C++大作业

学号：SA20218099 姓名：罗浩楠 先研院电子信息

设计

开发环境

- IDE：Qt Creator
- 运行环境：window10 专业版
- 配置要求：内存 4g

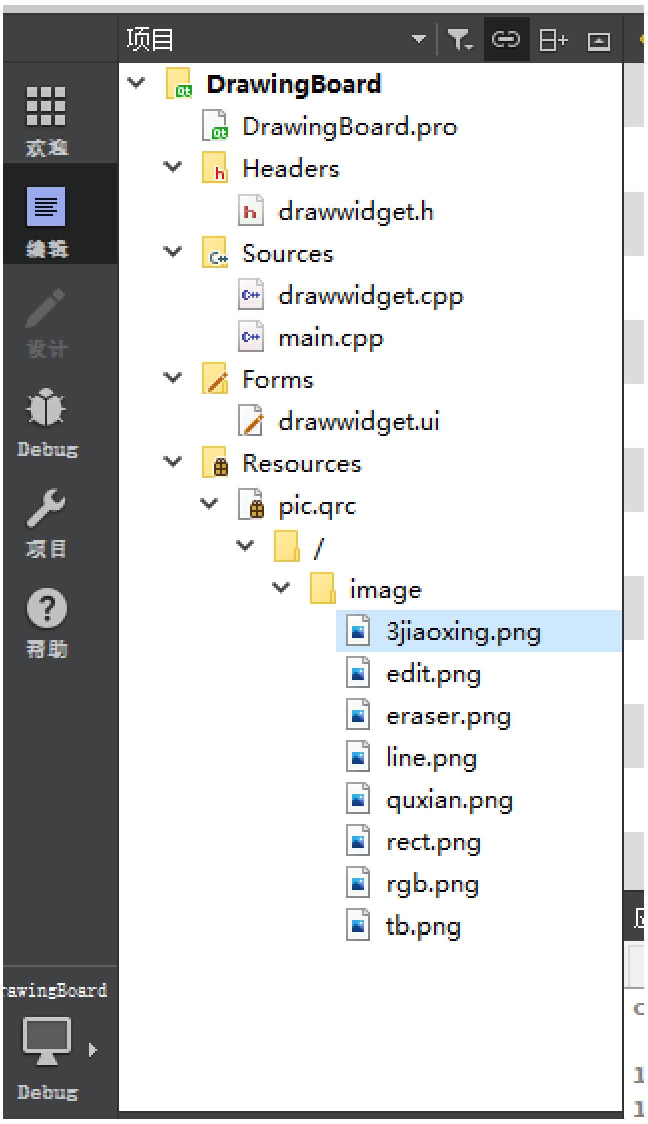
  显卡 无要求

  CPU 无要求

目标

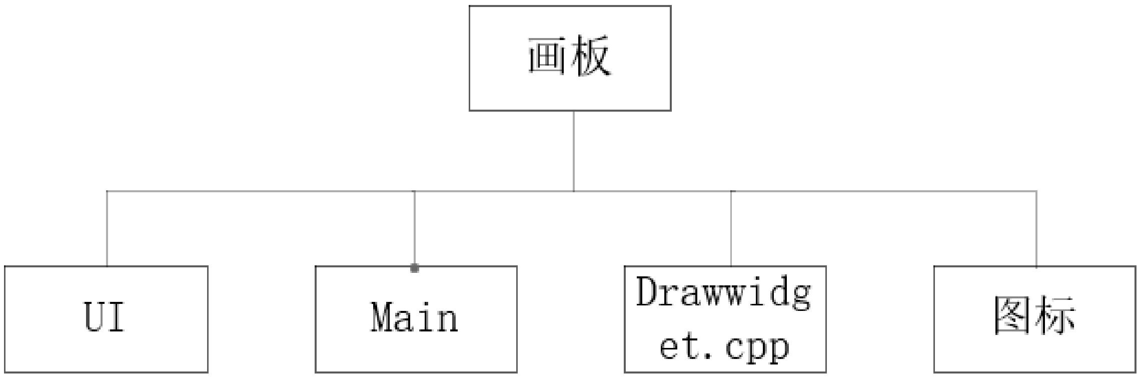
- 基于C++语言，设计/实现一个支持简单交互绘图小程序
- 要求
  - 提交本实验的设计报告以及实现的源代码
- 目标程序基本功能需求
  - 实现附录示意图的，含标题条、菜单、工具条和绘图工作区的窗口界面
  - 基本图形：直线、矩形、多边形、椭圆、文本
  - 支持对已工作区中绘制的图形，选择、拖动，修改大小，删除操作
  - 支持将当前工作区所有图形序列化存储到磁盘文件，以及将磁盘中的图形文件重载入工作区--编辑处理

结构化模块层次图



```
DrawingBoard
    DrawingBoard.pro
    Headers
        drawwidget.h
    Sources
        drawwidget.cpp
        main.cpp
    Forms
        drawwidget.ui
    Resources
        pic.qrc
            /
                image
                    3jiaoxing.png
                    edit.png
                    eraser.png
                    line.png
                    quxian.png
                    rect.png
                    rgb.png
                    tb.png
```

层次图：



注：

- UI：画板样式的设计，通过Qt creator设计
- Main：程序主函数
- Drawidget.cpp：内部逻辑
- 图标：绘制直线、线段、三角等的图标样式图片

UML类图

DrawWidget
-penColor：QColor -ui：Ui::DrawWidget -image：QImage -tempImage:QImage -setting_color:Qrgb -drawing:bool -shape:int -point:QPoint -from:QPoint -to:QPoint -change:QPoint -pointPolygon[3]:QPoint -width:int -height:int -lineEdit:QLineEdit
#paintEvent(QPaintEvent *event) #mousePressEvent(QMouseEvent *event) #mouseReleaseEvent(QMouseEvent *event) #mouseMoveEvent(QMouseEvent *event) -on_radioButton_clicked() -on_radioButton_2_clicked() -on_pushButton_clicked() -on_shape_clicked() -on_radioButton_3_clicked() -on_pushButton_2_clicked() -on_radioButton_4_clicked() -on_pushButton_3_clicked() -on_radioButton_5_clicked()

主要模块功能接口描述

drawwidget的所有on_radioButton_clicked()的功能为按照用户选择的不同的radioButton选择不同的形状，对应的对shape进行操作，比如：

```
void DrawWidget::on_radioButton_clicked()
{
    shape = 1;
}
```

