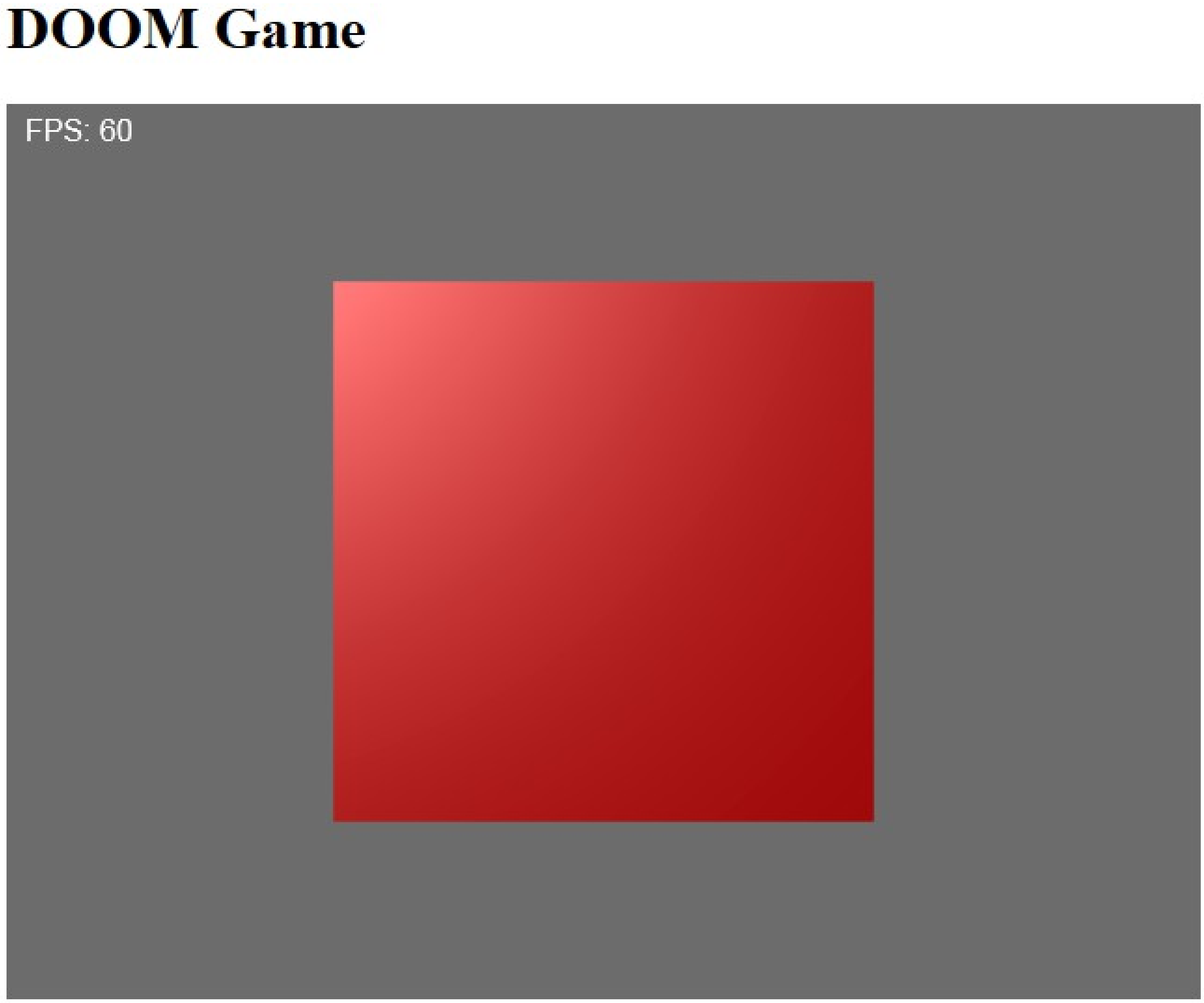
Matthew Gillatt
CS464
11/30/2015

Final Project Status Update

Up until now, the BSP Tree portion of the code is still being researched. All of the implemented code so far is strictly implementing new Javascript and WebGL functionality into the project. Some of the necessary features that haven't been part of previous projects include drawing text to a 2D canvas, and disabling automatic depth testing along with drawing individual polygons in a manual order. Here is a more thorough list of features implemented so far:

- 2D canvas layer to draw debug text, such as FPS, keyboard input, and positioning.
- Drawing polygons in a specified order, ignoring automatic depth testing, and drawing only some of the polygons out of the whole geometry.
- Vertex color attribute.
- Phong shading and lighting from previous assignments.
- Keyboard movement modified from previous assignment. Includes sideways strafing.

