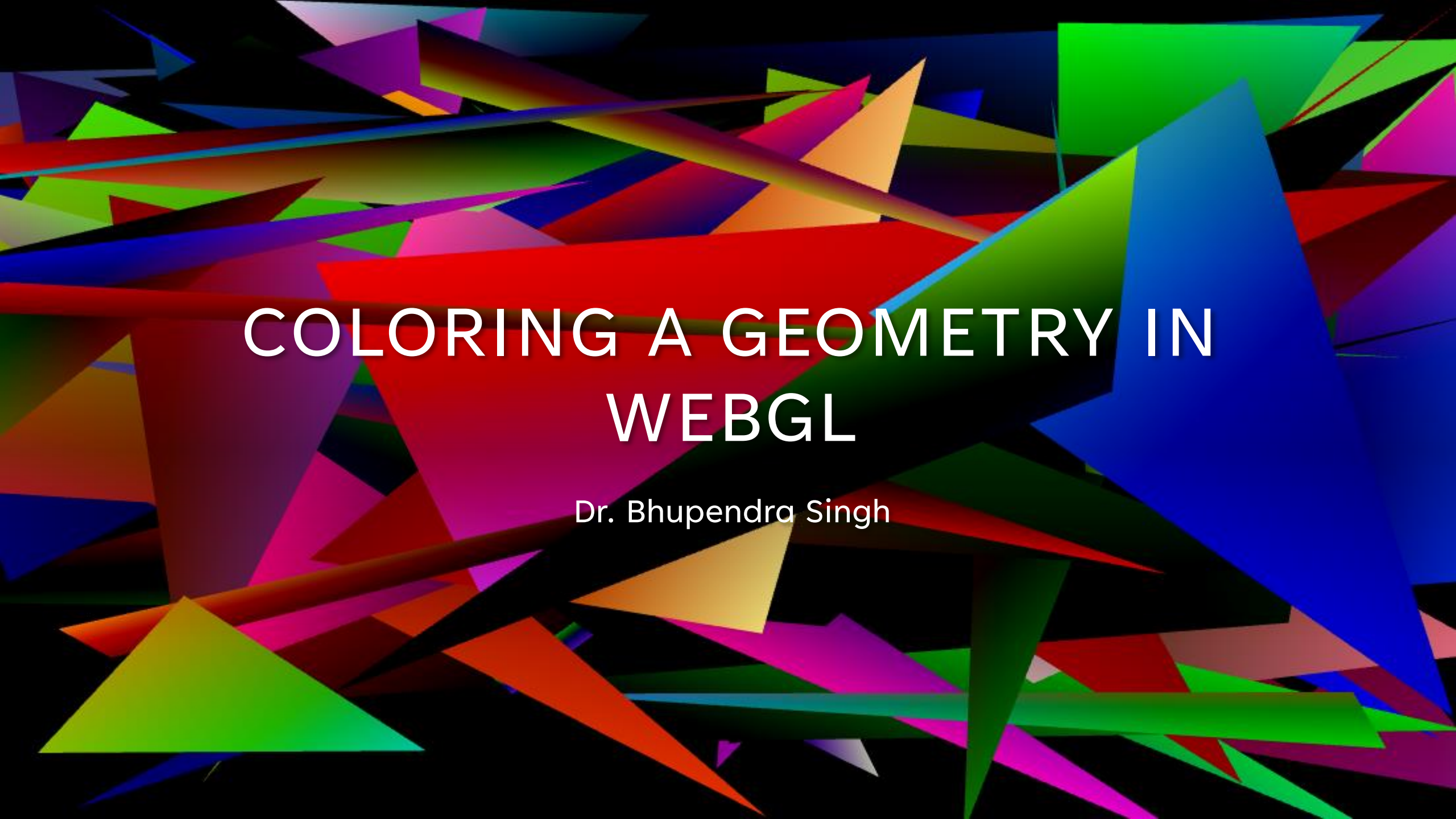

COLORING A GEOMETRY IN WEBGL
Dr. Bhupendra Singh

PREREQUISITE

UNDERSTANDING OF 18 STEPS OF WEBGL PROGRAMMING

TOPICS COVERED

Introduction

5 Steps for adding color

Coloring few more geometries

Coloring complexity

Random colored triangles

INTRODUCTION

- Coloring of the geometry is the first step towards adding realism to the geometries in the virtual world.
- A total of 5 steps are required to be updated to color any geometry in WebGL.
- Let’s color a triangle with three colors.

FIRST:
PROVIDE COLOR INFORMATION

- Updated Step 2: Define coordinates and color information for each vertex of the triangle in the JS array

```
var verticesDataArrayJS =
[ //X Y Z R G B
     0.0 , 0.8 , 0, 1, 0, 0, // Vertex A
     0.8 , -0.8, 0, 0, 1, 0, // Vertex B
    -0.8, -0.8, 0, 0, 0, 1 // Vertex C
];
```

- There are a total of 18 data points in the JS array with 6 data points for each vertex.
- For each vertex, the first 3 data points are for the X, Y, and Z coordinates
- And the next 3 data points are for the R, G, and B color information.

