Programming Fundamentals (SE-101)

Project Report Fruit Frenzy Game

Developed by:

1) Muhammad Bilal Nasir (SE-24069)
2) Ahad Sharif (SE-24073)
3) Kashif Ali (SE-24084)
4) Afnan Rauf (SE-24102)

Presented to:

Engr. Asma Khan

FRUIT FRENZY

Project Report

Introduction:-

Fruit Frenzy is a captivating arcade game where players catch falling fruits in a basket while dodging bombs. Designed to test reflexes and improve hand-eye coordination, the game offers adjustable difficulty levels for players of all skill levels. It is built using Python, utilizing the Pygame and Tkinter libraries for a smooth, interactive experience.

Objectives:-

- Deliver an engaging gaming experience for casual players and kids.

- Enhance reflexive and cognitive skills through interactive gameplay.

- Provide flexible difficulty levels to cater to different player skill sets.

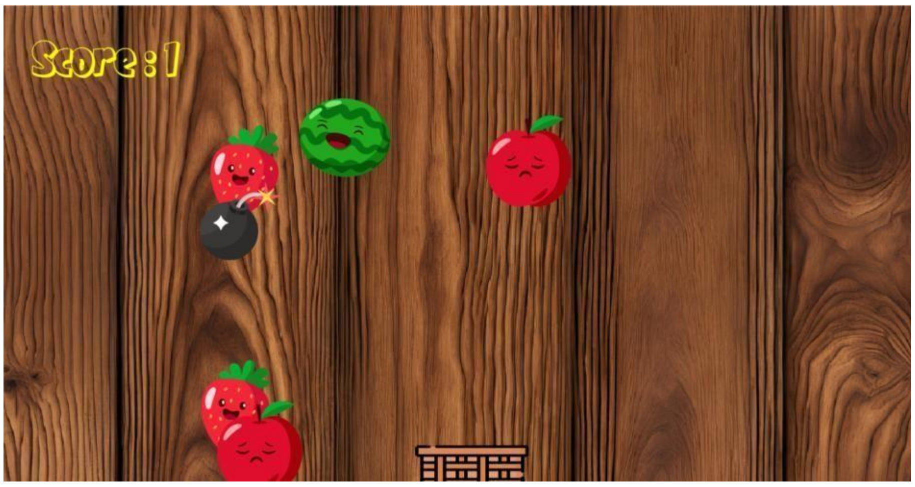
Features:-

1. Dynamic Gameplay:-

- Catch fruits to score points while avoiding bombs that end the game.

- Difficulty levels (Easy, Medium, Hard) adjust fruit and bomb speeds and spawn rates.

GAME VIEW:-



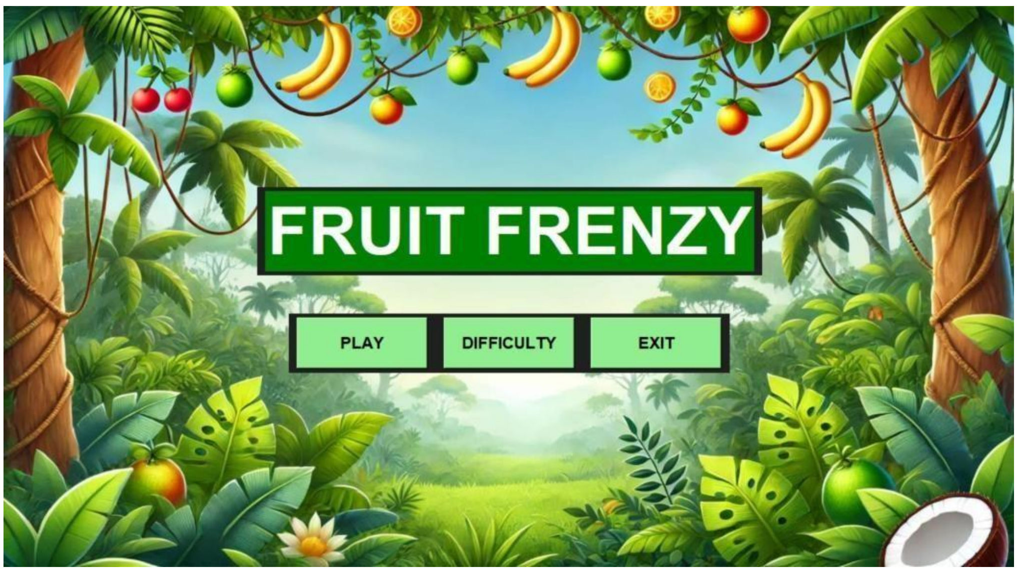
2. Interactive Start Page:-

- **Play Button:** Launches the game with the selected difficulty.

