RenderMan's Power to Visualization's Rescue

Julio Espinal
jae3161@rit.edu

Virginia Allen
vla8446@rit.edu

Kwesi Amable
kxa9006@rit.edu

Reynold Bailey
rjb@cs.rit.edu

Hans-Peter Bischof
hpb@cs.rit.edu

Department of Computer Science, Rochester Institute of Technology

ABSTRACT

Most visualization systems employ a data flow approach in order to create visual representations of data. The data flows along a directed graph through the different components, gets filtered, extracted, analyzed, and finally converted into an image. Most visualization systems use one graphic toolkit or library to create the image. These toolkits and libraries are not created equally; some are better suited than others to solve given problems. Being able to pick and choose would often generate a better result. Within the Spiegel framework any toolkit, which can be used in a Java environment, can be employed to create the image. In this paper, we explain the Spiegel framework and how Pixar's PhotoRealistic RenderMan® can be used to visualize scientific data.

Keywords
Visualization Framework, RenderMan®, Data Flow Languages.

1. INTRODUCTION

Most visualization systems employ a data flow approach along a directed graph to filter, extract, analyze, and finally convert data into an image. They generally use one specific, unchangeable graphic toolkit or library to create images. The features of these toolkits and libraries vary significantly; some are better suited than others to solve given problems. Simply changing from one toolkit or library to another often produces strikingly different results.

Additionally, visualization systems run on different hardware platforms, which use different drivers to access the graphics card. For example, OpenGL running on two different platforms, Mac OS X and Windows, using the same NVIDIA GeForce 9400 graphics processor will not execute all shaders in the same manner. As a result, the images generated from the same program may differ in quality.

Within the Spiegel framework [Bis05], any toolkit that can be used in a Java environment can be employed in order to create the best possible image. In this paper, we explain the Spiegel framework and how Pixar's PhotoRealistic RenderMan® can be used to visualize scientific data.

The rest of the paper is structured as follows: section 2 discusses general visualization principles; section 3 presents an overview of how a data flow architecture can be used for creating visualizations; a brief survey of related work is presented in section 4; Our approach for incorporating RenderMan® into the existing Spiegel visualization framework is described in sections 5, 6, 7, and 8. Finally, results and future work are presented in sections 9 and 10 respectively.

2. VISUALIZATIONS

Spiegel was designed as a visualization tool for the Center for Computational Relativity and Gravitation (CCRG) at Rochester Institute of Technology. Spiegel has been used mainly to visualize simulations of galactic events like black hole mergers, gravitational waves, and galaxy mergers. However, it can be used to visualize any type of data. For CCRG, the visualizations created by Spiegel are used to help debug and understand the simulations from which they are generated, as well as explain the science to the general public.

Certain galactic events like black hole mergers cannot be observed in practice. Therefore, a visualization of a black hole merger cannot be compared to a photograph. This makes it relatively easy to generate visualizations because it is not bound to a specific pre-conceived image. On the other hand the Hubble Space Telescope took images of nebulas like the one shown in Figure 2. It is difficult to accurately generate this scene in 3D on a computer.

A typical simulation writes the current state of the model into a file at discrete moments in time. The visualization of scientific data always follows the same rules. The state at successive time steps of the simulation data is read and subsequently converted to a visual representation.

This can be done for all time steps in parallel if the data of each time step is complete and independent of other time steps. If this is not the case, the data can always be pre-processed. Because of this, visualization systems are excellent candidates for execution on a cluster. The Spiegel framework is no exception. As shown in Figure 1, individual frames of the visualization can be generated in parallel thus reducing execution time.

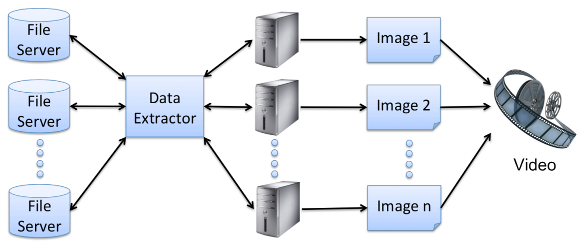

Figure 1: Overview of the Spiegel framework. Data is extracted from one or more file severs and distributed to a cluster of computers. Each processing unit in the cluster generates one (or more) frames of the complete visualization sequence in parallel. These frames are then combined to create a video.

