CSE4204

LAB-3 : Index buffer and Transformation Matrices

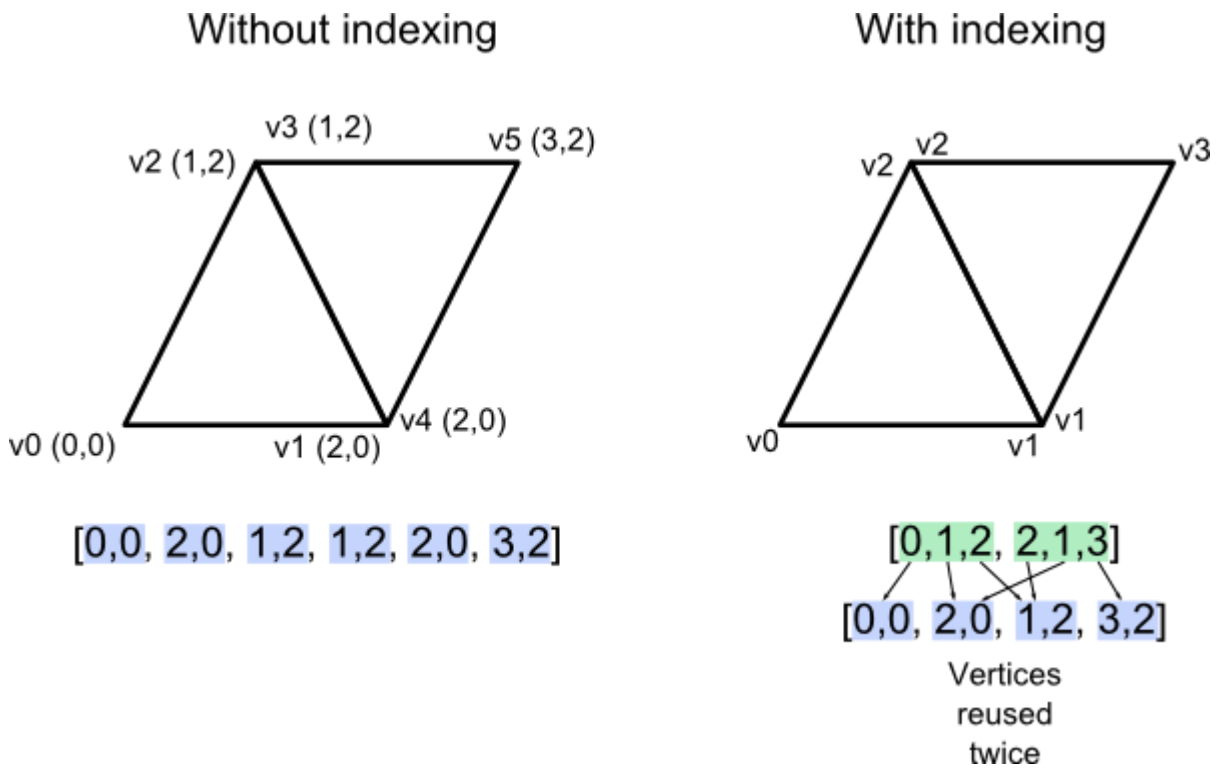
Mohammad Imrul Jubair

Recap: Uniform vs Attribute vs Varying

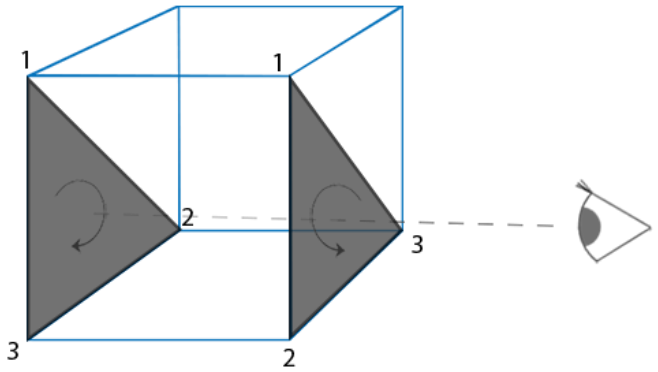
- uniform are per-primitive parameters
  - constant during an entire draw call
- attribute are per-vertex parameters
  - typically : positions, normals, colors, UVs, ...
- varying are per-fragment (or per-pixel) parameters
  - they vary from pixels to pixels

