21CSC206T - ARTIFICIAL INTELLEGENCE

TITLE: SNAKE GAME AI

Aim:

To implement an AI-controlled Snake Game using a Greedy Algorithm in Python where:

- The snake navigates a grid environment.
- The AI calculates the shortest path to the food.
- The snake grows upon eating food and avoids collisions.

Problem Statement:

Design and implement a Snake Game with the following features:

- A grid-based game board.
- Random food placement.
- AI-controlled snake that finds the shortest path to food using a Greedy algorithm.
- Avoids walls and self-collision.

The system should:

- Display the snake and food in real-time.
- Move the snake automatically using AI decisions.
- Show the score based on food collected.

Algorithm used:

Greedy Algorithm: The AI chooses the next move that brings the snake closest to the food based on the Manhattan distance.

Algorithm:

1. Initialize Game Grid and Snake Position.
2. Randomly place food on the grid.
3. At each move:
   - Check all valid directions (up, down, left, right).
   - Calculate the Manhattan distance to food for each.
   - Choose the move with the minimum distance.
   - Move the snake and update its body.
   - End game if collision occurs.
4. Display updated grid.

CODE:

```
import pygame
import sys
import random

# Initialize
pygame.init()

# Constants
CELL_SIZE = 20
GRID_WIDTH = 30
GRID_HEIGHT = 20
WIDTH = GRID_WIDTH * CELL_SIZE
HEIGHT = GRID_HEIGHT * CELL_SIZE
FPS = 10

# Colors
BLACK = (0, 0, 0)
GREEN = (0, 200, 0)
RED = (200, 0, 0)

# Setup
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Smart Snake AI")
clock = pygame.time.Clock()

# Snake and Food
snake = [(5, 5)]
direction = (1, 0)
food = (random.randint(0, GRID_WIDTH-1), random.randint(0, GRID_HEIGHT-1))

def draw():
    screen.fill(BLACK)
    for segment in snake:
```

```
 pygame.draw.rect(screen, GREEN, (segment[0]*CELL_SIZE,
segment[1]*CELL_SIZE, CELL_SIZE, CELL_SIZE))
 pygame.draw.rect(screen, RED, (food[0]*CELL_SIZE, food[1]*CELL_SIZE,
CELL_SIZE, CELL_SIZE))
 pygame.display.flip()

def is_collision(pos):
 return (
 pos in snake or
 pos[0] < 0 or pos[0] >= GRID_WIDTH or
 pos[1] < 0 or pos[1] >= GRID_HEIGHT
 )

def manhattan_distance(a, b):
 return abs(a[0] - b[0]) + abs(a[1] - b[1])

def get_next_move():
 head = snake[0]
 options = [(1, 0), (-1, 0), (0, 1), (0, -1)]
 valid_moves = []

 for dx, dy in options:
 new_pos = (head[0] + dx, head[1] + dy)
 if not is_collision(new_pos):
 dist = manhattan_distance(new_pos, food)
 valid_moves.append((dist, (dx, dy)))

 if valid_moves:
 valid_moves.sort()
 return valid_moves[0][1]
 else:
 return (0, 0) # No move possible

def update_snake():
 global food
 new_head = (snake[0][0] + direction[0], snake[0][1] + direction[1])

 if is_collision(new_head):
 pygame.quit()
 sys.exit()

 snake.insert(0, new_head)

 if new_head == food:
 food = (random.randint(0, GRID_WIDTH-1), random.randint(0, GRID_HEIGHT-1))
 while food in snake:
 food = (random.randint(0, GRID_WIDTH-1), random.randint(0, GRID_HEIGHT-1))
 else:
 snake.pop()
```

```
 # Game Loop
while True:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    direction = get_next_move()
    update_snake()
    draw()
    clock.tick(FPS)
```

Result: The Greedy Algorithm successfully controls the snake to find and reach the food while avoiding collisions.

Team members:

RA2311042010017 - Krishna P

RA2311042010019 - K Manthra