**基于面向对象方法的图形绘制演示系统设计与实现**

**组长： 姓名 曾猛徕 学号170703075 班级17计本二班**

**组员： 姓名 吴彤 学号170703068 班级17计本二班**

**姓名 熊澄丽 学号170703070 班级17计本二班**

**时间：2020．4．6-4.20**

## 1．引言

在学习软件工程课程中，为了更好的了解及运用所学知识以及锻炼同学们的团队合作能力，已经做过基于C语言的画简单图形的小系统，以及本系统的第一个版本，而本项目是在之前系统概念的基础上，使用面向对象语言Java实现的PLUS版本，相比于之前的版本，本项目更注重于用户交互能力，用途更广，功能更强大，修正了一些之前版本存在的问题，增加了登陆功能。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **文档名称** | **基于面向对象方法的图形绘制演示系统设计与实现** | | | | | |
| **文档编号** | 02 | | | | | |
| **版本号** | 02 | | | | | |
| **总页数** | 24页 | | | | | |
| **创建者** | 曾猛徕 | | **创建日期** | | | 2020.3.15 |
| **审核人** | 曾猛徕、熊澄丽、吴彤 | | **审核日期** | | | 2020.3.22 |
| **批准人** |  | | **批准日期** | | |  |
|  | |  |
| **版本修改记录** | | | | | | |
| **序号** | **日期** | **版本** | | **修改人** | **修改** | |
| 01 | 2020.4.18 | 3.0 | | 曾猛徕 | 架构调整 | |
| 02 | 2020.4.19 | 3.1 | | 吴彤 | 添加各种3d图形模型 | |
| 03 | 2020.4.19 | 3.2 | | 熊澄丽 | 将登陆界面改成Javafx,将3d图形放至画板 | |
| 04 | 2020.4.19 | 3.3 | | 曾猛徕 | 数据库设计、数据存储、用户密码加密 | |
| **项目名称** | **基于面向对象方法的图形绘制演示系统** | | | | | |
| **项目说明** | 该项目是基于面向对象语言JAVA实现的简单画板，为需要画简单图形的客户提供画任意颜色任意大小基础几何图形三角形、矩形、圆形以及3d图形球、长方体等功能，仅需选中功能，鼠标拖动即可绘制想要的图形，画画前需要使用用户名和密码登陆，可将绘制的图片保存到数据库。 | | | | | |
| **用户介绍** | 有简单画图需要的用户 | | | | | |
| **词汇表** | GUI:图形用户界面，Swing、awt:GUI工具包 | | | | | |
| **功能要求** | 绘制钢笔线条、直线、三角形、矩形、圆形等基础图形，可以画3d图形，可任意改变绘图粗细、颜色、背景，连接数据库登陆 | | | | | |
| **功能需求** | 用户界面要求，软硬件需求，性能要求，稳健性要求，安全要求，架构需求，其他要求 | | | | | |

