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

LAB REPORT

Experiment No: 06
Experiment Name: Time based 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 Facuty,
IIUC

Experiment Date: / /

Submission Date: / /

Remark

Experiment No: 06
Experiment Name: Time based Animation

Process:

1. **Initialize Pygame:** Set up the Pygame environment, display window, and frame rate.
2. **Define Game Elements:** Create the player, target, and obstacles with their initial positions, sizes, and colors.
3. **Game Loop:** Continuously run the game until the player quits or collides with an obstacle.
4. **Player Movement:** Capture user input (arrow keys) to move the player in four directions (up, down, left, right).
5. **Collision Detection with Target:** Check if the player collides with the target.
   If a collision occurs:
   - Increase the score.
   - Randomly reposition the target.
   - Relocate the obstacles to new random positions.
   - Increase the speed of the obstacles.
6. **Obstacle Movement:**
   - Move the obstacles vertically across the screen.
   - Reverse their direction when they hit the top or bottom boundaries.
7. **Collision Detection with Obstacles:**
   - Check if the player collides with any obstacle.
   - If a collision occurs, display "Game Over!" and end the game.
8. **Render Graphics:**
   - Clear the screen and redraw the player, target, obstacles, and score in each frame.
9. **Update the Display:**
   - Refresh the display to show the latest game state.
10. **Control Frame Rate:**
    - Limit the frame rate to ensure smooth gameplay

Code:

```
import pygame
import sys
import random
# Initialize Pygame
pygame.init()
# Screen dimensions
WIDTH, HEIGHT = 800, 600
# Set up display
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Time Base Animation")
# Clock for controlling frame rate
clock = pygame.time.Clock()
# Colors
WHITE = (255, 255, 255)
BLUE = (0, 0, 255)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLACK = (0, 0, 0)
# Font for displaying score
font = pygame.font.Font(None, 50)
# Player properties
player_radius = 20
```

```
player_x, player_y = WIDTH // 2, HEIGHT // 2
player_speed = 5
# Target properties
target_radius = 15
target_x = random.randint(target_radius, WIDTH - target_radius)
target_y = random.randint(target_radius, HEIGHT - target_radius)
# Obstacles properties
initial_obstacle_speed = 3
obstacle_speed = initial_obstacle_speed
obstacles = [#E221013
pygame.Rect(random.randint(0, WIDTH - 50), random.randint(0, HEIGHT - 50), 40, 40),
pygame.Rect(random.randint(0, WIDTH - 50), random.randint(0, HEIGHT - 50), 40, 40),
]
# Game variables
score = 0
running = True
# Function to display the score
def display_score():
score_text = font.render(f"Score: {score}", True, WHITE)
screen.blit(score_text, (10, 10))
# Main game loop
while running:#Tanim
for event in pygame.event.get():
if event.type == pygame.QUIT:
running = False
# Move player with arrow keys
keys = pygame.key.get_pressed()
if keys[pygame.K_LEFT] and player_x - player_radius > 0:
player_x -= player_speed
if keys[pygame.K_RIGHT] and player_x + player_radius < WIDTH:
player_x += player_speed#E221013
if keys[pygame.K_UP] and player_y - player_radius > 0:
player_y -= player_speed
if keys[pygame.K_DOWN] and player_y + player_radius < HEIGHT:
player_y += player_speed
# Check collision with target
if (player_x - target_x) ** 2 + (player_y - target_y) ** 2 < (player_radius + target_radius) ** 2:
score += 1
target_x = random.randint(target_radius, WIDTH - target_radius)
target_y = random.randint(target_radius, HEIGHT - target_radius)
obstacle_speed += 0.5 # Increase obstacle speed with every score
# Move obstacles to new random positions
for obstacle in obstacles:
obstacle.x = random.randint(0, WIDTH - 50)
obstacle.y = random.randint(0, HEIGHT - 50)
# Move obstacles and check for collisions
for obstacle in obstacles:
obstacle.y += obstacle_speed
if obstacle.bottom >= HEIGHT or obstacle.top <= 0:
obstacle_speed = -obstacle_speed # Reverse direction
```

```
if pygame.Rect(player_x - player_radius, player_y - player_radius, player_radius * 2,
player_radius * 2).colliderect(obstacle):
print("Game Over!")
running = False#Tanim
# Draw background, player, target, and obstacles
screen.fill(BLACK)
pygame.draw.circle(screen, BLUE, (player_x, player_y), player_radius)
pygame.draw.circle(screen, GREEN, (target_x, target_y), target_radius)
for obstacle in obstacles:
pygame.draw.rect(screen, RED, obstacle)
# Display score
display_score()
# Update the display
pygame.display.flip()
# Control the frame rate
clock.tick(60)
# End game
pygame.quit()
sys.exit()
```

