CS411 Assignment 2

Testing/Dev environment:

- Chromium 77 on Arch Linux 2019.9.20
- Google Chrome 77 on MacOS Mojave

Thought Process for Development:

- I initially skimmed through all the files to get comfortable with what was already written
- The libraries aren't how I normally write javascript, but they work so I'll work with them
- It took a few minutes to realize that the triangle is static and our transformations have no lasting effect
- I'm not sure why switching the render mode breaks my triangle but I'm fairly confident this isn't my doing
    - I feel like fixing it would involve interacting more closely with/modifying the provided libraries
        - which might be prohibited
- I'd be interested in doing future assignments in C/C++ if there was support code written for it
    - C++ is my strongest language and I feel like it's more powerful in this case
-

Question 2 From cs411-hw2.pdf

The javascript snippets were run in the browser console of my `cs411-assignment2-template.html`

- A. <1,1>T(<2,3>) = <3,4>
    - added vectors
- B. <1,1>S(<2,2>) = <2,2>
    - multiplied by scaling vector
- C. <1,1>R(45°) = <1.4142135381698608, 0>
    - produced by running this in console of AS2

```
function rotatePoint(angle, point) {
    const c = Math.cos(angle * Math.PI/180);
    const s = Math.sin(angle * Math.PI/180);
    let a = new Matrix4({ elements:
        new Float32Array([
            c,-s, 0, 0,
            s, c, 0, 0,
            0, 0, 0, 0,
            0, 0, 0, 0
        ]),
    });
    return a.multiplyVector3({elements: point.elements || point});
}
console.log(rotatePoint(45, [1,1,1]).str());
```

- D. <1, 1, 1>
- E. <1/2, 1/2, 1>
- F. <2, 4, 6>
- G. <1/3, 2/3, 1>
- H. It is a point at infinity because the z coordinate is zero, and represents a direction/line
- I. [6.062177658081055, 2.598076105117798, 1]

```
console.log(rotatePoint(30, [2, 5, 1]).str())
```

- J. [3.366025447845459, 4.098075866699219]

```
Vector3.prototype.add = function(other) {
    for (let i = 0; i < this.elements.length; i++)
        this.elements[i] += (other.elements || other)[i];
    return this;
};
console.log(rotatePoint(30, (new Vector3([2,5,1]))
    .add({ elements: [-1, -2, -1]})
).add({elements: [1,2,1]}).str());
```

- K. [9.899495124816895, 2.8284270763397217]

```
let p = new Vector3([2,5,1]);
p.add({elements: [3, 4, 1]});
console.log(rotatePoint(45, p).str());
```

- L. [7.949747562408447, 6.1213202476501465]

```
p = new Vector3([2,5,1]);
console.log(rotatePoint(45, p).add({elements: [3, 4, 1]}).str());
```

- M. [0, 5.656854152679443, 0]

```
rotatePoint(-45, (new Vector3([5,6,1])).add([-1,-2, 0]))
```

- N. TR
- O. translate(<2,2>) then scale(<1,2>)
- P. 1010
- Q.

- A line is completely inside if both of it's endpoints have a region code of 0000
- A line is completely outside if both it's points have a one at the same bit location