“Knight's treasures”

Carmine Varipapa

Fabio Gruppioni

Federico Di Franco

Matteo Sacchetti

13 febbraio 2025

Indice

Capitolo 1

Analisi

1.1 Requisiti

Il progetto a noi commissionato, si pone come obiettivo la realizzazione di un videogioco ispirato a "Super Mario Bros". Si tratta di un platform 2D a livelli, in cui un personaggio deve completare percorsi raccogliendo oggetti, eliminando nemici e cercando di non essere colpito.

Funzionalità minimali ritenute obbligatorie:

- Creazione del menù principale con tasti "Play" (avvio partita), "Quit" (uscita dal gioco), "Settings" (impostazioni di gioco) e una legenda per i comandi.
- Gestione della partita, compresa l'inizializzazione e la condizione di fine partita (vittoria o sconfitta).
- Gestione delle entità fondamentali del gioco: player (personaggio principale), nemici e oggetti collezionabili.
- Implementazione delle collisioni tra entità (player-nemici, player-ostacoli, player-oggetti).
- Input del giocatore: movimento del personaggio tramite tastiera (camminata, salto, ecc.).
- Sviluppo grafico del gioco e della View, con utilizzo di sprites e animazioni.

Funzionalità opzionali

- Selezione del personaggio ad inizio partita.
- Menù pausa con opzioni per riprendere, ricominciare o uscire dalla partita.
- Animazioni di introduzione e conclusione per ogni partita.
- Gestione pausa, ripresa del livello, ricominciare il livello e ritorno al menù durante la partita.
- Gestione del volume durante il gioco (volume on o off).

1.2 Analisi e modello del dominio

Il gruppo si propone di creare un videogioco in stile "Super Mario Bros", un platform 2D che si sviluppa su più livelli. L'obiettivo del gioco è quello di guidare un personaggio attraverso vari percorsi, dove dovrà raccogliere oggetti e affrontare nemici che popolano l'ambiente di gioco. Il giocatore dovrà utilizzare abilità come il salto, la corsa e l'attacco per superare ostacoli e sconfiggere i nemici, cercando di non essere colpito. Ogni livello offrirà una serie di difficoltà crescenti, ostacoli ambientali da evitare e percorsi più intricati da navigare. Il design artistico sarà ispirato ai classici medievali, con ambientazioni colorate e una colonna sonora vivace che accompagnerà il giocatore durante la sua avventura.

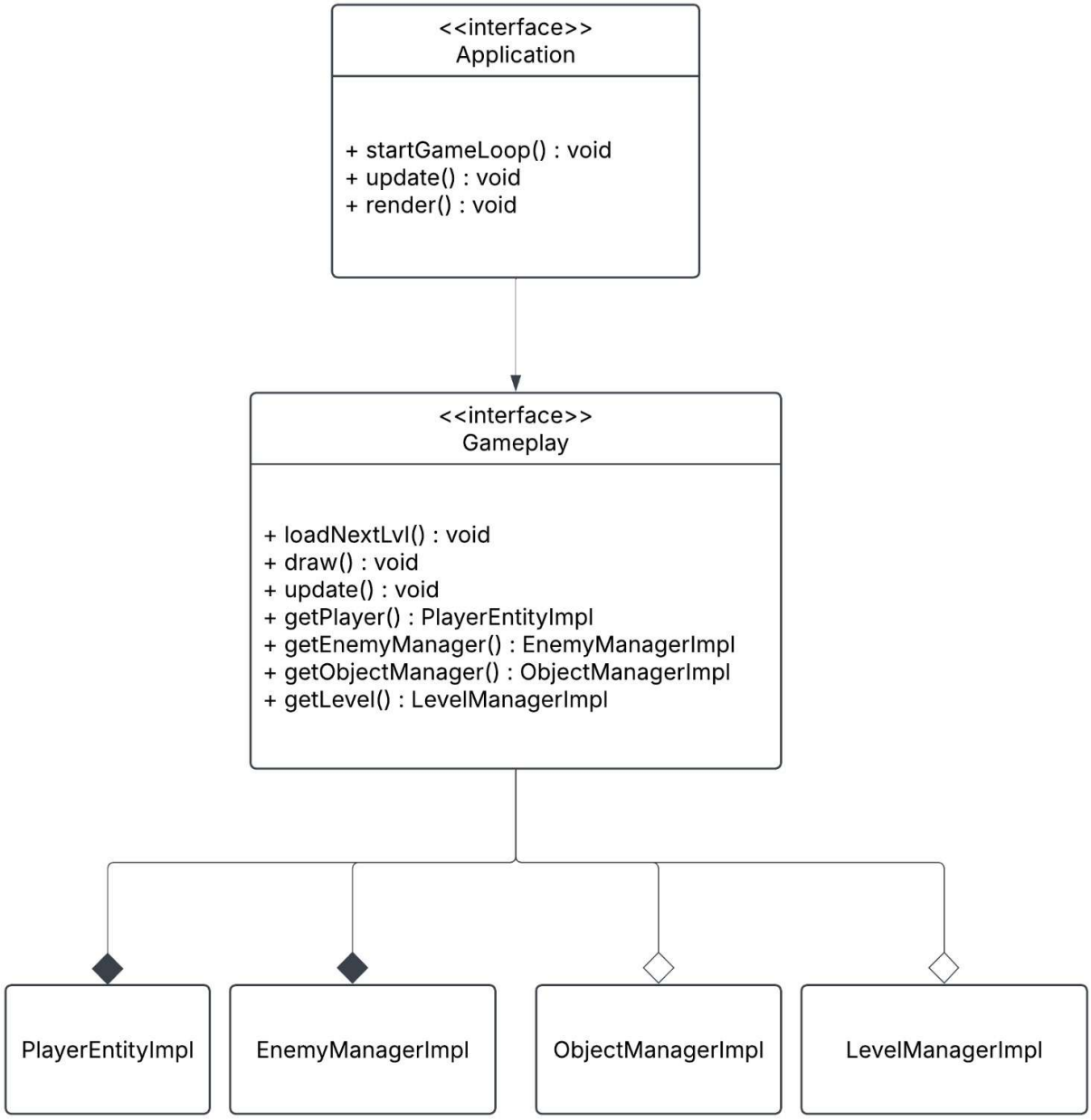

Figura 1.1: Schema UML che rappresenta le principali entità e i rapporti fra loro.

Capitolo 2

Design

2.1 Architettura

Per la realizzazione del progetto, ci siamo ispirati al pattern architetturale MVC (Model-View-Controller), cercando di adattarlo alle specifiche del nostro sistema in modo da renderlo il più simile possibile. In questo approccio, ogni classe, come ad esempio quelle relative al giocatore, agli oggetti o ai livelli, è stata progettata per contenere al suo interno sia la logica che la parte grafica, cioè la gestione delle funzionalità e la visualizzazione. Questo significa che, quando vengono inizializzati il menu o il gameplay, le classi vengono passate già pronte e complete, con tutte le funzionalità e gli aggiornamenti grafici implementati direttamente al loro interno. In altre parole, anziché separare esplicitamente il model, la view e il controller come in una tradizionale architettura MVC, abbiamo scelto di unire tutte queste componenti all'interno di ciascuna singola classe. Ad esempio, nel contesto del gameplay, tutte le classi relative al gioco vengono inizializzate e gestite direttamente, mentre la classe ApplicationImpl, che funge da controller principale, si occupa della gestione del ciclo di gioco (game loop), del rendering e degli stati del gioco, mantenendo così una struttura integrata e semplificata.

