《计算机图形学》10月进展报告

161220183 周梓康 - 2018年10月31日

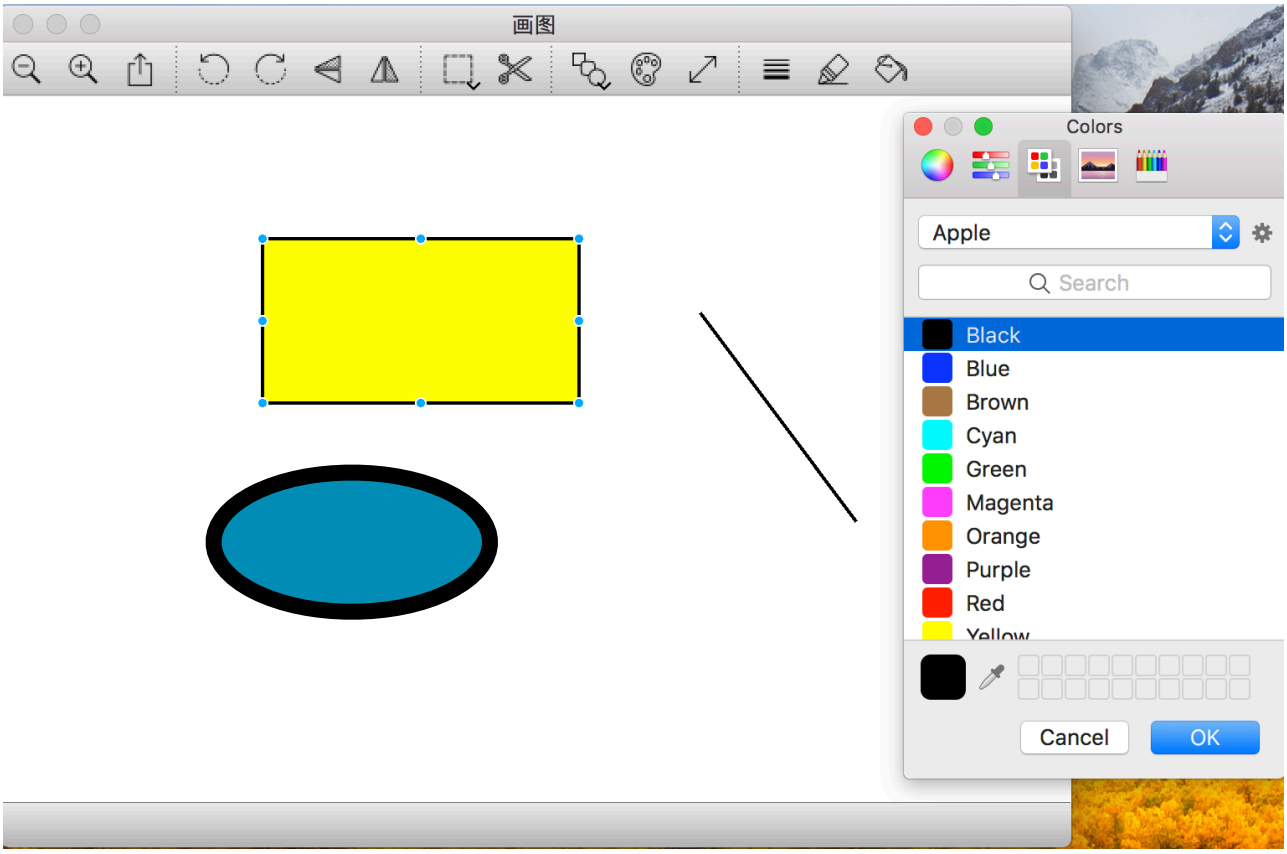
综述

开发环境

操作系统：macOS 10.13.6

开发工具：Python3.7+PyQt5.11.3+Pycharm

图形界面效果图



完成进度

1. 二维图形的输入功能：直线、曲线、多边形、填充区域
2. 二维图形的编辑功能：直线、曲线、多边形、填充区域
3. 二维图形的变换功能：平移、旋转、缩放

算法介绍

Bresenham画线算法

我们知道，在现实世界和数学中，一条直线是有无限精细的粒度的，而计算机屏幕上却是一个一个的像素，精度有限。所以，我们只有直线的数学表达式并不够，我们还需要决定屏幕上哪些像素应该被选中并组成离散化表示后用像素表示的直线，此时就是Bresenham画线算法发挥作用的时候。

