System Design for SE 4G06
An AI-based Approach to Designing Board Games

Team #6, Board Gamers
Ilao Michael, ilaom
Bedi Hargun, bedih
Dang Jeffrey, dangj12
Ada Jonah, karaatan
Mai Tianzheng, mait6

April 6, 2023

1 Revision History

Date	Version	Notes
April 3rd	1.0	Split up into 3 documents from original Rev0 feedback for Rev1

2 Reference Material

This section records information for easy reference.

2.1 Abbreviations and Acronyms

symbol	**description**
SRS	Software Requirements Specification
AI	Artificial Intelligence
A	Assumption
LC	Likely Change
FR	Functional Requirement
NFR	Non Functional Requirement
FSM	Finite State Machine
TA	Teaching Assistant

Contents

List of Tables

List of Figures

3 Introduction

A reference to the SRS can be found SRS, this contains the requirements that the system will implement.
The Module Guide that contains a module overview can be found here
The Module Interface Specification, which contains a detailed and formalized description of each module can be found here

4 Purpose

4.1 System Purpose

The purpose of the system is to simulate the board game An Age Contrived from Bellow Intent's using an AI-based game simulation engine. This system will help identify pitfall's in game mechanics and imbalances in the game's scoring system. The system will also be used to view decisions, scoring and other game state information through various plots, graphs and charts. The system will be used by Game Designers of An Age Contrived to help balance their game as well as AI Research Professors to further understand and develop AI-based game engines.

4.2 Document Purpose

The purpose of this document is to decompose the system into different modules, to show the module and system architecture and provide justifications for each module. This will serve as a guideline for development of the system. Our Module Interface Specification which is our formal specification can be found here. Our Module Guide which decomposes and describes are modules can be found here.

5 Scope

The system is to be used by game designers when creating and balancing their board games and AI researchers

5.1 Assumptions

A1: The game to be analyzed will have a clear reward function that AI Agent can optimize its learning towards.
Rationale: The AI needs a reward to choose which moves are optimal in certain states.

A2: The game to be analyzed will have a clear win condition where the game ends.
Rationale: It shouldn't be an open-ended, infinite game where the goal is to just

increase your score but rather there should be a specific condition(s) that players are playing towards to win and end the game. Otherwise, AI agents won't be able to find the optimal actions to win the game but rather keep outputting sub-optimal decisions which will have little to no value for game design improvements.

A3: The game to be analyzed will have concrete rules that a computer can represent and enforce.
Rationale: If the game cannot be simulated digitally, then AI Agent cannot be trained on the game and the system wouldn't be able to generate any meaningful data to aid the game design process.

5.2 System Context

The figure below describes how our system interacts with the environment surround it.

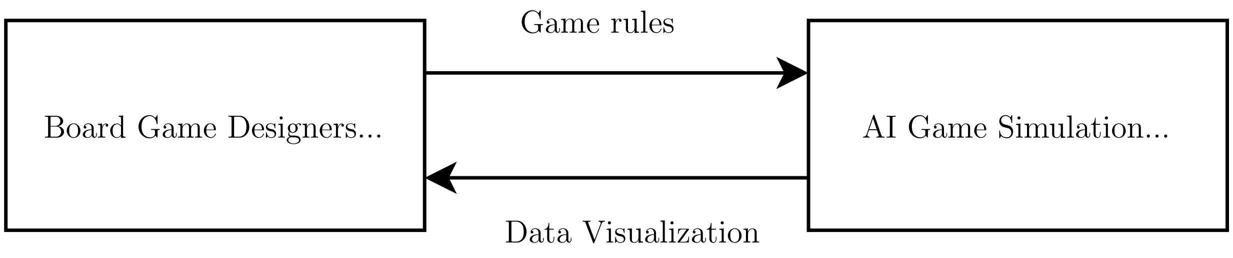


Figure 1: System Context

6 Project Overview

6.1 Normal Behaviour

This project will be used by board game designers to rapid prototype their development of the game rules and game metrics. Board game designers will be able to evaluate the balance of their game in terms if there exists an always winning sequence of moves that are not intended to be that strong to achieve the win condition in their game. Simulating many games through AI players will save a large amount of time in testing by analyzing the moves and winning sequences of the large data set of games they will generate.

The system is designed on a command pattern architecture. When the simulation is run, AI Agents are able to choose a selection of moves in. After a game move is chosen, the game engine will accept and change its state based on the move given. This will feedback on where the game engine will give the game state back to the AI Agents, which they can choose another action after receiving the game state. This will repeat until the end of the game is reached.

6.2 Undesired Event Handling

When the system experiences an undesired event, the system will immediately end any simulation and continue processing a new simulation. The data recorded from that undesired event will be discarded, the AI will gain a negative reward for taking that illegal move to bring the system into the undesired state. This will cause the AI to not make that move in the future.

