Homework 1 MiniDraw 画图小程序

85 夏子汐 PB22000057

2024 年 3 月 9 日

1 类图及说明

1.1 MiniDraw 主类（图 1）

图 1: MiniDraw 类图，draw 函数引导画布和功能按钮的绘制

1.2 实现 GUI 的类

实现 GUI 的类（view 文件夹内，如图 2）有：

1. Window：构建和管理 GUI 窗口，例如管理窗口的实时重绘。
2. Component：GUI 内的组件，派生类 Canvas（画布区域和背景）和 Image（为 demo 内所用，此项目没有使用）。
3. Shape：绘图的形状，有派生类 Line、Rect、Ellipse、Polygon、Freehand、ImageFile，分别对应直线、长方形、椭圆、多边形、自由绘图、图片文件。

部分方法的基本说明：

1.2.1 Window 类

- `run()`：实现不断的绘制窗口；
- `render()`：对于一帧，绘制背景和中间内容（调用`draw()`）；
- `draw()`：画出画布和按钮。

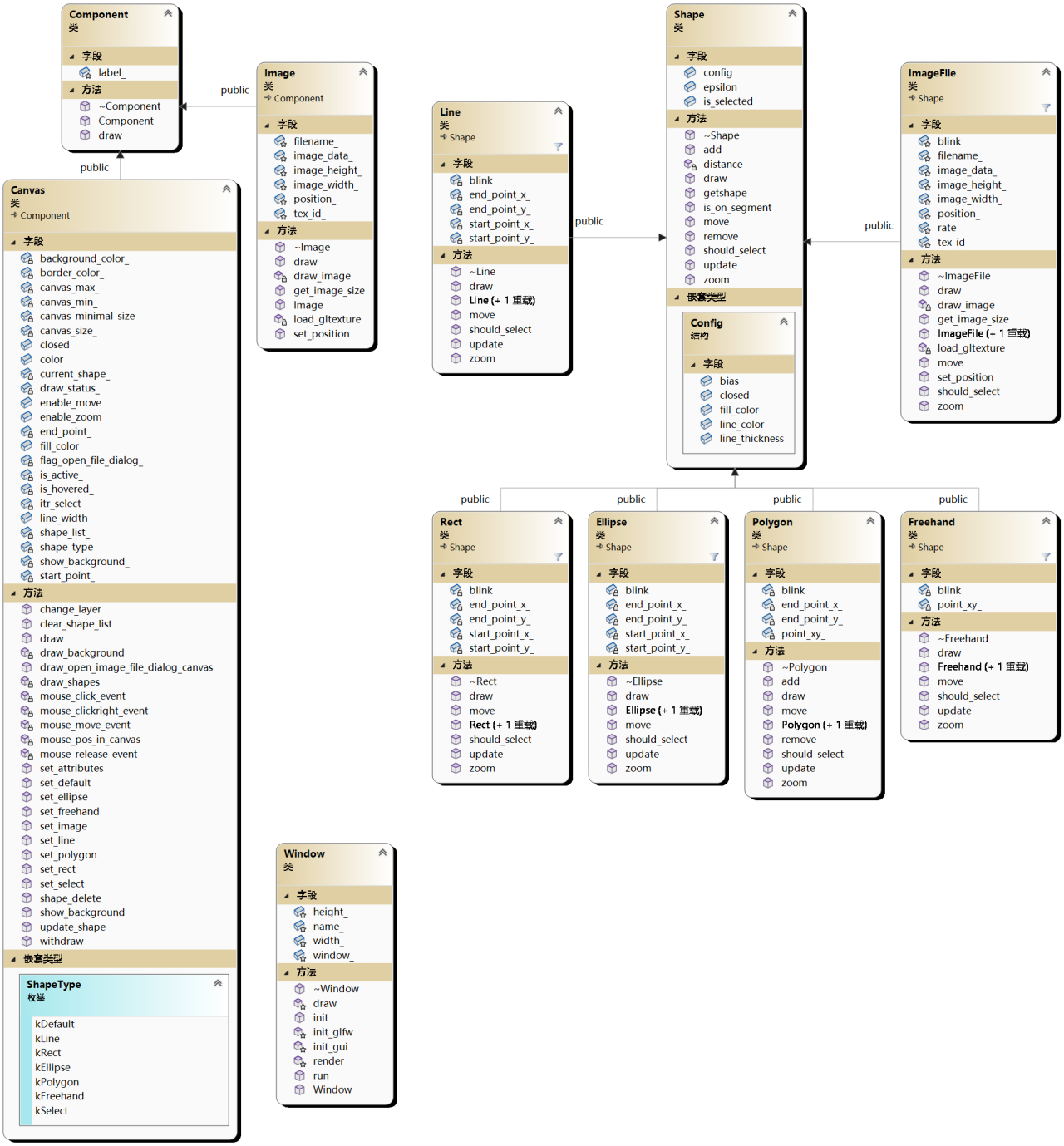
Component
类
字段
label_
方法
~Component
Component
draw
public
Image
类
Component
字段
filename_
image_data_
image_height_
image_width_
position_
tex_id_
方法
~Image
draw
draw_image
get_image_size
Image
load_gltexture
set_position
public
Canvas
类
Component
字段
background_color_
border_color_
canvas_max_
canvas_min_
canvas_minimal_size_
canvas_size_
closed
color
current_shape_
draw_status_
enable_move
enable_zoom
end_point_
fill_color
flag_open_file_dialog_
is_active_
is_hovered_
itr_select
line_width
shape_list_
shape_type_
show_background_
start_point_
方法
change_layer
clear_shape_list
draw
draw_background
draw_open_image_file_dialog_canvas
draw_shapes
mouse_click_event
mouse_clickright_event
mouse_move_event
mouse_pos_in_canvas
mouse_release_event
set_attributes
set_default
set_ellipse
set_freehand
set_image
set_line
set_polygon
