共享白板设计文档

在经历了10天的团队开发之后，主要完成了如下内容：

1. 完成了基本功能，匿名加入白板，快速创建白板，添加新页，设置编辑权限，以及在白板中创建各类图形（直线，矩形，椭圆，自由曲线等）。
2. 服务部署在了自己的服务器上，IP地址123.57.187.239，通过端口号8000可以访问
3. 基于websocket实现了web端和windows桌面端的跨平台。
4. web端采用js+django进行搭建，windows采用QWebsocket+ui实现。
5. 代码托管在https://github.com/QiNiu-Hackathon/White_Board

需求分析

1. 用户创建房间，其他人根据房间号加入并进行涂鸦。
2. 在白板创建图形时，其他人实时看到。
3. 白板创建者默认拥有管理员权限，可以修改其他用户的权限。

开发与进度管理

分工

设计：
网页客户端设计：崔孝俊
QT客户端设计：王冠博
跨平台服务端设计：岳金凤

开发

模块开发及分工

1. 网络连接模块：岳金凤
2. QT界面显示模块：王冠博
3. 网页显示模块：崔孝俊

开发语言

C++ JavaScript Python

进度管理

时间：内容
11.3–11.6 资料的查找与架构设计
11.7–11.10 web与QT客户端画板功能的实现，web服务器的设计
11.11–11.12 web、QT客户端与服务器的连接，功能同步

