CINEMATIC SCIENTIFIC VISUALIZATIONS

A Thesis

by

KENDALL RUTH LITAKER

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

MASTER OF SCIENCE

Chair of Committee,	Ergun Akleman
Committee Members,	Lucas Macri
	Ann McNamera
	Frank Summers
Department Head,	Timothy McLaughlin

August 2013

Major Subject: Visualization

ABSTRACT

The Hubble Space Telescope has provided the world with incredible imagery of the surrounding universe. The aesthetic quality of this imagery is limited by production resources; by creating a method to harness the highly refined detail and quality of CG elements in live-action films, we can inspire and educate at a much greater level. In this thesis, I create a rendering approach that allows camera movement around and through elements such as nebulae and galaxies, creating a more cinematic experience. The solution will also allow for reasonable scientific accuracy, visual appeal, efficiency, and extendibility to other astronomical visualizations.

3D meshes are constructed and textured using telescopic images as reference. Splats are volumetrically generated using a voxelized bounding box around the mesh. Valid splats within a user specified maximum distance receive initial color and alpha values from the texture map. Probability density functions are used to create a density falloff along the edges of the object, and modifications to the RGBA values are made to achieve the desired cloud-like appearance. The data sets are rendered using a C program developed at the Space Telescope Science Institute by Dr. Frank Summers. The methodology is applied to the test cases of a nebula, star-forming region Sharpless 2-106, and a galaxy, Messier 51, or the Whirlpool Galaxy.

The results of this thesis demonstrate the visual, scientific, and technical success of this solution. The code developed during this project generates the desired imagery with reasonable efficiency. A short animation moving from outside the galaxy to a close up of the nebula exhibits the flexibility in scale and camera movement. A careful balance between scientific accuracy and visual appeal were maintained through consultation with astronomers at the Space Telescope Science Institute. The favor-

able efficient, flexible, visual, and scientific results presented by this work make this process extendable to most other cases of nebula and galaxy visualizations.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION

The goal of this work is to create volumetric representations of astronomical features like nebulae and galaxies with a high level of detail and scientific accuracy. A volumetric structure ensures that there is flexibility in camera movements, allowing astronomers to present their research in a more cinematic way. We present a new solution for generating nebulae and galaxies as a large collection of particles, or *splats*. Splats are simply dots with a convolution filter applied, allowing for blending between splat edges to produce a smoothly rendered image. Splat positions are based upon the voxelization of a bounding box surrounding an initial 3D surface mesh. The resulting data set can be modified for the desired appearance and finally rendered using a Z-buffer based approach.

1.1 Motivation

Since its launching in April 1990, the Hubble Space Telescope has provided the world with incredible imagery of the surrounding universe. Through Hubble, astronomers have made some of the most important discoveries about the nature of the universe, including the evolution of galaxies from the Hubble Deep Field, and the existence of dark energy [39].

Hubble's discoveries have been the subject of many documentary films, most of which combine the use of telescopic and computer-generated images. These films inform and entertain the public, and have taken artistic license where necessary in order to achieve the camera motion and storytelling required [3]. Given the high resolution of the images that Hubble captures, a method of creating films of astronomical phenomena that are both more scientifically accurate and visually appealing can be determined.

In this thesis, we introduce a procedure of generating, texturing, and rendering volumetric sets of particles as a possible solution to achieving the distinctive look of feature film computer graphics while maintaining the scientific integrity of the original telescopic images. I demonstrate this method using the test cases of a nebula and a galaxy, as these situations present the greatest visual and technical challenge. Planets can be visualized fairly simplistically, using a texture mapped sphere, and stars are generally unresolved points of light. A galaxy or a nebula, however, exists on a much greater scale, and thus must maintain detail as the camera moves from far away to near. Also, their inherent cloud-like, semi-transparent nature requires a volumetric approach that can be very expensive to compute.

Previous astronomical visualizations of nebulae have utilized volume scene graph solutions, where they describe a volume using various shading functions [23]. This solution results in realistic and reasonably scientifically accurate images, but is computational expensive to render [14]. For this reason, I have chosen to use a splat rendering system to create my output images, as this method appears to be the most efficient and applicable to this situation.

A volumetric splat solution will create a cloud form that can be explored from multiple viewpoints, providing camera flexibility. By rendering a structure with a large number of splats, their edges will blend together and form a single surface. Such a system will allow astronomers to present their research in a clearer way, as a cinematic depiction can be more appealing and understandable to the public. Due to the extreme distances involved, it is difficult to find depth information in astronomical images, and the relationships between objects can become muddled. Adding the third dimension creates a more defined picture of how these phenomena most conceivably appear.

1.2 Introduction

The goal of this work is a procedure in which nebulae and galaxies can be accurately rendered at a high resolution with a large number of splats in a reasonable amount of time. This will enable a myriad of camera options, including fly throughs, fly overs, and zoom ins. The result will be subject to the following aesthetic rules and assumptions. (1) *Reasonable scientific accuracy.* This work will make scientific data more accessible to both researchers and the general public. Thus, fidelity to the original Hubble images must be maintained as much as possible. Scientific intuition and scientifically guided artistic license is used where data is incomplete or missing. (2) *Extendability.* This solution should ideally be applicable to other cases where galaxies and nebulae need to be created. (3) *Efficiency.* The final procedure must fast enough to implement on a small number of workstations with a limited selection of software. (4) *Camera flexibility.* By creating these objects in a volumetric manner, any cinematically inspired camera moves will be available for use. (5) *Visual appeal.* While scientific accuracy is of primary importance, the aesthetics of the final piece must also be taken into account. A visually dynamic film will capture the audience's attention and inspire their imaginations.

The process begins with modeling and texturing a surface mesh that is representative of the astronomical form. The physical structure of certain features can be estimated using the surrounding gravitational forces and the probable motion and flow of gases and dust. For the purposes of this thesis, a mesh must be developed using a combination of previous research work and scientific intuition. The original telescopic image must be edited to remove extraneous features like stars and background galaxies. Once this is done, the revised image can be applied to the mesh as a texture map.

Next, the splats must be generated and textured using the surface mesh as a basis. To form a volumetric cloud that is similar in shape to the surface mesh, only splats that are a within a user specified distance are kept. The initial set of test positions are established using a bounding box around the mesh. The bounding box is subdivided along each axis into voxels. The corners and centers of each voxel are jittered slightly to break up the inherent grid appearance of the voxelization process. The test positions are then culled down by checking the distance between the test position and the closest point on the mesh. If the position is within the allowable distance, a splat is generated and assigned a color from the texture map that corresponds to the closest point on the mesh.

Modifications can be made to the data set to improve the visual appeal of the rendered image. Specifically, variations in the radius size, scaling the alpha values, and the addition of noise have shown to be successful in recreating a volumetric structure with fidelity to the original image. Due to the subjective nature of this section of the process, many iterations are needed to perfect the appearance. The data is finally rendered using a Z-buffer approach implemented by Dr. Frank Summers at the Space Telescope Science Institute (STScI) in Baltimore, Maryland.

The rest of this thesis is organized as follows. Section 2 details background information on nebulae, galaxies, and stars that is relevant to the comprehension of this work. Section 3 summarizes the related work. Section 4 describes the process for generating, texturing, and rendering the splat sets. Section 5 details the implementation and results of the thesis, and Section 6 presents the conclusion.

2. BACKGROUND INFORMATION

There are many types of nebulae and galaxies, each with their own characteristics and physical idiosyncrasies. Since scientific accuracy is of such importance to the success of this process, some knowledge of how nebulae and galaxies are formed and their chemical and physical makeup will be helpful in understanding the choices made over the course of this work. This chapter summarizes such information. Section 2.1 explains the characteristics of nebulae, with the different types delineated in Section 2.1.1. Section 2.2 introduces the concept of a galaxy, with Section 2.2.1 describing the structure of a galaxy and Section 2.2.2 highlighting the different galaxy types. Section 2.3 explains the importance of stars to the work, as well as what characteristics are important for visualization purposes.

2.1 Nebulae

In astronomy, a nebula is a cloud of dust and gases such as hydrogen and helium. These gases are ionized by nearby stars, causing them to glow slightly at different wavelengths of light [59]. A large scale nebula close to Earth is the Orion Nebula [61]. It, like many other nebulae, is a massive area of star formation. Every particle of dust and gas in a nebula has an associated gravitational force, causing clumps of matter to form. As the pressure increases, nuclear fusion occurs and a star is born. These stars begin to trap more dust and gases as their gravitational pull increases, which can eventually coalesce into the beginnings of a solar system similar our own. By studying nebulae, astronomers gain insight into the formation of our solar system and others around the universe [4].

Nebulae are iconic astronomical images, making them ideal candidates for public outreach. The general color of a nebula is determined by its chemical composition.

As the atoms of different elements are energized, they release photons at different wavelengths of light. The photon travels through space, causing the nebula to appear the color of the spectrum of that element. Since nebulae are made up of many types of gases, their images can be quite breathtaking [4]. Using various filters, astronomers can isolate wavelength regions and capture separate images of each chemical element in a nebula. The images are assigned different colors and composited together, resulting in press release images that are a riot of color, delighting the public eye. A successful 3D representation of a nebula combined with cinematic camera shots will further inspire the public's imagination. For the purposes of this project, I will be using images which have been composited and altered for both clarity and aesthetic purposes.

2.1.1 Type Specific Information

There are many types of nebulae in the universe, each with its own individual properties and traits. In order to effectively recreate their appearance, it is important to understand these characteristics (see Fig. 2.1). Most nebulae can be categorized as *H II regions*, or star-forming regions. These nebulae are also called emission nebulae, and surround newly formed stars. The hydrogen in the cloud has been ionized by massive young stars, causing it to glow in the corresponding emission lines of the nebular spectrum. H II regions have defined boundaries formed by ionization fronts, which is where the ionized photons deplete over a relatively short distance. A less common form of nebulae are reflection nebulae, or *H I regions*, which reflect light from nearby stars. The gases of these nebulae scatter the shorter, blue wavelengths of light more efficiently, causing the nebula to appear blue [59].

Other nebulae are dense and cold and generally opaque in visible light; these are called *dark nebulae*. Using telescopes sensitive to longer wavelengths of light in the

(a) (b) (c)

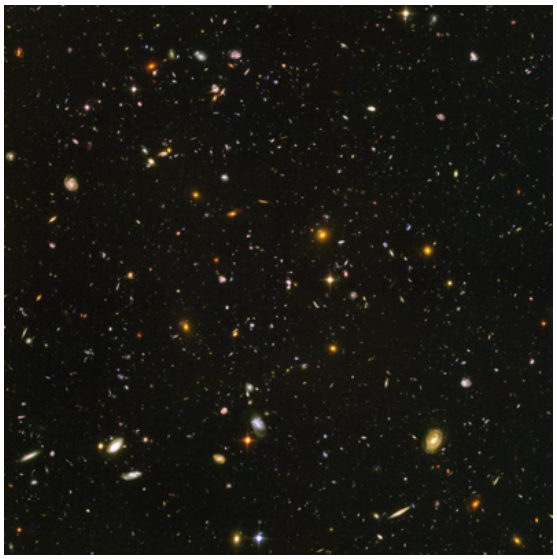
(d) (e)

Figure 2.1: Types of Nebulae: (a) H II region [28], (b) H I region [37], (c) Dark nebula [36], (d) Planetary nebula [27], and (e) Supernova remnant [29]

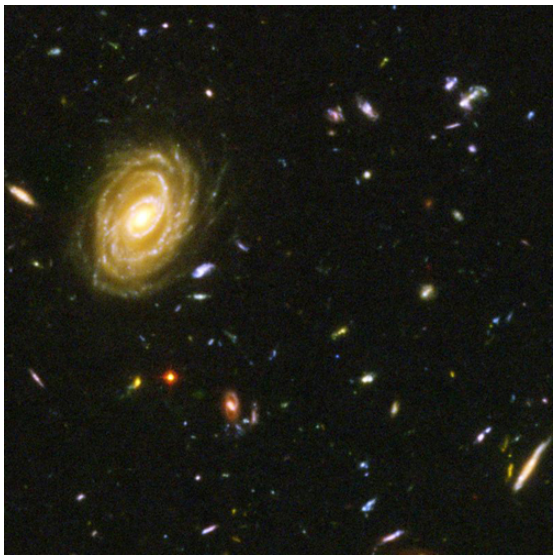
infrared part of the spectrum, these clouds can become slightly transparent, revealing information within and behind the nebula. *Planetary nebulae* form from the gases released by a solar mass star when it exhausts its fusionable material. The core of the nebula is the stellar remnant called a white dwarf. It is possible that bi-polar symmetric patterns can form from the gas emissions, jets, and winds that occur as the star evolves. *Supernova remnants* are the type of nebulae produced when a dying star ends it life in a giant explosion. Prior to the final explosion, there may be multiple bursts of energy causing the gases to form bubble shapes and symmetric patterns [59].

2.2 Galaxies

In 2004, the Hubble telescope was directed toward an empty area of space in the sky. After 800 exposures over 10 days, this supposedly empty area of sky was revealed to have approximately 10,000 galaxies. Using the Hubble Ultra Deep Field (see Fig. 2.2) as a yardstick, astronomers have estimated that there are about 100 billion galaxies in the universe, each made up of 100s of millions to 100s of billions of stars, dust particles, and various gases bound together by gravity [25]. Galaxies are important for astronomical research as they give us insight into the formation and structure of our own galaxy, the Milky Way. By observing galaxies that are farther and farther away, and hence looking back through time, we can understand how our galaxy began and how it evolved over billions of years. We can extrapolate what changes our galaxy might undergo in the future [24].



(a) Hubble Ultra Deep Field [24]



(b) Hubble Ultra Deep Field Inset [26]

Figure 2.2: Hubble Ultra Deep Field

2.2.1 Makeup of a Galaxy

Galaxies are made up of several components, each with its own distinct look and feel (see Fig. 2.3). Depending upon the type of galaxy, it may have all or only a few of these components. A central bulge is round in shape and usually includes older stars, gas, and dust. In a spiral galaxy, a disk can surround the bulge, with younger stars, gas, and dust collected into spiral arms, creating a "pinwheel" appearance. A halo encases the bulge and disk; it is made up of individual stars and globular clusters, or groups of stars. Stars span a range of masses which correspond to a range of temperatures. Massive stars are hot and blue, typically burning out after a short time; lower mass stars are redder and cooler, and live longer lives. Areas of gas and dust coalesce into nebulae of all types. All of these components will need to be understood and visualized in order to successfully create a realistic galaxy [4].

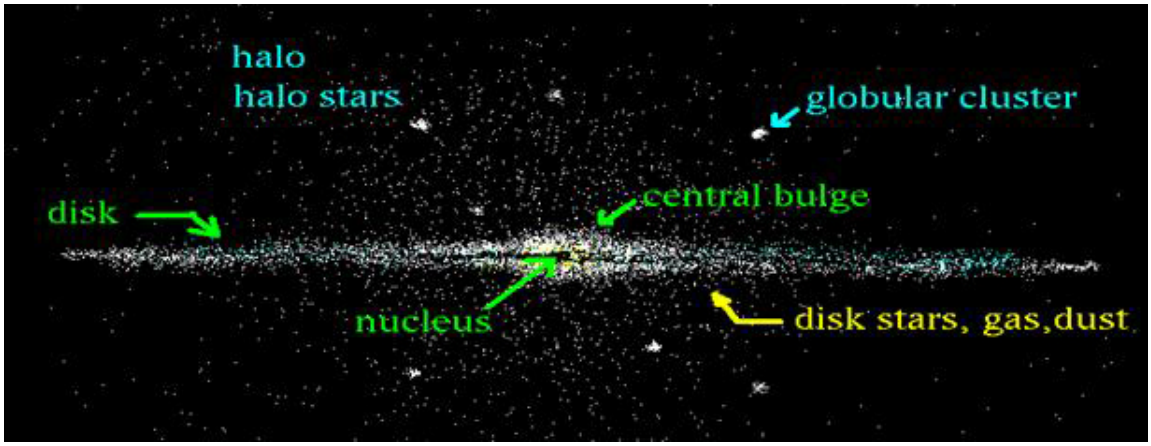


Figure 2.3: Basic Structure of a Spiral Galaxy [48]

2.2.2 Types of Galaxies

Astronomers classify galaxies into three main types, spiral, elliptical, and irregular, based upon the appearance of the central bulge and surrounding disk (see Fig. 2.4). As galaxies can be oriented in different directions, sometimes classifications must be made with only edge-on reference images available. In this case, the organization of the bulge and disk elements can help indicate the type of galaxy.

(a) Spiral Galaxy [33]

(b) Elliptical Galaxy [38]

(c) Irregular Galaxy [32]

Figure 2.4: Types of Galaxies

Most massive galaxies in the universe are *spiral galaxies*, including the Milky Way. They are characterized by spiral shaped arms of star formation in their disks; some galaxies have more defined spiral arms and others are more disorganized. These arms are the result of gas being condensed by density waves. As the gas becomes denser, stars form, creating the brighter sections. *Elliptical galaxies* have a distinct round shape when viewed from any angle, ranging from roughly spherical to more elongated. Their structure is disorganized, and they rarely show evidence of large-scale star formation areas as seen the arms of a spiral galaxy. The majority of the stars within an elliptical galaxy are older and orbit haphazardly around the galaxy's center of gravity, creating the ellipsoidal shape that such galaxies are named

for. Many astronomers suggest that elliptical galaxies result from past collisions or interactions with other galaxies [58]. Some galaxies don't have a defined structure and can be extremely varied in shape. These are called *irregular galaxies*, and they are much smaller in mass relative to spiral or elliptical galaxy; they have little to no recognizable components like bulges, disks, or arms [8].

2.3 Stars

Stars are also a vital component to the visualization of galaxies and nebulae; they help situate these objects in outer space. Stars can also help approximate the distances involved between objects; in certain circumstances, the stellar distances need to be compressed to make them spatially understandable. Depending on the temperature of the stars, they can appear warmer or cooler in light. Massive, hot stars emit most of their light at short wavelengths such as blue and ultraviolet. These stars fuse hydrogen very quickly, in approximately 10 to 100 million years, before dying out. Lower mass stars fuse hydrogen at a slower rate, over many billions of years, and emit most of their light at longer wavelengths, appearing more yellow or red [4]. In this project, stars are handled as simple point light sources with varying brightness. The Hipparcos data set of 100,000 stars is used to capture a good subset of star characteristics [60].

