# 第六周 小球游戏

## 第一版本 小球与球拍

### 小球类

class Ball():

    '''

    定义小球

    '''

    def \_\_init\_\_(self, canvas,  x=100, y=250, r=30, color='blue'):

        '''

        参数说明:

        画布  canvas

        中心点[x,y]

        半径 r

        小球颜色 color

        '''

        #print (x,y,r,color)

        # 保存画板

        self.canvas = canvas

        # 保存小球颜色

        self.color = color

        # 保存中心点坐标

        self.\_x = x

        self.\_y = y

        # 保存半径

        self.\_r = r

        # 随机生成XY方向的步长 \_dx \_dy

        ss = [-5, -3, -1, 1, 3, 5]

        s = random.choices(ss, k=2)

        #print (s)

        self.\_dx = s[0]  # random.choice(ss) # random.randint(-5,5)

        self.\_dy = s[1]  # random.choice(ss) # random.randint(-5,5)

        #print ('self.\_dx,self.\_dy',self.\_dx,self.\_dy)

        # 获取画布的宽高

        self.width = self.canvas.winfo\_width()

        self.height = self.canvas.winfo\_height()

        # 绘制小球

        self.id = self.canvas.create\_oval(

            self.\_x-self.\_r, self.\_y-self.\_r, self.\_x+self.\_r, self.\_y+self.\_r, fill=self.color)

    def draw(self):

        self.canvas.moveto(self.id, self.\_x-self.\_r, self.\_y-self.\_r)

    def go(self):

        '''

        小球随机运动

        '''

        # 计算新的中心点坐标 和  X Y方向的步长

        # 计算新的中心点 X 坐标

        self.\_x += self.\_dx

        # 如果 已经到达 右边

        if (self.\_x+self.\_r > self.width):

            self.\_x = self.width - self.\_r

            # X方向的步长值 为原值的相反  ？？思考

            self.\_dx = (-1)\*self.\_dx

        # 如果 已经到达 左边

        if (self.\_x-self.\_r < 0):

            self.\_x = self.\_r + 1

            self.\_dx = (-1)\*self.\_dx

        # 计算新的中心点 Y 坐标

        self.\_y += self.\_dy

        # 如果 已经到达 底边

        if (self.\_y+self.\_r > self.height):

            self.\_y = self.height - self.\_r

            # X方向的步长值 为原值的相反  ？？思考

            self.\_dy = (-1)\*self.\_dy

        # 如果 已经到达 顶边

        if (self.\_y-self.\_r < 0):

            self.\_y = self.\_r + 1

            self.\_dy = (-1)\*self.\_dy

        self.draw()

### 测试小球

if \_\_name\_\_ == '\_\_main\_\_':

    win = Tk()

    win.geometry('600x400+100+100')

    mycanvas = Canvas(win, width=300, height=200, bg='white')

    mycanvas.pack(padx=0, fill='both', expand=True)

    win.update()

    # 从CColors随机选择十种颜色

    #print (list(CColors.keys()))

    colors = random.choices(list(CColors.keys()), k=10)

    #print (colors)

    # 生成十种颜色的小球

    balls = []

    for color in colors:

        #print (i)

        #ball = Ball(mycanvas,x= 100,y=250,r=30,color = CColors[color] )

        # 随机的中心位置和半径

        ball = Ball(mycanvas, x=random.randint(50, 550), y=random.randint(

            50, 350), r=random.randint(10, 60), color=CColors[color])

        balls.append(ball)

    #ball2 = Ball(mycanvas,x= 200,y=150,r=35,color = 'yellow')

    win.update()

    # 让十个小球随机运动

    while True:

        for ball in balls:

            ball.go()

        # 测试 有没有 win.update() 语句 的不同效果

        win.update()

        time.sleep(0.05)

    win.mainloop()

### 运行结果：

### 程序分析

建立小球类 Ball

\_\_init\_\_()

**作用：**初始化属性

画布  canvas

中心点[x,y]

半径 r

小球颜色 color

生成XY方向的步长 \_dx \_dy

       获取画布的宽高

        self.width = self.canvas.winfo\_width()

        self.height = self.canvas.winfo\_height()

def go(self):

        '''

        小球随机运动

        '''

       # 计算新的中心点坐标 和  X Y方向的步长

       # 计算新的中心点 X 坐标

def draw(self):

绘制自己

### 测试程序：

创建窗口

win = Tk()

  win.geometry('600x400+100+100')

创建画布

  mycanvas = Canvas(win, width=300, height=200, bg='white')

  mycanvas.pack(padx=0, fill='both', expand=True)

  win.update()

    # 从CColors随机选择十种颜色

    #print (list(CColors.keys()))

    colors = random.choices(list(CColors.keys()), k=10)

    #print (colors)

    # 生成十种颜色的小球

    balls = []

    for color in colors:

        #print (i)

        #ball = Ball(mycanvas,x= 100,y=250,r=30,color = CColors[color] )

        # 随机的中心位置和半径

        ball = Ball(mycanvas, x=random.randint(50, 550), y=random.randint(

            50, 350), r=random.randint(10, 60), color=CColors[color])

        balls.append(ball)

    #ball2 = Ball(mycanvas,x= 200,y=150,r=35,color = 'yellow')

    win.update()

    # 让十个小球随机运动

    while True:

        for ball in balls:

            ball.go()

        # 测试 有没有 win.update() 语句 的不同效果

        win.update()

        time.sleep(0.05)

    win.mainloop()

### 球拍类

class Paddle():

    '''

    定义球拍

    '''

    def \_\_init\_\_(self, canvas, x=100, y=250, color='blue'):

        # 保存画板

        self.canvas = canvas

        # 保存小球颜色

        self.color = color

        # 保存左上角坐标

        self.\_x = x

        self.\_y = y

        # 球拍宽度

        self.PaddleWidth = 100

        self.PaddleHeight = 15

        # 随机生成X方向的步长 \_dx 使每次运行的速度不同

        ss = [-5, -3, -2,2, 3, 5]

        s = random.choice(ss)