- **Difficulty Button:** Opens a window to select difficulty via radio buttons.

- **Exit Button:** Closes the game.

-



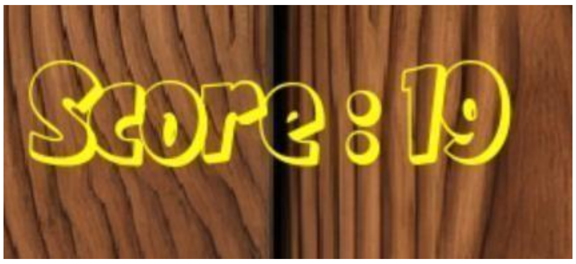
3. Sound Effects and Music:-

- Background music for immersion.

- Distinct sound effects for catching fruits and bomb collisions.

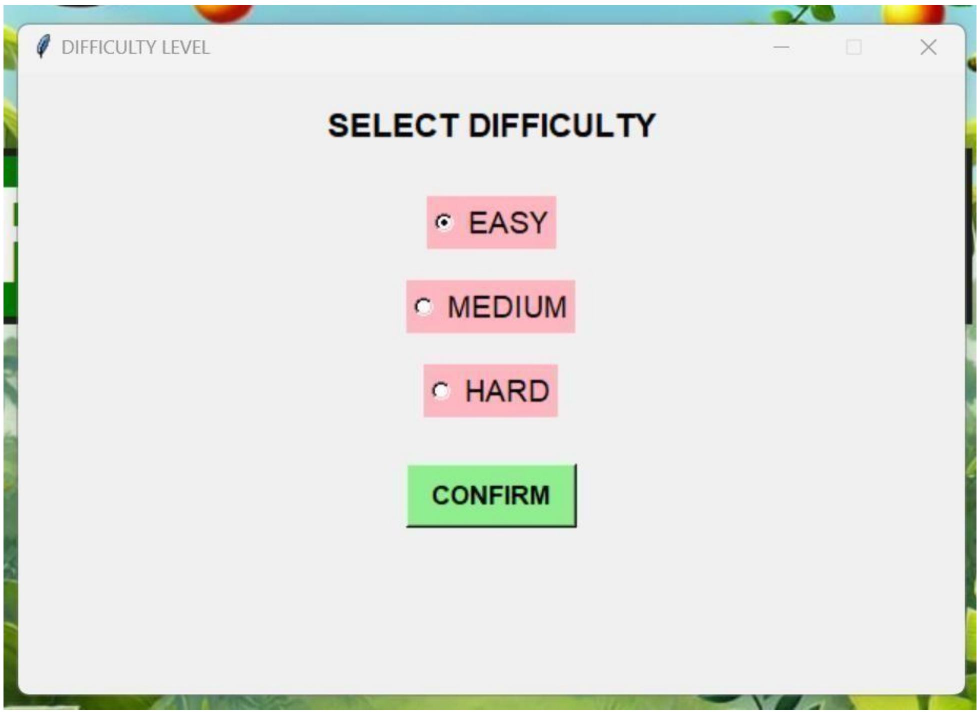
4. Score Tracking:-

- Real-time score updates displayed prominently during gameplay.



5. Customizable Experience:-

- Players can tailor the game to their skill level through difficulty settings.

