UNIVERSITÀ DEGLI STUDI DI CAGLIARI

FACOLTÀ DI SCIENZE

Corso di Laurea Triennale in Informatica

PAC-PAC: Debugging di giochi punta e clicca sviluppati da utenti non programmatori

Relatore
Prof. Lucio Davide Spano

Studenti
Marco Ortu
Matr. N. 65316

Andrea Re
Matr. N. 65421

ANNO ACCADEMICO 2018/2019

Titolo: PAC-PAC: Debugging di giochi punta e clicca sviluppati da utenti non programmatori

Relatore: Prof. Lucio Davide Spano

Candidati: Marco Ortu (60/61/65316), Andrea Re (60/61/65421)

Sommario: La nostra tesi si concentrerà in particolare sulla realizzazione della sezione di Debug della Web Application PAC-PAC, ovvero come è stata implementata l'interfaccia utente e la persistenza dei dati per quanto riguarda il salvataggio dello stato del gioco.

Il progetto è stato pensato con lo scopo di fornire agli utenti che non hanno conoscenze di programmazione strumenti e metodi semplificati per l'ideazione, progettazione e produzione di videogiochi d'avventura e di fiction interattive. In questo genere, chiamato "point-and-click", il giocatore controlla un personaggio e le sue azioni interagendo con un dispositivo di puntamento, tipicamente il mouse. Attraverso alcune modalità originali vengono integrate foto e riprese panoramiche a 360 gradi di ambienti reali con l'interattività tipica dei videogiochi point-and-click come la risoluzione di piccoli enigmi.

Nello specifico, abbiamo ideato un'interfaccia di Debug nell'editor per la creazione del gioco point-and-click, con l'obiettivo di mostrare all'utente creatore tutti gli stati del gioco che sta sviluppando, in modo da aiutarlo a prevenire e/o rimediare agli errori di definizione del gioco.

Indice

Capitolo 1

Introduzione

Il nostro contributo, dato per il progetto PAC-PAC, nasce dall'idea di dare la possibilità di creare un videogioco a tutte le persone che desiderano condividere con la comunità il loro prodotto. A causa delle poche conoscenze a livello di programmazione, la quale richiede dedizione e tempo, la maggior parte delle persone non riesce a concretizzare l'idea di creare un videogioco.

Questo progetto è orientato alla creazione di un videogioco "punta e clicca", che promuova l'ambiente e la cultura. Il giocatore potrà interagire con le "Scene", ovvero video e immagini a 360 gradi di ambienti realmente esistenti e spostarsi tra loro risolvendo degli enigmi presenti nel gioco. PAC-PAC fornisce un editor con cui l'utente possa creare il suo gioco in maniera semplice e intuitiva attraverso un interfaccia e l'uso del linguaggio naturale. L'editor permette di inserire scene e oggetti con comportamenti già definiti e dà la possibilità di formulare delle regole in linguaggio naturale per determinare le azioni che potrà eseguire il giocatore all'interno del gioco.

Nello specifico, ci siamo occupati della sezione di Debug dell'editor. In questa interfaccia dell'applicazione, l'utente potrà monitorare l'andamento del gioco da lui creato. Nella parte centrale verrà visualizzato il gioco in modalità play, dove l'utente potrà simulare il gioco e interagire con gli oggetti presenti nella scena. Se si interagisce con un oggetto, verranno evidenziate le sue regole attive nella parte inferiore della UI chiamata "Regole della scena", il nome dell'oggetto e le sue azioni nella sezione a destra dell'interfaccia chiamata "Rightbar". Questo permette all'utente di comprendere in maniera intuitiva il comportamento dell'oggetto considerato. Inoltre, nella rightbar, l'utente potrà modificare lo stato di un oggetto della scena corrente o di un'altra scena senza dover arrivare in quel punto del gioco. È stata anche implementata la possibilità di effetturare dei salvataggi dello

stato del gioco, in modo che questi possano essere ricaricati in qualsiasi momento, permettendo all'utente di continuare da quel punto preciso la fase di debugging. L'insieme di tutti i salvataggi viene visualizzato nella sezione omonima della left-bar, a sinistra della UI. Nei capitoli successivi argomenteremo nel dettaglio tutti i componenti dell'interfaccia citati qui sopra.

Nel capitolo 2 verrà spiegato lo stato dell'arte, trattando gli argomenti di interazione uomo-macchina, dei videogiochi punta e clicca, della realtà virtuale. Verranno inoltre introdotti lo scopo e gli obiettivi del progetto PAC-PAC. Il capitolo 3 tratterà gli strumenti e quindi le tecnologie utilizzate per sviluppare il nostro progetto. Nel capitolo 4 verrà descritto nel dettaglio il lavoro realizzato, trattando tutti i componenti dell'interfaccia di Debug e le funzionalità implementate. Infine, nel capitolo 5, saranno presenti le conclusioni e gli sviluppi futuri.

Capitolo 2

Stato dell'arte

2.1 Interazione uomo-macchina

> *"Lo Human-Computer Interaction è una disciplina che studia la progettazione, la valutazione e l'implementazione di sistemi computazionali interattivi destinati all'utilizzo umano ed i maggiori fenomeni attorno ad essi." [12]*

L'interazione uomo macchina è lo studio che si occupa della comunicazione tra persone e computer (o macchine). Questo studio non copre solamente aspetti di informatica, ma tratta anche la psicologia, le scienze cognitive, l'ergonomia, il design e numerose altre materie. Con l'aumentare dell'uso delle applicazioni informatiche, è richiesto un livello di progettazione volto a coprire gli innumerevoli contesti d'uso presenti.
I primi cenni di *HCI* si ebbero con la nascita degli schermi grafici e con la possibilità di interagire con essi. Uno dei pionieri di questa disciplina è sicuramente Ivan Sutherland, inventore di Sketchpad [4], il quale ha dato il via allo sviluppo di innumerevoli interfacce per migliorare l'interazione tra utente e computer.

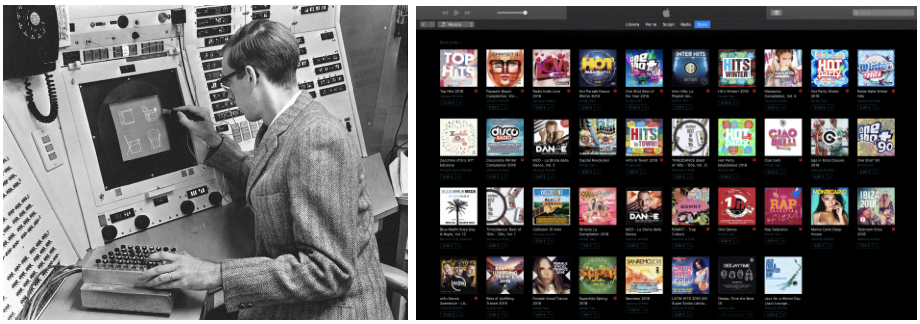

Figura 2.1: *Sketchpad e iTunes, progressione nelle interfacce.*

La disciplina dell'interazione uomo-macchina può essere suddivisa in due parti: interfaccia e interazione. La prima rappresenta il modello astratto tramite il qua-

le si interagisce con un dispositivo per compiere un determinato task, il secondo invece è il livello che si interpone tra macchina e utente, per consentire la comunicazione tra le due parti. Progettare l'interfaccia utente significa combinare un insieme di metafore di interazione, immagini e concetti usati, per veicolare funzioni e contenuti informativi sullo schermo, nel modo più semplice ed intuitivo possibile.

2.2 Videogiochi punta e clicca

L'avventura grafica è un genere di videogioco d'avventura, nato come evoluzione della avventure testuali e dotato di una interfaccia utente di tipo grafico. Comunemente, questa categoria rientra nel sottogenere Punta E Clicca (dall'inglese Point-And-Click) [5]: una classe di giochi in cui si controlla un personaggio in terza o in prima persona, utilizzando principalmente il mouse. L'espressione *punta e clicca* deriva dall'azione eseguita principalmente in questo tipo di gioco, ovvero puntare il cursore del mouse verso oggetti, personaggi o luoghi e cliccarci per interagire con essi. Tramite questo sistema di interazione il personaggio potrà combinare oggetti, parlare con altri personaggi, esaminare particolari dettagli di un ambiente e risolvere enigmi più o meno complessi.

Figura 2.2: *Interfaccia di uno dei giochi Punta e Clicca più famosi: Monkey Island.*

Si osservi comunque che esistono anche giochi Punta E Clicca che non hanno bisogno di mouse ma utilizzano strumenti come il joypad o, parlando di realtà virtuale, il visore con l'ausilio di strumenti per riconoscere i gesti.

