আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম

الجامعة الإسلامية العالمية شيتغونغ

International Islamic University Chittagong

Department of Computer & Communication Engineering(CCE)

PROJECT REPORT

Project No: 04
Project Name: Physics Simulation

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: 03 / 12 / 2024

Submission Date: / 12 / 2024

Remark

Project No: 4
Project Name: Physics Simulation

Process:

1. **Setup Environment:** Install Python and Pygame (`pip install pygame`), initialize Pygame, and create an 800x600 display.
2. **Define Ball Properties:** Set position, radius, velocity, and gravity.
3. **Add Controls:** Use keyboard inputs for movement:
    - **Left/Right Arrows** for horizontal motion.
    - **Spacebar** to jump (restrict double-jumps).
4. **Simulate Physics:** Apply gravity, update position, and handle collisions with walls and floor using elasticity.
5. **Render Graphics:** Clear the screen and redraw the ball at its new position.
6. **Frame Rate:** Run simulation smoothly at 60 FPS using `clock.tick(FPS)`.
7. **Exit Control:** End the program on user quit (`pygame.QUIT`).

Code:

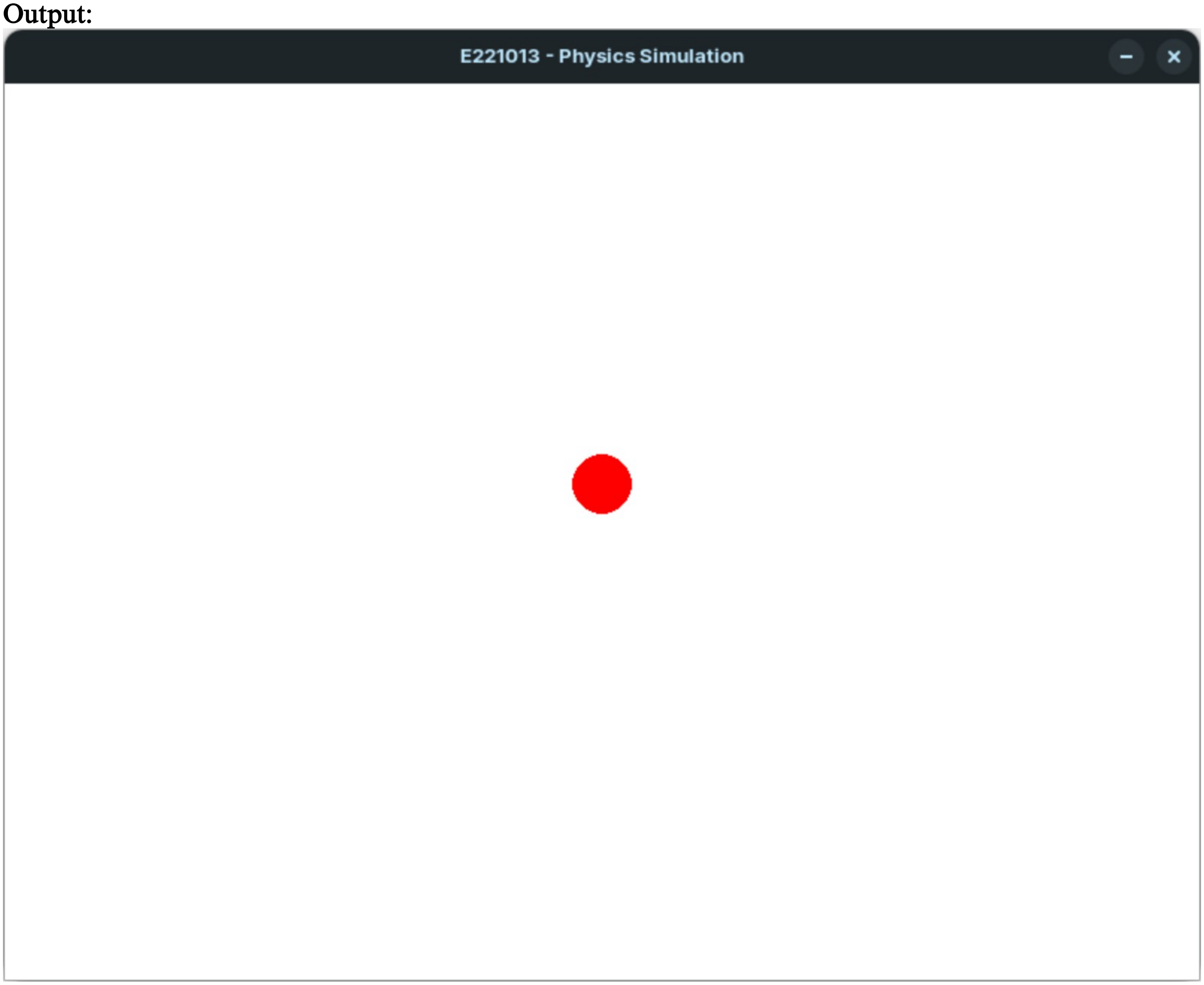
```
import pygame
import sys

# Initialize Pygame
pygame.init()

# Screen dimensions
WIDTH, HEIGHT = 800, 600
screen =
pygame.display.set_mode((WIDTH,
HEIGHT))
pygame.display.set_caption("E221013 -
Physics Simulation")

# Colors
WHITE = (255, 255, 255)
RED = (255, 0, 0)

# Clock for frame rate
clock = pygame.time.Clock()
FPS = 60

# Ball properties
ball_radius = 20
# Move right
elif keys[pygame.K_RIGHT] and
ball_position[0] + ball_radius < WIDTH:
ball_velocity[0] = move_speed
else:
```

```
ball_position = [WIDTH // 2, HEIGHT // 2]
# Start in the middle
ball_velocity = [0, 0] # Initial velocity (x, y)
gravity = 0.5 # Gravitational acceleration
jump_power = -18 # Upward force for
jumping
move_speed = 5 # Speed of left/right
movement
elasticity = 0.8 # Bounciness factorE221013
is_jumping = False # To prevent double
jumps

# Main game loop
running = True
while running:
