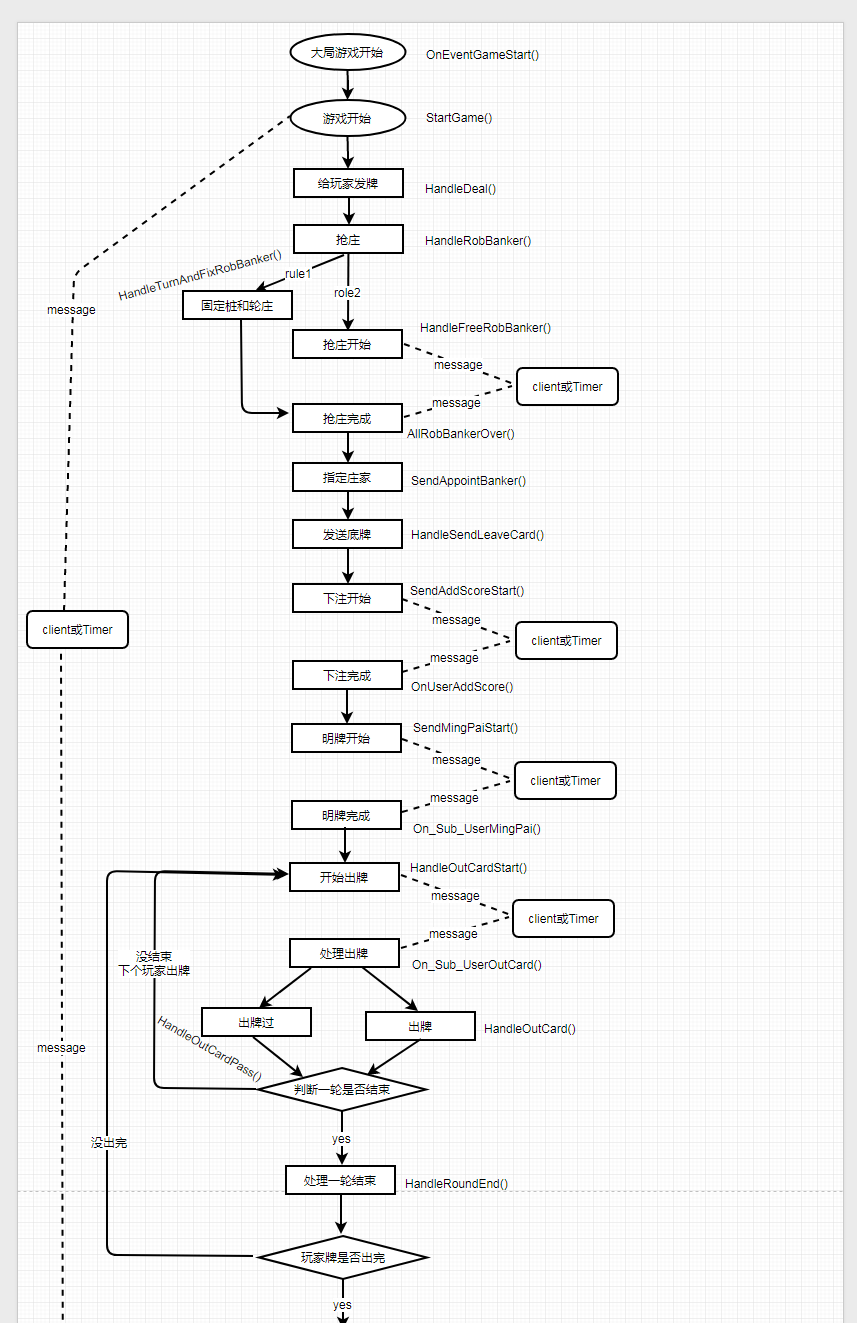
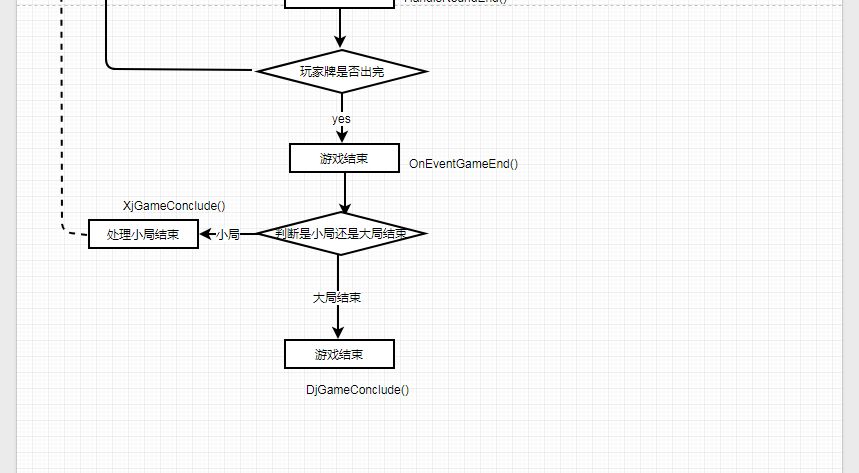
# 斗地主测试总结

