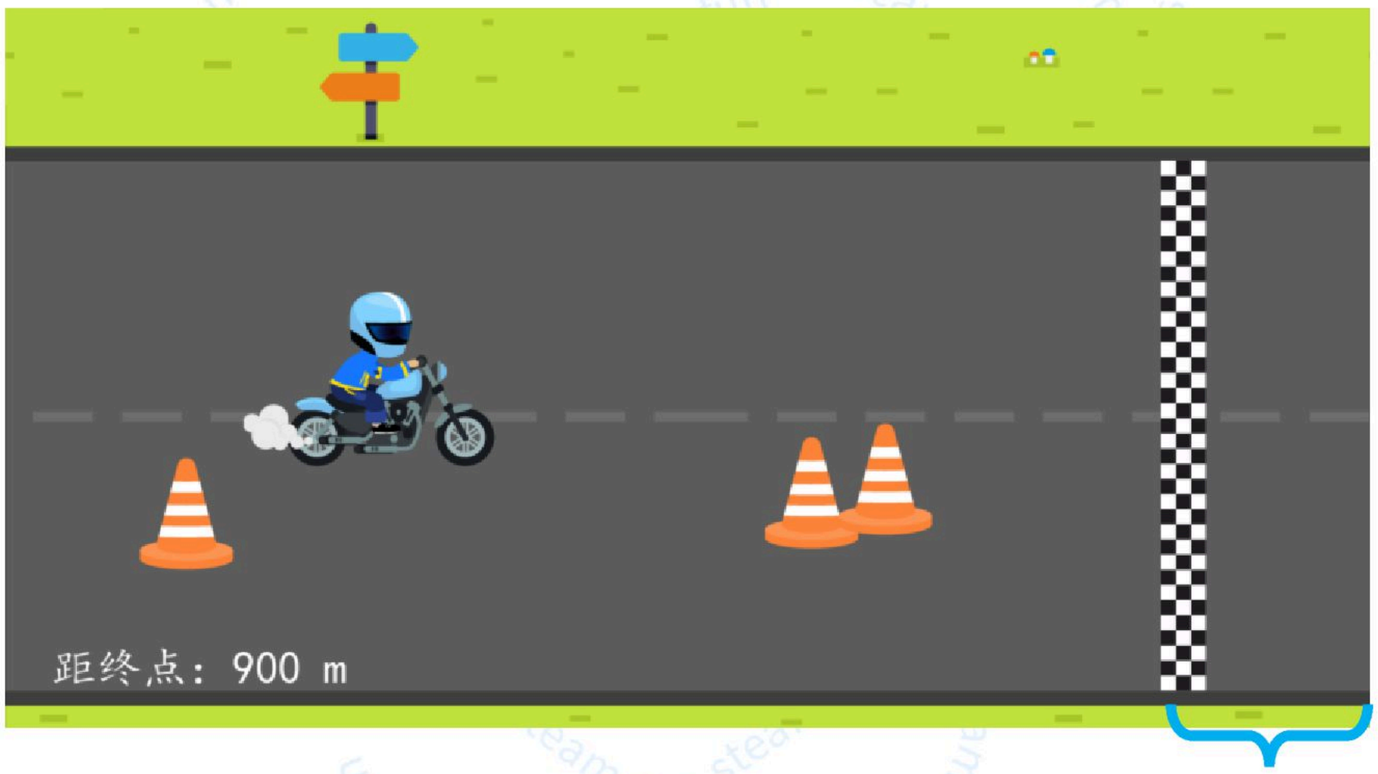
疯狂的摩托(二)

