Lab Assignment 01

Name: Abdelaziz Mohamed Abdelaziz.

I.D: 19015941.

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.

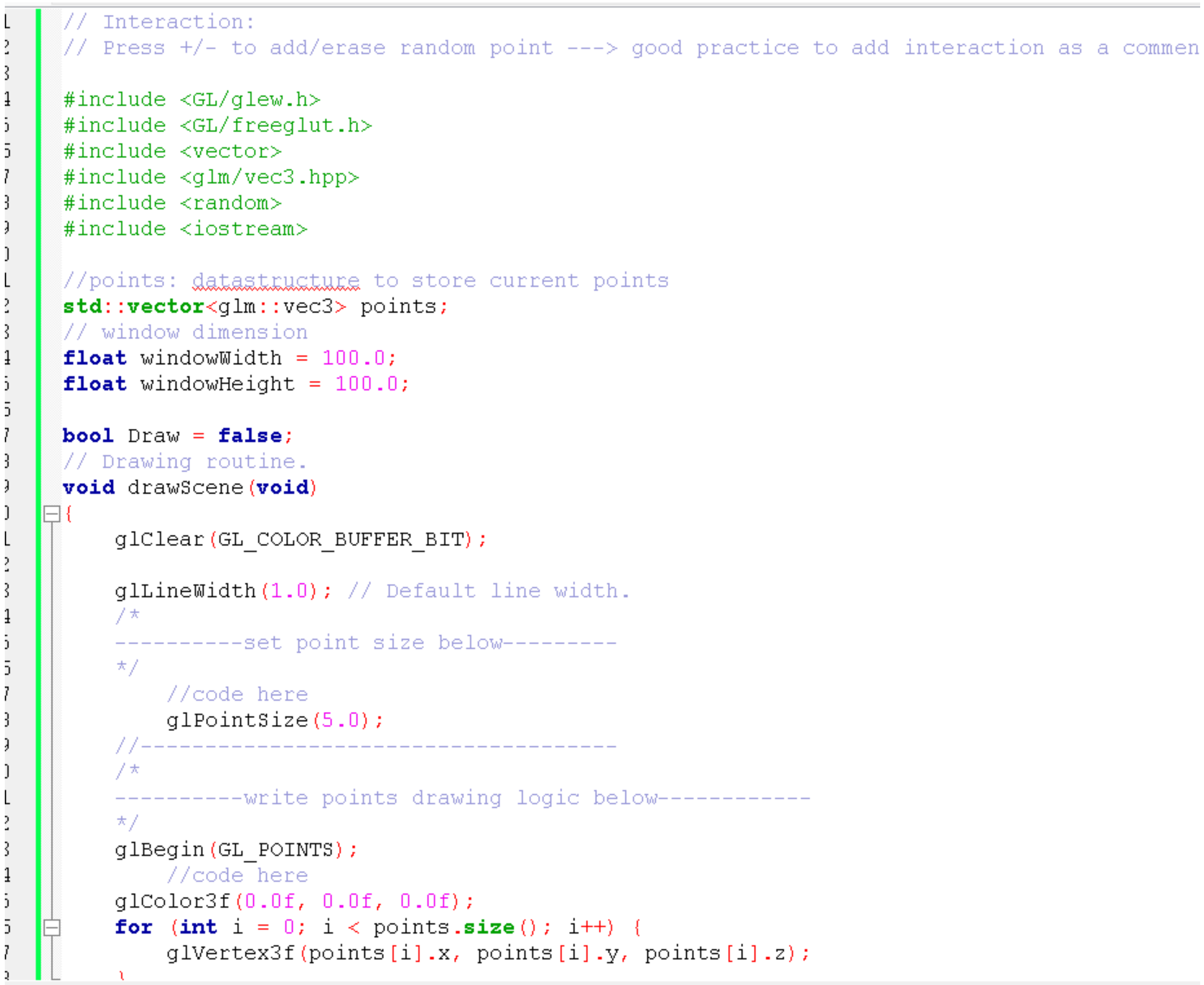
• For every two successive points, a line should be drawn connecting them.

A point is allowed to be part of only one line so that number of line 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).

Screenshot of working code :

I add Boolean variable draw which indicate when to draw line or not to draw (the lines are drawn if the number of points are more than 1).

