FACULTY OF COMPUTERS AND ARTIFICIAL INTELLIGENCE, CAIRO UNIVERSITY

CS213: Programming II
Year 2022-2023
First Semester

Assignment 3 – Version 1.0

Course Instructors:

Dr. Mohammad El-Ramly

TA:

Eng/Mayaa

Prepared by:

Osama Mohamed Mahmoud 20220056

Marawan Shrief Abdelatef Soliman 20220325

Wajih Samir Hussein Ahmed 20220376

Pyramid Class Documentation

For Osama

Overview

The pyramid class implements a game board representing asimplified pyramid structure for a board game. It is a subclass of the Board class and provides specific functionalities for the pyramid-shaped game board.

Class Details:

Class Name: pyramid

Base Class: Board

Board Dimensions: 3 rows by 5 columns

Member Functions pyramid()

Description:

Constructor for the pyramid game board. Parameters: None

Initialization:

Initializes the game board with a pyramid-shaped structure (3 rows by 5 columns).

Allocates memory for the board and sets all cells to empty (' '). Initializes n_moves to zero.

bool update_board(int x, int y, char mark)

Description: Updates the game board with a player's move on the pyramid.

Parameters:

x: Row coordinate.

y: Column coordinate.

mark: Character representation of the player's piece ('X' or 'O').

Returns: Boolean indicating the success of the move update.

Functionality:

Validates the move according to the pyramid's structure and boundaries.

Updates the board with the player's symbol if the move is valid.

Increments the move count.

void display_board()

Description: Displays the current state of the pyramid-shaped game board in the console.

Parameters: None

Functionality:

Renders the current board configuration in a pyramid-like structure.

Shows the position of each cell and its contents ('X', 'O', or empty).

bool is_winner()

Description: Checks if there's a winner in the pyramid game.

Parameters: None

Returns: Boolean indicating if a player has won.

Functionality:

Checks for winning conditions according to the specific pyramid-shaped game rules.

Determines if the game has reached a winning state.

bool is_draw()

Description: Checks if the game has ended in a draw.

Parameters: None

Returns: Boolean indicating a draw.

Functionality:

Checks if the game has reached a draw state based on the move count and winning conditions.

bool game_is_over()

Description: Checks if the game has concluded.

Parameters: None

Returns: Boolean indicating the end of the game.

Functionality:

Determines if the game has ended by reaching the maximum allowed moves.

Usage

Create an instance of the pyramid class.

Use member functions like update_board() to make moves, display_board() to show the board, and is_winner() / is_draw() to check game states.

Four-in-a-Row Game Documentation

For Marawan

Overview

The Four-in-a-Row game, also known as Connect Four, is a two-player board game where the players take turns dropping colored discs into a vertical grid. The objective is to connect four of one's own discs vertically, horizontally, or diagonally before the opponent does.

Features

The game provides a grid-based interface for two players to take turns placing their pieces.

Win condition detection to identify when a player has connected four pieces in a row.

Draw condition handling to recognize when the game reaches a stalemate.

Player input validation for placing pieces within the grid boundaries.

Clear display of the game board for user interaction.

Classes and Functions

Four_in_a_row Class

Attributes:

n_rows: Number of rows in the game board.

n_cols: Number of columns in the game board.

n_moves: Counter for the number of moves made.

board: 2D array to represent the game board.

Public Functions:

Four_in_a_row(): Constructor for initializing the game board.

bool update_board(int x, int y, char mark): Updates the game board with a player's move.

void display_board(): Displays the current state of the game board.

bool is_winner(): Checks if a player has won the game.

bool is_draw(): Checks if the game is drawn due to no more valid moves.

bool game_is_over(): Checks if the game has concluded (either win or draw).

Function Details

Four_in_a_row(): Initializes the game board by allocating memory for the grid and setting initial values.

bool update_board(int x, int y, char mark): Allows a player to place their piece on the board. Takes input parameters x and y for the coordinates and mark to represent the player's piece.

void display_board(): Renders the current state of the game board in the console, providing a visual representation of the grid with player pieces.

bool is_winner(): Checks for the win condition by examining the board for four consecutive pieces in a row, column, or diagonal.

bool is_draw(): Checks if the game has ended in a draw by verifying if no more valid moves can be made.

bool game_is_over(): Determines if the game has reached its conclusion by either a win or a draw condition.

Usage

Initialization:

Create an instance of the Four_in_a_row class to initialize the game board.

Gameplay:

Use update_board() to make moves by providing valid coordinates.

Continuously check is_winner() and is_draw() to determine game status.

Use game_is_over() to identify when the game has ended.

Display:

Regularly call display_board() to show the current state of the game.

Sure, here's a basic template for documenting the FiveByFiveBoard class along with its functions:

FiveByFiveBoard Class Documentation

For Waggeh

Overview

The FiveByFiveBoard class is designed to create a game board for a variant of the Tic-Tac-Toe game played on a 5x5 grid. It inherits from the Board class and implements specific functionalities for the game's board and rules.

Class Details

Class Name: FiveByFiveBoard

Base Class: Board

Grid Size: 5x5

Member Functions

FiveByFiveBoard()

Description: Constructor for the game board.

Parameters: None

Initialization:

Initializes the game board with a 5x5 grid.

Allocates memory for the board and sets all cells to empty (' ').

Initializes n_moves, count_X, and count_O to zero.

bool update_board(int x, int y, char symbol)

Description: Updates the game board with a player's move.

Parameters:

x: Row coordinate.

y: Column coordinate.

symbol: Character representation of the player's piece ('X' or 'O').

Returns: Boolean value indicating the success of the move update.

Functionality:

Checks if the provided coordinates are within the board's boundaries and if the selected cell is empty.

Updates the board with the player's symbol if the move is valid.

Increments the move count.

void display_board()

Description: Displays the current state of the game board in the console.

Parameters: None

Functionality:

Renders the current board configuration in a grid format.

Shows the position of each cell and its contents ('X', 'O', or empty).

bool is_winner()

Description: Checks if there's a winner in the game.

Parameters: None

Returns: Boolean indicating if a player has won.

Functionality:

Checks for winning conditions horizontally, vertically, and diagonally.

Updates count_X and count_O to keep track of the number of 'X' and 'O' sequences.

Determines if the game has reached a winning state.

bool is_draw()

Description: Checks if the game has ended in a draw.

Parameters: None

Returns: Boolean indicating a draw.

Functionality:

Checks if the game has reached a draw state based on the move count and the number of 'X' and 'O' sequences.

bool game_is_over()

Description: Checks if the game has concluded.

Parameters: None

Returns: Boolean indicating the end of the game.

Functionality:

Determines if the game has ended by reaching the maximum allowed moves (24 in this case).

Usage

Create an instance of the FiveByFiveBoard class.

Use member functions like update_board() to make moves, display_board() to show the board, and is_winner() / is_draw() to check game states.