CSC4140 Assignment III

Computer Graphics

March 19, 2022

Rasterization

This assignment is 5% of the total mark.

Strict Due Date: 11:59PM, Mar 19^{th}, 2022

Student ID: 118010335

Student Name: Wei WU

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

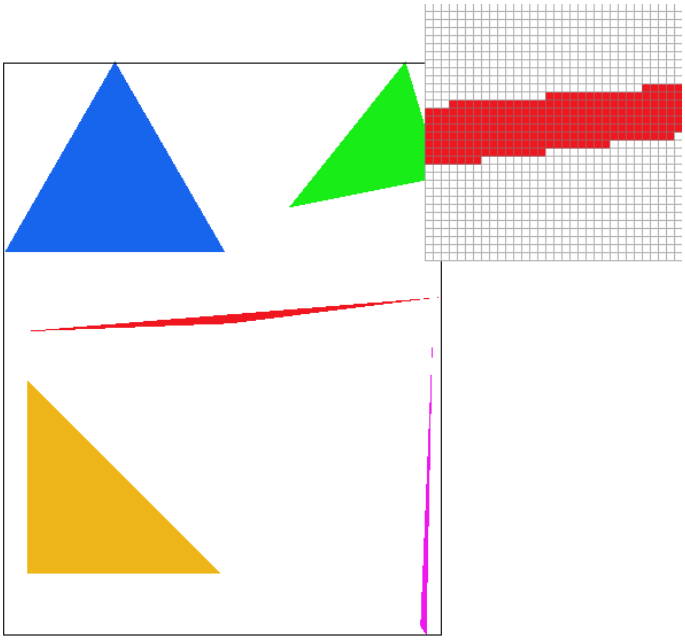
Signature:

1 Overview

This assignment aims to rasterize points, lines, and triangles in .svg files. It consists of a .svg parser, a .svg renderer, and a set of rasterizers. It supports three types of triangles: single-colored triangles, interpolated-colored triangles, and textured triangles. Furthermore, it supports transformation, mipmapping, and antialiasing including SSAA, MSAA, and FXAA.

2 Task 1

To rasterize a triangle, firstly, the axis-aligned bounding box of the triangle is identified according to its three vertices. Then, the rasterizer tests every pixel inside the bounding box if it is inside the triangle. This algorithm is efficient since it only checks the pixels inside the bounding box of the triangle.



3 Task 2

The rasterizer implements three types of antialiasing algorithms: SSAA, MSAA, and FXAA.

3.1 SSAA

In SSAA, an up-scaled sample buffer is used. The .svg is firstly rasterized in a high resolution and saved to the sample buffer. Then, the picture in the sample buffer is down-sampled into the render buffer.

Supersampling is useful since the number of sampling points is increased. During downsampling, the sampling points are averaged, making the image smoother.

To implement SSAA, the original rasterization pipeline is modified. First, an up-scaled sample buffer is added. Second, the rasterizer renders a up-scaled image into the sample buffer. Third, the supersampled image is downsampled into the target resolution by averaging all corresponding sampling points for each pixel. Besides, the size of the sample buffer is adjusted when the supersampling rate changes.

3.2 MSAA

In MSAA, there is no up-scaled sample buffer, i.e., the sample buffer is of the same size as the render target buffer. For each pixel inside each triangle, the supersampling points are directly averaged and stored into the sample buffer. On the edge of each triangle, the supersampling points are also blended with the background, either the white background or other triangles.

MSAA does not consume additional memory for supersampling. Furthermore, it only downsamples the pixels within the bounding boxes of the triangles, while SSAA must downsample the whole image. This makes MSAA more economic and efficient than SSAA.

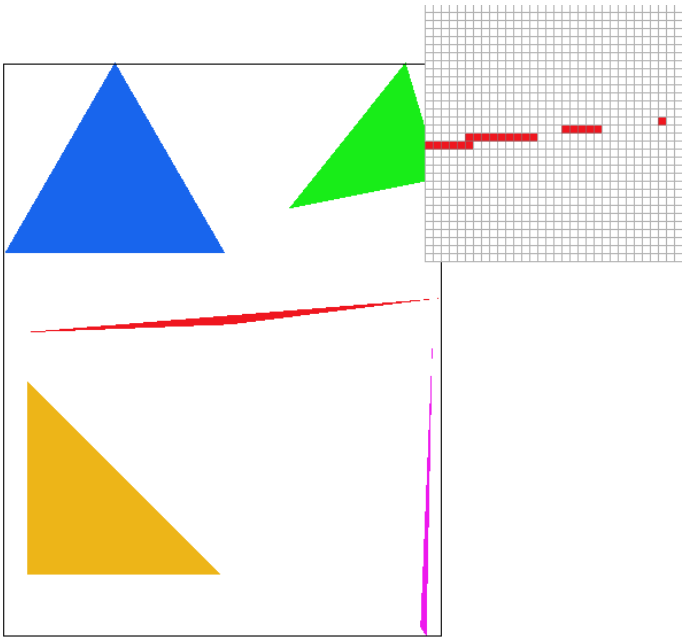
The drawbacks of MSAA is that it is not as accurate as SSAA. Also, in theory, when the number of triangles is large, it does more downsampling calculation than SSAA.

3.3 FXAA

FXAA does not do supersampling. Hence, it does not require extra memory as well. FXAA is a post-processing technique that identifies and blurs drastic color changes in the image. Therefore, it is much faster than SSAA and MSAA. Its disadvantages are also obvious: due to lack of sampling, the result is not as good as supersampling methods.

