Lab 2: Raytracing

Computer Graphics with Interaction DH2323

Author: Niklas Blomqvist, nblomqvi@kth.se
Lab partner: Annika Strålfors

May 26, 2014

Contents

1 Tracing rays

We first loaded a model of a room, made up by triangles in specific positions, into a vector. This is what we will use to decide which color a pixel on the canvas should have. We trace rays from the camera position to all positions on the canvas, and depending on what triangle is hit first, the color is decided by that triangle.

The dimensions of the room is:

$$-1 \leq x \leq 1$$

$$-1 \leq y \leq 1$$

$$-1 \leq z \leq 1$$

We put the camera on the position (0,0,-2) so we would be able to see the room clearly. With a width and height of 500 x 500 pixels we chose a focal length of 250 pixels thus having a vertical and horizontal angle of view(α) of 90 degrees(see wikipedia):

$$\alpha = 2 * arctan(\frac{d}{2f})$$

The distance between the camera(-2) and the start of the room (-1) is 1 and the length of the room is 2. This can be visualized by trigonometry and we see that the camera will cover the whole room.

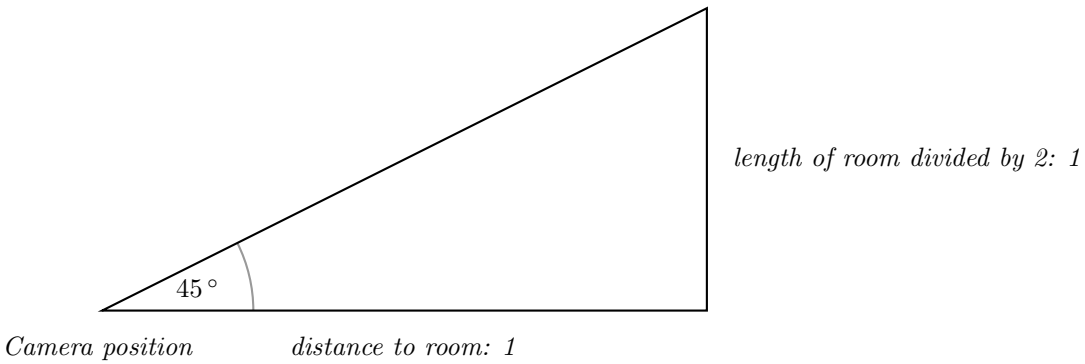

Figure 1: Visualization of the cameras view.

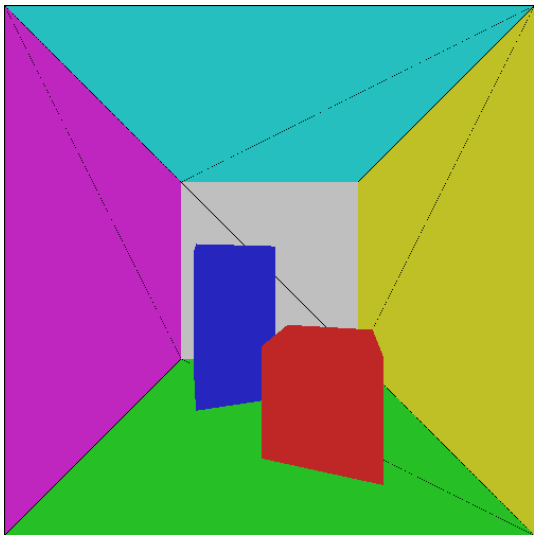

Figure 2: The first outcome of our raytracing.

We can see a lot of small black dots in what seems to be the borders of the triangles. After some time we noticed the if-rule in our closestIntersection-function had the following rules.