3. RELATED WORK

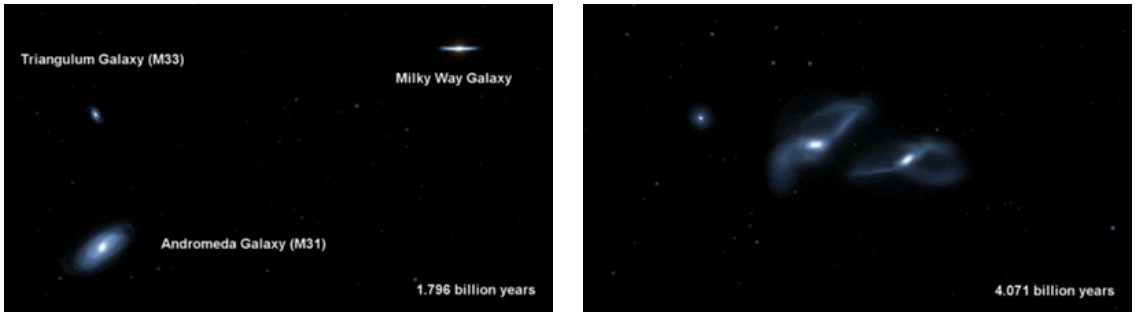
The field of astronomy has grown increasingly dependent upon computer generated imagery as research efforts have grown more complex. Astronomers attempt to match theoretical calculations to observed phenomena in the universe by using computers to simulate the object. More accurate and clear visualizations lead to better defined results. Inspired by the images captured by telescopes around the world, the entertainment industry has incorporated astronomical imagery into films, television, and interactive games. Advances in rendering natural phenomena such as clouds have created greater artistic control, allowing CG artists to sculpt these objects to their liking. New rendering algorithms have increased the speed of the calculations needed to be performed to achieve these effects. This section details work related to our splat generation, texturing, and rendering process, and each of the following three sections explains a focused influence upon this thesis. Section 3.1 describes astronomical visualizations for research, entertainment, and education as they have developed over recent years. Insight we can gain from rendering clouds is explained in Section 3.2. The history of splat rendering and a selection of algorithmic variations is highlighted in Section 3.3.

3.1 Astronomical Visualizations

In this thesis, we are attempting to combine the look and feel of computer generated effects for film and television with the scientific accuracy of traditional astronomical visualizations. It is helpful to review examples of such visualizations from those for research to visualizations for entertainment. Further, I will present some examples of astronomical visualization that attempt to combine both science accuracy and visual appeal.

3.1.1 Visualizations for Research

With the advent of digital imaging, the computer became one of the most pivotal tools in an astronomer's arsenal. An enormous amount of data has been captured from telescopes all over the world, creating the difficult problem of parsing and analyzing this information. When a particularly interesting object is observed, it is the astronomer's job to question and explain its existence and development. Using advanced physics and mathematical calculations, simulations can be made within the computer to test these hypotheses. These simulations have advanced exponentially over the last few years, culminating in somewhat visually appealing imagery.



(a) Before collision (b) After collision

Figure 3.1: Stills from Milky Way Andromeda Collision [31]

Astronomers can now reliably predict that the Milky Way and and our closest large neighbor, Andromeda, are on collision course that is expected to occur approximately 4 billion years from now (see Fig. 3.1). The findings resulted from precise measurements taken by the Hubble Telescope over the course of five to seven years. It will take a further 2 billion years for the galaxies to merge together, and the simulations indicate that our solar system will probably be thrown farther outward from

its current position. Collisions between stars in each galaxy are unlikely due to the large distances involved, but the night sky is predicted to change dramatically over the course of the interaction of the two galaxies. This collision was simulated by STScI and is simplistic in terms of visual style, but conveys a focused message to its audience concerning this new research development [31].

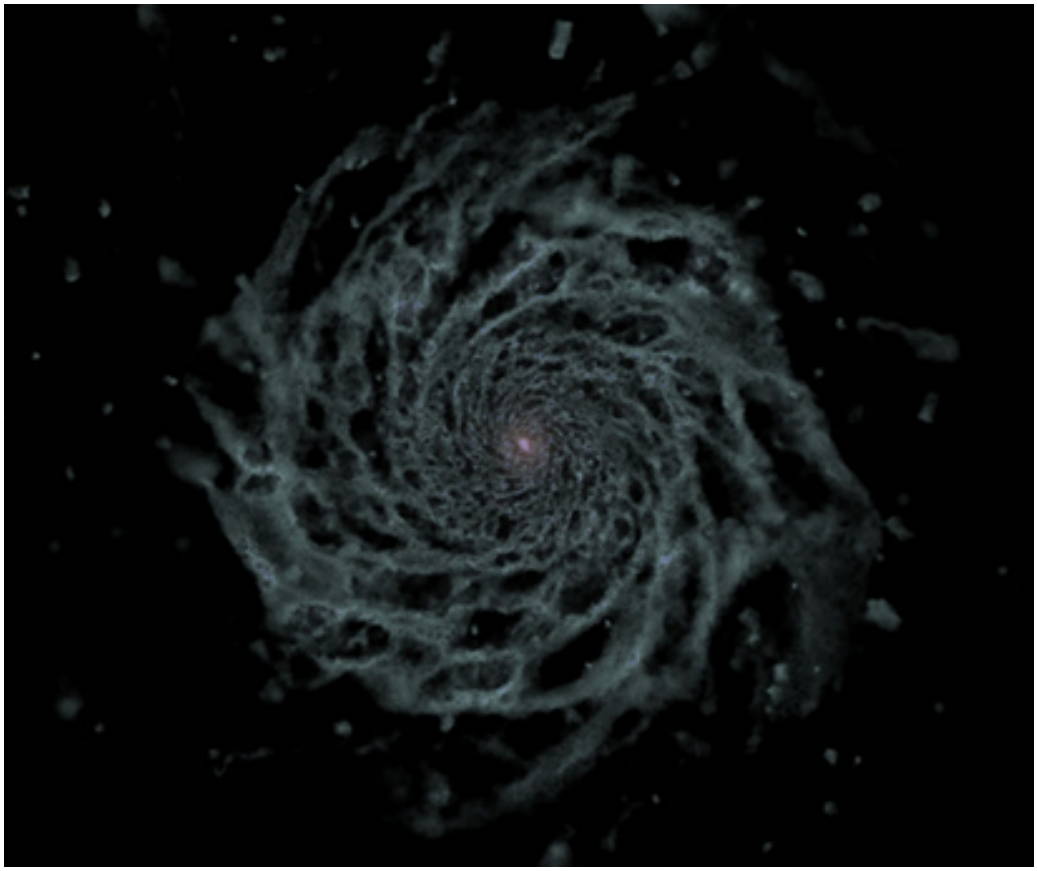


Figure 3.2: Eris Simulation [17]

Galaxy formation is another common area of research in astronomy. The Eris simulation took nine months of processing on NASA's Pleiades supercomputer, but upon finishing, it had created the most realistic and physically accurate representation of

a spiral galaxy (see Fig. 3.2) [49]. Previous simulations resulted in disproportionally large central bulges when compared to the target disk size; the new calculations incorporated a more physically correct approach for star formation. The simulation consists of over 60 million particles of dark matter and gas, and this high level of detail generated better results [18]. Other educational films about galaxy formation or movement are typically artist renderings, using scientific knowledge as the basis, but focusing primarily on developing the most dynamic and absorbing imagery [46].

3.1.2 Visualizations for Pure Entertainment

Film and television programs have begun to fashion their own universes using the computer. With faster computers and the development of new software, films have taken advantage of the wealth of inspiration available from astronomy. Using these images as research for their own galaxies and nebulae, studios have achieved beautiful results, that while not scientifically accurate, do manage to capture the viewers' imaginations.

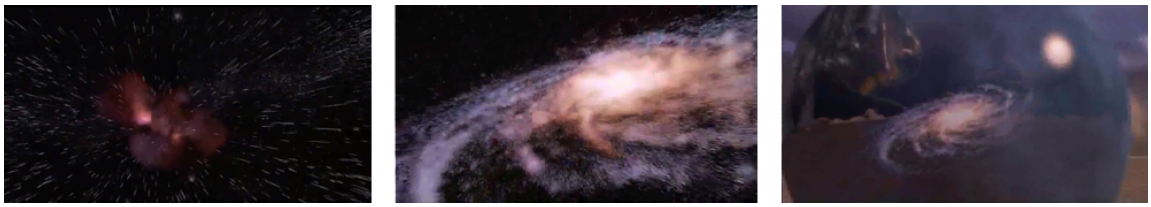


Figure 3.3: Astronomic Zoom Sequence Stills from *Men in Black* [47]

In 1997, Industrial Light and Magic ended their new alien film *Men in Black* with a technique called an *astronomic zoom*. An astronomic zoom is "an establishing shot that includes a wild planet-scale or even galaxy-scale zoom" [53]. This zoom is often used in astronomy films to help portray the size of the universe, but this was one

of the first mainstream films to successfully incorporate it. This camera move is reminiscent of the 1968 short documentary film "Powers of Ten" by Charles and Ray Eames [11]. Industrial Light and Magic created a CG solar system, nebulae, and a galaxy to pull off the shot, and it works well for the image quality of its time (see Fig. 3.3).

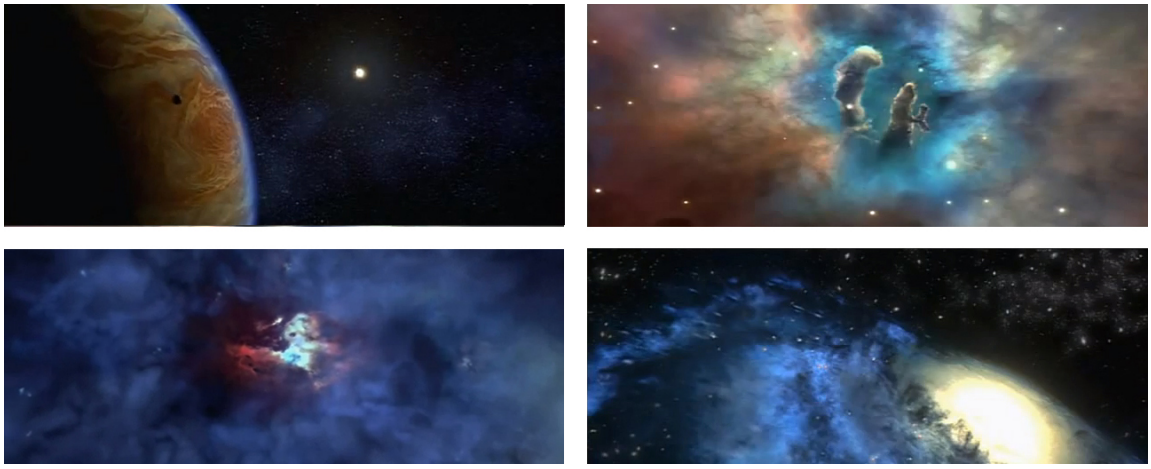


Figure 3.4: Opening Sequence Stills from *Contact* [62]

In the same year, *Contact* was released with a similar opening sequence. Based on a novel by Carl Sagan, the film attempted to design as scientifically accurate a film as possible while still telling an engrossing story. The camera zooms outward from Earth, passing the planets of our solar system, the Eagle Nebula, and finally a view from outside of our galaxy. While basing much of the detail scientific research, the actual camera move has some significant errors. In particular, the Pillars of Creation in the Eagle Nebula are presented as they would be seen from Earth, when they should be seen from the opposite side (see Fig. 3.4).

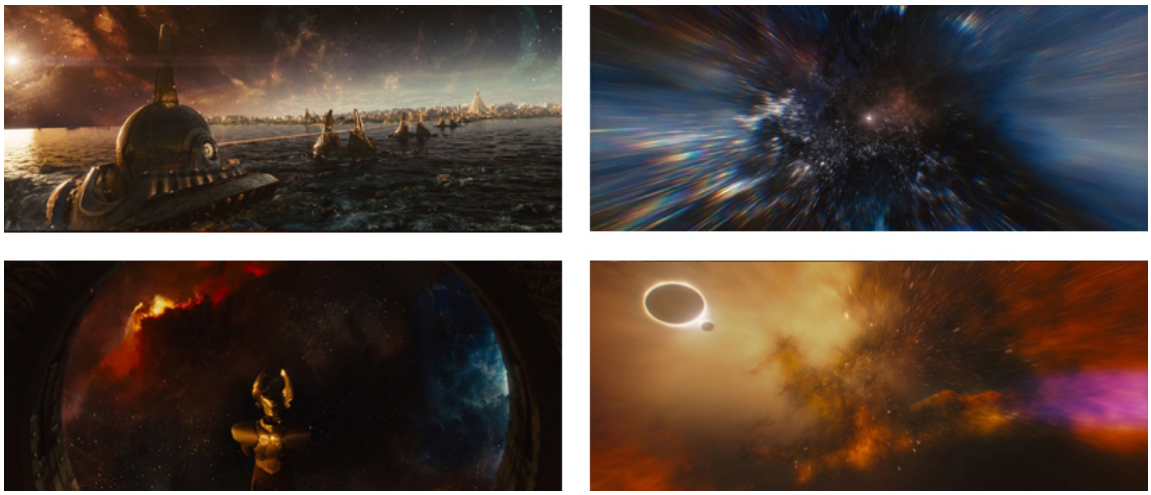


Figure 3.5: Production Stills from *Thor* [6]

The use of astronomical imagery in films was more successful in the visual effects created by French studio BUF for the movie *Thor* in 2011 (see Fig. 3.5). Asgard, the realm of the mythological Norse gods, exists on an island in the middle of space. To develop an imaginary, yet realistic section of the universe, BUF used images from the Hubble telescope combined with fractal imagery, Celtic mythology, and 3D particle effects [7]. The sky of Asgard is full of astronomical imagery taken straight from Hubble, Spitzer, and Herschel telescopes. The end credits feature nebular clouds going through the process of star formation, sculpted to appear infinite and chaotic [13].

3.1.3 Visualizations for Education

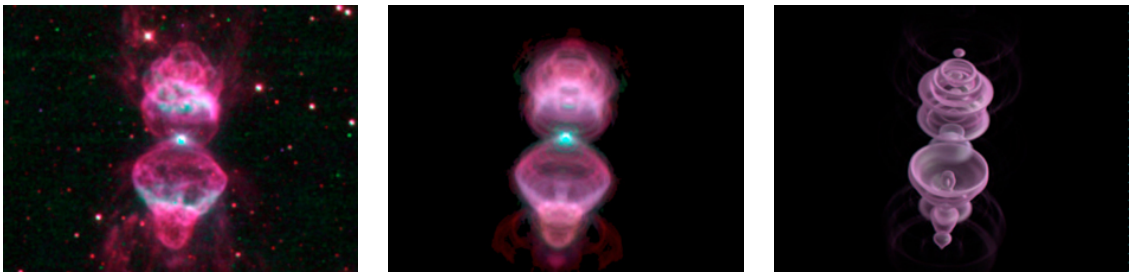
Some visualizations have attempted to blend accuracy and visual appeal together to educate in a more cinematic manner. Such research is ongoing, and it is primarily in this arena that our work is most applicable. Due to the distances involved, it is difficult to imagine the 3D structure of objects so far away. This process is further hampered by a single viewpoint and the ephemeral quality of the most visually

interesting objects. It is only through a thorough understanding of physics and astronomy that an educated guess can be made.

(a) (b)

Figure 3.6: Stills from Chrome's *100,000 Stars* [15]

Google recently launched a new Chrome Experiment called *100,000 Stars* based upon the stellar data collected by the Hipparcos Catalog. The interactive program allows viewers to zoom in and out of the approximately 100,000 stars that are within a few thousand light-years of Earth. The data uses level of detail controls to make the program run in real-time in the web browser. The positions of the stars are accurate, and the ability to explore the neighbors surrounding our solar system is an effective tool for both education and entertainment (see Fig. 3.6) [1].

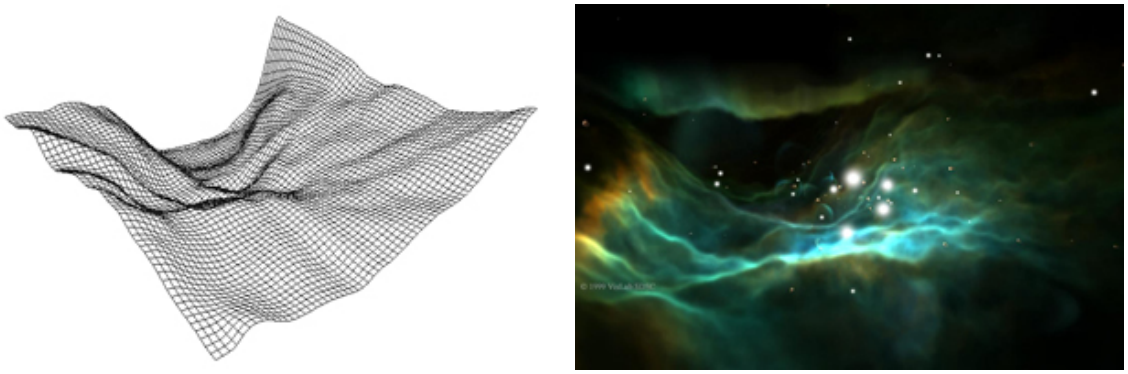


(a) Original image (b) 3D render (c) 3D model

Figure 3.7: Constrained Inverse Volume Rendering in Planetary Nebulae [21]

It is possible to exploit the symmetric nature of planetary nebulae to create 3D visualizations. As discussed in Section 2.1, planetary nebulae form from the gas released by dying stars. Magnor proposed a technique called *constrained inverse volume rendering*. The first step of this process is to use a general 3D model to determine the axial orientation of the nebula in relation to Earth. This 3D model is then rendered volumetrically and compared to the original astronomical image. A difference filter iteratively refines the 3D model into the correct placement and appearance (see Fig. 3.7) [21][22]. This technique utilizes the bi-polar symmetry evident in many planetary nebulae to make the 3D visualization problem more manageable. Other solutions are needed for more general nebulae cases.

