A Comparison of Rendering Techniques for Scenes with High Geometric Complexity

Devon Penney

Brown University, Department of Computer Science

1 Introduction

1.1 Abstract

We present a study that evaluates the effectiveness of using global illumination, texture, and OpenGL rendering for scenes with complex geometry, such as DTI streamtube models. Global illumination offers several advantages over OpenGL such as shadows, interreflection, and high quality shading models. It is now possible to harness this power in real time using methods such as Pre-computed Illumination for Isosurfaces[1] (Banks, et al). This type of rendering offers a much more realistic viewing environment compared to standard OpenGL shading and lighting.

Study participants are presented with a series of spatial analysis tasks consisting of following tubes, analyzing tumor penetration, and comparing relative depths of tubes. We render image sequences with Maya's Mental Ray plugin and the Maya Hardware renderer. We also test the use of texture mapping on tubes, and the effects of motion parallax.

We expect participants to find the OpenGL renders just as effective as the global illumination, since the differences between the two are subtle. Also, texture will likely help to define tube direction, so we expect to see better performance in follow tasks in this case. Finally, we believe performance will be significantly better in the motion case rather than still frames.

1.2 Previous Work

There are several studies that address similar issues through user studies. First, in Real-time Illustration of Vascular Structures[3] (Ritter, et al.), they compared GPU-based NPR methods with more standard ways of rendering vascular trees. They found that their "shadow like" depth indicators were useful as depth cues. Also, Visual Glue[4] (Thompson, et al.) analyzed how shadow and interreflection affected depth perception. The hypothesis was that these factors "glue" objects to the surfaces they are touching. They found that both the cues, or just one significantly improved depth perception.

2 Experiment Design

2.1 Overview

We display sequences of tasks to participants, each of which involves spatial analysis. Each task is rendering either with global illumination or OpenGL, with texture or without, and with or without rotational interaction. There are three primary task types. First is tube following where the endpoint for a tube is marked, and the user must click on the other endpoint. Next is tube occlusion, where users must determine which tube is in front of the other. Finally, we have tumor penetration, where the user determines how a tumor intersects tubes.

2.2 Rendering

There are two primary rendering algorithms used. First, we use fixed-pipeline OpenGL, which incorporates per-vertex lighting and gourraud shading. We implement this using Maya's hardware renderer. Next, for Global Illumination, we use Mental Ray, which is a popular photon mapping implementation distributed with Maya. In this case, we use three million photons, and high settings for shadow sampling. Texture mapping is done for both cases as well using a diamond-shaped pattern for the tubes.

2.3 Tasks

2.3.1 Tube Following

For this task, participants follow a tube marked with a blue sphere on one endpoint. They click on the screen where they believe the other endpoint is. We deduce this into three cases. First is when the two endpoints are 0-33 percent of the screen away from each other. The next is 34-66, and finall 67-100 percent. This way, we can analyze the relationship of distance between endpoints and time/accuracy to complete the task.

2.3.2 Tube Occlusion

Users are presented with a blue and green tube embedded in a dataset. They must decide which one is closer to their viewpoint, averaged over all rotations. This is broken down into cases where the relative screenspace lengths of the tubes are either with the top one larger, bottom one larger, or approximately the same size.

2.3.3 Tumor Penetration

A blue sphere is embedded in the dataset, and users must describe how it interacts with the tubes. First, we have the case where the tumor fully intersects the tubes. Next, the tumor can not intersect any tubes. Finally, the tumor can intersect only tangentially where it does not fully penetrate the tube. Participants answer which of the three cases is depicted.

2.4 Analysis

We track the time to complete the task, and the answer to the question. Also, for each task, we ask which cues they think they used. This is chosen from dataset motion, tube texture, tube size, tube color, shadows, context, occlusion, and shading.

2.5 Format

We used a Dell 3007WFP monitor, running at a resolution of 2560x1600. Images were rendered at 1600x1600, and sequences consisted of 23 frames with the dataset rotating from -5 to +5 degrees.

We created 54 tasks for the study. For each of the three task types, we used three different sizes (10, 16, 28), with a constant density of 20-21. For each size, we rendered two examples for each of the three per-task cases as listed in the previous section.

We used 48 tasks for the study; 12 follow tasks at sizes small and large, 16 occlusion (small, medium and large), and 16 tumor (small, medium and large). This was to facilitate easy blocking with the 8 combinations of variables. We merely took a pair of task type and size as a block. Next, the blocks are assigned a render style using an 8x8 latin square. Finally, the task order is specified per-participant by a 48x48 randomized latin square to ensure no two people see the same two tasks in a row. Participants were gathered from the study, faculty, and staff of Brown University.

3 Task Generation

3.1 Overview

Task generation is a two step process. First we must capture the tube geometry and task properties. Then, we must generate the global illumination and OpenGL renders for the task. I use a modified version of brainapp to generate the geometry by constraining the tasks to be sub-volumes of a DTI dataset. To render, I use Maya and Mental Ray by importing my geometry with a custom Mel/Maya API loader. The rendering pipeline is similar to an animation studio with lighting, shading, layout and render monitoring components.