$$0 < u$$

$$0 < v$$

$$u+v < 1$$

We figured that sometimes the points in the triangle borders are denied as intersections so we altered the rules to include these points:

$$0 \leq u$$

$$0 \leq v$$

$$u+v \leq 1$$

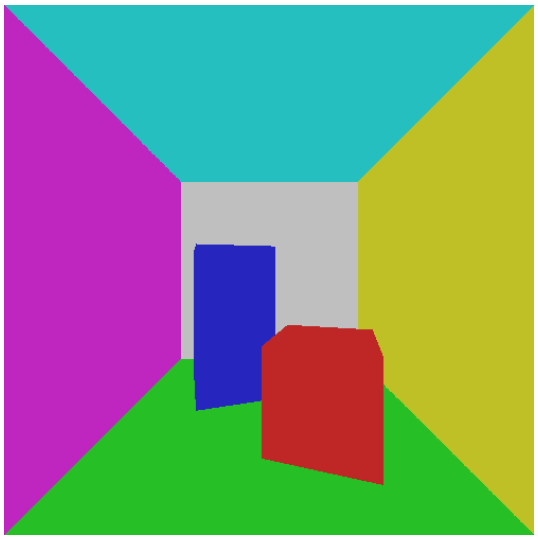

Figure 3: The result of including these points.

2 Moving the camera

The second assignment of this lab was to implement camera-movement and rotation. Moving the camera along the z-axis(forward or backwards) was implemented by adding or subtracting from the z-coordinate when pushing the UP or DOWN arrow keys. The rotation around the y axis was implemented by introducing a rotation matrix, R, see wikipedia. When pushing LEFT or RIGHT arrow keys a global variable, yaw(θ), was set to -0.1 or 0.1 and the triangles vector(the actual room) was updated with the new positions by multiplying the values v0, v1 and v2 with R.

It wasn't before reading the instructions and doing the third lab that we realised that this was not the right approach. Instead of updating the triangles vector, we now update the direction by multiplying it with the rotation matrix R, which gives us a proper yaw effect. When moving forward or backward we now multiply the forward vector with a constant value to move the camera in the direction of the camera or when moving backward, from the direction of the camera.

3 Illumination

This last part of the assignment was about adding illumination and shadows. Two global variables was added, a position for the light source and the power of the light source P. We then checked every coordinate in the room to see what intensity, D, this position should have with the following formula:

$$D = P * max(\hat{r} \cdot \hat{n}, 0)/4\pi r^2$$

where $\hat{n}$ is the unit vector describing the normal of the current surface and $\hat{r}$ is the unit vector describing the direction from the surface point to the light source. We use these unit vectors to decide in what angle the light hits the surface and therefore we can decide the intensity of the light at that position.

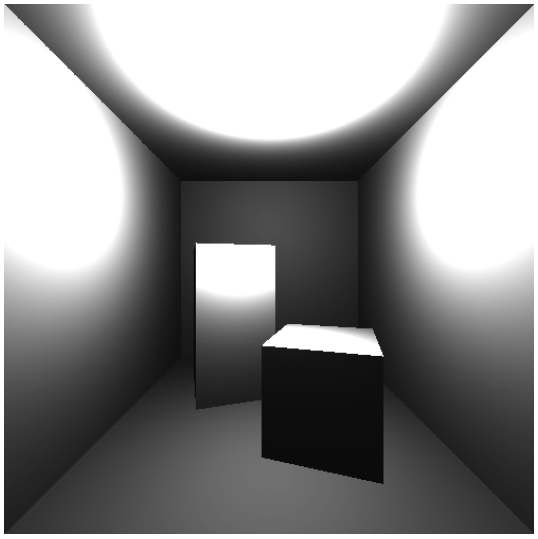

Figure 4: The incomming light.

We add the color of the surface to this formula and receive the following canvas.

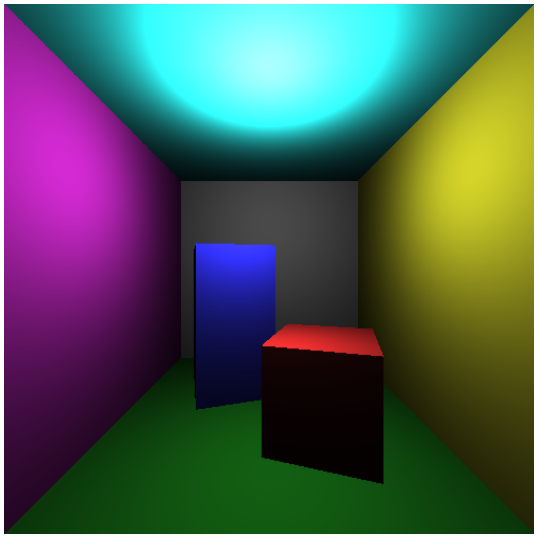

Figure 5: The illumination with colors.