set_rect
set_select
shape_delete
show_background
update_shape
withdraw
嵌套类型
ShapeType
枚举
kDefault
kLine
kRect
kEllipse
kPolygon
kFreehand
kSelect
Window
类
字段
height_
name_
width_
window_
方法
~Window
draw
init
init_glfw
init_gui
render
run
Window
Line
类
Shape
字段
blink
end_point_x_
end_point_y_
start_point_x_
start_point_y_
方法
~Line
draw
Line (+ 1 重载)
move
should_select
update
zoom
public
Shape
类
字段
config
epsilon
is_selected
方法
~Shape
add
distance
draw
getshape
is_on_segment
move
remove
should_select
update
zoom
嵌套类型
Config
结构
字段
bias
closed
fill_color
line_color
line_thickness
public
ImageFile
类
Shape
字段
blink
filename_
image_data_
image_height_
image_width_
position_
rate
tex_id_
方法
~ImageFile
draw
draw_image
get_image_size
ImageFile (+ 1 重载)
load_gltexture
move
set_position
should_select
zoom
public
Rect
类
Shape
字段
blink
end_point_x_
end_point_y_
start_point_x_
start_point_y_
方法
~Rect
draw
move
Rect (+ 1 重载)
should_select
update
zoom
public
Ellipse
类
Shape
字段
blink
end_point_x_
end_point_y_
start_point_x_
start_point_y_
方法
~Ellipse
draw
Ellipse (+ 1 重载)
move
should_select
update
zoom
public
Polygon
类
Shape
字段
blink
end_point_x_
end_point_y_
point_xy_
方法
~Polygon
add
draw
move
Polygon (+ 1 重载)
remove
should_select
update
zoom
public
Freehand
类
Shape
字段
blink
point_xy_
方法
~Freehand
draw
Freehand (+ 1 重载)
move
should_select
update
zoom

