

## **SkiFree Remake**

**Titolo del progetto:** SkiFree Remake  
**Alunno/a:** Christian Monga  
**Classe:** Info 3BB  
**Anno scolastico:** 2022/2023  
**Docente responsabile:** Geo Petrini

1	Introduzione .....	4
1.1	Informazioni sul progetto .....	4
1.2	Abstract .....	4
1.2.1	Situazione iniziale .....	4
1.2.2	Approccio .....	4
1.2.3	Risultati .....	4
1.3	Scopo .....	5
1.3.1	Scopi lavorativi .....	5
1.3.2	Scopi didattici .....	5
2	Analisi .....	6
2.1	Analisi del dominio .....	6
2.2	Analisi e specifica dei requisiti .....	6
2.3	Use case .....	9
2.4	Pianificazione .....	10
2.5	Analisi dei mezzi .....	11
2.5.1	Software .....	11
2.5.2	Hardware .....	11
3	Progettazione .....	12
3.1	Design del gioco completo .....	12
3.1.1	Menu principale .....	12
3.1.2	Menu di pausa .....	13
3.1.3	Gioco .....	14
4	Implementazione .....	15
4.1	Creazione progetto Unity 2D .....	15
4.2	Setup ambiente di lavoro .....	15
4.3	Menu principale .....	16
4.3.1	Script MenuPrincipale .....	17
4.4	Menu di pausa .....	18
4.4.1	Script MenuPausa .....	18
4.5	Camera segue il personaggio .....	19
4.5.1	Creazione personaggio .....	19
4.5.2	Implementazione della camera .....	20
4.5.3	Script PlayerMovement .....	21
4.6	Parte iniziale con ostacoli predefiniti .....	23
4.7	Generazione Mappa .....	24
4.7.1	Script LevelGenerator .....	24
4.7.2	Lupo .....	25
4.7.3	Yeti .....	26
4.7.4	Script Yeti .....	26
4.7.5	GestoreOstacoli .....	26
4.8	Dati di gioco .....	27
4.8.1	Script Dati .....	27
4.8.2	Script TimerController .....	27
5	Test .....	28
5.1	Protocollo di test .....	28
5.2	Risultati test .....	32
5.3	Mancanze/limitazioni conosciute .....	33
5.3.1	Movimento del personaggio .....	33
5.3.2	Ostacoli .....	33
5.3.3	Modello sciatore .....	33
5.3.4	Lupi .....	33
6	Consuntivo .....	34
7	Conclusioni .....	36
7.1	Sviluppi futuri .....	36
7.2	Considerazioni personali .....	36
8	Glossario .....	37
9	Bibliografia .....	38
9.1	Sitografia .....	38

9.2	Indice delle figure .....	39
10	Allegati .....	40

## **1 Introduzione**

---

### **1.1 Informazioni sul progetto**

Allievo: Christian Monga  
Docente responsabile: Geo Petrini  
Scuola: CPT Trevano  
Sezione: Informatica  
Classe: I3BB

Inizio Progetto: 09.09.2022

Fine Progetto: **23.12.2022**

### **1.2 Abstract**

#### **1.2.1 Situazione iniziale**

La consegna di questo progetto è quella di realizzare un remake del gioco SkiFree utilizzando Unity. Questo gioco consiste nello sciare per una montagna schivando gli ostacoli statici e dinamici cercando di ottenere il miglior punteggio possibile.

Al momento non esiste una versione evoluta di questo gioco, quindi questa potrebbe essere una buona opportunità per realizzarne una di qualità.

#### **1.2.2 Approccio**

Per questo progetto sono necessarie delle buone conoscenze nell'ambito della programmazione in C#, visto che la programmazione in Unity non è semplice.

Per la realizzazione di questo progetto è stato necessario pianificare la suddivisione dei requisiti in gruppi comuni, così da riuscir a creare degli script in modo ordinato.

Infatti per prima cosa è meglio creare i menu, poi il personaggio (con i suoi movimenti), poi la mappa e infine lo spazio che contiene i dati di gioco.

#### **1.2.3 Risultati**

Gli obiettivi di questo progetto sono stati raggiunti, il gioco è funzionante e rispecchia una buona parte dei requisiti, in conclusione ci si può giocare tranquillamente senza problemi di funzionamento.

### **1.3 Scopo**

#### **1.3.1 Scopi lavorativi**

Imparare ad usare Unity 2D per riuscire a ricreare il gioco SkiFree partendo da 0.

#### **1.3.2 Scopi didattici**

Imparare a gestire un progetto completo in modo autonomo, prendendo i requisiti, creando una pianificazione, documentando il lavoro svolto con documentazione e diari, saper tenere una presentazione sul lavoro svolto e imparare a gestire il tempo.

## 2 Analisi

### 2.1 Analisi del dominio

Questo progetto consiste nel ricreare il vecchio gioco singleplayer SkiFree utilizzando l'applicativo Unity. Il prodotto finale potrà essere eseguito su un qualsiasi computer windows 10 e verrà prettamente utilizzato a scopo ricreativo.

### 2.2 Analisi e specifica dei requisiti

<b>ID: REQ-001</b>	
<b>Nome</b>	Visuale del gioco
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Il gioco deve essere visualizzato in full screen
<b>Sotto requisiti</b>	
<b>001</b>	Deve essere in 2D

<b>ID: REQ-003</b>	
<b>Nome</b>	La mappa è infinita
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
<b>001</b>	La mappa viene generata automaticamente in base alla posizione del personaggio

<b>ID: REQ-004</b>	
<b>Nome</b>	La mappa contiene degli ostacoli
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
<b>001</b>	Vengono generati casualmente all'interno della mappa
<b>002</b>	<p>Gli ostacoli devono essere:</p> <ul style="list-style-type: none"> <li>• Alberi (quelli piccoli vengono bruciati quando saltati ad alta velocità)</li> <li>• Sassi</li> <li>• Seggiovie</li> <li>• Altri sciatori o snowboarders</li> <li>• Trampolini</li> <li>• Lupi</li> </ul>

<b>ID: REQ-005</b>	
<b>Nome</b>	I movimenti dello sciatore vengono dettati dal mouse
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
<b>001</b>	La rotazione del personaggio deve essere fluida
<b>002</b>	Se lo sciatore è messo orizzontalmente si ferma
<b>003</b>	Con il click del tasto sinistro lo sciatore salta
<b>004</b>	Se si tiene il tasto destro premuto in volo lo sciatore fa il mortale
<b>005</b>	Lo sciatore in volo non può cambiare traiettoria
<b>006</b>	Se atterra male o colpisce gli ostacoli si schianta, si ferma e dopo si rialza

<b>ID: REQ-006</b>	
<b>Nome</b>	Jeti
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
<b>001</b>	Compare uno jeti ogni 2500m percorsi
<b>002</b>	Lo jeti vuole mangiare lo sciatore
<b>003</b>	Lo jeti è più veloce dello sciatore

<b>ID: REQ-007</b>	
<b>Nome</b>	I dati dell'utente vengono memorizzati e mostrati
<b>Priorità</b>	2
<b>Versione</b>	1.0
<b>Note</b>	
<b>Sotto requisiti</b>	
<b>001</b>	Deve essere salvato il tempo di gioco della partita
<b>002</b>	Deve essere salvata la distanza percorsa
<b>003</b>	Deve essere mostrata la velocità attuale
<b>004</b>	Devono essere mostrato il punteggio

<b>ID: REQ-008</b>	
<b>Nome</b>	Il gioco finisce quando lo sciatore viene mangiato
<b>Priorità</b>	1
<b>Versione</b>	1.0
<b>Note</b>	Lo jeti mangia lo sciatore



<b>ID: REQ-009</b>	
<b>Nome</b>	Menù di pausa
<b>Priorità</b>	2
<b>Versione</b>	1.0
<b>Note</b>	Cliccando “esc” si apre il menù di pausa
<b>Sotto requisiti</b>	
<b>001</b>	Blocca il gioco
<b>002</b>	Si può chiudere l'applicazione
<b>003</b>	Si può tornare all'inizio

