

图形学系统使用说明书

张昱东

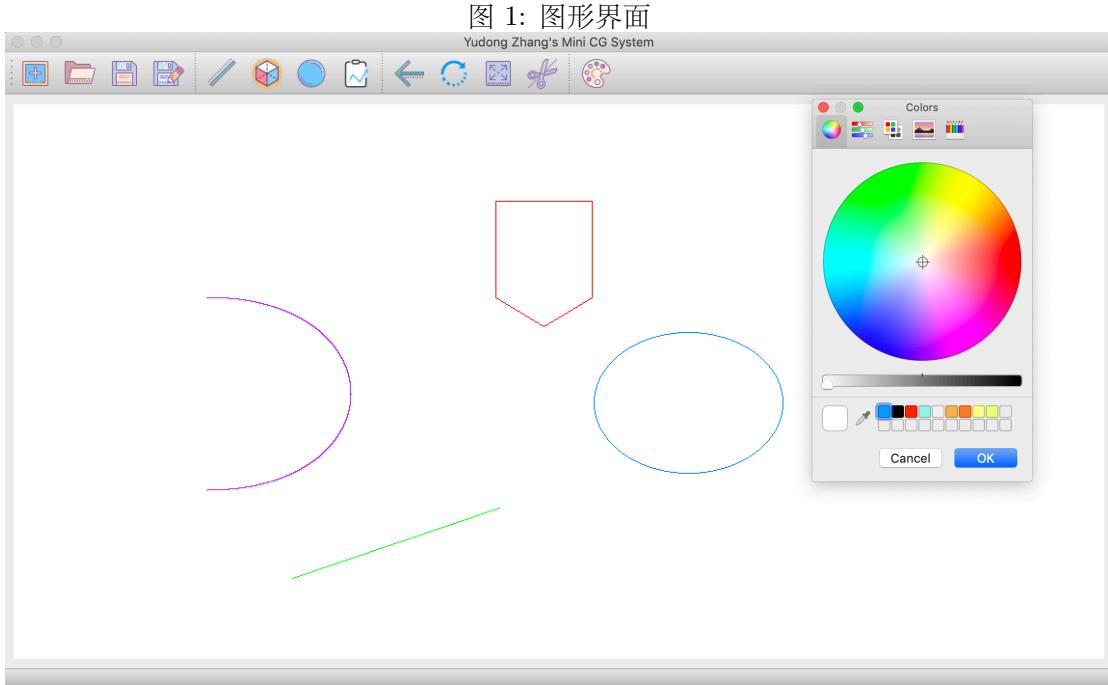
yudongzhang@smail.nju.edu.cn

2019 年 6 月 14 日

目录

1 综述	2
2 开发环境与运行方法	4
3 图形界面程序使用说明	5
3.1 图形界面介绍	5
3.2 新建画布	5
3.3 绘制线段	7
3.4 绘制多边形	8
3.5 绘制椭圆	9
3.6 绘制曲线	10
3.7 平移变换	11
3.8 旋转变换	12
3.9 比例变换	14
3.10 直线剪裁	15
3.11 设置颜色	17
3.12 导入/保存文件	17
4 命令行程序使用说明	19
5 总结	21

1 综述



我较为顺利地完成了实验要求的全部内容，并额外实现了导入图片等功能（如下所述）。设计的图形学系统的图形界面如上图。

核心算法

- 绘制直线: DDA 画线法, Bresenham 画线法, 中点画线法 (额外实现)
- 绘制多边形: DDA 画线法, Bresenham 画线法
- 绘制椭圆: 中点画圆法
- 绘制曲线: Beizer 曲线, B-Spline 曲线
- 二维图形变换: 平移变换, 旋转变换, 比例变换
- 直线剪裁: Cohen-Sutherland 算法, Liang-Barsky 算法

系统模式

- 命令行程序
 - 指令文件的读取与解析
 - 调用算法绘制图形并保存图像
- 图形界面程序
 - 键盘输入交互

其他功能

- 导入/导出图形 (BMP 格式)
- 设置画布大小
- 下拉菜单选择绘制算法
- 弹窗提醒/警告 (规范用户行为, 保证系统的鲁棒性)

2 开发环境与运行方法

开发环境

- 系统说明书和实验报告使用 L^AT_EX 编写。
- 整个图形学系统在 MacOS 10.14 操作系统下开发，使用 Python 3.7.3 编写。
- 使用的 Python 包有 PyQt5, numpy, random, argparse, os, sys，运行程序前需要自行安装。

运行方法

- 运行图形界面程序，在 Source 目录下在命令行中输入 `python gui.py` 后回车。
- 运行命令行程序，在 Source 目录下在命令行中输入 `python clt.py --path "/PATH/TO/SAVE/" --script "/PATH/OF/THE/SCRIPT.txt"` 后回车。
 - 其中 `--path` 参数设置存储图片的位置，如果未设置 `--path`，则默认存储在 `De-mos` 目录下
 - 其中 `--script` 参数设置指令序列文件的位置，以便命令行程序读取指令。
- 为了方便测试命令行程序，我在 CLT_test_scripts 目录下提供了自己写的 7 个测试脚本，另外加上助教提供的 `input.txt` 脚本，可以通过在 Source 目录下命令行中输入 `bash test_clt.sh` 一次性测试所有脚本。导出的图片结果在 CLT_test_scripts 目录的子目录 res 中。
- 如果对于运行程序还有任何疑问，请联系本人。

3 图形界面程序使用说明

图形界面支持键盘交互操作，通过输入点的坐标等参数来进行绘制或变换。注意，在图形界面模式下，创建图元时无需指定 ID，其 ID 根据图元的创建顺序从 0 开始自动生成。要运行图形界面程序，只需在 Source 目录下在命令行中输入 `python gui.py` 后回车即可。

