Lab 2

DH2323 DGI22

Josephine Kvarnberg
jkv@kth.se

13 May 2022

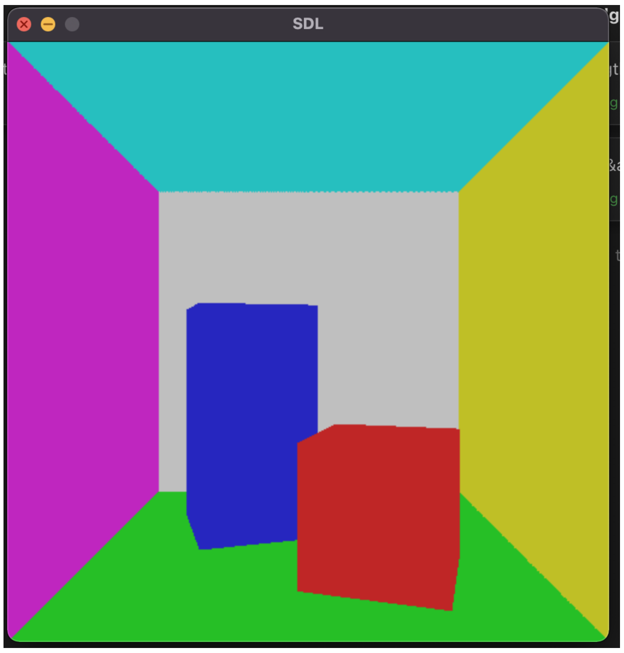
1. Representing surfaces

The surfaces and cornell box was set up after the instructions. To prepare for this lab, I researched ray tracing, ray triangle intersection, and other topics that relate to it. There were a lot of great resources online. I have collected these in the last part of this report, under References.

2. Intersection of Ray and Triangle

Part 2 of this lab was thoroughly described in the instructions. The equations for the intersection computations were implemented according to the description.

Equations 7, 8, 9, and 11 all have to be true for an intersection to occur. The input triangles are looped through to check for intersections.



Rendering of the scene.

3. Tracing rays

Computation for what ray each pixel corresponds to was implemented, to be able to render an image with the ray tracing.

I chose to have the focal length as the screen height, for simplicity. The camera position was tested out and I ended up putting it at (0,0,-3). The length of the room is 2, but I felt that (0,0,-2) was too close to the scene given the focal length. I could probably decrease the focal length and have the position be closer, but I liked how this looked.

Both horizontal and vertical field of view will be 90 degrees, given that the dimension of the screen (and room) is square.

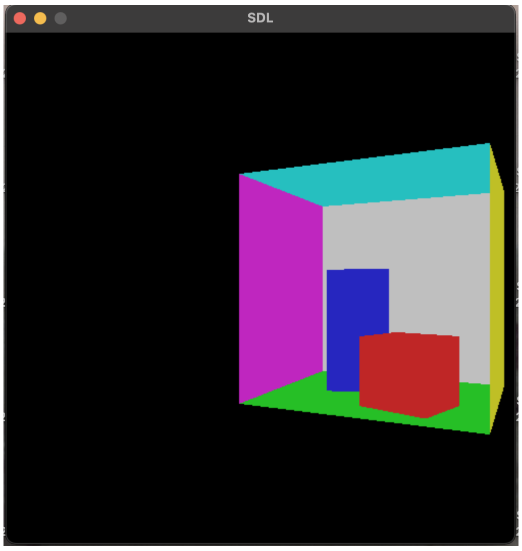
4. Moving the camera

Checking for pressed buttons were described to be done by using “Uint8 *keystate = SDL_GetKeyState(0);”. In SDL2 this does not work. instead I used “const Uint8 *keystate=SDL_GetKeyboardState(NULL);”.

I tested different values for moving and came to the conclusion that these should be small, since the movement looks better and smoother with small increments of the cameraPos.

Rotation was quite hard to get right, since I got weird results where it only spun in a small circle but the camera still pointed forward. When I got the rotation to work, the axes were not moving with the rotation. The overall movement (front to back, side to side, and rotation) were then fixed by using the right and forward vectors that were described in the lab instructions.

I got some good feedback from a TA on how to understand movement in the direction that the camera is facing[1].



