Scuola universitaria professionale della Svizzera italiana
Dipartimento tecnologie innovative

SUPSI

Code Quest

Studente/i
Matteo Cadoni
Adriano Cicco

Relatore
Masiar Babazadeh

Correlatore
-

Committente
Masiar Babazadeh

Corso di laurea
Ingegneria Informatica

Codice progetto
C10622

Anno
2022-2023

Data 05-05-2023

STUDENTSUPSI

Indice generale

Sommario

Indice delle figure

Abstract

Il gioco Code Quest è indirizzato agli studenti delle scuole medie e superiori per imparare la programmazione in modo divertente e coinvolgente. È stato sviluppato utilizzando Unity ed è un'applicazione educativa che insegna ai giocatori a programmare utilizzando il linguaggio di programmazione CQ, linguaggio creato ad hoc per il gioco.
L'obbiettivo consiste nel programmare un robot per completare una serie di missioni e avanzare nei livelli.

Ogni livello presenta una serie di obiettivi da raggiungere, come ad esempio segnalare nemici, superare ostacoli e altro ancora. I giocatori dovranno utilizzare il linguaggio di programmazione facile e intuitivo sviluppato da noi per programmare il comportamento del loro robot e superare le sfide proposte.

L'applicazione educativa è stata realizzata in modo da essere facilmente espandibile grazie alla flessibilità degli scripts che usano lo Unity Editor. L'applicazione è disponibile in inglese e in italiano.
Inoltre, il gioco assegna uno score per ogni soluzione ottenuta e l'obiettivo del giocatore è ottenere la valutazione più alta possibile. In questo modo, il gioco diventa anche una sfida per i giocatori, che devono cercare di migliorare costantemente le loro abilità di programmazione.

Questo gioco è un'ottima risorsa per chiunque voglia imparare a programmare. Le missioni proposte sono divertenti e coinvolgenti e permettono ai giocatori di acquisire una solida conoscenza di base della programmazione. Il gioco è facilmente espandibile, offre una grafica pulita e rappresenta anche una sfida per i giocatori che vogliono migliorare le loro abilità di programmazione.

English Version

The Code Quest game is aimed at middle and high school students to learn programming in a fun and engaging way. It was developed using Unity and is an educational application that teaches players how to program using the CQ programming language, a language created specifically for the game. The objective is to program a robot to complete a series of missions and advance through levels.

Each level has a series of objectives to achieve, such as flagging enemies, overcoming obstacles, and more. Players will have to use the easy and intuitive programming language developed by us to program their robot's behavior and overcome the proposed challenges.

The educational application is designed to be easily expandable due to the flexibility of scripts using the Unity Editor. The application is available in English and Italian.

In addition, the game assigns a score for each solution obtained, and the player's goal is to obtain the highest possible rating. In this way, the game also becomes a challenge for players to constantly try to improve their programming skills.

This game is an excellent resource for anyone who wants to learn programming. The missions offered are fun and engaging and allow players to gain a solid basic knowledge of programming. The game is easily expandable, offers clean graphics, and is also a challenge for players who want to improve their programming skills.

Progetto assegnato

1.1 Descrizione

Questo progetto di semestre prevede lo sviluppo di un applicativo per imparare a programmare, con target le scuole secondarie e scuole medie superiori. L'applicativo farà uso dei concetti di gamification e game-based learning per stimolare l'apprendimento della programmazione in un ambiente ludico e di sfida. Il programma infatti apparirà come un gioco e non come uno strumento didattico: per superare i livelli il giocatore dovrà dimostrare di aver compreso le basi della programmazione. L'idea del progetto è quella di sviluppare un prototipo dotato di tre livelli con difficoltà crescente, mantenendo il codice flessibile e pulito in modo che possa essere preso in mano da altri sviluppatori per renderlo un progetto completo. In particolare, sarà necessario semplificare il processo di level building, per esempio tramite un sistema di creazione di livelli e definire un documento di game design che descriva come il gioco è stato immaginato. Il relatore si occuperà di tutti gli aspetti legati alle competenze disciplinari da sviluppare tramite il gioco, sarà quindi importante definire un ciclo di sviluppo agile che permetta al relatore e agli sviluppatori di aggiornarsi e definire i passi futuri.

1.2 Compiti

- Definire un documento di game design
- Sviluppare un sistema prototipale di gioco educativo per l'insegnamento del coding Sviluppare delle interfacce e utilizzare dei supporti grafici adeguati ad un prototipo in modo da renderlo fruibile per eventuali test con classi
- Sviluppare un sistema di creazione di livelli che possa essere usato per sviluppare tutti i livelli del gioco anche in un secondo momento
- Creare tre livelli di demo

1.3 Obbiettivi

L'obiettivo principale è quello di sviluppare uno strumento educativo in forma prototipale che possa essere utilizzato per dei primi test sul campo e la seguente raccolta di feedback. Questo strumento sarà in seguito esteso con progetti di ricerca o tesi di bachelor/master per renderlo a tutti gli effetti un gioco educativo funzionale che può essere utilizzato in classe. È quindi importante svilupparne delle solide basi per poterci continuare a lavorare sopra. Sarà importante avere lungimiranza e sviluppare un codice pulito e flessibile.

1.4 Tecnologie

Unity

Introduzione

In questa relazione andremo a presentare il progetto Code Quest. Esso pone l'obbiettivo di realizzare un videogioco che insegni la programmazione agli studenti delle scuole medie e superiori.

La gamification è uno strumento che nella didattica si sta diffondendo sempre di più nel mondo dell'insegnamento e buoni risultati che si sono ottenuti negli anni dimostrano la sua utilità.

Molte volte i corsi di informatica possono essere noiosi e poco coinvolgenti per gli studenti e anche per i docenti. Tramite questo innovativo strumento con questo progetto si vuole tentare di rendere l'apprendimento della programmazione un'esperienza più divertente e coinvolgente per gli studenti.

Il motore di gioco utilizzato per lo sviluppo del progetto è Unity, uno dei software di sviluppo di videogiochi più utilizzati al mondo. La scelta di questo motore di gioco ci ha permesso di creare un'esperienza di gioco più coinvolgente e immersiva, in grado di fornire un ambiente di apprendimento dinamico e interattivo per gli studenti.

Durante lo sviluppo del progetto, abbiamo approfondito il funzionamento delle strutture portanti del gioco, come ad esempio il sistema di coding interno al gioco. Inoltre, abbiamo sviluppato dei plugin per agevolare lo sviluppo del gioco, come ad esempio il plugin di gestione dei dialoghi e il plugin di gestione delle animazioni.

Nella parte iniziale andremo a illustrare cosa ci ha ispirato e che cosa esisteva prima dello sviluppo di Code Quest.

Successivamente verrà illustrata la soluzione con particolari approfondimenti sui temi più importanti, spiegando in modo dettagliato il funzionamento delle strutture portanti del progetto e su come utilizzare i plugin che sono stati sviluppati per agevolare lo sviluppo.

Nella fase conclusiva della relazione, verranno presentati i risultati ottenuti grazie allo sviluppo del progetto Code Quest. Verranno poi esplorati i possibili sviluppi futuri del progetto, che potrebbero riguardare l'aggiunta di nuovi livelli di gioco, la creazione di nuovi strumenti per l'insegnamento della programmazione o l'introduzione di nuove funzionalità per migliorare l'esperienza di gioco degli utenti.

Motivazione e Contesto

2.1. Motivazione

La motivazione di questo progetto di semestre è quella di creare un'applicazione che utilizzi la gamification e il game-based learning per rendere l'apprendimento della programmazione un'esperienza ludica e coinvolgente per gli studenti delle scuole secondarie e medie superiori. L'obiettivo è di sviluppare un prototipo flessibile e pulito, in modo che possa essere successivamente migliorato e ampliato da altri sviluppatori. Inoltre, è importante definire un ciclo di sviluppo agile per consentire al relatore e agli sviluppatori di aggiornarsi e di definire i passi futuri in modo coerente con gli obiettivi didattici. In sintesi, la motivazione principale del progetto è quella di rendere l'apprendimento della programmazione più accessibile e divertente per gli studenti delle scuole secondarie e medie superiori

2.2 Contesto

Il contesto di questo progetto di semestre è quello dell'istruzione delle scuole secondarie e medie superiori, con l'obiettivo di rendere l'apprendimento della programmazione più stimolante e coinvolgente per gli studenti. L'applicativo che verrà sviluppato farà uso dei concetti di gamification e game-based learning per creare un ambiente ludico e di sfida, in cui il giocatore dovrà dimostrare di aver compreso le basi della programmazione per poter superare i livelli. L'idea è di sviluppare un prototipo dotato di tre livelli con difficoltà crescente, mantenendo il codice flessibile e pulito in modo da poter essere ampliato e migliorato in futuro da altri sviluppatori. Inoltre, è importante definire un ciclo di sviluppo agile per consentire al relatore e agli sviluppatori di aggiornarsi e di definire i passi futuri in modo coerente con gli obiettivi didattici.

Problema

L'insegnamento del coding rappresenta una sfida che richiede differenti approcci in base ai destinatari coinvolti. Tuttavia, esistono alternative alla didattica tradizionale in grado di agevolare l'apprendimento, ovvero il processo di acquisizione di nuove conoscenze, comportamenti, abilità, valori o preferenze e può riguardare la sintesi di diversi tipi di informazione [1].

Una di queste è la gamification, ossia, un processo di apprendimento in cui i giocatori risolvono problemi e superano sfide e superano sfide in contesti basati sul gioco per raggiungere i risultati di apprendimento desiderati risultati di apprendimento desiderati che si concretizza nella semplice forma di badge, punti e classifiche. [2]

In particolare, questa metodologia risulta molto efficace per i più giovani grazie a diversi fattori, tra cui l'elemento della sfida.

Stato dell’arte

Per valutare la tipologia di prodotto da sviluppare e non avendo particolari vincoli da rispettare se non quanto specificato dalla descrizione del progetto, abbiamo effettuato una fase di analisi del mercato videoludico e sulle proposte già esistenti.

Questo ci ha permesso di comprendere le tendenze attuali e di individuare le caratteristiche più apprezzate dai giocatori. Prima però di parlare del mercato è bene spiegare cosa lega la gamification con l’apprendimento.

