# MULTI-CAMERA: INTERACTIVE RENDERING OF ABSTRACT DIGITAL IMAGES

A Thesis

by

JEFFREY STATLER SMITH

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

December 2003

Major Subject: Visualization Sciences

# MULTI-CAMERA: INTERACTIVE RENDERING OF ABSTRACT DIGITAL IMAGES

A Thesis

by

JEFFREY STATLER SMITH

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

MASTER OF SCIENCE

Approved as to style and content by:

| | |
|---|---|
| Ergun Akleman (Chair of Committee) | Richard Davison (Member) |
| John Keyser (Member) | Phillip Tabb (Head of Department) |

December 2003

Major Subject: Visualization Sciences

# ABSTRACT

Multi-Camera: Interactive Rendering of
Abstract Digital Images. (December 2003)

Jeffrey Statler Smith, B.E.D., Texas A&M University
Chair of Advisory Committee: Dr. Ergun Akleman

The purpose of this thesis is the development of an interactive computer-generated rendering system that provides artists with the ability to create abstract paintings simply and intuitively. This system allows the user to distort a computer-generated environment using image manipulation techniques that are derived from fundamentals of expressionistic art. The primary method by which these images will be abstracted stems from the idea of several small images assembled into a collage that represents multiple viewing points rendered simultaneously. This idea has its roots in the multiple-perspective and collage techniques used by many cubist and futurist artists of the early twentieth century.

To my family and friends

## ACKNOWLEDGMENTS

I would like to express my sincere thanks to my thesis committee chair, Dr. Ergun Akleman, for sharing his expertise and for encouraging me to explore new creative ideas. I would also like to thank my committee members, Prof. Dick Davison and Dr. John Keyser, for providing their help and encouragement throughout this process. I would also like to thank Dr. Donald House, Prof. Robert Schiffhauer, and Michael Ringham for their valuable support and instruction.

My thanks go to Manfred Mohr, Scott Snibbe, Stephen Parker, Robert Schiffhauer, and Dick Davison for allowing me to display their artwork in this thesis, and to Vinod Srinivasan, Michael Mistrot, and Michael Stanley for allowing me to adapt portions of their code for use in my thesis program.

My gratitude goes also to all of the faculty and staff of the Viz Lab, who have provided answers to my questions and have enhanced my learning experience. Thanks also go to all the friends that I have made during my time in the Viz Lab, for sharing their ideas about art and for their all of their help and enthusiasm. Thanks especially to Lori Green, for her inspiration and for providing a great deal of encouragement and support. Finally, I would like to thank my parents, sister, and extended family for their love and guidance through the years.

# TABLE OF CONTENTS

# LIST OF FIGURES

FIGURE Page

FIGURE Page

FIGURE Page

# CHAPTER I

# INTRODUCTION

## I.1. Motivation

As an artistic tool, the use of the computer is unparalleled in its versatility. The computer allows ideas and methods related to a wide range of media to converge and be redefined through the adaptability and robustness of computer software. Hence, computer art has become an increasingly popular means of creative expression and experimentation.

As computer graphics technology advances toward an ability to reproduce reality through lifelike character animation and digital renderings indistinguishable from photographs, so grows the likelihood of a divergence from this goal in the spirit of the many non-representational art movements of the late nineteenth and early twentieth centuries. Just as technological advances have allowed computers to be used to create strikingly vivid images, so too should they allow computers to be used for the development of abstract images of high aesthetic value.

In computer graphics, although there has been an increasing interest in artistic (or non-photorealistic) approaches, only a few truly abstract rendering approaches have been developed to create abstract digital paintings [1, 20, 23]. The goal of this thesis is to use 3D computer graphics technology to create a new framework for producing art in the 3D realm, one that emphasizes the creation of images that are abstract in nature.

---

The journal model is *IEEE Transactions on Visualization and Computer Graphics*.

## I.2. Introduction

The departure from representational methods in traditional painting provides a distinction between abstraction and realism that can be carried over into computer graphics. Realistic paintings, like photography, have a definite subject, and possess certain degrees of adherence to strict conventions of representing true perspective and natural light as a means of reproducing human vision. Abstract paintings through some means redefine the form and/or the content of the work of art, and have a distinct non-photorealistic appearance. Many styles of abstract paintings were invented in the early twentieth century, including Cubism, Futurism, Surrealism, Constructivism, and Abstract Expressionism [18],

The distinction between abstraction and realism in traditional painting is analogous to computer graphics artwork and the difference between photo realistic and non-photo realistic renderings. The aim of photo-realistic rendering is to produce a high- quality image with realistic lighting of detailed 3D models, that when rendered becomes a convincing substitute for a photograph captured on film. Non-photo realistic rendering is concerned with modifying existing digital images or rendering methods to the extent that renderings appear to be rendered through a medium other than that of their origin. An additional category of computer graphics artwork is defined here as "abstract" rendering, and is described as methods that modify existing computer graphics rendering methods to produce images that are assuredly non-photo realistic, but are also non-objective.

Within the category of abstract rendering, many techniques have been explored, and can generally be sub-categorized into two groups. First among these is the group that represents the methods whose purpose is to create abstract imagery, which may be inspired by a particular artistic style, but are primarily synthesized through the use of established computer graphics techniques such as physically based modeling and animation or artificial intelligence [4, 20, 23]. The second category is a group of cubism-related techniques, which attempt to emulate the multiple-view and collage aspects of cubism to develop algorithms for creating abstract renderings of 3D scenes [7, 1].

The method presented in this study resembles most closely the cubism-related approaches [7, 1, 24]. In the cubism-related approaches, a scene can be thought of as a three-dimensional space which contains a number of objects that comprise the subject of the work. A camera or eye point provides a view, or window into the scene.

When one camera records a view of a given scene, the resulting image is representative of a single moment in time captured from a single point in space, as in most photographs and 3D renderings. In this kind of image, an artistic, painterly feel is absent due to the static, photorealistic nature of the work.

One way in which interesting imagery is created involves increasing the area over which the viewing mechanism exists. A number of cameras are spread out over this area, with the resulting images from each camera combined afterwards into a single image. In addition, the cameras, instead of each being aimed at the exact same point in the scene, can be aimed at unique points of view in the scene.

By creating multiple cameras and viewpoints and slightly altering the orientation of each one, a collage of the resulting images will appear to contain movement. The result of this movement is the addition of a time element to the work, and an increased aesthetic value of the image. Alternately, movement in the individual views can be represented by an advance through time, if the medium being used is a video sequence.

Each of the cubism-related methods is capable of creating rich, interesting compositions by distorting and warping the 3D camera structure. However, interactivity in these methods is absent, resulting in less direct control over the output image.

Before a user of one of these systems can receive feedback on the composition of the image, a full scene must be rendered. This usually takes many seconds, if not minutes, to complete. Generally, these systems have control only over the camera structure in a scene, with the modeling and rendering left for another piece of software to handle. One may control the nature of the subject matter, and also the nature of the viewing apparatus, but never both within the same interface.

One advantage of the approach presented here is that the OpenGL API [16] is used to create multiple viewports and to render objects in real time. This system features interactive multiple-camera control, which treats arrays of camera points as modifiable shapes, with each camera represented by a separate viewport in the rendering interface. The user can control the positions of every camera at once, while receiving real-time updates of the output image. Moreover, the user is able to distort the 2D image space of the output by interactively changing the 2D coordinates of the many viewports. An additional advantage lies in the user's ability to experiment with color, lighting and rendering operations in conjunction with camera and viewing parameters.

In this process, photorealism and instant capture systems are used only as an initial step in the artistic process. The artist controls distortions, abstractions, color effects, media, and decisions regarding composition. Rather than starting from scratch, the artist can use photorealistic rendering and correct perspective methods to create an initial motif. From there the artist can apply abstract rendering techniques to create disjunctions in the image, while also making decisions regarding color and lighting. Finally, the image may be reworked in a 2D paint application or by hand using traditional media.

# CHAPTER II

# RELATED WORK

The work presented in this thesis draws influence from many areas of traditional and computer artwork. The non-representational nature of the work is based on ideas and methods first explored by late-nineteenth and early-twentieth century artists, as well as in early computer art. The work presented here is also related in its non-photorealistic and non-objective nature to many studies in various computer graphics and digital image rendering methods.

## II.1. Traditional Artwork

In the nineteenth century, many renaissance conventions of picture making began to break down in art [18]. Methods of representing reality as defined by a similarity to human visual perception gave way to methods of abstraction that sought to represent knowledge more than appearance. Soon after the Impressionist painters experimented with the technique of broken color to simulate the way that the eye mixes elements of light, the widely used conventions of linear perspective and natural lighting were discarded in favor of a more expressionistic style. The idea that a painting should be nothing more than a copy of what the eye sees was no longer valid.

### *II.1.1. Cezanne*

The artist Paul Cezanne is recognized as one of the first modern painters to break from the use of traditional perspective. Many of Cezanne's paintings are characterized by odd distortions of scale and perspective, including compression of depth, discontinuous line use, and a tendency for objects to appear tilted upward toward the viewer [21]. These distortions can be attributed to Cezanne's method of recording different angles of view as his focus shifted throughout the course of the painting, and to his lack of reliance on established perspective conventions [21].

Cezanne's use of color differs from that of the impressionists, who attempted to re-create the phenomenon of observing light through a mixture of color [18]. Rather than using many small spots of color that are meant to blend in the viewer's eye and thus reproduce light effects, Cezanne developed color relationships by employing a grid-like structure of larger patches of color [18]. An example of a work inspired by Cezanne is shown in Figure 1.

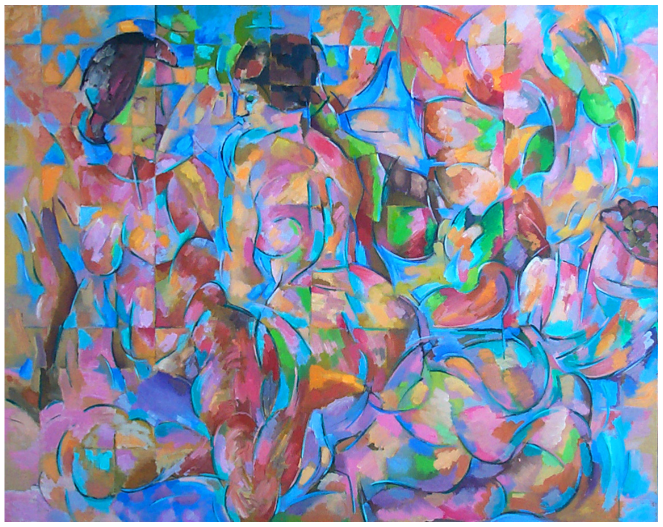


Fig. 1. Bathers I by Robert Schiffhauer. Image used with permission of Robert Schiffhauer.

### *II.1.2. Cubism*

A significant departure from traditional realistic methods of painting came with the Cubist painters [5]. The paintings of this movement were characterized mainly by a new way of handling space: a volumetric structure in which depth was flattened and objects and their surrounding environment were often simplified into many facets (see example in Figure 2). Objects were often depicted from many sides at once. The idea of multiple, simultaneous views allowed a reality of the object to be shown that was more clear and complete than by means of a single perspective. Time and memory were now added to the overall experience of a subject through painting. Later, non-painted objects were added to the surface of the canvas in an effort to recognize the medium of the work. Many of the collaged elements were related in some way to the subject, and were meant to add to the overall experience of the painting. The work of art was no longer defined as representative of two dimensions or three dimensions. With pieces jutting out of the surface of the canvas, the spatial structure of the painting was more ambiguous, yet the experience of the motif was communicated with more clarity.

### *II.1.3. Other Movements*

The realization that the motif in painting need not be static and realistic, but could be an imaginative arrangement of free-form elements, led to new ways of thinking about art [18]. Once this new way of describing reality became widespread, additional art movements were developed as extensions of cubist concepts. A group of Italian painters, known as Futurists, developed a style that was based on the multiple views of Cubism, but incorporated also the idea of movement through time and space into multiple-view compositions. In a writing entitled Manifesto of the Techincs of Futurist Painting, the Futurist painter Umberto Boccioni proclaimed that ”motion and light destroy the material of bodies” and that static subject matter should be done away with in favor of expression of the ”modern vortex of life”-steel, speed, and pride[18]. A work of the painter Marcel Duchamp, entitled Nude Descending a Staircase, is best

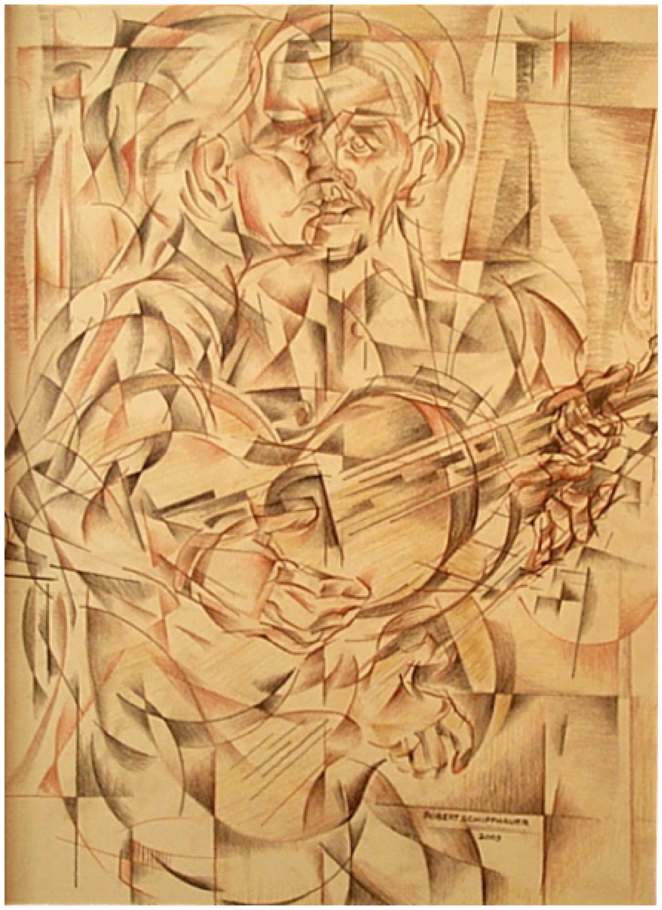


Fig. 2. Cubist Self-Portrait by Robert Schiffhauer. Image used with permission of Robert Schiffhauer.

known for the futurist elements the painting includes. In the painting, movement is represented by multiple views, each representing a successive motion of a figure walking down a staircase [5].

