*Artificial intelligence -Spring 2024  
National Yang Ming Chiao Tung University  
Instructor: Yuh-Jyh Hu*

**Term Project – Framework & Implementation**

Please read this file carefully to understand the details of **the framework and implementation**.

**Outline**

* **Framework**
  + **Coordinate**
  + **Python framework**
    - **Framework Description**
    - **Implementation Detail**
  + **C++ framework**
    - **Framework Description**
    - **Implementation Detail**

**Framework**

We provide **C++** and **Python** framework for Term Project. You can choose the one that you’re more familiar to implement your Minimax and MCTS code.

**Coordinate**

Both C++ and Python framework use the same coordinate system. Its general form is **(row, col)**. What you should notice is that row and col is the abstract coordinate for hexagon, but not the actual location in the window.

一張含有 黃色, 樣式, 對稱, 鮮豔 的圖片

自動產生的描述

**Python Framework**

This documentation provides an overview of the Python framework for Term project, which supports the development of AI algorithms using the Minimax and Monte Carlo Tree Search (MCTS) techniques. This framework is set up for a game played on a hexagonal grid, with specific focus on the logical management of game state rather than graphical representations.

**Framework Description**

**Overview**

The Python framework is designed to facilitate AI gameplay within a hexagonal coordinate system, abstracting the actual window coordinates to focus on the game logic. Each AI move must be computed within a 30-second time limit to ensure game flow and challenge.

**Game Board Initialization**

**\*\*Global Variables\*\***

`hexagon\_board`: A dictionary mapping (row, col) tuples to dictionaries containing hex properties such as coordinates and state.

The abstract coordinates of the hexagons mentioned above are used as keys in the dictionary (e.g., (-5, 0)).

* `x`, `y`: The center coordinates for the hexagon.
* `selected`: Indicates whether the position on the board has been selected.
* `color`: Indicates the color of the piece at this position, which can be black, white, or gray (neutral). If it hasn't been selected, it is None.



**Key functions**

`**draw\_hexagon** (surface, x, y, size, border\_color, fill\_color)`

**Parameters:**

* `surface`: The pygame surface to draw on.
* `x`, `y`: The center coordinates for the hexagon.
* `size`: The radius of the hexagon.
* `border\_color`, `fill\_color`: Colors for the hexagon's border and fill.

**Purpose**: Draws a single hexagon on the specified surface with provided visual characteristics.

`**draw\_hex\_shape\_grid** (surface, center\_row, center\_col, size)`

**Purpose**: Sets up the initial board layout by drawing hexagons in a grid pattern.

**Details**: Uses the `draw\_hexagon` function to place hexagons based on a centered grid system.

`**process\_selections** (x, y)`

**Purpose**: Handles user interactions for selecting hexagons during the game.

**Returns**: A list of hexagons that have been selected based on user input.

` **update\_selected\_hexes** (selected\_hexes, color) `

**Purpose**: When a position is selected, update the information on the board and use the `draw\_hexagon` function to place hexagons.

` **auto\_select\_remaining\_hexes** (color) `

**Purpose**: Randomly select a position from the unselected positions and randomly decide whether to place a neutral piece.

**Implementing Your AI**

To successfully integrate and execute your AI strategies using either Minimax or Monte Carlo Tree Search (MCTS) algorithms within our Python game framework, follow these steps:

1. **You need to do:**

一張含有 文字, 螢幕擷取畫面, 軟體 的圖片

AI 產生的內容可能不正確。

Your AI must return four return values.

`pos\_x, pos\_y`: The abstract coordinates of the hexagons(e.g., (-5, 0)).

`color `: The The color of the stone, can be gray (neutral).

`time\_limit\_exceeded`: Check if the 30-second time limit has been exceeded. If so, use the `auto\_select\_remaining\_hexes` function to randomly place.