FINAL PROJECT DOCUMENTATION

3D Tic-Tac-Toe

Group 3

Jared Tamulynas

Samba Diagne

Tarun Chopra

Mrinalini Chava

ITCS 6150 - Intelligent Systems

University of North Carolina Charlotte

Project Description

The objective of this project is to develop a one-player 3D Tic Tac Toe game with an AI opponent. The game features a 4x4x4 grid, resulting in 64 slots for gameplay. The user interface is inspired by traditional Tic Tac Toe interfaces but adapted for 3D gameplay. The user can select from three difficulty levels including the following:

- Easy means that the procedure is 2 levels deep.
- Difficult means that the procedure is 4 levels deep.
- Insane means that the procedure is 6 levels deep.

The AI employs the Alpha-Beta Pruning algorithm, a refined version of the Minimax algorithm. Alpha-Beta Pruning works by evaluating the possible future moves and countermoves, aiming to minimize the opponent's chances of winning while maximizing its own. Alpha-Beta Pruning enhances efficiency by eliminating the need to examine each node in the game's decision tree by pruning away branches that cannot influence the final decision.

The link to our GitHub repository is included below:
GitHub Repo

System Flow Diagram

Our program is organized into five distinct Python files, each serving a specific role in the overall system. A summary of each file and its function is included below:

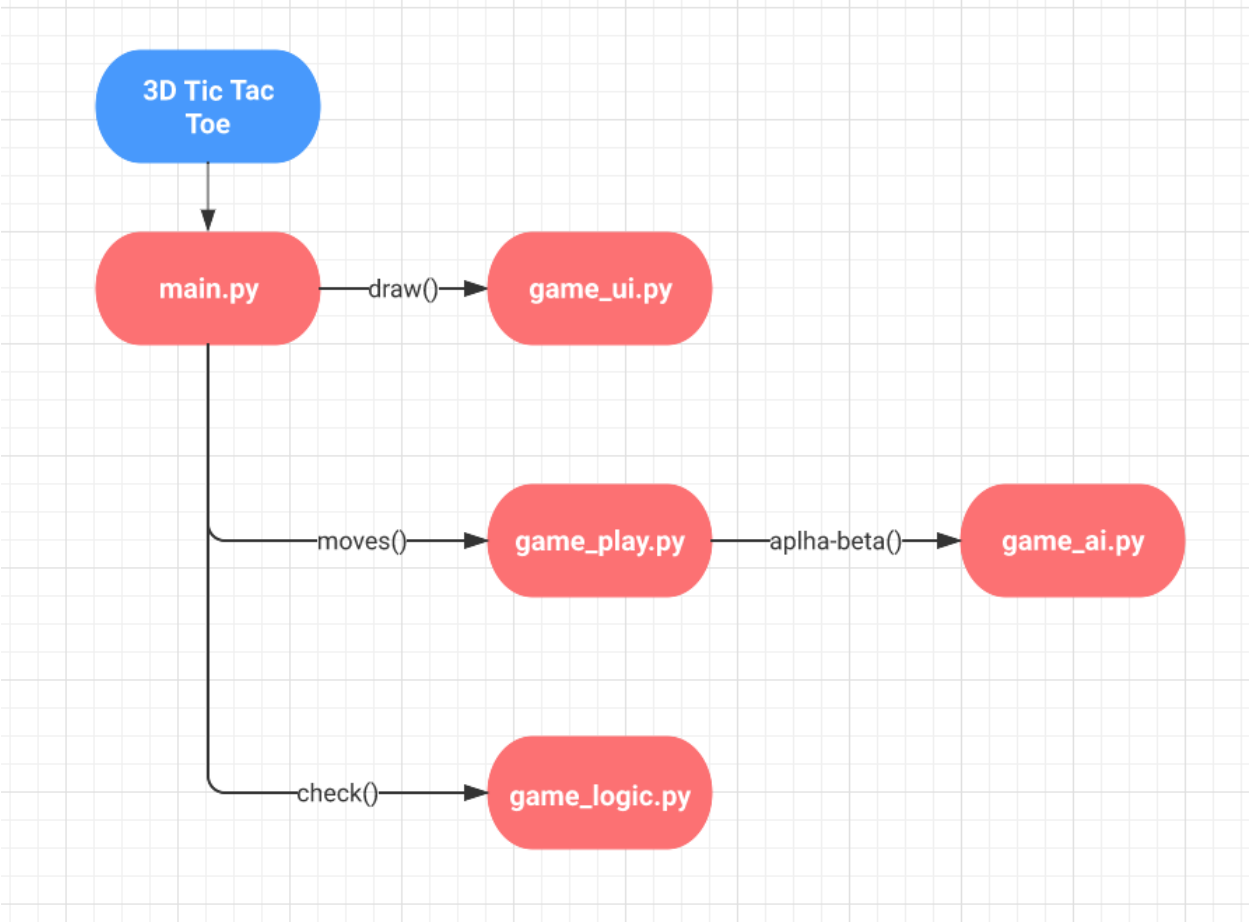
main.py
The entry point of the application. This file initiates the game combining all other modules (AI, logic, gameplay, and UI) and starting the game loop. It integrates all components into a cohesive application.

