Klotski Game

Progetto finale del corso
ELEMENTI DI INGEGNERIA DEL SOFTWARE

Marcato Matteo
matteo.marcato.5@studenti.unipd.it

Piras Matteo
matteo.piras.3@studenti.unipd.it

Quartucci Davide
davide.quartucci@studenti.unipd.it

Romio Simone
simone.romio@studenti.unipd.it

Indice

Capitolo 1

Manuale illustrativo

1.1 Obiettivo

Questo documento rappresenta l'insieme di 'deliverables' richieste per il progetto finale del corso che riguarda la creazione del famoso gioco: Klotski-Game. Il tutto sarà suddiviso in capitoli incentrati su: manuale illustrativo, specifiche del progetto, design, system testing, report di unit test e JavaDoc.

1.2 Cenni Storici

Klotski è un rompicapo a blocchi scorrevoli che si ritiene abbia avuto origine all'inizio del XX secolo. Il nome può riferirsi a un layout specifico di dieci blocchi o, in senso più ampio, ad un intero gruppo di rompicapi simili a blocchi scorrevoli in cui l'obiettivo è spostare un blocco specifico in una posizione predefinita.

1.3 Regole del gioco

Come in altri rompicapi a blocchi scorrevoli, diversi pezzi di blocchi di dimensioni diverse vengono posizionati all'interno di una scatola, di solito di dimensioni 4x5. Tra i blocchi c'è uno speciale (di solito il più grande) che deve essere spostato in un'area apposita designata dalla griglia di gioco. Al giocatore non è permesso rimuovere i blocchi e può solo farli scorrere in orizzontale e in verticale. Gli obiettivi comuni sono risolvere il rompicapo con un numero minimo di mosse o in un tempo minimo.

1.4 Strumenti utilizzati

Nome	Descrizione	Versione
Java	Ambiente e linguaggio di programmazione utilizzato per lo sviluppo	17
JavaFx	Piattaforma per lo sviluppo di applicazioni client, basata su Java	20
IntelliJ IDEA	Software di sviluppo IDE	2023.1.4
org.json.simple	È un formato leggero e di scambio dati basato su testo	1.1.1
org.junit.jupiter:junit-jupiter	framework di test JUnit 5 per Java	level 14
GrandleJVM	Sistema open-source per l'automazione dello sviluppo	17.0.8
ChatGPT	IA basata su un modello di linguaggio sviluppato da OpenAI	3.5

1.5 Descrizione generale del progetto

Nel processo di creazione di un gioco, emerge l'importanza di una pianificazione attenta e metodica. All'inizio, ci si è trovati di fronte alla sfida di suddividere le responsabilità all'interno del team, al fine di adottare un approccio ben strutturato che garantisse il rispetto delle tempistiche di sviluppo e consegna. Questa fase iniziale di organizzazione ha svolto un ruolo cruciale nell'assicurare che ogni componente del progetto fosse adeguatamente assegnato e che

le competenze di ciascun membro del team fossero sfruttate al meglio. Dopo aver concluso con successo la fase di definizione delle specifiche e di progettazione, si è potuto procedere con entusiasmo all'avvio dello sviluppo iniziale dell'applicazione. Questo passaggio ha segnato l'inizio tangibile della trasformazione dell'idea concettuale in una realtà virtuale. Il team si è dedicato all'implementazione delle meccaniche di gioco, all'integrazione delle caratteristiche uniche che definiscono il gioco e alla creazione di un'esperienza coinvolgente per i giocatori. Inoltre, è emersa un'idea guida che ha plasmato l'approccio alla realizzazione delle funzionalità software: l'adozione di una board come base di costruzione. Ogni componente del gioco, dai singoli pezzi con le loro varie configurazioni alle diverse posizioni di ciascun elemento, è stato modellato intorno a questa board. Questo concetto fondamentale ha guidato il team nello sviluppo, garantendo un'architettura solida e coerente.

1.5.1 L'interfaccia e il funzionamento

Per realizzare la GUI sono stati utilizzati la libreria grafica di JavaFX accompagnata da un Json che stabilisce i blocchi anche a livello grafico. I blocchi sono stati implementati tramite l'estensione del componente Rectangle di JavaFX, così da poter gestire la logica dello spostamento di questi ultimi, inoltre sono stati implementate delle funzioni per poter evitare di: sovrapporre i blocchi (checkOverlap), andare oltre la griglia (checkBounds), controllare i limiti (checkRanges) e creare un blocco provvisorio/fantasma per poter fare i vari check. Per completare l'interfaccia sono stati aggiunti dei bottoni, creati su misura. A ciascun bottone è stata associata una funzionalità, ognuna delle quali è stata implementata tramite una logica innescata da eventi scatenati dall'utente compiendo determinate azioni sulla GUI.

1.5.2 Possibilità di salvare lo stato corrente e di ripristinarlo

Premendo l'icona del menu di pausa oppure il tasto ESC si può, attraverso l'apposito bottone salvare la partita. Il comando permette di memorizzare la posizione corrente dei blocchi insieme alla configurazione iniziale. Successivamente dal Main Menu è possibile aprire attraverso il pulsante "load" tutti i salvataggi e poter caricare la configurazione desiderata.

1.5.3 Scegliere fra diverse configurazioni di partenza

Dopo aver premuto il bottone New Game è possibile scegliere tra 6 possibili diverse configurazioni. Le configurazioni iniziali selezionabili sono memorizzate all'interno del codice così che il tutto avvenga in maniera veloce.

1.5.4 Reset

Il programma memorizza la configurazione iniziale dei blocchi prima di iniziare la partita, una volta premuto il bottone apposito resetta la partita visualizzando la configurazione dalla quale si era iniziato a giocare.
NB: che il cambio di configurazione comporta un reset dello storico.

1.5.5 Undo

È possibile tornare indietro di una mossa. Attraverso una struttura dati vengono memorizzate le mosse fatte da inizio partita e, premendo l'apposito bottone, è possibile tornare allo stato precedente del gioco.
NB: viene disabilitato se il programma riconosce la configurazione iniziale.

1.5.6 Next best move

Tramite l'apposito bottone, è possibile fare eseguire al gioco la miglior mossa possibile partendo dallo stato corrente. Partendo da una configurazione già preimpostata precedentemente con degli ID il programma comprende la posizione dei blocchi spostati e capisce quale mossa fare successivamente.