        #print (s)

        self.Step = s  # random.choice(ss) # random.randint(-5,5)

        self.\_dx = self.Step

        # 获取画布的宽度

        self.width = self.canvas.winfo\_width()

        self.id = self.canvas.create\_rectangle(

            self.\_x, self.\_y, self.\_x+self.PaddleWidth, self.\_y+self.PaddleHeight, fill=self.color)

        # 画板绑定键盘的任何事件

        self.canvas.bind\_all('<Key>', self.AnyKey)

    def AnyKey(self, event):

        '''

        键盘的任何事件

        '''

        print (event.keysym,self.\_x ,self.\_dx)

        # 如果 不是左右键 则不处理

        if (event.keysym not in ['Left', 'Right']):

            return

        self.\_x += self.\_dx

        # 计算新左上角坐标 和  X方向的步长

        if event.keysym == 'Left' :

            self.\_dx = self.Step

            # 如果 已经到达 左边

            if (self.\_x < 0):

                self.\_x = 0

                self.\_dx = (-1)\*self.Step

            elif (self.\_x+self.PaddleWidth > self.width):

                self.\_x = self.width -self.PaddleWidth

                #不动

                self.\_dx = 0

        elif event.keysym == 'Right' :

            self.\_dx = (-1) \*self.Step

            # 如果 已经到达 右边

            if (self.\_x+self.PaddleWidth > self.width):

                self.\_x = self.width -self.PaddleWidth

                # X方向的步长值 为原值的相反  ？？思考

                self.\_dx = self.Step

            elif (self.\_x < 0):

                self.\_x = 0

                #不动

                self.\_dx = 0

        self.draw()

    def draw(self):

        self.canvas.moveto(self.id, self.\_x, self.\_y)

    def ChangeSize(self):

        #重新获取Canvas 的宽度

        self.width = self.canvas.winfo\_width()

### 测试球拍

if \_\_name\_\_ == '\_\_main\_\_':

    def ChangeConfigure(event):

        paddle.ChangeSize()

    win = Tk()

    win.geometry('600x400+100+100')

    mycanvas = Canvas(win, width=300, height=200, bg='yellow')

    mycanvas.pack(fill='both', expand=True)

    win.update()

    win.bind('<Configure>',ChangeConfigure)

    paddle = Paddle(mycanvas, x=100, y=250,color ='red')

    win.mainloop()

### 知识点

1. **类的定义**

抽象、类、实例的概念 实际小球抽象 =>Ball类， 实例化 => ball = Ball()

类的继承 class Ball() 从 Object中继承 等同 于class Ball(Object)

初始化函数 \_\_init\_\_(self)

1. **类中的函数定义**

go（self,…. ）

draw（self,…. ）

类内部属性的使用 self.\*

1. **事件和事件绑定**

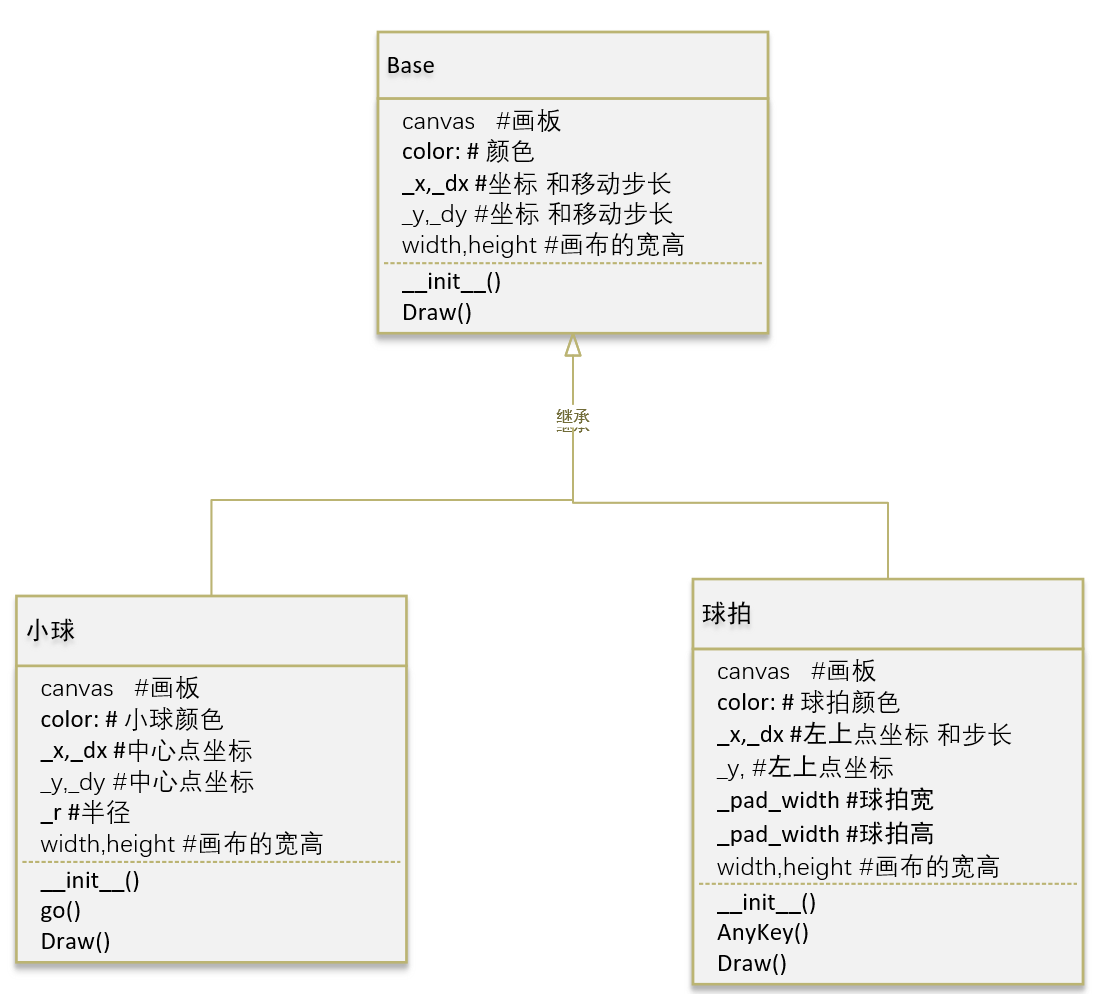
**Widget.**bind (事件，响应函数) **Widget部件对象**

