CAPOLAVRO

gIOCO UNITY RELAZIONE

FIORE ENRICO

2024

INDICE

[Progetto 2](#_Toc165559467)

[Inizio lavoro 2](#_Toc165559468)

[Scopo del gioco 2](#_Toc165559469)

[Storyboard del gioco 2](#_Toc165559470)

[Software utilizzato per la costruzione del gioco 2](#_Toc165559471)

[Github del progetto 3](#_Toc165559472)

[Pubblicazione futura 3](#_Toc165559473)

[Assets 3](#_Toc165559474)

[IA 3](#_Toc165559475)

[Sfondo menù 3](#_Toc165559476)

[Nemici 4](#_Toc165559477)

[Creazione livelli 6](#_Toc165559478)

[UNITY 9](#_Toc165559479)

[Schermata 9](#_Toc165559480)

[Hierarchy 9](#_Toc165559481)

[Scene 10](#_Toc165559482)

[Game 10](#_Toc165559483)

[Inspector 11](#_Toc165559484)

[Assets 13](#_Toc165559485)

[Console 14](#_Toc165559486)

[Prefab 15](#_Toc165559487)

[Codice 15](#_Toc165559488)

[Progetto in versione eseguibile (.exe) 17](#_Toc165559489)

[Protagonista 17](#_Toc165559490)

[Interfaccia menu 19](#_Toc165559491)

[Nel gioco: 19](#_Toc165559492)

[Nella pagina iniziale: 19](#_Toc165559493)

[Test 20](#_Toc165559494)

[Audio 20](#_Toc165559495)

[Conclusione 21](#_Toc165559496)

# Progetto

## Inizio lavoro

Lo sviluppo del gioco è iniziato a fine Luglio del 2023 dopo aver partecipato ad un corso di preparazione/ orientamento all’università di Camerino.

## Scopo del gioco

Il videogioco creato fa parte della categoria platform, i livelli sono costituiti da piattaforme 2D in cui il giocatore potrà camminare per esplorare il livello.

Lo scopo di ogni livello è quello di riuscire ad arrivare da un punto iniziale “A” a un punto finale “B” per proseguire nella storia.

In ogni livello sono presenti ostacoli di vario genere come: nemici, puzzle e ambientale.

Andando avanti nel gioco, i livelli diventeranno sempre più complicati.

Il giocatore quando dovrà combattere contro i nemici avrà a disposizione 3 cuori che non potranno essere curati finche non muore, a quel punto sarà resettata la vita e ricomincerà dall’ultimo checkpoint.

## Storyboard del gioco

La storia del gioco parla di un ragazzo (di cui non si sa il nome) che ha perso un suo caro amico (Marco) dopo aver partecipato ai funerali dell’amico esso parte per un viaggio di ritorno a casa, ma questo viaggio si mostrerà essere in realtà viaggio spirituale per il protagonista e riuscire così a superare il trauma della morte del suo amico.

Il gioco ha avuto una pre-costruzione della storia e dei livelli, creando con precisione i punti che si alternano tra livello e filmato.

All’interno di Unity è stata creata una scena apposita per provare le novità aggiunte all’interno del gioco prima che esse vengano aggiunte all’interno dei livelli così da evitare casini durante la creazione di pre-fabrricati con errori.

## Software utilizzato per la costruzione del gioco

Come software per costruire il mio gioco ho utilizzato Unity; ho deciso di utilizzare Unity per un gioco 2D rispetto ad un Godot (software specializzato per creare giochi in 2D) perché più familiare col software.

Come linguaggio di programmazione è stato quindi utilizzato C# con le importazione della libreria Unity Engine.

## Github del progetto

Su Github è stata aggiunta la demo del gioco.

Link: [GitHub - ENRICOSCHI/DemoGiocoTHEUNFORGOTTEN: Repository con demo del gioco platform attualmente in sviluppo](https://github.com/ENRICOSCHI/DemoGiocoTHEUNFORGOTTEN)

## Pubblicazione futura

In futuro il gioco sarà caricato nella piattaforma Itch.io (link: [Download the latest indie games - itch.io](https://itch.io/)), la piattaforma permette di pubblicare giochi (per lo più indie) per farli conoscere al pubblico.

# Assets

Per la creazione dell’ambiente di gioco e dei personaggi (sprites) ho utilizzato il sito Itch.io da cui è possibile scaricare (anche gratis) assets pre-fabrricati da importare all’interno del gioco.

Link per le pagine da cui ho preso alcuni assets:

Player: <https://rvros.itch.io/animated-pixel-hero>

Skeleton: <https://jesse-m.itch.io/skeleton-pack>

Slime: <https://rvros.itch.io/pixel-art-animated-slime>

## IA

### Sfondo menù

Il background nella schermata del menu principale è stata fatta tramite l’utilizzo della IA e più precisamente con Midjourney.



Spiegazione: Immagine creata con Midjourney per lo sfondo del menu.

### Nemici

Nel gioco ho creato 2 tipi differenti di nemici: scheletro e slime.

#### Scheletro

Lo scheletro è un personaggio poco più grande del protagonista e rappresenta la morte all’interno della storia, esso infatti si vede poco nei primi livelli ma andando avanti con la storia la paura del giocatore per la morte diventa talmente grande che lo scheletro diventa ogni presente nella scena.

