UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
Dipartimento di Ingegneria dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica e dell'Automazione

TESI DI LAUREA

Progettazione e implementazione di un serious game per la descrizione delle patologie causate dai virus sui cani da utilizzarsi in un Corso di Laurea in Veterinaria

Design and implementation of a serious game for the description of pathologies caused by viruses on dogs to be used in a Veterinary Degree Course

Relatore

Prof. Domenico Ursino

Candidato

Margherita Galeazzi

ANNO ACCADEMICO 2022-2023

"Only those who dare to fail greatly
can ever achieve greatly."

Robert F. Kennedy

Sommario

I serious game, in concerto con la gamification e con l'e-learning, stanno assumendo una sempre crescente rilevanza nell'ambito dell'apprendimento e stanno cambiando il modo in cui gli esseri umani apprendono nozioni e competenze. In questa tesi vengono descritte la progettazione e l'implementazione di un serious game per la descrizione delle patologie causate dai virus sui cani, da utilizzarsi in un corso di laurea veterinaria. Nello specifico è stata realizzata una fase di raccolta dei requisiti alla quale è seguita, poi, un'analisi degli stessi. Quest'ultima attività ha permesso la definizione delle classi, mentre il funzionamento generale del software è stato descritto mediante il diagramma di attività. Infine, il comportamento del programma è stato analizzato attraverso i diagrammi dei casi d'uso e di sequenza. Al seguito della fase di progettazione, attraverso l'utilizzo del motore grafico multipiattaforma Unity, abbiamo implementato, e successivamente testato, quanto è stato visto.

Keyword: Serious Game, E-learning, Gamification, Diagrammi UML, Specifica e Analisi dei Requisiti, Unity, Apprendimento, Virus, Motori grafici, Veterinaria, Blender, Modelli 3D, Animazioni

Indice

Elenco delle figure

Elenco delle tabelle

Elenco dei listati

Introduzione

I serious game rappresentano una tecnologia innovativa, nel campo dell'intrattenimento, che sta assumendo una sempre maggior rilevanza in diversi settori, quali l'istruzione, la salute e la simulazione. Lo scopo di questi giochi è quello di insegnare, addestrare o sensibilizzare gli utenti, combinando la componente ludica e intrattenitiva dei giochi con scopi seri ed educativi.

A differenza dei tradizionali metodi di insegnamento e formazione, i serious game consentono di apprendere attraverso l'esperienza diretta, la risoluzione di problemi e la simulazione di contesti reali creando, così, un ambiente di apprendimento motivante e coinvolgente. Attraverso l'immersione in scenari virtuali e la partecipazione attiva, gli utenti possono acquisire competenze e conoscenze in modo pratico ed esperienziale.

In questo contesto, la presente tesi si propone di esaminare la creazione e l'implementazione di un serious game dedicato alla descrizione delle patologie virali nei cani. La decisione di sviluppare un serious game per tale scopo è stata motivata da diverse ragioni di rilevanza educativa, scientifica e didattica.

La principale motivazione dietro lo sviluppo di questo serious game è il desiderio di rendere l'esperienza formativa degli studenti di veterinaria, circa le patologie virali nei cani, più dinamica e interessante, rispetto ai tradizionali metodi didattici, basati su letture di testi e presentazioni frontali, i quali possono risultare passivi e meno coinvolgenti per gli studenti.

Inoltre, i serious game offrono un ambiente di apprendimento esperienziale in cui gli studenti possono sperimentare direttamente il ruolo del virus e comprendere meglio le dinamiche della malattia nei cani. Questo approccio favorisce una comprensione più profonda della trasmissione, della sintomatologia e delle lesioni causate, appunto, da questi virus.

Inoltre, il nostro serious game può preparare in modo più efficace gli studenti per la pratica clinica, offrendo loro l'opportunità di esercitarsi in un ambiente virtuale prima di affrontare situazioni reali.

Infine, il feedback istantaneo fornito dal gioco risulta essere prezioso, in quanto permette agli studenti di correggere gli errori e migliorare le loro abilità diagnostiche in tempo reale.

Il presente elaborato si propone, quindi, di esaminare il processo di progettazione del serious game, partendo dalla raccolta dei requisiti espressi da un referente della Facoltà di Veterinaria dell'Università di Padova, i quali ci hanno, poi, permesso di delineare il funzionamento che il nostro serious game doveva avere e a rappresentarlo mediante il diagramma di attività.

Tali requisiti hanno costituito la base iniziale del nostro lavoro, fornendoci una visione chiara delle esigenze e delle aspettative.

Al fine di garantire la corretta implementazione di tali specifiche, e per delineare il funzionamento del nostro serious game, abbiamo adottato un approccio metodologico completo e strutturato.

A seguito della raccolta dei requisiti, abbiamo condotto un'analisi approfondita delle necessità degli utenti finali, ovvero gli studenti di Veterinaria, e questo processo ci ha permesso di identificare i casi d'uso principali e di sviluppare i diagrammi dei casi d'uso, il cui obbiettivo è illustrare in modo esaustivo le interazioni tra gli utenti e il sistema di gioco.

Inoltre, per rendere più chiara la sequenza temporale delle azioni e delle interazioni all'interno del serious game, abbiamo creato un diagramma di sequenza dettagliato; questo offre una visione delle operazioni svolte dagli utenti durante il gioco e delle risposte fornite dal sistema, contribuendo, così, a una comprensione più approfondita del funzionamento del gioco stesso.

Successivamente siamo passati alla fase di sviluppo del serious game; come da requisiti abbiamo utilizzato il motore grafico Unity; abbiamo, quindi, implementato tutte le funzionalità richieste mediante script in C#.

Inoltre, in questa fase è risultato necessario ricorrere alla modellazione 3D e alla creazione di animazioni, poiché non vi erano modelli esistenti adatti allo scopo.

Per permettere l'utilizzo del nostro gioco, abbiamo redatto un manuale utente che descrive i comandi da utilizzare e mostra alcune schermate di gioco.

Al fine di individuare i punti di forza, di debolezza, le opportunità e le minacce del nostro serious game abbiamo effettuato un'analisi SWOT ed un'analisi comparativa con i software già presenti sul mercato.

L'obiettivo finale di questa tesi è quello di contribuire all'arricchimento del processo formativo e alla preparazione dei futuri veterinari nella comprensione delle patologie virali nei cani.

Il presente elaborato è composto da 6 capitoli strutturati come di seguito specificato:

- Nel Capitolo 1 riporteremo i concetti chiave dei serious game e la loro efficacia nel processo di apprendimento.
- Nel Capitolo 2 analizzeremo i requisiti, funzionali e non, che derivano dalle richieste effettuate dalla Facoltà di Veterinaria dell'Università di Padova.
- Nel Capitolo 3 riporteremo la progettazione del software, mostrando il diagramma delle classi, con i relativi metodi ed attributi, che costituiscono il nostro videogioco, e le relazioni che vi sono tra esse. Inoltre, analizzeremo i diagrammi di comportamento, che definiscono cosa il nostro software può fare in relazione all'interazione di un utente e come si articola la consecutio temporum delle varie operazioni.
- Nel Capitolo 4 mostreremo il linguaggio ed i tool utilizzati nella parte implementativa, descriveremo le fasi di modellazione del design ed, infine, illustreremo come il nostro serious game è stato implementato, attraverso delle porzioni di codice relative alle funzionalità principali.
- Nel Capitolo 5 riporteremo il manuale utente, nel quale verranno descritti i comandi per giocare e verranno mostrate delle schermate di gioco.
- Nel Capitolo 6 esamineremo il serious game, prodotto di questa tesi, mediante un'analisi SWOT ed effettueremo un'analisi comparativa con i software attualmente presenti sul mercato.

CAPITOLO 1

I serious game

Nel capitolo iniziale di questa tesi, si approfondiranno le caratteristiche alla base dei serious game, nonchè le caratteristiche che li differenziano dalle altre tipologie di giochi. Inoltre, verranno analizzate le diverse categorie di serious game, ponendo particolare attenzione a quelle impiegate nell'ambito formativo e della sensibilizzazione sociale.

1.1 Definizione e caratteristiche distintive dei serious game

I serious game differiscono dalle altre tipologie di giochi per lo scopo; infatti a differenza dei classici giochi, il cui obiettivo è il mero intrattenimento, i serious game hanno un fine educativo. Le caratteristiche dei serious game che li distinguono dalle altre tipologie di giochi sono:

- **Fine educativo**: come citato in precedenza, lo scopo dei serious game è quello educativo; questi giochi sono progettati per far si che chi li utilizzi apprenda competenze specifiche o nuove conoscenze. Sono, quindi, usati come strumenti formativi e didattici.
- **Coinvolgimento**: per riuscire nel suo intento, un serious game deve essere in grado di mantenere alto e costante il livello di attenzione del giocatore. Per realizzare questo obiettivo gli strumenti principali sono la grafica, i suoni e il meccanismo di gioco. Inoltre, se il gioco riesce a coinvolgere emotivamente il giocatore, ciò fa si che le informazioni veicolate mediante esso siano meglio apprese.
- **Contesto realistico e simulazione**: i serious game possono modellare ambienti virtuali che simulano situazioni reali; questo fa si che l'utente che li utilizza possa applicare le conoscenze apprese e possa provare nuove strategie, in questi contesti, senza ripercussioni sul mondo reale.
- **Feedback**: in un serious game i feedback sono essenziali; essi forniscono all'utente una valutazione del proprio progresso, mettendo in luce gli aspetti su cui migliorare. I feedback possono essere di diverso tipo. Come, ad esempio, un punteggio, il raggiungimento di determinati traguardi, e così via; l'importante è che non distraggano l'utente dall'obiettivo formativo del gioco.

Altre caratteristiche dei serious game, che sono però possedute anche dai giochi di diversa tipologia sono:

- **Interattività**: i giocatori prendono parte attivamente al gioco, affrontando sfide e prendendo decisioni; ciò promuove l'apprendimento attivo e l'intrattenimento.
- **Progressione graduale**: i giochi sono strutturati in livelli il cui grado di difficoltà aumenta in maniera graduale consentendo agli utenti di sviluppare abilità e conoscenze in maniera progressiva.

1.2 I fattori chiave per l'efficacia di un serious game

I fattori fondamentali per il successo di un serious game si dividono in aspetti psicologici e aspetti implementativi e di design del gioco stesso. Entrambe le categorie, però, garantiscono il coinvolgimento e la predisposizione all'apprendimento del giocatore, decretando il successo del gioco.

1.2.1 Fattori psicologici

Prospettiva motivazionale

La natura interattiva e competitiva dei videogiochi porta a far si che si trascorra un tempo prolungato a giocarvi, con un'intensa concentrazione senza, però, percepire il tempo trascorso e la fatica. Inoltre se le sfide proposte dal gioco sono di difficoltà progressiva e tarate sulle capacità del giocatore, il gioco sarà in grado di soddisfare specifici bisogni intrinseci di quest'ultimo, quali la competenza e l'autonomia.

Prospettiva emotiva

Il coinvolgimento attivo del giocatore, associato al giusto equilibrio nella difficoltà della sfida, come accennato sopra, può anche influenzare le emozioni dei giocatori. Questo risultato può essere spiegato attraverso la teoria del valore di controllo (CVT) delle emozioni di realizzazione; una sfida ottimale (nè troppo difficile, nè troppo semplice) potrebbe favorire il grado di controllo percepito, che è legato a esperienze emotive positive.

1.2.2 Fattori implementativi e di design

Modalità di gioco

Si può optare per una modalità di gioco competitiva o collaborativa. Entrambe le tipologie hanno i loro vantaggi e svantaggi; in particolar modo, per quanto riguarda la modalità competitiva, essa potrebbe portare il giocatore ad essere più coinvolto e a cercare di scalare le classifiche migliorando progressivamente; per contro, potrebbe portare certi giocatori a non sentirsi all'altezza e, quindi, a perdere interesse nel gioco stesso.

Grado di controllo

Il grado di controllo fornito all'utente, ovvero le possibilità di personalizzazione dell'ambiente di gioco insieme alle possibili azioni che l'utente può svolgere nel modo stesso, aumentano l'interesse del giocatore.

Componente narrativa

L'inclusione di una narrazione aiuta a contestualizzare ciò che si sta apprendendo; è stato dimostrato che i serious game che contengono una parte narrativa favoriscono l'apprendimento.

Componente estetica

Varie ricerche sugli effetti dei colori, delle forme e del sottofondo musicale nei giochi hanno mostrato che i colori caldi e le forme arrotondate inducono emozioni positive e che il sottofondo musicale ha un impatto positivo sulla motivazione e sul divertimento. Va, però, prestata attenzione al fatto che la componente grafica, insieme a quella musicale, non deve distogliere l'attenzione dell'utente dall'obiettivo finale del gioco, ovvero l'apprendimento.

1.3 Tipologie di serious game

Nell'operare una classificazione dei serious game possono essere adottati differenti criteri, quali il settore di utilizzo, gli obiettivi o i risultati attesi. Per quanto riguarda gli obiettivi, i serious game si possono dividere in:

- **Educativi**: questi giochi sono progettati per insegnare concetti e competenze attraverso il gameplay; essi possono coprire un'ampia gamma di argomenti, quali l'alfabetizzazione, l'apprendimento delle materie scientifiche e matematiche e di quelle umanistiche.
- **Simulazione e addestramento**: questi giochi sono progettati per fornire scenari realistici e immersivi che aiutano i giocatori a sviluppare competenze specifiche, potendo sperimentare diverse strategie senza ripercussioni nel mondo reale. Questi giochi vengono utilizzati in settori come quello militare, la sicurezza, l'aviazione, la medicina e la gestione delle emergenze.
- **Salute e benessere**: questi giochi hanno il fine di promuovere comportamenti salutari, prevenire malattie o gestire condizioni mediche specifiche. Essi possono essere utilizzati per formare il personale sanitario, educare i pazienti o incoraggiare comportamenti di vita sani.
- **Problem solving**: questi giochi sono progettati per aiutare gli utenti a sviluppare abilità di pianificazione, analisi e problem-solving; essi vengono solitamente utilizzati in contesti aziendali o governativi ed aiutano a prendere decisioni informate.

1.4 L'efficacia del gameplay nel processo di apprendimento

L'efficacia dei serious game è probabilmente legata a processi ben noti del mondo della neuropsicologia. In particolare, sono da prendere in considerazione i seguenti processi:

- Il **sistema attentivo**: l'attenzione è una funzione cognitiva trasversale che può rafforzare a cascata tutto il resto del sistema cognitivo. Un buon serious game riesce a mantenere elevata la soglia di attenzione, assicurando un maggiore coinvolgimento cognitivo (Figura 1.1);
- Il **modello motivazione/prestazione**: Atkinson, nel suo modello motivazionale ad "U" rovesciata, ipotizza che le performance migliori si ottengono con una quota equilibrata

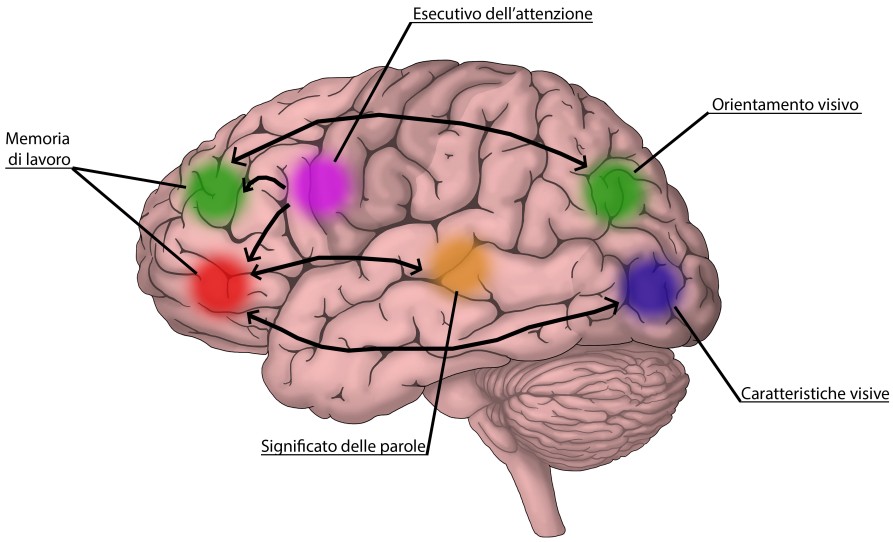


Figura 1.1: Rappresentazione del sistema attentivo "anteriore"

di motivazione. I serious game spesso riescono a centrare questa finestra di motivazione ottimale. Secondo Atkinson la motivazione alla riuscita dipende da due componenti, o tendenze motivazionali, contrapposti, speculari e potenzialmente conflittuali: la tendenza al successo e la tendenza ad evitare il fallimento.

Mentre la tendenza al successo porta a scegliere compiti di media difficoltà, in genere leggermente più difficili di quelli che già si sanno fare, la tendenza ad evitare il fallimento porta ad affrontare compiti più facili o più difficili, in quanto, in questo caso, l'insuccesso si reputa dovuto alla difficoltà e non alla mancanza di capacità. Ciò porta la motivazione ad avere un andamento a "U" rovesciata (Figura 1.2).

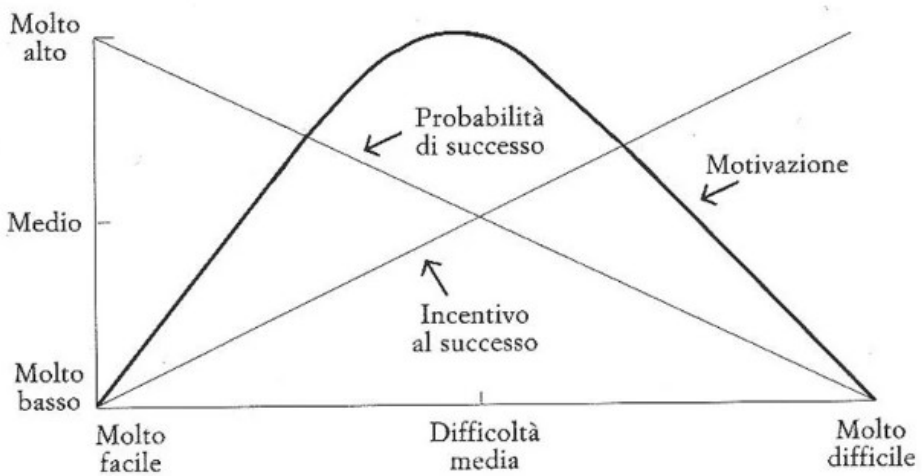


Figura 1.2: Il modello motivazione/prestazione di Atkinson

- Il **circuito del reward**: la ricompensa (reward) è un elemento fondamentale di qualsiasi gioco; essa alimenta il sistema attentivo e, sopratutto, quello motivazionale. Nella maggior parte dei serious game il sistema di reward si occupa anche della parte relativa ai feedback(Figura 1.3).

Va, però, sottolineato che, al giorno d'oggi, non esiste un modello unificato che spieghi l'efficacia dei serious game. Ciò, probabilmente, è dovuto al fatto che ogni serious game ha

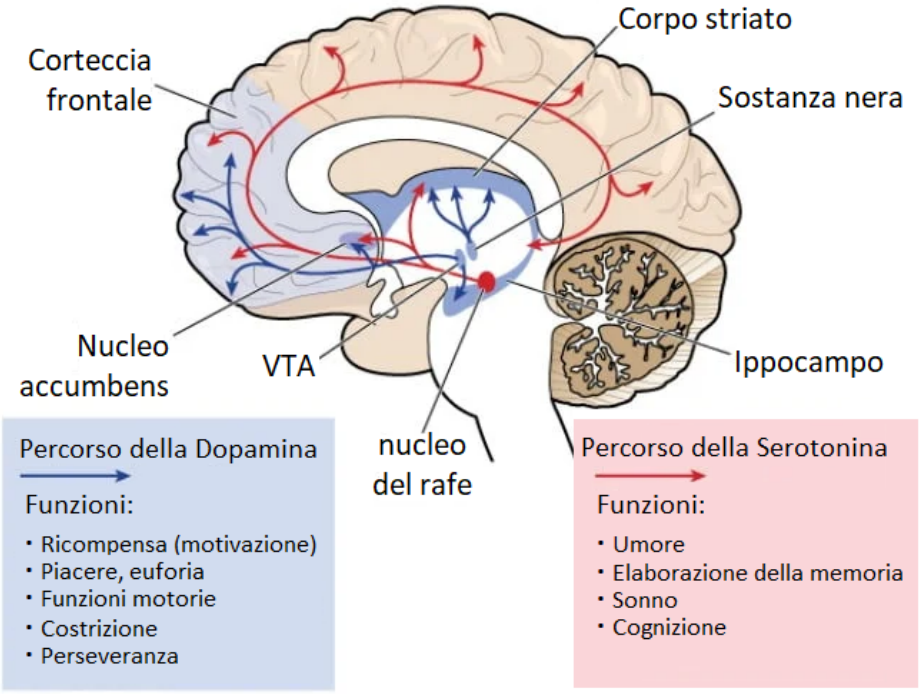


Figura 1.3: Il circuito del reward

caratteristiche che lo differenziano dagli altri e, quindi, la creazione di un modello unico risulta estremamente difficile.

CAPITOLO 2

Specifiche dei requisiti

Il capitolo corrente ha come obiettivo quello di analizzare le specifiche esposte dal cliente da documenti che esso ci ha fornito, procedendo poi con una raffinazione della descrizione del contesto di riferimento mediante una descrizione in linguaggio naturale. In seguito verrà illustrato il glossario dei termini, necessario per dettagliare i termini che potrebbero generare ambiguità. Infine, verranno esposte le specifiche del sistema attraverso l'analisi dei requisiti. Questi ultimi definiscono le funzionalità che il sistema stesso dovrà fornire ed, inoltre, mettono in evidenza i vincoli operativi da rispettare. Verranno, quindi, analizzati i requisiti, suddividendoli in funzionali e non funzionali.

2.1 Descrizione del sistema in linguaggio naturale

Il progetto proposto consiste nella realizzazione di un serious game per la descrizione delle patologie causate dai virus sui cani da utilizzarsi in un Corso di Laurea in Veterinaria.

Gli aspetti di interesse risultano essere:

- la possibilità, da parte dell'utente, di selezionare il virus con il quale intende "giocare";
- la visualizzazione di spiegazioni circa il virus selezionato;
- la capacità di spostarsi dalla schermata di selezione a quella di gioco;
- la visualizzazione e la spiegazione di come avviene la trasmissione del virus;
- la visualizzazione e la spiegazione del ciclo di replicazione del virus;
- la possibilità, da parte dell'utente, di selezionare la lesione associata al virus scelto;
- la visualizzazione e la spiegazione della lesione provocata dal virus;
- la possibilità, da parte dell'utente, di selezionare i sintomi del virus scelto;
- la visualizzazione di un'animazione relativa ai sintomi del virus.

2.2 Glossario dei termini

Il glossario dei termini è un insieme di definizioni che forniscono spiegazioni chiare e concise per le parole che possono causare ambiguità o incertezza nel contesto in cui vengono utilizzate. Questo strumento è utile per garantire una comunicazione accurata e comprensibile. Nella Tabella 2.1 riportiamo il glossario dei termini di nostro interesse.

Termine	Descrizione	Sinonimi
Utente	La persona che giocherà con il serious game sviluppato	Giocatore
Patologia	Indica una malattia in atto, uno stato patologico, una condizione di sofferenza dell'organismo	Malattia, Morbo, Disfunzione, Disturbo
Virus	Un virus è un agente infettivo microscopico composto da acido nucleico (DNA o RNA) avvolto da una proteina, che può infettare e replicarsi all'interno di cellule vive degli organismi	Germe, Microbo, Agente patogeno
Lesione	Un danno o un'anomalia causata dall'infezione virale che colpisce un tessuto o un organo nel corpo del cane	Effetto della malattia
Sintomi	Manifestazioni cliniche, o segni evidenti di una malattia o infezione virale, che possono includere cambiamenti nel comportamento, nel corpo o nella salute generale del cane	Segni
Animazione	Una sequenza di immagini o video che mostra in modo dinamico e interattivo come si manifestano i sintomi del virus selezionato	Clip
Selezione	La possibilità per l'utente di scegliere un determinato virus con cui interagire nel gioco	Scelta
Visualizzazione	Mostrare all'utente informazioni o immagini relative al virus, alla trasmissione, alla lesione o ai sintomi selezionati	

Tabella 2.1: Glossario dei termini relativi al serious game da realizzare

2.3 Specifiche dei requisiti

Prima di entrare nei dettagli dei requisiti, è fondamentale distinguere e classificare questi ultimi in due categorie principali, ovvero requisiti funzionali e requisiti non funzionali.

La distinzione tra requisiti funzionali e non funzionali è cruciale per una corretta analisi e progettazione. Per il nostro sistema, la suddivisione è rappresentata in Figura 2.1.

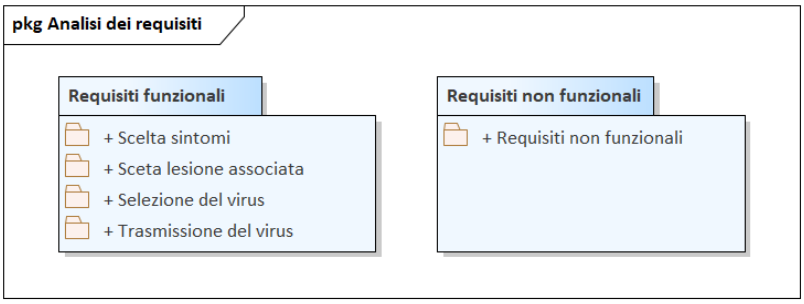


Figura 2.1: Analisi dei requisiti, funzionali e non funzionali

2.4 Requisiti funzionali

I requisiti funzionali descrivono, nel linguaggio naturale, le funzionalità e/o i servizi che il sistema dovrà fornire, a seguito della richiesta, o dell'azione, di un utente; in altre parole, essi specificano cosa il sistema dovrebbe essere in grado di fare, le azioni che dovrebbe svolgere o i risultati che dovrebbe produrre. Nel nostro caso i requisiti funzionali riguardano le interazioni che l'utente può avere con il software.

