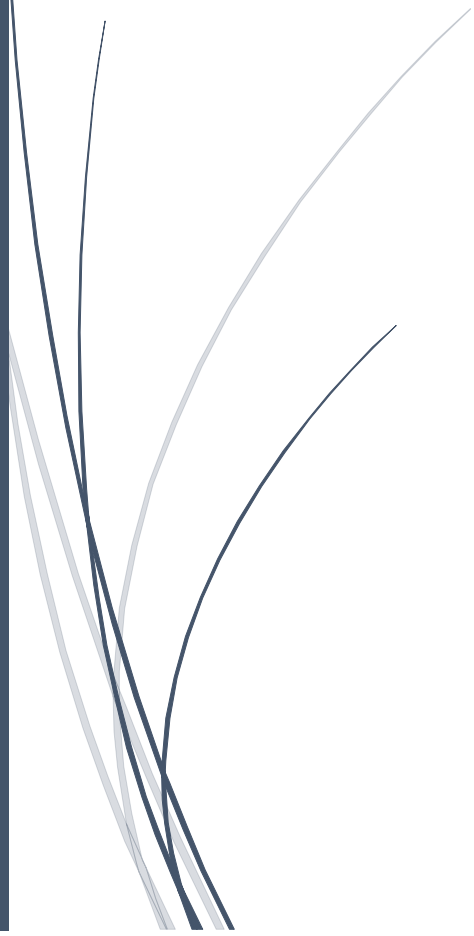
05/02/2021

Initial Plan

Multiplayer Hex Game Engine with Networked Server, 3D UI and Artificial Intelligence

Victoria Howells
STUDENT NUMBER: C1702957
SUPERVISOR: DR FRANK C LANGBEIN
MODERATOR: IRENA SPASIC

CM3203 - ONE SEMESTER INDIVIDUAL PROJECT
40 CREDITS



Contents

1 Project Description

The aim of this project is to implement a multi-player networked server-based game engine with a graphical user-interface for human and AI players. I have chosen the game 'Hex' for this project.

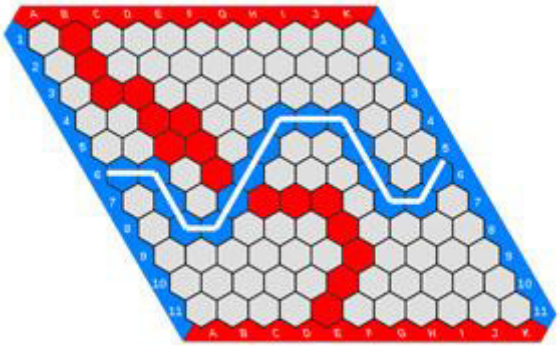
Ultimately, the focus/challenges for this project are two principal areas that have not been covered in detail within course modules:

1. Multi-User Networked Servers – using modern techniques found in industry for robust networked server applications that can be deployed on intranets/the internet
2. Advanced AI opponents – beyond simple minimax, utilising recent advances in reinforcement learning and neural networks

This project will balance both areas with equal importance.

1.1 Background to the game of 'Hex'

Created in 1942 by Piet Hein (and again in 1948 by John Nash) [1], Hex is a 2-player turn-based board game where the goal is for each player to connect opposite sides of the board. It is played on a rhombus shaped board (most commonly an 11 by 11 board) where each player is assigned two sides that are opposite each other. The players take turns where they can replace any hex/position on the board with a stone of their colour, but only if that position has not already been selected by another player. A player wins once they successfully connect their sides together through a chain/path of adjacent stones, or the other player chooses to resign.



Once a stone is placed, it cannot be moved or removed, except if the swap rule is used. The swap rule is where the second player can, on their first turn, choose to move normally or swap their counter with the one placed by the first player. This was introduced to combat the first player advantage.

1.2 Why Hex?

I chose Hex because I wanted to work with a turn-based game with perfect information, allowing me to develop a straightforward baseline AI and spend more time on a complex advanced AI to compete with it. At face value, Hex looks quite a simple game, however from an AI perspective, it has a high branching factor as players can choose any hex on the board and are not limited to specific moves or positions. This makes the heuristic more complex to determine the relative value of playing different positions and whilst it is visually obvious who has won, it takes effort to determine programmatically.

1.3 Approach

My implementation will focus on developing a networked server framework, with a core 'API' interface that will allow human and AI players to compete against each other - the game server will support multiple concurrent games and will store results in a suitable database. It will allow human vs human, human vs AI and AI vs AI, where the human and AI players will be remote to the server. I will also look at comparing the effectiveness of different AI's, since the search space is quite large for Hex (with 121 different positions on an 11 by 11 board for the first player).

