华东师范大学软件学院实验报告

实验课程：OOAD　　**年级：**2017 级软件方向　　**实验成绩：**

实验名称：00 分析与实现　　**姓名/学号/手机：**

实验日期：2019.9　　**彭钧涛/10164601140/13918300751**

实验编号：No.1　　**郭源杰/10164800305/18916837257**

李尚真/10161900218/13761841606

一、实验目的

1/ 练习概念模型的设计过程

2/ 初步掌握 OOAD 的方法，通过系统的实现，体会 OOAD 的思想

二、实验内容与实验步骤

问题：

有一根300 厘米的细木杆，在第30 厘米、80 厘米、110 厘米、160 厘米、250厘米这五个位置上各有一只蚂蚁。木杆很细，不能同时通过两只蚂蚁。开始时，蚂蚁的头朝左还是朝右是任意的，它们只会朝前走或调头，但不会后退。当任意两只蚂蚁碰头时，两只蚂蚁会同时调头朝相反方向走。假设蚂蚁们每秒钟可以走5 厘米的距离。

请编写一个程序，计算各种可能情形下所有蚂蚁都离开木杆的最小时间和最大时间。

【约束：蚂蚁是有区别的，比如编号、名字】

要求：利用 UML 工具，2~3 个人一组（同一组交一份报告即可）

1/ 对问题进行细致的分析，描述分析过程。

2/ 给出完整的用例设计：参与者、用例图、用例描述，非功能性约束（如果有的话）

3/ 给出概念模型、设计类图，系统交互图

4/ 用 C++或 Java 实现，并且需要一个用户界面（UI），把界面及问题的答案复制到实验报告中。把源代码整体打包通过 email 交给老师，实验报告电子版交给老师。

5/ （选项）用图形模拟出蚂蚁们爬行的全过程。

三、实验环境

1. 运行系统： Windows 10/ Ubuntu 18
2. UML 工具：Enterprise Architect
3. 开发环境：JAVA

四、实验过程与分析

（1）问题分析

已知条件：木杆的长度，5 只蚂蚁的位置，蚂蚁的速度，蚂蚁移动的规则
求解的问题：所有蚂蚁都离开木杆的最小时间和最大时间
待解决的游戏机制问题：

1. 判断蚂蚁是否爬出木杆
2. 判断两只蚂蚁是否碰头
3. 判断游戏是否结束
4. 如何模拟 5 名蚂蚁起始方向的各种情况

1.判断蚂蚁是否爬出木杆

对于蚂蚁爬出木杆，存在两种情况，从木杆左边或者右边爬出，因此需要知道知道蚂蚁的位置，以及木杆的长度。对于一只蚂蚁，当它的位置小于木杆最左边，或者大于木杆最右边时，我们认为蚂蚁爬出了木杆。

需求：蚂蚁的位置，木杆的长度

2.判断蚂蚁是否碰头

蚂蚁每时每刻都在运动着，因此对于游戏的每一个 Step，都需要判断蚂蚁是否满足碰头的条件，因为本题的数据正好满足距离是蚂蚁速度的整数倍，因此可以通过两只蚂蚁的位置是否相等判断蚂蚁是否碰头

需求：蚂蚁的位置

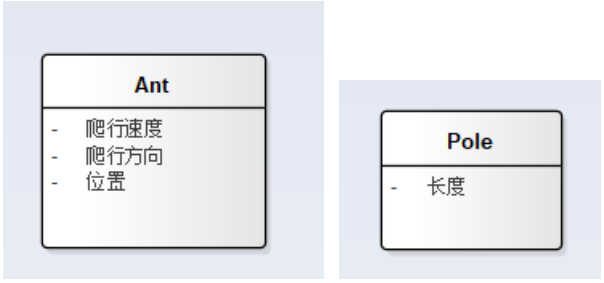
3.判断游戏是否结束

根据题意，当所有蚂蚁都离开木杆时游戏结束，因为当蚂蚁离开木杆时，不会再继续运动，并且不需要再判断是否爬出木杆，是否碰头等，因此我们可以认为此时蚂蚁“死亡”。我们可以给蚂蚁一个属性 isAlive 来标记蚂蚁是否死亡，一个属性 aliveNumber 来标记存活蚂蚁的数量，因此当所有蚂蚁死亡时（存活蚂蚁的数量为 0 时），游戏结束。

