CS 184 HW Write Ups

link to webpage: https://cal-cs184-student.github.io/hw-webpages-sp24-EmmanuelCobian/hw1/index.html

Homework 1 - Rasterizer

In this homework, I implemented a rasterizer that's able to draw triangles, transformations, supersamplings, texture mapping with antialiasing. I think texture mapping in specific was really interesting because I got to understand the importance and utility of Barycentric Coordinates. Being able to interpolate everything from color, position, and texture mappings, these seem to be the basis of what drives the functionality of features like mapping and antialiasing. I also found the process of implementing trilinear filtering interesting because it just boils down to a linear interpolation of two bilinear outputs. This logic could then be extended and applied to other forms of filtering.

Task 1: Drawing Single-Color Triangles

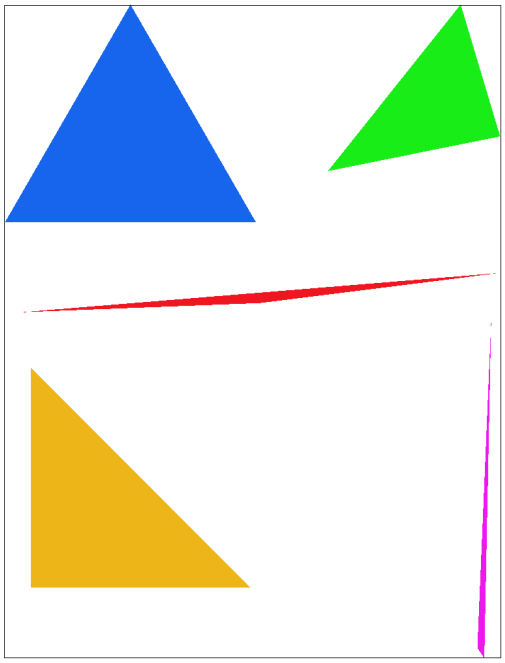
How to rasterize triangles

The way I go about rasterizing triangles is by looping through the grid of pixels around the triangle I wish to draw and then checking to see if the center of a pixel is inside the triangle or not. The way to check if the center of a pixel is inside the triangle or not is by using the formula from lecture and performing the 3 line test for overlapping planes. If the point passes this test, then I fill the pixel in with a specified color.

Algorithm Efficiency

This algorithm to rasterize triangles is no worse than than one that checks each sample within the bounding box of the triangle because when looping through the grid, we take into account the min and max values (x, y) can take and only loop through those. In other words, I only look at the pixels that surround the triangle, not the entire grid.

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



Framerate: 36 fps

Task 2: Antialiasing by Supersampling

Why is supersampling useful?

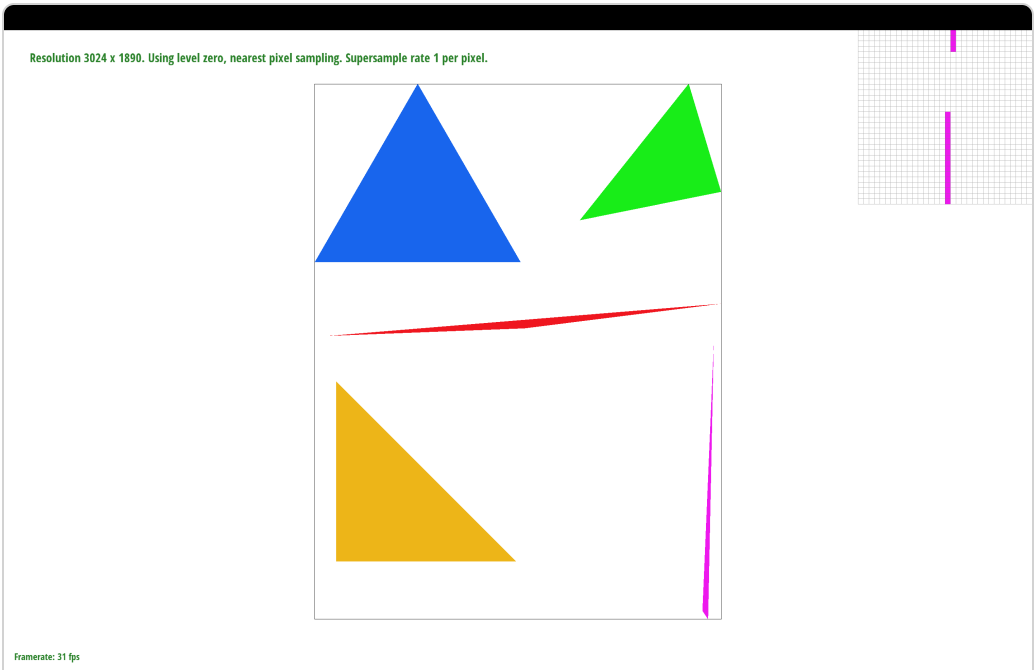
Supersampling is useful because it allows us to take multiple samples of a pixel and then average the color of those samples to get a more accurate color for the pixel. This is useful because it allows us to antialias the edges of the triangle and get a more accurate representation of the color of the triangle.