3.1 图形界面介绍

工具栏的图标按钮作用分别为：新建画布，导入图片，保存图片，另存为，绘制直线，绘制多边形，绘制椭圆，绘制曲线，平移变换，旋转变换，比例变换，线段剪切，设置颜色。

图 2: 工具栏



3.2 新建画布

注意，新建画布之后才能进行图元的绘制，若顺序颠倒，会出现弹窗提示操作顺序。

图 3: 点击“新建画布”图标。



图 4: 出现弹窗，是否保存上一张画布



图 5: 若选择保存, 则通过窗口选择保存位置, 保存后再设置新画布大小; 若不保存, 则直接设置新画布大小

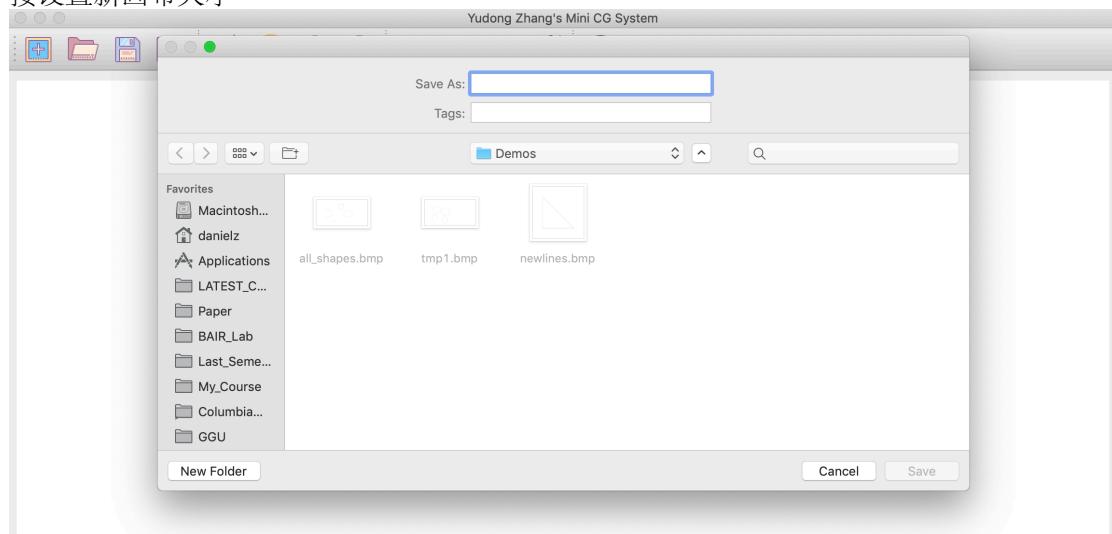


图 6: 设置新画布大小。如果 w 和 h 的设置不满足 $100 \leq w \leq 1000$ 和 $100 \leq h \leq 1000$, 则忽略 w 和 h 的数值而直接将画布填满整个窗口的剩余可用空间。

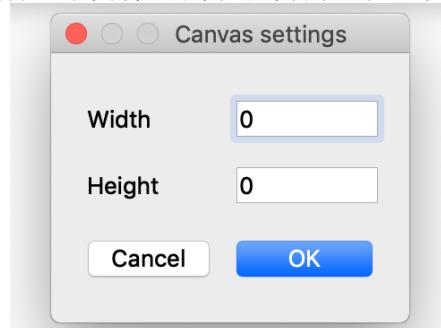
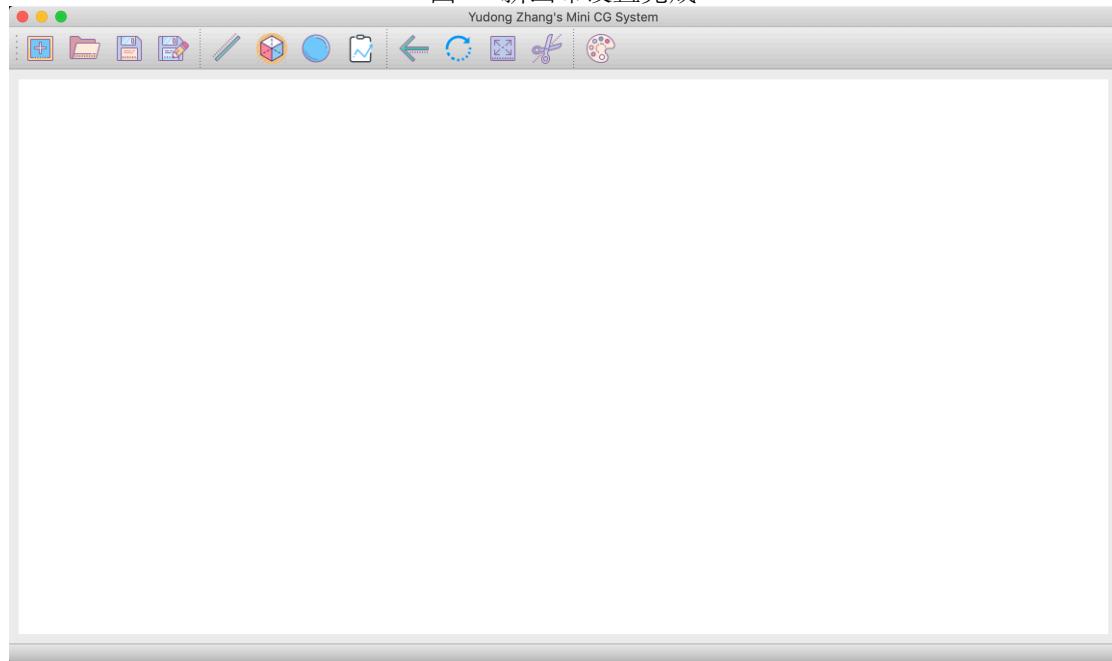


图 7: 新画布设置完成



3.3 绘制线段

图 8: 点击“绘制线段”图标

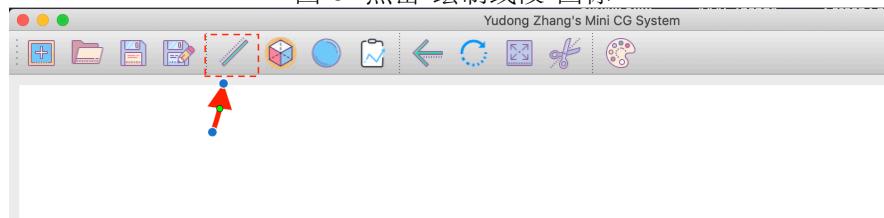


图 9: 键盘输入参数。 (x_1, y_1) 和 (x_2, y_2) 分别为线段的两个端点，顺序不影响绘制效果。可通过下拉菜单选择使用哪种算法绘制，可选算法有：DDA，Bresenham，以及中点画线法。设置完成后点击 OK 按钮，进行绘制

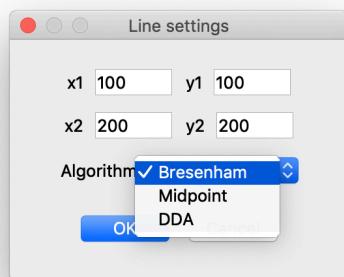
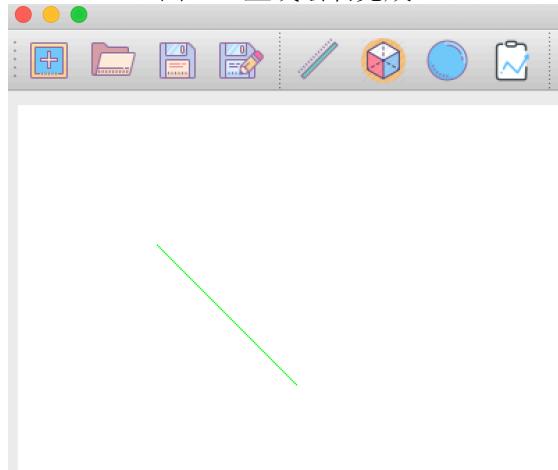


图 10: 直线绘制完成



3.4 绘制多边形

图 11: 点击“绘制多边形”图标

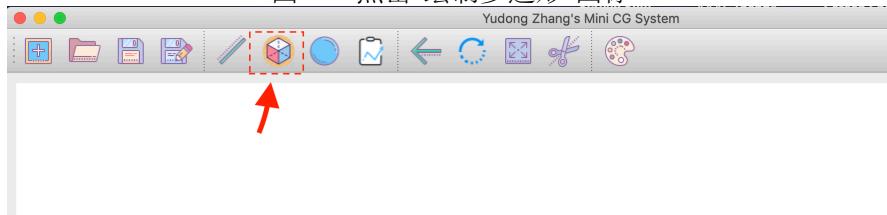


图 12: 键盘输入参数。通过设置 (x,y) 的值并点击 Add 按钮来指定多边形的顶点。其中 n 表示多边形的顶点个数，当已经添加的顶点数达到 n 时，将不能继续添加；此时若修改 n 的数值，则又可以继续添加顶点。可通过下拉菜单来选择使用哪种算法进行绘制，可选算法有 DDA 和 Bresenham。Current Points 下方显示目前已有的顶点，点击 OK 按钮后算法根据顶点的输入顺序进行逐边绘制（因此无法保证多边形是凸多边形）

