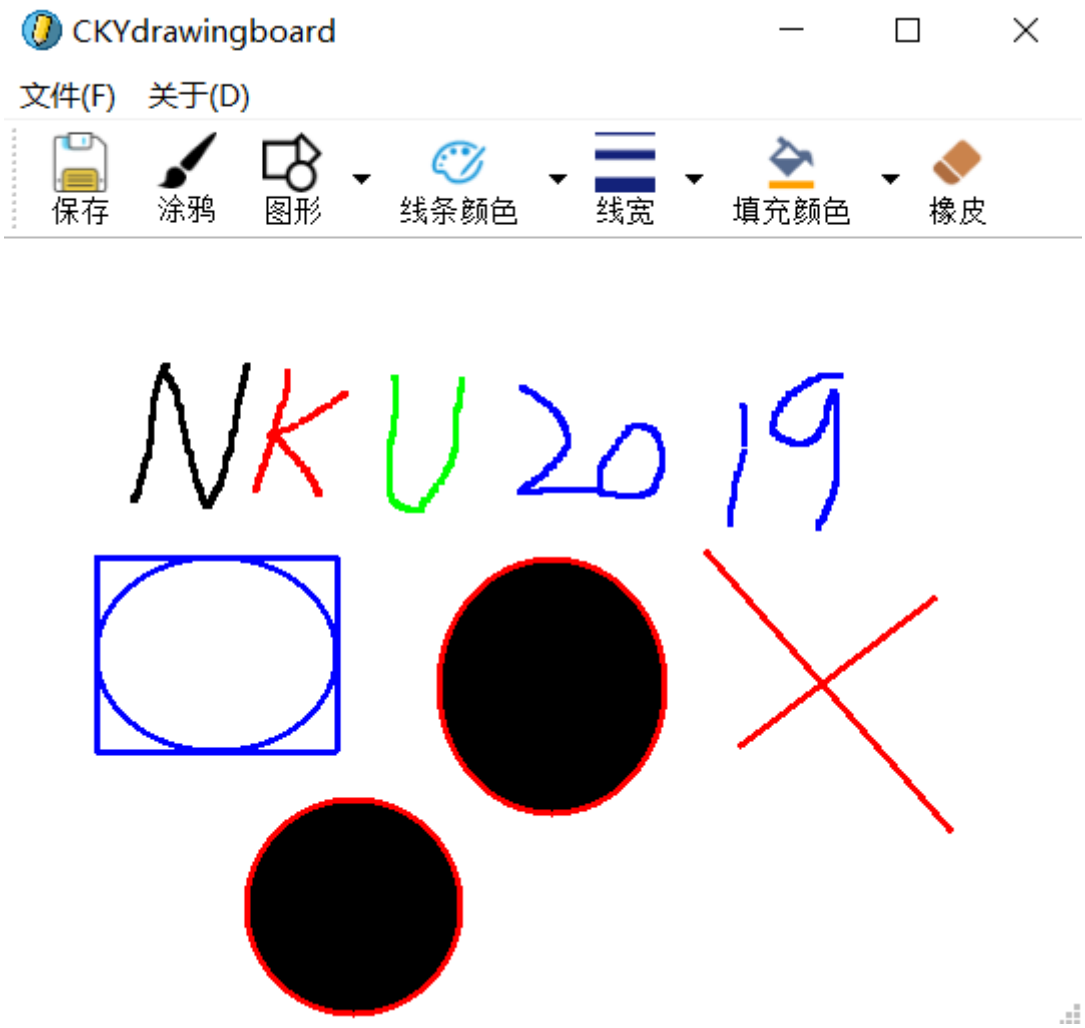
