CS112 Programming Assignment 0

The deadline is 11:55 PM, Tuesday Jan 16. Please submit the zip file to EEE.

1. Introduction.

In this assignment, we will get familiar with WebGL, which is “a JavaScript API for rendering interactive 3D and 2D graphics within any compatible web browser without the use of plug-ins. ” In this assignment, you won’t need to write a lot of code (maybe dozens of lines, or even less), but it might take some time to understand every part, so **please start early**.

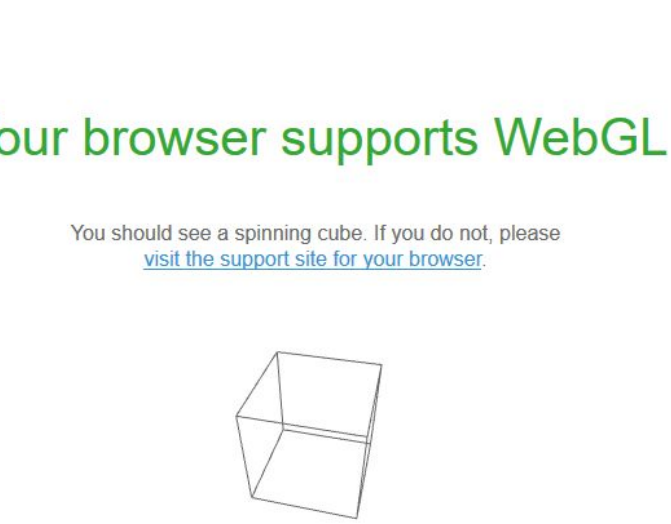
Software and hardware requirement: WebGL runs within the browser, so is independent of the operating and window systems. You may finish the assignment using any operating system you like, e.g. Windows, OSX or Linux.
Programming language: The assignment will be implemented in JavaScript. As we will minimize the use of obscure Javascript language features, it should not be too difficult for anyone with experience in a dynamic language like Python or familiar with the principles of Object Oriented Programming (like C++ or Java) to get a handle on Javascript syntax by reading through some of the code in code skeleton. For a more formal introduction to Javascript, checkout the nice tutorial from https://javascript.info/.
Cooperation and third party code:This is an individual programming assignment, and you should implement the code by your own. You may not share final solutions, and you must write up your own work, expressing it in your own words/notation. Third party codes are not allowed unless with professor’s permission.

2. Getting started with the code skeleton.

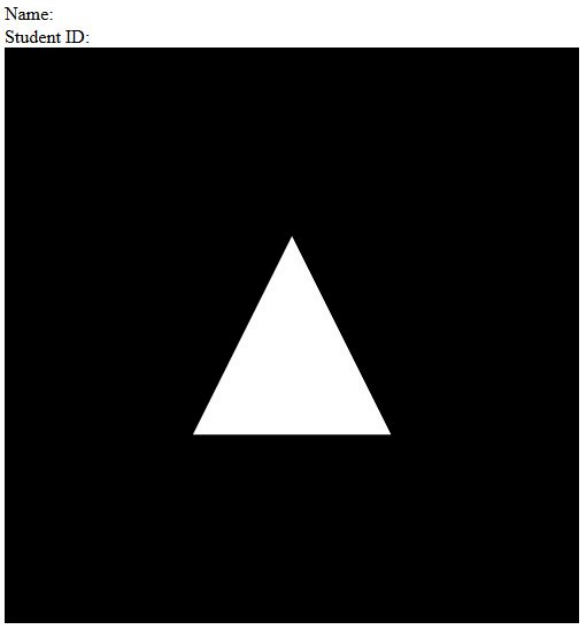
(1) Download pa1.zip, and extract it to any folder you like. You should have four files in the folder:

pa0.html	: html file which shows the WebGL canvas, you don’t need to change this file
pa0.js	: the actual functions for this assignment
gl-matrix-min.js	: utility functions for operating matrices
webgl-utils.js	: utility functions for WebGL animation

The only file you will need to change for finishing the assignment is pa0.js.
(2) Install Chrome. Most modern browsers support WebGL, but we recommend to use Chrome as it provides better compatibility and better debugging tools. After installing Chrome, to check if it supports WebGL, you may visit https://get.webgl.org/ . If your webpage is shown like below, you are all set with WebGL developing environment.