1.探索新知

1.1 最后冲刺

观察作品效果，摩托车什么时候开始冲刺的呢？



1．绘制终点线

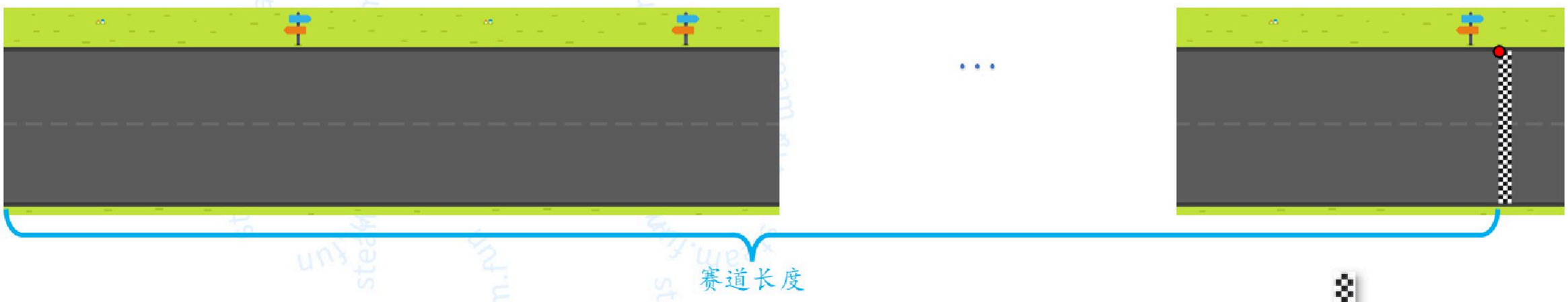
2．终点线随背景同步移动

3．【冲刺阶段】

背景等静止，摩托冲刺

假设终点线距离右边缘200时，进入【冲刺阶段】

绘制终点线



赛道长度自由设定，终点线左上角 topleft 的y坐标可以通过点击获取鼠标坐标的方式得到

终点线.png

1.1 最后冲刺

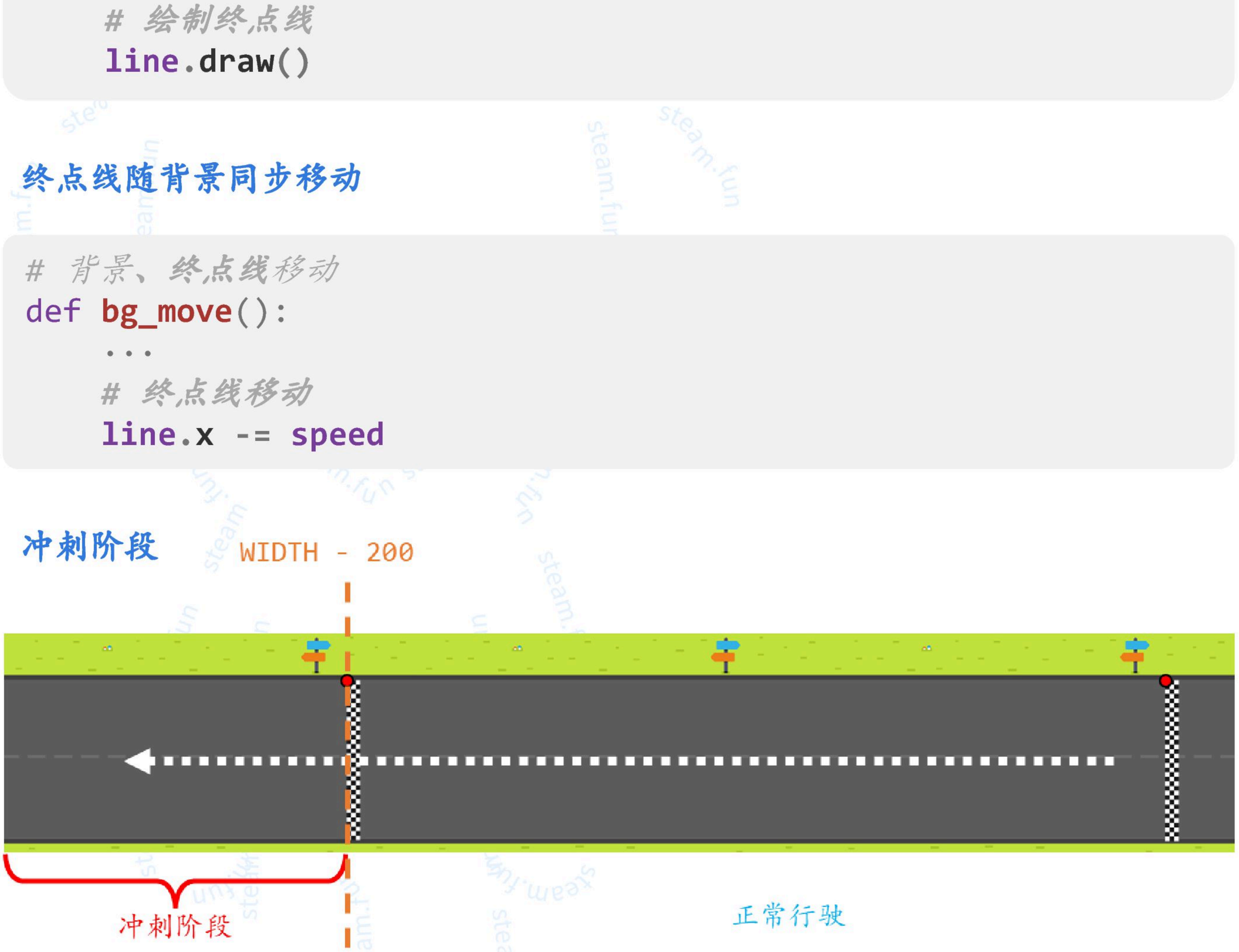
绘制终点线

```
# 创建终点线
line = Actor('终点线', topleft = (20000, 163))
# 初始化设置
speed = 15
...
def draw():
    ...
    # 绘制终点线
    line.draw()
```

