# *CSCI 3260 Principles of Computer Graphics*

## Assignment One: Creating an interactive 3D Scene

Due Time: 11:59 pm, Oct 03, 2021 (Sunday)

*Late penalty: 10 points per day.*

*Fail the course if you copy*

## I. Introduction

This first programming assignment will introduce you to the OpenGL graphics programming interface and programmable pipeline. In this programming assignment, you need to create a **3D scene** with user **interaction** (see the good examples in Fig. 1). This assignment aims to apply your understanding of the computer graphics concepts into practice, get familiar with the OpenGL programming library, and introduce you to the programmable pipeline.

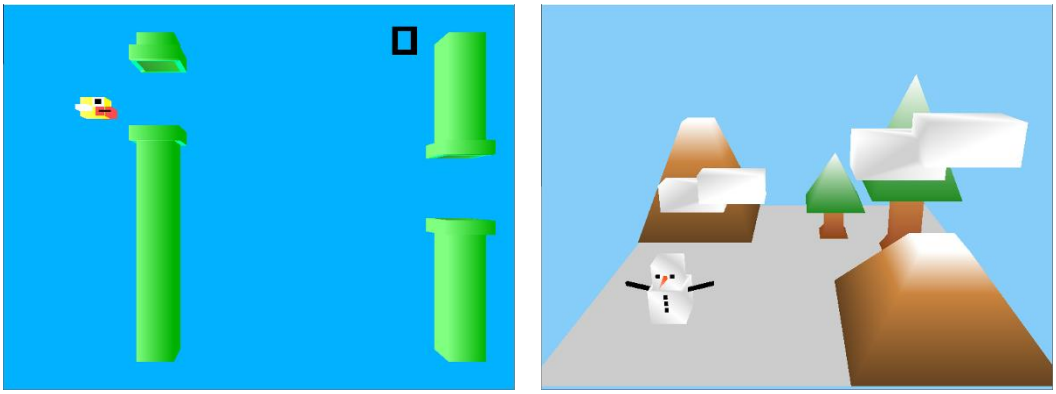

Fig. 1 Good examples of 3D scene.

Your goal is to design a 3D scene with user interaction. Specifically, in your scene, there must be **2D objects (e.g., plane)**, **3D objects (e.g., cube)**, and/or **lines (points)** (see Fig. 2 as an example), and you should be able to apply transformations including **translation**, **rotation**, and **scaling** to them. The user should be able to use the keyboard (and/or the mouse) to translate, rotate, and scale the object. The object color, window size, window title, and scene layout are all up to you. To make your scene more realistic, you should use the **perspective projection** instead of orthographic projection. You are required to draw objects with **indexing**. Your 3D scene shall not be limited by the demo picture and program.

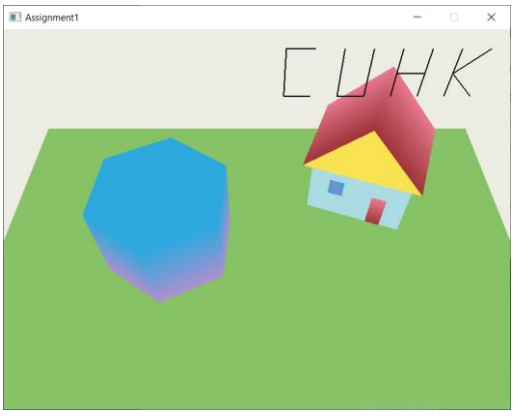


Fig. 2 Basic requirements of assignment 1.

## II. Implementation Details

In the assignment package, we have provided you with two shader programs (i.e., *VertexShaderCode.glsl* & *FragmentShaderCode.glsl*) and a template main program (i.e., *main.cpp*). They include the necessary functions you will use with event processing functions in the **GLFW** interface toolkit. Use this template as the basis for your implementation. You have to design your function to process the keyboard events, and you should also submit a file *readme.txt* to explain the keyboard (and/or mouse) events you designed in your program. Otherwise, the mark for related items will be deducted.

All programs should meet reasonable programming standards: header comment, in-line comments, good modularity, clear printout, and efficiency.

