# 2021 - 2022 学年第1学期

Java大作业报告

作业题目名称 Java版的幻灯片制作与播放软件

学院（系）计算机科学与工程学院 计算机科学与技术系

小组成员人数 3 人

小组成员名单(学号和姓名）：

组长 班级计科1907学号20194701 姓名白千一 成绩\_\_\_\_\_\_

组员 班级计科1907学号20194624 姓名李子豪 成绩\_\_\_\_\_\_

组员 班级计科1907学号20194614 姓名洪鑫城 成绩\_\_\_\_\_\_

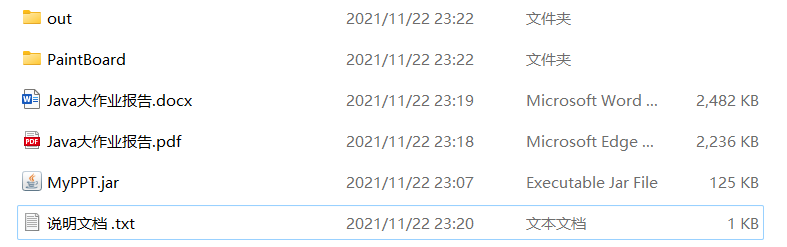
**幻灯片制作与播放软件设计**

## 一、作业内容

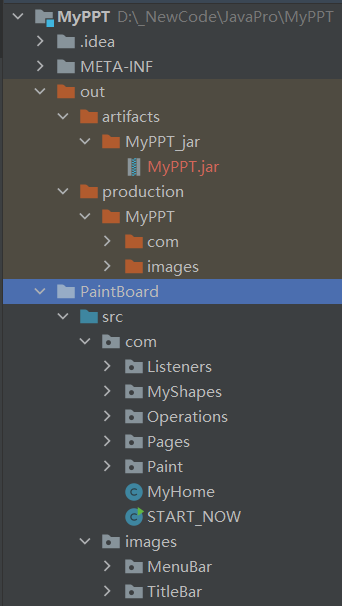
以团队的形式，使用Java语言，开发一个简易的幻灯片制作与播放软件（参照微软Office的Powerpoint）。

## 二、文件与执行环境

##### 2.1电子文档打包文件名及文件列表：



|  |  |
| --- | --- |
| 目录/文件 | 文件内容 |
| out | 编译生成的字节码目录 |
| PaintBoard | 源代码文件目录 |
| Java大作业报告.docx | Java大作业报告 |
| MyPPT.jar | 可执行字节码 |
| 说明文档.txt | 说明文档 |



项目目录结构

|  |  |
| --- | --- |
| 目录 | 文件内容 |
| .idea | IDE配置文件 |
| META-INF | jar包打包配置文件 |
| out/production | 编译生成的字节码文件 |
| out/artifacts | 打包生成jar包目录 |
| PaintBoard | 工程文件 |
| images | 图片资源 |
| src | 源代码文件 |

##### 2.2编译执行环境与命令

运行环境： IDEA + JDK1.8

命令行命令：Java START\_NOW

## 三、主要功能

(1)新建幻灯片，幻灯片由不少于一个页面组成；

(2)在幻灯片找那个插入一个空白的幻灯片页面；

(3)打开或保存整个幻灯片；

(4)在幻灯片页面上绘制基本图形（直线、矩形、圆、椭圆）；

(5)在幻灯片页面上绘制任意线；

(6)添加文字；

(7)设定颜色与文字风格（种类数≥3）；

(8)对幻灯片页面上已有的基本图形、线形、文字进行选取和移动；

(9)通过鼠标拖动完成上述绘制和添加文字等操作

(10)幻灯片的点击翻页；

(11)设定画笔、插入图像；

(12)可以设置作图线条的粗细

(13)对幻灯片页面上已有的基本图形、线形、文字选取进行颜色、线宽、字体大小修改；

(14)部分操作的撤销与重做；

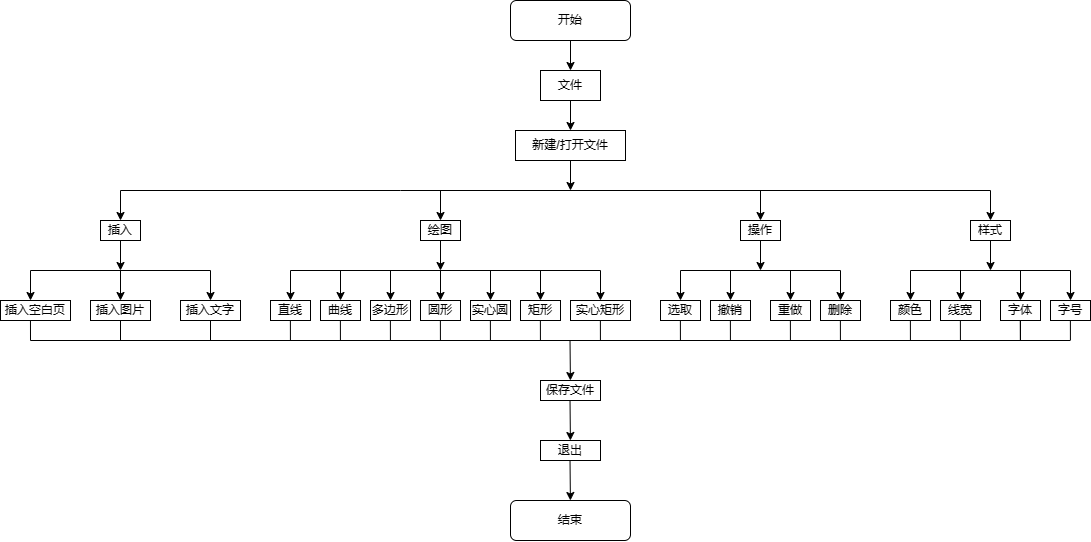
(15)菜单

(16)绘制填充（椭）圆形和填充矩形

(17)对选中图形进行删除

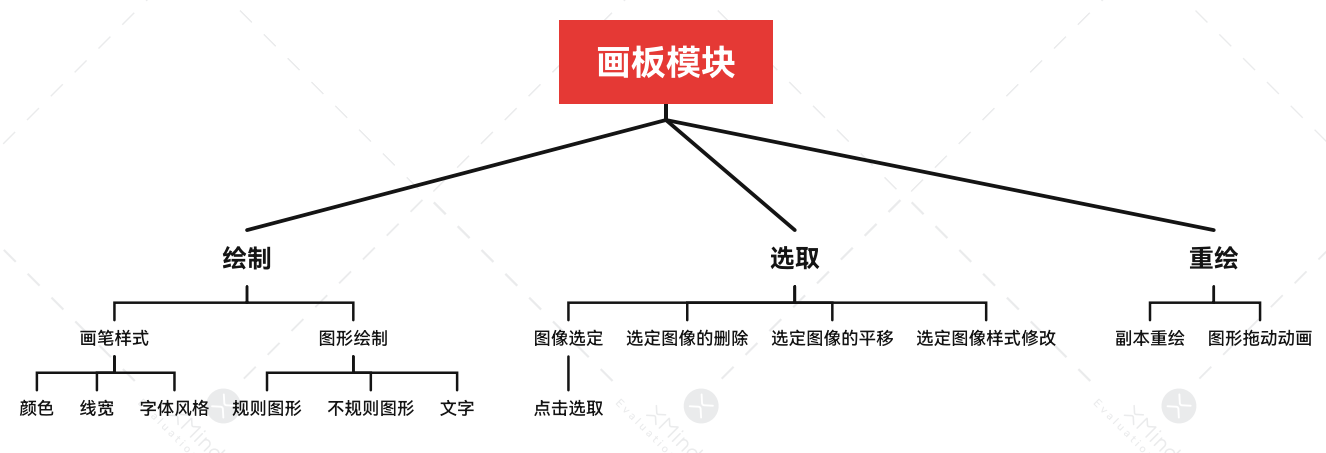
## 四、系统的设计

##### 4.1使用流程图



##### 4.2功能模块图及说明

###### 4.2.1 画板模块



画板模块功能图

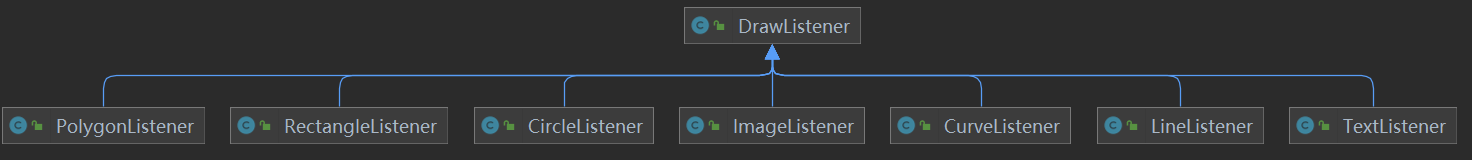
1. 模块功能

本模块负责在界面上开辟一个能进行绘画的画板。绘画功能的实现是通过改变同本画板绑定的鼠标监听器来实现的。

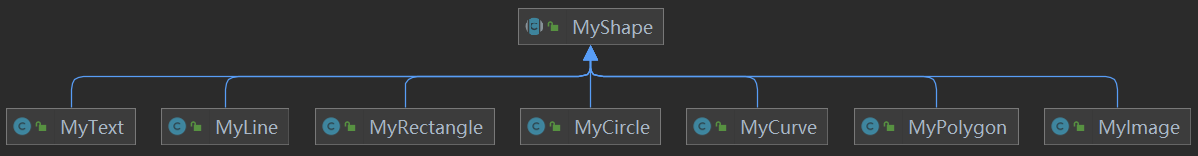
1. 实现原理

绘画功能要求在同一个画板上进行相同的鼠标动作时，产生不同的监听器效果。所以当切换要绘画的图形时，将同画板绑定的监听器改变成对应的监听器，便可达到要求。

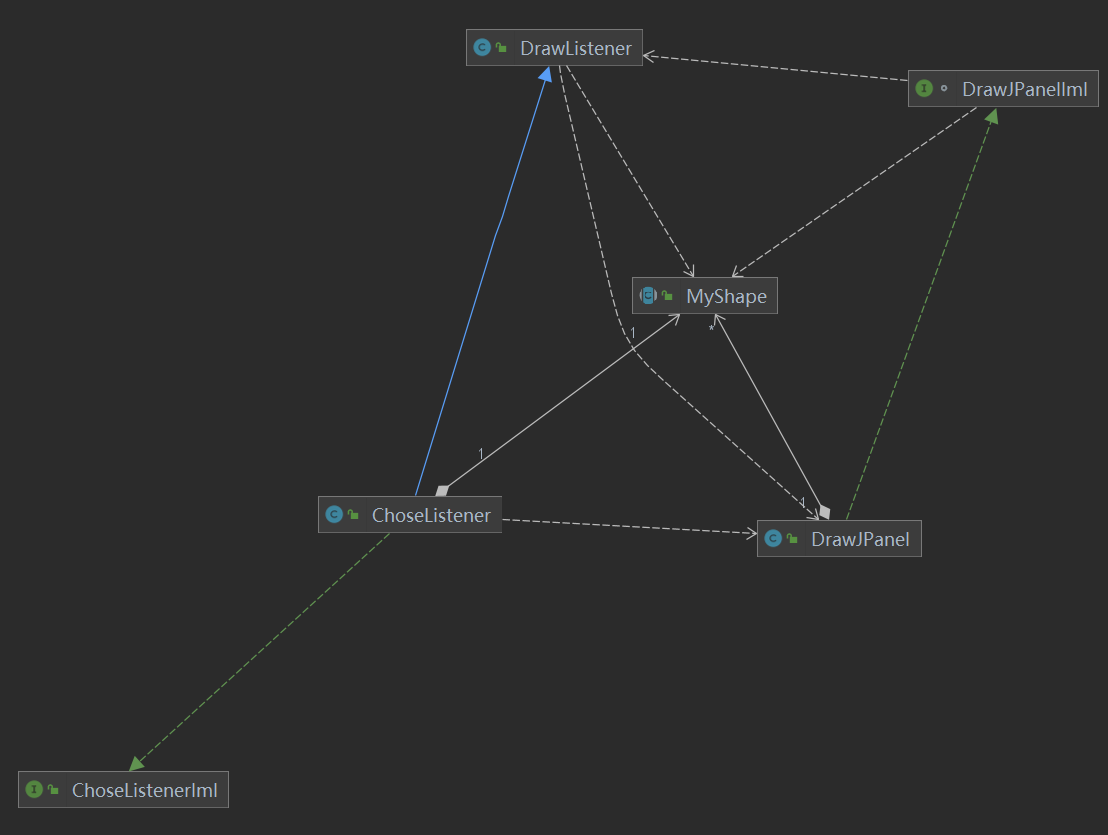
1. 模块构成类



图形监听器类关系图



自定义图形类（MyShape）关系图



画板模块类关系图

1. 画板类（DrawJPanel）

画板本身由一个DrawJPanel类构成，此类继承自JPanel，所以可以放置在界面的容器中。该类实现的功能有：改变画笔样式，保存当前画面的副本以重绘。画板绘制图形样式改变是通过改变DrawJPanel的画笔（drawPen：Graphics2D）的相关属性实现的。保存副本是通过绘制一个特定大小的图像（BufferedImage），进行和原本相同的操作来带到保存原本的目的。

1. 图形监听器类（XXXXListener）

根据绘制的图形实现了不同的图形监听器类：直线监听器（LineListner），曲线监听器（CurveListener）等。DrawJPanel的drawListener属性表示当前同此画板绑定的监听器。当在画板上进行鼠标操作时，执行的就是drawListener的监听器动作。