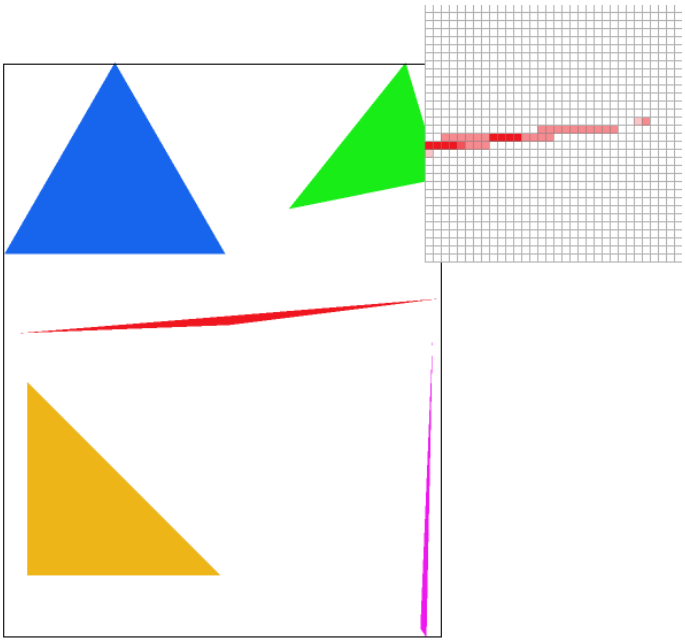
FXAA needs to process the whole image. Therefore, when the number of triangles is small, it may be slower than SSAA/MSAA.

