Relazione Progetto OOP

Luca Grandi, Federico Guidazzi, Magistri Melissa, Tomidei Luca

June 2021

Contents

1 Analisi

L'applicazione, portata come progetto d'esame per il laboratorio di Programmazione ad oggetti, mira a reinterpretare un vecchio gioco da cabinato. L'intenzione è quella di riprodurre il cabinato "Space Invaders" in chiave più moderna, aggiungendo particolari selezionati da noi. Il videogioco prodotto nel giugno del '78 è stato uno dei giochi con più successo della sua generazione intorno al quale si creò un contesto senza precedenti: nacquero infatti anche tornei mondiali, oltre al merchandising e il record di videogioco più redditizio della storia.

1.1 Requisiti

Requisiti funzionali

Il gioco consiste nell'eliminare i nemici, gli alieni e i differenti boss per livello, nelle diverse fasi del gioco prima che raggiungano la parte inferiore dello schermo, evitando di essere colpiti più di tre volte dai loro proiettili.
Il suddetto videogioco sarà costituito da tre stage: ogni stage sarà composto di due fasi, nella prima l'utente dovrà fronteggiare una serie di nemici rappresentati da navicelle aliene che cercano di arrivare sulla Terra, l'obbiettivo sarà quello di eliminarle tutte per riuscire ad affrontare il boss nella fase due. Alla conclusione di uno stage, il successivo ripartirà dalla prima fase ma a difficoltà incrementata.
L'utente può effettuare movimenti orizzontali, durante i quali sarà possibile anche sparare verso i nemici. L'utente dispone di tre vite. I nemici si muovono autonomamente. Le navicelle degli alieni si muoveranno orizzontalmente e arrivati ai bordi laterali si abbasseranno di un'unità e invertiranno la loro traiettoria. Infine i movimenti dei boss saranno differenti per ogni stage.

Requisiti non funzionali

Il gioco dovrà gestire l'opzione della scelta di lingua, della scelta dei tasti di gioco ed essere efficiente nell'uso delle risorse.

1.2 Analisi e modello del dominio

Il dominio del modello applicativo si basa sulle relazioni tra le varie entità di gioco presenti all'interno dell'applicazione. Tali relazioni si sviluppano in base alle entità considerate; le entità principali, infatti, sono rappresentate da: player, enemy e bullet. L'entità "**Player**" rappresenta l'utente, il quale effettua i suoi movimenti ricevuti da tastiera esclusivamente sull'asse orizzontale. Inoltre, esso ha la possibilità, sempre tramite input da tastiera, di sparare grazie alla costruzione di un'altra entità fondamentale del gioco, il "**Proiettile**". L'entità "**Enemy**" effettua movimenti lungo la linea orizzontale della mappa fino a quando non ne tocca il bordo, tale evento li muove di un'unità verso il

basso per poi riprendere il movimento orizzontale in direzione opposta a quella di partenza.

L'entità "**Boss**" rappresenta il nemico finale di ogni livello: esso si suddivide in altre tre sotto-entità, i quali possono compiere movimenti randomici, rappresentati dallo spostamento dell'entità in punti casuali nella parte superiore della mappa, oppure movimenti orizzontali nella mappa di gioco, a seconda dell'entità esso può effettuate in combinazione entrambi i movimenti descritti.

L'entità "**Bullet**" è rappresentata da un'immagine rettangolare creata in gioco: in modo randomico dall'entità "Boss" seguendo sia l'asse verticale sia quello obliquo, entrambi con direzione verso il basso; tramite input da tastiera, dall'entità "**Player**", il quale segue esclusivamente l'asse verticale diretto verso l'alto ed infine sempre in modo randomico dall'entità "**Enemy**" sull'asse verticale rivolto verso il basso. Oltre a ciò, questa entità al momento della collisione con le altre entità in gioco è in grado di causare danni, facendone scendere la vita.

Le entità descritte precedentemente, rappresentate in figura 1, sono alla base della costruzione di gioco, il quale si compone di sei fasi; ogni del livello è composto da due fasi: nella prima viene creato un gruppo di alieni con i quali il player dovrà scontrarsi. Successivamente, nella seconda fase, alla sconfitta degli alieni, comparirà un boss con il quale battersi. Questo procedimento avverrà per un totale di tre livelli, nei quali avremo una difficoltà crescente.