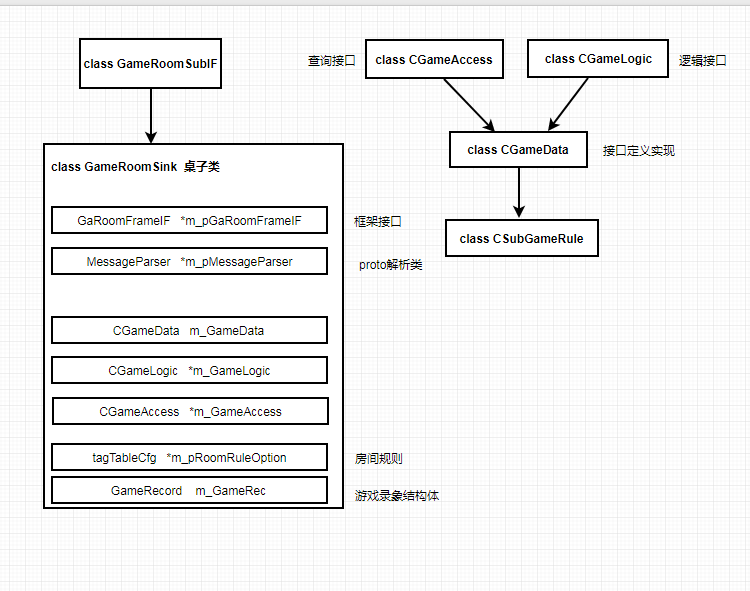
## 斗地主子游戏流程总结





## 桌子类框架（它是如何存储游戏逻辑所需的各种信息以及如何调用到大厅的游戏流程）

##### 各种类的关系



##### GameRoomSink类

游戏桌子类，游戏的主流程是在这里定义的，并且定义了从GameRoomSubIF继承的纯虚接口。

桌子类中分别定义了框架接口对象、proto解析类对象、房间规则类对象和接口类对象等；框架接口对象主要来调用在框架里定义的流程函数、消息的发送、设置定时器等功能；proto解析类对象主要是用来注册子消息号的，将C发往S的消息号和消息包关联起来；房间规则类对象是在初始化房间规则时调用的，记录通用规则和子游戏特有规则；接口类对象主要是记录查询我们所需的信息，调用游戏的逻辑函数等。

##### CGameData类和CGameLogic、CGameAccess

CGameLogic、CGameAccess是两纯虚接口类，申明在头文件Gameinterface.h中，CGameData类继承了这两类，并在CGameData类实现接口的定义。

接口主要用来查询存放游戏配置，游戏玩家配置，游戏卡牌配置，房间配置，棋牌配置等，以及子游戏的逻辑算法函数。

##### GameRoomSubIF类和GameRoomFrameIF类

GameRoomSubIF类是桌子类的基类、接口类，在frame的GameRoom中被调用，在SubGame的GameRoomSink中实现定义，主要功能是初始化GameRoomFrameIF对象和注册消息号，定时器和消息事件的处理，断线重连场景处理，设置规则等函数。

GameRoomFrameIF类是GameRoom的基类、接口类，在frame的GameRoom实现定义，在SubGame的GameRoomSink被调用，主要实现SubGame和frame之间发送大局小局结束的消息，以及发送消息包给客户端和设置删除定时器等功能。

