Computer Graphics (CSCI-GA 2270-001) - Assignment 5

Avadesh Meduri (N10537558)

December 1, 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. Exercise 1
   The silhoute generated after extending the rasterizer to load a mesh (bunny) is shown in Figure 1. The figure can be generated in the final submitted version of the rasterizer by disabling the lighting equations and using just the ambient colors along with flat shading.

2. Exercise 2
   The image generated by three shading options is shown in Figure 2. The rasterizer has three boolean uniform variables (wire_frame, flat_shading, per_vertex_shading) which can be triggered to render different shading. All shading methods are carried out in the vertex shader. I have tried to comment the code to explain what I am trying to do.
   For flat shading, the normals are computed at each vertex by looking at the triangle to which the vertex belongs to while considering it. In the case of per vertex shading the normals at the vertex is computed by computing the normals of each triangle it belongs to in the mesh and adding them up. After which the resulting normal at each vertex is normalized at the end. This approach is aligned with the one explained in class for per vertex shading.

3. Exercise 3
   The 3 gifs that rotate the bunny around its bary center while it is translated in the x and y direction is attached along with this submission and are located in the videos folder. Further, the location of the light source has been changed to create a video with light falling onto the mesh from the top instead of, from the front like in the previous question (this can be changed by altering the uniform.light_position variable). The generation of the gif can by triggered by setting the uniform variable render_gif to true.
   The matrix uniform.bc_rot_trans transforms the vertices of the mesh and also transforms the normal at the vertex in the vertex shader. After which the lighting equations are computed. A video with the mesh moving in the xy plane is shown instead of moving towards the camera because the rendered image does not change significantly while moving towards the camera (apart from rotating)

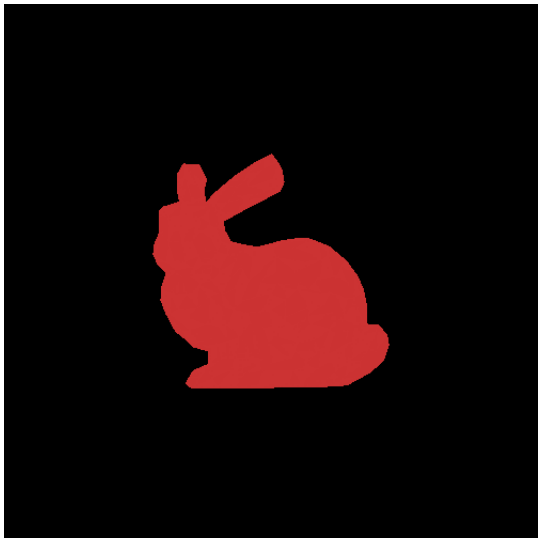

Figure 1: silhouette of bunny rendered by the rasterizer

since orthographic projection is used. Only the lighting changes while moving towards the camera. However, the code can be changed very easily to move the mesh towards the camera by changing the translation part of the SE3 matrix in the z direction. Further, the mesh can be rotate about its bary center in another axis by modifying the rotation part of the SE3 matrix. I have written comments at appropriate locations to try to explain how I compute the bary center and translate the mesh.

4. Exercise 4
   The image generated using perspective projection is shown in Figure 3. The uniform.view matrix is also implemented to take into account of resolution of the framebuffer and prevent distortion of the mesh. The perspective projection can be triggered by changing the boolean variable uniform.is_perspective to true.

