计算机图形与图像 **HW#1**

陈宇轩 **PB16060738**

Question 1

问题描述

使用中点算法推导用于生成具有任何斜率的线段的决策参数，并采用此算法编写程序实现线段的光栅化。

算法分析

先讨论在$0 \le k \le 1$的情况：

1. 直线的一般式方程: $F(x,y)=0$ 若 $F(x,y)>0$, 那么该点在这条直线的上方 若 $F(x,y)=0$, 那么该点在这条直线上 若 $F(x,y)<0$, 那么该点在这条直线下方
2. 由于假定了$0 \le |k| \le 1$，因此，每次在x方向上加1，y方向上加1或不变需要判断。
3. 记M点坐标为(x_m, y_m),代入直线方程为 $F(x_m,y_m)=Ax_m+By_m+C$

   $d_i=F(x_m,y_m)=F(x_i+1,y_i+0.5)=A(x_i+1)+B(y_i+0.5)+C$

   故$y=\begin{cases} y+1 & d<0 \\ y & d\ge 0\end{cases}$
4. 推导 d 的递推公式：

   首先$d_0=F(x_0,y_0)=A(x_0+1)+B(y_0+0.5)+C=A+0.5B$

   由于 $y=\begin{cases} y+1 & d<0 \\ y & d\ge 0\end{cases}$

   则当$d<0$时，$d_{i+1}=A(x_i+2)+B(y_i+1+0.5)+C=Ax_i+By_i+C+2A+1.5B$

   $d_i=A(x_i+1)+B(y_i+0.5)+C=Ax_i+By_i+C+A+0.5B$

   $d_{i+1}-d_i=A+B$

   当$d\ge 0$时，同理得$d_{i+1}-d_i=A$

   即$d_{i+1}=\begin{cases} d_i+A+B & d<0 \\ d_i+A & d\ge 0\end{cases}\ d_0=A+0.5B$
5. 由于只需根据d的符号来进行判断，则可以用2d代替d来拜托浮点运算。
6. 综上得：

   当$0\le k<1$时，$y=\begin{cases} y+1 & d<0 \\ y & d\ge 0\end{cases}\ d_{i+1}=\begin{cases} d_i+2A+2B & d_i<0 \\ d_i+2A & d_i\ge 0\end{cases}\ d_0=2A+B$

同理，可推导到其余斜率范围：

当$-1\le k<0$ 时，$y=\begin{cases} y & d<0 \\ y-1 & d\ge 0\end{cases}\ d_{i+1}=\begin{cases} d_i+2A & d_i<0 \\ d_i+2A-2B & d_i\ge 0\end{cases}\ d_0=2A-B$

当$k>1$时，$x=\begin{cases} x & d<0 \\ x+1 & d\geq 0 \end{cases}$ $d_{i+1}=\begin{cases} d_i+2B & d_i<0 \\ d_i+2A+2B & d_i\geq 0 \end{cases}$ $d_0=A+2B$

当$k<-1$时，$x=\begin{cases} x+1 & d<0 \\ x & d\geq 0 \end{cases}$ $d_{i+1}=\begin{cases} d_i+2A-2B & d_i<0 \\ d_i-2B & d_i\geq 0 \end{cases}$ $d_0=A-2B$

计算A、B的值：

易知$\frac{A}{B}=\frac{y_0-y_1}{x_1-x_0}$ 可将直线方程变形为 $y=-\frac{A}{B}x-\frac{C}{B}$ 由此可令A= $-\Delta y$ B=Δx