首先，算法根据直线斜率来确定每次取样一个单位是在x轴上进行，还是y轴上进行，取样后，在另一轴上的增量为0还是为1，则是取决于另外的因素——实际直线与最近像素点的距离。设一个直线方程为y=mx+b，以在x轴正方向取样为例，假设在(xk,yk)处画好了点，则下一个点应该是(xk+1,yk)或者(xk+1,yk+1)，前者在y轴上没有增量，后者在y轴上有1的增量。

在像素xk+1处， 线段的y坐标为：y = m(xk)+ 1 + b = m((xk) + 1) + b
而取样位置处上下两个像素与实际直线的垂直距离为d1,d2

d1 = y - yk = m(xk+1) + b - yk
d2 = (yk+1) - y = (yk+1) - m(xk+1) - b
d1 - d2 = 2m(xk+1) - 2yk +2b - 1
$m=\Delta y/\Delta x$

从而得到画线时第k步的决策参数：pk = $\Delta x(d1 - d2) = 2\Delta y * xk - 2\Delta x * yk + c$

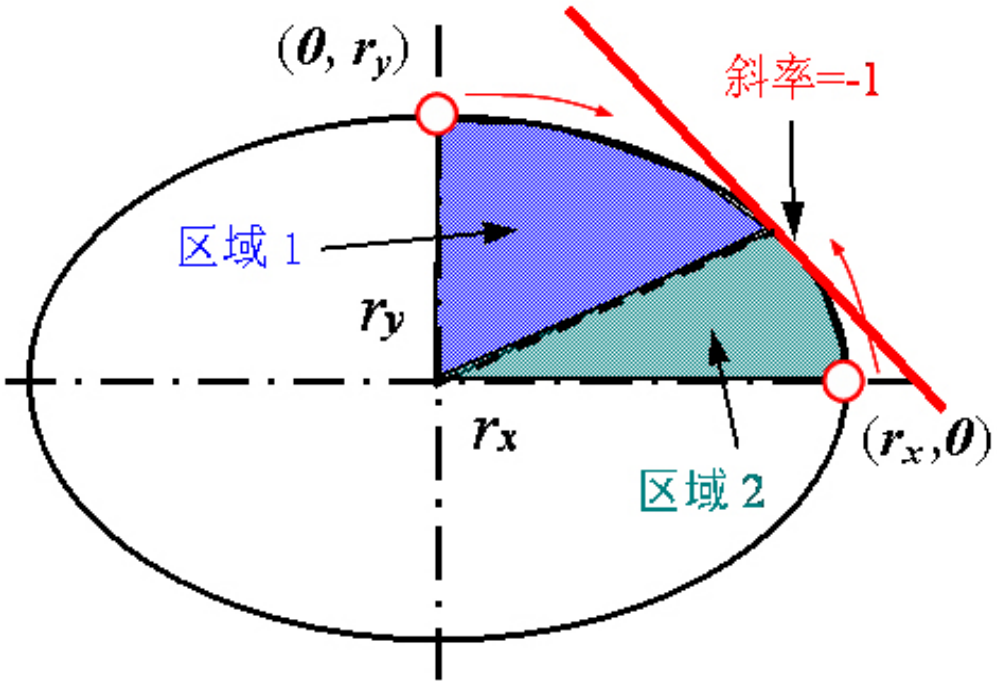
决策过程如下：

若pk > 0，则d1 > d2， (xk+1, yk+1)更接近实际直线
pk+1 = pk + $2\Delta y - 2\Delta x$
若pk <= 0，则d1 <= d2， (xk+1, yk)更接近实际直线
pk+1 = pk + $2\Delta y$