Matthew Gillatt
CS464
11/30/2015



DOOM Game

Illustration 1: 2D text canvas and FPS counter

DOOM Game

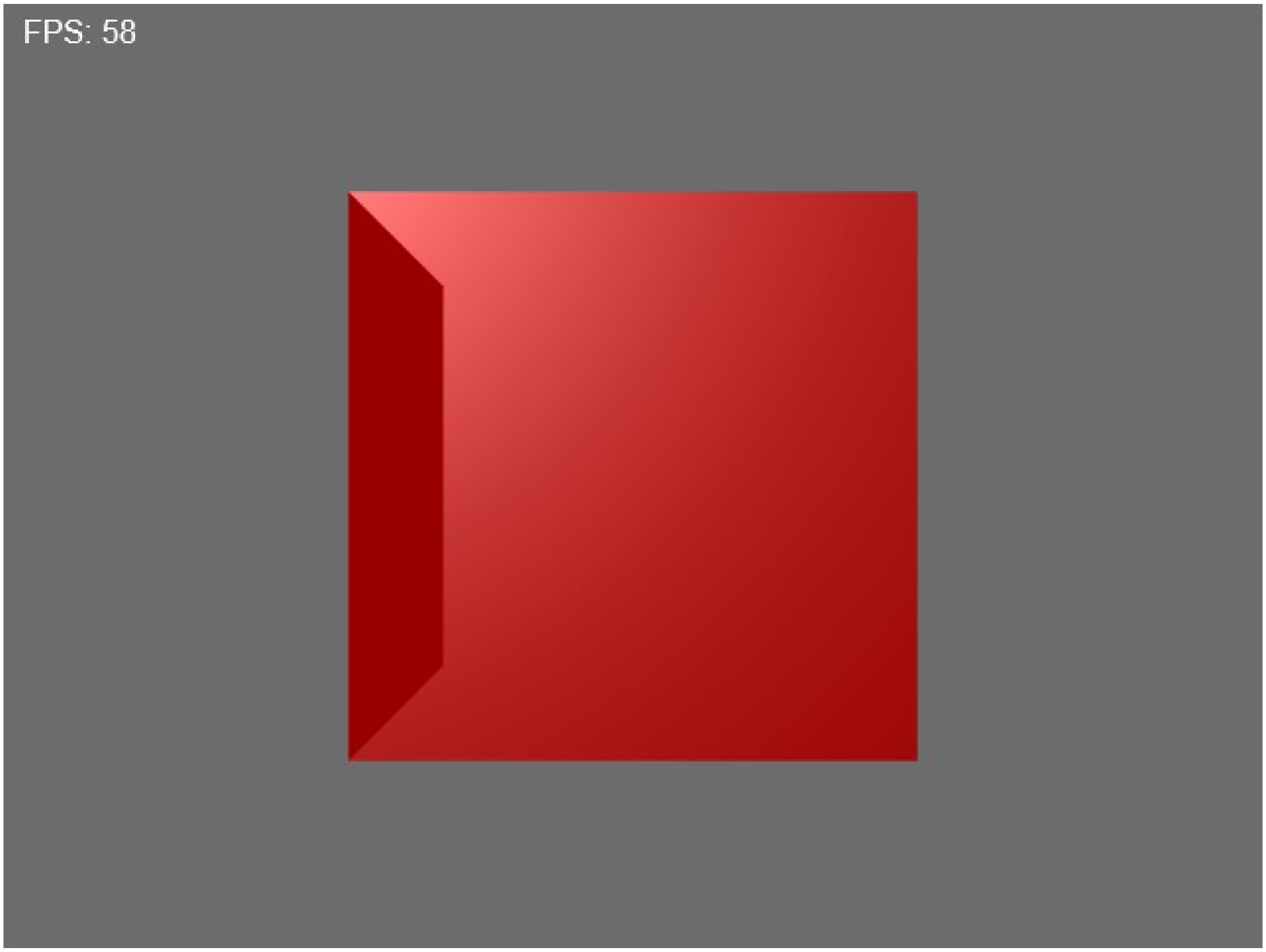
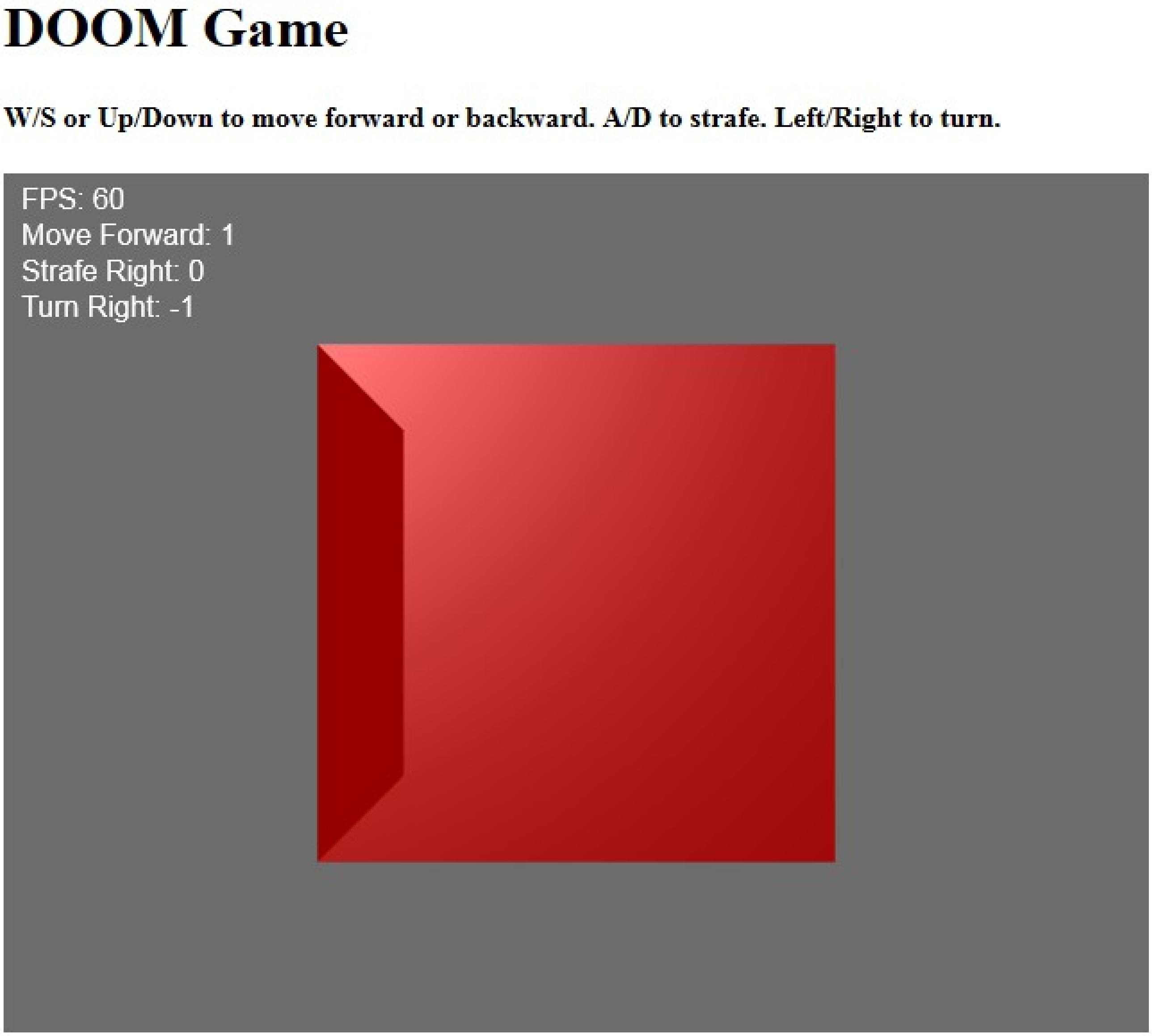


