전북대학교
JEONBUK NATIONAL UNIVERSITY

게임프로젝트 레포트

수 강 과 목 : 컴퓨터 과학과 코딩
담 당 교 수 : 홍득조
학 과 : 컴퓨터인공지능학부
학 번 : 202312759, 202312792, 202312802
이 름 : 이서준, 이혜원, 장유빈
제 출 일 : 2023.06.11

목차

1. 아이디어 도출 과정

1) 아이디어 도출

게임을 구현해내기 위해 먼저 어떤 종류의 게임을 만들지 논의했다. 검색 시 많이 보이는 게임은 피하면서 식상하지 않은 게임을 생각해낸 결과 슈팅 게임이 선정되었다. 라운드마다 나오는 몬스터를 처치하는 게임이며 세 라운드를 거쳐 마지막 4라운드에 최종 보스를 처치하는 게임이다. 먼저 슈팅 게임의 컨셉을 정한 후 게임 시작 방법, 케릭터의 외향, 라운드마다 몬스터의 외향, 배경 이미지, 단계마다 난이도 수준, 몬스터 처치 기준, 라운드를 넘어가는 방법 및 기준을 고려했다, 또한 차별성을 두고자 배경, 총알을 제외한 게임 케릭터와 몬스터는 아이패드 앱을 이용해 직접 그림을 그려 넣었다.

2) 게임의 전체적 구성 및 취지

게임 시작 방법은 Tkinter를 사용해 스페이스 바를 누르면 게임이 시작되도록 구현했다. 단계마다 숲, 성, 지옥, 파이썬 로고를 배경으로 숲에서부터 슬라임, 기사, 악마를 처리한다. 주인공 캐릭터는 졸라맨으로 그렸고 방향키를 사용하면 왼쪽, 오른쪽, 점프를 할 수 있으며 스페이스 바를 누르면 총알이 나가 공격을 할 수 있다. 게임에서 이기고 지고를 떠나 컴퓨터인공지능학부에 오면 파이썬 공부에서 벗어날 수 없다는 의미를 담고자 마지막 단계 최종 보스는 컴퓨터 과학과 코딩의 교수님을 그렸다. 따라서 4라운드는 사실상 보스를 처리할 수 없고 게임을 깰 수 없게 제작하였다.

3) 게임 세부 설정

몬스터는 왼쪽에서 오른쪽으로 케릭터에게 다가가며 몬스터가 왼쪽 벽으로 도달 시 다시 오른쪽으로 나오게끔 설정하였다. 1라운드는 슬라임 몬스터이며 총알 2번을 맞으면 처치 완료, 총 9마리를 죽이면 다음 라운드로 넘어갈 수 있고 2라운드는 기사 몬스터이며 총알 5번을 맞으면 처치 완료, 총 4마리를 죽이면 다음 라운드로 넘어갈 수 있다. 3라운드는 악마 몬스터이며 총알 9번을 맞으면 처치 완료, 총 2마리를 죽이면 다음 라운드로 넘어갈 수 있다. 몬스터를 앞 문장과 같이 적절하게 처리 시 더 이상 몬스터가 나오지 않게 되며 주인공이 오른쪽으로 걸어가 벽에 닿았을 때 다음 라운드 화면으로 배경이 전환되게 하였다. 주인공은 점프를 이용해 몬스터를 뛰어넘을 수도 있다. 마지막 4라운드는 홍득조 교수님이며 300번이상 공격을 받으면 게임이 종료되며 플레이어가 승리한다. 주인공의 목숨은 3개로 설정했으며 몬스터에게 닿을 때마다 목숨이 하나씩 줄어들도록 하고 플레이 중 목숨이 0이 되면 'GAME OVER' 표시가 뜨며 게임이 종료되고 승리 시 'YOU WIN' 표시가 뜨도록 구현했다.

2. 팀원 협력 정도

1) 역할 분배

일단 최대한 다 같이 게임 개발 및 레포트를 작성했다. 게임 개발 전 평일 저녁에 모여 아이디어를 구상하였고 게임 개발을 하기 위해 평일 저녁에 모여 다 같이 개발하였다. 이혜원은 전체적인 게임 이미지와 캐릭터 관련 코드를 위주로, 이서준과 장유빈은 몬스터 관련 코드를 위주로 담당했다. 레포트의 경우는 주제를 나눠 각자 하나씩 맡아 작성하기로 했다. 게임 제작 과정의 경우는 양이 많을 수 있어 세부적으로 나눠 작성했다.

2) 개임 개발 부분

처음 아이디어 회의 이후 각자 게임 개발에서 파트를 분배해 각자 역할을 맡아서 진행했다. 이혜원은 유저가 움직이는 캐릭터의 좌우로 움직이는 것과 UP 키를 누르면 캐릭터가 점프할 수 있도록 만들었고 게임에 필요한 이미지를 그려 게임

이미지의 전체적인 틀을 잡았다, 이서준은 몬스터가 캐릭터를 향해 오른쪽으로 움직이고, 각각의 몬스터가 랜덤한 속도와 텀을 가지며 나오게 했다. 장유빈은 이혜원과 함께 게임의 전체적인 틀을 구성하였고, 나머지 둘의 캐릭터와 몬스터의 기본적인 틀이 완성되면 각각의 세부적인 부분을 맡기로 했다. 게임의 기본적인 틀이 구성되고 난 후 장유빈이 캐릭터가 몬스터를 스페이스 바로 공격할 수 있게 만들고 공격당하면 몬스터가 사라지게 했다. 그 후 이혜원은 screen에 목숨이 표시되도록 하고 캐릭터가 몬스터에 닿았을 때 목숨이 하나씩 사라지는 코드를 작성했다. tkinter를 활용한 게임 시작하는 부분을 추가로 만들었다. 이서준은 몬스터에 관한 세부 설정을 맡았고 몬스터의 목숨 설정과 몬스터가 왼쪽 벽에 도달하면 다시 오른쪽에서 나타나도록 만들었다. 이후 게임 개발을 하면서 게임의 전체적인 틀, 캐릭터, 몬스터로 크게 나눠 개발을 하고 세부적인 부분들은 능동적으로 나눠서 개발을 진행했다.