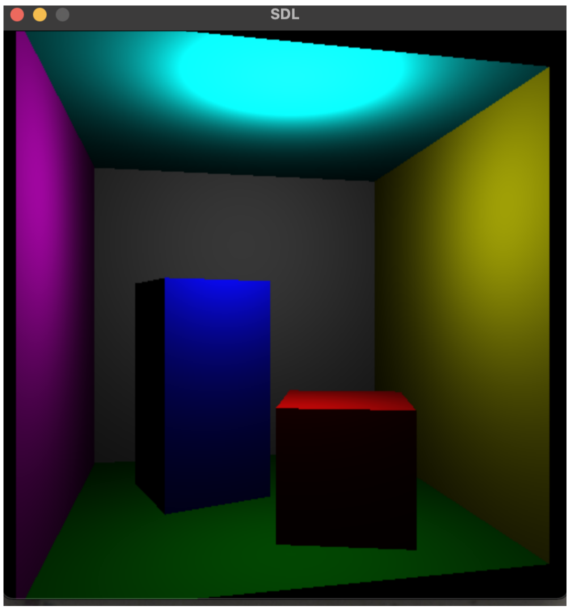
Rotation and movement of camera.

5. Illumination

The function DirectLight was implemented to add a light source for illuminating the scene. The following formulas were central to this:

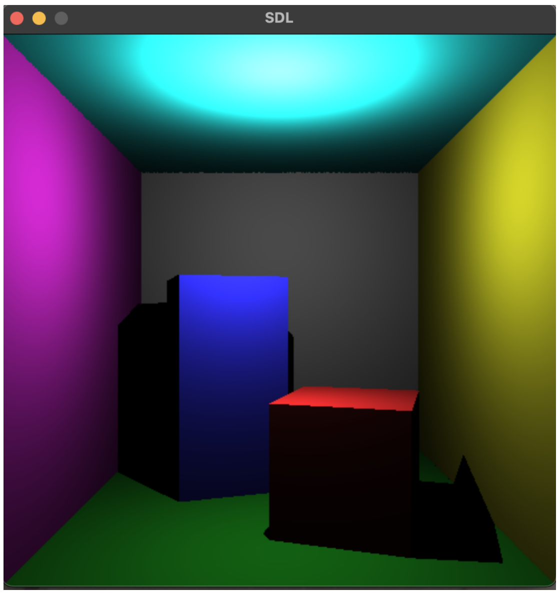
- Area of sphere A = 4πr^2
- The power per area reaching any surface point directly facing the light source B = P/A = P/4πr^2
- Power per real surface D = B max(r̂ . n̂ , 0) = (P max (r̂ . n̂ , 0))/4πr^2

[1] https://www.3dgep.com/understanding-the-view-matrix/



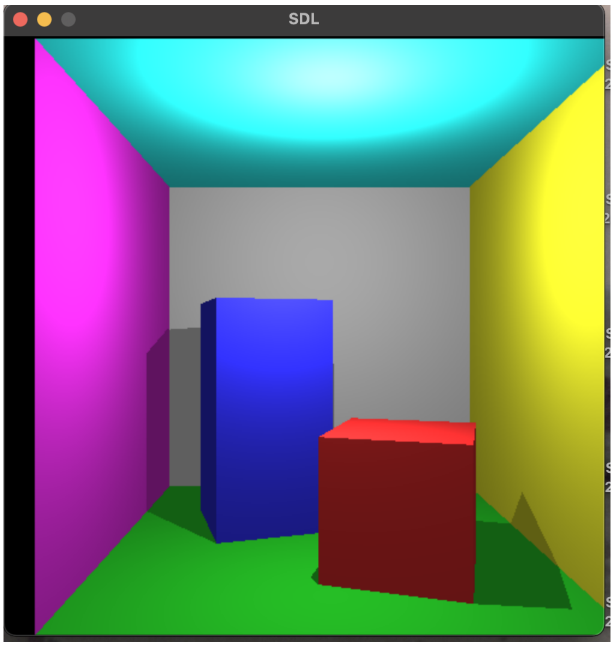
Results of implementing illumination from a light source.

Shadows were also implemented. At first they ended up in the wrong direction and on the side walls. I think it was because I sent in the wrong direction in ClosestIntersection in DirectLight, using rnorm which points from surface to the light. It should instead go the other way, so I changed its direction. The surfaces that are occluded from the light will be black, i.e shadows will be seen in the scene.



With shadows.

The last step was to add indirect illumination, to soften the scene and not have shadows be completely black. This was simply done by adding a global variable of vec3 and implementing it in the draw function together with the direct light and color of the triangle.



With shadows, and direct and indirect illumination.

Movement of the lightsource was implemented in the same way that the camera position was altered. The lightPos global variable was changed by detecting button presses and the variable changed by a factor and the axis vectors.

References

- https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-rendering-a-triangle/ray-triangle-intersection-geometric-solution
- https://cmu-graphics.github.io/Scotty3D/pathtracer/ray_triangle_intersection
- https://en.wikipedia.org/wiki/M%C3%B6ller%E2%80%93Trumbore_intersection_algorithm
- https://www.scratchapixel.com/lessons/3d-basic-rendering/introduction-to-ray-tracing/ray-tracing-practical-example
- https://www.scratchapixel.com/lessons/3d-basic-rendering/ray-tracing-generating-camera-rays/generating-camera-rays