Colorful Songs

1 Introduzione 3

1.1 Informazioni sul progetto 3

1.2 Abstract 3

1.3 Scopo 3

2 Analisi 4

2.1 Analisi del dominio 4

2.2 Analisi e specifica dei requisiti 4

2.2.1 Spiegazione elementi tabella dei requisiti: 7

2.3 Use case 7

2.4 Pianificazione 7

2.5 Analisi dei mezzi 8

2.5.1 Software 8

2.5.2 Hardware 8

3 Progettazione 8

3.1 Design dell’architettura del sistema 8

3.2 Design dei dati e database 9

3.3 Design delle interfacce 10

3.3.1 Design Login 10

3.3.2 Design Register 11

3.3.3 Design Leaderboard 11

3.4 Design degli Enigmi e ostacoli 12

3.4.1 Percorso con pedane finte 12

3.4.2 Percorso con pedana invisibile e soluzione sul muro 1

3.4.3 Portali finti 1

3.4.4 Percorso con cannoni 1

3.4.5 Percorso con parkour 1

3.5 Generazione dei Dungeon 2

3.6 Design procedurale 2

4 Implementazione 3

5 Test 3

5.1 Protocollo di test 3

5.2 Risultati test 4

5.3 Mancanze/limitazioni conosciute 4

6 Consuntivo 4

7 Conclusioni 4

7.1 Sviluppi futuri 4

7.2 Considerazioni personali 4

8 Glossario 4

9 Indice delle figure 4

10 Bibliografia 5

10.1 Bibliografia per articoli di riviste: 5

10.2 Bibliografia per libri 5

10.3 Sitografia 5

11 Allegati 5

# Introduzione

## Informazioni sul progetto

* Allievi: Lorenzo Berther, Sebastiano De Bertoldi, Simone Demarchi, Kamil Siddiqui
* Docente responsabile: Michel Palucci
* Scuola: Centro Professionale Tecnico Trevano, sezione informatica
* Data di inizio: 29.01.2025
* Data di Termine: 28.05.2025

## Abstract

Today's big video game companies continue to offer gamers the same brands year after year. The only thing that changes is the price, which is constantly increased with the release of a new title. For this reason, indie studios are attracting a lot of attention these days.

Colorful Songs is an indie game with an 80s theme, created by four students, that tries to offer different gameplay by integrating a variety of mechanics into one title: parkour, combat. speedrunning and obstacles are at the heart of Colorful Songs.

The game tells the story of a music composer who has lost his melody, now fragmented into various pieces. His goal? Recover all the fragments and reassemble it, trying to make it even more beautiful than before!

With this game, you want to make the gamer relive the graphic atmosphere of the 1980s, leaning heavily on the nostalgia factor. The plot is enhanced by engaging music, with the soundtrack playing a central role in the story.

## Scopo

Lo scopo didattico di questo progetto è sviluppare le competenze nella collaborazione di gruppo utilizzando le risorse scolastiche, acquisire familiarità con l’utilizzo di Unity e comprenderne le principali funzionalità.

Lo scopo operativo, invece, è la creazione di un gioco per PC del genere Roguelike, in cui l’obiettivo principale è proseguire il più possibile all’interno dei dungeon. Il gioco è ambientato negli anni '80, sia per aderire al tema di un contest sia per garantire la compatibilità con computer meno recenti. Questa scelta ha facilitato la gestione del tempo, grazie alla grafica semplice e minimale.

# Analisi

## Analisi del dominio

Il videogioco è destinato a qualunque fascia d’età, richiede solamente di registrarsi creando un account.  
Sarà giocabile da PC con tastiera o controller. L’obbiettivo principale è quello di creare un puzzle-game in Unity…

## Analisi e specifica dei requisiti

|  |  |
| --- | --- |
| **ID: REQ-001** | |
| **Nome** | Gestione utente |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Creare Database ‘ColorfulSongs’ |
| **002** | Creare le tabelle ‘User’ e ‘Friend’ |

|  |  |
| --- | --- |
| **ID: REQ-002** | |
| **Nome** | Effettuare un login |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Gestione Utente) |
| **Sotto requisiti** | |
| **001** | Deve spuntare una tastiera virtuale (per controller) |

|  |  |
| --- | --- |
| **ID: REQ-003** | |
| **Nome** | Effettuare una registrazione |
| **Priorità ID: REQ-003** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Gestione Utente) |
| **Sotto requisiti** | |
| **001** | Controlli per non creare utenti identici a quelli già esistenti |
| **002** | Deve spuntare una tastiera virtuale (per controller) |

|  |  |
| --- | --- |
| **ID: REQ-004** | |
| **Nome** | Leaderboard web |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Gestione utente) |
| **Sotto requisiti** | |
| **001** | Creare la tabella ‘Leaderboard’ |
| **002** | Creare una pagina web che prenda gli highscore degli utenti |
| **003** | Nel gioco deve esserci un collegamento alla pagina |