Prior to this work, the Spiegel framework utilized only Java3D or JOGL to create 3D images. However, these libraries were limited to rendering simplistic models, which in some cases, results in unattractive images for a general audience. Java3D and JOGL cannot be pushed to render extremely difficult visual scenes. For example, it is impossible with either library to generate an impressive looking nebula, like the Cat's Eye Nebula shown in Figure 2. The authors do not argue that an image like the Cat's Eye Nebula cannot be generated on a computer, but they argue that Java3D or JOGL are not the right tools with which to solve this problem.

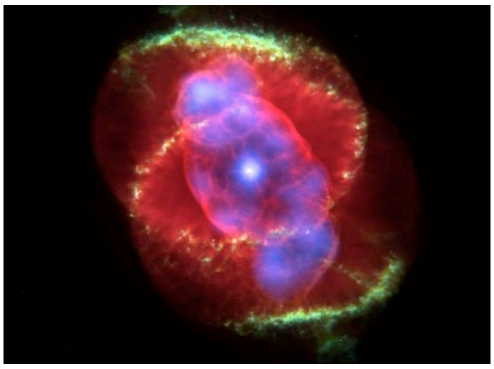

Figure 2: Cat's Eye Nebula. Courtesy of NASA and the European Space Agency. Image generated by the Hubble Space Telescope.

Pixar's PhotoRealistic RenderMan® has been widely used in the computer graphics community for over two decades to create stunning computer generated imagery. RenderMan's reputation has grown over the years and it is still used today to render scenes in many big-budget Hollywood films. Because of its power, huge benefits can be gained by incorporating RenderMan® into existing visualization systems.

3. DATA FLOW ARCHITECTURE

Most current visualization systems utilize a data flow architecture [Bis09]. Components have communication endpoints, which can be connected to form a 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.

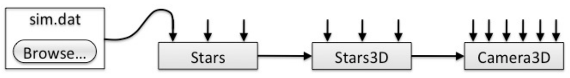

Figure 3: Example of a program created in Spiegel that illustrates the data flow architecture.

Figure 3 shows an example of a program created in Spiegel using its graphical interface. The node Stars reads the file named sim.dat specified as an argument and sends the data to the node Stars3D, and from there the data is sent to the last node in the graph, Camera3D. The Stars, Stars3D, and Camera3D components are simply small programs, which perform specific operations on the data.

Most visualization frameworks, like Iris Explorer, the grandfather of all visualization systems [Fou95], do not expose this functionality to the user. Vish [Ben07] and Spiegel [Bis09] are frameworks that expose this functionality to the user; consequently, they are very easy to extend.

The Unix operating system [Rit74] allows one to create a data flow architecture using pipes. This allows for the connection of multiple simple programs to create powerful systems. But more than this, it fosters the reuse of existing components. This increases the productivity of a developer. A Unix program like:

```
   sort file | uniq | sort –n  head -5
```

will print out the 5 most often occurrences of the same line in file.

Vish and Spiegel follow the same philosophy as Unix. In the end, this allows for the use of any tool/library that can convert data into an image.

The authors explored OpenGL, JOGL, and PhotoRealistic RenderMan® within the Spiegel visualization framework.

