Rochester Institute of Technology

A Thesis submitted to the Faculty of the
College of Imaging Arts and Sciences
in candidacy for the degree of
Master of Fine Arts

In All Directions: An Investigation of Multidirectional Camera-centric Rigs in Hypothetical Space.

by Brendan Bond

May 23, 2008

Approvals

Chief Advisor: Dan Deluna, Assistant Professor, Computer Graphics Design

Signature of Chief Advisor *Date*

Associate Advisor: Shaun Foster, Assistant Professor, Computer Graphics Design

Signature of Associate Advisor *Date*

Associate Advisor: Jon Schull, Associate Professor, Information Technology

Signature of Associate Advisor *Date*

School of Design Chairperson: Patti Lachance, Associate Professor, School of Design

Signature of Administrative Chairperson *Date*

Reproduction Granted:

I, Brendan Bond, hereby grant permission to Rochester Institute of Technology to reproduce my thesis documentation in whole or part. Any reproduction will not be for commercial use or profit.

Signature of Author *Date*

Inclusion in the RIT Digital Media Library Electronic Thesis Dissertation (ETD) Archive:

I, Brendan Bond, additionally grant to Rochester Institute of Technology Digital Media Library the non-exclusive license to archive and provide electronic access to my thesis in whole or in part in all forms of media in perpetuity. I understand that my work, in addition to its bibliographic record and abstract, will be available to the worldwide community of scholars and researchers through the RIT DML. I retain all other ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles and books) all or part of this thesis. I am aware that, if appropriate, I have obtained and attached written permission statements from owners of each third party copyrighted matter to be included in my thesis. I certify that the version I submit is the same as that approved by my committee.

Signature of Author *Date*

WITH THANKS

It is at this point that I would like to thank a great number of people for their support - it goes without saying that my family and friends have really helped me through this entire project, and I'd also like to thank my committee (Dan Deluna, Shaun Foster, and Jon Schull) for their guidance and wisdom. Furthermore, I would graciously like to recognize all those who took pictures or video of the thesis show for documentation purposes, including Tim Stephany, who spent the better part of an hour at the show filming.

A special thanks is necessary especially to my father, without whom I'd never have the time nor energy to create the rig built for the thesis show, and such a well crafted one, at that.

Finally, thank you to both of my parents for the patience, love, and support you've given me over the years - It's certainly more than I could ever repay in one lifetime, let alone several. I love you both dearly.

TABLE OF CONTENTS

ABSTRACT

Because the world as we know it is quite often dominated by visual information, most human beings have learned to extract and interpolate data efficiently from what they perceive optically. With the advent of new technologies, however, humans are now presented with challenges of viewing the world in many ways previously deemed conventionally impossible.

The ultimate purpose of this thesis is to explore the possibilities of multiple virtual "cameras" placed in hypothetical space (virtual space that mimics actual space, with three dimensional axis of translation) from a single point of origin, their integration in coordinate space, and the possibility of this unique output as an effective new form of visual communication. Quite simply, the integration of multidirectional, camera-centric rigs in coordinate space presents an interesting facet of three-dimensional space that has yet to be fully realized.

This thesis project, including video and imagery of the final project as well as a digital copy of this paper, is available for viewing at *http://www.brendanbond.com/thesis/index.html* .

INTRODUCTION

The idea of multiple simultaneous perspectives proves to be a concept that has a long history behind it, though it is only of recent that human beings have been able to explore this idea in virtuality. For the purposes of this thesis, virtuality is to be considered any hypothetical three-dimensional space, such as can be constructed in perspective-based computer programs having the ability to create synthetic world spaces with three axis of dimension. The phrase of "multiple perspectives" may also be slightly confusing to the reader, especially if they are viewing the final panoramic output, either on a screen or through a projection system (Refer to Appendix B for Images of one such projection system, which was constructed for the viewing of final output of the camera rig created for this thesis). Prior to this final output, any number of virtual "cameras" are generated. Since each camera is a unique perspective unto itself, it is here that the label of multiple simultaneous perspectives is considered to be accurate.

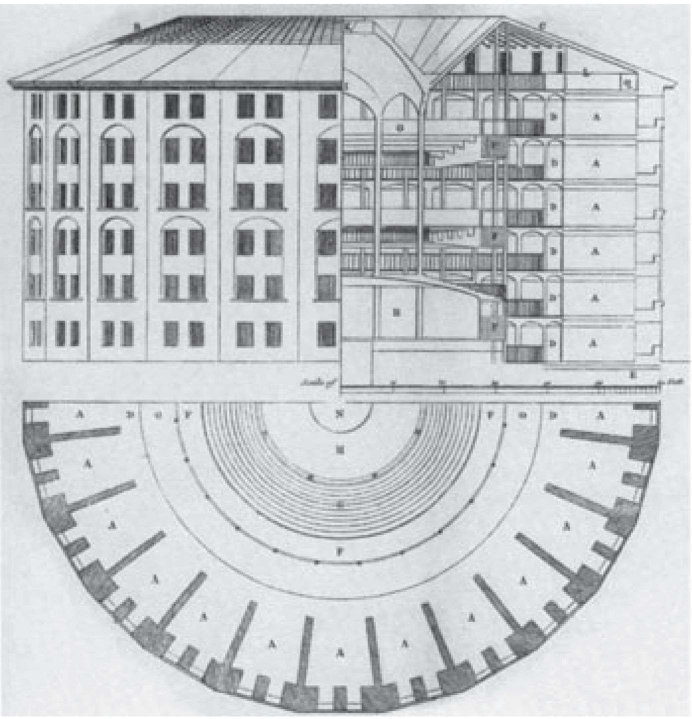


Fig 1.   Original 1791 Blueprint of the Panopticon, by Jeremy Bentham.

The use of computers and digital media for this specific project should not be read as an suggestion that the idea of multiple perspectives was borne with the computer; before the advent of this modern-day technology, many suggestions of having the ability to simultaneously view a number of varied angles had already been proposed. This ability to gather information has been used for many hundreds of years, including (but certainly not limited to) Bentham's Panopticon, a suggested theoretical surveillance technique allowing very few centrally-located guards to watch a much greater prison population effectively and efficiently. The integral aspect of the Panopticon is that a grid of cells are placed around the perimeter of a cylindrically shaped building. A single guard tower sits in the center, where all cells can be viewed by the least amount of watchmen. Interestingly, the guard tower is covered in blinds that allow for one-way viewing of the cells. Consequently, the implication of this pan (all) -optic (seeing) effect is that the guards are not limited to physically watching prisoners; since at any time the guard tower can be occupied or vacant (with no discernable difference to the prisoners themselves), inhabitants of these proposed structures were guided toward the notion of self-governance. This real-life scenario was one such previous development of a number of simultaneous views (being the omnipresent ability of those located within the central guard tower) at any given moment in time (Bentham, 45).