##### CSubGameRule类

是cGameData的子类，存放了一些更新的函数，在创建游戏时new一个CSubGameRule类对象并分别赋值给基类CGameLogic、CGameAccess，实现多态。

## 子游戏流程的控制

##### 定时器

定时器主要实现了交互时间控制，当服务端给Client发送某事件开始的消息时，服务端会进入阻塞状态等待Client返回事件消息，我们可以在发送消息给Client和接收到Client消息之间设置一个定时器SetTimer，当我们设置的定时器时间结束了但Client并没有返回消息时，frame会调用SubGame中的定时器事件函数执行指定的定时器事件，比如斗地主加倍过程，如果玩家时间用尽还是没有点加倍或不加倍按钮，这时候就执行定时器事件，强制不加倍。

注意，我们删除定时器时，会一次性全部删除当前事件的定时器

##### 消息处理

当发生某事件需要消息传递时，服务端和Client之间会定义一个消息号并对应一个消息包，然后调用GameRoomFrameIF对象里的单播或广播消息的函数发包给Client，接受包时，Frame会调用Subgame里的消息处理函数，根据消息号执行对应的消息

## 游戏流程中需要记录那些信息

##### 游戏信息

最大/小椅子数、抢庄模式、得分模式、桌子类型等

##### 游戏状态信息

//房间状态

#define GAME\_STATUS\_FREE 0 //空闲状态

#define GAME\_STATUS\_PLAY 100 //游戏状态 TODONOW

##### 定时器持续时间

#define IDI\_TIME\_ROB\_BANKER 10000 //抢庄定时器持续时间等

##### 椅子上的用户状态

#define USER\_NULL 0 //用户状态-椅子空

#define USER\_STANDUP 1 //用户状态-站立

#define USER\_SITDOWN 2 //用户状态-坐下

#define USER\_ONREADY 3 //用户状态-准备

#define USER\_PLAYING 4 //用户状态-游戏中

##### 子游戏特有的定义

斗地主定义、排序类型，牌型等

##### 卡牌的结构和该结构的数目

//分析结构

struct tagAnalyseResult

{

BYTE cbFourCount; //四张数目

BYTE cbThreeCount; //三张数目

BYTE cbDoubleCount; //两张数目