What modifications were made to the rasterization pipeline?

The modifications made to the rasterization pipeline were to add more loops that iterate through the grid of samples within a singular pixel. Then performing a line test for each of these samples. Then when it was time to resolve the sample buffer to the frame buffer, I take the average color value of each pixel and output that as the final color.

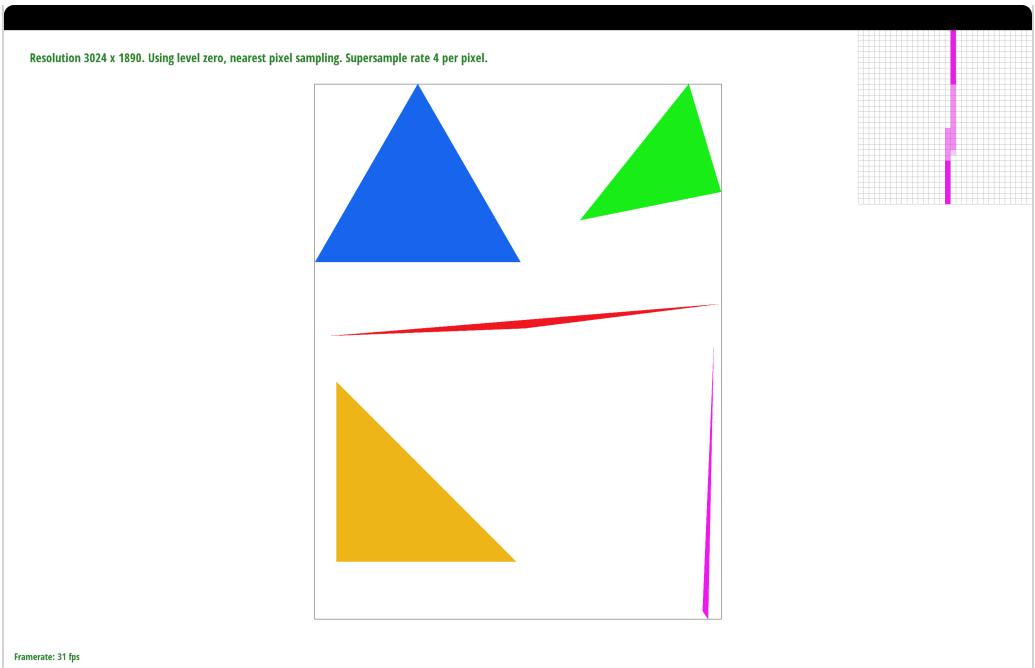
How supersampling was used to antialias triangles

Supersampling was used to antialias triangles by taking multiple samples of the pixels and then averaging the color of those samples to get a more accurate color for the pixel. In other words, the image was rasterized at a higher resolution and then downsampled to the output resolution of the framebuffer.