**canvas. b**ind\_all (事件，响应函数) **canvas画布对象**

1. **具体算法设计**

## 类的继承扩展

### 小球与球拍的相同点



找出 共同的属性和方法



### 基类定义

class Base():

    '''

    定义小球和球拍的基类

    '''

    def \_\_init\_\_(self, canvas,  x=100, y=250, color='blue'):

        '''

        参数说明:

        画布  canvas

        坐标 [x,y]

        小球颜色 color

        '''

        # 保存画板

        self.canvas = canvas

        # 保存小球颜色

        self.color = color

        # 保存中心点坐标

        self.\_x = x

        self.\_y = y

        # 获取画布的宽高

        self.width = self.canvas.winfo\_width()

        self.height = self.canvas.winfo\_height()

    def draw(self):

        # 绘制自己

        print('Base.draw')

        pass

    def move(self):

        print('Base.move')

        # 移动自己

        pass

    def go(self):

        # 重新定位 自己的XY 坐标

        pass

### Ball类定义

class Ball(Base):

    '''

    定义小球

    '''

    def \_\_init\_\_(self, canvas,  x=100, y=250, r=30, color='blue'):

        '''

        参数说明:

        画布  canvas

        中心点[x,y]

        半径 r

        小球颜色 color

        '''

        super().\_\_init\_\_(canvas, x=x, y=y, color=color)

        # 保存半径

        self.\_r = r

        # 随机生成XY方向的步长 \_dx \_dy

        ss = [-5, -3, -1, 1, 3, 5]

        s = random.choices(ss, k=2)

        #print (s)

        self.\_dx = s[0]  # random.choice(ss) # random.randint(-5,5)

        self.\_dy = s[1]  # random.choice(ss) # random.randint(-5,5)

        #print ('self.\_dx,self.\_dy',self.\_dx,self.\_dy)

        # 绘制小球

        self.draw()

    def draw(self):

        print('Ball.draw')

        # 绘制小球

        self.id = self.canvas.create\_oval(

            self.\_x-self.\_r, self.\_y-self.\_r, self.\_x+self.\_r, self.\_y+self.\_r, fill=self.color)

    def move(self):

        self.canvas.moveto(self.id, self.\_x-self.\_r, self.\_y-self.\_r)

    def go(self):

        '''

        小球随机运动

        '''

        # 计算新的中心点坐标 和  X Y方向的步长

        # 计算新的中心点 X 坐标

        self.\_x += self.\_dx

        # 如果 已经到达 右边

        if (self.\_x+self.\_r > self.width):

            self.\_x = self.width - self.\_r

            # X方向的步长值 为原值的相反  ？？思考

            self.\_dx = (-1)\*self.\_dx

        # 如果 已经到达 左边

        if (self.\_x-self.\_r < 0):

            self.\_x = self.\_r + 1

            self.\_dx = (-1)\*self.\_dx

        # 计算新的中心点 Y 坐标

        self.\_y += self.\_dy

        # 如果 已经到达 底边

        if (self.\_y+self.\_r > self.height):

            self.\_y = self.height - self.\_r

            # X方向的步长值 为原值的相反  ？？思考

            self.\_dy = (-1)\*self.\_dy

        # 如果 已经到达 顶边

        if (self.\_y-self.\_r < 0):

            self.\_y = self.\_r + 1

            self.\_dy = (-1)\*self.\_dy

        self.move()

### 球拍类定义

class Paddle(Base):

    '''

    定义球拍

    '''

    def \_\_init\_\_(self, canvas, x=100, y=250, color='blue'):

        super().\_\_init\_\_(canvas, x=x, y=y, color=color)

        # 球拍宽度

        self.PaddleWidth = 100

        self.PaddleHeight = 15

        # 随机生成X方向的步长 \_dx 使每次运行的速度不同

        ss = [-5, -3, -2, 2, 3, 5]

        s = random.choice(ss)

        #print (s)

        self.Step = s  # random.choice(ss) # random.randint(-5,5)

        self.\_dx = self.Step

        self.draw()

        # self.id = self.canvas.create\_rectangle(

        #    self.\_x, self.\_y, self.\_x+self.PaddleWidth, self.\_y+self.PaddleHeight, fill=self.color)

        # 画板绑定键盘的任何事件

        self.canvas.bind\_all('<Key>', self.AnyKey)

    def AnyKey(self, event):

        '''

        键盘的任何事件

        '''

        #print(event.keysym, self.\_x, self.\_dx)

        # 如果 不是左右键 则不处理

        if (event.keysym not in ['Left', 'Right']):

            return

        self.\_x += self.\_dx

        # 计算新左上角坐标 和  X方向的步长

        if event.keysym == 'Left':

            self.\_dx = self.Step

            # 如果 已经到达 左边

            if (self.\_x < 0):

                self.\_x = 0

                self.\_dx = (-1)\*self.Step

            elif (self.\_x+self.PaddleWidth > self.width):

                self.\_x = self.width - self.PaddleWidth

                # 不动

                self.\_dx = 0

        elif event.keysym == 'Right':

            self.\_dx = (-1) \* self.Step

            # 如果 已经到达 右边

            if (self.\_x+self.PaddleWidth > self.width):

                self.\_x = self.width - self.PaddleWidth

                # X方向的步长值 为原值的相反  ？？思考

                self.\_dx = self.Step

            elif (self.\_x < 0):

                self.\_x = 0

                # 不动

                self.\_dx = 0

        self.move()

    def draw(self):

        self.id = self.canvas.create\_rectangle(

            self.\_x, self.\_y, self.\_x+self.PaddleWidth,

            self.\_y+self.PaddleHeight, fill=self.color)

    def move(self):

        self.canvas.moveto(self.id, self.\_x, self.\_y)

    def ChangeSize(self):

