COSC363 Assignment 2

Ray Tracing

Max. Marks: 20

Ryan Sheridan (98351301)

Due: 11:55 pm - Thursday 31st May 2019

How to Build:

1. Open a terminal window and CD into my submissions folder
2. Run the command: mkdir bin
3. Run the command: CD bin/
4. Run the command: cmake ../
5. Run the command: make
6. Run the command: cp ../world.bmp .
7. Run the command: ./RayTracer.out

This should then provide you with the correct output, including textures.

Extra features implemented	Implemented?	Marking
Other Shapes - Cylinder	Y	1
Multiple lights	Y	1
Refraction	Y	1-2
Anti aliasing:	Y	2
Object Transformation	Y	1-2
Non-Planar Texturing	Y	1
Non-Planar procedural pattern	Y	2
camera motion	Y	1-5
Acceleration algorithm	Maybe?	1-5

Report

I have created a scene of objects using the Ray Tracing algorithm. It consists of 3 different spheres, a base plane, a rectangle and a cylinder.

My programs biggest success is the dynamic Anti-aliasing I have implemented. While it does cause the program to take a significant amount of time longer to display, it is the most interesting and complex part of my algorithm. With this on it will take 10 - 20 seconds to initially display, and under 5 seconds to refresh if a key has been pressed.

Extra features:

1. I have implemented a Cylinder

2. I have implemented a secondary light source. Both lights generate shadows which can be seen easily in [Image 1], especially at the base of the cylinder.

3. No spotlight

4. No Transparent object

5. I have created a refractive sphere with an ETA of 1.012 Objects can clearly be seen through it, and its reflection can also be seen on the reflective spheres surface.

6. I have implemented dynamic Anti-Aliasing to remove the jagged edges of textures, objects and shadows. While some spots of the scene still have these. The effects are greatly reduced through the use of my anti aliasing.The difference between my dynamic anti aliasing and “standard super sampling” can be seen most easily when the scene has a low number of divisions, as seen in the images below.

   I have implemented this by replacing the simple trace call in the display() function with a call to my anti_aliasing() function. This function then divides each cell into 4, and sends a ray into each of these to return the colour. If any of these 4 colours differ from the others it will then divide the cell further into 4 smaller cells to find their colours, and thus the colour of the whole cell will then be the average of all 16 smaller cells within.

   The difference between Dynamic anti aliasing, standard anti aliasing and no anti aliasing can be seen in images [Image 1], [Image 2] and [Image 3].
   The difference between dynamic and standard is small. But can be very easily seen if the number of cells is smaller (~100).

7. I have implemented a very simple shear transformation of the red rectangle.
   Pressing the up key will increase the size of the rectangle and the down key will decrease the size of the rectangle to a minimum of 1 unit size.
   I have implemented this by popping the whole shape upon pressing a key and re-appending it back onto sceneObjects with a larger size.

8. I have created a sphere with a texture of a world map mapped onto it. On top of this map texture, I have created a procedural pattern representing the maps latitudinal lines in addition to the texture.
   The equation for applying the texture coordinates is:

```
float textureCoordU = (0.5 - atan2(d.z, d.x) + M_PI) / (2 * M_PI);
float textureCoordV = 0.5 + (asin(d.y) / M_PI);
materialCol = texture.getColorAt(textureCoordU, textureCoordV);
```

   The equation for applying the latitudinal lines is:

```
int mody = int((ray.xpt.y) * 20) % 8;
if (!mody)
    materialCol = glm::vec3(1,1,1);
```

9. I have created a cylinder with a procedural pattern. The equation for this pattern is:

```
int modx = int((ray.xpt.x) * 5) % 4;
int mody = int((ray.xpt.y) * 5) % 4;
if (modx and mody)
   materialCol = glm::vec3(1, 0.27, 0); // Orange
else
   materialCol = glm::vec3(0.57, 0.01, 0.57); //Purple
```

   The pattern is very easy to change, with changes to the values in the first 2 lines differing the size of each colour in the pattern.
   The textured sphere and the ground plane of the scene also have procedural patterns.

10. No fog

11. In an effort to speed up my program, once the user presses a key(either to move the camera or to change the size of the rectangle) anti aliasing is turned off. This means that before a change in the display, the scene will still be of high quality. This optimization allows the user to see changes in the display very quickly.
    The results of this can be seen in [image 4], after movement and/or an object transformation. Notice the jagged edges along texture boundaries.