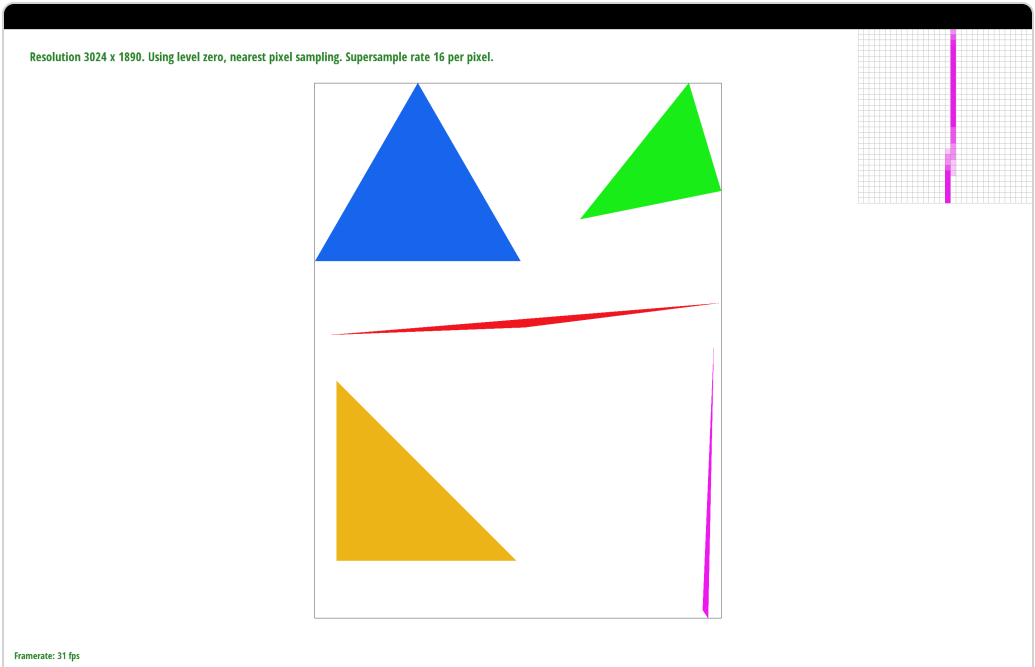
1 Sample

I notice that the tip of this purple triangle isn't even connected. This could be a result of the triangle not falling in the center of the pixel that's sampled.



4 Samples

Now the tip of the triangle connects. We get this result because each pixel is sampled more than once, allowing slimmer parts of the triangle to show up when before, they wouldn't because they didn't lie in the middle of the pixel.



16 Samples

These results are just an improved version of 4 pixel supersampling because now we sample at an even higher

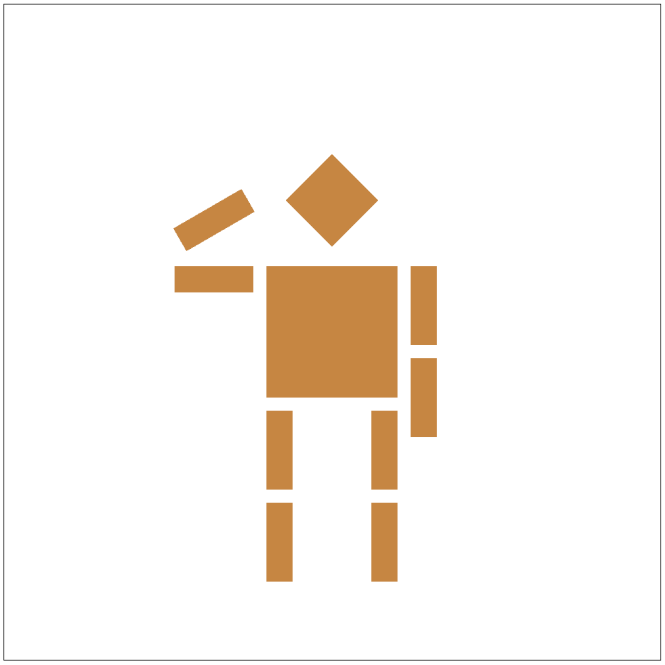
resolution, meaning that more detail of the traingle is able to come through.

