AM045: Introduction to Business Computing

Report for “Flappy Blocks Go!!!”

Group name: Jointly Carry Highest Gpa (JCHG)

Members: 吉雨薇 陈欣 黄欣纯 高雅

Teacher: Bao Yang

12nd/06/2018

Contents

Abstract

Our game was initially to control the flight route of the bird to avoid the pipes appearing randomly, thus getting scores. We later improved the game and made it realistic that the player can control the movement of the bird through laughing and speaking, in another word, through measurement of the his or her volume. The report is going to explain the reason why we choose the game, describe the process of our trial and error, point out some existing problems and state how we plan to improve it.

Problem and Tasks description

0.1 Background

Our project is adapted from a popular game *Flappy Bird*. *Flappy Bird* is a mobile game developed by a Vietnamese video game artist and programmer. The game is a side-scroller where the player controls a bird, attempting to fly between columns of green pipes without hitting them.

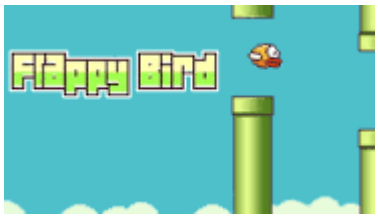

Fig0: flappy bird

The bird will fall down gradually without controlling, and the game player makes the bird fly up by tapping his cellphone screen. And we aim to recreate the classic game named flappy block in a simplified version on PC with Python.

0.2 Our tasks in this project

1. Creating a game interface

Using *pygame*(discussed later) to draw rectangles to display the whole game and to replace “birds”. In this interface, instructions and feedbacks can also be shown to players. Also, defining the colors of main objects is primarily necessary.

2. Defining the motion mode of the object

In the whole game, there are several different movement methods of the block. We need to define several functions to describe their movement, like free fall and horizontal

uniform movement.

3. Adding audio control

While the original version of *Flappy bird* is played with cellphone, we are also inspired by another popular game *八分音符（FYI:av8717160）which shares some similarity to our ideal 'audio-controlled Flappy Block'* . So we are inspired to apply some sound collection method to our program.

Chapter 1 Problem analysis and Basic module

1.1 Problem analysis

The tasks reveal that the project should be divided into three parts.

1. Defining functions needed

In addition to setting interface parameters, we need to define and class some critical variables in the program.

e.g. pipe for blocks; bird for the player; score and highest score

2. Updating the game logic to bird and pipe.

Probably this is the trickiest part in our codes. Though the game logic may seem complicated, but actually bird only have 2 action: flap(fly higher) or flop! So we basically figure out in what condition should the bird change its movement accordingly. Then we can combine the logic with pipe.Most importantly, we make it clear by introducing the important variable 'GameState' to define different stage in the game.

i.e. GameState=1 preparing

GameState=2 playing

GameState=3 resetting

3.Improving the control method

Actually in *pygame* there are prestored functions to allow space bar to control movement of the block, but now in order to realize audio control , we've learned a new module called *pyaudio*, by which we can get volume from the player and then it is stored in a variable 'k'

1.2 Module and Theorem: how we learn them

1.2.1 Pygame

Pygame (the library) is a free and open source python programming language library

for making multimedia applications. Initially we googled it to get some information and later we mainly learn it by reading the instructions on its official website https://www.pygame.org and mostly on https://www.pygame.org/docs/