6.3 Component Diagram

The component diagram below on Figure 2 shows the relation of components, how they are structured and how data flows between each component.

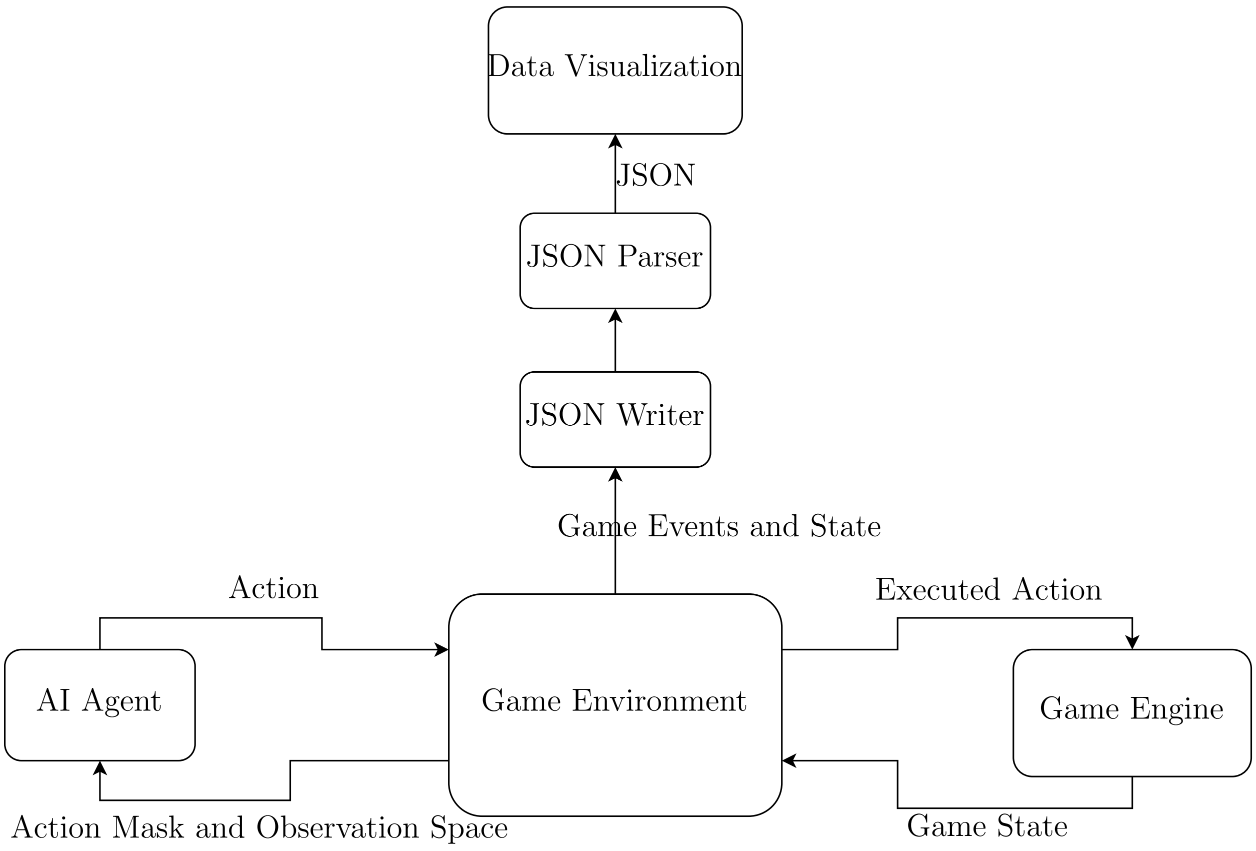


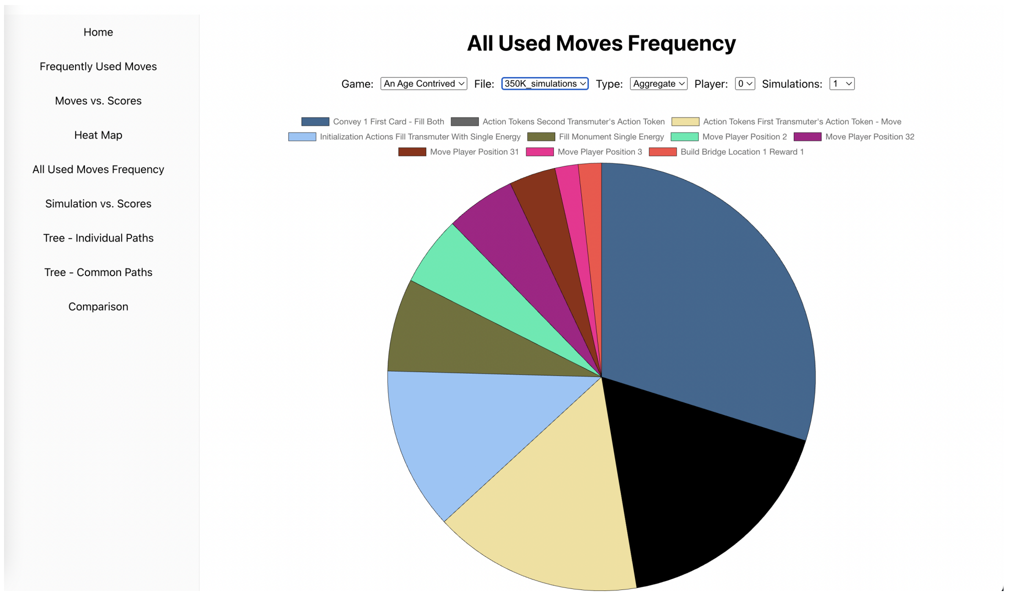
Figure 2: Component Diagram

6.4 Connection Between Requirements and Design

Req.	Decisions
FR12 FR14 NFR2 NFR7	Users must use the data visualizer to see results of their game
FR13 NFR3 NFR4	JSON is used since the Data Visualizer needs to be modular enough to be swapped out, so JSON is used to transfer data betwen Game Engine and Data Visualizer, to keep the implementation of either agnostic.
