HIT137
SOFTWARE NOW

Assignment No: 03

Submitted By

Group No: **SYD 24**

Group Details –

Student Name	Student ID
MD SAKIBUL ISLAM	S390852
Jasia Binte Alam	S391195
Md Nurul Islam Chowdhury	S390923

Submitted To

Reem Sherif

Lecturer

Charles Darwin University

Faculty of Science & Technology, Engineering

30th May 2025

Answer to the question number 02

Animal Hero: Rabbit vs Dragon

1. Initial Setup (The Foundation)

```
import pygame
import random
import math
```

- pygame: The main library for making the game (handles graphics, sound, input).
- random: Used to make enemies spawn unpredictably.
- math: Helps with flying enemy movements (sine waves).

2. Game Window Setup

```
SCREEN_WIDTH = 800
SCREEN_HEIGHT = 600
screen = pygame.display.set_mode((SCREEN_WIDTH, SCREEN_HEIGHT))
pygame.display.set_caption("Animal Hero: Rabbit vs Dragon")
```

- SCREEN_WIDTH **&** SCREEN_HEIGHT: Sets the game screen size (800x600 pixels).
- screen: The actual game window where everything is drawn.
- set_caption: The title at the top of the window.

3. Loading Images & Sounds

```
background_image =
pygame.transform.scale(pygame.image.load("background.jpg").convert(),
(SCREEN_WIDTH, SCREEN_HEIGHT))
dragon_image = pygame.transform.scale(pygame.image.load("dragon.png").convert_alpha(),
(50, 50))
shoot_sound = pygame.mixer.Sound("shoot.wav")
shoot_sound.set_volume(0.5)
```

- background_image: Loads and resizes the background image to fit the screen.
- dragon_image: Loads the dragon sprite and makes it 50x50 pixels.
- shoot_sound: The sound effect when the player shoots.
- set_volume(0.5): Makes the sound quieter (so it's not too loud).

4. Colors & Constants

```
WHITE = (255, 255, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLACK = (0, 0, 0)
FPS = 60
GRAVITY = 1
```

- WHITE, RED, GREEN, BLACK: Color codes (RGB format).
- FPS = 60: The game runs at 60 frames per second (smooth movement).
- GRAVITY = 1: How fast the bunny falls after jumping.

5. Sprite Groups (Managing Game Objects)

```
all_sprites = pygame.sprite.Group()
projectiles = pygame.sprite.Group()
enemies = pygame.sprite.Group()
fireballs = pygame.sprite.Group()
```

- all_sprites: Tracks **everything** on screen (player, enemies, bullets).
- projectiles: Only tracks the **player's bullets**.
- enemies: Only tracks **dragons**.
- fireballs: Only tracks **enemy fire attacks**.

6. Game State Variables

```
level = 1
level_timer = 0
```

- level: Current level (1, 2, or 3).
- level_timer: Used in **Level 3** to track the 60-second time limit.

7. The Player Class (Bunny Hero!)

Initialization (__init__)

```
def __init__(self):
super().__init__()
self.image = pygame.image.load("human_with_gun.png").convert()
self.image.set_colorkey((255, 255, 255))  # Makes white background transparent
self.image = pygame.transform.scale(self.image, (50, 50))  # Resizes bunny
self.rect = self.image.get_rect()  # Hitbox for collisions
self.rect.x = 100  # Starting X position
self.rect.y = SCREEN_HEIGHT - 70  # Starting Y position (near bottom)
self.speed_x = 0  # Left/Right movement speed
self.speed_y = 0  # Up/Down (jumping/falling)
self.is_jumping = False  # Prevents double jumps
self.health = 100  # Full health at start
self.lives = 3  # 3 tries before Game Over
self.score = 0  # Starts at 0
```

- self.image: The bunny sprite (loaded from a file).
- self.rect: A rectangle that defines the bunny's position and collision area.
- self.speed_x **&** self.speed_y: How fast the bunny moves left/right and up/down.
- self.health: Starts at 100 (shown as a green bar).
- self.lives: You get 3 lives before Game Over.