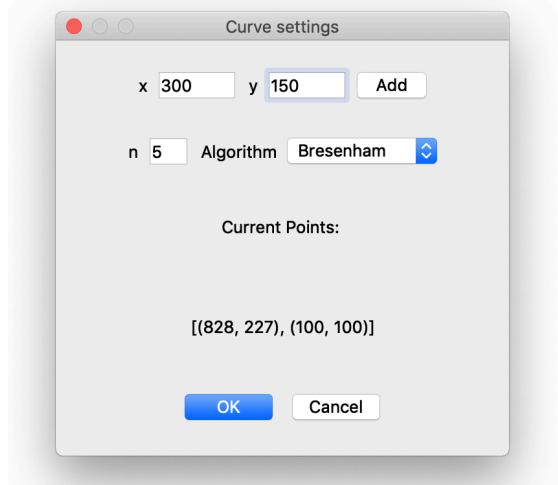
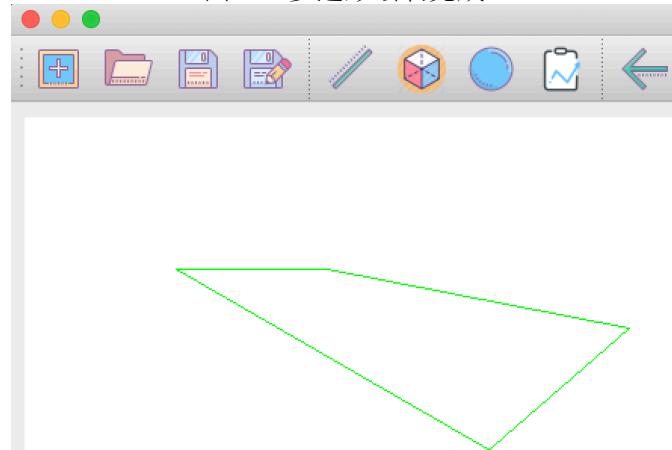


图 13: 多边形绘制完成



3.5 绘制椭圆

图 14: 点击“绘制椭圆”图标

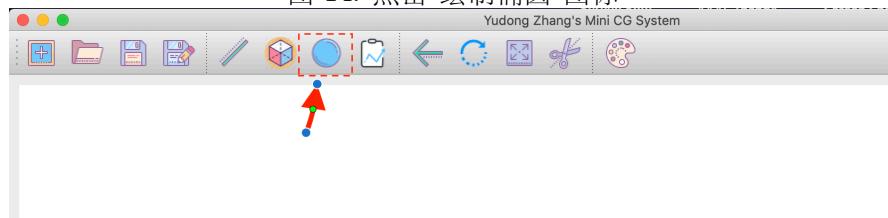


图 15: 键盘输入参数。 (x,y) 表示椭圆的中心坐标, rx 和 ry 表示椭圆在横轴和竖轴的两个半径。由于只实现了中点圆算法, 则可选算法只有这一个。设置完成后, 点击 OK 按钮进行椭圆的绘制。

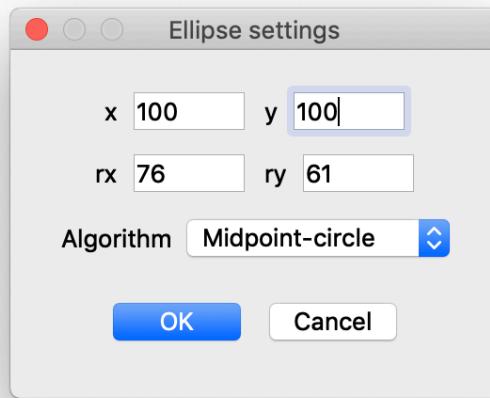
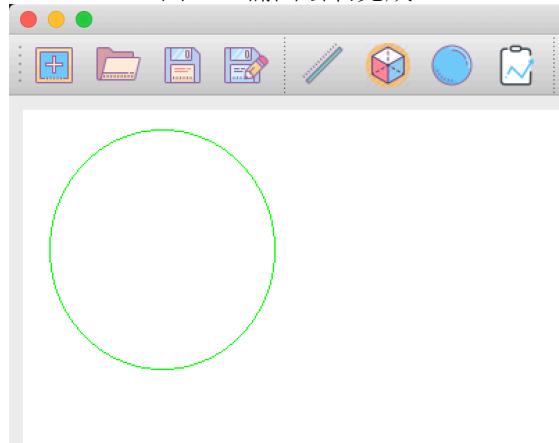


图 16: 椭圆绘制完成



3.6 绘制曲线

图 17: 点击“绘制曲线”图标



图 18: 键盘输入参数。类似多边形绘制，通过设置 (x,y) 的值并点击 Add 按钮来指定曲线的控制点。其中 n 表示控制点的个数，当已经添加的控制点个数达到 n 时，将不能继续添加；此时若修改 n 的数值，则又可以继续添加。可通过下拉菜单来选择需要绘制的曲线类型，可选类型有 Beizer 和 B-Spline。Current Points 下方显示目前已有的控制点，点击 OK 按钮后算法开始绘制曲线

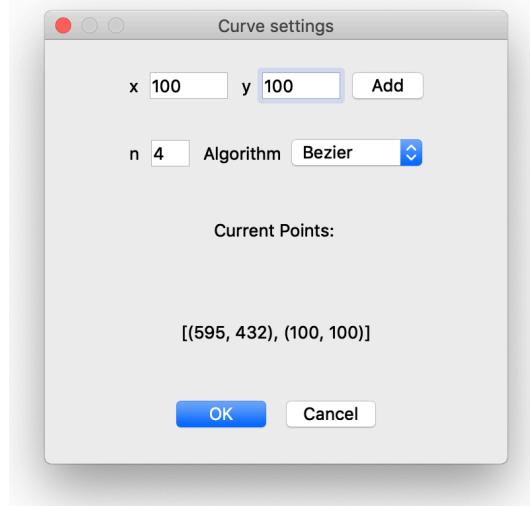
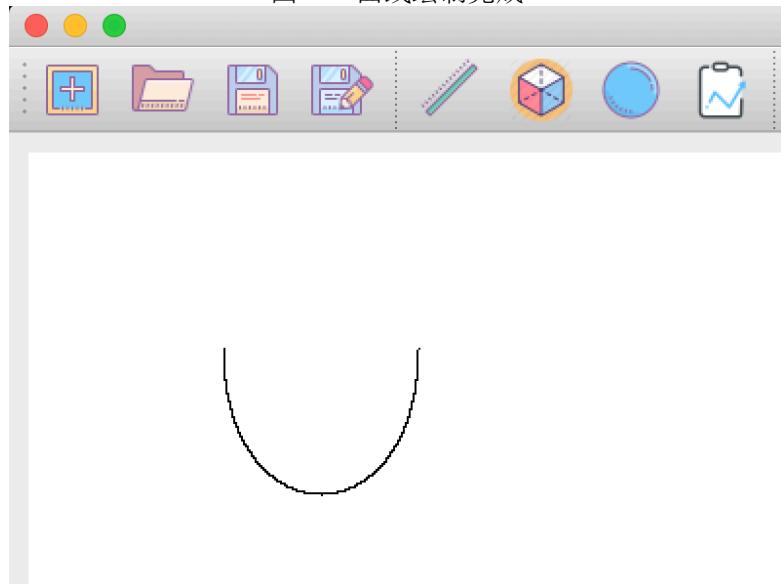


