BeeVenture

Adventure of an Ordinary Bee

Ei Myat Nwe 66011534

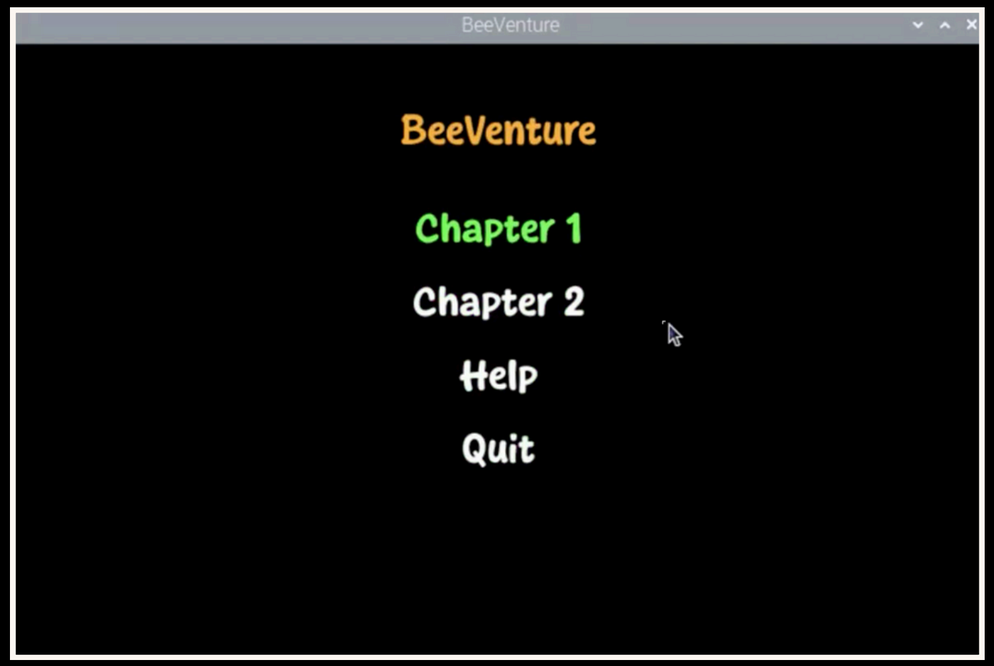
Thura Aung 66011606

Introduction

It is a Arcade inspired game in which our character “Bee” is trying to

- Collect its own honey spots 
- Avoiding the barriers 
- Avoiding and Shooting its enemies (hives) 
- When lives decrease, Eat Honey Pots 
- And kill the boss 