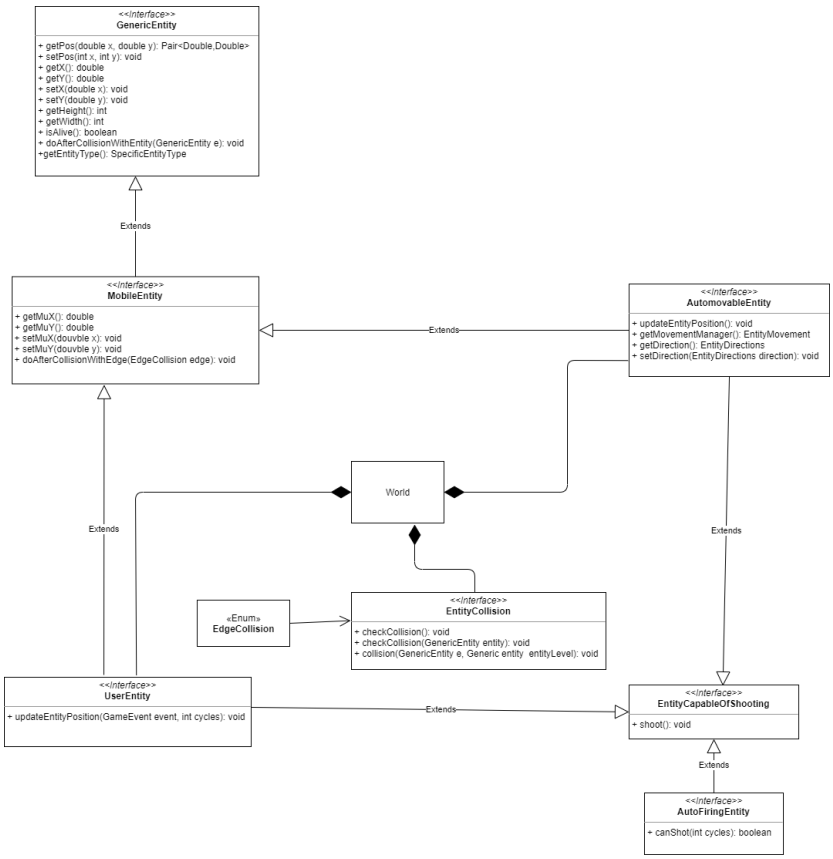

Figure 1: Rappresentazione delle entità di base

2 Design

2.1 Architettura

All'interno del progetto possiamo sottolineare l'uso del pattern MVC per l'implementazione del gioco: l'oggetto di tipo controller vieni generato all'interno della classe "Launcher", all'interno del suo costruttore vengono creati gli oggetti da cui si svilupperano il menù, quindi la View stessa, e il Model. Vedere figura 2 per inquadrare una struttara base della costruzione del patter MVC.

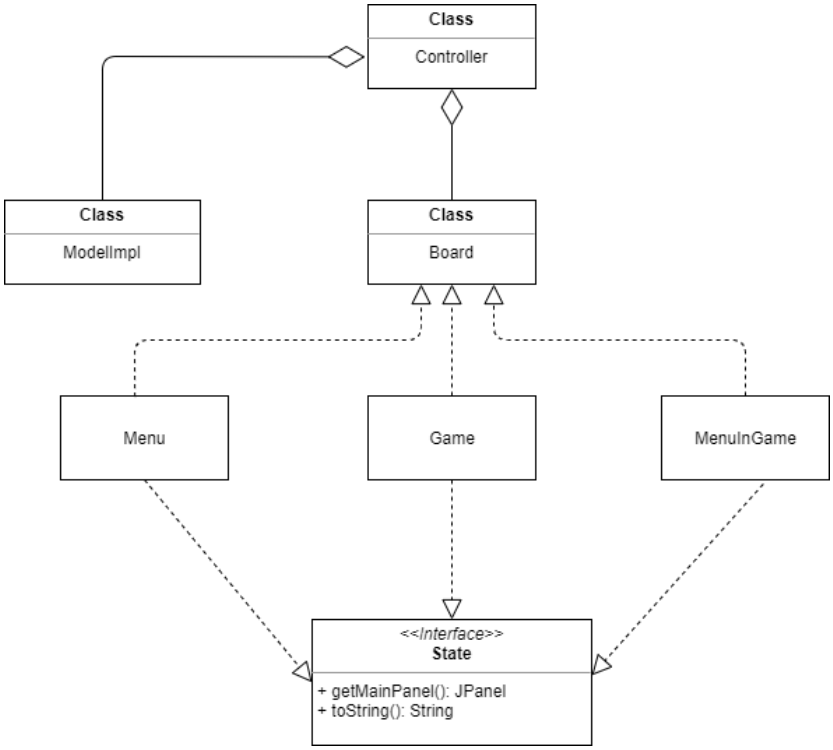

Figure 2: Rappresentazione della struttura di MVC.

In particolare il menu è formato da vari State diversi che vengono gestiti dal menuController; per la costruzione dei vari State sono state implementate diverse factory con lo scopo di sottostare alla regola "DRY", affinchè si possano evitare ripetizioni di parti di codice in classi diverse. Tutti gli stati del menu implementano l'interfaccia State, la quale si compone di un metodo, *getMainPanel*, che restituisce il panello utilizzato, vedi figura 3 per visualizzare l'implementazione del menù principale; mentre per il menù in gioco vedere l'implementazione alla figura 4. Inoltre è presente un controller che gestisce il cambiamento degli stati e dei suoni del menu.

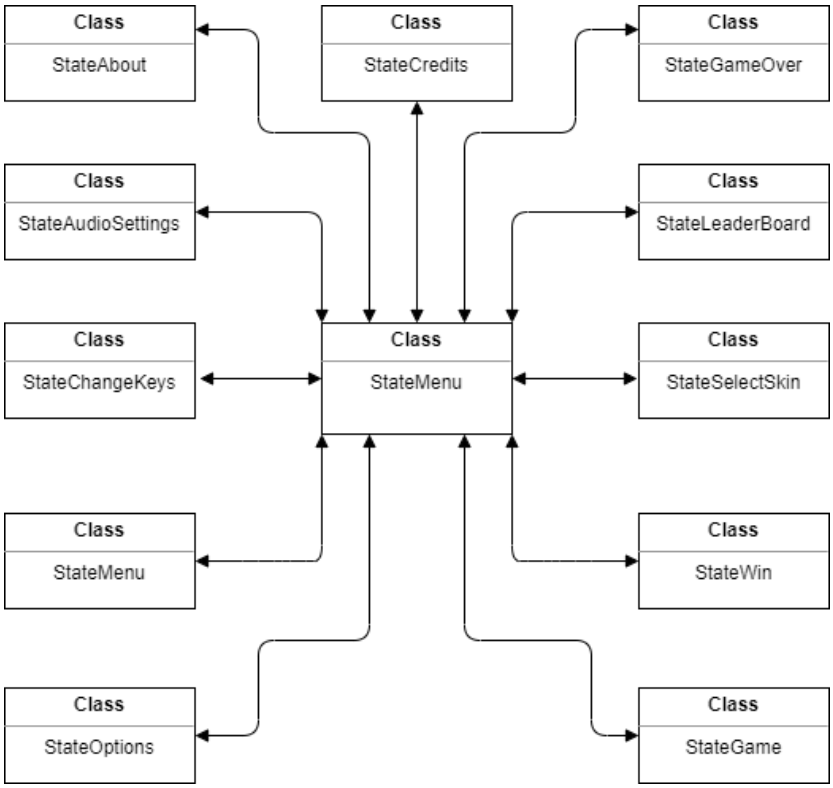

Figure 3: Rappresentazione della struttura del menù principale.

