

Biomes Run

Titolo del progetto:

Alunno/a:

Classe:

Anno scolastico:

Docente responsabile:

Biomes Run
Simone Riva, Nathan Chiarani
Info 3BC
2022 / 2023
Guido Montalbetti

Sommario

1	Introduzione.....	5
1.1	Informazioni sul progetto.....	5
1.2	Abstract	5
1.3	Scopo	5
2	Analisi.....	6
2.1	Analisi del dominio	6
2.2	Analisi e specifica dei requisiti.....	6
2.3	Use case	10
2.4	Pianificazione.....	11
2.5	Analisi dei mezzi	12
2.5.1	Software	12
2.5.2	Hardware	12
3	Progettazione.....	13
3.1	Design dell'architettura del sistema	13
3.2	Design dei dati e database.....	14
3.3	Design delle interfacce.....	15
3.4	Design procedurale.....	22
4	Implementazione.....	23
4.1	Costruzione labirinto	23
4.2	Costruzione 3 livelli in Unity.....	25
4.3	Decorazione Labirinto	26
4.3.1	Modelli 3D.....	26
4.3.2	Decorazione labirinto	27
4.3.3	Albero cartelle.....	29
4.3.4	Script Rotatore.....	29
4.4	Prima persona	30
4.4.1	Script telecamera	31
4.4.2	Script prima persona personaggio	33
4.5	Vite personaggio	35
4.6	Programmazione mostro	37
4.7	Programmazione schermate	38
4.7.1	Schermata 0 (Benvenuto)	38
4.7.2	Schermata 1 Descrizione Livello 1.....	39
4.7.3	Schermata 2 Gioco Livello 1.....	39
4.7.4	Schermata 3 (Secondo livello).....	40
4.7.5	Schermata 4 Gioco Livello 2.....	40
4.7.6	Schermata 5 (Livello3)	41
4.7.7	Schermata 6 Gioco Livello 3.....	41
4.7.8	Schermata 7 (Vittoria)	42
4.7.9	Schermata 8 (Chiave falsa)	42
4.7.10	Schermata 9 (Vite finite).....	43
4.7.11	Schermata 10 (Pausa)	44
4.7.12	Script gestione Schermate	46
4.8	Script Gemme livelli	47
4.9	Script Gestione apertura porte livelli	50
4.10	Script gestione velocità stelle	52
5	Test	53
5.1	Protocollo di test	53
5.2	Risultati test	56
5.3	Mancanze/limitazioni conosciute	57
6	Consuntivo.....	58
7	Conclusioni	59
7.1	Sviluppi futuri	59
7.2	Considerazioni personali	59
8	Glossario	59

9 Bibliografia.....	60
9.1 Sitografia	60

Indice delle figure

Figura 1 - Use Case	10
Figura 2 - Gantt Iniziale	11
Figura 3 - Diagramma delle classi	13
Figura 4 - Tabella classifica	14
Figura 5 - Schermata iniziale	15
Figura 6 - Schermata di pausa	16
Figura 7 - Schermata di vittoria	17
Figura 8 - Livello uno	18
Figura 9 - Livello due	18
Figura 10 - Livello tre	19
Figura 11 - Schermata perdita chiave falsa	20
Figura 12 - Schermata perdita vite	21
Figura 13 - Diagramma UML	22
Figura 14 - Empty Object	23
Figura 15 - Costruzione muri	23
Figura 16 - Inizio costruzione labirinto	24
Figura 17 - Layout livello 3	24
Figura 18 - Layout livello 2	24
Figura 19 - Layout livello 1	24
Figura 20 - Gerarchia modelli	26
Figura 21 - Modelli 3D	26
Figura 22 - Decorazione livello1	27
Figura 23 - Decorazione livello 2	27
Figura 24 - Decorazione livello3	28
Figura 25 - Ordine oggetti	29
Figura 26 - Script rotatore oggetti	29
Figura 27 - Chiave con script rotazione	30
Figura 28 - Pavimento Grezzo	30
Figura 29 - Personaggio Grezzo	31
Figura 30 - Settaggio posizione camera	31
Figura 31 - movimento telecamera	32
Figura 32 - Script completo movimento + funzionalità aggiuntive	34
Figura 33 - Inserimento immagine vite	35
Figura 34 - Script vite	35
Figura 35 - Inserimento immagini nell'array	36
Figura 36 - Bake del mostro	37
Figura 37 - tag del mostro	37
Figura 38 - script mostro	37
Figura 39 - Ordine scene	38
Figura 40 - Schermata 0 (Benvenuto)	38
Figura 41 - controllo nome	38
Figura 42 - Schermata 1 (Descrizione Livello 1)	39
Figura 43 - Schermata 2 (Schermata Gioco Livello 1)	39
Figura 44 - Schermata 3 (Livello 2)	40
Figura 45 - Schermata 4 (Schermata Gioco Livello 2)	40
Figura 46 - Schermata 5 (Livello 3)	41
Figura 47 - Schermata 6 (Gioco Livello 3)	41
Figura 48 - Schermata 7 (Vittoria)	42
Figura 49 - Schermata 8 (Chiave falsa)	42
Figura 50 - Schermata 9 (Vite finite)	43
Figura 51 - Schermata 10 (Pausa)	44
Figura 52 - Script di gestione del volume	44
Figura 53 - Script di gestione della schermata di pausa	45
Figura 54 - Script gestione schermate	46
Figura 55 - Script gemme livello1	47
Figura 56 - Script gemme livello2	48

Figura 57 - Script gemme livello3	49
Figura 58 - Script porte livello1	50
Figura 59 - Script porte livello2	50
Figura 60 - Script porte livello3	51
Figura 61 - Script gestione stelle	52
Figura 62 - Gantt Consuntivo	58

1 Introduzione

1.1 Informazioni sul progetto

For this project, we decided to make a game programmed with Unity.

This project was born because we like programming video games and we wanted to program one with Unity.

At the moment there are already games similar to the one we would like to create, but we would like to create a completely different version from those already present online.

in short: this work is mainly focused on entertaining users thanks to our passion for programming video games.

1.2 Abstract

For this project, basic knowledge of Unity will be needed.

Following the requirements set by us, for this project it was essential to organize ourselves to divide the various parts of the project in order to be able to restrict the time and stay with the deliveries.

Various objects will be implemented in the game, each with a specific task, which will lead the player to think while still having fun.

The game offers various levels, each with different characteristics that will not make the player bored during his gaming experience.

Results

This game is made from scratch with a graphics engine (Unity). Not having followed a real module related to it but a simple two-day course, it will be a new thing for us too.

1.3 Scopo

Lo scopo del progetto è principalmente riuscire a far divertire il maggior numero di persone con il nostro videogioco perché ci piace far divertire la gente, inoltre abbiamo deciso questo progetto per poter aumentare le nostre conoscenze con il linguaggio C# usando il motore grafico Unity.

2 Analisi

2.1 Analisi del dominio

Il gioco verrà utilizzato principalmente dai ragazzi, essendo un gioco progettato appositamente per loro, perché il nostro gioco ha come scopo di uscire da un labirinto nel minor tempo possibile.

Inoltre consente ai ragazzi di liberare la mente e svagarsi restando comunque attivi mentalmente attraverso un gioco tranquillo e di ragionamento.

2.2 Analisi e specifica dei requisiti

ID: REQ-01	
Nome	Costruzione labirinto
Priorità	1
Versione	1.0
Note	Costruire la base del labirinto
Sotto requisiti	
001	Analisi e progettazione completate

ID: REQ-02	
Nome	Prima persona
Priorità	2
Versione	1.0
Note	La visuale dell'utente sarà in prima persona
Sotto requisiti	
001	Il personaggio deve essere completato

ID: REQ-03	
Nome	Movimento personaggio
Priorità	2
Versione	1.0
Note	L'utente deve avere la possibilità di spostarsi all'interno del labirinto
Sotto requisiti	
001	Il layout del gioco deve essere completato in maniera grezza (Livelli)
002	Il personaggio deve essere completato

ID: REQ-04	
Nome	Decorazione labirinto
Priorità	2
Versione	1.0
Note	Decorare il labirinto grezzo, far sì che ci siano i 3 biomi dei 3 livelli ed aggiungere delle canzoncine spaventose e posizionare tutti gli oggetti 3d
Sotto requisiti	
001	Labirinto grezzo completo

ID: REQ-05	
Nome	Programmazione mostri
Priorità	2
Versione	1.0
Note	Nei livelli ci saranno dei mostri che cercheranno di fermare il giocatore
Sotto requisiti	
001	Il layout livelli deve essere completato
002	Il personaggio deve essere completato

ID: REQ-06	
-------------------	--

Nome	Vite personaggio
Priorità	2
Versione	1.0
Note	L'utente dispone di tre vite durante il gioco
Sotto requisiti	
001	I mostri devono essere completati
002	Il personaggio deve essere completato

ID: REQ-07	
Nome	Funzionamento schermata iniziale
Priorità	3
Versione	1.0
Note	Rendere funzionante la schermata iniziale
Sotto requisiti	
001	Design schermata iniziale completato
002	Funzionamento gioco completo

ID: REQ-08	
Nome	Funzionamento schermata impostazioni
Priorità	3
Versione	1.0
Note	Rendere funzionante la schermata iniziale
Sotto requisiti	
001	Design schermata impostazioni completato
002	Funzionamento gioco completo

ID: REQ-09

Nome	Database
Priorità	3
Versione	1.0
Note	Creare un database il quale contiene i migliori tempi dei giocatori ed i loro nickname
Sotto requisiti	
001	Design schermata vincita e perdita
002	Funzionamento gioco completo
003	Funzionamento schermata impostazioni e iniziale

ID: REQ-10	
Nome	Funzionamento schermata vincita (classifica)
Priorità	3
Versione	1.0
Note	Quando il giocatore vince la partita compare la schermata di vincita la quale contiene una classifica dei migliori tempi (DB)
Sotto requisiti	
001	Database funzionante
002	Gioco completo

2.3 Use case

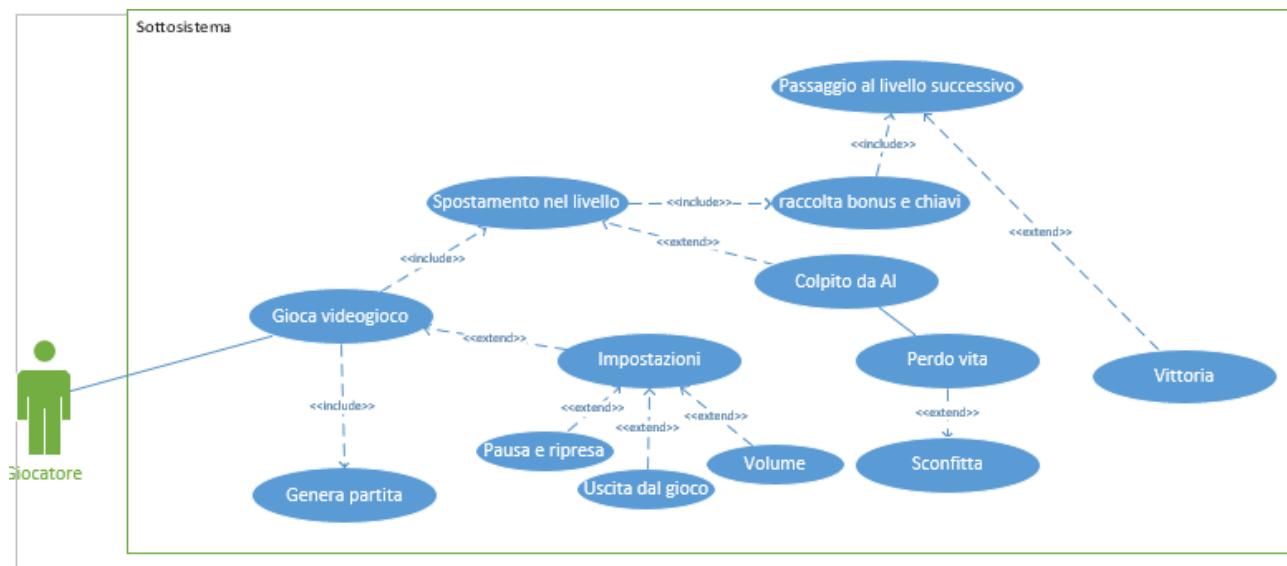


Figura 1 - Use Case

All'interno dell'applicativo c'è solo un utente che è il giocatore:

- Esso può giocare al videogioco, ha come scopo quello di scappare dal labirinto nel minor tempo possibile prendendo tutti gli oggetti e le chiavi.
- Giocando al videogioco esso genera una nuova partita.
- Esso può utilizzare la schermata di impostazioni contenenti:
 - Regolazione volume.
 - Pausa e ripresa durante il gioco.
 - Tasto per uscire dal gioco

2.4 Pianificazione

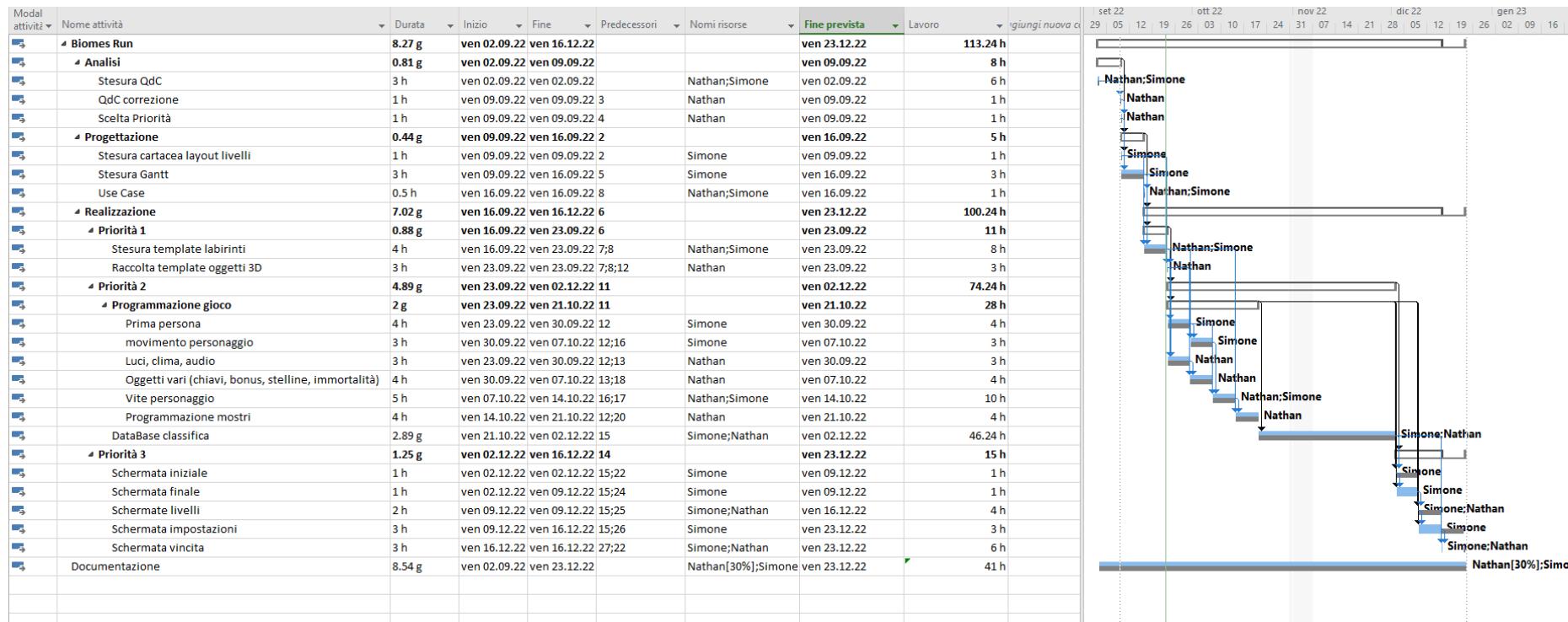


Figura 2 - Gantt Iniziale

La durata del nostro progetto, come si può vedere nell'immagine qui sopra, dura una settimana in meno rispetto alla data di consegna. Questo è perché ho deciso di inserire un margine di errore nel caso dovesse avere problemi oppure dovesse essere indietro rispetto al Gantt. Nel mio Gantt sono riuscito a ricavare varie attività con 29 righe da svolgere, inoltre ho ricavato 3 macrocategorie con 3 sottocategorie che sono:

1. **Analisi:** All'interno di questa macrocategoria ci sono le parti contenenti la pianificazione teorica del progetto con il Quaderno dei compiti e la sua relativa correzione e la scelta delle varie priorità delle parti del progetto.
2. **Progettazione:** All'interno di questa macrocategoria c'è l'inizio della vera e propria progettazione del progetto utilizzando un Diagramma di Gantt ed una Use Case, inoltre abbiamo iniziato a stendere i layout in maniera cartacea dei vari livelli.
3. **Realizzazione:** Questa parte riguarda la vera e propria realizzazione del progetto, l'ho suddiviso in tre categorie:
 - a. **Priorità 1:** In questa categoria c'è l'inizio della realizzazione dei livelli creandoli in Unity ed iniziando a cercare i vari oggetti per la loro decorazione.
 - b. **Priorità 2:** In questa categoria andremo a creare il DataBase per la classifica ed è presente una sottocategoria per la programmazione vera e propria del gioco.
 - i. **Programmazione gioco:** in questa micro-categoria sono presenti tutte le parti per la programmazione del videogioco, dalla prima persona, alla programmazione dei mostri rendendo il gioco quasi completo.
 - c. **Priorità 3:** In questa categoria sono presenti tutte le schermate relative al gioco che sono:
 - i. **Schermata benvenuto:** Questa schermata è la schermata iniziale che da il benvenuto nel gioco all'utente
 - ii. **Schermata impostazioni:** Questa schermata è la schermata che permette all'utente di mettere il gioco in pausa, inoltre permette all'utente di regolare il volume, di uscire dal gioco e di riprenderlo.
 - iii. **Schermata livelli:** Queste schermate appaiono all'inizio di ogni livello, esse comprendono una breve descrizione del livello e di quello che si deve fare per passarlo, inoltre comprende anche una descrizione dei pericoli che aspettano l'utente nello svolgersi del livello.
 - iv. **Schermata vincita:** Questa schermata è la schermata finale, quando l'utente riesce a scappare dall'ultimo livello appare questa schermata contenente le congratulazioni per la vittoria dell'utente.

2.5 Analisi dei mezzi

2.5.1 Software

- **Visual studio 2022**
- **Unity 2022.1**

2.5.2 Hardware

- 2 PC identici: Le componenti dei PC che verranno utilizzati per lo sviluppo dell'applicativo sono:
 - I7-9700 @ 3.00GHz
 - RAM 32GB
 - SSD 512GB
 - NVIDIA GeForce RTX 2060

3 Progettazione

3.1 Design dell'architettura del sistema

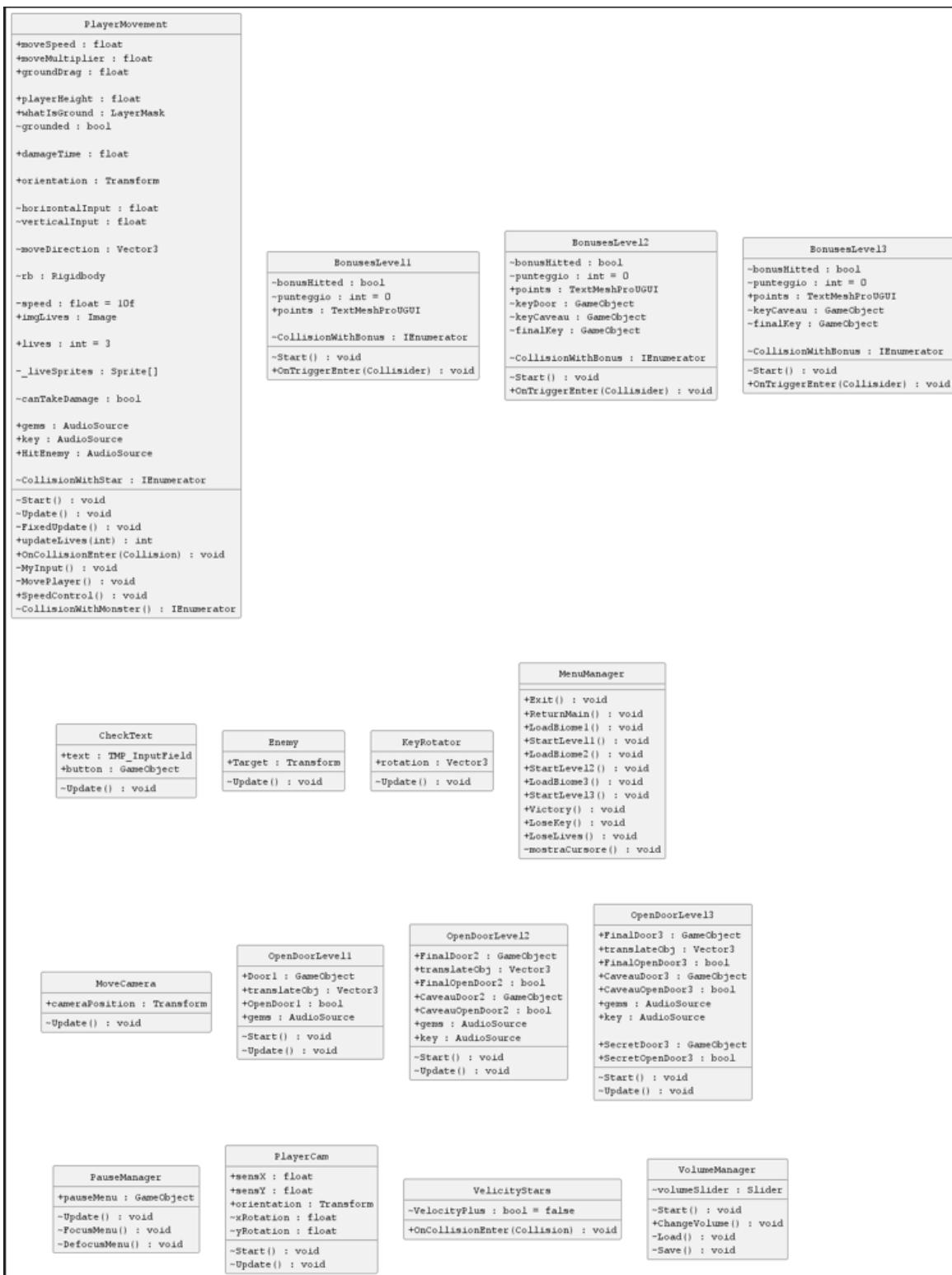


Figura 3 - Diagramma delle classi

3.2 Design dei dati e database

LeaderBoard	
ID	int
nickname	varchar(45)
time	int
place	int

ID -> È una chiave AUTO_INCREMENT con la quale distinguiamo i player
nickname -> È il nome utente del giocatore
time -> È il tempo che ha impiegato il giocatore per finire i 3 livelli
place -> È il posizionamento fatto ordinato in base al time (desc)

Figura 4 - Tabella classifica

3.3 Design delle interfacce

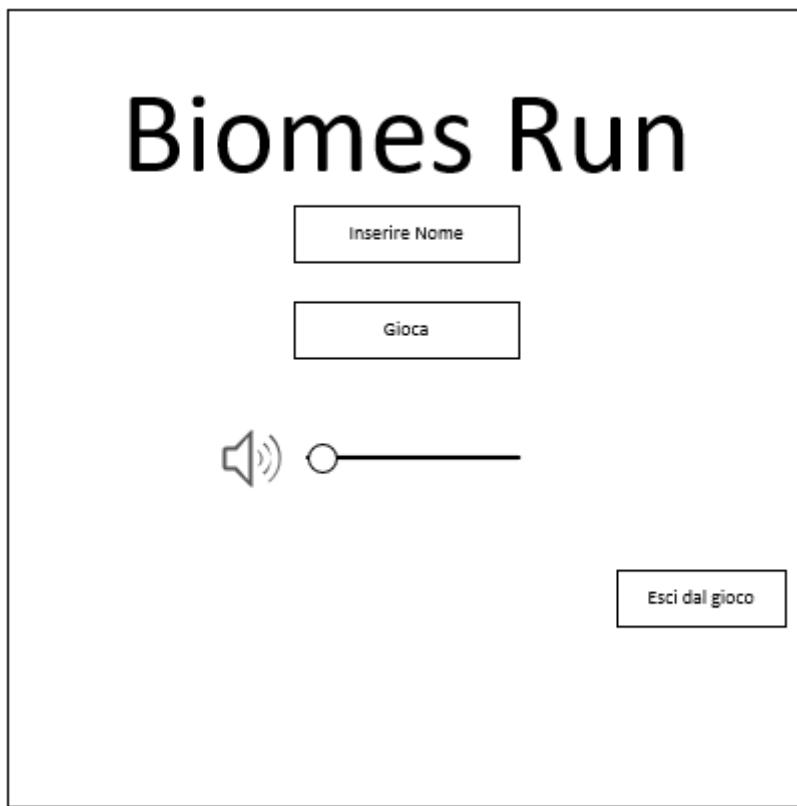


Figura 5 - Schermata iniziale

La prima schermata che appare appena si apre il gioco è quella di benvenuto, essa è composta da:

- Il titolo del gioco.
- Un textbox dove l'utente dovrà inserire il suo nome che servirà per essere memorizzato nel DataBase per la classifica.
- Un bottone “Gioca” che permetterà all'utente di iniziare la partita.
- Un bottone “Esci dal gioco” che permetterà all'utente di uscire dal gioco.
- Uno slider che permetterà di alzare a abbassare il volume.

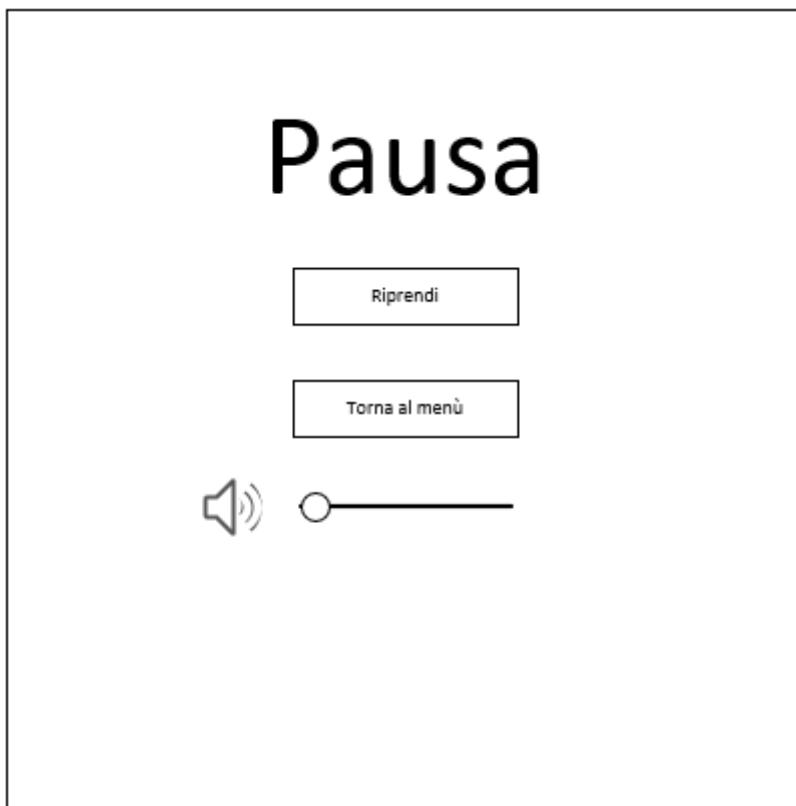


Figura 6 - Schermata di pausa

La schermata di pausa potrà essere accessibile durante il gameplay, l'utente potrà aprirla utilizzando il tasto “esc” e si aprirà questa schermata con:

- Un bottone “Riprendi” che riprenderà il gioco.
- Un bottone “Esci dal gioco” che permetterà all'utente di uscire dal gioco.
- Uno slider che permetterà di alzare a abbassare il volume.



Figura 7 - Schermata di vittoria

La schermata di vittoria apparirà all'utente quando riuscirà a completare anche l'ultimo livello, essa comprende:

- Il tempo impiegato per terminare il gioco
- La relativa posizione in classifica
- Un bottone “Rigjoca” per ricominciare il gioco
- Un bottone “Torna al menù” per tornare alla schermata iniziale

Benvenuto nel bioma della giungla

La giungla è una Vasta zona di terreno basso e umido, tipica dei paesi equatoriali e tropicali, interamente coperta da fitta e intricata vegetazione.

In questo livello dovrà prendere 6 bonus nascosti in giro per poter accedere al livello successivo.

Stai attento però, qualcuno cercherà di impedirtelo...

Gioca

Figura 8 - Livello uno

Ben fatto, hai passato il primo livello! Benvenuto nel bioma polare

Il bioma polare terrestre è costituito da terreni completamente ricoperti da ghiacciai, ed è caratterizzato da precipitazioni scarse e nevose.

In questo livello dovrà prendere 7 bonus di cui 2 nascosti in una stanza chiusa a chiave, trovandola si potrà accedere al livello successivo.

Stai attento però, qualcuno cercherà di impedirtelo...

Gioca

Figura 9 - Livello due

Complimenti, sei all'ultimo bioma!

La savana è un bioma terrestre soprattutto subtropicale e tropicale localizzato tra 10 e 20° di latitudine (N e S) e caratterizzato da una stagione secca e da una stagione umida.

In questo livello dovrà prendere 11 bonus di cui 4 nascosti in stanze chiuse a chiave. Fai attenzione però, una chiave È falsa, infatti ti riporterà all'inizio del gioco hahahah. Prendendo tutti i bonus e le chiavi potrai scappare e terminare il gioco.

Stai attento però, qualcuno cercherà di impedirtelo...

Gioca

Figura 10 - Livello tre

Le schermate dei tre livelli appariranno sempre prima dell'inizio del relativo livello e contengono:

- Una descrizione del relativo bioma.
- La spiegazione di quello che l'utente dovrà fare per accedere al livello successivo
- Un bottone "Gioca" per iniziare il livello

Mi dispiace, Hai Preso la chiave sbagliata

Peccato, sei caduto nella mia trappola come sospettavo. Ritenta, sarai più fortunato.