4.1 Competitività e Motivazione

L’utilizzo della gamification nell’insegnamento è un argomento di ricerca in continuo sviluppo, poiché ha dato diversi risultati sul miglioramento delle skills degli studenti; come riportato da uno studio condotto da un gruppo di ricerca cinese [2], l'utilizzo della gamification potrebbe migliorare la motivazione e le competenze degli studenti nella programmazione. In particolare, l'apprendimento tramite giochi con meccaniche competitive sembrerebbe avere un impatto positivo sulle abilità di pensiero e sulla motivazione degli studenti. Tuttavia, l'aspetto della competizione potrebbe penalizzare le abilità di cooperazione e condivisione.

Si può concludere che l'utilizzo di meccaniche competitive di problem-solving sembra dare migliori risultati sulle abilità di pensiero e sulla motivazione dell'apprendimento. Questa conclusione è in linea con gli studi condotti da Abbasi [3], che ha evidenziato come gli studenti che giocano ai giochi forniti dagli insegnanti raggiungano migliori risultati accademici rispetto a quelli che creano giochi da soli. Tuttavia, è importante fornire una guida adeguata agli studenti che devono creare i propri giochi, per tener conto della complessità del processo di programmazione, soprattutto per i principianti. [2]

4.2 Proposte del mercato

Il mercato attuale ha diversi giochi che hanno come obbiettivo l’insegnare la programmazione. Ci sono diverse tipologie, ma la maggior parte dei titoli tende a basarsi su meccaniche gestionali e puzzle.

Tra le tipologie più comuni di giochi di programmazione, troviamo quelli incentrati sull’hacking, sulla risoluzione di enigmi e l’uccisione di nemici mediante l’utilizzo di robot da combattimento.

Alcuni esempi di giochi che utilizzano queste meccaniche, e a cui ci siamo ispirati, sono:

Human Resource Machine

Questo gioco uscito nel 2016 sulla maggior parte delle piattaforme PC e mobile è un puzzle game basato sulla programmazione visuale. Lo scopo è quello di programmare degli addetti per eseguire task nell’azienda, ad esempio prendere e spostare degli oggetti dai nastri trasportatori.

La metafora a cui gli sviluppatori si sono ispirati è quella dei concetti del linguaggio assembly. Infatti, gli elementi del gioco rappresentano i vari elementi. Le istruzioni che vengono inserite rappresentano le istruzioni, l’abilità dell’avatar di tenere un oggetto rappresenterebbe i registri e lo spazio sul pavimento la memoria principale.

Le istruzioni di salto come i cicli sono rappresentate da una freccia per aiutare l’utente a identificare le istruzioni.

L’utente può velocizzare, rallentare e fermare il flusso del programma costruito. [4]

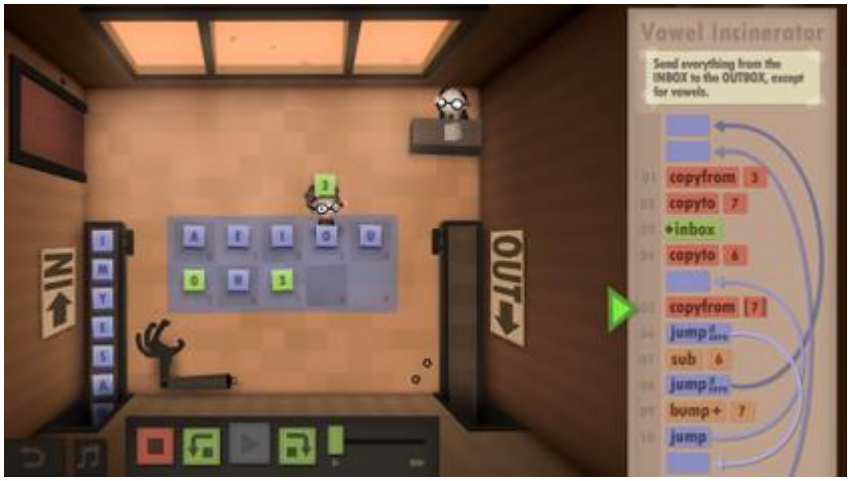


Figura 1 Human Resource Machine

RoboZZle

È richiesto di programmare un robot per risolvere una serie di enigmi. I livelli diventano sempre più difficili man mano che si procede nel gioco.

Lightbot

Il giocatore deve programmare un piccolo robot per accendere tutte le luci in un determinato livello. Il gioco richiede l'uso di concetti di base di programmazione come i cicli, le condizioni e le funzioni. [5]

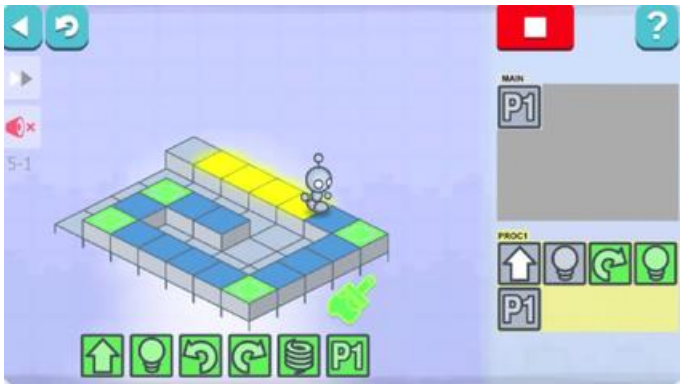


Figura 2 Lightbot

CodeCombat

Gioco uscito nel 2013 orientato all'insegnamento dei concetti e applicazione della programmazione.

Il gioco è realizzato per persone di età compresa tra i 9 e il 16 anni e i linguaggi che vengono insegnati sono JavaScript, Python e CoffeScript oltre ai concetti base dell'informatica.

CodeCombat lavora a contatto diretto con le scuole per offrire supporto alla didattica con la realizzazione di contenuti extra.

Nel 2014 è diventato open-source ed è stato rilasciato un editor per la realizzazione dei livelli.

Lo stile grafica è cartoon e l'ambientazione è in un universo fantasy. [6]

Il gioco ha anche una modalità multiplayer con inclusione di una sezione esport.

Le partite multiplayer si basano su scontri uno contro uno in modalità tower-defence [1]. [7]

[1] Tower-defence: modalità in cui si deve conquistare la torre dell'avversario, esempi Clash Royale Code Quest

Figura 3 CodeCombat

4.4 Analisi dei dati e considerazioni

Dalle informazioni raccolte durante l'analisi del mercato videoludico, abbiamo notato che molti giochi che sono atti all'insegnamento della programmazione si basano sull'utilizzo di linguaggi a blocchi e su sistemi a grafo. Tuttavia, abbiamo riscontrato che tali approcci, sebbene semplici, possono risultare disordinati e dispersivi quando applicati a problemi più complessi.

Inizialmente, abbiamo valutato la possibilità di utilizzare un approccio basato sui diagrammi flow-chart, ma anche in quel caso abbiamo riscontrato difficoltà nella gestione dei codici, anche quelli relativamente semplici, i vari collegamenti che interconnettono gli elementi del flow-chart [Figura 4], all'aumentare della complessità rendono contorto il seguire le connessioni tra i vari elementi rendendo tutto confusionario.

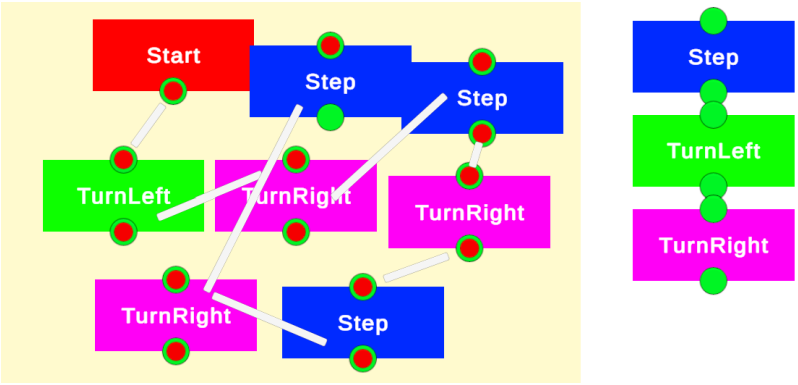


Figura 4 FlowChart implementato in fase di analisi

Dopo aver preso in considerazione questi aspetti, abbiamo deciso di sviluppare un linguaggio semplificato che consentisse un'interazione più fluida con l'ambiente di gioco. Inoltre, abbiamo notato che la componente competitiva è una meccanica piuttosto rara in questo tipo di giochi, con pochi titoli che integrano un multiplayer che richiede la programmazione di qualcosa.

Soluzione

Dopo le dovute analisi e considerazioni la nostra soluzione unisce la programmazione di un robot in un ambiente a singolo giocatore (successivamente a più giocatori), con meccaniche cooperative e competitive, permettendo agli utenti di migliorare le loro abilità di pensiero critico e di problem-solving mentre si divertono a giocare.

Prima di arrivare a realizzare il prodotto finito sono state valutate differenti alternative che, a seguito di test e riflessioni, sono state accantonate per la poca flessibilità che offrivano.

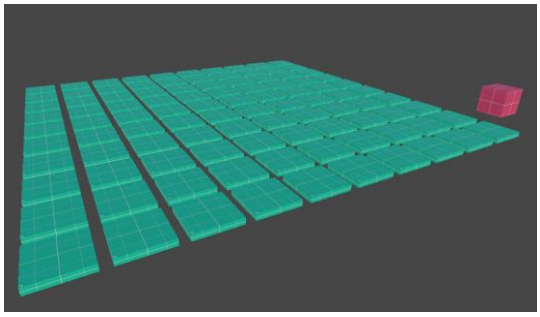


Figura 6 Griglia 3D

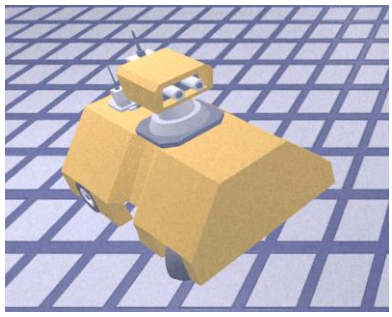


Figura 5 Robot

La prima alternativa che abbiamo valutato consente al robot di muoversi solamente all'interno di una grigia predefinita attraverso degli step. Questa alternativa rendeva il robot troppo prevedibile diminuendo notevolmente il numero di soluzioni che il giocatore poteva ideare per risolvere la sfida a cui il player era posto.