(3) Open pa0.html in the extracted folder with Chrome. In the given code skeleton, a triangle is drawn, as below:

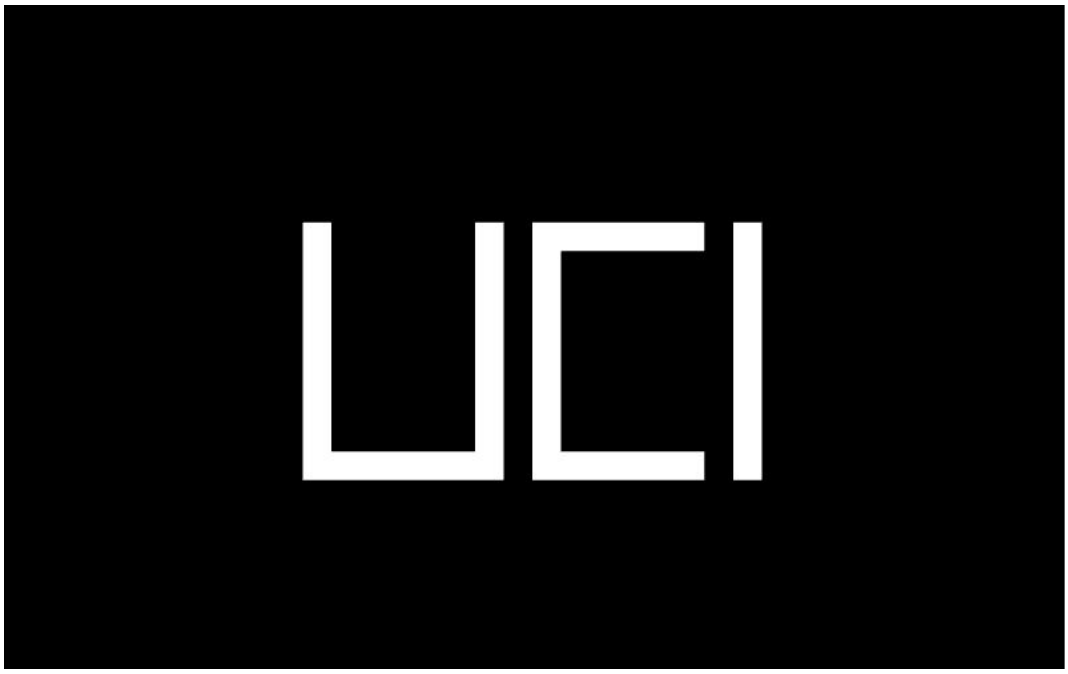


(4) Open pa0.js with a text Editor. Brackets(http://brackets.io/) or Sublime Text(www.sublimetext.com) are good choices, but if you prefer other editors, they should work too. Then you will see a lot of code in this file, and those code describes how the triangle above is drawn. For now, you don't have to understand every line of code in the file, they will be introduced later along with the course progressing.

