中山大學
SUN YAT-SEN UNIVERSITY

《计算机图形学作业》

实验报告

（作业三）

学 院 名 称 ：	数据科学与计算机学院
学 生 姓 名 ：	姚雪辉
学　　号 ：	15355119
专业（班级）：	16 软件工程四（7）班
时　　间 ：	2019　年　3　月　24　日

Basic

1. 使用 Bresenham 算法(只使用 integer arithmetic)画一个三角形边框：input 为三个 2D 点；output 三条直线（要求图元只能用 GL_POINTS ，不能使用其他，比如 GL_LINES 等）

算法主要分成三部分

第一：bresenham 部分，通过一个不断增加的坐标求得另一个坐标值，比如两点之间的 x 值不断增加，通过公式判断以及计算，得到 y 的值。

```
void bresenham(float y[], int p0, int numOfPoints, int deltaX, int deltaY) {
    if (numOfPoints < 1) {
        return;
    }
    for (int i = 0; i < numOfPoints; i++) {
        int pnext;
        if (p0 <= 0) {
            y[i + 1] = y[i];
            pnext = p0 + 2 * deltaY;
        }
        else {
            y[i + 1] = y[i] + 1;
            pnext = p0 + 2 * deltaY - 2 * deltaX;
        }
        p0 = pnext;
    }
}
```

第二：getAllPointsBetweenTwoPoints 部分，主要是因为斜率的不同，需要对 bresenham 的参数进行调整，主要有斜率小于-1，-1 到 0，0 到 1，大于 1 四种情况。

第三部分：利用 VAO 以及 VBO 画点，重载了 drawPoint 函数，可以使用

`drawPoint(float x, float y)`也可以使用

`void drawPoint(float x, float y,float r,float g,float b)`，然后 drawLine 直接调用 drawPoint 函数，drawLine 也有类似的重载。

2. 使用 Bresenham 算法(只使用 integer arithmetic)画一个三角形边框：input 为三个 2D 点；output 三条直线（要求图元只能用 GL_POINTS ，不能使用其他，比如 GL_LINES 等）直接调用 drawLine 函数，首尾相连。

3. 在 GUI 在添加菜单栏，可以选择是三角形边框还是圆，以及能调整圆的大小(圆心固定即可)

通过下拉选单选择画圆还是画三角形

```
ImGui::Begin("DrawLine and DrawCircle", &isOpen,ImGuiWindowFlags_MenuBar);
        if (ImGui::BeginMenuBar())
        {
            if (ImGui::BeginMenu("Menu"))
            {
                if (ImGui::MenuItem("DrawLine")) {
                    isLine = true;
                    isCircle = false;
                }
                if (ImGui::MenuItem("DrawCircle")) {
                    isLine = false;
                    isCircle = true;
                }
                ImGui::EndMenu();
            }
            ImGui::EndMenuBar();
        }
```

画圆也有通过 d 的公式，快速画圆，只用画出 45 度到 90 度的圆弧，其他的补上 8 个点就可以，这里需要归一化系数一样，而且不能随便改变界面的大小，否则会变成椭圆形。

```
void drawCircle(float inputX, float inputY, int inputR) {
    float newX = normlizeX(inputX);
    float newY = normlizeY(inputY);
    int x = 0;
    int y = inputR;
    int d = 1 - inputR;

    while (y >= x) {
        float realX = normlizeX(x);
        float realY = normlizeY(y);

        drawPoint(newX + realX, newY + realY);
```

```
            drawPoint(newX + realY, newY + realX);
            drawPoint(newX + realY, newY - realX);
            drawPoint(newX + realX, newY - realY);
            drawPoint(newX - realX, newY - realY);
            drawPoint(newX - realY, newY - realX);
            drawPoint(newX - realY, newY + realX);
            drawPoint(newX - realX, newY + realY);

            if (d < 0) {
                d = d + 2 * x + 3;
            }
            else {
                d = d + 2 * (x - y) + 5;
                y--;
            }
            x++;
        }
    }
```

Bonus

使用三角形光栅转换算法，用和背景不同的颜色，填充你的三角形。

首先得到包含三角形的最小的矩形

```
    int minX = minOfThree(x0, x1, x2);
    int minY = minOfThree(y0, y1, y2);
    int maxX = maxOfThree(x0, x1, x2);
    int maxY = maxOfThree(y0, y1, y2);
```

然后对每个点遍历，判断是否在三角形中，

```
bool isPointinTriangle(Vector3 A, Vector3 B, Vector3 C, Vector3 P)
```

判断 P 是否在 ABC 中，引入头文件 Vector3，其中定义了 Dot 方法

```
    float Dot(const Vector3& v) const
    {
        return x * v.x + y * v.y + z * v.z;
    }
```

采用网上的重心法判断点是否在三角形中。主要通过向量的方法，推到出容易判断的公式，主要参考下面的链接。

https://blog.csdn.net/wkl115211/article/details/80215421

对在三角形中的点，drawPoint。