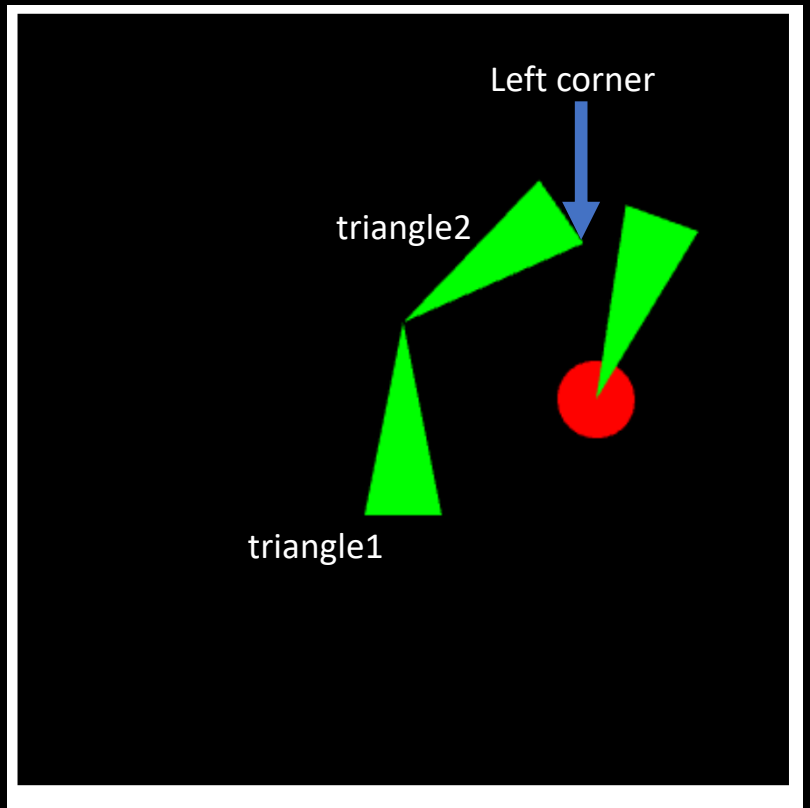
Lab 4

- Check the video in the folder
- This is an extension of Lab3
  - The template already gives you 2 triangles (same as Lab3). When you press ‘a’, ‘s’, ‘d’, ‘w’, you can move the 2 triangle to left, down, right, up. When you press ‘r’, one triangle can rotate
  - There are a circle and a rotating triangle (they are an object set)
- Requirements of this practice
  - When the left corner of the triangle 2 touches the circle, the color of the circle change to ‘light green’
  - When any part of the circle touches the left corner and the user press ‘g’ , the color of the circle change to ‘dark green’ and the center of the circle attaches on the left corner. Now, the circle is in the ‘grabbed mode’.
  - In the grabbed mode, when users move the triangle1 or rotate triangle2, the circle and the rotating triangle should move along with them.
  - In the grabbed mode, when users press ‘g’ again, it leaves ‘grabbed mode’. The circle stays at its current location



- This create a circle model
- The circle consists of many small triangles
- I also create three different color arrays for the normal mode, touch mode and grabbed mode
- The circle radius is 0.1 in the world space

```
/////// create circle model
var circleVertices = []
var circleColors = []
var circleColorsTouch = []
var circleColorsGrab = []
var circleRadius = 0.1;
for (i = 0; i <= 1000; i++){
    circleRadius = 0.1
    x = circleRadius*Math.cos(i * 2 * Math.PI / 200)
    y = circleRadius*Math.sin(i * 2 * Math.PI / 200)
    circleVertices.push(x, y);
    circleColors.push(1, 0, 0); //circle normal color
    circleColorsTouch.push(0, 1, 0); //color when the circle connect with the triangle corner
    circleColorsGrab.push(0, 0.5, 0); //color when the circle is grabbed by the triangle corner
}
```

- Setup the VBOs of triangles and circles with different colors

```
/////create vertex buffer of the two triangle models for later use
triangleModelA = initVertexBufferForLaterUse(gl, triangleVerticesA, triangleColorA);
triangleModelB = initVertexBufferForLaterUse(gl, triangleVerticesB, triangleColorB);

////create vertex buffer of the circle with red color, light green and dark green color
circleModel = initVertexBufferForLaterUse(gl, circleVertices, circleColors);
circleModelTouch = initVertexBufferForLaterUse(gl, circleVertices, circleColorsTouch);
circleModelGrab = initVertexBufferForLaterUse(gl, circleVertices, circleColorsGrab);
```

```
    ////For creating animation, in short this code segment will keep calling "draw(gl)"
    ////btw, this needs "webgl-util.js" in the folder (we include it in index.html)
    var tick = function() {
        draw(gl);
        requestAnimationFrame(tick);
    }
    tick();
}
```

- This is how we make the animation. In short, the function, draw(), is called repeatedly.
- We require a file, webgl-util.js in the working folder

- Draw the circle and the rotating triangle

```
/////// Draw the circle and its triangle
transformMat = new Matrix4(transformMatCircle1)

initAttributeVariable(gl, program.a_Position, circleModel.vertexBuffer);//set circle  vertex to shader varibale
initAttributeVariable(gl, program.a_Color, circleModel.colorBuffer); //set circle normal color to shader varibale
gl.uniformMatrix4fv(program.u_modelMatrix, false, transformMat.elements);//pass current transformMat to shader
gl.drawArrays(gl.TRIANGLES, 0, circleModel.numVertices);//draw the triangle

circle1Angle ++; //keep changing the angle of the triangle
transformMat.rotate(circle1Angle, 0, 0, 1);
initAttributeVariable(gl, program.a_Position, triangleModelB.vertexBuffer);//set triangle  vertex to shader varibale
initAttributeVariable(gl, program.a_Color, triangleModelB.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, triangleModelB.numVertices);//draw the triangle
```

Check triangle2 left corner touches the circle?