Illustration 2: Selective drawing order: only drew front face, then left

Matthew Gillatt
CS464
11/30/2015



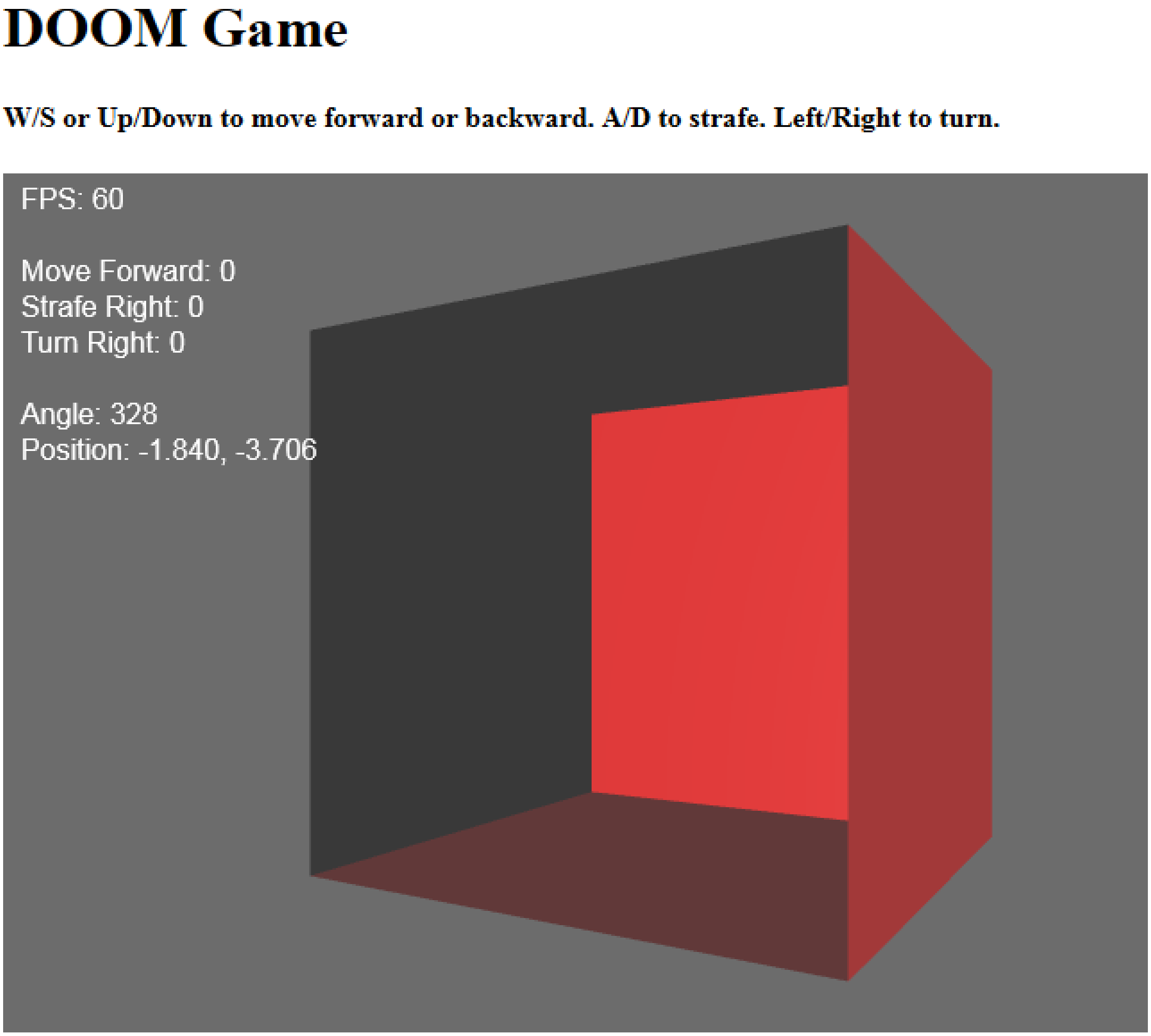
DOOM Game

W/S or Up/Down to move forward or backward. A/D to strafe. Left/Right to turn.

```
FPS: 60
Move Forward: 1
Strafe Right: 0
Turn Right: -1
```