1.5.7 Counter delle mosse

È previsto su game un "counter" per il conteggio delle mosse di una partita, il quale aumenta ogni volta che viene premuto un comando di gioco

1.5.8 Elementi aggiuntivi

Il tutto è stato accompagnato da un menu principale per la selezione della partita, delle grafiche inedite integrate per titoli, schermate di gioco e infine da musica di sottofondo.

1.6 Download

Per scaricare il programma in formato .jar, cliccare sul bottone di download:

Una volta terminato il download del file .zip, estrarne il contenuto in una locazione a piacere all'interno del proprio dispositivo. Sarà presente una directory contenente una cartella e al suo interno un file .jar, per far partire il gioco basta aprire il file. Nota per MacOS: Se, nel tentativo di eseguire il file .jar appare l'avviso "Impossibile eseguire "Klotski.jar" perché proviene da uno sviluppatore non identificato", è possibile eseguirlo manualmente tramite terminale:

```
java -jar Klotski.jar
```

1.7 Comandi di Gioco

Per iniziare o caricare una partita salvata seguire i comandi a schermo del menu principale. Appena inizierà la partita per poter muovere un blocco basterà selezionarlo cliccandolo una volta con il tasto destro e successivamente muoverlo utilizzando le combinazioni: W-A-S-D / frecce direzionali. Inoltre si può accedere al menu di pausa tramite l'apposito comando in alto a sinistra oppure il comando ESC ed infine per richiedere le diverse funzioni specifiche, basterà cliccare i relativi bottoni, in basso a sinistra: undo, reset e next best move.

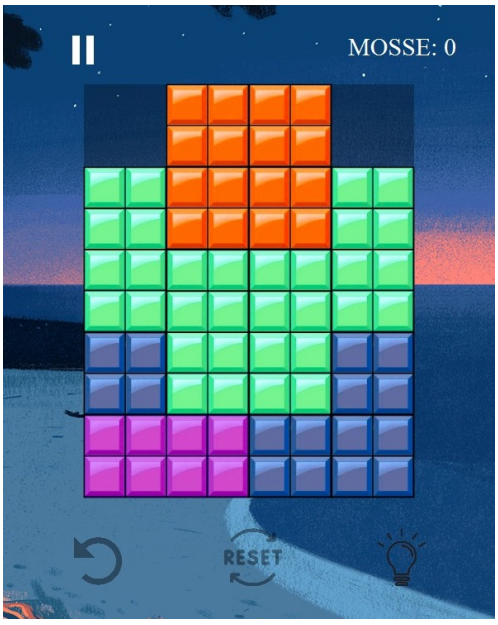

Figura 1.1: menu di gioco

1.8 Design Patterns utilizzati

Un design pattern descrive un problema che si presenta nel nostro ambiente e successivamente descrive una soluzione a tale problema, in modo tale che possa essere utilizzata milioni di volte senza mai farla nel medesimo modo.

1.8.1 Grasp

1. **Controller:** Nel nostro caso abbiamo 1 controller per 2 funzioni specifiche di sistema.

   ```
   Board //controlla tutti i blocchi dalla loro creazione alla loro eliminazione; inoltre si

   occupa del piano di gioco
   ```

2. **Low Coupling:** Nel programma viene usato per la gestione dei Json attraveso il salvataggio delle varie letture dei file per evitare il "coupling", cioè l'impatto che una modifica ad una classe potrebbe avere sull'intero progetto.

3. **High Cohesion:** Assegnando le responsabilità in modo che ogni oggetto faccia solo operazioni specifiche, ed inerenti per poterne aumentare le performance. È stato usato in Block e BlockGFX, dove Block gestisce atomicamente il blocco e BlockGFX gestisce la parte di grafica dei blocchi.

4. **Pure Fabrication:** Assegnando un insieme altamente coeso di responsabilità ad una classe artificiale disegnata da 0, che non rappresenta un concetto nel dominio, nel nostro caso usata con DuplicateMap per la gestione della cronologia.

1.8.2 GoF

1. **Factory:** Definisce un interfaccia per creare un oggetto, ma lascia alla sottoclasse di decidere quale classe va istanziata:

   ```
   Board & Game

   Block & BlockGFX
   ```

2. **Observer:** Stabilisce una dipendenza uno-a-molti tra gli oggetti, in modo che quando uno dei soggetti cambia il suo stato, tutti gli oggetti dipendenti da esso siano notificati e aggiornati automaticamente.

   ```
   Block - BlockGFX
   ```

3. **Facade:** Fornisce un'interfaccia semplificata per un insieme di classi complesse o un sottosistema, nascondendo la complessità e le dettagliate implementazioni dietro questa interfaccia unificata. Nel nostro caso è stato utilizzato per la lettura e scrittura dei dati su diversi file Json:

   ```
   JsonInterface, JsonConfigurationReader, JsonSolutionReader
   ```

Capitolo 2

Specifiche

2.1 Use Cases

La scelta del come gestire l'applicazione è stata basata su quattro scene principali che aiutano a definire il flusso di interazione dell'utente attraverso i vari menu e azioni disponibili in ognuna di esse:

1. **Scena #1** - Menu iniziale
2. **Scena #2** - Menu di selezione partita
3. **Scena #3** - Interfaccia di gioco
4. **Scena #4** - Menu di pausa

Questi casi d'uso possono essere utilizzati come base per la progettazione delle interfacce utente e per comprendere come il giocatore naviga attraverso i diversi menu nel gioco.

2.2 Use Case diagram

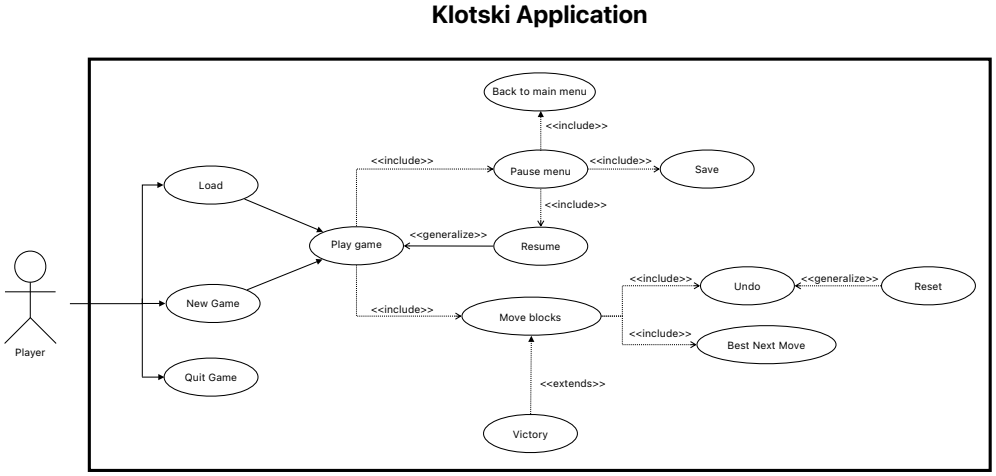

Figura 2.1: Use Case diagram

2.3 Descrizione Tabellare dei Use Cases

2.3.1 Use Case #1

Nome Use Case	New Game
Attori	Player
Descrizione	Il player fa partire una nuova partita
Precondizioni	Nessuna
Flow	Il player clicca sul bottone "New Game" nella scena #1 Il player sceglie la configurazione iniziale nella scena #2 Il player esegue lo use case #4
Post condizioni	Il player si trova nella Scena #3 con una nuova partita

