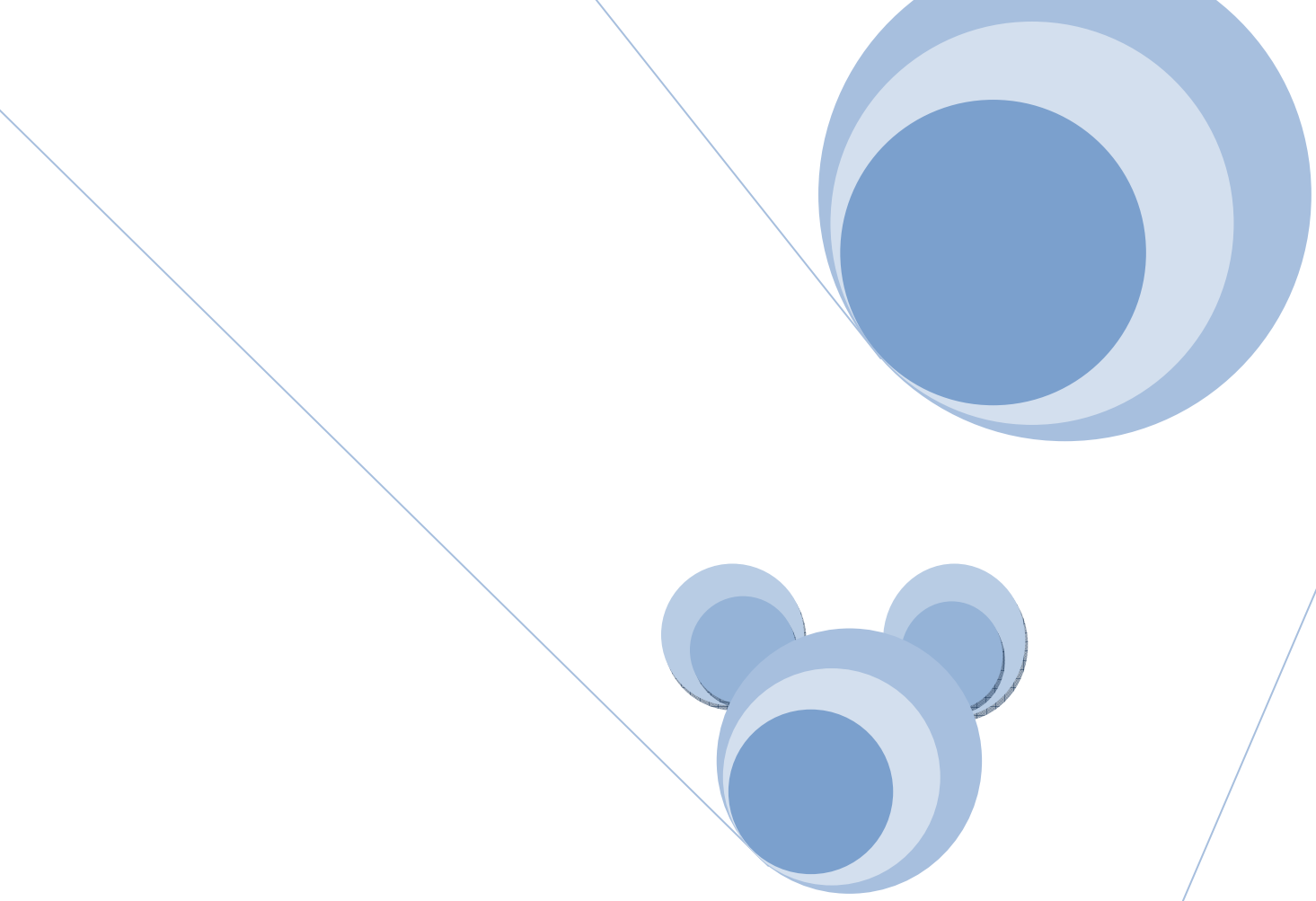


Maya FICA Tool

Fine Interactively-Lit Complex Aggregates

A Maya Plug-in Development Tool

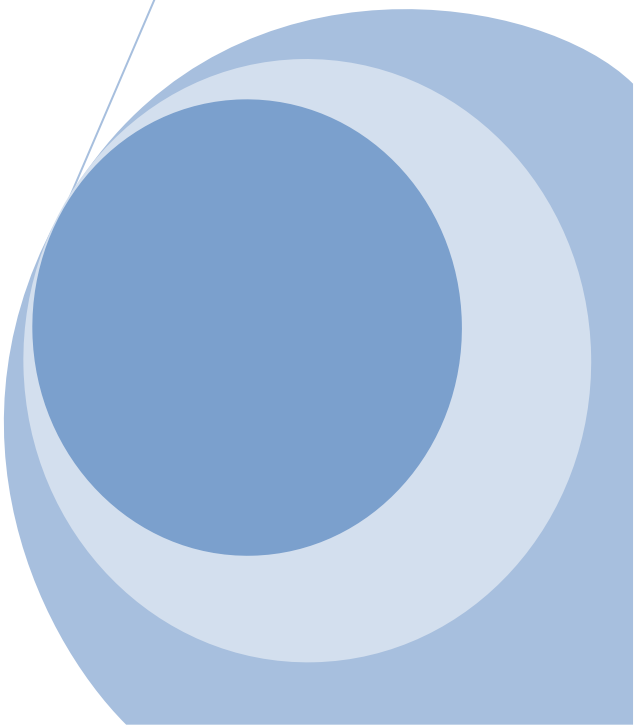
Kevin C. Chang
Kendra J. Gibbons

2/11/2009

Tool based on:

Perception of Complex Aggregates
Ramanarayana, G., Bala, K., Ferwerda, J. (2008)

Interactive Relighting with Dynamic BRDFs
Xin, S., Zhou, K., Chen, Y., Lin, S., Guo, B. (2007)



AUTHORING TOOL DESIGN DOCUMENT OUTLINE

TITLE: *Maya FICA Plug-in (Fine Interactively-Lit Complex Aggregates)*

PROJECT SUMMARY

The goal of this project is to design an interactively lit scene with automatically generated complex aggregates. The tool will help reduce the time spent by an artist in the production of a scene. Designing the lighting for a scene becomes a lengthy project due to long render times. In general, an artist will make small adjustments to the position and intensity of lights, and then is required to wait for a render. Speeding up this process will significantly increase the speed and ease of the workflow. Similarly, scenes that contain large aggregate groups lead to increased calculations and rendering times and can become difficult to maintain for the artist. By decreasing the number of polygons present in the aggregate while preserving the original scene, we can greatly improve the overall efficiency of the scene.

Aggregate groups present a formidable challenge in computer graphics. This is due to the inverse relation between the number of polygons in the scene and the focus of the viewer as the group size increases. When there is an individual object on screen, focus is drawn to all of the specific details of the object. However, as the number of objects increases, the viewer begins to shift focus towards the group as a whole, tending to see the overall features of the group instead of the individual details of specific objects in the group. However, this increase leads to a linear increase in the polygon count when it is clear that details aren't as necessary at this level. The Maya FICA tool aims to reduce the polygon count of the aggregate while maintaining the desired image of the group to prevent the unnecessary use of polygons.