圆（椭圆）中心点生成算法

圆（椭圆）在某一个象限内会被分为两个区域，这是因为切线斜率可能会大于1，也可以小于1，而等于1的那一个点就成为了两个区域的分界。

圆是椭圆的特殊情况（焦点重合，离心率为0）。下面说明椭圆中点生成算法。



考察从$(0,r_y)$开始顺时针方向生成椭圆的过程。

区域1中(|切线斜率|≤1)

假如第k步选择的像素为(x_k,y_k)，将取样位置x_{k+1}处两个候选像素间中点对 椭圆函数求值，即：计算决策参数：

$$p1_k=f_{ellipse}(x_{k+1}, y_k-1/2)=r_y^2(x_k+1)^2+r_x^2(y_k-1/2)^2-r_x^2r_y^2$$

假如$p1_k<0$，中点位于椭圆内，扫描线y_k上的像素更接近于椭圆边界，选择像素位置：(x_{k+1},y_k)。

假如$p1_k\geq 0$ ，中点在椭圆外或边界上，所选像素应在扫描线y_{k-1}上，选择像素位置：(x_{k+1},y_{k-1})。

旋转

- 物体沿xy平面内圆弧路径重定位。
  - 指定旋转基准点位置(X_r, Y_r)和旋转角0 (逆时针旋转为正);
  - 绕通过基准点且垂直于xy平面的旋转轴旋转。
- 基准点为坐标原点:
  - $x_1 = x\cos\theta - y\sin\theta$
  - $y_1 = s\sin\theta + y\cos\theta$
  - $R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$
- 任意基准位置:
  - $x_1 = x_r + (x - x_r)\cos\theta - (y - y_r)\sin\theta$
  - $y_1 = y_r + (x - x_r)\sin\theta + (y - y_r)\cos\theta$
- 与绕原点旋转的差异:
  - 一个加项(平移项)及坐标值上的多重系数。
- 列向量矩阵:$P_i = R \cdot P$(旋转矩阵R)
- 行向量矩阵:$P_1^T = (R \cdot P)^T = P^T \cdot R^T$ 。
  - R的转置R^T:交换行和列;C
  - R的置换:改变sin项符号。
- 不变形的刚体变换: 物体上的所有点旋转相同的角度:
  - 直线段:旋转每个线端点;
  - 多边形:每个顶点旋转指定旋转角;
  - 曲线: 旋转控制/取样点。

缩放

- 缩放变换改变物体尺寸——**多边形**。
- 相对于原点的缩放:
  - 将每个顶点坐标(x,y)乘缩放系数S_x和S_y,产生变换坐标(x_1, y_1)：$x_1 = x \cdot S_x, y_1 = y \cdot S_y$.
  - S_x和S_y分别为x和y方向的缩放。
  - 矩阵形式: $P_1 = S \cdot P$
  - 缩放矩阵S: $S = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix}$
- 缩放系数$s(S_x, S_y)$可赋予任何正数。
  - $s < 1$: 缩小，$s > 1$: 放大;
  - $S_x = S_y$: 一致缩放;
  - $S_x \neq S_y$:差值缩放。
- 多边形: 变换每个顶点的坐标。
- 其它物体: 变换定义物体的参数。
- 缩放变换缩放物体且对其重定位。
  - $s > 1$,坐标位置远离原点。
- 固定点缩放: 选择变换后不改变物体位置的点(x_f, y_f)进行缩放。
- 多边形顶点、物体中点。
  - 顶点(x, y)缩放后坐标(x_1, y_1)为:
    - $y_1 = X * S_x + x_f(1 - S_x)$;
    - $y_1 = Y * S_y + y_f(1 - s_y)$;
- $x_f(1 - s_x)$和$y_f(1 - s_y)$对任何点都是常数 → 原点缩放+平移:可将此项作为列向量，再将其加到原点缩放上。
- 多边形固定点缩放: 缩放每个顶点到固定点的距离。

