DIY 游戏之多彩弹弹乐

程成 2015335550055

武汉大学物理科学与技术学院 物理学基地二班

摘要

Python 拥有众多的功能，基于物理学基本原理的数值计算与模拟是计算机物理的主要内容，此次我们将介绍如花利用 Python 编写一个功能齐全的 DIY 游戏，运用 Python 来简单实现设计游戏的乐趣。由于此实验来源于初学者，所以诸如声音，视频处理方面的有趣特性就略过不讲了。

关键词

设计 DIY 弹球 完全碰撞

1 引言

1.1 背景

本次 DIY 游戏以安卓手机上排名第一的弹球游戏为基础游戏的基本内容：弹球游戏真实再现了一直以来最经典的弹珠台游戏的玩法。此款游戏在弹球游戏中为模拟真实球体的物理碰撞和图形细节处理设定了一个新的标准。你一定会为弹球真实度和顶尖的视觉效果感到震惊。

1.2 目标

本项目的具体目标是简单模拟弹球游戏的完全碰撞过程，可能具体的游戏性不强，但游戏状态可以作为另外一个有用的游戏需求，而游戏状态也是游戏设计的一部分。除此之外，添加新的状态也应该可以轻松实现。

2 工具

本项目需要的工具是 Python，从 Python 官网上下载版本 3.3.2

3 正文与设计

3.1 游戏规则

设计五彩色小球在画布中移动，他们之间会产生碰撞，当然小球和上下左右都会产生碰撞，碰撞后，小球会改变方向返回，而最下面的游标则用于调节小球的移动速度，游标的范围是[-200, 200]

3.2 基本判断

首先判断两个小球：圆心 1（A(x1,y1) 半径：r X 轴速度：Vax Y 轴速度：Vay）圆心 2（B(x2,y2) 半径：R X 轴速度：Vbx Y 轴速度：Vby）
碰撞的条件是：

1.两个小球的圆心距离不大于两小球半径之和(r+R)，即：

$$(x2 - x1)^2 + (y2 - y1)^2 \leq (r + R)^2$$

2.小球碰撞后，两小球的数度交换，即：

```
tempVax = Vax
tempVay = Vay
Vax = Vbx
Vay = Vby
Vbx = tempVax
Vby = tempVay
```

```
Vax = Vax + Vbx
Vbx = Vax - Vbx
Vax = Vax - Vbx
Vay = Vay + Vby
Vby = Vay - Vby
Vay = Vay - Vby
```

3.3 程序设计

3.3.1 创建控件

1.首先设计画布与画布比例，

```
self.scaling = 100.0
self.canvas_width = 10
self.canvas_height = 5.6
self.draw = Canvas(self, width=(self.canvas_width * self.scaling),
                   height=(self.canvas_height * self.scaling),
                   bg='white')
```

2.加入一个简单的游标

```
self.speed = Scale(self, orient=HORIZONTAL, label="ball speed",
                   from_=-200, to=200)

self.speed.pack(side=BOTTOM, fill=X)
```

3.小球与墙壁初始值

```
self.ball_d = 1.0
self.scaling_left = round(self.ball_d / 2, 1)
self.scaling_right = self.canvas_width - self.scaling_left
self.scaling_bottom = self.canvas_height - self.scaling_left
self.scaling_top = self.scaling_left

self.ball = self.draw.create_oval("0.60i", "0.60i", "1.60i", "1.60i",
                                  fill="red")
self.second_ball = self.draw.create_oval("2.0i", "2.0i", "3.0i", "3.0i",
                                         fill='black')
self.three_ball = self.draw.create_oval("4.0i", "4.0i", "5.0i", "5.0i",
                                        fill='brown')
self.four_ball = self.draw.create_oval("6.0i", "2.0i", "7.0i", "3.0i",
                                       fill='green')
self.five_ball = self.draw.create_oval("8.0i", "3.0i", "9.0i", "4.0i",
                                       fill='gray')
```

3.3.2 更新小球的坐标

把各个小球的圆心坐标信息以及速度信息存放到数组中，便于在后面循环遍历的时候使用。

```
        self.ball_x.append(self.x)
        self.ball_y.append(self.y)
        self.ball_v_x.append(self.velocity_x)
        self.ball_v_y.append(self.velocity_y)

        self.ball_x.append(self.second_ball_x)
        self.ball_y.append(self.second_ball_y)
        self.ball_v_x.append(self.second_ball_v_x)
        self.ball_v_y.append(self.second_ball_v_y)

        self.ball_x.append(self.three_ball_x)
        self.ball_y.append(self.three_ball_y)
        self.ball_v_x.append(self.three_ball_v_x)
        self.ball_v_y.append(self.three_ball_v_y)

        self.ball_x.append(self.four_ball_x)
        self.ball_y.append(self.four_ball_y)
        self.ball_v_x.append(self.four_ball_v_x)
        self.ball_v_y.append(self.four_ball_v_y)

        self.ball_x.append(self.five_ball_x)
        self.ball_y.append(self.five_ball_y)
        self.ball_v_x.append(self.five_ball_v_x)
        self.ball_v_y.append(self.five_ball_v_y)
```