Main Page



4 Menu Options

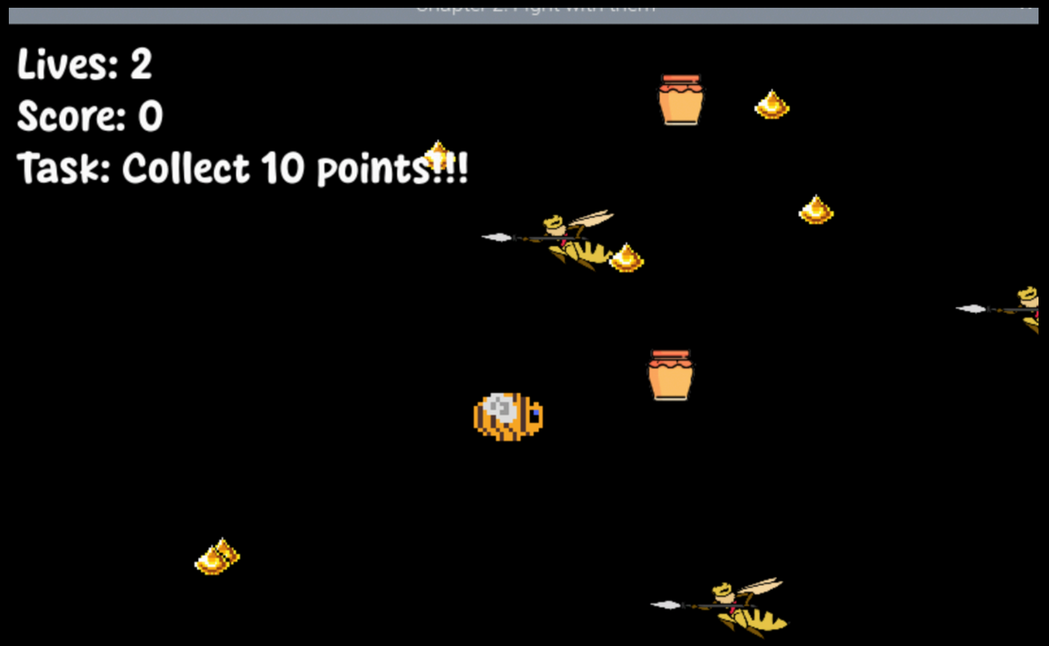
- Chapter 1
- Chapter 2
- Help
- Quit

Chapter 1

We include

- Task
- Lives
- Barriers we passed
- Inspired from Classic Flappy Bird

Chapter 2



Before Boss Spawn

- Collect 10 Points (Honey spots)
- Avoid or Shoot the enemies (Hives)
- If collide with enemies, lives will decrease
- If Honey Pot is spawned (collide with it), lives increase up to 3 (Maximum)
- If lives is zero, game over

Chapter 2

After Boss Spawn

- Avoid Boss' Bullets
- Kill the Boss
- There is no Honey Spot generation
- Only Honey Pot generation for health

How to Play

Chapter 1

Way to the enemy land

Avoid the tiles

Movement with our controller

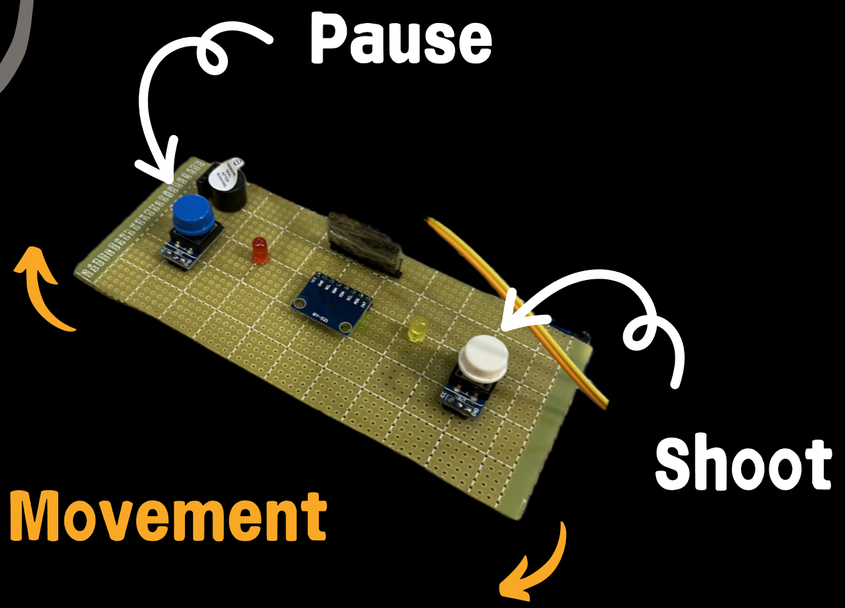
Chapter 2

Avoid or kill the enemy troppers

Movement with our controller

Collect the Honey Drop (Scores)

Collect the Honey Pot (Health)

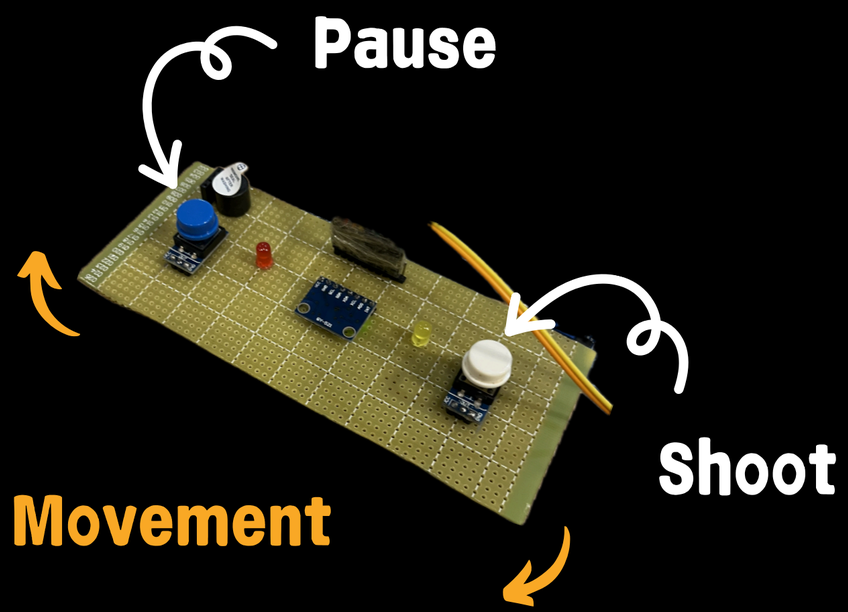


Help Page to Explain How to Play

Learning Outcomes

- Create a game with a custom game controller
- Familiar with RaspberryPi, Sensors, Low-level concepts and Pygame
- Teamwork and Task Sharing