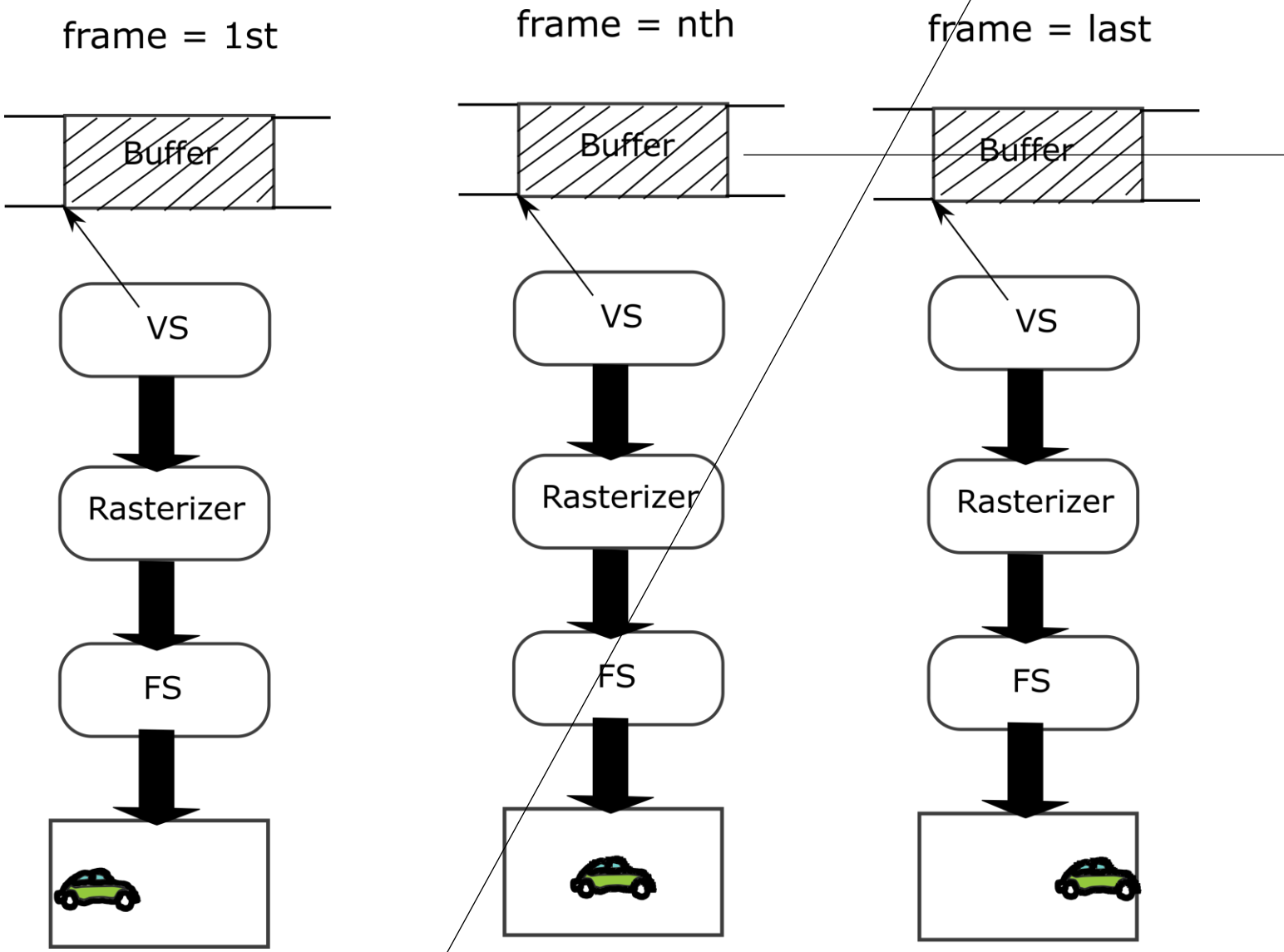
SECOND: UPDATE VERTEX SHADER

- Updated Step 7: Define the pointers for pointing to the VBO for coordinates and color information.

```
var vertexShaderText = '# version 300 es
 # pragma vscode_glslint_stage : vert
 in vec3 geometryCoordinatesGPU ;
 in vec3 colorInfoAttrib ;
 out vec3 colorInfoVarying ;
 void main ()
 {
   gl_Position = vec4 ( geometryCoordinatesGPU ,
                                1.0) ;
   colorInfoVarying = colorInfoAttrib ;
 }';
```

- Note: FS doesn't have access to the VBO by itself. For this purpose, we have declared one more variable colorInfoVarying of type out.

WORKING OF THE TRIO(REVISIT)



THIRD:
UPDATE FRAGMENT SHADER

- Update Step 10: let's update the FS code in this step as-

```
var fragmentShaderText = '# version 300 es
 # pragma vscode_glslint_stage : frag
 precision mediump float ;
 out vec4 fragColor ;
 in vec3 colorInfoVarying ;
 void main ()
 {
       fragColor = vec4 ( colorInfoVarying , 1.0) ;
 }';
```

FOURTH:
GET POINTERS FROM SHADERS

- Updated Step 15: Get the coordinates and color pointers

```
var coordinatesInfoPoniter = gl. getAttribLocation ( shaderProgram ,
                                                    ’ geometryCoordinatesGPU ’);
var colorInfoPointer = gl. getAttribLocation ( shaderProgram , ’colorInfoAttrib ’);
```

- Updated Step 16: Also enable the pointers to access the VBO

```
gl. enableVertexAttribArray ( coordinatesInfoPoniter );
gl. enableVertexAttribArray ( colorInfoPointer );
```

FIFTH:
DETAIL DATA ORGANIZATION(X,Y,Z)

- Updated Step 17: Specify to the VS, how to fetch the coordinates

```
vertexAttribPointer (
    coordinatesInfoPoniter , // Vertices pointer
    3, // Number of elements per attribute
    gl.FLOAT , // Type of elements
    gl.FALSE , // Data Normalization
    6 * Float32Array . BYTES_PER_ELEMENT ,
            // Size of an individual vertex
    0 // Offset from the beginning
);
```

FIFTH:
DETAIL DATA ORGANIZATION(R,G,B)

