

《随机过程实验》

实验报告

(最优停时游戏模拟)

学院名称 : 数据科学与计算机学院

年级(专业) : 16 级 信息安全专业

学生姓名 : 邓中天

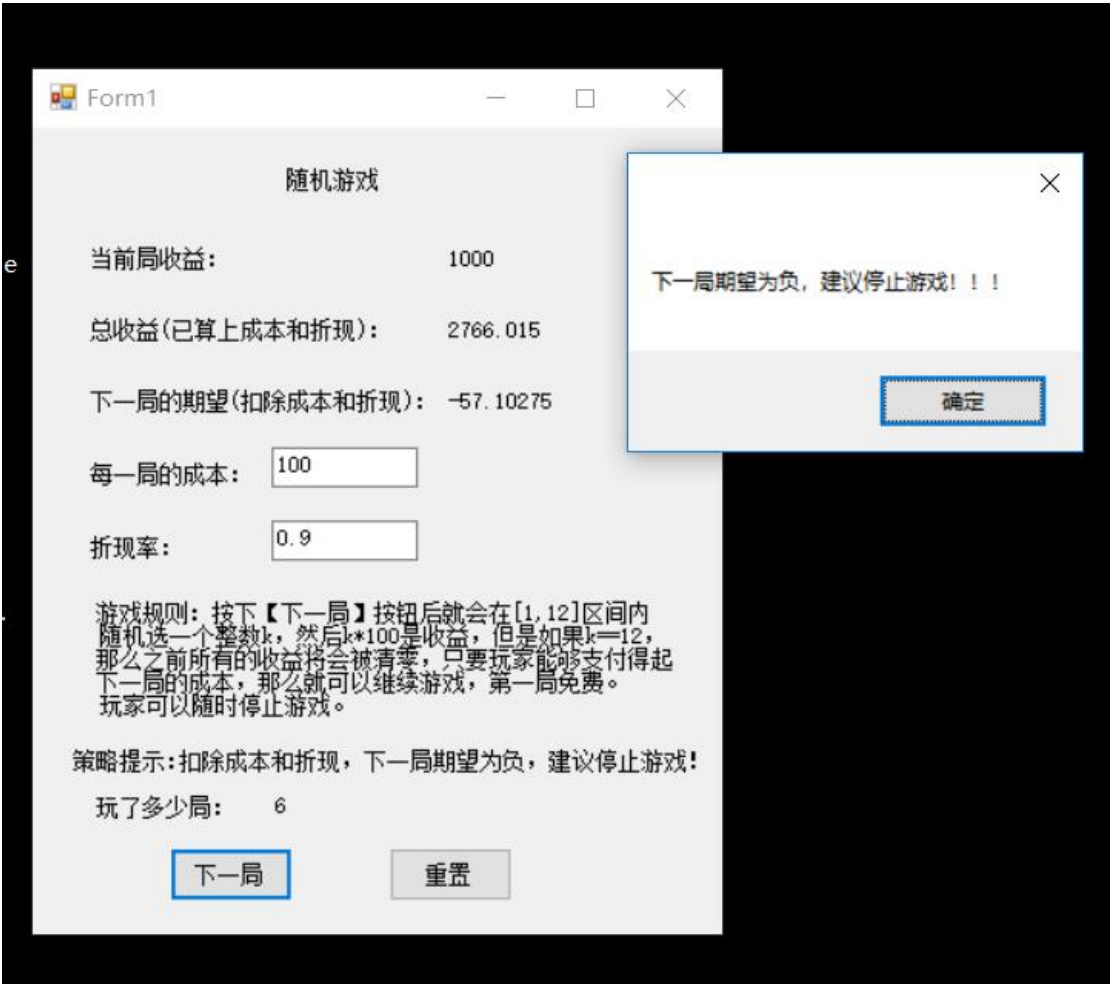
学号 : 16337045

时间 : 2018 年 6 月 10 日

一、实验题目：模拟幸运轮游戏

二、实验概述：使用 C#语言编写 Windows 窗口程序，实现一个可以自定义游戏成本，可以调整折现率的程序，模拟幸运轮随机旋转后得到的收益情况。

以下是效果图：



可以看到，在图片里面，用户可以自己输入每一局的成本和折现率，折现率表示每玩一局，用户的钱就会乘上折现率来贬值。图片里面还有当前局的收益、总收益和下一局的期望，当下一局的期望为负的时候，就说明现在是最优停时了，建议停止游戏。

用户在充分考虑收益和风险后，可以进行下一局。

重置的功能是重新开始游戏。

实际测试中，玩 10 局左右就能够达到最优停时。

三、计算公式

这里先声明一些变量的意义：

a: 一个 1 到 12 的随机整数

p: 折现率（用户自己输入），范围是(0,1]

g: 玩一局的成本（用户自己输入），范围是[0,1100)

m: 总收益

c: 当前局收益

n: 下一局期望

count: 玩了少局

这里假设第一局不要钱，当游戏顺利进行下去的时候，每次按下【下一局】按钮之后，有以下公式的变化：

总收益：

$$m = m * p + c - g$$

当前局的收益：

$$c = a * 100$$

如果 $a=12$ ，那么 $c = -m$

下一局的期望：

$$n = (-1) * m * (1 - p) - m / 12 + 550 - g$$

其中， $550 = (1/12) \sum (i * 100)$ ，i 从 1 到 11

核心公式就是上面的几个了。

不过具体实现起来，需要注意一些细节。比如要注意好第一局不要成本的时候，总收益不用减掉成本；用户输入的成本要在[0,1100)这个范围内，不然没办法把游戏继续下去；折现率也要考虑到要在(0,1]这个范围内，小于等于 0 则会让自己的总收益为 0 或负，大于 1 则是一个通货紧缩的环境，这里还是不考虑了吧，毕竟现在通货膨胀率为 7%左右，如果通货膨胀率为 7%，那么折现率就是 0.93。