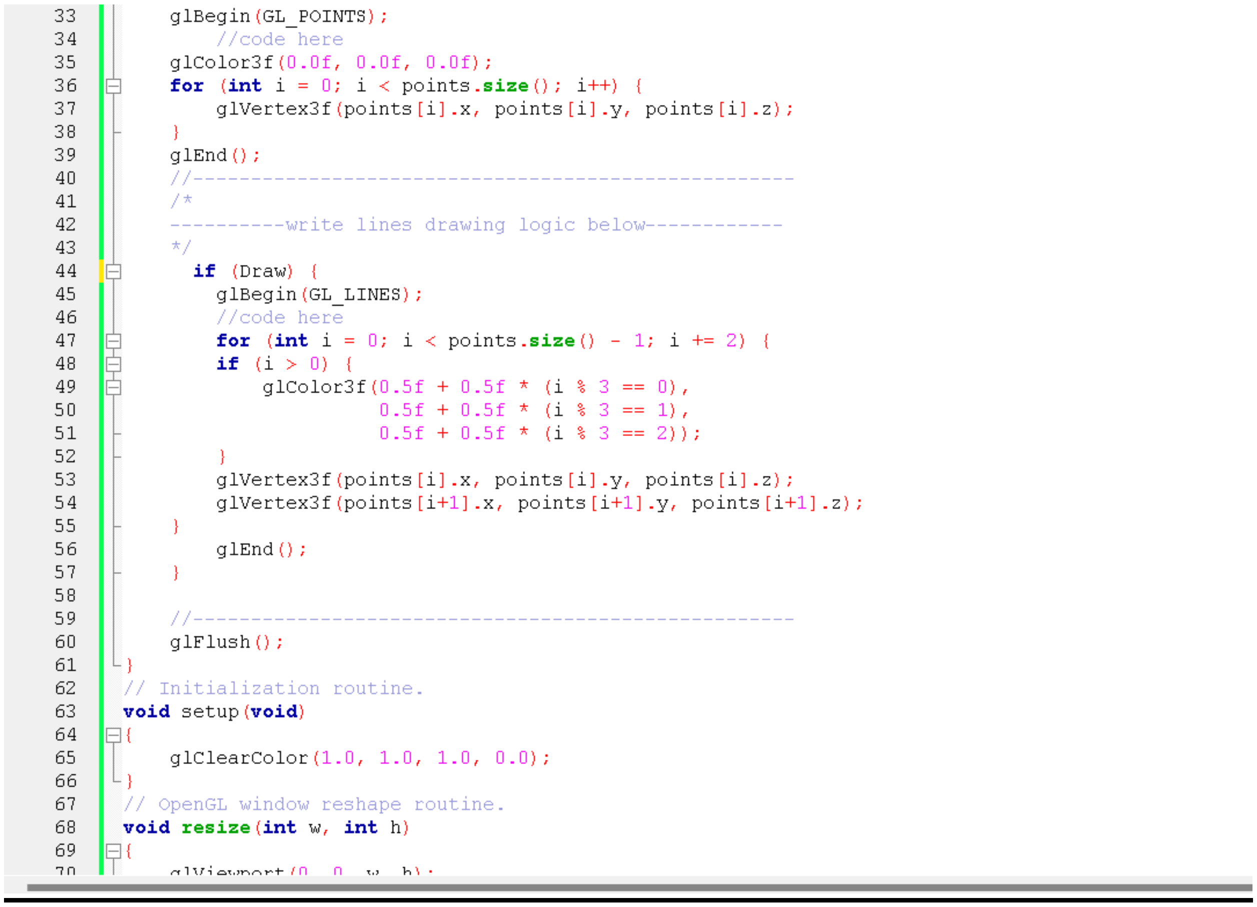
And then I loop on the vector of points then to start to draw each point in the vector and also point size is set to 5 pixels.



```
// Interaction:
// Press +/- to add/erase random point ---> good practice to add interaction as a commen

#include <GL/glew.h>
#include <GL/freeglut.h>
#include <vector>
#include <glm/vec3.hpp>
#include <random>
#include <iostream>

//points: datastructure to store current points
std::vector<glm::vec3> points;
// window dimension
float windowWidth = 100.0;
float windowHeight = 100.0;

bool Draw = false;
// Drawing routine.
void drawScene(void)
{
    glClear(GL_COLOR_BUFFER_BIT);

    glLineWidth(1.0); // Default line width.
    /*
    ----------set point size below---------
    */
        //code here
        glPointSize(5.0);
    //-----------------------------------
    /*
    ----------write points drawing logic below------------
    */
    glBegin(GL_POINTS);
        //code here
    glColor3f(0.0f, 0.0f, 0.0f);
    for (int i = 0; i < points.size(); i++) {
        glVertex3f(points[i].x, points[i].y, points[i].z);
```

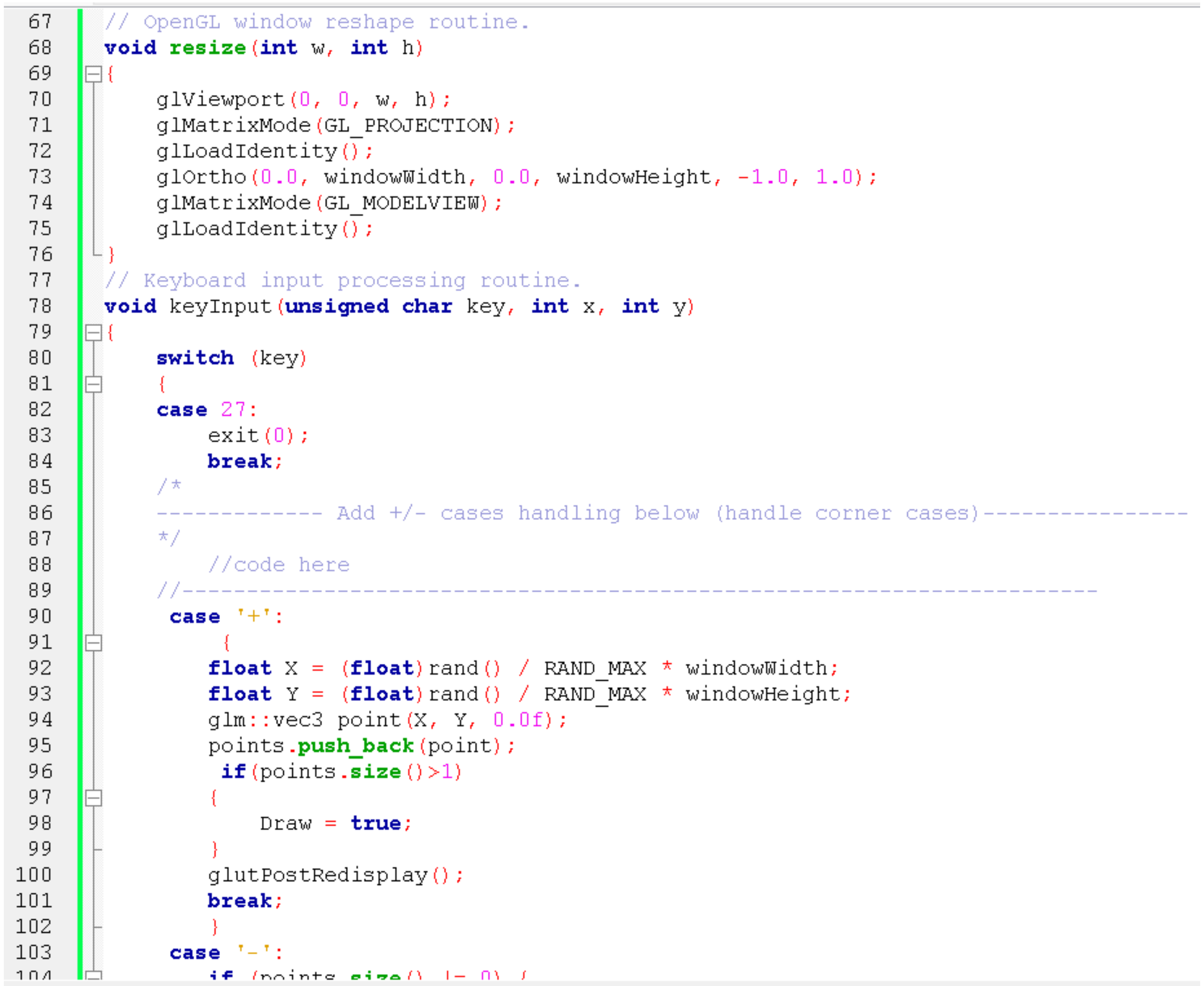
Then to draw the lines we check if the vector contain more than one point then start to draw lines between every two successive points

