UNIVERSITÉ Concordia UNIVERSITY

ACADEMIC YEAR: 2020-2021

ASSIGNMENT 1

Assignment Posted: July 07, 2021

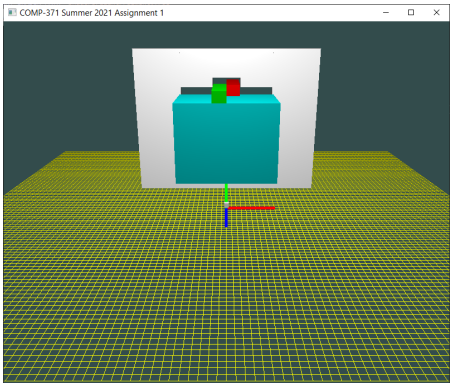
Assignment Due: July 16, 2021 before 11.59pm

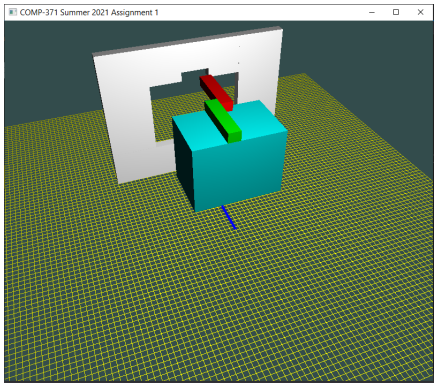
Final Deadline with 20% flat Penalty: July 19, 2021 before 11.59pm

Description:

This programming assignment will introduce OpenGL programming. More specifically, you will learn how to create simple virtual scenes consisting of 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 a pair of 3D object and a wall with matching hole. (We will be working towards the superhypercube game. The gameplay video is available at https://www.youtube.com/watch?v=54bpnxLlmZQ). Each member of your team must create a complex object comprising of at least 9 cubes and a wall with matching hole. A simple combination is depicted below in the image (for reference only).





Implementation Specifications:

Develop an OpenGL application with the following functionality and features:

- Creates a 100x100 square grid (ground surface) in the XZ plane centered at the origin.
- Creates a set of three lines 5 grid units in length, in 3 different colors, representing each coordinate axis in virtual world space, centered at the origin.
- Creates a complex object and a wall with matching hole as described earlier and like the one depicted in the figure by suitably transforming a unit cube. There must be one model

by each of the team members. Please make sure that the objects must be significantly complex and different for each team member.

- Four models are to be placed in the four corners of the grid, and the fifth model is initially positioned at the center of the grid facing along X axis.
- The model should consist of independent parts so they can be rotated/moved on their own. We recommend that you use hierarchical modelling, so that a single transformation applied to a model's origin will apply it to the 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 following input for each model which should be selected by pressing a key from 1 to 5:
  - 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.
  - The user can control the model position and orientation using keyboard input i.e. A → move left, D → move right, W → move up, S → move down, a → rotate left 5 degrees about Y axis, d → rotate right 5 degrees about Y axis. You may add other rotations about other axis, if you want.
  - The world orientation is changed by using keyboard input *i.e.* left arrow → Rx, right arrow → R-x, up arrow → Ry, down arrow → 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 right button is pressed → use mouse movement in x direction to pan; and
    - While middle button is pressed → use mouse movement in y direction to tilt.
  - The user can zoom in and out of the scene using just the camera perspective - while left button is pressed → 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.

Submission:

Assignment must be submitted only through Moodle. No other form of submission will be considered. Please create a zip file containing your C/C++ code, vertex shader, fragment shader, a readme file (.txt). 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.*

Additional Information

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

Bonus Features:

You can achieve an extra 20 points as bonus if you provide a shuffle feature to your object. By pressing a key, cubes from your object should move around changing the shape of your object with a constraint that the new object should still be able to pass through the wall.

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 timeslot. 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.

Grading Rubric for Assignment 1

Total	**100 pts**
Environment	**10 pts**
Drawing the Grid	5 pts
Drawing the three axes	5 pts
Model	**30 pts**
Object	10 pts
Wall	10 pts
Hierarchical Modeling	10 pts
Interaction	**40 pts**
Repositioning the Model (Spacebar)	5 pts
Sizing the Model up and down	5 pts
Update Model position (WSAD)	5 pts
Model rotation around Y-axis (a and d)	5 pts
Changing world orientation (arrows and HOME)	5 pts
Rendering modes (P, L, T)	5 pts
Camera tilt and pan	5 pts
Camera zoom (in and out)	5 pts
Demo QnA by the Grader	**20 pts**
Bonus Points	**20 pts**