Developing a Paradigm for Visualizing Architecture using Computational Methods: an Analysis of the Havana Project by Lebbeus Woods

by

Gregory E. Anderson

S.B. Art and Design
Massachusetts Institute of Technology, 1994

Submitted to the Department of Architecture in partial fulfillment of the requirements for the degree of

Master of Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 1996

Signature of Author: [illegible]
Department of Architecture
May 10, 1996

Certified by: [illegible]
William J. Mitchell
Dean, School of Architecture and Planning
Thesis Supervisor

Accepted by: [illegible]
Roy Strickland
Chairman, Departmental Committee on Graduate Students

Thesis Readers: Lebbeus Woods
Adjunct Professor of Architecture, Cooper Union

Developing a Paradigm for Visualizing Architecture using Computational Methods: an Analysis of the Havana Project by Lebbeus Woods

by

Gregory E. Anderson

Submitted to the Department of Architecture
on May 10, 1996 in Partial Fulfillment of the
Requirements for the Degree of Master of Science

ABSTRACT

This thesis is concerned with developing a more detailed and efficient process for visualizing architectural forms with computational tools. The thesis will examine the origins of computer visualization and its current implementation to ascertain its inherent deficiencies. A case study, in which a conceptual project by Lebbeus Woods will be represented using this process, will serve as a means for examining these deficiencies and proposing possible solutions. A series of studies will be conducted to test the viability of such solutions and the results will inform the development of a more structured model for applying computer visualization.

Thesis Supervisor: William J. Mitchell
Title: Dean, School of Architecture and Planning

Acknowledgments

I would like to extend my sincerest thanks to my Advisor, Bill Mitchell. Your unwavering faith in me during the most difficult times has been truly inspirational.

To my family (Mom, Dad, Melanie and Dot) - Without you all, I would not have made it through the many obstacles that were placed before me. Your unconditional love and sacrifices have made this possible.

To Dean Isaac Colbert - Your wisdom and guidance has been greatly appreciated and will shape my character for years to come.

To Larry Sass - We both know that I could not have finished this without you. Thank you for being a true friend and mentor.

To Lebbeus Woods - It was truly an honor working with you. Thank you for giving me the opportunity to learn from you.

To the Brothers of Kappa Alpha Psi - Thank you for helping me keep my sanity.

Last and definitely not least, my warmest thanks to Diane Holmes. I will never forget your undying support and love that carried me through this all.

Table of Contents

Chapter 1

Introduction

1.1 Introduction

The thesis is concerned with examining and developing the method of representing architecture using computer visualization. This process combines computer-aided design with physical-based simulation to produce imagery for visualizing architectural spaces and forms. While similar in purpose to more traditional, hand-rendering techniques, computer visualization involves a particular methodology that differs in its execution. This difference becomes significant for determining how and when this method is used. Since computer visualization for architecture is still in its infancy, an examination of the process will help discern not only proper techniques for clarifying spatial concepts, but also better understand the strengths and weaknesses of this medium.

The origins of computer visualization (CV) will serve as a background for understanding current and future uses. Computer visualization, as a technology and a technique, was initially developed nearly thirty years ago, but has only evolved into a mere drafting tool for many architectural practices. While its ability to represent complex physical environments is beginning to be explored by a minority of architects, its true potential remains to be discovered.

Visualization becomes of particular importance for understanding and developing architectural concepts which can not or will not be realized in any physical manifestation. Herein, the thesis will use a project by Lebbeus Woods as a case study for delineating a methodology. Entirely conceptual in design, such a project is uniquely suited for representation in an environment not constrained by physical limitations.

Computer visualization is a complex process involving multiple layers of planning and implementation. Traditional methods of visual representation group many tasks into broader and simpler steps. For instance, in hand-rendering, light and textures can be delineated simultaneously through the intensity of pencil strokes. However, to achieve similar effects using computational tools, these qualities must be adjusted separately then combined many times in the generation of one computer image. Modeling, lighting, rendering and animation are each separate processes, inextricably linked to form an entire visualization process. The thesis seeks to examine all of these processes and their relationships. In accomplishing this. the thesis seeks to illuminate the planning essential to generating computer imagery and how it is useful.

Chapter 2

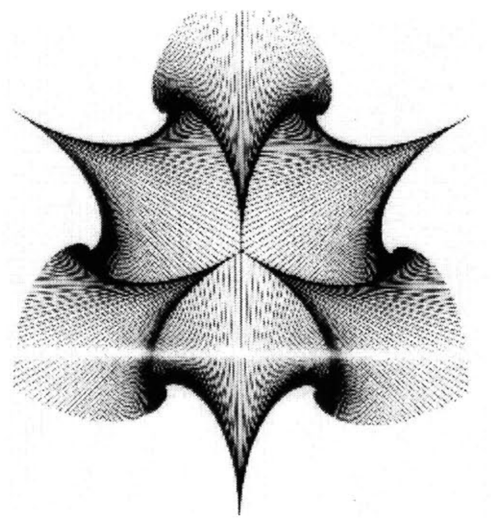
Fig. 2.1 - One of the first sophisticated computer graphics images. (CalComp , 1968)

Background

2.1 Origins of Computer Visualization

Before discussing issues of CAD applications to architecture, it is necessary to relay the origins of computer graphics, as a technology in itself. In the early 1960s, the construction of computers was entering its second generation. Transistors had replaced vacuum tubes as internal relays and contributed to both an increase in speed of computation and a decrease in overall size of machinery. These attributes allowed more commands to be executed in a shorter amount of time. This was essential to the development of computer graphics. In 1963, Ivan Sutherland, a Ph.D. student at Massachusetts Institute of Technology, pioneered the launch of computer graphics with the development of Sketchpad (Lewell 1985).

This program initially used a pen pointer and monitor to draw lines and shapes on the screen. However, the true significance of this software lied in its data structure, which was based on object topology. Essentially, within an object's data structure were descriptions of relationships of the individual components of an object. For instance, it could distinguish the fact that while an edge of an object may be hidden from view, it still existed and would be visible if the object were transformed(scaled,translated or rotated) in some manner. In this way, a clear distinction was made between the data structure and the visible image on-screen (Lewell 1985). By recording the components of an object and determining their positions in relation to each other, this system foreshadowed computer-aided design and computer visualization.

This breakthrough research sparked extreme interest from industry and academia alike.

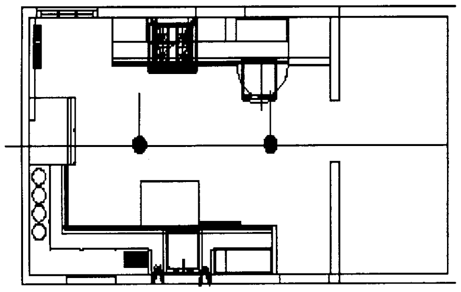

Fig. 2.2 - Image of 2D CAD drawing

The development of computer-aided design, which is simply the industrial use of computer graphics for designing objects, soon followed. While the simplification of CAD into specific categories is hotly contested by experts, the two areas that relate specifically to use within an architectural framework are drafting and geometric modeling. Drafting, the least sophisticated technique of the two, is the representation of 2-dimensional mechanical drawings in plan or elevation, as it were. Geometric modeling takes advantage of the types of data structures mentioned earlier and allows for the representation of 3-dimensional objects. The definition of objects can then be recorded by its edges, surfaces or volumes, with an increasing amount of data necessary for each method of recognition.

Despite an abundance of breakthroughs in CV, it did not begin to make a mark on the architectural profession until the 1980s, and even then, it was relegated to only the largest firms, due to the prohibitively expensive hardware. As the purpose of computers was thought to be the enhancement of productivity during the document drawing process, early CAD tools were only utilized for their ability to compute repetitive functions. By allowing the computer to handle the monotonous tasks during this stage of the design process, they were able to decrease the amount of time needed for generating documents, thereby increasing their efficiency and productivity. Hence, CAD existed primarily as a drafting tool. geometric modeling was hardly ever utilized. While this may seem counterintuitive in the design of 3-dimensional physical structures, there were fundamental reasons for this occurrence. First, the tools were not specifically designed for use within the architectural design process. Computation is inherently linear in its execution. A series of commands must be initiated, in a particular order, for a desired result to be achieved. This method is contrary to many established practices architects use to design structures. However, architects also contributed to the inefficiency of CAD for broader use, by not specifying their needs to CAD developers. Architects, generally, could not see how a computer, which ultimately simplifies all decisions to true/false, on/off or 1/0, could contribute in any meaningful way to a process dependent upon quantitative and qualitative decision making. The fundamental flaw in this rationale, which still pervades architecture today, is that computational tools and methods can not and should not be adapted to the standards of traditional ones.

2.2 Applications of CV in Architecture

Computer visualization in an academic setting takes a completely different form than CV in the profession. One primary reason for this disparity is the structure of the academic environment. First, many academic institutions can provide greater access to computer resources per person than firms. Many institutions are committed to providing students with the best equipment currently available. This results in students having access to the latest technology, whether its hardware (computers, peripherals, etc.) or software (CAD programs, etc.). Additionally, a sophisticated support system comprised of manuals and expert help is normally available to maintain equipment and assist in the use of software. This factor confines equipment problems to those skilled in troubleshooting and removes this burden from the user, in many cases. Also, particularly with software, professors, teaching assistants, and lab assistants can serve as resident experts on the use of a specific computation tool.

The types of CV tools available in academia differ significantly from those employed by firms. Typically, the structure and implementation of CV tools reflects the variance of the processes undertaken by architects and students. Software used in academia is less specific to the necessities of architectural practice. This flexibility permits the software to be manipulated in a variety a ways which may be unconventional or impractical for practice; like rendering multiple frames for a photorealistic computer animation which can take hours or days of processing time. Moreover, there are usually a greater number of tools with differing functions available. Specific programs exist for simulation of lighting effects, modeling, animation and a host of other CV techniques which give students the freedom to explore visualization across multiple platforms. These advantages afford students the increased flexibility they need for expanding their academic pursuits.

Implementation of CV is carried out through design studios and related coursework. Design studios may vary in their requirement for use of CV techniques. Depending on the format of the studio and the project, computational tools may not be used at all. Others will mandate a heavy use of them. The common thread that makes either situation better for experimentation than that of a firm's environment, is that both require the conceptualizing of forms and spaces that probably will never be constructed in any physical form other than small models. This liberty from the practical constraints of building construction allows for greater expansion of theories and concepts which exist beyond the scope of current architectural building technology. However, concepts of this nature demand a high degree of development and representation. Traditionally, this takes the form of sketches, drawings and models, as visualization is key to understanding spatial and formal components of design ideas. But, students, typically, are given the option of choosing their own method of visualization on most projects. For this reason, great potential exists for advancing CV as a supplementary means of representation along with other traditional ones.

Other coursework which supports the experimental nature of the academic environment for supporting CV are actual visualization courses. These classes specialize in depicting architecture-related imagery. Normally, they involve a series of small projects and one final project, which make use of the techniques taught during the course of the semester. Students are given a choice of buildings that interest them, and then must manipulate a particular building along specific guidelines. For example, a two-dimensional plan drawing may be the first step, followed by a three-dimensional model. From there, they may be responsible for applying textures and rendering the final building. The objective of these types of classes differ with their focus and complexity. While lower-level classes may center on the process outlined above, more advanced classes focus on explicit techniques and theories related to visualization.