3.4 Original Image



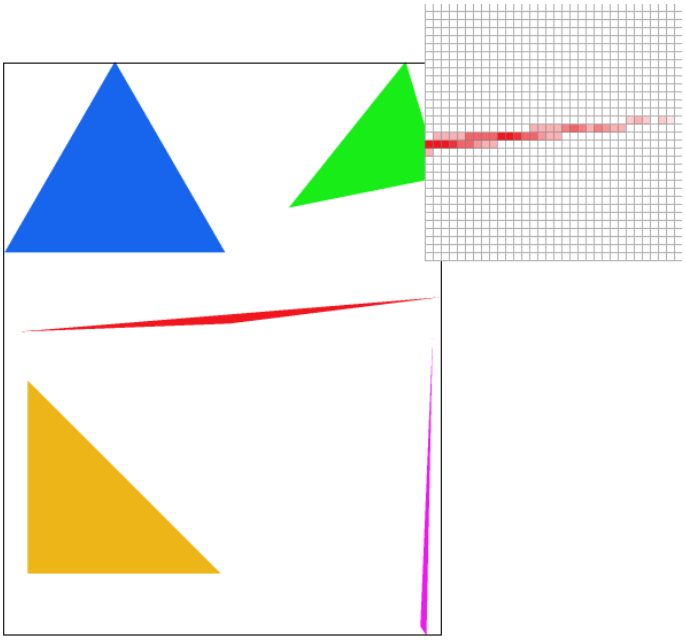
Performance: 5.695 ms / frame

3.5 SSAA 4x



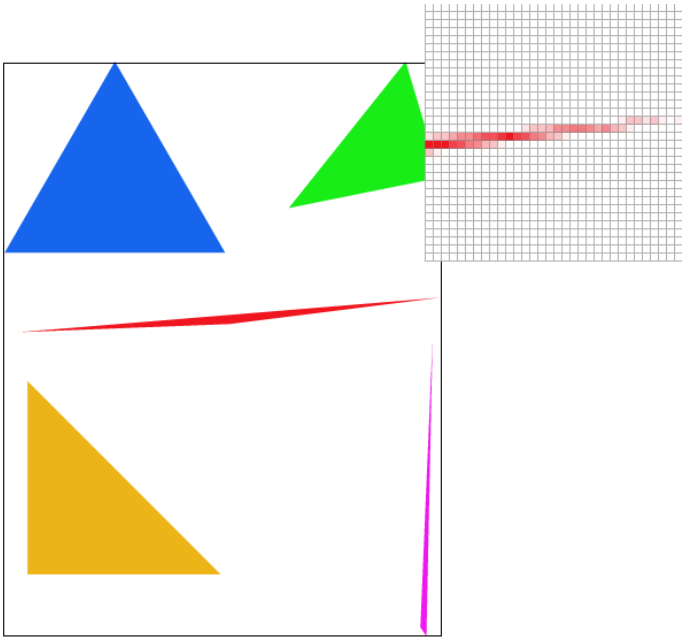
Performance: 12.364 ms / frame

3.6 SSAA 9x



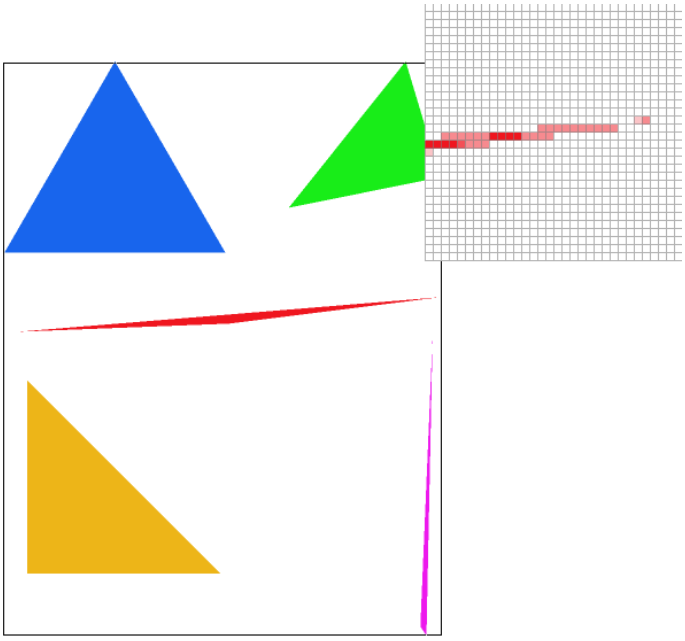
Performance: 26.849 ms / frame

3.7 SSAA 16x



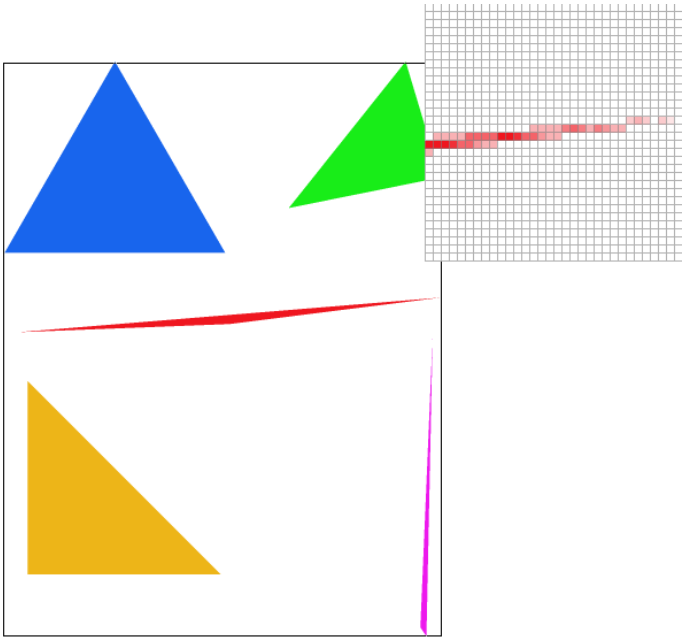
Performance: 50.192 ms / frame

3.8 MSAA 4x



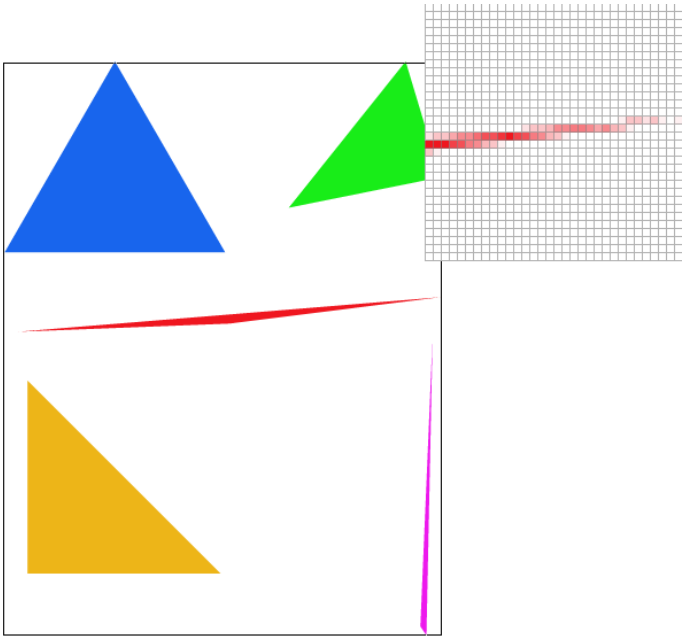
Performance: 8.091 ms / frame

3.9 MSAA 9x



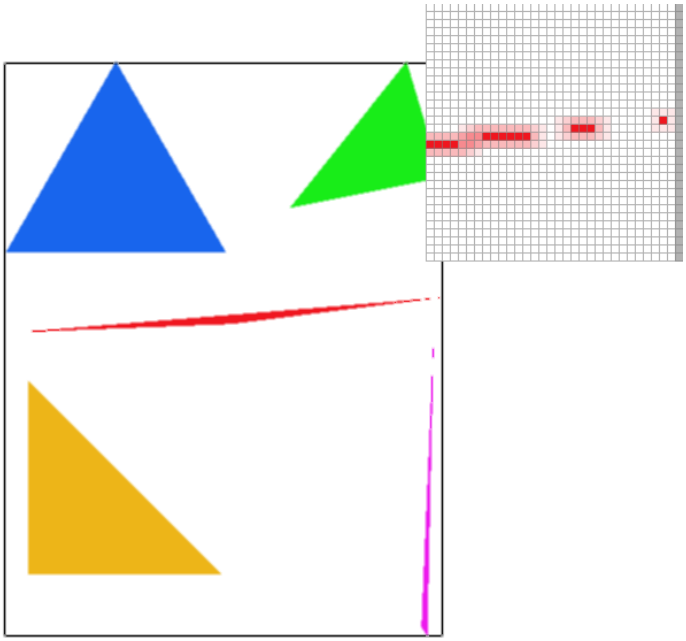
Performance: 9.755 ms / frame

3.10 MSAA 16x



Performance: 11.632 ms / frame

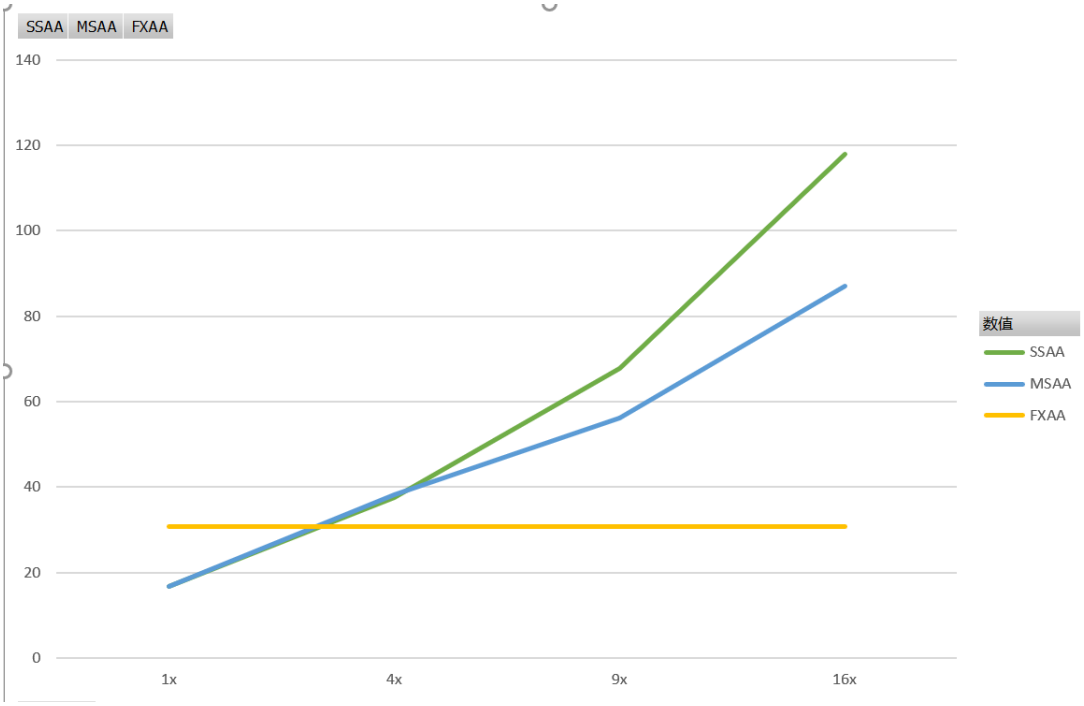
3.11 FXAA



