PHOTOREALISTIC RENDERING OF SCIENTIFIC DATA

Joel Ogden[1], Jabari Jordan[1], Chelsey Krol[2], Tanya Papazian[3], Hans-Peter Bischof[1] and Reynold Bailey[1]

[1]*Department of Computer Science, Rochester Institute of Technology, Rochester, U.S.A.*

[2]*Department of Computer Science, Mount Holyoke College, South Hadley, U.S.A.*

[3]*Department of Computer Science, Montclair State University, Montclair, U.S.A.*

Keywords: Scientific Visualization, Maya, Spiegel.

Abstract: Generating photorealistic images of astrophysical simulations can enhance the experience of watching galactic visualizations for both the specialists who study the data and the average person who is simply interested in outer space. Unfortunately, the astrophysicist who is creating the simulations typically lacks the expertise required to generate photorealistic images. Likewise, a 3D artist may be unaware of the physics behind certain astrophysical events. We aim to use Spiegel, a user interface that controls the rendering of astrophysical data and Maya, a high end 3D animation program, to allow a non-artist to easily create renders of photorealistic images. Spiegel provides a user-friendly interface for controlling the creation of potentially complex rendering applications by individuals with little experience in computer programming. Since Spiegel's basic visualization capabilities are limited to simple primitives like points and lines, it was necessary to develop an additional program for Spiegel to interface with Maya's 3D rendering capabilities. This software interface is called Miegel. Using Miegel, the astrophysicist now has access to Maya's 3D rendering capabilities allowing them to create stunning visualizations of astrophysical phenomena. In addition, new artistic effects can be created with Maya in the form of presets, which can be integrated into the user's visualization with minimal knowledge of computer programming.

1 INTRODUCTION

Visualizations of scientific processes have always been a crucial part of education and understanding to the student and scientist alike. Being able to see something as opposed to reading it from a textbook allows for a multi-dimensional learning experience and a more dynamic understanding of complex subject matter. Unfortunately not all visualizations are as effective as they could be. This is because most scientists do not have the training or skills in modern 3D rendering techniques to produce compelling visuals. Spiegel (Bischof et al., 2006) was developed to bridge this gap by bringing tool that were previously only available to a 3D artist into the hands of the scientist. Spiegel is a software package that allows individuals with little or no 3D rendering or programming experience to create simulations of scientific processes based on data. The program in its basic form is however limited to how the data is displayed. Most visualizations are displayed as simple points, which take on the roles of stars or particles. Figure 2 shows an example rendering created using Spiegel's basic rendering capabilities. We created Miegel, a software interface that allows Spiegel to take advantage of the rendering and animation capabilities of Maya. By using our system, the user is able to create more realistic and appealing visualizations such as the one shown in Figure 1.

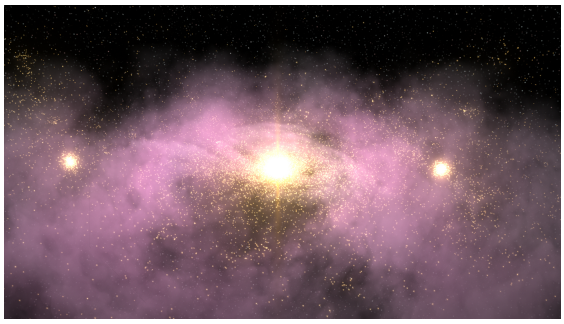

Figure 1: Photorealistic rendering of a black-hole merger created using our Spiegel-Maya interface called Miegel.

Ogden J., Jordan J., Krol C., Papazian T., Bischof H. and Bailey R..
PHOTOREALISTIC RENDERING OF SCIENTIFIC DATA.
DOI: 10.5220/0003851207190724
In *Proceedings of the International Conference on Computer Graphics Theory and Applications* (IVAPP-2012), pages 719-724
ISBN: 978-989-8565-02-0