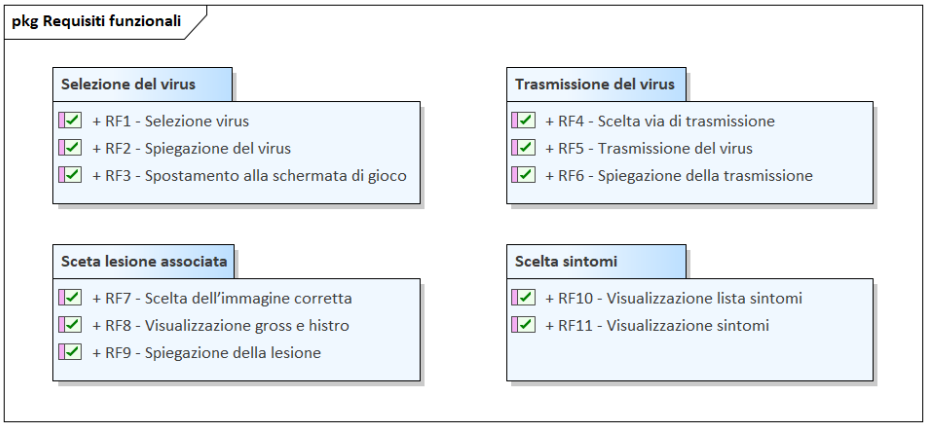


Figura 2.2: Analisi dei requisiti funzionali relativi al nostro sistema

2.4.1 Selezione del virus

Nella Tabella 2.2 vengono riportati i requisiti funzionali relativi alla selezione del virus.

Requisito	Descrizione
RF1 - Selezione virus	Il sistema dovrà consentire di scegliere il virus tra tutti quelli presenti
RF2 - Spiegazione del virus	Il sistema dovrà consentire la visualizzazione di una breve spiegazione sui virus da selezionare
RF3 - Spostamento alla schermata di gioco	Il sistema dovrà gestire lo spostamento dalla scena di selezione alla scena di gioco

Tabella 2.2: Requisiti funzionali: Selezione del virus

2.4.2 Trasmissione del virus

Nella Tabella 2.3 vengono riportati i requisiti funzionali relativi alla trasmissione del virus.

Requisito	Descrizione
RF4 - Scelta via di trasmissione	Il sistema dovrà consentire di scegliere la via di trasmissione mediante spostamento del virus stesso verso la zona target.
RF5 - Trasmissione del virus	Il sistema dovrà consentire la visualizzazione di un'animazione che mostra la modalità di trasmissione del virus, partendo dalla via di contagio ed arrivando agli organi target.
RF6 - Spiegazione della trasmissione	Il sistema dovrà consentire la visualizzazione di una spiegazione testuale di come il virus, una volta entrato nell'organismo, arriva all'organo target.

Tabella 2.3: Requisiti funzionali: Trasmissione del virus

2.4.3 Scelta della lesione associata

Nella Tabella 2.4 vengono riportati i requisiti funzionali relativi alla scelta della lesione causata dal virus.

Requisito	Descrizione
RF7 - Visualizzazione gross e immagini istologiche	Il sistema dovrà consentire di visualizzare immagini di gross e immagini istologiche
RF8 - Scelta dell'immagine corretta	Il sistema dovrà consentire di scegliere, tra le immagini presenti, quella corretta, e di notificare l'utente in caso di errore
RF9 - Spiegazione della lesione	Il sistema dovrà consentire la visualizzazione di un testo di spiegazione delle lesioni visibili nella gross e nelle immagini istologiche

Tabella 2.4: Requisiti funzionali: Scelta della lesione associata

2.4.4 Scelta dei sintomi del virus

Nella Tabella 2.5 vengono riportati i requisiti funzionali relativi alla scelta dei sintomi del virus.

Requisito	Descrizione
RF10 - Visualizzazione lista sintomi	Il sistema dovrà consentire di visualizzare una lista di possibili sintomi e di scegliere quelli corretti; in caso di errore dovrà effettuare l'opportuna notifica all'utente

Requisito	Descrizione
RF11 - Visualizzazione sintomi	Il sistema dovrà consentire, una volta scelti i sintomi corretti, di vedere una breve animazione relativa ai sintomi

Tabella 2.5: Requisiti funzionali: Scelta dei sintomi del virus

2.5 Requisiti non funzionali

I requisiti non funzionali riguardano gli aspetti di prestazione, affidabilità, sicurezza, usabilità e altri vincoli che il sistema deve soddisfare. Questi requisiti non riguardano direttamente le funzionalità specifiche del sistema, ma, piuttosto, le caratteristiche globali che esso deve possedere.

I requisiti non funzionali relativi al nostro sistema sono riportati in Figura 2.3.

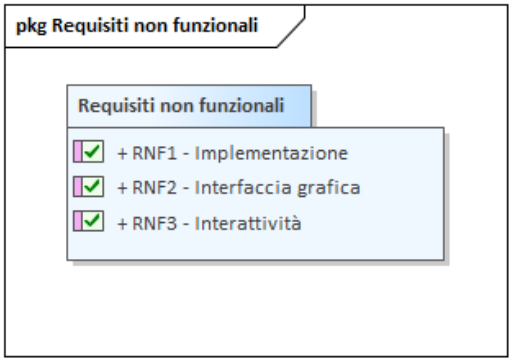


Figura 2.3: Analisi dei requisiti non funzionali

Nella Tabella 2.6 vengono riportati i requisiti non funzionali relativi al nostro sistema.

Requisito	Descrizione
RNF1 - Implementazione	Il sistema dovrà essere sviluppato utilizzando l'editor Unity e come linguaggio di programmazione il C#
RNF2 - Interfaccia grafica	Il sistema dovrà essere dotato di un'interfaccia grafica in 3D
RNF3 - Interattività	Il sistema dovrà offrire un'interattività dinamica per garantire un'esperienza coinvolgente e appagante per gli utenti

Tabella 2.6: Dettaglio dei requisiti non funzionali del sistema

CAPITOLO 3

Progettazione

In questo capitolo verranno trattate le fasi di progettazione e design del serious game oggetto di questa tesi. Si mostreranno i diagrammi delle classi, spiegando i vari attributi e metodi di ogni classe, nonché i diagrammi delle sequenze, che metteranno in luce le interazioni tra il giocatore e le classi e tra le classi stesse.

3.1 Obiettivi di progettazione e target di pubblico

L'obiettivo alla base dello sviluppo di questo progetto è la creazione di uno strumento di sostegno nell'apprendimento delle patologie causate dai virus ai cani. Nello specifico si intende mostrare in maniera semplice ed intuitiva come i virus infettino l'organismo dei cani partendo dalla loro trasmissione fino ad arrivare a vedere le lesioni da essi procurate e la sintomatologia connessa.

Il target di pubblico di questo videogioco sono gli studenti dei corsi di laurea in veterinaria.

3.2 Struttura del gioco

L'idea per la struttura del gioco è che esso sia composto da una sequenza di fasi nelle quali il giocatore dovrà:

1. *scegliere il virus*;
2. *muovere il virus* affinché arrivi alla via di trasmissione corretta;
3. *scegliere le micro-lesioni* associate al virus;
4. *scegliere le macro-lesioni* associate al virus;
5. *scegliere la sintomatologia* legata al virus.

3.3 Diagrammi delle classi

I Diagrammi delle classi sono uno strumento fondamentale nell'analisi e nella progettazione orientata agli oggetti, essi offrono una rappresentazione visuale delle classi mostrando le loro relazioni, gli attributi e i metodi che le compongono.

Le sezioni seguenti forniranno una descrizione approfondita di ogni classe, illustrandone gli attributi e i metodi.

3.3.1 Subtitle

Il package `Subtitle` contiene le classi necessarie ad estendere la timeline di Unity affinché sia possibile avere un sistema per gestire sottotitoli. Le classi che costituiscono questo package sono mostrate nella Figura 3.1.

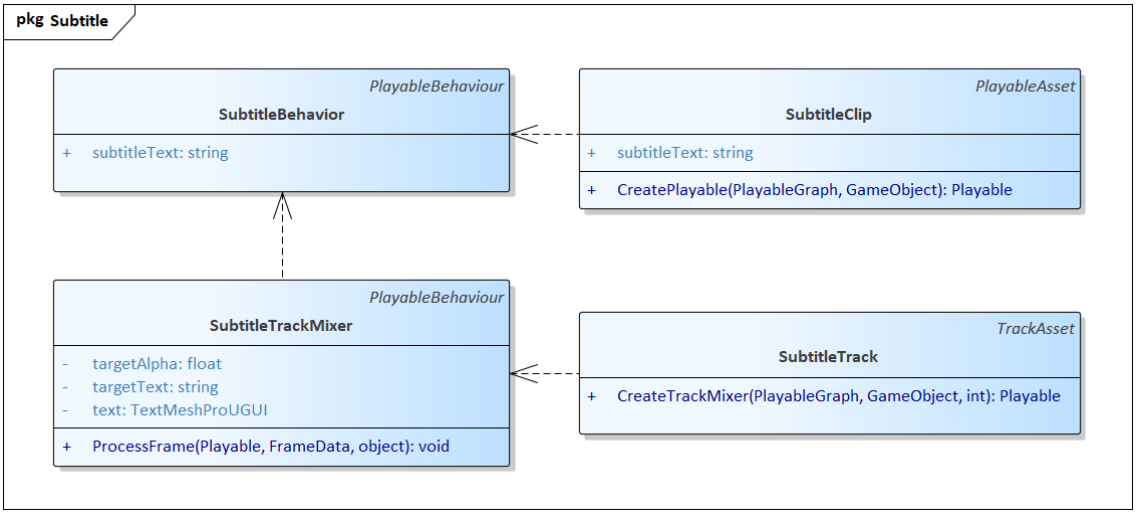


Figura 3.1: Package `Subtitle` e relative classi

SubtitleBehaviour

Questa classe estende `PlayableBehaviour` ed è responsabile di memorizzare il testo del sottotitolo. Viene utilizzata come componente per le clip dei sottotitoli. Gli elementi che la costituiscono sono mostrati nella Tabella 3.1.

Attributi		
Nome	**Tipo**	**Descrizione**
`subtitleText`	`string`	Memorizza il testo del sottotitolo per la clip

Tabella 3.1: Package `Subtitle`: elementi della classe `SubtitleBehavior`

SubtitleClip

Questa classe estende `PlayableAsset` ed è responsabile della creazione di un oggetto `Playable` per la clip dei sottotitoli. Viene utilizzata come asset per la traccia dei sottotitoli. Gli elementi che la costituiscono sono mostrati nella Tabella 3.2, mentre le relazioni sono mostrate nella Tabella 3.3.

Attributi		
Nome	**Tipo**	**Descrizione**
`subtitleText`	`string`	Memorizza il testo del sottotitolo per la clip
Metodi		
Nome	**Tipo**	**Descrizione**
`CreatePlayable`	`Playable`	Crea un oggetto `Playable` per la clip dei sottotitoli. Viene chiamato dal sistema di timeline di Unity. La funzione crea un oggetto `ScriptPlayable<SubtitleBehavior>` e imposta il testo del sottotitolo nel comportamento della clip

Tabella 3.2: Package `Subtitle`: relazioni della classe `SubtitleClip`

Elemento	Tipo
`SubtitleBehavior`	Dependency

Tabella 3.3: Package `Subtitle`: relazioni della classe `SubtitleClip`

SubtitleTrack

Questa classe estende `TrackAsset` ed è responsabile della creazione del mixer per la traccia dei sottotitoli. Gli elementi che la costituiscono sono mostrati nella Tabella 3.4, mentre le relazioni sono mostrate nella Tabella 3.5.

Metodi		
Nome	**Tipo**	**Descrizione**
`CreateTrackMixer`	`Playable`	Crea un oggetto `Playable` per il mixer della traccia dei sottotitoli. Viene chiamato dal sistema di timeline di Unity. La funzione crea un oggetto `ScriptPlayable<SubtitleTrackMixer>`.

Tabella 3.4: Package `Subtitle`: elementi della classe `SubtitleTrack`

Elemento	Tipo
`SubtitleTrackMixer`	Dependency

Tabella 3.5: Package `Subtitle`: relazioni della classe `SubtitleTrack`

SubtitleTrackMixer

Questa classe estende `PlayableBehaviour` ed è responsabile della gestione dei sottotitoli sulla traccia dei sottotitoli. Gli elementi che la costituiscono sono mostrati nella Tabella 3.7, mentre le relazioni sono mostrate nella Tabella 3.8.

Attributi		
Nome	**Tipo**	**Descrizione**
`targetAlpha`	`float`	Memorizza il valore corrente dell'opacità del testo
`targetText`	`string`	Memorizza il testo da visualizzare
`text`	`TextMeshProUGUI`	Memorizza un riferimento a un oggetto `TextMeshProUGUI` in cui verranno visualizzati i sottotitoli

Metodi		
Nome	**Tipo**	**Descrizione**
`ProcessFrame`	`void`	Questa funzione viene chiamata ogni frame dal sistema di timeline di Unity. Processa i frame per determinare il testo del sottotitolo e l'opacità corrispondenti all'istante corrente della timeline. Aggiorna quindi l'oggetto `TextMeshProUGUI` con il testo e l'opacità corrispondenti

Tabella 3.7: Package `Subtitle`: elementi della classe `SubtitleTrackMixer`

Elemento	Tipo
`SubtitleBehavior`	Dependency

Tabella 3.8: Package `Subtitle`: relazioni della classe `SubtitleTrackMixer`

3.3.2 GameLogic

Il package `GameLogic` contiene le classi relative alla logica del gioco, quindi quelle che consentono di selezionare il virus con il quale giocare, vedere le informazioni relative ad esso, far muovere il virus fino alla via di trasmissione, mostrare le animazioni e far scegliere al giocatore le foto relative alle lesioni provocate dal virus e i sintomi ad esso relativi. Le classi che costituiscono questo package sono mostrate nella Figura 3.2.

VirusDB

La classe `VirusDB` è una classe che eredita da `ScriptableObject` ed è utilizzata per memorizzare un database di virus. La classe è annotata con l'attributo `CreateAssetMenu`, che consente di creare istanze di questa classe come asset all'interno dell'Editor di Unity. Gli elementi che la costituiscono sono mostrati nella Tabella 3.9, mentre le relazioni sono mostrate nella Tabella 3.10.

Attributi		
Nome	**Tipo**	**Descrizione**
`viruses`	`Virus[]`	Un array di oggetti `Virus` che rappresenta ciascun virus nel database. Ogni elemento dell'array contiene informazioni specifiche sul virus
`NumVirus`	`int`	Una proprietà di sola lettura che restituisce il numero totale di virus presenti nel database. La lunghezza dell'array `viruses` viene restituita come risultato
Metodi		
Nome	**Tipo**	**Descrizione**
`GetVirus(int index)`	`Virus`	Un metodo che restituisce un oggetto `Virus` specifico nel database in base all'indice fornito. Esso permette di ottenere le informazioni di un virus specifico utilizzando l'indice desiderato

Tabella 3.9: Package `GameLogic`: elementi della classe `VirusDB`

Elemento	Tipo
`Virus`	Association

Tabella 3.10: Package `GameLogic`: relazioni della classe `VirusDB`

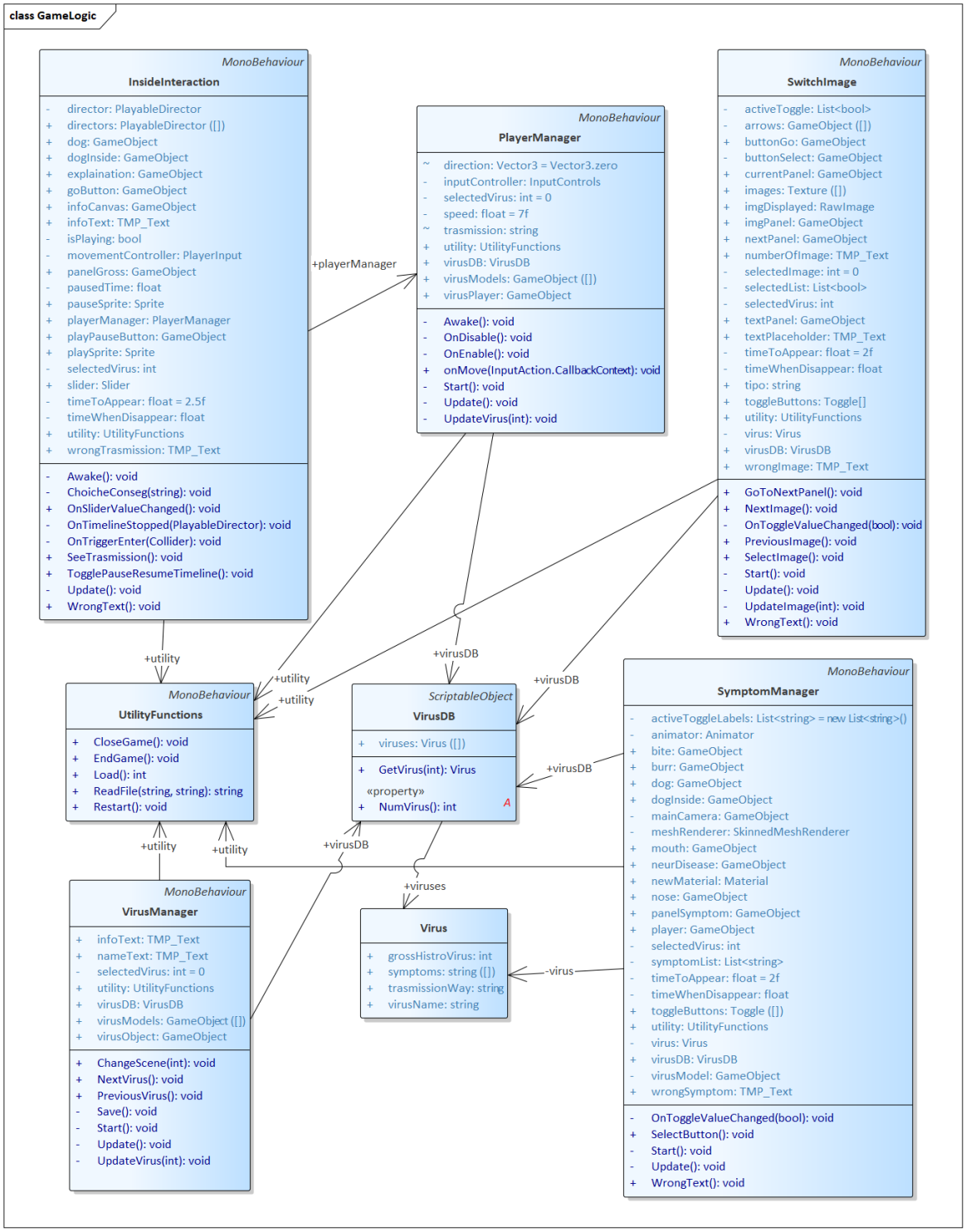


Figura 3.2: Package `GameLogic` e relative classi

Virus

La classe `Virus` è una classe serializzabile che rappresenta un virus nel contesto del sistema di gestione del database dei virus. Gli elementi che la costituiscono sono mostrati nella Tabella 3.11.

Attributi		
Nome	**Tipo**	**Descrizione**
`virusName`	`string`	Una stringa che rappresenta il nome del virus
`trasmissionWay`	`string`	Una stringa che rappresenta il modo in cui il virus viene trasmesso. Questo attributo fornisce informazioni sulle modalità di diffusione del virus
`grossHistoVirus`	`int`	Un intero che serve per trovare le immagini di gross e le immagini istologiche relative al virus in oggetto
`symptoms`	`string[]`	Un array di stringhe che rappresenta l'elenco dei sintomi associati al virus. Questo attributo contiene una serie di sintomi che possono verificarsi dopo l'infezione da parte del virus

Tabella 3.11: Package `GameLogic`: elementi della classe `Virus`

La classe `Virus` permette di organizzare e memorizzare le informazioni specifiche relative a un virus nel database dei virus. Ogni istanza di Virus contiene i dati necessari per identificare il virus, descriverne la trasmissione, trovare le immagini delle lesioni ad esso associate e fornire l'elenco dei relativi sintomi.

VirusManager

La classe `VirusManager` è responsabile della gestione e visualizzazione delle informazioni sui virus all'interno della scena di selezione del virus che viene mostrata prima di iniziare il gioco o di un'interfaccia utente basata su Unity. Gli elementi che la costituiscono sono mostrati nella Tabella 3.12, mentre le relazioni sono mostrate nella Tabella 3.13.

Attributi		
Nome	**Tipo**	**Descrizione**
`virusDB`	`VirusDB`	Una variabile di tipo `VirusDB` che contiene il riferimento al database dei virus
`nameText`	`TMP_Text`	Rappresenta il riferimento all'oggetto di testo in cui verrà visualizzato il nome del virus
`infoText`	`TMP_Text`	Rappresenta il riferimento all'oggetto di testo in cui verranno visualizzate le informazioni del virus
`virusObject`	`GameObject`	Un oggetto di gioco che rappresenta il modello 3D del virus visualizzato nella scena
`selectedVirus`	`int`	Un intero che tiene traccia dell'indice del virus selezionato nel database
`virusModels`	`GameObject[]`	Un array di oggetti di gioco che rappresentano i modelli 3D dei virus nel database. Ogni modello corrisponde a un virus nel database
`utility`	`UtilityFunctions`	Rappresenta il riferimento allo script contenente funzioni di "utility" condivise da più classi
Metodi		
Nome	**Tipo**	**Descrizione**
`Start()`	`void`	Questa funzione viene chiamata all'avvio della scena. Controlla se è presente un valore memorizzato nella chiave `selectedVirus` nel `PlayerPrefs`. Se non è presente, imposta `selectedVirus` su 0 e istanzia il primo modello di virus nella posizione desiderata. Altrimenti, carica il valore memorizzato. Poi istanzia il modello di virus corrispondente. Infine, chiama `UpdateVirus()` per visualizzare il virus selezionato
`Update()`	`void`	Questa funzione viene chiamata ad ogni frame dell'aggiornamento. Ruota l'oggetto `virusObject` lungo l'asse Y

Nome	Tipo	Descrizione
`NextVirus()`	`void`	Questa funzione viene chiamata quando viene premuto il pulsante "Next" per passare al virus successivo nel database. Incrementa il valore di `selectedVirus` e controlla se è stato raggiunto l'ultimo virus nel database. Se sì, riporta `selectedVirus` a 0. Infine, chiama `UpdateVirus()` per visualizzare il nuovo virus selezionato e salva l'indice del virus selezionato
`PreviousVirus()`	`void`	Questa funzione viene chiamata quando viene premuto il pulsante "Previous" per passare al virus precedente nel database. Decrementa il valore di `selectedVirus` e controlla se è stato raggiunto il primo virus nel database. Se sì, imposta `selectedVirus` all'indice dell'ultimo virus nel database. Infine, chiama `UpdateVirus()` per visualizzare il nuovo virus selezionato e salva l'indice del virus selezionato
`UpdateVirus(int selectedVirus)`	`void`	Questa funzione aggiorna le informazioni sul virus selezionato. Ottiene l'istanza del virus dal database utilizzando l'indice fornito. Istanzia il modello 3D corrispondente al virus selezionato e aggiorna `virusObject` con il nuovo modello. Aggiorna il testo dei campi `nameText` e `infoText` con il nome e le informazioni del virus selezionato
`Save()`	`void`	Questa funzione salva il valore corrente di `selectedVirus` nel `PlayerPrefs`
`ChangeScene(int sceneID)`	`void`	Questa funzione cambia la scena corrente caricando una nuova scena identificata da `sceneID`

Tabella 3.12: Package `GameLogic`: elementi della classe `VirusManager`

Elemento	Tipo
`VirusDB`	Association
`UtilityFunctions`	Association

Tabella 3.13: Package `GameLogic`: relazioni della classe `VirusManager`

In generale, la classe `VirusManager` si occupa di gestire la selezione dei virus, visualizzare i modelli 3D corrispondenti, mostrare i nomi e le informazioni dei virus, e consentire la navigazione tra i virus nel database.

PlayerManager

La classe `PlayerManager` gestisce il personaggio del giocatore nel gioco. Gli elementi che la costituiscono sono mostrati nella Tabella 3.14, mentre le relazioni sono mostrate nella Tabella 3.15.

Attributi		
Nome	**Tipo**	**Descrizione**
`virusDB`	`VirusDB`	Una variabile che contiene il riferimento al database dei virus
`virusPlayer`	`GameObject`	Un oggetto di gioco che rappresenta il modello 3D del personaggio (ovvero il virus) del giocatore
`selectedVirus`	`int`	Un intero che tiene traccia dell'indice del virus selezionato nel database
`virusModels`	`GameObjects[]`	Un array di oggetti di gioco che rappresentano i modelli 3D dei virus nel database. Ogni modello corrisponde a un virus nel database
`speed`	`float`	Una variabile che rappresenta la velocità di movimento del personaggio del giocatore
`direction`	`Vector3`	Un vettore di tipo `Vector3` che rappresenta la direzione di movimento del personaggio del giocatore
`inputController`	`inputControls`	Un'istanza di `InputControls` che gestisce gli input del giocatore utilizzando il sistema di input di Unity
`trasmission`	`string`	Una stringa che indica la corretta via di trasmissione del virus

Nome	Tipo	Descrizione
utility	UtilityFunctions	Rappresenta il riferimento allo script contenente funzioni di "utility" condivise da più classi
		Metodi
Nome	**Tipo**	**Descrizione**
Awake(), OnEnable(), OnDisable()	void	Queste funzioni gestiscono l'abilitazione e la disabilitazione del componente inputController
onMove(InputAction .CallbackContext callBackContext)	void	Questa funzione viene chiamata quando viene rilevato un input di movimento da parte del giocatore. Legge il valore del vettore di movimento da callBackContext e lo assegna alla variabile direction
Start()	void	Questa funzione viene chiamata all'avvio del gioco o della scena. Controlla se è presente un valore memorizzato nella chiave selectedVirus nel PlayerPrefs. Se non è presente, imposta selectedVirus su 0 e istanzia il primo modello di virus nella posizione desiderata. Altrimenti, carica il valore memorizzato in selectedVirus e istanzia il modello di virus corrispondente. Infine, chiama UpdateVirus() per visualizzare il virus selezionato
Update()	void	Questa funzione viene chiamata ad ogni frame dell'aggiornamento. Controlla se c'è un oggetto ostacolo davanti al personaggio del giocatore utilizzando un raycast. Se non ci sono ostacoli, sposta il personaggio del giocatore nella direzione specificata dalla variabile direction con una velocità basata sul valore di speed
UpdateVirus(int selectedVirus)	void	Questa funzione aggiorna il modello del personaggio del giocatore in base al virus selezionato. Ottiene l'istanza del virus dal database utilizzando l'indice fornito. Istanzia il modello 3D corrispondente al virus selezionato e aggiorna virusPlayer con il nuovo modello

Tabella 3.14: Package GameLogic: elementi della classe PlayerManager

Elemento	Tipo
VirusDB	Association
UtilityFunctions	Association

Tabella 3.15: Package GameLogic: relazioni della classe PlayerManager

In generale, la classe PlayerManager gestisce il movimento del personaggio del giocatore e l'aggiornamento del modello 3D del personaggio in base al virus selezionato. Utilizza il sistema di input di Unity per controllare il movimento del personaggio tramite le funzioni onMove() e Update().