系统介绍

框架设计

基于Graphics View Framework实现。

系统主要由三个部分组成：

场景类Scene：统一管理所有绘制的图形项。

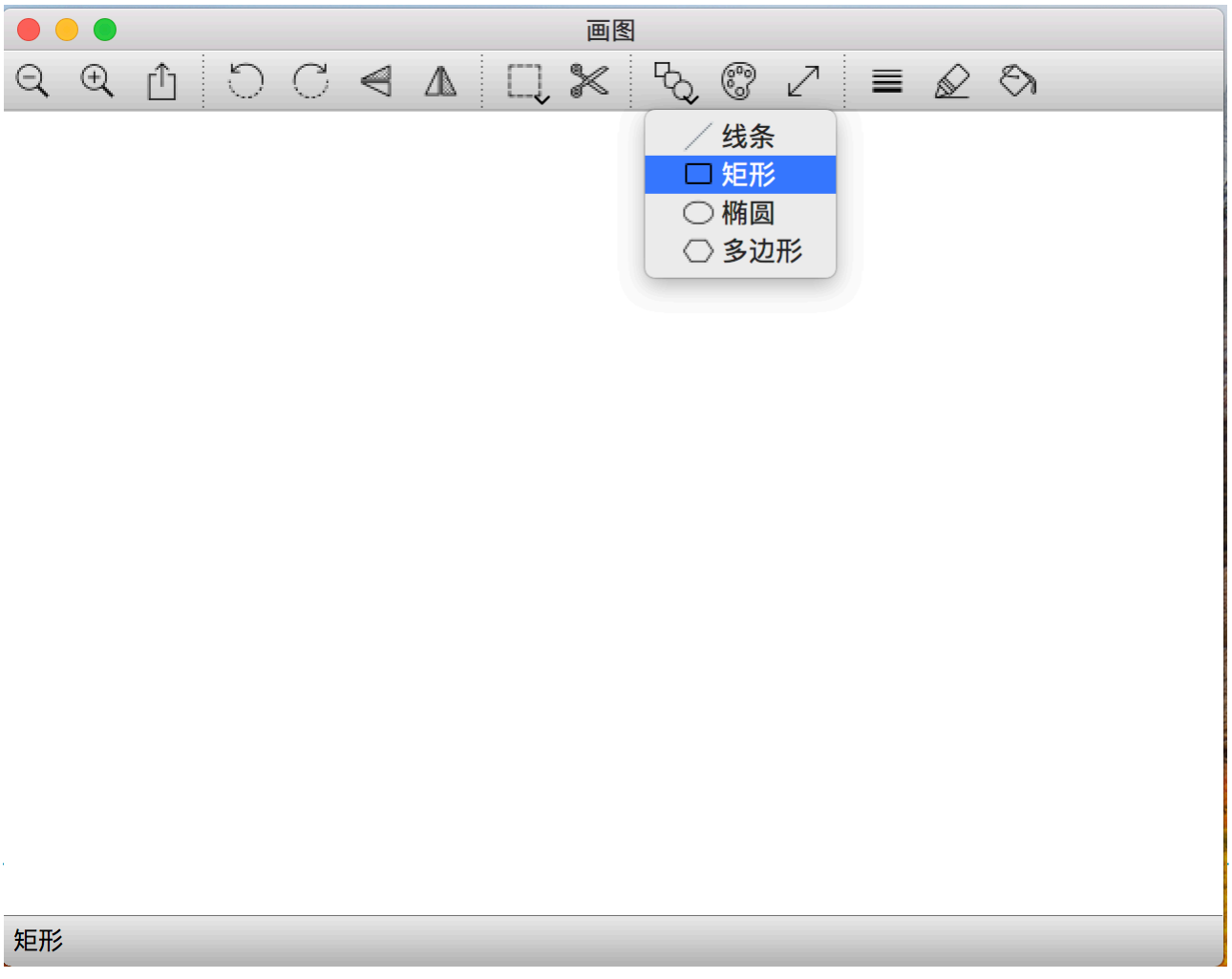
若干图形项Item：对应每个图形实体，包含位置信息、形状信息等。可以处理键盘事件以及鼠标事件。放置于场景Scene中。

