CSC4140 Assignment 2

Computer Graphics

March 19, 2022

Learn to use VirtualBox and Mathematic Review

This assignment is 10% of the total mark.

Student ID: 120090266

Student Name: Feng Yutong

This assignment represents my own work in accordance with University regulations.

Signature: Feng Yutong

1 Overview

The task of this assignment is to complete following tasks:

Draw a single color triangles

Antialiasing

Transforms

Barycentric coordinate

Texture mapping

Mipmapping

This assignment implements different sampling and texture mapping method. The report will analyse their advantages and disadvantages by comparing antialiasing power, time and space.

2 Implement and Results

2.1 Draw a single color triangles

This task is to implement point sampling. It colors a pixel if its center is inside a triangle. It calls **inside** function to check whether a pixel is inside the triangle by three cross products. Since the winding order of vertices can be clock-wise and counter-clockwise, the results of three cross products having the same sign indicates inside a triangle. Only pixels in a bounding box, which is limited by $x_{min}, y_{min}, x_{max}, y_{max}$, are checked to same time. Result is shown in Figure 1.

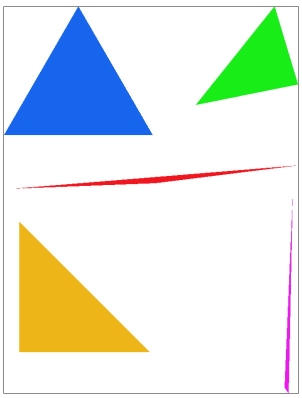

Figure 1: test4.svg

2.2 Antialising

Since sample rate of point sampling is one (very slow), jaggies appear at where signals are changing too fast. One way to solve it is supersampling (MSAA), which increases sample rate. It approximates the effect of one-pixel box filter by sampling multiple locations within a pixel and averaging their values. The locations of subpixel can be presented as

$$x_{sub} = x + \frac{0.5 + x_{step}}{\sqrt{sample\_rate}}, x_{step} = 0, 1, ...\sqrt{sample\_rate}$$

$$y_{sub} = y + \frac{0.5 + y_{step}}{\sqrt{sample\_rate}}, y_{step} = 0, 1 \ldots \sqrt{sample\_rate}$$

Once the formula of subpixel is found, the point sampling can be regarded as a special case of MSAA (with sample rate = 1). Therefore, the main procedure is the same as task1. Since one pixel is divided into more subpixel, common way to solve it is dynamically updating the size of frame buffer (Version 1 in rasterizer.cpp). However, it costs more space with the increase of sample rate. **To avoid consuming extra memory** (Version 2 in rasterizer.cpp, only implement MSAA rasterize_triangle), the program interpolates the triangle color and boundary color as follows:

$$color = \frac{cnt_{inside}}{sample\_rate} * color + \frac{sample\_rate - cnt_{inside}}{sample\_rate} * boundary\_color$$

. Since some polygon consists of many triangles, the procedure repeats three times to eliminate the "transparent pixels" on the triangle boundary but inside polygon. The program exchanges time for space.

