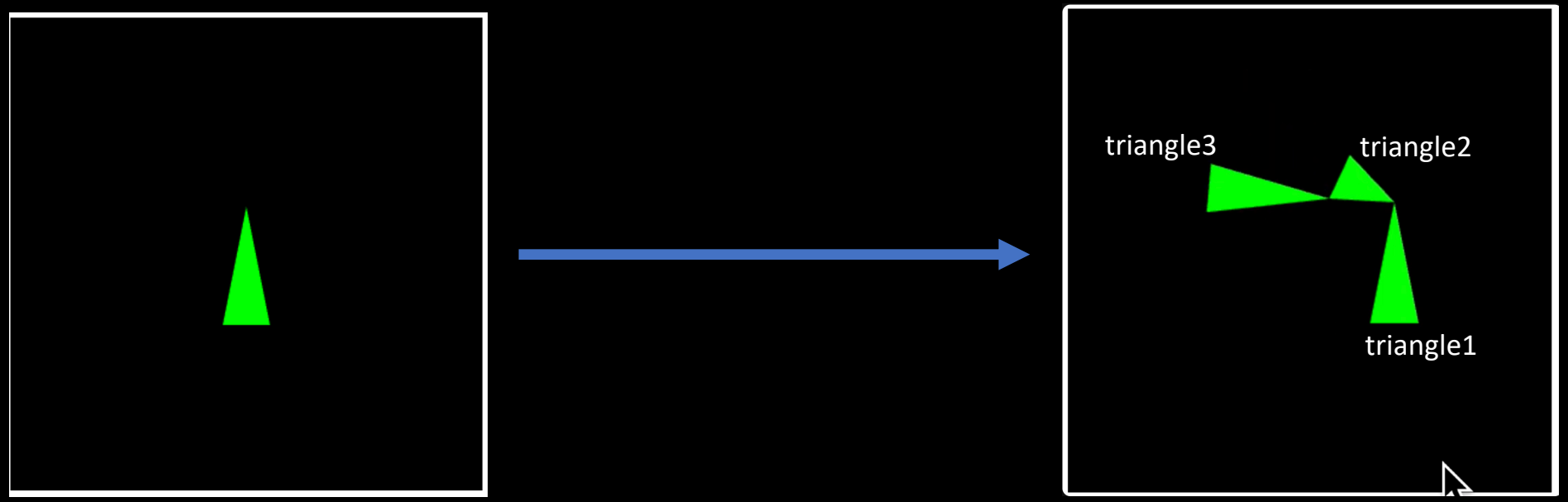
Lab 3

- Download this lab and run it, you will see the left figure (one triangle only).
- This practice ask you to add 2 extra triangles and meet the following requirement
  - Check the video here: https://www.youtube.com/watch?v=uyJhV_ozx_k&ab_channel=Ko-ChihWang
  - 1. triangle2's tip always attach at triangle1's tip
  - 2. triangle3's tip always attach at triangle2's right corner
  - 3. consider the three triangles is a complete object. Tips of triangle2 and triangle3 are two joint.
  - 4. if the user press "a" move the whole object to the left. If the user press "d" move the whole object to the right.
  - 5. if the user press "r", rotate triangle2 along its tip
  - 6. if the user press "l (L)", elongate triangle2. if the user press "s", shorten triangle2
  - 7. if the user press "o (O)", rotate triangle3 along its tip



- In WebGL.js, I set vertices information here (check the comment), do NOT change any code segment here
- I provide two triangle models (A and B). Check the differences and feel free to select any one you prefer to draw your triangles

```
var transformMat = new Matrix4(); //cuon 4x4 matrix

//NOTE: You are NOT allowed to change the vertex information here
var triangleVerticesA = [0.0, 0.2, 0.0, -0.1, -0.3, 0.0, 0.1, -0.3, 0.0]; //green rotating triangle vertices
var triangleColorA = [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0 ]; //green trotating riangle color

//NOTE: You are NOT allowed to change the vertex information here
var triangleVerticesB = [0.0, 0.0, 0.0, -0.1, -0.5, 0.0, 0.1, -0.5, 0.0]; //green rotating triangle vertices
var triangleColorB= [0.0, 1.0, 0.0, 0.0, 1.0, 0.0, 0.0, 1.0, 0.0 ]; //green trotating riangle color

var triangle1XMove = 0;
var triangle2Angle = 125;
var triangle2HeightScale = 1;
var triangle3Angle = 0;
```

- triangle1XMove: triangle1's displacement along the horizontal direction
- triangle2Angle: triangle2' rotation angle
- triangle2HeightScale: the scale factor of triangle2 along its 'height' direction
- triangle3Angle: triangle3' rotation angle