3.2 Geometry Acquisition

Task generation starts with a streamtube model generated from a DTI dataset. This is the basis for the underlying geometry for the individual tasks. To start, I generate cube-shaped sub-volumes. This is done by finding the intersecting components of a box with the triangles of the tube model stored in a Kd-tree. We constrain the box to have a certain size and "density" of tubes. Density is calculated heuristically by analyzing the total number of triangles in the box. This is appropriate since the individual tubes are composed of cylinders. Thus, we can find a bound on the volume of the tube. The Kd-tree intersection process destroys tube caps and leaves ragged edges on the geometry. This is fixed by generating caps for the geometry, and completing the tubes so each tube is manifold. Lastly, we generate texture coordinates for the tubes in a way similar to the striping project at Brown.

Each task must also be generated at this step. This involves choosing tubes for the follow and occlusion tasks, and placing a tumor in the tube model. For the follow task, I randomly select a tube and its two endpoints. I save the screenspace distance between the two tubes and the screenspace location of the endpoints in each frame for the rotation sequence. We constrain that both endpoints are visible in the sequence. The occlusion tubes are selected at random as well, and are evaluated as follows. They either are both the same length (in screen space), the top one is longer or the top one is shorter. We constrain that both tubes do not intersect or cross in the images, and are mostly visible. Finally, for the tumor case, we generate a random sphere with a constant radius and place it within the volume. It is categorized as either not intersecting any tubes, only intersecting tangentially (ie: never fully penetrating a tube), or fully intersecting (tubes go all the way through the tumor). We constrain that the tumor must be visible and not free floating in front of all the tubes.

3.3 Render Setup

Next, we have the process of bringing the data into Maya for rendering. I save the geometry as a obj file, and also have a metadata specification file for task-specific information like the tumor size and radius. I have a custom MEL/Maya API loader which has a multi-step process. First, it loads the obj file, and task metadata. Next, it loads a lighting and camera rig, and sets up the task for viewing. Finally, it saves four Maya ascii files for the 4 different rendering

combinations (GL, GI, GL/Texture, GI/Texture). Lastly, a Maya batch file is generated for easy rendering.

3.4 Image Synthesis

The final step is the image synthesis. First, I aggregate all of the render scripts from the previous step and split them between all the computers at my disposal. Next, I start the render on said machines, and monitor it. The monitoring software takes care of converting the raw TIF images to G3D friendly pngs. Also, it keeps track of all the frames that are needed to finish the render job, and generates a Maya batch file in case computers crash.

Once we have our images rendered, we are almost done with the task synthesis. First, it is very common for there to be missing or malformed frames due to bugs in Maya or other. Thus, I pick out the bad frames, and render everything that is missing in the same fashion as above. If all is done correctly, this should result in a nice set of images for the study. Using this method, I generated 54 tasks, with a total of 4968 frames. The rendering took 22 computers approximately 10 days to complete.

4 Experiment Software

4.1 Overview

Once we have a nice set of rendered tasks, we must load them correctly into the software for showing images to the participants. First, we generate scripts with task ordering. Next, the images are displayed in the viewing environment, with options for answering the tasks.

4.2 Participant Scripts

In this stage, we generate the participant-specific scripts. Each defines an ordering of the tasks and for each, a rendering/viewing style.

4.3 Display Environment

The participant script is loaded into the display software, and the participant proceeds one task at a time. They answer tasks by either clicking on the image (in the case of the tube following task), or buttons for choice tasks like tube occlusion and tumor penetration. After each task, the participant is asked to list the cues they found useful for determining their answer. State data is saved to a file for each participant, which is aggregated to perform statistical analysis.

5 Discussion

5.1 Predictions

While the study has yet to be conducted, there are several predictions. First, there is doubt with respect to the usefulness of global illumination in this context, especially considering the computational complexity of rendering such images. Next, cases with no interaction will likely have much worse performance compared to with interaction.

5.2 Work Left

The study will be completed during the period of June 26-June 30th. We are currently in the phase of tweaking testing software, recruiting participants, and developing questionnaires. Statistical analysis and final writeup will be done sometime in August.

5.3 Future Work

It would be interesting to repeat the experiment with different render settings and a huge render cluster to facilitate easy image synthesis. Also, comparing the lighting rig used for this experiment vs. traditional lighting could be fruitful. This is because many 3D applications do not design lighting and use default headlight-type setups. Using global illumination may not be the best way to address visualization problems with high geometric complexity like DTI imaging. However, we can apply the same novel experimental design to studies with similar interests.

6 Works Citated

1 - KM Beason, J Grant, DC **Banks**, B Futch, MY Hussaini - Proceedings of SPIE, 2006
3 - F Ritter, C Hansen, V Dicken, O Konrad, B Preim, … - IEEE Transactions on Visualization and Computer Graphics, 2006
4 - WB Thompson, P Shirley, B Smits, DJ Kersten, C … - University of Utah Technical Report UUCS-98-007, March, 1998