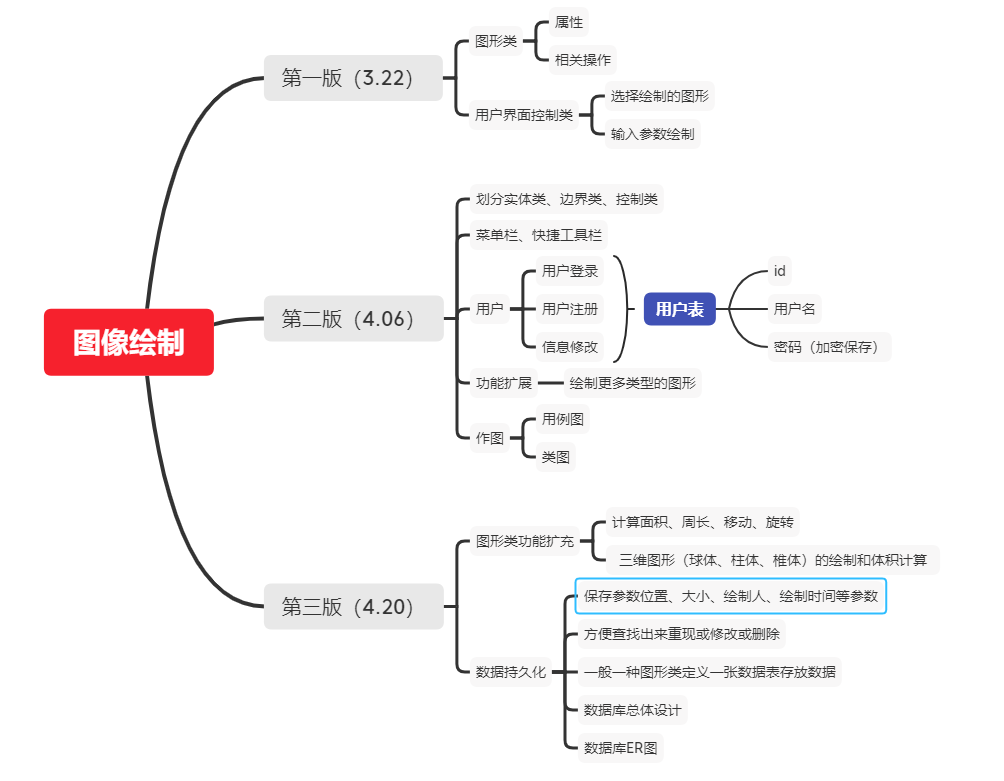
## 2．需求分析

第一阶段：需要用Java输入参数绘制正方形、圆形、三角形、直线等基础图形。

第二阶段：为了实用性，把第一阶段绘制集成到画板上，参数为图形界面工具栏参数和鼠标拖动点击产生的参数。

第三阶段：添加用户登录注册功能、将绘制的图片保存到数据库

第四阶段：添加画3D图形的功能，运用render技术将旋转中的3d图形截图至画板



## 3．系统总体设计

### 3．1 系统结构图

### 

后期加入了service层

分为entity层、mapper层（dao层）、service层、controller层、view层

Entity层为实体类，用于存储数据

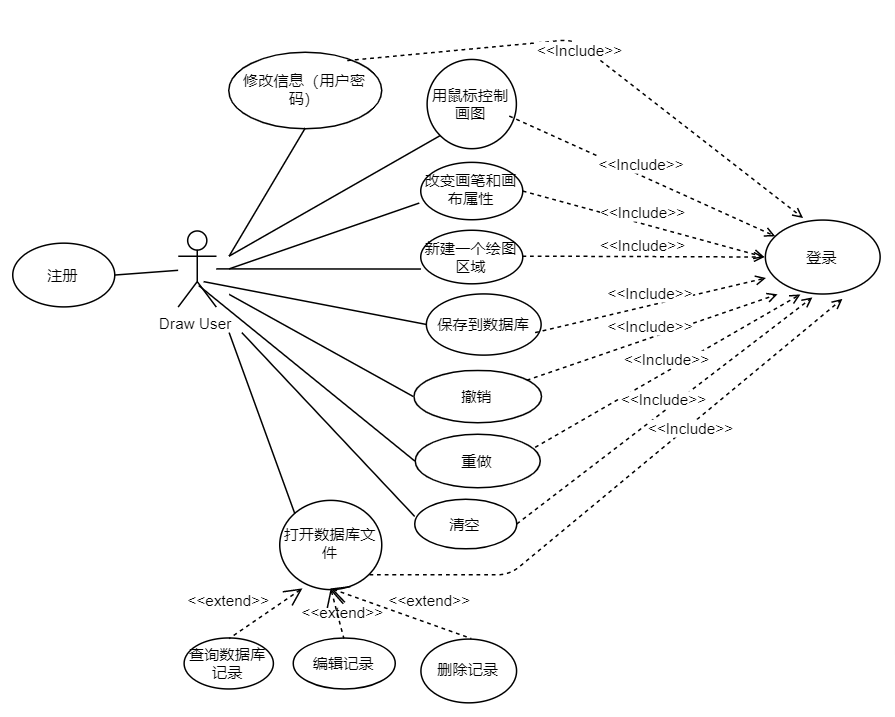
dao层用于数据库交互，所有数据库操作都放在dao层

service层做业务逻辑处理，例如将controller层传来的用户密码跟dao层数据库获取的密码对比，判断用户登录是否成功

controller层控制及联系service层、view层

view层创建窗体图形界面，与用户交互

用例图

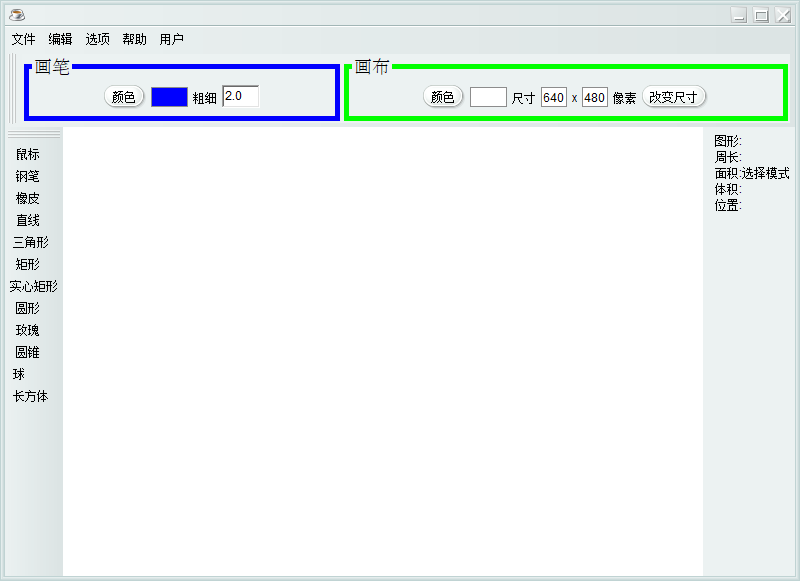


### 3．2 系统界面设计

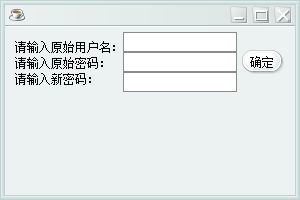
**登录、注册界面：**输入用户名、密码，选择登录/注册

### 

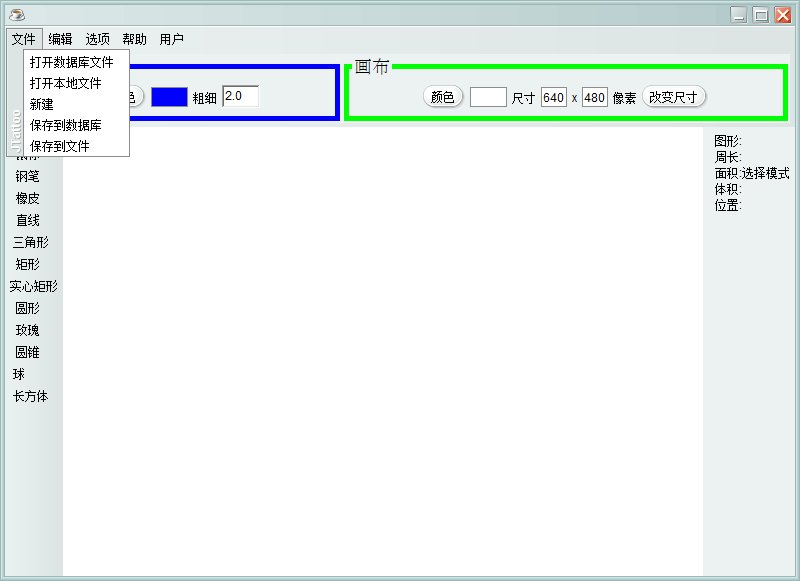
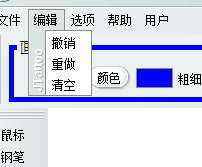
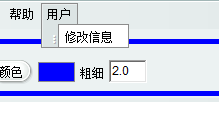
**绘图主界面：**分为菜单栏、顶部工具栏、左侧工具栏、绘图区域。（使用JTattoo包美化后）



**修改信息界面：**输入用户名、旧密码、新密码,即可修改用户密码。



**菜单栏：**有文件、编辑、选项、帮助、用户

“文件”又分为“打开数据库文件”、“打开本地文件”、“新建”、“保存到数据库”和“保存到文件”；

“编辑”设“撤销”、“重做”及“清空功能”；

“用户”设“修改用户信息”功能；

另外两个部分功能待开发。

**顶部工具栏**：设“画笔”设置和“画布”设置；



其中，“画笔”主要用于设置画笔的颜色及粗细；

“画布”主要用于设置画布的背景颜色及大小。

**左侧工具栏**：设“鼠标”、“钢笔”、“橡皮擦”、“直线”、“三角形”、“矩形”、“实心矩形”，“圆形”，“玫瑰”几个部分，点击相应按钮即可在绘图区域绘制相应图形；

其中，点击“鼠标”按钮可切换为鼠标操作模式，用于改变设置，点击“钢笔”按钮可切换为“随心画”模式，点击“橡皮擦”使用橡皮擦功能擦掉你不喜欢的地方，点击“直线”切换为画直线模式，点击“三角形”可画任意大小三角形，点击“矩形”、“实心矩形”、“圆形”按钮同理。