InsideInteraction

La classe InsideInteraction gestisce l'attivazione della visualizzazione interna del cane e l'interazione del giocatore con essa. Questa classe controlla che la via di trasmissione sia corretta ed, in caso affermativo, attiva la visualizzazione delle informazioni circa la trasmissione del virus. Successivamente mostra un'animazione del percorso del virus dal punto di trasmissione fino al punto target. Gli elementi che la costituiscono sono mostrati nella Tabella 3.16, mentre le relazioni sono mostrate nella Tabella 3.17.

		Attributi
Nome	**Tipo**	**Descrizione**
playerManager	PlayerManager	Un riferimento all'oggetto PlayerManager che gestisce il personaggio del giocatore

Nome	Tipo	Descrizione
dog, dogInside	GameObject	Gli oggetti di gioco che rappresentano, rispettivamente, il cane visto dall'esterno e il cane visto dall'interno
wrongTrasmission	TMP_Text	Un componente TMP_Text che mostra un messaggio di errore nel caso in cui la via di trasmissione scelta dal giocatore sia errata
infoCanvas, goButton, explaination, panelGross	GameObject	Gli oggetti di gioco che rappresentano rispettivamente il pannello delle informazioni, il pulsante di conferma, le spiegazioni e il pannello delle immagini gross e delle immagini istologiche
infoText	TMP_Text	Un componente TMP_Text che visualizza il testo delle informazioni sul virus
timeToAppear, timeWhenDisappear	float	Variabili utilizzate per controllare il tempo di visualizzazione del messaggio di errore
directors	PlayableDirector[]	Un array di PlayableDirector che rappresentano le timeline associate ai diversi virus nel gioco
selectedVirus	int	Un intero che tiene traccia dell'indice del virus selezionato
director	PlayableDirector	Un riferimento a PlayableDirector che rappresenta la timeline associata al virus selezionato
movementController	PlayerInput	Un riferimento a PlayerInput che gestisce l'input del giocatore per il movimento
utility	UtilityFunctions	Rappresenta il riferimento allo script contenente funzioni di "utility" condivise da più classi
playPauseButton	GameObject	Rappresenta il riferimento al pulsante che permette di mettere in pausa la timeline e di farla ripartire
pausedTime	float	Variabili utilizzate per memorizzare l'istante in cui la timeline e stata messa in pausa
slider	Slider	Riferimento alllo slider che permette di gestire la timeline
pauseSprite, playSprite	Sprite	Le immagini per il pulsante di pausa e di avvio
isPlaying	bool	Una variabile boolena che dice se la timeline è in esecuzione o meno

Metodi

Nome	Tipo	Descrizione
OnTriggerEnter (Collider other)	void	Questa funzione viene chiamata quando il personaggio del giocatore collide con un oggetto nel gioco. Controlla se l'oggetto con cui si collide ha il tag wall. In caso negativo, chiama la funzione ChoicheConseg() passando il tag dell'oggetto come parametro
ChoicheConseg (string choice)	void	Questa funzione gestisce l'azione da intraprendere in base alla scelta della via di trasmissione effettuata dal giocatore. Se la scelta di quest'ultimo corrisponde alla corretta via di trasmissione del virus associato attiva la visualizzazione del cane dall'interno, disattiva la visualizzazione dall'esterno e mostra le informazioni sul virus. Altrimenti, visualizza un messaggio di errore e riposiziona il personaggio del giocatore in una posizione predefinita
Update()	void	Questa funzione viene chiamata ad ogni frame dell'aggiornamento. Controlla se il messaggio di errore è abilitato e se è il momento di disabilitarlo in base al tempo trascorso
WrongText()	void	Questa funzione abilita il messaggio di errore per un certo periodo di tempo
Awake()	void	Questa funzione viene chiamata all'avvio del gioco o della scena. Carica il valore di selectedVirus da PlayerPrefs e assegna il riferimento corrispondente a director. Inoltre, si registra per l'evento stopped di director per gestire l'azione quando la timeline si interrompe
OnTimelineStopped (PlayableDirector director)	void	Questa funzione viene chiamata quando la timeline associata al virus selezionato si interrompe. Disattiva l'input del movimento del personaggio del giocatore, nasconde le spiegazioni e il pannello delle informazioni e mostra il pannello delle immagini gross e delle immagini istologiche
SeeTrasmission()	void	Questa funzione viene chiamata quando il giocatore ha completato la lettura sulla trasmissione del virus e clicca sul pulsante avanti. Questo metodo avvia la timeline associata al virus selezionato e mostra le spiegazioni circa quanto si sta visualizzando
OnSliderValueChanged()	void	Questa funzione viene chiamata quando il giocatore sposta lo slider e di conseguenza la timeline viene aggiornata

Nome	Tipo	Descrizione
TogglePauseResume-Timeline()	void	Questa funzione viene chiamata quando il giocatore clicca l'apposito pulsante e mette in pausa o riprende la visione dell'animazione di trasmissione

Tabella 3.16: Package GameLogic: elementi della classe InsideInteraction

Elemento	Tipo
PlayerManager	Association
UtilityFunctions	Association

Tabella 3.17: Package GameLogic: relazioni della classe InsideInteraction

SwitchImage

La classe SwitchImage è responsabile della gestione delle immagini delle lesioni associate al virus. Questa classe permette all'utente di scorrere tra le diverse immagini e controlla se l'immagine selezionata corrisponde all'immagine delle lesioni del virus scelto. Gli elementi che la costituiscono sono mostrati nella Tabella 3.18, mentre le relazioni sono mostrate nella Tabella 3.19.

Attributi		
Nome	**Tipo**	**Descrizione**
images	Texture[]	Un array di immagini rappresentanti le immagini delle lesioni causate dal virus
tipo	string	Una stringa che dice se il pannello corrente è quello relativo alle immagini di gross o alle immagini istologiche
selectedImage	int	L'indice dell'immagine selezionata
imgDisplayed	RawImage	Un componente RawImage che visualizza l'immagine della lesione corrente
virusDB	VirusDB	Un riferimento al database dei virus
selectedVirus	int	L'indice del virus selezionato
currentPanel, nextPanel	GameObject	Gli oggetti di gioco che rappresentano, rispettivamente, il pannello delle immagini relative alle lesioni e il pannello dei sintomi
imgPanel,textPanel	GameObject	Gli oggetti di gioco che rappresentano, rispettivamente, la zona dove vengono mostrate le immagini e quella dove viene inserito il testo
wrongImage	TMP_Text	Un componente TMP_Text che mostra un messaggio di errore nel caso in cui l'immagine selezionata non corrisponda all'immagine di gross e alle immagini istologiche associata al virus
timeToAppear, timeWhenDisappear	float	Variabili utilizzate per controllare il tempo di visualizzazione del messaggio di errore
buttonSelect, buttonGo	GameObject	Gli oggetti di gioco che rappresentano, rispettivamente, il pulsante da cliccare per selezionare l'immagine che è attualmente visibile e quello per andare avanti una volta che si è letta la spiegazione della lesione
arrows	GameObject[]	Gli oggetti di gioco che rappresentano le frecce da cliccare per scorrere avanti o indietro nelle immagini
utility	Utility-Functions	Rappresenta il riferimento allo script contenente funzioni di "utility" condivise da più classi
toggleButtons	Toggle[]	Rappresenta i toggleButton relativi alla selezione o meno di ogni immagine istologica
activeToggle, selectedList	List<bool>	Liste di bool che indicano tutti i toggleButton attivi, e la lista che indica, per ogni virus, quali pulsanti devono essere attivi e quali no
numberOfImage	TMP_Text	Label che indica il numero di immagini che devono essere selezionate per ogni virus
virus	Virus	Oggetto di tipo virus che indica il virus selezionato

Nome	Tipo	Descrizione
`textPlaceholder`	`TMP_Text`	Area di testo che verrà riempita con le informazioni relative alle lesioni
	Metodi	
Nome	**Tipo**	**Descrizione**
`Start()`	`void`	Questo metodo viene chiamato all'avvio della scena. Inizializza il valore di `selectedVirus` da `PlayerPrefs` e imposta l'immagine della lesione da visualizzare inizialmente
`NextImage()`	`void`	Questo metodo viene chiamato quando viene premuto il pulsante "Next". Avanza alla prossima immagine, controllando se si è raggiunta la fine dell'array. Aggiorna quindi l'immagine visualizzata
`PreviousImage()`	`void`	Questo metodo viene chiamato quando viene premuto il pulsante "Previous". Torna all'immagine precedente, controllando se si è raggiunto l'inizio dell'array. Aggiorna, quindi, l'immagine visualizzata
`UpdateImage(int selectedImage)`	`void`	Questo metodo aggiorna l'immagine visualizzata in base all'indice dell'immagine selezionata
`selectImage()`	`void`	Questo metodo viene chiamato quando viene premuto il pulsante "Select". Verifica se l'immagine selezionata corrisponde all'immagine della lesione associata al virus selezionato. In caso affermativo nasconde il pannello delle immagini e mostra quello dei sintomi. Altrimenti, visualizza un messaggio di errore
`Update()`	`void`	Questo metodo viene chiamato ad ogni frame. Controlla se il messaggio di errore è abilitato e se è il momento di disabilitarlo in base al tempo trascorso
`WrongText()`	`void`	Questo metodo abilita il messaggio di errore per un certo periodo di tempo
`GoToNextPanel()`	`void`	Questo metodo permette di passare alla visualizzazione del pannello successivo

Tabella 3.18: Package `GameLogic`: elementi della classe `SwitchImage`

Elemento	Tipo
`VirusDB`	Association
`UtilityFunctions`	Association

Tabella 3.19: Package `GameLogic`: relazioni della classe `SwitchImage`

SymptomManager

La classe `SymptomManager` è responsabile della gestione, della selezione dei sintomi associati al virus selezionato nel gioco. Questa classe controlla i toggle button corrispondenti ai sintomi e verifica se quelli selezionati corrispondono ai sintomi del virus scelto. Gli elementi che la costituiscono sono mostrati nella Tabella 3.20, mentre le relazioni sono mostrate nella Tabella 3.21.

	Attributi	
Nome	**Tipo**	**Descrizione**
`toggleButtons`	`Toggle[]`	Un array di oggetti `Toggle` che rappresentano i pulsanti di selezione dei sintomi
`virusDB`	`VirusDB`	Un riferimento al database dei virus
`selectedVirus`	`int`	L'indice del virus selezionato
`activeToggleLabels`	`List<string>`	Una lista delle etichette dei pulsanti di selezione attivi
`symptomList`	`List<string>`	Una lista dei sintomi associati al virus selezionato

Nome	Tipo	Descrizione
panelSymptom, dogInside, dog, player, mouth, nose, bite, burr, neurDisease	GameObject	Gli oggetti di gioco che rappresentano, rispettivamente, il pannello dei sintomi, il cane visto dall'interno, il cane visto dall'esterno, il virus, la bocca del cane, il naso del cane, la ferita da morso che ha il cane sulla gamba, la saliva (dovuta alla rabbia) ed un pannello informativo
virusModel, mainCamera	GameObject	Gli oggetti di gioco che rappresentano, rispettivamente, il modello del virus e la videocamera
wrongSymptom	TMP_Text	Un componente TMP_Text che mostra un messaggio di errore nel caso in cui i sintomi selezionati non corrispondano ai sintomi associati al virus
timeToAppear, timeWhenDisappear	float	Variabili utilizzate per controllare il tempo di visualizzazione del messaggio di errore
animator	Animator	Un componente Animator che gestisce le animazioni del cane esterno
virus	Virus	L'oggetto Virus che rappresenta il virus selezionato
meshRenderer	SkinnedMeshRenderer	La variabile che rappresenta il componente che gestisce il rendering del modello 3D animato nella scena di Unity necessaria per impostare il corretto materiale da visualizzare per il modello, in base al virus con il quale si sta giocando
utility	UtilityFunctions	Rappresenta il riferimento allo script contenente funzioni di "utility" condivise da più classi
newMaterial	Material	Il materiale da utilizzare nel caso il virus scelto non sia quello della rabbia (texture senza morso)
	Metodi	
Nome	**Tipo**	**Descrizione**
Start()	void	Questo metodo viene chiamato all'avvio della scena. Inizializza il valore di selectedVirus da PlayerPrefs e registra gli eventi dei pulsanti di selezione dei sintomi
OnToggleValueChanged(bool isOn)	void	Questo metodo viene chiamato quando viene modificato lo stato di uno dei pulsanti di selezione dei sintomi. Aggiorna la lista delle etichette dei pulsanti di selezione attivi in base allo stato dei pulsanti
SelectButton()	void	Questo metodo viene chiamato quando viene premuto il pulsante "Select". Verifica se i sintomi selezionati corrispondono a quelli associati al virus selezionato. In caso affermativo gestisce le azioni per la conferma della selezione dei sintomi corretti. Altrimenti, visualizza un messaggio di errore
Update()	void	Questo metodo viene chiamato ad ogni frame. Controlla se il messaggio di errore è abilitato e se è il momento di disabilitarlo in base al tempo trascorso
WrongText()	void	Questo metodo abilita il messaggio di errore per un certo periodo di tempo

Tabella 3.20: Package GameLogic: elementi della classe SymptomManager

Elemento	Tipo
VirusDB	Association
Virus	Association
UtilityFunctions	Association

Tabella 3.21: Package GameLogic: relazioni della classe SymptomManager

UtilityFunctions

La classe UtilityFunctions contiene diversi metodi utili per alcune funzionalità di gioco. Gli elementi che la costituiscono sono mostrati nella Tabella 3.22.

Metodi		
Nome	**Tipo**	**Descrizione**
`ReadFile()`	`string`	Questo metodo è responsabile della lettura di un file di testo (`.txt`) che contiene informazioni sul virus. Riceve in ingresso come parametro: `virusName` (il nome del file da aprire). Il metodo apre il file specificato, legge tutto il suo contenuto come una stringa e lo restituisce
`Load()`	`int`	Questo metodo restituisce un intero, che rappresenta il virus selezionato ad inizio partita dal giocatore. Questo può essere usato per mantenere lo stato del gioco tra le sessioni o per altre funzionalità di salvataggio
`EndGame()`	`void`	Questo metodo viene chiamato quando il gioco termina, esso caricherà un'altra scena, la quale mostra un'interfaccia di fine partita
`Restart()`	`void`	Questo metodo permette di richiamare una funzione di "riavvio" nel gioco, in modo che il giocatore possa tornare alla schermata iniziale e ricominciare da capo

Tabella 3.22: Package `GameLogic`: elementi della classe `UtilityFunctions`

3.4 Diagramma di attività

I diagrammi di attività consentono di rappresentare graficamente il flusso delle attività, delle azioni e delle decisioni all'interno di un processo o di un sistema. Essi servono a fornire una visione chiara e strutturata di come le diverse azioni si svolgono nel tempo e di come esse si relazionano.

Nello specifico nell'ambito dello sviluppo del software, i diagrammi delle attività vengono utilizzati per definire il flusso delle operazioni all'interno di un programma o di un'applicazione. Questo aiuta i programmatori a comprendere come interagiscono le diverse parti del sistema e come le azioni vengono eseguite in sequenza. Nella Figura 3.3 viene mostrato il diagramma di attività che mostra la sequenza di azioni che compongono il gioco.

3.5 Diagrammi dei casi d'uso

Un caso d'uso può essere considerato come una semplice descrizione di ciò che un utente si aspetta in tale interazione. In questa prima sezione analizzeremo i casi d'uso del nostro serious game.

3.5.1 Selezione del virus

Questa prima sottosezione ha lo scopo di illustrare i casi d'uso riguardanti la selezione del virus, come riportato in Figura 3.4

Visualizzazione del virus

Questo caso d'uso si verifica all'avvio del gioco quando il giocatore, prima di effettuare la scelta del virus con cui giocare, può visionarli tutti, scorrendo. Le informazioni relative sono mostrate nella Tabella 3.23.

Elementi	**Descrizione**
Pre-Condizioni	Nessuna
Post-Condizioni	Nessuna

Elementi	Descrizione
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il gioco è avviato 2. Il sistema recupera il modello e le informazioni del virus che è al momento visibile 3. Il sistema mostra il modello e le informazioni
Sequenza degli eventi alternativa	Nessuna

Tabella 3.23: Caso d'uso relativo alla visualizzazione dei virus

Visualizzazione informazioni virus

Questo caso d'uso si verifica all'avvio del gioco da parte del giocatore, prima di effettuare la scelta del virus con cui giocare. Le informazioni relative sono mostrate nella Tabella 3.24.

Elementi	Descrizione
Pre-Condizioni	I file di testo contenenti le informazioni devono essere presenti nel sistema
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il gioco è avviato 2. Il sistema recupera le informazioni del virus che è al momento visibile
Sequenza degli eventi alternativa	Nessuna

Tabella 3.24: Caso d'uso relativo alla visualizzazione delle informazioni relative al virus

Caricamento delle informazioni

Questo caso d'uso si verifica all'avvio del gioco a parte del giocatore, prima di effettuare la scelta del virus con cui giocare. Le informazioni relative sono mostrate nella Tabella 3.25.

Elementi	Descrizione
Pre-Condizioni	I file di testo contenenti le informazioni e i modelli dei virus devono essere presenti nel sistema
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il gioco è avviato 2. Il sistema recupera le informazioni ed il modello del virus che è al momento visibile
Sequenza degli eventi alternativa	Nessuna

Tabella 3.25: Caso d'uso relativo al caricamento delle informazioni del virus

Selezione del virus

Questo caso d'uso si verifica quando il giocatore sceglie il virus con il quale giocare. Le informazioni relative sono mostrate nella Tabella 3.26.

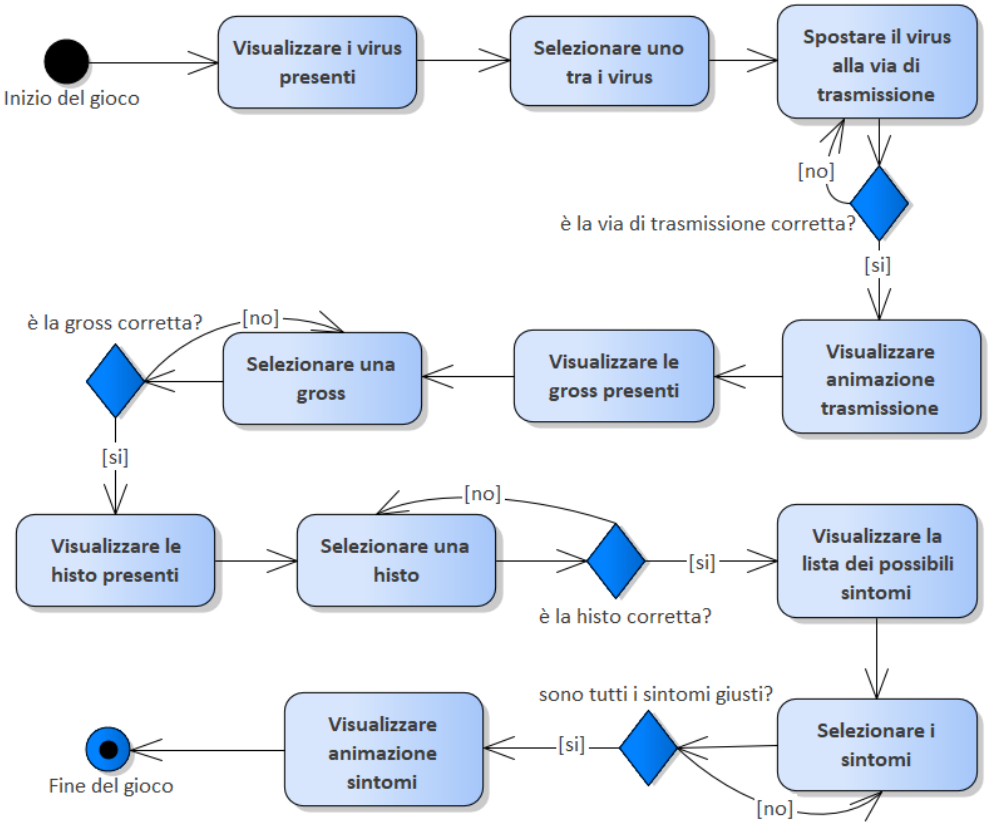


Figura 3.3: Diagramma di attività dell'intero gioco

Elementi	Descrizione
Pre-Condizioni	Nessuna
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore sceglie il virus con cui giocare 2. Il sistema salva il numero corrispondente al virus selezionato
Sequenza degli eventi alternativa	Nessuna

Tabella 3.26: Caso d'uso relativo alla selezione del virus con cui giocare

Salvataggio del virus scelto

Questo caso d'uso si verifica una volta che il giocatore ha selezionato il virus con il quale intende giocare. Le informazioni relative sono mostrate nella Tabella 3.27.

Elementi	Descrizione
Pre-Condizioni	Nessuna
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore ha scelto il virus con cui giocare 2. Il sistema memorizza il numero corrispondente al virus scelto (per garantire persistenza nelle sessioni di gioco)

Elementi	Descrizione
Sequenza degli eventi alternativa	Nessuna

Tabella 3.27: Caso d'uso relativo al salvataggio del virus scelto

3.5.2 Movimento del virus

Questa sottosezione ha lo scopo di illustrare i casi d'uso riguardanti il movimento del virus nella scena, come mostrato in Figura 3.5.

Spostamento nello spazio

Questo caso d'uso si verifica quando il giocatore muove il virus cliccando gli appositi tasti. Le informazioni relative sono mostrate nella Tabella 3.28.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato un virus
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore preme sui tasti per far muovere il virus 2. Il sistema sposta il modello del virus nella direzione relativa al tasto cliccato
Sequenza degli eventi alternativa	Nessuna

Tabella 3.28: Caso d'uso relativo allo spostamento nello spazio del virus

Raggiungimento via di trasmissione

Questo caso d'uso si verifica quando il giocatore muove il virus verso la via di trasmissione. Le informazioni relative sono mostrate nella Tabella 3.29.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato un virus
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore arriva ad una delle vie di trasmissione 2. Il sistema controlla che la via di trasmissione raggiunta sia quella corretta 3. Il sistema mostra l'anatomia del cane
Sequenza degli eventi alternativa	3. Il sistema mostra un messaggio di errore in quanto la via di trasmissione non è quella corretta 4. Il sistema riporta il virus nella posizione iniziale cosicchè il giocatore possa riprovare

Tabella 3.29: Caso d'uso relativo al raggiungimento della via di trasmissione

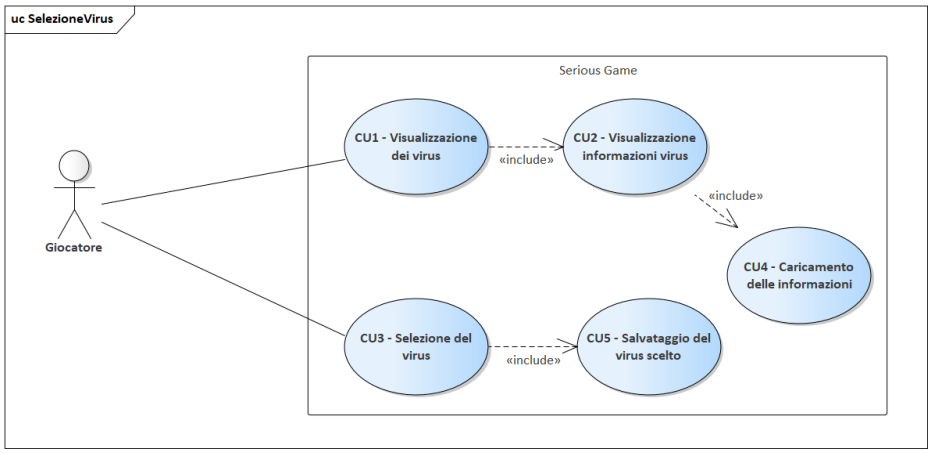


Figura 3.4: Diagramma dei casi d'uso relativi alla selezione del virus

Visualizzazione interno del cane

Questo caso d'uso si verifica quando il giocatore raggiunge la via di trasmissione corretta. Le informazioni relative sono mostrate nella Tabella 3.30.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato un virus e raggiunto la via di trasmissione corretta
Post-Condizioni	L'anatomia del cane è mostrata nella scena al posto del cane stesso
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore arriva alla via di trasmissione corretta 2. Il sistema mostra l'anatomia del cane al posto del modello del cane
Sequenza degli eventi alternativa	Nessuna

Tabella 3.30: Caso d'uso relativo alla visualizzazione dell'interno del cane

Segnalazione via di trasmissione errata

Questo caso d'uso si verifica quando il giocatore raggiunge la via di trasmissione errata. Le informazioni relative sono mostrate nella Tabella 3.31.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato un virus e raggiunto la via di trasmissione errata
Post-Condizioni	Il giocatore viene riportato alla posizione di partenza
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore arriva alla via di trasmissione errata 2. Il sistema mostra un messaggio di errore 3. Il sistema riporta il giocatore nella posizione iniziale
Sequenza degli eventi alternativa	Nessuna

Tabella 3.31: Caso d'uso relativo alla visualizzazione del messaggio di errore nel caso in cui il giocatore raggiunga la via di trasmissione sbagliata

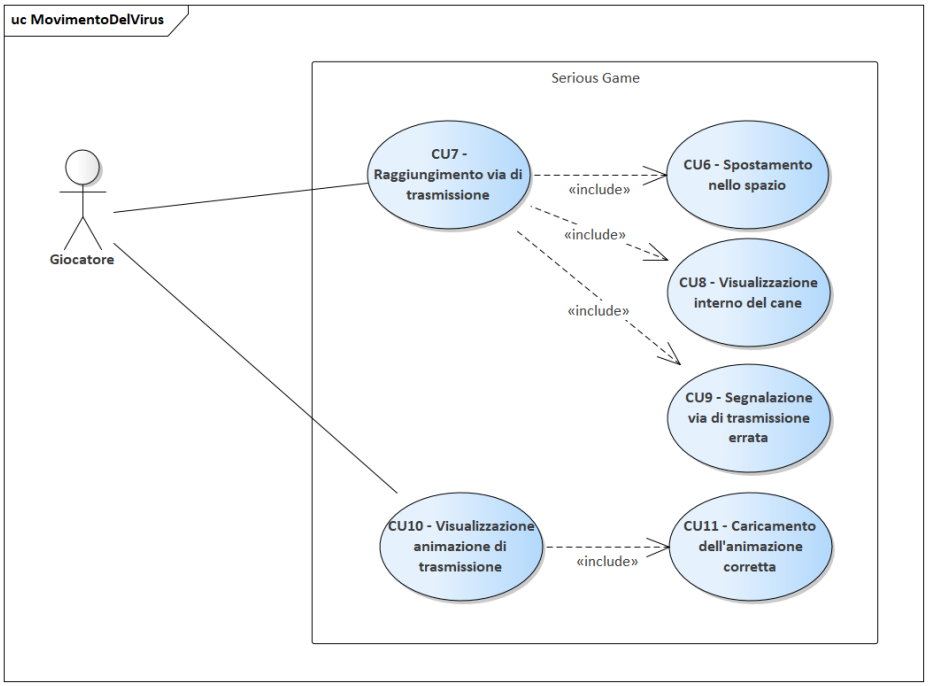


Figura 3.5: Diagramma dei casi d'uso relativi al movimento del virus nella scena di gioco

Visualizzazione animazione di trasmissione

Questo caso d'uso si verifica quando il giocatore raggiunge la via di trasmissione corretta. Le informazioni relative sono mostrate nella Tabella 3.32.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato un virus e raggiunto la via di trasmissione corretta
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore raggiunge la via di trasmissione corretta 2. Il sistema recupera la timeline da visualizzare 3. Il sistema fa partire la timeline corretta 4. Il sistema consente al giocatore di mettere in pausa e riprendere la visione dell'animazione, e consente anche di spostarsi avanti e/o indietro nella timeline 5. Il sistema una volta terminata l'animazione carica la schermata successiva
Sequenza degli eventi alternativa	Nessuna

Tabella 3.32: Caso d'uso relativo alla visualizzazione dell'animazione relativa alla trasmissione del virus

Caricamento dell'animazione corretta

Questo caso d'uso si verifica quando il giocatore raggiunge la via di trasmissione corretta. Le informazioni relative sono mostrate nella Tabella 3.33.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato un virus e raggiunto la via di trasmissione corretta
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore raggiunge la via di trasmissione corretta 2. Il sistema recupera la timeline relativa al virus selezionato dal giocatore
Sequenza degli eventi alternativa	Nessuna

Tabella 3.33: Caso d'uso relativo al caricamento dell'animazione relativa alla trasmissione, corretta

3.5.3 Selezione delle lesioni

Questa sottosezione ha lo scopo di illustrare i casi d'uso riguardanti la selezione delle lesioni, come mostrato in Figura 3.6. Questo caso d'uso si ripete due volte, una per le macro-lesioni ed una per le micro-lesioni.