FICA will provide studios with a method to solve lighting and aggregate bottlenecks in the existing pipeline. Lighting is one of the last steps of the pipeline and there is still a lot of room to increase the efficiency of this step. FICA's interactive lighting algorithm will light a static scene using pre-computed data. The data, a transit tensor, will be based on the objects in the scene and the lighting position. After computation the user will be able to change the material properties of the objects, the light positions, and the view of the camera. Because the user is unable to move objects in the scene without re-computing, this tool will be best suited for environments. Despite the large pre-computation time an artist can then light the environment quickly and accurately. The power to change lighting and surface attributes of a scene and render interactively will dramatically increase artist workflows, allowing them to spend more time on creating effects they desire.

We will be implementing FICA using MEL scripting and the C++ API. The graphical user interface and the aggregate tools will be developed using MEL. The algorithms for the aggregate population scheme and ratio adjustment will be based on "Perception of Complex Aggregates" by Ramanarayana et al. The interactive lighting algorithm will be written in C++ according to

Xin Sun's paper "Interactive Relighting with Dynamic BRDFs." The alpha release will be completed in the first four weeks and will implement the majority of the aggregation code as well as the initial BRDF decomposition calculations. The beta release will finish the pre-computed transfer tensor, begin the direct lighting calculations, and include the completed aggregation code. This will be completed by week seven. The final release will be completed by week ten and will finish the final lighting calculations, implement specular lobe separation, and tie everything together into the final product.