- Get the corner’s coordinate and circle’s coordinate in the “world space” to compare

- worldSpaceCoordinate = transformationMatrix * objectSpaceCoordinate

Hierarchical structure of this practice (non-grabbed mode)

Transformation that locates triangle1 in the world space

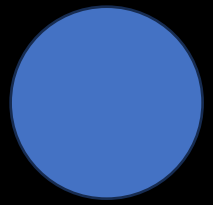
Draw triangle1

Transformation information that includes attaching two tips of triangles and rotation angles of the triangle2

Draw triangle2

Transformation that locates the circle in the world space

Draw circle



Transformation information that includes attaching the tip of the triangle at the circle center and rotation angles of the triangle

Draw rotating triangle

Hierarchical structure of this practice (non-grabbed mode)

Transformation that locates triangle1 in the world space

Draw triangle1

Transformation information that includes attaching two tips of triangles and rotation angles of the triangle2

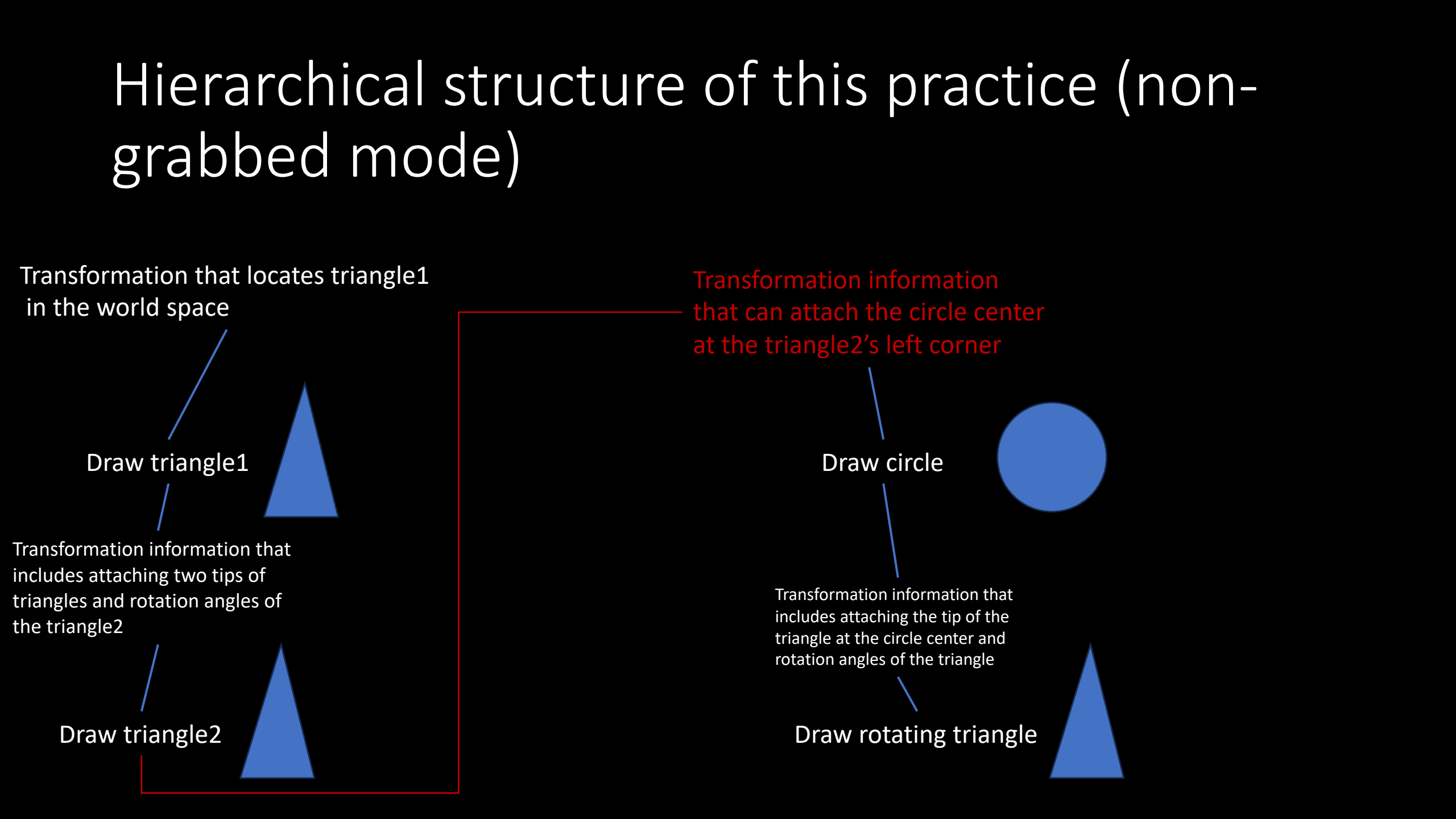
Draw triangle2

Transformation information that can attach the circle center at the triangle2's left corner

Draw circle

Transformation information that includes attaching the tip of the triangle at the circle center and rotation angles of the triangle

Draw rotating triangle



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.