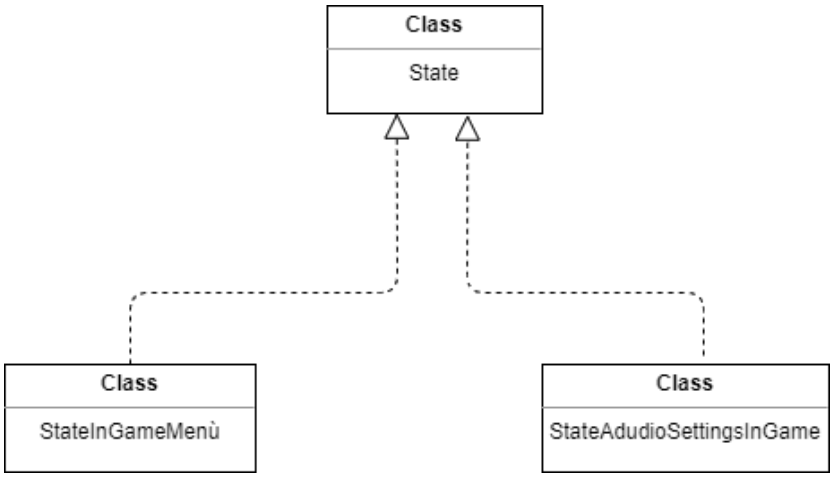

Figure 4: Rappresentazione della struttura del menù in gioco.

2.2 Design dettagliato

2.2.1 Luca Grandi

Interfacce ed Entità di base

L'implementazione delle entità presenti in gioco vengono fatte tramite l'ulitizzo di molteplici interfacce: vedere figura 5 per vedere come sono collegate tra di loro.

- **GenericEntiy**, rappresenta la base per la creazione delle entità di gioco. Infatti si compone di metodi che definiscono l'entità, tra cui: altezza, larghezza, posizione ed tipologia.
- **MobileEntity**, rappresenta tutte le entità che sono in grado di muoversi.
- **AutoMovableEntity**, rappresenta tutte le entità che possono muoversi in maniera automatica, cioè senza input da tastiera.
- **EnemyCapableOfShooting**, rappresenta tutte le entità che all'interno del gioco hanno la possibilità di sparare.
- **AutoFiringEntity**, rappresenta tutte le entità che sono in grado id sparare in maniera automatica.
- **UserEntity**, rappresenta l'entità che l'utente potrà usare in gioco, essa utilizza.
- **Enemy**, rappresenta l'insieme dei nemici che il giocatore dovrà sconfiggere; essa implementa le interafface dedicate al movimento ed allo sparo autonomo.
- **Bullet**, rappresneta l'entità proiettile di gioco, sia del giocatore sia quelli dei nemici; varia il modo in cui sono create e il modo in cui sono gestite le collisioni.

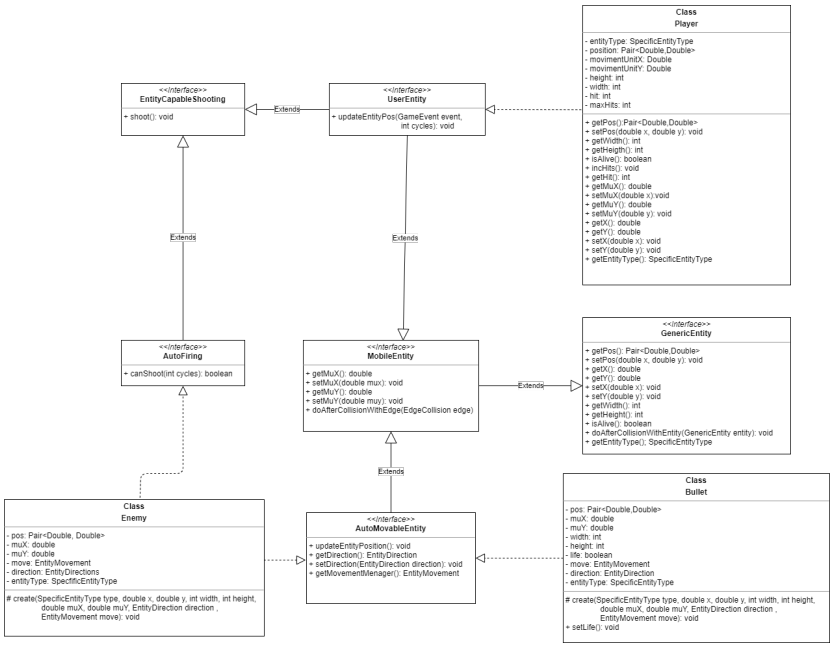

Figure 5: Rappresentazione delle interfacce delle Entità.