L'approccio della griglia andava di pari passo all'utilizzo del flow chart per la programmazione del robot. Il flow chart collegato con il basso numero di comandi necessari per comandare il robot all'interno di una griglia consentiva di creare situazioni di bassa complessità rendendo il gioco poco accattivante e basilare.

Per non rendere il gioco troppo prevedibile e dargli una possibile utilità futura abbiamo deciso di creare un interfacciamento simile alla programmazione di un robot con Arduino nella realtà dove per ridirezionare il robot ed effettuare delle azioni si fa affidamento a sensori e misurazioni.

La fisica nel gioco è basata sul motore fisico integrato in Unity. Grazie a questo motore fisico, è stato possibile creare una simulazione realistica del movimento di un piccolo robot controllabile con 2 ruote sterzanti e 4 ruote, compresa la simulazione degli ammortizzatori.

Abbiamo deciso di utilizzare la punta e clicca come sistema di movimento per il giocatore dove quest'ultimo si muoverà in automatico verso il punto da noi cliccato fino a quando non lo avrà raggiunto. È stato adottato questo sistema di movimento per la sua netta intuitività e per rendere possibile con discreta facilità la sua compatibilità per mobile.

La soluzione è stata implementata utilizzando Unity, un motore grafico e di sviluppo di videogiochi multipiattaforma offrendo allo sviluppatore un ambiente di sviluppo integrato (IDE) con funzionalità per la gestione delle risorse, il design visuale, la programmazione, la simulazione fisica, la grafica 2D e 3D, l'audio e la pubblicazione su diverse piattaforme.

5.1 Game Design

In questo paragrafo andremo a riassumere ciò che è specificato nel documento di game design allegato (**AL1**).
L'obbiettivo di Code Quest è quello di insegnare la programmazione in modo semplice e divertente.
Il gioco si basa su livelli di difficoltà crescente nei quali l'utente dovrà risolvere dei problemi utilizzando un linguaggio creato ad hoc per il gioco, CQ.
Il gioco ha meccaniche puzzle e lo scopo principale del giocatore è quello di programmare un robot, utilizzando tutte le sue funzionalità, con l'obbiettivo di superare le varie missioni.
Il codice scritto dall'utente sarà anche valutato e, in base al punteggio ottenuto, verranno date un numero di stelle che va da 1 a 3.
Il gioco è ambientato nei laboratori Quest, una specie di area 51 nella quale si creano dei robot particolari e vengono addestrati i piloti che dovranno comandarli.
Lo stile del gioco low-poly[2], questo ha consentito uno sviluppo più semplice dal punto di vista grafico, poiché non richiede grosse abilità in campo di modellazione 3D. Inoltre, consente di avere una grafica più leggera, e quindi facilmente eseguibile da dispositivi non troppo performanti in campo grafico.

L'interfaccia grafica e il linguaggio tentano di riprodurre nel modo più simile un ambiente di sviluppo di base con un linguaggio molto simile a linguaggi come Java, C, C++, C#… tendendo conto della semplificazione di alcune istruzioni.
Quanto sviluppato a livello di semplificazione andrà testato sul campo per valutare l'effettiva funzionalità.

Figura 7 Livello di CodeQuest

[2]Low-Poly: i modelli 3D che possiedono un basso numero di poligoni [9]

5.2 CQ Interpreter for Unity

"CQ Interpreter for Unity" ha la funzione di tradurre un linguaggio di programmazione personalizzato all'interno di un motore di gioco, nel nostro caso unity. Ciò richiede la creazione di una struttura che segua il ciclo di vita dell'engine, in quanto bisogna considerare fattori quali il tempo di esecuzione di alcune operazioni, come la fisica, e altre variabili che dipendono dal funzionamento di Unity.

Il sistema sviluppato si compone di diversi componenti:

- **CQInterpreter**: converte il codice da CQ a chiamate C# interpretabili dal CodeManager.
- **CodeManager:** gestisce l'esecuzione del codice e le chiamate delle istruzioni
- **ExecutionContext**: esegue le istruzioni inviate dal CodeManager.

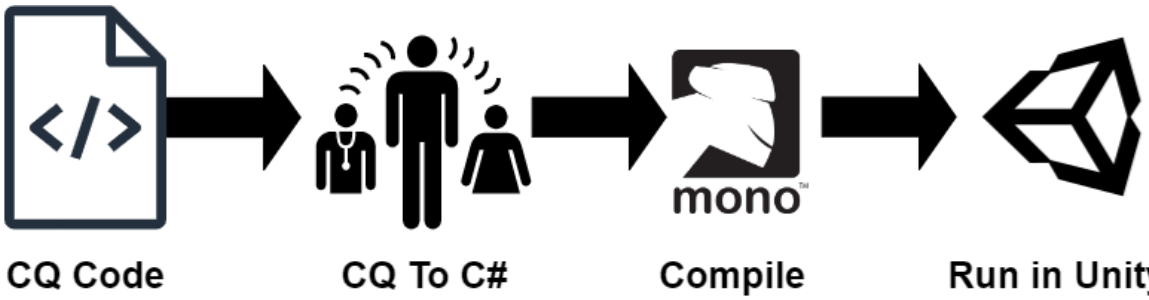


Figura 8 Flusso dell'interprete CQ per essere eseguito in unity

Il sistema si divide in cinque fasi:

1) Scrittura del codice in linguaggio CQ
2) Conversione delle istruzioni scritte in CQ, trasformandole in codice C# interpretabile dal manager del codice
3) Compilazione del codice C# ottenuto precedentemente
4) Esecuzione della funzione Main statica che viene costruita e compilata in c#, che preparerà la lista delle istruzioni da eseguire.
5) Esecuzione del CodeManager

5.2.1 Linguaggio CQ

Il linguaggio di programmazione creato per CodeQuest (CQ) permette di utilizzare molteplici costrutti presenti in qualsiasi linguaggio di programmazione moderno, consentendo di scrivere il codice che si adatta meglio al problema da affrontare.
Per la sua implementazione è stato utilizzato l'Interpreter Pattern, utile per comprendere il significato di una riga attraverso la sua composizione e riutilizzare l'informazione appresa per la conversione effettuata in seguito.
Tra i costrutti disponibili in CQ possiamo trovare:

- Loop
- If
- Else
- Variabili
- Metodi

5.2.2 Loop

Il loop permette di eseguire delle operazioni contenute nel suo corpo fintanto che la condizione che presenta al suo interno è vera (analogo al while in qualsiasi linguaggio di programmazione), è stata scelto il nome loop per rendere il nome del costrutto più intuitivo per le persone che si approcciano alla programmazione per la prima volta

5.2.3 If-else

l'if e l'else hanno delle funzionalità analoghe a qualsiasi altro linguaggio di programmazione, se la condizione contenuta all'interno dell'if è vera esegue il codice contento nel suo corpo, altrimenti esegue le istruzioni contenute nel corpo dell'else.
Sia dei loop che negli if-else è possibile eseguire istruzioni annidate.

5.2.4 Variabili:

Permettono di contenere al loro intero un valore, utile per memorizzare lo stato in un determinato momento, esse supportano ufficialmente tipi interi e float. Il supporto a stringhe e booleani è stato implementato parzialmente e al momento non è possibile utilizzarli in CQ.

5.2.5 Metodi:

i metodi, come indicato dal loro nome, permettono di effettuare chiamate a funzione. La lista di metodi consentiti e utilizzabili all'interno di CQ e passata all'Interpreter attraverso una lista di Scriptable Objects (creabili direttamente dalla sezione project dello Unity Editor), ogni elemento della Collection contiene, oltre alla sua chiamata in CQ, la sua traduzione in codice C#. Questo meccanismo permette di creare nuovi metodi direttamente dall'Inspector ed effettuare il bind tra la chiamata in CQ e rispettiva chiamata in C# con maggiore facilità.

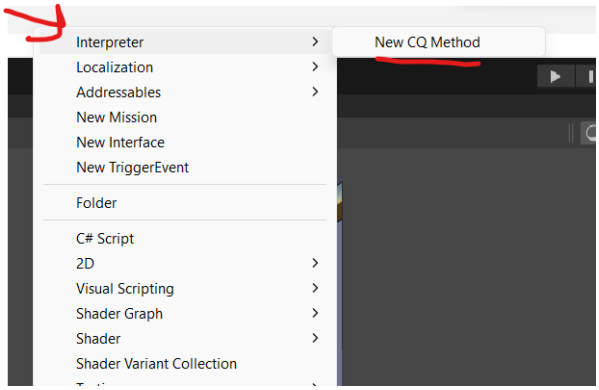


Figura 9 Aggiunta di un nuovo metodo

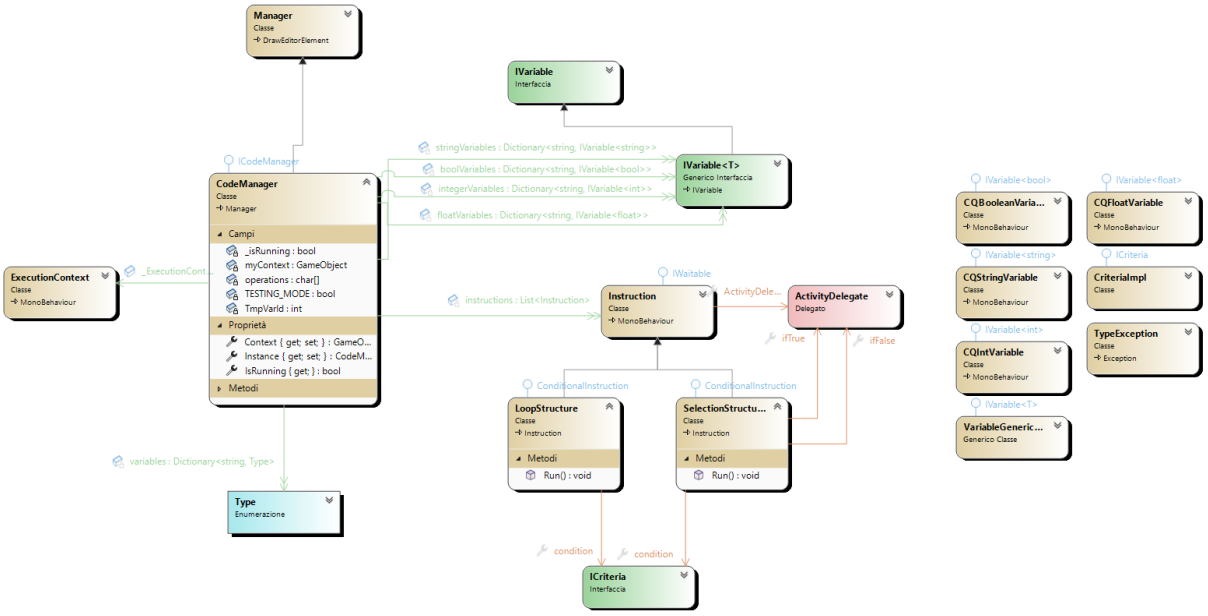


Figura 10 schema di classi per la gestione dell'interprete

5.3.0 L'interprete nel dettaglio

Il codice in CQ deve essere convertito in codice C# riadattato alla struttura di Unity utilizzata all'interno del progetto, per effettuare questa trasformazione ci affidiamo a diversi meccanismi che consentono di trasformare in oggetti i diversi costrutti presenti nel linguaggio di programmazione di CodeQuest
In [Figura 10] è illustrato lo schema delle classi che compongono l'interprete.