Le caratteristiche dello scheletro sono quattro:

1. Egli si muoverà solo da un punto “A” a un punto “B” quando il protagonista entra nell’area d’azione dello scheletro, egli lo attaccherà.
2. I danni inflitti dallo scheletro tolgono al giocatore 2 cuori su 3.
3. Lo scheletro ha 5 di vita (un colpo del protagonista toglie 1).
4. Lo scheletro si muove lentamente ma ha un attacco veloce.

Parte di codice dello scheletro per attacare il protaginsta (player) quando entra nella zona d’attacco:

private void OnTriggerEnter2D(Collider2D other)

{

// Verifica se l'oggetto che ha attivato il trigger ha un componente Transform

if (other.GetComponent<Transform>() != null)

{

if (other.CompareTag("Player"))

{

// Ottieni la posizione dell'oggetto che ha attivato il trigger

Vector3 otherPosition = other.transform.position;

// Ottieni la posizione del trigger

Vector3 triggerPosition = transform.position;

// Confronta le posizioni X per determinare se other è a sinistra o a destra del trigger

if (otherPosition.x > triggerPosition.x)

{

//destra

IsInside = true;

AttackChecker.SetActive(true);

spriteRenderer.flipX = false;

// Posiziona a destra del personaggio

AttackChecker.transform.position = new Vector3(transform.position.x + xOffset, transform.position.y, transform.position.z);

UpdateAnimationState();

}

else if (otherPosition.x < triggerPosition.x)

{

//sinistra

IsInside = true;

AttackChecker.SetActive(true);

spriteRenderer.flipX = true;

// Posiziona a sinistra del personaggio

AttackChecker.transform.position = new Vector3(transform.position.x - xOffset, transform.position.y, transform.position.z);

UpdateAnimationState();

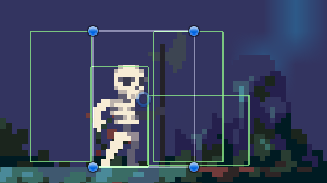
}

UpdateAnimationState();

}

}

}



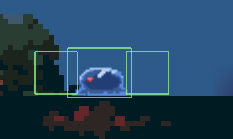
Spiegazione: differenti box collider per rilevare il giocatore all’interno dell’area.

#### Slime

Lo slime è un personaggio molto piccolo rispetto al protagonista e rappresenta la sua paura di rimanere solo, questa paura è talmente grande che fin dai primi livelli si mostra come la più presenta rispetto allo scheletro.

Le caratteristiche dello slime sono tre:

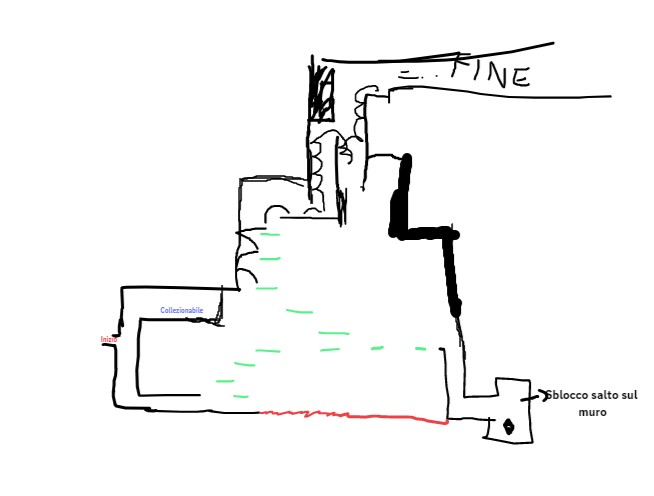
1. Egli si muoverà solo da un punto “A” a un punto “B” quando il protagonista entra nell’area d’azione dello slime, egli lo attaccherà.
2. I danni inflitti dallo slime tolgono al giocatore 1 cuore su 3.
3. Lo slime ha 2 di vita.
4. Lo slime si muove velocemente ma ha un attacco lento.

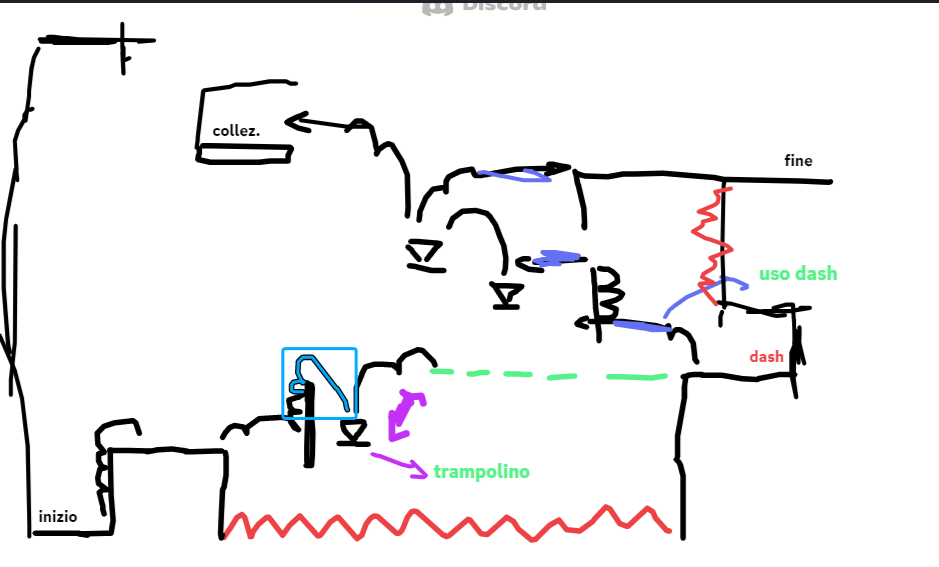


Spiegazione: box collider per rilevare la presenza del giocatore all’interno dell’area.