用户点击了radioButton1就将shape置为1。

mouseReleaseEvent(QMouseEvent *event)的程序的功能是在鼠标释放时程序的操作。

paint(QImage &theImage)是绘画函数，会依据shape的数值绘制不同的图像，如下所示：

```
switch (shape) {
    case 0:thePainter.drawLine(change,point);change = point;break;
    case 1:thePainter.drawLine(from,point);break;
//thePainter.drawLine(from,to);break;
    case 2:thePainter.drawRect(from.x(),from.y(),width,heigh);break;
    case 3:thePainter.eraseRect(point.x(),point.y(),ui->penWidth->value()+5,ui-
>penWidth->value()+5);break;
    case 4:thePainter.drawPolygon(pointPolygon,3);break;
    case 5:
                lineEdit.move(point.x(),point.y());
                lineEdit.setVisible(true);
                thePainter.drawText(change,lineEdit.text());
                lineEdit.clear();
                if(lineEdit.text()!="")
                {
                    lineEdit.setVisible(false);
                }
            break;


    default:break;
    }
```

代码实现

drawwidget.h

```
#ifndef DRAWWIDGET_H
#define DRAWWIDGET_H

#include <QWidget>
#include <QPainter>
#include <QImage>
#include <QPoint>
#include <QLineEdit>

namespace Ui {
class DrawWidget;
}

class DrawWidget : public QWidget
{
    Q_OBJECT

public:
    explicit DrawWidget(QWidget *parent = nullptr);
    ~DrawWidget();
    void paint(QImage &theImage);
    QColor penColor;
private:
    Ui::DrawWidget *ui;
    QImage image;   //
    QImage tempImage;
    QRgb setting_color;//背景色

    bool drawing;
    int shape;
    QPoint point;
    QPoint from;
    QPoint to;
    QPoint change;
    QPoint pointPolygon[3];
    int width,heigh;
    QLineEdit lineEdit;



protected:
    void paintEvent(QPaintEvent *event);
    void mousePressEvent(QMouseEvent *event);
    void mouseReleaseEvent(QMouseEvent *event);
    void mouseMoveEvent(QMouseEvent *event);
private slots:
    void on_radioButton_clicked();
    void on_radioButton_2_clicked();
    void on_pushButton_clicked();
    void on_shape_clicked();
    void on_radioButton_3_clicked();
    void on_pushButton_2_clicked();
    void on_radioButton_4_clicked();
    void on_pushButton_3_clicked();
    void on_radioButton_5_clicked();
};

#endif // DRAWWIDGET_H
```

drwawidget.cpp

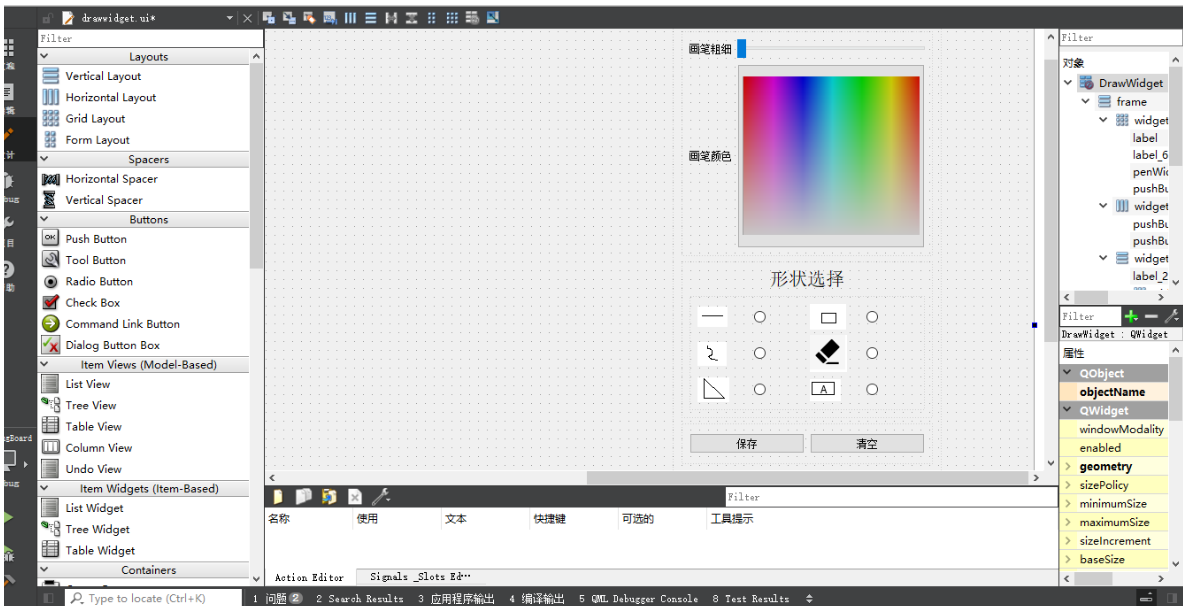
```
#include "drawwidget.h"
#include "ui_drawwidget.h"
#include <QPainter>
#include <QPen>
#include <QMouseEvent>
#include <QMessageBox>
#include <QColorDialog>
#include <QFileDialog>
#include <QLineEdit>
#include <QBrush>
DrawWidget::DrawWidget(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::DrawWidget)
{

    ui->setupUi(this);
    image=QImage(this->size().width()-420,this-
>size().height(),QImage::Format_RGB32);
    //image=QImage(980,780,QImage::Format_RGB32);//设定一张采用32位图（最常用的）的规
模为900*600的画布
    setting_color=qRgb(255,255,255);//选定背景色为白色
    image.fill(setting_color);//将背景色填充在画布上
    tempImage = image;

    drawing = false;
    shape = 0;
    ui->shape->setChecked(true);
    width = 0;
    heigh = 0;
    for(int i=0;i<3;i++)
    {
        pointPolygon[i].setX(0);
        pointPolygon[i].setY(0);
    }
    lineEdit.setParent(this);
    lineEdit.resize(70,20);
    lineEdit.setText(" ");
    lineEdit.setVisible(false);
   // ui->lineEdit->show()


}

DrawWidget::~DrawWidget()
{
    delete ui;
}
void DrawWidget::paintEvent(QPaintEvent *event)
{
    QPainter painter(this);
    if(drawing == true)
    {

        painter.drawImage(0,0,tempImage);   //鼠标按住但在拖动时在临时画布上画
    }
    else {
        painter.drawImage(0,0,image);//在image上绘画
    }
    //lineEdit.setVisible(false);

}

void DrawWidget::paint(QImage &theImage)
{
    QPainter thePainter(&theImage);
    QPen pen;

    pen.setWidth(ui->penWidth->value());
    pen.setColor(penColor);


    //draw
    thePainter.setPen(pen);
    //shape = 4;
   // tempImage.fill(setting_color);
    switch (shape) {
    case 0:thePainter.drawLine(change,point);change = point;break;
    case 1:thePainter.drawLine(from,point);break;
//thePainter.drawLine(from,to);break;
    case 2:thePainter.drawRect(from.x(),from.y(),width,heigh);break;
    case 3:thePainter.eraseRect(point.x(),point.y(),ui->penWidth->value()+5,ui-
>penWidth->value()+5);break;
    case 4:thePainter.drawPolygon(pointPolygon,3);break;
    case 5:
            lineEdit.move(point.x(),point.y());
            lineEdit.setVisible(true);
```

```
                thePainter.drawText(change,lineEdit.text());
                lineEdit.clear();
                if(lineEdit.text()!="")
                {
                    lineEdit.setVisible(false);
                }
            break;


    default:break;
    }

    thePainter.end(); //结束绘图
    update();

}

void DrawWidget::mousePressEvent(QMouseEvent *event)
{
    if(event->button()==Qt::LeftButton)
    {
        drawing = true;
        point = event->pos();
        from = event->pos();
        change = event->pos();
        width=0;heigh=0;
        pointPolygon[0]=point;
        pointPolygon[1].setX(point.x());

    }


}

void DrawWidget::mouseMoveEvent(QMouseEvent *event)
{
        point = event->pos();
        width = point.x()-from.x();
        heigh = point.y()-from.y();
        pointPolygon[1].setY(point.y());
        pointPolygon[2]=point;

        tempImage = image;
        if(shape == 0||shape==3)
        {
            paint(image);
        }
        else {
            paint(tempImage);
        }


}

void DrawWidget::mouseReleaseEvent(QMouseEvent *event)
{
    if(event->button()==Qt::LeftButton)
    {
        to = event->pos();
        point = event->pos();
        width = to.x()-from.x();
        heigh = to.y()-from.y();
        pointPolygon[2]=point;

        drawing = false;
        paint(image);

    }


}

void DrawWidget::on_radioButton_clicked()
{
    shape = 1;
}

void DrawWidget::on_radioButton_2_clicked()
{
    shape = 2;
}



void DrawWidget::on_pushButton_clicked()
{
    QColorDialog color;//调出颜色选择器对话框
        penColor = color.getRgba();

}

void DrawWidget::on_shape_clicked()
{
    shape = 0;
}

void DrawWidget::on_radioButton_3_clicked()
{
    shape = 3;
}

void DrawWidget::on_pushButton_2_clicked()
{
    QString filename = QFileDialog::getSaveFileName(this,
        tr("Save Image"),
        "",
        tr("*.bmp;; *.png;; *.jpg;; *.tif;; *.GIF")); //选择路径
    if(filename.isEmpty())
    {
        return;
    }
    else
    {
        if(! (image.save(filename) ) ) //保存图像
        {
            QMessageBox::information(this,
                tr("Failed to save the image"),
                tr("Failed to save the image!"));
            return;
        }
    }
}

void DrawWidget::on_radioButton_4_clicked()
{
    shape = 4;
}

void DrawWidget::on_pushButton_3_clicked()
{
    image.fill(setting_color);//将背景色填充在画布上
    update();
}

void DrawWidget::on_radioButton_5_clicked()
{
    shape = 5;
}
```

