Space Invaders

Riccardo Azzali

Michele Pasi

Enrico Baroni

Alberto Rossi

DATA DI CONSEGNA

**Sommario**

Questo documento `e una relazione del progetto di OOP (Object-Oriented Programming) dell’anno accademico 2020-21.

Lo scopo di questo documento è quello di descrivere i punti principali della progettazione e del funzionamento dell’applicazione. Tale descrizione sarà suddivisa in diverse sezioni e sottosezioni in modo da partizionare le varie fasi di sviluppo del sistema applicativo. Per ciascuna delle sezioni del documento sarà fornita una descrizione generale o particolare della corrispettiva fase di analisi, progettazione o implementazione.

Il modello della relazione segue il processo tradizionale di ingegneria del software fase per fase (in maniera ovviamente semplificata).

**Indice**

[**1 Analisi**](about:blank) **2**

[1.1 Requisiti](about:blank) 2

[1.2 Analisi e modello del dominio](about:blank) 3

[**2 Design**](about:blank) **6**

[2.1 Architettura](about:blank) 6

[2.2 Design dettagliato](about:blank) 7

[**3 Sviluppo**](about:blank) **17**

[3.1 Testing automatizzato](about:blank) 17

[3.2 Metodologia di lavoro](about:blank) 17

[3.3 Note di sviluppo](about:blank) 22

[**4 Commenti finali**](about:blank) **24**

[4.1 Autovalutazione e lavori futuri](about:blank) 24

[4.2 Difficoltà incontrate](about:blank) 25

[**A Guida utente**](about:blank) **26**

**Capitolo 1**

# Analisi

## 1.1 Requisiti

Questo software tenta di replicare il famosissimo videogame arcade anni ’70 **“Space Invaders”**. Un videogioco arcade è un videogioco che si gioca in una postazione pubblica apposita a gettoni o a monete, costituita fisicamente da una macchina posta all'interno di un cabinato. Gli arcade rappresentarono la prima generazione di videogiochi di largo consumo. In Space Invaders, l’utente controlla una navicella aerospaziale la quale ha lo scopo di sopravvivere ai vari nemici che si presentano in differenti livelli.

**Requisiti funzionali**

* L’applicazione garantisce il movimento della navicella, attraverso le frecce direzionali, all’utente nella finestra di gioco.
* Sono presenti differenti tipologie di nemici, che cercano di sconfiggere la navicella del giocatore colpendola con colpi di arma o tentando di scontrarsi con essa.
* La partita termina quando le navicelle nemiche raggiungono il fondo dello schermo decretando l'avvenuta invasione o quando si esaurisce la vita disponibile.
* Durante il corso del gioco è possibile sbloccare dei power-up aggiuntivi al giocatore.
* E’ presente la musica durante il gioco.
* E’ presente un menu grafico composito di vari sottomenu i quali guidano l’utente nell’esplorazione delle funzionalità del gioco.
* A partita terminata verrà memorizzato in una classifica il punteggio totalizzato dall’utente in base alle azioni compiute in gioco. I punteggi sono visualizzabili graficamente in una tabella dei migliori giocatori.

**Requisiti non funzionali**

* Il gioco potrebbe supportare varie lingue.
* Potrebbe essere aggiunta una modalità di gioco *Sopravvivenza*, ove l’utente gestisce la navicella in modo macchinoso differentemente dalla modalità di gioco principale.

## 1.2 Analisi e modello del dominio

Il sistema applicativo avrà varie tipologie di entità, che dovranno rispettare i parametri definiti nell’interfaccia Entity che rappresenta il fulcro base di tutte loro:

* il personaggio principale (player);
* i proiettili del personaggio principale
* le navicelle nemiche;
* i proiettili delle navicelle nemiche;
* i power-up;
* le meteore.

L’interfaccia Game è la base logica del gioco e mantiene al suo interno tutti i riferimenti alle entità prima citate.Questa interfaccia, inoltre si occuperà di controllare le collisioni, in particolare tra il player e tutte le altre entità.

Durante lo svolgimento del gioco il personaggio può utilizzare determinati power-up, per esempio: scudo protettivo, vita bonus e il freeze delle navicelle nemiche.

Il sistema prevede lo scontro tra le diverse entità:

* i proiettili del giocatore si possono scontrare con le navicelle nemiche e viceversa;
* le navicelle nemiche possono collidere contro il giocatore;
* il giocatore può scontrarsi contro le meteore;
* il giocatore può ottenere power-up scontrandosi con loro.

All’interno di Game ci sarà un riferimento a Level per potrerci permettere di sapere a che livello il giocatore è arrivato.

Le difficoltà che potrebbero presentarsi nello sviluppo del software consistono nell’ identificazione delle collisioni fra entità, ottimizzazione nel caso di presenza di un numero elevato di entità, raccolta di power-up, creazione casuale delle meteore.

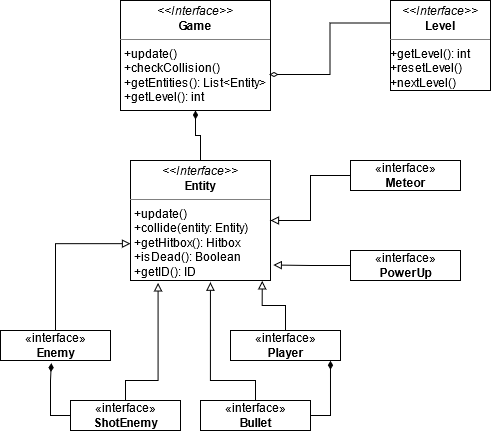


Figura 1: Schema UML con le principali entità.

**Capitolo 2**

# Design

## 2.1 Architettura

L'architettura della nostra applicazione segue il pattern MVC (Model-View-Controller).

Il main del gioco crea la view e il controller.

Il controller gestisce l’applicazione, comunica sia con la view, a cui invia periodicamente l'ordine di aggiornarsi e ridisegnare le entità, sia con il game da cui ottiene informazioni sullo stato del gioco. In particolare è vitale che il controller possa ottenere informazioni riguardo alle entità presenti in quel momento, al punteggio e al livello, per poterle passare alla view e quindi aggiornarla.

Abbiamo utilizzato il controller come observer, la view infatti non può comunicare direttamente con il controller, quello che succede è che la view notifica al controller l’input ricevuto ed è poi lui a decidere cosa fare.