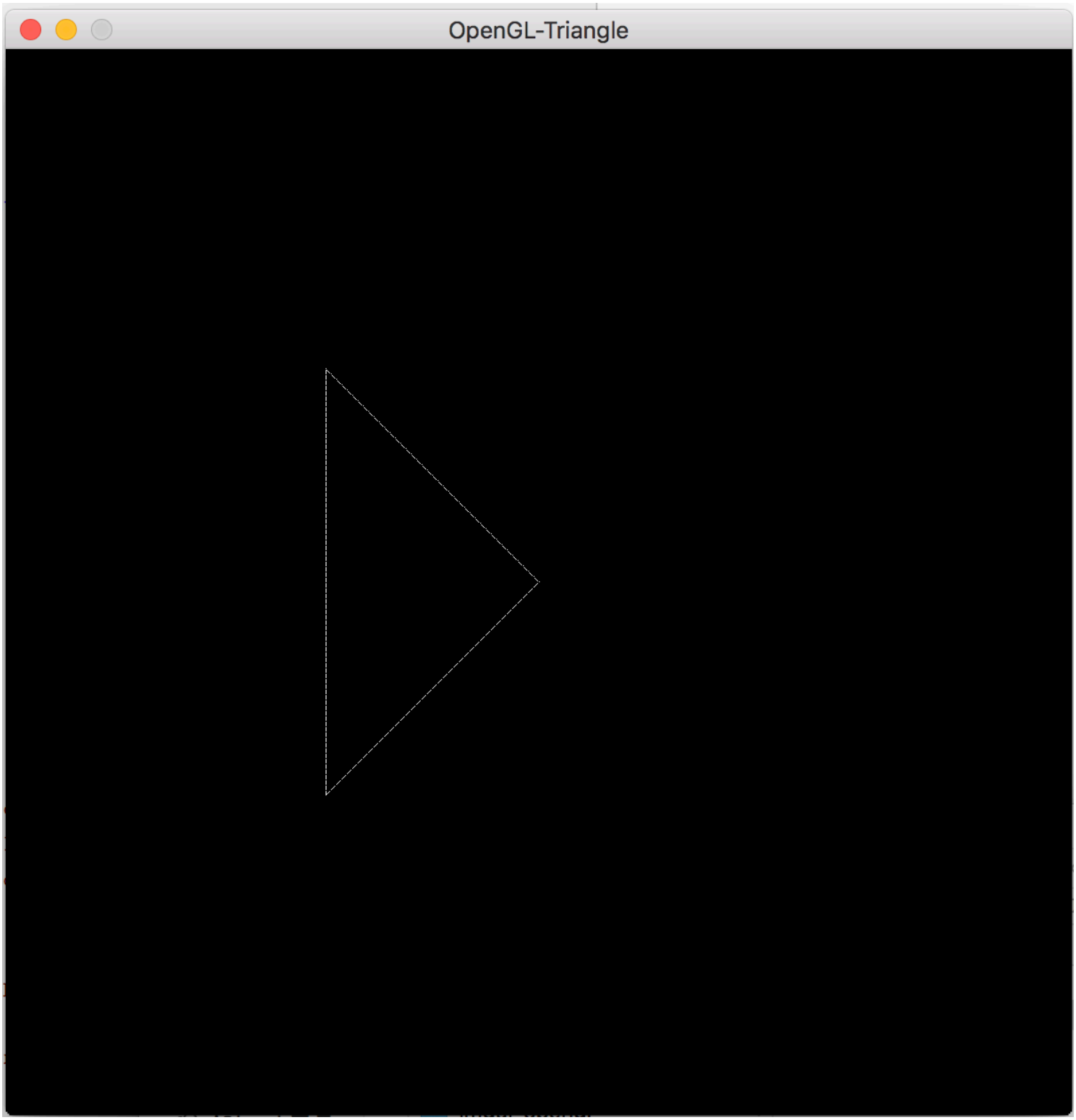
计算机图形学——hw3

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)。

- 效果截图

```
Please gives the x and y value of three points
with one space between them (like: 'x y')
Notice: the x and y value should be integers between -500 and 500

the x and y value of the first point?
