上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Project Name:	**Angry Piggy**
Team Name:	译茗璇转跳跃我不停歇
Faculty Advisor:	**Bao Yang**
Team Members:	赵子璇 **518120910224**
	王译晗 **518120910199**
	陈逸茗 **518120910213**

I Abstract

This project focuses on creating a reversed version of the game "Angry Birds", in which the protagonist becomes the piggy because we try to view them from a different angle with the story background set in an opposite situation. How we worked out, polished and developed our coding and more details will be present as follows.

II The tools we use

Python is used world-wide because of its powerful and various modules for multiple uses. Thanks to these modules, we are able to define and create our own python game.

2.1 Pygame Module

Pygame is a Python wrapper over around the SDL library with a few unique libraries with an emphasis on game programming, written by Pete Shinners. There are a number of libraries developed on top of Pygame such as GUI libraries like GooeyPy and PGU's gui. It is licensed under the LGPL. Pygame and SDL serve as an excellent C engine for 2D games. Pygame also has modules to do things like input handling for the keyboard, mouse, and joystick. It can mix audio and decode streaming music. With the Surfaces one can draw simple shapes, rotate and scale the picture, and even manipulate the pixels of an image in realtime as numpy arrays. Pygame also has the ability to act as a cross platform display layer for PyOpenGL. Most of the pygame modules are written in C, few are actually done in Python.

With the aid of Pygame, we realize our expectation of the game.

2.2 Random Module

Random module implements pseudo-random number generators for various distributions. Almost all module functions depend on the basic function random (), which generates a random float uniformly in the semi-open range [0.0, 1.0). In our case, we take advantage of it to randomly generate enemy of the piggy.

2.3 Audio Converter

To optimize the game, we decide to play the game with background music. Therefore, we downloaded the theme music of Angry Birds. However, to make it work in our codes, we need to transfer the MP3 file into WAV file. And we solved the problem with Audio Converter.

III Reasons for Choosing This Topic

The inspiration comes from the world-famed game Angry Birds, whose latest version is set in the universe and related to the well-known movie series Star Wars. We are all fan of both, but we are so tired of the traditional type of “fight back against the evil side” games. The official declared that the piggy came from alien stars, so we wonder why not create our own Python game to help the piggy travel through space.

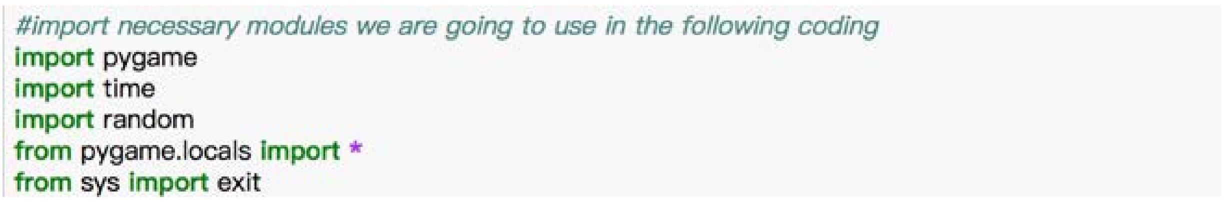
And since we have mastered basic skills for writing a python game, why not help our angry piggy reunite with its family? ;)