3) 레포트 작성

레포트는 우선 목차를 나눴다. 상대적으로 양이 적은 주제들은 각자 하나씩 맡아서 작성하였고, 게임 제작 과정의 경우에는 자신이 직접 만든 부분을 위주로 작성하였다.

3.게임 제작 과정

1) 코드 설명

```
import tkinter as tk
from PIL import ImageTk, Image

def key_press(event):
    if event.keysym == 'space':
        start_game()
```

```
# Tkinter 창 생성
root = tk.Tk()
root.title("Game Start")
```

tkinter, 객체 활용

tkinter를 사용하여 GUI 창을 생성하고, 창의 제목을 "Game Start"로 설정하는 부분이다. root라는 변수에 창 객체를 할당하고 title()메서드를 사용하여 창의 제목을 "Game Start"로 설정했다. 위의 코드를 실행하면 "Game Start"라는 제목을 가진 tkinter 창이 생성된다.

```
# 창 크기 설정
window_width = 800
window_height = 600
root.geometry(f"{window_width}x{window_height}")
```

tkinter 활용

tkinter를 사용하여 GUI 창의 크기를 설정하는 부분이다. window_width와 window_height 변수를 사용하여 창의 너비와 높이를 지정하면 창의 너비는 800 픽셀, 높이가 600 픽셀로 설정했다. geometry()메서드를 사용하여 창의 크기를 설정하면 root.geometry("800x600")와 같이 창의 초기 크기가 800x600 픽셀로 설정된다.

```
# 프레임 생성
frame = tk.Frame(root)
frame.pack(pady=50)
```

tkinter, 객체 활용

tkinter에서 프레임을 생성하고, 생성된 프레임을 창에 배치하는 부분이다. tk.Frame()을 사용하여 frame이라는 변수에 새로운 프레임 객체를 생성했다. pack()메서드를 사용하여 프레임을 창에 배치하면 pady=50으로 수직 방향 50 픽셀의 여백을 가진 상태로 프레임을 배치된다.

```
# 시작 화면 이미지 로드
image = Image.open("start_image.png")
image = image.resize((500, 300), Image.LANCZOS)
photo = ImageTk.PhotoImage(image)
```

tkinter, .png, 객체 활용

시작 화면 이미지를 로드하고, 크기를 조정한 후 tkinter에서 사용할 수 있는 형식으로 변환하는 부분이다. PIL의 Image.open()함수를 사용하여 "start_image.png" 파일을 열고 resize()메서드를 사용하여 이미지의 크기를 조정했다. Image.LANCZOS는 리샘플링 필터를 지정하는 매개변수로, 이미지 크기 조정에 사용했다. PIL의 ImageTk.PhotoImage()함수를 사용하여 PIL 이미지 객체를 tkinter에서 사용할 수 있는 형식인 PhotoImage객체로 변환하고 photo변수에 이 PhotoImage객체가 할당되었다. 위의 코드를 실행하면 "start_image.png" 이미지 파일을 읽어와서 크기를 조정한 후, photo변수에 변환된 이미지 객체가 저장된다.

```
# 이미지 라벨 생성
image_label = tk.Label(frame, image=photo)
image_label.pack()
```

tkinter, 객체 활용

프레임에 이미지를 표시하기 위해 이미지 라벨을 생성하고, 이미지를 라벨에 배치하는 부분이다. tk.Label()을 사용하여 image_label이라는 변수에 새로운 이미지 라벨 객체를 생성했다. image=photo는 이미지 라벨에 표시할 이미지로 photo변수에 저장된 PhotoImage객체를 사용하고 pack()메서드를 사용하여 이미지 라벨을 프레임에 배치했다. 위의 코드를 실행하면 image_label변수에 새로운 이미지 라벨 객체가 생성되고, 해당 이미지가 라벨에 배치된다.

```
# 텍스트 라벨 생성
text_label = tk.Label(frame, text="Press Space Bar to Start", font=("Arial", 24))
text_label.pack(pady=60)
```

tkinter 활용

프레임에 텍스트를 표시하기 위해 텍스트 라벨을 생성하고, 텍스트를 라벨에 배치하는 부분이다. tk.Label()을 사용하여 text_label이라는 변수에 새로운 텍스트 라벨 객체를 생성하고 frame은 text="Press Space Bar to Start"는 텍스트 라벨에 표시될 텍스트 내용을 지정했다. pack()메서드를 사용하여 텍스트 라벨을 프레임에 배치하면 텍스트 라벨이 프레임 내에서 적절한 위치에 표시되며 위 아래로 60 픽셀의 여백이 생긴다. 위의 코드를 실행하면 "Press Space Bar to Start"라는 텍스트가 프레임 내에서 보여지게 된다.

```
# 스페이스 바 이벤트 핸들러
def key_press(event):
    if event.keysym == 'space':
        root.destroy()
```

이벤트, def, if 조건문, 함수 활용

key_press()라는 함수를 정의하고, 이 함수가 호출될 때 키 이벤트를 처리하는 부분이다. key_press라는 함수를 정의하고 event라는 매개변수를 받았다. event.keysym을 사용하여 사용자가 누른 키의 이름을 확인하고 destroy()메서드를 사용하여 root창을 닫았다. 위의 코드를 실행하면 스페이스 바를 눌러서 게임을 시작할 수 있도록 구현했다.

```
# 스페이스 바 이벤트 바인딩
root.bind("<space>", key_press)
```

함수 활용

<space>키에 대한 바인딩을 설정하여, 해당 키가 눌렸을 때 key_press()함수를 호출하는 부분이다. root창에 대한 이벤트 바인딩을 설정하는 코드로 key_press는 바인딩 된 이벤트가 발생했을 때 호출될 함수를 지정했다. 위의 코드를 실행하면 root창이 열려있을 때 스페이스 바 키를 누르면 key_press()함수가 호출하게 된다.

```
# Tkinter 창 실행
root.mainloop()
```