|  |  |
| --- | --- |
| **ID: REQ-005** | |
| **Nome** | Login nella Leaderboard Web |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Gestione utente) |
| **Sotto requisiti** | |
| **001** | Creare una pagina di login |

|  |  |
| --- | --- |
| **ID: REQ-006** | |
| **Nome** | Creare gli enigmi |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Creare il “[Percorso con pedane finte](#_Percorso_con_pedane)” |
| **002** | Creare il “[Percorso con pedana invisibile e soluzione sul muro](#_Percorso_con_pedana)” |
| **003** | Creare i “[Portali finti](#_Portali_finti)” |
| **004** | Creare il “[Percorso con cannoni](#_Cannoni)” |
| **005** | Creare il “[Percorso con parkour](#_Parkour)” |

|  |  |
| --- | --- |
| **ID: REQ-007** | |
| **Nome** | Creazione del dungeon casuale |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Generare le stanze e gli ostacoli. |
| **002** | Creare id per il dungeon |
| **003** | Generare casualmente gli NPC. |

|  |  |
| --- | --- |
| **ID: REQ-008** | |
| **Nome** | Salvataggio highscore |
| **Priorità** | 1 |
| **Versione** | 1.0 |

|  |  |
| --- | --- |
| **ID: REQ-009** | |
| **Nome** | Tutorial testuale |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Creare una pagina con la descrizione dei comandi |

|  |  |
| --- | --- |
| **ID: REQ-010** | |
| **Nome** | Aggiungere amici |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | Dipende dal requisito REQ-001 (Creazione DB) |

|  |  |
| --- | --- |
| **ID: REQ-011** | |
| **Nome** | Giocare con tastiera e/o controller |
| **Priorità** | 1 |
| **Versione** | 1.0 |
| **Note** | In caso di controller, generare una tastiera virtuale, per soddisfare requisito REQ-002 e REQ-003 |

|  |  |
| --- | --- |
| **ID: REQ-012** | |
| **Nome** | Personalizzazione skin |
| **Priorità** | 2 |
| **Versione** | 1.0 |
| **Sotto requisiti** | |
| **001** | Creare uno shop in cui acquistare le skin |
| **002** | A fine del dungeon dare delle monete |

|  |  |
| --- | --- |
| **ID: REQ-013** | |
| **Nome** | Tutorial interattivo |
| **Priorità** | 3 |
| **Versione** | 1.0 |

### Spiegazione elementi tabella dei requisiti:

**ID**: identificativo univoco del requisito

**Nome**: breve descrizione del requisito

**Priorità**: indica l’importanza di un requisito nell’insieme del progetto, definita assieme al committente. Ad esempio, poter disporre di report con colonne di colori diversi ha priorità minore rispetto al fatto di avere un database con gli elementi al suo interno. Solitamente si definiscono al massimo di 2-3 livelli di priorità.

**Versione**: indica la versione del requisito. Ogni modifica del requisito avrà una versione aggiornata.

Sulla documentazione apparirà solamente l’ultima versione, mentre le vecchie dovranno essere inserite nei diari.

**Note**: eventuali osservazioni importanti o riferimenti ad altri requisiti.

**Sotto requisiti**: elementi che compongono il requisito.

## Use case

I casi d’uso rappresentano l’interazione tra i vari attori e le funzionalità del prodotto.

## Pianificazione

Prima di stabilire una pianificazione bisogna avere almeno una vaga idea del modello di sviluppo che si intende adottare. In questa sezione bisognerà inserire il modello concettuale di sviluppo che si seguirà durante il progetto. Gli elementi di riferimento per una buona pianificazione derivano da una scomposizione top-down della problematica del progetto.

La pianificazione può essere rappresentata mediante un diagramma di Gantt:

|  |
| --- |
| Figura 1: Esempio di diagramma di Gantt. |

Se si usano altri metodi di pianificazione (p.es. scrum), dovranno apparire in questo capitolo.

## Analisi dei mezzi

### Software

* Unity 6
* MS-Office
* MS-Project
* GitHub (Desktop)
* PHP Storm 2024.1.1
* MySQL Workbench 8.0
* Bootstarp 5.2.3
* Figma

### Hardware

PC forniti dalla scuola e Server scolastico dove mettere il progetto e il Database

# Progettazione

Questo capitolo descrive esaustivamente come deve essere realizzato il prodotto fin nei suoi dettagli. Una buona progettazione permette all’esecutore di evitare fraintendimenti e imprecisioni nell’implementazione del prodotto.