4. RELATED WORK

Other options besides RenderMan® exist for rendering realistic images. These include OpenGL with GLSL shaders and DirectX. Because Spiegel was designed to be platform independent, DirectX was an impractical choice because it is not fully supported on all platforms. Additionally, not every extension in OpenGL is supported on all graphics cards. These factors led us to consider RenderMan®. Due to the intuitive shading language and film-quality rendering, RenderMan® is superior to OpenGL. Even though it takes more time to render an image, the quality is significantly better and appeals to a general audience. The RenderMan® Interface is well documented and its reputation has been proven in the field for over twenty years. RenderMan® automatically performs many calculations that need to be performed manually in OpenGL. For example, with lighting enabled, the normal and view vectors are automatically calculated. Furthermore, setting up the camera and the scene is easier compared to OpenGL. The RenderMan® standard defines five types of shaders: surface, light, volume, imager and displacement; on the other hand, GLSL only supports vertex and fragment shaders. RenderMan's shaders have a very modular design; therefore, it is possible to edit certain parts of the pipeline without affecting other aspects. It is also possible to have multiple variations of a base shader, which facilitates the evaluation of the effects. Scene setup is also easier in RenderMan® as parameters can be added to a RIB (RenderMan® Interface Bytestream) file to guide scene generation as opposed to explicitly defining the scene in OpenGL.

5. RENDERMAN®

As stated previously, part of RenderMan's appeal is its modular design and multiple shader types. They can also be layered together to create unique textures. Once a shader is compiled, it can be used in any RenderMan® Interface Bytestream (RIB) file. A RIB file describes the environment and the various objects within a scene. RIB files can reference other RIB files in order to add existing objects to other scenes.

In many cases, the data set requires much processing time to produce a movie. The processing time increases drastically when rendering high-quality, photorealistic scenes. The Spiegel framework allows for distribution over a cluster, as shown in Figure 1, to generate the images in parallel, which reduces execution time.

Spiegel splits the RenderMan® interface into several components. These components include lighting, shader extractor, RIB generator, and camera settings. Because RenderMan® is very flexible, it is possible to have multiple instances of most of the components. As data flows through each component, a RIB file depicting the scene is generated and ultimately processed by the PhotoRealistic RenderMan® renderer to produce the desired image.

6. ARCHITECTURE

A modeling application is used to create and compile the RIB file. During compilation, the modeling application will parse each line of the RIB file and call the corresponding RenderMan® Interface (RI) routine. Once all of the information is gathered, RenderMan® will then bound and split each primitive. Figure 4 illustrates all the phases involved in the architecture.

During the bound and split phase, each primitive is checked whether or not it is within the bounding box. The bounding box is the viewing area in which the scene will be depicted. It is based on the current location of the camera and the size of the screen. If an entire primitive is not within the bounding box, then it is discarded; however, if a primitive is partially in the bounding box, then it is split. When a primitive is split this means that it is made into smaller polygons until a single one can fit into the bounding area. This can be seen in Figure 5 when a sphere is split into smaller polygons that create the whole sphere. Once the smaller polygons of the primitive fit into the bounding box, the polygons that are still not within the bounding box are discarded.

Once each primitive is bound and split, they are diced into a grid of micro-polygons. These micro-polygons will be small enough to approximately represent a pixel on the screen. As seen in Figure 5, these grids will allow for the shaders to manipulate the primitives. The first shader applied, if one is specified, is the displacement shader. These shaders need to be applied first because they manipulate the vertices' data, such as the position or normals, and this information is a basis for other shaders. Once the displacement shader is applied, the surface shaders are used next to manipulate the surface of the primitive. In order to apply the surface shaders, the lighting also needs to be taken into account to produce appropriate shadows. The location of the lights also needs to be considered, because if a light is directed towards a primitive, then the surface shader needs to adjust the color according to the type of surface and make that area brighter than the rest of the object's surface. Last, the atmosphere shader is applied in order to make changes to the primitive's color along with its opacity. After the objects are bounded and split, diced, and shaded, the image is rendered and displayed onto the screen.