Task 3: Transforms

Custom Robot

I was aiming at giving cubeman a human skin tone like color. I also made some transforms to their arms to make it look like cubeman is saluting. As to what or who they're saluting to, that's unclear.

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

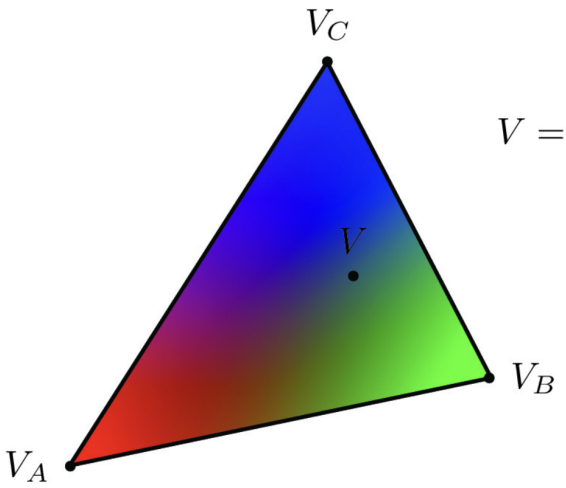


Framerate: 37 fps

Task 4: Barycentric Coordinates

What are Barycentric Coordinates?

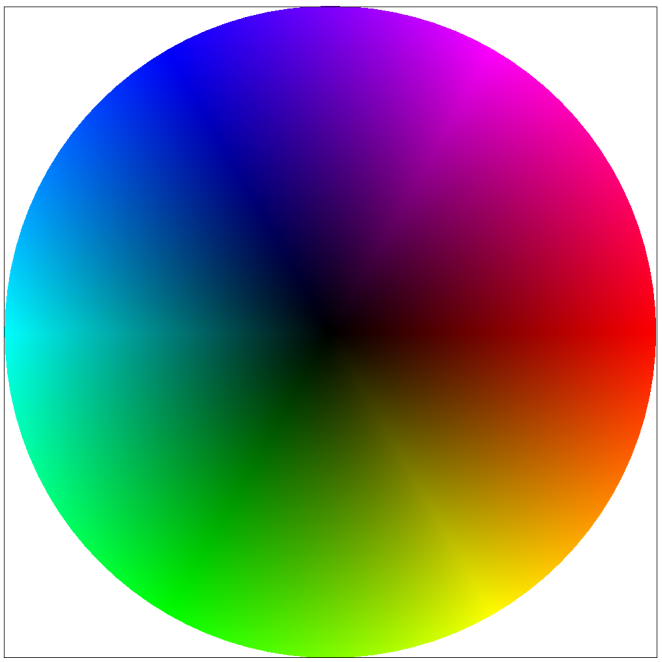
Barycentric coordinates are a set of three numbers that describe the position of a point with respect to a triangle. The constants (α, β, γ) can then be used to interpolate colors, textures, and geometry of any shape. These coordinates can then also be used to check whether a point (x, y) lies inside a triangle or not. For example, if any of α, β, γ aren't in the range [0, 1], then we know that point isn't inside the triangle.



$$V = \alpha V_A + \beta V_B + \gamma V_C$$

V_A, V_B, V_C can be positions, texture coordinates, color, normal vectors, material attributes...

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



Framerate: 36 fps

Task 5: "Pixel Sampling" For Texture Mapping

What is Pixel Sampling?

Pixel sampling is the process of taking a sample of a pixel, translating that pixel position to a texture position, and then sampling that texture image to get the color of the pixel. This is useful because it gives us a standard way of mapping texture coordinates with pixel coordinates. Additionally, different modes of pixel sampling helps reduce aliasing by averaging neighboring texture color positions.

Nearest vs. Bilinear Sampling

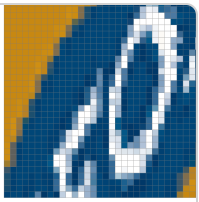
Nearest sampling is the process of taking the color of the nearest texture color to the sample point. On the other hand, bilinear sampling is the process of taking the average of the colors of the 4 nearest texture positions to the sample point. Bilinear sampling is better than nearest sampling because it gives a more accurate color for the pixel.