1. AUTHORING TOOL DESIGN

1.1. Significance of Problem or Production/Development Need

A stunning rendered scene is made by attending to the small, sometimes forgotten aspects of the scene. Humans are incredibly adept at discovering inaccuracies of a scene, so we hope our tool will help artists improve the believability of aggregates and scene global illumination. FICA will help to reduce the time an artist needs to complete these two important tasks.

Aggregate scenes are a common place for incredible complexity in computer graphics. Large aggregates of objects greatly increase the polygon count of the scene, causing increased rendering times for shots in movies and increased calculations per frame in video games. Interestingly enough, large groups of objects have an inverse relationship to the amount of detail required for each model. When there is only one model present, the eye tends to focus on the specific details that comprise that model. However, as the number of models increases in the scene, focus tends to go from individual models to the entire scene as a whole. As a result, the individual model details are not as important, which provides the opportunity to decrease the polygon count of the scene.

There has been significant research focusing on level-of-detail (LOD) methods for reducing polygon counts. While this is certainly an effective strategy for objects in the distance or further away from the screen, it can be noticed if placed alongside the same object at a higher resolution, a problem when dealing with groups of similar objects like an army of soldiers or a garden of flowers. In those situations, soldiers with reduced meshes will certainly be spotted when placed next to similar soldiers of higher detail.

We plan on creating an alternative solution for complex aggregate scenes. Instead of providing a LOD solution for an aggregate group of objects, we propose to change the ratio of the objects that populate the group. Consider a garden with ferns and flowers, a model with a much higher polygon count than the fern. By changing the ratios of the two and increasing the distribution of ferns in the garden, we can decrease the number of polygons used in the scene while maintaining the same appearance in the eye of the viewer. The tool will also allow for LOD changes if the user wishes to incorporate a combination of the two solution so model meshes can be smoothed or reduced in order to blend into the aggregate group.

Lighting in general has a very high computational cost at render-time, but many applications are moving toward real time simulations and rendering speeds. Full global illumination, the most accurate form of scene lighting, is still not feasible for scenes of consequence in real time. This paper focuses on increasing the speed of rendering dynamic lighting so such scenes can be lit with full quality lighting and still render at interactive speeds. Other papers can dynamically light a scene but previously either the BRDF parameters, the viewpoint, or the light positions had to be constrained. This paper focuses on implementing all three simultaneously. Currently this is most useful method for increasing the workflow of the lighting and shading artists. Instead of having to minutes or hours to see the differences light positions or shader properties have on a scene as a whole, the artist can see their changes almost immediately. Hopefully one day these papers will lead to a full real time render of the scene which will drastically change what the gaming world can do with its interactive lighting.

1.2. Technology

We will be combining elements of two SIGGRAPH papers to develop our Maya FICA plug-in. The first paper, "Perception of complex aggregates" by Ramanarayanan, Bala, Ferwerda, 2008, adjusts the ratio of objects in order to populate an aggregate scene while decreasing the overall polygon count. Ramanarayanan, et. al, conducted multiple experiments to adjust the ratio of various objects in aggregate scenes in order to determine if viewers could detect a change in the aggregate group. The overall conclusion was that given a certain starting ratio of objects, one could adjust the ratio in order to use more objects of lower polygon counts and achieve the same end result. Figure 1 shows an example of the experiments, the end result, and a 20% difference in the polygon count.

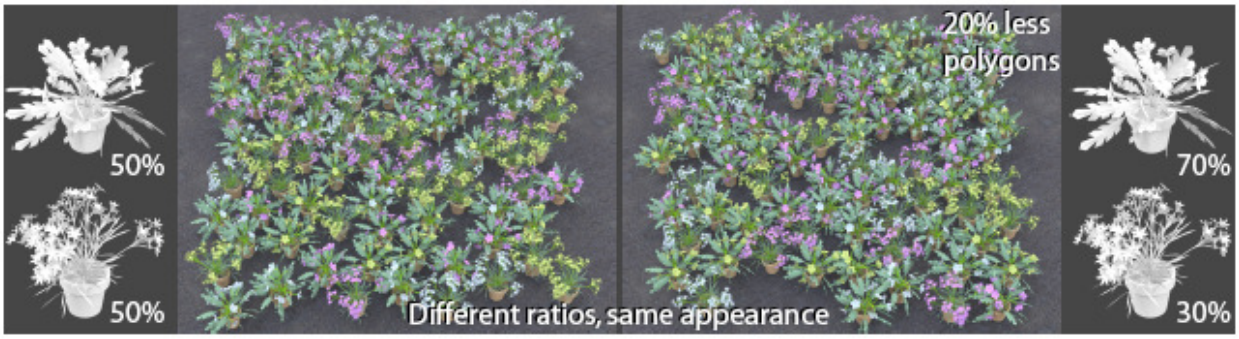


Figure 1: Ramanarayanan et. al, conducted experiments with the prediction and resulting confirmation that the above two images had the same appearance. However, the image on the right uses a higher ratio of the first plant which has a lower polygon count, resulting in a 20% difference in polygon count.