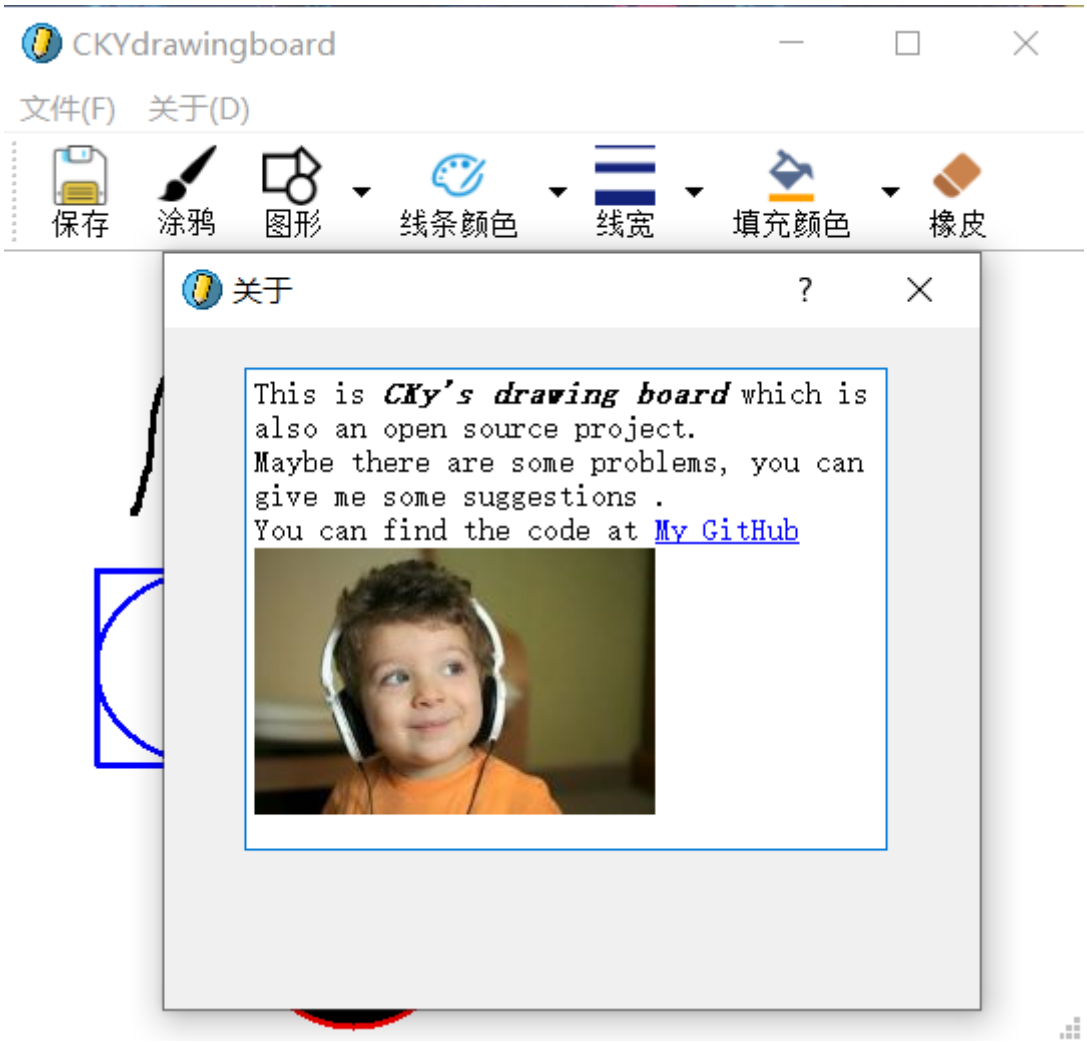
MY Qt Drawing Board Project

