Documento di System Design
Pixel Arena

Riferimento	
Versione	0.8
Data	08/10/2024
Destinatario	Carmine Gravino
Presentato da	NC10: Antonio Ferrentino, Emanuele Rosapepe, Francesco Perilli
Approvato da	

Sommario

Team Composition

Nome	Matricola	Contatti
Antonio Ferrentino	0512113367	a.ferrentino50@studenti.unisa.it
Emanuele Rosapepe	0512113418	e.rosapepe2@studenti.unisa.it
Francesco Perilli	0512113802	f.perilli2@studenti.unisa.it

Revision History

Data	Versione	Descrizione	Autori
20/11/2023	0.1	Prima stesura	Emanuele Rosapepe Francesco Perilli
21/11/2023	0.2	Inizio completamento	Antonio Ferrentino Emanuele Rosapepe Francesco Perilli
25/11/2023	0.3	Aggiunta Component Diagram	Antonio Ferrentino Emanuele Rosapepe Francesco Perilli
27/11/2023	0.4	Aggiunta Persistenza dei dati e completamento Component Diagram	Antonio Ferrentino Emanuele Rosapepe Francesco Perilli
30/11/2023	0.5	Mapping, Accesso e Boundary	Emanuele Rosapepe Francesco Perilli
1/12/2023	0.6	Correzioni	Antonio Ferrentino
2/12/2023	0.7	Specifiche dei sottosistemi	Antonio Ferrentino Emanuele Rosapepe Francesco Perilli
8/10/2024	0.8	Ulteriori correzioni e revisione	Antonio Ferrentino Emanuele Rosapepe Francesco Perilli

1. Introduzione

1.1 Scopo del Sistema

Nel corso degli anni l'evoluzione del mondo videoludico è stata sia veloce che impressionante, spalancando le porte ad un'era d'oro per i videogiochi e per le aziende che ne trattano la loro creazione. La nostra idea è quella di portare alla luce un platform ortogonale con annessi combattimenti, il tutto rappresentato da una grafica pixelata: ciò potrebbe far sembrare il nostro gioco non molto dinamico rispetto ad altri giochi che si adattano perfettamente agli sviluppi sia hardware che software a cui abbiamo assistito in questi anni, unendo una grafica mozzafiato ad un'accuratezza nel gameplay. Ma per le dinamiche e per le sfide che il nostro gioco "Pixel Arena" ha intenzione di proporre, metterà sicuramente alla prova le abilità del videogiocatore, non negandogli anche il giusto divertimento.

1.2 Obiettivi di Design

In questo paragrafo vengono esposti i Design Goals ("Obiettivi di Design"), che descrivono le qualità chiave del sistema (quelle che devono essere ottimizzate) e stabiliscono le loro priorità.
Sviluppando in modo corretto i Design Goal sarà possibile stabilire un percorso di sviluppo ben delineato, solido e dettagliato.
Nella tabella che segue, i design goal sono descritti in base ai seguenti campi, ognuno corrispondente a una colonna:
• **Rank**: ne indica la priorità rispetto agli altri design goal;
• **ID Design Goal**: un identificatore;
• **Descrizione**: ne descrive le caratteristiche;
• **Categoria**: raggruppa i design goal che descrivono qualità che rientrano nella stessa macroarea;
• **Origine**: il requisito non funzionale da cui è stato generato;

In particolare, le categorie in cui sono stati suddivisi i design goal sono:
• **Performance**: indica, in maniera quantificabile, il livello di prestazioni del sistema software;
• **Dependability**: relativo all'affidabilità del sistema (gestione degli errori, resistenza ai
crash…);
• **Cost**: indice dell'impatto economico dell'implementazione di una qualità sui costi di
sviluppo;
• **Maintenance**: indica quanto effort è necessario per apportare modifiche al sistema dopo la
prima release;
• **End user**: relativo ai canoni dell'interazione uomo-macchina e di ingente importanza per
valutare la compatibilità di un sistema software con l'utenza.

Rank	ID	Descrizione	Categoria	RNF di origine
1	**DG_1 Usabilità**	Il sistema deve risultare facilmente utilizzabile secondo le "8 regole d'oro di Shneirderman"	End User	RNF_USE1 RNF_USE2
2	**DG_2 Performance**	Garantire che il gioco funzioni in modo fluido, mantenendo tempi di caricamento rapidi, non superiori a 5 secondi e tempi di risposta inferiori ad 1 secondo.	Performance	RNF_PRES 1
3	**DG_3 Affidabilità**	Assicurare che il gioco sia stabile e privo di bug, offrendo un'esperienza di gioco senza interruzioni	Dependability	RNF_AFF1
4	**DG_4 Manutenibilità**	Il software è facilmente manutenibile ed estensibile nel tempo grazie alla grande modularità dello stesso	Maintenance	RNF_SUPP1 RNF_SUPP2
5	**DG_5 Sicurezza**	Proteggere i dati degli utenti e dei salvataggi.	Affidability	RNF_AFF2
6	**DG_6 Immersione**	Creare un'esperienza visiva e sonora che immerga completamente i giocatori nel mondo di "Pixel Arena"	End User	RNF_USE3

Questi obiettivi di design assicurano che il gioco sia non solo tecnicamente solido, ma anche coinvolgente e accessibile a un vasto pubblico.