11.13日 收尾工作，代码优化

系统功能介绍

客户端功能介绍

网页URL：http://123.57.187.239:8000/

初始界面

匿名加入白板

输入房间号加入白板

快速创建白板

白板界面

工具栏

新建/删除白板

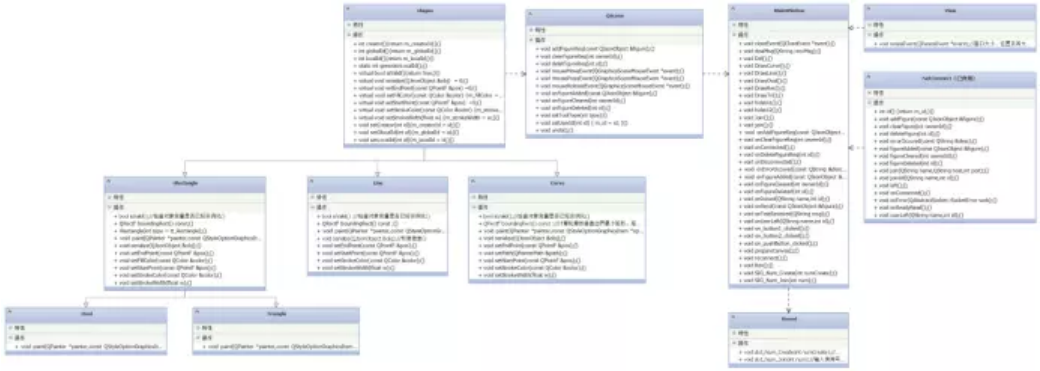
邀请加入/离开房间

基础绘画功能：曲线画笔 直线

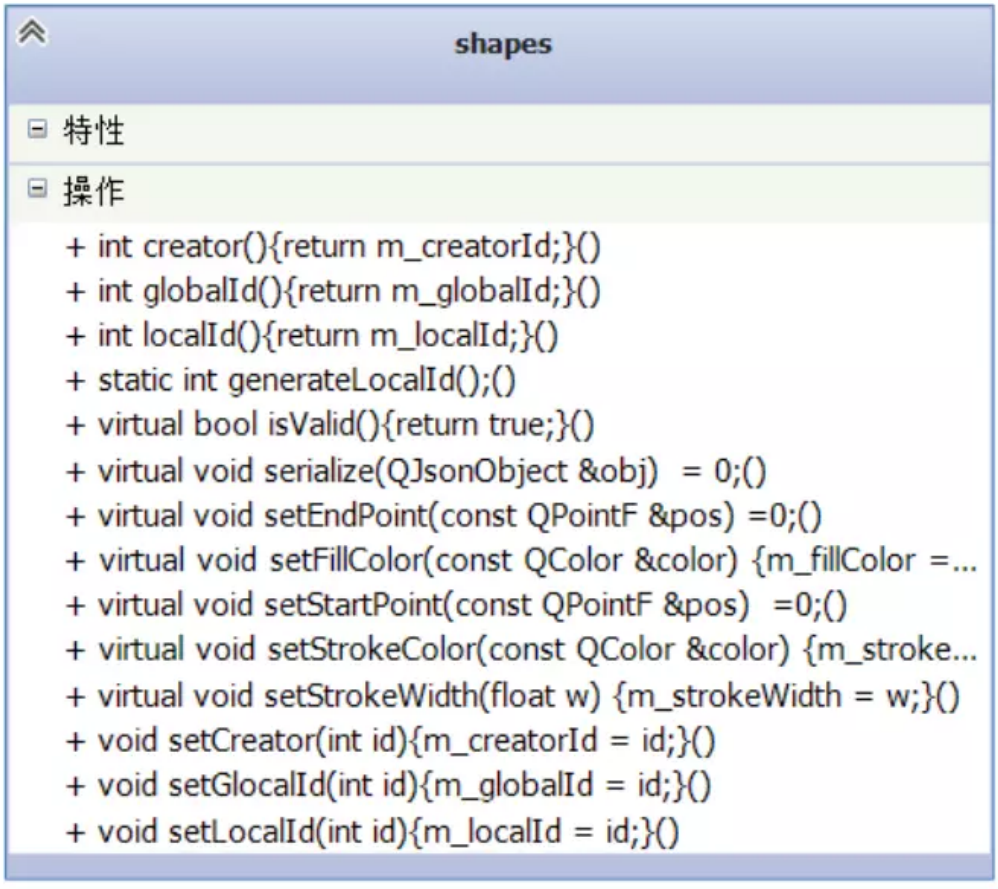
基础图形：矩形 椭圆 三角形

清空画布功能

QT客户端整体框架（类图）



主要函数及功能



`virtual void setStartPoint(const QPointF &pos) =0;`:设置绘制起点

virtual void setEndPoint(const QPointF &pos) =0;:设置绘制终点

virtual void setStrokeWidth(float w) {m_strokeWidth = w;}:设置线宽

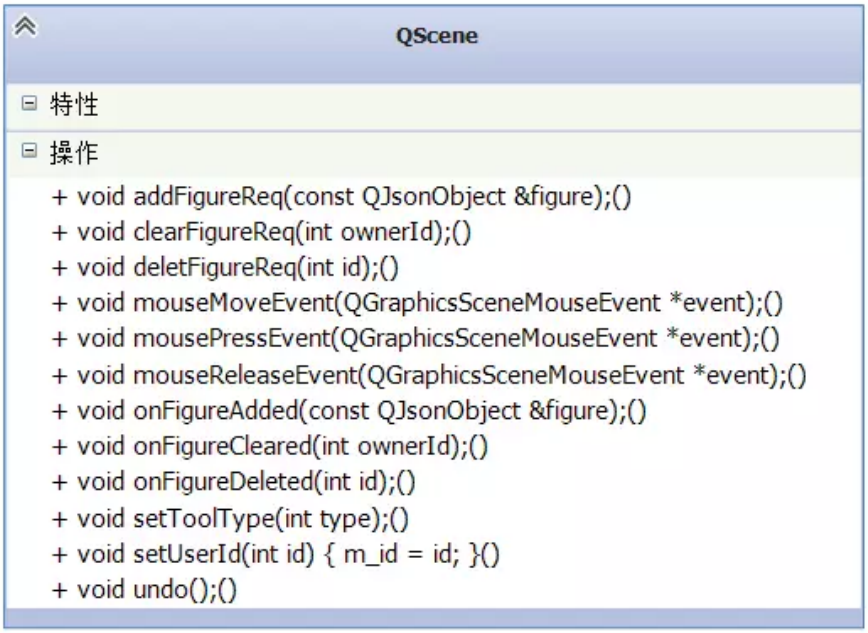
virtual void setStrokeColor(const QColor &color) {m_strokeColor = color;}:设置线条颜色

