//本文档应该是个废物了

本文档将手动模拟用户使用软件的各种操作，并描述各系统之间详细的交互方式。

1. 用户双击打开jar文件包运行游戏。
2. 此时，软件从前端的main入口开始运行。
3. 前端生成一个新的staticplatform，并调用staticplatform的loaddefaultdata功能，加载默认存档，此功能会把staticplatform设置成存档的模样。至此，staticplatform读取完毕。
4. 前端用上一步获得的staticplatform生成一个staticscene，staticscene的功能是显示人物（人物模型，可以通过把人物肢体/武器贴图放在一个组件中，并把组件设置为透明来实现）外观（加入静态震颤以优化用户体验），完成staticscene的显示。（任务、成就等面板以stage的形式展现）。
5. 静态操作：staticscene负责各个按钮的监听，比如用户点击“任务”时，弹出任务stage（注意此stage要设置成置顶）等。
6. 各个子stage将会与后端相应的支持/拓展系统相对应。每个stage就是后端一个系统的映像，可以根据用户的操作（比如更换装备）来修改后端的staticscene（修改player1的属性//玩家永远是player1）.
7. Staticscene是全屏的，而子系统stage是窗口化的，可以应用统一的css风格样式。
8. 用户在staticscene的所有操作称为静态操作
9. 用户所有修改staticplayer（档案数据）数据的操作也称为静态操作。
10. 动态操作：用户选择某模式开始一次battle，battle中，所有不修改staticplayer（档案数据）的操作称为动态操作，即一场battle内部的操作称为动态操作，因为他只与一次battle有关，而在battle结束之后，这些操作不会留下任何痕迹。动态操作由dramaticscene（实际上是他的具体子类）与后端的相应battleplatform进行交互来完成。
11. 前端只需要轮询两个actionpo包（playeractionpo和aiactionpo）来实时显示人物动作，轮询一个poppo包，显示棋盘更新信息，并轮询battlepo包来检测比赛结束信号。同时需要将用户的移动指令传递给后端。
12. 后端需要接收到用户移动指令时，检测是否为合法消除，当前端要求进行消除时，进行消除，并且一次消除更新一次playeractionpo和poppo包。
13. 后端的AI有自己的线程来直接要求平台修改人物属性，并更新aiactionpo。
14. 后端需要轮询更新battlepo包
15. 在一次battle结束之时，由后端负责进行静态更新。前端负责显示各系统的更新信息，通过stage的形式。