[1] https://en.wikipedia.org/wiki/Hex_(board_game)

My core architecture will follow a Model-View-Controller (MVC) pattern using RESTful web-services (most public facing open web-services today follow the REST pattern[2]) where the main game engine will be accessible to multiple different players on different clients. Using a REST based Framework/API will allow any remote players (Human or AI) to submit moves and will also allow for new AI's to be created independently by players by simply following the published API specification (programming language and platform independent).

Both the game server and graphical UI will be developed in Java, using a suitable MVC framework on the server and JOGL/OpenGL for the 3D client, with a standard 2D interface (such as Swing or JavaFX) for non-3D elements. OpenGL is a multi-platform open standard (the most widely adopted 2D and 3D graphics API in the industry[3]) and was part of the final year Graphics module this year. As I am familiar with OpenGL/JOGL, development should be rapid allowing me to focus on the Server and AI components.

The AI's will be written in Java and Python (to show that the RESTful web-services can be supported in multiple languages) and will run separately to the game server. I will run a 'competition' between a baseline AI and a more sophisticated AI and evaluate the effectiveness of different AI strategies in playing Hex.

The baseline AI will be a simple minimax and is not intended to be particularly sophisticated. Initial research into turn based games with perfect information has shown that a Monte Carlo Tree search, and more advanced approaches such as Google DeepMind AlphaZero [4] (utilizing Monte Carlo Tree search and reinforcement learning with a neural networks) are likely to be more effective. This is because, given the large branching factor for Hex, a normal tree-search attempting to follow all possible combinations will likely fail to deliver an optimal result. Monte-Carlo will prioritise the most promising branches aided by the output from reinforcement learning. Therefore, I will start off by developing an AI that implements the Monte Carlo approach, then extend it with a neural network in the style of AlphaZero.

[2] https://en.wikipedia.org/wiki/Representational_state_transfer
[3] https://www.khronos.org/opengl/
[4] David Silver, Thomas Hubert, Julian Schrittwieser, Demis Hassabis. (2018). *AlphaZero: Shedding new light on chess, shogi, and Go.* https://deepmind.com/blog/article/alphazero-shedding-new-light-grand-games-chess-shogi-and-go

2 Project Aims and Objectives

- Implement a playable 'Hex' Game Engine
    - Java game implementing the default rules of Hex with options for board size and using the swap rule
    - Game results and statistics for each game stored in a database
    - Support for multiple concurrent games, including player management and registration
- Implement a networked REST Server to deploy Game Engine (API interface)
    - Java based RESTful API using a suitable framework, running on a remote networked server
    - Players (both AI and Human) can connect via the API interface and play games
    - Effective synchronised gameplay between players with support for disconnection/reconnection to re-join an active game later
    - Documentation/Specification for API available to players for creating their own AI's
- Develop a 3D GUI for human players
    - Client in Java allowing a human player to play and visualise the game
    - 3D board using OpenGL with 2D menus and game controls
    - Enable game management (re-join game, suspend game)
    - Controls should be simple and intuitive
    - UI will run cross-platform (Windows, Mac, Linux)
- Develop Multiple AI Players using different algorithms
    - AI players can connect to games and play via the RESTful API
    - AI will be capable of selecting a legal move for the game within the allotted turn time
    - Baseline AI (minimax with a simple heuristic)
    - Advanced AI (Monte Carlo Search tree and/or AlphaZero with additional neural network/reinforcement learning)
    - AI can be platform and language independent (players can create their own AI's)
- Competition: Evaluation of AI's against humans and other AI's
    - The AI should be able to play against different AI's and human players
    - The AI should be able to beat human opponents in most cases, in particular the more advanced AI's
    - The more advanced AI's should beat the baseline AI in most cases

3 Work Plan

Assumptions:

- Project will follow Agile principles – get a basic working solution quickly, then iterate and enhance
- Model-View-Controller pattern will be adopted
- Bug/Feature tracker will be used to record additional features and classify bugs (reviewed weekly)
- Activity Log/Diary will be kept to record weekly progress, support supervisor meetings and Final Report
- Weekly Supervisor meetings with two longer Supervisor Review meetings in weeks 5 & 8