-200 200
the coordinate of the first point: (-200, 200).
the x and y value of the second point?
-200 -200
the coordinate of the second point: (-200, -200).
the x and y value of the first point?
0 0
the coordinate of the Third point: (0, 0).
```



- **代码说明**
  在使用顶点着色器，片段着色器，着色器程序，VAO，VBO这些与渲染管道有关的代码与上次作业画三角形的基本一样，只是这次在渲染循环中用到的 `glDrawArrays` 函数的第一个参数不再是 `GL_TRIANGLES` 而是 `GL_POINTS` ,因为这一次是一个点一点来绘制三角形，所以重点是得到三角形的每个点的坐标的数组，这就用到了这次作业提到的Bresenham画线算法。
  Brensenham算法：先只考虑位于线段所在直线经过第一象限与x轴夹角小于45度，也就是01时,y的变化是向上递增的，x的变化由bresenham算法决定,当m>1时，也如上图，只不过这次传入的是x的数组，当m<-1时，x值变为相反数传入。

```
void bresenham(int array[], int p, int i, int length, int dx, int dy) {
```

```
    if(i == length-1)
        return;
    int next_p;
    if(p <= 0) {
        array[i+1] = array[i];
        next_p = p + 2*dy;
    } else {
        array[i+1] = array[i] + 1;
        next_p = p + 2*dy - 2*dx;
    }
    bresenham(array, next_p, i+1, length, dx, dy);
}
void Get_Points_of_Line(int xarray[], int yarray[], int* length, int x0, int x1, int y0, int y1) {
    if(x0 == x1) { //斜率不存在是，x不变，y递增
        if (y0 > y1) {
            swap(&y0, &y1);
        }
        *length = y1 - y0 + 1;
        for (int i = 0; i < *length; i++) {
            xarray[i] = x0;
            yarray[i] = y0 + i;
        }
    } else {  //斜率存在
        float m = float(y1-y0)/float(x1-x0);
        //|m|<=1,让点1在点0的右边；|m|>1，让点1在点0的上方
        if((fabs(m) <= 1 && x0 > x1)||(fabs(m) > 1 && y0 > y1)) {
            swap(&x0, &x1);
            swap(&y0, &y1);
        }

        int dx=x1-x0;
        int dy=y1-y0;
        m = float(dy)/float(dx);

        if(fabs(m)<=1) {
            *length = x1-x0+1;
            for (int i = 0; i < *length; i++)
                xarray[i] = x0 + i;

