《Flappy Bird》项目报告

小组名称：Hakuna Matata

小组成员：刘雨林

余悦杨

许馨艺

指导教师：鲍杨

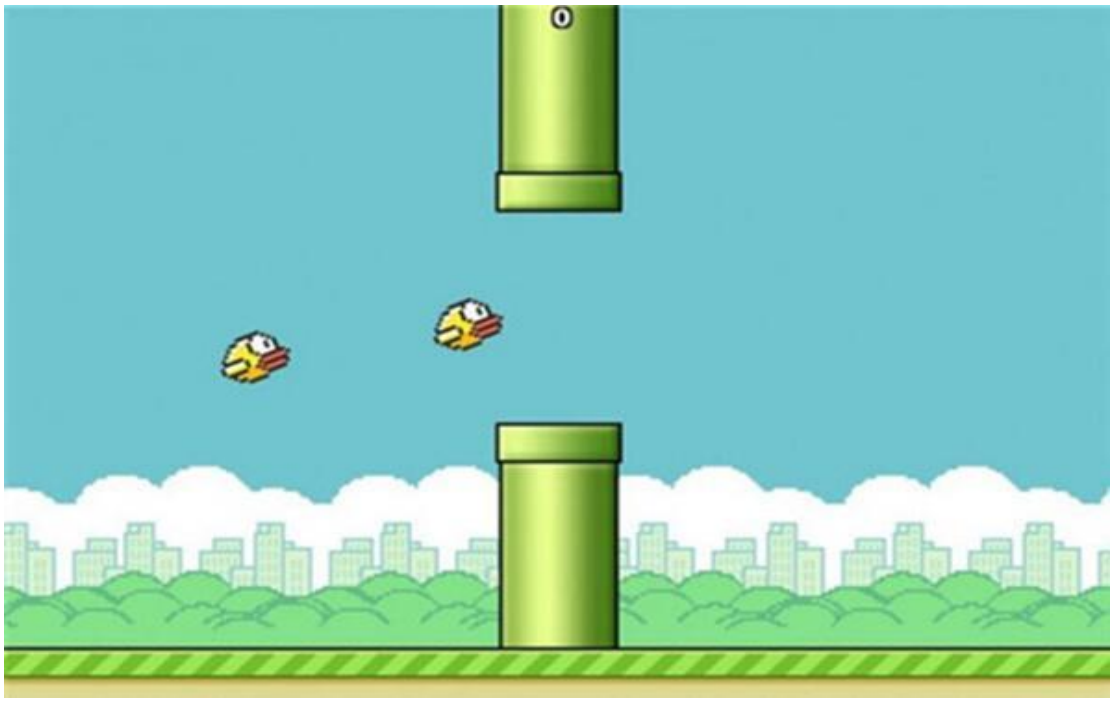
2020/06/08

目录

CS159 程序设计课程

一、课题背景

我们所创建的小游戏程序灵感来源于知名游戏《Flappy Bird》。《Flappy Bird》是一款由越南的独立游戏开发者 Dong Nguyen 开发的作品。游戏中玩家必须控制一只小鸟，跨越由各种不同长度水管所组成的障碍。在《Flappy Bird》这款游戏中，玩家只需要用一根手指来操控，点击触摸屏幕（或按空格键），则小鸟向上飞，不断的点击就会不断的向高处飞。放松手指，则会快速下降。所以玩家若要控制小鸟一直向前飞行，就需要注意躲避途中高低不平的管子。游戏的得分规则为：小鸟安全从上下一组管子间隙穿过得 1 分，如果小鸟撞上柱子，则游戏结束。我们小组在原版小游戏《Flappy Bird》的基础上，更改了一部分游戏规则，在一定程度上增加了游戏通关的难度，同时也增加了游戏的趣味性和新颖度。



二、游戏介绍

我们想设计一款横板 2D 像素风小游戏，主要内容为：首先，运行代码，摄像头开启，游戏玩家在计算机视野中框选其将要追踪的对象(例如头部、鼻子、手等部位)，完成框选后，按下空格键，游戏开始。游戏玩家通过将所追踪物体上下移动的方式，在控制游戏界面上的游戏主角（一只小鸟）向上飞和向下飞的同时躲避出现的障碍物（无数根长短不一的管子）。在游戏界面上，障碍物不停地向左移动，以此达到小鸟向右飞的效果。当小鸟安全通过一组管道，游戏玩家得 1 分；如果小鸟撞到障碍物则死亡，此时游戏结束。系统会自动记录小鸟飞过的管道个数，并将其作为游戏的最终得分。