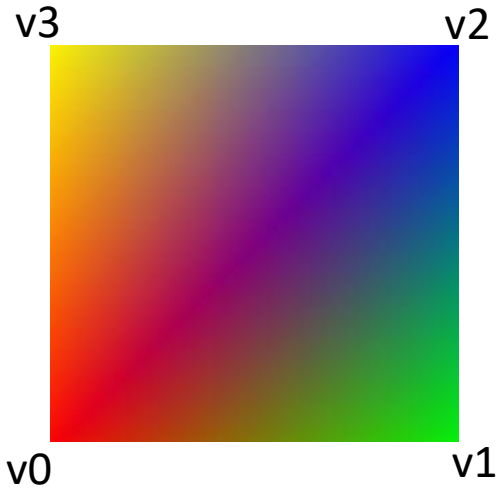
Source: https://stackoverflow.com/questions/17537879/in-webgl-what-are-the-differences-between-an-attribute-a-uniform-and-a-varying

Index Buffer



Index Buffer





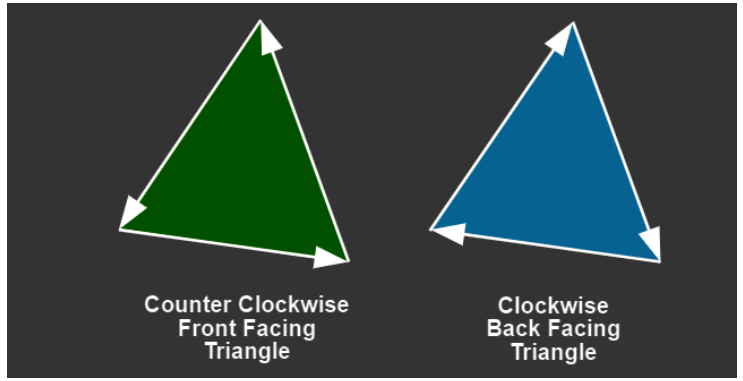

```
var coords = new Float32Array( [
                    -0.5, -0.5,  0.0, //v0
                     0.5, -0.5,  0.0, //v1
                     0.5,  0.5,  0.0, //v2
                    -0.5,  0.5,  0.0  //v3
                                    ] );
```

```
var colors = new Float32Array( [
                        1.0, 0.0, 0.0, //color at v0
                        0.0, 1.0, 0.0, //color at v1
                        0.0, 0.0, 1.0, //color at v2
                        1.0, 1.0, 0.0 //color at v3
                                    ] );
```

```
var indices = new Uint8Array([0, 1, 2,  0, 2, 3]);
```

indexing cannot be negative, so used unsigned integer.

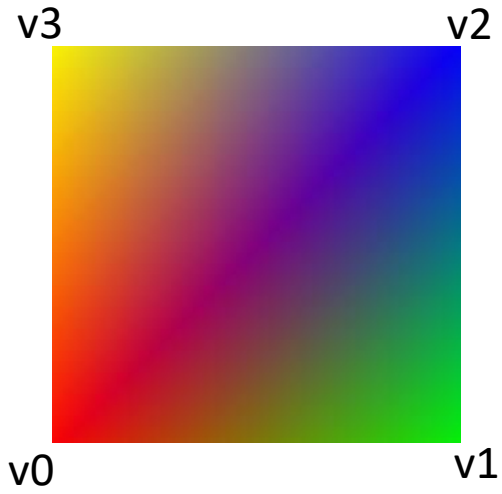
We will place the indexes in the counter clockwise direction. This is for the front face. In the top left diagram, 312 is counter clockwise For backface, we use clockwise. For the above diagram, clockwise is 231. If we want to look from left side, then the left side will become



https://webglfundamentals.org/webgl/lessons/webgl-3d-orthographic.html

Index Buffer

```
var coords = new Float32Array( [
                    -0.5, -0.5,  0.0, //v0
                     0.5, -0.5,  0.0, //v1
                     0.5,  0.5,  0.0, //v2
                    -0.5,  0.5,  0.0  //v3
                                    ] );
```



