|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 学号 | 班级 | 选题 | 论述 | 结论 | 总分 |
|  |  |  |  |  |  |  |

**尝试用Python写一个小游戏**

**--------------掷骰子**

物基二班 汪凌涛 2013301020102

**摘要：本文将着重介绍笔者用python语言写掷骰子游戏的思路与过程。**

**关键词：python 骰子**

正文：

1. 介绍

笔者发现在我们的课本中与课后习题里，大部分的时间我们用python语言来解题，这样略显枯燥乏味。遂笔者一直想用python来编写一个小游戏，借此来放松心情培养兴趣。但因为水平有限，很难编写复杂的程序，所以笔者在这次计算物理的结课作业上选择用python语言来编写一个我们大家都常玩但又简单的小游戏-----掷骰子比赛。在这个程序中，笔者将筹码系统加入其中，使这个程序不至是单纯的比大小，增加了游戏的趣味性。同时笔者又在其中加入了一些额外的规则如反杀等等，让其与一般的掷骰子有所不同。

二、正文

笔者在写这个程序的时候是将这个程序分成了两个部分来编写，这样既可以简化工作也能减少错误的出现。下面笔者就来讲述下写这个程序的思路：

1. 最初思路：笔者认为掷骰子游戏可分为两个大部分：其一就是比大小，这是最重要的也是决定谁赢谁输的部分，这部分语句比较简单，逻辑也比较清晰，还算容易；另一部部分就是笔者为增加游戏乐趣而增加的筹码系统，玩家可在开骰子前进行下注，赚取对手的筹码。这部分逻辑稍稍有些复杂，主要体现在电脑下注的不确定性：笔者不能让电脑自己像人一样根据对手下注的大小与自己的牌面来判断是否弃牌或下注的数目，只能单纯的random，这使得电脑的AI较低，玩家可以较轻松地获胜。
2. 最初版本：笔者是将这两部分分开编写的。

第一部分在最开始要给出双方掷出点数，

笔者用下列语句来实现：def get\_dice():

return random.randint(1,6)

后面就是简单的判断语句：（简化）

If A > B： print ’A赢’

elif B>A： print ‘B赢’

Else： print ‘平局’

这部分并未加入任何与分数有关的语句逻辑，只是单纯的随机掷出骰子的点数再进行比较大小，输出结果。

第二部分就是筹码系统，笔者在最初规定玩家与电脑都各有100个数目的初始金钱，并规定在有一方金钱数目为零的时候游戏结束，否则就会一直进行下一局游戏。在每局游戏开骰子前都会让玩家选择是否下注并注明下注的数目，一局里赢得或输掉的筹码即你下注的筹码，在玩家下注后电脑会自己下注，但因为技术所限，电脑只会随机决定下不下注，并且只能随机在1---它当前拥有最大数目的金钱之间选择下注数目。

这一部分的难点是电脑随机弃牌与随机下注，可用下列语句实现：

if random.choice('yn') == str('y'):

computer\_bets = random.randint(1,computer\_money)

1. 两部分的连接：在两大部分都相继完成且都能完美运行以后就要将两部分程序连接在一起，关键就在于判断大小后输赢与分数加减的联系。可用下列语句进行实现：

If player win: player\_money += computer\_bets

computer\_money -= computer\_bets

If computer win:player\_money -= player\_bets

computer\_money += player\_bets

1. 优化程序：两大部分连接后程序已经可以做到:

显示双方当前筹码数量---->>双方掷出骰子点数(玩家只知道自己点数）---->>玩家下注---->>电脑随机下注---->>显示双方点数---->>比大小，声明结果---->>进行下一局游戏

的简单循环，但这个程序还过于简单，趣味性也不高，并不是我想要的最终版本。我还想在这基础上增加一些特别的机制来增添游戏的乐趣，我想到了我家乡玩骰子的一个规则：反杀。就是在有一方掷出1点的时候，它虽然比别的点数都要小，但却可以反杀6点，而且可以赢得对方所有的筹码。反杀机制的加入让我的程序不同于一般的掷骰子，增添了游戏的多变性。

除此之外，我还发现游戏过程中缺少了一项重要的东西：带入感。如果程序只是机械性地运行中间没有停顿的话那么玩家就没有了带入感与紧张感，这大大减少了游戏的乐趣，为解决这一问题，我想到了使用停顿：time.sleep()语句，即在每行输出时停顿一定时间，让玩家身临其境。比如在每次电脑下注前都要停顿1-2秒钟，仿佛玩家对面的对手是个正在思考下什么注的真人玩家，让玩家有了一种带入感，大大提高可玩性。

同时，我还优化了一些判断语句中的逻辑问题（如玩家/电脑弃牌后游戏依然进行等），消除了BUG的发生。

至此，最终版本的程序终于出炉，

1. 结论

掷骰子程序基本完成，但其中还有着一些问题未能解决：比如电脑AI较低，还不能根据计算胜率来下注；在一方下高注时，低注者可以以较低筹码赢得高注。这些问题过于复杂，虽可以用最基本的逻辑一步步编写出来但太过冗长，导致程序错误百出。经过这次计算物理的大作业我发现，随着我们编写的程序的逻辑越来越复杂，程序编写的难度也逐步提升，若只是用最基本最普通的逻辑编写程序，那工作量十分吓人，而且其中的一个小小的错误就会导致程序的崩溃，实属不易。我终于发现一个编程者之所以被称为大神，就是因为他可以用最少的语句，最简单的逻辑来写出毫无漏洞的复杂程序，这十分值得我们学习，也是我以后努力的方向。