Movement & Jumping

```
def update(self):
    self.rect.x += self.speed_x  # Move left/right
    self.speed_y += GRAVITY  # Apply gravity (falling)
    self.rect.y += self.speed_y  # Move up/down
    if self.rect.y > SCREEN_HEIGHT - 70:  # Stop at ground
        self.rect.y = SCREEN_HEIGHT - 70
        self.is_jumping = False
        self.speed_y = 0
    self.rect.x = max(0, min(SCREEN_WIDTH - self.rect.width, self.rect.x))  # Prevent going
off-screen
```

```
def jump(self):
    if not self.is_jumping:  # Only jump if on ground
        self.speed_y = -15  # Negative Y = upward jump
        self.is_jumping = True


def move_left(self): self.speed_x = -5  # Move left
def move_right(self): self.speed_x = 5  # Move right
def stop(self): self.speed_x = 0  # Stop moving
```

- update(): Runs every frame to move the bunny and apply gravity.
- jump(): Makes the bunny leap upward (if not already jumping).
- move_left() **&** move_right(): Sets horizontal speed.
- stop(): Stops movement when keys are released.

Shooting (Spacebar)

```
def shoot(self):
    bullet = Projectile(self.rect.centerx, self.rect.centery)  # Create bullet at bunny's center
    all_sprites.add(bullet)  # Add to all_sprites so it's drawn
    projectiles.add(bullet)  # Add to bullet group for collision checks
    shoot_sound.play()  # Play "pew!" sound
```

- Projectile(): Creates a new bullet.
- all_sprites.add(): Makes the bullet appear on screen.
- projectiles.add(): Tracks it for hitting enemies.
- shoot_sound.play(): Plays the shooting sound.

8. Projectile Class (Bullets)

```
class Projectile(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.image = pygame.Surface((10, 5))  # Small red rectangle
```

```
        self.image.fill(RED)
        self.rect = self.image.get_rect(center=(x, y))  # Position at bunny's center
        self.speed_x = 10  # Moves right fast

    def update(self):
        self.rect.x += self.speed_x  # Move bullet
        if self.rect.left > SCREEN_WIDTH:  # Delete if off-screen
            self.kill()
```

- self.image: A tiny red rectangle (the bullet).
- self.speed_x = 10: Moves right quickly.
- update(): Moves the bullet and removes it if it leaves the screen.

9. Enemy Classes (Dragons)

Basic Enemy (Walks Left)

```
class Enemy(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.image = dragon_image  # Uses loaded dragon sprite
        self.rect = self.image.get_rect(x=x, y=y)  # Position
        self.health = 10  # Takes 10 hits to kill
        self.speed_x = 2  # Moves left slowly

    def update(self):
        self.rect.x -= self.speed_x  # Move left
        if self.health <= 0 or self.rect.right < 0:  # Remove if dead or off-screen
            self.kill()
```

- self.health = 10: Dies after 10 bullet hits.
- self.speed_x = 2: Moves left at speed 2.

Flying Enemy (Sine Wave Movement + Fireballs)

```
class FlyingEnemy(pygame.sprite.Sprite):
    def __init__(self, x):
        super().__init__()
        self.image = pygame.transform.scale(dragon_image, (60, 40))  # Slightly bigger
        self.rect = self.image.get_rect()
        self.rect.x = x  # Starts at right side
        self.base_y = SCREEN_HEIGHT - 80  # Flying height
        self.rect.y = self.base_y
        self.speed_x = random.randint(2, 4)  # Random speed
        self.amplitude = 10  # How high/low it flies
        self.frequency = 0.12  # How fast it waves
        self.tick = 0  # Timer for movement
        self.health = 30  # Tougher than basic enemy
        self.fireball_timer = random.randint(60, 120)  # Random fireball delay

    def update(self):
        self.rect.x -= self.speed_x  # Move left
        self.rect.y = self.base_y + int(self.amplitude * math.sin(self.frequency * self.tick))  #
Sine wave flight
        self.tick += 1  # Increase timer
        self.fireball_timer -= 1  # Countdown to next fireball
        if self.fireball_timer <= 0:  # Shoot fireball when timer hits 0
            fireball = Fireball(self.rect.centerx, self.rect.centery)
            fireballs.add(fireball)
            all_sprites.add(fireball)
            self.fireball_timer = random.randint(90, 150)  # Reset timer
        if self.health <= 0 or self.rect.right < 0:  # Remove if dead or off-screen
            self.kill()
```

- math.sin(): Makes the dragon fly in a wave pattern.