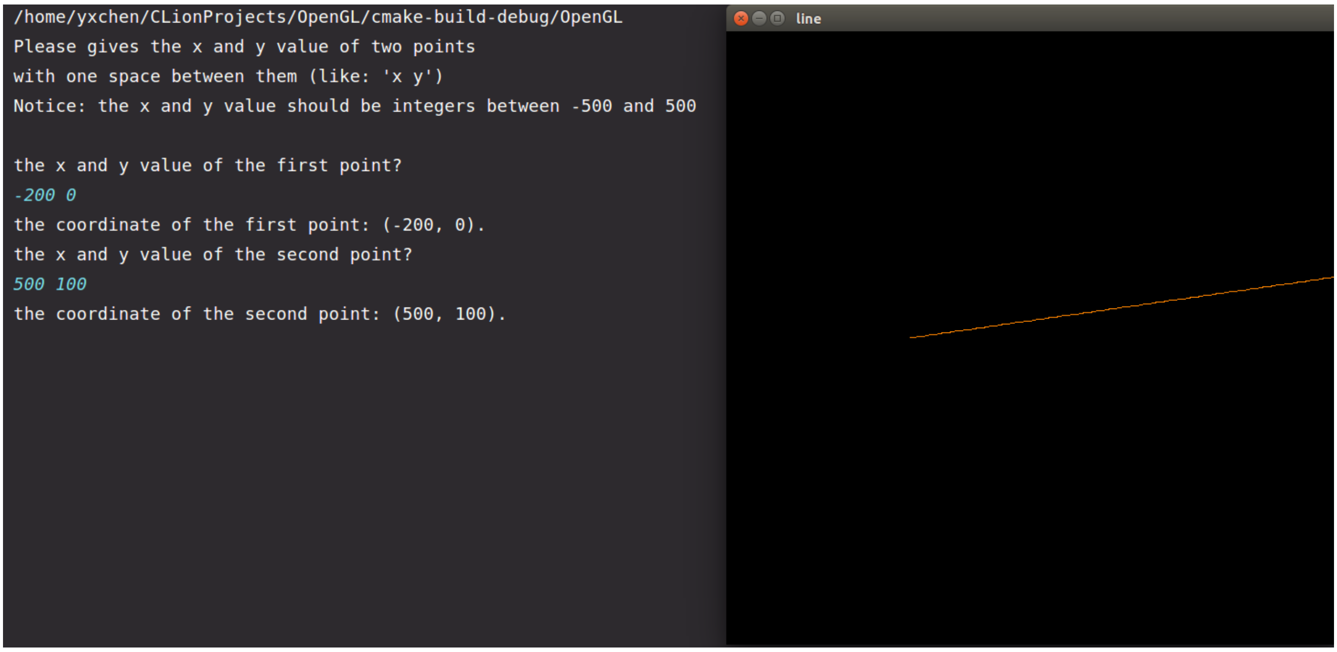
代码实现

使用glfw与OpenGL，编译环境为CLion。

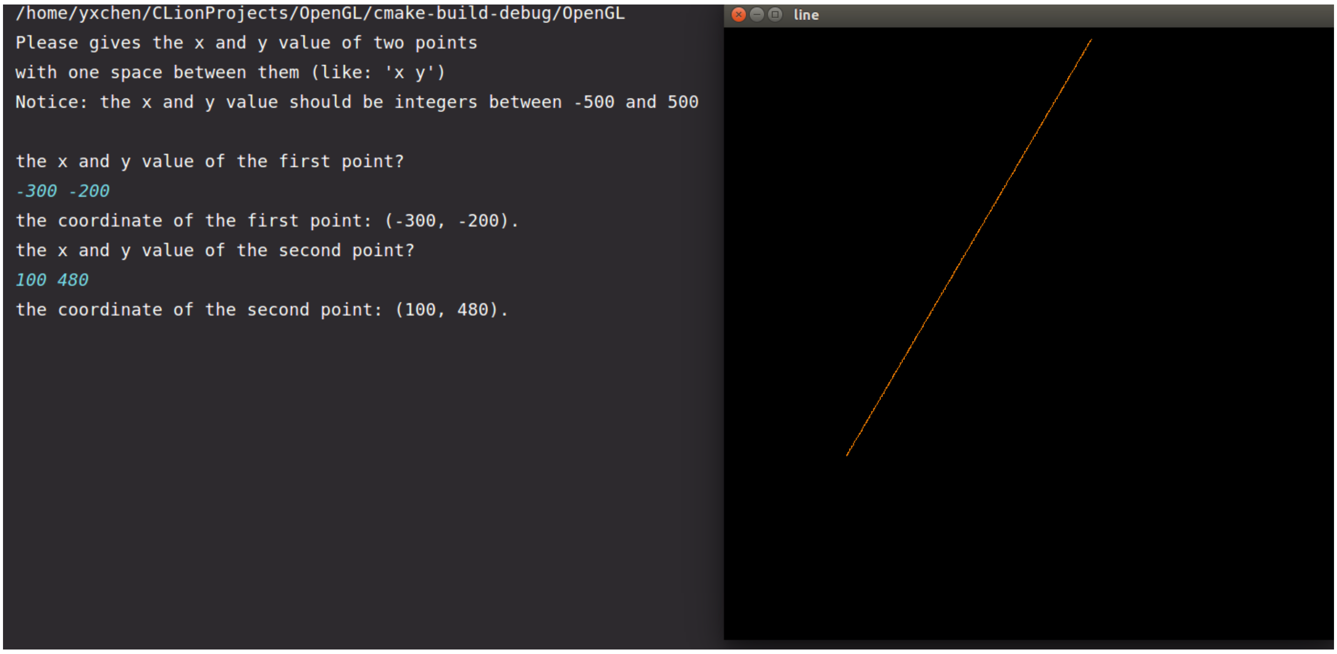
