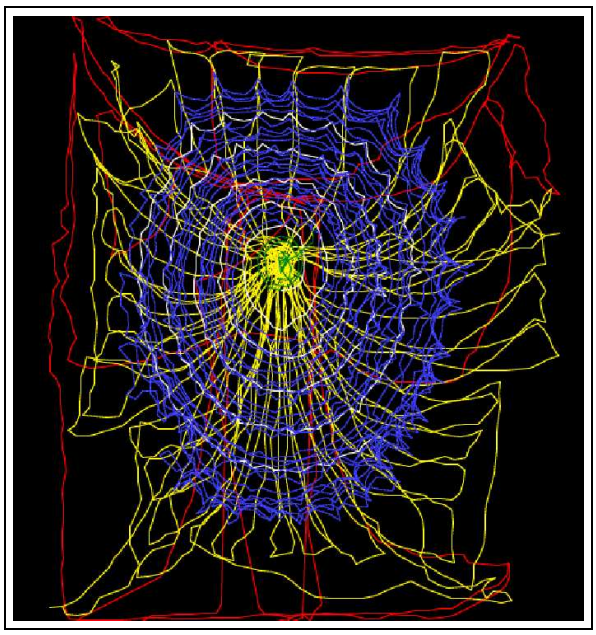

Figure 3.1: Construction tracks of Orb Web of Araneus Diadematus[47]

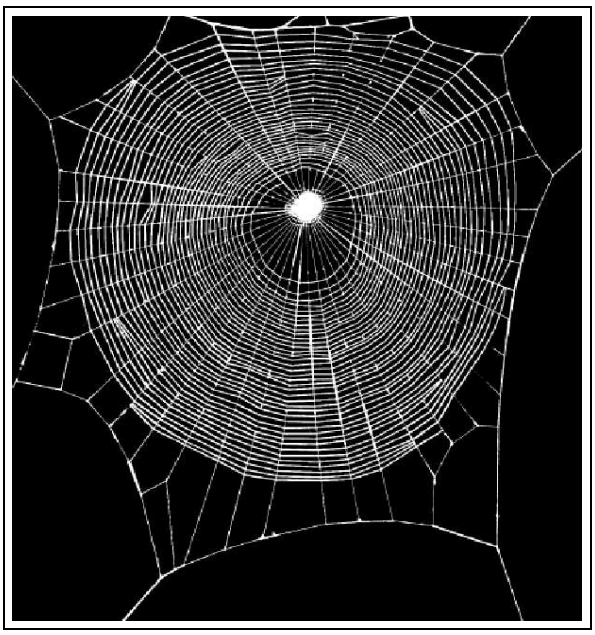

Figure 3.2: Orb Web of Araneus Diadematus[47]

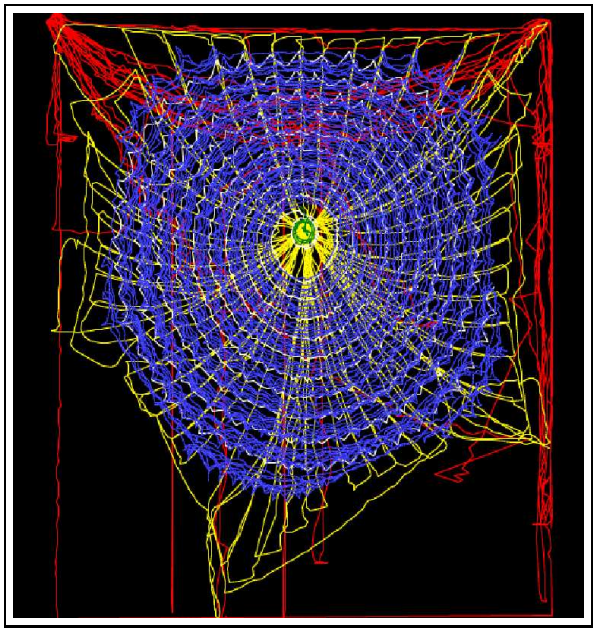

Figure 3.3: Construction tracks of Orb Web of Zilla Diodia[47]

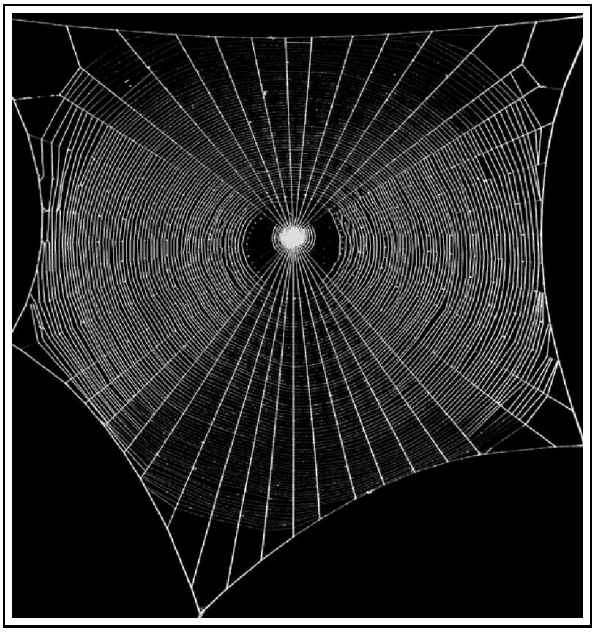

Figure 3.4: Orb Web of Zilla Diodia[47]

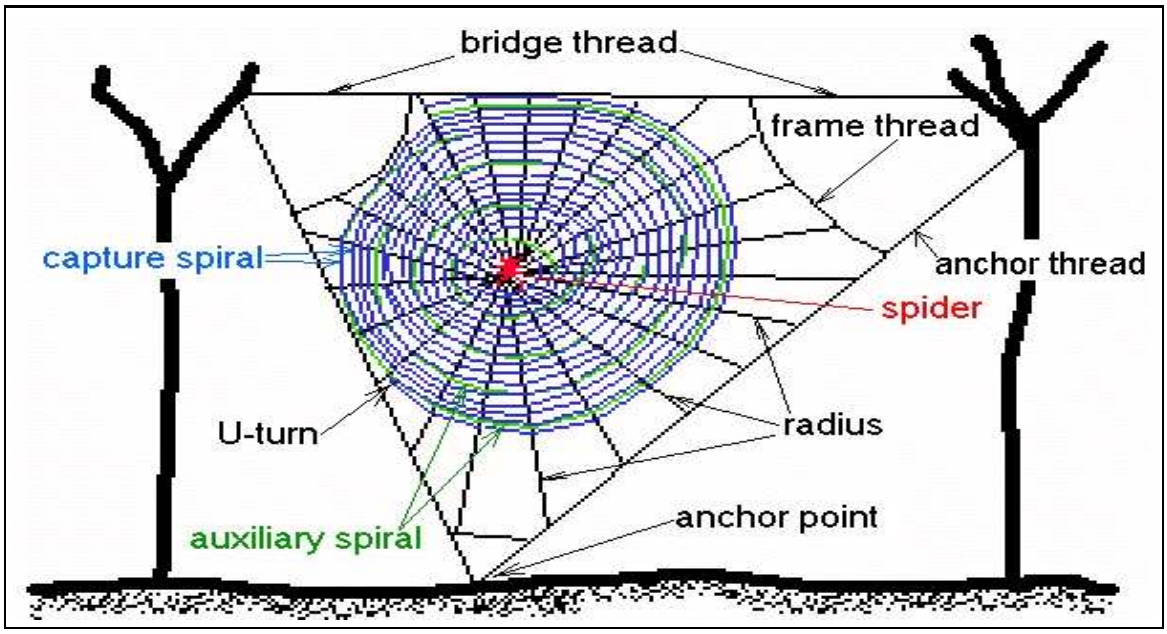

Figure 3.5: Structure of an Orb Web[7]

captured with a video camera connected to an image scanning device [3]. Although these tracks are of great interest to ethologists and taxonomists, who study the web building behavior of spiders, they also provide interesting and relevant information regarding the attachment positions and thickness of spider threads [49]. The spider initially creates and strengthens the outer frame and the hub of the orb web. Then the spider lays out the initial radii, which completes the main support structure of the orb web. The final shape of the orb web is arrived at when the spider creates more attachment points, adds more radii, and finally adds the spiral.

As can be seen from the orb web pictures, the web of *Zilla Diodia* is very finely meshed and more circular than the web of *Araneus Diadematus*. In general such webs are up to a meter or two in diameter and are usually built a meter or two above the ground. As shown in figure 3.5, the spider web itself is made up of various sub-structures. These sub-structures can be used to decompose the complex web into its components, which can then be used as a basis for modeling the web. A list

of the sub-structures of an orb web is as follows:

- Bridge Thread: This thread is the starting point in the web construction process. The spider makes it strong by the spider by repeatedly going back and forth over the thread and laying more silk in each such pass. This thread should be able to support the entire weight of the thread.

- Anchor Thread: This is the next thread that is woven by the spider and is used to support the web. The anchor threads are attached to the supporting structure at the anchor points.

- Frame Thread: The frame thread, when present, is attached to the both sides of the anchor thread. Together with the anchor thread, it forms the outer frame of the spider web.

- Radius Threads: These threads are the connectors from the web center to the frame. They provide further support to the web structure.

- Auxiliary Spiral: The auxiliary spiral is used as reference for laying the capture spiral, the sticky silk. The auxiliary spiral spiral is laid by the spider starting at the hub and then moving outwards towards the frame. Most spiders remove the auxiliary spiral when they lay the capture spiral.

- Capture Spiral: The capture spiral is constructed by the spider by starting the construction on the periphery of the web frame, and then winding inwards towards the hub. It is the only sticky silk in the web and it is used to capture prey.

Some important characteristics of the spider web and the spider silk that have been used to guide the development of the orb web model are:

1. The combined length of silk in a spider's web is about 20-60m [26].
2. Tensile Strength of spider web silk is around 5 denier[4]. A *denier* is equivalent to the weight in grams of 9000 meters of continuous filament fiber.
3. Diameter of a single spider silk strand is around $1\mu m$ [16].
4. Spider silk used in the radii of the web can be extended up to 40% before breaking, and the tensile strength of the radial thread is around 1100 MPa [9].

Using the first two of the above listed facts and taking into account the fact that spiders have multiple fibers between connected points, the weight of a spider web can be roughly estimated to be about 50 mg. With an expected 5000-7000 masses in the web, a mass of 10μg is chosen for each particle.

The orb web is modeled using the concepts of particle and spring-mass systems, which are described in detail in the next chapter. The orb web model is made up of a number of particles and the motion of a particle is governed by Newton's second law of motion, $\mathbf{f} = m\mathbf{a}$. Here $\mathbf{f}$ is the force on a particle, m is the mass of the particle, and $\mathbf{a}$ is the acceleration of the particle [1]. We use the above calculated value of the mass of a particle to calculate the acceleration of a particle under the influence of various forces such as gravity and air friction. This acceleration in turn is used to calculate the motion of a particle under the influence of a set of forces.

[1] Note: Boldface is used to represent vector quantities.

Chapter 4

Particle and Spring-Mass Systems

In this section we describe in detail the particle and the spring-mass systems and their dynamics. Apart from detailing the mathematical and the physical theory behind the systems, we also describe our spring-mass simulator.

4.1 Particle System

A particle system is a collection of many minute particles that model some object [23]. The main use of particle systems is as a method of modeling fuzzy objects with no well-defined surfaces, such as fire, clouds, smoke and water. In a particle system, in each new frame of an animation, new particles are generated and assigned attributes. The particles that have outlived their predetermined life are then destroyed. The remaining particles are transformed and moved according to their dynamic attributes. Finally, an image of the remaining particles is rendered. Since the creation and the attributes of the particles are procedural, these can be the results of other computations based on particular algorithms. For example, the color and size of a fire particle can be set algorithmically, based on the results desired and the algorithm used to achieve these results.

Particle behavior can be specified in terms of *differential equations*. Differential

Figure 4.1: Example Of A Fire Created Using A Particle System[28]

equations describe a relationship between an unknown function and its derivatives [42]. In our system we are concerned with a class of differential equations called *initial value problems*, where the system behavior is described by a system of *ordinary differential equations* of the form

$$\dot{\mathbf{x}} = f(\mathbf{x}, t), \tag{4.1}$$

where f is a known function, $\mathbf{x}$ is the state of the system and $\dot{\mathbf{x}}$ is the derivative of $\mathbf{x}$ and t. Also we are given $\mathbf{x}(t_0) = \mathbf{x}_0$ at some starting time t_0.

The motion of a particle is governed by Newton's second law of motion, $\mathbf{f} = m\mathbf{a}$. Here $\mathbf{f}$ is the force on a particle, m is the mass of the particle and $\mathbf{a}$ is the acceleration of the particle. We can write this as a second order ODE,

$$\ddot{\mathbf{x}} = \mathbf{f}/m, \tag{4.2}$$

where $\ddot{\mathbf{x}}$ is the acceleration. We can convert this second order ODE to a first order

[1] Note: boldface is used to represent vector quantities and t is the time.

ODE by introducing a variable $\mathbf{v}$, which is the velocity of the particle, such that

$$\dot{\mathbf{x}} = \mathbf{v} \tag{4.3}$$

and

$$\dot{\mathbf{v}} = \mathbf{f}/m. \tag{4.4}$$

Now we can use a numerical method to solve the system.

Numerical solutions of an initial value problem can be obtained using a variety of methods. The basic idea in all these methods is to take discrete time steps starting with an initial value $\mathbf{x}(t_0)$. We use the value of the derivative of $\mathbf{x}$ to calculate an approximate change in $\mathbf{x}$ over a time interval and then increment $\mathbf{x}$ by this amount to get the new value. The simplest numerical method is the *explicit Euler* method. It computes an approximate solution by calculating the new value of $\mathbf{x}$ at the beginning of of each step, as follows:

$$\mathbf{x}(t_0 + h) = \mathbf{x}_0 + h\dot{\mathbf{x}}(t_0). \tag{4.5}$$

Since this method assumes a single constant value of the derivative over the time interval, the *step size* h is limited. If we take large steps then we risk moving away from the actual solution by an amount that renders our solution unusable. We can reduce the error size by taking smaller steps at the cost of increasing the overall computation time. This increase in computation time might not be an acceptable trade-off for the system under consideration. Further discussion of this problem is provided in the next section.

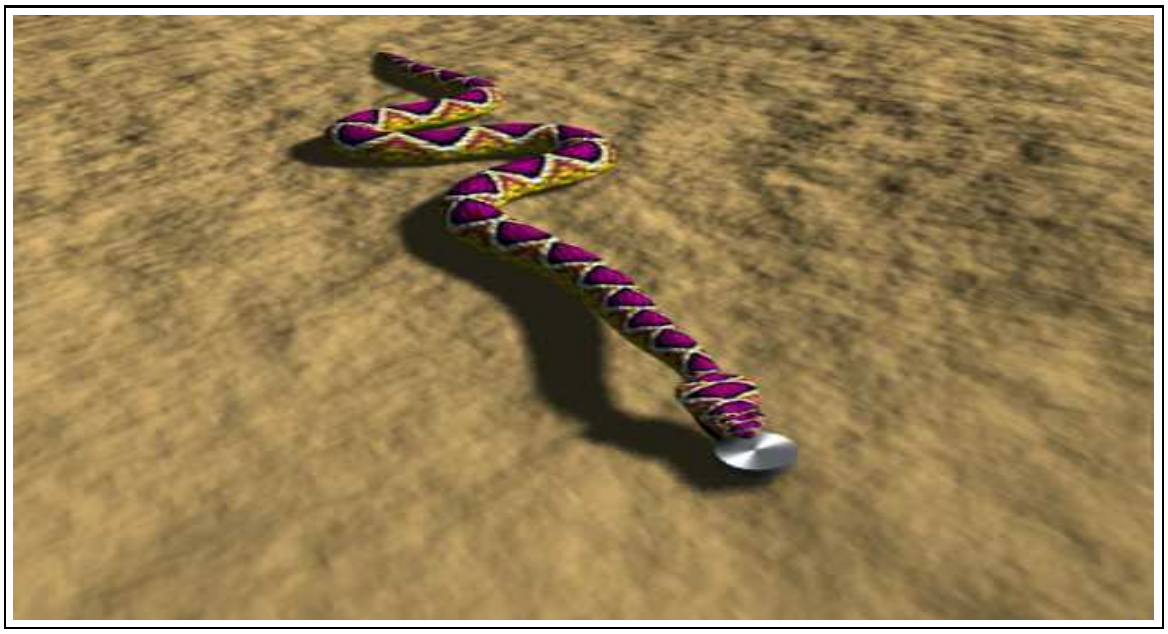

Figure 4.2: Example Of A Snake Made Using A Spring-Mass System[14]

4.2 Spring-Mass System

A spring-mass system is an extension to the particle system that consists of a system of particles interconnected by springs. Such systems have been successfully used to represent a wide range of deformable objects such as fire [23], trees [25], snakes and worms [12] and fluids [34].

The particles in a spring-mass system are acted on by various forces. The whole spring-mass system is implemented such that it is easy to introduce new types of forces. The forces in our system can be grouped into two main categories as *unary* forces and n-ary forces [43].

The unary forces act independently on each particle and can be constant or might depend on the particle position, velocity or time. In our system there are two unary

forces acting on the particles. The first is a constant *gravitational force* given by

$$\mathbf{f} = m\mathbf{g}, \tag{4.6}$$

where $\mathbf{g}$ is a constant vector with a magnitude equal to the gravitational constant and m is the particle mass. The second unary force in our system is the *viscous drag* given by the equation

$$\mathbf{f} = -k_d\mathbf{v}, \tag{4.7}$$

where k_d is a constant called the *coefficient of drag* and $\mathbf{v}$ is the particle velocity. This force resists motion, therefore it causes the system to settle down more quickly than it would have if this force was not present. If too much of this force is applied it makes the system of particles seem like they are immersed in a thick fluid, thus making the system look less realistic.