Performance: 21.154 ms / frame

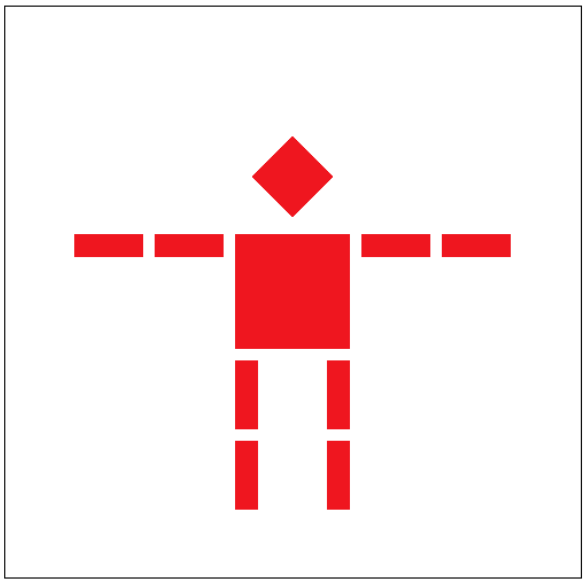
3.12 Performance

The performance of texmap/test2.svg is



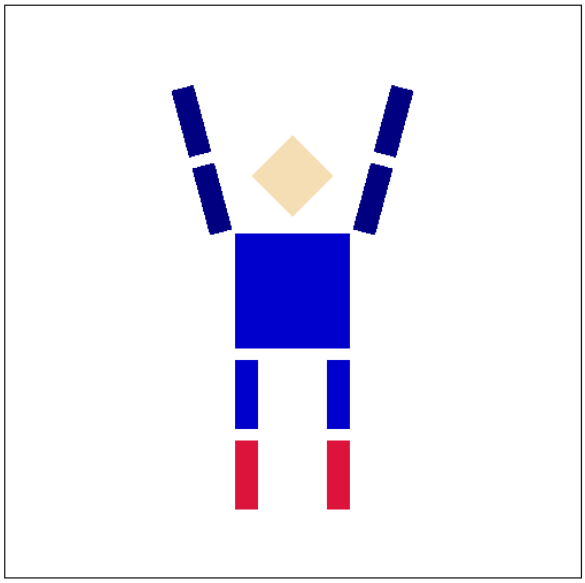
4 Task 3

4.1 Original Cubeman



4.2 My Robot

By changing color, translation, and rotation, the following image is generated:



5 Task 4

For any triangle, barycentric coordinates can be used to express any point located in the plane containing the triangle, in the form of:

$$p = \alpha\ A + \beta\ B + \gamma\ C$$

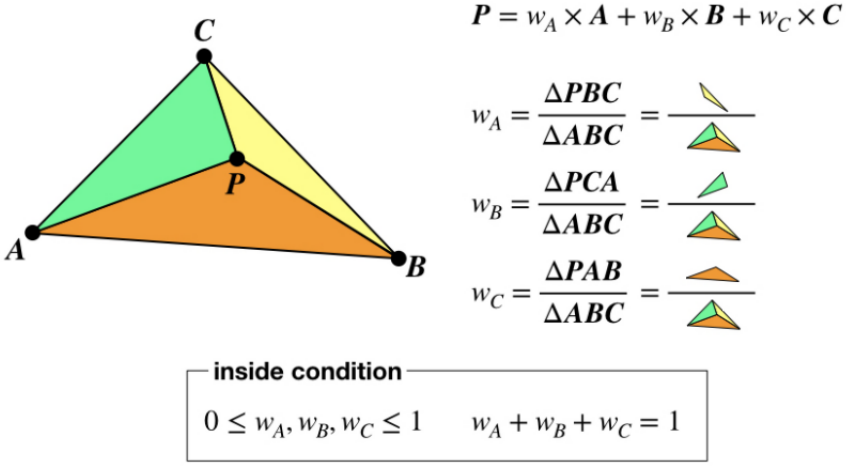
The ratio of the coordinates is equal to the ratio of the areas of opposite small triangles.