And it draw each line with different color .



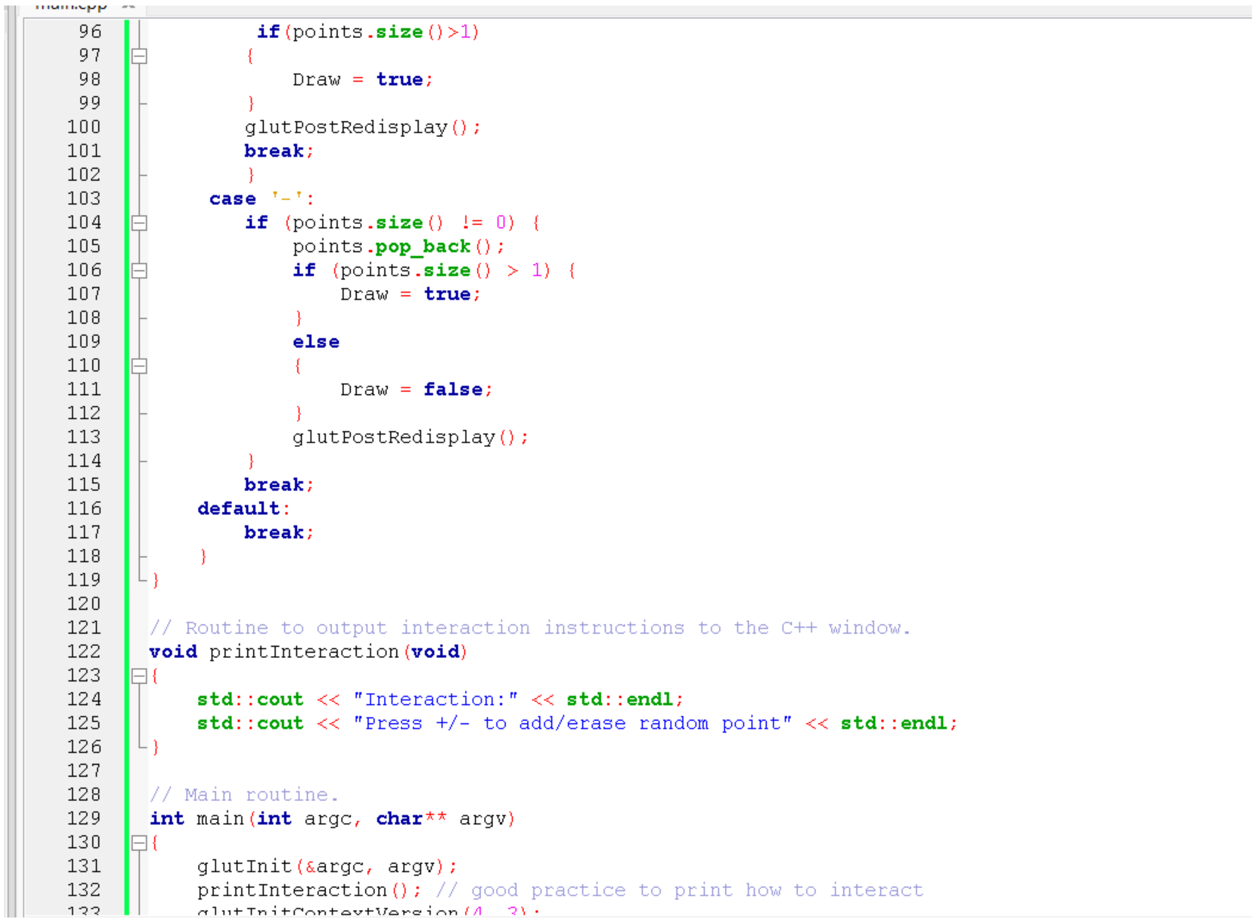
```
    glBegin(GL_POINTS);
        //code here
    glColor3f(0.0f, 0.0f, 0.0f);
    for (int i = 0; i < points.size(); i++) {
        glVertex3f(points[i].x, points[i].y, points[i].z);
    }
    glEnd();
    //---------------------------------------------------
    /*
    ----------write lines drawing logic below------------
    */
      if (Draw) {
        glBegin(GL_LINES);
        //code here
        for (int i = 0; i < points.size() - 1; i += 2) {
        if (i > 0) {
            glColor3f(0.5f + 0.5f * (i % 3 == 0),
                      0.5f + 0.5f * (i % 3 == 1),
                      0.5f + 0.5f * (i % 3 == 2));
        }
        glVertex3f(points[i].x, points[i].y, points[i].z);
        glVertex3f(points[i+1].x, points[i+1].y, points[i+1].z);
    }
        glEnd();
    }

    //---------------------------------------------------
    glFlush();
}
// Initialization routine.
void setup(void)
{
    glClearColor(1.0, 1.0, 1.0, 0.0);
}
// OpenGL window reshape routine.
void resize(int w, int h)
{
```

then when the user try to add new point he press on “+” button then the system start drawing points and between every two points it draw a line and the position of the point is at random place in the screen .

