CS184: Computer Graphics

Sping 2022

Assignment 1: Rasterizer

Prince Wang

Overview

An overview of the project, your approach to and implementation for each of the parts, and what problems you encountered and how you solved them. Strive for clarity and succinctness. On each part, make sure to include the results described in the corresponding Deliverables section in addition to your explanation. If you failed to generate any results correctly, provide a brief explanation of why. The final (optional) part for the art competition is where you have the opportunity to be creative and individual, so be sure to provide a good description of what you were going for and how you implemented it. Clearly indicate any extra credit items you completed, and provide a thorough explanation and illustration for each of them.

Task 1: Drawing Single-Color Triangles

How I rasterized triangles:

My algorithm is very simple, it is essentially the same algorithm as described in lecture 2:

- Step 1: I calculate the smallest bounding boxes that bounds the triangle that I am about to rasterize
- Step 2: I iterate over all the pixels in this bounding box and test whether each pixel is inside the triangle or outside.

The point-in-triangle test I implemented follows this formula:

$$P_i = (X_i, Y_i)$$

$$dX_i = X_{i+1} - X_i$$
$$dY_i = Y_{i+1} - Y_i$$

$$L_i(x, y) = -(x - X_i)\, dY_i + (y - Y_i)\, dX_i$$
$$= A_i\, x + B_i\, y + C_i$$

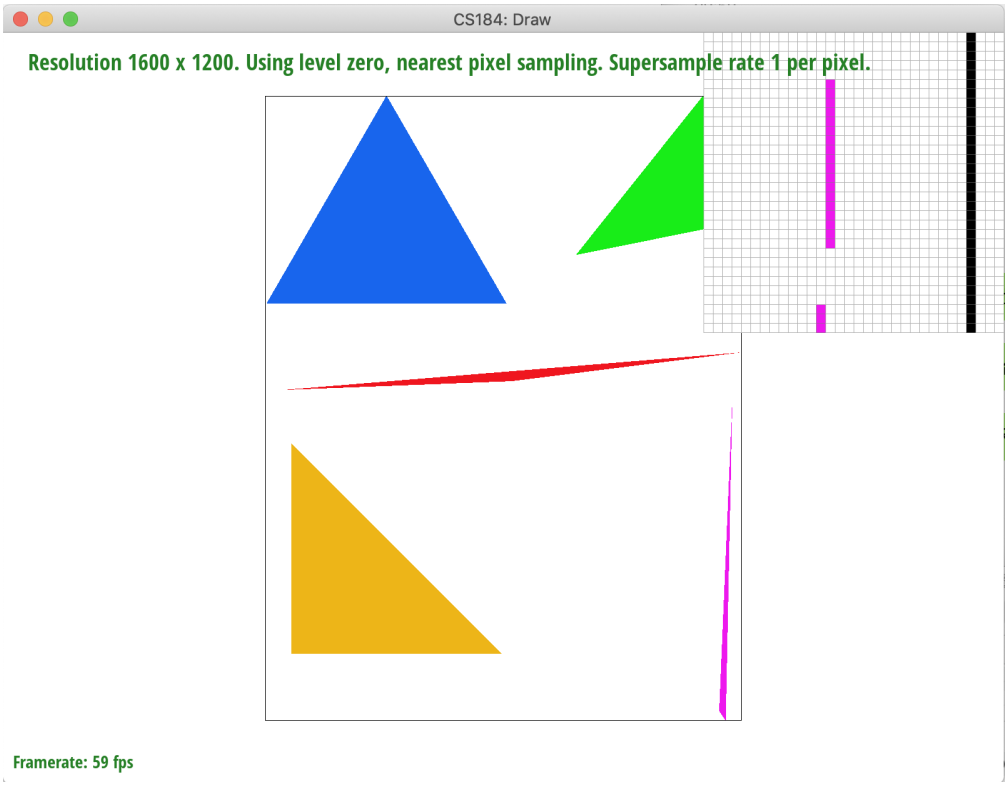
$$inside(sx, sy) =$$
$$L_0(sx, sy) > 0 \;\&\&$$
$$L_1(sx, sy) > 0 \;\&\&$$
$$L_2(sx, sy) > 0;$$

To handle special cases, I also set a point as "in triangle" if all three conditions are negative.