This project may not look so perfect(It does not have a lot of extended functions), because I have to prepare for the final project while dealing with this project.Please don't care.

The main UI about my drawing board.

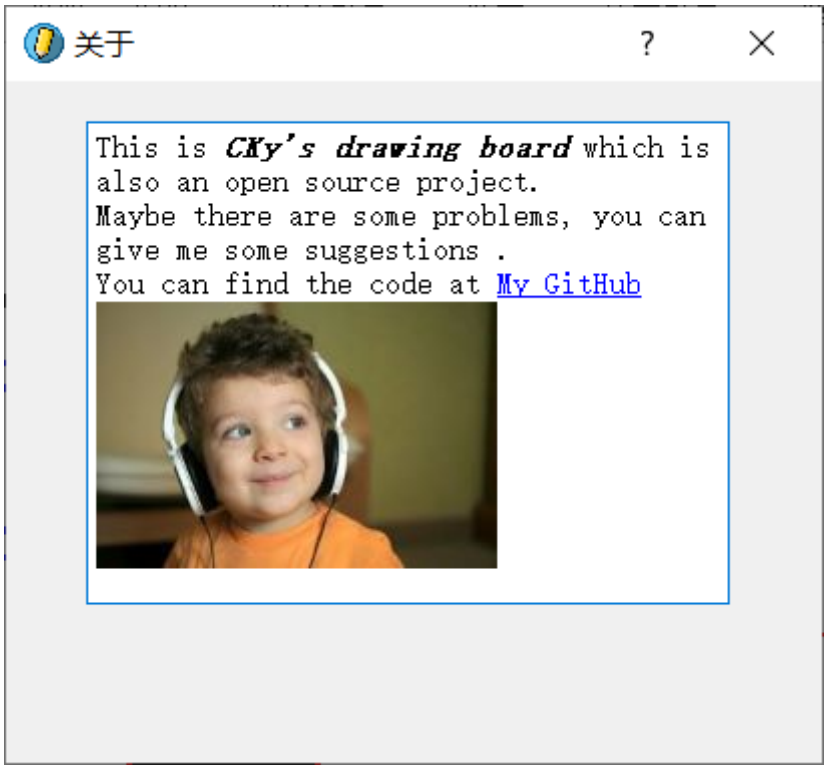

CKYdrawingboard
文件(F)
关于(D)
保存
涂鸦
图形
线条颜色
线宽
填充颜色
橡皮
NKU2019



About the *extra function* of my drawing board and its implementation

1. Having a Qaction to see about this program(designer)
   Implementation:Create a new Dialog without Buttons and add a slot function to the QAction to show this new Dialog which contents something About Designer. It has a picture and a hyperlink.