1.3 Trade-offs

Trade-off	Descrizione
Tempo di distribuzione vs Qualità	Essendo la deadline invalicabile, si è deciso di dare più importanza ai tempi di consegna; nonostante questo la qualità del software rimarrà entro determinati standard
Tempo di distribuzione vs Funzionalità	Le funzionalità a più bassa priorità potrebbero essere non implementate in caso di mancanza di tempo
Tempo di distribuzione vs Compatibilità	Essendo la deadline invalicabile, si è deciso di dare più importanza ai tempi di consegna, quindi il software sarà disponibile solo per PC all'inizio
Costi vs Prestazioni	Utilizzare risorse più avanzate o servizi gestiti può aumentare i costi, ma migliorare le prestazioni

1.4 Definizioni, acronimi e abbreviazioni

Acronimi	
SDD	System Design Document
RAD	Requirement Analysis Document
NPC	Non-Player Characters
HUD	Head-Up Display
AI	Intelligenza Artificiale

Abbreviazioni	
PS	Punti Salute
HP	Healt Point
ATK	Attacco
SPD	Speed

Definizioni	
Sottosistema	Corrisponde alla parte di lavoro che può essere svolta autonomamente da un singolo sviluppatore o da un gruppo di sviluppatori. Si ottiene decomponendo il sistema
Design Goal	Proprietà del sistema sulla quale ci si concentra maggiormente
Trade-off	Possibilità di ridurre una certa qualità per aumentare il valore di un'altra qualità e viceversa. Il termine è espresso talvolta come costo opportunità, riferendosi a più alternative alle quali si è preferito rinunciare a vantaggio di un'altra scelta

Definizioni	
Mapping Hardware-Software	Descritto per indicare i vari device hardware utilizzati dal sistema e la loro interazione con le componenti software
Dati Persistenti	Dati che devono sopravvivere all'esecuzione singola dell'applicazione. Sono i dati che vengono salvati nel database

1.5 Riferimenti

• Libro “Object-Oriented Software Engineering: Conquering Complex and Changing Systems”, terza edizione, Bernd Bruegge & Allen Dutoit, 2014.
Autori: Bernd Bruegge, Allen H. Dutoit
• SOW
• RAD

1.6 Panoramica

Il presente documento è strutturato in quattro sezioni principali:

- **Introduzione**: in questa sezione del documento è possibile trovare una descrizione dello scopo del sistema, i vari design goals ordinati per priorità ed arricchiti di descrizione, i trade-offs ed informazioni circa il linguaggio ed i riferimenti utilizzati.

- **Architettura del sistema corrente**: in questa sezione del documento è possibile analizzare l'architettura del sistema attualmente esistente.

- **Architettura del sistema proposto**: in questa sezione del documento è possibile analizzare tramite una descrizione accurata l'architettura del sistema da noi proposto.
Tale sezione è a sua volta suddivisa in:
  - decomposizione dei sottosistemi;
  - mapping hardware/software;
  - gestione dati persistenti;
  - controllo degli accessi e sicurezza, del flusso globale del sistema;
  - condizioni boundary.

- **Servizi dei sottosistemi**: in questa sezione del documento è possibile leggere la descrizione dei servizi di ogni sottosistema proposto.

- **Glossario**: sezione del documento che associa ad ogni sigla/termine menzionato/a all'interno del documento una definizione/descrizione per evitare che il lettore vada a ricercare informazioni all'esterno del documento, rendendolo autosufficiente sulla materia trattata.

2. Architettura del sistema corrente

Rimarcando il fatto che "Pixel Arena" sia un gioco del tutto nuovo, e che quindi non ha nessuna base di partenza per quanto riguarda l'architettura, riesce comunque idealmente a prendere spunto da vari titoli già presenti ed affermati nel panorama videoludico, come "The Legend of Zelda" e "Super Mario". "Pixel Arena" intende trarre da questi le funzionalità più influenti e divertenti per un'ottima giocabilità.

3. Architettura del sistema proposta

3.1 Overview

Il sistema da noi proposto è basato sull'architettura Three-Layer utilizzando come linguaggio di programmazione JAVA e il framework LibGDX. Tale modello architetturale è stato scelto in quanto ha il vantaggio di fornire una separazione logica delle componenti software così da aumentarne la manutenibilità, la scalabilità ed il riutilizzo del codice.