2.2.1 Immersività del videogioco con realtà virtuale

La realtà virtuale è una simulazione grafica generata al computer che fornisce l'illusione di una partecipazione in un ambiente sintetico, dando la possibilità all'utente di interagire con oggetti e navigare in uno spazio tridimensionale. La rappresentazione può avere un livello di fedeltà superiore a quello che si può raggiungere con il multimedia: la realtà virtuale introduce un senso di presenza nel quale l'utente si sente completamente immerso nell'esperienza.
In questi ultimi anni, grazie alle nuove tecnologie, sia hardware che software, lo sviluppo della realtà virtuale ha fatto grandi passi avanti, soprattutto nel settore videoludico, ma non solo: questo tipo di interazione sta trasformando in chiave ludica anche la formazione e la cultura, dando all'utente la possibilità di vivere un esperienza video a 360 gradi, trasformando il classico storytelling lineare in un gioco virtuale.

Figura 2.3: *Test di una nuova realtà per museo con il supporto del VR.*

2.3 Il progetto PAC-PAC

Negli ultimi anni si è iniziato a sfruttare la realtà virtuale per la creazione di esperienze immersive a 360 gradi. In qualsiasi luogo, qualsiasi persona ha la possibilità di promuovere il proprio prodotto in modo immersivo, mediante l'uso di strumenti come visori e sensori di movimento. La realtà che si è creata non ha mai dato la possibilità, ad utenti inesperti nell'ambito della programmazione, di poter creare videogiochi d'avventura interattivi, ambientati in luoghi reali. Questa esigenza è stata catturata da PAC-PAC [6], un progetto iniziato nel febbraio del 2018 che ha come principale obiettivo quello di dotare imprese ed enti che promuovono il patrimonio ambientale e culturale di strumenti per la realizzazione di videogiochi d'avventura ambientati in luoghi reali. Il progetto, finanziato da Sardegna Ricerche, si concentra sulla formula di videogiochi che prende il nome di *point-and-click*. L'idea proposta dagli sviluppatori del progetto sta nel creare riprese fotografiche e video 3D, per poi integrarle - mediante un editor user-friendly realizzato per utenti non programmatori - con funzionalità interattive, dando la possibilità di creare un'esperienza immersiva composta da storie ed enigmi creati su ambienti reali. L'originalità di questo progetto sta nel fatto che l'ambiante sintetico in cui si immergerà il giocatore non sarà un modello 3D creato digitalmente, ma sarà realizzato con immagini e riprese di luoghi reali, trasformando un classico videogioco di avventura in un *first-person interactive cinema*, cinema interattivo in prima persona. La realizzazione di questo progetto porterebbe i seguenti vantaggi:

1. valorizzare ambienti e luoghi ai fini culturali e turistici;
2. utilizzare PAC-PAC non solo come videogioco ma anche come tool per esplorare delle zone caratterisitiche prima di andare a visitarle;
3. anche se la realizzazione del gioco mediante l'editor permette di programmare un gioco senza conoscere la programmazione, questo progetto promuove l'ingegno e l'arte, perchè l'utente dovrà comporre le scene, occuparsi delle composizioni musicali e la realizzazione dei video;
4. riduce i costi di produzione e il fabbisogno di competenze specifiche relative alla modellazione 3D e successivamente alla loro programmazione.

Capitolo 3

Strumenti utilizzati

Il progetto di tesi è stato scritto interamente nell'IDE WebStorm [7]. Per la realizzazione dell'interfaccia è stato usata la libreria React, affiancata all'architettura Flux. Associato allo stile definito in locale, è stata utilizzata la libreria Bootstrap per alcuni template. Infine, per quanto riguarda la persistenza dei dati, sono stati utilizzati Db a grafo forniti dal sw Neo4j.

Figura 3.1: *Tecnologie usate nell'applicazione.*

3.1 React

React [8] (noto come react.js) è una libreria JavaScript per la costruzione di interfacce utente. È gestito da Facebook e da una comunità di singoli sviluppatori e aziende. Questa libreria consente la realizzazione di interfacce per ogni stato dell'applicazione, pertanto è affiancata all'architettura di Flux. Ad ogni cambio di stato React aggiornerà efficientemente solamente le parti della UI (User Interface) che dipendono da tali dati.
La natura dichiarativa di React rende il codice più prevedibile e facile da debuggare. Un'altra caratteristica è la possibilità di creare componenti isolati e comporli per creare UI complesse. Sia l'interazione che la logica dei componenti sono

implementate in JavaScript e questo permette di passare e accedere facilmente a strutture dati complesse in vari punti dell'applicazione, senza dover salvare informazioni sul DOM (Document Object Model). Un esempio di applicazione web che utilizza React è Netflix.

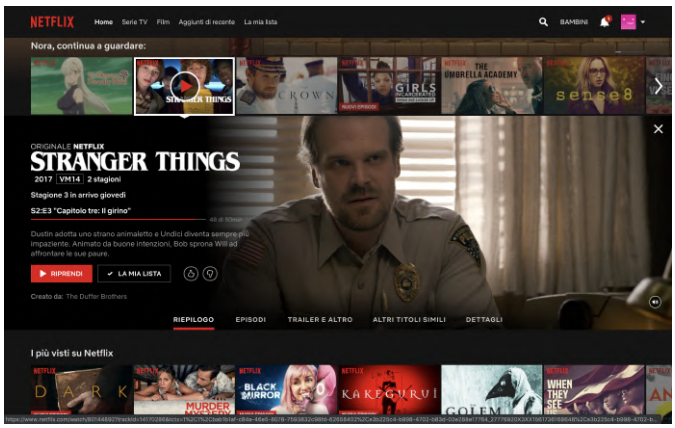

Figura 3.2: *Schermata di Netflix.*

3.2 Flux

Flux [9] è un'architettura usata da Facebook per la creazione di web applications client-side. Questa tecnologia è affiancata a React, completandone i componenti di visualizzazione mediante l'uso di un flusso dati unidirezionale. Le applicazioni basate su architettura Flux, possono essere suddivise in tre componenti principali:

- **Dispatcher**: è il fulcro dell'architettura e gestisce tutto il flusso di dati in un'applicazione Flux. Esso è il meccanismo che permette la redistribuzione di azioni ad altri componenti, detti Store. Le azioni sono dei metodi trasmessi dal dispatcher agli Store, che contengono tutti i dati relativi all'applicazione. Ogni Store si registra e fornisce una callback. Quando un generatore di azioni fornisce al dispatcher una nuova azione, invia a tutti gli Store l'azione, che ricevono tramite i callback nel registro;
- **Stores**: Sono le unità che memorizzano la logica e lo stato dell'applicazione. Come già spiegato, uno Store si registra con il Dispatcher e fornisce una callback che riceve l'azione come parametro. All'interno di un singolo Store, tutte le singole azioni genereranno una modifica dello stato dell'applicazione. Dopo aver effettuato l'aggiornamento, gli store notificano le views (React) del cambiamento di stato: a questo punto le views potranno interrogare il nuovo stato ed aggiornarsi;

- **Views**: È il componente che funge da tramite: esso si occupa di prendere i dati dagli Stores e passarli a chi li richiede. Le views stanno in ascolto di eventi che vengono emessi dagli Store da cui dipendono e si aggiornano in base ad essi. Infine, sempre tramite le views, è possibile richiamare un'azione.

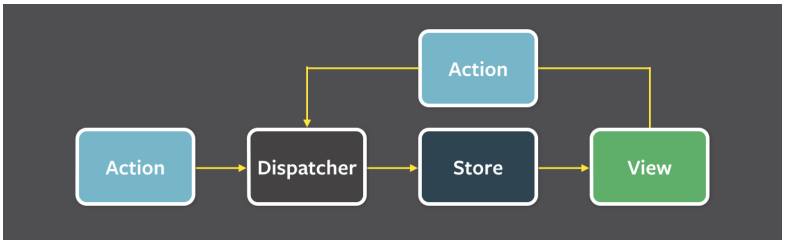

Figura 3.3: *Data flow di Flux.*

3.3 A-Frame

A-Frame [10] è un web framework per la creazione di esperienze virtuali (VR). Originariamente è stato concepito da Mozilla come progetto open source indipendente. Ad oggi, è mantenuto dai co-creatori in Supermedium. A-Frame è un linguaggio di markup e permette di costruire scene 3D, ma il nucleo è un potente framework entità-componente che fornisce una struttura dichiarativa, estensibile e componibile alla libreria Three.js [11] utilizzata per creare e mostrare ambienti e oggetti 3D. Le caratteristiche principali di A-Frame sono:

- **Semplicità della realtà virtuale**: attraverso i tag *<script>* e *<a-scene>*, A-Frame gestirà il boilerplate 3D, l'impostazione VR e i controlli predefiniti;
- **HTML dichiartivo**: essendo basato su HTML, A-Frame è accessibile a tutti, a partire dai sviluppatori web, designer fino ai bambini;
- **Architettura Entità-Componente**: A-Frame è un framework potente che fornisce una struttura dichiarativa, componibile e riutilizzabile;
- **Cross-Platform VR**: A-Frame permette di creare applicazioni VR per diverse piattaforme come Vive e Rift;
- **Prestazioni**: A-frame è ottimizzato per WebVR. Mentre usa il DOM, i suoi elementi non toccano il motore di layout del browser. Gli aggiornamenti degli oggetti 3D vengono effettuati in memoria senza sovraccarichi grazie ad un efficiente garbage collector;
- **Visual Inspector**: A-Frame fornisce un pratico inspector 3D integrato;

- **Componenti**: A-frame dispone di una vasta scelta di componenti che possono essere utilizzati: geometrie, materiali, luci, animazioni, modelli, raycast, ombre ecc.

