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](#_Toc471337814)

[Final version for V0.0.X 2](#_Toc471337815)

[V0.0.4 2](#_Toc471337816)

[V0.0.3 2](#_Toc471337817)

[V0.0.2 3](#_Toc471337818)

[V0.0.1 3](#_Toc471337819)

# 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, a few pre-requisites for AI, and a way to represent a tree of moves.

Goals:

* Have a class to represent the game board **[V0.0.2]**
* Have a basic controller class setup for the Player (AI will come later) to interact with the board via a GUI **[V0.0.3]**
* The GUI should at the moment, only bother displaying the game board, as well as who’s turn it is **[V0.0.3]**
* 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.) **[V0.0.2]**
* Have a way to hash the game board **[V0.0.1]**
* Have a way to represent a tree of moves (the Node class and the MoveTree class provide this) **[V0.0.1]**
* 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) **[V0.0.1]**
* Allow any Node class to be serialised/unserialised. **[V0.0.1]**

## Final version for V0.0.X

The final version of every Minor version update should contain general clean-ups/refactoring of code. The final version is generally also the next minor version. So the version 0.0.4 is also v0.1

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. **[Done in V0.0.2]**
* Use the ‘Board’ class as a namespace for the ‘Hash’ class. So ‘Hash’ becomes ‘Board.Hash’. This better reflects its purpose, and becomes consistent with ‘Board.Piece’. **[Done in V0.0.4]**
* ~~Change ‘Node’ and ‘Hash’ to be created using~~ [~~Object Initialisers~~](https://msdn.microsoft.com/en-us/library/bb384062.aspx)~~, as it creates cleaner (and clearer) code.~~ (After a change of thought, this could be more of a burden, as it would require me to make some read-only properties non read-only)

## V0.0.4

Achieved:

* Put the ‘Hash’ class under the ‘Board’ namespace, so it is used like ‘Board.Hash’.

## V0.0.3

Achieved:

* Add support for running a thread for all the game logic, and providing a way for the two threads to speak to each other.
* Add a GUI, and a controller to let the player interact with the board.
* Add code to handle when two controllers tie.
* Fixed a bug in the win checking code. It was checking a wrong slot when checking “Top right to bottom right”.

## V0.0.2

Achieved:

* Add basic serialisation support.
* Add the ‘Board’, and ‘Controller’ classes, with very basic tests. (This was a lot of code)
* Implement serialisation for the ‘Hash’ and ‘Node’ class.
* Remove the ‘MoveTree’ class, and move its functionality to ‘Node’.

## 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