Boss

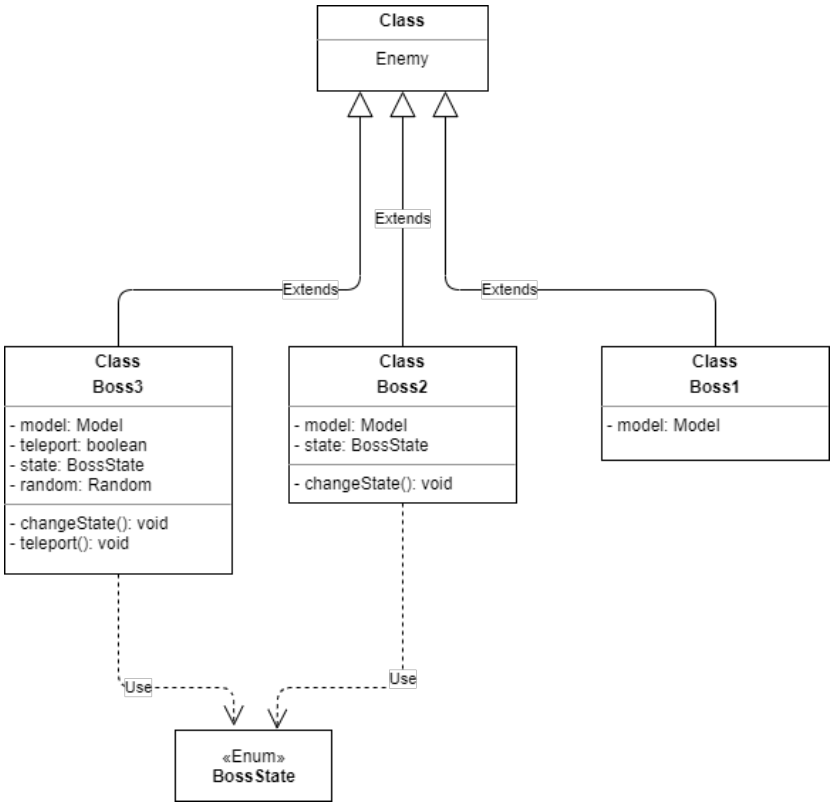

Figure 6: Rappresentazione delle entità Boss.

L'entità Boss si suddivide in tre tipologie di nemici di alto livello; vedere figura 6. Ciò che li contraddistingue è rappresentato dai loro movimenti, dal numero e tipologia del proiettile sparato.

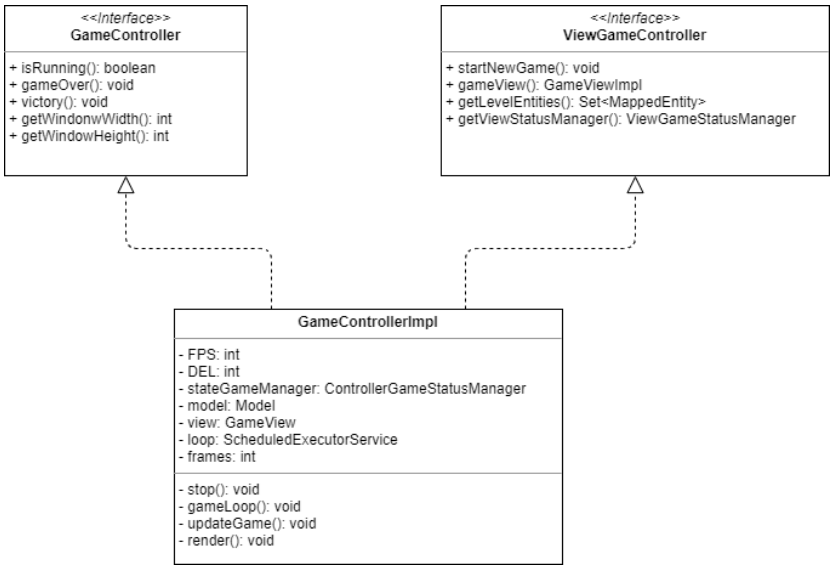

Figure 7: Rappresentazione del controller.

Controller

Il Controller rappresenta la classe che gestice il corretto funzionamento del gioco. Esso tramite il "gameloop" gestisce l'aggiornamento sia dei livelli sia della parte grafica dell'applicativo. Vedere la figura per una maggior chiarezza dei collegamenti all'interno del Controller.

StatusManager

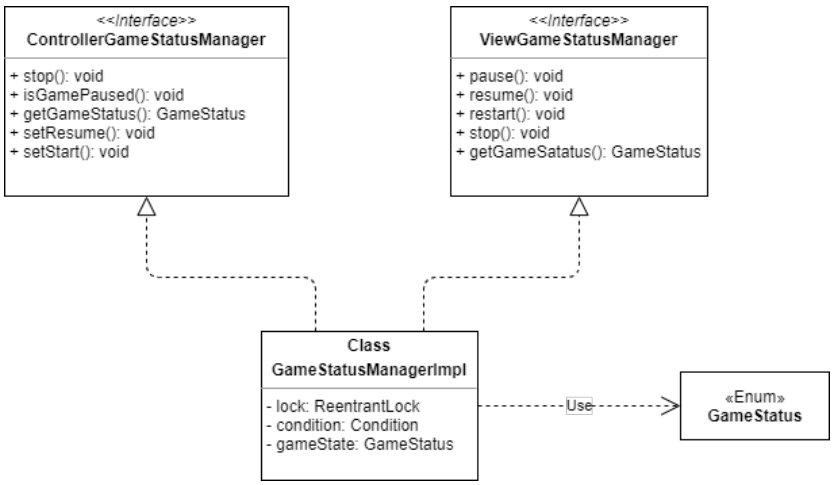

Figure 8: Rappresentazione del controller.

La classe che gestisce come il thread del Controller deve andare in pausa e come deve essere risvegliato. Si osservi la figura 8.

Fisica di gioco

La fisica di gioco comprende la gestione sia delle collisioni tra più entità sia quelle con i bordi del mondo di gioco ed il controllo dei movimenti automatici delle entità che lo permettono.

2.2.2 Federico Guidazzi

Menù di gioco

Il menù principale si compone di molteplici **State**:

- **StateAbout**, contiene delle informazioni riguardo il gioco originale, a cui ci siamo ispirati per la costruzione del nostro progetto. Inoltre ha anche una label della factory LabelFactory, che se clickata apre, tramite il browser predefinito del sistema, la pagina wikipedia di Space Invaders.
- **StateCredits**, contiene i nomi degli sviluppatori di questo progetto.
- **StateGameOver**, contiene una label creata attraverso la labelFactory, che riporta al menu principale, questo state infatti verrà visualizzato unicamente in caso di sconfitta dell'utente.

