Test Plan
Pixel Arena

Riferimento	
Versione	0.3
Data	18/01/2024
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
4/12/2023	0.1	Prima stesura	Antonio Ferrentino Emanuele Rosapepe Francesco Perilli
8/12/2023	0.2	Category Partition	Antonio Ferrentino Emanuele Rosapepe Francesco Perilli
18/01/2024	0.3	Revisione e correzioni	Antonio Ferrentino Emanuele Rosapepe Francesco Perilli

1. Introduzione

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.

In questo documento andremo a documentare:

- Le caratteristiche da testare;
- L'approccio al Testing;
- I criteri di vario tipo
- Test case

Con l'obiettivo di individuare le carenze di correttezza, completezza e affidabilità delle componenti software in corso di sviluppo e poter consegnare al committente un sistema adeguato e senza malfunzionamenti.

2. Documenti correlati

Il presente documento è in stretta relazione con i documenti precedentemente rilasciati. Esso è in forte relazione anche con documenti che verranno sviluppati e rilasciati in futuro. I test case sono basati sulle funzionalità individuate nel documento di raccolta ed analisi dei requisiti.

Relazione con il RAD

I test case pianificati nel TP sono stati creati basandosi sui casi d'uso o sugli scenari presenti nel RAD.

Relazione con il SDD

I test case pianificati nel TP sono collegati alla suddivisione in sottosistemi presente nell'SDD.

3. Panoramica del sistema

Come definito nel System Design Document, il sistema da noi proposto è basato sull'architettura Three-Layer utilizzando come linguaggio di programmazione JAVA e il framework LibGDX.

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

Il sistema è stato decomposto nei seguenti sottosistemi:

- Motore di Gioco (Game Engine);
- Logica di Gioco (Gameplay Logic);
- Intelligenza Artificiale;
- Interfaccia Utente e HUD;
- Audio e Musica;
- Persistenza e File di testo;

4. Funzionalità da testare

Le funzionalità da testare sono relative ai casi d'uso e agli scenari definiti nel Requirement Analysis Document.

- Rigenerazione vita
- Attacco
- Raccolta Gemme

5. Approccio

In base ai vincoli progettuali, questi sono i punti su cui preme maggiormente il testing:
• Ogni studente dovrà effettuare il **testing di unità**, tramite category partition, di esattamente un metodo di una classe sviluppata;
• Ogni studente dovrà effettuare il **testing di sistema**, tramite category partition, di esattamente una funzionalità del sistema sviluppato.

Principali strumenti utilizzati per il testing di unità sono Junit, Mockito, e Jacoco.

Per quanto riguarda il testing di sistema, verrà effettuato manualmente dai Team Members, controllando che i risultati siano quelli definiti nei casi di test.

6. Criteri Pass/Fail

I criteri determinanti per un pass della fase di testing sono l'individuazione di una failure, che si verifica quando l'output osservato sarà diverso dall'oracolo, ossia l'output atteso.

La fase di testing avrà esito fail se l'output osservato sarà lo stesso dell'oracolo.

7. Criteri di sospensione e ripristino

In questa sezione verranno illustrati i criteri di sospensione del test e le attività di test che dovranno essere ripetute quando si riprende il test. Essendo una fase delicata, essa può causare imprevisti e slittamenti dei tempi a causa di errori e malfunzionamenti del sistema.

Criteri di sospensione

Il testing non verrà sospeso fino alla sua terminazione, anche in caso di rilevazione di una failure. Il testing potrà essere momentaneamente sospeso nel caso venga restituito, al momento dell'esecuzione, un errore nella definizione di uno dei test stessi.

Criteri di ripristino

Il testing verrà ripreso solo quando tutti i problemi relativi alla sospensione dello stesso sono stati risolti. L'attività di testing riprenderà dal test case che ha causato la sospensione.

8. Test Cases

In questa sezione viene rappresentata l'applicazione del metodo "Category Partition", utile ad individuare i test-case del sistema partizionando i possibili parametri in ingresso in categorie.

8.1 Rigenerazione vita

Parametro: HP	
Nome categoria	**Scelte per la categoria**
Valore [VH]	1. Valore > 0 AND Valore < 100 [PROPERTY_VH_OK] 2. Valore <= 0 AND Valore >= 100 [errore]

Test Case ID	Test frame	Esito
TC_1.1	VH1	Corretto
TC_1.2	VH2	Errato: HP fuori dal valore prestabilito

8.2 Attacco

Parametro: HP	
Nome categoria	**Scelte per la categoria**
Valore [VH]	1. Valore > 0 AND Valore < 100 [PROPERTY_VH_OK] 2. Valore <= 0 AND Valore >= 100 [errore]
Parametro: ATK	
Nome categoria	**Scelte per la categoria**
Valore [VA]	1. Valore > 0 AND Valore <= 50 [PROPERTY_VA_OK] 2. Valore <= 0 AND Valore > 50 [errore]

Test Case ID	Test frame	Esito
TC_2.1	VH1, VA1	Corretto
TC_2.2	VH1, VA2	Errato: ATK fuori dal valore prestabilito
TC_2.3	VH2, VA1	Errato: HP fuori dal valore prestabilito
TC_2.4	VH2, VA2	Errato: HP e ATK fuori dal valore prestabilito

8.3 Raccolta Gemme

Parametro: ContatoreGemme	
Nome categoria	**Scelte per la categoria**
Range [RN]	1. Range >= 0 [PROPERTY_VA_OK] 2. Range < 0 [errore]

Test Case ID	Test frame	Esito
TC_3.1	RN1	Corretto
TC_3.2	RN2	Errato: ContatoreGemme fuori dal valore prestabilito

9. Strumenti per il testing

Per l'attività di Testing è necessario un computer con l'applicazione pronta all'uso. Per effettuare il testing c'è bisogno:

- Dell'utilizzo dei vari strumenti per il testing Junit, Mockito, e Jacoco;
- Di una simulazione del sistema delle 3 funzionalità (una per Team Member) per il testing di sistema.