main.cpp

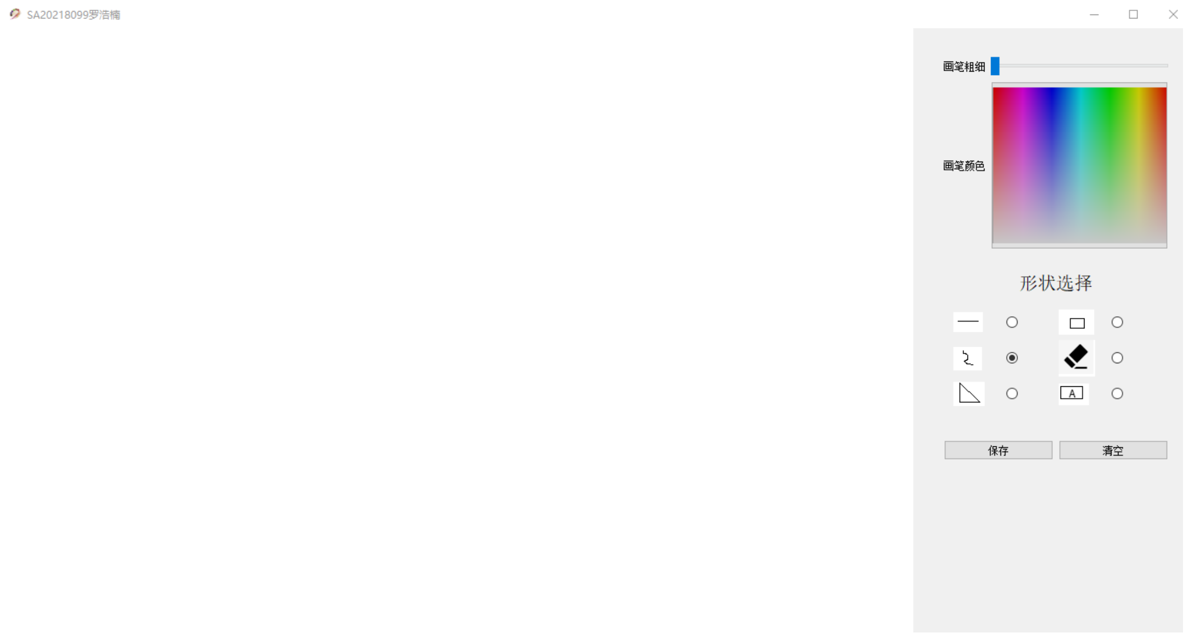
```
#include "drawwidget.h"
#include <QApplication>
#include <QIcon>

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    DrawWidget w;
    w.setWindowTitle("SA20218099罗浩楠");
    w.setWindowIcon(QIcon(":/image/tb.png"));
    w.show();

    return a.exec();
}
```