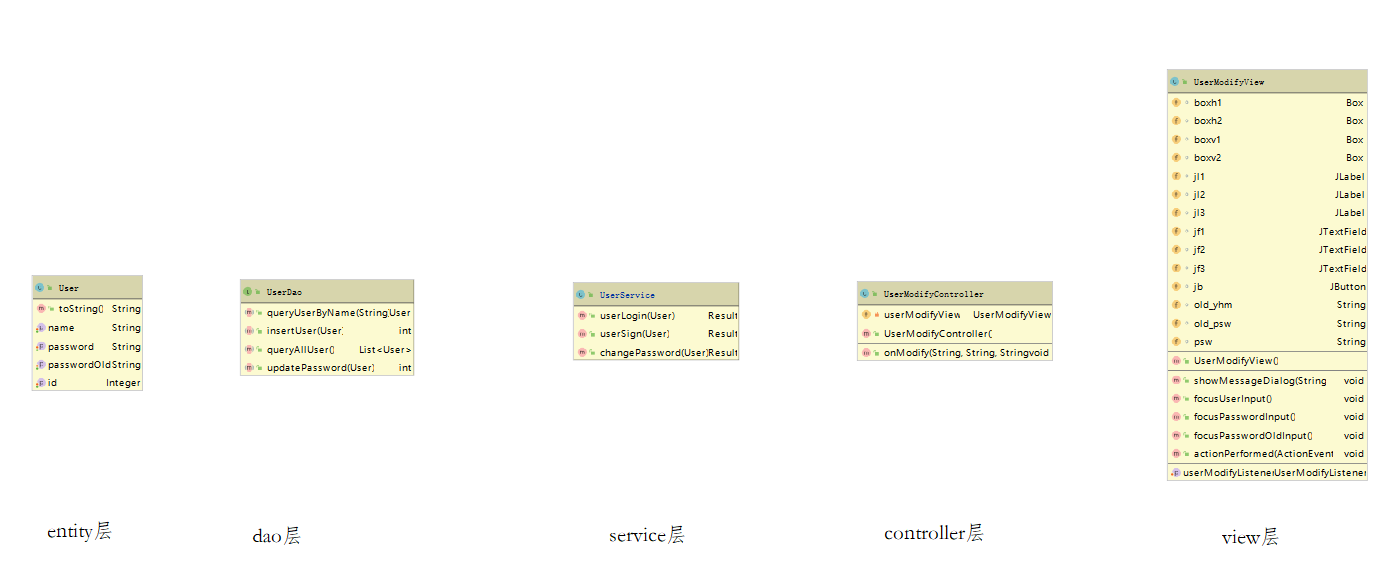
**右侧信息栏：**显示图形的面积、体积、位置等信息

**数据库图片管理主界面：**可查询、编辑、删除数据库中保存的图片，点击编辑即可将图片放到绘图界面中继续绘制。

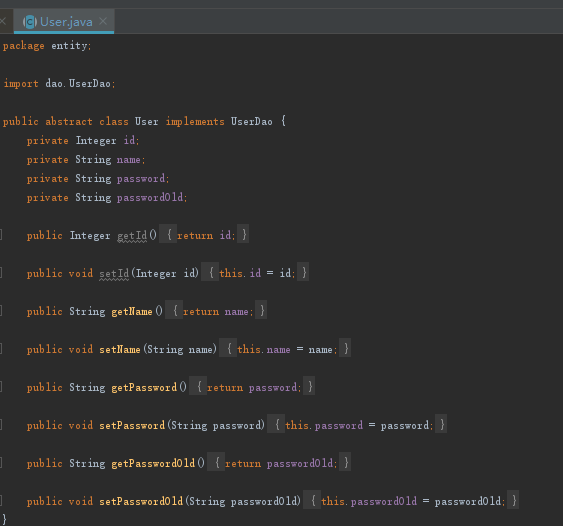


### 3．3 类的设计

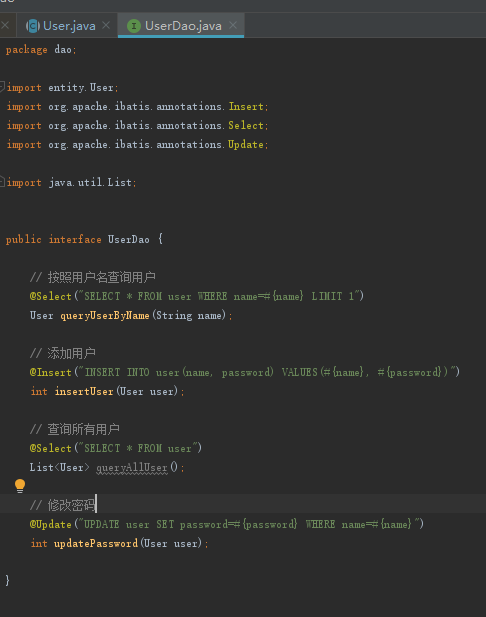
**以修改密码举例，说明每层的结构：**



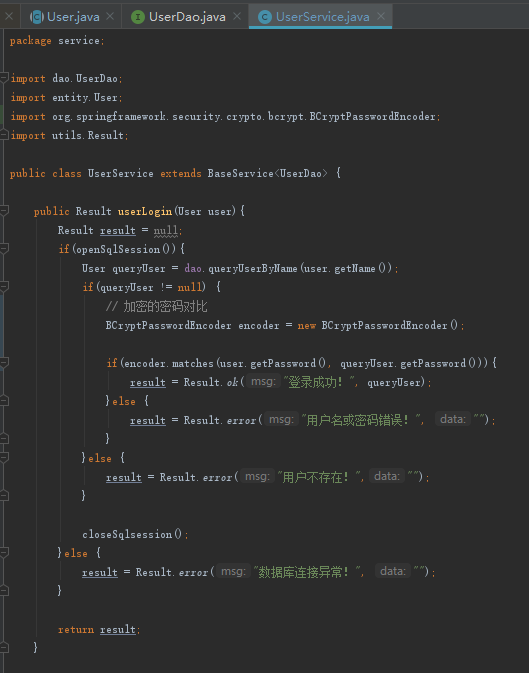
Entity



Dao层



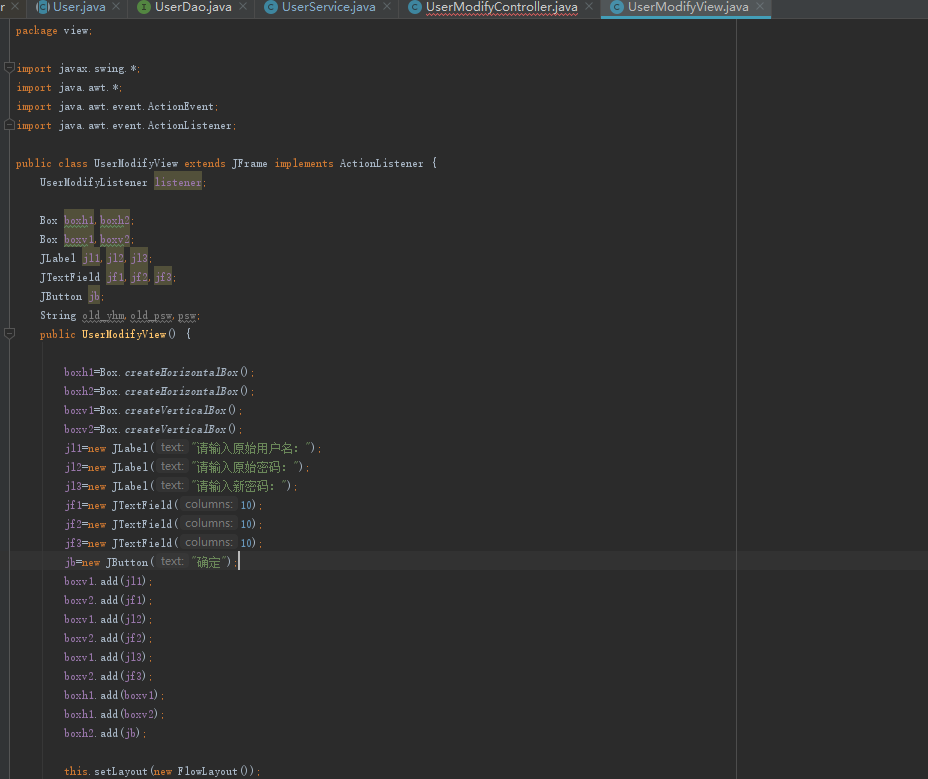
Service层



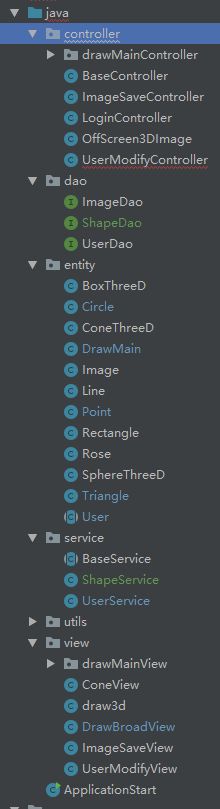
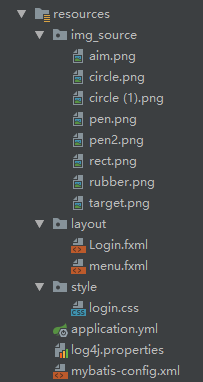
Controller层