IV The Process of Coding

4.1 Preparations:

4.1.1 Some “import” s

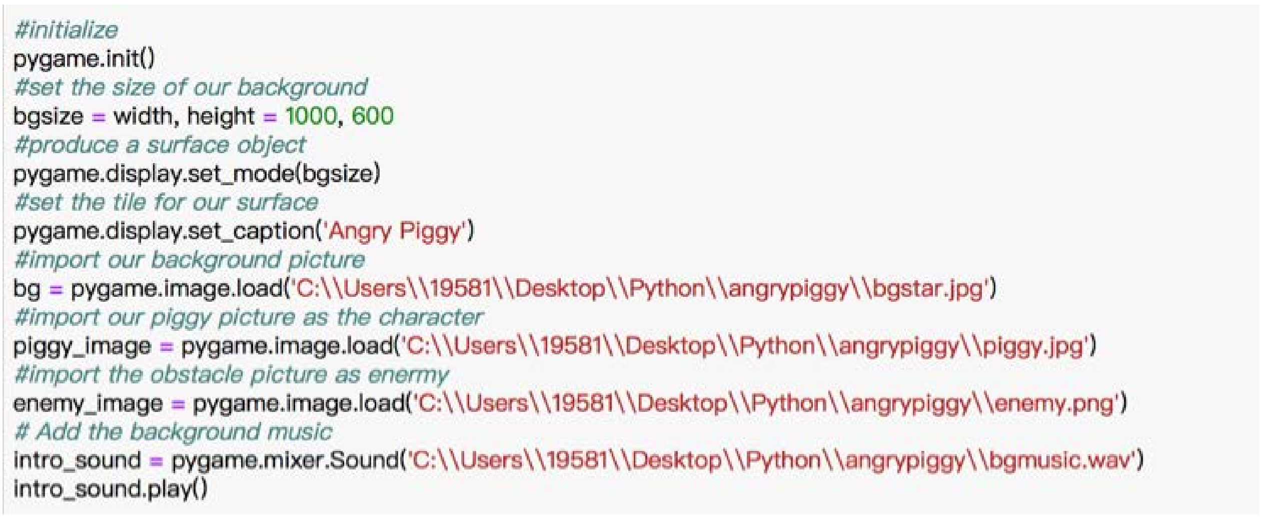
Pygame (the library) is a free and open source python programming language library for making multimedia applications. We will use its build in functions in the following. And we borrow” exit” to be used to get out from the game.



```
#import necessary modules we are going to use in the following coding
import pygame
import time
import random
from pygame.locals import *
from sys import exit
```