2.3.2 Use Case #2

Nome Use Case	Load
Attori	Player
Descrizione	Il player fa partire una partita salvata
Precondizioni	lo use case #9 deve essere stato eseguito almeno una volta
Flow	Il player clicca sul bottone "Load Game" nella scena #1 Il player sceglie la partita salvata nella scena #2 Il player esegue l'use case #4
Post condizioni	Il player si trova nella Scena #3 con la partita salvata

2.3.3 Use Case #3

Nome Use Case	Quit Game
Attori	Player
Descrizione	Il player chiude la finestra di gioco
Precondizioni	Nessuna
Flow	Il player clicca sul bottone "Quit Game" nella scena #1 Il player esce dalla scena #1
Post condizioni	Nessuna

2.3.4 Use Case #4

Nome Use Case	Play Game
Attori	Player
Descrizione	Il player può giocare al gioco secondo le regole di klotski e risolvere il puzzle
Precondizioni	Use case #1 oppure #2
Flow	Il player può muovere un blocco e passare allo use case #8 oppure #11 Il player clicca l'icona "pause menu" ed esegue lo use case #5 passando alla Scena#4
Post condizioni	Il player è nella scena #4 e visualizza il menu di pausa, oppure l'utente esegue lo use case #9

2.3.5 Use Case #5

Nome Use Case	Pause Menu
Attori	Player
Descrizione	Il player può selezionare le opzioni nel menu di pausa
Precondizioni	lo use case #4 deve essere stato eseguito
Flow	Il player clicca sull'icona del menu di pausa Il player clicca sul bottone "Resume" nella scena #4 eseguendo use case #7 Il player clicca sul bottone "Save" nella scena #4 eseguendo use case #6 Il player clicca sul bottone "Main Menu" nella scena #4 eseguendo use case #8
Post condizioni	Il player rimane nella scena #3

2.3.6 Use Case #6

Nome Use Case	Save
Attori	Player
Descrizione	Il player può salvare la partita corrente
Precondizioni	Use case #5
Flow	Il player clicca il bottone "Save" nella scena #4
Post condizioni	Nessuna

2.3.7 Use Case #7

Nome Use Case	Resume
Attori	Player
Descrizione	Il player può tornare alla partita
Precondizioni	Use case #5
Flow	Il player clicca il bottone "Resume" nella scena #4
Post condizioni	Il player torna nella scena #3

2.3.8 Use Case #8

Nome Use Case	Back to Main Menu
Attori	Player
Descrizione	Il player può tornare al menu principale
Precondizioni	Use case #5
Flow	Il player clicca il bottone "Main menu" nella scena #4
Post condizioni	Il player torna nella scena #1

2.3.9 Use Case #9

Nome Use Case	Move Blocks
Attori	Player
Descrizione	Il player può fare una mossa
Precondizioni	lo use case #4 deve essere stato eseguito
Flow	Il player clicca un blocco e lo muove nella scena #3 Il player può eseguire use case #10 #11 #12
Post condizioni	Il player esegue lo use case #13 in caso di vittoria

2.3.10 Use Case #10

Nome Use Case	Undo
Attori	Player
Descrizione	Il player può annullare la mossa precedente
Precondizioni	Use case #9
Flow	Il player clicca l'icona "Undo" Il player può muovere un blocco e passare allo use case #9
Post condizioni	Tornare alla mossa precedente

2.3.11 Use Case #11

Nome Use Case	Reset
Attori	Player
Descrizione	Il player può tornare allo stato iniziale della partita
Precondizioni	Use case #9
Flow	Il player clicca l'icona "Reset" Il player può muovere un blocco e passare allo use case #9
Post condizioni	Ripristinare lo stato iniziale della partita

2.3.12 Use Case #12

Nome Use Case	Next Best Move
Attori	Player
Descrizione	Il player avanza di una mossa
Precondizioni	Use case #9
Flow	Il player clicca l'icona "Next Best Move" Il player può muovere un blocco e passare allo use case #9
Post condizioni	Nessuna

2.3.13 Use Case #13

Nome Use Case	Victory
Attori	Player
Descrizione	Il player visualizza lo stato di vittoria
Precondizioni	Use case #9
Flow	Al player viene segnalato che ha vinto.
Post condizioni	Il player torna nella scena #1