How Game Works



Challenges

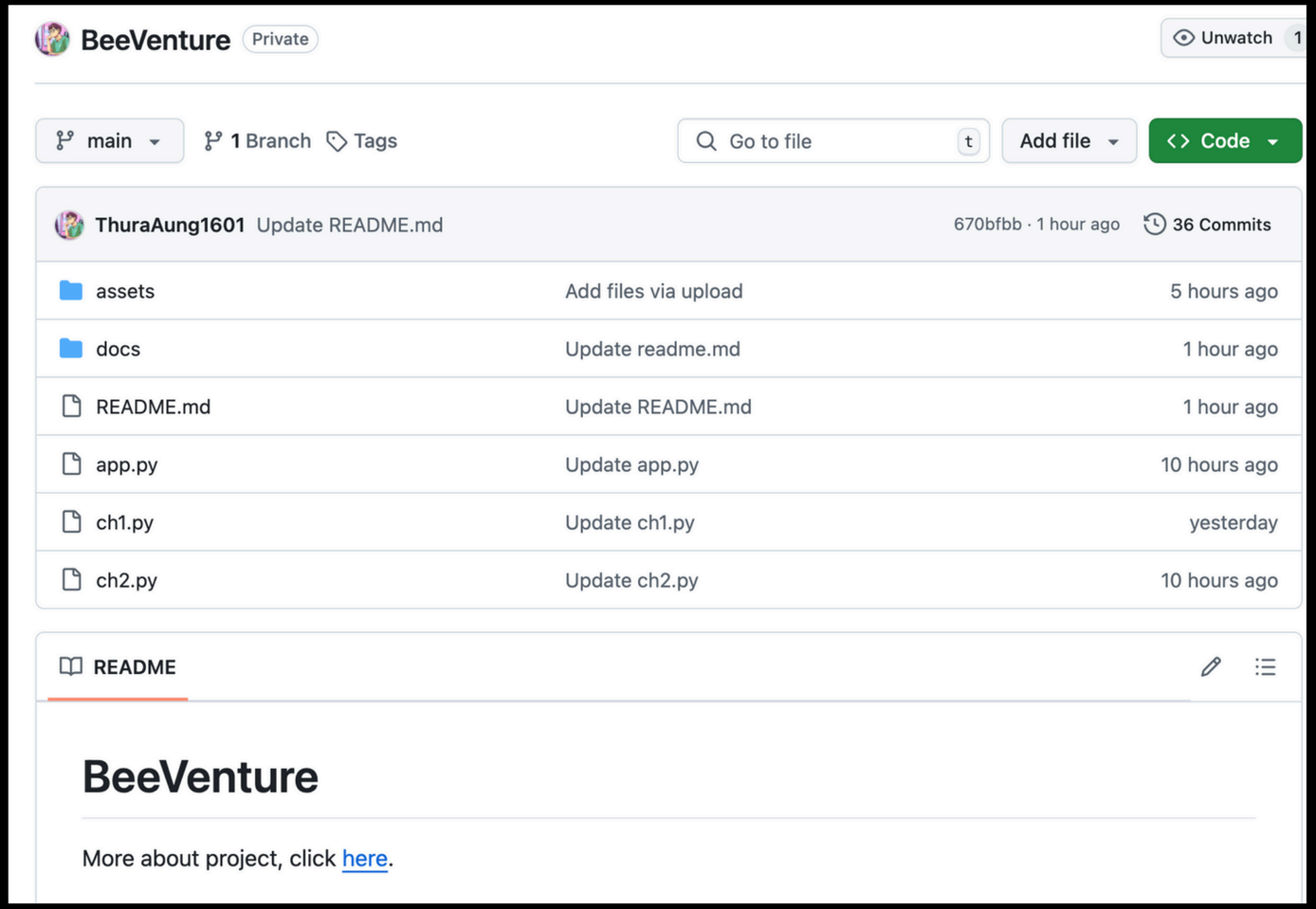
Main Challenge:

Handling Real-time Button Inputs

Solution:

- Multithreading: To ensure that the button press detection doesn’t block other game operations, a separate thread was created to monitor the shooting button. This allowed the button press detection to run concurrently with the main game loop, ensuring immediate responses to player actions.
- Debouncing: A common issue when using mechanical buttons is "bouncing," where a single press might register multiple inputs due to the button's physical characteristics. Debouncing was implemented in software to ignore repeated button presses that occur within a short period after the initial press, ensuring a single press is detected.

