ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

COME SI PROGETTA UN GIOCO IN REALTÀ AUMENTATA COOPERATIVA USANDO WEBXR

Elaborato in
SISTEMI EMBEDDED ED INTERNET OF THINGS

Relatore
Prof. ALESSANDRO RICCI

Corelatore
Dott. Ing. SAMUELE BURATTINI

Presentata da
FILIPPO GURIOLI

Anno Accademico 2022 – 2023

Alla mia famiglia.

Indice

Introduzione

I videogiochi sono sempre stati una componente fondamentale che accompagna lo sviluppo teconologico, a partire da uno dei primi computer della storia, il DEC PDP-1, che nel 1961 ospitava il primo videogioco della storia, *"Spacewar!"*, fino ad arrivare ai giorni nostri, in cui il settore videoludico si è espanso in ogni sorgente tecnologica che ci circonda, dallo smartphone, al computer, fino alle console. Da quello che si potrebbe definire uno svago dei programmatori è nata una disciplina a tutti gli effetti, suddivisa su varie tematiche: dal game design, al game programming, fino alla computer graphics. Per ognuna di queste branche la letteratura è vastissima e in continua evoluzione. Questa tesi si pone l'obiettivo di estendere la letteratura riguardante il game programming, ovvero quella disciplina che si occupa di gestire la logica del modello di gioco, in particolare, si vuole esplorare un nuovo campo di ricerca che sta prendendo sempre più piede: la realtà aumentata.
L'arrivo di nuove tecnologie ha sempre portato a nuovi modi di giocare, e con la realtà aumentata non è stato da meno. La realtà aumentata è una tecnologia che permette di sovrapporre elementi della realtà che ci circonda con elementi virtuali, creando un'esperienza ibrida, unica nel suo genere.
Come ogni grande scoperta nell'ambito informatico, anche l'arrivo della realtà aumentata ha portato nuove sfide per i programmatori, in particolare, nella scena videoludica e non, un campo di ricerca tutt'ora molto attivo è la condivisione dell'esperienza.
È da queste premesse che nasce il progetto di tesi, nelle prossime pagine si vuole infatti definire pattern noti e best practice per lo sviluppo di videogiochi in realtà aumentata cooperativa tramite l'analisi di un caso di studio particolare: il gioco di carte *"Yu-Gi-Oh!"*.

Capitolo 1

Stato dell'arte

Di seguito si riporteranno informazioni circa lo stato dell'arte riguardante tecnologie e metodologie utilizzate per lo sviluppo di videogiochi e come vengano adattate alla realtà aumentata. Si affronteranno le problematiche della gestione cooperativa dell'esperienza di gioco e quali tecnologie ne permettano la realizzazione. Infine si darà una panoramica delle strutture messe a disposizione per creare mondi virtuali attualmente presenti in letteratura.

1.1 Game Design Pattern

In questa sezione si farà principalmente riferimento al libro *"Game Programming Patterns"* di Robert Nystrom[2], in cui vengono descritti i pattern più comuni nel game design. Nel testo vengono citate 5 macrocategorie di pattern:

- Design Pattern rivisitati;
- Sequencing pattern;
- Pattern comportamentali;
- Pattern di disaccoppiamento;
- Pattern di ottimizzazione.

Di seguito si riporta una breve descrizione per ognuna di esse e se ne analizzano i pattern più significativi.

1.1.1 Design Pattern rivisitati

I design pattern sono i pattern più comuni e noti, sono stati descritti per la prima volta nel libro *"Design Patterns: Elements of Reusable Object-Oriented Software"* [1]. Nel libro di Nystrom vengono rivisitati alcuni di questi pattern in chiave videoludica ed in particolare si analizzano i seguenti: Command, Flyweight, Observer, Prototype, Singleton e State. Data la vastità di argomenti che copre quel libro non verranno analizzati tutti, bensì solo quelli che si ritengono più rilevanti per questo progetto di tesi.

Command Pattern Il command pattern è definito nel libro come *"la reificazione di una chiamata di funzione"*, in pratica si tratta di modellare una classe che permetta di creare un livello di astrazione tra quando un comando viene invocato e quando viene eseguito. Si crea un'interfaccia 'Command' che specifica solo un metodo 'execute', dopodichè si creano delle classi che implementano tale interfaccia e che rappresentino un comando ben specifico (e.g. salta, corri, attacca etc.). Nella sua essenza il command pattern è quanto appena detto ma il modo in cui viene utilizzato è forse la cosa più interessante: nell'ipotetica classe 'InputHandler' si potrà dichiarare una istanza della classe 'Command' per ogni tipo di interazione che l'utente può eseguire (e.g. premere un tasto, drag and drop, gestures di vario genere etc.) ed assegnare a questo oggetto una qualsiasi delle classi precedentemente definite, svincolando così l'interazione dall'esecuzione di un determinato evento. I vantaggi del command pattern, spiega Nystrom, non si fermano solo alla possibilità di creare il key binding ma, serializzando i comandi, si possono mandare in rete per realizzare un gioco multiplayer. Si lascia il codice 1.1 come esemplificazione di quanto appena spiegato.

```
interface Command {
    execute(): void;
}

class Jump implements Command {

    /*stuff*/

    execute() {
        character.jump();
    }
}

class InputHandler {
    buttonX: Command;
```

```
    constructor() {
        //some logics that decide which key bind to which command
        this.buttonX = new Jump();
    }

    handleInput(): void {
        if (isPressed(Key.X)) {
            this.buttonX.execute();
        }
    }
}
```