[Continua](#)

[Torna al menù](#)

Figura 11 - Schermata perdita chiave falsa

Questa schermata appare all'utente se nell'ultimo livello dovesse prendere la chiave sbagliata, essa contiene due buttoni:

- Il bottone “Continua” fa ricominciare il gioco all’utente
- Il bottone “Torna al menù” riporta l’utente alla schermata iniziale



Figura 12 - Schermata perdita vite

Questa schermata appare all'utente se dovesse perdere tutte le vite per via de mostri. Essa contiene due bottoni:

- Il bottone “Rigioca” fa ricominciare il gioco all’utente
- Il bottone “Torna al menù” riporta l’utente alla schermata iniziale

3.4 Design procedurale

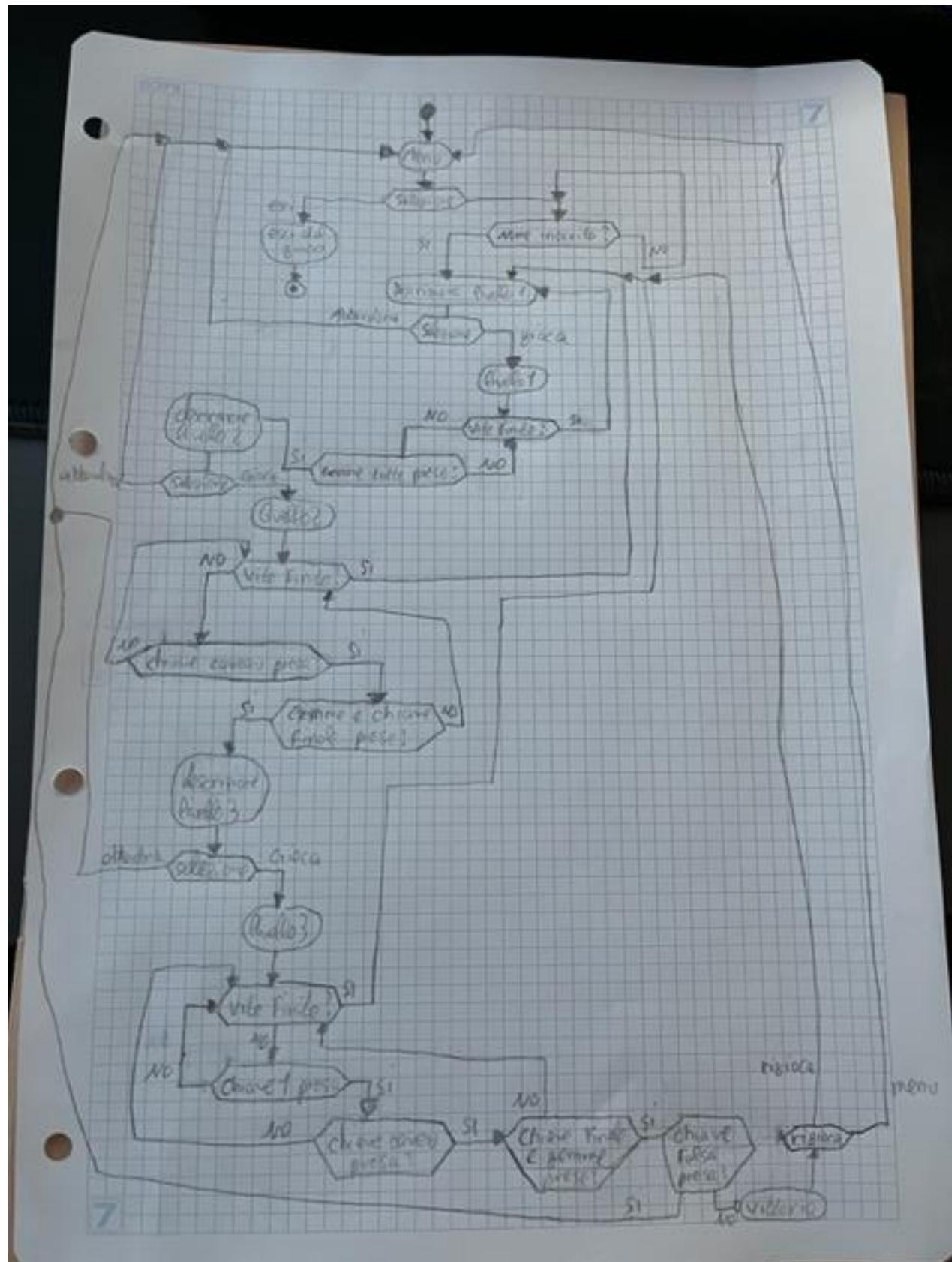


Figura 13 - Diagramma UML

4 Implementazione

4.1 Costruzione labirinto

Per avere ordine nella barra della gerarchia creerò man mano durante la costruzione del labirinto degli empty object nei quali dividerò tutti i modelli 3d. Per esempio: ho creato un empty object chiamato Walls nel quale metterò tutti i muri

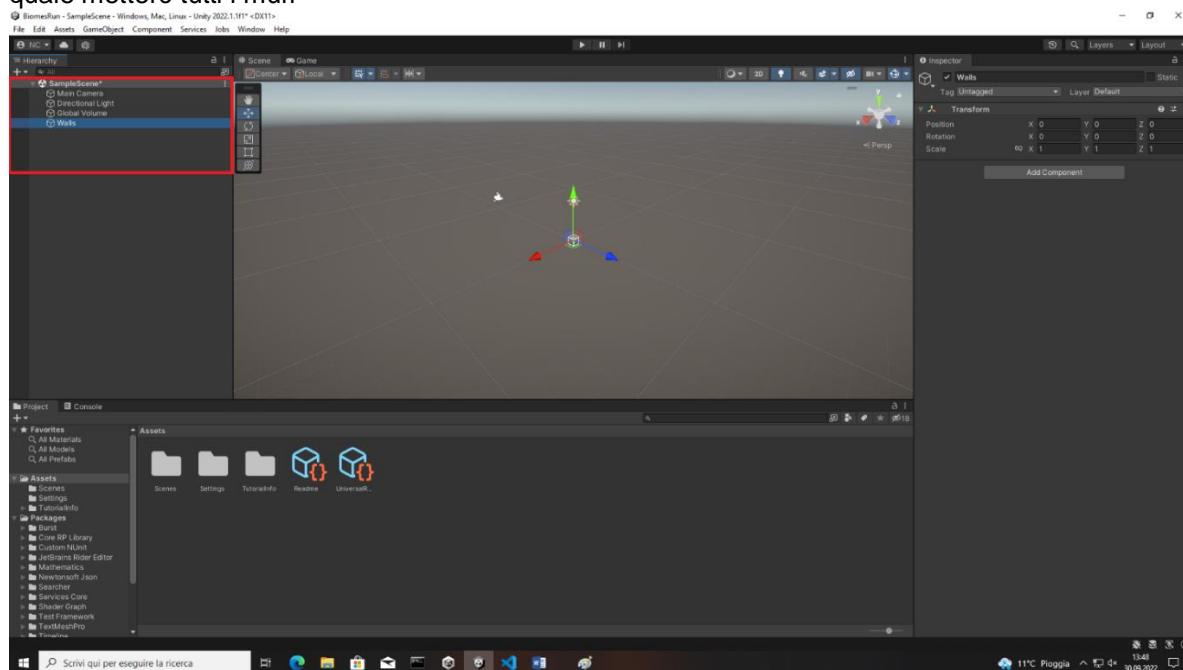


Figura 14 - Empty Object

Ho creato i muri e il piano per costruire la base del labirinto. Nella barra della gerarchia si vede che i muri sono nell' empty object Walls.

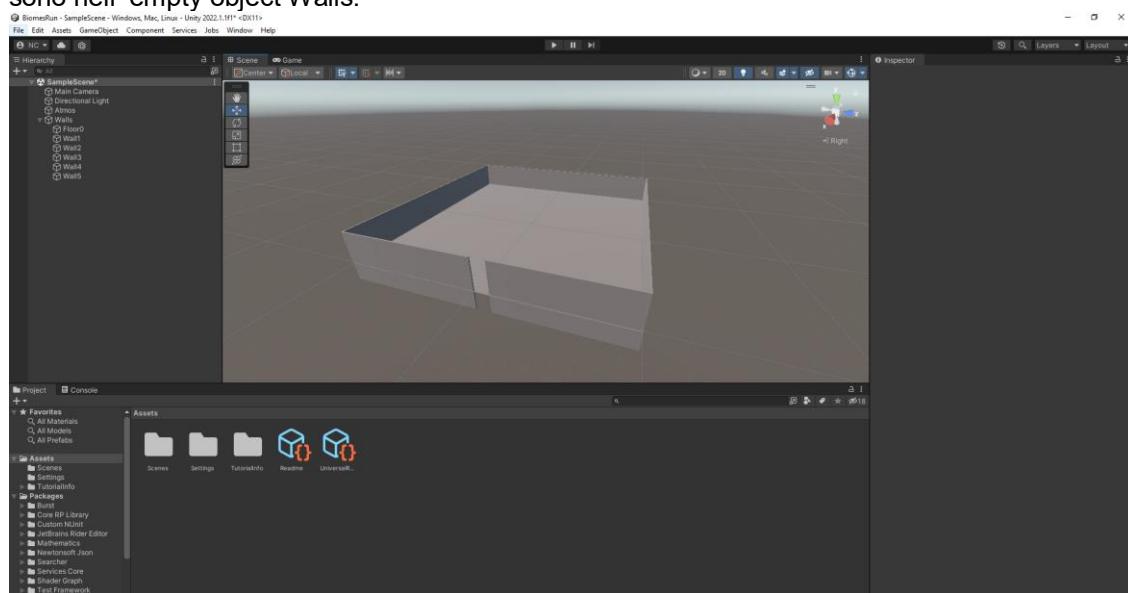


Figura 15 - Costruzione muri

4.1.1.1 Layout labirinto dei 3 livelli

Prima di creare i labirinti direttamente su Unity, ho sviluppato casualmente il loro layout con Paint.

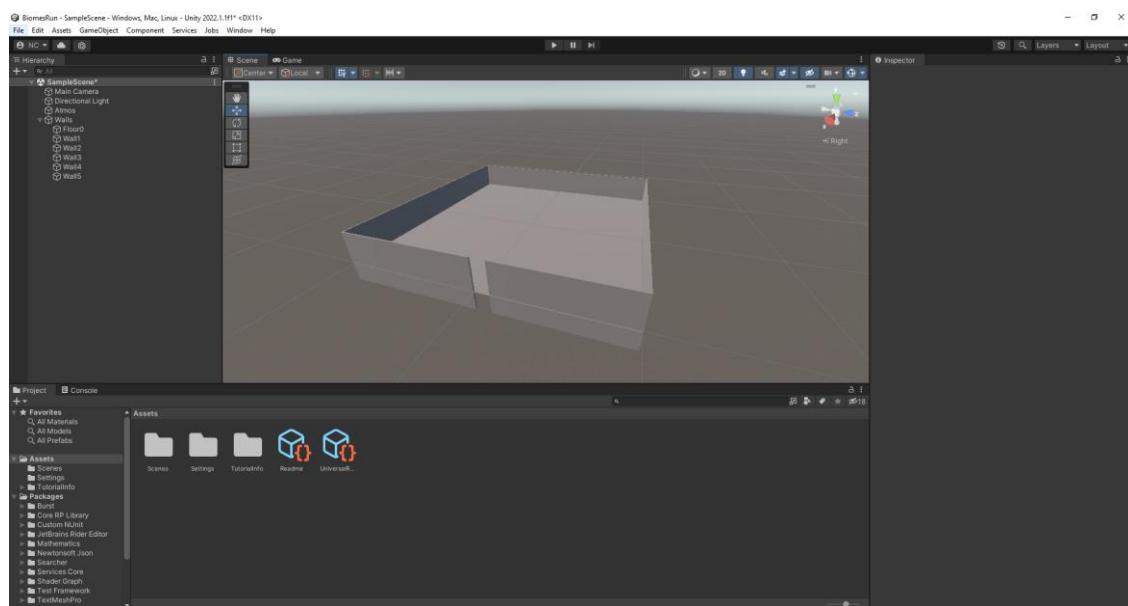


Figura 16 – Inizio costruzione labirinto

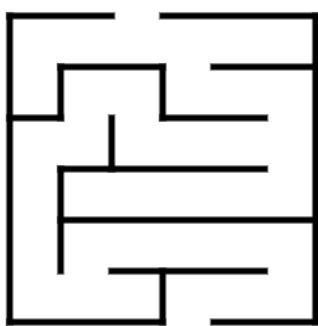


Figura 19 – Layout livello 1

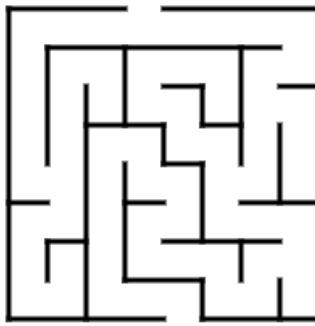


Figura 18 - Layout livello 2

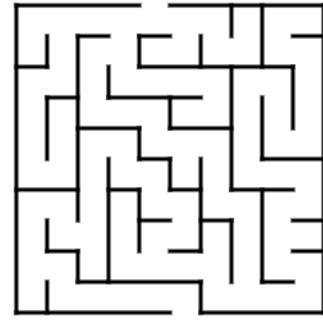


Figura 17 - Layout livello 3

4.2 Costruzione 3 livelli in Unity

Ho fatto il layout dei 3 labirinti seguendo più o meno i layout creati in Unity.

Legenda dei colori utilizzati:

- I buchi presenti in alto a sinistra dei labirinti segnano il punto di inizio di ogni livello
- La parete bucata in mezzo è la fine del labirinto, e quando si avrà varcato la porta c'è ritroverà in alto a sinistra del livello successivo.
- I muri verdi rappresentano le porte quali si apriranno con le chiavi mentre il muro verde rappresenta una porta falsa, in quella stanza sarà presente la chiave falsa, la quale ti farà riiniziare il gioco

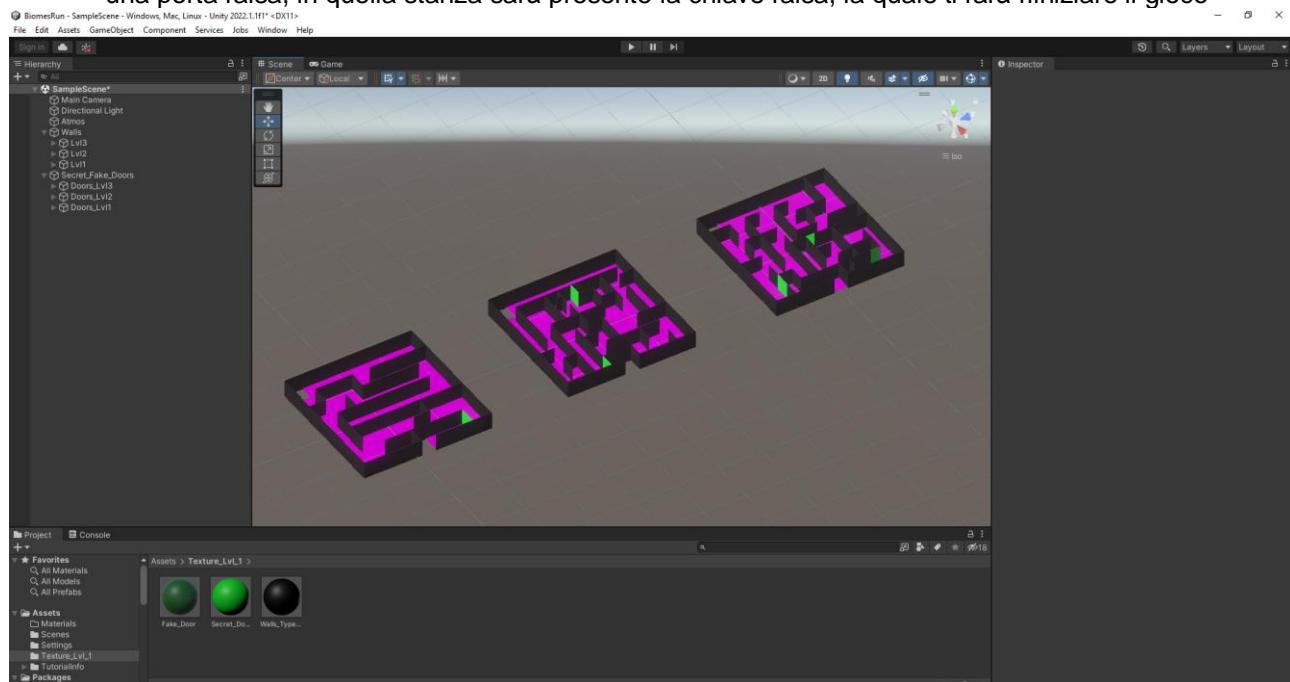


Figura 18 - Layout 3 labirinti

4.3 Decorazione Labirinto

4.3.1 Modelli 3D

Per decorare il labirinto ho scaricato dei modelli 3D e gli ho messi in una cartella che mi contiene tutti gli oggetti. Per avere ordine ho creato una struttura di cartelle in questo modo:

Livello_nr/3D Models/*.fbx

Ogni livello ha la propria cartella dei modelli 3D.