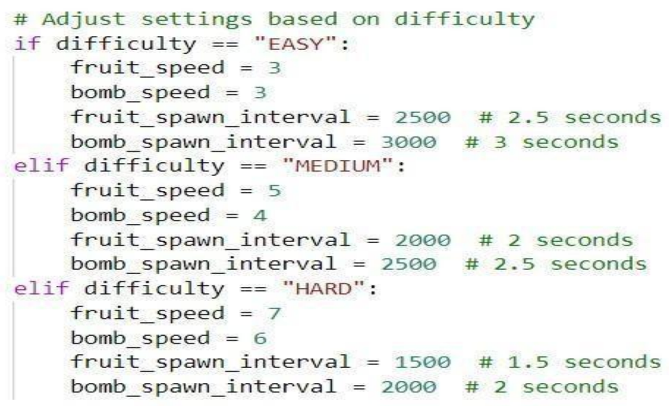


SCOPE:-

Fruit Frenzy is a versatile arcade game designed for players of all ages, offering adjustable difficulty levels to cater to both beginners and advanced gamers. It improves reflexes and decisionmaking skills through fast-paced gameplay. The project demonstrates the use of Python libraries like Pygame and Tkinter for game development, making it an excellent learning tool for aspiring developers. With a modular and scalable design, it supports easy future enhancements such as multiplayer modes, power-ups, leaderboards, and mobile adaptation. Its cross-platform compatibility ensures accessibility on Windows, macOS, and Linux.

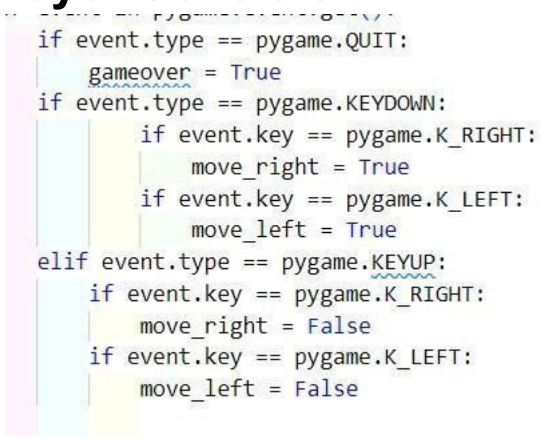
Key Features and Code Snippets:

1. Dynamic Difficulty Adjustment:-



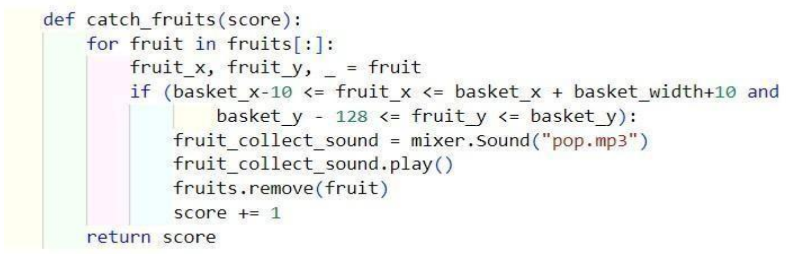
```
# Adjust settings based on difficulty
if difficulty == "EASY":
    fruit_speed = 3
    bomb_speed = 3
    fruit_spawn_interval = 2500  # 2.5 seconds
    bomb_spawn_interval = 3000  # 3 seconds
elif difficulty == "MEDIUM":
    fruit_speed = 5
    bomb_speed = 4
    fruit_spawn_interval = 2000  # 2 seconds
    bomb_spawn_interval = 2500  # 2.5 seconds
elif difficulty == "HARD":
    fruit_speed = 7
    bomb_speed = 6
    fruit_spawn_interval = 1500  # 1.5 seconds
    bomb_spawn_interval = 2000  # 2 seconds
```

2. **Player Controls:-**

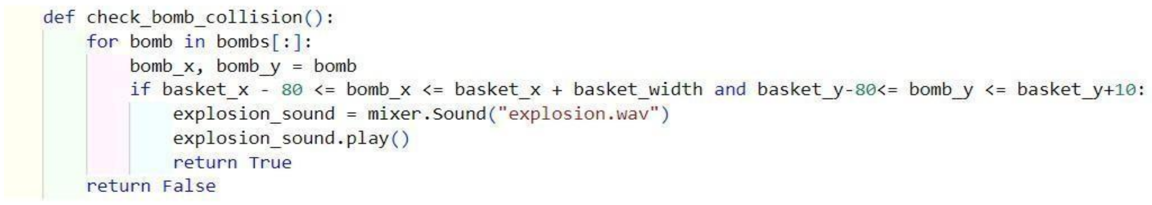


```
if event.type == pygame.QUIT:
    gameover = True
if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_RIGHT:
            move_right = True
        if event.key == pygame.K_LEFT:
            move_left = True
elif event.type == pygame.KEYUP:
    if event.key == pygame.K_RIGHT:
        move_right = False
    if event.key == pygame.K_LEFT:
        move_left = False
```