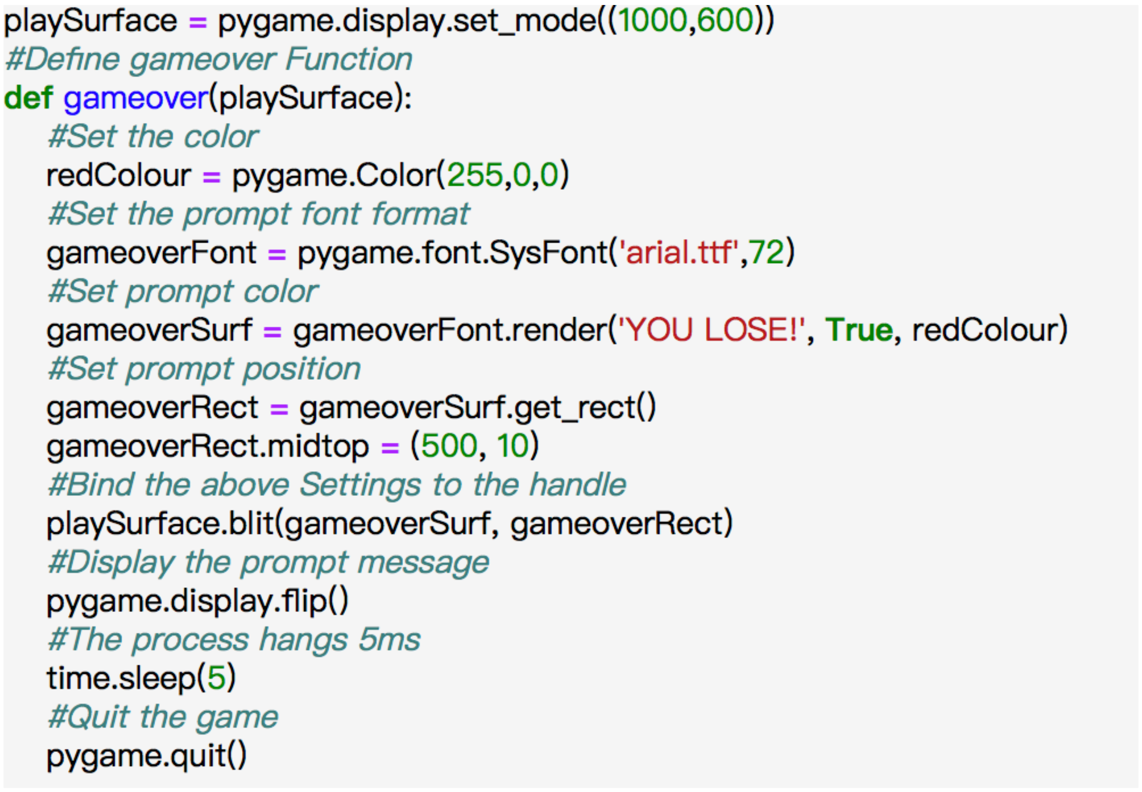
4.1.2 Some basic settings

In this part, we set our background, piggy character and the enemy use” pygame. image. Load” to transfer images into the program.



```
#initialize
pygame.init()
#set the size of our background
bgsize = width, height = 1000, 600
#produce a surface object
pygame.display.set_mode(bgsize)
#set the tile for our surface
pygame.display.set_caption('Angry Piggy')
#import our background picture
bg = pygame.image.load('C:\\Users\\19581\\Desktop\\Python\\angrypiggy\\bgstar.jpg')
#import our piggy picture as the character
piggy_image = pygame.image.load('C:\\Users\\19581\\Desktop\\Python\\angrypiggy\\piggy.jpg')
#import the obstacle picture as enermy
enemy_image = pygame.image.load('C:\\Users\\19581\\Desktop\\Python\\angrypiggy\\enemy.png')
# Add the background music
intro_sound = pygame.mixer.Sound('C:\\Users\\19581\\Desktop\\Python\\angrypiggy\\bgmusic.wav')
intro_sound.play()
```

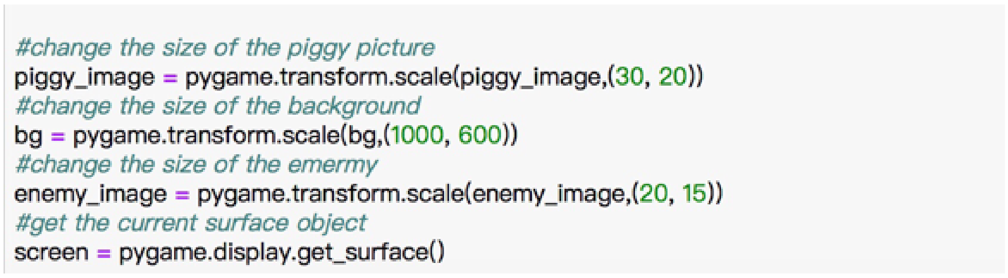
When it comes to quitting the game, we define a game over class to warn the player.



```
playSurface = pygame.display.set_mode((1000,600))
#Define gameover Function
def gameover(playSurface):
    #Set the color
    redColour = pygame.Color(255,0,0)
    #Set the prompt font format
    gameoverFont = pygame.font.SysFont('arial.ttf',72)
    #Set prompt color
    gameoverSurf = gameoverFont.render('YOU LOSE!', True, redColour)
    #Set prompt position
    gameoverRect = gameoverSurf.get_rect()
    gameoverRect.midtop = (500, 10)
    #Bind the above Settings to the handle
    playSurface.blit(gameoverSurf, gameoverRect)
    #Display the prompt message
    pygame.display.flip()
    #The process hangs 5ms
    time.sleep(5)
    #Quit the game
    pygame.quit()
```

4.1.3 Modify the size

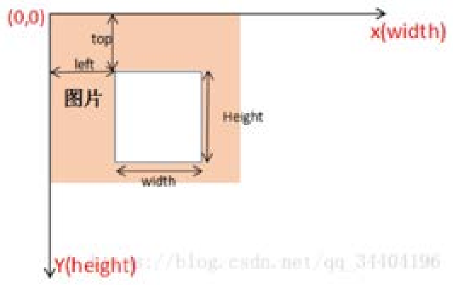
We change the size of the background, piggy and enemy to a more suitable size, using the build-in “pygame.transform.scale” command. And get the current surface.