## 2.3 Use case

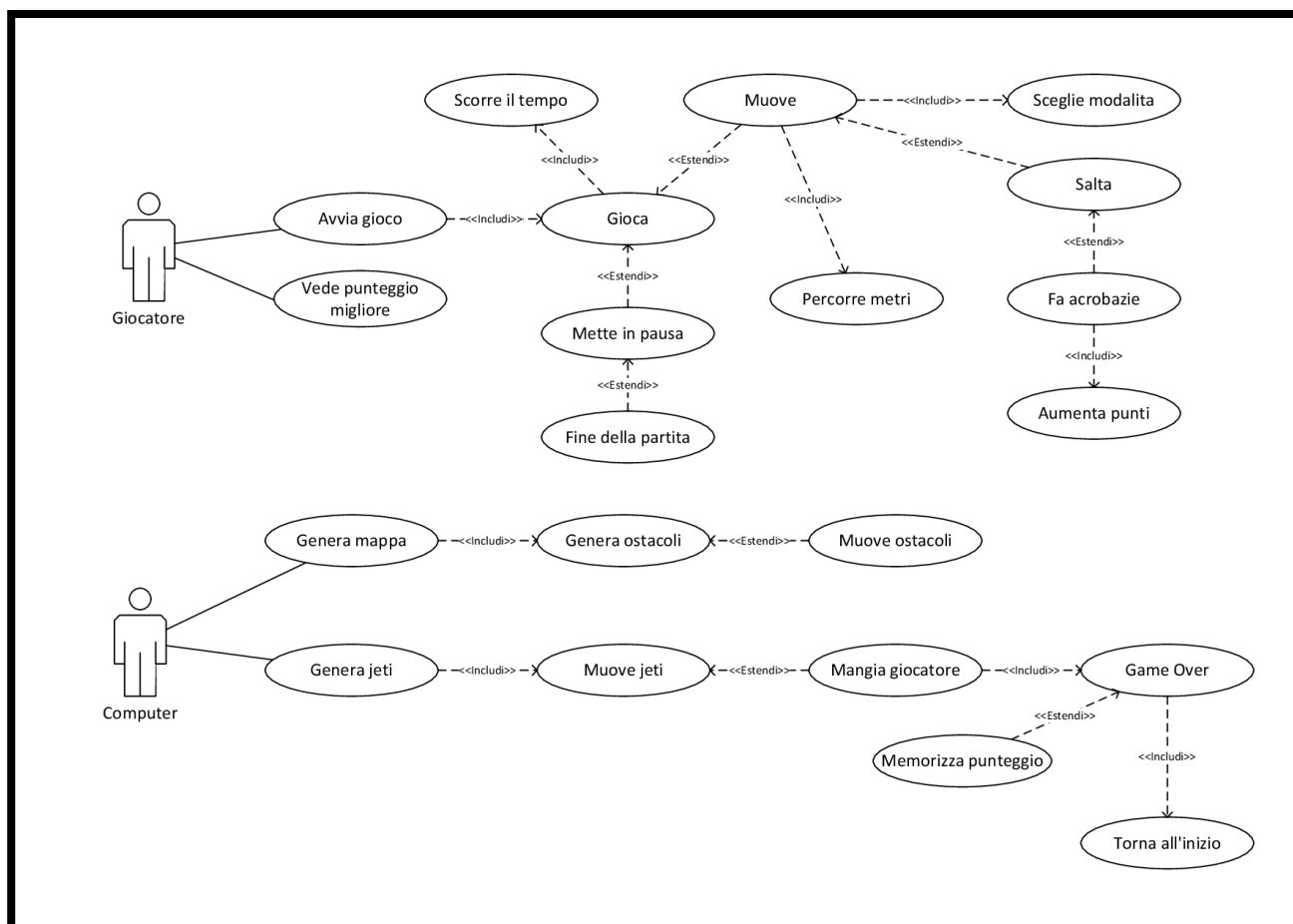


Figura 1 UML Use Case del progetto

## 2.4 Pianificazione

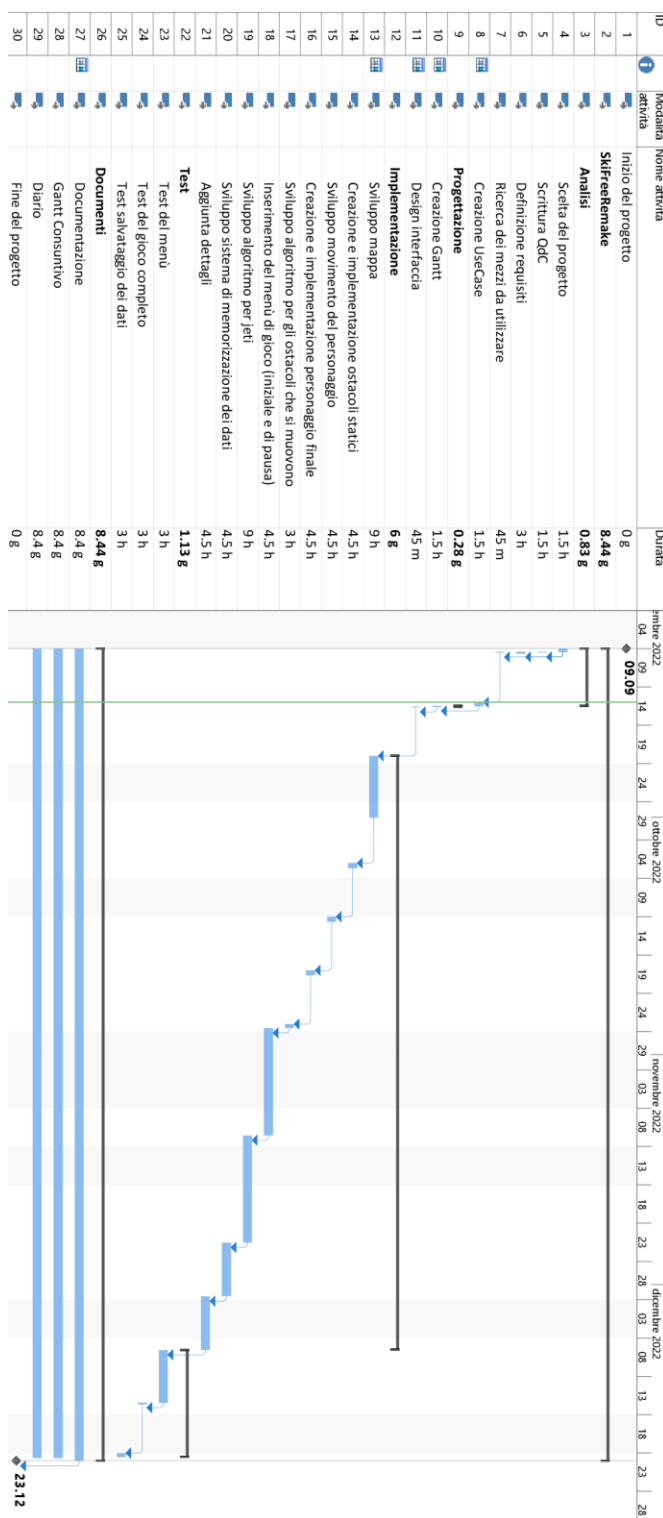


Figura 2 Diagramma di Gantt Iniziale

Per questo progetto utilizzo la pianificazione Iterativa evolutiva

## **2.5 Analisi dei mezzi**

Per la realizzazione di questo progetto è stato utilizzato un PC scolastico con Unity e Visual Studio Code installati.

### **2.5.1 Software**

- Unity 2022.1.1f1
- Microsoft Visio
- Microsoft Project
- Microsoft Word
- Visual Studio Code con estensione C# v1.25.2
- Github con Git
- Paint 3D

### **2.5.2 Hardware**

Il progetto dovrà essere eseguito su una macchina con windows 10 installato.

HW computer utilizzato:

- CPU: I7-9700
- Scheda video RTX-2060
- 32GB di RAM DDR4

### 3 Progettazione

In questo capitolo si può vedere come sono state progettate le interfacce del gioco.

#### 3.1 Design del gioco completo

Il gioco ha tre interfacce differenti:

- Il menu principale
- Il menu di pausa
- Il gioco

##### 3.1.1 Menu principale

Il menu principale è la prima interfaccia visualizzata all'apertura del gioco ed è molto semplice, contiene soltanto due pulsanti: GIOCA ed ESCI che permettono di iniziare una nuova partita o di uscire dall'applicazione.

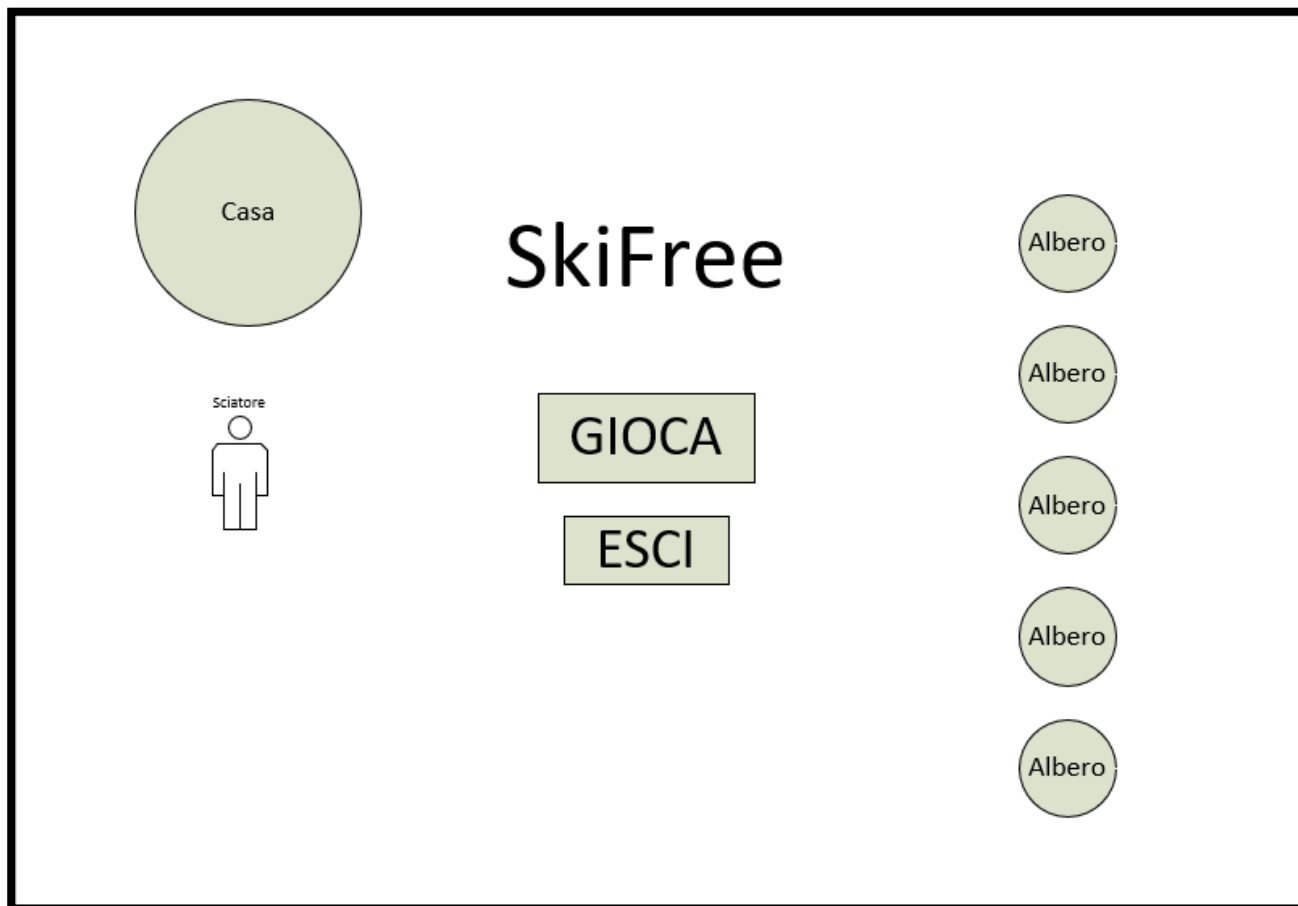


Figura 3 Interfaccia menu principale

Sullo sfondo è presente la schermata iniziale della partita.

### 3.1.2 Menu di pausa

Il menu di pausa si apre premendo il tasto “esc” della tastiera e permette all’utente di tornare al menu principale o di riprendere la partita.