How My Algorithm is no worse than checking sample within bbox:

Since my algorithm is literally the bounding box method, it is no worse.

test4.png



Task 2: Antialiasing by supersampling

Algorithm and data structure

- Step 1: I clear the previous buffer and I initializes a sample buffer of size (height *sqrt(sample_rate) + width* sqrt(sample_rate))
- Step 2: I then supersample into this enlarged sample buffer. For each triangle I used a minimal bounding box to locate them and iterate through the pixel. This time because we have a larger sample buffer for supersampling, we store more values about the colors for each triangle.
- Step 3: When we resolve to targeted frame buffer, for the color value of each pixel on the frame buffer, we set the color to the average over the local sqrt(sample_rate) * sqrt(sample_rate) pixels on the sample buffer.
- **data structure used:** the given sample buffers. We used two: a sample buffer of dimension (height *sqrt(sample_rate) + width* sqrt(sample_rate)) and a final frame buffer of size(height, width)

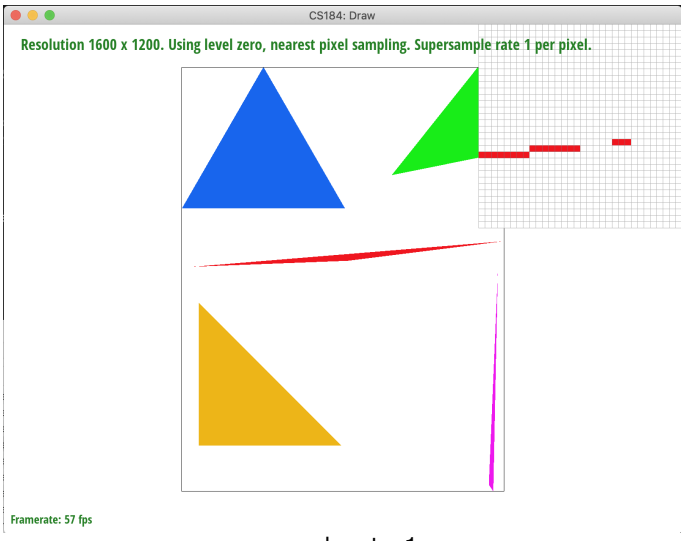
Why is supersampling useful and how it helped antialiasing?

Supersampling is useful because it approximates the effect of convolving a 1-pixel box filter. This filtering smoothens the image and filters out high frequencies in the image prior to sampling, thus antialiasing the image.

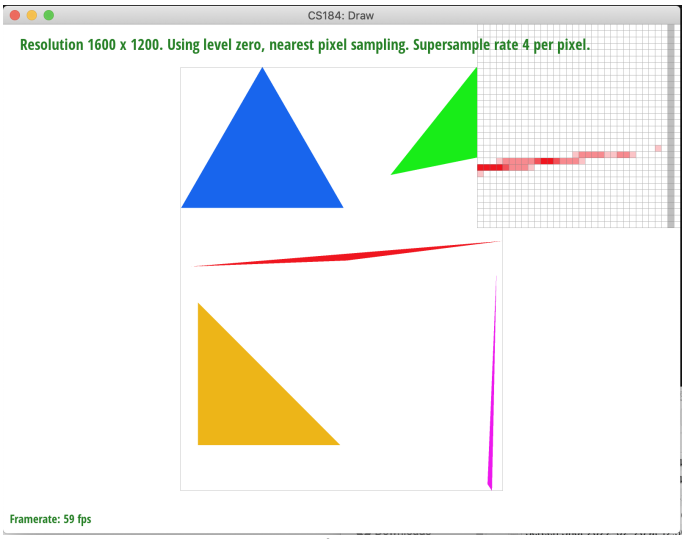
Modifications to the pipeline

I essentially modified step 5 of the rasterizer pipeline. I changed the size of the internal buffer, and when I populate the screenbuffer I did averaging.

