Concordia University
Department of Computer Science and Software Engineering

COMP371: Computer Graphics
TENTATIVE PROJECT DESCRIPTION
(Revision 1.0)

COMP371		**WINTER 2021**
Instructor (WW):	Serguei A. Mokhov	mokhov@encs.concordia.ca
Issued:		January 28, 2021
PA1 due:		February 10, 2021, 23:59

Revisions

Revision 1.0:	January 28, 2021

1 Objectives and Overview

We learn practical aspects of OpenGL programming through a series of programming assignments/project that will be elaborated here.

1.1 Resources

- Project template provided by the instructors.
- `opengl-tutorial.org`
- `learnopengl.com`
- SIGGRAPH Course: `https://dl.acm.org/doi/10.1145/2037636.2037643`
- ...

1.2 Groups

The project is to be done in groups of 5 (all parts). Each team is provided with a group account to coordinate their group activities and manage the project's content. to share documents, code and resources alike related to the group's project.

1.3 Procedure

Teams sequentially complete each part of the project as "programming assignments" 1–3 that constitute deliverables followed by a complete project and a report. Parts 1–3 are to be demoed to the lab instructors during the week after they are due. Whatever you demo, it

must be identical to whatever is submitted to the EAS by the deadline (usually a midnight of the day preceding the demo week or day, see schedule for more details). You (with the lab instructor present) or the lab instructor download the submitted code from the EAS and demo – demo need not be done the same lab days after the due date and can be done earlier if you are ready. **All team members should normally be present during the demo and be prepared to answer any related questions that come up.**

2 Deliverables

All parts are to be done in a group. Each team member would be responsible for providing one or more models, components, etc. as subdivided by the team, and the team for the overall game environment.

1. 10% Programming Assignment 1 (PA1): Modeling and Camera Control
2. 20% Programming Assignment 2 (PA2): Lighting and Texture mapping
3. 30% Programming Assignment 3 (PA3): Final: animation, demo, report with the software design and user manual

In general, better quality works get better grades. The PA1–PA3 demos are evaluated by the implemented functionality in accordance with the requirements as well as the quality of the source code and modeling (procedural modeling, modular design of the code, well documented non-trivial code pieces in the form of comments, etc.), and a proper README on deployment of your project (compiling and running your project).

2.1 Part 1 (PA1): Modeling and Camera Control.

This programming assignment will introduce OpenGL programming. More specifically, you will learn how to create simple virtual scenes consisting of various objects. How to display them as wireframe meshes by drawing triangles, and furthermore how to set up, and manipulate a virtual camera to view the scene from varied angles and distances.

A mesh is a set of polygons that share vertices and edges, which describe the shape of a geometric object. In this assignment you have to first create a unit cube in 3D at the origin, then using appropriate modeling transformations create first and last letters of your first name and first and last digits of your student ID (see Figure 1 – an illustration of a sample half of the output).

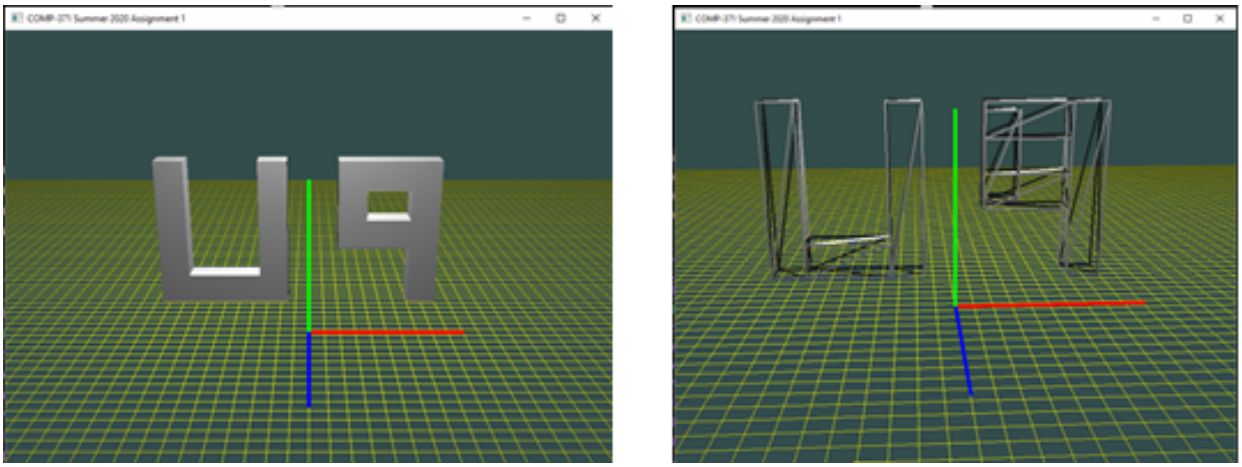

Figure 1: Example a letter and a digit on a grid.