If the point is inside the triangle, the sum of its barycentric coordinates is 1, i.e.

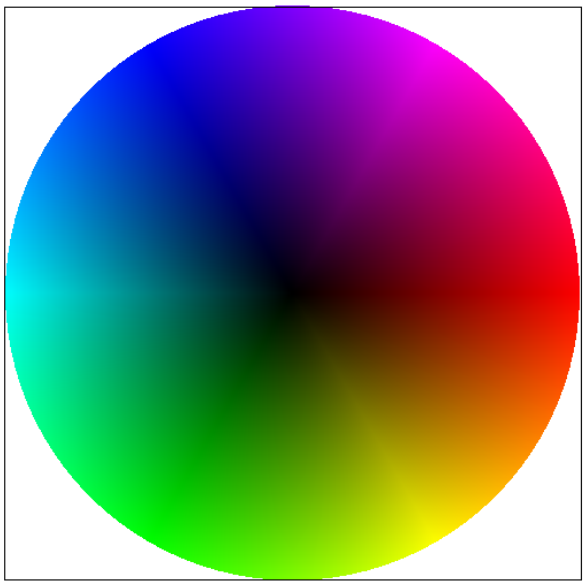
$$\alpha + \beta + \gamma = 1$$

And also,

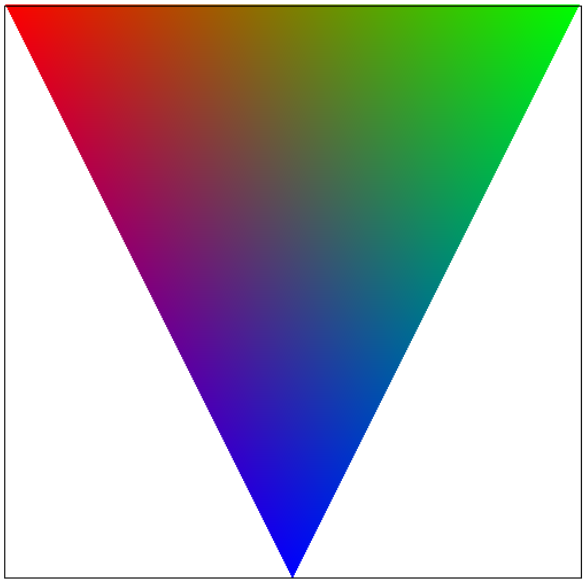
$$0 <= \alpha\ ,\ \beta\ ,\ \gamma <= 1$$



5.1 basic/test7.svg



5.2 basic/test9.svg



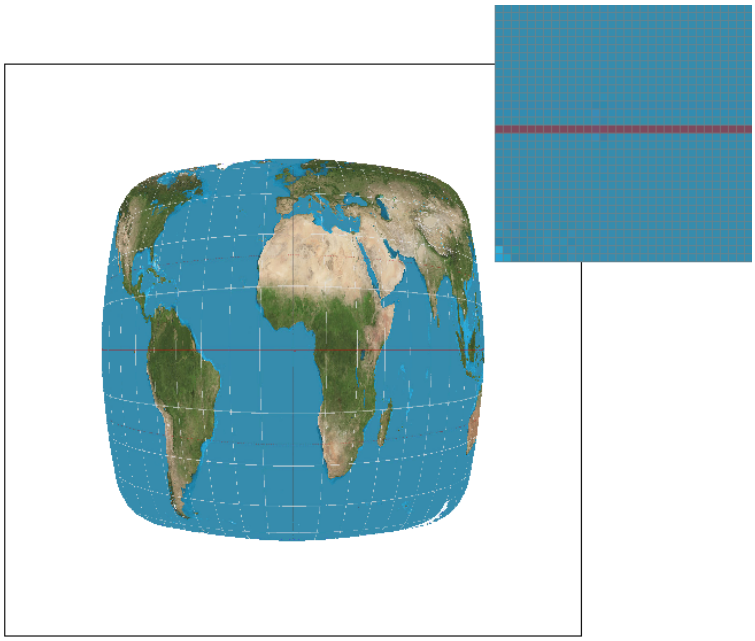
6 Task 5

For each point in the textured .svg files,there are two sets of coordinates. one is view coordinates, the other is uv coordinates. View coordinates correspond to position in the canvas, while uv coordinates correspond to position in the texture map.

For a triangle, if both view and uv coordinates of each vertex are known, we can use barycentric coordinates to estimate the uv coordinates of any point inside it.

