## Team members:

| | |
|---|---|
| 1 – Ahmed Tamer Fathy | ID: 20220013 |
| 2 – Mohamed Mostafa | ID: 20220309 |
| 3 – Youssef Nasser Mohamed | ID: 20220416 |

Application description:

It is an app that contains 4-different games that the user can play with another player, a random computer or with a trained AI.

## Board functions description:

**1 - Pyramic board:**

**Pyramid_X_O_Board::Pyramid_X_O_Board():**

Description: Constructor for the Pyramid_X_O_Board class.

Explanation: Initializes the game board, sets the number of rows (n_rows) to 3, and columns (n_cols) to 5. Initializes the board with specific cells marked as allowed (0) based on the pyramid shape.

**bool Pyramid_X_O_Board::update_board(int x, int y, char mark):**

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

Explanation: Takes the coordinates (x, y) and the player's mark (X or O). Checks if the move is valid and updates the board if the selected cell is allowed and not already marked.

**bool Pyramid_X_O_Board::undo_move(int x, int y):**

Description: Undoes a player's move.

Explanation: Takes the coordinates (x, y) and reverts the board to its previous state by marking the specified cell as allowed (0). Decrements the total number of moves.

**void Pyramid_X_O_Board::display_board():**

Description: Displays the current state of the game board.

Explanation: Prints the game board to the console, showing the coordinates of each cell and the marks (X, O, or space if the cell is not allowed).

**bool Pyramid_X_O_Board::is_winner():**

Description: Checks if there is a winner.

Explanation: Hard-coded logic to check for winning conditions, including diagonal, row, and column wins.

**bool Pyramid_X_O_Board::is_draw():**

Description: Checks if the game is a draw.

Explanation: Returns true if the total number of moves is 9 and there is no winner, indicating a draw.

**bool Pyramid_X_O_Board::game_is_over():**

Description: Checks if the game is over.

Explanation: Returns true if the total number of moves is greater than or equal to 9.

**int Pyramid_X_O_Board::get_n_moves() const:**

Description: Gets the current number of moves.

Explanation: Returns the total number of moves made in the game.

**string Pyramid_X_O_Board::get_board() const:**

Description: Gets the string representation of the game board.

Explanation: Returns a string representing the current state of the game board.

**int Pyramid_X_O_Board::get_n_rows() const:**

Description: Gets the number of rows in the game board.

Explanation: Returns the number of rows in the game board.

**int Pyramid_X_O_Board::get_n_cols() const:**

Description: Gets the number of columns in the game board.

Explanation: Returns the number of columns in the game board.

**2 - Connect-4 board:**

**CollectFourBoard::CollectFourBoard():**

Description: Constructor for the CollectFourBoard class.

Explanation: Initializes the game board with 6 rows and 7 columns, setting each cell to 0 (empty).

**bool CollectFourBoard::update_board(int x, int y, char mark):**

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

Explanation: Checks if the move is valid, considering if the selected column is full. If valid, updates the board with the player's mark and increments the move counter.

**void CollectFourBoard::display_board():**

Description: Displays the current state of the game board.

Explanation: Prints the game board to the console, showing the coordinates of each cell and the marks (X, O, or empty if the cell is not filled).

**bool CollectFourBoard::is_winner():**

Description: Checks if there is a winner.

Explanation: Checks for four consecutive marks horizontally, vertically, and diagonally from the last played position.

**bool CollectFourBoard::is_draw():**

Description: Checks if the game is a draw.

Explanation: Returns true if the total number of moves is 42 (board is full) and there is no winner.

**bool CollectFourBoard::game_is_over():**

Description: Checks if the game is over.

Explanation: Returns true if the total number of moves is greater than or equal to 42.

**int CollectFourBoard::getMoves():**

Description: Gets the current number of moves.

Explanation: Returns the total number of moves made in the game.

**bool CollectFourBoard::undo_move(int x, int y):**

Description: Undoes a player's move.

Explanation: Takes the coordinates (x, y) and reverts the board to its previous state by marking the specified cell as empty. Decrements the total number of moves.

**int CollectFourBoard::get_n_moves() const:**

Description: Gets the current number of moves.

Explanation: Returns the total number of moves made in the game.

**int CollectFourBoard::get_n_rows() const:**

Description: Gets the number of rows in the game board.

Explanation: Returns the number of rows in the game board.

**int CollectFourBoard::get_n_cols() const:**

Description: Gets the number of columns in the game board.

Explanation: Returns the number of columns in the game board.

**string CollectFourBoard::get_board() const:**

Description: Gets the string representation of the game board.

Explanation: Returns a string representing the current state of the game board.

**3 - 5x5 X-O board:**

**specialBoard::specialBoard():**

Description: Constructor for the specialBoard class.

Explanation: Initializes the game board with 5 rows and 5 columns, setting each cell to 0 (empty).

**bool specialBoard::update_board(int x, int y, char mark):**

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

Explanation: Checks if the move is valid and the selected cell is empty. If valid, updates the board with the player's mark. If 24 moves have been made, automatically fills the remaining empty cells with 'X'.