for event in pygame.event.get():
if event.type == pygame.QUIT:
running = False

# Get keyboard inputs
keys = pygame.key.get_pressed()

# Move leftTanim
if keys[pygame.K_LEFT] and
ball_position[0] - ball_radius > 0:
ball_velocity[0] = -move_speed
# Control the frame rate
clock.tick(FPS)
```

```
ball_velocity[0] = 0 # Stop horizontal
movement when no key is pressed

# Jump when theTanim space key is pressed
if keys[pygame.K_SPACE] and not
is_jumping:
ball_velocity[1] = jump_power
is_jumping = True

# Physics calculations
ball_velocity[1] += gravity # Apply gravity
ball_position[0] += ball_velocity[0] #
Update horizontal position
ball_position[1] += ball_velocity[1] #
Update vertical position

# Collision with the floor
if ball_position[1] + ball_radius >=
HEIGHT:
ball_position[1] = HEIGHT - ball_radius #
Reset to floor level
ball_velocity[1] = -ball_velocity[1] *
elasticity # Bounce
is_jumping = False # Allow jumping again

# Collision with the ceiling
if ball_position[1] - ball_radius <= 0:
ball_position[1] = ball_radius
ball_velocity[1] = -ball_velocity[1] *
elasticity

# Collision with the walls
if ball_position[0] - ball_radius <= 0 or
ball_position[0] + ball_radius >= WIDTH:
ball_velocity[0] = -ball_velocity[0] *
elasticity # Bounce horizontally

# Clear theE221013 screen
screen.fill(WHITE)

# Draw the ball
pygame.draw.circle(screen, RED,
(int(ball_position[0]), int(ball_position[1])),
ball_radius)
```

```
# Update the display
pygame.display.flip()
# Quit Pygame
pygame.quit()
sys.exit()
```

Output:



Discussion: This experiment demonstrates a physics-based simulation that integrates interactive movement. The ball is subjected to gravity and responds to collisions with the screen boundaries, exhibiting bouncing behavior. Users can move the ball left or right and make it jump by pressing the respective keys. This blend of natural physics with user control enhances the realism and interactivity of the simulation, making it suitable for applications such as games or educational tools to visualize motion dynamics.