FAT WORM

by the group

WE NEED CACHE

FINAL PROJECT REPORT

VANESSA JIA LU HOW HOK LIN 517120990081

KARINA ELENA LIANG LIANG 517120990057

HYUNTAE JANG 517120990075

JINWOO JEONG 517120990035

1. Abstract

This project was inspired by our childhood game: Snake Game, and we created our own version of it. This report will talk about the problem at hand, the approach we took to make it, the process of our coding, and finally the outcome.

2. Introduction

Our first project using Python, called “Fat Worm” is based on the popular “Snake Game”. *Snake Game* is the common name for a video game concept where the player maneuvers a line which grows in length, with the line itself being a primary obstacle. The concept originated in 1976 from the arcade game Blockade, and the intuitive and ease of implementing Snake resulted in numerous versions. After a variant was preloaded on Nokia’s mobile phones in 1998, many people have become interested. There are more than 300 Snake-like games in iOS alone.

3. Objective

Primary aim of the project is to embody the whole “Snake” game concept through the language of Python, using the simplest codes and methods to create high-quality game that could be enjoyed by the general public.

4. Approach

4.1. Game play

The player controls a dot, square, or object on a bordered plane. As it moves forward, it leaves a trail behind, resembling a moving snake. The player loses when the snake runs into the screen border, a trail or other obstacle, or itself.

4.2. Data Collection

During the research, we have found numbers of program models based on Python which were efficient and outstanding. While considering those options, one member found the idea of Pygame, and also found interesting examples including Frets on Fire, Dangerous High School Girls in Trouble etc. Inspired by the idea of Pygame and its great progress in the community, we decided to produce "Fat Worm" through this Pygame platform.

4.3. Pygame

Pygame is a cross-platform set of Python modules designed for writing video games. It includes computer graphics and sound libraries designed to be used with the Python programming language. Originally written by Pete Shinners, Pygame was to replace PySDL after its development stalled. It has been a community project since 2000 and is released under the open source free software GNU Lesser General Public License.

Built over the Simple DirectMedia Layer (SDL) library, with the intention of allowing real-time computer game development without the low-level mechanics of the C programming language and its derivatives. This is based on the assumption that the most expensive functions inside games (mainly the graphics part) can be abstracted from the game logic, making it possible to use a high-level programming language, such as Python, to structure the game.

5.Coding process

Before starting to write the program, we laid out all the elements of our game, and then work on each one by one. First of all, we had to import the Pygame package, and then we needed to initialize it before being able to do anything with it with the function pygame.init(). The modules that are initialized also need to be quitted at the end to clean everything up with the function quit().

5.1.Game features

5.1.1. Screen display

We defined the game's screen display and the name that will be displayed on top of the window.

```
display_width = 800
display_height = 600
size = display_width, display_height
gameDisplay = pygame.display.set_mode(size)
pygame.display.set_caption("Worm Game!")
```

We also set the game's icon

```
icon = pygame.image.load("wormhead.png")
pygame.display.set_icon(icon)
```

A "clock" had to be created to keep track of frame rates

```
clock = pygame.time.Clock()
```

```
FPS = 15
```

5.1.2. Colors

We defined the colors that we would use for the game.

```
white = (255, 255, 255)
black = (0, 0, 0)
red = (255, 0, 0)
green = (0, 155, 0)
```

5.1.3. Bob the Worm

The player is represented as the worm, which grows if it eats an apple. Bob's head and body is separate.