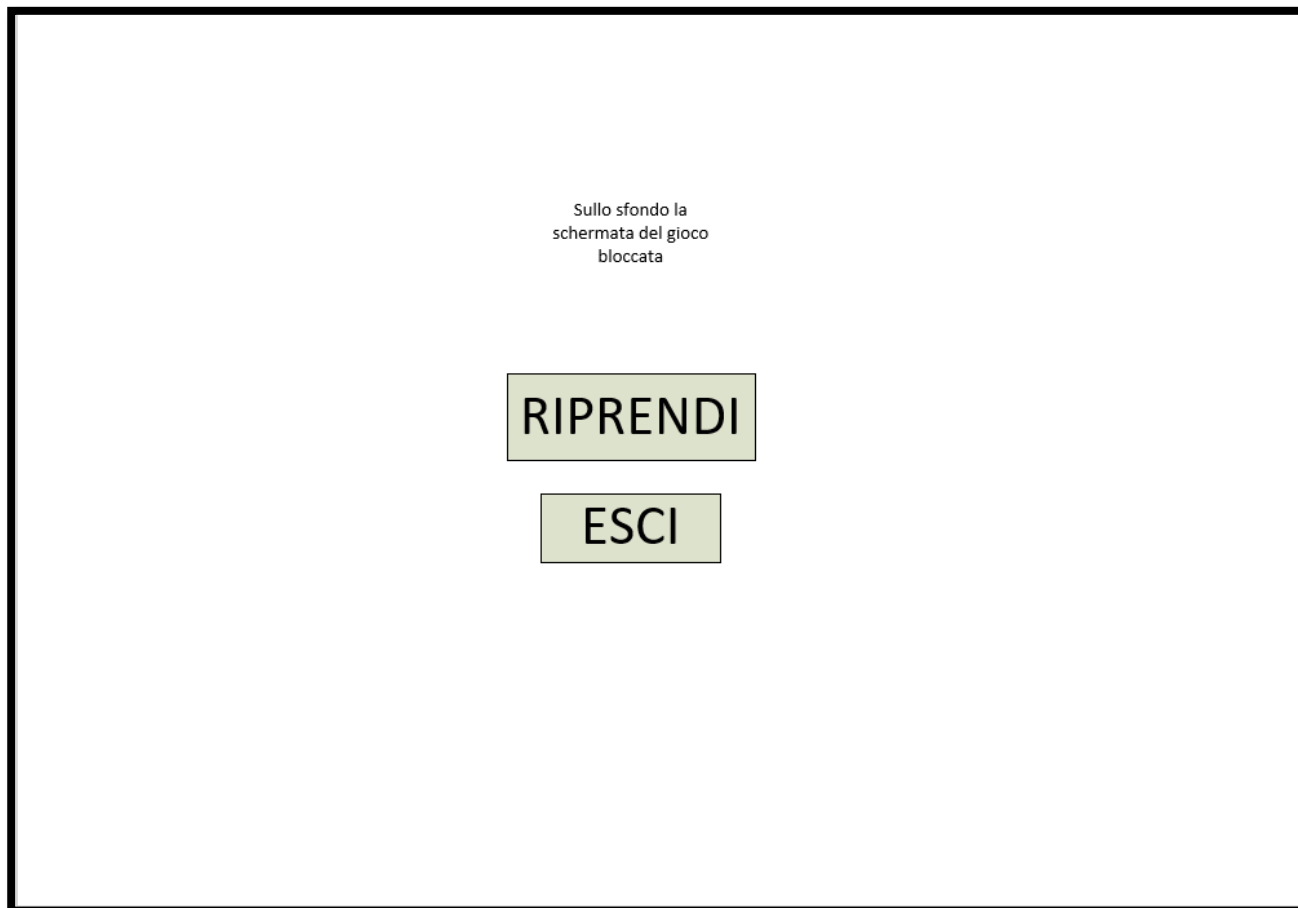


Figura 4 Schermata di pausa

Sullo sfondo si deve vedere la schermata del gioco bloccata e più appannata.

### 3.1.3 Gioco

Durante la partita è necessario visualizzare in alto a sinistra dello schermo i vari dati di progresso dell'utente. Si devono visualizzare i metri percorsi, il punteggio e il tempo di gioco.

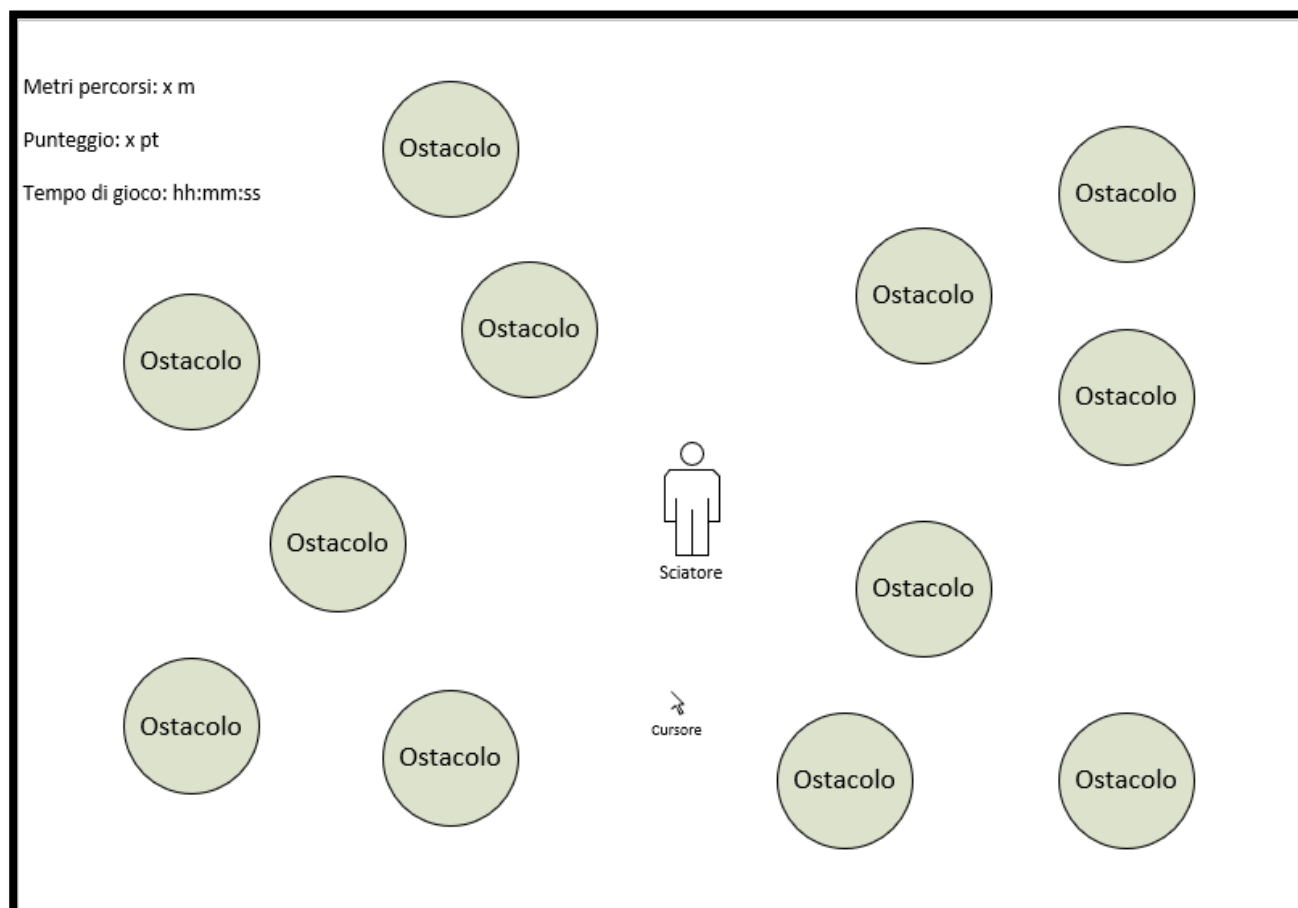


Figura 5 Interfaccia della partita

## 4 Implementazione

### 4.1 Creazione progetto Unity 2D

Per prima cosa bisogna creare un nuovo progetto di Unity in 2D e bisogna chiamarlo SkiFree-Remake.

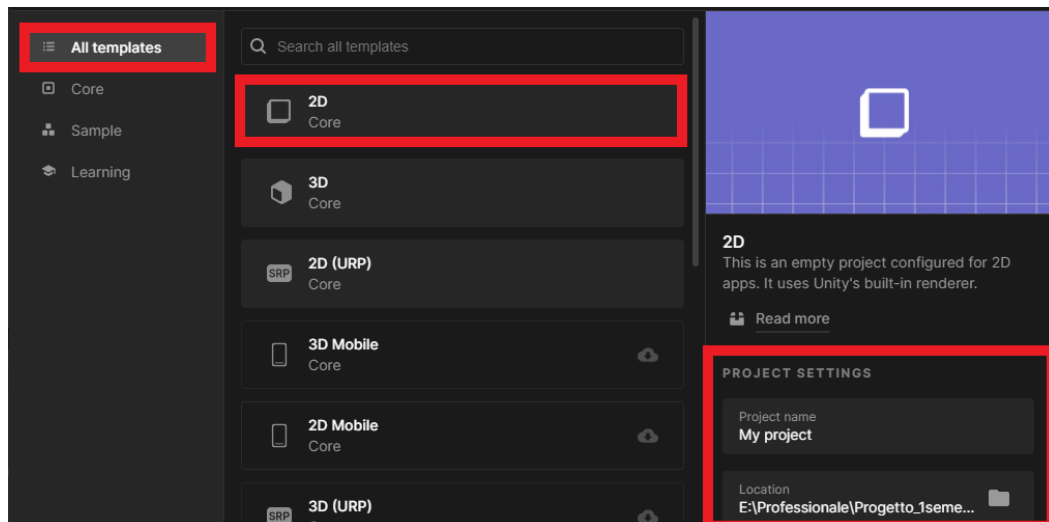


Figura 6 Creazione progetto Unity 2D

### 4.2 Setup ambiente di lavoro

Una volta creato il progetto di Unity, prima di iniziare a lavorare, si devono creare specifiche cartelle che contreranno i vari script, sprite, prefabs, animazioni e scene.

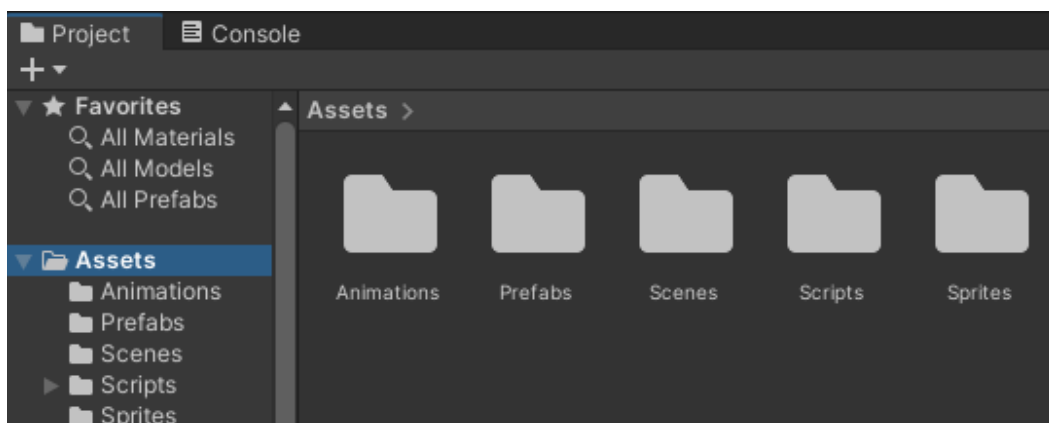


Figura 7 Cartelle necessarie

Queste cartelle porteranno ordine all'interno del proprio progetto. Inoltre, un'altra buona cosa da fare prima di iniziare è quella di rinominare la scena base con un nome esplicativo, per esempio Gioco o ScenaGioco.

### 4.3 Menu principale

Per creare il menu principale bisogna aggiungere una nuova scena.  
Dopo aver creato la scena bisogna creare un ordinamento di esse, per farlo bisogna andare sotto il menu File e poi Build Settings.