终点线随背景同步移动

```
# 背景、终点线移动
def bg_move():
    ...
    # 终点线移动
    line.x -= speed
```

冲刺阶段




```
if line.x > WIDTH - 200:
    # <正常行驶>
else:
    # <冲刺阶段>
```

1.1 最后冲刺

冲刺阶段

```
def update():
    bg_move()
    obs_move()
```

```
# 最后冲刺
def line_collide():
    motor.x += 15

def update():
    if line.x > WIDTH - 200: # 正常行驶
        bg_move()
        obs_move()
    else:  # 最后冲刺
        line_collide()
```

新增代码：

```
# （新增）创建终点线
line = Actor('终点线', topleft = (20000, 163))
# 初始化设置
speed = 15
def draw():
    ...
    # （新增）绘制终点线
    line.draw()
def bg_move():
    ...
    # （新增）终点线移动
    line.x -= speed
...
# （新增）最后冲刺
def line_collide():
    motor.x += 15
# （修改）
def update():
    if line.x > WIDTH - 200:
        bg_move()
        obs_move()
    else:
        line_collide()  # 最后冲刺
    ...
```

1.2 碰撞效果

回忆：

我们学过什么检测碰撞的方法？

actor.collidepoint((x, y))　检测角色是否与指定点产生碰撞

- 检测角色是否与指定点产生碰撞
- 如果碰撞，返回 True ，否则返回 False

actor.colliderect(actor)　检测角色是否与指定 actor 矩形框产生碰撞

- 判断角色是否与指定 actor **矩形框**产生碰撞
- 如果碰撞，返回 True ，否则返回 False

摩托.colliderect(障碍物)

此时摩托车的矩形框和路障的矩形框是碰到的状态，返回值为True

思考：

如何侦测非透明像素部分的碰撞？

pixel:
像素

Actor.collide_pixel(actor)

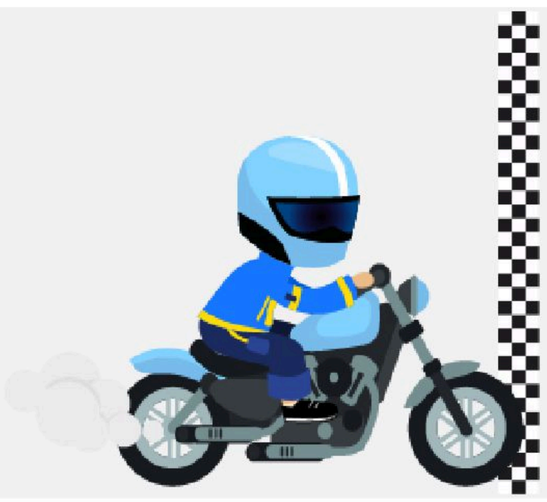
- 借助pgzhelper库，实现忽略透明区域的碰撞检测
- 检测一个角色对象的非透明部分是否和另一个角色对象的非透明部分重叠
- 如果是，返回一个元组(x,y)，碰撞点距离Actor左上角的x距离和y距离
- 否则返回None

1.2 碰撞效果

```
# 判断是否与障碍物相撞
def obs_collide():
    for obs in obstacles:
        if motor.collide_pixel(obs):
            # <游戏失败>

def update():
    ...
    # 是否碰撞障碍物
    obs_collide()
```

```
# 最后冲刺
def line_collide():
    if motor.collide_pixel(line):
        # <游戏成功>
    else:
        motor.x += 15
```