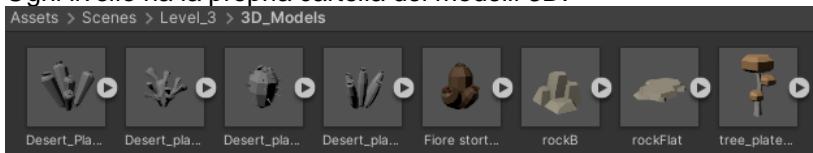


Figura 21 - Modelli 3D

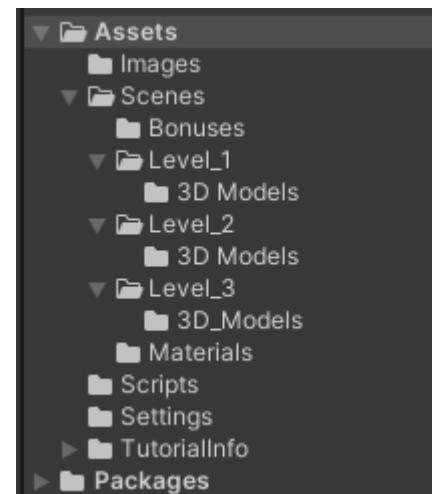


Figura 20 - Gerarchia modelli

4.3.2 Decorazione labirinto

Ho decorato i 3 labirinti utilizzando i modelli 3D scaricati. I 3 labirinti gli ho decorati in base ad un bioma

- Livello 1 è basato sul bioma della giungla



Figura 22 - Decorazione livello1

- Livello 2 è basato su un bioma polare



Figura 23 - Decorazione livello 2

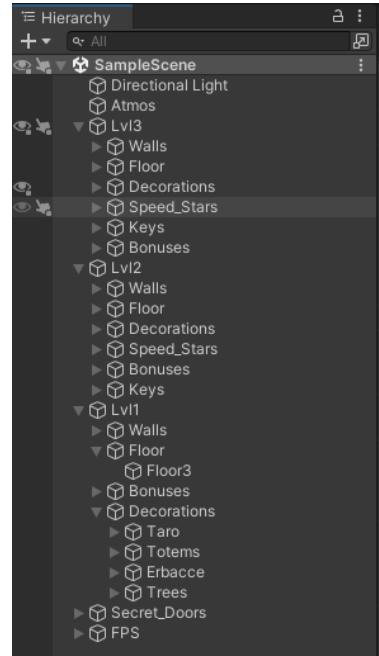
- Livello 3 è basato sul bioma della savana



Figura 24 - Decorazione livello3

4.3.3 Albero cartelle

Alla fine della decorazione per avere ordine nei miei oggetti ho creato diversi empty object per smistare il tutto per livelli.



4.3.4 Script Rotatore

Chiavi, stelline e punti bonus gli ho fatti roteare su sé stessi utilizzando uno script. Ho creato la mia cartella degli Scripts in Assets e ho creato uno script chiamato KeyRotator, all'interno ho scritto questo codice:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class KeyRotator : MonoBehaviour
{
    public Vector3 rotation;

    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        GetComponent<Transform>().Rotate(rotation);
    }
}
```

Figura 25 - Ordine oggetti

Figura 26 - Script rotatore oggetti

E su Unity collego lo script agli oggetti, e nello script potrò scegliere la velocità e verso che verso farli girare

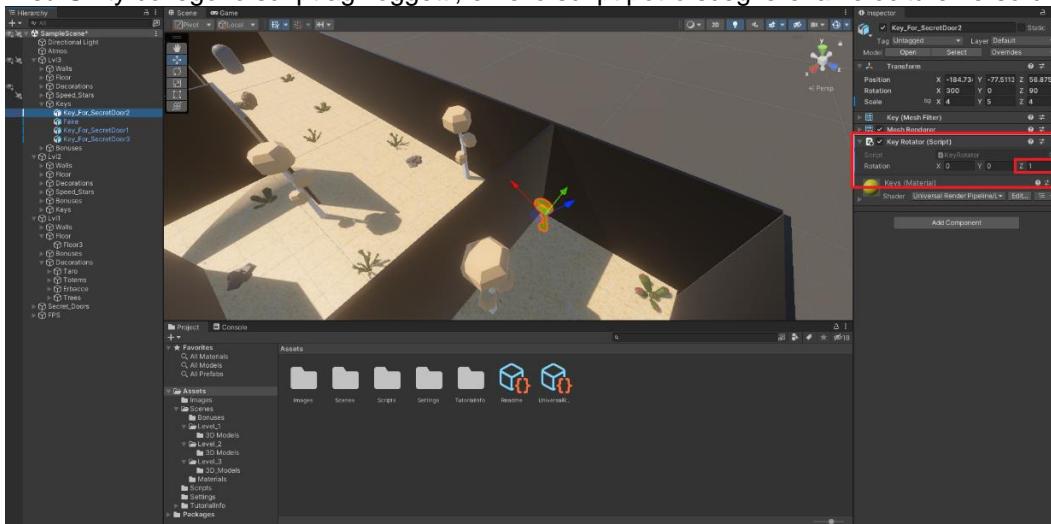


Figura 27 - Chiave con script rotazione

4.4 Prima persona

Per il movimento in prima persona ho creato un piano al quale ho aggiunto il componente Rigidbody, ho tolto la gravità e 'ho reso statico per far sì che non si sposti durante la partita

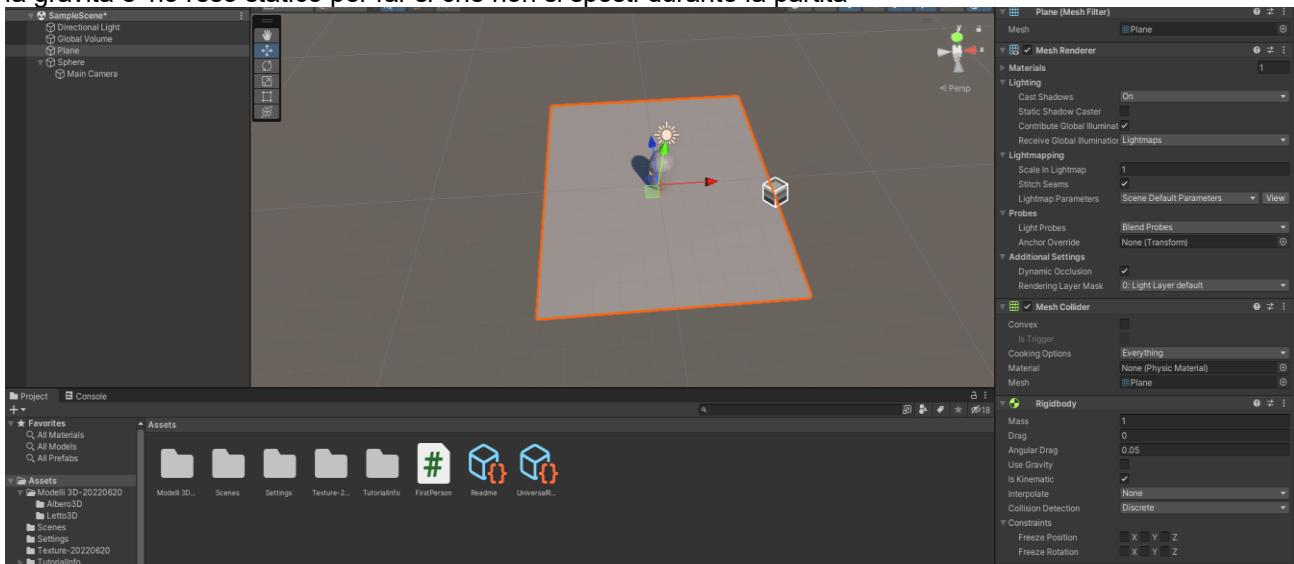


Figura 28 - Pavimento Grezzo

Per introdurre la camera in prima persona ho creato un personaggio grezzo (Capsule) al quale ho bloccato la rotazione X e Z perché altrimenti si sarebbe spostato.

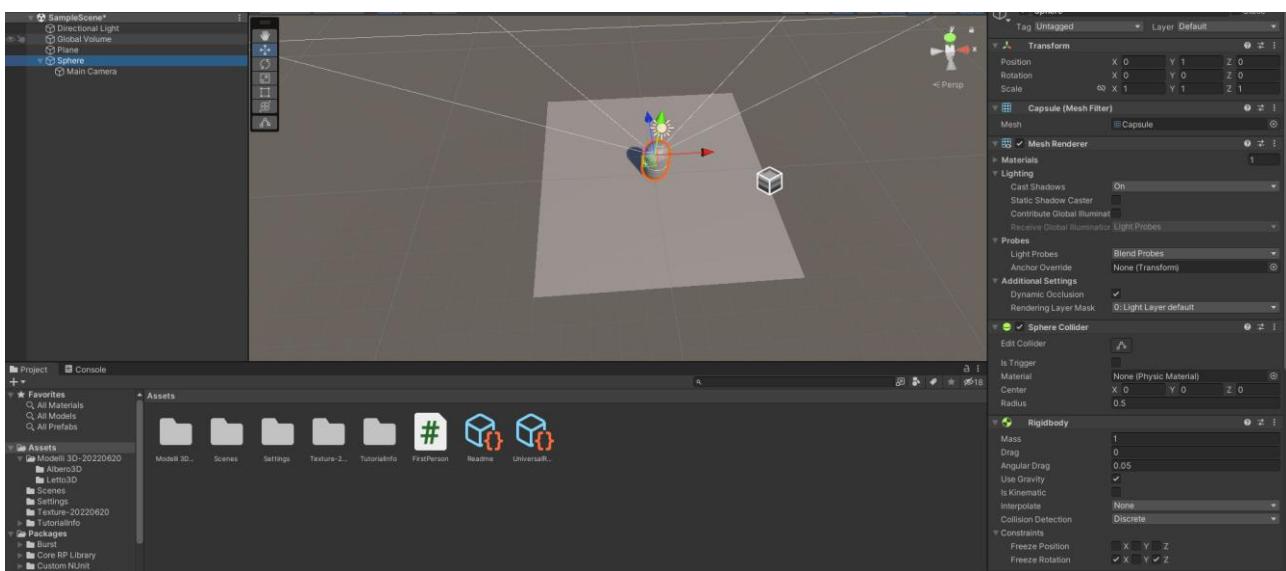


Figura 29 - Personaggio Grezzo

4.4.1 Script telecamera

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class MoveCamera : MonoBehaviour
6  {
7      public Transform cameraPosition;
8      // Update is called once per frame
9      void Update()
10     {
11         //prende la posizione della main camera
12         transform.position = cameraPosition.position;
13     }
14 }
```

Figura 30 - Settaggio posizione camera

Questo script permette di dare la posizione per il movimento della telecamera

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  // Script Unity (3 riferimenti ad asset) | 0 riferimenti
6  public class PlayerCam : MonoBehaviour
7  {
8
9      public float sensX;
10     public float sensY;
11
12     public Transform orientation;
13
14     float xRotation;
15     float yRotation;
16
17     // Start is called before the first frame update
18     void Start()
19     {
20         Cursor.lockState = CursorLockMode.Locked;
21         Cursor.visible = false;
22     }
23
24     // Update is called once per frame
25     void Update()
26     {
27         //get mouse input
28         float mouseX = Input.GetAxisRaw("Mouse X") * Time.deltaTime * sensX;
29         float mouseY = Input.GetAxisRaw("Mouse Y") * Time.deltaTime * sensY;
30
31         yRotation += mouseX;
32         xRotation -= mouseY;
33         xRotation = Mathf.Clamp(xRotation, -90f, 90f);
34
35         //rotate cam and orientation
36         transform.rotation = Quaternion.Euler(xRotation, yRotation, 0);
37         orientation.rotation = Quaternion.Euler(0, yRotation, 0);
38     }
}
```

Figura 31 - movimento telecamera

Questo script prende l'input del mouse del player e permette il movimento della telecamera con il mouse bloccando il cursore al centro dello schermo e rendendolo invisibile

4.4.2 Script prima persona personaggio

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using TMPro;
4  using UnityEngine;
5  using UnityEngine.SceneManagement;
6  using UnityEngine.UI;
7
8  // Script Unity (3 riferimenti ad asset) | 0 riferimenti
9  public class PlayerMovement : MonoBehaviour
10 {
11     [Header("Movement")]
12     public float moveSpeed;
13     public float moveMultiplier;
14
15     public float groundDrag;
16
17     [Header("Ground Check")]
18     public float playerHeight;
19     public LayerMask whatIsGround;
20     bool grounded;
21
22     [Header("Player damage manager")]
23     public float damageTime;
24
25     public Transform orientation;
26
27     float horizontalInput;
28     float verticalInput;
29
30     Vector3 moveDirection;
31
32     Rigidbody rb;
33
34     public float speed = 10.0f;
35     public Image imgLives;
36
37     private int lives = 3;
38
39     [SerializeField]
40     private Sprite[] _liveSprites;
41
42     bool canTakeDamage;
43
44     public AudioSource hitEnemy;
45     public AudioSource key;
46     public AudioSource star;
47
48     void Start()
49     {
50         Cursor.lockState = CursorLockMode.Locked;
51         Cursor.visible = false;
52
53         canTakeDamage = true;
54
55         rb = GetComponent<Rigidbody>();
56         rb.freezeRotation = true;
57     }
58
59     // Update is called once per frame
60     // Messaggio Unity | 0 riferimenti
61     void Update()
62     {
63         //imposto il player attaccato al terreno
64         grounded = Physics.Raycast(transform.position, Vector3.down, playerHeight * 0.5f + 0.2f, whatIsGround);
65
66         MyInput();
67         SpeedControl();
68
69         //verifico di essere attaccato al terreno
70         if (grounded)
71         {
72             rb.drag = groundDrag;
73         }
74         else
75         {
76             rb.drag = 0;
77         }
78
79         private void FixedUpdate()
80         {
81             MovePlayer();
82         }
83
84         public int updateLives(int currentLives) // metodo che aggiorna le vite del personaggio
85         {
86             imgLives.sprite = _liveSprites[currentLives];
87             return currentLives;
88         }
89 }
```

```

85
86
87     public void OnCollisionEnter(Collision collision)
88     {
89         if (collision.gameObject.tag == "mostro") // se collido con un oggetto con il tag mostro mi toglie una vita
90         {
91             if (canTakeDamage)
92             {
93                 canTakeDamage = false;
94                 StartCoroutine(CollisionWithMonster());
95             }
96         }
97         if (lives == 0) // se le vite sono 0 carica la schermata di perdita
98         {
99             GetComponent<MenuManager>().LoseLives();
100        }
101
102        if (collision.gameObject.tag == "stella") // se collido con un oggetto con il tag stella mi aumenta la velocità
103        {
104            Destroy(collision.gameObject);
105            StartCoroutine(CollisionWithStar());
106        }
107
108        if (collision.gameObject.tag == "Portal1")
109        // se collido con un oggetto con il Portal1 mi porta alla schermata di spiegazione del secondo bioma
110        {
111            GetComponent<MenuManager>().LoadBiome2();
112        }
113
114        if (collision.gameObject.tag == "Portal2")
115        // se collido con un oggetto con il Portal2 mi porta alla schermata di spiegazione del terzo bioma
116        {
117            GetComponent<MenuManager>().LoadBiome3();
118        }
119
120        if (collision.gameObject.tag == "Portal3")
121        // se collido con un oggetto con il Portal3 mi porta alla schermata di vittoria
122        {
123            GetComponent<MenuManager>().Victory();
124        }
125
126        if(collision.gameObject.tag == "WrongKey")
127        // se collido con un oggetto con il WrongKey mi porta alla schermata di perdita per la chiave falsa
128        {
129            GetComponent<MenuManager>().LoseKey();
130        }
131    }
132
133    riferimento
134    private void MyInput() // metodo che prende le assi del mouse
135    {
136        horizontalInput = Input.GetAxisRaw("Horizontal");
137        verticalInput = Input.GetAxisRaw("Vertical");
138    }
139
140    riferimento
141    private void MovePlayer() //metodo che serve per muovere il player
142    {
143        //calcolo direzione di movimento
144        moveDirection = orientation.forward * verticalInput + orientation.right * horizontalInput;
145
146        rb.velocity = moveDirection.normalized * moveSpeed * moveMultiplier;
147
148    riferimento
149    public void SpeedControl() // metodo che rende la velocità costante
150    {
151        Vector3 flatVel = new Vector3(rb.velocity.x, 0f, rb.velocity.z);
152
153        //limit velocity if needed
154        if(flatVel.magnitude > moveSpeed)
155        {
156            Vector3 limitedVel = flatVel.normalized * moveSpeed;
157            rb.velocity = new Vector3(limitedVel.x, rb.velocity.y, limitedVel.z);
158        }
159
160    riferimento
161    IEnumerator CollisionWithMonster() // coroutine che danneggia il giocatore togliendogli una vita
162    {
163        if(damageTime == 0)
164        {
165            damageTime = 1f;
166        }
167        lives--;
168        imgLives.sprite = _liveSprites[updateLives(lives)];
169        yield return new WaitForSeconds(damageTime);
170        canTakeDamage = true;
171    }

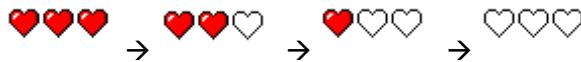
```