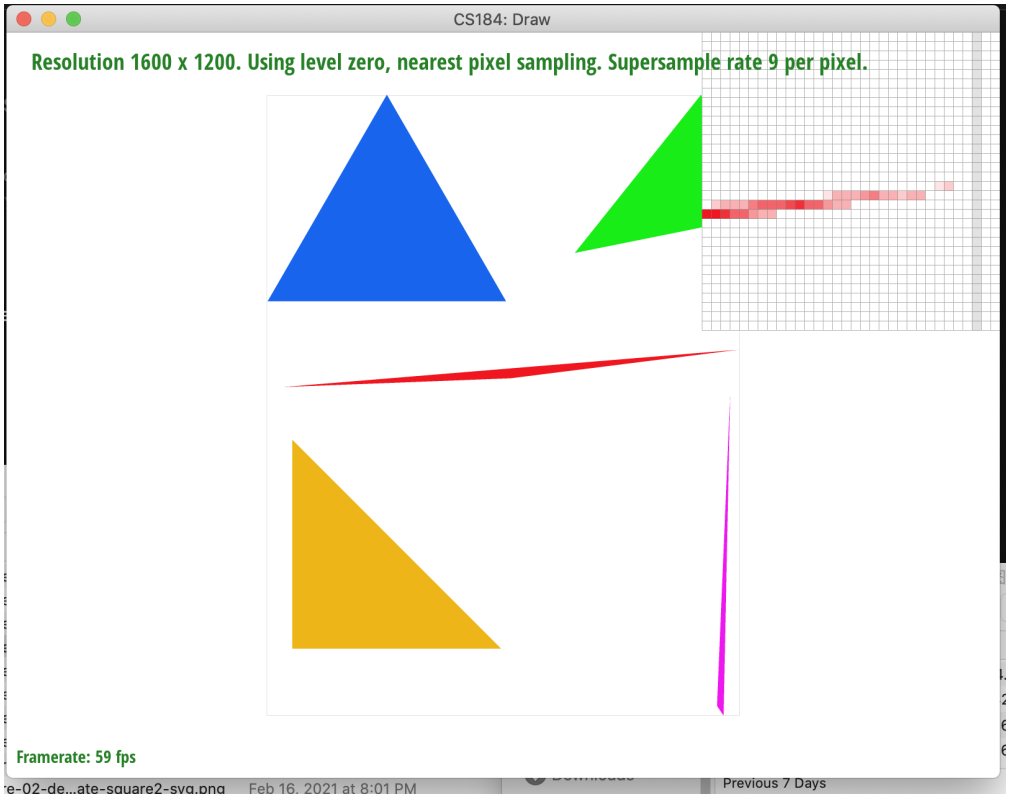
Comparison of test4.svg



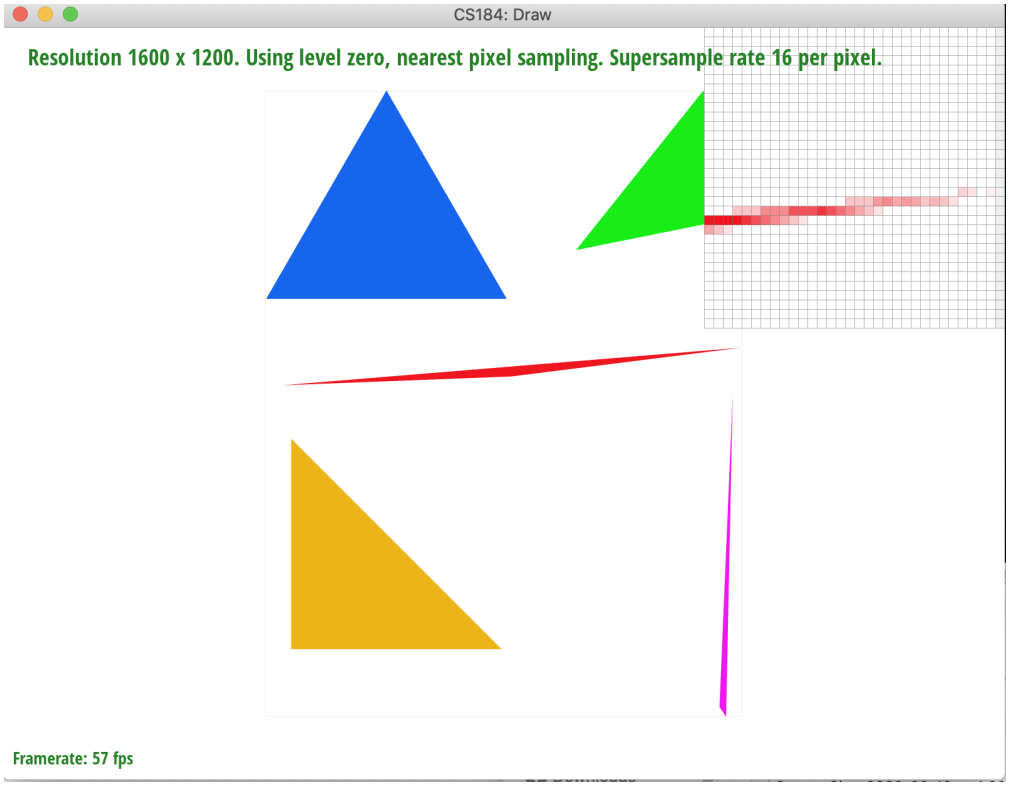
sample rate: 1



sample rate: 4



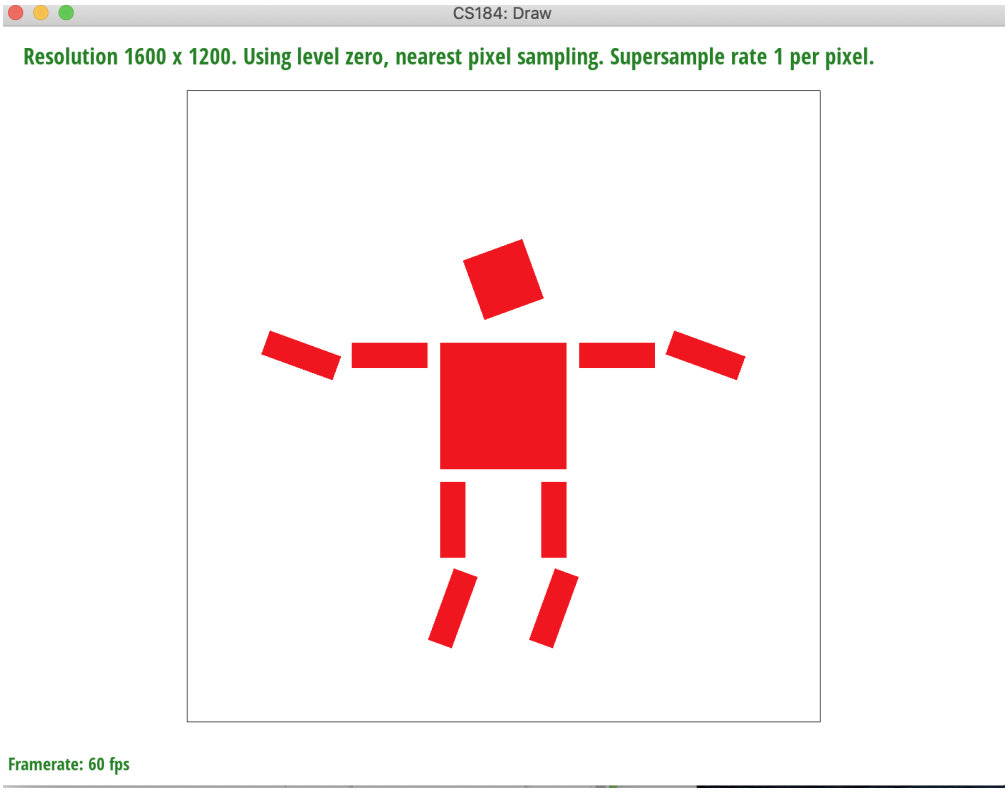
sample rate: 9



sample rate: 16

As we can see, the higher the supersample rate is, the more continuous and blurry (less defined) the sharp angle of our triangle is. This is similar to a smoothing effect, the higher the sample rate, the more smooth it is. The reason why we see this visual difference is because with higher rate of supersampling, we have more subsample pixels with color value of 1, thus when we aggregate them and average them onto the screen buffer, we have way more lighter pixels than in the case of low sample rate, making the edges appear to be smoother.