# Creazione livelli

I livelli sono stati creati facendo una schematizzazione iniziale del livello in file paint, per poi costruire il livello in Unity.

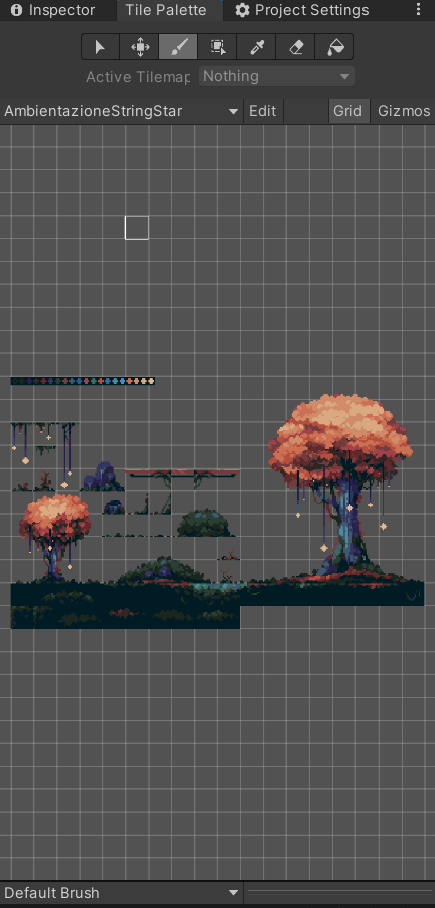






Spiegazione: Schema dei livelli 4-5-8.

In Unity per “disegnare” il livello è stato utilizzato il Tile Palette, funzione in Unity che permette di dividere un’imaggine in piccoli quadretti così da poter fare i livelli più precisi e variegati.



Spiegazione: schermata del Tile Palette.

# UNITY

## Schermata

La schermata di Unity è divisa in 6 parti (principali):

Hierarchy,

Scene,

Game,

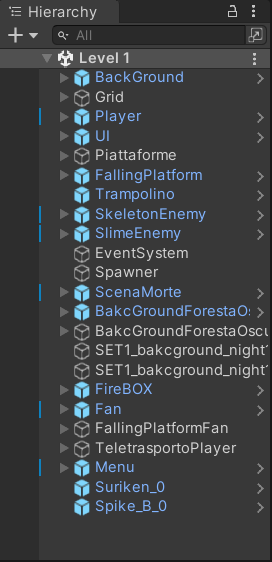
Inspector,

Assets,

Console.

### Hierarchy

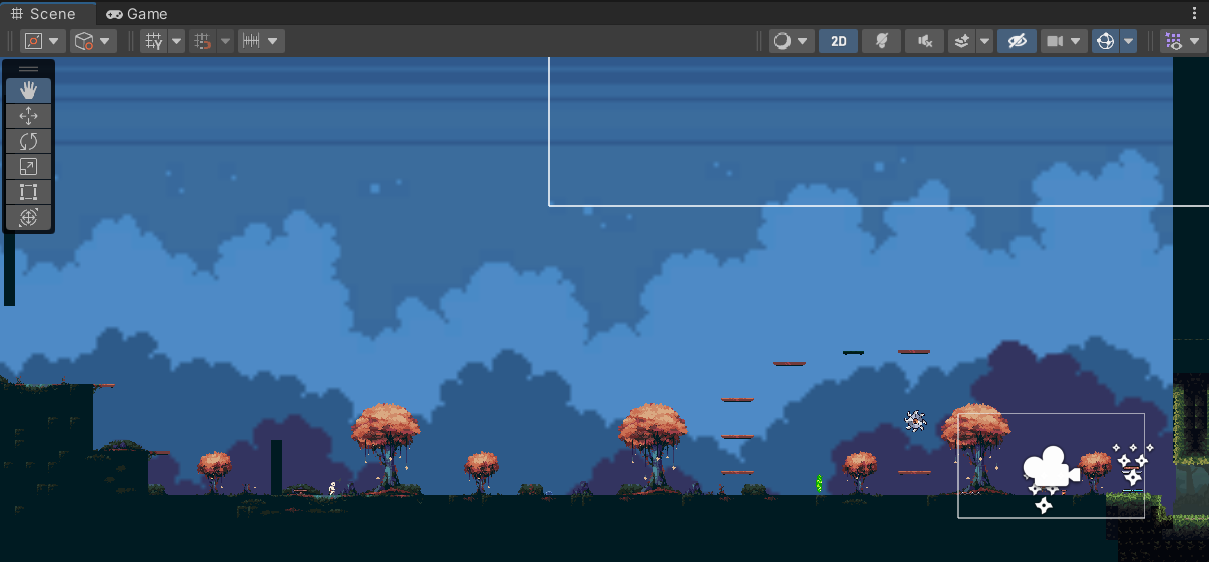
Nella Hierarchy sono presenti gli oggetti che verrano poi visualizzati nella schermata Scene.



Spiegazione: Hierarchy in Unity.

### Scene

Nella Scene saranno visibili gli oggetti presenti nella Hierarchy e sarà possibile spotarli/modificarli a proprio piacimento per creare l’ambinete di gioco.

