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

LAB REPORT

Experiment No: 07
Experiment Name: Diagonal Movement

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: 07
Experiment Name: Diagonal Movement

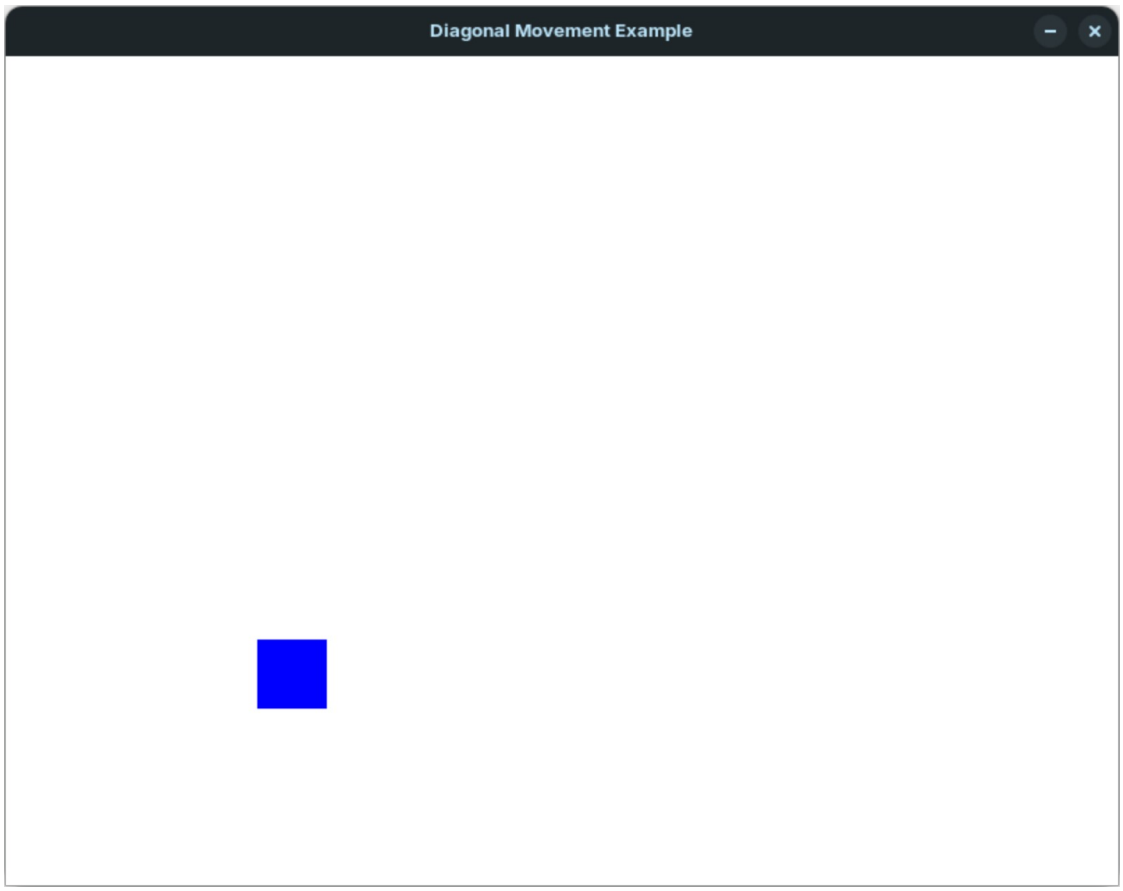
Theory: Diagonal movement is a crucial mechanic in 2D games, allowing intuitive navigation through simultaneous horizontal and vertical movement. The implementation involves creating a 2D game-window using Pygame, defining a player object, and handling keyboard inputs (W, A, S, D keys or arrow keys). When two directional keys are pressed together, the player moves diagonally. The game logic includes screen setup, input handling, and continuous rendering to achieve smooth game-play.

Code:

```
import pygame
import sys
# Initialize Pygame
pygame.init()
# Screen dimensions
WIDTH, HEIGHT = 800, 600
# Colors
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
# Player properties
player_size = 50
player_pos = [WIDTH // 2, HEIGHT // 2]
player_speed = 5
# Set up the game window
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Diagonal Movement Example")
# Clock to control frame rate
clock = pygame.time.Clock()
# Movement logic
move_x, move_y = 0, 0
# Main game loop
running = True
while running:#E221013
for event in pygame.event.get():
if event.type == pygame.QUIT:
running = False
# Get all keys pressed
keys = pygame.key.get_pressed()
# Movement logic
if keys[pygame.K_w]: # Move up
move_y -= 1
if keys[pygame.K_s]: # Move down
move_y += 1
if keys[pygame.K_a]: # Move left
move_x -= 1
if keys[pygame.K_d]: # Move right
move_x += 1#Tanim
# Normalize diagonal movement
if move_x != 0 and move_y != 0:
move_x *= 0.7071 # 1/sqrt(2)
move_y *= 0.7071
```

```
# Update player positionTanim
player_pos[0] += move_x * player_speed
player_pos[1] += move_y * player_speed
# Ensure the player stays within the screen boundaries
player_pos[0] = max(0, min(WIDTH - player_size, player_pos[0]))
player_pos[1] = max(0, min(HEIGHT - player_size, player_pos[1]))
# Clear screen
screen.fill(WHITE)
# Draw playerE221013
pygame.draw.rect(screen, BLUE, (*player_pos, player_size, player_size))
# Update the display
pygame.display.flip()
# Cap the frame rate
clock.tick(60)
# Quit Pygame
pygame.quit()
sys.exit()
```

Output:



Discussion: This lab successfully demonstrates diagonal movement by processing simultaneous key presses, a fundamental element of 2D character control. Adjusting player_speed allows fine-tuning movement speed, and integrating a sprite can enhance the visual experience. Adding boundary constraints or further movement mechanics could expand the functionality, making it suitable for more complex games.