Report

1. 实现两个函数。一个函数 drawLine 用于画线，输入 4 个参数（两个点各自的 x，y 坐标）画一条直线，无返回值。一个用于 drawTriangle，输入 6 个参数（三角形三个顶点的坐标），函数内调用 drawLine 三次即可。

   画直线需要先将直线两个点转换成 Bresenham 算法可作用的直线——斜率绝对值小于 1、从左往右画，从下往上画。

   比较两点 x 值，将 x 值小的点交换到起点。

   比较两点 y 值，若右点 y 值小于左点，则两点 y 值同时取相反数。

   计算斜率，若绝对值大于 1，则两点各自交换 x、y 值，得到关于 y=x 对称的斜率绝对值小于 1 的点。

   初始化一个数组用于存储坐标信息，其长度需大于等于窗口长边的三倍，并初始化为 0.

   开始使用 Bresenham 算法。

   - **draw** (x_0, y_0)
   - **Calculate** $\Delta x, \Delta y, 2\Delta y, 2\Delta y - 2\Delta x, p_0 = 2\Delta y - \Delta x$
   - **If** $p_i \le 0$ **draw** $(x_{i+1}, \bar{y}_{i+1}) = (x_i + 1, \bar{y}_i)$

     and compute $p_{i+1} = p_i + 2\Delta y$
   - **If** $p_i > 0$ **draw** $(x_{i+1}, \bar{y}_{i+1}) = (x_i + 1, \bar{y}_i + 1)$

     and compute $p_{i+1} = p_i + 2\Delta y - 2\Delta x$
   - **Repeat the last two steps**

