Project: Left and right inter-fighting game

By Team 509

张佳诚 518120910072

谭智壕 518120910069

王逸其 518120910071

Abstract:

This report will provide a brief overview of our team project resolution process, including:

1. The background of our project
2. The brief introduction of our game
3. The process of coding
4. Our shortages
5. Conclusion

Background

As is mentioned in our project proposal, we choose to create a game by using Pygame since all of us are keen on playing games. Meanwhile, a friend of one of our team members in NYU has already programmed an amusing game. In his game, there are two players controlling two chesses in a circle ‘court’. One player is called chaser, whose goal is to use direction keys to collide his chess with the other chess. The other player is called escapee, whose goal is to use ‘wasd’ keys to control his chess to evade the capture of the chaser. Besides, both chesses are not allowed to run out of the court and the escapee will win if the chaser don’t capture the escapee in 20 seconds. We decided to borrow some of his idea but modify his game’s rules and code by ourselves so that we can create our own game.

Our game

Inspired by the game above, we make our own rule: there are also two players, but they control the same chess. One player is called attacker, whose goal is to use direction keys to push the chess out of the court, while the other player is called defender, whose goal is to use ‘wasd’ keys to keep the chess inside the court. The defender will win if the attacker can’t push the chess out of the court in 20 seconds. What’s more, the power of the attacker will increase as time passes by so that the game can be fiercer in the last several seconds. We call our game ‘Left and right inter-fighting game’.

Although it seems that we simplify the game in a way, it is still challenging to us because we are about to use Pygame which is quite unfamiliar to us. We need to

search useful methods of Pygame in www.pygame.org and learn how to use them correctly by ourselves. However, we eventually did it! In the next part, we will describe our coding process in detail.

Coding process

(1) Install & Import Pygame

First of all, we need to enter the following code to install Pygame:

```
pip install pygame
```

And import pygame:

```
import pygame
```

(2) Create a pygame screen and all the objects

```
pygame.init()
screen = pygame.display.set_mode((700, 700))
bgm = pygame.mixer.Sound("resource/bgm.wav")
im_chess = pygame.image.load("resource/black.png")
im_court = pygame.image.load("resource/court.png")
logo = pygame.image.load("resource/steam.jpg")
im_start = pygame.image.load("resource/start.png")
im_restart = pygame.image.load("resource/restart.png")
```

We first initialize the game and create a screen. Then we create objects we need including the background music, chess, court, logo and the start & restart button.

(3) Put these objects on the screen

```
screen.blit(im_court, (0, 0))
screen.blit(im_chess, (327, 327))
screen.blit(im_start, (200, 289))
pygame.display.set_icon(logo)
pygame.display.flip()
pygame.mixer.Channel(1).play(bgm)
```

(4) Define several needed functions

```
def timer(time, timerevent):
    while time[1] == True:
        time[1] = False
        print(time[0])
        pygame.time.set_timer(timerevent, 1000)

def countdown(time):
    if time[0] > 0:
        time[0] -= 1
        time[1] = True
```

These two functions combined together can serve as a timer.

'pygame.time.set_timer(timerevent,1000)'can create a timerevent on the event queue every 1000 milliseconds so that we can count down the time every 1000 milliseconds(=1 second).

```
def get_action(action):
    pygame.time.set_timer(action, 40)
```

This function can help us check players' action every 40 milliseconds in our game loop later.

```
def checkwin(pos):
    if (pos[0] + 22.5 - 358) ** 2 + (pos[1] + 22.5 - 350) ** 2 > 265 ** 2:
        return True
    else:
        return False
```

The checkwin() function can check if the chess has run out of the court(if the attacker has won the game).

(5) Prepare some variables to be used

```
running = True
playing = False
moving1 = [0, 0]
moving2 = [0, 0]
pos = [327, 327]
time = [20, True]
timerevent = pygame.USEREVENT + 1
action = pygame.USEREVENT
winstatus = 0
difficulty = 2.5
```

moving1 and moving2 are two variables which record the distance that players want the chess to move.

pos is the initial location of the chess(in the center of the court).

time & timerevent will be used in the 'timer' function.

Action will be used in the get_action() function.

'+1' after pygame.USEREVENT means that timerevent is a different kind of event in the event queue. It helps the computer to distinguish these two kinds of events.

winstatus helps to present if anyone has won the game.

difficulty represents the attacker's power(how far can the attacker move the chess by just pressing the button one time).

