Manual_for Python w/ Project 2

Overview

This guide explains how to set up and execute a Python-based project where AI algorithms (Minimax or MCTS) are implemented for automated gameplay within time constraints.

Requirements

- **Python 3** is needed to ensure compatibility with syntax and libraries.
- **Pygame** library for game development and running the graphical interface.
- **Command Line Interface** (CLI) for running the game with different settings.

Setup

Download the Framework

- Start by downloading the latest version of the provided Python game framework for Project 2 from E3.

Framework Structure

- **Main Module**: Contains the game loop and the primary game logic.
- **Hexagon Board**: Manages the game state through a global `hexagon_board` dictionary mapping `(row, col)` to dictionaries with keys `x`, `y`, and `label`.
- **Player Interaction**: Controlled through command-line arguments to specify player types.

Implementing Your AI

1. Modify AI Functions:

- Replace the `select_hexes_by_random` function with your AI logic utilizing Minimax or MCTS algorithms.

2. Ensure Timely Decisions:

- AI decisions must comply with a 30-second time limit per move. Implement checks and fallbacks within your algorithm to handle scenarios where a decision cannot be made within the limit.

3. Interact with Game State:

- Use the **`hexagon_board`** dictionary to access and modify the game state for decision-making.

Key Functions to Implement

- `**select_hexes_by_random**`: Initially picks hexes randomly; you must integrate AI logic into this function. Ensure compliance with the 30-second rule by incorporating timeout checks.

- `**check_timeout_and_autocomplete**`: Modify to manage the 30-second constraint effectively, forcing a fallback to random selection if the AI's processing time exceeds the limit.

Running the Game

To run the game, use command-line arguments to define the types of players:

```
python main_r4.py [player1_type] [player2_type]
```

Where `player1_type` and `player2_type` should be 'human' or 'random'.

Examples:

- `python main.py human random`: Player 1 is a human, and Player 2 is controlled by AI.
- `python main.py random human`: Player 1 is controlled by AI, and Player 2 is a human.

Testing Your AI

- **Simulate Games**: Execute multiple game simulations to assess the effectiveness and reliability of your AI.
- **Logging**: Implement logging within your AI logic to trace decisions and outcomes for debugging and strategy refinement.

Deployment

Package your application ensuring all dependencies are correctly configured. Include a README with comprehensive instructions on running your AI in different configurations and scenarios.

Conclusion

This guide provides you with the instructions needed to set up, implement, and test your AI within the Python framework for Project 2. Focus on the AI's performance, particularly its ability to make strategic decisions within the strict time constraints.