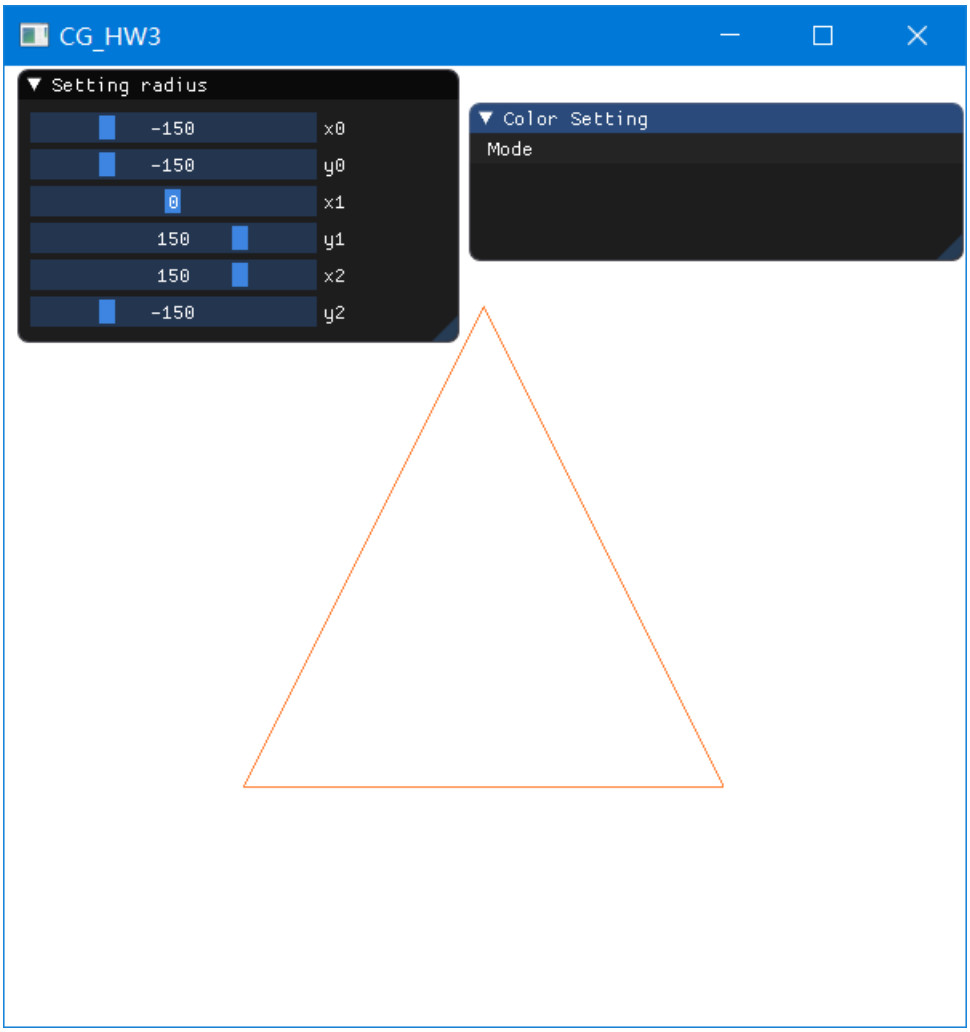
   图自课程 ppt

   每得到一个新的点的坐标，将其根据上面操作进行与否映射回正确的坐标加入数组对应位置（数组不一定填满）。

   加入数组是要对整数转换成浮点数并按照窗口比例缩小为绝对值小于 1 的小数。

   调用函数，根据Δx 的值读取数组的部分内存信息，并用 GL_POINTS 画出Δx 个点。

   画三角形时对三个点两两分别作为参数传入画线函数即可。



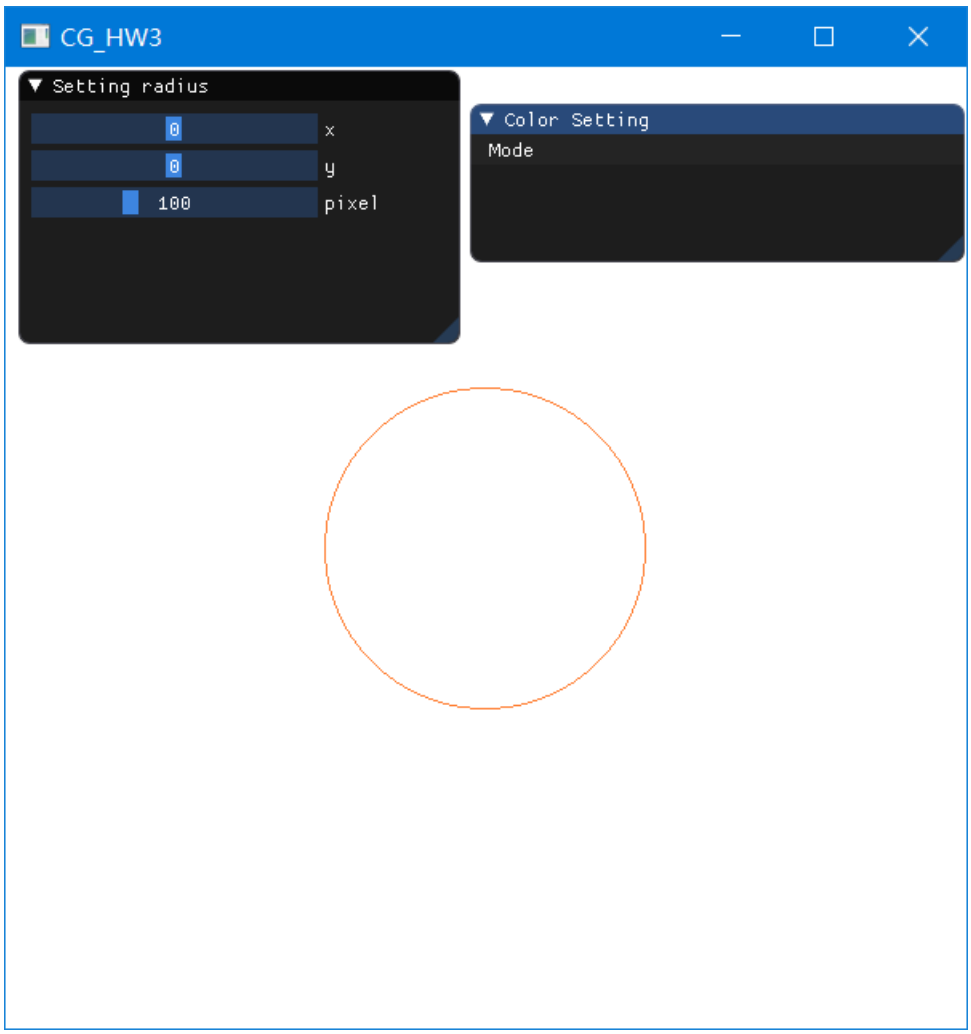
最终实现如上图。可拖动滑条改变点的坐标信息并实时变换三角形。

2. 画圆只需要确定圆 1/8 的点的坐标，即可通过取反、交换等得到整个圆。
首先假设圆心在原点。从（0，r）开始往直线 y=x 方向确定点。
比画直线更简单的是，我们只需要增加 x 的值即可确定下一个点，无需扩展算法进行点的映射变换。
对于 p，我们需要两次近似简化计算过程，第一次近似为从待选两点到圆与竖线交点的距离差近似为待选两点到圆的距离差，第二次减法两边同时平方消根号。
于是我们得到类似画直线的递归方法：

