5-调试过程和实验结果

运行主程序：

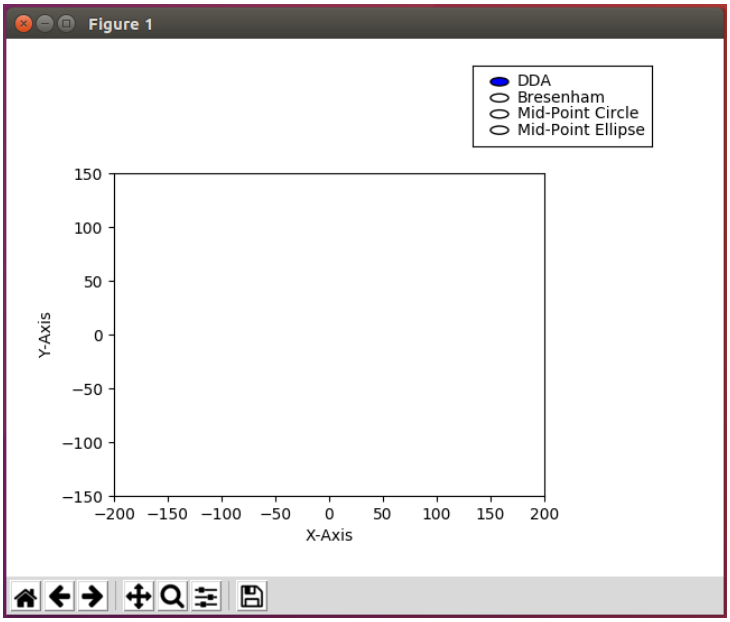

图5-1 绘图程序的主界面

主程序构造了绘图的初始图形界面，其中坐标轴已经绘制，并给出了绘图选项，等待接收输入。

通过鼠标单击绘图选项输入绘图参数：单击之后，将弹出形如图5-2的绘图窗口来接收输入。

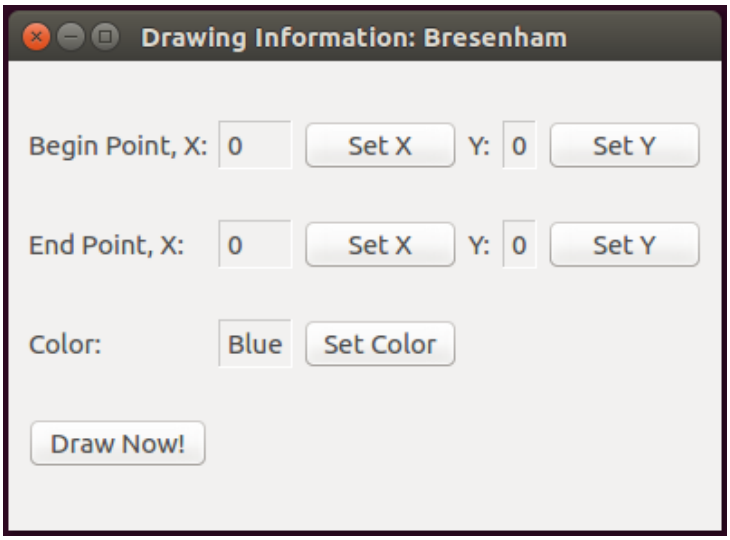

图5-2 接收绘图参数

接收绘图参数后，程序逐点绘制指定图形，并在界面上显现。效果如下所示：

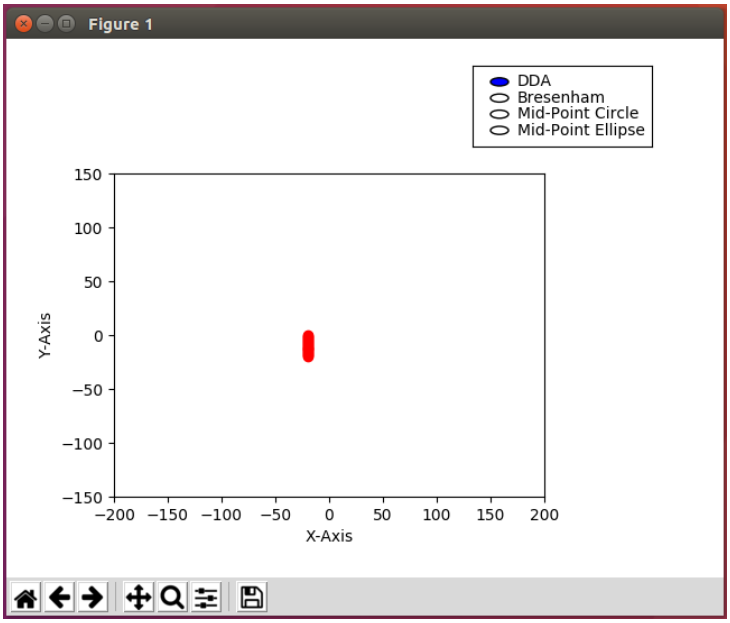

图5-3-a DDA算法的绘制结果（与y轴平行，斜率不存在的线）

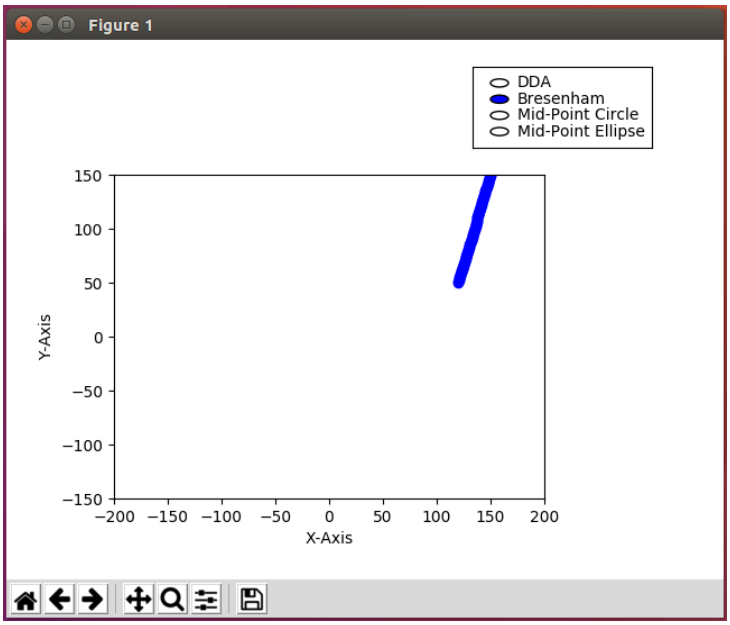

图5-3-b Bresenham算法的绘制结果（直线在边界截断）

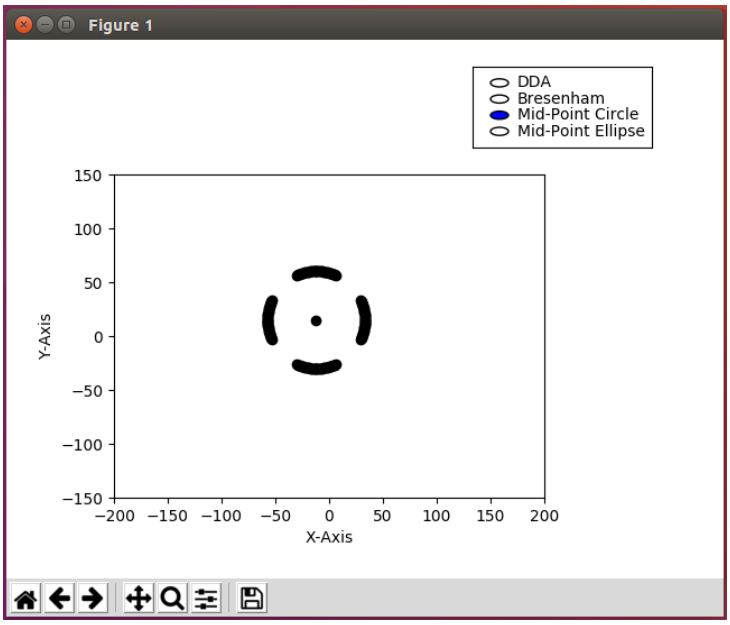

图5-3-c 中点圆算法的绘制结果（图为正在绘制中的过程）

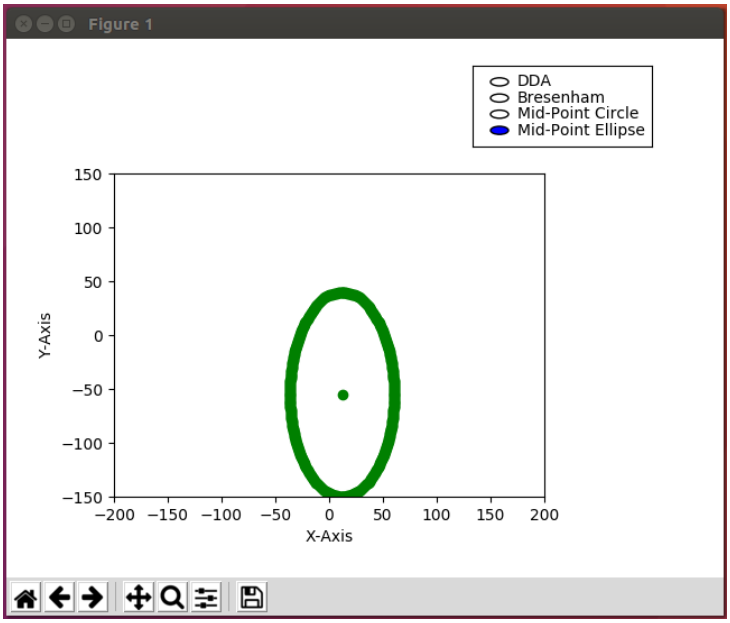