Figura 32 - Script completo movimento + funzionalità aggiuntive

Questo script comprende tutto quello che concerne il movimento del player, inoltre comprende anche tutte le funzionalità per il passaggio dei livelli collidendo con i Portal, la perdita per colpa della chiave falsa, il counter delle vite con il caricamento della schermata di perdita se si dovessero perdere tutte, il controllo della velocità per fare in modo che sia omogenea e l'aumento di velocità se si collide con una stella e tutto quello che riguarda gli input del mouse per il movimento, inoltre comprende gli effetti audio per il collezionamento degli oggetti.

4.5 Vite personaggio

Per creare le vite del personaggio ho utilizzato un sito che mi ha permesso di avere delle immagini con background trasparente. Il risultato è questo:



Per poter avere un player con delle vite ho dovuto creare all'interno del gameObject del player un canvas con all'interno un immagine contenente l'immagine delle tre vite:

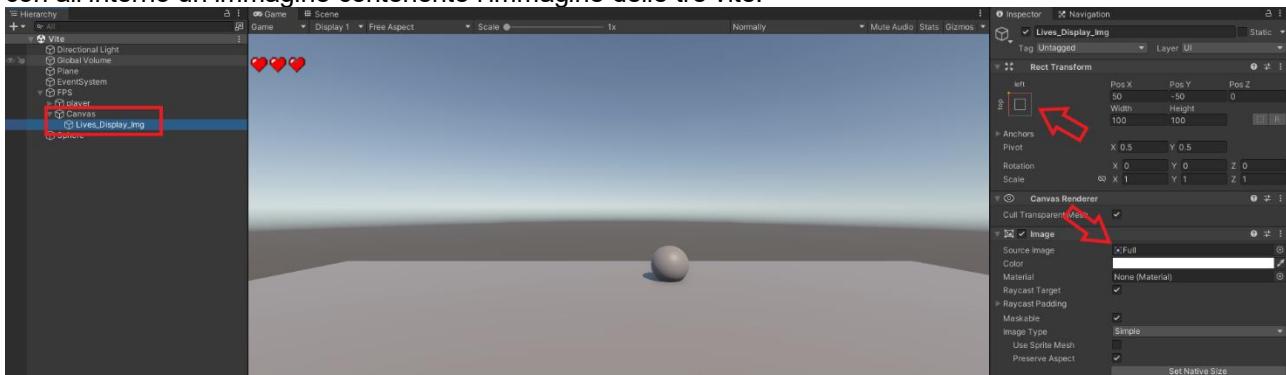


Figura 33 - Inserimento immagine vite

Dopodiché ho creato uno script contenente un array di immagini dove ogni volta che il player viene colpito, incrementa una variabile che aumenta la posizione dell'array e l'immagine delle vite cambia. Dopo averlo creato ho inserito nella lista dello script le vite.

```

1 riferimento
public int updateLives(int currentLives) // metodo che aggiorna le vite del personaggio
{
    imgLives.sprite = _liveSprites[currentLives];
    return currentLives;
}

@ Messaggio Unity | 0 riferimenti
public void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.tag == "mostro") // se collido con un oggetto con il tag mostro mi toglie una vita
    {
        if (canTakeDamage)
        {
            canTakeDamage = false;
            StartCoroutine(CollisionWithMonster());
        }
    }
}

1 riferimento
IEnumerator CollisionWithStar() // coroutine per la velocità se si prende una stella
{
    moveSpeed += 5;
    yield return new WaitForSeconds(3f);
    moveSpeed -= 5;
}

```

Figura 34 - Script vite

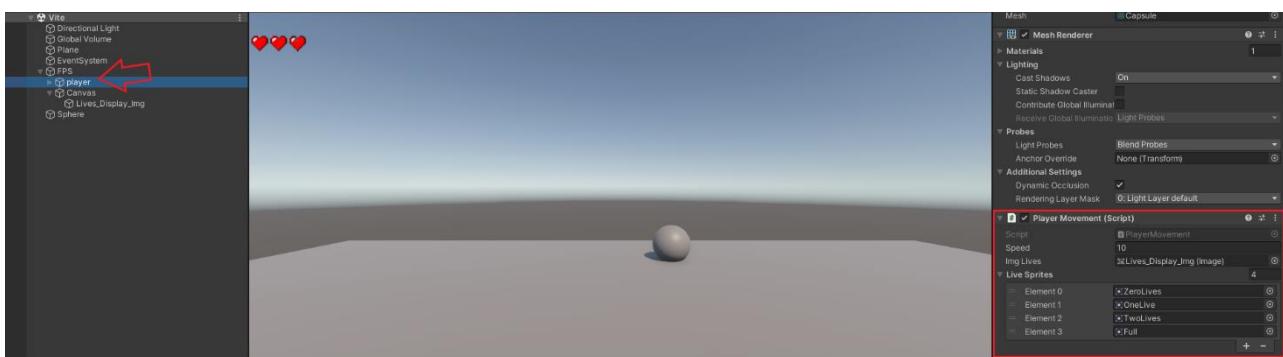


Figura 35 - Inserimento immagini nell'array

Facendo così basta dare al nemico il tag “mostro” e il gioco è fatto. Se il player dovesse finire le vite, verrebbe caricata la schermata di perdita.

4.6 Programmazione mostro

Per la programmazione dei mostri ho creato uno script che inseguisse il player. Inoltre, bisogna "dire" al nemico dove può passare grazie al bake, esso permette di stabilire le parti camminabili e non.

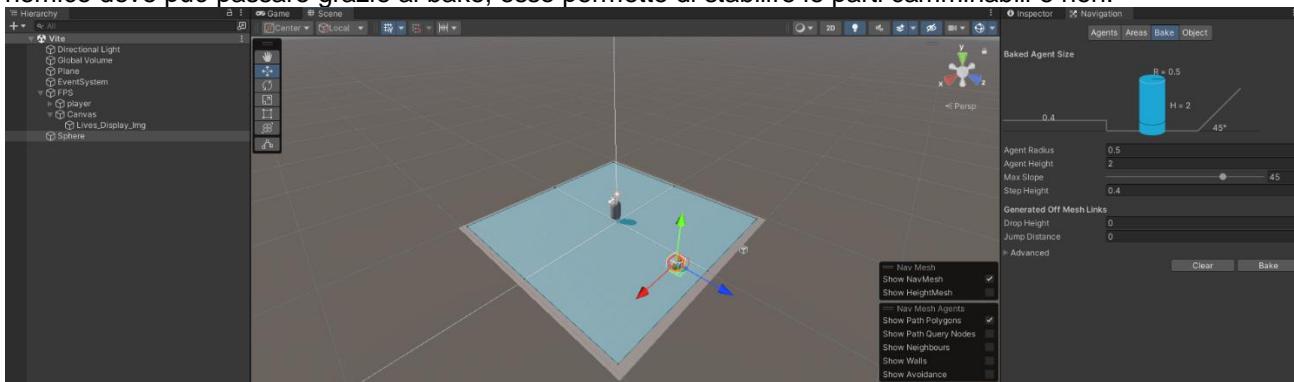


Figura 36 - Bake del mostro

Il mostro deve avere il tag mostro per poter far funzionare il counter delle vite e l'inseguimento

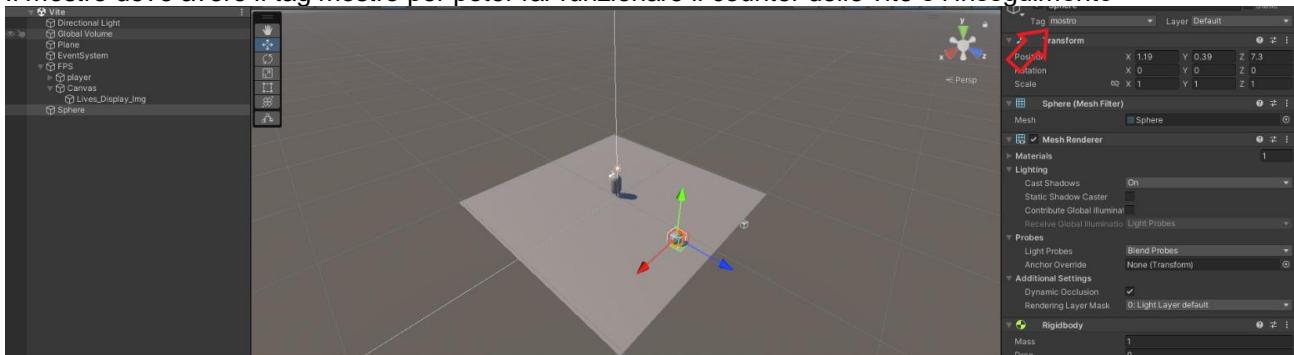


Figura 37 - tag del mostro

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.AI;
5
6  public class Enemy : MonoBehaviour
7  {
8      public Transform Target;
9
10     // Update is called once per frame
11     void Update()
12     {
13         // l'agent insegue il player (Target)
14         NavMeshAgent agent = GetComponent<NavMeshAgent>();
15         agent.destination = Target.position;
16     }
17 }
```

Figura 38 - script mostro

4.7 Programmazione schermate

Oltre alla schermata delle vite, ho creato diverse schermate:

✓ Scenes/Benvenuto	0
✓ Scenes/Bioma1	1
✓ Scenes/SampleScene	2
✓ Scenes/Bioma2	3
✓ Scenes/Level2	4
✓ Scenes/Bioma3	5
✓ Scenes/Level3	6
✓ Scenes/Vittoria	7
✓ Scenes/Perdita_Vite	8
✓ Scenes/Perdita_Chiave	9
✓ Scenes/Pausa	10

Figura 39 - Ordine scene

4.7.1 Schermata 0 (Benvenuto)

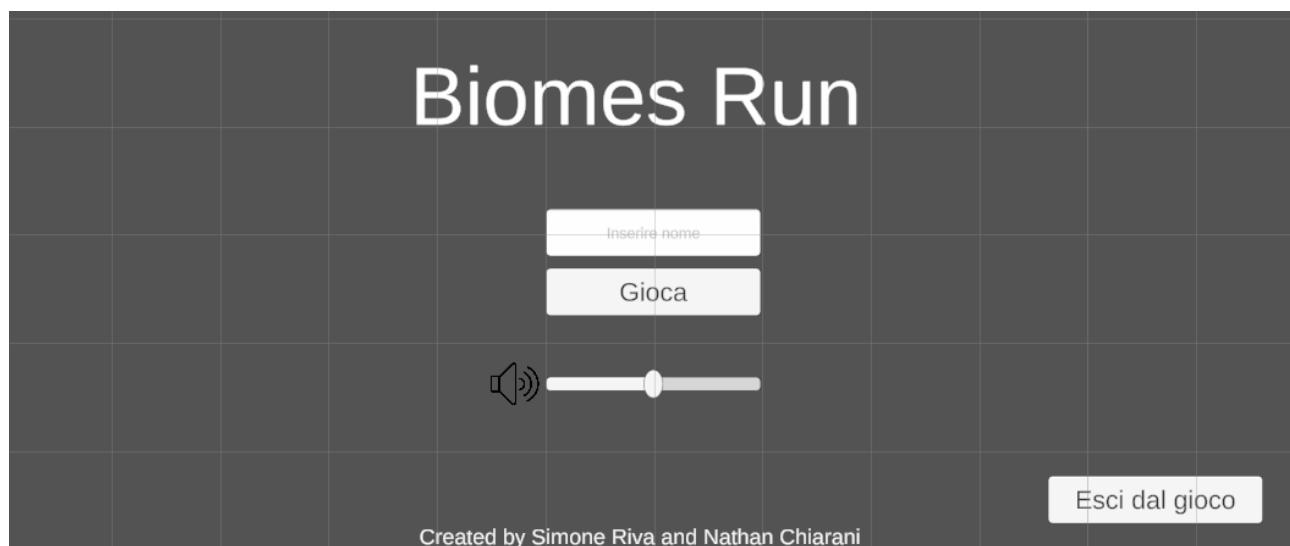


Figura 40 - Schermata 0 (Benvenuto)

Per verificare che il nome sia stato inserito, utilizzo uno script di controllo:

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using TMPro;
4  using UnityEngine;
5  using UnityEngine.UI;
6
7  public class CheckText : MonoBehaviour
8  {
9      public TMP_InputField text;
10     public GameObject button;
11
12     void Update()
13     {
14         if(text.text.Length > 0) // verifico se nel textbox del nome è stato inserito qualcosa, se vuoto il textbox è enabled
15         {
16             button.GetComponent<Button>().enabled = true;
17         }
18         else
19         {
20             button.GetComponent<Button>().enabled = false;
21         }
22     }

```

Figura 41 - controllo nome

Se il nome non è inserito, il tasto gioca è enabled, altrimenti si può iniziare a giocare.