virtual void setFillColor(const QColor &color) {m_fillColor = color;}:设置填充颜色

virtual bool isValid(){return true;}:检查对象变量是否已经实例化

virtual void serialize(QJsonObject &obj) = 0;:封装信息为QJson

Line、rRectangle、Curve类继承于Shapes类，Oval、Triangle类继承于rRectangle类，内容主要是对以上函数进行实现。



public:

void setToolType(int type);:设置工具类型

void undo();:撤销上一步操作

void onFigureAdded(const QJsonObject &figure);:添加图元操作

void onFigureDeleted(int id);:删除图元操作

void onFigureCleared(int ownerId);:清空图元操作

signals:

void addFigureReq(const QJsonObject &figure);:请求添加图元信号

void deletFigureReq(int id);:请求删除图元信号

void clearFigureReq(int ownerId);:请求清空图元信号

protected:

void mousePressEvent(QGraphicsSceneMouseEvent *event);:鼠标按下事件

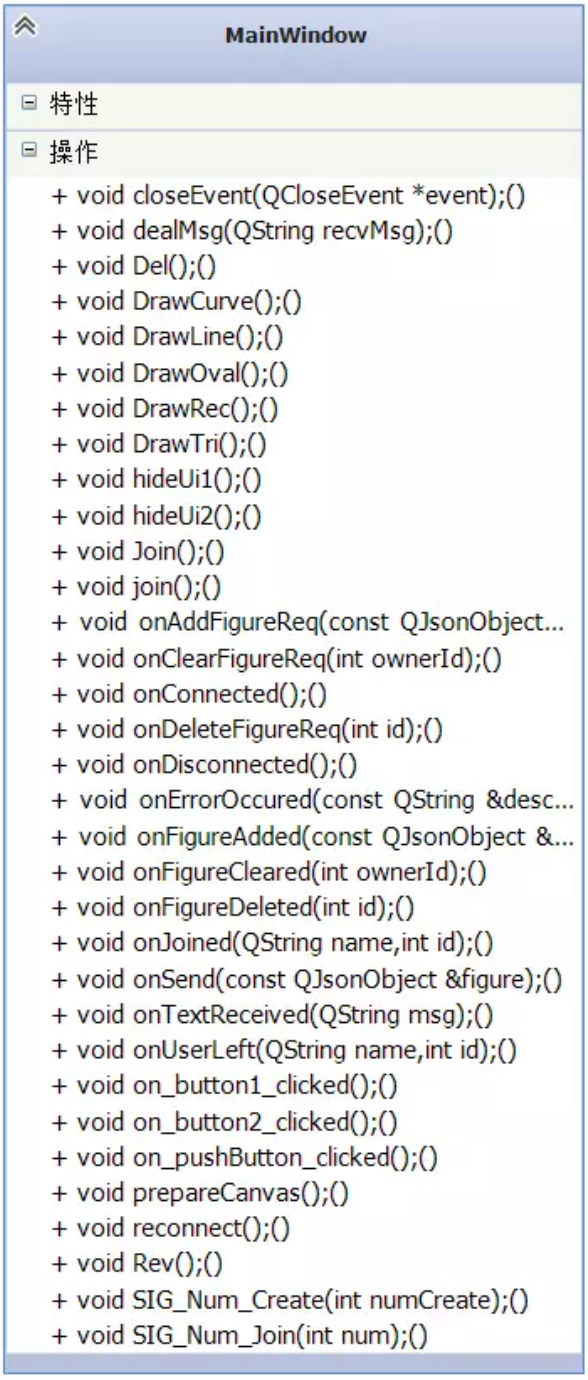
void mouseMoveEvent(QGraphicsSceneMouseEvent *event);:鼠标移动事件