图5-3-d 中点椭圆算法的绘制结果（绘制了椭圆中心，截断濒临边界的点）

绘图程序同样支持在同一界面下绘制多个几何图形，效果如图5-4所示：

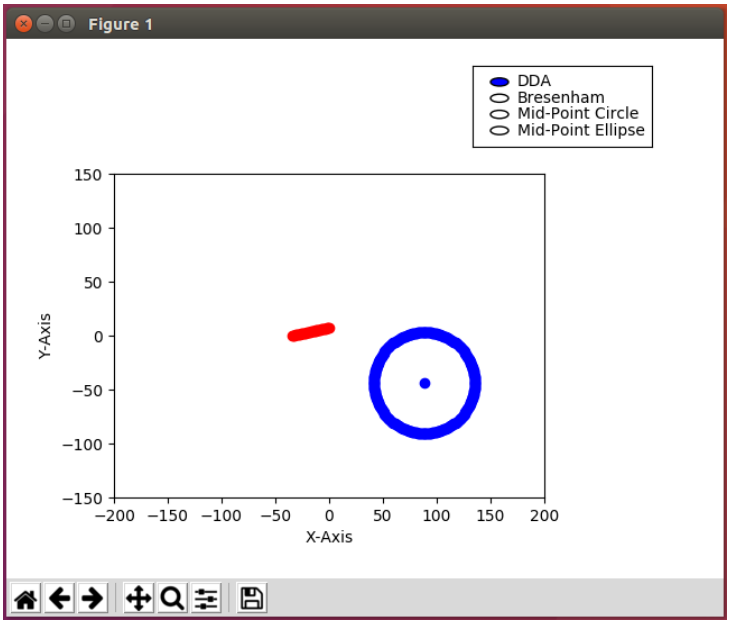

图5-4 绘制多个几何图形的展示效果

6-总结

通过本次实验，我更加熟悉了四种图形绘制算法的原理和实现，并明确了绘制时需要注意的问题。

- 对于DDA和Bresenham算法，以绘制斜率$m \in (0,1)$的情况为例，算法的本质是将整数横坐标上的每一个点的纵坐标四舍五入确定为整数。Bresenham算法实际上优化了这个过程，避免执行$O(n)$次的大浮点数计算和取整，而是将整数和小数分开储存：算法中计算的决策参数p实际对应当前绘制的整数坐标舍入时超出或不足的部分，所以可以根据

p的正负对舍入进行判断。对于斜率为其他或绘制方向不同的情况，可以通过轴对称、中心对称和坐标对称来生成对应的点列。

- 中点圆算法中继承了上述的三处比较重要的思想：一是利用中间点的判断确定绘图点的整数坐标。中点圆算法依然判断半整数点所处的位置来确定绘制哪个点更合理，并把这个特性提炼成为绘制的决策参数。二是利用迭代代替重复计算。