**详细设计文档**

**Manila 1.0**

流程控制&机器算法

1. **概述**

为了避免并行设计和开发中的不一致问题，我们拟采用串行设计、开发的方法，分三步完成全部的开发。Manila 1.0包括流程控制和朴素机器算法的实现，流程控制模块是客户端的核心模块，因而最先完成；机器算法的编写可以有效测试流程控制模块的性能。因此，Manila 1.0在整个项目中具有基础性地位。

Manila1.0的完成标准是可实现四个机器算法在同一台机器上对战。

1. **流程控制模块**

2.1程序描述

将桌游Manila的规则实现为代码，并可以通过调用Player类的实例获取游戏决策。具体功能包括产生骰子值，维护棋盘状态、玩家资产等公共数据，推动游戏进行直至结束。程序会以日志的方式记录游戏中的所有事件，终局时输出到文件中。

2.2性能

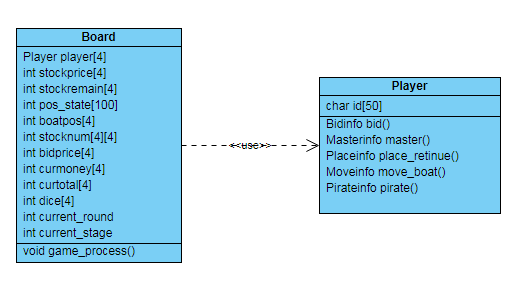
精度：处理的数据都为整型，计算方面没有精度问题。

响应时间：不涉及复杂计算，可以忽略不计。

灵活性：程序严格按照游戏流程顺序执行。下级模块会保证操作的合法性，最终版本中本模块对操作合法性不做排查，但在测试阶段，采用try-catch机制捕捉异常，防止机器算法出现错误。

2.4接口

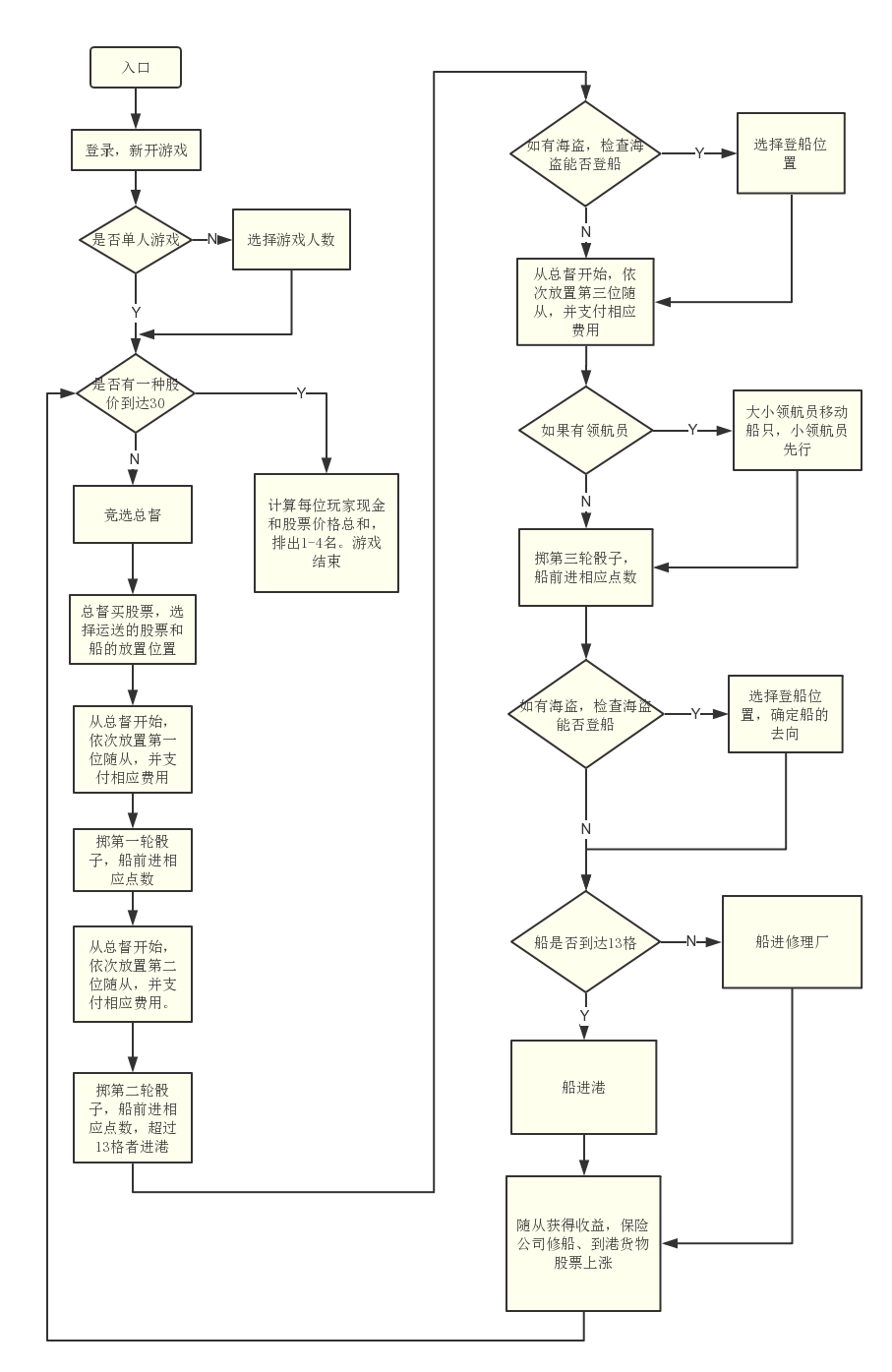
本模块通过Player类与机器算法、消息同步、用户界面三个模块交互，具体方法为以无参方式调用Player实例的以下决策函数：竞价-bid(), 总督行使职权-master(), 放置随从-place\_retinue(), 海盗登船-pirate(), 领航员操纵船只-move\_boat()。函数的返回值、调用时机由下面具体说明。