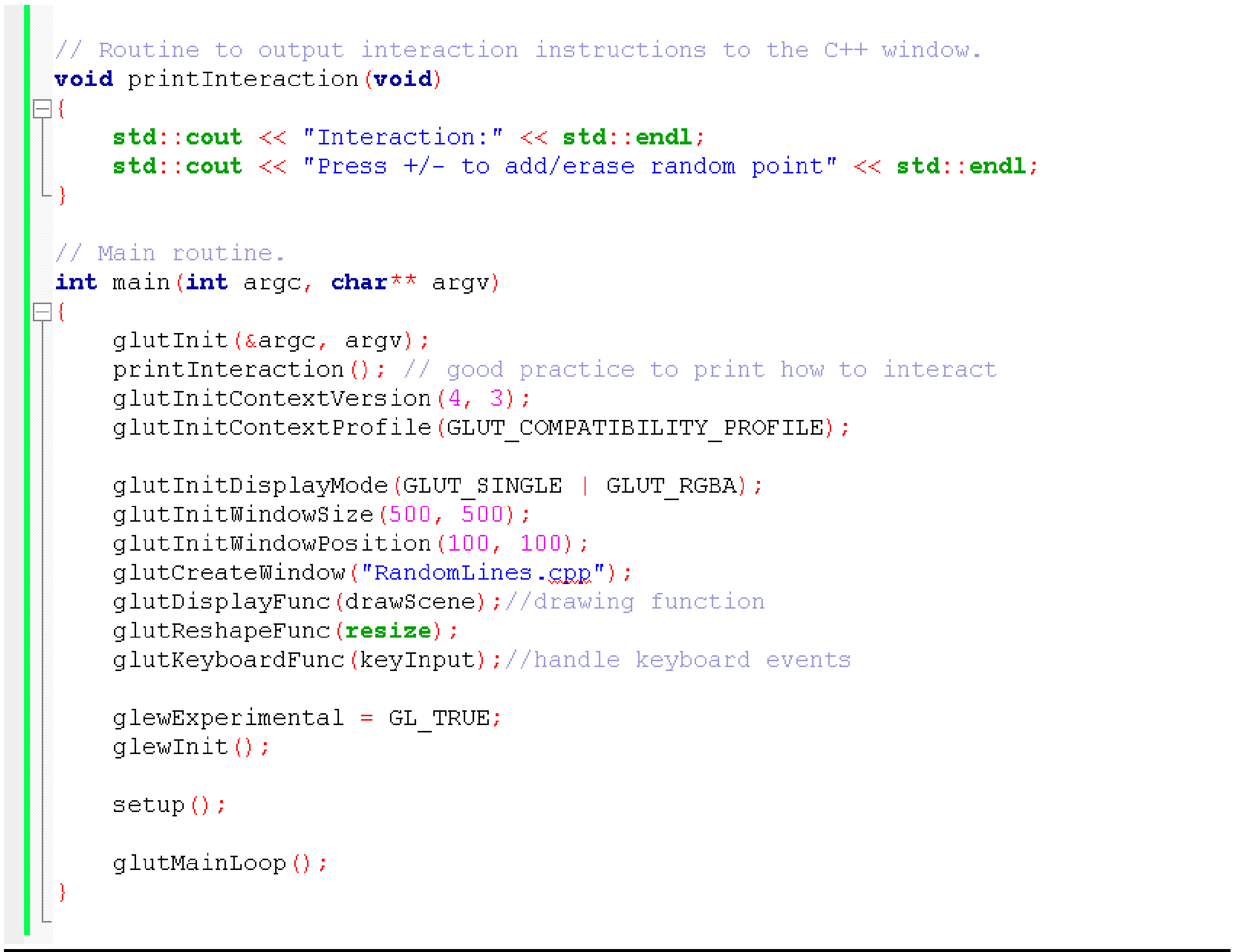


```
// OpenGL window reshape routine.
void resize(int w, int h)
{
    glViewport(0, 0, w, h);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, windowWidth, 0.0, windowHeight, -1.0, 1.0);
    glMatrixMode(GL_MODELVIEW);
    glLoadIdentity();
}
// Keyboard input processing routine.
void keyInput(unsigned char key, int x, int y)
{
    switch (key)
    {
    case 27:
        exit(0);
        break;
    /*
    ------------- Add +/- cases handling below (handle corner cases)----------------
    */
        //code here
    //-----------------------------------------------------------------------
     case '+':
         {
        float X = (float)rand() / RAND_MAX * windowWidth;
        float Y = (float)rand() / RAND_MAX * windowHeight;
        glm::vec3 point(X, Y, 0.0f);
        points.push_back(point);
         if(points.size()>1)
        {
            Draw = true;
        }
        glutPostRedisplay();
        break;
        }
     case '-':
        if (points.size() != 0) {
```

And when the user try to delete a point he press a “-“ and if this point contain a line the point and the line will be deleted . and check if it vector size contain more than one point to set Boolean with true else it will be set to be a false .