1. 自定义图形类（MyXXXX）

将画板上的图形抽象成数据保存起来，包括颜色，线宽，坐标，偏移量等。每当画板上画下一个图形，画板自身的图形栈（ContentGroup）属性就加入一个自定义图形，实际上是副本的另一种表现形式。

1. 选取监听器类（ChoseListener）

遍历ContentGroup，判断每个自定义图形是否包含鼠标所在的位置，如果包含，则将此图形从图形栈里提出来，然后后续进行的各种操作针对此图形。以此实现选取的功能。

###### 4.2.2 撤销重做模块（不完整）

1. 模块功能

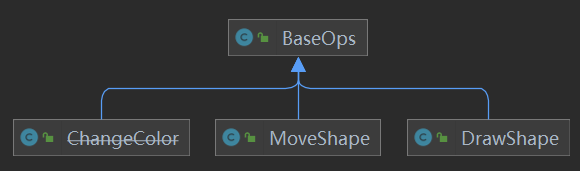
实现各种操作的撤销与重做，如画图、移动图像、改变颜色等操作。由于时间有限，目前只实现了画图的撤销与重做。

1. 实现原理

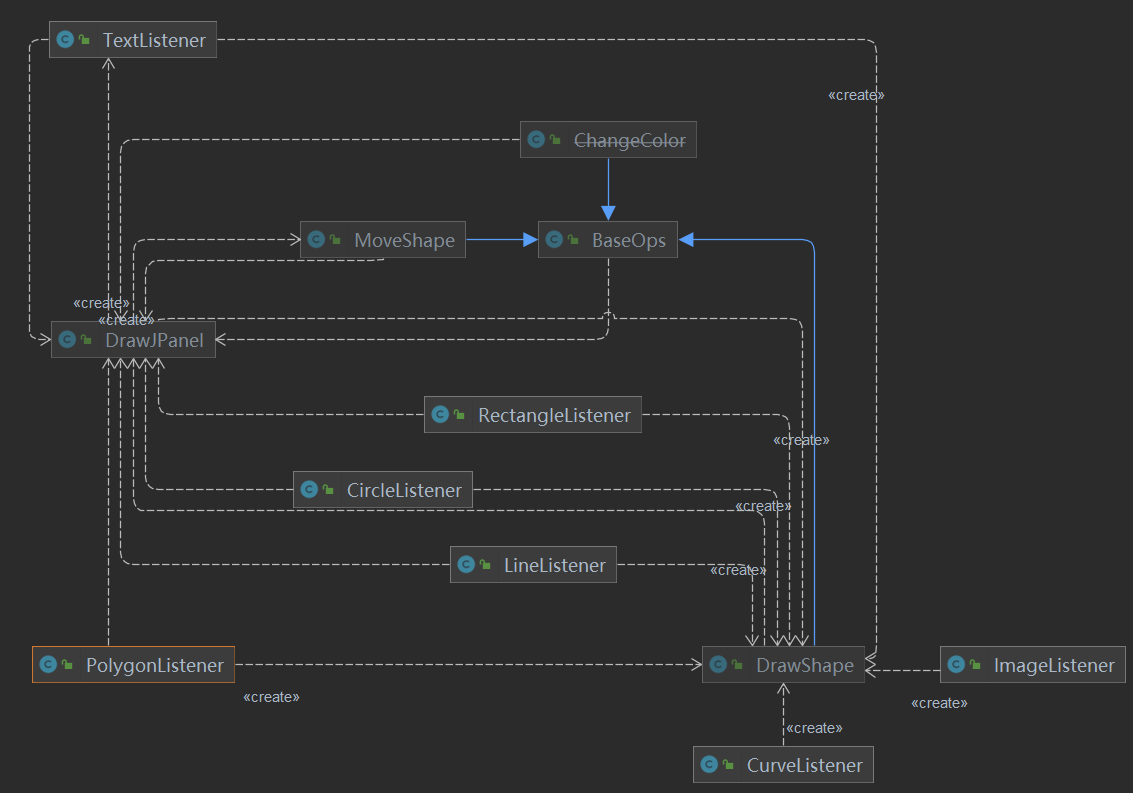
在画板类（DrawJPanel）中原有图形栈（ContentsGroup）的基础上，加入了重做图形栈（RedoContentsGroup），操作栈（Operations）以及重做操作栈（RedoOperations），通过保存操作以及相关图形的变化来实现撤销与重做。

以画图操作为例：

1. 在新绘制的图形保存到图形栈（ContentsGroup）中后，将对应操作存入操作栈（Operations）中。
2. 点击撤销时，操作栈（Operations）栈顶操作压入重做操作栈（RedoOperations），同时将图形栈（ContentsGroup）栈顶图形压入重做图形栈（RedoContentsGroup）。此时，撤销操作完成。
3. 点击重做时，与撤销操作相反，重做操作栈（RedoOperations）栈顶操作压入操作栈（Operations），同时将重做图形栈（RedoContentsGroup）栈顶图形压入图形栈（ContentsGroup）。此时，重做操作完成。
4. 模块构成类



操作类关系图



撤销模块类关系图

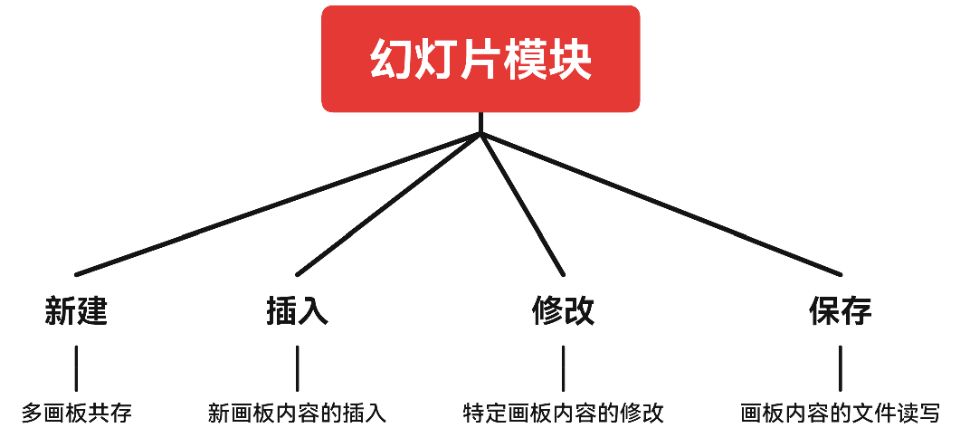
1. 父操作类（BaseOps）

包含子类操作方法的原型，以及一个清空所有相关栈的方法。

1. 子操作类（DrawShape等）

重写父类中的方法，以便于处理具体操作的撤销与重做。

###### 4.2.3 幻灯片模块



幻灯片模块功能图

1. 模块功能

实现类PPT的展示功能，以及组成PPT的画板的文件保存。

1. 实现原理

实现类PPT滑动显示是利用里JList实现，重写了JList的单元渲染器（ImageCellRenderer）使其能够渲染图标文件（ImageIcon）形成图像列表。同时利用画板图形栈和列表选中图像图形栈的同一性来实现同画板画面的同步性，以及实现通过点击选取特定画面展示在画板（DrawLPanel）的功能。滑动列表（ImageShowBoard）类实现。

画板文件保存本质是保存图形栈，利用重写toString方法，将图形的类别及参数存入文件，读取时根据参数重建图形并压入图形栈中，以实现PPT的保存与加载。

1. 模块构成类
2. 滑动列表（ImageShowBoard）类

PPT界面不可缺少的画面下拉列表。继承自JList，重写了JList的单元渲染器（ImageCellRenderer）使其能够渲染图标文件（ImageIcon）。

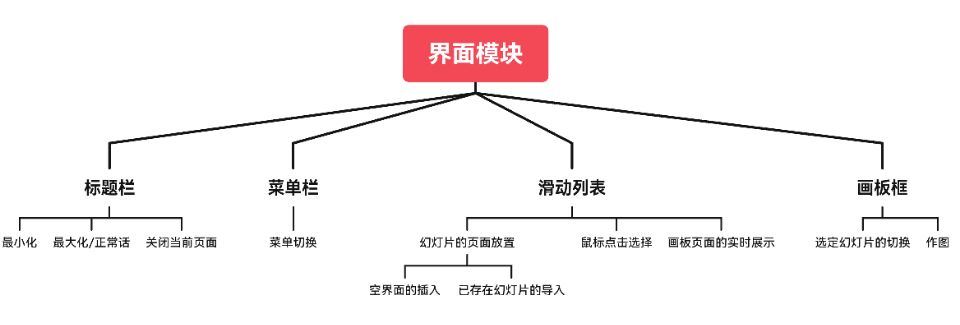
1. 画板类（DrawJPanel）
2. 画板文件类（DrawJPanelFileUtil）

包含一些将画板载入与保存的方法。

1. 自定义PPT类（MyDrawPPT）

本质上是DrawJPanel的列表，包含从文件读取生成DrawJPanel列表的函数。

###### 4.2.4 界面模块

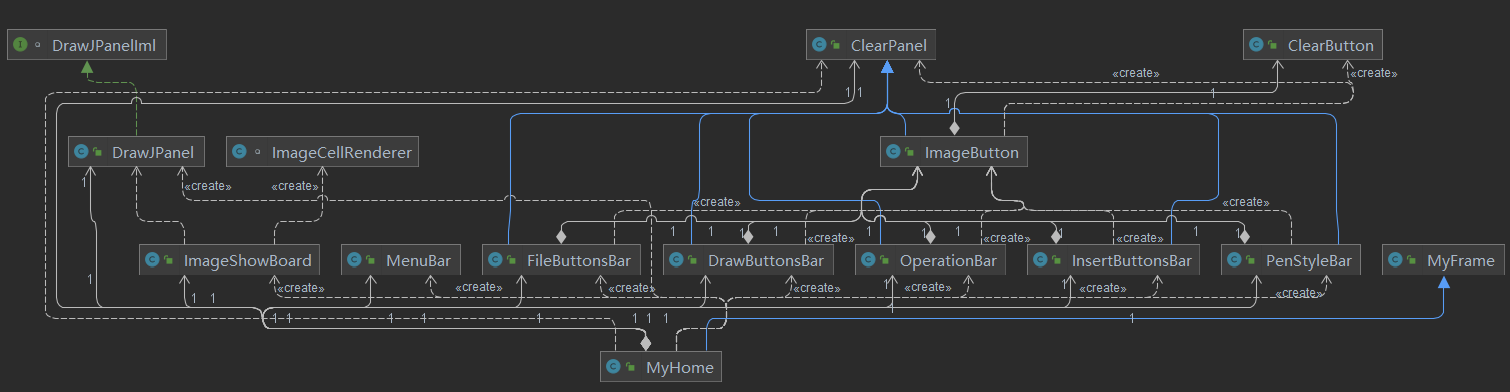


界面模块功能图

1. 模块功能

同用户进行界面交互，用户的操作板，通过用户对各种按钮的点击展示实现的各项功能。

1. 模块构成类



界面模块UML类图

1. 基础界面类包（BasePages）

此类包包含自定义容器类继承自JPanel或JFrame。在界面的实现过程中，具有一些特殊属性的容器，比如透明无背景容器，会频繁被使用。所以抽象出父类实现。

1. 菜单栏类包（MenuBar）

自定义菜单栏的实现。由于Java自带菜单过于丑陋，所以手撕简单菜单以实现功能。

1. 画板类（DrawJPanel）

实现绘画和幻灯片的单页展示功能。

1. 滑动列表（ImageShowBoard）类

PPT界面不可缺少的画面下拉列表。实现同画板画面的同步性，以及实现通过点击选取特定画面展示在画板（DrawLPanel）的功能。利用JList实现，重写了JList的单元渲染器（ImageCellRenderer）使其能够渲染图标文件（ImageIcon）形成图像列表