图 19: 曲线绘制完成



3.7 平移变换

图 20: 原始图像

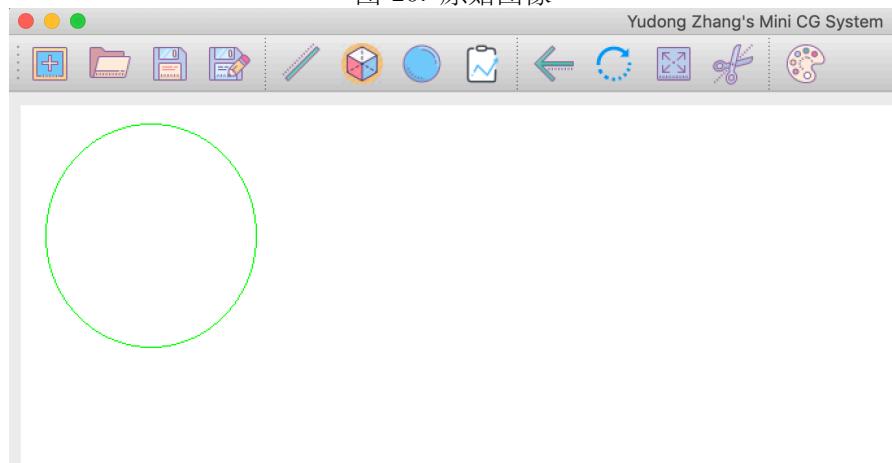


图 21: 点击“平移变换”图标



图 22: 键盘输入参数。分别为图元 ID, 横轴平移距离, 和纵轴平移距离。点击 OK 按钮开始剪裁。

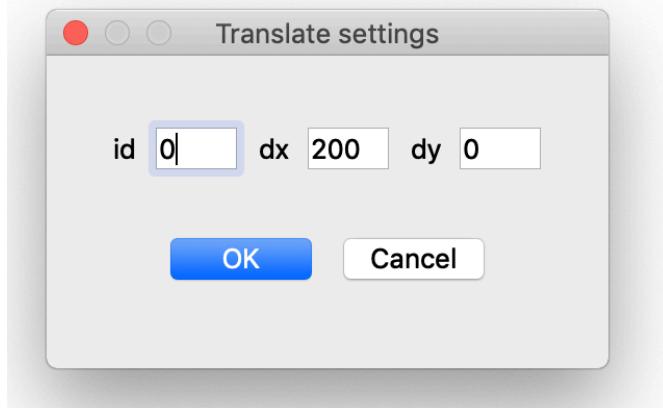
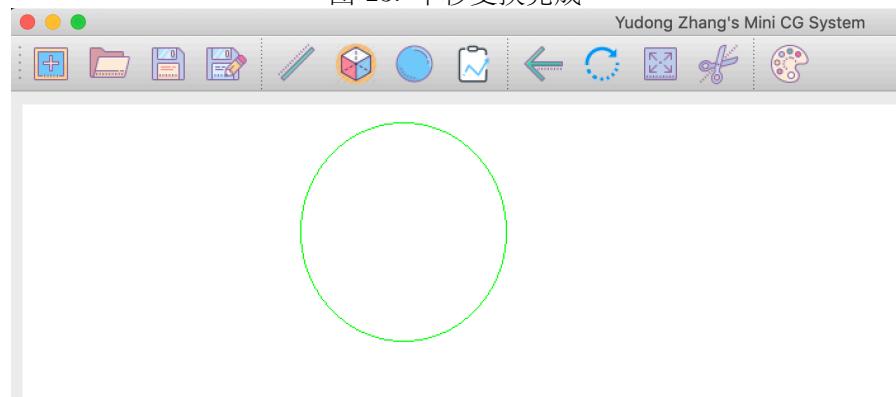


图 23: 平移变换完成



3.8 旋转变换

图 24: 原始图像

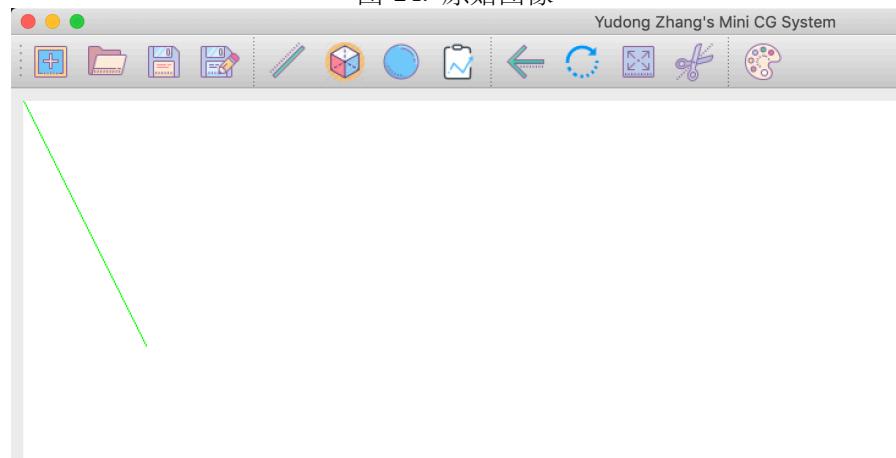


图 25: 点击“旋转变换”图标



图 26: 键盘输入参数。分别为图元 ID, 旋转角度, 旋转中心的坐标 (x,y)。点击 OK 按钮开始剪裁。

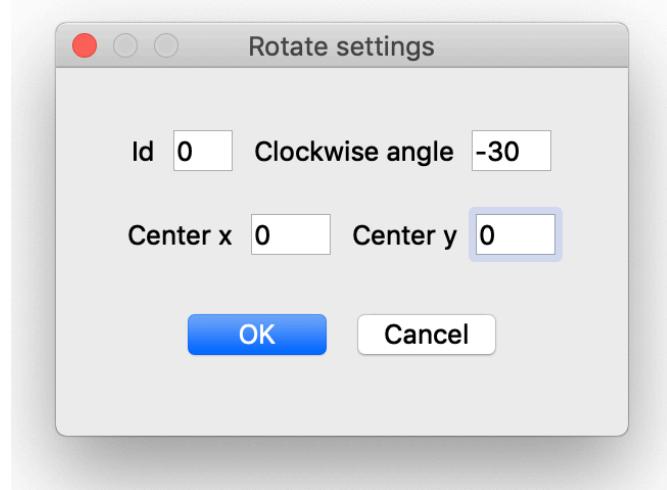
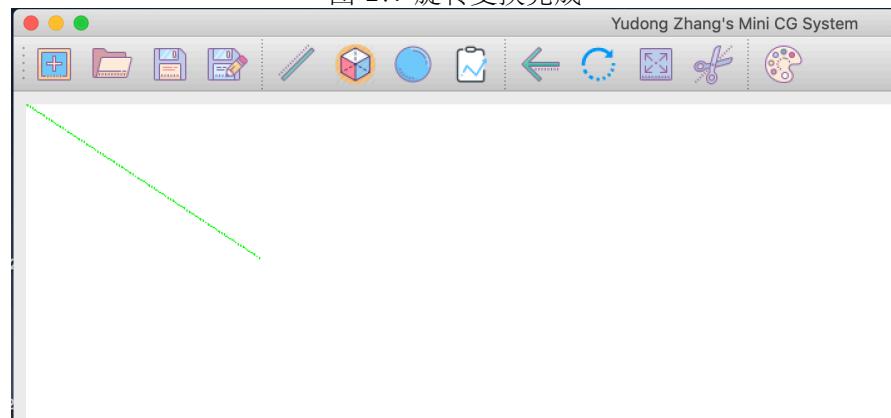