previously we passed only ARRAY_BUFFER, here we are passing ELEMENT_ARRAY_BUFFER for the indexes.

```
var colors = new Float32Array( [
                    1.0, 0.0, 0.0, //color at v0
                    0.0, 1.0, 0.0, //color at v1
                    0.0, 0.0, 1.0, //color at v2
                    1.0, 1.0, 0.0 //color at v3
                                    ] );
```

```
var indices = new Uint8Array([0, 1, 2,  0, 2, 3]);
```

```
var bufferInd = gl.createBuffer();
gl.bindBuffer(gl.ELEMENT_ARRAY_BUFFER, bufferInd);
gl.bufferData(gl.ELEMENT_ARRAY_BUFFER, indices, gl.STATIC_DRAW);
```

```
//gl.drawArrays(gl.TRIANGLES, 0, 3);
gl.drawElements(gl.TRIANGLES, 3*2, gl.UNSIGNED_BYTE, 0);
```

Get the code

rb.gy/pnoyvj

Transformation Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

$$V' = S \times V$$

```
var vertexShaderSource =
    `attribute vec3 a_coords;
    attribute vec3 a_colors;
    uniform mat4 u_Scale;
    varying vec3 v_color;

    void main() {
        gl_Position = u_Scale*vec4(a_coords, 1.0);
        v_color = a_colors;
    }`;
```

mat4 create 4*4 dimension data.
Our matrix a_coords is 4d data.

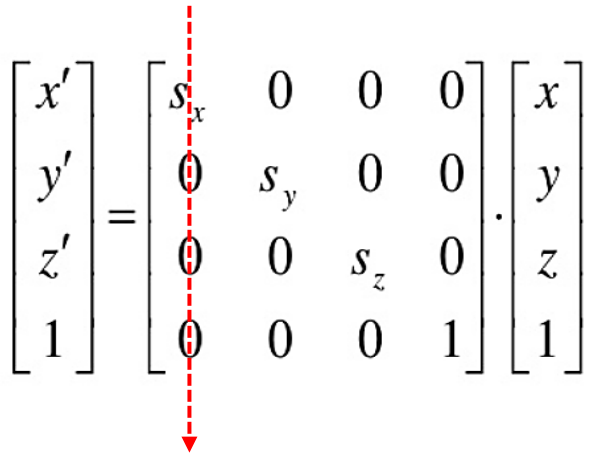
Scale Matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

```
u_scale_location = gl.getUniformLocation(prog, "u_Scale");
var Sx = 1.5;
var Sy = 0.75;
var Sz = 1.0;
var scaleMatrix = new Float32Array( [Sx,   0.0,  0.0,  0.0,
                                     0.0,  Sy,   0.0,  0.0,
                                     0.0,  0.0,  Sz,   0.0,
                                     0.0,  0.0,  0.0,  1.0] );

gl.uniformMatrix4fv(u_scale_location, false, scaleMatrix);
```

webgl is column major, the
false in above means donot transpose data.

Column Major

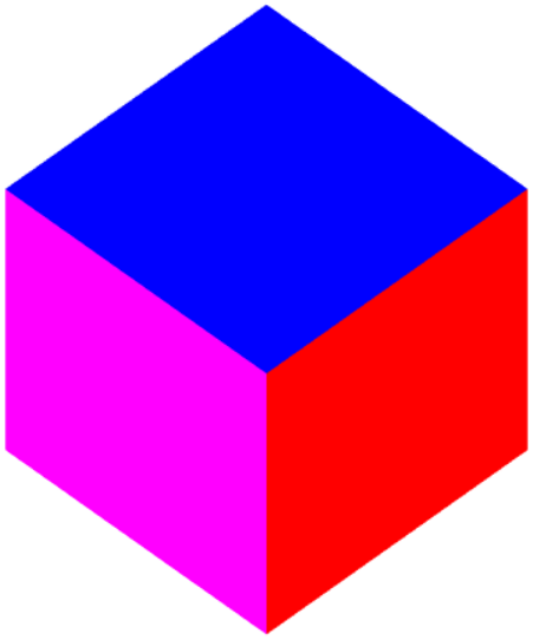



```
u_scale_location = gl.getUniformLocation(prog, "u_Scale");
var Sx = 1.5;
var Sy = 0.75;
var Sz = 1.0;
var scaleMatrix = new Float32Array( [Sx,  0.0,  0.0,  0.0,
                                     0.0,  Sy,   0.0,  0.0,
                                     0.0,  0.0,  Sz,   0.0,
                                     0.0,  0.0,  0.0,  1.0] );

gl.uniformMatrix4fv(u_scale_location, false, scaleMatrix);
```

