Chess Engine

{Engine Name}

SFU OS Development Club

Start Date: February 2021

Revision History

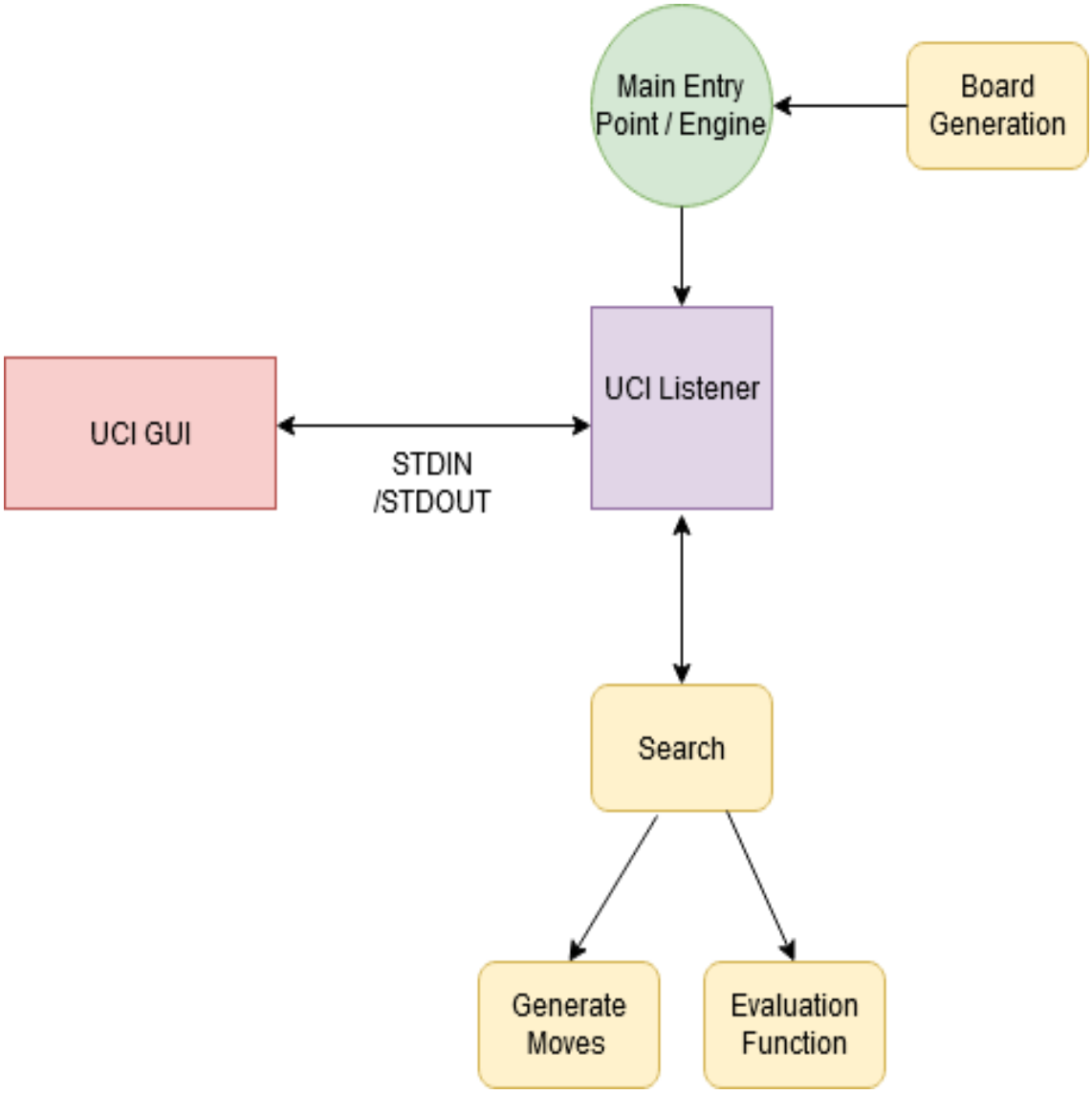
Revision	Name	Date	Description
v1.0.0	Michael Zaghi	2021/02/06	Creating base requirments doc and adding preliminary requirements
v1.0.1	Michael Zaghi	2021/02/13	Adding preliminary architecture diagram

1 Purpose

The purpose of the project is to practice software engineering principles (clean code, testing, code review), data structures and algorithms, design patterns, and open source contribution in a collaborative environment. The chess engine project was chosen because its implementation sufficiently covers these points. It is also a useful piece of software that can be benchmarked against other engines for performance.

2 Architecture

We will draw out a UML class diagram as we get a better understanding of the overall functional requirements. Currently adding a preliminary architecture.



3 Functional Requirements

REQ-1:
Move logic should be implemented for each piece, as well as special moves (castling, en passant, and pawn promotion)

REQ-2:
The board representation should be implemented using bit boards. This means that REQ-1 will be implemented using bit manipulation.

REQ-3:
The search tree should be built using depth first search (DFS) using Minimax and Alpha-Beta pruning algorithms.

REQ-4:
The engine should be able to *ponder* (search the tree while it is the opponents move).

REQ-5:
As search is conducted each node containing a board position needs to be evaluated. Initially, the evaluation function can be implemented to only consider material (piece values: Pawn=1, Bishop/Knight=3, Rook=5, Queen=9). The evaluation function should also determine checkmate. Later, different heuristics can be added to the evaluation function to improve the engine's overall playing strength (King safety, Knight outposts, space, etc.).

REQ-6:
The engine should be able to integrate with a UCI (Universal Chess interface) capable GUI. UCI is a protocol that allows the engine to receive commands (event-driven) and send commands (event-listening). It uses an MVC deign pattern, where the UCI capable GUI incorporates the view, controller and the model (game state such as fifty-move-rule, time, search depth, pondering etc.).

Resources

General Chess Programming Information:
https://www.chessprogramming.org/Main_Page

Overview of UCI protocol:
http://page.mi.fu-berlin.de/block/uci.htm