具体代码见附件项目HW1.1。

实验结果

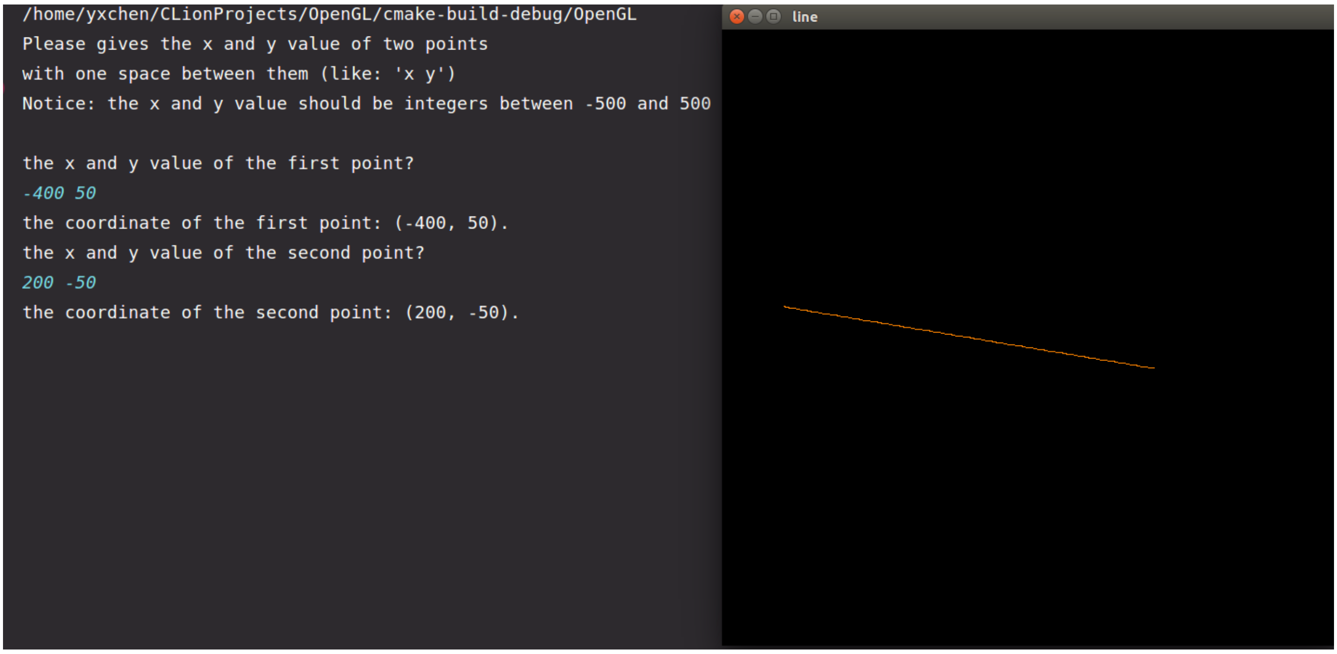
测试样例**1**：(-200,0) (500,100)