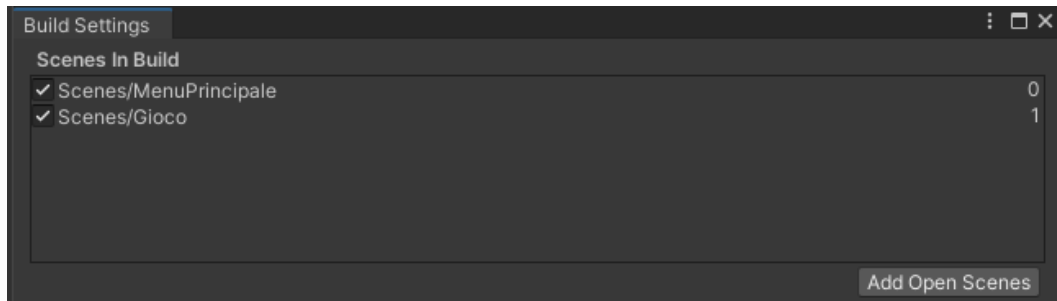


Figura 8 Build Settings delle scene

In questo punto si devono ordinare le scene e per comodità si imposta la scena MenuPrincipale come prima e la scena Gioco come seconda.

A questo punto si può iniziare a lavorare sul menu principale.

Questa scena dovrà contenere due pulsanti: GIOCA ed ESCI, essi permetteranno all'utente di avviare una partita o di uscire dall'applicazione.

Per creare il menu basta aggiungere un GameObject Canvas con all'interno un altro GameObject chiamato Menu e all'interno si possono applicare i due pulsanti, lo sfondo e l'immagine di SkiFree.

In questo modo si può ottenere un risultato grafico simile a questo:



Figura 9 Scena del menu principale



#### 4.3.1 Script MenuPrincipale

Per gestire i due pulsanti c'è bisogno di uno script; quindi, si deve creare lo script MenuPrincipale. Dentro questo script dovranno essere presenti tanti metodi quanti i pulsanti che si vogliono gestire.

Nel metodo dedicato al pulsante GIOCA bisogna inserire il codice che permette il caricamento della scena principale. Mentre nel metodo per il pulsante ESCI si deve inserire il codice per chiudere completamente l'applicazione.

Per far sì che lo script funzioni correttamente nell'Inspector dei due pulsanti bisogna prima assegnare lo script alla proprietà OnClick() e in seguito bisogna selezionare il metodo corrispondente a quel pulsante.

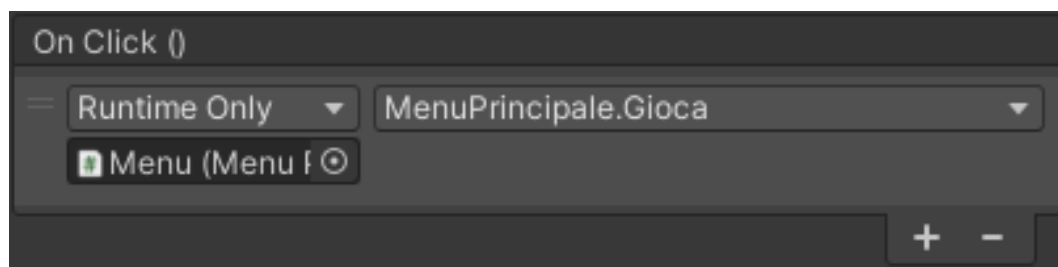


Figura 10 OnClick() dei pulsanti

Infine allo script principale si può impostare uno sfondo<sup>1</sup> e per farlo basta scaricarlo e inserirlo all'interno del GameObject menu come immagine.

<sup>1</sup> <https://www.vecteezy.com/vector-art/4371984-winter-landscape-with-mountains-pines-forest-snow-falling-background-for-your-arts>

#### 4.4 Menu di pausa

Per il menu di pausa invece bisogna tornare sotto la scena del gioco.

Qui bisogna fare come per la scena principale e bisogna creare un Canvas con due pulsanti all'interno.

Questi due pulsanti saranno RIPRENDI ed ESCI e serviranno per far riprendere la partita in corso o per tornare al menu principale.

Anche qui, come in precedenza, si deve creare uno script apposito, MenuPausa, che gestirà questo menu.



Figura 11 Menu di pausa

##### 4.4.1 Script MenuPausa

In questo script si dovranno sempre creare i due metodi che permetteranno il funzionamento dei due pulsanti.

Il metodo Riprendi(), quando viene richiamato, deve disattivare il Canvas del menu di pausa, deve reimpostare il tempo di Unity al valore di default e deve segnalare che il gioco non si trova più in pausa.

Il metodo TornaAlMenu() deve caricare la scena del menu principale e deve riportare il valore del tempo di Unity a quello di default, in questo modo se dal menu principale si vuole ricominciare una partita lo si potrà fare.

Infine deve esserci anche il metodo che attivi il menu mentre si è in partita. Questo menu si deve attivare in caso di pressione del tasto ESC della tastiera dell'utente.

Per controllare se l'utente preme o no il tasto ESC bisogna, nel metodo Update(), continuare a controllare se esso viene premuto. Se viene premuto mentre è in gioco in corso caricherà il menu, mentre se viene premuto quando il menu è aperto ricaricherà la partita.

Il metodo Pausa() deve attivare il Canvas del menu, deve bloccare il tempo e deve segnalare che il gioco si trova in pausa.

## 4.5 Camera segue il personaggio

Per questo gioco è strettamente necessario che la camera segua e punti sempre al player.

### 4.5.1 Creazione personaggio

Per creare il personaggio si deve creare un Empty GameObject contenente un altro Empty GameObject. (Player contiene P\_Body)

Al primo GameObject si applicheranno tutti i vari componenti per realizzare il personaggio mentre al secondo si applicherà solamente il modello grafico del personaggio.

Per prima cosa si deve implementare il modello grafico del personaggio e per farlo bisogna inizialmente scaricare uno sprite e questo si può fare da questo [link](https://github.com/aidan-waite/OpenSkiFree/blob/main/Assets/Art/skifreesprites.png)<sup>2</sup>.

Questo è uno sprite multiplo e per poterlo utilizzare bisogna dividerlo in tanti piccoli sprite singoli.

#### 4.5.1.1 Suddivisione sprite multiplo

Per suddividere uno sprite multiplo basta semplicemente impostare la proprietà Sprite Mode a Multiple, poi bisogna aprire lo sprite editor, premere slice e separare tutti gli sprite.

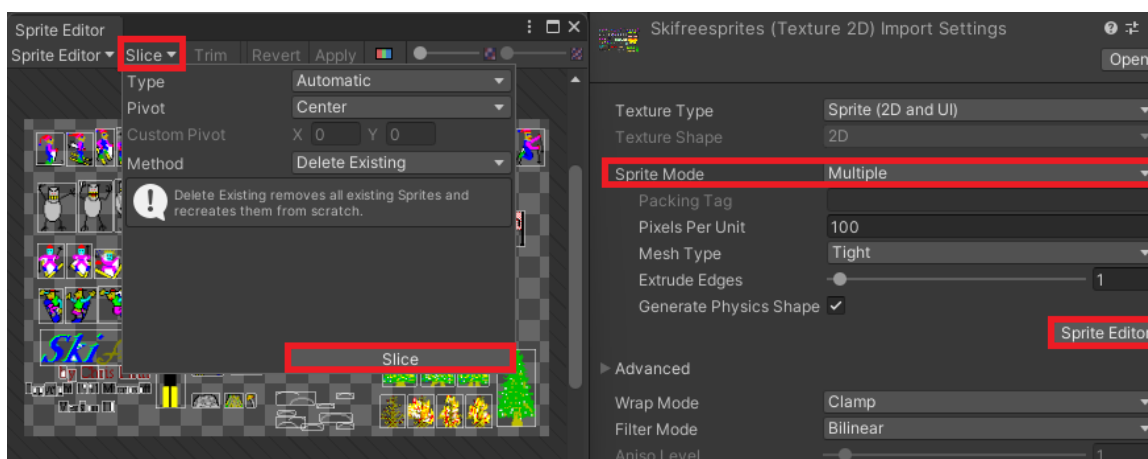


Figura 12 Separazione sprites

Dopo averli separati si potranno usare singolarmente.

In seguito si dovrà trascinare lo sprite dello sciatore all'interno della proprietà Sprite Renderer del GameObject creato in precedenza.

Dopo aver inserito il modello bisogna andare a lavorare sul GameObject padre. Qui si dovranno inizialmente aggiungere il Rigidbody2D e il PolygonCollider2D che permetteranno al personaggio di avere un corpo e di fare collisioni con gli altri ostacoli.

È importante impostare la proprietà IsTrigger, nel PolygonCollider2D, a false, altrimenti non si scontrerà con niente. Inoltre si deve anche impostare il valore del Gravity Scale, in Rigidbody2D a 0, altrimenti il player cadrà all'infinito.

<sup>2</sup> <https://github.com/aidan-waite/OpenSkiFree/blob/main/Assets/Art/skifreesprites.png>

#### 4.5.2 Implementazione della camera

A questo punto si può implementare la camera.

Per farlo bisogna scaricare una Cinemachine, essa si trova sotto il percorso:

Window/Package Manager/Unity Registry.

Dopo averla installata bisogna aggiungere alla scena una 2D Camera del package Cinemachine e, per fissarla sul player, bisogna prendere il GameObject di esso e bisogna trascinarlo nella proprietà Follow.

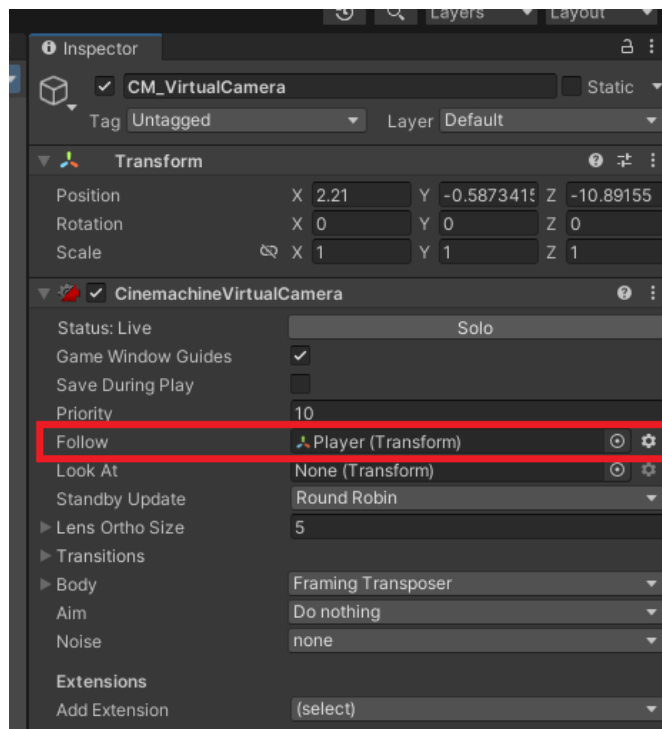


Figura 13 Camera segue il player

Cinemachine è molto comoda perché semplicemente trascinando il GameObject del player nella proprietà apposita, essa lo seguirà e, al contrario degli altri metodi, qui non c'è bisogno di scrivere alcun codice.

