CMP_SC 4610 - Computer Graphics

Ye Duan
Samuel Bishop

01/26/21

a. Objectives

Run the HTML files associated with the assignment and look at the HTML and JavaScript source code in order to understand how the web graphics are being drawn. Running these programs will allow students to become familiar with common computer graphics conventions in WebGL. Within this assignment students should learn how to render basic objects created in JavaScript, understand the concept of vertices, understand why composition of triangles is important in modern graphics, and understand how to use HTML code to format the JavaScript objects.

The List of Tasks to be Completed is Included Below:

Part A: (20 points)

Run the program triangle.html

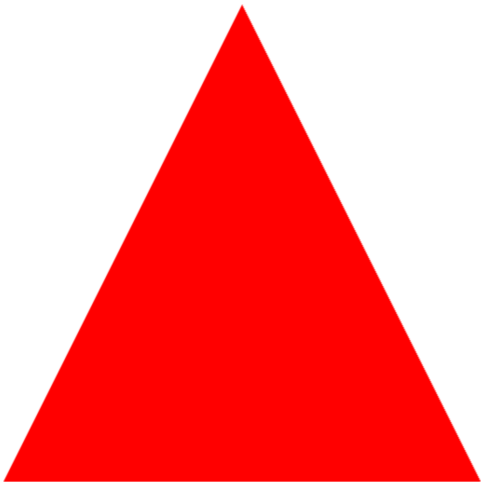

Figure 1. The graphical result of executing the triangle.html file

Part B: (20 points)

Run the Sierpinski Gasket programs gasket2.html

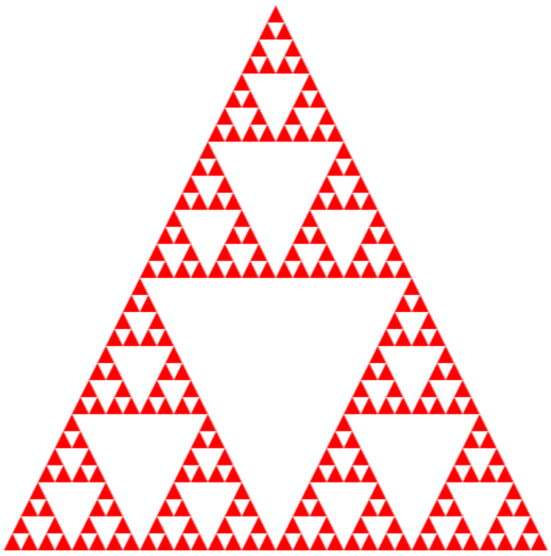

Figure 2. The graphical result of executing the gasket2.html file

Part C: (20 points)

Run the cube.html

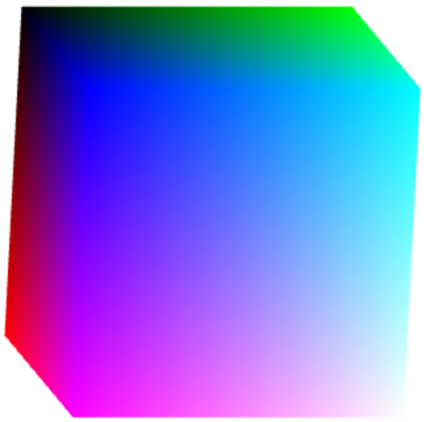

Figure 3. The graphical result of executing the cube.html file

Part D: (40 points)

Modify the program in Part A:

- Display a square instead of a triangle.
- Display the square in a different color.

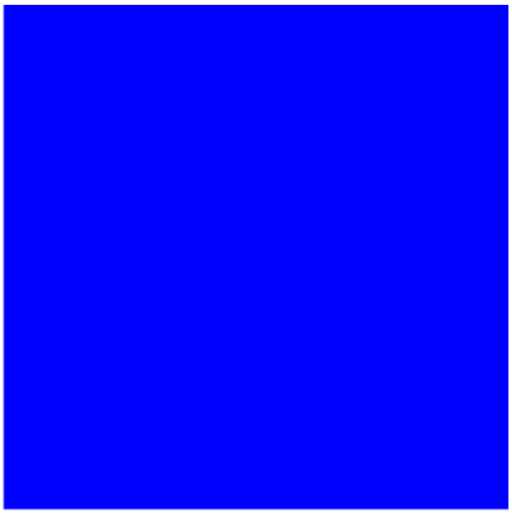

Figure 4. The graphical result of executing the partd.html file

b. Methods

Methods used to complete tasks are listed task-by-task below:

Number	Method(s)	Parameter(s)	Explanation
1	gl.viewport()	0,0, height, width	Set the viewbox of WebGL window
2	glclearColor()	Red, green, blue, alpha	Set background color
3	initShaders()	Resource, vertexShaderSource, FragmentShaderSource	Initialize shaders
4	gl.useProgram()	program	Takes in shader information and uses it for rendering

5	gl.createBuffer()	NULL	Creates and initializes an empty WebGLBuffer object
6	gl.bindBuffer()	Target, buffer	Binds a given WebGLBuffer object to a target
7	gl.bufferData()	Target, sourceData, usage	Initializes and creates the buffer object's data store.
8	gl.getAttribLocation()	Program, name	Returns the location of an attribute variable in a WebGLProgram
9	gl.vertexAttribPointer()	Index, size, type, normalized, stride, offset	Binds the buffer currently bound to gl.ARRAY_BUFFER to a generic vertex attribute of the current vertex buffer object and specifies layout
10	gl.enableVertexAttribArray()	index	Turns on the generic vertex attribute at the specified index into a list of attribute arrays
11	gl.clear()	mask	Clears buffers to preset values
12	gl.drawArrays	Mode, first, count	Renders primitives from array data

c. Encountered Issues and Solutions

Very few issues were encountered throughout the course of this assignment.They were all eventually solved after a minimal amount of troubleshooting.

Below are some of the problems encountered while rendering the square object:

The biggest issue that took place during completion of this assignment was creating the square JavaScript object. Initially when copying and pasting the triangle.js object, I had thought it would be as easy as changing the three vertices to four vertices and then the WebGL shaders

would draw a square object. When changing these vertices did not work, I began researching why.

A few minutes later I located the answer on StackOverflow. The rendering engine for WebGL (and pretty much all other graphics engines), functions by drawing a series of triangles, as shown in the render function gl.drawArrays(gl.Triangles, 0, 3); statement. Instead of viewing a square object as one cohesive square unit, one must break the square into two triangles and label the vertices individually.

After tinkering around with the vertices for a couple attempts and changing the amount of vertices drawn in the drawArray method. A square object was formed.

A separate issue was encountered when trying to change the color of the square. I could not find any code relating to the square object's color in the JavaScript file. I fixed this problem by going to the *partd.html* file and finding a variable labeled fColor. I changed the values within the fColor variable from [1.0, 0.0, 0.0, 1.0] to [0.0, 0.0, 1.0, 1.0] (red, green, blue, alpha) which resulted in the square changing blue.

d. Youtube Video Demo

https://youtu.be/7_ZA1U8PEpI

References

Angel, Edward, and Dave Shreiner. *Interactive Computer Graphics*. 8th ed., Pearson, 2020.