Snake with Pygame

By 第②组

Snake With Pygame

相信大家都玩过贪吃蛇，它是有着简单逻辑的一种街机游戏，我们今天想通过贪吃蛇这个例子来阐述怎样用pygame来做一个游戏。

基础结构和事件控制

这里先定义了一个snake类，它可以控制snake在屏幕的位置和移动的速度，我们还定义了它能做的动作。

```
class snake:
    x = 10
    y = 10
    speed = 2

    def moveRight(self):
        self.x = self.x + self.speed

    def moveLeft(self):
        self.x = self.x - self.speed

    def moveUp(self):
        self.y = self.y - self.speed

    def moveDown(self):
        self.y = self.y + self.speed
```

那么这时我们可以创建一个snake对象，并且能通过刚刚各种动作的方法来修改变量。我们将这些方法联系到事件上。

```
pygame.event.pump()
keys = pygame.key.get_pressed()

if (keys[K_RIGHT]):
   print "Right arrow pressed."
```

```
while( self._running ):
            pygame.event.pump()
            keys = pygame.key.get_pressed()

            if (keys[K_RIGHT]):
                self.player.moveRight()

            if (keys[K_LEFT]):
                self.player.moveLeft()

            if (keys[K_UP]):
                self.player.moveUp()

            if (keys[K_DOWN]):
                self.player.moveDown()

            if (keys[K_ESCAPE]):
                self._running = False
```

构造snake

我们开始控制这条有初始长度的蛇。这条蛇总是在移动，当按下对应的按键蛇能改变方向

```
class snake:
    x = 0
    y = 0
    speed = 32
    direction = 0

    def update(self):
        if self.direction == 0:
            self.x = self.x + self.speed
        if self.direction == 1:
            self.x = self.x - self.speed
        if self.direction == 2:
            self.y = self.y - self.speed
        if self.direction == 3:
            self.y = self.y + self.speed

    def moveRight(self):
        self.direction = 0

    def moveLeft(self):
        self.direction = 1

    def moveUp(self):
        self.direction = 2

    def moveDown(self):
        self.direction = 3
```

开始表现得有蛇的样子了，但是还没有基本长度。我们需要追踪蛇原先的位置并且能移动蛇的头，再把蛇画出来。

----see 2_1.py

游戏逻辑

逻辑规则

蛇吃了苹果，苹果要移动到新位置

蛇吃了苹果，长度增加

蛇撞到自己，游戏结束

创建一个苹果的类

```
class Apple:
    x = 0
    y = 0
    step = 44

    def __init__(self,x,y):
        self.x = x * self.step
        self.y = y * self.step

    def draw(self, surface, image):
        surface.blit(image,(self.x, self.y))
```

想知道蛇的位置是否与苹果的位置相匹配，我们必须做碰撞检测。这意味着正在验证蛇的坐标与相交苹果的坐标，当然也可以确定蛇和本身是否相撞

```
def isCollision(self,x1,y1,x2,y2,bsize):
    if x1 >= x2 and x1 <= x2 + bsize:
        if y1 >= y2 and y1 <= y2 + bsize:
            return True
    return False
```

Thanks