이벤트 루프 활용

root.mainloop()는 tkinter의 이벤트 루프를 실행하는 코드로 root.mainloop()를 호출해 tkinter 창이 열리게 했다.

```
import pygame
import random
import os

# 게임 초기화
pygame.init()
```

함수 활용

pygame.init()은 pygame 라이브러리를 초기화하는 함수입니다.

```
score = 0
FPS = 60
clock = pygame.time.Clock()
```

```
# 게임 화면 설정
screenWidth = 1300
screenHeight = 700
screen = pygame.display.set_mode((screenWidth, screenHeight))
pygame.display.set_caption("Kill the Monster")
```

함수 활용

게임 창을 생성하고 창의 크기와 제목을 설정하는 부분이다. 게임 창의 가로 크기를 1300 픽셀, 세로 크기를 700 픽셀로 설정하는 변수를 생성했다. pygame.display.set_mode()함수를 사용하여 게임 창을 만들고 생성된 게임 창은 screen변수에 할당했으며 pygame.display.set_caption()함수를 사용하여 게임 창의 제목을 설정했다.

```
# 배경 이미지 로드
background1 = pygame.image.load("tree.png")
background2 = pygame.image.load("castle.png")
background3 = pygame.image.load("hell.png")
background4 = pygame.image.load("python.png")
background_list = [background1, background2, background3, background4]
current_background = 0
```

객체, .png 활용

게임에서 사용할 여러 배경 이미지를 로드하고, 이미지 객체를 리스트에 저장하는 부분이다. background1, background2, background3, background4변수를 리스트에 저장하여 background_list변수에 할당하고 배경 이미지들이 순서대로 background_list에 저장했다. 위의 코드를 실행하면, "tree.png", "castle.png", "hell.png", "python.png"의 네 가지 배경 이미지 파일을 로드하고, 해당 이미지 객체가 리스트에 저장된다.

```
#캐릭터 이미지 로드
character = pygame.image.load("hi.png")
character_size = character.get_rect().size
character_right = pygame.image.load("hi.png")
character_left = pygame.transform.flip(character_right, True, False)
character_width = character_size[0]
character_height = character_size[1]
character_x_pos = (screenWidth / 2) - (character_width / 6)
character_y_pos = screenHeight - character_height
```

.png 활용

"hi.png" 파일을 로드하여 character변수에 캐릭터 이미지를 저장했다. 캐릭터의 크기는 character_size 변수에 저장

했고 오른쪽을 보는 이미지는 character_right에 저장되며 character_left 변수는 character_right 이미지를 좌우로 뒤집은 이미지로 초기화된다. 캐릭터의 초기 위치는 character_x_pos와 character_y_pos 변수에 설정했다.

```
#몬스터 이미지 로드
monster1 = pygame.image.load("slime1.png")
monster2 = pygame.image.load("slime2.png")
monster3 = pygame.image.load("slime3.png")
monster_list = [monster1, monster2, monster3]
monsters = []
monster4 = pygame.image.load("iron.png")
monster5 = pygame.image.load("devil.png")
monster6 = pygame.image.load("hong.png")
```

.png, 객체 활용

게임에서 사용할 여러 몬스터 이미지를 로드하고 몬스터 이미지 객체를 리스트에 저장하는 부분이다.
위의 코드를 실행하면 "slime1.png", "slime2.png", "slime3.png"의 세 가지 몬스터 이미지 파일을 로드하고, 해당 이미지 객체를 리스트에 저장한다. 추가로 "iron.png", "devil.png", "hong.png" 등의 이미지도 로드하여 몬스터 이미지에 추가했다.

```
# 공격 이미지 불러오기
bullet = pygame.image.load("bullet.png")
bullet_width = 40  # 원하는 너비로 설정
bullet_height = 20  # 원하는 높이로 설정
bullet = pygame.transform.scale(bullet, (bullet_width, bullet_height))
bullet_x_pos = 0
bullet_y_pos = 0
bullet_speed = 10
bullet_state = "ready“
```

.png, 함수 활용

총알 이미지가 로드되고, 총알의 크기와 초기 상태를 설정하는 부분이다. "bullet.png" 파일을 pygame.image.load()함수를 사용하여 로드하여 bullet변수에 저장했다. bullet이미지를 지정된 너비와 높이로 변환하고 pygame.transform.scale()함수를 사용하여 이미지 크기를 조정했다. 위의 코드를 실행하면, "bullet.png" 파일을 로드하여 bullet변수에 총알 이미지가 저장된다. 총알의 너비와 높이는 bullet_width와 bullet_height변수에 설정되며 이미지 크기 조정을 위해 pygame.transform.scale()함수를 사용하여 총알 이미지의 크기를 조정했다. 총알의 초기 좌표는 bullet_x_pos와 bullet_y_pos변수에 설정했으며, 총알의 이동 속도는 bullet_speed변수에 저장된다. 총알의 초기 상태는 "ready"로 설정했다.

```
# 폰트 파일 경로
font_path = os.path.join("폰트_파일_경로.ttf")
```

함수 활용

폰트 파일의 경로를 설정하는 부분이다. os.path.join()함수를 사용하여 폰트 파일의 경로를 설정했다. 위의 코드에서 os.path.dirname(__file__)은 현재 스크립트 파일의 디렉토리 경로를 반환했다.

```
# 폰트 초기화
pygame.font.init()
game_font = pygame.font.Font(font_path, 36)
```

