

UNIVERSITÀ DEGLI STUDI DI SALERNO

DIPARTIMENTO DI INFORMATICA



Corso di Laurea in INFORMATICA

**Progettazione e sviluppo di un ambiente collaborativo nel
metaverso**

Relatore:

Prof.ssa

Rita Francese

Candidato:

Luca Memoli

Mat. 0512105467

ANNO ACCADEMICO 2022/2023

INDICE

1 Introduzione.....	4
2 Background e tecnologie utilizzate	6
2.1 Metaverso.....	6
2.2 WebGL	7
2.3 Typescript	8
2.4 Formato dei modelli	8
2.5 Unity	9
2.6 Blender	10
2.6 Decentraland	11
2.6.1 Strumenti di Decentraland per i creatori	13
2.6.2 Creazione di una scena (Decentraland-Editor).....	17
2.6.3 Creazione di una scena (CLI).....	20
2.7 Spatial	24
2.7.1 Creazione di una scena (Spatial).....	25
2.7.2 Creazione di una scena (Spatial-SDK)	29
3 Soluzione proposta.....	33
3.1 Scena porta a combinazione (Decentraland)	33
3.1.1 Analisi del problema	33
3.1.2 Modellazione della scena	34
3.1.3 Fase di scripting	35
3.1.4 Risultato finale.....	44
3.2 Scena Spatial.....	49
3.2.1 Analisi del problema	49
3.2.2 Modellazione della scena	49
3.2.3 Fase di sviluppo	50
3.2.4 Risultato finale.....	53
4 Conclusioni e sviluppi futuri	56
Riferimenti	57

1 Introduzione

Il metaverso è un'ipotetica iterazione di internet come unico mondo virtuale e immersivo, attraverso l'utilizzo di specifiche apparecchiature.

L'avanzamento tecnologico nel campo della realtà virtuale, con la nascita e lo sviluppo dei visori, sta contribuendo alla concretizzazione di questa nuova realtà.

Attualmente sono diverse le piattaforme che si stanno focalizzando sullo sviluppo di questa nuova realtà, in particolare Decentraland e Spatial.

Lo studio di queste due piattaforme ha come scopo la realizzazione di un ambiente realistico, come ad esempio un'anteprima virtuale di una stanza oppure la trasposizione virtuale di un'aula scolastica.

L'analisi della struttura e delle funzionalità di quest'ultime ha permesso una comprensione più chiara di ciò che sia realmente realizzabile in questo nuovo ambiente.

Inoltre, sono stati realizzati due progetti al fine di mostrare le potenzialità di queste due piattaforme, come ad esempio, un sistema di sicurezza basato sull'inserimento di un codice numerico per accedere ad una stanza oppure la possibilità di sedersi all'interno della scena.

In particolare, entrambe le piattaforme utilizzano il motore grafico Unity, principalmente utilizzato nello sviluppo di videogiochi.

In Decentraland, modificando il codice in Typescript della scena, è stato possibile realizzare componenti che si occupano di:

- gestire l'animazione dei modelli;
- realizzare pulsanti interattivi;
- gestire la telecamera dell'utente in particolare aree;
- verificare la combinazione immessa dall'utente.

In Spatial, la gestione della scena viene effettuata tramite l'editor di Unity.

L'editor permette di assegnare specifiche funzionalità agli elementi presenti nella scena, ad esempio, è possibile creare un pannello per la visualizzazione di contenuti multimediali.

Entrambe permettono la creazione di un ambiente nel metaverso seppur con alcune differenze, come ad esempio, in Decentraland per poter rendere pubblica la propria scena occorre acquistare un terreno mentre in Spatial ciò non necessario. In conclusione, seppur entrambe ancora in fase di sviluppo, l'attuale interesse in questo campo farà sì che ambedue le piattaforme possano ampliare il proprio panorama di funzionalità.

Il lavoro di tesi è articolato come segue:

- nel capitolo 2 è prevista una panoramica generale sui concetti e le tecnologie utilizzate per la realizzazione di un ambiente nel metaverso;
- nel capitolo 3 sono state presentate due scene per mostrare le potenzialità di queste due piattaforme;
- nel capitolo 4 sono state tratte delle conclusioni e mostrati dei possibili sviluppi futuri in questo campo.

2 Background e tecnologie utilizzate

Nel seguente capitolo verrà presentata una panoramica generale sul metaverso e sulla progettazione di un ambiente utilizzando le piattaforme Decentraland e Spatial. Di seguito alcune definizioni e strumenti essenziali per la comprensione dell'argomento.

2.1 Metaverso

Il **metaverso** è un'ipotetica iterazione di Internet come un unico, condiviso, immersivo, persistente, spazio 3D virtuale dove è possibile vivere esperienze di vita inattuabili nel mondo reale.

Il termine "**metaverso**" proviene dal romanzo di fantascienza di Neal Stephenson *Snow Crash* scritto nel 1992 , ed risulta essere la combinazione dei suffissi:

- "**meta-**": che proviene dalla metafisica Aristotelica ed indica "ciò che va oltre";
- "**-universo**": dal latino "*universus*" ovvero interezza dello spazio.

Lo scrittore descrive il Metaverso come un'immensa sfera nera nella quale ogni persona può realizzare in 3D ciò che desidera: negozi, uffici e altro; il tutto potenzialmente visitabile dagli utenti.

Con il crescente sviluppo di tecnologie quali i visori di realtà virtuale (**VR**) o le lenti di realtà aumentata (**AR**), si sta favorendo la realizzazione di questo utopistico spazio. Nel 2003, abbiamo una prima implementazione del metaverso con la piattaforma Second Life, in quanto rappresentava gli utenti sottoforma di avatar virtuali in uno spazio tridimensionale.

Recentemente, Facebook è stata rinominata "Meta" e il suo direttore Mark Zuckerberg ha assicurato l'impegno della compagnia nello sviluppo di questo settore. Al momento, le imprese hanno rivolto il proprio interesse al metaverso, considerandola un'ottima opportunità per la creazione di un'esperienza lavorativa in remoto più realistica ed immersiva rispetto al passato.

2.2 WebGL

Durante lo studio sulle due piattaforme, è emerso che entrambe utilizzano WebGL per la creazione dei propri ambienti tridimensionali.

WebGL (Web-based Graphics Library) è un'API JavaScript per il rendering di grafica 2D e 3D interattiva eseguibile all'interno di qualsiasi browser web compatibile, senza l'uso di plug-in esterni o di una scheda video altamente performante.

Un'**API**, acronimo di **Application Programming Interface**, è interfaccia che permette la comunicazione tra diversi computer o tra diversi software o tra diversi componenti di software tra di loro.

Spesso, tale termine viene usato per indicare le librerie software di un linguaggio di programmazione.

JavaScript è un linguaggio di scripting lato client utilizzato per rendere interattive le pagine web, comunemente utilizzato nella programmazione Web lato client per la creazione, in siti web e applicazioni web, di effetti dinamici a seguito della interazione dell'utente con la pagina.

Gli elementi WebGL possono essere aggiunti ad elementi di una pagina o essere usati come background.

I programmi in WebGL sono composti da:

- codice per la gestione generale scritto in JavaScript;
- codice shader scritto in OpenGL ES Shading Language (GLSL ES), un linguaggio simile a C o C++, eseguito dall'unità di elaborazione grafica (GPU) del computer.

La natura di basso livello delle WebGL API favorisce una grafica 3D veloce e piacevole, contribuendo allo sviluppo di librerie specializzate nella creazione di grafica in 3D (per esempio trasformazioni che modificano la dimensione e la posizione di modelli all'interno della scena).



2.3 Typescript

La piattaforma Decentraland per la componente di scripting delle scene utilizza il linguaggio TypeScript.

Esso viene utilizzato per diverse operazioni sulla piattaforma:

- per la creazione di oggetti;
- per il caricamento di texture;
- per la gestione della fisica, dei pagamenti e delle interazioni tra gli utenti.

Typescript è un linguaggio di programmazione open source fortemente tipizzato sviluppato da Microsoft nel 2012.

Il linguaggio estende la sintassi di JavaScript, cosicché, qualsiasi codice scritto in **JavaScript**, è anche compatibile con la sintassi e la semantica TypeScript.

Le funzionalità di auto completamento e controllo del tipo velocizzano i tempi di sviluppo delle scene e garantiscono una maggiore solidità al codice.



2.4 Formato dei modelli

Ogni scena è caratterizzata dalla presenza di oggetti 3D, i quali possono essere:

- semplici: es. cubi e sfere;
- più complessi: es. una sedia o una stanza.

Su entrambe le piattaforme è possibile importare modelli con estensioni .gltf o .glb.

Il **glTF** (diminutivo di **Graphics Language Transmission Format**) è un formato standard per modelli e scene tridimensionali, sviluppato da Khronos Group nel 2015.

I file glTF solitamente hanno due estensioni:

- .gltf (JSON/ASCII)
- .glb (binario)

Il **Glb** (diminutivo di GL Transmission Format Binary file) è la forma binaria di un file con estensione gltf , ed è stato introdotto come estensione di glTF 1.0 e incorporato direttamente in glTF 2.0.

I file glTF necessitano di allegare risorse esterne (materiali, gerarchie e camere) oltre al modello in 3D, mentre i file glb contengono già al loro interno tutte queste risorse (materiali, texture, ...) in un singolo file compresso.



2.5 Unity

Entrambe le piattaforme, pur usando differenti strumenti per la creazione di ambienti, si basano su Unity.

Unity è un motore di gioco multiplatforma sviluppato da Unity Technologies utilizzato prevalentemente nello sviluppo di videogiochi e contenuti interattivi, quali visualizzazioni architettoniche o animazioni 3D in tempo reale.

Il motore può essere usato per la creazione di giochi sia tridimensionali che bidimensionali, così come per lo sviluppo di simulazioni interattive e altre esperienze. Ci sono innumerevoli industrie che lo utilizzano non solo in campo videoludico ma anche per la realizzazione di film, in architettura, nell'industria automobilistica o in campo ingegneristico.

Il motore offre una API di scripting in C# usando Mono (framework), per l'editor sotto forma di plugin, che per i giochi stessi con una funzionalità di tipo “drag and drop”. Un framework è un'architettura logica di supporto sulla quale un software può essere progettato e realizzato.



2.6 Blender

Nella creazione e modifica dei modelli per entrambe le piattaforme è stato utilizzato il software Blender.

Blender è un software libero e multiplatforma sviluppato nel 1998 di:

- Modellazione, manipolazione e modifica di oggetti.
- Rigging: tecnica di animazione digitale, che attraverso la rappresentazione di oggetti in due parti : **mesh** (rappresentazione della sua superficie) e **rig** (un insieme di ossa interconnesse) permette l'animazione di questi ultimi;
- montaggio video;
- composizione, rendering e texturing di immagini tridimensionali e bidimensionali.

Dispone inoltre di funzionalità per mappature UV, simulazioni di fluidi, di rivestimenti, di particelle, altre simulazioni non lineari e creazione di applicazioni/giochi in 3D.

Blender ha fama di essere un programma difficile da imparare in quanto quasi tutte le funzioni possono essere richiamate con scorciatoie da tastiera.

L'interfaccia presenta due modalità di gestione degli oggetti:

- la modalità oggetto che viene usata principalmente per manipolare la posizione, la rotazione e la scala di oggetti singoli all'interno della scena
- la modalità modifica che viene usata generalmente per la modifica i vertici della mesh degli oggetti.

Da quando è stato pubblicato come opensource, la **GUI** (interfaccia grafica) è stata notevolmente modificata, introducendo la possibilità di modificare il colore, l'uso di widget trasparenti, una nuova e migliorata visualizzazione e gestione dell'albero degli oggetti e altre piccole migliorie (scelta diretta dei colori, ecc.).

2.6 Decentraland

Decentraland nasce come un esperimento per assegnare il possesso di un'immobile digitale agli utenti sulla blockchain.

Questo mondo digitale era inizialmente implementato come una griglia infinita in 2D di pixel, dove ogni pixel conteneva dei metadati che identificavano il proprietario e ne descrivevano il colore a lui assegnato.

Ad oggi, Decentraland è una piattaforma decentralizzata di realtà aumentata supportata dalla blockchain Ethereum.

Con la piattaforma di Decentraland gli utenti possono creare, sperimentare nuove esperienze e monetizzare i propri contenuti e applicazioni.

Lo spazio virtuale attraversabile con Decentraland è chiamato LAND, un pacchetto digitale non replicabile gestito da uno smart contract Ethereum.

Le Land sono divise in parcels(terreni), identificati da coordinate cartesiane (x, y).

Queste terre vengono acquistate permanentemente dai membri della community e sono pagate in MANA, la criptovaluta di Decentraland.

Ciò offre agli utenti il pieno controllo sull'ambiente e sulle applicazioni che essi sono intenzionati a realizzare.