            if(dy>=0) { //当 0<m<=1
                yarray[0] = y0;
                yarray[*length-1] = y1;
                int p0 = 2*dy - dx;
                bresenham(yarray, p0, 0, *length, dx, dy);
            } else {//-1<=m<0
                yarray[0] = -y0;
                yarray[*length-1] = -y1;
                int p0 = 2*(-dy) - dx;
                bresenham(yarray, p0, 0, *length, dx, -dy);
                for(int i = 0;i < *length; i++)
                    yarray[i] = -yarray[i];
            }
        } else {
            *length=y1-y0+1;
            //当|m|>1时，y的变化是向上递增的，x的变化由bresenham算法决定
            for (int i = 0; i < *length; i++)
                yarray[i] = y0 + i;
            if( dx >= 0) {
                //m>1
                xarray[0] = x0;
                xarray[*length-1] = x1;
                int p0 = 2*dx - dy;
                bresenham(xarray, p0, 0, *length, dy,dx);
            } else {
                //m<-1
                xarray[0] = -x0;
                xarray[*length-1] = -x1;
                int p0 = 2*(-dx) - dy;
```

```
                bresenham(xarray, p0, 0, *length, dy, -dx);
                for(int i = 0; i < *length; i++)
                    xarray[i] = -xarray[i];
            }
        }
    }
}
```

这样得到的只是一条直线，画一个三角形，需要三个顶点画三条直，所以将输入的三个顶点，两两一组作为 `Get_Points_of_Line` 的参数，得到三条直线坐标，再将它们汇总到一个数组 `points` 中，这个数组每两个元素组成一组，表示点的x,y坐标值，要注意的是这个数组是 `float` 类型，因为在显示的时候要将坐标归一化。

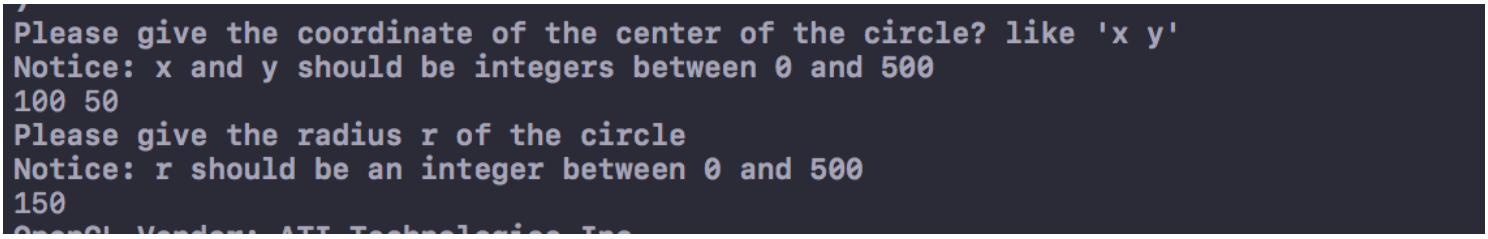
```
void Gat_Points_of_Triangle(float points[], int*length) {
int x0, x1, x2, y0, y1, y2;
cout << "Please gives the x and y value of three points" << endl;
cout << "with one space between them (like: 'x y')" << endl;
cout << "Notice: the x and y value should be integers between -500 and 500" << endl << endl;