Figura 3.4: *Esempio di realtà virtuale creata con A-Frame.*

3.4 Bootstrap

Bootstrap [2] nasce nel 2011 come progetto interno di Twitter. Successivamente diventa indipendente e utilizzabile dagli sviluppatori di tutto il mondo come base per la realizzazione di interfacce web. Bootstrap è una raccolta di strumenti grafici, stilistici e di impaginazione che permettono di avere a disposizione una grande quantità di funzionalità e stili modificabili e adattabili a seconda delle proprie esigenze. Bootstrap è utilizzato principalmente per il responsive web design, ovvero rendere il design delle pagine web in grado di adattarsi dinamicamente a seconda della grandezza e delle caratteristiche del dispositivo utilizzato.

3.5 Neo4j

Neo4j [1] è un software per basi di dati a grafo open source sviluppato interamente in Java. È un database totalmente transazionale, che viene integrato nelle applicazioni permettendone il funzionamento stand alone. Neo4j è definito come nativo, perché implementa il property graph model sino al livello di archiviazione, memorizzando tutti i dati in una cartella.
I punti di forza su cui Neo4j punta sono:

1. **Tempo di attraversamento costante**, indipendentemente dalle dimensioni del grafo, garantendo un'alta scalabilità, sino a miliardi di nodi memorizzati su un hardware di modeste prestazioni;
2. **Flessibilità** in quanto è possibile aggiungere e modificare relazioni e nodi in qualsiasi momento, in modo semplice, anche attraverso interfaccia grafica.

Lo scopo principale di questa tipologia di database è la semplicità nel trattare sia i nodi che le relazioni allo stesso modo. Neo4j, per la definizione delle query, utilizza Cypher [3], un linguaggio dichiarativo aperto, progettato per i database a grafo. Creato dagli stessi sviluppatori di Neo4j, è reso disponibile a tutti tramite il progetto openCypher per fornire uno standard per l'interrogazione e la definizione di database a grafo.

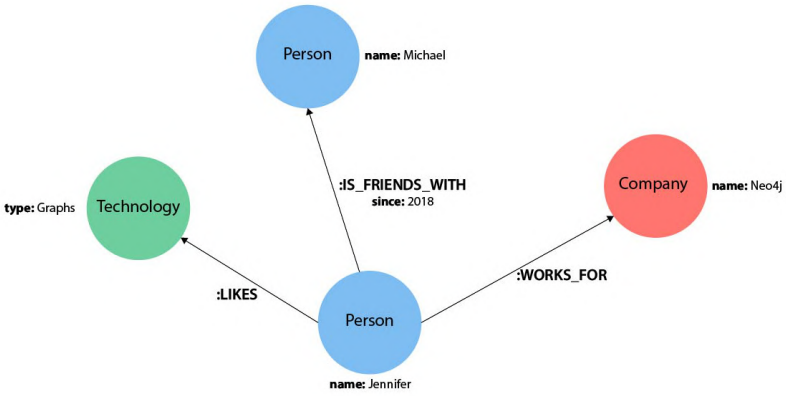

Figura 3.5: *Esempio di Db a grafo creato con Neo4j.*

Capitolo 4

PAC-PAC

Il nostro lavoro è stato quello di realizzare un'interfaccia che permettesse all'utente non programmatore - ma designer del gioco - la possibilità di testare e modificare l'insieme delle proprietà degli oggetti presenti nelle scene del gioco e vedere come queste modifiche abbiano impatto sul flusso del gioco stesso. In secondo luogo, abbiamo curato la parte di salvataggio in modo persistente dello stato del gioco: questo permette all'utente di ricaricare diversi salvataggi generati in precedenza, senza ricreare lo stato del gioco desiderato ad ogni avvio del test.

4.1 Background

La struttura messa a disposizione dell'utente per la creazione (detto anche *authoring*) di un gioco, già presente prima del lavoro svolto in questa tesi, può essere sintetizzata brevemente nel seguente modo: un gioco è composto da una a più scene, che nel game engine chiameremo come *bolle*.
All'interno di ogni bolla possono essere inseriti degli oggetti con i quali interagire. Ogni oggetto è creato sulle primitive messe a disposizione da A-Frame ed è caratterizzato da:

- un identificatore (o riferimento) ed un nome;
- una tipologia di oggetto (transizione, chiave, interruttore, lucchetto);
- la visibilità dell'oggetto stesso (negli enigmi potrebbe essere necessario inserire degli oggetti invisibili);
- uno stato (ad esempio *raccolto* per una chiave o *off* per un interruttore);
- un insieme di file multimediali, quali media (immagini o video sovrapposte alla scena) e audio direzionali (suono di un oggetto);
- una geometria, cioé un insieme di vertici che stabiliscono la regione cliccabile dell'oggetto.

Tutti gli oggetti presenti nel gioco sono memorizzati in ObjectStore, uno Store di Flux. Ciascun oggetto, comprese le transizioni, è posizionato all'interno di una scena in base alla sua posizione (anch'essa definita dal creatore del gioco nella sezione di editing della geometria): l'interazione tra il giocatore ed un oggetto passa per la geometria di quest'ultimo.

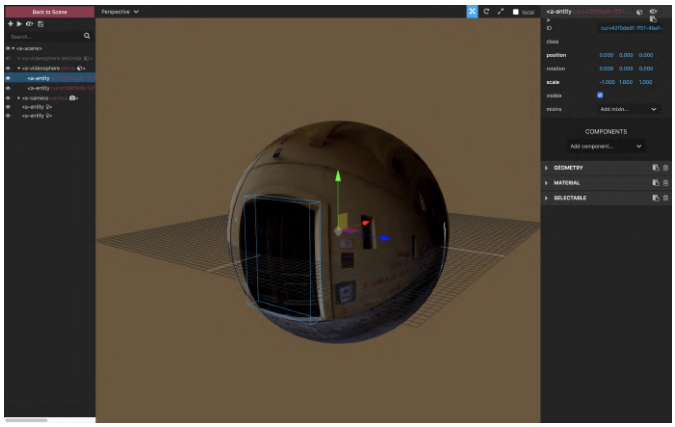

Figura 4.1: *Una bolla contenente la geometria di un oggetto.*

Le bolle del gioco sono collegate tra di loro mediante dei collegamenti, che prendono il nome di *transizioni*. L'insieme di bolle e transizioni generano una struttura a grafo - che chiameremo *gameGraph* - all'interno del quale il giocatore si potrà muovere. Per motivi di efficienza, quando un giocatore avvia il gioco, non vengono caricate tutte le bolle ma solamente quelle limitrofe alla scena corrente. Ad esempio, prendendo in esame la *figura 4.2*, se il giocatore si trova nella bolla con nome Scena1, le bolle caricate saranno solamente Scena1 e Scena2, poiché Scena3 non è limitrofa.