We uploaded the image of Bob's head.

```
img = pygame.image.load("wormhead.png")
```

We also defined Bob's body. The length of the body is 1 at start, then for each apple that he eats, it will add to his body.

```
snakeLength = 1
```

```
        if lead_x > randApplex and lead_x < randApplex + AppleThickness or lead_x + block_size > randApplex and lead_x + block_size < randApplex + AppleThickness:
            if lead_y > randAppley and lead_y < randAppley + AppleThickness:
                randApplex, randAppley = randAppleGen()
                snakeLength += 1
            elif lead_y + block_size > randAppley and lead_y + block_size < randAppley +
AppleThickness:
                randApplex, randAppley = randAppleGen()
                snakeLength += 1
```

5.1.4. Apple

We uploaded the image of the apple.

```
appleimg = pygame.image.load("apple.png")
```

The apple needs to be regenerated randomly every time after it is eaten.

```
AppleThickness = 30
```

```
def randAppleGen():
    randApplex = round(random.randrange(0, display_width - AppleThickness))
    randAppley = round(random.randrange(0, display_height - AppleThickness))
    return randApplex, randAppley
```

5.1.5. Texts

We defined how any text will appear and set the font.

```
smallfont = pygame.font.SysFont("times", 20)
medfont = pygame.font.SysFont("times", 35)
largefont = pygame.font.SysFont("times", 60)
```

```
def text_objects(text, color, size):
    if size == "small":
        textSurface = smallfont.render(text, True, color)
    elif size == "medium":
        textSurface = medfont.render(text, True, color)
    elif size == "large":
        textSurface = largefont.render(text, True, color)
    return textSurface, textSurface.get_rect()
```

5.2.Gameplay

5.2.1. Play

Firstly, we wrote the instructions for the player. The player can press C to start the game or press Q to quit the game.

```
def game_intro():
    intro = True
    while intro:
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit()
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_c:
                    intro = False
                if event.key == pygame.K_q:
                    pygame.quit()
                    quit()
        gameDisplay.fill(white)
        bg = pygame.image.load("introbackground.png")
        gameDisplay.blit(bg, [0,0])
        pygame.display.update()

        message_to_screen("Welcome to our Worm Game!", red, -100,"large")
        message_to_screen("As you know, the more apple you eat, the longer you become and the more
scores you get!", black, -30)
        message_to_screen("But if you eat yourself or you run into the walls, you die!", black,
50)
        message_to_screen("Press C to play, P to pause, Q to quit.", black, 180)
        pygame.display.update()
        clock.tick(15)
```

Then, player can press ‘up’, ’down’, ’left’, ’right’ to control the snake.

```
def snake(block_size, snakeList):
    if direction == "right":
        head = pygame.transform.rotate(img, 270)
    if direction == "left":
        head = pygame.transform.rotate(img, 90)
    if direction == "up":
        head = img
    if direction == "down":
        head = pygame.transform.rotate(img, 180)

    gameDisplay.blit(head, (snakeList[-1][0], snakeList[-1][1]))

    for XnY in snakeList[:-1]:
        pygame.draw.rect(gameDisplay, green, (XnY[0], XnY[1], block_size, block_size))
```

5.2.2. Pause

The player can press ‘P’ to pause the game if he wants to.

```
def pause():
    paused = True
    message_to_screen("Paused", white, -100, size="large")
    message_to_screen("Press C to continue or Q to quit", white, 40)
    pygame.display.update()
```

And while paused, player can press ‘C’ to continue the game or press ‘Q’ to quit the game.

```
while paused:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            quit()
        if event.type == pygame.KEYDOWN:
            if event.key == pygame.K_c:
                paused = False
            elif event.key == pygame.K_q:
                pygame.quit()
                quit()
    clock.tick(5)
```

5.2.3. Game Over

There are two situations in which Bob will die.

(1)Bob dies if it crashes into the wall.

```
if lead_x >= display_width or lead_x < 0 or lead_y < 0 or lead_y >= display_height:
    gameOver = True
    dead_sound.play()
```

(2)Bob dies if it eats itself.

```
if len(snakeList) > snakeLength:
    del snakeList[0]
for eachSegment in snakeList[:-1]:
    if eachSegment == snakeHead:
        gameOver = True
        dead_sound.play()
```

“gameOver” is False in the gameLoop, this is the lines for when gameOver is True, that is when Bob dies and it is Game Over.

```
while running:
    if gameOver == True:
        message_to_screen("Game over",red ,-50,size="large")
        message_to_screen("Press C to play again, or Q to quit", white, 50, size="medium")
        pygame.display.update()
```