void mouseReleaseEvent(QGraphicsSceneMouseEvent *event);:鼠标抬起事件

QScene类依赖于Shapes类，实现了鼠标绘制各种图形以及添加图元等操作。

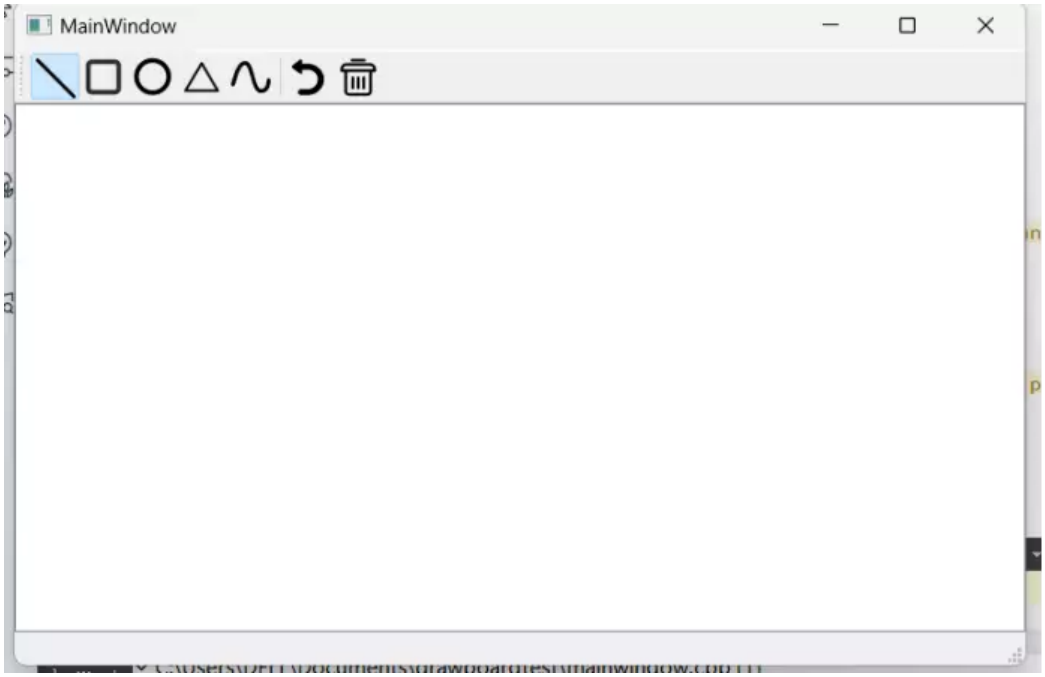
void resizeEvent(QResizeEvent *event);:窗口大小、位置及其大小改变引起的事件

QResizeEvent()




```
public:
  void hideUi1();
```

快速创建白板



点击创建白板时的隐藏ui函数

```
void hideUi2();
```

选择加入时，隐藏ui的函数

```
void prepareCanvas();准备画布的函数，包括按钮的绘制与槽函数的绑定
```

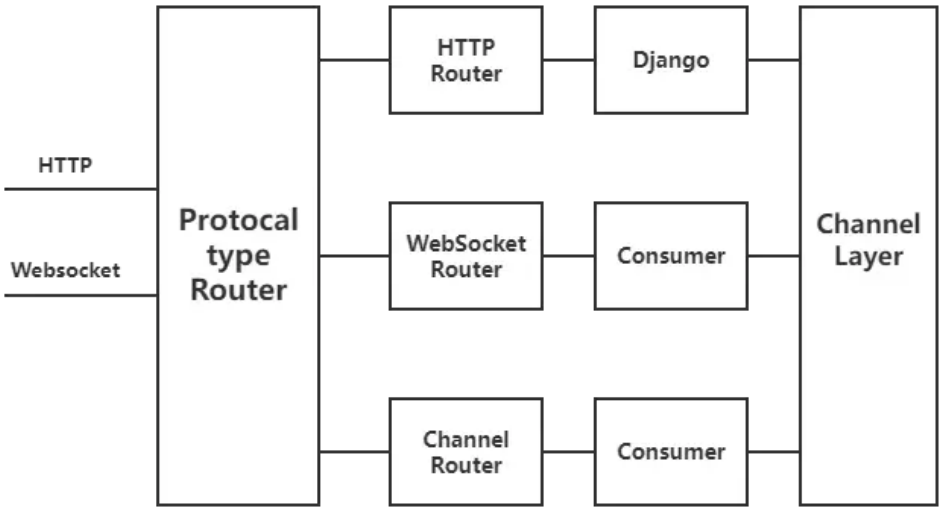
```
void closeEvent(QCloseEvent *event);重写关闭窗口事件函数
void  dealMsg(QString recvMsg);:处理收到的信息，讲QString类型转为QJsonObject类型并进行分析处理
void Join();加入服务器函数
private slots:
void on_button2_clicked();匿名加入按钮
void on_button1_clicked();快速创建函数
void on_pushButton_clicked();房间号确定按钮，获取文本框内的内容
public slots:
void DrawCurve();自由绘制函数
void DrawLine();直线绘制函数
void DrawOval();椭圆绘制函数
```

```
void DrawRec();矩形绘制函数
void DrawTri();三角形绘制函数
void Rev();撤销函数
void Del();清空函数
void onConnected(); () socket建立成功后，触发该函数
void onTextReceived(QString msg);收到Sev端的数据时，触发该函数
void onDisconnected();socket连接断开后，触发该函数
void reconnect();
void onSend(const QJsonObject &figure); 将消息由QJson转为QString类型并发送给服务器
```