cout << "the x and y value of the first point?" << endl;
cin >> x0 >> y0;
cout << "the coordinate of the first point: (" << x0 << ", " << y0 <<")." << endl;
cout << "the x and y value of the second point?" << endl;
cin >> x1 >> y1;
cout << "the coordinate of the second point: (" << x1 << ", " << y1 <<")." << endl;
cout << "the x and y value of the first point?" << endl;
cin >> x2 >> y2;
cout << "the coordinate of the Third point: (" << x2 << ", " << y2 <<")." << endl;

int length1, length2, length3;
int xarray1[1001], yarray1[1001],xarray2[1001], yarray2[1001],xarray3[1001], yarray3[1001];

Get_Points_of_Line(xarray1, yarray1, &length1, x0, x1, y0, y1);
Get_Points_of_Line(xarray2, yarray2, &length2, x0, x2, y0, y2);
Get_Points_of_Line(xarray3, yarray3, &length3, x1, x2, y1, y2);

float fxarray1[1001], fxarray2[1001], fxarray3[1001];
float fyarray1[1001], fyarray2[1001], fyarray3[1001];
int j=0;
for(int i=0; i < length1; i++) {
    fxarray1[i] = normalize(xarray1[i]);
    fyarray1[i] = normalize(yarray1[i]);
    points[j++] = fxarray1[i];
    points[j++] = fyarray1[i];
}
for(int i=0; i<length2; i++) {
    fxarray2[i] = normalize(xarray2[i]);
    fyarray2[i] = normalize(yarray2[i]);
    points[j++] = fxarray2[i];
    points[j++] = fyarray2[i];
}
for(int i=0; i<length3; i++) {
    fxarray3[i] = normalize(xarray3[i]);
    fyarray3[i] = normalize(yarray3[i]);
    points[j++] = fxarray3[i];
    points[j++] = fyarray3[i];
}
*length = 2*(length1+length2+length3);
}
```

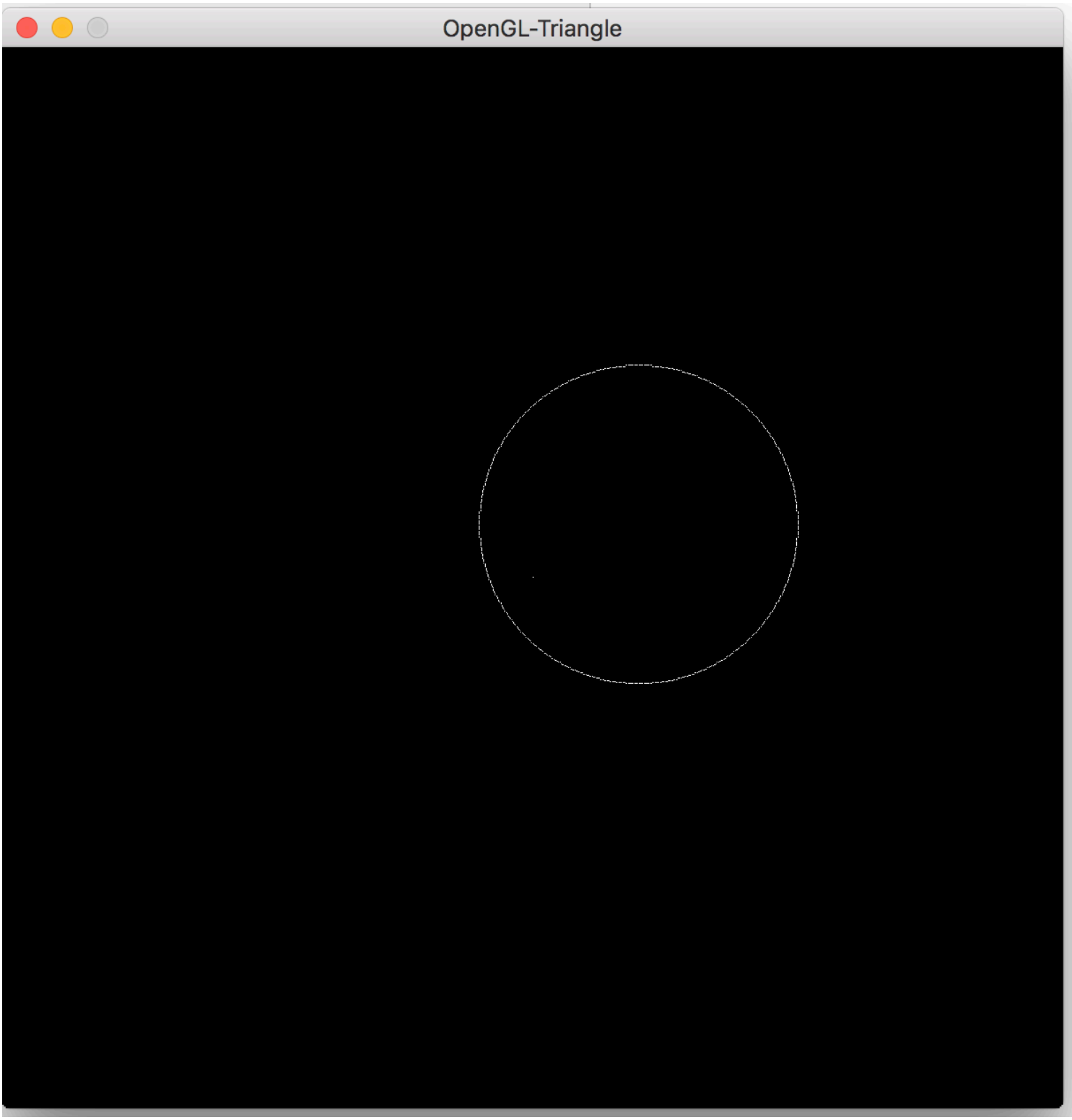
根据得到点的坐标数组，就可以绘制出每一个点，和在一起就是一个三角形了，还要注意的是在解析数组的时候因为是每两个一组，所以偏移量和大小有所改变，GLSL顶点着色器的源代码也有所改变

```
glVertexAttribPointer(0, 2, GL_FLOAT, GL_FALSE, 2 * sizeof(GLfloat), (GLvoid*)0);
...