La piattaforma offre alle aziende la possibilità di promuovere i propri prodotti, servizi ed eventi, con l'utilizzo di cartelloni virtuali posizionati in spazi con un maggior afflusso di utenti.



Decentraland

Alcuni quartieri possono diventare la versione virtuale di un quartiere realmente esistente, come ad esempio Times Square (New York).

Ciascun'azienda potrà mostrare i propri prodotti e creare un'esperienza condivisa per pubblicizzarsi.

Ci sono dei requisiti minimi da rispettare per eseguire correttamente Decentraland, ciò dipende se viene eseguito su web client o su client desktop.

Desktop Client	
Minimum GPU	<ul style="list-style-type: none"> • AMD Radeon HD 6700 series or equivalent
Recommended GPU	<ul style="list-style-type: none"> • AMD Radeon RX 500 series or equivalent
Minimum GPU MEMORY:	<ul style="list-style-type: none"> • 1GB
Recommended GPU MEMORY:	<ul style="list-style-type: none"> • 4GB
Web Client	
Minimum GPU	<ul style="list-style-type: none"> • Intel HD/UHD 9th gen or equivalent
Recommended GPU	<ul style="list-style-type: none"> • Geforce 900 series or equivalent
Minimum RAM:	<ul style="list-style-type: none"> • 4GB
Recommended RAM	<ul style="list-style-type: none"> • 16GB

2.6.1 Strumenti di Decentraland per i creatori

Ci sono alcuni tools messi a disposizione da Decentraland che permettono di organizzare i modelli 3D all'interno delle scene.

Tuttavia, per aggiungere funzionalità interattive agli oggetti nella scena è richiesto un SDK (acronimo di **Software Development Kit**), cioè un insieme di strumenti per lo sviluppo e la documentazione di software.

- **DCL Edit:** è un tool costruito dalla community che permette di effettuare il drag and drop dei modelli 3D all'interno della propria scena.

Questo tool velocizza i tempi di posizionamento degli oggetti all'interno della scena, permettendo al creatore di focalizzarsi maggiormente sulla componente interattiva.

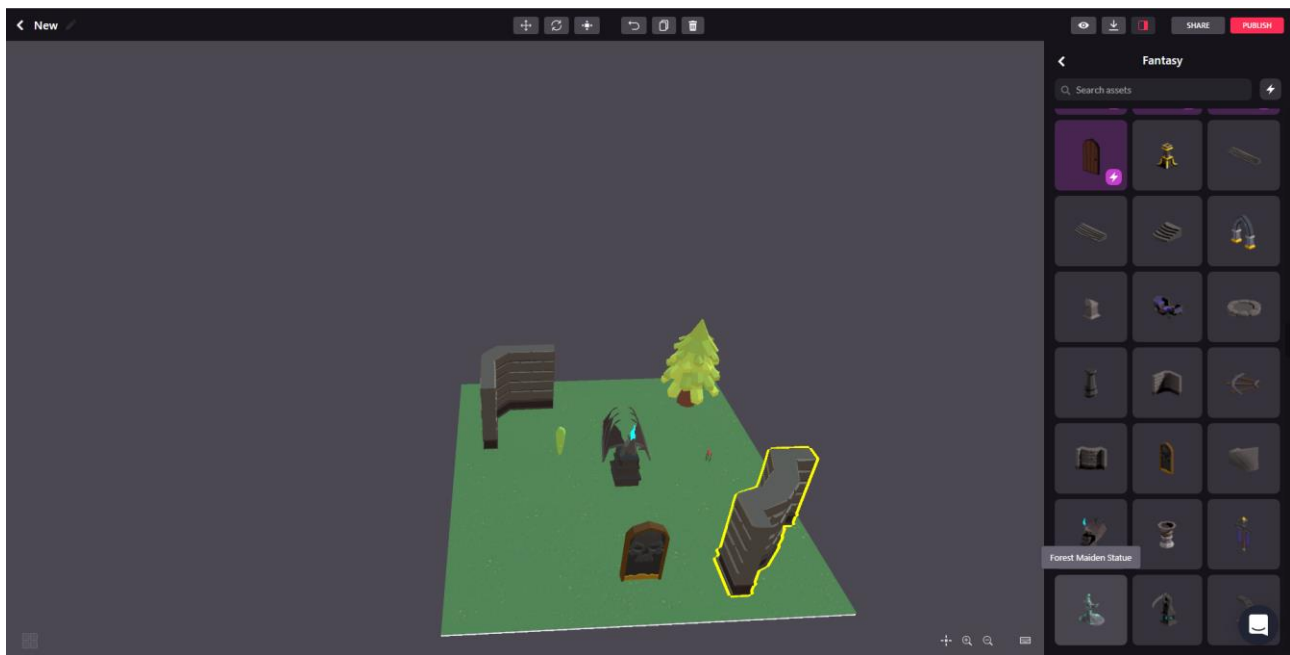
Al termine dell'utilizzo è possibile aggiungere le funzionalità di interattività usando l'SDK.



- **Legacy Builder:** è un editor con funzionalità drag and drop.

Per il suo utilizzo non viene richiesta nessuna competenza da programmatore, in quanto alcune funzionalità sono state già implementate in esso.

Al termine della creazione della scena è possibile esportare il risultato e aggiungere l'interattività agli oggetti usando l'SDK.



Il builder viene eseguito sul web browser; la sua interfaccia è composta da pulsanti:

- per il posizionamento (spostamento, rotazione e scala);
- duplicazione;
- cancellazione di oggetti.

È possibile visionare un'anteprima della scena, esportarla in locale e pubblicarla sulla piattaforma.

Inoltre, l'editor possiede una raccolta molto vasta di asset per la personalizzazione della scena.

L'SDK di Decentraland è uno strumento che permette l'integrazione di differenti funzionalità all'interno delle scene mediante la scrittura di codice Typescript.

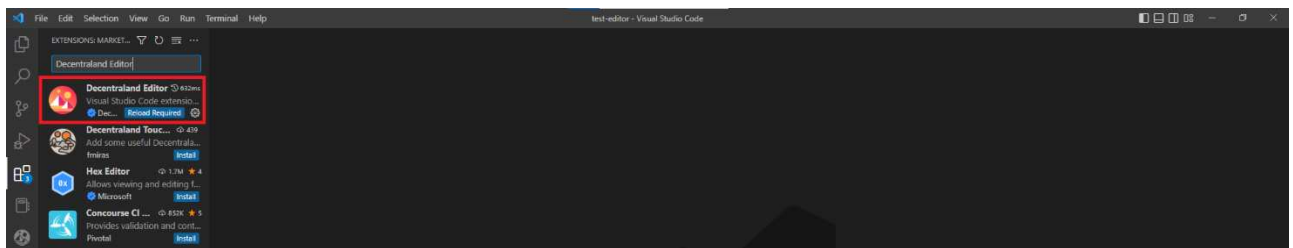
Ci sono due modi per creare una scena:

- **Decentraland Editor:** è una estensione dell'editor Visual Studio Code, che permette di creare, mostrare una preview ed eseguire il deploy della scena.

Visual Studio Code è un editor di codice sviluppato da Microsoft nel 2015.

L'editor offre differenti strumenti per agevolare il compito del programmatore, come ad esempio, il supporto al debugging, colorazione della sintassi e auto-completamento del codice.

L'editor possiede un'interfaccia semplificata che nasconde all'utente l'esecuzione di comandi sulla CLI.



- La **Command Line Interface** (CLI): un'interfaccia a riga di comando, basata su comandi testuali per eseguire azioni, come ad esempio, inizializzazione di una scena, preview di una scena, ...


```
C:\Windows\system32\cmd.exe
C:\Users\Luca\Desktop\Progetti_Tesi\combi>dcl -help

Decentraland CLI

Usage: dcl [command] [options]

Commands:

  init          Create a new Decentraland Scene project
  build         Build scene
  start         Start a local development server for a Decentraland Scene
  install       Sync decentraland libraries in bundleDependencies
  install package Install a package
  deploy        Upload scene to a particular Decentraland's Content server
  deploy-deprecated Upload scene to Decentraland's legacy content server (deprecated).
  export        Export scene to static website format (HTML, JS and CSS)
  info [args]   Displays information about a LAND, an Estate or an address
  status [args] Displays the deployment status of the project or a given LAND
  help [cmd]    Displays complete help for given command
  version       Display current version of dcl
  coords        Set the parcels in your scene
  workspace subcommand Make a workspace level action, dcl help workspace for more information.

Options:

  -h, --help    Displays complete help for used command or subcommand
  -v, --version  Display current version of dcl

Example:

  - Show complete help for the subcommand "deploy"

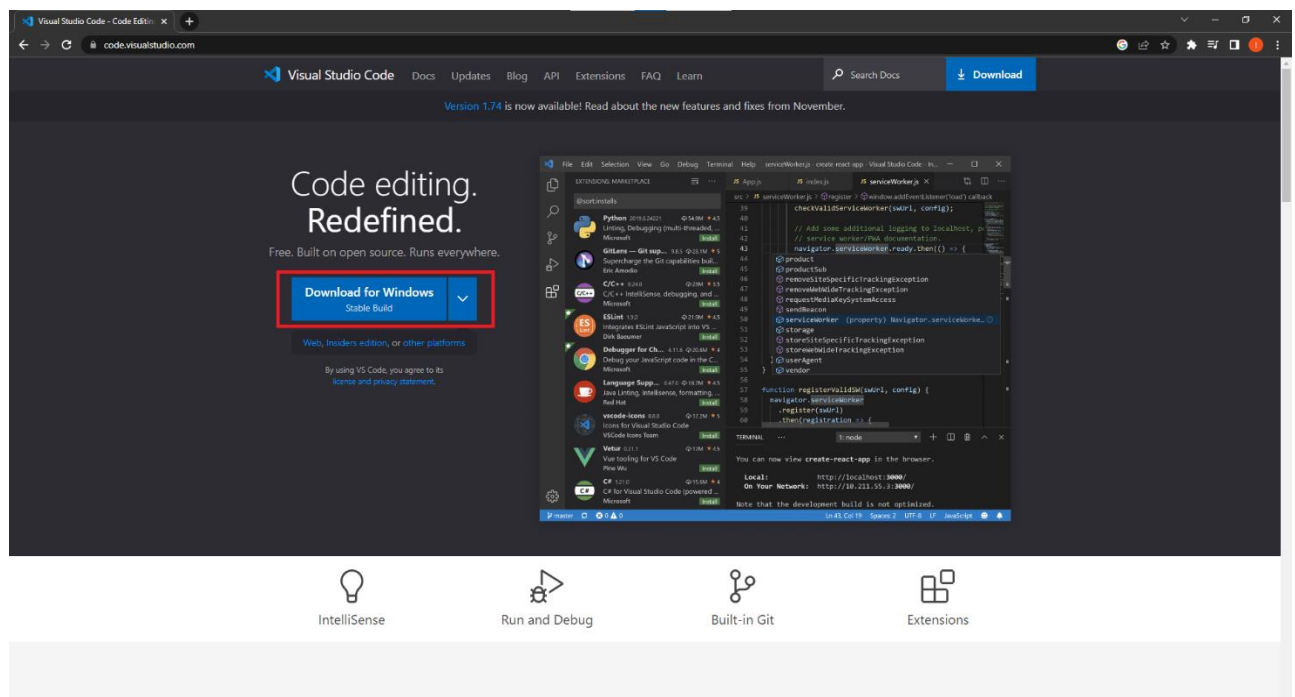
  $ dcl help deploy

C:\Users\Luca\Desktop\Progetti_Tesi\combi>
```