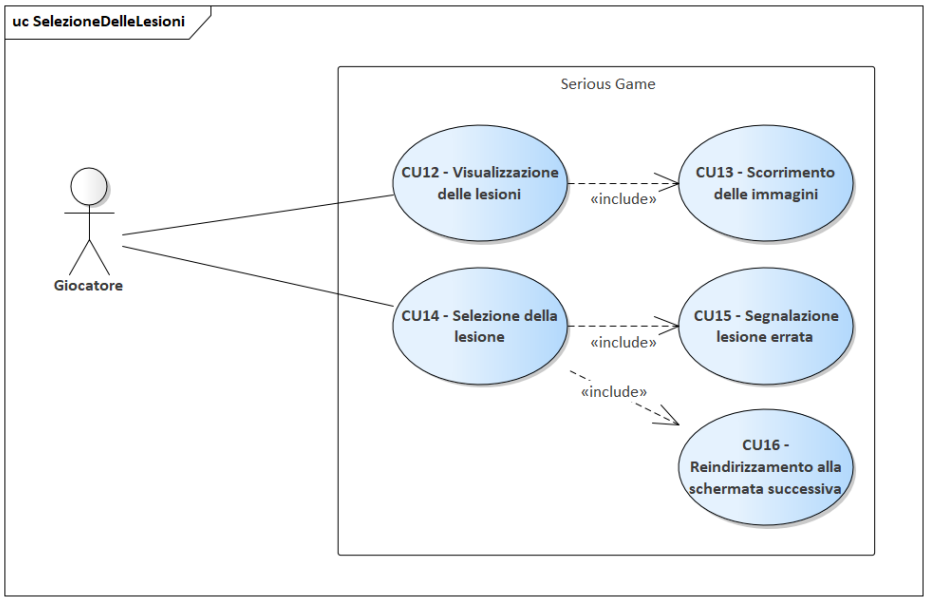


Figura 3.6: Diagramma dei casi d'uso relativi alla selezione delle lesioni

Visualizzazione delle lesioni

Questo caso d'uso si verifica in seguito alla visualizzazione dell'animazione sulla trasmissione del virus. Le informazioni relative sono mostrate nella Tabella 3.34.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver raggiunto la via di trasmissione corretta
Post-Condizioni	Nessuna

Elementi	Descrizione
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore ha terminato la visione dell'animazione di trasmissione 2. Il sistema recupera le immagini relative alle lesioni 3. Il sistema mostra le immagini relative alle lesioni
Sequenza degli eventi alternativa	Nessuna

Tabella 3.34: Caso d'uso relativo alla visualizzazione delle lesioni

Scorrimento delle immagini

Questo caso d'uso si verifica quando il giocatore si trova nella schermata relativa alle lesioni e clicca sulle frecce per scorrere le immagini. Le informazioni relative sono mostrate nella Tabella 3.35.

Elementi	Descrizione
Pre-Condizioni	Il giocatore si deve trovare nella schermata di selezione della lesione
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore clicca le frecce per spostarsi tra le immagini 2. Il sistema recupera le immagini
Sequenza degli eventi alternativa	Nessuna

Tabella 3.35: Caso d'uso relativo allo scorrimento delle immagini presenti

Selezione della lesione

Questo caso d'uso si verifica quando il giocatore decide qual è la lesione corretta tra quelle presenti. Le informazioni relative sono mostrate nella Tabella 3.36.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato una lesione
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore seleziona una, o più, immagini tra le lesioni presenti 2. Il sistema controlla che la lesione (o le lesioni) selezionata sia quella corretta 3. Il sistema mostra le informazioni relative alla lesione 4. Il sistema mostra la schermata successiva

Elementi	Descrizione
Sequenza degli eventi alternativa	3. Il sistema mostra un messaggio di errore in quanto la lesione non è quella corretta 4. Il sistema rimane nella schermata attuale invitando il giocatore a riprovare

Tabella 3.36: Caso d'uso relativo alla selezione della lesione

Segnalazione lesione errata

Questo caso d'uso si verifica quando il giocatore seleziona la lesione errata. Le informazioni relative sono mostrate nella Tabella 3.37.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato la lesione errata
Post-Condizioni	Il giocatore rimane nella schermata corrente
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore seleziona la lesione errata 2. Il sistema mostra un messaggio di errore 3. Il sistema continua a mostrare la schermata corrente (fino a che il giocatore non seleziona la lesione corretta)
Sequenza degli eventi alternativa	Nessuna

Tabella 3.37: Caso d'uso relativo alla visualizzazione del messaggio di errore nel caso in cui il giocatore selezioni la lesione sbagliata

Reindirizzamento alla schermata successiva

Questo caso d'uso si verifica quando il giocatore seleziona la lesione corretta. Le informazioni relative sono mostrate nella Tabella 3.38.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato la lesione corretta
Post-Condizioni	Il giocatore visualizza la schermata successiva
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore seleziona la lesione corretta 2. Il sistema mostra la schermata successiva
Sequenza degli eventi alternativa	Nessuna

Tabella 3.38: Caso d'uso relativo allo spostamento alla schermata successiva

3.5.4 Selezione dei sintomi

Questa sottosezione ha lo scopo di illustrare i casi d'uso riguardanti la selezione dei sintomi, come mostrato in Figura 3.7.

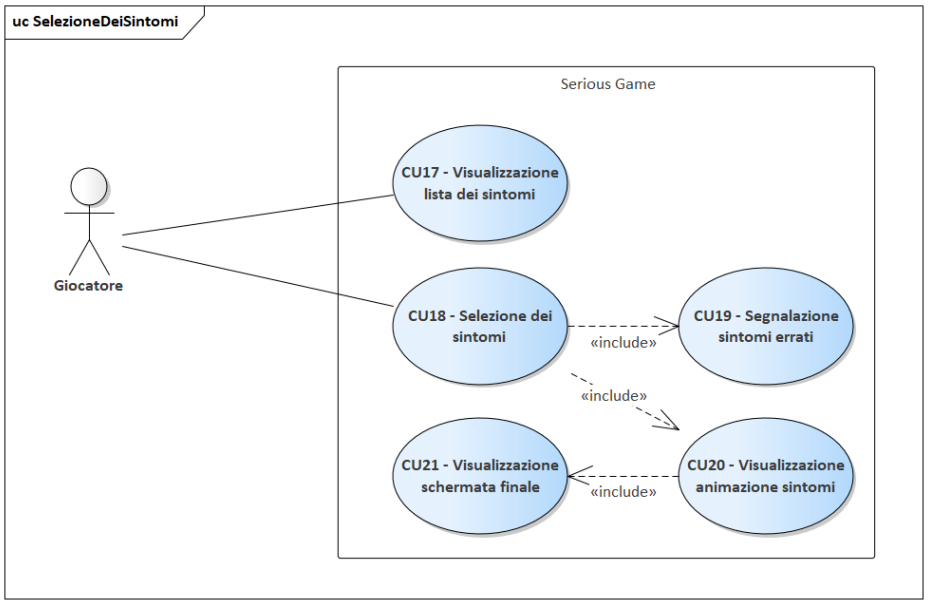


Figura 3.7: Diagramma dei casi d'uso relativi alla selezione delle lesioni

Visualizzazione lista dei sintomi

Questo caso d'uso si verifica in seguito alla selezione delle lesioni corrette. Le informazioni relative sono mostrate nella Tabella 3.39.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato le lesioni corrette
Post-Condizioni	Nessuna
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore ha selezionato le lesioni corrette 2. Il sistema mostra la lista di tutti i possibili sintomi
Sequenza degli eventi alternativa	Nessuna

Tabella 3.39: Caso d'uso relativo alla visualizzazione dei sintomi

Selezione dei sintomi

Questo caso d'uso si verifica quando il giocatore decide quali tra i sintomi presenti nella lista sono quelli corretti. Le informazioni relative sono mostrate nella Tabella 3.40.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato uno o più sintomi
Post-Condizioni	Nessuna

Elementi	Descrizione
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore ha selezionato uno o più sintomi 2. Il sistema controlla che i sintomi selezionati siano quelli corretti 3. Il sistema mostra le informazioni relative ai sintomi 4. Il sistema mostra un'animazione dei sintomi
Sequenza degli eventi alternativa	3. Il sistema mostra un messaggio di errore in quanto i sintomi non sono quelli corretti 4. Il sistema rimane nella schermata attuale invitando il giocatore a riprovare

Tabella 3.40: Caso d'uso relativo alla selezione dei sintomi

Segnalazione sintomi errati

Questo caso d'uso si verifica quando il giocatore seleziona i sintomi errati. Le informazioni relative sono mostrate nella Tabella 3.41.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato i sintomi errati
Post-Condizioni	Il giocatore rimane nella schermata corrente
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore seleziona i sintomi errati 2. Il sistema mostra un messaggio di errore 3. Il sistema continua a mostrare la schermata corrente (fino a che il giocatore non seleziona i sintomi corretti)
Sequenza degli eventi alternativa	Nessuna

Tabella 3.41: Caso d'uso relativo alla visualizzazione del messaggio di errore nel caso in cui il giocatore selezioni i sintomi sbagliati

Visualizzazione animazione sintomi

Questo caso d'uso si verifica quando il giocatore seleziona i sintomi corretti. Le informazioni relative sono mostrate nella Tabella 3.42.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato i sintomi corretti
Post-Condizioni	Il giocatore visualizza l'animazione relativa ai sintomi
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore seleziona i sintomi corretti 2. Il sistema mostra l'animazione relativa ai sintomi
Sequenza degli eventi alternativa	Nessuna

Tabella 3.42: Caso d'uso relativo alla visualizzazione dell'aniamzione relativa ai sintomi

Visualizzazione schermata finale

Questo caso d'uso si verifica quando il giocatore seleziona i sintomi corretti e termina la visione dell'animazione. Le informazioni relative sono mostrate nella Tabella 3.43.

Elementi	Descrizione
Pre-Condizioni	Il giocatore deve aver selezionato i sintomi corretti e terminato la visione dell'animazione
Post-Condizioni	Il giocatore visualizza la schermata finale
Sequenza degli eventi principali	1. Il caso d'uso inizia quando il giocatore termina la visione dell'animazione relativa ai sintomi 2. Il sistema mostra la schermata finale
Sequenza degli eventi alternativa	Nessuna

Tabella 3.43: Caso d'uso relativo alla visualizzazione della schermata finale

3.6 Diagrammi di sequenza

In questa sezione verranno utilizzati i diagrammi di sequenza per modellare le interazioni tra gli attori e le componenti del nostro sistema e tra le componenti stesse.

I seguenti diagrammi mostreranno la sequenza delle interazioni che vengono svolte in ogni caso d'uso.

3.6.1 Selezione del virus

Questa sottosezione riguarda la sequenza degli eventi relativi alla scena di selezione del virus; in essi è possibile visualizzare il virus e le informazioni ad esso relative nonché scegliere il virus con il quale giocare. Il diagramma è mostrato nella Figura 3.8.

3.6.2 Movimento del virus

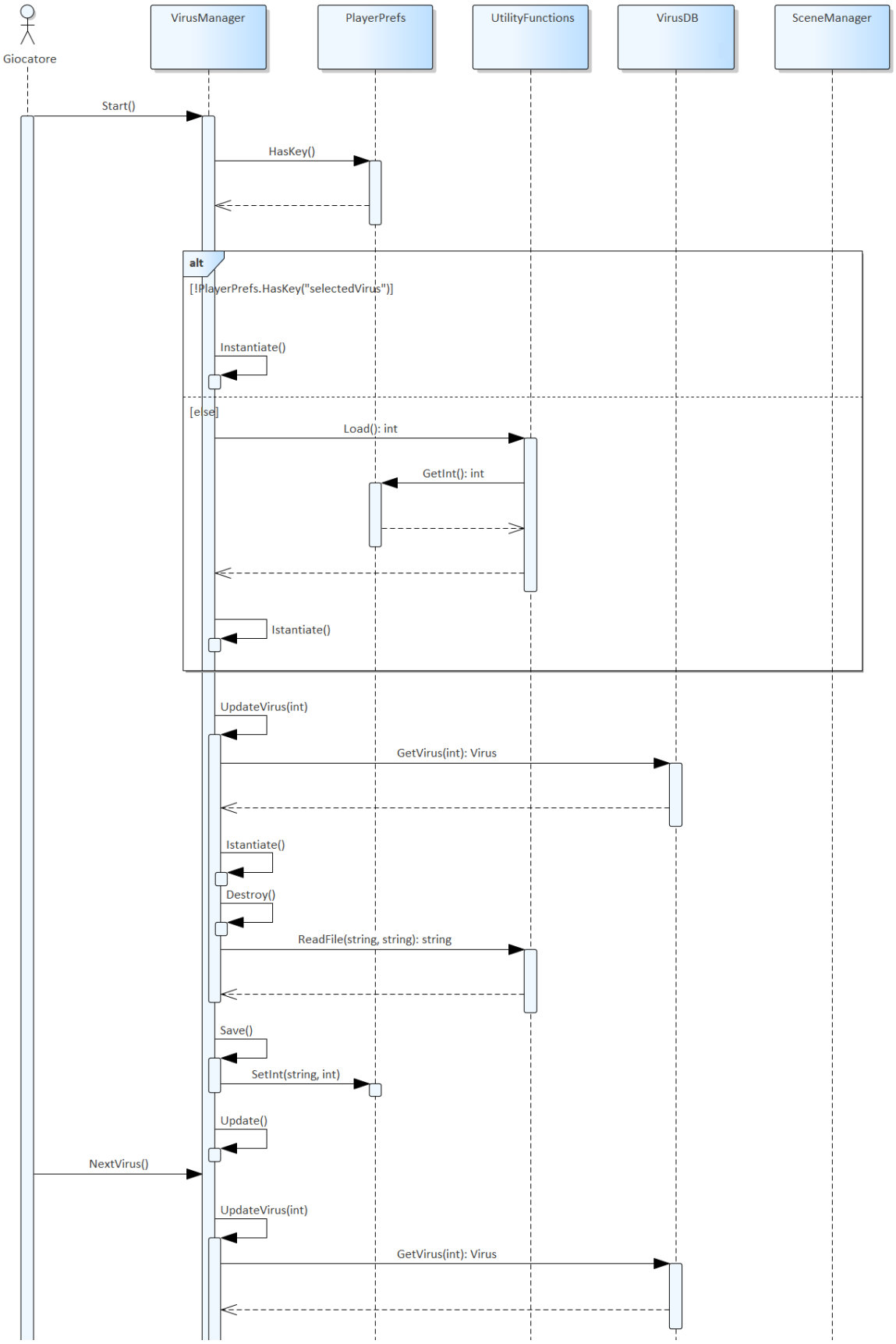
Questa sottosezione riguarda la sequenza degli eventi relativi allo spostamento del virus; in essi è possibile spostare il virus per raggiungere la via di trasmissione corretta. Il diagramma è mostrato nella Figura 3.9.

3.6.3 Selezione delle lesioni

Questa sottosezione riguarda la sequenza degli eventi relativi alla scena di selezione delle lesioni associate al virus scelto. In questa scena è possibile scorrere tra le immagini presenti e visualizzarle, scegliere la lesione che si ritiene corretta e leggere una spiegazione di tale lesione. Il diagramma è mostrato nella Figura 3.10.

3.6.4 Selezione dei sintomi

Questa sottosezione riguarda la sequenza degli eventi relativi alla scena di selezione dei sintomi associati al virus scelto. In questa scena è possibile selezionare e deselezionare uno o più sintomi tra quelli presenti nella lista, e quindi sceglierli. A seguito della selezione viene mostrata un animazione dei sintomi che il cane riporta. Il diagramma è mostrato nella Figura 3.11.


Giocatore
VirusManager
PlayerPrefs
UtilityFunctions
VirusDB
SceneManager
Start()
HasKey()
alt
[!PlayerPrefs.HasKey("selectedVirus")]
Instantiate()
[else]
Load(): int
GetInt(): int
Istantiate()
UpdateVirus(int)
GetVirus(int): Virus
Istantiate()
Destroy()
ReadFile(string, string): string
Save()
SetInt(string, int)
Update()
NextVirus()
UpdateVirus(int)
GetVirus(int): Virus