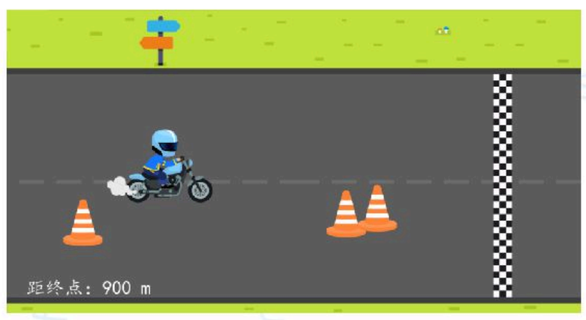
新增代码：

```
...
# （修改）最后冲刺
def line_collide():
    if motor.collide_pixel(line):
        # <游戏成功>
    else:
        motor.x += 15
# （新增）判断是否与障碍物相撞
def obs_collide():
    for obs in obstacles:
        if motor.collide_pixel(obs):
            # <游戏失败>
def update():
    ...
    # （新增）是否碰撞障碍物
    obs_collide()
```

1.3 成功失败

设置变量 flag 记录游戏状态—进行中、成功、失败
初始值为 **0** ，表示游戏在 **进行状态**。

flag	游戏状态
0	进行中
1	成功
2	失败



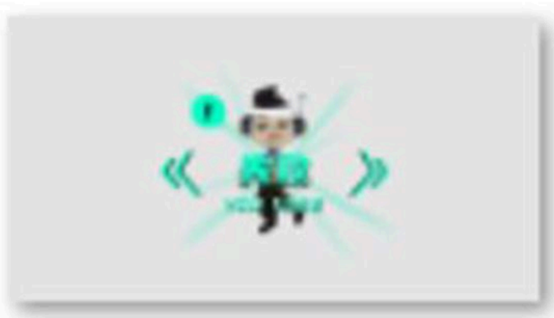
游戏进行中

flag = 0

摩托成功.png

成功条件： 赛车撞线

flag = 1



摩托失败.png

失败条件： 碰到路障

flag = 2

这个任务分为下面几步来完成：

1. 初始化游戏状态
2. 冲刺碰到终点线后，记录游戏成功状态
3. 碰到路障后，记录游戏失败状态
4. 绘制相应的背景

```
# 初始化设置
speed = 15
flag = 0
```

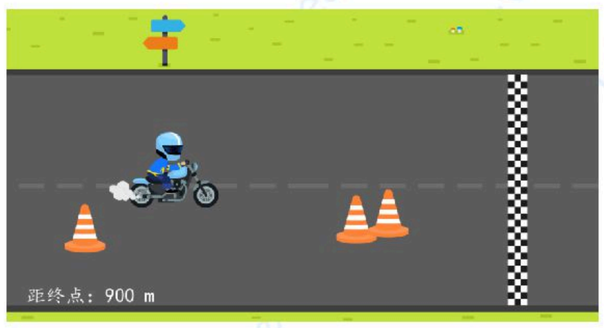
```
# 最后冲刺
def line_collide():
    global flag
    if motor.collide_pixel(line):
        flag = 1
    else:
        motor.x += 15
```

```
# 判断是否与障碍物相撞
def obs_collide():
    global flag
    for obs in obstacles:
        if motor.collide_pixel(obs):
            flag = 2
```