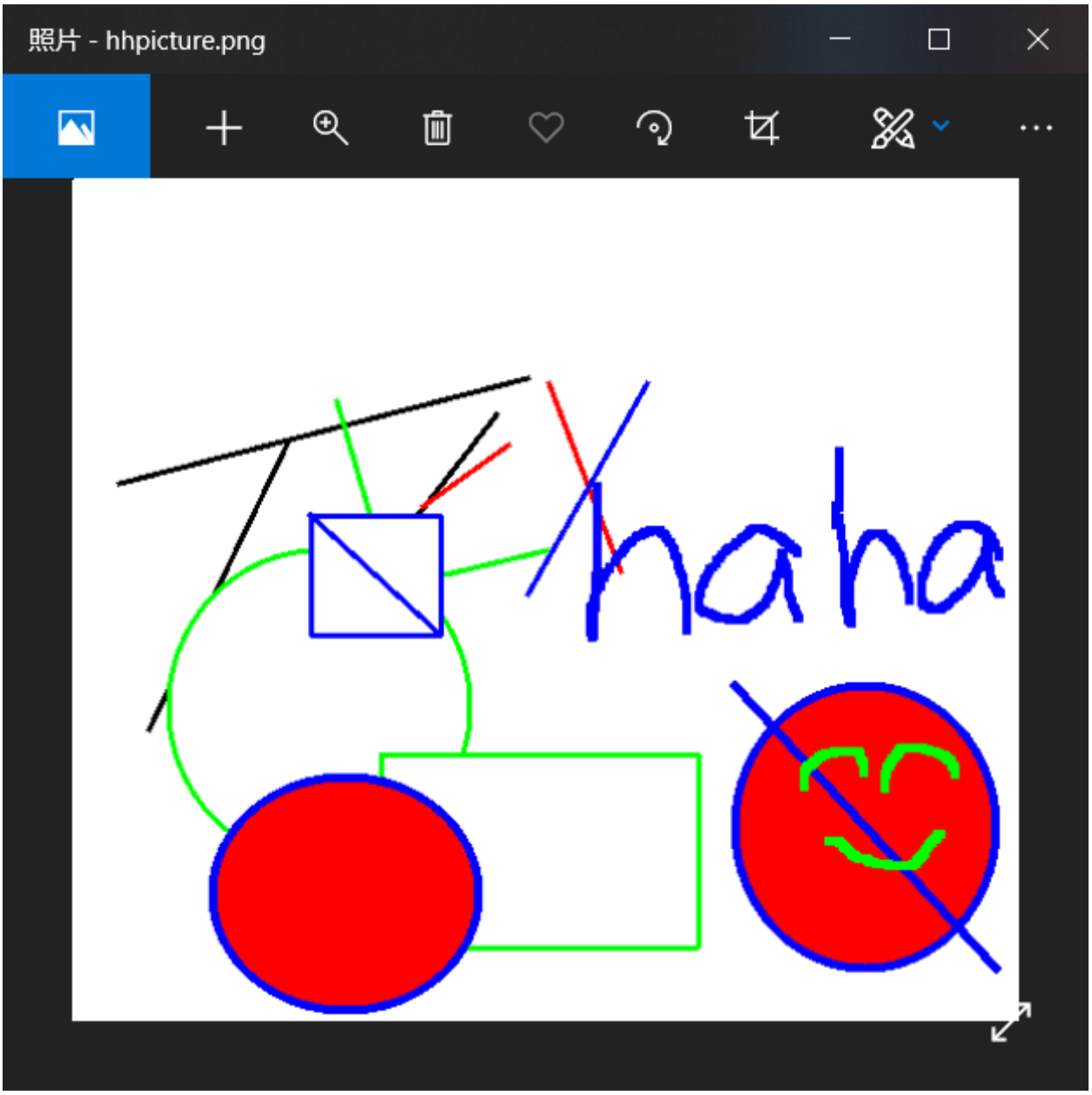
```
void Maindrawingboard::on_action123_5_triggered()
{
        about *about1=new about(this);
        about1->setWindowTitle(QStringLiteral("关于"));
        about1->show();
}
```



2. Having the SAVE function.

   Implementation:Use the member function save of class QPixmap object.The graffiti files will be saved in the project root directory as PNG.

