GAME2012 - 3D Graphics Programming
Assignment 3

Instructors: Alex Richard (arichard6@georgebrown.ca)
Hooman Salamat (Hooman.Salamat@georgebrown.ca)

Due: See D2L for due date

1 Introduction

In this assignment you need to finally make 3D objects look much better by adding a texture.

2 Logistics

This assignment is to be completed individually. All submissions for this assignment are electronic via D2L. This assignment is worth 10% of the course grade. You must use Hooman's template as normal. You cannot use any sprites in Alex's or Hooman's examples, including the die texture that Hooman has in one of his examples.

3 Instructions

There are two main parts to this assignment. In the first part, you will need to setup the project. In the second part, you need to render a couple of cubes. And animate the cubes along with the camera!

4 Deliverables

Submit your **entire project** zipped through the Assignment link on D2L, make sure that your project includes the following:

a. The sources file: .cpp/.h files (including those that load shaders)
b. Your shader files
c. Name it: GAME2012_A3_YourlastnameYourfirstname.zip

You will also need to submit **8 screenshots** of your running application. You can submit them separately over D2L or include them in a folder in your zip.

5 What you need to do

Part 1 (0 marks)

1. Use any project as a starting point. You will probably get tips on which to use. You also need to find your own dice texture that has 6 unique sides.

2. Change the title of the window to: GAME2012_A3_LastnameFirstname

3. Change the main .cpp to the following format: GAME2012_A3_LastnameFirstname.cpp

4. Add the following comment header in your file and make sure to change my name to your name and add your ID. Also, make sure to add some useful description to your file.

```
//*********************************************************************
// GAME2012_A3_LastFirst.cpp by Your Name (C) 2020 All Rights Reserved.
//
// Assignment 3 submission.
//
// Description:
//   Click run to see the results.
//
*********************************************************************
```

5. Customize the project according to your taste:
    a. Change the background color from black to a different color.
    b. Change the window size to a different size.
    c. Change the color of the points/lines.

6. If you fail to do any of the above steps, you will be penalized 1 mark.

*Part 2 (**10 marks**)*

2. You need to setup your camera into Perspective mode.
    a. Set the starting location of the camera somewhere on the Y-Z plane.

b. The camera should be looking at the origin point (0,0,0).
c. If you fail to set your camera properly, you will be penalized 1 mark.

3. You need to create vertices and connect them in such a way that renders a cube. The cube size is going to be up to you. Your cube should have 6 faces - as cubes normally do. Apply the following texture to each face, you can download the texture from blackboard.
   a. Create a VAO to describe your cube
   b. Create a VBO for your cube vertex positions
   c. Create another VBO for the vertex texture UV map
   d. Bind all VBOs to a single VAO
   e. If you do not have just one VAO with all VBOs bound to it, you will be penalized 1 mark.
   f. Translate your cube to place it at (0,0,0)
   g. Render your cube without the texture applied and **take the first screenshot! (1 mark)**

4. Now it's time for the texturing.
   a. Load your texture with stb_image **(1 mark)**
   b. Generate a texture object and bind it to one of the IDs **(1 mark)**
      i. These are lines of code using *textureID*
   c. Load your texture and assign it to the texture object and link it to the shader **(1 mark)**
      i. This is the *glTexImage2D* line, texture parameters, and linking the texture to the uniform in the shader

5. You need to fix the mapping so that each face of the cube gets one part of the texture. Resulting in a white, six-sided die.
   a. You must have mapping correct for all six sides **(3 marks / 0.5 per each correct side)**
      i. It doesn't have to be dead on, so long as I can see each die digit individually
   b. **Take 6 screenshots, one showing EACH side**

6. You need to render a second die using the same VAO you created for the first cube and animate both cubes. **(1 mark)**
   a. Translate the first cube to place it at (-3,0,0)
   b. Translate the second cube to place it at (3,0,0)
   c. Make sure both are visible, adjusting if needed **(1 mark)**
   d. Rotate each cube constantly in opposite directions **(1 mark)**
   e. **Take a final screenshot**

6 Penalties

- You don’t name your .cpp and .zip properly: 1 mark off each
- Late penalty: -10% per day up to -50% and then not accepted
- Missing screenshots: ½ mark off for each missing screenshot
- Handing in code from an example without significant changes: 0% and a plagiarism report. Adding a few changes to comments or title text is not significant.
- Old OpenGL used: If you used the fixed function pipeline (legacy OpenGL) with functions like glBegin() and glEnd() to draw, you get 0. We teach modern OpenGL in this course and if you’re using FFP, then I know you’re plagiarizing from another source.
- You don’t use Hooman’s template: 0% for assignment