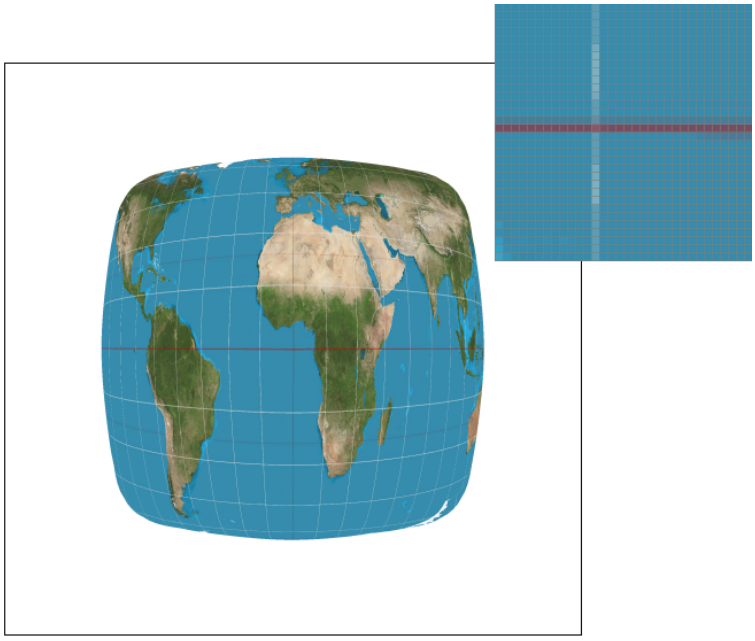
There are two types of pixel sampling. One is nearest pixel sampling, where we find and use the nearest pixel in the texture map to the calculated uv coordinates of a point. The other is bilinear pixel sampling, where we find the four pixels bounding the calculated uv coordinates of a point in the texture map and then calculate the weighted average according to distance.

6.1 texmap/test2.svg Nearest SSAA 1x



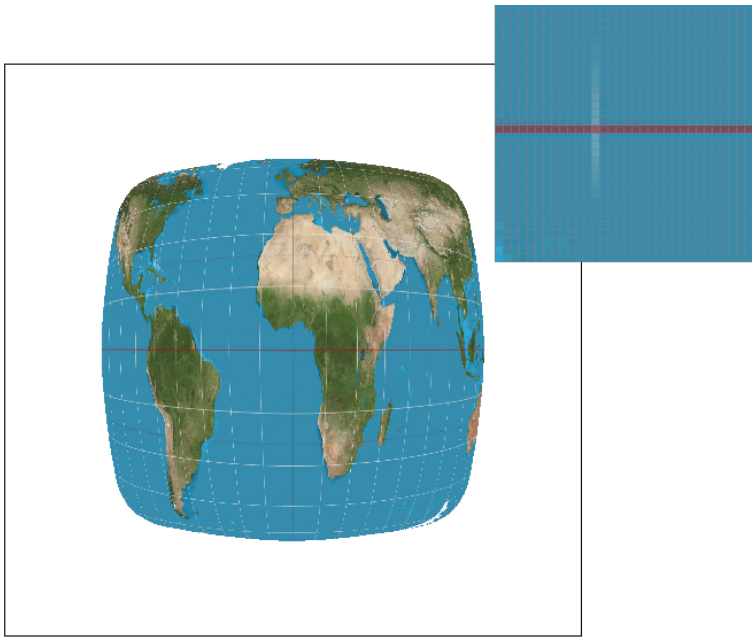
Performance: 19.250 ms / frame

6.2 texmap/test2.svg Nearest SSAA 16x



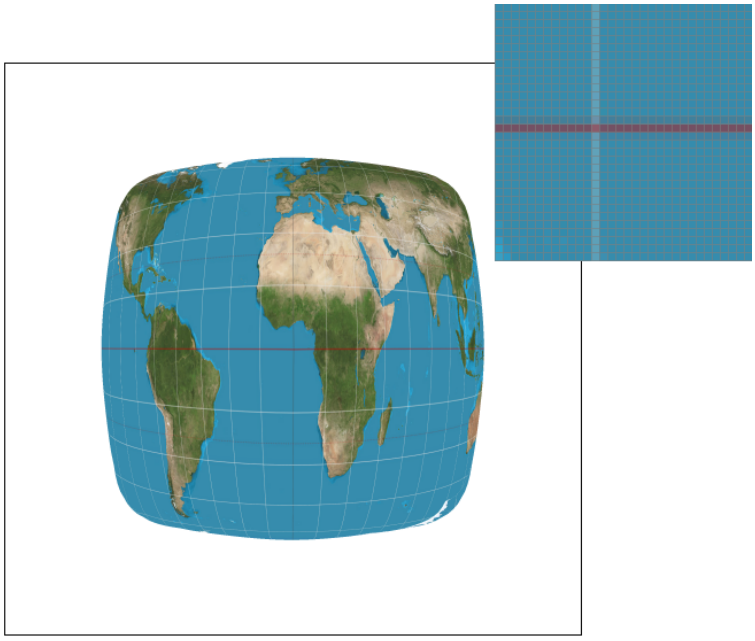
Performance: 162.455 ms / frame

6.3 texmap/test2.svg Bilinear SSAA 1x



Performance: 21.230 ms / frame

6.4 texmap/test2.svg Bilinear SSAA 16x



Performance: 194.007 ms / frame

In the pixel inspector, when using SSAA 16x, there is a clear white line for both nearest and bilinear pixel sampling. When using nearest pixel sampling with SSAA 1x, there is no white line at all. However, when using bilinear pixel sampling with SSAA 1x, there is a faint white line.

Therefore, the quality of bilinear pixel sampling with SSAA 1x is somewhere between nearest pixel sampling with SSAA 1x and neaerest pixel sampling with SSAA 4x.

Although bilinear pixel sampling is slightly slower than nearest pixel sampling, it is much faster and more memory-saving than using SSAA.

In the high frequency areas of an image, i.e., in the areas with sharp color changes, nearest pixel sampling can be erroneous, since it loses information in the surrounding texels. In these areas, bilinear pixel sampling gives better quality.