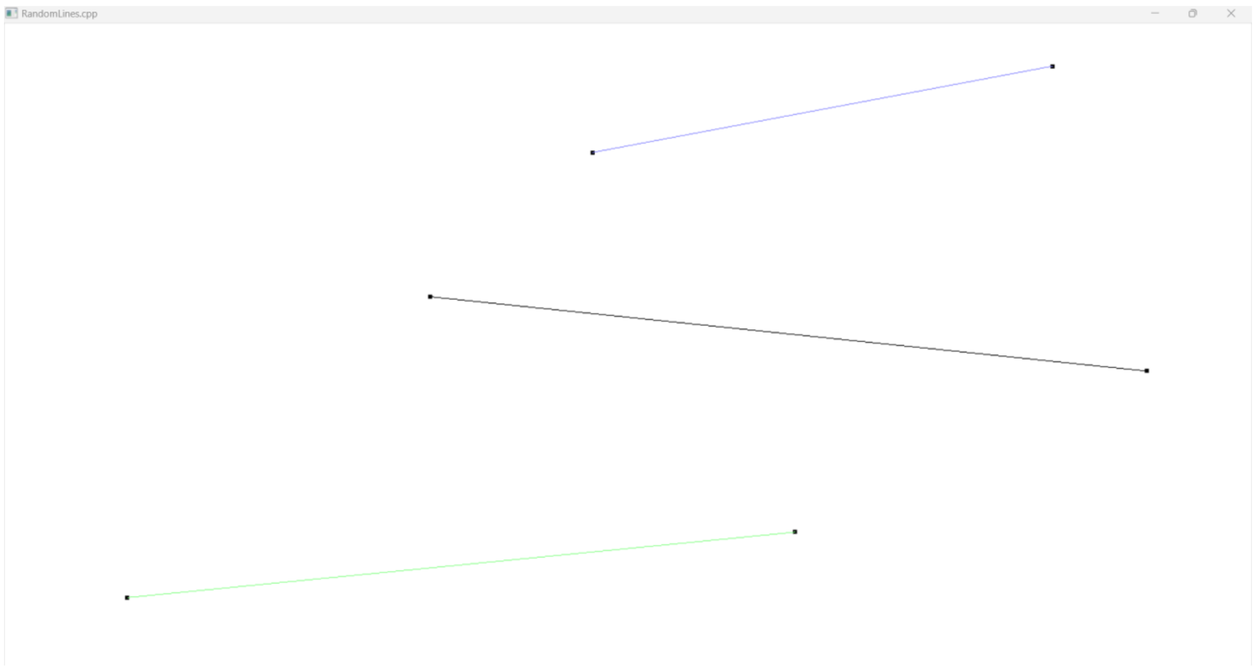
```
         if(points.size()>1)
        {
            Draw = true;
        }
        glutPostRedisplay();
        break;
        }
     case '-':
        if (points.size() != 0) {
            points.pop_back();
            if (points.size() > 1) {
                Draw = true;
            }
            else
            {
                Draw = false;
            }
            glutPostRedisplay();
        }
        break;
    default:
        break;
    }
}

// Routine to output interaction instructions to the C++ window.
void printInteraction(void)
{
    std::cout << "Interaction:" << std::endl;
    std::cout << "Press +/- to add/erase random point" << std::endl;
}

// Main routine.
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    printInteraction(); // good practice to print how to interact
    glutInitContextVersion(4, 3);
```



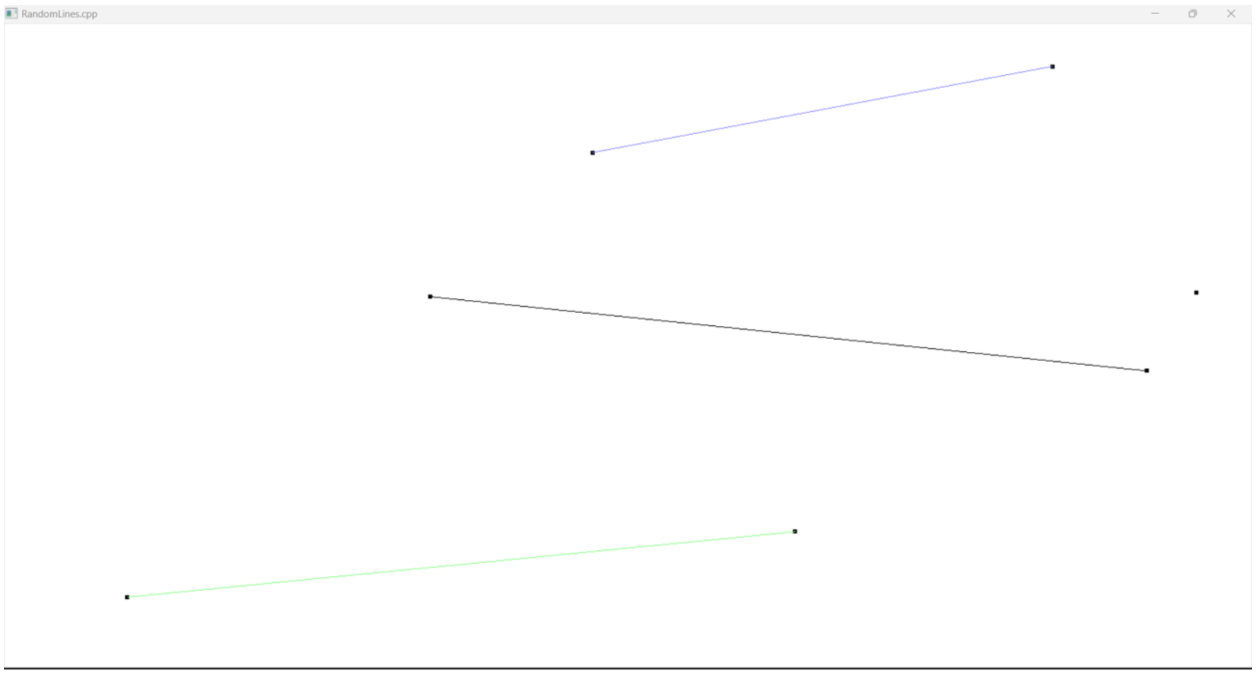
```cpp
// Routine to output interaction instructions to the C++ window.
void printInteraction(void)
{
    std::cout << "Interaction:" << std::endl;
    std::cout << "Press +/- to add/erase random point" << std::endl;
}

// Main routine.
int main(int argc, char** argv)
{
    glutInit(&argc, argv);
    printInteraction(); // good practice to print how to interact
    glutInitContextVersion(4, 3);
    glutInitContextProfile(GLUT_COMPATIBILITY_PROFILE);

    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowSize(500, 500);
    glutInitWindowPosition(100, 100);
    glutCreateWindow("RandomLines.cpp");
    glutDisplayFunc(drawScene);//drawing function
    glutReshapeFunc(resize);
    glutKeyboardFunc(keyInput);//handle keyboard events

    glewExperimental = GL_TRUE;
    glewInit();

    setup();

    glutMainLoop();
}
```