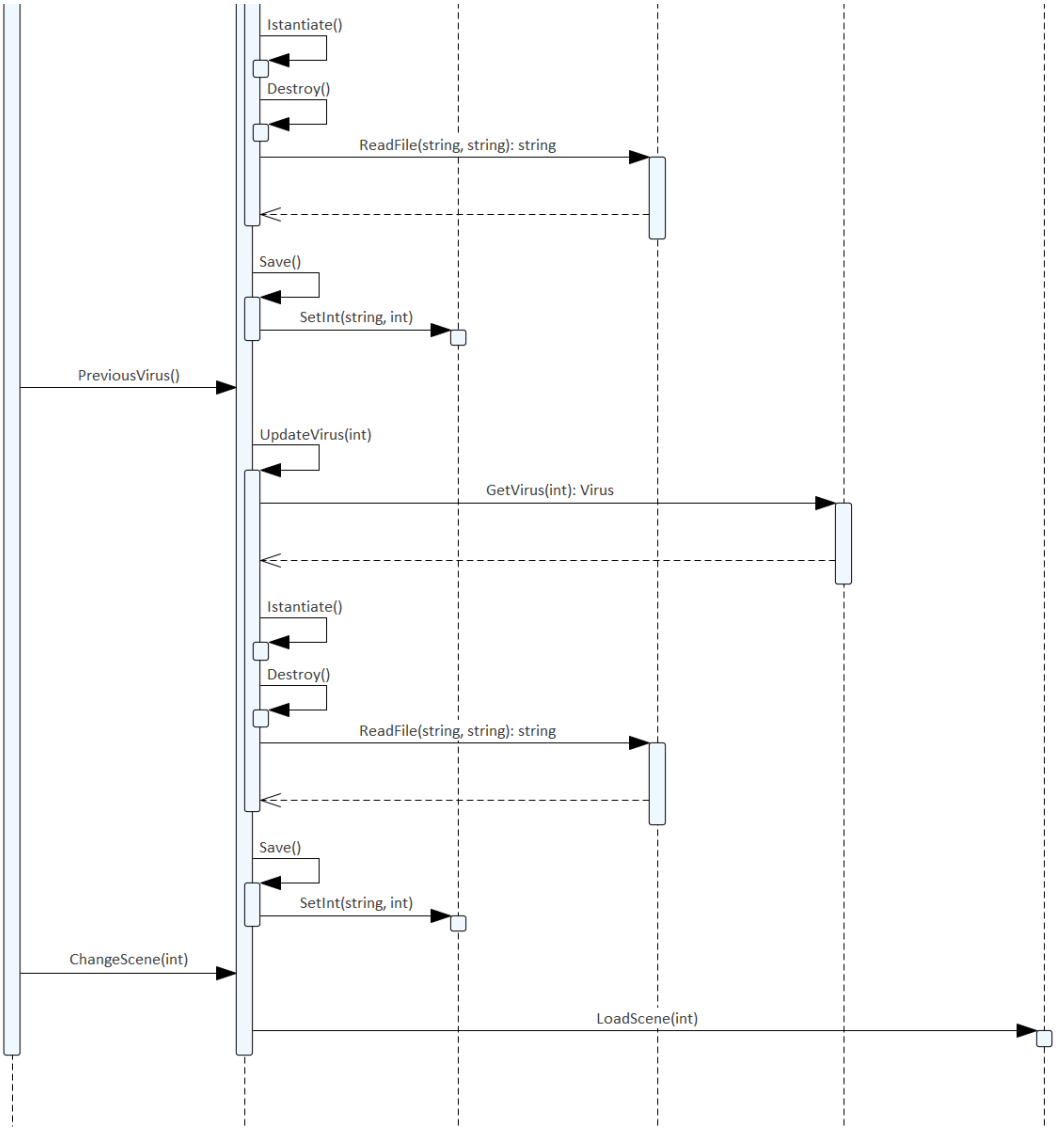


Figura 3.8: Diagramma di sequenza relativo alla selezione del virus

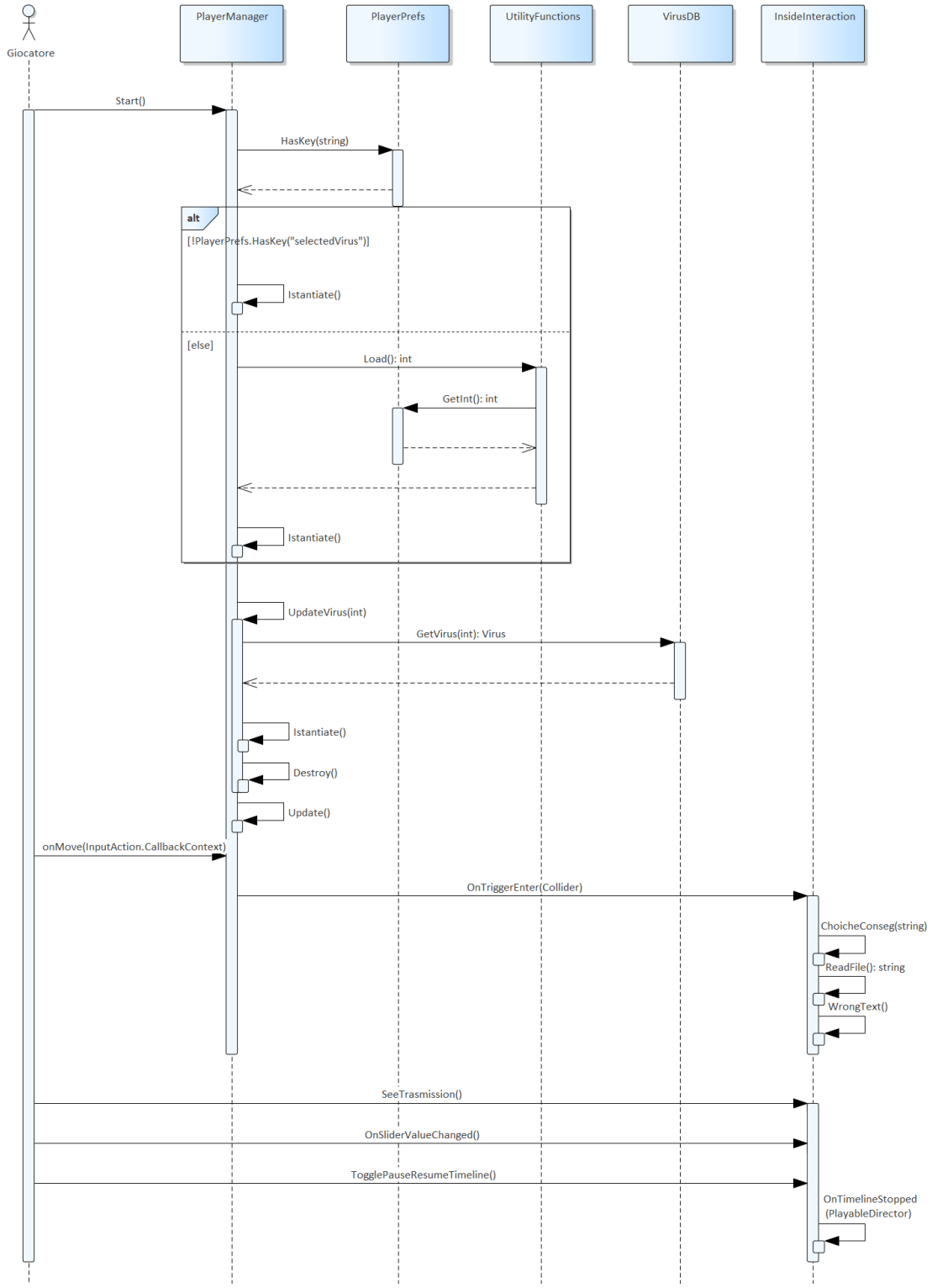


Figura 3.9: Diagramma di sequenza relativo allo spostamento del virus

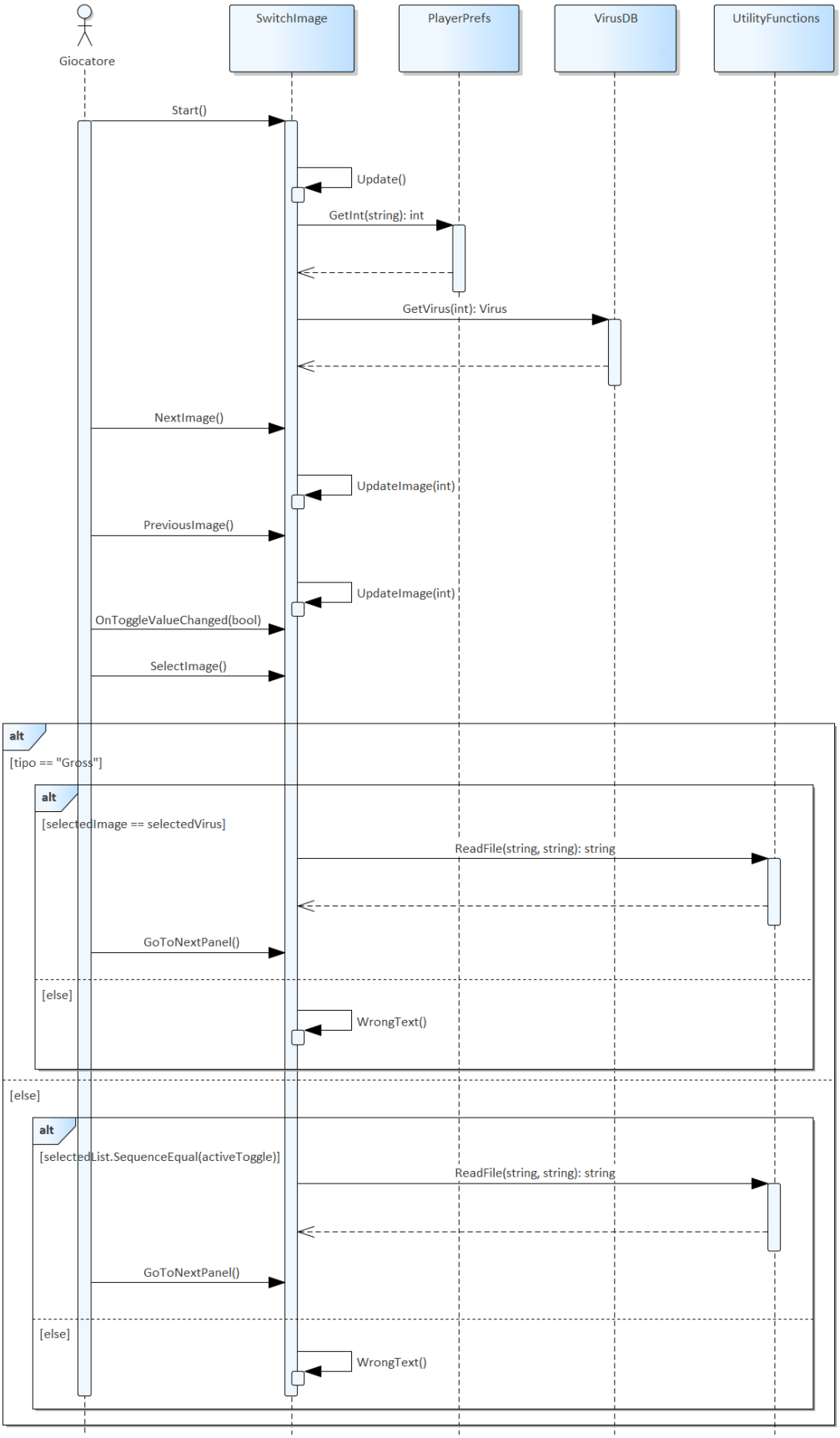


Figura 3.10: Diagramma di sequenza relativo alla selezione delle lesioni

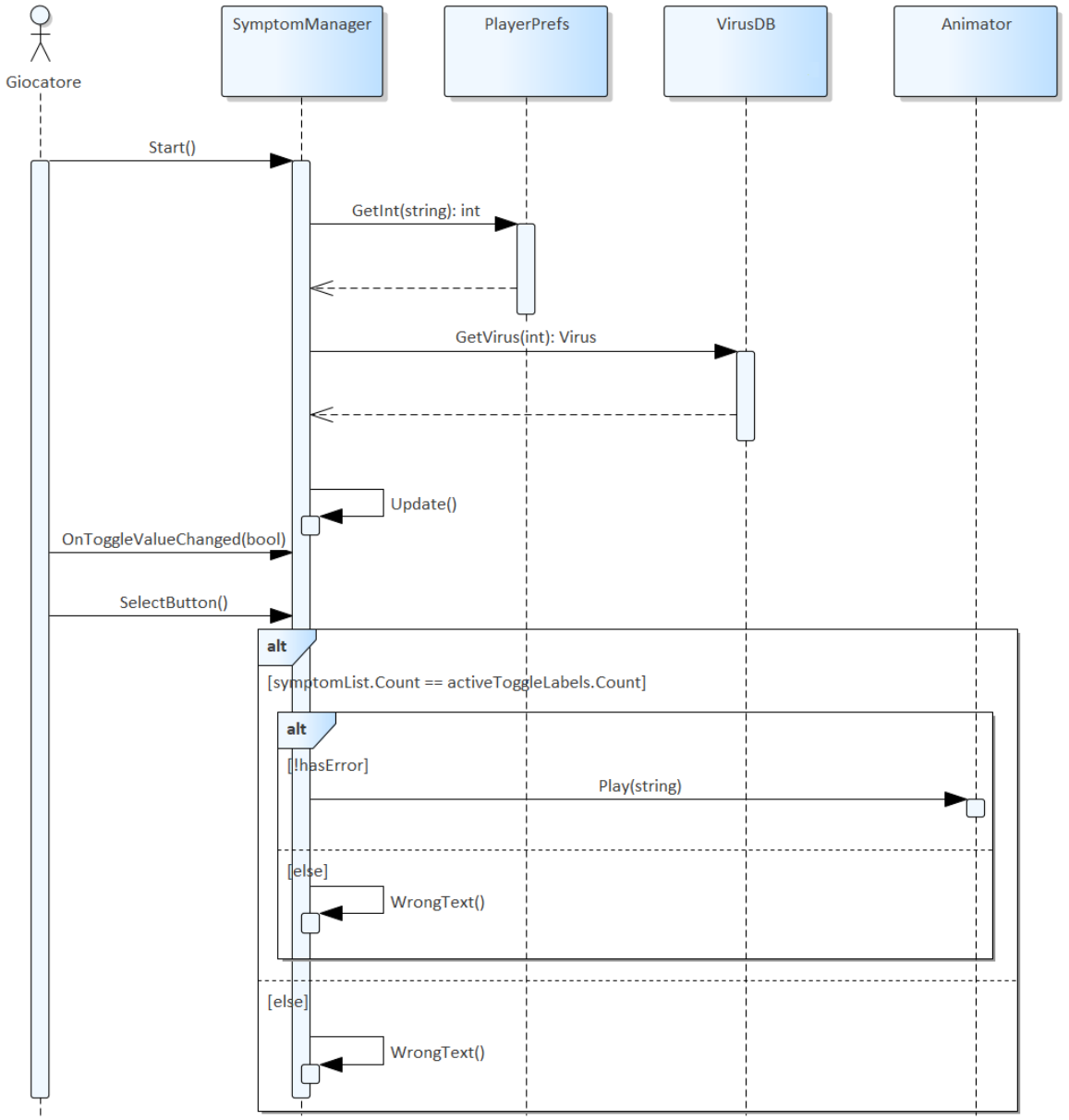


Figura 3.11: Diagramma di sequenza relativo alla selezione dei sintomi

CAPITOLO 4

Implementazione

In questo capitolo verrà trattata l'implementazione del serious game oggetto di questa tesi.

4.1 Practical Design

In questa sezione verranno elencati e spiegati i tool utilizzati nello sviluppo di questo serious game nonché il linguaggio utilizzato. In particolare verranno illustrati:

- Unity
- Blender
- Visual Studio Code
- C#
- Adobe Photoshop

4.1.1 Unity

Unity è una piattaforma di sviluppo software, composta da un game engine, un IDE e una service suite, che permette la creazione di videogiochi in 2D e 3D, di applicazioni interattive multipiattaforma e di esperienze di realtà virtuale.

Il robusto motore fisico, insieme al supporto per le dinamiche del corpo rigido, al rilevamento delle collisioni e al raycasting, consentono la creazione di ambienti realistici e coinvolgenti oltre alla possibilità di creare oggetti interattivi che rispondono in modo realistico alle azioni del giocatore.

L'architettura di Unity si basa sui componenti e ciò consente agli sviluppatori di modificare facilmente le funzionalità presenti nei loro videogiochi, semplicemente aggiungendo o rimuovendo componenti vari (come, ad esempio, un controller di personaggi o uno script) ai GameObject presenti nel videogioco.

Per quanto riguarda l'ambito relativo alla grafica, Unity supporta una vasta gamma di funzionalità, grazie alle quali è possibile creare mondi di gioco altamente realistici. Queste funzionalità sono:

- luci;

- ombre;
- riflessi;
- effetti di post-elaborazione.

Infine, Unity supporta più piattaforme, tra cui Windows, Mac, iOS, Android e altre ancora, e ciò consente agli sviluppatori di distribuire i loro giochi su più piattaforme con uno sforzo aggiuntivo minimo.

4.1.2 Blender

Blender è un software open-source e multipiattaforma di modellazione, rigging, animazione, montaggio video, composizione, rendering e texturing di immagini tridimensionali e bidimensionali.

Questo strumento offre una vasta gamma di funzionalità e una comunità attiva di utenti e sviluppatori.

Blender può essere utilizzato per:

- *Modellazione 3D*: Blender consente di creare modelli complessi e dettagliati di oggetti, personaggi e ambienti.
- *Animazione*: è possibile animare i modelli 3D per creare movimenti fluidi e realistici. Blender offre una vasta gamma di strumenti e opzioni per l'animazione.
- *Rendering*: il motore di rendering di Blender permette di creare immagini fotorealistiche dei modelli 3D. È possibile gestire l'illuminazione, gli shader e ottenere risultati di alta qualità.
- *Simulazione*: Blender offre strumenti per simulare fenomeni fisici, come fluidi, particelle, tessuti e simulazioni di fisica avanzata.
- *Compositing e post-produzione*: Blender dispone di un potente nodo di composizione che permette di combinare diversi passaggi di rendering, applicare effetti speciali, correggere il colore e fare editing video.
- *VFX (Effetti visivi)*: Blender è utilizzato nell'industria cinematografica per la creazione di effetti visivi complessi e realistici, come esplosioni, fuoco, simulazione di massa e molto altro ancora.
- *Game development*: Blender supporta la creazione di giochi 3D, fornendo strumenti di modellazione, animazione e rendering specifici per il game development.

4.1.3 Visual Studio Code

Visual Studio Code (o, più semplicemente, VS Code) è un editor di codice sorgente, open-source, sviluppato da Microsoft per Windows, Linux e macOS. Esso include il supporto per debugging, un controllo per Git integrato, il syntax highlighting, IntelliSense, gli snippet e il refactoring del codice.

Unity possiede un supporto integrato per utilizzare Visual Studio Code come editor esterno predefinito per gli script, sia in Windows che in macOS. Inoltre, questo tool permette di installare estensioni che aggiungono ulteriori funzionalità.

4.1.4 C#

C# è un linguaggio di programmazione a oggetti sviluppato e mantenuto da Microsoft, molto simile a C++ e Java. Si tratta di uno dei linguaggi più potenti per manipolare gli oggetti del framework .NET.

4.1.5 Photoshop

Adobe Photoshop è un software proprietario prodotto da Adobe specializzato nell'elaborazione di fotografie (fotoritocco) e, più in generale, di immagini digitali.

Questo programma è in grado di effettuare ritocchi di qualità professionale alle immagini, offrendo enormi possibilità creative grazie ai numerosi filtri e strumenti. Un'importante funzione del programma è data dalla possibilità di lavorare con più livelli, permettendo di gestire separatamente le diverse componenti che costituiscono l'immagine. Nel nostro progetto questo software si è reso necessario per modificare le immagini delle texture, come viene illustrato nel seguito di questo capitolo.

4.2 Modellazione

Al fine di realizzare il nostro serious game si è resa necessaria una fase di modellazione durante la quale:

- sono stati *creati* alcuni *modelli 3D*, i quali non erano disponibili su internet;
- sono state *modificate* alcune *texture*;
- sono state *create le animazioni* necessarie.

4.2.1 Creazione dei modelli 3D

In questa fase si è proceduto a creare i modelli 3D che non erano disponibili sul web (ad esempio l'anatomia del cane).

Anatomia del cane

Per la modellazione dell'anatomia del cane si è partiti da un modello di una sezione di scheletro già esistente, la quale è stata duplicata e specchiata al fine di avere lo scheletro intero; al modello così ottenuto si è applicato uno sharder che lo rendesse in parte trasparente in maniera tale che fossero ben visibili gli organi.

Successivamente sono stati modellati i vari organi, adottando un approccio basato su ricerche preliminari, volte a raccogliere immagini di riferimento e ispirazioni visive, al fine di rendere il modello il più accurato possibile.

In questa fase, per alcuni organi, sono state create le mesh ex novo; per altri, invece, si è proceduto adattando mesh già esistenti.

Dopo aver perfezionato la forma e la struttura dell'anatomia, è stata dedicata una fase all'applicazione di materiali e texture per rendere il modello finale visivamente accattivante e realistico. L'uso di texture procedurali e mappe di dettaglio ha contribuito a simulare i tessuti dei vari organi.

L'elaborazione del modello con Blender è visibile in Figura 4.1.

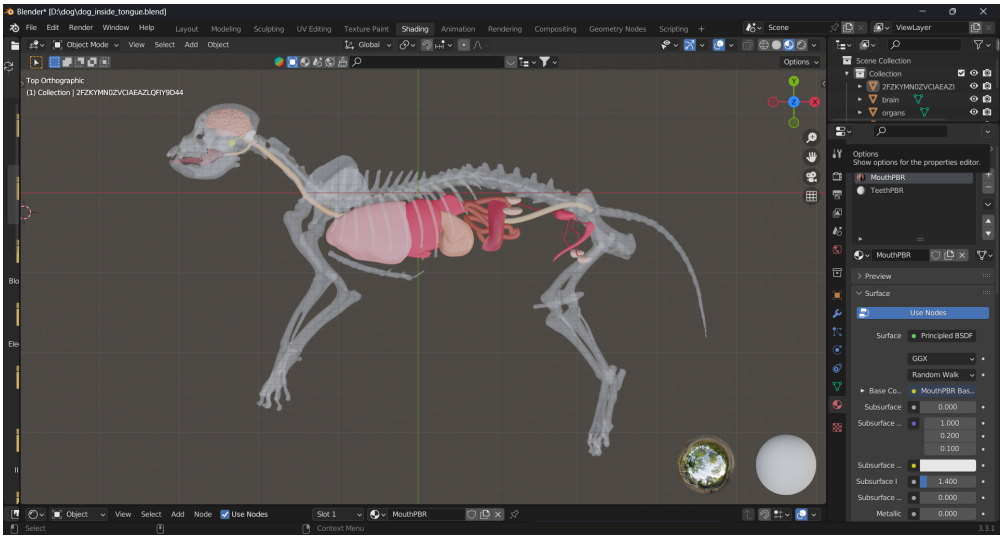


Figura 4.1: Modello dell'anatomia del cane

Clinica veterinaria

Per la modellazione della clinica veterinaria sono state selezionate le mesh di vari componenti di arredo, che sono stati poi posizionati nella stanza manualmente cercando, appunto, di ricreare un ambiente simile a quello di una clinica veterinaria.

4.2.2 Modifica delle texture

Mantello del cane

La texture del mantello del cane è presente in due versioni (Figura 4.2) nel nostro serious game, ovvero:

1. *la versione di base*, che è quella scaricata con il modello;
2. *la versione modificata*, che è quella che presenta il morso di un altro animale sulla zampa anteriore destra.

Figura 4.2: Texture di base (a), immagine del morso di animale (b), e risultato della sovrapposizione delle due, ovvero texture modificata (c)

La seconda versione è necessaria in quanto la via di trasmissione della rabbia è, appunto, una ferita da morso inferta da un altro animale infetto. Per ottenere questa seconda texture sono state compiute operazioni di fusione tra la texture di base e un'immagine di un morso

animale su Photoshop, al fine di ottenere una transizione fluida e armoniosa tra l'immagine superiore e quella inferiore.

Clinica veterinaria

Si è deciso di modificare il colore di alcune texture dei mobili della clinica veterinaria al fine di avere una colorazione omogenea per tutto lo studio.

4.2.3 Creazione delle animazioni

Volendo al termine del gioco mostrare in maniera visiva come il cane si comporta in presenza della malattia legata al virus selezionato, è stato necessario modellare le animazioni relative a tali comportamenti. Al fine di creare le nostre animazioni sono stati seguiti i seguenti step:

1. Mediante l'utilizzo dell'add-ond *Auto-Rig Pro*, si è effettuato il rigging del modello del cane. Il processo di rigging implica la realizzazione di una struttura scheletrica virtuale e l'attribuzione di pesi di verniciatura ai vertici della mesh.

2. Sono state create le pose chiave (keyframe); queste pose rappresentano i momenti chiave dell'animazione. A seguito della creazione delle pose chiave, Blender genererà automaticamente gli stati intermedi (conosciuti come "inbetweening" o "interpolazione") tra di esse al fine di rendere l'animazione fluida.

3. Infine, sono stati regolati i tempi e i ritmi dell'animazione, modificando il timing delle pose chiave e delle transizioni per ottenere l'effetto desiderato.

Nella Figura 4.3 è possibile vedere l'implementazione delle animazioni in Blender.

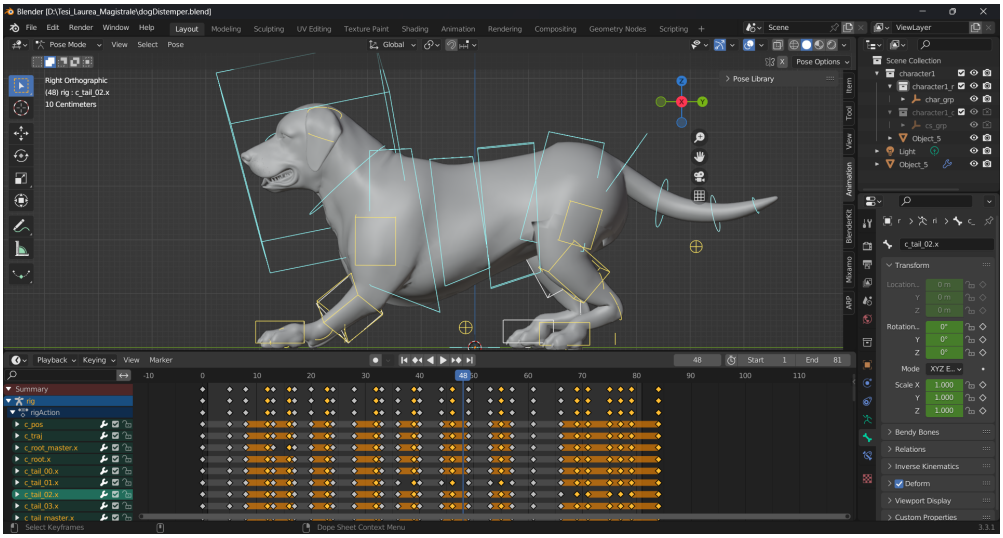


Figura 4.3: Creazione di un animazione in Blender

4.3 Implementazione delle funzionalità principali

Nella fase di sviluppo del nostro serious game, l'implementazione delle funzionalità gioca un ruolo fondamentale nel conferire interattività e dinamicità all'esperienza dell'utente.

Per poter implementare nuove funzionalità in Unity è necessario scrivere degli script in C# che controllino il comportamento degli oggetti nel gioco o gestiscano altre operazioni.

Dopo aver realizzato lo script, esso va collegato all'oggetto a cui si desidera aggiungere la funzionalità.

Per lo sviluppo del nostro serious game le funzionalità che sono state implementate sono contenute nei seguenti script:

- InsideInteraction;
- PlayerManager;
- SubtitleClip;
- SubtitleTrack;
- SubtitleTrackMixer;
- SwitchImage;
- SymptomManager;
- UtilityFunctions;
- Virus;
- VirusDB;
- VirusManager.

Nel seguito esamineremo in dettaglio ciascuna di queste funzionalità.

4.3.1 InsideInteraction

Nello script `InsideInteraction`, che è legato all'oggetto corrispondente al virus, vengono considerate tutte le funzionalità per gestire il raggiungimento, da parte del virus, delle zone di trasmissione e per mostrare la timeline corretta nel caso in cui si raggiunga la via corretta.

La funzione `ChoiceConseg` (Listato 4.1) controlla se la via di trasmissione raggiunta dal giocatore è quella corretta; nel caso in cui lo fosse, disattiva la visualizzazione del cane dall'esterno e attiva la visualizzazione interna, riempiendo la stringa di info con le informazioni sulla trasmissione del virus scelto. Nel caso in cui la via di trasmissione raggiunta non fosse quella corretta, `ChoiceConseg` richiama la funzione `WrongText` che rende visibile, per un certo lasso di tempo, un messaggio di errore, e riporta il virus alla sua posizione iniziale per ricominciare.

```
private void ChoiceConseg(string choice){
      if(playerManager.trasmission==choice){
         dogInside.SetActive(true);
         dog.SetActive(false);
         infoCanvas.SetActive(true);
         goButton.SetActive(true);
         string info=utility.ReadFile((utility.Load()+"Trasmission"));
         infoText.text=info;
      } else {
         WrongText();
         transform.position=new Vector3(-2f,2.95f,4.9f);
      }
   }
```

Listato 4.1: Funzione che gestisce il raggiungimento di una delle vie di trasmissione

La funzione `SeeTrasmission` (Listato 4.2), quando viene richiamata, fa partire la timeline corretta e nasconde le componenti di interfaccia in maniera tale che non interferiscano con la visualizzazione del video.

```
public void SeeTrasmission()
    {
        director.Play();
        goButton.SetActive(false);
        explaination.SetActive(true);
        infoCanvas.SetActive(false);
    }
```

Listato 4.2: Funzione che fa partire la timeline

La funzione `Awake` (Listato 4.3) viene richiamata quando la timeline inizia. Questa funzione sceglie la timeline corretta tra quelle presenti, e disabilità la possibilità di muovere il virus da parte del giocatore.

Inoltre, questa funzione collega lo slider presente in fondo alla schermata, alla timeline, facendo si che il valore massimo dello slider corrisponda alla durata della timeline. Lo slider fa si che il giocatore possa decidere di andare avanti e indietro, nella visualizzazione dell'animazione, utilizzando la funzione `OnSliderValueChanged` (Listato 4.4).

```
    private void Awake()
    {
        selectedVirus=PlayerPrefs.GetInt("selectedVirus");
        director=directors[selectedVirus];
        if (director!=null && slider!=null)
        {
            slider.minValue=0.0f;
            slider.maxValue=(float)director.duration;
            slider.value=(float)director.time;
        }
        director.stopped+=OnTimelineStopped;
        movementController=this.GetComponent<PlayerInput>();
    }
```

Listato 4.3: Funzione che viene richiamata quando la timeline comincia

```
public void OnSliderValueChanged()
    {
        if (director!=null && slider!=null)
        {
            director.time=slider.value;
            pausedTime=slider.value;
        }
    }
```

Listato 4.4: Funzione che viene richiamata quando cambia il valore dello slider

La funzione `TogglePauseResumeTimeline` (Listato 4.5) viene richiamata per mettere la timeline in pausa e per riprendere la visione di essa.

```
    public void TogglePauseResumeTimeline()
        {
            if (director!=null)
            {
                if (director.playableGraph.GetRootPlayable(0).GetSpeed()==1.0)
                {
                    director.playableGraph.GetRootPlayable(0).SetSpeed(0.0);
                    pausedTime=(float)director.time;
                    playPauseButton.GetComponent<Image>().sprite=playSprite;
                }
                else if (director.playableGraph.GetRootPlayable(0).GetSpeed()==0.0)
                {
                    director.playableGraph.GetRootPlayable(0).SetSpeed(1.0);
                    director.time=pausedTime;
                    playPauseButton.GetComponent<Image>().sprite=pauseSprite;
```

```
            }
        }
    }
```

Listato 4.5: Funzione che viene richiamata quando si preme il pulsante di pausa o di avvio

La funzione `OnTimelineStopped` (Listato 4.6) viene richiamata quando la timeline termina. Questa funzione disattiva il controllo del movimento da parte del giocatore, disattiva tutte le parti di interfaccia relative alla timeline e attiva la visualizzazione del pannello dove si deve effettuare la scelta tra le varie immagini delle lesioni.

```
private void OnTimelineStopped(PlayableDirector director)
    {
        movementController.DeactivateInput();
        explaination.SetActive(false);
        infoCanvas.SetActive(false);
        goButton.SetActive(false);
        panelGross.SetActive(true);
    }
```

Listato 4.6: Funzione che viene richiamata quando la timeline finisce

4.3.2 PlayerManager

Nello script `PlayerManager`, si utilizza la funzione `Update()` (Listato 4.7) per controllare il movimento del virus nella scena di gioco. La chiave per questo controllo è l'uso di un raggio, rappresentato da Ray, che parte dalla posizione dell'oggetto (`transform.position`) e si estende nella direzione specificata (`direction`).