Listato 1.1: Codice d'esempio per l'implementazione del command pattern

Observer pattern L'observer pattern è uno dei più famosi e più utilizzati pattern presenti in letteratura. La stessa Microsoft ne ha fornito una implementazione all'interno del framework .NET. L'observer pattern si basa sui concetti di *evento* e *osservatore*: quando un oggetto modifica il suo stato e produce un qualche risultato visibile genera un *evento* il quale può essere osservato dall'*osservatore* che, a sua volta, può reagire, eseguendo del codice, all'evento stesso. Questa idea ribalta il concetto normalmente utilizzato in cui per permettere ad un oggetto B di reagire ad un cambiamento di stato di un oggetto A, quest'ultimo deve avere un riferimento all'oggetto B. Questo pattern crea un layer di astrazione tra l'oggetto che genera l'evento e l'oggetto che vuole reagirci svincolando il generatore dal conoscere l'osservatore. Nel contesto di un videogioco questo pattern è spesso usato in modo verticale su tutta la codebase, a partire dalla gestione generale di eventi di gioco fino ad arrivare alla gestione di contesti particolari come la gestione degli input, degli achievements o delle collisioni./*qui inserire il fatto che l'applicazione del pattern potrebbe sfuggire di mano*/ Nystrom spiega che il pattern andrebbe applicato ogni qualvolta l'oggetto che genera l'evento non ha senso che conosca l'osservatore. Si lascia il codice **??** come esempio di implementazione del pattern observer.

1.2 Framework per la gestione cooperativa

1.3 Framework per sviluppo 3D

Capitolo 2

Progettazione e tecnologie

Analisi dei requisiti + tecnologie utilizzate.

2.1 Analisi dei requisiti

L'obiettivo del progetto è creare un'ambiente di realtà aumentata condivisa in cui l'utente possa giocare contro un altro al gioco di carte Yu-Gi-Oh. L'esperienza che il giocatore proverà dovrà essere quanto più simile alla versione proposta nella serie animata omonima.

Al momento dell'avvio l'utente dovrà affrontare un duello contro un'altra persona a Yu-Gi-Oh. Per la decisione del regolamento da seguire si è optato per una versione semplificata del gioco. Il giocatore potrà giocare carte mostro che rappresentano delle truppe schierate dalla parte del possessore. Queste truppe potranno quindi attaccare l'avversario per ridurne i punti vita. Saranno presenti anche carte magia e trappola che, tra i vari effetti, potranno modificare i punti vita, l'ambiete di gioco in cui gli utenti giocano o anche l'attacco e la difesa dei mostri propri e avversari. L'obiettivo del gioco consiste quindi nell'azzerare i punti vita dell'avversario, che comporterà la conclusione della simulazione.

2.1.1 Requisiti funzionali

- Il giocatore sarà in grado di vedere gli ologrammi propri e dell'avversario in tempo reale;

- il giocatore potrà interagire con un mazzo di carte virtuale pescando la prima carta;

- il giocatore potrà posizionare le carte che ha in mano sul campo e di conseguenza far apparire la corrispondente carta nello spazio di gioco condiviso;
- l'utente potrà ordinare l'attacco di un mostro, come attivare effetti di carte o passare il turno, tramite la selezione da un menù apposito;
- Ad ogni danno (o cura) subito (o inflitto) verrà visualizzato un ologramma condiviso che mostra i punti vita rimanenti del giocatore.

2.1.2 Requisiti non funzionali

- L'applicazione dovrà usare la tecnologia webXR per rendere fruibile, tramite un qualsiasi browser compatibile, l'esperienza di gioco.

2.2 Tecnologie

Elenco delle tecnologie utilizzate nell'elaborato.

Capitolo 3

Sviluppo

Design architetturale, più filosofico che implementativo + implementazione dettagliata.

3.1 Design Architetturale

Illustra l'architettura generale del software, includendo diagrammi e spiegazioni delle componenti principali e delle interazioni tra di esse.

3.2 Progettazione dettagliata

Se necessario, fornisce dettagli aggiuntivi sulle specifiche tecniche e progettuali, come diagrammi di sequenza, diagrammi delle classi, modelli di dati e così via. (Probabilmente da rimuovere)

3.3 Implementazione

Offre una panoramica dell'implementazione del software, inclusi gli strumenti utilizzati, la tecnologia impiegata e le scelte di sviluppo.

Conclusioni

Qui il testo delle conclusioni alla tesi. Non deve essere un riepilogo di quanto fatto nella tesi ma piuttosto le conclusioni raggiunte relative al lavoro svolto.

Ringraziamenti

Bibliografia

[1] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley Professional, Reading, MA, 1994.

[2] Robert Nystrom. *Game Programming Patterns*. Genever Benning, San Francisco, CA, 2014.