<u>Screenshot of points and lines :</u>

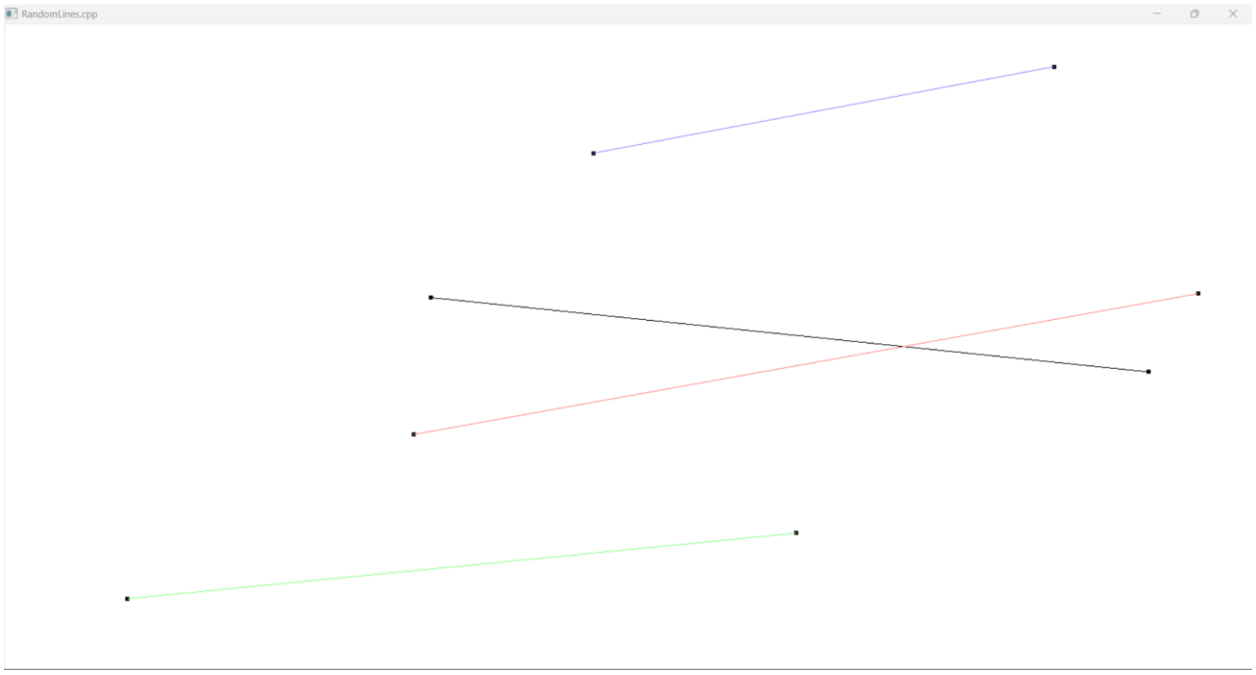
When the user enter “+” it start drawing .



Then user add “+” the program add a point without a line

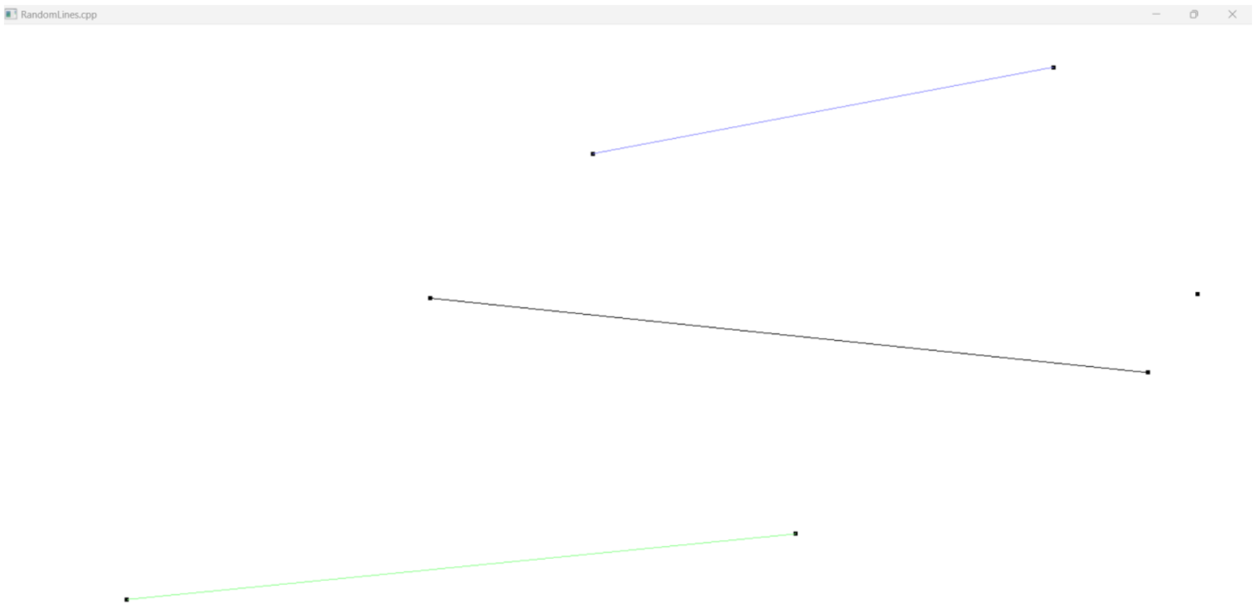


Then when user add another point the program connect with a line between these two points.