5.3.1 Variabili:

Le variabili utilizzate hanno un'interfaccia base *IVariable* e una classe base *VariableGeneric*<T>, che fa da base ai vari tipi, quali *CQIntVariable*, CQFloatVariable, *CQStringVariable* e *CQBooleanVariable*. Le variabili dall'interfaccia *IVariable* espongono i metodi *GetValue*() e *SetValue*() che consentono di scrivere e leggere i valori.
È possibile aggiungere e modificare le variabili nel CodeManager tramite i metodi *AddVariable* e *EditVariable*, utilizzando il parametro *Func*<object> per ritornare il valore di una funzione passata come parametro.

5.3.2 Metodi:

AddFunction permette di gestire chiamate di funzioni che consentono di interagire con oggetti, mondo di gioco e codice stesso.
AddSelectionStructure e *AddLoop* consentono di creare if e loop e necessitano di un parametro condition di tipo ICriteria, che espone il metodo *verify*() per verificare la condizione passata nel costruttore di un oggetto che estende *ICriteria* ossia *CriteriaImpl*.

5.3.3 Esecuzione del codice:

La funzione Run esegue il codice vero e proprio, facendo partire una coroutine [3]che cicla le istruzioni del flusso principale del programma e le aggiunge a un *ExecutionContext* . All'interno di esso c'è ha una coda di *Instruction*. Questa è utilizzata per gestire eventuali istruzioni annidate e in caso di if o loop interni verrà creato in modo ricorsivo un altro *ExecutionContext*. Ogni istruzione ha un *ExecutionContext* che al caso minimo avrà una sola istruzione nella coda.

In conclusione, il sistema di "CQ Interpreter for Unity" consente di programmare in un linguaggio di programmazione personalizzato all'interno di Unity, utilizzando una struttura che segue il ciclo di vita di un game engine utilizzando diversi componenti, come il CQInterpreter, il CodeManager e l'ExecutionContext. Grazie alle funzioni *AddVariable*, *EditVariable*, *AddFunction*, *AddSelectionStructure* e *AddLoop* è possibile creare istruzioni complesse e gestire le istanze di esecuzione come specificato nel linguaggio di partenza.

5.4 Sistema di Markup:

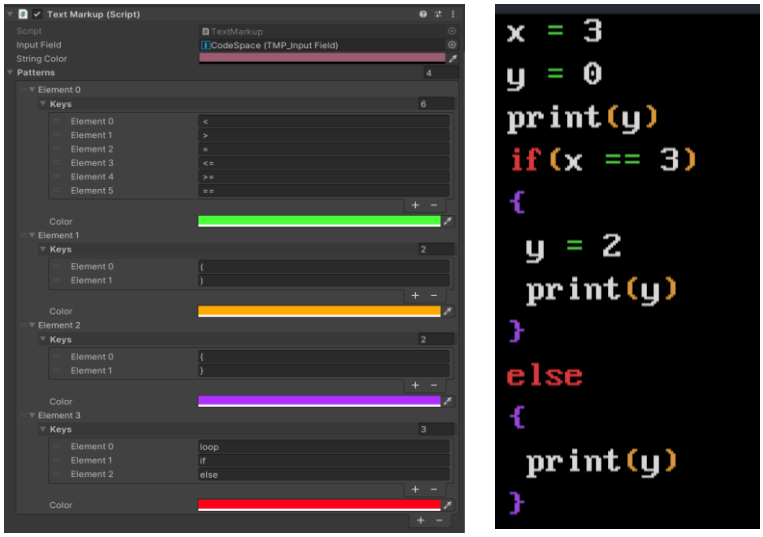


Figura 11 Pagina di configurazione colore a destra, risultato finale

Il Text Markup si occupa di colorare il testo in modo dinamico all'interno di una input field appartenente a Text Mesh Pro di unity.
Per rendere lo script altamente personalizzabile è stato deciso di creare una Struct serializzabile in modo che essa diventasse visibile nell'Ispector se fosse stata associata ad una variabile pubblica o SerializeField.[Figura 12]
Per sfruttare questa caratteristica della struct pattern è stata inserita la lista patterns esposta nell'Ispector in modo da permettere una maggiore personalizzazione dei diversi pattern dal punto di vista dell'editor di unity rendendo il codice maggiormente flessibile e applicabile a diverse situazioni.

```
[Serializable]
 struct Pattern
 {
     public List<string> keys;
     public Color color;
 }
```

Figura 12 Struct Pattern

[3] Coroutine: metodo di unity che non interrompe il thread principale, consentendo di effettuare operazioni asincrone senza bloccare il gioco

Sfortunatamente le Input Field Text Mesh Pro di Unity non sono fatte per essere modificate mentre l'utente ci sta scrivendo al suo interno rendendo imprevedibile il comportamento del Caret. Per ovviare a questa cosa all'interno di Text Markup viene memorizzata la posizione del Caret prima e dopo ogni modifica in modo da gestire internamente la posizione del Caret della input field.
È stato gestito separatamente il caso in cui il giocatore incolli del testo all'interno della input field bloccando il parsing all'interno della input Field fino a quando la procedura di incollo del testo non è terminata.
Il componente Text Markup verrà passato alla classe ExecutorCompiler in modo che quest'ultima possa capire che il testo da cui deve leggere il codice contiene dei tag di Rich text e li rimuova prima di convertire il codice in c# riadattato alla nostra struttura.

```
string code;
if (textMarkup != null)
{
     code = textMarkup.TextWithoutTags(text.text);
}
else
{
     code = text.text;
}
CSharpCode = CQInterpreter.Instance.Convert(code);
```

Figura 13 ExecutorCompiler

5.5 Sistemi di triggering

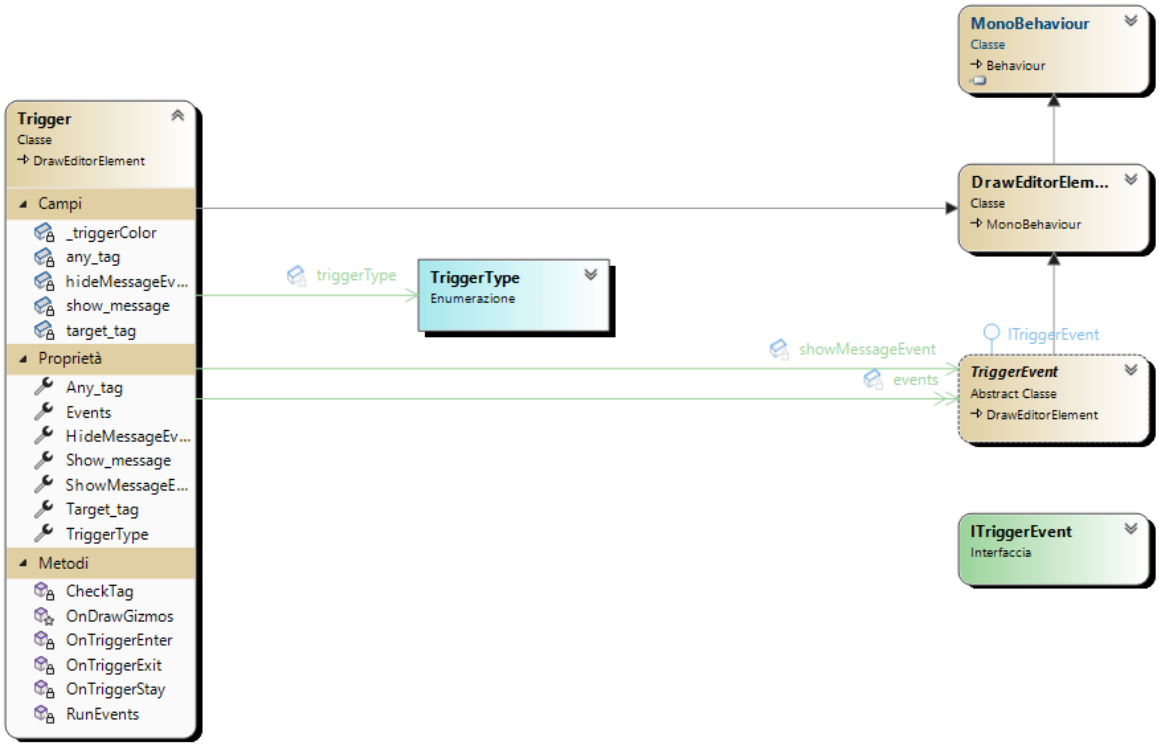


Figura 14 Struttura del pattern trigger sviluppato

Nei videogiochi molti eventi sono scatenati all'arrivo del giocatore in una determinata zona; questo spazio è detto trigger e può variare il comportamento in base al tipo di evento che viene scatenato.
In unity i trigger sono implementati a basso livello, ciò significa che non c'è un supporto ad alto livello per gestirli, ma è lasciato al programmatore la programmazione di tale comportamento. Infatti l'engine ci consente di sapere se l'evento viene scatenato ed esegue quello che specifichiamo all'interno delle callbacks all'attivazione del trigger, ma non dà la possibilità di generalizzare degli eventi da eseguire in modo nativo.

