COL781 Assignment 1

Salil Gokhale
2021MT10237

Aniruddha Deb
2020CS10869

6 Feb, 2024

1 Parallel Rasterization Algorithm

Our rasterization algorithm iterates over all the pixels in the bounding box of a given triangle and computes the barycentric coordinates of the pixel. If all the coordinates are positive-valued, then the point lies inside the triangle. If it does, it does a z-test and if this passes, runs the fragment shader with these barycentric coordinates (post perspective-correct interpolation). Supersampling is implemented by setting the framebuffer size to spp $\times$ (width|height), and averaging the values when blitting the framebuffer to the display

This algorithm is trivially parallelizable, so we implement a workqueue, which spawns K(=4) threads. Each thread is given a sequence of pixel chunks (16x16 squares) to check. The threads run the fragment shader independently and write to the sections of the framebuffer they have been assigned to.

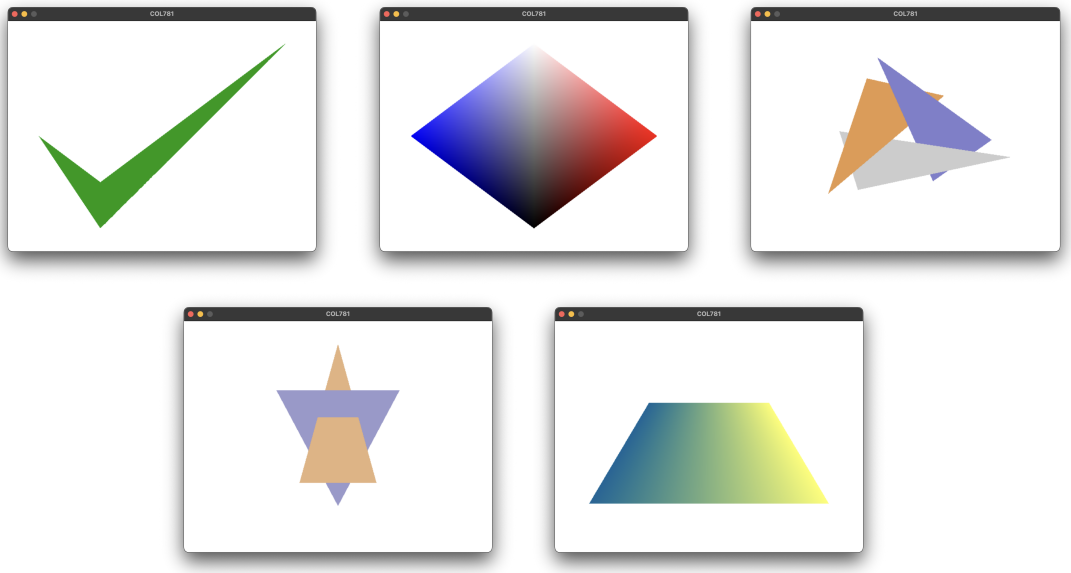

Figure 1: Example programs rendered using the software rasterizer

Note the Z-buffer is correctly implemented as shown in example 3 and 4, and there is no distortion of the gradient in example 5. The parallel rasterizer gives us 60fps on average, compared to 20fps using a single-threaded bounding box rasterizer. All programs were compiled with the '-O3' option enabled