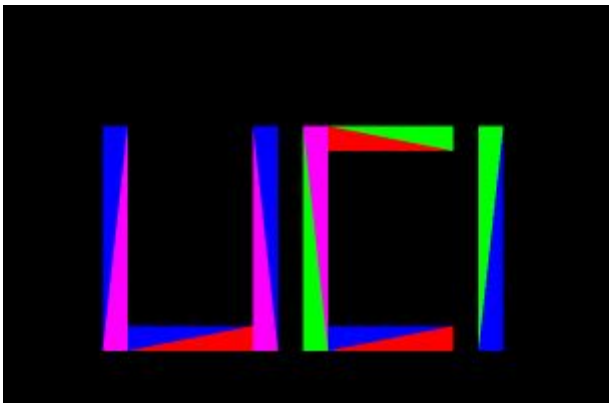
```
var gl;
var canvas;
var shaderProgram;
var vertexPositionBuffer;

// Create a place to store vertex colors
var vertexColorBuffer;

var mvMatrix = mat4.create();

function setMatrixUniforms() {
  gl.uniformMatrix4fv(shaderProgram.mvMatrixUniform, false, mvMatrix);
}

// Converts degrees to radians.
function degToRad(degrees) {
  return degrees * Math.PI / 180;
}

function createGLContext(canvas) {
  var names = ["webgl", "experimental-webgl"];
  var context = null;
  for (var i=0; i < names.length; i++) {
    try {
      context = canvas.getContext(names[i]);
    } catch(e) {}
    if (context) {
      break;
    }
  }
  if (context) {
    context.viewportWidth = canvas.width;
    context.viewportHeight = canvas.height;
  } else {
    alert("Failed to create WebGL context!");
  }
  return context;
}
```

2. Modeling.

You will need to model the letters U,C,I with 2-D meshes of triangles. The expected result is, for example, as below. Your result is not required to be exactly the same as below, it is Okay as long as the letters are recognizable as U, C, and I.



(1) Determine the coordinates of vertices. If you don't know what triangles the meshes should have, one approach for creating the mesh is to get some scratch paper, draw the letters, and figure out a set of coordinates for the vertices and edges. Below is an example of what the triangles might be (each triangle is colored differently, you don't have to color them in this way in your submitted assignment) .