The second paper "Interactive Lighting with Dynamic BRDFs" by Xin Sun, et al. 2007, allows the artist to control the lighting, viewpoint and dynamic BRDFs in a scene all interactively. Previous papers had been able to achieve two of the three freedoms, but Sun's use of PTTs instead of PRTs allowed him to give the user control over the BRDF parameters.

In general an artist would wait around to see the rendered changes when he or she applied a new lighting position or moved the renderable camera. In the case of a scene with many repeated objects and a high polygon count, this could take an unruly amount of time. Pairing up these two papers could allow the artist to finish two tedious tasks in a much shorter time line and with much less stress and attention to detail.

1.3. Design Goals

Our tool will allow users to populate and interactively light large aggregate groups while reducing the overall polygon count. This will allow for increased performance when interacting with and rendering the scene. In addition, users working with a polygon count constraint can specify this limit so the tool populates up to the max number to generate the aggregate group. With the help of this tool artists will be able to generate large and complex scenes more efficiently, saving both time and effort. In addition, the tool will provide a graphical interface for the level-of-detail solutions for the mesh including smoothing the model as well as reducing its complexity.

1.3.1 Target Audience.

The target audience for this tool will be technical artists in both the CG movie and video game industries. This tool will greatly reduce the time needed for artists to model and accurately light large aggregate scenes. This gives more time for fine tuning the scene in order to achieve the desired image or animation.

In addition, lighting technical artists and technical directors will be particularly interested in the interactive lighting methods offered by the tool. They will be able to see in real time the full lighting effects of BRDFs in the scene, which will save time as draft renderings and images will no longer be needed as often in order to determine correct lighting for the scene.

Another area where this tool may be useful is in the area of crowd simulation. When dealing with large groups of entities, this tool can be used to ease the polygon count and resulting computations required for the animation of the crowd.

1.3.2 User goals and objectives

The technical artists will use this tool to generate large aggregate groups while reducing the polygon count needed for the overall scene. In addition, they will be able to more accurately determine the lighting of the objects using the pre-computed tensors instead of having to render draft images in order to determine the final output. In addition to generating aggregate groups while reducing polygon counts, the tool can also be used to simply generate a large number of models using a friendlier interface than the commands in Maya.

1.3.3 Tool features and functionality

The FICA tool will be a very user-friendly graphical interface which will allow the artist full control of the inputs to populate the scene. The artist will be able to specify a variety of options before clicking a button to populate a scene with an aggregate group. Pre- calculating the lighting tensors will require no user input but the interactive lighting will allow the user to adjust the view, the lighting and the BRDF.

In order to populate the scene, the artist will first specify the models that will be included. There will be optional inputs such as the total number of models desired in the group or the maximum number of faces allowed for the entire group. These will be used to constrain the size of the resulting aggregate group. The model meshes can also be smoothed or reduced to provide LOD solutions for the group or to use the new models as the base of the aggregate. When the aggregate is generated, the user will have the option of deleting the group and regenerating it in order to reach a better solution using the random seeding method of the tool.

Once the tensor has been calculated the user will have full control over the parameters of the BRDF through sliders, and inputs boxes. The user will not be able to move objects in the scene after the tensor has been calculated, as that will invalidate the calculated data.

1.3.4 Tool input and output

The tool will take several inputs before it can generate the expected results. First the artist will select the models desired for the aggregate and for the light tensor pre-calculations. The next step is to use the graphic interface for the tool to select constraints for the aggregate as described above. The artist will be able to input constraints on the maximum number of polygons or the maximum number of models for the aggregate and then click on a button to generate the resulting aggregate on the screen. Similarly, the user will be able to click on a separate button to do the light tensor pre-calculations for the interactive lighting for the scene.

The output for the tool will be either the calculated tensors for the scene or the aggregate group on the screen. The group would consist of an adjusted ratio of the specified models located around the models originally selected by the artist. The group will be grouped and given a default name as will each individual model. The tensors would allow the user to adjust the lighting and see the renders appear in the IPR renderer.