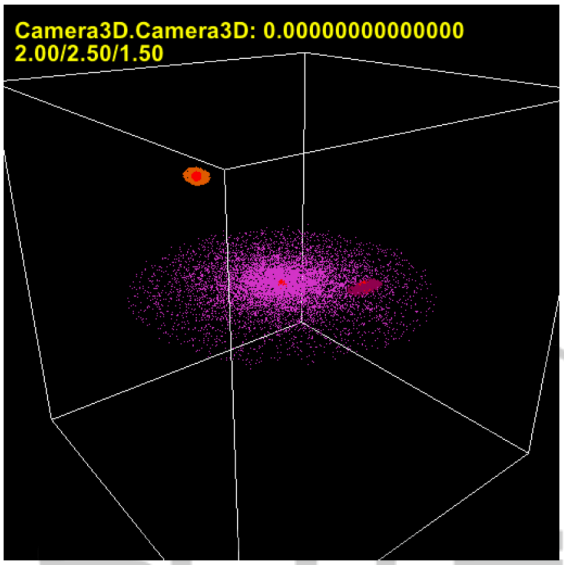

Figure 2: A visualization of a black-hole merger created using Spiegel's basic rendering capabilities. The stars are displayed as 3D points.

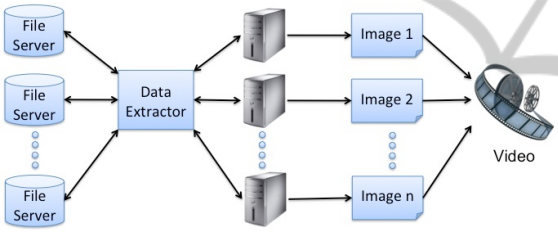

Figure 3: Overview of the Spiegel framework. Data is extracted from one or more file servers and distributed to a cluster of computers. Each processing unit in the cluster generates one (or more) frames of the complete visualization sequence in parallel. These frames are then combined to create a video. Image courtesy of (Espinal et al., 2010).

2 SPIEGEL

Spiegel is a software package that allows users with little experience in 3D digital graphics or computer programming to create visualizations based on raw simulation data. Spiegel makes this possible through a graphical user interface that allows users to combine components to create their desired visualization. Each component represents an individual program designed to perform a specific task. The Spiegel system is designed to be extensible by allowing new components and functionality to be added to the basic framework. This is possible because Spiegel is designed using a data flow architecture. Components have communication endpoints, which can be connected to form a complete visualization program. When the program is executed, data is passed from one component to another. Each component performs specific operations that contribute to the final result (Espinal et al., 2010). Communication between components is made possible by a variety of built-in data structures (Bischof and Dong, 2011). Data is taken in via an extractor and placed in these structures which are then passed between components as optional parameters. Figure 4 shows a simple visualization program created using Spiegel's graphical user interface.

Figure 4: Example of a program created in Spiegel that illustrates the data flow architecture. Image courtesy of (Espinal et al., 2010).

3 PREVIOUS WORK

Spiegel's basic rendering capabilities are provided using OpenGL primitives such as points and lines. As can be seen in Figure 2, the resulting renderings are useful but not necessarily compelling or realistic.

Several efforts have been undertaken to improve the quality of the renderings produced by the Spiegel system. One such effort expanded the visualization capabilities of Spiegel by using Pixar's PhotoRealistic RenderMan® (Cassidy et al., 2011). By using the shader library accessible through the RenderMan® program, the researchers were able to apply displacement shaders to better represent stars and other cosmic forms. They also used noise patterns and color palettes to develop realistic models of stars. A further advantage of RenderMan® was its ability to incorporate realistic lighting effects, which proved to produced far more compelling visualizations.

In addition, other work has been done to study the gaze patterns of expert and novice astrophysicists to gain insight into what makes a favorable or compelling visualization (Arita et al., 2011). By having participants view a number of images of galactic events, both photographed and computer generated, the researchers were able to gain a better understanding of preferred camera positions and zoom settings. This information was then used to modify existing Spiegel visualization programs to create more favorable renderings.

Despite these advances, the improved Spiegel framework had several major drawbacks. In particular, developing custom shaders for the RenderMan® component required significant programming expertise. Furthermore, the system still lacked the capa-

bility to handle complex animations and fluid simulations.

To overcome these limitations, we decided to use Maya because of its ability to produce compelling visualizations with minimal programming. Furthermore, Maya has a robust fluid simulation system and the ability to store artist created presets, which novices can easily access. This fluid system proved to be the ideal solution for simulating space dust and gases.