**Basic Requirements:**

1. OpenGL code should use the **programmable** pipeline with OpenGL 3.0+ instead of the fixed pipeline.
2. Draw at least **one** 2D object and **two** 3D objects.
3. Ensure at least one object is drawn with **indexing**.
4. Create at least **three** kinds of keyboard and/or mouse events, such as rotation, translation, and scaling.
5. Use the given **perspective** projection (45.0 degree, any aspect, 0.1, 20.0) to draw the scene.
6. Enable **depth test** to realize occlusion.

**Additional self-design requirements:**

You are free to add objects, move them, organize them, and whatever you wish to make your scene interesting.

## III. Grading Scheme

Your assignment will be graded by the following marking scheme:

| | |
|---|---|
| **Basic (80%)** (e.g., Fig. 2) | |
| Draw 2D objects and 3D objects. | 25% |
| At least one object is drawn with indexing. | 10% |
| At least three kinds of keyboard (and/or mouse) events. | 15% |
| Include three kinds of object transformations (rotation, translation, scaling). | 15% |
| Perspective projection (given). | 10% |
| Depth test. | 5% |
| **Advanced (at most 20%)** (e.g., Fig.1) | |
| Complex and meaningful objects and scenes constructed by different primitives. | 10% |
| Interesting and creative interactions, e.g., scene interaction or view angle change. | 10% |
| Try with different parameters in the perspective projection (e.g., fov, etc.) and discuss the effect. | 5% |
| **Total (maximum):** | **100%** |

**Note: no grade will be given if the program is incomplete or fails compilation or using fixed pipeline.**

## IV. Guidelines to submit programming assignments

1) You can write your programs on Windows and macOS. The **official grading platform** should be Windows with Visual Studio. If we encounter problems when execute/ compile your program, you may have to show your demo to the tutor in person.

2) Modify the provided *main.cpp & VertexShaderCode.glsl & FragmentShaderCode.glsl* and provide all your code in this file. It is **not recommended** to create or use other additional **.cpp** or **.h** files. Type your full name and student ID in *main.cpp*. ***Missing such essential information will lead to mark deduction (up to 10 points).***

3) We only accept OpenGL code written in the programmable pipeline. No points will be given if your solution is written in the fixed pipeline.

4) We only accept OpenGL code implemented with **GLFW** and **GLEW**. No points will be given if you use other windowing and OpenGL extension libraries (unless you have strong enough reasons).

5) You should write a succinct *readme.txt* about what you have done. Otherwise, we may ignore some of your efforts.

6) Zip the source code file (i.e., *main.cpp & VertexShaderCode.glsl & FragmentShaderCode.glsl*), **the executable file** (e.g., *Assignment1.exe*) (if you use Windows for your assignment), and the readme file (i.e., *readme.txt*) in a .zip (see Fig. 3). Name it with your own student id (e.g., *1155012345.zip*).

7) We may check your code. And we may deduct your marks if you have seriously wrong implementation.

8) Submit your assignment via eLearn Blackboard. (https://blackboard.cuhk.edu.hk).

9) Please submit your assignment before 11:59 p.m. of the due date. **Late submission will be penalized by 10 points deduction per day.**

10) In case of multiple submissions, only the latest one will be considered.

11) ***Fail the course if you copy.***

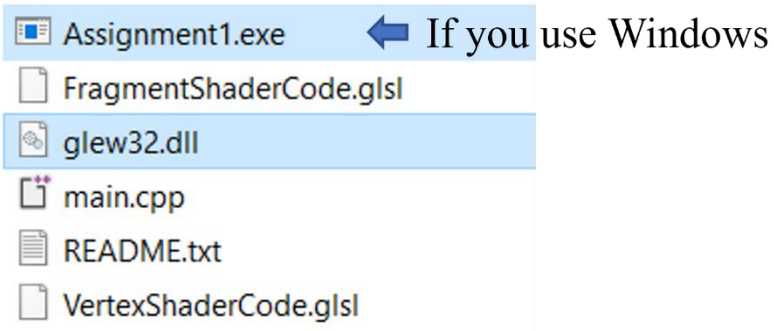


Fig. 3 Files to submit.