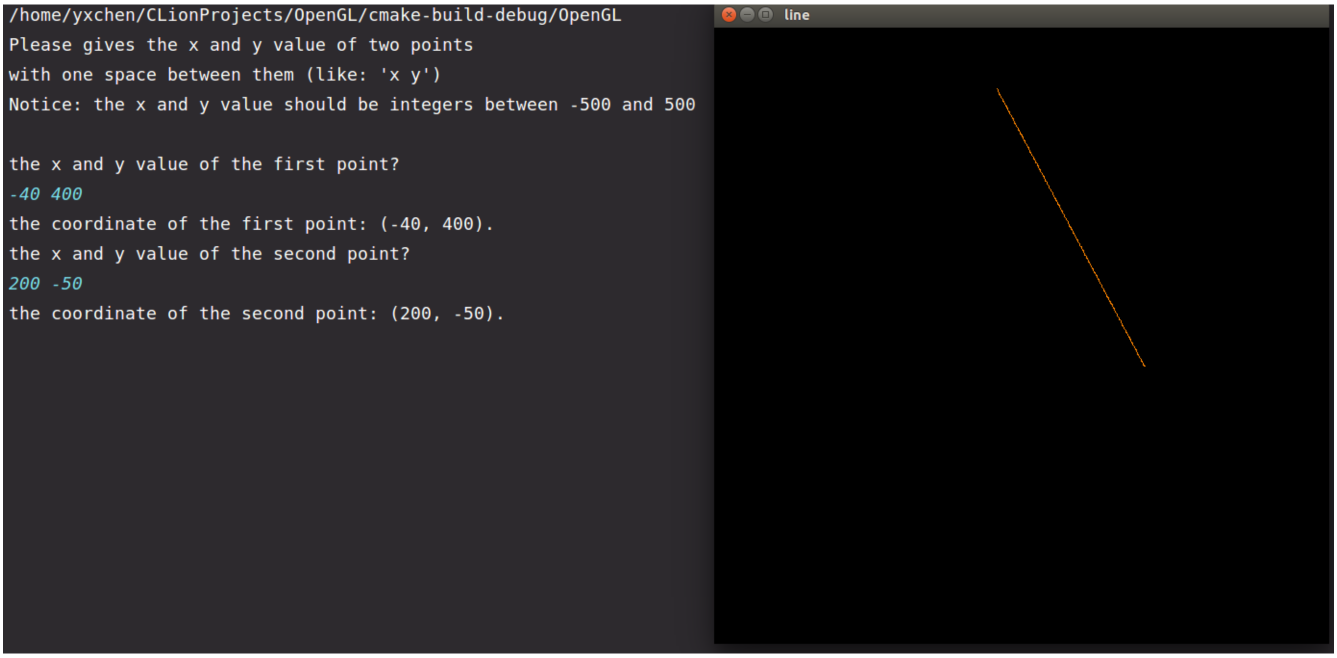
测试样例**2**：(-300,-200) (100,480)