12. I have implemented simple camera motion in my program by changing the origin position of the eye.
    Pressing the left arrow key moves the camera to the left, pressing the right arrow key moves the camera to the right.
    The results of this can be seen in [image 4], where the camera has been moved to the left.

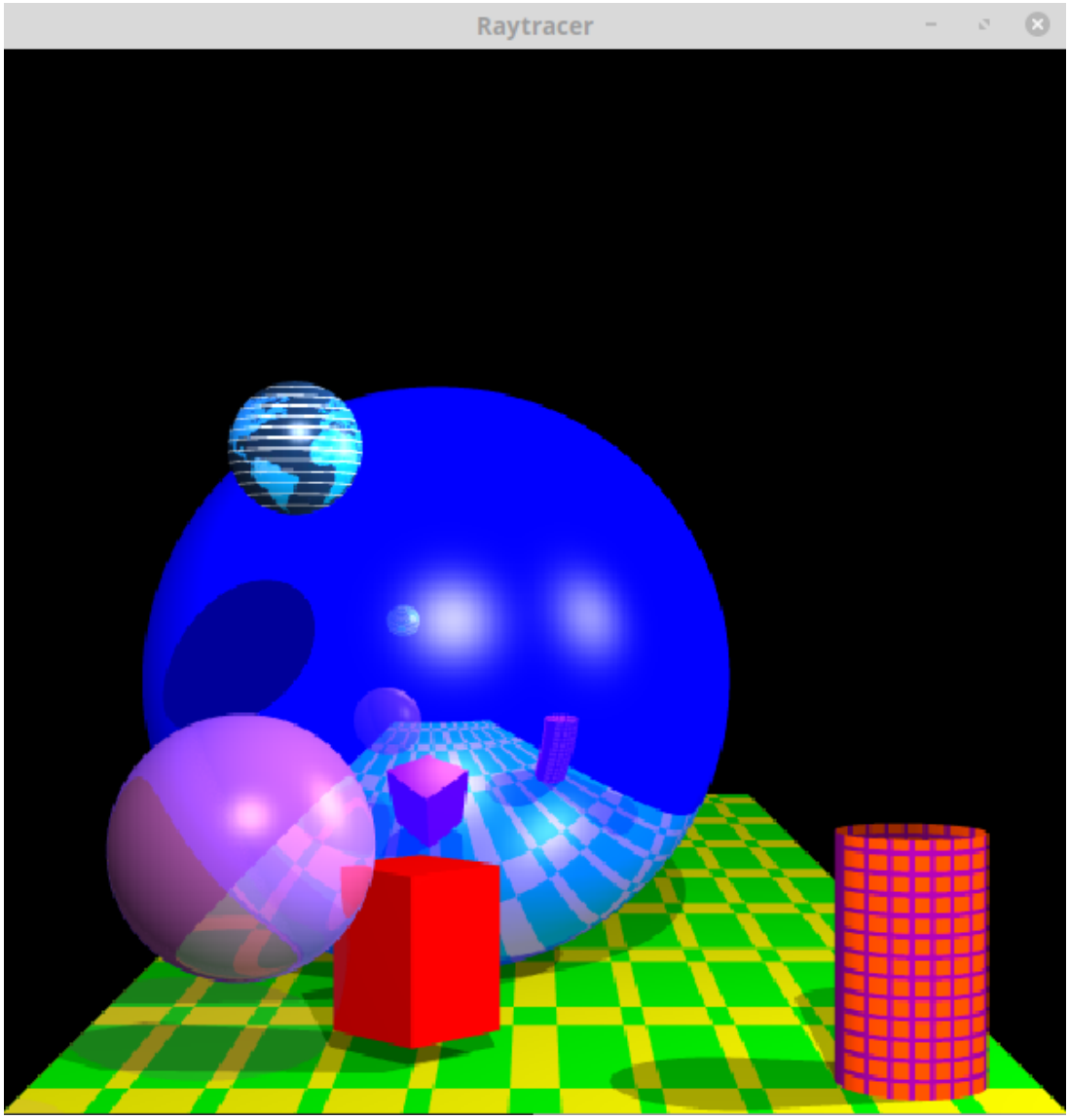
References