The *n*-ary forces act on a fixed set of particles. Consider a simple spring-mass system in which a mass is attached to the end of a linear spring and hung under the force of gravity (figure 4.3). According to *Hooke's Law*, the restoring force due to a spring is proportional to the length that the spring is stretched, and acts in the opposite direction:

$$\mathbf{f} = -k_s\mathbf{x}, \tag{4.8}$$

where k_s is the spring constant and $\mathbf{x}$ is the displacement of the mass from its equilibrium position. The above described spring-mass system is in an equilibrium state when the weight of the mass, $m\mathbf{g}$, is balanced by the restoring force of the spring as given by Hooke's Law, $-k_s\mathbf{x}$. Here m is the mass, $\mathbf{g}$ is gravity, and $\mathbf{x}$ is the displacement of the mass from the rest position. This can be written as

$$m\mathbf{g} = -k_s\mathbf{x}. \tag{4.9}$$

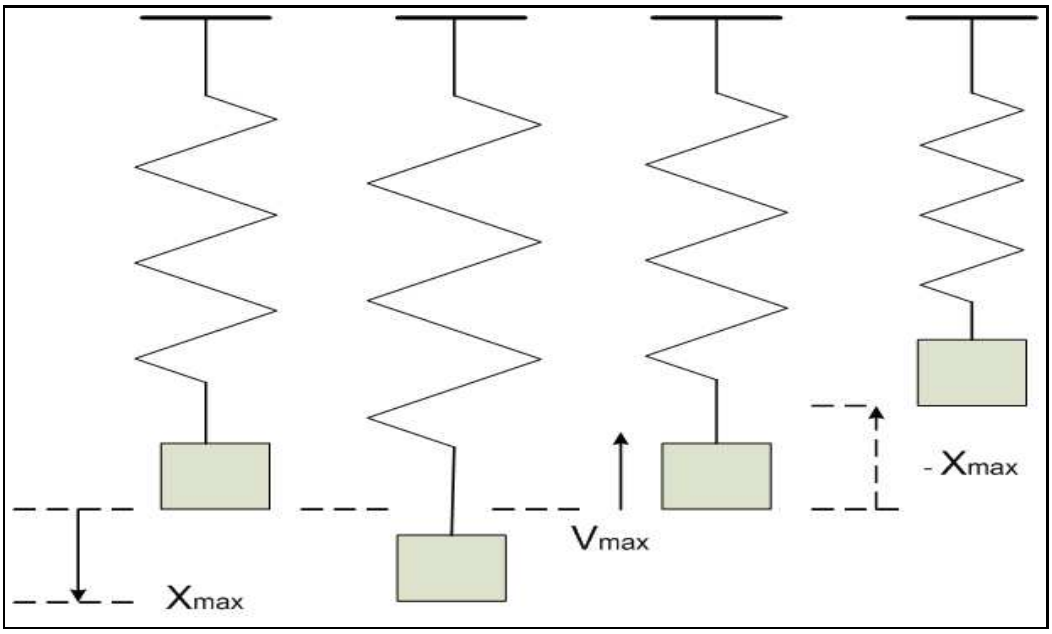

Figure 4.3: A Simple Spring-Mass System

Since **g** is acceleration due to gravity, we can write the above equation as

$$m\ddot{\mathbf{x}} = -k_s\mathbf{x}, \tag{4.10}$$

where $\ddot{\mathbf{x}}$ is the acceleration of the mass. This can be written as an ODE as

$$m\ddot{\mathbf{x}} + k_s\mathbf{x} = 0. \tag{4.11}$$

If the mass is displaced from its equilibrium position there is a net restoring force on the mass which tends to bring it back to equilibrium. However, in moving the mass back to the equilibrium position the mass acquires inertia, which keeps the mass moving beyond that position establishing a new restoring force, now in the opposite direction. This leads to oscillation of the mass about the mean position. In order to remove some energy from this system and to restore the system to its equilibrium we add a damping force $k_d\dot{\mathbf{x}}$ to the system. Here $\dot{\mathbf{x}}$ is the velocity of the mass. With this damping force, equation 4.11 can be written as

$$m\ddot{\mathbf{x}} + k_d\dot{\mathbf{x}} + k_s\mathbf{x} = 0. \tag{4.12}$$

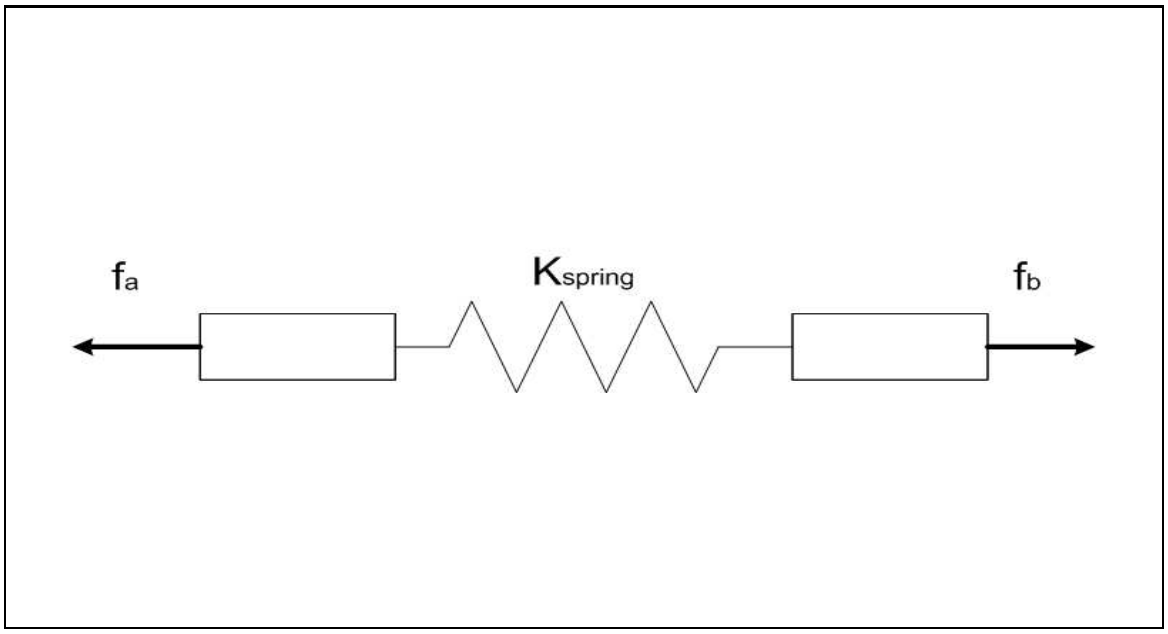

Figure 4.4: A Coupled Spring-Mass System

The above equation can be reduced to a system of first order equations of the form given by equation 4.1 by introducing additional variables, as was done for equation 4.2. Equation 4.12 can be re-written as, $m\ddot{\mathbf{x}} = -[k_d\dot{\mathbf{x}} + k_s\mathbf{x}]$ or further as,

$$\mathbf{f} = -[k_d\dot{\mathbf{x}} + k_s\mathbf{x}]. \tag{4.13}$$

For a coupled spring-mass system with a pair of particles at positions $\mathbf{a}$ and $\mathbf{b}$, the above equation leads to the forces acting on the particles being given by the equations:

$$\mathbf{f_a} = -\left[k_s(|\mathbf{l}| - r) + k_d\frac{\dot{\mathbf{l}}\cdot\mathbf{l}}{|\mathbf{l}|}\right]\frac{\mathbf{l}}{|\mathbf{l}|} \tag{4.14}$$

and

$$\mathbf{f_b} = -\mathbf{f_a}. \tag{4.15}$$

Here $\mathbf{f_a}$ and $\mathbf{f_b}$ are the forces on the particles $\mathbf{a}$ and $\mathbf{b}$ respectively and $\mathbf{l}$ is the difference in position of $\mathbf{a}$ and $\mathbf{b}$. Rest length is represented by r, k_s is the *spring constant* and k_d is the *damping constant*. Finally, $\dot{\mathbf{l}}$ is the difference in velocities of

the two particles. Thus it can be seen that equal and opposite forces are acting on the two particles along the line joining them (figure 4.4).

In our system, each orb web strand consists of a number of coupled spring-mass pairs. As can be observed from figures 3.2 and 3.4, the strands of a real orb web are very taut. One way to match this tautness of the web strands is to make the springs in our system stiffer by increasing their spring constants. However, the explicit Euler scheme that was discussed in section 4.1 requires an integration time step dt which must be inversely proportional to the square root of the spring stiffness. This criterion is known as the Courant condition [44]. This is the general problem of stiff sets of equations: stability can be achieved only at very small time scale with explicit schemes [21]. Thus, if we reduce the time step we will increase the overall time taken by the system to reach its equilibrium. We devised another solution to this problem that involved creating over-extended springs, by placing the masses at the ends of a spring at a distance more than the springs rest length. More details are given in section 5.1.

Another well known method of dealing with the stiff system of equations is the use of implicit numerical methods. An example of such a method is the implicit Euler integration. Unlike the explicit Euler method, which is based solely on conditions at the starting of a particular time step, the implicit Euler calculates the solution in terms of conditions at the end of the time step [2]. The implementation of such a solution is non-trivial and not recommended unless there is no alternative [1] .

Among the alternatives available in the literature are a wide range of more sophisticated explicit methods to solve an ODE with lesser error and using larger time-steps

```
typedef struct {
vector position;
vector velocity;
vector sum-forces;
double mass;
} a_mass;
```

Figure 4.5: Structure of a particle

```
typedef struct {
double rest_length;
double spring_constant;
masstype *mass_on_end1;
masstype *mass_on_end2;
} a_spring;
```

Figure 4.6: Structure of a spring

[6]. Perhaps the method most used is the *Runge-Kutta fourth order* method. In the Runge-Kutta method we replace the derivative of $\mathbf{x}$ at time t_0 by a weighed average of the derivative evaluated at four different points during the step interval h. Since the Runge-Kutta method of order four requires four evaluations per step, it gives a more accurate solution than the Euler method. The formula for the Runge-Kutta method of order 4 is given below.

$$k_1 = hf(\mathbf{x}_0, t_0) \tag{4.16}$$

$$k_2 = hf(\mathbf{x}_0 + k_1/2, t_0 + h/2) \tag{4.17}$$

$$k_3 = hf(\mathbf{x}_0 + k_2/2, t_0 + h/2) \tag{4.18}$$

$$k_4 = hf(\mathbf{x}_0 + k_3, t_0 + h)\ \mathbf{x}(t_0 + h) = \mathbf{x}_0 + 1/6k_1 + 1/3k_2 + 1/3k_3 + 1/6k_4 \tag{4.19}$$

In the implementation of this thesis, the numerical integration method used is the Runge-Kutta method of order four.

4.2.1 The Simulator

The simulator for the system consists of the system itself and an ODE solver. Care must be taken such that the system or the model on which the solver operates is independent of the solver. This makes it easy to change the solver if we so desire later and also makes it possible for us to reuse the solver code for another model.

The solver needs to be able to generate the new state of the system at each iteration by evaluating the ODE.

Our spring-mass system has three main parts: the masses, the springs and the forces acting on these masses. Since a mass has a position, a velocity, and is subject to various forces, the structure of the mass can be defined in C as is given in figure 4.5. Similarly, a spring which has a rest length, a spring constant and is connected to up to two other masses can be defined as shown in figure 4.6.

In order to determine the new state of the system, we first have to clear all the force accumulators to zero in each mass at the start of each iteration. Next we calculate the various forces acting on a particle: the spring force, the spring damping force, the gravitational force, air friction and any external force such as wind. Once we have summed all the forces acting on a particle, we use the numerical Runge-Kutta method to move the state of the system forward by a small amount of time, as given by the chosen time step.

However, before we start the simulation, the first step in the process of creating the complete orb web system is to lay out the various orb web strands as per the orb web modeling algorithm, which is detailed in the next section. Once we have created the orb web model we can then start the simulation process to arrive at successive stages of the state of the system. The creation of the orb web model is described in the next chapter.

Chapter 5

Web Construction And Animation

This chapter describes in detail the algorithms used to construct the orb web. It also presents the results of the orb web construction process and carries out a discussion of the presented results.

5.1 A Strand of Spider Web

One of the most important components of the orb web model is the single strand of the orb web. The orb web strand consists of a series of particles connected by Hookean springs. The success of the entire spring-mass orb web model depends on the stability of this structure.

In order to create an orb web strand we first need to arrive at the mass of a single particle. A *denier* is the weight, in grams, of 9000 meters of a single fiber strand. Spider silk has a tensile strength of about 5 denier [4], i.e., 9000 meters of spider silk weighs about 5 grams. For an estimated total length of 50 meters for an orb web, the weight of an orb web can be estimated to be about 25mg. With about 5000 particles in our system, we can arrive at an estimate for the mass of the particle as 10μg. Values for the *spring constant*, the *air friction constant* and the *time step* were chosen empirically and are given in section 5.2.1.

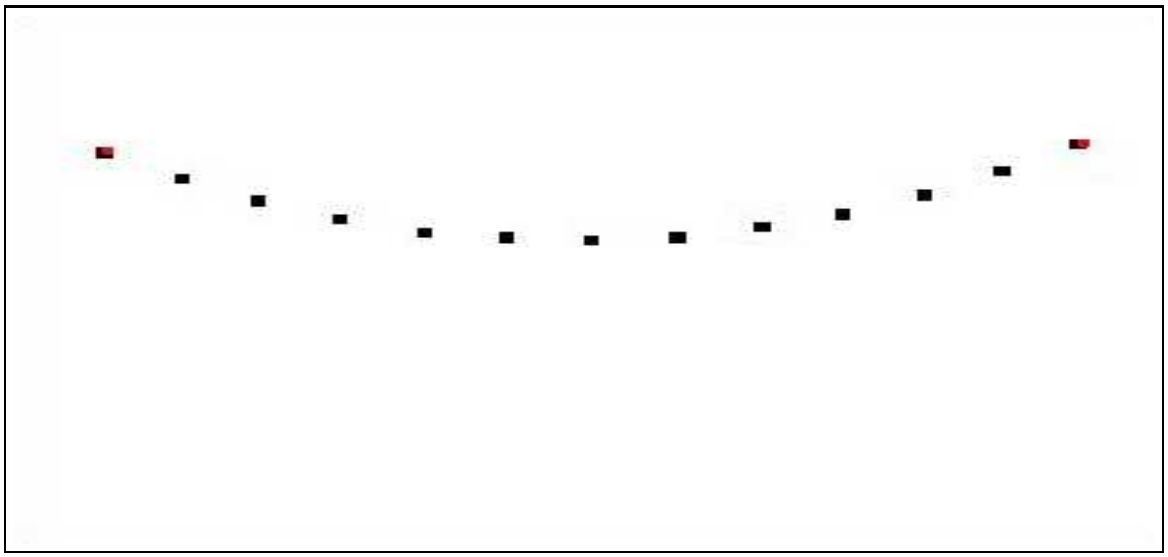

Figure 5.1: A spider web strand with springs at their rest lengths

To create a single strand of spider web, the spring-mass pairs were laid out in series, with each spring placed such that its length was equal to its *rest length*. This produced a system which was stable while at the same time the system settled down to its final state quickly (in about 5-10 seconds). A single strand of such a system is shown in figure 5.1. As can be observed from this figure, the strand shows a noticeable curvature when placed under gravity. This is not the case in a typical orb web strand, which is very taut.

Despite efforts directed at empirically finding a set of values for the system which would make the web strand as taut as a real spider web strand, such a set of values could not be found. Invariably, such efforts would run up against the *stiff* properties of a spring-mass system. Since it is desirable for the system to settle down as soon as possible it is not feasible to decrease the time step below a threshold value. Applying an implicit integration method to solve a system involving a large number of spring-mass pairs is a very complex task and not recommended [1] unless there is no other option and even then the desired performance might not be achieved.

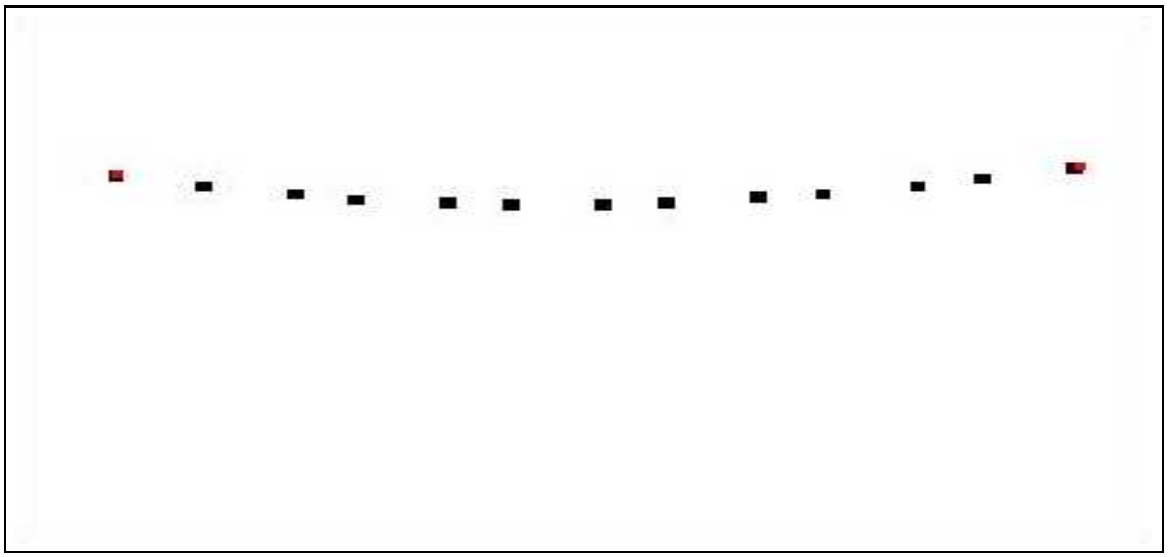

Figure 5.2: Same strand, but now constrained using the Provot method

One solution to this problem is laid out by Provot [22] and allows for very large time steps to be used. The Provot method involves applying a series of constraints to the springs after the integration has occurred. The key idea in these constraints is to consider the direction that the springs are moving to be correct, but to limit the deformation of each spring to a value consistent with empirical observations. Therefore in this method we iterate over all the springs in the model and if any of them have exceeded the maximum deformation we move the nodes at the ends of the springs closer together.

This method is not without disadvantages. First of all, by stepping away from the strictly physics-based simulation the solutions will only be qualitatively correct. In addition, the procedure is not proven to work in all cases; worse yet, it is dependent on the order the springs are examined and correcting one spring may overextend another, and there is never guaranteed to be a correct traversal order. The procedure works in most cases because often the result of shortening one spring and lengthening another is to propagate deformations throughout the spring-mass structure, but

nevertheless over-extended springs can still manifest, as is evident in figure 5.2. The results are better when the procedure is run multiple times in a row, but despite the assertions of the author of the method, the process was not found to always converge to a completely non-extended state, even in common situations.

In this thesis we have employed another simple solution in practice after having studied the feasibility of and testing some of the published solutions. Our solution simply involves laying the springs out at length which is 75% to 250% more than their rest lengths. The whole idea behind this approach is that instead of trying to create super stiff springs out of the current highly elastic ones by somehow increasing the spring constant, while keeping the simulation fast enough, the springs are initially placed in an highly over-extended state instead of rest lengths. When these spring-mass pairs with highly extended springs are subjected to gravity they do not extend by a large amount. This enables us to get a web strand out of such a system of spring-mass pairs with the ability to make the strand as taut as required. Surprisingly, such a system of spring-mass pairs is quite stable. We were able to achieve desired tautness of the strand of spider web with a settling time for the system being around 5 to 10 seconds.

5.2 Web Construction

This section describes the process used to develop the overall structure of the orb web. The whole process of orb web creation is divided into a number of stages. First, the *outer frame* and the *inner frame* are created. This is followed by the creation of the *hub* and the *radial spokes*. Finally, on this structure is overlayed the *spiral*. To

assist in understanding the orb web construction process we have presented images and *pseudo-code* to supplement the explanatory text. Pseudo-code is a generic way of describing an algorithm without the use of any specific programming language syntax.

As a first step in the orb web creation process, the user chooses the attachment points for the orb web. Currently the user can choose up to ten attachment points. These user-chosen attachment points serve as the outer most periphery of the orb web. In nature, these points would have represented the points where the spider would attach its web to a tree or a bush. Spiders tend to place their webs so that the air current would pass quickly through the web; this would increase the likelihood of airborne prey being caught in the web.

The first section of web developed was the inner frame. Once the attachment points have been chosen by the user, a point is chosen such that this point is roughly in the center of the web. Figure 5.3 shows the center point, marked *C*. This center point is used as a reference point to create the inner frame and inside this inner frame the web is constructed. Figure 5.5 gives the pseudo-code for the construction of the inner frame. To get the points for the inner frame, we start with the locations of the user chosen attachment points and move them towards the center reference point by a fixed amount. These inner frame points are shown in the figure 5.4, and marked *I*. Next we create the strands between the inner frame points so as to create the outer shape and structure of the web.