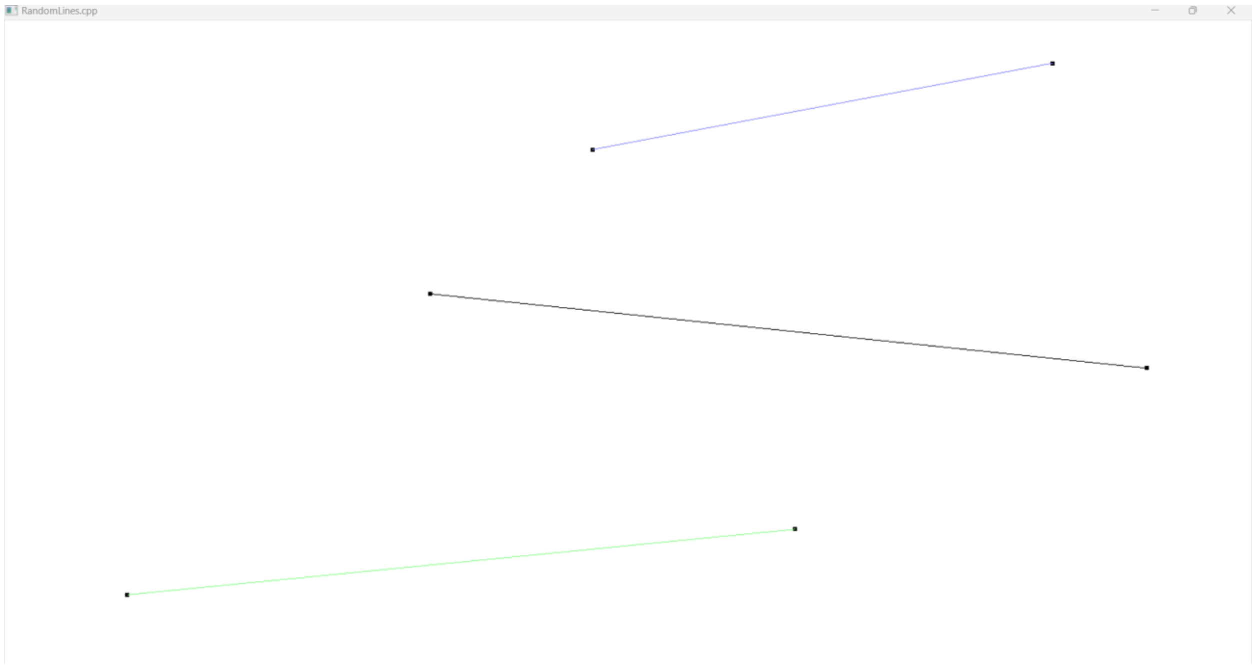


When user want to delete he press “-“ then it start to delete the last thing have been drawn :

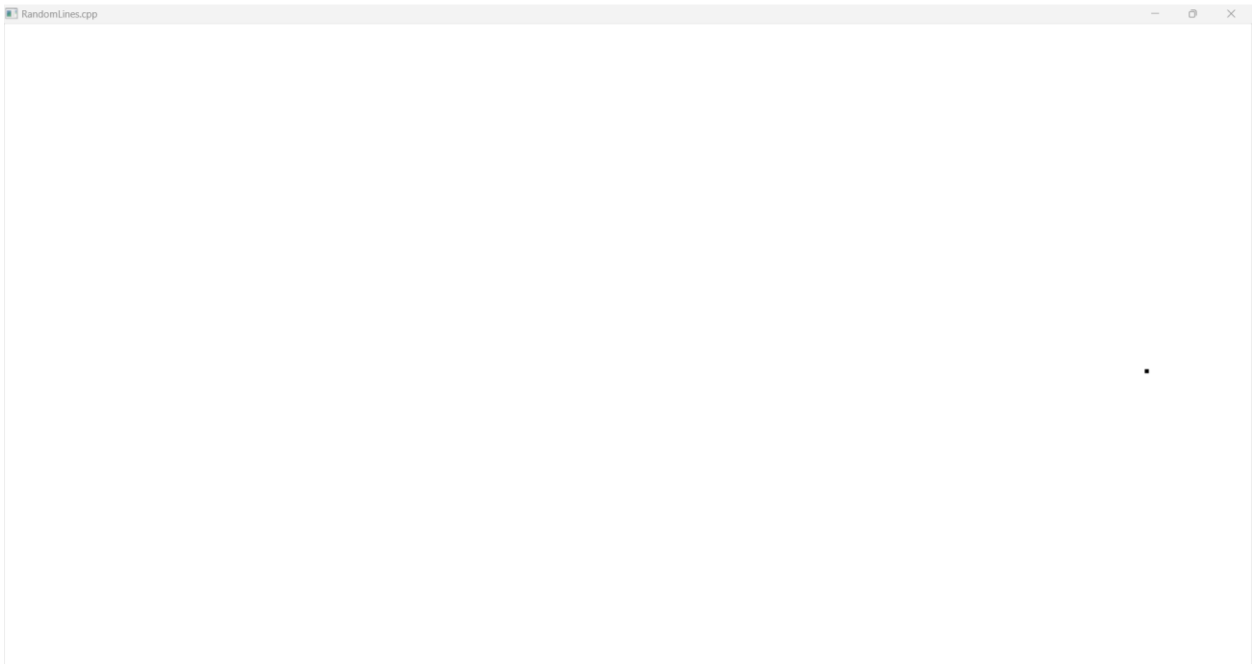
So the program delete the red line with point which is last thing have been drawn