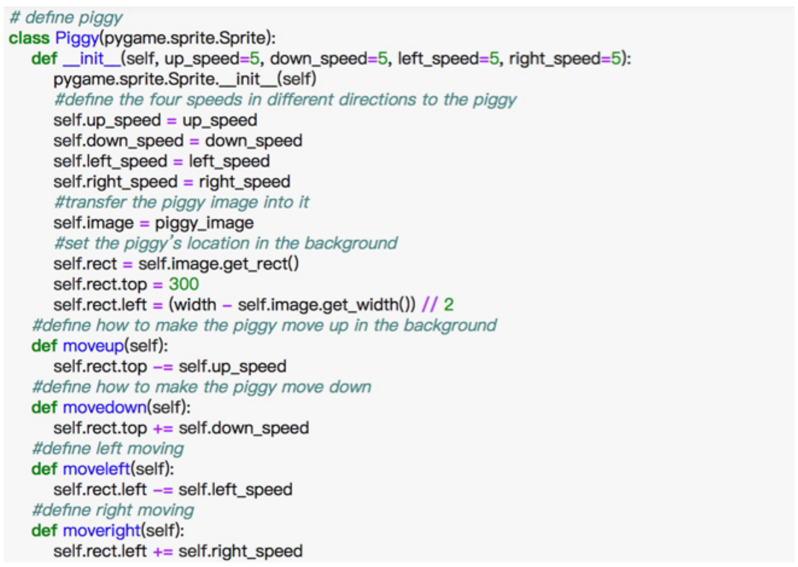


```
#change the size of the piggy picture
piggy_image = pygame.transform.scale(piggy_image,(30, 20))
#change the size of the background
bg = pygame.transform.scale(bg,(1000, 600))
#change the size of the emermy
enemy_image = pygame.transform.scale(enemy_image,(20, 15))
#get the current surface object
screen = pygame.display.get_surface()
```

4.2 Define our character——the super piggy

We define a class called piggy and there’s four variables: the speeds at which it can move towards four different directions. We use “.image” command for picture transferring. We use “get_rect” plus setting top and left to give the piggy its location. The following picture can explain what is “rect”.
Then define functions to tell how the piggy moves in four directions in the “rect” coordinate.

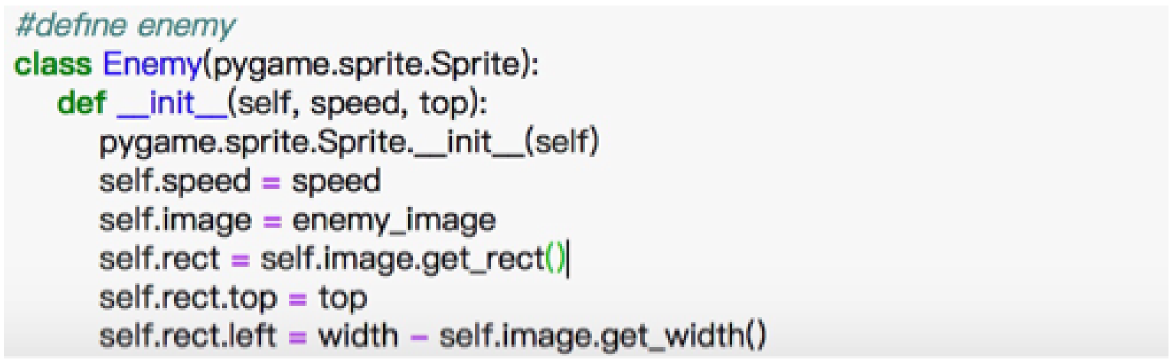




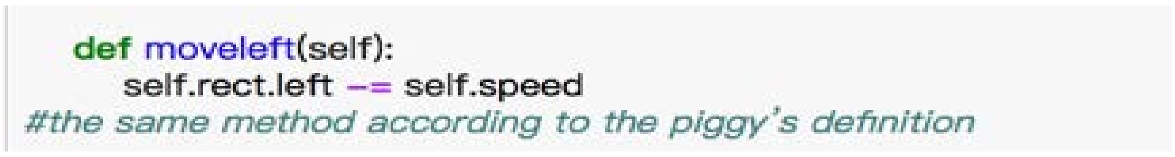
```
# define piggy
class Piggy(pygame.sprite.Sprite):
    def __init__(self, up_speed=5, down_speed=5, left_speed=5, right_speed=5):
        pygame.sprite.Sprite.__init__(self)
        #define the four speeds in different directions to the piggy
        self.up_speed = up_speed
        self.down_speed = down_speed
        self.left_speed = left_speed
        self.right_speed = right_speed
        #transfer the piggy image into it
        self.image = piggy_image
        #set the piggy’s location in the background
        self.rect = self.image.get_rect()
        self.rect.top = 300
        self.rect.left = (width - self.image.get_width()) // 2
    #define how to make the piggy move up in the background
    def moveup(self):
        self.rect.top -= self.up_speed
    #define how to make the piggy move down
    def movedown(self):
        self.rect.top += self.down_speed
    #define left moving
    def moveleft(self):
        self.rect.left -= self.left_speed
    #define right moving
    def moveright(self):
        self.rect.left += self.right_speed
```