2.3 Visualization Issues

Despite the use and potential of CV in the academic environment, there still remains a great deal of lost or untapped potential. When dealing with CV, the focus centers too heavily on the production of polished, final images. These images are then scrutinized on their clarity, composition, and usefulness to evoking the ideas of the project. While those issues are of importance, they do not comprise the total significance of CV because final imagery does not relate the difficulties involved with the translation process from conventional media to computer media. In other words, to represent images of physical forms and spaces, drawings, models, and photographs must be used as a basis to create a computational model of the architecture. That process is not straightforward and encompasses a specific type of abstraction with its own set of unique problems. Additionally, due to advances in CV technology, photorealistic images can be generated relatively quickly. Considering the short amount of time required for the technology to reach this level of refinement, it's evident that future explorations in computer graphics will have to focus in areas beyond photorealism. The process for creating images, however, is unexplored at best. By examining the intricacies of this translation process, new and clearer methods can be created which add a different dimension to representing architecture.

Chapter 3

Methodology

3.1 Intent

The primary purpose of this thesis is to analyze and develop the process of applying computer visualization tools for the representation of architectural forms. To examine this process, a case project, an architectural landscape designed by Lebbeus Woods, will be translated into a digital model and further developed through a series of lighting, rendering and animation studies. The case was selected because it possesses specific qualities that are suited for representation in a virtual environment. First, the project is unbuilt and will not be carried through any construction-related phases. Second, the design is entirely conceptual in its intention and scope. No measured drawings will be used at any time. Third, the project is delineated only as a series of sketches and small-scale models. These qualities allow the latitude necessary to explore unconventional methods of representation without the constraints of design and construction issues. A sequence of experiments will test the viability of said methods and their use in representing the physical characteristics of the landscape.

3.2 Visualization Phase: Modeling

During this stage of development, I will document the steps taken to translate the ideas portrayed in the architects' sketches and models into a digital model. The intent is to define how one merges design information from two distinct, conventional mediums into a computational environment. This will be accomplished through analysis of the landscape and the software . Technical constraints will play a significant role in the abstraction of design elements to the computation environment.

Modeling experiments will also be conducted to determine the benefits of particular modeling techniques and their effect on files sizes and image rendering.

3.3 Visualization Phase: Lighting

After construction of the initial digital model, the author will conduct a series of lighting studies to simulate various natural and artificial lighting effects. All applied textures will be rendered as flat, mid-grey and mid-brown tones, used strictly for examining modeling details. They will also be employed for introductory lighting studies with software designed to determine sun location and shadow position. Experiments will be carried out that simulate the interaction of natural light with the landscape, based on the site's global coordinates, season, and time of day. These studies will influence future lighting schemes, to be designed in the original modeling environment, by providing qualitative and quantitative data on the behavior of natural light. It is anticipated that significant changes in the digital model's design will take place during the course of these lighting trials. Due to the interdependence of the modeling and lighting, adjustments will be necessary to each, as unpredicted information is received. The complex relationship of planning, modeling, and lighting will be explored and documented as encountered.

3.4 Visualization Phase: Rendering

During this stage, various images will be rendered of the digital model using the modeling, lighting and texture data. Due to the complexity of the actual technical process of generating imagery, only the factors which are directly relevant to the completion of this research will be studied and documented. These factors affect image quality, rendering speed, resolution and ray-tracing method. To examine rendering methods, a series of tests will be executed to compare each method's performance and resultant image quality.

Additionally, one model will be rendered with two separate rendering techniques to determine the benefits of render speed vs. image quality. Last, images of one model will be rendered and timed using various camera positions. This will assist the selection of particular final camera views based on not just their aesthetic composition, but their render performance as well.

3.5 Visualization Phase: Animation

The last phase will entail developing the animation of the digital model. Animation involves creating a sequence of images in which motion is simulated through an environment, either through camera movement, object movement or any other change in the environment over time. While this technique lends itself to a great deal of creative expression, the thesis will focus on the technical details of how an animation is created. The process of storyboarding, its description and design, will be reviewed. The thesis will also explore potential new methods of storyboard development that are unique to computer animation. Various methods of animation will be analyzed and discussed, comparing the processes of camera and object animation, detailing their uses and inherent qualities.

3.6 Justification of Technical Experiments

Technical studies which gauge render times, file size, and memory constraints are of great significance in planning the size and scope of a project that requires CV techniques. The thesis will attempt to show the importance of recognizing as many technical limitations as possible, by examining how the software interprets designl information from the user. By documenting these "real-world" problems, the author can better understand and therefore, anticipate, future obstacles, their reason for existence, and methods for avoiding them through technical and creative solutions.

3.7 The Case Study

The case study being used is the Havana sea wall project by Lebbeus Woods. In its conception, the sea wall exists along a six mile stretch of the Havana coastline and obstructs the ocean during high tides of storm season. The wall is comprised of independent wall sections which rotate up to face the ocean during high tide and rotate down to a horizontal position, once the tide subsides. In its horizontal position, it becomes an habitable landscape between the boardwalk and the ocean.

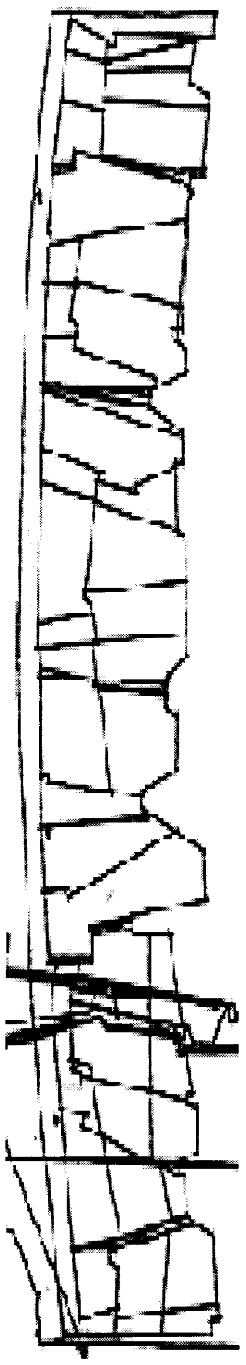
Fig. 3.1 - Plan view of sea wall

This project was chosen because it provides an excellent opportunity to represent an unbuilt, and arguably unbuildable, architectural concept. The large, moving structures, which would be impossible to represent physically, lend themselves to depiction through digital means which are better suited for simulation. Additionally, since the project is still under conceptualization, feedback from the architect is possible and will enhance the discussion of CV for conceptual projects.

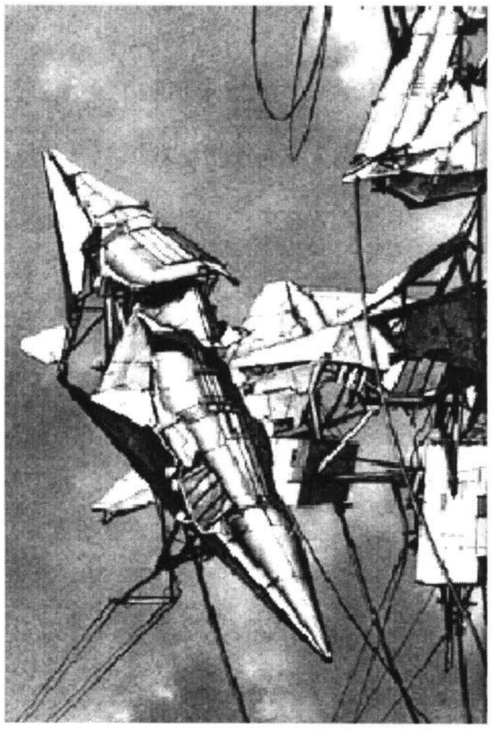
Fig. 3.2 - Design by Lebbeus Woods (ANARCHITECTURE: Architecture as a Political Act)

3.8 The Architect

The author chose a design by Lebbeus Woods because his work focuses on abstract concepts within architecture. Measured drawings and built structures are not necessary for the discussion of his ideas. His work relies heavily on its representation, most of which is in the form of elaborate two-dimensional drawings. Many of the structures he has envisioned are dynamic, large scale moving pieces of architecture, which seem to defy real-world limitations. The interactions between these structures can not be simulated in any tangible way, other than small models. However, CV provides a means of simulating these interactions without the physical constraints, thereby representing them in an ideal form, analogous to the manner in which they were originally conceptu-

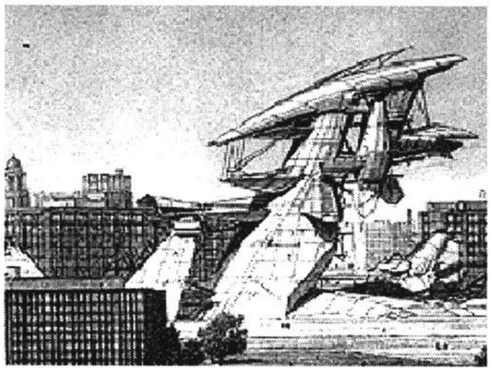
Fig. 3.3 - Design by Lebbeus Woods (ibid.)

alized. Furthermore, his illustrations of these environments possess certain cinematic qualities that contribute greatly to understanding his design theories and lends itself to interpretation through CV techniques.

Chapter 4

Analysis

4.1 Strategy

The analysis will examine the process of visualizing architectural spatial and formal components in a computational environment. This will entail translating these components from traditional sketches and models to a digital model, in addition to analyzing other computational methods for representation, such as rendering and animation. By studying this process through a case example, a conceptual project by Lebbeus Woods, I intend to document some of the inherent inefficiencies of this medium and generate possible solutions. Ideally, these efforts will assist the development of a better paradigm for understanding and using this process.

4.2 Purpose of Computer Visualization

Computer visualization exists as a means of

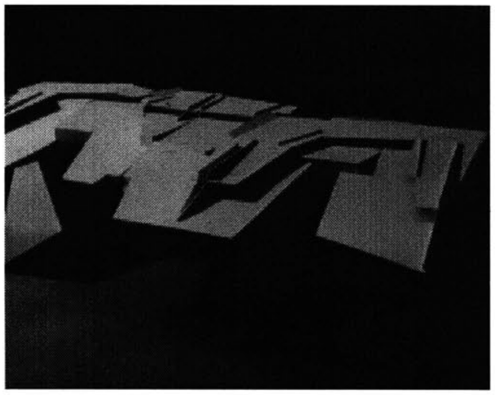
Fig. 4.1 - Havana Sea Wall by Lebbeus Woods

clarification through abstraction. Abstraction is defined as the quality of a thing that has been separated from the thing itself (Perron and Miller, 1993). Hence, through computational abstraction of a landscape, in this case, a conceptual structure, integral elements are extracted and idealized to form the basis of a new virtual landscape. This provides for a clearer understanding of the original because the computational model illuminates the relationships and patterns of the most fundamental aspects that define the landscape, which would otherwise be obscured by the complexities of the physical environment (Perron and Miller, 1993).

In the case described here, a conceptual landscape is being represented. It exists only in the form of drawings and models, as originally delineated by the architect (see Fig. 4.2) . Thus, it is already in an abstracted form. However, due to the limitations of conventional visualization techniques, I argue that there remains a great deal of information to be discovered through computational means. Since the virtual environments created in the computer are not bound by the conventions and laws of the physical world, computer visualization seems uniquely suited to represent an unreal, and arguably already "virtual" landscape of this nature.

