Initial Plan - Chess Analysis Engine

Author: Sophie Greenwood (c1013107)
Supervisor: Christine Mumford
Moderator: Kirill Sidorov
Module: CM0333 Individual Project (30 credits)

Project Description

This project aims to design and implement a chess analysis engine. This is a program that takes as input a given position in the game of chess and returns as output what it believes to be the strongest continuation. "Strongest" in this context can be subjective, but generally refers to the moves that give each player the best odds of winning the game, or failing that, salvaging a draw.

This functionality may then be expanded upon to allow humans (or other computers) to play full games against the engine, with the program analysing each new position as it arrives.

In essence, this project is an investigation into various techniques that may be used to allow a computer to play effective chess while keeping the running time to an acceptable level.

Project Aims and Objectives

As mentioned above, this project aims to implement a chess analysis engine. This will require the following core functionality:

- Full compliance with all rules of chess
    - Specifically, this requires the program to be able to compute what is and isn't a valid move in any given position, taking into account the limitations surrounding the "special" moves of castling and en passant. It must also be able to understand the various ways a game may end.
- Assessment of chess positions
    - The engine must be able to take as input and evaluate a static position, assigning to it a numerical assessment indicating who it believes to have the advantage and how significant this advantage is. Standard convention is that a positive number corresponds to an advantage for White, and a negative numbers corresponds to an advantage for Black. The precise method of assessing a position varies between chess engines and often plays a part in determining the computer's style of play.
- Analysis of chess positions
    - Building on the previous point, the engine must be able to scan ahead from the initial position given to it as input, establishing what it believes to be the strongest continuation. For reasons that will be discussed below, it should also track second- and perhaps even third-best continuations.

Once these have been established, the following are extra features that would be desirable to implement given sufficient time:

- Game mode
    - Essentially adapting the analysis feature in order to allow the engine to play full games against a human/computer opponent instead of examining a single position. On any given turn (including those of the opponent), the engine should be analysing the current position to determine its next move and predict those of the opponent. This may (and probably should) also implement use of opening and endgame databases, which cover known theoretical positions that may otherwise be difficult to analyse correctly (especially in the endgame).
- Variable skill level
    - Hand-in-hand with the game mode feature. A typical modern chess engine is capable of giving a grandmaster a tough match, and most players would prefer to play a lesser opponent. This would most likely be implemented using the second- and third-best continuations as mentioned above.
- Improved performance
    - This is an intentionally vague objective, but a simple one; the engine should play the strongest chess it possibly can without sacrificing too much in regards to runtime. Implementing this objective is essentially an extension of what this project is about!

- GUI
    - While a chess engine can function perfectly well through the medium of command line, it is much more user-friendly and visually attractive to use a graphically-based interface. A command line program would require the player to either memorise the game state (!) or make the moves on a physical chess board, and neither of these solutions are especially elegant.
- Learning
    - The engine ought to be able to learn from past mistakes and adapt its play to be better prepared for future opponents. This is tricky to implement in the context of chess, where a bizarre tactic that generally will not work may on occasion happen to do so. It would be remiss to miss this tactic purely on the basis of past experience (although this would lead to more "human-like" chess), but it would also be remiss to repeat the same type of mistake over and over. As such, this functionality would have to make certain compromises. Done well, however, a capacity for learning would be an excellent means to improve the engine's playing strength.
- Training facilities
    - Chess is a rather complex game that can be a little overwhelming to new players. It might therefore be useful to provide functionality geared towards these newer players to help familiarise them with the game. In this hypothetical "training mode", the computer might, for example, highlight where a selected piece can be moved to, or indicate pieces that are under threat of capture.

Work Plan

- Autumn Semester (100 hours):
    - Ongoing throughout semester: Update documentation in preparation for final report
    - Week 1: Initial Plan - setting general goals of project
    - Week 2: Research - study of existing products, identify common shortcomings
    - Week 3: Research - existing methods used in chess analysis
    - Week 4: Design - identify use cases / user stories
    - Week 5: Design - derive base classes and construct class diagram
    - Week 6: Design - GUI design, testing strategy
    - Week 7: Implementation - establish framework; basic rules
    - Week 8: Implementation - basic rules (if incomplete); static position assessment
    - Week 9 onward: Implementation - static position assessment

- Spring Semester (200 hours):
    - Ongoing throughout semester: Collate documentation into formal report
    - Week 1 - 5: Implementation - establish basic analysis capabilities
    - Week 6: Testing - check capabilities of current implementation and note potential areas for improvement. Run time analysis?
    - Week 7 onward: Implementation - optional features, with emphasis on implementing GUI and "game mode"; Testing - continued tests of performance and stability

Final report due by 3rd May 2013.