**《Manila Online》概要设计说明书**

《软件工程原理》第二小组

姚照原、戴琪、郭浩、金恬、张溶倩

1. **软件开发任务**
   1. 开发目标

遵循桌游的整体流程和规则，将桌游马尼拉在PC端实现，并配以美观友好的界面。玩家可以选择单人模式与机器算法对战，也可以选择多人模式与网络上的其他玩家联网对战。开发期限为本学期末。

* 1. 运行环境

客户端硬件环境：客户端要求x86或x86-64处理器，性能高于Intel Core i5-5200U,需要128MB或以上可用内存和150M可用外存；软件环境：程序面向Windows系统开发，客户端需要安装Python3.x及必要的第三方库

服务器硬件环境：基于KVM搭建，CPU: QEMU Virtual CPU version (cpu64-rhel6)，RAM: 520MB，DISK: 10G SSD。软件环境：基于CentOS release 6.8，除此之外还需要安装MySQL。

* 1. 需求概述

功能：1）实现游戏流程的控制，包括按时向玩家发出信息、获取玩家决策；维护当前游戏状况（个人财产，随从和船的位置，货物股价）；产生随机数推动船前进；自动判定胜负。2）实现对局的管理和信息在网络中的传输（两位或以上人类玩家时）。3）实现能够模拟人类做出决策的机器算法。4）实现用户信息的管理。

性能：精度方面，游戏中只用到整数，不涉及浮点误差；时间方面，要求程序在毫秒级时间内响应、传输数据、更新显示；机器算法在秒级时间内作出反应。

此外，要求程序能通过图形用户界面与玩家交互；管理玩家信息和历史对局数据；处理网络故障、游戏中的不合法操作等错误。

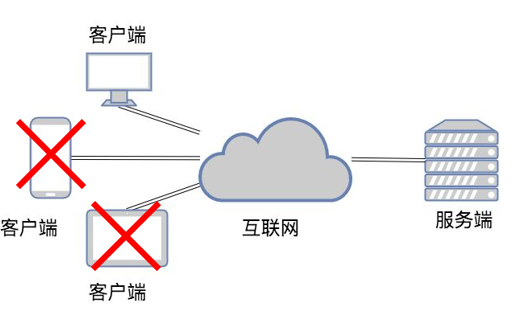
* 1. 程序限制

同时处理的并行用户数为百级，不能超过服务器的处理能力。

服务器端存储空间有限，能够保存的复盘数据有上限，或者需要超级管理员每隔一定时间将数据从远端进行备份。

1. **总体设计**
   1. 总体结构

本程序采用C-S框架，即分为服务器端和用户端。服务器端主要负责安排对局以及玩家决策信息在网络间的传输，其余流程控制、游戏状态的更新和显示均在用户端完成。服务器端和用户端各有一个基于TCP/IP协议的通信模块，用于消息的传输。