- **StateLeaderboard**, contiene la classifica del gioco, questo state viene chiamato su richiesta dell'utente dal menu principale, oppure dopo la vittoria dell'utente con uno score abbastanza alto per entrare nella classifica.
- **StateMenu**, è lo state del menu principale, ovvero lo State che verrà visualizzato non appena aperta l'applicazione, contiene varie label create attraverso la factory labelFactory, che se cliccate visualizzeranno gli state richiesti.
- **StateWin**, è lo state che viene visualizzato in caso di vittoria del player, e a seconda del punteggio ottenuto durante la partita dall'utente, cambierà. Infatti in caso di punteggio abbastanza alto da permettere al player di entrare in classifica verrà mostrata una label che se clickata riporterà allo stateLeaderboard, e una JTextField dove poter inserire il proprio nome. Mentre in caso contrario a quello appena descritto verrà visualizzata solamente una label che permette il ritorno al menu principale.
- **StateOptions**, è uno State di collegamento tra il menù principale e gli state AudioSettings, Leaderboard e ChangeKey; anch'esso utilizza le label create dalla factory per il suo scopo.

Board

E' la classe che gestisce il frame e alcuni metodi per il corretto funzionamento del gioco.

Menu Factories

Per rispettare la legge "DRY", ho anche creato varie factories, che adesso descriverò in breve:

- **LabelFactory**, grazie a questa factory si possono creare delle label clickabili, e che cambiano colore attraverso l'interazione con il cursore del mouse, inoltre esse gestiscono il cambiamento di stato del menu attraverso uno switch che opera in base al nome della label clickata.
- **LeaderboardFactory**, questa factory serve per la creazione della classifica in StateLeaderboard, essa legge da file i nomi e i punteggi e attraverso questi, una volta riordinati e riscritti all'interno del file da cui aveva letto, restiruisce una lista contenente la classifica ordinata e pronta ad essere visualizzata.
- **PanelBackgroundFactory**, attraverso questa factory è possibile avere un pannello che abbia come background un'immagine passata al costruttore della classe.

- **PanelFactory**, questa factory invoca PanelBackgroundFactory, cosi da avere un pannello con una certa immagine in background, inoltre aggiunge anche una stringa come titolo, attraverso la TitleFactory, e una label, creata attraverso a labelFactory, che riporta alla pagina principale del menu.
- **TitleFactory**, questa factory permette di creare una stringa "su misura", infatti chiamando il metodo createTitle a cui viene passata una stringa, un intero e un colore, esso ritorna una stringa con le dimensioni dell'intero passato, del colore passato e con il contenuto uguale a quello passato in input al metodo.

Nemici

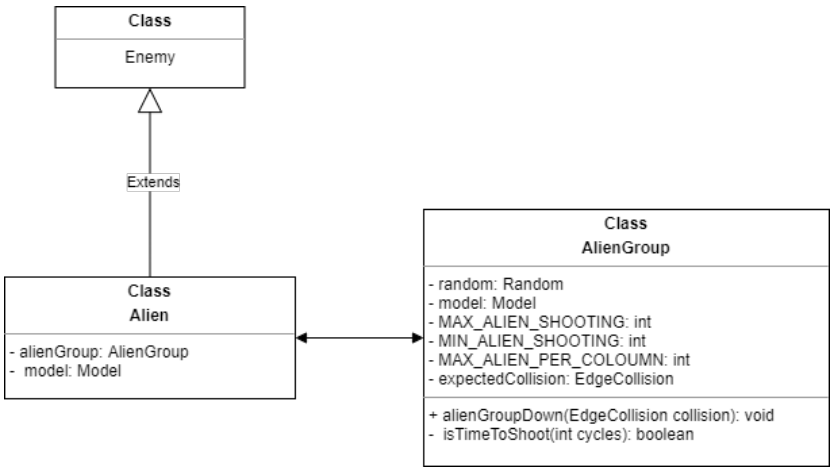

Figure 9: Rappresentazione delle entità Alien e AlienGroup.

Ho inoltre creato due entità di gioco, ovvero Alien e AlienGroup, la prima crea l'oggetto alieno e ne implementa i principali metodi di gestione dell'entità, come la gestione della collisione con i bordi dello schermo e lo sparo delle singole entita, mentre la seconda serve per la creazione dei singoli alieni e il loro posizionamento iniziale, inoltre ha anche altri due metodi che permettono il movimento di tutti gli alieni in caso di collisione con un bordo dello schermo, e la decisione di che alieni devono sparare contemporaneamente. Si puo trovare l'uml riguardante queste due entità nell'immagine 9.

2.2.3 Melissa Magistri

Menù di gioco

Per quanto riguarda lo sviluppo del menù principale ho implementato i seguenti State:

- **StateAudioSettings**, rappresenta la schermata nella quale è possibile modificare il volume della musica del menù tramite uno slider, cioè una barra numerata da 1 a 10.
- **StateChangeKeys**, rappresenta lo state in cui si dovrebbe avere la possibilità di cambiare i tasti di gioco., ma che al momento si trova non implementata.
- **StateGame**, rappresenta la schermata di gioco.
- **StateSelectSkin**, rappresenta la schermata in cui l'utente avrà la possibilità scegliere la propria skin tra le quattro disponibili oppure, attraverso il bottone "RandomSkin" si sceglierà in modo randomico la skin.

Menù in game

Il menù in gioco si attiva quando viene premuto da tastiera il tasto esc. Esso permette di effetturare delle azioni grazie a degli stati:

- **StateAudioSettingInGame**, la schermata in gioco che permette di modificare il volume della musica attraverso uno slider.
- **StateInGameMenu**, il panello che crea il menu in gioco.

Menu Factories

Per poter sviluppare gli state precedentemente elencati sono state implemenatate le seguenti factories:

- **SliderFactory**, grazie a questa factory viene creata uno slider: una barra numerata con la quale è possibile alzare e abbassare il volume della musica.
- **ButtonFactory**, questa factory crea i bottoni, tramite i quali sarà possibile scegliere la skin dell'utente e verrà collegata l'immagine scelta all'entità player.