1. 主界面类（MyHome）

继承自MyFrame(自定义容器）。用来容纳各个子界面，用户交互的直接界面。

1. 入口类（START\_NOW）

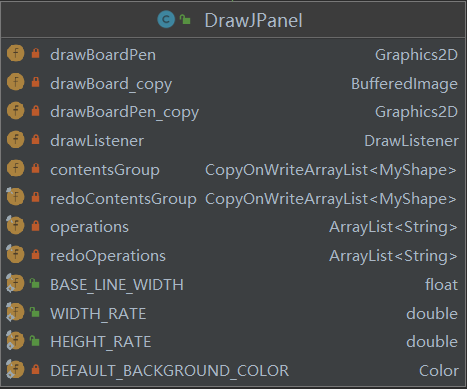
运行主界面类，程序入口。

## 五、程序关键类的实现

##### 5.1 画板模块

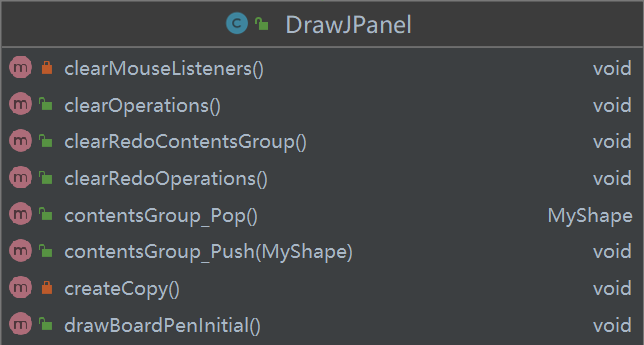
###### 5.1.1 画板类（DrawJPanel）

1. 域成员

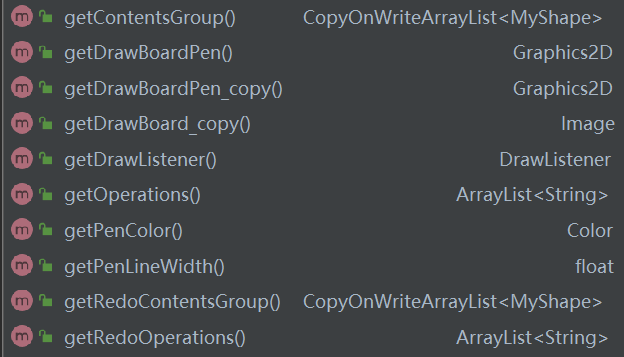


UML类域成员图

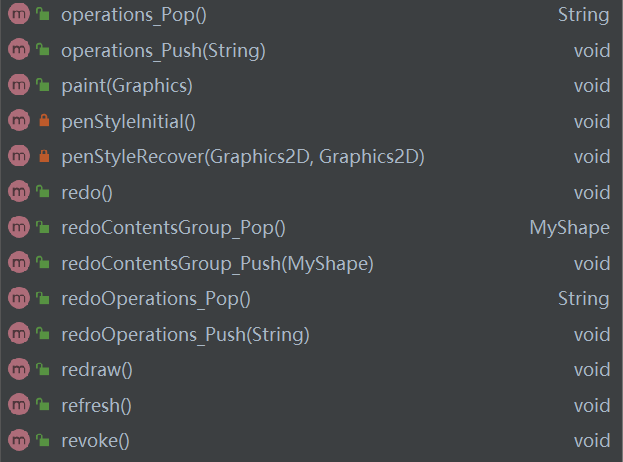
1. drawBoardPen：画笔。负责在画板上作画，当鼠标动作发生时，产生对应的图形或动作。
2. drawBoard\_copy：画板副本，执行和画板相同的绘画动作，区别是在界面大小变化时，不会丢失。
3. drawBoardPen\_copy：画笔副本，由画板副本产生的画笔。
4. drawListener：绘画监听器，当前画板绑定的监听器，负责控制画笔。
5. contentsGroup：图形栈，储存画过的图形
6. redoContentsGroup：
7. operations：
8. redoOperations：
9. BASE\_LINE\_WIDTH：基础线宽，画笔初始化时设置的线宽
10. WIDTH\_REATE, HEIGTH\_RATE：副本高度宽度占整个屏幕的百分比
11. DEFAULT\_BACKGROUND\_COLOR：默认背景颜色
12. 成员方法



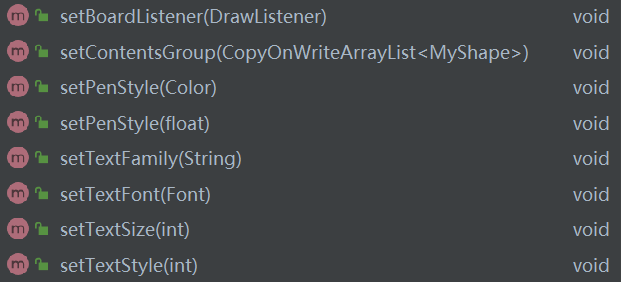
1. clearMouseListeners()：清空本画板的全部绘画监听器，将原来的监听器同本画板解绑。涉及到的鼠标监听器有MouseListener，MouseMotionListener。执行过程中，调用removeMouseXXXListener
2. clearOperations():清空操作栈
3. clearRedoContentsGroup():清空重做图形栈
4. createCopy()：私有函数，创造本画板的副本以及本画板画笔的副本。首先调用系统函数获得屏幕大小，然后创造特定比例的副本。同时调用createGraphics()函数获得副本画笔。
5. drawBoardPenInitial(): 画笔初始化，在父窗口显示后调用。完成的任务有：将父类画笔赋值给本身画笔。创建画板副本，同时创建副本画笔。调用getGraphics()函数获得窗口的画笔。然后根据副本是否为空采取不同的操作，副本为空咋创造副本，画笔初始化。否则说明是界面重绘了，则副本画笔样式还原给原本画笔。



1. 各种域成员的getter获取函数。



1. paint(Graphics):窗口改变大小之后，调用此函数。原先绑定的Graphic全部换新了，指向了另一个对象。重新绑定画笔，样式还原。首先执行父类的paint函数将组件重画，然后用画笔载入副本。然后画笔初始化，重新获得窗口画笔。
2. penStyleInitial()：初始化画笔，设置画笔的相关属性，使线条更加圆滑，减小锯齿。看起来更美观。
3. redo():根据操作栈中的操作，实例化相应操作的类，调用类中redo方法。
4. redraw():重画，画笔更新，副本更新。遍历图形栈，将储存的图形在副本上画一遍。
5. refresh():刷新本画板，将副本内容载入画板。
6. revoke():根据操作栈中的操作，实例化相应操作的类，调用类中revoke方法。

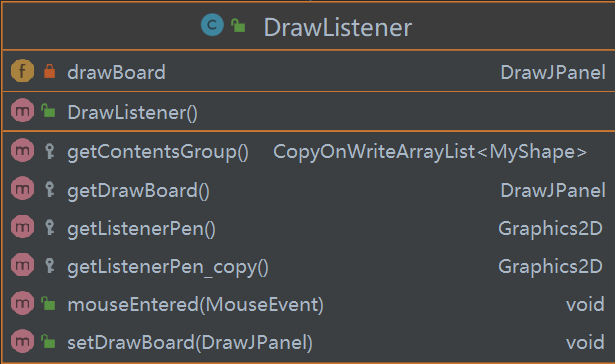


1. 各属性的setter



1. toString():文件读写时调用。

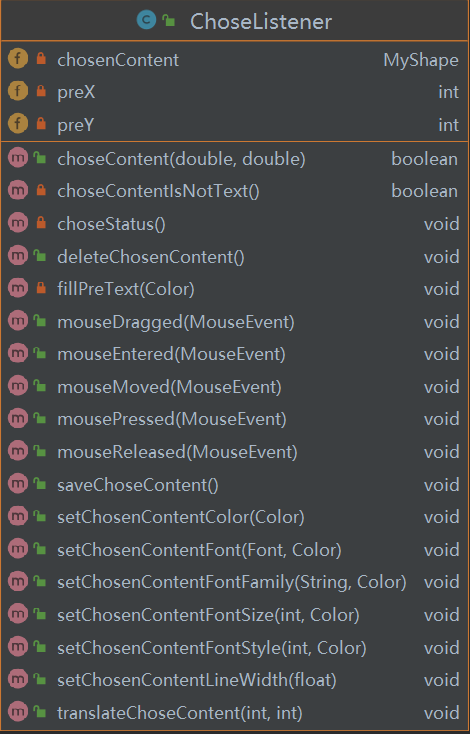
###### 5.1.2 绘画监听器类（DrawListener，XXXXListener）



1. drawBoard:私有成员，拥有该监听器的画板。
2. getXXXX：获得该监听器的画笔副本等属性
3. setDrawBoard：设置拥有该监听器的画板
4. mouseEntered():监听器动作，鼠标进入画板范围内，则改变图标为十字形。

图形绘制监听器均继承此类，根据不同图形的绘制方法设置不同的图形动作。

###### 5.1.3 选取监听器类（ChoseListener）

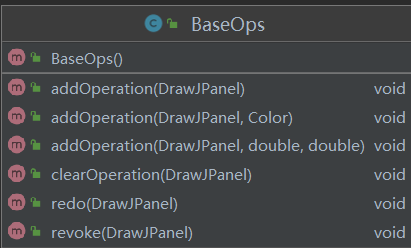


1. 域成员
2. choseContent：选取器当前选中的图形栈中的对象
3. preX，preY:拖动的图形之前的横纵坐标。用于图形拖动效果的实现。
4. 成员函数
5. choseContent(double，double):选择制定坐标出的图形。遍历图形栈，如果图形含有当前坐标，则选中此图形。首先将该图形从图形栈中移除，然后重画副本，后将此图形放到图形栈栈顶。因为整个过程中没有动原本，所以图形并没有在画面上消失。但是副本上没有了该图形，所以当拖动过程中不断载入副本，不会显示该图形，就达成了拖动的动画效果。
6. deleteChoseContent():删除选中的图形。将选中的图形从图形栈栈顶删除，然后重画，更新副本。
7. MouseDragged():鼠标拖动监听器，不断获取当前鼠标坐标。重载副本，后在当前坐标原本上画出图形。
8. MousePressesed():鼠标按下监听器，根据按下的坐标调用choseContent函数，获取选择图形。
9. MouseReleased()：鼠标松开监听器，调用保存函数。
10. MouseEntered(),MouseMoved():调整鼠标类型的函数。
11. saveContent()函数：保存当前选中的图形，实际上是把选中的图形在副本上重画出来。
12. setCChoseContentXXXX():设置当前选中图形的对应属性
13. translateChoseContent():平移当前选中图形，修改当前图形的偏移量属性。

##### 5.2 撤销重做模块

###### 5.2.1 操作类（BaseOps）

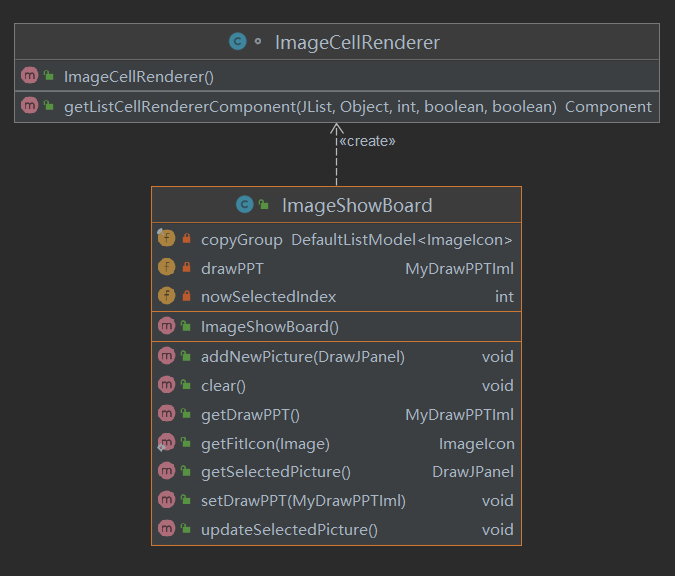
1. 成员方法



1. addOperation():执行操作时将操作行为压栈
2. clearOperation():清空图形栈以外的所有栈
3. redo(执行撤销操作时的处理
4. revoke():执行重做操作时的处理

##### 5.3 幻灯片保存模块

###### 5.3.1 滑动列表类（ImageShowBoard）



1.ImageCellRenderer

ImageShowBoard的单元渲染器，当ImageShowBoard重新加载的时候，调用getListCellRendererComponent函数对ImageShowBoard的每一个列表子项进行渲染获得图像。构造函数设置每个子项的字体。

1. ImageShowBoard

继承自JList类，以实现列表的功能。

域成员：

1. copyGroup是JList默认模型，放置列表的子项。
2. drawPPT是同本下拉列表绑定的MyDrawPPT类。
3. nowSelectIndex记录当前选中的列表子项。

成员函数：

（1）addNewPicture()为本列表添加子项，然后渲染到图像中。

（2）clear()清楚列表，图像展示为空。

（3）getDrawPPT()获得本列表绑定的MyDrawPPT类。

（4）getFitIncon()静态成员函数，将过大的image图像转换成大小合适ImageIcon图像。

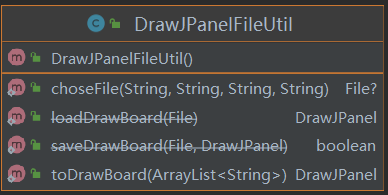
（5）getSelectPicture()获得选中的画板类用来设置界面上的画板块，实现点击选中PPT的关键函数。

（6）setDrawPPT()将外键MyDrawPPT同本列表绑定，同时将内部的每个画板的读本渲染成列表子项在图像上展示。

（7）updateSelectPicture()更新当前选中的图像，负责将因为画板上绘画造成的图像改变传回到列表子项上。以实现类实时展示的效果。

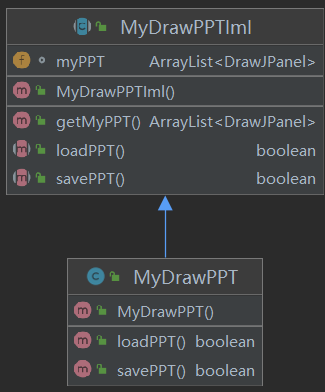
###### 5.3.2 画板文件类（DrawJPanelFileUtil）

1. 成员方法



1. choseFile():调用系统的文件选择类。选择文件，形成文件对象
2. loadDrawBoard():读取画板，用于测试单张PPT的读取，已弃用
3. saveDrawBoard():保存画板，用于测试单张PPT的保存，已弃用
4. toDrawBoard():String转换成画板，读取整个PPT时调用