三、理论模型

小组在设计改编版本的《Flappy Bird》小游戏时，使用到 Python 中的 Pygame 库和 OpenCV。

1、Pygame

Pygame 是一个利用 SDL 库的写就的游戏库，是一组用来开发游戏软件的 Python 程序模块。Pygame 允许在 Python 程序中创建功能丰富的游戏和多媒体程序，是一个高可移植性的模块可以支持多个操作系统，非常适合用于小游戏的开发。

2、OpenCV

OpenCV 的全称是 Open Source Computer Vision Library，是一个基于 BSD 许可(开源)发行的跨平台计算机视觉库，可以在 Linux、Windows 和 Mac OS 操作系统上运行，实现了图像处理和计算机视觉方面的很多通用算法，其应用领域广

泛，如人机互动、物体识别、图像分割、人脸识别等。小组游戏的制作基于其物体追踪的功能。

四、代码中的思路与流程

4.1 插入库、常量等

我们使用 Pygame 作为小游戏的主要制作工具，同时，为了完成物体追踪的相关内容，我们导入 cv2 库。

```
import sys
import random
import pygame
import cv2
```

4.2 设置游戏中需要的各种变量

分别设置游戏的屏幕宽高、游戏中的障碍物(管道)的宽高、上下两个管道之间的空隙、小鸟自身的宽高。设置地面高度，游戏的有效高度为屏幕高度-地面高度。

```
# FPS
FPS = 30
# 屏幕宽高
SCREEN_WIDTH = 512
SCREEN_HEIGHT = 512
# 管道宽高
PIPE_WIDTH = 50
PIPE_HEIGHT = 300
# 管道之间空隙
PIPE_GAP_SIZE = 150
# 小鸟
BIRD_WIDTH = 15
BIRD_HEIGHT = 12
# 地面高度
FLOOR_HEIGHT = 40
# 游戏有效高度
BASE_HEIGHT = SCREEN_HEIGHT - FLOOR_HEIGHT
```

4.3 选定追踪物体

参照老师提供的物体追踪部分代码，实现计算机摄像头捕捉游戏启动时的画

面，以及追踪区域的自主框选。我们返回所框选矩形的中心坐标实现与游戏的对接，设定在中心坐标超过屏幕上方四分之一区域时小鸟向上飞翔。

```
class Tracker:
    def __init__(self,video_file):
        # set up tracker: https://docs.opencv.org/3.4.0/d0/d0a/classcv_1_1Tracker.html
        # tracker = cv2.TrackerMIL_create()
        tracker = cv2.TrackerBoosting_create()
        # read video
        video = cv2.VideoCapture(video_file)
        # exit if video not opened.
        if not video.isOpened():
            print("Could not open video")
            sys.exit()
        # read first frame
        ok, frame = video.read()
        frame = cv2.flip(frame, 1)
        if not ok:
            print("Cannot read video file")
            sys.exit()
        # define an initial bounding box
        # bbox = (287, 23, 86, 320)
        bbox = cv2.selectROI(frame, False)
        # initialize tracker with first frame and bounding box
        ok = tracker.init(frame, bbox)
        self.initCam1=[ok, bbox, frame, tracker, video]
        cv2.destroyWindow("ROI selector")
    def roundTrack(self):
        ok, bbox, frame, tracker, video = self.initCam1
        # read a new frame
        ok, frame = video.read()
        frame = cv2.flip(frame, 1)

        # start timer
        timer = cv2.getTickCount()
        # update tracker
        ok, bbox = tracker.update(frame)
        # calculate Frames per second (FPS)
        fps = cv2.getTickFrequency() / (cv2.getTickCount() - timer)
        center = (frame.shape[1]//2,frame.shape[0]//2)
        # draw bounding box
        if ok:
            # tracking success
            p1 = (int(bbox[0]), int(bbox[1]))
            p2 = (int(bbox[0] + bbox[2]), int(bbox[1] + bbox[3]))
            cv2.rectangle(frame, p1, p2, (255, 0, 0), 2, 1)
            center=(((int(bbox[0])+(int(bbox[0] + bbox[2])/2)),(int(bbox[1])+int(bbox[1] + bbox[3]))/2))
        else:
            # tracking failure
            cv2.putText(frame, "Tracking failure detected", (100, 80), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (0, 0, 255), 2)

        # display tracker type on frame
        cv2.putText(frame, " Tracker", (100, 20), cv2.FONT_HERSHEY_SIMPLEX, 0.75, (50, 170, 50), 2);

        # display result
        cv2.imshow("Tracking", frame)

        # Exit if ESC pressed
        k = cv2.waitKey(1) & 0xff
        print(center[1],frame.shape[1])
        if center[1]< frame.shape[1]//4:
            return True
        else:
            return False
```