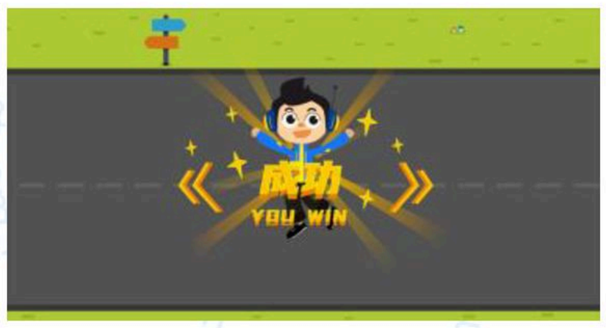
1.3 成功失败

绘制相应的背景

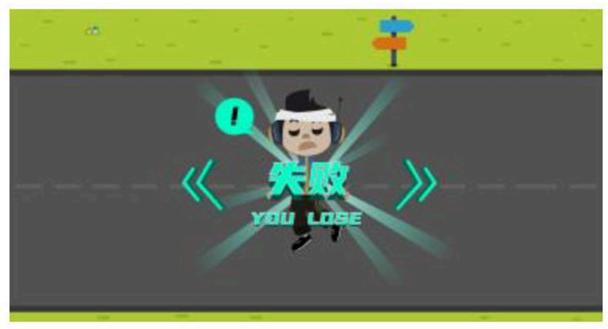
flag == 0



flag == 1



flag == 2




```
def draw():
    # 绘制连续背景
    bg1.draw()
    bg2.draw()
    if flag == 1:
        screen.blit('摩托成功', (0,0)
    elif flag == 2:
        screen.blit('摩托失败', (0,0)
    else:
        # 绘制障碍物
        for obs in obstacles:
            obs.draw()
        # 绘制摩托
        motor.draw()
        # 绘制终点线
        line.draw()
```

1.3 成功失败

新增代码：

```
...
# 初始化设置
speed = 15
flag = 0
def draw():
    # 绘制连续背景
    bg1.draw()
    bg2.draw()
    if flag == 1:
        screen.blit('摩托成功', (0,0))
    elif flag == 2:
        screen.blit('摩托失败', (0,0))
    else:
        # 绘制障碍物、摩托、终点线
        ...
...
# 最后冲刺
def line_collide():
    global flag
    if motor.collide_pixel(line):
        flag = 1
    else:
        motor.x += 15
# 判断是否与障碍物相撞
def obs_collide():
    global flag
    for obs in obstacles:
        if motor.collide_pixel(obs):
            flag = 2
...
```

《疯狂的摩托》基本效果已经实现，自己运行作品，一起聊聊看，还有哪些可以优化的地方呢？

1.4 效果优化

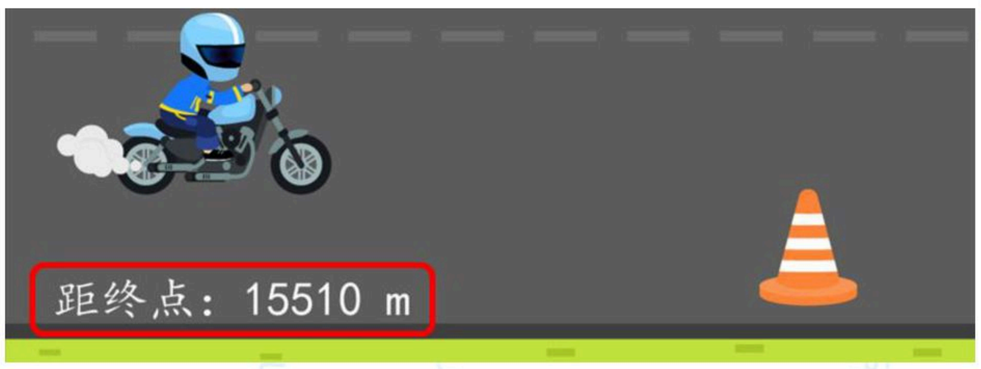
接下来我们进行三点优化：

1. 游戏结束后，背景等停止滚动
2. 设置文字效果
3. 添加背景音乐、胜利、失败音效

只有 flag == 0 的时候，也就是游戏进行中，背景等才会滚动，
我们可以给背景等滚动的代码设置一个执行条件：

```
def update():
    if flag == 0:
        if line.x > WIDTH - 200:
            bg_move()
            obs_move()
        else:
            line_collide()  # 最后冲刺
        # 控制赛车移动，限制不能出界
        ...
        # 是否碰撞障碍物
        obs_collide()
```

【设置文字效果】




```
x = line.x - motor.x
```

左下角文字呈现的是摩托车到终点的距离，该如何表示呢？

【绘制距离文字】

用到的字体需放在【fonts】文件夹中，本节课准备的是楷体

1.4 效果优化

```
def draw():
    ...
    # 绘制距离文字
    x = int(line.x - motor.x)     # 保留距离的整数部分
    screen.draw.text(f'距终点: {x} m', (50,700), fontname = 'simkai', fontsize = 50)
```