객체, 함수 활용

폰트를 초기화하고, 폰트 객체를 생성하는 부분이다. pygame.font.init() 함수를 호출하여 폰트를 사용하기 전에 초기화했다. pygame.font.Font()함수를 사용하여 폰트 객체를 생성했다.

```
#목숨 표시 설정
life = 3
life_text = game_font.render("Life: {}".format(life), True, (255, 255, 255))
life_text_rect = life_text.get_rect()
life_text_rect.topleft = (10, 10)

# 목숨 텍스트 렌더링 함수
def render_life_text():
    life_text = game_font.render("Life: {}".format(life), True, (255, 255, 255))
    life_text_rect = life_text.get_rect()
    life_text_rect.topleft = (10, 10)
    screen.blit(life_text, life_text_rect)
```

def 활용

life 변수로 현재 사용자의 목숨 개수를 저장했다. life_text로 목숨 개수를 표시하는 텍스트를 렌더링한 결과를 저장했고 render_life_text() 함수로 목숨 텍스트를 렌더링하고 화면에 표시했다.

```
# 이벤트 루프
character_speed = 5
character_jump = False
jump_count = 12

spawn_interval = 3000  # 몬스터 생성 간격(밀리초)
```

```
last_spawn_time = pygame.time.get_ticks()  # 마지막 몬스터 생성 시간
monster_count1 = 0  # 생성된 몬스터 개수
monster_count2 = 0
monster_count3 = 0
monster_count4 = 0

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False
```

for문, if 조건문 활용

```
    if character_x_pos > screenWidth and monster_count1 >= 10:
        character_x_pos = -character_width
        current_background = (current_background + 1) % len(background_list)

    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        if character_x_pos > 0:  # 왼쪽 벽에 닿았을 때만 움직임
            character_x_pos -= character_speed
            character = character_left  # 캐릭터 반전

    if keys[pygame.K_RIGHT]:
        character_x_pos += character_speed
        character = character_right  # 캐릭터 원래 방향으로

    if keys[pygame.K_UP] and not character_jump:
        character_jump = True

    if keys[pygame.K_SPACE]:
        if bullet_state == "ready":
            bullet_x_pos = character_x_pos + character_width / 2 - bullet_width / 2
            bullet_y_pos = character_y_pos + character_height / 2 - bullet_height / 2
            bullet_state = "fire"
            if keys[pygame.K_LEFT]:
                bullet_speed = -15
            else:
                bullet_speed = 15
```

```
    if character_jump:
        if jump_count >= -12:
            neg = 1
            if jump_count < 0:
                neg = -1
            character_y_pos -= (jump_count ** 2) * 0.5 * neg
            jump_count -= 1
        else:
            character_jump = False
            jump_count = 12
```

if 조건문 활용

왼쪽 방향키가 눌렸을 경우 캐릭터가 왼쪽으로 이동하게 했다. 오른쪽 방향키가 눌렸을 경우 캐릭터가 오른쪽으로 이동하게 했고 위쪽 방향키가 눌렸으며 캐릭터가 점프 중이 아닐 경우 캐릭터의 점프 동작을 시작하게 설정했다.

```
    current_time = pygame.time.get_ticks()
```

함수 활용

```
    #1단계 몬스터
    if current_background == 0 and monster_count1 < 10 and current_time - last_spawn_time > spawn_interval:
        last_spawn_time = current_time
        monster_x_pos = screenWidth
        monster_y_pos = screenHeight - monster1.get_height()  # 바닥에 붙이기 위해 y 좌표 설정
        monster = {
            "image": random.choice(monster_list),
            "x": monster_x_pos,
            "y": monster_y_pos,
            "speed": random.randint(4, 6),
            "collision_count": 0
        }
        monsters.append(monster)
        monster_count1 += 1


        #2단계 몬스터
    if current_background == 1 and monster_count2 < 5 and current_time - last_spawn_time > spawn_interval:
        last_spawn_time = current_time
        monster_x_pos = screenWidth
```

```
        monster_y_pos = screenHeight - monster4.get_height()  # 바닥에 붙이기 위해 y 좌표 설정
        monster = {
            "image": monster4,
            "x": monster_x_pos,
            "y": monster_y_pos,
            "speed":random.randint(2, 6),
            "collision_count": 0
        }
        monsters.append(monster)
        monster_count2 += 1

        #3단계 몬스터
    if current_background == 2 and monster_count3 < 3 and current_time - last_spawn_time > spawn_interval:
        last_spawn_time = current_time
        monster_x_pos = screenWidth
        monster_y_pos = screenHeight - monster5.get_height()  # 바닥에 붙이기 위해 y 좌표 설정
        monster = {
            "image": monster5,
            "x": monster_x_pos,
            "y": monster_y_pos,
            "speed": random.randint(2, 6),
            "collision_count": 0
        }
        monsters.append(monster)
        monster_count3 += 1

        #4단계 몬스터
    if current_background == 3 and monster_count4 < 1 and current_time - last_spawn_time > spawn_interval:
        last_spawn_time = current_time
        monster_x_pos = screenWidth
        monster_y_pos = screenHeight - monster6.get_height()  # 바닥에 붙이기 위해 y 좌표 설정
        monster = {
        "image": monster6,
        "x": monster_x_pos,
        "y": monster_y_pos,
        "speed": 2,
        "collision_count": 0
        }
        monsters.append(monster)
        monster_count4 += 1
```

if 조건문, 랜덤문, 함수 활용

n단계 몬스터를 생성하는 부분이다. n단계 몬스터의 생성 개수가 m개 미만인지 확인하고 이전 몬스터 생성 이후로 일정 시간이 경과했는지 확인하면 새로운 몬스터를 생성하게 된다. 랜덤하게 선택한 몬스터 이미지를 가져오고 몬스터들이 monsters 리스트에 추가되면 monster_count(n) 변수를 증가시킨다.

```
clock.tick(FPS)
```

게임 루프의 실행 속도를 제어하는 부분이다. FPS 변수는 초당 프레임 수를 나타내는 값이며, clock.tick(FPS)로 게임 루프가 초당 지정된 프레임 수로 실행되도록 설정했다.

```
screen.blit(background_list[current_background], (0, 0))
screen.blit(character, (character_x_pos, character_y_pos))
```