### 4.5.3 Script PlayerMovement

Infine per gestire il movimento del player serve uno script dedicato che deve essere aggiunto al GameObject padre.

In questo script sono presenti i metodi che permettono di conoscere la posizione del mouse (PrendiPosizioneMouse) e della videocamera, che permettono al player stesso di puntare (guardare) sempre verso il cursore (PuntaMouse), che permettono di gestire i salti (salto) e le collisioni (OnCollisionEnter2D) e che infine permettono il movimento corretto del player (SeguiMouseDelay).

In particolare per far sì che il player punti sempre al cursore si deve utilizzare la proprietà `up` del transform del player stesso. Quindi se si lasciasse così com'è di default sarebbe il punto più alto dello sprite a puntare il mouse. Per evitare questo problema si deve semplicemente cambiare la rotazione del GameObject `P_Body` sull'asse delle X; bisogna impostarla a 180.

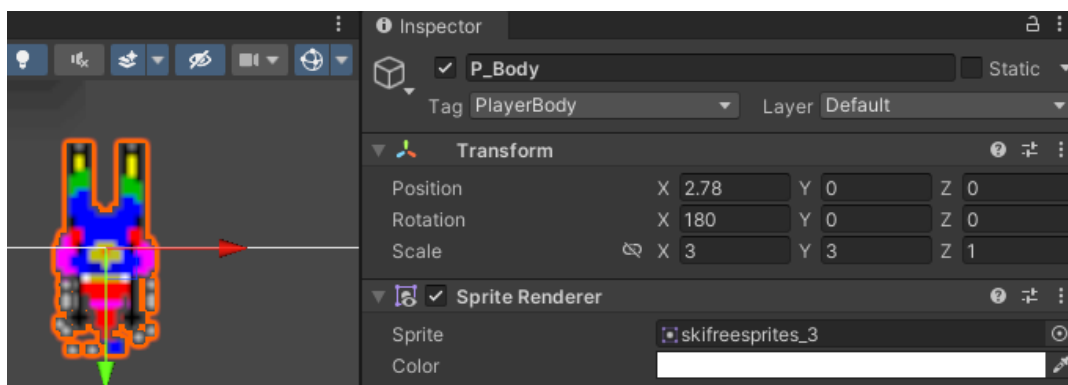


Figura 14 Rotazione di partenza del personaggio

In questo modo saranno gli sci a puntare verso il cursore e quindi si avrà lo sciatore che segue in modo corretto il cursore.

Tutti questi metodi vengono richiamati nel metodo principale `Update()` che si occupa di controllare la posizione del mouse e di farla seguire dal player.

#### 4.5.3.1 Movimento a terra

Per il movimento dello sciatore a terra ci sono una serie di metodi che permettono, come scritto in precedenza, di prendere la posizione del cursore e della camera.

Mettendo insieme queste coordinate si potrà far puntare il player al cursore e si potrà far muovere il player nella stessa direzione del cursore.

Per il movimento del player si deve creare un'animazione con all'interno il singolo sprite che mostra lo sciatore mentre scia. Dopo averla inserita all'interno dell'animazione si deve aggiungere il componente Animator al GameObject `P_Body`, qui si deve impostare come unica animazione quella appena creata, in questo modo scierà sempre.

#### 4.5.3.2 Salto

Inoltre in questo script deve essere contenuto anche il codice per i salti, il player, quando l'utente preme il tasto sinistro del mouse, deve saltare e quindi viene richiamata la Coroutine che gestisce questa azione. Se, mentre si trova in aria, viene premuto il tasto destro del mouse il player dovrà svolgere delle piccole rotazioni fino a completare un intero giro prima di atterrare e nel caso in cui atterrasse correttamente devono essere assegnati dei punti.

Per far sì che il salto funzioni correttamente bisogna impostare la proprietà `IsTrigger` dei vari ostacoli a `true` e bisogna spostare il player più vicino alla camera sull'asse delle `z`. Facendo in questo modo si andrà a creare un'illusione del salto dove il player passerà sopra gli alberi.

Il player mentre si trova in volo non può cambiare direzione e per questo esso manterrà la sua posizione orizzontale iniziale e cambierà solamente quella verticale. In questo modo non seguirà e non punterà più il cursore.

#### 4.5.3.3 Collisioni

Questo script deve anche saper gestire le collisioni del player con gli altri oggetti presenti nel progetto. Se si scontra con un albero oppure se atterra in modo non corretto dopo un salto, dovrà fermarsi per pochi secondi prima di poter ripartire, come a simulare uno scontro/schianto.

Per gestire queste collisioni c'è il metodo apposito di Unity: `OnCollisionEnter2D(Collision2D c)`. In questo metodo viene specificato il comportamento del player in caso di collisione, ad ogni collisione viene fermato il player e viene iniziata la Coroutine che lo blocca per due secondi prima di farlo ripartire, mentre, in modo più specifico, se si scontra con un ostacolo dinamico la velocità di quell'ostacolo diventerà nulla. Infine se si scontrerà con lo Yeti verrà ricaricata la scena del menu principale e la partita terminerà.

Per queste collisioni bisogna aggiungere un'animazione al personaggio dove, al momento dello scontro, cambia posizione simulando una caduta. Per farlo bisogna creare un'animazione al `GameObject` del modello dello sciatore, in questa animazione si deve aggiungere solamente lo `sprite` del player a terra. Dopo averla inserita si deve tornare nell'Animator di `P_Body` e collegare l'animazione del movimento con quella dello scontro.

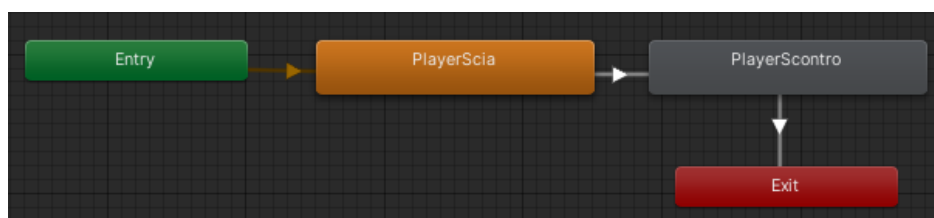


Figura 15 Animator del personaggio

Questo collegamento deve avvenire solamente nel momento in cui lo sciatore si scontra contro un ostacolo, perciò si deve aggiungere un nuovo parametro e lo si deve applicare al collegamento tra le due animazioni. Questo parametro verrà modificato nello script `PlayerMovement`, al momento di ogni collisione.

Infine bisogna fare in modo che quando l'utente preme il tasto `ESC` il gioco si deve fermare e quindi il player non deve più "obbedire" ai comandi dell'utente. Questo si fa con una semplice condizione che controlla se il gioco è in pausa oppure no.

#### 4.6 Parte iniziale con ostacoli predefiniti

Prima di creare l'algoritmo che permetterà la generazione della mappa bisogna creare una parte statica iniziale.

Inizialmente bisogna creare un Empty GameObject che conterrà tutto il contenuto per tenerlo più ordinato. In questa parte iniziale si deve inserire, intorno al player, degli alberi<sup>3</sup>, una casa<sup>4</sup> e un cartello di benvenuto<sup>5</sup>. Per farlo basterà scaricare gli sprite, come fatto in precedenza, e importarli all'interno del progetto. Questi sprite però, prima di poter essere utilizzati, dovranno essere trasformati in dei prefabs, così da poterci aggiungere i componenti necessari per le collisioni. Per trasformarli basta trascinare lo sprite nella sezione Hierarchy e ritrascinarlo all'interno della sua cartella apposita (Prefab).

Come per il player bisogna aggiungere a tutti gli ostacoli che si andranno a generare un PolygonCollider2D e un Rigidbody2D con le stesse configurazioni, l'unica cosa che cambia è la proprietà BodyType del Rigidbody2D che dovrà essere statica.

Dopo aver svolto questi passaggi si può creare una semplice parte iniziale come questa:

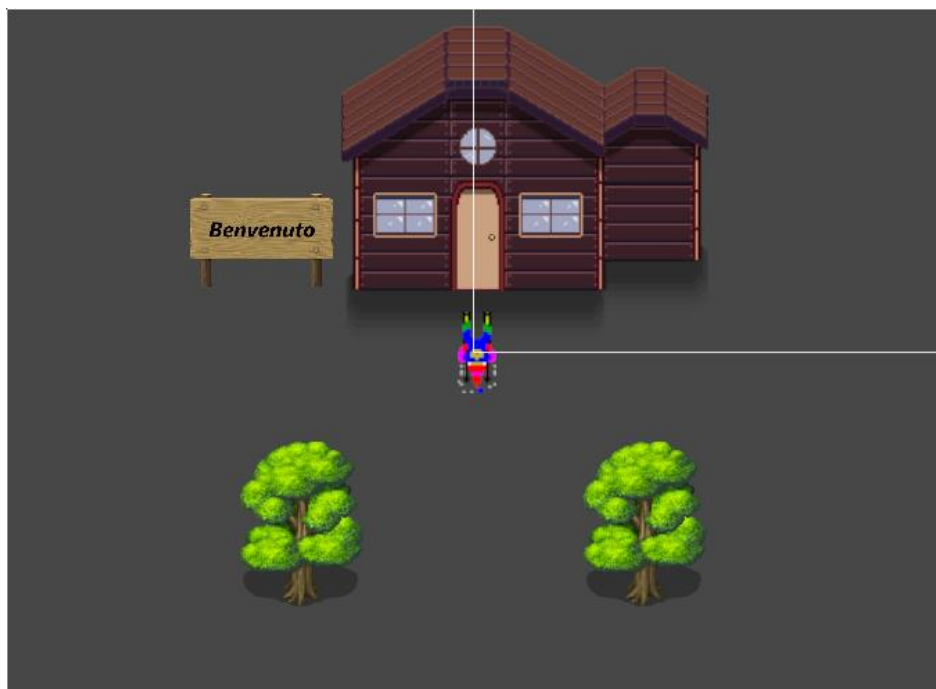


Figura 16 Parte iniziale del gioco

<sup>3</sup> <https://www.cleanpng.com/png-rpg-maker-mv-tree-tile-based-video-game-rpg-maker-639273/>

<sup>4</sup> <https://opengameart.org/content/wooden-house-with-animated-door-pixel-art>

<sup>5</sup> <https://toppng.com/photo/173082/sign-post-signage-wood-wooden-nails-brown-wooden-sign-transparent-background>

## 4.7 Generazione Mappa

A questo punto si è pronti per creare l'algoritmo per la generazione della mappa. La mappa dovrà contenere e generare i vari ostacoli in modo randomico e dovrà essere infinita.

