আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম
International Islamic University Chittagong
Department of Computer & Communication Engineering(CCE)

PROJECT REPORT

Project No: 02
Project Name: Sprite Animation

Course Title: Computer Animation and Game Development Sessional
Course Code: CCE-3606

Submitted By

Student Name : Ahsanul Karim Tanim
Student Id : E221013
Semester : 6th
Section : A

Submitted To

Sukanta Paul,

Adjunct Faculty,
IIUC

Experiment Date: 26 / 11 / 2024

Submission Date: 03 / 12 / 2024

Remark

Project No: 2
Project Name: Sprite Animation

Process:

- **Initialization:**
  - Import the necessary Pygame module.
  - Initialize Pygame.
  - Set up the game window with desired dimensions and title.
- **Loading Assets:**
  - Load the background image.
  - Load the character's standing image.
  - Load the image sequences for walking left and right.
- **Game Loop:**
  - **Event Handling:**
    - Check for user input, such as keyboard presses or mouse clicks.
    - Handle the quit event to close the game.
  - **Game Logic:**
    - Update the character's position based on user input and game physics.
    - Update the animation counter to cycle through the sprite sheet frames.
  - **Rendering:**
    - Clear the screen.
    - Draw the background image.
    - Draw the character's current frame based on its animation state.
    - Update the display.
- **Main Loop:**
  - Continuously iterate through the game loop until the user quits the game.

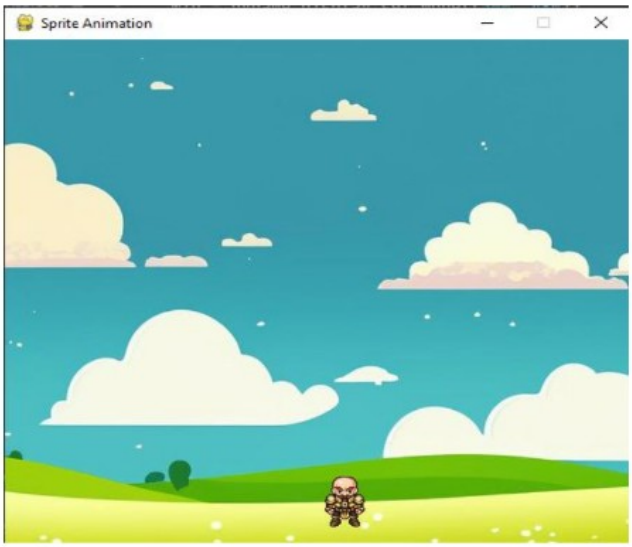
Code:

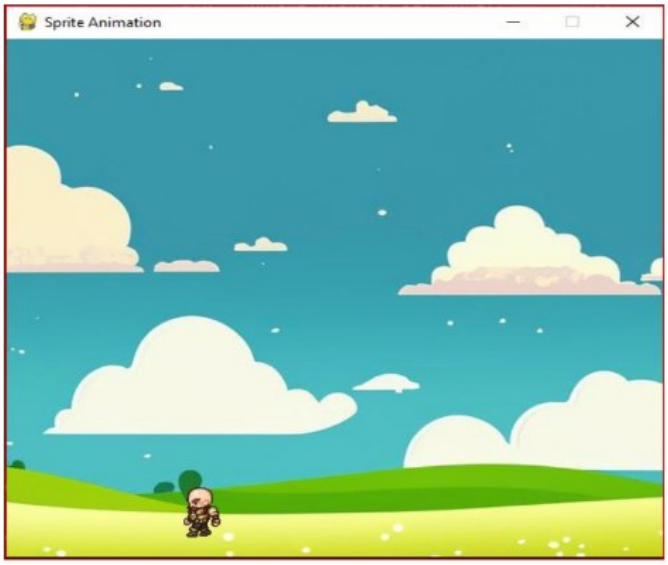
```
import pygame

# Initialize Pygame
pygame.init()

# Set up the window
win = pygame.display.set_mode((500, 480))
pygame.display.set_caption("Sprite Animation")

# Load images for animations and background
walkRight = [
pygame.image.load('Game/R1.png'),
pygame.image.load('Game/R2.png'),
pygame.image.load('Game/R3.png'),
pygame.image.load('Game/R4.png'),
pygame.image.load('Game/R5.png'),
pygame.image.load('Game/R6.png'),
pygame.image.load('Game/R7.png'),
pygame.image.load('Game/R8.png'),
pygame.image.load('Game/R9.png'),
]
walkLeft = [
pygame.image.load('Game/L1.png'),
pygame.image.load('Game/L2.png'),
pygame.image.load('Game/L3.png'),
pygame.image.load('Game/L4.png'),
pygame.image.load('Game/L5.png'),
pygame.image.load('Game/L6.png'),
pygame.image.load('Game/L7.png'),
pygame.image.load('Game/L8.png'),
pygame.image.load('Game/L9.png'),
]

bg = pygame.image.load('Game/bg1.jpg')
char = pygame.image.load('Game/standing.png')

# Character attributes
x, y = 50, 400
width, height = 40, 60
vel = 5

# Animation variables
clock = pygame.time.Clock()
isJump = False
jumpCount = 10
left = False
```

