Computer Graphics (CSCI-GA 2270-001) - Assignment 2

Avadesh Meduri (N10537558)

October 6, 2020

1 Common Information

The code was written in Ubuntu 18.04, with a cmake version 3.10.2 and C++ version 7.5.0. I have implemented all the algorithms in this assignment using the template functions provided in the github repo and have only used standard C++ libraries. I have commented the code so as to explain what I am trying to do in each block of the function.

2 Answers to Exercises

1. Answer to Exercise 1

   (a) Intersection with parallelogram

      i. The equation of the parallelogram used in the code is of the form $o+a*u+b*v$, $0 \leq a, b \leq 1$ where o is the bottom left corner of the parallelogram, u,v are vectors that represent two sides of the parallelogram and a,b are scalars.

      ii. The function "intersection_parallelogram" checks if the a given ray (explicit form) intersects with the parallelogram. The function solves the linear equation Ax = b to, where x = (a,b,t) which are the parameters of the parallelogram and ray equation.

      iii. The parameters returned by the "intersection_parallelogram" function x = (a,b,t) are used to compute the exact location of intersection by plugging them into the equations (ray or parallelogram equation).

      iv. The normal at the point of intersection is computed by taking the cross product of the two sides of the parallelogram ($u \times v$). it is assumed the angle between u and v is counter clockwise (positive) so that the normal direction is towards the camera.

   The figure generated by the ray tracer (function "raytrace_parallelogram()") is shown in Figure 1.

   (b) Perspective projection
   The function "raytrace_perspective()" has been modified to generate perspective projection of both a parallelogram and sphere. Figure 2,3 shows the generated scene for different chosen parameters of camera location and screen distance for the sphere and parallelogram. The parameters of the parallelogram have been chosen so that it is not in the xy plane, which is why the parallelogram is not uniformly bright.

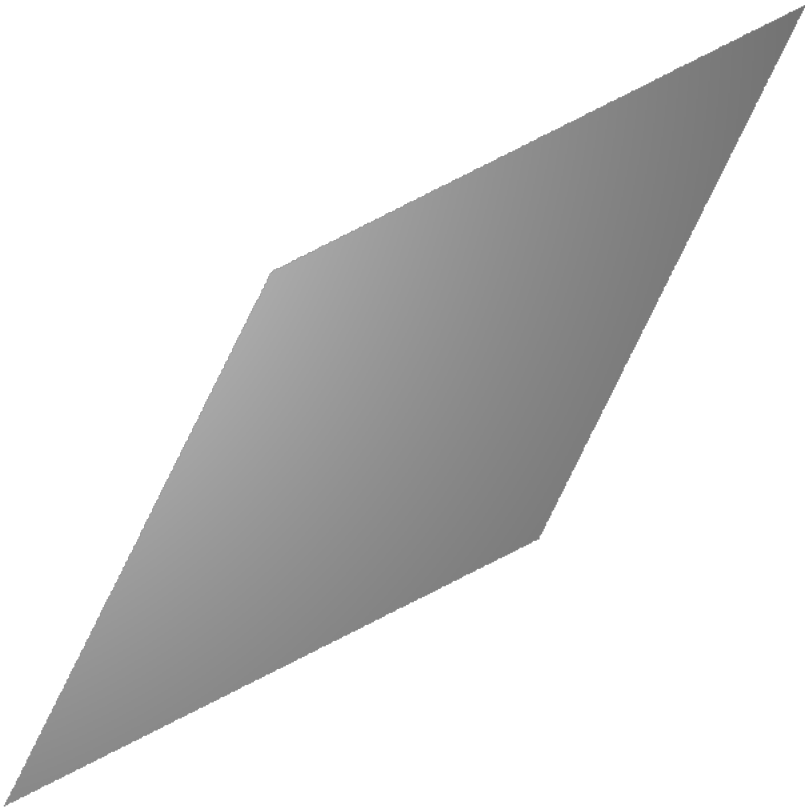

Figure 1: The figure generated by ray tracer for orthographic projection of a plane (parallelogram)

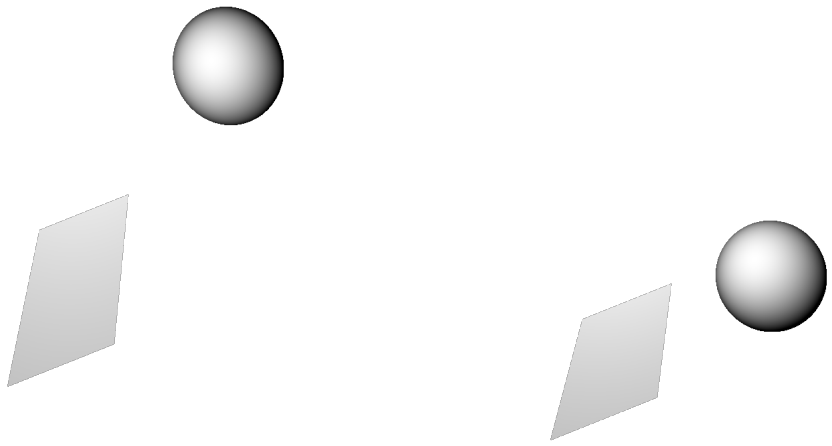

Figure 2: The figures generated by ray tracer using perspective projection of a plane (parallelogram) and sphere for different camera locations. The left figure is when the camera is at (-1, 1, 1), the right figure is for (0,0,1)

