Reinforcement Learning Game from Scratch

The goal of this project is to design, implement, and analyze a game environment where reinforcement learning (RL) agents can learn optimal policies. The project will require students to apply three different RL algorithms (at least one based on dynamic programming) to train agents and compare their performance.

Team Composition: 3 Students.

Requirements:

1. **Game Environment:**
    - Design a 2D game using Python libraries (e.g., Pygame or custom implementations).
    - The environment should include:
        - **Obstacles**: Fixed or dynamic obstacles the agent must avoid.
        - **Rewards**: Collectibles that provide positive reinforcement.
        - **Goals**: Endpoints that agents should reach for high rewards.
    - Provide visualization tools to display the game and agent's performance.

    What You'll Build

    You will design and implement simple 2D games such as:

    1. **Gridworld Game**: An agent moves on a grid to reach a goal while avoiding obstacles.
    2. **Catch Game**: An agent catches falling objects to gain rewards.
    3. **Snake Game** (optional): An RL agent plays the classic snake game.
2. **Reinforcement Learning Algorithms:**
    - Implement the following algorithms from scratch:
        1. **Dynamic Programming**: Policy Iteration or Value Iteration.
        2. **Temporal Difference**: Q-Learning or SARSA.
        3. **Policy-Based Method**: Any variant of Policy Gradient.
    - Train the agents using these algorithms and evaluate their policies.

3. **Analysis and Comparison:**
    - Compare the performance of each algorithm based on metrics such as convergence speed, reward accumulation, and computational efficiency.
    - Visualize the training progress with graphs (e.g., reward vs. episodes, convergence plots).
4. **Documentation:**
    - **Code Documentation:** Clear comments and organized structure for readability.
    - **Final Report:** Include:
        - Overview of the game environment.
        - Detailed explanation of each RL algorithm.
        - Experimental results and analysis.
        - Challenges faced and solutions implemented.
    - **Presentation:** Demonstrate the game and agent performance in class.
5. **Evaluation Criteria**

Criterion	Weight
Game Design and Environment	20%
Correctness of RL Algorithms	30%
Analysis and Comparison	25%
Documentation and Report	15%
Presentation and Demo	10%