4.7.2 Schermata 1 Descrizione Livello 1

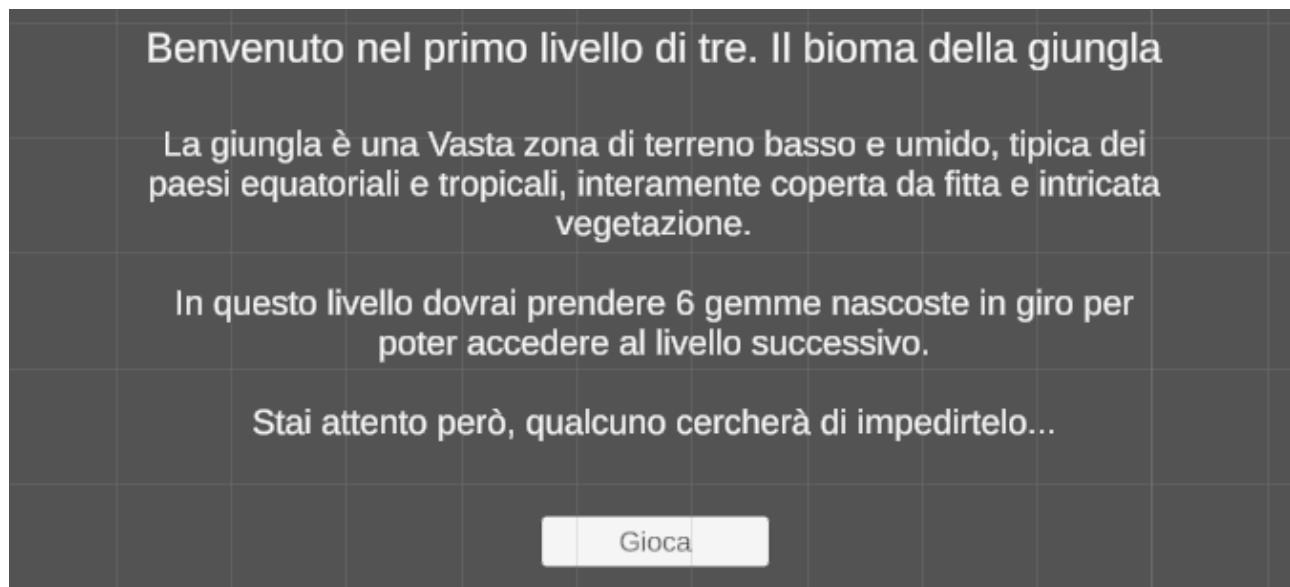


Figura 42 - Schermata 1 (Descrizione Livello 1)

4.7.3 Schermata 2 Gioco Livello 1

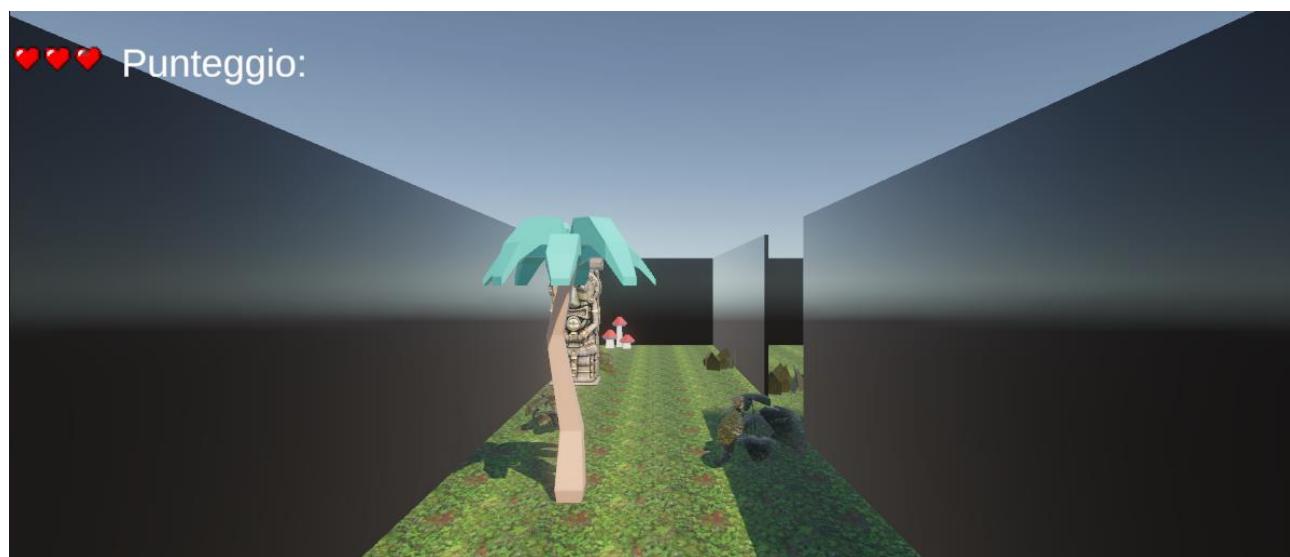


Figura 43 - Schermata 2 (Schermata Gioco Livello 1)

4.7.4 Schermata 3 (Secondo livello)

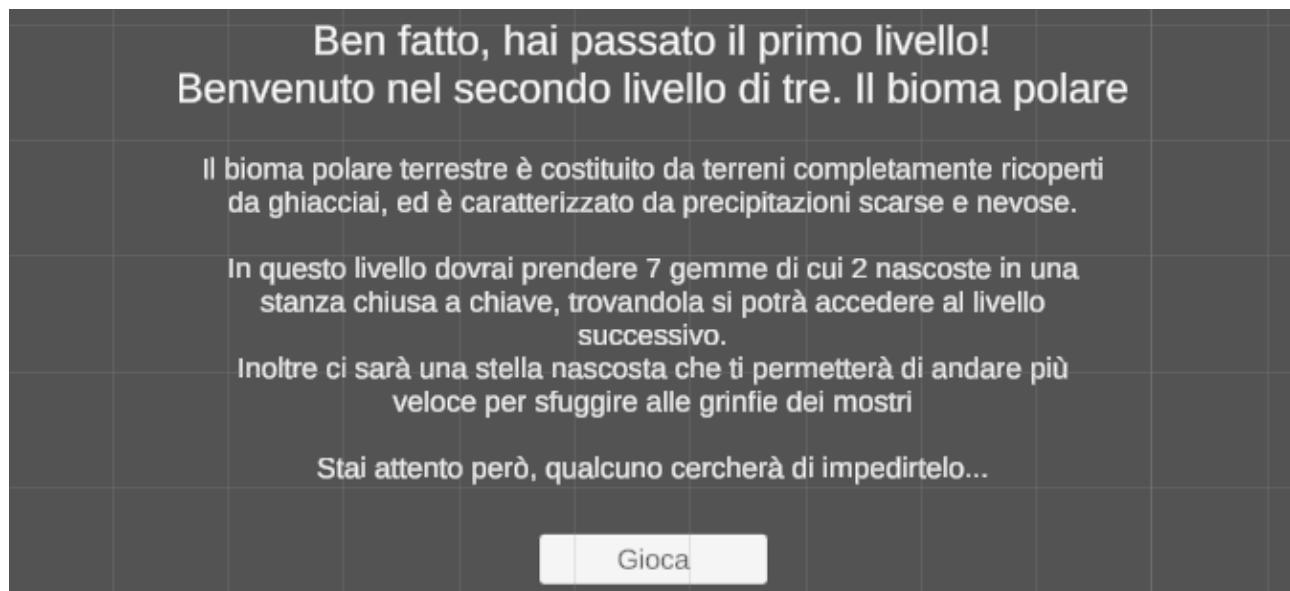


Figura 44 - Schermata 3 (Livello 2)

4.7.5 Schermata 4 Gioco Livello 2

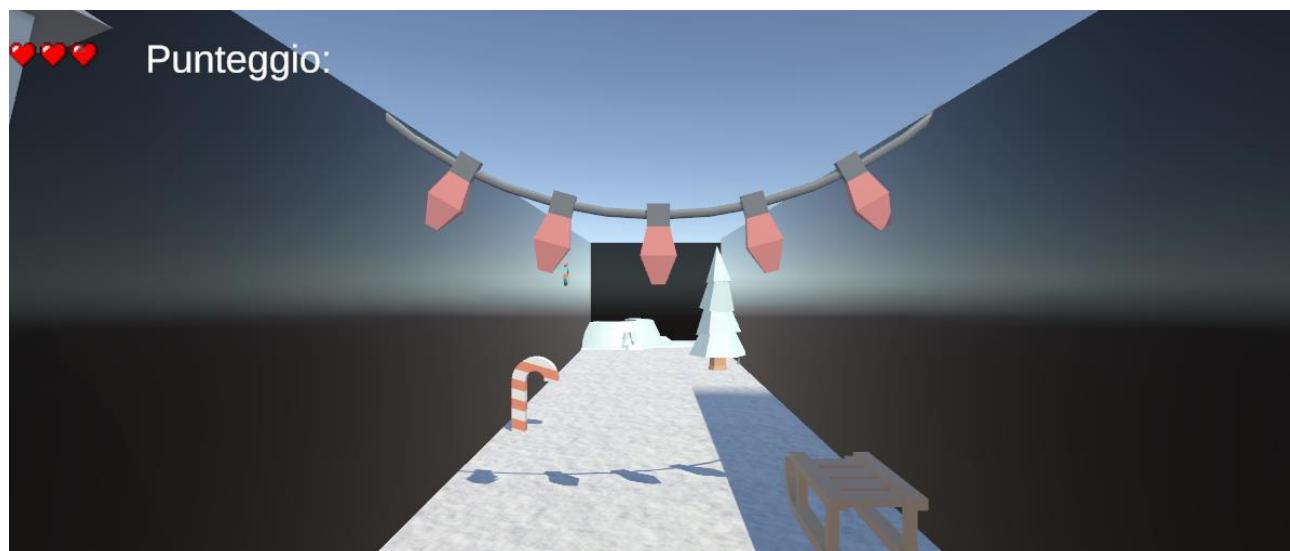


Figura 45 - Schermata 4 (Schermata Gioco Livello 2)

4.7.6 Schermata 5 (Livello3)

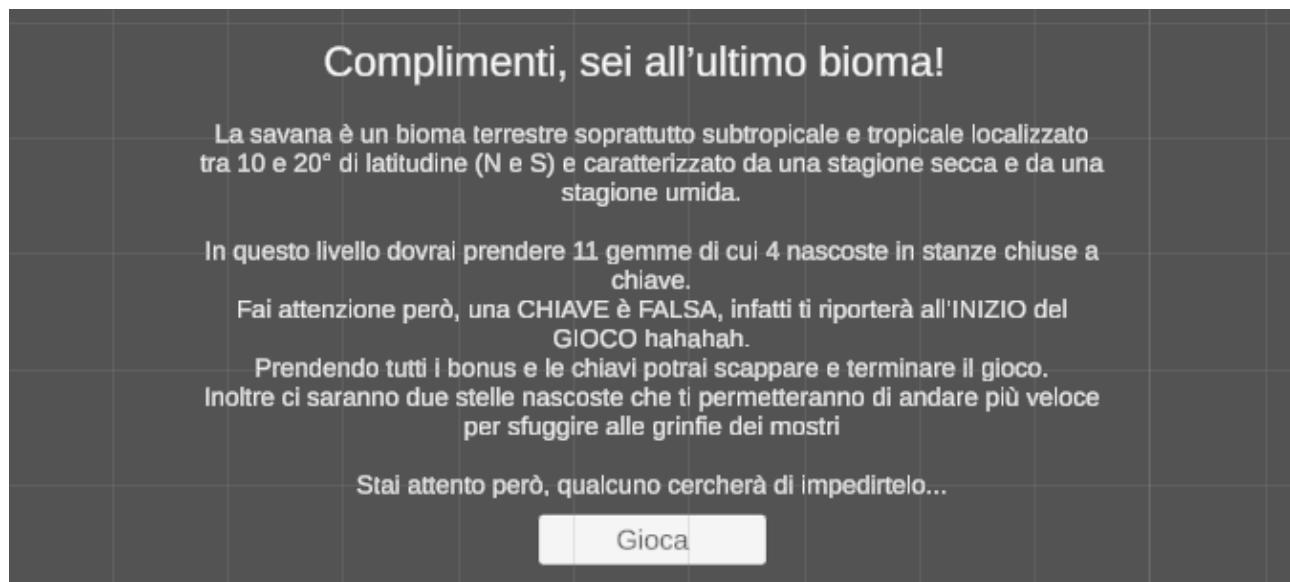


Figura 46 - Schermata 5 (Livello 3)

4.7.7 Schermata 6 Gioco Livello 3



Figura 47 - Schermata 6 (Gioco Livello 3)

4.7.8 Schermata 7 (Vittoria)



Figura 48 - Schermata 7 (Vittoria)

4.7.9 Schermata 8 (Chiave falsa)

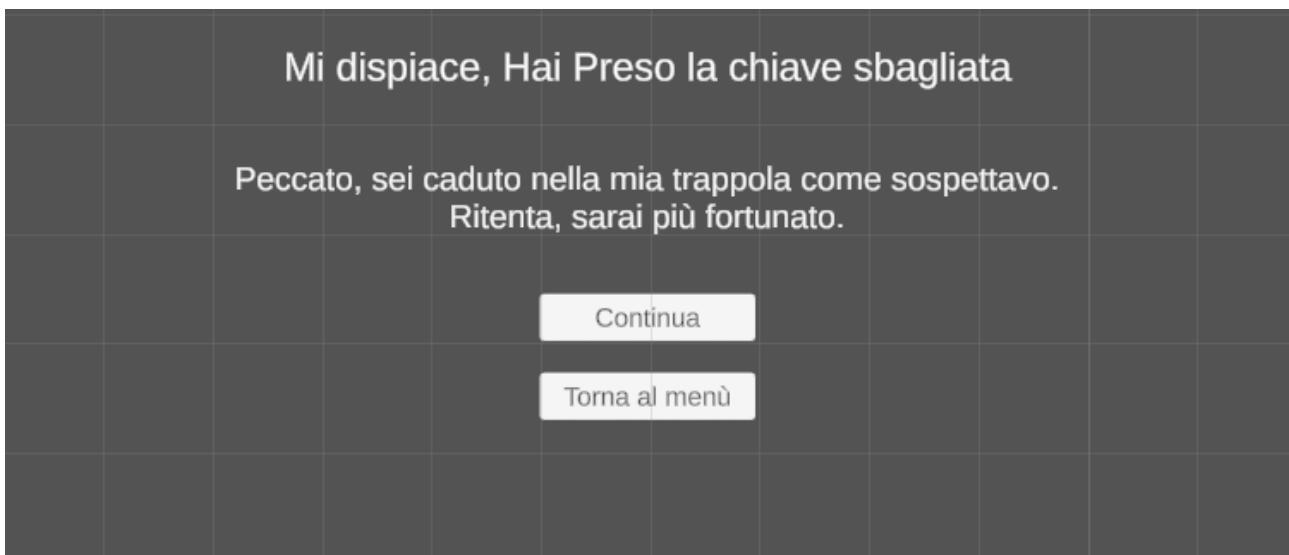


Figura 49 - Schermata 8 (Chiave falsa)