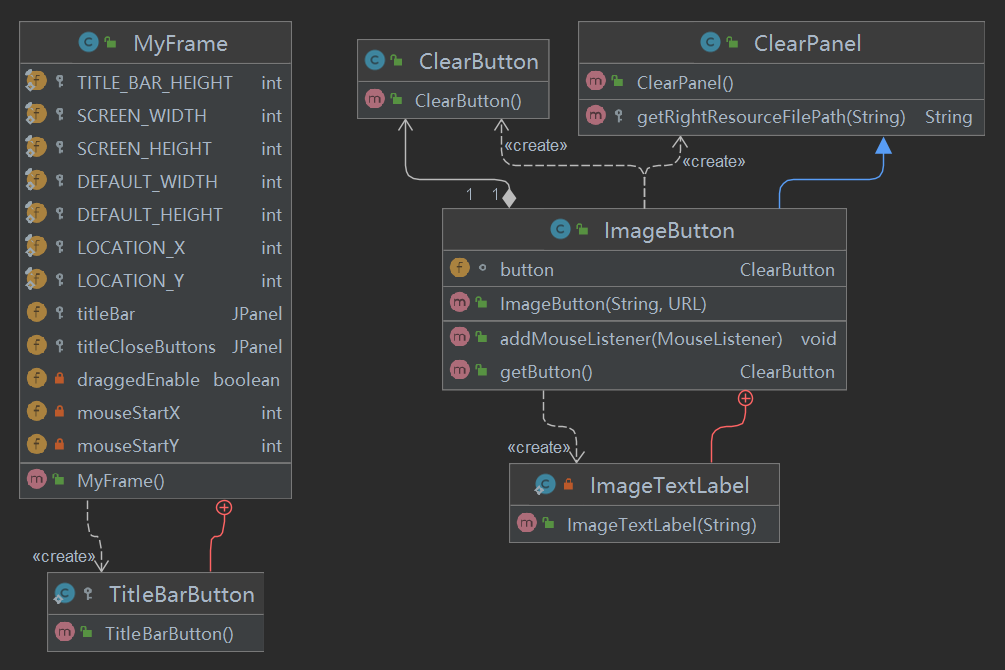
###### 5.3.3 自定义PPT类（MyDrawPPT）



1. 域成员
2. myPPT:构成PPT的画板数组
3. 成员方法
4. getMyPPT():获得画板数组
5. loadPPT():从一个.myppt文件里，读取获得多个DrawJPanel实例，存入myPPT。
6. savePPT():非空myPPT形成.myppt文件。存到电脑里。

##### 5.4 界面模块

###### 5.4.1 基础界面类包（BasePages）



1. MyFrame

自定义框架，包含界面的各个距离信息：起始坐标点，起始大小，屏幕大小，默认的标题栏大小。

1. ClearPanel

继承自JPanel调整了透明度。同时实现了根据类加载器获得正确资源文件路径的函数。

1. ClearButton

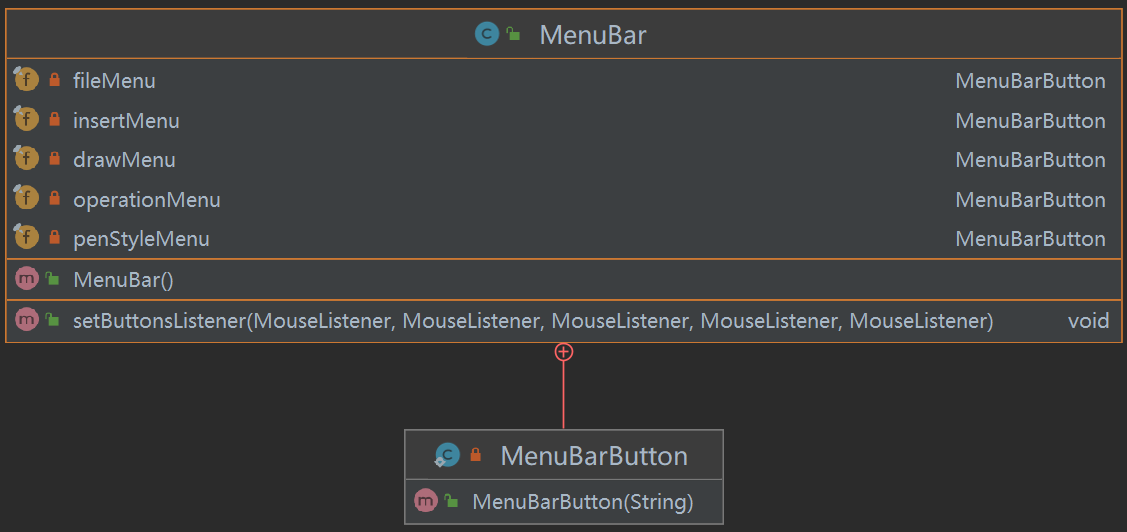
继承自JButton，调整了透明度，背景，边框，焦点。

1. ImageButton

由装在ClearPanel中的自定义JLabel和ClearButton组成。重写了addMouseListener()函数，使得添加监听器时会为按键和标签都添加。构造时必须提供按钮名字和按钮图像。

###### 5.4.2 菜单栏类包（MenuBar）

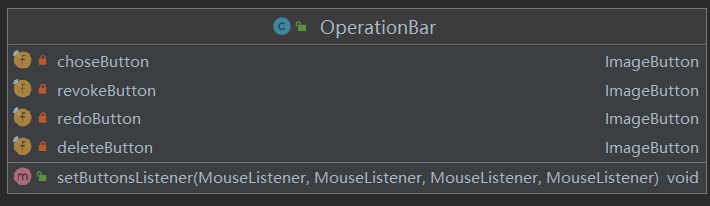
1. 菜单切换栏（MenuBar）



MenuBarButton内部定义按钮类，继承自ClearButton，然后调整了字体大小，字体颜色，按钮大小。内部域成员均为此类按钮。

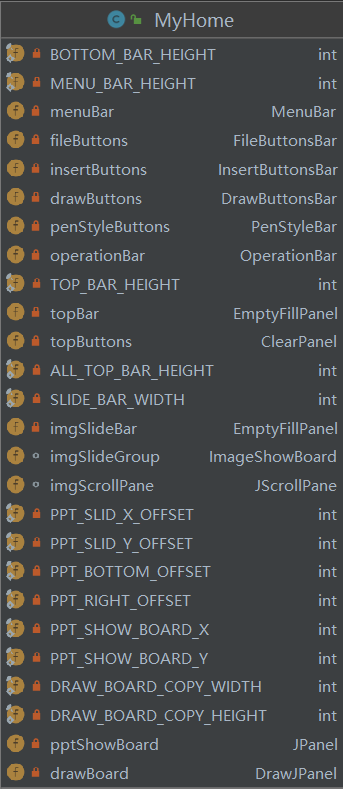
setButtonsListener(MouseListener.....)：设置内部按钮的鼠标监听器。

1. 菜单按钮框（XXXXBar）

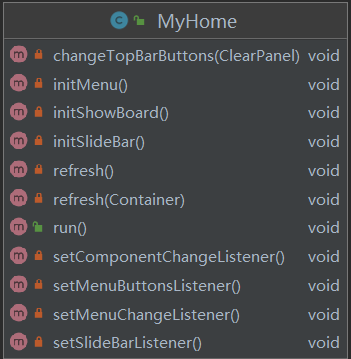


以OperationsButtonBar为例，其他菜单按钮框相同：内部域成员为ImageButton的实例对象。然后调用setButtonsLisetener函数为各个按钮添加监听器。

###### 5.4.3 主界面类



域成员函数均为各个界面的实例对象，以及把不同子界面的距离常量。



成员函数包括和界面的初始化函数，以及监听器设置函数。同时界面的可视化函数（run（））

1. 程序入口

package com;

public class START\_NOW {

public static void main(String[] args) {

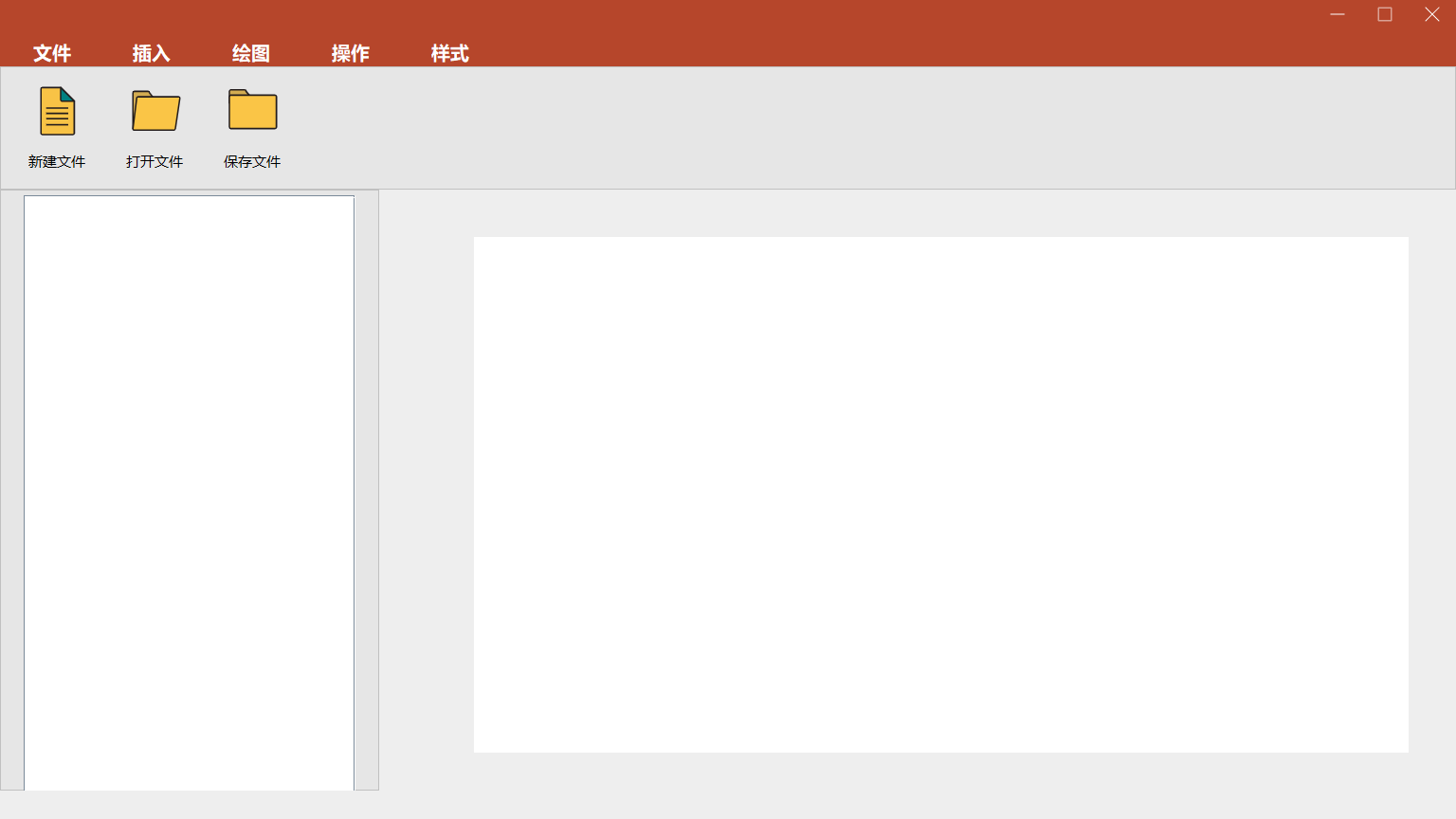
new MyHome().run();

