Homework 3 - Draw line

16340082 黄俊凯

Bresenham Triangle

实现思路

1. 我的思路是先实现画一条线的，然后三个定点两两作为线段端点，即可画出三角形。在只能使用 `GL_POINTS` 图元的情况下，有两种画线的思路，一是生成一个点画一个点，二是将需画的点存在一个顶点数组中，然后再绑定到VBO上，一次性绘画。我选择的是第二个思路。
2. Algorithm (摘自 wiki)

```
plotLine(x0,y0, x1,y1)
    if abs(y1 - y0) < abs(x1 - x0)
        if x0 > x1
        plotLineLow(x1, y1, x0, y0)
        else
        plotLineLow(x0, y0, x1, y1)
        end if
    else
        if y0 > y1
        plotLineHigh(x1, y1, x0, y0)
        else
        plotLineHigh(x0, y0, x1, y1)
        end if
    end if

plotLineLow(x0,y0, x1,y1)
    dx = x1 - x0
    dy = y1 - y0
    yi = 1
    if dy < 0
        yi = -1
        dy = -dy
    end if
    D = 2*dy - dx
    y = y0

    for x from x0 to x1
        plot(x,y)
        if D > 0
        y = y + yi
        D = D - 2*dx
        end if
        D = D + 2*dy

plotLineHigh(x0,y0, x1,y1)
```

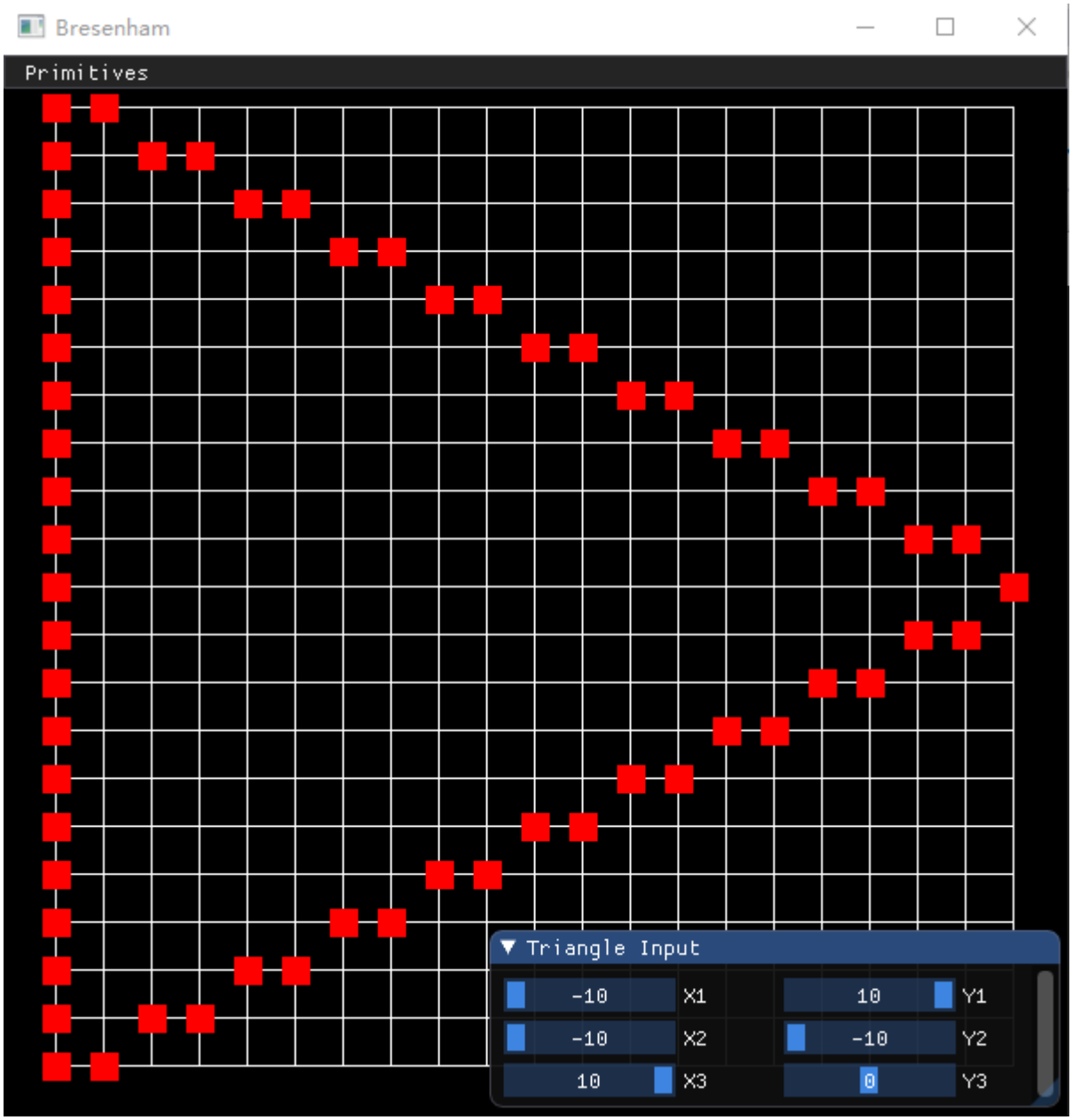
```
    dx = x1 - x0
    dy = y1 - y0
    xi = 1
    if dx < 0
        xi = -1
        dx = -dx
    end if
    D = 2*dx - dy
    x = x0

    for y from y0 to y1
        plot(x,y)
        if D > 0
        x = x + xi
        D = D - 2*dy
        end if
        D = D + 2*dx
```