game_logic.py
The backbone of the game, managing the fundamental rules and mechanics. It includes functions to check for win conditions, maintain the state of the game board, and manage turn sequences. Ensures that the game adheres to the standard Tic Tac Toe rules and logic.

game_ui.py
This file focuses on the user interface. This module is responsible for rendering the game board, displaying the game status, and updating the UI based on game events.

game_ai.py
This module contains the core AI logic for the game. It implements the Alpha-Beta Pruning algorithm, handling the AI's decision-making process for moves. It acts as the brain of the AI opponent, determining its strategy and responses during the game.

game_play.py
This file bridges the game's logic with user interactions. It processes player inputs, communicates with the game logic and AI modules. It serves as the intermediary between the player and the game logic.



The diagram above shows how main.py is the main entry into the program. Main.py makes direct calls to game_ui.py, game_play.py, and game_logic.py during the game loop. The game_play.py file manages the calls to game_ai.py as shown in the diagram above. By separating files this way, we maintained an organized project that allowed us to easily update and improve our user interface throughout development.

Data Flow Diagram

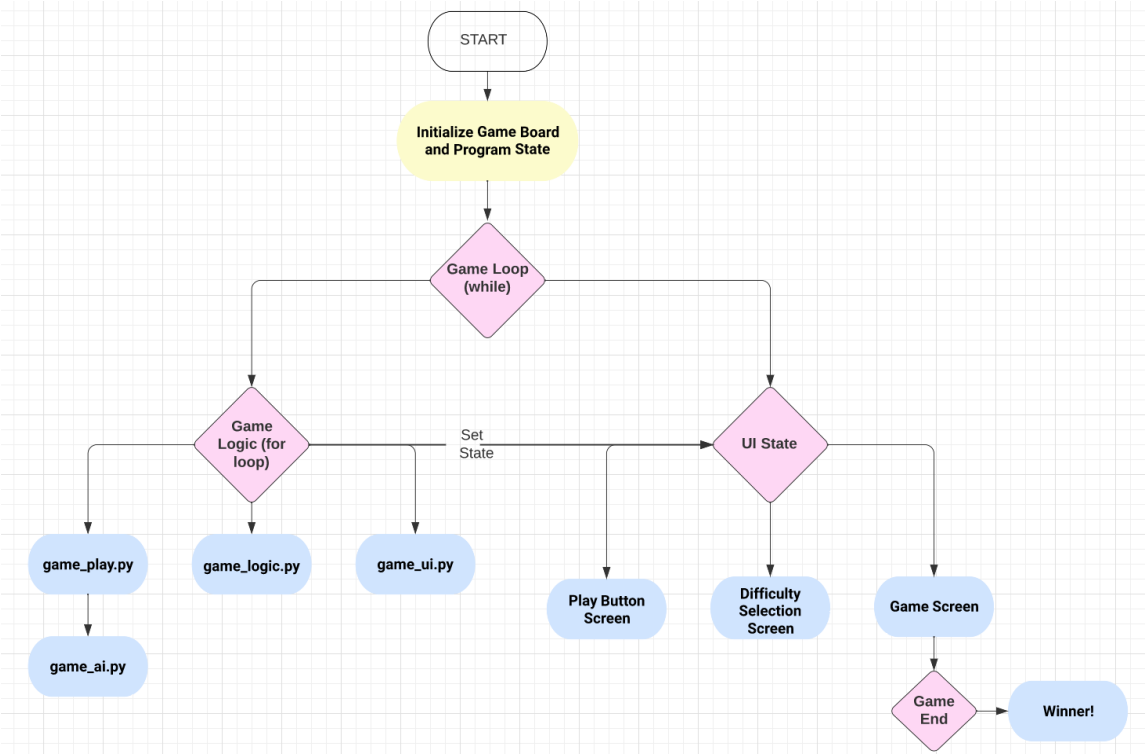
The main.py file manages state changes and interactions among different components. The main.py file acts as the coordinator for the game, managing the flow of data and state changes between various files: game_logic.py, game_ai.py, game_play.py, and game_ui.py.

Initial Game Setup
The game is initialized with main.py calling game_ui.py to draw the required components and with main.py setting the program's initial state.

Gameplay
During the player's turn, main.py uses game_play.py to process player input (like choosing a grid slot). The player's move is sent to game_logic.py to update the game state and check for win conditions. After the player's turn, main.py activates the AI's turn, using Alpha-Beta Pruning to pick the AI's move. After each turn, main.py directs game_ui.py to update the board visually to reflect new moves.