本模块包括对Player类开放的公共数据区，即Board类的属性成员。

Board类与Player类成员一览

Player类五种info结构体包含当前步骤所有决策信息。具体来说，Bidinfo包括竞选报价（整型，下同；-1代表退出竞选）和抵押股票的数目；Masterinfo包括四种货物所在船的起点（0-5之间，有一种为-1，代表不装船），购买股票的类型，抵押股票的数目；Placeinfo包括防置随从的位置（可为空位置，与位置表对应），抵押股票的数目；Moveinfo包括四种货物所在船的移动格数（-2到+2）；Pirateinfo包括两个整型变量，第一个返回值表示选择登船的位置；在第二轮，不使用第二个返回值；在第三轮，第二个返回值表示船的去向。

Player的一些函数事实上需要传入参数（上图中略去）。Bid有一个参数表示上一未退出的玩家报价；move\_boat有参数1或2（分别代表小领航、大领航）；pirate有三个参数，第一个为身份（海盗船长为1，海盗船员为2）；对于身份为海盗船员的情形，第二轮中第二个参数为海盗船长的登船位置，第三个参数不使用；第三轮中第二个参数为海盗船长选择劫掠的船，第三个参数为海盗船长指定的船的去向。

2.5流程逻辑

流程控制模块流程图（摘自需求文档）

对每一回合游戏流程中的系列事件，归纳处理方法如下：

|  |  |  |
| --- | --- | --- |
| Stage值 | 事件名称 | 处理逻辑 |
| 0 | 竞价 | 从1号玩家开始，依次调用player成员的bid函数，直到三个player返回-1，保存总督成员的序号，之后放置随从从总督开始。  约束：底价为10，报价不超过手中现金 |
| 0 | 总督行使职权 | 调用总督成员的master函数  约束：四条船的起点中一个返回-1，其余在0-5之间且和为9. 所购买股票价格不超过手中现金。 |
| 1 | 放置随从 | 循环调用place\_retinue函数一次  约束：空位（pos\_state为-1）才可以放置，放置费用不超过手中现金。 |
| 1 | 掷第一轮骰子 | 为dice数组产生随机数 |
| 2 | 放置随从 | 同第一轮放置随从 |
| 2 | 掷第二轮骰子 | 为dice数组产生随机数，超过13格的船进港  例外：及其罕见的情形之下，第二轮骰子后三只船全部超过13格（如起始都为3，两次骰子都为6）进港，则跳过第三轮骰子，直接进入本轮结算。） |
| 2 | 海盗登船 | 若有船位置恰好为13且有海盗，调用对应player的pirate函数（先调用海盗船长，后调用船员）  约束：船员不能选择海盗船长登船的位置 |
| 3 | 放置随从 | 注意有船到达的空位也不能再放置随从，其余同第二轮放置随从 |
| 3 | 领航员操纵船只 | 若有领航员，调用对应player的move\_boat函数（先调用小领航，后调用大领航）  约束：小领航返回值移动绝对值之和为1，大领航为2 |
| 3 | 掷第三轮骰子 | 为dice数组产生随机数，超过13格的船进港 |
| 3 | 海盗登船 | 若有船位置恰好为13且有海盗，调用对应player的pirate函数（先调用海盗船长，后调用船员）；根据返回值确定船的去向 |
| 3 | 结算 | 除被海盗劫掠的船外，达到13格的船进港，否则进修理厂（建立中间变量记录船的去向）；向player分配随从收益，保险公司赔付（修改curmoney）；修改股价值（stockprice）；判断游戏是否结束。  注：若保险公司赔不起修理费，现金扣至零即可。 |