The Orion Nebula is an H II region that has been studied in great detail. A 3D model was constructed after examining imagery from both Hubble and ground-based telescopes [56]. Nadeau et al. utilized this model in creating a volumetric render of the nebula for the Hayden Planetarium in 1999. Their solution was a based upon a *volume scene graph*. Similar to procedural texture functions, their solution generated a volume from the 3D mesh using functions that varied color and opacity based upon the distance from the surface (see Fig. 3.8) [23]. The entire volume was rendered



(a) Orion Nebula 3D Mesh (b) Orion Nebula Render

Figure 3.8: Volume Scene Graph Render of the Orion Nebula [23]

using a ray casting solution at the San Diego Supercomputer Center [14].

Further work was done with the Orion Nebula in the IMAX film *Hubble 3D* in 2010. The film explores the nebula using images from Hubble, bringing the camera as far into the structure as possible, giving audiences an upclose view of newly born stars and future solar systems. The original 300 million pixel image was used as the basis for the final texture map on a 3D mesh. From there, layers of particles were added to create a cloud-like surface for the nebula (see Fig. 3.9) [51]. Our work uses a similar initial mesh setup, yet explores a splat based, Z-buffer rendering approach in the hopes of increased speed and detail. As both nebulae and galaxies have cloudy components, studying CG cloud simulation and rendering is helpful in determining what properties create a cloudy appearance.

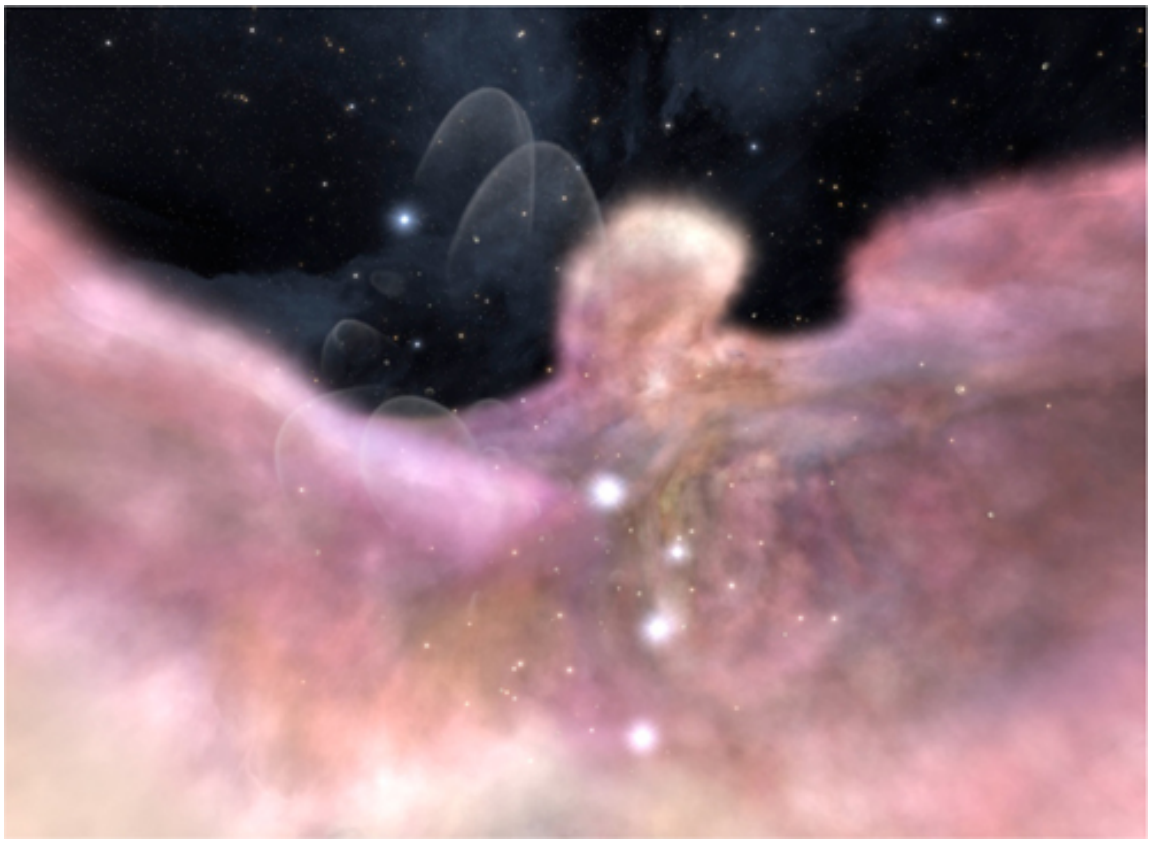


Figure 3.9: Particle Overlay Render of the Orion Nebula [50]

3.2 Rendering Clouds

Since the advent of computer graphics, a great deal of research has been done in recreating natural phenomena like clouds and smoke. The transparency, noise, and variable densities of clouds make simulating light transference through them very difficult and computationally heavy. In the case of our work, lighting is not necessary, as shadows and highlights are already built into the initial texture map. Methods of cloud simulation can provide insight into the texture choices for splats as well as the algorithms available for rendering. The following paragraphs review the two main rendering algorithms used to simulate clouds in computer graphics and

highlight what qualities of terrestrial cloud rendering can be helpful in constructing nebulae.

Image-order algorithms look at each pixel in the image and calculate what data is visible at that point. Traditional ray casting and ray tracing algorithms fall into the image-order class. In cases where you have opaque objects, image-order algorithms allow you to mostly ignore the objects that are not in view, resulting in a shorter render time. Fully or partially transparent objects present a greater challenge. A ray tracer must shoot a ray into a volume, such as a cloud, calculating light contributions at each step, until the color becomes opaque [19].

Object-order algorithms transform the data into the image plane and calculate in which pixels the data should be placed in the final image. This requires the renderer to touch every element in the render set, which can cause slower render times; however, some elements can be eliminated using judicious use of visibility culling [19]. In the case of volumetric data, an object-order algorithm doesn't require a ray marching solution, as the color simply builds up from back to front. Examples of object-order algorithms are the painter's algorithm and Z-buffer algorithm [41]. For this thesis, I used a Z-depth based splat rendering system as opposed to a ray casting solution as our process requires very large particle sets which are more efficiently rendered using an object-order algorithm.

In the late 1970s, initial tests of simple object representation and animation had been completed; however, even displaying this data was taxing to the computers of that time. Csuri et al. continued this work with the Computer Graphics Research Group (CGRG), searching for faster and more complex rendering algorithms as well as possible solutions to represent non-solid objects similar to clouds, fire, and running water. In their experiments, a point-based representation was used to simulate smoke. Each point had its own set of parameters, including position, orientation,

color, and opacity. Large numbers of points and the use of calculations that take into account each point's nearest neighbors generated more visually accurate results [9].

The first steps in rendering clouds were taken by Blinn in Light Reflection Functions for Simulation of Clouds and Dusty Surfaces. In his work with NASA's Jet Propulsion Laboratory, Blinn simulated a fly-by animation of Saturn; the rings of Saturn prompted his research into how light interacts with materials such as dust and clouds. His paper presented a statistical model of how light passes and scatters through a cloud of small, similar particles. These functions formed the basis of modern subsurface scattering algorithms used today [5].

The introduction of particle systems by Reeves targeted the modeling aspect of objects like fire and clouds. Particle systems can create fluid and dynamic objects, and sculpting the general form is possible with stochastic functions. Reeves uses each particle as a point light source, recommending that future work treat them as individual light reflecting objects for cloud and smoke rendering. Although this research does not touch on clouds specifically, it does acknowledge that cloud modeling using particle systems would be a natural, if more complex, extension of particle systems. Cloud modeling is a particularly involved process, as cloud shapes depend upon multiple forces like wind direction, temperature, humidity, and terrain. In the case of nebulae, most motion is indiscernible to astronomical instruments, so it is primarily the overall shape and rendering choices that will successfully recreate the 3D form [42].

Continuing the idea of particle-based clouds, Wang created a system that produces random clouds using textured sprites. By combining sixteen different cloud textures, they were able to generate many types of clouds, from the wispy cirrus to fluffy cumulonimbus (see Fig.3.10). Using these as reference, we can identify what

types of noise and texture patterns will be most successful in creating the feel of a specific nebula. The first texture is similar to the shape and falloff of the splats that were used in this work. There has also been a lot of work in understanding how light passes and scatters through a cloud [55] [43]. Although the self-shadowing and lighting of clouds is vitally important, by working from astronomical images which already consist of highlights and shadows from nearby stars, it is not necessary to consider the shading methodology for a nebula. Clouds have been rendered using textured particles and sprites, and volumetric splats are natural extension of those methods.

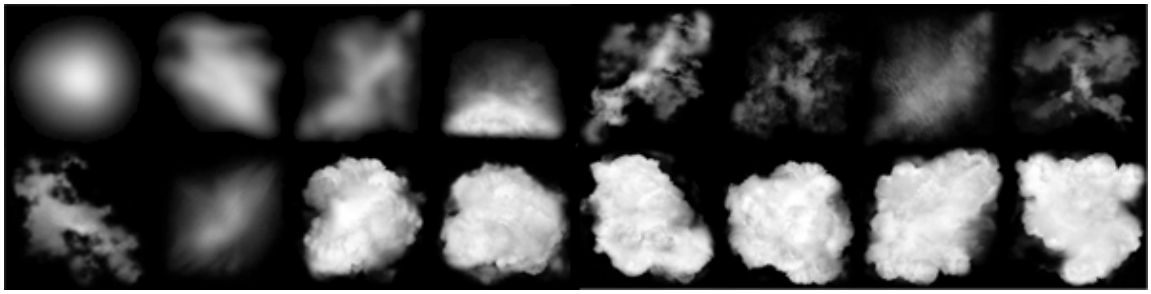


Figure 3.10: 16 Cloud Sprite Textures [55]

3.3 Splat Rendering

Efficient rendering algorithms are vitally important for visualizing volumetric data. Splat rendering is a technique that works well for highly detailed objects since they can be scaled up or down to fit any sized detail; it was initially designed for medical, astronomical, and geological data visualizations [57]. By using splats, we can take advantage of previous cloud rendering research that used similar data types such as points, particles, and sprites. The next pages summarize the splat rendering research relevant to the work of this thesis.

In the search for efficient rendering algorithms, Westover proposed a method that would be fast enough to interactively render volume data. *Volume rendering* is data sampled in the three dimensions. Previous volume rendering research preprocessed the 3D data into a solution that will work with more common line or polygonal renderers. The key to the efficiency of Westover's solution is the use of precomputed lookup tables that eliminate calculations at render time. Specifically relevant to our work, is his introduction of splats as a visualization primitive. As mentioned before, a splat is a point with a convolution filter applied to it. In order to increase speed, the weights of the filter's kernel are calculated before render time and stored in a lookup table. Westover defines *splatting* as the process of looking up the kernel values in a table, weighting the samples of the point, and combining the new data into the image buffer. The process also implements a object-order rendering solution, sorting the data points in Z-space and calculating either front-to-back or back-to-front [57].

Linsen et al. explored the possibility of using an image-order solution in combination with splats. Given a surface represented with a point cloud, this method uses splats to represent the surface instead of polygons. Neighborhoods of points are determined in a preprocessing step, and splats with varying radii are calculated to cover each neighborhood. A ray tracing algorithm requires smoothly interpolated surface normals to display a smooth photorealistic surface. Surface normals are estimated at each point and then adjusted to create a smooth field of normals across the surface of the splat. All of this splat and normal data is given to a simple ray trace renderer. Rays are shot into the data, and intersections with the splats are found; if the ray intersects with two overlapping splats, the intersection point that is closest to the center of its respective splat is used. This eliminates most rendering artifacts that occurred in overlapping areas. A higher sampling rate was also used when aliasing became an issue. This process works well for visualizing dense point

clouds, but the end results are opaque surfaces; our work will require transparent splats, therefore an object-order algorithm appears to be the most efficient solution [20].

For the *Digital Michelangelo Project*, Rusinkiewicz and Levoy implemented a system called QSplat to visualize 3D scan data collected from Michelangelo's sculptures. 3D scanners create point cloud results of tremendous size and detail. For the *Digital Michelangelo Project*, models contained anywhere from 100 million to 1 billion samples. In order to make their system interactive, they used a bounding sphere hierarchy to store the splat data during rendering. The hierarchy is calculated before rendering time, and allows for efficient frustrum culling, backface culling, and level-of-detail control. Each node in the tree stores information on sphere center, radius, normal, width of the normal cone, and an optional color. During rendering, the algorithm performs visibility culling first, flagging each node as visible or hidden. Then, each node is checked for visibility as the tree is traversed. The depth of the traversal is determined by a heuristic that is designed to choose a level-of-detail that will maintain a constant frame rate on the display machine. Splats are used as the renderable primitive, and the Z-buffer is used to check for splats occluding those behind [44].

Splat rendering can be performed more efficiently by using visibility algorithms, like frustrum culling and backface culling [16] [19]. Frustrum culling uses the angle of view of the camera to identify what objects are in the camera's image plane and removes the extra data. Backface culling is helpful with opaque objects, as it performs a check to see if the surface normals of an object are facing toward the camera; if they are not, those faces can be ignored during the rendering process. As my splats are semi-transparent, only frustrum culling can be used to remove unnecessary splats. Splats can also cause aliasing issues. It is the paradoxical nature

of a splat that overlapping will cause visual errors, yet overlapping is the only way to achieve a smooth appearance. Swan addresses this issue using an integration grid [52].

4. PROCESS AND IMPLEMENTATION

4.1 Process Overview

In this section we present the three main stages in our process of generating splat-based scientific visualizations.

1. *Mesh Modeling and Texturing:* A 3D mesh for the astronomical object must be estimated based upon various characteristics such as density, composition, emission, absorption, stellar winds, and photo-ionization. Images taken from both space and ground-based telescopes are combined to create a texture image which is mapped to the 3D surface.

2. *Splat Generation:* Once the mesh has been determined, a bounding box is calculated around it. From there, the volume is voxelized and jittered to create test positions. For each test position, a check is performed to see if the test position is within an allowable distance. If so, a splat is generated containing data for position, color, opacity, and distance from the surface.

3. *Splat Rendering:* Based upon the distance of the splats from the mesh surface, modifications are made to the splat set until the desired look is achieved. Splats are created in multiple sets to allow maximum artistic control. The splats are then passed into the render code. The code outputs TIFF images that can be composited and edited into the complete sequence.

4.2 Mesh Modeling and Texturing

The first step in our process is to prepare the 3D mesh that the splat generation will be based upon. The sculpting of this mesh will be the foundation that allows

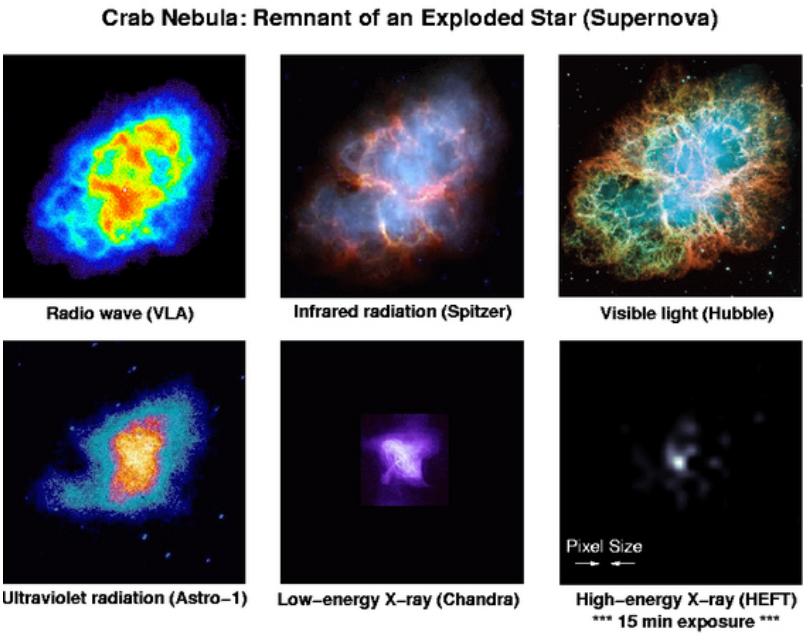
the nebula or galaxy to have structure and volume. The theoretical physical form of the object must be researched so that the final mesh is as accurate as possible. Then, the mesh must be textured using various telescopic images. In the case of a galaxy, multiple textures for the stars, dust, and gas, are needed to allow for artistic control.

4.2.1 Mesh Modeling

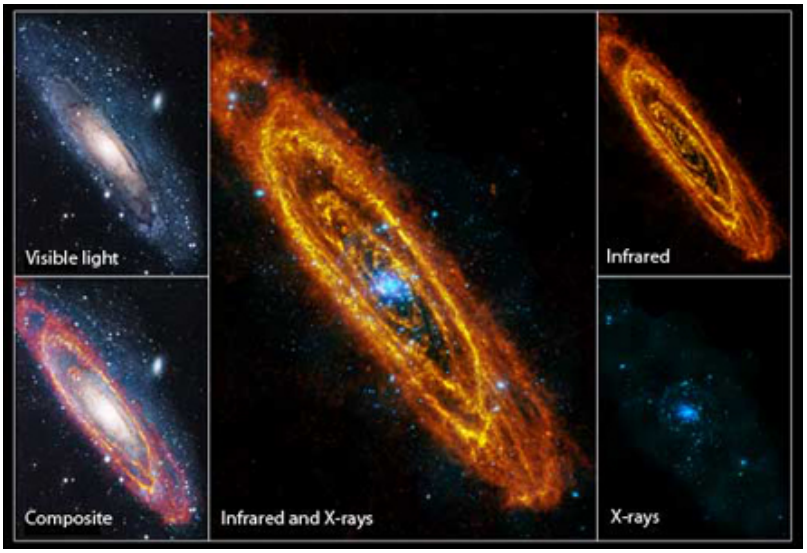
In modeling a nebula, the first step is to conduct research into what astronomers and physicists have estimated to be the three dimensional form. Most of this research is obscure to the average person, requiring a substantial scientific background for comprehension. With a general idea of the 3D structure, a mesh can be sculpted from an initial plane. For some nebulae, it is possible that multiple meshes, perhaps a foreground and a background, may be a more logical solution. The modeling process is more straightforward for a spiral galaxy, as a simple textured plane can be used. The thickness of the disk and bulge of the galaxy can be sculpted with noise functions and additional texture maps as necessary. The only modeling restriction for any mesh is that it must be triangulated for later stages in the process. Without access to a sculpted mesh built by scientists, input from an astronomer would greatly enhance the scientific accuracy of the formation of any original meshes. For the purposes of this thesis, I used meshes developed by the visualization team at STScI.