The main.py file checks the game's state through game_logic.py for outcomes like win, loss, or draw. It provides player feedback via UI, showing AI moves and end-game messages.

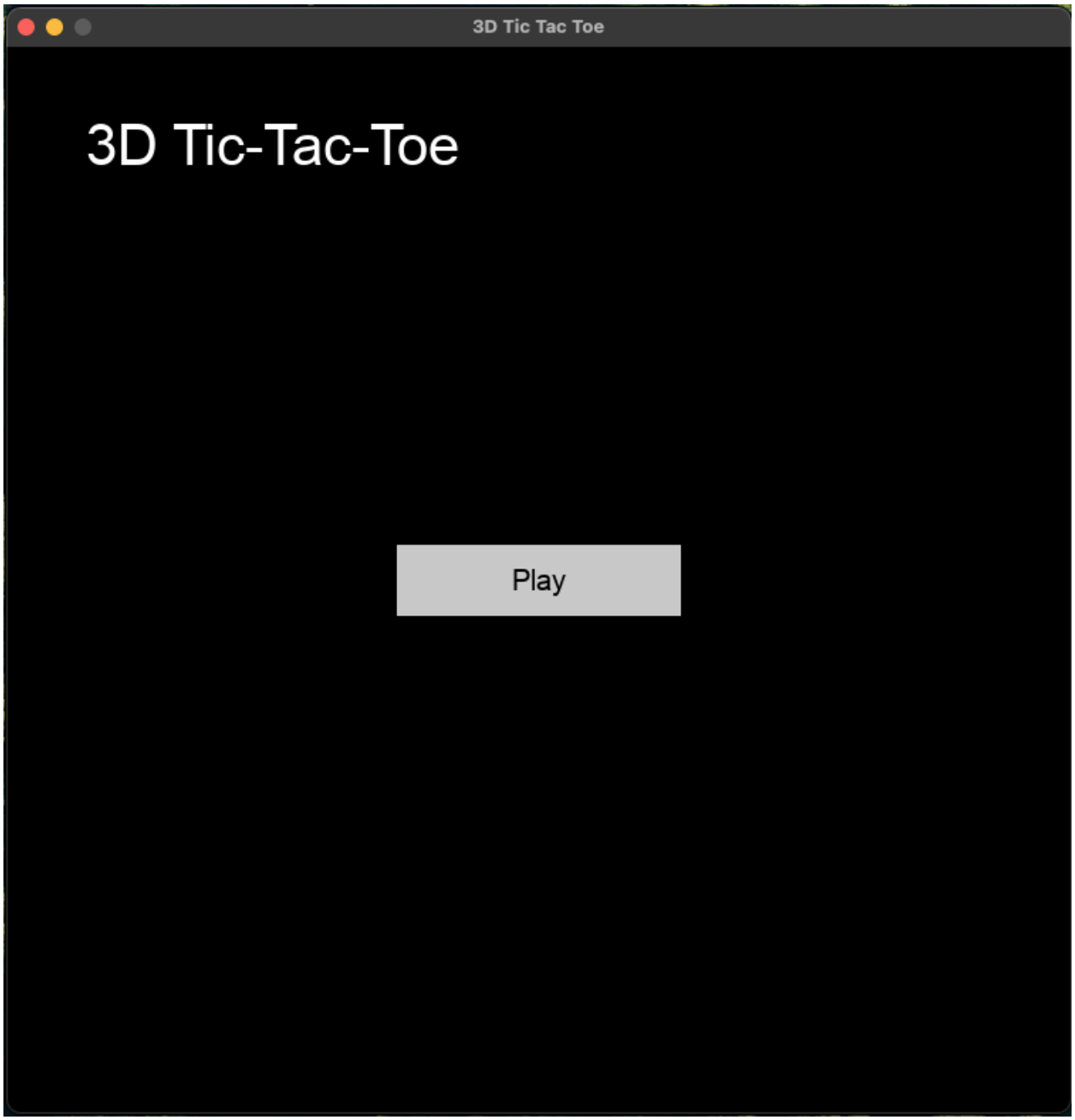
End Game
After a winner is determined, main.py instructs game_ui.py to show the relevant end-game message. It then offers a reset option, reinitializing all components for a new game.