- self.fireball_timer: Random delay between fireball shots.
- Fireball(): Shoots a fireball at the player.

Boss Enemy (Big & Strong)

```
class BossEnemy(pygame.sprite.Sprite):
    def __init__(self, health=30):
        super().__init__()
        self.image = pygame.transform.scale(dragon_image, (120, 120))  # HUGE dragon
        self.rect = self.image.get_rect()
        self.rect.x = SCREEN_WIDTH  # Starts at right edge
        self.rect.y = SCREEN_HEIGHT - 150  # Slightly above ground
        self.health = health  # Default 30 health
        self.speed_x = random.randint(1, 3)  # Moves slowly

    def update(self):
        self.rect.x -= self.speed_x  # Move left
        if self.health <= 0 or self.rect.right < 0:  # Remove if dead or off-screen
            self.kill()
```

- 120x120 **size**: Much bigger than other dragons.
- health=30: Takes **30 hits** to kill (unless modified).

10. Fireball Class (Enemy Attacks)

```
class Fireball(pygame.sprite.Sprite):
    def __init__(self, x, y):
        super().__init__()
        self.image = pygame.Surface((10, 10))  # Small orange square
        self.image.fill((255, 165, 0))  # Orange color
        self.rect = self.image.get_rect(center=(x, y))  # Position at dragon's center
        self.speed_x = -6  # Moves left toward player

    def update(self):
```

```
        self.rect.x += self.speed_x  # Move left
        if self.rect.right < 0:  # Remove if off-screen
            self.kill()
```

- self.speed_x = -6: Moves left toward the player.
- (255, 165, 0): RGB color for orange.

11. Drawing UI (Health, Score, Lives, Level)

```
def draw_health_bar():
    pygame.draw.rect(screen, RED, (10, 10, 200, 20))  # Red background
    pygame.draw.rect(screen, GREEN, (10, 10, 200 * (player.health / 100), 20))  # Green fill (shrinks when hurt)

def draw_score():
    screen.blit(pygame.font.Font(None, 36).render(f"Score: {player.score}", True, WHITE), (SCREEN_WIDTH - 150, 10))

def draw_lives():
    screen.blit(pygame.font.Font(None, 36).render(f"Lives: {player.lives}", True, WHITE), (SCREEN_WIDTH - 150, 50))

def draw_level():
    screen.blit(pygame.font.Font(None, 36).render(f"Level: {level}", True, WHITE), (10, 40))
```

- draw_health_bar(): Shows a red/green bar at the top-left.
- draw_score(): Displays the player's score (top-right).
- draw_lives(): Shows remaining lives (top-right, below score).
- draw_level(): Shows current level (top-left, below health).

12. Game State Functions (Winning/Losing)

```
def game_over():
    screen.fill(BLACK)
    text = pygame.font.Font(None, 64).render("GAME OVER", True, RED)
    screen.blit(text, (SCREEN_WIDTH // 2 - 160, SCREEN_HEIGHT // 2 - 50))
    pygame.display.flip()
```

```
    pygame.time.wait(3000)  # Show for 3 seconds
    show_restart_menu()  # Ask to restart or quit
```

- Displays **"GAME OVER"** in red for 3 seconds.
- Then shows the restart menu.

Restart Menu

```
def show_restart_menu():
    font = pygame.font.Font(None, 48)
    while True:
        screen.fill(BLACK)
        text = font.render("Press R to Restart or Q to Quit", True, WHITE)
        screen.blit(text, (SCREEN_WIDTH // 2 - 250, SCREEN_HEIGHT // 2 - 30))
        pygame.display.flip()
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit(); exit()
            elif event.type == pygame.KEYDOWN:
                if event.key == pygame.K_r:  # Restart
                    main()  # Reset the game
                    return
                elif event.key == pygame.K_q:  # Quit
                    pygame.quit(); exit()
```

- R: Restarts the game by calling main() again.
- Q: Closes the game.

