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

LAB REPORT

Experiment No: 03
Experiment Name: A simple event program

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 Facuty,
IIUC

Experiment Date: / /

Submission Date: / /

Remark

Experiment No: 03
Experiment Name: A simple event program

Theory: Events in game development are actions or signals that occur during a game, such as pressing a key, clicking the mouse, or closing the game window.

Key Concepts

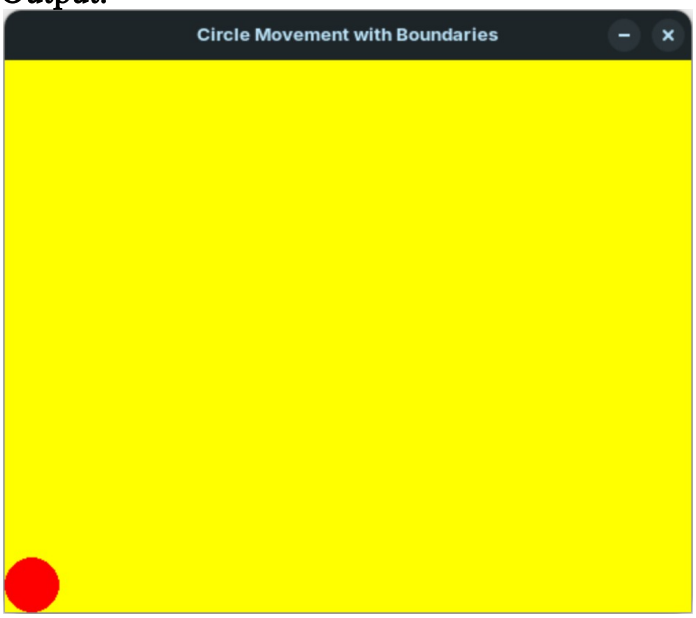
1. **Event Loops:** An event loop is a continuous cycle that checks for and processes events during the game. It ensures the game responds in real-time to inputs like key presses or mouse clicks while updating the visuals and logic.
2. **Event Handling:** Event handling is the process of identifying specific events (e.g., a key press) and deciding what the game should do in response. For example, pressing an arrow key might move a character, or clicking the mouse might select an object.
3. **Game State Updates Based on Events:** When events occur, the game's state is updated accordingly. For instance, if a player presses the jump button, the character's position changes to reflect the jump action.

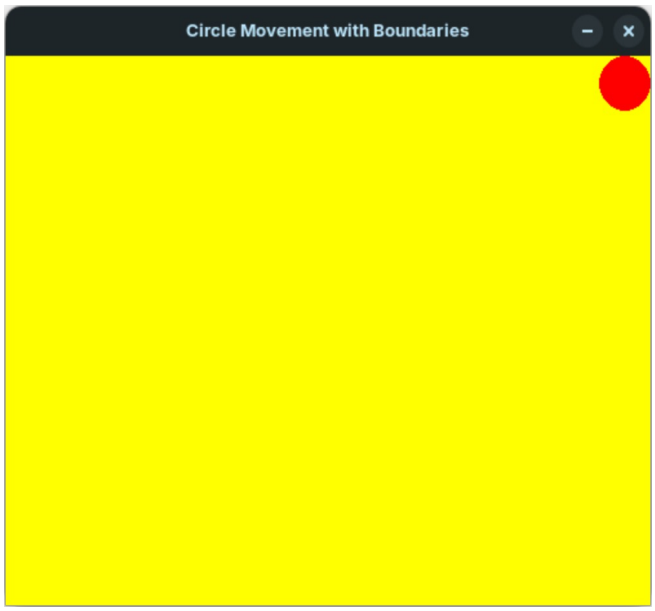
Code:

```
import pygame
pygame.init()
# Set up the screen
screen = pygame.display.set_mode((500, 400))
pygame.display.set_caption("Circle Movement with Boundaries")
# Define colors
YELLOW = (255, 255, 0)
RED = (255, 0, 0)
# Circle properties
circle_x = 250 # x-coordinate
circle_y = 200 # y-coordinate
circle_radius = 20
circle_speed = 5 # Speed of movement
# Game loop
running = True
while running:
screen.fill(YELLOW) # Fill the screen with the background color
# Draw the circle
pygame.draw.circle(screen, RED, (circle_x, circle_y), circle_radius)
# Event handling
for event in pygame.event.get():
if event.type == pygame.QUIT:
running = False
# Movement handling
keys = pygame.key.get_pressed()
if keys[pygame.K_UP] and circle_y - circle_radius > 0:
circle_y -= circle_speed # Move up
if keys[pygame.K_DOWN] and circle_y + circle_radius < 400:
circle_y += circle_speed # Move down
if keys[pygame.K_LEFT] and circle_x - circle_radius > 0:
circle_x -= circle_speed # Move left
if keys[pygame.K_RIGHT] and circle_x + circle_radius < 500:
circle_x += circle_speed # Move right
```

```
# Update the display
pygame.display.flip()
# Quit the program
pygame.quit()
```

Output:





Discussion: This lab showcase, the core concepts of event handling and object movement in game development, but several challenges and limitations were encountered. One limitation was ensuring smooth and consistent movement, as the responsiveness of the circle's movement depended on the keyboard input timing, which might feel abrupt without additional features like acceleration or easing. The implementation of boundary checks addressed logical issues but highlighted the need for more advanced collision detection methods in complex games. Additionally, debugging real-time interactions posed challenges, especially when ensuring the program responded correctly to multiple simultaneous inputs. Despite these issues, the exercise provided a clear understanding of how event-driven architectures work in games and emphasized the importance of refining user input handling and game logic for a smoother gameplay experience.