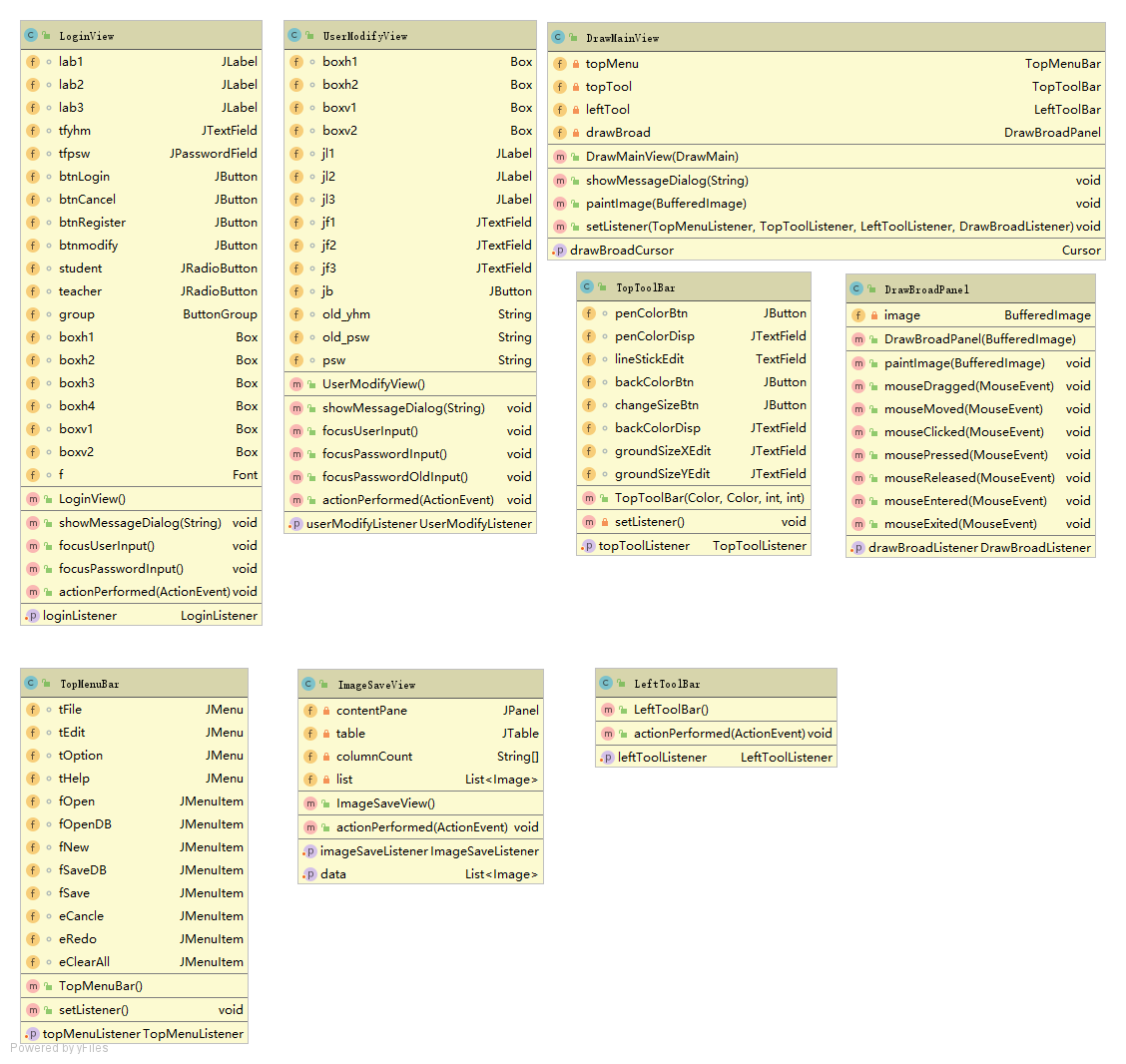
View层



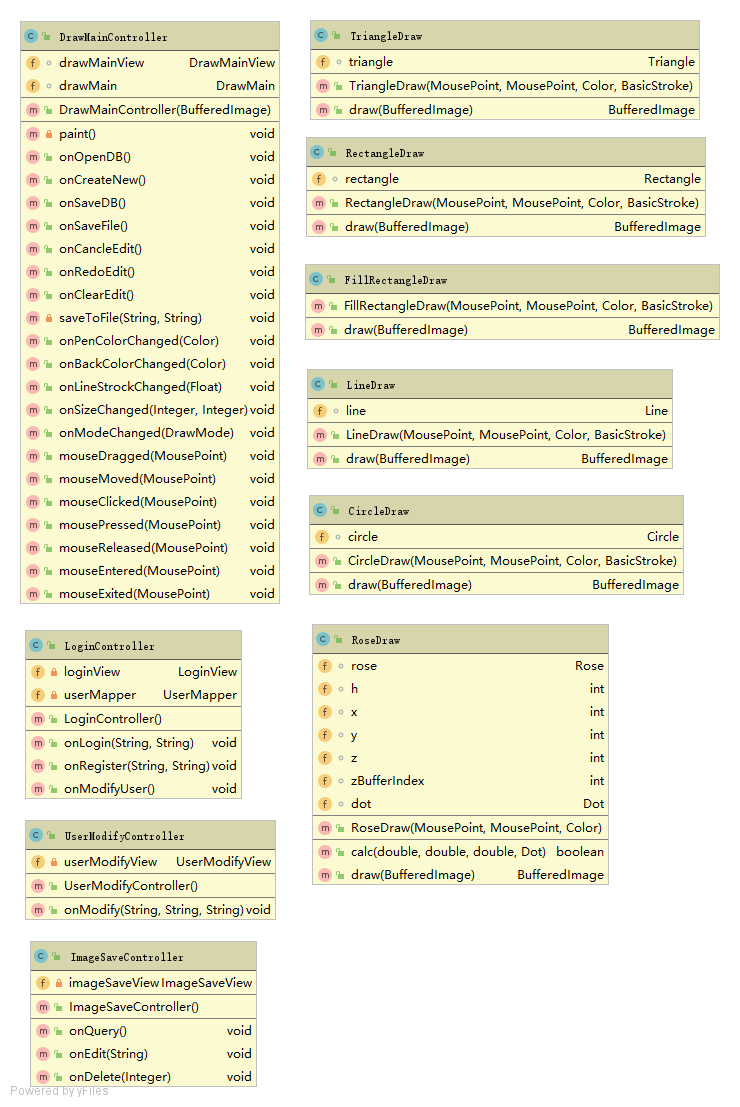
整个目录结构如下：

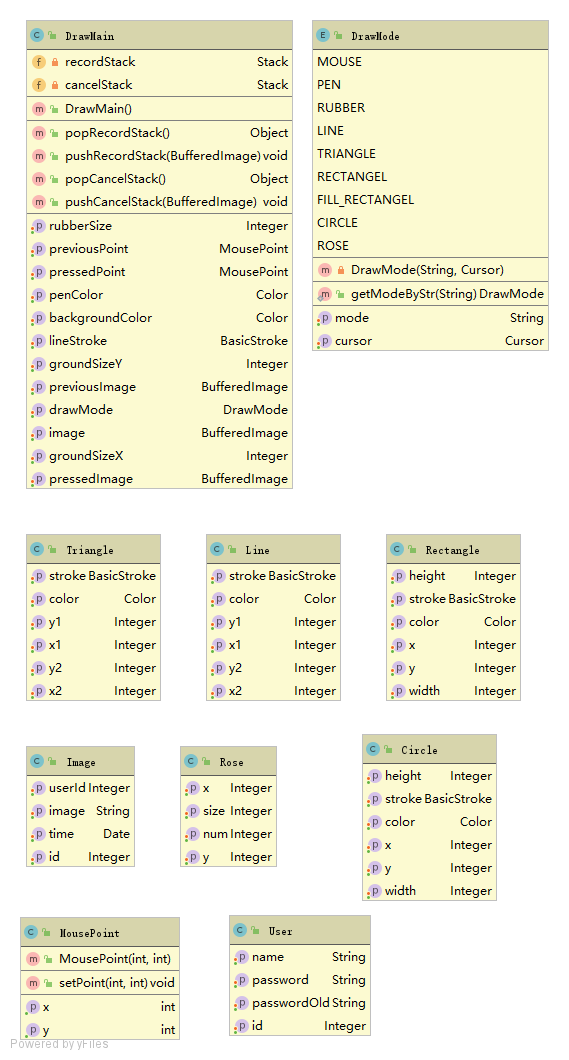
view层：



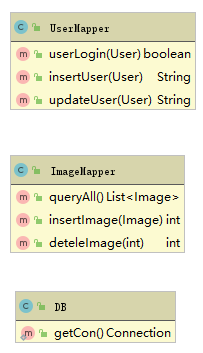
Model层



Entity层



Mapper层

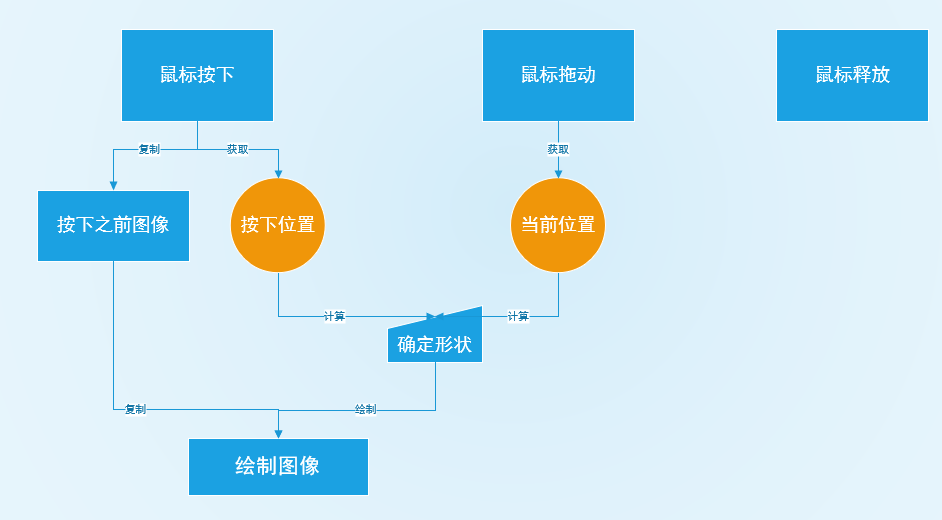


## 4.系统的详细设计与关键代

**4.1绘图功能实现**

1. **绘制直线、三角形、圆形、矩形的原理及过程**

原理：利用鼠标拖动进行绘制 ，鼠标拖动会触发的事件有：鼠标按下事件->鼠标拖动事件->鼠标释放事件。为了实现图形大小和高矮随着鼠标拖动距离和位置变化而变化的绘图预览效果，在鼠标按下时，记录鼠标按下位置，将图片保留一份备份，通过按下点和当前点计算出图形的位置和高宽参数，在之前备份图片的基础上绘制。



1. **绘制直线、三角形、圆形、矩形的关键部分代码**

**鼠标按下：**

先保存鼠标按下之前的图像：

@Override

**public void** mousePressed(MousePoint p) {

*// 记录鼠标按下时的点和图像*

**drawMain**.setPreviousPoint(p);

**drawMain**.setPressedPoint(p);

**drawMain**.setPressedImage(**drawMain**.getImage());

*// 按下就一定会产生图像绘制，此时先保存记录用以撤销*

**if**(**drawMain**.getDrawMode() != DrawMode.***MOUSE***){

**drawMain**.pushRecordStack(ImageUtil.*imageCopy*(**drawMain**.getImage()));

}

}

鼠标拖动的同时产生绘图参数，将参数传入各个图形绘制类，进行绘制

*// 鼠标拖动*

@Override

**public void** mouseDragged(MousePoint p) {

**switch**(**drawMain**.getDrawMode()) {

**case *PEN*** :{ *// 钢笔绘制*

**if**(**drawMain**.getPreviousPoint() != **null**) {

Graphics g = **drawMain**.getImage().getGraphics();

g.setColor(**drawMain**.getPenColor());

g.drawLine(**drawMain**.getPreviousPoint().getX(), **drawMain**.getPreviousPoint().getY(), p.getX(), p.getY());

}

}**break**;

**case *RUBBER*** :{ *// 橡皮擦*

Graphics g = **drawMain**.getImage().getGraphics();

g.setColor(**drawMain**.getBackgroundColor());

Integer radius = **drawMain**.getRubberSize(); *// 橡皮擦半径*

g.fillOval(p.getX() - (radius/2),p.getY() - (radius/2), radius, radius);

}**break**;

**case *FILL\_RECTANGEL***: { *// 绘制实心矩形*

FillRectangleDraw fillRectangleDraw = **new** FillRectangleDraw(**drawMain**.getPressedPoint(), p, **drawMain**.getPenColor(), **drawMain**.getLineStroke());

BufferedImage image = fillRectangleDraw.draw(**drawMain**.getPressedImage());

**drawMain**.setImage(image); *// 更新绘制图像*

}**break**;

**case *RECTANGEL***:{ *// 绘制矩形*

*// 传入参数创建矩形控制类*

RectangleDraw rectangleController = **new** RectangleDraw(**drawMain**.getPressedPoint(), p, **drawMain**.getPenColor(), **drawMain**.getLineStroke());

BufferedImage image = rectangleController.draw(**drawMain**.getPressedImage()); *// 绘制矩形到图像上*

**drawMain**.setImage(image); *// 更新绘制图像*

}**break**;

**case *CIRCLE***:{ *// 绘制圆形*

CircleDraw circleController=**new** CircleDraw(**drawMain**.getPressedPoint(), p, **drawMain**.getPenColor(), **drawMain**.getLineStroke());

BufferedImage image=circleController.draw(**drawMain**.getPressedImage());

**drawMain**.setImage(image); *// 更新绘制图像*

}**break**;

**case *LINE*** :{ *// 画直线*

LineDraw lineDraw =**new** LineDraw(**drawMain**.getPressedPoint(), p, **drawMain**.getPenColor(), **drawMain**.getLineStroke());

BufferedImage image=lineDraw.draw(**drawMain**.getPressedImage());

**drawMain**.setImage(image); *// 更新绘制图像*

}**break**;

**case *TRIANGLE*** :{ *// 画三角形*

TriangleDraw triangleController= **new** TriangleDraw(**drawMain**.getPressedPoint(), p, **drawMain**.getPenColor(), **drawMain**.getLineStroke());

BufferedImage image = triangleController.draw(**drawMain**.getPressedImage()); *// 绘制三角形到图像上*

**drawMain**.setImage(image); *// 更新绘制图像*

}**break**;

**case *ROSE***:{ *// 画玫瑰*

RoseDraw roseController=**new** RoseDraw(**drawMain**.getPressedPoint(), p, **drawMain**.getPenColor());

BufferedImage image=roseController.draw(**drawMain**.getPressedImage());

}**break**;

**default** : **break**;

}

*// 把更新后的图像绘制到界面上*

paint();

*// 记录当前坐标*

**drawMain**.setPreviousPoint(p);

}

图形类处理转化成相应图形参数

**public** CircleDraw(MousePoint p1, MousePoint p2, Color penColor, BasicStroke lineStroke){

**circle** = **new** Circle();

*/\*\**

*\* 设置起始位置和宽高度*

*\* 小的作为起始位置，大的减去小的即为宽高度*

*\*\*/*

**if**(p1.getX() < p2.getX() ){

**circle**.setX(p1.getX());

**circle**.setWidth(p2.getX() - p1.getX());

}**else**{

**circle**.setX(p2.getX());

**circle**.setWidth(p1.getX() - p2.getX());

}

**if**(p1.getY() < p2.getY()){

**circle**.setY(p1.getY());

**circle**.setHeight(p2.getY() - p1.getY());

}**else**{

**circle**.setY(p2.getY());

**circle**.setHeight(p1.getY() - p2.getY());

}

**circle**.setColor(penColor);

**circle**.setStroke(lineStroke);

}

绘制到图像对象上

**public** BufferedImage draw(BufferedImage image){

*// 复制一份用于绘制*

BufferedImage drawImage = ImageUtil.*imageCopy*(image);

Graphics2D g2 = (Graphics2D)drawImage.getGraphics();

g2.setColor(**circle**.getColor());