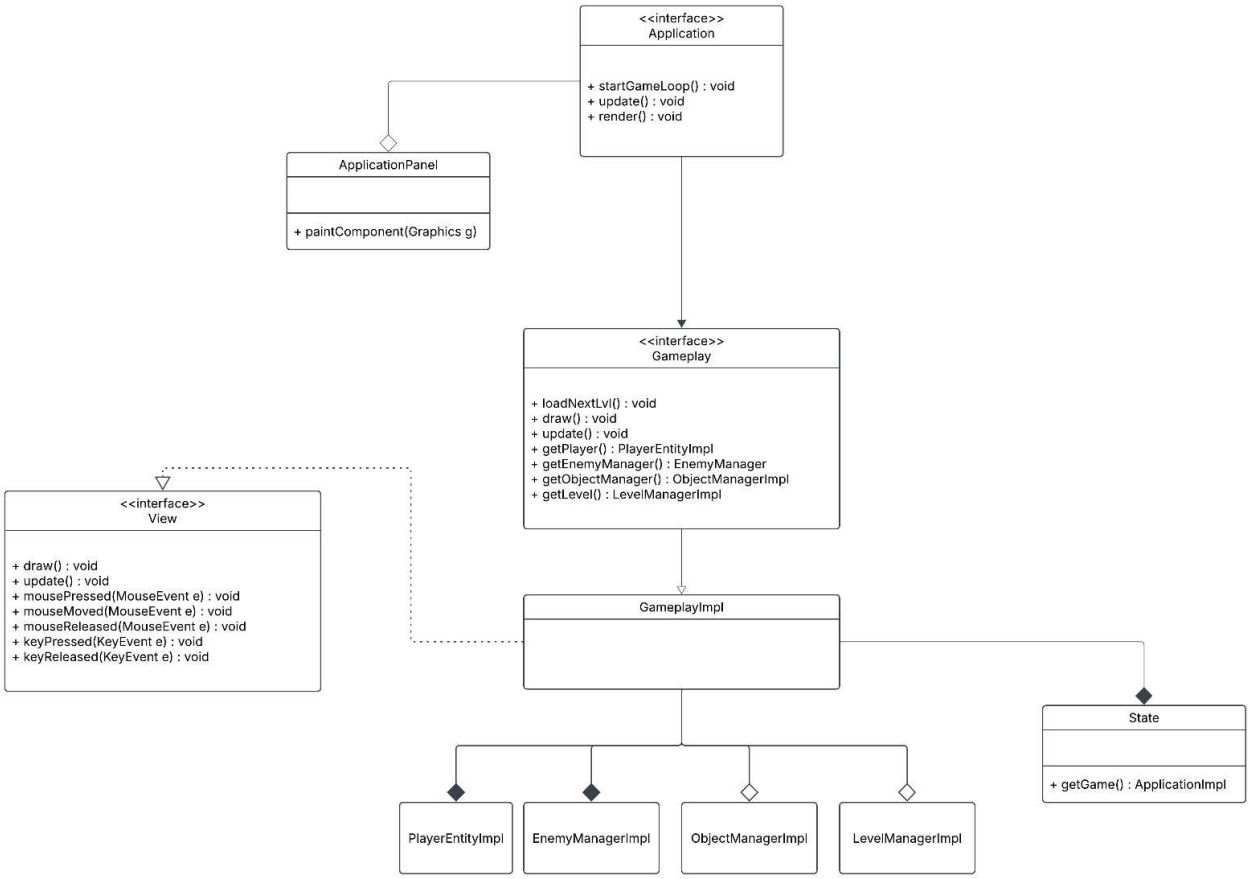

Figura 2.1: Schema UML architetturale di Knight's treasures.

2.2 Design Dettagliato

2.2.1 Carmine Varipapa

Gestione del menù

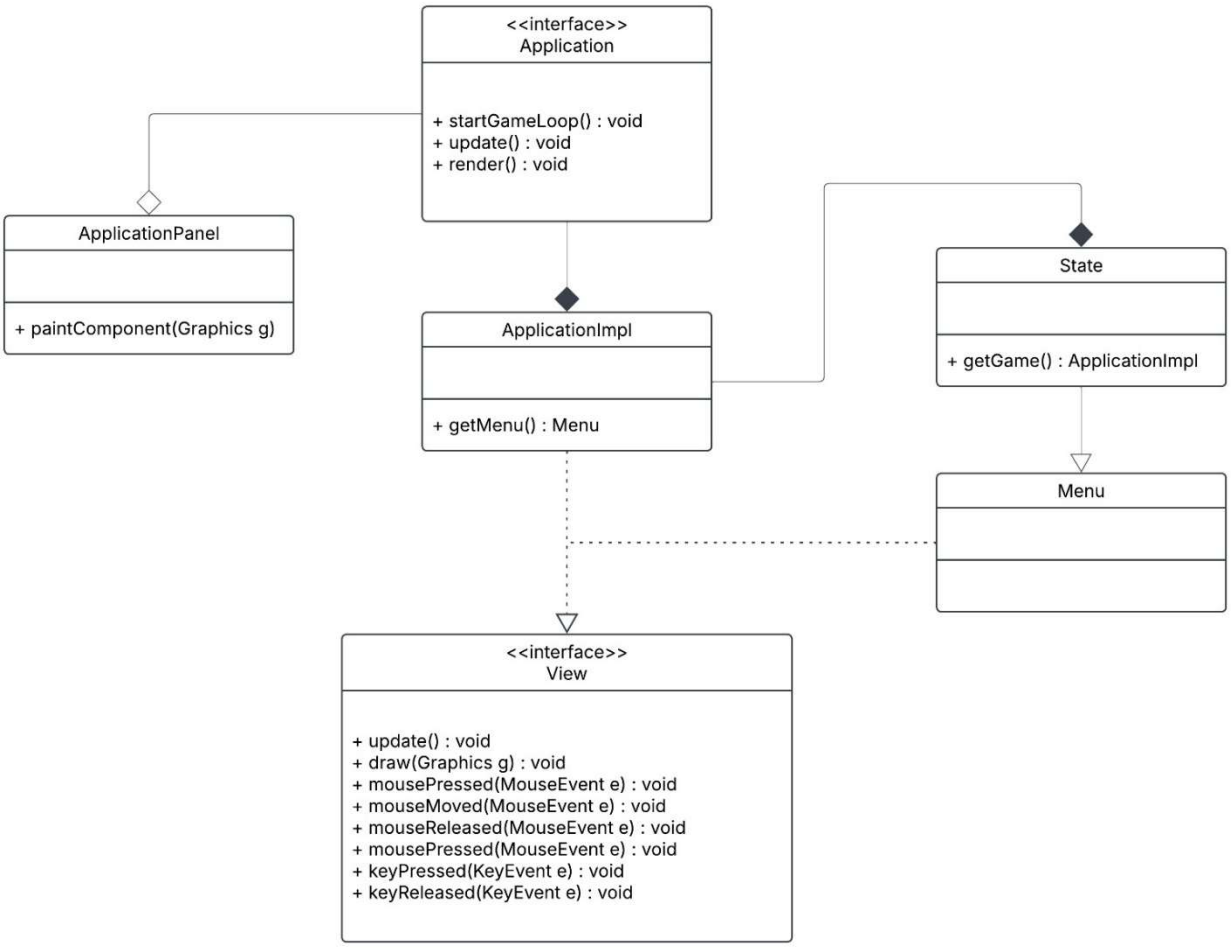

Figura 2.2: Rappresentazione UML della gestione del menu.