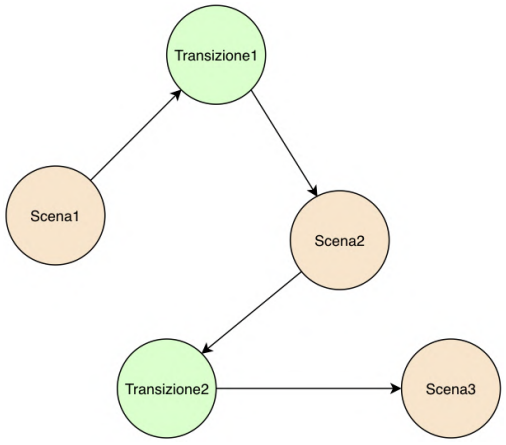

Figura 4.2: *Grafo che spiega la struttura di un gioco.*

L'insieme dei riferimenti agli oggetti interattivi (interruttori, chiavi, lucchetti) e alle scene formano lo stato del gioco, il *runState*: una mappa <*K*,*V*> che associa ad ogni riferimento (*K*) uno stato (*V*). Questo concetto è fondamentale per il tema della tesi e verrà trattato nuovamente nel paragrafo che discute la modifica e il salvataggio dello stato del gioco.
Il flusso del gioco definito dall'utente è stabilito da delle regole generate nella fase di editing: il designer del gioco potrà creare dei veri e propri enigmi da far risolvere al giocatore. Ciascuna regola è suddivisa in tre parti:

1. un evento che definisce quando una regola deve essere eseguita;
2. una o più condizioni (opzionali);
3. una lista di azioni, eseguite solamente se tutte le condizioni precedenti sono rispettate.

Tutte le valutazioni vengono fatte sullo stato del gioco, ovvero sul *runState*.

4.2 Interazione

L'interfaccia è suddivisa in 5 diversi componenti: Topbar, Scena centrale, Rightbar, Leftbar e Sezione delle regole.

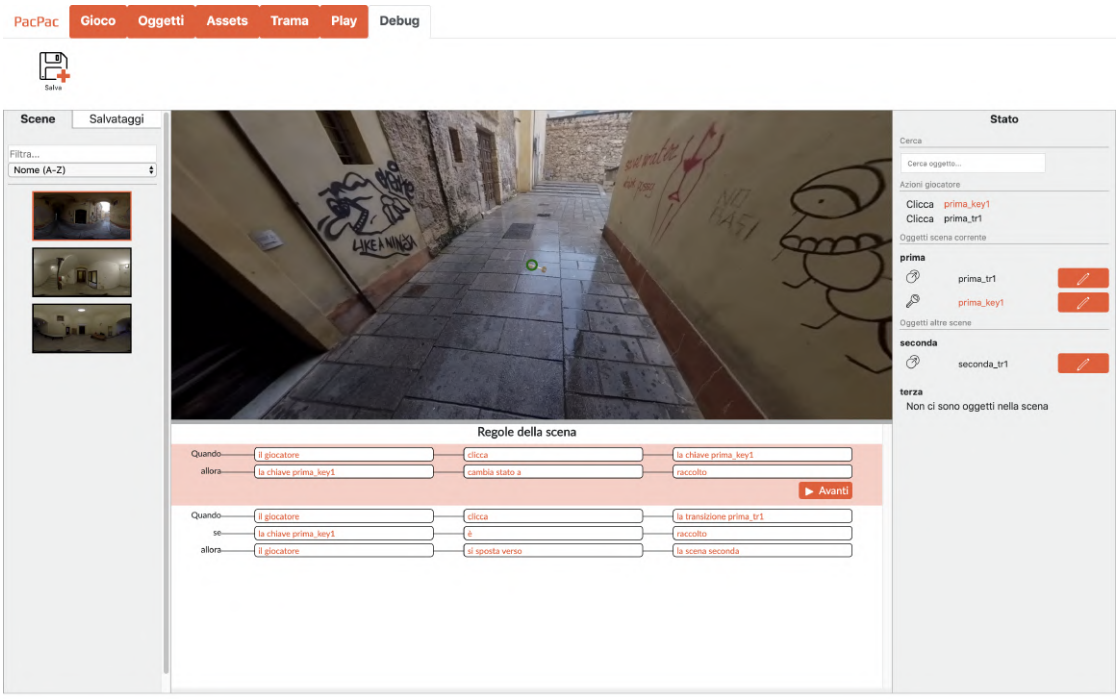

Figura 4.3: *Interfaccia in modalità di Test/Debug.*

4.2.1 TopBar

Nella Topbar l'utente potrà visualizzare le varie sezioni dell'editor premendo i rispettivi pulsanti:

1. **Gioco**: è la sezione che si occupa dell'authoring del gioco, in questa tab l'utente definisce la struttura della storia;

2. **Oggetti**: consente l'inserimento di nuovi oggetti interattivi all'interno di una scena;

3. **Assets**: è la sezione che si occupa di elencare i file multimediali di un gioco;

4. **Trama**: si occupa di generare in modo automatico una struttura di gioco prova, successivamente personalizzabile dall'utente;

5. **Play**: sezione dove viene lanciato il gioco;

6. **Debug**: è la tab che si occupa del debugging del gioco.

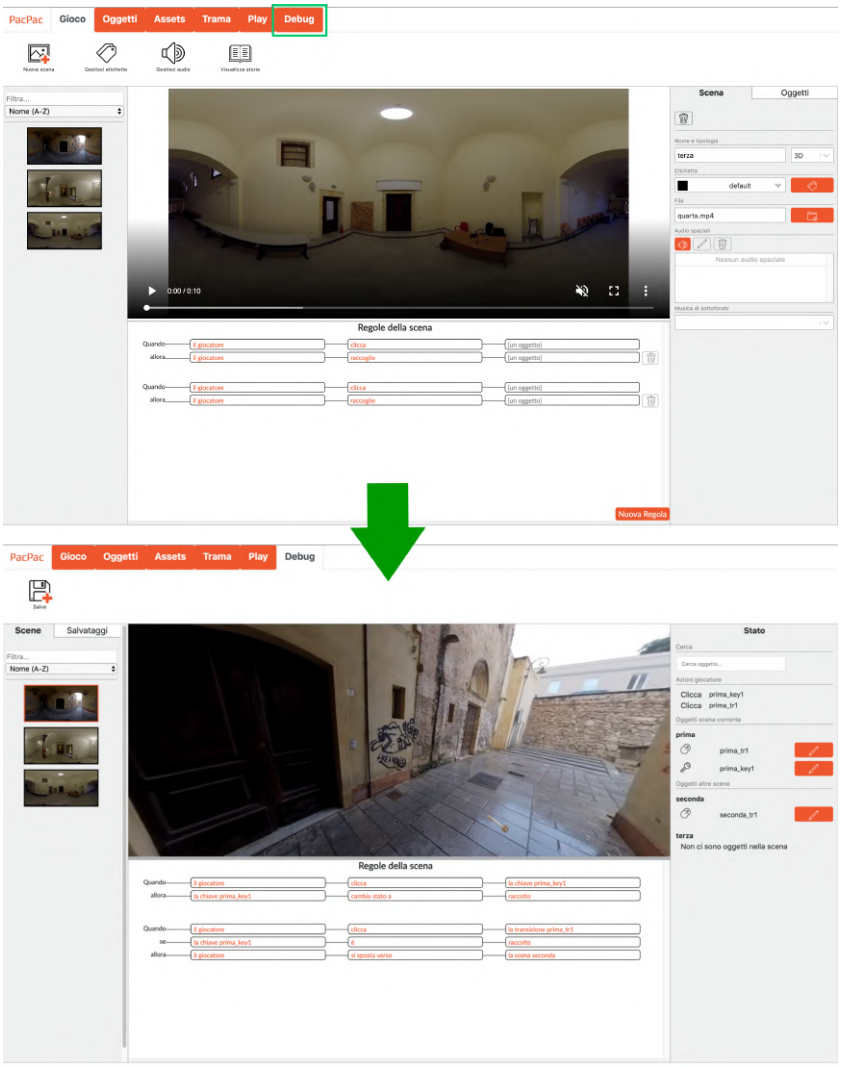

Figura 4.4: *Topbar - cambiare tab nell'editor.*

Nella parte inferiore del componente Topbar - nella sezione di Debug dell'editor - potrà essere creato un nuovo salvataggio premendo sulla apposita icona *Salva* (*vedi figura 4.5*) che farà comparire un pop up che invita l'utente a inserire il nome del salvataggio e confermare o annullare premendo gli appositi pulsanti.