Player e PlayerBullet

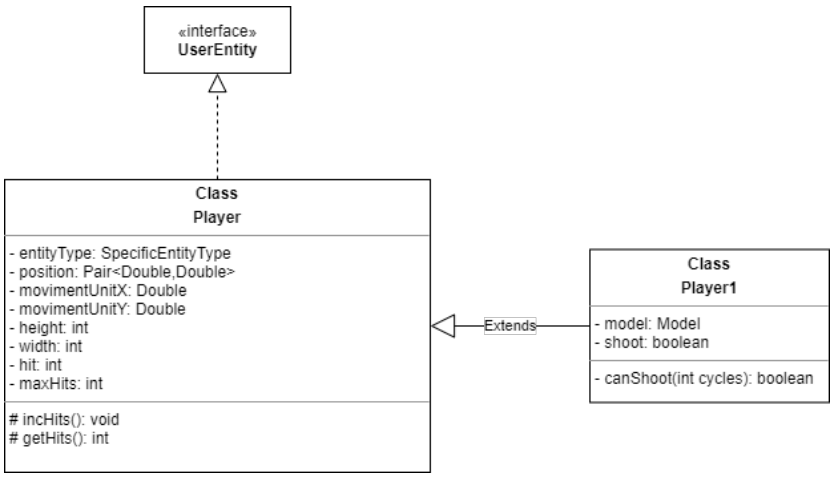

Figure 10: Rappresentazione della costruzione dell'entità Player.

Dalla sottoscritta è stato implementao tutto ciò che riguarda l'entità player, cioè il personaggio che verrà usato dall'utente. Il Player viene creato nella classe Player, la quale implementa l'interfaccia UserEntity. All'interno di questa classe viene gestita la posizione, attraverso le coordinate x e y; viene gestita la quantità di colpi che esso ha ricevuto tramite i metodi incHits, dove si incrementa il numero di colpi subiti, getHit, che ritorna la quantità di colpi al momento che sono stati ricevuti dal player e isAlive che controlla se i coplpi subiti siano maggiori di quelli consentiti. Tale classe viene poi estesa dalla classe del Player1, è possibile vedere questa estensione in figura 10 che permette la presenza di più giocatori, grazie al fatto che ciascuno di essi possa implementare in modo specifico la gestione delle collisioni sia con il bordo, sia con le altre entità. Inoltre, sempre all'interno della classe Player1, viene gestitta l'implementazione dello sparo e l'update della posizione dell'entità.

View Grafica

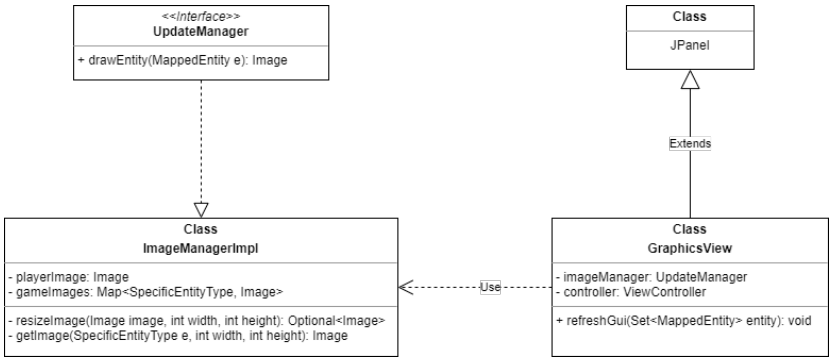

Figure 11: Rappresentazione della View.

La grafica di gioco viene gestita nelle classi visibili in figura UML. Esse gestistiscono l'aggionamneto delle immagini quando l'entità effettua dei cambiamenti di posizione. e' possibile vedere la gerarchia dell'implemetazione di queste strutture alla figura 11.

2.2.4 Luca Tomidei

Model

Il **Model** gestisce il mondo di gioco attraverso la classe "World" e svolge i seguenti comipiti:

- caricamento dei livelli(inizialmente trattati con file JSon ma successiamente modificati per problemi tecnici).
- posiziona l'entità.
- mappa le coordinate del mondo e delle entità.
- gestisce l'aggironamento delle entità e delle collisioni.
- coordina la creazione e la morte delle entità.
- gestisce lo score di gioco.

Di seguito è rappresentato lo schema uml del Model, alla figura 12, mentre alla figura **??** si può osservare l'implementazione del Mondo di gioco.

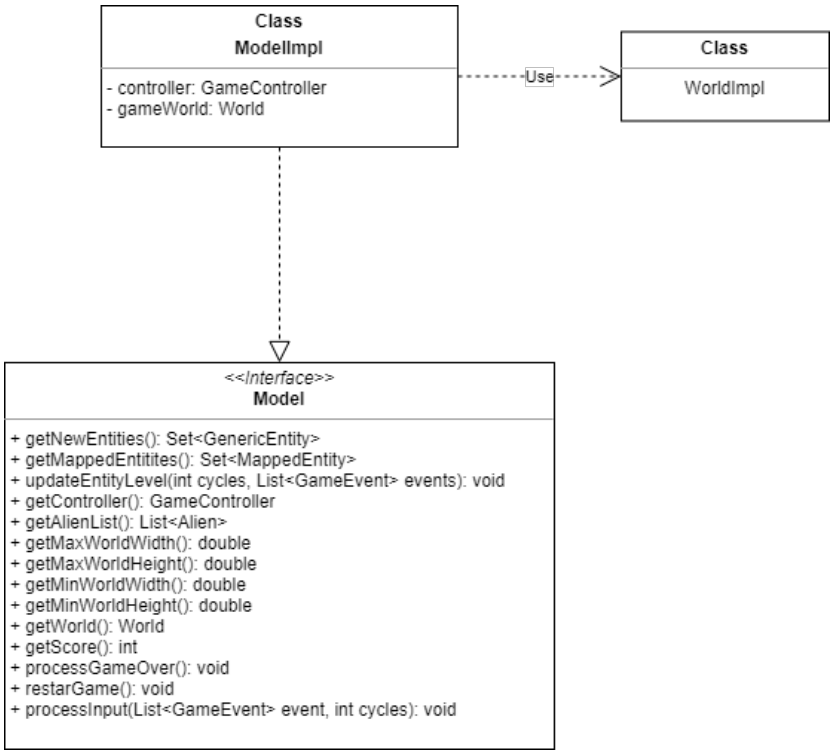

Figure 12: Rappresentazione del Mondo e del caricamento dei livelli.

