আন্তর্জাতিক ইসলামী বিশ্ববিদ্যালয় চট্টগ্রাম

الجامعة الإسلامية العالمية شيتاغونغ

International Islamic University Chittagong

Department of Computer & Communication Engineering(CCE)

PROJECT REPORT

Project No: 03
Project Name: AI for enemy behavior

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: 10 / 12 / 2024

Remark

Project No: 3
Project Name: AI for enemy behavior

Process:

1. **Objective**: Create a game in Pygame where enemies exhibit intelligent behavior using basic AI principles.
2. **Player and Enemy**:
   - The player moves with arrow keys.
   - Enemies move toward the player or follow predefined patterns.
3. **AI for Enemies**:
   - Enemies use distance calculations to chase the player.
   - Some enemies patrol predefined paths.
4. **Collision**:
   - The game ends when an enemy collides with the player.

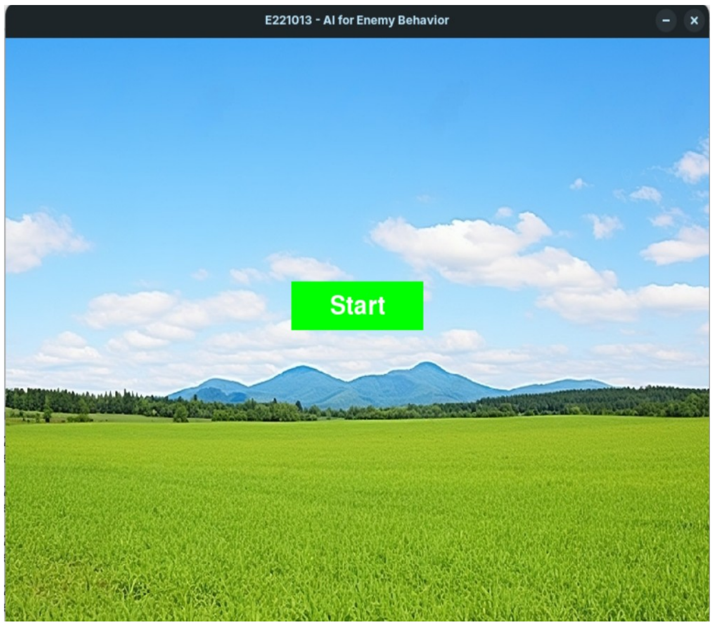
Code:

```
import pygame
import math
import sys
from datetime import datetime

# Initialize Pygame
pygame.init()

# Screen dimensions
WIDTH, HEIGHT = 800, 600
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("E221013 - AI for Enemy Behavior")

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

# Load images
background_image = pygame.image.load("field.png")
player_image = pygame.image.load("deer.png")
enemy_image = pygame.image.load("tiger.png")

# Scale images
background_image = pygame.transform.scale(background_image, (WIDTH, HEIGHT))
player_image = pygame.transform.scale(player_image, (80, 80))
enemy_image = pygame.transform.scale(enemy_image, (50, 50))

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

# Fonts
font = pygame.font.Font(None, 40)
```