**void specialBoard::display_board():**

Description: Displays the current state of the game board.

Explanation: Prints the game board to the console, showing the coordinates of each cell and the marks ('X', 'O', or empty if the cell is not filled).

**bool specialBoard::is_winner():**

Description: Checks if there is a winner.

Explanation: Checks for horizontal, vertical, and diagonal wins by counting consecutive marks ('X' or 'O') in rows, columns, and diagonals.

**bool specialBoard::is_draw():**

Description: Checks if the game is a draw.

Explanation: Returns true if the number of 'X' marks is equal to the number of 'O' marks, indicating a draw.

**bool specialBoard::game_is_over():**

Description: Checks if the game is over.

Explanation: Returns true if 25 moves have been made, indicating that the game is over.

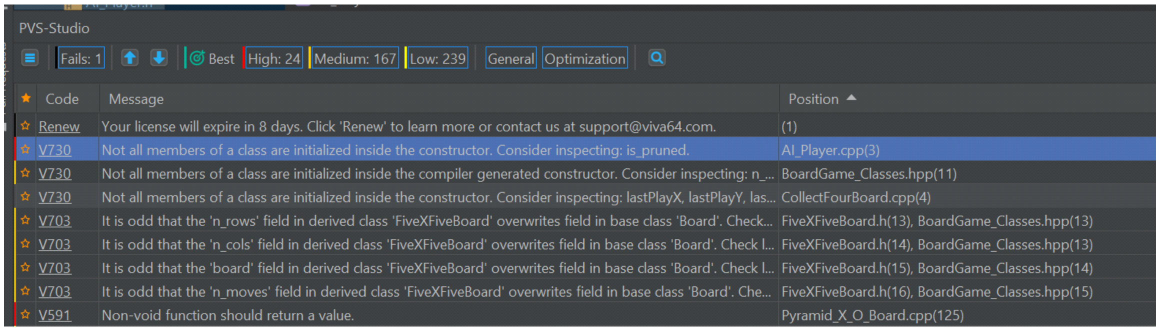
**Eval_game functions:**

used for the AI to be able to determine if the current position of the board is a winning position for it or a losing position so it can act in the most optimal way possible.

## Breakdown table of who did what:

| Mohamed Mostafa 20220309 | Youssef Nasser Mohamed 20220416 | Ahmed Tamer Fathy 20220013 |
|---|---|---|
| 1- Connect-4 Game<br>2- AI for connect 4 game<br>3- GUI work<br>4- Pvs Analysis | 1- 5x5 tic-tac-toe game<br>2- AI for the game<br>3- GUI work<br>4- Pdf and function breakdown | 1- Pyramic tic-tac-toe<br>2- AI for the game<br>3- Integration of three games code in one app<br>4- GUI work |

## Static code analysis and Code Quality:


PVS-Studio

Fails: 1 | Best | High: 24 | Medium: 167 | Low: 239 | General | Optimization

| Code | Message | Position |
|---|---|---|
| Renew | Your license will expire in 8 days. Click 'Renew' to learn more or contact us at support@viva64.com. | (1) |
| V730 | Not all members of a class are initialized inside the constructor. Consider inspecting: is_pruned. | AI_Player.cpp(3) |
| V730 | Not all members of a class are initialized inside the compiler generated constructor. Consider inspecting: n_... | BoardGame_Classes.hpp(11) |
| V730 | Not all members of a class are initialized inside the constructor. Consider inspecting: lastPlayX, lastPlayY, las... | CollectFourBoard.cpp(4) |
| V703 | It is odd that the 'n_rows' field in derived class 'FiveXFiveBoard' overwrites field in base class 'Board'. Check... | FiveXFiveBoard.h(13), BoardGame_Classes.hpp(13) |
| V703 | It is odd that the 'n_cols' field in derived class 'FiveXFiveBoard' overwrites field in base class 'Board'. Check l... | FiveXFiveBoard.h(14), BoardGame_Classes.hpp(13) |
| V703 | It is odd that the 'board' field in derived class 'FiveXFiveBoard' overwrites field in base class 'Board'. Check l... | FiveXFiveBoard.h(15), BoardGame_Classes.hpp(14) |
| V703 | It is odd that the 'n_moves' field in derived class 'FiveXFiveBoard' overwrites field in base class 'Board'. Che... | FiveXFiveBoard.h(16), BoardGame_Classes.hpp(15) |
| V591 | Non-void function should return a value. | Pyramid_X_O_Board.cpp(125) |

Most of the errors are positive errors in the built-in c++ libraries and cmake.

Major errors that we found in our code were (Non-void function that did not return a value in some cases (AI_Player.cpp(3)), Not all of class members are initialized in the constructor (Pyramid_X_O_Board.cpp(125)).

You can see of the errors in the above image.

This tool is very useful since it helped us know where some of the problems are so it helped us solve it in a lot more efficient way.