视图类View：负责与用户交互，场景Scene中的图形项Item绘制于视图View中。

实现思路

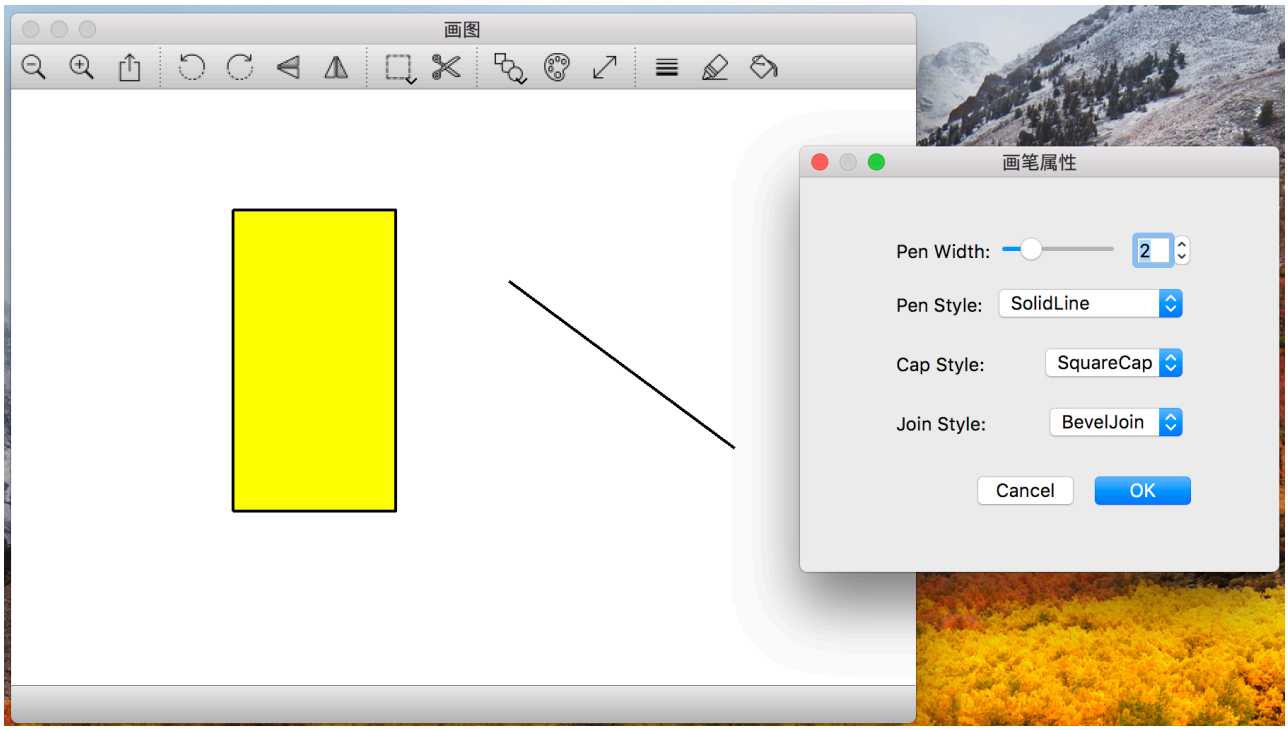
1. 二维图形的输入功能