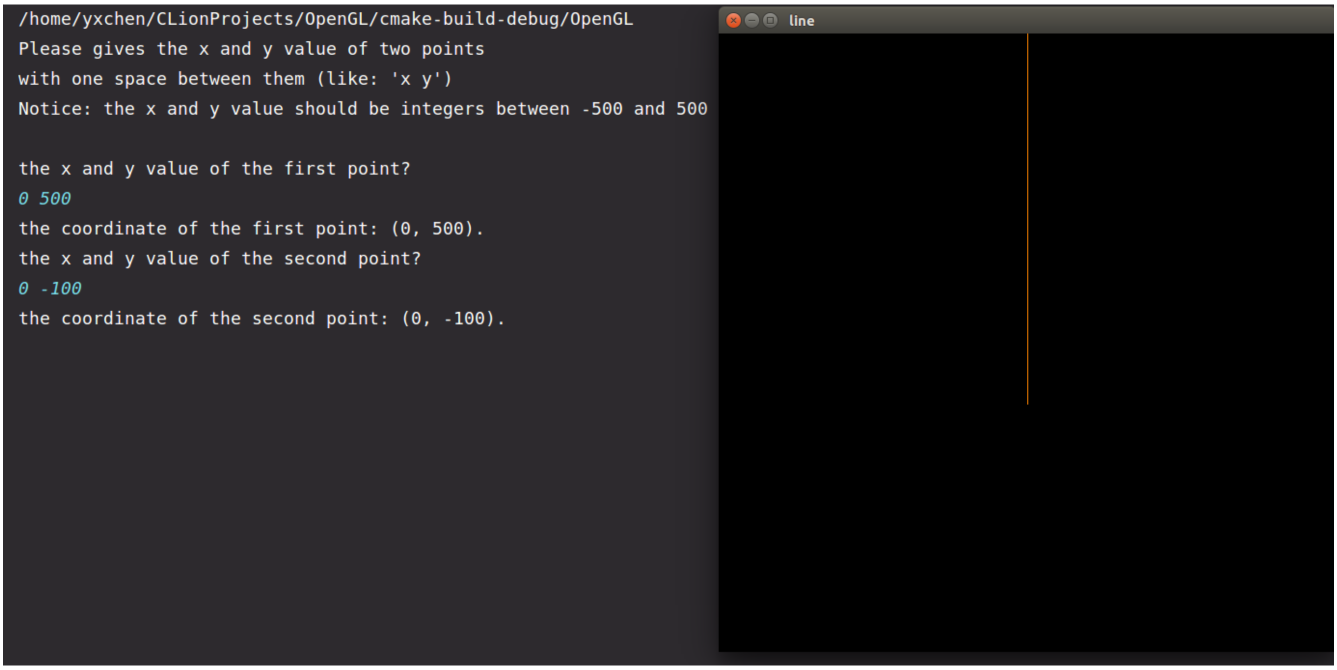
测试样例3：(-400,50) (200,-50)