3. 其中具体的公式推导这里就不赘述了，上面的链接都有，画线需要考虑4种情况：`m > 1`, `0 < m <= 1`, `-1 <= m < 0` 及 `m < -1`。其中 `|m| < 1` 和 `|m| >= 1` 可以分别考虑，区别就是步进：`xi` 和 `yi` 取 1 还是 -1。

相关函数

- `int Bresenham_line(int x0, int y0, int x1, int y1, float* &points, float scale)`
  - 其中 `x0, y0, x1, y1` 为线段端点，`points` 为指向定点数组的指针，`scale` 与本算法无关。
  - 返回值为定点数组大小，即所画点个数。
  - 具体实现与上述伪代码中的 `plotLine` 一致。
- `void plotLineLow(int x0, int y0, int x1, int y1, float* &points, float scale);`
  - 其中参数与上述 `Bresenham_line` 一致
  - 该函数针对 `|m|<1` 的情况
- `void plotLineHigh(int x0, int y0, int x1, int y1, float* &points, float scale);`
  - 其中参数与上述 `Bresenham_line` 一致
  - 该函数针对 `|m|<1` 的情况

效果



Bresenham Circle

实现思路

1. 与画线一样，我选择将需要画的点存放在定点数组中，然后绑定到VBO上一次性画完。
2. Algorithm (参考 A Fast Bresenham Type Algorithm For Drawing Circles)

```
procedure PlotCircle(CX, CY, R : longint);
begin
    var X, Y : longint;
        XChange, YChange : longint;
        RadiusError : longint;
        X := R;
        Y := 0;
    XChange := 1 - 2*R;
    YChange := 1;
```