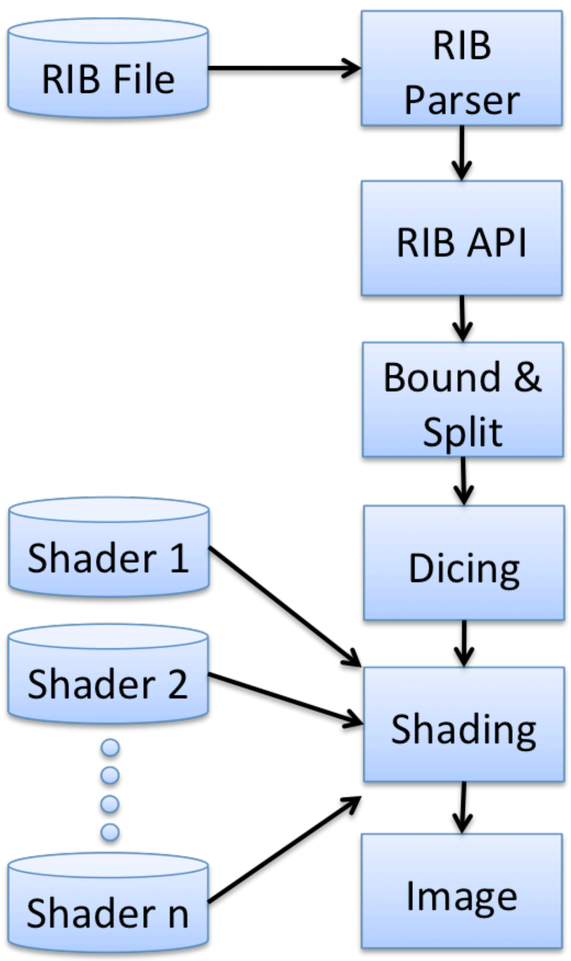

Figure 4: The stages involved in the RenderMan® architecture.

7. RENDERMAN® PROGRAM

Consider the example of trying to render stars in a galaxy. The following snippet of code shows part of the RIB file that is generated:

```
...
TransformBegin
   Translate -0.1 0.6 -0.3
   Scale 0.1 0.1 0.1
   Color [0.46 0.46 0.4]
   Surface "glow"
"attenuation" "2"
   Sphere 1 -1 1 360
TransformEnd
TransformBegin
  Translate 0.1 -0.3 0.4
  Scale 0.2 0.2 0.2
  Color [0.0, 0.0, 0.0]
  Sphere 1 -1 1 360
TransformEnd
...
```

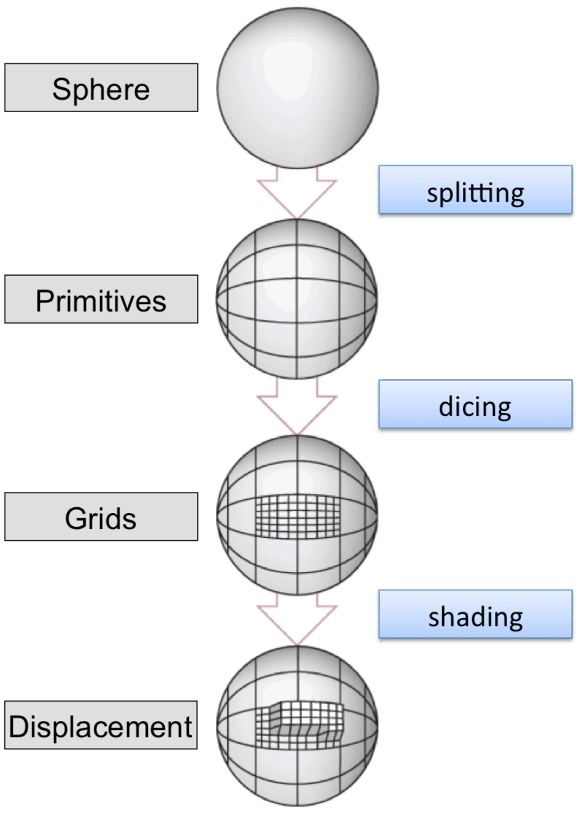

Figure 5: Illustration of a sphere being split, diced, and shaded. Image adapted from renderman.pixar.com [Pix09].