4.2.2 Texturing the Mesh

Astronomers use different wavelengths of the electromagnetic spectrum to gain further insight into the structure of galaxies, nebulae, and supernovae (see Fig. 4.1). For example, in visible light, the shield of dust and gas of a nebula might obscure bright stars, additional nebulae, or even new galaxies; however, when an infrared image is taken of the same area, new information is revealed. Gamma and X ray

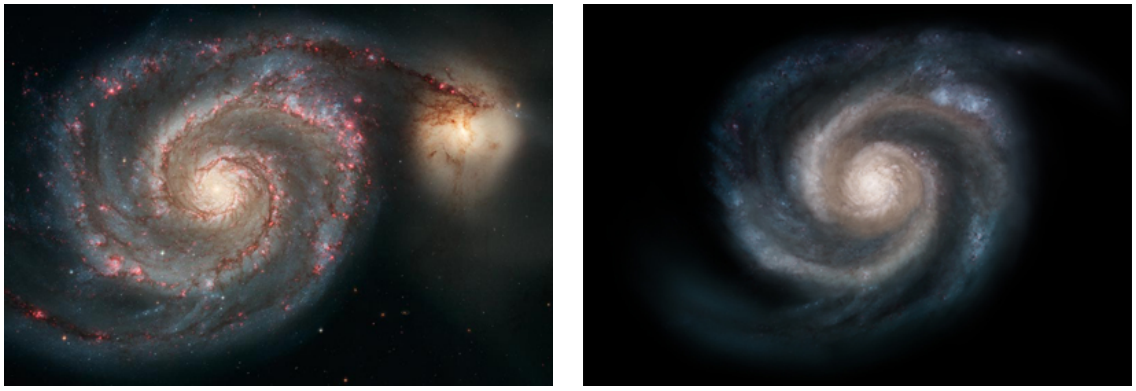


(a)



(b)

Figure 4.1: Objects at Different Wavelengths of Light: (a) Crab Nebula [2], (b) Andromeda Galaxy [45]



(a) (b)

Figure 4.2: Whirlpool Galaxy Image Processing: (a) Original visible light image [30], (b) Processed star texture image

wavelengths are helpful in exposing bright, hot stars and supernova remnants. The same process has been done with other galaxies, including our own. Its possible to isolate cooler areas of a galaxy, such as the dust, in an infrared wavelength. This makes creating separate texture layers for different disk components much easier.

Once images have been selected that provide the most detailed information about the object, they must go through some initial preprocessing steps. All of the wavelengths must be aligned so that their features line up; different telescopes take images at coarser or finer resolutions, so there is an expected quality exchange during the alignment process. Next, foreground and background stars and galaxies need to be removed from all of the images so that the object is isolated. By using image manipulation software, an artist can paint a mask that can be applied to all texture layers. After removal, some holes might need to be filled in with digital painting techniques. If the mesh has been separated into multiple pieces, for example a background and a foreground, the texture must be split as well; the overlapping sections will require holes to be filled in based upon scientific intuition and aesthetic appeal. Alpha must be added to the new texture layers where necessary (see Fig. 4.2); again, scientific

experience is extremely useful here. While some modifications can be made to the alpha after splats have been created, it is best to get the initial values as close as possible at the beginning of the process.

4.3 Splat Generation

Once the mesh (or set of meshes) has been modeled and textured, the next part in our process is the generation of volumetric splats. This stage consists of three main steps, generating test positions, culling test positions, and assigning initial color and alpha values to valid splats. The 3D meshes were chosen to be exported as OBJ files, as this allows for a quick processing of the mesh using the inherent structure of OBJ files (see Fig. A.1). Fig. 4.3 shows the test position generation process using a hypothetical mesh. A bounding box is formed using the minimum and maximum vertex values for all three axes. The bounding box is expanded to include a user-defined buffer to allow splats to be positioned above or below the surface. This provides flexibility in terms of the thickness of the object. The bounding box is then voxelized to a density defined by the user. Next, test positions are created at the centers and corners of every voxel. The test positions are jittered slightly to create a more natural distribution before being written out to a text file. This voxelization and jitter process was chosen to generate the test positions as it ensures a evenly distributed density of splats; Monte Carlo solutions occasionally result in areas of uneven distribution that could create odd patches of particles.

To create a set of textured splats, each test position is analyzed with respect to the 3D mesh and texture. Fig. 4.4 illustrates our culling and color assignment algorithm using a hypothetical mesh and test position set. A script is run to find the closest point on the mesh to each position, and to check if it is within the maximum distance parameter specified by the user. If a test point is valid, then a texture

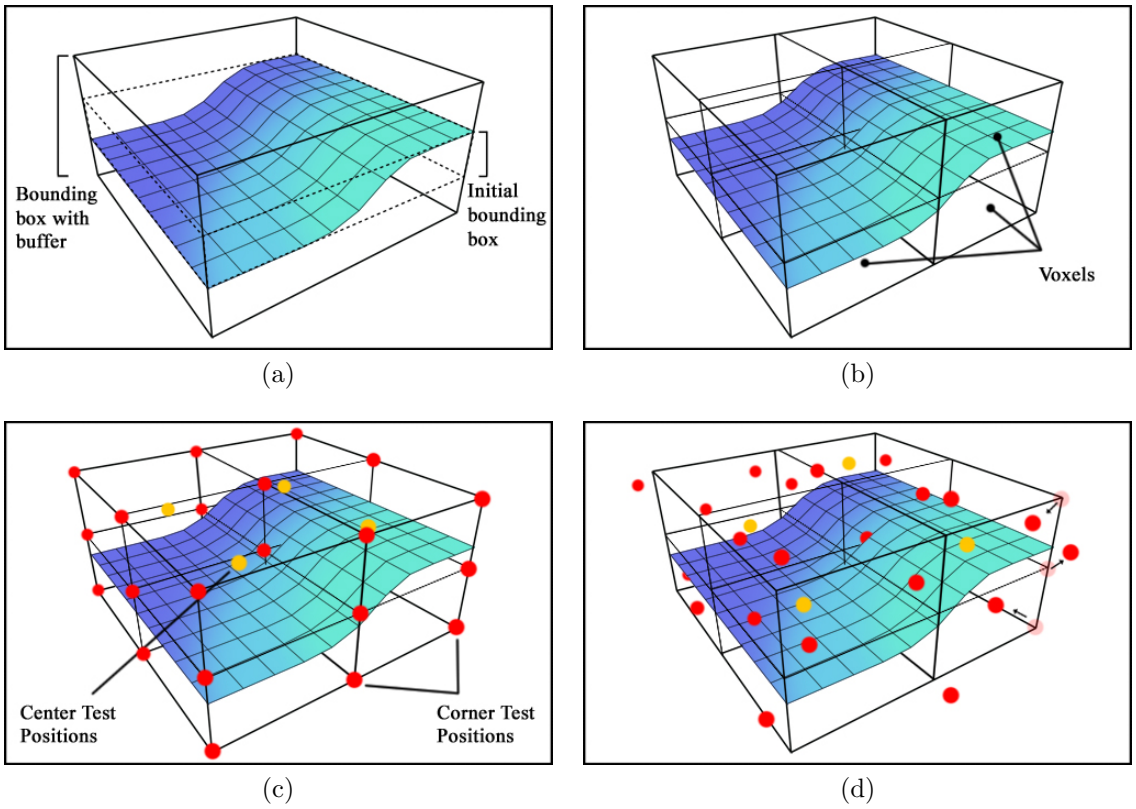
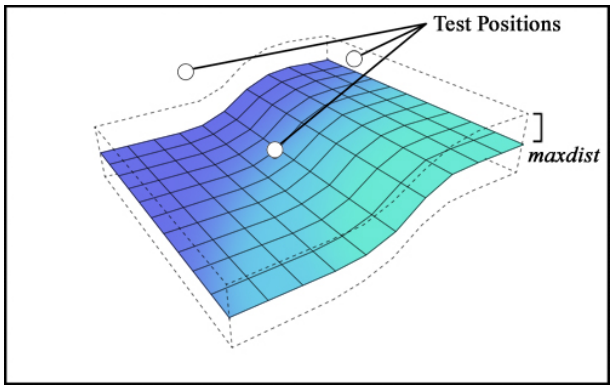


Figure 4.3: Generating Splat Test Positions: (a) Mesh with bounding box and buffer, (b) Voxelized bounding box, (c) Corner and center test positions, and (d) Jittered test positions

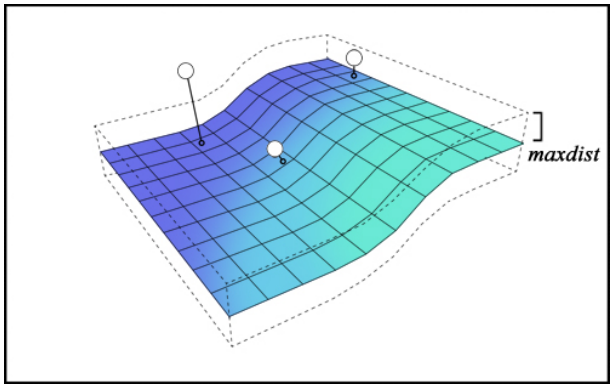
lookup is performed and color is assigned to that splat.

The closest point on a surface to any point can be computed by finding the closest point among all vertices, edges, and faces to the test point. The closest of these is the shortest overall distance from the surface mesh to the test point. For this check to work, each face must be planar, so the mesh must be triangulated.

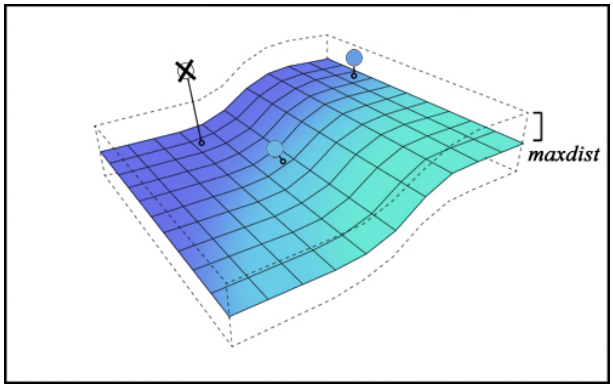
To find the closest vertex to the test point we find the distance from that point to each vertex in the mesh. The equation to find the distance D between points $p_1 = (x_1, y_1, z_1)$ and $p_2 = (x_2, y_2, z_2)$ is below. The distance between the test point and closest vertex is saved, as is the vertex's coordinates. By saving the closest point



(a)



(b)



(c)

Figure 4.4: Creating and Texturing Splats: (a) *maxdist* parameter with respect to mesh, (b) Measuring distance between test positions and closest point on mesh, and (c) Assigning RGBA to viable splats

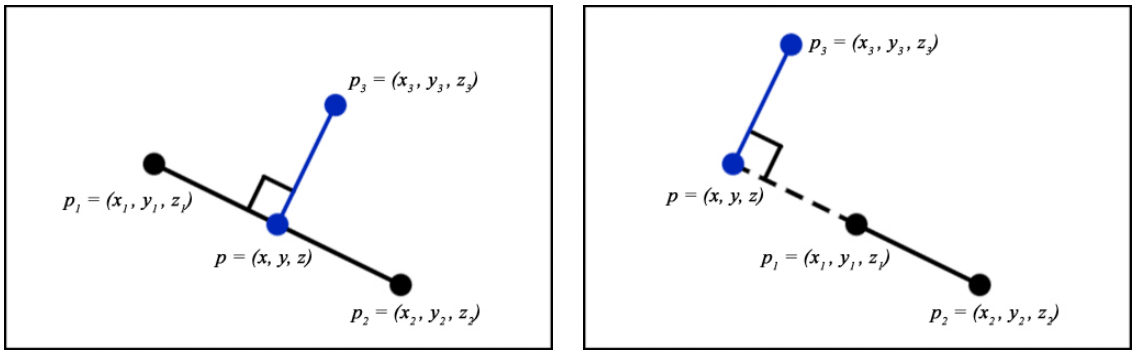


Figure 4.5: Checking if the Point is within the Edge: (a) Closest point within the edge, (b) Closest point outside the edge

on mesh, we can access its texture coordinates and perform a color lookup later in the process.

$$D(p_1, p_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2} \tag{4.1}$$

The next step is to check all edges for the shortest distance to the test point. An edge is simply a segment of an infinite line in space. For each edge, we find the closest point p on the infinite line defined by the edge. If p lies within the edge, then it is a possible candidate for the closest point on the mesh and we can compute D from the test point to p (see Fig. 4.5). If this new D is less than the currently saved D, the new value is stored and the closest point on mesh coordinates are updated.

Below are the equations for calculating the closest point p on an edge.

- Let p_1 and p_2 be the vertices of an edge.
- Let p_3 be the test point from which we are calculating.

$$p = p_1 + u(p_2 - p_1) \tag{4.2}$$

$$(p_3 - p) \cdot (p_2 - p_1) = 0 \tag{4.3}$$

Equation 4.2 defines the infinite line that the edge lies on. u is a parameter along that line. At p_1, u is 0, and at p_2, u is 1, such that if $u \in [0, 1]$, then p is in the line segment between p_1 and p_2. Equation 4.3 states that the vector defined by $(p_3 - p)$ is perpendicular to the vector defined by $(p_2 - p_1)$. This holds true because the shortest distance from a point to a line $(p_3 - p)$ is always perpendicular to the line direction $(p_2 - p_1)$. To calculate p on an edge, it is necessary to solve for u by substituting Equation 4.2 for p into Equation 4.3, as seen below.

$$u = \frac{(p_3 - p_1) \cdot (p_2 - p_1)}{(p_2 - p_1) \cdot (p_2 - p_1)} \tag{4.4}$$

If $u \notin [0, 1]$, then p is not within the line segment. Otherwise, p is within the edge and D can be calculated and updated if necessary.

Determining the closest point among all faces is the last step in the process. Similarly to the edge case, we can find the closest point on the infinite plane on which the face lies and test if this point is within the triangular face of the mesh. Below are the equations needed to calculate the closest point p on a face.

- Let p_0, p_1, and p_2 be the vertices of a face.
- Let p_3 be the test point from which we are calculating.

$$n \cdot (p - p_0) = 0 \tag{4.5}$$

$$p = p_3 + n_3 u \tag{4.6}$$

Equation 4.5 is the equation for the infinite plane with normal n on which vertex

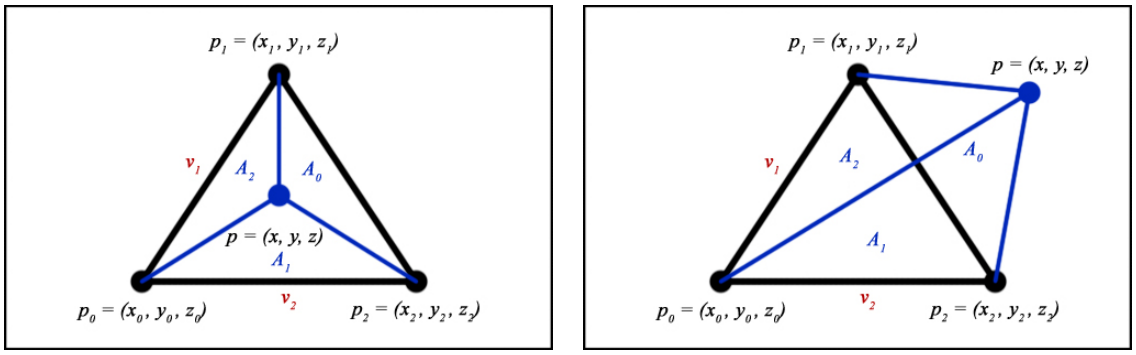


Figure 4.6: Checking if the Point is within a Face: (a) Closest point within the face, (b) Closest point outside the face

p_0 lies. Our triangular face is a subset of this infinite plane. Equation 4.6 defines a ray which originates at point p_3 and travels in the direction n_3, and u is a parameter along this ray. The point on a plane that is closest to p_3 will be a projection of p_3 onto the plane in the opposite direction of the plane's normal. Thus, $n_3 = n$. If u is positive, the point along the ray it describes is in the direction of the infinite plane. If u is negative, the point on the ray is behind p_3. To calculate p on a face, first solve for u by substituting Equation 4.5 into Equation 4.6, as seen below.

$$u = \frac{n \cdot (p_0 - p_3)}{n \cdot n_3} \tag{4.7}$$

If $u < 0$, then the plane is parallel to or behind the ray, and the face is not a candidate for the closest point on the mesh. If $u > 0$, then p must be checked to see if it lies within the face; this is possible by looking at the triangular areas formed by each of the edges and p (see Fig. 4.6).

To determine if p lies within the face formed by p_0, p_1, and p_2, we calculate two

vectors, v_1 and v_2, which define two edges of the face.

$$v_1 = p_1 - p_0 \tag{4.8}$$

$$v_2 = p_2 - p_0 \tag{4.9}$$

The area vector, A, is the cross product of v_1 and v_2.

$$A = v_1 \times v_2 \tag{4.10}$$

The area of a triangle can be defined as half the magnitude of the area vector.

$$area = \frac{1}{2}|A| \tag{4.11}$$

In the same manner, we can define area vectors for the smaller triangles formed by the edges and p, the intersecting point in the plane calculated earlier.

$$A_0 = (p_1 - p) \times (p_2 - p) \tag{4.12}$$

$$A_1 = (p_2 - p) \times (p_0 - p) \tag{4.13}$$

$$A_2 = (p_0 - p) \times (p_1 - p) \tag{4.14}$$

By checking the direction of each of these area vectors, we can determine if p lies within the triangular face. If all of the vectors face the same direction, then p is within the face; otherwise, p lies outside of the face. We can test if all of the vectors face in the same direction by comparing the signs of the same non-zero component of each vector. Once p is determined to be a valid point within the face, D can be computed and compared to the saved D from the previous vertex and edge tests.