4.3 Strategy

Computer visualization (CV) involves a series of individual processes with complex interdependencies. While CV may seem to progress in a linear fashion, (see Fig. 4.3) I have found that it is more cyclical in nature (see Fig. 4.4). For example, it is obvious that Rendering and Animation depend on and are informed by discrete Modeling information. Generally, rendering is executed much faster when proper modeling techniques are used, which in turn, allows for faster generation of animations. However, Rendering also informs Modeling by illustrating model dis-

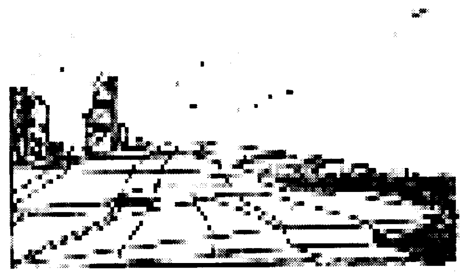
Fig. 4.2 - sketch of Sea Wall by Lebbeus Woods

crepancies like gaps and overlapping surfaces (Kerlow 1995). Thus, it becomes evident that a strategy must be developed which anticipates this back and forth communication and expedites it as much as possible. By observing the technical constraints of the software and the design requirements of the landscape, I can better determine how to undertake each process, thereby minimizing this cyclical communication.

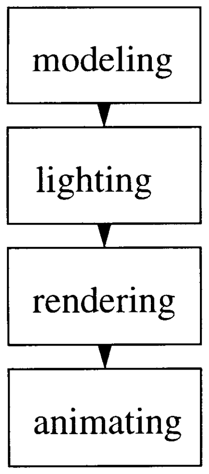

Fig. 4.3 - Linear model of CV process

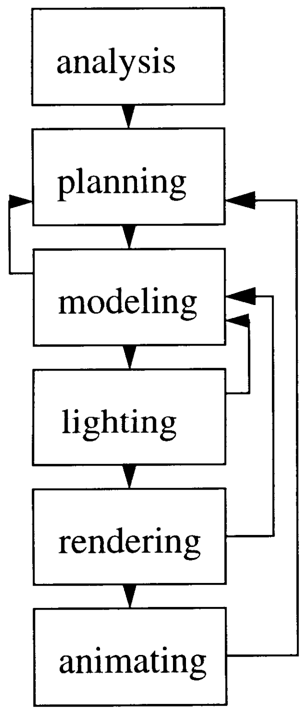

Fig. 4.4 - Cyclical model of CV process

4.4 Technical Constraints

The primary software environment, Alias Studio ver.6, serves as a multi-functional platform capable of supporting much of the visualization demands. It's fairly intuitive GUI (Graphical User Interface) offers seamless integration of multiple viewing windows for observing models in perspective, plan or elevation. In addition, Alias possesses comprehensive functionality for manipulating modeled objects. It has an extensive database of functions to control construction of various modeling elements. This flexibility frees the user to create nearly any form imaginable, which is essential for architectural abstractions which employ complex formal and spatial components.

However, despite these advantages, Alias does bear a few limitations. It's modeler relies on NURBS (Non-Uniform, Rational B-Splines) to create surfaces (see Fig. 4.6). NURBS are defined, and ultimately confined, by mathematical formulae which determine the shape of a surface based on weighted control vertices (Foley 1994). While this feature can be a powerful tool, it does constrain the alteration of surfaces to mathematical functions, which can slow the rendering of a model considerably by increasing computation time (Kerlow 1995). Other constraints involve lighting of models. Alias does not offer a means of computing accurate day-lighting simulations. These are useful in architectural visualization by

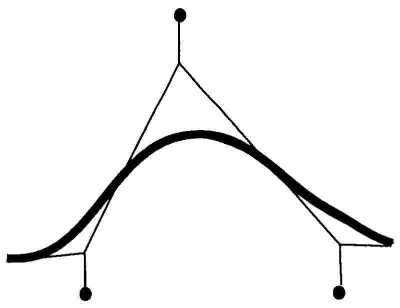
Fig. 4.6 - NURBS curve

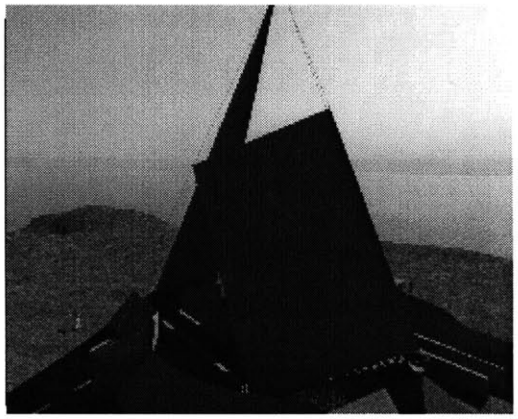
Fig. 4.7 - Radiance image

predicting how natural light interacts with architectural forms (see Fig. 4.7 and 4.8). While Alias provides a high degree of control over artificial light sources, the lack of attention to natural lighting mandates the use of Radiance ver. 2.5, an architectural lighting package.

Radiance is a ray-tracing program which allows for a high degree of control over many of the variables essential to calculating the behavior of natural light. Because it simulates the physics of light within a user-defined environment, the user can achieve precise, predictable lighting effects. By placing a model within this environment, sun position and shadow qualities can be obtained according to the model's positions in global coordinates, date and time of day. But, Radiance requires particular file formats to translate model data, which Alias can not provide. So, a conversion program was necessary to transfer files from Alias to Radi-

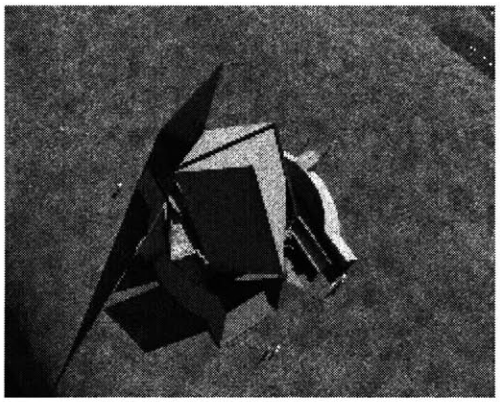
Fig. 4.8 - Radiance Image

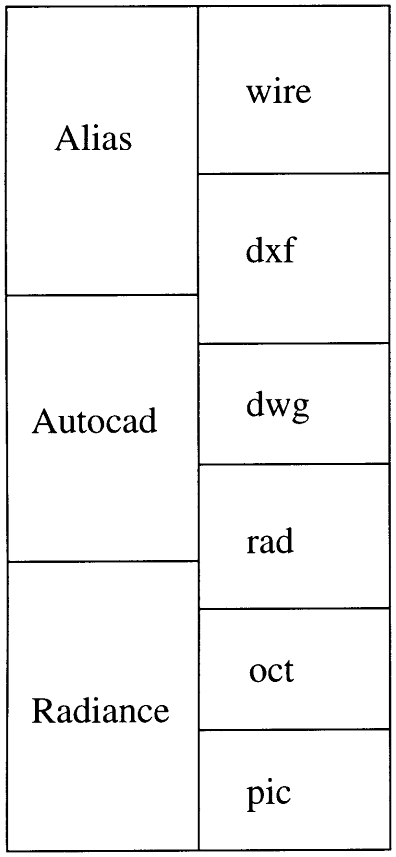

Fig. 4.9 - File Conversion process from Alias to Radiance

ance.

Autocad release 12 was used to accomplish this task. Autocad can interpret DXF files from Alias and , ultimately, convert them to the files that Radiance needs to generate imagery (see Fig. 4.9).

By examining the intricacies of the technical constraints of the software, I was better prepared to analyze the design components of the case project. I could anticipate and push the limits of the software, without sacrificing modeling efficiency or time spent troubleshooting.

4.5 Design Analysis

Through analysis of the conceptual landscape, based on experience in the use of CV and knowledge of the specific technical constraints of the software, critical elements can be separated for abstraction to a computational environment. While the selection of

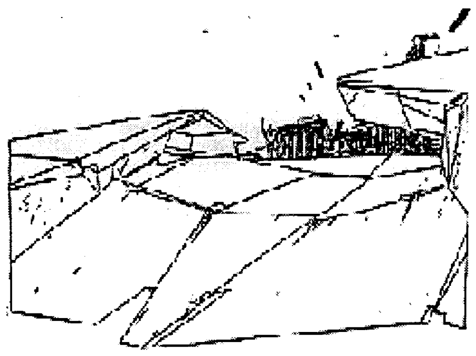
Fig. 4.10 - Sketch of sea wall

some elements may be software-specific, depending on the software's power and limitations, abstraction with these criteria can be quite practical and revealing. As an example of this, I examined a particular design component of the case project.

The sea wall forms an intricate, undulating tiled landscape with shade-giving outcroppings that project from the surface (see Fig. 4.10). By magnifying the area that forms the connection between the surface and outcroppings, an even more complex tiled surface becomes evident (see Fig. 4.11). Two separate factors can help ascertain whether these details should be considered and represented literally in the computational model. First, other imagery from a different medium (physical models) can be examined. From this data, for reasons not specified by the designer, this tiling is not present (see Fig. 4.12). Also, an attempt to model these types of details using NURBS would be extremely

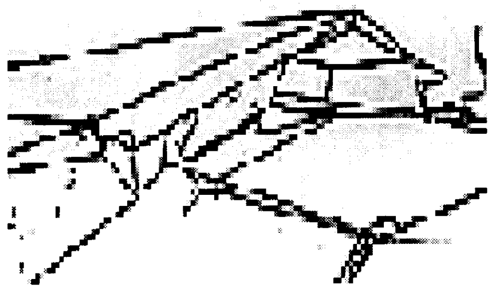
Fig. 4.11 - Magnified sketch

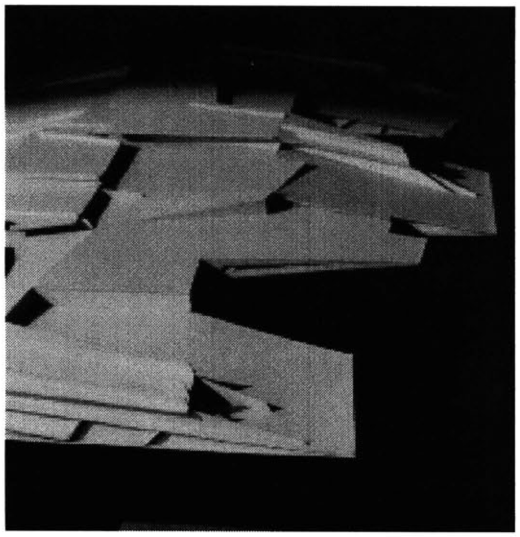
Fig. 4.12 - Image of physical sea wall model.

difficult and time-consuming. Thus, it can be concluded that those details are not integral to representing and understanding this landscape with respect to the software and imagery in use. Perron and Miller support this assumption in stating:

> The physical environment is a complex entity and therefore, it's not possible to explore all aspects involved in its composition. Essential abstraction should deal with the representation of aspects integral to the physical world to extend our understanding of reality.

While this landscape is only conceptual in nature, it does present much of the layered complexity of a physical environment, thus, allowing it to stand as such, for the scope of this thesis.

Computer visualization, ideally, enhances the understanding of a landscape. For this to happen, the landscape must be simplified, in order to be converted to a language that the software can interpret. Formal components must be idealized to their underlying ele-

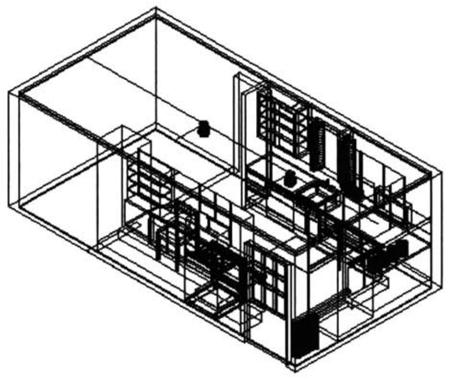
Fig. 4.13 - Wireframe model

ments, (platonic solids, NURBS, etc.) before being manipulated into more complex geometries. Thus, necessity mandates the inclusion of this design analysis step as part of an ideal model for executing the CV process.