Con LibGDX ci occuperemo del motore di gioco, mentre per l'interfaccia, la logica del gioco e altre caratteristiche useremo JAVA.

3.2 Decomposizione in sottosistemi

Il sistema è stato decomposto nei seguenti sottosistemi:

- **Motore di Gioco (Game Engine)**: Gestisce le funzionalità di base come rendering grafico, fisica, animazioni e input dell'utente. Può includere moduli per la gestione delle risorse come texture, modelli 3D e suoni.
- **Logica di Gioco (Gameplay Logic)**: Contiene le regole del gioco, la gestione degli stati del gioco, e la logica degli eventi. Include sistemi per gestire il punteggio, i livelli, le condizioni di vittoria/sconfitta e le interazioni tra gli oggetti di gioco.
- **Intelligenza Artificiale**: Responsabile del comportamento degli NPC e degli agenti all'interno del gioco. Include algoritmi per il pathfinding, la presa di decisioni e l'adattamento al comportamento del giocatore.
- **Interfaccia Utente e HUD**: Gestisce la presentazione delle informazioni al giocatore e l'interazione con il gioco attraverso menu, pulsanti, indicatori di stato e dialoghi.
- **Audio e Musica**: Si occupa della gestione degli effetti sonori, della musica di sottofondo e del suono ambientale. Include moduli per il controllo del volume, la sincronizzazione audio con eventi di gioco e l'audio spaziale.
- **Persistenza e File di testo**: Responsabile della memorizzazione e del recupero dei dati di gioco, come progressi del giocatore, configurazioni e punteggi. Include sistemi per il salvataggio/caricamento di partite.