4 MAYA

Maya is a high-end 3D animation program capable of producing stunning visual graphics. A major shortcoming is that it takes a considerable amount of time and training in order for an individual to produce these images. Through the Miegel (i.e. Maya-Spiegel) interface that we created, the user can take preset files created by an expert in the use of Maya and insert them into the scene with relative ease. Using the controls provided in the Miegel interface, the user can also customize the visualization by simply adjusting various attributes.

In order to produce the visual effects of space dust and gases, we utilized Maya's fluid simulation system. Through the careful application of noise attributes and color gradients we were able to construct cloud shapes that were much like those seen in galactic imagery. These settings were then saved as presets, which allow for faster render times and greater control. These types of presets can easily be loaded and adjusted by a novice user to quickly create appealing visualizations.

Another powerful feature that Maya provides is the ability to map an image onto a fluid. This is done by using the color values of the image to represent specific densities in the fluid. Maya's fluid simulator can then use this information to produce the effect of vortices and cloud formations (Brinsmead, 2007). We were able to add further detail and realism to the fluid simulation through the use of photographs of various cloud formations made available through the National Oceanic and Atmospheric Administration, as well as other open source venues. This same technique can be applied for any photographed astrophysical phenomena, allowing for an added degree of dimensionality to the visualization. By utilized Maya's wide array of physically accurate modifiers, we were able to manipulate the fluid as if it were acted upon by forces found in space.

5 MIEGEL

Miegel is a software interface designed to integrate Maya's rendering capabilities with Spiegel. Miegel was created because the existing solutions provided by OpenGL and RenderMan® were too complicated for novices to use since they required significant programming experience. The process of adding new types of effects to the visualization pipeline, for example, required many hours of implementation, testing, and debugging. Miegel accomplished this by using the options available through Maya to produce stunning visualizations, which in tern require less programming hours. In addition, Miegel aimed to allow for visuals produced by a 3D graphics artist to be easily integrated into the visualization. With Miegel, the task of creating visual effects such as those seen in professional productions as well as the process of integrating them into the visualization is greatly simplified. This is done by the user interface in Miegel, which allows for visuals produced by a 3D graphics artist in Maya to be easily integrated into the visualization through a drop-down menu. The applicable attributes of the specific preset are then displayed for the user to modify, giving the user full control over the aesthetics of the finished rendered image.

Miegel works by producing a MEL (Maya Embedded Language) script and a data cache PDC (Particle Disk Cache) and passing them to Maya. The MEL script contains information about the objects Maya should create, such as color and luminance values for the given particles, as well as scripts to import additional scene files. The PDC is a raw file format used by Maya as a way of caching information about the particles. The cache maps attribute information about particles to disk allowing for faster drawing of particles, versus having to re-calculate the positions for each particle for each frame (Center, 2005). The PDC contains the vector data for the particles that represent the stars in the simulation. Maya can then render this out into a batch of image files, which can be at full HD resolution (in contrast with the previous solution which could only produce standard resolution images). The big advantage with this system is that Miegel automates most of the process and allows users unfamiliar with Maya to still take advantage of its powerful capabilities.

Figure 5 illustrates the Miegel architecture. A MEL script is a language provided by Maya that allows users to create macro programs to automate many of Maya's functions and operations. Because the Maya GUI is itself written in MEL, any operation that can be performed by a human using Maya can also be programmed in MEL. This means that a pro-

gram like Miegel that generates a MEL script can perform any action in Maya that is available to a user. The simplicity of MEL scripts mean that Miegel can easily be expanded to utilize any of Maya's features (Autodesk, 2010).

An example of a MEL script is

```
particle n star p 0 0 0;
setAttr star.prt 8;
setAttr particleCloud1.incandescence
type double3 1 1 1;
setAttr particleCloud1.glowIntensity 1;
```

These few lines instruct Maya to place a single particle named star at vector (0, 0, 0), convert the type of the particle to a cloud, then change the incandescence color to white with maximum glow. Once this single particle is created, it can be attached to the PDC that contains all of the particles (potentially tens of thousands) and an image that matches the data in the simulation can be rendered. Scripts such as the one shown above are automatically generated by Miegel and utilized by Maya to produce the resulting image.

