Ferretti

Contents

1 Obbiettivi del progetto

Lo scopo di questo progetto è implementare il gioco degli scacchi in una versione CLI[1]. Auspicabilmente, vorremo fare in modo che l'utente possa giocare in due modi:

1. Una versione locale ove il programma gestisce solo la logica del gioco e il risultato su schermo.
2. Una versione in cui si gioca contro un modello locale di CPU.

Punti cardini dello sviluppo

1. È un progetto didattico. Esistono centinaia di progetti già funzionanti online.
2. Vorremmo che fosse efficiente e non solo funzionante.
3. Auspichiamo una buona lettura del progetto da parte di terzi. Per questa ragione siamo interessati a documentare bene il progetto e le scelte progettuali prese.

2 Lista delle cose da fare

Presentiamo una lista di cose da fare per ogni componente del progetto. Nota: L'ordine conta! Trovi prima le cose più necessarie e in fondo quelle meno.

2.1 Board

1. Controllare se effettivamente il costruttore di Board funzioni.
2. Aggiungere controlli ed eccezioni per la costruttore della posizione a parte dal fen.
3. Fare funzioni più piccole a partire da funzione da fen to board.

2.2 Engine

1. Aggiungere dei test per tutto.
2. Aggiungere stampa risultati e menu di continuazione.
3. Prendere il numero come string e poi parsarlo a uint8_t. Attualmente fa crashare il programma se in input si mette un -n.
4. Sistemare *isMate*.

[1] Command Line Interface

2.3 Coords

1. Aggiungere dei test per tutto.

2.4 Printer

1. Aggiungere dei test per tutto.

2.5 Manu

1. Attualmente tutto fatto.

2.6 Piece

1. Aggiungere dei test per tutto.

2.7 Stato_0

1. Creare la classe
2. Spostare funzioni da Engine a Stato_0
3. Sistemare *saveGame*.
4. Sistemare *loadGame*.
5. Sistemare *takePlayerTurn*.
6. Sistemare *takeEngineTurn*.
7. Sistemare *playGameVsEngine*.

2.8 Runner

1. Tutto completato fino a questo momento.

2.9 Nuove funzionalità

1. Implementare una versione di gioco con orologio funzionante in tempo reale.
2. Creare un motore per giocare contro il computer offline.
3. Aggiungi supporto multipli salvataggi

2.10 Idee per Evaluation

2.10.1 Generale:

- delta di materiale
- mobilita' pezzi
- centralita' pezzi
- king's safety
- pawn islands
- pedoni doppi/tripli
- pedoni in prossimita' di promozione (7o e 2o rank)
- pinned pieces
- numero mosse legali totali
- occupazione delle ali.
- triangolazione re per guadagnare tempo.
- dono greco.
- sacrificio pezzi minori per vantaggio sviluppo.
- forchetta
- alfieri in fianchetto
- attacchi di scoperta
- pezzi sospesi (+malus)
- pezzi difesi (+bonus)
- sacrificio semplificativo
- evitare il cambio nello svantaggio
- cambio forzante
- valore nel far perdere l'arrocco
- cavallo messo in avamposto non scacciabile
- scacco di scoperta

2.10.2 Endgame (+middlegame)

- mobilita' re
- pedoni passati
- pedoni passati supportati da altri pedoni
- Cattura pedoni centrali contro laterali.
- Mosse Zugzwang

2.11 Idee per la search

2.11.1 Generale

- cercare prima e piu' a fondo se si da' scacco
- cercare prima ogni mossa che cattura un pezzo
- cercare piu' a fondo se si vince un pezzo
- cercare per ultimo ogni mossa che perde materiale
- cercare prima ogni mossa che centralizza un pezzo (o che si avvicina al re opposto)

