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

LAB REPORT

Experiment No: 08
Experiment Name: An experiment using vector

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: 08
Experiment Name: An experiment using vector

Process:

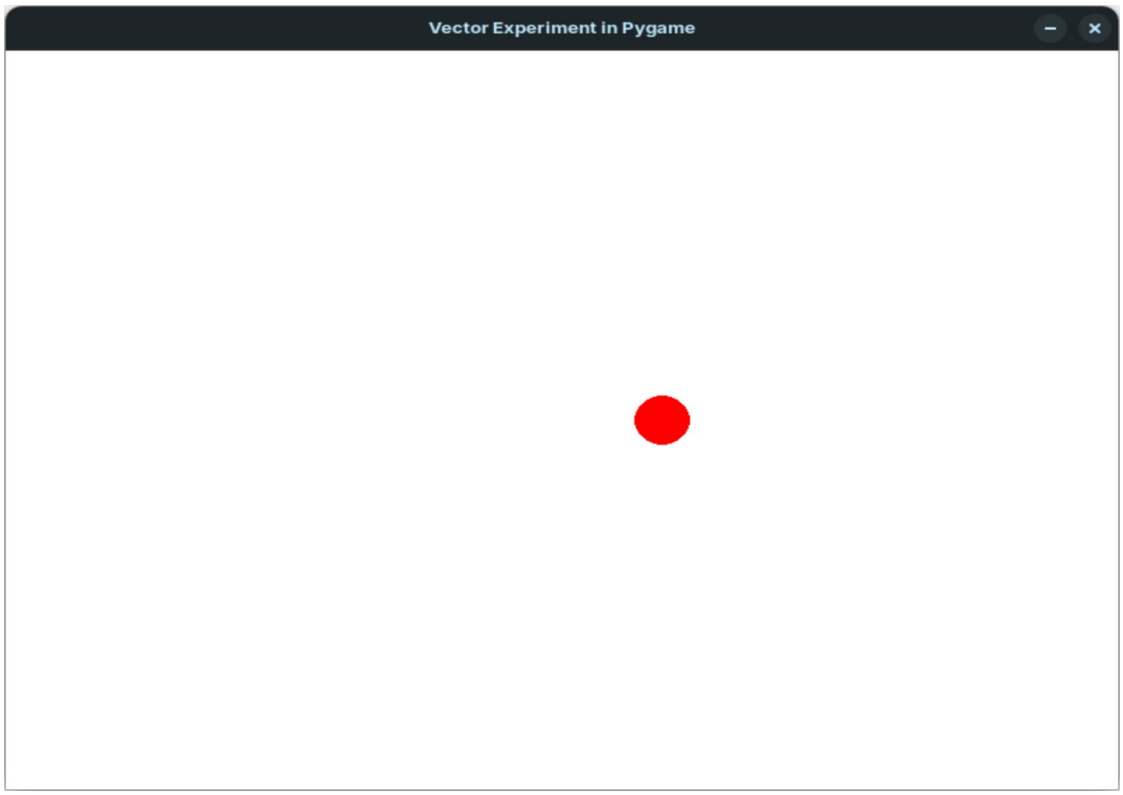
1. **Setup the Environment:**
   - Install Python and the Pygame library: Use the command pip install pygame to ensure Pygame is ready for use.
2. **Define the Experiment's Framework:**
   - Initialize Pygame and create a display window (WIDTH = 800, HEIGHT = 600).
   - Set up the frame rate using pygame.time.Clock() to control the speed of the simulation.
3. **Represent Objects Using Vectors:**
   - Define the ball's position and velocity using Pygame's Vector2 class:
     - ball_position holds the ball's current position.
     - ball_velocity determines the speed and direction of movement.
4. **Update the Ball's Position:**
   - Use vector addition (ball_position += ball_velocity) to update the ball's location in each frame.
5. **Handle Boundary Collisions:**
   - Check if the ball reaches the screen's edges:
     - If it touches the left or right edge, invert the x-component of the velocity (ball_velocity.x *= -1).
     - If it touches the top or bottom edge, invert the y-component of the velocity (ball_velocity.y *= -1).
6. **Draw and Display the Ball:**
   - Clear the screen in each frame with a background color.
   - Draw the ball at its updated position using pygame.draw.circle().
7. **Control Frame Rate:**
   - Use clock.tick(60) to ensure the simulation runs at 60 frames per second for smooth movement.
8. **Exit the Simulation:**
   - Monitor user inputs to close the window when the quit event is triggered (pygame.QUIT).

Code:

```
import pygame
from pygame.math import Vector2
# Initialize Pygame
pygame.init()
# Screen settings
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Vector Experiment in Pygame")
# ColorsE221013
WHITE = (255, 255, 255)
RED = (255, 0, 0)
# Clock to control frame rate
clock = pygame.time.Clock()
# Ball propertiesTanim
ball_position = Vector2(WIDTH // 2, HEIGHT // 2) # Starting position
ball_velocity = Vector2(4, 3) # Velocity vector (speed and direction)
ball_radius = 20
# Main loopE221013
running = True
while running:
```

```
for event in pygame.event.get():
if event.type == pygame.QUIT:
running = False
# Clear the screen
screen.fill(WHITE)
# Update ball position using vector addition
ball_position += ball_velocity
# Check for collisions with screen boundaries and reflect velocity
if ball_position.x - ball_radius <= 0 or ball_position.x + ball_radius >= WIDTH:
ball_velocity.x *= -1 # Reverse x-direction
if ball_position.y - ball_radius <= 0 or ball_position.y + ball_radius >= HEIGHT:
ball_velocity.y *= -1 # Reverse y-direction
# Draw the ball
pygame.draw.circle(screen, RED, (int(ball_position.x), int(ball_position.y)), ball_radius)
# Update the display
pygame.display.flip()
# Limit frame rate
clock.tick(60)
# Quit Pygame
pygame.quit()
```

Output:



Discussion: This experiment demonstrates the use of vectors in simulating motion and collisions in 2D space. The ball's position is updated using vector addition, while collisions with boundaries are handled by inverting the respective velocity components. By using Pygame's Vector2 class, the mathematical complexity of vector operations is significantly reduced. The experiment highlights key physics principles, such as uniform motion and reflection, making it a foundational example for understanding motion in games and simulations. While simplified, this model serves as a stepping stone for more complex implementations like gravity, friction, and object-to-object collisions.