图 2: view 类图

1.2.2 Component 的派生类 Canvas

- `mouse_xxx()`：处理鼠标的各种行为；
- `set_xxx()`：将待画的内容设置为该类型；
- `draw()`：画出背景和所有图形（包括插入的图片，下同）；
- `draw_open_image_file_dialog_canvas()`：画出打开图片文件的界面；
- `withdraw()`：绘画模式下撤销上一次操作；
- `update_shape(...)`：修改当前（或选中元素）的绘画参数，包括线宽、颜色等（存储在 Config 结构体里面）；
- `shape_delete()`：删除选中图形；
- `change_layer(x)`：改变选中图形在画面中的层次。

1.2.3 Shape 类

（派生类的方法不能直接在 Canvas 层访问，故不列出）

- `draw()`：画出当前图形；
- `update(x,y)`：更新图形状态（用于鼠标移动时绘图的更新）；
- `getshape(...)`：修改当前图形的绘画参数；
- `should_select(x,y)`：判断点击 (x,y) 点时是否需要选中该图形；
- `move(x,y)`：对图形进行 (x,y) 的位移；
- `zoom(x)`：将图形放大到原来的 x 倍；
- `add(x,y)`/`remove()`：（仅对 Polygon）增加或删除一个顶点。

2 功能及其实现说明

2.1 画图

Line 和 Rect 已经由代码框架给出。

- Ellipse：和 Rect 一样确定对角的两个点，利用 AddEllipse 函数画出；
- Polygon：由多条直线构成；
- Freehand：每帧在 Polygon 中画出一个点，由很多细小的直线构成；
- Rect 和 Ellipse 在绘制时按下 Shift 键，可以绘画正方形和圆形；
- 主界面右上可以选择”Filled” 或”Closed”（`ImGui::Checkbox`元素），这两个参数传递到 Config 内，控制是否填充或者多边形是否封闭，如图 3 所示；
- 主界面第二行可以控制线宽度（`ImGui::SliderScalar`元素）和线条/填充颜色（`ImGui::ColorEdit4`元素），参数传递到 Config 来改变绘图行为。示例如图 4 所示。可以在画图的中途修改这些参数；
- 第一行红色的”Withdraw” 键可以用于删除最后画的图形（对`shape_list_`操作）；如果正在画图，可以撤销最后一次操作（对`current_shape_`操作），例如对于 Polygon 可以删去最后添加的一个顶点。