The first piece of information, between the first TransformBegin/End, describes the characteristics of one star within the image. The surface of a star is described by using a shader called "glow"; this can be seen on the "Surface" line. The second piece of information is for a black hole. The scale of a black hole is slightly larger than the stars and the color is black.

8. SHADERS

The key to generating realistic images from RenderMan® is shaders. A shader is a function written in the RenderMan® shading language that calculates the color and position of a point on the surface of the object. The RenderMan® plug-in for Spiegel allows the user to select the shader from a file. The program will parse the header of the shader file to determine the parameters it takes. It will then dynamically add an input to Spiegel's shader module for each of these parameters. This module contains the name of the shader and a list of variables with their values. The camera module generates the main RIB file. It imports the previously generated RIB file that contains the models. After generating the RIB file, the RenderMan® renderer (*prman*) is invoked to produce an image. This process is illustrated in

Figure 6. The module gets the camera position, image size, and the render quality in as parameters. It also has parameters for information about the interpolation and the motion blur. To create an interpolated movie, the program reads each time step until it has four time steps, it will then render all of the frames that should go in between these four time steps.

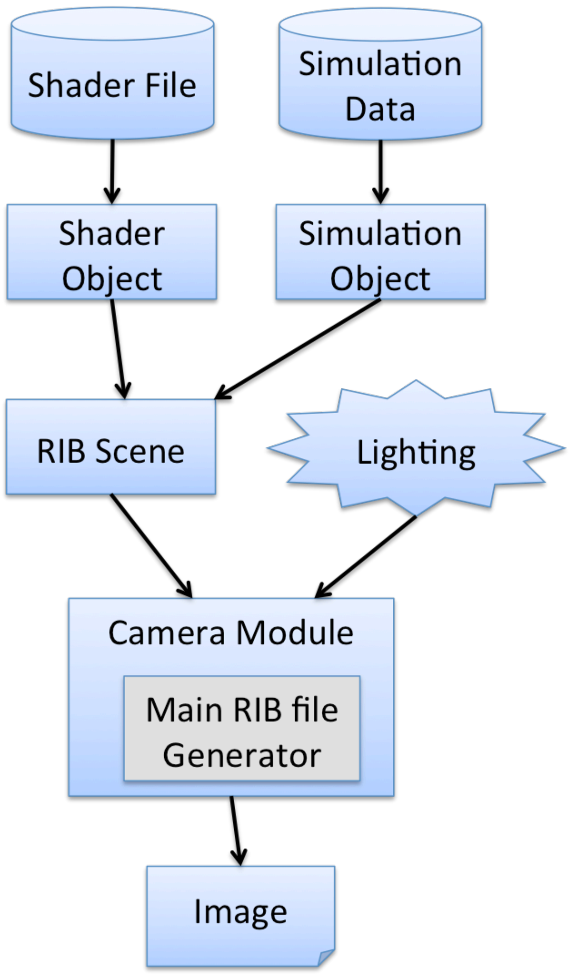

Figure 6: Illustration of how the main RIB file is built and used.

There are several Spiegel modules for RenderMan® lighting. These modules add support for ambient, distant, spot, and point lights. To add a light, connect the light module to the RenderMan® camera module. The "lights" input supports the connection of multiple lights at the same time. The parameters of the lights can be changed via Spiegel's interface. These parameters include light intensity and color along with others depending on the type of light. It is important to note that some shaders, do not use lighting to determine how to render the objects. This means that, when using these shaders, adding lights will have no effect on the final image.

9. RESULTS

The Spiegel framework was used to create video clips of black hole mergers for the show "The Universe: Cosmic Holes" which aired on the History Channel in 2008. The videos were rendered using OpenGL and depicted black holes as simple Gouraud shaded spheres against a static texture mapped background. Figure 7 (left) shows a single frame of a three black hole merger that was rendered using the old Spiegel/OpenGL approach. Figure 7 (right) shows an image that was rendered using the new Spiegel/RenderMan framework. In this case, the individual stars surrounding the central black hole are rendered using a shader which gives a more realistic glowing effect.