Problema: Il gioco possiede un menu iniziale dove è possibile avviare una partita, modificare le impostazioni, o uscire dal gioco.

Soluzione: Quando viene chiamato il metodo draw() all'interno della view, vengono disegnati tutti gli elementi grafici relativi all'attuale stato del gioco. Ogni volta che si riceve un input, viene aggiornato lo stato di gioco. Se il bottone premuto è quello etichettato come “Gioca”, l'azione avvierà una nuova partita, partendo dal primo livello, considerato come il livello iniziale.

Gestione delle settings

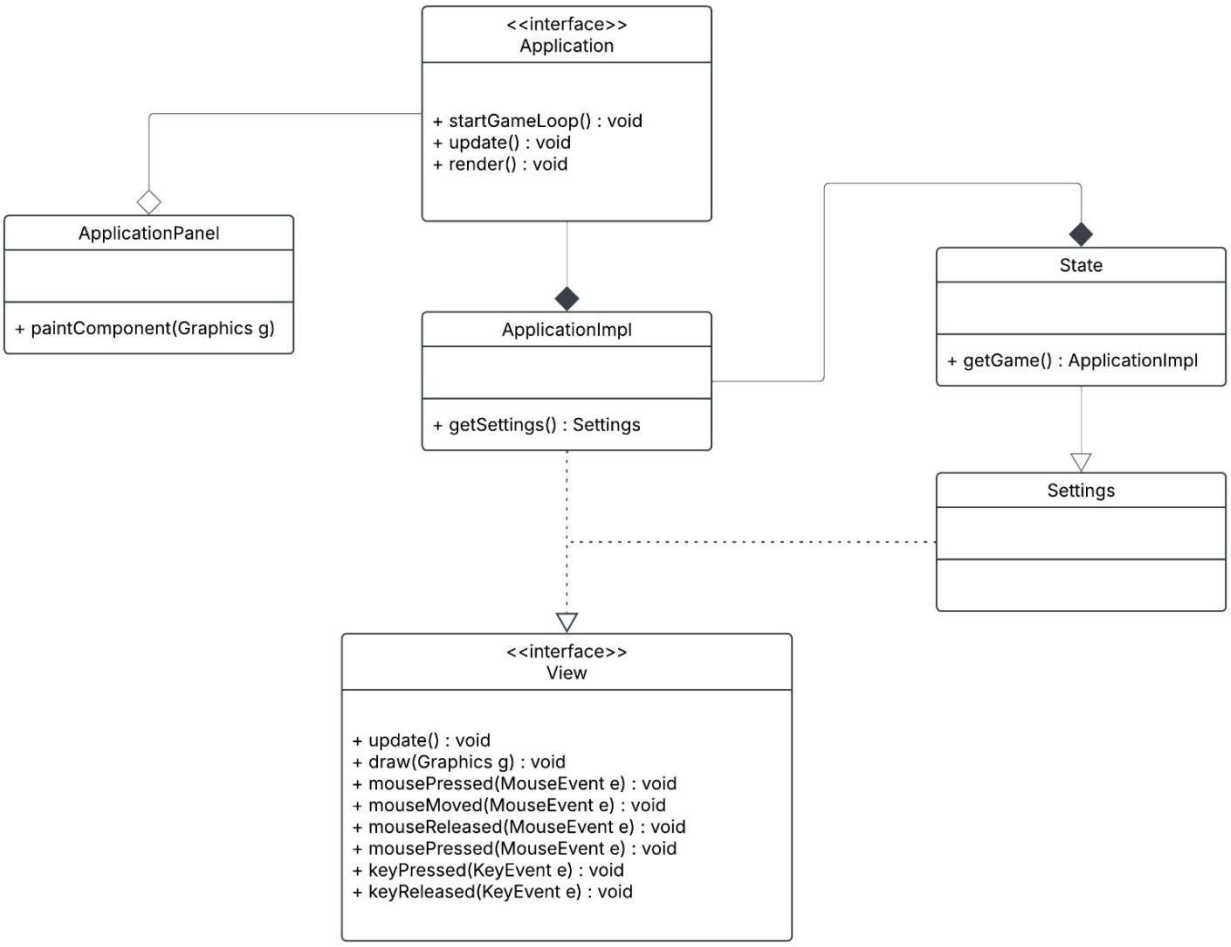

Figura 2.3: Rappresentazione UML della gestione per la schermata di settings.

Problema: Quando ci ritroviamo nella schermata delle impostazioni è possibile tornare al menu, modificare l’audio disattivandolo o attivandolo.

Soluzione: Similmente al menù, l'unica differenza principale è che non sarà possibile avviare una nuova partita. Tuttavia, sarà sempre possibile tornare al menù principale tramite il tasto "Home". Quando uno dei pulsanti di gestione audio verrà premuto, verrà eseguita l'azione corrispondente.

Gestione della schermata finale, di pausa e di completamento livello

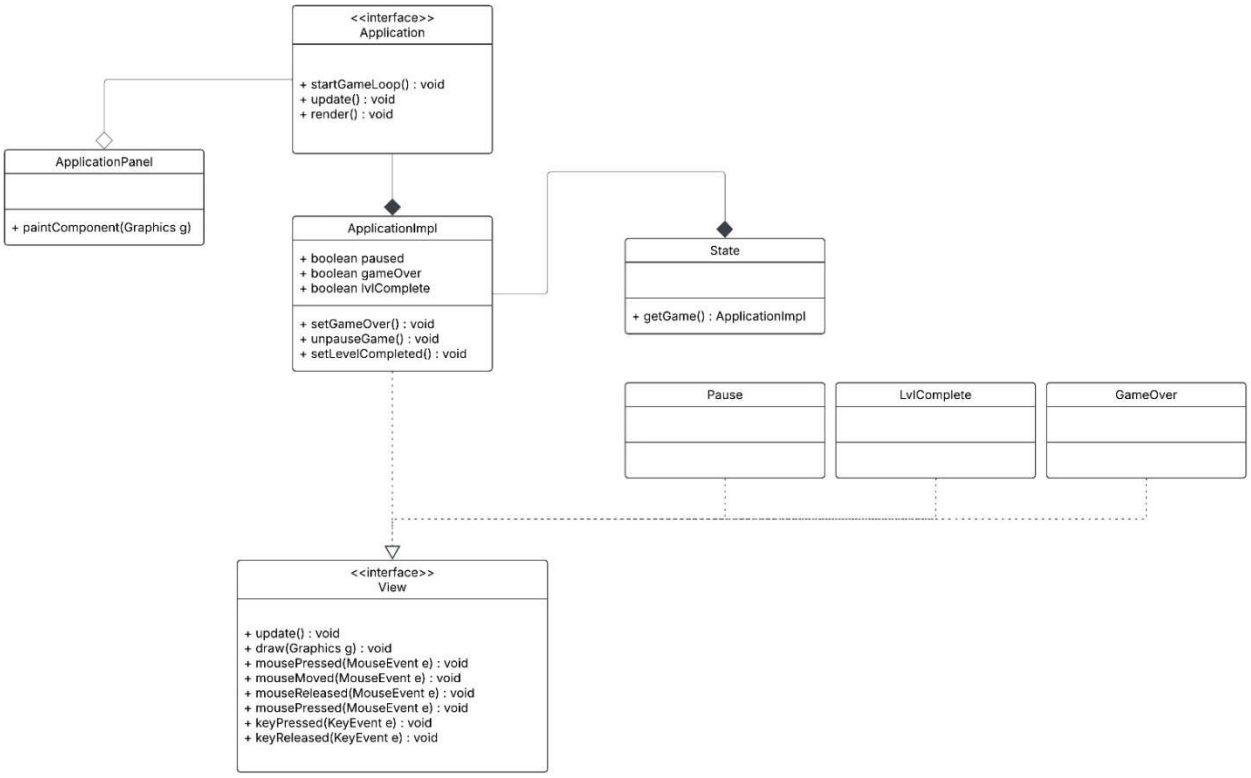

Figura 2.4: Rappresentazione UML per la gestione della schermata finale, di pausa e di completamento livello.