        # 重新获取Canvas 的宽度

        self.width = self.canvas.winfo\_width()

### 测试部分 小球与球拍同框

# 测试小球

if \_\_name\_\_ == '\_\_main\_\_':

    def ChangeConfigure(event):

        paddle.ChangeSize()

    win = Tk()

    win.geometry('600x400+100+100')

    mycanvas = Canvas(win, width=300, height=200, bg='white')

    mycanvas.pack(padx=0, fill='both', expand=True)

    win.update()

    win.bind('<Configure>', ChangeConfigure)

    paddle = Paddle(mycanvas, x=100, y=250, color='red')

    # 从CColors随机选择十种颜色

    #print (list(CColors.keys()))

    colors = random.choices(list(CColors.keys()), k=10)

    #print (colors)

    # 生成十种颜色的小球

    balls = []

    for color in colors:

        #print (i)

        #ball = Ball(mycanvas,x= 100,y=250,r=30,color = CColors[color] )

        # 随机的中心位置和半径

        ball = Ball(mycanvas, x=random.randint(50, 550), y=random.randint(

            50, 350), r=random.randint(10, 60), color=CColors[color])

        balls.append(ball)

    #ball2 = Ball(mycanvas,x= 200,y=150,r=35,color = 'yellow')

    win.update()

    # 让十个小球随机运动

    while True:

        for ball in balls:

            ball.go()

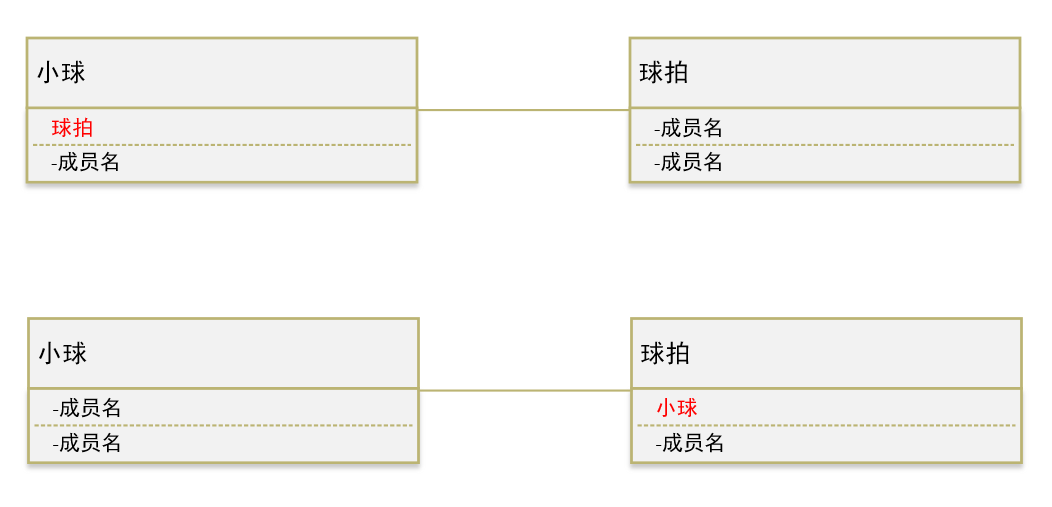
        # 测试 有没有 win.update() 语句 的不同效果

        win.update()

        time.sleep(0.05)

    win.mainloop()

## 如何让小球与球拍有关系呢？



是小球的成员中有球拍 还是球拍的成员中有小球 ？？？

我们看一下 小球 和球拍是如何 运动的？

**小球运动**

# 让十个小球随机运动

    while True:

        for ball in balls:

            ball.go()

        # 测试 有没有 win.update() 语句 的不同效果

        win.update()

        time.sleep(0.05)

**球拍运动**

# 画板绑定键盘的任何事件

 self.canvas.bind\_all('<Key>', self.AnyKey)

def AnyKey(self, event):

所以在 中小球中加入 球拍成员 pad

在 def \_\_init\_\_(self, canvas,  x=100, y=250, r=30, color='blue'): 加入 pad 参数

在 def \_\_init\_\_(self, canvas,  x=100, y=250, r=30, **pad:Paddle**,color='blue'):

**pad:Paddle 说明 形参 pad 是Paddle 类型**

def \_\_init\_\_(self, canvas,  x=100, y=250, r=30, **pad:Paddle**,color='blue'):

        '''

        参数说明:

        画布  canvas

        中心点[x,y]

        半径 r

        小球颜色 color

        '''

        super().\_\_init\_\_(canvas, x=x, y=y, color=color)

#保存球拍

self.pad = pad

        # 保存半径

        self.\_r = r

        # 随机生成XY方向的步长 \_dx \_dy

        ss = [-5, -3, -1, 1, 3, 5]

        s = random.choices(ss, k=2)

        #print (s)