测试样例4：(-40,400) (0,-300)



测试样例**5**：(0,500) (0,-100)



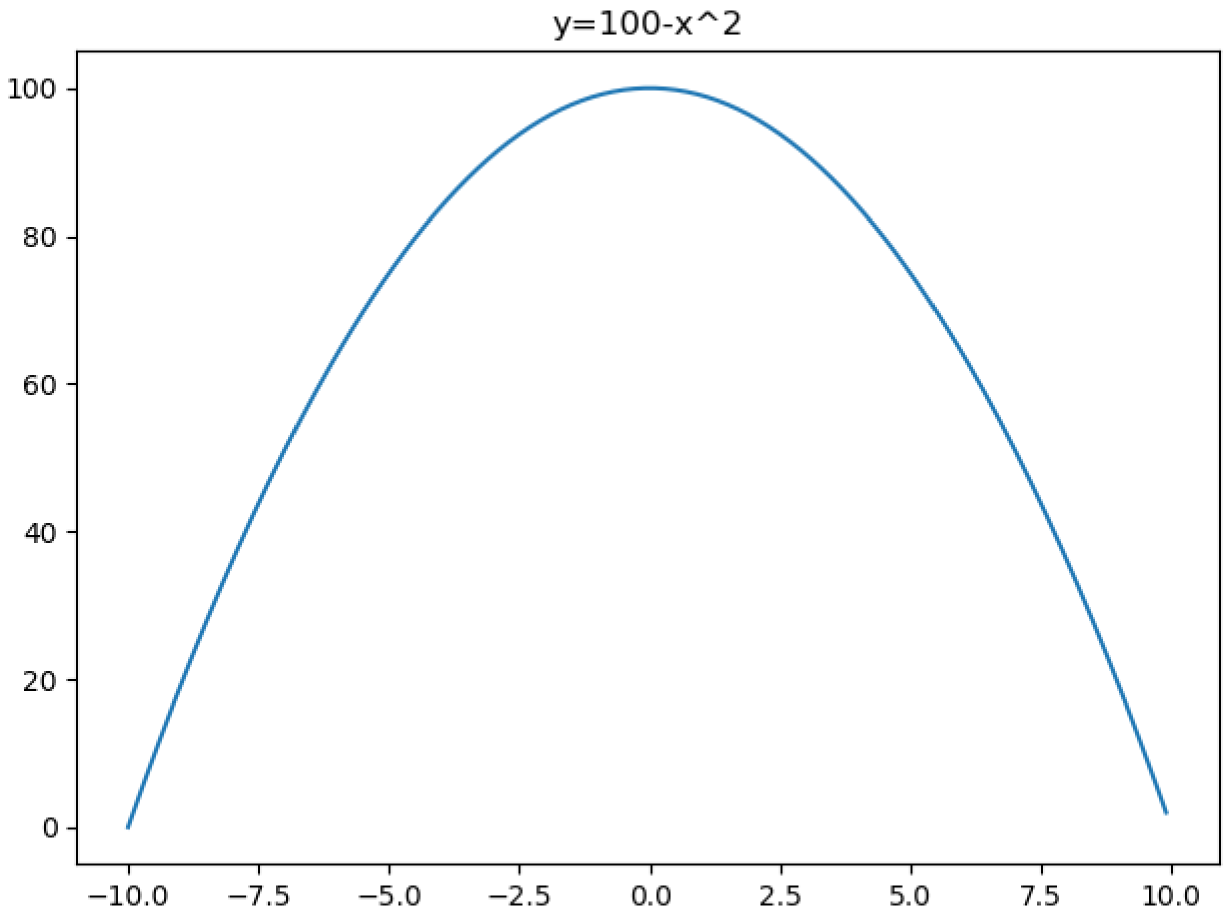
Question 2

问题描述

利用中点算法并考虑对称性，对抛物线 $y = 100 - x^2$ 在区间**【-10,10】**进行扫描转换。

算法分析

考虑对称性，先只考虑第二象限的部分。



再根据斜率k是否大于1分成两部分来考虑。由于$y‘=-2x$ 故x=-0.5处为分界。

设$F(x,y)=x^2+y-100$

1. 当$x\le -0.5$时，$k>1$，$d_i=F(x_m,y_m)=F(x_i+0.5,y_i+1)=(x_i+0.5)^2+(y_i+1)-100$

   $x=\begin{cases} x & d<0 \\ x+1 & d\ge 0 \end{cases}$

   当$d_i<0$时，$\Delta d=F(x_i+0.5,y_i+2)-F(x_i+0.5,y_i+1)=1$

   当$d_i\ge 0$时，$\Delta d=F(x_i+1+0.5,y+2)-F(x_i+0.5,y_i+1)=2x_i+3$

   故$d_{i+1}=\begin{cases} d_i+1 & d_i<0 \\ d_i+2x & d_i\ge 0 \end{cases}$ $d_0=x_0+1.25$

2. 当$-0.5<x\le 0$时,$k\le 1$，$d_i=F(x_m,y_m)=F(x_i+1,y_i+0.5)=(x_i+1)^2+(y_i+0.5)-100$

   $y_{i+1}=\begin{cases} y_i+1 & d<0 \\ y_i & d\ge 0 \end{cases}$

   当$d_i<0$时，$\Delta d=F(x_i+2,y_i+1.5)-F(x_i+1,y_i+0.5)=2x_i+4$

   当$d_i\ge 0$时，$\Delta d=F(x_i+2,y+0.5)-F(x_i+1,y_i+0.5)=2x_i+3$

   故$d_{i+1}=\begin{cases} d_i+2x_i+4 & d_i<0 \\ d_i+2x_i+3 & d_i\ge 0 \end{cases}$ $d_0=2x_0+1.5$