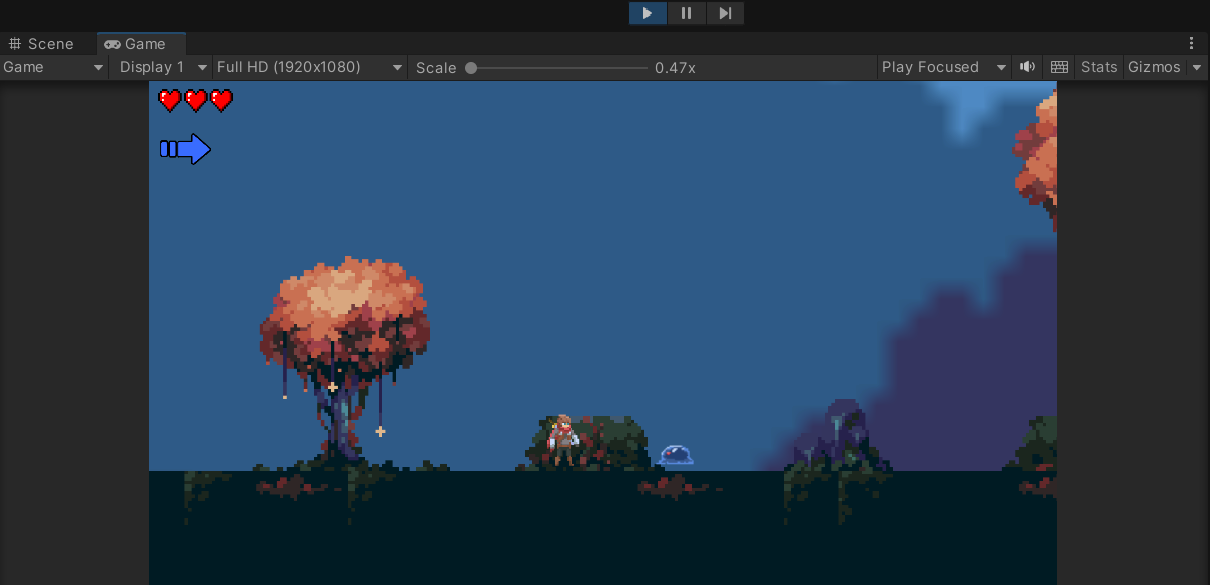


Spiegazione: Scene in Unity.

### Game

Nella finestra Game è possibile provare le modifiche apportate nella Scene e quindi poter provare di fatto il gioco.

Per avviare il gioco bisognerà cliccare sul tasto play in alto nella schermata di Unity.

96

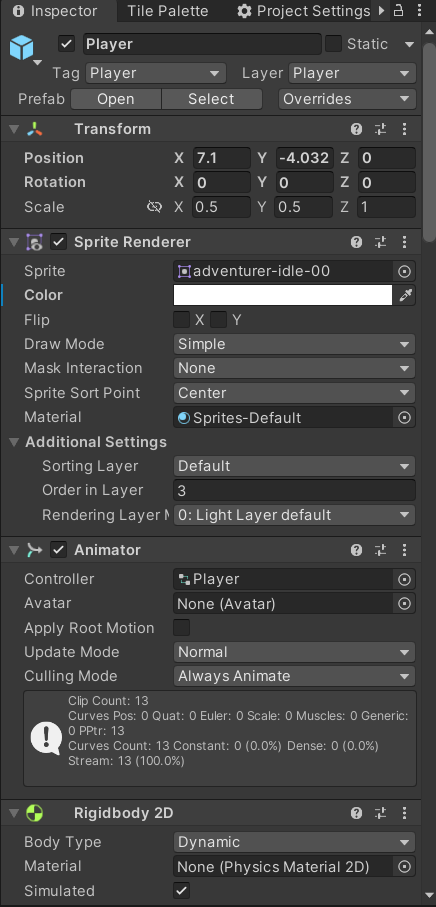
Spiegazione: finestra Game in esecuzione in Unity.

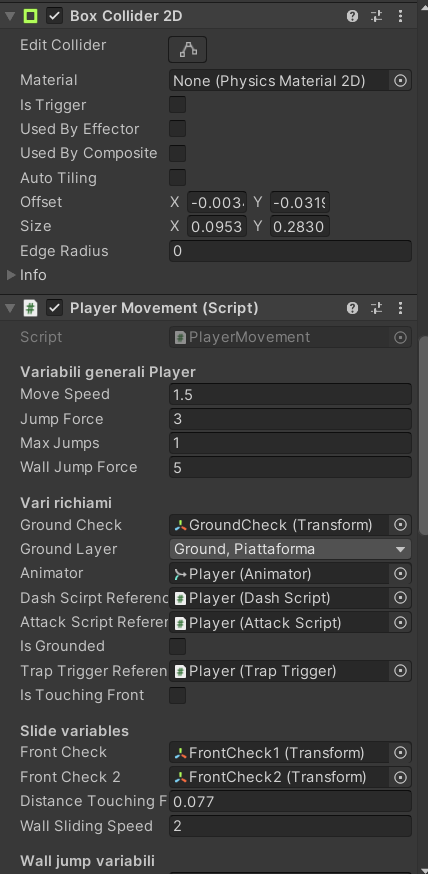
### Inspector

Nell’Inspector è possibile vedere tutte le modifiche applicabili all’oggetto selezionato nella Hierarchy.

Nell’Inspector è possibile aggiungere gli Script (codice c#) creato in Visual Studio e assegnare così quel codice all’oggetto selezionato.