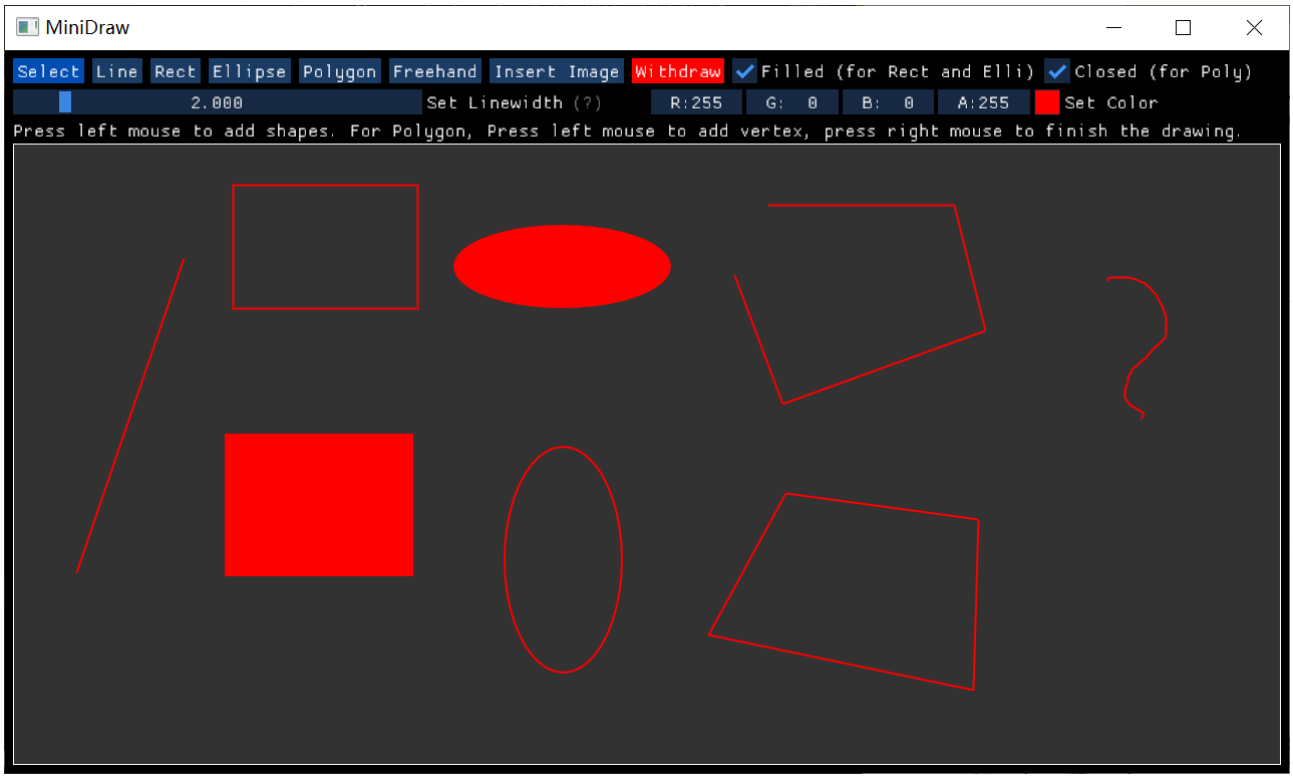

图 3: 画图功能

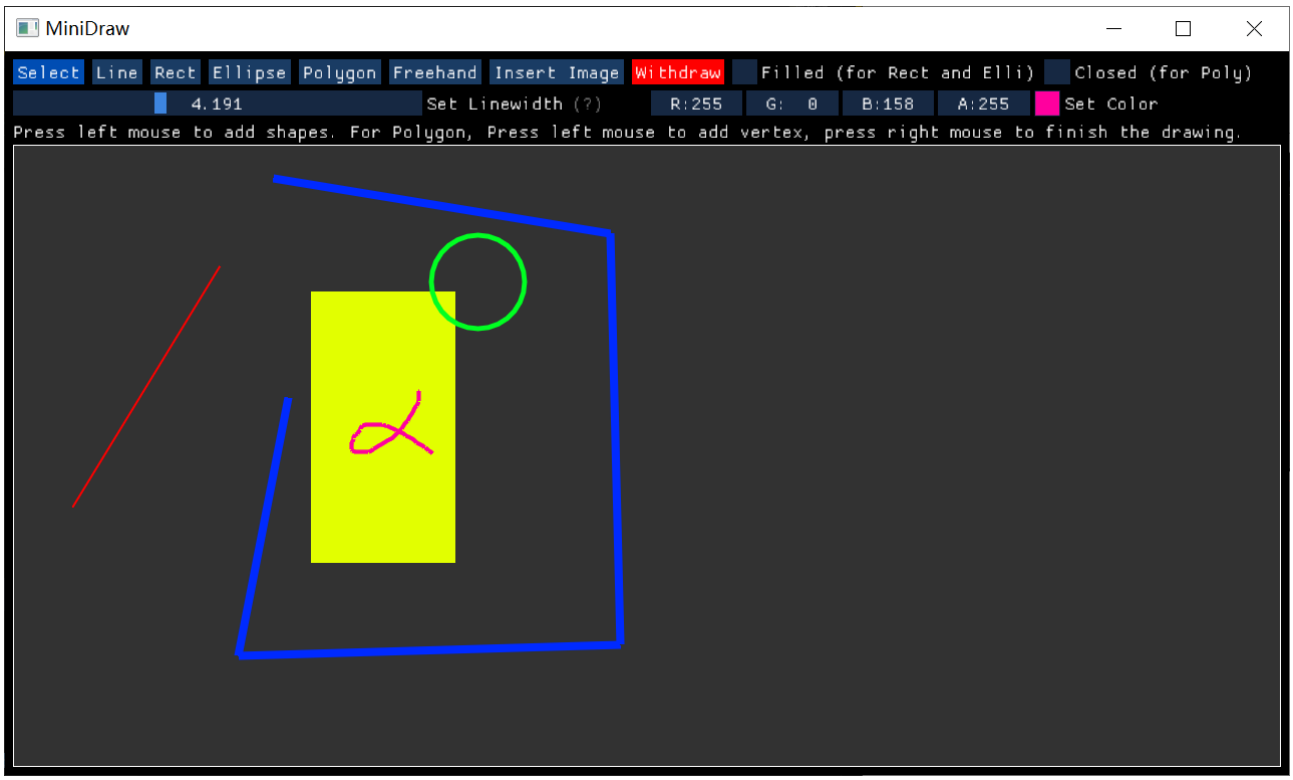

图 4: 调节画图参数

2.2 插入图片

图片作为一个图形，是 Shape 类的一个派生类。

- 按下”Insert Image”，会进入一个文件选择窗口，选择要插入的”.png” 或”.jpg” 图片。利用 demo 里面 `drag_open_image_file_dialog_`函数实现。如图 5、6 所示；
- 插入的图形会按照像素一比一显示，所以可能会出现图片无法完整显示的现象。图片作为一个 Shape 同样可以使用”Withdraw” 删除。