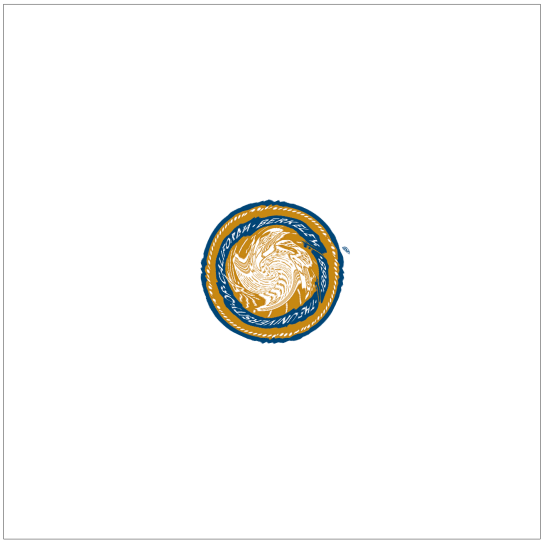
Nearest Neighbor / 1 Pixel Supersampling

The lettering here looks ok, but clearly shows a lot of aliasing. Especially around edges of the letter O, it's borders aren't all that

clear. I notice a lot of blue bleeding into where white pixels should be.

Resolution 2688 x 1666. Using level zero, nearest pixel sampling. Supersample rate 16 per pixel.



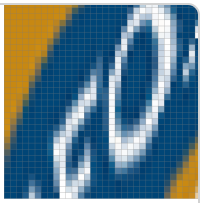


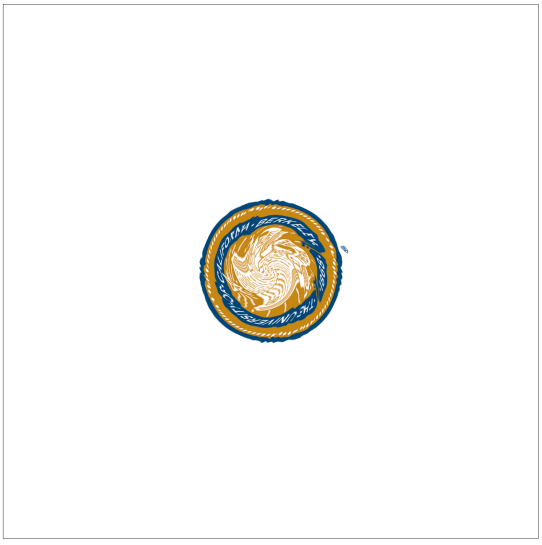
Framerate: 32 fps

Nearest Neighbor / 16 Pixel Supersampling

Because of supersampling, we get better results here. The lettering has more definition and the border between the blue and gold planes has less jaggies. Nevertheless, the lettering still doesn't have clear boundaries.

Resolution 2688 x 1666. Using level zero, bilinear pixel interpolation sampling. Supersample rate 1 per pixel.