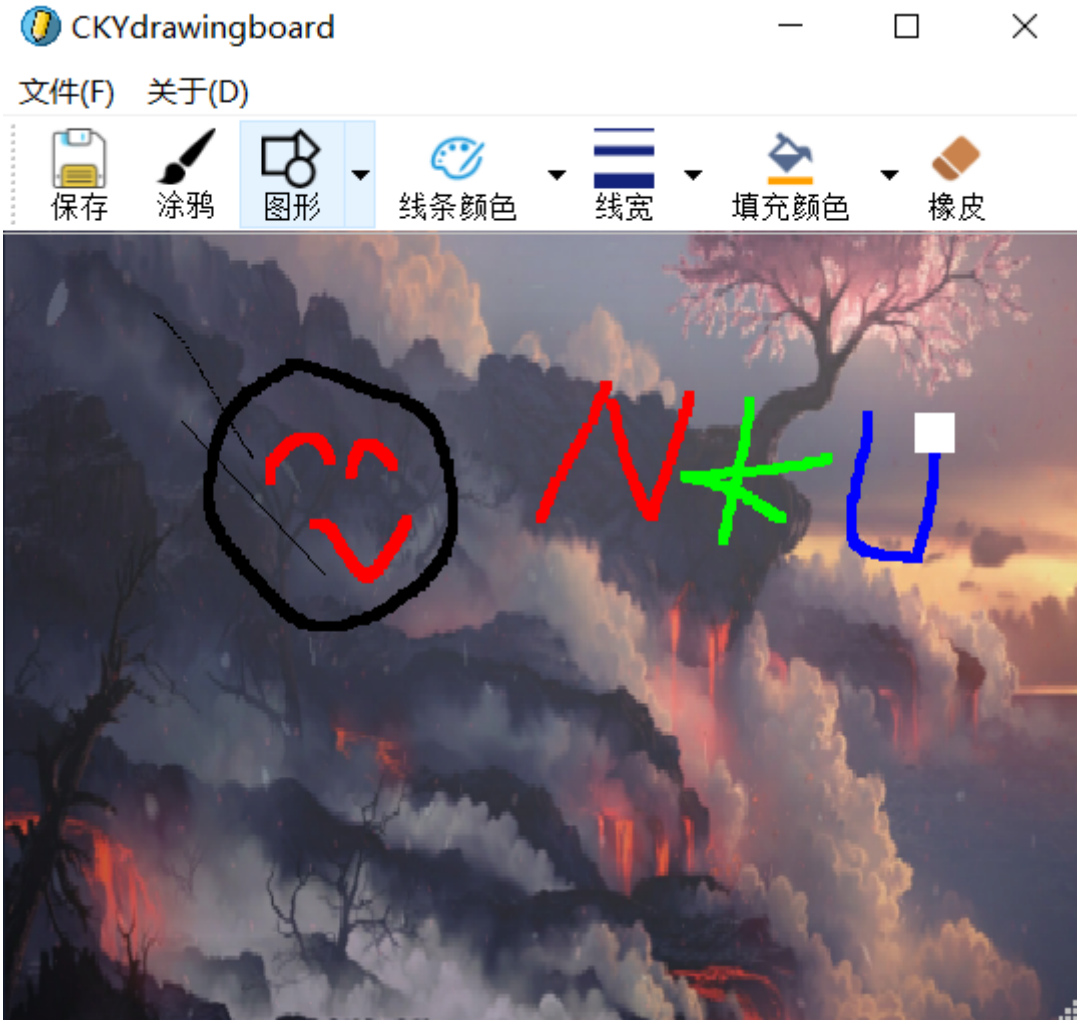
```
void Maindrawingboard::clicksave()
{
    pix.save("hhpicture.png","PNG");
    return;
}
```



3. Having the OPEN function.

   Implementation:Use the member funtion load of class QPixmap object.Then use the member funtion scale to adjust the size of picture you want to open.