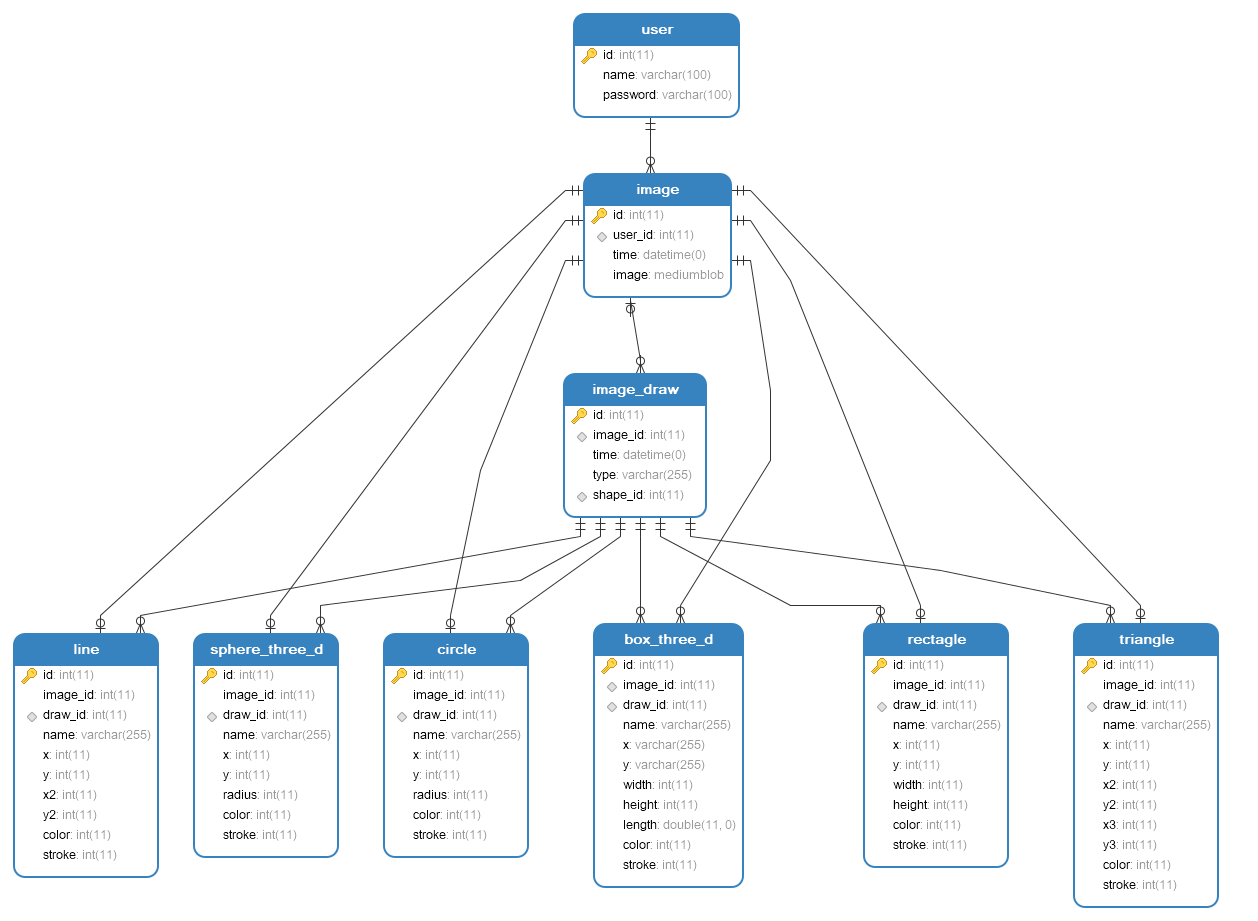
g2.setStroke(**circle**.getStroke());

g2.drawOval(**circle**.getX(), **circle**.getY(), **circle** .getWidth(), **circle**.getHeight());

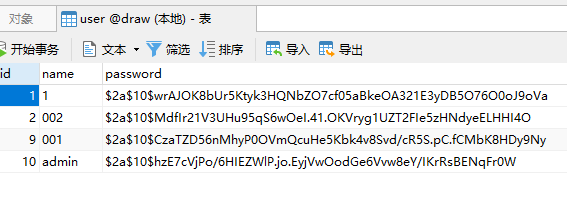
**return** drawImage;

}

**4.2数据库设计**



用户密码加密



**4.3绘制3D图形**

**1.主要步骤**

1. 创建一个OffScreen3DImage类在该类中定义好绘制3D图形需要的Canvas3D、SimpleUniverse、BranchGroup等，并创建BufferedImage类对象以便将3D图形导出。
2. 在创建BranchGroup对象时设置Color3f颜色和子BranchGroup，将3d图形模型建好后放至子BranchGroup，再将子BranchGroup和Color3f对象放入主BranchGroup，以此设置图像类型和颜色。
3. 将Canvas3D放入JFram中自动旋转显示，点击Render按钮可以使用截图至ImageLable预览，然后在画板上画出。

**2.关键代码**

创建导出的image

**void** createOffScreenImage() {

**canvas3D** = getOffScreenCanvas();

**onscreenCanvas3D** = **new** Canvas3D(SimpleUniverse.*getPreferredConfiguration*());

SimpleUniverse u = **new** SimpleUniverse(**onscreenCanvas3D**);

u.getViewingPlatform().setNominalViewingTransform();

BranchGroup bg = createSceneGraph(getColor3f(),getGeometry3d());

u.addBranchGraph(bg);

u.getViewer().getView().addCanvas3D(**canvas3D**);

createBufferImage();

**canvas3D**.setOffScreenBuffer(**buffer**);

**canvas3D**.renderOffScreenBuffer();

**canvas3D**.waitForOffScreenRendering();

**imageLabel**.setIcon(**new** ImageIcon(**buffer**.getImage()));

setDrawImage(**buffer**.getImage());

System.***out***.println(**"Frame: "** + **canvas3D**.getView().getFrameNumber());

}

创建主BranchGroup

BranchGroup createSceneGraph(Color3f color3f,BranchGroup group) { //参数

*// Create the root of the branch graph*

BranchGroup objRoot = **new** BranchGroup();

*// rotate object has composited transformation matrix*

Transform3D rotate = **new** Transform3D();

Transform3D tempRotate = **new** Transform3D();

rotate.rotX(Math.***PI*** / 4.0d);

tempRotate.rotY(Math.***PI*** / 5.0d);

rotate.mul(tempRotate);

TransformGroup objRotate = **new** TransformGroup(rotate);

*// Create the transform group node and initialize it to the*

*// identity. Enable the TRANSFORM\_WRITE capability so that*

*// our behavior code can modify it at runtime. Add it to the*

*// root of the subgraph.*

TransformGroup objSpin = **new** TransformGroup();

objSpin.setCapability(TransformGroup.***ALLOW\_TRANSFORM\_WRITE***);

objRoot.addChild(objRotate);

objRotate.addChild(objSpin);

*// Create a new Behavior object that will perform the desired*

*// operation on the specified transform object and add it into*

*// the scene graph.*

Transform3D yAxis = **new** Transform3D();

Alpha rotationAlpha = **new** Alpha(-1, 4000);

RotationInterpolator rotator = **new** RotationInterpolator(rotationAlpha,

objSpin, yAxis, 0.0f, (**float**) Math.***PI*** \* 2.0f);

rotator.setSchedulingBounds(**new** BoundingSphere(**new** Point3d(), 1000));

objSpin.addChild(rotator);

*// a bounding sphere specifies a region a behavior is active*

*// create a sphere centered at the origin with radius of 1*

*//BoundingSphere bounds = new BoundingSphere();*

BoundingSphere bounds =

**new** BoundingSphere(**new** Point3d(0.0,0.0,0.0), 1000.0);

*//新建光源*

Color3f light1Color = color3f;

Vector3f light1Direction = **new** Vector3f(-1.0f, -1.0f, -1.0f);

DirectionalLight light1

= **new** DirectionalLight(light1Color, light1Direction);

light1.setInfluencingBounds(bounds);

objRoot.addChild(light1);

*//设置背景颜色*

Color backgroundColor=**new** Color(255, 240, 227);

Background bg = **new** Background(backgroundColor.getRed()/255f, backgroundColor.getGreen()/255f, backgroundColor.getBlue()/255f);

bg.setApplicationBounds(bounds);

objRoot.addChild(bg);

*// Create a simple shape leaf node, add it to the scene graph.*

*// ColorCube is a Convenience Utility class*

objSpin.addChild(group);

*// objSpin.addChild(new Sphere());*

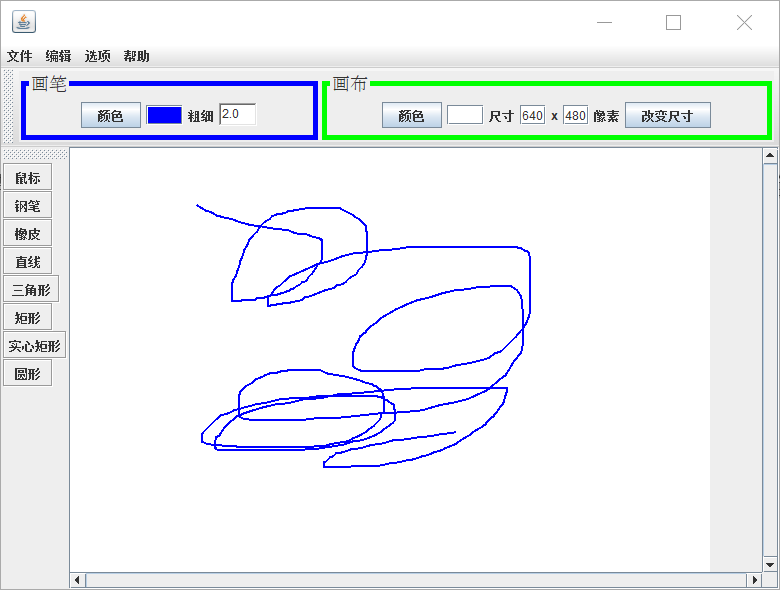
objRoot.compile();