13. Level Management

Starting a Level

- lvl == 1: Spawns **6 basic dragons**.
- lvl == 2: Spawns **1 boss + 8 small dragons**.
- lvl == 3: Starts a **60-second timer** and spawns waves of enemies.

14. Main Game Loop (main()**)**

This is where the **entire game runs**:

1. **Handles player input** (movement, shooting).
2. **Updates all sprites** (movement, collisions).
3. **Checks win/lose conditions**.
4. **Draws everything on screen**.

Key Parts of the Loop

```
while running:
    clock.tick(FPS)  # Runs at 60 FPS
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False  # Close game
        elif event.type == pygame.KEYDOWN:
            if event.key == pygame.K_LEFT: player.move_left()
            elif event.key == pygame.K_RIGHT: player.move_right()
            elif event.key == pygame.K_UP: player.jump()
            elif event.key == pygame.K_SPACE: player.shoot()
        elif event.type == pygame.KEYUP:
            if event.key in [pygame.K_LEFT, pygame.K_RIGHT]: player.stop()
        elif event.type == pygame.USEREVENT + 1 and level == 3:  # Level 3 enemy waves
            for i in range(7):
                f = FlyingEnemy(SCREEN_WIDTH + i * 90)
                enemies.add(f)
                all_sprites.add(f)
            for i in range(7):
                b = BossEnemy(health=30)
                b.rect.x = SCREEN_WIDTH + i * 120
                enemies.add(b)
                all_sprites.add(b)
            pygame.time.set_timer(pygame.USEREVENT + 1, 0)  # Stop spawning
```

- clock.tick(FPS): Keeps the game running smoothly at 60 FPS.
- event.type == pygame.KEYDOWN: Detects key presses (movement, shooting).
- event.type ==pygame.USEREVENT +1: Triggers enemy waves in **Level 3**.

15. Collision Detection (Bullets, Fireballs, Enemies)

```
# Check bullet hits on enemies
for bullet in projectiles:
    for enemy in pygame.sprite.spritecollide(bullet, enemies, False):
```

```
        enemy.health -= 10  # Reduce enemy health
        bullet.kill()  # Remove bullet
        player.score += 10  # Add 10 points

# Check fireball hits on player
for fb in fireballs:
    if player.rect.colliderect(fb.rect):
        fb.kill()  # Remove fireball
        player.health -= 10  # Hurt player
        if player.health <= 0:  # If health reaches 0
            player.lives -= 1  # Lose a life
            player.health = 100  # Reset health
            if player.lives <= 0:  # If no lives left
                game_over()  # Game Over screen

# Check enemy collisions with player
for enemy in enemies:
    if player.rect.colliderect(enemy.rect):
        player.health -= 20  # Big damage
        enemy.kill()  # Remove enemy
        if player.health <= 0:
            player.lives -= 1
            player.health = 100
            if player.lives <= 0:
                game_over()
```

- spritecollide(): Checks if bullets hit enemies.
- colliderect(): Checks if fireballs or enemies touch the player.

16. Level Completion & Winning

```
if len(enemies) == 0:  # If all enemies are dead
    if level != 3 or pygame.time.get_ticks() <= level_timer:  # Check if time remains (Level 3)
        screen.fill(BLACK)
        screen.blit(pygame.font.Font(None, 64).render("Level Complete!", True, GREEN),
(SCREEN_WIDTH//2 - 180, SCREEN_HEIGHT//2 - 50))
        pygame.display.flip()
        pygame.time.wait(2000)  # Show for 2 seconds
        level += 1  # Go to next level
        if level > 3:  # If all levels beaten
```

```
            screen.fill(BLACK)
            screen.blit(pygame.font.Font(None, 64).render("You Win! Congratulations!", True,
GREEN), (SCREEN_WIDTH // 2 - 300, SCREEN_HEIGHT // 2 - 50))
            pygame.display.flip()
            pygame.time.wait(3000)  # Show for 3 seconds
            break  # End game
        else:  # Otherwise, start next level
            player.health = 100  # Heal player
            start_level(level)  # Load next level
```

- len(enemies) == 0: If no enemies left, level is complete.
- level += 1: Advances to the next level.
- level > 3: If all 3 levels are beaten, show **"You Win!"**.