3.1 Direct shadows

To add shadows we do an extra check before deciding the light intensity in each point of the canvas. We cast another ray from that position to the light source, and if we find an intersection that is closer than the light source the current position must be in shade. Implementing this we received the following canvas:

This took a while figure out the solution. We finally realised that changing the acceptable t-values for an intersection from

$$0 \leq \mathrm{t}$$

to

$$0.0001 \leq \mathrm{t}$$

fixed this problem. We think that somehow the camera can interfere with our rays and removing the values directly at the same position as the camera solves this.

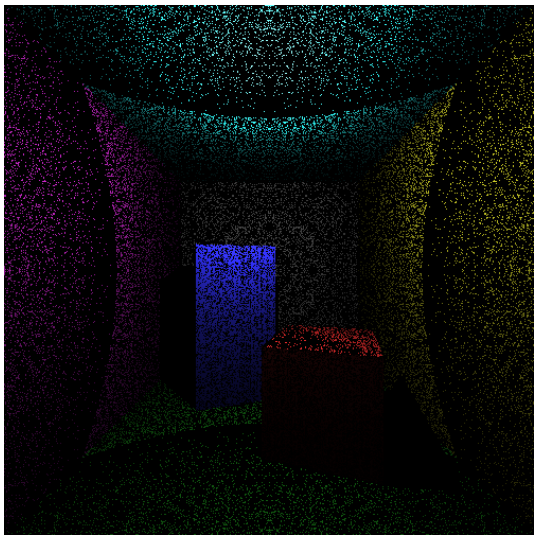

Figure 6: First try at shadows with a lot of black dots.

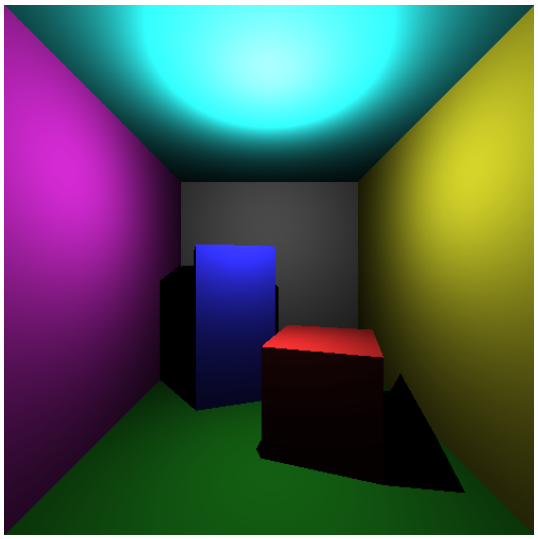

Figure 7: Shadows with the t-fix.

3.2 Indirect Illumination

The last thing we did was to add indirect illumination. This is a hard and expensive process so we used an approximation instead. All we did was add a fixed variable to the formula before colouring the surface.

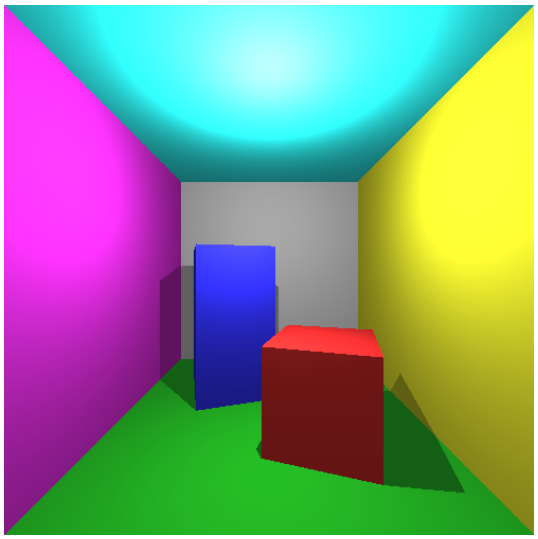

Figure 8: Constant approximation of the indirect illumination.

4 Summary

This lab was hard and took a long time to finish. It took time to understand the underlying theories and a lot of knowledge in the linear algebra department needed rehearsing. I cannot count the number of bugs and problems we faced during this lab. We mention many of them, probably the biggest ones in this report, but not all. The end result was satisfactory and even though we had to struggle a long the way the execution was fun.