Name: Buys Enock Onkarabile

Student No:219013044

Module: Artificial Intelligence **Deliverable 1**

University of Johannesburg

Academy of Computer Science and Software Engineering

Table of Contents

1. Background and problem statement

The box and Dot game is a strategic turn-based game for two or more players where players take turns drawing lines to complete boxes from dots drawn on a paper, which was released in the 19th century by Édouard Lucas(gametable 2019). The player with more complete boxes at the end is the winner.

There has been a promise in training intelligent agents for being strategic when making decisions to learn without making complicated lines of code, this has been achieved through a subset of machine learning, which is Reinforcement Learning. We have huge examples of games like AlphaGo Zero, which learned to play at a superhuman level through learning on its own through RL(Silver, Schrittwieser et al. 2017).

Now, my aim is to develop a smart game-playing agent that can make decisions through Reinforcement Learning in the box-and-dot game. Game-playing agents often struggle with sample efficiency, reward sparsity in turn-based strategy games like box-and-dot, and generalization to new states, which limits their effectiveness.

2. Classification of an Agent

This agent is a single agent playing against a human opponent, making decisions based on reinforcement learning principles.It learns by observing the game state,making strategic decisions using RL Model,Receiving rewards when it completes a box and updates its strategy based on past games outcomes to make sure it improves over time.

2.1 Agent Task Environment Properties

Below are properties that describe the box and dots environment:

a) Fully observable: The game board is visible to the agent
b) Multi Agent: The game involves an opponent
c) Stochastic: The opponents' actions introduce an element of unpredictability (Schrittwieser, Antonoglou et al. 2020).
d) Sequential: the outcome of the game is determined by a sequence of decisions over time.
e) Semi-dynamic: the environment doesn't change unless players make moves
f) Discrete: There is a finite number of possible actions
g) Known: the rules and mechanics are fully known to the agent

2.2 PAES Analysis

The table below shows the description of the task environment:

Agent Type	**Performance Measure**	**Environment**	**Actuators**	**sensors**
Goal-driven, model-based RL gent	Evaluated based on win rate, decision efficiency, and learning rate	The box and dot game environment (virtual grid-based board)	The agent makes moves by drawing lines between dots	The agent perceives the board state, opponent's moves and available actions

3. Use Cases and Use Case Diagrams

3.1 Use Case diagrams

There are 4 defined subsystem from how the game starts and ends:

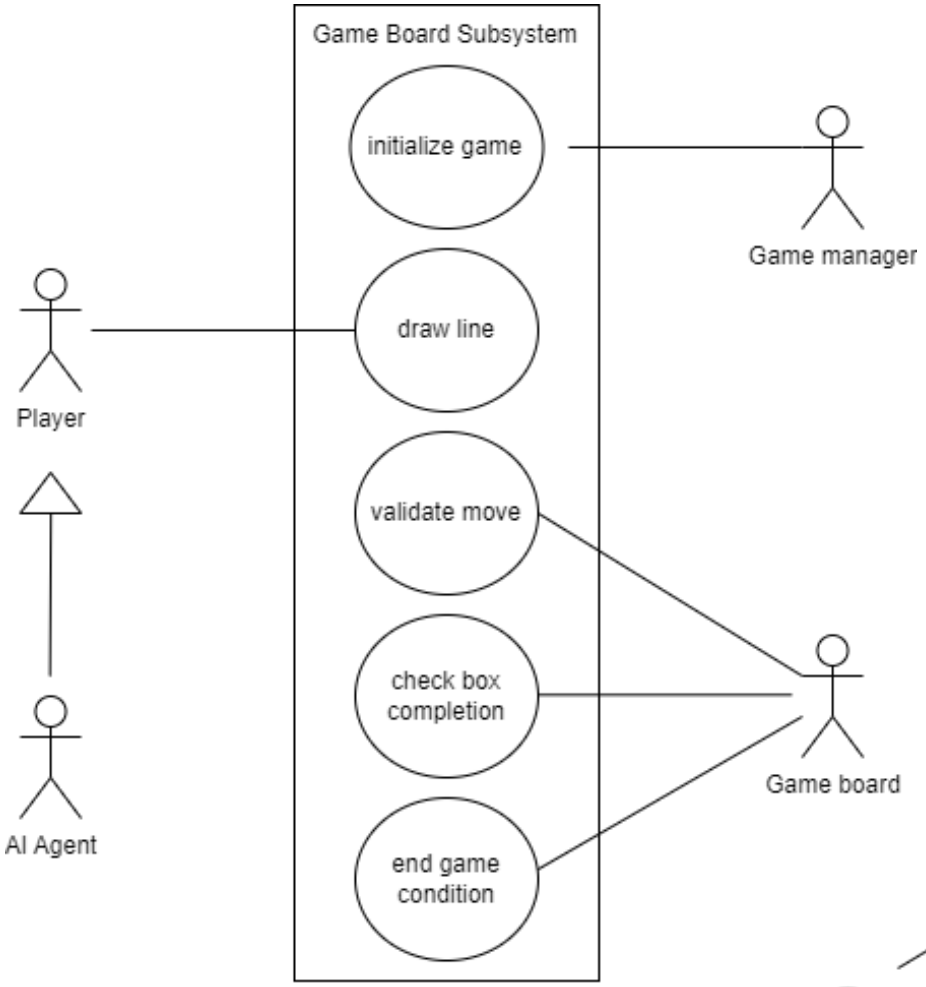

Figure 1.1: Game board subsystem

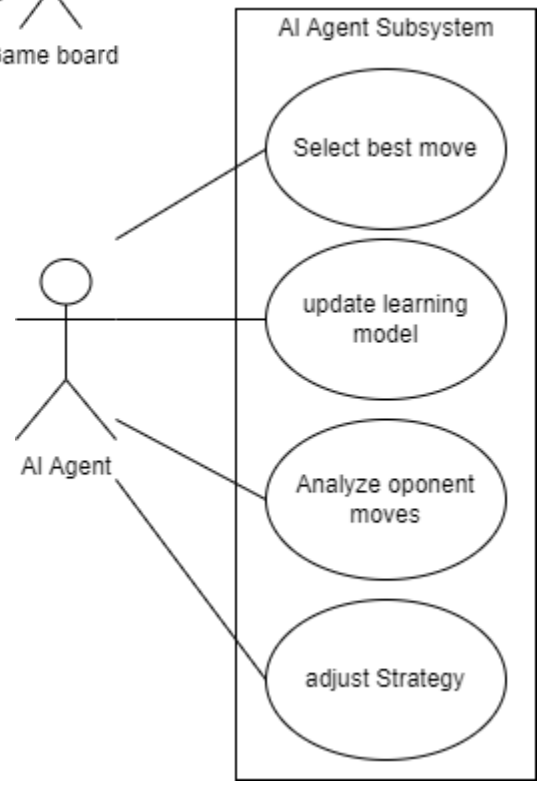

Figure 1.2: AI Agent subsystem

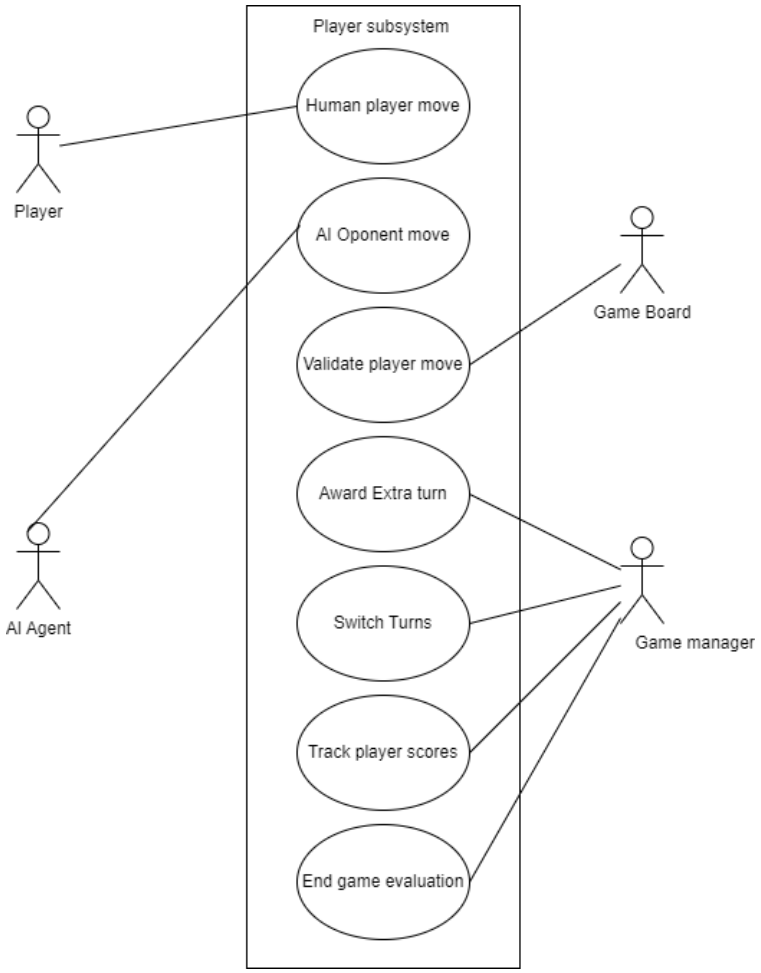

Figure 1.3: Player subsystem

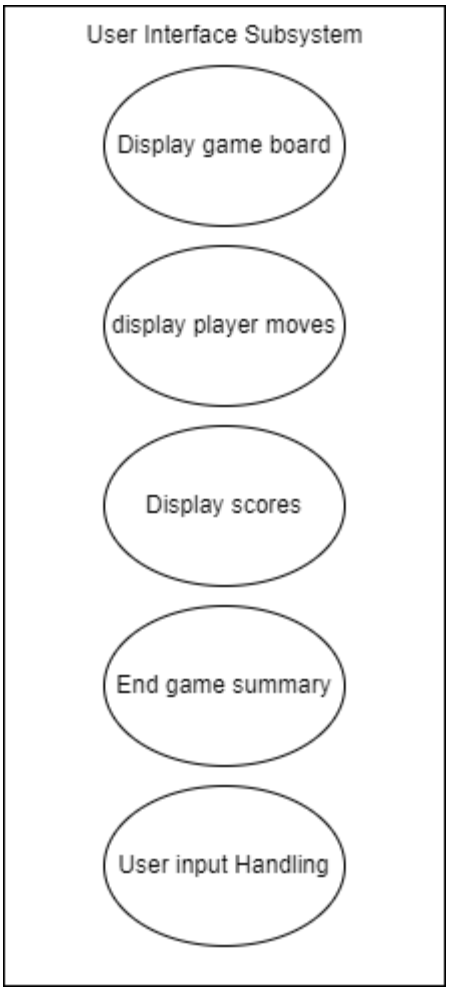

Figure 1.4: User interface subsystem

4 Use Case Descriptions

Use Case	Actor	Subsystem	Action
Initialize Game Board	Game Manager	Game Board	Sets up the grid, initializes available moves, and resets all values to start a new game.
Draw Line	Player / AI Agent	Game Board S	Updates the board when a player selects a valid move and visually reflects changes.
Validate Move	Game Board	Game Board	Ensures selected move follows the game rules and does not overlap with an existing line.
Check Box Completion	Game Board	Game Board	Determines if a completed box should be awarded and updates scores accordingly.
End Game Condition	Game Board	Game Board	Detects when all moves are made and triggers the end game