图 27: 旋转变换完成



3.9 比例变换

图 28: 原始图像

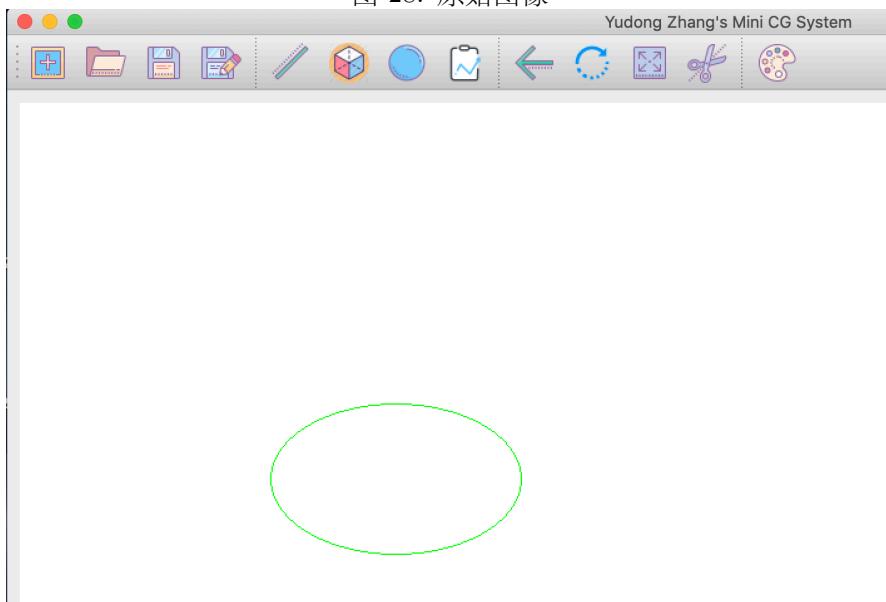


图 29: 点击“比例变换”图标



图 30: 键盘输入参数。分别为图元 ID, 缩放比例, 缩放中心的坐标 (x,y)。点击 OK 按钮开始剪裁。

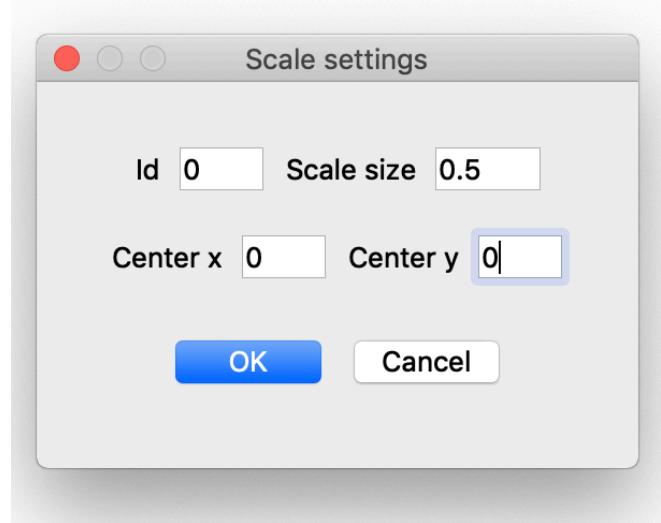
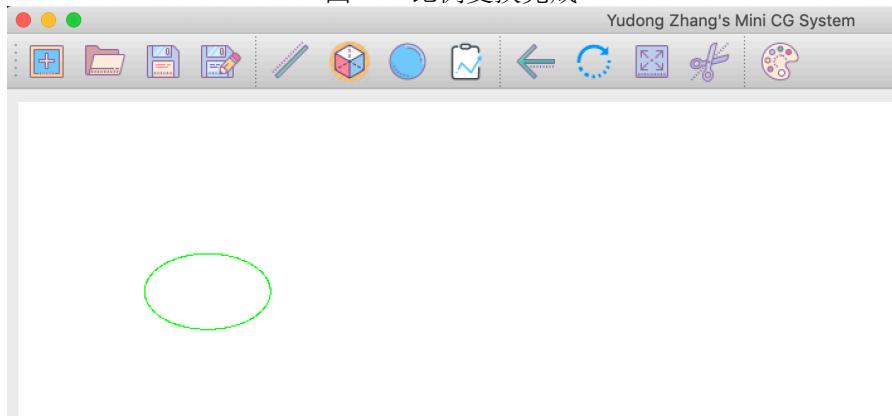


图 31: 比例变换完成



3.10 直线剪裁

图 32: 原始图像

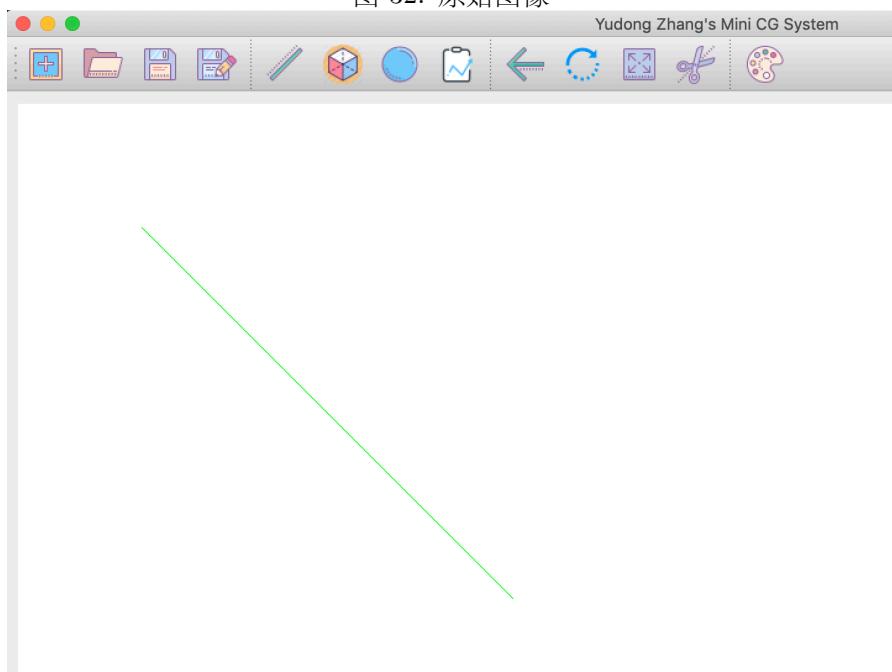


图 33: 点击“直线剪裁”图标

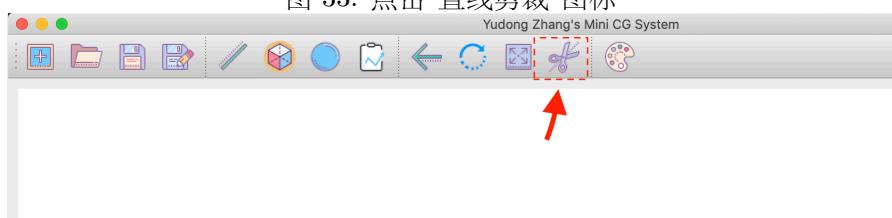


图 34: 键盘输入参数。分别为图元 ID, 剪裁窗口的左上角坐标 (x_1, y_1) 和右下角坐标 (x_2, y_2)。可通过下拉菜单选择线段剪裁使用的算法, 可选算法有 Cohen-Sutherland 和 Liang-Barsky。点击 OK 按钮开始剪裁。