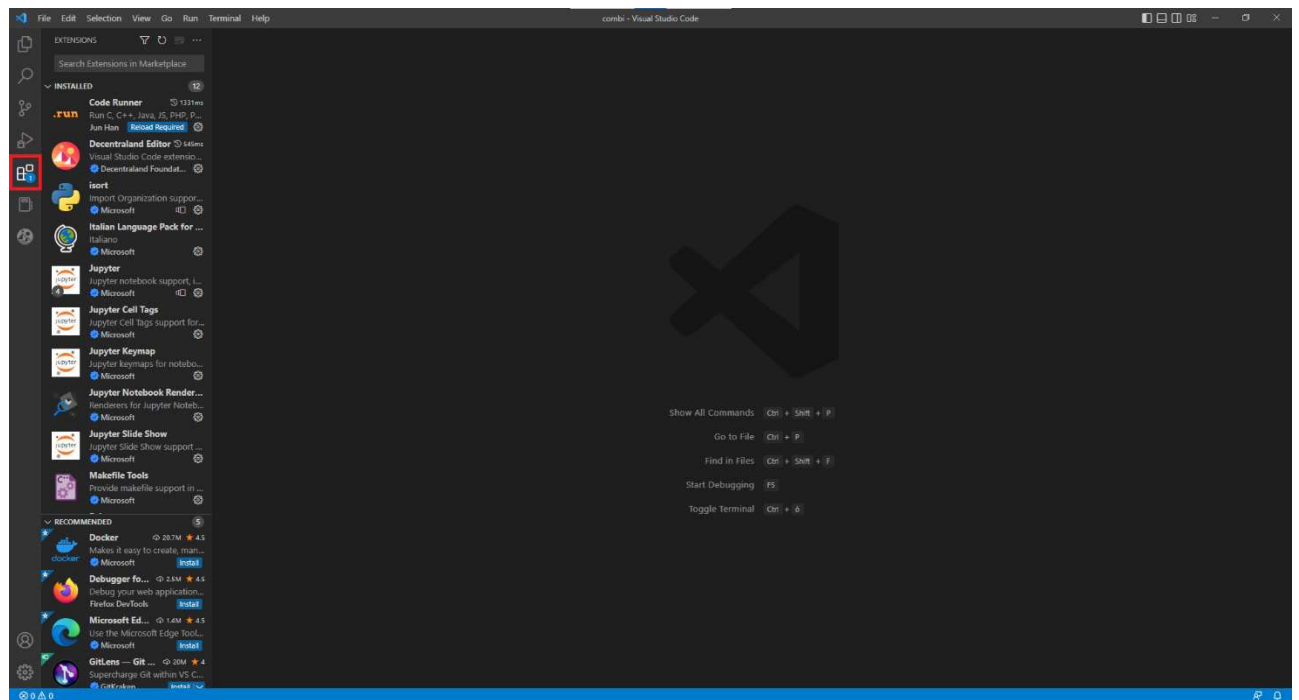
2.6.2 Creazione di una scena (Decentraland-Editor)

Prima di procedere occorre:

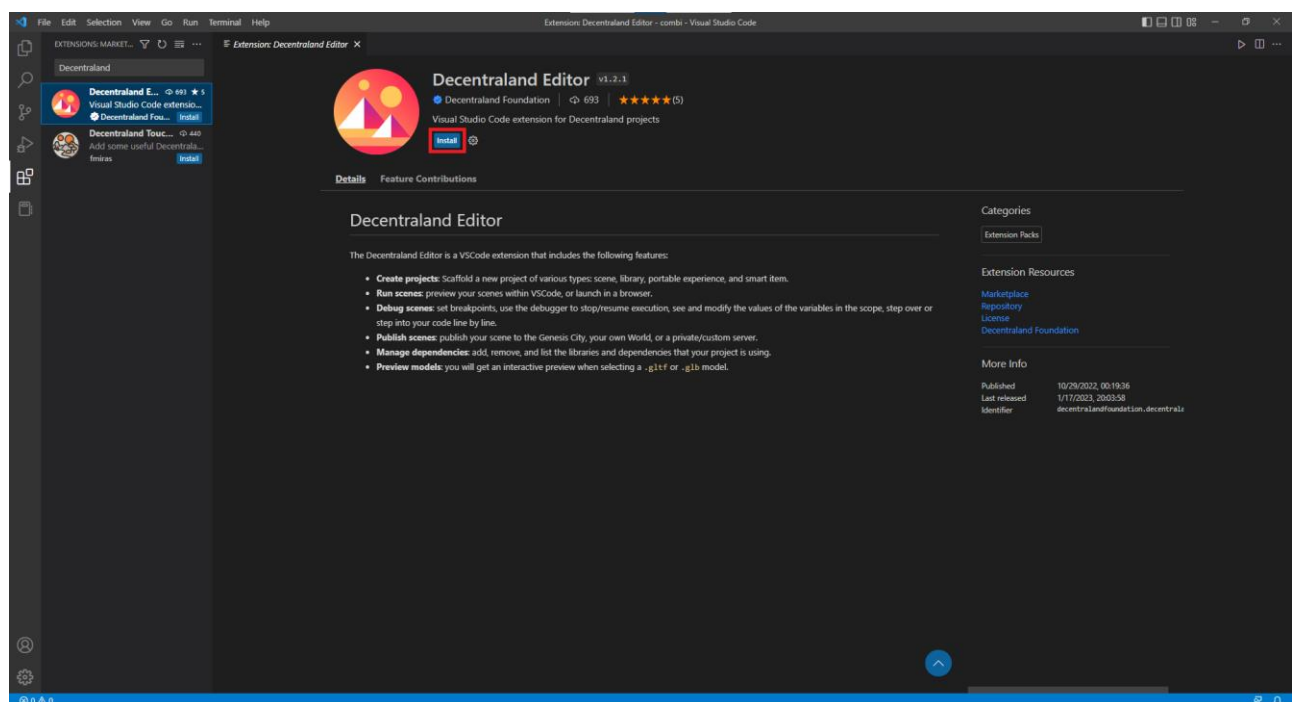
- 1) installare Visual Studio Code reperibile al link: <https://code.visualstudio.com/>



- 2) Eseguire Visual Studio Code e selezionare **Extensions** presente sulla barra a sinistra.

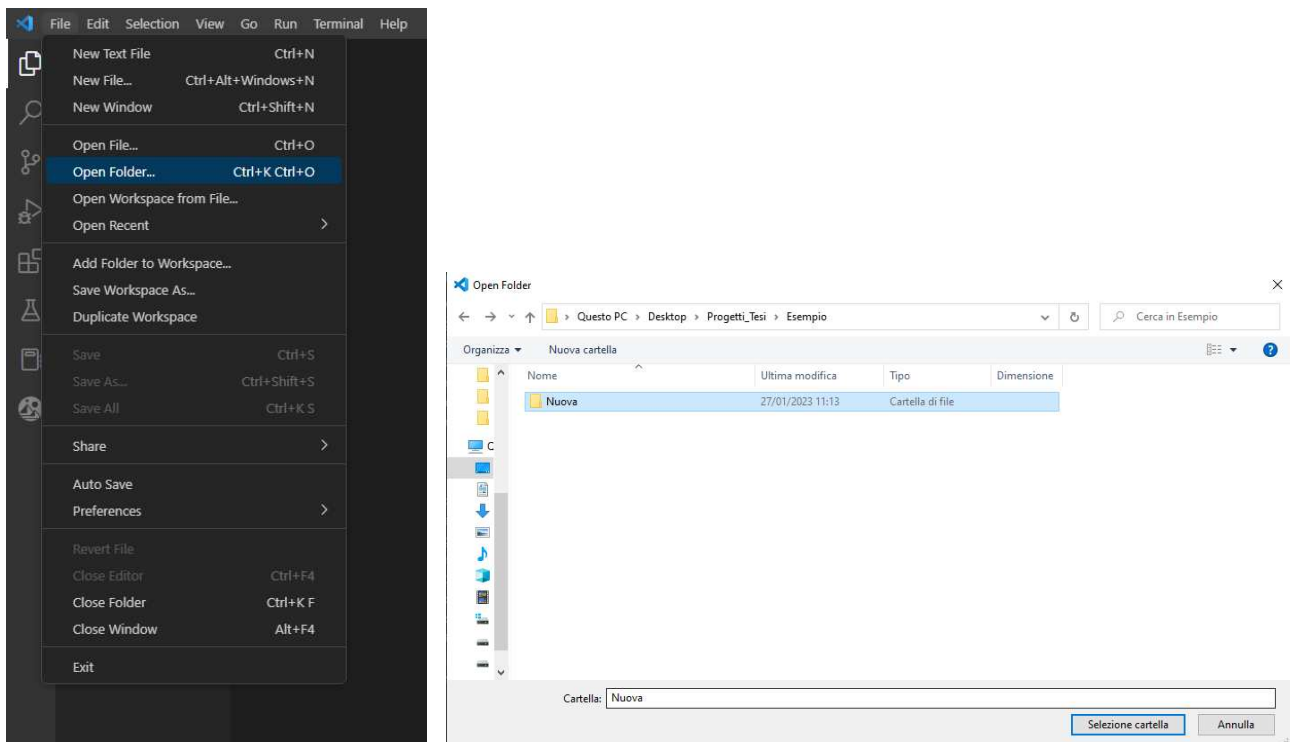


- 3) Cercare nella barra di ricerca **Decentraland Editor**, e successivamente cliccare su **Install**.

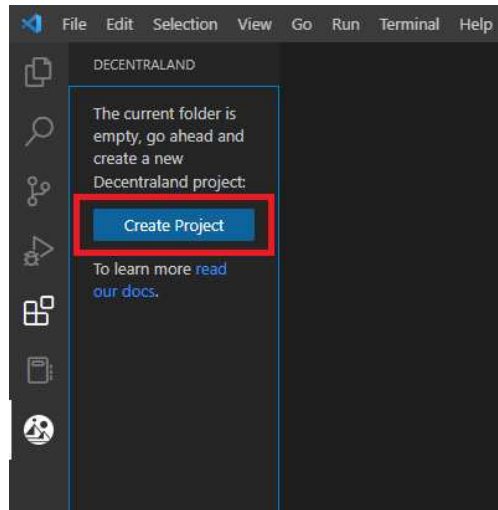


Dopo aver installato l'editor di Decentraland è necessario:

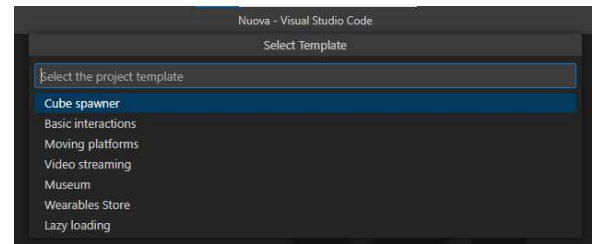
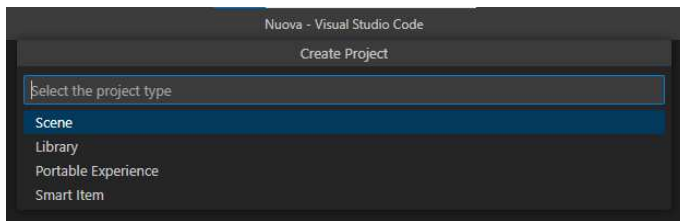
1. Dalla finestra di Visual Studio, aprire una cartella vuota.



2. Selezionare Decentraland sul margine sinistro di Visual Studio, e cliccare su **Create Project**.

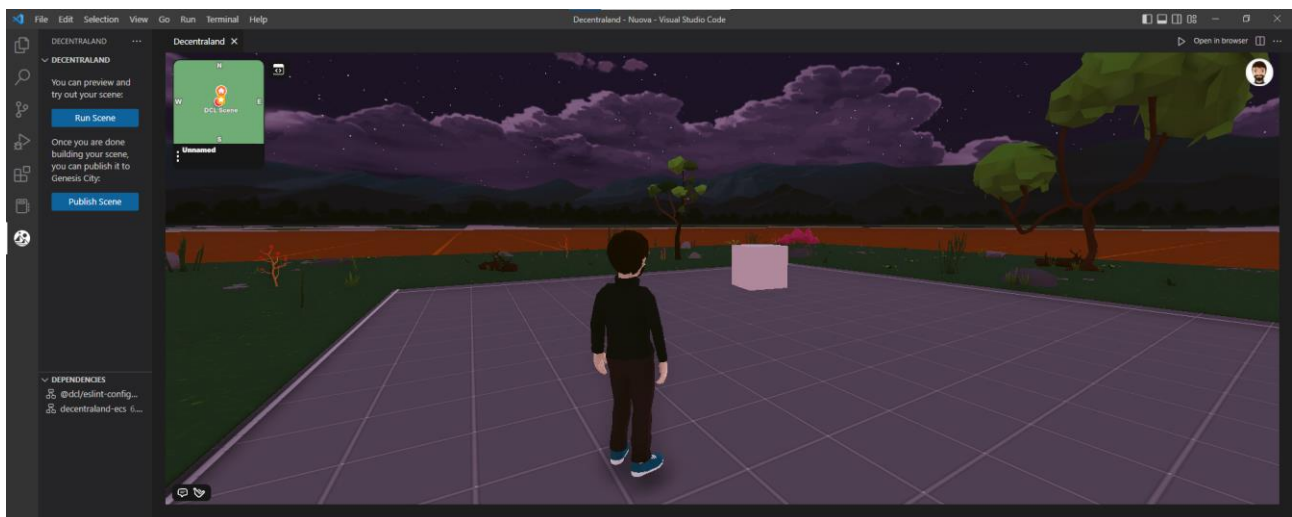


3. L'editor ci chiederà quale tipo di progetto si vuole sviluppare : selezionare **Scene** nel menù a tendina.



4. Scegliamo un template e attendiamo che l'editor installi le dipendenze nella cartella del progetto.

Infine, cliccando su **Run Scene**, si aprirà una finestra a lato dove sarà possibile visionare un'anteprima della scena.