BYTE cbSignedCount; //单张数目

BYTE cbFourCardData[60]; //四张扑克

BYTE cbThreeCardData[200]; //三张扑克

BYTE cbDoubleCardData[200]; //两张扑克

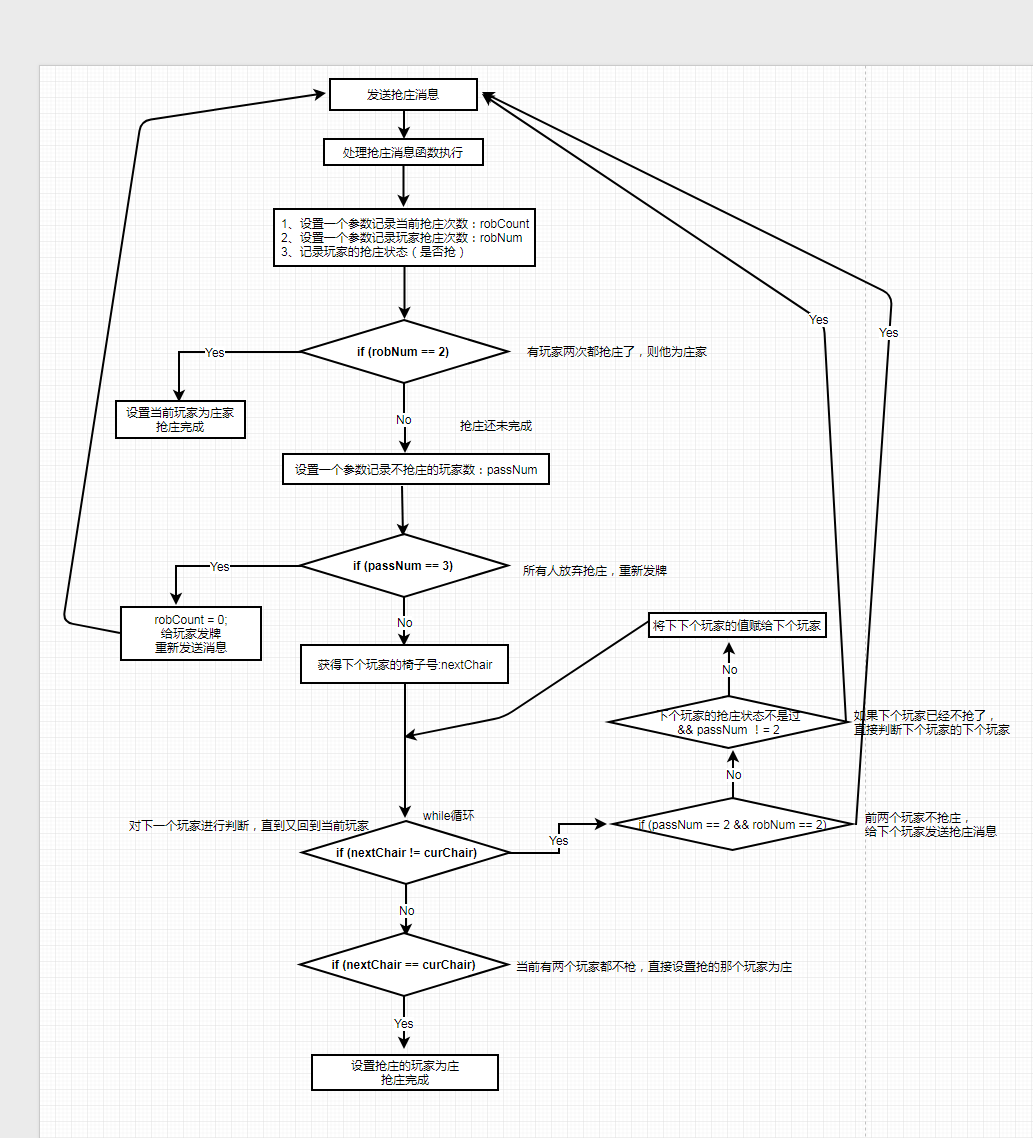
BYTE cbSignedCardData[200]; //单张扑克

bool cbLaiZi; //赖子类型

};

## 游戏流程中各种逻辑思路的总结

##### 判断抢庄各种类型的思路



##### 发送出牌消息时，用二进制显示按钮的思路

设置二进制的第一位为出牌按钮开关，第二位为要不起开关，发送消息前，判断一轮是否结束，结束发送1，没结束发送1<<1。

##### 处理用户出牌的思路

先用一个参数记录用户出牌的牌型，判断玩家是不是一轮中第一个出牌的，是第一个出牌判断牌型是否正确，正确继续执行；玩家不是第一个出牌的，将他出的牌进行校验（和上一个出牌玩家出的牌进行对比），校验成功继续往下执行；将玩家出的牌和删除玩家已经出的牌后的手牌记录在包里发给Client，最后判断该玩家出完牌后是不是没牌了，没牌了一轮结束，有牌则通知下一个玩家出牌。

##### 处理一轮结束的思路

判断此轮的赢家还有没有手牌，有牌则通知他继续出牌，没牌他就赢了。

## 游戏中二进制的使用和复杂算法思路的总结

#define CT\_MISSILE\_CARD 14 //火箭类型

(CardType )& ( 1<<CT\_MISSILE\_CARD)(牌型宏) != 0 说明CardType 中存在CT\_MISSILE\_CARD（王炸）牌型

原理：将0~14类牌型当成二进制中15个位，如果当前位为1，表示有位数所代表的的牌型 如0100 0000 0000 0000 表示 14（火箭类型）

<< 是位移运算符 1 << 14 是将二进制的1向左位移14位 结果就是 0100 0000 0000 0000

子游戏中CardType传来的就是二进制牌型 与需要判断的牌型做&（与运算）

结果不为零 说明牌型为火箭

源代码：

//类型判断

if (cbNextType==CT\_ERROR) return false;

if ((cbFirstType&(1 << CT\_MISSILE\_CARD)) != 0) return false;

if ((cbNextType&(1<<CT\_MISSILE\_CARD)) != 0) return true;

........