Select Best Move	AI Agent	AI Agent	Uses reinforcement learning to choose the best move based on past experience and learned strategies.
Update Learning Model	AI Agent	AI Agent	After game completion, updates its internal policy based on rewards from the game.
Analyze Opponent Moves	AI Agent	AI Agent	Evaluates the human player's move patterns to improve decision-making in future games.
Adjust Strategy	AI Agent	AI Agent	Adapts its approach dynamically to counter the human player's strategies.
Human Player Move	Human Player	Player	The human player selects a valid move and submits it. The system validates and updates the board.
AI Opponent Move	AI Agent	Player	The AI opponent calculates the best move using reinforcement learning and executes it.
Validate Player Move	Game Board	Player	Ensures the chosen move follows game rules and is not invalid.
Award Extra Turn	Game Manager	Player	Grants an additional turn if the move completed a box.
Switch Turns	Game Manager	Player Subsystem	Alternates turns between the human player and the AI.
Track Player Scores	Game Manager	Player	Maintains a record of player points based on completed boxes.
End Game Evaluation	Game Manager	Player Subsystem	Determines the final winner and displays the result.
Display Game Board	UI System	User Interface	Renders the grid, lines, and boxes in real-time.
Show Player Moves	UI System	User Interface	Displays the current move on the board when selected by a player or AI.
Display Scores	UI System	User Interface	Updates and shows the current scores of both players.
End Game Summary	UI System	User Interface	Shows the final results, including the winner and performance statistics.
User Input Handling	UI System	User Interface	Captures player interactions such as clicking on a move and submitting