- Updated Step 17: Specify how to fetch the color information

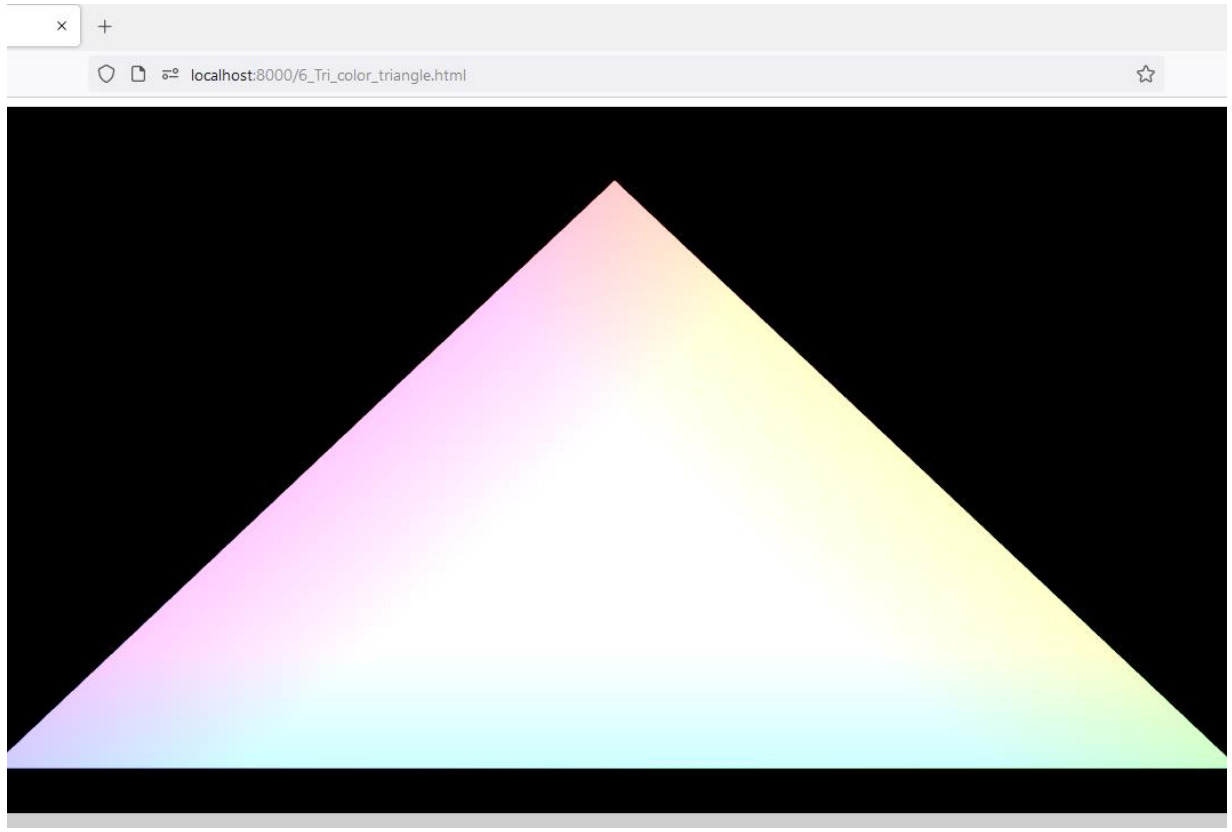
```
gl. vertexAttribPointer (
        colorInfoPointer , // Color pointer
        3, // Number of elements per attribute
        gl.FLOAT , // Type of elements
        gl.FALSE , // Data Normalization
        6 * Float32Array .
                BYTES_PER_ELEMENT ,
        // Size of an individual vertex
        3 * Float32Array .
                BYTES_PER_ELEMENT
        // Offset from the beginning
);
```

FINALLY

Issue draw command

```
gl. drawElements (gl. TRIANGLES , IndicesArrayJS . length , gl.
                                    UNSIGNED_SHORT , 0);
```