함수, 객체 활용

배경 이미지와 캐릭터 이미지를 화면에 그리는 부분이다. background_list는 배경 이미지들을 담고 있는 리스트이고, current_background 변수를 사용하여 현재 선택된 배경을 가져왔다. screen.blit() 함수를 사용하여 배경 이미지를 화면 좌상단 (0, 0) 위치에 그렸다. haracter_x_pos와 character_y_pos 변수는 캐릭터의 x 좌표와 y 좌표를 나타내며, screen.blit() 함수를 사용하여 캐릭터 이미지를 해당 좌표에 그렸다.

```
for monster in monsters:
    screen.blit(monster["image"], (monster["x"], monster["y"]))
    monster["x"] -= monster["speed"]
    if monster["x"] + monster["image"].get_width() < 0 or monster["x"] > screenWidth:
        monster["x"] = screenWidth
```

for 반복문, if 조건문 활용

monsters리스트에 있는 각 몬스터의 위치를 업데이트하고 화면에 그리는 부분이다 monsters리스트에 있는 각 몬스터에 대해서 반복했고 if monster["x"] + monster["image"].get_width() < 0 or monster["x"] > screenWidth:을 사용해서 만약 몬스터가 화면 왼쪽 경계를 넘어가거나 화면 오른쪽 경계를 넘어간 경우라면, 몬스터의 x 좌표를 화면 오른쪽 끝으로 설정하여 다시 나타나도록 했다.

```
# 총알과 몬스터 충돌 체크
        monster_rect = pygame.Rect(monster["x"], monster["y"], monster["image"].get_width(), monster["image"].get_height())
        bullet_rect = pygame.Rect(bullet_x_pos, bullet_y_pos, bullet_width, bullet_height)
        if monster_rect.colliderect(bullet_rect):
            monster["collision_count"] += 1
            if current_background == 0 and monster["collision_count"] >= 3:
                monsters.remove(monster)
```

```
            elif current_background == 1 and monster["collision_count"] >= 7:
                monsters.remove(monster)
            elif current_background == 2 and monster["collision_count"] >= 12:
                monsters.remove(monster)
            elif current_background == 3 and monster["collision_count"] >= 30:
                monsters.remove(monster)
```

객체, if 조건문 활용

충돌을 감지하고, 충돌한 경우 몬스터를 제거하는 부분이다. monster딕셔너리에 저장된 몬스터의 좌표와 크기를 기반으로 몬스터의 충돌용 사각형 객체 monster_rect를 생성했다. 총알의 좌표와 크기를 기반으로 총알의 충돌용 사각형 객체 bullet_rect를 생성하고 monster_rect와 bullet_rect사이의 충돌을 검사한다. 두 사각형이 충돌하면 조건문 내부의 코드가 실행되고 monster["collision_count"] += 1은 몬스터의 충돌 횟수를 증가시킨다. 조건문 내부의 if-elif 블록이 현재 배경(background)에 따라 충돌 횟수를 검사하고, 일정 횟수 이상 충돌한 경우 해당 몬스터를 monsters 리스트에서 제거하게 했다. 몬스터와 총알의 충돌을 감지하여 충돌이 발생하면 해당 몬스터의 충돌 횟수를 증가시키고, 일정 횟수 이상 충돌한 경우 해당 몬스터를 monsters리스트에서 제거하는 동작을 구현했다.

```
            bullet_state = "ready"
            bullet_x_pos = 0  # 총알의 x 좌표 초기화
            bullet_y_pos = 0  # 총알의 y 좌표 초기화
    if bullet_state == "fire":
        screen.blit(bullet, (bullet_x_pos, bullet_y_pos))
        bullet_x_pos += bullet_speed
        if bullet_x_pos > screenWidth or bullet_x_pos < 0:
            bullet_state = "ready“
```

if 조건문, 함수 활용

총알을 발사하고, 총알이 화면을 벗어나면 다시 초기화하는 부분이다. 총알 상태를 "ready"로 초기화합니다. if bullet_state == "fire":는 총알 상태가 "fire"인 경우에만 총알을 화면에 그리게 했다. 총알 이미지를 지정된 좌표(bullet_x_pos, bullet_y_pos)에 그리고 screen.blit()함수는 이미지를 화면에 그리는 역할을 한다. if bullet_x_pos > screenWidth or bullet_x_pos < 0:는 총알이 화면을 벗어나는 경우를 체크하게 했다. 만약 총알이 화면의 오른쪽 경계를 벗어나거나 왼쪽 경계를 벗어난다면, 총알 상태를 "ready"로 변경한다. 총알을 발사하고, 총알이 화면을 벗어나면 다시 초기화하는 로직을 구현했다.

```
if current_background == 3 and monster["collision_count"] >= 30:
                    monsters.remove(monster)
                    game_over_image = pygame.image.load("youwin.png")
                    screen.blit(game_over_image, (0, 0))
                pygame.display.flip()
                 pygame.time.delay(2000)  # 2초간 대기
                running = False  # 게임 종료
```

if 조건문, .png 활용

현재 배경이 3인 경우에 몬스터의 충돌 횟수가 30 이상인 경우 처리하는 부분이다. 해당 몬스터를 monsters리스트에서 제거하고 "youwin.png" 이미지를 게임 오버 이미지로 로드하고 pygame.display.flip(): 화면을 업데이트하여 게임 오버 이미지를 표시했다. 게임을 종료하기 위해 running변수를 False로 설정했다. 위의 코드는 현재 배경이 3이고 몬스터의 충돌 횟수가 30 이상인 경우에 해당하는 처리를 수행한다. 해당 몬스터를 제거하고 게임 오버 이미지를 화면에 표시한 뒤, 2초간 대기한 후 게임을 종료한다.

```
    # 몬스터와 캐릭터 충돌 검사
    character_rect = pygame.Rect(character_x_pos, character_y_pos, character_width, character_height)
    collision = False  # 충돌 여부 확인 변수
    for monster in monsters:
        monster_rect = pygame.Rect(monster["x"], monster["y"], monster["image"].get_width(),
monster["image"].get_height())
        if character_rect.colliderect(monster_rect):
            collision = True
            break
```

for 반복문, if 조건문 활용

주인공 캐릭터와 몬스터 간의 충돌을 체크하는 부분이다. 충돌 박스는 주인공 캐릭터의 현재 위치와 크기를 바탕으로 생성됩니다. 충돌 여부를 확인하는 변수를 초기화한다. monsters리스트에 있는 각각의 몬스터에 대해 반복하고 if character_rect.colliderect(monster_rect):는 주인공 캐릭터의 충돌 박스와 몬스터의 충돌 박스가 서로 겹치는지 체크하게 했다. 충돌이 발생했으므로 collision변수를 True로 설정했고 break: 부분은 충돌이 발생한 경우 반복문을 종료하게 했다. 위의 코드는 주인공 캐릭터와 몬스터 간의 충돌을 체크하고 주인공 캐릭터와 몬스터의 충돌 박스를 생성하며 두 충돌 박스가 겹치는지 확인하여 충돌이 발생한 경우를 대비해 collision변수를 True로 설정했다.

```
    render_life_text()  # 목숨 텍스트 렌더링
```