drawwidget.ui设计



运行样例

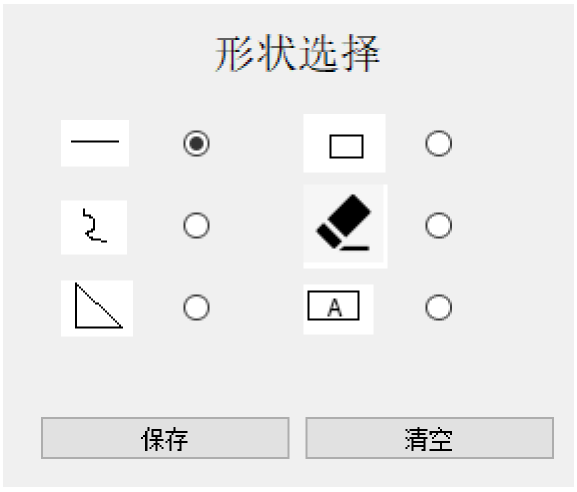
界面



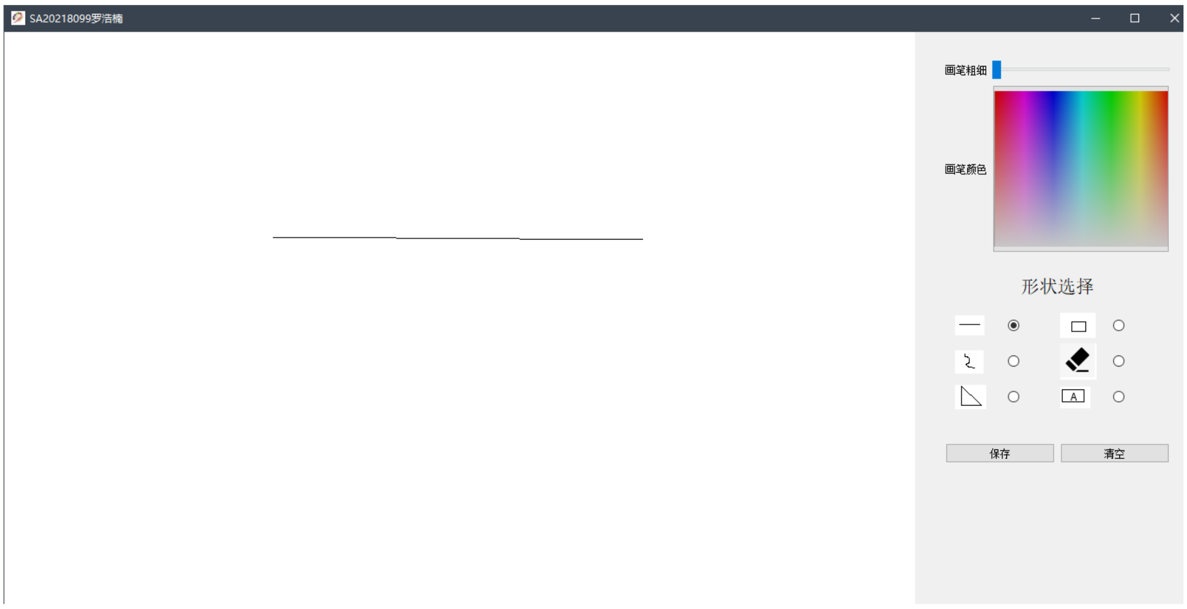
注:

左侧空白区域为工作区，即绘图区，右侧为工具栏，用于选择绘制图形的样式和保存等操作。

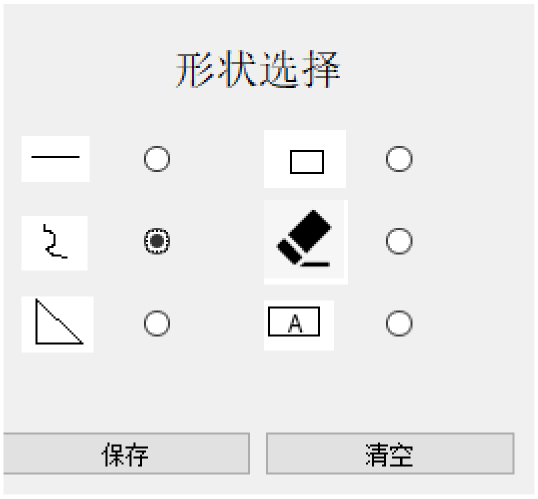
绘制直线



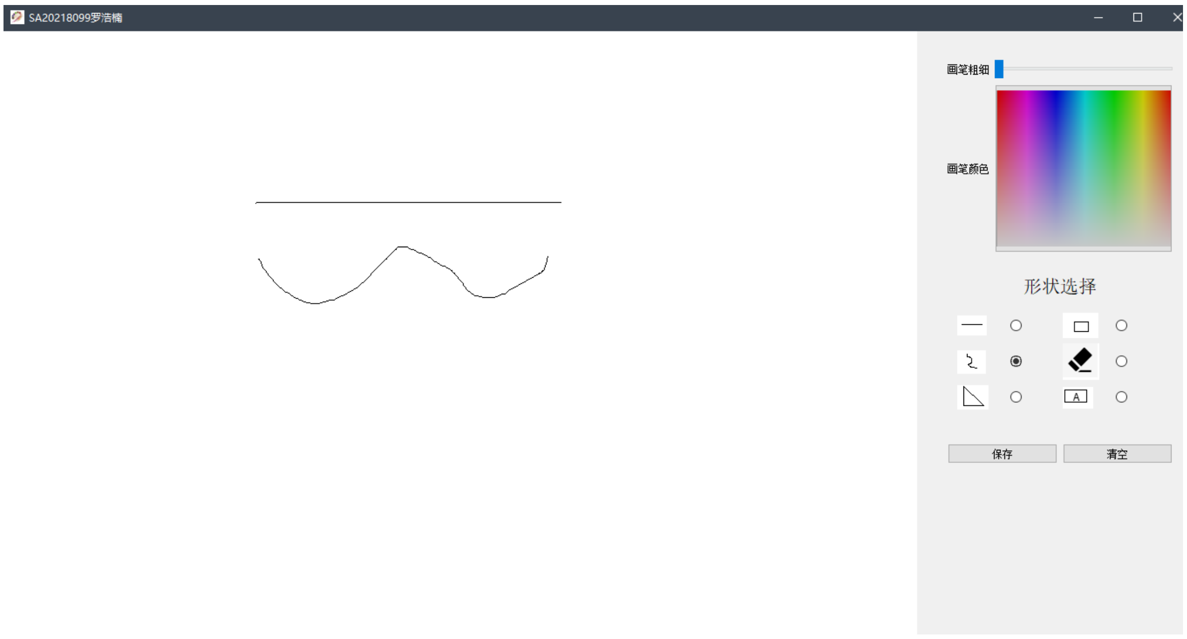
在右侧选择直线button后即可绘制直线，如下图所示：



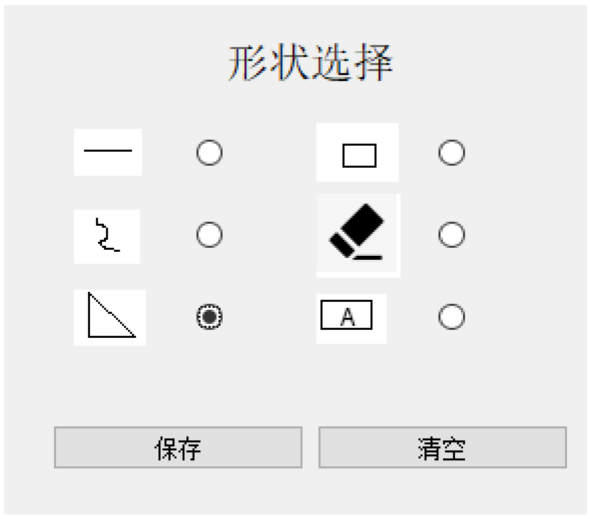
绘制线段



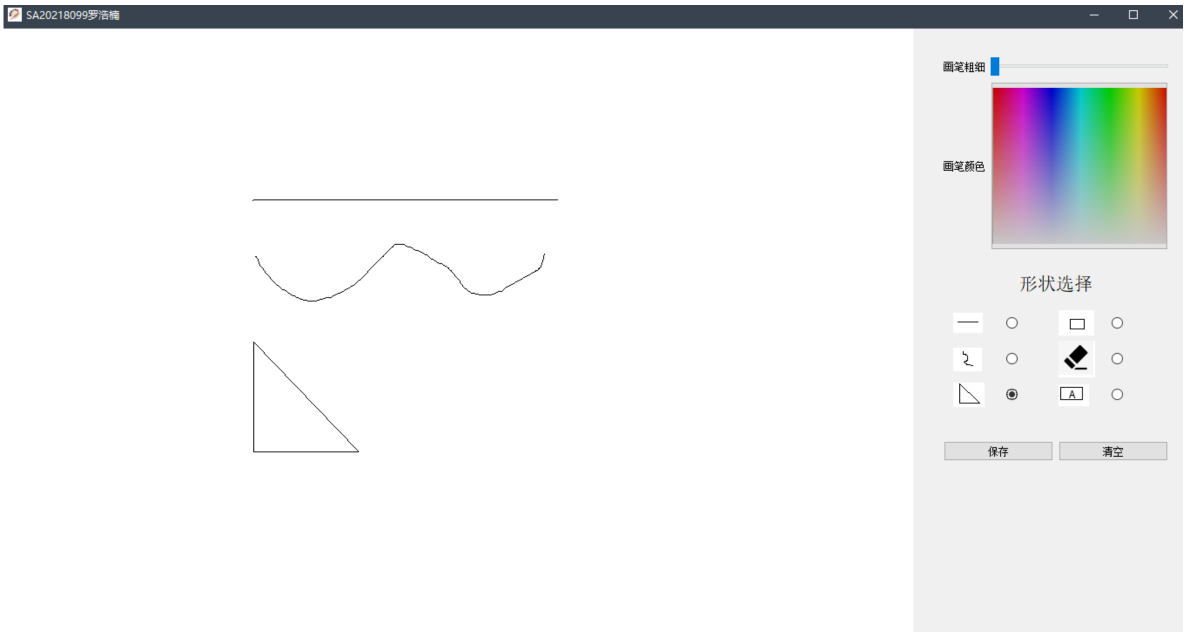
在右侧选择线段button后即可绘制线段，如下图所示：



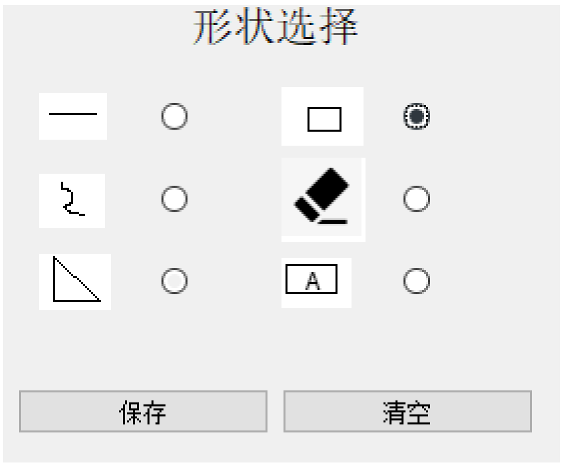
绘制三角



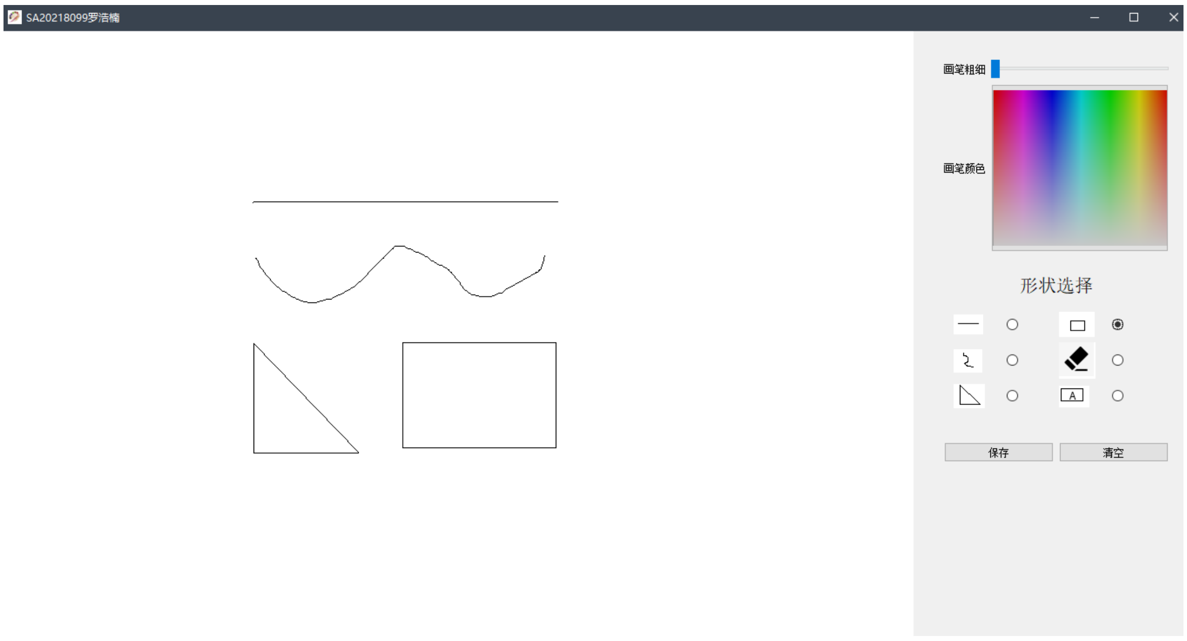
在右侧选择三角button后即可绘制三角，如下图所示：