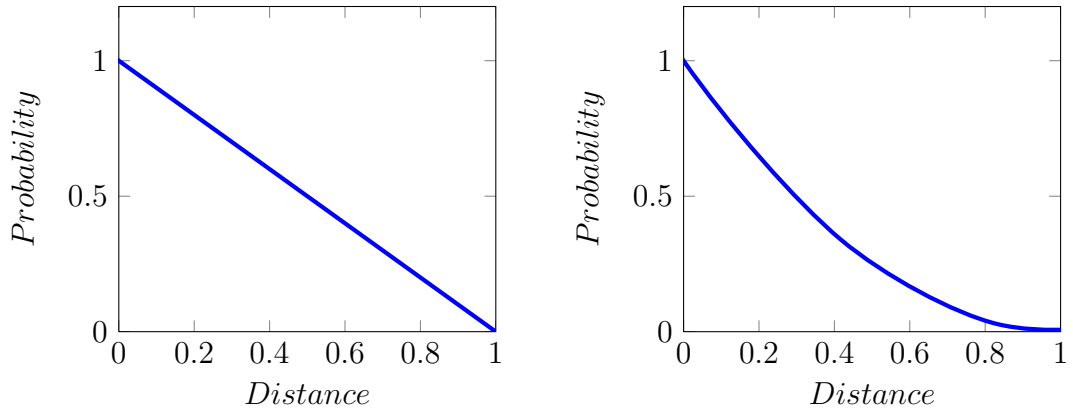
Now that we have checked all vertices, edges, and faces in the mesh for the shortest distance to the test point, we can check its length against the maximum distance parameter specified by the user. If the test position is within this maximum distance, a color lookup is performed using the texture coordinates of the closest point on the mesh. Position, color, opacity, and distance information is stored and written out in a native Maya file format called PDA. The full algorithm for this culling and texturing process as well as the format for a PDA file are available in Appendix A (see Fig. A.3 and Fig. A.2).

4.4 Splat Rendering

The final step in our process is rendering the splats. Depending upon the look desired, some modifications may be necessary. For my test cases, I developed some utility programs that perform some of these adjustments. I will present the general changes that benefited this work, although they would need to be tweaked for new cases. A C program initially developed by Dr. Frank Summers from STScI was used to render the test cases. Alterations to this code were discussed with Dr. Summers prior to him implementing them. Specific information on look development and rendering choices for the test cases is discussed in Section 5.

4.4.1 Look Development Modifications

Cloudy objects tend to have a sparser density along the outskirts of their forms, helping to produce a softer overall shape. After the initial cull, the splats have a consistent density throughout the thickness of the object, causing it to have a constant silhouette without soft edges. A probability density function was used to accomplish a natural drop off of splats as we move farther from the initial surface mesh. This function calculates the probability that a splat would exist as its distance from the surface mesh increases. Multiple functions were implemented, allowing the



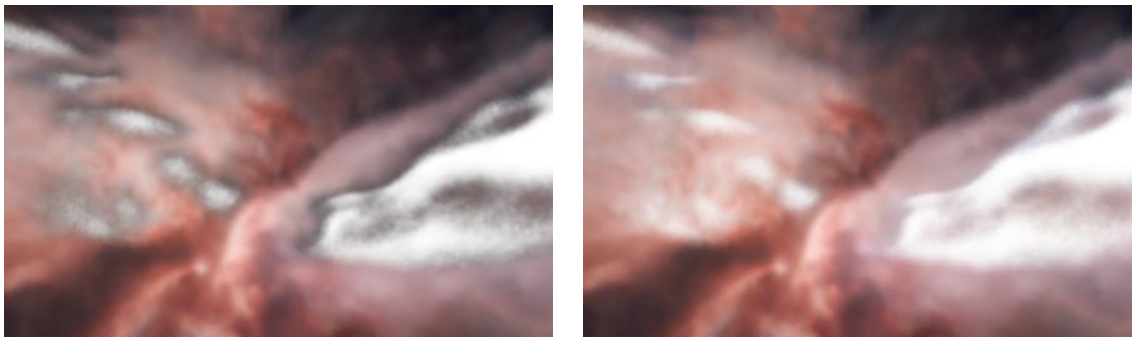
(a) $y = 1 - x$ (b) $y = (x - 1)^2$

Figure 4.7: Examples of Probability Density Functions

user to choose the silhouette that best approximates the object they are trying to recreate (see Fig. 4.7).

Besides a density modification, an opacity falloff is also necessary as the splats reach the boundaries of the object. Originally, a simple falloff function was used to reduce the alpha where the splats were farther from the mesh; however, this only results in a softer appearance in the Y axis of the splat set. Also, in nearly invisible sections of the texture map, the alpha values build up, causing these colors to become too prominent. Since most astronomical images are matted on black, and thus are premultiplied, this creates cloudy black edges along semi-transparent areas (see Fig. 4.8). For this reason, I chose to separate the opacity changes into two categories, horizontal and vertical.

Horizontal opacity modifications are necessary in the X and Z axes to solve the issue of dark edges. My solution was a combination of unpremultiplying the colors of the semi-transparent areas and scaling the original alpha values. In the original Hubble image, the colors are initially premultiplied with the alpha; this causes the pure hue to be diluted in value to black. By dividing the RGB values of the color



(a) (b)

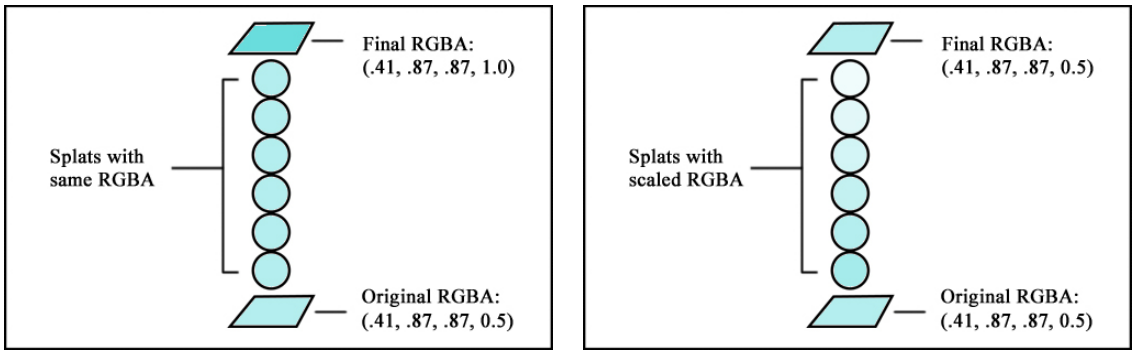
Figure 4.8: Horizontal Opacity Modification: (a) Dark edge issue, (b) After unpremultiplying color and rescaling opacity

by the alpha, we can return the color to its original value. To scale the alpha values, the original alpha is raised to a new power. The higher the power, the sharper the falloff into transparency.

Vertical opacity modifications are needed in the Y axis to create a smooth transition to transparency as the splats get farther away from the surface mesh. Each of the splats along the Y axis is assigned to a unique pixel in the original image, creating a column of splats with the same color and opacity attributes. This creates a build up of density that results in an alpha value that is greater than the original texture map. To get the correct alpha values, another form of scaling is needed (see Fig. 4.9).

The object has an initial density determined by the number of voxels in the Y axis. We can estimate the final alpha value by stepping through this density and accumulating the alpha at each stage. This calculation is demonstrated by the equation below.

$$\sum_{i=min}^{max} pdf(distance) * alpha\_func(distance, alpha) \tag{4.15}$$



(a) (b)

Figure 4.9: Vertical Opacity Modification: (a) Splats with original alpha, (b) Splats with density scaling applied

The user defines a scaling function for the alpha based upon a splat's distance from the surface; typically, an exponential falloff works well. The density has been previously modified by a probability density function, making it more likely that splats exist closer to the surface. Thus, the splats can be weighted based upon distance using the same probability density function. Once a summation of these values has been performed, a scale factor can be calculated using the original alpha of the splat as a target. When the new alpha for the splat is calculated, it is multiplied by this scale factor to ensure that the final alpha sum is close to the original texture map value. The algorithm for this calculation is available in Appendix A (see Fig. A.4).

Following the culling of the test positions, the splat set has attributes for color, opacity, position, and distance. The addition of a radius size to the splats can also aid in creating both variety and a smooth falloff into transparency along the edges. By setting a minimum and maximum range and then finding a random value within that range, a more natural appearance can be established. Larger splat sizes have a soft, gradual transparency falloff, while smaller splat sizes allow for greater detail. Therefore, instituting an interpolation from smaller to larger radii in both

the horizontal and vertical directions of the cloudy object will smoothly transition detailed areas into softer, more transparent edges. In the horizontal direction, the size change is motivated by the modified alpha values, increasing in size as the alpha approaches zero. The vertical modification is driven by the distance parameter, enlarging the radii as the distance gets larger to produce more diffuse and softer splats located at the extremes of the Y axis.

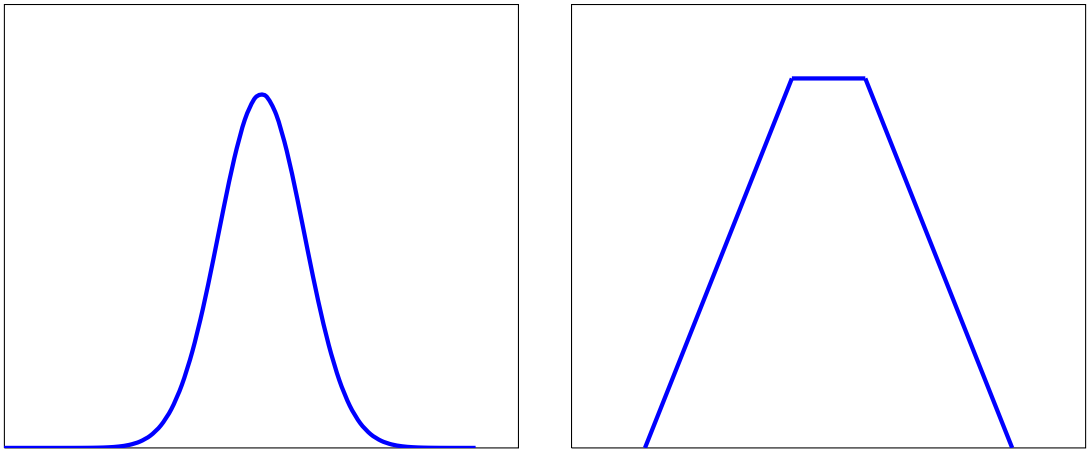
Further variety can be developed with the addition of fractal noise; fractal noise was chosen as it provides the most opportunities for generating naturalistic phenomena like clouds . A noise function can be determined by interpolating between values of a seeded random number generator. Fractal noise is made by adding multiple noise functions with different frequencies and amplitudes together [12]. This noise can be applied to any splat attributes, though it was primarily utilized in my test cases to create variation in the alpha and the density.

4.4.2 Rendering Modifications

The rendering code for this work was developed by Dr. Frank Summers at STScI; it is a continual work in progress, and has undergone many alterations over the course of this thesis. The modifications were designed to achieve a specific visual quality, but they also were implemented to enhance efficiency. The rendering time can be estimated using the number and size of the of splats. If we let num be the number of splats in the render, and $size$ be the average size of the splats on the screen in pixels, then the render time can be given by:

$$time \propto num \times (size)^2 \tag{4.16}$$

The number of splats in the set can be somewhat expensive, but it is their size on the screen that causes the render time to increase the most substantially. The code



(a) Gaussian Falloff (b) Trapezoidal Falloff

Figure 4.10: Examples of Splat Falloffs

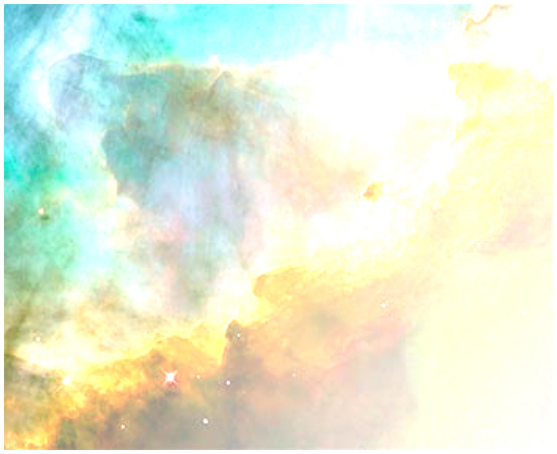
also incorporates many visualization parameters that can be manipulated to get the desired look. I will cover the major adjustments I collaborated on with Dr. Summers in the following pages.

The convolution filter applied to a splat can be a myriad of functions. A constant falloff would result in a splat that is the same color and opacity across its entire radius, or a basic dot. Other functions such as linear, exponential, and gaussian were implemented as well. Gaussian and exponential filters require a larger area to calculate over and can be rather intensive to compute; thus a more cost effective method was implemented. The *trapezoidal* falloff combines the constant and linear equations together to create a similar visual effect as a gaussian falloff. The user specifies the size of a constant core and a linear falloff is calculated for the edges (see Fig. 4.10). This results in a similar appearance to the gaussian falloff, but requires less computation.

The rendering code uses a painter's algorithm approach to render the data set. Splats are added to the image buffer from back to front and composited on top of each

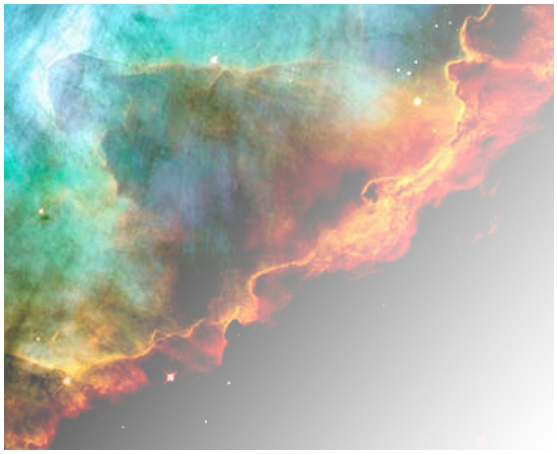
(a) Top Image

(b) Bottom Image



(c) Addition: $f(a, b) = a + b$

(d) Screen: $f(a, b) = 1 - (1 - a)(1 - b)$



(e) Lighten: $f(a, b) = max(a, b)$

Figure 4.11: Compositing Modes

other. The traditional compositing equation functions as a simple over operation; however, there are many different compositing operations possible, and some of these are implemented as additional options in the rendering code. For an object such as a galaxy, which is made up of a multitude of stars, compositing modes like lighten, add, and screen can more accurately visually represent the build up of light from these stars. Also, at the galactic scale, each star is an infinitesimally small object that functions as a point light source. The light does not interact with any physical matter such as dust or gas; there is only an accumulation of the light of the galaxy's stars. Thus, an alternate compositing mode is justified both visually and astronomically. Fig. 4.11 demonstrates the effects of these compositing modes and how they are calculated.

Astronomical objects exist on an extremely large scale. Nebulae can be anywhere from a few to 10s of light years in size, and most galaxies have a diameter that is thousands of light years across. In order to allow for any possible camera move, the detail of an object must be maintained in both far away and close up situations. To help create object resolution at any distance, a level of detail control was implemented. When splats reach a user specified size in the image, they begin to split into new splats of the same color; their collective opacity is equal to the original splat. The distribution of the new splats is controlled by the convolution filter of the splat. A significant jump in the splat count can occur depending upon the camera move and the settings of the level of detail control; however, this control also limits the size that the splats can reach, which causes the rendering process to be more efficient.

5. RESULTS

For this thesis, I used a nebula, star-forming region Sharpless 2-106, and a galaxy, Messier 51, as my test cases. The initial meshes were modeled and textured in Maya, and a C++ program was used to generate an initial set of test positions. Another C++ program was used to cull these positions and generate and texture the splats. Initial look development work was done in Maya, and then translated into multiple C++ utilities. Rendering was done using STScI's splat renderer, with some modifications to accommodate the desired visualization characteristics. A camera move was designed in Maya, and the rendered images were composited, color corrected, and edited together into a short animation sequence using Adobe After Effects. In this section, I describe the results of each of the three process stages: (1) *Mesh Modeling and Texturing*, (2) *Splat Generation*, and (3) *Splat Rendering*.

5.1 Test Cases

Star-forming region Sharpless 2-106 (S106) is an H II region a few thousand light years away; from Earth, it lies in the Cygnus (The Swan) constellation of our galaxy (see Fig. 5.1). Star formation takes place throughout the nebula and a large, hot newborn star in the middle emits large quantities of UV radiation which ionizes the nearby gas causing these areas to glow blue. Cooler gas and dust exist in the pink areas of the image, forming an hourglass shape. This nebula was chosen as a test case because HII regions are the most common and prevalent nebulae in our galaxy. Also, the structure of S106 has been studied by astronomers and the initial mesh and textures were already available at STScI. The modeling and texturing stage is the most scientifically rigorous step in the process, so using previous modeling work allowed me to use the most accurate test case for a nebula as possible.

(a) Star-forming Region Sharpless 2-106 [34]

(b) Messier 51 [30]

Figure 5.1: Test Cases

Messier 51 (M51) is a spiral galaxy located approximately 25 million light years away; from Earth, it appears to lie in the Canes Venatici (The Hunting Dogs) constellation of the sky (see Fig. 5.1). It is also known as the Whirlpool Galaxy due to its tightly wound spiral arms. Multiple pink and blue pockets of star formation can be seen throughout the arms in the disk. The central bulge of M51 is made up of older and cooler stars, as is evident in that area's yellow coloration. M51 was chosen as a test case primarily because of its positioning to Earth. From our viewpoint, the galaxy is completely face on; all of the details are easily seen and undistorted so only minimal scientific and visual guesswork is needed to decipher its structure. It is also a nice representation of other spiral galaxies in the universe and one of the most visually appealing galaxies in Hubble's collection.

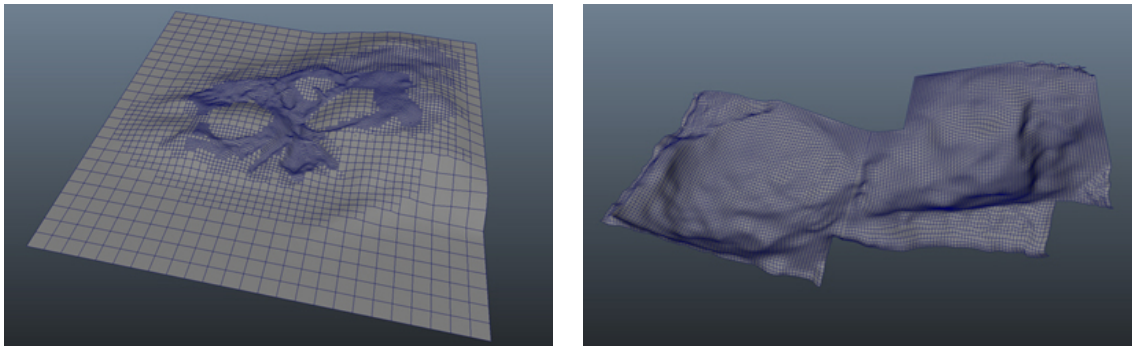
5.2 Mesh Modeling and Texturing for S106 and M51

Modeling for both S106 and M51 was done in Maya, and the texture maps for M51 were created by manipulating Hubble and Spitzer telescopic images in Adobe Photoshop and Gimp. The textures were applied in Maya so that the sculpted

structure aligned with the imagery in the maps.