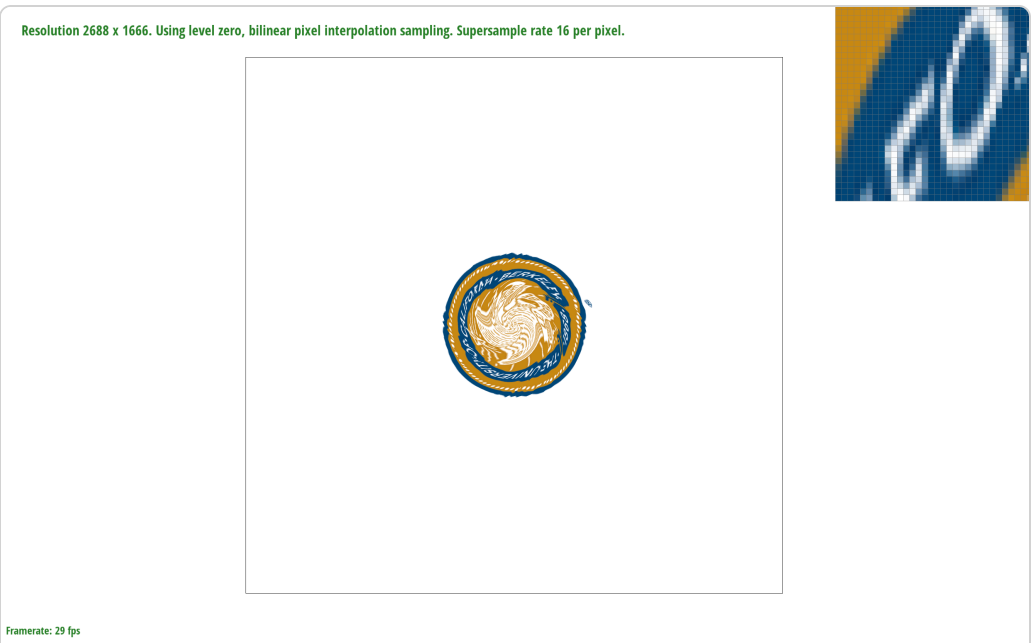




Framerate: 29 fps

Bilinear / 1 Pixel SuperSampling

Even though this samples once per pixel, this already looks magnitudes better than nearest pixel sampling. Elements in the image are well defined with minimal color bleeding.



Bilinear / 16 Pixel Supersampling

This result looks very similar to the same interpolation sampling but fewer pixels for supersampling. I think this is expected because supersampling averages the interpolated averages that were already calculated when doing bilinear interpolation, so the colors aren't going to be all that different.

Task 6: "Level Sampling" With Mipmaps For Texture Mapping

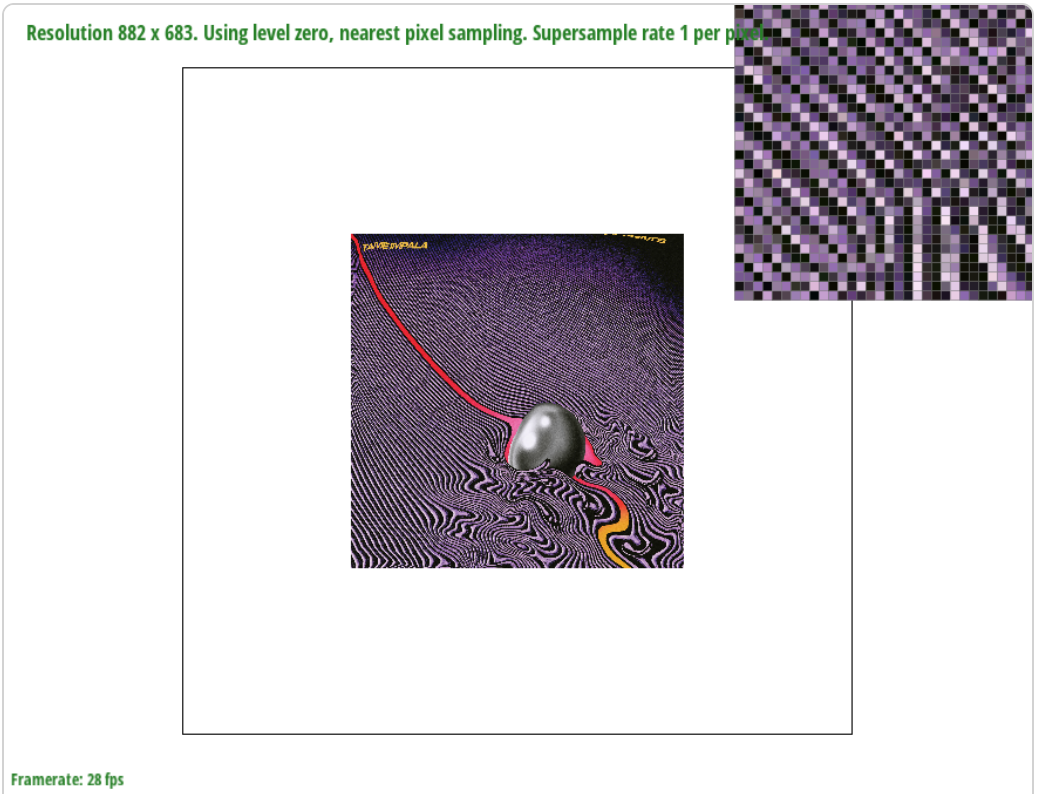
What is level sampling?

