Shri Vile Parle Kelavani Mandal's

INSTITUTE OF TECHNOLOGY

DHULE (M.S.)

DEPARMENT OF COMPUTER ENGINEERING

Name: Sakshi Manojkumar Pagariya **Roll No.**: 17

Subject : Artificial Intelligence Lab **Subject Code :** BTCOL707

Class: BTech Computer. **Expt. No. :** 09

Title : Non-player characters (NPCs): Creating intelligent Al-controlled characters in games.

Problem Staement :	Non-player characters (NPCs): Creating intelligent Al-controlled characters in games.
Software Required :	Prolog
Theory :	Intelligent AI-Controlled NPCs in Games Non-Player Characters (NPCs) play a crucial role in enhancing the overall gaming experience. Creating intelligent AI-controlled NPCs involves implementing algorithms that simulate human-like behavior, decision-making, and interactions within the game environment. Various techniques, such as Finite State Machines, Behavior Trees, and Machine Learning, can be employed to enhance the realism and responsiveness of NPCs. Experiment Steps: 1. NPC Behavior Design: Define the desired behaviors and interactions of NPCs in the game.

2. Decision-Making Algorithms: Implement decision-making algorithms to enable NPCs to make informed choices based on the game context.
3. Finite State Machines (FSM): Use FSM to model different states and transitions in NPC behavior.
4. Behavior Trees: Implement Behavior Trees to represent complex decision-making processes and actions.
5. Pathfinding Algorithms: Integrate pathfinding algorithms to enable NPCs to navigate the game environment intelligently.
6. Testing and Evaluation: Test the NPCs in various game scenarios to evaluate their responsiveness and behavior.

```
% Prolog Code for AI-controlled NPCs in Games

% Example Facts (Replace with your actual game data)
player_health(70).
enemy_health(50).
% ... (more game-related facts)

% Decision-Making Rules (Modify and expand based on your game logic)
make_decision(AttackDecision) :-
    player_health(PlayerHealth),
    enemy_health(EnemyHealth),

    % Rule 1: Attack if player health is lower than enemy health
    PlayerHealth < EnemyHealth,
    AttackDecision = attack.

make_decision(AttackDecision) :-
    player_health(PlayerHealth),
    enemy_health(EnemyHealth),

    % Rule 2: Retreat if player health is higher than enemy health
    PlayerHealth >= EnemyHealth,
```

	AttackDecision = retreat. % Main NPC Behavior Loop npc_behavior_loop :- % Implement game loop logic here % Call the decision-making function and perform actions accordingly make_decision(Decision), perform_action(Decision), % Continue the loop... % Perform Action (Replace with your actual game actions) perform_action(attack) :- write('NPC decides to attack the player!'), nl. perform_action(retreat) :- write('NPC decides to retreat from the player!'), nl. % Example Usage :- npc_behavior_loop.
Conclusion:	This experiment explores the fascinating realm of creating intelligent AI-controlled NPCs in games. The implementation of diverse algorithms and techniques allows game developers to enhance the realism and engagement of NPCs, contributing significantly to the overall gaming experience. Understanding the principles of AI in game development opens up avenues for creating immersive and dynamic virtual worlds..