}

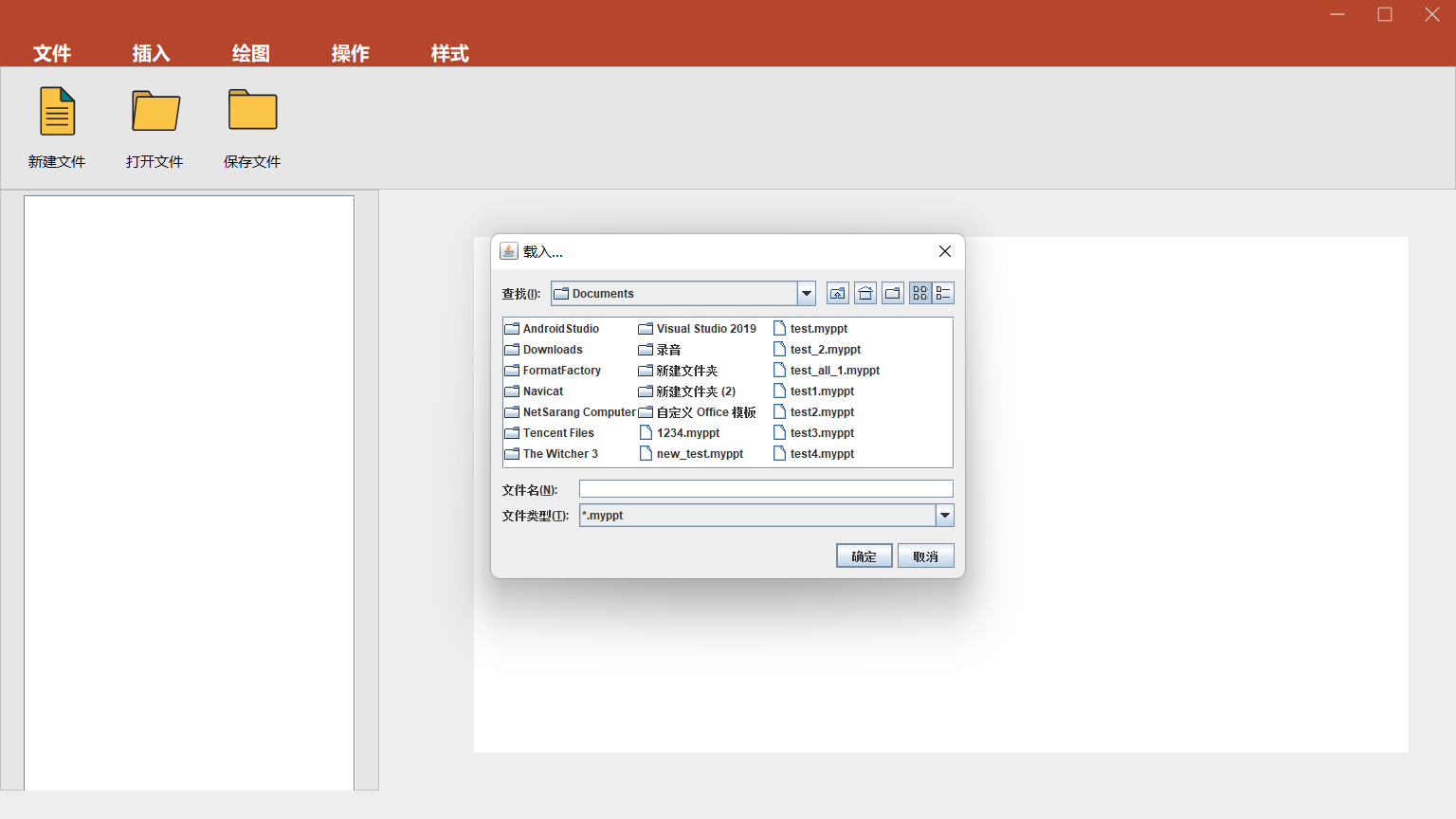
}

## 六、软件的主要界面截图

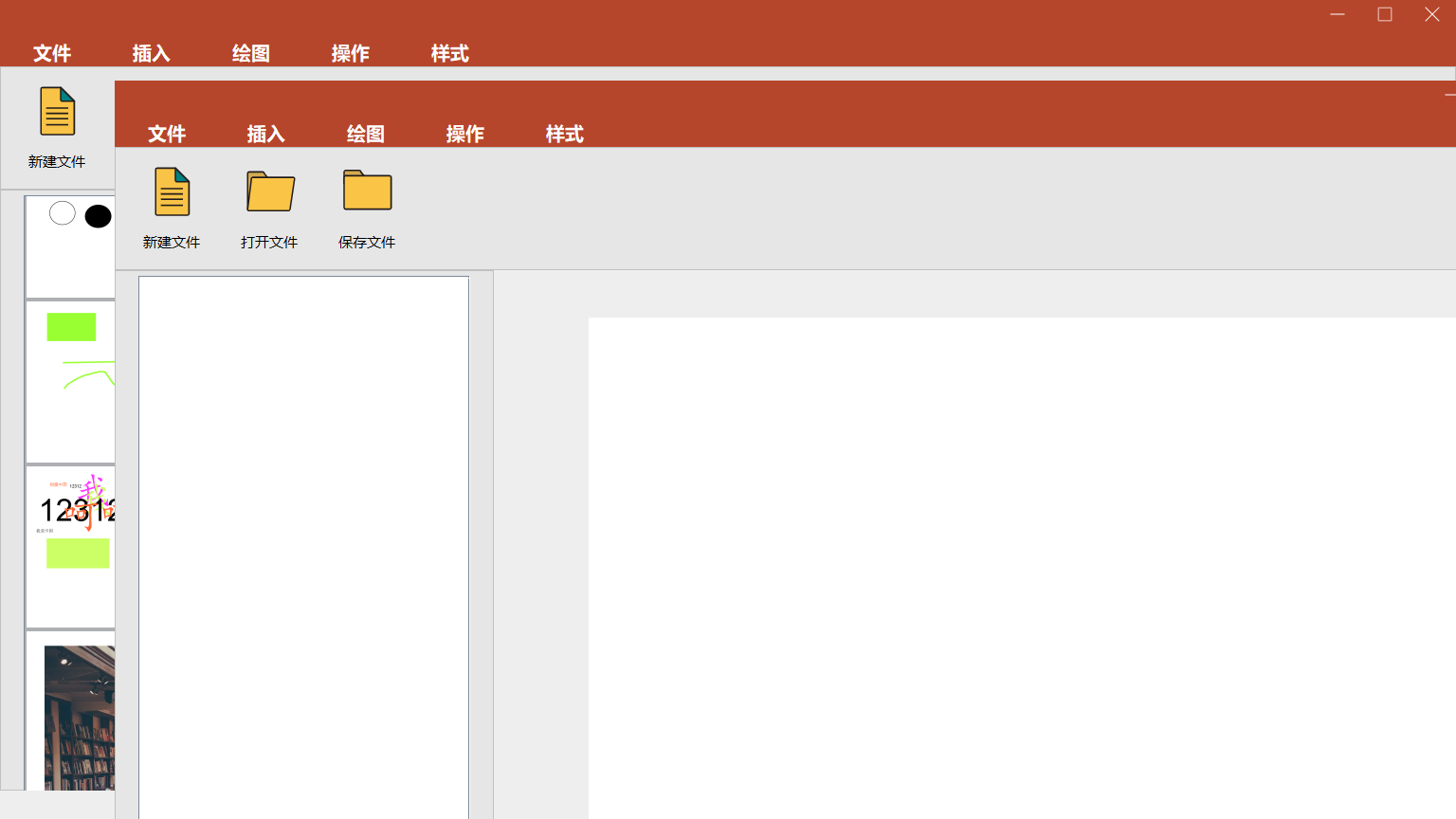
1. 主页面，程序进入首先呈现的界面



1. 文件选择界面，当点击“打开文件”或“保存文件”时，选择文件路径。

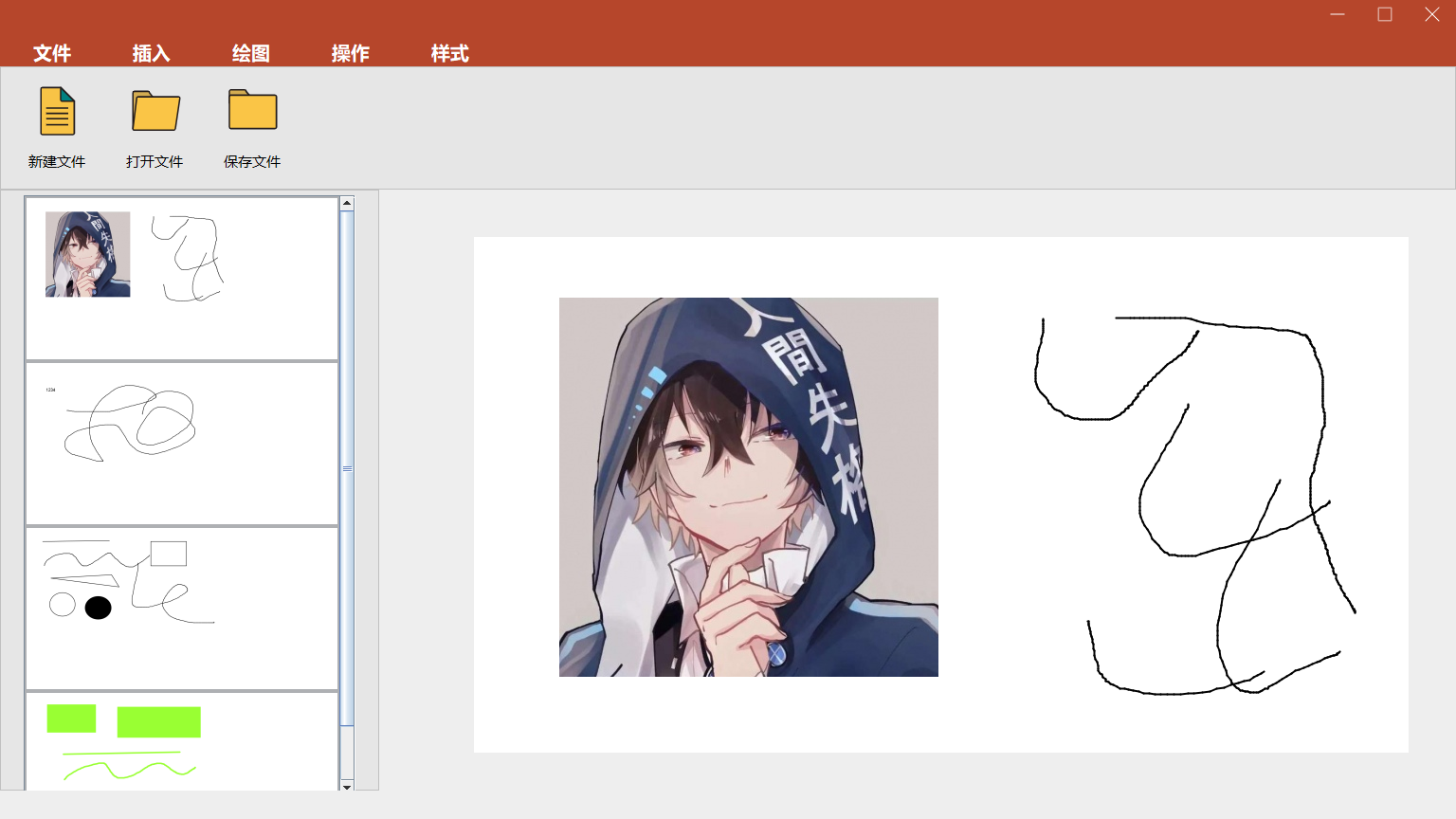


打开（保存）文件