As the stylistic conventions of painting were redefined, so were the conventional motifs of artwork. Cubists and Futurists had invented new ways of displaying static or dynamic scenes, but their paintings remained tied to the subject in those scenes. The Bauhaus artist Wassily Kandinsky, in his writing, Concerning the Spiritual in Art, spoke of the idea that works of art should be related to music, and that form and color should be used to express inner emotions, and not appear real or represent a real subject [18].

With new challenges in expression and abstraction, art reached new audiences and began to incorporate scientific ideas. Constructivism was a bridging of art and science through the movement's founders, Antoine Pevsner, a painter, and Naum Gabo, trained in the medical field. Their work included many sculptures made of wood, metal and plastic, which were non-objective and abstract in form. The goal of the constructivists was a "synthesis of the plastic arts", the discovery of newer and broader art forms that were born out the merging of arts and science [18].

#### *II.1.4. Hockney*

One principle of Cubism that gave paintings of that genre a distinct look was the idea of combining multiple fragments of an image, each taken from a different perspective, into one final image. One recent artist, David Hockney, translated this idea into a medium other than painting. In an effort to represent reality in a non-static form, Hockney produced many collages composed of Polaroid and 35mm photographic prints [13]. In Hockney's collages, any attempt at viewing the objects normally is disrupted by the grid structure, and each view must be investigated separately in order to gain a clear understanding of the subject. The fragmented composition and multiple perspectives of Hockney's collages are reminiscent of a Cubist style, and produced a painterly feel in the work, instead of photographic realism.

## II.2. Computer Graphics Work

In the field of computer graphics, many early attempts at creating artwork with a computer were abstract, due to the limited capability of early graphical displays [10]. The first graphical images made with a computer were electronic abstractions of light beams on the cathode ray tube of an oscilloscope, created by mathematician Ben F. Laposky in 1950. In the constructivist tradition, these images were technical in nature, yet aesthetic and non-objective in appearance. Another mathematician, A. Michael Noll, used a computer to emulate works of art from the great masters, including those of the cubist and modern movements [10]. Noll used a type of programmed randomness to create his version of Composition with Lines, after Piet Mondrian. The German artist, Manfred Mohr, along with a group called Art et Information, sought to explore uses for the computer as an artistic medium [10]. Mohr produced a number of images by using a plotter and a program that generatedrandomized arrangements of cubic forms (see Figure 3).

With the advent of more sophisticated software programs and graphical displays, the efforts of computer scientists turned more toward the creation of photorealistic images. Using the technique of raytracing, for example, renderings of 3D models achieved a high degree of realism through the ability to render specular highlights, shadows, reflections and refractions [10].

As the frontier of simulating reality in computer graphics became within reach,

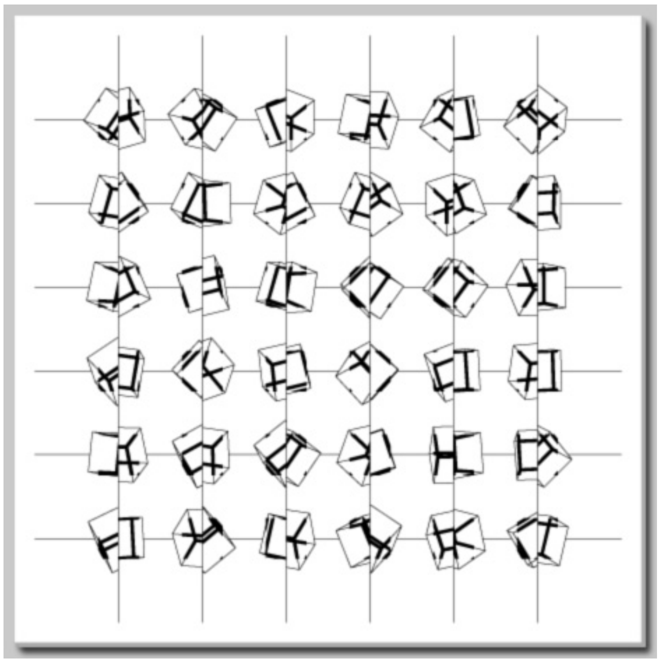


Fig. 3. P-197-K, acrylic on canvas, 136 cm x 136 cm, 1977, Collection Daimler-Chrysler, Germany. Copyright 1977 Manfred Mohr. Image used with permission of Manfred Mohr [15].

many efforts were shifted to the production of non-photorealistic images. Many of these techniques produce the effect of traditional media in digital images, without relying on physically accurate, photo-realistic rendering methods [9].

Paul Haeberli of Silicon Graphics Computer Systems developed one of the first systems of digital image abstraction [11]. Using photographs or digital renderings as a point of departure, Haeberli's system allowed the user to specify brush strokes by clicking and dragging the mouse over the original image. Each click of the mouse over the input image sampled colors to use for each brush stroke in the final rendering. Each brush stroke was saved in the output image, allowing additional operations to be performed on the brushstrokes in order to create painterly or abstract effects.

Other examples of non-photorealistic techniques include programs that can create the look of watercolor and oil painting in digital images or 3D renderings [6, 14]. While non-photorealistic rendering techniques have grown in number and complexity, these tools are mainly used to render pre-defined content in an artistic style, and are not aimed at creating abstract content in digital paintings.

### *II.2.1. Abstract Rendering Applications*

In 1991, Karl Sims created a system of artificial evolution to produce wildly abstract digital images [20]. Sims' system used lisp expressions as genotypes that were able to evolve into increasingly complex images as a result of user selection and programmed mutation and mating. Each generation of the mutation algorithm produced a unique and progressively intricate design.

Scott Snibbe and Golan Levin, using a system of two-dimensional interactive dynamic abstraction, performed additional experiments in abstract computer graphics [23]. Inspired by the abstract art and writing of Wassily Kandinsky and animation by

Oscar Fischinger, which attempted to unify the senses and relate form and color to music, these programs incorporated human motion, recorded through a mouse, into abstract two- dimensional animation loops consisting of imagery created by dynamic line drawing, Voronoi diagrams, and particle systems (Figure 4).

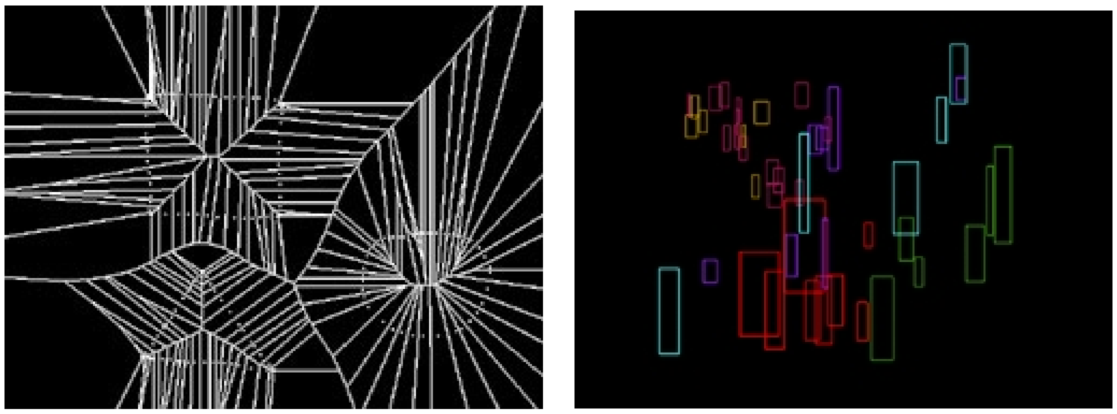


Fig. 4. Bubble Harp and Motion Phone renderings. Copyright 1991-98 Scott Snibbe. Images used with permission of Scott Snibbe [22].

### *II.2.2. Multiple-View Applications*

A number of applications have been developed that use ideas related to cubism, most notably multiple perspectives and simultaneous views, to perform various tasks. Many of these applications use methods similar to those presented in this thesis, though the work serves a different purpose.

Wood, Finkelstein, Hughes, Thayer, and Salesin used the idea of multiple perspectives to develop a system for creating backdrops for cel animations to mimic camera movement [25]. Panoramic images composed of multiple viewpoints from a 3D scene can be created using their system, which provides an alternative to hand creation of backdrop paintings as was done in traditional hand-drawn cel animation.

Rendering from multiple viewpoints has applications in many computer graphics areas of interest. Holographic and stereo image displays make use of multiple viewpoints of a static image, so that the effect of viewing a three-dimensional image can be created [12]. In this process, a two-dimensional surface displays information from different viewpoints depending on the direction from which the surface is viewed (examples: lenticular displays, baseball cards). Animations that change perspective or show an object moving can also benefit from multiple perspective rendering [12]. If a set of images of the object from multiple viewpoints is rendered prior to the animation, the different viewpoints can represent differences between the positions of the camera and the object, and frames of the final animation can be created by interpolation of the pre-rendered views.

In Michael Halle's multiple-viewpoint rendering technique [12], a set of pre-rendered images from multiple perspectives is compiled, and from these images corresponding rows of pixels are compiled into reference images. One reference image for each row of pixels of the original image size is made, and from the pixel information in this set of reference images, a final image can be produced that represents a view from a range of possible perspectives. The purpose of this process is to optimize rendering of animations with changing views.

Rademacher and Bishop in 1998 developed a system of collecting image data for image-based rendering, an emerging field of research in computer graphics [17]. With image-based rendering, realistic images can be produced by sampling color and lighting information recorded from a real-world scene and then applying that information to different viewing parameters than those from which the real-world scene was sampled.

In their system, images are produced by sampling information in a 3D scene from multiple vantage points, allowing a greater amount of information about an object to be sampled than would be possible from a single viewpoint. All of the information recorded by cameras animated along a path around an object is compiled into one data set, known as the multiple-center-of-projection image. The positions of a set of cameras can be interpolated, forming a continuous surface. By interpolating lighting and color information from each camera's position, a smooth and continuous image is created which becomes a data set from which to sample information for creating a new view of the object from any point in the scene.

In a study by Alan Z. Chen, artistic digital rendering styles were applied to computer animations sequences to determine the viewer's perception of speed in the animation [4]. The rendering styles were inspired by cubism and futurism, and included such features as the simultaneous drawing of various positions of a runner, as well as a rendering style that reduced the human form in motion to overlapping geometric forms.

### *II.2.3. Cubism-Related Techniques*

The Cubist principle of creating images from multiple points of view rendered simultaneously has also been used for creating new methods of visual storytelling. Andrew Glassner devised a "Cubist" camera system [7] through which stories can be told from various viewpoints as reflected in multiple-viewpoint images. Glassner implements the free-form camera system using the 3D modeling and rendering package, 3D Studio Max. The system uses raytracing to render images on a per-pixel basis, taking one unique sample viewpoint for each pixel in the final image. Viewing vectors are produced by sampling points from the surfaces of two NURBS planes, one designated as the "eye" plane and another as the "lens" plane. By distorting the shape of the eye and lens planes, multiple-view renderings can be produced. This method is limited, however, by the lack of a simple way to create the appropriate camera coordinates for the Cubist renderings. The connection between the manipulation of the eye and lens planes and the composition of each rendering is not seen until after the scene is rendered, which may take considerable time with raytracing.

Ergun Akleman and Scott Meadows in 2000 developed an abstract rendering system, known as camera painting [1], that used the color information from digital images to distort 3D scenes rendered with raytracing. The system uses the r,g,b color components of a digital image to replace the x,y,z coordinate information of the camera point for each pixel in a raytraced image. The output produced is an abstraction of a normal raytraced scene, controlled by an input image of the user's choice. In this way, control over the final image is based on the user's production of the input image by means of painting or photography, or selection of other imagery. Although interesting looking images can be created with relative ease with camera painting (see Figure 5), the method does not allow interactivity.

The digital artist Camille Utterback in 2000 devised an interactive installation that explored the idea of cubism as it applied to video sequences [24]. Through his program, frames from a video sequence were each divided into slices, representing vertical regions of equal size in each frame. These slices were projected onto a screen, forming a picture that would be distorted based on motion tracking equipment that recorded the position of visitors to the installation. Based on the visitor's position, different frames from the sequence would be shown in each slice region, producing a video collage effect. This installation is capable of producing intriguing interactions between movement, images, and time, and could be easily applied to an interactive software program. Generally the output images of this system are largely based on the original sequence, which limits the user's control over the image.

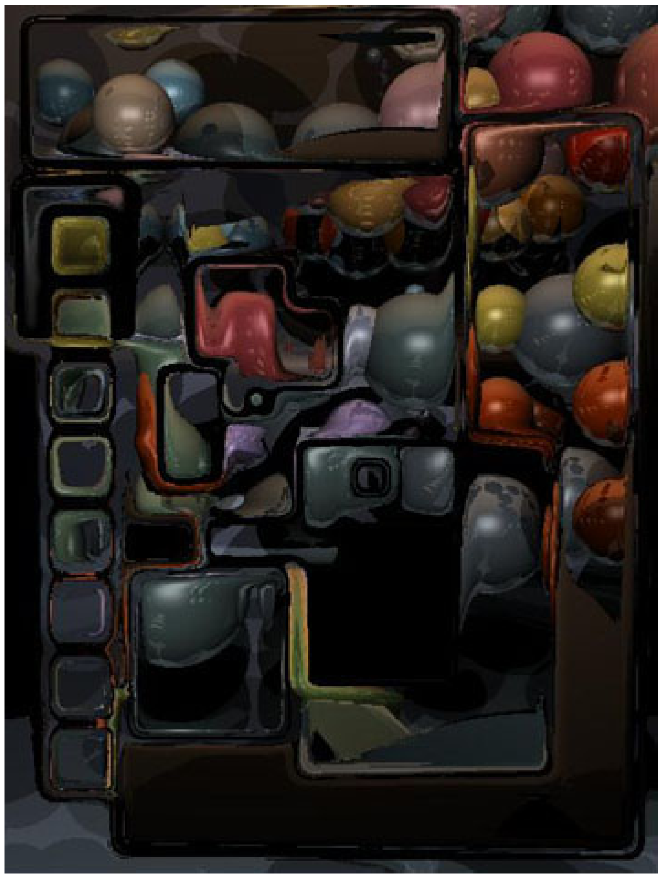


Fig. 5. Camera Painting by Akleman and Meadows [2].

# CHAPTER III

# METHODOLOGY

## III.1. Scene Methods

The method presented here for creating abstract imagery is based on 3D modeling and rendering methods. A scene in the MultiCam system is a 3D coordinate space containing a 3D model, a number of cameras from which the model can be viewed, and two lights (Figure 6). User interaction is provided through the ability to rotate and translate the model, and to change the camera's angle of view (or zoom) in real time.