Component Diagram UML

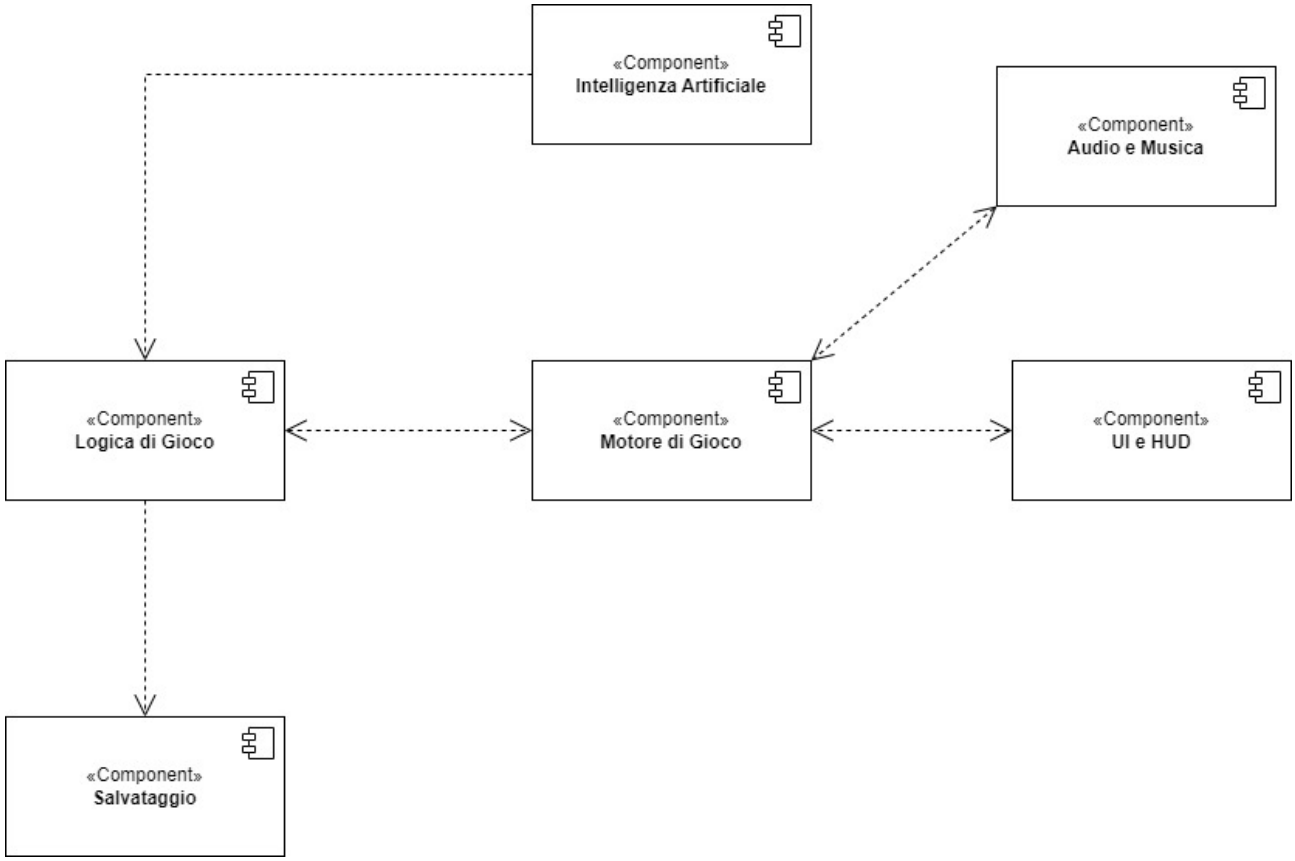
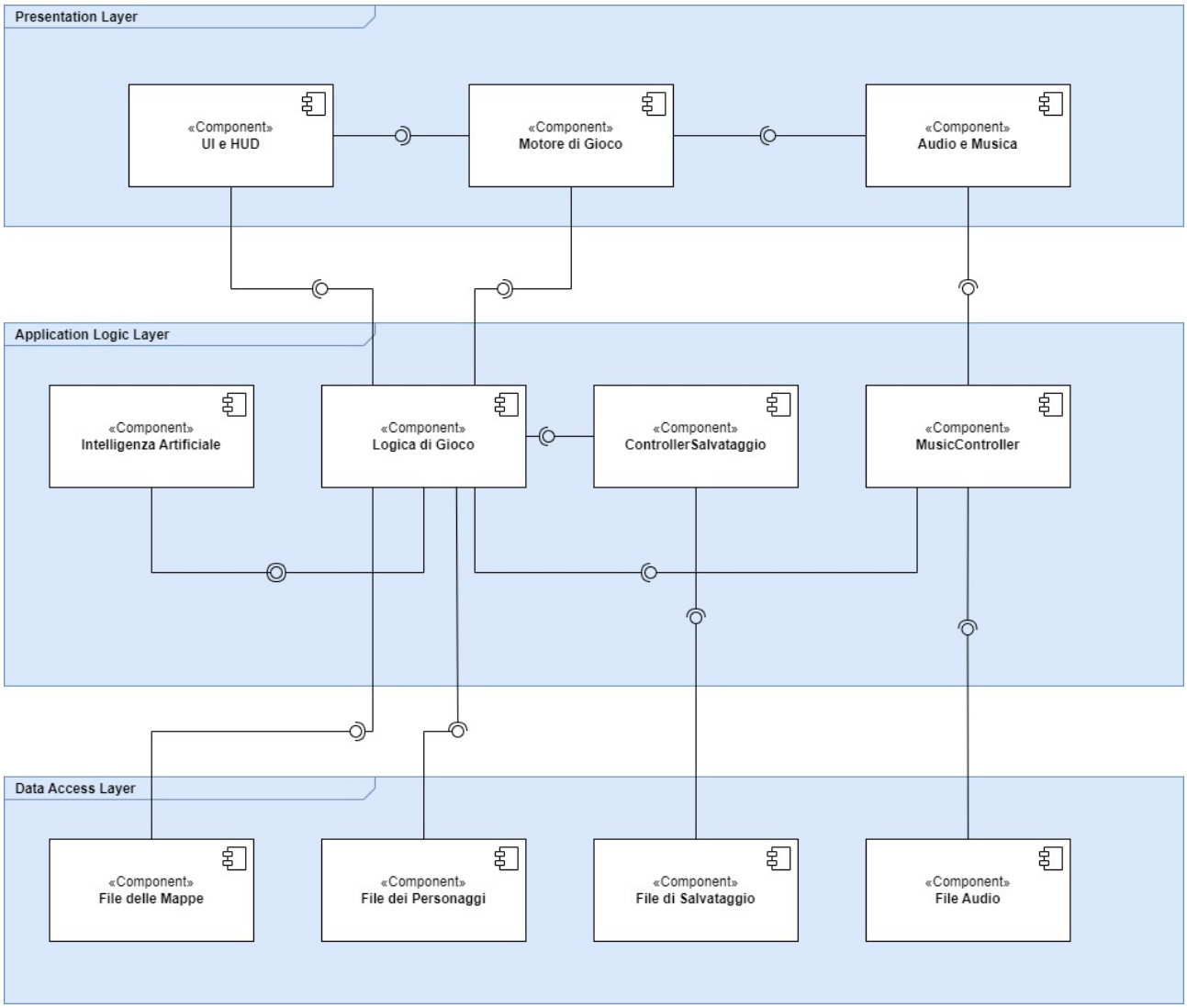