Get the code

rb.gy/1zrhvj

3D Cube



```
var indices = new Uint8Array([
        0, 1, 2,      0, 2, 3,    // Front face
        4, 5, 6,      4, 6, 7,    // Back face
        8, 9, 10,     8, 10, 11,  // Top face
        12, 13, 14,   12, 14, 15, // Bottom face
        16, 17, 18,   16, 18, 19, // Right face
        20, 21, 22,   20, 22, 23  // Left face
    ]);
```

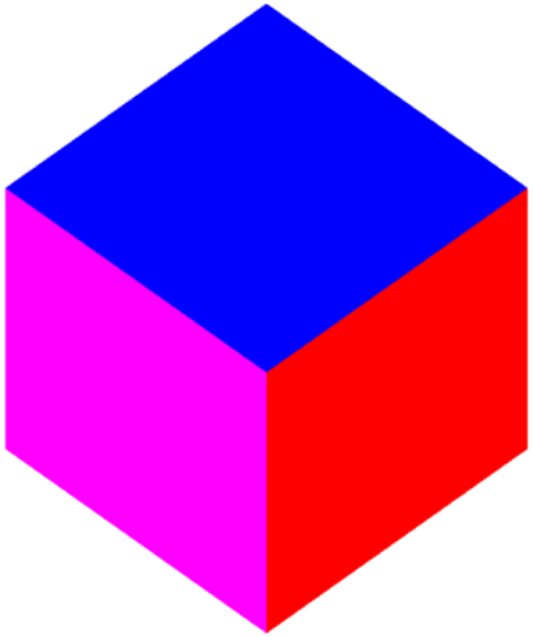
```
var coords = new Float32Array( [
                 // Front face
                -0.5, -0.5,  0.5,
                 0.5, -0.5,  0.5,
                 0.5,  0.5,  0.5,
                -0.5,  0.5,  0.5,

                // Back face
                -0.5, -0.5, -0.5,
                -0.5,  0.5, -0.5,
                 0.5,  0.5, -0.5,
                 0.5, -0.5, -0.5,

                // Top face
                -0.5,  0.5, -0.5,
                -0.5,  0.5,  0.5,
                 0.5,  0.5,  0.5,
                 0.5,  0.5, -0.5,

                // Bottom face
                -0.5, -0.5, -0.5,
                 0.5, -0.5, -0.5,
                 0.5, -0.5,  0.5,
                -0.5, -0.5,  0.5,

                // Right face
                 0.5, -0.5, -0.5,
                 0.5,  0.5, -0.5,
                 0.5,  0.5,  0.5,
                 0.5, -0.5,  0.5,

                // Left face
                -0.5, -0.5, -0.5,
                -0.5, -0.5,  0.5,
                -0.5,  0.5,  0.5,
                -0.5,  0.5, -0.5
                                    ] );
```

```
var colors = new Float32Array( [
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,
        1.0, 0.0, 0.0,

        0.0, 1.0, 0.0,
        0.0, 1.0, 0.0,
        0.0, 1.0, 0.0,
        0.0, 1.0, 0.0,

        0.0, 0.0, 1.0,
        0.0, 0.0, 1.0,
        0.0, 0.0, 1.0,
        0.0, 0.0, 1.0,

        1.0, 1.0, 0.0,
        1.0, 1.0, 0.0,
        1.0, 1.0, 0.0,
        1.0, 1.0, 0.0,

        0.0, 1.0, 1.0,
        0.0, 1.0, 1.0,
        0.0, 1.0, 1.0,
        0.0, 1.0, 1.0,

        1.0, 0.0, 1.0,
        1.0, 0.0, 1.0,
        1.0, 0.0, 1.0,
        1.0, 0.0, 1.0
        ] );
```

Depth Test + Face Culling

DEPTH_TEST and CULL_FACE IS BUILT IN, it hides the background with the foreground. CULL_FAce doesn't do the calculation for the vertexes in the background that will not be shown. So saves computation.
It does all the calculation automatically. A cube has 6 face, but we are seeing 3 face.

