**关于贪吃蛇大作战简化版的实验报告**

计算机取向： 杜洪超 苏柏瑞

物理取向： 刘全城 李存东

**【前言】**

我们的游戏是参考手机游戏贪吃蛇大作战制作的一个简化版小游戏，整体基于Java图形用户界面，前后历时一个多月，基本达到了预期效果。

【最初的构想】

分好组后，由物理取向的同学最先提出了做这个游戏的想法，我们便开始讨论其可行性。讨论后发现原版的游戏中蛇身的移动是个难点，很难实现360度的旋转，绘制以及建立和棋盘有效的联系等问题，于是我们决定退而求其次，以传统的贪吃蛇界面为基础，保留加速功能，以及AI死亡之后的产生食物，基本规则不发生改变。

**【分工】**

1.界面与游戏主体部分（计算机取向）

该部分交由计算机取向的同学负责实现，其中杜洪超主要负责棋盘Grid类，Gameview和GameControl类，而苏柏瑞同学负责其他成员类和辅助类。

2.AI（物理取向）

因为不同取向的同学编程语言不同，于是我们把较为独立的AI部分交给物理取向的同学，经我们一起讨论出具体算法后交由他们实现。最后把物理取向的同学写好的AI移植到程序中以实现效果。

**【程序的大体框架】**

我们在实现游戏主体部分时，首先确定了主体框架为模型、视图与控制器三部分组成，分别对应了Grid、Snake等，GameView和GameControl三种类，再加上由Snake继承来的子类SnakeAI，主类Snakeapp用于加载游戏，辅助类Direction控制方向，以及基本元素Node类与Food ，这样就建立了我们的大体框架。

**【具体实现中的问题】**

1.棋盘Grid类与各属性成员类之间的联系

这是我们面临的第一个难题，经过不断尝试，我们决定采用三维数组来记录地图，在第三维中采用统一编码来表示各成员属性，再每一步移动后都更新用于存储成员的链表，在根据链表维护三维数组及绘制图形，进而实现了三者之间的有效联系。

2.地图及其效果

由于我们采用传统的上下左右移动方式，并且地图大小被固定，很难实现原版的游戏中的移动地图的效果，故我们改用为全屏显示地图即同时显示所有AI与食物，并用闪动来标识因为地图过大造成的食物太小无法看清的问题。

3其他

在此期间，我们还遇到了很多小问题，如方向的判定，玩家与AI在下一步的先后顺序造成结果不同，等等，诸多问题解决完后主体的框架已经构建完毕。

**【AI的实现】**

1.AI方案一

最初我们制定的方案为搜索加当前局面评估，对AI蛇头为中心的一定范围内进行搜索，对不同元素不同区域进行赋权值求和，最终判断出是否需要转弯或者转弯方向，并增加随机属性以实现移动更为自然。然而最终实现的效果很不理想，AI很难形成较长的蛇身且对障碍物判断不足。

1. AI方案二

我们在原有基础上进行了第二次设计，保持原有的估值系统，同时加上了对墙壁判定的特殊处理，更改前后方向的全值，更改随机数的比例等等。第二个版本的AI相当于在第一个版本上增加了不少补丁，短期上似乎达到了预期的效果，但随着运行时间的增加，问题似乎并没有减少，AI的移动较为死板，且出现了同时移向某个角落导致死亡频率上升的情况，最终我们决定放弃这一版本的AI。

3.AI方案三

经过慎重考虑后，我们重新设计了AI的实现，抛弃原有的估值系统，仍采用搜索的方法，将AI的实现分为三个部分：

第一部分对障碍进行特判，以增强AI的存活能力。

第二部分为AI设立目标机制，在安全情况下，AI会尝试设立一个目标并加速前进，以减少AI随机游动造成聚集的现象。

第三部分由原有估值系统加随机数处理演变过来，对AI进行适当的有效随机控制。后两步均意图使其在游戏中更“智能”。

完成实现后的AI基本达到了最初的要求。

4.AI的弱化

在游戏调试中我们发现AI设计与原版游戏的AI相比有些过强，原版游戏中的AI有犯错的几率从而增强玩家的游戏体验。于是我们对AI进行了部分弱化，并保留原有设计，将其设计为游戏的不同难度。总的来说，弱化后的版本更贴近游戏需求，更符合玩家体验，而难度略高的版本更适合玩家挑战技术，适合不同人群。

**【后期】**

在后期制作中，我们又分别实现了游戏暂停与开始，死亡后的复活，复活后的无敌状态与特效，背景音乐的播放与控制，控制游戏快慢供玩家选择等等，最终确定游戏项目基本完成。

**【感悟】**

总的来说，在这次实验中，我们对图形用户界面与AI的实现都有了深入的理解，很好的提升了我们的代码实现能力与整体设计的能力，是一次非常宝贵的学习经历。

成员信息：

201605130116 计算机取向 杜洪超

201600301108 计算机取向 苏柏瑞

201617100008 物理取向 刘全城

201617100002 物理取向 李存东