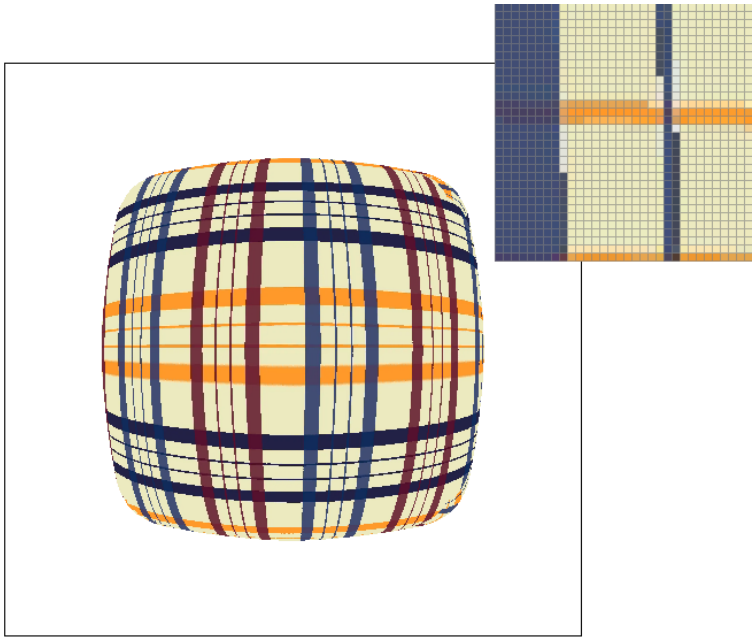
7 Task 6

When the texture needs to be downsampled, if we only sample one texel from several texels, much information is discarded. If we sample and average a number of texels at runtime, there is significant performance overhead. Therefore, we can calculate mipmaps of multiple levels in advance. Then, at runtime, according to the change rate of uv coordinates at each sampling point, we can pick the most suitable levels of mipmap for sampling.

There are two different sampling strategies. One is nearest level sampling, which uses the nearest mipmap level for sampling. The other is bilinear level sampling, which samples the two nearest mipmap levels and then calculates weighted average.

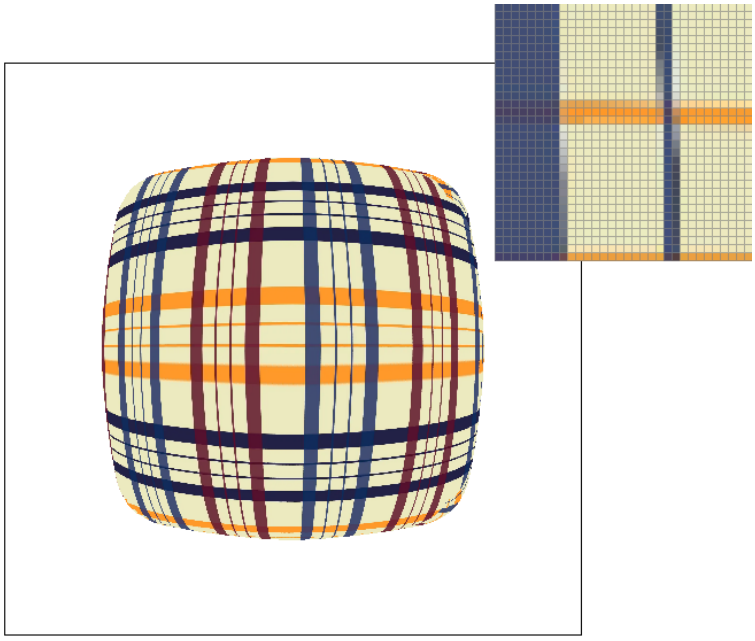
There are three ways to improve the sampling quality, namely pixel sampling, level sampling, and supersampling. Bilinear pixel sampling is only a bit slower than nearest pixel sampling. It needs no extra memory and improves the image quality moderately. Level sampling using mipmaps, especially bilinear level sampling, is slower than pixel sampling. It requires additional memory of 1/3 the size of the original texture map. However, it provides better image than pixel sampling. SSAA is the slowest way. In a sense, it is similar to computing the mipmap at runtime. Also, it requires additional memory proportional to sample rate and render target size. However, it has the strongest antialiasing power.