1.4. User Interface

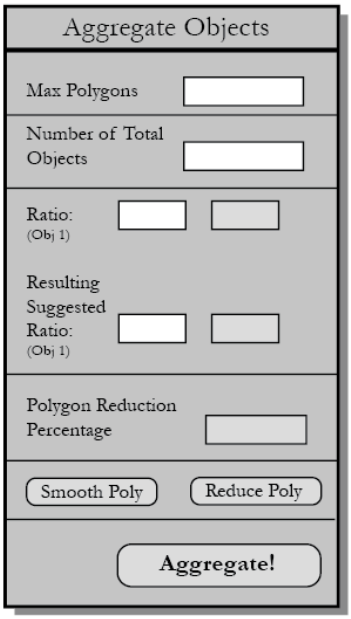
The user interface will be a large window with various buttons, input boxes, scroll bars, and check boxes. The interface will be an intricate part of the tool as it is what the artist will see and

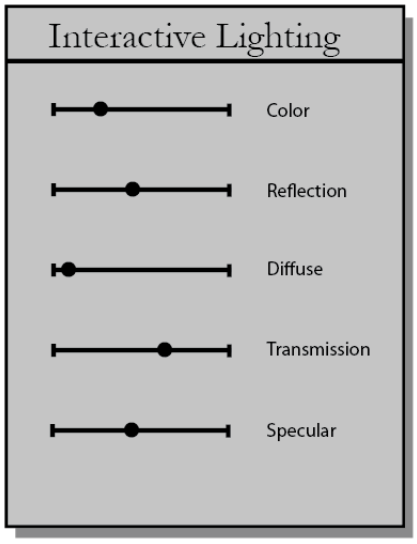
use in order to generate the desired output. In addition, a very friendly interface will make learning the tool significantly easier as most of the functions will be self-explanatory and will not require significant amounts of time or effort.

1.4.1 GUI Components and Layout

The graphical user interface will consist of input boxes and buttons. In addition, the user will be required to select models in object mode outside of using the GUI. After selecting the models, the rest of the inputs are handled in the interface before generating the aggregate.

The GUI will be laid out as seen at the right. At the top will be the numerical constraints for the aggregate including the maximum number of models parameter with a textbox for the user input and a maximum polygon counts parameter and textbox for user input. Below those will be a new set of textboxes for the user to specify what ratio of the objects he or she would like in the resulting group. Alongside these textboxes is a second set of textboxes, which will show the resulting experimental ratio, which can be used to adjust the aggregate group while still maintaining the appearance of the original user specified group.





The lighting GUI is composed of two stages, a computation stage and an interactive stage. The computation stage will be a button on the custom Maya shelf, as it will take multiple hours to complete.

After the computation is finished the user can open a window to control the various sliders relation to the BRDF parameters. View and lighting can be controlled as usual in Maya. The BRDF parameters include the specular reflection, the spectrum, the diffuse, etc.

1.4.2 User Tasks

The graphical user interface allows for smooth and easy aggregate generation and interactive lighting of the BRDFs in the scene. First, the user will be required to select the models in object mode before interacting with the user interface. The rest is handled in the GUI with the user just specifying certain constraints and hitting the generate button to execute the aggregate command. The user will not

need to have any other knowledge in order to execute the command as the backend algorithm will determine the adjusted ratio and will display it for the user to see. If the resulting aggregate group isn't desired, the user can also input his or her own adjusted ratio to change its appearance. The only skills required by the user are artistic in nature.

To control the lighting aspect the user will not need to control any parameters for the pre-computation. After the objects have been aggregated one function will begin the pre-computation of the transfer tensors. After these have been computed the artist can no longer move the objects in the scene, although they then will be able to change the lighting, viewpoint and the parameters of the dynamic BRDF. The BRDF changes will occur by adjusting sliders while the other two components will be handled as always in the Maya scene.

1.4.3 Work Flow

The workflow begins with the artist selecting the models to aggregate. These will be the models that are populated to create the aggregate group. After selecting the objects, the artist opens up the GUI and inputs the constraints desired for the aggregate, including the total number of models to create, the maximum number of polygons to use, and the ratio of the objects. Based on the specified initial ratios, the tool will show the suggested adjusted ratio to use to create the same appearance for the aggregate while reducing the polygon count. With all of these values set, the artist then clicks on the generate button and the scene is populated with the selected models.