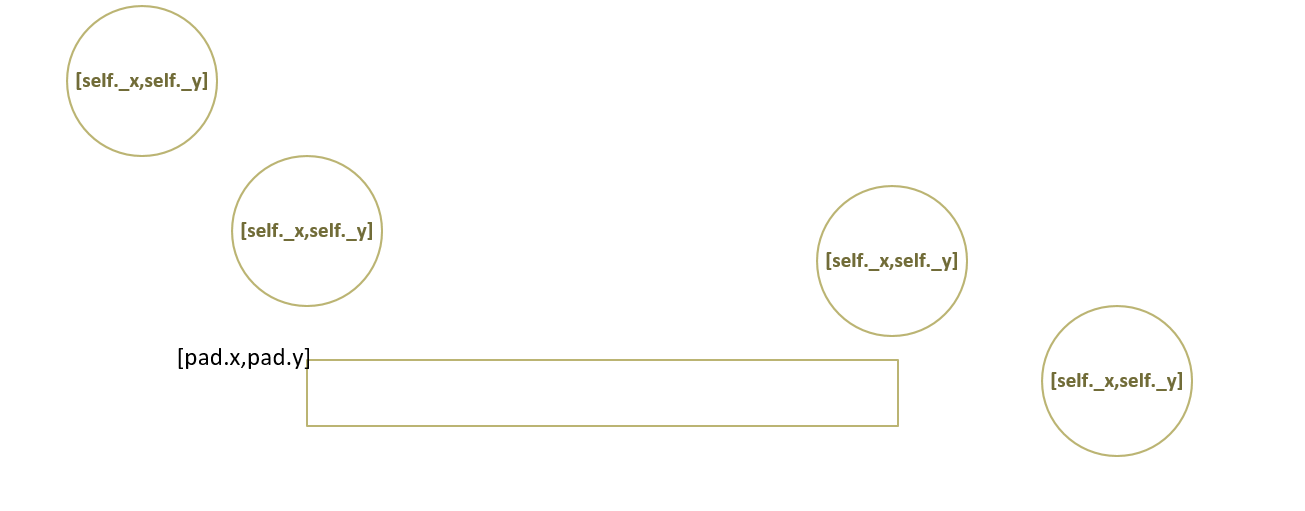
        self.\_dx = s[0]  # random.choice(ss) # random.randint(-5,5)

        self.\_dy = s[1]  # random.choice(ss) # random.randint(-5,5)

        #print ('self.\_dx,self.\_dy',self.\_dx,self.\_dy)

        # 绘制小球

        self.draw()

**在小球衣重新定位时，判断是否触碰球拍 如何做？？**

## 修改 完整游戏

定义一个函数 def hit\_paddle(self): #判断是否触碰 球拍

如果小球的X方向 在球拍范围内

同时 小球的Y方向 在球拍的高度范围内

def hit\_paddle(self):  # 声明函数，以供调用

        # 园心的X坐标落在拍上

        if ((self.\_x >= self.paddle.\_x) and

                (self.\_x <= self.paddle.\_x+self.paddle.width)):

            # 园心的Y + 半径 坐标落在球拍上的 Y坐标 +self.paddle.height

            if ((self.\_y + self.\_r >= self.paddle.\_y) and (self.\_y + self.\_r <= self.paddle.\_y+self.paddle.height)):

                # 比较小球y轴是否在球拍y轴内

                return True  # 表示小球碰到了球拍

        return False  # 表示小球没有碰到球拍

修改小球的 go () 函数 ,增加判断是否触碰 球拍

增加属性hit\_bottom 表示小球是否碰到画布底端,

初值为False

碰到画布底端则返回hit\_bottom为True

# 小球碰到了球拍，则改变Y轴方向向上运动

        if self.hit\_paddle() == True:

            print ('hit\_paddle() == True')

            self.\_dy = -self.\_dy

        if self.\_y + self.\_r >= self.height:  # 如果小球碰到画布底端 则返回hit\_bottom为True

            self.hit\_bottom = True

# 从球拍到坦克

**球拍 => 矩形 => 坦克**

## 代码一

from tkinter import \*

import random

Tank\_Left = None

Tank\_Right = None

class Tank():

    '''

    定义球拍

    '''

    def \_\_init\_\_(self, canvas, x=100, y=250, color='blue'):

        # 保存画板

        self.canvas = canvas

        # 保存小球颜色

        self.color = color

        # 保存左上角坐标

        self.\_x = x

        self.\_y = y

        # 球拍宽度

        self.width = 48

        self.height = 48

        #X方向的步长

        self.Step = 5

        self.\_dx = self.Step

        # 获取画布的宽度

        self.CancasWidth = self.canvas.winfo\_width()

        # 画板绑定键盘的任何事件

        self.canvas.bind\_all('<Key>', self.AnyKey)

        self.draw()

        self.StepCount = 0

    def AnyKey(self, event):

        '''

        响应键盘事件

        '''

        # 如果 不是左右键 则不处理

        if (event.keysym not in ['Left', 'Right']):

            return

        # 计算新左上角坐标 和  X方向的步长

        if event.keysym == 'Right' :

            self.\_dx = self.Step

            # 获取图形项 gettags(self, \*args)：获取指定图形项的所有tag。

            # 修改图形项  itemconfig(item, \*\*options)

            #-- 修改指定 item 的选项的当前值

            #-- item 可以是单个画布对象的 ID，也可以是某个 Tag

            if (self.StepCount == 0):

                self.canvas.itemconfig(self.id,image = Tank\_Right)

            else :

                self.canvas.itemconfig(self.id,image = Tank\_Right\_1)

            # 如果 已经到达 右边

            if (self.\_x+self.width > self.CancasWidth):

                self.\_x = self.CancasWidth -self.width

                #不动

                self.\_dx = 0

        elif event.keysym == 'Left' :

            #print ('event.keysym == Left',self.\_dx ,self.Step)

            if (self.StepCount == 0):

                self.canvas.itemconfig(self.id,image = Tank\_Left)

            else:

                self.canvas.itemconfig(self.id,image = Tank\_Left\_1)

            self.\_dx = (-1) \*self.Step

            # 如果 已经到达 左边

            if (self.\_x < 0):

                self.\_x = 0

                #不动

                self.\_dx = 0

        #交替更换 两幅图

        self.StepCount = (1 if self.StepCount ==0 else 0)

        #print (' self.StepCount' ,self.StepCount)

        #print (event.keysym,self.\_x ,self.\_dx)

#新坐标

        self.\_x += self.\_dx

        self.move()

    def draw(self):

        #self.id = self.canvas.create\_rectangle(

        #    self.\_x, self.\_y, self.\_x+self.width, self.\_y+self.height, fill=self.color)

        self.id = self.canvas.create\_image((self.\_x,self.\_y),image = Tank\_Left)

    def move(self):

        self.canvas.moveto(self.id, self.\_x, self.\_y)

if \_\_name\_\_ == '\_\_main\_\_':

    win = Tk()

    win.geometry('600x400+100+100')