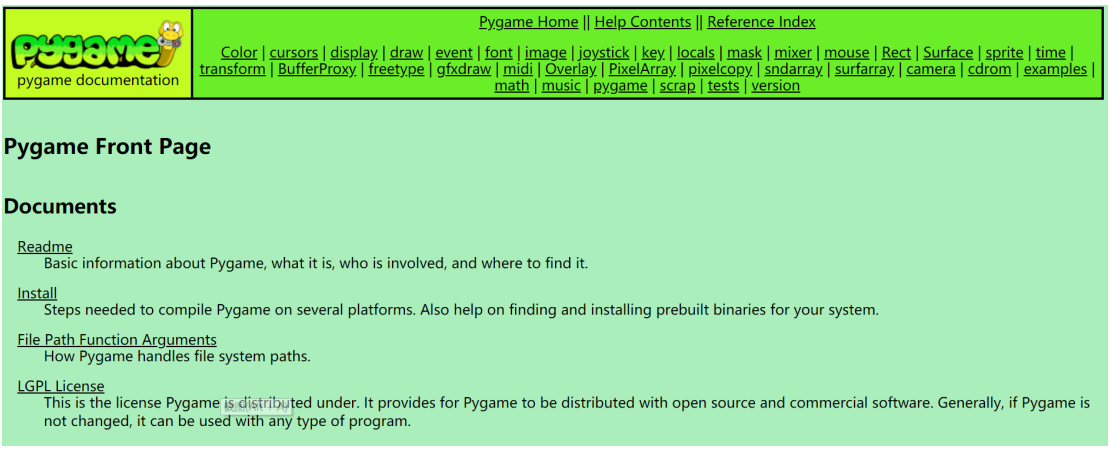

Fig1.1:pygame

In this way, we use the build-in functions *display* to create an interface, *draw* to show some instructions and *font* to set the font.

1.2.2 Pyaudio

Unfortunately we can't find an official library of directions to pyaudio, but on CSDN there are plenty of courses to introduce it, like codes below:

```
15  p = pyaudio.PyAudio()
16
17  stream = p.open(format=p.get_format_from_width(wf.getsampwidth()),
18                  channels=wf.getnchannels(),
19                  rate=wf.getframerate(),
20                  output=True)
```

Fig1.2:pyaudio 1

However, these codes are mainly used to record sound which is a little different from what we want to. Finally we've notice a slide from Google can greatly meet our need

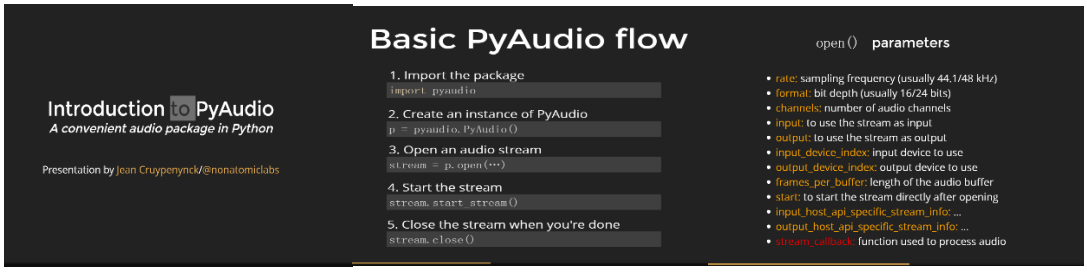

Fig1.3:pyaudio 2

1.2.3 Class

According to what we planned, in the three different states of the game, we all need to define functions of birds so that these functions can be called in the loop. But later we

found this assumption is complicated to achieve because the process should restart again and again. So we googled and find a new method “class”, which seems that it is able to simplify our code.
After the class is instantiated, its properties can be used. In fact, after creating a class, its properties can be accessed through the class name. In our project, we use class to set different modes of bird and pipe.

Chatpter 2 Solution design

2.1 Import some libraries

In order to design the game, we import some libraries in advance which we introduce before

```
import random
import pygame
from pyaudio import PyAudio,paInt16
import struct
pygame.init() #初始化Pygame
global k #声音变量
#import math
```

2.2 Some preparations for the game

We use RGB coordinates to select the color of different objects.
At the same time, We choose the size of the screen and present the words “Flappy Block" on it.

```
WHITE = (255,255,255)
BLACK = (0,0,0)
PIPE = (117,190,49)
SKY = (78,192,202)
GROUND = (224,215,146)
DARK_GROUND = (124,115,46)
BIRD = (241,186,62)
```

```
size = (800,700)
screen = pygame.display.set_mode(size)#创建窗口

pygame.display.set_caption("Flappy Block")
done = False
clock = pygame.time.Clock()
arial18 = pygame.font.SysFont('arial',18, False, False)#字体
arial30 = pygame.font.SysFont('arial',30, False, False)#字体
```

Then we give the game an initial situation

```
gameState = 1
#1 游戏开始前  2 游戏进行中  3.游戏结束
pipes = []
score = 0
highScore = 0
```

2.3 Description of objects

In order to work with a group of images, we had to use “class”. It is used to group data and functions.