5 Class Diagram

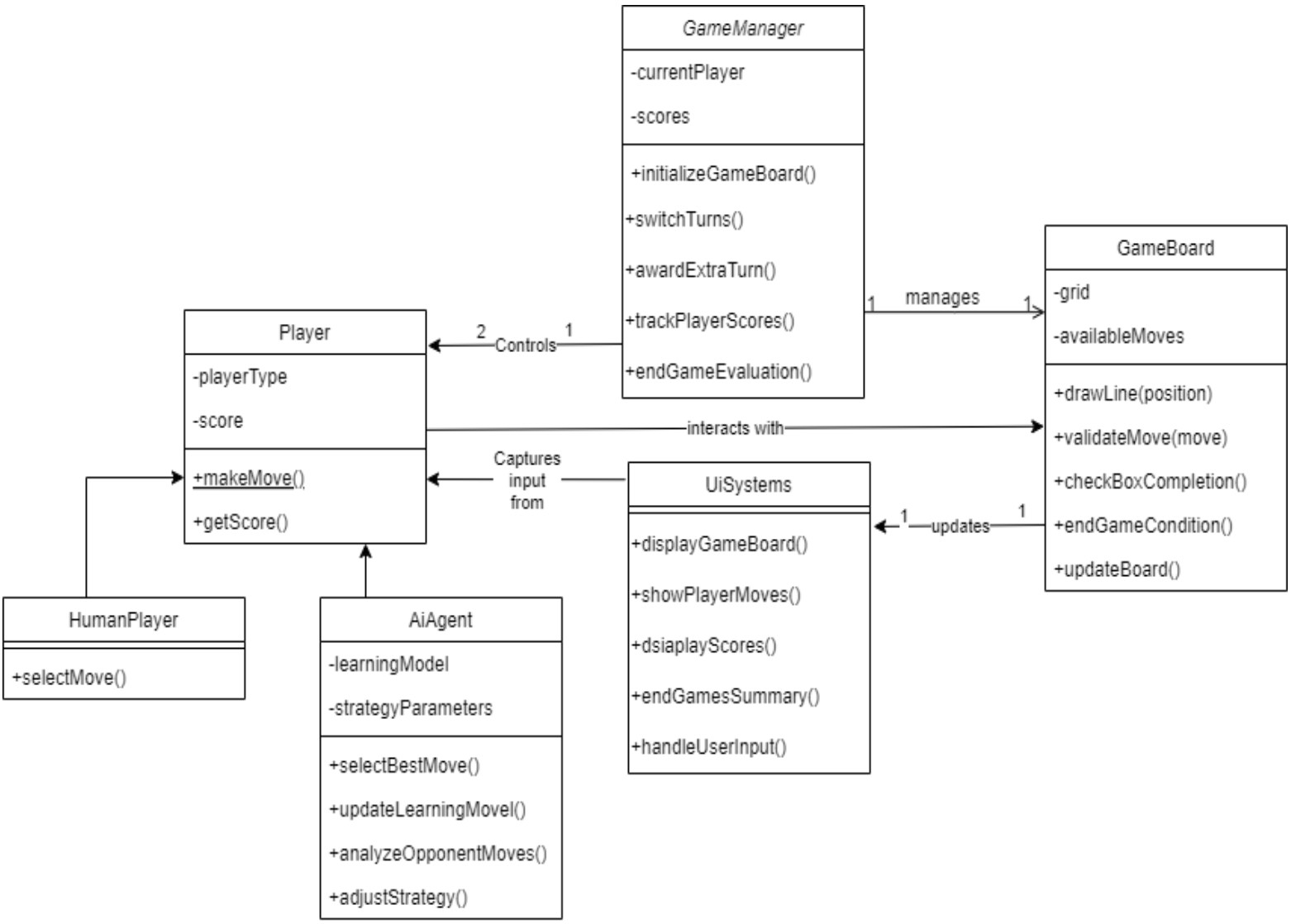

Figure 2: Class Diagrams

6 Interaction sequence Diagrams

Game initialization Sequence

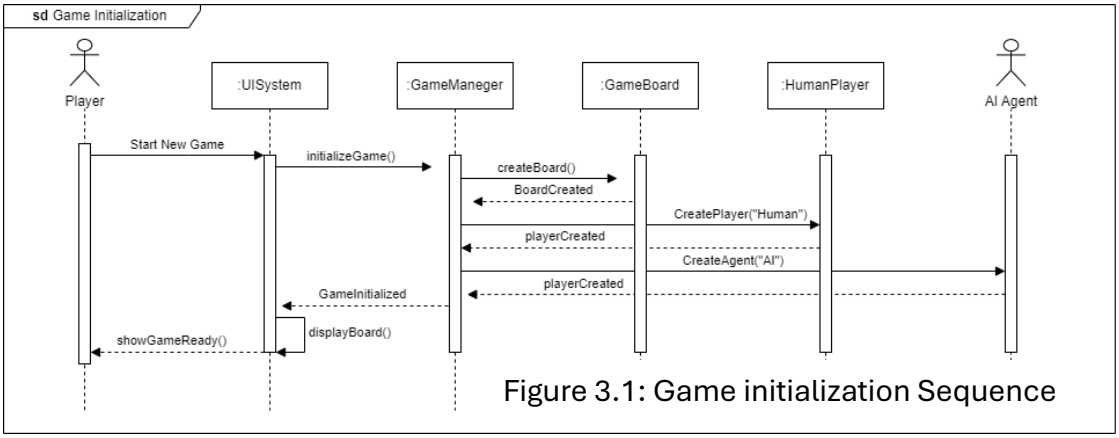

Figure 3.1: Game initialization Sequence

Player Turn Sequence

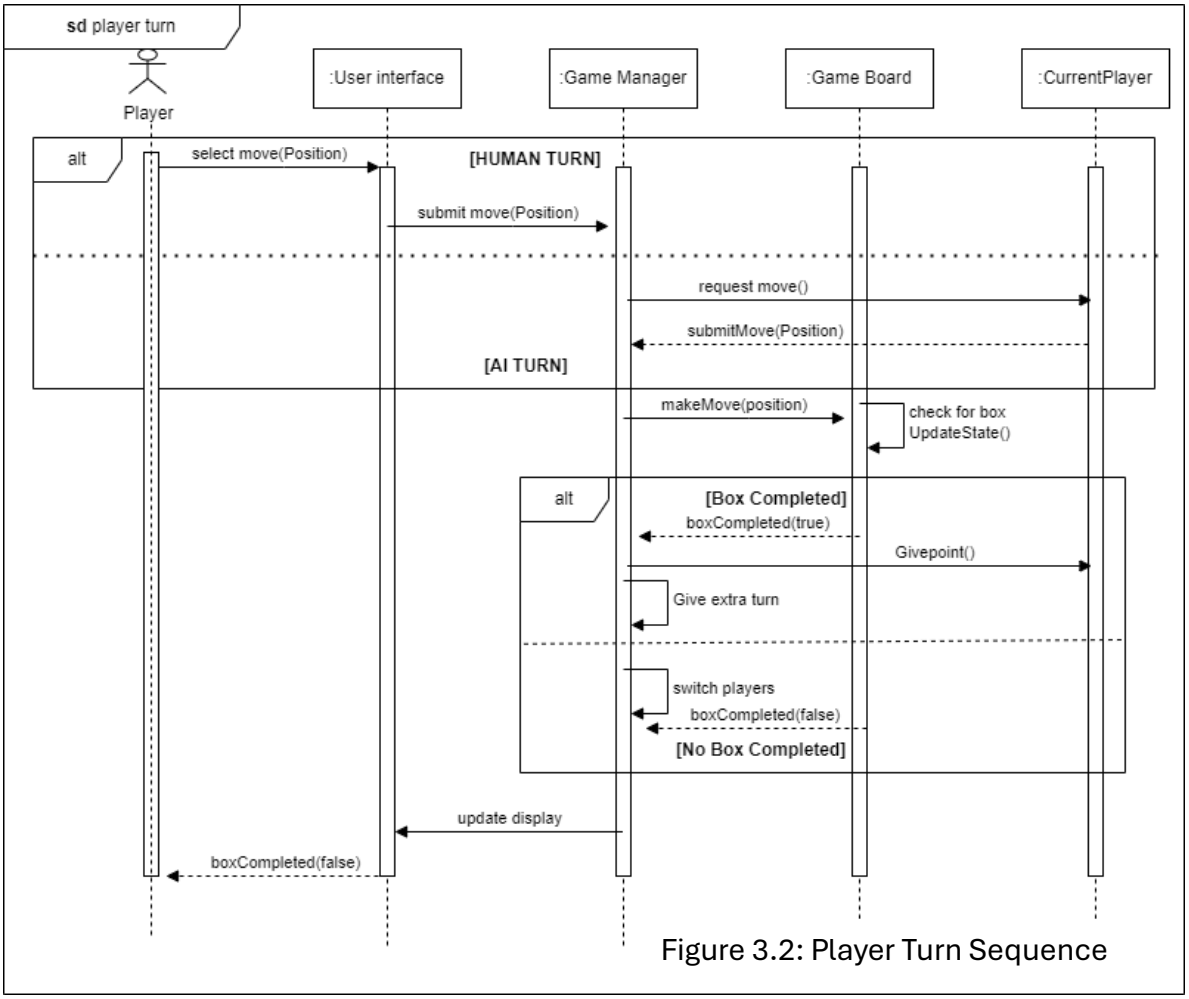

Figure 3.2: Player Turn Sequence

Player Turn Switching Sequence

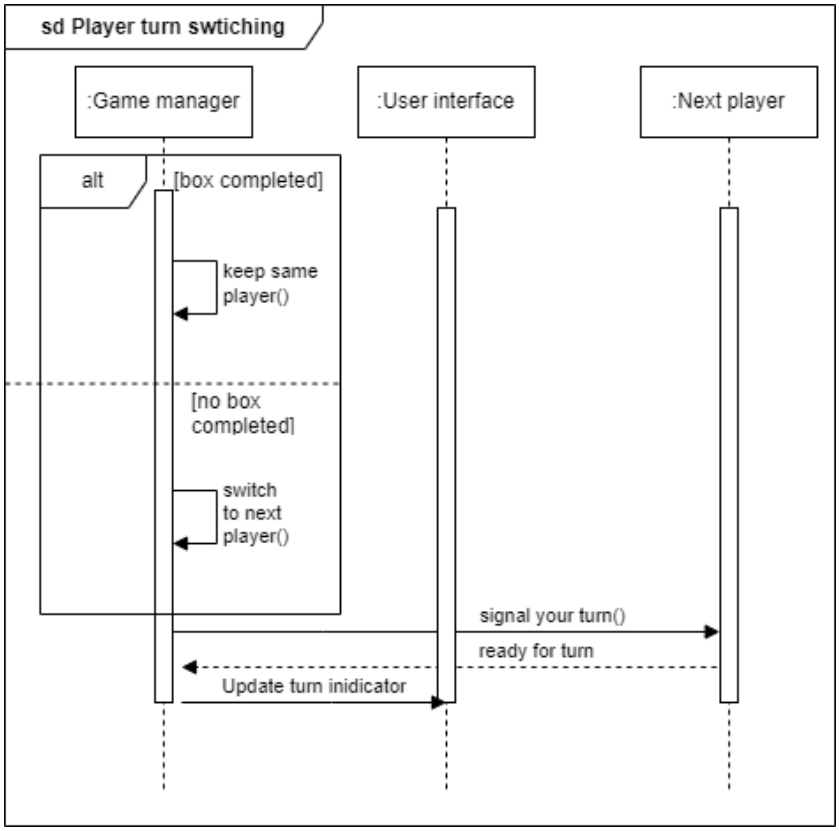

Figure 3.3 : Player Turn Switching Sequence

End Game Sequence

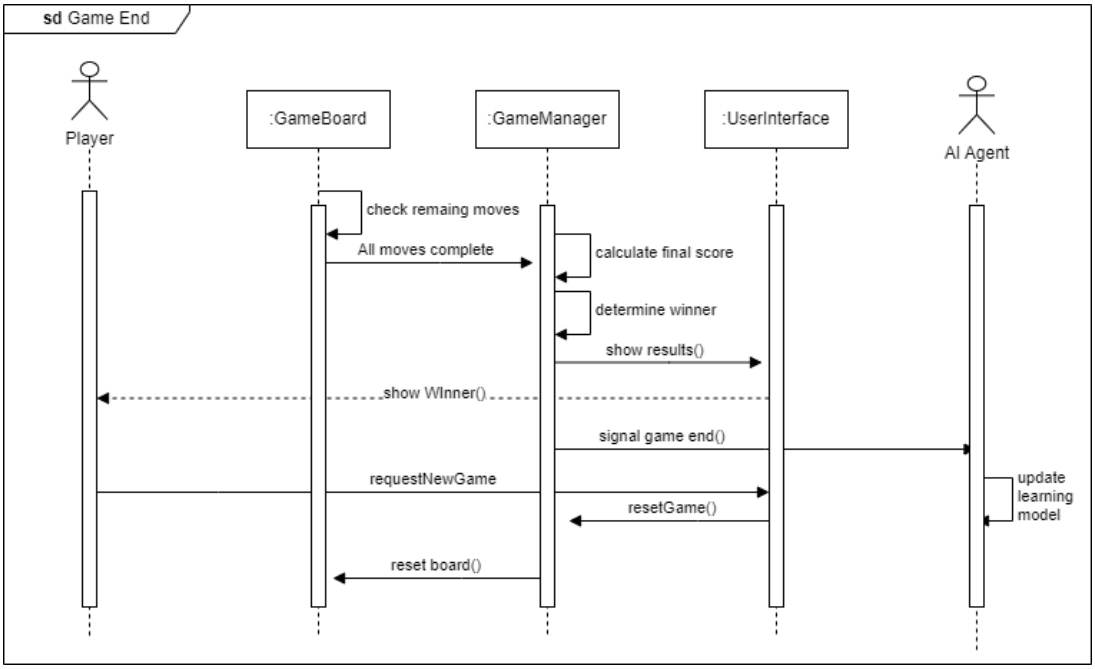

Figure 3.4 : End Game sequence

AI Training Sequence

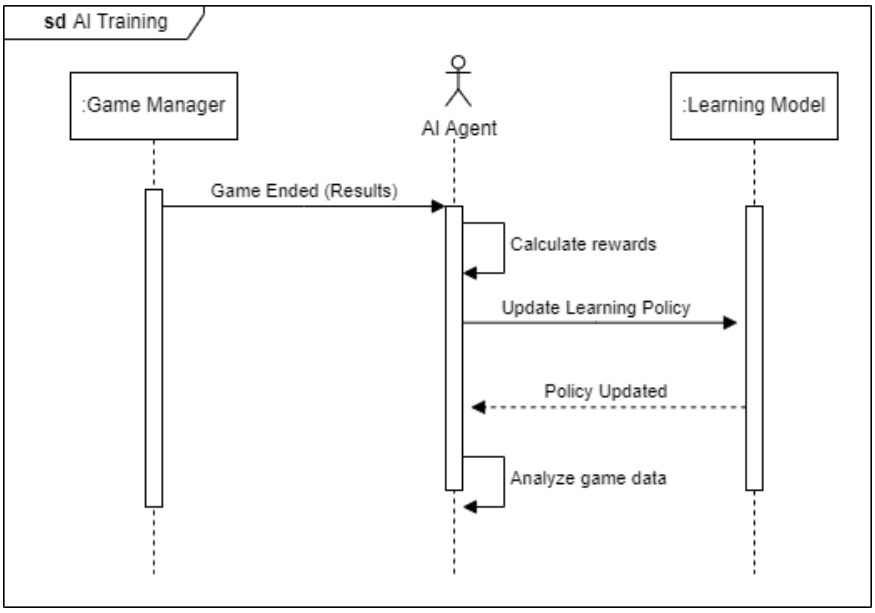

Figure 3.5: AI Training Sequence

7 Intelligence of Agent

The agent will play as an opponent versus a human in the box and dot game, with the aim to make it as competitive as possible to make the chances of the human winning very slim. The agent will learn