COMPUTER GRAPHICS

LAB#1

RAMY AHMED EL SAYED

19015649

PROBLEM STATEMENT

You are required to create OpenGL project using project template. You should implement an application that handle user input at runtime. Input handling should be as follows:

• When user presses '+' button, a new point should be drawn at random location within application window.

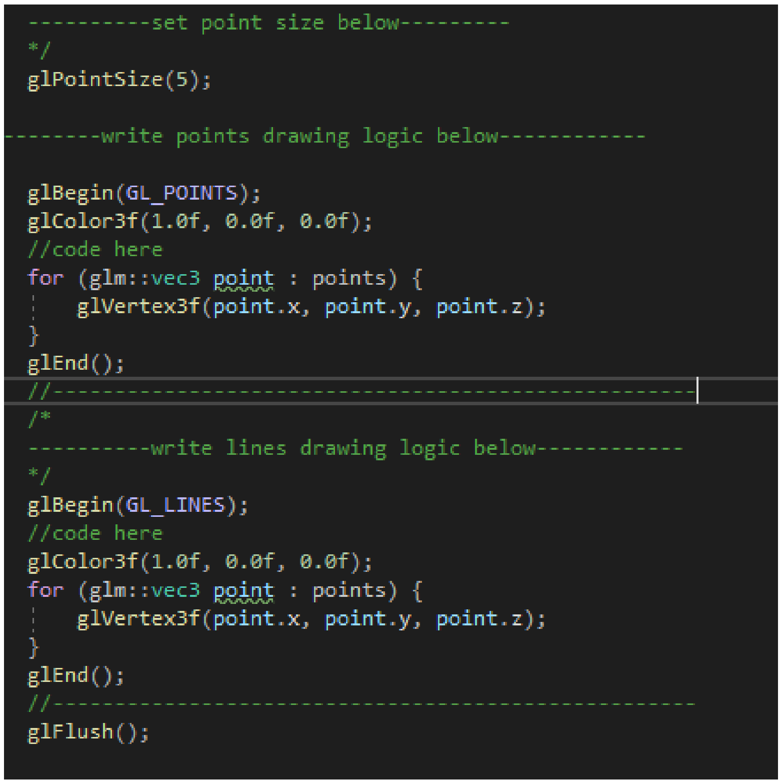
• When user presses '-' button, the last point drawn should be erased.

• For every two successive points, a line should be drawn connecting them. A point is allowed to be part of only one line so that number of lines is the half number of points. (note: if number of points is odd, the last point will not be part of any line until user adds a new point).

Also, set point size to 5units.Use code in this link as starter code.

Code Description

drawScene()