4.7.10 Schermata 9 (Vite finite)

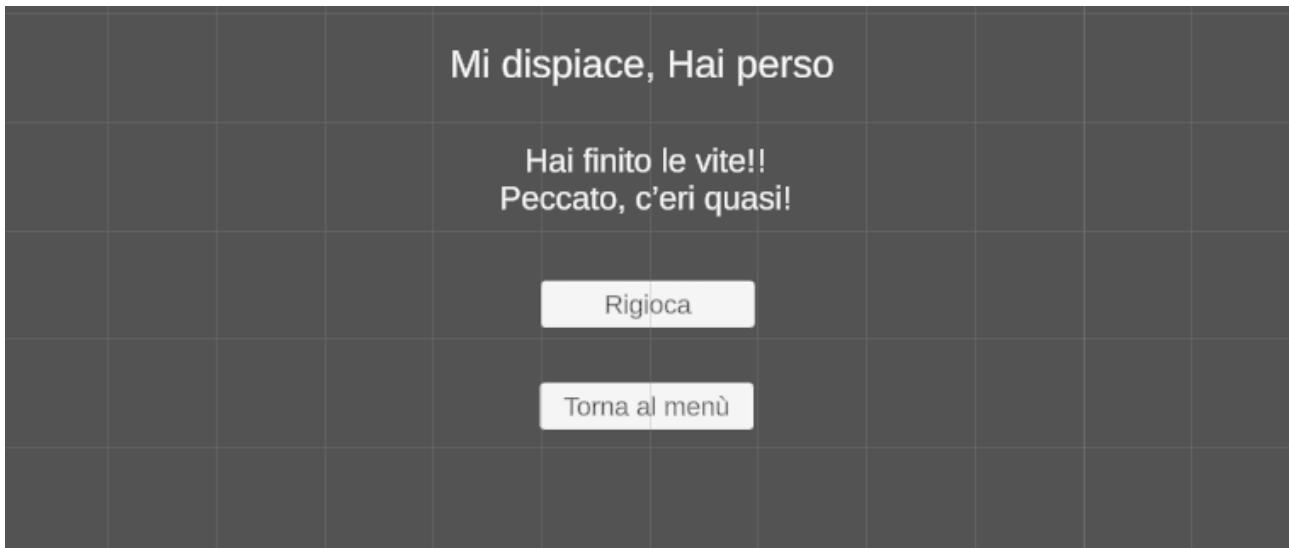


Figura 50 - Schermata 9 (Vite finite)

4.7.11 Schermata 10 (Pausa)



Figura 51 - Schermata 10 (Pausa)

Per il volume regolabile dagli slider ho utilizzato il seguente script che imposta il volume, se non era stato impostato, lo imposta a 0.2:

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
5
6  public class VolumeManager : MonoBehaviour
7  {
8      [SerializeField] Slider volumeSlider;
9      // Start is called before the first frame update
10     void Start()
11     {
12         // se non ci sono settate le impostazioni le setta, altrimenti tienne quelle già impostate
13         if (!PlayerPrefs.HasKey("musicVolume"))
14         {
15             PlayerPrefs.SetFloat("musicVolume", 0.2f);
16             Load();
17         }
18         else
19         {
20             Load();
21         }
22     }
23
24     public void ChangeVolume() // modifica il volume del gioco
25     {
26         AudioListener.volume = volumeSlider.value;
27         Save();
28     }
29
30     private void Load() // carica le impostazioni dello slider
31     {
32         volumeSlider.value = PlayerPrefs.GetFloat("musicVolume");
33     }
34
35     private void Save() // salva le impostazioni dello slider
36     {
37         PlayerPrefs.SetFloat("musicVolume", volumeSlider.value);
38     }
39 }
```

Figura 52 - Script di gestione del volume

Per l'apparizione della schermata di pausa ho utilizzato il seguente script:

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  Script Unity (3 riferimenti ad asset) | 0 riferimenti
6  public class PauseManager : MonoBehaviour
7  {
8
9      [Header("GUI")]
10     public GameObject pauseMenu;
11
12     [System.Obsolete]
13     @ Messaggio Unity | 0 riferimenti
14     private void Update()
15     {
16
17         if (Input.GetKeyDown(KeyCode.Escape)) // se l'utente schiaccia esc
18         {
19             // se la schermata di pausa è presente la toglie e riparte il gioco, altrimenti la visualizza e ferma il gioco
20             if (pauseMenu.active)
21             {
22                 DefocusMenu();
23             }
24             else
25             {
26                 FocusMenu();
27             }
28         }
29
30         public void FocusMenu() // metodo che rende il cursore visibile e visualizza la schermata di pausa
31         {
32             Cursor.lockState = CursorLockMode.Confined;
33             Cursor.visible = true;
34             pauseMenu.SetActive(true);
35             Time.timeScale = 0f;
36         }
37
38         public void DefocusMenu() // metodo che rende il cursore invisibile e rimuove la schermata di pausa
39         {
40             Cursor.lockState = CursorLockMode.Locked;
41             Cursor.visible = false;
42             pauseMenu.SetActive(false);
43             Time.timeScale = 1f;
44         }
45 }
```

Figura 53 - Script di gestione della schermata di pausa

Questo script inizia quando l'utente preme “esc” e controlla se la schermata di pausa è in visualizzazione o meno, se l'utente la sta visualizzando, la rimuove e il gioco riparte, altrimenti la visualizza e ferma il gioco.

4.7.12 Script gestione Schermate

Questo script ha il compito di gestire lo spostamento tra scene tramite metodi.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
5
6  public class MenuManager : MonoBehaviour
7  {
8
9
10    public void Exit() // esce dal gioco
11    {
12        Application.Quit();
13    }
14
15    public void ReturnMain() // carica la schermata di benvenuto
16    {
17        SceneManager.LoadScene(0);
18        mostraCursore();
19    }
20
21    public void LoadBiome1() // carica la spiegazione del primo livello
22    {
23        SceneManager.LoadScene(1);
24        mostraCursore();
25    }
26
27    public void StartLevel1() //carica il primo livello
28    {
29        SceneManager.LoadScene(2);
30        mostraCursore();
31    }
32
33    public void LoadBiome2() // carica la spiegazione del secondo livello
34    {
35        SceneManager.LoadScene(3);
36        mostraCursore();
37    }
38
39    public void StartLevel2() //carica il secondo livello
40    {
41        SceneManager.LoadScene(4);
42        mostraCursore();
43    }
44
45    public void LoadBiome3() // carica la spiegazione del terzo livello
46    {
47        SceneManager.LoadScene(5);
48        mostraCursore();
49    }
50
51    public void StartLevel3() //carica il terzo livello
52    {
53        SceneManager.LoadScene(6);
54        mostraCursore();
55    }
56
57    public void Victory() //carica la schermata di vittoria
58    {
59        SceneManager.LoadScene(7);
60        mostraCursore();
61    }
62
63    public void LoseKey() //carica la schermata di perdita per la chiave falsa
64    {
65        SceneManager.LoadScene(8);
66        mostraCursore();
67    }
68
69    public void LoseLives() //carica la schermata di perdita per l'esaurimento delle vite
70    {
71        SceneManager.LoadScene(9);
72        mostraCursore();
73    }
74
75    private void mostraCursore() // mostra il cursore per poter interagire con i bottoni
76    {
77        Cursor.lockState = CursorLockMode.Confined;
78        Cursor.visible = true;
79    }
80 }
```

Figura 54 - Script gestione schermate

4.8 Script Gemme livelli

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using TMPro;
4  using UnityEngine;
5
6  // Script Unity (1 riferimento ad asset) | 0 riferimenti
7  public class BonusesLevel1 : MonoBehaviour
8  {
9      bool bonusHitted;
10     int punteggio = 0;
11     public TextMeshProUGUI points;
12     public AudioSource gems;
13     // Start is called before the first frame update
14     void Start()
15     {
16         bonusHitted = true;
17     }
18     // Message Unity | 0 riferimenti
19     private void OnTriggerEnter(Collider other)
20     {
21         if (other.gameObject.tag == "gemma") // se collido con un oggetto con il tag gemma la distruggo e aggiungo un punto
22         {
23             if (bonusHitted)
24             {
25                 gems.Play();
26                 Destroy(other.gameObject);
27                 bonusHitted = false;
28                 StartCoroutine(CollisionWithBonus());
29             }
30             if (punteggio == 6) // se il punteggio è 6 apro la porta
31             {
32                 GetComponent<OpenDoorLevel1>().OpenDoor1 = true;
33             }
34         }
35     }
36     // 1 riferimento
37     IEnumerator CollisionWithBonus() // coroutine che aggiorna il punteggio
38     {
39         punteggio++;
40         points.text = "Punteggio: " + punteggio.ToString();
41         yield return new WaitForSeconds(0.5f);
42         bonusHitted = true;
43     }
44 }
```

Figura 55 - Script gemme livello1

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using TMPro;
4  using UnityEngine;
5
6  // Script Unity (1 riferimento ad asset) | 0 riferimenti
7  public class BonusesLevel2 : MonoBehaviour
8  {
9      // Start is called before the first frame update
10     int punteggio = 0;
11     bool bonusHitted;
12     public GameObject keyCaveau;
13     public GameObject finalKey;
14     public TextMeshProUGUI points;
15     public AudioSource gems;
16     public AudioSource key;
17
18     // Messaggio Unity | 0 riferimenti
19     void Start()
20     {
21         bonusHitted = true;
22     }
23
24     // Messaggio Unity | 0 riferimenti
25     private void OnTriggerEnter(Collider other)
26     {
27         if (other.gameObject.tag == "gemma") // se collido con un oggetto con il tag gemma lo distruggo e aggiungo un punto
28         {
29             if (bonusHitted)
30             {
31                 gems.Play();
32                 Destroy(other.gameObject);
33                 bonusHitted = false;
34                 StartCoroutine(CollisionWithBonus());
35             }
36         }
37         if (other.gameObject.tag == "keyCaveau") // se collido con un oggetto con il tag keyCaveau lo distruggo e apro la porta associata
38         {
39             key.Play();
40             Destroy(other.gameObject);
41             GetComponent<OpenDoorsLevel2>().CaveauOpenDoor2 = true;
42         }
43         if (other.gameObject.tag == "finalKey") // se collido con un oggetto con il tag finalKey lo distruggo
44         {
45             key.Play();
46             Destroy(other.gameObject);
47         }
48         if (punteggio == 9 && finalKey.gameObject == null) // se ho preso tutte le gemme e la final key apro la porta finale
49         {
50             GetComponent<OpenDoorsLevel2>().FinalOpenDoor2 = true;
51         }
52
53         IEnumerator CollisionWithBonus() // coroutine che aggiorna il punteggio
54         {
55             punteggio++;
56             points.text = "Punteggio: " + punteggio.ToString();
57             yield return new WaitForSeconds(0.5f);
58             bonusHitted = true;
59         }
60     }
61 }
```

Figura 56 - Script gemme livello2

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using TMPro;
4  using UnityEngine;
5  using UnityEngine.SceneManagement;
6
7  public class BonusesLevel3 : MonoBehaviour
8  {
9      int punteggio = 0;
10     bool bonusHitted;
11     public GameObject keyDoor;
12     public GameObject keyCaveau;
13     public GameObject finalKey;
14     public TextMeshProUGUI points;
15     public AudioSource gems;
16     public AudioSource key;
17
18     private bool canFinalDoorOpen = false;
19
20     void Start()
21     {
22         bonusHitted = true;
23     }
24
25
26     private void OnTriggerEnter(Collider other)
27     {
28         if (other.gameObject.tag == "gemma")
29         {
30             if (bonusHitted)
31             {
32                 gems.Play();
33                 Destroy(other.gameObject);
34                 bonusHitted = false;
35                 StartCoroutine(CollisionWithBonus());
36             }
37         }
38
39         if (other.gameObject.tag == "FirstKey") // se collido con un oggetto con il tag FirstKey lo distruggo e apro la porta associata
40         {
41             key.Play();
42             Destroy(other.gameObject);
43             GetComponent<OpenDoorLevel3>().SecretOpenDoor3 = true;
44         }
45
46         if (other.gameObject.tag == "keyCaveau") // se collido con un oggetto con il tag keyCaveau lo distruggo e apro la porta associata
47         {
48             key.Play();
49             Destroy(other.gameObject);
50             GetComponent<OpenDoorLevel3>().CaveauOpenDoor3 = true;
51         }
52
53         if (other.gameObject.tag == "fakeKey") // se collido con un oggetto con il tag fakeKey mi carica la schermata di perdita
54         {
55             key.Play();
56             SceneManager.LoadScene(8);
57         }
58
59         if (other.gameObject.tag == "finalKey") // se collido con un oggetto con il tag finalKey lo distruggo
60         {
61             key.Play();
62             Destroy(other.gameObject);
63             canFinalDoorOpen = true;
64         }
65         if (punteggio == 11 && canFinalDoorOpen) // se ho preso tutte le gemme e la final key apro la porta finale
66         {
67             GetComponent<OpenDoorLevel3>().FinalOpenDoor3 = true;
68         }
69     }
70
71     IEnumerator CollisionWithBonus() // coroutine che aggiorna il punteggio
72     {
73         punteggio++;
74         points.text = "Punteggio: " + punteggio.ToString();
75         yield return new WaitForSeconds(0.5f);
76         bonusHitted = true;
77     }
78 }

```

Figura 57 - Script gemme livello3

Questi script vengono utilizzati per verificare che tutte le gemme appartenenti ai livelli siano state collezionate, inoltre verificano che le chiavi dei rispettivi livelli siano state prese e gestiscono lo spostamento in altre schermate tramite lo script MenuManager. Inoltre nello script del terzo livello viene anche gestita la schermata nel caso venga presa la chiave sbagliata. Inoltre comprende gli effetti audio per quando colleziona una gemma.

4.9 Script Gestione apertura porte livelli

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class OpenDoorLevel1 : MonoBehaviour
6  {
7      public GameObject Door1;
8      private Vector3 translateObj;
9      public bool OpenDoor1;
10     // Start is called before the first frame update
11     void Start()
12     {
13         // velocità di spostamento porta
14         translateObj = new Vector3(0, -0.1f, 0);
15     }
16
17     // Update is called once per frame
18     void Update()
19     {
20         // se la porta non è in quella posizione la trasla tramite il translateobj facendola scivolare verso il basso
21         if (Door1.GetComponent<Transform>().localPosition.y > -34f && OpenDoor1)
22         {
23             Door1.GetComponent<Transform>().Translate(translateObj);
24         }
25     }
26 }
```

Figura 58 - Script porte livello1

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class OpenDoorsLevel2 : MonoBehaviour
6  {
7      public GameObject FinalDoor2;
8      private Vector3 translateObj;
9      public bool FinalOpenDoor2;
10
11     public GameObject CaveauDoor2;
12     public bool CaveauOpenDoor2;
13     // Start is called before the first frame update
14     void Start()
15     {
16         // velocità di spostamento porta
17         translateObj = new Vector3(0, -0.1f, 0);
18     }
19
20     // Update is called once per frame
21     void Update()
22     {
23         // se la porta non è in quella posizione la trasla tramite il translateobj facendola scivolare verso il basso
24         if (FinalDoor2.GetComponent<Transform>().localPosition.y > -34f && FinalOpenDoor2)
25         {
26             FinalDoor2.GetComponent<Transform>().Translate(translateObj);
27         }
28         if (CaveauDoor2.GetComponent<Transform>().localPosition.y > -34f && CaveauOpenDoor2)
29         {
30             CaveauDoor2.GetComponent<Transform>().Translate(translateObj);
31         }
32     }
33 }
```

Figura 59 - Script porte livello2

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class OpenDoorLevel3 : MonoBehaviour
6  {
7      public GameObject FinalDoor3;
8      private Vector3 translateObj;
9      public bool FinalOpenDoor3;
10
11     public GameObject CaveauDoor3;
12     public bool CaveauOpenDoor3;
13
14     public GameObject SecretDoor3;
15     public bool SecretOpenDoor3;
16     // Start is called before the first frame update
17     void Start()
18     {
19         // velocità di spostamento porte
20         translateObj = new Vector3(0, -0.1f, 0);
21     }
22
23     // Update is called once per frame
24     void Update()
25     {
26         // se la porta non è in quella posizione la trasla tramite il translateobj facendola scivolare verso il basso
27         if (FinalDoor3.GetComponent<Transform>().localPosition.y > -34f && FinalOpenDoor3)
28         {
29             FinalDoor3.GetComponent<Transform>().Translate(translateObj);
30         }
31         if (CaveauDoor3.GetComponent<Transform>().localPosition.y > -34f && CaveauOpenDoor3)
32         {
33             CaveauDoor3.GetComponent<Transform>().Translate(translateObj);
34         }
35         if (SecretDoor3.GetComponent<Transform>().localPosition.y > -34f && SecretOpenDoor3)
36         {
37             SecretDoor3.GetComponent<Transform>().Translate(translateObj);
38         }
39     }
40 }
```

Figura 60 - Script porte livello3

Questi script non fanno nient'altro che dare l'istruzione di far scivolare le porte grazie alla variabile di tipo Vector3 translateObj

4.10 Script gestione velocità stelle