**return** objRoot;

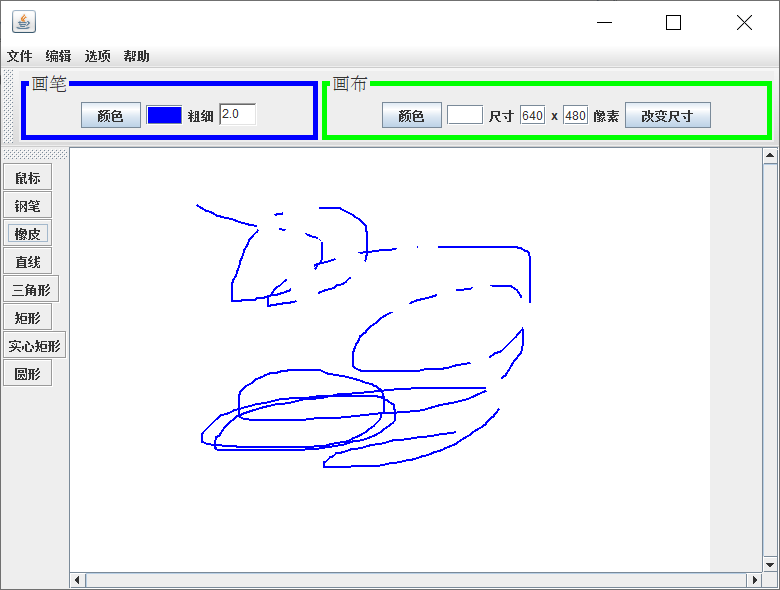
}

## 5.系统测试

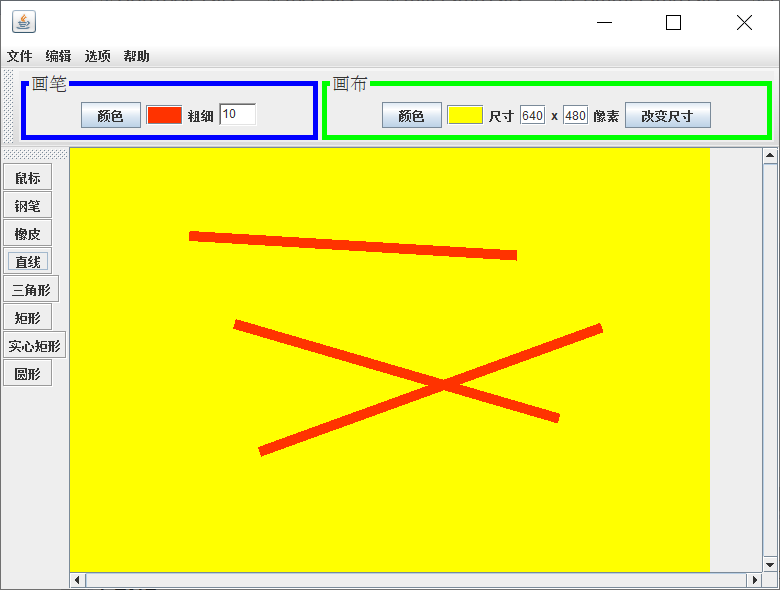
1.“钢笔”功能，颜色：蓝色 粗细：2.0 画布颜色：白色



2.“橡皮”功能



3.“直线”功能，颜色：红色 粗细：10 画布颜色：黄色



4.“撤销”功能

4.1“撤销”1次

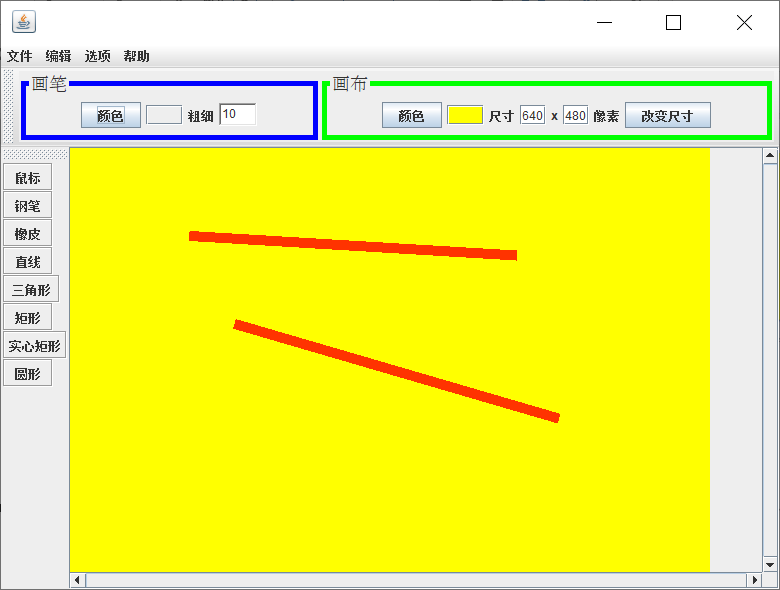


4.2撤销2次

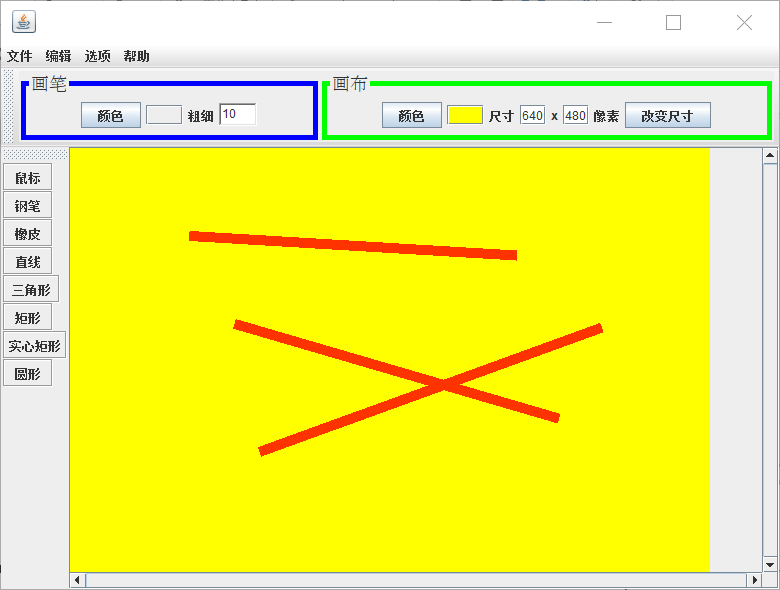


5.“重做”功能

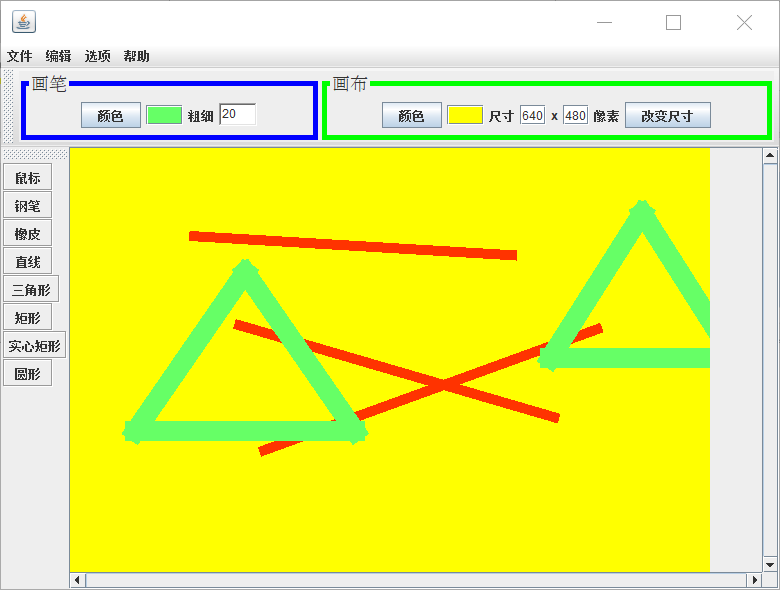
5.1“重做”一次



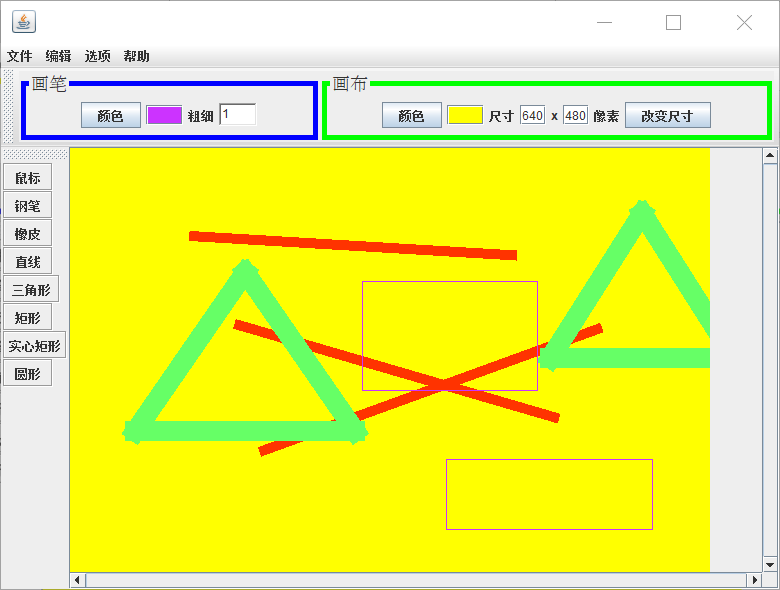
5.2“重做”2次



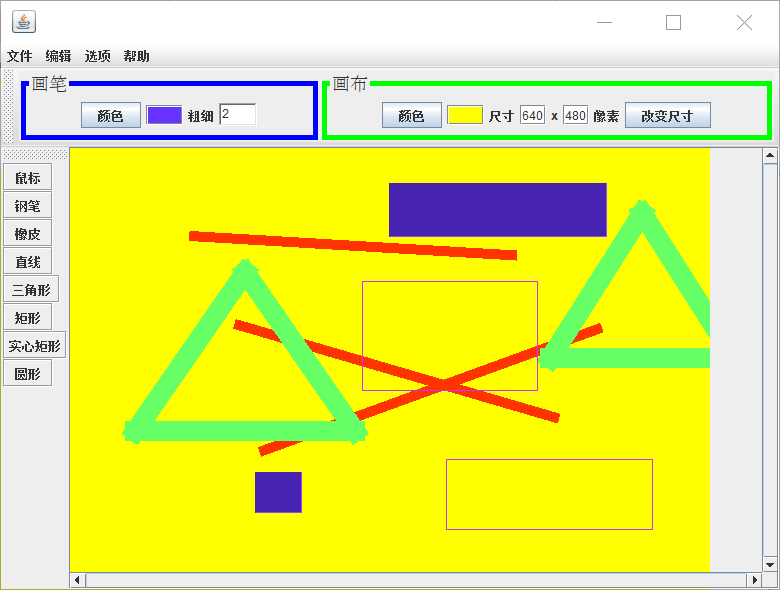
6.“三角形” 颜色：绿色 粗细：20



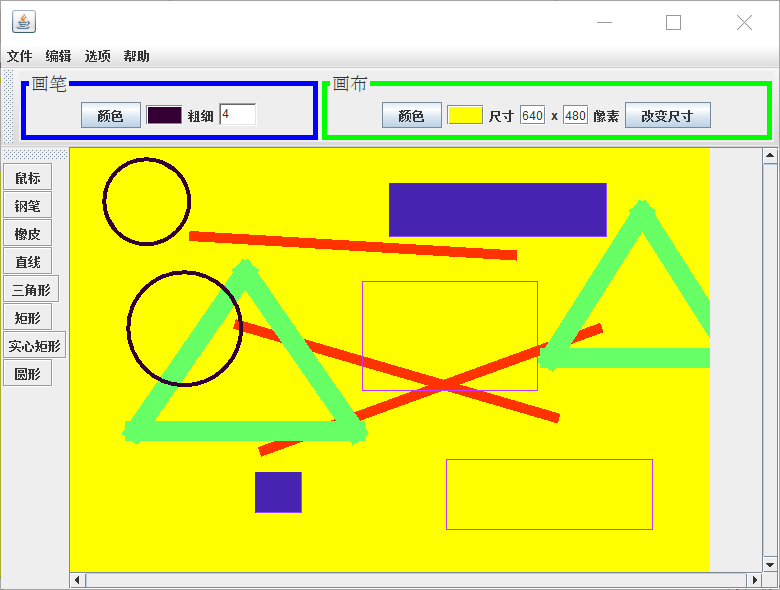
7.“矩形” 颜色：紫色 粗细：1



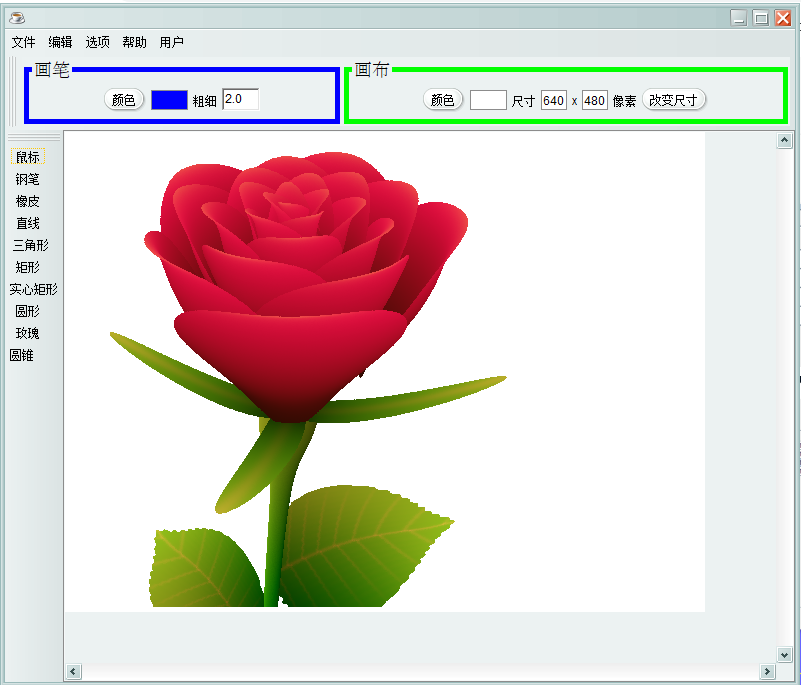