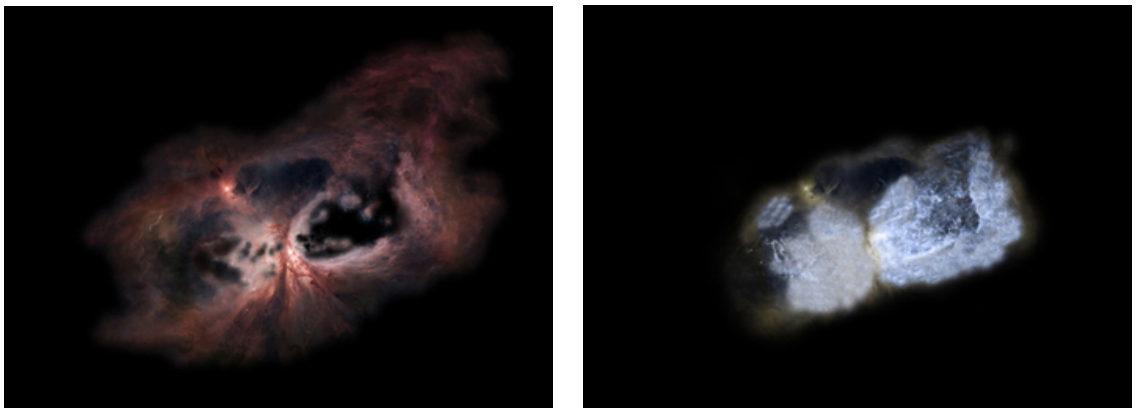
5.2.1 S106

In visible light, S106 appears to have a bi-polar hourglass shape. Noel observed S106 in both the visible and infrared wavelengths of light, proposing a 3D structure based upon the probable positions and flows of gases in the cloud [40]. By looking at the emission line spectrum of the nebula, Vallée was able to map the magnetized shell around the hot star-forming region within the nebula [54]. Based upon this work, the visualization team at STScI sculpted a mesh for S106. Due to the birth of a large, hot star at the center of the nebula, a pocket or blister structure has been formed. This configuration necessitates that two meshes be constructed, one for the top layer of cooler pink gas and another for the hot, blue areas beneath (see Fig. 5.2). There are two textures for S106, one for each of the meshes (see Fig. 5.3). These textures were also provided by STScI, using the original Hubble image as the primary source. The stars were removed from the image, but their positions and magnitudes were measured using scientific software and stored for later rendering.



(a) (b)

Figure 5.2: S106 Meshes



(a) (b)

Figure 5.3: S106 Textures

5.2.2 M51

Since M51 is a face on galaxy, its model is a simple plane with two triangular faces. Most galaxies are extremely large objects which means that generalizations in shape must be made. An edge on galaxy, NGC 891, was used as a reference to determine the relative thickness of the different components of M51 (see Fig. 5.4). The disk of this galaxy is a generally flat, pancake shape, with a slight bulge in the center. Within the image, there are also different thicknesses for the stars (bright) and the dust lanes (dark) of the disk. Using the relationships between these thicknesses and the diameter of the galaxy, an estimate can be made for the thicknesses of the stars and dust lanes in the disk of M51. These dimensions can be sculpted from the initial bounding box size during the test position generation and the addition of noise functions and thickness texture maps, which are discussed later.

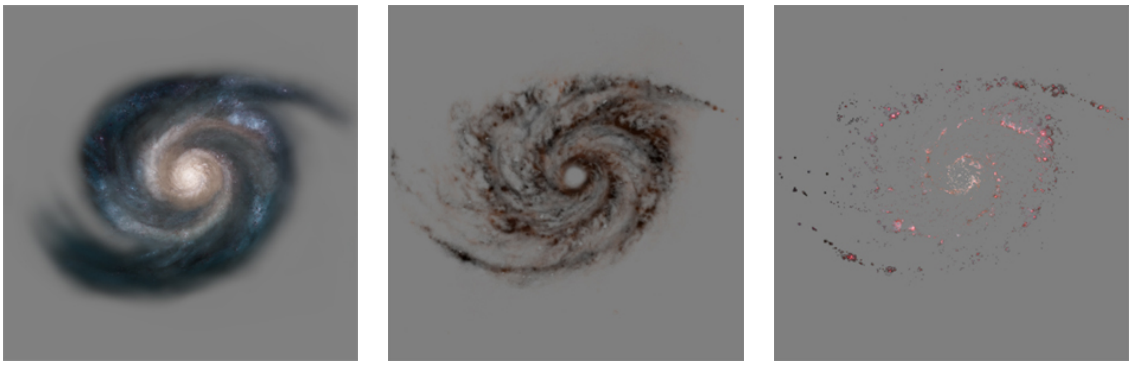
Figure 5.4: NGC 891 [10]

The texture maps for M51 are based upon the original Hubble Telescope visible light image and an infrared image from the Spitzer Telescope (see Fig. 5.5). Following the process outlined in Section IV, the first step is to align these two textures together in an image manipulation software. I then removed the bright stars, background galaxies, and the interacting galaxy in the top right corner so that M51 was isolated on a black background.

(a) (b)

Figure 5.5: M51 Source Images: (a) Visible light image [30], (b) Infrared image [35]

Based upon the visualization parameters available in the render code, I divided up the galaxy into three components: stars, dust, and nebulaic gases. The visible light image provided the information needed for both the star and gas layers; each element was carefully separated from the background and extra information was painted out. The infrared image was used to create the dust layer by isolating the red channel and using it as a mask for a brown and black color map. Alpha layers were constructed using the luminance as a starting point. Further painting was needed to bring out areas of greater detail and push back unnecessary sections of the image. The final textures are aligned, cleaned of excess information, and have alpha layers (see Fig. 5.6).



(a) (b) (c)

Figure 5.6: M51 Textures: (a) Stars, (b) Dust, and (c) Nebulae

5.3 Splat Generation for S106 and M51

To generate the splats for S106 and M51, I triangulated the meshes in Maya and exported them as OBJ files. I wrote a C++ program, generator.cpp, that takes in the OBJ file and allows the user to specify a buffer amount for the bounding box and the number of voxel subdivisions to create. This results in a text file with a set of

jittered test positions. To cull the test positions and texture the valid splats, I wrote another C++ program, culler.cpp, that finds the closest point on the mesh to each test position, checks that it is within the maximum distance parameter, and performs a color lookup to assign initial RGBA values. This program works best with simple meshes that have a small number of faces. I wrote a second script in Python, for Maya, and it executes the culling process much faster for complex meshes. Refer to Table 5.1 to see the efficiency tests performed using both programs. The Python script runs significantly faster with meshes that contain larger numbers of faces, almost three times as fast as the C++ code operating on the same mesh. I suspect that this speed difference is due to the way that Maya stores meshes, making these distance calculations more efficient. On the other hand, the C++ code runs almost 200 times faster on meshes that contain a small number of faces. Thus, in the spirit of efficiency, I used the C++ program for M51 and the Python script for S106. The final data was written into a PDA file for further look development modifications and rendering. All distance values in the following paragraphs are in Maya units.

Table 5.1: Culling Code Comparison

Test Sets	# of Faces	Python Run Times	C++ Run Times
10K	12,992	61.1s	134s
500K	12,992	2220s	6420s
500K	2	2100s	8s

5.3.1 S106

For S106, I decided to use two different sets of splats with different look parameters for each mesh. The first set consisted of mostly opaque splats that are close to the surface, with a maximum distance parameter of 0.1 units from the surface. These splats become semi-transparent following the values in the texture map. This "blanket" of mostly opaque splats is a platform upon which we can build a larger set of softer, cloud-like splats. This second set of semi-transparent splats extends farther off of the surface to a maximum distance of 1.0 units. Table 5.2 shows the initial set amounts, the final culled sets, and the run times for these programs. To run the Python culling code, I used a networked set of computers at Texas A&M University, leveraging the processing speed of approximately four CPUs for each of the twenty workstations to speed up my workflow.

Table 5.2: S106 Initial Sets, Culled Sets, and Run Times (Python Script)

S106 Splat Sets	Initial Set	Culled Set	Run Times (hrs)
Low Opaque	56M	402,166	74.9
Low Transparent	61M	3,715,018	80.9
Top Opaque	115M	778,473	152.9
Top Transparent	134M	7,812,353	178.7

5.3.2 M51

M51 is divided into three separate splat types based upon the different particle types within the disk of the galaxy. Using NGC891 as a reference, I determined that the stars extended to 1.5 units from the surface, relative to the diameter of

the galaxy. The dust and nebulae layer is slightly thinner, reaching out to about 1.2 units on either side of the mesh. These thicknesses were varied further using thickness maps during the look development stage. Initial sets were generated for all three sets, and the C++ program was used to cull them down to a more manageable and accurate size. Refer to Table 5.3 for these set amounts and run times. Due to the efficient nature of the C++ code with simple meshes, only a single computer was needed to run the code in a reasonable time.

Table 5.3: M51 Initial Sets, Culled Sets, and Run Times (C++ Program)

M51 Splat Sets	Initial Set	Culled Set	Run Times (min)
Stars	61M	24,104,450	16.3
Dust	16M	6,064,978	4.3
Nebulae	34M	2,469,731	9.1

5.4 Rendering Final Sequence

For look development and rendering, I used a combination of Maya, C++ code, and C code. Maya was a useful tool in the initial stages of look development, since it allowed for quick iterations on low resolution sets. I was able to tweak the functions in the Expression Editor until I found a look that worked best for each situation. C++ programs were used to modify the alpha values, add the radius attributes, create noise, and calculate probability density functions. The rendering code is written in C, and some alterations were made in conjunction with Dr. Frank Summers of STScI in order to provide the functionality needed for this situation.

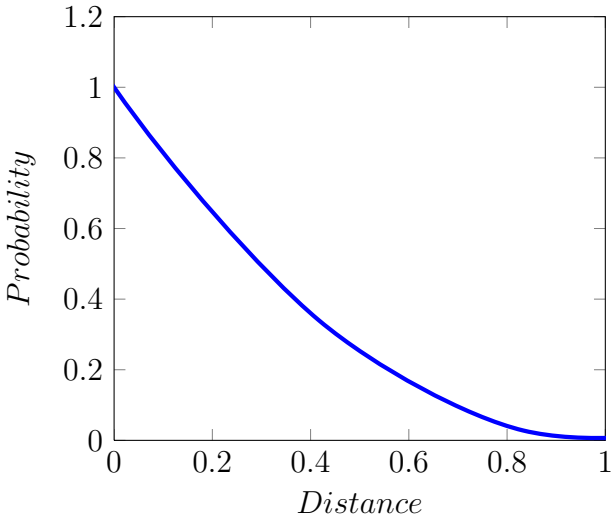


Figure 5.7: S106 Probability Density Function: $y = (x-1)^2$

5.4.1 S106

For S106, the majority of the look development choices were made to the data set prior to rendering. I used a C++ program to create a vertical falloff with a quadratic probability density function; this falloff most closely resembled the look I desired. The graph of this function is presented in Fig 5.7. The function is valid for $x \in [0, 1]$.

Following this step, I wrote another C++ code that modified the alpha in both the vertical and horizontal directions as discussed in Section IV. The primary issue that needed to be corrected was the dark edges around the holes in the top layer of S106. The majority of the splats were assigned alpha values greater than 0.95, so I was able to isolate the splats in the problem areas by applying the horizontal alpha modification to splats with alpha values lower than 0.95. This resulted in a sharper falloff around the problem areas, removing the dark edges. The horizontal alpha was scaled using Equation 5.1 below.

$$alpha = alpha^6 \tag{5.1}$$

I suspect that the need for a horizontal alpha modification may be eliminated with further refinement to the original alpha layer in the texture map. Also, the modification could be unnecessary with nebulae that are mostly opaque or only consist of a single layer of splats. Once the horizontal alpha was corrected, the vertical alpha calculations were applied. I developed a custom equation to scale the alpha based on its distance from the surface, as seen in Equation 5.2 and Fig. 5.8.

$$f(alpha) = \begin{cases} d \in [0, 0.33) & alpha = alpha * (-1.515 * d + 1.0) \\ d \in [0.33, 0.66) & alpha = alpha * (-1.212 * d + 0.9) \\ d \in [0.66, 1.0) & alpha = alpha * (-.294 * d + 0.294) \\ d >= 1.0 & alpha = 0 \end{cases} \tag{5.2}$$

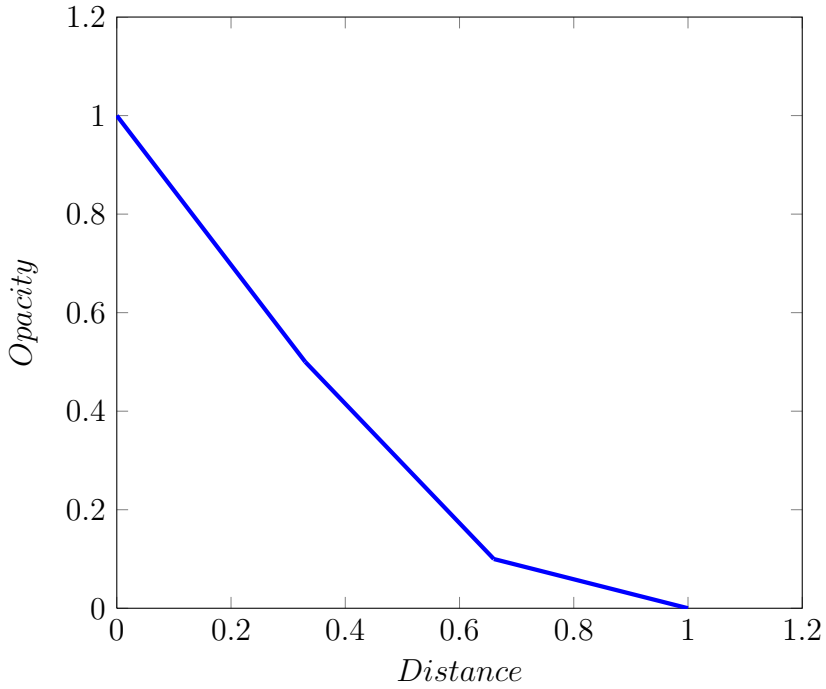


Figure 5.8: S106 Alpha Function

A radius was added to each splat as well. For the opaque sets, the splats were assigned an initial random radius between 0.1 and 0.15 units. The radius was then altered in the horizontal direction, getting larger as the alpha became more transparent. For the semi-transparent sets, the splats closest to the surface were assigned radius values in the same manner as the opaque sets. Farther away from the mesh, the radius was scaled larger. Fig. 5.9 shows these alpha and radius modifications using a side view of the S106 top splat set.

Figure 5.9: S106 Top Splats with Alpha and Radius Modifications

Rendering for S106 was performed using the computers in the lab at Texas A&M University as well as a set of computers at STScI, managed by Dr. Frank Summers. Render times varied based upon how large the splats were in the image; thus, for our sequence, the initial frames were processed quickly and later frames took progressively longer. Table 5.4 shows the final splat counts after look development and average render times for each frame.

Table 5.4: S106 Render Times

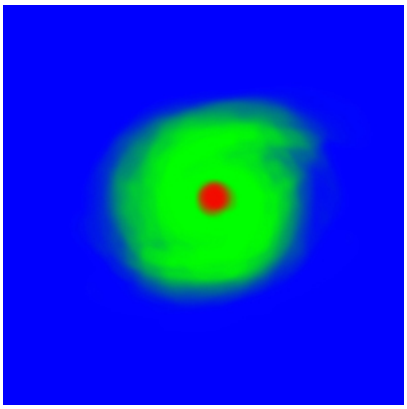
S106 Splat Sets	Splat Count	Avg. Time(sec) per Frame
Top Opaque	778,473	11.85
Top Transparent	2,600,015	30.35
Low Opaque	402,166	7.39
Low Transparent	1,295,449	14.31

5.4.2 M51

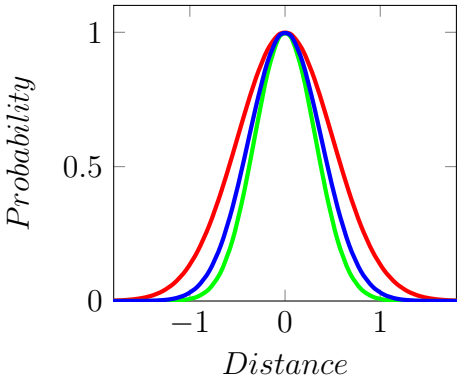
The look development process for M51 required some new code to be written to adjust the set before and during the rendering process. The previous probability density functions did not match the distribution of the edge-on galaxy reference images, so I implemented a version that uses a gaussian function. The edge-on renders still appear to be too stiff and mathematical; however, of all of the density falloffs implemented, the gaussian function presented the most similar distribution to the reference images. Additionally, I painted thickness maps that were used to interpolate between probability density functions to create gradual changes in thickness with each of the galaxy disk components. The disk of a galaxy has a generally flat shape that flares outward. In the center, the bulge of the galaxy is slightly thicker than the disk. Therefore, for the star splat set, I used a three color (RGB) map that denoted areas of thickness for the bulge, the thinner part of the disk, and the thicker outer edges. This map and the graphs of the probability density functions related to each color section are shown in Fig. 5.10.

The dust lanes of the galaxy exist within the disk, so the thickness map for the star layer was altered by removing the third color that indicated bulge thickness.