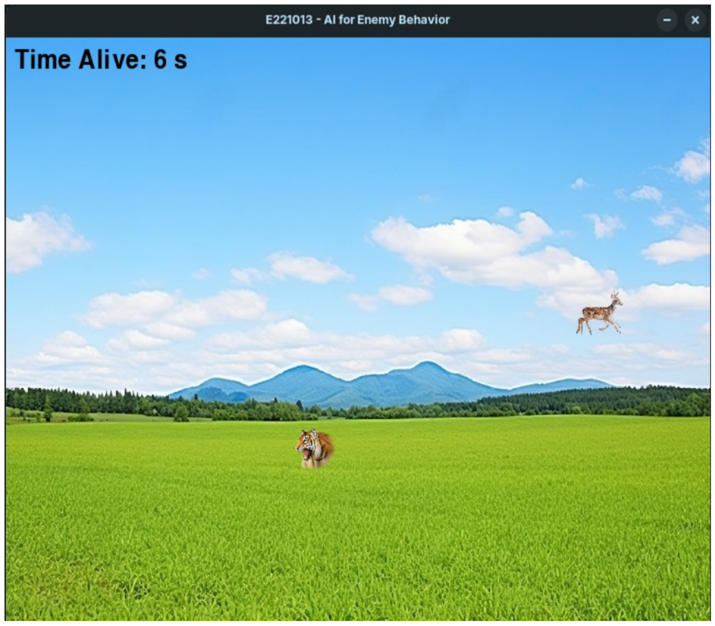
```
# Button class
class Button:
def __init__(self, x, y, width, height, text, color, hover_color, action=None):
self.rect = pygame.Rect(x, y, width, height)
self.text = text
self.color = color
self.hover_color = hover_color
self.action = action

def draw(self, screen):
mouse_pos = pygame.mouse.get_pos()
color = self.hover_color if self.rect.collidepoint(mouse_pos) else self.color
pygame.draw.rect(screen, color, self.rect)#E221013 - Tanim
text_surface = font.render(self.text, True, WHITE)
text_rect = text_surface.get_rect(center=self.rect.center)
screen.blit(text_surface, text_rect)

def is_clicked(self, event):
if event.type == pygame.MOUSEBUTTONDOWN and self.rect.collidepoint(event.pos):
return True
return False

# Player class
class Player(pygame.sprite.Sprite):
def __init__(self):
super().__init__()
self.image = player_image
self.rect = self.image.get_rect(center=(WIDTH // 2, HEIGHT // 2))
self.speed = 5

def update(self, keys):
if keys[pygame.K_UP]:
self.rect.y -= self.speed
if keys[pygame.K_DOWN]:
self.rect.y += self.speed
if keys[pygame.K_LEFT]:
self.rect.x -= self.speed
if keys[pygame.K_RIGHT]:
self.rect.x += self.speed

# Keep player within bounds
self.rect.x = max(0, min(self.rect.x, WIDTH - self.rect.width))
self.rect.y = max(0, min(self.rect.y, HEIGHT - self.rect.height))

# Enemy class
class Enemy(pygame.sprite.Sprite):
def __init__(self, x, y, behavior="chase"):
super().__init__()
self.image = enemy_image
self.rect = self.image.get_rect(center=(x, y))
self.speed = 3
self.behavior = behavior

def update(self, player_pos):
if self.behavior == "chase":
dx, dy = player_pos[0] - self.rect.x, player_pos[1] - self.rect.y
distance = math.sqrt(dx**2 + dy**2)
if distance != 0:
self.rect.x += int(self.speed * dx / distance)
self.rect.y += int(self.speed * dy / distance)

# Initialize game variables
player = Player()
enemies = pygame.sprite.Group(
Enemy(700, 500, behavior="chase")
)
all_sprites = pygame.sprite.Group(player, enemies)

start_time = None
game_running = False
game_over = False
survival_time = None

# Buttons
start_button = Button(WIDTH // 2 - 75, HEIGHT // 2 - 50, 150, 50, "Start", GREEN, (0, 200, 0))
restart_button = Button(WIDTH // 2 - 100, HEIGHT // 2 - 50, 150, 50, "Restart", GREEN, (0, 200, 0))
exit_button = Button(WIDTH // 2 - 100, HEIGHT // 2 + 20, 150, 50, "Exit", RED, (200, 0, 0))#E221013
# Main game loop
running = True
while running:
screen.fill(WHITE)

for event in pygame.event.get():
if event.type == pygame.QUIT:
running = False

if not game_running and not game_over and start_button.is_clicked(event):
game_running = True
start_time = datetime.now()
player = Player()
enemies = pygame.sprite.Group(
Enemy(700, 500, behavior="chase")
)
all_sprites = pygame.sprite.Group(player, enemies)

if game_over and restart_button.is_clicked(event):
game_running = True
game_over = False
start_time = datetime.now()
player = Player()#E221013 - Tanim
```

```
enemies = pygame.sprite.Group(
Enemy(700, 500, behavior="chase")
)
all_sprites = pygame.sprite.Group(player, enemies)

if game_over and exit_button.is_clicked(event):
pygame.quit()
sys.exit()

# Gameplay
if game_running:
keys = pygame.key.get_pressed()

# Update sprites
player.update(keys)
for enemy in enemies:
enemy.update(player.rect.center)

# Check for collisions
if pygame.sprite.spritecollideany(player, enemies):
game_running = False
game_over = True
survival_time = datetime.now() - start_time

# Draw everything
screen.blit(background_image, (0, 0)) # Draw the background
all_sprites.draw(screen)

# Draw survival time
elapsed_time = (datetime.now() - start_time).seconds
time_text = font.render(f"Time Alive: {elapsed_time} s", True, BLACK)
screen.blit(time_text, (10, 10))
# Main menu or "Game Over" screen
else:
screen.blit(background_image, (0, 0))
if game_over:
survival_time_text = font.render(f"Survived: {survival_time.seconds} seconds", True, BLACK)
screen.blit(survival_time_text, (WIDTH // 2 - survival_time_text.get_width() // 2, HEIGHT // 2 - 100))
font_big = pygame.font.Font(None, 60)#E221013 - Tanim
dead_text = font_big.render("DEAD", True, RED)
screen.blit(dead_text, (WIDTH // 2 - dead_text.get_width() // 2, HEIGHT // 2 - 150))
restart_button.draw(screen)
exit_button.draw(screen)
else:
start_button.draw(screen)

pygame.display.flip()
clock.tick(60)

pygame.quit()
```