Task 3: Transform



dancing robot

I am trying to draw a dancing robot, so I slightly rotated its limbs.

Task 4: Barycentric coordinates

What is the barcentric coordinates?

Barycentric coordinate is a coordinate system that is specified by the three vertices in a triangle. This coordinate system is extremely helpful for interpolation across triangle. For example, if you know the color of each of the vertices in a triangle, then at any given point the color of that coordinate can be viewed as a weighted combination of the color of the three vertices, where the weights are the barycentric coordinates. This image is an example of such color combination:

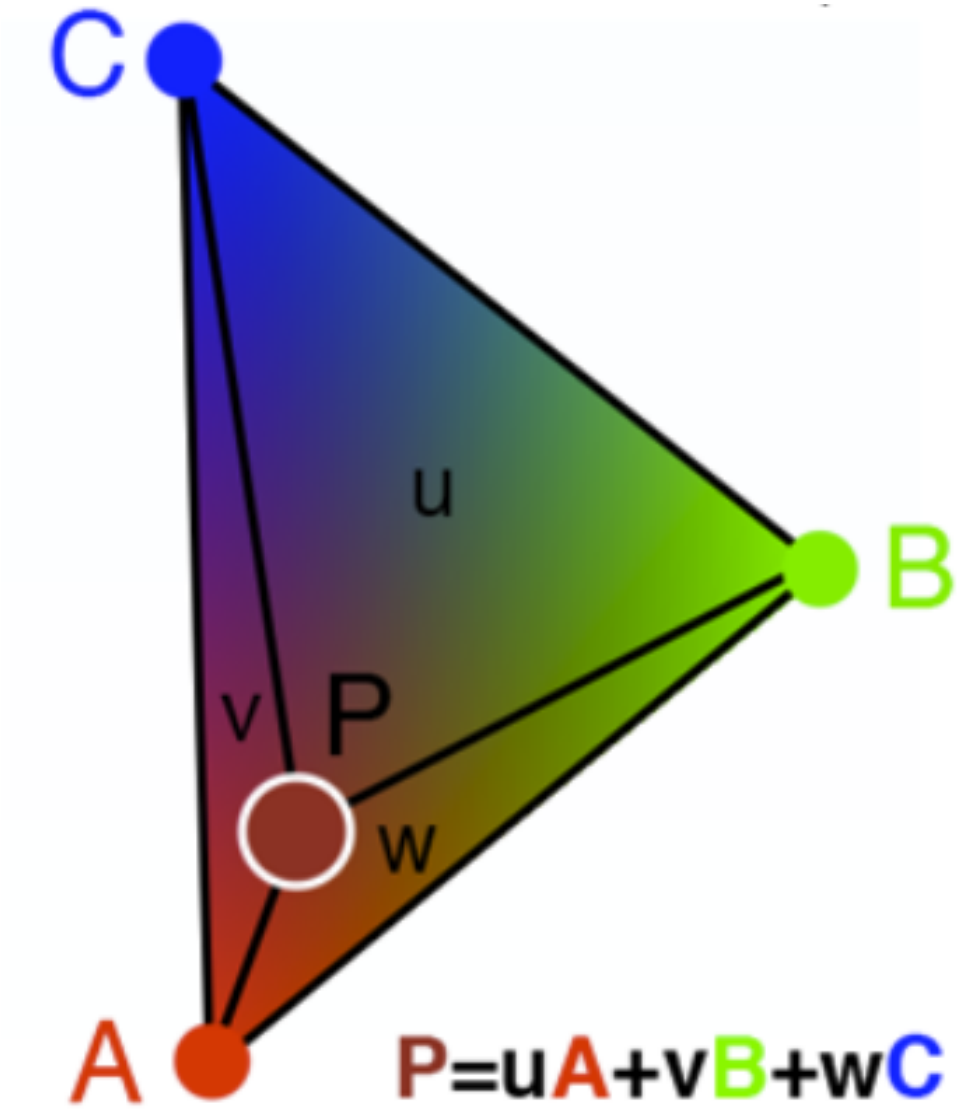
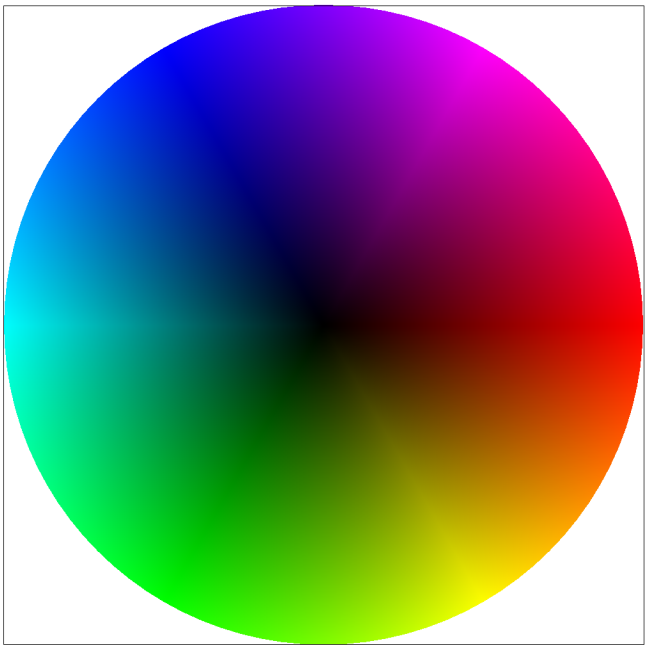


