CMPUT 411/511

Assignment 3

October 1, 2021

Due date: **October 15, 2021 at 23:55** Total marks: 20

Important notes

- A late penalty of 10% per day applies to all late submissions. The maximum number of days late permitted is 1. After 1 day late, your assignment will be given a mark of **ZERO**.
- You are expected to complete the assignment on your own **without** collaboration with others. Discussions at the conceptual level but not at the coding level are permitted.
- You must write your own code. Using codes from the specified text or codes given in this course is permitted. Using any other codes is not permitted.
- Looking at other's codes or letting others to see your codes is not permitted.
- If your codes are used by another student, under the Code of Student Behaviour, you will be dealt with as participating in a plagiarism and cheating offence.
- If you hire a tutor, the tutor **cannot** give you step-by-step or code level instructions to solve the problem. This is defined as cheating.
- You are **not** permitted to upload this assignment to an online site to solicit solutions or to post your solutions to an online site for others to see. Violating this policy will result in severe penalty. These are defined as plagiarism and participation in a plagiarism offence, respectively.
- All your codes will be analyzed for potential plagiarism using MOSS.
- Do not wait until the last minute to work on your assignment. Start as soon as possible.
- Debugging programs and understanding of the materials take time. Debugging is part of the learning experience. You **must** not ask the TA to help you debug your programs.
- During lab, the TA is instructed to give directions but **not** solutions.
- For the programming part,
  - Your code should have sufficient comments to make it readable.
  - Include a README file to document features or bugs, if any, of your program.
- If there are questions, please ask the TA in the lab or via email.

Marking scheme:

- Full assigned marks - no observable bug
- Portions of the assigned marks - observable bugs. The TA has the discretion of assigning a fraction of the assigned marks depending on the number and the severity of the bugs.
- 0 marks - does not work at all

It is important that you read through the whole assignment first before starting. The difficulty of the assignment depends on your ability to design the solution properly. The goal of the assignment is to help you understand the use of transformation in WebGL by implementing the folding of a cube as shown below (Fig 1.):

Fig. 1. Illustration of the folding and unfolding of a cube.

Special notes

- In this assignment, use the following shaders:

```
// Vertex shader program
var VSHADER_SOURCE = `
  attribute vec4 a_Position; // attribute variable
  attribute vec4 a_Color;    // attribute color
  uniform mat4 u_MvpMatrix;
  varying vec4 color;
  void main() {
    gl_Position = u_MvpMatrix*a_Position;
    color = a_Color;
  }
`;
// Fragment shader program
var FSHADER_SOURCE = `
  precision mediump float;// This is required
  varying vec4 color;
  void main() {
    gl_FragColor =  color;
  }
`;
```

- Use gl.getAttribLocation to get the location of the attribute variable and gl.getUniformLocation to get the location of the uniform variable and then use appropriate functions to set them
- To enable interactive manipulations, i.e. to fold and unfold the cube, you need to add a mouse event handler
- Read Chapter 7 and Chapter 9 of text: learn how to use the following functions:
  - `setLookAt`, `setPerspective`, and different transformation operations
  - `initArrayBufferForLaterUse`
  - Creating an object in Javascript and use it to return multiple buffer objects

1. (1 mark) Implement a `Square` object with a constructor and a `draw` method. **Note, in**

your `Square` constructor, it must allow the user to specify the location of the square, its colour, and in what way it can rotate, etc. You may need to add more properties as you work on this assignment. A `Square` object also needs a `draw` method to draw the square. Without `draw`, no one can see it. See below for an example.

2. (1 mark) Use the constructor to create another square of a different colour. See below

3. (1 mark) Modify the `draw` method such that you can bend, i.e. rotate, the red square when you move the mouse, as shown below

4. (1 mark) Use the constructor and create another square of a different color of your choice. See below

5. (1 mark) Modify (if needed) the `draw` method such that you can bend both the red and the green squares

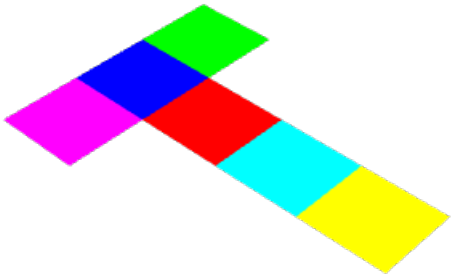
6. (1 mark) Create another square of a different color of your choice. See below

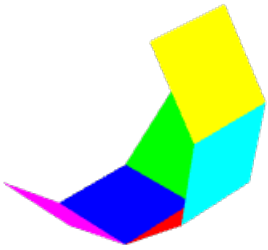
7. (1 mark) Modify (if needed) the `draw` method such that you can bend the red, the purple and the green squares when you move the mouse. See below

8. (1 mark) Create another square of a different color of your choice and extend the string of squares along, say, the x direction (the other direction on the plane is z). See below

9. (2 marks) Modify the `draw` method such that you can bend all the squares except the blue one when you move the mouse. See below

10. (5 marks) Complete the cube by adding another square with color of your choice and modify the `draw` method such that when you move the mouse, all the squares will move. See below





Note: The user must be able to control the folding and unfolding of the cube continuously using the mouse (see the animation provided). Failing to provide this feature will have a penalty of 5 marks.

11. (5 marks) Modify your implementation in the `Cube` constructor such that the squares are laid out as shown on the left of Fig.2. The other figures in Fig. 2 show the folding in progress when the squares are laid out along a different direction.

Fig. 2. Illustration of the folding and unfolding of a cube.