4.6 Modeling

The most critical step in CV is, by far, modeling. Not only does it directly influence image creation, the process is a powerful visualization tool in and of itself. From a wire-frame model, a user can glean a wealth of information by rotating and manipulating it in real-time(see Figs. 4.13 and 4.14). In this way, modeling possesses the dual attributes of drawing and physical modeling. Gianni states:

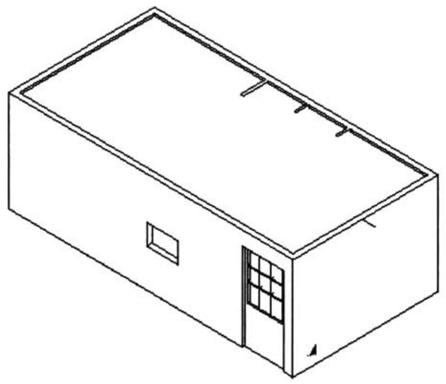
Fig. 4.14 - Hidden line wireframe model

In creating a computer model, then, one is acutely aware of interacting with something 3-dimensional, yet, in the absence of a tactile experience of the model, one's experience is still limited to a 2-dimensional projected views. Thus the computer model always exists for the viewer somewhere between the second and third dimension.

During construction of the computational model, much can be learned about the landscape being represented. In the case example, fundamental issues are addressed in the beginning steps. Before the actual modeling process can continue, the landscape's scale must be established. This becomes especially important with a conceptual project of this nature because no measured drawings were completed for its realization. Scale, then, becomes a basic component of analysis. Without this, there is no way to determine the human relationship to the landscape. Perron and Miller support this by saying:

> The physical world is composed of various layers of reality dependent upon scale for its perception and understanding. Spatial cognition exists relative to the scale of the environment at which the participant interacts.

Thus, a landscape without scale relationships becomes impossible to inhabit, thereby confounding the unique capabilities of this medium to assist visualization.

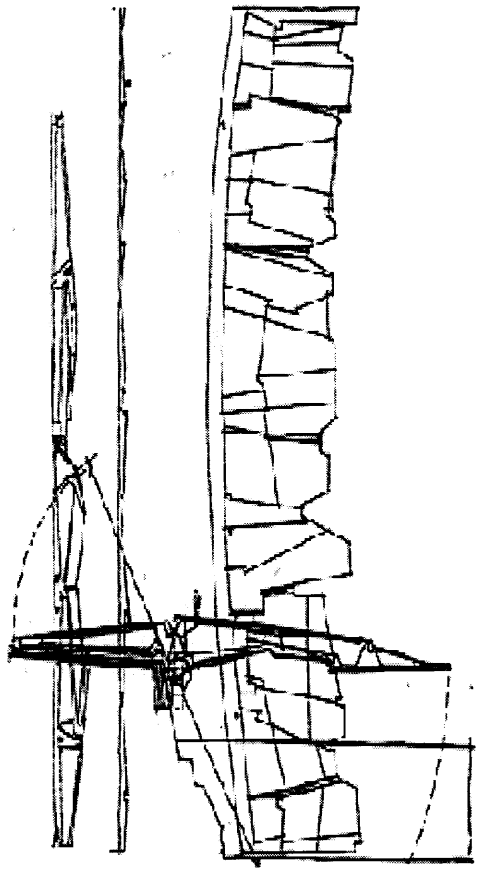

Fig. 4.15 - Plan and sections of sea wall with human figure.

With the sea wall, the architect sketched architectural plan and section views in relation to human figures (see Fig. 4.15). The relationship between the wall and the figure helps to calculate the relative size of the wall, if the height of the figure is approximated and compared to the height/length of the wall section. This knowledge allows an equivalent computation unit of measure to be determined. But, exact measurements aren't really necessary. In a computational environment, the user controls the views on the screen by adjusting the camera's position and lens. An object that is 1 foot in length can, theoretically, appear to be 100 feet, by placing the camera closer to the object and adjusting the focal length, which is directly proportional to the magnification of a scene (Kerlow 1994). Therefore, true measurements are inapplicable and of little use.

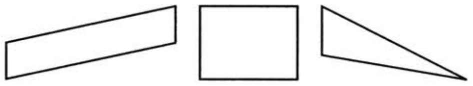

Fig. 4.16 - Polygon form of sea wall section

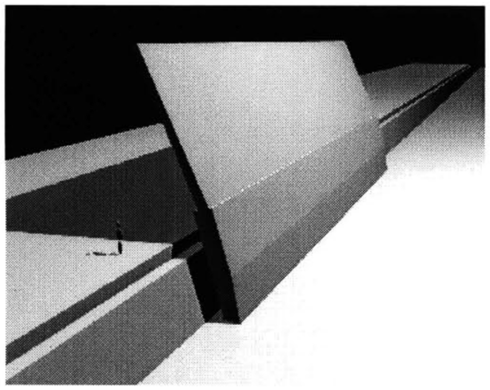

Fig. 4.17 - Ideal wall section

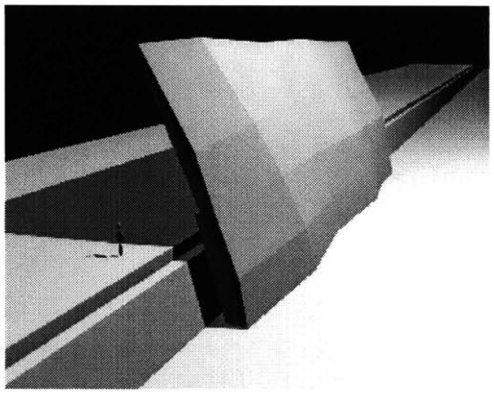

Fig. 4.18 - Wall section with grid

according to the sketches. In analyzing the section, the form of the wall can be idealized to three distinct polygons (see 4.16). By extruding these polygons along a straight path, an idealized 3-dimensional wall section is constructed (see Fig. 4.17). This relates back to the previously discussed necessity of establishing the integral, simplified qualities of a landscape and constructing those first in a computational environment. Details can be developed later in the process.

After construction of the tiled surface, which was simply an irregular grid whose intersections were manipulated in 3d space (see Fig. 4.18) overlaid on the original wall section, the edge conditions of the wall sections could be addressed. The irregular edges evident in the sketches and models (see Figs. 4.19 and 4.20), while idiosyncratic in their representation, were deemed essential to the conceptual understanding of the landscape. Multiple methods could be used to delineate these conditions. To determine which method would be the most efficient and compatible with the constraints of the software, I conducted an experiment with two separate modeling trials. The goal of these trials was to measure file sizes and rendering times for one wall section that had been modeled nearly identically with two separate, distinct modeling processes.

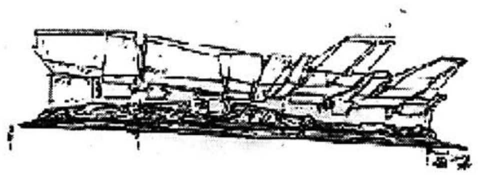
Fig. 4.19 - Sketch of wall edges

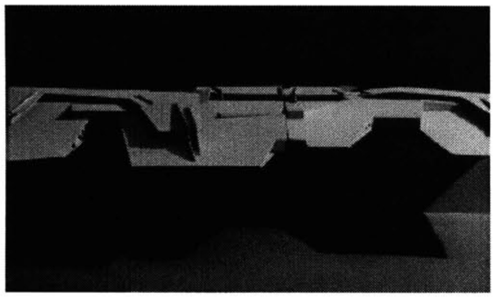
Fig. 4.20 - Model of wall edges

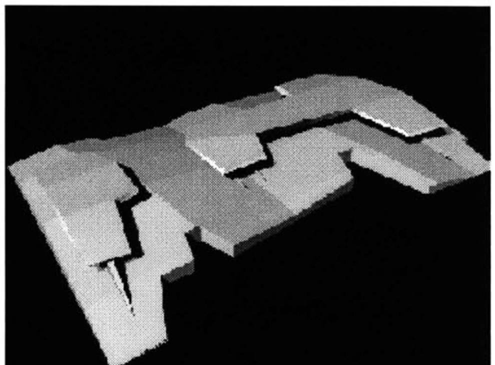
Fig. 4.21 - Model using trimmed surfaces

Trial 1: Trimmed Surfaces

The first method of construction involved trimming, a logical operation used to create models by [conceptually] subtracting shapes from [other] shapes (Kerlow 1993). By trimming away a section shaped like the edge condition from the original wall, a carved edge on the original wall section was achieved (see Fig. 4.21). This proved to be relatively straight-forward and fast to accomplish.

Trial 2: Patched surfaces

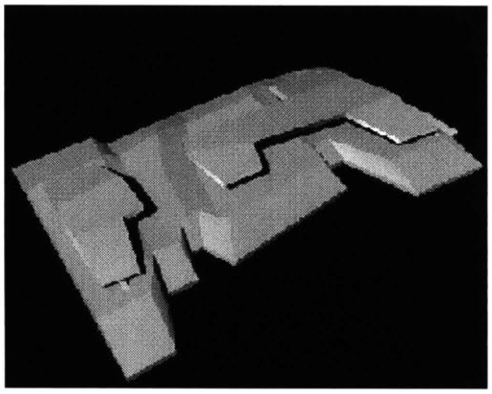
Fig. 4.22 - Model using patched surfaces

The second method used entailed patching, a process that connects 3d lines and curves (splines) with gridded surfaces. I constructed the second model by "tracing" over the edges of the first model with splines and connecting these splines with patches. This method was slightly more time-consuming in its execution, but did represent nearly an identical replica of the first model(see Fig. 4.22).

Results

Upon rendering both models, the differences between the two process became readily apparent. The differences were due to the way in which the software interprets the model based on the modeling function used. The trimming process is not truly subtractive in nature. Once the user specifies the area to be trimmed away, the software does not eliminate the model data, it simply hides it from the view of the user. No data has been removed. Because the patched surface was

	Simple	Complex
wire	189 KB 29%	267 KB
sdl	335 KB 14%	390 KB
render time	4:03 mins 23.5%	5:18 mins

% savings

Fig. 4.23 - Diagram of file size, render time, and % savings.

constructed without the data from the original wall that was trimmed away, the model file contained much less model data. Hence, when both model files were examined and rendered, the patched model provided significant savings in file size and render time (see Fig. 4.23). These findings are advocated by Kerlow:

> In general, models that were built properly render quicker than models that were built clumsily... it is not uncommon to have to return to the modeling stage, fix the modeling problems and then return to the rendering stage with a proper model file.

Additionally, it was noticed that interaction with the trimmed model within the software interface slowed considerably, due in part to increased file size. In conclusion, these results confirm that while trimming may be faster in it's execution, it can slow down rendering substantially and increase file sizes, both of which are undesirable for manipulation and rendering of the model. This reinforces the importance of design analysis and software constraint assessment, prior to

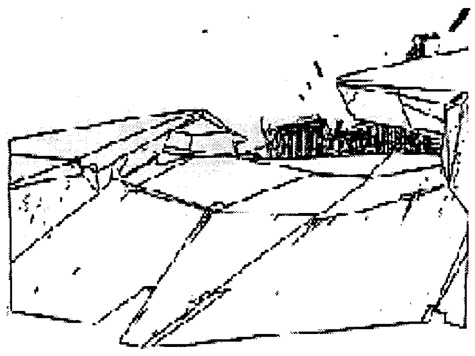
Fig. 4.24 - Sketch of sea wall shading element

engaging in visualizing a landscape with computational methods.