Questa funzione verifica in ogni frame se il raggio, sopra descritto, interseca con un oggetto nel mondo di gioco utilizzando la funzione `Physics.Raycast()`. Questa funzione restituisce `true` se il raggio colpisce un oggetto, e `false` se non lo fa. Se il raggio non colpisce nulla, il blocco di codice all'interno dell'istruzione `if` viene eseguito, e quindi il virus viene spostato lungo la direzione specificata (`direction`) con una velocità determinata da `speed`. Il tempo tra i fotogrammi, misurato da `Time.deltaTime`, viene utilizzato per garantire un movimento fluido e uniforme. Nel caso in cui, invece, il raggio colpisca un oggetto, il virus sarà bloccato e non potrà, quindi, muoversi ulteriormente nella direzione data.

```
void Update()
    {
      RaycastHit hit;
      Ray ray=new Ray(transform.position, direction);

      if (!Physics.Raycast(ray, out hit, speed*Time.deltaTime))
        {
         transform.Translate(direction*speed*Time.deltaTime);
        }
    }
```

Listato 4.7: Funzione che non permette al giocatore di muoversi al di fuori dei limiti della stanza

Mediante la funzione `UpdateVirus` (Listato 4.8) quando la nuova scena, ovvero quella di gioco, viene caricata, viene istanziato, ridimensionato e posizionato il modello del virus scelto nella precedente scena di selezione.

```
private void UpdateVirus(int selectedVirus)
    {
        Virus virus=virusDB.GetVirus(selectedVirus);
        trasmission=virus.trasmissionWay;
        GameObject thisModel=Instantiate(virusModels[selectedVirus],transform.position,
        transform.rotation) as GameObject;
        Destroy(virusPlayer);
        thisModel.transform.parent=transform;
```

```
        thisModel.transform.localScale=new Vector3(1,1,1);
        virusPlayer=thisModel;
    }
```

Listato 4.8: Funzione che carica il virus scelto

4.3.3 SubtitleClip

Questa funzione è parte di un asset personalizzato (`Custom Playable Asset`) in Unity, progettato per gestire sottotitoli o testi di dialogo all'interno di una timeline di gioco. L'obiettivo di questa funzione è creare un oggetto "playable" che sarà utilizzato all'interno della timeline per controllare l'apparizione dei sottotitoli in momenti specifici del gioco (Listato 4.9).

```
public override Playable CreatePlayable(PlayableGraph graph,GameObject owner) {
        var playable=ScriptPlayable<SubtitleBehavior>.Create(graph);
        SubtitleBehavior subtitleBehavior=playable.GetBehaviour();
        subtitleBehavior.subtitleText=subtitleText;
        return playable;
    }
```

Listato 4.9: Funzione per la gestione delle clip dei sottotitoli

4.3.4 SubtitleTrack

La funzione `CreateTrackMixer` fa parte di un asset personalizzato (`Custom Playable Asset`) in Unity ed è utilizzata per creare un "mixer di traccia" (track mixer) associato a una traccia di sottotitoli all'interno di una timeline di gioco. Questo mixer di traccia è responsabile di combinare o mischiare i dati provenienti da più tracce di sottotitoli per produrre un output finale coerente durante la riproduzione della timeline nel gioco (Listato 4.10).

```
public override Playable CreateTrackMixer(PlayableGraph graph, GameObject go,
    int inputCount) {
        return ScriptPlayable<SubtitleTrackMixer>.Create(graph, inputCount);
    }
```

Listato 4.10: Funzione per la gestione delle proprietà delle clip dei sottotitoli

4.3.5 SubtitleTrackMixer

La funzione `ProcessFrame` (Listato 4.11) fa parte di un mixer di traccia personalizzato in Unity ed è responsabile di combinare e gestire le informazioni provenienti da diverse tracce di sottotitoli per produrre un output finale coerente durante la riproduzione della timeline. Questo codice viene eseguito in ogni fotogramma del gioco mentre la timeline è in esecuzione.

```
public override void ProcessFrame(Playable playable, FrameData info, object playerData)
    {
        text=playerData as TextMeshProUGUI;
        if (!text)
            return;

        int inputCount=playable.GetInputCount();
        for (int i=0; i<inputCount; i++)
        {
            float inputWeight=playable.GetInputWeight(i);
            if (inputWeight>0f)
            {
                ScriptPlayable<SubtitleBehavior>inputPlayable=
                (ScriptPlayable<SubtitleBehavior>)playable.GetInput(i);
```

```
            SubtitleBehavior input=inputPlayable.GetBehaviour();
            targetText=input.subtitleText;
            targetAlpha=inputWeight;
        }
    }
    text.text="<font=LiberationSans SDF><mark=121345>"+targetText+"</mark>";
    text.color=new Color(1,1,1, targetAlpha);
}
```

Listato 4.11: Funzione per la gestione del miscelamento delle tracce dei sottotitoli

4.3.6 SwitchImage

Le funzioni `NextImage` e `PreviousImage` (Listato 4.12) permettono all'utente di scorrere tra le immagini presenti. Tali funzioni sono, rispettivamente, collegate ad una freccia "avanti" e ad una freccia "indietro" e vengono richiamate quando tali frecce sono cliccate. Entrambe le funzioni richiamano la funzione `UpdateImage` (Listato 4.13), la quale carica l'immagine da visualizzare e attiva il toggle button relativo all'immagine stessa.

```
public void NextImage()
    {
        selectedImage++;
        if (selectedImage>images.Length-1)
        {
            selectedImage=0;
        }
        UpdateImage(selectedImage);
    }

public void PreviousImage()
    {
        selectedImage--;
        if (selectedImage<0)
        {
            selectedImage=images.Length-1;
        }
        UpdateImage(selectedImage);
    }
```

Listato 4.12: Funzioni che permettono di scorrere tra le immagini

```
private void UpdateImage(int selectedImage) {
        imgDisplayed.texture=images[selectedImage];
        int i=0;
        foreach (Toggle toggleButton in toggleButtons)
        {
            if(i==selectedImage)
            {
                toggleButton.gameObject.SetActive(value: true);
            }
            else{
                toggleButton.gameObject.SetActive(false);
            }
            i++;
        }
    }
```

Listato 4.13: Funzione che mostra l'immagine corretta

La funzione `OnToggleValueChanged` (Listato 4.14) viene richiamata ogni volta che viene selezionato o deselezionato un toggle button. Nel caso in cui il pulsante viene selezionato, aggiorna la lista dei toggle button attivi mettendo `true` nell'indice ad esso corrispondente. Se, invece, il pulsante viene deselezionato, il valore nella lista viene posto a `false`.

```
private void OnToggleValueChanged(bool isOn)
    {
```

```
        Toggle toggle=UnityEngine.EventSystems.EventSystem.current
        .currentSelectedGameObject.GetComponent<Toggle>();
        if (isOn)
        {
          activeToggle[selectedImage]=true;
        }
        else
        {
           activeToggle[selectedImage]=false;
        }
        Debug.Log("Active Toggle: "+string.Join(", ", activeToggle.ToArray()));
    }
```

Listato 4.14: Funzione che gestisce i toggle button delle immagini

La funzione fondamentale di questo script è `SelectImage` (Listato 4.15), la quale gestisce la selezione delle immagini da selezionare viene effettuata in maniera differente se queste sono immagini di gross o immagini istologiche, quindi in base alla tipologia di immagine da selezionare gestisce in maniera differente la selezione, ma, l'operatività di fondo consiste nel verificare se le immagini selezionate sono quelle corrette, ed in caso affermativo procedere nella visualizzazione del pannello di scelta seguente. Se, invece, la scelta è quella sbagliata verrà visualizzato un messaggio di errore.

```
public void SelectImage(){
        if(tipo=="Gross")
        {
           if (selectedImage==selectedVirus){
             imgPanel.SetActive(false);
             buttonSelect.SetActive(false);
             numberOfImage.enabled=false;
             foreach (GameObject arr in arrows)
               {
                arr.SetActive(false);
               }
             string info=utility.ReadFile((utility.Load()+tipo));
             textPlaceholder.text=info;
             textPanel.SetActive(true);
             buttonGo.SetActive(true);
           }
           else
              {
                WrongText();
              }
        }
        else
        {
           selectedList=new List<bool>(virus.grossHistoVirus);
           for (int i=0; i <selectedList.Count; i++)
           {
              if (selectedList.SequenceEqual(activeToggle))
              {
                 imgPanel.SetActive(false);
                 buttonSelect.SetActive(false);
                 numberOfImage.enabled=false;
                 foreach (GameObject arr in arrows)
                    {
                      arr.SetActive(false);
                    }
                 foreach (Toggle toggleButton in toggleButtons)
                    {
                      toggleButton.gameObject.SetActive(false);
                    }
                 string info=utility.ReadFile((utility.Load()+tipo));
                 textPlaceholder.text=info;
                 textPanel.SetActive(true);
                 buttonGo.SetActive(true);
              }
              else
              {
```

```
                WrongText();
                Debug.Log(selectedList[i]+" è diverso da "+activeToggle[i]+" ed i = ");
            }
        }
    }
}
```

Listato 4.15: Funzione che gestisce la selezione delle immagini

Se si selezionano le immagini giuste appare una spiegazione delle lesioni alle quali tali immagini corrispondono e, contestualmente ad essa, un pulsante che permette di procedere nel gioco. Tale pulsante richiama la funzione `GoToNextPanel` (Listato 4.16), che disattiva il pannello corrente e attiva il successivo.

```
public void GoToNextPanel()
    {
        currentPanel.SetActive(false);
        textPanel.SetActive(false);
        buttonGo.SetActive(false);
        nextPanel.SetActive(true);
    }
```

Listato 4.16: Funzione che permette di passare al pannello successivo

4.3.7 SymptomManager

La funzione `OnToggleValueChanged` (Listato 4.17) viene richiamata ogni volta che viene selezionato o deselezionato un toggle button. Nel caso in cui il pulsante viene selezionato, aggiorna la lista dei toggle button attivi aggiungendo ad essa la label del toggle button. Se, invece, il pulsante viene deselezionato, il valore corrispondente alla label del pulsante viene rimosso dalla lista.

```
private void OnToggleValueChanged(bool isOn)
    {
        Toggle toggle=UnityEngine.EventSystems.EventSystem.current
        .currentSelectedGameObject.GetComponent<Toggle>();
        string label=toggle.GetComponentInChildren<Text>().text;

        if (isOn)
        {
            if (!activeToggleLabels.Contains(label))
            {
                activeToggleLabels.Add(label);
            }
        }
        else
        {
            if (activeToggleLabels.Contains(label))
            {
                activeToggleLabels.Remove(label);
            }
        }
        Debug.Log("Active Toggle Labels: " +string.Join(", ",
        activeToggleLabels.ToArray()));
    }
```

Listato 4.17: Funzione che gestisce i toggle button dei sintomi

La funzione più importante di questo script è la funzione `SelectButton` (Listato 4.18), la quale gestisce la selezione dei sintomi. Dopo aver riordinato alfabeticamente la lista dei sintomi presente nel database e quella risultante dai sintomi selezionati dal giocatore, tale funzione controlla che esse abbiano la medesima lunghezza. Nel caso in cui tale condizione non sia rispettata, verrà segnalato l'errore al giocatore; se, invece, questa condizione è rispettata, si procede alla verifica dell'uguaglianza tra le due liste. Se le due liste sono differenti viene

segnalato l'errore, mentre, se le due liste sono uguali, vengono disattivati tutti i componenti di UI che non sono più necessari, e viene fatta partire l'animazione relativa al comportamento del cane contagiato dal virus scelto.

```
public void SelectButton()
{
   symptomList=new List<string>(virus.symptoms);
   symptomList.Sort();
   activeToggleLabels.Sort();

   if (symptomList.Count==activeToggleLabels.Count)
   {
      bool hasError=false;
      for (int i=0; i<symptomList.Count; i++)
      {
         if (!string.Equals(symptomList[i].Trim().ToLower().Replace(" ", ""),
         activeToggleLabels[i].Trim().ToLower().Replace(" ", "")))
         {
            hasError=true;
            Debug.Log(symptomList[i].Trim().ToLower().Replace(" ", "")+" is different
            from "+activeToggleLabels[i].Trim().ToLower().Replace(" ", ""));
            break;
         }
      }

      if (!hasError)
      {
         virusModel=player.transform.Find(virus.virusName+"Virus(Clone)").gameObject;
         player.transform.position=new Vector3(-6.82f,0.72f,-0.38f);
         player.transform.rotation=Quaternion.Euler(0f, 270f,0f);
         panelSymptom.SetActive(false);
         dogInside.SetActive(false);
         virusModel.SetActive(false);
         dog.SetActive(true);
         mouth.SetActive(false);
         nose.SetActive(false);
         bite.SetActive(false);
         if (selectedVirus!=1)
         {
            meshRenderer.material=newMaterial;
            mainCamera.transform.localPosition=new Vector3(119,101,86);
            mainCamera.transform.localRotation=Quaternion.Euler(0,180,0);
            if(selectedVirus==0)
            {
               neurDisease.SetActive(true);
            }
         }
         else
         {
            burr.SetActive(true);
            mainCamera.transform.localPosition=new Vector3(600,100,-400);
            mainCamera.transform.localRotation=Quaternion.Euler(0,-90,0);
         }
         animator=dog.GetComponent<Animator>();
         animator.Play(utility.Load().ToString());
         Debug.Log("Right Symptoms!");
      }
      else
      {
         Debug.Log("Error, these are not the correct symptoms!");
         WrongText();
      }
   }
   else
   {
      WrongText();
   }
}
```

Listato 4.18: Funzione che gestisce la selezione dei sintomi

4.3.8 UtilityFunctions

La funzione `ReadFile` (Listato 4.19), quando viene richiamata, legge il testo contenuto in un file e lo restituisce come stringa.

```
public string ReadFile(string virusName){
      string text="";
      var textFile=Resources.Load<TextAsset>("Texts/"+virusName);
      text=textFile.text;
      return text;
   }
```

Listato 4.19: Funzione che legge un testo da un file

Mediante la funzione `Load` (Listato 4.20), l'indice del virus selezionato (nella schermata di selezione) viene ricaricato dalle `PlayerPrefs`.

```
public int Load()
   {
      return PlayerPrefs.GetInt("selectedVirus");
   }
```

Listato 4.20: Funzione che carica l'indice del virus selezionato

La funzione `EndGame` (Listato 4.21), una volta giunti al termine del gioco, viene richiamata per caricare la scena finale del gioco.

```
public void EndGame()
   {
      SceneManager.LoadScene(2);
   }
```

Listato 4.21: Funzione che carica la scena finale del gioco

Dalla scena finale del gioco è possibile ricominciare una nuova partita, cliccando su un apposito pulsante della UI, che richiama la funzione `Restart` (Listato 4.22).

```
public void Restart()
   {
      SceneManager.LoadScene(0);
   }
```

Listato 4.22: Funzione che permette di far ripartire il gioco dalla selezione

Inoltre, in ogni momento del gioco, è possibile chiudere l'applicazione di gioco stessa cliccando sulla "X" in alto a destra, la quale richiama la funzione `CloseGame` (Listato 4.23).

```
public void CloseGame()
   {
      Application.Quit();
      Debug.Log("Game's window is closed");
   }
```

Listato 4.23: Funzoine che chiude l'applicazione di gioco

4.3.9 VirusDB

La proprietà `NumVirus` (Listato 4.24) della classe `VirusDB` restituisce il numero di virus presenti nel database.

```
public int NumVirus
   {
      get
      {
         return viruses.Length;
      }
```

```
    }
```

Listato 4.24: Proprietà che restituisce il numero dei virus presenti nel DB

La funzione `GetVirus` (Listato 4.25) restituisce l'oggetto Virus contenuto nel database accedendovi mediante il suo indice.

```
public Virus GetVirus(int index)
    {
        return viruses[index];
    }
```

Listato 4.25: Funzione che restituisce il virus desiderato

4.3.10 VirusManager

Le funzioni `NextVirus` e `PreviousVirus` (Listato 4.26) permettono all'utente di scorrere tra i modelli dei virus presenti. Tali funzioni sono, rispettivamente, collegate ad una freccia "avanti" e ad una freccia "indietro" e vengono richiamate quando tali frecce sono cliccate. Entrambe le funzioni richiamano la funzione `UpdateVirus`.

```
public void NextVirus()
    {
        selectedVirus++;
        if (selectedVirus>=virusDB.NumVirus)
        {
            selectedVirus=0;
        }
        UpdateVirus(selectedVirus);
        Save();
    }

    public void PreviousVirus()
    {
        selectedVirus--;
        if (selectedVirus<0)
        {
            selectedVirus=virusDB.NumVirus-1;
        }
        UpdateVirus(selectedVirus);
        Save();
    }
```

Listato 4.26: Funzioni che permettono di scorrere tra i virus presenti

Mediante la funzione `UpdateVirus` (Listato 4.27), quando il giocatore clicca sulle frecce per scegliere il virus, viene istanziato il modello del virus da visualizzare nell'istante corrente; tale modello viene, poi, ridimensionato e posizionato. Infine vengono aggiornati il nome e le informazioni, visualizzati a schermo.

```
private void UpdateVirus(int selectedVirus)
    {
        Virus virus=virusDB.GetVirus(selectedVirus);
        GameObject thisModel=Instantiate(virusModels[selectedVirus],
        transform.position,transform.rotation) as GameObject;
        Destroy(virusObject);
        thisModel.transform.parent=transform;
        thisModel.transform.localScale=new Vector3(1,1,1);
        virusObject=thisModel;
        nameText.text=virus.virusName;
        string info=utility.ReadFile(virus.virusName);
        infoText.text=info;
    }
```

Listato 4.27: Funzione che mostra il virus corretto

Mediante la funzione `Save` (Listato 4.28) viene salvato nelle `PlayerPrefs` l'indice corrispondente al virus selezionato al fine di poter garantire la persistenza della scelta e di poter accedere anche in altre scene alle informazioni del virus.

```
private void Save()
    {
        PlayerPrefs.SetInt("selectedVirus", selectedVirus);
    }
```

Listato 4.28: Funzione che salva l'indice del virus selezionato

La funzione `ChangeScene` (Listato 4.29) viene richiamata una volta che il giocatore ha deciso con quale virus procedere ed avvia il gioco tramite un pulsante preposto della UI. Questa funzione si occupa semplicemente di caricare la scena di gioco successiva.

```
public void ChangeScene(int sceneID)
    {
        SceneManager.LoadScene(sceneID);
    }
```

Listato 4.29: Funzione che carica la scena successiva

CAPITOLO 5

Manuale utente

Il presente capitolo si prefigge di introdurre il manuale utente del videogioco sviluppato come parte integrante della presente tesi. All'interno di queste pagine, i giocatori avranno accesso a un'esposizione esauriente, la quale comprende non soltanto istruzioni dettagliate e l'elenco esaustivo dei comandi di gioco, ma anche immagini delle schermate con le quali l'utente deve interagire. In aggiunta, verranno fornite indicazioni passo passo per l'installazione agevole del videogioco.

5.1 Avvertenze per la salute

Giocare sempre in un ambiente ben illuminato. Fare pause regolari di circa 15 minuti per ogni ora di gioco. Smettere immediatamente di giocare se si viene colti da vertigini, nausea, affaticamento o cefalea. I soggetti sensibili a luci intermittenti, o a particolari forme o configurazioni geometriche, potrebbero soffrire di una forma di epilessia non diagnosticata ed essere soggetti a crisi epilettiche guardando la TV o giocando con i videogiochi. Se si è soggetti ad attacchi epilettici, consultare il medico prima di giocare con i videogiochi e contattarlo immediatamente qualora si dovessero riscontrare uno o più dei seguenti sintomi durante il gioco: alterazioni della vista, contrazioni muscolari, altri movimenti involontari, perdita di coscienza, confusione mentale e/o convulsioni.

5.2 Installazione e avvio

Per garantire un'esperienza di gioco agevole fin dall'inizio, si è deciso di rendere disponibile un pratico installer che semplifica il processo di installazione del videogioco.

Per installare correttamente il gioco, sarà sufficiente seguire attentamente i passaggi di seguito esposti:

1. *Download del File di Installazione*: scaricare il file di installazione del videogioco dal link: `https://github.com/MargheritaGaleazzi/VetGo_Dog_Edition`.
2. *Avvio dell'installer*: una volta completato il download, fare doppio click sul file di installazione. Questo avvierà l'installer e inizierà il processo di installazione.
3. *Configurazione della lingua di installazione*: l'installer darà la possibilità di scegliere la lingua con la quale eseguire l'installazione (italiano o inglese).

4. *Creazione di collegamenti*: In seguito l'installer offrirà la possibilità di creare il collegamento sul desktop. Questo collegamento semplificherà l'avvio del gioco in futuro.
5. *Selezione della directory di installazione*: l'installer chiederà di selezionare la cartella in cui si desidera installare il videogioco. È possibile accettare la cartella predefinita o scegliere una diversa destinazione sul sistema.
6. *Avvio dell'installazione*: una volta selezionate le opzioni desiderate, avviare l'installazione. L'installer copierà i file necessari e configurerà il videogioco sul sistema.
7. *Completamento dell'installazione*: dopo che tutti i file sono stati copiati e le impostazioni configurate, l'installer informerà che l'installazione è stata completata con successo.
8. *Avvio del videogioco*: infine è possibile avviare il gioco. Ciò può essere fatto mediante l'utilizzo del collegamento creato o ricercando l'eseguibile del videogioco nel sistema.

5.3 Elenco completo dei comandi e interfaccia utente

In questa sezione verranno elencati i comandi con i quali l'utente può interagire con la nostra applicazione e verrà illustrata l'interfaccia grafica mediante la quale tale interazione è possibile.

5.3.1 Schermata di selezione

Nella Tabella 5.1 sono mostrati i comandi mediante i quali il giocatore può selezionare il virus con il quale giocare. Nella Figura 5.1 viene mostrata l'interfaccia grafica relativa, appunto, alla selezione.

Azione	**Comando**
Scorrere tra i modelli dei virus	Click, con il tasto sinistro del mouse, sui pulsanti a forma di frecce avanti (›) e indietro (‹).
Avviare il gioco	Click, con il tasto sinistro del mouse, sul pulsante di avvio (▶).
Chiudere l'applicazione	Click, con il tasto sinistro del mouse, sul pulsante × .

Tabella 5.1: Elenco dei comandi per la schermata di selezione

5.3.2 Spostamento del virus

Nella Tabella 5.2 sono mostrati i comandi mediante i quali il giocatore può spostare il virus all'interno dell'area di gioco. Nella Figura 5.2 viene mostrata, appunto, l'area di gioco nella quale il giocatore si può muovere.

Azione	**Comando**
Spostare il virus a destra	Tasto [→], tenere premuto.
Spostare il virus a sinistra	Tasto [←], tenere premuto.
Spostare il virus avanti	Tasto [↑], tenere premuto.

Azione	Comando
Spostare il virus indietro	Tasto [↓], tenere premuto.
Spostare il virus in alto	Tasto [Space], tenere premuto.
Spostare il virus in basso	Tasto [ctrl], tenere premuto.
Proseguire dopo aver letto le informazioni sulla trasmissione del virus	Click, con il tasto sinistro del mouse, sul pulsante avanti (Avanti).
Mettere in pausa e riprendere la visione dell'animazione	Click, con il tasto sinistro del mouse, sul pulsante di avvio/pausa (▶/⏸)
Chiudere l'applicazione	Click, con il tasto sinistro del mouse, sul pulsante ×.

Tabella 5.2: Elenco dei comandi per lo spostamento del virus

Figura 5.1: Schermata di selezione

5.3.3 Selezione delle lesioni (gross)

Nella Tabella 5.3 sono mostrati i comandi mediante i quali il giocatore può selezionare le immagini di gross relative al virus scelto. Nella Figura 5.3 viene mostrata l'interfaccia grafica relativa alla selezione delle immagini di gross, mentre nella Figura 5.4 viene mostrata l'interfaccia contenente la spiegazione della gross.

Azione	Comando
Scorrere tra le immagini di gross	Click, con il tasto sinistro del mouse, sui pulsanti a forma di frecce avanti (›) e indietro (‹).
Selezionare l'immagine corrente	Click, con il tasto sinistro del mouse, sul pulsante di selezione (SELEZIONA).

Azione	Comando
Proseguire dopo aver letto le informazioni sulla lesione	Click, con il tasto sinistro del mouse, sul pulsante avanti (AVANTI).
Chiudere l'applicazione	Click, con il tasto sinistro del mouse, sul pulsante ×.

Tabella 5.3: Elenco dei comandi per la selezione delle immagini di gross

Figura 5.2: Schermata relativa al raggiungimento delle vie di trasmissione

5.3.4 Selezione delle lesioni (immagini istologiche)

Nella Tabella 5.4 sono mostrati i comandi mediante i quali il giocatore può selezionare le immagini istologiche relative al virus scelto. Nella Figura 5.5 viene mostrata l'interfaccia grafica relativa alla selezione delle immagini istologiche, mentre nella Figura 5.6 viene mostrata l'interfaccia contenente la spiegazione dell'immagine istologica.

Azione	Comando
Scorrere tra le immagini istologiche	Click, con il tasto sinistro del mouse, sui pulsanti a forma di frecce avanti (>) e indietro (<).
Selezionare le singole immagini	Click, con il tasto sinistro del mouse, sul toggle button ( ).
Confermare la selezione	Click, con il tasto sinistro del mouse, sul pulsante di selezione (SCEGLI).
Proseguire dopo aver letto le informazioni sulla lesione	Click, con il tasto sinistro del mouse, sul pulsante avanti (AVANTI).

Azione	Comando
Chiudere l'applicazione	Click, con il tasto sinistro del mouse, sul pulsante ×.

Tabella 5.4: Elenco dei comandi per la selezione delle immagini istologiche

Figura 5.3: Schermata di selezione delle immagini di gross delle lesioni

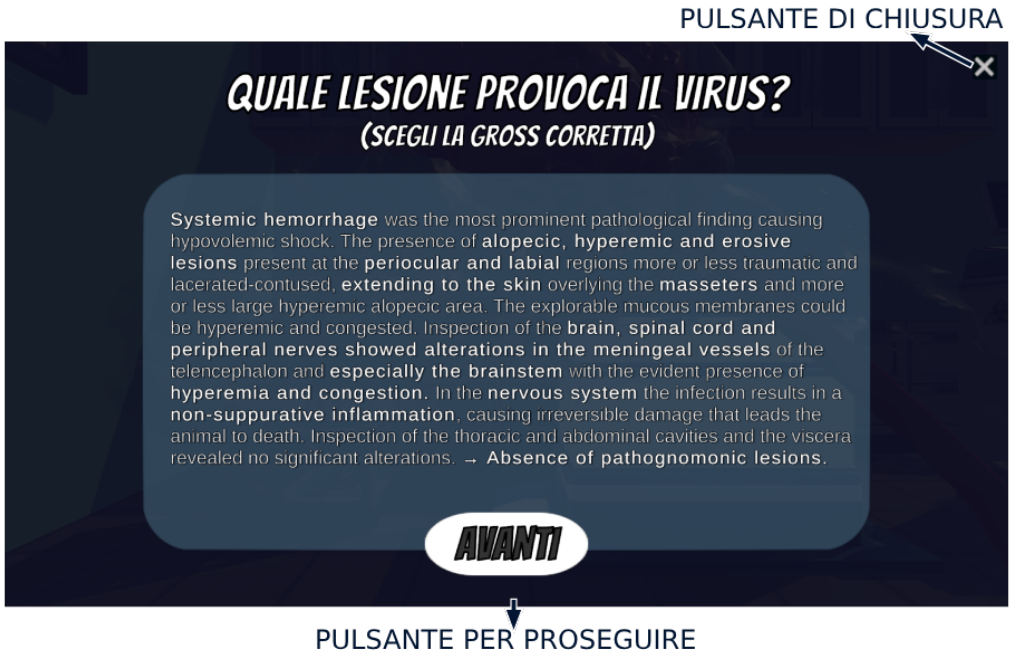


Figura 5.4: Schermata di spiegazione delle immagini di gross

Figura 5.5: Schermata di selezione delle immagini istologiche delle lesioni

Figura 5.6: Schermata di spiegazione delle immagini istologiche

5.3.5 Selezione dei sintomi

Nella Tabella 5.5 sono mostrati i comandi mediante i quali il giocatore può selezionare i sintomi legati al virus scelto in precedenza. Nella Figura 5.7 viene mostrata l'interfaccia grafica relativa, appunto, alla selezione.