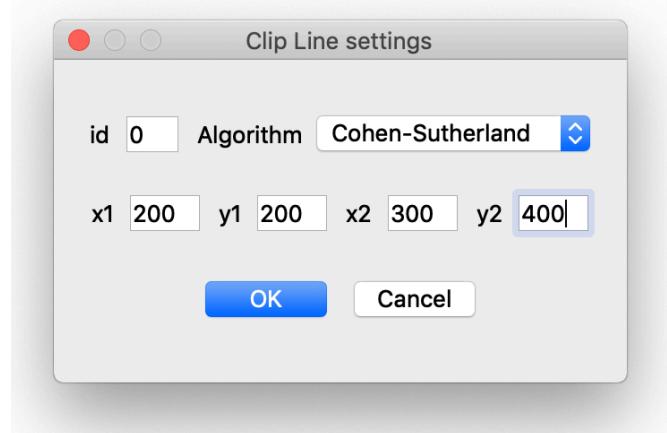
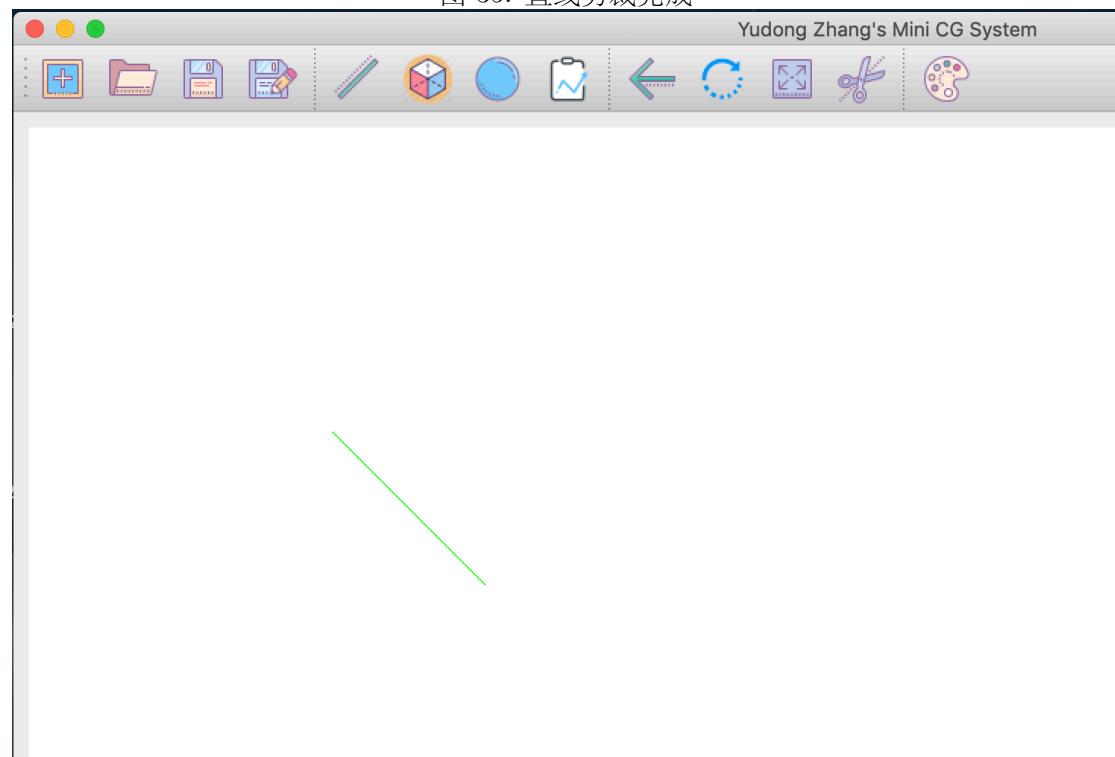


图 35: 直线剪裁完成



3.11 设置颜色

图 36: 点击“设置颜色”图标

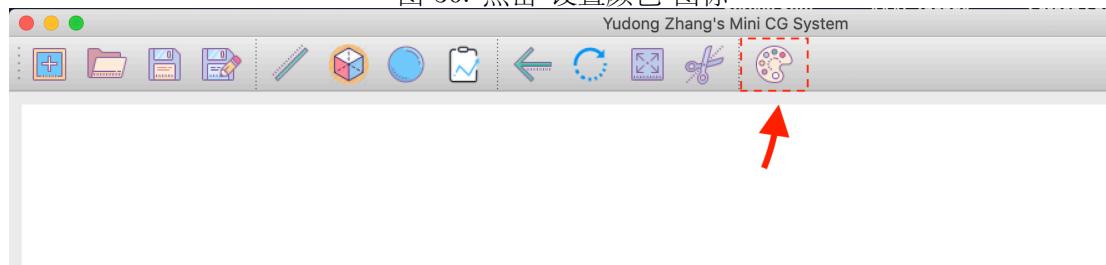
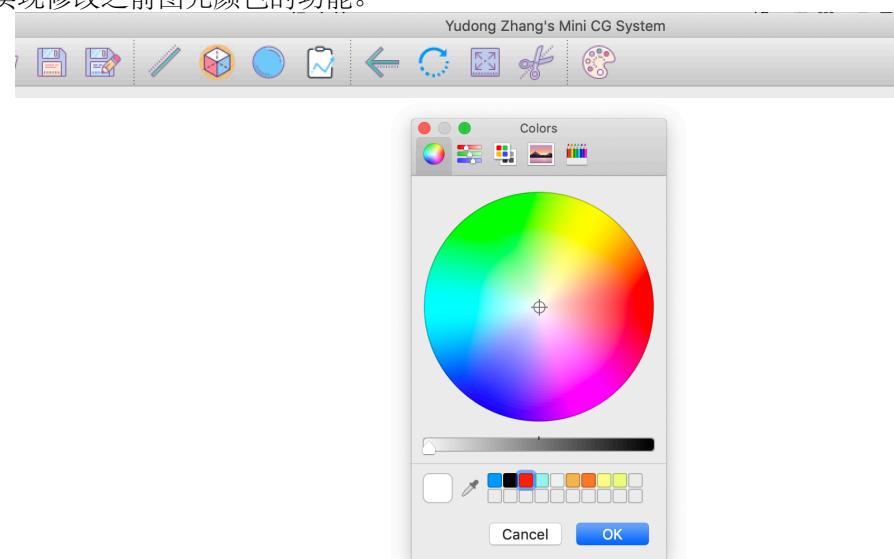