Spiegel utilizes simulation files with the extension .sim. A sim file is a human readable file written in plain text, which contains information such as the number of stars in the simulation, the positions of the stars for each frame, as well as their velocity, acceleration and other information. Miegel creates a disk cache first be parsing out information from the sim file. For the purpose of the visualization, we only cared for the positions and frame number of the stars. To convert the .sim to PDC, Miegel makes two passes over the .sim file. First, the positions of the stars and their frame number are extracted, at the same time filling any gaps in the data. This information is then written as raw bytes into an .msf (Maya-Spiegel Format) file. From here, Miegel can quickly extract the data for any particular star in file from any frame. The benefits of having the .msf file is that the .sim file only has to be parsed once and any subsequent visualizations using the same data does not have to be run through the converter again. For reference, a 7GB .sim file took over an hour to be convert to msf. Other benefits of the .msf format include a much smaller file size and predictable data placement allowing for close to real-time access of information, this allows for frames deep in the simulation to be quickly extracted and used for the visualization.

After the MEL script is generated and the PDC completed, Maya can read the script and output a Maya scene file (.ma) which holds all the information needed by the renderer. The .ma file is the standard file format used by Maya that can be read and edited, allowing Maya users to verify the visualization before

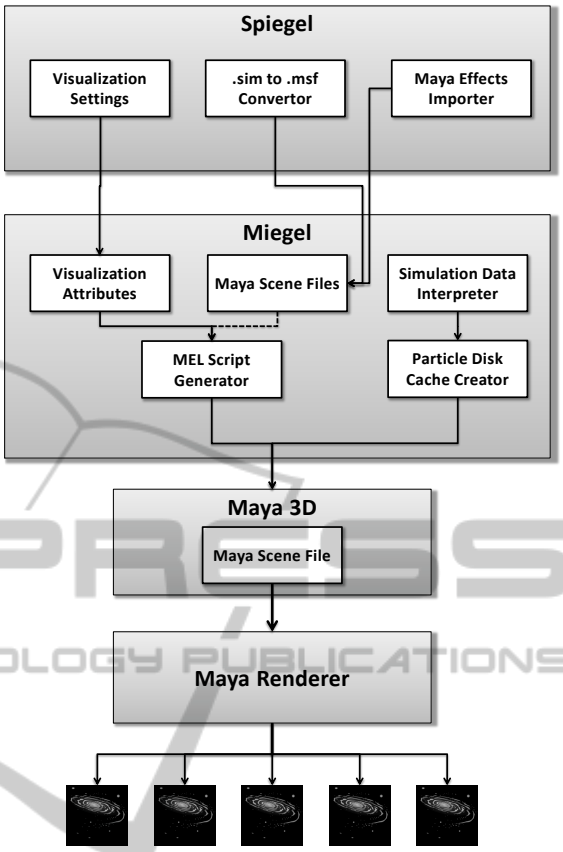

Figure 5: Architecture of Miegel. The Simulation Data Interpreter takes the position data of the stars, and creates the Particle Disk Cache, which is needed for Maya to map the location of the stars to the particles objects. The visualization attribute module takes in information from the user about the desired appearance of the stars (including color, size, and glow intensity) and produces a MEL script. The script can also include instructions to integrate other Maya files into the visualization. Once the script is generated, it is executed by Maya to produce the individual frames of the visualization.

rendering and import other Maya scene files. The .ma file is read by the Maya renderer to produce the final images. The render can be flagged with arguments to change the attributes of the outputted file, such as resolution, and what rendering techniques it should use, such as anti-aliasing. This provides other advantages over the OpenGL and RenderMan® solutions, where such features must be explicitly programmed.

Spiegel components were created to allow for easy access to Miegel from Spiegel. A user familiar with Spiegel can easily use the new Miegel component to access Maya. The user can import the required simulation file, then choose which frames in the simulation to render. Furthermore, they can import as many 3rd party Maya scenes that are available, having access to

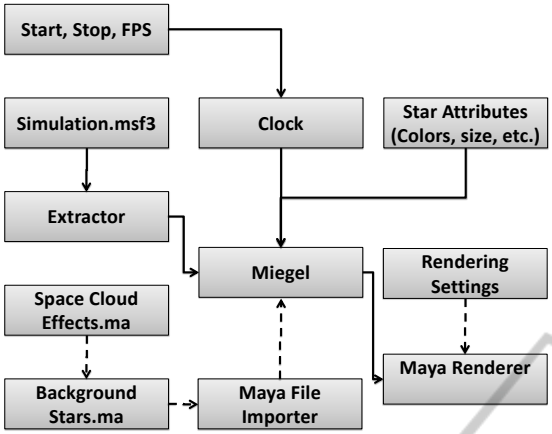

Figure 6: : The Spiegel components available to the user and how they interact to produce visualization.

easy to use sliders and drop down boxes to choose attributes, which change the look of the stars as well as the image output format, location, resolution, as well as other renderer attributes. Figure 6 illustrates how the information flows through Spiegel into Miegel and the renderer.