    Tank\_Left = PhotoImage(file='Image\\Yellow\_Left\_0.png')

    Tank\_Left\_1 = PhotoImage(file='Image\\Yellow\_Left\_1.png')

    Tank\_Right = PhotoImage(file='Image\\Yellow\_Right\_0.png')

    Tank\_Right\_1= PhotoImage(file='Image\\Yellow\_Right\_1.png')

    mycanvas = Canvas(win, width=300, height=200, bg='white')

    mycanvas.pack(fill='both', expand=True)

    win.update()

    tank = Tank(mycanvas, x=100, y=250,color ='red')

    win.mainloop()

## 重点：

1. Tkinter 图像获取 PhotoImage(file='Image\\Yellow\_Left\_0.png')
2. 动态修改 canvas中的图像对象 itemconfig(item, \*\*options)

# 获取图形项 gettags(self, \*args)：获取指定图形项的所有tag。

    # 修改图形项  itemconfig(item, \*\*options)

    #-- 修改指定 item 的选项的当前值

    #-- item 可以是单个画布对象的 ID，也可以是某个 Tag

## 代码 优化 X方向

from tkinter import \*

import random

Tank\_Left = None

Tank\_Right = None

class Tank():

    '''

    定义球拍

    '''

    def \_\_init\_\_(self, canvas, x=100, y=250, color='blue'):

        # 保存画板

        self.canvas = canvas

        # 保存小球颜色

        self.color = color

        # 保存左上角坐标

        self.\_x = x

        self.\_y = y

        # 球拍宽度

        self.width = 48

        self.height = 48

        #X方向的步长

        self.Step = 5

        self.\_dx = self.Step

        # 获取画布的宽度

        #self.CancasWidth = self.canvas.winfo\_width()

        self.draw()

        # 画板绑定键盘的任何事件

        #self.canvas.bind(self.id,'<Key>', self.AnyKey)

        self.canvas.bind\_all('<Key>', self.AnyKey)

        self.StepCount = 0

    @property

    def CancasWidth(self):

        '''

        取画布宽度

        '''

        return self.canvas.winfo\_width()

    def AnyKey(self, event):

        '''

        响应键盘事件

        '''

        # 如果 不是左右键 则不处理

        if (event.keysym in ['Left', 'Right']):

            # if 三元表达式

            #更换 两幅图

            self.StepCount = 1 if self.StepCount == 0 else 0

            # 计算新左上角坐标 和  X方向的步长

            if event.keysym == 'Right' :

                self.\_dx = self.Step

                # 如果 已经到达 右边

                if (self.\_x+self.width > self.CancasWidth):

                    self.\_x = self.CancasWidth -self.width

                    #不动

                    self.\_dx = 0

            elif event.keysym == 'Left' :

                #print ('event.keysym == Left',self.\_dx ,self.Step)

                self.\_dx = (-1) \*self.Step

                # 如果 已经到达 左边

                if (self.\_x < 0):

                    self.\_x = 0

                    #不动

                    self.\_dx = 0

            #print (' self.StepCount' ,self.StepCount)

            #print (event.keysym,self.\_x ,self.\_dx)

            # 获取图形项 gettags(self, \*args)：获取指定图形项的所有tag。

            # 修改图形项  itemconfig(item, \*\*options)

            #-- 修改指定 item 的选项的当前值

            #-- item 可以是单个画布对象的 ID，也可以是某个 Tag

            self.canvas.itemconfig(self.id,image = eval('Tank\_' + event.keysym +'\_'+str(self.StepCount) ))

            self.\_x += self.\_dx

            self.move()

    def draw(self):

        #self.id = self.canvas.create\_rectangle(

        #    self.\_x, self.\_y, self.\_x+self.width, self.\_y+self.height, fill=self.color)

        self.id = self.canvas.create\_image((self.\_x,self.\_y),image = Tank\_Left)

    def move(self):

        #print (self.id, self.\_x, self.\_y)

        self.canvas.moveto(self.id, self.\_x, self.\_y)

if \_\_name\_\_ == '\_\_main\_\_':

    win = Tk()

    win.geometry('600x400+100+100')

    Tank\_Left\_0 = PhotoImage(file='Image\\Yellow\_Left\_0.png')

    Tank\_Left\_1 = PhotoImage(file='Image\\Yellow\_Left\_1.png')

    Tank\_Right\_0 = PhotoImage(file='Image\\Yellow\_Right\_0.png')

    Tank\_Right\_1= PhotoImage(file='Image\\Yellow\_Right\_1.png')

    mycanvas = Canvas(win, width=300, height=200, bg='white')

    mycanvas.pack(fill='both', expand=True)

    win.update()

    tank = Tank(mycanvas, x=100, y=250,color ='red')

    #tank1 = Tank(mycanvas, x=100, y=150,color ='red')

    win.mainloop()

## 代码 优化 XY方向

from tkinter import \*

import random

Tank\_Left = None

Tank\_Right = None

class Tank():

    '''

    定义坦克

    '''

    def \_\_init\_\_(self, canvas, x=100, y=250, color='blue'):

        # 保存画板

        self.canvas = canvas

        # 保存小球颜色

        self.color = color

        # 保存左上角坐标

        self.\_x = x

        self.\_y = y

        # 球拍宽度

        self.width = 48

        self.height = 48

        #X方向的步长

        self.Step = 5

        self.\_dx = self.Step

        self.\_dy = self.Step

        # 获取画布的宽度

        #self.CancasWidth = self.canvas.winfo\_width()

        self.draw()

        # 画板绑定键盘的任何事件

        #self.canvas.bind(self.id,'<Key>', self.AnyKey)

        self.canvas.bind\_all('<Key>', self.AnyKey)

        self.StepCount = 0

    @property

    def CancasWidth(self):

        '''

        取画布宽度

        '''

        return self.canvas.winfo\_width()

    @property

    def CancasHeight(self):

        '''

        取画布高度

        '''

        return self.canvas.winfo\_height()

    def AnyKey(self, event):