3. **Collision Detection:-**



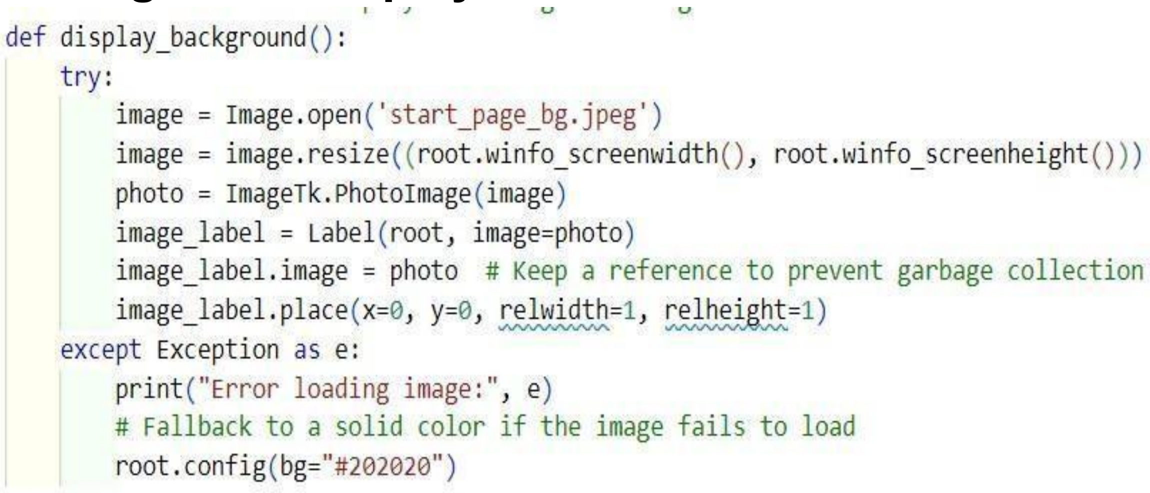
```
def catch_fruits(score):
    for fruit in fruits[:]:
        fruit_x, fruit_y, _ = fruit
        if (basket_x-10 <= fruit_x <= basket_x + basket_width+10 and
                basket_y - 128 <= fruit_y <= basket_y):
            fruit_collect_sound = mixer.Sound("pop.mp3")
            fruit_collect_sound.play()
            fruits.remove(fruit)
            score += 1
    return score
```



```
def check_bomb_collision():
    for bomb in bombs[:]:
        bomb_x, bomb_y = bomb
        if basket_x - 80 <= bomb_x <= basket_x + basket_width and basket_y-80<= bomb_y <= basket_y+10:
            explosion_sound = mixer.Sound("explosion.wav")
            explosion_sound.play()
            return True
    return False
```

▪

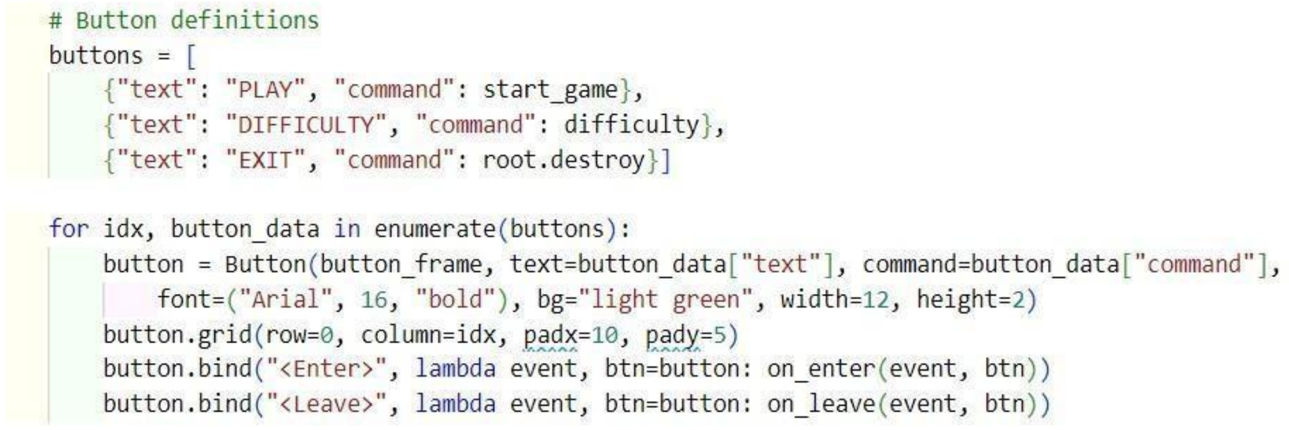
4. **Background Display:-**



```
def display_background():
    try:
        image = Image.open('start_page_bg.jpeg')
        image = image.resize((root.winfo_screenwidth(), root.winfo_screenheight()))
        photo = ImageTk.PhotoImage(image)
        image_label = Label(root, image=photo)
        image_label.image = photo  # Keep a reference to prevent garbage collection
        image_label.place(x=0, y=0, relwidth=1, relheight=1)
    except Exception as e:
        print("Error loading image:", e)
        # Fallback to a solid color if the image fails to load
        root.config(bg="#202020")
```