Per rendere più semplice lo sviluppo è stato sviluppato un componente generico che aggiunge un livello che consente di centralizzare queste operazioni in un unico componente e che dia la possibilità di usare classi evento dedicate al trigger.
Tutto questo facilmente configurabile all'interno dell'editor dando così al programmatore solo il compito di programmare gli eventi.

Il sistema utilizza la classe base "*TriggerEvent*" come punto di partenza per la creazione di nuovi eventi di triggering. Questa classe astratta implementa l'interfaccia "*ITriggerEvent*", che definisce il metodo "*Execute*" da implementare nelle classi derivate. La classe "Trigger" rappresenta un'area di triggering e ha un'elenco di eventi di triggering, una lista di tag da monitorare, un booleano per specificare se controllare tutti i tag o solo uno specifico, un booleano per mostrare un messaggio, un enum per specificare il tipo di triggering e due eventi di triggering specifici per mostrare e nascondere il messaggio.

Il metodo "*OnTriggerEnter*", "*OnTriggerExit*" e "*OnTriggerStay*" vengono utilizzati per gestire l'attivazione degli eventi di triggering. In particolare, quando un oggetto entra, esce o sta all'interno dell'area di triggering, vengono eseguiti i relativi eventi di triggering. Nel metodo "*OnTriggerEnter*" viene verificato se l'oggetto in entrata ha uno dei tag specificati e quindi viene eseguito l'evento di triggering corrispondente. Nel metodo "*OnTriggerExit*" viene eseguito l'evento di triggering corrispondente se l'oggetto in uscita ha uno dei tag specificati. Nel metodo "*OnTriggerStay*" viene eseguito l'evento di triggering corrispondente se l'oggetto all'interno dell'area di triggering ha uno dei tag specificati.

Il sistema è estensibile, in quanto è possibile creare nuovi eventi di trigger derivando dalla classe base "*TriggerEvent*" e implementando il metodo "*Execute*". Inoltre, è possibile specificare diversi tipi di triggering utilizzando l'enum "*TriggerType*".

5.6 Sistema di gestione delle missioni

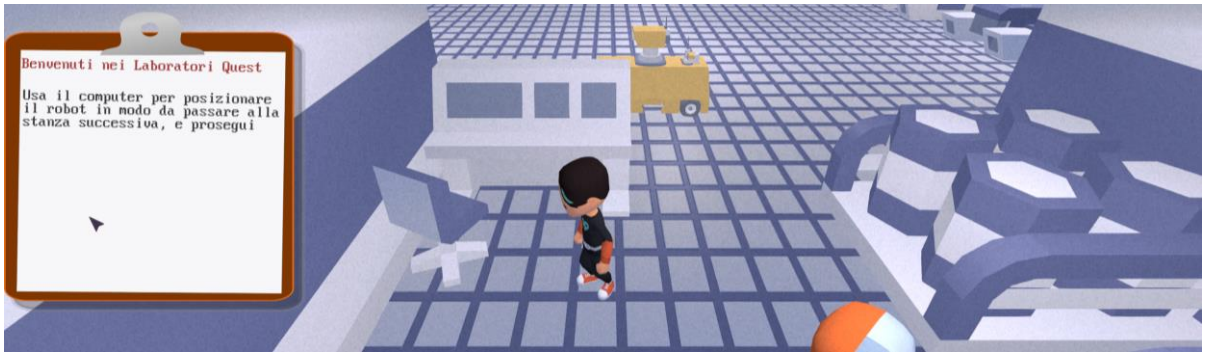


Figura 15 UI che mostra la missione da completare

In ogni videogioco le missioni sono la colonna portante dell'esperienza, poiché danno l'obbiettivo da raggiungere.
In un mondo 3D è possibile avere diverse missioni sparse in giro per la mappa, ed esse devono essere poter gestite da un'unità centrale.
Nel nostro caso le missioni sono indispensabili per poter progredire nei livelli successivi.
Abbiamo sviluppato così un sistema che ci consenta la gestione dei vari task, esso ha una classe base astratta "*Mission*" che implementa l'interfaccia "*IMission*" che impone l'uso di una serie di proprietà e metodi come "*MissionCompleted*", "*MissionStarted*", "*MissionID*", "*CheckMissionCompletion*()", "*StartMission*()" e "*StopMission*()". La classe "Mission" è progettata per essere estesa da altre classi come "*CodingMission*" che aggiunge funzionalità specifiche per gestire missioni che richiedono obiettivi basati sul codice scritto dall'utente.
Il sistema centrale è il "*MissionManager*", una classe che ha il compito di gestire le missioni correnti, mostrare la descrizione della missione attraverso una UI e aggiornare lo stato della missione in base alle azioni del giocatore. Il "*MissionManager*" ha un metodo "*StartNewMission*()" che prende come parametro un'istanza di una classe derivata da "Mission" e lo avvia, aggiornando la UI e lo stato della missione corrente.
Il sistema funziona in modo che quando un giocatore entra in contatto con un trigger, il sistema verifica se la missione è stata avviata o completata. Se la missione non è stata ancora avviata, il "*MissionManager*" inizia una nuova missione attraverso il metodo "*StartNewMission*()". Se la missione è già iniziata, il sistema controlla se la missione è stata completata o meno. Se la missione è stata completata, il sistema aggiorna la UI e lo stato della missione, altrimenti viene verificato se l'obiettivo della missione è stato raggiunto e il "*MissionManager*" aggiorna lo stato della missione di conseguenza.

In conclusione, il sistema di gestione delle missioni è composto da diverse classi e interfacce che lavorano insieme per gestire le missioni del gioco. Utilizzando questa struttura, è possibile creare facilmente nuove missioni e adattarle alle esigenze del gioco.

5.7 Gestione delle impostazioni

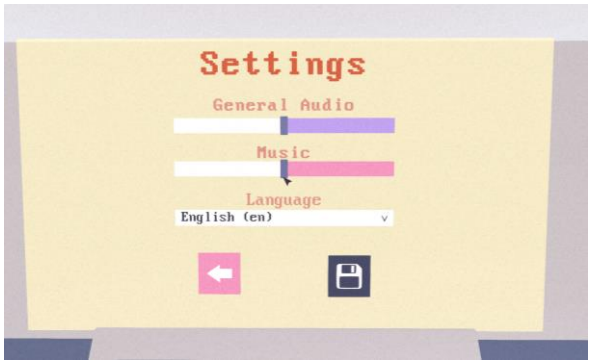


Figura 17 Schermata delle impostazioni

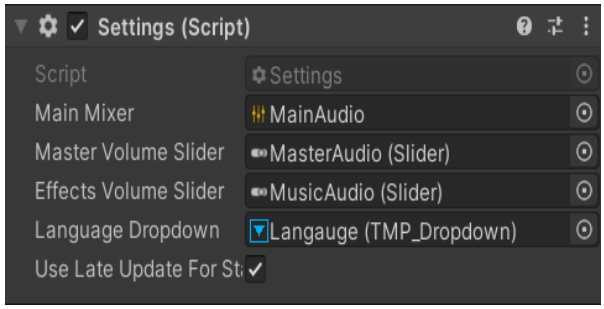


Figura 16 Setup delle impostazioni

La classe *Settings* è una classe Singleton che ha lo scopo di gestire le impostazioni dell'applicazione, come il volume e la lingua.
La classe usa l'AudioMixer di Unity per gestire il volume e la libreria di Localizzazione per controllare la lingua dell'applicazione.

All'avvio del gioco lo script verificherà se sono presenti salvataggi delle impostazioni nei PlayerPrefs. Se è la prima volta che si avvia il gioco verranno impostati per l'audio generale e quello relativo alla musica di gioco dei valori predefiniti.
Viene anche caricata la lingua, chiamando la funzione *LoadLanguage().* Se la variabile *useLateStartForStartLanguage* è impostata su **true**, verrà utilizzato il metodo *LateStart()* per caricare la lingua. La lingua all'avvio della scena ha bisogno di qualche istante in più per caricarsi e il LateStart ci permette di fornire alla Localize il tempo sufficiente per inizializzarsi.

La funzione *LoadLanguage()* recupera tutte le lingue disponibili dalla libreria di Localizzazione e le aggiunge al selettore della lingua. Se nessuna lingua è stata selezionata, verrà impostata la lingua predefinita su "en" (inglese). Successivamente, viene chiamata la funzione *SetLanguage()*, che imposta la lingua corrente dell'applicazione in base alla lingua salvata nella memoria del giocatore.

La funzione *SaveSettings()* viene chiamata quando l'utente cambia le impostazioni dell'applicazione. Salva i valori degli slider nella memoria del giocatore, imposta i valori del volume dell'AudioMixer in base ai valori degli slider, salva la lingua selezionata nella memoria del giocatore e imposta la lingua corrente dell'applicazione in base alla lingua selezionata.

5.8 Gestione del Menu principale

Le classi *MenuStateManager* e *MenuBaseState* sono parte di un sistema di gestione degli stati del menu. Il MenuStateManager è la classe principale che tiene traccia dello stato corrente del menu e gestisce il passaggio da uno stato all'altro.

```
public class MenuBaseState : MonoBehaviour
{
    public UnityEvent OnEnterState;
    public UnityEvent OnExitState;

}
```

Figura 18 Menu Base State

La classe *MenuBaseState* è una classe base astratta che definisce le proprietà e i metodi comuni a tutti gli stati del menu, come gli eventi *OnEnterState* e *OnExitState* che vengono chiamati quando si entra o esce da uno stato.