FINAL OUTPUT

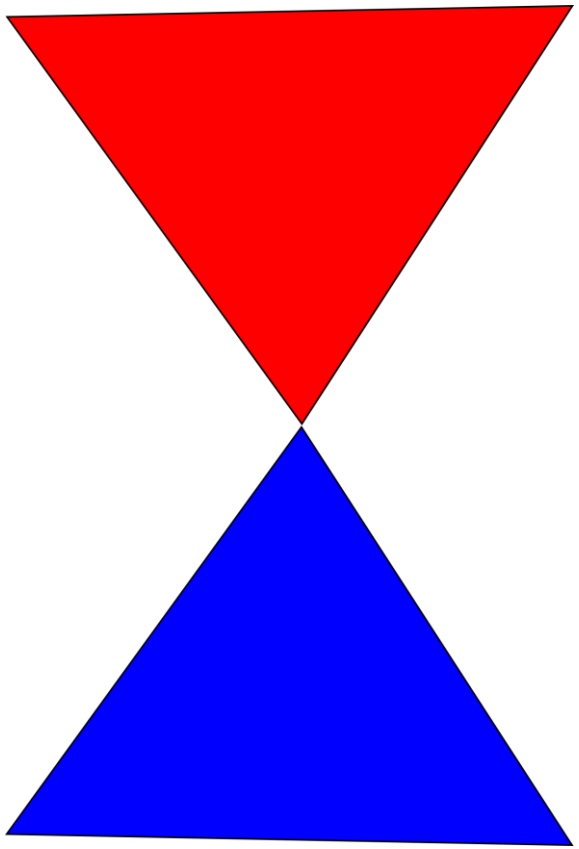


alpha = 0.7

alpha = 0.1

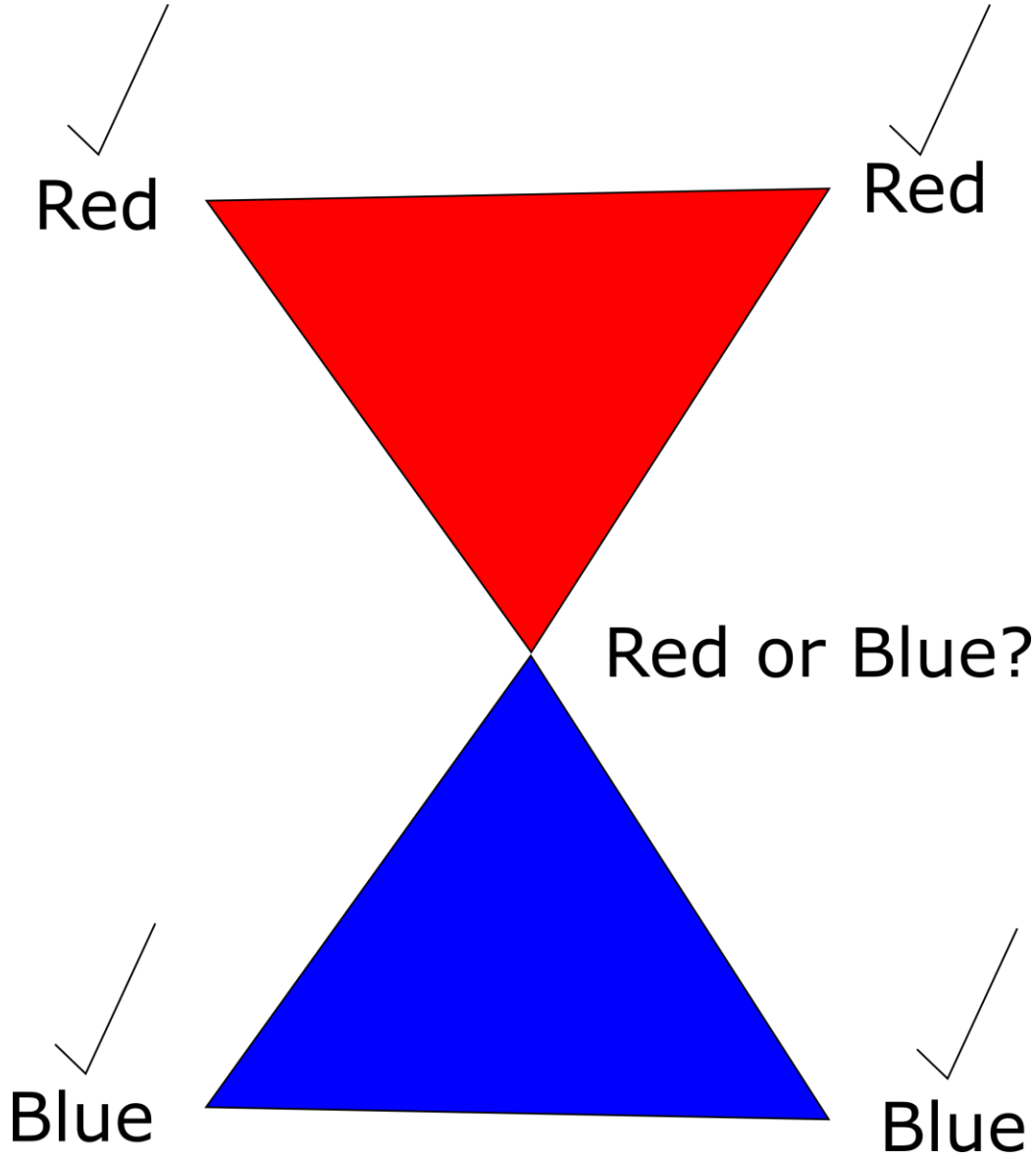
PLAYING WITH ALPHA

COMPLEXITIES IN COLORING

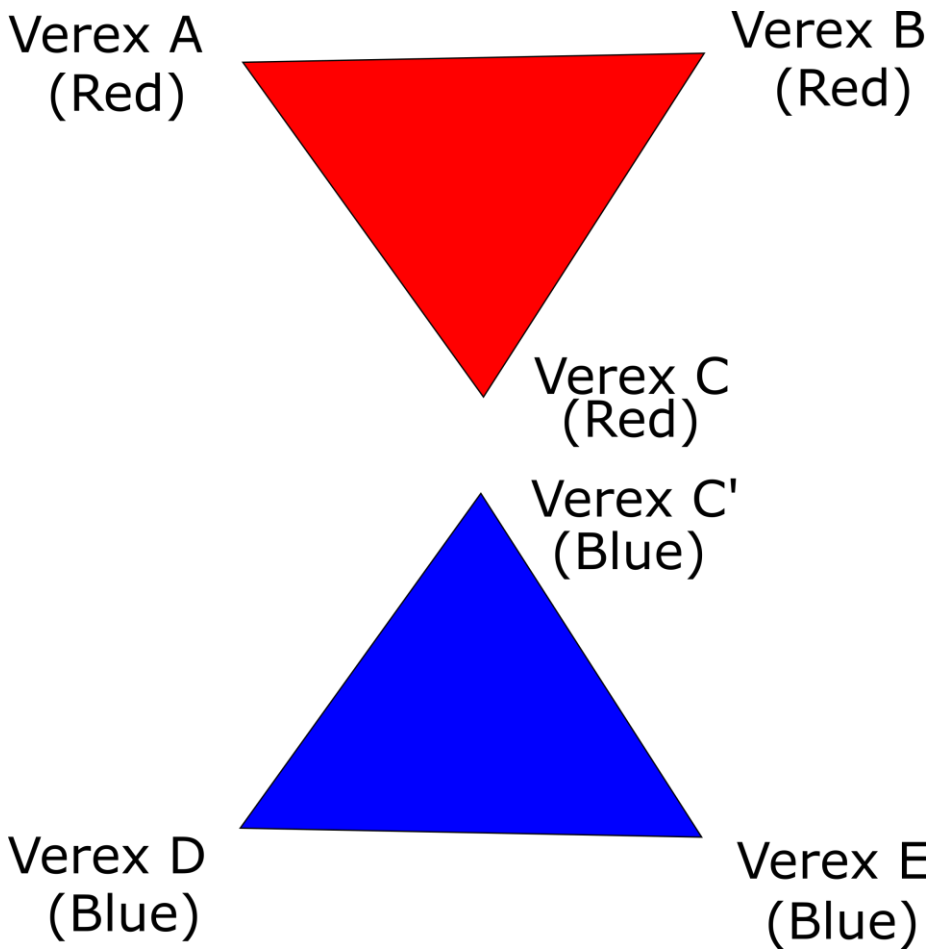


Color like this?

CONFUSION



IDEA



Vertex C: 0, 0, 0, 1, 0, 0

Vertex C’: 0, 0, 0, 0, 0, 1

LET'S CODE FOR IT

- Updated Step 2: Define the vertices with XYZ and RGB information:

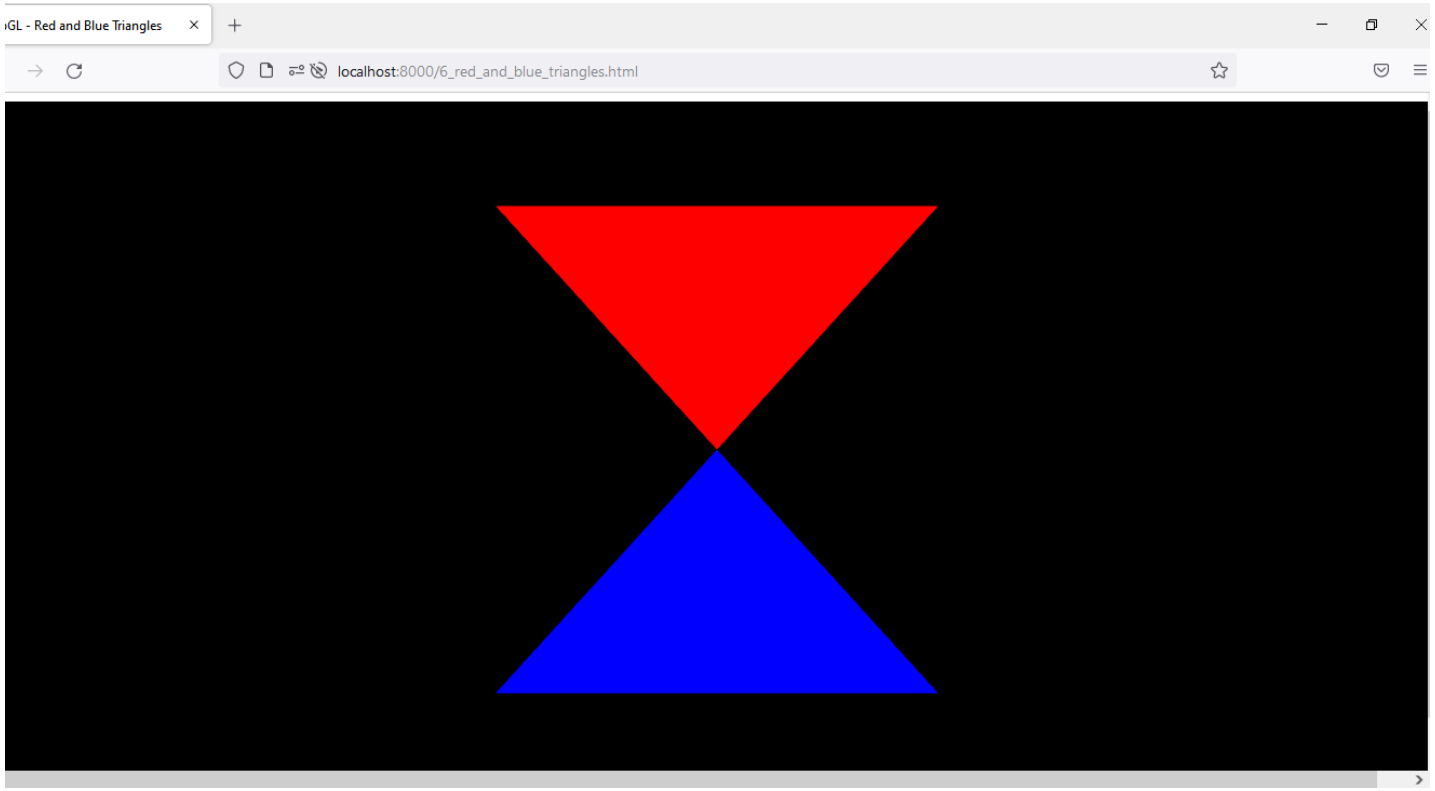
```
var verticesDataArrayJS =
[ // X, Y, Z
    -0.3, 0.7 , 0, 1, 0, 0, // Vertex A
     0.3 , 0.7 , 0, 1, 0, 0, // Vertex B
     0, 0, 0, 1, 0, 0, // Vertex C
     0, 0, 0, 0, 0, 1,
         // Vertex C ', repeat of 'Vertex C for different color
     0.3 , -0.7, 0, 0, 0, 1, // Vertex D
    -0.3, -0.7, 0, 0, 0, 1 // Vertex E
];
```

LET'S CODE FOR IT

- Updated Step 3: Specify the order of the indexes of the vertices to draw the geometry.

```
var IndicesArrayJS =
[
      // A -0, B -1, C -2, C ’-3, D -4, E -5
       0, 1, 2, // ABC
       3, 4, 5 //C’DE
];
```

- Done! Let’s see the output.



OUTPUT COLORED GEOMETRY

TASK: RANDOM COLORED TRIANGLES

- To do: Fill the screen with a random color and random position triangles.

Hint: use Math.random()

CODE

- Updated Step 2: Specify the code for randomly generating the vertices data.

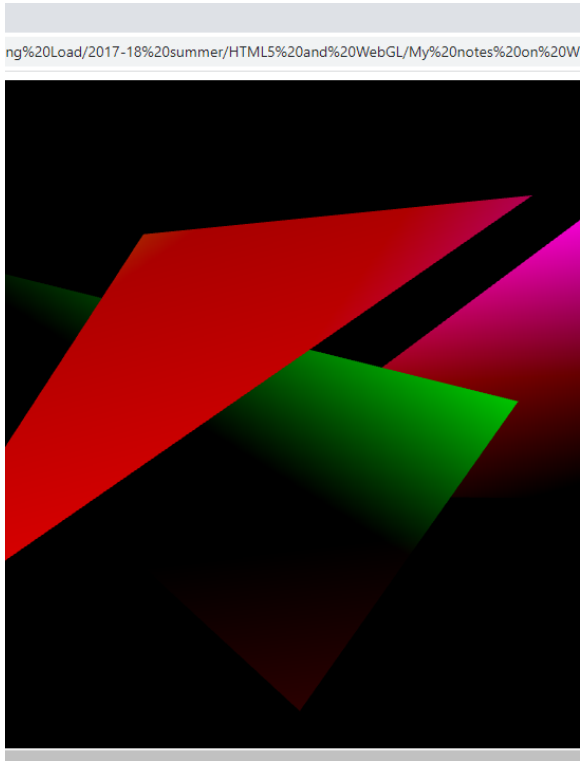
```
var total_vertices = 3
var verticesDataArrayJS = [];
for(let i = 0; i < total_vertices *6 ; i++)
{
	randomData = Math.random() * 2 - 1;
	verticesDataArrayJS.push( randomData );
}
```