### 4.7.1 Script LevelGenerator

Per la generazione della mappa si deve creare uno script, LevelGenerator, che conterrà tutto il codice necessario e che deve essere aggiunto ad un Empty GameObject chiamato LevelGenerator. In seguito nell'Inspector di questo GameObject si dovranno trascinare, nelle variabili dedicate, i vari ostacoli e altri GameObject necessari. Questo script permette di generare delle file di ostacoli<sup>6</sup> sotto al player prima che la camera passi da quel determinato punto.

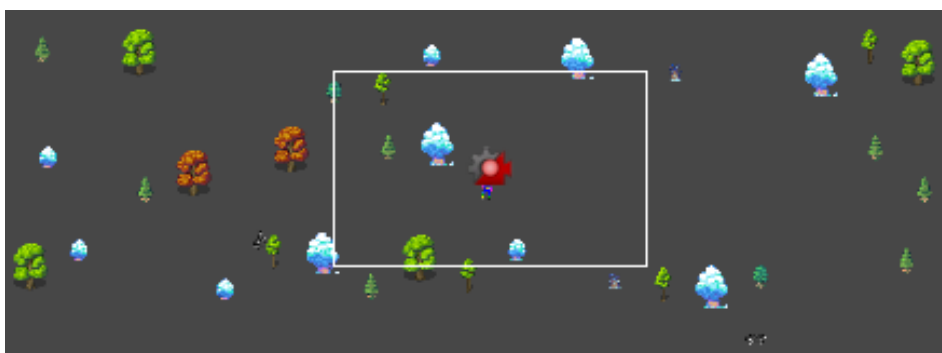


Figura 17 Generazione della mappa

In questo modo l'utente non vedrà mai la mappa generarsi mentre passa perché la mappa si genererà fuori dal campo visivo della camera.

Nel caso in cui l'utente facesse andare il personaggio in diagonale la generazione si sposterebbe insieme ad esso, questo accade perché la generazione parte dalla posizione orizzontale del player.

#### 4.7.1.1 Generazione ostacoli statici

Per generare gli ostacoli dinamici ho creato un algoritmo che, utilizzando le posizioni della camera, crea delle righe di alberi fuori dal campo visivo di essa. Questi alberi vengono generati casualmente su 40 posizioni possibili e vengono spostati di pochi pixel in su o in giù così da simulare la più totale casualità di generazione.

Gli ostacoli hanno comunque una distanza minima di generazione in tutti e quattro i loro lati, così da evitare sovrapposizioni tra di essi.

#### 4.7.1.2 Generazione ostacoli dinamici

Gli ostacoli dinamici, oltre a venir generati in una posizione inferiore rispetto al player, vengono generati più lateralmente.

L'algoritmo per il movimento di questi ostacoli gestisce il fatto della generazione a destra o sinistra. Se verranno generati a destra dello schermo correranno verso sinistra, mentre se verranno generati a sinistra correranno verso destra. In questo modo andranno sempre ad ostacolare il player.

<sup>6</sup> <https://www.pinterest.ch/pin/312507661637980697/>



#### 4.7.2 Lupo

Per il lupo si deve scaricare uno sprite multiplo<sup>7</sup> con i suoi diversi movimenti per poter creare delle animazioni.

Si devono creare tre animazioni, una per la corsa, una per la morte in caso di scontro e l'ultima per tenerlo bloccato una volta deceduto.

La prima contiene 5 immagini del lupo che, messe in sequenza, formano la corsa. La seconda, quella della morte, contiene 7 immagini per simulare la morte del lupo e, infine, l'ultima contiene unicamente l'immagine del lupo steso a terra per tenerlo fermo.

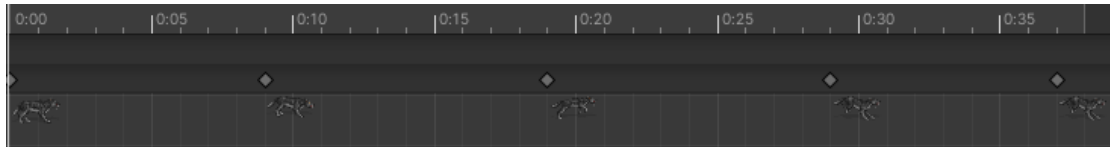


Figura 18 Esempio di animazione del lupo

Dopo aver creato queste tre animazioni bisogna aprire l'animator del lupo dove si devono collegare le varie animazioni seguendo delle condizioni.

Di default il lupo deve correre, nel momento in cui si scontra deve morire e infine deve stare fermo in quella posizione.

Quindi si deve creare un parametro che controlli se il lupo si è scontrato o no, al cambiamento di questo parametro il lupo dovrà cambiare animazione.

Per la gestione di questo valore si deve creare lo script Lupo, che va aggiunto al prefab base del lupo, dove, in caso di collisione con un ostacolo, verrà segnalato il cambiamento del valore di questo parametro.

Infine, dopo aver svolto solamente una volta l'animazione della morte, dovrà rimanere bloccato.

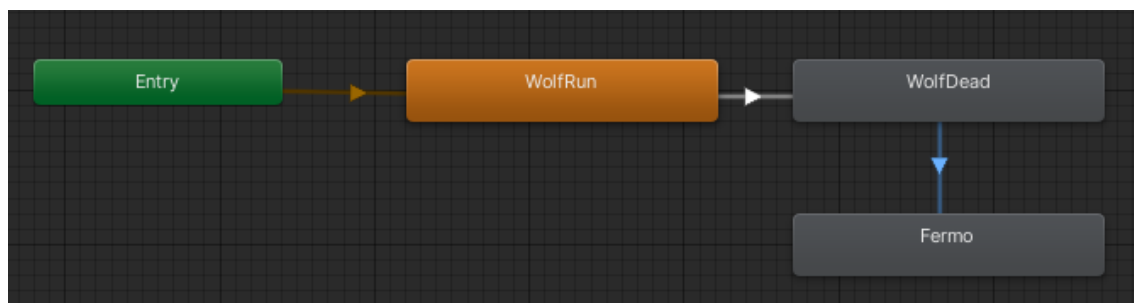


Figura 19 Animator del lupo

Queste animazioni danno un po' più di vita al gioco, dove il lupo simula una vera corsa e una vera morte, non si limita a fluttuare da destra a sinistra o viceversa.

<sup>7</sup> <https://www.pngegg.com/en/png-eyhsg>

#### 4.7.3 Yeti

Lo Yeti è la creatura che, dopo una determinata distanza percorsa, viene generato e cerca di uccidere lo sciatore.

Prima di tutto si deve scaricare lo sprite dello Yeti<sup>8</sup> con tutte le animazioni già incorporate.

Poi bisogna inserire all'interno dello script LevelGenerator il codice che permette la generazione dello Yeti dopo una certa distanza percorsa. Per farlo basta aggiungere una condizione che controlli se è il momento di generare lo Yeti o meno, nel caso in cui si debba generare verrà creato un po' sopra al player e inizierà istintivamente a seguirlo per poterlo uccidere.

Per fare in modo che lo Yeti inseguia il player bisogna creare un nuovo script per gestire i suoi movimenti.

#### 4.7.4 Script Yeti

In questo script c'è un algoritmo che permette allo Yeti di conoscere le coordinate del player e di muoversi verso di esse ad una velocità elevata.

Inoltre è compresa anche una condizione che fa sempre puntare lo Yeti verso il player, sia che si trovi alla sua sinistra, sia che si trovi alla sua destra.

Infine, per non rischiare di bloccare lo Yeti, ad ogni collisione l'ostacolo con cui si scontra verrà distrutto, compreso il player.

Questo script deve essere aggiunto al prefab dello Yeti.

Anche in questo caso si devono aggiungere i componenti per le collisioni come in tutti gli altri ostacoli.

Infine lo per lo Yeti, avendo le animazioni già incorporate, si deve soltanto andare nell'Animator e impostare come animazione di default la sua corsa.

#### 4.7.5 GestoreOstacoli

Per ultima cosa bisogna creare uno script che gestisca gli ostacoli, dinamici e non.

All'interno di questo script si deve inserire il codice che faccia diventare gli ostacoli come invisibili durante il salto del player e che li elimini quando sono stati superati.

Grazie a questo script gli ostacoli superati dal player verranno automaticamente distrutti per non riempire la memoria e, nel caso in cui venga premuto il tasto sinistro del mouse, verrà impostata la proprietà IsTrigger a true per tutta la durata del salto del player, così da permettere la simulazione di un salto sopra gli alberi.

Sono gli alberi a diventare come se fossero invisibili e non il player perché nel caso in cui ci fosse anche lo Yeti esso può mangiare comunque, anche se si trova in volo, lo sciatore.

Questo script deve essere aggiunto a tutti i prefab di ostacoli presenti.

Svolgendo tutti questi passaggi si avrà una mappa generata casualmente, infinita e con tutti gli ostacoli statici e dinamici necessari.

<sup>8</sup> <https://assetstore.unity.com/packages/2d/characters/enemycow-227480>

## 4.8 Dati di gioco

Durante la partita i dati di gioco, come metri percorsi, tempo di gioco e punteggio, devono essere memorizzati e mostrati a schermo.

Quindi la prima cosa da fare è quella di creare un Canvas con all'interno tre Text GameObject. Questi text devono essere posizionati e ancorati in alto a sinistra dello schermo, così da non infastidire l'utente durante la partita.

In seguito bisogna creare due script, Dati (che deve essere associato al text per i metri percorsi e il punteggio) e TimeController (che deve essere associato al text per il tempo di gioco).

### 4.8.1 Script Dati

Il primo script da creare è quello per la gestione dei dati generali.

Questo script permette di far visualizzare a schermo i metri percorsi e il punteggio, quindi si deve creare un metodo per i metri percorsi e uno per i punteggi.

In questo progetto lo spostamento di 1y verso il basso corrisponde a 1m di distanza, quindi basterà prendere la posizione del player e scriverla nel text dedicato. Questo metodo deve essere richiamato nell'Update() così da aggiornare sempre la distanza percorsa.

Il metodo del punteggio viene utilizzato dallo script PlayerMovement dove, ad ogni acrobazia andata a buon fine, viene aggiunto un punto in più e viene mostrato a schermo il punteggio totale.

### 4.8.2 Script TimerController

Questo script serve per la gestione del tempo di gioco.

All'interno della Coroutine principale viene salvato il tempo in millisecondi dall'inizio della partita e viene convertito in secondi e formattato in "hh:mm:ss". Questa Coroutine deve funzionare fino a quando il gioco non si conclude.