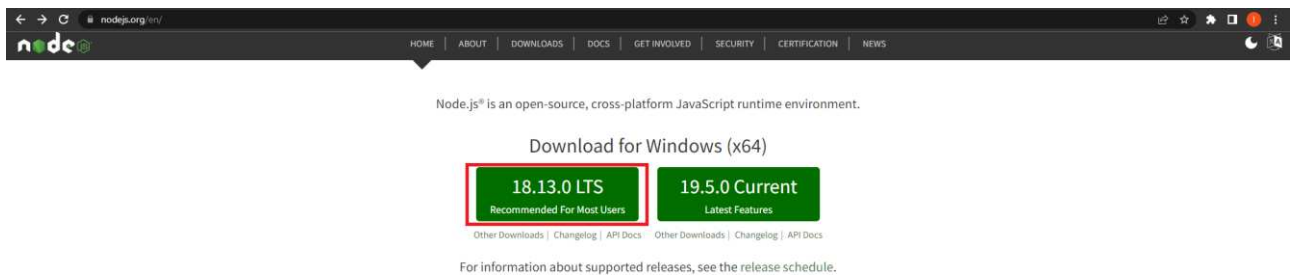
2.6.3 Creazione di una scena (CLI)

Per utilizzare i comandi della CLI occorre installare il software Node.js.

Node.js è un ambiente run-time multiplatforma orientato ad eventi per l'esecuzione di codice JavaScript.

Esso consente di utilizzare JavaScript anche per scrivere codice da eseguire lato server, ad esempio per la produzione del contenuto delle pagine web dinamiche, prima che la pagina venga inviata al browser dell'utente.

- 1) Installiamo il software Node.js(versione 8 o successiva) collegandoci a :
<https://nodejs.org/en/>



- 2) Dopo averlo installato, aprire il terminale ed eseguire il comando :
npm install -g decentraland

```
Ca. Prompt dei comandi
Microsoft Windows [Versione 10.0.19045.2486]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Luca>npm install -g decentraland
npm WARN deprecated @types/form-data@2.5.0: This is a stub types definition. form-data provides its own type definitions
, so you do not need this installed.
npm WARN deprecated stable@0.1.8: Modern JS already guarantees Array#sort() is a stable sort, so this library is depreca
ted. See the compatibility table on MDN: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Object
s/Array/sort#browser_compatibility
npm WARN deprecated interface-ipfs-format@1.0.1: This module has been superseded by the multiformats module
npm WARN deprecated opn@6.0.0: The package has been renamed to `open`
npm WARN deprecated cids@1.1.9: This module has been superseded by the multiformats module
npm WARN deprecated multibase@4.0.6: This module has been superseded by the multiformats module
npm WARN deprecated multicodec@3.2.1: This module has been superseded by the multiformats module
npm WARN deprecated ipfs-dag-pb@0.22.3: This module has been superseded by @ipfs/dag-pb and multiformats

added 4 packages, changed 425 packages, and audited 430 packages in 59s

47 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities

C:\Users\Luca>
```

- 3) Posizionarsi in una cartella vuota, ed eseguire il comando : **dcl init**

```
Ca. npm
Microsoft Windows [Versione 10.0.19045.2486]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Luca\Desktop\Progetti_Tesi\Esempio\Nuova>dcl init
? Choose a project type (Use arrow keys)
> Scene
  Smart Item
  Smart Wearable (Beta)
  Library
```

- 4) Selezionando **Scene**, possiamo creare una scena partendo dai template presenti sulla piattaforma.

```
npm
Microsoft Windows [Versione 10.0.19045.2486]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Luca\Desktop\Progetti_Tesi\Esempio\Nuova>dcl init
? Choose a project type Scene
? Choose a scene (Use arrow keys)
> (1) Cube spawner
  (2) Basic interactions
  (3) Moving platforms
  (4) Video streaming
  (5) Museum
  (6) Wearables Store
  (7) Lazy loading
(Move up and down to reveal more choices)
```

5) Dopo aver selezionato il template attendiamo che le dipendenze vengano installate.

```
C:\Windows\system32\cmd.exe
Microsoft Windows [Versione 10.0.19045.2486]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Luca\Desktop\Progetti_Tesi\Esempio\Nuova>dcl init
? Choose a project type Scene
? Choose a scene (1) Cube spawner
✓ Example downloaded
✓ Dependencies installed.

Success! Run 'dcl start' to see your scene

C:\Users\Luca\Desktop\Progetti_Tesi\Esempio\Nuova>
```

Per eseguire la scena utilizziamo il comando: **dcl start** ,che aprirà una pagina sul browser dal quale sarà possibile visionare un'anteprima della scena.


```
C:\Windows\system32\cmd.exe
Success! Run 'dcl start' to see your scene

C:\Users\Luca\Desktop\Progetti_Tesi\Esempio\Nuova>dcl start
✓ Latest SDK installation found.
✓ Checking decentraland libraries

> dcl-project@1.0.0 watch
> build-ecs --watch

> dev mode: true
> working directory: C:\Users\Luca\Desktop\Progetti_Tesi\Esempio\Nuova

> processing C:/Users/Luca/Desktop/Progetti_Tesi/Esempio/Nuova/src/game.ts
> bundling:
  - node_modules\@dcl\amd\dist\amd.js
  - node_modules\decentraland-ecs\dist\src\index.js
  - src\game.ts
> writing C:/Users/Luca/Desktop/Progetti_Tesi/Esempio/Nuova/bin/game.js
> writing C:/Users/Luca/Desktop/Progetti_Tesi/Esempio/Nuova/bin/game.js.lib

The compiler is watching file changes...

<<< Initializing components >>>
<<< Wiring app >>>
<<< Starting components >>>
2023-01-29T12:17:32.705Z [LOG] (http-server): Listening 0.0.0.0:8000

Preview server is now running

Available on:

  http://127.0.0.1:8000?position=0%2C0&SCENE_DEBUG_PANEL
  http://169.254.85.118:8000?position=0%2C0&SCENE_DEBUG_PANEL
  http://192.168.1.3:8000?position=0%2C0&SCENE_DEBUG_PANEL

Details:

Press CTRL+C to exit
```

2.7 Spatial

Spatial è una piattaforma di realtà aumentata in cui l'utente può sperimentare il metaverso attraverso la creazione o la visita di stanze create da altri utenti.

La piattaforma offre un grande supporto ai creatori ed ai grandi marchi nella costruzione del proprio spazio all'interno del metaverso.

La semplicità della sua interfaccia permette:

- il caricamento di contenuti multimediali (video, immagini o pdf);
- il caricamento di modelli 3D;
- la gestione dell'ambiente e dello skybox (sfondo) della scena;
- il collegamento con altri spazi creati sulla piattaforma, attraverso i portali;
- la funzionalità di Share-Screen, ovvero la possibilità di condividere una finestra o un'applicazione dell'utente all'interno della scena.

Quest' ambiente offre la possibilità di incontrare persone da tutto il mondo, visitare gallerie d'arte, creare e condividere il proprio spazio personale attraverso un'interfaccia semplificata adatta a qualsiasi tipo di utente.

Recentemente, la piattaforma si sta focalizzando sullo sviluppo di un toolkit per agevolare lo sviluppo ai creatori chiamato **Spatial Creator Toolkit**.

Spatial Creator Toolkit è basato su Unity, include strumenti e funzioni che permettono la gestione dell'ambiente 3D e non solo.

Nonostante sia ancora in fase di sviluppo, attualmente esso offre alcune funzionalità come la:

- creazione di **pannelli** per l'inserimento di immagini o finestre di share-screen, particolarmente utili, ad esempio per l'allestimento di una mostra d'arte;
- gestione dei **punti di spawn**, ovvero l'area in cui verranno generati gli avatar degli utenti che accedono alla scena;
- costruzione di **basi** su cui l'avatar potrà sedersi all'interno della scena.

Per usufruire della piattaforma, tuttavia, è necessario che il computer supporti WebGL 2.0 e venga usato un browser compatibile :

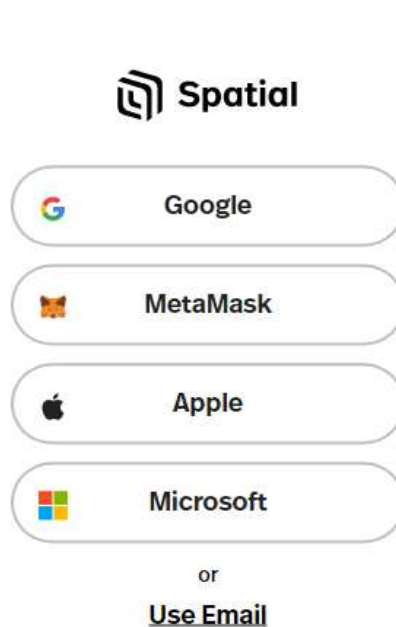
- Chrome >= 72, Firefox >= 72, Safari >= 15, Edge >= 79.

2.7.1 Creazione di una scena (Spatial)

Prima di cominciare è opportuno registrarsi a Spatial.io collegandosi al sito

<https://www.spatial.io/>

Cliccando su login e successivamente registrarsi attraverso il menu in basso:



Inserire i campi e successivamente eseguire login sulla piattaforma.



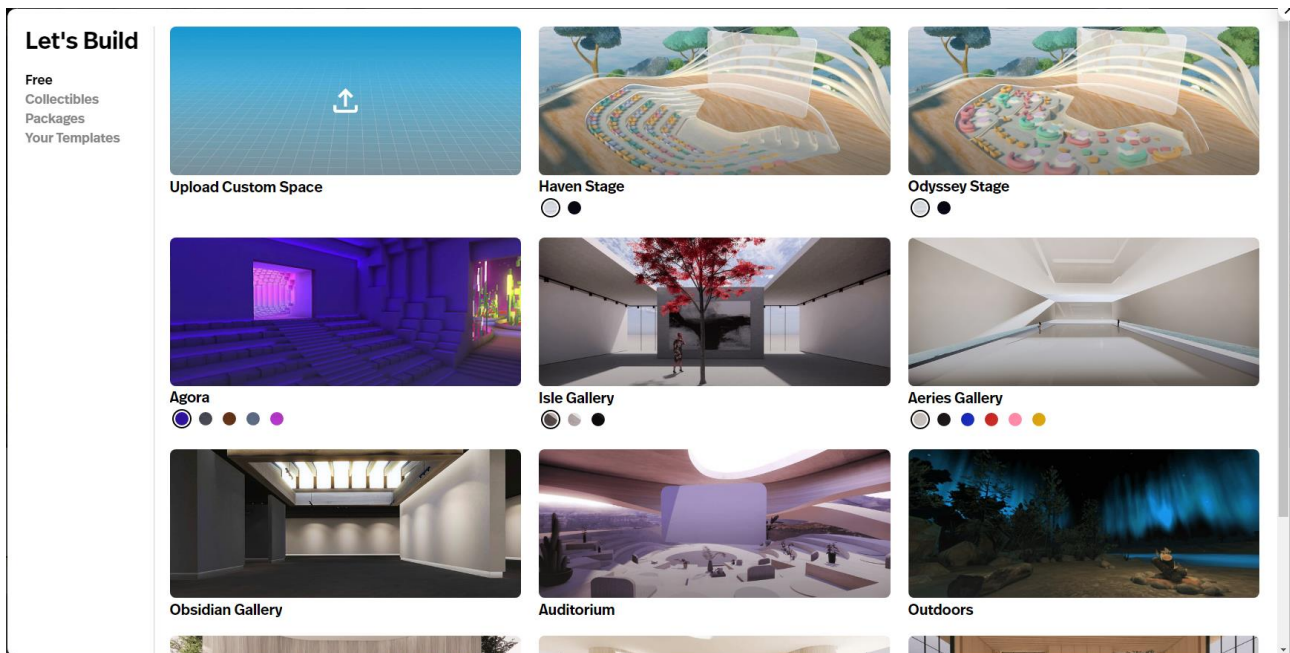
Per creare il proprio spazio, in alto alla destra del nostro avatar è presente il bottone:



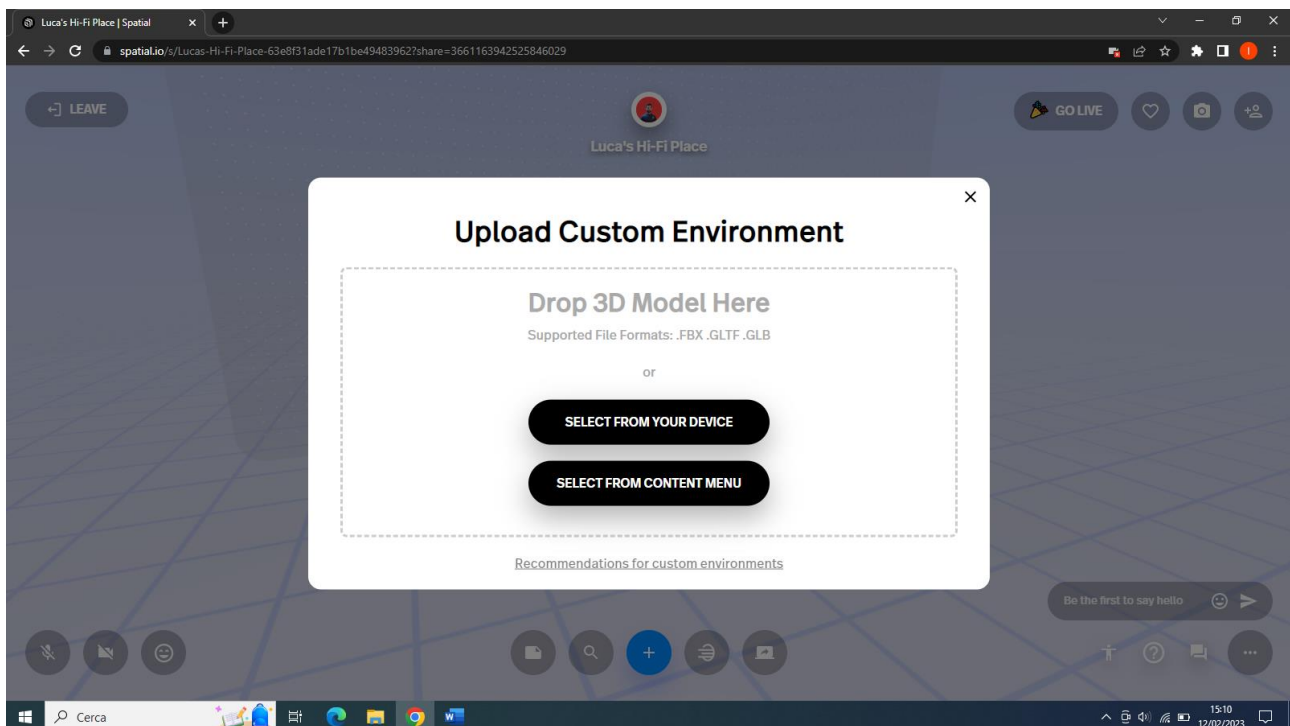
Cliccandoci è possibile scegliere tra differenti opzioni:

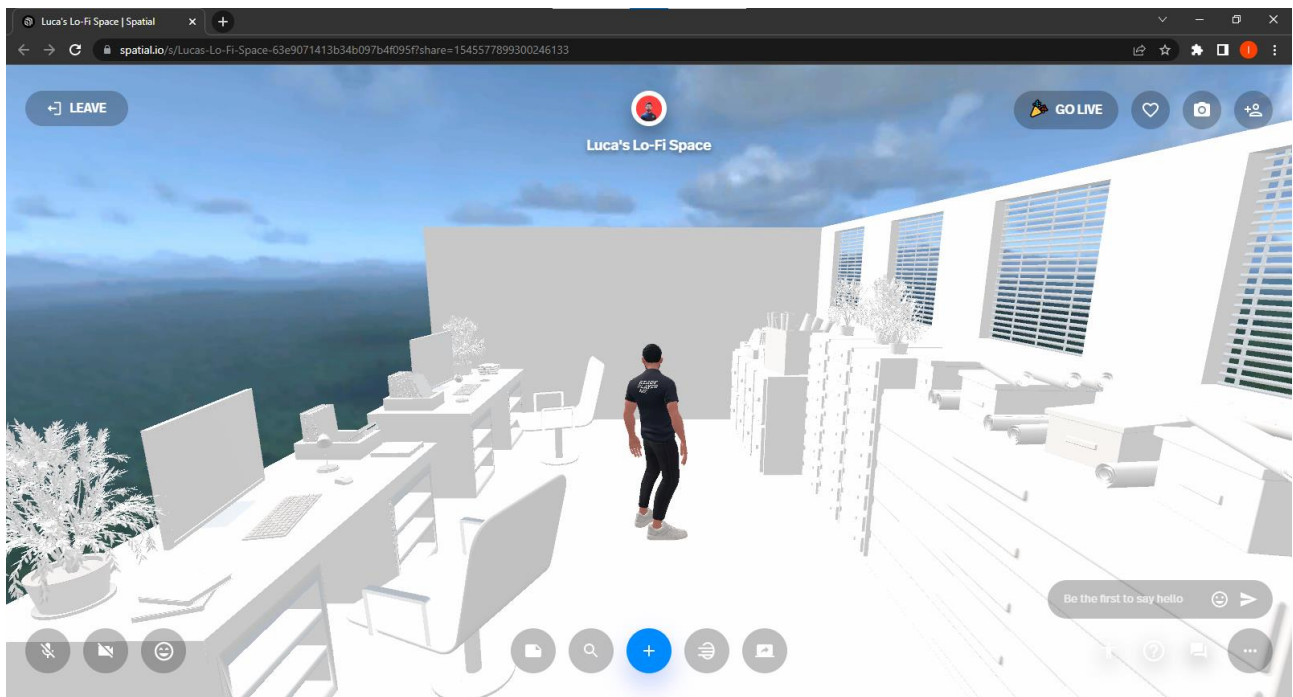
- **Free:** dove è possibile creare una scena personalizzata oppure utilizzare delle scene gratuite offerte dalla piattaforma.
- **Collectibles:** dove è possibile utilizzare delle scene create da altri, e per il cui utilizzo occorre pagare il creatore.
- **Packages:** dove è possibile accedere alle scene create attraverso il toolkit Spatial-SDK.

- **Your Templates:** dove è possibile accedere alla propria raccolta di Template ovvero tutte quelle scene che salveremo e in cui cambieremo solo il contenuto (immagini, video, ...).



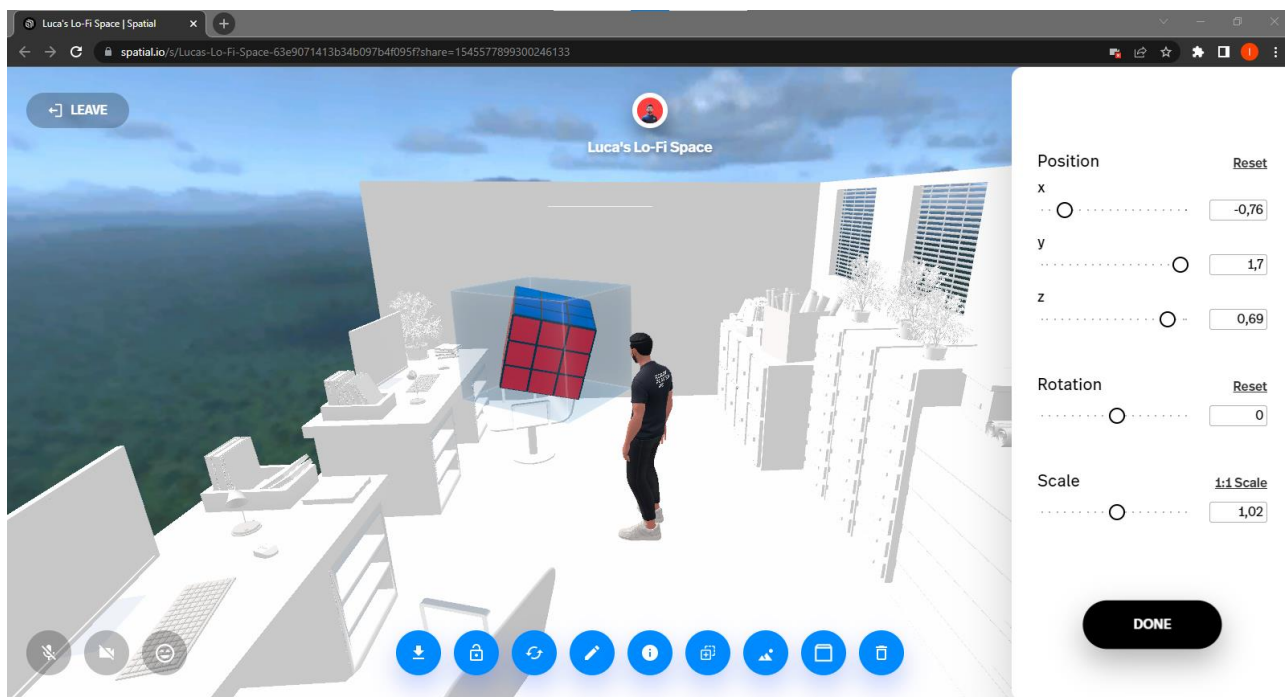
Selezionando **Upload Custom Space**, sarà possibile caricare in locale o dal menu, un modello 3D che fungerà da ambiente.





In particolare, in basso al centro abbiamo rispettivamente i bottoni:

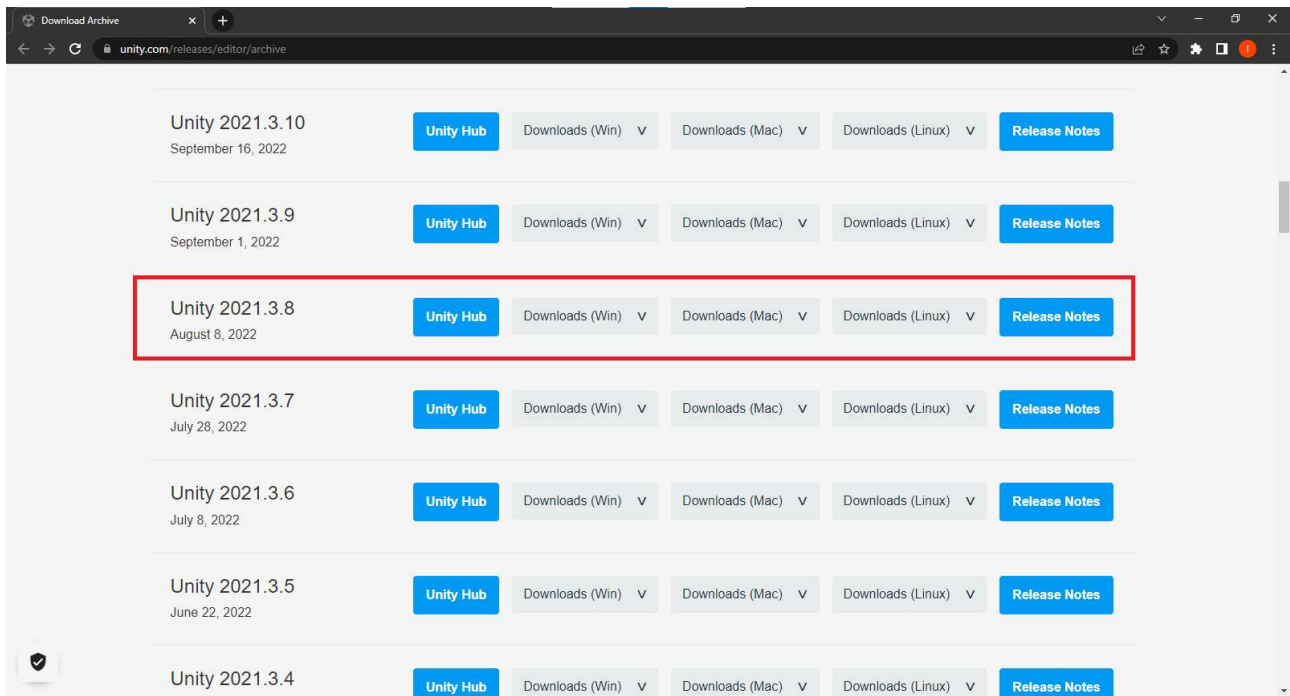
- **Sticky note:** per inserire un appunto testuale nella scena ;
- **Search or URL:** attraverso cui è possibile aprire una pagina web o cercare un contenuto e inserire la finestra nella scena, spostandola a piacimento;
- **Add content:** che permette il caricamento di contenuti multimediali , come ad esempio immagini o video, oggetti 3D(con o senza animazioni), e documenti;
Esso permette anche l'apertura di un menù dove è possibile scegliere tra differenti opzioni per popolare la scena come oggetti 3D precedentemente inseriti, oggetti presenti sulla piattaforma Sketchfab e altro ancora...;
- **Add portal:** in cui è possibile creare portali per altri spazi;
- **Share-screen:** dove è possibile condividere una finestra di un'applicazione del computer e posizionarla all'interno della scena.