Oltre al codice è anche possibile aggiungere componenti aggiungitivi nel codice come: box collider, rigidbody ecc.. che andranno ad aggiungere senza il bisogno di crearlo tramite codice, delle feature aggiuntive all’oggetto.

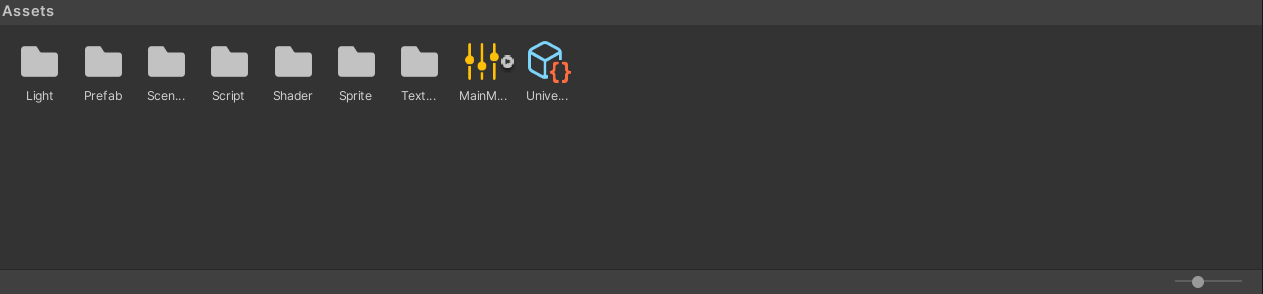




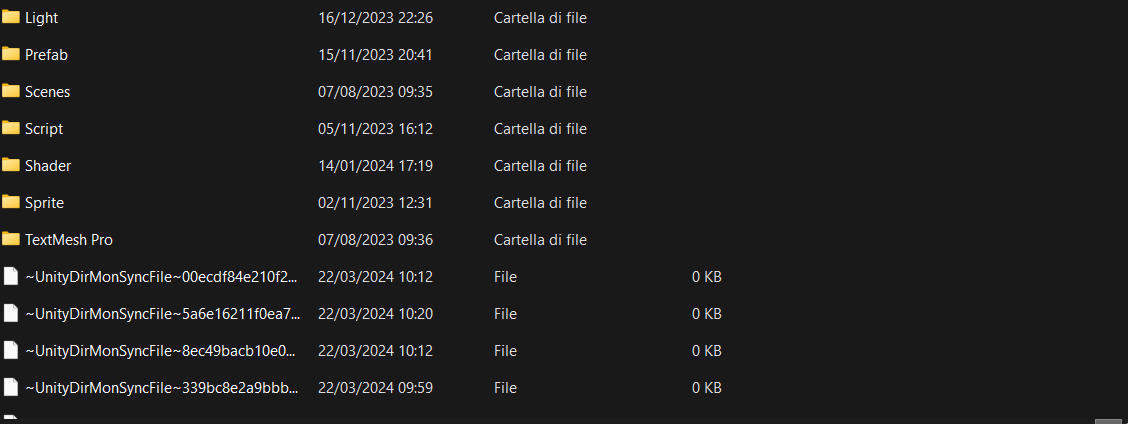
Spiegazione: parte dell’Inspector per l’oggetto Player.

### Assets

Nell’assets sono presenti le cartelle salvato all’interno della cartella Assets che sta a sua volta all’interno della cartella del gioco.



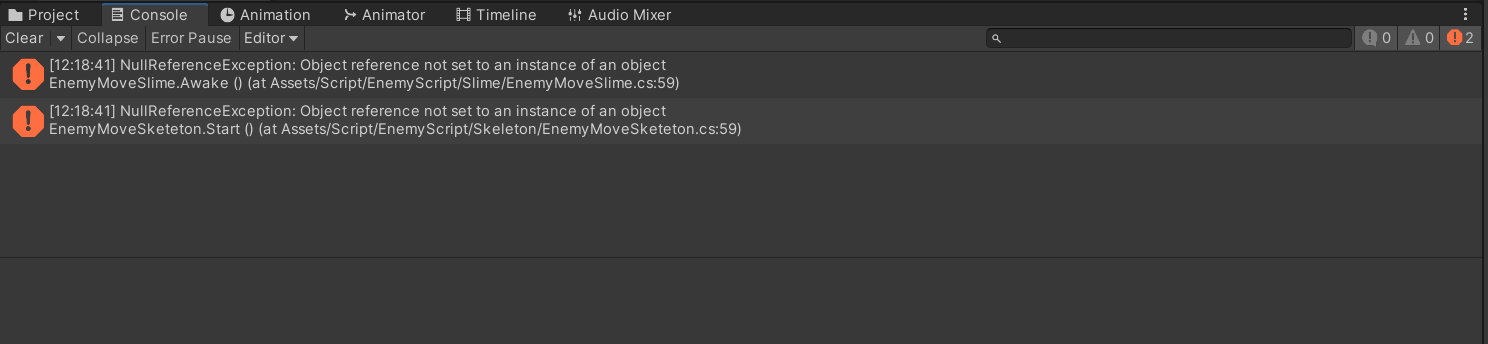
Spiegazione: cartella Asstets all’interno di Unity.



Spiegazione: cartella Assets nella cartella del gioco.

### Console

Nella Console sono presenti gli errori e avvertenze che sono presenti all’interno dell’Editor di Unity o di uno Script appena modificato.