The next section of the web to be constructed is the outer frame. The outer frame is defined as the set of strands connecting the user chosen attachment points to the

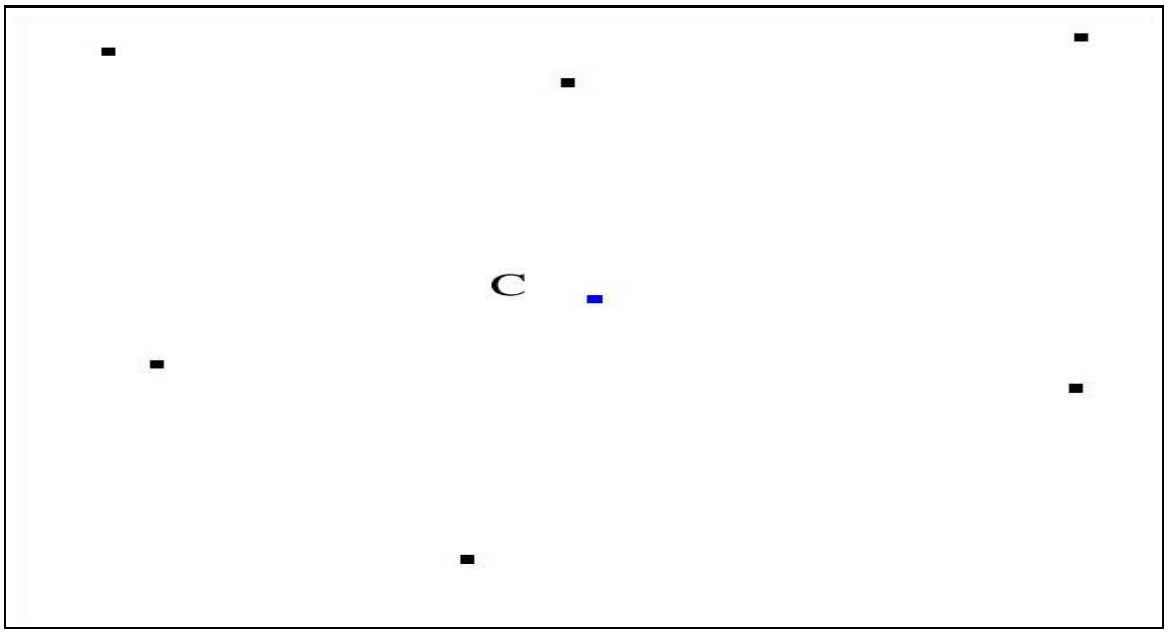

Figure 5.3: The "center" point (marked with a "C")

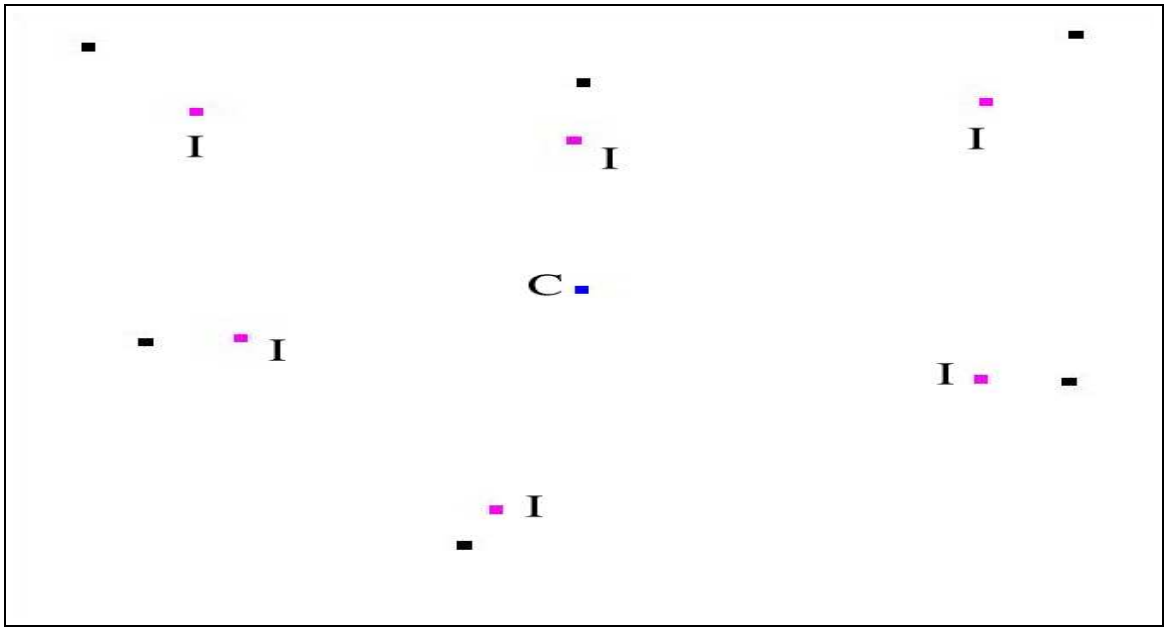

Figure 5.4: The "inner frame"(marked with "I")

```
1. frame_attach_locations[] <-- get attachment points from user
2. xmin, ymin <-- find smallest co-ordinates in frame_attach_locations[]
3. xmax, ymax <-- find largest  co-ordinates in frame_attach_locations[]
4. find center: x_center = (xmin + xmax)/2
                y_center = (ymin + ymax)/2
5. get inner frame points,
   frame_inner[] = move frame_attach_locations[] points towards
                   x_center, y_center by a fixed amount, MOVE_OUTER.
6. calculate distance between corresponding outer and inner frame points
7. - create strands by laying spring-mass pairs from outer to
     inner frame points
   - each mass is laid at a distance of 5 units from other
    /* first create the masses */
    frame_mass[frame_mass_count].position.x = mass x position
    frame_mass[frame_mass_count].position.y = mass y position
    /* set initial velocity to zero */
    frame_mass[frame_mass_count].velocity.x = 0
    frame_mass[frame_mass_count].velocity.y = 0
    frame_mass[frame_mass_count].mass = 0.00000001
    frame_mass_count = frame_mass_count + 1
    /* now create the springs between the masses */
    frame_spring[frame_spring_count].rest_length = 2
    frame_spring[frame_spring_count].spring_constant = 0.001
    frame_spring[frame_spring_count].end1 = 1st mass
    frame_spring[frame_spring_count].end2 = 2nd mass
8. mark the outer most masses in the outer frame as fixed
9. calculate distance between consecutive inner frame points
10. - create strands by laying spring-mass pairs between
       inner frame points
    - each mass is laid at a distance of 20 units from other
    - spring rest length = 15 units
    - spring constant = 0.001 N/m
```

Figure 5.5: Inner and Outer Frame Construction

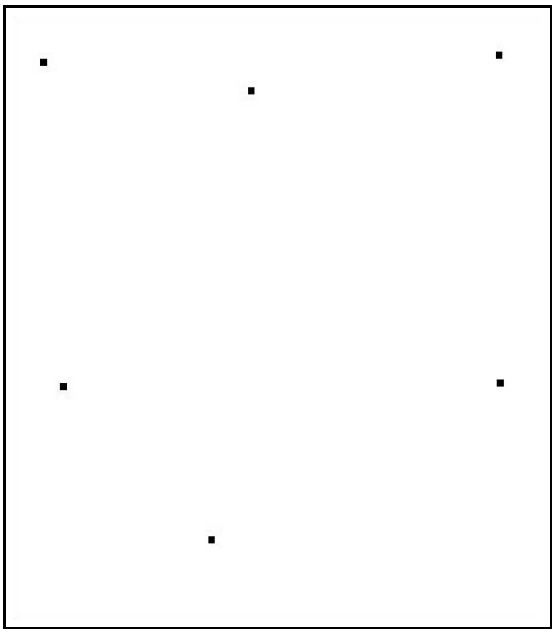

Figure 5.6: The User Chosen Attachment Points

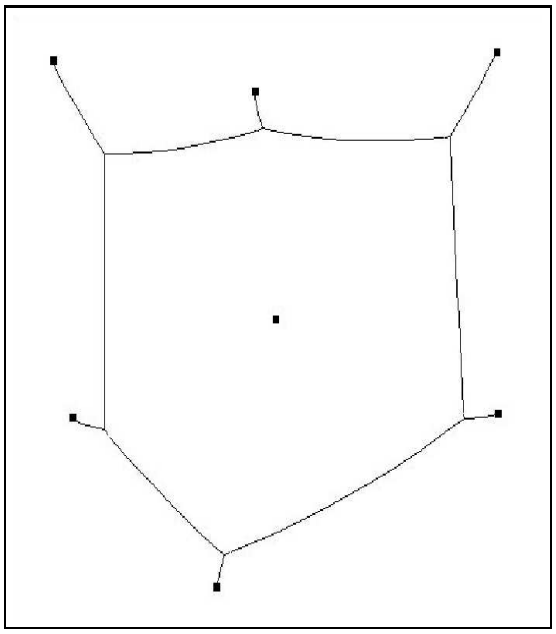

Figure 5.7: The Orb Web Frame

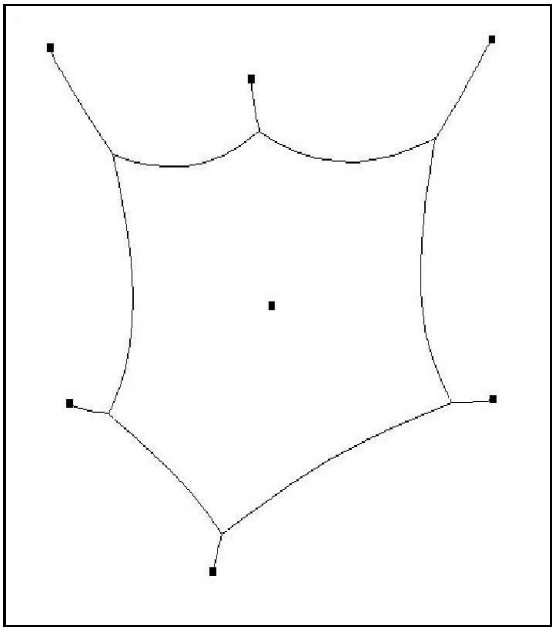

Figure 5.8: The Constrained Orb Web Frame

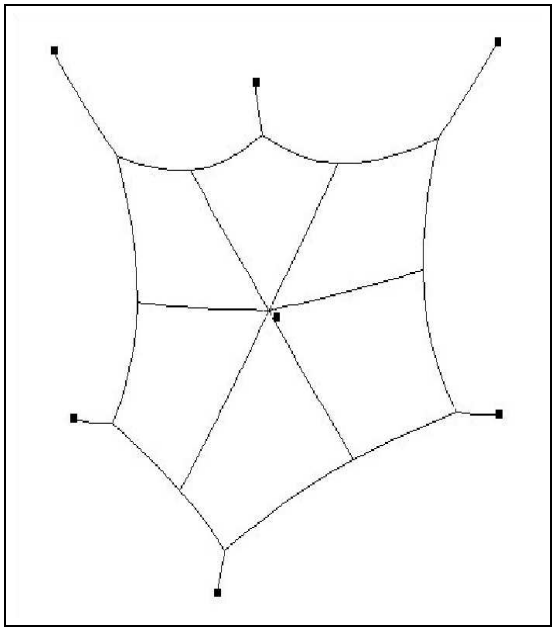

Figure 5.9: The Primary Orb Web Spokes

inner frame and is created by attaching the user chosen attachment points to the inner frame by means of short stiff strands. Figure 5.5 gives the pseudo-code for the construction of the outer frame. The orb web is constrained such that the masses at the user chosen attachment points are fixed and do not change their position when the simulation is started. The length and stiffness of these strands from the inner frame to the user attachment points can be altered independently of the rest of the orb web.

Once we have the initial inner frame for the orb web constructed (figure 5.7), we can see that the curvature of the frame does not resemble the curvature of the frame of a real orb web (figure 3.2 or figure 3.4). Also it might be expected that the radial spokes of the web when created would exert a force that would pull the inner frame inwards. This does happen, but to further aid and to tailor the curvature, the inner frame of the web is constrained by inward radial forces. The result of this process is shown in figure 5.8. As can be seen, the frame now resembles the natural curvature of a spider web frame more closely.

This is followed by the creation of *primary radial spokes* (figure 5.9). These initial few spokes serve as reference points for the rest of the orb web creation process, provide structural support to the orb web and also serve as the attachment points for the hub. The primary radial spokes are created by drawing a strand from a section of the orb web's inner frame to a section of the inner frame opposite it. The pseudo code for this process is given in figure 5.5. To create a radial spoke, we start with the first inner frame segment and find the mass which lies in its center, say *frame_middle_mass* (see figure 5.10). Next we iterate through the rest of the inner

frame segments and find the mass which is farthest from *frame_middle_mass*, which is *farthest_mass*. Finally, we draw a web strand between *frame_middle_mass* and *farthest_mass*. This process is then repeated for the other inner frame segments.

Once the primary radial spokes have been created, the *hub* is then created at the intersection of the primary radial spokes. To create the hub we first choose a mass in the first primary radial spoke which is close to the intersection point of the primary radial spokes, *mass_close_intersection*. Now in the second radial spoke choose a mass closest to *mass_close_intersection*, *closest_next_mass*. Then we lay a strand of the web from *mass_close_intersection* to *closest_next_mass*. Repeat the above process for other primary radial spokes to get the complete hub.

The *hub* can now be used to attach the *secondary radial spokes* (see figure 5.12). To create the secondary radial spokes we start with the first mass in the inner frame, *inner_frame_first_mass*, and find the mass in the hub which lies closest to this inner frame mass, *hub_mass*. Next we create a strand between *inner_frame_first_mass* and *hub_mass*. This process is then repeated for all the masses in the inner frame, leading to the construction of the secondary radial spokes which are connected randomly to the masses in the hub.

The most important and probably the most complex problem is to devise an algorithm for the spiral. The spiral has to have a small amount of randomness in it and also the spiral should fill in the available web area such that the whole web does not look overly symmetric. The next paragraph presents the naive algorithm which was used at first to achieve the spiral construction and the paragraph after it presents an evolved version of this naive algorithm which was used in the thesis

```
/* first create the primary radial spokes */
1. frame_middle_mass[] <-- get middle masses of all inner frame segments
2. farthest_mass[] <-- masses farthest from each corresponding
                       middle mass in frame_middle_mass[]
3. - create strands by laying spring-mass pairs between frame_middle_mass[]
     and corresponding mass in farthest_mass[]
   - each mass is laid at a distance of 7 units from other
   - spring rest length = 3 units
   - spring constant = 0.01 N/m
/* now create the hub */
4. mass_close_intersection <-- mass in first radial spoke close to
                               intersection of primary radial spokes
5. closest_next_mass <-- mass in second radial spoke closest
                         to mass_close_intersection
6. create strand between: mass_close_intersection and closest_next_mass
7. - repeat steps 4, 5 and 6 for other radial spokes
   - stop when you reach back at the first radial spoke
/* improve the orb web shape */
8. apply inward radial force on inner frame segments
/* create the secondary radial spokes */
9. inner_frame_first_mass <-- first mass of inner frame
10. hub_mass <-- mass closest to inner_frame_first_mass, in the hub
11. create strand between: inner_frame_first_mass and hub_mass
12. repeat steps 9, 10 and 11 for all inner frame masses
```

Figure 5.10: Hub and Radial Spokes Construction

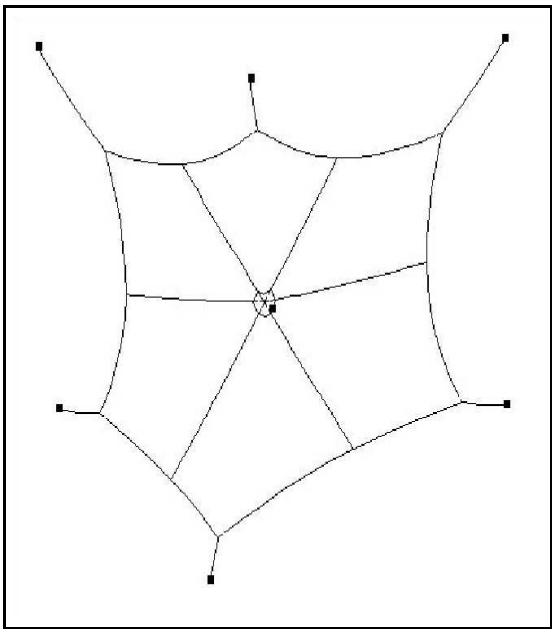

Figure 5.11: The Hub

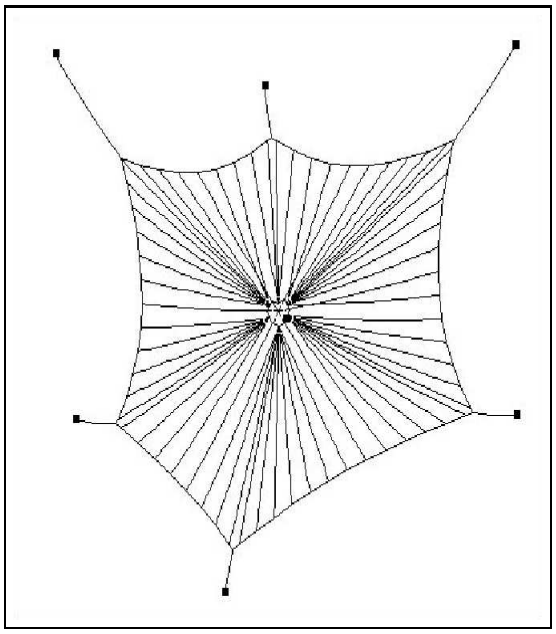

Figure 5.12: The Orb Web Spokes

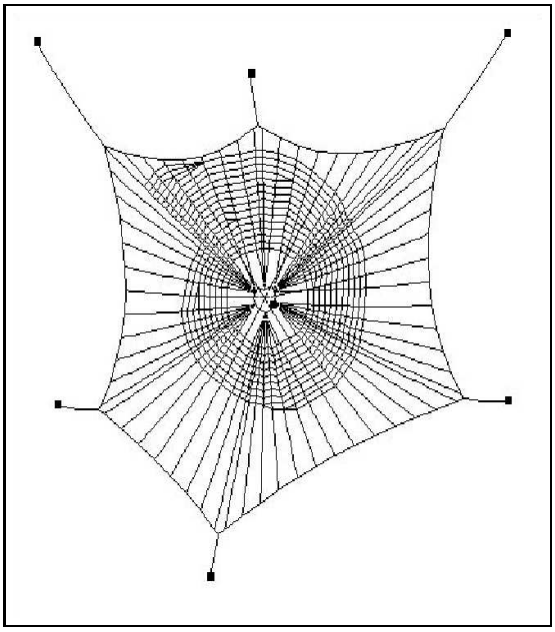

Figure 5.13: The Initial Spiral

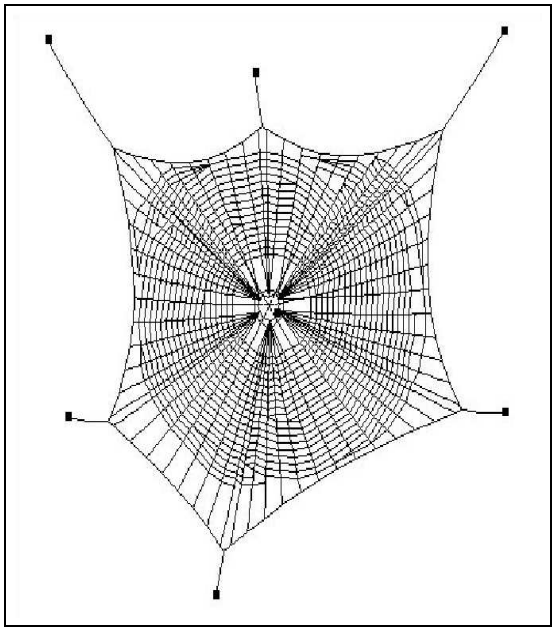

Figure 5.14: The Final Spiral (After Processing)

```
/* create the initial spiral */
for all masses in first spoke
(start with mass closest to inner frame)
{
  initial_mass <-- get current mass
  for all spokes
  {
    next_mass <-- mass closest to initial_mass on next spoke
    create strand between: initial_mass and next_mass
    initial_mass <-- next_mass
  }
}
```

Figure 5.15: Initial Spiral Construction

implementation.