Equally as interesting in regards to multiple coincidental aspects is the idea of cubism of the early 20th century. Cubism portrays a number of simultaneous perspectives, whereas the cubic forms of paint are intended to create a multitude of viewpoints in order to portray the context of the painting's subject matter uniquely. The late Austrian-born art historian E.H. Gombrich explained Cubism as "... a return to what we have called the Egyptian principles, in which an object was drawn from the angle from which its characteristic form came out most clearly" (Gombrich, 432). One painting or drawing could contain many of these such angles — though the subject matter would remain intact, the artist would portray various facets of the object through many varying perspective-based views. Though this draws immediate parallels with the notion of multiple simultaneous perspectives in virtuality, obvious differences are easily inferred. It goes without saying that this multiple-perspective "view"is physically limited in regards to canvas space in one moment of time, whereas technology allows for a digital "canvas" (that being the screen, which likewise is a rectangle of one particular size) to refresh, and ultimately adjust perspective over time.

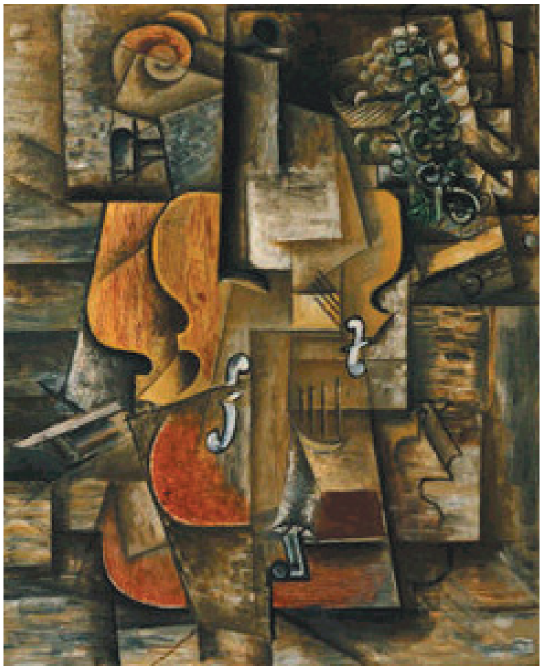


Fig 2. Violin and Grapes [Violons et Raisins], Pablo Picasso, 1912. An exceptional example of the many simultaneous persepectives of Cubist works. Used by permission.

Though the aforementioned perspectives are based in actuality as opposed to virtuality, they were all exceptional precedents for the recreation of this phenomenon in virtual space. After all, much of what we attempt to create in virtual space starts as an idea in actual space (or reality as we know it), which we attempt to understand and mimic. By studying and contemplating what has come before us, we can use this ideation as a logical path to travel towards future notions and ideas. Though the medium is considerably different, it is here where we are given the ability to realize the bounds and constraints of previous systems and further develop them.

As mankind increases in technological ability and know-how, so too do the tools to recreate this phenomenon of multiple perspectives. One such example of a digital predecessor of this project is the many CAVE (a recursive acronym for Cave Automatic Virtual Environment) projects and environments around the world. These environments were first explored in the early 1990's at the University of Chicago, and have since become popular learning and communicative tools around the world (Stevens, 1). Essentially, CAVEs are enclosures with walls made up of projection screens that surround the user - which is where the similarity of multiple perspectives is relevant to the aims of this thesis. Caves, however, are limited by physical resources, such as space available, the number of allocatable projectors, etc. The aim of the virtual rig created in this thesis was to explore the ability to create virtual rigs with no limit on the number of perspective cameras allowed by the user to render for final output, which is something CAVEs (being located solely in a physical environment) are limited to.

In this paper to this point, I've illustrated several examples of previous work regarding multiple perspectives, from the abstraction of the Cubist movement through more recent technological advances. The work this thesis was based upon is the next logical conversation to have, to move from past to present, and discuss the notions of this thesis — and how it relates to (or differs from) previous work.

As an individual who studies both the technical trade and aesthetic of computer graphics, The aim of this thesis was to create a tool that would enable users to explore the notion of multiple perspectives, and then to use this tool to showcase the possibilities of this tool through artistic vision. Unlike previous multiple perspective attempts in our physical world, I am not limited to the same parameters of reality. The ability to create a virtual camera rig that has any number of cameras the user specifies is something that could not be accomplished, since several limitations (cost of multiple physical cameras, the size of said cameras, etc) never enter into the equation at all. That is not to say that this project is not without any limitation, however. Similar to the canvas of an artist, the works of art that are generated on a computer screen are still (at least currently) limited to a format defined by the size and resolution of screens being used to view the art. The confines of this virtual "canvas" can still act as a limitation to the experience of viewers and users alike.

In respect to current systems of virtual multiple perspective, this thesis strove to add portions to the conversation that I thought were lacking. To develop an intuitive workflow for any individual who cares to undertake projects of panoramics, to not set limits to user-defined parameters — all of these steps were necessary in the development of a tool that could potentially further the notion of working with multiple simultaneous perspectives, and the implications therein. The uses of these new tools we find ourselves possessing (not this thesis specifically, but the broader spectrum of new technologies regarding how we perceive visual information) can, in fact, change our world, from education, to advertising and entertainment, and everything in between — that's an idea considered to be of highest importance.

With that said, *In All Directions* began as an idea borne from recent advancements in physical camera rigs which use multiple cameras to collect visual data in a panoramic fashion. It then follows that my design process in generating a multiple camera rig started with an attempt to create an automated system (using scripting) to generate a similar effect.

I spent a considerable amount of time researching and developing a working prototype in Autodesk Maya, a popular high-end 3D application. Maya is essentially controlled by either its built in MEL API (Maya Embedded Language), or more recently, the Python programming language. As new versions of Autodesk Maya have been rumored to be converted entirely to Python, I believed it best to spend some time to learn the language, and from there, write the Camera Generation script in Python, that the script itself might prove useful long after the syntax of the MEL API was deemed obsolete.

Once the tools to create multiple-perspective imagery were built and put into place, I was free to act the part of artist; to paint a digital canvas from a unique point of view—that being many different angles simultaneously.

PROCESS

The process of this thesis was two-fold; first, the design process focused upon the creation of the camera rig script, and second, the integration of said script to create imagery that would test the bounds of the scripts' panoramic capabilities. As these are two distinctly different design processes, they will be discussed individually.