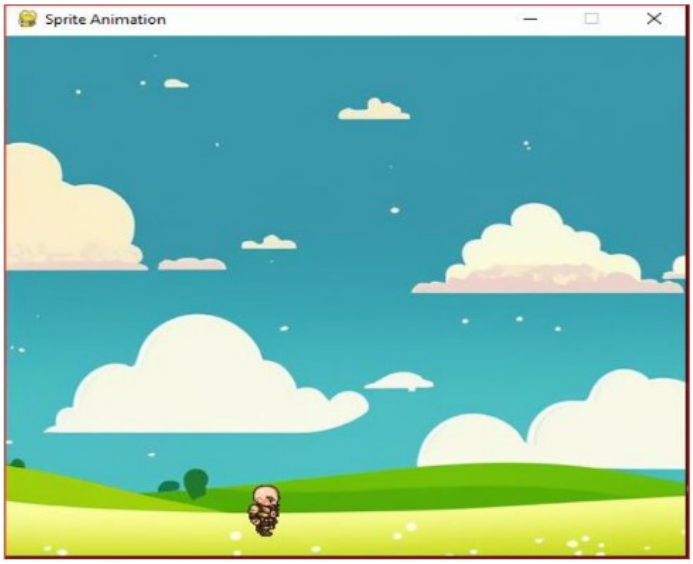
```
right = False
walkCount = 0

# Function to redraw the game window
def redrawGameWindow():
global walkCount
win.blit(bg, (0, 0))

# Handle animation frame updates
if walkCount + 1 >= 27:
walkCount = 0

if left:
win.blit(walkLeft[walkCount // 3], (x, y))
walkCount += 1
elif right:
win.blit(walkRight[walkCount // 3], (x, y))
walkCount += 1
else:
win.blit(char, (x, y))
walkCount = 0

pygame.display.update()

# Main game loop
run = True
while run:
clock.tick(27)

for event in pygame.event.get():
if event.type == pygame.QUIT:
run = False

keys = pygame.key.get_pressed()

# Horizontal movement
if keys[pygame.K_LEFT] and x > vel:
x -= vel
left = True
right = False
elif keys[pygame.K_RIGHT] and x < 500 - vel -
width:
x += vel
left = False
right = True
else:
left = False
right = False
walkCount = 0
```

```
# Jumping
if not isJump:
if keys[pygame.K_SPACE]:
isJump = True
left = False
right = False
walkCount = 0
else:
if jumpCount >= -10:
neg = 1 if jumpCount > 0 else -1
y -= (jumpCount ** 2) * 0.5 * neg
jumpCount -= 1
else:
jumpCount = 10
isJump = False

redrawGameWindow()

pygame.quit()
```

Output:







Discussion: This project focused on creating a basic 2D game using Python's Pygame library. We successfully implemented a character that can move left, right, and jump. The character's animation is driven by a sprite sheet, providing a smooth visual experience. The game loop handles user input, updates the character's position and animation, and renders the game world to the screen. While the current implementation is a simple foundation, it demonstrates the core concepts of game development using Pygame. Future improvements could involve adding more complex animations, advanced physics, and diverse game levels to enhance the overall gameplay experience.