One particular process by which we might create the spiral involves connecting a mass on one spoke to the mass on the following spoke, starting with the masses closest to the hub. At the end of each spiral we choose a mass closer to the outer frame (i.e., we take a step outwards). This process is then repeated until we reach near the inner frame. The algorithm described above, although simple, has the drawback that it is too uniform. Further, the stopping condition for the spiral (i.e., stop the spiral when you are close to the frame) did not produce desirable results. Usually the spiral stopped too soon and there were large gaps in the orb web near the inner frame. In order to improve the results of this process we evolved the algorithm described below. It has the advantages of being simple and of producing excellent results.

The algorithm for the creation of initial spiral is given in figure 5.15. The creation of the *initial spiral* is done by starting with the mass on the first spoke which is closest to the inner frame, *initial_mass*. Then we find the mass on the second radial spoke

```
/* create the processed spiral */
1. max_spoke <-- spoke with maximum number of masses
                 not covered by a spiral
2. for all uncovered masses in max_spoke
   (start with mass closest to inner frame)
   {
     initial_mass <-- get current mass
     for all spokes
     {
       next_mass <-- mass closest to initial_mass on next spoke
       create strand between: initial_mass and next_mass
       initial_mass <-- next_mass
     }
   }
3. repeat steps 3 if number of uncovered masses
   in any spiral > SPIRAL_DENSITY (preset value:5)
```

Figure 5.16: Processed Spiral Construction

which is closest to *initial_mass*, *next_mass* and create a web strand between these two masses. This process is repeated for all the spokes in the web. In the next iteration we start with the next mass down the first spoke (i.e., *initial_mass + 1*) and repeat the above process. We end this process a little distance away from the hub of the web so that a distinct hub is still visible when the process for creating the spiral is over. The result of this process can be seen in figure 5.13.

We follow the creation of the *primary spiral* by a *processed spiral* so as to cover the area of the web where there is still space for the spiral to be drawn without necessarily covering the whole web with the spiral. The algorithm for this process is given in figure 5.16 and is essentially the same as that used to create the initial spiral, but now instead of starting with the first spoke in each iteration we start with the spoke that has the greatest number of masses not covered by the spiral, *max_spoke*. The result of this process can be seen in figure 5.14. The amount of web

to be covered can be controlled with ease by altering the selection of the starting and/or ending masses. The density of the spiral can also be controlled by altering the number of masses in the radial spokes. A larger number of masses in the radial spokes would lead to a denser spiral and vice-versa.

5.2.1 Implementation Details

Given below is the list of parameters used in modeling the orb web and the values assigned to them.

Mass of a particle	10μg
Number of particles	2000
Diameter of the orb web	up to 1 meter
Total mass of the orb web	25mg
Spring Constant	0.01 N/m
Air Friction Constant	0.00000001 Ns/m
Time Step	0.02 seconds
Rest Length	3mm

5.3 Results and Discussion of the Results

In this section the various results from this thesis work are presented and a discussion of the results is carried out.

5.3.1 Orb Webs of Varying Sizes

The four images in figure 5.17 show orb webs of varying sizes. The standard canvas size in all these images was kept constant at 600 by 600 pixels and all other parameters are kept constant. There are some interesting observations that can be made from these images. As the size of the orb web is decreased, the number of radial spokes in the orb web decreases. Also evident is the decrease in density of the orb web spiral with the decrease in orb web size. Finally, it is observed that the empty space around the spiral increases with the decrease in size. All these observations can be attributed to the fact that the parameters governing the number of spokes and spiral density are initially optimized to an orb web which occupies the whole canvas. With the decrease in size of the orb web the space available for creating the orb web decreases but the parameters are still optimized for a bigger canvas.

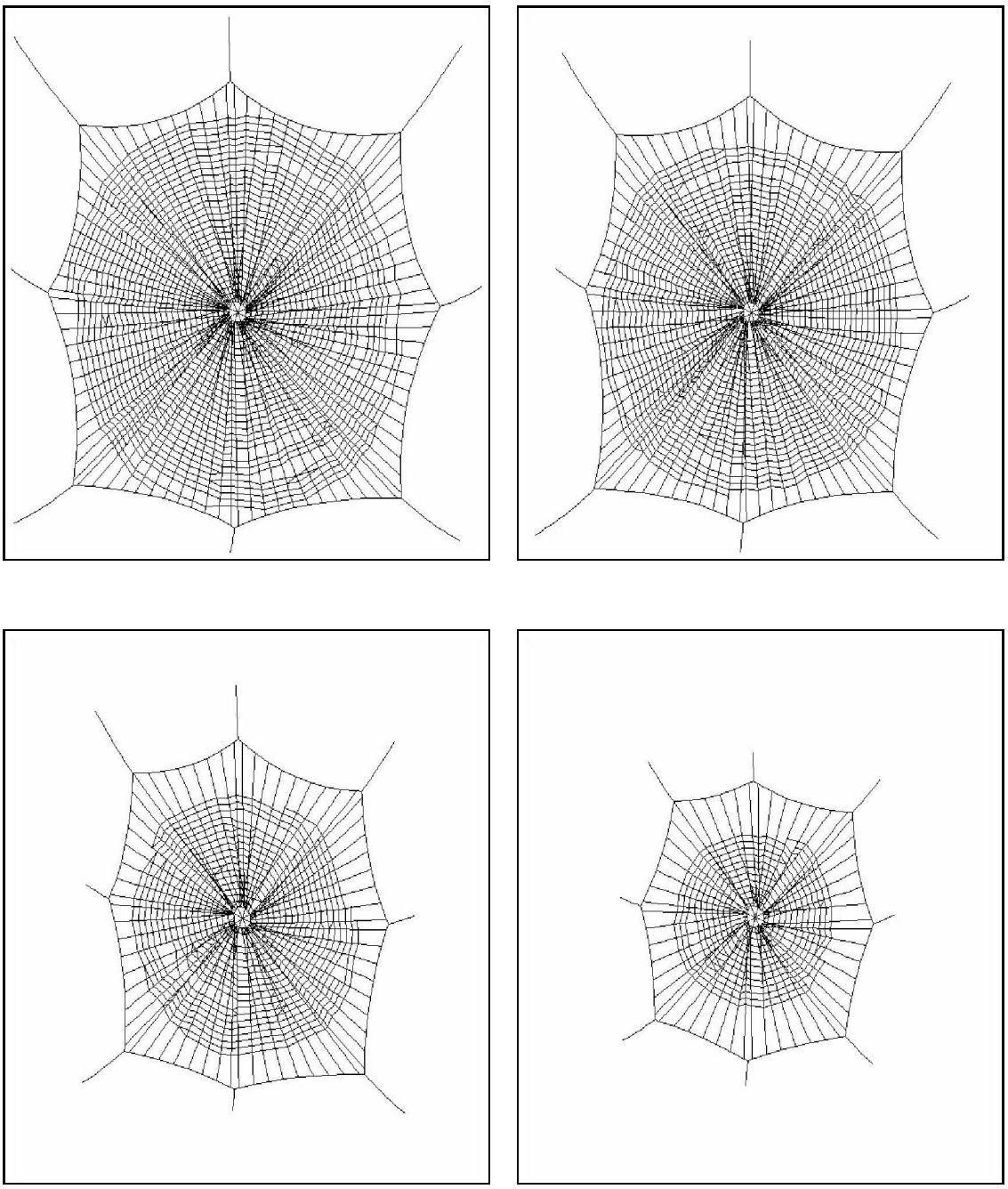

Figure 5.17: Orb Webs of Varying Sizes

5.3.2 Orb Webs of Varying Shapes

The four images in figure 5.18 show orb webs of various shapes. Although we had concentrated on creating an orb web model which would produce images resembling the commonly observed orb web shapes, the images in figure 5.18 show the rich set of results that can be obtained. The first two images show a triangular shaped orb web. The next two images show orb webs shapes which might be more commonly observed. While creating an orb web shape, we can tailor the images by altering the number and location of the attachment points. For instance, if a particular section of the orb web is found to be sagging too much under gravity, we can add an additional attachment point and thus provide further support to the orb web structure. These images can be further tuned to the desired look by altering the parameters that control the spiral density and the number of spokes.

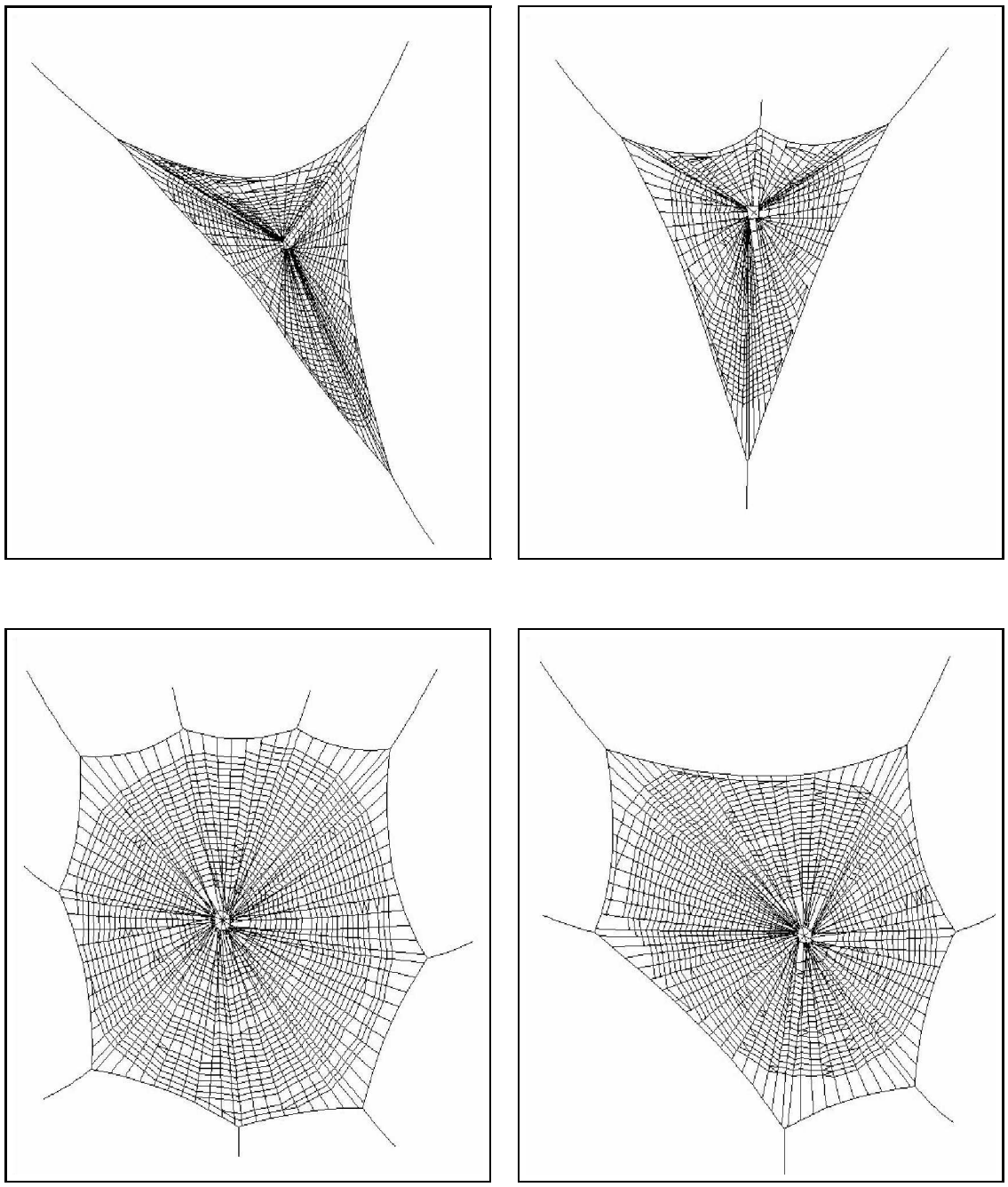

Figure 5.18: Orb Webs of Varying Shapes

5.3.3 Orb Webs With Varying Spring Constants

The images in figure 5.23 show orb webs in which the spring constant is varied while all other parameters are kept constant. The first image (figure 5.19) shows the orb web with all the original values of the three spring constants (i.e., the outer frame spring constant set to 0.001 N/m, the inner frame spring constant set to 0.001 N/m and the radial spokes spring constant set to 0.01 N/m).

In figure 5.20, the spring constant of the outer frame springs is reduced by a factor of 10 to 0.0001 N/m. The effect of this lower spring constant value is very evident in this image. The outer frame, which connects the user chosen attachment points to the inner frame, is highly elongated compared to the frame seen in figure 5.19.

The next image (figure 5.21) captures the effect of reducing the spring constant value for the inner frame by a factor of 10 to 0.0001 N/m. It can be observed that the inner frame is no longer as taut as in figure 5.19. Further, this elongation of the inner frame is much less as compared to the elongation of the outer frame in figure 5.20. This can be attributed to the better distribution of the weight of the orb web in the case of the inner frame.

The final image (figure 5.22) shows the effect of reducing the spring constant value of the radial spokes by a factor of ten to 0.001 N/m. Although the radial spokes are more curved than the radial spokes in figure 5.19, they are the least affected by reduction of the spring constant. This can be attributed to better distribution of the stress among the numerous radial spokes as compared to the inner or the outer frame.

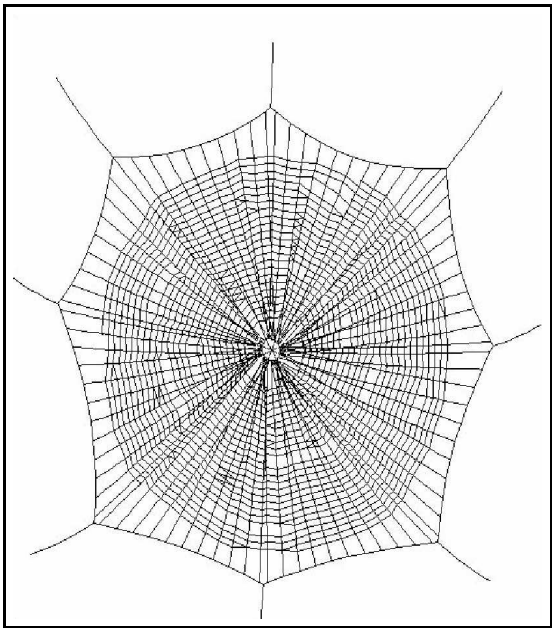

Figure 5.19: Original Spring Constants

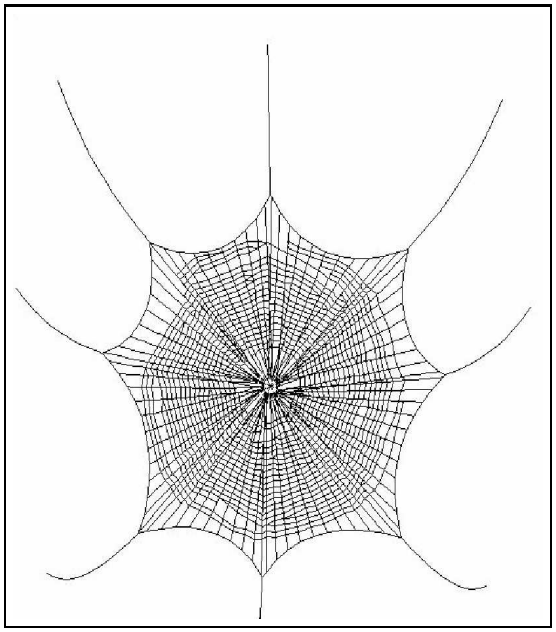

Figure 5.20: Outer Frame Spring Constant Reduced

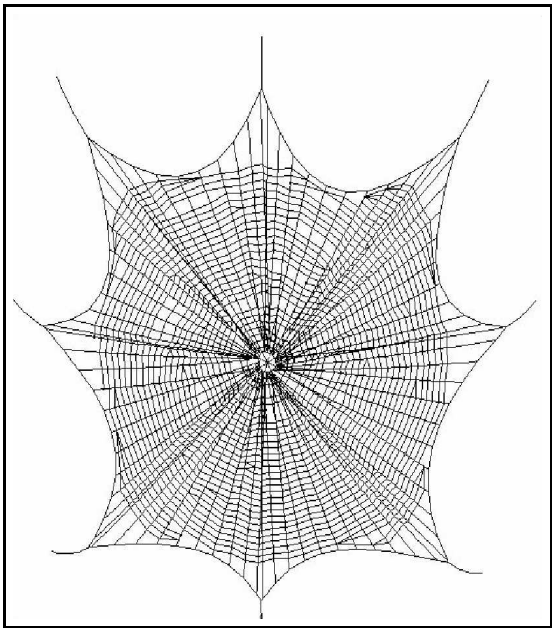

Figure 5.21: Inner Frame Spring Constant Reduced

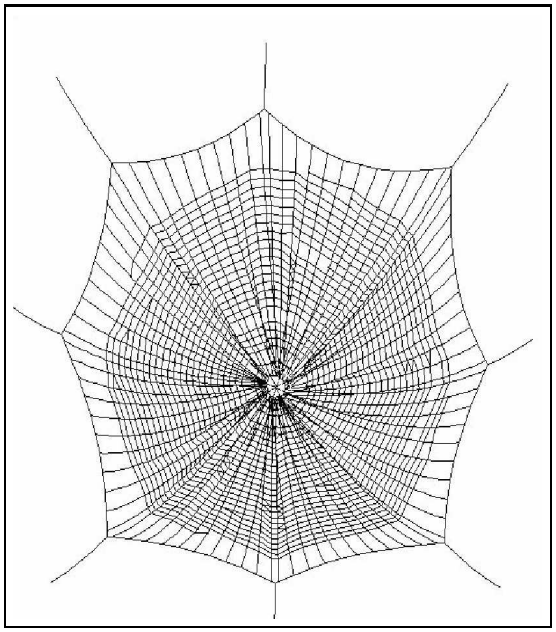

Figure 5.22: Radial Spokes Spring Constant Reduced

Figure 5.23: Effect of Changing Spring Constant Values

5.3.4 Orb Webs With Varying Spirals

The four images in figure 5.28 are examples of structurally similar orb webs whose spirals have been created using different parameters. The two parameters used to control the density of the spiral are: the rest length of the springs in the radial spokes and the placement length of the springs in the radial spokes. The placement length of the springs gives the distance between two consecutive masses on a radial spoke. As a result, if we increase the placement length of the springs , i.e., the distance between two consecutive masses on a radial spoke, this will result in fewer masses in the radial spokes. Since the spiral is made by connecting a mass in a radial spoke to a mass in the following radial spoke, fewer masses would result in a spiral which is less dense. The rest length of the springs has to be changed along with the placement length in order to keep the coupled spring-mass system stable.

The image in figure 5.24 shows an orb web with the preset spiral parameters. Its spirals are quite symmetric with the spirals covering a large portion of the web structure. The image in figure 5.25 shows a similar orb web with a spiral that is less dense and does not cover as much of the web area as does the spiral in figure 5.24. The orb web in figure 5.26 not only has a spiral which is less dense, but has a much bigger gap in the hub or the central portion of the web. Finally, the orb web in figure 5.27 has a very sparse spiral.

The table below shows the parameters and their values, which were used to generate the images shown in figure 5.28. The rest length and the placement length are of the springs in the radial orb web strands.