View

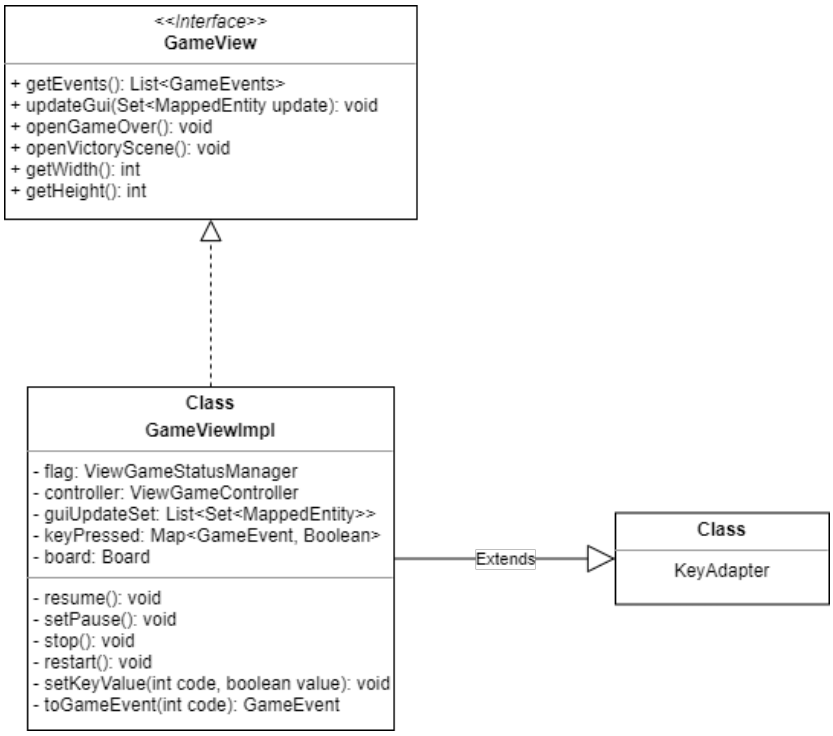

Figure 13: Rappresentazione della GameView.

La GameView si occupa della gestione dei GameEvent, cioè gli eventi eseguibili successivamente all'input da tastiera, ad esempio i movimenti e lo sparo del player o opzionalità di metter il gioco in pausa. Nella figura 13 è possbile osservare lo schema uml della parte di GameView.

View

All' interno di questo package viene gestito l'aggiornamento di StateGame e gli input che vengono da tastiera: **esc** per aprire ed uscire dal meù in gioco, **R** per far ricominciar eil gioco dal primo livello, **S** per uscire dal gioco e aprire il menu iniziale, **barra spaziatrice** per permettere all'utente di sparare e le **frecce** per effettuare i movimenti orizzoantali del player.

3 Sviluppo

3.1 Testing automatizzato

All'interno del progetto sono stati implementati diversi test per verificare il corretto funzionamento delle entità.

I test sono stati raggruppati all'interno del package spaceInvadersTest e sono eseguibili tramite la libreria JUnit premendo tasto destro sul progetto alla voce "Run as" -¿ "JUnit Test".

3.2 Testing manuale

Inoltre sono stati effettuati test manuali per verificare funzionalità e corretta eseguibilità della applicazione sia su sistemi Windows sia su sistemi Linux.

3.3 Metodologia di lavoro

Lo sviluppo del progetto si è svolto in modalità cooperativa, comunicando attraverso la piattaforma GitHub per spiegare dettagliatamente i vari cambiamenti fatti nello sviluppo delle proprie classi, la piattaforma Discord per confronti diretti tramite una chat vocale e altre applicazioni di messagistica per comunicazioni veloci e periodiche sul lavoro svolto, affinchè potessimo mantenere costanza. La suddivisione del lavoro, definita successivamente alla progettazione dei requisiti dell'applicazione, si svolge come segue:

3.3.1 Luca Grandi

Durante lo sviluppo del progetto ho gestito in maniera autonoma:

- L'implementazione del Controller.
- L'implementazione dei Boss.
- L'implementazione delle interfacce alla base delle entità.
- La fisica di gioco.

3.3.2 Federico Guidazzi

Durante lo sviluppo del progetto ho gestito in maniera autonoma:

- L'implementazione del Menù principale.
- L'implementazione dei Nemici (Alien e AlienGroup).
- Gestione e scelte della musica.

3.3.3 Melissa Magistri

Durante lo sviluppo del progetto ho gestito in maniera autonoma:

- L'implementazione del Menù in gioco.
- L'implementazione dell'entità Player.
- Gestione della Grafica di gioco.

3.3.4 Luca Tomidei

Durante lo sviluppo del progetto ho gestito in maniera autonoma:

- L'implementazione del Model.
- L'implementazione della parte di Grafica relativa agli aggiornamenti.
- Caricamento dei livelli e gestione del mondo di gioco.

3.4 Note di sviluppo

Nella seguente sezione verrano approfondite le scelte, di ciascuno dei componenti, per l'utilizzo di alucne feature avanzate.

3.4.1 Luca Grandi

Ho utilizato le seguenti feature:

- *ScheduledExecutorService*, nella classe GameControllerImpl per la gestione del gameLoop.
- *Stream*, utilizzaati con l'obbiettivo di selezionare delle entità specifiche per poter gestire le collesioni di esse con altre entità specifiche.

3.4.2 Federico Guidazzi

Nello sviluppo delle mie classi ho utlizzato le seguenti feature:

- *Stream e Lambda Expression*, nella classe AlienGroup.
- *Optional*, nella classe LeaderboardFactory per la lettura da file delle stringe relative alla classifica.