8.“实心矩形” 颜色：深蓝色 粗细：2



9．“圆形” 颜色：黑色 粗细：4

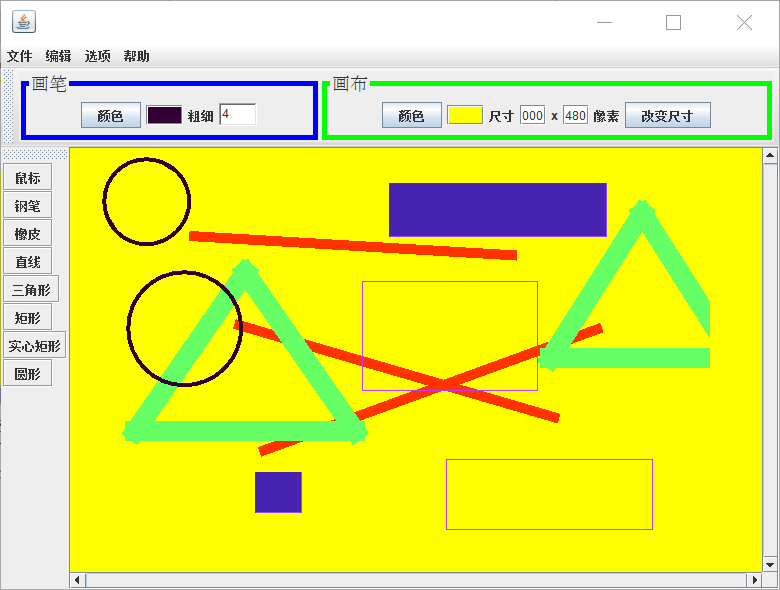


9.画玫瑰花



11.改变画布尺寸

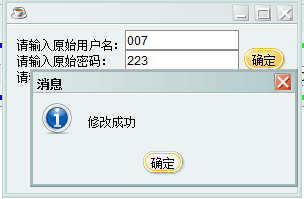
由640\*480变为1000\*480

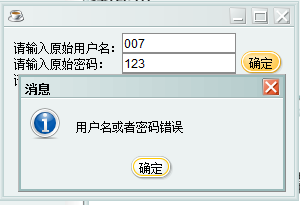


11.用户登录、注册、修改密码

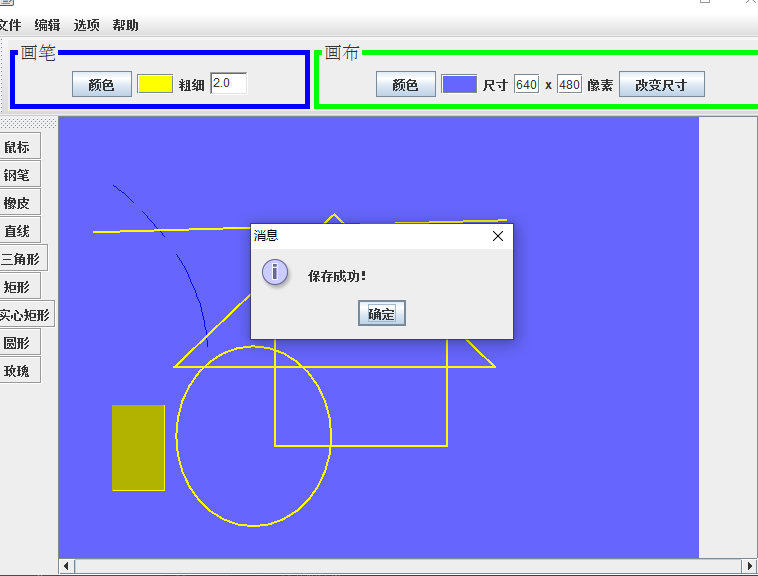




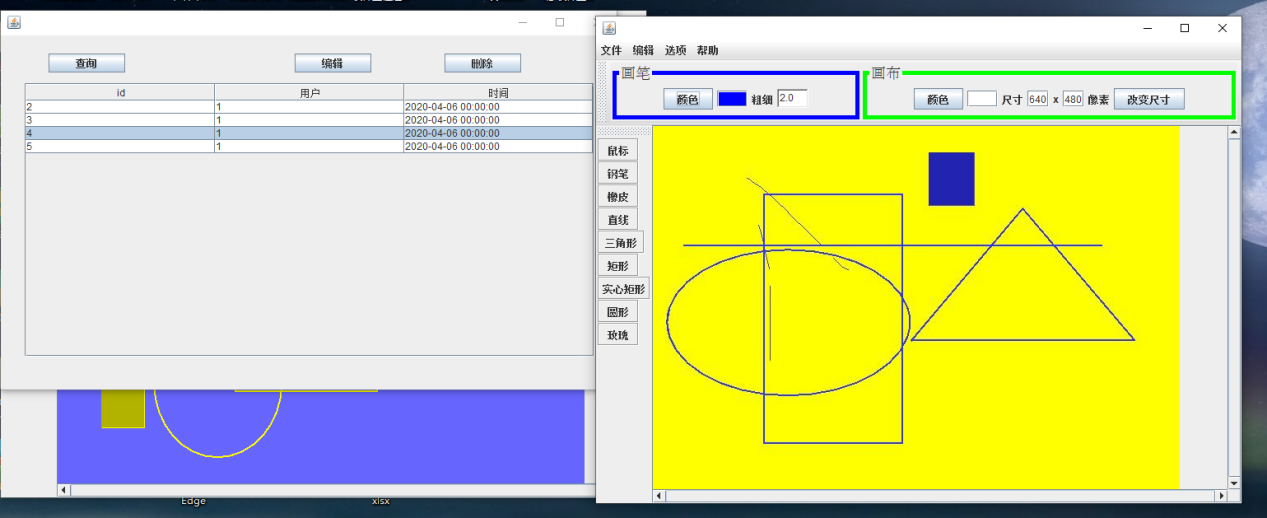




12.保存到数据库，从数据库中加载图片





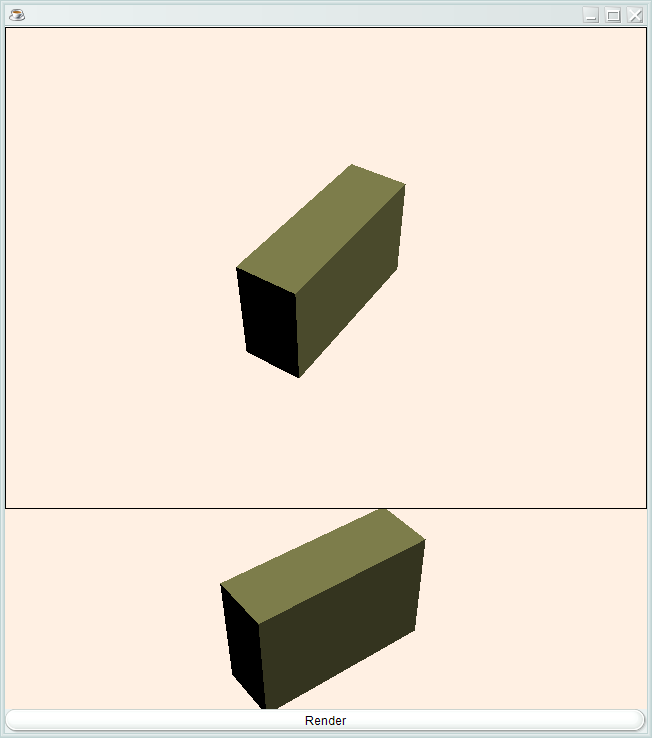


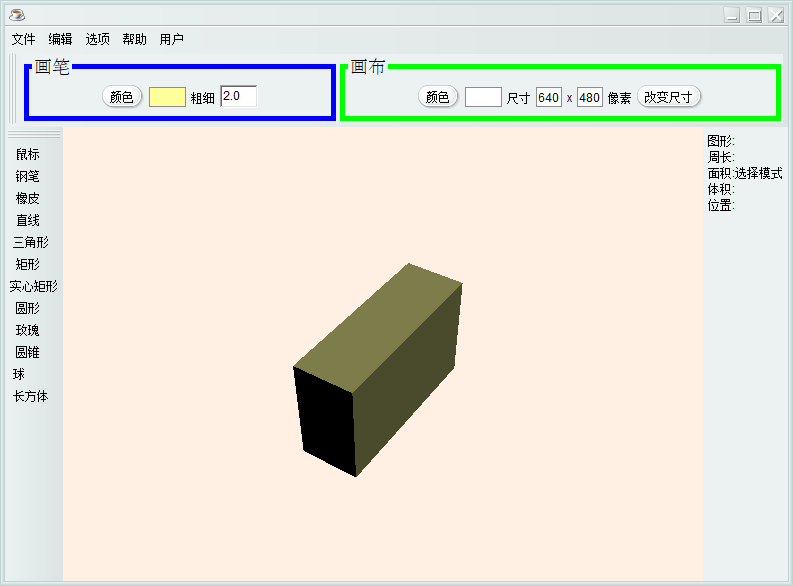
13.绘制3D图形

点击Render按钮截图至上面的JLable预览

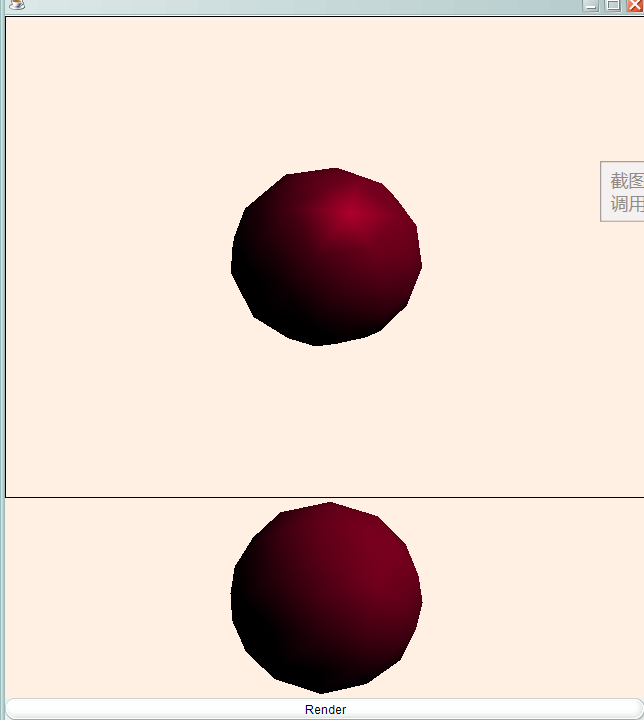
再画到画板

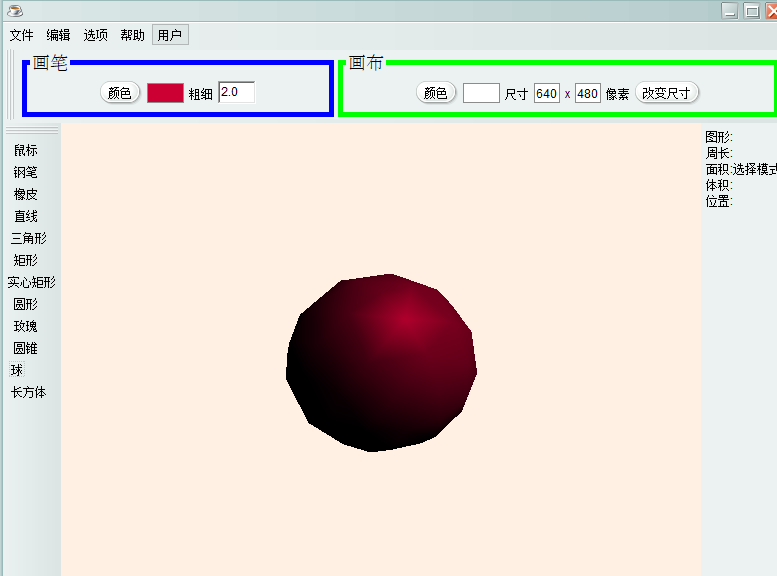
13.1长方体