Github


BeeVenture
Private
Unwatch 1
main
1 Branch
Tags
Go to file t
Add file
Code
ThuraAung1601 Update README.md
670bfbb · 1 hour ago
36 Commits
assets Add files via upload 5 hours ago
docs Update readme.md 1 hour ago
README.md Update README.md 1 hour ago
app.py Update app.py 10 hours ago
ch1.py Update ch1.py yesterday
ch2.py Update ch2.py 10 hours ago
README
BeeVenture
More about project, click here.

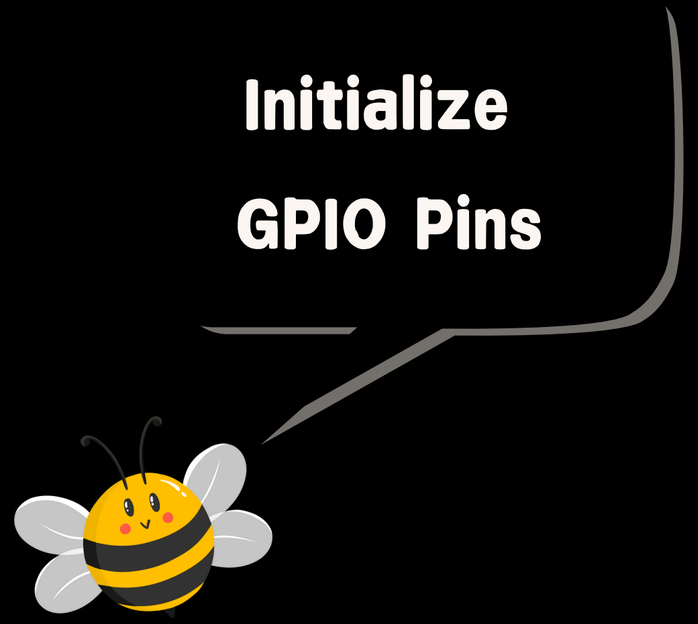
Code Analysis

```
# Initialize MPU6050
sensor = mpu6050(0x68)

# GPIO Pin configuration for buttons
BUTTON_1_PIN = 17  # Pause
BUTTON_2_PIN = 27  # Shoot
BUZZER_PIN = 24     # Buzzer
LED_PIN1 = 22      # LED 1 (for lives decrease)
LED_PIN2 = 23      # LED 2 (for shooting feedback)

# Setup GPIO
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(BUTTON_1_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(BUTTON_2_PIN, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)
GPIO.setup(BUZZER_PIN, GPIO.OUT)
GPIO.setup(LED_PIN1, GPIO.OUT)
GPIO.setup(LED_PIN2, GPIO.OUT)

# Ensure the buzzer is off initially
GPIO.output(BUZZER_PIN, GPIO.LOW)  # Start with the buzzer off (active-low configuration)
GPIO.output(LED_PIN1, GPIO.LOW)
GPIO.output(LED_PIN2, GPIO.LOW)
```



Code Analysis

```python
# Check for collisions
player_rect = pygame.Rect(rect_x - player_width // 2, rect_y - player_height // 2, player_width, player_height)

for obstacle in obstacles:
    top_rect = pygame.Rect(obstacle[0], 0, obstacle_width, obstacle[1])
    bottom_rect = pygame.Rect(obstacle[0], obstacle[1] + obstacle_gap, obstacle_width, SCREEN_HEIGHT - (obstacle[1] + obstacle_gap))

    if player_rect.colliderect(top_rect) or player_rect.colliderect(bottom_rect):
        lives -= 1

        # Buzzer and LED feedback for life loss
        GPIO.output(BUZZER_PIN, GPIO.HIGH)  # Turn on buzzer
        GPIO.output(LED_PIN1, GPIO.HIGH)    # Turn on LED
        time.sleep(0.1)                     # Short delay for feedback
        GPIO.output(BUZZER_PIN, GPIO.LOW)   # Turn off buzzer
        GPIO.output(LED_PIN1, GPIO.LOW)     # Turn off LED

        obstacles.remove(obstacle)  # Remove the collided obstacle
        if lives <= 0:
            running = False
```