Diagramma architetturale



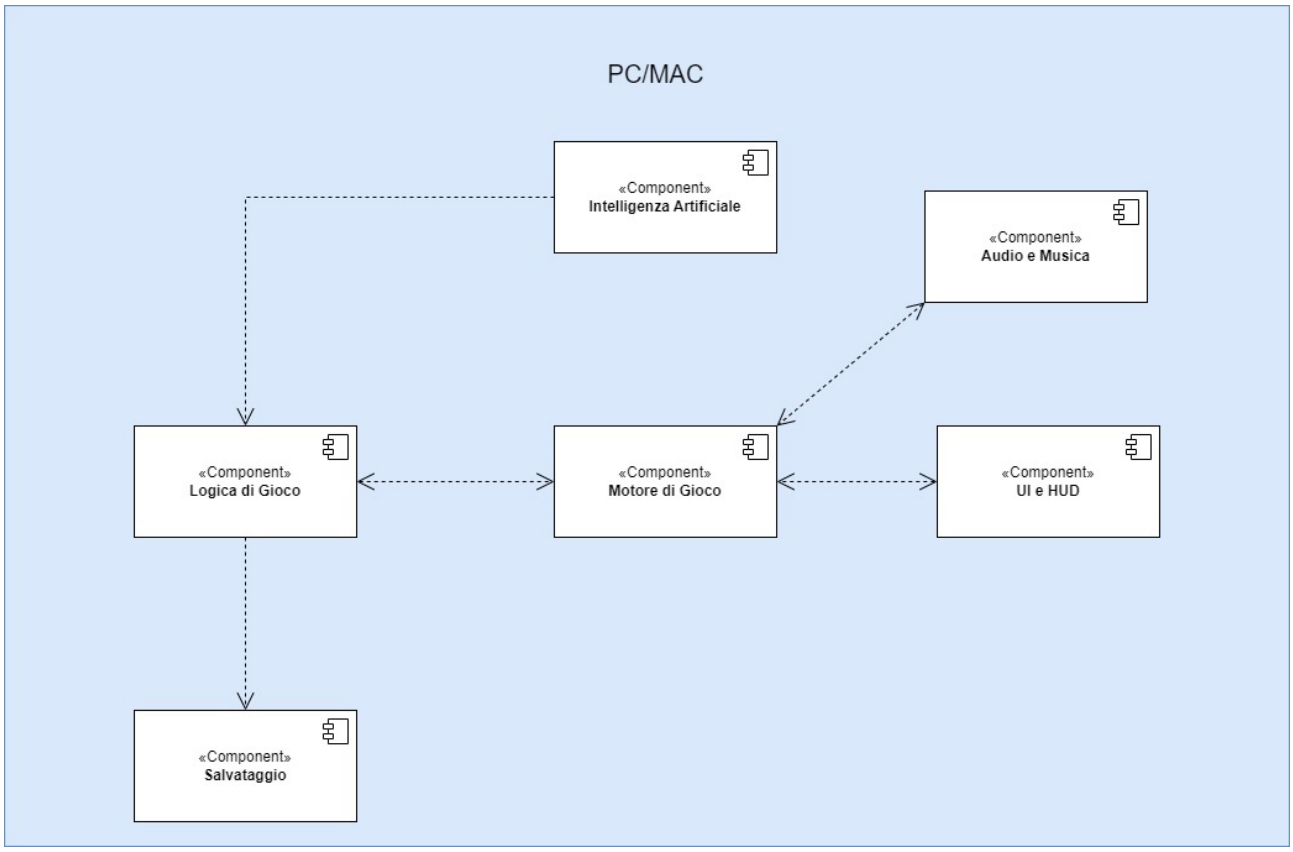
Con l'uso dell'architettura Three-layer abbiamo deciso di dividere i sottosistemi ottenuti in modo tale che questi possano interagire tra di loro in maniera coerente ed efficiente, rispettando le regole imposte dall'architettura usata.
Nel livello di Presentazione abbiamo inserito Il motore di gioco che andrà a mostrare a schermo tutto ciò che riguarda l'esperienza di gioco. Il Motore di Gioco si interfaccia con UI, HUD, video e musica, per poter integrare queste nel motore e renderle visibili/udibili.
Nel livello Applicazione, abbiamo la logica di gioco che governa la progressione e le meccaniche del gioco e fornisce gli eventi aggiornati al motore di gioco e all'UI/HUD. Inoltre, abbiamo una componente di Intelligenza Artificiale che controlla i personaggi non giocabili e prende decisioni basate sulla logica di gioco. Abbiamo due Controller che, comandati dalla logica di gioco, procedono ad interagire con i file di salvataggio.

Nel Livello Data Access abbiamo quattro componenti di raccolta file, che contengono tutti i file audio utilizzati per musica ed effetti sonori, tutti i file delle partite salvate dall'utente, tutti i file per le texture dei personaggi e quelli delle mappe.