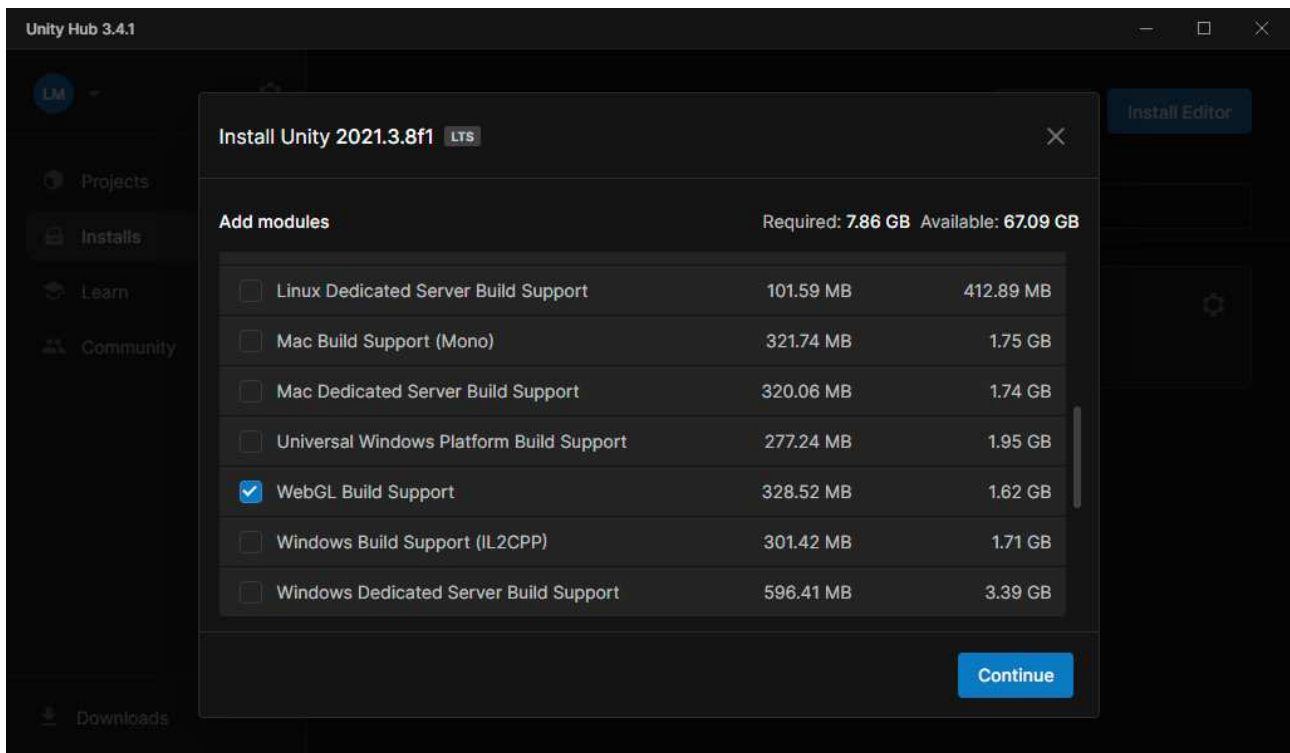
Dopo aver completato il proprio spazio, sarà possibile attraverso gli appositi bottoni rendere pubblica la propria scena sulla piattaforma oppure renderla accessibile solo a coloro che posseggono il link di accesso.

2.7.2 Creazione di una scena (Spatial-SDK)

Innanzitutto, occorre installare Unity, in particolare la versione 2021.3.8f1 reperibile al sito: <https://unity.com/releases/editor/archive>



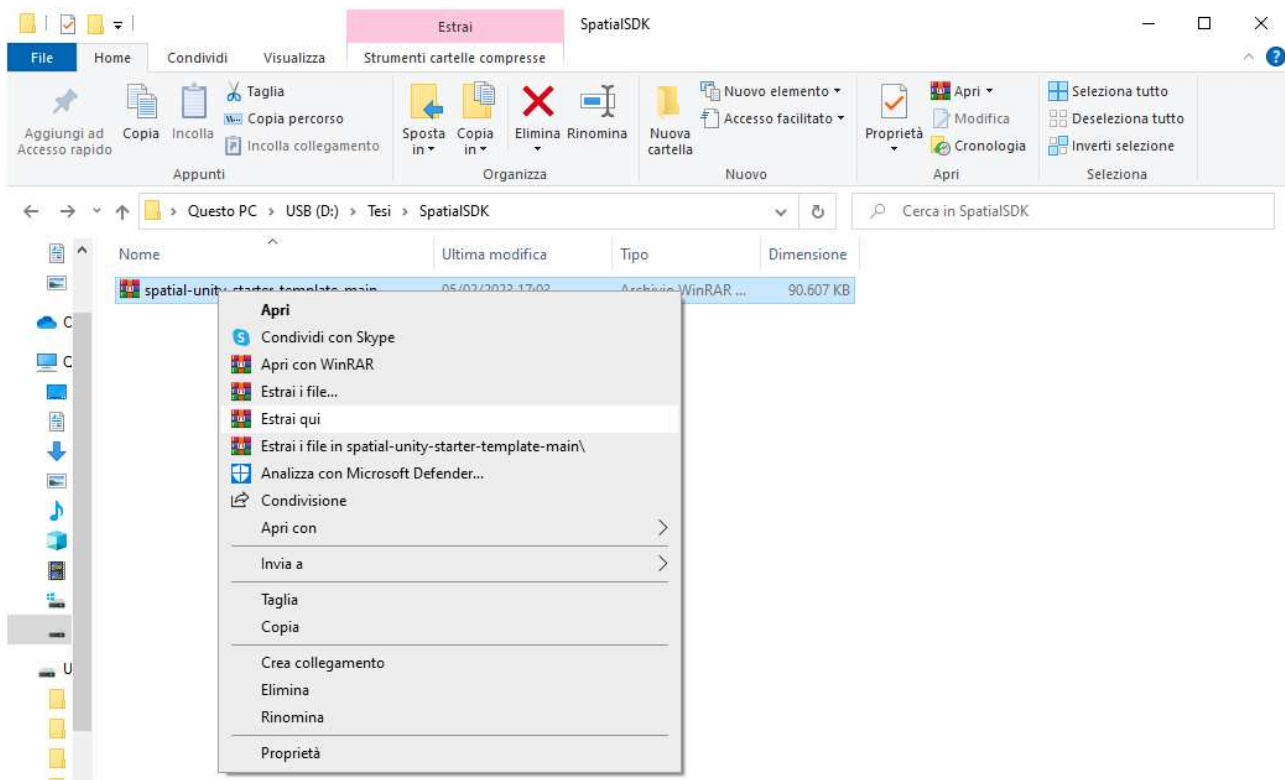
Inoltre, è necessario selezionare il modulo “WebGL Build Support”.

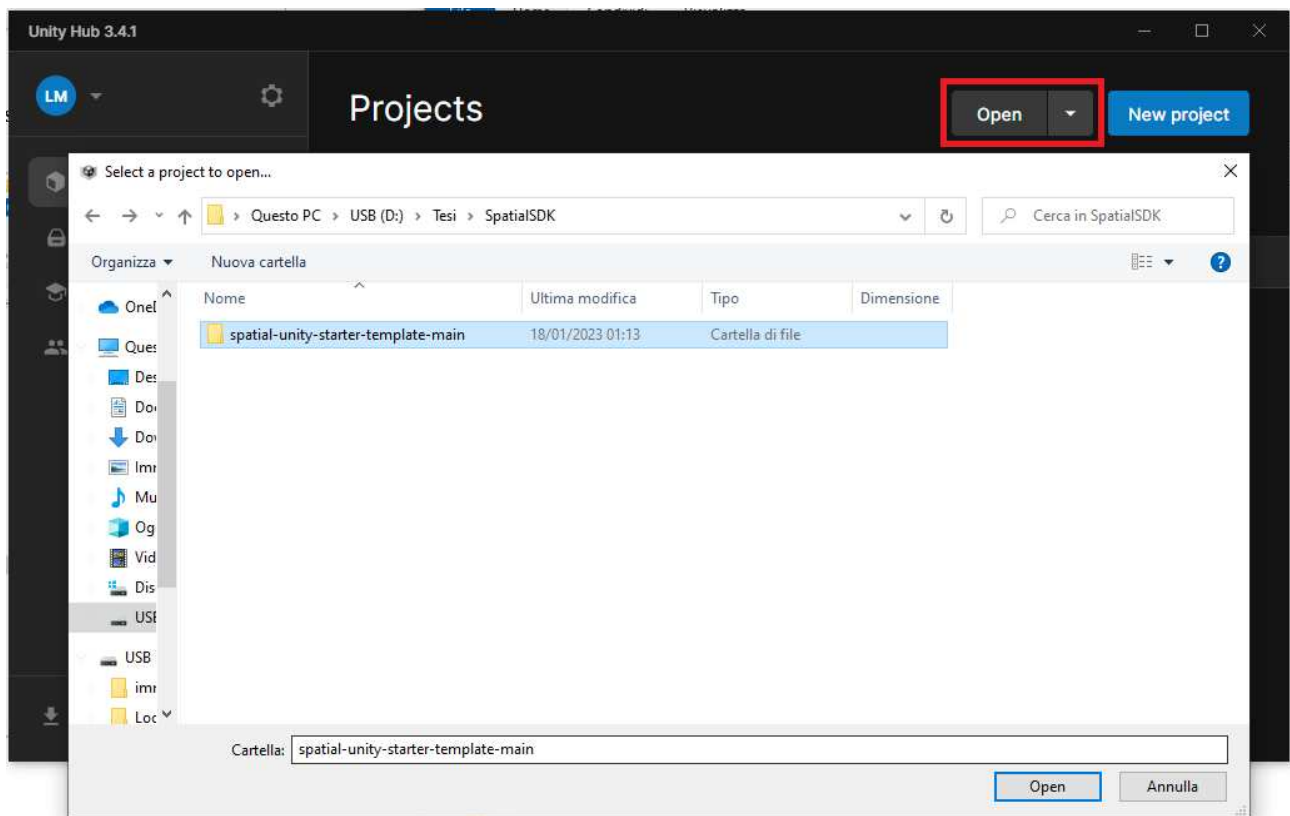


Poi successivamente eseguire il download del template iniziale al link:

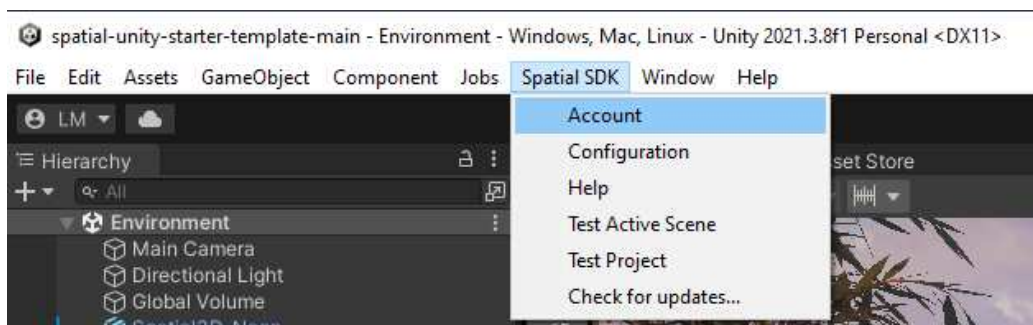
<https://github.com/spatialsys/spatial-unity-starter-template/archive/refs/heads/main.zip>

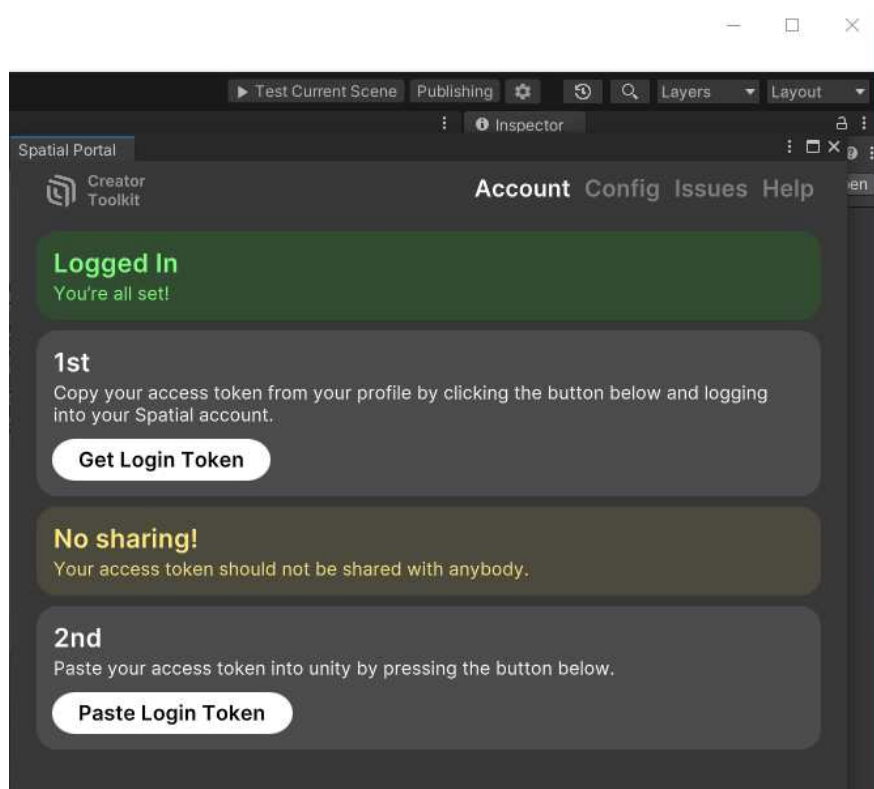
Dall'archivio estrarre la cartella “spatial-unity-starter-template-main” ed aprirla con Unity Hub.