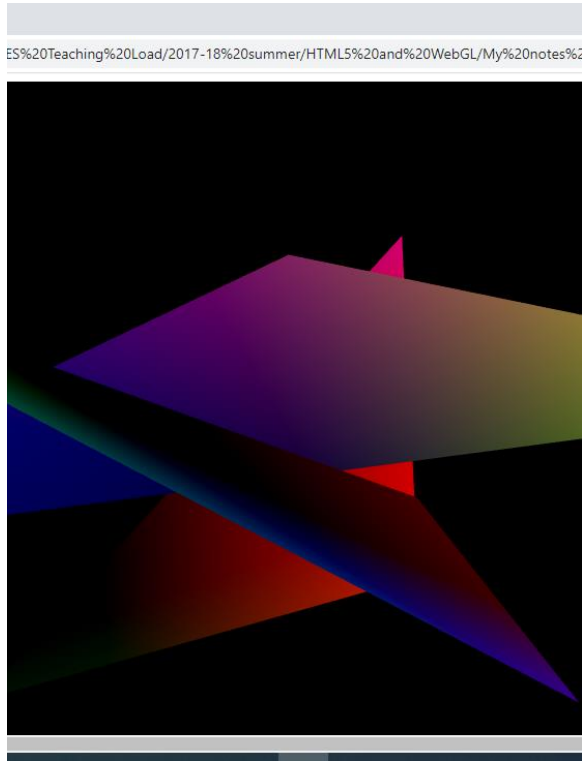
CODE

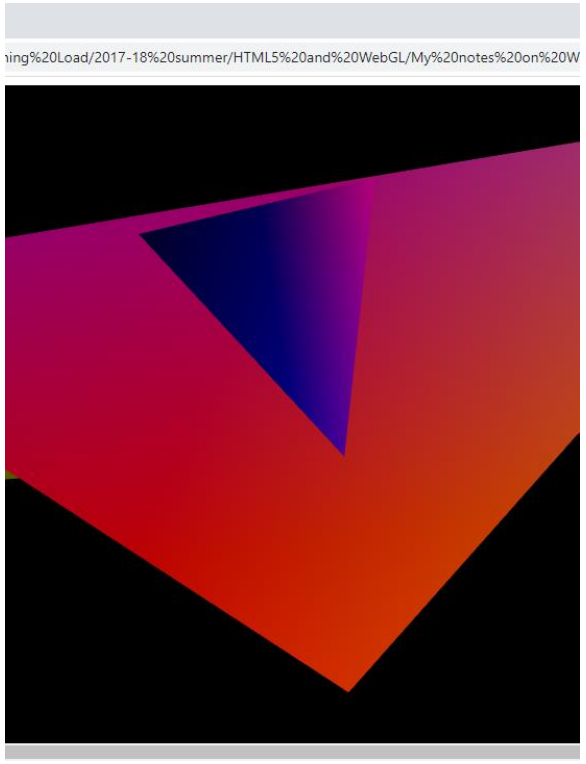
• Updated Step 3: Specify the code for getting the indices array.

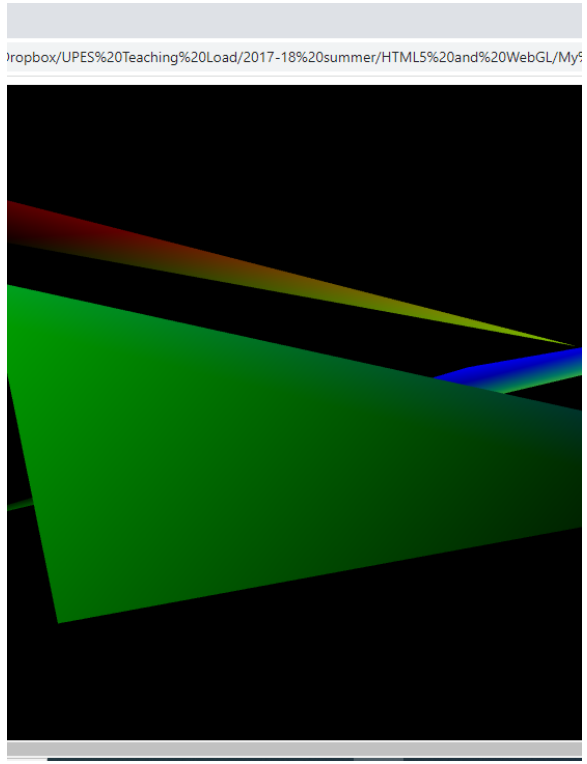
```
var IndicesArrayJS = [];
 for(let i = 0; i< total_vertices ; i++)
{
      IndicesArrayJS . push (i);
}
```