Image	Rest Length	Placement Length
figure 5.24	3mm	7mm
figure 5.25	3mm	9mm
figure 5.26	5mm	11mm
figure 5.27	7mm	15mm

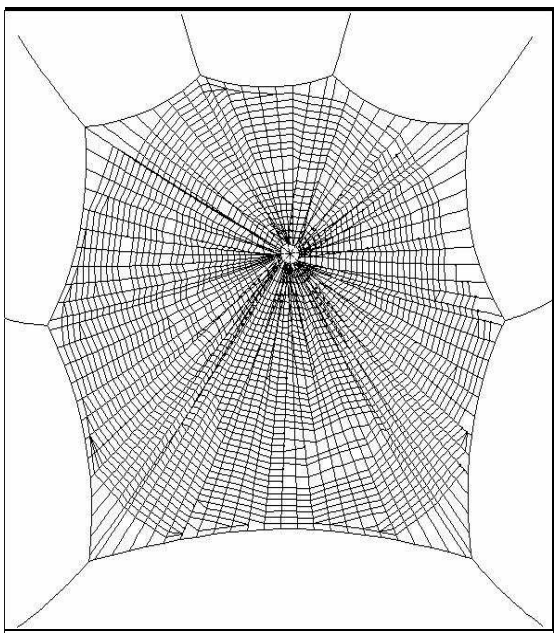

Figure 5.24: Spiral With Preset Parameters

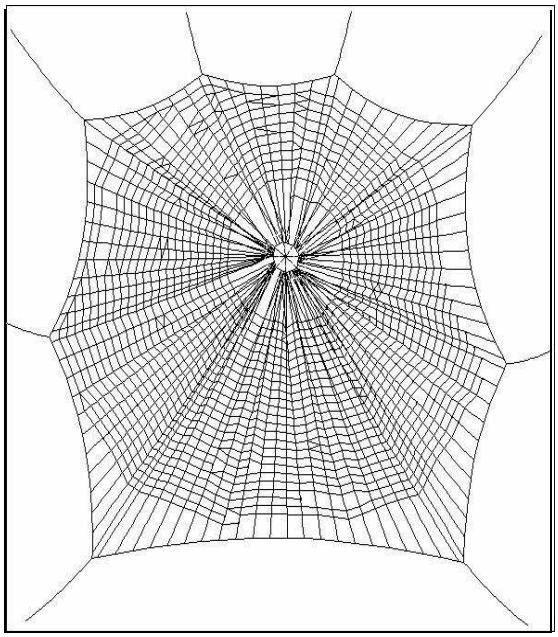

Figure 5.25: Less Dense Spiral

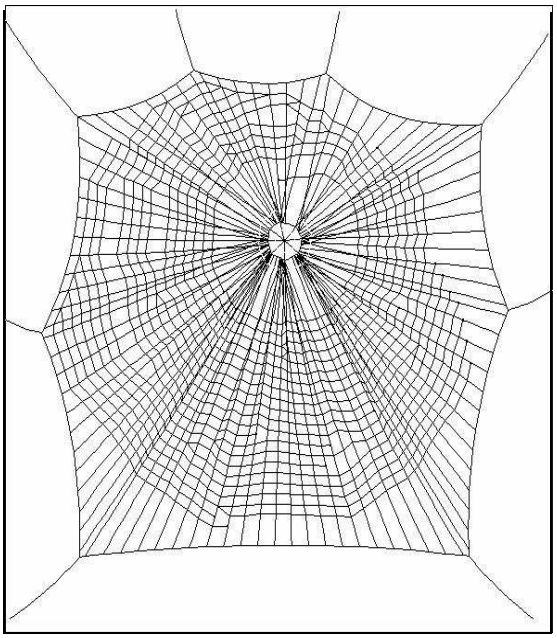

Figure 5.26: Less Dense Spiral With Less Web Coverage

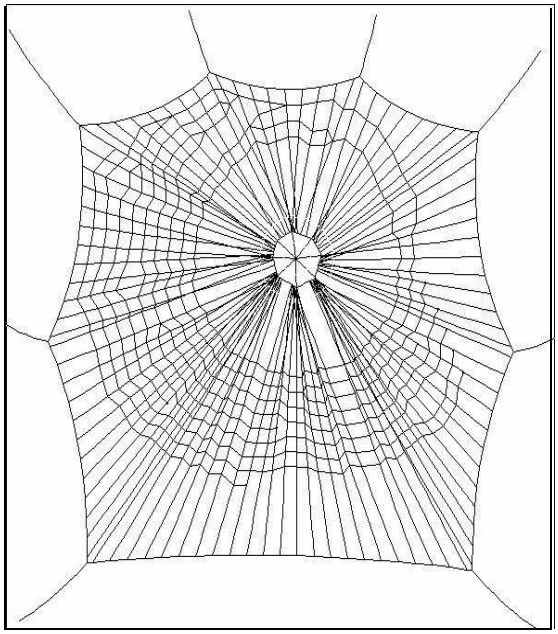

Figure 5.27: Least Dense Spiral With Least Web Coverage

Figure 5.28: Varying Spirals

5.3.5 Orb Webs With Varying Radial Spokes

In the next four images, shown in figure 5.33, the number of radial spokes that form the structure on which the spiral is overlayed is varied. The two parameters used to control the number of radial spokes are: the rest length of the springs in the frame and the placement length of the springs in the frame. The placement length of the springs gives the distance between two consecutive masses on a frame strand segment. As a result, if we increase the placement length of the springs, i.e., the distance between two consecutive masses on a frame strand segment, this will result in fewer masses in the frame strand segment. Since the radial spokes are made by connecting the masses on the frame to the hub, fewer masses in the frame will result in fewer radial spokes. The rest length of the springs has to be changed along with the placement length in order to keep the coupled spring-mass system stable.

As usual, the first image in figure 5.29 shows an orb web created with the preset parameters. In the next image in figure 5.30, the number of radial spokes is reduced while keeping all the other parameters the same. In comparison to the image in figure 5.29, the second image in figure 5.30 has a larger spacing between the radial spokes. This space between consecutive radial spokes is further increased in figures 5.31 and 5.32.

The table below shows the parameters and their values, which were used to generate the images shown in figure 5.33. The rest length and the placement length are of the springs in the orb web frame strands.

Image	Rest Length	Placement Length
figure 5.29	15mm	20mm
figure 5.30	25mm	30mm
figure 5.31	35mm	40mm
figure 5.32	50mm	57mm

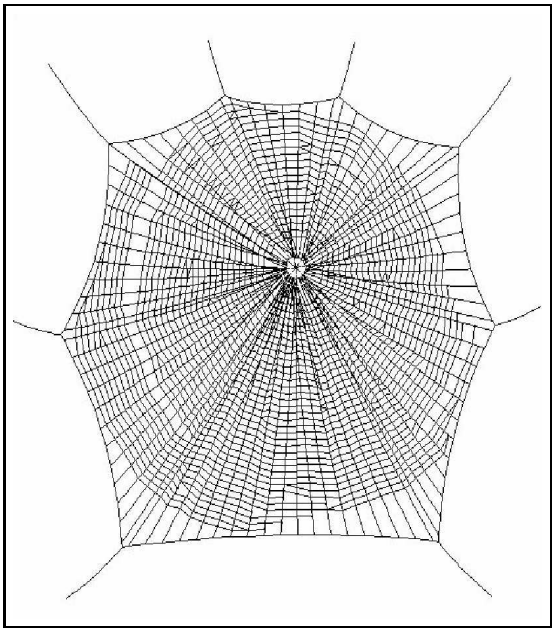

Figure 5.29: Spokes Created With Preset Parameters

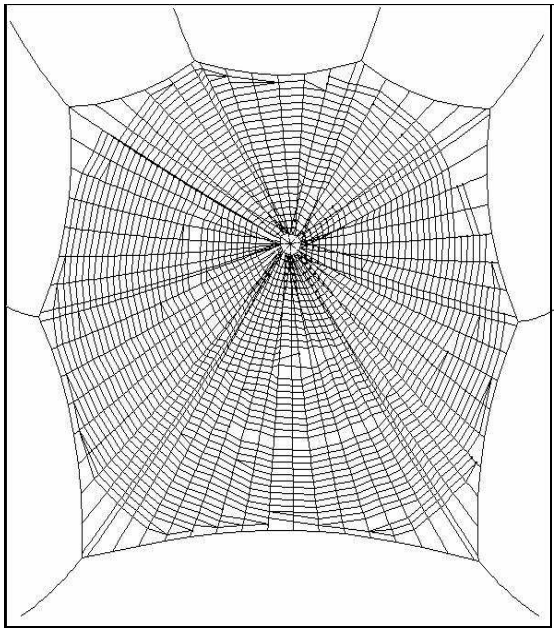

Figure 5.30: Lesser Number of Spokes 1

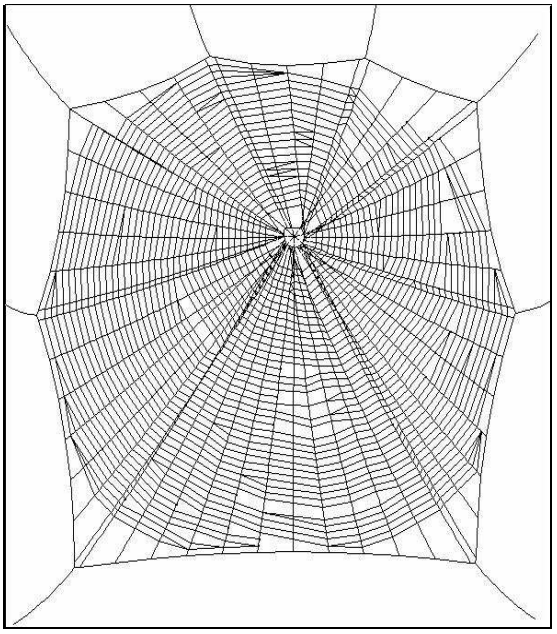

Figure 5.31: Lesser Number of Spokes 2

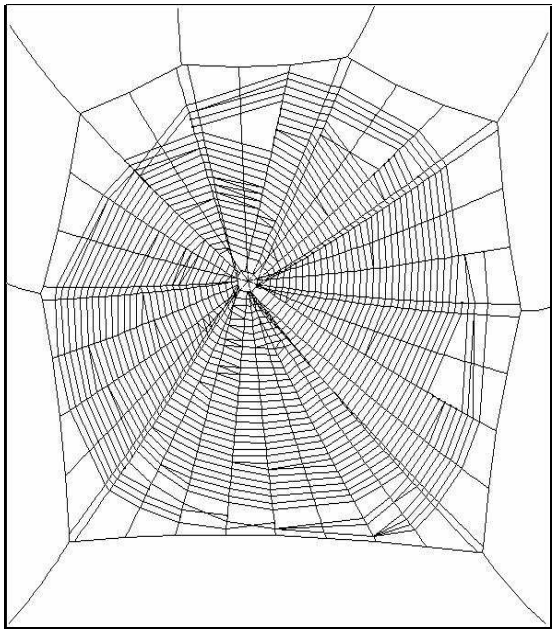

Figure 5.32: Least Number of Spokes

Figure 5.33: Varying the Number of Radial Spokes

5.3.6 Orb Webs With Varying Radial Spokes and Varying Spirals

In the next four images, shown in figure 5.38, the effect of varying both the spiral density and the number of radial spokes is shown. In other words these images combine the effects demonstrated separately in section 5.3.4 and in section 5.3.5. In the images shown, the density of spiral and the number of spokes are both decreased in each successive image.

The table below shows the parameters and their values, which were used to generate the images shown in figure 5.33.

Image Number	Rest Length (Spoke)	Placement Length (Spoke)	Rest Length (Frame)	Placement Length (Frame)
figure 5.34	3mm	7mm	15mm	20mm
figure 5.35	3mm	9mm	25mm	30mm
figure 5.36	5mm	11mm	35mm	40mm
figure 5.37	7mm	12mm	40mm	45mm

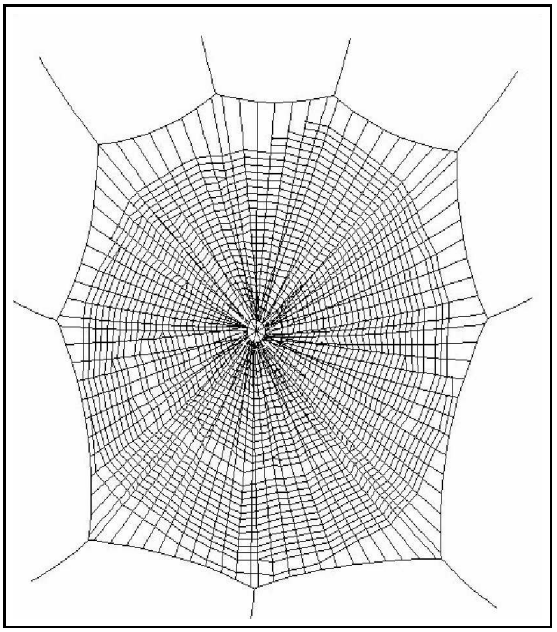

Figure 5.34: Image with Preset Parameters

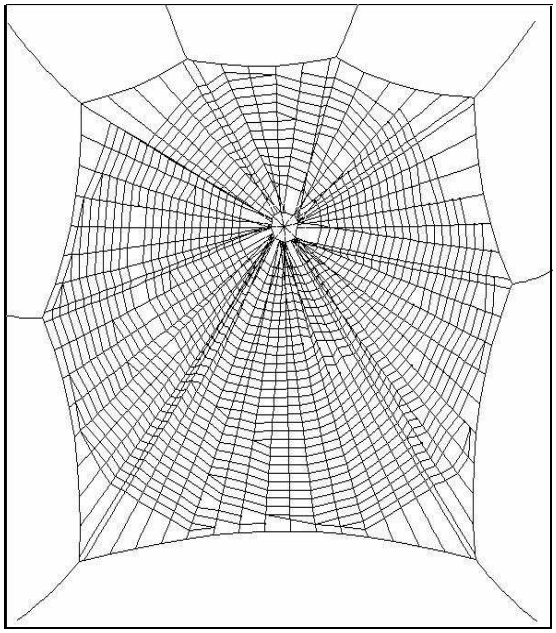

Figure 5.35: 1: Fewer Spokes and Less Dense Spiral

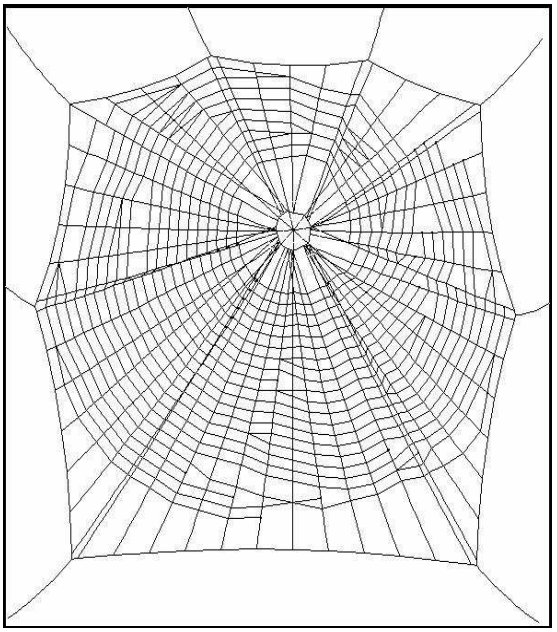

Figure 5.36: 2: Fewer Spokes and Less Dense Spiral

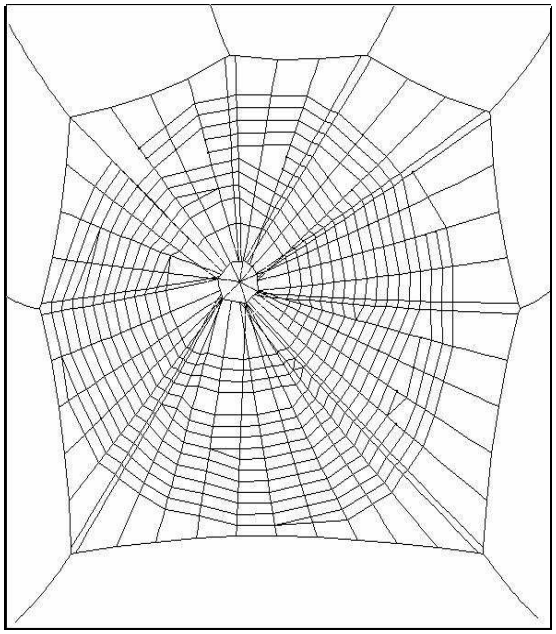

Figure 5.37: Least Spokes and Least Dense Spiral

Figure 5.38: Varying the Number of Radial Spokes and Spirals

5.3.7 The Orb Web Against Different Backgrounds

The next few images show the orb web against different backgrounds. The background image has been texture mapped to a polygon which covers the entire screen. The orb web itself is created a little distance ahead of the background image and thus is superimposed on the background image. In order to increase the visibility of the orb web against the background, the orb web has been shaded with uniform black color.

Figure 5.39: Orb Web Against Different Backgrounds

5.3.8 Effect of Shading and Alpha Blending

To further improve the quality of the results shown in figure 5.39, in which the orb web was shaded black, the orb web is shaded using different shades of grey. From the images of real orb webs it was observed that the strands of the web exhibit, in general, different shades of gray. Figure 5.44 shows an orb web against the same background but with different shading schemes for the orb web. The figures on the left have been created using different shades of grey but no alpha blending. The corresponding images on the right have the same shading scheme as the image on its left, but has alpha blending enabled. The shading and alpha values for each image are given in the table below. The shade value is described by three color components, red, green and blue, and each color component can range from 0.0 to 1.0 where 0.0 is no color and 1.0 is maximum color. The background image chosen has a lighter background to increase visibility and to help demonstrate the above effects with clarity.

Image	Shade (Red, Green, Blue)	Alpha
figure 5.40	(0.86,0.88,0.87)	0
figure 5.41	(0.86,0.88,0.87)	0.7
figure 5.42	(0.69,0.71,0.73)	0
figure 5.43	(0.69,0.71,0.73)	0.7

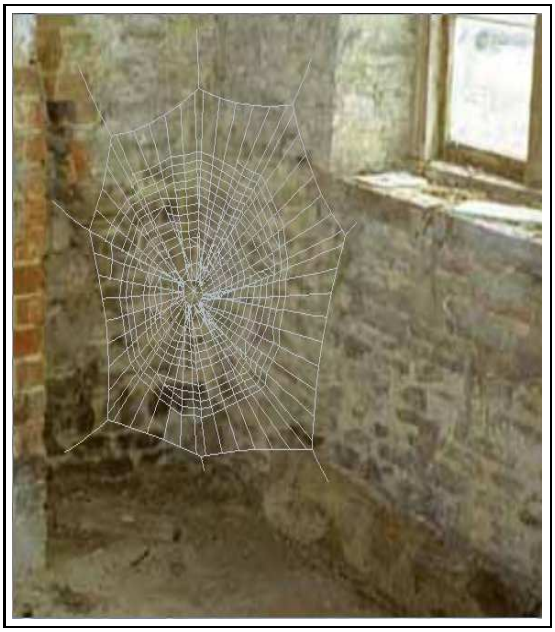

Figure 5.40: No Alpha Blending

Figure 5.41: With Alpha Blending

Figure 5.42: No Alpha Blending

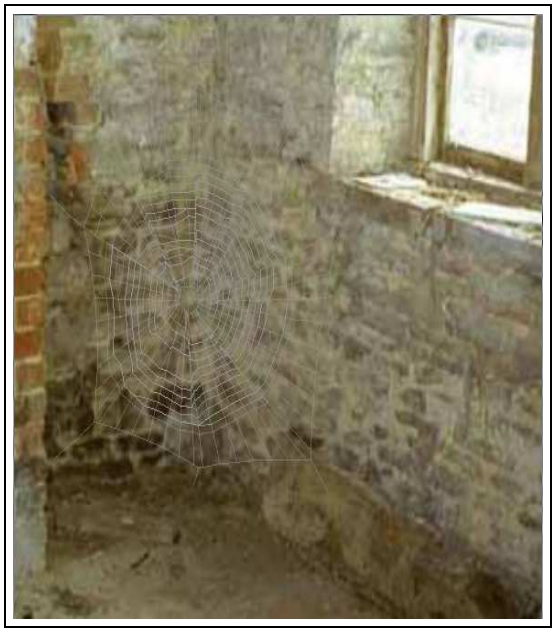

Figure 5.43: With Alpha Blending

Figure 5.44: Effect of Shading and Alpha Blending

5.3.9 Animation: Wind

A wind model that fits in well with the spring-mass system is used. This model introduces wind as an external force into the system and provides controls for the wind velocity and direction. The algorithm for wind generation is given in figure 5.45 and is divided into two segments. First segment of the algorithm is the *windgen* function, which is used to give the wind its velocity and direction. In this function we generate a new random number between +3 and -3 every 50 degrees and this random number, along with the sine term, is used to modulate the value of the wind vector, *wind_vector*. This wind vector, which is a force term, gives the wind direction and velocity. The sine term helps to add a predictable pattern to orb web animation and provides control of the orb web movement under the wind.