Capitolo 3

Design

3.1 Domain Model

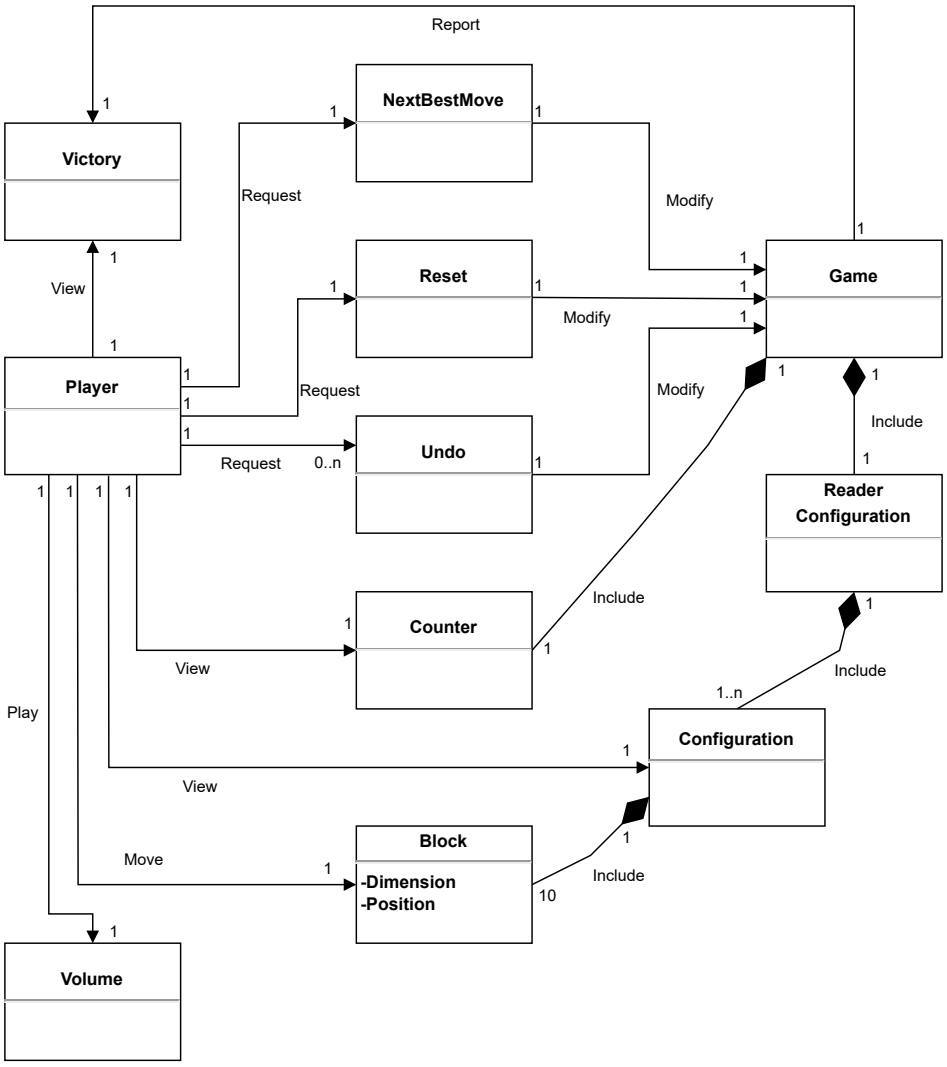

Figura 3.1: Domain model diagram

3.2 Design Class diagram

Il design class diagram fornisce tutte le informazioni della progettazione. Il team si è concentrato sullo sviluppo di tutte le entità software. Successivamente vengono divisi nei vari pacchetti e sottoclassi.

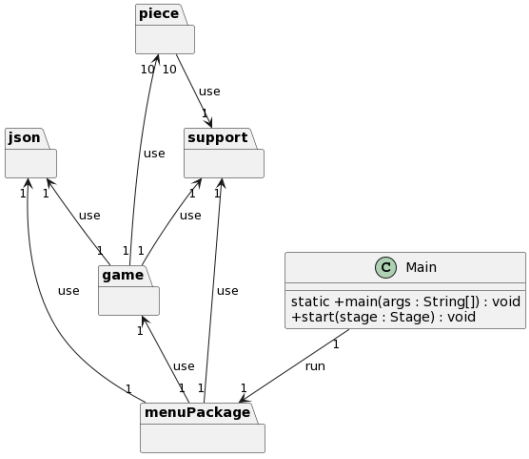

Figura 3.2: Design class diagram riassuntivo

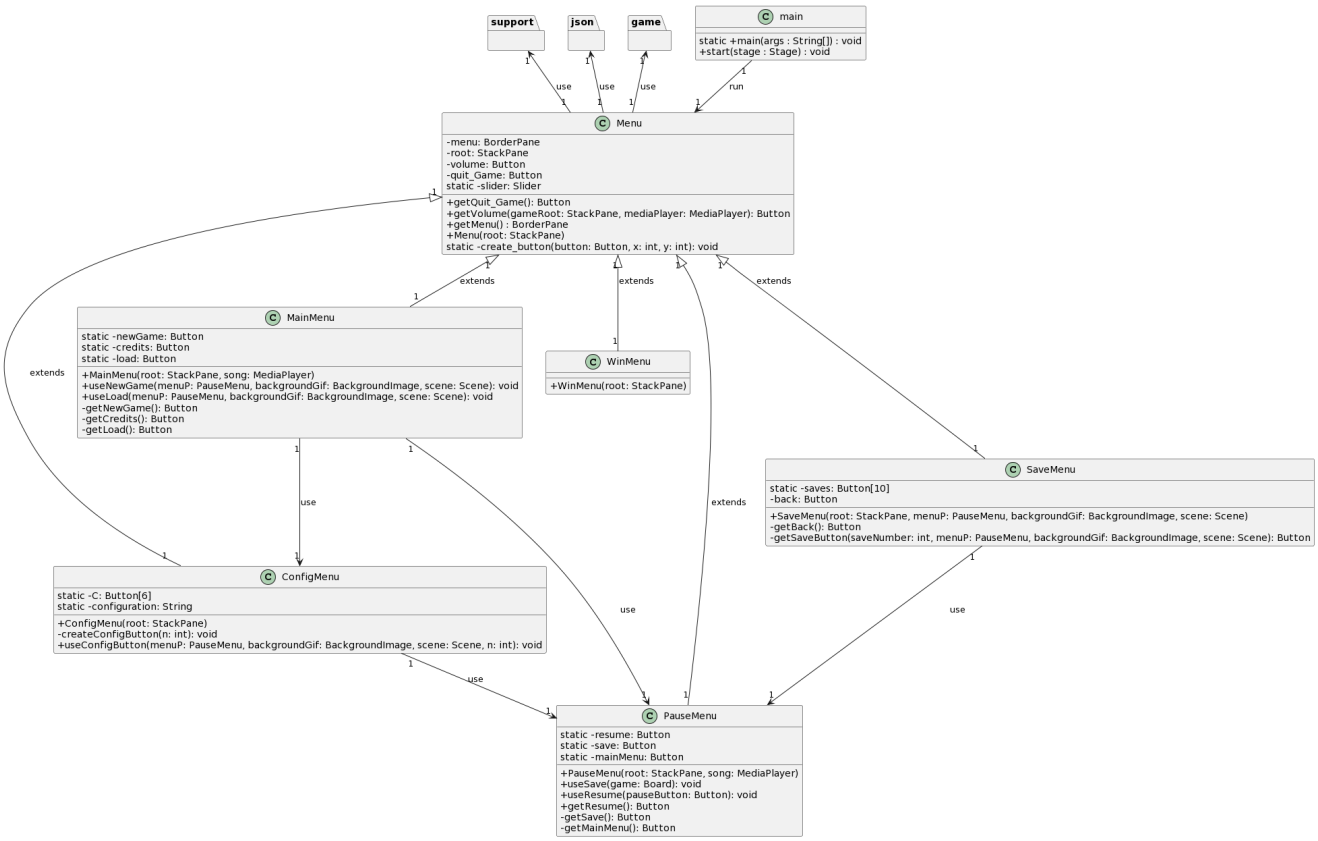

Figura 3.3: Design class diagram di 'menuPackage'