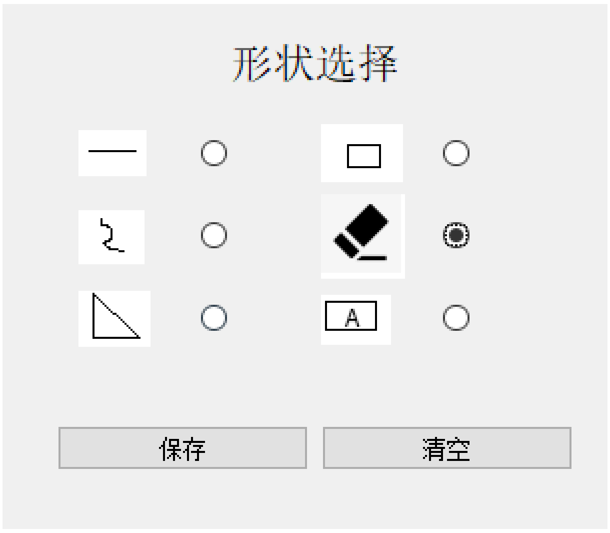
绘制矩形



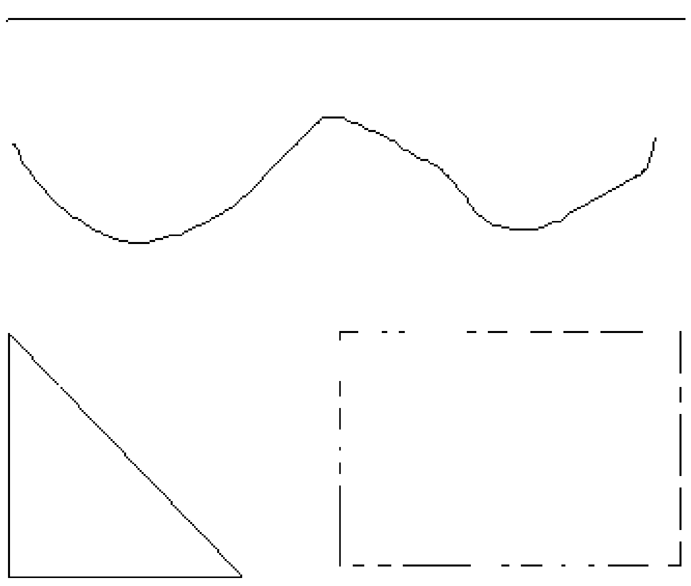
在右侧选择矩形button后即可绘制矩形，如下图所示：



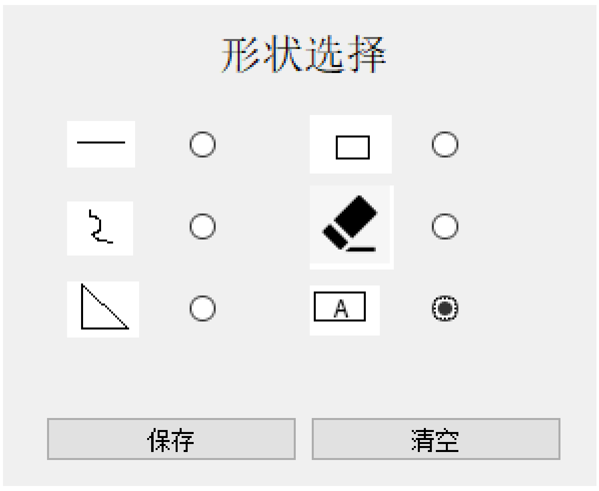
橡皮擦



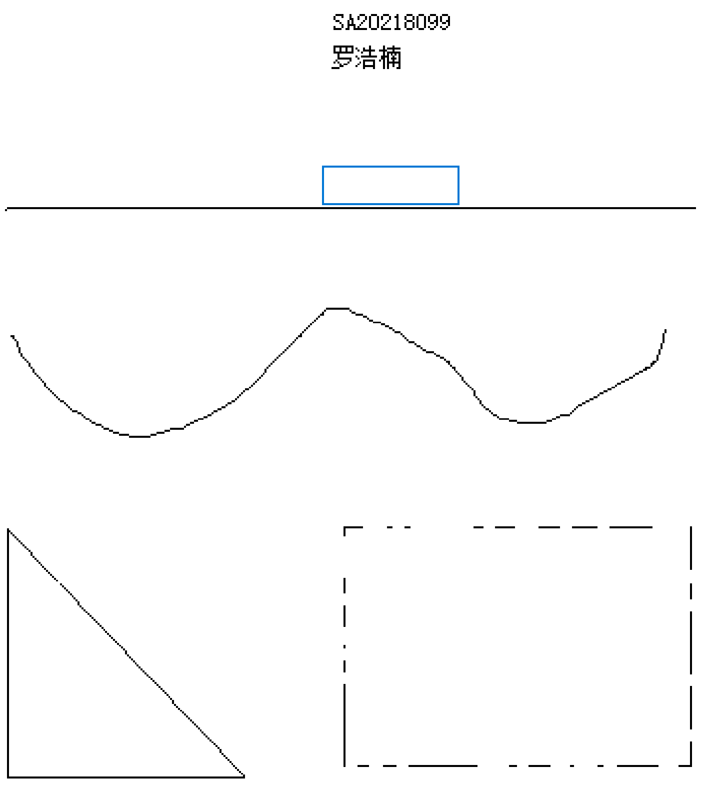
在右侧选择橡皮擦button后即可使用橡皮擦，如下图所示：



文字输入

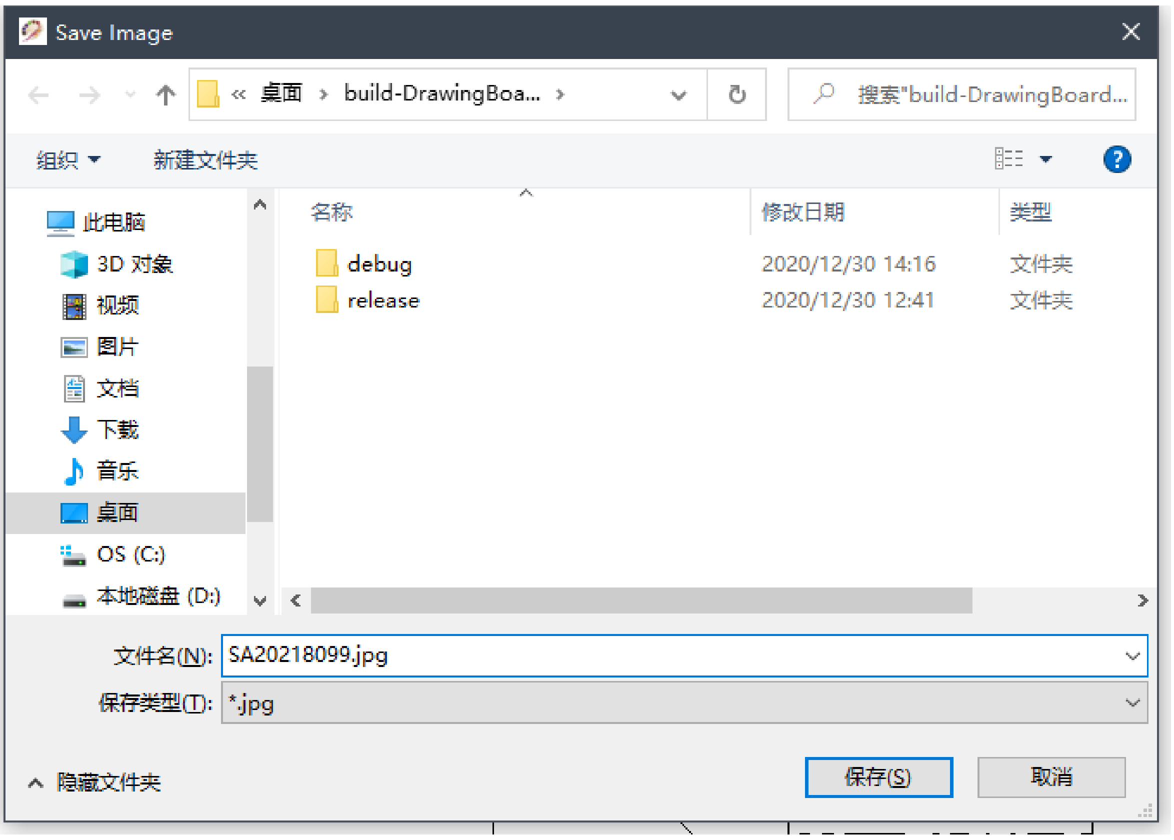


在右侧选择添加文字button后即可添加文字，如下图所示：



保存与清空

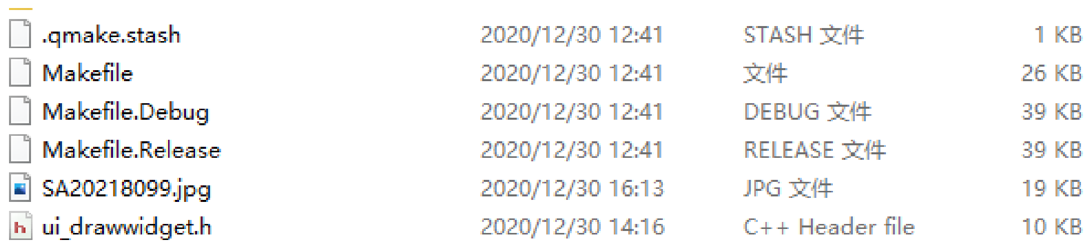
点击右侧保存按钮即可保存，如下图所示：



名称	修改日期	类型