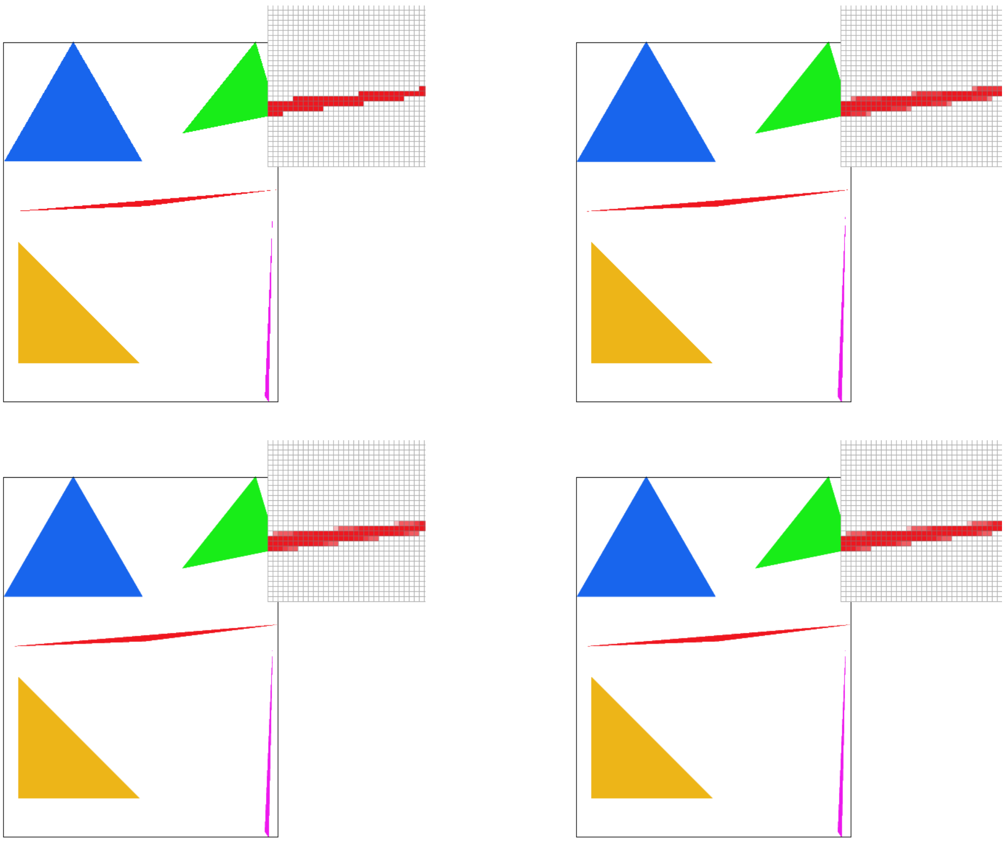

Figure 2: test4.svg under four sample rate: 1, 4, 9, 16

2.3 Transforms

This task is to implement a translation, scaling and rotation matrix to achieve the transformation of a robot. Figure 3 is a series of a robot tranforms: the robot jumps and tries to throw something to the right side and cheers happily after throwing it away.

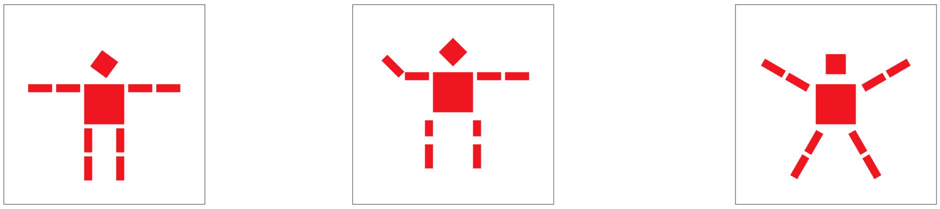

Figure 3: robot transforms

2.4 Barycentric coordinate

Barycentric coordinate system is a coordinate system where the location of a point is specified by reference to a simplex. In two dimensional space, any vertex on the same plane can be expressed as a linear combination of the verices of a triangle, i.e., $(x, y) = \alpha V_A + \beta V_B + \gamma V_C, \alpha + \beta + \gamma = 1$. V_A, V_B, V_C can be positions, texture, coordinates, color and other material attributes. For example, linear interpolation of color at vertices of a triangle (see Figure 4).

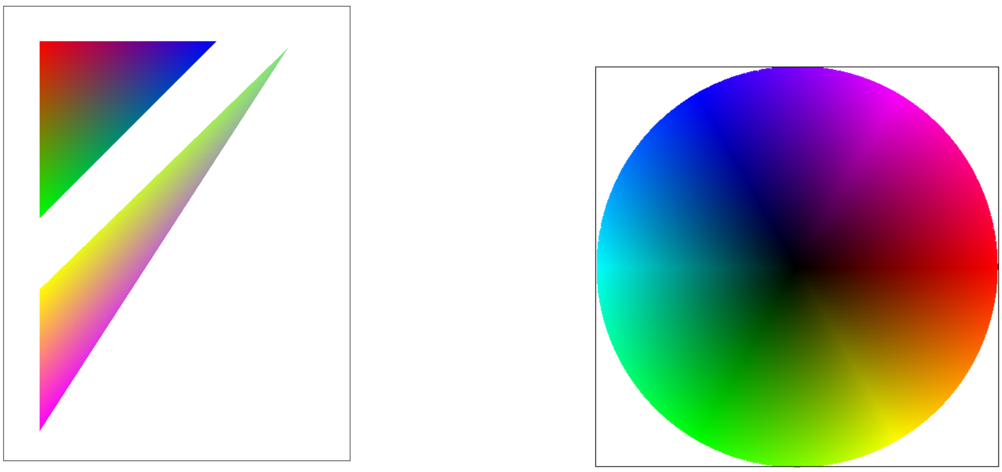

Figure 4: color gradients in a triangle and a circle

2.5 Texture mapping

Pixel sampling is similar to point sampling. The differences are that pixel sampling is full-resolution and color is replaced by texture resutling from a mapping relation. Two methods are used for texture mapping: nearest sampling and bilinear sampling. Nearest sampling chooses the texture of whose location is closest to the barcyentric coordinates of the pixel. Bilinear sampling linearly interpolates the texture of the pixel's four neighbours. Bilinear sampling usually gives better results (see Figure 5 & 6). Bilinear sampling generates blurred image but smooth curve at distinct boundary because it takes weighted sum of nearby pixel texture, while nearest sampling can be blocky or miss some texture for its texture can be greatly different from its nearby but not nearest pixel. However, bilinear sampling takes about twice the time of nearest sampling (but both at the same time level) and the differences becomes small with the increase of sample rate (seed Figure).