实际算法和上面的公式是一样的，用 C#编写窗口程序还是非常方便的。

四、代码分析

原理：给按钮绑定函数，当按下按钮的时候就可以进行各种计算，并将结果展示出来。

定义变量：

```
15      static double c = 0; //当前局收益
16      static double m = 0; //总收益
17      static double n = 0; //下一局的期望
18      static double g = 0; //玩一局的成本
19      static double p = 0; //折现率
20      static int count = 0; //玩了多少局
21
```

【重置按钮】绑定的函数：前半部是清零操作，后半部分控制窗口的显示

```
1 个引用
32 private void button2_Click(object sender, EventArgs e)
33 {
34     count = 0;
35     m = 0;
36     c = 0;
37     n = 0;
38     count = 0;
39
40     label17.Text = c.ToString();
41
42     label19.Text = m.ToString();
43
44     label110.Text = n.ToString();
45
46     label118.Text = count.ToString();
47
48     // MessageBox.Show("重置成功!");
49 }
50
```

【下一局】按钮分析：

控制窗口内容显示部分：

```
131     label17.Text = c.ToString();
132
133     label19.Text = m.ToString();
134
135     label110.Text = n.ToString();
136
137     label118.Text = count.ToString();
138
139     if (n >= 0) //策略提示
140     {
141         label111.Text = "扣除成本和折现，下一局期望为正，可以继续玩!";
142     }
143     else
144     {
145         label111.Text = "扣除成本和折现，下一局期望为负，建议停止游戏!";
146         MessageBox.Show("下一局期望为负，建议停止游戏!!!");
147     }
148
```

核心算法，和第三部分讲的计算公式是一样的

```
93 //
94 count++; //玩了多少局
95
96 Random rNumber = new Random();
97 int a = rNumber.Next(1, 13);
98
99 if (a == 12)
100 {
101
102     c = (-1) * m;
103
104     m = 0;
105
106     label17.Text = c.ToString();
107
108     label19.Text = "抽到12, 总收益清零!! ";
109
110     label10.Text = n.ToString();
111
112     label18.Text = count.ToString();
113
114     return;
115 }
116
117 c = a * 100; //当前局收益
118
119 if (count == 1)
120 {
121     m = m * p + c; //总收益, 第一局不需要成本
122 }
123 else
124 {
125     m = m * p + c - g; //总收益
126 }
127
128
129 n = (-1) * m * (1 - p) - m / 12 + 550 - g; //下一局的期望
130
```

这个是一些基础的提醒，防止用户不输入或者输入非法数据

```
1 个引用
51 private void button6666_Click(object sender, EventArgs e)
52 {
53
54
55     if (textBox2.Text.Trim() == String.Empty)
56     {
57         MessageBox.Show("请填写每一局的成本");
58         return;
59     }
60     if (textBox1.Text.Trim() == String.Empty)
61     {
62         MessageBox.Show("请填写折现率");
63         return;
64     }
65     g = double.Parse(textBox2.Text.ToString());
66     p = double.Parse(textBox1.Text.ToString());
67     if (g < 0 || g >= 1100)
68     {
69         MessageBox.Show("每一局的成本要在[0, 1100)这个区间内!");
70         return;
71     }
72
73     if (p <= 0 || p > 1)
74     {
75         MessageBox.Show("折现率要在(0, 1]这个区间内!");
76         return;
77     }
78
79     if (count > 0 && m < g)
80     {
81         MessageBox.Show("成本不足, 无法继续游戏!");
82         return;
83     }
84
```

五、总结

这个模拟实验，应该是完美的完成了要求了。

1. 版本复杂到考虑了成本和折现率，并且每一局都是可以修改成本和折现率的，这样就更加贴合实际。当成本为 0，折现为 1 的时候，就是最简单的最优停时方案。成本和折现的四种组合，其实就对应了教材的不同的解决方案。

2. 界面虽然不是非常的优雅，不过也做到了简单实用，功能都齐全了，也可以跟踪每一局的实际情况，为下一局做出提示策略。

3. 算法上面，充分考虑了最优停时策略计算期望的部分，成功实现了通过判断下一局期望的正负来做决策，并且期望的公式也足够复杂和贴近实际。

4. 程序写的还算是规整，该考虑的非输入都考虑了，也有人性化的重置按钮，使用 C# 确实提高了开发效率。

六、注意事项

C# 的窗口程序要依赖 Windows 的 .net 环境，运行程序之前要记得打开 .net 环境，不然程序没办法运行。