Figura 3.4: Design class diagram del pacchetto 'game'

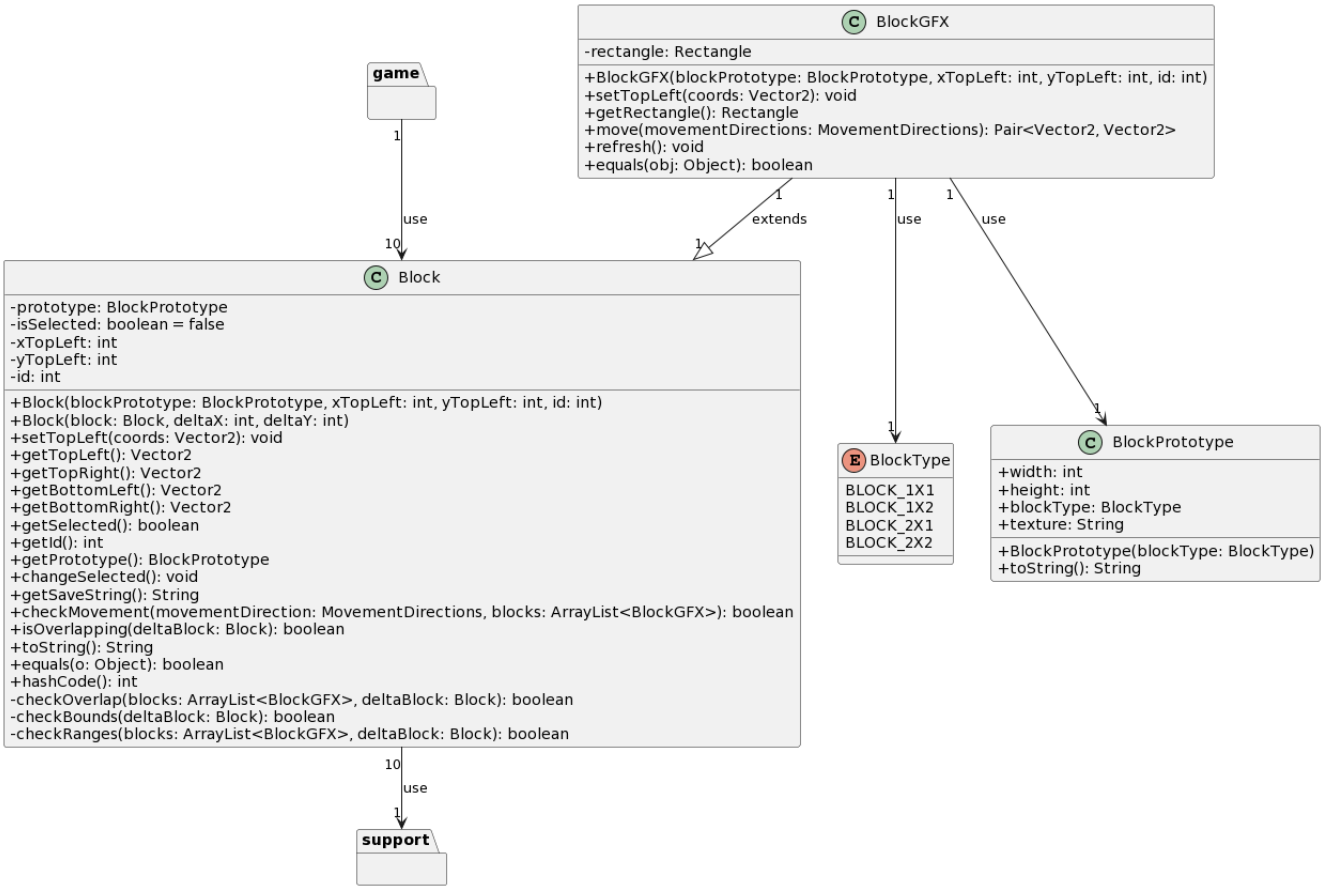

Figura 3.5: Design class diagram del pacchetto 'piece'