(a) M51 with Star Thickness

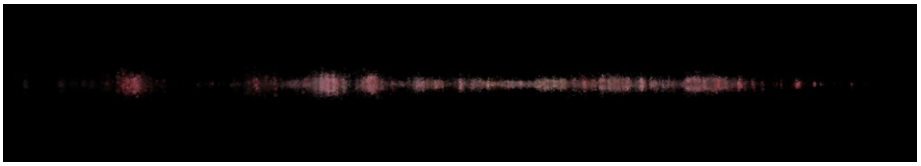


(b) Star Thickness Map

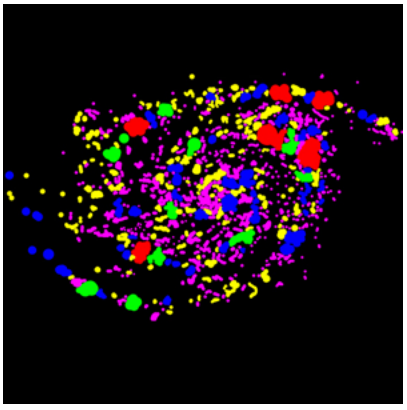


(c) Star Thickness Functions

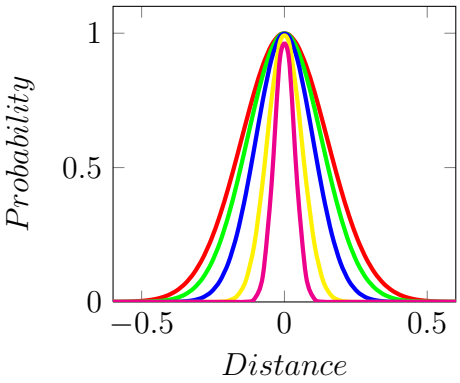
Figure 5.10: M51 Probability Density Functions for Star Thicknesses



(a) M51 with Nebulae Thickness



(b) Nebulae Thickness Map



(c) Nebulae Thickness Functions

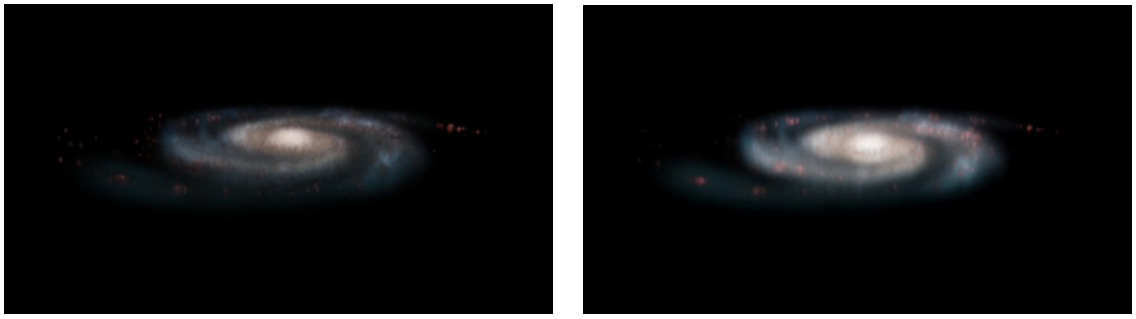
Figure 5.11: M51 Probability Density Functions for Nebulae Thicknesses

For the nebulae splats, a more detailed map was needed. The thickness of a nebula can be chosen using a simple assumption that most nebulae are about as thick as they are wide, and would not extend through the entire thickness of the galaxy. Using the original texture as a reference, I used a five-color map to denote five different thicknesses for the nebulae under the assumption that the larger nebulae would be thicker. This resulted in some nice thickness variation for the nebulae splats; however, a more accurate solution would slightly translate each nebula in the Y axis so that all of splats are not positioned along the central plane of the disk. Further research and refinement would also allow for more realistic thicknesses for the nebulae. Fig. 5.11 displays the nebula thickness map as well as the matching probability density functions.

The texture image for the dust set uses a lower resolution source, resulting in the loss of finer details. I added three dimensional fractal noise to the dust layer to try to bring some of these elements back. A larger noise function was added to cull the dust set in the Y axis to break up the constant silhouette. I also implemented a smaller noise amount as a jitter in the Y axis to the remaining splats. The noise created a slightly more naturalistic and accurate edge-on appearance for M51. The results of these culling operations can be seen in Table 5.5. The last numbers in each row are the final set counts used for rendering purposes.

Table 5.5: M51 Look Development Culls

M51 Splat Sets	Original Sets	PDF Cull	3D Noise Cull
Stars	24,104,450	7,838,588	-
Dust	6,064,978	2,128,847	1,563,518
Nebulae	2,469,731	266,101	-



(a) M51 Normal Mode (b) M51 Screen Mode

Figure 5.12: M51 Normal vs. Screen Compositing

A radius attribute was also added to all splats prior to rendering. The radii of the star splats were set to 0.1, and dust and nebula splats were assigned random values between 0.05 and 0.1. Once all look development alterations were made, I tested the splats in the existing render code. Due to the extreme change of scale, the level of detail control was needed to both maintain resolution and prevent extreme render times. This control was only applied to the star splats of M51, since they were the largest percentage of splats and affected the look the most; also, system limitations required that there be no more than 44 million splats for any given frame. A 43 million splat count was reached for frames that made the most extensive use of the level of detail control. This limitation was most likely due to a lack of memory allocation space for the static arrays implemented in the code; on a different system, or with a few changes, the code should compile and execute normally. Due to time constraints, possible solutions to this issue were not explored.

The star set also used screen mode compositing instead of the normal compositing operation. Without this change, the galaxy appeared to be dark when viewed edge-on (see Fig. 5.12). Screen compositing was a more scientifically and visually accurate method as it was the best likeness of the accumulation of light from the many stars, or point light sources, that each splat represents.

The rendering was performed on the machines at Texas A&M University and at STScI. Render times varied greatly depending upon how close the splats were to the camera, as seen in Table 5.6. Render artifacts were also an issue in frames where the splats were minified to less than a pixel in size (see Fig. 5.13). It was necessary to add animated render parameters to fix this issue. By interpolating between a constant splat falloff to a trapezoidal splat falloff, the render artifacts were removed and the look was maintained when the galaxy became larger in the frame.

Table 5.6: M51 Render Times

M51 Frames	Splat Counts w/ LOD	Avg. Render Time (min)
Farther from camera	9,668,207	~ 10
Closer to camera	10M - 44M	~ 30

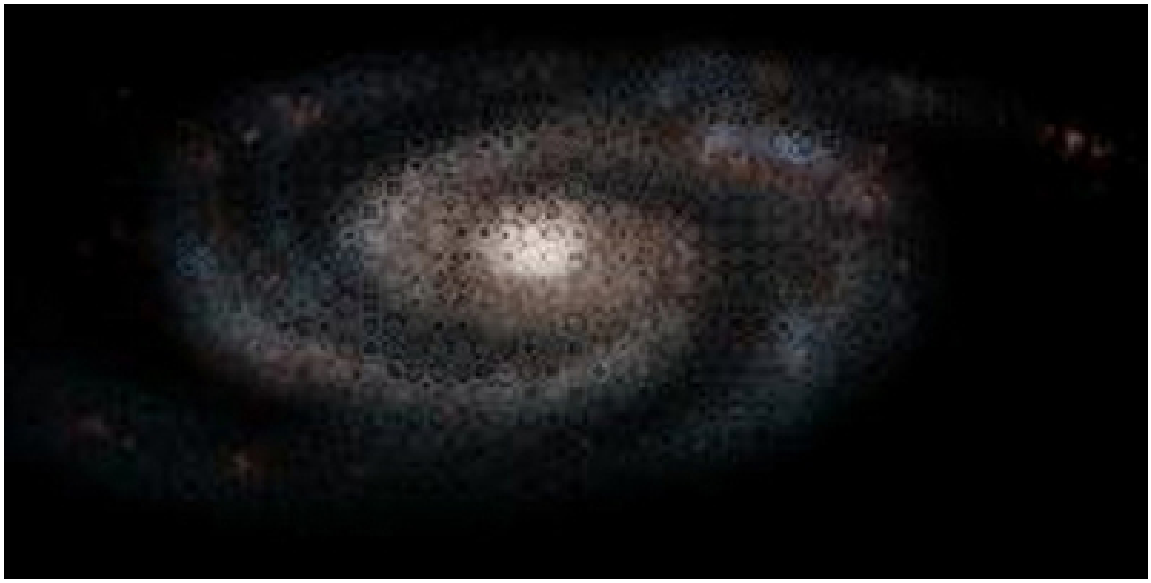
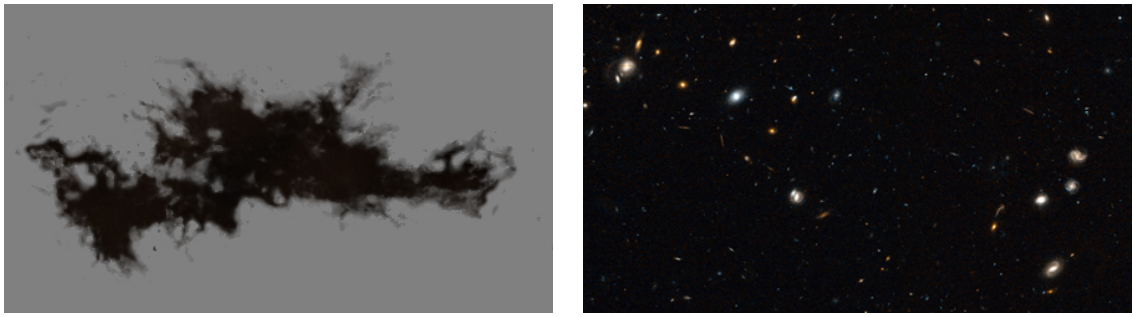


Figure 5.13: Minification Render Artifacts

5.4.3 Additional Elements

To situate S106 and M51 into a recognizable environment, some extra elements were needed (see Fig. 5.14). As an H II region, S106 sits on the outskirts of a dark dust and gas cloud. I used a reference image to create a texture map from which I could generate a set of dark brown and black dust splats for S106 to sit against. A probability density function and noise were applied to the dust to vary the overall thickness and opacity. At the beginning of the sequence, the camera is positioned outside of the galaxy. On such a large viewing scale the background area behind M51 would be filled with images other galaxies, not stars. I used a high resolution Hubble image containing thousands of galaxies to make a texture map for the environment cube surrounding the camera's path.

The presence of individual stars can make changes in three dimensional camera motion more comprehensible to the audience, and they also help with the transition between viewing M51 and viewing the much smaller S106. Two sets of stars were generated by Dr. Frank Summers. The first set of approximately 430 stars are placed around S106 using information from the original Hubble image. For the positions, the X and Y coordinates were provided by the image, and the Z coordinate was determined using each star's magnitude. The second set of approximately 800,000 stars were placed randomly along the camera path into the galaxy, so that they would be slowly revealed as the camera moved closer. The Hipparcos catalog was used as a data reference so that a full cross section of the different magnitudes of stars would be represented. Once I had the star data, I used the original S106 and M51 data sets as a mask by rendering them with an RGB value of black. The stars could then be easily composited on top of the previously rendered images of S106 and M51.



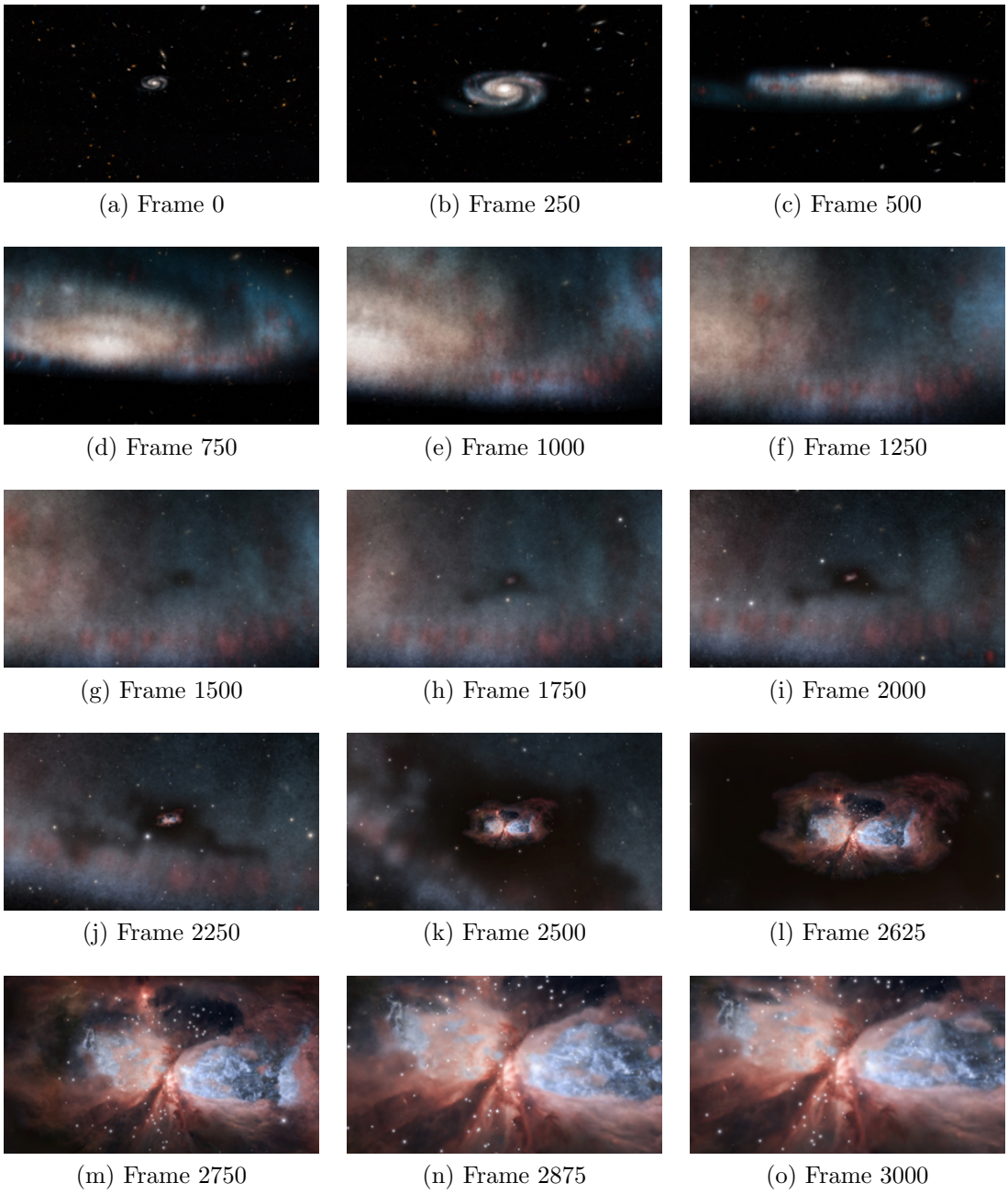
(a) (b)

Figure 5.14: Additional Elements: (a) S106 background dust, (b) Subsection of the background galaxy texture

5.5 Results

The final test of this work was a short animated sequence that begins outside M51, and moves closer, slowly approaching S106. The camera move was animated in Maya using dummy geometry to represent the galaxy and nebula positions. The elements presented above were composited in Adobe After Effects to produce a final film of approximately two minutes in length. Fig. 5.15 shows a selection of screenshots from the sequence.

Fig. 5.16 shows a comparison between the rendered version of S106 and its original reference image. A slight loss in definition was expected since the splats do not have the same resolution as the original image. Based upon discussions with astronomers at STScI, the splat version of S106 retained most of the scientific accuracy and visual appeal of the original Hubble press release image. The process was efficient in generating the splats and rendering the data sets as demonstrated by the previous data tables. The camera movement is a bit limited due to the somewhat two dimensional structure of S106. The camera has no reason to rotate around the nebula since it is positioned against a patch of dust; however, a multitude of camera options are available within those restrictions. The process could easily be extended



(a) Frame 0 (b) Frame 250 (c) Frame 500

(d) Frame 750 (e) Frame 1000 (f) Frame 1250

(g) Frame 1500 (h) Frame 1750 (i) Frame 2000

(j) Frame 2250 (k) Frame 2500 (l) Frame 2625

(m) Frame 2750 (n) Frame 2875 (o) Frame 3000

Figure 5.15: Screen Shots from Animated Sequence

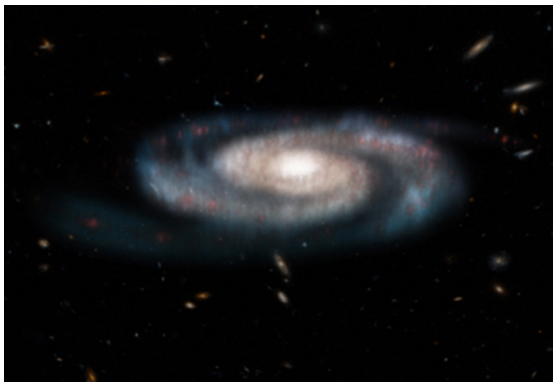
(a) Original S106 (b) Rendered S106

Figure 5.16: S106 Comparison

to new nebula visualizations, particularly those that have a dual layer structure like that of S106.

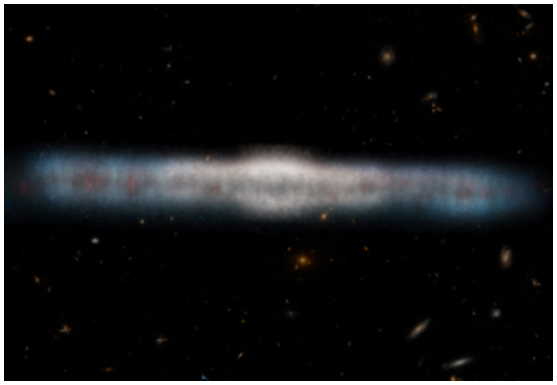
Fig. 5.17 displays a comparison between the rendered front view and edge view of M51 and its reference images. In terms of scientific accuracy, M51 succeeds reasonably well in the edge-on galaxy case. The stars and dust layers falloff in similar amounts to the reference image of NGC891, although the general shape is too stiff and inorganic. The bulge is evident, and the disk shape corresponds to previous scientific research regarding the size relationships between the bulge and disk sizes of other spiral galaxies. The rendered front view of M51 has elements of the original image, but much of the detail has been lost due to a lack of resolution and contrast. This lack of resolution is particularly evident as the camera comes closer to the galaxy. Future work in this area could improve the fidelity to the Hubble image. The splat generation and look development process was exceedingly efficient for M51, allowing large sets of splats to be created in a matter of minutes. The rendering process took significantly longer to complete than S106, indicating another possible area of for future work. The current camera flexibility is infinite, and the

(a) Original M51



(b) Rendered M51

(c) Original Edge-on Reference



(d) Rendered Edge-on M51

Figure 5.17: M51 Comparison

level of detail control could be expanded to be more effective with larger data sets. Any other mostly front facing galaxy could be visualized using this method.