Infine i dati si mostreranno in questo modo in alto a sinistra dello schermo:

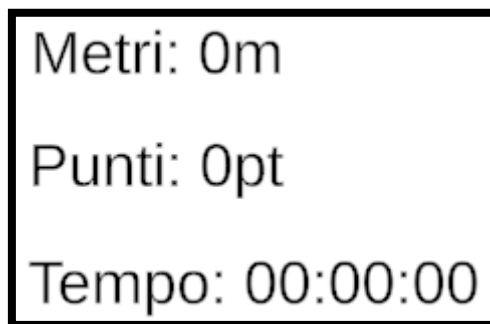


Figura 20 Visualizzazione dei dati

## 5 Test

### 5.1 Protocollo di test

<b>Test Case:</b>	TC-001	<b>Nome:</b>	Controllare la risoluzione della finestra di gioco
<b>Riferimento:</b>	REQ-01		
<b>Descrizione:</b>	Controllare che il gioco si adatti sempre al full screen del monitor		
<b>Prerequisiti:</b>			
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Avviare il gioco</li> <li>2. Cambiare risoluzione e controllare che si adatti sempre</li> </ol>		
<b>Risultati attesi:</b>	Il gioco si adatta ad ogni risoluzione in tempo reale		

<b>Test Case:</b>	TC-002	<b>Nome:</b>	Movimenti del personaggio
<b>Riferimento:</b>	REQ-02		
<b>Descrizione:</b>	Controllare se i movimenti del personaggio funzionano correttamente		
<b>Prerequisiti:</b>	Avvio del gioco		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Usare il mouse</li> <li>2. Posizionare il mouse sotto il personaggio e verificare che vada in avanti</li> <li>3. Spostare il mouse verso sinistra e verificare che il personaggio giri e che in posizione orizzontale si fermi</li> <li>4. Spostare il mouse verso destra e verificare che il personaggio giri e che in posizione orizzontale si fermi</li> </ol>		
<b>Risultati attesi:</b>	Il personaggio si muove verso la direzione del mouse e curva in modo più definito rispetto al gioco originale		

<b>Test Case:</b>	TC-003	<b>Nome:</b>	Movimenti del personaggio (in aria)
<b>Riferimento:</b>	REQ-02		
<b>Descrizione:</b>	Controllare se i movimenti del personaggio quando si trova in aria funzionano correttamente		
<b>Prerequisiti:</b>	Aver verificato i movimenti base		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Posizionare il mouse sotto il personaggio</li> <li>2. Cliccare una volta il tasto sinistro</li> <li>3. Quando il personaggio si trova in volo tenere premuto il tasto destro</li> </ol>		
<b>Risultati attesi:</b>	Il personaggio salta e prova a fare un mortale, se ci riesce continua a sciare mentre se non ci riesce si schianta al suolo, si ferma e poi si rialza		

<b>Test Case:</b>	TC-003	<b>Nome:</b>	Movimenti del personaggio (in aria)
<b>Riferimento:</b>	REQ-03		
<b>Descrizione:</b>	Controllare che il personaggio non possa cambiare traiettoria in volo		
<b>Prerequisiti:</b>	Aver verificato i movimenti base		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Posizionare il mouse sotto il personaggio</li> <li>2. Cliccare una volta il tasto sinistro</li> <li>3. Mentre si trova in volo muovere il mouse a destra e a sinistra per far curvare il personaggio</li> </ol>		
<b>Risultati attesi:</b>	Il personaggio mentre si trova in volo non deve poter cambiare traiettoria		

<b>Test Case:</b>	TC-004	<b>Nome:</b>	La mappa si genera correttamente
<b>Riferimento:</b>	REQ-03		
<b>Descrizione:</b>	Controllare se la generazione della mappa avviene prima del passaggio effettivo del personaggio		
<b>Prerequisiti:</b>	Aver verificato i movimenti base		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Posizionare il mouse sotto il personaggio</li> <li>2. Andare avanti e verificare che si generi una mappa con tutti gli ostacoli</li> <li>3. Andare a sinistra e verificare che si generi una mappa con tutti gli ostacoli</li> <li>4. Andare a destra e verificare che si generi una mappa con tutti gli ostacoli</li> </ol>		
<b>Risultati attesi:</b>	Man mano che il personaggio si sposta la mappa genera gli ostacoli automaticamente		

<b>Test Case:</b>	TC-005	<b>Nome:</b>	Collisione con gli ostacoli
<b>Riferimento:</b>	REQ-03		
<b>Descrizione:</b>	Controllare se esiste una collisione con i vari ostacoli		
<b>Prerequisiti:</b>	Aver verificato i movimenti base Aver verificato la generazione della mappa		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Posizionare il mouse sotto il personaggio</li> <li>2. Andare a sbattere contro ad ogni ostacolo e provare il trampolino</li> </ol>		
<b>Risultati attesi:</b>	Il personaggio deve fermarsi alla collisione con tutti gli ostacoli e deve saltare più in alto guadagnando velocità se salta su un trampolino.		

<b>Test Case:</b>	TC-006	<b>Nome:</b>	I dati vengono visualizzati a schermo
<b>Riferimento:</b>	REQ-04		
<b>Descrizione:</b>	Controllare se durante la partita i dati vengono visualizzati a schermo		
<b>Prerequisiti:</b>	Aver verificato i movimenti base e quelli aerei Aver verificato la generazione della mappa		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Posizionare il mouse sotto il personaggio</li> <li>2. Muoversi e controllare l'aumento dei metri</li> <li>3. Controllare l'aumento del tempo di gioco</li> <li>4. Fare un'acrobazia e controllare l'assegnazione del punteggio</li> </ol>		
<b>Risultati attesi:</b>	I metri percorsi aumentano ogni volta che il personaggio si muove, il tempo aumenta costantemente fino alla fine del gioco e dopo un'acrobazia andata a buon fine vengono assegnati 5 punti.		


<b>Test Case:</b>	TC-007	<b>Nome:</b>	Controllare punteggio
<b>Riferimento:</b>	REQ-04		
<b>Descrizione:</b>	Controllare se quando si fa un'acrobazia vengono assegnati i punti in modo corretto		
<b>Prerequisiti:</b>	Aver verificato i movimenti base e quelli aerei Aver verificato che i dati vengano visualizzati a schermo		
<b>Procedura:</b>	<ol style="list-style-type: none"> <li>1. Posizionare il mouse sotto il personaggio</li> <li>2. Saltare su un trampolino</li> <li>3. Fare un'acrobazia</li> </ol>		
<b>Risultati attesi:</b>	Ogni volta che viene effettuata un'acrobazia vengono aggiunti 10 punti		

<b>Test Case:</b>	TC-008	<b>Nome:</b>	Generazione dello Yeti
<b>Riferimento:</b>	REQ-05		
<b>Descrizione:</b>	Controllare se dopo 2500m percorsi compare uno yeti		
<b>Prerequisiti:</b>	Aver verificato i movimenti base e quelli aerei Aver verificato la generazione della mappa Aver verificato la visualizzazione dei dati della partita		
<b>Procedura:</b>	1. Posizionare il mouse sotto il personaggio 2. Percorrere 2500m		
<b>Risultati attesi:</b>	Dopo i 2500m deve comparire uno yeti		

<b>Test Case:</b>	TC-009	<b>Nome:</b>	Comportamento dello yeti
<b>Riferimento:</b>	REQ-05		
<b>Descrizione:</b>	Controllare che lo yeti voglia uccidere lo sciatore		
<b>Prerequisiti:</b>	Aver verificato i movimenti base e quelli aerei Aver verificato la generazione della mappa Aver verificato la visualizzazione dei dati della partita Aver verificato che lo yeti compare dopo 2500m		
<b>Procedura:</b>	1. Posizionare il mouse sotto il personaggio 2. Percorrere 2500m 3. Muoversi		
<b>Risultati attesi:</b>	Lo yeti prende lo sciatore e lo mangia		

<b>Test Case:</b>	TC-010	<b>Nome:</b>	Fine della partita
<b>Riferimento:</b>	REQ-05		
<b>Descrizione:</b>	Controllare che quando lo sciatore viene mangiato la partita finisce e si torna in cima		
<b>Prerequisiti:</b>	Aver verificato i movimenti base e quelli aerei Aver verificato la generazione della mappa Aver verificato la visualizzazione dei dati della partita Aver verificato che lo yeti compare dopo 2500m Aver verificato che lo yeti mangia lo sciatore		
<b>Procedura:</b>	1. Posizionare il mouse sotto il personaggio 2. Percorrere 2500m 3. Farsi mangiare dallo yeti		
<b>Risultati attesi:</b>	La partita deve finire e il personaggio torna in cima alla montagna		

## 5.2 Risultati test

ID	Risultato	Note	Data
TC-001	Passato	Il gioco è full screen ad ogni tipo di risoluzione.	23.12.2022
TC-002	Passato	Il personaggio si muove correttamente seguendo e puntando il mouse.	16.12.2022
TC-003	Passato	Il personaggio salta e può fare acrobazie.	16.12.2022
TC-004	Passato	Il personaggio in volo non si muove e tiene una direzione costante.	16.12.2022
TC-005	Passato	La mappa si genera completamente in modo casuale e all'infinito.	16.12.2022
TC-006	Passato	Il personaggio si blocca quando si scontra con gli altri ostacoli.	16.12.2022
TC-007	Passato	<p>I dati della partita vengono visualizzati in alto a sinistra dello schermo.</p>  <p>Figura 21 Dati visualizzati</p>	16.12.2022
TC-008	Passato	Il punteggio viene assegnato solo nel caso in cui lo sciatore atterra in modo corretto.	16.12.2022
TC-009	Passato	Lo Yeti viene generato dopo una certa distanza percorsa. (impostata nello script).	16.12.2022
TC-010	Passato	Lo Yeti appena viene generato parte e insegue il player fino a quando non lo uccide.	16.12.2022
TC-011	Passato	La partita finisce se il giocatore viene mangiato, si ricarica il menu principale.	16.12.2022



### **5.3 Mancanze/limitazioni conosciute**

#### **5.3.1 Movimento del personaggio**

Non è implementata l'accelerazione del personaggio, quindi scia sempre alla stessa velocità costante. Per mancanza di tempistiche e per non rischiare di rovinare completamente il movimento del personaggio questa funzionalità non è stata implementata. Inoltre il salto ha una durata fissa di due secondi e il salto in diagonale non è concesso.

#### **5.3.2 Ostacoli**

Sono presenti pochi ostacoli, solamente alberi e lupi, questo perché è difficile trovare dei modelli gratuiti da scaricare adatti a questo progetto. Inoltre l'aggiunta di altri ostacoli era tra le ultime cose della lista, in termini di necessità, quindi è stata tralasciata per provvedere ad un miglior risultato complessivo.

#### **5.3.3 Modello sciatore**

Il modello utilizzato è lo stesso di quello del gioco originale, questo perché non esistono dei modelli di sciatori in 2D adatti a questo progetto. Per implementare un buon modello sarebbe stato perso troppo tempo visto che avrei dovuto creare il modello da 0.

#### **5.3.4 Lupi**

I lupi corrono orizzontalmente e quindi c'è una grande probabilità che la maggior parte di essi si scontri con gli ostacoli presenti. Questo perché gli alberi generati sono piuttosto vicini e lo spazio per passare non è molto.