## Design dell’architettura del sistema

## Design dei dati e database

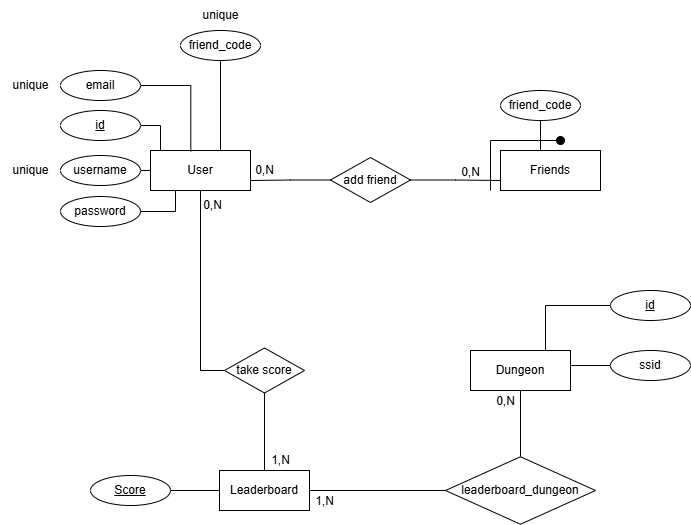


Figura 2: Schema E-R



Figura 3: Schema logico

## Design delle interfacce

### Design Login



Figura 4: Design interfaccia di Login

### Design Register



Figura 5: Design interfaccia di Registrazione

### Design Leaderboard



Figura 6 Design interfaccia leaderboard

#### Design della mappa



Figura 7: Design di una stanza

## Design degli ostacoli

### Percorso con pedane finte

In questo percorso il giocatore dovrà percorrere un percorso lungo 4 pedane, dove alcune pedane saranno finte. Se l’utente ci passerà sopra, verrà riportato all’inizio della stanza, perdendo eventuali oggetti che possedeva, come la chiave per la porta d’uscita, e la pedana sparirà, lasciando un buco.

Questo percorso occuperebbe uno spazio di colore rosso sulla mappa.



Figura 8: Design del percorso con pedane finte



Figura 9: Design della pedana (Come apparirebbe all'utente dopo qualche tentativo)

### Percorso con pedana invisibile e soluzione sul muro

Questo percorso è simile all’enigma “Percorso con pedane finte”, ma a sua differenza le pedane sono invisibili e il percorso da percorrere è disegnato sul muro, qua se si sbaglia e si passa su una pedana inesistente si perderà la partita.

Questo percorso occuperebbe uno spazio di colore rosso sulla mappa.

Ecco un esempio di come un’ipotetica pedana potrebbe essere e come potrebbe venir generata:

La parte rossa è bianca sono invisibili/trasparenti. Sul muro grigio vediamo la strada corretta da percorrere (bianca).



Figura 10: Design dell’enigma “Percorso con pedana invisibile e soluzione sul muro”



Figura 11: Design dell'enigma "Percorso con pedana invisibile e soluzione sul muro" visto dall'utente

### Portali finti

Nella stanza saranno presenti dei portali, solo uno di questi porterà il player dalla parte opposta del percorso. Se l’utente prende il portale sbagliato perderà la partita, per riconoscere il portale giusto bisognerà mettersi davanti il portale e ascoltare l’audio, che sarà diverso per quello giusto.

Se il player ha problemi d’udito, nelle impostazioni potrà attivare il fatto che questo enigma non gli appare.



Figura 12: Design enigma dei portali



Figura 13: Design dell'enigma dei portali visto dall'utente

### Percorso con cannoni

Questo percorso avrà sui muri dei cannoni che spareranno a intervalli regolari, alternandosi tra i cannoni in posizione pari con quelli in posizione dispari, se si verrà colpiti da cannoni si perderà la partita