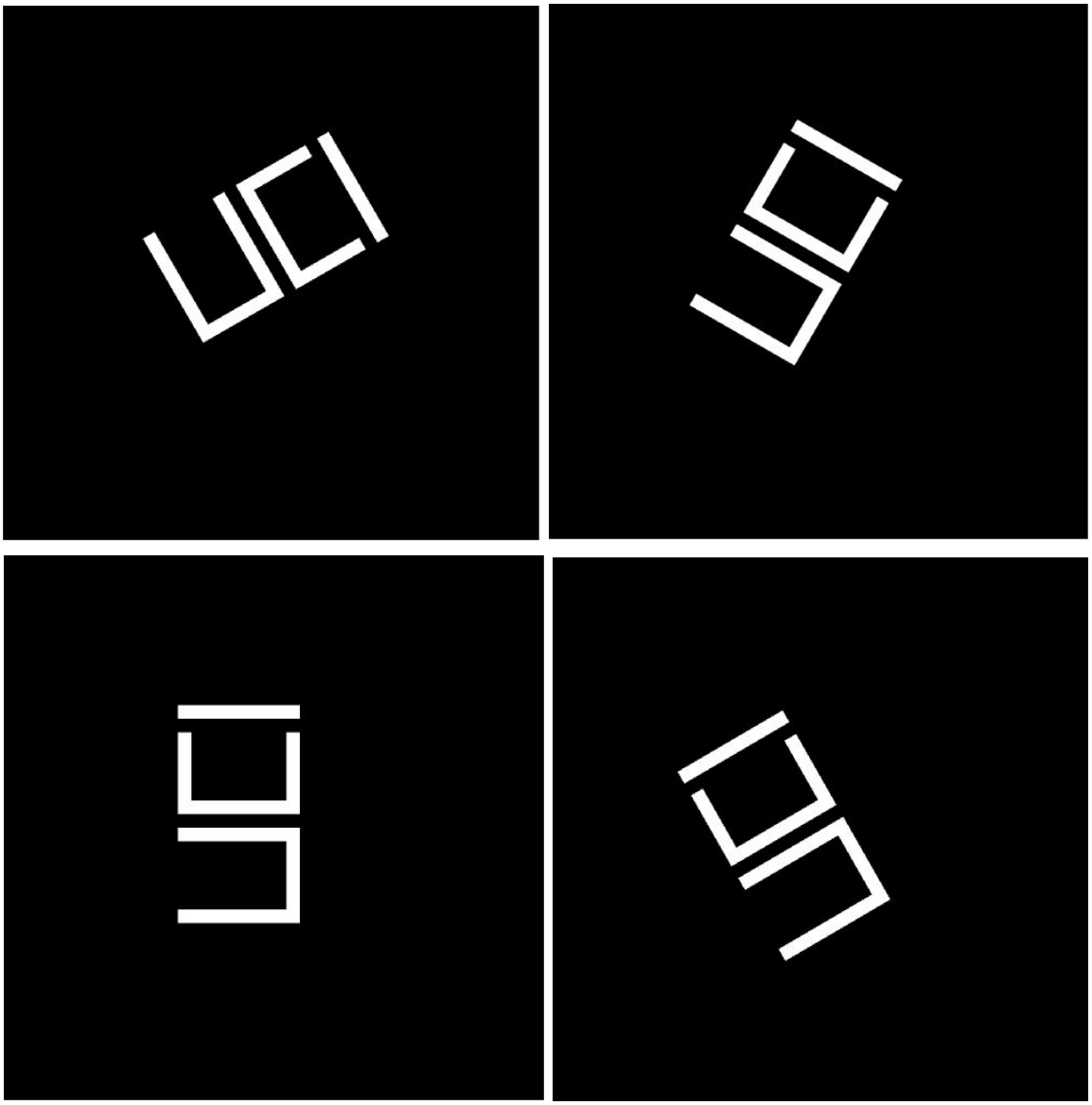


(2) Modify the code of methods *setupBuffers*() and *draw*() in pa0.js
Modify the *vertices* array to make it store the coordinates you have just figured out above.
Note that if the letters you draw are out of the bounds of canvas, you may use *scale* or *translate* function to transform them and keep them inside the scope.

```
mat4.scale(mvMatrix, mvMatrix, [sx, sy, sz]); // sx means scale x axis using sx, similarly for sy and sz
mat4.translate(mvMatrix, mvMatrix, [tx, ty, tz]); // tx means translate x axis for tx, similarly for ty and tz
```

3. Animation.

You will need to animate the letters by spinning them as shown below.



(1) Rotate the letters. The *draw()* method is called regularly with fixed time interval, so you just need to declare a global variable which stores the *rotated degrees*, and inside draw() method, increase the value by fixed interval (every time *draw()* is called, the value is changed). And then you need to rotate the letters using *mat4.rotate()* function to rotate the entire scene by *rotated degrees* along z axis.
(2) Keep the letters inside the canvas, and make the letters rotate along the center. **You will lose partial credit if the letters are out of canvas during spinning.** To make the letters spin around the center, you might need to use mat4.translate to move the center of letters to origin (0, 0, 0), and to make the letters fit into the canvas, you might need mat4.scale to rescale the letters into proper size.
Functions you might use:

```
mat4.scale(mvMatrix, mvMatrix, [sx, sy, sz]); // sx means scale x axis using sx, similarly for sy and sz
mat4.translate(mvMatrix, mvMatrix, [tx, ty, tz]); // tx means translate x axis for tx, similarly for ty and tz
mat4.rotateZ(mvMatrix, mvMatrix, angle_in_radian); // angle_in radian is the rotated angle in radian, a
                                                   // utility function to convert degrees into radians is
                                                   // provided in the code skeleton, called degToRad().
```

For more details, please read the comments starting with TODO in the code skeleton.

4. Submission.

(1) Make sure your name and ID is filled above the canvas, in pa0. html:
(2) You will need to submit the following files on EEE in **a zip archive**, please DO NOT submit individual files.

pa0.html
pa0.js
glMatrix.min.js
webgl-utils.js

5. Grading.

(1) Modeling: Your program should render letters U, C, and I, as described above (70%)
(2) Animation: Your program should animate the letters as described. (30%)

6. Useful references.

(1) WebGL tutorial:
http://learningwebgl.com/blog/?page_id=1217
(2) JavaScript
https://developer.mozilla.org/en-US/docs/Web/JavaScript
(3) JS style guide:
https://google.github.io/styleguide/javascriptguide.xml?showone=Comments#Comments