4.4 设置小鸟的参数

我们使用了一个长方形来代表游戏的主角——小鸟，并且使用了 pygame 中的 sprite 基类。我们先设定了小鸟的初始位置，并且将初始的速度设置为 0。

```
class Bird(pygame.sprite.Sprite):
    def __init__(self, position):
        pygame.sprite.Sprite.__init__(self)
        self.rect = pygame.Rect(*position, BIRD_WIDTH, BIRD_HEIGHT)
        # 定义飞行变量
        self.is_flapped = False
        self.up_speed = 10
        self.down_speed = 0

        self.time_pass = FPS / 1000
```

为了表现出小鸟正在受到重力影响，我们使小鸟上升的速度越来越小，在上升速度为 0 时变为下降且下降的速度越来越大。

```
    # 更新小鸟的位置
    def update(self):
        # 判断小鸟是上升还是下降
        if self.is_flapped:
            # 上升速度越来越小
            self.up_speed -= 50 * self.time_pass
            self.rect.top -= self.up_speed
            # 上升速度小于等于0，改为下降状态
            if self.up_speed <= 0:
                self.down()
                self.up_speed = 10
                self.down_speed = 0
        else:
            # 下降速度越来越大
            self.down_speed += 30 * self.time_pass
            self.rect.bottom += self.down_speed
```

当小鸟碰撞到管道上边界或者下边界时，小鸟判定为死亡，游戏失败。

```
        # 判断小鸟是否撞到了边界死亡
        is_dead = False
        if self.rect.top <= 0:  # 上边界
            self.up_speed = 0
            self.rect.top = 0
            is_dead = True

        if self.rect.bottom >= BASE_HEIGHT:  # 下边界
            self.up_speed = 0
            self.down_speed = 0
            self.rect.bottom = BASE_HEIGHT
            is_dead = True
```

定义小鸟的上升状态和下落状态，以及其 rect 属性。

```
    # 下落状态
    def down(self):
        self.is_flapped = False

    # 上升状态
    def up(self):
        if self.is_flapped:
            self.up_speed = max(12, self.up_speed + 1)
        else:
            self.is_flapped = True

    def draw(self, screen):
        pygame.draw.rect(screen, (255, 255, 255), self.rect, 1)
```

4.5 设置管道的参数

管道类中定义了管道的 rect 属性，用于在画布上渲染管道，类中定义了一个静态方法，生成一组管道的上下两个 left 值和 top 值。

```
class Pipe(pygame.sprite.Sprite):
    def __init__(self, position):
        pygame.sprite.Sprite.__init__(self)
        left, top = position
        # 如果是下边的管道，通过定义管道高度，删除地面以下的管道
        pipe_height = PIPE_HEIGHT
        if top > 0:
            pipe_height = BASE_HEIGHT - top + 1
        self.rect = pygame.Rect(left, top, PIPE_WIDTH, pipe_height)
        # 用于计算分数
        self.used_for_score = False

    def draw(self, screen):
        pygame.draw.rect(screen, (255, 255, 255), self.rect, 1)

    def generate_pipe_position():
        # 生成上下两个管道的坐标
        top = int(BASE_HEIGHT * 0.2) + random.randrange(
            0, int(BASE_HEIGHT * 0.6 - PIPE_GAP_SIZE))
        return {
            'top': (SCREEN_WIDTH + 25, top - PIPE_HEIGHT),
            'bottom': (SCREEN_WIDTH + 25, top + PIPE_GAP_SIZE)
        }
```