DESIGN PROCESS–SCRIPT

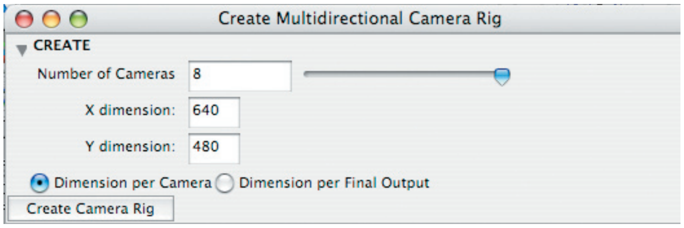
INTRODUCTION

In creating a script in Maya, I chose to spend a number of months investigating MEL (*Maya Embedded Language*, the API upon which everything in Maya is developed) as well as the Python programming language. Python is soon to be the successor of MEL; it was upon this realization that I chose to write the script primarily in Python. This decision allows for me to work with this script much further into the future than if I were to have written it in MEL.

As ease of use really became a central focus of the design process here, I chose to create a User Interface that would highlight accessibility and simplicity over many parameters that could potentially clutter the interface and ultimately confuse the user.

USER INTERFACE

The design of the interface was centered around two distinct concepts. First, the interface had to be intuitively as simple as possible, with all calculations of the script being generated behind the scenes, to speak. When I first began to develop drafts for what the script might look like, I decided that a tab-down system would work best to segregate each of the three separate modes of the script (create, edit, and render modes). In creation mode (where the camera rig is actually called into hypothetical space), the script had to be able to generate all necessary variables while limiting user input to one slider and two text fields *(Fig. 3).*


Create Multidirectional Camera Rig

▼ CREATE

Number of Cameras 8

X dimension: 640

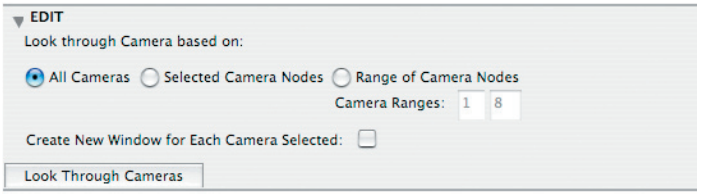
Y dimension: 480

● Dimension per Camera ○ Dimension per Final Output

Create Camera Rig

Figure 3–Creation Mode.

The former was to determine the number of cameras brought into the scene; the latter was to determine resolution of output on a per-camera or per-total-final-output basis. For example, were the user to know they wanted eight cameras at one particular resolution (e.g. 800 pixels by 600 pixels), they might input strictly those numbers. On the other hand, were they to know that their final resolution had to be 800 pixels by 4800 pixels total, they could also choose to adjust global parameters thusly as well. In the case of per-total-final-output, the script would determine the resolution of each of the cameras dynamically. This versatility in creation allows for the user to input whatever information they have, and not necessarily have to rely on calculation whatsoever. Difficulty in a simple user interface arises when one considers the ability of the program to also achieve all goals set out by the user— that is to say, functionality is equally as (if not more than) important as simplicity! Though a difficult balance to achieve, this script strives for and (according to user testing) exceeds expectation in this category. For example, Edit Mode of this script was created in order to preview various cameras in a multiple-camera rig *(Fig. 4)*. If the user chooses to view three of eight cameras, the script will allow them to do so within the confines of the default four-up display in Maya, with the fourth view allowing for the perspective viewpoint to be visible as an overview of the User's scene *(Fig. 5)*.


▼ EDIT

Look through Camera based on:

● All Cameras ○ Selected Camera Nodes ○ Range of Camera Nodes

Camera Ranges: 1 8

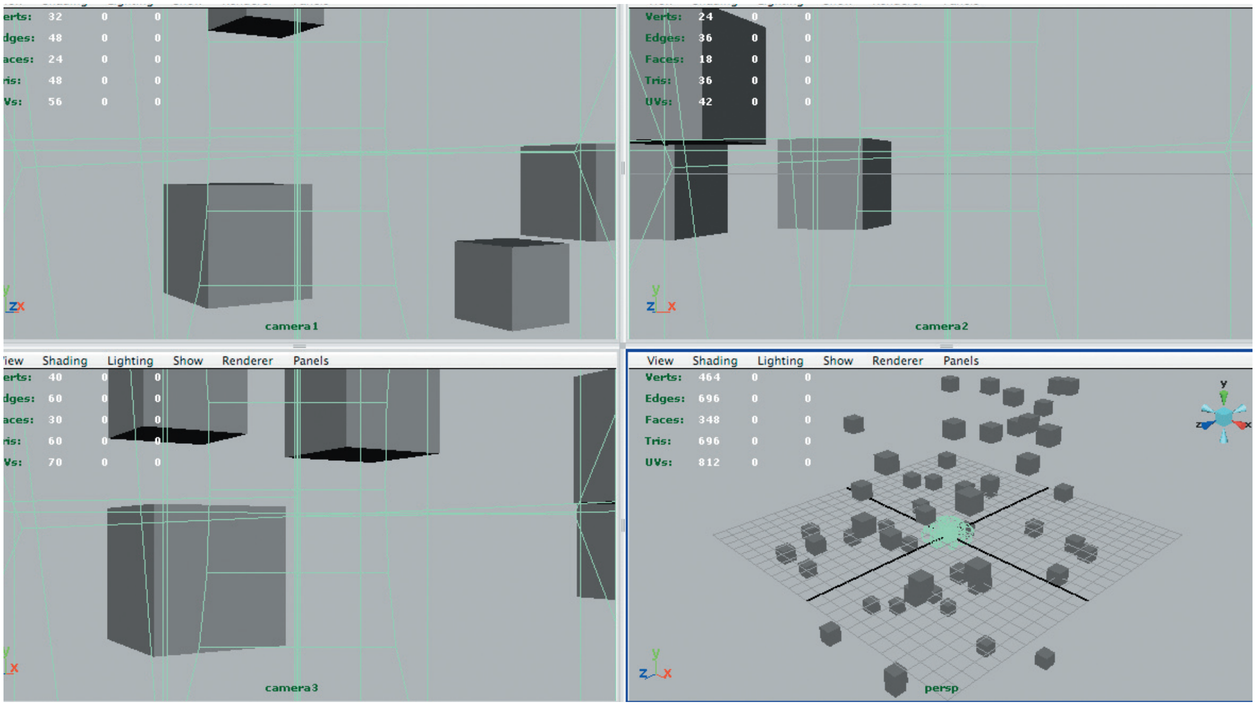
Create New Window for Each Camera Selected: □

Look Through Cameras

Figure 4–Edit Mode.

What if, however, the user wanted to preview six of eight cameras? The script recognizes that the user's selection will not fit within the traditional four-up display (where four perspectives are shown simultaneously - refer to Fig. 5 for example), and will automatically generate a new floating viewport for each of the cameras. Furthermore, it will try to place these new viewports in the space of the user's monitor, based upon the resolution settings the user had initially determined in creation mode. Lastly, Edit Mode allows the user to select the cameras for preview based upon any one of a number of parameters, whether the user wishes to look through cameras they've selected in-scene, or whether they wish to look through all cameras, or a selected array of cameras (e.g. cameras one through five). As simple as this functionality may seem, it really does come in handy, having the ability to preview any subset of cameras based on a variety of parameters.