2.3.1 Description of the bird

we use one rectangle to represent the bird and then give it an initial position and its initial speed equals to zero. Then we use function to determine the speed and direction of its movement. Initially the bird is supposed to imitate the free fall motion, but as the location of the bird should be at the center of the interface, so we only need to control the bird in the vertical direction(y). Once the player wants the bird to fly higher, we add an upward speed to it.

```
class Bird():#鸟的逻辑
    def __init__(self):#初始位置
        self.x = 250
        self.y = 250
        self.yV = 0

    def flap(self):#控制鸟的上升
        # self.yV = -math.log(k) 更平滑的函数
        self.yV=-0.0008*k


    def update(self):#控制鸟的下降
        self.yV += 0.5#模拟重力加速度
        self.y += self.yV
        if self.y >= 600:
            self.y = 600
            self.yV = 0
        if self.yV > 20:
            self.yV = 20

    def draw(self):
```

```
    def draw(self):
        pygame.draw.rect(screen, BIRD, (self.x, self.y, 40, 40))
        #（位置，颜色，（横坐标，纵坐标（左上角），宽度，高度））

    def reset(self):#重设鸟的位置
        self.x = 250
        self.y = 250
        self.yV = 0

bird = Bird()
```

2.3.2 Description of the pipes

We create some pipes with random locations through making the rectangle between two pipes appear randomly.

```
class Pipe():#管子的逻辑
    def __init__(self):
        self.centerY = random.randrange(130, 520)
        #管子间隔矩形的中点纵坐标
        self.x = 800
        self.size= 150 #半个高
```

Then we make the pipes move leftwards at an appropriate rate to make it look like that the bird is flying to the right. They should move at a right-to left uniform speed(x) and we are going to import “random” to set the location(y) of the hole.

```
def update(self):
    global pipes
    global bird
    global gameState#游戏状态
    global score
    self.x -= 4 #管子左移
    if self.x == 300:
        pipes.append(Pipe())
    if self.x <= -100:
        del pipes[0]
```

After it, we describe in which case can the bird get through the pipes successfully and when the game is over

Only when the location of the bird is lower than the upper pipe and higher than the lower pipe can the player get one more score.

```
            pipes.append(Pipe())
        if self.x <= -100:
            del pipes[0]
        if self.x >= 170 and self.x <= 290 and bird.y <= (self.centerY - self.size) \
                or self.x >= 170 and self.x <= 290 and (bird.y + 40) >= (self.centerY + self.size):
            gameState = 3#结束状态
        if self.x == 168 and bird.y > (self.centerY - 100) and bird.y < (self.centerY + 100):
            score += 1
        if bird.y >= 600:
            gameState = 3

    def draw(self):
        pygame.draw.rect(screen, PIPE, (self.x, 0, 80, (self.centerY - self.size)))
        pygame.draw.rect(screen, PIPE, (self.x, (self.centerY + self.size), 80, (548 - self.centerY)))


pipes.append(Pipe())
```

2.4. Main project loop

We use while loop to describe three possible situation of the game
First, we give the requirement of the sound for the player to begin or reset the game.

```
if k>3000:#声音>3000时
    if gameState == 1:#若游戏未开始则开始游戏
        gameState = 2
    elif gameState == 3:#结束阶段
        bird.reset()
        pipes = []
        pipes.append(Pipe())
        gameState = 2
        score = 0
    else:
        bird.flap()#如果游戏正在进行时执行flap

screen.fill(SKY)
pygame.draw.rect(screen, GROUND, (0, 650, 800, 50))
pygame.draw.line(screen, DARK_GROUND, (0, 650), (800, 650), 5)
pygame.draw.line(screen, DARK_GROUND, (0, 650), (800, 650), 5)
```

Situation 1: If the game hasn't begin
Draw the scene on the screen

```
if gameState == 1:
    pygame.draw.rect(screen, GROUND, (300, 300, 200, 100))
    pygame.draw.rect(screen, DARK_GROUND, (300, 300, 200, 100), 5)
    text = arial18.render("Laugh loudly to play :-D", True, DARK_GROUND) #true 加粗
    textX = text.get_rect().width
    textY = text.get_rect().height
    screen.blit(text, ((400 - (textX / 2)), (350 - (textY / 2))))
```

Situation two: If the game is underway
Judge whether the player get one score

```
if gameState == 2:
    bird.update()
    bird.draw()

    for pipe in pipes:
        pipe.update()
        pipe.draw()

    if score > highScore:
        highScore = score

    text = arial30.render(str(score), True, WHITE)
    textX = text.get_rect().width
    textY = text.get_rect().height
    screen.blit(text, ((400 - (textX / 2)), (50 - (textY / 2))))
```