Gl.DEPTH_BUFFER_BIT has z axis data, that is a buffer data. It does all the calculation of which face two show and which not when we rotate view.



```
gl.clearColor(1.0, 1.0, 1.0, 1.0);
gl.enable(gl.DEPTH_TEST);
gl.enable(gl.CULL_FACE);
gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);
gl.drawElements(gl.TRIANGLES, 3*12, gl.UNSIGNED_BYTE, 0);
```

Rotation in 3D

CCW → +ve rotation

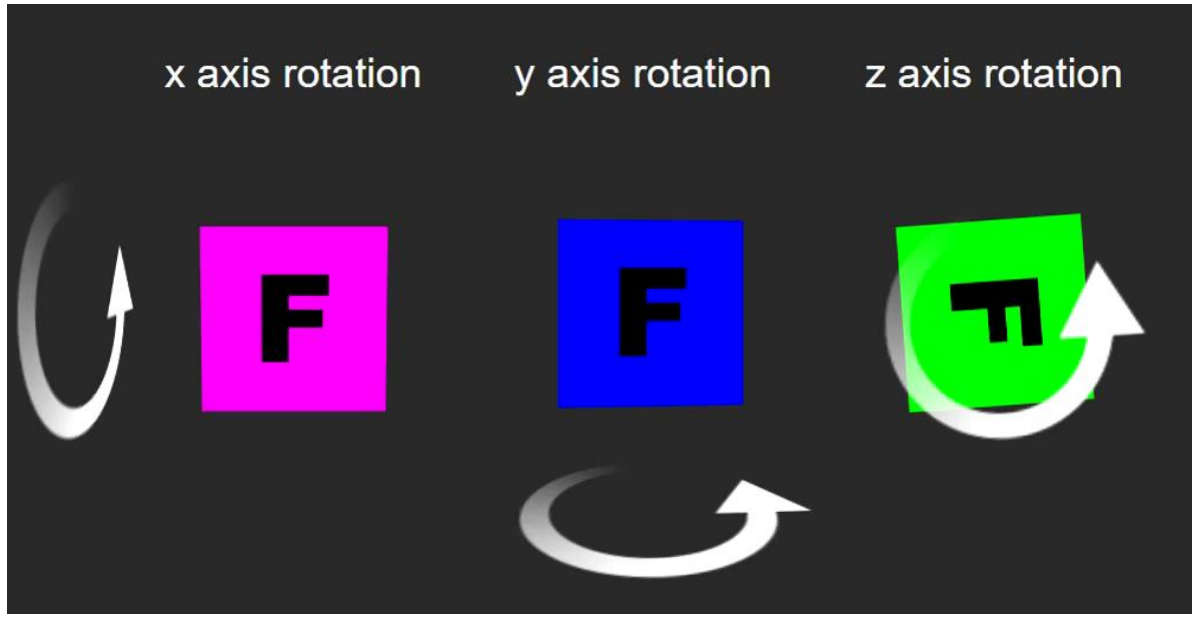
here, the rotations are all anticlockwise

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$R_z(\gamma) = \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 & 0 \\ \sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$\boldsymbol{V' = R \times V}$$



Rotation in 3D

```
var vertexShaderSource =
    `attribute vec3 a_coords;
    attribute vec3 a_colors;
    uniform mat4 u_RotY;
    varying vec3 v_color;

    void main() {
        gl_Position = u_RotY*vec4(a_coords, 1.0);
        v_color = a_colors;
    }`;
```