In the second segment of the algorithm, first the windgen function is called to obtain the new wind vector value, *wind_vector*. Then we calculate the distance, *distance*, of each mass in the web from a chosen center mass in the hub. This distance is then scaled and raised to an exponent to get the final value of *distance*. This gives us a value for *distance* which is very small for masses close to the hub and relatively large for masses away from the hub. Thus, when we use this *distance* value to modulate the wind vector and calculate the *mass_force*, the masses close to the hub will have a smaller force acting on them as compared to the masses farther away from the hub. This in turn generates an animation where the orb web near the hub, which is most sensitive to the applied external forces, billows out in proportion to the rest of the orb web.

The set of images given below have been obtained from a short 20 second anima-

tion of the orb web moving under a wind force. By altering the wind force direction and or the magnitude we can obtain any desired animation of the orb web moving in the wind.

```
 /* wind generation function */
 /* initialize the variables */
 MASS = 0.00000001
 GRAVITY = 9.8
 WIND = MASS*GRAVITY*10
 degree = 0
 random = 1
 wind_vector_x = 0, wind_vector_y = 0, wind_vector_z = 0
 windgen
 {
   if degree%50 = 0
      random = new random number between -3 and + 3
   wind_vector_x = random*WIND*sin(degree)
   wind_vector_y = random*WIND*sin(degree)
   wind_vector_z = - WIND*sin(degree)
   degree = degree + 1
 }

/* generate the wind */
call the windgen function to get new value for wind_vector
for all masses
  distance = calculate distance of this mass from the center mass
             (a mass in the hub is used as the center mass)
  alpha = orb web radius*50
  distance = distance/alpha
  distance = exponent(distance)
  mass_force_x = mass_force_x + distance*wind_vector_x
  mass_force_y = mass_force_y + distance*wind_vector_y
  mass_force_z = mass_force_z + distance*wind_vector_z
```

Figure 5.45: Wind Generation

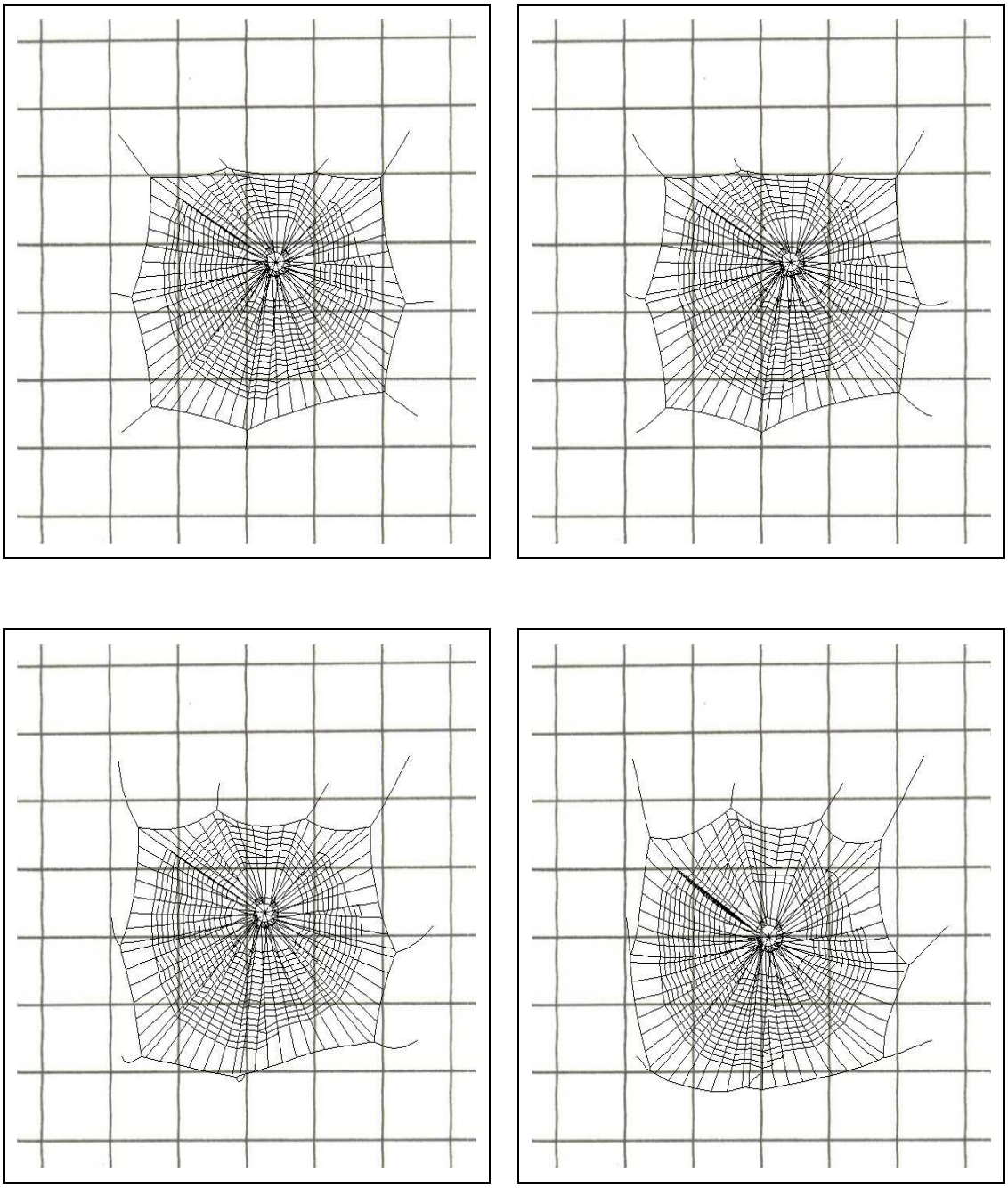

Figure 5.46: Wind Animation

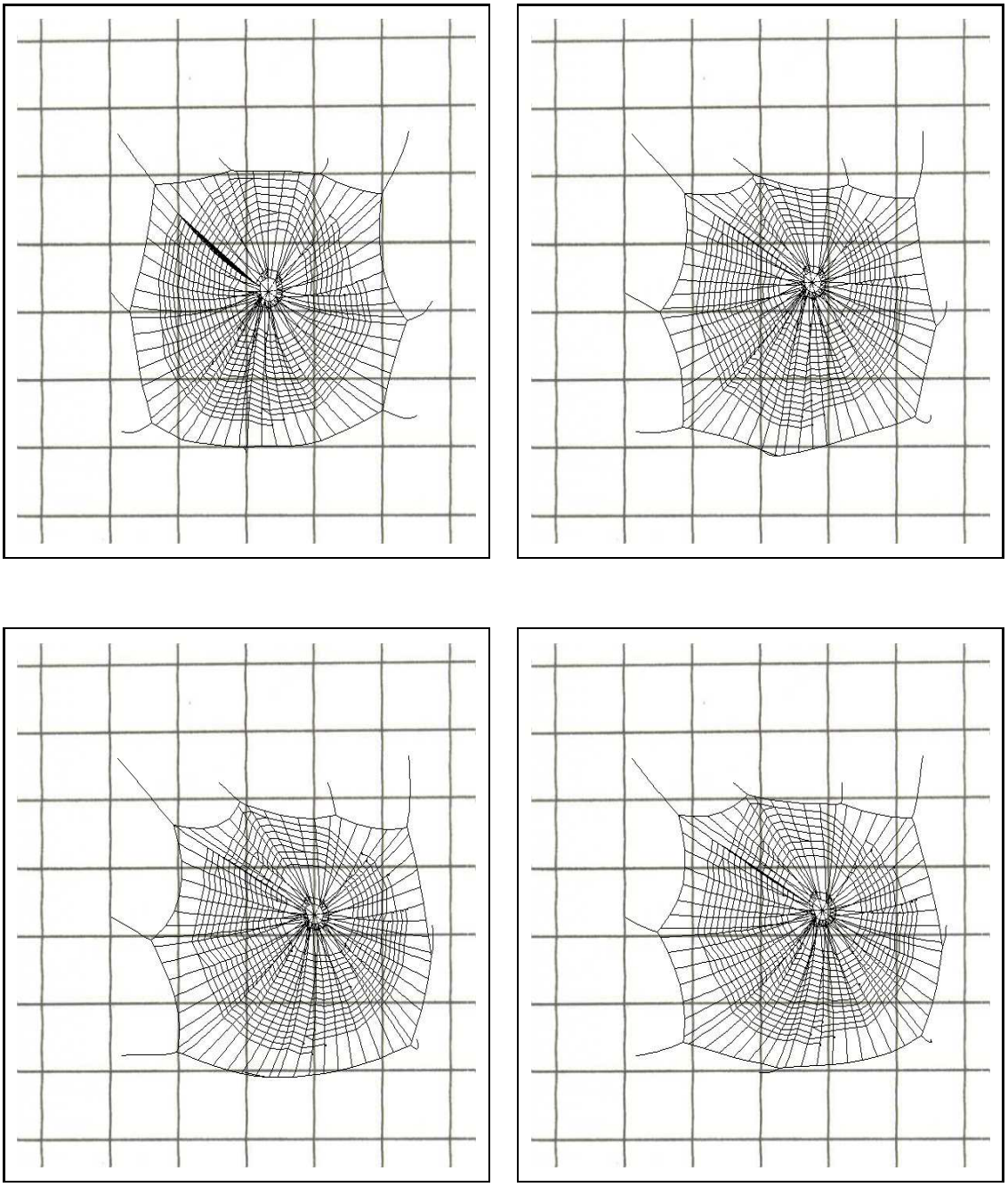

Figure 5.47: Wind Animation

Chapter 6

Charlotte's Web

This phase of the work derives its inspiration from the children's book "Charlotte's Web" [41]. In this book, Charlotte the spider spins webs with messages inscribed on them in order to save her friend Wilbur the pig (see figure 6.5 for an image from the book). It is highly unlikely that a real spider could weave such webs. We aim to inscribe our artificial webs with messages/symbols in them, while still keeping the same overall appearance of the web as much as possible and not altering its physically-based nature.

For this purpose, the first step is to create an orb web using the modeling procedure described earlier and then to allow the web simulation to settle down. The message which is to be inscribed on this web, called the *target image*, is then created using a simple image editor. The target image can either be text containing a few letters or a simple bi-level image (a black and white image) containing mainly low frequency content. Two such images are shown in figure 6.1 and 6.6. To inscribe the target image on the artificial web, we remove the spiral strands from the orb web spiral in the region where the image is to be superimposed on the web. This process is described next.

The orb web spiral is composed of short line segments which run from a mass in

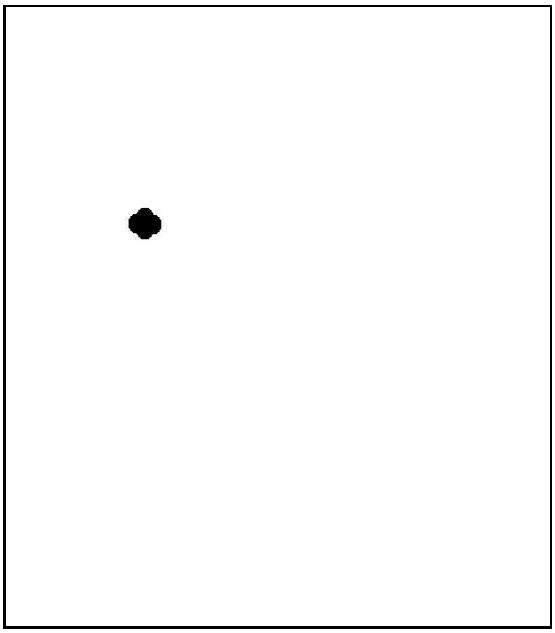

Figure 6.1: The Target Image

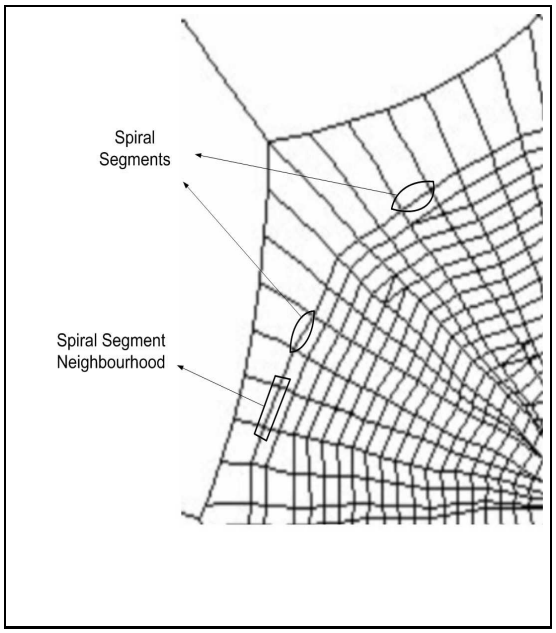

Figure 6.2: A Source Image Section (Enlarged)

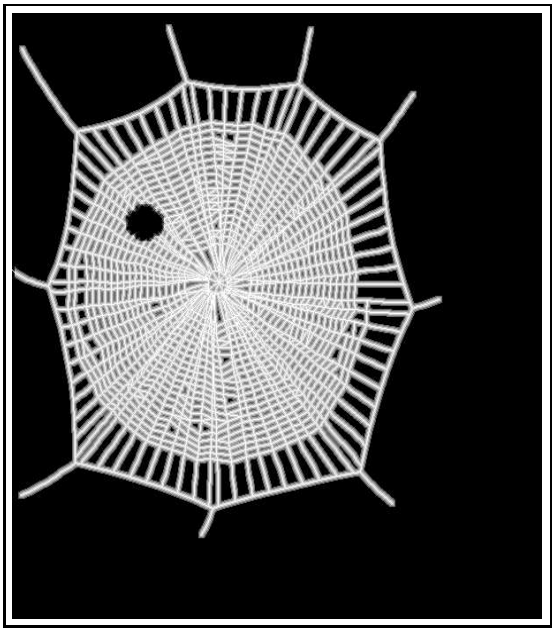

Figure 6.3: The Overlayed Images

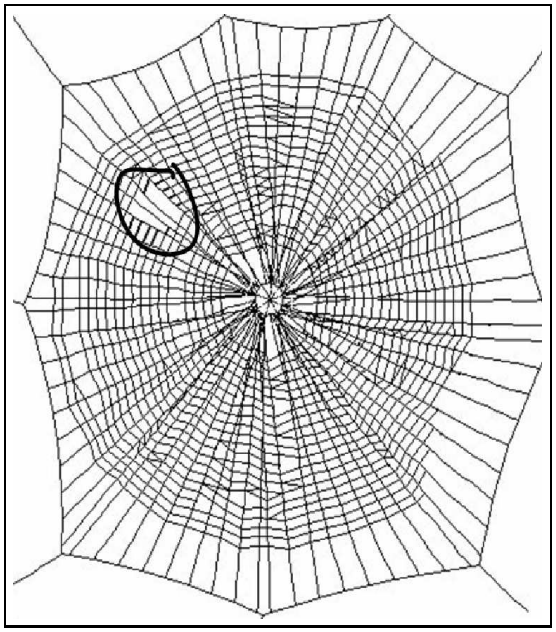

Figure 6.4: The Resultant Image

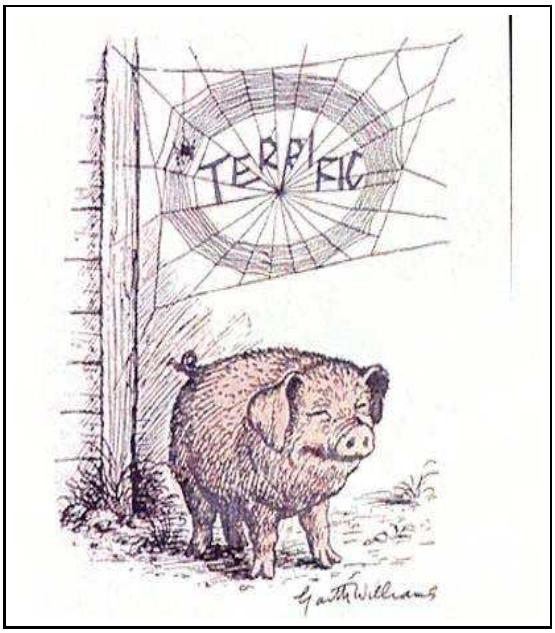

Figure 6.5: Charlotte's Web [41]

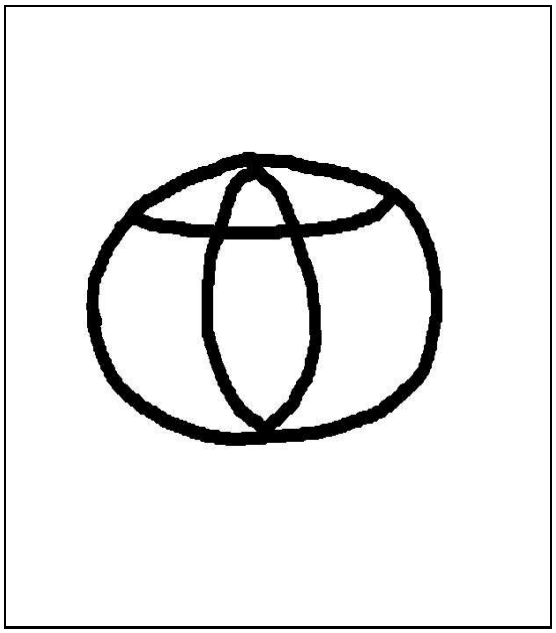

Figure 6.6: Toyota Logo Target Image

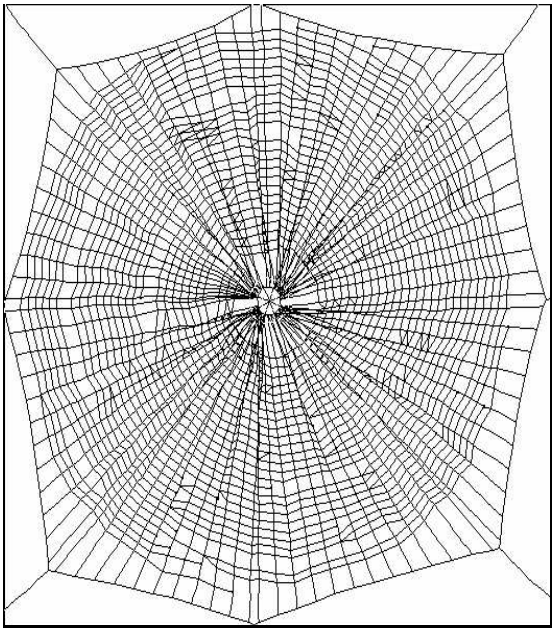

Figure 6.7: The Original Orb Web

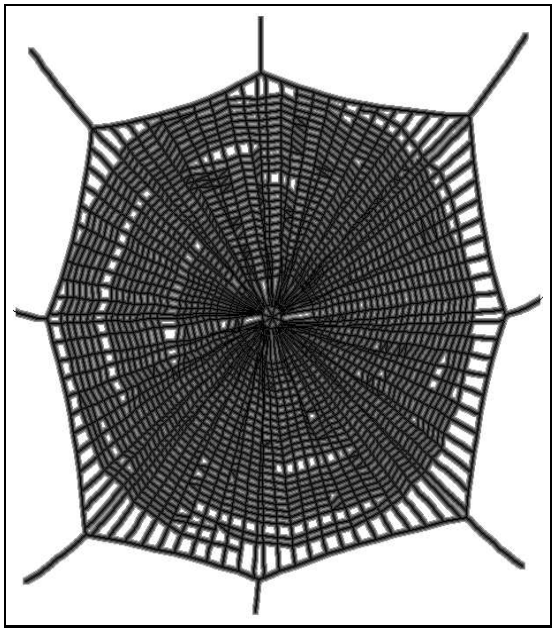

Figure 6.8: The Dilated Orb Web

a particular radial spoke to another mass in the next radial spoke. Two such spiral segments are shown in figure 6.2, which shows an enlarged section of an artificial orb web. These short orb web spiral segments are compared, segment by segment, with the target image. This comparison is done by taking a segment of the orb web spiral and comparing the small region in and around this segment with the corresponding region from the target image. This small region consists of the pixels in the neighbourhood of a particular line segment and one such region is depicted in image 6.2. More specifically, the *neighbourhood* of a spiral line segment consists of a rectangular region, five pixels wide (including the single pixel width of the spiral segment itself), running along the length of the spiral line segment. Figure 6.3 shows the target image from figure 6.1 overlayed over the source image. The large dark dot like shape clearly shows the region where the target image would lie on the orb web spiral when the images are overlayed.