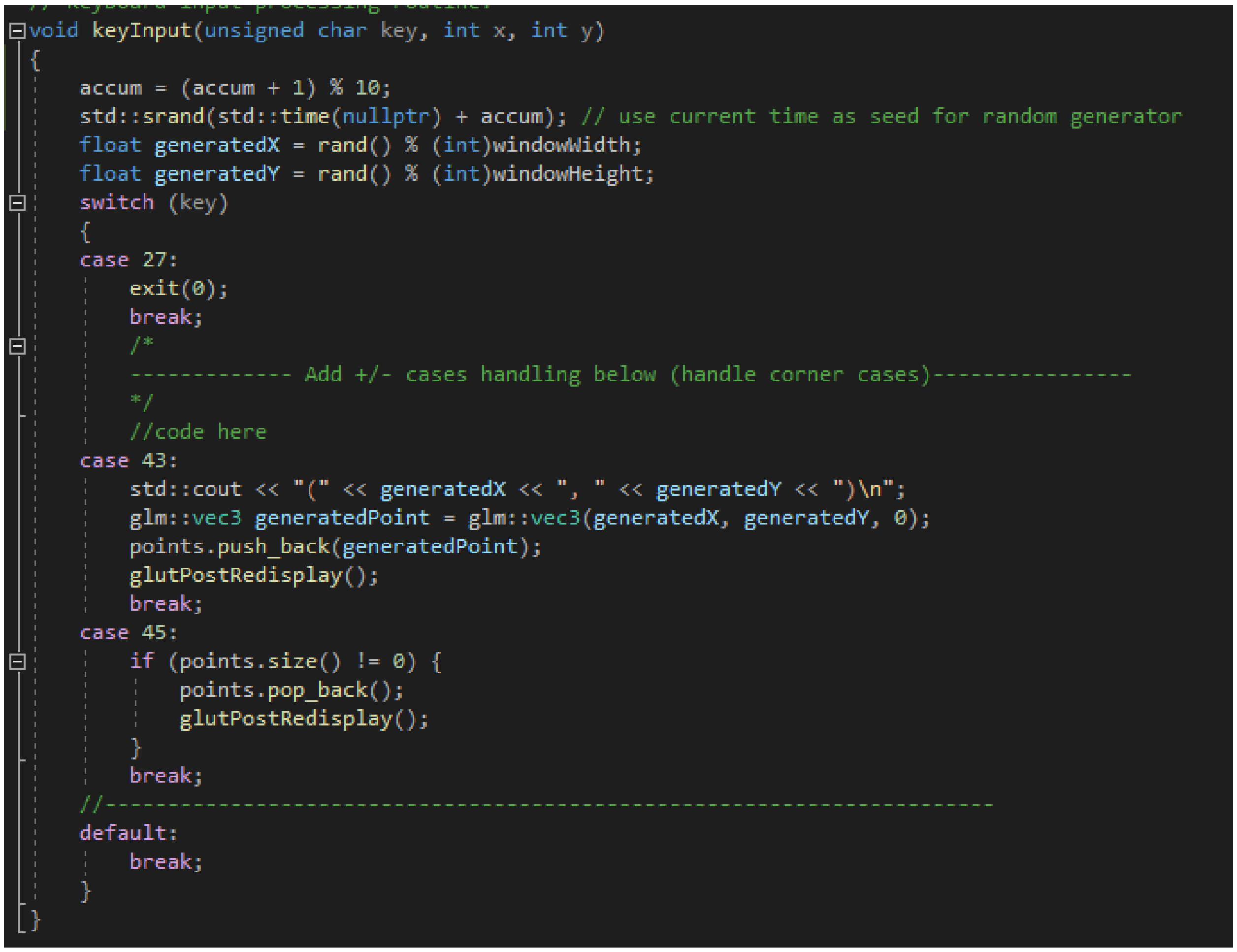
```
 ----------set point size below---------
 */
 glPointSize(5);

--------write points drawing logic below------------

 glBegin(GL_POINTS);
 glColor3f(1.0f, 0.0f, 0.0f);
 //code here
 for (glm::vec3 point : points) {
     glVertex3f(point.x, point.y, point.z);
 }
 glEnd();
 //------------------------------------------------------
 /*
 ----------write lines drawing logic below------------
 */
 glBegin(GL_LINES);
 //code here
 glColor3f(1.0f, 0.0f, 0.0f);
 for (glm::vec3 point : points) {
     glVertex3f(point.x, point.y, point.z);
 }
 glEnd();
 //------------------------------------------------------
 glFlush();
```

In the main update method (drawScene()) we initialize the primitive for a point and a line, and then iterate over the points vector on every update and draw its content.

KEYINPUT()



```
void keyInput(unsigned char key, int x, int y)
{
    accum = (accum + 1) % 10;
    std::srand(std::time(nullptr) + accum); // use current time as seed for random generator
    float generatedX = rand() % (int)windowWidth;
    float generatedY = rand() % (int)windowHeight;
    switch (key)
    {
    case 27:
        exit(0);
        break;
        /*
        ------------- Add +/- cases handling below (handle corner cases)----------------
        */
        //code here
    case 43:
        std::cout << "(" << generatedX << ", " << generatedY << ")\n";
        glm::vec3 generatedPoint = glm::vec3(generatedX, generatedY, 0);
        points.push_back(generatedPoint);
        glutPostRedisplay();
        break;
    case 45:
        if (points.size() != 0) {
            points.pop_back();
            glutPostRedisplay();
        }
        break;
    //------------------------------------------------------------------------
    default:
        break;
    }
}
```

The keyInput() method is divided into 3 parts:

THE RNG INITIALIZER

Initialize a seed that changes with time and define our random X and Y coordinates at which the points will be drawn.