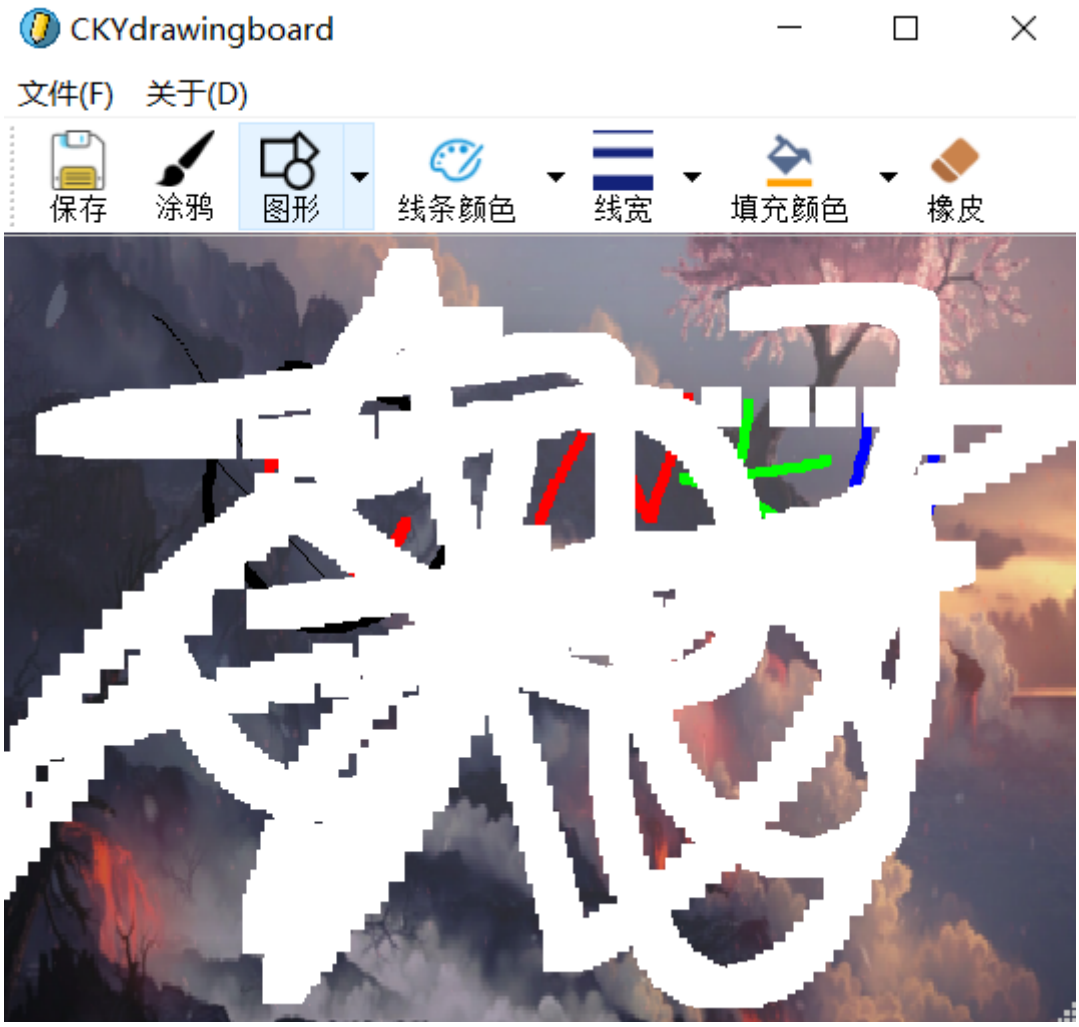
```
void Maindrawingboard::on_action123_2_triggered()
{
    QPixmap mypix;
    mypix.load("tree.png");
    pix =mypix.scaled(540,480);
    return;
}
```



4. Having the eraser function.

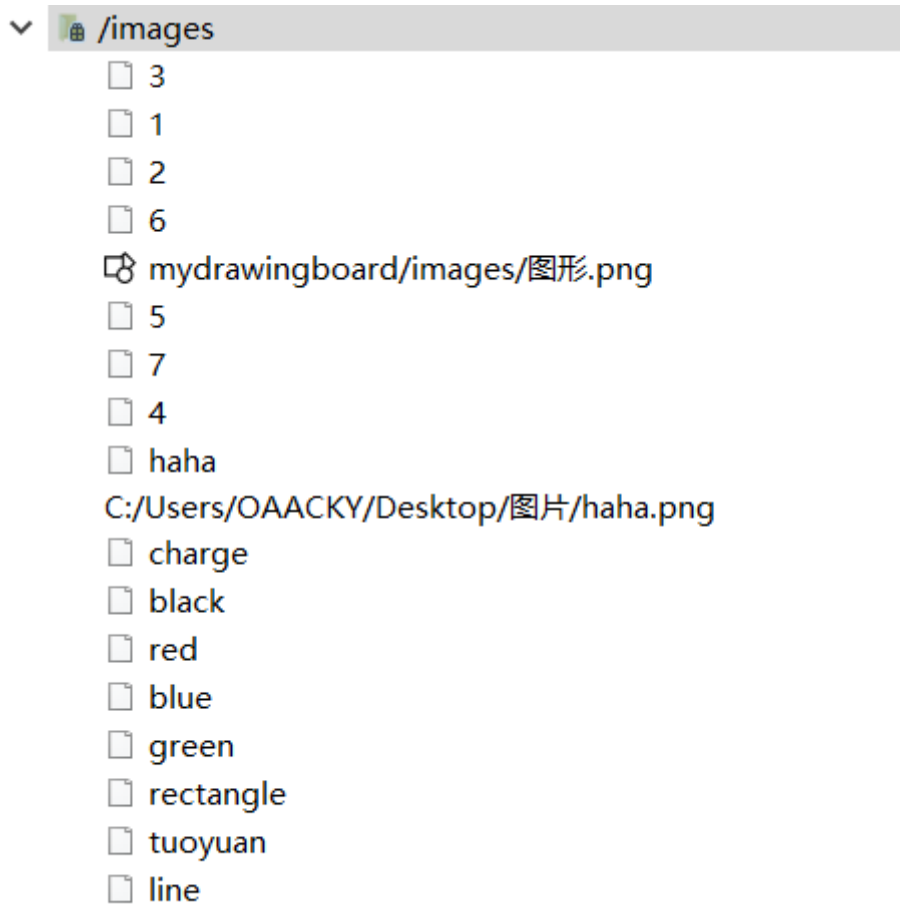
   Implementation:In fact, the eraser is just another large white pen.Create a slot funtion and set the Qpen is OK.

```
void Maindrawingboard::clickeraser()
{    ispen=1;
     mypen.setColor(QColor(255,255,255));
     mypen.setWidth(20);
     return;
}
```