2.1.1 Implementation Specifications:

Develop an OpenGL application with the following functionality and features that:

- Creates a 128x128 square grid (ground surface) in the XZ plane centered at the origin.
- Creates a set of three lines 7 grid units in length, in 3 different colors, representing each coordinate axis in virtual world space, centered at the origin.
- Creates an letters-and-digits model (4 units: 2 letters and 2 digits) as described earlier and similar to the one depicted in the figure by suitably transforming a unit cube. There must be one such model by each of the team members based on their name and student ID.
- Four models are to be placed on quarter arcs of a circle touching the imaginary edges of the grid as if they are on a circle embedded within the grid. The fifth model is initially positioned at the center of the grid facing along the X axis; also arched like the others. You have to find out what the arching angle be for them experimentally.

- Makes it such that the model should consist of independent parts (units) so they can be rotated/moved on their own when it will be required. We also recommend that you **use hierarchical modelling**, so that a single transformation applied to a model's origin will apply it to a complete model.
- Places a virtual camera with the world space origin as the point of focus for display and animation:
  - Create a GLFW window of size 1024x768 with double buffering support.
  - Render the coordinate axis, ground and all the models in the window.
  - The application should use a perspective view to display all the objects and enable hidden surface removal.
- The application should handle the following input:
  - The user can incrementally size up the model by pressing 'U' to scale-up and 'J' to scale-down. Each key press should result in a small size change (small step).
  - The user can control the model position and orientation using keyboard input, i.e., 'A' $\rightarrow$ to move "left", 'D' $\rightarrow$ to move "right", 'W' $\rightarrow$ to move "up", 'S' $\rightarrow$ to move "down", 'a' $\rightarrow$ to rotate "left" 5 degrees about Y axis, 'd' $\rightarrow$ rotate "right" 5 degrees about Y axis. (You may add other rotations about other axes, if you want. Document if do.)
    - You may define keys '1' to '5' to allow you to select which model to manipulate.
    - You may predefine 5 cameras as well accordingly with the default position in front of each model, or use a singe FPS-like camera to move around to each model under your control.
  - The world orientation is also changed by using keyboard input, i.e., left arrow $\rightarrow$ Rx, right arrow $\rightarrow$ R-x, up arrow $\rightarrow$ Ry, down arrow $\rightarrow$ R-y. (Rx denotes a small anti-clockwise rotation about positive x axis, R-x about negative x axis, etc.) Pressing 'Home' button should reset to the initial world position and orientation.
  - The user can change rendering mode for the model, i.e., points, lines, triangles based on keyboard input, namely, key 'P' for points, key 'L' for lines, key 'T' for triangles.
  - The user can pan and tilt the camera as follows:
    - While the right button is pressed $\rightarrow$ use mouse movement in the x direction to pan; and
    - While the middle button is pressed $\rightarrow$ use mouse movement in the y direction to tilt.
  - The user can zoom in and out of the scene – while left button is pressed $\rightarrow$ use mouse movement to move into/out of the scene.

- The application must use OpenGL 3.1 and onwards and must include brief comments explaining each step in a form of README.md.

2.1.2 Submission:

Assignment must be submitted only through EAS. No other form of submission will be considered. Please create a zip file containing your C/C++ code, files required to build your code (cmake, VS Studio, XCode, etc.), vertex shader, fragment shader, a READNE file (.md). **Please do NOT submit Debug, build, or .vs folders alike**. The zip file should be named `Assignment#_YourTeamID`. In the README file document, the features and functionality of the application, and anything else you want the grader to know, i.e., control keys, keyboard/mouse shortcuts, etc.

2.1.3 Additional Information

- You can use the skeleton code provided during the lab sessions to get started.

2.1.4 Evaluation Procedure

You MUST demonstrate your program to the lab instructor during a pre-scheduled zoom session. All the team members must be present during the chosen time slot.

You must run your submitted code, demonstrate its full functionality and answer questions about the OpenGL programming aspects of your solution. Major marking is done on the spot during the demonstration. Your code will be further checked for structure, non-plagiarism, etc. However, ONLY demonstrated submissions will receive marks. Other submissions will not be marked.

2.2 Part 2 (PA2): Lighting and Texture Mapping.

To follow...

2.3 Part 3 (PA3): Final Project

To follow...

References

[1] Edward Angel and Dave Shreiner. Introduction to modern OpenGL programming. In *ACM SIGGRAPH 2011 Courses*, SIGGRAPH'11, pages 7:1–7:136, New York, NY, USA, 2011. ACM.