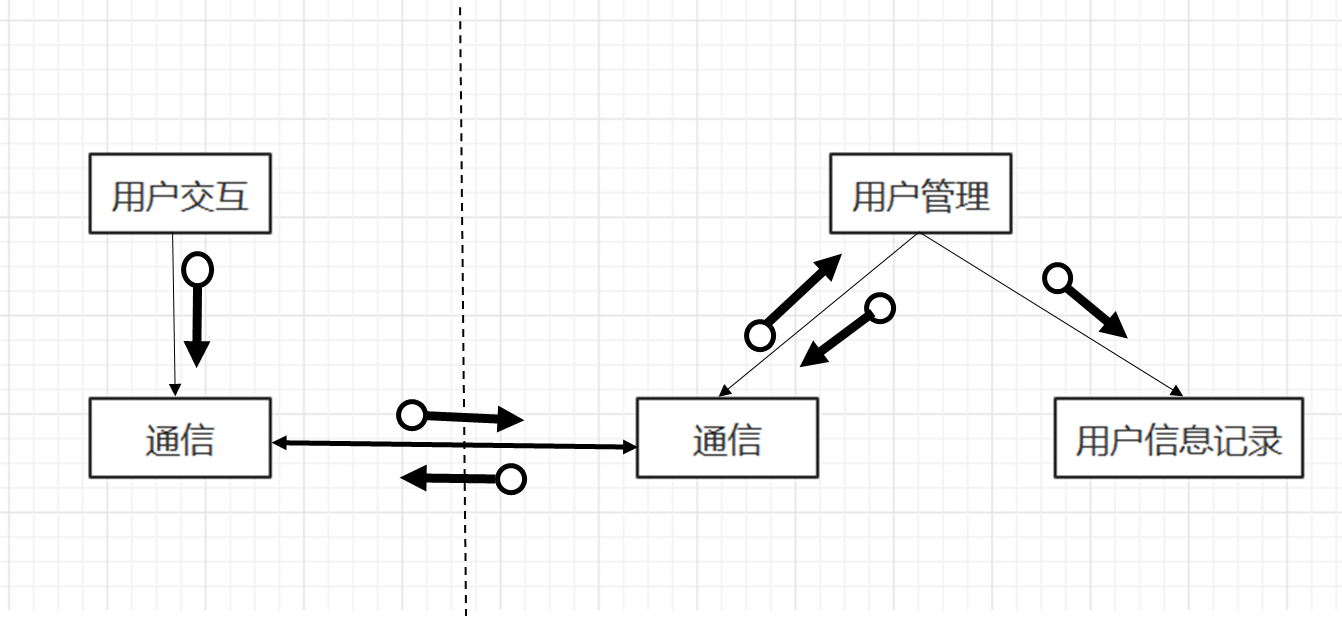
* 1. 用户端模块

用户端程序按功能分为四个基本模块：**流程控制**，用于按规则控制游戏进程；**用户界面**，用于可视化显示当前局面，接收用户的决策信息；**机器算法**，用于参与游戏，有决策能力；**消息同步**，用于处理、解读服务器发来的其它玩家的决策信息。用户界面、机器算法模块产生决策信息后，一方面将结果发送给通信模块，由服务器向其它玩家广播；另一方面将结果返回给流程控制模块。消息同步模块从通信模块接收信息，解读后返回给流程控制模块。

* 1. 服务器端模块

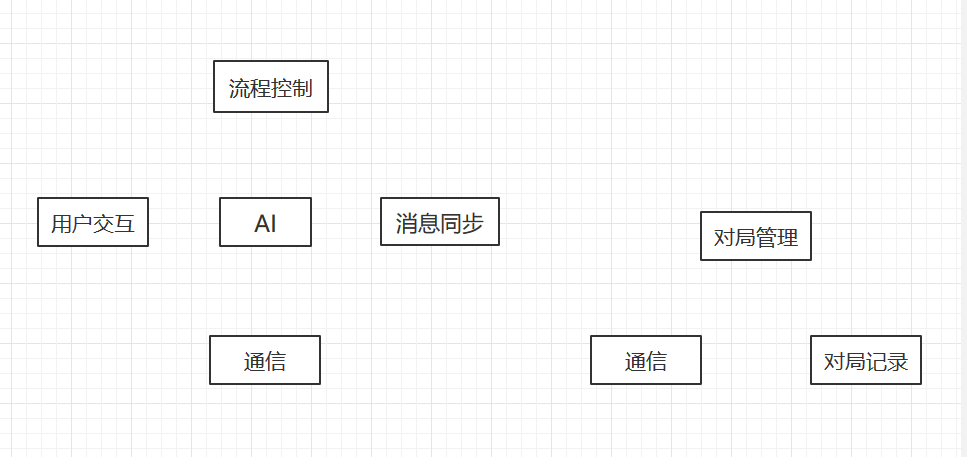
服务器端按功能分为两个模块：

A: **数据存储**模块，基于关系型数据库管理，包括4张表单：用户信息表，荣誉榜单，历史战绩表，全网排名表；

B: **对盘管理**模块：基于TCP协议建立长连接，实现P2P的即时传输，用于管理对局，主要功能为：匹配对手（包括人机），发送和接收用户决策信息，记录对局过程和结果，判定游戏结束与否；每盘对局都包括至少一个至多4个真人玩家。对盘管理模块有两个子模块：数据通信子模块和对战记录子模块，分别负责客户端与服务器间的数据通信和本局游戏的对战结果记录。