Azione	Comando
Selezionare i sintomi	Click, con il tasto sinistro del mouse, sul toggle button (▢).
Confermare la selezione	Click, con il tasto sinistro del mouse, sul pulsante di selezione (SCEGLI).
Chiudere l'applicazione	Click, con il tasto sinistro del mouse, sul pulsante ×.

Tabella 5.5: Elenco dei comandi per la selezione dei sintomi

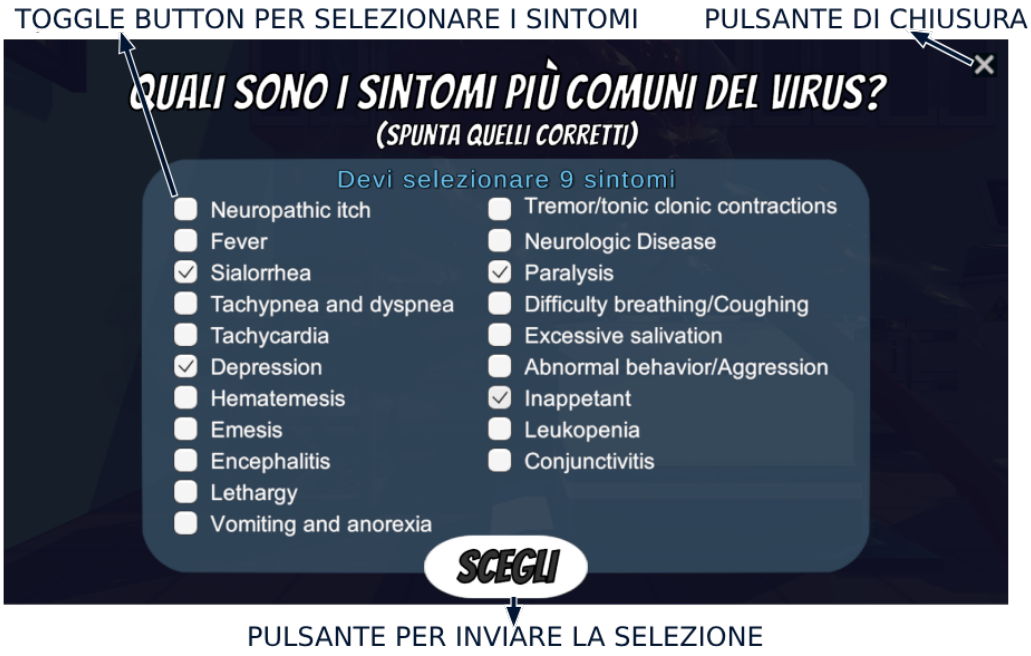


Figura 5.7: Schermata di spiegazione dei sintomi

5.3.6 Fine del gioco

Nella Tabella 5.6 sono mostrati i comandi mediante i quali il giocatore può decidere, una volta terminato il gioco, di ricominciare o di uscire dall'applicazione. Nella Figura 5.8 viene mostrata l'interfaccia grafica della schermata finale del gioco.

Azione	Comando
Ricominciare il gioco	Click, con il tasto sinistro del mouse, sul pulsante ricomincia (RICOMINCIA).
Chiudere l'applicazione	Click, con il tasto sinistro del mouse, sul pulsante ×.

Tabella 5.6: Elenco dei comandi per la selezione dei sintomi

Figura 5.8: Schermata finale

5.4 Come si gioca

Il gioco inizia con una schermata di selezione nella quale il giocatore deve selezionare il virus con il quale giocare.

In seguito alla selezione, il giocatore si troverà nella prima schermata di gioco, dove, utilizzando i comandi esposti nella sezione precedente dovrà muoversi fino a raggiungere la corretta via di trasmissione del virus prescelto.

Una volta che il giocatore riesce a raggiungere la via di trasmissione corretta, parte un breve video dove viene mostrato come il virus si "comporta" all'interno dell'organismo del cane. Terminata la visione di tale video, il giocatore deve, nella nuova schermata visualizzata, scegliere l'immagine di gross che pensa essere quella legata al virus con il quale sta giocando e cliccare sul pulsante che permette di selezionarla.

Se il giocatore seleziona la gross corretta gli viene mostrata una breve spiegazione sulla lesione stessa; poi, cliccando sull'apposito pulsante, gli viene data la possibilità di proseguire nel gioco.

A questo punto il giocatore si troverà a dover scegliere le immagini istologiche relative al virus; in questo caso, essendo la selezione multipla, il giocatore, per scegliere le immagini che ritiene corrette, dovrà spuntare il toggle button che si trova vicino ad esse, ed, una volta soddisfatto delle immagini selezionate, potrà proseguire cliccando sull'apposito pulsante.

Se il giocatore seleziona le immagini istologiche corrette gli viene mostrata una breve spiegazione su di esse; poi, cliccando sull'apposito pulsante, gli viene data la possibilità di proseguire nel gioco.

Il giocatore arriva, quindi, a dover compiere l'ultima scelta del gioco, ovvero, dall'elenco dei sintomi disponibili dovrà spuntare quelli che ritiene essere i sintomi del virus selezionato.

Se la selezione è quella corretta al giocatore verrà mostrata una breve animazione di alcune delle conseguenze che il virus ha sul comportamento del cane ed, infine, gli verrà mostrata la schermata finale nella quale gli verrà data l'opportunità di ricominciare il gioco o di uscire.

CAPITOLO 6

Discussione in merito al lavoro svolto

L'obiettivo di questo capitolo è quello di esaminare criticamente il nostro serious game, identificandone i punti di forza e di debolezza attraverso un'analisi SWOT (Strengths, Weaknesses, Opportunities, Threats) e confrontandolo con le alternative esistenti. Sarà fondamentale comprendere come il nostro gioco si distingue dagli altri e quali aspetti lo rendono unico.

6.1 Analisi SWOT

L'analisi SWOT è una metodologia strategica che permette di esplorare in maniera approfondita le dinamiche del nostro progetto (interne) e dell'ambiente circostante (esterne). Questa analisi fornisce una visione nitida dei punti di forza e di debolezza interni al progetto, nonché delle opportunità e minacce esterne che possono influenzare il suo successo.

In questa sezione esamineremo con particolare attenzione ognuna delle quattro categorie chiave dell'analisi SWOT. Andremo, quindi, a delineare ciò che rende il progetto competitivo (Strengths), i punti che richiedono miglioramento (Weaknesses), le opportunità che possiamo sfruttare per crescere (Opportunities) e le sfide che dobbiamo affrontare per mantenere il nostro impatto positivo (Threats).

Nella Figura 6.1 è riportata una panoramica di quanto segue. L'analisi SWOT risulta essere uno strumento fondamentale per l'elaborazione di strategie solide e basate su dati.

6.1.1 Strengths

Nell'analisi SWOT, i *punti di forza* (strengths) sono fattori interni che hanno un impatto positivo e differenziano il prodotto da quello dei concorrenti. I punti di forza che sono emersi dall'analisi del nostro serious game sono i seguenti:

- *Gratuità*: il serious game oggetto di questa tesi è offerto gratuitamente agli utenti come risorsa educativa, a differenza della maggior parte dei software/applicazioni che trattano argomenti similari. Offrendo gratuitamente questo serious game, vengono rimosse le barriere economiche che potrebbero scoraggiare l'adozione e l'utilizzo del gioco, rendendolo facilmente accessibile a una vasta gamma di utenti. In un panorama in cui l'accesso a risorse educative di qualità può essere costoso, il fatto che il nostro software sia completamente gratuito costituisce un punto di forza significativo, il quale potrebbe influenzare positivamente l'adozione e l'impatto del nostro serious game.

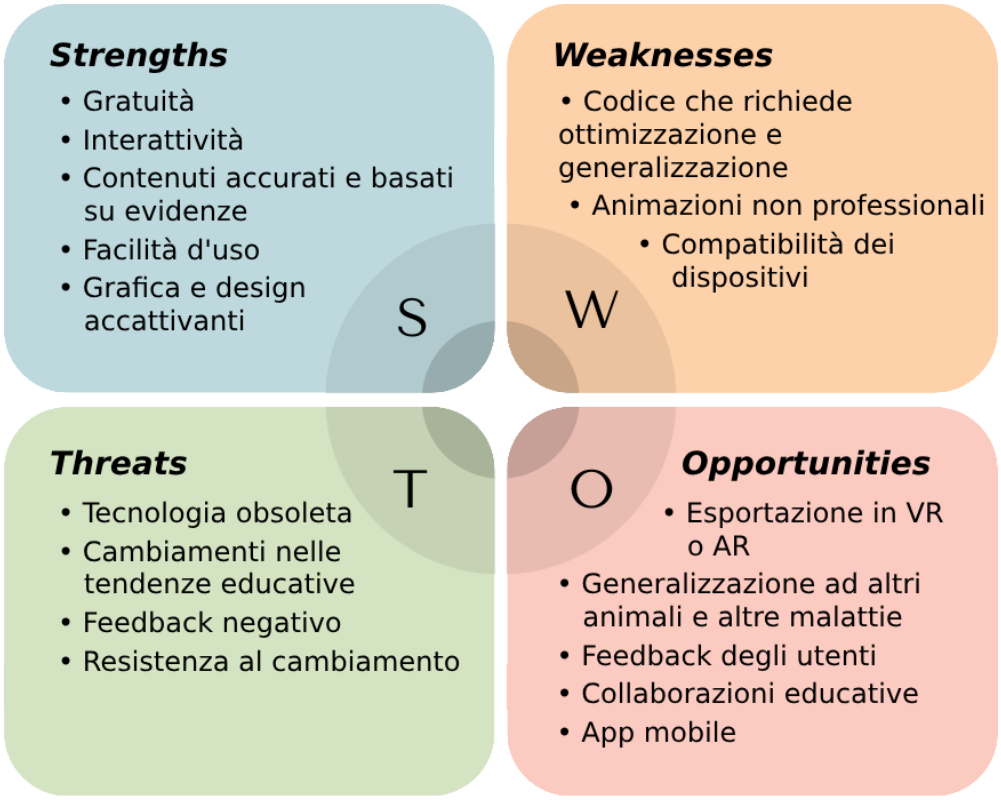


Figura 6.1: Analisi SWOT del nostro serious game

- *Interattività*: il nostro serious game, a differenza di altri giochi circa la trasmissione dei virus, presenta una maggiore interattività, in quanto, nella prima parte del gioco, il giocatore si ritrova calato nella parte del virus e, quindi, si trova ad agire in maniera tale da infettare il cane. L'utente partecipa, quindi, attivamente all'esperienza di apprendimento, prendendo decisioni e sperimentando direttamente i concetti relativi ai virus canini. L'utilizzo di un approccio attivo alla didattica può essere una modalità per migliorare la comprensione dei concetti e stimolare l'interesse degli utenti, contribuendo, così, al successo complessivo del nostro serious game e alla sua efficacia come strumento educativo.

- *Contenuti accurati e basati su evidenze*: nel nostro serious game ogni informazione e ogni dettaglio relativo ai virus canini ci sono stati forniti da un referente del dipartimento di veterinaria dell'Università di Padova. Questo garantisce che gli utenti ricevano informazioni corrette ed affidabili sull'argomento. I contenuti accurati presenti nel nostro serious game non solo contribuiscono alla credibilità del nostro gioco, ma rappresentano anche un valore aggiunto per gli utenti interessati a informazioni attendibili. In un mondo in cui la disinformazione può essere diffusa facilmente, la nostra enfasi sull'accuratezza e sulla solidità scientifica dei contenuti fa si che il nostro serious game possa essere visto come una risorsa educativa affidabile.

- *Facilità d'uso*: l'interfaccia utente del nostro serious game è stata progettata in modo semplice ed intuivo, per far si che gli utenti possano navigare facilmente attraverso il gioco. Questa facilità d'uso è fondamentale per raggiungere un pubblico ampio e diversificato, che includa, anche, coloro che potrebbero non avere una vasta esperienza nell'uso di giochi o applicazioni digitali. Rendere il nostro serious game accessibile a tutti contribuisce significativamente alla sua efficacia come strumento educativo,

poiché gli utenti possono concentrarsi sull'apprendimento degli argomenti anziché sul superamento di barriere tecnologiche.

- *Grafica e design accattivanti*: l'aspetto visivo del gioco è stato curato con attenzione per offrire un'esperienza esteticamente piacevole e coinvolgente agli utenti. I colori, le illustrazioni ed i modelli sono stati selezionati con cura per rendere il gioco visivamente stimolante e interessante. L'attenzione al design e alla grafica contribuiscono a catturare l'attenzione degli utenti fin dall'avvio del gioco e a far si che essi rimangano coinvolti e attenti durante tutto il percorso di apprendimento.

6.1.2 Weaknesses

Nell'analisi SWOT, i *punti di debolezza* (weaknesses) sono fattori interni che hanno un impatto negativo e differenziano il prodotto, in maniera negativa, da quello dei concorrenti. I punti di debolezza che sono emersi dall'analisi del nostro serious game, sono i seguenti:

- *Codice che richiede ottimizzazione e generalizzazione*: sebbene il codice sorgente del nostro serious game funzioni come previsto e fornisca un'esperienza di gioco soddisfacente, è evidente la necessità di ottimizzarlo e generalizzarlo per migliorare la manutenibilità, la scalabilità e la flessibilità del progetto.

- *Animazioni non professionali*: le animazioni, attualmente presenti nel gioco, sono state create da una programmatrice, il che ha comportato alcune limitazioni in termini di qualità visiva e impatto estetico. La mancanza di competenze specifiche in design grafico ha, probabilmente, limitato la qualità delle animazioni, e ciò potrebbe influire negativamente sull'aspetto complessivo del gioco e sulla sua capacità di coinvolgere gli utenti in modo visivamente accattivante.

- *Compatibilità dei dispositivi*: il nostro serious game è utilizzabile soltanto su computer che abbiano un sistema operativo Windows. Questa restrizione potrebbe escludere potenziali utenti che utilizzano altri sistemi operativi, come macOS o Linux, riducendo così l'accessibilità complessiva del gioco. La mancanza di diversificazione nella compatibilità dei dispositivi potrebbe limitare il raggiungimento di un pubblico più ampio.

6.1.3 Opportunities

Nell'analisi SWOT, le *opportunità* (opportunities) rappresentano fattori esterni che, se adeguatamente sfruttati, permettono di differenziare il prodotto da quello dei concorrenti. Le opportunità che sono emerse dall'analisi del nostro serious game sono le seguenti:

- *Esportazione in Realtà Aumentata (AR) o Realtà Virtuale (VR)*: un'opportunità chiave per migliorare il nostro serious game riguarda l'esplorazione della realtà aumentata (AR) e della realtà virtuale (VR). La trasposizione del gioco in queste tecnologie emergenti consentirebbe agli utenti di immergersi completamente nell'ambiente di gioco, rendendo il gioco ancor più coinvolgente e offrendo un approccio di apprendimento unico.

- *Generalizzazione ad altri animali e altre malattie*: un'altra importante opportunità per il nostro serious game riguarda la sua generalizzazione per coprire altre specie animali e altre malattie; ciò trasformerebbe il nostro serious game in uno strumento educativo versatile e flessibile. L'espansione, oltre che ad altre specie animali e ad altre malattie, potrebbe anche coinvolgere argomenti correlati, come la prevenzione delle zoonosi o l'igiene animale, e ciò potrebbe aumentare ulteriormente l'utilità del gioco come risorsa educativa.

- *Feedback dagli utenti*: un'altra opportunità per migliorare il nostro serious game riguarda la raccolta sistematica del feedback degli utenti. Mediante questi feedback possono essere messi in luce punti di forza del nostro gioco e punti di debolezza sui quali lavorare per adattare il nostro serious game alle esigenze degli utilizzatori.
- *Collaborazioni educative*: un'ulteriore opportunità è rappresentata dalla possibilità di stabilire collaborazioni educative con istituzioni accademiche, scuole, università o organizzazioni educative, oltre quella già presente con l'Università di Padova. Questa partnership potrebbe aprire porte importanti per espandere l'impatto educativo del nostro serious game, in quanto esso potrebbe essere integrato nei programmi di studio e nei corsi, fornendo agli studenti un approccio innovativo e coinvolgente all'apprendimento; inoltre, gli esperti accademici potrebbero contribuire con le loro conoscenze e le loro ricerche a garantire che i contenuti del gioco siano accurati ed aggiornati.
- *App Mobile*: un'opportunità notevole per ampliare il nostro serious game riguarda l'espansione su piattaforme mobili, come applicazioni per smartphone e tablet. Ciò consentirebbe agli utenti di accedere al gioco in maniera flessibile, ovunque si trovino, aumentando la sua accessibilità e portabilità. Inoltre, le applicazioni mobili possono sfruttare funzionalità specifiche dei dispositivi (come, ad esempio, il touchscreen) per creare un'esperienza di gioco ancora più coinvolgente e interattiva.

6.1.4 Threats

Nell'analisi SWOT, le *minacce* (threats) rappresentano fattori esterni che possono rappresentare ostacoli, sfide o potenziali problemi. Le minacce rappresentano fattori negativi o condizioni sfavorevoli che possono influire negativamente sull'obiettivo in questione. Le minacce che sono emerse dall'analisi del nostro serious game sono le seguenti:

- *Tecnologia obsoleta*: una minaccia per il nostro serious game è data dalla veloce obsolescenza delle tecnologie nell'attuale panorama tecnologico in rapida evoluzione ed espansione. Ciò potrebbe portare il nostro serious game a diventare meno compatibile con le piattaforme e i dispositivi di nuova generazione; inoltre potrebbe anche intaccare in maniera negativa la user experience del gioco.
- *Cambiamenti nelle tendenze educative*: un'altra minaccia è rappresentata dai cambiamenti nelle tendenze educative, in quanto le modalità e le priorità educative possono evolversi nel tempo, influenzando l'adozione e l'efficacia del nostro serious game.
- *Feedback negativo*: il feedback negativo da parte degli utenti rappresenta una minaccia che può avere un impatto assai negativo sulla reputazione e sull'adozione del nostro serious game; l'insoddisfazione e le possibili critiche degli utenti potrebbero avere un impatto sulla fiducia nel nostro progetto e sulla sua efficacia educativa.
- *Resistenza al cambiamento*: le persone tendono a resistere ai cambiamenti quando vi sono metodi o approcci consolidati nel tempo o quando il cambiamento pare loro come sconosciuto o minaccioso. Ciò potrebbe, quindi, portare ad un calo dell'adozione del nostro serious game come risorsa educativa.

6.2 Analisi comparativa dei software

In questa sezione effettueremo un'analisi comparativa dei diversi software disponibili, esplorando le loro caratteristiche chiave, i loro punti di forza e le loro limitazioni.

Questa sarà un'occasione per valutare i software attualmente presenti sul mercato e paragonarli a quello da noi sviluppato permettendoci di ottenere una visione completa e approfondita delle prestazioni e delle funzionalità di quest'ultimo.

È importante sottolineare che i software oggetto del confronto non sono progettati con lo stesso scopo del nostro. Poiché non abbiamo identificato soluzioni dirette con obiettivi simili, ci concentreremo sulla comparazione del nostro software con applicazioni che trattano la diffusione di virus ed epidemie e con software che permettono di esplorare l'anatomia canina. I software in questione sono:

- ESCAPE COVID-19;
- Plague Inc.;
- Micro-Combat;
- EasyAnatomy - Canine.

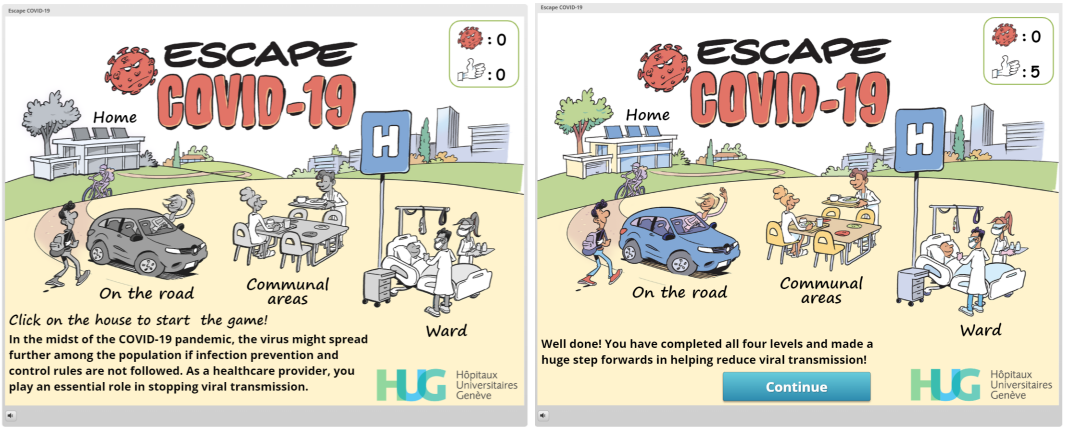
6.2.1 ESCAPE COVID-19

"Escape COVID-19" è un serious game online, sviluppato nel 2020 dall'Hôpitaux universitaires de Genève (HUG).

Lo scopo di questo gioco era quello di dotare di conoscenze e competenze approfondite nelle pratiche di prevenzione e controllo delle infezioni (IPC), il personale ospedaliero.

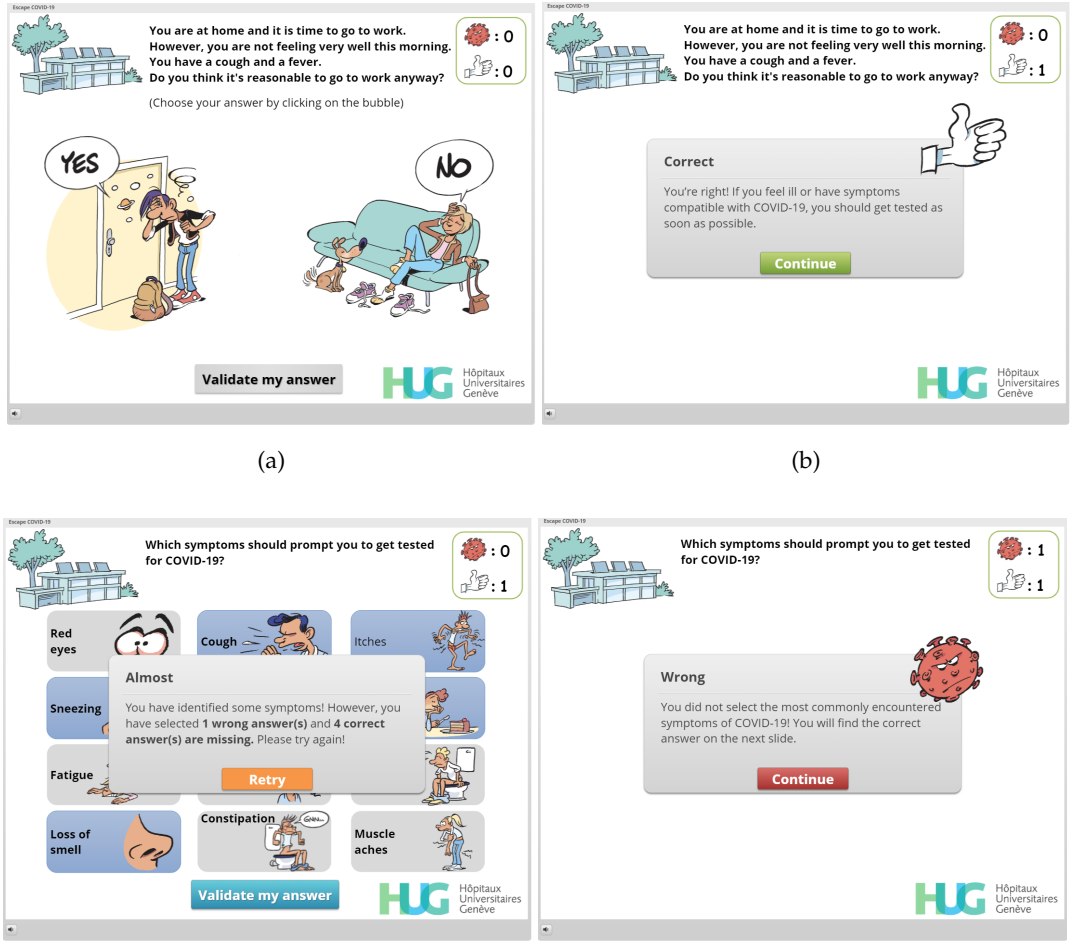
Il gioco è composto da una serie di scenari (Figura 6.2), ognuno dei quali contiene diverse domande (Figura 6.3 (a)). Alla fine di ogni domanda viene rivelata la risposta corretta insieme alla spiegazione delle indicazioni consigliate e ciò avviene indipendentemente dal fatto che la risposta, data dal giocatore, sia corretta o errata (Figura 6.3 (b),(c),(d)).
Nello specifico gli scenari presenti sono:

- *"a casa"*;
- *"in strada"*;
- *"aree comuni"*;
- *"in reparto"*.



(a) (b)

Figura 6.2: Schermata Iniziale (a) e finale (b) del gioco ESCAPE COVID-19



(a) (b)

(c) (d)

Figura 6.3: Schermata con domanda (a), con risposta corretta (b), con risposta parziale (c) e con risposta errata (d) del gioco ESCAPE COVID-19

Il livello finale (*"in reparto"*) è stato appositamente progettato per le esigenze informative dei professionisti che lavorano nei reparti di alta dipendenza e di terapia intensiva.

I punti di forza di questo gioco risiedono nella sua ricchezza di informazioni estremamente dettagliate ed esaustive, e nel fatto che, essendo giocabile online, può essere fruito da qualsiasi dispositivo dotato di una connessione a Internet. Tuttavia, è importante notare che presenta anche alcuni punti di debolezza, come le informazioni che non sono aggiornate e che sono valide solo per alcune zone (come viene evidenziato da un disclaimer all'inizio del gioco) e la poca interattività; infatti, l'unica interazione che il giocatore ha risiede nel cliccare sulla risposta corretta.