Level sampling is the process of taking 1 or more samples of a texture color from a mipmap and then averaging the color(s) of those sample(s) to get a more accurate color for the pixel. Level sampling is useful because it helps create a smooth transition between textures depending on their relative distance to the viewer. It's also good for

optimization because mipmaps are precomputed ahead of time and stored in memory.

How was level sampling used?

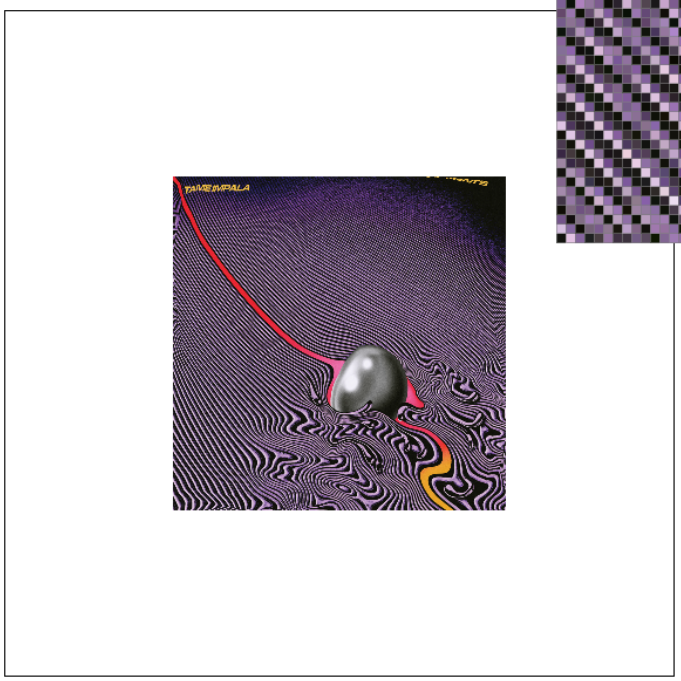
I used level sampling to implement both nearest and bilinear mipmap levels. With nearest, I just take the mipmap level that's closest to the sample given the process of calculating L, from lecture. On the other hand, bilinear mipmap levels interpolate between the two nearest levels with L as the weight dictating how much to take from each color.



L_ZERO / P_NEAREST

With the default values, the image looks pixelated and there's a lot of the moire effect.

Resolution 882 x 683. Using level zero, bilinear pixel interpolation sampling. Supersample rate 1 per pixel.