A *score* which indicates the degree of matching between the target image region and the corresponding orb web segment is then computed. To compute the score we first capture the current orb web image from the screen to obtain the source image. The obtained source image and the target image are then *dilated*. Dilation is a *morphological operator*. Morphological operators take an image and a structuring element as input and combine them using a set operator, such as intersection, union, inclusion and complement [10]. In our implementation, we used a 7×7 structuring element. The basic effect of dilation is to enlarge the boundaries of orb web strands and of the dark region in the target image. This effect is shown in figures 6.7 and 6.8. The corresponding regions of the source and the target image can now be compared

```
/* first calculate the intensity ratio */
1. intensity_ratio = total_source_intensity/total_target_intensity
/* calculate the score */
2. thescore = floor(10 * intensity_ratio)
```

Figure 6.9: Calculating The Score

over a larger intensity range and a score, indicating the degree of intensity match, ranging between 1 to 10 can be computed.

Next, the cumulative pixel intensity, called *total source intensity*, of the small region in the neighbourhood of the strand segment is computed by summing the individual pixel intensities of each pixel in the neighbourhood. Similarly we compute the cumulative pixel intensity, called *total target intensity*, of the corresponding region in the target image. Next a ratio of the intensities is computed by dividing the total source intensity by the total target intensity. This ratio is then used to calculate the score, which ranges from 1 to 10. The score calculation algorithm is shown in figure 6.9.

Once a score has been calculated for each strand segment in the spiral, the segments with a score above a certain threshold score (set to 8 in our trials) are removed from the final image. By altering this threshold we can control the number of spiral segments that are removed from the final result. The final result of this process is shown in figure 6.4. It can be seen from this image that in the region where there was an overlap of the target and the source images (overlap is shown in figure 6.3) spiral strand segments have been removed, while at the same time the rest of the orb web has been left intact. Figure 6.11 shows another result obtained using the above process. Here the letters P,I and G are inscribed on the orb web. Figure 6.10 shows

the target image, used to generate figure 6.11, overlayed over the source image. As can be seen, the result of this process is an orb web which is still physically based but has the intriguing look of a web with text inscribed on it.

The text in figure 6.11, although visible with a little effort, is not easily readable. The viewer has to make some effort to decipher the inscribed letters. In order to further emphasize the inscribed text and to increase its readability, several possible solutions were investigated. The results of three such efforts are displayed in figures 6.12, 6.13 and 6.14. The underlying idea behind all three methods is to make the region where the text is inscribed stand out, while leaving the rest of the web unaltered.

To arrive at the first image (see figure 6.12), the orb web was modified by introducing additional strand segments between masses along the spiral. These additional segments connect a particular mass to two more masses, in addition to the one mass that to which it is currently connected. Now, in addition to the earlier segment's, these additional strand segments are compared against the target image and a score is computed. In the region where there is an intensity match, these additional strand segments are not removed from the final model and the result is the cross-hatch pattern as seen in the image. This resultant cross-hatch pattern increases the visibility of the inscribed letters, not only due to the fact that the cross-hatch pattern is at variance with the rest of the spiral but also due to the increase in spiral segment density in the region.

It can be observed from the initial image, shown in figure 6.11, that there exist large gaps in the regions inside the letters. In these gaps there are no spiral seg-