The process is a very simple and quick procedure that leads to the generation of very complex scenes in a relatively short amount of time. In addition, it allows for a reduction in polygons while preserving the desired aggregate. It requires the models as inputs generates as output the aggregate group consisting of the selected models.

2. AUTHORING TOOL DEVELOPMENT

2.1. Technical Approach

2.1.1 Algorithm Details

The key algorithm for the aggregate generation is altering the ratio of objects to maintain the same perceived group by using the experiment results from the Ramanarayanan et al. paper. There are several properties of aggregates that are of importance when creating same appearance aggregates. These include numerosity, variety, and arrangement.

Numerosity is the number of objects visible in the scene. It is the fundamental defining property of aggregate groups. As the size of a group increases and the number of models increases, focus goes from individual objects to the group as a whole. There is no set number for when this focus shift takes place, but numerosity is the key component of aggregates.

Variety is the range of variations in the properties of the objects in the aggregate including shape, size, color, material properties, and other aspects of the objects. Variety impacts aggregate perception based on the varying differences of the objects in the scene.

Arrangement is the layout or spatial arrangement of the objects in the aggregate. In addition, it is multidimensional and includes regularity, density, and spatial dimensions. Regularity specifies the pattern of the aggregate, for example the effect of a group of trees that are lined in a row in an orchard versus a randomly distributed layout often seen in natural forests. Density describes the compactness of the aggregate. Finally, spatial dimensions can also vary for different aggregates, whether it is one dimensional in a string of beads, two dimensional in plants or terrain, or three dimensional with leaves on a tree.

The aggregate group experiments conducted by Ramanarayanan et al. took all of these properties into account in order to see what reductions in the overall polygon count could be achieved by adjusting the ratio of lower count objects while still maintaining the desired aggregate appearance. We will be using their results and findings as a guideline for our tool when populating aggregates in Maya.

The lighting algorithm is broken up into multiple stages, transferred incident radiance computation and the relighting computation, and specular lobe separation.

The pre-computation transfer tensor (PTT) for the transferred incident radiance must be completed first as it will contain much of the data needed at render time. PTTs decompose indirect lighting into pre-computable components that are each a function of BRDFs in the scene. These components can then be rapidly combined at run time to determine the incident radiance. In order to create these components the BRDFs need to be discretized into a third order tensor first. Using the resulting tensor, the source radiance and the viewing angle the transfer incident radiance can be computed, broken into components based on the BRDF the ray has hit. These components will then be stored in PTTs. Instead of implementing multiple bounces we will be simplifying the algorithm and only reproducing one bounce. The transferred incident radiance and the BRDF along with the current viewing and lighting positions can then be used to evaluate the relighting equation. The final implementation detail reduces the computational requirements for achieving accurate results, the specular lobe separation. The BRDF is broken down in order to limit the number of basis terms needed to

represent the entire BRDF. This reduces the errors created when the BRDF is approximated to create the tensor.

2.1.2 Maya Interface and Integration

The graphical user interface for the tool will be developed using MEL scripting. This GUI will be the bridge between the front end the artists sees and the back end where the algorithms and calculations are performed.

The pre-computed transfer tensor calculations will be implemented using Visual Studio using the C++ API. The final lighting equations will also be implemented in C++ and then

The aggregate population scheme and algorithms will be implemented using MEL scripts. The majority of the functionality and features of this tool will call functions and commands already available in Maya. For example, the level-of-detail changes will use the Maya mesh reduce and smooth commands. The main focus of the aggregate generation will be the algorithm to populate the objects in the scene. We chose MEL for this implementation to allow for a faster output and generation of the aggregate.

2.1.3 Software Design and Development

PTT Structure – This will most likely be represented as a vector of matrices. This will be done to keep the structure standard and intuitive as well as to take advantage of the basic linear algebra functions present in the C++ API.

Aggregate Populate – This algorithm is a semi-random populate function that will place things in a grid so that objects are dispersed evenly throughout. A 2D array will be used to keep track of where objects are placed so the other object type can be populated randomly around the first. In addition, we will keep track of a max and min value for a bounding box which will contain all of the populated objects.

This function will not require any data structures as we will be able to call the Maya duplicate function to copy the object and then translate it around the screen.