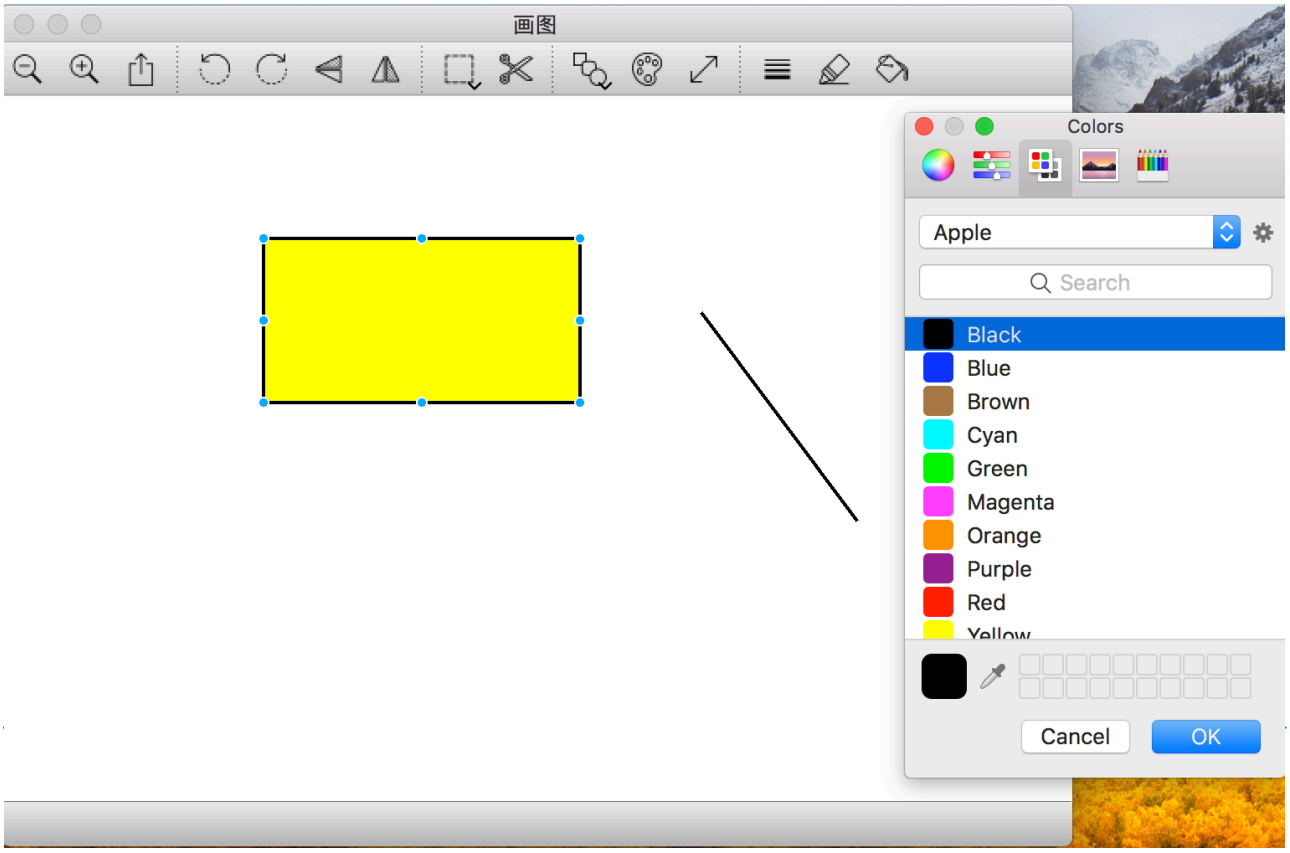
View类
选择将要绘制的形状：



完成选择后，发送一个信号以识别将要绘制的是何种形状。该信号触发Scene类中的set_cur_shape()方法，设置cur_shape属性以确定将要绘制的形状。