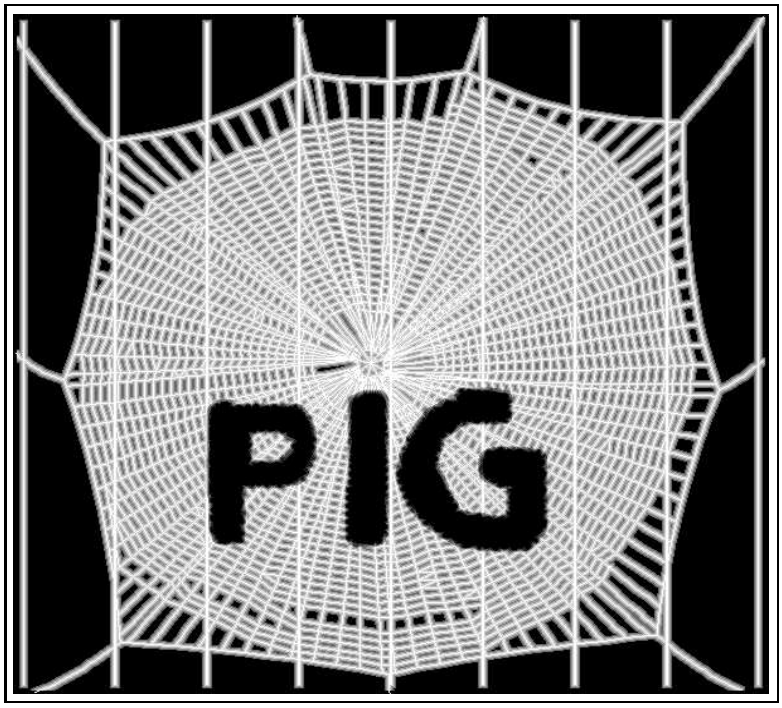

Figure 6.10: The Overlayed Images

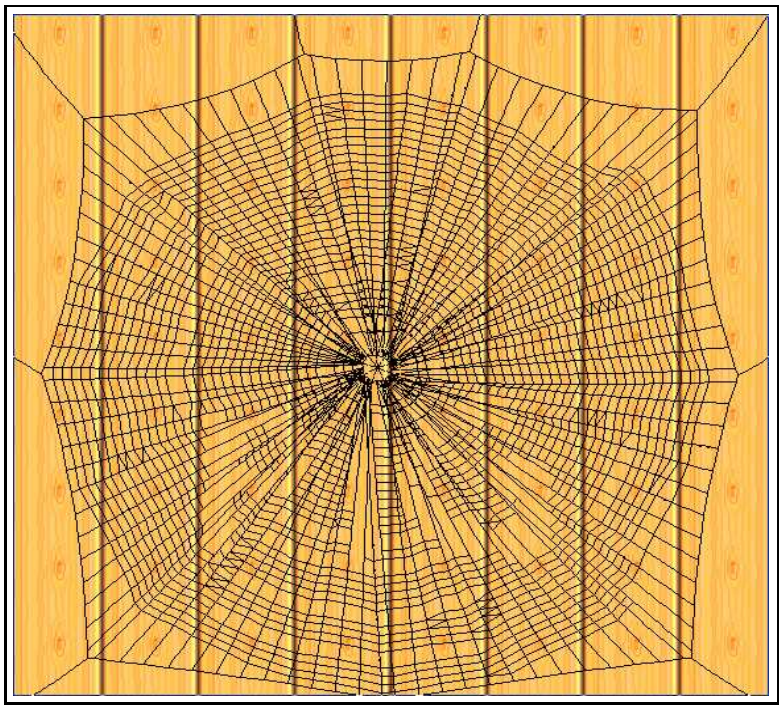

Figure 6.11: Web With Letters P, I, G Inscribed

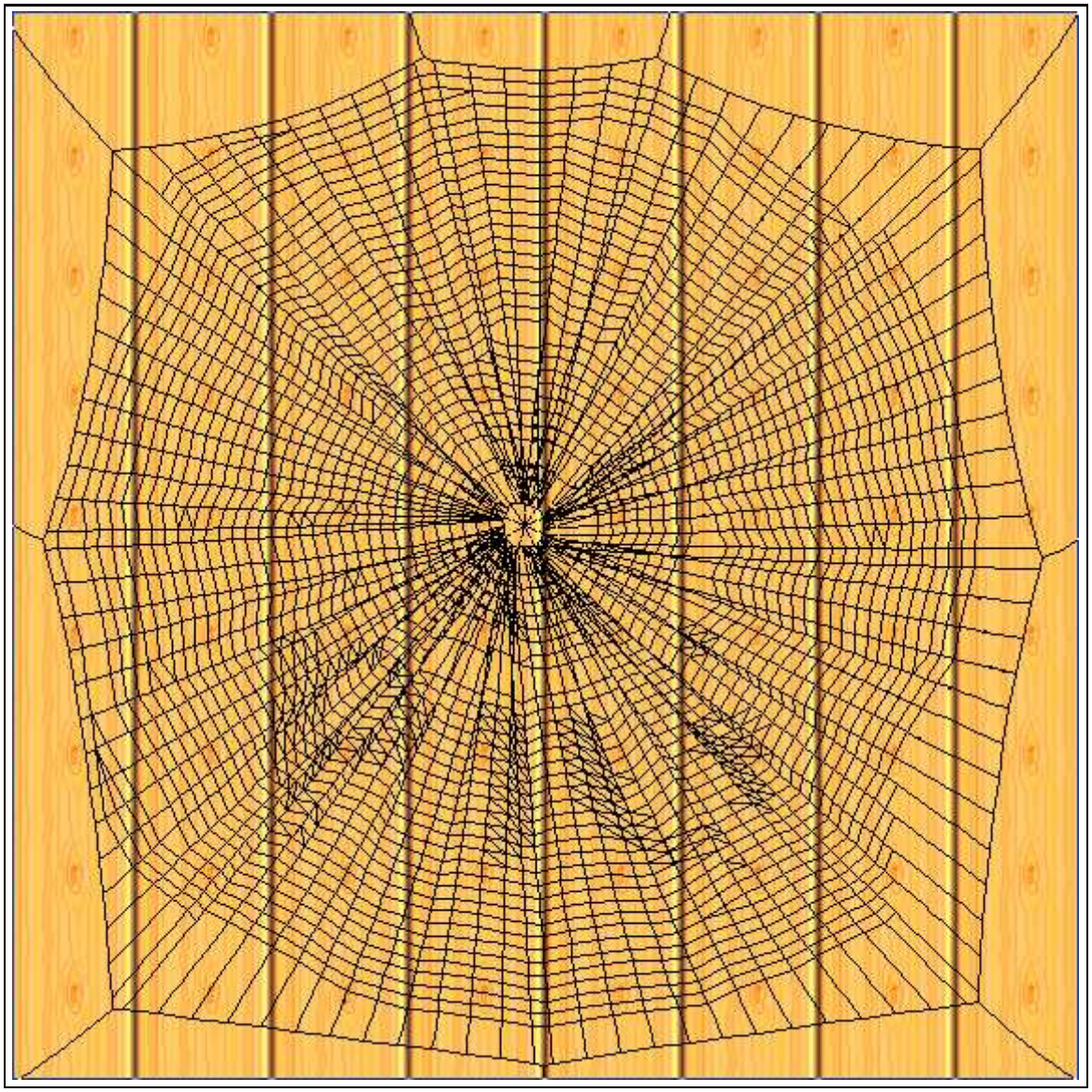

Figure 6.12

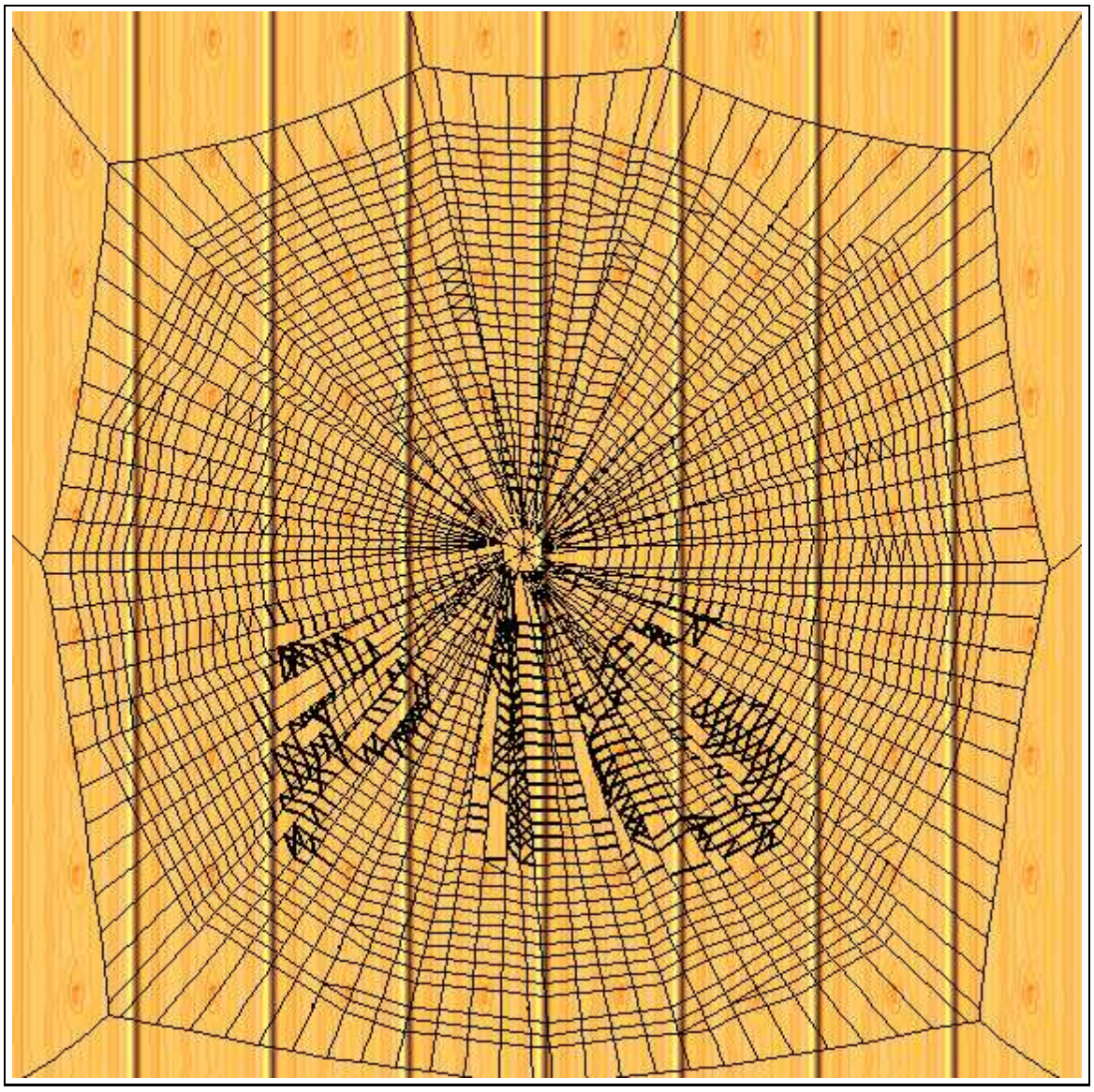

Figure 6.13

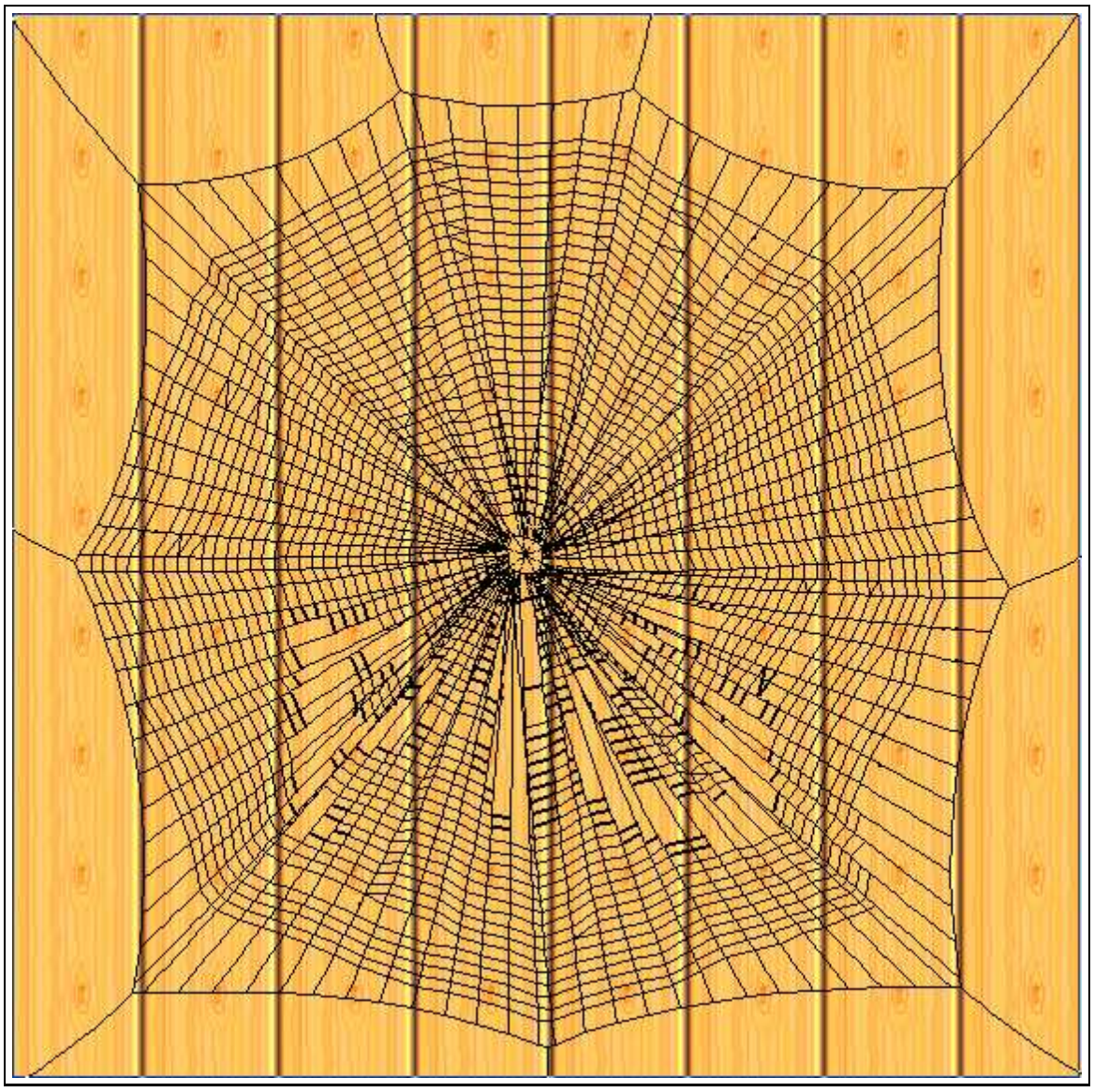

Figure 6.14

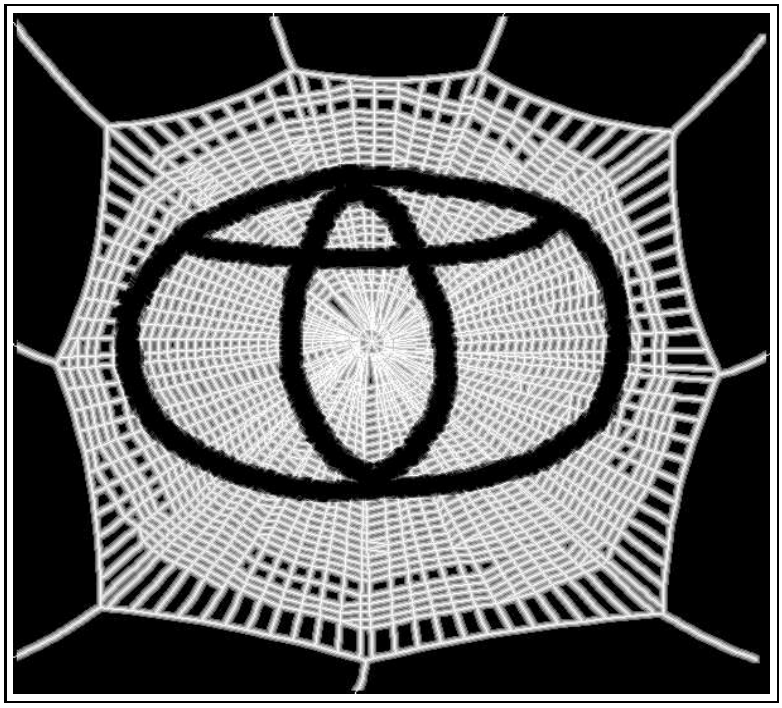

Figure 6.15: The Overlayed Images

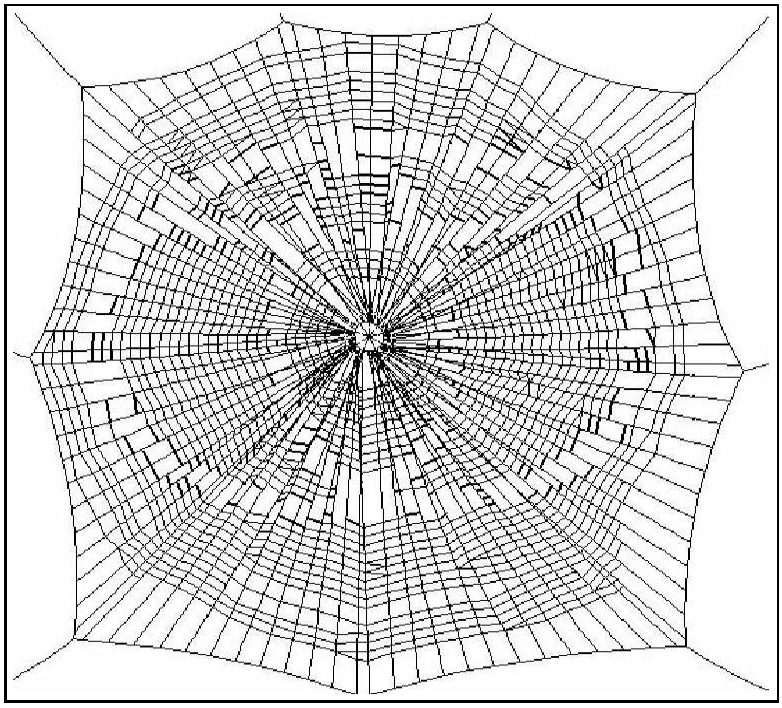

Figure 6.16: The Toyota Logo

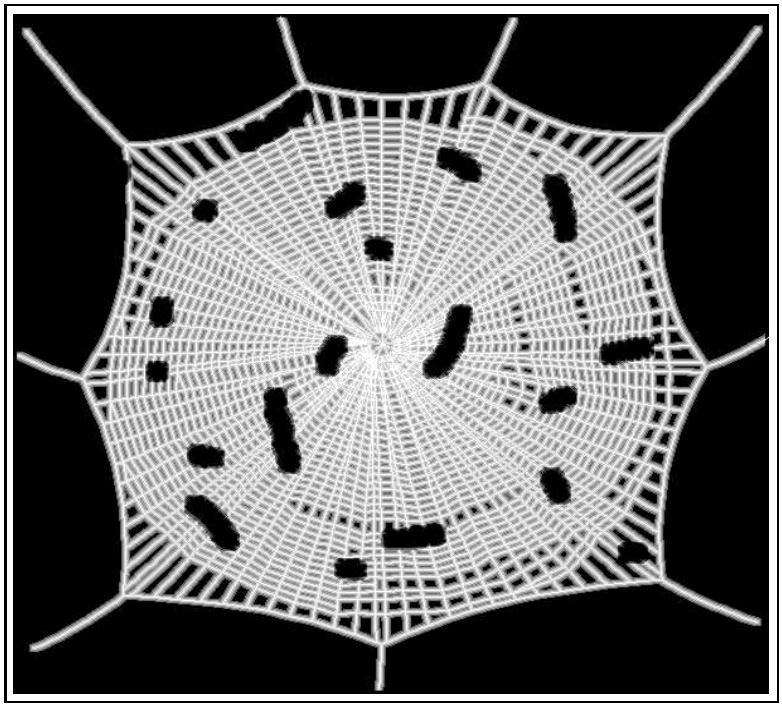

Figure 6.17: The Overlayed Images

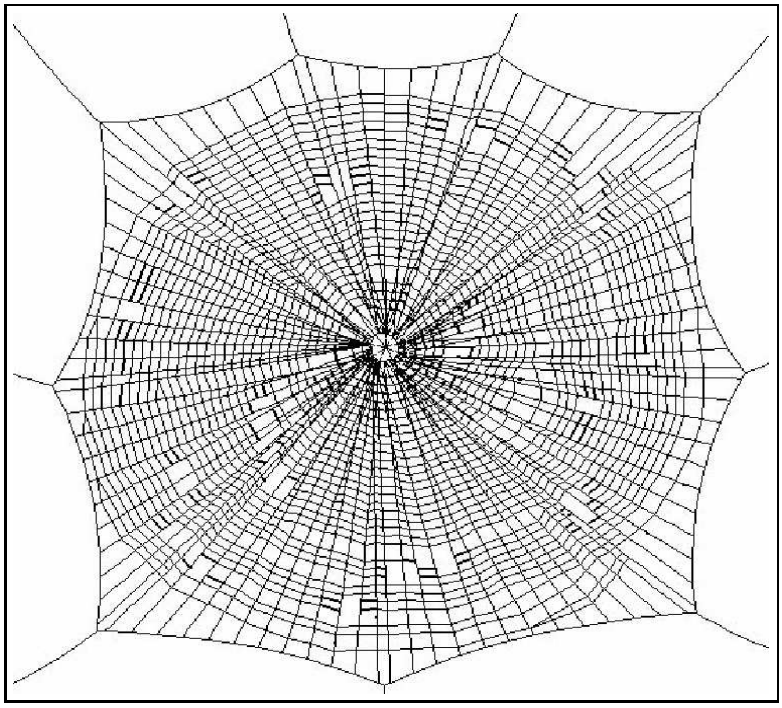

Figure 6.18: Spiral with random gaps

ments and only the underlying spokes are visible. For the second image, shown in figure 6.13, we have modified the initial image by introducing random attachments between the masses within the letters. These random spiral segments provide further emphasis to the inscribed letters. Moreover, these random spiral segments have been shaded darker than the rest of the orb web spiral segments to further augment the visibility of the inscribed letters. Although we might not be able to find an orb web in nature which has meaningful text inscribed on it, the St Andrews Cross spider (*Argiope Keyserlingi*) is an example of a spider whose webs have a region with a noticeable pattern, a cross, inscribed on the web. The cross is created by the spider by repeated laying of more silk in the region of the cross, and thus the cross is highly visible by virtue of its higher intensity than the rest of the orb web, similar to the technique used to create image 6.13. The final image, shown in figure 6.14, is an effort to strengthen the outline of the letters by increasing the intensity of the strands surrounding the letters.

The various methods to emphasize the letters produce results which conform to the original goal of inscribing text on the web, while trying to preserve as much of its natural look as possible. Perhaps figure 6.11 has the most natural look but the text in it lacks the clarity of the other figures.

The above process can be also be used to impress the web with simple images such as the Toyota logo shown in figure 6.16. Another interesting application of this process is that we can use it to alter the spiral to our purposes without having to resort to changing the underlying code. As an example, in the image in figure 6.18 we have introduced random gaps in the spiral. Although these gaps have been

darkened in the image to increase visibility, a more natural look can be obtained by not darkening these spots.

In this phase of the thesis we aimed to inscribe our artificial orb webs with messages/symbols, in a non-photorealistic style, inspired by a children's book. The attempt here was not to create webs similar to those illustrated in the book, but to adapt the concept to our physically-based orb web model. It was observed that our initial method to inscribe the web, while conforming to the original goals of keeping the natural look of the web and not altering its physically-based nature, produced images with hard to read inscriptions. Therefore, to further emphasize the inscriptions we investigated three other methods. The results of these three methods generate images with inscriptions that are easier to read, but at the loss of the natural look of the orb web.

Chapter 7

Conclusions and Future Work

The work done in this thesis was guided by the following goals:

- The creation of a model of an orb web, based on observed orb web characteristics.
- The animation of the created orb web when subjected to a wind force.
- The creation of the non-photorealistic orb web, which could be inscribed with text or symbols.
- The placement of the orb web in appropriate surroundings to enhance the appeal of the orb web.
- The creation of the additional effect of an orb web being covered with dew drops.

The first four of the above goals were achieved and the results of that work have been presented in this thesis. We have made an effort to incorporate the various characteristics of orb webs, such as their spatial density, size and weight, in a physically-based model. This model of the orb web is then modified to incorporate an interesting effect; using the children's book *Charlotte's Web* as an inspiration, we have inscribed the orb web with text/symbols, while keeping the physically based

nature of the model. Perhaps, this is the most important contribution of this work. The orb web model has various parameters that can be altered to control attributes such as spiral density, radial strand density, hub placement and hub density. Since a real orb web is not easy to observe and photograph, we have emphasized the orb web in the images by rendering it in a darker shade than a real orb web under similar lighting conditions.

There are a number of possible improvements and extensions that can be made to the work done in this thesis. While the created orb web model provides results that demonstrate the applicability of the spring-mass model to modeling orb webs, this model can be further enhanced and tailored to the specific needs or purposes at hand. For instance, if performance is a major concern then the model can be further optimized or if a more accurate representation of an orb web is required then the orb web construction algorithms can be altered. While in this thesis the rendering of the web was not a major concern, if desired more sophisticated rendering and shading techniques can be used. The effect of an orb web covered with dew drops was not pursued in this thesis as a result of the limited time available. The initial study into water droplets and their modeling suggests that this is an excellent area for future research, both in the context of this thesis and also as a stand-alone problem.

References

[1] David Baraff and Andrew Witkin. Online siggraph '97 course notes: Implicit methods for differential equations, 1997. Available on the Internet: http://www-2.cs.cmu.edu/ baraff/sigcourse/.

[2] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 25th annual conference on Computer Graphics and Interactive Techniques*, 1998.

[3] Suresh P. Benjamin and Samuel Zschokke. A computerised method to observe spider web building behaviour in a semi-natural light environment. In *European Arachnology. Proceedings of the 19th European Colloquium of Arachnology*, 2000.

[4] Nexia Biotechnologies. General information about spider silk. Available on the Internet: http://www.nexiabiotech.com/en/03_bio/05.php.

[5] Carles Bosch, Xavier Pueyo, Stéphane Mérillou, and Djamchid Ghazanfarpour. A physically-based model for rendering realistic scratches. In *Eurographics 2004. Proceedings of the 2004 Eurographics annual conference*, 2004.

[6] John C. Butcher. *Numerical Methods for Ordinary Differential Equations*. Wiley, 1987.

[7] Peter Chew. Spider scientific facts. Available on the Internet: http://www.geocities.com/brisbane _weavers/ScientificPage.htm.

[8] Paul Fearing. Computer modelling of fallen snow. In *SIGGRAPH 2000 Conference Proceedings*, pages 37–46, New Orleans, LA, USA, 23–28 July 2000. ACM SIGGRAPH. Published as Computer Graphics Proceedings, Annual Conference Series, 2000.

[9] The Institute for Environmental Modeling. Spider silk: Stress-strain curves and youngs modulus. Available on the Internet: http://www.tiem.utk.edu/ gross/bioed /bealsmodules/spider.html.

[10] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.

[11] Paul Haeberli. Paint by numbers: abstract image representations. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 17th annual conference on Computer Graphics and Interactive Techniques*, 1990.

[12] Gavin Miller and Andrew Pearce. Globular dynamics:a connected particle system for animating viscous fluids. In *Computers and Graphics*, 1989.

[13] Gavin S. P. Miller. The motion dynamics of snakes and worms. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 15th annual conference on Computer Graphics and Interactive Techniques*, 1988.

[14] Gavin S. P. Miller. The motion dynamics of snakes and worms. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 15th annual conference on Computer Graphics and Interactive Techniques*, 1988.

[15] James V. Miller, David E. Breen, William E. Lorensen, Robert M. O'Bara, and Michael J. Wozny. Geometrically deformable models: A method of extracting closed geometric models from volume data. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 18th annual conference on Computer Graphics and Interactive Techniques*, 1991.

[16] Australian Museum. Silk structure. Available on the Internet: http://www.amonline.net.au/spiders/ toolkit/silk/structure.htm.

[17] Burke Museum of Natural History and Culture . University of Washington. Available on the Internet: http://www.washington.edu/burkemuseum/ spidermyth/myths/orbweb.html.

[18] Australian Museum Online. Types of spider webs. Available on the Internet: http://www.amonline.net.au/spiders/life /prey/web_types.htm.

[19] John C. Platt and Alan H. Barr. Constraint methods for flexible models. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 15th annual conference on Computer Graphics and Interactive Techniques*, 1988.

[20] Stephen M. Platt and Norman I. Badler. Animating facial expressions. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 8th annual conference on Computer Graphics and Interactive Techniques*, 1981.

[21] William Press, Saul Teukolsky, William Vetterling, and Brian Flannery. *Numerical Recipes in C, second edition.* Cambridge University Press, 1992.

[22] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behavior. In *Graphics Interface.*, 1995.

[23] William T. Reeves. Particle systems - a technique for modeling a class of fuzzy objects. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 10th annual conference on Computer Graphics and Interactive Techniques*, 1983.

[24] William T. Reeves and Ricki Blau. Approximate and probabilistic algorithms for shading and rendering structured particle systems. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 12th annual conference on Computer Graphics and Interactive Techniques*, 1985.

[25] William T. Reeves and Alain Fournier. A simple model of ocean waves. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 13th annual conference on Computer Graphics and Interactive Techniques*, 1986.

[26] Manaaki Whenua Landcare Research. Insects and spiders of new zealand. Available on the Internet: http://www.landcareresearch.co.nz/education /insects_spiders/facts.html.

[27] Craig W. Reynolds. Flocks, herds and schools:a distributed behavioural model. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 14th annual conference on Computer Graphics and Interactive Techniques*, 1987.

[28] RuneVision. Rune's particle system. Available on the Internet: http://runevision.com/3d/include/particles/.

[29] Mikio Shinya and Alain Fournier. Stochastic motion-motion under the influence of wind. In *Computer Graphics Forum*, 1992.

[30] Karl Sims. Particle animation and rendering using data parallel computation. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 17th annual conference on Computer Graphics and Interactive Techniques*, 1990.

[31] Jos Stam. Stochastic dynamics: simulating the effects of turbulence on flexible structures. In *Computer Graphics Forum*, 1997.

[32] Jos Stam and Eugene Fiume. Turbulent wind fields for gaseous phenomena. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 20th annual conference on Computer Graphics and Interactive Techniques*, 1993.

[33] Richard Szeliski and David Tonnesen. Surface modeling with oriented particles. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 19th annual conference on Computer Graphics and Interactive Techniques*, 1992.

[34] Demetri Terzopoulos, John Platt, and Kurt Fleischer. From gloop to glop: Heating and melting deformable models. In *Graphics Interface*, 1989.

[35] David Tonnesen. Modeling liquids and solids using thermal particles. In *Graphics Interface*, 1991.

[36] Xiaoyuan Tu and Demetri Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 21st annual conference on Computer Graphics and Interactive Techniques*, 1994.

[37] Greg Turk. Generating textures on arbitrary surfaces using reaction diffusion. In *SIGGRAPH 1991. Proceedings of the 1991 SIGGRAPH annual conference*, 1991.

[38] Jarke J. van Wijk. Flow visualization with surface particles. In *IEEE Computer Graphics and Applications*, 1993.

[39] Henrik Wann Jensen, Frédo Durand, Michael M Stark, Simon Premoze, Julie Dorsey, and Peter Shirley. A physically-based nightsky model. In *SIGGRAPH 2001. Proceedings of the 2001 SIGGRAPH annual conference*, 2001.

[40] Jakub Wejchert and David Haumann. Animation aerodynamics. In *International Conference on Computer Graphics and Interactive Techniques. Proceedings of the 18th annual conference on Computer Graphics and Interactive Techniques*, 1991.

[41] E. B. White. *Charlotte's Web*. HarperCollins, 1974.

[42] Andrew Witkin and David Baraff. Online Siggraph '97 course notes: Differential equation basics, 1997. Available on the Internet: http://www-2.cs.cmu.edu/ baraff/sigcourse/.

[43] Andrew Witkin and David Baraff. Online Siggraph '97 Course Notes: Particle System Dynamics, 1997. Available on the Internet: http://www-2.cs.cmu.edu/ baraff/sigcourse/.

[44] O.C. Zienkiewicz and R.L. Taylor. *The Finite Element Method*. McGraw-Hill, 1991.

[45] Victor Brian Zordan, Bhrigu Celly, Bill Chiu, and Paul C. DiLorenzo. Breathe easy:model and control of simulated respiration for animation. In *Symposium on Computer Animation. Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation*, 2004.

[46] Samuel Zschokke. Construction and structure of orb webs. Available on the Internet: http://www.unibas.ch/dib/nlu /staff/sz/webconstruct.html.

[47] Samuel Zschokke. Samuel's spider web construction gallery. Available on the Internet: http://www.unibas.ch/dib/nlu /staff/sz/spidergallery.html.

[48] Samuel Zschokke. *Web construction behaviour of the orb weaving spider Araneus Diadematus*. PhD thesis, Universität Basel, 1994.

[49] Samuel Zschokke and Fritz Vollarath. Web construction patterns in a range of orb-weaving spiders. In *European Journal of Entomology . Proceedings of the 15th European Colloquium of Arachnology*, 1995.