CIS367 - GFX

Winter 2024

**HW1 : WebGL // Sierpinski Gasket**

**Due 01/30/2024 @ 11:59pm**

**Synopsis:**

While we've played around with WebGL a bit in class, this lab will have you (inter)actively creating your first program. To do this, we'll use the book's example of the Sierpinski Gasket program as a playground to get a feel for what is possible with WebGL.

*Note: this is your first lab with me. Keep an eye out for things marked in* ***red*** *as they tend to be required screenshots or actions for you to fill in. Your lab writeup is at the end of the document.*

**0 – Grab a copy of the Sierpinski Gasket:**

<https://efredericks.github.io/gvsu-cis367/demos/>

(Alternatively, you can grab it from the book's GitHub repository in Chapter02: <https://github.com/esangel/WebGL>)

**1 - Open up the first Sierpinski Gasket example**

First, create a copy of gasket1.html and gasket1.js (Sierpinski Gasket 1). Add a hyphen and your last name to the filename. For instance, you should now have gasket1-fredericks.html (and a similar .js file) if you had my last name. **Make sure that you change the reference to gasket1.js to be gasket1-*lastname*.js in the HTML file! Otherwise, your updates below won't make any changes.**

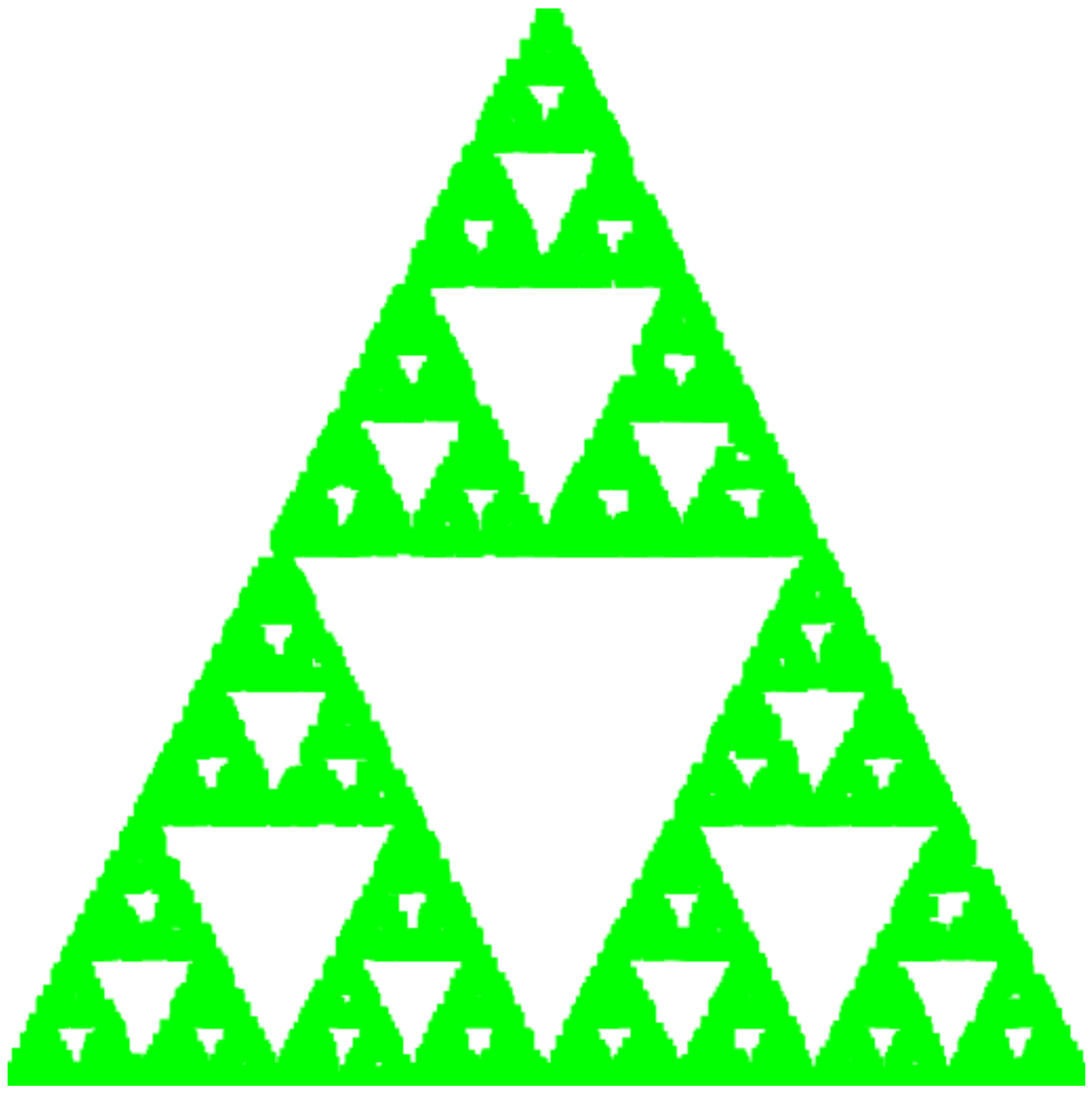
Open up gasket1-*lastname*.html in your browser (without much tweaking, Chrome seems to be best at this, though I'm sure Firefox is fine too these days). You should see something that looks like this:

Shape

Description automatically generated

**Your first task is to change this to this:**

Shape

Description automatically generated ⇨ 

**There should only be two lines of code you have to modify. Copy and paste your changed code into Q1.**

*If you are stuck, the only changes that need to be made are in the HTML file.*

gl\_PointSize = 10.0;

gl\_FragColor = vec4( 0.0, 1.0, 0.0, 1.0 );

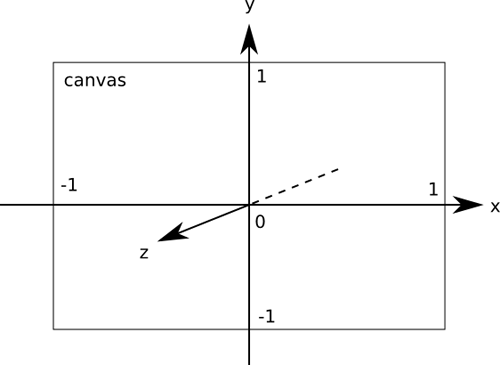
Great, you've done the thing! Onto the next thing!

###

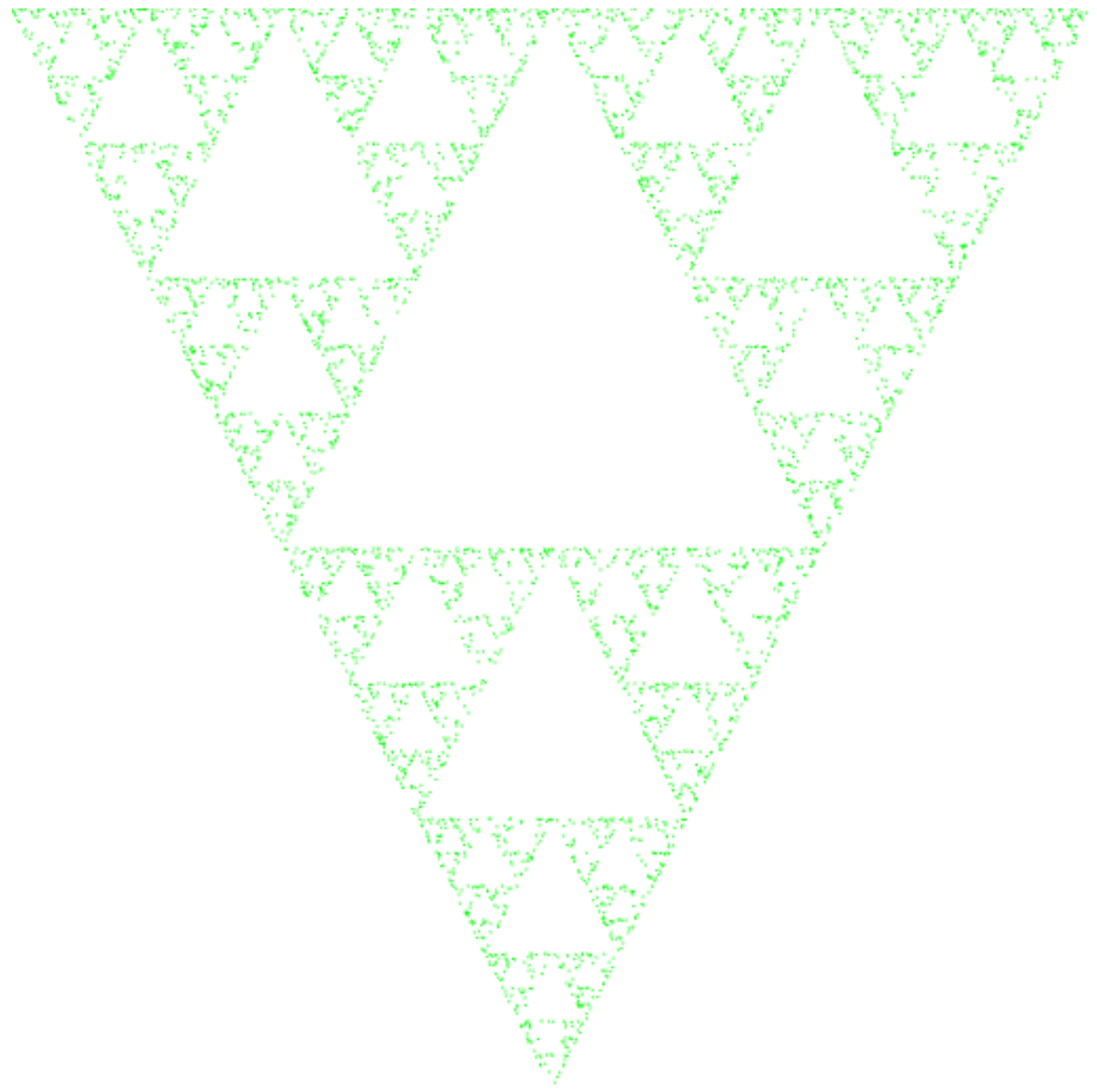
Assuming you understand what changed, you have modified the **vertex** shader and the **fragment** shader. Change the value in the **vertex shader** back to what it was previously (let's go back to smaller points). Now let's rotate this thing manually.

**2 - Manual Rotation**

What is rotation but specifying a series of vertices in a different fashion? Remember that your drawing area has axes that look like this:



Now, do this with your gasket **by modifying the vertices you specified**:



**Copy the code you changed into Q2.**

gl\_Position = vec4(vPosition.x, -vPosition.y, vPosition.zw);

and now let's perturb some things. And, we'll introduce some randomness.

**3 - Perturbation**

In the JavaScript file, add the following function above the call to window.onload:

function getRandomArbitrary(min, max) {

return Math.random() \* (max - min) + min;

}

℅: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random>

This function returns a random floating point number in between a specified minimum and maximum value. In contrast, the Math.random() function returns a random float on the range of [0.0, 1.0]. Let's muck around with the bisector, shall we?

Look for the call to scale(0.5, p). Replace 0.5 with Math.random(). **Take a screenshot and paste it into Q4**.

A green dot pattern on a white background

Description automatically generated

Now, replace Math.random with getRandomArbitrary, and replace min and max with two floating point values of your choosing. Repeat this two more times with different min/max values. **Paste those into Q4 as well** (you should have a total of 4 screenshots for Q4!).

.25, .75

**A green dots on a white background

Description automatically generated**

**0, .5**

**A green particles in the shape of a butterfly

Description automatically generated**

**.5, 1**

**A white surface with green dots

Description automatically generated**

**Report / Homework (100 pts)**

1. Copy and paste the code changes you made to change the color of the first triangle. What *values* have you changed?

gl\_PointSize = 10.0;

gl\_FragColor = vec4( 0.0, 1.0, 0.0, 1.0 );

1. Triangle rotation
   1. Copy/paste the code you made to rotate the triangle.

gl\_Position = vec4(vPosition.x, -vPosition.y, vPosition.zw);

* 1. Can you think of any other way to rotate the triangle? Describe one possibility.