17. Final Steps

```
if __name__ == "__main__":
    main()  # Starts the game
```

🎮 How the Game Works (Big Picture)

1. **You're the Bunny Hero** - Control a rabbit character at the bottom left
2. **Dragons Attack** - They come from the right side flying or walking
3. **Shoot Them!** - Press SPACE to fire your weapon
4. **Don't Get Hit!** - If dragons or their fireballs touch you, you lose health
5. **Three Levels** - Each gets harder with more/bigger dragons
6. **Win/Lose** - Beat all dragons to win, lose all 3 lives and you're toast!

🕹️ Controls (How to Play)

- ⬅️ **Left Arrow** = Move bunny left
- ➡️ **Right Arrow** = Move bunny right
- ⬆️ **Up Arrow** = Make bunny jump (careful, gravity pulls you down!)
- **SPACE** = Shoot bullets at dragons

🐲 Meet the Bad Guys

1. **Basic Dragons** - Walk left towards you (10 health)
2. **Flying Dragons** - Fly in wave patterns and shoot fireballs (30 health)
3. **Boss Dragons** - Big scary ones that take lots of hits (30+ health)

💖 Health System

- Start with 100 health (green bar at top)
- Each hit removes 10-20 health
- When health reaches 0, lose 1 life (start with 3 lives)
- Health refills when you lose a life or beat a level

🏆 Scoring & Levels

- **+10 points** per dragon hit
- **3 Levels** total:
    - Easy - Just walking dragons
    - Medium - Boss dragon + helpers
    - Hard - Flying dragons + time limit (1 minute to win!)

Output:

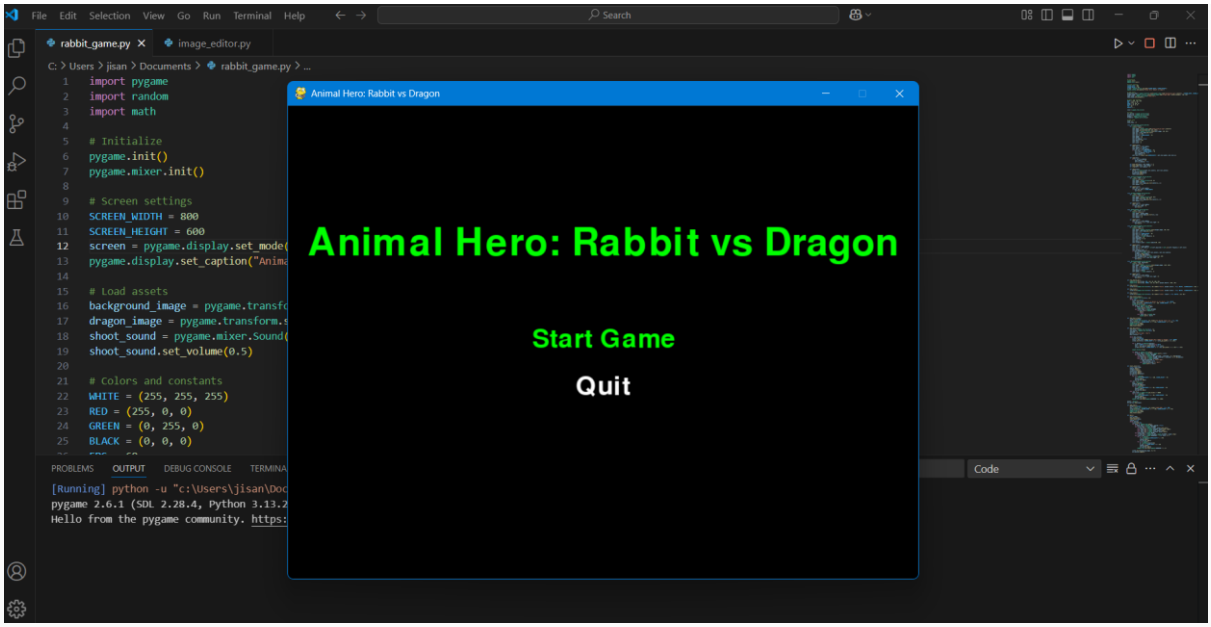

Figure 01: How it start

Figure 02: This is Level 1

Figure 03: Firing hero

Figure 04: Level 2 with score and health bar with lives

Figure 05: Jumping hero

Figure 06: Moving hero

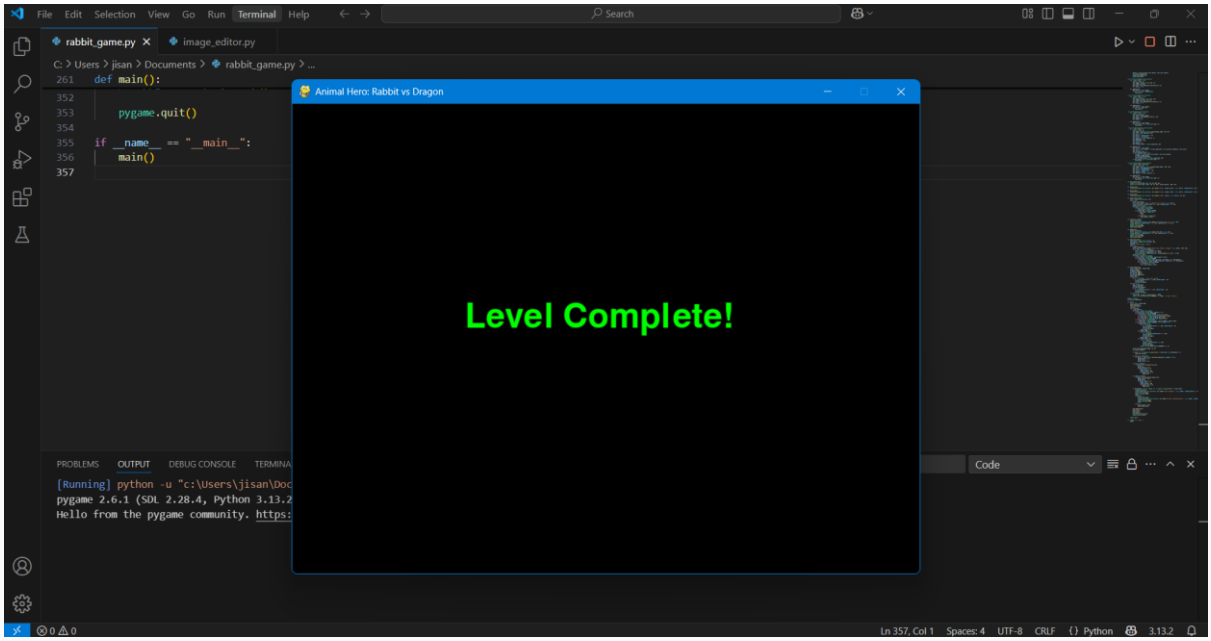