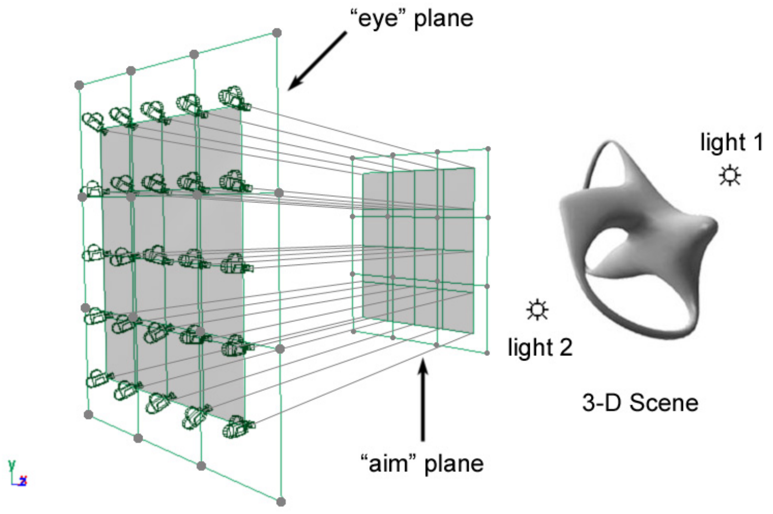


Fig. 6. Multi-Camera scene description.

## III.2. Camera Methods

The positions in three-dimensional space formed by the cameras make up a surface, which can be thought of as a "camera surface". By default, each of the camera surfaces may take a planar, cylindrical, or spherical form, as shown in (Figure 7). The entire camera system, based on the Glassner model [7], is composed of two camera surfaces, known as an "eye" surface and an "aim" surface. The eye surface contains the points at which the cameras are placed, while the aim surface contains the points at which the cameras are aimed. Each camera shape is based on a unique arrangement of control points according to the specified form, as in (Figure 8). It should be noted that the camera system may contain differently-shaped eye and aim surfaces.(i.e. an eye sphere and an aim plane).

To provide increased control over the shape of this camera surface, each initial form is derived based on a set of control points. In this algorithm, the control points are placed in three-dimensional space, and a free-form, parametric surface of cameras is then generated from these points. A diagram of the relationship between the control points and the camera surface is shown in Figure 9. Once the camera surface has been generated, the control points can be manipulated in order to shape the camera surface, which in turn will warp the output image.

An additional step can be taken to randomly dislocate, or "jitter" each camera point, so that the final outcome will be an image with a disrupted perspective. A random decimal value is added to the position of each camera point after its position on the surface is determined, creating a slightly altered surface, as shown in Figure 10.

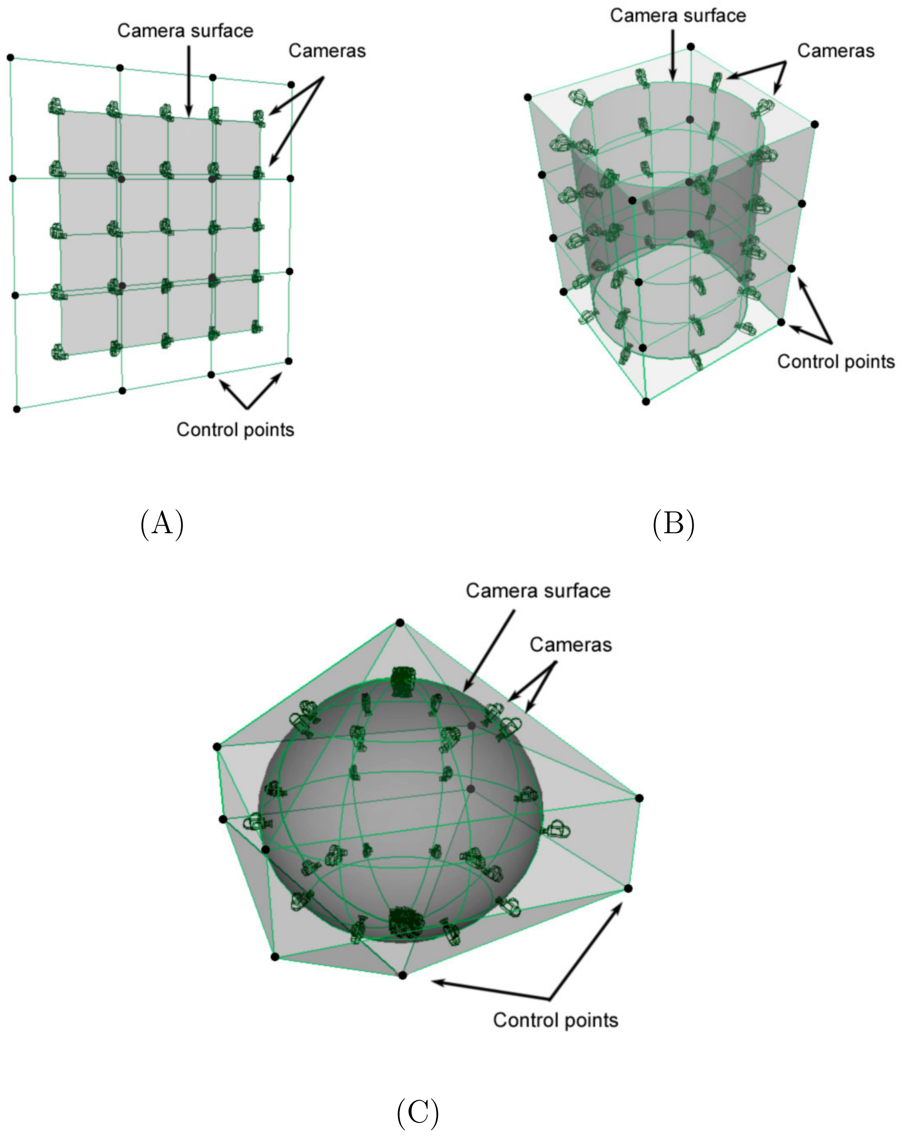
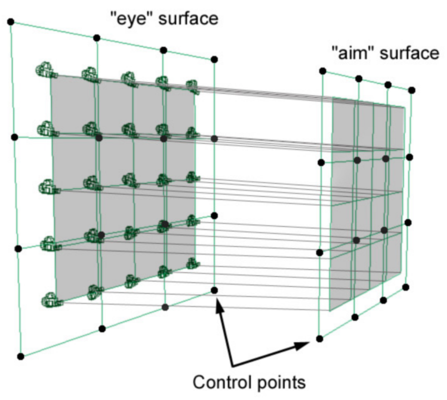
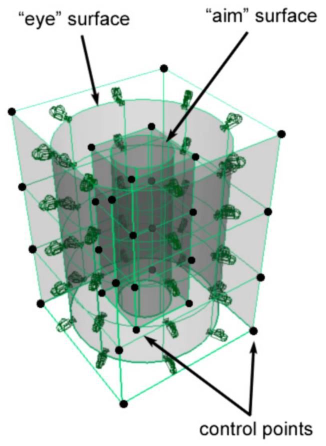


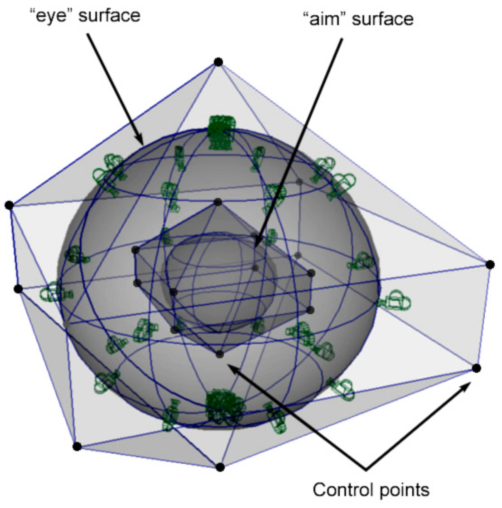
Fig. 7. Examples of multiple-camera "eye" surface shapes: (A) planar, (B) cylindrical, and (C) spherical.



(A)



(B)



(C)

Fig. 8. Examples of multiple-camera "eye" and "aim" surface shapes: (A) planar, (B) cylindrical, and (C) spherical.

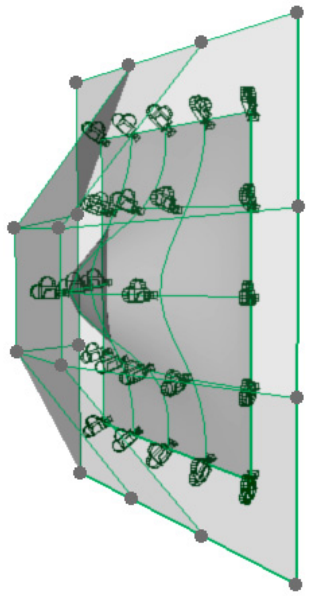


Fig. 9. Control point-camera surface relationship.

## III.3. Image Placement Methods

In addition to creating variations in the structure of the 3D camera setup, an additional step can be taken in the form of abstraction of the 2D image space. Once the initial views as determined by the camera surface positions are captured, each view is compiled into a two-dimensional collage. Dividing an initial drawing area into a grid provides the structure of the collage, with each grid square containing a unique view of the scene, as shown in Figure 11(A). Each window shows updated views of the object as it is rotated or translated interactively.

After each view is initially placed in a grid, it can then be jittered slightly to enhance the senses of movement and rhythm in the image, as shown in Figure 11(B).

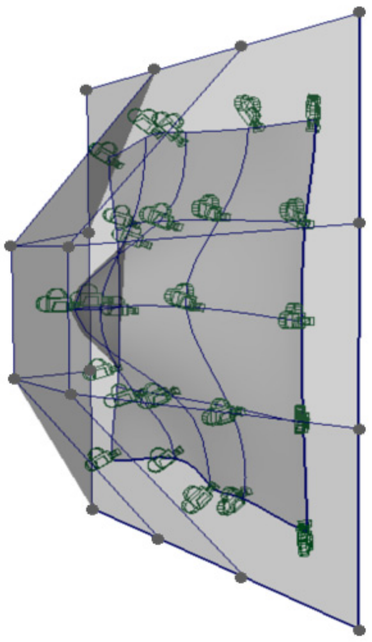


Fig. 10. Control point-camera surface relationship after addition of random value to camera positions.

To jitter the placement of an individual view, a random decimal value is added to the position in 2D coordinate space of the view. In addition to jittering, and to allow negative space to be filled,the size of each viewport can be increased or decreased causing the viewports to overlap, as shown in Figure 11(C).

To provide increased control over the placement of individual views in the collage, a system of "view-control", which is similar to the previously mentioned camera control system, is included. In the view-control system, a parametric surface of individual views is described by a set of sixteen control points, with each control point existing in two-dimensional image space. Initially, the control points are placed in positions so that each small view is arranged in a grid structure. By manipulating one or more of the 2D control points as shown in Figure 12, the shape of the grid

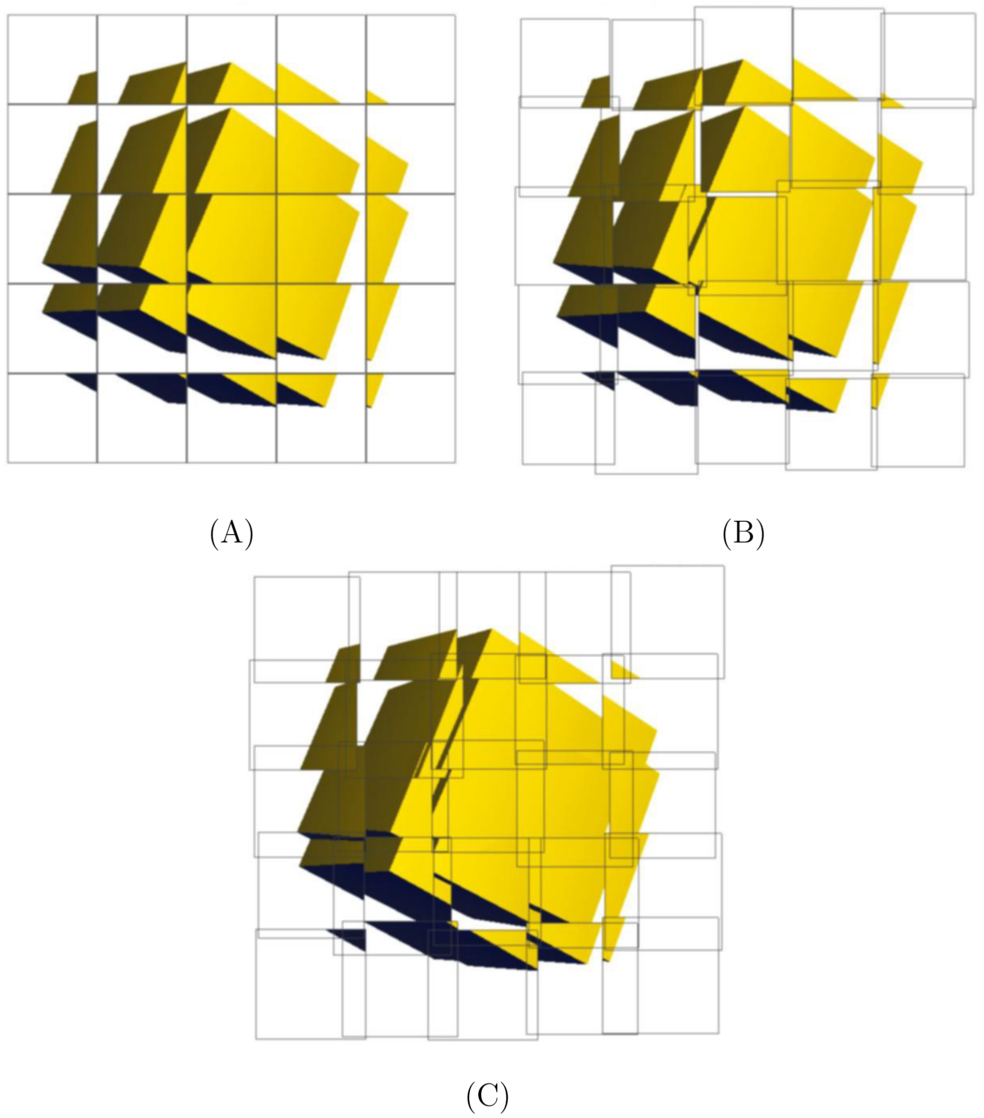


Fig. 11. Multiple-view images of a cube composed from (A) a grid of individual views, (B) individual views with random values added, and (C) individual views with random and overlap values added.

structure can be modified to a large extent.