debug	2020/12/30 14:16	文件夹
release	2020/12/30 12:41	文件夹

文件名(N): SA20218099.jpg

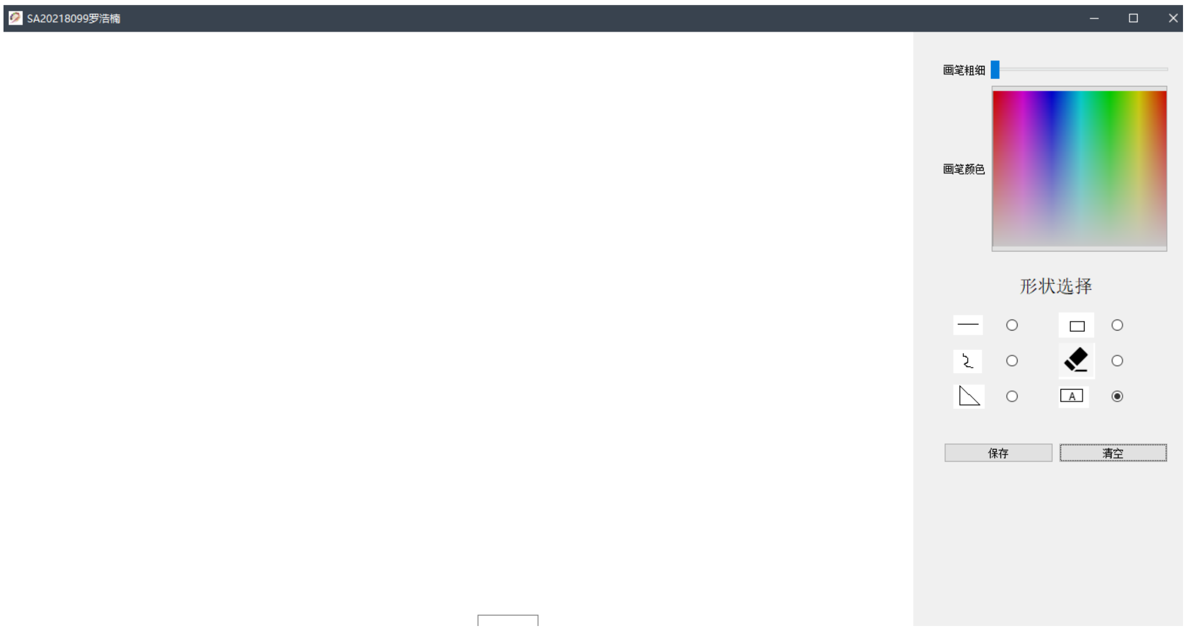
保存类型(T): *.jpg



.qmake.stash	2020/12/30 12:41	STASH 文件	1 KB
Makefile	2020/12/30 12:41	文件	26 KB
Makefile.Debug	2020/12/30 12:41	DEBUG 文件	39 KB
Makefile.Release	2020/12/30 12:41	RELEASE 文件	39 KB
SA20218099.jpg	2020/12/30 16:13	JPG 文件	19 KB
ui_drawwidget.h	2020/12/30 14:16	C++ Header file	10 KB