3.1 Tasks & Timeline

Week	Tasks
1 *(1st Feb)*	**Write Initial Plan/Report:** submit by 8th February **Request Access**: request access to University OpenShift, OpenStack, MongoDB and PostgreSQL (to evaluate and select in weeks 2 & 3) **Background Research:** Some background research into Hex, Monte Carlo, AlphaZero and REST frameworks like SpringBoot has already been completed. I will continue to investigate these areas and fill in background sections of the report as I go **Design Hex Game Engine (Model)**: This involves creating a UML class diagram and figuring out how the game will run, and how its different classes will link **Start Game Engine Prototype:** Start implementing Game Engine for 'single game' (e.g. without swap rule, database or multi-player - running locally within IDE)
2 *(8th Feb)*	**Ethics Course**: Complete the ethics course. **Complete Basic Game Engine:** Complete basic game engine **Game Database:** Design (and decide on SQL or NoSQL) database and add to Game Engine using ORM Framework (e.g. Spring Data/JPA/Hibernate) **Player Management:** Design and implement basic player registration and multi-game management within Game Engine **Unit Testing:** Create initial test cases for the game engine and implement using test framework
3 *(15th Feb)*	**Request Ethics Approval**: request access to allow people to play games on server **Design REST API (Controller):** design REST API for game engine **Select Server Platform:** select server platform (e.g. OpenShift, OpenStack, Azure, AWS etc.) **Implement REST Server:** Implement REST API's and link to Game Engine **Document API:** Ensure all endpoints are documented and accessible (e.g. in Swagger) **Testing:** test REST endpoints using Swagger/Postman

4 *(22nd Feb)*	**Background Research:** Select Java 2D GUI frameworks e.g. Swing, JavaFX **UI Design (View):** Create wireframes and heuristic evaluation to ensure design meets usability requirements **Client Class Design**: Create a UML diagram for the classes used in the Client and how they interact **Prototype GUI client:** Start off by implementing the 3D game board using OpenGL and connect to REST services
5 *(1st Mar)*	**Complete GUI Client:** human GUI should be completed this week (game should be playable by human players) **Test Play:** Human vs Human **Heuristic Evaluation of GUI:** Ensure that usability standards have been met for the user and add any weaknesses to bug tracker **First Supervisor Review Meeting:** First of the longer supervisor review sessions (Game Engine, REST Server and most of the GUI for human players should have been implemented). This is before AI development starts so would be a good point to assess the projects progress
6 *(8th Mar)*	**Baseline AI Design:** Create a UML Class diagram design for the classes to be implemented for the baseline AI **Baseline AI Implementation:** implement minimax with simple heuristic **Test Play:** Human vs Baseline AI, Baseline AI vs Baseline AI
7 *(15th Mar)*	**Background Research:** Complete research into the advanced AI approaches (Monte Carlo Search Tree and AlphaZero) **Start Monte Carlo AI player:** Develop Monte Carlo AI (probably in Python) and link to REST API services
8 *(22nd Mar)*	**Complete Monte Carlo AI player:** complete Monte Carlo implementation **Test Play:** Baseline AI vs Monte Carlo AI **Start AlphaZero AI:** add AlphaZero to Monte Carlo implementation **Second Supervisor Review Meeting:** Second of the longer supervisor review sessions (Advanced AI should have started). This meeting will be useful to reflect on what milestones have been achieved, and what work should be prioritised over the 3 Easter weeks
Easter *(29th Mar)* *(5th Apr)* *(12th Apr)*	**Complete AlphaZero AI:** complete the Alpha Zero implementation **Train AlphaZero:** train AlphaZero AI by playing against baseline AI **Bug fixing and Enhancements:** bug fixes and final enhancements to end-to-end Game Engine, Server, GUI and AI's **Evaluation:** determine how competition/evaluation will be performed

9 *(19th Apr)*	**Test Play:** Baseline AI vs Monte Carlo, Baseline AI vs AlphaZero, Monte Carlo vs AlphaZero **Final Bug Fixes to support Evaluation:** complete any final fixes to support good evaluation results
10 *(26th Apr)*	**Evaluation:** Run AI evaluation and collect/analyse results **Final Report:** start final report using weekly logs, designs created through implementation and evaluation results
11 *(3rd May)*	**Final Report:** compete the bulk of the report
12 *(10th May)*	**Final Report:** review the report and make any final changes necessary. **Submit Project**: Gather all data, appendixes, and application code and submit by 14th May

3.2 Milestones

1. Submit Initial Plan [Week 1]
2. Complete Basic Game Engine (Model) [Week 2]
3. Complete API Design/Specification [Week 3]
4. Deployed REST Server with Database and Game Engine (Model & Controller) [Week 3]
5. Complete 3D GUI (View) [Week 5]
6. Complete Baseline AI [Week 6]
7. Complete Monte Carlo AI [Week 8]
8. Complete AlphaZero AI [Week 9]
9. Evaluation Data Gathered [Week 10]
10. Submit Final Report and Supporting Files [Week 12]

3.3 Deliverables

8th February 2021

- Initial Report

14th May 2021

- Game Engine and Server
- API Specification
- 3D GUI
- AI Players (Baseline and Advanced)
- Final Report (including AI Evaluation)