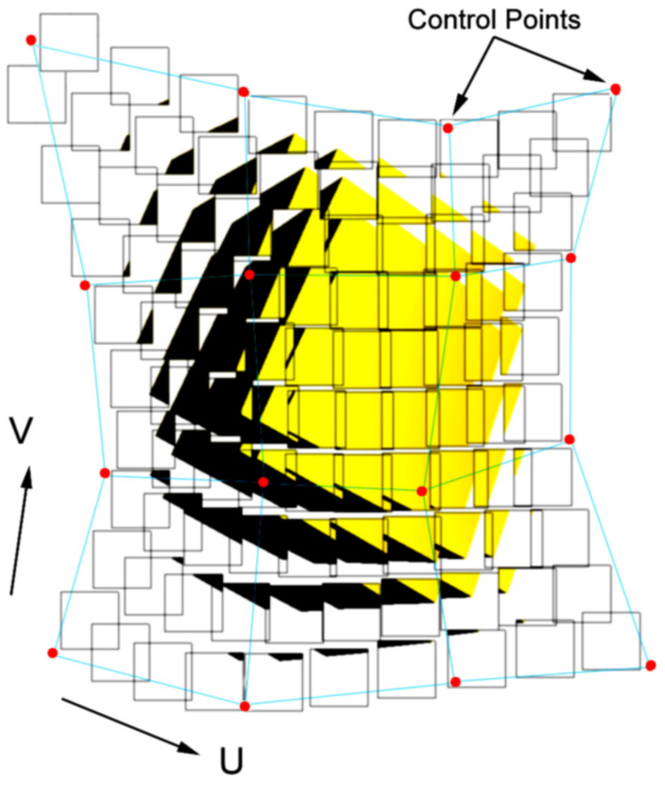


Fig. 12. Two-dimensional view surface in image space showing control points and interpolated surface of individual views of a cube.

## III.4. Lighting and Rendering Methods

An artist's control over the composition of an image depends largely on the choice of a color scheme and lighting parameters for the subject of the scene (in this case a 3D model). For these reasons, the scene in the Multi-Camera system includes two lights, whose intensities and color values may be adjusted in real-time. The specified light colors have a direct effect on the appearance of the model in the scene, as shown in

Figure 13(A). To enable a modulation effect similar to the technique used by Cezanne and other cubist painters, warm and cool light colors and intensities are allowed to mix.

To incorporate the use of line into the Multi-Camera drawing system, a ”silhouette” edge drawing option is included, which provides an outline for the 3D model as shown in Figure 13(A). The use of a silhouette outline is meant to emulate the use of line by Cezanne and the cubist painters.

Also important to an object’s appearance are surface properties, such as surface color, shininess, and texture, as well as the color of the background. The Multi-Camera system allows direct manipulation of the object’s diffuse color (local or reflected color, meaning the inherent color of the object), shininess, and specular color (the color of reflected light or highlight on the object), as well as the background color (see Figure 13(B)). Texture mapping, which allows a 2D image to be mapped onto a 3D object, resulting in an increased level of detail, is also featured in the Multi-Camera system (see Figure 13(C)). Each of these properties are manipulated with reference to the 3D model in the scene, and are constant throughout the rendering of individual views.

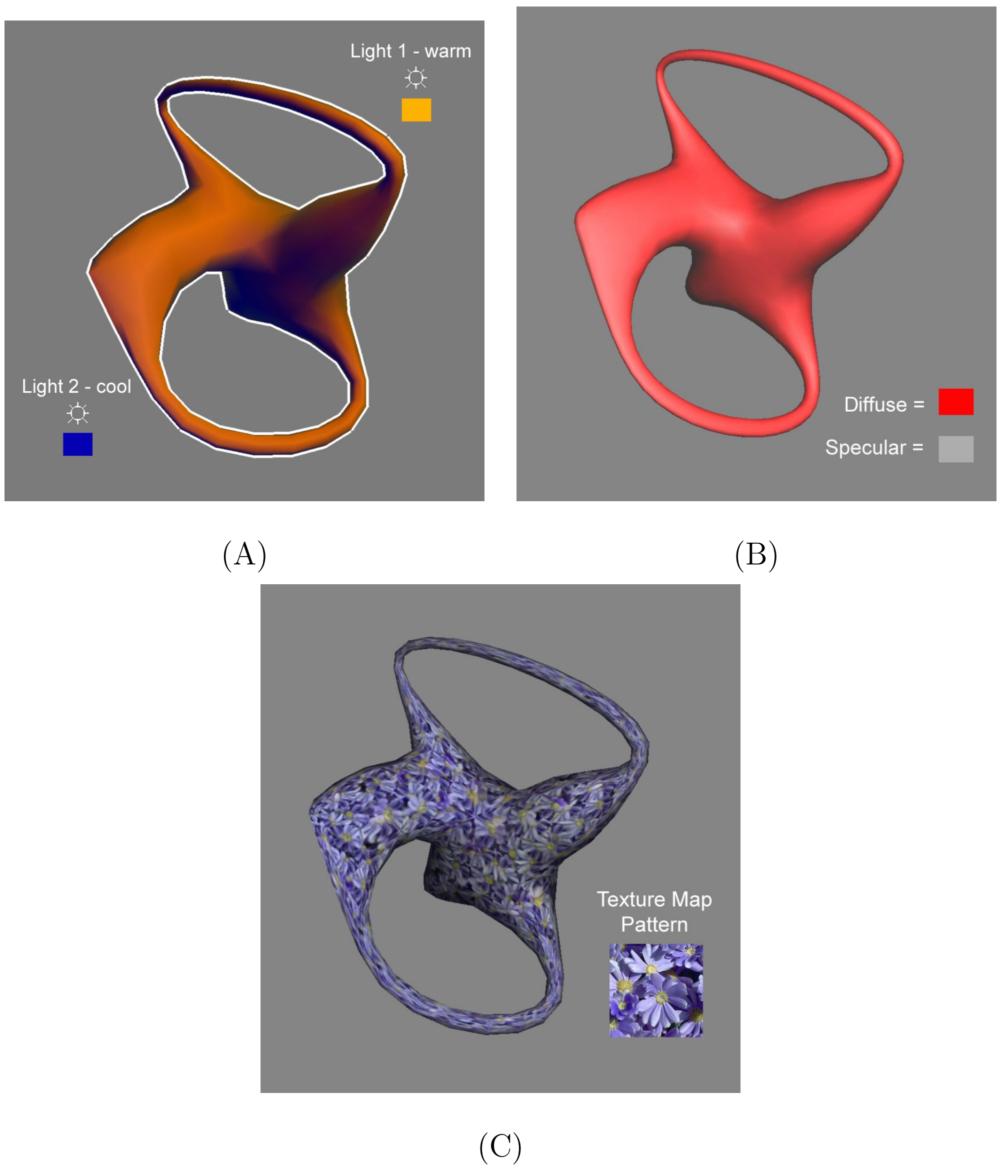


Fig. 13. (A) Diagram showing warm and cool light colors. (B) An example of diffuse and specular material colors with a shininess value of 5.0. (C) An example of a texture-mapped object.

# CHAPTER IV

# IMPLEMENTATION

This system uses the OpenGL Application Programming Interface [16] in a C++ program called "MultiCam (multiple cameras)" to create the many views and distortions needed to render objects in an abstract style. OpenGL allows the user to have control over 3D viewing transformations as well as image space or "screen" coordinates, and also is capable of rendering objects in real time. The MultiCam OpenGL program is capable of running on both the IRIX and LINUX operating systems.

This system includes a user interface that was developed using the Fast Light Toolkit (FLTK). The interface provides an organized set of controls that the artist may use to execute procedures that create abstract images. The main application window consists of a drawing area, in which the image composition can be viewed and updated, and a side option panel, which contains menu bars that display the many options for creating what appears in the drawing area (see Figure 14). The drawing area is a GLUT window, which resides inside the main window, using routines defined in the GL utility library to create images. The side option panel uses a structure of FLTK interface controls that are based on code written by Vinod Srinivasan and Michael Stanley. This section includes a number of illustrations of the MultiCam interface which show various options related to the display of a cube from multiple views.

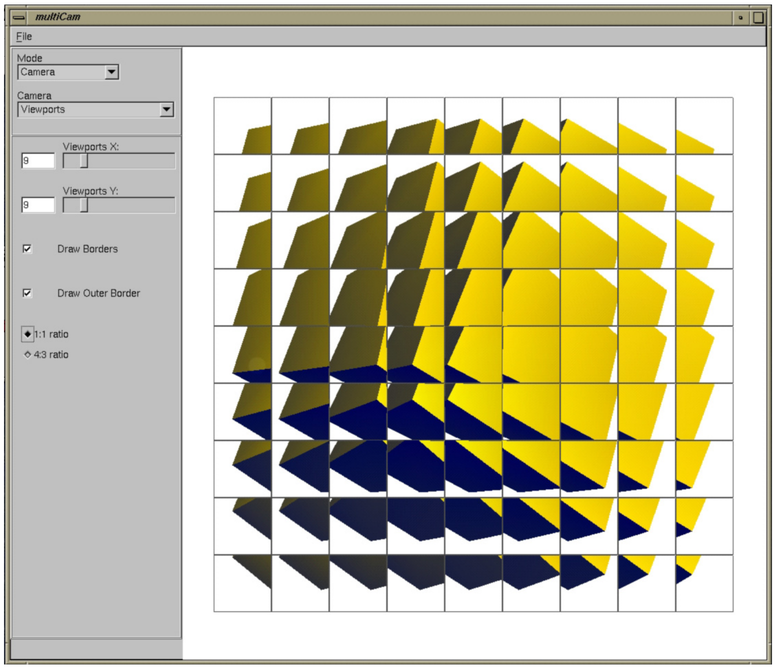
## IV.1. Viewport Options

In the MultiCam system, an individual camera is represented in the drawing area by an OpenGL viewport, which is any rectangular area on the screen specified in which to draw. This program uses the GL scissor test, which is an OpenGL technique that allows the display window to be divided into many viewports. The scissor test provides the primary break from the correct perspective of 3D viewing in OpenGL. The algorithm for using the scissor test is based on code from an OpenGL example program written by Nate Robbins. A number of options exist for controlling the number and placement of the viewports within MultiCam's drawing area.

### *IV.1.1. Viewport Placement*

From each small viewport in the main drawing area, a unique view of the scene is shown, with each view having a different viewing transformation, or vantage point. The user of MultiCam has the option of specifying the number of camera views to be placed in the horizontal (x) and vertical (y) directions in the drawing area. A view of the MultiCam interface with a multiple-view image created using the scissor method is shown in Figure 14. The series of small views of the same scene when combined will show the scene from many viewing angles, thus providing more information about the object than can be shown from a single perspective.

To provide an additional break from conventional viewing, an option for randomly "jittering" the placement of viewports in two dimensions is provided. For the cases in which the jittering produces undesired space between viewports, an "overlap" factor can be set by the user and included in the calculation for the size of each viewport. This option causes all viewports to overlap near their edges, which effectively covers the spaces between each viewport. By retaining the Z-Buffer information between the drawing of each viewport, instances of each object appear to intersect when views overlap. Examples of images with jittered and overlapping views are shown in Figure 15.

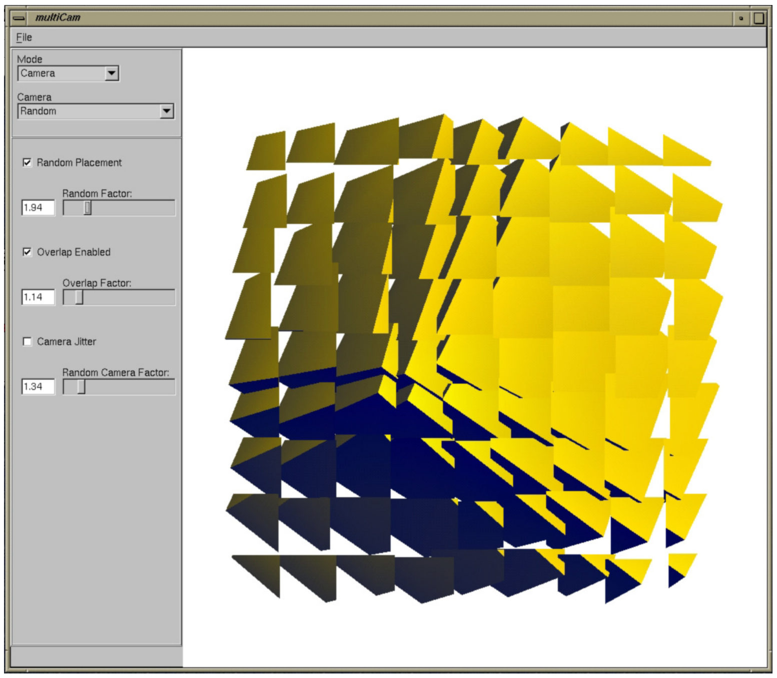


Mode: Camera
Camera: Viewports
Viewports X: 9
Viewports Y: 9
[✓] Draw Borders
[✓] Draw Outer Border
[✓ 1:1 ratio] 4:3 ratio

Fig. 14. MultiCam interface with drawing area and side option panel.

### *IV.1.2.* *Viewport Control*

An option for ”Viewport Control” is included, which allows the user to have direct control over the placement of OpenGL viewports within the drawing area. When the Viewport Control option is set, a series of sixteen control points appear in the drawing area along with the initial grid of viewports. The control points make up a control cage for a 2D spline surface of viewports. To enable varying degrees of control over the viewport surface, the user is given two interpolation options, one which is based on a uniform cubic b-spline algorithm[3], and one based on a bezier surface algorithm[19]. Through mouse and keyboard interaction, a user can manipulate individual control points and define the shape of the viewport surface (see Figure 16).