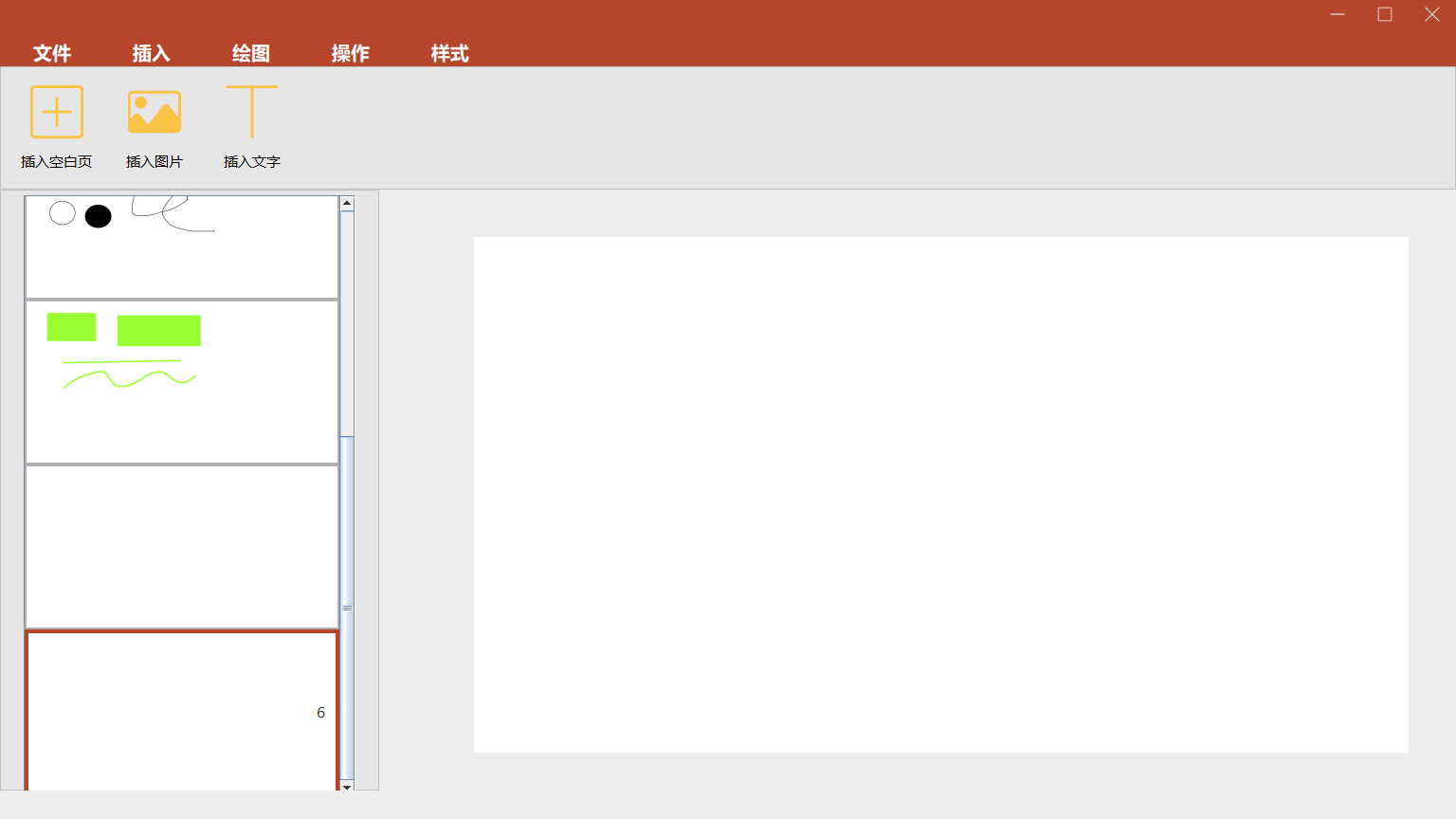


新建文件

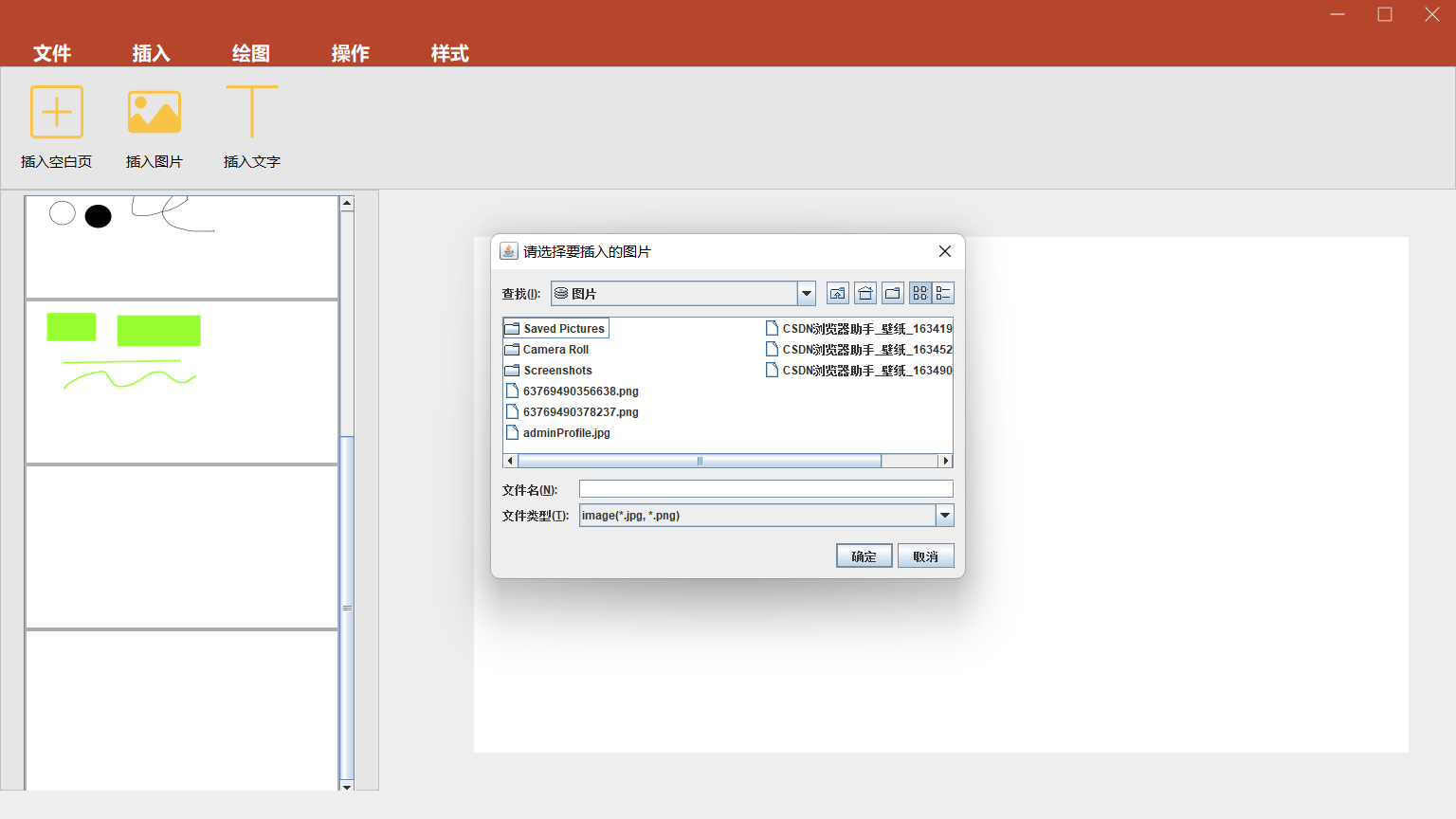
1. 幻灯片展示界面，当打开正确的幻灯片文件（.myppt）后，幻灯片在界面上出现。点击不同的页面可以切换



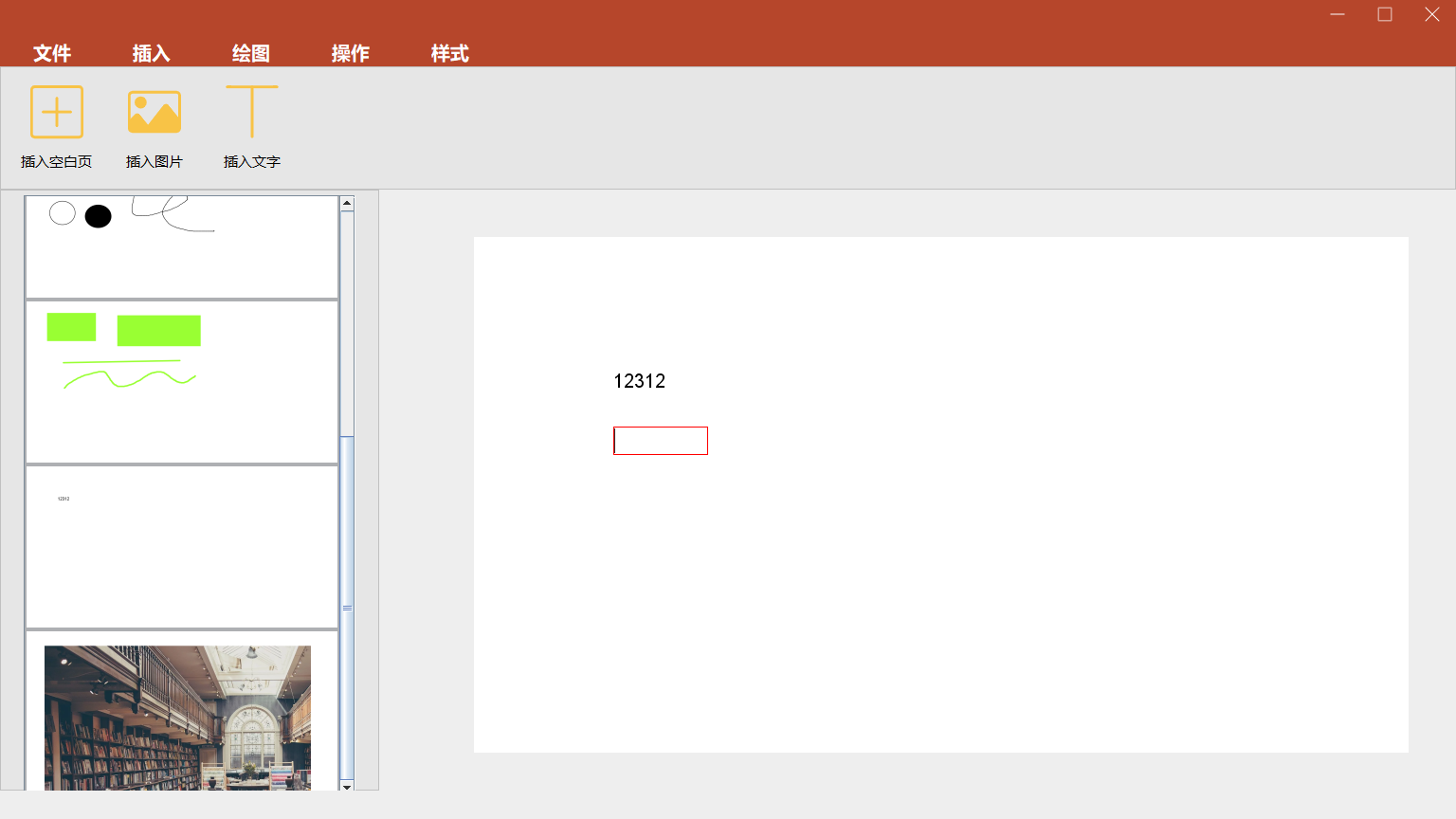
1. 幻灯片插入界面。点击菜单栏“插入”，切换到插入菜单。点击不同按钮插入不同功能。