需求：蚂蚁的属性 isAlive，蚂蚁的存活数量

名词分析法找概念

蚂蚁，木杆
可以直接发现两个对象以及他们应该具有的属性：



整个问题作为一个游戏，蚂蚁在其中扮演角色，游戏开始，并且随着时间与规则前进，

当满足一定的规则时游戏结束，因此整个游戏可以作为一个类，用来管理游戏进程，制定游戏机制

应该具有的属性：游戏运行时间、游戏是否结束

4. 如何模拟 5 名蚂蚁起始方向的各种情况

对于蚂蚁爬杆的游戏，5 名蚂蚁起始方向是未知的，因此排列组合 5 只蚂蚁的不同起始方向，存在 2^5 种情况，对于每只蚂蚁的速度以及位置，可能也会有不同的初始值，因此设立一个新的类来配置游戏的初始条件

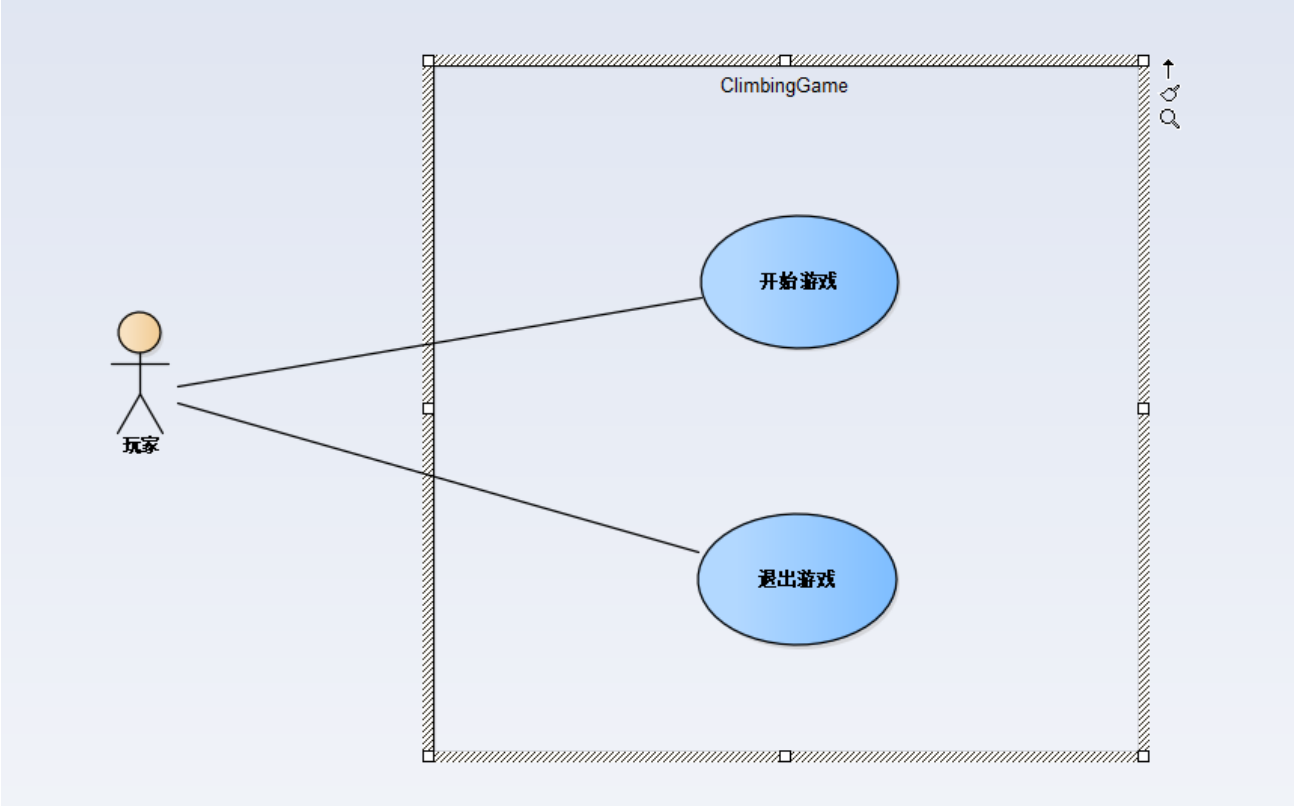
应该具有的属性：5 只蚂蚁的起始方向，蚂蚁的速度，蚂蚁初始位置

（2）用例设计

整个问题看作是一个游戏系统，因此参与者为外部启动游戏的玩家，玩家开始游戏后，游戏运行，输出结果。

参与者：玩家

用例图：

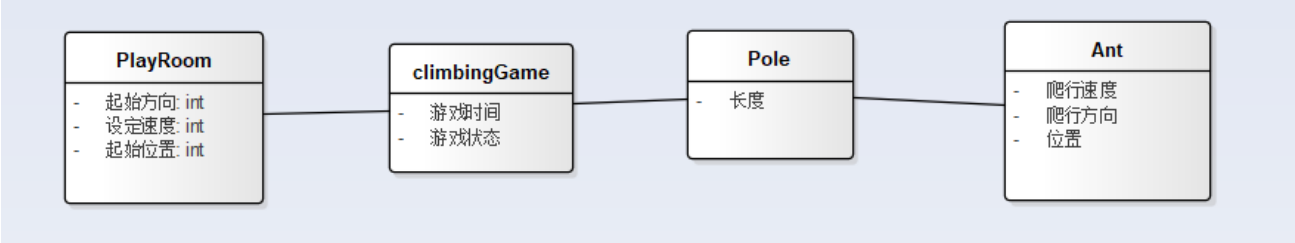


用例描述

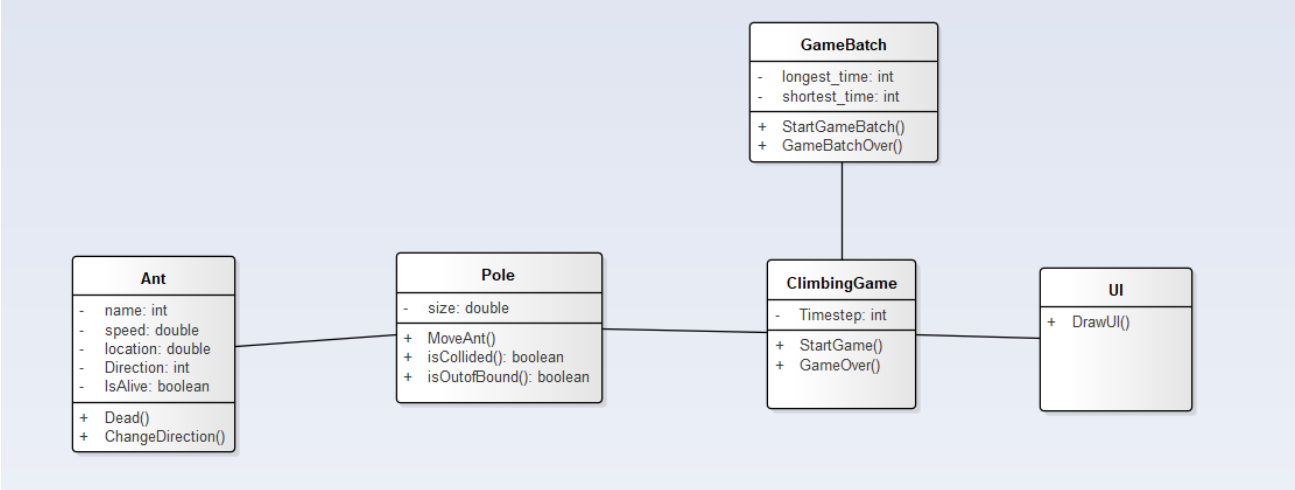
用例名称：开始游戏
用例描述：玩家开始游戏
参与者：玩家
基本事件流：
1.玩家开始游戏，系统开始运行游戏
2.游戏结束，系统显示游戏结果
3.用例结束