图 37: 鼠标选择颜色后, 点击 OK 后接下来绘制的图元将显示当前选择的颜色。注意, 目前尚未实现修改之前图元颜色的功能。

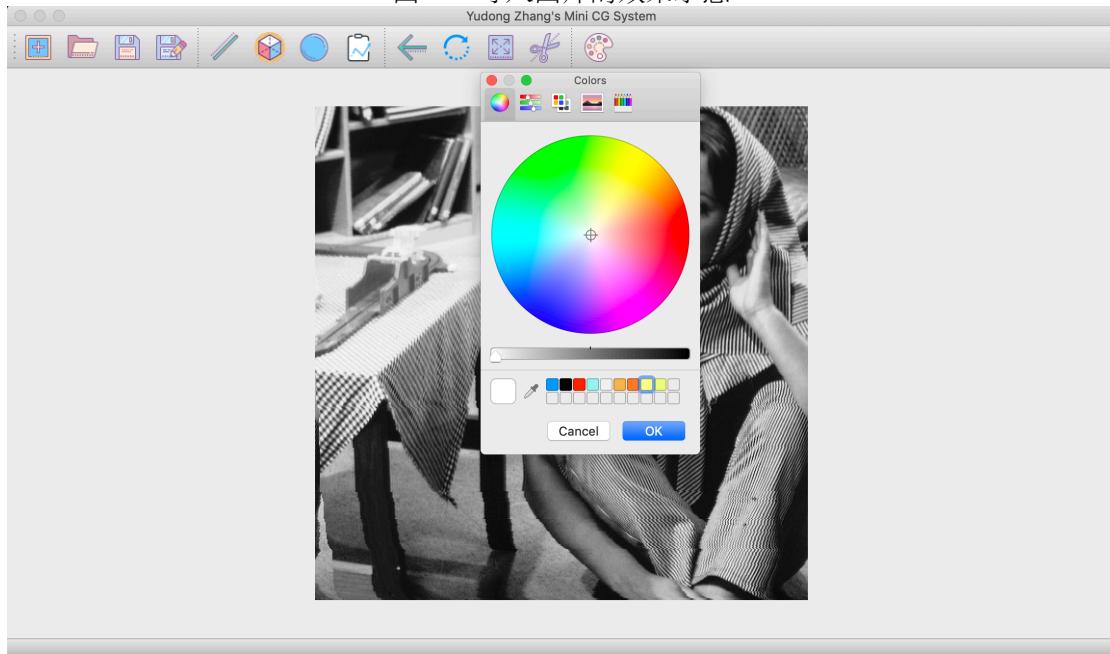


3.12 导入/保存文件

图 38: 图中所示三个图标分别为: 导入图片, 保存图片, 以及另存为。由于操作与大部分软件都一样, 因此在这里不再赘述



图 39: 导入图片的效果示意



4 命令行程序使用说明

命令行程序读取和解析指令序列文件，然后逐条执行命令。在运行过程中不会出现图形界面，运行完后系统会自动退出。。在使用命令行程序时，需要在指令中指定图元的 ID。

要运行命令行程序，只需在 Source 目录下在命令行中输入 `python clt.py --path "/PATH/TO/SAVE/" --script "/PATH/OF/THE/SCRIPT.txt"` 后回车即可。

我在 CLT_test_scripts 目录下提供了自己写的 7 个测试脚本，另外加上助教提供的 `input.txt` 脚本，可以通过在 Source 目录下命令行中输入 `bash test_clt.sh` 一次性测试所有脚本（绘制直线、多边形、椭圆，平移变换，旋转变换，比例变换，线段剪切，设置颜色等操作）。导出的图片结果在 CLT_test_scripts 目录的子目录 res 中。

图 40: 命令行程序运行 log

```
(base) ➔ Source git:(master) ✘ bash test_clt.sh
Test CLT mode

python clt.py --path "../CLT_test_scripts/res/" --script "../CLT_test_scripts/test_new_line.txt"
New Element: {'ID': 0, 'type': 'Line', 'point1': (100, 100), 'point2': (400, 400), 'algorithm': 'DDA'}
New Element: {'ID': 1, 'type': 'Line', 'point1': (100, 100), 'point2': (100, 400), 'algorithm': 'Mid point'}
New Element: {'ID': 2, 'type': 'Line', 'point1': (100, 400), 'point2': (400, 400), 'algorithm': 'Bresenham'}
Image successfully saved at directory: /Users/danielz/Documents/Last_Semester/ComputerGraphics/Project/CLT_test_scripts/res/test_new_line.bmp

python clt.py --path "../CLT_test_scripts/res/" --script "../CLT_test_scripts/test_new_polygon.txt"
New Element: {'ID': 0, 'type': 'Polygon', 'points': [(100, 100), (100, 200), (200, 200), (200, 100)], 'algorithm': 'Bresenham'}
New Element: {'ID': 1, 'type': 'Polygon', 'points': [(400, 400), (450, 350), (500, 400), (500, 500), (450, 550), (400, 500)], 'algorithm': 'DDA'}
Image successfully saved at directory: /Users/danielz/Documents/Last_Semester/ComputerGraphics/Project/CLT_test_scripts/res/test_new_polygon.bmp
```