Mode: Camera

Camera: Random

☑ Random Placement

Random Factor: 1.94

☑ Overlap Enabled

Overlap Factor: 1.14

☐ Camera Jitter

Random Camera Factor: 1.34

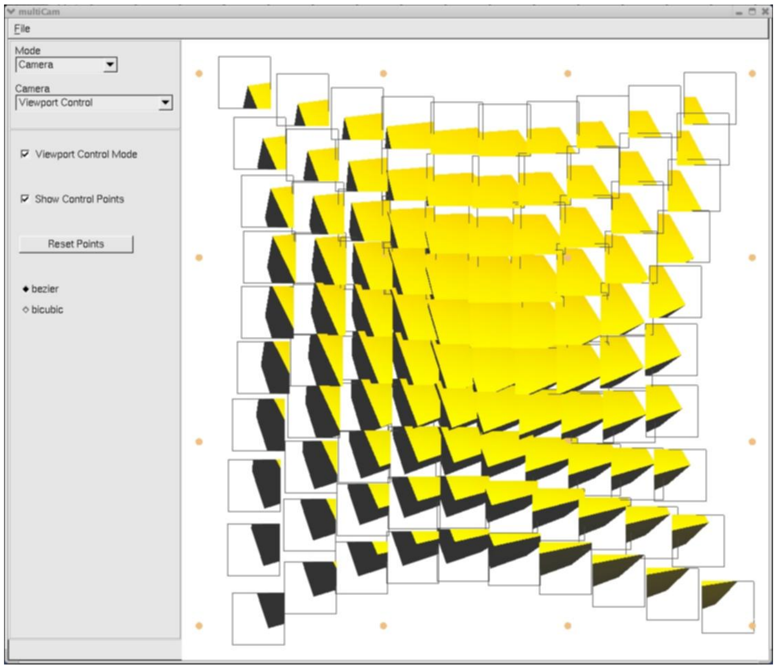
Fig. 15. MultiCam interface with jittered placement of overlapping viewports.

## IV.2. Camera Options

The MultiCam system treats the group of cameras in the viewing mechanism as well as their aim points as surfaces that can be manipulated. Each camera point in the viewing mechanism is shown in an individual view in the drawing area. Many options exist for determining the shape of the camera surface, which affects what appears in the drawing area of the MultiCam interface.

### *IV.2.1. Camera Placement*

The differing points of view for each small viewport in the drawing area are specified as a series of "eye" vectors and "aim" vectors, representing the points at which each camera is placed and aimed, respectively. Each "eye" and "aim" vector contains x, y,

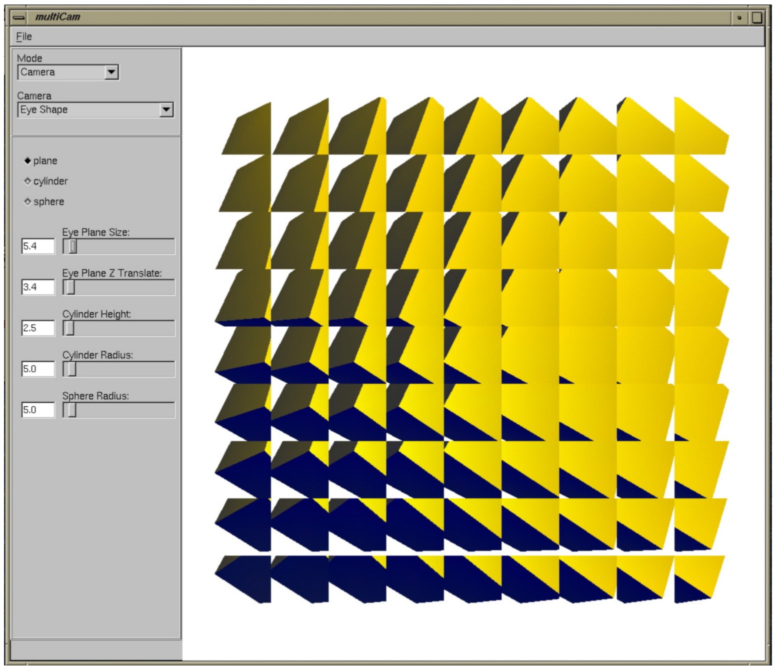


Mode: Camera
Camera: Viewport Control
[✓] Viewport Control Mode
[✓] Show Control Points
● bezier
○ bicubic

Fig. 16. MultiCam interface showing viewport-control functionality.

and z values in a 3D coordinate system. The viewing transformation of each window is determined by supplying these vectors to the "gluLookAt" function in the GL utility library. With the supplied "eye" and "aim" vectors, the "gluLookAt" function effectively places the camera object at the eye point and directs the view toward the aim point. The creation of arrays of eye and aim vectors allows for each viewpoint to be changed by performing various operations on the data in the arrays. As these values are interactively changed, the final composite view of the scene is controlled.

The area over which the viewing vectors are spread is determined by parameters entered by the user of the MultiCam system. In the MultiCam system, the overall camera mechanism is comprised of an "eye" and an "aim" shape, which refer to the forms taken by the grouped placement in space of eye and aim vectors, respectively. An eye or aim shape in the MultiCam system can begin as one of three options: a plane, a cylinder, or a sphere. In the plane option, the user specifies the size of an imaginary plane on which to spread the viewing vectors, determining the size of the eye or aim plane in the 3D space. The user can also specify the depth of each plane in the viewing direction. For the cylinder option, the user can change the height and the radius of the cylinder shape, while the sphere option allows only for the radius value to be augmented. Examples of the effects of different camera shapes on the drawing area are shown in Figures 17, 18, and 19.



Mode: Camera
Camera: Eye Shape

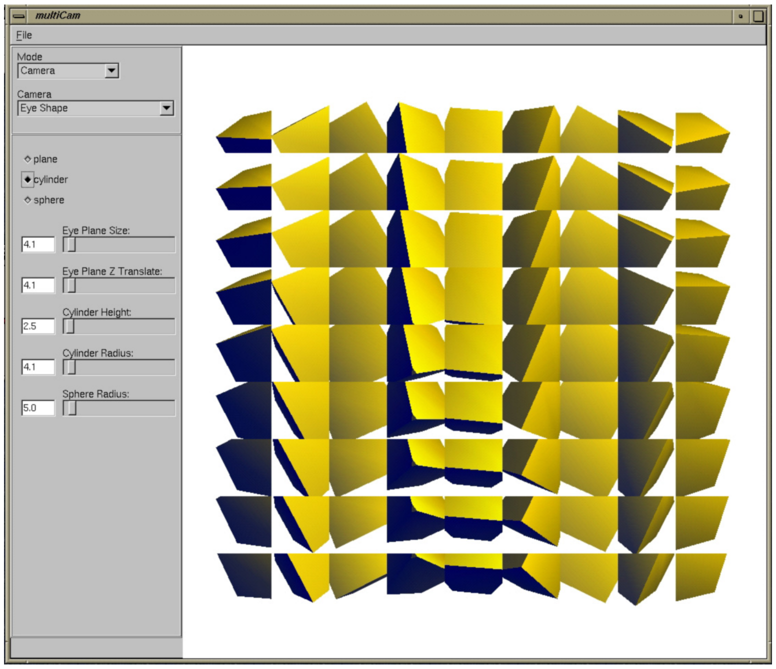
[✓ plane] cylinder sphere

Eye Plane Size: 5.4
Eye Plane Z Translate: 3.4
Cylinder Height: 2.5
Cylinder Radius: 5.0
Sphere Radius: 5.0

Fig. 17. MultiCam interface with planar camera eye shape.

## *IV.2.2. Camera Control*

The camera eye and aim shapes are determined not by explicit operations defining these shapes, but by a set of sixteen control points that determine how each camera shape is built. The control points are specified as arrays of vectors in the same way



Mode: Camera

Camera: Eye Shape

plane [✓ cylinder] sphere

Eye Plane Size: 4.1

Eye Plane Z Translate: 4.1

Cylinder Height: 2.5

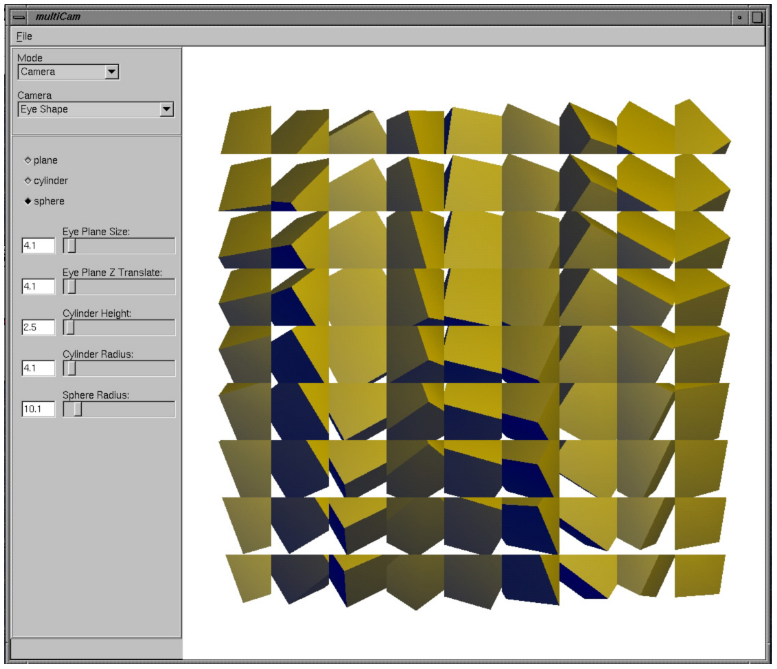
Cylinder Radius: 4.1

Sphere Radius: 5.0

Fig. 18. MultiCam interface with cylindrical camera eye shape.

that the eye and aim vectors are specified. Based on where the control points are located in 3D space, a free-form surface of cameras is generated from these points. For example, when the camera eye plane shape is selected, the size variable first places the sixteen control points in a 4x4 grid arrangement in space that is the same size as the specified size variable. A function is then called that uses a free-form surface generation scheme to determine the placement of each camera in the eye or aim surface. The user has the option of using either a uniform bi-cubic spline algorithm[3] or a bezier surface algorithm[19] to create the surface. U and V values that are generated to describe different parts of a surface are used as values for the placement of each camera in the eye or aim shape surface (see Figure 20).

The user has a number of options for manipulating camera-control points to sculpt the camera surface shape. By selecting an option to go into camera-control



Mode: Camera
Camera: Eye Shape
plane [✓ sphere] cylinder
Eye Plane Size: 4.1
Eye Plane Z Translate: 4.1
Cylinder Height: 2.5
Cylinder Radius: 4.1
Sphere Radius: 10.1

Fig. 19. MultiCam interface with spherical camera eye shape.

mode, the user is given the ability to select, by clicking with the mouse, a point in the drawing area representative of a camera-control point. Through mouse and keyboard interaction, the user can move control points in the 3D object space (see Figure 21). For example, pressing and holding the 'alt' key while dragging with the middle mouse button after clicking on a control point translates the selected camera-control point and manipulates the surface shape. Similarly, dragging with the left mouse button pressed rotates the control point for the camera aim surface relative to its corresponding control point for the eye surface. Options are available allowing the user to specify in which direction to translate and about which axis to rotate.

Additionally, the user has the option to modify the camera shape by "jittering" each view vector. Much like the option to jitter the viewports in the drawing area mentioned above, the camera jitter option allows a random value to be added to the

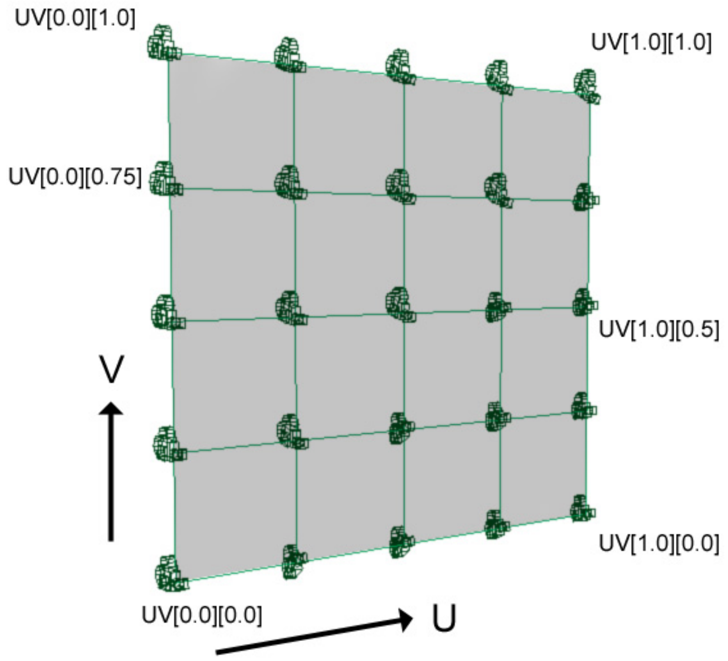


Fig. 20. U and V coordinates of a camera surface.

placement of each eye camera vector, disrupting the regular placement of the vector arrays.