Output:

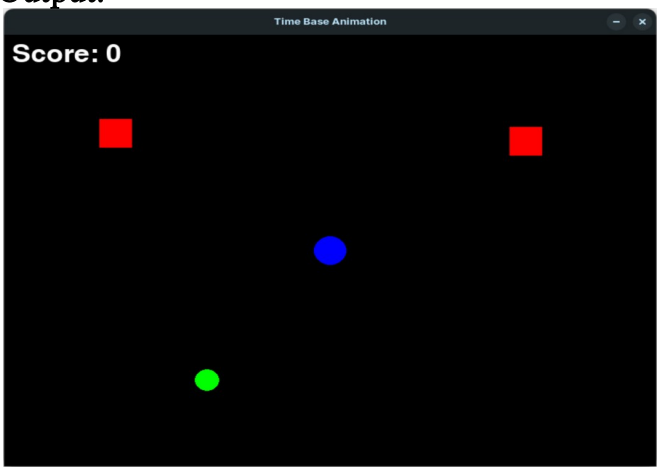

Fig 1: Start of the game

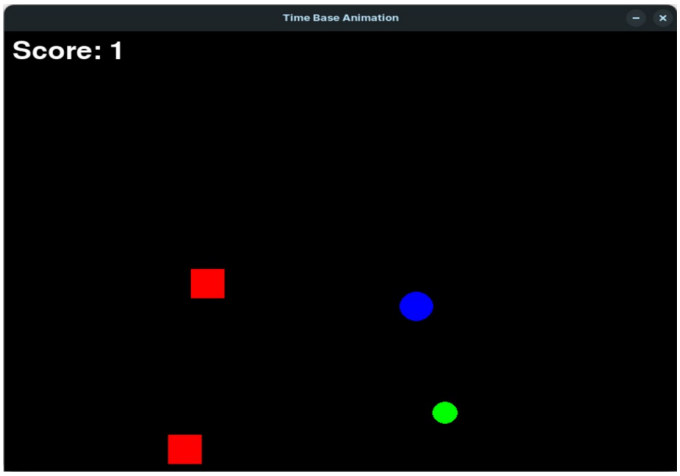

Fig 2: Scored one

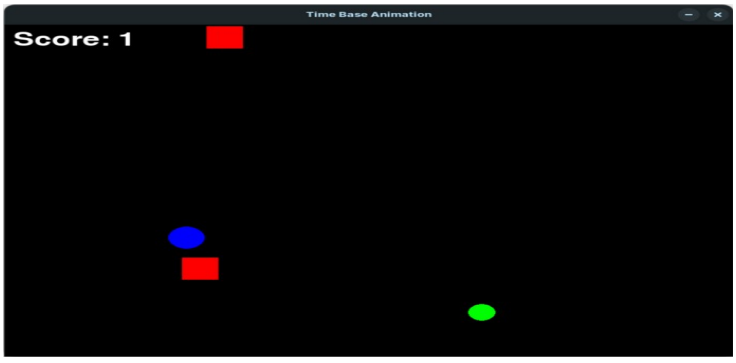

Fig 3: Game Ending moment

Discussion: The "Collect the Targets" game effectively demonstrates the use of Pygame for time-based animations, collision detection, and real-time user interaction. The increasing difficulty and randomized elements make the game engaging, while the simple controls make it accessible to a wide audience. With further enhancements, this game could provide an even richer and more complex gameplay experience.