Construction of the wall shading elements and the remaining wall sections was possible after the previous experiment. As discussed in the design analysis, the shading elements were found to be highly idiosyncratic in their depiction and varied greatly not just between the models and the sketches, but also between individual wall sections (see Fig. 4.24). The wall sections were found to possess these same qualities. However, by the examining the sea wall as a whole, it was found that while each wall section was highly individual in its design, an overriding archetype for the design of the elements was discernible (see Fig. 4.25). Using this archetype, I constructed a series of elements which would serve as a basis for building the wall in its entirety (see Fig. 4.26).

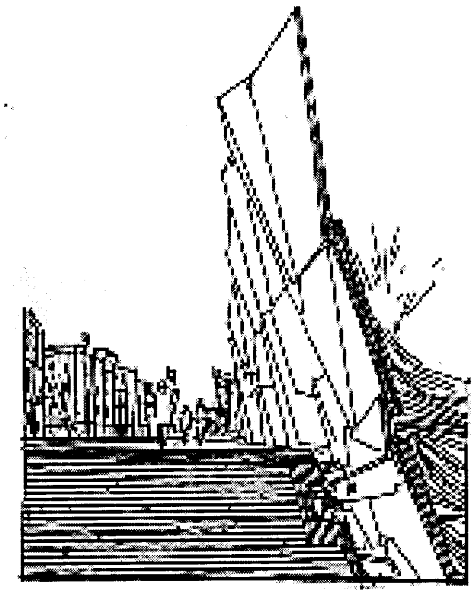
Fig. 4.25 - Sketch of entire sea wall

Computational tools provide an easy method

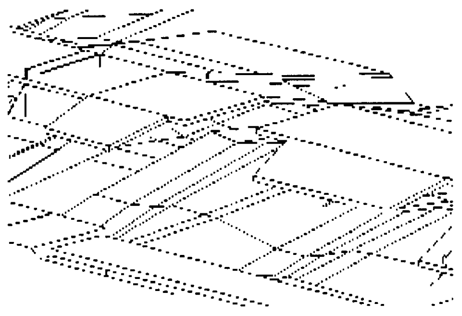

Fig. 4.26 - Image of individual shading element

for replication and transformation (translation, rotation, and scaling) of modeled objects. For this case, with a small set of elements, through this method I was able to generate a much larger set of elements that were sufficiently dissimilar, without sacrificing the great deal of time needed to construct each one individually. Some argue that this approach to using computational tools is detrimental to advancing the potential of the medium. Smulevich states:

> Those who are seriously incorporating CAD into their design processes are generally doing so by translating traditional architectural compositional strategies into CAD based operational terms ... notions such as generating 3d components that define "design kits" for repetitive, industrial like assemblies only perpetuate a methodology ... that predates our acquisition of alternate design environments. We must question their validity...

In my research, I have found that because computational tools do, indeed excel at repetitive tasks, then these functions should be included when analyzing the tools' potential uses. In this case, it proved to be highly

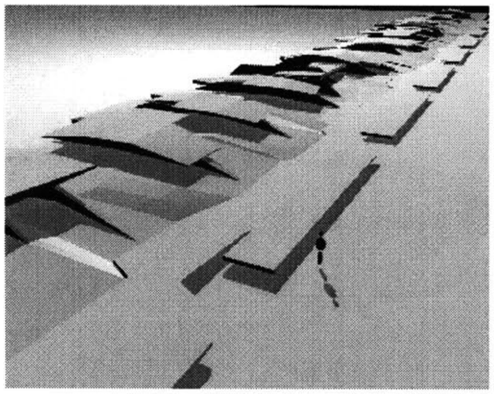

Fig. 4.27 - Modeled entire sea wall

efficient to make mass alterations of elements quickly. How these elements were incorporated into the design scheme was the task that could not be assigned to the software. The results of this use of computation are scarcely recognizable when the landscape is viewed in its entirety (see Figs. 4.27 and 4.28). For this reason, the work presented here disputes the validity of Smulevich's argument.

4.7 Lighting

Lighting is the crucial supplement to modeling, because, quite simply, without it, output imagery would not be visible. Lighting exists as an excellent resource for analysis of landscapes by illuminating its modeled intricacies, contrasts, and position in virtual space. Computation provides multiple methods for simulating natural, artificial and non-existant lighting schemes. Each process is vastly different in its effect nd execution. For a forthcoming book by Steven Oles and

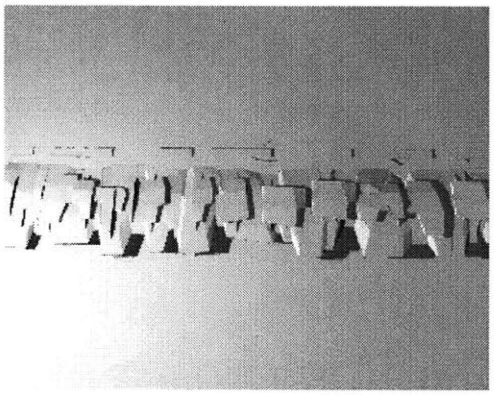

Fig. 4.28 - Modeled entire sea wall

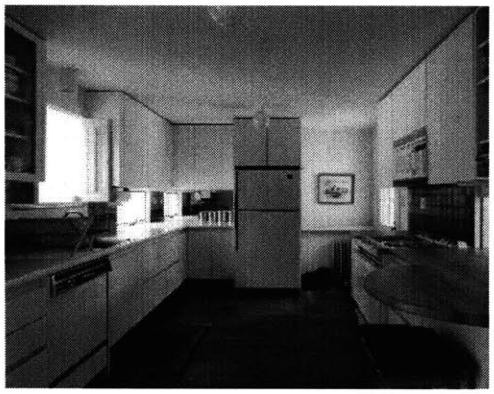
Fig. 4.29 - Photo of a kitchen

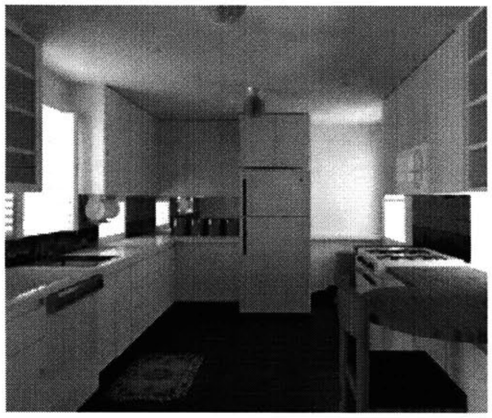
Fig. 4.30 - Radiance rendering of the kitchen

William Mitchell, a photo rendition of an architectural space was needed for comparison to an existing space (see Fig. 4.29). Using Radiance (described previously), one final still image was created with subtle, natural lighting effects (see Fig. 4.30). To achieve this level of quality, many parameters that control various lighting phenomena were adjusted to near their maximum limits. The end result was an image that took 216 hours to render. For the scope of that particular project, that amount of time for one image was allowable. However, in lighting the case example being analyzed here, much shorter render times would be necessary and the resultant imagery would serve a different purpose.

Until this stage, in all previous renderings, a basic spotlight was used to highlight modeling details. The placement of this light was not based on any previously gathered data and was not influential in any particular way.

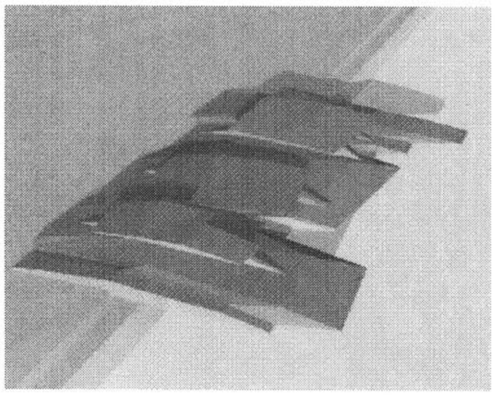
Fig. 4.31 - Jun. 20, 7am. image

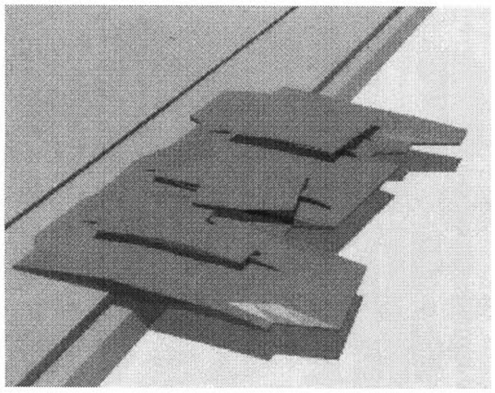
Fig. 4.32 - Jun. 20, 12pm. image

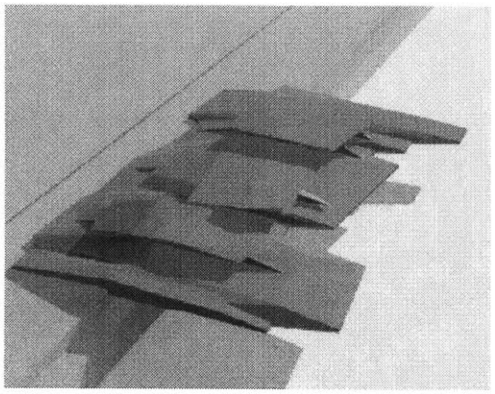
Fig. 4.33 - Jun 20, 5pm. image

Because the site conditions are important to the understanding of the landscape, I chose to rely on Radiance for gathering site-related lighting data.

The proposed sea wall landscape would be sited along the northern coast of Havana, Cuba. By inputting the latitude (23.07o) and longitude (82.25o) into Radiance, renderings were generated that possessed accurate lighting data relative to those global coordinates. In addition, the landscape was rendered over various dates and times with constant atmospheric conditions (see Figs. 4.31 - 4.36). This allowed for a clearer description of the quality of light and shadow within the landscape. With careful observation, subtle differences in the intensity of light, as it occurs in the physical world can be seen. This level of accuracy and complexity was useful for designing an analogous scenario within Alias.