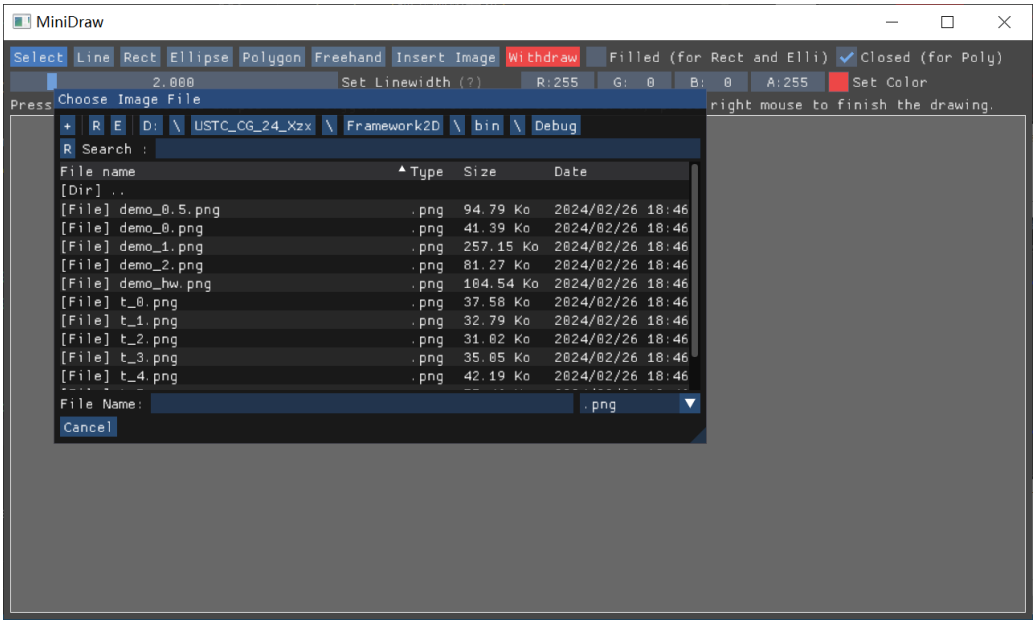

图 5: 文件选择窗口

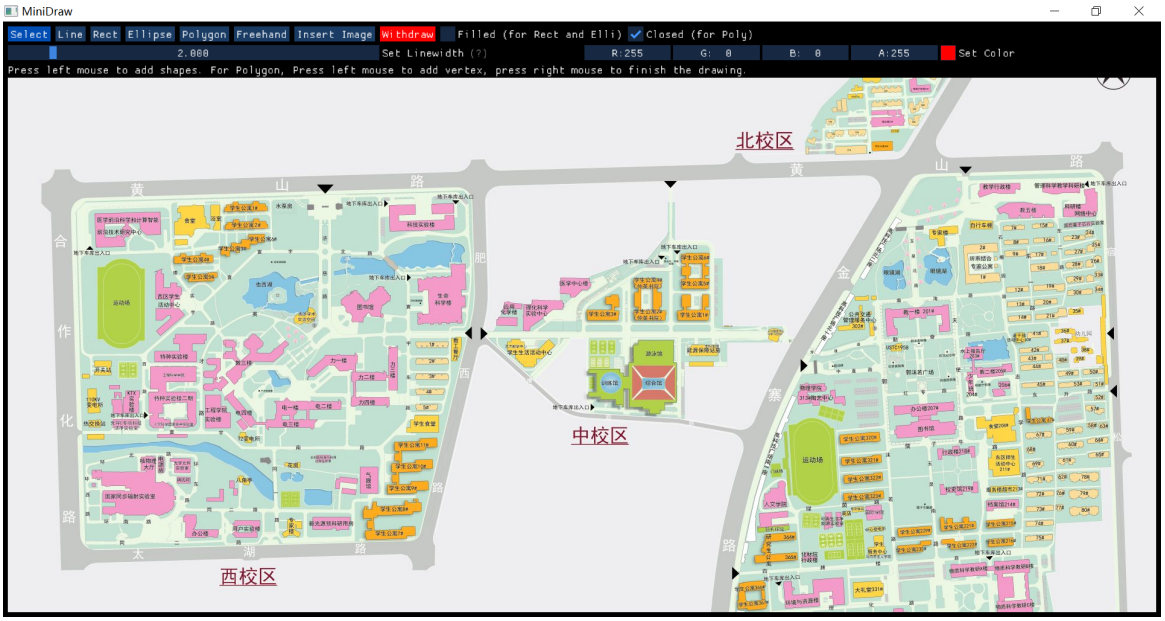

图 6: 插入图片（此处图片大小较大，无法完整显示）

2.3 选择模式

2.3.1 进入和退出

- 按下”Select” 按钮进入选择模式（如图 7 所示），然后按钮变为”Draw”，按下可以返回画图模式；
- 选择模式下无法画图和插入图片，相关按钮消失；
- 界面上由许多”(?)”，靠近可以查看关于此功能的一些提示。（利用 demo 的`HelpMarker`方法实现）

