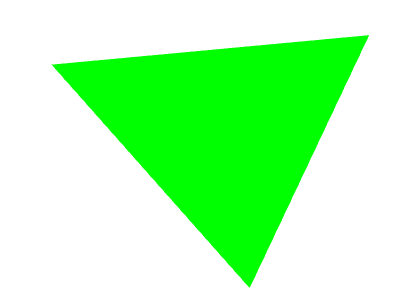
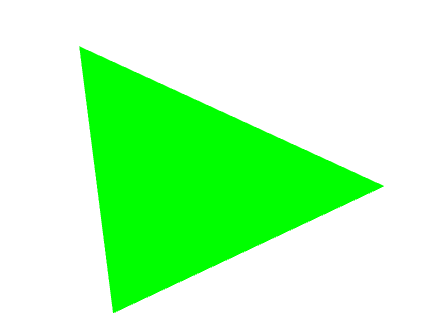
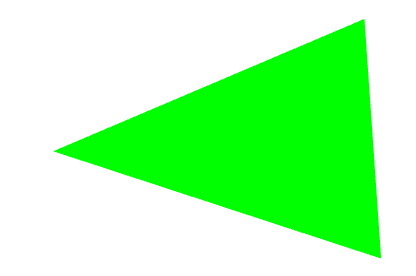
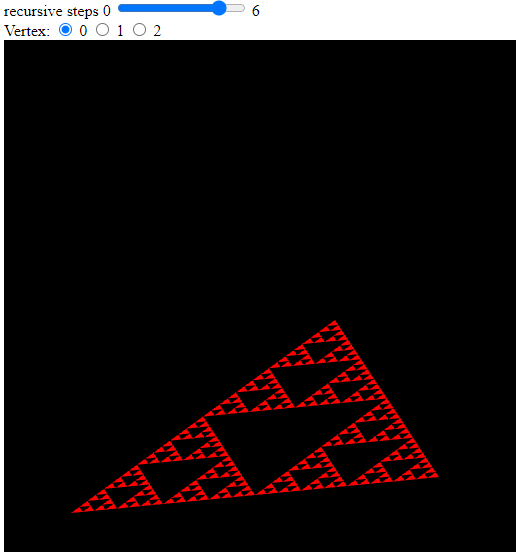
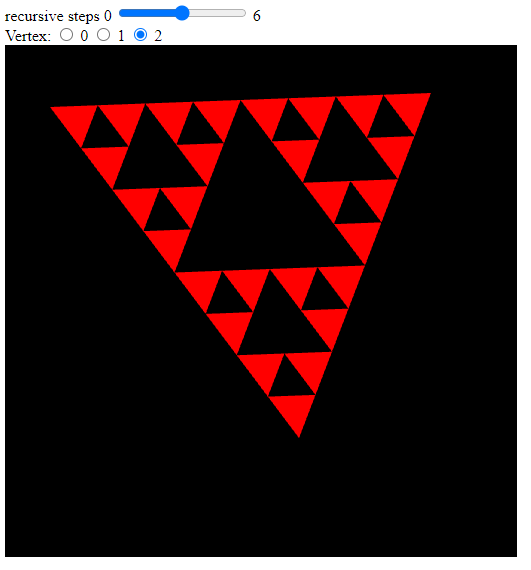
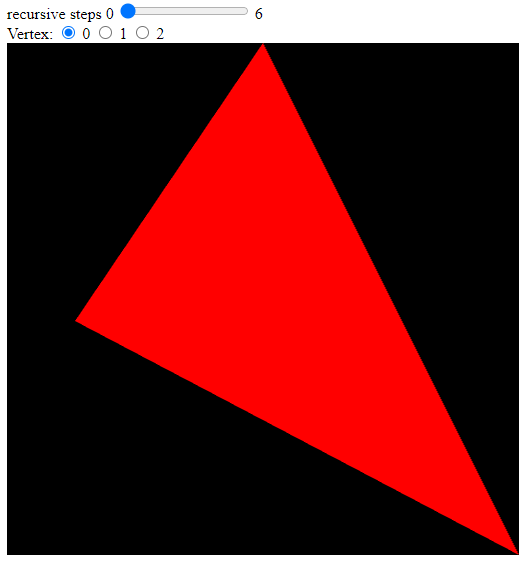
1. Screenshots.





1. What is the purpose of gl\_Position.w = This vertex is called homogeneous vertex coordinate.

Modern [OpenGL](https://en.wikipedia.org/wiki/OpenGL) and [Direct3D](https://en.wikipedia.org/wiki/Microsoft_Direct3D) [graphics cards](https://en.wikipedia.org/wiki/Graphics_card) take advantage of homogeneous coordinates to implement a [vertex shader](https://en.wikipedia.org/wiki/Vertex_shader) efficiently using [vector processors](https://en.wikipedia.org/wiki/Vector_processor) with 4-element registers.

<https://en.wikipedia.org/wiki/Homogeneous_coordinates#:~:text=Homogeneous%20coordinates%20are%20ubiquitous%20in,which%20the%20vector%20is%20multiplied>. ------------- Basically the “w” coordinate can be used to translate, rotate, and scale the vector/point.

<https://stackoverflow.com/questions/2422750/in-opengl-vertex-shaders-what-is-w-and-why-do-i-divide-by-it> -------------- MATRIX MATH UGH WHERE’S MY GRAPHING CALCULATOR

1. The triangle FLEW off to the top right of the frame. We fixed it almost immediately by adding “bounds” to the frame. More specifically, if we went over 0.9 in the x or y coordinate, then reverse direction. Note that there aren’t else statements, because the triangle COULD hit a corner, which would reverse both directions at the same time. However, it is possible (and maybe more efficient depending on the application) to sacrifice the responsiveness of one direction’s bounds to avoid extra computation by using “if else” statements (would skip all the “if” statements past the one satisfied).

function render() {

    setTimeout(() => {

      x += 0.05 \* xDir; //0.001 makes it like the DVD bouncy screen

      y += 0.1 \* yDir; //0.0025 makes it like the DVD bouncy screen

      if (y > 0.9) {

        // top hit -- reverse y but keep x

        y = 0.9;

        yDir \*= -1.0;

      }

      if (x > 0.9) {

        // right hit -- reverse x but keep y

        x = 0.9;

        xDir \*= -1.0;

      }

      if (y < -0.9) {

        // bottom hit -- reverse y but keep x

        y = -0.9;

        yDir \*= -1.0;

      }

      if (x < -0.9) {

        // left hit -- reverse x but keep y

        x = -0.9;

        xDir \*= -1.0;

      }

      gl.uniform1f(xLoc, x);

      gl.uniform1f(yLoc, y);

      gl.clear(gl.COLOR\_BUFFER\_BIT);

      gl.drawArrays(gl.TRIANGLES, 0, 3);

      window.requestAnimFrame(render);

    }, 100);

  }

};

1. This is only showing us the physical pixel coordinate on the entire display. This doesn’t serve our purpose, as we’re focused only on the frame within the browser. As such, we need to take this number and modify it into a decimal fraction of our canvas size.
2. As described in Q5, this is the part of the code that will change the actual pixel location on the display (X, Y) into a decimal fraction of where the point exists on our canvas.