插入空白页

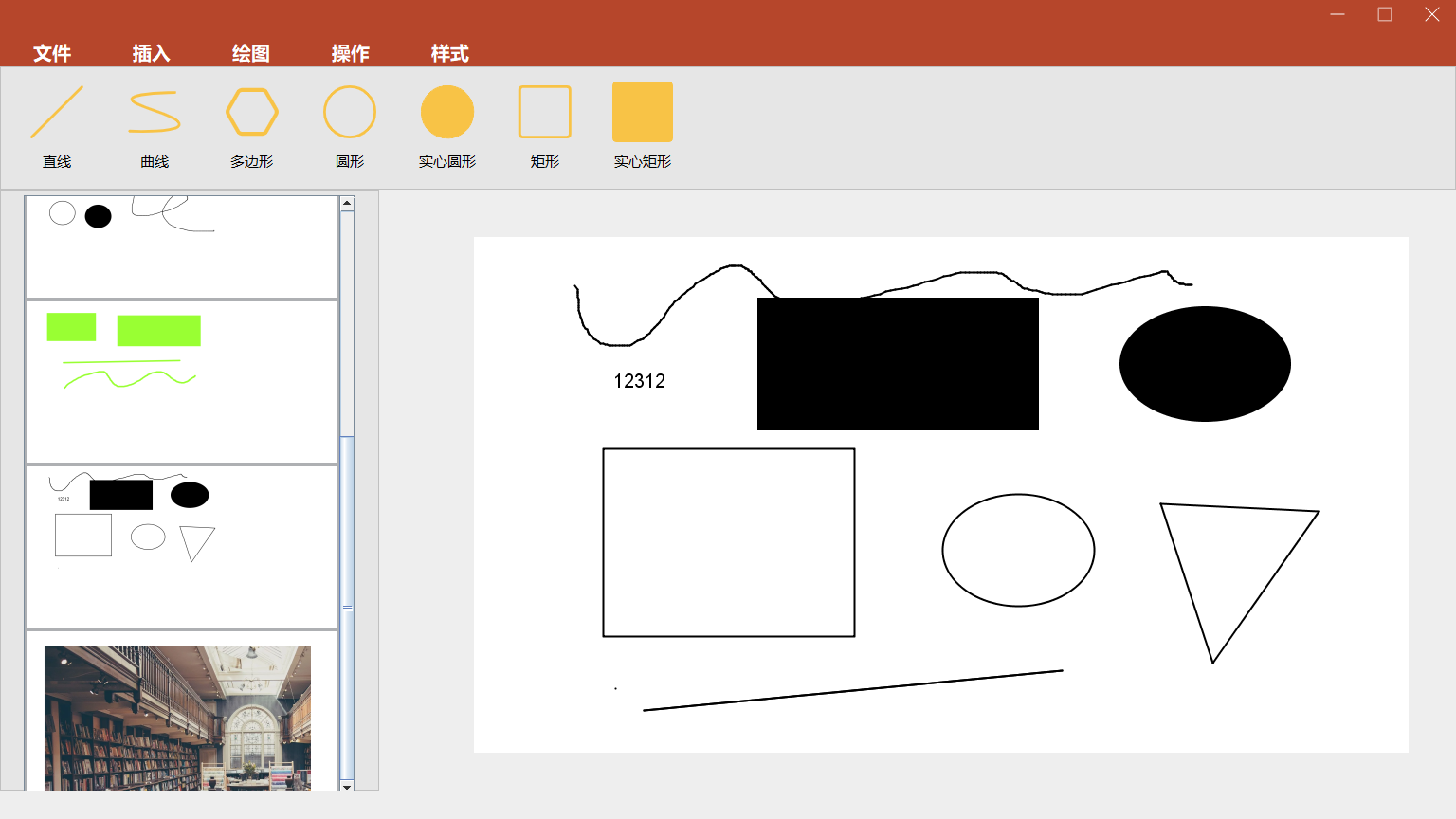


插入图片

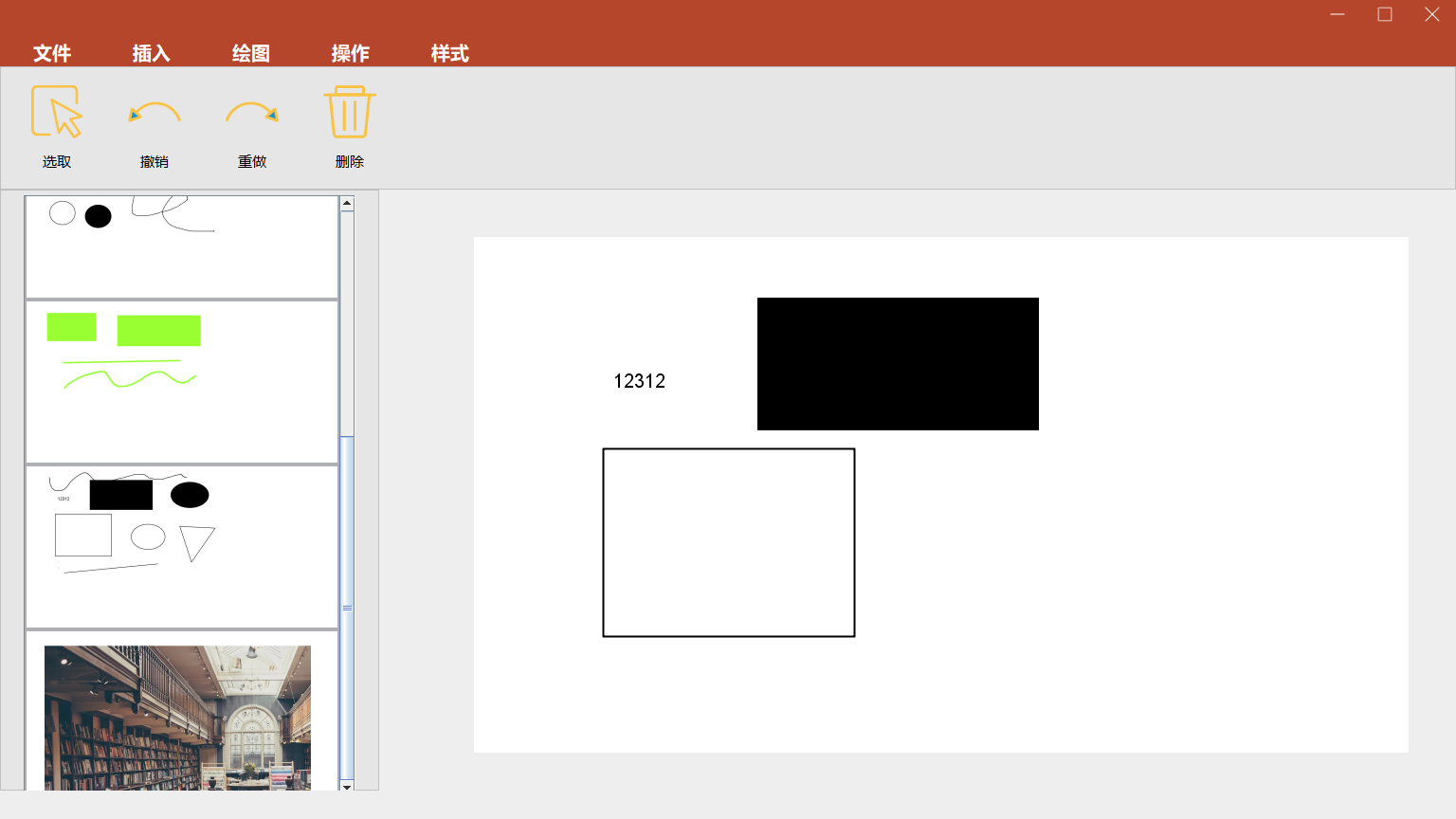


插入文字

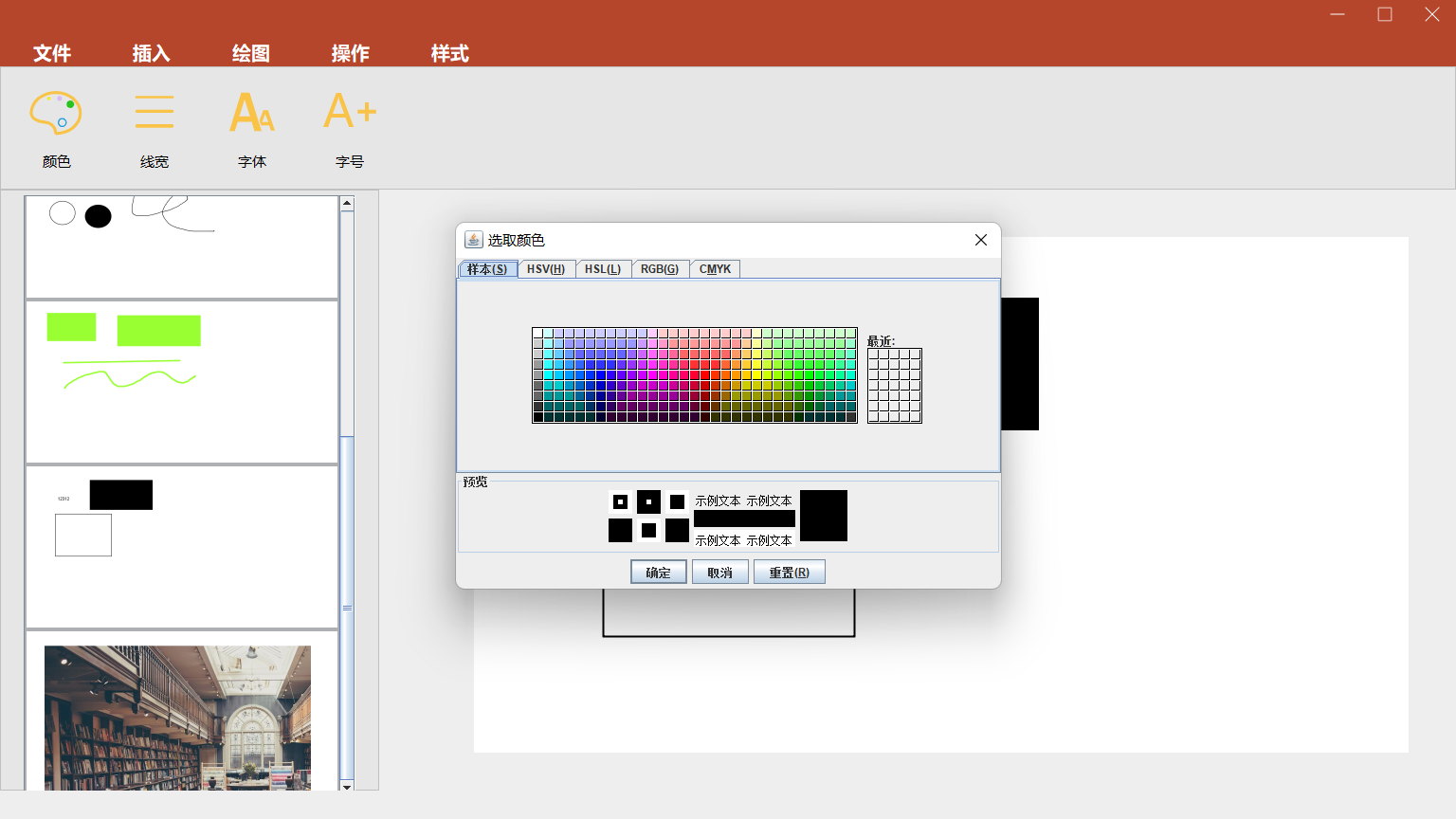
1. 幻灯片绘画界面。点击菜单栏“绘画”，切换到绘画菜单。点击不同按钮进行不同图形的绘制。



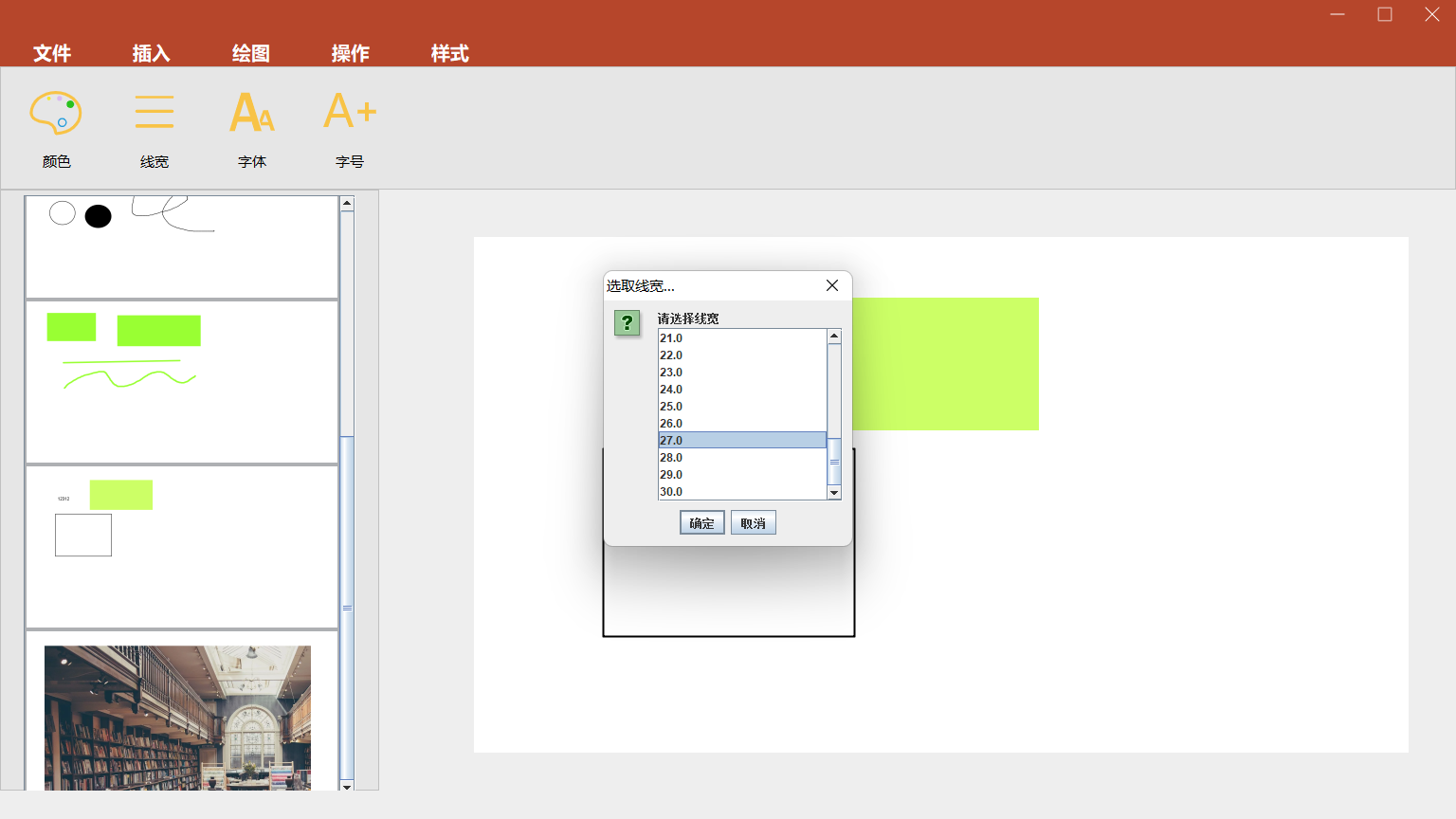
1. 幻灯片绘画界面。点击菜单栏“操作”，切换到操作菜单。点击不同按钮进行不同操作。



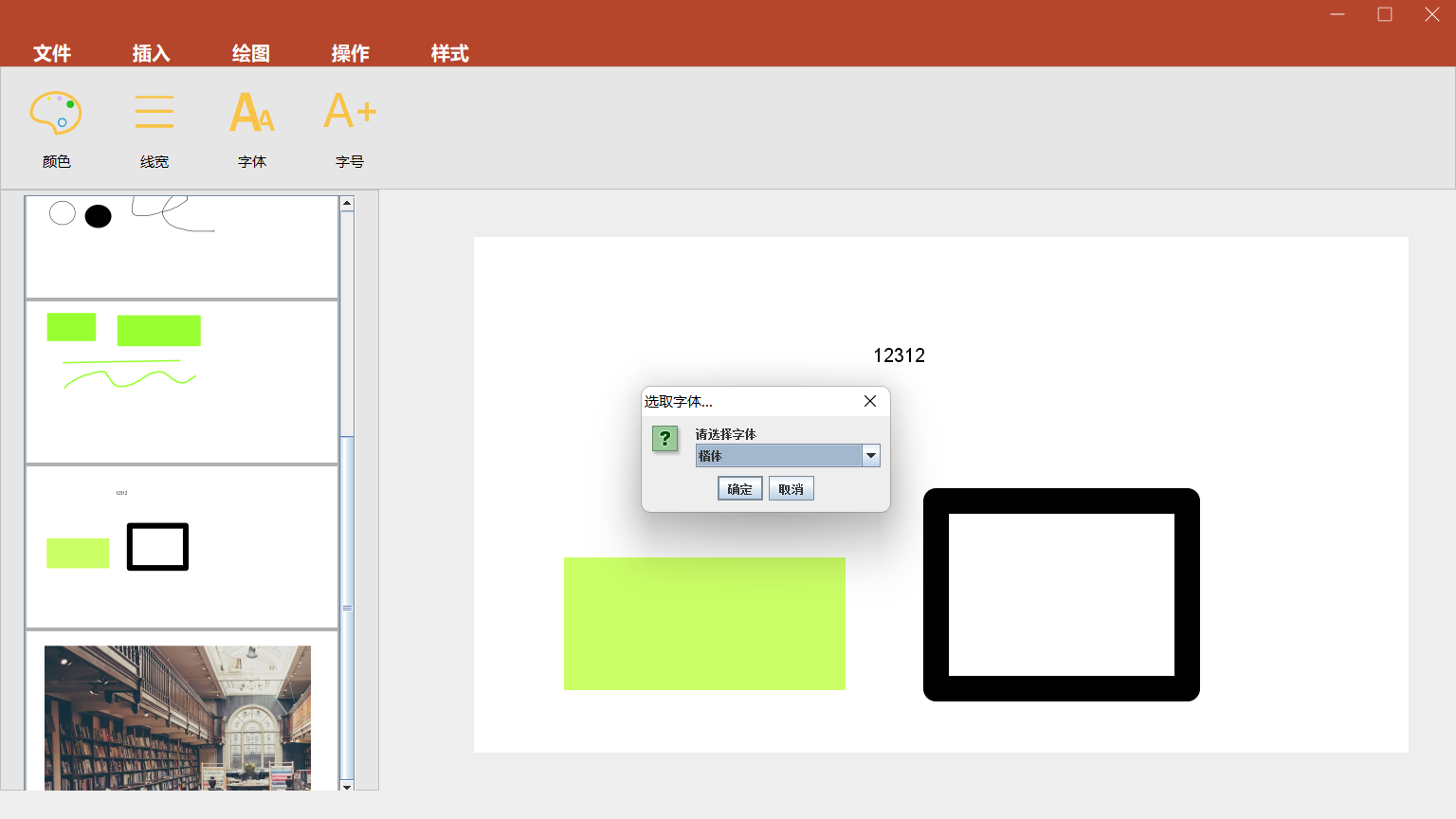
1. 幻灯片绘画界面。点击菜单栏“样式”，切换到样式菜单。点击不同按钮切换画笔样式或选中图形的样式。