3.3 Mapping Hardware/Software

Il gioco sviluppato sarà disponibile attraverso download dal sito GitHub, dove è possibile trovare tutta la documentazione e i file usati per lo sviluppo.
L'applicazione è pensata esclusivamente per essere supportata ed eseguita da PC, grazie all'uso di JVM. Ciò non limita però un futuro della stessa su altri dispositivi. L'esclusività della piattaforma di deployment per il gioco è stata dettata da una maggiore performance in termini di prestazioni, tempi di risposta e velocità di caricamenti.

Di seguito un UML Deployment Diagram che descrive il mapping hardware/ software:

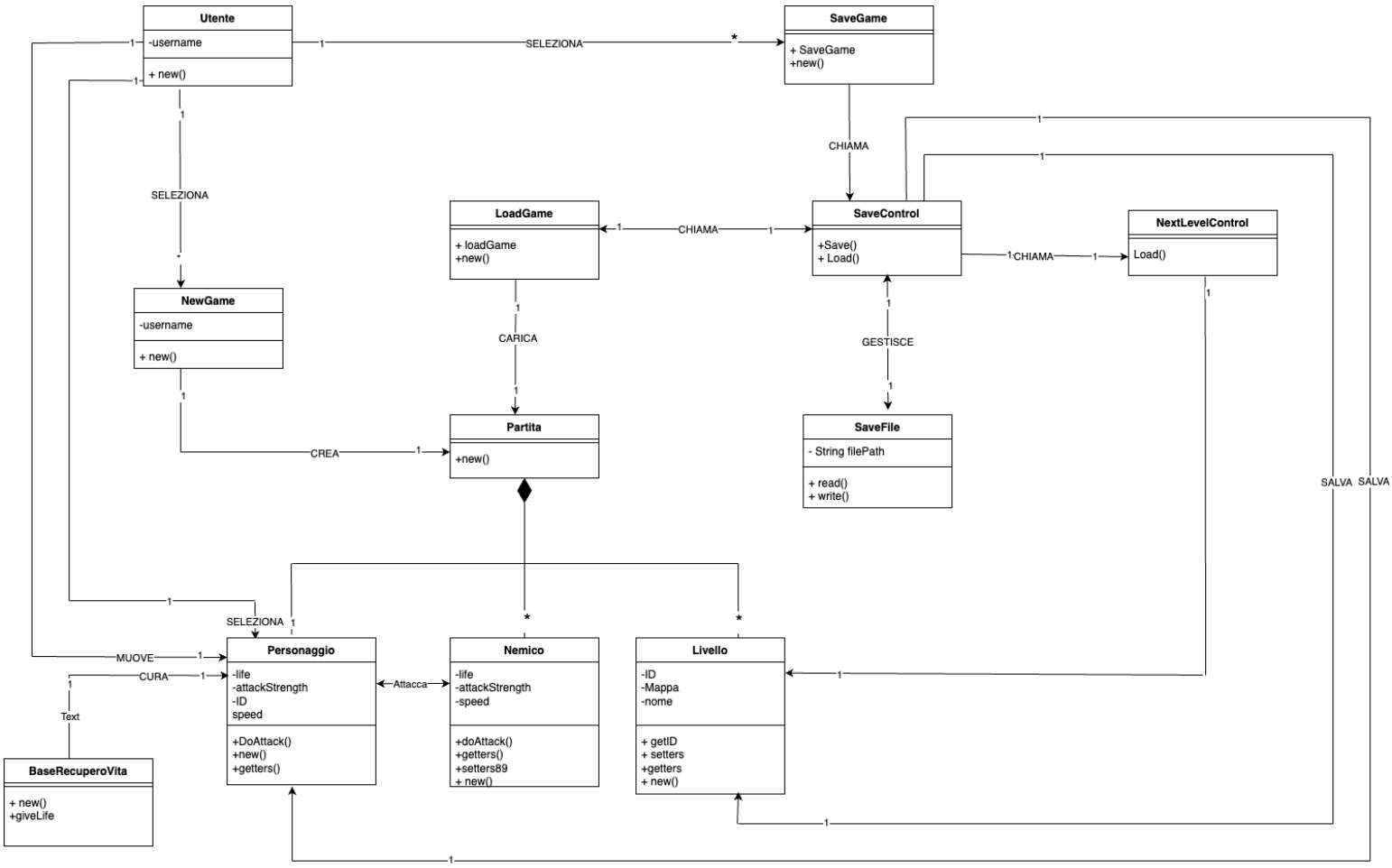


3.4 Gestione dati persistenti

Per la gestione dei dati persistenti del sistema abbiamo deciso di utilizzare un file di testo(XML), e questa decisione è stata dettata dalla voluta efficienza e velocità data dalla funzionalità di salvataggio e caricamento, che grazie a questo metodo di salvataggio è molto più rapido da effettuare, grazie alle poche statistiche che dovranno essere immagazzinate. Proprio la quantità minima di dati che devono essere salvati, ha portato ad una scelta diversa da un database relazionale, che poteva rallentare, anche se di poco, l'efficienza di questa funzionalità.