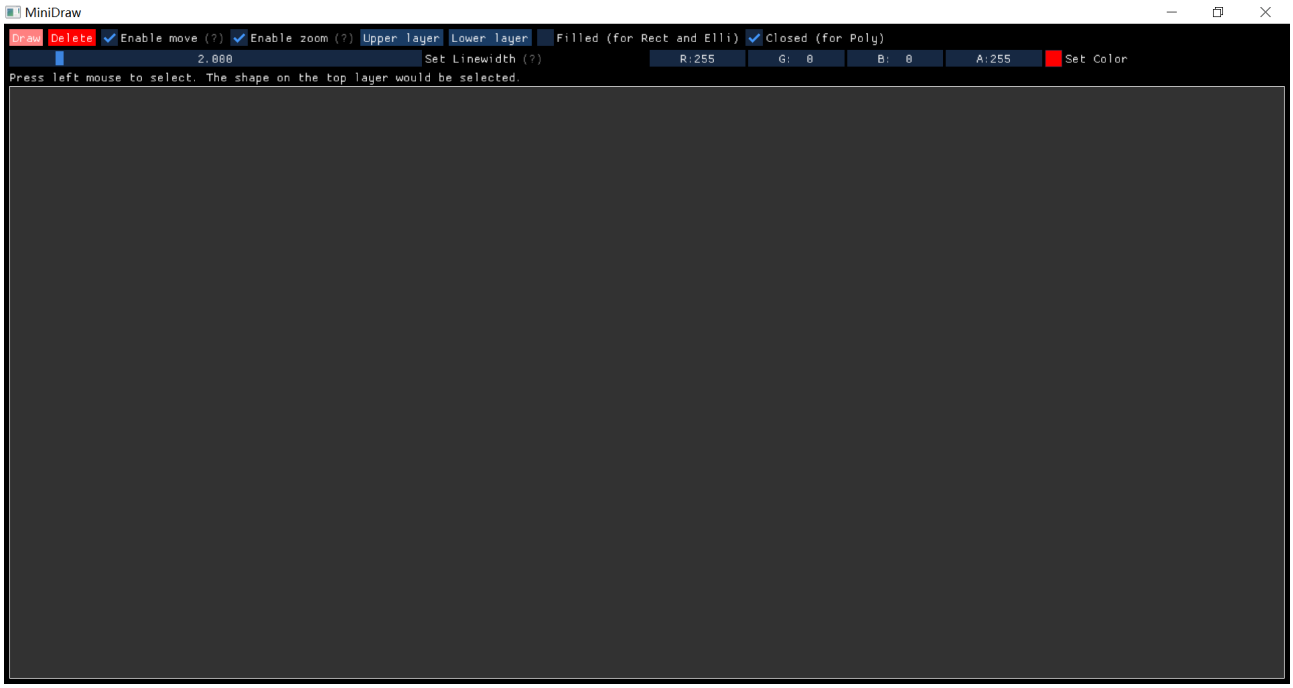

图 7: 选择模式

2.3.2 选择图形

根据直觉，按下一个图形上或附近很近的像素，可以选中这个图形，采用如下的原则和方法判断是否选中：

- 根据直觉，堆叠在上方的图形会优先选中；
- 对于 Line，若记为 AB，点击的点为 C，设置宽容度 ε，满足 $AC+BC-AB<\varepsilon$ 即认为靠近直线，选中；
- 对于 Rect(not filled)、Polygon、Freehand，只要选中其中一个直线就选中整个图形；
- 对于 Rect(filled) 和 ImageFile，判断点击点是否在长方形区域内即可；
- 对于 Ellipse，设方程 $\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}=1$，计算点击的点 (x,y) 的 $\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}$，那么选中条件为：

$$|(\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2})-1|<\varepsilon \qquad \text{(not filled)} \tag{1}$$

$$\frac{(x-x_0)^2}{a^2}+\frac{(y-y_0)^2}{b^2}<1+\varepsilon \qquad \text{(filled)} \tag{2}$$

- 选中的图形显示出动态效果：普通的 shape 会出现“若隐若现”效果，ImageFIle 会出现缩放效果。这是在渲染的过程中修改颜色/大小参数实现的。

2.3.3 编辑图形

- 修改 Config 参数：和画图模式相同，可以修改 Filled、Closed、linewidth、color 参数并实时渲染（不能使用 Withdraw 撤销），例子如图 8 所示；