Figure 07: Level completion

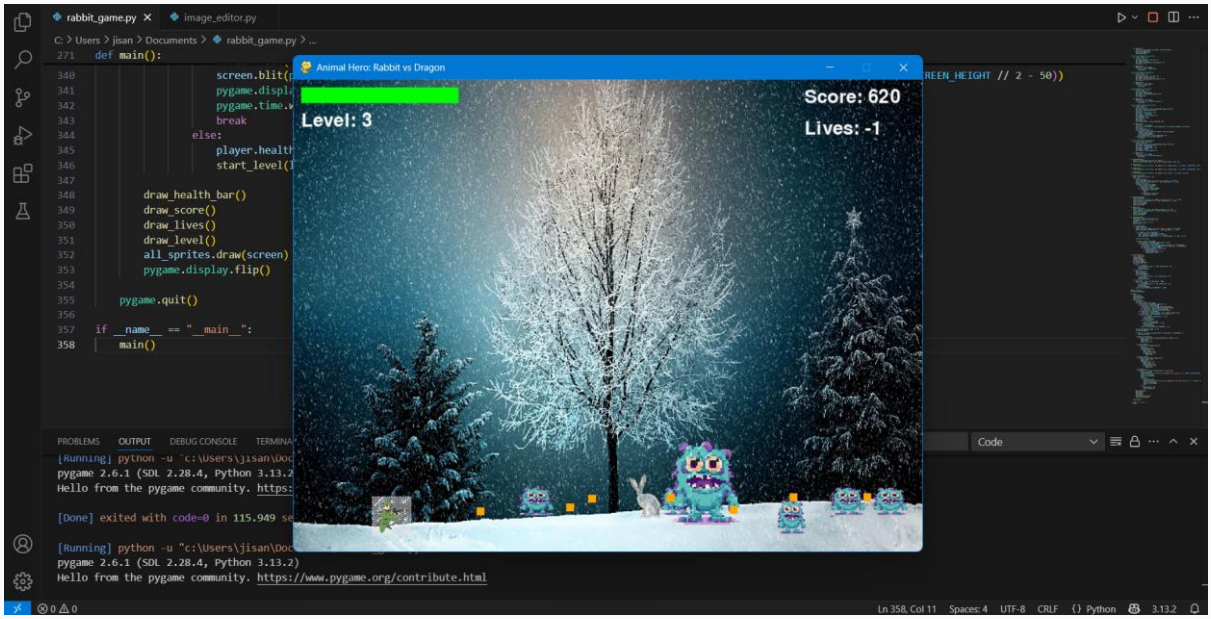

Figure 08: Level 3 (Running Flying and Big Enemy)

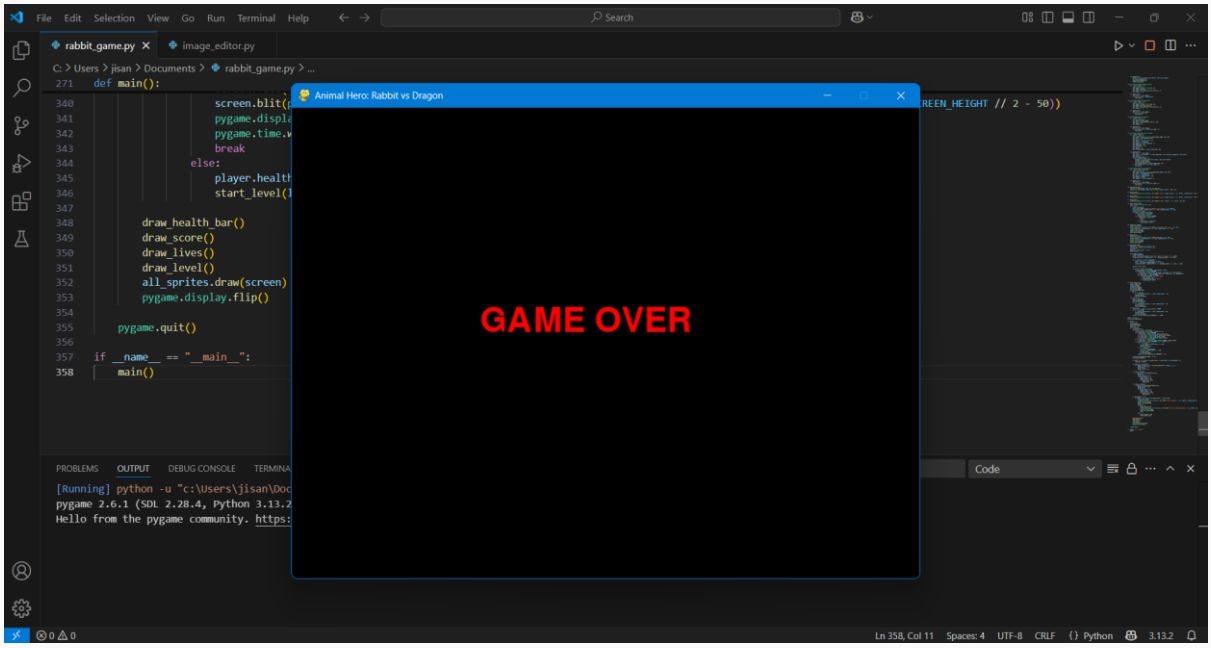

Figure 09: After losing 3 lives

Figure 10: After completing all the levels