Assignmnet 2 Triangles and z-buffering

GMNG 3311

1. Overview

   We draw a triangle frame last assignment. We will try to fill the triangle with colors, which means we will do rasterization for the triangles. If you remember in last assignment, after the viewport transformation, we call rasterizer_wireframe(const Triangle& t) to do rasterization. This time, you will need to implement the function rasterize_triangle(const Triangle& t).

   The function works as follows:

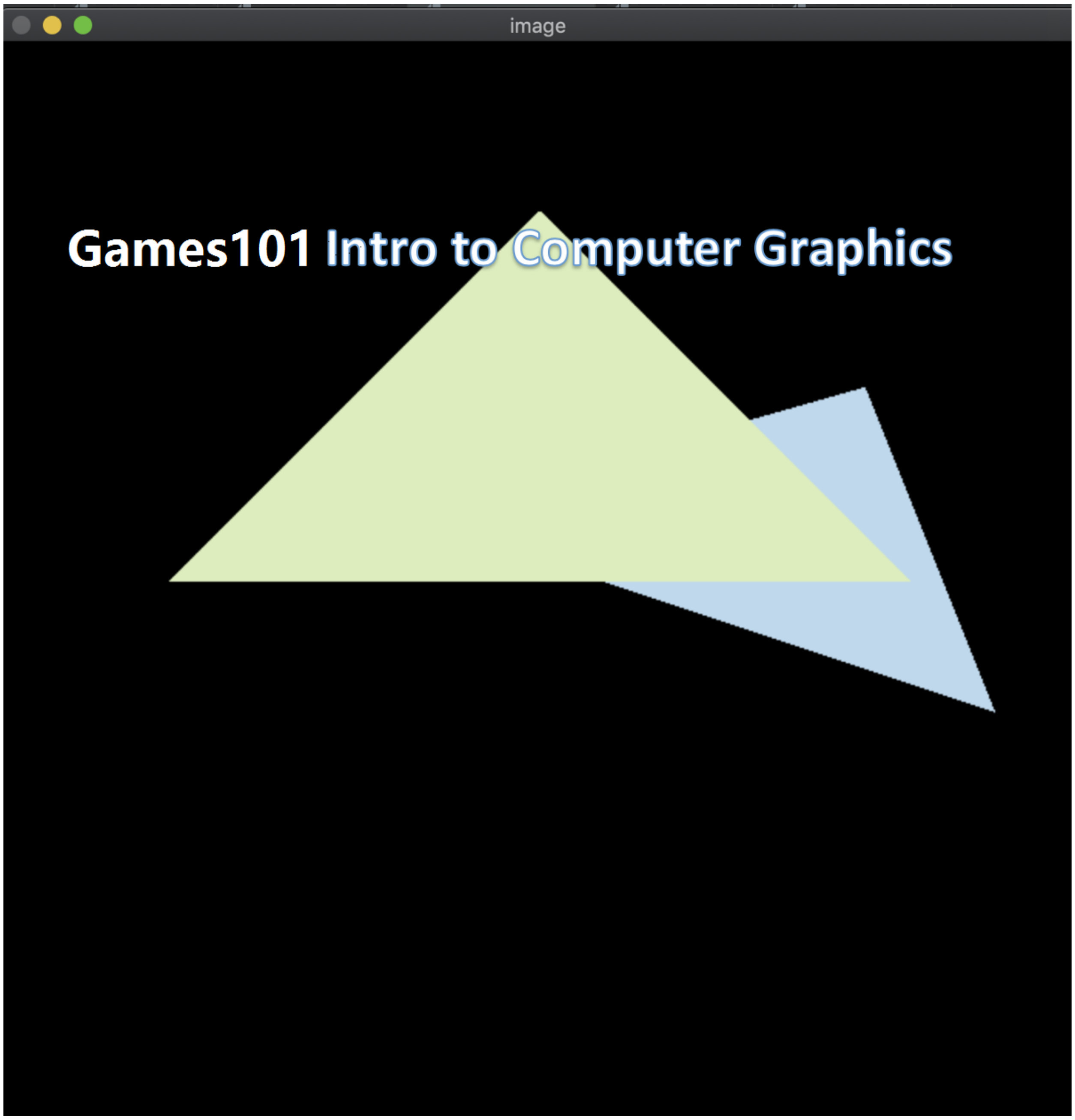
   a. Create a 2D bounding box for the triangle.
   b. Iterate all pixels inside the bounding box, then utilize the coordinate of the center of the pixels in screen coordinate system to check if the point located inside the triangle.
   c. If the point located inside the triangle, compare the interpolated depth value with relative value in depth buffer.
   d. If current point near the camera, set the color to the pixel and update depth buffer.

   You will need to change the following functions:
   rasterize_triangle(): run the rasterization algorithm.
   static bool insideTriangle()： test if the point inside the triangle. You should change the definition of the

function and you can edit the return data type and passing by parameters.

Because we know the depth value of the three vertices for the triangle, so for the pixel inside the triangle, we need to do interpolation to get the depth value. We already provide you with this part. The depth value is assigned to variable z_interpolated.

Focus on how we initialize the depth buffer and the sign of z values. For convenient, we will make sure the z values are positive and larger z values means larger distance from the camera/screen.
In this assignment, you do not need to do MVP transformation. If you implemented right, you will get the following output:



2. Compile
   You may notice the function get_projection_matrix() in main.cpp is empty. You can utilize the implementation if assignment1 to fill this function.

The rest of the compiling step should same with assignment 1.

3. Grading
   [30 points] submit all necessary files and the code can be compiled without any error.
   [20 points] implement the rasterization algorithm.
   [30 points] implement the insideTriangle() function.
   [20 points] implement the z-buffer algorithm and draw the triangles in the correct order.
   [extra credits (10 points)] utilize super-sampling to do antialiasing: we utilize a 2*2 resolution to sample each pixel, calculate the depth value for each sample point.

4. Submission
   Submit all the files under the assignment folder. (remember to include the CMakeLists.txt
   and all the files). Write a README.txt file to inform if you implement the extra part. Zip
   your folder and name with “Name_homework2.zip” and submit it through blackboard.