함수 활용

render_life_text()함수를 호출해 주어진 생명 개수를 텍스트로 표시했다.

```
 #충돌 검사 후 목숨 처리

    if collision:
        life -= 1
        if life == 0:
            game_over_image = pygame.image.load("gameover.png")
            screen.blit(game_over_image, (0, 0))
            pygame.display.flip()
```

```
            pygame.time.delay(2000)  # 2초간 대기
            running = False  # 목숨이 0 게임 종료
        else:
            character_x_pos = (screenWidth / 2) - (character_width / 6)  # 캐릭터 초기 위치로 이동
            monsters.clear()  # 몬스터 초기화

    life_text_surface = game_font.render("Life: {}".format(life), True, (255, 255, 255)) # 목숨 텍스트 렌더링
    screen.blit(life_text_surface, (10, 10))  # 목숨 텍스트 그리기

    pygame.display.flip()

# 게임 종료
pygame.quit()
```

if 조건문, .png 활용

충돌이 발생했을 때의 처리를 담당하는 부분이다. if life == 0은 생명 개수가 0인 경우 실행되고 game_over_image = pygame.image.load("gameover.png"):는게임 오버 이미지를 로드하게 했다. 게임 실행을 종료하기 위해 running 변수를 False로 설정했고 else:는 생명 개수가 0이 아닌 경우 실행된다. 생명 개수가 0이 아니라면 캐릭터를 초기 위치로 이동하고, 몬스터들을 초기화한다. 위의 코드는 충돌이 발생했을 때 생명 개수를 감소시키고, 생명 개수에 따라 게임 오버 또는 게임을 계속 진행하는 처리를 수행하며 생명 개수를 텍스트로 표시하여 화면에 렌더링한다.

2) 챗 gpt 사용

챗 gpt는 먼저 전체적인 틀을 짠 후에 게임 제작 중 필요한 세부사항을 만들 때 이용하였다

2.1) 주인공 캐릭터
캐릭터를 방향 키로 조절하고 스페이스바를 누르면 지정한 이미지의 공격을 일정한 간격으로 발사하는 코드와 몬스터와 닿을 때 목숨이 닳는 코드를 만들 때 사용했다.

2.2) 몬스터
몬스터들이 각 단계마다 다르게 나오고 단계별 몬스터들이 입는 대미지를 정해서 일정 공격을 받고 없어지게 하는 코드와 랜덤 한 속도로 등장하는 코드를 만들 때 사용했다.

2.3) 세부사항 및 화면 표시
게임이 종료될 수 있는 경우를 게임이 끝나기 전 목숨이 닳아 게임이 끝날 때와 4단계 몬스터를 죽여 게임이 끝나는 2가지로 나눠 경우에 따라 다른 화면이 나오는 코드와 단계별 몬스터들을 다 죽이면 다음 단계로 넘어가 배경과 몬스터들이 바뀌어 등장하게 하는 코드를 만들 때 사용했다.