TECHNICAL ISSUES–SCRIPT



```
erts:   32   0   0
dges:   48   0   0
aces:   24   0   0
ris:    48   0   0
Vs:     56   0   0
```

```
Verts:  24   0   0
Edges:  36   0   0
Faces:  18   0   0
Tris:   36   0   0
UVs:    42   0   0
```

```
erts:   40   0   0
dges:   60   0   0
aces:   30   0   0
ris:    60   0   0
Vs:     70   0   0
```

```
Verts:  464   0   0
Edges:  696   0   0
Faces:  348   0   0
Tris:   696   0   0
UVs:    812   0   0
```

camera1
camera2
iew Shading Lighting Show Renderer Panels
camera3
View Shading Lighting Show Renderer Panels
persp
y
z
x

Figure 5–Edit Mode. Notice Viewports have conformed to the viewing of an array containing cameras one through three.

Beyond obvious technical issues regarding scripting (and subsequent troubleshooting), a number of discrepancies emerged in the writing of this script. The troubleshooting phase of writing took up a vast amount of time, with each victory only adding minute amounts of code to the script.

One of the greatest issues to resolve was as follows: During testing, I was offsetting four cameras in a rig, each by 90 degrees (90 degrees apiece times four camera nodes equals 360 degrees total). Strangely, however, even with the angle of view set to 90 degrees, none of the camera angles would line up!

I discussed the issue with my committee as well as fellow student Elizabeth Skripp.   Since Ms. Skripp had worked on a project within the last year that involved multiple cameras (a flythrough of a Monkey's brain, initially proposed and created by Julia Lehman, a medical illustration student at RIT), I asked her how they had adjusted for discontinuity in each of the four cameras' field of view.

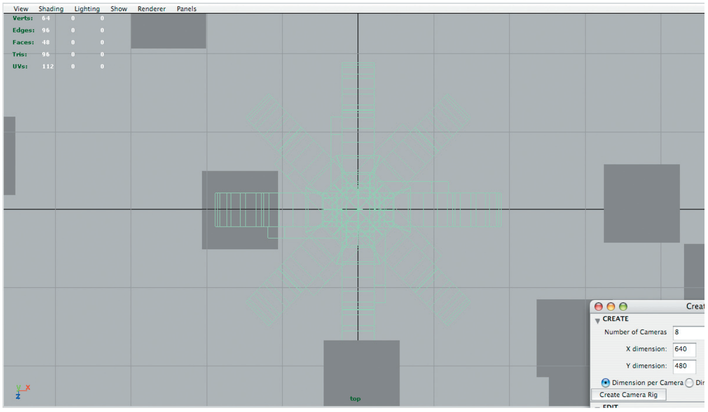
Since their project required only the creation of one camera rig, they had manually adjusted the angle of view of each camera to compensate for the lack of tiling camera outputs. Eventually, each camera in their rig lined up at an angle of interest of 97 degrees apiece; though this seemed to work for one particular camera rig, mine was to be generated dynamically by code for any number of camera nodes the user wished. Ergo, this solution could not work for my particular problem.

After consulting a number of books and technical manuals, I came across the Autodesk Training DVD *Learning Autodesk Maya - The Aesthetics of Cameras*, which was narrated by Cathy McGinnis. Though most chapters throughout the movie didn't relate much to my thesis, I gained a much more firm understanding of how cameras in hypothetical space work, as well as a solution as to why my multidirectional perspectives weren't lining up. In short, the issue that needed resolution was that the aspect ratio of the film gate had to match the ratio of each camera's output resolution in order to tile each perspective in space. Of course, the solution to one problem always seems to lead to another question needing yet another answer. Now that aspect ratios were creating tileable imagery, I noticed severe fish-eye distortion occurring at the edges of rendered output —but why?

Cameras (both virtual and actual) work on similar premises. Basically, the angle of interest (also known as field of view) and the focal length of the lens (the length in millimeters between the lens and the cameras filmback, where the actual light information is burned to film) are proportional (McGinnis). As the angle of interest increases, the focal length of the camera decreases, and vice versa. With the four-camera rig I had created (each camera having a field of view of 90 degrees), the focal length of each camera was set to 18mm. Typically, a wide-angle lens is anything of 35mm or less—this would quite obviously explain the reason for distortion!

After some experimentation with focal length, I determined a perfect compromise between focal distortion and render time would be an eight camera node system *(Fig. 7)*. Optimally (to match the level of distortion of the human eye), ten camera nodes would produce a per-camera focal length of 55.398 mm. Eight cameras, however, presented a decent focal length of 43.456 mm, and would also yield two cameras less to render, which would save a tremendous amount of time when it came to the render of final output from the rig.

In development of the script, one other such technical issue emerged. While one could easily determine what the angle of view for each of the cameras might be, this attribute is not keyable in Maya, and thus cannot be set by code. As a return to previous examples, if we have 360 degrees of imagery to capture and eight camera nodes equally spaced amongst the full panoramic perspective, each camera should have a field of view of 45 degrees, and a corresponding focal length of 43.456 millimeters. Interestingly, focal length is a keyable attribute, and as such, could be generated dynamically through code to properly set each camera node to it's correct angle of interest. To view this portion of the code, please refer to Appendix B at the end of this report.



```
Verts:  64   0   0
Edges:  96   0   0
Faces:  48   0   0
Tris:   96   0   0
UVs:   112   0   0
```

Number of Cameras 8
X dimension: 640
Y dimension: 480
● Dimension per Camera ○ Dir
Create Camera Rig

Figure 6–Eight Node camera system, viewed from top perspective.

DESIGN PROCESS – RENDERED OUTPUT

INTRODUCTION

After the creation of the script, the next phase of the thesis was centered around creating imagery using the script. Initially, three scenes were proposed as environments through which to test the script's capabilities, but given render times (to be explained later in depth), two scenes were instead created: An underwater environment, and an outer space environment.

DESIGN– UNDERWATER ENVIRONMENT

The underwater environment was the first to be created, and took the most substantial amount of time to render, given that the entire scene was encompassed with volumetric fog, to simulate the natural occurrence of perspective depth that occurs in oceanic environments. I consulted the Gnomon DVD *Dynamics: Underwater Scenes* for a considerable amount of insight in how to achieve underwater effects such as caustics and volume fog; it was a boon of information regarding special effects, and was overall quite beneficial as a starting point of an underwater landscape.

CAUSTICS

I initially generated a 512 pixel by 512 pixel caustic map (*http://www.lysator.liu.se/~kand/caustics/* – open-source caustic map generation) to create the underwater caustic effect. From there I brought it straight into Maya as a file image sequence, and plugged it into the color channel of a high-intensity spotlight. I then remapped the color, that it might have a slightly more yellowed tint to it, as the caustics generator generated black and white image sequences only (*Fig. 7*).

