Graphics Programming HW 1

Due: 3/25 11:59 PM

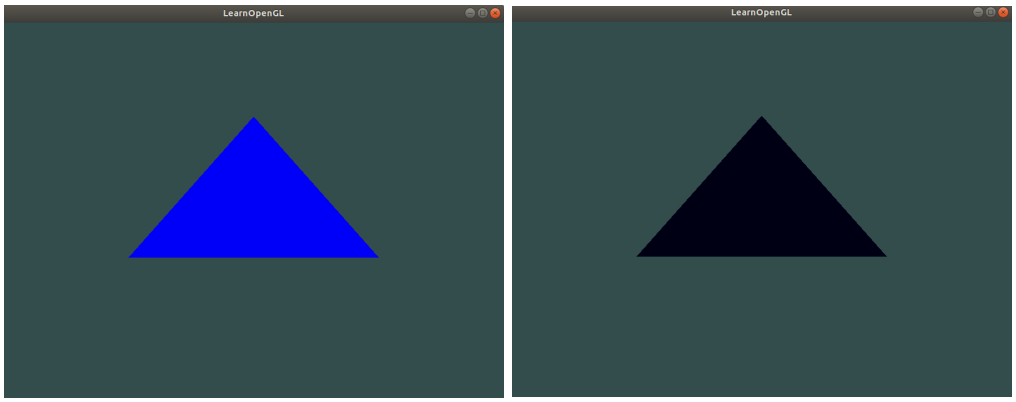
Throughout the all of the homework, it is discouraged to use fixed function pipeline such as $glBegin, glVertex$, which is deprecated from OpenGL 3.0. Please use shader instead.

Prob 1. Getting Started with Triangle

In this problem, you will implement a program drawing a triangle using shader. Fill the blanks in the provided code to satisfy following requirements.

1.1 Flickering blue triangle

Your task here is to make periodically flickering triangle with blue color. The triangle should be flickering which means that the color of the triangle should be changed from black to blue. The triangle should be flickering periodically, but the period doesn't matter. *Hint*: Use these functions: *glfwGetTime*, *glUniformXX*, *glGetUniformLocation*, *getID(in Shader Class)*



(a) (b)

Figure 1: Flickering triangle

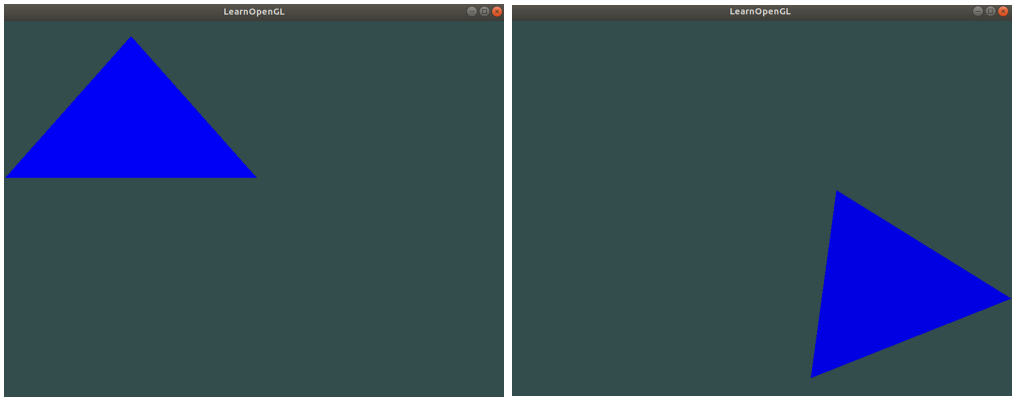
1.2 Moving triangle

This problem follows the above 1.1 problem. Your task here is to make the triangle move in your control. Specifically, the triangle should move in the direction of pressed keys. It should be possible to press several

keys to move the triangle diagonally. There are no constraint of choosing the keys. *Hint*: Use these functions: *glfwGetKey, setXXX (in shader Class)*

1.3 Rotating triangle

This problem follows the above 1.2 problem. Your task here is to add a rotating function to the triangle. Specifically, the triangle should rotate counter-clockwise around the center of gravity when you're pressing the key 'r'. Rotating the triangle also should work with moving it simultaneously.