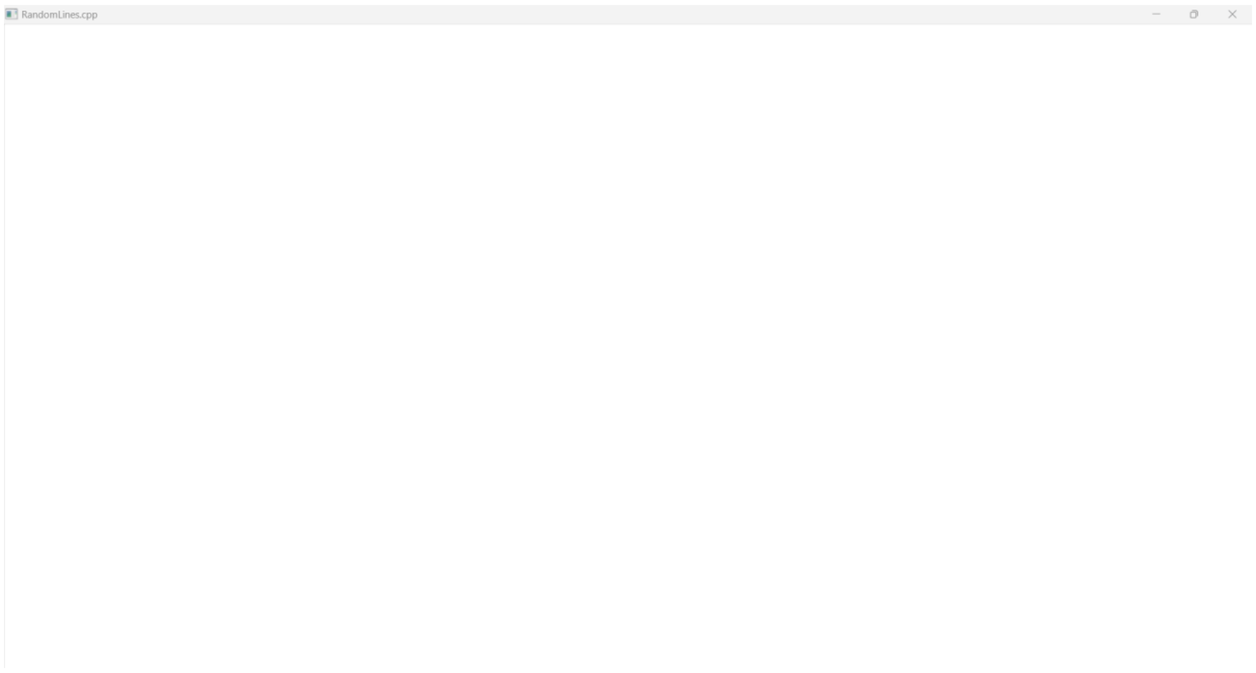
When user press “-“ another time the point will be deleted:



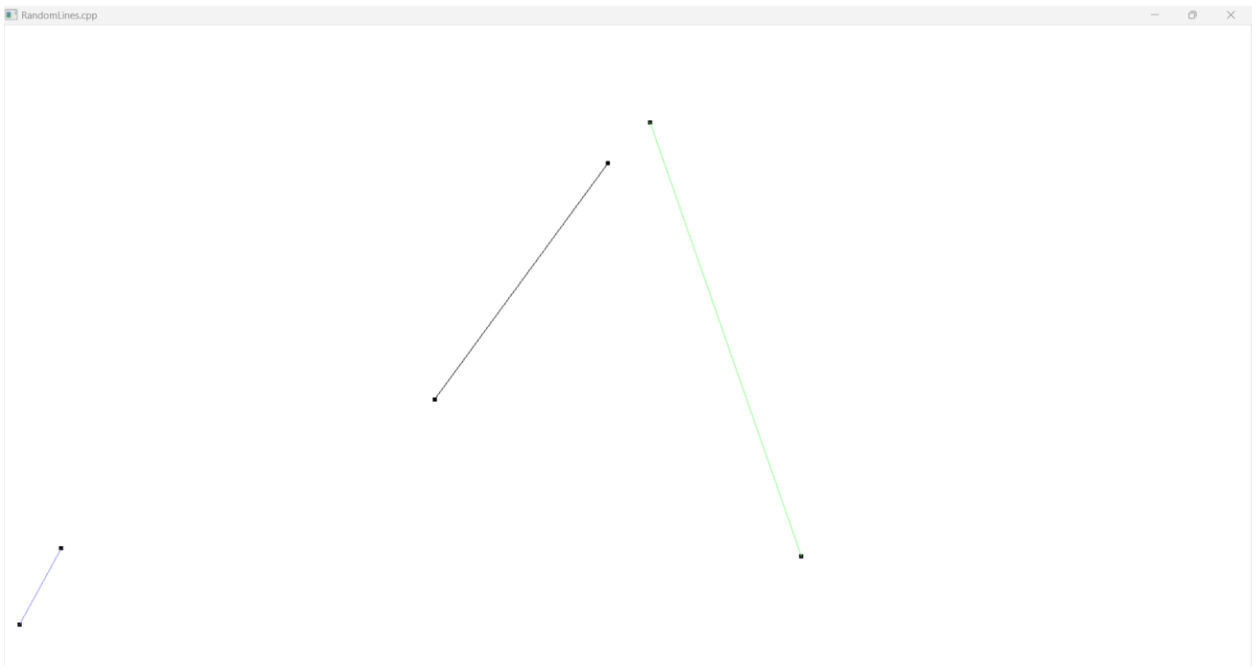
And handled the corner case when there is no points and the user try to delete it will not crash :



If user try to press “-“ the system will not crash



And user also can press”+” to start drawing again:


RandomLines.cpp