## IV.3. Drawing Options

The MultiCam system includes a number of drawing routines designed to modify and enhance the appearance of an object viewed in the scene by controlling the object's world-space drawing parameters. The objects are polygonal objects that are imported into the scene from Wavefront OBJ files using functions written by Nate Robbins for importing objects using OpenGL. The OBJ files consist of a series of vertices, which serve as the corners of each polygon in the model, as well as texture coordinates that can be imported and used to display texture maps on the objects. Once imported, the

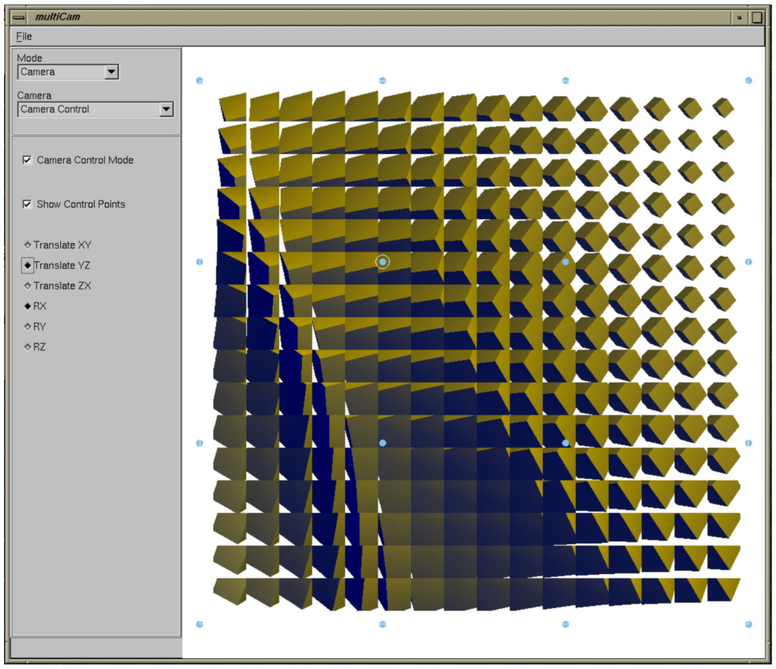
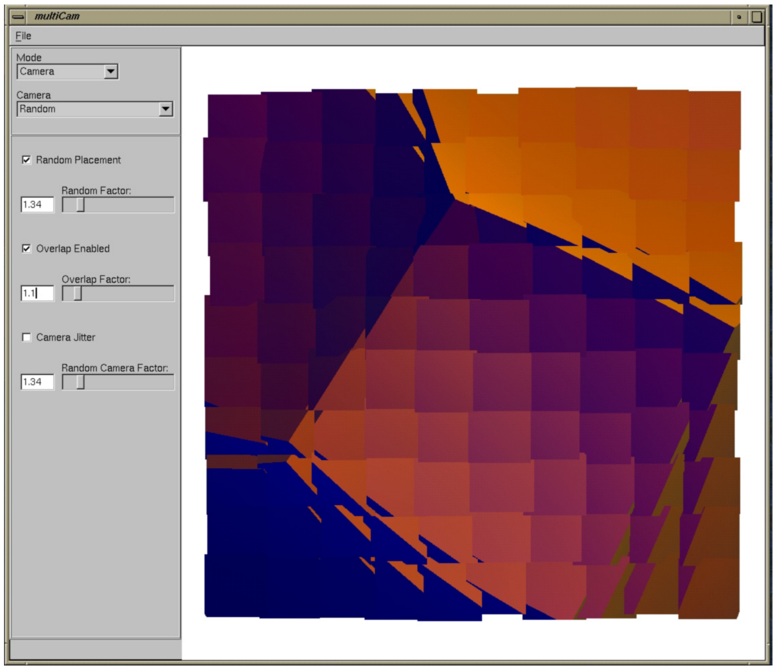


Fig. 21. MultiCam interface with camera control point manipulation.

object can be rotated through mouse control using code derived from the "Arcball" functions written by Paul Rademacher. The Arcball algorithm converts x and y screen coordinates into quaternion rotation values. Also featured is an interactive zoom control that changes the angle of view of every camera as well as mouse-driven object translation.

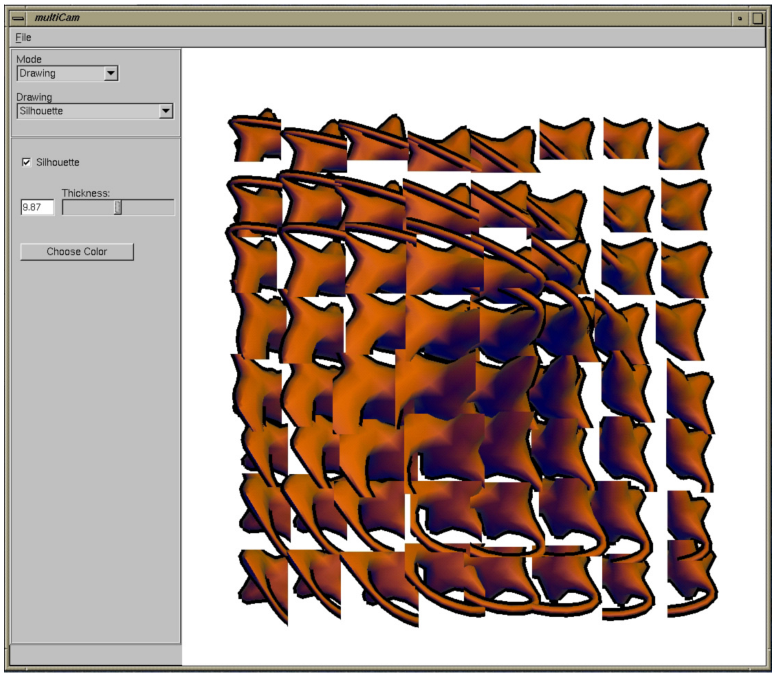
To light the object in the scene and enhance the artist's choice of color, a warm-cool lighting system is included, based on the system developed for technical illustration and adapted to use OpenGL functions by Gooch, Gooch, Shirley, and Cohen[9]. The user has the choice of using one or two lights, one designated as warm and the other as cool, and is also able to interactively choose their color. The two light colors mix across the surface of the object, adding a rich color effect to the object through a smooth color transition. This effect along with the fragmented multiple-camera view creates a modulation effect, in which colors pass from one hue to another through mixing in gradual steps (see Figure 22).



Mode: Camera
Camera: Random
[✓] Random Placement
Random Factor: 1.34
[✓] Overlap Enabled
Overlap Factor: 1.1
[ ] Camera Jitter
Random Camera Factor: 1.34

Fig. 22. MultiCam interface showing warm-cool color modulation.

Other user controls include diffuse color value, which determines the object's local color, as well as specular intensity and color value, which determine how shiny an object looks and the color of the object's highlights. Textures can be displayed through either of two OpenGL texture display modes: modulate mode, in which the color of the texture is scaled in comparison to the light color, and blend mode, in which the texture and object colors are mixed evenly[16]. Additional options for adding color to the scene are the choice of a static background color and a "silhouette" option (see Figure 23), which allows the user to experiment with line weight by specifying the color and thickness of the object's outline. The silhouette edge utility is based on code developed by Vinod Srinivasan, in which a model is drawn twice in OpenGL, once in wireframe mode without lighting calculations to create an outline, and another time with normal lighting calculations, drawn over the wireframe. All lighting and drawing options are specified in reference to the 3D model, and remain constant throughout the drawing of individual views.



Mode: Drawing
Drawing: Silhouette
☑ Silhouette
Thickness: 9.87

Fig. 23. MultiCam interface showing silhouette-edge rendering.

## IV.4. Rendering Options

The appearance of the final rendered scene will vary depending on the position of each viewing vector in object space, the position of each viewport in screen space, the rotation angle of the object, and surface and lighting properties given to the objects in the scene. For single images, a function to write TIFF (Tagged Image File Format, copyright Adobe Systems, Inc.) files has been included that was adapted from the TIFF library specification in code written by Michael Mistrot.

# CHAPTER V

# RESULTS

The MultiCam system is a tool for 3D artists that is capable of producing expressive 3D renderings of high aesthetic quality. In this section, a number of examples of artwork produced with MultiCam are shown, along with a brief discussion of the process used to create the works. The Digital Paintings section discusses MultiCam renderings that are imported into a paint program to create digital still images. In the Traditional Media section, examples of MultiCam renderings enhanced by an artist using traditional drawing media are shown. The final section provides images from abstract animations created with MultiCam.

## V.1. Digital Paintings

The worth of MultiCam as a tool to create digital paintings lies in its ability to create an initial motif for the artist. Using this system, a 3D object can be viewed from a number of perspectives and then have that viewing mechanism shaped and distorted to suit the artist's needs. In this way, the program allows the artist to "paint" the collage onto the digital drawing area by sculpting the 2D viewing mechanism and by changing the shape of the camera surface. Another way to "paint" digitally is to import the MultiCam rendering into a paint program in order to enhance the rendering using the program's many available drawing tools. Both painting methods described here were used to create the images in this section.

The first image, shown in Figure 24, was created using a simple cube object as the 3D model in the scene. Initially, the cube was rotated until the orientation was deemed satisfactory, and then the number of viewports was increased substantially in both directions, totalling 55 in X and 67 in Y. To provide a sense of texture to the rendering, viewports were jittered, with and overlap factor included. The size of the camera eye plane was increased in order to spread the viewing area out over a large distance, allowing five sides of the cube to be viewed at once. Tweaking the central control points with the camera control option along with the rotation of the original cube allowed the composition to take the form of a skewed, reversed perspective view of a cube.

After importing the initial rendering into Adobe Photoshop, the image was copied onto a new layer and blurred to provide a background texture layer. The opacity of the original image layer was decreased by a small amount in order for the background to show through. A small amount of noise was applied using the noise filter to enhance the textured look of the image. The textured look of figure along with the large number of overall cameras in the MultiCam image provides a fluid, yet somewhat jittered feel to the image.

In Figure 25, an abstract 3D model with two handles was used for the initial object. A red-green complimentary color scheme was used to provide a contrast of hue in the composition. To add highlight values to the image, a silhouette outline was used with a light green color. Random and overlap values were used, with and overlap value less than 1.0 to provide spacing between viewports. Camera and view-control parameters were adjusted so that the spacing of the viewports followed the form of the 3D model, in an attempt to introduce dynamism to the composition. In Photoshop, a background layer was produced by copying the red shapes many times and blurring them into the dark green background, slightly echoing the pronounced foreground forms. Noise was added to the entire composition to soften the originally crisp 3D rendering.

The rendering shown in Figure 26 emulates a cubist technique using a slightly

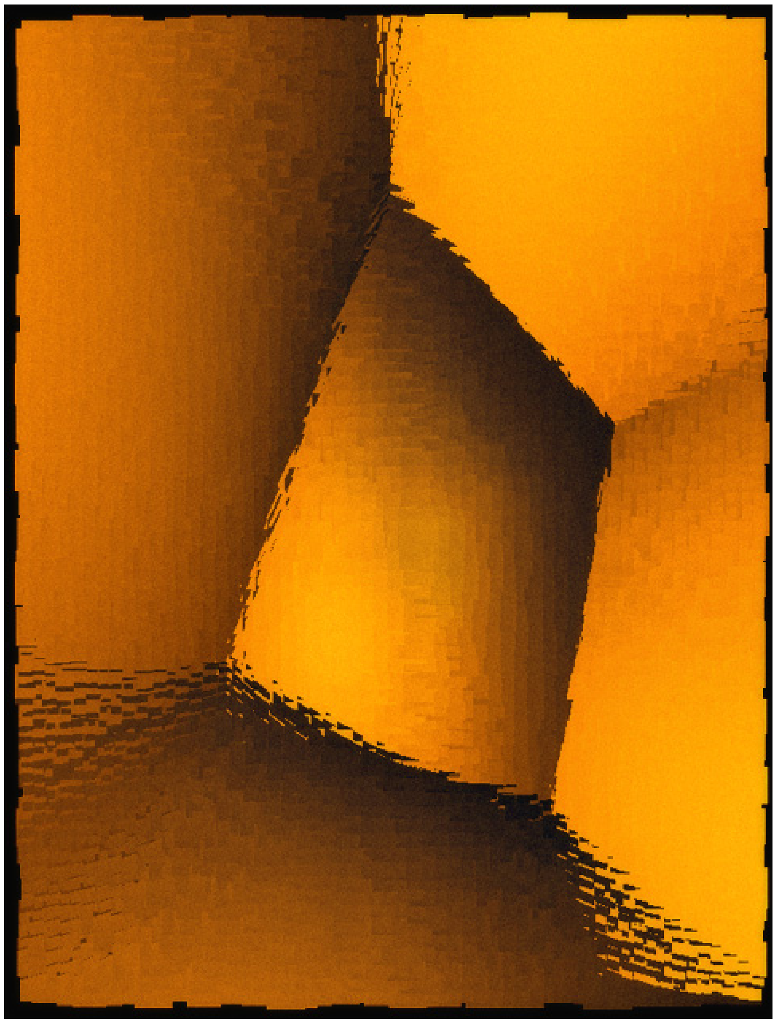


Fig. 24. Digital painting created in MultiCam showing five sides of a cube.

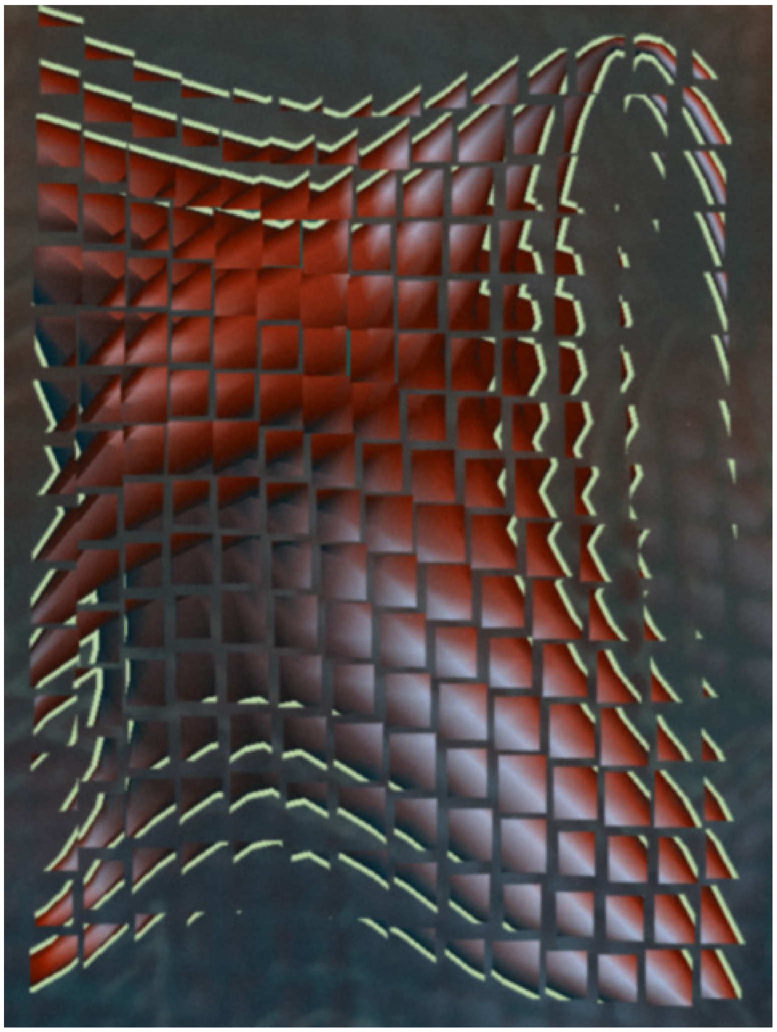


Fig. 25. Digital painting using an abstract model created in MultiCam.

randomized grid structure to display various views of a single subject, in this case, an .obj model of a head provided by Stephen Parker. In this image, 6 views in the X direction and 10 in the Y direction were created. Random and overlap factors were used to place the viewports, and additionally the view directions were skewed using the "Camera Jitter" option. For the color scheme, the warm-cool lighting system was used with a yellow warm light color mixing with a blue cool light color. To create a background image, portions of the original image were duplicated, resized, and rotated in Photoshop.