综上，为了避免浮点数运算，可另4d代替d。

代码实现

使用glfw与OpenGL，编译环境为CLion。

具体代码见附件项目HW1.2。

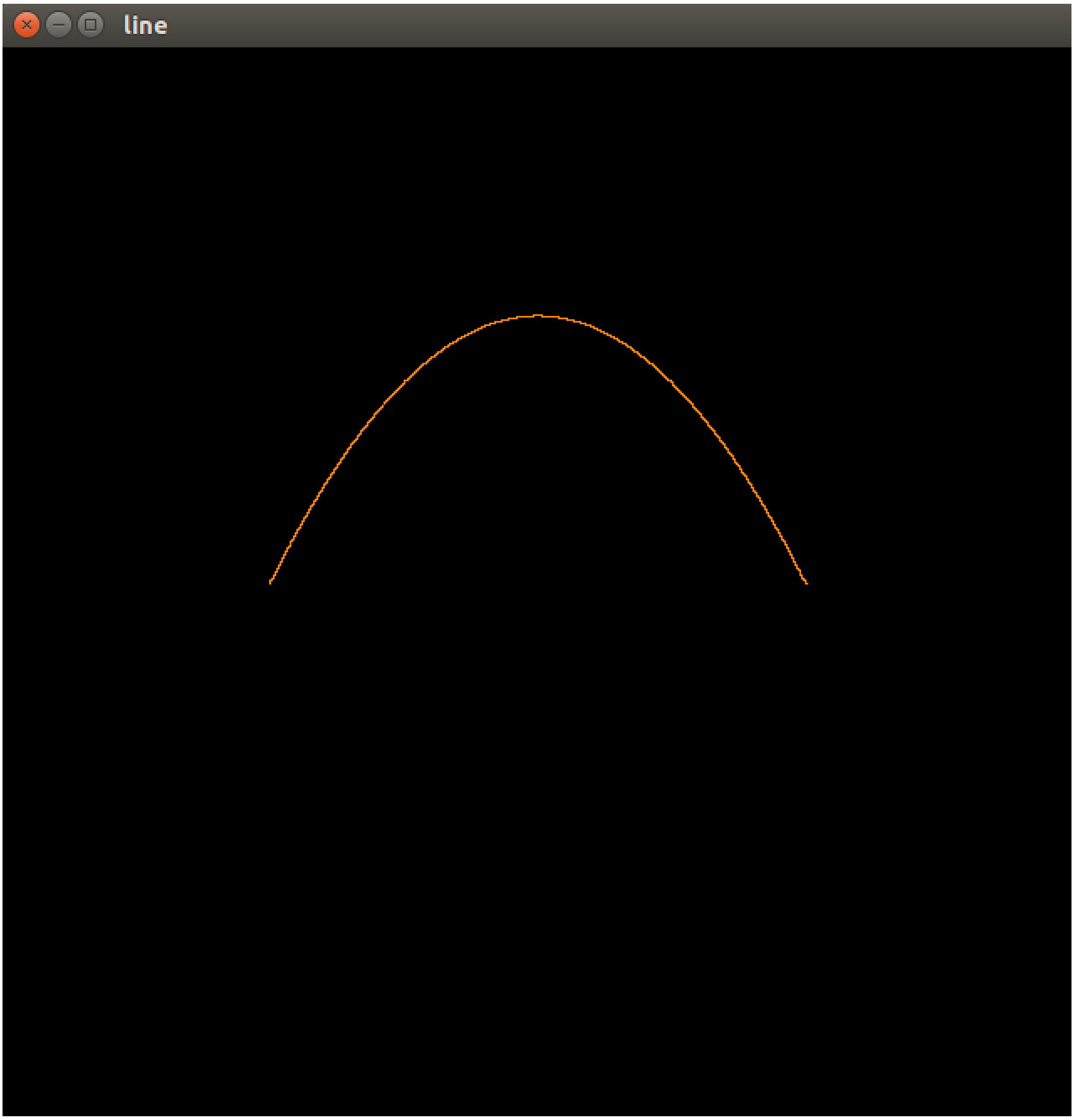
在实际编写过程中，花费了大量时间思考如何处理[-10,10]之间点的坐标比例关系，显然若是按照上面推导公式计算，点会非常稀疏，无法得到好的扫描转换效果。故重新使用0.01的步长，带入上面推导中，得到新的参数关系：

当$x \le -0.5$时，$x = \begin{cases} x & d < 0 \\ x + 0.01 & d \ge 0 \end{cases}$，$d_{i+1} = \begin{cases} d_i + 0.01 & d_i < 0 \\ d_i + 0.02x + 0.0102 & d_i \ge 0 \end{cases}$ $d_0 = 0.01x_0 + 0.010025$

当$-0.5 < x \le 0$时，$y_{i+1} = \begin{cases} y_i + 0.01 & d < 0 \\ y_i & d \ge 0 \end{cases}$，$d_{i+1} = \begin{cases} d_i + 0.02x + 0.0103 & d_i < 0 \\ d_i + 0.02x + 0.0003 & d_i \ge 0 \end{cases}$

$d_0 = 0.02x_0 + 0.0051$

实验结果



最后画图时，按照一定比例缩放。否则由于glfw支持的点坐标只能为[-1,1]，有些点无法画出。

总结

本次实验让我对中点法画图的理解大大加深，对于OpenGL的编程也有了初步认识。其中，第二问花费了大量时间纠结如何处理比例，最后还是回到公式推导。因此在实际编程之前，应该先思考好如何实际执行算法。