...
const char *vertexShaderSource = "#version 330 core\n"
"layout (location = 0) in vec2 aPos;\n"
"void main()\n"
"{\n"
"   gl_Position = vec4(aPos.x, aPos.y, 0.0, 1.0);\n"
"}\0";
```

2. 使用Bresenham算法(只使用integer arithmetic)画一个圆:input为一个2D点(圆心)、一个integer半径;output为一个圆。

- **效果截图**





- **代码说明**
  通过查阅资料得到brensenham画圆算法的过程： 圆是中心对称的特殊图形，所以可以八等分，只需对八分之一的圆弧求解，其他圆弧可以由对称变化得到。
  我们求的八分之一圆弧为(0, R)-(R√2,R√2)，可知最大位移是方向， `x0 = 0` ，`y0 = R` ，每次对x自增，然后判断y是否减1，直到 `x>=y` 为止(从点(0, R)到圆的八分之一处就有这种情况)。
  误差量由 `F(x, y) = x^2 + y^2 - R^2` 给出。先找递推关系
  - 若当前 `d = F(x + 1, y - 0.5) > 0` ，则y须减1，则下一d值为 `d = F(x + 2, y - 1.5) = (x + 2)^2 + (y - 1.5)^2 - R^2 = (x + 1)^2 + (x - 0.5)^2 - R^2 + 2x + 3 - 2y + 2 = d + 2x - 2y + 5` ;
  - 若当前 `d = F(x + 1, y - 0.5) < 0` ，则y不变，只有x增1，则下一d值为 `d = F(x + 2, y - 0.5) = d + 2x + 3` 。

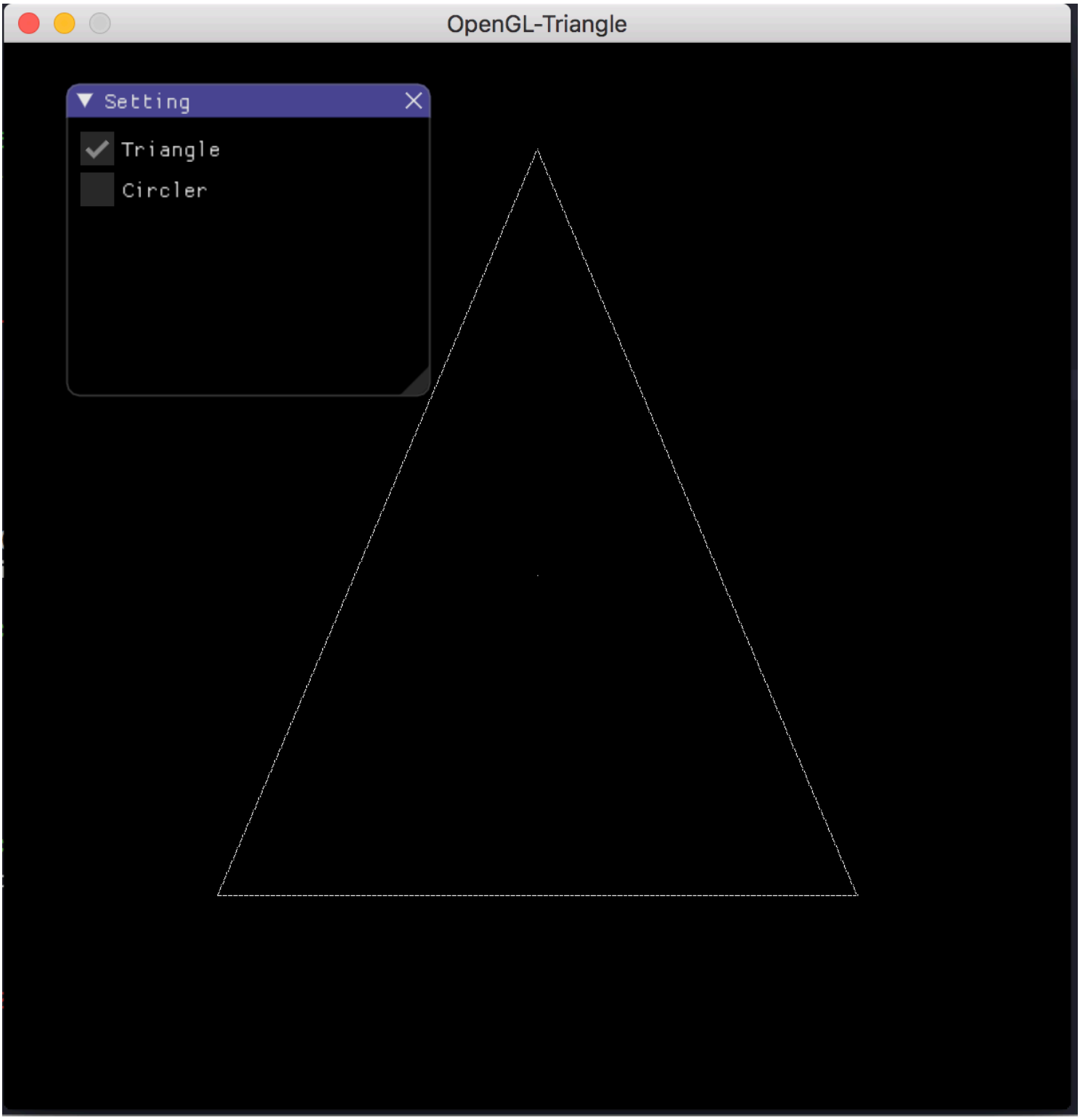
d的初值， `d0 = F(1, R - 0.5) = 1.25 - R` ，则可以对d - 0.25进行判断，因为递推关系中只有整数运算，所以d - 0.25 > 0即d > 0.25，这和d > 0等价，所以d取初值 `1 - R` 。

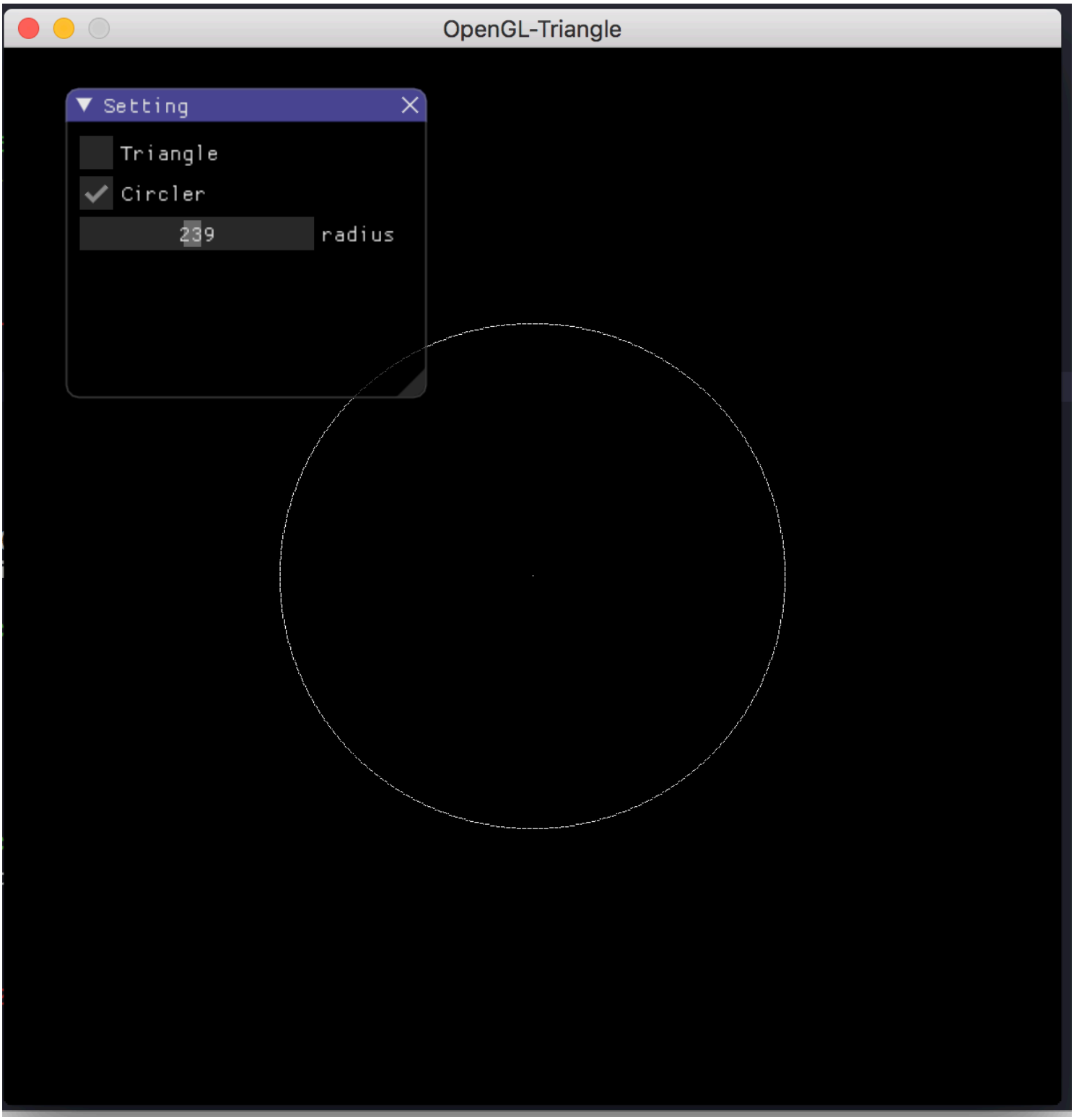
```
void Get_Points_Circle(float points[], int* index) {