Ora nel menu bisogna posizionarsi su **Spatial-SDK > Account** ed autenticarsi con il proprio account Spatial seguendo le istruzioni a schermo.





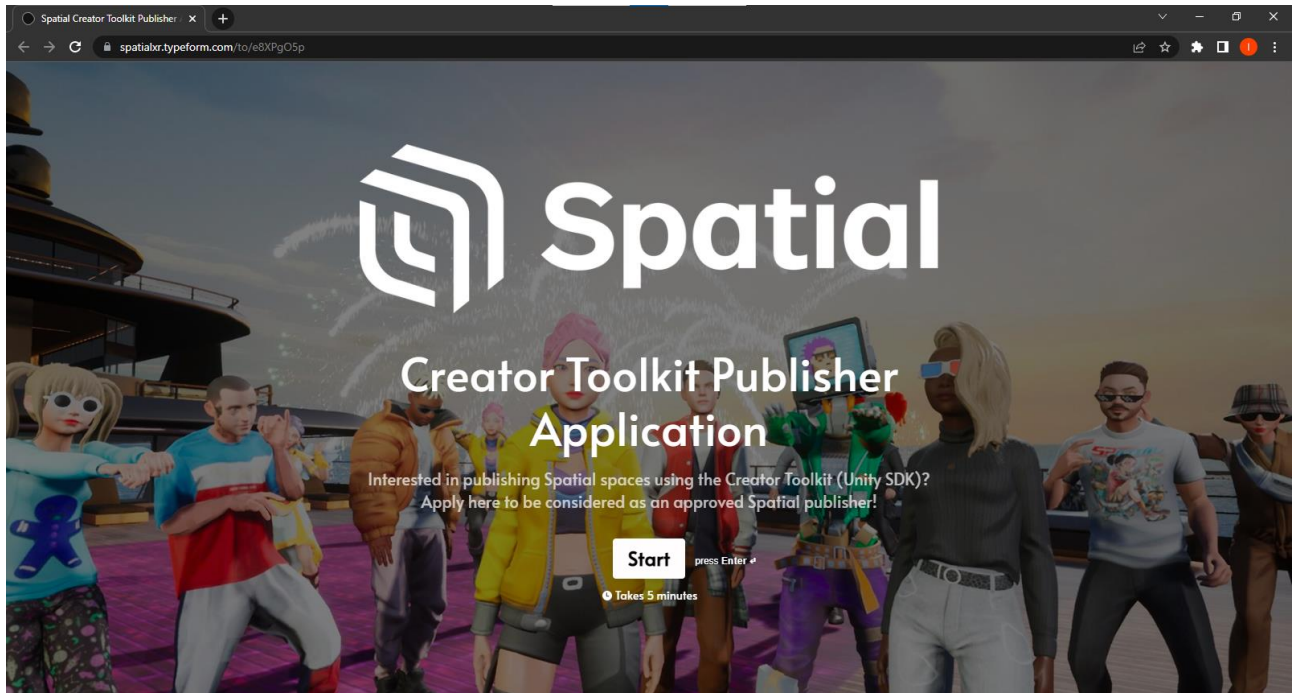
Ora, cliccando sul pulsante **Test Current Scene** sarà possibile visionare un'anteprima su una pagina del browser.



Dopo aver visionato l'anteprima, è possibile caricare l'ambiente su Spatial, sottoforma di package.

Tuttavia, essendo questa funzionalità ancora in Beta, occorre compilare un form e aspettare l'accettazione da parte della piattaforma.

Successivamente, dopo averne ricevuto l'approvazione, se si desidera pubblicare il package, occorre attendere all'incirca 15 minuti.



3 Soluzione proposta

Nel seguente capitolo verranno mostrate due sene: la prima realizzata utilizzando l'ambiente offerto da Decentraland, la seconda utilizzando l'ambiente offerto da Spatial.

3.1 Scena porta a combinazione (Decentraland)

3.1.1 Analisi del problema

Una persona vuole entrare in una stanza, però questa stanza possiede una porta dotata di un sistema di sicurezza per cui è necessaria una combinazione di 4 cifre per entrare.

La combinazione di numeri va da 0 a 9 e l'ordine di inserimento è essenziale per la corretta apertura della porta.

Se la persona non ricorda la combinazione non potrà entrare nella stanza.

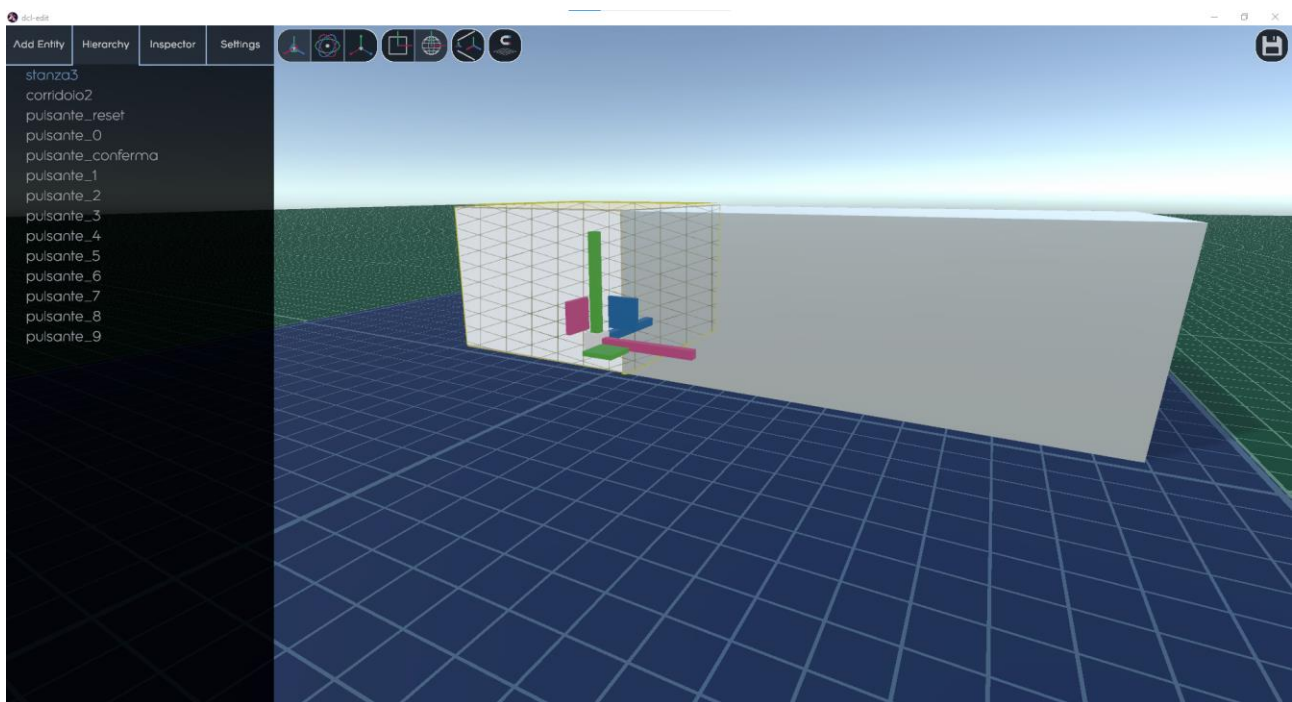
3.1.2 Modellazione della scena

Per la realizzazione di questa scena sono stati utilizzati due modelli presenti sulla piattaforma Sketchfab:

- Ufficio: <https://skfb.ly/6wGM6>
- Pulsantiera: <https://skfb.ly/6GspX>
- Porta: <https://skfb.ly/696o6>
- Il corridoio è stato realizzato modificando modelli di base presenti su Blender.

Inoltre, è stata utilizzato lo strumento Decentraland-Editor per gestire il posizionamento, le dimensioni dei modelli 3D all'interno della scena.

All'interno dei modelli sono stati inseriti dei **collider** ovvero delle forme tridimensionali invisibili che delineano il modello e permettono la collisione con altri oggetti nella scena, ad esempio, alle mura della stanza è stato applicato un collider così da impedire all'utente di passarne attraverso.



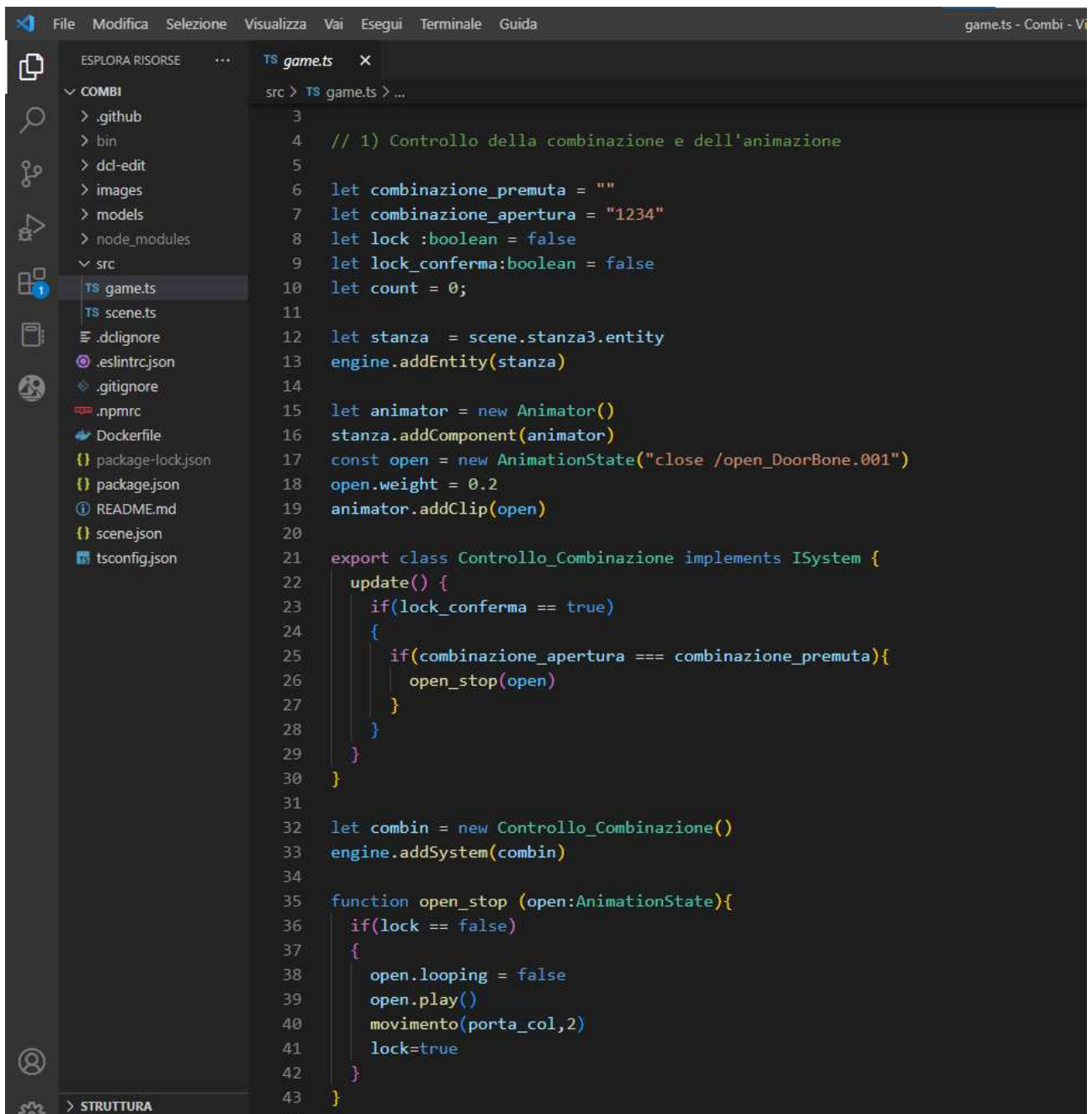
3.1.3 Fase di scripting

Inizialmente sono stati importati due elementi:

- **Ecs-scene-utils**: ossia una libreria che possiede al suo interno una vasta raccolta di componenti, metodi e funzioni messe a disposizione della community per aiutare i creatori nello sviluppo delle proprie scene.
- **scene**: ovvero tutti i modelli creati con il Decentraland-Editor vengono inseriti nella scena.