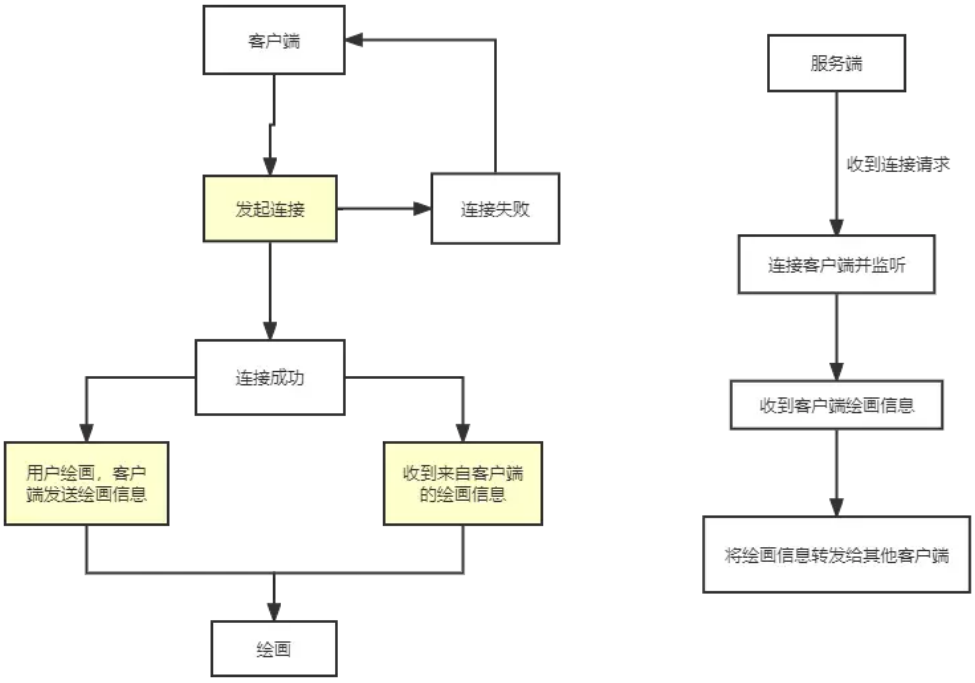
主窗口类

客户端-服务器端连接

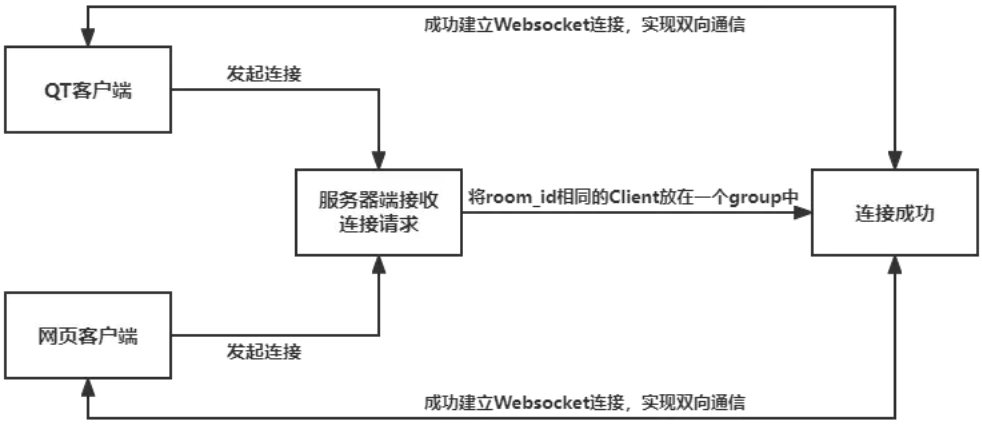
web一般的模式是客户端发送请求给服务端，服务端给出响应，在一些需要服务端主动发送消息给客户端时，一般采用客户端轮循访问服务端获取信息，这种方式非常占用资源，因此我们使用WebSocket实现服务器端和客户端的TCP连接，使两者都能主动发送信息给对方。