Illustration 3: Keyboard handling, using WASD and Cursor keys

Matthew Gillatt
CS464
11/30/2015



DOOM Game

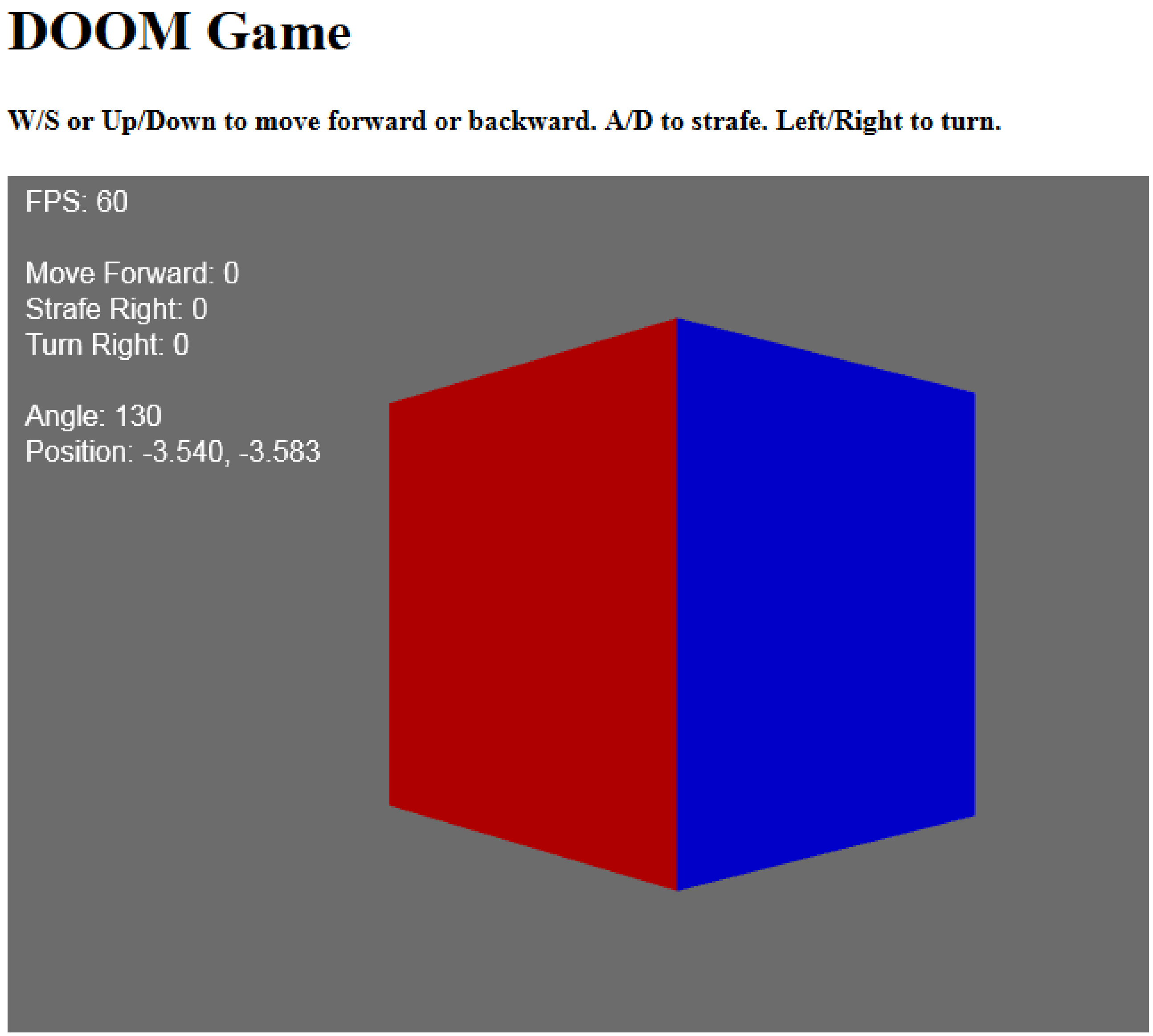
W/S or Up/Down to move forward or backward. A/D to strafe. Left/Right to turn.

```
FPS: 60

Move Forward: 0
Strafe Right: 0
Turn Right: 0

Angle: 328
Position: -1.840, -3.706
```

Illustration 4: Complete movement controls, and depth testing disabled



DOOM Game

W/S or Up/Down to move forward or backward. A/D to strafe. Left/Right to turn.

```
FPS: 60

Move Forward: 0
Strafe Right: 0
Turn Right: 0

Angle: 130
Position: -3.540, -3.583
```

Illustration 5: Vertex Color attribute. Now white light on colored polygons, instead of red light on white cube

There's still plenty to go even to implement a BSP Tree with Painter's algorithm. 1St, I'll move the geometry to another file, and expand it to a game level instead of a cube. I'll make a segment object class and make my map out of them, so my main program can generate polygons using the 2D segments. Then, I can get to splitting the map and making a BSP Tree. Finally, I can traverse the tree from back to front to draw the segments in the right order. Hidden surface removal may be beyond the scope of the project at this point. Since it is the focus of the project, I'll focus on hidden surface removal before anything that makes it into a playable game, such as preventing the player from moving through walls, or 2D objects in the map. The worst case scenario is just implementing the BSP Tree with Painter's Algorithm traversal, proving that I don't need automatic depth sorting.