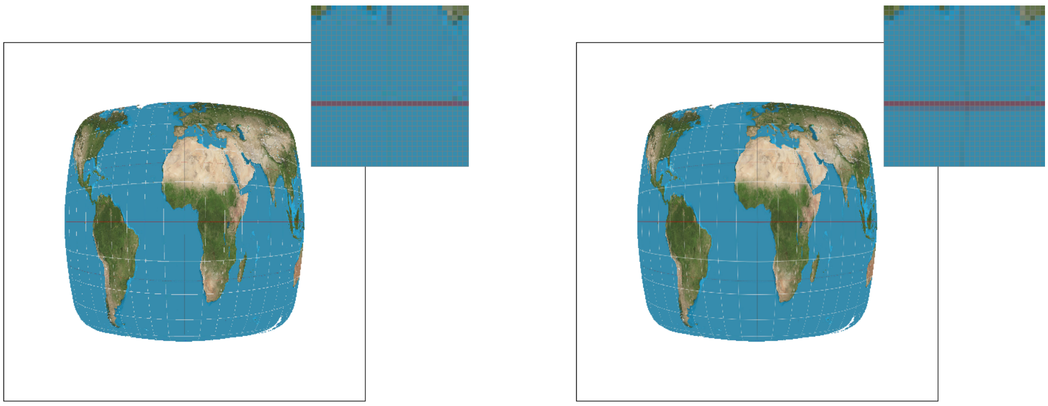

Figure 5: nearest (left) and bilinear (right) sampling at 1 sample per pixel

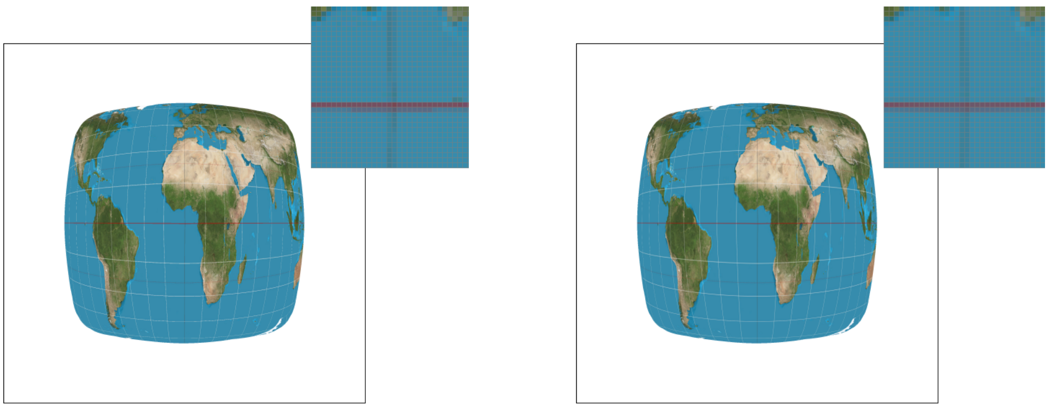

Figure 6: nearest (left) and bilinear (right) sampling at 16 sample per pixel

2.6 Mipmapping

Mipmapping is an antialiasing method, which renders an object depending on the distance between the object and camera. When image is closed to camera or rendered largely, larger texture are used. Implement:

1. Calculate the uv barycentric coordinates of (x,y), (x+1, y), (x, y+1) by formula in task 4. Then calculate the differences and scale up different vectors by the width and height.

2. Fix the the mipmap level D: $D = \log_2 L, L = max(\sqrt{(\frac{du}{dx})^2 + (\frac{dv}{dx})^2}, \sqrt{(\frac{du}{dy})^2 + (\frac{dv}{dy})^2})$. For zero level, D = 0; for nearest level, D = round(D); for trilinear interpolation, D = ceil(D) and floor(D).

3. Do nearest sampling or bilinear sampling on the level D.

4. For trilinear interpolation, interpolate the results of level ceil(D) and level floor(D).

Mapping a picture of colorful tiles on a building (see Figure 7). Results are shown by Figure 8, 9 and 10, we can get following conclusion:

Level	Antialiasing power	Speed (Under the same sample method)	Space	Application
Zero level	jaggies on the boundries and distinct pattern	Quickest,$O(wh)$	$O(wh)$	Use for classification
Nearest Level	distinct but smooth boundaries and a little blurred pattern	O(1) for getting level, still $O(wh)$	$log(wh)O(wh)$	Use for elevation or raw slope values
Trilinear interpolation	very smooth boundaries and blurred pattern	About twice the time of nearest level, still $O(wh)$	$log(wh)O(wn)$	Use for smoothing continuous data

