**Realizzazione di un Clone di Space Invaders in Java: Architettura, Implementazione ed Estensioni**

––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

**Abstract**

Questo progetto ha l’obiettivo di realizzare un clone del classico videogioco Space Invaders utilizzando il linguaggio Java e la libreria Swing. Il sistema, denominato **JSpaceInvaders**, riprende le dinamiche fondamentali del titolo originale e le arricchisce con estensioni che ne ampliano la giocabilità. Tra le innovazioni implementate si annoverano:

• Una struttura modulare organizzata in più package (main, model, controller, view, utility) per separare la logica di gioco, il rendering e la gestione degli input;

• Un sistema di asset management per immagini (sprite per il giocatore, nemici, proiettili, esplosioni e background), suoni (effetti e musica di sottofondo) e font personalizzati;

• Meccaniche di gioco che includono il movimento fluido della navicella, lo sparo e il rilevamento delle collisioni tramite hitbox personalizzate;

• Un sistema di vite che consente al giocatore di iniziare con **3 vite**, visualizzate come icone nella parte superiore centrale. Quando la navicella viene colpita e se sono presenti vite residue, viene attivato un breve periodo di invulnerabilità accompagnato da un suono di avviso, mentre la perdita dell’ultima vita attiva l’animazione dell’esplosione e il game over;

• Generazione di ondate di nemici suddivise in tre tipologie (Green, Blue e Yellow) con punteggi differenziati (10, 20 e 40 punti rispettivamente) e un incremento progressivo della difficoltà ad ogni livello;

• Uno sfondo dinamico che scorre orizzontalmente da destra verso sinistra per creare un’atmosfera arcade;

• Gestione degli effetti sonori, inclusa la possibilità di attivare/disattivare l’audio tramite un comando dedicato.

––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

**1. Descrizione del Problema e Scopo del Sistema**

**1.1 Il Videogioco Space Invaders**

Space Invaders è uno dei videogiochi arcade più iconici, in cui il giocatore controlla una navicella posizionata nella parte inferiore dello schermo per difendersi dall’invasione di nemici alieni che avanzano dall’alto. JSpaceInvaders si propone di replicare questa esperienza, offrendo un gameplay reattivo e dinamico che combini le meccaniche classiche – come il movimento della navicella, lo sparo e il rilevamento delle collisioni – con alcune estensioni moderne. L’obiettivo del progetto è quello di realizzare un sistema in cui il giocatore possa:

• Muovere la navicella in maniera fluida (principalmente orizzontalmente, con limiti di movimento verticale);

• Sparare proiettili per abbattere ondate di nemici;

• Gestire un numero prestabilito di vite (3 iniziali), con un breve periodo di invulnerabilità dopo ogni impatto per evitare danni consecutivi;

• Affrontare ondate di nemici suddivise in tre tipologie, ognuna con un valore in punti differente e caratteristiche specifiche, che incrementano progressivamente la difficoltà man mano che il livello aumenta.

**1.2 Aspetti Fondamentali del Gioco**

I punti chiave che JSpaceInvaders deve riprodurre sono:

• **Controllo dell’Astronave:**

 – Il giocatore controlla la navicella, che si muove in modo fluido e reattivo lungo la parte inferiore dello schermo.

 – La navicella spara proiettili tramite un tasto dedicato.

 – Il sistema prevede 3 vite, rappresentate graficamente nella parte superiore centrale. In caso di impatto, se il giocatore ha vite residue, viene riprodotto un suono di avviso e il giocatore entra in un breve stato di invulnerabilità; altrimenti, l’animazione dell’esplosione viene attivata e la partita termina.

• **Generazione delle Ondate di Nemici:**

 – Il gioco genera ondate di nemici in tre tipologie:

  • **Green:** Nemici di base, 5 al primo livello, che si muovono in linea retta da destra verso sinistra.

  • **Blue:** Nemici intermedi, 3 al primo livello, caratterizzati da una velocità leggermente superiore.

  • **Yellow:** Nemici di difficoltà elevata, 1 al primo livello, che seguono traiettorie rettangolari complesse.

 – Al termine di un ciclo di tre ondate, il livello aumenta. Con l’incremento del livello, il numero dei nemici e la loro velocità aumentano, così come la frequenza di sparo, aumentando la difficoltà complessiva.

• **Rilevamento delle Collisioni:**

 – Il sistema rileva le collisioni tra i proiettili del giocatore e i nemici, e tra i proiettili dei nemici e la navicella.

 – Le collisioni provocano l’attivazione di un’animazione di esplosione centrata sull’oggetto colpito e l’assegnazione del punteggio corrispondente (10 punti per i Green, 20 per i Blue, 40 per i Yellow).

• **Sistema di Punteggio e Classifica:**

 – Il punteggio viene aggiornato in tempo reale in base ai nemici abbattuti e, al termine della partita, viene salvato per la visualizzazione di una classifica degli HI-SCORE.

• **Gestione del Game Over e delle Pause:**

 – La partita termina quando il giocatore esaurisce tutte le vite. In tal caso, viene mostrata una schermata di Game Over con il punteggio finale e le opzioni per riprovare o tornare al menu.

 – Il gioco permette la pausa e la ripresa tramite un comando da tastiera.

• **Audio e Grafica:**

 – Il sistema integra effetti sonori per spari, esplosioni e perdita di vita, oltre a una colonna sonora.

 – È possibile attivare/disattivare l’audio tramite un comando.

 – Lo sfondo è dinamico e scorre orizzontalmente da destra verso sinistra per creare un ambiente arcade immersivo.

