Laurea Triennale in informatica - Università di Salerno
Corso di *Ingegneria del Software* - Prof. Carmine Gravino

Test Plan

Dashing Cube

Riferimento	Nc11_dashingcube-tp.docx
Versione	0.4
Data	08/12/2024
Destinatario	Prof. Carmine Gravino
Presentato da	Vincenzo Beniamino Fresa, Francesco Botta
Approvato da	

Revision History

Data	Versione	Descrizione	Autori
08/12/2024	0.1	Stesura dell'introduzione	Francesco Botta Vincenzo Beniamino Fresa
10/12/2024	0.2	Completamento prima stesura documento	Francesco Botta
20/12/2024	0.3	Prima Revisione del Documento	Francesco Botta Vincenzo Beniamino Fresa
09/01/2025	0.4	Seconda Revisione del Documento	Francesco Botta Vincenzo Beniamino Fresa

1 Introduzione

L'interesse per il mercato videoludico è alle stelle: il medium è diventato la forma di intrattenimento più remunerativa sul mercato e nel panorama italiano non sono presenti abbastanza prodotti tali da soddisfare l'attuale richiesta.

Dashing Cube si propone come prodotto tale da poter catturare l'attenzione di un vasto pubblico, in grado non soltanto di ottenere una vasta utenza ma volenteroso di essere un sistema facile da programmare e costruire.

Seppur il gioco non possa sembrare originale, il prodotto si pone come un'idea che possa piacere sia per i neofiti del campo che per i veterani ispirandosi a diversi prodotti già presenti nel settore.

Questo documento si pone ad elencare le caratteristiche da provare in fase di test con l'approccio utilizzato e i criteri di vario tipo. Ciò sarà utile per fornire un prodotto pulito e privo di errori.

2 Documenti Correlati

Il presente documento sfrutterà diversi punti già menzionati precedentemente nella descrizione del sistema.

Fungerà anche da tramite per le specifiche dei testing da analizzare. I test case sono basati sulle funzionalità individuate nel documento di raccolta ed analisi dei requisiti:

-dal RAD, vengono estrapolati diversi scenari e use case da mettere sotto esame
-dal SDD, viene sfruttata la suddivisione in sottoinsiemi del sistema per la creazione dei test.

3 Panoramica del sistema

Il sistema sfrutta l'architettura Three-Layer sfruttando l'engine Unity, ideale per il Game Design, che gestirà le varie caratteristiche grafiche del videogioco. Il linguaggio utilizzato è quello C# e si utilizzeranno funzioni di Database sfruttando MySQL. Sarà lo stesso engine Unity a gestire le fasi di testing.
Il sistema è stato decomposto nei seguenti sottosistemi:

- Motore di Gioco (Unity);
- Logica di Gioco (Gameplay Logic);
- Intelligenza Artificiale;
- Interfaccia Utente;
- Audio e Musica;
- Memoria Fissa;

4 Funzionalità da Testare

Le funzionalità da testare sono relative ai casi d'uso e agli scenari definiti nel RAD:

- Salto del Giocatore
- Selezione di un Cosmetico
- Acquisto di un Cosmetico

5 Criterio generale del test

Il testing viene eseguito per rilevare eventuali anomalie del sistema proposto per poi andarli a correggere con specifiche modalità d'approccio e strumenti.

Il risultato atteso dei vari test case è confrontato con un oracolo (specifica del risultato atteso secondo i criteri progettuali, vincoli e requisiti funzionali richiesti).

In caso di successo, ovvero la situazione in cui c'è corrispondenza tra oracolo e test case, si ha il successo del test; nel caso contrario si ha un fallimento.

Per cercare di minimizzare i difetti del software, si sa che c'è bisogno di una continua e precisa attività di verifica e convalida che va effettuata e applicata durante tutta la progettazione; per questo, il test ha dei requisiti ben precisi per essere ritenuto valido:

- Testare tutti i requisiti funzionali elencati nel punto 4;
- Eseguire test di regressione nel momento in cui si vanno ad aggiungere altre caratteristiche al sistema oppure modificando quelle già comprese in precedenza.

6 Approccio

Il testing verrà approcciato sfruttando i seguenti punti:

-Testing di unità: il team, tramite category partition e conoscenza del codice, testerà diversi metodi del sistema per accertarsi del corretto funzionamento di esso.

-Testing di sistema: il team, tramite category partition, verifica l'affidabilità del sistema osservando il comportamento front-end rispetto a ciò che si prevede; in questa fase, non è di primaria importanza avere a disposizione il codice sorgente (black box).

7 Criteri di sospensione e ripristino

Criteri di sospensione

Il testing continuerà fino alla sua terminazione, anche in caso di rilevazione di una failure. Il testing potrà essere momentaneamente sospeso nel caso venga deciso manualmente da chi gestisce il test.

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

Per la creazione di Test Cases sfrutteremo la tecnica di "Category Partition", dove si identificheranno per le funzionalità da testare dei parametri. Ogni partizione sarà disgiunta dalle altre ed ogni parametro avrà una caratteristica in comune con gli altri parametri presenti nella partizione.

I test cases verranno definiti nel documento di Test Cases Specification (TCS).

8.1 Salto del Giocatore

Parametro: CoordinataY	
Nome categoria	**Scelte per la categoria**
Validità [VA]	1. onTheField = true [PROPERTY_VA_OK] 2. onTheField = false [errore]

Test Case ID	Test frame	Esito
TC_1.1	VA1	Corretto
TC_1.2	VA2	Errato: Giocatore in Aria

8.2 Selezione di un Cosmetico

Parametro: CosmeticoPosseduto	
Nome categoria	**Scelte per la categoria**
Validità [VA]	1. Valida [PROPERTY_VA_OK] 2. Non Valida

Test Case ID	Test frame	Esito
TC_2.1	VA1	Corretto
TC_2.2	VA2	Errato: Cosmetico non acquisito o sbloccato

8.3 Acquisto di un Cosmetico

Parametro: Monete	
Nome categoria	**Scelte per la categoria**
Validità [VA]	1. Valida [PROPERTY_VA_OK] 2. Non Valida
Correttezza [CD]	1. Monete >= costoCosmetico [PROPERTY_CD_OK] 2. Monete < costoCosmetico [errore]

Test Case ID	Test frame	Esito
TC_3.1	VA1	Corretto
TC_3.2	VA2	Errato: Cosmetico già sbloccato
TC_3.3	CD1	Corretto
TC_3.4	CD2	Errato: Numero di monete insufficiente