La classe *PressToStartState* è una sottoclasse di *MenuBaseState* che rappresenta lo stato iniziale del menu, dove si attende che l'utente prema un tasto per passare allo stato successivo. In particolare, il metodo Update controlla se lo stato corrente è *PressToStartState* e se l'utente ha premuto un tasto, in tal caso passa allo stato successivo tramite il metodo *SwitchState* del *MenuStateManager*.

```
public class MenuStateManager : Manager
{

    public static MenuStateManager Instance;
    private void Awake()
    {
        if (Instance == null)
        {
            Instance = this;
        }
        else
        {
            Destroy(gameObject);
        }
    }

    public MenuBaseState currentState;

    // Start is called before the first frame update
    void Start()
    {
        currentState.OnEnterState?.Invoke();
    }

    public void SwitchState(MenuBaseState state)
    {
        currentState.OnExitState?.Invoke();
        currentState = state;
        currentState.OnEnterState?.Invoke();
    }

}
```

Figura 19 Menu State Manager

Il *MenuStateManager* utilizza il pattern Singleton per assicurarsi che esista solo un'istanza dell'oggetto nella scena. Il metodo *SwitchState* cambia lo stato corrente, invocando prima l'evento *OnExitState* dello stato corrente e poi l'evento *OnEnterState* del nuovo stato.
SwitchState può essere chiamato direttamente dagli eventi dell'inspector passando il nuovo stato che dovrà raggiungere il menù.

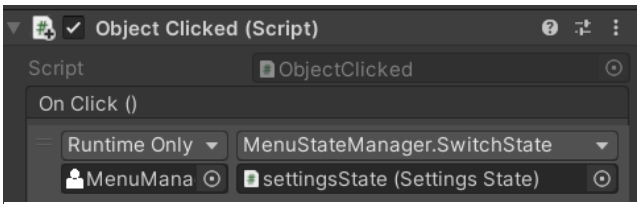


Figura 20 Scene Transition Example

5.9 Integrazione dello Unity Editor

Per velocizzare lo sviluppo e rendere più semplice l'aggiunta di componenti nelle scene sono stati sviluppati alcune estensioni per lo Unity editor.
I plugin comprendono funzionalità di aggiunta di elementi in scena, creazione di script personalizzati e modifica della visualizzazione delle proprietà di alcuni componenti.

Il primo componente che vediamo è lo strumento per modificare la visualizzazione dei trigger.
La classe che gestisce tale funzionalità estende la classe editor, come tutti i plugin che interagiscono con l'editor stesso, e dato l'oggetto trigger andrà a modificare la visualizzazione nell'inspector in base allo stato dell'oggetto che deve modificare. Le modifiche interessano la lista dei tag, che sarà disponibile solo se il flag any tag è false e la funzionalità show message utile quando si vuole mostrare un messaggio sulla gui quando si scatena l'evento settato quando si interagisce con il trigger.

L'altro plugin sviluppato consente la creazione di componenti predefiniti direttamente all'interno della scena attuale. I componenti che si possono creare sono:

- Trigger
- Dialoghi
- Missioni
- Computer

Questi componenti sono tutti componenti che derivano dalla classe trigger, ma hanno già pronti i settaggi necessari per comportarsi come dialoghi e missioni.
Una volta creato l'oggetto esso verrà posizionato nella posizione in cui si trova la telecamera dell'editor nella scena.

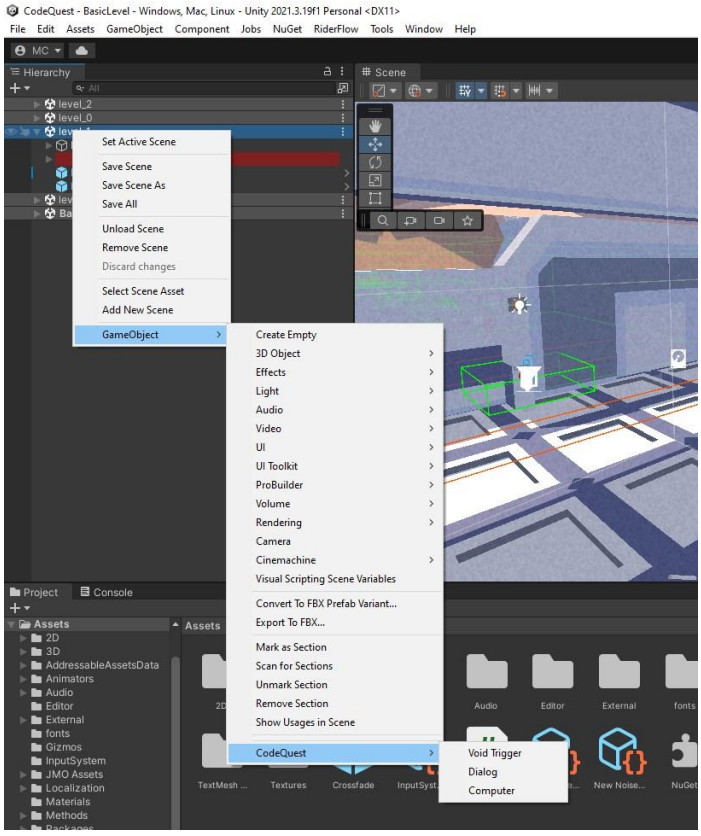


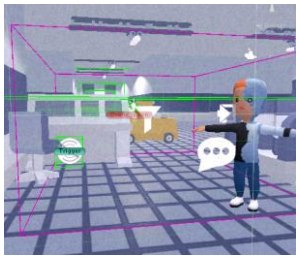
Figura 21 Prefab Creation

La funzionalità di generazione di codice personalizzato non ha richiesto la scrittura di un vero plugin, poiché unity dà già un pattern per poter creare dei template di codice.
Creando la cartella ScriptTemplates, una cartella speciale dell'editor, e inserendo al suo interno dei file di testo con il nome che segue il seguente pattern:
Indice-Nome della funzionalità-Nome del file.cs
Esempio:
03-New TriggerEvent-NewTriggerEvent.cs

Una volta riavviato l'editor potremo crea degli script che seguono la struttura scritta nel file di testo.

Esempio di template:

```
using UnityEngine;
public class #SCRIPTNAME#: TriggerEvent
{
    public override void Execute(Collider other)
    {
        #NOTRIM#
    }
}
```



La macro *#SCRIPTNAME#* prenderà il nome dello script specificato nel pattern, mentre *#NOTRIM#* lascerà uno spazio tra le due parentesi.
Come ultima aggiunta sono state aggiunte dei Gizmos[4], personalizzati per i vari componenti, in modo da essere subito riconoscibili nell'editor.
Questa funzionalità consente anche di rendere molto più facile la gestione di trigger che se non sono selezionati non sono visibili, e quindi difficili da identificare

Figura 22 Gizmos Examples

[4] Gizmos: I Gizmo sono utilizzati per fornire aiuti visivi per il debug o la configurazione nella vista Scena.

5.10 Internazionalizzazione e Localizzazione

Come tutti i videogiochi e software moderni la possibilità di avere più traduzioni del software è sempre apprezzata, nel nostro caso ha un ulteriore punto che ha motivato la sua implementazione, ovvero il campo applicativo.

L'applicazione è destinata alle scuole e non sempre ci sono studenti che hanno capacità linguistiche abbastanza sviluppate per riuscire a comprendere l'inglese ad esempio.
Tramite l'internazionalizzazione abbiamo provveduto a creare le traduzioni in italiano e in inglese, ma sarà possibile aggiungere ulteriori lingue in futuro.
La soluzione utilizzata per applicare questa funzionalità è il framework di internazionalizzazione fornito da unity, che consente di gestire tutte le stringhe e le relative tabelle in modo semplice senza dover modificare stringhe nel codice.

Poiché utilizziamo un framework di internazionalizzazione abbiamo dovuto da subito tenere in considerazione che qualsiasi stringa dovrà essere stampata a schermo dovrà avere la relativa traduzione e quindi non era più sufficiente utilizzare variabili di tipo string, ma LocalizedString.
Questa classe ha i metodi che consentono di recuperare la lingua attualmente selezionata e ritornare una stringa con la corrispondente traduzione.