用例名称：退出游戏
用例描述：退出游戏
参与者：玩家
基本事件流：
1.玩家退出游戏，系统关闭游戏
2. 用例结束

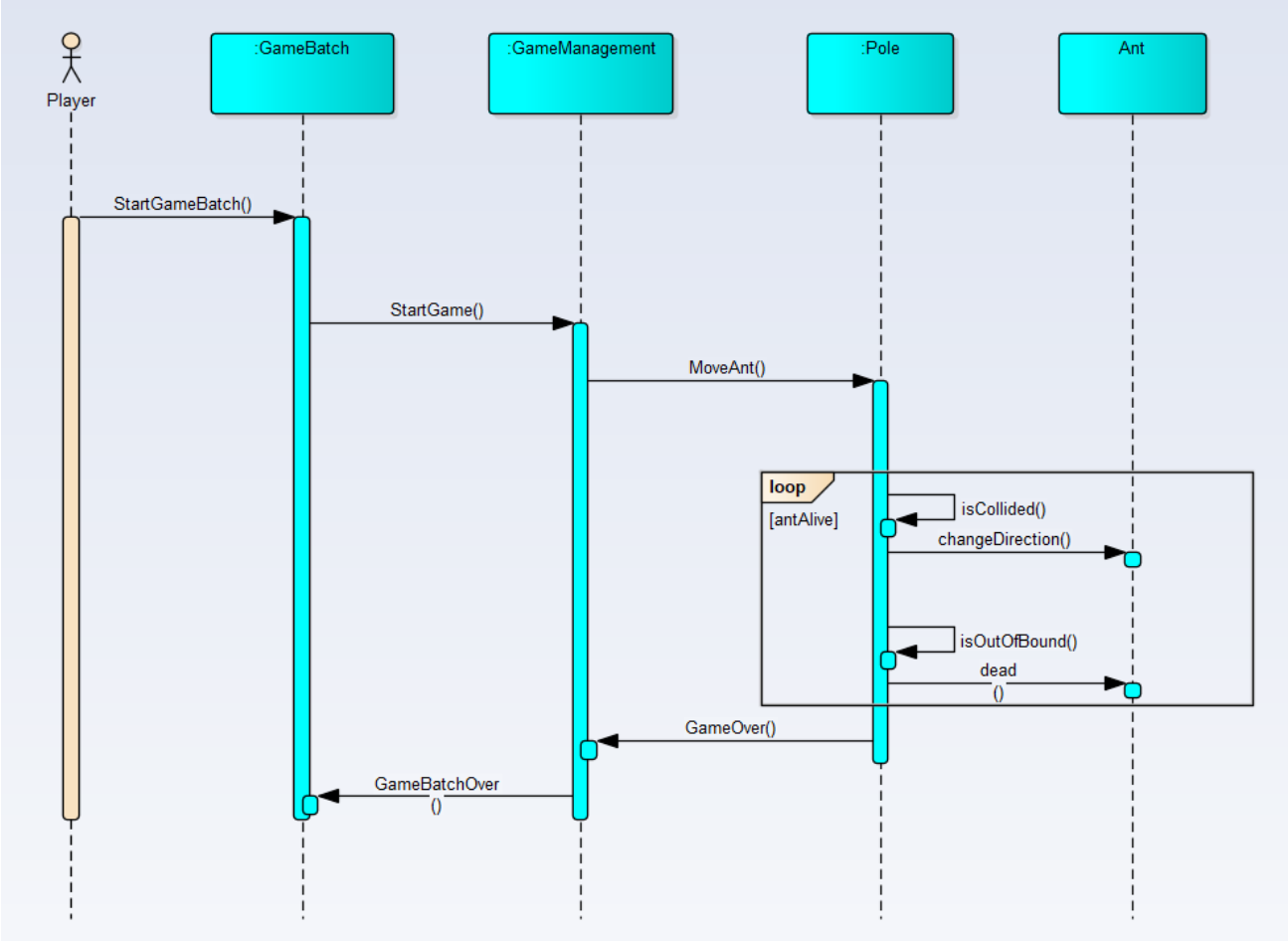
概念模型



设计类图

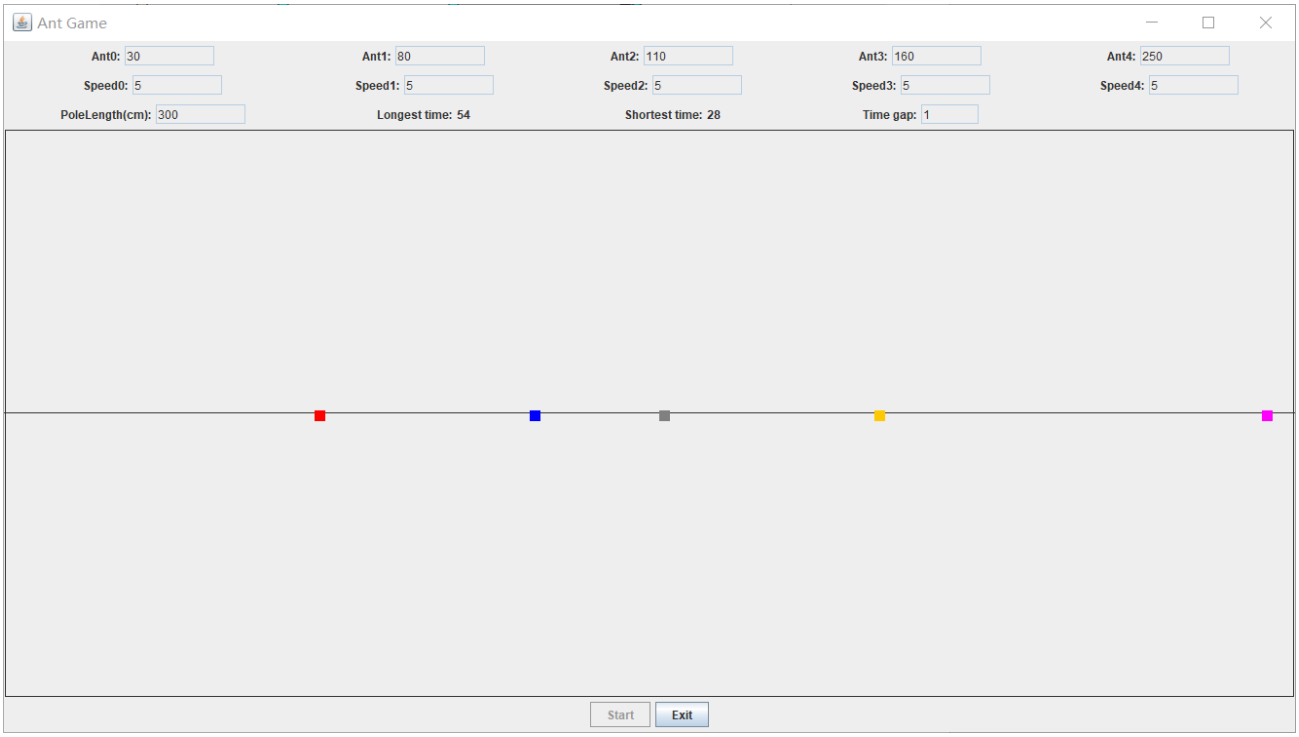


顺序图



（4）UI 界面内容及问题答案

UI 界面内容



问题答案

（5）图形模拟蚂蚁爬行的全过程
见附录中的录屏

五、实验结果总结

设计中碰到的问题：

1) 计算机无法模拟连续的时间流，而运用离散的时间计数会导致某些情况下蚂蚁相互错过，而不是碰撞回头。
2) 图形界面与后端游戏逻辑的同步显示问题
3) 图形界面刷新速度与后端游戏运行速度问题

解决的办法

1) 因为时间在每次循环需要自增一个固定值，所以将时间改为浮点数不可行。我们转而将木杆的长度和蚂蚁的速度更改为浮点型。并且判断当两个蚂蚁的位置小于一个值的时候判定为碰撞。这会引出另一个问题，即如何确定这一个值。我们选择的方法是获取两个蚂蚁的速度和，如果在下一个时间片过后他们的移动距离（速度和*时间片）大于他们实际距离，则判定为碰撞。这样做的缺陷是由于浮点数运算的误差和提前判定碰撞的误差，会导致最终结果与理想的数学计算结果有所不同。但是经过分析发现，由于我们的时间片取得很小（20 纳秒-50 纳秒中的一个固定值），这个误差与最终结果不在一个数量级上，即可以忽略不计。
