CS419 Final Project

Nishant Bhushan, Richard Gunter, Huizi \_\_\_

**Abstract**— Features implemented for the ray tracer are: Motion Blur, Volumetric Rendering and Image-based Texture Mapping. Voumetric Rendering is a set of techniques used to render a 2-D projection of a 3-D discretely sampled data set . Motion Blur enables one to render images of objects in motion. Texture Mapping is a technique used to map a texture onto an image using uv-coordinates fo that image.

**Index Terms**—Ray Tracer, Volumetric Rendering, Motion Blur, Tecture Mapping



**Implementation (TODO)**

The Ray Tracer is implemented in C/C++. The hardware information of the machine used to render each of the final images is given below:

1. Motion Blur: \_\_\_
2. Volumetric Rendering: \_\_\_
3. Image-based Texture Mapping: \_\_\_

The Operating System (OS) of the machine used to render each of the final images is given below:

1. Motion Blur: \_\_\_
2. Volumetric Rendering: \_\_\_
3. Image-based Texture Mapping: \_\_\_

# Motion Blur (TODO)

Motion Blur is an effect that occurs when the image being rendered changes during the recording of a single exposure. This can occur if the object or camera is in motion and if the length of time of the exposure is sufficiently long. It gives a sense of realism and high speed.

Motion blur for a physical camera happens because the “shutter” (exposure) is open for a certain amount of time. Therefore, when a camera creates an image, that image represents a scene over a period of time, not at a single instance of time. The exposure period is usually small enough so that the image rendered appears to capture an instantaneous moment. However, a fast-moving object or a longer exposure time can result in artifacts that are blurred along the direction of relative motion. As objects in a scene move, an image of that scene must represent an integration of all positions of those objects, as well as the camera's viewpoint, over the period of exposure determined by the shutter speed. This was implemented by allowing the ray class to accommodate rays that exist at a specific time and by passing the value of time in the constructor. After that, we updated the camera class to store information about the shutter open and shutter close time. This allowed us to create rays for a random time in that interval and then cast those rays into the scene. A new *moving\_sphere* class was added to be able to calculate the hit point of a moving sphere. The difference between this class and the sphere class is that the center of the moving sphere is dependent on the time the ray exists whereas the center of the sphere remains fixed.

[[1]](#footnote-0)

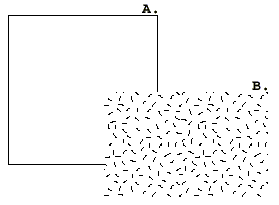


Fig. 1. Image demonstrating Motion Blur.

Q. How does Fig 1 demonstrate Motion Blur?

A. \_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Table 1. Motion Blur Performance | | | |
| Year | Submitted | Accepted | Accepted (%) |
| 1994 | 91 | 41 | 45.1 |
| 1995 | 102 | 41 | 40.2 |
| 1996 | 101 | 43 | 42.6 |
| 1997 | 117 | 44 | 37.6 |
| 1998 | 118 | 50 | 42.4 |
| 1999 | 129 | 47 | 36.4 |
| 2000 | 151 | 52 | 34.4 |
| 2001 | 152 | 51 | 33.6 |
| 2002 | 172 | 58 | 33.7 |
| 2003 | 192 | 63 | 32.8 |
| 2004 | 167 | 46 | 27.6 |
| 2005 | 268 | 88 | 32.8 |
| 2006 | 228 | 63 | 27.6 |
| 2007 | 216 | 56 | 25.9 |
| 2008 | 197 | 50 | 25.4 |

Q. How performant is the Ray Tracer?

A. \_\_\_\_\_\_\_\_

# Volumetric Rendering (TODO)

Volumetric Rendering involves a set of techniques used for rendering high quality 2-D images from a 3-D volumetric data set. (TODO: Which technique? Volumetric Ray Tracing?) The technique does not spawn a secondary ray, instead, the ray simply “pushes” through the surface of the object and computation is resumed along the ray. This is how volumetric data is gathered. Weather phenomena such as smoke and fog can be rendered this way. We can model them as surfaces and have a probability function tell us whether the surface exists at that point or not. As the ray passes through the volume, it may scatter at any point. A denser volume increases the probability of scattering. Our implementation assumes the shape of the boundary of the surface is convex such as a box or sphere. It does not work with a torus or other non-convex shapes. We also assume that the participating media have constant density.