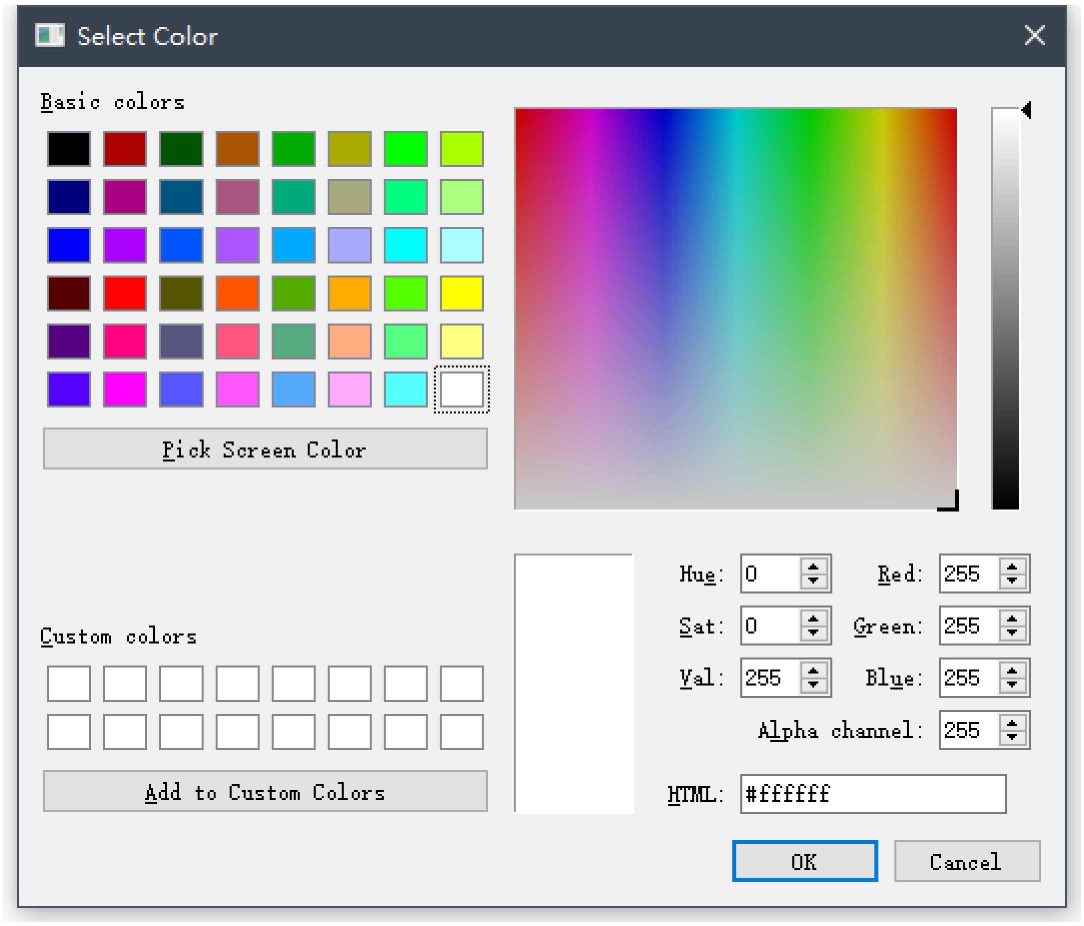
保存成功。

点击清空即可清空工作区所有绘制图形，如下图所示：



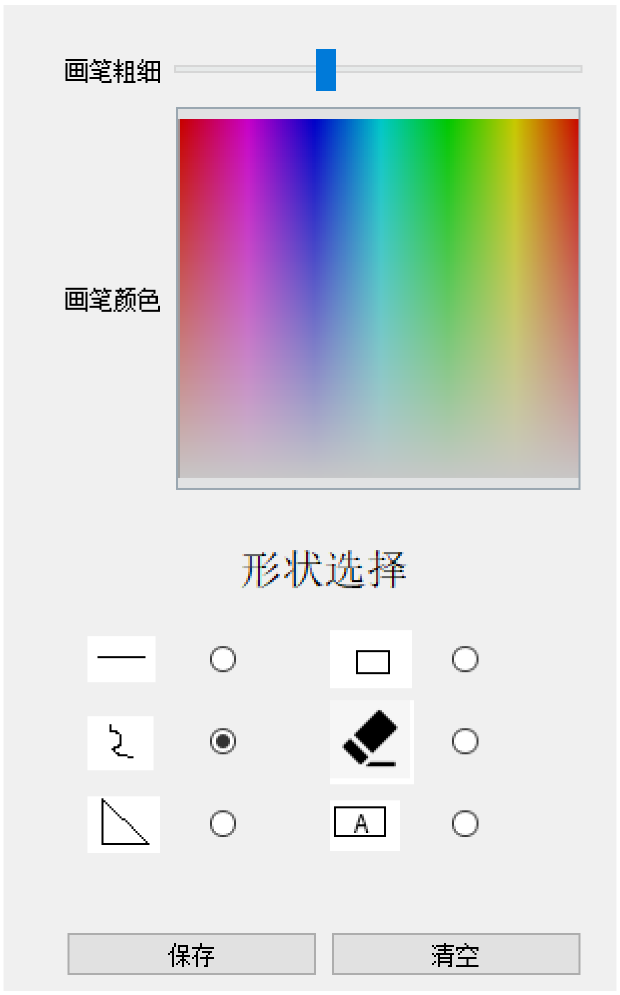
画笔粗细与画笔颜色

点击修改颜色可以选择想要的颜色，如下图所示：



Hue: 0
Sat: 0
Val: 255
Red: 255
Green: 255
Blue: 255
Alpha channel: 255
HTML: #ffffff

拖动拉杆可以设置线条粗细，下图是设置样例：



实验总结

通过本次实验，我收获良多，动手实现了基于C++的绘图软件，使我对C++有了更加深刻的认识，也对C++绘图更加了解。