Assignment 1 report

Giulia Benvenuto

March 27, 2022

Environment:

- Windows 11
- Visual studio 2022 17.1

1 Ex.1: Basic Ray Tracing

1.1 Ray Tracing a Parallelogram

The orthographic projection of a parallelogram is implemented in the **raytrace_parallelogram** function.

At first I set the parameters of the parallelogram: its origin (bottom left corner) called "*pgram_origin*" and two vectors to describe its edges called "*pgram_u*" and "*pgram_v*".

I prepared the ray by calculating its origin (p: the position of the pixel on the image) and its direction (-w).

Then I implemented a function, called **check_pgram_intersection_t** to check if there was an intersection between the ray and the parallelogram. The inputs of the function are: the origin and the two edges u and v of the parallelogram, the origin and the direction of the ray. These data are used to fill the matrix A and the vector b of a 3x3 system, which I solved using an Eigen provided function. The output of the function is t, one of the three solutions.

If t different from zero I computed the exact intersection point, its normal and the diffuse model.

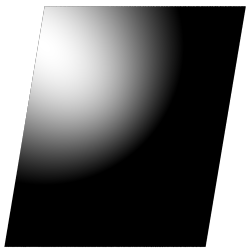


Figure 1: Parallelogram in orthographic perspective.

1.2 Ray Tracing with Perspective Projection

1.2.1 Sphere

The perspective projection of a sphere is implemented in the **raytrace_sphere_perspective** function.

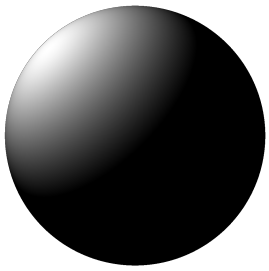
I set the "*sphere_radius*" and the "*sphere_centre*".

I prepared the ray by calculating its origin (in this case it's e: the origin of the camera reference system) and its direction (in this case it's p-e where p is the position of the pixel and e the origin of the ray).

I implemented a function, called **check_sphere_intersection_t** to check if there was an intersection between the ray and the sphere. The inputs of the function are: the centre and the radius of the sphere, the origin and the direction of the ray. These data are used to compute the coefficients a, b, and c of a second degree equation. The output t is the solution of this equation and its value allowed me to calculate the exact intersection point.

The intersection point between the ray and the sphere is computed with the equation: $p(t) = e + td$ then I calculated also its normal and a simple diffuse model to shade the object.

Figure 2: Sphere in perspective projection.



1.2.2 Parallelogram

Implemented in the **raytrace_pgram_perspective** function.

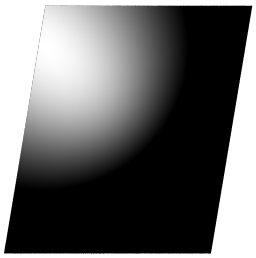
I set the "*pgram_origin*" and the two edges "*pgram_u*" and "*pgram_v*".

I prepared the ray as in the sphere perspective projection case.

I called the function **check_pgram_intersection_t** described in the section 1.1.

I computed the exact intersection point between the ray and the parallelogram, its normal and a simple diffuse model.

Figure 3: Parallelogram in perspective projection.

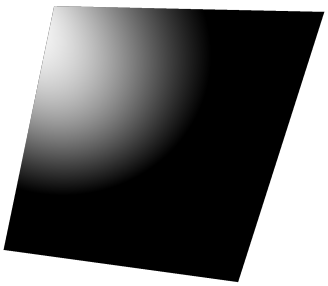


1.2.3 Difference in the result for a sphere and for a parallelogram

In the perspective raytrace, if the parallelogram plane is parallel to the screen, then no matter how I move the perspective, the shape won't change, the only thing that changes is the size of the result.

Otherwise if I decide to slope the parallelogram and change the position of the origin of the rays then the perspective it's visible.

Figure 4: Sloped parallelogram in perspective projection.



While for the sphere, just changing the origin of the rays it will be visible the change of perspective. And the projection will be an ellipse.

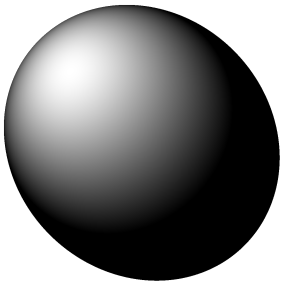


Figure 5: Sphere in a particular perspective projection.

2 Ex.2: Shading

In this exercise I added some matrices to store the colors. (R, $\mathbf{G}$, B) I decided the color of the sphere and saved it iside one vector. I also defined the diffuse coefficient, the specular one, and the Phong exponent, giving to them some selected values. I computed the amounts of reflected light for the diffuse and the specular models and then I applied them inside the final shading equation.

Depending on the choosen color stored inside the dedicated vector and depending on the value assigned to the coefficients I got different results for my shaded sphere. One of them is the following.

Figure 6: Shaded RGBA sphere.