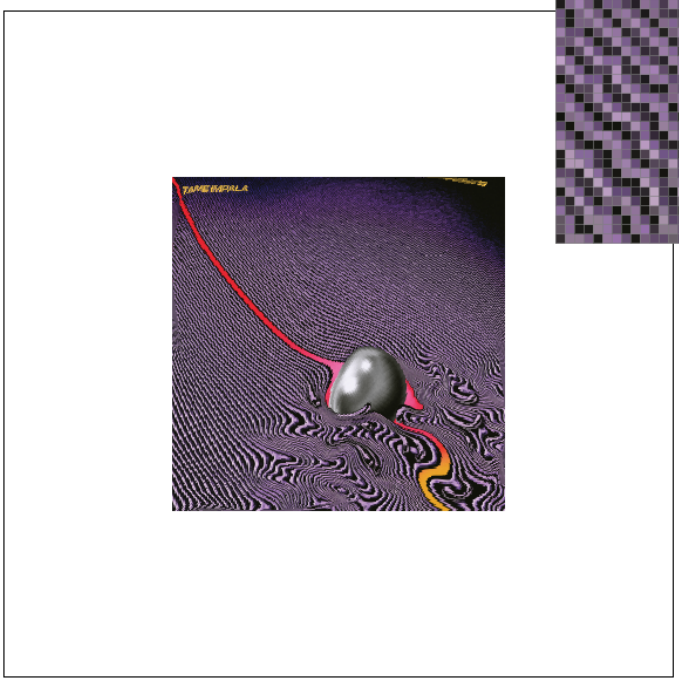


Framerate: 28 fps

L_ZERO / P_LINEAR

With Bilinear interpolation, the image looks a lot better. There's still some of the moire effect, but lines in the distance look slightly better.

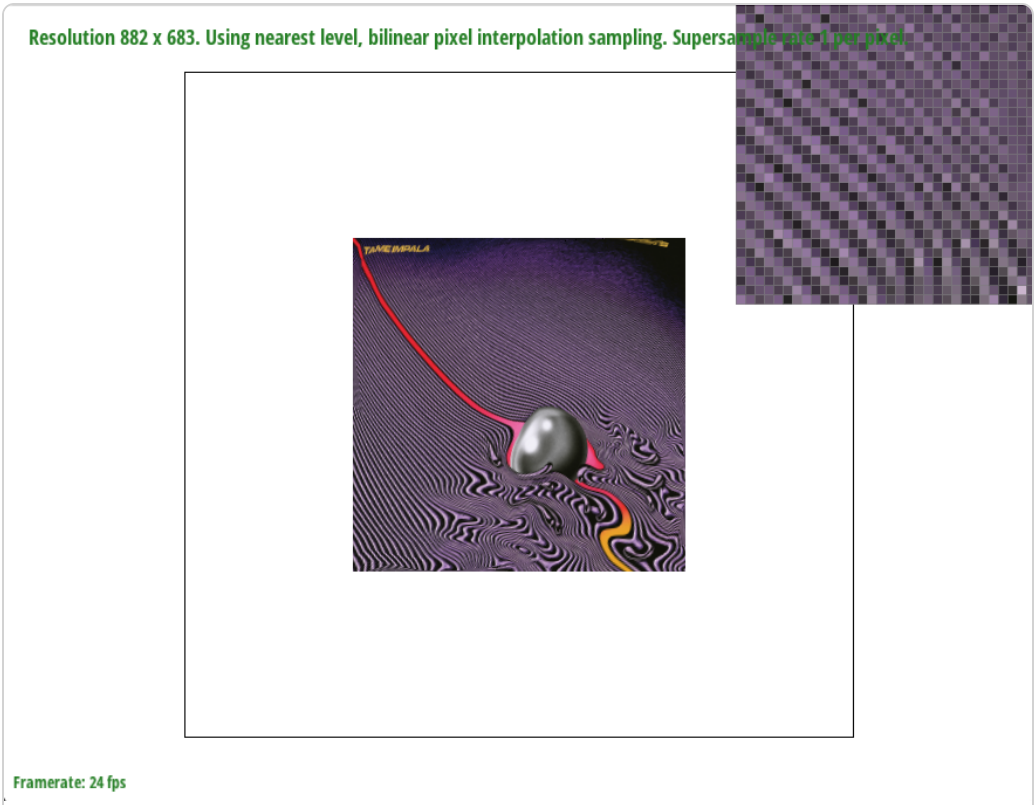
Resolution 882 x 683. Using nearest level, nearest pixel sampling. Supersample rate 1 per pixel.



Framerate: 27 fps

L_NEAREST / P_NEAREST

When taking the nearest mipmap level and the nearest pixel, the image looks even better than previous configurations. Lines in the distnace don't have the moire effect anymore. However, the image looks a bit off in certain areas still.



L_NEAREST / P_LINEAR

With the nearest mipmap level and bilinear interpolation, I believe the image looks the best. Lines both in the distance and up close straighten out and waves in the forefront look more well defined.

Emmanuel Cobian Duarte - CS 184 - Spring 2024