```
1  using System.Collections;
2  [using System.Collections.Generic;
3  using UnityEngine;
4
5  Script Unity | 0 riferimenti
6  public class VelocityStars : MonoBehaviour
7  {
8      bool VelocityPlus = false;
9
10     Messaggio Unity | 0 riferimenti
11     private void OnCollisionEnter(Collision collision)
12     {
13         // se collido con un oggetto con il tag stella rendo il player più veloce
14         if (collision.gameObject.tag == "stella")
15         {
16             VelocityPlus = true;
17         }
18     }
}
```

Figura 61 - Script gestione stelle

Questo script entra in gioco quando avviene una collisione con un oggetto chiamato “stella”, dopo la collisione la velocità del giocatore aumenterà di 5 per 3 secondo grazie alla coroutine CollisionWithStar() impostata nella classe PlayerMovement.cs

5 Test

5.1 Protocollo di test

Test Case:	TC-001	Nome:	Costruzione labirinto
Riferimento	REQ-001		
Descrizione	La base del gioco, La costruzione dei tre livelli		
Prerequisiti	- Layout personaggio completato		
Procedura	1. Aprire il Unity 2. Disegnare i tre livelli		
Risultati attesi	SI avrà la base del gioco completata		

Test Case:	TC-002	Nome:	Prima persona
Riferimento	REQ-002		
Descrizione	Il movimento del personaggio è di 360° in orizzontale e 180° e l'utente sposterà la visuale con il mouse		
Prerequisiti	- Layout personaggio completato		
Procedura	3. Aprire il gioco 4. Iniziale la partita 5. Muovere la visuale		
Risultati attesi	Facendo partire il gioco, l'utente è in grado di cambiare la visuale del personaggio utilizzando il mouse		

Test Case:	TC-003	Nome:	Movimento personaggio
Riferimento	REQ-003		
Descrizione	Durante la partita, l'utente può muovere il personaggio tramite "WASD" all'interno dei labirinti		
Prerequisiti	- Labirinti completati - Personaggio completato - Visuale personaggio completata		
Procedura	1. Aprire gioco 2. Iniziare la partita 3. Muovere il personaggio		

Test Case: Riferimento	TC-004 REQ-004	Nome:	Programmazione mostri
Descrizione	L'utente dovrà scappare dai guardiani dei biomi che avranno come obiettivo quello di far perdere tutte le vite all'utente		
Prerequisiti	<ul style="list-style-type: none"> - Raccolta template mostri - Labirinti completati - Personaggio completato 		
Procedura	<ol style="list-style-type: none"> 1. Aprire gioco 2. Iniziare la partita 3. Aspettare l'arrivo dei mostri 		
Risultati attesi	I mostri dovranno inseguire l'utente		

Test Case: Riferimento	TC-005 REQ-0055	Nome:	Vite personaggio
Descrizione	Quando l'utente viene toccato da un mostro perde una vita, quando perde tutte le vite il gioco finisce.		
Prerequisiti	<ul style="list-style-type: none"> - Personaggio completato - Mostri completati 		
Procedura	<ol style="list-style-type: none"> 1. Aprire il gioco 2. Iniziare la partita 3. Farsi colpito dai mostri 		
Risultati attesi	I mostri dovranno inseguire l'utente e se lo toccano, esso perde una vita		

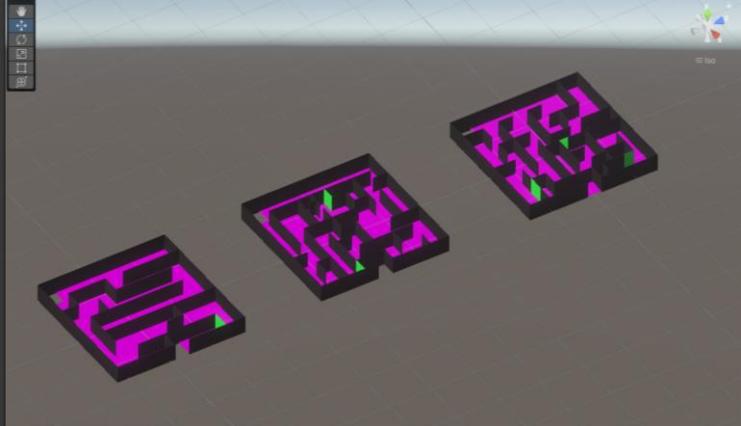
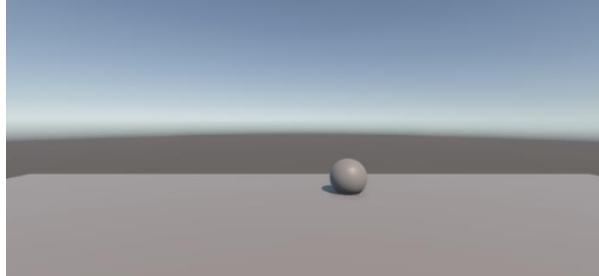
Test Case: Riferimento	TC-006 REQ-006	Nome:	Funzionamento schermata iniziale
Descrizione	L'utente ha la possibilità di inserire il proprio nome e visualizzare la classifica dei tempi migliori.		
Prerequisiti	<ul style="list-style-type: none"> - Design schermata iniziale completata - Gioco completato 		
Procedura	<ol style="list-style-type: none"> 1. Aprire il gioco 2. Inserire un nome 3. Schiacciare il bottone classifica 4. Schiacciare il bottone gioca 		
Risultati attesi	L'utente potrà visualizzare la classifica cliccando sul pulsante classifica, inoltre potrà inserire il suo nome ed impostare il volume, infine l'utente potrà iniziare una partita cliccando sul tasto "Gioca".		

Test Case: Riferimento	TC-007 REQ-007	Nome:	Funzionamento schermata impostazioni
Descrizione	L'utente ha la possibilità di aprire le impostazioni durante il corso della partita		
Prerequisiti	<ul style="list-style-type: none"> - Design schermata impostazioni completata - Gioco completato 		
Procedura	<ol style="list-style-type: none"> 1. Aprire il gioco 2. Aprire la schermata impostazione 3. Verificare i vari pulsanti 		
Risultati attesi	L'utente potrà visualizzare nella schermata impostazioni tramite che il tasto "ESC", potrà riprendere la partita cliccando sul bottone riprendi, uscire dal gioco schiacciando sul bottone "Torna al menù" ed impostare il volume		

Test Case: Riferimento	TC-008 REQ-008	Nome:	DataBase
Descrizione	Il DataBase contiene tutti i nomi dei giocatori e i loro relativi tempi che saranno inseriti nella classifica		
Prerequisiti	<ul style="list-style-type: none"> - Schermata delle impostazioni e schermata iniziale finite - Gioco completato 		
Procedura	<ol style="list-style-type: none"> 1. Aprire il gioco 2. Schiacciare sul pulsante "Classifica" 3. Vedere i posizionamenti 		
Risultati attesi	L'utente potrà visualizzare la classifica nella schermata iniziale con il tasto "Classifica" e vedere così il suo tempo effettivo.		

Test Case: Riferimento	TC-09 REQ-09	Nome:	Funzionamento schermata vincita
Descrizione	L'utente, quando finirà la partita, potrà vedere subito il suo tempo per completare il gioco.		
Prerequisiti	<ul style="list-style-type: none"> - Database finito - Gioco completato 		
Procedura	<ol style="list-style-type: none"> 1. Aprire il gioco 2. Finire il gioco 3. Controllare il tempo effettivo e la posizione 		
Risultati attesi	L'utente potrà visualizzare alla fine della partita il suo tempo e la sua posizione in classifica.		

5.2 Risultati test

Test Case	Esito	Risultati	Data
TC-001	Passato	 Prima visualizzazione: parte grezza dei livelli ancora da decorare	14.10.2022
TC-002	Passato	 Il giocatore sarà in grado di non visualizzare se stesso e vedere intorno a se in prima persona	18.11.2022
TC-003	Passato	Il giocatore sarà in grado di spostarsi nella mappa tramite i pulsanti WASD e le frecce della tastiera	07.10.2022
TC-004	Passato	Il giocatore vedrà i mostri che lo rincorrono in giro per la mappa	14.10.2022
TC-005	Passato	 Punteggio:  Il giocatore perderà vite se un mostro dovesse colpirlo	21.10.2022

TC-006	Passato	<p>Il giocatore sarà in grado di inserire un proprio nome, far partire il gioco, regolare il volume ed uscire dal gioco</p>	21.10.2022
TC-007	Passato	<p>Il giocatore, premendo il tasto "esc" sarà in grado di riprendere la partita, tornare al menù e regolare il volume</p>	21.12.2022
TC-008	Non Passato	Purtroppo per la mancanza del mio collega durante più della metà del progetto non ho avuto abbastanza tempo per utilizzare il DataBase per la memorizzazione del tempo dei giocatori nel completamento del gioco	23.12.2022
TC-009	Parzialmente Passato	<p>La schermata viene visualizzata e il giocatore può interagire correttamente con i pulsanti ma non può vedere ne il suo tempo ne la posizione in classifica</p>	23.12.2022

5.3 Mancanze/limitazioni conosciute

La mancanza principale del nostro progetto è stata la mancanza del mio collega di progetto per malattia. Purtroppo, sono rimasto da solo a fine ottobre dovendo svolgere il resto del progetto senza il supporto del mio collega, i suoi lavori li ho dovuti svolgere io stesso non riuscendo a finire per poco l'intero progetto come stabilito nel QdC (mancanza DataBase).

Con la presenza del mio collega sicuramente saremmo riusciti a terminare tutto ciò che ci eravamo prefissati a settembre.

6 Consuntivo

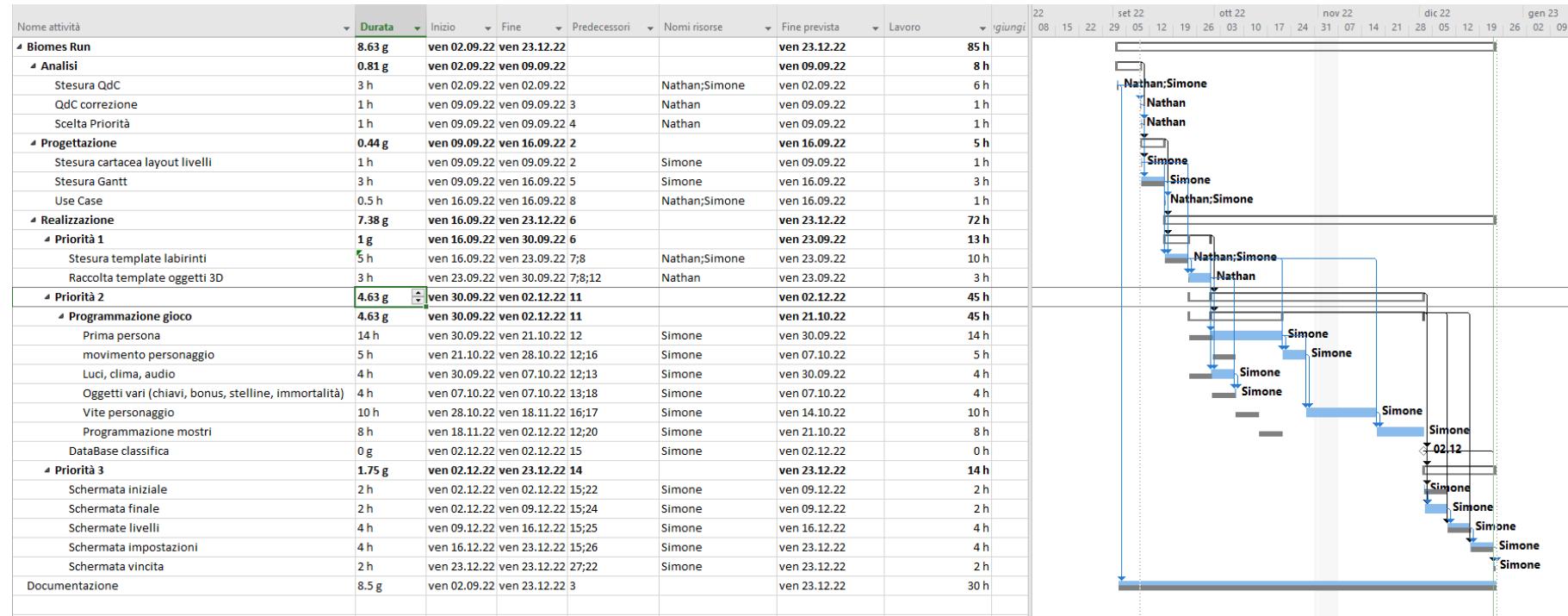


Figura 62 - Gantt Consuntivo

7 Conclusioni

7.1 Sviluppi futuri

Questo progetto può essere migliorato in moltissimi modi aggiungendo tantissime funzionalità, come per esempio dei nuovi livelli più complessi, oppure con delle versioni differenti del gioco (a tempo, senza tempo, con nemici, senza nemici, dare possibilità al personaggio di difendersi dai nemici tramite superpoteri, nemici più cattivi e molte altre).

7.2 Considerazioni personali

In conclusione, posso dire che questo progetto mi ha insegnato a programmare su Unity tramite C#, insegnandomi tutte le funzionalità di base unite alle funzionalità complesse. Inoltre, questo progetto mi ha insegnato ad essere molto ordinato e organizzato creando una vera e propria gerarchia di attività da svolgere durante le giornate lavorative. Inoltre, grazie a questo progetto ho potuto aumentare la mia capacità di lavorare in gruppo (fino a quando ho potuto)

8 Glossario

Nome	Definizione
Array	Insieme di elementi dello stesso tipo, identificati da un nome
Assets	Cartella contenente le parti principali del progetto (scene, modelli, oggetti 3D,...)
Consuntivo	risultati di un dato periodo di attività
Database	archivio di dati strutturato in tabelle contenenti attributi per la gestione e l'aggiornamento delle informazioni e la gestione di ricerche complesse
Diagramma	Disegno o rappresentazione schematica
Empty Object	Oggetto vuoto che andrà a contenere dei sotto oggetti in Unity (usato per ordinare le gerarchie di oggetti)
Gantt	Diagramma per pianificare un intero progetto e dividerlo in varie attività rendendolo più organizzato
Hardware	Parte fisica di un sistema
Rigidbody	Proprietà che rende il “corpo” dell’oggetto importato rigido in maniera da non passarci attraverso
Script	File che contiene righe di codice
Sprite	Immagine bidimensionale di gioco
Use case	Caso d’uso, contiene tutte le funzionalità di un progetto

9 Bibliografia

9.1 Sitografia

- <https://kenney.nl/assets/nature-pack> 16.09.2022
- <https://www.piskelapp.com/> 14.10.2022
- <https://medium.com/codex/creating-a-lives-counter-ui-element-in-unity-9d48b485555e> 11.11.2022
- <https://www.youtube.com/watch?v=f473C43s8nE> 16.12.2022
- <https://www.youtube.com/watch?v=yWCHaTwVblk> 21.12.2022
- <https://sketchfab.com/store/3d-models> 09.12.2022
- <https://assetstore.unity.com/> 09.12.2022