类似直线画线算法，中点圆算法中也为每次更新决策参数设计了复杂的机制，以尽量使用整数计算和整数判断，减少计算量。三是充分利用对称性。由圆的八对称性质，中点圆算法只需要计算角坐标$\theta \in [\frac{\pi}{4}, \frac{\pi}{2}]$的部分，而其他部分直接对称绘制。类似地，因为椭圆的性质弱化到四对称，所以中点椭圆算法分别绘制角坐标为$\theta \in [\frac{\pi}{4}, \frac{\pi}{2}]$和$\theta \in [0, \frac{\pi}{4}]$的两个部分，再对称到其他位置。
- 在程序设计方面，我的精力主要花费在图形界面搭建上。图形界面实现的主要难点在于延迟绘制、算法选择和参数输入，我通过查找资料了解了一些实现方案，根据这次作业的需要比对各种实现方案的优劣，最终选择了比较适合的matplotlib和PyQt.在这个过程中，我的自学能力和工程水准得到了进一步的提高。
- 在改进上，目前的设计为了保证安全，在输入参数时进行判断，如果超过画布范围则不能输入。未来实际上可以放宽要求，只检查非法值（比如负的半径或非数值等），使得更大的、局部超过画布边界的图形也允许绘制（只画在画布内的部分），以绘制更多样化的图形。