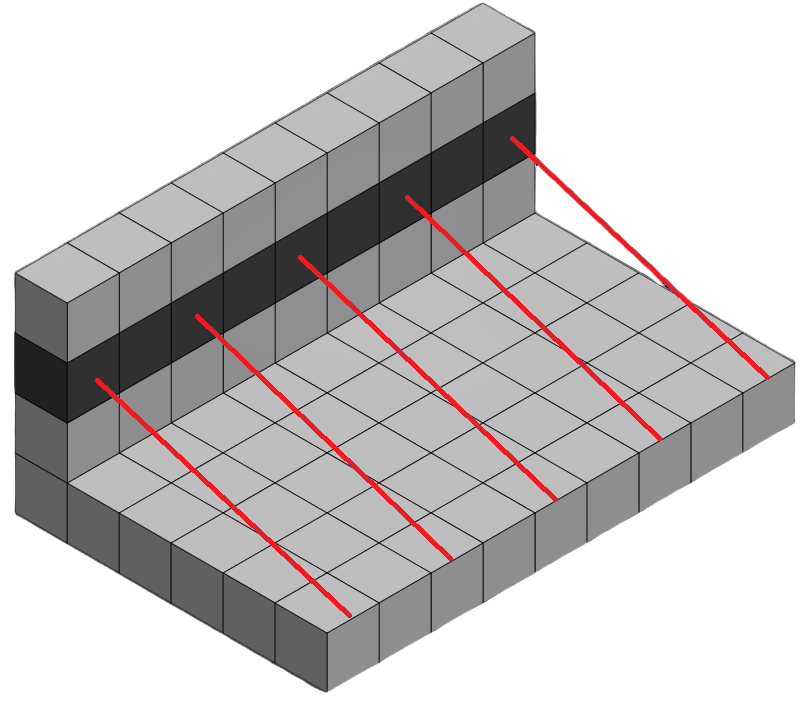


Figura 14: Cannoni in posizione pari sparano

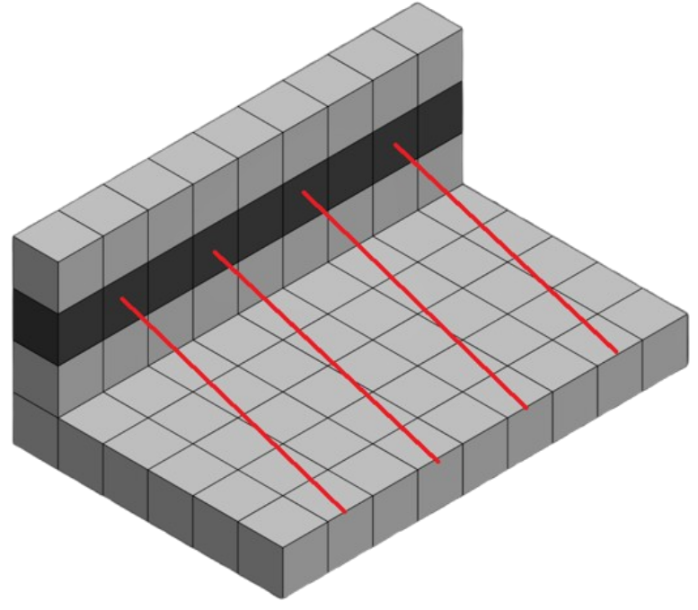


Figura 15: Cannoni in posizione dispari sparano

### 

### Percorso con parkour

Questo percorso sarà un semplice parkour, l’utente dovrà superarlo usando la meccanica del salto. Se cadrà perderà la partita.

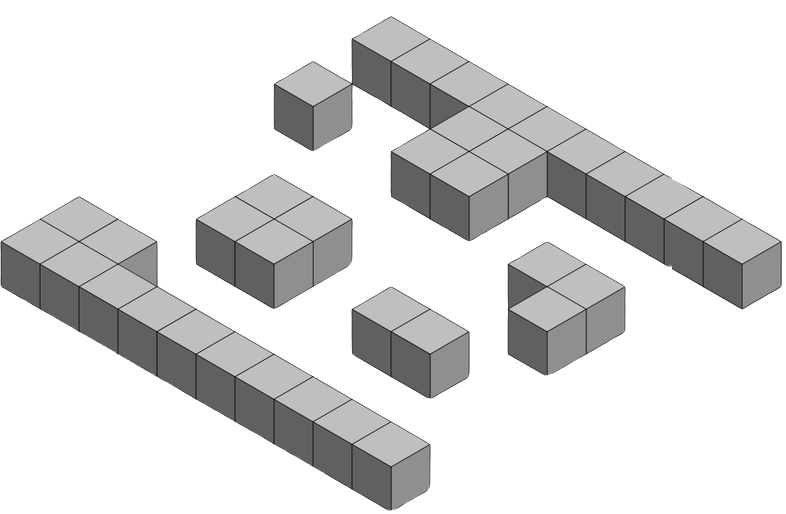


Figura 16: Percorso parkour

## Generazione dei Dungeon

I dungeon sono composti da cinque stanze, ognuna contenenti due enigmi.

Per generare le stanze useremo una stanza vuota come base e aggiungeremo due enigmi casualmente. Bisognerà quindi assegnare degli id alle stanze vuote e agli enigmi, così da poter usare un generatore di numeri per selezionare stanze ed enigmi. Questo renderà facile generare gli seed dei dungeon, che saranno composti dall’id della stanza e gli id egli enigmi, un esempio di seed di un dungeon potrebbe essere:  
“115-424-322-112-552”, quindi la stanza con id=1 aveva al suo interno gli enigmi con id=1 e id=5.

## Design procedurale

Descrive i concetti dettagliati dell’architettura/sviluppo utilizzando ad esempio:

* Diagrammi di flusso e Nassi.
* Tabelle.
* Classi e metodi.
* Tabelle di routing
* Diritti di accesso a condivisioni …

Questi documenti permetteranno di rappresentare i dettagli procedurali per la realizzazione del prodotto.