$$V' = R_y \times V$$

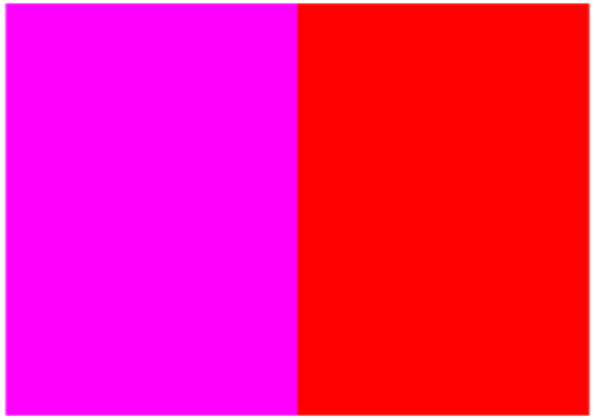
Rotation in 3D

$$R_y(\beta) = \begin{bmatrix} \cos\beta & 0 & \sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

```
var u_rotateY_location = gl.getUniformLocation(prog, "u_RotY");

var thetaY = 45;
var rad = thetaY*Math.PI/180;
var rotateYMatrix = new Float32Array( [
        Math.cos(rad),  0.0,  -Math.sin(rad),  0.0,
        0.0,            1.0,   0.0,            0.0,
        Math.sin(rad),  0.0,   Math.cos(rad),  0.0,
        0.0,            0.0,   0.0,            1.0 ] );

gl.uniformMatrix4fv(u_rotateY_location, false, rotateYMatrix);
```

column major

Rotation in 3D



```
var u_rotateY_location = gl.getUniformLocation(prog, "u_RotY");

var thetaY = 45;
var rad = thetaY*Math.PI/180;
var rotateYMatrix = new Float32Array( [
        Math.cos(rad),  0.0,  -Math.sin(rad),  0.0,
        0.0,            1.0,   0.0,            0.0,
        Math.sin(rad),  0.0,   Math.cos(rad),  0.0,
        0.0,            0.0,   0.0,            1.0 ] );

gl.uniformMatrix4fv(u_rotateY_location, false, rotateYMatrix);
```

Get the code

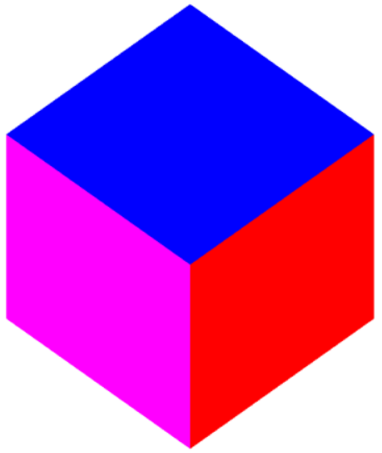
rb.gy/ah1cft

Composite Transformation

$$V' = R_x \times R_y \times V$$

```
var vertexShaderSource =
    `attribute vec3 a_coords;
    attribute vec3 a_colors;
    uniform mat4 u_RotY;
    uniform mat4 u_RotX;
    varying vec3 v_color;

    void main() {
        gl_Position = u_RotX*u_RotY*vec4(a_coords, 1.0);
        v_color = a_colors;
    }`;
```

Composite Transformation



```
var u_rotateY_location = gl.getUniformLocation(prog, "u_RotY");
var thetaY = 45;
var rad = thetaY*Math.PI/180;
var rotateYMatrix = new Float32Array( [Math.cos(rad),  0.0,  -Math.sin(rad),  0.0,
                                       0.0,            1.0,   0.0,            0.0,
                                       Math.sin(rad),  0.0,   Math.cos(rad),  0.0,
                                       0.0,            0.0,   0.0,            1.0] );

gl.uniformMatrix4fv(u_rotateY_location, false, rotateYMatrix);

var u_rotateX_location = gl.getUniformLocation(prog, "u_RotX");
var thetaX = 45;
var rad = thetaX*Math.PI/180;
var rotateXMatrix = new Float32Array( [1.0,  0.0,            0.0,            0.0,
                                       0.0,  Math.cos(rad),  Math.sin(rad),  0.0,
                                       0.0, -Math.sin(rad),  Math.cos(rad),  0.0,
                                       0.0,  0.0,            0.0,            1.0] );
gl.uniformMatrix4fv(u_rotateX_location, false, rotateXMatrix);
```

Get the code

rb.gy/1zmo7c

Composite Transformation

- Example

$$V' = R_x \times R_y \times S \times V$$