L'unica nota a sfavore di questo sistema di salvataggio potrebbe essere la sicurezza maggiore fornita da un database rispetto ad un file di testo, ma la locazione del file nei file system del gioco garantisce comunque una sicurezza ottima.

Class Diagram



3.5 Controllo degli accessi e sicurezza

Oggetti \ Attori	Giocatore
Intelligenza Artificiale	-
Logica di gioco	Movimento Attacco Cura SceltaPotenziamenti
Motore di gioco	-
Salvataggio	SalvaPartita CaricaPartita
UI e HUD	-
Audio e Musica	-

3.6 Controllo del flusso globale del sistema

Pixel Arena è ovviamente un gioco interattivo: attraverso degli input dati da tastiera il giocatore potrà vedere ciò che succede attraverso un'interfaccia grafica ed un motore di gioco. Una volta dato un input, questo verrà processato dalla logica di gioco che poi reindirizzerà verso altri componenti. Ci sono inoltre input dati da mouse, presenti nelle varie schermate di selezione che si presentano al giocatore durante il gioco.

3.7 Condizioni Boundary

In questo paragrafo verranno illustrati le condizioni boundary riguardanti il fallimento del sistema, l' errore di accesso ai file di salvataggio, avvio e chiusura del sistema.

Avvio del sistema

Identificativo	*UCBC1- Avvio del Sistema*	*Data*	30/11/23
		Vers.	1.0
		Autore	Francesco Perilli
Descrizione	Avvio del sistema da parte del giocatore		
Attore Principale	Giocatore		
Attori secondari	NA		
Entry Condition	Il Gioco è installato		
Exit condition On success	Il Gioco si avvia correttamente		
Exit condition On failure	Il Gioco non viene avviato correttamente		

FLUSSO DI EVENTI PRINCIPALE		
1	Giocatore	Clicca due volte l'icona del gioco.
2	Sistema	Esegue l'applicazione.

SCENARIO/FLUSSO DI EVENTI ALTERNATIVO		
2.a1	Sistema	Notifica al giocatore che non è stato possibile avviare il gioco.
2.a2	Giocatore	Riparte dal punto 1.

Spegnimento del sistema

Identificativo	*UCBC2- Spegnimento del Sistema*	*Data*	30/11/23
		Vers.	1.0
		Autore	Emanuele Rosapepe
Descrizione	Spegnimento del sistema da parte del giocatore		
Attore Principale	Giocatore		
Attori secondari	NA		
Entry Condition	Il gioco è avviato		
Exit condition On success	Il gioco si arresta correttamente		
Exit condition On failure	Il gioco non viene arrestato correttamente		

FLUSSO DI EVENTI PRINCIPALE		
1	Giocatore	Clicca sul pulsante di spegnimento
2	Sistema	Arresta il gioco

Crash del sistema

Identificativo	*UCBC3- Crash del sistema*	*Data*	30/11/23
		Vers.	1.0
		Autore	Antonio Ferrentino
Descrizione	Il sistema smette di funzionare improvvisamente		
Attore Principale	Giocatore		
Attori secondari	NA		
Entry Condition	Il Gioco non risponde ai comandi		
Exit condition On success	Il gioco viene riavviato dall'utente		
Exit condition On failure	Il gioco non può essere riavviato		

FLUSSO DI EVENTI PRINCIPALE		
1	Sistema	Notifica che c'è un errore
2	Giocatore	Tenta di riavviare il gioco

Errore di accesso ai dati persistenti

Identificativo	*UCBC4- Errore di accesso ai dati persistenti*	*Data*	30/11/23
		Vers.	1.0
		Autore	Francesco Perilli
Descrizione	Comportamento dell'utente in caso di errore di accesso ai dati persistenti da parte del gioco		
Attore Principale	Giocatore		
Attori secondari	NA		
Entry Condition	Il gioco non riesce ad accedere ai dati persistenti OR I dati risultano corrotti		
Exit condition On success	Il Gioco riprende il suo normale funzionamento		
Exit condition On failure	Il Gioco non riprende il suo normale funzionamento		
FLUSSO DI EVENTI PRINCIPALE			
1	Sistema	Restituisce un messaggio di errore e rimanda alla schermata di salvataggio	

4. Servizi dei sottosistemi

In questa sezione vengono descritti i servizi di ogni sottosistema precedentemente elencati.