4.3 Define the obstacle—-the enemy

The way is similar to the way we define piggy. Since the enemy just needs to move leftward, just one speed need to be defined.



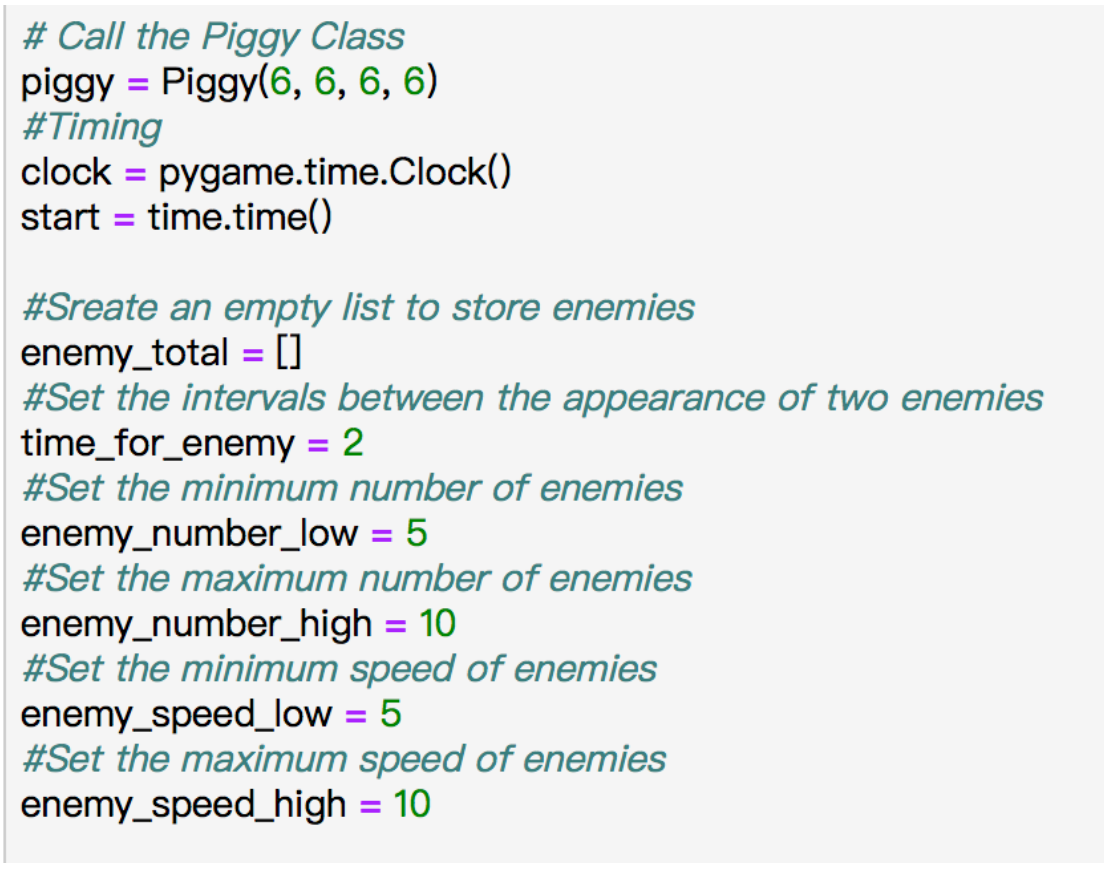
```
#define enemy
class Enemy(pygame.sprite.Sprite):
    def __init__(self, speed, top):
        pygame.sprite.Sprite.__init__(self)
        self.speed = speed
        self.image = enemy_image
        self.rect = self.image.get_rect()
        self.rect.top = top
        self.rect.left = width - self.image.get_width()
```



```
    def moveleft(self):
        self.rect.left -= self.speed
#the same method according to the piggy's definition
```

4.4 Preparations for calling Enemy Class

In this section, we first call the Piggy class, then time it, and create some basic parameters of the enemy，such as the time interval between the appearance of the enemies, the maximum and minimum number of enemies, the maximum and minimum speed of the enemies, so as to randomly generate enemies later.



