# Numerical Evaluation of Ray Tracer of Group 48

Sina Sen

***Abstract*—In this document, we explored how our ray tracer performs given different scenes. Our primary criterion is the render time that our ray tracer takes to render the scene we give as the input. We use different scenes for this. (*Abstract*)**

***Keywords—component, formatting, style, styling, insert (key words)***

## I. Introduction

In the research paper published by Utah University[1], it is said that "We observe that even for smaller scenes, that can essentially fit into cache, memory still is the highest contributor in energy and latency, suggesting that even in a case of balanced compute workload, compute remains inexpensive". So primary thing that affects the rendering speed of the raytracer is apparently the memory of the device which run our ray tracer on. To make sure we will get the best result, we ran all of our scenes in the best laptop our group had. So, the results of this paper will probably not be the same on any other computer. For the evaluation, the main parameter we are using is the relationship between the rendering time of the scenes vs. the number of faces our scene has in total. We will also sketch some graphs to evaluate this relationship more visually and provide the rendered image of the scene for accuracy. We are also running all of the scenes in 1000x1000 resolution

## II. Computer Specs

Specs of the computer we are using:

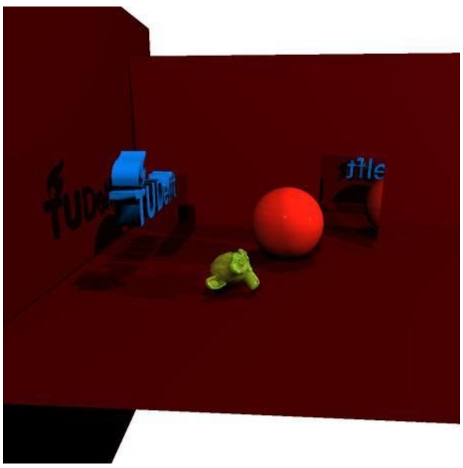
RAM: DDR4 16GB

Processor: Intel Core i7

Processor Speed: 2.6 GHz

Chipset: AMD Radeon Pro 555x

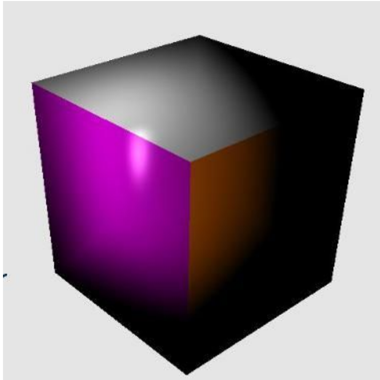
## III. Render Times of scenes



Number of faces: 4823 Rendering Time: 1258.23 seconds

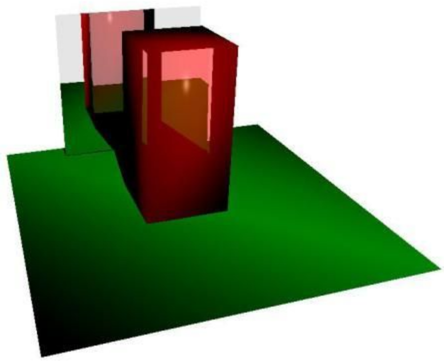
This is the original scene created by us, it contains multiple light sources, hard and soft shadows, refraction and reflection and shading that slows down the rendering.

Scene 2



Number of faces: 12, Rendering Time 6.839 seconds

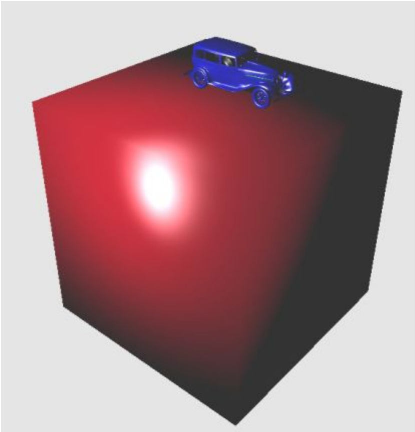
Scene 3



Number of faces: 16, Rendering Time: 13.34 seconds

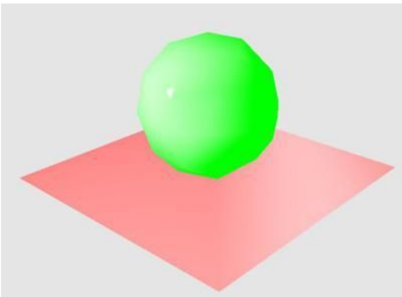
1) http://www.cemyuksel.com/research/papers/rt_performance_CGI18.pdf