6. CONCLUSION

This thesis presents a highly efficient and inexpensive way to volumetrically render nebulae and galaxies, as all of the calculations can be performed on a common workstation in a reasonable amount of time. The volumetric rendering solution presents the most opportunities for dynamic camera shots from a variety of angles and distances. The implementation of the level of detail control assures that the resolution and detail of the object will be maintained as the camera moves closer an object. The efficiency of the process and the flexibility in camera choice makes this process extendable to most other cases of nebula and galaxy visualizations. The scientific accuracy and visual appeal of the results are confirmed through discussion with astronomers at STScI, how similar the splat renders are to the original reference images, and the lack of render artifacts. Overall, it demonstrates reasonable scientific accuracy with nebula cases and takes some significant steps towards visualizing galaxy cases.

Our splat generating, texturing, and rendering process has potential as a visualization tool for astronomers; their research can be presented to a wider audience in a more cinematic style. Discoveries could be presented to the public in a more comprehensible and enlightening way. This process could also be helpful in the creation of educational planetarium shows; artists would not be limited to purely artistic interpretations or heavily scientific simulations.

There are many areas where this method can be improved. The look development process is inherently tedious in computer graphics, requiring many small tweaks and iterations to achieve the desired appearance. In our work, look development has been moved to a highly technical system that requires changes to be made within

the code. Artists and astronomers without a background in computer science and programming would find the look development and rendering steps of the process technically challenging. Integrating our code with a graphical user interface and a 3D environment such as Maya or OpenGL would make look development less time consuming and more artist friendly.

The initial test position culling process revealed some underlying efficiency issues with our C++ code. Currently, the algorithm to find the closest point on mesh is a simple brute force method, performing a standard loop of the vertices, edges, and faces of the mesh. There are redundant calculations during the edge case, as each edge is actually considered twice since its connected to two faces. A more elegant data storage solution that takes three dimensional position into consideration, like a grid or a tree, would increase the speed of our code when it is run using a mesh with numerous faces.

The minification of splats causes significant aliasing in the rendered image. Our solution of altering the splat falloff amounts appears to be sufficient for our needs. Other situations may require a more rigorous solution. Previous antialiasing solutions have been investigated, but may not be feasible as our work deals primarily with semi-transparent splats instead of opaque splats.

For galaxies, one of the major issues with the current process is the limitation to the amount of splats. The render code would not compile with an array size limit higher than 44 million units. With the addition of the level of detail control, this severely limited the resolution of rendered data set. Ideally, significantly more splats would be used for galaxies to allow the camera to move closer without a large loss in detail.

Another issue with the galaxy is the use of screen mode compositing. Although it works well for the few frames where the galaxy is viewed edge-on, the general

brightening of the splat set results in a loss of contrast and detail that post processing color correction cannot rectify. The normal compositing creates a better face on view of the galaxy and the colors more closely resemble the original image; however, the illusion of a multitude of stars is ruined by the darker splats along the edges. Unpremultiplying these darker areas would only work if the resulting bright colors were brought down again by low alpha values, as seen with S106. Its possible that a better alpha layer for M51 would facilitate this solution. Another possible fix is a merging of the two compositing modes, interpolating between normal and screen modes based upon the angle of the galaxy to the camera.

In conclusion, we have developed a process of generating, texturing, and rendering astronomical phenomena like galaxies and nebulae using volumetric splats. The process was implemented using Maya to construct and texture the surface meshes, C++ code to generate and texture splats based upon these meshes, and C code to render the data sets. (1) Any number of camera movements can be used, and the rendered images have (2) reasonable scientific accuracy and (3) visual appeal. The process is (4) extendable to different astronomical visualizations and runs (5) efficiently on a common workstation.

REFERENCES

[1] Albanesius, C. Google Launches Stunning '100,000 Stars' Chrome Experiment. *PCMag*. www.pcmag.com/article2/0,2817,2412197,00.asp, (November 2012).

[2] American Institute of Physics. Crab Nebula in X Rays. http://www.aip.org/png/html/crabneb.html, (September 1999).

[3] Astronomind: Guide to Cosmos. Space documentary (best astronomy movies). http://www.astronomind.com/fiction/space-astronomy-documentary, (December 2011).

[4] Bennett, J., Donahue, M., Schneider, N., and Voit, M. *The Essential Cosmic Perspective*, 6th ed. Boston: Addison-Wesley, 2012.

[5] Blinn, J. F. Light Reflection Functions for Simulation of Clouds and Dusty Surfaces. In *Proceedings of the 9th Annual Conference on Computer Graphics and Interactive Techniques* (1982), SIGGRAPH '82, ACM, pp. 21–29.

[6] Branagh, K. *Thor*. Paramount Pictures, 2011. Film.

[7] BUF. Thor Production Notes. http://www.buf.fr/visual_effects.php, (May 2011).

[8] Coffey, J. Types of Galaxies. http://www.universetoday.com/83839/types-of-galaxies/, (February 2011).

[9] Csuri, C., Hackathorn, R., Parent, R., Carlson, W., and Howard, M. Towards an interactive high visual complexity animation system. *SIGGRAPH Computer Graphics 13*, 2 (August 1979), 289–299.

[10] Cuillandre, J.-C., Starlight, H., and CFHT. Astronomy Picture of the Day: Interstellar Dust Bunnies of NGC 891. http://apod.nasa.gov/apod/ap020703.html, (July 2002).

[11] Eames, C., and Eames, R. *A Rough Sketch for a Proposed Film Dealing with the Powers of Ten and the Relative Size of Things in the Universe.* Commission on College Physics, 1968. Documentary Short Film.

[12] Elias, H. Perlin noise. http://freespace.virgin.net/hugo.elias/models/m_perlin.htm, (December 1998).

[13] Frank, A. 'Thor' Blends Myth And Science. *NPR.* http://www.npr.org/blogs/13.7/2011/05/11/136169971/, (May 2011).

[14] Genetti, J. Volume-rendered Galactic Animations. *Communications of the ACM 45*, 11 (November 2002), 62–66.

[15] Google Chrome. 100,000 Stars. http://workshop.chromeexperiments.com/stars/, (November 2012).

[16] Greene, N., Kass, M., and Miller, G. Hierarchical Z-Buffer Visibility. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques* (1993), SIGGRAPH '93, ACM, pp. 231 – 238.

[17] Guedes, J. Eris Simulation. https://webhome.phys.ethz.ch/~jguedes/eris.html, (November 2011).

[18] Guedes, J., Callegari, S., Madau, P., and Mayer, L. Forming Realistic Late-type Spirals in a ΛCDM Universe: The Eris Simulation. *Astrophysics Journal 742* (December 2011), 76.

[19] Levoy, M., and Whitted, T. The Use of Points as a Display Primitive. *Technical Report TR85-022* (1985).

[20] Linsen, L., Mülcer, K., and Rosenthal, P. Splat-based Ray Tracing of Point Clouds. *Journal of WSCG 15* (2007), 51–58.

[21] Magnor, M., Kindlmann, G., and Hansen, C. Constrained Inverse Volume Rendering for Planetary Nebulae. In *Proceedings of the Conference on Visualization '04* (2004), IEEE Computer Society, pp. 83–90.

[22] Magnor, M., Kindlmann, G., Hansen, C., and Duric, N. Reconstruction and Visualization of Planetary Nebulae. *IEEE Transactions on Visualization and Computer Graphics 11*, 5 (September 2005), 485–496.

[23] Nadeau, D. R., Genetti, J. D., Napear, S., Pailthorpe, B., Emmart, C., Wesselak, E., and Davidson, D. Visualizing Stars and Emission Nebulae. In *Computer Graphics Forum* (March 2001), vol. 20(1), pp. 27 – 33.

[24] NASA, ESA, Beckwith, S., and the HUDF Team. Hubble Ultra Deep Field Image Reveals Galaxies Galore. http://hubblesite.org/newscenter/archive/releases/2004/07/image/a/format/large_web/, (March 2004).

[25] NASA, ESA, Beckwith, S., and the HUDF Team. Hubble's Deepest View Ever of the Universe Unveils Earliest Galaxies. http://hubblesite.org/newscenter/archive/releases/2004/07/, (March 2004).

[26] NASA, ESA, Beckwith, S., and the HUDF Team. Spiral Galaxy in the Hubble Ultra Deep Field. http://hubblesite.org/newscenter/archive/releases/2004/07/image/f/, (March 2004).

[27] NASA, ESA, HEIC, and the Hubble Heritage Team. HubbleSite Gallery: The Cat's Eye Nebula: Dying Star Creates Fantasy-like Sculpture of Gas and Dust. http://hubblesite.org/gallery/album/pr2004027a/, (September 2004).

[28] NASA, ESA, and Hester, J. HubbleSite Gallery: A Perfect Storm of Turbulent Gases in the Omega/Swan Nebula (M17). http://hubblesite.org/gallery/album/pr2003013a/, (April 2003).

[29] NASA, ESA, Hester, J., and Loll, A. HubbleSite Gallery: A Giant Hubble Mosaic of the Crab Nebula. http://hubblesite.org/gallery/album/pr2005037a/, (December 2005).

[30] NASA, ESA, S.Beckwith, and the Hubble Heritage Team. HubbleSite Gallery: Out of This Whirl: the Whirlpool Galaxy (M51) and Companion Galaxy. http://hubblesite.org/gallery/album/entire/pr2005012a/, (April 2005).

[31] NASA, ESA, Summers, F., Besla, G., and van der Marel, R. NASA's Hubble Shows Milky Way is Destined for Head-On Collision. http://www.nasa.gov/mission_pages/hubble/science/milky-way-collide.html, (May 2012).

[32] NASA, ESA, and the Hubble Heritage Team. HubbleSite Gallery: Galaxy NGC 1427A Plunges Toward the Fornax Galaxy Cluster. http://hubblesite.org/gallery/album/pr2005009a/, (March 2005).

[33] NASA, ESA, and the Hubble Heritage Team. HubbleSite Gallery: Spiral Galaxy M74. http://hubblesite.org/gallery/album/pr2007041a/, (November 2007).

[34] NASA, ESA, and the Hubble Heritage Team. HubbleSite Gallery: Star-Forming Region S106. http://hubblesite.org/gallery/album/pr2011038a/, (December 2011).

[35] NASA, JPL-Caltech, Kennicutt, R., and DSS. Spitzer View of the Spiral Galaxy M51 ("Whirlpool Galaxy"). http://www.spitzer.caltech.edu/images/, (November 2004).

[36] NASA, NOAO, ESA, and the Hubble Heritage Team. HubbleSite Gallery: The Horsehead Nebula. http://hubblesite.org/gallery/album/pr2001012a/, (April 2001).

[37] NASA, and the Hubble Heritage Team. HubbleSite Gallery: Reflection Nebula NGC 1999. http://hubblesite.org/gallery/album/entire/pr2000010a/, (March 2000).

[38] NASA, and the Hubble Heritage Team. HubbleSite Gallery: Tightly Wound Arms of Dust Encircle Nucleus of Galaxy NGC 2787. http://hubblesite.org/gallery/album/galaxy/elliptical/pr2002007a/, (April 2002).

[39] National Geographic News. Hubble's Top Ten Discoveries. http://news.nationalgeographic.com/news/2005/04/photogalleries/hubble/, (October 2010).

[40] Noel, B., Joblin, C., Maillard, J. P., and Paumard, T. New Results on the Massive Star-forming Region S106 by BEAR Spectro-imagery. *Astronomy and Astrophysics 436*, 2 (2005), 569–584.

[41] Owen, G. S. Visible Surface Determination: The Painter's Algorithm. http://www.siggraph.org/education/materials/HyperGraph/scanline/visibility/painter.htm, (May 1998).

[42] Reeves, W. T. Particle Systems- A Technique for Modeling a Class of Fuzzy Objects. *ACM Transactions on Graphics 2*, 2 (April 1983), 91–108.

[43] Riley, K., Ebert, D. S., Kraus, M., Tessendorf, J., and Hansen, C. Efficient rendering of atmospheric phenomena. In *Proceedings of the Fifteenth Eurographics conference on Rendering Techniques* (2004), Eurographics Association, pp. 375–386.

[44] Rusinkiewicz, S., and Levoy, M. QSplat: A Multiresolution Point Rendering System for Large Meshes. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (2000), ACM Press/Addison-Wesley Publishing Co., pp. 343–352.

[45] Science in School. More Than Meets the Eye: The Electromagnetic Spectrum. http://www.scienceinschool.org/print/2412, (August 2011).

[46] Science Visualization Studio. The James Webb Space Telescope: Science Videos. http://www.jwst.nasa.gov/videos_science.html, (November 2010).

[47] Sonnenfeld, B. *Men in Black*. Columbia, 1997. Film.

[48] Soper, D. E. Structure of the Galaxy. http://zebu.uoregon.edu/~soper/MilkyWay/structure.html, (April 1996).

[49] Stephens, T. Astrophysicists report first simulation to create a Milky Way-like galaxy. *University of California Santa Cruz NewsCenter* (August 2011).

[50] Summers, F., Bacon, G., Frattare, L., and Levay, Z. Experience Hubble's Universe in 3-D. http://hubblesite.org/newscenter/archive/releases/2010/12/image/a/, (March 2010).

[51] Summers, F., Bacon, G., Frattare, L., and Levay, Z. Science, Data, and Art in the Imax Film *Hubble 3D*. In *American Astronomical Society Meeting Abstracts #215* (January 2010), vol. 42 of *Bulletin of the American Astronomical Society*, p. 206.01.

[52] Swan II, J. E., Mueller, K., Möller, T., Shareef, N., Crawfis, R., and Yagel, R. An Anti-Aliasing Technique for Splatting. In *Proceedings of the 8th Conference on Visualization '97* (1997), IEEE Computer Society Press, pp. 197 – ff.

[53] TV Tropes. Astronomical Zoom. http://tvtropes.org/pmwiki/pmwiki.php/Main/AstronomicZoom, (1999).

[54] Vallée, J., and Fiege, J. D. A Cool Magnetized Shell Wrapped Around the Hot H II Region S106: Geometry, Kinematics, Magnetic Vectors, and Pressure Balance. *The Astrophysical Journal 627* (July 2005), 263–276.

[55] Wang, N. Realistic and Fast Cloud Rendering. In *Computer Games. SIGGRAPH 2003* (2003), pp. 21–20.

[56] Wen, Z., and O'Dell, C. R. A Three-Dimensional Model of the Orion Nebula. *The Astrophysical Journal 438* (January 1995), 784 – 793.

[57] Westover, L. Interactive Volume Rendering. In *Proceedings Volume Visualization Workshop* (1989), University of North Carolina at Chapel Hill.

[58] Wikipedia. Galaxy. http://en.wikipedia.org/wiki/Galaxy, (September 2001).

[59] Wikipedia. Nebula. http://en.wikipedia.org/wiki/Nebula, (September 2001).

[60] Wikipedia. Hipparcos Catalog. http://en.wikipedia.org/wiki/Hipparcos_catalog, (April 2003).

[61] Wikipedia. Orion Nebula. http://en.wikipedia.org/wiki/Orion_Nebula, (January 2003).

[62] Zemeckis, R. *Contact*. Warner Brothers, 1997. Film.

APPENDIX A

ALGORITHMS AND FILE FORMATS

Figure A.1 Format of OBJ File

```
{List of vertices in the format: v x y z}
v 0.1 0.2 0.2
v 0.3 0.8 0.2
v ...
...

{List of vertex texture coordinates in the format: vt u v}
vt 0.5 0.4
vt 0.8 0.2
vt ...
...

{List of vertex normals in the format: vn x y z}
vn 0.2 0.4 0.4
vn 0.1 0.5 0.4
vn ...
...

{List of faces in the format: f v/vt/vn}
f 6/4/1 3/5/3 7/6/5
f 7/6/5 8/7/2 12/9/4
f ...
```

Figure A.2 Format of PDA File

{Header Information (example with typical attributes for this project)}

ATTRIBUTES
position rgbPP opacityPP DistanceFromSurfaceMeshPP whichMeshPP
TYPES
V V R R R
NUMBER_OF_PARTICLES: 100
BEGIN DATA

{Particle Attribute Data: X Y Z R G B A D W}

1.0 0.0 1.0 0.5 0.5 0.5 1.0 0.9 1
1.5 0.0 1.5 0.2 0.8 0.8 0.8 0.6 1
...

Figure A.3 Algorithm for checking test positions and generating textured splats

```
test_positions; {list of test positions}
obj; {3D mesh}
maxdist; {maximum distance bound}
point_on _mesh; {closest point on mesh}
splats; {list of viable splats}

for test_point in test_positions do

    d = MAX_FLOAT

    {Find closest point among vertices }
    for vert in vertices of obj do
        tempD = distance between vert and test_point
        if tempD < d then
            d = tempD
            point_on_mesh = vert
        end if
    end for

    ...

    {Find closest point among edges }
    for edge in edges of obj do
        p = closest point on line to test_point
        if p in edge then
            tempD = distance between p and test_point
            if tempD < d then
                d = tempD
                point_on_mesh = p
            end if
        end if
    end for
    ...
```

```
    {Find closest point among faces }
    for face in faces of obj do
        p = closest point on face plane to test_point
        if p in face then
            tempD = distance between p and test_point
            if tempD < d then
                d = tempD
                point_on_mesh = p
            end if
        end if
    end for

    {Check distance against maximum distance bound }
    if d <maxdist then
        RGBA = color at point_on_mesh
        add test_point, RGBA, and d to splats
    end if

end for
```

Figure A.4 Vertical Opacity Modification Algorithm

```
target_alpha; {alpha from texture image}
distance; {distance of splat from center}
density; {thickness of splats in Y axis}
ymax; {maximum bound of splats in Y axis}
ymin; {minimum bound of splats in Y axis}

step = (ymax - ymin) / density;
total_alpha; {sum of alphas at each density step}

{Integrate the alpha through the density}
for i in density do

    d = ymin + (i * step);
    pdf = pdf_function(d); {Find density at this distance.}
    total_alpha = total_alpha + pdf * alpha_function(d, total_alpha);

end for

{Find scale_factor.}
if total_alpha != 0 then
    scale_factor = target_alpha / total_alpha;
end if

{Find final alpha and scale it with scale_factor.}
new_alpha = scale_factor * alpha_function(distance, target_alpha);
```