Collision Detection

```python
# Move enemy bullets and check for collision with player
for e_bullet in enemy_bullets[:]:
    e_bullet[0] -= 10
    e_bullet_rect = pygame.Rect(e_bullet[0], e_bullet[1], 5, 5)
    if e_bullet_rect.colliderect(player_rect):
        lives -= 1
        enemy_bullets.remove(e_bullet)

# Collision with health items
for health in health_items[:]:
    health_rect = pygame.Rect(health[0], health[1], health_img.get_width(), health_img.get_height())
    if player_rect.colliderect(health_rect):
        if lives < 3:  # Only increase lives if below maximum
            lives += 1
        health_items.remove(health)

# Collision with food items
for food in food_items[:]:
    food_rect = pygame.Rect(food[0], food[1], food_img.get_width(), food_img.get_height())
    if player_rect.colliderect(food_rect):
        score += 1
        food_items.remove(food)

# Check for bullet collision with the boss
if boss_spawned:
    boss_rect = pygame.Rect(boss_x, boss_y, enemy_boss_img.get_width(), enemy_boss_img.get_height())
    for bullet in bullets[:]:
        bullet_rect = pygame.Rect(bullet[0], bullet[1], 5, 5)
        if bullet_rect.colliderect(boss_rect):
            boss_health -= 1
            bullets.remove(bullet)
            if boss_health <= 0:
                running = False
```

Code Analysis

Debounce and Multithreading

```python
# Debounce settings
debounce_time = 0.1
button1_last_state = GPIO.LOW
button2_last_state = GPIO.LOW

# Function to shoot bullets in a separate thread
def shoot_bullet():
    global bullets
    bullet_x = rect_x + player_img.get_width()
    bullet_y = rect_y + player_img.get_height() // 2
    bullets.append([bullet_x, bullet_y])
    GPIO.output(BUZZER_PIN, GPIO.HIGH)  # Activate buzzer (sound on)
    GPIO.output(LED_PIN2, GPIO.HIGH)    # LED 2 on
    time.sleep(0.1)                      # Duration of the beep (100 ms)
    GPIO.output(BUZZER_PIN, GPIO.LOW)   # Deactivate buzzer (silent)
    GPIO.output(LED_PIN2, GPIO.LOW)     # LED 2 off

# Function to handle shooting in a thread
def handle_shooting():
    while True:
        button2_state = GPIO.input(BUTTON_2_PIN)
        if button2_state == GPIO.HIGH:
            shoot_bullet()
            time.sleep(debounce_time)  # Debounce delay
        time.sleep(0.01)  # Short sleep to reduce CPU usage

# Start the shooting thread
shooting_thread = threading.Thread(target=handle_shooting, daemon=True)
shooting_thread.start()
```

Team Cooperation

Ei Myat Nwe

66011534

Game Logic

Button Buzzer LED

Chapter 1

Thura Aung

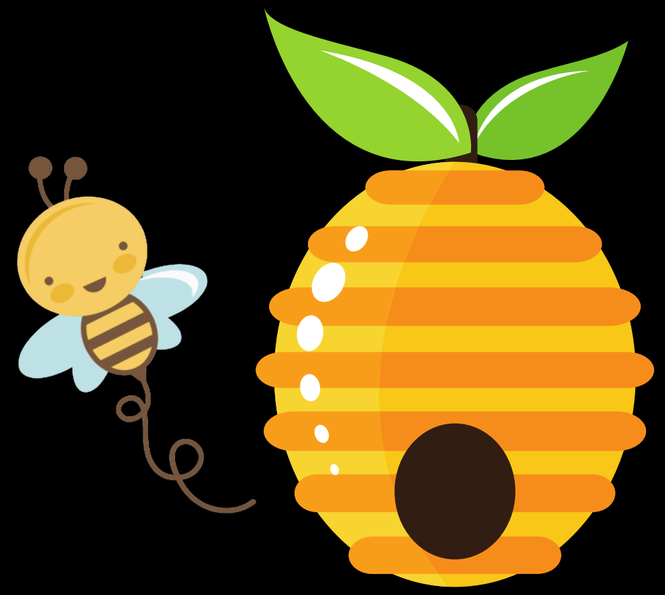
66011606

Game Logic

Gyro Setup

Chapter 2

Q & A?



Thank you for your time!!