Situation three: If the game is over
Draw the scene on the screen

```
if gameState == 3:
    for pipe in pipes:
        pipe.draw()
    bird.draw()
```

```
pygame.draw.rect(screen, GROUND, (300, 250, 200, 200))
pygame.draw.rect(screen, DARK_GROUND, (300, 250, 200, 200), 5)
text = arial18.render(("Score: " + str(score)), True, DARK_GROUND)
textX = text.get_rect().width
textY = text.get_rect().height
screen.blit(text, ((400 - (textX / 2)), (300 - (textY / 2))))
text = arial18.render(("High Score: " + str(highScore)), True, DARK_GROUND)
textX = text.get_rect().width
textY = text.get_rect().height
screen.blit(text, ((400 - (textX / 2)), (350 - (textY / 2))))
text = arial18.render("Press space to play", True, DARK_GROUND)
```

```
textX = text.get_rect().width
textY = text.get_rect().height
screen.blit(text,((400 - (textX / 2)),(400 - (textY / 2))))
text = arial30.render(str(score),True,WHITE)
textX = text.get_rect().width
textY = text.get_rect().height
screen.blit(text,((400 - (textX / 2)),(50 - (textY / 2))))
```

2.5. Improvement : gather sound

At first, we control the flying of the bird through one click of the space, which means every time the bird can only move for a fixed distance

```
class Bird():
    def __init__(self):
        self.x = 250
        self.y = 250
        self.yV = 0

    def flap(self):
        self.yV = -10
```

Then ,we are inspired by another game and decided to try to control the bird through the measurement our volume.

Through the learning of the pyaudio, we made it to turn the volume into physical numbers.

```
pipes.append(Pipe())

NUM_SAMPLES = 1000#声音采集量

# 开启声音输入
pa = PyAudio()
SAMPLING_RATE = int(pa.get_device_info_by_index(0)['defaultSampleRate'])
stream = pa.open(format=paInt16, # 16进制整数格式
                 channels=1, # 单声道
                 rate=SAMPLING_RATE,  # 采样率（默认44.1khz）
                 input=True, # 输入
                frames_per_buffer=NUM_SAMPLES)# 采集量
```

Then we use the number to control the flying of the bird and even to determine its ascending speed through measurement of the volume!

```
class Bird():#鸟的逻辑
    def __init__(self):#初始位置
        self.x = 250
        self.y = 250
        self.yV = 0

    def flap(self):#控制鸟的上升
        # self.yV = -math.log(k) 更平滑的函数
        self.yV=-0.0008*k
```

2.6. Select the fps and display the game on the screen

```
    pygame.display.flip()

    clock.tick(60)

pygame.quit()
```

Chapter 3 Some Problems and Further Improvement

3.1 Problems: the function “flap”

Although we have tried different functions with the independent variable *k* to define how sensitive the block is to the sound of the same volume, we still can’t tell which one is the best. Whether the change in ‘yV’ should be gentle or steep still remains to be resolved. Due to our limited Calculus knowledge, we’ve tried several elementary functions. To achieve better presentation performance we applied the Positive proportion function， but practically it may be more suitable to use the ln() ,because it is more smooth.

3.2 Further improvement

3.2.1 Optimize the image

In this project, we use the most simple graphic to represent the objects, we expect that the real image of bird and pipe can be applied with our further effort. Additionally, perhaps the background pattern can also be changeable automatically to make the scene more vivid.

3.2.2 Add some sound effects

Generally as we play games, music and sound effects are important parts of a game, such as the game *Plants V.S. Zombies*. Due to the fact we are using sound to control the bird, obviously music is not so appropriate for the game. However, attractive sound effects are really necessary, such as congratulations for getting a highest score and some negative sounds for a low score.

3.2.3 Set more levels

Since our Flappy Block is a single-time game. If players want to play again, he can only play the same mode of game. Maybe we can improve it to a multiple-level game, like *Angry bird*. As the level goes up, the game will be more challenging. For example, the size of holes in the middle of pipes can be smaller and smaller, or the flying speed of the block can rise gradually.

Summary

Through the process of this project task, our group has not only reproduce a classic game with our joint effort, but also gained a large amount of new knowledge . Although the codes are not that easy, we still have great fun testing our program, as the bird's flying path is often out of control. Luckily we have finished this project pretty well , we think :) .

Acknowledgement

We are sincerely grateful to Mr. Bao, our tutor, with extraordinary patience and consistent encouragement. His tutorial lessons have contributed a lot to our success of this project!