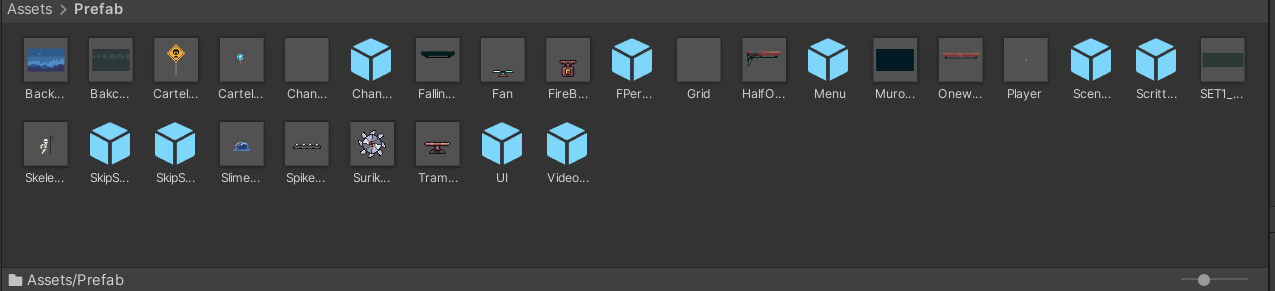


Spiegazione: Console con errori all’interno di Unity.

### Prefab

I prefab (o prefabricati) in Unity sono oggetti che vengono creati all’interno di una scena e poi siccome tale oggetto deve essere presente in altre scene, esso verrà salvato all’interno della cartella assets.

Questi oggetti sono ben visibili nella Hierarchy perché evidenziati in blu.



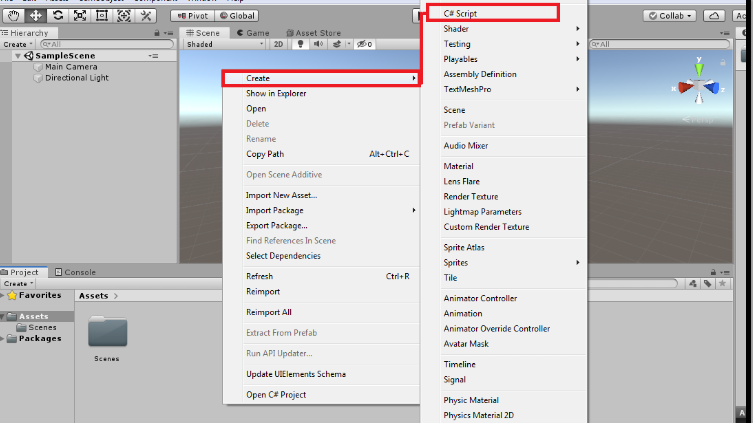
Spiegazione: cartella prefab all’interno di assets.



Spiegazione: Evidenziati in blu sono i “prefabbricati” all’interno della Scene.

## Codice

In Unity gli script vengono creati tramite Visual Studio e usano come linguaggio di programmazione C#.

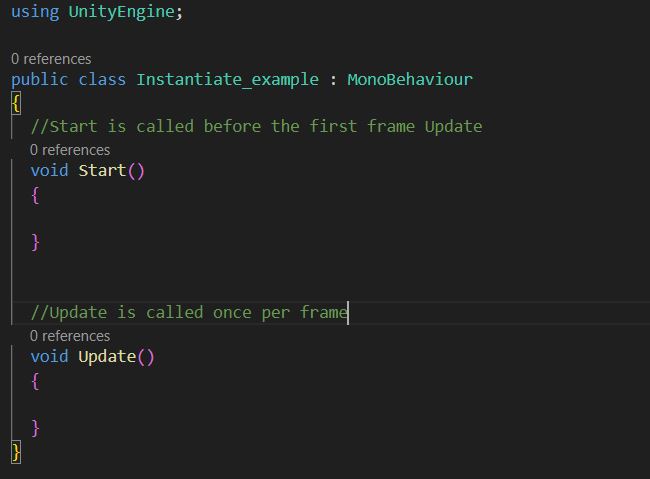


Spiegazione: Creazione script C#

Una volta creato lo script, verrà aperto Visual Studio con già alcune linee di codice scritte in predefinito.

All’apertura del codice abbiamo a disposizione 2 funzioni base Start e Update

* Start: nella funzione Start bisognerà scrivere tutte quelle azioni che devono essere svolte al frame 1 (quindi all’inizio della scena).
* Update: nella funzione Update bisognerà scrivere tutte quelle azioni che essere svolte in loop all’interno della scena.
* Awake: la funzione Awake non è predefinito nel codice alla creazione ma è una funzione predefinita di Unity che ti da la possibilità di fare quello scritto nel codice nel frame 0 (prima dello Start).



Spiegazione: Script appena creato.

## Progetto in versione eseguibile (.exe)

Per provare il proprio progetto in formato eseguibile bisognare “costruire” il progetto, per fare ciò bisognerà andare su (file 🡪 build settings 🡪 build).

All’interno della cartella del progetto verrà creata una nuova cartella con all’interno il file eseguibile (.exe).

# Protagonista

Il protagonista della storia è un personaggio senza nome.

Durante l’avanzamento della storia, con il personaggio si potranno sbloccare 4 abilità: doppio salto, salto dal muro, dash e combattimento.