5.2.4. Score

Each time Bob eats one apple, the score increases by one.

```
def score(score):
    text = smallfont.render("Score: " + str(score), True, white)
    gameDisplay.blit(text, [0, 0])
```

5.3. Optimization

5.3.1. Background

The game looked dull and empty, therefore we decided to add cute backgrounds! For example a nice patch of grass for our worm to roam in! We had to first make sure that the image was in the same folder as the python file. Then, first we must fill the screen white, then we can add our desired image on top of it.

```
gameDisplay.fill(white)
bg = pygame.image.load("background.png")
gameDisplay.blit(bg, [0,0])
pygame.display.update()
```

5.3.2. Sound

We added an intro background music for the instructions display, and also a sound effect for when our poor Bob dies. The audio had to be in .wav format for it to play.

First we had to load the sound.

```
intro_sound = pygame.mixer.Sound('intro.wav' )
```

Then, to play the sound effect, we had to call the function as follows:

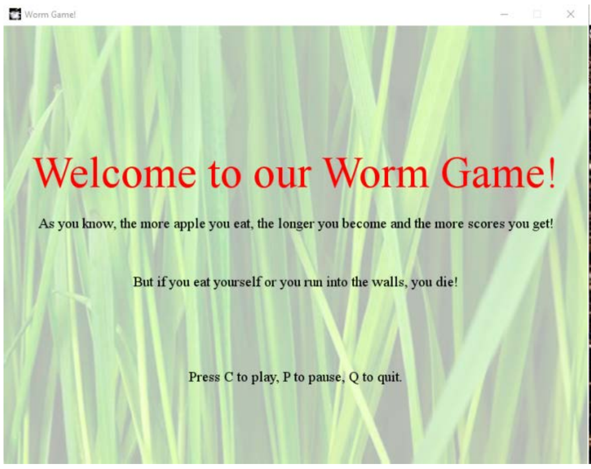
```
intro_sound.play()
```

6. Result

6.1. Outcome

We found that programming requires some math, which was interesting. It took us many many tries before actually running our game successful! The good thing about this game is that it is one of the most basic game to make using Pygame, and is pretty simple for beginners to understand. However, the bad thing is that it is 2-dimensional only and looks quite plain, so it is not so appealing. However, we enjoyed the whole process, from brainstorming about what project to do, to writing the program and see the final result play before our eyes.

The following are screenshots of our game.

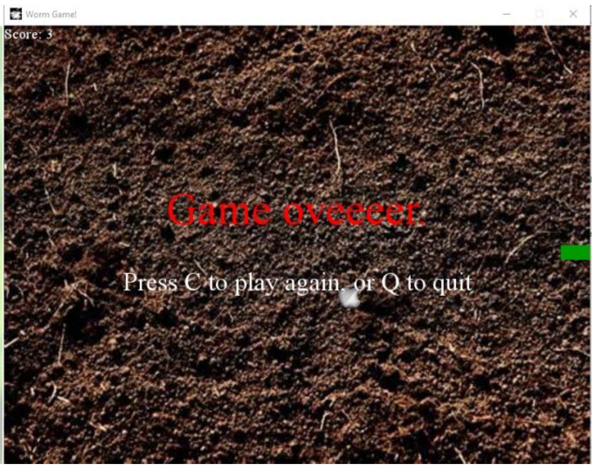


Welcome to our Worm Game!

As you know, the more apple you eat, the longer you become and the more scores you get!

But if you eat yourself or you run into the walls, you die!

Press C to play, P to pause, Q to quit.



Score: 3

Game oveeeer.

Press C to play again, or Q to quit

6.2. Challenges faced

The Programming course did not introduce Pygame to us, therefore almost all our problems arose due to our zero experience in Pygame. However, we spent time to research and study a lot about it, and big thanks to our teacher Baoyang who made us go from a zero to a good eight in programming! One of the challenges was the audio. We were so determined to make it work so we searched high and low on why the program would not play the music, and then finally found out that the audio should be .wav file. We were able to understand Pygame better and we very diligently managed to complete our first python project.