Paragonandolo al nostro serious game (Figura 6.4), anche se i due giochi sono relativi a due argomenti differenti, notiamo che questo gioco presenta una grafica più semplice e meno accattivante; inoltre, le informazioni risultano ormai datate oltre che relative ad un determinato territorio. Va, anche, sottolineato il fatto che il nostro serious game sia relativo a tre malattie differenti, mentre ESCAPE COVID-19 è relativo soltanto al Covid-19, come si può evincere dal nome.

	ESCAPE COVID-19	Vet-Go
Gratuità	✓	✓
Interattività	✗	✓
Contenuti accurati	✓	✓
Contenuti aggiornati	✗	✓
Grafica e design accattivanti	✗	✓
Piattaforma	ONLINE	PC
Facilità d'uso	✓	✓
Spiegazioni esaustive	✓	✓
Relativo a più malattie/infezioni	✗	✓

Figura 6.4: Comparazione di ESCAPE COVID-19 con il nostro serious game (Vet-Go)

6.2.2 Plague Inc.

"Plague Inc." (Figura 6.5 (a)) è un gioco di simulazione ad alta strategia. In questo gioco gli utenti scoprono e imparano come funzionano le infezioni virali.

(a) (b)

Figura 6.5: Schermata Iniziale (a) e di gioco (b) di Plague Inc.

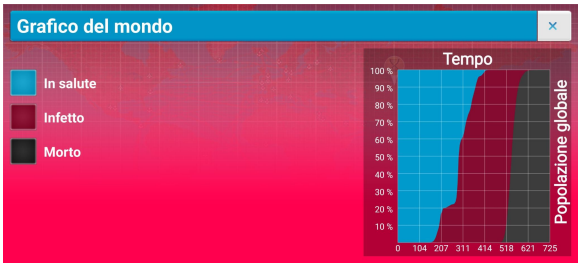
L'obiettivo del giocatore è quello di far crescere e controllare la distribuzione globale di un pericoloso agente patogeno; la difficoltà sta nel bilanciare contagiosità, letalità e visibilità (Figura 6.6 (a)) mentre si cerca di infettare e uccidere (Figura 6.6 (b)) quante più persone possibile prima che venga creata una cura (Figura 6.6 (c)).

Nel gioco, i giocatori possono utilizzare vari tipi di agenti patogeni, come batteri, virus, parassiti, prioni, funghi e nanovirus.

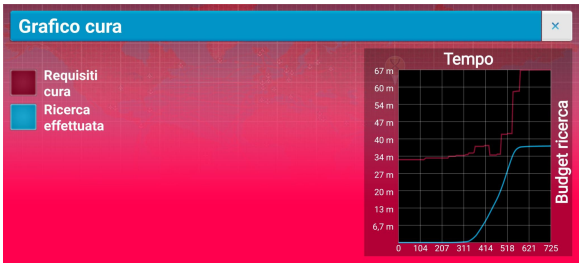
Questo gioco, sebbene non sviluppato con questa intenzione, funge da strumento educativo per aumentare la consapevolezza su come i virus possano diffondersi a livello globale e sulle contromisure che possono rivelarsi efficaci per debellarli.

L'interfaccia del gioco è costituita da una mappa del mondo (Figura 6.5 (b)), mediante la quale gli utenti ricevono un feedback visivo, grazie al cambiamento dei colori, sul livello di

(a)



(b)



(c)

Figura 6.6: Schermata con le capacità di trasmissione (a), grafico dello stato di salute nel mondo (b) e grafico sugli investimenti nella cura (c)

successo nel controllo della pandemia; nello specifico, il rosso è un indicatore di infezione mentre il nero di decessi.

Uno dei tratti più distintivi del gioco è il grado di coinvolgimento; esso cattura l'attenzione del giocatore con il suo approccio fantasioso al gameplay educativo.

I punti di forza di Plague Inc. sono dati dal grande grado di coinvolgimento che riesce a generare e dalla grafica semplice, intuitiva, ma alquanto accattivante.

Risulta un gioco completo e ben fatto; però non essendo sviluppato con l'intento di essere un serious game, presenta l'aspetto negativo di non fornire spiegazioni accurate su quanto avviene.

Inoltre, il gioco è a pagamento e ciò potrebbe costituire un ostacolo per alcuni utenti.

Paragonando Plague Inc. al nostro serious game (Figura 6.7), sebbene essi trattino tematiche leggermente differenti, notiamo che questo gioco fornisce un livello limitato di informazioni educative ed anche il grado di interattività è limitato al click di alcuni pulsanti sull'interfaccia.

Tuttavia, ciò è controbilanciato dal fatto che richiede un elevato coinvolgimento, grazie alla complessa strategia necessaria per giocarci.

	Plague Inc.	Vet-Go
Gratuità	✗	✓
Interattività	✗	✓
Contenuti accurati	✓	✓
Contenuti aggiornati	✓	✓
Grafica e design accattivanti	✓	✓
Piattaforma	SMARTPHONE	PC
Facilità d'uso	✓	✓
Spiegazioni esaustive	✗	✓
Relativo a più malattie/infezioni	✓	✓

Figura 6.7: Comparazione di Plague Inc. con il nostro serious game (Vet-Go)

6.2.3 Micro-Combat

"Micro-Combat" è un gioco di carte nel quale il giocatore deve mettersi nei panni di medici, ricercatori e personale sanitario (Figura 6.8).

(a) (b)

Figura 6.8: Schermata Iniziale (a) e di gioco (b) di Micro-Combat

Lo scopo del giocatore è quello di impedire che la popolazione si ammali a causa dell'attacco degli agenti patogeni che circolano nella città; il suo obiettivo sarà, quindi, quello di lavorare in squadra per garantire che nessuno dei personaggi in gioco perda tutte le proprie difese.

Il giocatore avrà a disposizione diverse misure preventive e farmaci per proteggere i cittadini, ma questi non saranno sempre sufficienti.

Questa App si basa sull'idea originale del gioco di carte creato da ISGlobal in collaborazione con i Laboratori de Jocs, e prodotto e validato con il finanziamento dello European Health Forum Gastein e dello European Antibiotic Awareness Day, un'iniziativa dell'ECDC (European Center Disease Control).

L'app Micro-combat è stata finanziata dallo European Joint Action on Antimicrobial Resistance and Healthcare-Associated Infections (EU-JAMRAI) e progettata in collaborazione con ISGlobal.

Uno dei maggiori punti di forza di questa applicazione risiede nei contenuti curati ed aggiornati e nel fatto che tratti diverse malattie e agenti patogeni. Inoltre, essa presenta una grafica accattivante, con colori vividi e disegni curati, che crea un'esperienza visiva coinvolgente per il giocatore.

Un grande limite dell'applicazione risiede, però, nella sua disponibilità soltanto su iPhone, il che limita il bacino di utenti.

Paragonando Micro-Combat a Vet-Go (Figura 6.9), si può evidenziare il fatto che Micro-Combat non presenti un elevato grado di interattività; infatti, esso è la trasposizione in applicazione di un gioco di carte. Inoltre, sebbene il numero di comandi sia limitato, il gioco non risulta di facile utilizzo, probabilmente per la mancanza di un tutorial o di spiegazioni esaustive sulla modalità di gioco. Infine si può evidenziare il fatto che sebbene sia un serious game, il numero di informazioni circa le malattie e gli agenti patogeni presenti risulta scarso.

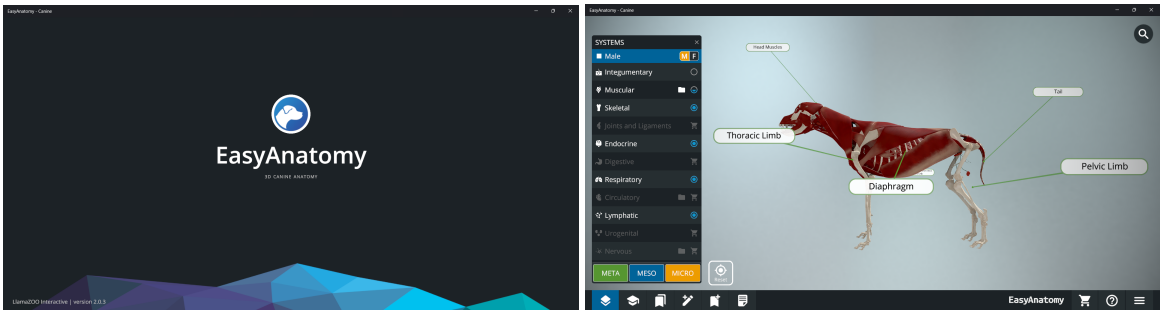
	Micro-Combat	Vet-Go
Gratuità	✓	✓
Interattività	✗	✓
Contenuti accurati	✓	✓
Contenuti aggiornati	✓	✓
Grafica e design accattivanti	✓	✓
Piattaforma	IPHONE	PC
Facilità d'uso	✗	✓
Spiegazioni esaustive	✗	✓
Relativo a più malattie/infezioni	✓	✓

Figura 6.9: Comparazione di Micro-combat con il nostro serious game (Vet-Go)

6.2.4 EasyAnatomy - Canine

"EasyAnatomy - Canine" è un software che permette di esplorare l'anatomia canina (Figura 6.10 (a)).

La schermata principale (Figura 6.10 (b)) del software è composta da un modello 3D di un cane, da un menù sul lato sinistro e da una barra di navigazione. Mediante il menù laterale è possibile effettuare una "dissezione virtuale", rimuovendo strati, nascondendo i vari componenti ed isolando le diverse regioni per poter apprendere in maniera ottimale le strutture anatomiche, mentre, mediante la barra di navigazione, è possibile accedere all'area quiz per testare le proprie conoscenze, visualizzare ed aggiungere segnalibri ed, infine, prendere note e rivederle.



(a) (b)

Figura 6.10: Schermata Iniziale (a) e principale (b) di EasyAnatomy

Selezionando una parte del modello in 3D dalla schermata principale, verrà visualizzata la finestra di dettaglio (Figura 6.11).

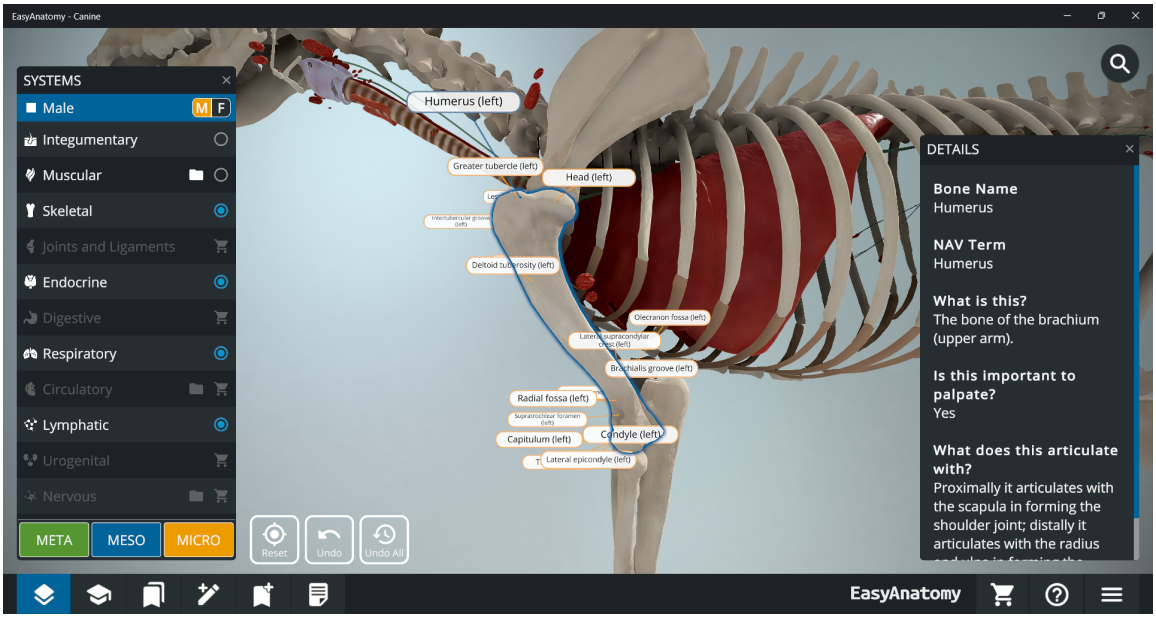
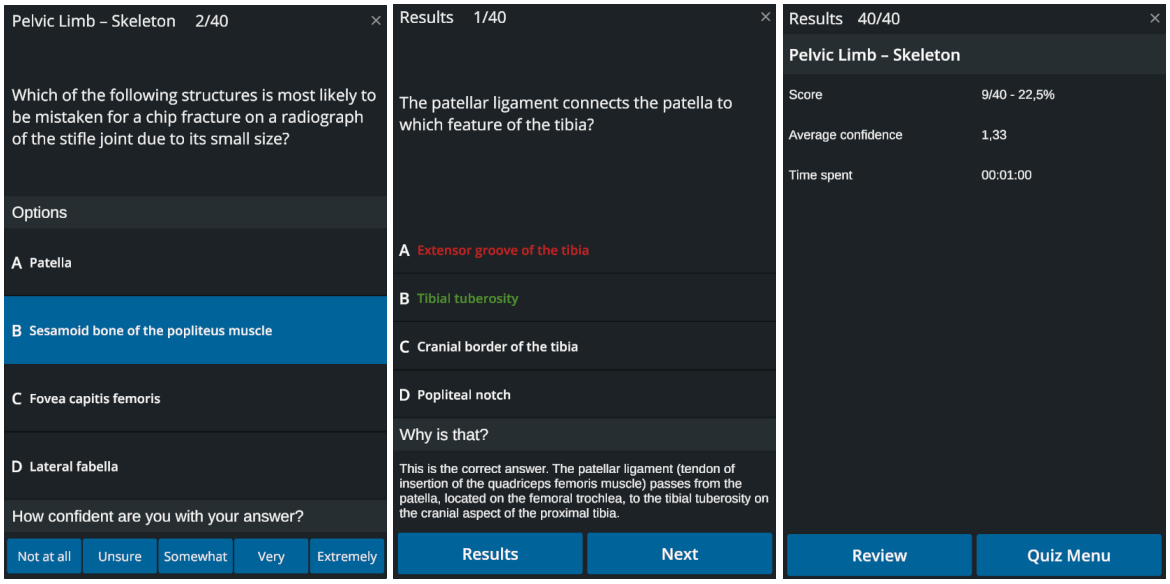


Figura 6.11: Schermata di dettaglio di EasyAnatomy

La sezione dei quiz permette di testare le proprie conoscenze mediante questionari pre-impostati e con la possibilità di crearne anche di personalizzati.

Terminato il test, rispondendo a tutte le domande (Figura 6.12 (a)), viene mostrata la parte di review (Figura 6.12 (b)), nella quale viene mostrata la risposta corretta ed una breve spiegazione. Infine, come mostrato nella Figura 6.12 (c), si può visualizzare la finestra dei risultati, che mostra il punteggio ottenuto, il livello di sicurezza nelle risposte date e il tempo impiegato per il quiz. EasyAnatomy - Canine risulta essere un software molto completo, che offre la possibilità di "navigare" nell'anatomia canina e apprendere o rinforzare l'apprendimento di quest'ultima; inoltre, nella versione EasyAnatomy+ per iPhone ed iPad, è possibile esplorare, studiare e sezionare l'anatomia virtuale del cane in realtà aumentata (AR).

In aggiunta, il software presenta diverse animazioni delle patologie più comuni.



(a) (b) (c)

Figura 6.12: Schermata del quiz (a), risposte corrette e spiegazioni (b), risultati (c) di EasyAnatomy

Paragonando EasyAnatomy al nostro serious game (Figura 6.13), si può sottolineare il fatto che, sebbene completo e con un'ottima grafica, il software EasyAnatomy risulta limitato nelle interazioni; inoltre questo software richiede la sottoscrizione di un piano a pagamento.

Va anche sottolineato che EasyAnatomy, a differenza di Vet-Go, non è un gioco.

	EasyAnatomy	Vet-Go
Gratuità	✗	✓
Interattività	✗	✓
Contenuti accurati	✓	✓
Contenuti aggiornati	✓	✓
Grafica e design accattivanti	✓	✓
Piattaforma	PC, MACOS, IPHONE, IPAD, SURFACE	PC
Facilità d'uso	✓	✓
Spiegazioni esaustive	✓	✓
Relativo a più malattie/infezioni	✓	✓

Figura 6.13: Comparazione di EasyAnatomy - Canine con il nostro serious game (Vet-Go)

Conclusioni e sviluppi futuri

Nei precedenti capitoli abbiamo esposto le varie fasi del processo di progettazione ed implementazione di un serious game, come supporto per l'apprendimento delle patologie causate dai virus sui cani, per gli studenti di corsi di laurea in Veterinaria.

Siamo partiti da una consultazione con la Facoltà di Medicina Veterinaria dell'Università di Padova, a seguito della quale abbiamo ricavato i requisiti.

Avendo, quindi, delineato quanto il nostro serious game doveva garantire, abbiamo potuto determinare le classi di interesse ed i casi d'uso.

Successivamente si è proceduto all'implementazione del codice. Durante lo sviluppo, abbiamo adottato una strategia Agile, effettuando periodiche revisioni con alcuni rappresentanti dell'Università di Padova, finalizzate a conoscere il gradimento e la soddisfazione di questi ultimi e a rispondere prontamente ai cambiamenti dei requisiti.

Infine, una volta terminata la fase di implementazione, si è proceduto a testare il nostro serious game e a generare l'installer.

Dopo aver sintetizzato gli aspetti chiave del nostro lavoro allo stato attuale, considereremo ora le possibili direzioni per il successivo sviluppo del software. Più specificatamente, i possibili sviluppi futuri risultano essere i seguenti:

- L'*esportazione in AR o VR* del serious game in quanto queste tecnologie risultano essere in continua evoluzione, con miglioramenti significativi in termini di prestazioni e con un CAGR [1] pari al 38,69%. Sarebbe, inoltre, auspicabile muoversi in questa direzione per rimanere al passo con l'evoluzione tecnologica e, quindi, rimanere rilevanti e competitivi.

- Il *refactoring del codice*, per ottimizzarlo e renderlo maggiormente leggibile, senza modificare le funzionalità dal punto di vista dell'utente.

- L'*aggiunta di altre malattie e altri animali* a quelli già presenti, il che renderebbe il nostro software ancora più efficiente come strumento di supporto allo studio.

- La *creazione di un'app mobile*, per rendere il gioco fruibile ovunque il giocatore si trovi.

- Il *miglioramento delle animazioni*, il quale renderebbe più coinvolgente e piacevole il gioco per gli utenti.

[1] Tasso annuo di crescita composto, più comunemente noto come CAGR dall'acronimo anglosassone Compounded Average Growth Rate.

- L'aggiunta della *modalità multiplayer*, in cui i giocatori possono collaborare o competere per risolvere sfide legate alla prevenzione delle malattie virali nei cani. Questa modalità potrebbe promuovere la discussione e la condivisione di conoscenze tra i giocatori.
- La capacità di rendere *disponibile su computer con sistema operativo macOs e Linux*, in modo che non ci siano limitazioni per gli utenti.
- L'*integrazione di strumenti di analisi dei dati*, per monitorare il comportamento dei giocatori, i loro progressi ed evidenziare le aree in cui potrebbe essere necessario fornire ulteriori informazioni o supporto.

Bibliografia

Articoli

A. J. Stapleton. “Serious games: Serious opportunities”. In: *Australian Game Developers Conference, Academic Summit, Melbourne* (2004).

T. Susi, M. Johannesson e P. Backlund. “Serious Games : An Overview”. In: *University of Skövde, School of Humanities and Informatics, Institutionen för kommunikation och information*. IKI Technical Reports HS-IKI-TR-07-001 (2007).

J. Breuer e G. Bente. “Why so serious? On the relation of serious games and learning”. In: *Journal for Computer Game Culture* 4 (1) (2010), pp. 7–24.

M. Romero et al. “Can Serious Games Contribute to Developing and Sustaining 21st-Century Skills?” In: *Games and Culture: A Journal of Interactive Media* 10 (2014).

K. Starks. “Cognitive behavioral game design: a unified model for designing serious games”. In: *Frontiers in psychology* 5 (2014), p. 28.

D. Drummond, A. Hadchouel e A. Tesnière. “Serious games for health: three steps forwards”. In: *Advances in Simulation* 2.1 (2017).

Libri

U. Ritterfeld, M. Cody e P. Vorderer. *Serious games: Mechanisms and effects*. Routledge, 2009. ISBN: 978-041-59-9370-8.

A. Maestri, J. Sassoon e P. Polsinelli. *Giochi da Prendere sul serio: Gamification, storytelling E game design*. FrancoAngeli, 2018. ISBN: 978-889-17-7924-3.

C. Dürnberger. “50. Playing a vet – serious games in the context of veterinary ethics”. In: 2022, pp. 328–333. ISBN: 978-90-8686-387-7.

C. Schrader. “Serious Games and Game-Based Learning”. In: 2023, pp. 1255–1268. ISBN: 978-981-19-2079-0.

Sitografia

A. Castello. *Compiti di Prestazione e compiti di Apprendimento Capitolo 1 - parte terza*. Psicologia del lavoro. Accessed: 20 May 2023. 2019. URL: https://www.psicologiadellavoro.org/compiti-di-prestazione-e-compiti-di-apprendimento-capitolo-1-parte-terza/.

Evidence based serious gaming, L'immersività come fattore critico di successo per la formazione. Assessment Online. Accessed: 20th May 2023. 2020. URL: https://www.assessmentonline.it/2020/07/06/evidence-based-serious-gaming-limmersivita-come-fattore-critico-di-successo-per-la-formazione/.

Serious games: apprendere un comportamento, giocando. Helaglobe. Accessed: 20 May 2023. 2020. URL: https://helaglobe.com/serious-games-apprendere-un-comportamento-giocando/.

G. Chiaramonte. *I serious games: cosa sono e perché utilizzarli*. Restorative Neurotechnologies. Accessed: 19 May 2023. 2022. URL: https://www.restorativeneurotechnologies.com/articoli-serious-games/serious-games-cosa-sono-e-perche-utilizzarli-saranno-efficaci.

Serious game: Significato, esempi e applicazioni. Viteco. Accessed: 19 May 2023. 2022. URL: https://www.vitecoelearning.eu/serious-game-e-apprendimento/.

M. Segatto. *Guida ai Serious Game*. Project Fun. Accessed: 19 May 2023. 2023. URL: https://www.projectfun.it/serious-game/guida/.

Serious game. Wikipedia. Accessed: 19 May 2023. 2023. URL: https://it.wikipedia.org/wiki/Serious_game.

A. Stephenson. *Mercato di Realtà aumentata e virtuale (Ar VR): Crescente Domanda e tendenze di mercato*. Game Is Hard. Accessed: 18 September 2023. 2023. URL: https://gameishard.gg/it/news/mercato-di-realta-aumentata-e-virtuale-ar-vr-crescente-domanda-e-tendenze-di-mercato/164974/.

Ringraziamenti

Questo spazio lo dedico alle persone che, con il loro supporto, mi hanno aiutato e sono state al mio fianco in questo meraviglioso percorso.

Vorrei innanzitutto ringraziare il mio relatore Domenico Ursino, che mi ha seguito, passo dopo passo e che è sempre stato pronto a fornirmi strumenti, migliorie, suggerimenti utili ai fini della stesura dell'elaborato.

Ringrazio Marta Giacomazzo, dottoranda di veterinaria dell'Università di Padova, che mi ha affiancato durante il mio tirocinio formativo, per la sua disponibilità nel fornirmi tutte le informazioni necessarie a far si che i contenuti di questo serious game fossero accurati.

Desidero esprimere la mia sincera gratitudine ai miei genitori, che sono stati costantemente al mio fianco. Vorrei ringraziare mia madre, Rosanna, per la sua incessante fiducia in me. Anche nei momenti in cui non ho dato il massimo, lei non ha mai smesso di credere in me. Il suo sostegno incondizionato e la sua capacità di vedere il mio potenziale, quando io stessa dubitavo, sono stati una fonte inesauribile di ispirazione. E mio padre, Paride, per il suo costante appoggio. In particolare, desidero sottolineare il suo incoraggiamento a non sacrificare completamente il mio benessere mentale e la mia felicità a favore dello studio, ricordandomi che entrambi sono importanti. La loro presenza e il loro supporto hanno reso possibile il mio successo in questo percorso accademico.

Un ringraziamento speciale a mia sorella Sofia, compagna di mille avventure e amica con la quale ho sempre condiviso tutto. Inoltre, le sono grata per aver accettato con entusiasmo il ruolo di beta tester per tutti i miei progetti universitari. Grazie, cara sorella, per essere stata al mio fianco in questo viaggio straordinario.

Desidero esprimere la mia profonda gratitudine al mio fidanzato, Lorenzo, per avermi trasmesso la sua immensa forza, dedizione e determinazione. Grazie per avermi dedicato tanto tempo e amore. Ti sono grata per il tuo supporto incondizionato e l'aiuto prezioso che mi hai offerto nel corso del mio percorso universitario. Inoltre, non posso fare a meno di ringraziarti per avermi costantemente spinto a superare i limiti, poiché hai sempre creduto che avrei potuto raggiungere obiettivi sempre più ambiziosi. Il tuo incoraggiamento è stato davvero importante, e ti sono profondamente grata per tutto quello che hai fatto per me.

Desidero esprimere la mia sincera gratitudine a quattro amici straordinari che sono stati al mio fianco. In primo luogo, vorrei ringraziare Emma e Matilde, le mie "amiche molto belle", per la loro presenza, le lunghe conversazioni, e l'incoraggiamento costante. La vostra amicizia è stata una fonte di forza e gioia in questi anni.

Inoltre, vorrei estendere il mio ringraziamento a Giacomo e Andrea, i miei amici di vecchia data, ormai parte della famiglia, per il loro costante sostegno e comprensione durante i momenti di stress e pressione. La vostra presenza ha reso questo viaggio più significativo e piacevole.

A tutti gli altri amici che hanno condiviso con me gioie, risate e preziosi momenti di vita durante questi anni voglio dedicare un ringraziamento speciale. Le vostre amicizie sono state una parte fondamentale della mia vita e hanno contribuito a rendere questo percorso un'esperienza indimenticabile.

Desidero esprimere la mia sincera gratitudine a Chiara, la mia compagna di corso e di quasi tutti i progetti. La tua dedizione, il tuo impegno e la collaborazione preziosa durante i nostri progetti e corsi comuni sono stati determinanti per il nostro successo collettivo.

Infine, non posso dimenticare di ringraziare me stessa, per la determinazione e l'ardore con cui ho affrontato ogni sfida. Questa tesi rappresenta il mio impegno e la mia passione per il mio campo di studio.