SHARK

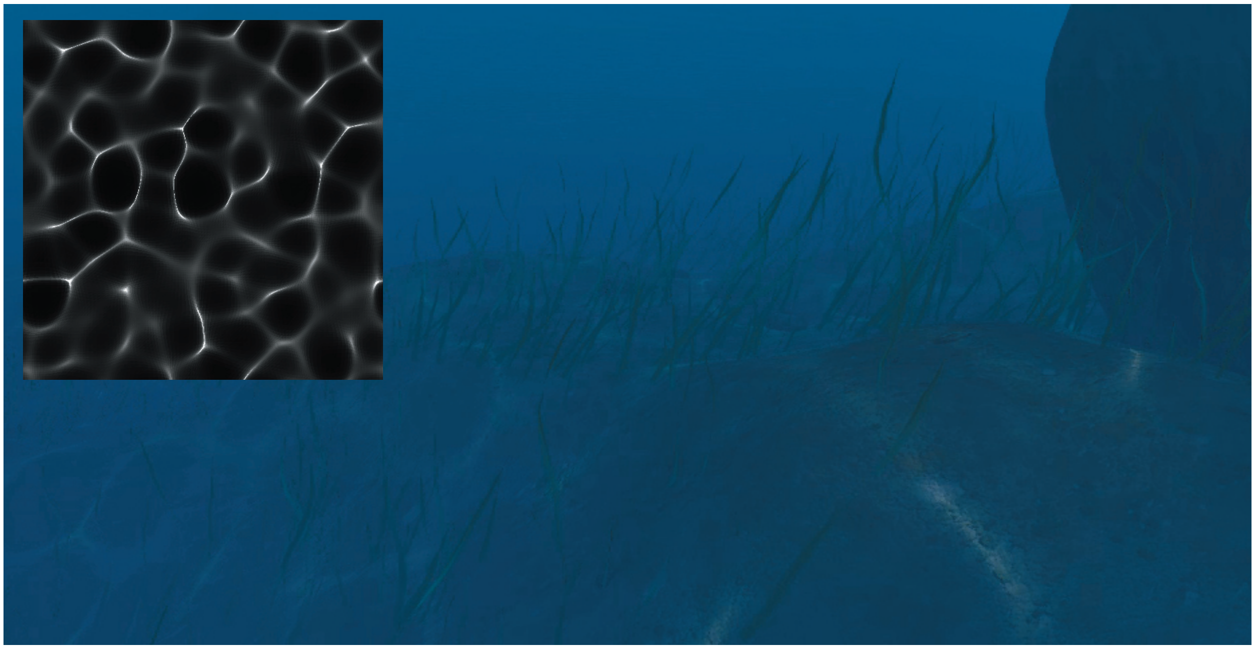


Figure 7– Caustic Light in Underwater Scene. Inset - Original Caustic Map.

The idea of including a shark came about early—after all, what is a underwater environment without these fascinating denizens of the deep? The shark in this scene follows a specific path lasting the duration of the entire scene, though almost every other motion created by the shark is actually hand-keyed (e.g. rotation of shark body along curvatures in path), with the exception of waveform functions guiding his sweeping tail-fin.

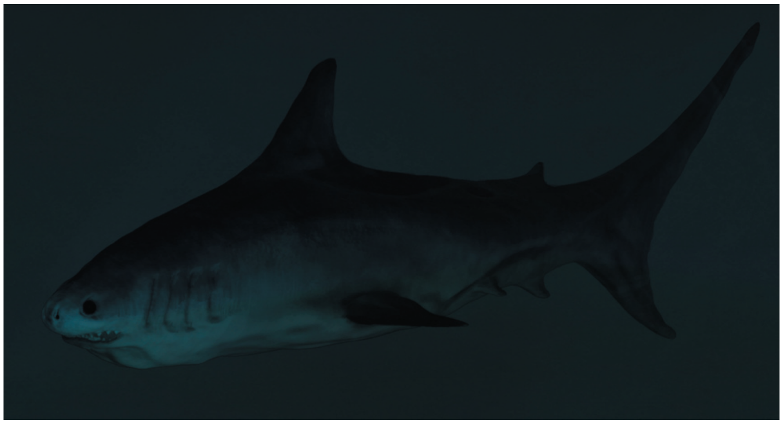


Figure 8– Original Shark Concept.

The original shark concept (*Fig. 8*) was created as an experiment with

Pixologic zBrush, a 2.5D displacement/painting program that I used in conjunction with Maya. In the final rendered output, however, careful observation will allow for one to see that any displacements in the proposed concept shark have been lost. The reason for this, like so many other things, is render time. I chose not to run a separate pass to render out displaced geometry due to the fact that this pass would be time consuming from one renderable camera, let alone eight! One of the greatest things I've learned from this entire thesis is that render settings need to fit time constraints, and vice versa. At thirty frames per second, most of the minute details added during renders are lost to the human eye. In this instance, the viewer needs to understand that this is a shark in order to form a narrative in their own mind from the visuals presented — though not perfect per sé, this in-scene shark seems to be effective in doing just that (*Fig. 9*).

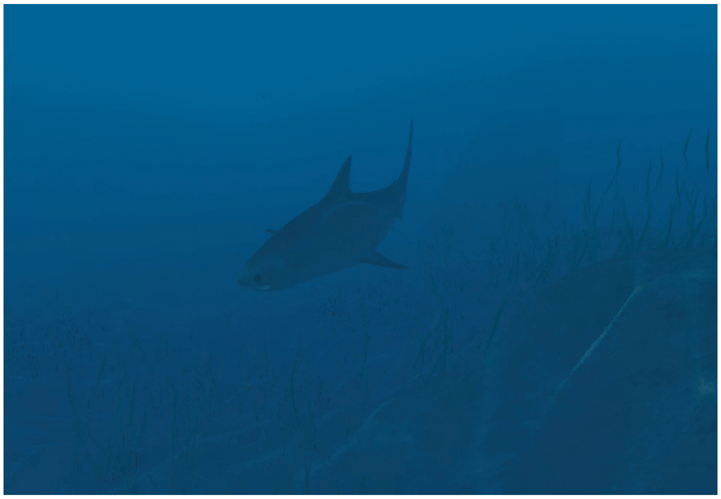


Figure 9–Shark in final rendered output.

One last facet I wish to discuss in regards to the underwater scene is the interaction between the school of sardines and the shark. As previously mentioned, the shark is placed upon a path through the scene. I also placed the school of sardines along a path—and because I like to see a bit of dynamic conflict, I decided it would be interesting to place the path of the sardines directly in line with the path of the shark.

This is the interesting thing about dynamic systems like this - you can simulate a scenario over and over, and always achieve different results. In this instance, I told the school of sardines that they had two proverbial "goals" in life. The first was that they needed to (as closely as possible) follow the curve they were assigned. Some offset was to be expected, but for the most part, that was how they, as a particle system, were to "reward" themselves. The second goal was a bit more nefarious — to avoid the shark at all costs!

