Week 7 Report

Jakub Dutkowski, s121937
Alexander Birke, s124044

December 19, 2012

1 INTRODUCTION

The purpose of this week's set of exercises is to introduce anti-aliasing to the previously developed ray casting. This is done to make edges in the render look more smooth.

2 DESCRIBTION

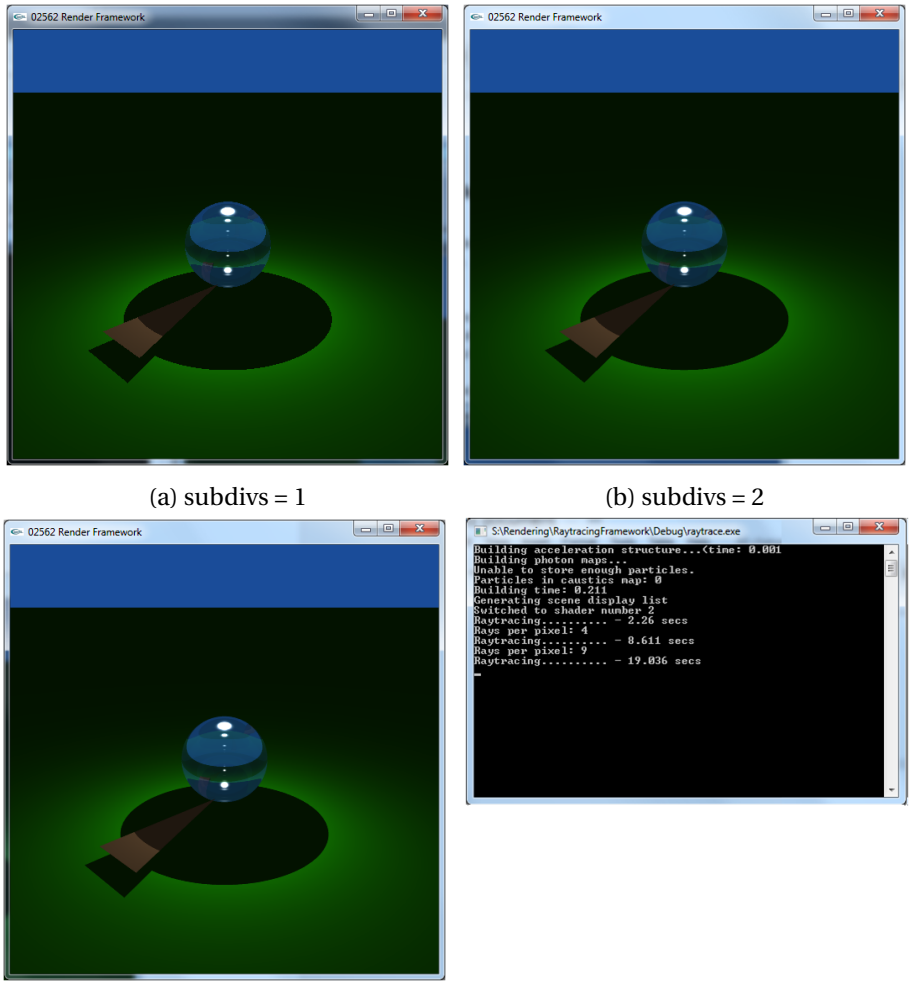
- *Describe how the pixel subdivision level is changed while the program is running*
  The `subdivs` variable is incremented and `compute_jitters` function is called, which fills the `jitter` array.
- *Explain what the function `compute_jitters` stores in the vector array `jitter`.*
  The `jitter` array stores coordinates of random points inside subpixels, which are calculated based on coordinates of the pixel and number of its subpixels.
- *Explain how many subpixels we get for each pixel when the pixel subdivision level is `subdivs = s`.*
  Each dimension of a pixel is divided into `s` parts, therefore the number of subpixels equals s^2.

3 CODE

Listing 1: Part of the displayMyPolygons function from radiosity.cpp file

```
float3 RayCaster::compute_pixel(unsigned int x, unsigned int y)
     const
{
  float2 viewportCoords = optix::float2();

  viewportCoords.x = lower_left.x + win_to_ip.x*x;

  viewportCoords.y = lower_left.y + win_to_ip.y*y;

  float3 result = make_float3(0);

  // for each subpixel
  for(int i = 0; i < subdivs; i++)
       for(int j = 0; j < subdivs; j++)
       {
          // create a ray and hit
          optix::Ray ray = scene->get_camera()->
               get_ray(viewportCoords + jitter[i*subdivs + j]);

          HitInfo info;

          // and then trace it
          if(scene->closest_hit(ray, info))
          {
               result +=  get_shader(info)->shade(ray,info);
          }
          else
          {
               result += get_background(ray.direction);
          }

       }

       // when all rays have been traced, divide by the
       // number of subpixels to get the correct result

  return result/(subdivs*subdivs);
}
```

4 Render Results



(a) subdivs = 1 (b) subdivs = 2

(c) subdivs = 3 (d) render output

Figure 4.1: screenshots of render results

5 Final Questions

- *What is the relationship between the pixel subdivision level and the render time?*
  The time is proportional to squared subdivision level.
- *What is the relationship between the pixel subdivision level and the aliasing error in the render result?*
  Using a higher number of rays per pixel decreases the aliasing error. High number of rays shot through one pixel means that the resulting color of the pixel is calculated as an average of bigger number of values and therefore produces a more accurate result.
- *At what pixel subdivision level would you say that the improvement is no longer visible?*
  We cannot see any improvement between subdivision level 2 and 3.
- *At what pixel subdivision level would you say that the improvement is no longer worth the increase in render time?*
  It depends on the situation, for a test render we can say that there is no point using subdivision, for the final render it's worth waiting for render with subdivision level 2, but not more as there is no further improvement visible.