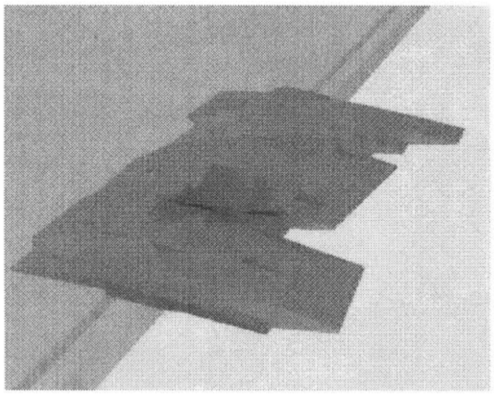
Fig. 4.34 - Dec. 20, 7am. image

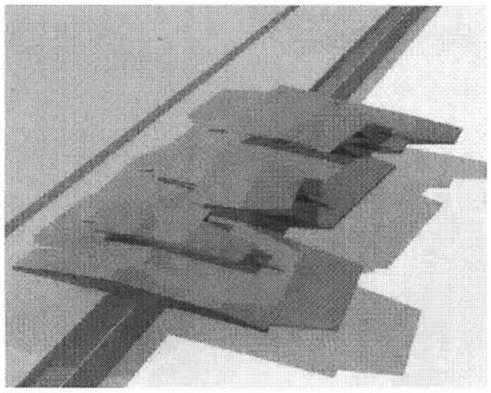
Fig. 4.35 - Dec. 20, 12pm. image

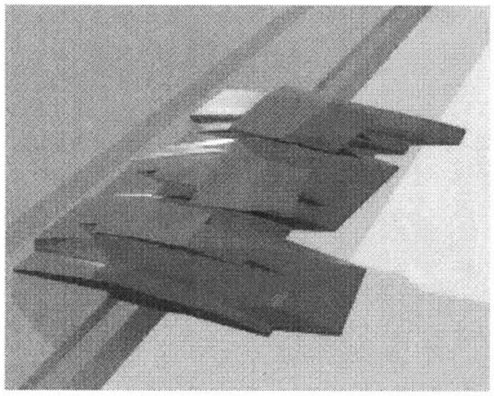
Fig. 4.36 - Dec. 20, 5pm. image

Alias provides an versatile interface for creation of artificial light sources with control over such qualities as intensity, color and falloff, but offers very little control over the physics of natural light, found in Radiance. For instance, within Alias, the user is given no control over a factor called ambient bounce. Within an enclosed, physical environment a ray of light will reflect off all the surfaces in the space and will continue to do so to infinity. With each bounce, the ray will decrease in intensity as some percentage of the ray is absorbed by the material that reflected it. While the computer obviously can not handle infinite calculations, Radiance gives you control over the number of bounces a ray of light will take, thereby allowing a finer degree of control over the quality of light in a space. Without this level of control, other methods must be used to compensate. The challenge in this case, was to develop a means of natural light with artificial sources. Using information gathered

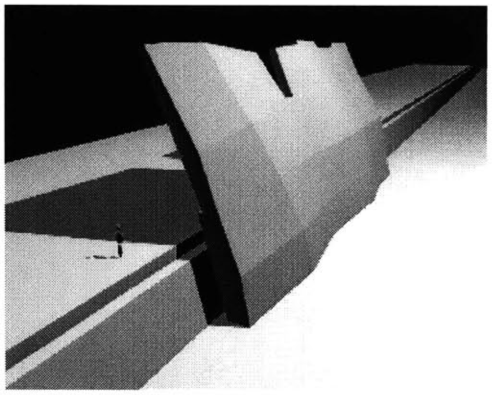
Fig. 4.37 - Image with 1 directional light source

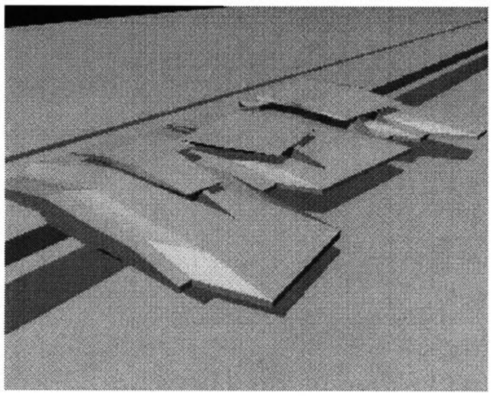
Fig. 4.38 - Image with multiple light sources

from Radiance, I was able to approximate the computational "sun"'s direction and intensity using a directional light source, which in Alias emulates very distant point sources, like the real sun. However, in a physical environment, light is scattered through the atmosphere, thus complicating the phenomenon. WIthout this scattering in Alias, shadowed areas in scenes were very dark(see Fig. 4.37). To compensate for this, other ambient and non-shadow casting point sources were included (see Fig. 4.38). The results of this study were that while Alias offers a high degree of control over artificial sources, it is woefully inadequate for simulating physical lighting. In contrast, Radiance was well-suited for this task, but could not handle the many other complex computational functions like modeling or animation without a great amount of time and difficulty. Therefore, careful planning and evaluation of the types of imagery needed is required before undertaking CV tasks involving lighting.

4.8 Rendering

Rendering is the process by which images are created. It involves calculating a projection (perspective) of the model, according to camera position and focal length. Visible surfaces within the newly formed scene are determined and finally shaded (Mitchell 1995). The shading entails some type of light casting algorithm. Alias supports two algorithms for generating final imagery: ray-tracing and ray-casting.

Ray-tracing determines the visibility of surfaces by tracing rays of light backwards from the camera, through the scene and back to its original source. This tracing occurs for each pixel in an image (Foley 1994). Ray-casting uses the same general algorithm with one major difference. Ray-traced light, upon striking a surface, is divided into numerous component light rays (reflection, refraction,

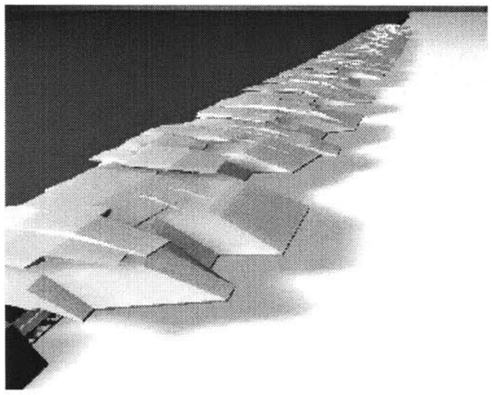
Fig. 4.39 - Ray-cast image
Camera 1

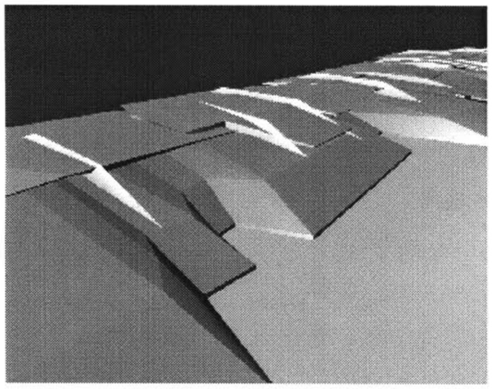
Fig. 4.40 - Ray-cast Image
Camera 2

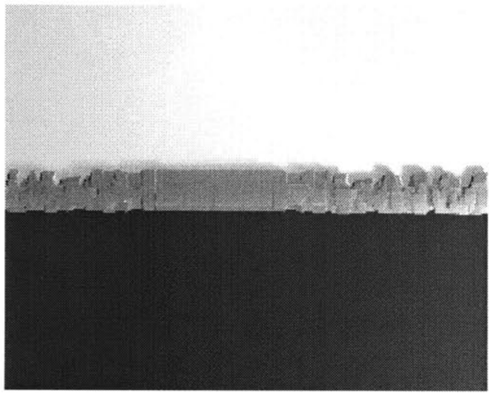
Fig. 4.41 - Ray-cast image
Camera 3

shadow, and transmitted) which are, in turn, followed along their respective paths until their termination. Ray-casting does not do this, so it can not handle reflective or refractive surfaces and generally does not provide the shadow quality of ray-tracing. In an effort to ascertain the relative benefits of both methods, I conducted a series of trial renderings. One model was rendered with both methods using multiple camera views (see Figs. 4.39 - 4.44). The intent of these trials was to compare image quality vs. image render speed in an effort to determine the trade-offs. A secondary goal was to determine how camera position affects render speed.

The results of these trials showed that ray-tracing provides crisper shadows and greater illumination of details at the cost of increased rendering times. Ray-casting provided less exact shadows and less detail, but faster rendering times. Depending on the

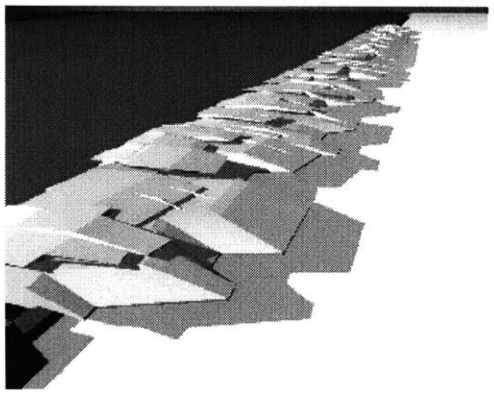
Fig. 4.42 - Ray-traced image
Camera 1

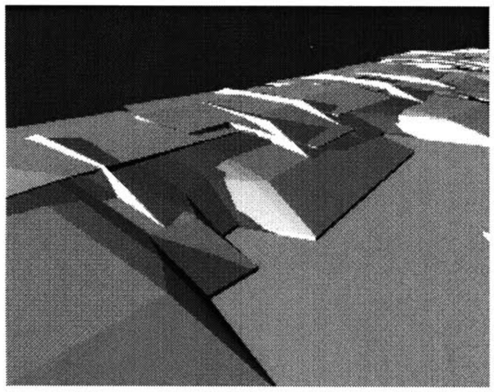
Fig. 4.43 - Ray-traced image
Camera 2

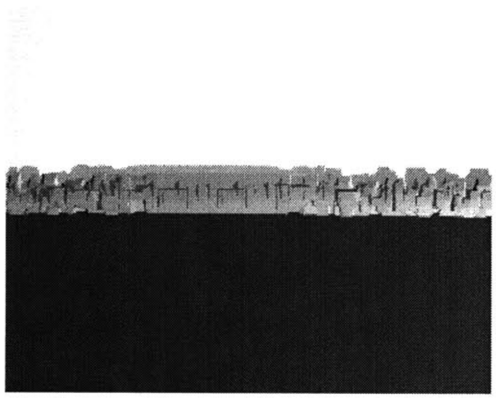
Fig. 4.44 - Ray-traced image
Camera 3

type of imagery to be created, use of either of these methods will vary. If accuracy is essential, then ray-tracing is necessary. If scenes don't require such functions as reflection or refraction or accurate shadows, then ray-casting could be sufficient. It should be noted that both algorithms took much longer to render scenes with more geometry visible, i.e. long and wide angle shots have more surfaces visible to the camera. This type of study benefits the development of a CV paradigm by providing knowledge of various methods and outcomes, which allow for more predictability, and thus, more control over the appearance of images.

4.9 Animation

The final, yet optional, stage of CV is animation. While animation is not necessary for all uses of CV, it can provide invaluable insight in visualizing an architectural space. Animation involves a simulation of movement or some change in environment over a period of

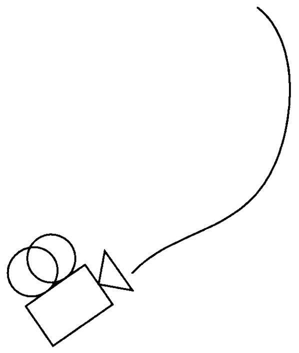

Fig. 4.45 - Animation motion path

time. It is commonly used to visualize spaces in a continuous manner analogous to how people visually perceive real physical spaces. This is achieved through two distinct methods: camera animation and object animation.

Camera animation is most often utilized in the development of architectural walk-throughs. A motion path, which governs the movement of the camera through a space, is created within the environment to be animated. The camera then follows this path from its start to the end, viewing the architectural space around the path (see Fig. 4.45). Each subtle movement of the camera is characterized with an individual frame, just as in film or traditional animation. When these individual frames are viewed at high speed (24 frames per second for film and 30 frames per second for video), the illusion of movement is created. This type of animation is often called a walk-through or fly-by

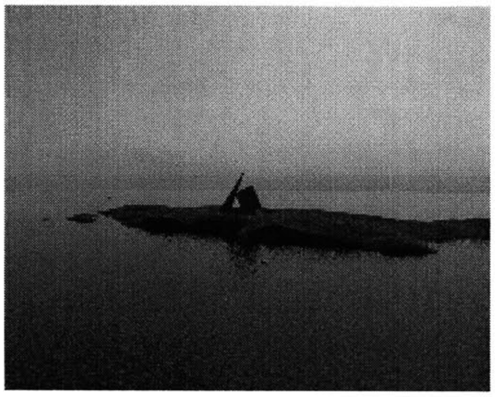

Fig. 4.46 - Middle Passage Project animation still

because it simulates an experience of a human moving through the modeled environment.