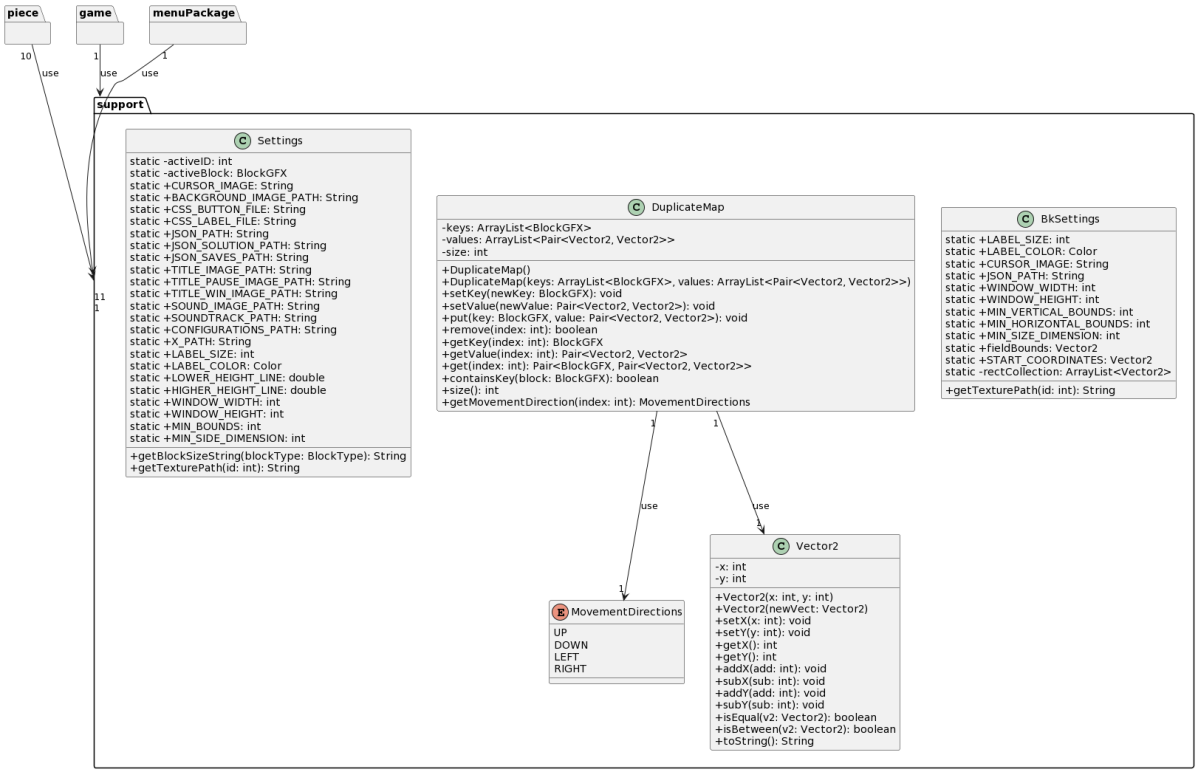

Figura 3.6: Design class diagram del pacchetto 'support'

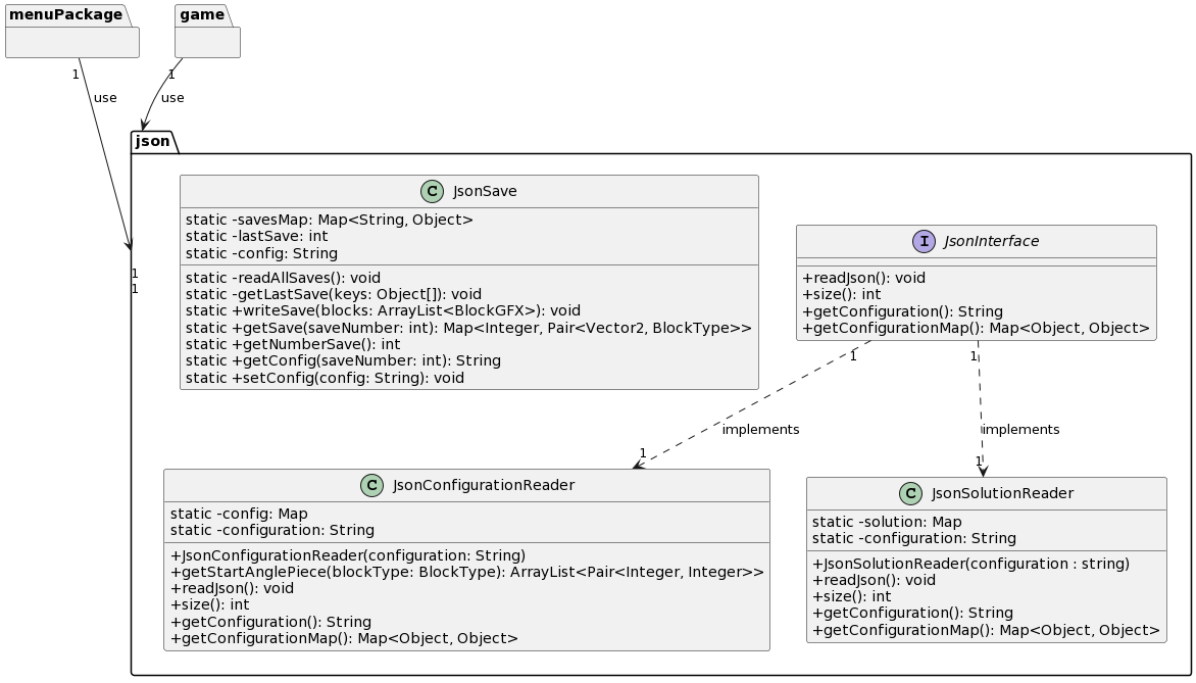

Figura 3.7: Design class diagram del pacchetto 'json'

3.3 Sequence Diagram

Il system diagram illustra le relazioni, le connessioni e le interazioni tra i componenti del sistema.