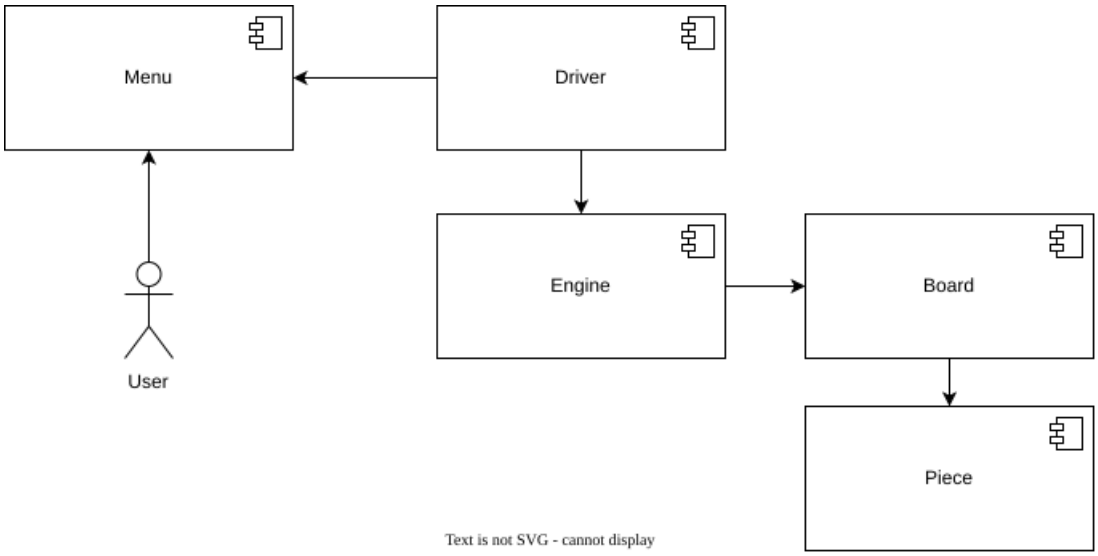
2.11.2 Middlegame

- Cercare in profondità di attaccare le caselle deboli dell'avversario

2.11.3 Endgame

- cercare prima le mosse che diminuiscono la mobilita' del re avversario
- cercare prima le mosse che obbligano il re avversario ad avvicinarsi al bordo scacchiera
- cercare prima le promozioni di pedoni

3 Architettura del progetto



3.1 Coords

Descrizione classe: È una classe che rappresenta le coordinate nella scacchiera. Quindi, l'insieme delle righe sarà così composto: $R := \{A, B, C, D, E, F, G, H\}$

Mentre l'insieme delle colonne: $C := \{1, 2, 3, 4, 5, 6, 7, 8\}$ Una coordinata non è altro che: RXC.

Costruttori Sono presenti quattro costruttori:

1. Costruttore vuoto
2. Costruttore passando indici riga e colonna
3. Costruttore passando notazione stringa scacchistica
4. Costruttore tramite altra Coords

Attributi:

- file: Colonna della scacchiera
- rank: Riga della scacchiera

Metodi:

1. Operatori:
   (a) ==

(b) !=

(c) =

2. update, per modifica le coordiante
   (a) Passando un'altra coordianta
   (b) Passando le esplicite coordinate riga e colonna
3. Metodi statici di controllo:
   (a) isValid: per l'indice
   (b) isLetter: per la colonna
   (c) isNumber: per la riga
   (d) isInBounds: Se non eccede la scacchiera 8x8.

Decisioni progettuali Da subito ci siamo accorti che le coordiante sarebbero state qualcosa di importante. Il primo approccio è stato quello di usare una *struct*. Tuttavia, dopo alcuni commit abbiamo preferito passare a una *class* avendo *Coords* diversi metodi propri e controlli da effettuare.

3.2 Board

Descrizione classe: Rappresenta la foto di una scacchiera in questo momento.

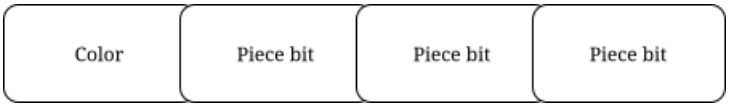
Costruttori Sono presenti due costruttori:

1. Costruttore vuoto: Genera la posizione classica degli scacchi.
2. (std::string): Costruttore che prende un FEN costruendo la relativa posizione.

`bitmap board`

La bitmap contiene in totale 3 informazioni:

1. Indice, che viene trasformato in coordinate
2. Tipo del pezzo tramite 3 bit
3. Colore del pezzo tramite 1 bit



La posizione *board* 0 rappresenta la casella: a8. Procedendo con l'indice di *board*, iniziando dalla casella *a8* (casa in alto a sinistra nella scacchiera), si procede da da sinistra verso destra e dell'alto verso il basso.

Riposto alcune corrispondenze per spiegare ulteriormente:

- Indice 7: h8
- Indice 9: b7
- Indice 18: c6
- Indice 37: f4

Metodi: Ancora in fase di definizione.

Decisioni progettuali Sapendo che altrove nel progetto si creeranno diverse istanze di *Board* abbiamo cercato di renderla il più possibile leggera a livello di memoria. L'attuale decisione ultima è quella descritta della bit-board. Questa soluzione permette di avere una board contenuta in *32 byte*. ($4bit * 64caselle = 256bit/8byte/bit = 32byte$) Il resto dei metodi ove possibile è statico per non avere peso sulla singola istanza.

3.3 Engine

Descrizione classe: Classe con il motore scacchistico e metodi per valutare una posizione e scegliere la mossa migliore.

Costruttori E' presente un solo costruttore:

1. Costruttore vuoto: serve a inizializzare le componenti interne.

```
BoardTree bt;
```

Metodi:

Decisioni progettuali

3.4 Driver

Descrizione classe:

Costruttori

1.

Attributi:

-

Metodi:

1.

Decisioni progettuali

3.5 Stato

Descrizione classe:

Costruttori

1.

Attributi:

-

Metodi:

1.

Decisioni progettuali