4. 최종 구현물

```
import tkinter as tk
from PIL import ImageTk, Image

def key_press(event):
    if event.keysym == 'space':
        start_game()

root = tk.Tk()
root.title("Game Start")

window_width = 800
window_height = 600
root.geometry(f"{window_width}x{window_height}")

frame = tk.Frame(root)
frame.pack(pady=50)

image = Image.open("start_image.png")
image = image.resize((500, 300), Image.LANCZOS)
photo = ImageTk.PhotoImage(image)


image_label = tk.Label(frame, image=photo)
image_label.pack()


text_label = tk.Label(frame, text="Press Space Bar to Start", font=("Arial", 24))
text_label.pack(pady=60)

def key_press(event):
    if event.keysym == 'space':
        root.destroy()

root.bind("<space>", key_press)

root.mainloop()

import pygame
import random
import os

pygame.init()

score = 0
FPS = 60
clock = pygame.time.Clock()

screenWidth = 1300
screenHeight = 700
screen = pygame.display.set_mode((screenWidth, screenHeight))
pygame.display.set_caption("Kill the Monster")
```

```
background1 = pygame.image.load("tree.png")
background2 = pygame.image.load("castle.png")
background3 = pygame.image.load("hell.png")
background4 = pygame.image.load("python.png")
background_list = [background1, background2, background3, background4]
current_background = 0


character = pygame.image.load("hi.png")
character_size = character.get_rect().size
character_right = pygame.image.load("hi.png")
character_left = pygame.transform.flip(character_right, True, False)
character_width = character_size[0]
character_height = character_size[1]
character_x_pos = (screenWidth / 2) - (character_width / 6)
character_y_pos = screenHeight - character_height

monster1 = pygame.image.load("slime1.png")
monster2 = pygame.image.load("slime2.png")
monster3 = pygame.image.load("slime3.png")
monster_list = [monster1, monster2, monster3]
monsters = []
monster4 = pygame.image.load("iron.png")
monster5 = pygame.image.load("devil.png")
monster6 = pygame.image.load("hong.png")

bullet = pygame.image.load("bullet.png")
bullet_width = 40
bullet_height = 20
bullet = pygame.transform.scale(bullet, (bullet_width, bullet_height))
bullet_x_pos = 0
bullet_y_pos = 0
bullet_speed = 10
bullet_state = "ready“

font_path = os.path.join("폰트_파일_경로.ttf")

pygame.font.init()
game_font = pygame.font.Font(font_path, 36)

life = 3
life_text = game_font.render("Life: {}".format(life), True, (255, 255, 255))
life_text_rect = life_text.get_rect()
life_text_rect.topleft = (10, 10)

def render_life_text():
    life_text = game_font.render("Life: {}".format(life), True, (255, 255, 255))
    life_text_rect = life_text.get_rect()
    life_text_rect.topleft = (10, 10)
    screen.blit(life_text, life_text_rect)

character_speed = 5
character_jump = False
jump_count = 12
```

```
spawn_interval = 3000
last_spawn_time = pygame.time.get_ticks()
monster_count1 = 0
monster_count2 = 0
monster_count3 = 0
monster_count4 = 0

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False


    if character_x_pos > screenWidth and monster_count1 >= 10:
        character_x_pos = -character_width
        current_background = (current_background + 1) % len(background_list)

    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT]:
        if character_x_pos > 0:
            character_x_pos -= character_speed
            character = character_left

    if keys[pygame.K_RIGHT]:
        character_x_pos += character_speed
        character = character_right

    if keys[pygame.K_UP] and not character_jump:
        character_jump = True

    if keys[pygame.K_SPACE]:
        if bullet_state == "ready":
            bullet_x_pos = character_x_pos + character_width / 2 - bullet_width / 2
            bullet_y_pos = character_y_pos + character_height / 2 - bullet_height / 2
            bullet_state = "fire"
            if keys[pygame.K_LEFT]:
                bullet_speed = -15
            else:
                bullet_speed = 15

    if character_jump:
        if jump_count >= -12:
            neg = 1
            if jump_count < 0:
                neg = -1
            character_y_pos -= (jump_count ** 2) * 0.5 * neg
            jump_count -= 1
        else:
            character_jump = False
            jump_count = 12

    current_time = pygame.time.get_ticks()
```

```
    if current_background == 0 and monster_count1 < 10 and current_time - last_spawn_time
> spawn_interval:
        last_spawn_time = current_time
        monster_x_pos = screenWidth
        monster_y_pos = screenHeight - monster1.get_height()
        monster = {
            "image": random.choice(monster_list),
            "x": monster_x_pos,
            "y": monster_y_pos,
            "speed": random.randint(4, 6),
            "collision_count": 0
        }
        monsters.append(monster)
        monster_count1 += 1

    if current_background == 1 and monster_count2 < 5 and current_time - last_spawn_time
> spawn_interval:
        last_spawn_time = current_time
        monster_x_pos = screenWidth
        monster_y_pos = screenHeight - monster4.get_height()
        monster = {
            "image": monster4,
            "x": monster_x_pos,
            "y": monster_y_pos,
            "speed":random.randint(2, 6),
            "collision_count": 0
        }
        monsters.append(monster)
        monster_count2 += 1

    if current_background == 2 and monster_count3 < 3 and current_time - last_spawn_time
> spawn_interval:
        last_spawn_time = current_time
        monster_x_pos = screenWidth
        monster_y_pos = screenHeight - monster5.get_height()
        monster = {
            "image": monster5,
            "x": monster_x_pos,
            "y": monster_y_pos,
            "speed": random.randint(2, 6),
            "collision_count": 0
        }
        monsters.append(monster)
        monster_count3 += 1

        #4단계 몬스터
    if current_background == 3 and monster_count4 < 1 and current_time - last_spawn_time
> spawn_interval:
        last_spawn_time = current_time
        monster_x_pos = screenWidth
        monster_y_pos = screenHeight - monster6.get_height()
        monster = {
        "image": monster6,
        "x": monster_x_pos,
        "y": monster_y_pos,
        "speed": 2,
```

```
            "collision_count": 0
            }
            monsters.append(monster)
            monster_count4 += 1


    clock.tick(FPS)

    screen.blit(background_list[current_background], (0, 0))

    screen.blit(character, (character_x_pos, character_y_pos))

    for monster in monsters:
        screen.blit(monster["image"], (monster["x"], monster["y"]))

        monster["x"] -= monster["speed"]

        if monster["x"] + monster["image"].get_width() < 0 or monster["x"] > screenWidth:
            monster["x"] = screenWidth

    for monster in monsters:
        screen.blit(monster["image"], (monster["x"], monster["y"]))
        monster["x"] -= monster["speed"]
        if monster["x"] + monster["image"].get_width() < 0 or monster["x"] > screenWidth:
            monster["x"] = screenWidth


        monster_rect = pygame.Rect(monster["x"], monster["y"], monster["image"].get_width(),
monster["image"].get_height())
        bullet_rect = pygame.Rect(bullet_x_pos, bullet_y_pos, bullet_width, bullet_height)
        if monster_rect.colliderect(bullet_rect):
            monster["collision_count"] += 1
            if current_background == 0 and monster["collision_count"] >= 3:
                monsters.remove(monster)
            elif current_background == 1 and monster["collision_count"] >= 7:
                monsters.remove(monster)
            elif current_background == 2 and monster["collision_count"] >= 12:
                monsters.remove(monster)
            elif current_background == 3 and monster["collision_count"] >= 30:
                monsters.remove(monster)

            bullet_state = "ready"
            bullet_x_pos = 0
            bullet_y_pos = 0
    if bullet_state == "fire":
        screen.blit(bullet, (bullet_x_pos, bullet_y_pos))
        bullet_x_pos += bullet_speed
        if bullet_x_pos > screenWidth or bullet_x_pos < 0:
            bullet_state = "ready"


if current_background == 3 and monster["collision_count"] >= 30:
                monsters.remove(monster)
                game_over_image = pygame.image.load("youwin.png")
                screen.blit(game_over_image, (0, 0))
                pygame.display.flip()
```

```
                pygame.time.delay(2000)
                running = False


    character_rect = pygame.Rect(character_x_pos, character_y_pos, character_width, character_height)
    collision = False
    for monster in monsters:
        monster_rect = pygame.Rect(monster["x"], monster["y"], monster["image"].get_width(), monster["image"].get_height())
        if character_rect.colliderect(monster_rect):
            collision = True
            break

    render_life_text()

    if collision:
        life -= 1
        if life == 0:
            game_over_image = pygame.image.load("gameover.png")
            screen.blit(game_over_image, (0, 0))
            pygame.display.flip()
            pygame.time.delay(2000)
            running = False
        else:
            character_x_pos = (screenWidth / 2) - (character_width / 6)
            monsters.clear()

    life_text_surface = game_font.render("Life: {}".format(life), True, (255, 255, 255))
    screen.blit(life_text_surface, (10, 10))

    pygame.display.flip()

pygame.quit()
```