    int r, cx,cy;
    cout << "Please give the coordinate of the center of the circle? like 'x y'" << endl << "Notice: x and y should be integers be
tween 0 and 500" << endl;
    cin >> cx >> cy;
    cout << "Please give the radius r of the circle" << endl << "Notice: r should be an integer between 0 and 500" << endl;
    cin >> r;
    float fcx=normalize(cx);//偏移值
    float fcy=normalize(cy);
    int x = 0, y = r;
    int yy =(int)(r*1.0/(sqrt(2)));
    int d = 1-r;
    while(yy >= x) {
        float fx = normalize(x); //0
        float fy = normalize(y); //r
        //(x,y) (-x,-y) (-x,y) (x,-y)
        points[++(*index)] = fcx+fx;
        points[++(*index)] = fcy+fy;
        points[++(*index)] = fcx-fx;
        points[++(*index)] = fcy-fy;
        points[++(*index)] = fcx-fx;
        points[++(*index)] = fcy+fy;
        points[++(*index)] = fcx+fx;
        points[++(*index)] = fcy-fy;
        //(y,x) (-y,-x) (-y,x) (y,-x)
        points[++(*index)] = fcx+fy;
        points[++(*index)] = fcy+fx;
        points[++(*index)] = fcx-fy;
        points[++(*index)] = fcy-fx;
        points[++(*index)] = fcx-fy;
        points[++(*index)] = fcy+fx;
        points[++(*index)] = fcx+fy;
        points[++(*index)] = fcy-fx;

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

---

3. 在GUI在添加菜单栏，可以选择是线还是圆，以及能调整圆的大小(圆心固定即可)。

- **运行结果截图**


OpenGL-Triangle
Setting
Triangle
Circler



- **代码说明**
  将上面的输入代码改为GUI即可

```
if (setting) {
    ImGui::Begin("Setting", &setting);
    ImGui::Checkbox("Triangle", &Triangle);
    ImGui::Checkbox("Circler", &Circle);
    if (Triangle) {
        for (int i = 0; i < 10000; i++)
            points[i] = 0;
            Gat_Points_of_Triangle(points, &index);
            Circle = false;
        }else if (Circle) {
            int radius;
            ImGui::SliderInt("radius", &radius, 0, 500);
            index = -1;
            for (int i = 0; i < 10000; i++)
                points[i] = 0;
            Get_Points_Circle(points, &index, radius);
            Triangle=false;
        }
        ImGui::End();
    }
}
```