We generated images based on a simulation of a three-galaxy merger. Figure 8 shows one frame of the merger viewed from the side and Figure 9 shows the merger viewed from the top. For these images, a different shader which emphasizes the appearance of the back holes was used.

10. FUTURE WORK

RenderMan has been successfully incorporated into the Spiegel visualization framework and has been used to create visualizations of galactic events such as black hole mergers. The new framework allows for distribution over a cluster. This was successfully verified for a small cluster. In the future, we will have access to Blue Waters [NCS09]. Blue Waters will consist of 100,000 nodes and the peak performance will be in the Peta-flop range. The Spiegel framework will be ported to this cluster and its scalability will be analyzed.

Sonification [Her05], the art of representing data by using sound, is a rapidly evolving area of research. We plan to explore various approaches for using sonification models to further enhance our visualizations.

Many visualization algorithms are designed to visualize a very specialized problem. Unfortunately these algorithms cannot be used outside the tool in which they are implemented. A language named *Sprache* is used to describe a visualization program in Spiegel [Bis05]. However, it is not well suited for working with data that is distributed over multiple servers. We plan to redesign this language to handle distributed data and distributed rendering for the new Spiegel/RenderMan framework.

Finally, one of the major limitations of our project was the time it took to render images. The use of a cluster to render individual frames in parallel helps to reduce the overall rendering time for a video sequence, however each individual frame could potentially take a long time. Although PhotoRealistic

RenderMan is an efficient software renderer, it is still subject to long processing times for complex scenes. We plan to explore the use of multi-core GPUs to speed up the rendering time.

11. ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation under Award No. CCF-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.

The authors would also like to thank to Manuela Campanelli, Carlos Lousto and Yosef Zlowocher from the Center for Computational Relativity and Gravitation for their help and fruitful discussions. This team provided the data used and the interpretation of the visual representation.

12. REFERENCES

[Ant00] A. A., & Larry, G. (2000). Basic Geometric Pipeline. In Advanced RenderMan: creating CGI for motion pictures (pp. 136-143). San Diego, CA: Academic Press.

[Ben07] Werner Benger and Georg Ritter and René Heinzl, The Concepts of VISH, 4th High-End Visualization Workshop, Obergurgl, Tyrol, Austria, June 18-21, 2007, 978-3-86541-216-4.

[Bis05] Hans-Peter Bischof, Jonathan Coles: A Movie Is Worth More Than a Million Data Points, Lecture Notes in Computer Science Publisher: Springer-Verlag GmbH, ISSN: 0302-9743 Subject: Computer Science Volume 3514/2005, Title: Computational Science ICCS 2005: 5th International Conference, Atlanta, GA, USA, May 22-25, 2005

[Bis09] Hans-Peter Bischof, Swathi Annamala: The KISS Principle Applied to Dataflow Languages Paradigms for Visualization Frameworks, Proceedings of the 2009 Conference on Modeling Simulations and Visualization Methods, p. 48-55, ISBN: 1-601320-120-1.

[Fou95] David Foulser. Iris explorer: a framework for investigation. SIGGRAPH Computer Graphic, 29(2):13{16, 1995.

[Her05] Thomas Hermann, Andy Hunt, "Guest Editors' Introduction: An Introduction to Interactive Sonification," IEEE MultiMedia, vol. 12, no. 2, pp. 20-24, Apr. 2005, doi:10.1109/MMUL.2005.26.

[NCS09] Blue Waters Announcement. Retrieved October 20, 2009, from NCSA's website http://www.ncsa.illinois.edu/BlueWaters.

[Pix09] Pixar's RenderMan Performance. (2009). Retrieved November 30, 2009, from Pixar website: https://renderman.pixar.com/products/whats_renderman/2.html

[Rit74] D. M. Ritchie and K. Thompson. The Unix time-sharing system. Communications of the ACM, 17:365-375, 1974.