3.3.3 更新各个小球速度信息

小球碰撞到四周和另外的小球索要更新的速度信息游标值

```
self.scale_value = self.speed.get() * 0.1
if (self.ball_x[index] > self.scaling_right) or (self.ball_x[index] < self.scaling_left):
    self.ball_v_x[index] = -1.0 * self.ball_v_x[index]
if (self.ball_y[index] > self.scaling_bottom) or (self.ball_y[index] < self.scaling_top):
    self.ball_v_y[index] = -1.0 *  self.ball_v_y[index]


for n in range(len(self.balls)):
    if (round((self.ball_x[index] - self.ball_x[n])**2 + (self.ball_y[index] - self.ball_y
        temp_vx = self.ball_v_x[index]
        temp_vy = self.ball_v_y[index]
        self.ball_v_x[index] = self.ball_v_x[n]
        self.ball_v_y[index] = self.ball_v_y[n]
        self.ball_v_x[n] = temp_vx
        self.ball_v_y[n] = temp_vy
```

3.3.4 获取小球 XY 轴坐标移动距离

1.获取小球 X 轴坐标移动距离并且更新小球的圆心 X 坐标，返回 X 轴所需移动距离.

```
        deltax = (self.ball_v_x[index] * self.scale_value / self.scaling)
        self.ball_x[index] = self.ball_x[index] + deltax
        return deltax
```

2.获取小球 Y 轴坐标移动距离并且更新小球的圆心 Y 坐标，返回 Y 轴所需移动距离。

```
        deltay = (self.ball_v_y[index] * self.scale_value / self.scaling)
        self.ball_y[index] = self.ball_y[index] + deltay
        return deltay
```

3.3.5 移动各个小球

移动各个小球，编号为：分别为 01234,这是根据数组：self.balls 确定的。

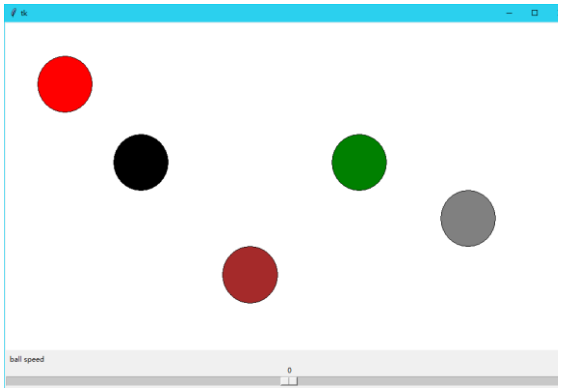
```
def moveBall(self, *args):
    self.update_ball_velocity(0)
    deltax = self.get_ball_deltax(0)
    deltay = self.get_ball_deltay(0)
    self.draw.move(self.ball,  "%ri" % deltax, "%ri" % deltay)
    self.after(10, self.moveBall)

def move_second_ball(self, *args):
    self.update_ball_velocity(1)
    deltax = self.get_ball_deltax(1)
    deltay = self.get_ball_deltay(1)
    self.draw.move(self.second_ball,  "%ri" % deltax, "%ri" % deltay)
    self.after(10, self.move_second_ball)
```

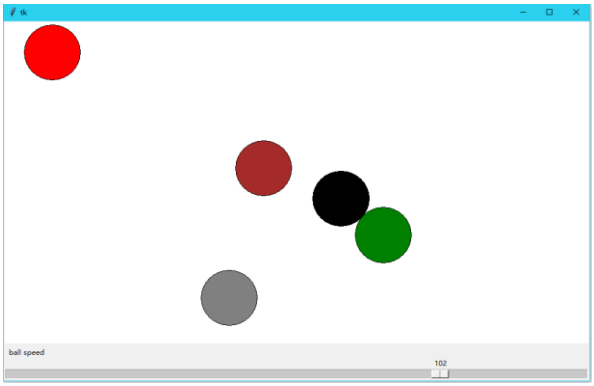
4 运行游戏

在 Windows10 中运行 python_tkinter_peng.1.1.py

1.此为游戏初始截图：



2.此为游戏过程截图：



5 缺陷或 BUG

1. 在修改游标数据从而改变小球移动速度的时候，小球移动的距离得不到及时的更新导致小球可能会逃离画布

2. 小球在运动的过程中，有时候也有可能逃离画布

6 进一步探索

下面是一些改进游戏的想法，介于初学者实在能力有限，以下有待开发：

1. 增加游戏背景声音，与碰撞音效。
2. 增大游戏自主性，可自主改变小球的运动轨迹。
3. 增加记录分数的功能，可随碰撞次数的增加，提高分数。
4. 创建游戏的独立可执行版本，并安装程序打包等。

7 致谢

感谢上一届大量学长的资料。