Sottosistema Intelligenza Artificiale

Servizio	Descrizione	Interfaccia
MovimentoNPC	Il seguente servizio permetterà all' NPC di muoversi e sapersi orientare nella mappa con l'obbiettivo di trovare il personaggio principale	NPCServices
AttaccoNPC	Il seguente servizio permetterà all'NPC di attaccare il personaggio principale in maniera coerente	NPCServices

Sottosistema Motore di Gioco

Servizio	Descrizione	Interfaccia
RenderizzaGioco	A partire dalle informazioni dei sottosistemi collegati mostra a schermo l'ambiente, personaggi, audio, musica e UI/HUD del gioco, aggiornati ad ogni frame	GameRender

Sottosistema Logica di Gioco

Servizio	Descrizione	Interfaccia
InizializzazioneLivello	Carica la mappa del livello e le posizioni iniziali del personaggio principale ed NPCs	LoadLevel
Movimento del personaggio principale	Gestisce il movimento del personaggio in quattro direzioni, caricando l'animazione corretta che viene fornita al motore di gioco per essere renderizzata	CharacterServices
Attacco del personaggio principale	Gestisce la logica di attacco del personaggio principale, caricando l'animazione corretta che viene fornita al motore di gioco per essere renderizzata	CharacterServices
Recupero vita del personaggio principale	Gestisce la logica per il recupero dei punti vita del personaggio principale nel caso in cui esso dovesse posizionarsi sull'apposita piattaforma	CharacterServices
Servizio di salvataggio dati	Prende i dati da salvare e li inoltra al sotto-sistema di salvataggio	DataServices
Servizio di caricamento dati	Prende i dati inoltrati dal sotto-sistema di salvataggio e li utilizza per gli altri servizi del sotto-sistema	DataServices
Seleziona Audio	In base a gli eventi che accadono nel gioco indica al sotto-sistema musica quale Audio caricare	AudioService
Raccolta gemme	Gestisce	DataServices

Sottosistema UI e HUD

Servizio	Descrizione	Interfaccia
Menu principale	Offre la visualizzazione di un menu all'utente con varie opzioni di scelta tra cui "Nuova partita", "Carica Partita", ed invia le scelte al sotto-sistema logica di gioco	UIServices
Schermata di salvataggio e potenziamento	Offre la visualizzazione di una schermata con due opzioni di salvataggio e tre per il potenziamento delle caratteristiche del personaggio, ed invia le scelte al sotto-sistema logica di gioco	UIServices
Menu scelta personaggio	Offre la visualizzazione di diversi personaggi selezionabili, ed invia la scelta al sotto-sistema di logica	UIServices
HUD Contatore Gemme	Visualizza a schermo il numero di gemme raccolte dal personaggio principale	HUDServices
HUD Nemici Mancanti	Visualizza a schermo il numero di nemici mancanti.	HUDServices
HUD Vita Personaggio Principale	Visualizza a schermo la vita del personaggio principale	HUDServices

Sottosistema Audio e Musica

Servizio	Descrizione	Interfaccia
CaricaMusica	Carica i dati presi dai file audio e li invia al motore di gioco	AudioServices

Sottosistema Salvataggio

Servizio	Descrizione	Interfaccia
CaricaSalvataggio	Prende i dati da il File di salvataggio e li manda al sottosistema logica di gioco	DataServices
ScriviSalvataggio	Prende i dati dal sottosistema della Logica di Gioco e li scrive sul file di salvataggio	DataServices

5. Glossario

Termine	Definizione
Platform	Il videogioco a piattaforme è un sottogenere dei videogiochi d'azione dove la meccanica di gioco implica principalmente l'attraversamento di livelli costituiti da piattaforme, spesso disposte su più piani
Giocatore	Il giocatore, al femminile giocatrice, è il partecipante a un gioco, come dedizione momentanea o come praticante assiduo. Nell'ambito dei videogiochi si usa anche il termine videogiocatore
Pixel	Con il termine pixel, o px, si fa riferimento all'unità minima di cui si compone un'immagine digitale
Bug	È un'anomalia che porta al malfunzionamento di un software, per esempio producendo un risultato inatteso o errato, tipicamente dovuto a un errore nella scrittura del codice sorgente di un programma
Framework	È un'architettura logica di supporto sulla quale un software può essere progettato e realizzato, spesso facilitandone lo sviluppo da parte del programmatore
Crittografia	È la branca della crittologia che tratta delle "scritture nascoste", ovvero dei metodi per rendere un messaggio non comprensibile/intelligibile a persone non autorizzate a leggerlo, garantendo così, in chiave moderna, il requisito di confidenzialità o riservatezza tipico della sicurezza informatica