Problema: Nella schermata di pausa, è possibile riprendere il gioco, tornare al menù, ricominciare il livello oppure attivare o disattivare l'audio. Nella schermata di sconfitta, invece, si può ripartire dal livello corrente o tornare al menù. In caso di vittoria, si ha la possibilità di tornare al menù principale o proseguire con il livello successivo.

Soluzione: Il funzionamento del pulsante "Menù" è simile nelle diverse schermate, ma durante il gioco, la visualizzazione dei pannelli di pausa, sconfitta e livello completato viene gestita direttamente nel gameplay tramite variabili booleane. Nella schermata di pausa, le impostazioni audio sono le stesse presenti nella schermata "Settings".

Gestione del game loop

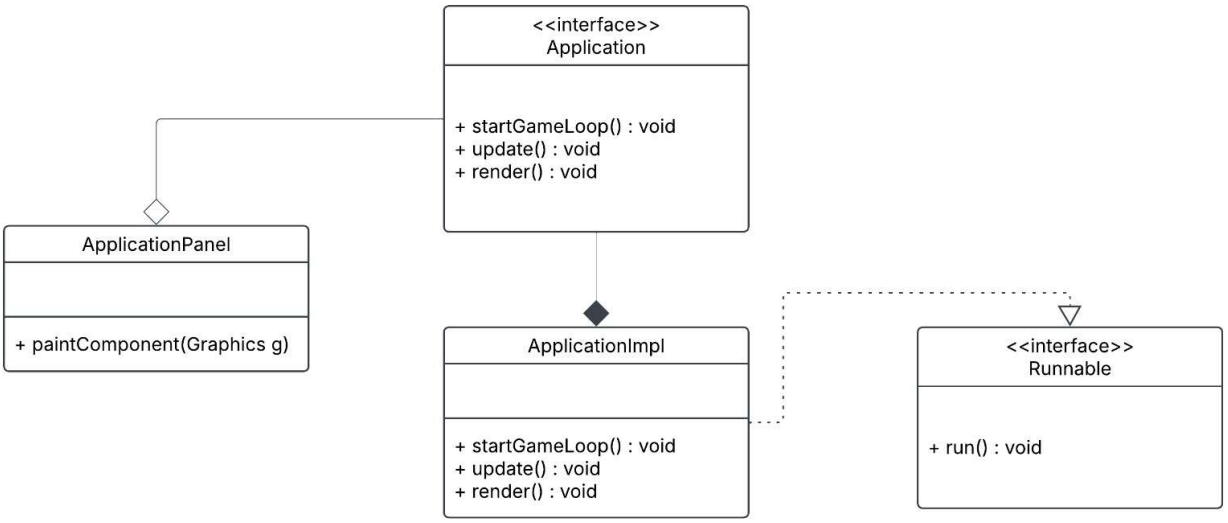

Figura 2.5: Rappresentazione UML del game loop.

Problema: Il gioco deve essere in grado di permettere un gameplay fluido senza lag, aggiornando ciclicamente la grafica e la logica di gioco in contemporanea.

Soluzione: La gestione del game loop avviene direttamente nel file principale, denominato “ApplicationImpl”. In questo file, il ciclo di gioco è controllato e aggiornato in modo continuo, garantendo il corretto flusso di esecuzione delle varie fasi del gioco. Questo approccio centralizza il controllo del gioco, permettendo una gestione più semplice e coerente delle diverse dinamiche che compongono l'esperienza di gioco.

2.2.2 Fabio Gruppioni

Implementazione dei nemici e degli oggetti collezionabili

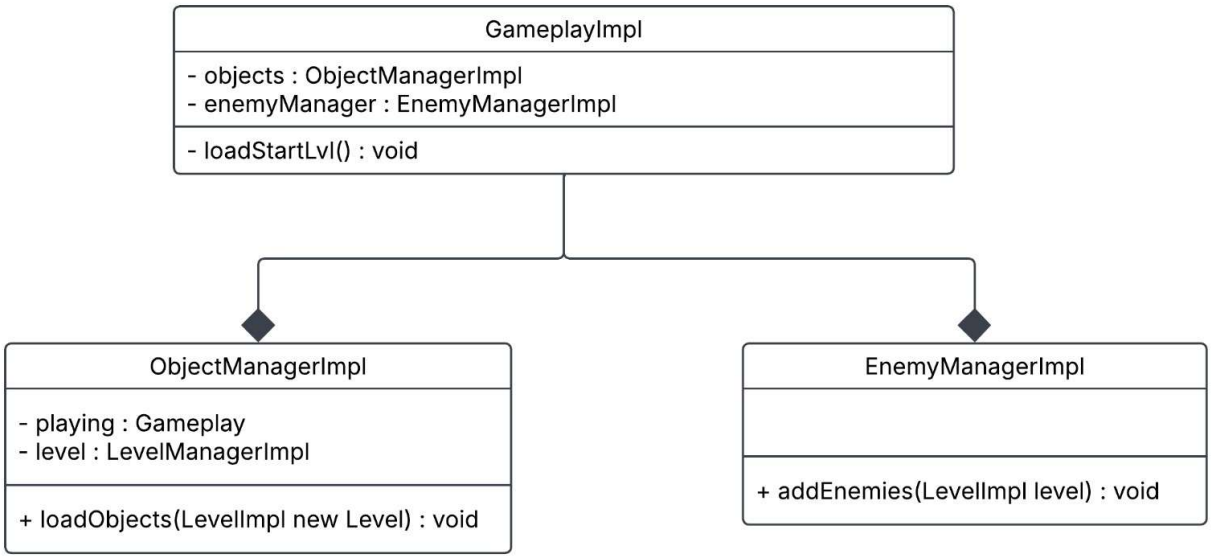

Figura 2.6: Rappresentazione UML dell'implementazione degli oggetti e dei nemici.

Problema: Implementare, a seconda del livello, la posizione dei nemici e degli oggetti collezionabili.

Soluzione: Nel file principale, "GameplayImpl", vengono richiamati dei metodi per recuperare tutti gli oggetti e i nemici (loadStarLvl). Nelle classi che gestiscono separatamente oggetti e nemici, vengono restituiti tutti gli elementi, le cui posizioni sono già preimpostate in base all'immagine del livello. In questo modo, ogni oggetto e nemico è correttamente posizionato secondo la disposizione prevista per il livello.