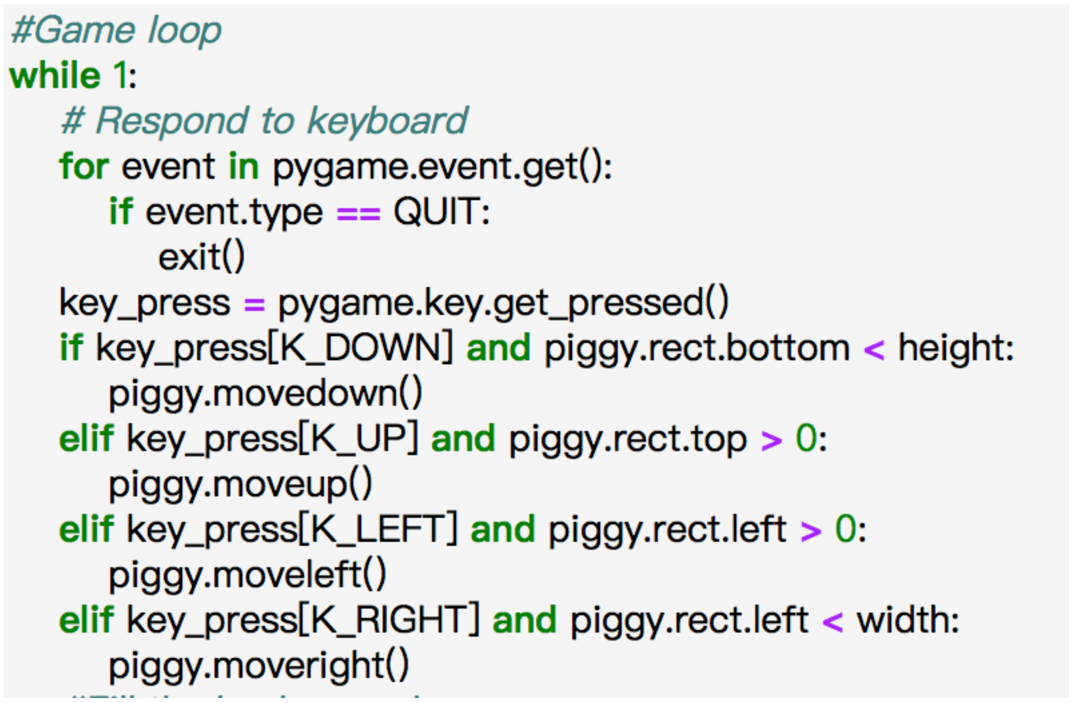
```
# Call the Piggy Class
piggy = Piggy(6, 6, 6, 6)
#Timing
clock = pygame.time.Clock()
start = time.time()

#Sreate an empty list to store enemies
enemy_total = []
#Set the intervals between the appearance of two enemies
time_for_enemy = 2
#Set the minimum number of enemies
enemy_number_low = 5
#Set the maximum number of enemies
enemy_number_high = 10
#Set the minimum speed of enemies
enemy_speed_low = 5
#Set the maximum speed of enemies
enemy_speed_high = 10
```