2.2. Target Platforms

2.2.1 Hardware

The hardware requirements for running the tool only require that Maya can be launched and the plug-in loaded. As a result, the minimum requirements are the same as those for installing and running Maya 2008:

- Windows: Intel Pentium 4 or higher, AMD Athlon 64, or AMD Opteron processor
- Macintosh: Intel-based Macintosh computers
- 2GB RAM
- 2 GB hard disk space
- Qualified hardware-accelerated OpenGL graphics card
- Three button mouse with mouse driver software
- DVD-ROM drive
- Keyboard

The pre-computation of the PTTs can become very expensive; finishing the calculations in a reasonable amount of time will require a render farm with the computation power equivalent to 30 computers with 2.8 Ghz processors.

2.2.2 Software

The tool will be developed using Microsoft Visual Studio 2008 in order to build the plug-in using C++. The graphic user interface will be created using MEL script for Maya 2008. In order to use the tool, the end user will need Maya 2008 to run the plug-in. To ensure that Maya can be installed and executed, the user will need to have one of the following operating systems if using the 32-bit version of Maya:

- Microsoft Windows Vista Business
- Microsoft Windows XP Professional OS (SP2 or higher)
- Red Hat Enterprise Linux 4.0 WS (U4)
- openSuSE Linux 10.2
- Fedora Core 5
- Apple Mac OS X 10.4.9 (or higher)

In addition, if using the 64 bit version of Maya, the user will require one of these operating systems:

- Microsoft Windows Vista Business
- Microsoft Windows XP x64 edition (SP1 or higher)
- Red Hat Enterprise Linux 4.0 WS (U6)
- openSuSE Linux 10.2
- Fedora Core 5

2.3. Software Versions

2.3.1 Alpha Version Features (first prototype)

The alpha version of our tool is designed to allow for the completion of the majority of the graphical features of our tool. We adopted this strategy to allow for a more interactive presentation of our tool for the first deadline March 16th, 2009, when we will be presenting our current progress in class. The alpha release will include the complete graphic user interface that the artist will use as well as several initial working functions for it. Users will be able to select a model and reduce or smooth the mesh. In addition, the initial design for populating the aggregate group will be available.

In addition, we will begin work on the BRDF decomposition, which is the initial step needed for interactively lighting the scene. The steps following that step all require the successful completion of step 3 so it is essential that the discretization is completed fully and on time.

2.3.2 Beta Version Features

In the beta version of our tool we will have all of the aggregate functions completed and working. Following the alpha release we will finish the algorithms and ratio adjustments for the populate function and will be primarily focused on the interactive lighting portion of the tool. There is also another bottleneck at this stage in step 5, the PTT computations. Without the tensor calculations, the interactive lighting will not work properly, so the primary focus will be to successfully complete this step by week 6 of the project. It is important to note that we will be testing and debugging throughout the project as the lighting calculations require a heavy amount of mathematical calculations which will require constant debugging and testing to ensure its proper functionality.

2.3.3 Description of any demos or tutorials

To show the full capabilities of the tool, we will include a readme file which will explain all of the steps needed to load the tool as a Maya plug-in and execute example commands to show its capabilities. In addition, we will include a quick mpeg video to go step by step through the tool and how to get it fully working in Maya. This will allow the user to follow along and see exactly how to take full advantage of the FICA tool.

3. WORK PLAN

3.1. Tasks

There are a total of 10 tasks needed for the completion of this tool. They are outlined below by section based on the release time of the task. We will both be working together on the various tasks, and the Gantt chart below shows the person who is primarily responsible for each task.

Instead of tackling our papers in parallel, we chose to combine resources and focus on the Aggregate groups first and the interactive lighting more towards the beta release of the tool. We chose this strategy to allow for a more interactive presentation on after the alpha release when we show our progress to the class and demo our GUI, which will have more functionality if we finish the aggregate functionality. The interactive lighting portion will provide very little information to display and show until the final steps are completed, and was thus set as our focus in the beta.

3.1.1 Alpha Version

Task 1: Graphical User Interface

The graphical user interface is the bridge between the artist and the functionality of the tool. It will allow the user to set constraints for the lighting calculations and the aggregate group as well as generate the final output. To generate the GUI we will use MEL to create the window and other components of the interface.