类似地，可以设置画笔的属性（粗细、样式、边框颜色、填充颜色等），View类将与用户交互所得到的属性数据向下传递给Scene类。





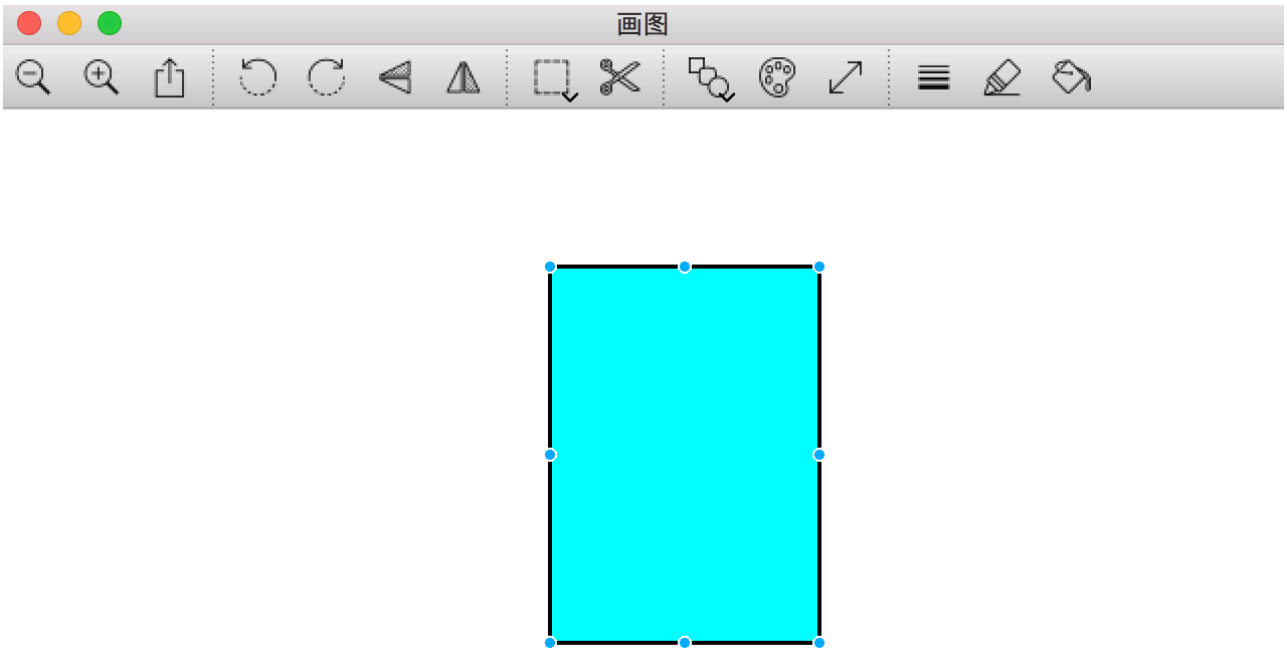
Scene类
图案选择及画笔属性设置完毕后，Scene类开始持续聆听用户的鼠标及键盘事件，并将从View类得到的属性数据继续向下传递给各图形项Item。
mousePressEvent()、mouseMoveEvent()以及mouseReleaseEvent()协同作用，记录在绘制过程中图形的坐标位置。一旦绘制完成，Scene类将新绘制的图形项加入自身拥有的容器中对图形项进行统一管理。
另外，可以通过点击右键或点击Backspace的方式删除图形项（对于Scene类来说，此操作相当于将图形项移出管理容器）。