管理用户时的系统结构图

游戏进行时的系统结构图



* 1. 处理流程

玩家打开游戏后首先选择ID登录或创建新ID（本机创建过的ID及对应的历史记录、成就信息在本地保存，所有ID在服务器端保存）。之后主界面有以下选项：单人游戏、多人游戏、历史记录、成就、设置（后两项不在基础功能范围内）。

选择 “单人游戏”，本地程序直接随机选择三个机器算法与之开局，所有控制、运算、维护在本地进行。（游戏流程与各种事件处理已在需求文档中详细叙述，这里不再赘述，下同）

选择“多人游戏”，则向服务器发送请求，服务器端持续更新当前等待开局的用户数，满4人则组成一局游戏。如果2min内仍不满4人，则用机器算法补齐空位开局。机器算法在其中一位游戏者机器上运行。运行过程中，每个用户端各自维护游戏进程，服务器只负责消息的传递。

游戏结束后，计算并展示游戏排名和技术统计（非基础功能），将游戏时间、结果等数据存入历史记录文件，经用户确认后回到主界面。

选择“历史记录”，则打开本地存储历史记录的文件，显示该ID历次游戏的时间、排名情况。

1. **数据结构**

3.1 棋盘类

棋盘类（Class Board）有唯一的成员ManilaBoard，用于维护当前游戏状态，推进游戏进程。其**属性成员**包括股票价格和剩余数目，各随从位置的状态（可用，被占据或不可用），各船的位置，各位玩家的金钱、股票类型及数量、竞价信息、骰子点数。这些信息是公共成员，所有玩家可以访问。除此之外，四个玩家（Class Player）作为成员变量，提供决策函数接口，供流程控制程序调用。**函数成员**包括game\_process()，用于流程控制。

3.2 玩家类

玩家类（Class Player）为抽象类，有三类子类：当前玩家（PlayerCur），用于从用户界面接收决策信息；在线玩家（PlayerOL），用于从网络中接收和解读决策信息；在本机运行的机器算法玩家（PlayerAI），直接根据一定规则运算，反馈决策信息。它们共同的作用是被game\_process()调用，实现玩家的功能。具体而言，Player类需要具备处理以下事件的决策函数：竞价选总督；总督行使权力（选择货物，安排工作，买股票）；放置随从；海盗登船；领航员操纵。

3.3 历史记录

历史记录为一个结构体，包括游戏时间，轮数、最终股价、现金、总财富、排名、与冠军的分差等。

3.4 通讯数据

通讯数据为规定格式的字符串：<开始标记> <操作时间> <玩家序号> <操作代码> <操作信息><结束标记>

其中开始标记为$$；操作时间为时：分：秒格式；操作代码为整型，根据操作代码表确定；玩家序号为1-4的整数；操作信息为操作的参数（如放置随从的位置代码）；结束标记为/$$

3.5 服务器端数据表

服务器端使用关系型数据库，主要有四张表：

a.用户信息表：用户ID，用户昵称，用户验证邮箱，用户登录密码，用户成就；

b.荣誉榜单：荣誉编号，荣誉名称；

c.历史战绩表：用户ID，对局者ID\*3，最终得分（即每局均有四条对战结果记录）；

d.全网排名表：用户ID，用户得分；

1. **模块设计**
   1. 流程控制模块

包括Board类，主要体现在其流程控制函数game\_process()。每局游戏开始时，创建Board类并为其Player成员对象赋值，通过这一机制与用户接口、机器算法、消息同步三个模块对接，是软件的核心模块。

当所有玩家的决策信息被完整传输和正确解译时，不同终端上的棋盘状态能够保持一致。只需注意投骰子的环节，不能分别产生随机数。我们约定总督玩家的用户端产生随机数，服务器负责传输给其它终端。