Implementazione delle entità

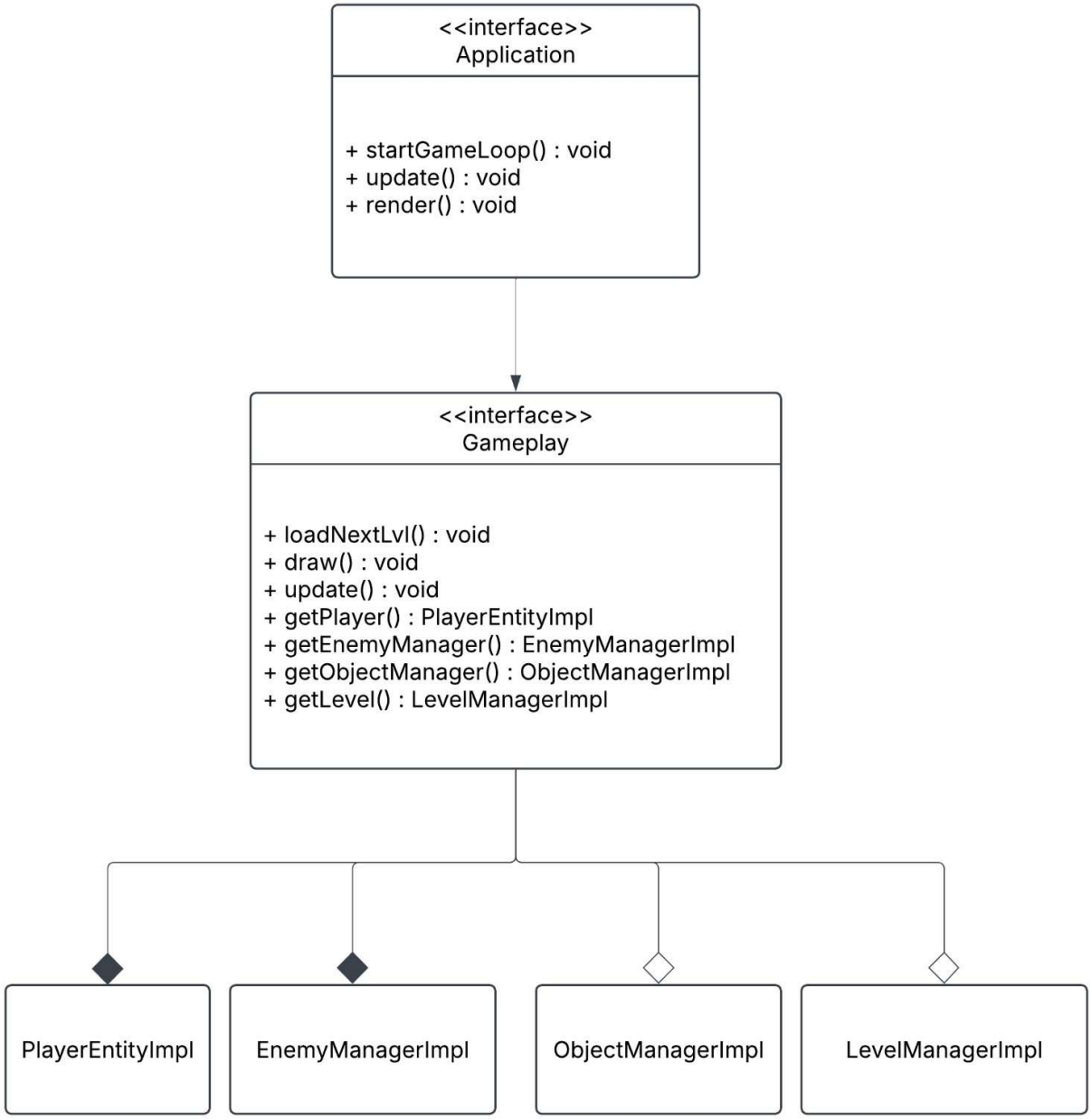

Figura 2.7: Rappresentazione UML dell'implementazione delle entità.

Problema: Ogni singola entità deve essere implementata corettamente nel file principale.

Soluzione: Ogni entità viene creata in modo indipendente all'interno della propria classe e successivamente inizializzata nel file principale "GameplayImpl". In questo modo, quando il gioco viene avviato, tutte le entità del livello corrente vengono caricate correttamente, pronte per essere utilizzate nel gioco.

Creazione delle animazioni per il giocatore ed i nemici

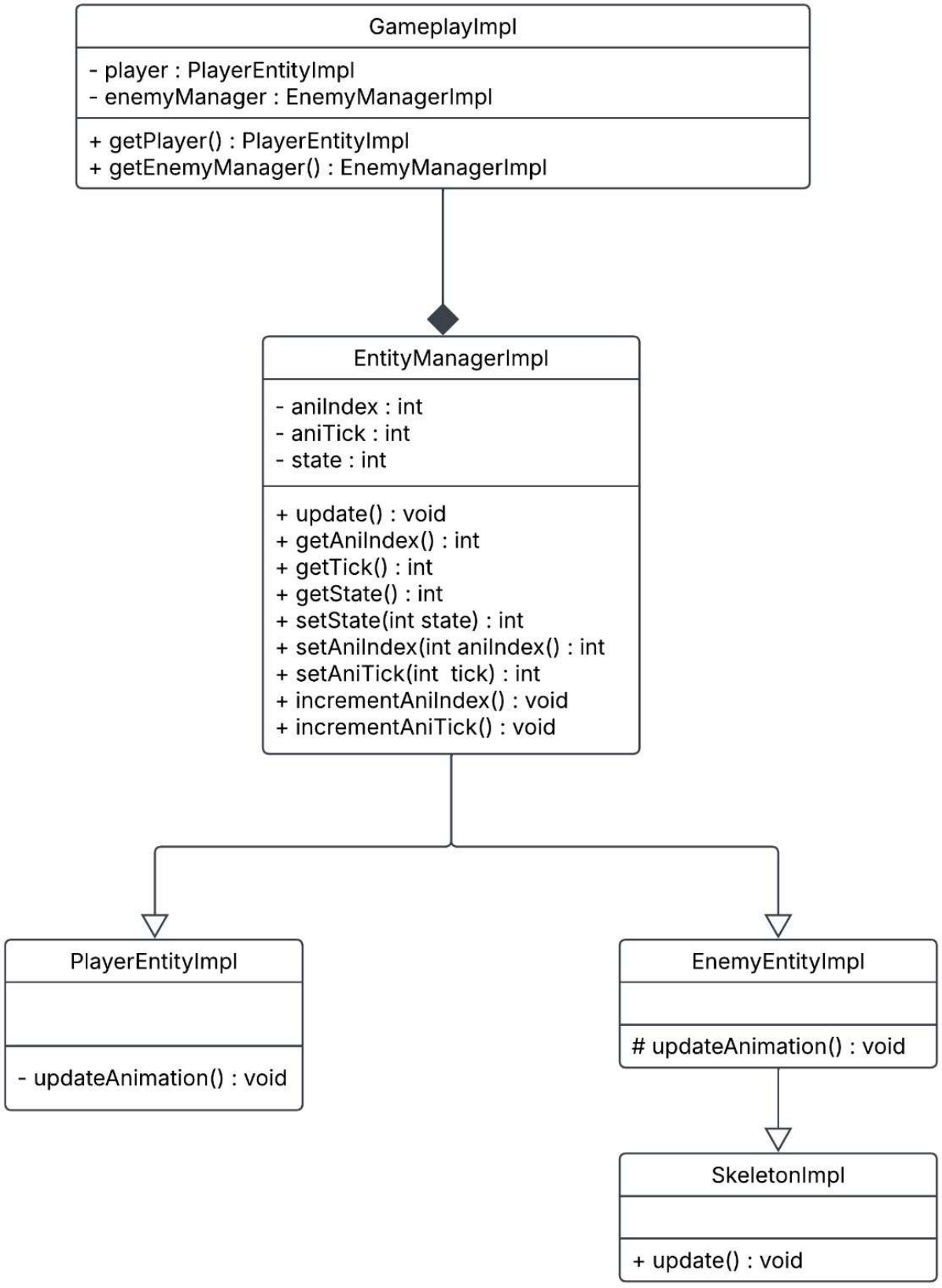

Figura 2.8: Rappresentazione UML delle animazioni del giocatore e dei nemici.