5. **Button Functions:-**

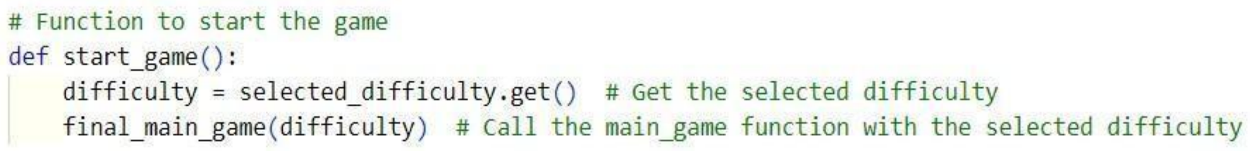
- Play
- Difficulty:-
- Exit Button:



```
# Button definitions
buttons = [
    {"text": "PLAY", "command": start_game},
    {"text": "DIFFICULTY", "command": difficulty},
    {"text": "EXIT", "command": root.destroy}]

for idx, button_data in enumerate(buttons):
    button = Button(button_frame, text=button_data["text"], command=button_data["command"],
        font=("Arial", 16, "bold"), bg="light green", width=12, height=2)
    button.grid(row=0, column=idx, padx=10, pady=5)
    button.bind("<Enter>", lambda event, btn=button: on_enter(event, btn))
    button.bind("<Leave>", lambda event, btn=button: on_leave(event, btn))
```

-



```
# Function to start the game
def start_game():
    difficulty = selected_difficulty.get()  # Get the selected difficulty
    final_main_game(difficulty)  # Call the main_game function with the selected difficulty
```

-

Future Enhancements

- **Multiplayer Mode:** Enable competitive or cooperative gameplay for two players.
- **Power-Ups:** Add shields, score multipliers, or speed boosters.
- **Leaderboard:** Implement local or global score tracking.
- **Themed Levels**: Introduce varying backgrounds and game mechanics.
- **Mobile Version**: Adapt the game for touch-screen devices.

Requirements

- **Hardware:** A computer with basic processing capability and a keyboard.

- **Software:** Python 3.x with libraries:

- Pygame - Tkinter - PIL

SOURCE CODE

START PAGE CODE:-

```
from tkinter import * from final_main_game import final_main_game  # Ensure this is
correctly implemented in main_game.py from PIL import Image, ImageTk

# Create the main application window root = Tk()
root.title("FRUIT-FRENZY") root.attributes('-fullscreen',
True)  # Enable fullscreen

# Global variable to store selected difficulty selected_difficulty
= StringVar(value="EASY")  # Default difficulty
```

```
# Function to exit fullscreen
def exit_fullscreen(event=None):
    root.attributes('-fullscreen', False)
# Function to display the difficulty selection window def
difficulty():
    difficulty_window = Toplevel(root)
difficulty_window.title("DIFFICULTY LEVEL")
difficulty_window.geometry("600x400+330+200")
difficulty_window.resizable(False, False)

    Label(difficulty_window, text="SELECT DIFFICULTY", font=("Arial", 16,
"bold")).pack(pady=20)

    # Radio buttons for difficulty levels
difficulties = ["EASY", "MEDIUM", "HARD"]     for
level in difficulties:
        Radiobutton(difficulty_window, bg='light pink', text=level,
variable=selected_difficulty, value=level, font=("Arial", 14),
anchor=W).pack(pady=10, anchor=CENTER)

    # Confirm button to close the difficulty window
    Button(difficulty_window, text="CONFIRM", command=difficulty_window.destroy,
font=("Arial", 12, "bold"), bg="light green", padx=10, pady=5).pack(pady=20)
# Function to load and display the background image
def display_background():     try:
        image = Image.open('start_page_bg.jpeg')     image =
    image.resize((root.winfo_screenwidth(), root.winfo_screenheight()))
        photo = ImageTk.PhotoImage(image)        image_label = Label(root,
 image=photo)        image_label.image = photo  # Keep a reference to prevent
 garbage collection        image_label.place(x=0, y=0, relwidth=1, relheight=1)
 except Exception as e:
        print("Error loading image:", e)
        # Fallback to a solid color if the image fails to load
root.config(bg="#202020")

# Function to start the game def
start_game():
    difficulty   =   selected_difficulty.get()      #   Get   the   selected   difficulty
final_main_game(difficulty)  # Call the main_game function with the selected difficulty
# Function to create buttons def create_buttons():     # Frame for
buttons     button_frame = Frame(root, bg="#202020")  # Match fallback
background     button_frame.place(relx=0.5, rely=0.6, anchor=CENTER)  #
Center frame
    # Button hover effect
def on_enter(event, button):
        button.config(bg="orange", fg="white")
     def on_leave(event,
button):
        button.config(bg="light green", fg="black")

    # Button definitions
buttons = [
        {"text": "PLAY", "command": start_game},
        {"text": "DIFFICULTY", "command": difficulty},
        {"text": "EXIT", "command": root.destroy}]
     for idx, button_data in
enumerate(buttons):
        button = Button(button_frame, text=button_data["text"],
command=button_data["command"],             font=("Arial", 16, "bold"),
bg="light green", width=12, height=2)         button.grid(row=0, column=idx,
padx=10, pady=5)
        button.bind("<Enter>", lambda event, btn=button: on_enter(event, btn))
button.bind("<Leave>", lambda event, btn=button: on_leave(event, btn))
# Add a welcome label def
display_welcome_text():
    name_frame = Frame(root, bg="#202020") # frame to display game name
 name_frame.place(relx=0.5, rely=0.4, anchor=CENTER) name = Label(name_frame,
 text="FRUIT FRENZY", font=("Arial", 64, "bold"), bg="green",
fg='white')     name.pack(padx=10, pady=5,
anchor=CENTER)

# Initialize the GUI
display_background()
display_welcome_text() create_buttons()

# Bind Escape key to exit fullscreen root.bind("<Escape>",
exit_fullscreen)

# Start the Tkinter main loop
root.mainloop()
```