# Implementazione

## Classe TerrainGenerator (Generazione della stanza)

|  |
| --- |
| public class TerrainGenerator : MonoBehaviour  {  private int x;  private int z;  private int h;  private bool empty;  private int firstObstacle;  private int secondObstacle;  private int startingX = 0;  private int startingZ = 0;  [SerializeField] private int row;  [SerializeField] private int column;  [SerializeField] private int y;  [SerializeField] private int wallHeight;  [SerializeField] private int startObstacle1X;  [SerializeField] private int endObstacle1X;  private int startObstacle1Z;  private int endObstacle1Z;  [SerializeField] private int startObstacle1Y;  [SerializeField] private int startObstacle2X;  [SerializeField] private int endObstacle2X;  private int startObstacle2Z;  private int endObstacle2Z;  [SerializeField] private int startObstacle2Y;  [SerializeField] private GameObject floor;  [SerializeField] private GameObject wall;  [SerializeField] private GameObject parent;  [SerializeField] private GameObject finishPortal;  public GameObject[] obstacles; |

Questa prima parte di codice della classe TerrainGenerator inizializza tutte le variabili che verranno utilizzate.

Le variabili x,z e h sono variabili usate per cicli for per generare i blocchi della stanza.

La variabile bool empty serve per dire al ciclo for quando non generare alcun blocco e lasciare spazio vuoto per gli ostacoli.

firstEnigma e secondEnigma sono due variabili che conterranno la posizione dell’ostacolo nell’indice enigmas[].

startingX e startingZ sono le coordinate dalla quale si inizierà a generare il terreno della stanza.

row, column e y sono le variabili che identificano la grandezza e l’altezza della stanza.

wallHeight è l’altezza dei muri laterali.

startObstacle1 e endObstacle1 sono tutte le variabili con le coordinate per la generazione del primo ostacolo.

startObstacle2 e endObstacle2 sono invece le variabili per il secondo ostacolo.

I GameObject floor e wall contengono i prefab, blocchetti pre-fabbricati, da usare per generare la stanza.

Il GameObject parent sarà il parente di tutti i blocchi usati per generare la stanza.

Il GameObject finishPortal è un prefab che si inserisce a fine stanza e serve da portale per il prossimo livello.

L’array GameObject obstacles[] contiene gli oggetti con gli script per generare gli ostacoli.

|  |
| --- |
| void Start()  {  int l = obstacles.Length;  do  {  firstObstacle = Random.Range(0, l);  secondObstacle = Random.Range(0, l);  } while (firstObstacle == secondObstacle);  obstacles[firstObstacle].SetActive(true);  obstacles[secondObstacle].SetActive(true);  generateStraightTerrain();  var \_finishPortal = Instantiate(finishPortal,  new Vector3(row-3, y+1.5f, column-column/2),  Quaternion.identity);  \_finishPortal.name = "FinishPortal";  \_finishPortal.transform.SetParent(parent.transform);  } |

Nella funzione Start() inizializzo una variabile l uguale alla lunghezza dell’array obstacles.

In un ciclo do while genero due numeri casuali in un range 0 a l (non incluso) finchè non sono differenti.

Attivo i due ostacoli agli indici firstObstacle e secondObstacle.

Chiamo la funzione generateStraightTerrain() che genera la stanza dritta.

Creo l’oggetto \_finishPortal che verrà messo a fine stanza e servirà come portale per la stanza seguente.

|  |
| --- |
| public void generateStraightTerrain()  {  startObstacle1Z = 0;  endObstacle1Z = column;  startObstacle2Z = 0;  endObstacle2Z = column;  for (int r = startingX; r <= row; r++)  {  x = r;  if (  ((r < startObstacle1X || r >= endObstacle1X)) &&  ((r < startObstacle2X || r >= endObstacle2X))  )  {  empty = false;  }  else  {  empty = true;  }  for (int c = startingZ; c <= column; c++)  {  z = c;  if (!empty)  {  string name = "pavimento[" +  r.ToString() +  ";" + y.ToString() +  ";" + c.ToString() + "]";  var cube = Instantiate(floor,  new Vector3(x, y, z),  Quaternion.identity);  cube.name = name;  cube.transform.SetParent(parent.transform);  }  if (r == row || c == column)  {  for (int a = y; a <= wallHeight; a++)  {  string nameWall = "wall[" + r.ToString() +  ";" + a.ToString() + ";" +  c.ToString() + "]";  var wallObj = Instantiate(wall,  new Vector3(x, a, z),  Quaternion.identity);  wallObj.name = nameWall;  wallObj.transform.SetParent(parent.transform);  }  }  }  }  } |

Tramite due cicli for annidati, tratto la generazione del terreno come una tabella, e con un terzo for annidato gestisco la generazione dei muri.

Controllo se la r (variabile del ciclo for) non è nei range dei due ostacoli, se non lo è la variabile empty è uguale a true.

In un altro ciclo for, genero i blocchi del pavimento.

Al suo interno, con un if verifico se la r o la c sono uguali a row e column (i valori massimi), se si in un ciclo for genero i muri.