Some technical details

1. Using resource system.
   Implementation: Add a new class named Qt resource file. Then add your files into the folder.
   When you want to use some files ,you can just only by using :/name .



2. Add Qmenu to the QTool button,and add QAction to the QTool.
   Implementation:Create QMenu, QAction class objects, and use their member functions.

```
QMenu *colorMenu = new QMenu(this);
QMenu *shapemenu =new QMenu(this);
QMenu *fillcolormenu =new QMenu(this);
QMenu *widthmenu=new QMenu(this);
QAction *pencolor[4];
QAction *fillcolor[4];
QAction *shape1[3];
QAction *width1[3];
```

```
for(int i=0;i<4;i++)
{
colorMenu->addAction(pencolor[i]);
fillcolormenu->addAction(fillcolor[i]);
   if(i<3)
      { shapemenu->addAction(shape1[i]);
        widthmenu->addAction(width1[i]);
       }
}

toolBtn2->setMenu(shapemenu);
toolBtn2->setPopupMode(QToolButton::MenuButtonPopup);
```

3. Like the previous project, I used Qsignalmapper to connect the signal to the slot function. Implementation:First connect the Qaction-triggered event with the map() in signalmapper,then use setmapping function. At last connect the map with slot function.

```
QSignalMapper* signalMapper3 = new QSignalMapper;        △ 'QSignalMapper'
for(int i=0;i<4;i++)
{
 pencolor[i]=new QAction;
 connect(pencolor[i],SIGNAL(triggered()),signalMapper2,SLOT(map()));
 signalMapper2->setMapping(pencolor[i],i);
 fillcolor[i]=new QAction;
 connect(fillcolor[i],SIGNAL(triggered()),signalMapper3,SLOT(map()));
 signalMapper3->setMapping(fillcolor[i],i+1);
 if(i<3)
    { shape1[i]=new QAction;
     connect(shape1[i],SIGNAL(triggered()),signalMapper,SLOT(map()));
     signalMapper->setMapping(shape1[i],i+1);
     width1[i]=new QAction;
     connect(width1[i],SIGNAL(triggered()),signalMapper1,SLOT(map()));
     signalMapper1->setMapping(width1[i],i*2+1);
 }
}
connect(signalMapper, SIGNAL(mapped( int )),this, SLOT(clickshape(int)));
```

4. The change line color and change line width funtion. Implementation:Both them are same as the change shape and fill color funtion.I just need to write the corresponding slot function to change Qpen or the QBrush's property.Sometime I should also judge their property in the paintevent funtion.

```
void Maindrawingboard::clickfillcolor(int t)
{
    if(t==1)
    mybrush.setColor(QColor(0,0,0));
    else if (t==2) {
        mybrush.setColor(QColor(255,0,0));
    }
    else if (t==3) {
        mybrush.setColor(QColor(0,255,0));
    }
    else if (t==4) {
        mybrush.setColor(QColor(0,0,255));
    }
    return ;
}

void Maindrawingboard::clickshape(int t)
{
    ispen=0;
    myshape=t;
    return;
}

void Maindrawingboard::clicklinewidth(int t)
{

    mypen.setWidth(t);
    return;
}
```