In the for-loop that generates the points, instead of switch ‘add’ to subtract

1. Why does our code need to be placed into a window.onload function?

Since our script is in <head> block and is not deferred, the script will be loaded and execute before the canvas exists. Since our script needs the canvas, we need to wait until the window finishes loading so the canvas exists.

1. Paste your screenshots here on perturbing things.

A green dot pattern on a white background

Description automatically generated**A green dots on a white background

Description automatically generatedA green particles in the shape of a butterfly

Description automatically generatedA white surface with green dots

Description automatically generated**

1. In the JavaScript file, there are two uses of the scale function. For the second (scale(0.5, p);) modify the value sent in. What do you think is happening here (i.e., what is the purpose of scaling)?

This is what ensures the fractal doesn’t diverge from outside the triangle. Points are generating by choosing some point in the fractal, picking a random corner, adding the position of the corner to the chosen point, and dividing by 2. If you don’t divide by 2, you would just end up with a 2D lattice of points made by moving along the vectors defined by the corners of the triangle.

1. We have the following three lines of code in all our gasket files, and they will show up in one form or another in pretty much everything we do. What is the purpose of *each line* of this code block specifically (i.e., get comfortable with reading documentation)?

var vPosition = gl.getAttribLocation( program, "vPosition" );

gl.vertexAttribPointer( vPosition, 2, gl.FLOAT, false, 0, 0 );

gl.enableVertexAttribArray( vPosition );

getAttribLocation: looks in passed program for vertex attrib named “position” and returns index of that vertex attribute.

vertexAttribPointer: specifies to OpenGL exactly how to interpret the data currently bound in GL\_ARRAY\_BUFFER. Our arguments indicate that our buffer has pairs of floats that shouldn’t be normalized with no space between them in memory that start at index 0.

enableVertexAttribArray: enables the vPosition attribute to be used in our shader.

1. Create your portfolio website.

**REMINDER NOTICE**

There is something funny going on with caching on EOS. Ensure that whatever you upload is “ready to go” as changing scripts server-side doesn’t seem to update.

We are not expecting you to be a web designer extraordinaire! This site should be something that you can point to later on for projects. For this question, you will be setting up the website that you'll use for the remainder of the semester to demonstrate your work.

You have the option to either edit on EOS directly or edit locally and upload your files. *I will leave it to you to handle this step as you should have had this experience in 241. If you have no idea how to do this please let me know.*

*Tips*:

If are editing locally, you can use SCP (either a standalone application or a VSCode plugin) to upload your files to the EOS space. You can also use sFTP if you're comfortable with the command line. Or email it to yourself. Regardless, your files need to be on EOS to be graded.

*Alternatively, you can use GitHub pages as well -* [*https://docs.github.com/en/pages*](https://docs.github.com/en/pages)

If you are editing directly on EOS, you can use the RDP connection to use VSCode up there. Otherwise, if you're comfortable with the command line you can use nano/vim.

***GVSU EOS Student Website Instructions***

Please see the setup guide in Blackboard! You'll need to put your files in /WEB\_STUDENT/<your-username>

**Don't forget the Common folder that has the library code we use!**

Here is some boilerplate code for your index.html page - replace my name with yours:

<!DOCTYPE html>

<html>

<head>

<title>Erik Fredericks - CIS367 Project Webpage</title>

</head>

<body>

<h1>Erik Fredericks - CIS367 Project Webpage</h1>

<hr size="1" />

<ul>

<li><a href="labs/triangle.html">Triangle</a></li>

<li><a href="labs/gasket1-fredericks.html">Gasket 1</a></li>

</ul>

</body>

</html>

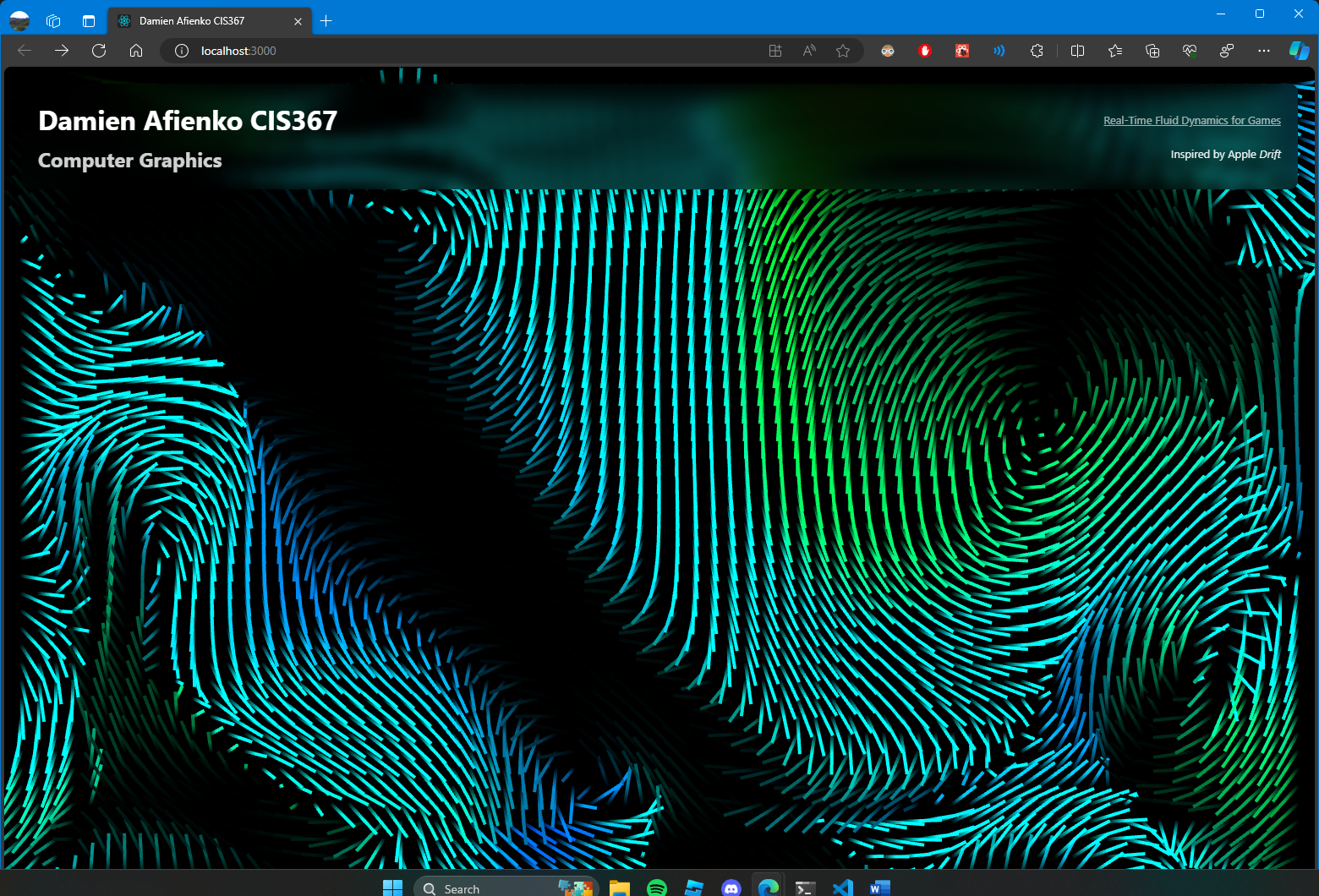
You should end up with something that looks like this (obviously with a different name):

A picture containing text

Description automatically generated

Last but not least, upload your triangle and gasket code. Note that you need both the HTML and JS files in the same location.

**Take a screenshot of your new homepage and paste it here. Make sure your labs work (i.e., that the paths to the libraries and your scripts are correct)!**

****

**Also, paste the URL to your website here:**

I made a fluid simulator.