MAIN GAME CODE:-

```
def final_main_game(difficulty):
    import pygame
import random     import os
from pygame import mixer

    os.environ['SDL_VIDEO_WINDOW_POS'] = "0,0"
    pygame.init()

    # Screen dimensions     info =
pygame.display.Info()
screen_width = info.current_w
screen_height = info.current_h

    game_window = pygame.display.set_mode((screen_width, screen_height))
pygame.display.set_caption("FRUIT-FRENZY")

    # Background image
    background_image = pygame.image.load('game_bg.png')     background_image =
pygame.transform.scale(background_image, (screen_width, screen_height))

    # Background music
    mixer.music.load("bg_music.mp3") mixer.music.play(-1)

    # Colors yellow =
    (255, 255, 0)

    # Basket attributes
    basket_width = 160
    basket_length = 128

    # Adjust settings based on difficulty
if difficulty == "EASY":
        fruit_speed = 3         bomb_speed = 3
fruit_spawn_interval = 2500  # 2.5 seconds
bomb_spawn_interval = 3000  # 3 seconds     elif
difficulty == "MEDIUM":
        fruit_speed = 5         bomb_speed = 4
fruit_spawn_interval = 2000  # 2 seconds
bomb_spawn_interval = 2500  # 2.5 seconds
elif difficulty == "HARD":
        fruit_speed = 7         bomb_speed = 6
fruit_spawn_interval = 1500  # 1.5 seconds
bomb_spawn_interval = 2000  # 2 seconds

    # Game variables
fruits = []     bombs
= []     basket_x =
700
    basket_y = screen_height - 128
score = 0     gameover = False

    clock = pygame.time.Clock()
    # Load images      basket_img =
pygame.image.load('basket.png').convert_alpha()
    bomb_image = pygame.image.load('bomb.png').convert_alpha()
    basket_image = pygame.transform.scale(basket_img, (basket_width, basket_length))

    fruit_images =[        pygame.image.load(f'{fruit}.png')        for fruit in
 ['apple', 'orange', 'mango', 'watermelon','strawberry','banana']    ]
    def create_fruits(num_fruits):
    for _ in range(num_fruits):
           fruit_x = random.randint(5, screen_width - 128)
    fruit_y = random.randint(-100, -20)
    fruit_image = random.choice(fruit_images)
    fruits.append([fruit_x, fruit_y, fruit_image])

    def create_bomb():
        bomb_x = random.randint(25, screen_width - 25)
bomb_y = random.randint(-100, -20)
bombs.append([bomb_x, bomb_y])
     def catch_fruits(score):
for fruit in fruits[:]:
            fruit_x, fruit_y, _ = fruit             if (basket_x-10 <=
fruit_x <= basket_x + basket_width+10 and                    basket_y -
128 <= fruit_y <= basket_y):                 fruit_collect_sound =
mixer.Sound("pop.mp3")                 fruit_collect_sound.play()
fruits.remove(fruit)                  score += 1        return score
     def
check_bomb_collision():
for bomb in bombs[:]:
            bomb_x, bomb_y = bomb            if basket_x - 80 <= bomb_x <= basket_x +
basket_width and basket_y-80<= bomb_y
<= basket_y+10:
                explosion_sound = mixer.Sound("explosion.wav")
explosion_sound.play()                 return True
return False
    # Main game loop     last_fruit_spawn_time =
pygame.time.get_ticks()     last_bomb_spawn_time =
pygame.time.get_ticks()

move_right=False
move_left=False
            while not gameover:
 current_time = pygame.time.get_ticks()

        # Event handling        for
 event in pygame.event.get():
 if event.type == pygame.QUIT:
                gameover = True
           if event.type == pygame.KEYDOWN:
    if event.key == pygame.K_RIGHT:
                       move_right = True
    if event.key == pygame.K_LEFT:
                        move_left = True
    elif event.type == pygame.KEYUP:
                if event.key == pygame.K_RIGHT:
    move_right = False
                if event.key == pygame.K_LEFT:
                    move_left = False

if move_right:
            basket_x = min(screen_width - basket_width, basket_x + 20)
                 if
move_left:
            basket_x = max(0, basket_x - 20)

        # Spawn fruits         if current_time - last_fruit_spawn_time
>= fruit_spawn_interval:
            create_fruits(3) # spawn three fruits
last_fruit_spawn_time = current_time

        # Spawn bombs         if current_time - last_bomb_spawn_time
>= bomb_spawn_interval:
            create_bomb()
last_bomb_spawn_time = current_time

        # Update positions
for fruit in fruits:
            fruit[1] += fruit_speed
for bomb in bombs:
            bomb[1] += bomb_speed

        # Remove off-screen objects         fruits = [fruit for fruit in
fruits if fruit[1] <= screen_height]         bombs = [bomb for bomb in
bombs if bomb[1] <= screen_height]

        # Collision checks
score = catch_fruits(score)
        if check_bomb_collision():
            gameover = True
 break

        # Draw everything
 game_window.blit(background_image, (0, 0))
 for fruit in fruits:
            game_window.blit(fruit[2], (fruit[0], fruit[1]))
      for bomb in bombs:
          game_window.blit(bomb_image, (bomb[0], bomb[1]))
    game_window.blit(basket_image, (basket_x, basket_y))

        # Display score     font =
    pygame.font.Font("Komigo3D-Regular.ttf", 64)
        score_text = font.render(f"Score : {score}", True, yellow)
        game_window.blit(score_text, (40, 20))
pygame.display.update()         clock.tick(60)
    # Game Over screen
if gameover:
        font = pygame.font.Font("Komigo3D-Regular.ttf", 128)         text =
font.render("GAME OVER!", True, yellow)         game_window.blit(text,
(screen_width // 2 - 300, screen_height // 2 - 100))
pygame.mixer.music.stop()         pygame.display.update()
pygame.time.wait(3000)

pygame.quit()
```