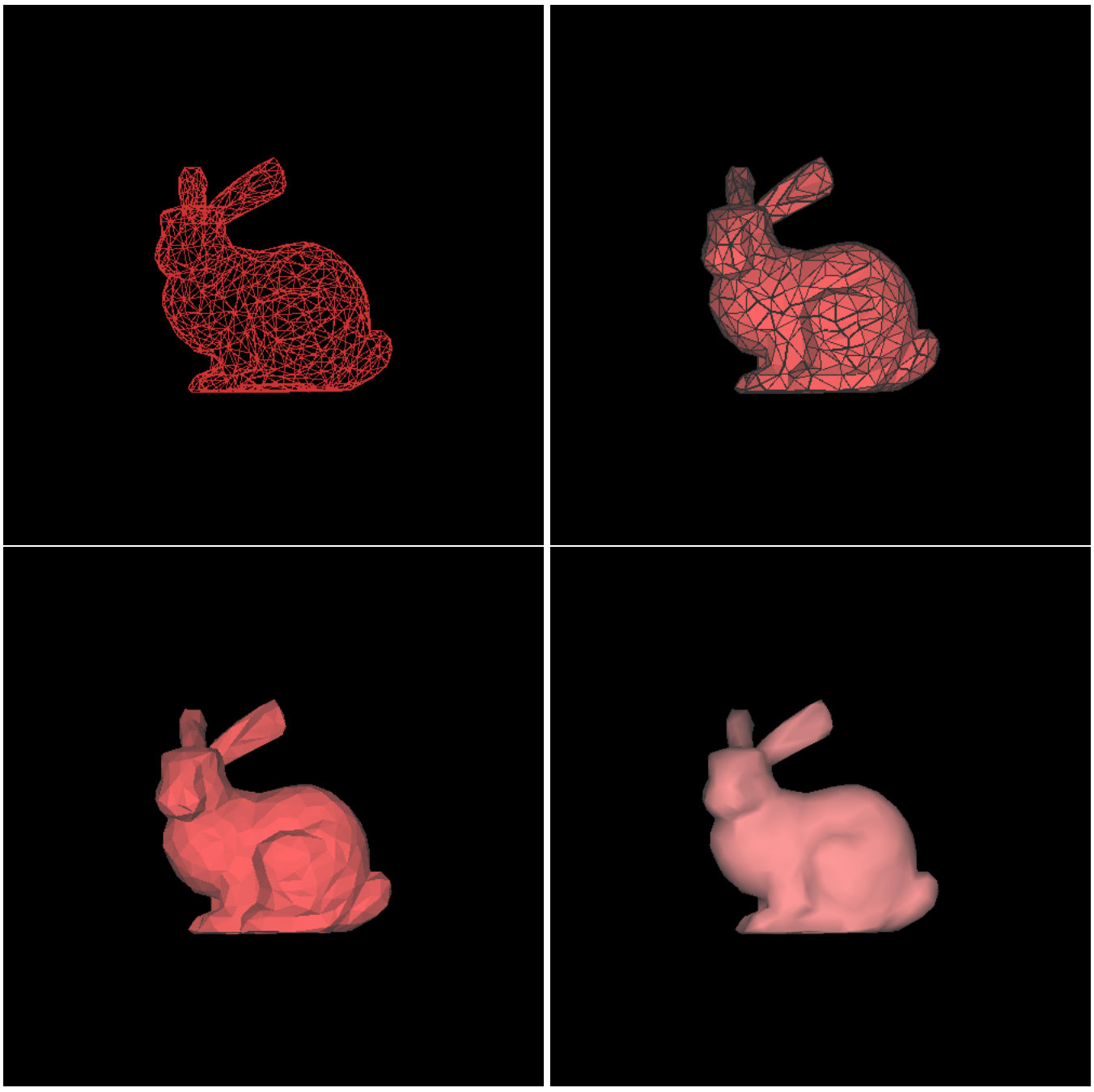

Figure 2: The image of the bunny rendered by the rasterizer using different shading techniques. Top left is wireframe, top right is flat shading with wireframe, bottom left is with flat shading and bottom right image is per vertex shading