Item类
最后，Item类中对paint()方法负责对图形的最终绘制。使用上述提及算法对具有不同形状的图形分别进行绘制。

2. 二维图形的编辑及变换功能

该部分功能主要由Item类实现。下面以二维图形缩放为例对实现思路进行介绍。

mousePressEvent()：
检测鼠标当前位置是否位于某个图形的边界区域内，若是，则将对应的图形选中：



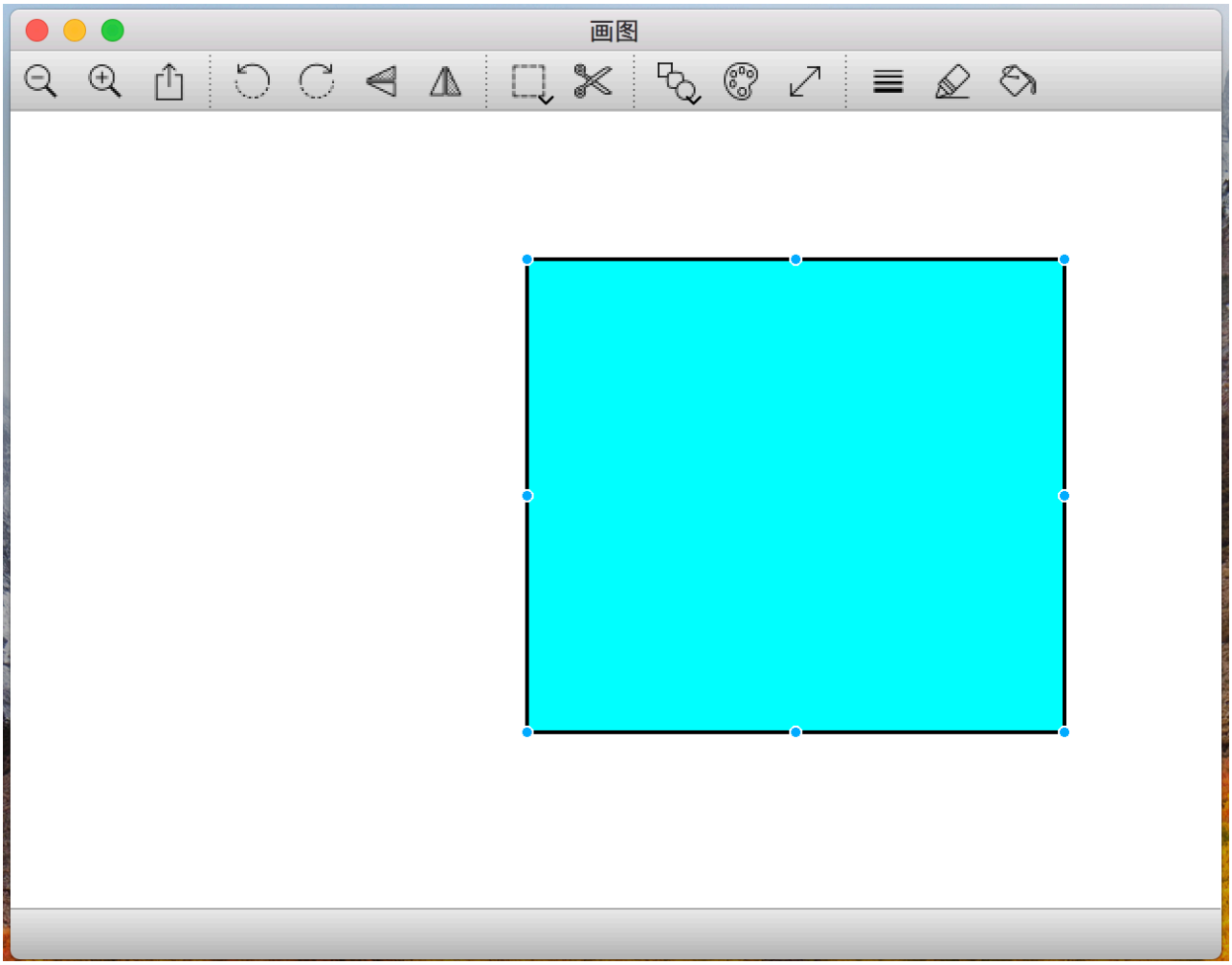
如上图所示，选中矩形后，在矩形的各顶点以及各边的中点处绘制用于缩放的小圆点状把柄。

检测鼠标是否位于某个小把柄之上。若是，设置一个布尔标记当前状态为“缩放”状态。

mouseMoveEvent():
检测当前是否处于“缩放”状态。若是， 则根据把柄所在的位置来确定缩放过程中的固定点或固定边，再配合鼠标的移动距离及方向确定缩放的比例。

mouseReleaseEvent():
当检测到鼠标释放后，解除“缩放”状态。

下图为拖动右下角把柄效果：



总结

本阶段实现的功能罗列如下：

绘制直线、多边形、椭圆等

改变图形外观，包括边框颜色、填充颜色、边框粗细、边框样式等

左键点击选中图形

左键+Ctrl同时选择多个图形

右键或BackSpace删除选中图形

选中图形后，拖动鼠标对所选图形进行任意位置拖动（可同时拖动多个选中图形）

选中图形后，拖动四周小把柄对图形进行缩放

选中图形后，点击工具栏上相应按键使选中图形进行向左旋转、向右旋转、水平翻转或垂直翻转

参考文献

https://blog.csdn.net/wwj_ff/article/details/50014123
https://blog.csdn.net/liang19890820/article/details/53504323
https://forum.qt.io/topic/7776/resize-a-qgraphicsitem-into-a-qgraphicsscene/2