**6 Consuntivo**

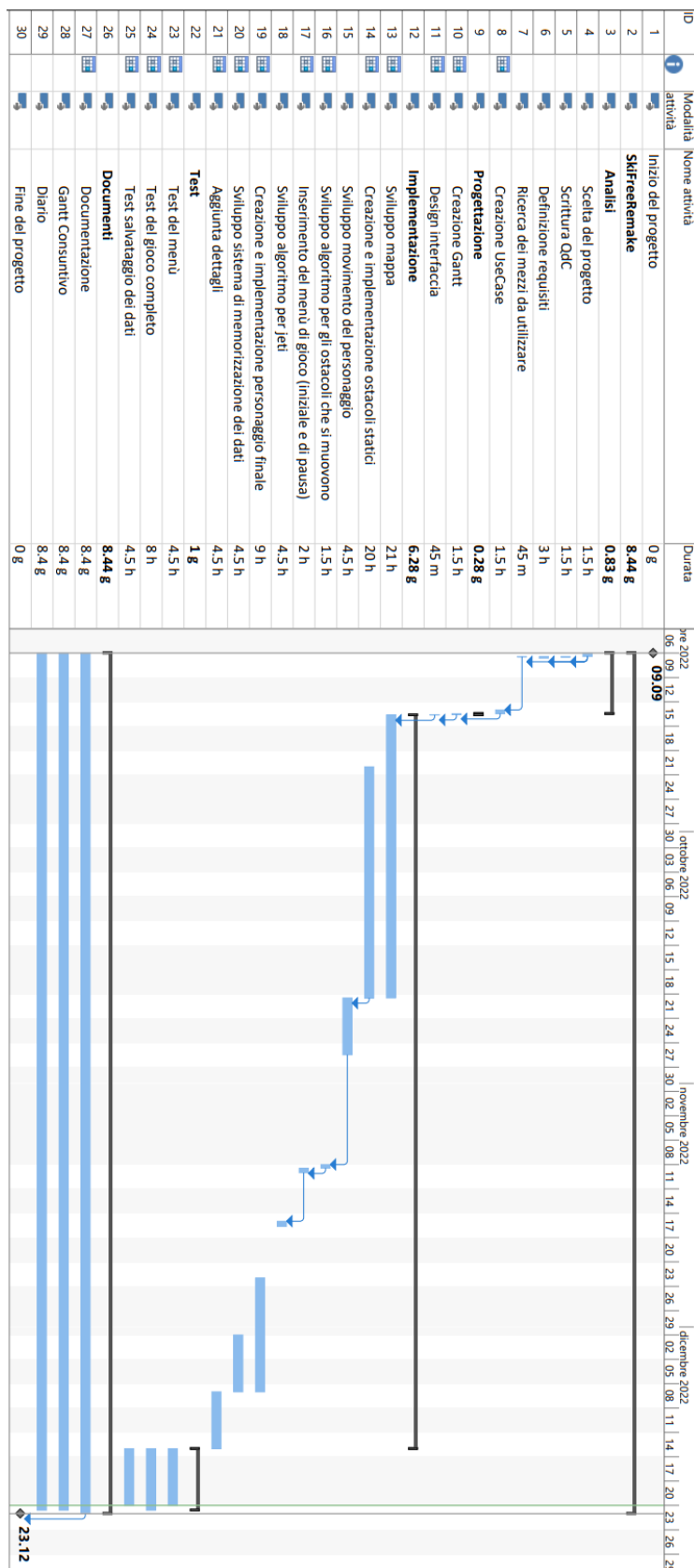


Figura 22 Gantt consuntivo

Come si può vedere dall'immagine il Gantt Consuntivo è cambiato molto da quello iniziale, ma nonostante questo sono comunque riuscito a finire il progetto nei tempi prestabiliti.

La parte di implementazione è completamente differente, infatti il tempo per la creazione della mappa e l'implementazione degli ostacoli è aumentato di molto. Invece la parte successiva, quella dello sviluppo degli algoritmi per gli ostacoli dinamici e la creazione dei menu, è durata relativamente poco vista l'inaspettata facilità di realizzazione.

Inoltre molte attività che erano state pianificate per essere eseguite in sequenza, durante l'implementazione, si sono potute realizzare in parallelo.

## **7 Conclusioni**

---

### **7.1 Sviluppi futuri**

Il prodotto finale ha ancora molte limitazioni e sicuramente delle buone migliorie potrebbero essere:

- Implementare l'accelerazione al personaggio (sia in volo sia a terra)
- Aggiungere nuovi ostacoli oltre ad alberi e lupi
- Implementare un nuovo modello dello sciatore con buone animazioni
- Migliorare le collisioni
- Fare in modo che si possa salvare la miglior partita giocata dall'utente
- Aggiungere dei suoni al gioco
- Implementare il cambio di risoluzione all'interno del gioco (in finestra e schermo intero)

### **7.2 Considerazioni personali**

Svolgendo questo progetto ho imparato a realizzare un prodotto seguendo una pianificazione, dei requisiti e delle tempistiche ben precise. Mi ha insegnato come una buona pianificazione iniziale decida le sorti dell'andamento del progetto e quanto sia importante avere dei requisiti ben specificati.

Ho inoltre imparato come gestire il mio tempo per cercare di sfruttarlo al meglio per produrre il miglior prodotto finale possibile, cosa non semplice visto i molteplici problemi riscontrati durante l'implementazione.

Con questo progetto ho anche imparato come creare un gioco 2D su Unity, cosa che da sempre mi interessava, e sono contento e soddisfatto del lavoro svolto.

Infine posso dire di aver lavorato con impegno e serietà a questo progetto e che questo mi ha permesso di ampliare notevolmente le mie conoscenze nell'ambito della programmazione.

## 8 Glossario

Termine	Descrizione
sprite	Immagini png che vengono implementate nei progetti Unity per poter avere dei modelli grafici 2D
prefab	Prefabbricati sono come dei GameObject, si possono applicare i componenti
GameObject	I GameObject sono degli oggetti di gioco che vengono creati e usati nel workspace di Unity. (per esempio dei cerchi, rettangoli, testi)
Inspector	L'inspetcor è la sezione contenente tutte i componenti e le relative proprietà di un GameObject.
Rigidbody2D	Rigidbody2D è un componente che da un corpo al GameObject a cui è associato.
BoxCollider2D	BoxCollider2D è un componente che permette di riconoscere e gestire le collisioni con gli altri GameObject presenti.
Coroutine	Coroutine è una funzione temporanea che può mettere in pausa l'esecuzione per poi riprendere dal frame successivo.

## 9 Bibliografia

### 9.1 Sitografia

- <https://wireframepro.mockflow.com/#/space/default>, 16-09-2022.
- <https://www.pinterest.ch/pin/312507661637980697/>, 16-09-2022.
- <https://gamedevacademy.org/procedural-2d-maps-unity-tutorial/>, *Procedural 2D maps*, 30-09-2022.
- <https://www.youtube.com/watch?v=TrbbkCrFD00>, *The Unity interpretation of a 2D Endless Terrain Generation using Perlin noise*, 30-09-2022.
- <https://www.youtube.com/watch?v=WP-Bm65Q-1Y>, *Procedural Landmass Generation*, 07-10-2022
- [https://www.youtube.com/watch?v=NtY\\_R0g8L8E](https://www.youtube.com/watch?v=NtY_R0g8L8E), *Endless Runner Level Generator in Unity*, 07-10-2022.
- <https://www.youtube.com/watch?v=Ehk9fKBwS3Y>, *Smooth Mouse Follow in Unity*, 14-10-2022.
- <https://youtube.com/shorts/2KWHMSCxibA?feature=share>, *Unity 2D Camera Follow Player*, 14-10-2022.
- <https://www.youtube.com/watch?v=D4EOgZyNk-k>, *Random Map Generator*, 21-10-2022.
- <https://github.com/aidan-waite/OpenSkiFree/blob/main/Assets/Art/skifreesprites.png>, *OpenSkiFree*, 21-10-2022.
- <https://www.youtube.com/watch?v=Bc9ImHjQLZc>, *Collisions and Triggers*, 28-10-2022.
- <https://assetstore.unity.com/packages/2d/characters/enemycow-227480>, 28-10-2022.
- <https://www.pngegg.com/en/png-eyhsg>, 11-11-2022.
- [https://www.youtube.com/watch?v=zc8ac\\_qUXQY](https://www.youtube.com/watch?v=zc8ac_qUXQY), *Start Menu in Unity*, 11-11-2022.
- <https://www.cleanpng.com/png-rpg-maker-mv-tree-tile-based-video-game-rpg-maker-639273/>, 11-11-2022.
- <https://opengameart.org/content/wooden-house-with-animated-door-pixel-art>, 11-11-2022
- <https://toppng.com/photo/173082/sign-post-signage-wood-wooden-nails-brown-wooden-sign-transparent-background>, 11-11-2022.
- <https://www.youtube.com/watch?v=JivuXdriHK0>, *Pause Menu in Unity*, 11-11-2022.
- <https://www.youtube.com/watch?v=2SXa10ILJms>, *Unity simple 2D Enemy AI Follow Tutorial*, 18-11-2022.
- <https://gamedevelopertips.com/unity-collision-detection-2d/>, *Unity Collision Detection*, 18-11-2022.
- <https://www.pngegg.com/en/png-eyhsg>, *Sprite Gray wolf*, 18-11-2022.
- [https://www.youtube.com/watch?v=2DHy\\_l4Ffe0](https://www.youtube.com/watch?v=2DHy_l4Ffe0), *Player Following The Mouse*, 25-11-2022.
- <https://www.youtube.com/watch?v=qc7J0iei3BU&t>, *Make an In-Game Timer*, 09-12-2022.
- <https://www.vecteezy.com/vector-art/4371984-winter-landscape-with-mountains-pines-forest-snow-falling-background-for-your-arts>, 09-12-2022

## 9.2 Indice delle figure

Figura 1 UML Use Case del progetto .....	9
Figura 2 Diagramma di Gantt Iniziale .....	10
Figura 3 Interfaccia menu principale .....	12
Figura 4 Schermata di pausa .....	13
Figura 5 Interfaccia della partita .....	14
Figura 6 Creazione progetto Unity 2D .....	15
Figura 7 Cartelle necessarie.....	15
Figura 8 Build Settings delle scene .....	16
Figura 9 Scena del menu principale .....	16
Figura 10 OnClick() dei pulsanti .....	17
Figura 11 Menu di pausa .....	18
Figura 12 Separazione sprites.....	19
Figura 13 Camera segue il player .....	20
Figura 14 Rotazione di partenza del personaggio .....	21
Figura 15 Animator del personaggio .....	22
Figura 16 Parte iniziale del gioco .....	23
Figura 17 Generazione della mappa .....	24
Figura 18 Esempio di animazione del lupo.....	25
Figura 19 Animator del lupo.....	25
Figura 20 Visualizzazione dei dati .....	27
Figura 21 Dati visualizzati.....	32
Figura 22 Gantt consuntivo .....	34

## 10 Allegati

---

Elenco degli allegati, esempio:

- Use Case
- Gantt iniziale e consuntivo
- Interfacce progettate
- Diari giornalieri
- Quaderno dei compiti