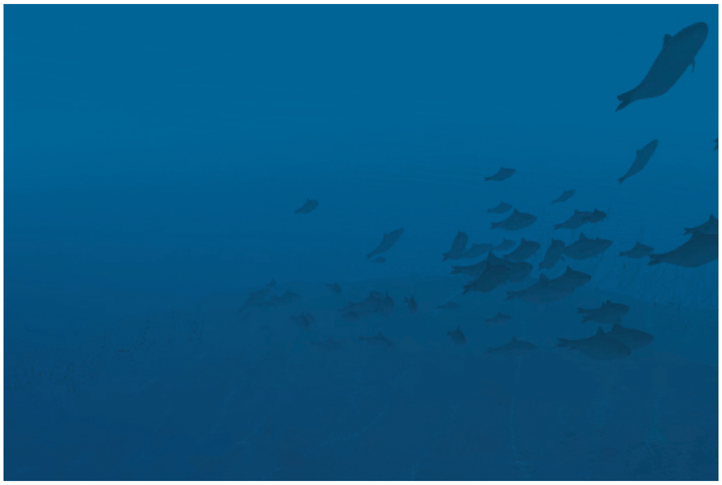


Figure 10–The school of sardines reacts to the shark, off-screen to lower right.

Of course, I gave them an imperfect scenario in which to attempt this, as I had purposefully made their path cross the shark's. As sheer experiment, I attached a radial field to the shark that also would also repel the fish if they got too close (*Fig. 10*).

After running several simulations under these conditions (some of them were down-right laughable, such as fish vibrating in place until the shark passes, etc.), I finally found one that didn't look too unrealistic, and baked in the simulation, that the fish would from this point forward no longer interact dynamically, but always follow this one path they had created. This was done to ensure that all eight camera nodes would render an identical motion path of the sardines - otherwise, each camera may have rendered different motion, and needless to say, that would defeat the purpose of aligning multiple simultaneous perspectives.

DESIGN− OUTER SPACE

INTRODUCTION

Outer Space is a very difficult ambient environment to recreate for two reasons: First, we must consider the vast amount of emptiness that is outer space. Planets are light years apart from one another; to recreate this phenomenon in virtuality is difficult, at best. Since there are very few depth cues between these objects (and furthermore, the vast differences between the scale of objects tends to distort any perceivable notions of closer and further), it becomes rather difficult for objects placed in space to have an assumed depth of location in regards to the viewer. Second, due to the lack of adjacent objects of interest, it's difficult to draw the viewer's attention for more than a moment. In my outer space environment, I strove for a balance between reality and exaggeration, trying to walk a fine line that would be of interest to the viewer, yet still hold true to the fact that outer space is relatively vacuous.

ATMOSPHERIC SHADING

While looking at images of planetary bodies in outer space, we notice that most planets have a haze or atmosphere surrounding them. While these atmospheres vary in color and density, one thing is true of all of them— due to atmospheric perspective, the haze around the edge of the planets seems to be much more concentrated than the haze on the surface. The more the planetary normals shift from parallel to the viewer (in this case, the camera) to perpendicular, the more densely the haze will appear. To recreate this phenomenon in virtuality, utility nodes such as the *sampler info* node may be used in order to collect camera versus geometric normal information (Clark, Maya Dynamics : Outer Space Environments).

In this instance, the output of the sampler info node is plugged into a black and white ramp node, which is then connected to the input of a shader group's transparency node. When the facing ratio to the camera is perpendicular (90 degrees), the sampler info node returns a value of one (or white), which in turn changes the transparency of the object at that point to opaque. Conversely, when the facing ratio to the camera nears parallel (0 degrees), the sampler info node returns a value of zero (or black). The transparency of the object at this point corresponds thusly, and remains completely transparent (*Fig. 11)*.

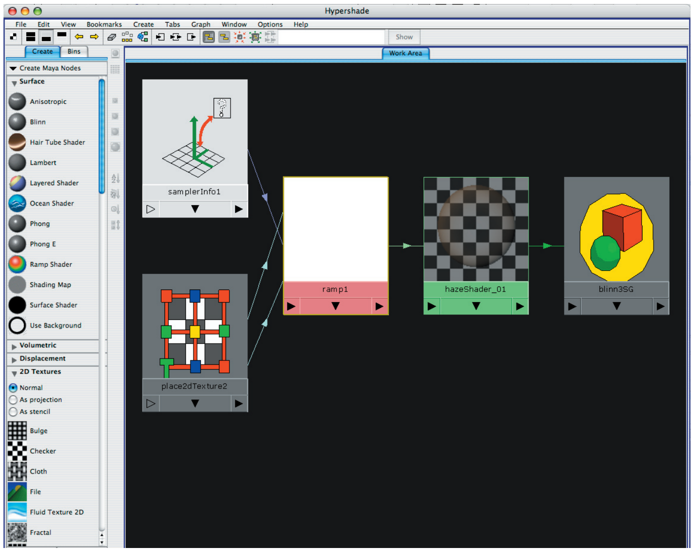


Figure 11–Hypershade perspective of sampler info node being plugged into ramp, followed by transparency of the shader network.

STAR CLOUDS / NEBULAE

Were one to travel into outer space, they would notice not only a plethora of stars, but also a vast expanse of swirling gases creating beautiful technicolor effects that stretch across the sky!

Using similar projection techniques as the caustic light in the underwater scene, we can achieve this sort of effect. Interestingly, in learning this technique, I learned a great deal about projection, but also in regards to particles and emitters, and how they can be formed to create exactly the visual effect I'm looking for (*Fig. 12)*.

By creating a nurbs sphere in Maya, I created a surface that I could make paintable, and did such that - using a grayscale gradient. In this instance, I tried to imagine all the areas that were black as the areas where the nebular fields would be located. From there, when I emit particles from the surface of the nurbs sphere, I can actually choose to "emit from color", which will give preferential location of particles to the places on the sphere which are colored black. All places colored white will receive no random particles, gray areas will receive some, et cetera.

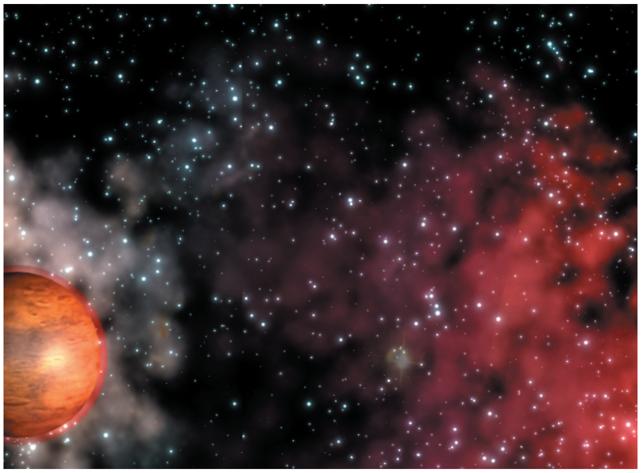


Figure 12–Resultant "hazed" atmosphere surrounds the planet.