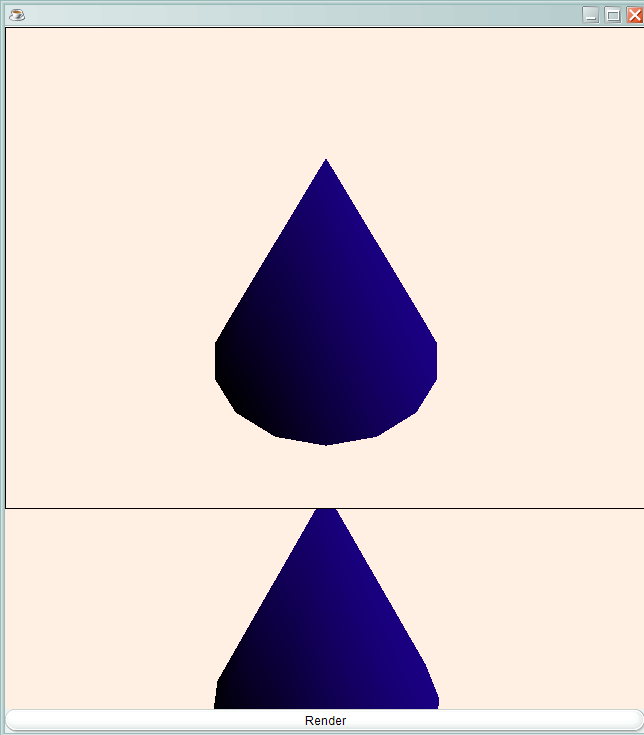


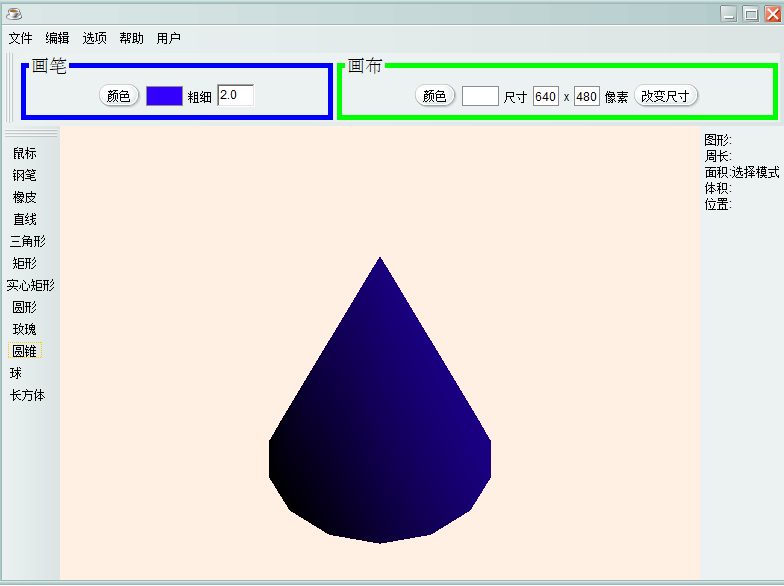
13.2球





13.3圆锥





## 6.系统的发布与维护

gitHub开源库： <https://github.com/Zeng1168/draw>

使用技术：java、maven、mybatis、mysql、java3d、美化包jtattoo

开发环境：IntelliJ IDEA 2019.以上 64位、 mysql5.7以上、JDK1.8 64位

## 7.小结

经过这次项目开发，对Java的图形用户开发界面有了更深入的学习认识。以前不熟练的知识点通过这次开发训练温故而知新，学以致用才能学得好。

越了解JAVA越能体会到编程的乐趣，也明白了学习永无止境，自己本身有很大的不足，也有很多知识仅限于课堂上的了解而不能自如的运用，比如这次的鼠标监听，还有很多想法想的很好要实现很难（更可怕的是没有想法），通过在网络上查找资料以及小组成员间的互助，我想我们都成长了很多。

## 参考文献：

1.java图形用户界面设计

2.Java图形绘制

3.面向对象方法

4.MVC编程

5.maven快速入门

6.mybatis注解开发

## 附录：