To create the image in Figure 27, again a large number of views were used (41 in X and 69 in Y). Large eye plane and aim plane sizes were used, to spread the view vectors over a large area and create a wide overall field of view for the object, which in this case was again a cube. Each view of the cube was zoomed out, so that in each small window a unique and partially complete view of the cube would be seen. Using camera control, the views in the central portion of the screen were brought closer to the object, while the outer views remained zoomed out. The large number of cameras brought a fluid feel to the image, while the negative space in the outer portion of the image created a semi-transparent appearance, reminiscent of an afghan blanket or a curtain of beads. Using the view-control system, views were shifted to enhance the feeling of flow in the image. To create a background in Photoshop, portions of the original image were copied, offset, and blended into the background dark blue color by decreasing the opacity levels of each portion.

## V.2. Traditional Media

To test the applicability of MultiCam's output to traditional art media such as drawing and painting, a number of abstract images produced in MultiCam, including the

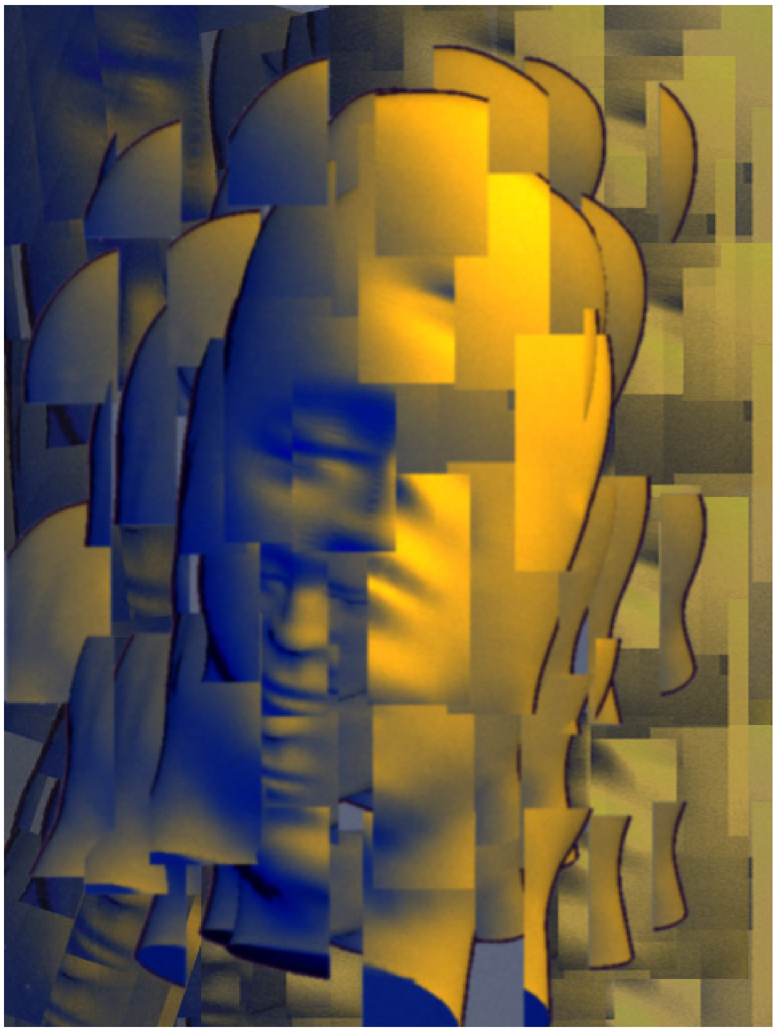


Fig. 26. Cubist-style digital painting created in MultiCam.

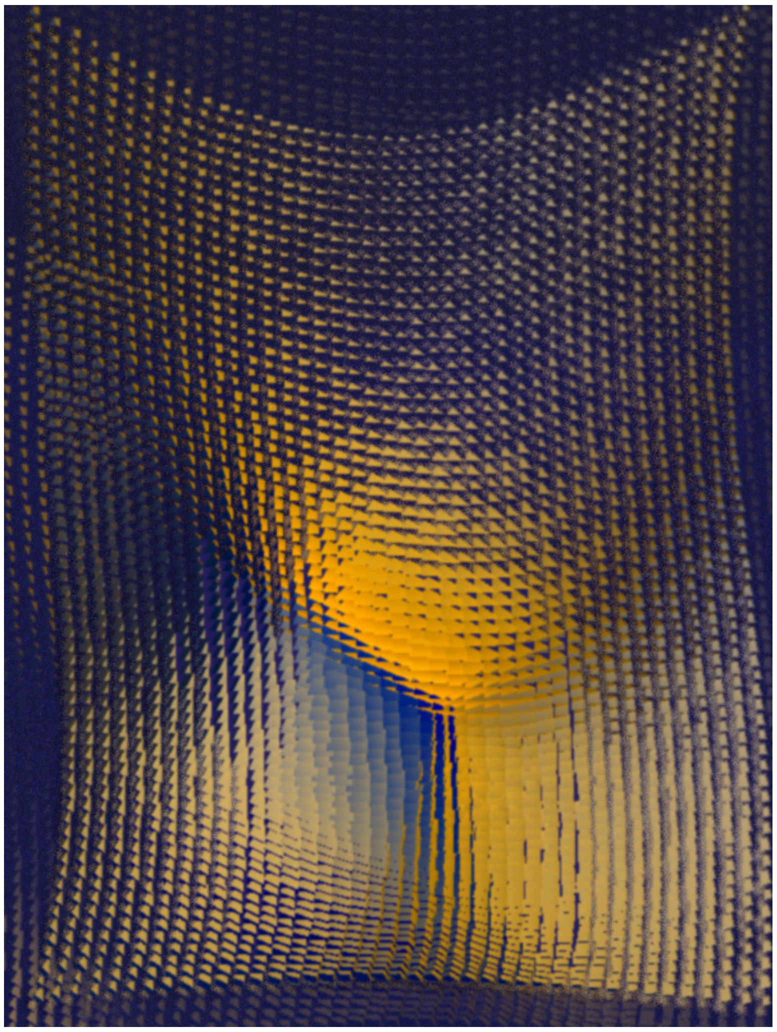


Fig. 27. Abstract digital painting created in MultiCam.

one shown in Figure 28, were given to an artist, Prof. Dick Davison, to work with. In the image reprinted in Figure 29, the artist used pastel chalks and acrylic paint to develop a color scheme based on the geometry in the original greyscale image. Another version of the rendering, shown in Figure 30, was done in charcoal and tinted with pastel chalks. In both images the space in the original rendering was extended through line and color work.

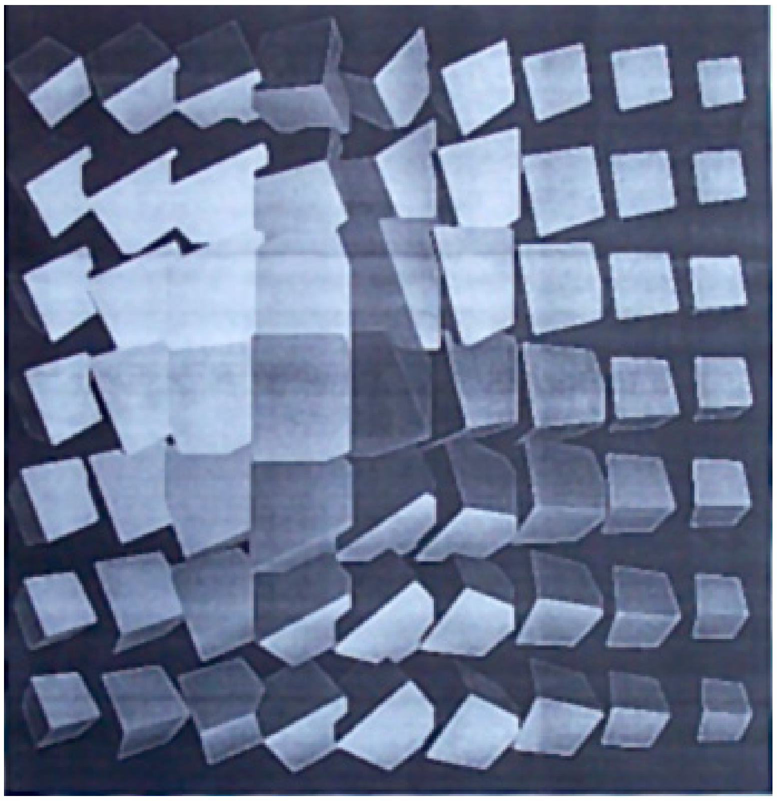


Fig. 28. Reprint of original MultiCam rendering.

## V.3. Animation

In addition to still images, a number of abstract animations have been created using MultiCam. Abstract movement was obtained by interpolating 3D model rotation and

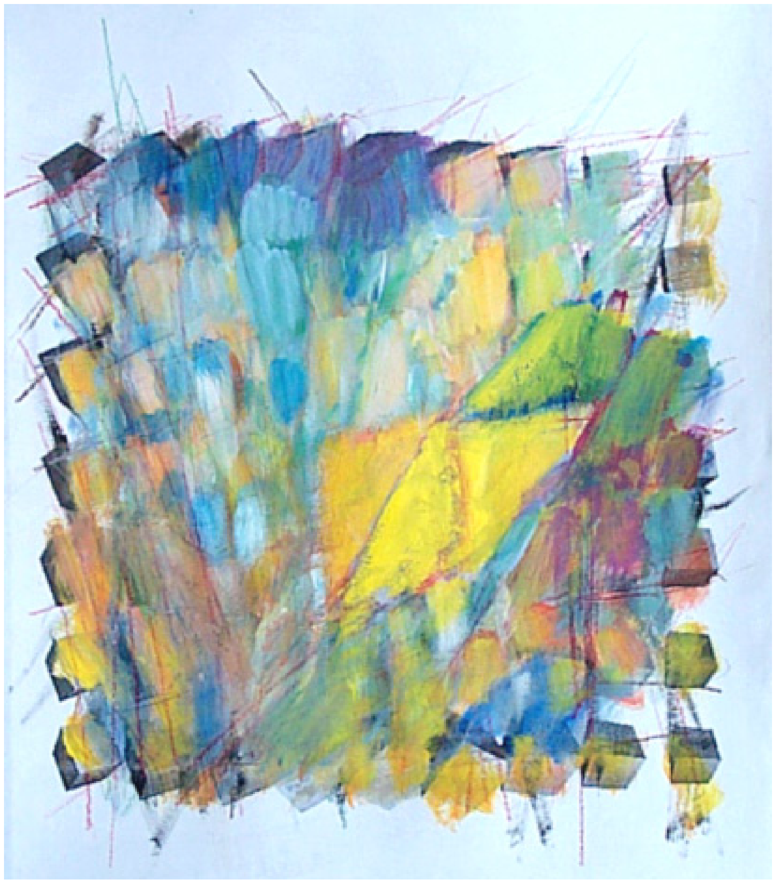


Fig. 29. MultiCam rendering worked over with acrylic paint and pastel chalk by Richard Davison. Image used with permission of Richard Davison.

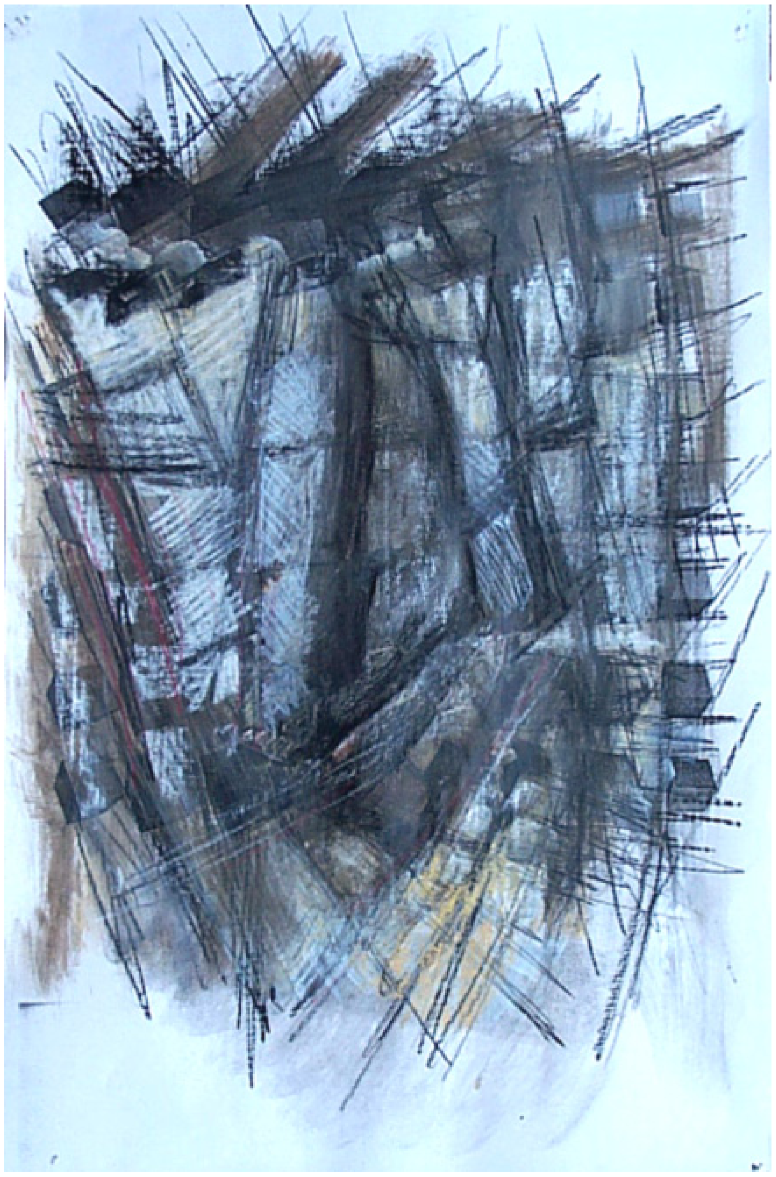


Fig. 30. MultiCam rendering worked over with charcoal and pastels by Richard Davison. Image used with permission of Richard Davison.

translation values, sizes of camera eye and aim shapes, and positions of view-control points and camera-control points. In Figure 31, frames are shown from an animation involving 48 cameras with morphing camera surfaces viewing a rotating 3D model. In Figure 32, frames are shown from an animation involving approximately 250 cameras horizontally placed along a cylindrical camera surface. Because a large number of cameras in one direction were specified, the disc-shaped model is viewed from nearly all sides and the camera interpolation appears smooth.