Once the particles were adequately placed, I solved the initial state of the particles and turned their dynamic conservation to zero, that they would be indefinitely “frozen” in place. From there, I adjusted the particle type to a cloud type, that it could be rendered in Maya Software mode (*Fig. 13)*. Once all this was set up, I once again used large, high-intensity spotlights with nebular images from NASA (masked by their alpha channels) mapped to the color channel of the spotlight, and using the light-linking editor, detached all lights other than the projecting spotlights from the particle clouds (*Fig. 14)*. NASA graciously places all of it’s online image gallery in the public-use domain; it seems an understatement to say that this is an incredible resource for attempting the recreation of a gallactic environment. The end result of this environment was not as photorealistic as I had hoped, but given the vast amount of visual information presented to the viewer in a panoramic fashion, still conveyed the sense of the awe-inspiring vastness that is outer space.

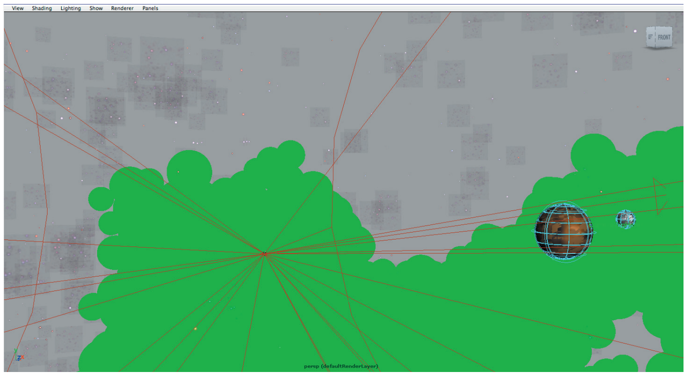


Figure 13– Cloud Particles to be projected upon in final output render.

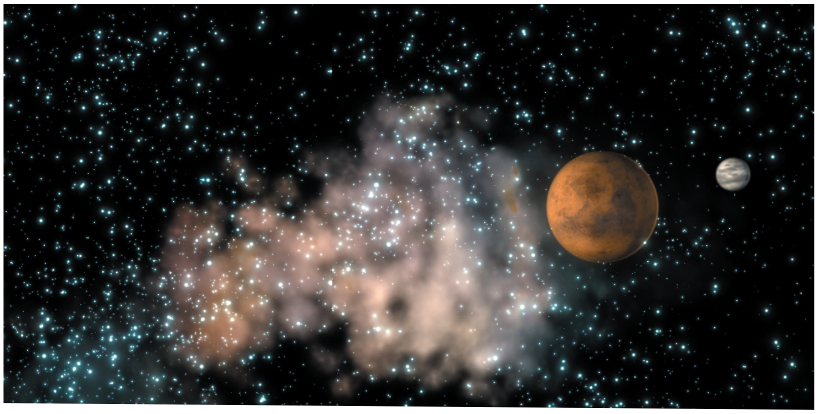


Figure 14–Final render of projected nebulae.

DESIGN–PRESENTATION

On May 23rd, 2008, I presented my thesis to my professors and peers at *Pixel Science,* the 2008 Rochester Institute of Technology Computer Graphics Design Thesis Show. The chosen medium in which to display these massive panoramic images was created using pine, lucite rods, and vellum stretched over the lucite, to act as the projection screen. Professor Jon Schull lent me six floor-projection systems, and my entire family lent a hand in installing the rig in three hours. Please refer to appendix B for select images from the presentation of this thesis.

SUMMARY

When this thesis began, many questions arose that were considerably challenging to answer. Since not much work had been done in regards to multiple camera rigs in hypothetical space, a good number of these challenges were tackled without a good foundation upon which decision making could be based. As such, a good deal of the work surrounding this thesis involved a great deal of research, followed by copious amounts of trial and error.

In regards to the success of this project, I believe a great deal of the aforementioned questions have, in fact, been answered. As with all things, however, hindsight now allows me to believe that were I to construct this thesis over, some things would be done differently. For example, one such thing that wasn't really calculated was render time. To render complex scenes through one perspective takes a considerable amount of time; to render eight separate perspectives takes what felt to be an eternity! Were I looking to addend this script, I would keep this in mind, and perhaps under the render tab, I would consider integrating the ability to render separate passes straight from the user interface, in order to hasten the process in any way possible.

Further additions might include the ability to render particle effects to disk cache from the UI. The reason for this, as mentioned in the underwater scene, is simple. Without baking a simulation of particles, each perspective would render out tileable images — except, of course, any particles (or particle instances, such as the school of sardines) within the scene.

As this thesis wrapped up, a number of uses and integrations were proposed. One interested party wished to look into this thesis in conjunction with RIT's Collaboratorium, that they might teach medical imaging classes with this script as a real-time panoramic visual aid in Maya. Though this idea never came to fruition, it exemplifies one of a number of uses for which this new form of visual communication might be utilized.

In conclusion, this thesis strove to push the envelope in regards to visual communication in hypothetical space — and did so quite successfully. The framework has been established for future work and possibilities, and it will be interesting to see what, from here, will emerge.

WORKS CITED

Bentham, Jeremy, The Panopticon Writings. Ed. Miran Bozovic. London, UK: Verso Publishers, 1995.

Dawson, Michael, Python Programming. Boston, MA: Thomson Course Technology PTR, 2006.

Gombrich, E. H. (Ernest Hans), The Story of Art. New York, NY: Phaidon Publishers, 1958.

High End 3D Website. Sept. 2007 - May 2008. <http://www.highend3d.com>

Kazmier, Chris, and Mark R.Wilkins, MEL Scripting for Maya Animators. San Francisco, CA: Morgan Kaufmann Publishing, 2005.

Learning Autodesk Maya - The Aesthetics of Cameras. Cathy McGinnis. DVD. Autodesk Publishing, 1997.

Maya Dynamics: Outer Space Environments. John Clark. DVD. Gnomon Workshop, 2000.

Maya Dynamics: Underwater Environments. John Clark. DVD. Gnomon Workshop, 2007.

Python Scripting in Maya. Digital Tutors. PL Studios. October 2007.

Stevens, Jane. “Exploring the Cave.” MIT’s Technology Review. March 1998.

APPENDIX A.