3.3.1 System Sequence Diagram : Start a game

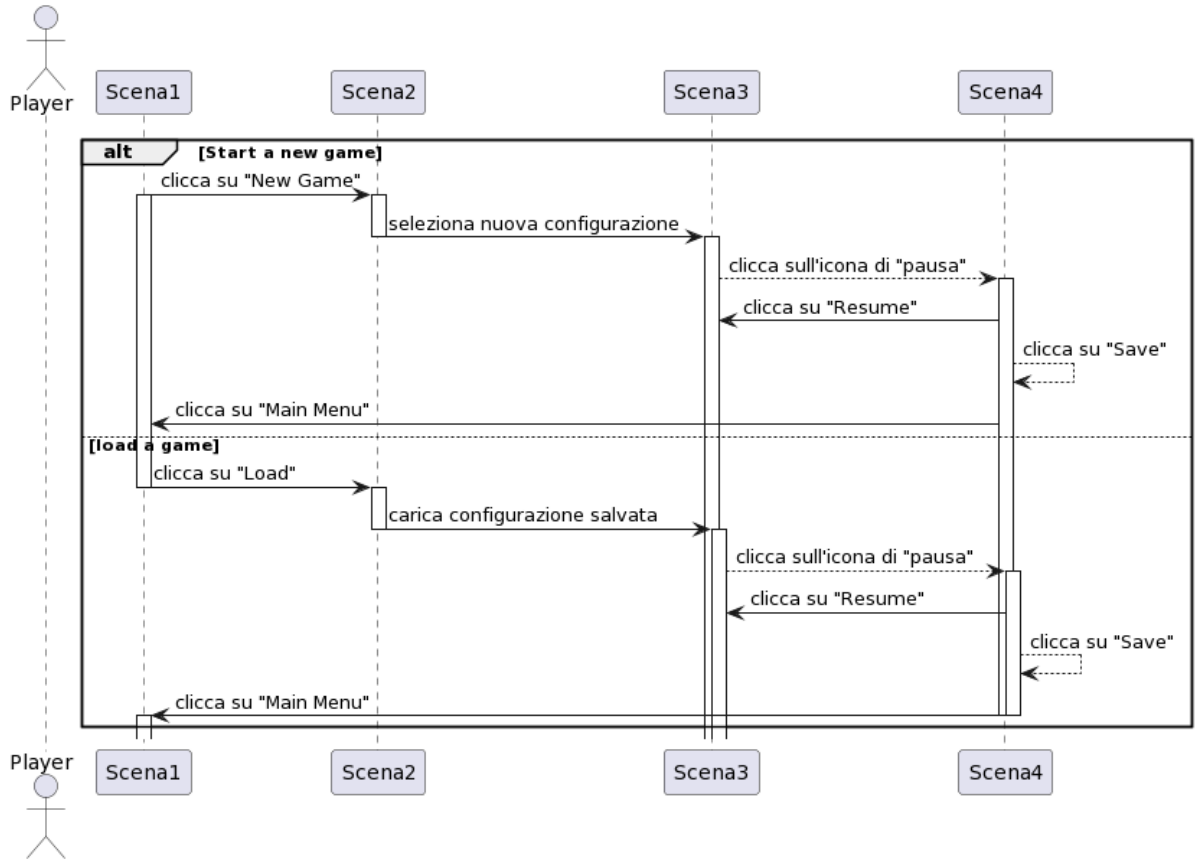

Figura 3.8: Start a game

3.3.2 Internal Sequence Diagram : Start a game

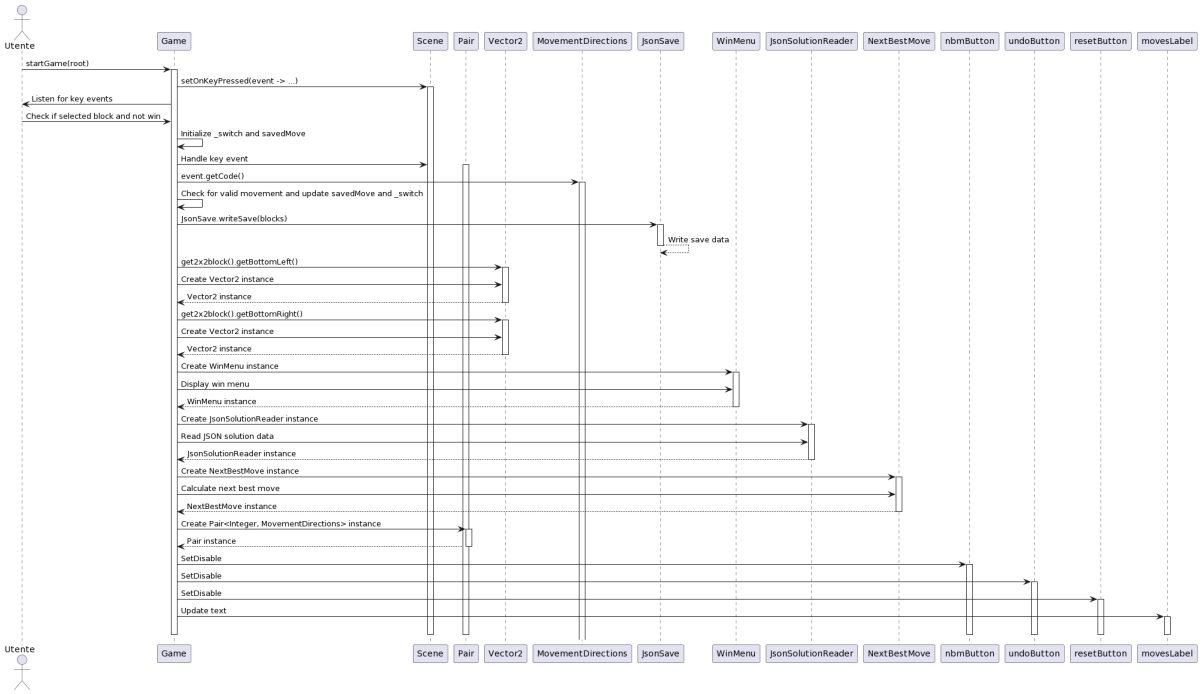

Figura 3.9: Internal - Start a game

3.3.3 System Sequence Diagram : Play a game

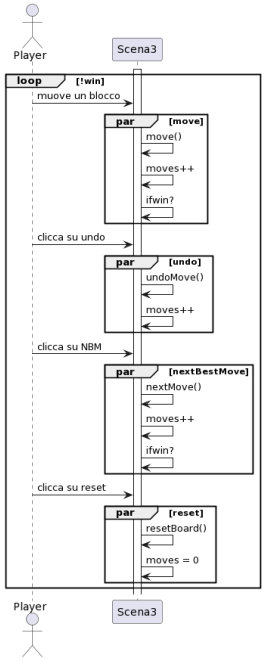

Figura 3.10: Play a game

3.3.4 Internal Sequence Diagram : move

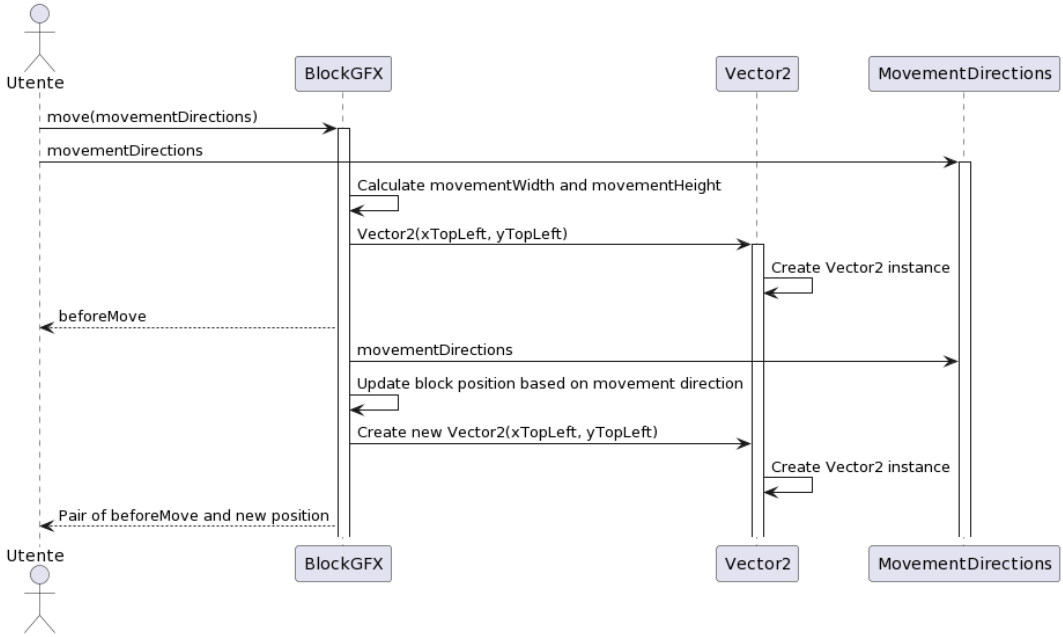

Figura 3.11: Internal : move

3.3.5 Internal Sequence Diagram : undo

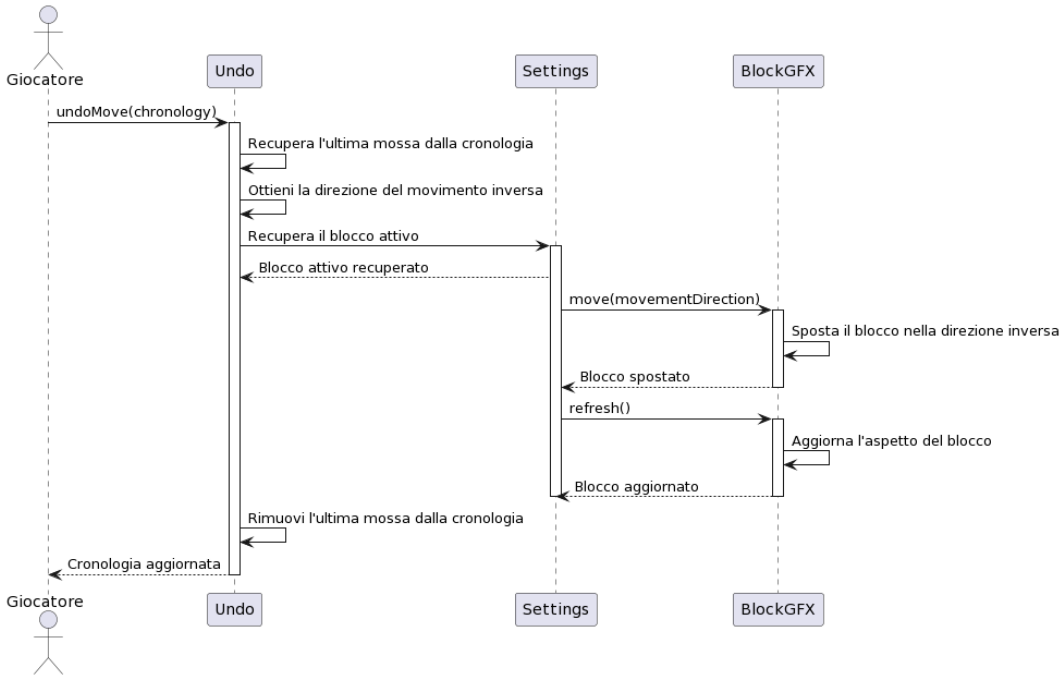

Figura 3.12: Internal : undo