2) 用例中，当用户按下图形界面的 START 按钮，游戏会启动。但是实际设计中，如果在 START 按钮的 ActionListener 初始化游戏并且开始游戏，会导致 ActionListener 在后台游戏结束前不会返回，间接导致了图形界面的线程被阻塞。我们的解决方法是添加了同步块，START 按钮的 ActionListener 仅仅用于初始化游戏系统，初始化完成后释放同步块的锁（使用 notifyAll()方法）。当游戏主程序获取到锁后再开始游戏，否则要等待图形界面初始化完成并释放锁。
3) 在最初的设计中，我们通过调整后端游戏的运行速度来控制图形界面的刷新速度。但是在调试的过程中发现，由于运行速度是从图形界面的输入文本框中获取的，当用户期望界面慢速刷新（以节省系统资源）而输入一个过大的值时，正如问题 1 中所提到的，程序后端的浮点数运算会产生足以影响程序正常运行的误差。我们将后端的计时器每次自增的量设定为固定值，只有当计时器每自增一个用户指定的值时，才刷新图形界面。这样的修改使得游戏执行速度和图形界面刷新速度分离，避免图形界面刷新速度影响到游戏的正常运行。

六、附录

录屏：蚂蚁爬杆.mp4
项目仓库：https://github.com/uncooleben/OOAD-Labs.git