FR10 NFR6	The Game Engine is fully separate from the game environment as all business logic is stored in the Game Engine to allow for extensibility of game mechanics.
FR2 NFR3	AI Agent is fully seperate from the game environment to allow AI Agents to be swapped easily, without much development time.

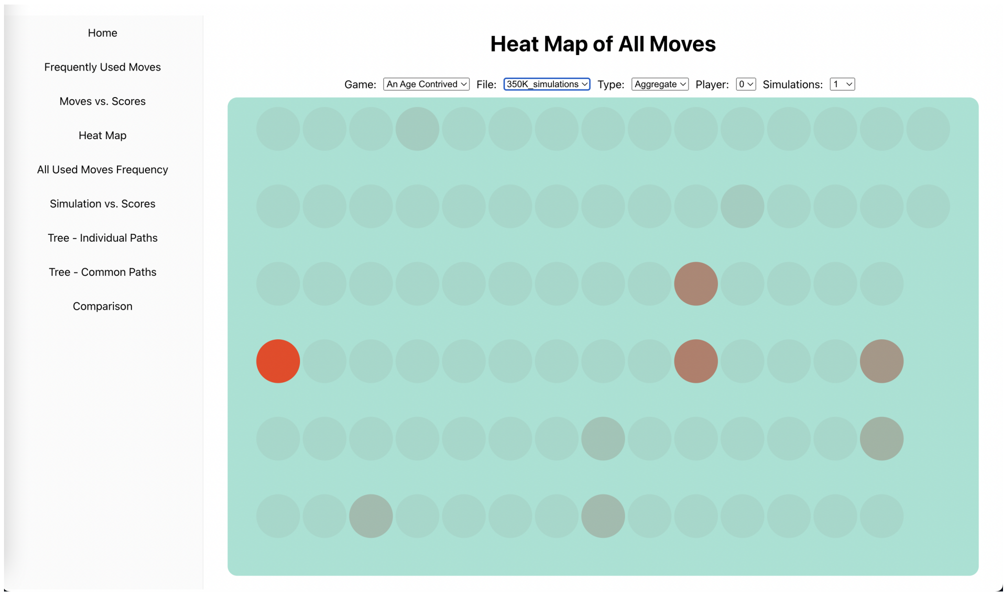
Table 1: Requirements and Design Decisions made

7 User Interfaces



Game: An Age Contrived File: 350K_simulations Type: Aggregate Player: 0 Simulations: 1