2 Clock

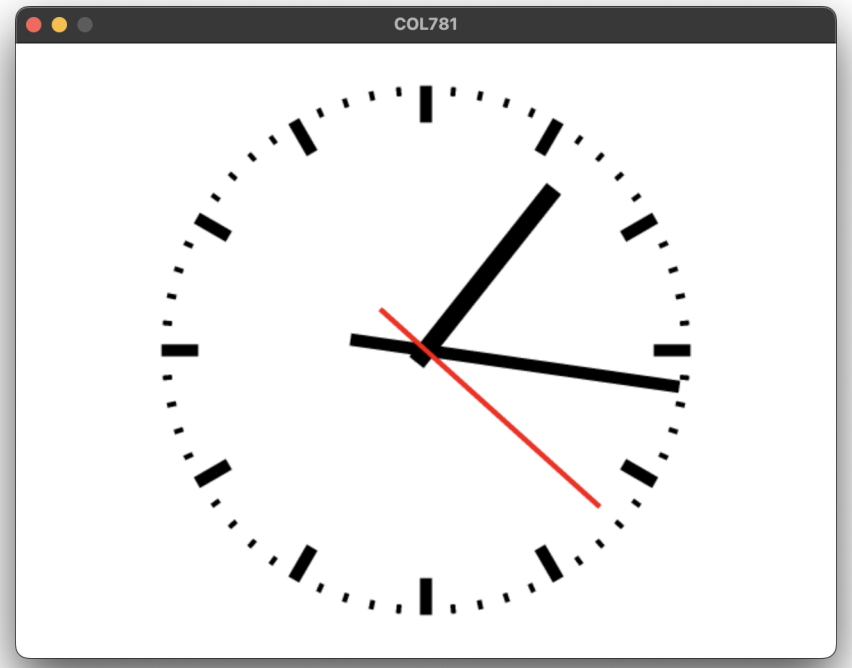

Figure 2: Clock program

The only mesh in this program is a unit square made up of two 4 vertices and 2 triangles. All the rectangles in the scene have been drawn by applying affine transformations to this mesh.

3 Interesting Scene

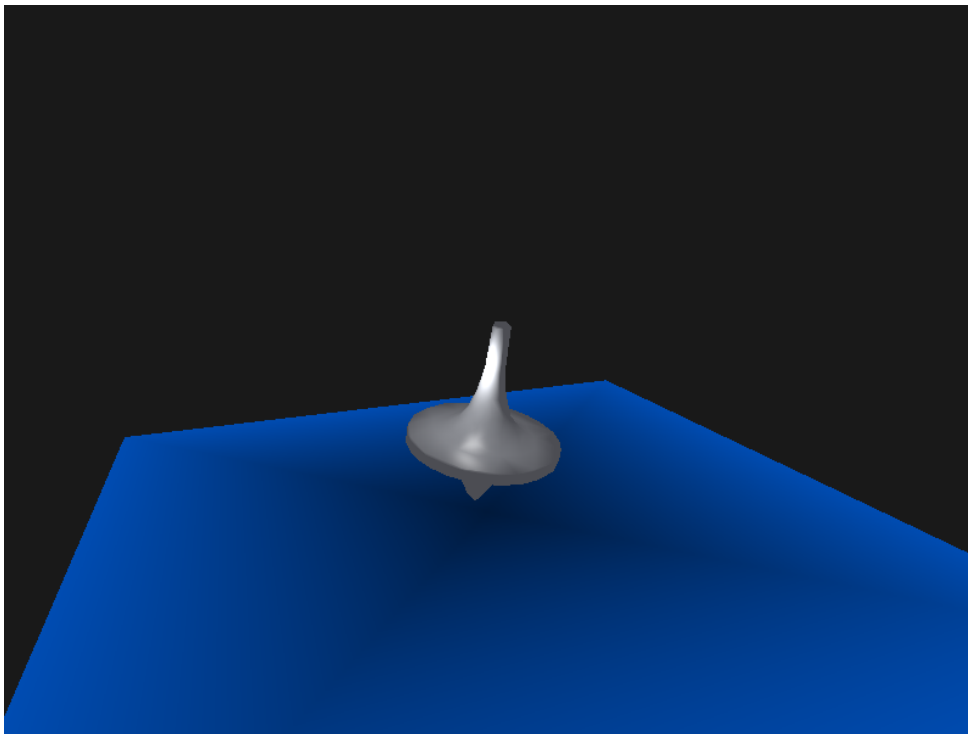

Figure 3: Top program

We designed a small animation of a spinning top bouncing on a blue plane. The spinning top .obj file was taken from [1] and reduced to a 500-face mesh in MeshLab. We implemented Blinn-Phong reflection model shaders from [2]. The parallel rasterizer gives us 19fps on this scene.

We were thus able to implement **Every component of the assignment**.

4 Teapot

Figure 4: Teapot program

To test our pipeline, we rendered the Utah Teapot model. Obj file was taken from [3]. Blinn-Phong shaders were used for this render too.

References

[1] URL: https://sketchfab.com/3d-models/day31-spinning-top-d090504b2d994d7c812c14bc96

[2] URL: https://en.wikipedia.org/wiki/Blinn%E2%80%93Phong_reflection_model.

[3] URL: https://graphics.stanford.edu/courses/cs148-10-summer/as3/code/as3/teapot.obj.