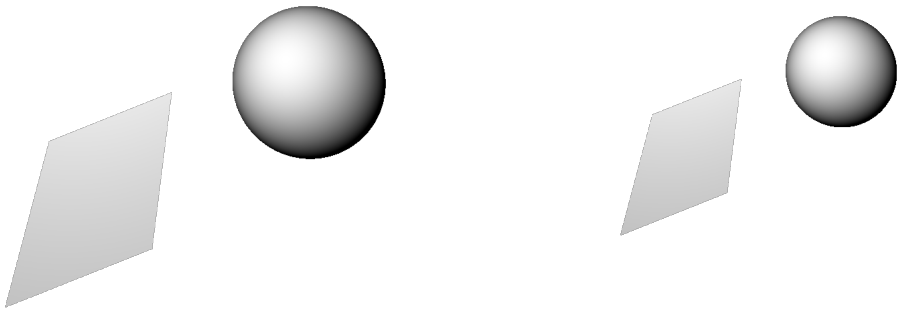

Figure 3: The figures generated by ray tracer using perspective projection of a plane (parallelogram) and sphere for different screen distance. The left figure is when screen is closer to the objects than the right figure

i. A new parameter called screen_dist (screen distance from the camera) is initialised so that the direction of the ray for the given camera position and pixel considered can be computed. The screen is assumed to perpendicular to the z axis. The ray direction is computed by subtracting the location of the pixel being considered in the screen and the camera origin.

ii. The figures generated by the perspective ray tracer are sensitive to the camera location and screen distance, which is not the case for the orthographic projection. In Figure 2, the location of the camera is changed while the rest of the parameters are kept constant. The location of the objects in the scene differ due to this change. Also, the sphere became more ovular in the rendered image when the camera was kept at the corner (as shown in the left image of figure 2). For particular camera positions when the same ray intersects with the same object, the two objects overlapped each other in the generated image while in reality the two objects are not intersecting. In Figure 3, the screen distance is modified while the rest is kept the same. When the screen was moved closer to the objects, they became larger in the resulting image.

2. Answer to Exercise 2 (Shading)

(a) Different shading components have been implemented in the "raytrace_shading" function. Figure 4 shows the images generated using different shading parameters.

(b) The implementation of shading with RGB images is in the "raytrace_shading_rgb" function. Figure 5, 6 shows the images generated using different shading parameters for different color components.

(c) The results of images with different shading parameters are shown in Figures 4, 5, 6. In Figure 4 the rho values (phong constant) are increased across the images. The generated images moved from being more shiny (left most), to glossy and then like a mirror (right most).
In Figure 5, the rho values were increased between the red and the green sphere. A similar result of the sphere becoming more shiny with lower rho values and more glossy with higher rho value is seen. The right most sphere was generated by increasing the ambient parameter (k_a). Increasing the ambient parameter makes the overal object in the scene brighter and reduces the contrast between the darkest and lightest part of the sphere if it is increased significantly.
In Figure 6, all spheres are generated with the same constants except the diffuse lighting constant, which is increased from the left to right(left sphere has least and the right has the largest diffuse constant). A higher diffuse parameter increases the region with more light intensity (The white part on the sphere is larger on the right most sphere). The specular shading parameter increases the prominence of the image segment where the light from the source bounces directly into the camera.

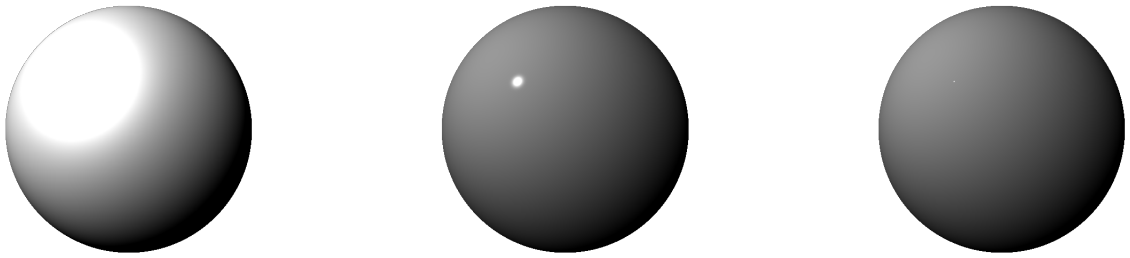

Figure 4: The figure generated by perspective projection of a sphere with different shading parameters. The phong constant is increased from left to right (10, 1000, 10000).

Figure 5: The figure generated by perspective projection of a sphere with different shading parameters for different colors.

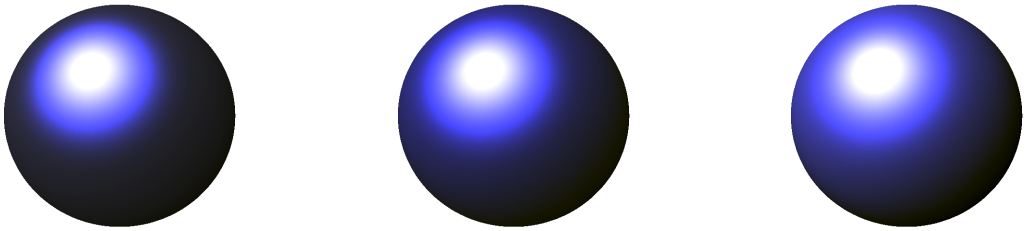

Figure 6: The figure generated by perspective projection of a sphere with different diffuse shading parameters. Increasing diffuse shading constant from left to right.