        '''

        响应键盘事件

        '''

        # 如果 不是左右键 则不处理

        if (event.keysym in ['Left', 'Right','Up','Down']):

            # if 三元表达式

            #更换 两幅图

            self.StepCount = 1 if self.StepCount == 0 else 0

            # 计算新左上角坐标 和  X方向的步长

        #左右方向

            if event.keysym == 'Right' :

                self.\_dx = self.Step

                self.\_dy = 0

                # 如果 已经到达 右边

                if (self.\_x+self.width > self.CancasWidth):

                    self.\_x = self.CancasWidth -self.width

                    #不动

                    self.\_dx = 0

            elif event.keysym == 'Left' :

                #print ('event.keysym == Left',self.\_dx ,self.Step)

                self.\_dx = (-1) \*self.Step

                self.\_dy = 0

                # 如果 已经到达 左边

                if (self.\_x < 0):

                    self.\_x = 0

                    #不动

                    self.\_dx = 0

        #上下方向

            if event.keysym == 'Down' :

                self.\_dy = self.Step

                self.\_dx = 0

                # 如果 已经到达 右边

                if (self.\_y+self.height > self.CancasHeight):

                    self.\_y = self.CancasHeight -self.height

                    #不动

                    self.\_dy = 0

            elif event.keysym == 'Up' :

                #print ('event.keysym == Left',self.\_dx ,self.Step)

                self.\_dy = (-1) \*self.Step

                self.\_dx = 0

                # 如果 已经到达 左边

                if (self.\_y < 0):

                    self.\_y = 0

                    #不动

                    self.\_dy = 0

            # 获取图形项 gettags(self, \*args)：获取指定图形项的所有tag。

            # 修改图形项  itemconfig(item, \*\*options)

            #-- 修改指定 item 的选项的当前值

            #-- item 可以是单个画布对象的 ID，也可以是某个 Tag

            self.canvas.itemconfig(self.id,image = eval('Tank\_' + event.keysym +'\_'+str(self.StepCount) ))

            self.\_x += self.\_dx

            self.\_y += self.\_dy

            self.move()

    def draw(self):

        #self.id = self.canvas.create\_rectangle(

        #    self.\_x, self.\_y, self.\_x+self.width, self.\_y+self.height, fill=self.color)

        self.id = self.canvas.create\_image((self.\_x,self.\_y),image = Tank\_Left)

    def move(self):

        #print (self.id, self.\_x, self.\_y)

        self.canvas.moveto(self.id, self.\_x, self.\_y)

if \_\_name\_\_ == '\_\_main\_\_':

    win = Tk()

    win.geometry('600x400+100+100')

    Tank\_Left\_0 = PhotoImage(file='Image\\Yellow\_Left\_0.png')

    Tank\_Left\_1 = PhotoImage(file='Image\\Yellow\_Left\_1.png')

    Tank\_Right\_0 = PhotoImage(file='Image\\Yellow\_Right\_0.png')

    Tank\_Right\_1= PhotoImage(file='Image\\Yellow\_Right\_1.png')

    Tank\_Up\_0 = PhotoImage(file='Image\\Yellow\_Up\_0.png')

    Tank\_Up\_1 = PhotoImage(file='Image\\Yellow\_Up\_1.png')

    Tank\_Down\_0 = PhotoImage(file='Image\\Yellow\_Down\_0.png')

    Tank\_Down\_1= PhotoImage(file='Image\\Yellow\_Down\_1.png')

    mycanvas = Canvas(win, width=300, height=200, bg='white')

    mycanvas.pack(fill='both', expand=True)

    win.update()

    tank = Tank(mycanvas, x=100, y=250,color ='red')

    #tank1 = Tank(mycanvas, x=100, y=150,color ='red')

    win.mainloop()

# PILlow 库

切分图像

from PIL import Image

#im = Image.open("Images\Login\_00.png")

im = Image.open("SmallBall\\Image\\Tank\_Yellow.png")

'''

print(im.size) ## 打印出尺寸信息

print(im.mode)   ## 打印出模式信息

print(im.format) ## 打印出格式信息

print(im.palette)

print(im.info)

'''

fix = 'Yellow\_'

Dir = ['Up','Down','Left','Right']

for row in range(4):

    for col in range(2):

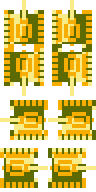
        box = (col\*48, row\*48, (col+1)\*48-1, (row+1)\*48-1)              ##确定拷贝区域大小

        region = im.crop(box)                   ##将im表示的图片对象拷贝到region中，大小

        filename = fix +Dir[row] +'\_'+str(col) + '.png'

        print (filename)

        region.save(filename)





# 场地测试一

from tkinter import \*

img\_Brick = None

img\_Iron= None

Tank\_Left = None

Tank\_Right = None

class Tank():

    '''

    定义坦克

    '''

    def \_\_init\_\_(self, canvas, x=100, y=250, color='blue'):

        # 保存画板

        self.canvas = canvas

        # 保存小球颜色

        self.color = color

        # 保存左上角坐标

        self.\_x = x

        self.\_y = y

        # 球拍宽度

        self.width = 48

        self.height = 48

        #X方向的步长

        self.Step = 5

        self.\_dx = self.Step

        self.\_dy = self.Step

        # 获取画布的宽度

        #self.CancasWidth = self.canvas.winfo\_width()

        self.draw()

        # 画板绑定键盘的任何事件

        #self.canvas.bind(self.id,'<Key>', self.AnyKey)

        self.canvas.bind\_all('<Key>', self.AnyKey)

        self.StepCount = 0

    @property

    def CancasWidth(self):

        '''

        取画布宽度

        '''

        return self.canvas.winfo\_width()

    @property

    def CancasHeight(self):

        '''

        取画布高度

        '''

        return self.canvas.winfo\_height()

    def AnyKey(self, event):

        '''

        响应键盘事件

        '''

        # 如果 不是左右键 则不处理

        if (event.keysym in ['Left', 'Right','Up','Down']):