5. Draw a graphic.

   Implementation:Draw with double-buffers.I will use two maps,a temp pixmap and the realpixmap.I will first draw the content onto tempPix, then draw tempPix onto the interface.The readpixmap will save the picture I have already finished.When I release the mouse to complete the drawing of the graphic, copy the contents of tempPix to the readpixmap.

```
void Maindrawingboard::paintEvent(QPaintEvent *event)
{
    int x = startPoint.x();
    int y = startPoint.y();
    int width = endPoint.x() - x;
    int height = endPoint.y() - y;
    QPainter painter;
    if(!ispen)
  {  painter.begin(&tempPix);
    painter.setPen(mypen);
    painter.setBrush(mybrush);

    if(myshape==1)
    painter.drawLine(x, y, endPoint.x(), endPoint.y());
    else if (myshape==2) {
         painter.drawRect(x, y,width, height);

    }
    else if(myshape==3)
    {
        painter.drawEllipse(x,y,width,height);
    }
    painter.end();
    painter.begin(this);
    painter.drawPixmap(0, 0, tempPix);
    if(!isDrawing)
        pix = tempPix;
    }
```

6. Drawing graffiti.

   Implementation:Use the mouse click, move, release event well.Graffiti is like a lot of straight lines.So I should to update startpoint in the event of mouse movement,then draw a line between startpoint and the endpoint.

```
void Maindrawingboard::mousePressEvent(QMouseEvent *event)
{
    if(event->button() == Qt::LeftButton) {
        startPoint = event->pos();
        if(!ispen)
       {  isDrawing = true;}
        else
        {
          endPoint=startPoint;
        }
    }
}

void Maindrawingboard::mouseMoveEvent(QMouseEvent *event)
{
    if(event->buttons() & Qt::LeftButton) {
        endPoint = event->pos();
        if(ispen)
        startPoint=endPoint;
        tempPix = pix;
        update();
    }
}

void Maindrawingboard::mouseReleaseEvent(QMouseEvent *event)
{
    if(event->button() == Qt::LeftButton) {
        endPoint = event->pos();
        if(!ispen)
        isDrawing = false;
        update();
    }
}
```

```
painter.begin(&pix);
    painter.setPen(mypen);
    painter.drawLine(startPoint,endPoint);
    painter.end();
    painter.begin(this);
    painter.drawPixmap(0,0,pix);
```

The problems I meet and solve.

1. Failed to open a .png file by using the relative path.
   The reason is that I did't put the file in the resource folder.
   Solve:Use this function: QDir::setCurrent(app.applicationDirPath())
2. Faild to use the resource system.
   Solve:The path is incorrect,change it to the right path.
3. Painting with Qpainter does not display lines.(QWidget::paintEngine: Should no longer be called
   QPainter::begin: Paint device returned engine == 0, type: 1)
   The reason is that I use Qpainter in the constructor function.The class object has not been created yet. So I can just only use it in its member function.
4. Why is the member function PaintEvent() executed without a call?
   Reason:The function PaintEvent() executes when building the class object it belongs to.
5. Prompt error: QPainter::setPen: Painter not active
   Reason:The QPainter object calls setpen() before its begin() function. So I just put the function between the begin() and the end().
6. I used QBrush but no fill effect.
   Reason:The default style is Qt :: NoBrush, so I should define the fill style first.

Reference

> 阿里巴巴矢量图标库
> Qt学习二、添加资源文件
> 关于Qt Creater中资源文件和文件路径的记录
> qt中的菜单QMenu QAction
> pushbutton文字在图片正下方
> QPaintEvent原理
> QT关键问题解决之paintevent理解
> Qt 2D绘图之一：基本图形绘制和渐变填充
> qt控件绘图