* 1. 用户接口模块

这一模块包括Player的当前玩家子类PlayerCur，以及若干窗体的设计与响应代码。

用户端软件的界面主要包括主界面、游戏界面、历史数据界面、技术统计界面、成就界面（后两者非基本功能）。

主界面展示游戏名称及背景图，显示单人游戏、多人游戏、历史记录、成就、设置六个按键，并提供ID输入框用于注册或登录。注册/登录成功后按键被激活。

游戏界面以游戏棋盘为主，还包括骰子区、玩家面板、股价表、状态栏等控件。**游戏棋盘**上有船只，船只可移动；船上和岸边的岗位均可点选用于放置随从。已有随从的位置不再能点选，显示对应玩家的颜色。**骰子区**显示每次随机产生的骰子点数。**玩家面板**共4个，其中3个为对方玩家，显示当前现金、股票及其类型、总财富；1个为我方玩家，除以上信息外增加输入框（用于输入竞选总督的报价）、确认按钮、抵押股票按钮。**股价表**显示四种商品当前股价，程序自动维护。**状态栏**显示提示信息，如“3号玩家XXX正在选择随从放置位置”，“请输入竞选总督报价，退出请输入-1”等。除此之外，领航员、海盗、总督的操作需要单独的窗口来进行交互。

历史数据界面显示一个表格，包含游戏时间、总财富、名次等项。

这部分与用户的交互主要通过键盘和鼠标实现。

* 1. 机器算法模块

该模块为PlayerAI型的若干子类，具有非常强的独立性，任何基于Board公有数据成员和接口的实现均可添加到软件中。每一种实现算法为一个子类，使用时将其实例化。

对于该模块的设计，目前的几个思路是：1）人类玩家的“感觉”模拟，如倾向于在最靠前的船放随从等；2）稳健投资，选择风险小的有利可图的位置，从不做海盗；3）随机决策，按随机数行动，根据局面调整各种选择的概率；4）数学期望计算，总选择数学期望最大的位置；5）机器学习，采用“局面-选择”类型的历史数据进行训练（非基础功能）。

* 1. 对盘管理模块

该模块主要包括ManilaProcess类，成员包括四个对局方的ID和IP地址与连接状态，方法有监听消息和发送消息。分发数据使用Socket库实现长连接，负责将接受自一个客户端的决策信息打包分发至其他的用户，如果超过每步决策的默认时长（假设为30秒），自动向所有用户广播“当前用户弃权”消息；若超过30秒没有接受到客户端发送的探测报文，则广播当前用户掉线的消息。报文的结构仍需进一步定义，报文至少应包括决策者ID，决策者操作编号（可以预制操作表），决策时间。

1. **异常处理及其它**
   1. 异常处理

决策信息错误：在选项不可用时使其无法点选；在处理输入信息时加入断言、条件判断或try-catch机制以发现错误；在机器算法中设置异常值的合法化语句（返回默认合法值）。

通信错误：通信成功以后会返回固定的accept报文来确认，否则认为是数据丢失，记录当前出错的位置，用于恢复连接后的数据同步；对于非法的消息，忽略掉，不做任何理会。客户端每隔固定长时间（如30秒）向服务器发送探测报文检测通讯是否畅通；服务器负责将接受自一个客户端的决策信息打包分发至其他的用户，如果超过每步决策的默认时长（假设为30秒），自动向所有用户广播“当前用户弃权”消息；若超过30秒没有接受到客户端发送的探测报文，则广播当前用户掉线的消息，并断开与此用户的连接。

* 1. 安全性与维护

考虑到消息加密成本太高，客户端与服务器的通信采取明文通信的方式（也考虑增加简单的加密方式，如移位密码），对战数据在客户端和服务器均有备份，通过数据库统一管理。维护主要是通过超级服务员在远端拷贝对战数据文件和荣誉榜单实现，服务器定期清除这些对战数据（仅保留七天）。

* 1. 扩展功能

如果开发顺利，可能实现的扩展功能包括：

1. 编写机器学习型AI算法
2. 对局的技术统计及显示（如每人选取位置的频次，各回合的收益等）
3. 成就系统（设置成就界面，完成的成就将被点亮；在游戏中监视成就事件的发生，若发生，在对局结束时提示）
4. 设置（扩展游戏，可以选择新的货物类型和股价上限）

起草人：姚照原 郭浩

2018年12月4日创建

2018年12月5日第一次修改

2018年12月7日补充图片等