图 41: 命令行程序测试结果 1。我自己写的测试脚本运行结果（由于画布设置太大，所以测试出来的结果线条看起来比较细）

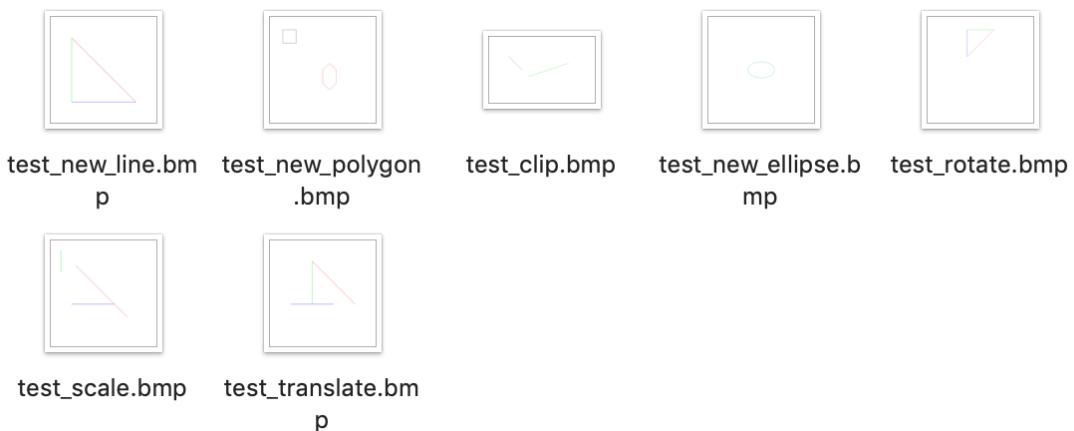


图 42: 命令行程序测试结果 2。助教提供的测试脚本在我的程序上运行的结果

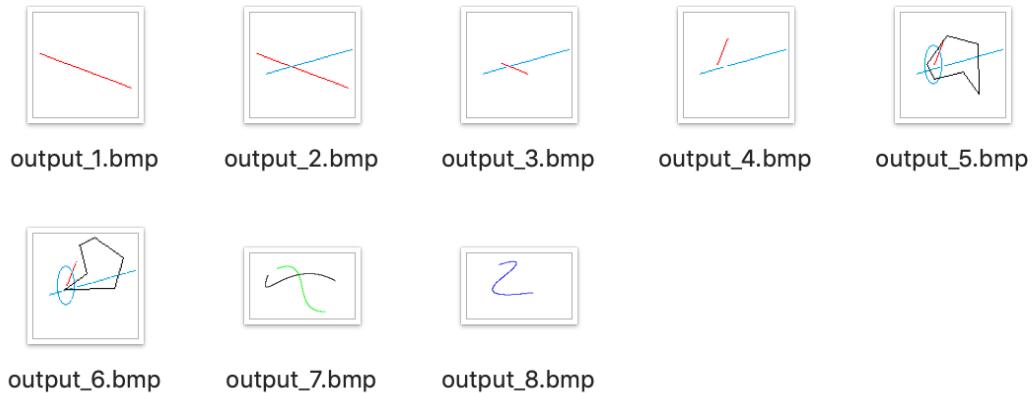
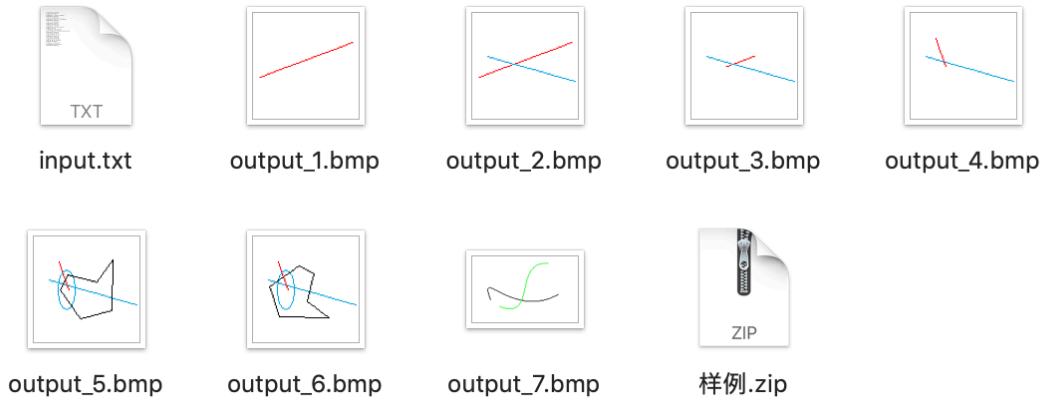


图 43: 标准样例。助教提供的测试脚本在标准程序上运行的结果



我们可以观察到，测试结果 2 和标准输出有一定的差异。

第一点是图元的方向，这是由于标准程序定义左下角为原点，y 轴向上，x 轴向右；而我的程序定义的是左上角为原点，y 轴向下，x 轴向右，因此方向是相反的。

第二点是我的程序在对图元进行平移时，由于不能撤销之前已经绘制的图元，因此只能通过将颜色设置成白色然后再画一遍，从而“消除”原来的图形，再绘制平移后的图形。这样的坏处是和别的图元的重叠部分可能会被消除，导致图元有一些缺陷（我尚未找到更好的方法）。

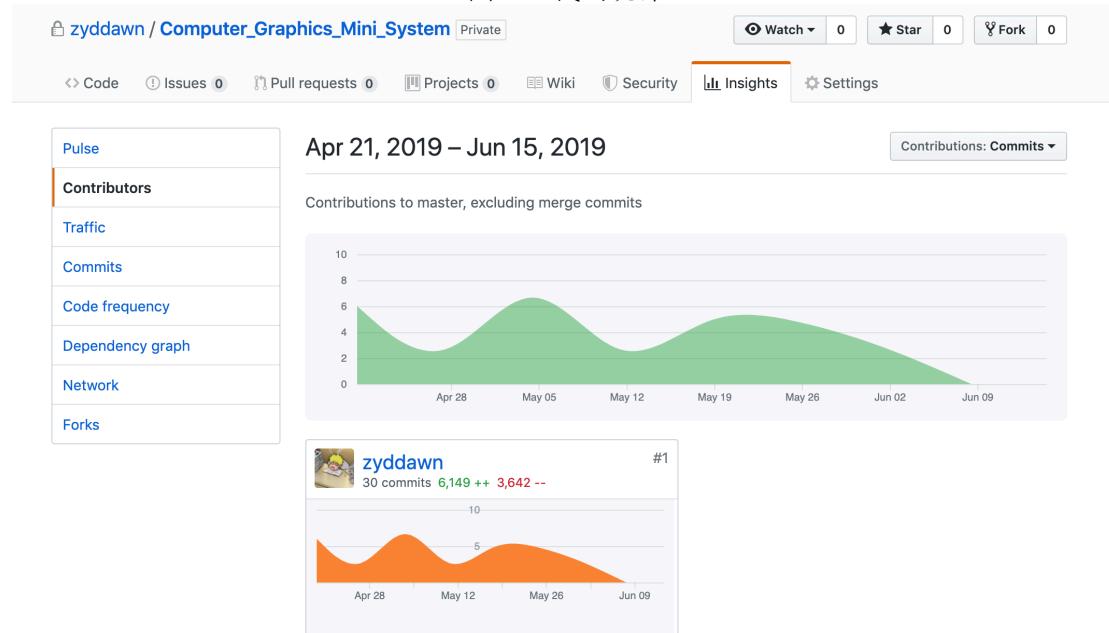
但是整体上两个结果是大致相同的，也说明了我的程序绘制的图元是正确的。

5 总结

在实现这个系统的过程中，我自己重新推导了算法中的公式以及使用代码将书上文字描述的绘制过程实现。经过这些锻炼后，我掌握了许许多多图形学的知识与算法，包括直线与圆的生成算法，直线裁剪算法、二维变换算法、Bezier 曲线、椭圆生成算法等等……同时，我也对如何有机组合这些图形操作有了自己的认知。

通过对 PyQt5 文档的仔细研读和实践，以及对整个系统框架的精心设计，我花费一个多月的时间成功从零开始实现了一个健壮的、自洽的图形操作软件，期间共写了 6000+ 行 Python 代码。这是我第一次使用 PyQt5 来设计图形界面程序，记得最开始时不管是新建画布还是绘制直线，得到的都是一串奇怪的 Error，而界面上什么变化也没有出现；还有自己在实现中点圆算法时，由于一个变量的符号错误，导致一直不能画出正确的圆弧；以及在实现曲线绘制时，由于久久不能理解书中的算法而异常苦恼。但是这些曾经的痛苦换来的是我现在对 PyQt5 的熟练掌握，以及对图形学算法更深的理解，感谢图形学课程提供给我的这一次机会，让我在 bug 中挑战自我，锻炼了我的 code 能力，也增强了我对图形学这一学科的理解，让我能够不断提高自己的能力。

图 44: 代码统计



由于临近期末时间紧迫，这个系统尚有很多不完美之处，遗憾未能在这学期将其一一完善。希望等到学期之后的暑假对自己实现的算法做一些性能上的优化，以及交互方式的扩展（例如加入鼠标交互），争取让这个系统的用户体验更好。