5. 느낀점

이서준

팀 프로젝트로 이 코드를 작성하면서 많은 것을 배우고 느끼게 되었습니다. 팀원들과의 협업을 통해 아이디어를 나누고 문제를 해결하는 과정에서 저의 역량을 발전시킬 수 있었습니다. 팀원들과 함께 코드를 작성하고 게임을 구성하는 과정에서 팀원들의 다양한 관점을 고려하여 게임을 완성시키는 것은 팀 프로젝트의 큰 장점이었습니다. 프로젝트의 업무 분담도 중요한 요소였습니다. 팀원들과의 원활한 소통과 조율을 통해 업무를 원활하게 진행할 수 있었습니다. 또한, 코드 공유와 피드백 과정은 저에게 많은 도움을 주었습니다. 팀원들이 작성한 코드를 살펴보고 개선점을 제안하며 서로의 실력을 향상시킬 수 있었습니다. 이러한 피드백 과정을 통해 더욱 효율적이고 완벽한 코드를 작성할 수 있었습니다. 마지막으로, 팀 프로젝트를 하며 협업과 소통의 중요성을 다시 한 번 깨닫게 되었습니다. 서로의 아이디어와 의견을 효과적으로 소통하며 문제를 해결하는 과정에서 팀원들과의 유대감을 형성할 수 있었습니다. 팀 프로젝트를 통해 동기들과 게임 개발에 대한 열정과 흥미를 나누고 함께 협력하는 과정에서 즐거움과 성취감을 느낄 수 있었습니다. 이러한 경험은 저에게 소중한 경험이 되었고, 앞으로의 프로젝트에서도 적극적으로 적용해 나갈 것입니다.

이혜원

첫 팀 프로젝트였기에 처음에 어떻게 해야 할지 막막했으나 팀원들과 조율해가며 역할도 나누고 소통하다보니 점점 감을 잡게 되었다. 혼자 했더라면 못했을 것들을 팀원들과 함께하며 아이디어를 공유하고 협업하여 게임을 개발하는 과정에서 소통과 협업의 중요성을 깨닫게 되었다. 그리고 수업에서 배운 내용들을 실제로 현실에 적용해보고 게임을 개발해봄으로써 코드에서 발생하는 오류를 찾고 수정하는 과정에서 문제 해결 능력과 디버깅 기술을 향상시키는 데 도움이 되었다. 오류를 해결하면서 새로운 기술과 개념도 함께 공부할 수 있었고 문제 해결에 대한 창의적인 사고를 많이 하게 된 거 같다. 또한 항상 플레이어 입장에서만 게임하다 개발자 입장이 돼 보며 우리가 평소에 보는 게임은 정말 높은 퀄리티였고 매우 전문적이었음을 알게 되었다. 피드백을 어떻게 수용해야 할지 또 어떻게 디자인해야 할지 등 개발자의 입장에서 바라보는 계기가 되며 개선 사항을 파악하고 반영하는 능력을 기를 수 있었다.

게임 제작은 흥미로운 도전이었고, 이를 통해 다양한 측면에서 성장할 수 있는 힘들었지만 좋은 경험이 된 거 같다. 앞으로도 더 도전할 미래에 대한 발판이 될 거 같다.

장유빈

파이썬을 이용해 게임을 만든다는 게 컴퓨터를 처음 접하고 아직 파이썬에 능숙하지 못한 나에게는 너무 큰 과제처럼 느껴졌고 과연 내가 잘할 수 있을지 의문이 들었다. 하지만 게임을 어떻게 만들지 구상하는 시간에 팀원들과 이야기하고 서로의 아이디어들 나누는 과정에서 흥미를 느꼈고 전체적인 틀을 짜니 왠지 게임 만들기가 재밌을 것 같다는 생각을 했다. 게임에서 필요한 코드들을 만들고 그 여러 개의 코드들을 모아서 게임을 실행해 내가 만든 캐릭터들이 내가 원하는 대로 움직이는 것 보니까 신기했다. 1부터 100까지 우리가 다 만들어야 해서 처음에는 쉽지 않았지만 수업 시간에 배웠던 내용들을 활용하고 챗 GPT와 함께 머리를 맞대서 코드를 짜다 보니까 게임에 세부적인 모션들도 넣을 수 있었고 몰랐던 개념들도 알게 돼서 유익한 시간이었던 것 같다. 직접 게임을 만들어 보니까 내가 그동안 해왔던 게임을 만드는 개발자들이 대단해 보였고 그들은 어떻게 게임을 만드는지 궁금해졌다. 그리고 팀원들과 함께 같은 작업물을 만든다는 것에서 동료애도 느껴졌다 서로의 의견과 생각을 공유하고 각자 코드에 필요한 부분을 채워주며 하나 하나 만들어 가는 과정에서 누구 하나 빠짐없이 열심히 해줘서 고마웠고 내가 부족하진 않았나 한편으로는 걱정도 됐다. 짧지 않은 시간 동안 만들어 보지 않은 결과물을 만들어 내는 게 어쩌면 막연하고 힘들 것 같다고 생각했지만 걱정과 달리 배운 것도 느낀 것들도 많아서 의미 있는 시간이었던 것 같다. 시간이 된다면 다른 언어와 프로그램들을 공부해서 색다르고 독창적인 게임들을 만들어 보고 싶다.