3.3.6 Internal Sequence Diagram : reset

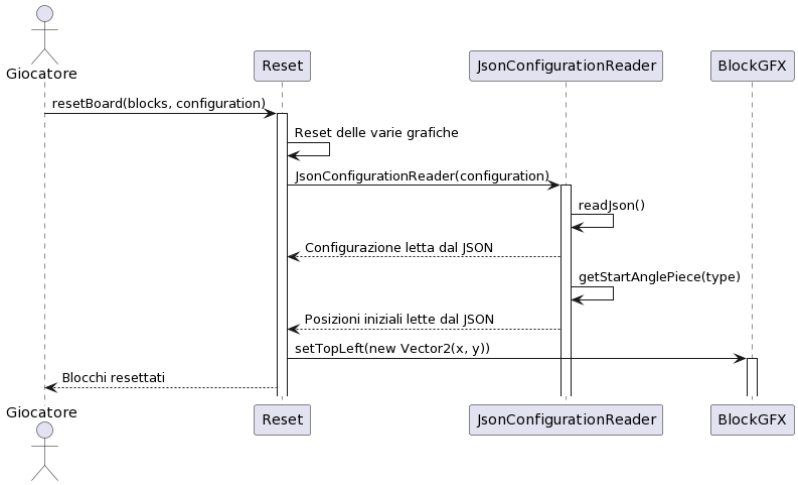

Figura 3.13: Internal : reset

3.3.7 Internal Sequence Diagram : next best move

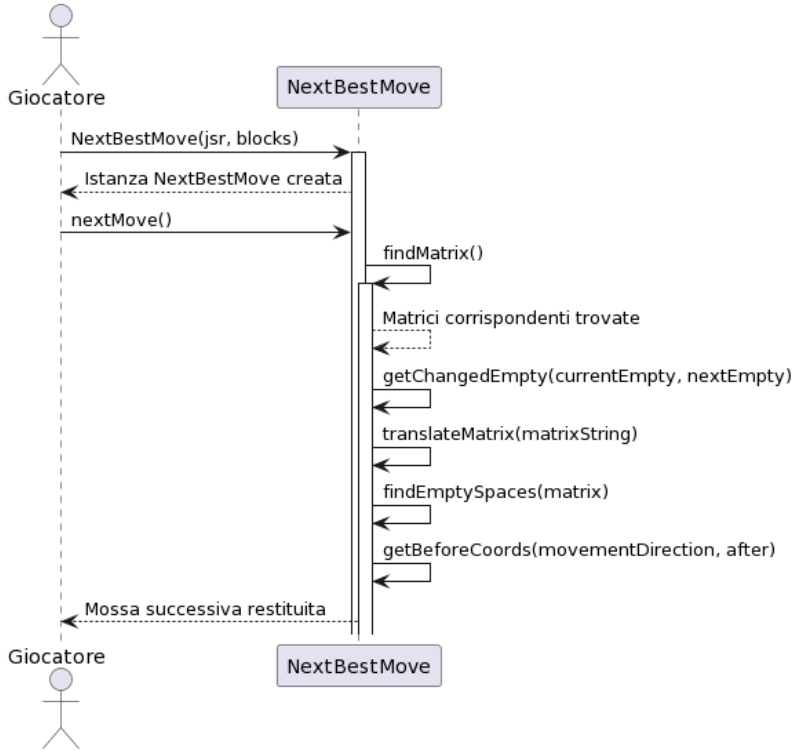

Figura 3.14: Internal : next best move

Capitolo 4

System Test

4.1 System Test Case

Figura 4.1: Immagine cliccabile

4.2 Unit Test Report

Figura 4.2: Immagine cliccabile

Capitolo 5

JavaDoc

Figura 5.1: link al javaDoc

Capitolo 6

Crediti

Figura 6.1: link alla pagina GIT

Figura 6.2: logo del gioco progettato dal gruppo e contenente un Easter-egg