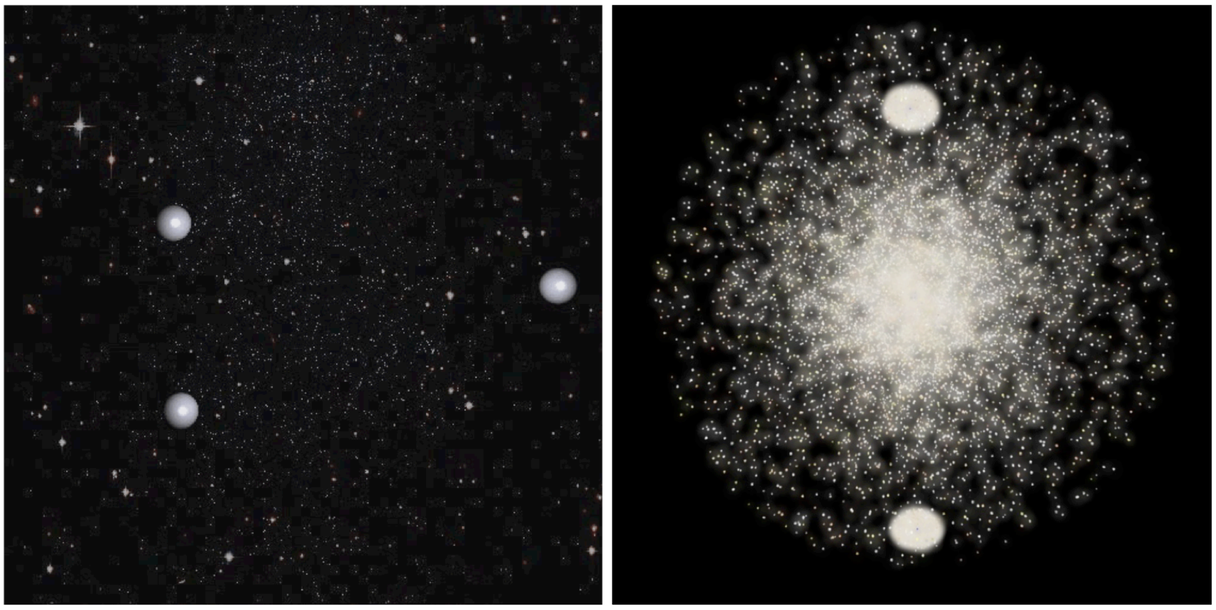

Figure 7: Image rendered using old Spiegel/OpenGL framework (left). Image rendered using new Spiegel/RenderMan® framework using shaders.