Note: we use an accumulator to avoid generating points with the same coordinates if the input is way too quick (equal seed due to equal time)

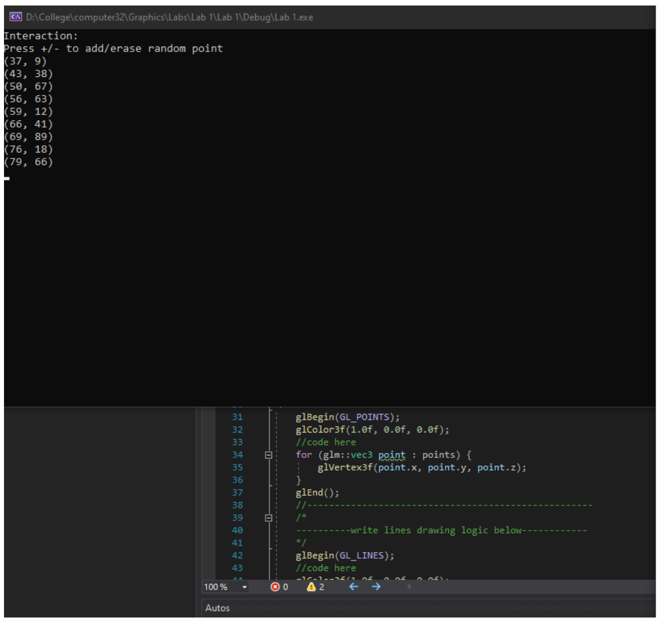
THE “+” BUTTON FUNCTIONALITY

Outputs the coordinates of the new created point and adds it to our points vector and glutPostRedisplay() is always called whenever “+” or “-“ are pressed to update the current screen.

THE “-“ BUTTON FUNCTIONALITY

Simply removes the last added point from our points vector if it isn’t empty.

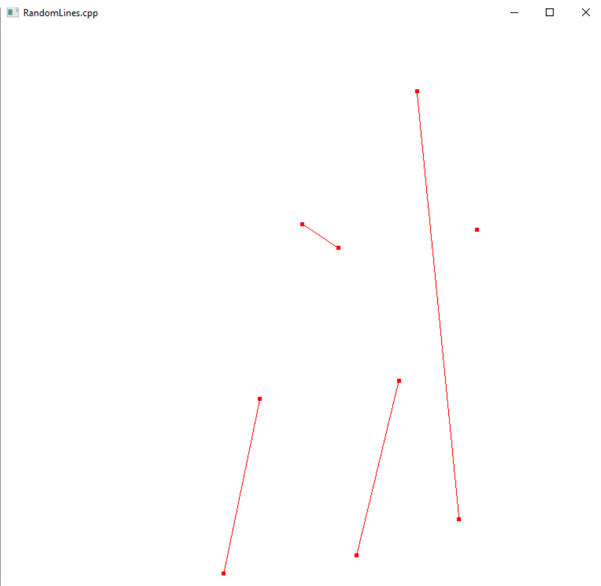
Example Of Running Code



D:\College\computer32\Graphics\Labs\Lab 1\Lab 1\Debug\Lab 1.exe

```
Interaction:
Press +/- to add/erase random point
(37, 9)
(43, 38)
(50, 67)
(56, 63)
(59, 12)
(66, 41)
(69, 89)
(76, 18)
(79, 66)
```

```
glBegin(GL_POINTS);
glColor3f(1.0f, 0.0f, 0.0f);
//code here
for (glm::vec3 point : points) {
    glVertex3f(point.x, point.y, point.z);
}
glEnd();
//--------------------------------------------------
/*
----------write lines drawing logic below------------
*/
glBegin(GL_LINES);
//code here
```



Challenges

The main challenge of the assignment was figuring out where to place the glutPostRedisplay() function, since in other game engines such as Unity or Lua, the scene would automatically get updated with newer input.

Additionally, figuring out a way to avoid randomly generated points to be equal (two points with the same coordinates).