Cylinder intersection equation	http://woo4.me/wootracer/cylinder-intersection/

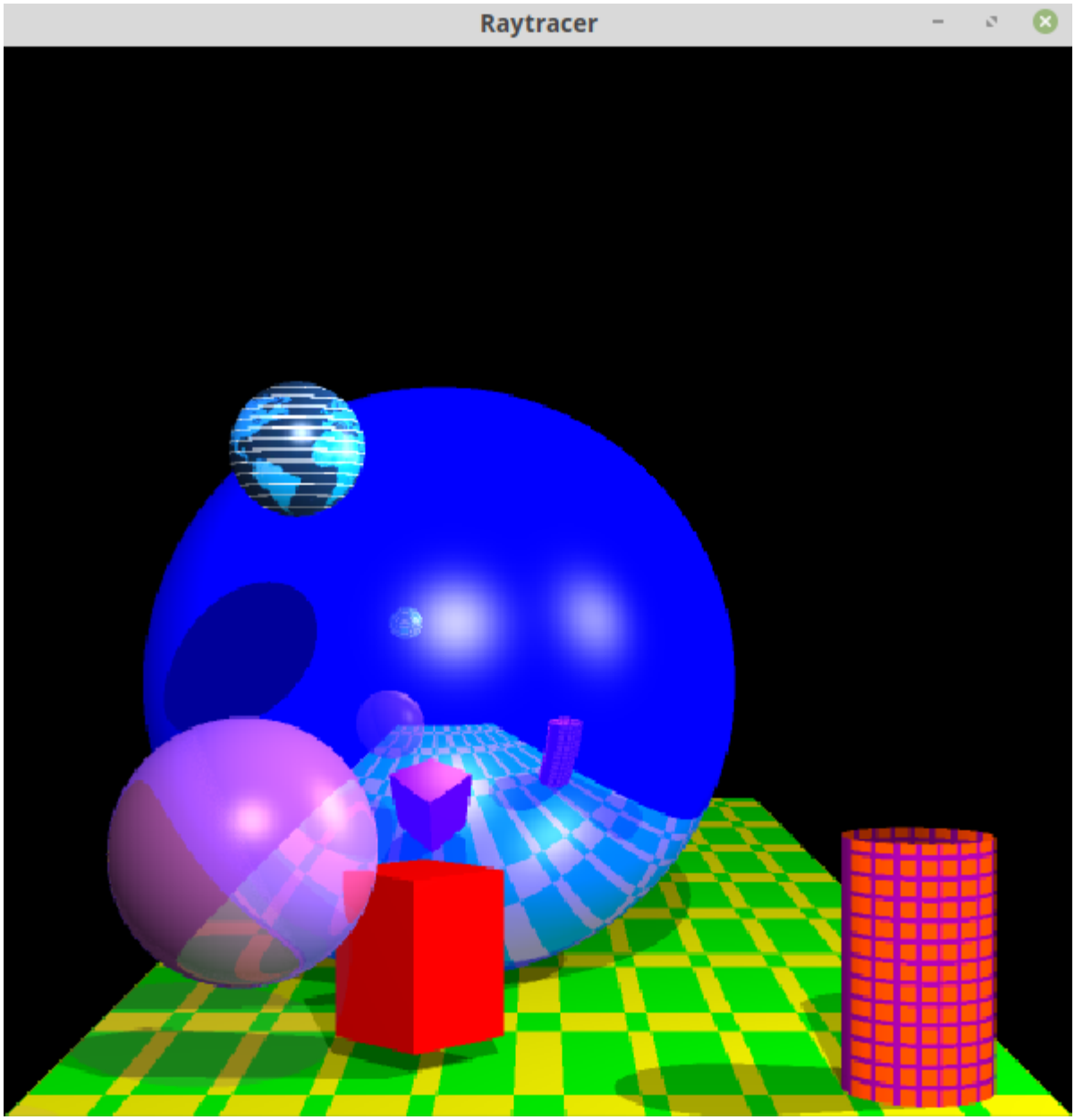
BMP image loader	R. Mukundan Department of Computer Science and Software Engineering University of Canterbury, Christchurch, New Zealand.
Glm library functions	https://glm.g-truc.net/0.9.4/api/a00131.html#gabbb4909d3e99a7a2411cc63252afbbd8