            # if 三元表达式

            #更换 两幅图

            self.StepCount = 1 if self.StepCount == 0 else 0

            # 计算新左上角坐标 和  X方向的步长

        #左右方向

            if event.keysym == 'Right' :

                self.\_dx = self.Step

                self.\_dy = 0

                # 如果 已经到达 右边

                if (self.\_x+self.width > self.CancasWidth):

                    self.\_x = self.CancasWidth -self.width

                    #不动

                    self.\_dx = 0

            elif event.keysym == 'Left' :

                #print ('event.keysym == Left',self.\_dx ,self.Step)

                self.\_dx = (-1) \*self.Step

                self.\_dy = 0

                # 如果 已经到达 左边

                if (self.\_x < 0):

                    self.\_x = 0

                    #不动

                    self.\_dx = 0

        #上下方向

            if event.keysym == 'Down' :

                self.\_dy = self.Step

                self.\_dx = 0

                # 如果 已经到达 右边

                if (self.\_y+self.height > self.CancasHeight):

                    self.\_y = self.CancasHeight -self.height

                    #不动

                    self.\_dy = 0

            elif event.keysym == 'Up' :

                #print ('event.keysym == Left',self.\_dx ,self.Step)

                self.\_dy = (-1) \*self.Step

                self.\_dx = 0

                # 如果 已经到达 左边

                if (self.\_y < 0):

                    self.\_y = 0

                    #不动

                    self.\_dy = 0

            # 获取图形项 gettags(self, \*args)：获取指定图形项的所有tag。

            # 修改图形项  itemconfig(item, \*\*options)

            #-- 修改指定 item 的选项的当前值

            #-- item 可以是单个画布对象的 ID，也可以是某个 Tag

            self.canvas.itemconfig(self.id,image = eval('Tank\_' + event.keysym +'\_'+str(self.StepCount) ))

            self.\_x += self.\_dx

            self.\_y += self.\_dy

            self.move()

    def draw(self):

        #self.id = self.canvas.create\_rectangle(

        #    self.\_x, self.\_y, self.\_x+self.width, self.\_y+self.height, fill=self.color)

        self.id = self.canvas.create\_image((self.\_x,self.\_y),image = Tank\_Left)

    def move(self):

        #print (self.id, self.\_x, self.\_y)

        self.canvas.moveto(self.id, self.\_x, self.\_y)

# 关卡一

def stage1(canvas ):

    for x in [2, 3, 6, 7, 18, 19, 22, 23]:

        for y in [2, 3, 4, 5, 6, 7, 8, 9, 10, 17, 18, 19, 20, 21, 22, 23]:

            canvas.create\_image((3 + x \* 24+12, 3 + y \* 24+12),image = img\_Brick)

    for x in [10, 11, 14, 15]:

        for y in [2, 3, 4, 5, 6, 7, 8, 11, 12, 15, 16, 17, 18, 19, 20]:

            canvas.create\_image((3 + x \* 24 +12, 3 + y \* 24+12),image = img\_Brick)

    for x in [4, 5, 6, 7, 18, 19, 20, 21]:

        for y in [13, 14]:

            canvas.create\_image((3 + x \* 24+12, 3 + y \* 24+12),image = img\_Brick)

    for x in [12, 13]:

        for y in [16, 17]:

            canvas.create\_image((3 + x \* 24+12, 3 + y \* 24+12),image = img\_Brick)

    for x, y in [(11, 23), (12, 23), (13, 23), (14, 23), (11, 24), (14, 24), (11, 25), (14, 25)]:

        canvas.create\_image((3 + x \* 24+12, 3 + y \* 24+12),image = img\_Brick)

    for x, y in [(0, 14), (1, 14), (12, 6), (13, 6), (12, 7), (13, 7), (24, 14), (25, 14)]:

        canvas.create\_image((3 + x \* 24+12, 3 + y \* 24+12),image = img\_Iron)

if \_\_name\_\_ == '\_\_main\_\_':

    win = Tk()

    win.geometry('630x630+100+100')

    mycanvas = Canvas(win, width=300, height=200, bg='white')

    mycanvas.pack(fill='both', expand=True)

    img\_Background = PhotoImage(file='Image\\background.png')

    mycanvas.create\_image((315,315),image = img\_Background)

    img\_Tree = PhotoImage(file='Image\\scene\\Tree.png')

    for x in range(0,48+4):

        for y in range(0,48+4):

            mycanvas.create\_image((3 + x \* 12 +6, 3 + y \* 12+6),image = img\_Tree)

    img\_Home = PhotoImage(file='Image\\home1.png')

    #mycanvas.create\_image((315,315),image = img\_Home)

    mycanvas.create\_image((315,630-3-24),image = img\_Home)

    img\_Brick = PhotoImage(file='Image\\scene\\brick.png')

    img\_Iron  = PhotoImage(file='Image\\scene\\Iron.png')

    stage1(mycanvas)

    Tank\_Left\_0 = PhotoImage(file='Image\\Yellow\_Left\_0.png')

    Tank\_Left\_1 = PhotoImage(file='Image\\Yellow\_Left\_1.png')

    Tank\_Right\_0 = PhotoImage(file='Image\\Yellow\_Right\_0.png')

    Tank\_Right\_1= PhotoImage(file='Image\\Yellow\_Right\_1.png')

    Tank\_Up\_0 = PhotoImage(file='Image\\Yellow\_Up\_0.png')

    Tank\_Up\_1 = PhotoImage(file='Image\\Yellow\_Up\_1.png')

    Tank\_Down\_0 = PhotoImage(file='Image\\Yellow\_Down\_0.png')

    Tank\_Down\_1= PhotoImage(file='Image\\Yellow\_Down\_1.png')

    #mycanvas.create\_image((0,0),image = img\_Background)

    win.update()

    tank = Tank(mycanvas, x=3+24, y=3+12\*48,color ='red')

    #tank1 = Tank(mycanvas, x=100, y=150,color ='red')

    win.mainloop()