OUTPUT # 9 VERTICES

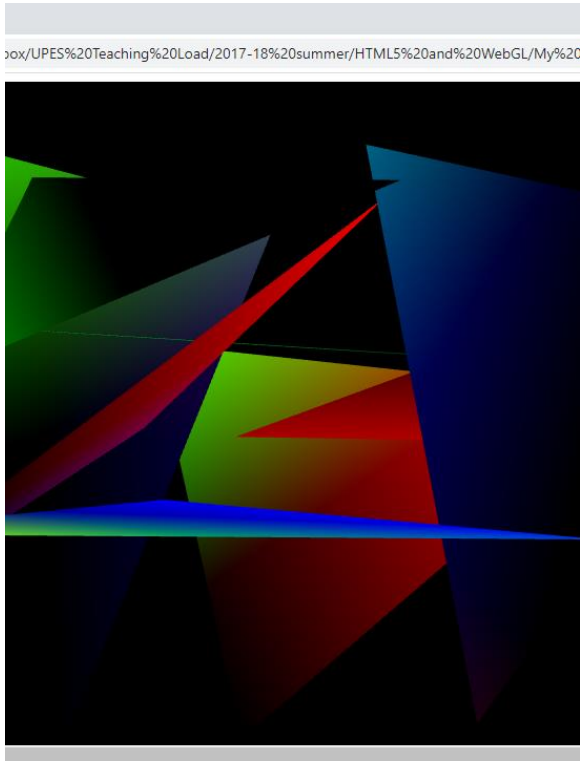


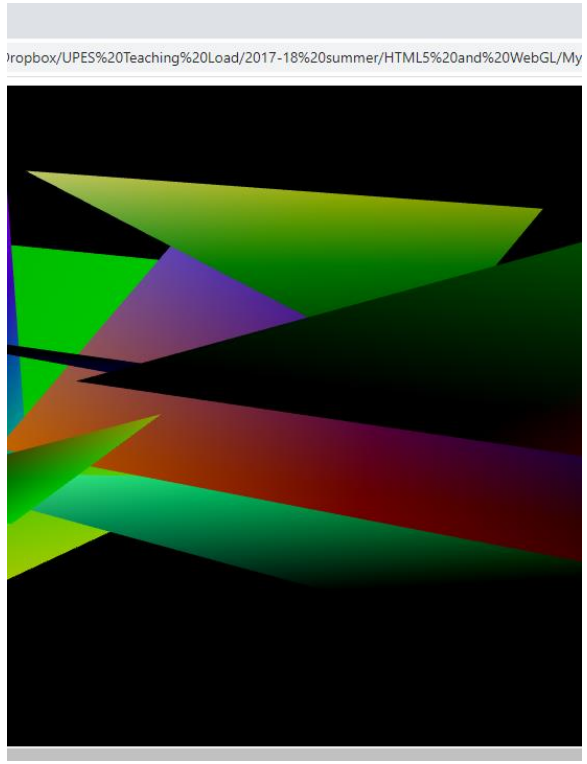


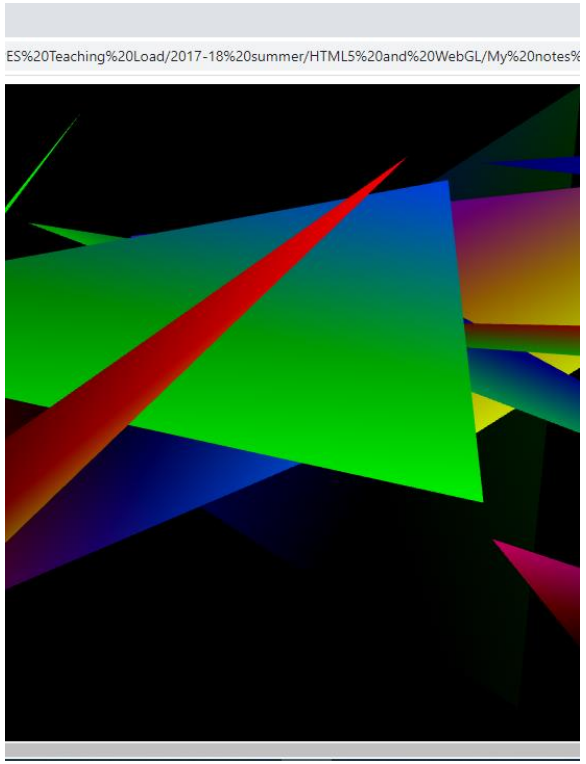


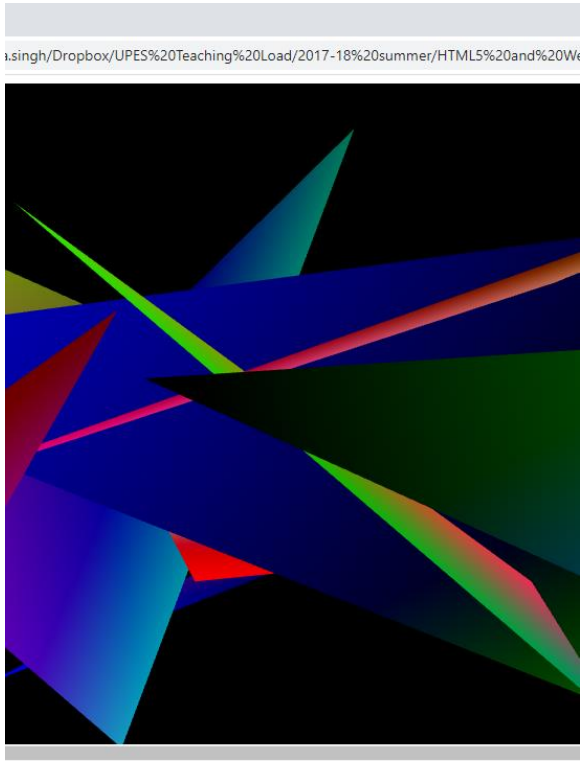


OUTPUT # 30 VERTICES

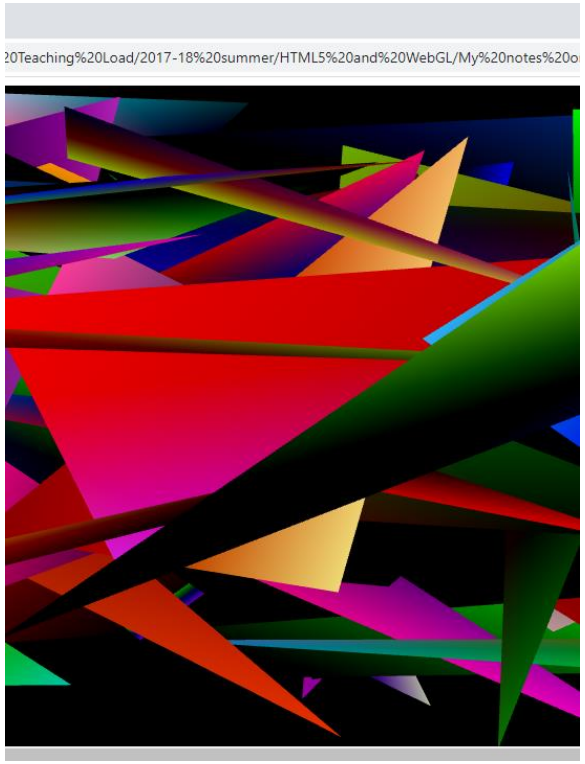


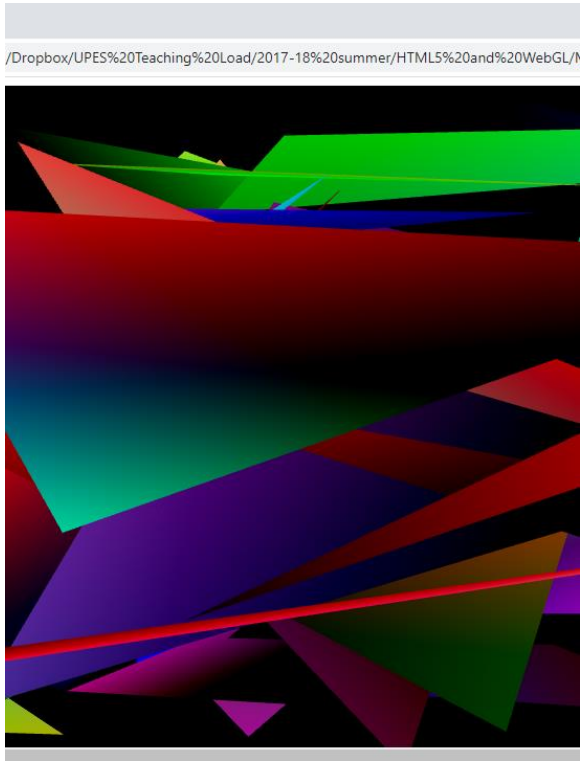


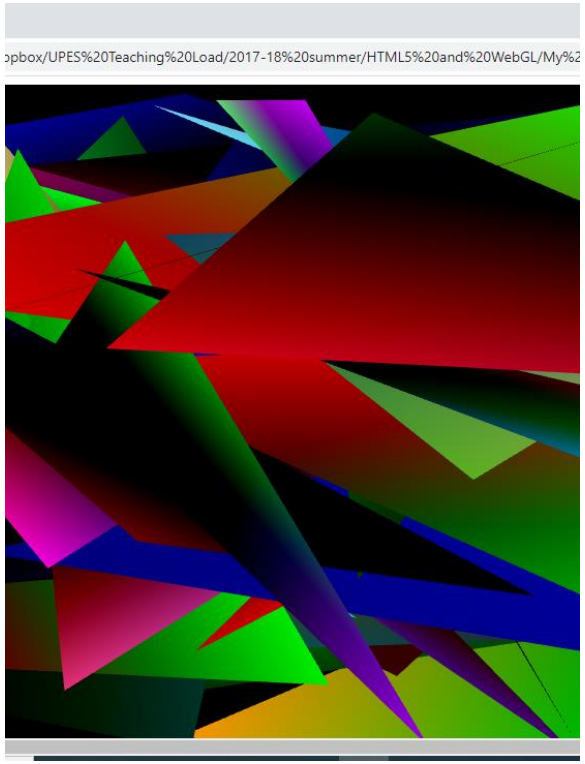


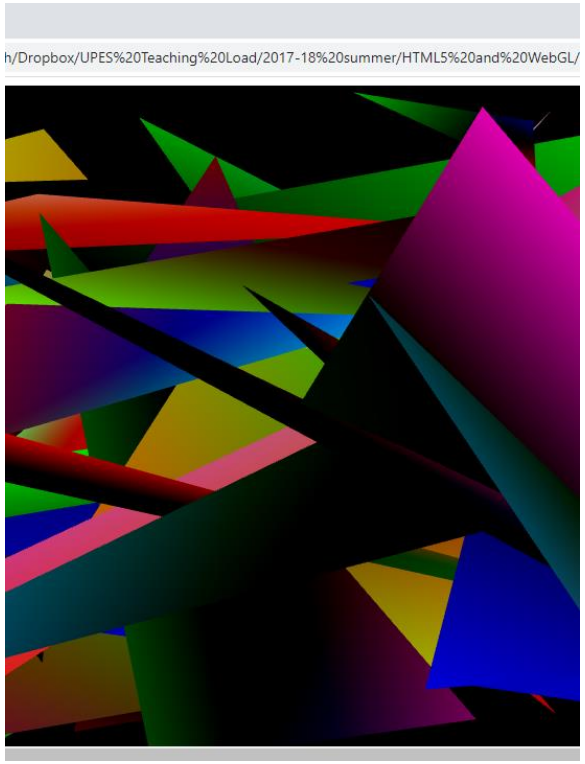


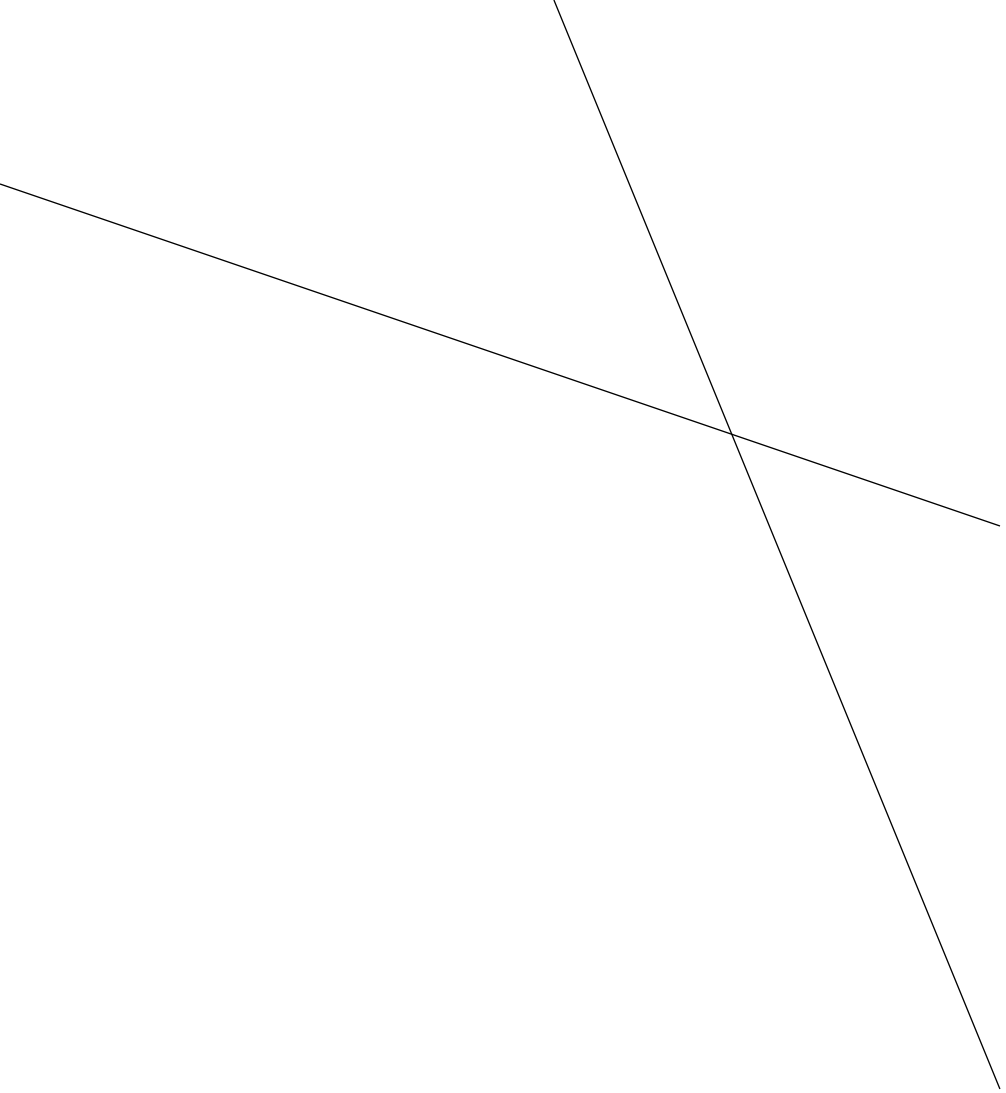
OUTPUT # 180 VERTICES











SUMMARY

Coordinates, colors, normals and all other information has to be provided in a single JS array. It is later loaded to the GPU memory as VBO. FS don't have access to the VBO and thus the color information has to be passed from the VS to the FS.

QUESTIONS, QUERIES?

You can check out my short book titled “Getting started with WebGL” available on kindly for detailed description of the topics covered here.

If you still have any queries or doubt regarding the content, do connect with me @ bhupendra.bisht59@gmail.com with subject line Questions in WebGL