The Middle Passage Project, produced by Larry Sass and the author, is an example of the use and effectiveness of this method of animation. An unbuilt monument was designed by Donald Stull and the animation was created to give the experience of people visiting the monument after construction (see Figs. 4.46 - 4.47). The production team applied Radiance software to generate the imagery and used a specially written program to control camera motion. This method was ideal and efficient for this type of task, since the architecture did not include any other moving objects or special effects, like changing atmospheric conditions or moving water. To represent these kinds of effects, object animation is better suited and serves as an excellent complement to camera animation, particularly for studying the case

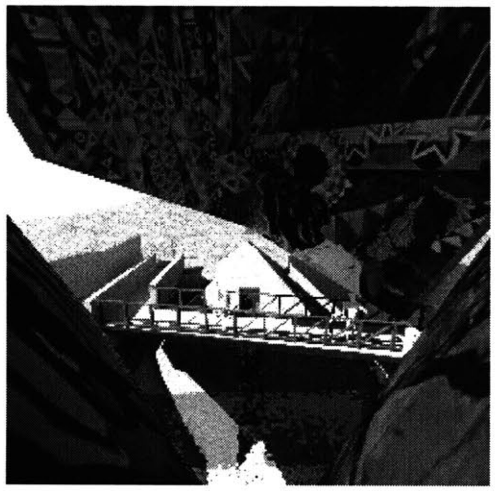

Fig. 4.47 - Middle Passage Project animation still

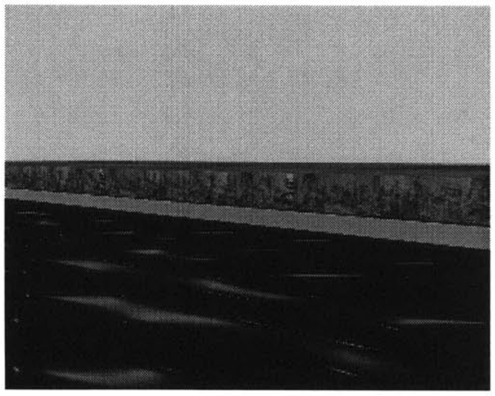
Fig. 4.48 - Animation image
Frame 1

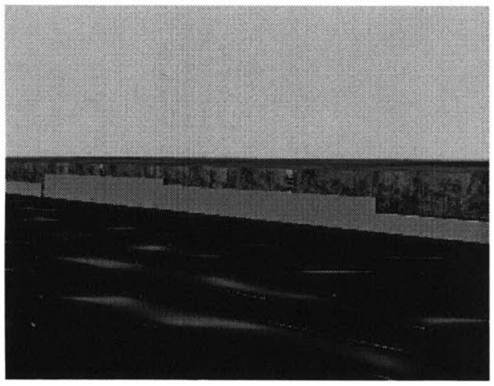
Fig. 4.49 - Animation Image
Frame 75

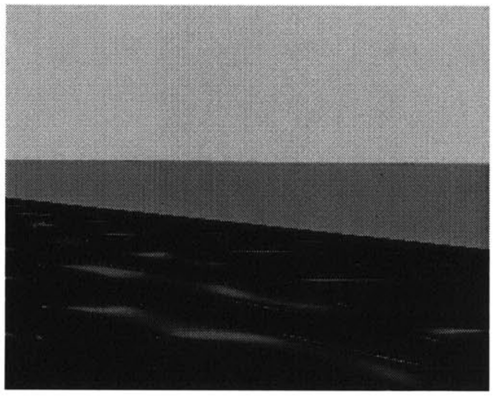
Fig. 4.50 - Animation Image
Frame 150

project presented here.

An animation was created that metaphorically represented the initial formation of the landscape out of the pre-existing forms (see Figs. 4.48 - 4.50). To accomplish this involves setting what are called key frames, special frames during the course of an animation that govern important positions of an object at a specific time, like the starting and ending position of a moving object. Once the keyframes are established, the computer interpolates the anticipated position of the object between the key frames. For example, a key frame was set before the landscape's formation. This would be frame 1. At frame 150, the wall has risen completely from the water, so another key frame is set at this point. The software then calculates the position of the wall at each frame between the two keyframes. When the animation is then viewed, the wall seems to rise from the water in the span of 150 frames, or 5 seconds.

This method varies from the one used to create the Middle Passage Project. Because the software written to support animation within Radiance did not include object animation, key frames on the nature discussed previously could not be used. However, start/end points of the motion path for the camera could be specified, in addition to changes in camera views. So, the software did conduct a frame interpolation, but it was just restricted to the motion of the camera. In essence, the processes are similar but the results are different. With Alias, it is possible to combine camera and object animation techniques together, to create a more visually complex animation. This ability satisfies dual requirements of simulating a person moving through the space and the object interacting with the environment or person. To begin computer animation, as with traditional cel animation, a storyboard is first developed. This documents key aspects of

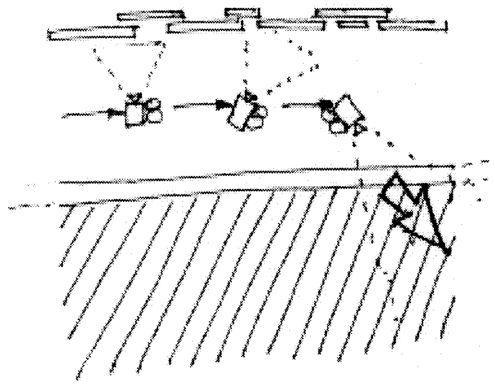
Fig. 4.51 - Storyboard image

the animation in a still form through a series of typically hand drawn pictures, with text describing important parts of the scene, such as lighting, camera changes, movement, mood, etc. (see Fig. 4.51). Any changes in the scene throughout the course of the animation process are depicted in the storyboard. This allows entire scenes to be pre-visualized and planned which saves valuable time in completing the final animation by providing a framework of stills to the build from. For this case, a mixture of hand drawn and computer rendered images were used to develop a storyboard. Additionally, the use of computer rendered images provided another unanticipated benefit. As discussed previously, rendered scenes with wide angle camera lens and long views of the environment took substantially more time to render than scenes with tighter camera views, due to the amount of geometry visible by the camera. By comparing these two types of scenes for the storyboard, the author could deter-

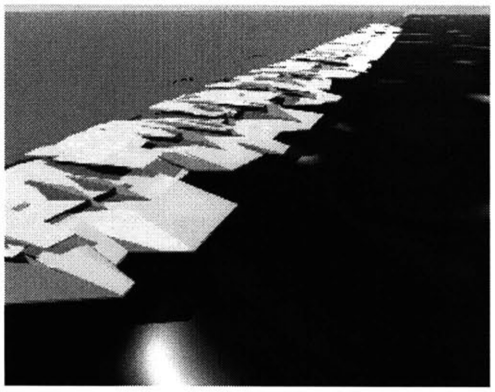
Fig. 4.52 - Computer image from storyboard

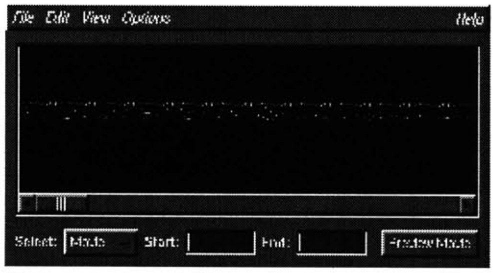

Fig. 4.53 - Animated "storyboard" SGI Moviemaker

mine how often to use wide angle shots by letting the render times limit the frequency of use (see Fig. 4.52). In other words, because rendering animations is already a time consuming and labor intensive process for the computer, using the storyboard in this method prevented rendering too many scenes that were time inefficient and excessive. After the scene is storyboarded, most CV tools allow the user to render test animations. Typically, these tests are low-quality, small scale (one quarter to one half the size of the final images) animations that provide valuable information on the specific qualities of the animation while being much faster to render. These miniature animations can be assembled to form movies (see Fig. 4.53)of various formats (Quicktime, JPEG, etc.) which can be examined repeatedly to ascertain an element's actual appearance in relation to the desired appearance. If they do not coincide, changes can be made in the CV tool's environment and the animation can be quickly re-rendered until the desired effect is achieved. This method of animating, reviewing, and re-animating is another, more advanced method of storyboarding. It is unique to CV because of the high speed of the entire process. For example, in this case, the author experimented with animating the effect of water being affected by wind. The scene was test-rendered 8 separate times in a 3 hour time span, until the exact effect was obtained. If the scene had not been test-rendered, the entire process of rendering the final scene would have taken the computer 40 hours. Savings such as this are key to developing not only high quality, informative animations but essential to conserving time and resources, both constant concerns when using CV.

Chapter 5

Conclusion

5.1 Objective

The intention of this research was to examine and refine the intricacies of the computer visualization process in an effort to develop a better model for its implementation. By advancing a conceptual landscape from conventional forms of representation to a computational form, my intent was to document the abstraction process and resolve some the common problems associated with it. The focus was intentionally shifted away from the analysis of photorealistic imagery to the process, where I believe the most potential for exploration remains.

5.2 Results: Design Analysis

Critical analysis of the landscapes to be abstracted to a computational environment is a necessary and often overlooked process.

Due to the complexity of the physical environment, it is impossible to represent each layer and all the physical relationships present. For this reason, simplification of the landscape to its integral components becomes essential. In this case, which already existed as an abstraction of a physical environment, clarification of minute details, while considering the software's constraints, allowed me to develop a quick and easy method for representing the primary design issues without getting mired in the intricate functions of the software.

5.3 Results: Modeling

In the modeling analysis, different modeling methods for representing architectural landscapes were examined for their efficiency and utility. Two methods for constructing surface forms were studied: trimming and patching. It was found that trimming, while intuitively analogous to sculpting an object into a particular form, was actually quite

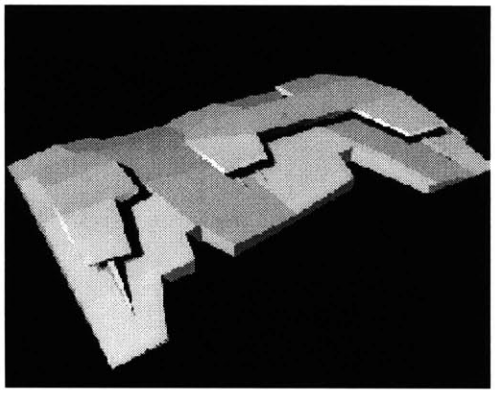
Fig. 5.2 - Image of trimmed surface

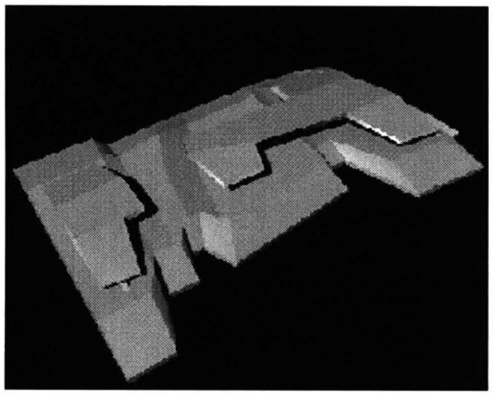
Fig. 5.3 - Image of patched surface

costly for the software to handle. Trimming increased the size of model files; thereby slowing the manipulation of the model in the software environment and substantially increasing rendering times. The second method, patching, the equivalent of connecting a surface between two edges, was slower in its initial execution than trimming, but it more than compensated for this by providing significant savings in file sizes and rendering times. Although both models constructed were nearly visually identical (see Figs. 5.2 and 5.3), this study proved that while there are many methods for modeling, some are innately more efficient and should be used more than others.