Images

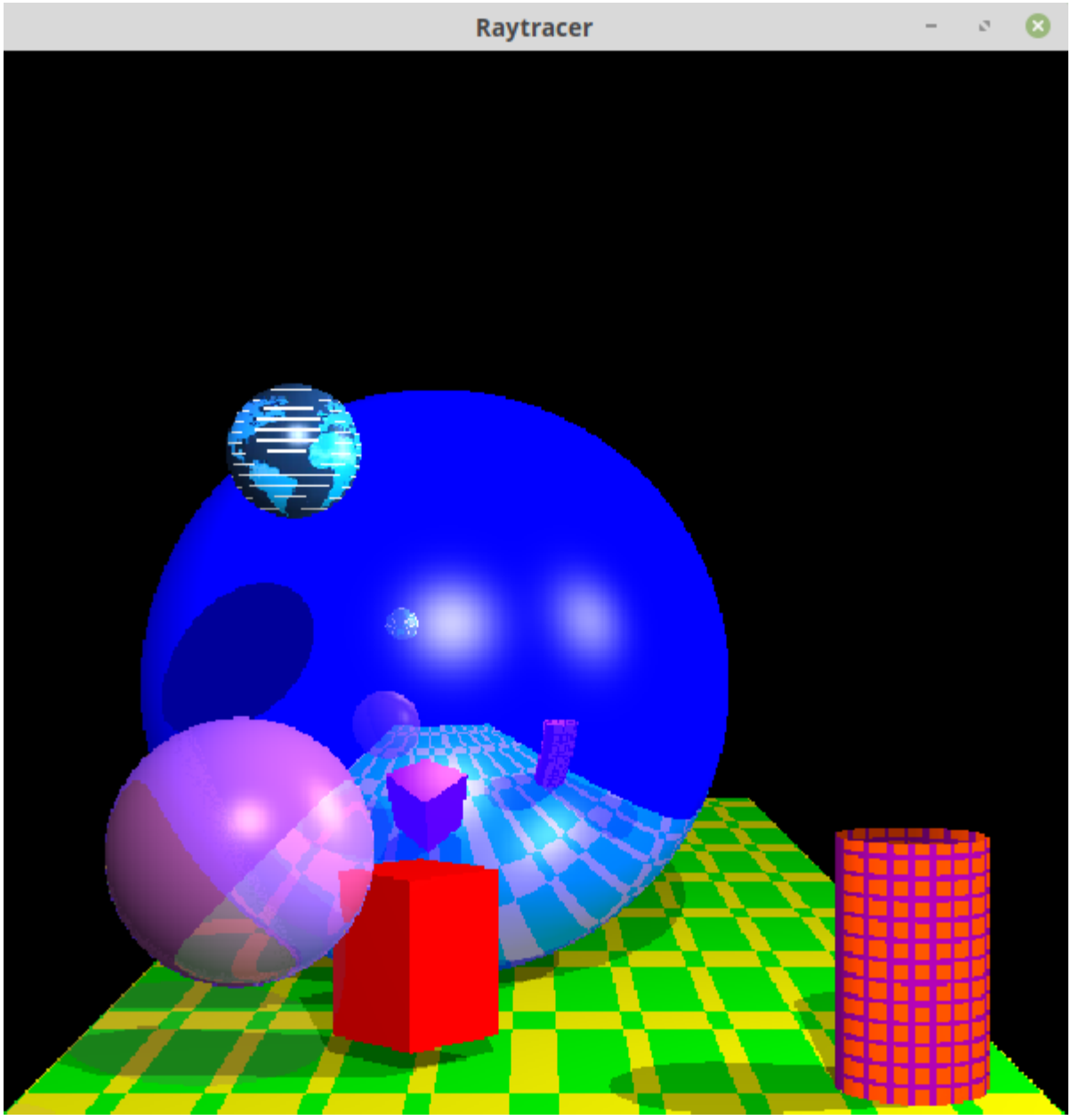
[Image 1] - This image is of the ray tracer with Dynamic Anti Aliasing on.



[Image 2] - This image is of the ray tracer with standard Anti Aliasing on



[Image 3] - This image is of the ray tracer with no anti aliasing at all.



[Image 4] - This image is of the ray tracer after moving the camera to the left and increasing the size of the cube.