Problema: Le animazioni del giocatore e dei nemici sono composte da una serie di sprite differenti, ma simili per comportamento, che variano in base allo stato dell'entità. L'aggiornamento dell'animazione deve essere sincronizzato con il cambiamento dello stato dell'entità, garantendo una transizione fluida e coerente.

Soluzione: La classe EntityManagerImpl gestisce l'indice e i tick dell'animazione, applicabili sia alla classe PlayerEntity che alla classe EnemyEntity. L'aggiornamento dell'animazione, indipendentemente dal tipo di entità, viene effettuato nel metodo update. Per il giocatore, il numero di sprite è fisso, quindi l'aggiornamento dell'indice di animazione viene gestito in modo uniforme e fluido, con l'animazione aggiornata all'interno della stessa classe. Inizialmente, per i nemici, si era pensato di aggiornare l'animazione in base al tipo di nemico presente nel gioco. Tuttavia, poiché alla fine è stato deciso che l'unico nemico è lo scheletro, la classe SkeletonImpl eredita direttamente dalla classe EnemyEntity il metodo per l'aggiornamento dell'animazione.

2.2.3 Federico Di Franco

Sviluppo della fisica

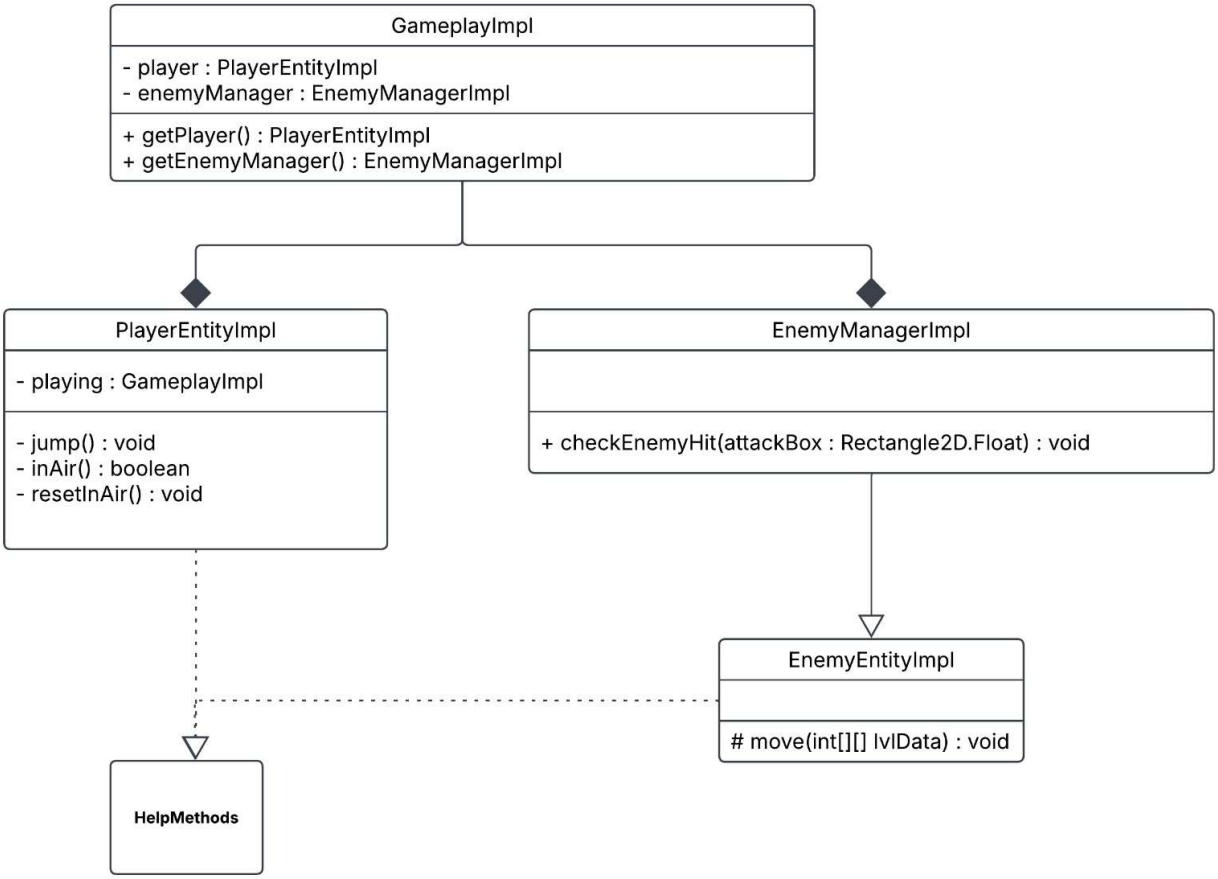

Figura 2.9: Rappresentazione UML della logica di movimento.

Problema: Gestire correttamente i movimenti delle varie entità.

Soluzione: La gestione del movimento per le entità del giocatore e dei nemici viene affidata a una classe di supporto, denominata "HelpMethods", che contiene metodi statici. Per garantire una maggiore visibilità e centralizzazione, è stato deciso di utilizzare un'unica classe per gestire sia i movimenti delle entità che le collisioni tra le entità stesse e con il livello.