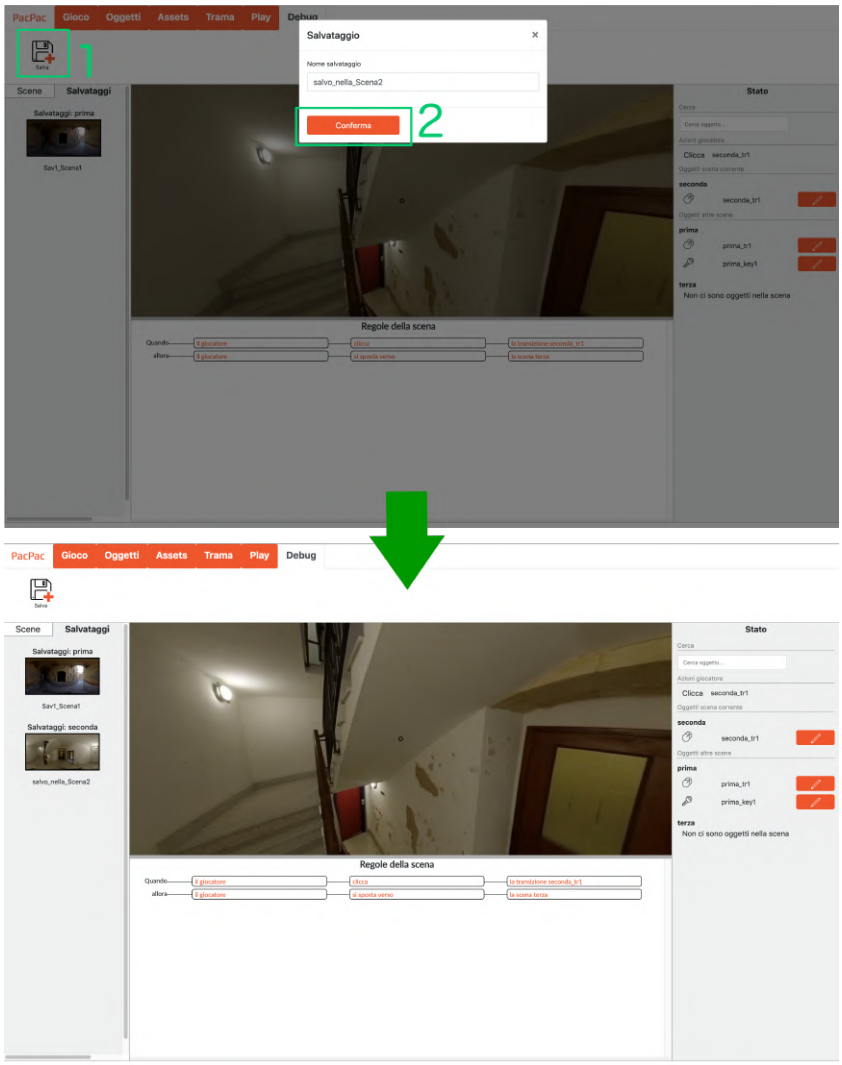

Figura 4.5: *Topbar - effettuare un nuovo salvataggio.*

4.2.2 Scena centrale

Nella sezione centrale troviamo la scena in realtà virtuale in cui il giocatore si ritroverà all'avvio del gioco. Pur non indossando alcun visore, l'utente che sta testando il gioco può comodamente muoversi all'interno della scena con il mouse e procedere nella storyline definita nella sezione di editing. Inoltre, se l'utente clicca su un oggetto verrà evidenziata la sua regola e se le condizioni sono rispettate cambierà il suo stato. Per riconoscere che l'utente sta puntando con il mouse un oggetto, il cursore si ingrandirà e prenderà una colorazione verde.

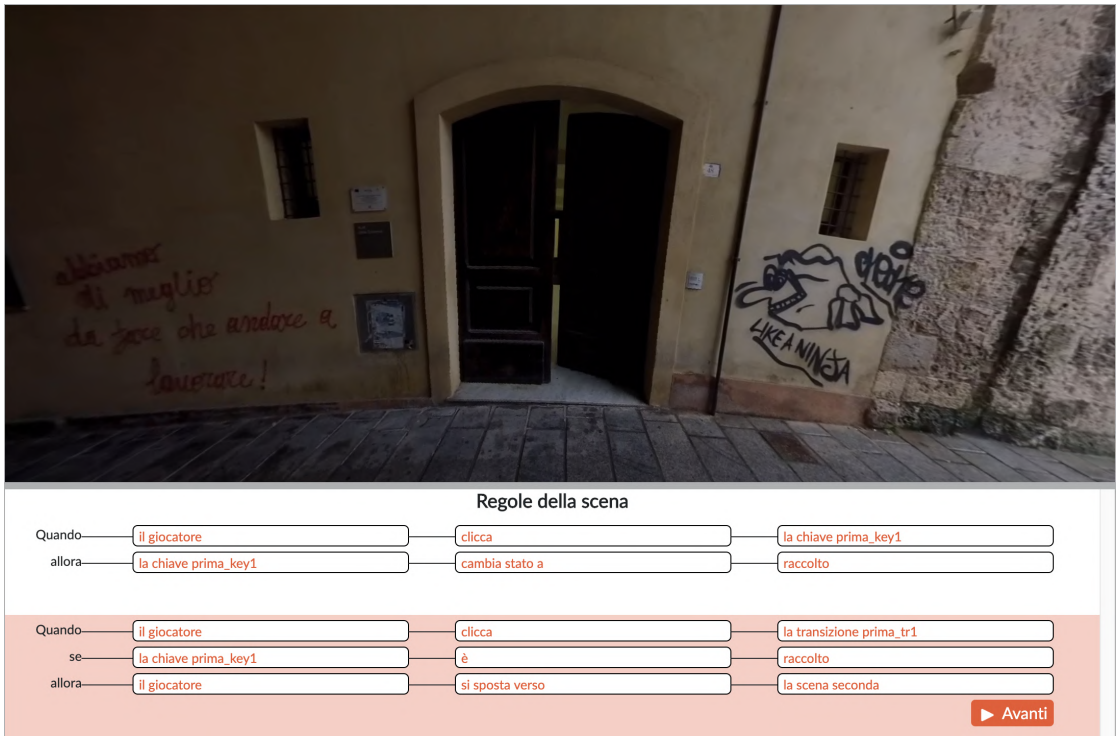

Figura 4.6: *Scena centrale - evidenziazione regola al click su un oggetto.*

4.2.3 RightBar

La Rightbar si occupa della gestione degli oggetti presenti nelle scene. Possiamo suddividere questo componente in due parti: la prima si occupa di elencare tutti gli oggetti della scena, la seconda si occupa di mostrare e modificare lo stato di un singolo oggetto. Se l'utente clicca su un elemento della rightbar verrà evidenziato il nome dell'oggetto, la sua regola nella sezione delle regole e le azioni che il giocatore può compiere su quell'oggetto, in modo da agevolare visivamente tutte le proprietà dell'elemento cliccato.

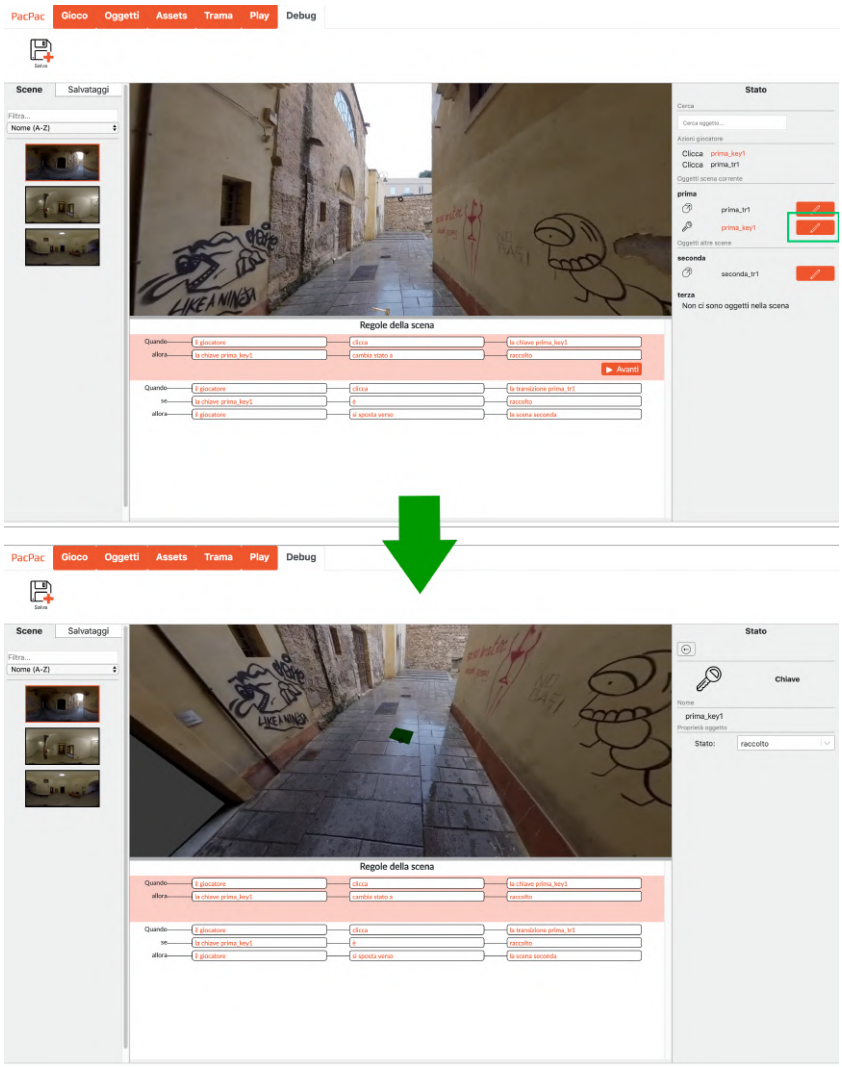

Figura 4.7: *Rigthbar - evidenziazione geometria e visualizzazione proprietà di un oggetto.*

