✓ 报文及进程间通信机理

Server ⟷ Client（进程间）

Pre-game messages 预处理报文

`200` **Connection Established**

当 Client 连接到 Server 之后:

- Server 向 Client 发送 `200` 信号，Client 接收后返回 `200` 信号，完成第一次通信
- Server 在收到 `200` 信号后，会生成棋盘内容，然后发送 `201` 信号，将棋盘数据传输给客户端

`201` **InitChessboard**

- 使用格式: `201 <Chessboard Data>`, `201 Received`
- 服务端使用 `201` 信号来传输数据，客户端收到 `201` 信号后返回 `201` 信号，此时两个进程同时进入 `WAIT_PLAY_CONFIRMATION` 状态

`202` **Client Ready**

- 两者都准备就绪是由服务端来判断的，因此客户端点击 `Start` 按钮后，向服务端发送报文 `202` 以表示自己准备完毕
- 此时服务端可以在客户端之前或之后完成准备过程

`203` **Game Start**

- 使用格式 `203 <Offensive>` ，其中 0 代表服务端先手，1 代表客户端先手
- 一旦 Server 自身完成准备，也接受到了 `202` 信号，便会生成一个介于 0 和 1 之间的随机数，然后向客户端发送该报文
- 服务端与客户端同时由 `WAIT_PLAY_CONFIRMATION` 状态转换为 `PLAYING_WAITING` 或 `PLAYING_THINKING` 状态，游戏开始

In-game messages 数据操作报文

`300` **Unreveal Chess Piece**

- 使用格式: `300 <position>`
- 将会把位于 position 位置的棋子翻开

`301` **Move Chess Piece**

- 使用格式: `301 <from> <to>`
- 将会把位于 from 位置的棋子移动到 to 位置

`302` **Eat Chess Piece**

- 使用格式：`302 <from> <to>`
- 将会把位于 `to` 位置的棋子标记为死亡，然后把 `from` 位置的棋子移动到 `to` 位置

`303` **Eat Chess Piece and Destroy Self**

- 使用格式：`303 <from> <to>`
- 将会把位于 `from, to` 位置的棋子标记为死亡

Switch-control messages 控制权交换报文

`400` **Switch Turn**

- 一个**回合**结束后的进程将发送 `400` 报文
- 接收到 `400` 报文的机器将会开始一个新的**回合**

`401` **Win Game**

- 使用格式：`401 <isMe>` 其中 `1` 代表自方胜利，`0` 代表对方胜利
- 使用后游戏转入 `End` 状态

Game ⟷ GameConnection（多线程间）

以上进程间传递的所有带有*的信号

`0 <Color> <Message>`

- 说明：设置主界面的说明文字
- 参数
  - `Color`：报文的颜色(六位十六进制数，或是HTML支持的颜色标签)，如`ff0000`
  - `Message`：说明文字.

✓ 类的设计

Game

以 `GameLogic` 类为友元.

- 棋盘 `board`：`Vector<ChessPiece*>`

ChesePiece

棋子的 InitID 与其兵种的对应关系如下。其中偶数为蓝方，奇数为红方。

InitID	兵种
1~2	军旗
3~4	司令
5~6	军长
7~10	师长
11~14	旅长
15~18	团长
19~22	营长
23~26	炸弹
27~32	连长
33~38	排长
39~44	工兵
45~50	地雷

✓ TODO

- 还有一个 GAME_STATUS == READY 的 cancel 逻辑没有写
- 断线的业务逻辑
- 通信
- 走游戏流程 计时器 判胜负