声音分为音效【sounds】和音乐【music】两种，我们将音乐文件放置在不同的文件夹内

一般情况下，

背景音乐用 music 的方法；

音效使用 sounds 的方法


music


sounds

【背景音乐】


music


速度与激情.mp3

播放背景音乐“速度与激情”

如果游戏胜利或者失败，就停止播放背景音乐

指令手册

音乐文件必须放在 music 文件夹内

music.play(name)

- 播放给定的音乐，循环播放
- 将替换当前播放的音乐，取消之前音乐排队序列

music.stop()

- 用于停止播放当前音乐

1.4 效果优化

```
def draw():
    # 绘制连续背景
    ...
    if flag == 1:
        music.stop()
        screen.blit('摩托成功', (0,0))
    elif flag == 2:
        music.stop()
        screen.blit('摩托失败', (0,0))
    else:
        ...
...
music.play('速度与激情')
pgzrun.go()
```

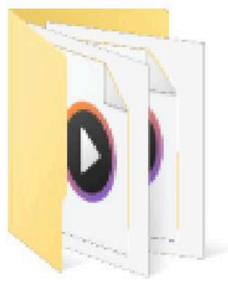
【胜利失败音效】

指令手册

音效文件必须放在 sounds 文件夹内

sounds.xxx.play()

- xxx 是音效文件名称
- 仅支持 .wav 音频格式



1.4 计时效果

```
# 最后冲刺
def line_collide():
    global flag
    if motor.collide_pixel(line):
        flag = 1
        sounds.胜利.play()
    else:
        motor.x += 15

# 判断是否与障碍物相撞
def obs_collide():
    global flag
    for obs in obstacles:
        if motor.collide_pixel(obs):
            flag = 2
            sounds.失败.play()
```

新增代码：

```
...
def draw():
    # 绘制连续背景
    ...
    if flag == 1:
        music.stop()
        screen.blit('摩托成功', (0,0))
    elif flag == 2:
        music.stop()
        screen.blit('摩托失败', (0,0))
    else:
        # 绘制障碍物、摩托、终点线
        ...
        # 绘制距离文字
        x = int(line.x - motor.x)
        screen.draw.text(f'距终点: {x} m', (50,700), fontname = 'simkai', fontsize = 50)
```

1.4 计时效果

新增代码：

```
...
# 最后冲刺
def line_collide():
    global flag
    if motor.collide_pixel(line):
        flag = 1
        sounds.胜利.play()
    else:
        motor.x += 15
# 判断是否与障碍物相撞
def obs_collide():
    global flag
    for obs in obstacles:
        if motor.collide_pixel(obs):
            flag = 2
            sounds.失败.play()
def update():
    if flag == 0 :
        if line.x > WIDTH - 200:
            bg_move()
            obs_move()
        else:
            line_collide()  # 最后冲刺
        # 控制赛车移动，限制不能出界
        ...
        # 是否碰撞障碍物
        obs_collide()
...

music.play('速度与激情')

pgzrun.go()
```

完整代码

```
import pgzrun
import random
from pgzhelper import *
# 游戏窗口大小
WIDTH = 1500
HEIGHT = 800
# 创建背景角色
bg1 = Actor('连续道路', topleft = (0,0))
bg2 = Actor('连续道路', topleft = (1500,0))
# 创建障碍物
obstacles = []
for _ in range(3):  # 障碍物数量
    x = random.randint(WIDTH, WIDTH + 1500)  # 障碍物出现在屏幕右侧外
    y = random.randint(100, HEIGHT - 50)
    obstacles.append(Actor('路障', (x, y)))
# 创建摩托车
motor = Actor('摩托1', (200, HEIGHT // 2))
# 将摩托造型存放在列表内
motor.images = ['摩托1','摩托2']
# 创建终点线
line = Actor('终点线', topleft = (20000, 163))
speed = 15  # 初始化速度
flag = 0    # 初始化游戏状态
def draw():
    # 绘制连续背景
    bg1.draw()
    bg2.draw()
    if flag == 1:
        music.stop()
        screen.blit('摩托成功', (0,0))
    elif flag == 2:
        music.stop()
        screen.blit('摩托失败', (0,0))
    else:
        # 绘制障碍物、摩托、终点线
        for obs in obstacles:
            obs.draw()
        motor.draw()
        line.draw()
        # 绘制距离文字
        x = int(line.x-motor.x)
        screen.draw.text(f'距终点: {x} m', (50,700), fontname = 'simkai', fontsize = 50)
```