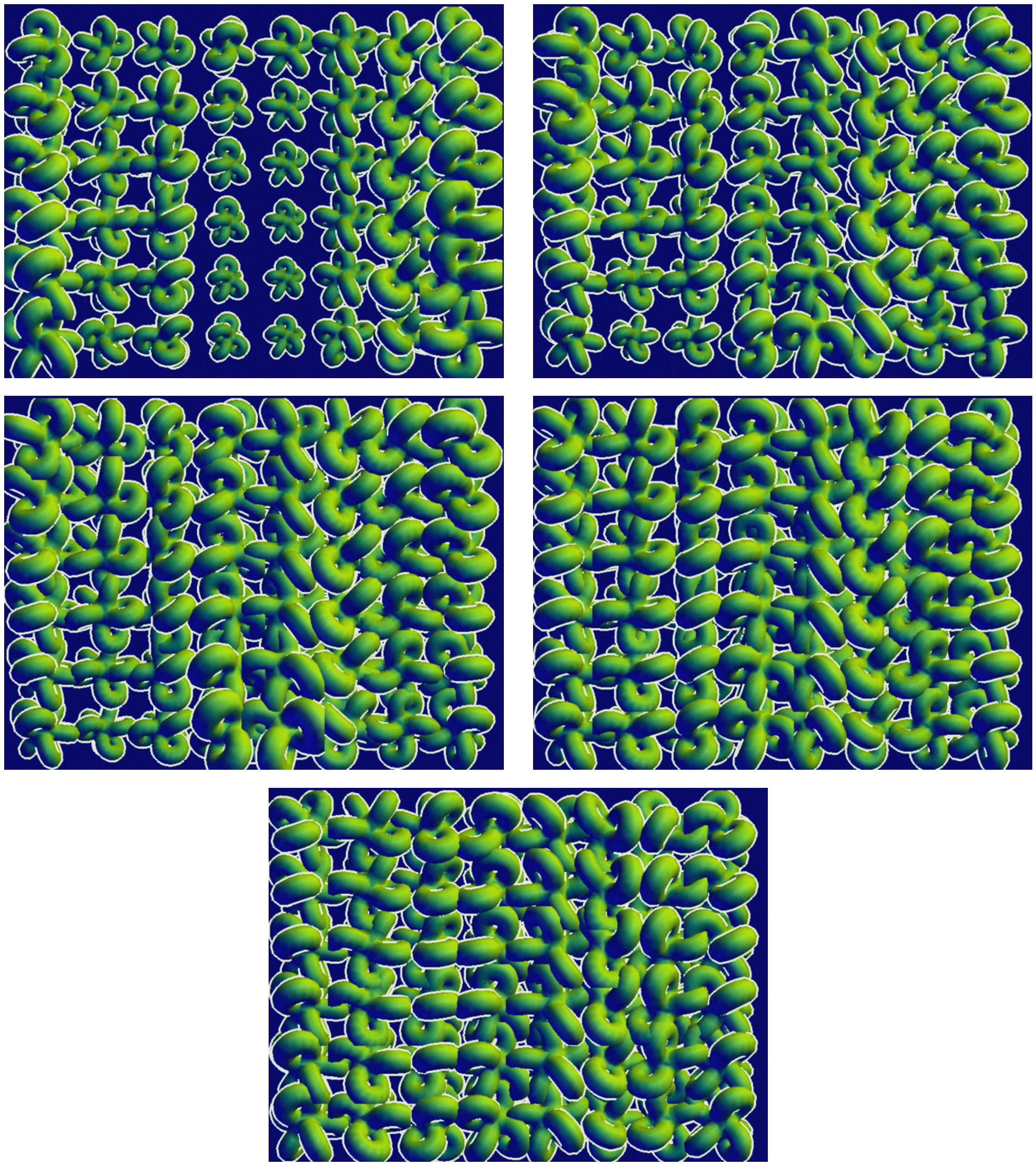


Fig. 31. Frames from an abstract animation created with 48 cameras in MultiCam.

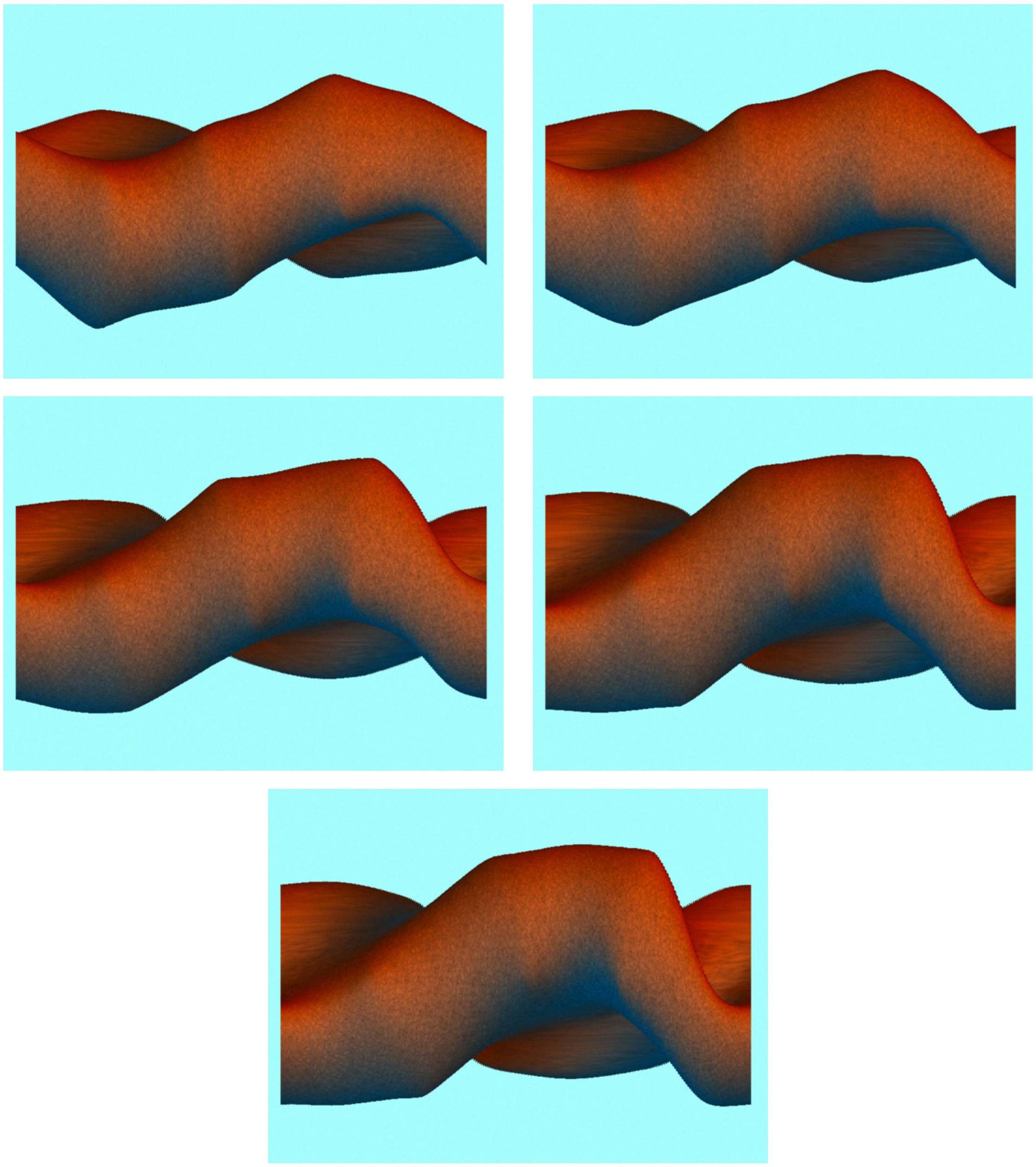


Fig. 32. Frames from an abstract animation created with approximately 250 cameras in MultiCam.

# CHAPTER VI

# CONCLUSION AND FUTURE WORK

The tools available to create 3D computer graphics are becoming increasingly sophisticated. While these tools push the limits of what can be done in the photorealistic realm, the limits to what can be done if current highly-developed rendering algorithms are applied to the creation of abstract works are virtually unimagined.

A number of previous abstract rendering approaches from an array of sources are linked by their goal of warping camera parameters to create abstractions and by the inspiration of traditional abstract art movements. While these tools laid important groundwork in developing a multi-perspective viewing apparatus, they lacked the ability for artists to see the results of the changing of multi-perspective viewing parameters in real time.

In this research, a tool has been developed that allows artists to experiment with multi-perspective viewing parameters in real-time through mouse-driven control of the size and shape of camera surfaces. Through an efficient and user-friendly interface, artists can adjust color and lighting information, material properties, and drawing techniques in addition to being able to control the multi-perspective camera properties of a 3D scene. Additionally, the real-time nature of the MultiCam viewing mechanism allows artistic computer image manipulation to take on an interesting mixture of two- and three-dimensional forms in the same interface, providing a unique system of creating computer-generated art.

Future research endeavors in the area of Multi-camera rendering for artistic purposes can include a number of options for improving the functionality and increasing the flexibility of the MultiCam software. Functions to read in more complex scenes and use more complex surface material properties can be implemented. Alternatively, the camera surface information can be exported to separate, more sophisticated photo-realistic rendering software.

A number of ideas exist for building on functionality already present in the MultiCam system as well. In its current state, MultiCam provides controlled interpolation over the placement of viewports and camera positions only. During the process of rendering each small view, any number of modeling and rendering options can also be interpolated. For instance, object motion (not implied by camera changes) can be interpolated across viewports. Light colors and positions can also change during the rendering process and be illustrated by multiple camera views. Bounding volumes can be interpolated by the camera views so that the interiors of the objects can be seen through sections cut by the bounding volumes. Another example would be to have the program read in two .obj models, and render one or the other in each viewport based on a blending function, in order to create a morphing effect.

An additional limitation of the MultiCam interface is that even though each small window is scaleable in size, because they are OpenGL viewports they must remain rectangular and cannot be rotated. Functionality can be added that provides non-rectangular windows that can be manipulated and adjust in size in real time, giving the user additional control over the output.

# REFERENCES

[1] E. Akleman and S. Meadows, "Abstract Digital Paintings Created with Painting Camera Technique", *Proc. D'ART 2000/Information Visualization 2000*, London, United Kingdom, July, 2000.

[2] E. Akleman and S. Meadows, "Camera Painting", http://www-viz.tamu.edu/faculty/ergun/research/artisticdepiction/

[3] R.H. Bartels, J.C. Beatty, and B.A. Barsky, *An Introduction To Splines for Use in Computer Graphics & Geometric Modeling.* Los Altos, Calif.: Morgan Kaufmann Publishers, Inc., 1987.

[4] A. Chen, K. Knudtzon, J. Stumpfel, and J. Hodgins. "Artistic Renderings of Dynamic Motion", *Conference Abstracts and Applications of SIGGRAPH 2000*, p. 177, July, 2000.

[5] D. Cooper, *The Cubist Epoch.* New York: Phaidon Publishers, Inc., 1971.

[6] C. Curtis, S. Anderson, K. Fleischer, and D. Salesin. "Computer-Generated Watercolor", *Proc. SIGGRAPH '97*, pp. 421-430, Aug., 1997.

[7] A. Glassner, "Cubism and Cameras: Free-form Optics for Computer Graphics", *Microsoft Technical Report*, MSR-TR-2000-05, http://www.glassner.com/andrew/cg/research/cubism/cubism.htm

[8] A. Gooch, B. Gooch, P. Shirley, and E. Cohen, "A Non-Photorealistic Lighting Model for Automatic Technical Illustration", *Proc. SIGGRAPH '98*, pp. 447-452, Aug., 1998.

[9] B. Gooch and A. Gooch, *Non-Photorealistic Rendering.* Natick, Mass.: A.K. Peters, Ltd., 2001.

[10] C. Goodman, *Digital Visions Computers and Art.* New York: Harry N. Abrams, Inc., 1988.

[11] P. Haeberli, "Paint by Numbers: Abstract Image Representations", *Proc. SIGGRAPH '90*, pp. 207-214, Aug., 1990.

[12] M. Halle. "Multiple Viewpoint Rendering", *Proc. SIGGRAPH '98*, pp. 243-254, Aug., 1998.

[13] C. Knight, "Composite Views: Themes and Motifs in Hockney's Art." *David Hockney: A Retrospective.* Organizers, M. Tuchman and S. Barron. New York: Harry N. Abrams, Inc., 1988.

[14] B. Meier, "Painterly Rendering for Animation", *Proc. SIGGRAPH '96*, pp. 477-484, Aug., 1996.

[15] M. Mohr, Personal website, http://www.emohr.com/index.shtml.

[16] OpenGL Architecture Review Board, M. Woo, J. Neider, T. Davis, D. Shreiner, *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2.* Reading, Mass.: Addison-Wesley, 1999.

[17] P. Rademacher and G. Bishop. "Multiple-Center-of-Projection Images", *Proc. SIGGRAPH '98*, pp. 199-206, Aug., 1998.

[18] H. Read, *A Concise History of Modern Painting.* London: Thames and Hudson, 1974.

[19] D. Rogers and J.A. Adams, *Mathematical Elements for Computer Graphics.* New York: McGraw-Hill, Inc., 1976.

[20] K. Sims, "Artificial Evolution for Computer Graphics", *Proc. SIGGRAPH '91*, pp. 319-328, July, 1991.

[21] P. Smith, *Interpreting Cezanne.* London: Tate Publishing, 1996.

[22] S. Snibbe, Personal website, http://www.snibbe.com/scott/index.html.

[23] S. Snibbe and G. Levin, "Interactive Dynamic Abstraction", *Proc. NPAR 2000*, pp. 21-29, June, 2000.

[24] C. Utterback. "Liquid Time: An Exploration of Video Cubism", *SIGGRAPH 2000 Conference Abstracts and Applications*, p. 223, July, 2000.

[25] D. Wood, A. Finkelstein, J. Hughes, C. Thayer, and D. Salesin, "Multiperspective Panoramas for Cel Animation", *Proc. SIGGRAPH '97*, pp. 243-250, Aug., 1997.

# VITA

**Jeffrey Statler Smith**
16213 Wall St.
Houston, TX 77040
jeffsmitty77@hotmail.com

**Education**

| | |
|---|---|
| M.S. in Visualization Sciences | Texas A&M University, December 2003 |
| B.E.D. | Texas A&M University, August 2000 |

**Employment**

| | |
|---|---|
| Graduate Assistant, Teaching | Texas A&M University, September 2000 - May 2002 |