optimal strategies through self-play by employing Deep Q-Networks as it was demonstrated by Atari games(Mnih, Kavukcuoglu et al. 2013).The experience replay mechanism will allow the agent to learn from past actions to improve learning efficiency. Incorporating motivation strategies will address sparse reward challenges by encouraging exploration, this is a concept explored in recent AI findings(Wirtz, Patterson et al. 2018).This application of the modern RL technique aims to create a robust and intelligent Dots and boxes game agent.

8 References:

- gametable (2019). "Dots and Boxes." Retrieved 09 March 2025, 2025, from https://gametable.org/games/dots-and-boxes/#:~:text=Dots%20and%20Boxes%20has%20been,favorite-%20Pigs%20in%20a%20Pen!

- Mnih, V., et al. (2013). "Playing atari with deep reinforcement learning." arXiv preprint arXiv:1312.5602.

- Schrittwieser, J., et al. (2020). "Mastering Atari, Go, chess and shogi by planning with a learned model." Nature **588**(7839): 604-609.

- Constructing agents with planning capabilities has long been one of the main challenges in the pursuit of artificial intelligence. Tree-based planning methods have enjoyed huge success in challenging domains, such as chess1 and Go2, where a perfect simulator is available. However, in real-world problems, the dynamics governing the environment are often complex and unknown. Here we present the MuZero algorithm, which, by combining a tree-based search with a learned model, achieves superhuman performance in a range of challenging and visually complex domains, without any knowledge of their underlying dynamics. The MuZero algorithm learns an iterable model that produces predictions relevant to planning: the action-selection policy, the value function and the reward. When evaluated on 57 different Atari games3—the canonical video game environment for testing artificial intelligence techniques, in which model-based planning approaches have historically struggled4—the MuZero algorithm achieved state-of-the-art performance. When evaluated on Go, chess and shogi—canonical environments for high-performance planning—the MuZero algorithm matched, without any knowledge of the game dynamics, the superhuman performance of the AlphaZero algorithm5 that was supplied with the rules of the game.

- Silver, D., et al. (2017). "Mastering the game of go without human knowledge." Nature **550**(7676): 354-359.

- Wirtz, J., et al. (2018). "Brave new world: service robots in the frontline." Journal of Service Management **29**(5): 907-931.

*NB: All diagrams were drawn with draw.io