4.6 初始化游戏

将游戏初始化，把小鸟和管道添加入游戏

```python
# 初始化游戏
def init_game():
    pygame.init()
    screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
    pygame.display.set_caption('Flappy Bird')
    return screen


# 初始化精灵
def init_sprite():
    # 小鸟类
    bird_position = [SCREEN_WIDTH * 0.2, (SCREEN_HEIGHT - BIRD_HEIGHT) / 3]
    bird = Bird(bird_position)
    # 管道类
    pipe_sprites = pygame.sprite.Group()
    for i in range(2):
        pipe_pos = Pipe.generate_pipe_position()
        # 添加上方的管道
        pipe_sprites.add(
            Pipe((SCREEN_WIDTH + i * SCREEN_WIDTH / 2,
                  pipe_pos.get('top')[-1])))
        # 添加下方的管道
        pipe_sprites.add(
            Pipe((SCREEN_WIDTH + i * SCREEN_WIDTH / 2,
                  pipe_pos.get('bottom')[-1])))
    return bird, pipe_sprites
```

4.7 碰撞和更新

检测小鸟和管道是否发生碰撞，若发生碰撞，判定小鸟死亡。

```python
# 精灵类碰撞检测和小鸟更新位置
def collision(bird, pipe_sprites):
    # 检测碰撞
    is_collision = False
    for pipe in pipe_sprites:
        if pygame.sprite.collide_rect(bird, pipe):
            is_collision = True

    # 更新小鸟
    is_dead = bird.update()
    if is_dead:
        is_collision = True

    return is_collision
```

4.8 移动管道实现小鸟往前飞的效果

初始管道坐标在屏幕右侧生成，当小鸟飞过一个管道，减少管道的 x 坐标值，因此屏幕上的管道不断向左移，从而达到小鸟一直向右飞的效果，与此同时，不在屏幕上的管道会被删除，因此标志位会不断更新。当小鸟成功飞过管道后获得加分。

```
# 移动pipe实现小鸟往前飞的效果
def move_pipe(bird, pipe_sprites, is_add_pipe, score):
    flag = False  # 下一次是否要增加新的pipe的标志位
    for pipe in pipe_sprites:
        pipe.rect.left -= 4
        # 小鸟飞过pipe 加分
        if pipe.rect.centerx < bird.rect.centerx and not pipe.used_for_score:
            pipe.used_for_score = True
            score += 0.5
        # 增加新的pipe
        if pipe.rect.left < 10 and pipe.rect.left > 0 and is_add_pipe:
            pipe_pos = Pipe.generate_pipe_position()
            pipe_sprites.add(Pipe(position=pipe_pos.get('top')))
            pipe_sprites.add(Pipe(position=pipe_pos.get('bottom')))
            is_add_pipe = False
        # 删除已不在屏幕的pipe, 更新标志位
        elif pipe.rect.right < 0:
            pipe_sprites.remove(pipe)
            flag = True
    if flag:
        is_add_pipe = True
    return is_add_pipe, score
```

4.9 分数和游戏结束提示

为了让玩家知道自己的得分，将分数显示于游戏页面上方。此外，当游戏失败时，需要提醒玩家游戏已经结束，需要开始下一局。

```python
# 画分数
def draw_score(screen, score):
    font_size = 32
    digits = len(str(int(score)))
    offset = (SCREEN_WIDTH - digits * font_size) / 2
    font = pygame.font.SysFont('Blod', font_size)
    screen.blit(font.render(str(int(score)), True, (255, 255, 255)),
                (offset, SCREEN_HEIGHT * 0.1))

# 画Game Over
def draw_game_over(screen, text):
    font_size = 24
    font = pygame.font.SysFont('arial', font_size)
    screen.blit(font.render(text, True, (255, 255, 255), (0, 0, 0)),
                (60, SCREEN_HEIGHT * 0.4))
```

4.10 操作方法

尽管小鸟本身是由镜头前的物体控制，但按键的设定仍然不可缺少。在这个游戏中，我们需要设定一个退出键以及重新开始游戏键来使游戏体验更加流畅。

```python
# 按键
def press(is_game_running, bird):
    for event in pygame.event.get():
        if event.type == pygame.QUIT:  # 点击关闭按钮退出
            pygame.quit()
            sys.exit()
        elif event.type == pygame.KEYDOWN:
            if event.key == 13 and not is_game_running:  # 游戏结束时回车键继续
                return True
```