(a) 1.2 Moving triangle (b) 1.3 Rotating triangle

Figure 2

Prob 2. Simple 2D Game with OpenGL

In this problem, you will make a simple 2D game using OpenGL. Triangles fall from the top of the screen, and player moves the red bar at the bottom to gather them. The picture of the game is on Fig. 3. We provide a skeleton code and you have to implement followings.

- For each time interval, new randomly shaped triangle appears at the top of the screen. It's x position is randomly selected. It falls in $-y$ direction with constant speed, and also rotates with constant angular velocity. You should update each triangle's position and orientation for each frame. Use uniform variable to update it.
- You then have to make a red rectangular bar near the bottom of the screen. Define additional VAO, VBO to render a quad. The bar should be able to be moved left and right by pressing the arrow keys.
- Try to implement simple collision detection. Because it is hard to thoroughly detect intersection between two polygons, you may use simple approximation. Just check whether a center of the polygon is inside the box or not (Fig. 4). Also check if polygon has gone out of the boundary, and remove it.
- Finally, you will make a simple score board. It is a picky task to render texts using OpenGL, so we will just draw triangles at the bottom as much as the score. Initialize the score if it reaches the max score, maybe 20, to prevent drawing too many triangles going over the boundary.

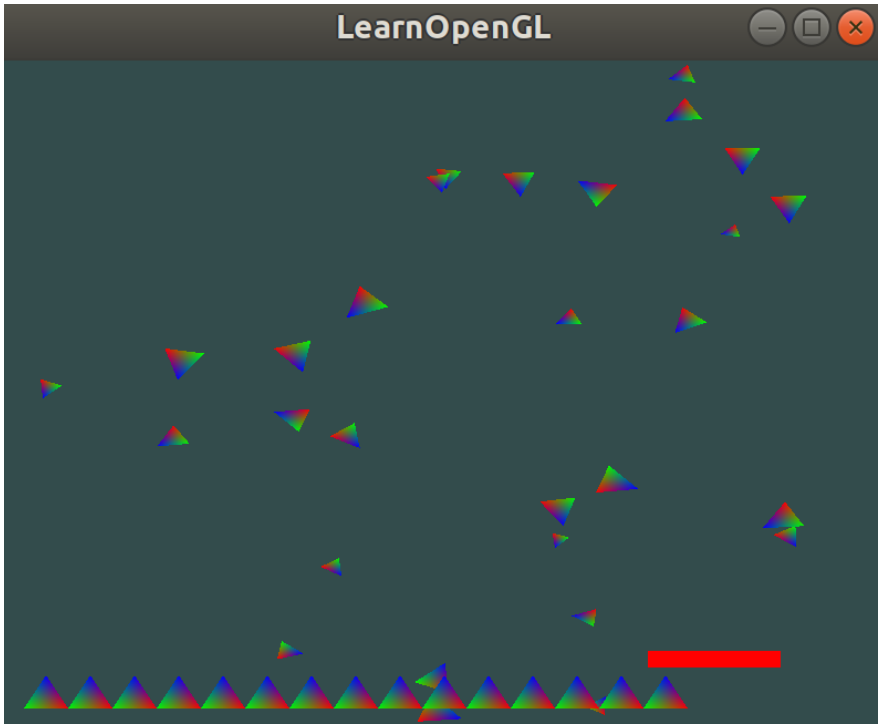


Figure 3: The game you have to implement in Problem 2.

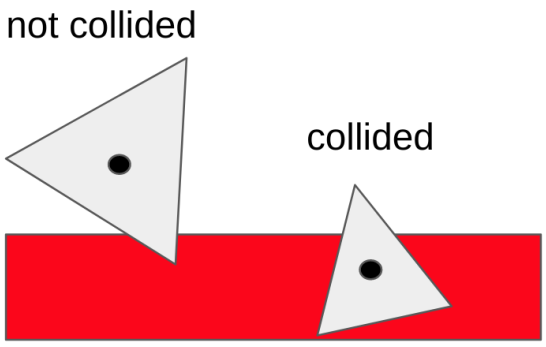


Figure 4: Simple collision detection