(6) Game Loop

① The beginning of the game

```
while running:
    for event in pygame.event.get():
        if event.type == pygame.MOUSEBUTTONDOWN:
            if not playing:
                playing = True
                get_action(action)
                timer(time, timerevent)
                screen.blit(im_court, (0, 0))
                screen.blit(im_chess, (327, 327))
                winstatus = 0
                difficulty = 2.5
                pos = [327, 327]
                pygame.display.update()
                time[0] = 20
```

If the user click the mouse, the game begins.

```
if event.type == timerevent:
    countdown(time)
    timer(time, timerevent)
```

The timer begins to work.

```
if time[0] == 20 and playing:
    difficulty = 2.5
if time[0] == 15 and playing:
    difficulty = 3.5
if time[0] == 10 and playing:
    difficulty = 4.5
if time[0] == 5 and playing:
    difficulty = 5
```

The attacker's strength begins to gradually increase. In the last five seconds, the attacker's power is equal to the defender's.

② The game process

```
if event.type == action:
    if playing:
        get_action(action)
        if pygame.key.get_focused():
            keys = pygame.key.get_pressed()
            if keys[pygame.K_d] != 0 and keys[pygame.K_a] == 0:
                moving1[0] = 1
            elif keys[pygame.K_a] != 0 and keys[pygame.K_d] == 0:
                moving1[0] = -1
            else:
                moving1[0] = 0
            if keys[pygame.K_w] != 0 and keys[pygame.K_s] == 0:
                moving1[1] = 1
            elif keys[pygame.K_s] != 0 and keys[pygame.K_w] == 0:
                moving1[1] = -1
            else:
                moving1[1] = 0
            if keys[pygame.K_RIGHT] != 0 and keys[pygame.K_LEFT] == 0:
                moving2[0] = 1
            elif keys[pygame.K_LEFT] != 0 and keys[pygame.K_RIGHT] == 0:
                moving2[0] = -1
            else:
                moving2[0] = 0
            if keys[pygame.K_UP] != 0 and keys[pygame.K_DOWN] == 0:
                moving2[1] = 1
            elif keys[pygame.K_DOWN] != 0 and keys[pygame.K_UP] == 0:
                moving2[1] = -1
            else:
                moving2[1] = 0
        pos[0] += 10 * moving1[0]
        pos[0] += moving2[0] * 2 * difficulty
        pos[1] -= 10 * moving1[1]
        pos[1] -= moving2[1] * 2 * difficulty
    if checkwin(pos):
        winstatus = 1
    screen.blit(im_court, (0, 0))
    screen.blit(im_chess, tuple(pos))
    pygame.display.update()
```

During the game process, if players do some actions like pressing ‘wasd’ or ‘direction keys’, the movement of the chess will be reflected on the screen.(The code is too long so we cut it into two pictures.)

```
if winstatus == 1 and playing:
    print("player attacker wins!")
    time[0] = 0
if time[0] == 0 and playing:
    if winstatus == 0:
        print("player defender wins!")
    playing = False
    screen.blit(im_court, (0, 0))
    screen.blit(im_restart, (200, 300))
    pygame.display.update()
```

Every time an event is checked, the computer will check if any player has win the game.

③ The end of the game

```
if event.type == pygame.QUIT:
    pygame.quit()
    running = False
```

If the players want to quit the game, the game will stop running.

(7)Enable the game to run

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

Shortages

Finally, we successfully create the game on our own. But there are still some shortages we can optimize.

(1) The balance of the game

After testing different value of the variable ‘difficulty’, we still can’t find out an appropriate value which can ensure that both players have the same possibility to win the game.

(2) The conciseness of the code

Admittedly, our code works. However, there is no denying that the code can be simplified further. For instance, the timer(including the countdown() function and the timer() function) is likely to be merged into one function, but we didn’t come up with a good idea.

(3) The complexity of our project

After finishing the game, we feel that our game is not complex enough. We can and should do a more challenging job so as to train our programming skill.

Conclusion

It was a unique experience being able to experiment with game development through our application of Python and Pygame. We were able to strengthen our Python and general programming skills while making something that was functional and operational in the real world. All of us are satisfied with our product. Also, we hope to handle more challenging difficulties in the future.

Our grateful thanks to our instructor Bao Yang!