Roadmap

The roadmap is used to plan out (and log) features for each tagged version of the game. Git tags will be used as well so a simple [git checkout tags/v0.1.0] can be used to check what the code was like at a certain point.

Contents

[V0.1.0 2](#_Toc470554883)

[Final version for V0.1 2](#_Toc470554884)

[V0.0.1 2](#_Toc470554885)

# V0.1.0

This version should have the raw foundation of the game, including the GUI for the player, the class to represent a game board, and the movement tree.

Goals:

* Have a class to represent the game board
* Have a basic controller class setup for the Player (AI will come later) to interact with the board via a GUI
* The GUI should at the moment, only bother displaying the game board, as well as who’s turn it is
* The game board class should have the basic game rules implemented (can only place pieces on empty spaces. Game is won if 3 pieces are in a row, etc.)
* Have a way to hash the game board
* Have a way to represent a tree of moves (the Node class and the MoveTree class provide this)
* Have a way to calculate the statistically best path of moves to make (Average.statisticallyBest sorts this out, I know it’s a bit strange to have this before working on the AI, but I just needed it out the way)
* Allow any Node class to be serialised/unserialised.

## Final version for V0.0

The final version of every Minor version update should contain general clean-ups/refactoring of code.

Goals:

* Remove the ‘MoveTree’ class. First, create a Node.root property to create an empty node (suitable as a root). Then, move the MoveTree.walk function into the Node class, since it makes more sense and provides more flexibility there.

## V0.0.2

Achieved:

* Added basic serialisation support.
* Implemented serialisation for the ‘Hash’ class.

## V0.0.1

Achieved:

* Have a way to hash the game board
* Have a way to represent a tree of moves
* Have a way to calculate the statistically best path to make
* Included this document