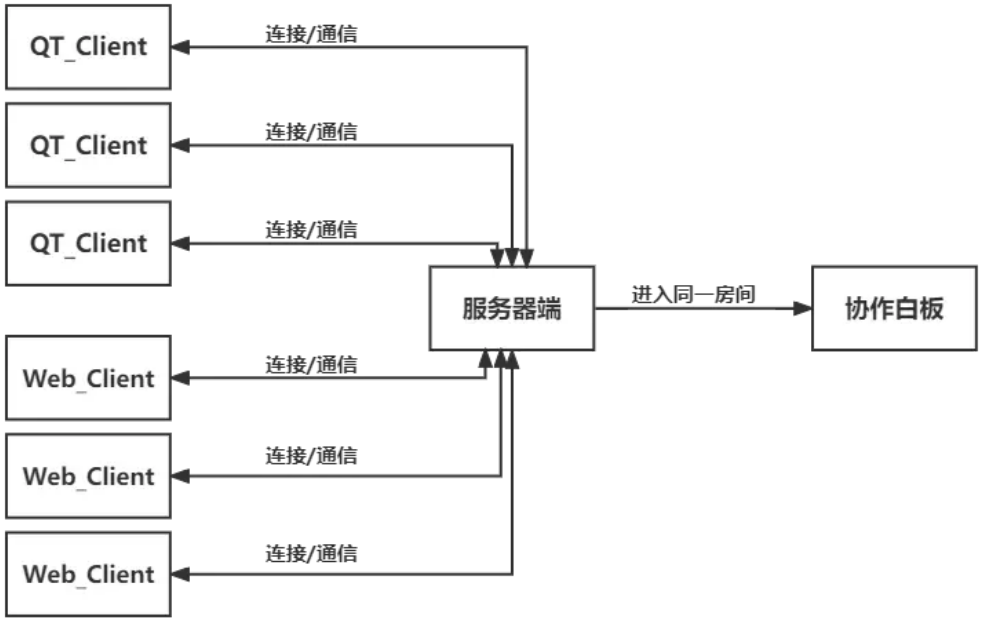


客户端流程和服务器流程如下图：



架构图





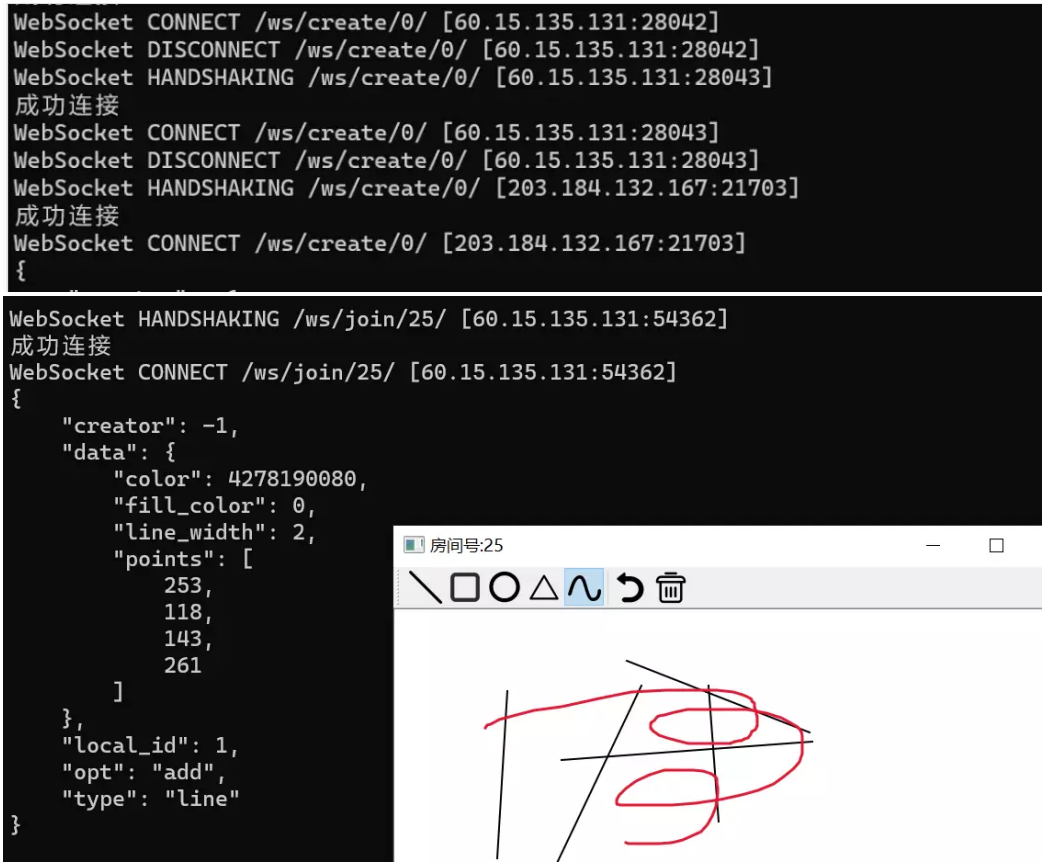
客户端Websocket

主要功能函数	作用
void join()	建立信号和槽函数的connect，并通过open函数连接服务器
void onConnected()	监听是否连接成功
void onTextReceived(QString msg)	监听是都收到服务器发来的信息
void onDisconnected()	监听是否断开连接
void reconnect();	断开连接时会触发该槽函数，并重启定时器重新连接服务器。
void onSend(const QJsonObject &obj);	向服务器发送Json数据

主要使用成员函数	作用
void open(const QUrl &url)	使用给定的url打开一个WebSocket连接
void abort()	中止当前Socket并重置Socket
qint64 sendTextMessage(const QString &message)	通过Socket将给定数据作为文本消息发送并返回实际发送的字节数
qint64 sendBinaryMessage(const QByteArray &data)	通过Socket将给定数据作为二进制消息发送并返回实际发送的字节数
[signal] void connected()	成功建立连接时发送
[signal] void disconnected()	断开连接时发送
[signal] void textMessageReceived(const QString &message)	每当收到文本消息时都会发出此消息。该消息包含接收到的文本。

服务端websocket

服务器端接收来自QT客户端和网页客户端的连接，并向同一group中转发数据，
客户端创建一个room为，服务端为他分配一个空的room_id(25)，该用户可以向其他人分享该房间号，其他人可以通过房间号加入房间，协作画板。



总结与不足

本次团队开发共计10天，成果主要显示在demo中。
（链接：https://www.bilibili.com/video/BV1YP411c7dN）
不足：

1. 开发前期方向有误，导致前期开发效率低下，团队协作不明显。
2. 预计完成功能时间和实际开发使用时间矛盾，没有写好单元测试，导致debug时间过长，进度条缓慢。
3. Undo和Redo功能还没有完全实现（不能同步），之后会继续完善。