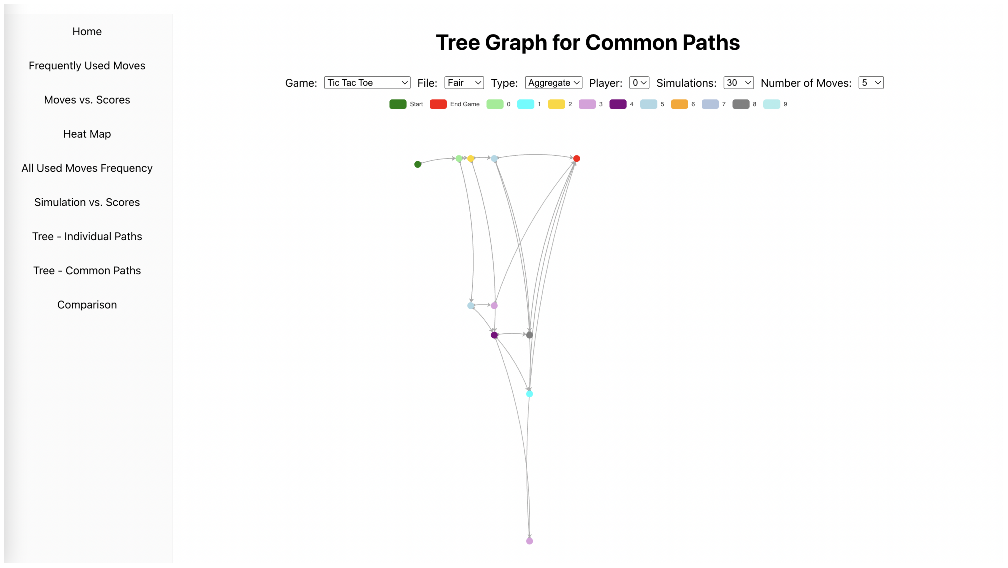
Figure 3: Pie Chart Interface



Heat Map of All Moves

Game: An Age Contrived File: 350K_simulations Type: Aggregate Player: 0 Simulations: 1

Figure 4: Heat Map Interface



Tree Graph for Common Paths

Game: Tic Tac Toe File: Fair Type: Aggregate Player: 0 Simulations: 30 Number of Moves: 5

Figure 5: Common Path Interface

8 Design of Hardware

N/A

9 Design of Electrical Components

N/A

10 Design of Communication Protocols

N/A

11 Development Timeline

Below is an estimate for module completion start and end date, the resource will be in charge of main development, but all developers will aid in architecture and integration.

Module	Start Date	Completion Date	Resource	Description
AI Agent	11-7-2022	12-11-2022	Michael	Apart of initial MVP to setup the architecture, components will be highly coupled to get initial framework
Game Environment	11-7-2022	12-11-2022	Michael	Apart of initial MVP to setup the architecture, components will be highly coupled to get initial framework
Game Engine	11-7-2022	12-11-2022	Michael	Apart of initial MVP to setup the architecture, components will be highly coupled to get initial framework
Game Loop	12-12-2022	12-31-2022	Hargun	Core game mechanic
Actions (Commands)	1-1-2023	1-22-2023	Jonah	Decoupling to add extensibility to additional game mechanics (Architecture change)
JSON	1-1-2023	1-15-2023	Jeffrey	Needed to bridge simulation to data visualizer
JSONDataParser	1-1-2023	1-15-2023	Tianzheng	Need to fetch the output log data to data visualizer
Graph	1-1-2023	1-29-2023	Tianzheng	Visualize data to solve business goals, can be done in parallel with other modules

Table 2: Module Timeline

A Interface

N/A

B Mechanical Hardware

N/A

C Electrical Components

N/A

D Communication Protocols

N/A

E Reflection

E.1 Limitations of Design

The high modularity of the design, each specific component is its own module and they communicate through a specified interface allows for each module to be developed independently. Though the limitation of this is one cannot fully test their module without integration with the other modules, specifically the AI Agent Module cannot be fully tested and approved as working until the Game Engine Modules and Game Environment Modules are completed. This leads to integration errors that take significant development time to solve. For example if a new game mechanic wants to be added, it is first added in the Game Engine Modules, this can be tested only within the game engine without any AI interaction. Only once the AI Agent and Game Engine modules are both updated can they be integrated and checked for bugs and errors, which occur mostly during runtime as the AI can explore many different edge cases which may not have been tested and covered in the initial testing of the Game Engine modules. Another limitation of this system is the architecture can only support turn based games as it can only accept one action for one player at a time, this means for a parallel (real-time) game the architecture will not be supported.

E.2 Benefits and Trade offs of Other Solutions

Our group found that there are not any other solutions available for the system that we are developing. While standalone application of parts of our system have been created before, no one has incorporated our 3 major components of an AI, game engine and data visualization before. For example, a standalone AI game player exists like AlphaGO. Developed by Google, it is the first computer program to defeat a professional human Go player, the first to defeat a Go world champion, and is arguably the strongest Go player in history. Our system does not focus on creating the best AI player that could beat any human playing Age Contrived but rather it is an essential part in simulating multiple iterations of the game with different strategies. We found one system that incorpates the AI and game engine but lacks the data visualization that we are aiming for. Blockus Game Solver, https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1305&context=cpesp Chao, is quite similar to our system as it aims to find winning strategies and loopholes in a multiplayer game. The report outlines strategies that they found had better outcomes, however, it lacks in providing the reader with an easy and intuitive way to visualize the outcomes and does not provide the level of modularity that our system is being developed with.

References

Chin Hung Chao. Blokus game solver. URL https://digitalcommons.calpoly.edu/cgi/viewcontent.cgi?article=1305&amp;context=cpesp.