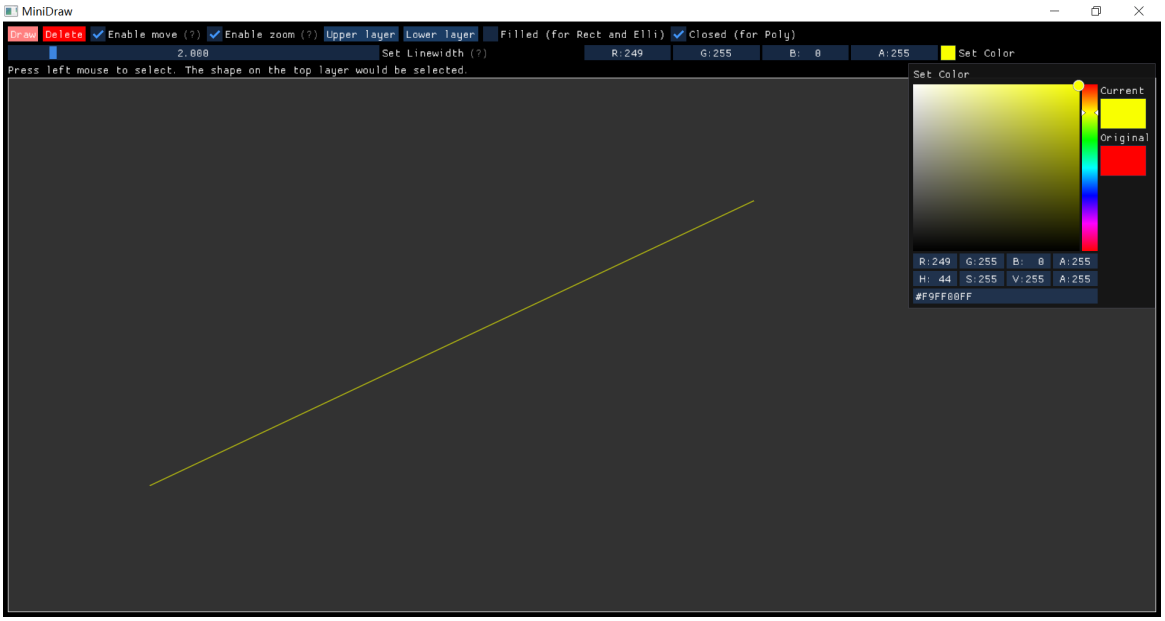

图 8: 修改一条线的颜色

- 左上角”Delete” 键可以删除选中的图形，从`shape_list_`中取出这个图形；
- 在框选”Enable move” 后，可以使用键盘方向键移动被选中的物体；
- 在框选”Enable zoom” 后，可以使用鼠标滚轮对选中的物体进行放大和缩小，例如把图 6 缩小，如图 9 所示；

图 9: 缩小图片

- 使用”Upper layer”、”Lower layer” 或者键盘 Ctrl+Up/Down 来修改选中物体的层级，如图 10、11 所示。
- 利用以上特性可以进行一些综合绘图，例如在地图上标注常用轨迹，如图 12 所示。