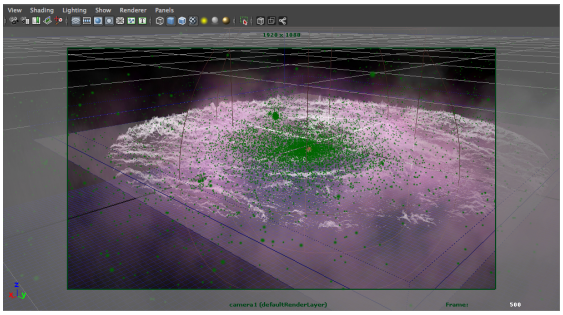

Figure 7: Maya viewport view of the simulation.

6 CONCLUSIONS AND FUTURE WORK

We have created a highly customizable and robust program called Miegel that is able to take simulation data in its raw form from Spiegel and send it to Maya to generate compelling visualizations. This allows the user to utilize the full gambit of Maya's capabilities in Spiegel in order to further enhance their visualization. With Miegel, Spiegel could be expanded to use professional grade animation programs to better its ability to create realistic and eye catching simulations. This provides the ability to add numerous special effects and/or programs to Spiegel in the future. In future work it would be possible for the user to use the integrated camera capabilities, allowing for nearly seamless incorporation of stereoscopic cinematography into scientific simulations. Maya also has sonification capabilities, allowing the user to coordinate sound with the aesthetics of the visualization, which would greatly enhance the viewing experience. Maya also allows the use of it's diverse library of shaders and the ability to create customizable shaders, giving even greater customization to the aesthetics of scene. In addition, Maya's newest release has the capability to render images using a GPU based rendering system that has the capability to display these images in the viewport in their rendered state. Figure 7 shows the viewport view in Maya of a simulation that was generated from our program Miegel. It is entirely feasible to integrate other animation and rendering programs into Spiegel, which will better enhance the simulations that are produced. The framework we have chosen allows for the addition of any preset the user may require, giving the program an endless potential for expansion that is only limited to one's imagination.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Award No. 0851743. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and not necessarily reflect the views of the National Science Foundation.

REFERENCES

Arita, J., Feliz, J., Rodriguez, D., Bischof, H. P., Rege, M., and Bailey, R. (2011). Creating audiencespecific galactic simulations using eye-tracking technology. In *in proceedings of International Conference on Information Visualization Theory and Applications*.

Autodesk, I. (2010). Maya api reference. http://download.autodesk.com/us/maya/2009help/API/main.htm.

Bischof, H.-P., Dale, E., and Peterson, T. (2006). Spiegel - a visualization framework for large and small scale systems. In Arabnia, H. R., editor, *Proceedings of the 2006 International Conference on Modeling, Simulation and Visualization Methods*, pages 199–205. CSREA Press.

Bischof, H.-P. and Dong, A. (2011). Directing a visualization ala kubrick. In *Proceedings of The 2011 International Conference on Modeling, Simulation and Visualization Methods*.

Brinsmead, D. (2007). Volume displacement method for clouds using fluids. http://area.autodesk.com/blogs/duncan/volume_displacement_method_for_clouds_using_fluids.

Cassidy, P., Kilburn, T., Salemink, V., Bailey, R., and Bischof, H.-P. (2011). Improving the visualization of galactic events using pixars renderman. In *Poster Papers Proceedings of 19th International Conference on Computer Graphics, Visualization, and Computer Vision, 2011*.

Center, S. D. S. (2005). Maya pdc format simplified. http://www.sdsc.edu/us/visservices/software/maya/pdc_table.html.

Espinal, J., Allen, V., Amable, K., Bailey, R., and Bischof, H.-P. (2010). Renderman's power to visualization's rescue. In *Communication Papers Proceedings of 18th International Conference on Computer Graphics*, pages 243–249.