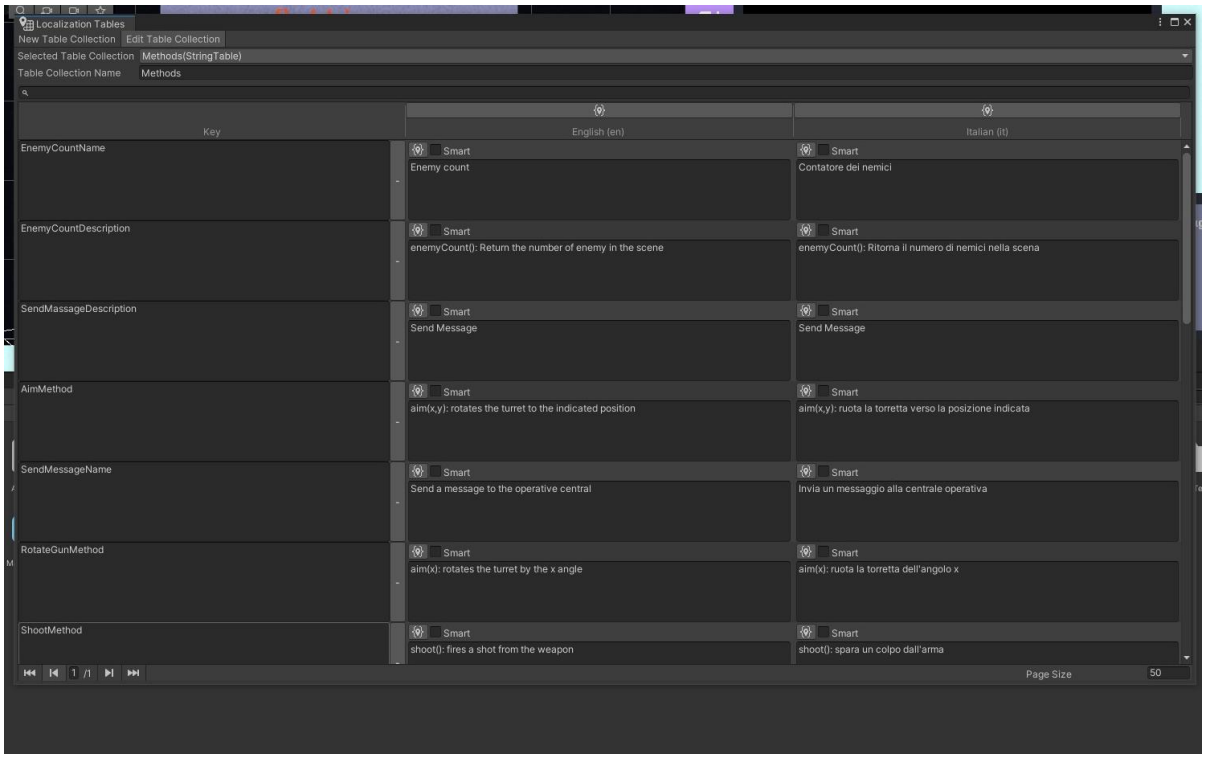


Figura 23 Localization Panel

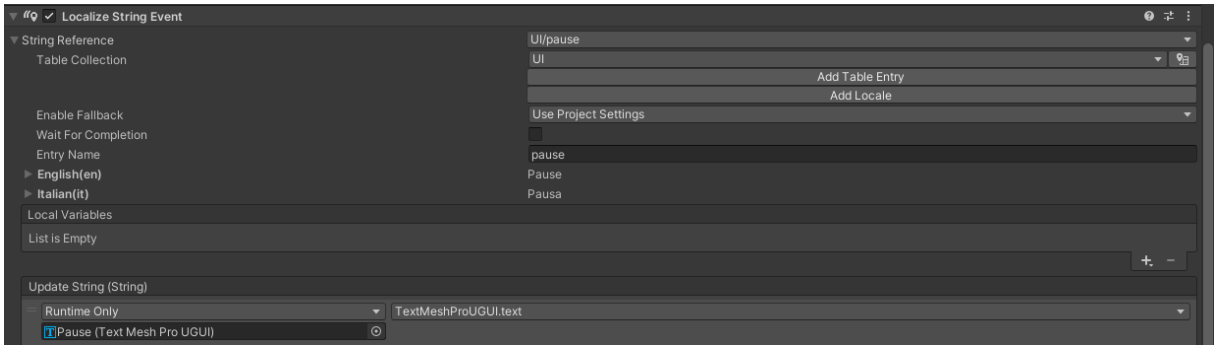


Figura 24 Localize String Bind

Nella [Figura 23] possiamo vedere la struttura della tabella di traduzione. Essa ha una chiave che identifica il record e successivamente le varie colonne nelle varie lingue con le relative traduzioni. Le lingue sono selezionabili dallo sviluppatore, non ci sono lingue preimpostate.
Nella [Figura 24] è mostrato un esempio di configurazione di una stringa, in questo caso di un elemento dell'interfaccia grafica del menu di pausa. Essa richiede il valore chiave e il nome della tabella da cui prendere il dato.
Se si andasse a creare un campo che richiede una stringa che dovrà essere tradotta nelle nuove classi utilizzando LocalizedString come tipo si avrà una configurazione simile a quella rappresentata.

In conclusione, abbiamo ritenuto una funzionalità fondamentale poiché è destinato a studenti di scuole medie e superiori e bisogna tenere in considerazione le disparità linguistiche.

5.11 Audio

Per quanto riguarda la gestione dell'audio il contesto generale del mondo di gioco è gestito da un mixer. Tutte le fonti audio per essere bilanciate con tutto il contesto globale devono passare attraverso questo componente.
Attualmente si compone di 2 canali audio:

- Master
- Music

Il canale master controlla tutte le fonti audio dei sotto canali e degli effetti. Il canale Music controlla tutte quelle sorgenti che generano musiche di sottofondo.
In futuro si potrebbero aggiungere ulteriori canali per gestire al meglio più fonti audio differenti applicando effetti diversi.
Infatti, i mixer di unity danno la possibilità di aggiungere diversi effetti come riverberi, equalizzatori, compressori ecc.

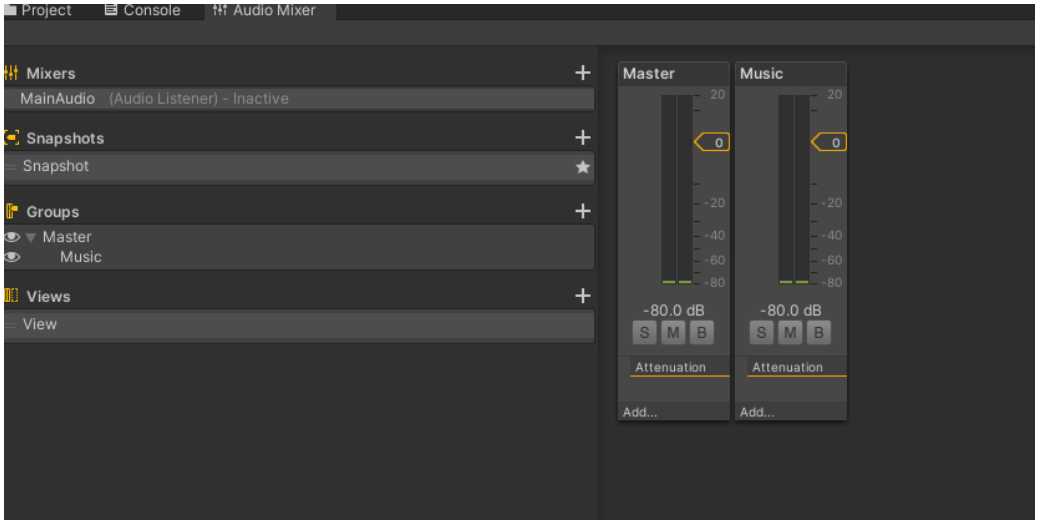


Figura 25 Mixers

La gestione dell'Audio Listener, ovvero il componente che consente di sentire l'audio è gestita dal Camera Manager contenuto nel Player, poiché è l'unità centrale del gioco.
In conclusione, la gestione dell'audio è un aspetto cruciale nella creazione di un'esperienza di gioco coinvolgente e realistica. Il mixer rappresenta il componente principale per bilanciare le fonti audio con il contesto globale, e i mixer di Unity offrono la possibilità di integrare una vasta gamma di effetti audio. Inoltre, l'Audio Listener, gestito dal Camera Manager del Player, consente di percepire l'audio del gioco in modo efficace. Un'adeguata gestione dell'audio può contribuire a creare un'atmosfera coinvolgente e immersiva, aumentando così la qualità e l'esperienza di gioco.

5.12 Rendering e effetti

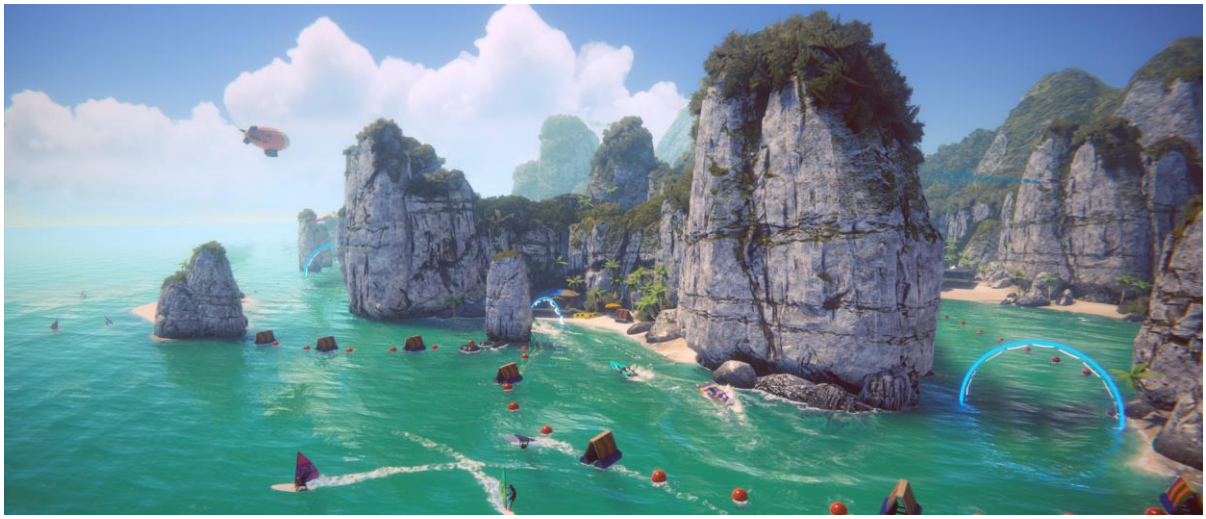


Figura 26 URP Unity

Il rendering del gioco è gestito dal sistema URP, Universal Render Pipeline, un sistema di rendering delle immagini implementato da unity che ha l'obbiettivo di dare un'interfaccia semplice per la gestione degli effetti e del rendering in maniera ottimizzata su diverse piattaforme, dal mobile alle moderne console. [8]

Per quanto riguarda gli effetti, abbiamo creato un oggetto predefinito per mantenere lo stile coerente per tutte le scene.

Esso si trova come prefab sotto il nome di Global Volume e contiene la pipeline di effetti che abbiamo impostato per ottenere il risultato grafico finale.

gli effetti inseriti sono:

- Bilanciamento del bianco
- Correzione Colore sui canali RGB
- Film Grain
- Shadows Midtones Highlights