5.4 Results: Lighting

During the lighting design, studies were conducted to simulate the effects of natural lighting within a computational environment. It was realized that while Alias (the primary working environment) was proficient at simulating artificial light, specialized tools, like Radiance, were needed for natural lighting (see Fig. 5.4). This was proven by generating multiple lighting scenarios within Radiance that accounted for global position, date and time and attempting to duplicate the subtle lighting effects within Alias, which could not be accomplished in any reasonable time frame (see Fig. 5.5). However, by using the Radiance results as a reference, other lighting effects could be effectively designed within Alias. This complementary relationship was found to highly useful for the creation of lighting environments by combining the accuracy of Radiance and the flexibility of Alias.

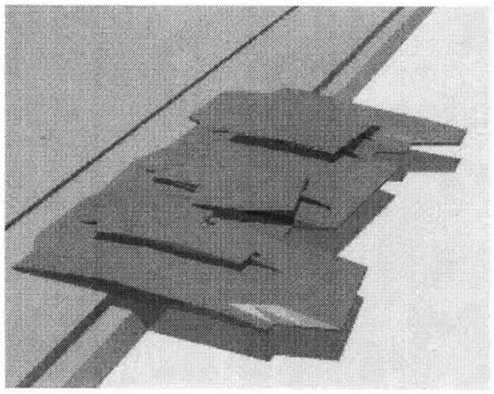
Fig. 5.4 - Radiance image

5.5 Results: Rendering

During the rendering analysis, different ray-tracing methods were compared for their speed and image quality. A set of identical scenes were rendered with two similar but distinct rendering methods: ray-casting and ray-tracing (see Figs. 5.6 and 5.7). While

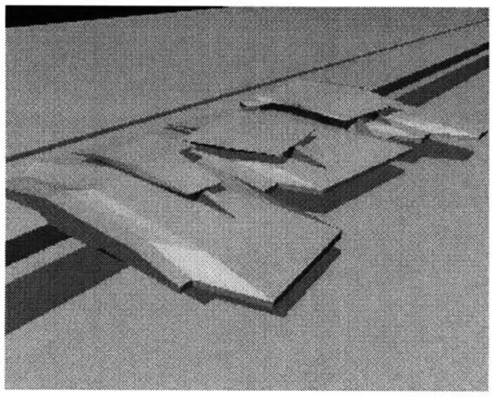
Fig. 5.5 - Alias Image

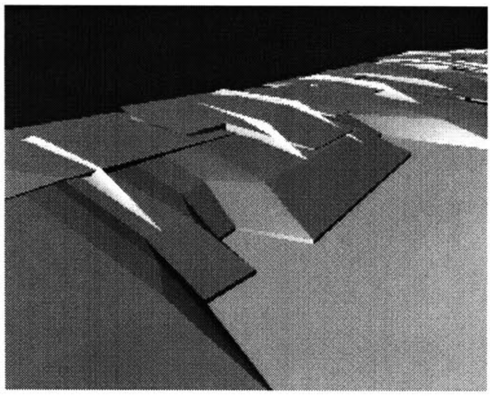
Fig. 5.6 - Ray-cast image

ray-casting, a simpler form of ray-tracing that does not handle reflections or refractions, is the faster method, the image quality was noticeably lower. Ray-tracing gives much crisper shadows and lighting details at a cost of slower rendering. Another result of these render trials that was not as obvious, was that scenes composed of wide or long angle shots, took a dramatically longer amount of time to render. Upon further inspection, this is logical because in those scenes, more geometry is visible to the camera, so there is more information to render. Selection of rendering method was still found to be highly user-dependent based on project demands of quality vs. speed.

5.6 Results: Animation

For the animation phase, the development of test animation as storyboard proved to be useful in pre-visualizing animated sequences . While static storyboards were created as a basic framework, the animated storyboards, which consisted of miniaturized, low-quality

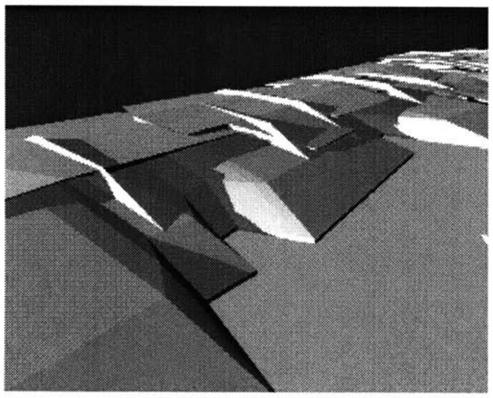
Fig. 5.7 - Ray-traced image

animations, assisted greatly by allowing me to see the motion being simulated as it would appear in final animations, without the cost of timely and CPU intensive rendering of full-size, high-quality images. By offering this ability, CV distinguishes itself as a unique medium by giving the user a high degree of flexibility and power to better understand the uses of animation to visualize architecture.

5.7 Justification of methods

These experiments were important to understanding not only how the individual processes work, but also how they are linked to each other. For instance, the results from the modeling study, greatly affect how efficiently an image is rendered, which in turn, would also affect how fast an animation is rendered. This cyclical relationship also works in reverse. Rendering can illuminate discrepancies or mistakes present in a model, thereby alerting the user to return to the

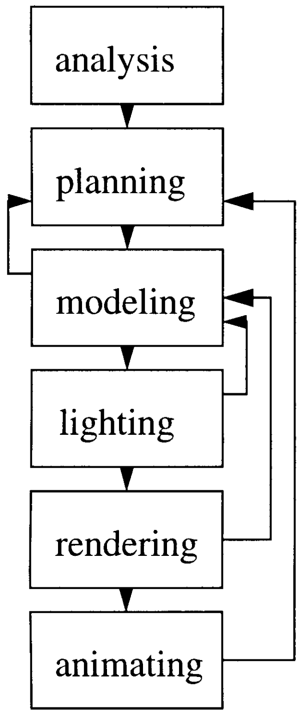

Fig. 5.8 - Diagram of cyclical nature of CV process

model to adjust or rebuild it. These relationships between individual processes forms a web of planning and execution that can overlap and repeat many times before a user completes the entire CV process (see Fig. 5.9). The importance of this occurrence to the user is that individual processes can not be completed in a vacuum. For instance, it is not possible to complete all of the modeling tasks, without lighting and rendering the model during its construction. Likewise, a model can not be rendered efficiently without understanding the nature of the construction of the model and the inherent advantages and disadvantages. This complex inter-relationship stresses the importance of planning before undertaking a CV project. By fully understanding the techniques and the capabilities of the tools being used as well as tools' ability to represent the particular architectural project, a plan can be developed which maximizes the potential of the tools while minimizing the time required for execution. This understanding adds predictability to the entire CV process by reducing the amount of time the user spends troubleshooting unanticipated problems, which often arise when novice users employ CV tools, and increasing the time spent on using the process to understand the architecture, which is the primary purpose of CV.

5.8 Computation Issues

The analysis of the CV process also shed light on many of the problems with current tools. First, a major problem exists in the necessity of multiple platforms to complete certain tasks, like lighting. Constantly switching back and forth between platforms proved to be time consuming. In addition, for users not well versed in the use of different software packages, the transition between adapting and learning multiple interfaces can deter the use of CV completely. A related problem to the issue of multiple platforms is the transferral of data between them. The author had a great deal of difficulty in figuring out how to transfer model data from one program to another (see Fig. 5.10). While supposedly “standard” file formats exist for such transferral, the author found that the method in which each program interpreted these file formats was very different. Thus, in this case, Autocad was needed as an intermediate platform to transfer files from Alias to Radiance. If not for the need of this file translation, Autocad would not have been necessary at all during the entire CV process. The final major problem in this process is the manipulation of the tools themselves. Some programs like Radiance, while imminently useful, have no user interface worthy of mention. With this software, all commands are type-driven and rely heavily on the keyboard for manipulation of data, models and imagery. This limitation severely hampers visualizing a model in any intuitive or natural manner. While Alias' interface was much more user-friendly, I still

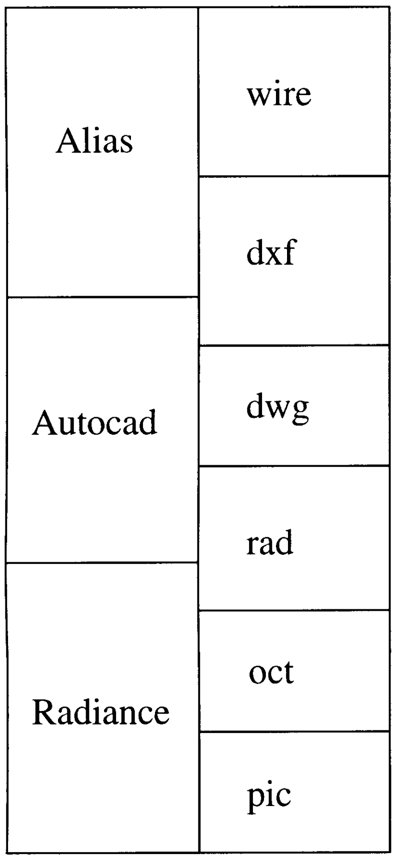

Fig. 5.10 - Diagram of file translation process

found that its flexibility is also its largest obstacle. There are simply too many commands, nested within other commands, which are, in turn, nested within text menus. Even after finding the command I thought might be of use, determining what that command did upon execution involved constant referrals to the 1500+ page manual. Before this tool can really be used for understanding architecture, the interface would need to be completely rebuilt to rely on intuitive visual data, like icons, as opposed to nested text menus.

5.9 Significance and Future Directions

The importance of this work lies in the clarification of many issues users have with CV technology. I feel that this medium has the potential to revolutionize the way architects visualize spaces, not unlike the impact that drawing made on architecture during The Renaissance. Until now, efforts have not been made to understand the process in relation to the specific tools nor develop a useful model for its implementation that take into account the tools' constraints. The increasing demand for the use of CV will inevitably necessitate its further exploration beyond current interests which center on generation of presentational imagery.

Despite the problems of the medium, which will inevitably be addressed as the technology advances, CV is poised to make a considerable impact in the realm of representing architecture. It combines the clarity of 2-dimensional drawing with the utility of 3-dimensional modeling. In addition, it offers the user the ability to intuitively abstract and manipulate forms in a much quicker manner than most traditional methods. Further, through simulation of materials and animation, CV allows the user to perceive and comprehend computational landscapes in a visual manner that is closest to the way our eyes and brain understand physical forms. If seen as the process for understanding, CV will eventually become an established method, alongside the other conventional methods of sketching and modeling, for depicting and understanding architecture.

Bibliography

(Foley 1994) Foley, J., van Dam, A., Feiner, S., Hughes, J., Phillips, R., Introduction to Computer Graphics, Addison-Wesley, 1994

(Gianni 1991) Gianni, B., "Building, Seeing, Thinking: The Use of the Computer in the investigation of Visual Logic", Reality and Virtual Reality, ACADIA 1991

(Kerlow 1996) Kerlow, I., The Art of 3-D Computer Animation and Imaging, Van Nostrand Reinhold, 1996

(Lewell 1985) Lewell, J. Computer Graphics, Orbis Publishing, 1985

(Mitchell 1995) Mitchell, W., Digital Design Media, Van Nostrand Reinhold, 1995

(Perron and Miller 1991) Perron, R., Miller, D., "Landscape of the Mind", Reality and Virtual Reality, ACADIA 1991

(Smulevich 1993) Smulevich, G., "CAD in the Design Studio: The Discovery of Inhabitation", Education and Practice: The critical interface, ACADIA 1993