––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

**2. Specifica dei Requisiti**

**2.1 Requisiti Funzionali**

1. **Controllo dell’Astronave:**

 – Il giocatore deve poter muovere la navicella orizzontalmente (con possibilità di movimento verticale entro limiti definiti).

 – La navicella spara proiettili tramite un tasto dedicato (ad es. la barra spaziatrice).

 – Il giocatore inizia con **3 vite**, visualizzate nella parte superiore centrale. In caso di impatto, se sono rimaste vite, viene riprodotto un suono di perdita vita e il giocatore entra in un breve stato di invulnerabilità; se si perde l’ultima vita, viene attivata l’animazione dell’esplosione e il gioco termina.

2. **Generazione delle Ondate di Nemici:**

 – Il gioco genera ondate di nemici in tre tipologie:

  • **Green:** 5 nemici di base al primo livello, che si muovono in linea retta da destra verso sinistra.

  • **Blue:** 3 nemici intermedi al primo livello, con velocità leggermente superiore.

  • **Yellow:** 1 nemico di difficoltà elevata al primo livello, che segue traiettorie rettangolari complesse.

 – Al termine delle tre ondate, il livello aumenta. Con l’avanzare del livello, i numeri dei nemici vengono incrementati (ad es. greenCount = 5 + (level–1)\*2, blueCount = 3 + (level–1), yellowCount = 1 + (level–1)/2), così come la loro velocità e la frequenza di sparo.

3. **Rilevamento delle Collisioni:**

 – Il sistema rileva le collisioni tra i proiettili del giocatore e i nemici, e tra i proiettili dei nemici e la navicella.

 – Le collisioni attivano un’animazione di esplosione centrata sull’oggetto colpito e assegnano un punteggio differenziato (10 punti per i Green, 20 per i Blue, 40 per i Yellow).

4. **Sistema di Punteggio e Classifica:**

 – Il punteggio viene aggiornato in tempo reale e visualizzato durante la partita.

 – Al termine della partita, il punteggio viene salvato per la creazione di una classifica degli HI-SCORE.

5. **Gestione del Game Over e delle Pause:**

 – La partita termina quando il giocatore esaurisce tutte le vite, mostrando una schermata di Game Over con il punteggio finale e opzioni per riprovare o tornare al menu.

 – Il gioco consente di mettere in pausa e riprendere la partita tramite un comando da tastiera.

6. **Audio e Grafica:**

 – Il sistema integra effetti sonori per spari, esplosioni e perdita di vita, insieme a una colonna sonora.

 – È possibile attivare/disattivare l’audio tramite un comando dedicato.

 – Lo sfondo scorre orizzontalmente da destra verso sinistra, creando un ambiente arcade dinamico.

**2.2 Requisiti Non Funzionali**

• **Fluidità e Prestazioni:** Il gioco deve girare a circa 60 FPS, garantendo aggiornamenti e rendering fluidi.

• **Portabilità:** Il sistema deve essere eseguibile su qualsiasi piattaforma dotata di JVM (Windows, macOS, Linux).

• **Modularità ed Espandibilità:** Il codice deve essere organizzato in package distinti (main, model, controller, view, utility) per facilitare manutenzione ed estensioni future.

• **Usabilità e Interfaccia:** L’interfaccia deve essere intuitiva, con menu chiari e una visualizzazione non invadente di punteggi, vite e HI-SCORE.

• **Robustezza e Sicurezza:** Il sistema deve gestire errori ed eccezioni in modo da garantire transizioni di stato fluide e operazioni sicure, anche durante il salvataggio e la lettura dei punteggi.

––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

**Discussione**

L’architettura adottata, pur non essendo una rigorosa implementazione del pattern MVC, consente di separare efficacemente le responsabilità: il **modello** (GameModel e le classi relative agli oggetti di gioco) gestisce lo stato e la logica, il **controller** (InputHandler, e in parte il GameController per funzioni specifiche) traduce gli input in azioni sul modello, e la **vista** (GamePanel e GameFrame) si occupa del rendering e dell’interfaccia utente. L’utilizzo di hitbox personalizzate per il rilevamento delle collisioni ha aumentato la precisione, evitando falsi positivi dovuti a bounding box troppo ampie. Il sistema di vite, che prevede 3 vite iniziali e un breve periodo di invulnerabilità dopo ogni impatto, offre al giocatore una possibilità di ripresa senza danni consecutivi. La generazione delle ondate di nemici, suddivisa in tre tipologie – **Green** (5 nemici, 10 punti ciascuno), **Blue** (3 nemici, 20 punti ciascuno) e **Yellow** (1 nemico, 40 punti) – garantisce una progressione della difficoltà, con l’incremento del livello che porta ad un aumento del numero dei nemici e della loro velocità. Lo sfondo dinamico, che scorre orizzontalmente da destra verso sinistra, insieme agli effetti sonori e alla possibilità di gestire l’audio, contribuisce a creare un ambiente arcade immersivo e moderno.

––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––––

**Conclusioni**

JSpaceInvaders dimostra che, partendo da un’architettura modulare e utilizzando concetti base di programmazione in Java, è possibile realizzare un videogioco 2D fedele all’esperienza arcade originale di Space Invaders. Il sistema, sviluppato con Swing, integra meccaniche moderne quali la gestione delle vite con invulnerabilità temporanea, la generazione progressiva di ondate di nemici con punteggi differenziati e l’incremento della difficoltà ad ogni livello. La solida struttura del codice, unita alla cura per gli aspetti grafici e sonori, garantisce fluidità, portabilità e manutenibilità, offrendo una base eccellente per futuri miglioramenti ed estensioni.