1. 当 $p_i \geq 0$ 时，应选D点，即：
$$y_{i+1} = y_i - 1$$
$$p_{i+1} = p_i + 4(x_i - y_i) + 10$$
2. 当 $p_i < 0$ 时，应选H点，即：
$$y_{i+1} = y_i$$
$$p_{i+1} = p_i + 4x_i + 6$$

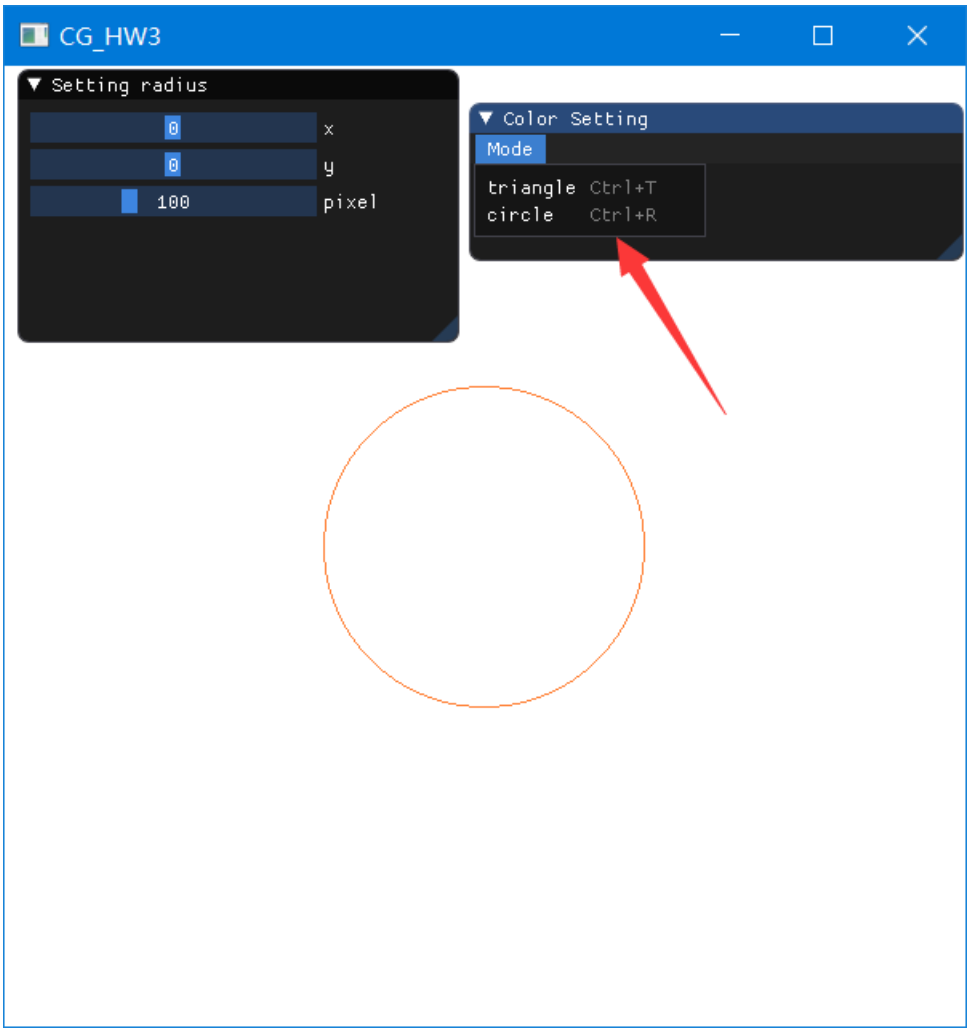
（图自博客 https://blog.csdn.net/sinat_41104353/article/details/82961824）

且 p_0 为 3-2r
于是我们得到 1/8 圆，并通过坐标取反、交换等操作得到完整圆，再对每个点坐标与真正的圆心坐标相加，平移圆得到我们需要的圆。



最终实现如上图。可调整圆心和半径。

3. GUI 如图所示，可切换图形。


CG_HW3
Setting radius
0
x
0
y
100
pixel
Color Setting
Mode
triangle Ctrl+T
circle Ctrl+R