Lista degli oggetti

In questa parte, possiamo suddividere la Rightbar in quattro sezioni distinte:

1. Una sezione dove poter ricercare gli oggetti presenti all'interno del gioco, filtrando l'insieme per nome;

2. Un elenco delle azioni compiute dal giocatore sugli oggetti che vengono coinvolti nella valutazione delle regole;

3. L'insieme degli oggetti raggruppati per scena corrente: per ciascun oggetto troviamo un icona che ne rappresenta il tipo (transizione, chiave, lucchetto o interruttore), il suo nome e un tasto che ci consente di vedere e/o modificare le proprietà dell'oggetto stesso;

4. L'insieme degli oggetti raggruppati per altre scene: verrà visualizzato la lista degli oggetti separati per scene, visualizzandoli in modo analogo alla sezione precedente.

Stato del singolo oggetto

In questa parte della Rightbar vengono mostrate le proprietà dell'oggetto: esso è rappresentato da un nome, un tipo e uno stato.
Se l'utente clicca sul pulsante (*vedi figura 4.7*) per la modifica di un oggetto posto affianco del suo nome:

1. la Rightbar si aggiornerà, mostrando le proprietà dell'oggetto;

2. la scena centrale ruoterà la visuale in direzione della geometria dell'oggetto;

3. la regola in cui è presente l'oggetto selezionato viene evidenziata.

4.2.4 LeftBar

La Leftbar si divide in due sezioni: la prima si occupa di listare le scene create nel gioco mentre la seconda mostra la lista dei salvataggi effettuati.

Scene

Nella sezione *Scene* la Leftbar mostrerà all'utente tutte le scene del gioco tramite delle miniature, che permettono all'utente di riconoscere immediatamente la scena guardando l'immagine. L'utente potrà capire in maniera veloce e intuitiva la scena corrente del gioco grazie al bordo intorno alla miniatura. Inoltre, le scene potranno essere filtrate per nome nell'apposita barra di ricerca e catalogate in ordine alfabetico, dalla prima all'ultima o viceversa.

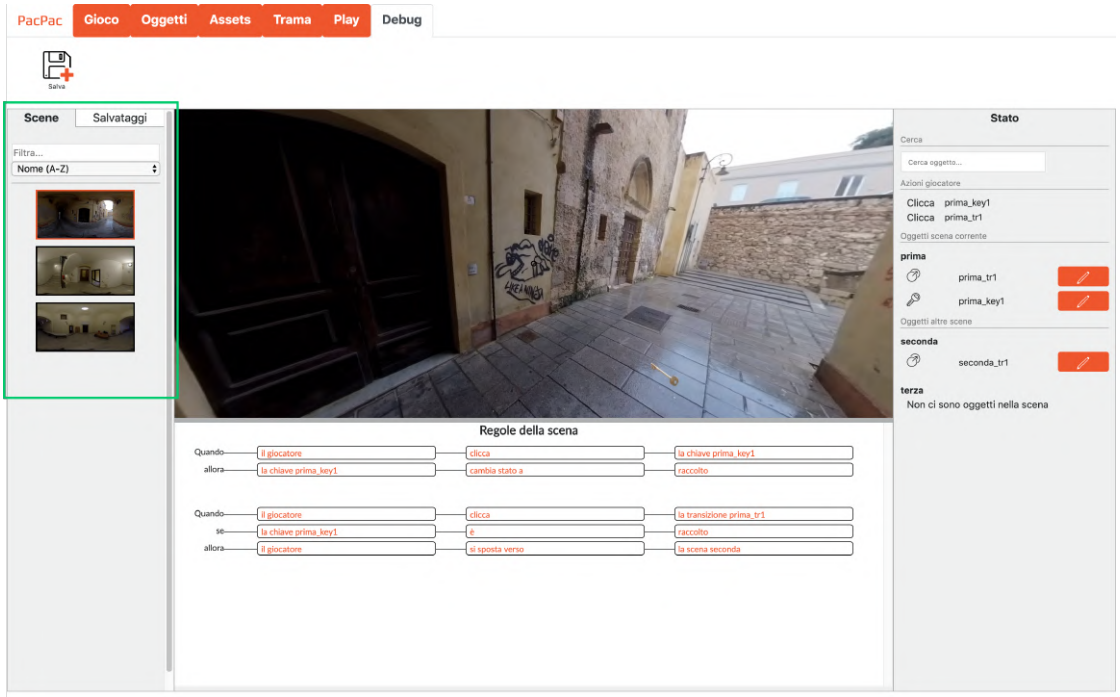

Figura 4.8: *Leftbar - lista delle scene presenti nel gioco.*

Salvataggi

Nella sezione *Salvataggi*, la Leftbar mostrerà i salvataggi divisi per scena effettuati dall'utente. Per ogni scena verrà visualizzato il suo nome, la sua miniatura seguita dai relativi salvataggi. Se l'utente clicca sul nome di un salvataggio questo cambierà colore e comparirà accanto il pulsante per caricare il salvataggio. Quando si carica un salvataggio comparirà un *alert* che chiederà un ulteriore conferma: in caso positivo la rightbar, scena centrale e la sezione delle regole verranno aggiornati.

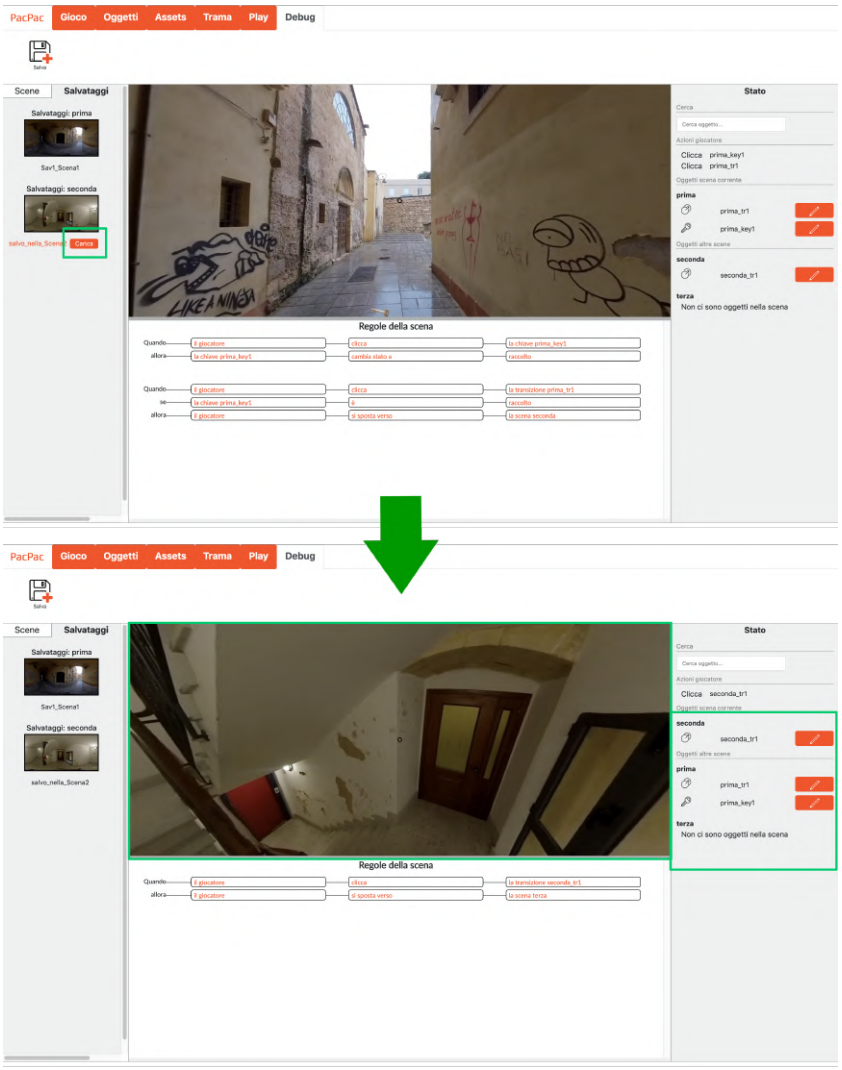

Figura 4.9: *Leftbar - caricamento di un salvataggio.*

4.2.5 Sezione delle regole

Nel componente delle regole troviamo l'elenco di tutte le regole, ciascuna creata in fase di design del gioco. Ogni regola definisce il comportamento di un oggetto e l'insieme di tutte regole determina in flusso del gioco. Quando una regola è selezionata, viene visualizzato un pulsante *Avanti* per poter eseguire la regola.