image taken from the internet

test7.svg

Resolution 1600 x 1200. Using level zero, nearest pixel sampling. Supersample rate 1 per pixel.



Framerate: 59 fps

Task 5: Pixel Sampling

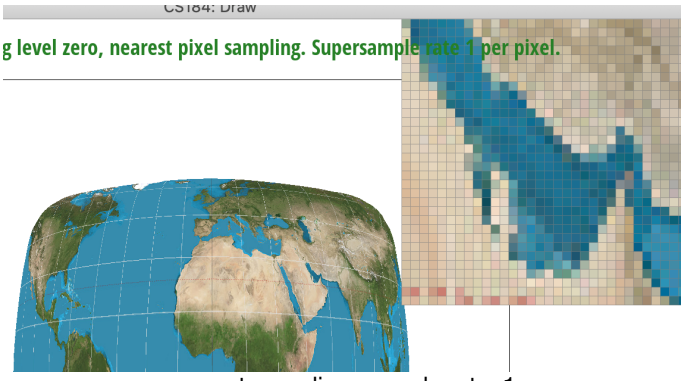
What is the Pixel Sampling?

- Explain pixel sampling in your own words and describe how you implemented it to perform texture mapping. Briefly discuss the two different pixel sampling methods, nearest and bilinear.

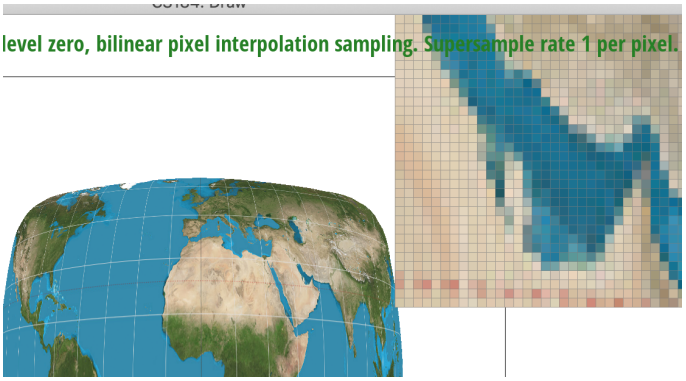
My Responese:

Pixel sampling is the method of estimating the color of a specific location using the color of nearby locations. In this task, I implemented two pixel sampling methods: neareast sampling and bilinear sampling. For neareast sampling, I sample nearby pixel's color and use the barycentric coordinates as weights to determine the color. For bilinear sampling, rather than simply taking the avergae of nearby locations, I weigh the colors of the nearby points with the distances between the point for determination and these points. To simplify the process, I applied linear interpolation on the two pairs of points, and then did another linear interpolation on the result of two pairs. Comparing to the neareastr sampling, bilinear sampling is more computationally expensive, but it provides a smoother transition of color among pixels.