1. **机器算法模块**

3.1程序描述

用于参与单人模式（或多人模式玩家不足的情形）的游戏。在程序中的实体为Player的派生类，需要时将其实例化。其核心是决策函数中的算法。这一模块将会有多种实现，每一种实现可作为独立的组件加入程序，只要其符合流程控制模块定义的接口。

3.2性能

响应时间：运算的时间视算法的复杂性而定，朴素算法（不涉及机器学习）可在毫秒级出结果，实际游戏时可人为设定延迟，以带来真实的游戏体验。机器学习算法的响应时间为秒级。

精度：通过访问公共数据区中的棋盘当前状态，保证所作决策的合法性。

3.4算法

3.4.1 朴素经验算法（Mr.Naive）

|  |  |
| --- | --- |
| 函数 | 决策思路 |
| Bid() | 对总督的支付意愿定义为20与（手中现金减去20）的较小值，高于该支付意愿时退出竞价。 |
| Master() | 选择自己拥有股票的货物装船，不足三种时随机选择填满。拥有一种股票时位置设为（5,2,2），两种设为（5,4,0），三种时根据股票数量分配（如三种各一张，设为（3,3,3）；有一种为2张时设为（5,2,2））  注：若有四种股票（少见），选择拥有张数较多的三种装船。 |
| Place\_retinue() | 第一轮：选择有空位的船只中最靠前的一只放置随从  第二轮：以每回合骰子点数3.5估算船能否到达，对船上、港口、修理厂的所有空位，根据船去向的估算结果计算收益，取收益与成本之差最大的一个。  第三轮：假定位置在10及以上的船可以到达，其它船不能到达，计算所有位置的收益，取收益与成本之差最大的一个。（注：领航员的收益为可以救回的有自己随从的船只的收益；海盗的收益为处于7到12（含7和12）之间船只货价之和的六分之一） |
| Move\_boat() | I．选择位置在7至10之间（含7和10）且有自己随从的船，有多只时选择到港收益较大的一只向前移动。  II.当I不能实现，选择位置在7至10之间且没有自己随从的船只向后移动。  III.当II不能实现，选择位置11至13之间且有自己随从的船，有多只时选择到港收益较大的一只向前移动。  IV.当III不能实现，选择位置11至13之间且没有自己随从的船向后移动。 |
| Pirate() | 若在第二轮，选择收益较大的船登船，替代船上（除海盗船长外）当前总资产最高玩家的随从。  若在第三轮，选择收益较大的船劫掠，并让船开到港口。 |

* + 1. 随机决策算法（Random）

为了避免决策的恒定性，放置随从的决策改为：根据局面调整选择各位置的概率，然后由随机数产生决策。

* + 1. 精确期望算法（Expector）

枚举掷骰子的所有结果可以求出各船到港的准确概率，据此可求出船上、码头、修理厂、保险公司、海盗等位置收益的准确期望，为放置随从提供更可靠的依据。

* + 1. 进取与稳健切换算法（Weatherlight）

在精确计算期望和方差的基础上，根据自己的资产排名在稳健、进取两种模式间切换，稳健型以作为决策依据，进取型以型决策依据。

（后三种算法的细节待完善）

3.5接口

每一个机器算法参与游戏时被Board类的game\_process函数调用。决策时访问Board类的公共数据区。具体机制参见流程控制模块的接口描述。

3.6其它设想

1.把其它玩家的即时总资产加入决策，在自己发展的同时遏制领先玩家获利。

2.实现机器学习算法，采用“局面-选择”类型的历史数据进行训练。

1. 测试计划

在同一台机器上运行流程控制程序，调用四个机器算法进行对战，通过处理运行中异常、检查日志文件调试这两个模块。

（本部分起草人：郭浩）

2018年12月11日