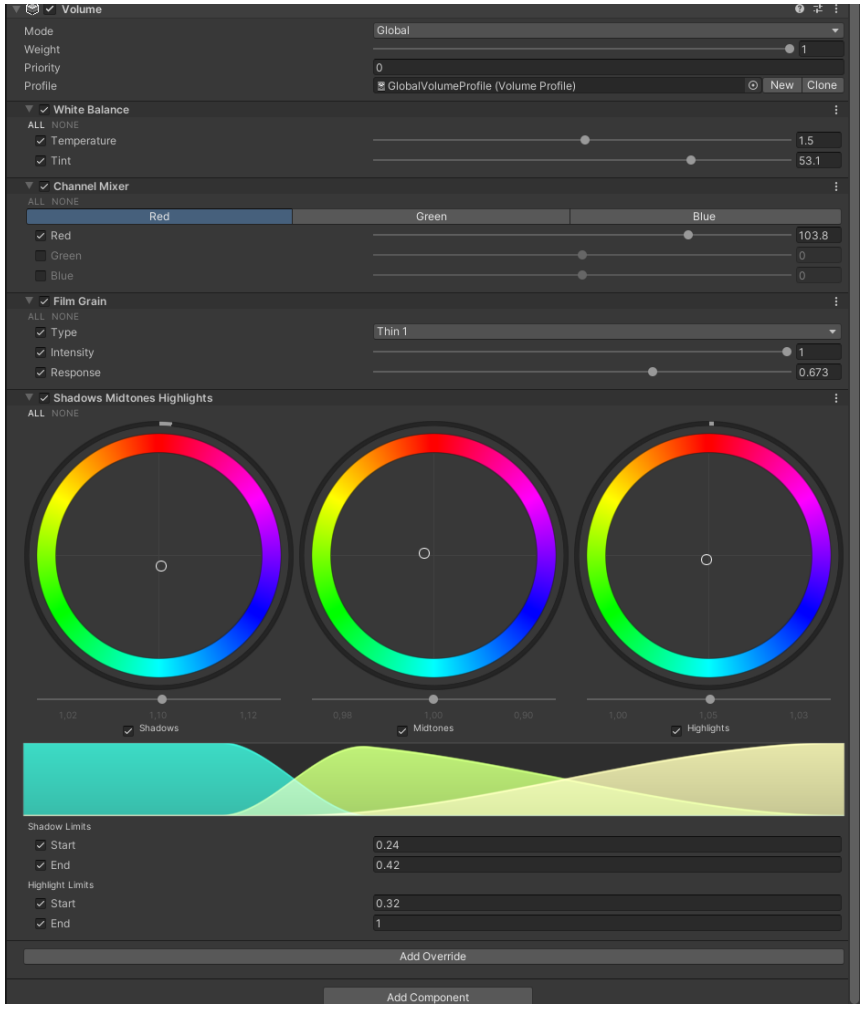


Figura 27 Effetti Volumetrici

Il bilanciamento ha consentito di dare il giusto equilibrio dei colori chiari e della luce, rimuovere sfumature poco realistiche.
La correzione colore ha permesso di gestire il tono di colore generale del rendering 3D, modificando i 3 canali rosso, verde e blu (RGB).
Il Film Grain ha aggiunto l'effetto di rumore visivo, è stato inserito per non rendere piatto il rendering.
Il Shadows Midtones Highlights ci ha consentito di gestire i colori di ombre, mezzitoni e punti luce.

In conclusione, di questo paragrafo possiamo dire che il sistema URP di Unity è fondamentale per la gestione del rendering delle immagini del gioco, poiché permette di ottenere un risultato grafico ottimizzato su diverse piattaforme, mantenendo al tempo stesso una gestione semplice ed efficace degli effetti. In particolare, gli effetti volumetrici inseriti nel gioco, come il bilanciamento del bianco, la correzione colore sui canali RGB, il film grain e lo Shadows Midtones Highlights, hanno permesso di creare un'esperienza visiva coerente e realistica, senza compromettere la fluidità del gioco su diverse piattaforme. L'attenzione ai dettagli e alla coerenza visiva può aumentare l'immersione del giocatore e rendere il gioco ancora più coinvolgente.

5.13 Gestione delle Camere

Le varie telecamere sono gestite dal framework Cinemachine. Questa tecnologia consente di avere telecamere intelligenti e personalizzabili senza dover creare script ad hoc come, ad esempio, per realizzare telecamere sensibili alle collisioni.
Il giocatore principale viene seguito da una telecamera fissa che lo segue nell'ambiente 3D, essa rileva eventuali collisioni che ostacolano la visuale.
Per quanto riguarda la telecamera del robot è una telecamera statica che segue il robot.
La gestione delle telecamere del menu è sempre delegata a Cinemachine grazie al sistema automatico di transizioni che consente di abilitare diverse telecamere una volta raggiunte determinate condizioni, nel nostro caso dei punti di riferimento, waypoint.

5.14 Testing

Per quanto riguarda lo unit testing, in unity è presente un framework dedicato ai test del sorgente.
Per questo progetto è stata testata la parte più critica del progetto, ovvero la ricostruzione nelle istruzioni CQ in istruzioni eseguibili in unity tramite il CodeManager.

Tutto il linguaggio è stato testato compreso di tipi e esecuzione di funzioni. Per quanto riguarda altre funzionalità più legate all'editor e all'engine sono stati effettuati test in game per verificare il funzionamento.
Il testing in unity ha due tipi di funzioni di test i classici test annotati con *[Test]* e i test che interagiscono con il mondo di unity annotati con *[UnityTest],* questi metodi vengono usati per testare la creazione di oggetti, rimozione uso di componenti di fisica ecc.; mentre i test normali vengono usati per test di unità di metodi che non interagiscono con unity, ad esempio l'assegnamento delle variabili nel nostro caso.
In unity è presente un test runner che consente l'esecuzione di tutti i test.

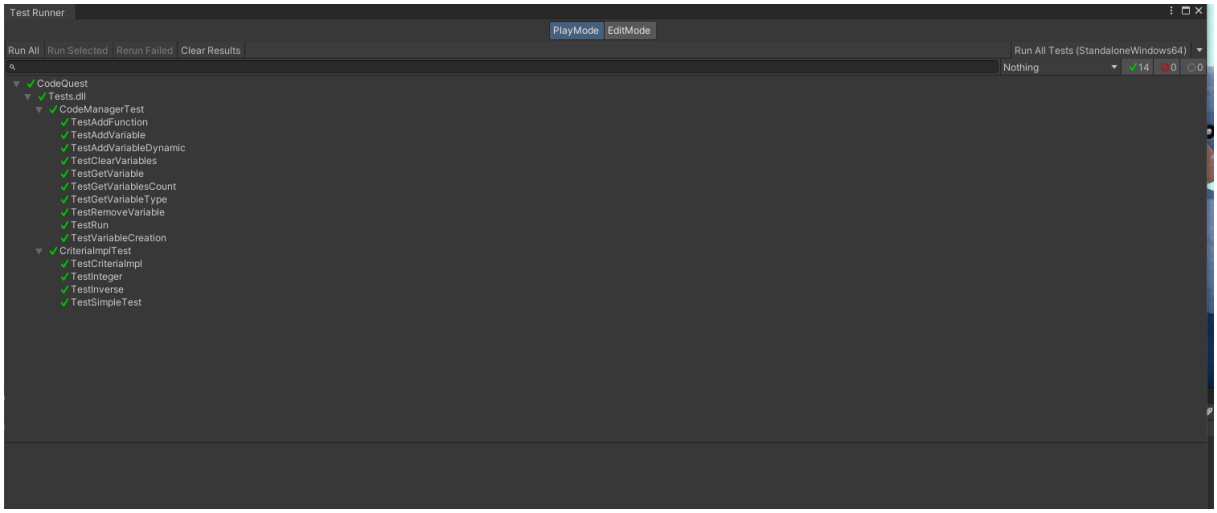


Figura 28 Unit Testing

In conclusione, il codice sorgente del gioco è stato testato sia tramite unit test, che tramite test non automatizzati. Abbiamo ritenuto fondamentale sviluppare test di unità per la parte più critica, il backend che esegue il codice e ne fa la traduzione dal linguaggio CQ, poiché è molto corposa a livello di codice e il suo funzionamento potrebbe essere compromesso in caso di cambiamento; grazie ai test ogni componente dell'infrastruttura può essere verificato in modo semplice.

Conclusioni e Risultati

6.0 Conclusioni

In conclusione, possiamo dire che il progetto soddisfa i requisiti richiesti e dà una base per essere portato avanti con nuovi livelli e modalità.
Molti aspetti legati all'insegnamento possono essere migliorati analizzando i dati raccolti dai test in aula e con l'intervento di docenti e professionisti del settore nel processo di sviluppo dei vari livelli.
Il gioco richiede fin da subito all'utente di adottare un approccio di ragionamento orientato alla programmazione, che lo aiuta a sviluppare le capacità di pensiero logico e di risoluzione dei problemi tipiche di questa disciplina.

6.1 Risultato finale e difficoltà riscontrate

Siamo giunti al termine della prima fase di prototipazione, fornendo un videogioco che include tre livelli di difficoltà crescente. Grazie alla flessibilità del linguaggio CQ, abbiamo creato le basi per realizzare un gioco con molte possibilità nella realizzazione dei livelli, offrendo la possibilità di espandere il gioco con ulteriori funzionalità in futuro.
Il passo successivo consiste nel testare il gioco per valutare le sue funzionalità attuali e identificare eventuali aree di miglioramento. I risultati dei test che verranno condotti nelle aule scolastiche saranno utili per individuare le funzionalità future da implementare e per apportare eventuali migliorie al gioco.
Siamo consapevoli che lo sviluppo dei giochi è un processo continuo e che ci saranno sempre nuove sfide da affrontare, ma siamo molto soddisfatti del risultato raggiunto finora.

Figura 29 Level 2

6.2 Sviluppi futuri

Per quanto riguarda gli sviluppi futuri di Code Quest, l'idea è quella di sviluppare una modalità multigiocatore, ampliando così il campo di abilità da andare a sviluppare grazie alla componente competitiva e cooperativa.
Un ulteriore sviluppo sarà la definizione di una serie di livelli mirati per l'insegnamento effettivo della programmazione.

Bibliografia

Riferimenti

[1] wikipedia, «apprendimento,» 02 05 2023. [Online]. Available: https://it.wikipedia.org/wiki/Apprendimento.

[2] L. H. Y. T. X. L. S. G. X. L. Zehui Zhan, «https://doi.org/10.1016/j.caeai.2022.100096,» 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666920X22000510.

[3] S. &. K. H. &. K. K. Abbasi, «systematic Review of Learning Object Oriented Programming through Serious Games and Programming Approaches,» 2017. [Online].

[4] wikipedia, «Human_Resource_Machine,» 05 05 2023. [Online]. Available: https://en.wikipedia.org/wiki/Human_Resource_Machine.

[5] wikipedia, «lightbot,» 05 05 2023. [Online]. Available: https://en.wikipedia.org/wiki/Lightbot.

[6] wikipedia, «CodeCombat,» 05 05 2023. [Online]. Available: https://en.wikipedia.org/wiki/CodeCombat.

[7] codecombat, «codecombat league,» 05 050 2023. [Online]. Available: https://codecombat.com/league.

[8] Unity, «Unity URP,» 05 05 2023. [Online]. Available: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@16.0/manual/index.html.

[9] it.wikipedia, «wikipedia,» 04 05 2023. [Online]. Available: https://it.wikipedia.org/wiki/Low_poly.

Allegati

AL1 Documento di Game Design