7.1 texmap/my_test.svg L_ZERO and P_NEAREST



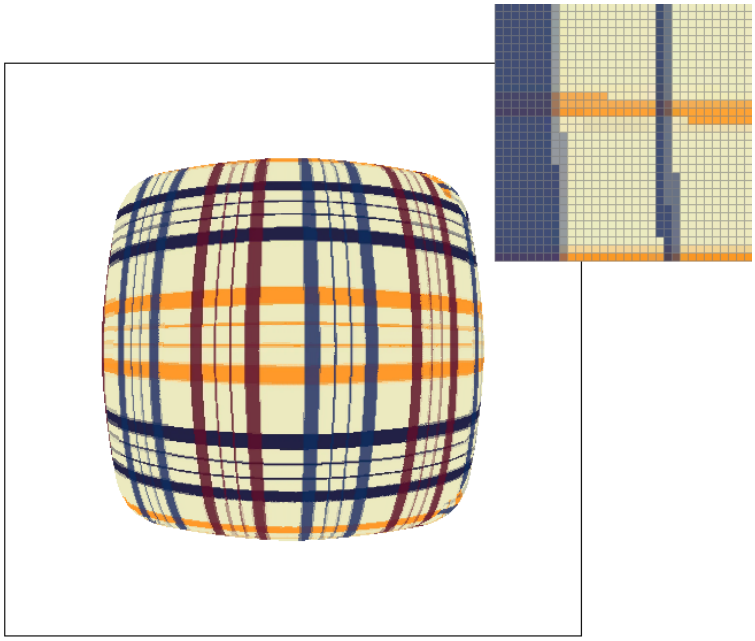
Performance: 22.203 ms / frame

7.2 texmap/my_test.svg L_ZERO and P_LINEAR



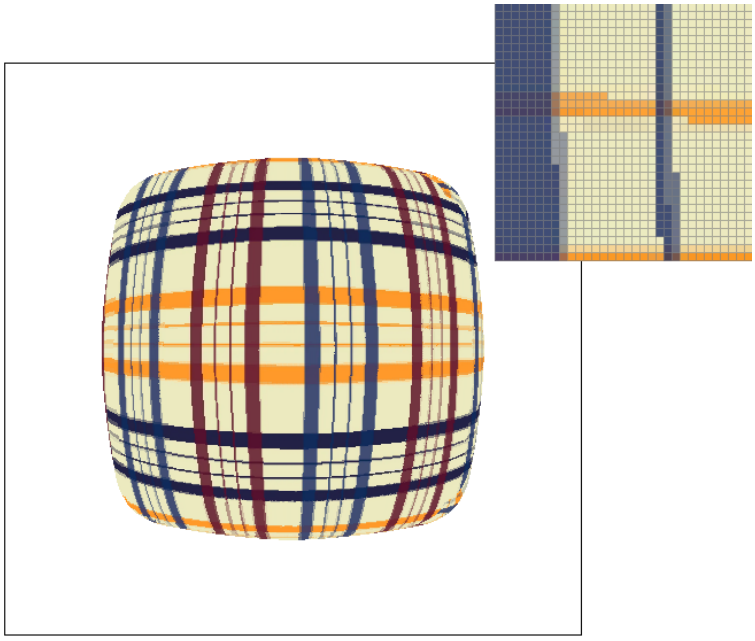
Performance: 24.145 ms / frame

7.3 texmap/my_test.svg L_NEAREST and P_NEAREST



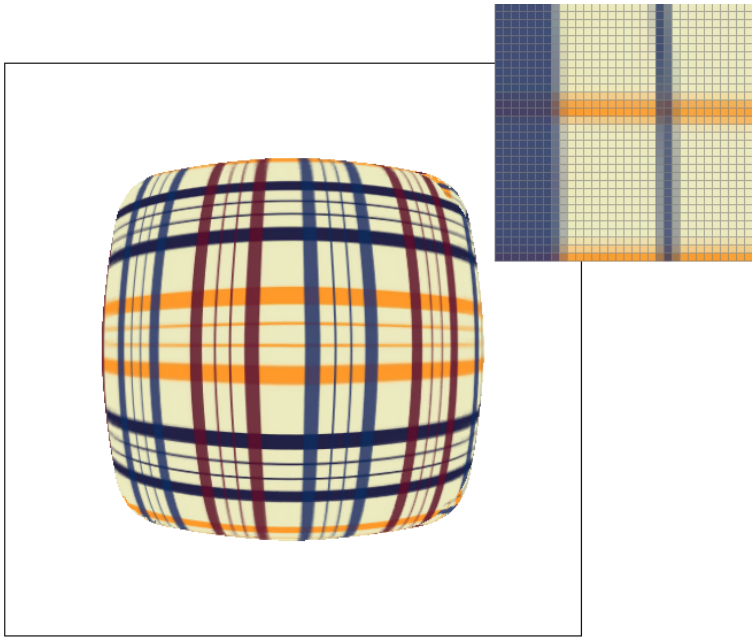
Performance: 23.586 ms / frame

7.4 texmap/my_test.svg L_NEAREST and P_LINEAR



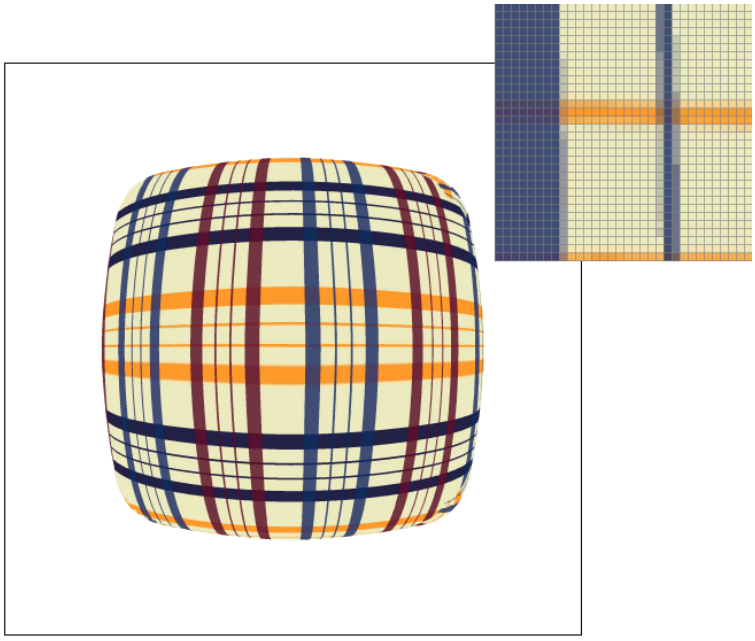
Performance: 24.771 ms / frame

7.5 texmap/my_test.svg L_LINEAR and P_LINEAR



Performance: 33.946 ms / frame

7.6 texmap/my_test.svg L_ZERO and P_NEAREST with SSAA 16x



Performance: 167.053 ms / frame