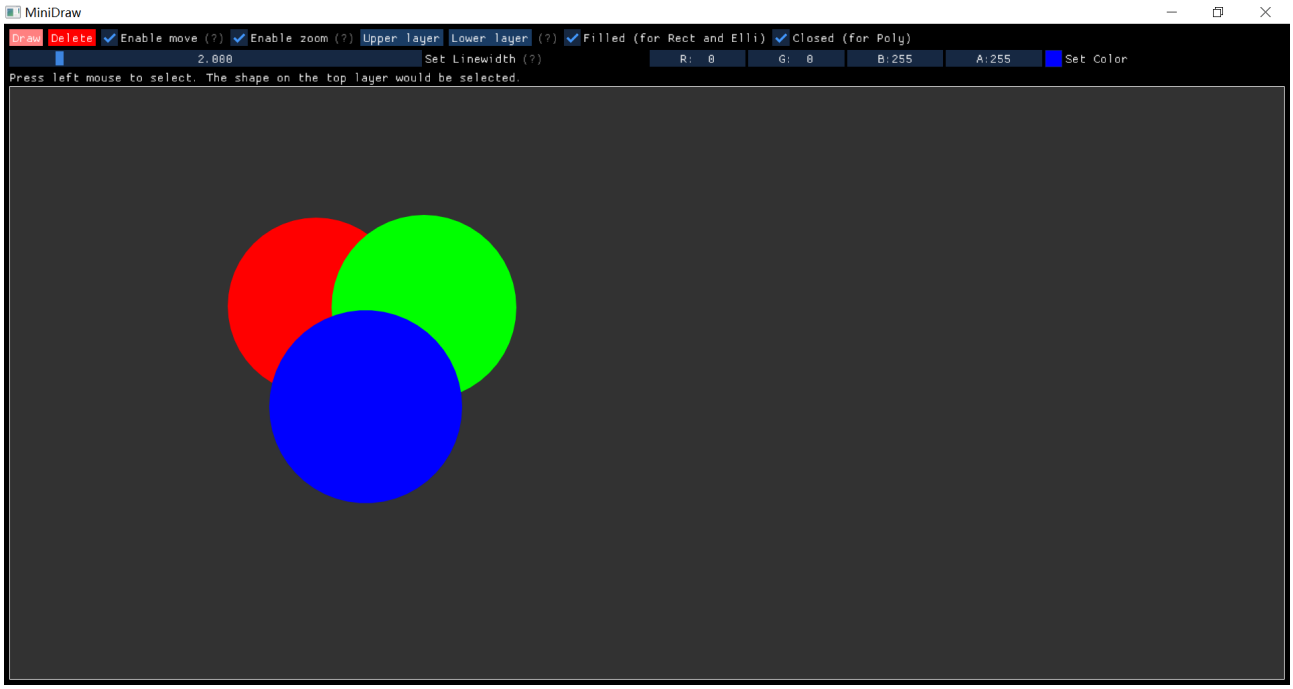

图 10: 互相遮罩的图形

图 11: 将红色块向上移一层

图 12: 在地图上标注常用轨迹

3 总结和展望

该作业中充分利用 C++ 类的继承和多态思想，利用 Shape 基类存储通用的形状信息（同时利用指针给出了一个可以访问的通道），派生类有着不同的字段和方法（名字相同，可以统一地从 Shape 处调用）；同时对于基本的几何关系和代码逻辑有一些涉及，得到了一个功能比较完备的画图小程序。

3.1 做作业过程中遇到的一些问题

- 绘制多边形中“增加顶点”的操作不同于`update(x,y)`，需要一个新的函数，但是直接在 Polygon.h/.cpp 里面增加这样的函数`add(x,y)`是不可以的，因为这里的`current_shape_`已经是一个抽象类 Shape 的指针了，不能直接访问到`add(x,y)`，编译失败。

  这里可以在基类 Shape 里面加上`add(x,y)`函数，但是在 Polygon 以外的派生类中均不实现这个函数（实际上有实现，但是是空的）即可。

- Freehand 和 Polygon，在`line_width`较大、线段长度比较小时，转折点处会出现明显的瑕疵，可以通过增加圆角过渡解决，如图 13、14 所示。

图 13: 不加圆角，有明显瑕疵

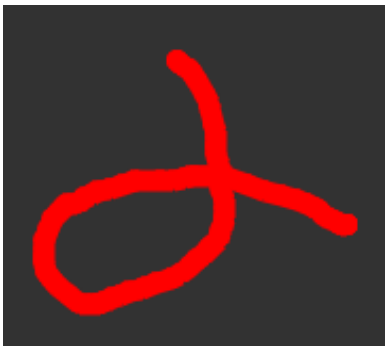

图 14: 加圆角，曲线更加平滑

图 15: 透明度有问题

- 如何判断被选中，已经在 2.3.2 节叙述。

- 如何记录选中的状态，使得既能快捷知道也没有东西选中了（Shape 层次），又能知道它在`shape_list_`中的位置？

  这里使用`vector::iterator`存储选中元素的指针，这样能够方便地在`shape_list_`上定位。具体实现上，出于直觉考虑，为了能使上层的 Shape 先被选中，需要倒序遍历`shape_list_`，使用`reverse_iterator<vector<shared_ptr<Shape>>::iterator>` 类型的反向迭代器`itr_select`。

  如何判断被选中？由于 iterator 遍历后会停止在`shape_list_.rend()`（反向迭代器，指向第一个元素的前面），判断是否有东西被选中，只用`itr_select==shape_list_.rend()`判断。

  此时仍然会有一个问题，就是在进行 delete 操作会改变 vector 的结构，同时改变了`shape_list_.rend()`，导致出现违规的访问。只要在每一次 delete 操作后更新`itr_select=shape_list_.rend()`即可。

- 如何展示被选择的状态？使用`Shape::blink`变量作为一个每一次`Shape::draw()`时都会改变的值，达到不同帧画面不同，“闪烁”的效果。

3.2 在目前的框架下仍然可以有一些提升

- 批量选择操作，利用 vector 记录选择内容；
- 高级填充操作，例如渐变色填充、封闭多边形填充；
- 对透明色有更好的兼容性（例如设置图片透明度；处理半透明`line_color`下 Freehand 形状颜色设置问题，如图 15 所示）；
- 改进类的结构和变量、方法的命名和实现方式；
- 一些未知 bug 的发现和修复。