Table 1: comparison between different texture mapping methods

Figure 7: original picture for texture mapping

Figure 8: left: P_NEAREST, L_ZERO; right: P_NEAREST, L_NEAREST

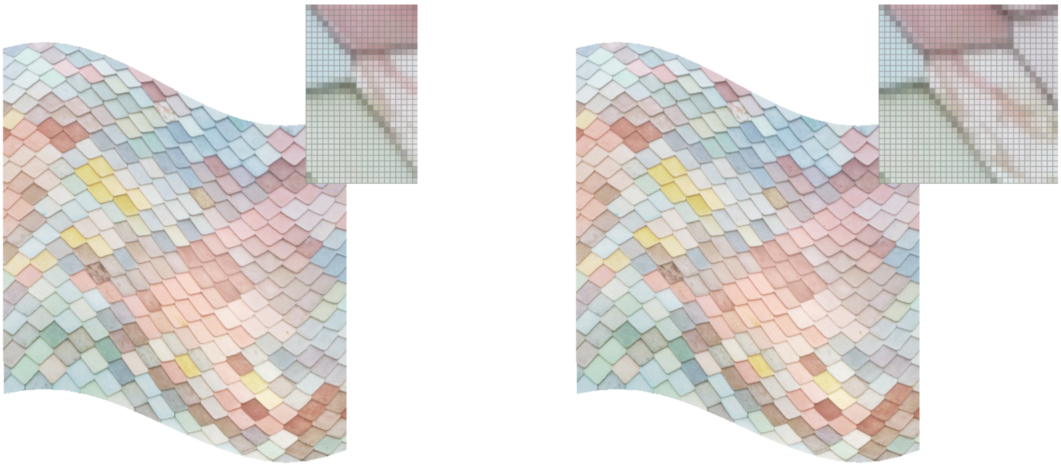

Figure 9: left: P_LINEAR, L_ZERO; right: P_LINEAR, L_NEAREST

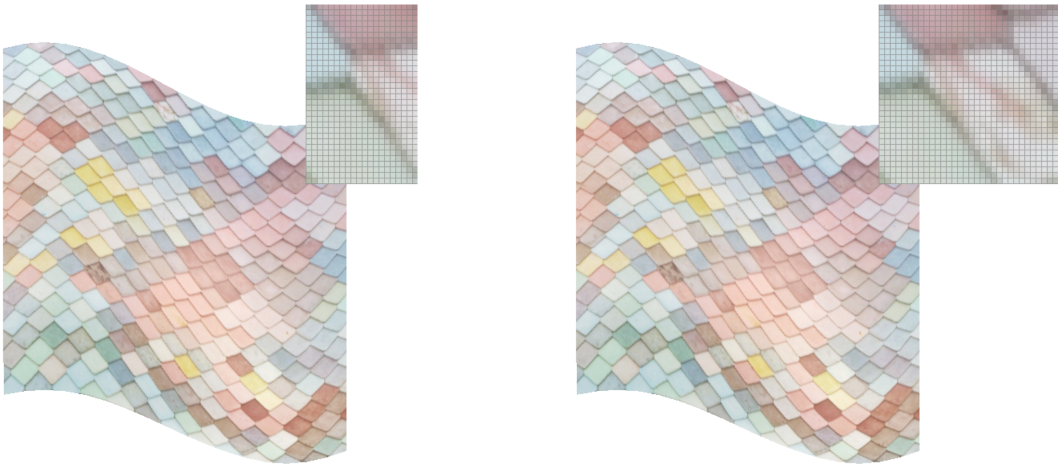

Figure 10: (Extra) trilinear: left: nearest; right: linear

3 Conclusion

The project finishes 6 tasks and passes test cases except cases considering texture map (this may due to different implement approach and precision of floating number) and opacity. There are several chanllenges and interesting happening.

1. When implementing MSAA with no extra memory, first I try to solve it by just using the average sum of the subpixel, but then I find black boundaries. Inspiring from the texture mapping, I use the number of pixel inside the triangle to do interpolation. Since some pattern consists of muliple triangles, "transparent boundaries" appears instead. That is because program rasterizes triangle one by one. For the boundaries of triangle inside the pattern, it still considers the neigbour pixel to be white. Thus, increasing the times of rasterizing can diminish the differences. In practice, three times is suitable for all svg under basic folder.

2. During bilinear interpolation of texture mapping, I first interpolate the results of four texel on xy coordinates (just like interpolation of color), then I find all the uppper half triangle of a rectangle become a little dark. Barycentric coordinate seems to have linear relation with the xy coordinates. However, they change after projection transformation. (Task 4 is special because it ignores depth and there is no projection) Therefore, interpolation should be done before converting coordinates.

I gain better understanding of rasterize and texture mapping from this project and practice the knowledge learned in class.