Task 2: Level of Detail Commands

The level-of-detail commands will call the Maya reduce and smooth commands from our GUI. These commands will be made readily available here because they are additional options that the artist may desire before creating the aggregate group. Both will consist of buttons which when pushed, will reduce or smooth the model by one level. While the smooth command does have attached options, the button for our GUI will use the standard options for the command and smooth the model mesh one level up. If additional reduction or smoothing is required, the button can simply be presses additional times. The level-of-detail commands will be implemented using simple MEL scripts to call the associated Maya commands

Task 3: BRDF Decomposition

In order to calculate the transfer tensor in a reasonable amount of time, the BRDF needs to be discretized into a third order tensor. The orders of the tensor are the three factors, the reflectance, the view and the light. Any point's BRDF can be quickly calculated using these tensors along with the BRDF's core.

Task 4: Aggregate Group Population

Aggregate group population is the actual execution to create the adjusted aggregate group. The algorithm for this population method will be based off of the experiments conducted by Ramanarayanan et al, and can be implemented using MEL. Based on the ratio provided by the user of the objects, we will adjust the ratio to reduce polygon counts while preserving the appearance of the original aggregate.

3.1.2 Beta Version

Task 5: Transferred Incident Radiance Computation

The transferred incident radiance (TIR) is the illumination that arrives at each viewed point. The calculation of the radiance after it bounces on a BRDF is calculated based on the BRDF, the source radiance and the viewing angle. The TIRs are divided into components based on the BRDFs they hit, in the specific order they hit. These quantities are then stored in a PTT.

Task 6: Direct Lighting

Direct lighting includes no light bounces, so this stage will allow for relatively quick computations. This will facilitate the resolution of any final bugs in the pre-computation states. Here they relighting equation will be written, combining the BRDF and TIR PTTS.

3.1.3 Final Version

Task 7: One-Bounce Interreflection

In order to see the true benefits of global illumination, indirect lighting is necessary. At this step we will add in the algorithm to compute the multiple bounces of the light throughout the scene.

Task 8: Specular Lobe Separation

To achieve accurate specular highlights with the original lighting calculations numerous BRDF basis terms need to be recorded, taking up time and memory. In order to maintain the accuracy and expedite the process Sun proposes dividing the BRDFs into multiple specular lobes and a base lambert like function. I will separate one lobe component, to achieve a more accurate representation.

Task 9: Integration / Finalizing Tool

We will allot a week at the end of the project to make final adjustments to the tool and ensure that all the functionality is correct. During this week we will ensure that the GUI is presentable and useable for our target audience, and finish up the

instructional videos and readme files needed to show its use. Any other unforeseen issues that arise will also be addressed during this step.

Task 10: Testing / Debugging

We include task 10 as the testing and debugging task which is carried on throughout the duration of the project. During each step we will be incrementally testing the tool and ensuring that the algorithms and calculations are correct. There are quite a few steps that require successful completion of the step before it, so it is vital that those steps are completed in the time we allotted in order to avoid setbacks and delays, which could cause to the incompletion of the tool.

3.2. Schedule

We have broken down our tasks in the form of a Gantt chart listed below. The first week is designated towards completing the design document and beginning work on the graphical user interface. The first two weeks will be spent getting to know MEL scripting and its use for graphical interfaces. In addition, we will begin work on the aggregate groups portion of the tool, focusing initially on the simpler requirements where we call Maya commands to execute to the level of detail solutions for the models.

The aggregate groups tool will give us a chance to get comfortable with the MEL scripting language and learn more about Maya plug-ins and its advanced capabilities. We will then begin the initial stages for developing the interactive lighting portion of the tool, which will be done in C++ as it requires a significant computational effort. Following the alpha release which is aimed primarily at completing the graphical portion of the tool, we will be leaning heavily towards the mathematical calculations needed to add the various components of interactive lighting, including the transferred incident radiance calculations, lighting, and specular lobe separation.

Gantt Chart

ID	Task	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10
0	Design Document										
1	Graphical User Interface										
2	Level of Detail Commands										
3	BRDF Decomposition										
4	Aggregate Group Population										
5	TIR Computation										
6	Direct Lighting										
7	One-Bounce Interreflection										
8	Specular Lobe Separation										
9	Integrating / Finalizing Tool										
10	Testing / Debugging										
		Alpha Version Release				Beta Version Release			Final Version Release		

Kevin Chang
Kendra Gibbons
Both

Works Cited

Ramanarayanan, G., Bala, K., and Ferwerda, J. 2008. Perception of Complex Aggregates. *Association for Computing Machinery*. 60: 1 – 10.

Xin, S., Zhou, K., Chen, Y., Lin, S., Shi, J., and Guo, B. 2007. Interactive Relighting with Dynamic BRDFs. *Association for Computing Machinery.* 27: 1 – 10.