## Classe ParkourGenerator

|  |
| --- |
| public class ParkourGenerator : MonoBehaviour  {  private int startingX;  private int endingX;  private int startingZ;  private int endingZ;  private int startingY;  private CharacterController characterController;  [SerializeField] private GameObject parkour;  [SerializeField] private GameObject parent;  [SerializeField] private GameObject generator;  void Start()  {  startingX = generator.GetComponent<TerrainGenerator>().getStartX("ParkourGenerator");  startingZ = generator.GetComponent<TerrainGenerator>().getStartZ("ParkourGenerator");  startingY = generator.GetComponent<TerrainGenerator>().getStartY("ParkourGenerator");  endingX = generator.GetComponent<TerrainGenerator>().getEndX("ParkourGenerator");  endingZ = generator.GetComponent<TerrainGenerator>().getEndZ("ParkourGenerator");  int oldZ = -1;  int z = 0;  int x = 0;  bool first = true;  while(x < endingX)  {  if (first){  x = startingX + Random.Range(0,3);  first = false;  }    if (oldZ == -1)  {  z = Random.Range(startingZ, endingZ);  }  else  {  while(z>=endingZ || z < 0 || z==oldZ)  {  z = Random.Range(oldZ-2, oldZ + 3);  }  }  oldZ = z;    var cube = Instantiate(parkour,  new Vector3(x, startingY, z),  Quaternion.identity);  cube.name = "parkour[" + x + "; "+z+"]";  cube.transform.SetParent(parent.transform);  x += Random.Range(2,4);  z = -1;  }  }  } |