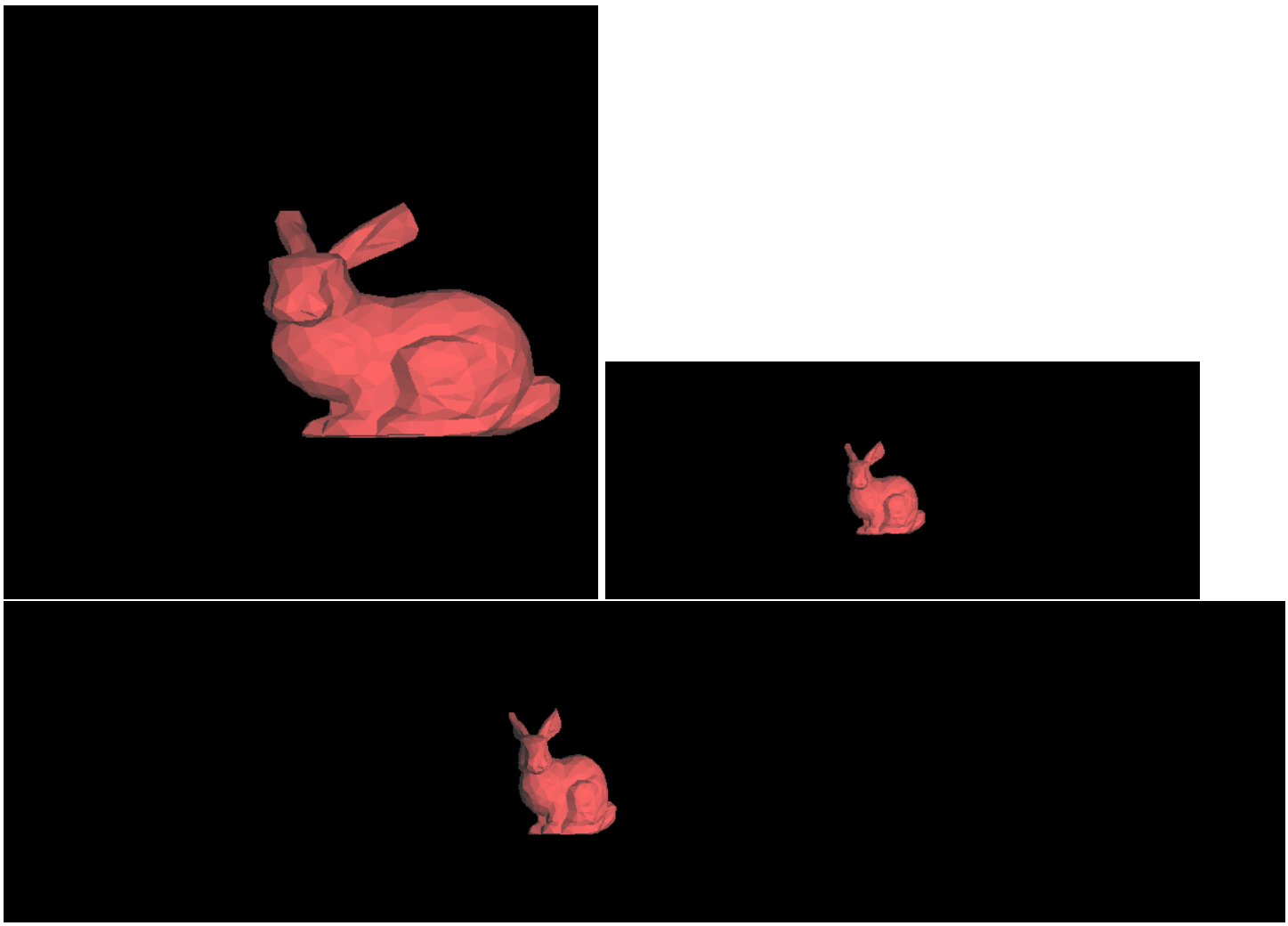

Figure 3: The image of the bunny rendered by the rasterizer using perspective projection. The three images are rendered with at different resolutions by altering the framebuffer size. The rasterizer ensures the the image is not distorted.