Input Design

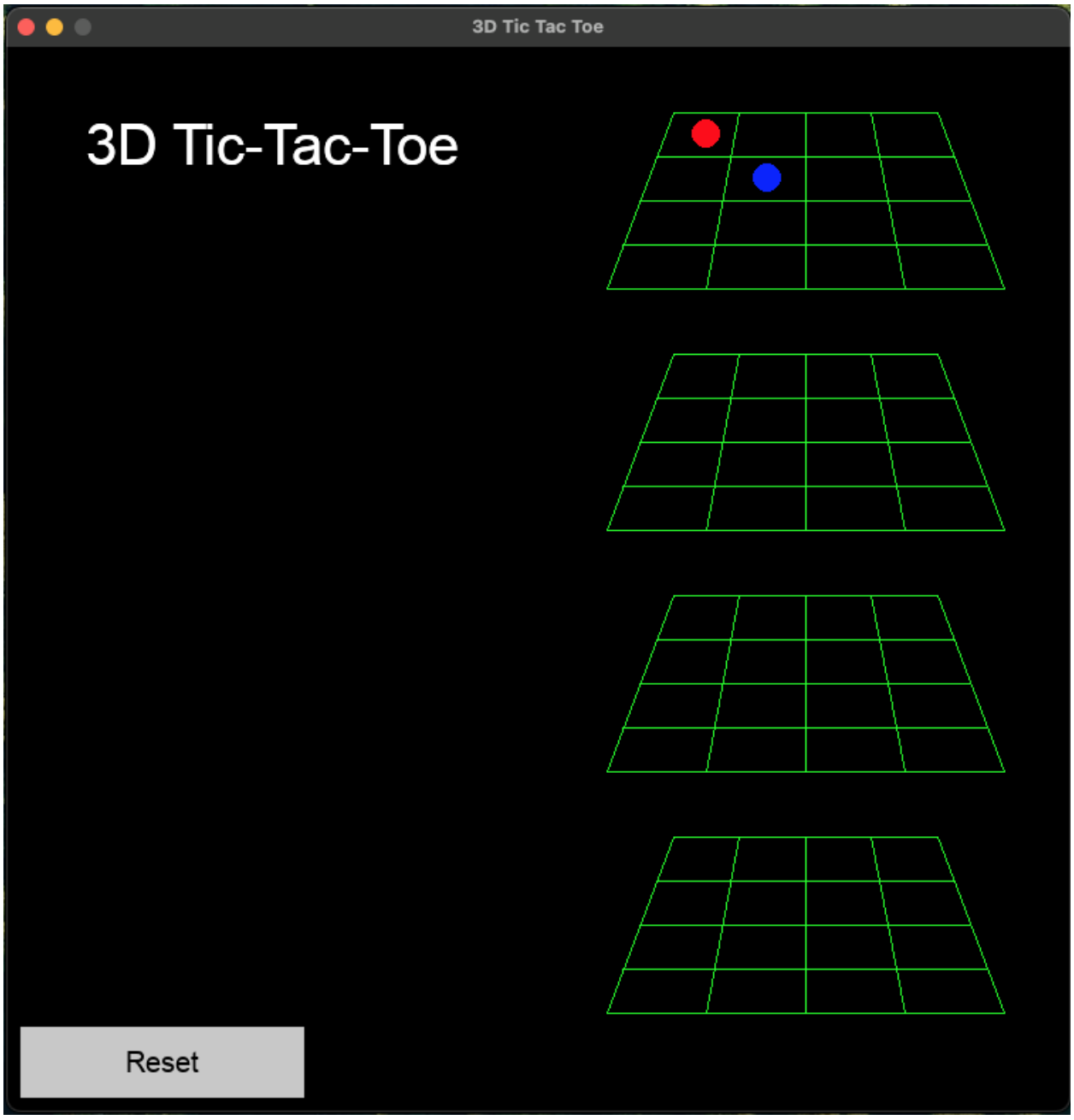
The input design of our project is centered around simplicity and intuitiveness, ensuring that players can easily navigate and interact with the game. Below are the key input design components:

Start Game Screen
A play button is placed in the center of the game's initial screen. This button is designed to be easily noticeable, inviting the player to start the game.



Select Difficulty Screen
There are three distinct buttons, each representing a different difficulty level. These buttons are clearly labeled and will provide feedback when pressed.

Game Grid Interaction
The game grid is presented as four stacked Tic Tac Toe boards, creating a 3D effect. This design makes it visually clear and understandable, even for players new to 3D Tic Tac Toe. Players can select a square on any level of the grid with a simple click. The clickable areas are designed to be large enough to avoid misclicks, ensuring a smooth game experience. Each square, when selected will show the player or ai move as a filled in circle.



The design is straightforward, minimizing the learning curve for new players.

Output Design

The output design of the 3D Tic Tac Toe game focuses on how information is visually communicated to the player. This section addresses the specific elements of output design.

The game grid is composed of 64 slots arranged in a 4x4x4 matrix. Each slot will display either a red or blue circle marker or remain empty, representing player moves, AI moves, and available moves respectively. Player moves result in instant visual feedback.

The end of the game triggers a clear alert, such as "You win - Congrats!". This output design aims to make the game intuitive, ensuring that players of all levels can enjoy.