Sviluppo delle collisioni

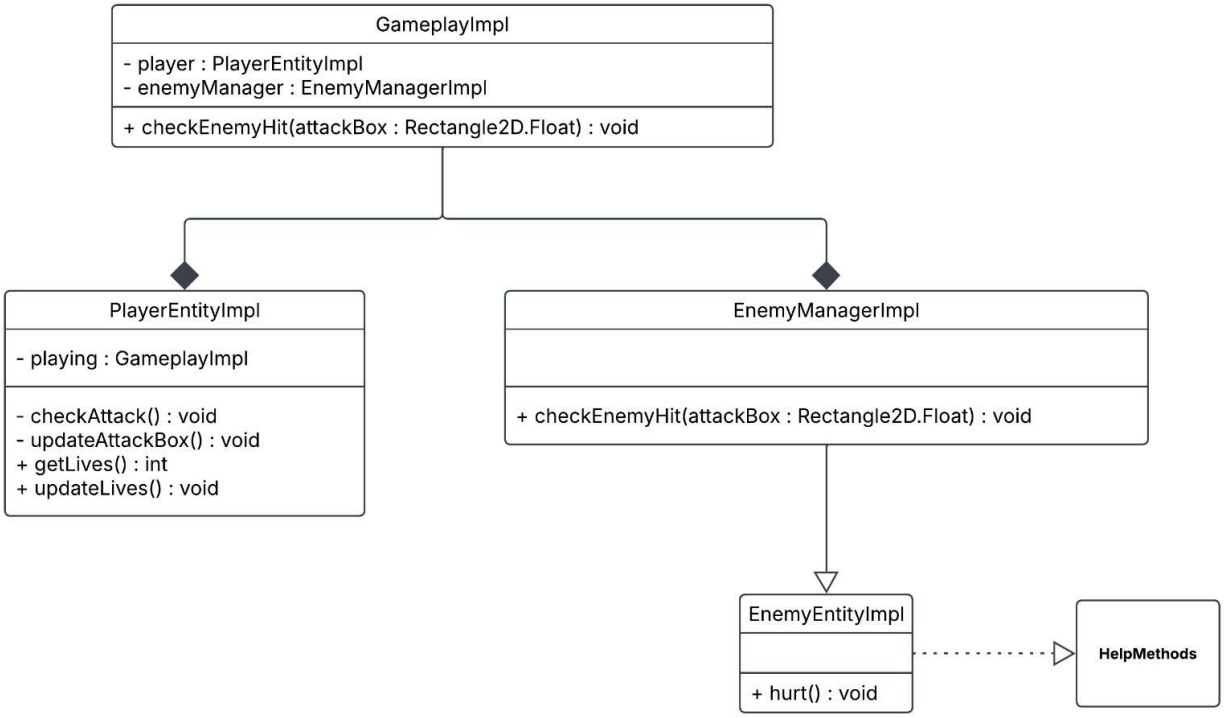

Figura 2.10: Rappresentazione UML delle collisioni avanzate tra il giocatore ed i nemici.

Problema: Gestire in modo accurato le interazioni tra l'entità del giocatore e i nemici, assicurando un aggiornamento corretto della logica di gioco e mantenendo la fluidità dell'esperienza di gioco.

Soluzione: In entrambe le classi delle entità è presente almeno un metodo per verificare le collisioni tra di esse. Nel file che gestisce il gameplay, viene infatti richiamato il metodo “checkEnemyHit”, già implementato nella classe dei nemici. Pertanto, esistono più controlli e invocazioni tra queste classi, che, come già sottolineato, fanno uso di una classe di supporto, denominata "HelpMethods".

2.2.4 Matteo Sacchetti

Gestione listener generici di input da tastiera e mouse

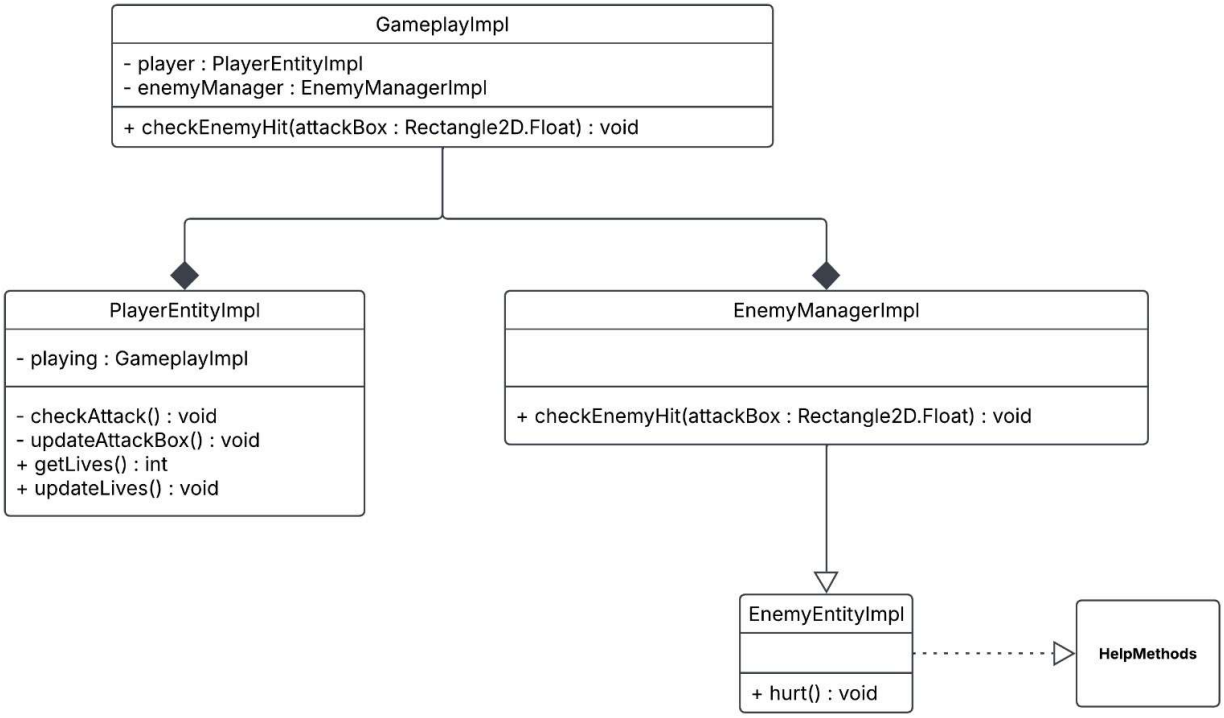

Figura 2.11: Rappresentazione UML del mouse e keyboard input listener.

Problema: Notificare le view quando avviene un mouse event o un key event.

Soluzione: Poiché ogni view può ricevere gli stessi eventi, abbiamo deciso di creare due listener generici: MouseInputs e KeyboardInputs. Questi vengono aggiunti alla classe ApplicationPanel e, quando ricevono un evento, lo inviano alla view corretta in base allo stato attuale del gioco. Durante il gioco, quando viene rilevata una pressione che corrisponde ad un movimento, l'evento viene notificato direttamente a GameplayImpl, che provvede a impostare il movimento appropriato.

Collisioni del giocatore con la mappa e gli oggetti

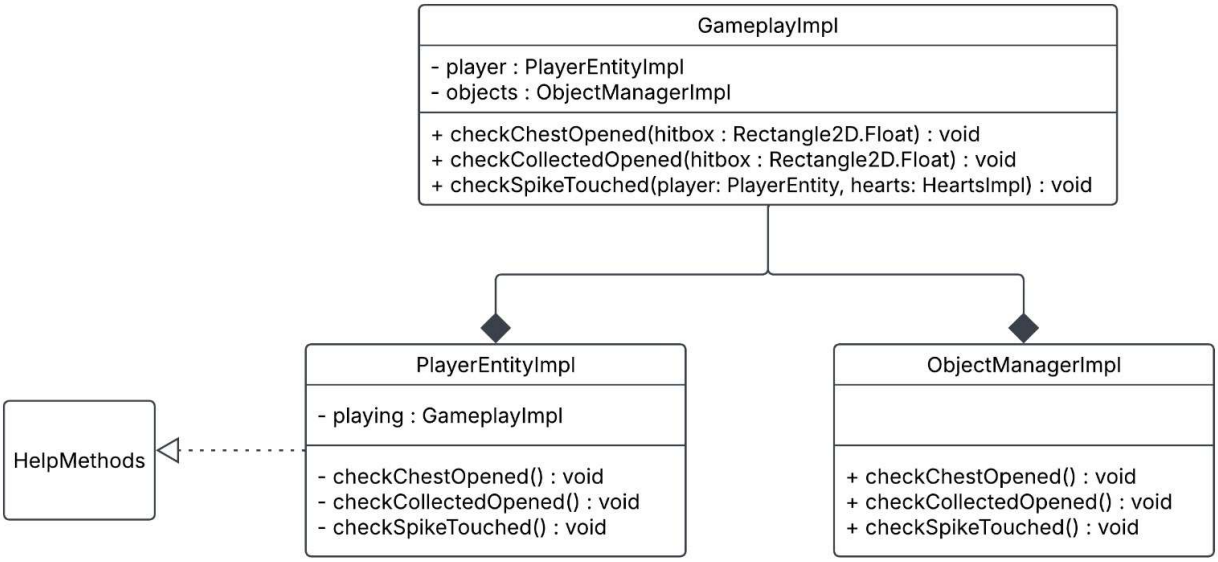

Figura 2.12: Rappresentazione UML dell'interazione con gli oggetti e il livello.

