Mojave Chess engine

Welcome

This is a chess engine written by me Oxyn in c++. It is a combination of many chess programming techniques.

Note: Currently, this software is linux exclusive as it relies on Gnu c compiler inbuilt funcitons, this is because linux this is because linux is my devolopment enviroment eventually ill port it to windows.

The majority of useful doucmentation including a changelog and a todo list for features to be implemented into the project can be found in the docs folder.

- Todo list
- Changelog

Compilation

Mojave uses cmake to generate a makefile which can then be used to build Mojave.

The main binary has no dependancies other than the c++ standard library.

However, Mojave does have tests which are dependant on googletest

and a benchmark dependant on hayai

These are optional and **do not** need to be compilied unless desired.

To compile the Mojave binary:

```
mkdir build
cd build
cmake .. -DDEBUG=<OFF | ON> -DBUILD_TESTS=<OFF | ON> -BUILD_BENCHMARK=<OFF | ON>
make
```

This will build the binaries under /build/bin

These binaries includes:

- **mojave**, this is the mojave chess engine program
- **tests** this runs all the tests and returns results
- **benchmark** shows preformance of some preformance critical functions in source code

The **tests** and **benchmark** will not be built unless the cmake flags are set **ON**

The debug option enables debugging mode on the **mojave** binary this allows:

- output useful for debugging
- compilation with the -g flag useful for gdb debugging

Usage

To functionaly use mojave you will need another piece of software called a Chess GUI. This is a program that will provide the interface and filicate interaction with the engine.

Mojave was built with Chess Arena in mind.

Features

This engine uses many technique outlined on the Chessprogramming wiki for efficency and effectiveness.

Board Representation

Keeping track of board states:

- Bitboards

Leaping Piece Move Generation:

leaping pieces use precomputated lookup arrays

- Pawn move & attack generation
- King move & attack generation
- Knight move & attack generation

Sliding Piece Move generation:

Sliding pieces use the "Classical approach" of a Ray table combined with isolating blocking pieces and a bitscan

- Queen move & attack generation
- Pawn move & attack generation
- Bishop move & attack generation