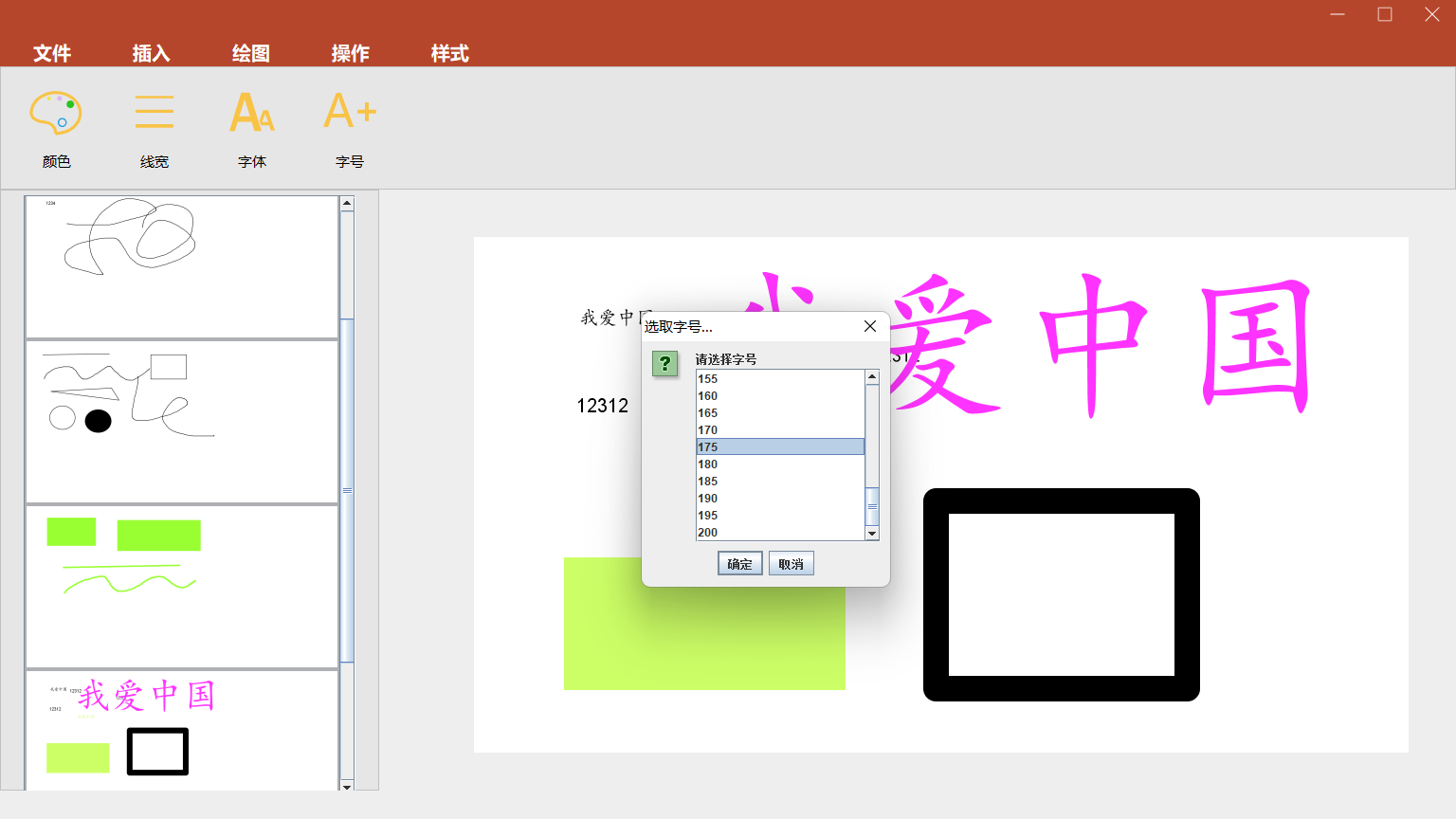
选择画笔（选中图形）颜色



选择画笔（选中图形）线宽



选择文本（选中文本）字体



选择文本（选中文本）字号

## 七、任务分工与联系方式：

|  |  |  |
| --- | --- | --- |
| 姓名 | 完成内容 | 联系方式(手机号) |
| 白千一 | 分配任务，完成整体架构的设计。完成画板模块自定义图形（MyShape）及其子类，选取器（ChoseListener），画板（DrawJPanel）部分，绘画监听器（DrawListener）及其子类的设计以及曲线、字体监听器编写。完成幻灯片模块下拉列表（ImgDrawBoard）的设计与编写。完成界面模块界面的设计与基础界面（BasePages）编写。完成界面的部分组装。编写报告相关部分。 | 15369919490 |
| 李子豪 | 完成组长分配的任务。编写圆形（CircleListener）矩形（RectangleListener）及其填充图形绘画监听器。编写画板（DrawJPanel）部分。完成界面模块部分菜单界面（MenuBar）的编写。完成界面的部分组装。编写文档相关部分。 | 13151299873 |
| 洪鑫城 | 完成组长分配的任务。编写直线（LineListener）多边形（PolygonListener）以及图像（ImageListener）绘画监听器的编写。完成画板模块换（DrawJPanel）的文件读写。完成幻灯片模块自定义PPT的文件读写。完成撤销重做模块全部。完成界面模块部分菜单界面（BasePages）的编写。完成界面的组装。编写文档相关部分。 | 18807758555 |

##### 7.1白千一代码:

//绘画监听器父类

/\*\*

\* 同画板绑定的鼠标监听器。

\*/

public class DrawListener extends MouseAdapter {

private DrawJPanel drawBoard; //同此监听器绑定的画板

protected DrawJPanel getDrawBoard() {

return drawBoard;

}

/\*\*

\* 一定注意本方法的执行时机，一定要在画板的初始化后执行。一般供给画板的`setBoardListener()`执行

\* @param drawBoard 绑定的画板

\*/

public void setDrawBoard(DrawJPanel drawBoard) {

this.drawBoard = drawBoard;

}

/\*\*

\*

\* @return 获得本监听器画笔，一般供子类调用

\*/

protected Graphics2D getListenerPen() {

return drawBoard.getDrawBoardPen();

}

/\*\*

\*

\* @return 获得本监听器画笔副本，一般供子类调用

\*/

protected Graphics2D getListenerPen\_copy() {

return drawBoard.getDrawBoardPen\_copy();

}

protected CopyOnWriteArrayList<MyShape> getContentsGroup() {

return drawBoard.getContentsGroup();

}

@Override

public void mouseEntered(MouseEvent e) {

drawBoard.setCursor(new Cursor(Cursor.CROSSHAIR\_CURSOR));

}

}

//选择监听器接口

/\*\*

\* 选择监听器的接口，方便查找函数

\*/

public interface ChoseListenerIml {

boolean choseContent(double x, double y);

/\*\*

\*

\* @param tx 横坐标平移量

\* @param ty 总坐标平移量

\*/

void translateChoseContent(int tx, int ty);

/\*\*

\* 删除选中的图形

\*/

void deleteChosenContent();

/\*\*

\* 更新选中的图形颜色。

\*

\*/

void setChosenContentColor(Color newColor);

/\*\*

\* 更新选中的图形线宽

\*/

void setChosenContentLineWidth(float newLineWidth);

void setChosenContentFont(Font newFont, Color backgroundColor);

// /\*\*

// \* 改变选中的文本内容

// \*/

// void setChosenText(String newText);

/\*\*

\* 将选中后的图形在副本上画出来

\*/

void saveChoseContent();

}

其余部分见源代码文件PaintBoard

##### 7.2李子豪代码：

//圆形监听器

public class CircleListener extends DrawListener {

int originX, originY;

int startX, startY;

int endX, endY;

boolean isFilled;

Ellipse2D ellipse;

public CircleListener(boolean isFilled) {

this.isFilled = isFilled;

}

//最后鼠标释放的点B和最开始点击的点A可能不是A左上B右下的关系（人眼来看，实际上在画板的坐标系里是A右下，B左上）

//但是只有A左上B右下的情况才能画出图形，所以调整一下

private void swapXY(){

if (startX < endX){

if(startY > endY){

int temp = startY;

startY = endY;

endY = temp;

}

} else{

int temp = startX;

startX = endX;

endX = temp;

if(startY > endY){

temp = startY;

startY = endY;

endY = temp;

}

}

}

private void drawCircle(MouseEvent e, boolean need\_backward){

endX = e.getX(); endY = e.getY(); //记录终点位置

startX = originX; startY = originY;

swapXY();

ellipse = new Ellipse2D.Double(startX, startY, endX-startX, endY-startY);

if(isFilled){

getListenerPen().fill(ellipse);

if(need\_backward){

getListenerPen\_copy().fill(ellipse);

}

} else{

getListenerPen().draw(ellipse);

if(need\_backward){

getListenerPen\_copy().draw(ellipse);

}

}

}

@Override

public void mousePressed(MouseEvent e) {

originX = e.getX(); originY = e.getY();

startX = originX; startY = originY; //记录起点位置

}

@Override

public void mouseReleased(MouseEvent e) {

drawCircle(e, true);

getContentsGroup().add(new MyCircle(

ellipse,

startX,

startY,

getListenerPen().getColor(),

((BasicStroke)getListenerPen().getStroke()).getLineWidth(),

isFilled

));

//记录操作

DrawShape drawShape = new DrawShape();

drawShape.addOperation(getDrawBoard());

}

@Override

public void mouseDragged(MouseEvent e) {

getDrawBoard().refresh();

drawCircle(e, false);

}

}

其余部分见源代码文件PaintBoard

##### 7.3洪鑫城代码：

//多边形监听器

public class PolygonListener extends DrawListener {

private static final int lastDict = 10; //判定构成多边形的距离参数

Vector<Integer> x = new Vector<>(); //x坐标集合

Vector<Integer> y = new Vector<>(); //y坐标集合

int n = 0; //点的个数

@Override

public void mousePressed(MouseEvent e) {

//记录点击的点坐标

int nowX = e.getX(), nowY = e.getY();

x.add(nowX);

y.add(nowY);

n++;

//当前点与第一个点的距离在lastDict以内，则形成收尾相连的多边形

int startX = x.get(0), startY = y.get(0);

if ((nowX - startX) \* (nowX - startX) + (nowY - startY) \* (nowY - startY) <= lastDict \* lastDict && n > 2) {

//最后一个点与第一个点坐标相同

x.set(n - 1, startX);

y.set(n - 1, startY);

//最后一次选点多画的线删掉

getDrawBoard().refresh();

//画出多边形最后一条线

Line2D line = new Line2D.Double(x.get(n - 2), y.get(n - 2), startX, startY);

getListenerPen().draw(line);

getListenerPen\_copy().draw(line);

try { //将画的的多边形送入图形栈

getContentsGroup().add(new MyPolygon(

x,

y,

getListenerPen().getColor(),

((BasicStroke) getListenerPen().getStroke()).getLineWidth()

));

} catch (IndexOutOfBoundsException ex) {

System.out.println("多边形越界");

ex.printStackTrace();

}

//记录操作

DrawShape drawShape = new DrawShape();

drawShape.addOperation(getDrawBoard());

//初始化

n = 0;

x.clear();

y.clear();

}

//将点连线

if (n > 1) {

Line2D line = new Line2D.Double(x.get(n - 2), y.get(n - 2), nowX, nowY);

getListenerPen().draw(line);

getListenerPen\_copy().draw(line);

}

}

@Override

public void mouseMoved(MouseEvent e) {

//当多边形未完成时线条随鼠标移动

if (n != 0) {

getDrawBoard().refresh();

Line2D line = new Line2D.Double(x.get(n - 1), y.get(n - 1), e.getX(), e.getY());

getListenerPen().draw(line);

}

}

}

其余部分见源代码文件PaintBoard

## 八、Java课程学习心得与改进意见

白千一：

本次Java大作业要求制作一款类似微软PowerPoint幻灯片的播放和绘制软件。作为组长，要带领组员完成这么一项很陌生的任务，压力还是很大的。但是在查阅了大量资料和开会讨论后，终于设计出了整个软件的系统框架。然后就是软件的分模块编写。编写过程中，很多系统类都不知道如何调用，在翻阅Java文档浏览Java源码以及阅读他人博客的过程中，增长了大量关于Java的知识，加深了对Java“万物皆对象”的理解。

同时，在本次大作业过程中，我主动提出使用Git来协同工作。对于之前并没有使用过Git的我来说，这无疑也是一场挑战。但是在使用的过程中，我发现了这种版本控制工具的优越性以及支持多人协作的方便性。在完成了Java大作业后，我已经能够熟练掌握这项技能，这是大作业带给我的。无疑为后来的学习或工作能够带来极大便利。

但是在这个过程中，也发现了自己的很多问题。敲代码时会很粗心，经常出现一些因此导致的小bug。过于追求架构的逻辑性，会导致项目增加大量冗余代码。这些都是在以后的学习过程中要提升的地方。

提点小意见：赶上考试，时间有点紧张。建议时间根据实际情况稍作调整。JavaGUI已经不是当前社会界面的主流工具了，建议使用更为普及的Web来完成界面部分。

李子豪：

本次Java大作业我们这组做的是幻灯片制作播放软件，刚开始拿到题目我们组员都不知道如何下手，经过大家一起查找资料，并且开会讨论，我们确定了设计的设计目标以及具体实现方式，包括如何将Java的思想运用到实际系统的详细设计之中。制作过程中，首先要将系统分为几大模块分别完成，具体编写中，我加深了对Java的理解，更加熟悉了用Java语言编写、调试程序，并将算法转化为程序实现。

此外，本次大作业我们组使用了Git来协同工作，这也是我第一次学习使用Git，我们在Gitee平台创建项目，在本地创建自己的分支，完成各部分后合并分支，极大的提高了开发效率。

当然，通过这次大作业使我也发现了许多需要改进的地方，一是自己不会的东西还有太多，今后需要长期积累，在以后的学习、工作中都应该不断地学习，将课本的理论知识与实践相结合，不断地提高自己的能力；二是要学习更好的团队合作，提高团队开发效率。

洪鑫城：

本次Java大作业的题目是Java版幻灯片的制作与播放软件，实话说一开始看到这个题目还真不知道如何下手，在查阅了一些资料后，我们决定先从画板做起。在制作画板的过程中，我们逐渐对整个工程有了具体的构想，后续的编程就比较有思路了。尽管有些功能还没来得及做，这个软件也有着很高的完成度。

同时，这次大作业我们第一次使用了Git来进行多人编程，这让我感受到了这种编程方式的方便，也让我学会了使用方法，相信这项技能在我未来的学习和工作中能带来极大的便利。

当然，这次大作业也有些不足之处。由于时间问题，撤销与重做功能没有做完。希望有机会能继续完善这个功能。