Figura 4.10: *Regole - elenco delle regole presenti in una scena.*

4.3 Implementazione

Questo paragrafo discute delle funzionalità che abbiamo curato nel nostro progetto di tesi. Ci siamo occupati principalmente di implementare le funzionalità necessarie per gestire l'interfaccia utente e quelle per la gestione del database in caso di salvataggio o caricamento di uno stato del gioco.

4.3.1 Funzionalità dell'interfaccia

TopBar

Il componente React TopBar già implementato prima dell'inizio del nostro lavoro, è integrato con la sezione che si occupa del debugging del gioco. Le funzioni aggiunte a questo componente si occupano di:

- **Passaggio all'interfaccia di Debug**: come visto alla *figura 4.4*, premendo il pulsante *Debug* la funzione controlla dallo store che gestisce le parti dell'editor se l'utente non è già nella sezione Debug: in caso positivo carica nella scena centrale la prima scena del gioco e aggiorna la pagina visualizzando la modalità Debug;
- **Effettuare un salvataggio**: come precedentemente spiegato alla *figura 4.5*, premendo l'icona *Salva* la topbar interagisce con il componente React InputSaveForm che genera un pop up per inserire il nome del nuovo salvataggio e utilizza una funzione che controlla se il campo inserito dall'utente sia corretto e carica il salvataggio nel database servendosi dell'API di Debug (il salvataggio lo vedremo nello specifico nel paragrafo delle funzionalità della persistenza dei dati).Dopo l'inserimento la leftbar verrà aggiornata inserendo il nuovo salvataggio.

DebugVRScene

DebugVRScene è un componente React che abbiamo implementato per la gestione della scena centrale della UI di Debug (*vedi figura 4.6*). Questo componente genera lo stato del gioco e gestisce la modalità play, quindi si occupa di generare gli assets e le bolle delle varie scene che verranno visualizzati con gli appositi tag della libreria A-Frame.

DebugTab

DebugTab è un componente React che abbiamo implementato per la gestione della sezione a destra della UI di Debug, Il componente si divide in due funzioni pricipali:

- **Lista degli oggetti delle scene**: Questa funzione genera una barra di ricerca per filtrare gli oggetti , le azioni che il giocatore può compiere con gli oggetti della scena corrente e la lista di tutti gli oggetti. Nella sezione di ricerca ci serviamo di uno store dell'architettura Flux per filtrare gli oggetti contemporaneamente all'input che inserisce l'utente nella barra di ricerca. Il resto della rightbar viene gestita da tre funzionalità:

  - **Lista delle azioni del giocatore**: si occupa di prendere le azioni che il giocatore può compiere in base agli oggetti presenti nella scena corrente dallo store delle regole e visualizzarli. Inoltre, se l'utente interagisce con un'azione, questa verrà evidenziata e contemporaneamente anche l'oggetto e la regola in questione;
  - **Lista degli oggetti della scena corrente**: ha il compito di prelevare dallo store degli oggetti e visualizzare solo quelli appartenenti alla scena corrente. Quando l'utente clicca sul nome dell'oggetto questo verrà evidenziato nelle regole in cui compare e la DebugVRScene sposterà la visuale della camera in direzione dell'oggetto stesso;
  - **Lista degli oggetti presenti nelle altre scene**: utilizza la funzione precedente ma filtra gli oggetti della scena corrente e visualizza per ogni altra scena i rispettivi oggetti.

- **Modifica dello stato di un oggetto**: viene richiamata quando l'utente desidera visualizzare o modificare lo stato di un oggetto (*interazione spiegata alla figura 4.7*). Questa funzione interagisce con il componente che gestisce la scena centrale, grazie a una funzione apposita la camera viene ruotata verso l'oggetto desiderato. Anche la rightbar viene aggiornata mostrando le proprietà dell'oggetto, che potranno essere modificate. C'è stata la necessità di salvare nello store una variabile che tiene conto della scena precedente, perchè l'utente può modificare anche un oggetto di un'altra scena e, visulizzando quest'ultima nella scena centrale, si sarebbe perso il riferimento della scena effettiva ove l'utente stava lavorando.

Leftbar

Nonostante il componente React Leftbar fosse già implementato, c'è stata la necessità di aggiungere delle funzionalità per gestire la sezione di Debug. Se l'editor è in modalità Debug verranno visualizzate due tab, *Scene* e *Salvataggi*: il passaggio dall'una all'altra è garantito dalla funzione che gestisce lo stato di *leftbarSelection*, contenuto nello store dell'editor. Se *leftbarSelection* ha come valore *'scene'*, la Leftbar conterrà (come in *figura 4.8*) la lista di tutte le scene del gioco: per fare ciò

sono state riutilizzate delle funzioni già presenti nel progetto. Nel caso contrario, cioé quello in cui *leftbarSelection* ha valore *'saves'* la Leftbar comunicherà con il componente *SavesOption*, che si occupa di elencare tutti i salvataggi.

SavesOptions

SavesOptions è un componente React da noi implementato per la gestione dei salvataggi della Leftbar. È stata creata una funzione che, dallo store dell'editor, nello specifico da *debugSaves* (uno stato contenente l'insieme dei salvataggi creati dell'utente) elenca tutti i nomi di tutti i salvataggi e li visualizza raggruppati per scena. Se l'utente preme sul nome di un salvataggio comparirà un pulsante che ne permette il caricamento (*vedi figura 4.9*) da database, servendosi dell'API di Debug (il caricamento verrà discusso nuovamente alla *sezione apposita 4.3.2*). Dopo l'esecuzione di un caricamento vengono aggiornati tutti i componenti della modalità Debug, la scena corrente viene eventualmente aggiornata e lo stato di tutti gli oggetti viene ricostruito in base a quello del salvataggio appena caricato.

4.3.2 Persistenza dei salvataggi su database

Salvataggio dello stato del gioco

Quando un utente decide di salvare uno stato di gioco (vedi *figura 4.5*), le API di debug interrogano il database chiedendo l'inserimento di un nuovo salvataggio. Per ciascuno di questi è richiesto un nome univoco e un riferimento alla scena corrente, cioé quella in cui è stato effettuato il salvataggio. Dal punto di vista di Neo4j, la creazione di un salvataggio genera un nodo (con tipo *DebugState*) che ha una struttura pari a quella mostrata in *figura 4.11*

Figura 4.11: *Struttura del nodo di un salvataggio creato su Neo4j.*

Da questo nodo partono un numero relazioni (con etichetta *STORES_OBJECT*) pari a quello di tutti gli oggetti presenti nel gioco. Ogni nodo (con tipo *DebugObject* e struttura simile a quella mostrata in *figura 4.12*), contiene:

1. un riferimento all'oggetto memorizzato nello store degli oggetti (*uuid*);
2. un riferimento al salvataggio appena creato (*saveName*);
3. l'insieme delle sue proprietà, quali stato (*state*) e visibilità (*visibile*).