当然，最重要的用于操作小鸟的“虚拟按键”自然是必不可少的。

```python
def visual_key(is_game_running,bird,tracker):
    up = tracker.roundTrack()
    if up:
        print(up)
        bird.up()
```

4.11 组合

在进行了如此多的定义之后，我们只需要把它们都组合起来，游戏便完成了。

```python
def main():
    screen = init_game()  # 初始化游戏
    bird, pipe_sprites = init_sprite()  # 初始化精灵
    clock = pygame.time.Clock()
    is_add_pipe = True  # 是否需要增加管道
    is_game_running = True  # 是否在游戏中
    score = 0  # 初始分数
    tracker = Tracker(0)
    while True:
        restart = press(is_game_running, bird)  # 按键
        visual_key(is_game_running,bird,tracker)
        if restart:
            return
        screen.fill((0, 0, 0))  # 填充背景
        is_collision = collision(bird, pipe_sprites)  # 碰撞检测
        if is_collision:
            is_game_running = False  # 如果碰撞 游戏结束
        if is_game_running:
            is_add_pipe, score = move_pipe(bird, pipe_sprites, is_add_pipe,
                                           score)  # 不碰撞 移动管道
        else:
            draw_game_over(screen, 'Press Enter To Start!')  # 游戏结束
        bird.draw(screen)  # 画鸟
        draw_score(screen, score)  # 画分数
        # 画地面
        pygame.draw.line(screen, (255, 255, 255), (0, BASE_HEIGHT),
                         (SCREEN_WIDTH, BASE_HEIGHT))
        # 画管道
        for pipe in pipe_sprites:
            pipe.draw(screen)

        # 更新画布
        pygame.display.update()
        clock.tick(FPS)

if __name__ == "__main__":
    while True:
        main()
```

五、有待改进的地方

1、我们设计的这款游戏没有背景音乐，可能会使游戏玩家的体验度下降，我们希望能够添加更多的音乐元素，以此来增强游戏的趣味性，提高游戏玩家的体验

程度。

2、游戏中小鸟、管道均用矩形替代，不如原版形象生动，我们希望可以增添图像和色彩使游戏界面更活泼有趣。

3、我们设计的这款游戏的难度层级较为单一，我们希望可以增加游戏的难度层级，分为简单、复杂、困难等层级，满足不同水平的游戏玩家的需求，增强了游戏的挑战性与趣味性。

4、小组成员最初设想是运用 openCV 的人脸识别功能，识别人脸或人脸中的某一部位进行游戏操控。但尝试之后发现人脸识别不够精确（也可能是小组自身技术原因），当矩形框消失后游戏无法继续。虽然权衡后采用了物体追踪，但我们仍希望能够利用其自动识别的功能，而非需要玩家手动框选追踪区域。

六、总结

小组成员通过对 Pygame 的学习，设计出了改编版的《Flappy Bird》这款小游戏。尽管我们的游戏在设计中还有许多不足之处，但是我们小组成员在游戏的设计、改错，到最后完成的过程中，学习到一些查阅文献、搜集资料的技能，学习到优化游戏的方法，将所学到的 python 编程技能运用到实际中，加深了对 python 编程知识的理解。同时，我们小组成员之间进行了合理的分工与合作，体验到了团队合作的乐趣，在学习中收获了知识、技能、友情等，我们在过程中不断地成长。

七、 项目分工

1、**代码方面**:

刘雨林: 物体追踪方面的处理

余悦扬：代码中管道、小鸟及碰撞的编写

许馨艺：代码中背景、按键及最终组合

2、**论文方面：**

刘雨林：第五、六章及论文最终整理修改

余悦扬：第四章

许馨艺：第一、二、三、七、八章

3、**Demo 制作：**刘雨林

八、致谢

我们由衷地感谢鲍杨老师对我们悉心的指点和教导，鲍杨老师带领我们进行了一段奇妙的编程之旅，帮助我们完成这次的游戏设计。

最后，我们要感谢 Github、CSDN 网站，网站给我们提供了丰富的资源，帮助我们顺利地完成此次游戏的设计。