1. Doppio salto: il doppio salto è un’abilità che permette al protagonista di saltare una seconda volta quando è ancora in aria.
2. Salto dal muro: il salto dal muro permetterà il protagonista di saltare anche dalle pareti permettendogli di raggiungere nuovi punti nel livello.
3. Dash: il dash è uno scatto che il protagonista fa verso una direzione predesignata.
4. Combattimento: il combattimento permetterà al giocatore di sconfiggere i nemici presenti nel livello che lo ostacolano.

Parte di codice per gestire il doppio salto:

// Gestione del double jump

if (isGrounded)

{

jumpsRemaining = maxJumps; // Resetta il numero di salti disponibili quando il personaggio tocca terra

isJumping = false;

}

Parte di codice per gestire il salto dal muro:

if (wallSliding == true && Input.GetButtonDown("Jump"))

{

IsWallJumping = true;

Invoke("SetWallJumpingFalse", wallJumpTime);

Parte di codice per gestire il dash:

isDashing = true;

canDash = false;

isCooldown = true; // Attiva il cooldown

tr.emitting = true;

\_dashingDir = new Vector2(Input.GetAxisRaw("Horizontal"), Input.GetAxisRaw("Vertical"));

if (\_dashingDir == Vector2.zero)

{

\_dashingDir = new Vector2(transform.localScale.x, 0);

}

animDashUI.SetBool("Load", true);

StartCoroutine(StopDashing());

Invoke("DashCooldown", \_dashCooldown);

Parte di codice per gestire il combattimento:

void Update()

{

// Incrementa il timer se il giocatore non ha attaccato

if (canAttack)

{

timeSinceLastAttack += Time.deltaTime;

}

// Check per l'attacco

if (Input.GetKeyDown(KeyCode.E) && canAttack && sr.isGrounded/\* && !IsMoving\*/)

{

AttackChecker.SetActive(true);

rb.constraints = RigidbodyConstraints2D.FreezePositionX | RigidbodyConstraints2D.FreezePositionY;

currentAttackCounter++;

canAttack = false;

timeSinceLastAttack = 0f;

}

// Resetta il contatore se il limite di tempo � stato superato o il giocatore inizia a muoversi

if (timeSinceLastAttack >= resetTimeThreshold /\*|| Input.GetAxis("Horizontal") != 0\*/ || currentAttackCounter>=4)

{

currentAttackCounter = 0;

canAttack = true;

}

/\*// Controlla se il giocatore smette di muoversi

if (Input.GetAxis("Horizontal") ==0)

{

IsMoving = false;

}

else

{

IsMoving = true;n

}\*/

}

# Interfaccia menu

## Nel gioco:

Nel gioco è presente un piccolo menù che se attivato mette in pausa il gioco.

Il menù al suo interno ha 3 opzioni: salva, torna al menu e esci.

1. Tramite il pulsante “Salva” il gioco potrà essere salvato così da poter ricaricare l’ultima posizione quando si ricarica la parita.
2. Tramite il pulsante “Torna al menu” sarà possibile tornare al menù, attenzione però perché il gioco non verrà salvato.
3. Tramite il pulsante “Esci” il gioco verrà chiuso senza salvare.

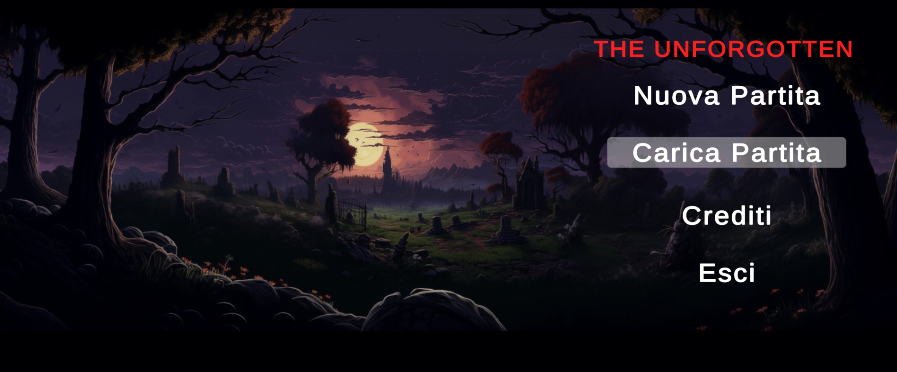


Spiegazione: interfaccia Menù.

## Nella pagina iniziale:

Appena si entra nel gioco ci appare il menu iniziale da cui è possibile scegliere varie opzioni: nuova partita, carica partita, crediti e esci.

* Nuova partita: facendo nuova partita verrano cancellati i file di salvataggio e si potrà iniziare da 0 la storia.
* Carica partita: schiacciando il bottone “carica partita” il gioco caricherà i file di salvataggio presenti dall’ultima sessione. Non sarà possibile cliccare il bottone se non vi sono file di salvataggio, esso sarà quindi rappresentato con un coloro grigio.
* Crediti: con questo bottone sarà possibile vedere i crediti del gioco.
* Esci: con questo bottone il gioco verrà chiuso senza salvare.



Spiegazione: Menu iniziale del gioco.

# Test

Il gioco durante il suo sviluppo ha bisogno di svariati test per riuscire a capire se il videogiocatore riesca effettivamente a capire come giocare il gioco e se sono presenti bug che durante lo sviluppo non sono usciti fuori.