Code Sample : dynamically setting focal length for all cameras in rig

```
#----------------------------------------------------------------------------------------------------------------------------------------------

for cam_num in range(numCameras):
        bb_cam = mc.camera() #create camera
        mc.rotate(0, angles[cam_num], 0) # rotate camera in Y based upon number of cameras in rig # angles = 360.0/ num_cam
        mc.setAttr ((bb_cam[0] + “.tx”), lock = True) # lock translation and rotation for each individual camera
        mc.setAttr ((bb_cam[0] + “.ty”), lock = True)
        mc.setAttr ((bb_cam[0] + “.tz”), lock = True)
        mc.setAttr ((bb_cam[0] + “.rx”), lock = True)
        mc.setAttr ((bb_cam[0] + “.ry”), lock = True)
        mc.setAttr ((bb_cam[0] + “.rz”), lock = True)

        “”” The following determines the fov by determining focal length. Since
        fov is not an attribute that can be set, we must adjust the focal length,
        and expect the field of view to follow accurately.

        This formula was derived from AECameraTemplate.mel. “””

        aperture = mc.getAttr((bb_cam[0] + “.horizontalFilmAperture”))
        fov = 360.0 / numCameras
        focal = math.tan( 0.00872665 * fov) # .00872665 = conversion of 1 degree to radians (3.14159265 / 360 degrees)
        focal = (0.5 * aperture) / (focal * 0.03937) # .03937 = conversion of 1mm to inches
        mc.setAttr ((bb_cam[0] + “.focalLength”), focal) # Set the focal length (and thus, FOV) for each camera
        mc.setAttr ((bb_cam[0] + “.renderable”), 1) # Set each camera created to be renderable
        mc.setAttr(“perspShape.renderable”, 0)

mc.select( “camera*”, r = True) # Once all cameras have been created, select all cameras.
mc.group( name = “RigTransform”) # Group all cameras as “Rig Transform”.

#----------------------------------------------------------------------------------------------------------------------------------------------
```

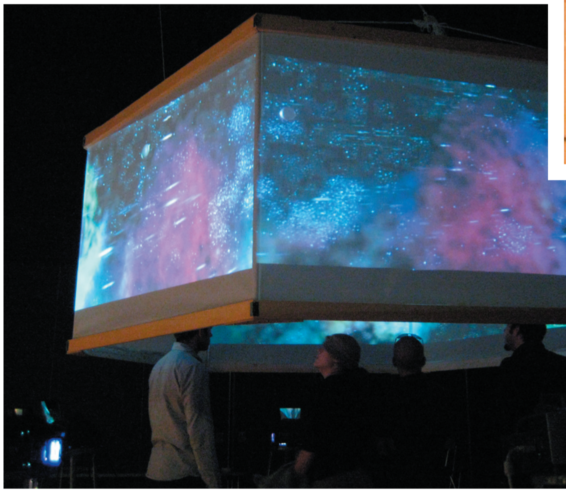
The above section of code is straight from the function getCreationValues, which parses information directly from the “create” section of the script. Based upon user input, the script is fed a number of variables, such as the number of cameras (numCameras) selected by the user, as well as the angle of each camera’s Y rotation (angles[cam_num]).

Notice that the latter half of the script shown above is dedicated to setting the focal length of each camera. Once focal length is determined, the field of view is also set appropriately. In order to come to exactly how to set the focal length of each camera, I actually had to look into pre-existing MEL scripts built into Maya in order to extrapolate the mathematical function to call — in this case, *AECameraTemplate.mel* .

APPENDIX B.

Select images, Rochester Institute of Technology, Rochester NY - PIXEL SCIENCE Thesis Show, 5/23/08

In All Directio
Brendan Bond Thesis Exhibition Master of Fine Arts Degree
Twenty-three May Two Thousand and Eight Rochester Institute of Technolo



APPENDIX C.

Review Of Literature

Python Programming, Second Edition
Michael Dawson
Thomson Course Technology PTR, Boston, MA, 2006

Python Programming for Beginners is an exceptional introduction to the world of the Python programming Language. The unique approach of learning Python through hands-on tutorials leads the user to a thorough understanding of the syntax of this highly versatile programming language.

Digital Tutors : Python Scripting in Maya
Author Unknown
PL Studios, October 2007

Python Scripting in Maya gives a first-hand look of how to construct Python code within Maya. Since (as of Version Nine, or 2008) Maya's Embedded language and Python are for the time-being closely integrated, it is essential for the author of Python Scripts in Maya to have not only a working understanding of both languages, but also the syntax that allows for these languages to communicate integrally.

MEL Scripting for Maya Animators, Second Edition
Mark R. Wilkins, Chris Kazmier
Morgan Kaufmann Publishing, San Fransisco, CA, July 2005

As previously mentioned, MEL and Python are currently still interwoven, and as such, an understanding of the Maya Embedded Language is also necessary to write Python scripts in Maya. MEL Scripting for Animators is an exceptional resource for those looking to delve into the more technical aspects of Maya's programming capabilities.

Maya Dynamics : Outer Space Environments DVD
John Clark
Gnomon Workshop DVD, November 2000

Outerspace Environments tackles difficult scenarios regarding outer space scenes in hypothetical space, including particle starfields, haze atmosphere shaders for planetary bodies, and paint FX starscapes. One difficulty of working in outer space environments is making them look as real as possible while managing such a vast expanse of virtual space, and this resource explains this phenomenon (and subsequent solutions) quite well.

Maya Dynamics : Underwater Environments DVD
John Clark, Alex Alvarez
Gnomon Workshop DVD, February 2007

The second of two Gnomon Workshop discs regarding dynamics, Underwater Environments discusses a number of difficult technical issues whilst recreating underwater environments, including caustics, particulate matter, and instancing in order to create particle-driven schools of fish.

High End 3D Website
http://www.highend3d.com
Visited 9/07 - 5/08

Highend3D is an exceptional resource regarding all things three dimensional. Since forums on this site are maintained and addended regularly, any questions pertaining to 3D are often answered here. In particular, this site is considerably beneficial whilst working with technical aspects of Maya, such as programming and troubleshooting.

“Exploring the Cave”
Jane Stevens
MIT’s Technology Review, March 1998

This article explores the benefits of virtual environments in a real-world scenario. Through visual cues in a CAVE (a recursive acronym for Cave Automated Virtual Environment), engineers and designers can solve design problems before they actually occur in the real world. This aspect of multiple perspective environments proposes not only an aesthetic panoramic virtual environment, but also produces a state of the art tool to tackle the most difficult of design problems, potentially saving corporations millions of dollars.

Learning Autodesk Maya - The Aesthetics of Cameras
Cathy McGinnis
Autodesk Publishing, April 2006

The Aesthetics of Cameras is A DVD published by Autodesk. The narrator, Cathy McGinnis, thoroughly describes the attributes of cameras in hypothetical space. Since these virtual cameras are based on real-life lenses, the technical knowledge of their operation is truly crucial to understanding not only multiple-projection rigs, but also quite useful in regards to single-camera technique, as well.