# Test

## Protocollo di test

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-001  REQ-001 | **Nome:** | Gestione utenti |
| **Descrizione:** | Avere una gestione utenti accessibile dalla pagina Admin, questo comprende aggiunta/modifica/eliminazione dei dati. | | |
| **Prerequisiti:** | Il database ColorfulSongs accessibile.  Pagina Admin accessibile. | | |
| **Procedura:** | 1. Aprire il proprio browser. 2. Andare all’indirizzo della pagina Admin. 3. Fare l’accesso come account Admin. 4. Aggiungere/modificare/eliminare dei dati. | | |
| **Risultati attesi:** | La pagina di Admin è raggiungibile, il login viene effettuato con successo e si possono eseguire tutte le azioni. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-002  REQ-002 | **Nome:** | Effettuare il login |
| **Descrizione:** | Poter effettuare il login all’interno del gioco stesso. | | |
| **Prerequisiti:** | Il database ColorfulSongs accessibile.  Il gioco deve comunicare con il servizio API. | | |
| **Procedura:** | 1. Aprire il proprio gioco. 2. Sul menù principale cliccare il tasto di Login. 3. Eseguire il login con le credenziali valide. 4. Reindirizzamento al menu principale, con un feedback che si è loggati all’interno del proprio account. | | |
| **Risultati attesi:** | Quando si apre il gioco, si deve poter cliccare sul pulsante che permette il Login, le credenziali devono essere verificate tramite l’API, effettuato il login si deve essere reindirizzati al menu principale e avere un feedback che si è loggato all’interno dell’account. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-003  REQ-003 | **Nome:** | Effettuare la registrazione |
| **Descrizione:** | Poter effettuare la registrazione all’interno del gioco stesso. | | |
| **Prerequisiti:** | Il database ColorfulSongs accessibile.  Il gioco deve comunicare con il servizio API. | | |
| **Procedura:** | 1. Aprire il proprio gioco. 2. Sul menù principale cliccare il tasto di Register. 3. Eseguire la registrazione con le credenziali valide. 4. Reindirizzamento al menu principale, con un feedback che l’account è stato creato correttamente | | |
| **Risultati attesi:** | Quando si apre il gioco, si deve poter cliccare sul pulsante che permette la registrazione, le credenziali devono essere verificate tramite l’API, effettuata la registrazione si deve essere reindirizzati al menu principale e avere un feedback che l’account è stato creato. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-004  REQ-004 | **Nome:** | Leaderboard Web |
| **Descrizione:** | Visualizzare la classifica tramite il proprio browser. | | |
| **Prerequisiti:** | Pagina Leaderboard accessibile.  Aver effettuato il Login sulla pagina Leaderboard. | | |
| **Procedura:** | 1. Aprire il proprio browser. 2. Andare all’indirizzo della pagina Leaderboard. 3. Consultare la Leaderboard. | | |
| **Risultati attesi:** | La pagina della Leaderboard è raggiungibile, il login viene effettuato con successo e si può visualizzare la Leaderboard. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-005  REQ-005 | **Nome:** | Login nella Leaderboard Web |
| **Descrizione:** | Visualizzare la pagina del Login dal proprio Browser | | |
| **Prerequisiti:** | Pagina Login Leaderboard accessibile.  Il database ColorfulSongs | | |
| **Procedura:** | 1. Aprire il proprio browser 2. Andare all’indirizzo della pagina di Login Leaderboard 3. Effettuare il login con credenziali valide, avere un feedback in caso di riuscita e di errore. 4. Reindirizzamento alla pagina Leaderboard. | | |
| **Risultati attesi:** | La pagina Login Leaderboard è disponibile, si deve poter effettuare il login tramite credenziali che saranno controllate con il Database, avere un feedback in caso di riuscita o di errore. Essere reindirizzati alla pagina Leaderboard. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-006  REQ-006 | **Nome:** | Creare ostacoli |
| **Descrizione:** | I diversi script di gioco creano degli ostacoli che saranno presenti nella mappa di gioco. | | |
| **Prerequisiti:** | Creare gli script degli ostacoli. | | |
| **Procedura:** | 1. Avviare il gioco 2. Premere il pulsante “Start” per avviare una nuova partita. 3. Durante il gioco verranno creati degli ostacoli | | |
| **Risultati attesi:** | Quando si gioca in ogni stanza saranno presenti degli ostacoli. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-007  REQ-007 | **Nome:** | Creazione del dungeon casuale |
| **Descrizione:** | Il dungeon deve essere creato casualmente ad ogni nuova partita per permettere un’esperienza diversa dalla precedente. | | |
| **Prerequisiti:** | Creare uno script che permetta la creazione casuale di stanze con integrazioni al REQ-006 ovvero gli ostacoli. | | |
| **Procedura:** | 1. Avviare il gioco. 2. Premere il pulsante “Start” per avviare una nuova partita. 3. Durante il gioco i dungeon saranno creati casualmente. | | |
| **Risultati attesi:** | Quando si gioca ogni stanza dovrà essere diversa dalle altre. Può capitare che la generazione casuale crei stanze uguali ma non è un errore. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-008  REQ-008 | **Nome:** | Salvataggio dell’highscore |
| **Descrizione:** | Completata la propria partita, se è il punteggio stabilito nel dungeon è più alto del precedente viene salvato come highscore. | | |
| **Prerequisiti:** | Avere il dungeon ID.  Il database ColofulSongs. | | |
| **Procedura:** | 1. Avviare il gioco. 2. Completare la partita 3. Al completamento della partita, se si è stabilito l’highscore viene salvato sul database. | | |
| **Risultati attesi:** | Quando il giocatore completa la partita, se si è stabilito un nuovo highscore, il gioco comunica tramite l’API al database di aggiornare il vecchio punteggio con il nuovo punteggio. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-009  REQ-009 | **Nome:** | Tutorial Testuale |
| **Descrizione:** | Nel menu principale, sotto le impostazioni, avere una pagina che presenta i comandi al giocatore. | | |
| **Prerequisiti:** | Possedere il gioco. | | |
| **Procedura:** | 1. Avviare il gioco. 2. Dal menu principale, andare sotto le opzioni, comandi. 3. Visualizzare i comandi con le loro spiegazioni. | | |
| **Risultati attesi:** | Il giocatore, andando sotto le impostazioni di gioco, comandi, deve poter consultare la lista dei comandi presenti nel gioco. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-010  REQ-010 | **Nome:** | Aggiungere amici |
| **Descrizione:** | Dal menu principale del gioco, sarà possibile inviare richieste di amicizia ad altri giocatori tramite il loro ID utente. | | |
| **Prerequisiti:** | Possedere il gioco. | | |
| **Procedura:** | 1. Avviare il gioco. 2. Dal menu principale, inviare la richiesta di amicizia ad un amico tramite il suo ID utente. 3. Se l’amico vuole, può accettare o rifiutare la richiesta di amicizia, avere un feedback in entrambi i casi. | | |
| **Risultati attesi:** | Il giocatore tramite l’ID del proprio amico potrà mandare una richiesta di amicizia, che dovrà poi essere accettata o rifiutata dal proprio amico, in entrambi i casi si avrà un feedback. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-011 | **Nome:** | Giocare con tastiera e/o controller |
| **Descrizione:** | Il gioco deve essere giocabile sia da una tastiera sia da un controller. | | |
| **Prerequisiti:** | Avere una tastiera e/o controller. | | |
| **Procedura:** | 1. Avviare il gioco. 2. Giocare tramite una tastiera e/o controller. | | |
| **Risultati attesi:** | Il giocatore può giocare con l a tastiera e/o il controller, in entrambi i casi devono funzionare tutte le funzionalità. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-011  REQ-011 | **Nome:** | Giocare con tastiera e/o controller |
| **Descrizione:** | Il gioco deve essere giocabile sia da una tastiera sia da un controller. | | |
| **Prerequisiti:** | Avere una tastiera e/o controller. | | |
| **Procedura:** | 1. Avviare il gioco. 2. Giocare tramite una tastiera e/o controller. | | |
| **Risultati attesi:** | Il giocatore può giocare con la tastiera e/o il controller, in entrambi i casi devono funzionare tutte le funzionalità. | | |
| **Test Case:**  **Riferimento**: | TC-012  REQ-012 | **Nome:** | Personalizzazione della skin |
| **Descrizione:** | Il giocatore tramite il negozio in gioco potrà acquistare Skin.. | | |
| **Prerequisiti:** | Possedere abbastanza monete per acquistare la Skin. | | |
| **Procedura:** | 1. Andare nella pagina dello negozio. 2. Acquistare una Skin. 3. Equipaggiare la Skin. | | |
| **Risultati attesi:** | Il giocatore a fine dungeon riceverà delle monete, con cui nel negozio potrà acquistare varie Skin ed equipaggiarle. | | |