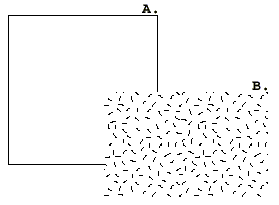


Fig. 2. Image demonstrating Volumetric Rendering.

Q. How does Fig 2 demonstrate Volumetric Rendering?

A. \_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Table 2. Volumetric Rendering Performance | | | |
| Year | Submitted | Accepted | Accepted (%) |
| 1994 | 91 | 41 | 45.1 |
| 1995 | 102 | 41 | 40.2 |
| 1996 | 101 | 43 | 42.6 |
| 1997 | 117 | 44 | 37.6 |
| 1998 | 118 | 50 | 42.4 |
| 1999 | 129 | 47 | 36.4 |
| 2000 | 151 | 52 | 34.4 |
| 2001 | 152 | 51 | 33.6 |
| 2002 | 172 | 58 | 33.7 |
| 2003 | 192 | 63 | 32.8 |
| 2004 | 167 | 46 | 27.6 |
| 2005 | 268 | 88 | 32.8 |
| 2006 | 228 | 63 | 27.6 |
| 2007 | 216 | 56 | 25.9 |
| 2008 | 197 | 50 | 25.4 |

Q. How performant is the Ray Tracer?

A. \_\_\_\_\_\_\_\_

# Image-based Texture Mapping (TODO)

Image-based Texture Mapping is a technique used to generate colors for a surface by producing a view-independent texture map from a set of images taken from different viewpoints. From the hit point, we compute the surface coordinates (u,v). We then use these to index into our procedural solid texture. We can also read in an image and use the 2D (u,v) texture coordinate to index into the image. The *image\_texture* class holds an image and contains a function that, given a hit point and (u,v) coordinates, will return the color of the surface at that hit point by converting into pixel (x,y) coordinates and then grabbing the color of the image at that coordinate.

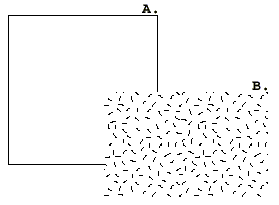


Fig. 3. Image demonstrating Texture Mapping.

Q. How does Fig 3 demonstrate Texture Mapping?

A. \_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Table 3. Texture Mapping Performance | | | |
| Year | Submitted | Accepted | Accepted (%) |
| 1994 | 91 | 41 | 45.1 |
| 1995 | 102 | 41 | 40.2 |
| 1996 | 101 | 43 | 42.6 |
| 1997 | 117 | 44 | 37.6 |
| 1998 | 118 | 50 | 42.4 |
| 1999 | 129 | 47 | 36.4 |
| 2000 | 151 | 52 | 34.4 |
| 2001 | 152 | 51 | 33.6 |
| 2002 | 172 | 58 | 33.7 |
| 2003 | 192 | 63 | 32.8 |
| 2004 | 167 | 46 | 27.6 |
| 2005 | 268 | 88 | 32.8 |
| 2006 | 228 | 63 | 27.6 |
| 2007 | 216 | 56 | 25.9 |
| 2008 | 197 | 50 | 25.4 |

Q. How performant is the Ray Tracer?

A. \_\_\_\_\_\_\_\_

# Conclusion (TODO)

short paragraph stating anything interesting you learned or how you would extend the code if you had time.

|  |  |
| --- | --- |
|  |  |

**References (TODO)**

1. J. Allebach. Binary display of images when spot size exceeds step size. Applied Optics, 15:2513–2519, August 1980.
2. E. Catmull. A tutorial on compensation tables. In Computer Graphics, volume 13, pages 1–7. ACM SIGGRAPH, 1979.

1. Resources Used:

   1) Books by Peter Shirley:

   1. Ray Tracing in One Weekend-<https://raytracing.github.io/books/RayTracingInOneWeekend.html>
   2. Ray Tracing The Next Week-<https://raytracing.github.io/books/RayTracingTheNextWeek.html>

   [↑](#footnote-ref-0)