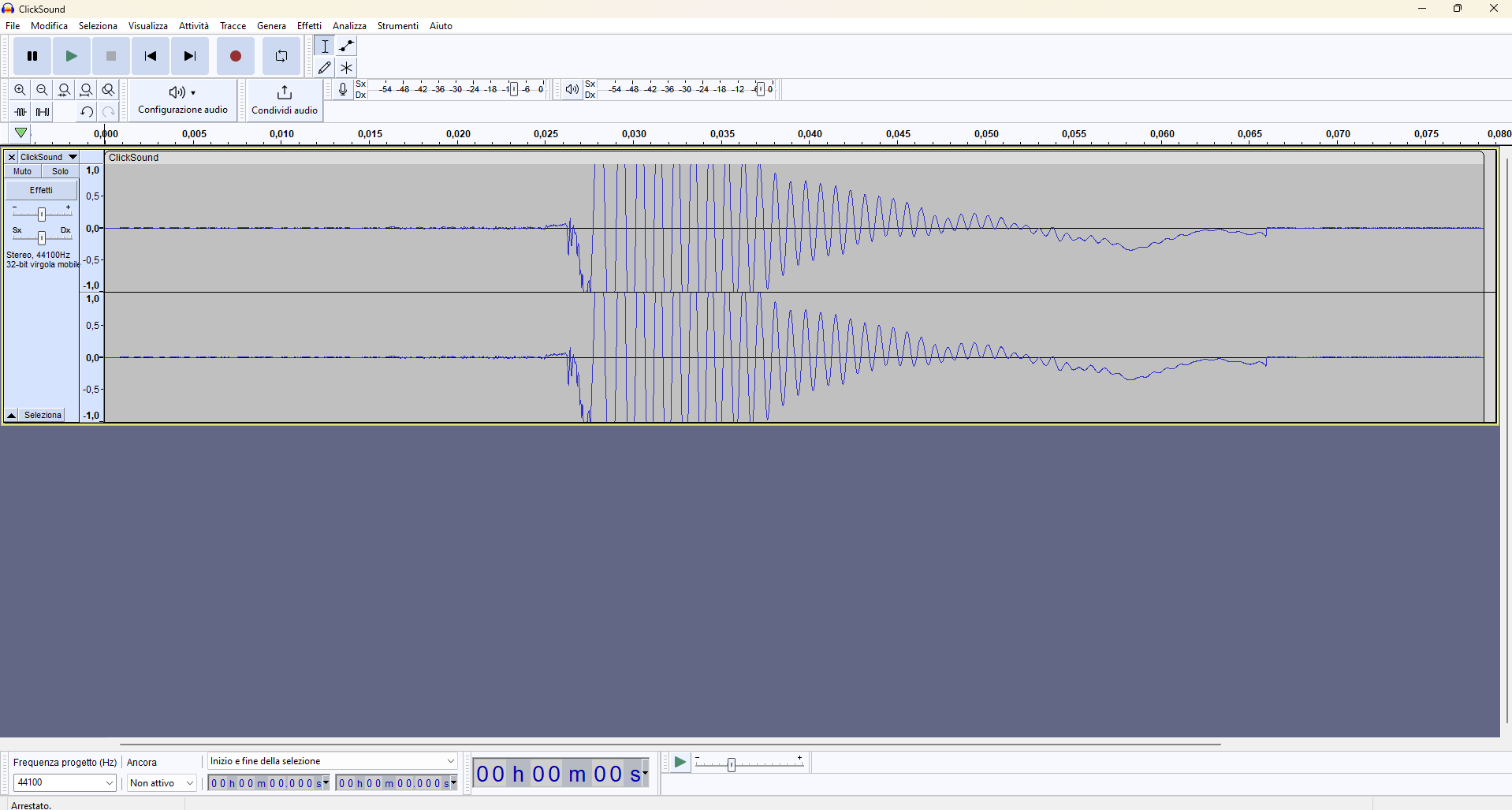
Il gioco è stato fatto provare da diversi range di età e di esperienza coi videogiochi, è stato notato che per chi non aveva molta esperienza con i videogiochi incotrava più difficolta a terminare i livelli sia per la sua poca esperienza che per la difficolta tecnica che si pone andato avanti coi livelli.

# Audio

Nel gioco saranno inseritti effetti sonori e tracce audio per accompagnare il giocatore nell’eseprienza videoludica.

Alcuni effetti sono stati registrati da me stesso tramite l’utilizzo di Audacity (applicazione per personalizzare l’audio).

Questo campo è ancora in fase lavorativa, in quanto sto affrontando diversi problemi con l’esecuzione degli effetti sonori.



Spiegazione: Schermata Audacity.

# Conclusione

Il gioco è ancora in lavorazione, ma è già possibile giocare a una demo con all’interno i primi 5 livelli (di tutorial) giocabili.

Il gioco al momento è arrivato al livello 12 di 23, i restanti 11 livelli sono stati programmati per esser finiti entro la fine dell’anno 2024.

L’esperienza per ora guadagnata durante lo sviluppo autonomo del gioco mi ha fatto capire di quanto ci sia bisogno di una pianificazione maggiore e più dettagliata e di testare più accuratamente (in una scena apposita) il corretto funzionamento di ciò che era stato pianificato.

Una volta aver organizzato il tutto e aver già creato un codice/ambiente stabile e flessibile in Unity, è possibile passare alla creazione vera e proprio del gioco.

Lo sviluppo di questo codice mi ha inoltre aiutato a orientare il mio piano di studi una volta finite le scuole superiori, ho quindi deciso di andare all’UNIVPM e frequentare il corso di Game Design.