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case:**  **Riferimento**: | TC-013  REQ-013 | **Nome:** | Tutorial Interattivo |
| **Descrizione:** | Il giocatore tramite un tutorial interattivo imparerà a giocare il gioco. | | |
| **Prerequisiti:** | Possedere il gioco. | | |
| **Procedura:** | 1. Avviare il gioco. 2. Cliccare su un pulsante apposito che porterà il giocatore nel tutorial interattivo. 3. Completare il tutorial. | | |
| **Risultati attesi:** | Il giocatore inesperto se ha difficoltà a capire come funziona il gioco, potrà avviare questo Tutorial Interattivo che spiega passo per passo come si gioca. | | |

## Risultati test

Tabella riassuntiva in cui si inseriscono i test riusciti e non del prodotto finale. Se un test non riesce e viene corretto l’errore, questo dovrà risultare nel documento finale come riuscito (la procedura della correzione apparirà nel diario), altrimenti dovrà essere descritto l’errore con eventuali ipotesi di correzione.

## Mancanze/limitazioni conosciute

Descrizione con motivazione di eventuali elementi mancanti o non completamente implementati, al di fuori dei test case. Non devono essere riportati gli errori e i problemi riscontrati e poi risolti durante il progetto.

# Consuntivo

Consuntivo del tempo di lavoro effettivo e considerazioni riguardo le differenze rispetto alla pianificazione (cap. 1.7) (ad esempio Gantt consuntivo).

# Conclusioni

Quali sono le implicazioni della mia soluzione? Che impatto avrà? Cambierà il mondo? È un successo importante? È solo un’aggiunta marginale o è semplicemente servita per scoprire che questo percorso è stato una perdita di tempo? I risultati ottenuti sono generali, facilmente generalizzabili o sono specifici di un caso particolare? ecc.

## Sviluppi futuri

Migliorie o estensioni che possono essere sviluppate sul prodotto.

## Considerazioni personali

Cosa ho imparato in questo progetto? ecc.

# Glossario

|  |  |
| --- | --- |
| **Termine** | **Descrizione** |
| Godot Engine | Engine per sviluppare videogiochi 2D/3D in GDscript |

# Indice delle figure

[Figura 1: Esempio di diagramma di Gantt. 7](#_Toc188452039)

[Figura 2: Schema E-R 9](#_Toc188452040)

[Figura 3: Schema logico 9](#_Toc188452041)

[Figura 4: Design interfaccia di Login 10](#_Toc188452042)

[Figura 5: Design interfaccia di Registrazione 11](#_Toc188452043)

[Figura 6: Design di una stanza 11](#_Toc188452044)

[Figura 7: Design del percorso con pedane finte 12](#_Toc188452045)

[Figura 8: Design dell’enigma “Percorso con pedana invisibile e soluzione sul muro” 13](#_Toc188452046)

# Bibliografia

## Bibliografia per articoli di riviste:

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo dell’articolo (tra virgolette),
3. Titolo della rivista (in italico),
4. Anno e numero
5. Pagina iniziale dell’articolo.

## Bibliografia per libri

1. Cognome e nome (o iniziali) dell’autore o degli autori, o nome dell’organizzazione,
2. Titolo del libro (in italico),
3. ev. Numero di edizione,
4. Nome dell’editore,
5. Anno di pubblicazione,
6. ISBN.

## Sitografia

* URL del sito (se troppo lungo solo dominio, evt completo nel diario),
* Eventuale titolo della pagina (in italico),
* Data di consultazione (GG-MM-AAAA)
* <https://www.youtube.com/watch?v=dclc8w6JW7Y>, *Tutorial Godot ‘ThinkWithGames’,* 15-01-2025
* <https://www.youtube.com/watch?v=8HvcHtauKoc>, *Tutorial Collision* *‘ThinkWithGames’*, 15-01-2025

**Esempio:**

* http://standards.ieee.org/guides/style/section7.html, *IEEE Standards Style Manual*, 07-06-2008.

# Allegati

Elenco degli allegati, esempio:

* Diari di lavoro
* Codici sorgente/documentazione macchine virtuali
* Istruzioni di installazione del prodotto (con credenziali di accesso) e/o di eventuali prodotti terzi
* Documentazione di prodotti di terzi
* Eventuali guide utente / Manuali di utilizzo
* Mandato e/o QdC
* Prodotto
* …