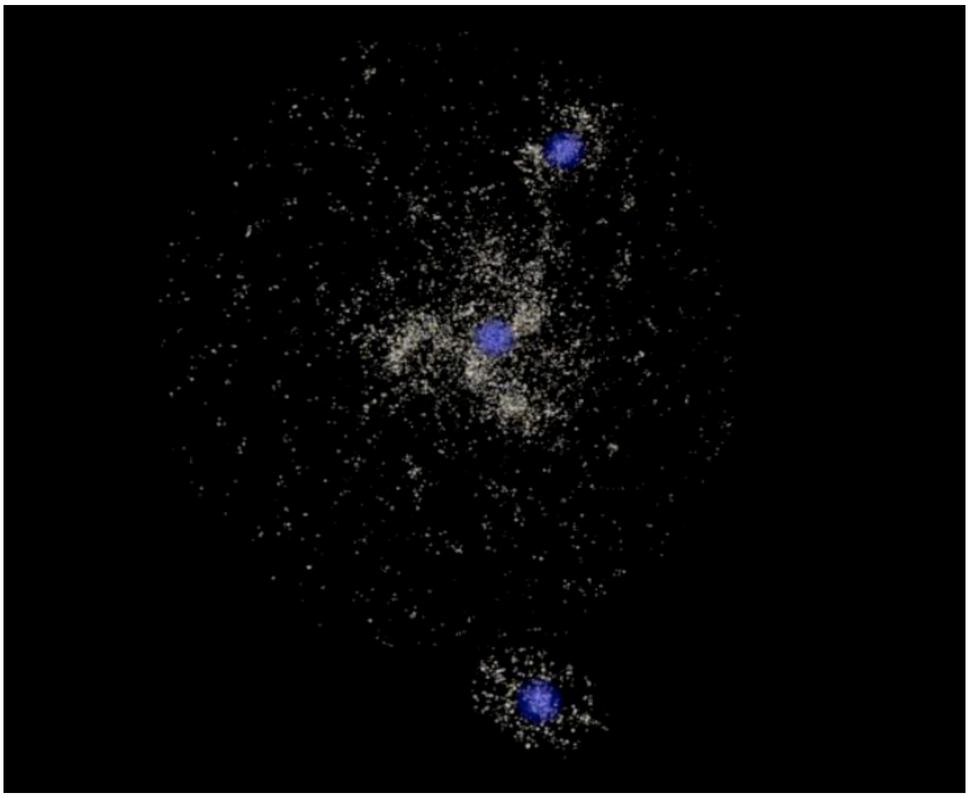

Figure 8: One frame from a three-galaxy merger viewed from the side. Image created by the Spiegel Visualization System using RenderMan®.

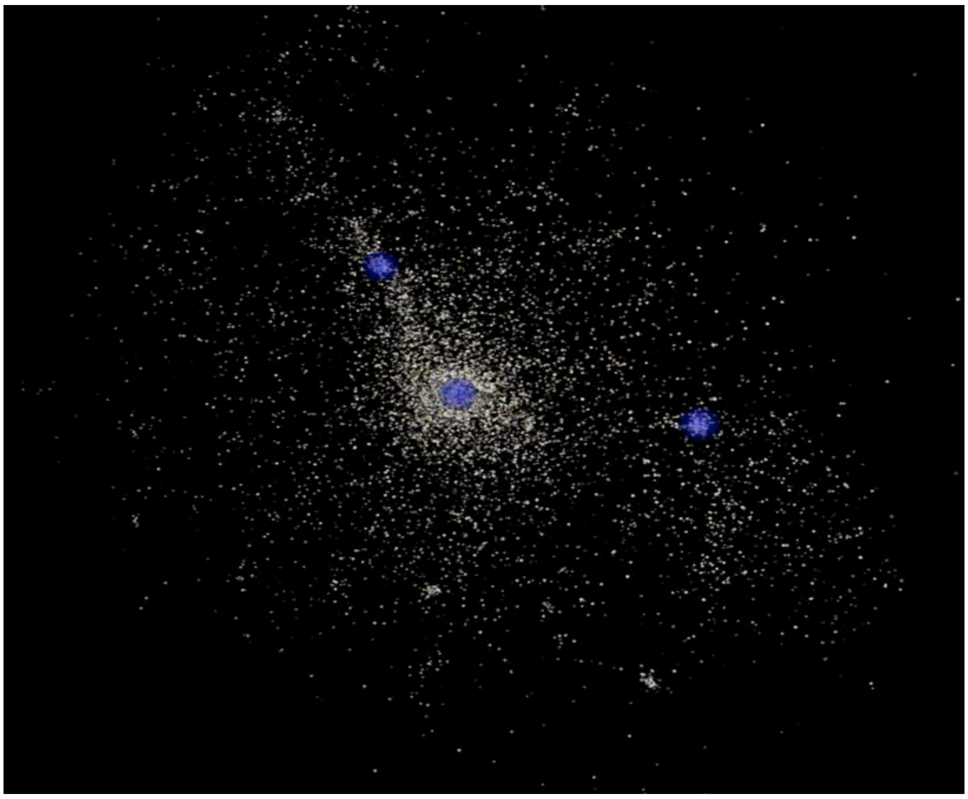

Figure 9: One frame from a three-galaxy merger viewed from the top. Image created by the Spiegel Visualization System using RenderMan®.