Scene 4



Number of Faces: 16311, Rendering Time: 5086.4 seconds

Scene 5



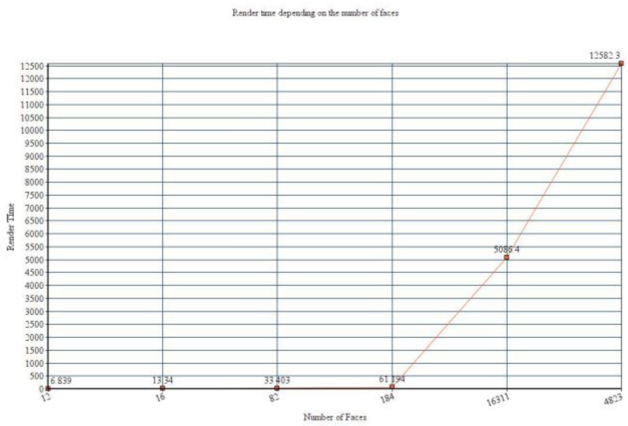
Number of Faces: 82, Rendering Time: 33.403 seconds

Scene 6



Number of Faces: 184, Rendering Time: 61.194 seconds

Graph 1: Rendering time vs. number of faces




(error, point 4823 should be 1258.23, not 12582.3)

| **Number of Faces** | ***Rendering Time*** |
|---|---|
| 12 | 6.839 |
| 16 | 13.34 |
| 82 | 33.403 |
| 184 | 61.194 |
| 4823 | 1258.23 |
| 16311 | 5086.4 |

Fig. 1. Number of Faces and Rendering Time

## III. Evaluation

As one can see from the graph and from the table, it is apparent that the rendering time increases as the number of faces (triangles) that the scene has increases. However, this is not the only criterion. If we take the last two data points of our table, although, the fifth data point has approximately 4 times less number of faces compared to the last datapoint, it is a much more complex scene in terms of scene hierarchy. However, this does not really affect the rendering time of it to be more as it follows a similar trend as other data points. So we can conclude that number of faces (triangles) is the most important factor in rendering time. However, we dont have enough data to argue against the fact that the other factors don’t have any effect at all. he reason behind it can be evaluated as but not limited to (as we need more data for our dataset to be sure and precise) the complexity of the scene, This complexity can include the number of light points the scene has, the light type, number of meshes, reflection, refraction and shading styles. So, if the scene can get much more complex, the rendering speed can decrease as well. To calculate the speed of our raytracer, we can use the original scene we created, and divide the rendering speed to the number of faces our scene has.

Thus we can get;

Speed(rendered faces per second) = 4823/1258.23 = 3.833

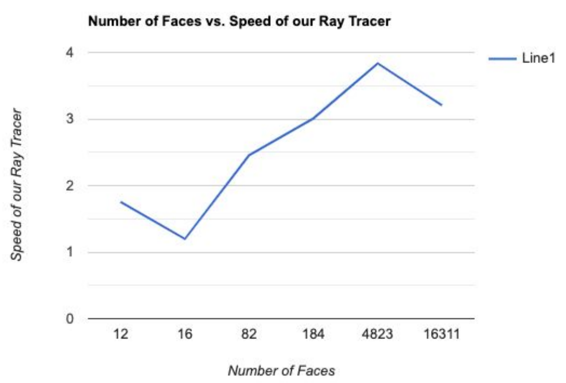
which is hopefully good for this project.

Let’s try this for our other data points as well:

12/6.839 = 1.754
16/13.34 = 1.199
82/33.403 = 2.454
184/61.194 = 3.006
16311/5086.4 = 3.206

Average speed = 2.575

So, there’s a easily recognizable trend going on in these calculations.  As the number of faces (triangles) increase, the speed of our ray tracer (rendered faces per second) increase, with only one exception, which can be interpreted as it is a consequence of the complexity of the scene . To visualize this:




One last thing to keep in mind is that without the acceleration structure, the same scene was rendered in much more time. So, the implementation of our acceleration structure and the multicore support (12 core processor) helped our ray tracer to get much faster.