Problema: Gestire in modo accurato le interazioni tra l'entità del giocatore con la mappa e gli oggetti, assicurando un aggiornamento corretto della logica di gioco e mantenendo la fluidità dell'esperienza di gioco.

Soluzione: Nella classe del giocatore e in quella che gestisce gli oggetti, vengono richiamati gli stessi metodi presenti nel file che gestisce il gameplay. Pertanto, il funzionamento è simile a quello utilizzato per la gestione delle collisioni tra il giocatore e i nemici.

Capitolo 3

Sviluppo

3.1 Testing automatizzato

Per realizzare i seguenti test è stato utilizzato JUnit 5:

• CollisionTest, controlla il funzionamento delle collisioni entità-blocchi.

•GameplayTest, controlla la corretta inizializzazione del gameplay e le conseguenti situazioni di vittoria e sconfitta.

•GameStateTest, controlla la corretta visualizzazione delle view in base allo stato del gioco.

•InputsTest, controlla il corretto funzionamento degli input da tastiera.

•LevelAndImageTest, controlla che le immagini vengono caricate correttamente.

•SkeletonsTest, controlla il corretto funzionamento dei nemici.

Capitolo 4

Commenti finali

4.1 Autovalutazione e lavori futuri

Carmine Varipapa

All’inizio del progetto ammetto di aver avuto molte difficoltà nella crea
zione del gioco da zero. Man mano che il progetto progrediva, ho notato che la mia
comprensione del materiale e del processo di sviluppo migliorava gradualmente
permettendomi di trovare soluzioni sempre più efficaci. Ho trovato molto
stimolante e divertente la creazione di questo progetto, tanto che in futuro
non mi dispiacerebbe cimentarmi in altri. Ho apprezzato lavorare in gruppo
perché ciò spinge a pensare in modo più ampio e creativo e ad essere chiari
e flessibili. Penso che sia uno dei migliori progetti a cui abbia partecipa
to finora, anche se ovviamente ci sono parti da migliorare. In conclusione
penso questa sia stata un’ottima esperienza formativa sia tecnicamente come
programmatore che umanamente come persona.

Fabio Gruppioni

Il progetto è stato un’esperienza molto significativa sia dal punto di vista tecnico sia umano. Ho apprezzato il fatto che ci siamo sempre incontrati di persona per lavorare: questo ci ha permesso di conoscerci meglio e di creare una bellissima amicizia, oltre a un ottimo gruppo di lavoro. Mi sono concentrato soprattutto sulla realizzazione degli oggetti e dei nemici, aspetti che inizialmente mi spaventavano un po’, poiché mi sembravano molto complicati da realizzare, ma con l'aiuto e il supporto degli altri sono riuscito a completarli. È stato motivante vedere come il nostro lavoro prendeva forma grazie al contributo di tutti. Sono orgoglioso del nostro lavoro e, dopo tutto l'impegno e gli sforzi, posso dire di essere molto soddisfatto del risultato ottenuto, frutto di un vero lavoro di squadra.

Federico Di Franco

Sin dall'inizio di questo progetto, ho avuto la fortuna di lavorare con compagni competenti e molto disponibili con cui ho condiviso non solo il lavoro, ma anche momenti di confronto e crescita. Lavorare fianco a fianco, incontrandoci dal vivo per gestire ogni fase del progetto passo dopo passo, ha reso il processo molto più efficace e coinvolgente. Il confronto diretto ci ha permesso di affrontare insieme le difficoltà, chiarire dubbi e trovare soluzioni più rapidamente, rafforzando lo spirito di squadra. L'ambiente in cui abbiamo lavorato è stato estremamente stimolante e positivo: il supporto reciproco e la collaborazione ci hanno aiutato a superare le sfide tecniche, permettendoci di acquisire nuove competenze e affinare quelle già esistenti. Anche di fronte alle difficoltà, siamo riusciti a mantenere un buon equilibrio tra impegno e motivazione, portando a termine un progetto di cui siamo orgogliosi. Sono convinto che ogni progetto possa sempre essere migliorato, e il nostro non fa eccezione: ci sono ancora margini di crescita e tanto potenziale da sviluppare. Tuttavia, sono molto soddisfatto del risultato che abbiamo ottenuto e dell'esperienza che abbiamo vissuto insieme. Questa non è stata solo un'opportunità per mettere in pratica le mie competenze tecniche, ma anche un'occasione preziosa per crescere come individuo, lavorando in un gruppo straordinario.

Matteo Sacchetti

Questo progetto è stata un'esperienza estremamente arricchente, sia dal punto di vista tecnico che umano. Fin dall'inizio, abbiamo scelto di lavorare in gruppo incontrandoci di persona. Il confronto diretto ci ha permesso di discutere le idee in modo più immediato, chiarire eventuali dubbi e prendere decisioni in modo condiviso ed efficace. Come in ogni progetto, le prime fasi sono state caratterizzate da alcune difficoltà: comprendere a fondo i requisiti, definire una struttura solida e organizzare il lavoro in modo che fosse equilibrato per tutti. Tuttavia, grazie alla collaborazione e alla voglia di migliorarci, siamo riusciti a superare queste sfide, trovando sempre soluzioni. Guardando il risultato finale, sono molto soddisfatto del lavoro svolto. Ogni membro del gruppo ha dato un contributo significativo, e il progetto che abbiamo sviluppato riflette l'impegno, le competenze acquisite e la sinergia che siamo riusciti a costruire. È stata un'esperienza che ha rafforzato non

solo le nostre capacità di programmazione, ma anche la nostra attitudine al lavoro di squadra, elemento essenziale in qualsiasi contesto professionale. Spero vivamente in futuro di poter collaborare ancora con dei compagni così motivati ed affiatati.

Appendice A

Guida Utente

L’obiettivo del gioco è sconfiggere tutti i nemici presenti nel livello cercando di non prendere danno, evitando le trappole e raccogliendo tutti gli oggetti.

I comandi di gioco sono i seguenti:

- **A**: Movimento verso sinistra.
- **M**: Attacco.
- **D**: Movimento verso destra.
- **SPACE**: Salto.
- **P**: Pausa.

Per ottenere 3 stelle bisogna:

- Non prendere danno.
- Sconfiggere tutti i nemici.
- Prendere tutti gli oggetti prima di aver sconfitto l’ultimo nemico.

Bibliografia

1. L'immagine background del livello:

https://arludus.itch.io/2d-pixel-art-medieval-backgrounds-pack

2. Ispirazione:

https://www.kaaringaming.com/platformer-tutorial

3. Legenda:

https://egordorichev.itch.io/key-set?download

4. Menù e pannelli:

https://www.freepik.com/free-photos-vectors/game-menu

5. Cavaliere e scheletri:

https://craftpix.net/?s=skeleton+Pixel+Art+Sprite+Sheets

6. Chest:

https://itch.io/game-assets/tag-chest/tag-pixel-art

7. Musiche:

https://alkakrab.itch.io/free-medieval-soundtrack-no-copyright