完整代码

```
# 背景、终点线移动
def bg_move():
    bg1.x -= speed
    bg2.x -= speed
    if bg1.right <= 0:
        bg1.left = bg2.right
    if bg2.right <= 0:
        bg2.left = bg1.right
    line.x -= speed

# 更新障碍物位置
def obs_move():
    for obs in obstacles:
        # 障碍物随背景一起移动
        obs.x -= speed
        # 障碍物继续出现在屏幕外
        if obs.x < 0:
            obs.x = random.randint(WIDTH, WIDTH + 1500)
            obs.y = random.randint(100, HEIGHT - 50)

# 最后冲刺
def line_collide():
    global flag
    if motor.collide_pixel(line):
        flag = 1
        sounds.胜利.play()
    else:
        motor.x += 15

# 判断是否与障碍物相撞
def obs_collide():
    global flag
    for obs in obstacles:
        if motor.collide_pixel(obs):
            flag = 2
            sounds.失败.play()
```

完整代码

```
def update():
    if flag == 0 :
        if line.x > WIDTH - 200:
            bg_move()
            obs_move()
        else:
            line_collide()  # 最后冲刺
        # 控制赛车移动，限制不能出界
        if keyboard.up and motor.top > 0:
            motor.y -= 7
        if keyboard.down and motor.bottom < HEIGHT-50:
            motor.y += 7
        # 是否碰撞障碍物
        obs_collide()

# 切换摩托造型
def change_image():
    motor.next_image()
clock.schedule_interval(change_image,0.2)

music.play('速度与激情')

pgzrun.go()
```

2.强化练习

1. 当我们想要在屏幕上显示文字时，使用的函数是？（　　）

A. screen.show.text()

B. screen.draw.text()

C. screen.write.text()

D. screen.print.text()

2. 下面哪个方法可以检测鼠标是否点击角色？（　　）

A. actor.collidepoint((x, y))

B. actor.colliderect(actor)

C. actor.collide(actor)

D. actor.collide_pixel(actor)

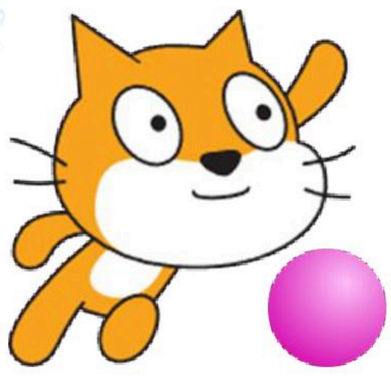
3. 如果用 ball.colliderect(cat) 来检测小球是否碰到了猫，当两者位置如下所示时，返回值是（　　）？

A. True

B. False

C. 一个元组

D. 报错



2.强化练习

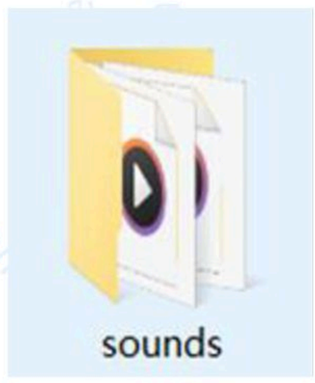
4．已知音效文件如图所示，如何播放该音效？（　　）

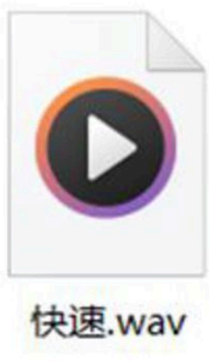
A. music.play(快速)

B. music.快速.play()

C. sounds.快速.wav.play()

D. sounds.快速.play()





5．想要用作背景音乐的文件，应该放在下面哪个文件夹中最合适？（　　）

A. 
images

B. 
fonts

C. 
music

D. 
sounds

3.术语箱

collide	碰撞
pixel	像素
music	音乐
sound	声音

4.课后挑战

游戏优化

要求如下：

1. 绘制‘钻石’角色
2. 钻石随背景同步移动，移出舞台后从右侧重新进入
3. 摩托碰到钻石后，钻石从右侧重新进入
4. 变量 score 记录收集到的钻石数量
5. 在窗口内，绘制文字‘收集宝石：×颗’