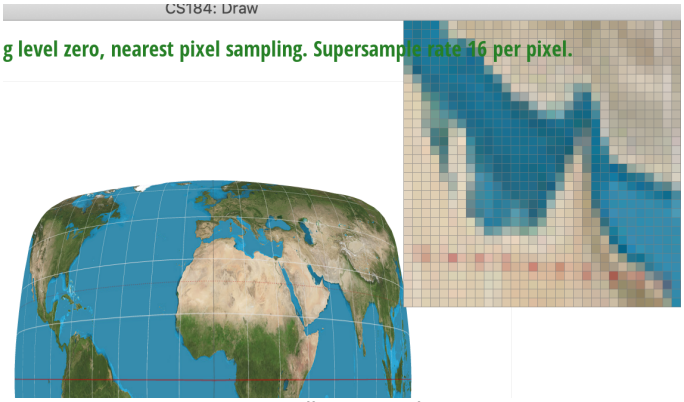
svg/texmap/test2.svg demo and comparison between the two methods



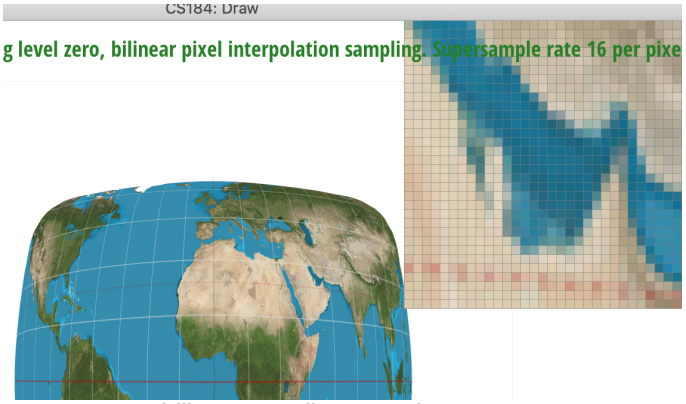
nearest sampling; sample rate: 1



bilinear sampling; sample rate: 1



nearest sampling; sample rate: 16



bilinear sampling; sample rate: 16

comment on the relative difference and why

We can take the Persian Gulf on the world map as an example and analyze the difference. Notice that, in the upper two images when sampling rate is 1, the west coast of the Persian Gulf appears very jaggied for the nearest sampling method, whereas we can see the upper right image (bilinear sampling) has much smoother coastline. However the difference is much smaller when the sampling rate is 16.

It seems like when sampling rate is low and when the texture is full of curves, then we will see a large difference. This is due to nearest sampling not taking into account of distance. It weighs nearby colors equally. So imagine a complicated texture with many curves and rapid change in color, nearest sampling is too coarse to capture the nuances, and its neglecting of distance as weights for color will cause inaccurate color.

Task 6: Level Sampling

What is the Level Sampling?

Level sampling is a useful antialiasing technique when we are dealing with images with depth or any objects in perspective. It is effective at removinng aliasing or any moire effects in the above said images. To implement level sampling, we first calculate the mipmap level at different parts of our image. Then, based on what method we are using we do things differently. For bilinear sampling, we take the two adjacent mipmap levels and then interpolate with the weighted sum. For nearest level sampling, we just round our level to the nearest integer and use whatever texture we get at that level.

Super sampling requires more memory but is also very effective against aliasing. Supersampling can be thought of as producing a higher resolution image and then scaling down, so it requires whatever memory the higher resolution requires. Therefore, it will also be slower in terms of speed. Level sampling is slightly better in terms of memory usage and speed. It is also effect against aliasing and any moire patterns. Point sampling is the weakest among the threee, but requires the least amount of memory.

To summarize, the pattern seems to be that, the more effective antialiasing it is, the more memory the method needs.

demo

L_nearest; P_linear

L_zero; P_linear

L_nearest; P_nearest

L_zero; P_nearest

As we can see, L_zero P_linear appears tro have the best effect. L_nearest P_nearest also looks good