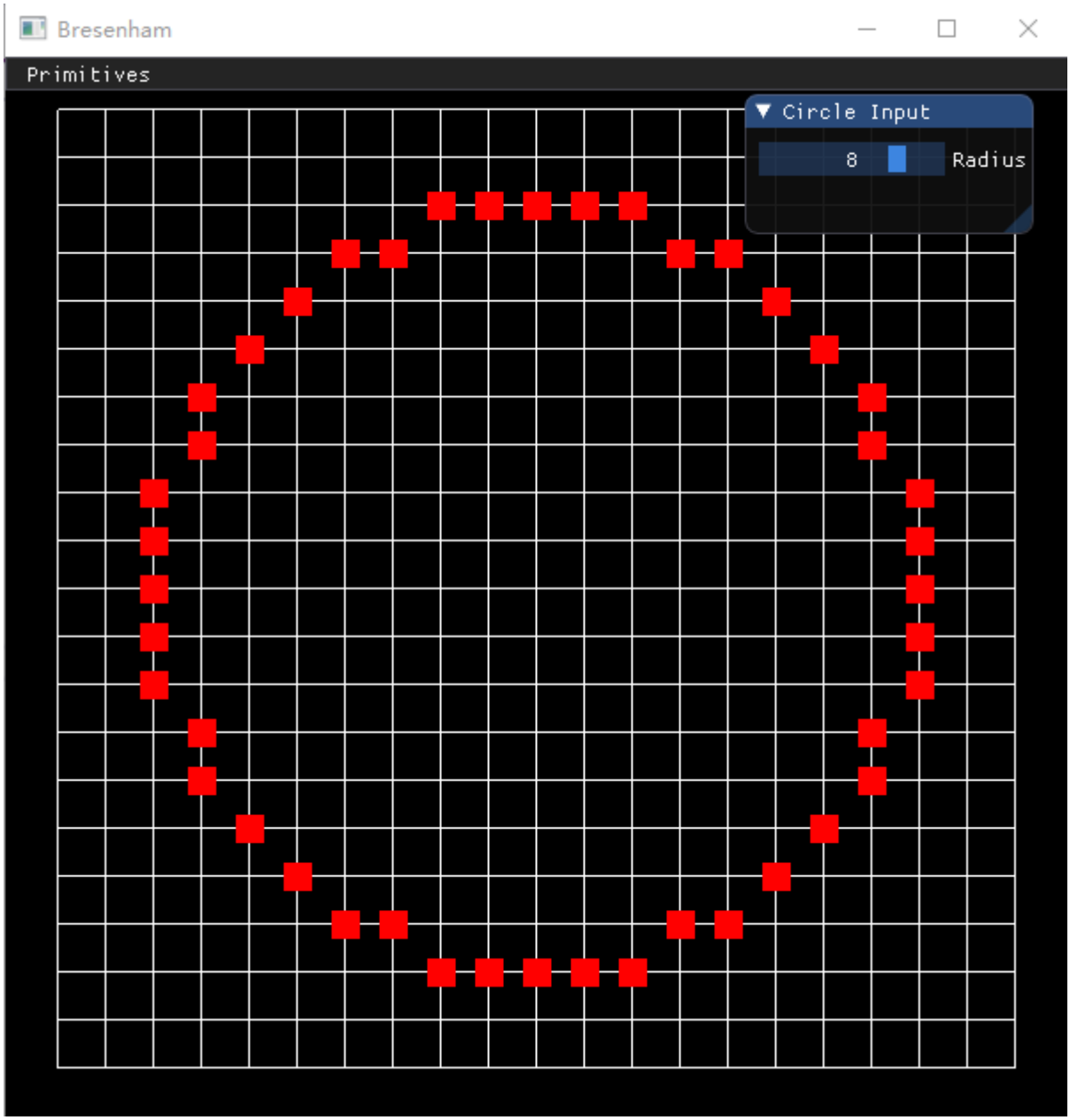
```
    RadiusError := 0;
    while ( X Y ) do
    begin
        Plot8CirclePoints(X,Y); {subroutine appears below}
        inc(Y);
        inc(RadiusError, YChange);
        inc(YChange,2);
        if ( 2*RadiusError + XChange > 0 ) then
        begin
            dec(X);
            inc(RadiusError, XChange);
            inc(XChange,2)
        end
    end
end; {procedure PlotCircle}

procedure Plot8CirclePoints(X,Y : longint);
begin
    PutPixel(CX+X, CY+Y); {point in octant 1}
    PutPixel(CX-X, CY+Y); {point in octant 4}
    PutPixel(CX-X, CY-Y); {point in octant 5}
    PutPixel(CX+X, CY-Y); {point in octant 8}
    PutPixel(CX+Y, CY+X); {point in octant 2}
    PutPixel(CX-Y, CY+X); {point in octant 3}
    PutPixel(CX-Y, CY-X); {point in octant 6}
    PutPixel(CX+Y, CY-X); {point in octant 7}
end; {procedure Plot8CirclePoints}
```

3. 其中最基本的思路就是从X轴的正方向开始，即 `(radius, 0)`，判断 $|(x_i-1)^2+(y_i+1)^2-r^2|<|x_i^2+(y_i+1)^2-r^2|$ 是否成立，如果成立则 x_{i-1} 等于 x_i-1，直到 $x_i=y_i$ 我们就可以认为已经画完一个八分圆，之后其他点只需从这八分圆分别做对称变换即可得到坐标值。

相关函数

- `int Bresenham_circle(int radius, float* &points, float scale);`
  - 其中 `radius` 为所画圆的半径，`points` 为顶点数组指针，`scale` 与算法无关
  - 返回值为点的个数

效果



GUI 相关设置

为了让每次只展示一个图形（一个输入框），进行如下操作：

```
if (ImGui::BeginMainMenuBar()) {
    if (ImGui::BeginMenu("Primitives")) {
        if (ImGui::MenuItem("Line")) { primitive_type = 1; }
        if (ImGui::MenuItem("Triangle")) { primitive_type = 2; }
        if (ImGui::MenuItem("Circle")) { primitive_type = 3; }
        ImGui::EndMenu();
    }
    ImGui::EndMainMenuBar();
}

if (primitive_type == 1) {...}
else if (primitive_type == 2) {...}
else {...}
```

即每次渲染循环中，只会处理一个图形。