Figura 4.12: *Struttura del nodo di un oggetto creato su Neo4j dopo il salvataggio.*

Dato che, come detto nella sezione di Background (*vedi 4.1*), il *runState* contiene riferimenti ad oggetti e scene, si è deciso di salvare solo i riferimenti dei primi dei

due, poiché l'unica scena di cui dobbiamo tenere conto è quella corrente: possiamo ricavarci questa informazione accedendo alle *props* dell'editor. Tutte le proprietà degli oggetti (scena alla quale sono collegati, geometria ecc) sono memorizzati nell'*ObjectStore* e possono essere recuperate mediante lo uuid presente nel nodo salvato. Dopo aver effettuato un salvataggio, il grafo generato avrà una struttura non dissimile da quella mostrata in *figura 4.13*

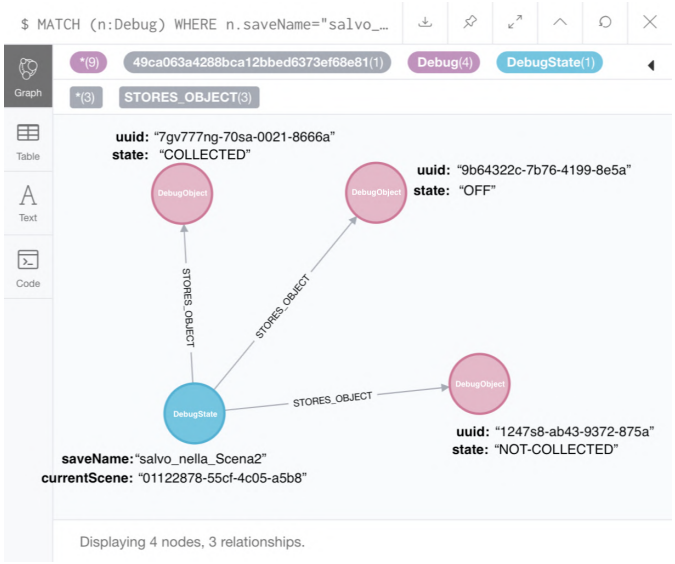

Figura 4.13: *Struttura del salvataggio di uno stato di gioco su Neo4j.*

Tutti i salvataggi sono memorizzati all'interno di uno stato (chiamato *debugSaves*) dello store dell'editor. Ad ogni nuovo inserimento, verrà richiamata una *Action* che andrà ad aggiungere al suddetto store il nuovo salvataggio. Internamente, *debugSaves* è una mappa <*K*,*V*> che ad ogni identificatore di una scena (*K*) associa un set (*V*) contenente tutti i nomi dei salvataggi effettuati nella scena.

Caricamento dello stato del gioco

Come già spiegato nella sezione Salvataggi della Leftbar (*vedi 4.2.4*), un utente può rigenerare lo stato di un gioco precedentemente salvato. Dato che ciascun salvataggio è identificato da un nome univoco, le API di Debug interrogano il database richiedendo semplicemente di restituire il grafo che abbia come *saveName* il nome del salvataggio.

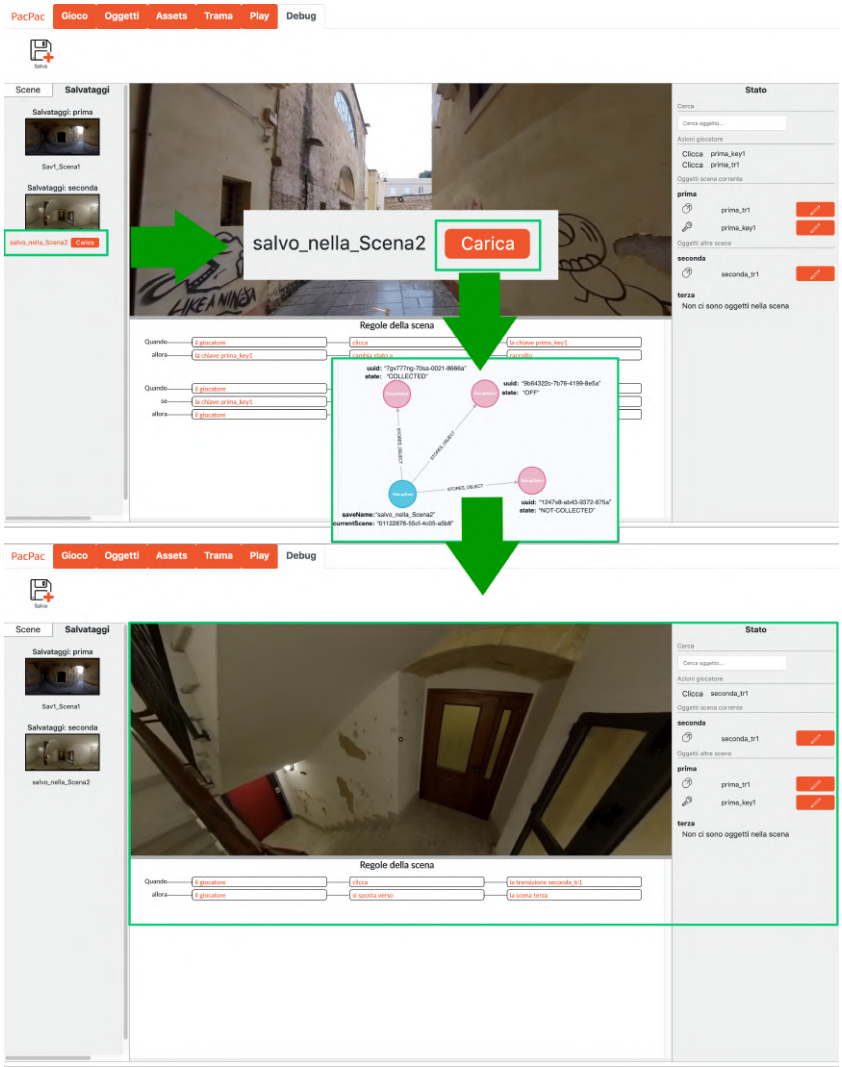

Figura 4.14: *Caricamento di un salvataggio.*

Quando il database risponde restituendo un nuovo stato del gioco, quello attuale viene sovrascritto:

- per ogni oggetto viene modificato lo stato;

- la Scena centrale viene aggiornata mostrando la bolla collegata alla nuova scena corrente;
- la Rightbar rigenera gli elenchi degli oggetti poiché la scena corrente è stata cambiata;
- la Sezione delle regole mostrerà quelle appartenenti alla nuova scena.

Capitolo 5

Conclusioni

Come discusso nell'introduzione, l'obiettivo della nostra tesi è quello di creare un'interfaccia di test del gioco che permetta all'utente che lo sta creando di verificare, controllare e testare tutte le parti del suo gioco proprio come un debugger di un ambiente di sviluppo. Per completare questo lavoro ci siamo serviti del web-framework A-Frame, che ha semplificato la creazione degli oggetti tridimensionali che generano la scena centrale della nostra interfaccia. Inoltre, la combinazione del framework JavaScript React e della architettura Flux hanno permesso di salvare gli stati dei nostri componenti e aggiornare immediatamemente la pagina HTML. Infine l'utilizzo del database non relazionale di Neo4j e la sua struttura a grafo ci ha permesso di scrivere facilmente le query per salvare lo stato del gioco e caricare un salvataggio.

5.1 Sviluppi futuri

Per quanto riguardo gli sviluppi futuri del nostro lavoro sarà necessario implementare una coda degli eventi in modo da poter passare alla regola successiva premendo il pulsante avanti posizionato a destra di ogni regola nell'apposita sezione. Sarà necessario fondere i componenti *DebugVRScene* e *VRScene* poichè hanno una funzionalità molto simile e tenerli separati risulta ridondante. Un'altra funzionalità da implementare sarebbe l'aggiunta della visibilità di un oggetto alle proprietà modificabili nel componente *DebugTab*.

Bibliografia

[1] Neo4j - Graph Database platform. https://it.wikipedia.org/wiki/Neo4j, 2019.

[2] Bootstrap. The most popular HTML, CSS and JS library. https://getbootstrap.com/, 2019.

[3] Cypher Query Language. https://neo4j.com/developer/cypher/, 2019.

[4] Sketchpad - precursore dei moderni CAD. https://it.wikipedia.org/wiki/Sketchpad, 2019.

[5] Wikipedia-Videogiochi Point And Click. https://it.wikipedia.org/wiki/Avventura_grafica#Punta_e_clicca.

[6] PAC-PAC - Videogiochi d'avventura e fiction interattive per la promozione del patrimonio ambientale e culturale. https://pac4pac.wordpress.com/, 2019.

[7] Jet Brains-WebStorm. WebStorm: The Smartest JavaScript IDE. https://www.jetbrains.com/webstorm/, 2019.

[8] Facebook-React. React - a JavaScript library for building user interfaces. https://reactjs.org/, 2019.

[9] Facebook-Flux. Flux architecture – overview. https://facebook.github.io/flux/docs/overview.html, 2019.

[10] Mozilla-Aframe. https://aframe.io/docs/0.9.0/, 2019.

[11] Three.js. JavaScript 3D Library. https://threejs.org/, 2019.

[12] Thomas T Hewett, Ronald Baecker, Stuart Card, Tom Carey, Jean Gasen, Marilyn Mantei, Gary Perlman, Gary Strong, and William Verplank. *ACM SIGCHI curricula for human-computer interaction.* ACM, 1992.

Elenco delle figure

Ringraziamenti

Giunti alla fine di questo percorso, ci teniamo a ringraziare il nostro relatore Davide Spano per averci dato la possibilità di lavorare a questo progetto e soprattutto per la disponibilità e supporto che ci ha dato durante il periodo di lavoro.

Ringraziamo anche Alessandro, Fabio e Martina che si sono dimostrati sempre disponibili ad aiutarci durante lo sviluppo del progetto.