3.4.3 Melissa Magistri

Nell'implemenzaione delle classi da me sviluppate ho potuto utilizzare delle feature, tra cui:

- *Lambda Expression*, all'interno degli aciton listener per l'azione che deve fare il pulsante al momento della pressione in: ButtonFactory per selezionare la skin del player, in StateSelectSkin per sclegliere randomicamente la skin del player, in GraphicsView per poter scorrere il set delle entità e associarcìvi la propria immagine.
- *Optional*, All'interno della classe ImageManagreImpl con lo scopo di ritornare un optionale vuoto nel caso in cui il metodo resizeImage non riesca a trovare l'immagine a cui cambiare le dimensioni.
- *Collection*, in particolare l'uso di un set sincronizzato per evitare accessi simultanei allo stesso thread.

3.4.4 Luca Tomidei

Nello sviluppo delle classi ho utilizzato le seguenti feature:

- *JSon*, per l'inizzializzazione dei livelli.
- *Stream*, nella classe WorldImpl per filtrare determinate entità.
- *Set e Mappe sincronizzate*, per la gestione dei tasti e degli eventi di gioco.

4 Commenti finali

4.1 Autovalutazione e lavori futuri

4.1.1 Luca Grandi

Dopo aver riscontrato numerose difficoltà nella fase di progettazione mi ritengo soddisfatto della produzione da noi sviluppata. L'aspetto positivo del progetto consiste nel aver avuto la possibilità di approfondire gli argoemnti visti a lezioni e poterli applicare all'interno del progetto. Mentre quelli negativi, a casua di mancanza di esperienza, sono state rilevate nella fase di progettazione. Sicuramente in futuro questa esperienza mi tornerà utile se dovessi trovarmi in una situazione simile.

4.1.2 Federico Guidazzi

Sono abbastanza felice del risultato ottenuto nel progetto. Grazie ad esso infatti ho potuto aumentare le mie conoscenza nella programmazione con Java.

Inoltre grazie a questo progetto sono riuscito anche a rendermi conto della difficoltà della creazione di un gioco e dell'importanza dell'organizzazione del progetto prima di iniziare a metterci effettivamente "le mani sopra". Sono certo che questi problemi saranno sicuramente minori in un prossimo progetto, poiche probabilemente presenti dettati dell'inesperienza.

4.1.3 Melissa Magistri

Ritengo di essere soddisfata dello sviluppo di questo progetto, nonostante le difficoltà riscrontrate durante la progettazione. In particolare ho riscontrato difficolta nelle parti non approfondite a lezione ed è stato abbastanza complicato effettuare le scelte migliore per il progetto. Gli aspetti positivi del progetto sono sicuramente aver avuto la possibilità di confrontarsi con la prima esperienza di gruppo e, quindi, il dovre svolgere un "lavoro" in gruppo, effetturando delle scelte non solo utili ai propri fini ma anche a quelli degli altri. Infine ritengo che questa esperienz adi prottegazione di un videogioco possa ampliarmi la possibilità di scelta di lavoro futuro e possa preparmi ad esperienza sia di studio che lavorative con le quali potrei confrontarmi nel corso del mio percorso.

4.1.4 Luca Tomidei

Le consclusioni che ritengo utili comunicare penso possano essere le seguenti: ritengo che l'importanza del lavoro di gruppo sia fondamentale in questo ambito lavorativo e in particolare sia fondamentale per lo sviluppo di un progetto entro i termini dati. Nel complesso ammetto che la progettazione di un videogioco sia estremamente complessa, soprattutto il concetto di approfondire in maniera autonoma gli argomenti utili allo sviluppo dell'applicativo. In conclusione ritengo che questa esperienza sia utile al miglioramento delle mie conoscenze sia tecniche che mentali, verso il mondo del lavoro.

4.2 Difficoltà incontrate e commenti per i docenti

A Guida utente

A.1 Avvio e Menù

All'avvio dell'applicazione all'utente verrà mostraro il menù principale nel quale potrà spostarsi con il cursore. Cliccando sui pulsanti verranno aperte le schermate relative.

Figure 14: Immagine del Menù principale.

A.2 Gameplay

Prima dell'effettivo inizio del gioco si aprirà una finestra, vedere figura 15 con l'elenco dei comandi utili per il GamePlay dell'applicazione.

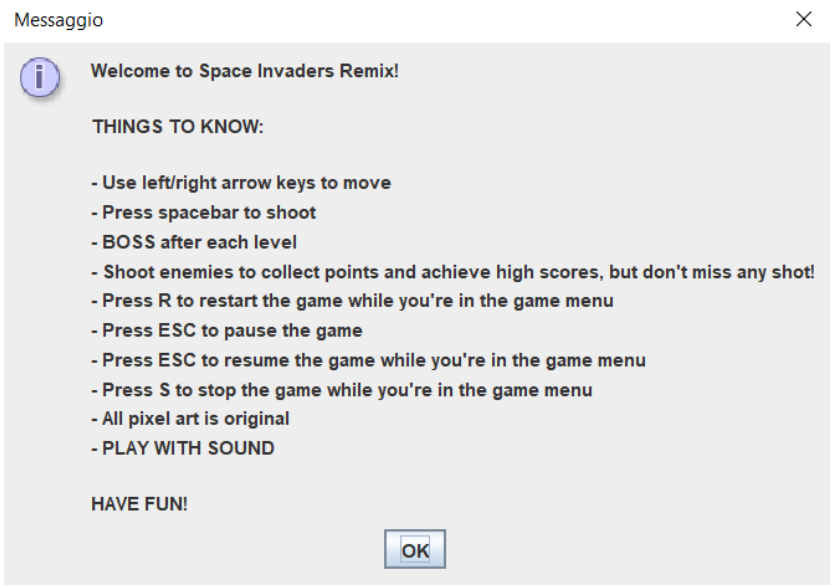

Figure 15: Immagine della finestra di spiegazione dei comandi di gioco.

Avviato il gioco l'utente si troverà di fronte una serie di nemici a cui potrà sparare e, allo stesso tempo, dovrà evitare i loro colpi per non essere sconfitto.

All'uccisione di tutti gli alieni, il giocatore dovrà scontrarsi con il boss. Questo schema avverrà per tre livelli ma con difficiltà diversa.