4.5 Game loop

4.5.1 Respond to keyboard

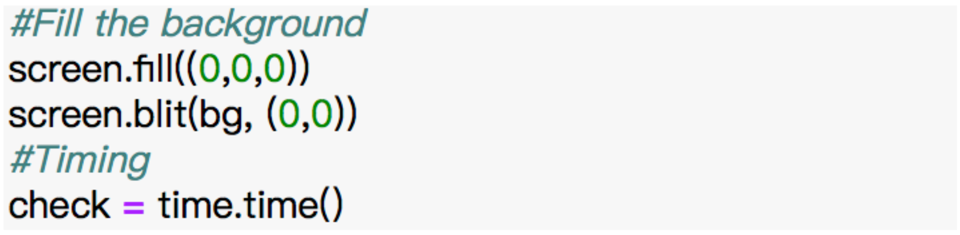
In this part, the code mainly responds to keyboard actions. For example, the piggy will move down when the user presses down and the bottom of the pig does not exceed the screen.



```
#Game loop
while 1:
    # Respond to keyboard
    for event in pygame.event.get():
        if event.type == QUIT:
            exit()
    key_press = pygame.key.get_pressed()
    if key_press[K_DOWN] and piggy.rect.bottom < height:
        piggy.movedown()
    elif key_press[K_UP] and piggy.rect.top > 0:
        piggy.moveup()
    elif key_press[K_LEFT] and piggy.rect.left > 0:
        piggy.moveleft()
    elif key_press[K_RIGHT] and piggy.rect.left < width:
        piggy.moveright()
```

4.5.2 Fill the background and Timing

In this part, we fill in the background and time the game.



```
#Fill the background
screen.fill((0,0,0))
screen.blit(bg, (0,0))
#Timing
check = time.time()
```

4.6 Call Enemy Class

4.6.1 Create enemy_total list

The mechanism for invoking an enemy is as follows:

- The number of enemies is a random number between 5 and 10
- Repeat the loop as many times as the number of enemies
- The enemy's velocity is a random number between 5 and 10
- In every cycle, attach the enemy to the enemy_total list

```
# Call Enemy Class
if check - start > time_for_enemy:
    enemy_num = random.randint(enemy_number_low, enemy_number_high)
    for i in range(enemy_num):
        enemy = Enemy(speed = random.randint(enemy_speed_low, enemy_speed_high)
        top = height*(i+1)/enemy_num + random.randint(-20,20))
        enemy_total.append(enemy)
    start = time.time()
```

4.6.2 Display the enemy image

For each randomly generated enemy in the list of enemy_total list, enemy move to the left. And then display enemy image.

```
# Display the enemy image
for enemy in enemy_total:
    enemy.moveleft()
    screen.blit(enemy_image, enemy.rect)
```

4.6.3 Quit the game

If the following conditions are satisfied at the same time, show the prompt ‘’Game Over’, print ‘You lose ‘and jump out of the loop.

```
if enemy.rect.left <= piggy.rect.left and enemy.rect.left >= piggy.rect.left-40 and enemy.rect.top >= piggy.rect.top and enemy.rect.top <= piggy.rect.to
    gameover(playSurface)
    print('You Lose!')
    exit()
```

4.7 Run our game

In the last part, we mainly carry on the preparation work of running the whole game, including showing the piggy image, controlling the maximum frame rate of game drawing to no more than 25 frames per second and visualizing the window.

```
#Display pig image
screen.blit(piggy_image, piggy.rect)
#Control the maximum frame rate of game drawing to 25 –No more than 25 frames per second
clock.tick(25)
#Visualiaze the window
pygame.display.flip()
```

VI Weakness

Our game is just a much simpler and basic game, compared to more developed games in our lives, which allows multiplayer's and various changes and difficulty levels. Some ways our piggy game can be improved is that using programs to allow 2players to compete, or increase the level of difficulty when the player reaches a certain number of scores. But it requires further learning.

VII Conclusion

In fact, at the beginning, we had no idea about the whole project. We wrote, changed, and crashed hundreds of times, but we grit our teeth and held on. When we saw that our game could run successfully and our game could say 'Hello, world', we realized that it was worth all the efforts and pains. We are satisfied with ourselves, for we not only applied what we have learned in class, but also learned to use Pygame, Time module and found the available resources in the Github. In the process of working together on the project, we harvest the ability of self-learning and cultivate the team cooperation spirit. We have the loveliest teammates. We have the loveliest teacher. Python, we love you 3000 times.

Team Member's Contributions

王译晗	陈逸茗	赵子璇
Coding line 78-123	Coding line 37-74	Proposal
PPT Page 10-16	PPT Page 4-9	Coding line 1-37
Coding Polishing	Coding Polishing	PPT Page 1-3、17-19
Report ⅣⅤⅦ	Report ⅤⅣ	Report Ⅰ ⅡⅢ
Debugging	Debugging	Debugging