- **The first triangle**

- The only code segment you can change in this practice.

```
function draw(gl)
{
    ////clear background color by black
    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);

    transformMat.setIdentity(); //set identity matrix to transformMat

    //Note: You are NOT Allowed to change the following code
    transformMat.translate(triangle1XMove, 0, 0);
    initAttributeVariable(gl, program.a_Position, triangleModelA.vertexBuffer);//set triangle  vertex to shader varibale
    initAttributeVariable(gl, program.a_Color, triangleModelA.colorBuffer); //set triangle  color to shader varibale
    gl.uniformMatrix4fv(program.u_modelMatrix, false, transformMat.elements);//pass current transformMat to shader
    gl.drawArrays(gl.TRIANGLES, 0, triangleModelA.numVertices);//draw a triangle using triangleModelA
    ////////////// END: draw the first triangle

    //////////TODO: draw the other 2 triangles////////////////
    /////////The code segment from here to the end of draw() function
    /////////   is the only code segment you are allowed to change in this practice
}
```

```
function main(){
    //////Get the canvas context
    var canvas = document.getElementById('webgl');
    var gl = canvas.getContext('webgl2');
    if(!gl){
        console.log('Failed to get the rendering context for WebGL');
        return ;
    }

    /////compile shader and use it
    program = compileShader(gl, VSHADER_SOURCE, FSHADER_SOURCE);
    gl.useProgram(program);

    /////prepare attribute reference of the shader
    program.a_Position = gl.getAttribLocation(program, 'a_Position');
    program.a_Color = gl.getAttribLocation(program, 'a_Color');
    program.u_modelMatrix = gl.getUniformLocation(program, 'u_modelMatrix');
    if(program.a_Position<0 || program.a_Color<0 || program.u_modelMatrix < 0)
        console.log('Error: f(program.a_Position<0 || program.a_Color<0 || .....');

    /////create vertex buffer of rotating point, center points, rotating triangle for later use
    triangleModelA = initVertexBufferForLaterUse(gl, triangleVerticesA, triangleColorA);
    triangleModelB = initVertexBufferForLaterUse(gl, triangleVerticesB, triangleColorB);
```

- We initialize triangle models here for later use

```
function draw(gl)
{
    ////clear background color by black
    gl.clearColor(0.0, 0.0, 0.0, 1.0);
    gl.clear(gl.COLOR_BUFFER_BIT);

    transformMat.setIdentity(); //set identity matrix to transformMat

    //Note: You are NOT Allowed to change the following code
    transformMat.translate(triangle1XMove, 0, 0);
    initAttributeVariable(gl, program.a_Position, triangleModelA.vertexBuffer);//set triangle  vertex to shader varibale
    initAttributeVariable(gl, program.a_Color, triangleModelA.colorBuffer); //set triangle  color to shader varibale
    gl.uniformMatrix4fv(program.u_modelMatrix, false, transformMat.elements);//pass current transformMat to shader
    gl.drawArrays(gl.TRIANGLES, 0, triangleModelA.numVertices);//draw a triangle using triangleModelA
    /////////////// END: draw the first triangle

    //////////TODO: draw the other 2 triangles/////////////////
    /////////The code segment from here to the end of draw() function
    /////////   is the only code segment you are allowed to change in this practice
}
```

- If you want to draw a triangle by ‘triangleModelB’ you need lines similar to them, but change ‘triangleModelA’ -> ‘triangleModelB’
- We will explain more next week

```
    document.addEventListener('keydown', (event)=> {
        if( event.key == 'a' || event.key == 'A'){ //move triangle1 to the left
            console.log('A')
            triangle1XMove -= 0.05;
            draw(gl)
        }else if ( event.key == 'd' || event.key == 'D'){  //move triangle1 to the right
            console.log('D')
            triangle1XMove += 0.05;
            draw(gl)
        }else if ( event.key == 'r' || event.key == 'R'){  //rotate the second triangle
            console.log('R')
            triangle2Angle += 10;
            draw(gl)
        }else if ( triangle2HeightScale < 1.5 && (event.key == 'l' || event.key == 'L')){ //elongate the second triangle
            console.log('L')
            triangle2HeightScale += 0.1;
            draw(gl)
        }else if ( triangle2HeightScale >0.2 && (event.key == 's' || event.key == 'S')){ //shorten the second triangle
            console.log('S')
            triangle2HeightScale -= 0.1;
            draw(gl)
        }else if ( (event.key == 'o' || event.key == 'O')){ //rotate the third triangle
            console.log('O')
            triangle3Angle += 10;
            draw(gl)
        }
    });
    draw(gl)
}
```

Click to collapse the range.

- In main(), this code segment is trigger when the user pressed a key. Then, we call draw(gl) and update the screen.
- You do not have to change any here.

What You Should Do for “Submission”

Submission Instruction

- Create a folder
  - Put the html and js files in the folder
  - Zip the folder
  - Rename the zip file to your student ID
    - For example, if your student ID is “40312345s”, rename the zip file to “40312345s.zip”
  - Submit the renamed zip file to Moodle

- Make sure
  - you put all files in the folder to zip
  - You submit the zip file with correct name
- You won’t get any point if
  - the submitted file does not follow the naming rule,
  - TA cannot run your code,
  - or cannot unzip your zip file.