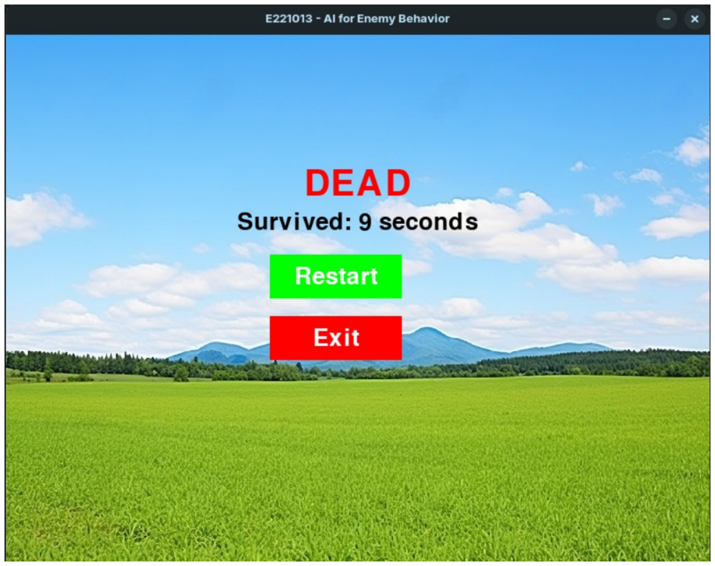
Output:



Starting Screen



Game Screen



Game End Screen

Limitations:

- **Basic AI**: The AI uses simple distance-based calculations or predefined paths. It lacks advanced algorithms like pathfinding.
- **Performance**: More enemies or larger maps may lead to lag.
- **Collision Detection**: The game ends upon collision, but no scoring or retry mechanism is implemented.
- **Limited Behaviors**: Only "chase" and "patrol" behaviors are implemented.

Discussion: This project demonstrates basic AI behaviors in a game setting using Pygame. The "chase" and "patrol" behaviors highlight how AI can enhance game interactivity. Adding more behaviors like fleeing, ambushing, or using obstacles for cover would further improve the AI. Additionally, integrating pathfinding algorithms like A* or Dijkstra's could make enemy movement more realistic, especially in complex maps.