Per avere una panoramica più chiara del codice presente nella scena, quest'ultimo è stato suddiviso in più parti (ciascuna parte si occupa di gestire una funzionalità della scena):

- 1) Controllo della combinazione e dell'animazione



```
File Modifica Selezione Visualizza Vai Esegui Terminale Guida game.ts - Combi - V...

ESPLORA RISORSE ... TS game.ts x
src > TS game.ts > ...

3
4 // 1) Controllo della combinazione e dell'animazione
5
6 let combinazione_premuta = ""
7 let combinazione_apertura = "1234"
8 let lock :boolean = false
9 let lock_conferma:boolean = false
10 let count = 0;
11
12 let stanza = scene.stanza3.entity
13 engine.addEntity(stanza)
14
15 let animator = new Animator()
16 stanza.addComponent(animator)
17 const open = new AnimationState("close /open_DoorBone.001")
18 open.weight = 0.2
19 animator.addClip(open)
20
21 export class Controllo_Combinazione implements ISystem {
22     update() {
23         if(lock_conferma == true)
24         {
25             if(combinazione_apertura === combinazione_premuta){
26                 open_stop(open)
27             }
28         }
29     }
30 }
31
32 let combin = new Controllo_Combinazione()
33 engine.addSystem(combin)
34
35 function open_stop (open:AnimationState){
36     if(lock == false)
37     {
38         open.looping = false
39         open.play()
40         movimento(porta_col,2)
41         lock=true
42     }
43 }
```

Le variabili sono rispettivamente:

- **combinazione_premuta**: che conserva la combinazione inserita dall'utente, inizialmente è vuota;
- **combinazione_apertura**: che conserva la combinazione per la corretta apertura della porta;
- **lock**: che impedisce di eseguire in loop l'animazione dell'apertura della porta;

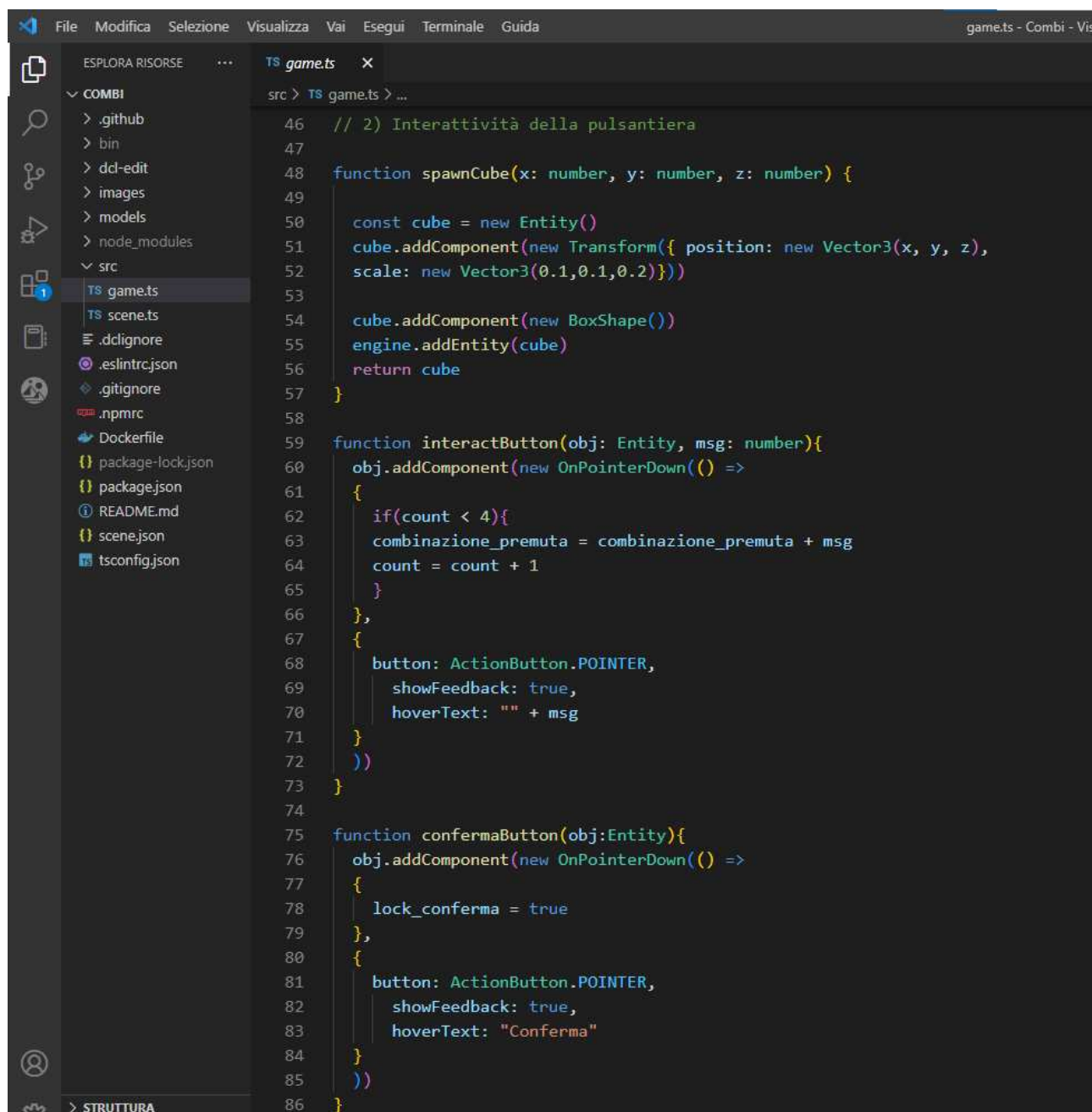
- **lock_conferma**: che permette alla componente **Controllo_Combinazione** di eseguire il controllo solo quando l'utente ha confermato la propria combinazione;
- **count**: che conta quanti numeri sono stati cliccati dall'utente sulla pulsantiera.

Successivamente è stata aggiunta la stanza e creata una componente per la gestione dell'animazione della porta.

La componente di sistema **Controllo_Combinazione** si occupa di controllare se la combinazione inserita dall'utente attraverso i bottoni della pulsantiera corrisponde alla combinazione impostata per l'apertura, dopodiché se la combinazione risulta corretta la porta si aprirà e l'utente potrà entrare nella stanza.

La funzione **open_stop** si occupa di gestire l'animazione della porta evitando che l'animazione si ripeta in loop e dello spostamento del collider quando l'utente inserisce la combinazione corretta.

2) Interattività della pulsantiera

A screenshot of a code editor with a dark theme. The left sidebar shows a file explorer with a project structure including folders like .github, bin, dcl-edit, images, models, node_modules, and src. The src folder is expanded, showing files like TS game.ts, TS scene.ts, .dclignore, .eslintrc.json, .gitignore, .npmrc, Dockerfile, package-lock.json, package.json, README.md, scene.json, and tsconfig.json. The main editor area displays the content of TS game.ts. The code defines two functions: spawnCube and interactButton. spawnCube creates a new Entity with a Transform component (position and scale) and a BoxShape component, then adds it to the engine and returns it. interactButton adds an OnPointerDown component to an entity, which checks a count and updates a combination variable. A third function, confermaButton, is also shown, adding another OnPointerDown component that sets a lock variable and has a specific hoverText.

```
46 // 2) Interattività della pulsantiera
47
48 function spawnCube(x: number, y: number, z: number) {
49
50     const cube = new Entity()
51     cube.addComponent(new Transform({ position: new Vector3(x, y, z),
52     scale: new Vector3(0.1,0.1,0.2)}))
53
54     cube.addComponent(new BoxShape())
55     engine.addEntity(cube)
56     return cube
57 }
58
59 function interactButton(obj: Entity, msg: number){
60     obj.addComponent(new OnPointerDown(() =>
61     {
62         if(count < 4){
63             combinazione_premuta = combinazione_premuta + msg
64             count = count + 1
65         }
66     },
67     {
68         button: ActionButton.POINTER,
69         showFeedback: true,
70         hoverText: "" + msg
71     }
72     ))
73 }
74
75 function confermaButton(obj:Entity){
76     obj.addComponent(new OnPointerDown(() =>
77     {
78         lock_conferma = true
79     },
80     {
81         button: ActionButton.POINTER,
82         showFeedback: true,
83         hoverText: "Conferma"
84     }
85     ))
86 }
```

La funzione **spawnCube** crea un cubo all'interno della scena alle coordinate prese in input, quest'ultima è stata utilizzata nella creazione del display.

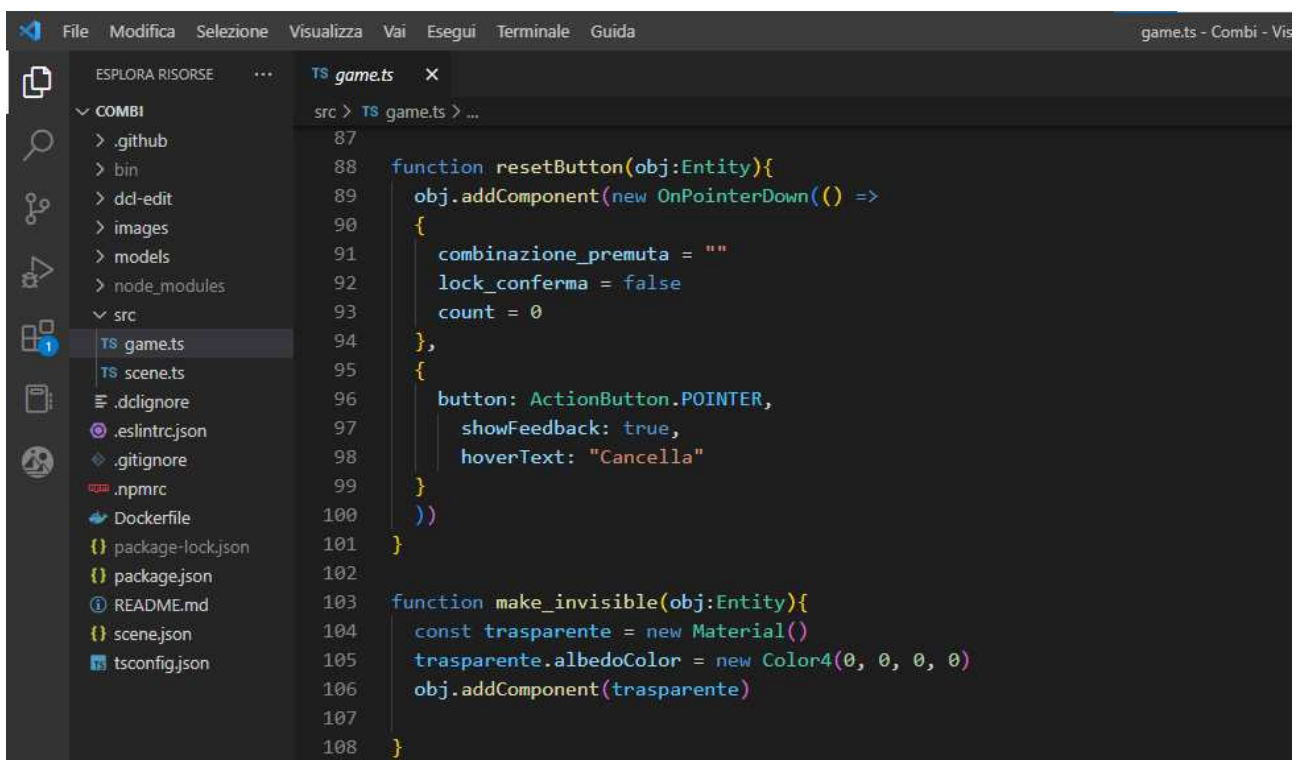
La funzione **interactButton** si occupa di rendere un oggetto passato in input un pulsante interattivo.

In particolare, se l'utente inserisce:

- meno di quattro numeri: il prossimo numero cliccato dall'utente si aggiungerà alla combinazione.
- più di quattro numeri: il prossimo numero cliccato non verrà inserito nella combinazione.

La funzione **confermaButton** si occupa di rendere un oggetto passato in input un pulsante di conferma.

In particolare, la variabile `lock_conferma` viene impostata a `true` così da permettere alla componente di sistema `Controllo_Combinazione` di eseguire il confronto tra la combinazione inserita dall'utente e la combinazione per l'apertura.



```

87
88 function resetButton(obj:Entity){
89     obj.addComponent(new OnPointerDown(() =>
90     {
91         combinazione_premuta = ""
92         lock_conferma = false
93         count = 0
94     },
95     {
96         button: ActionButton.POINTER,
97         showFeedback: true,
98         hoverText: "Cancella"
99     })
100 }
101
102
103 function make_invisible(obj:Entity){
104     const trasparente = new Material()
105     trasparente.albedoColor = new Color4(0, 0, 0, 0)
106     obj.addComponent(trasparente)
107
108 }
  
```

La funzione **resetButton** si occupa di rendere un oggetto passato in input un pulsante di ripristino, così da permettere l'inserimento di una nuova combinazione all'utente.

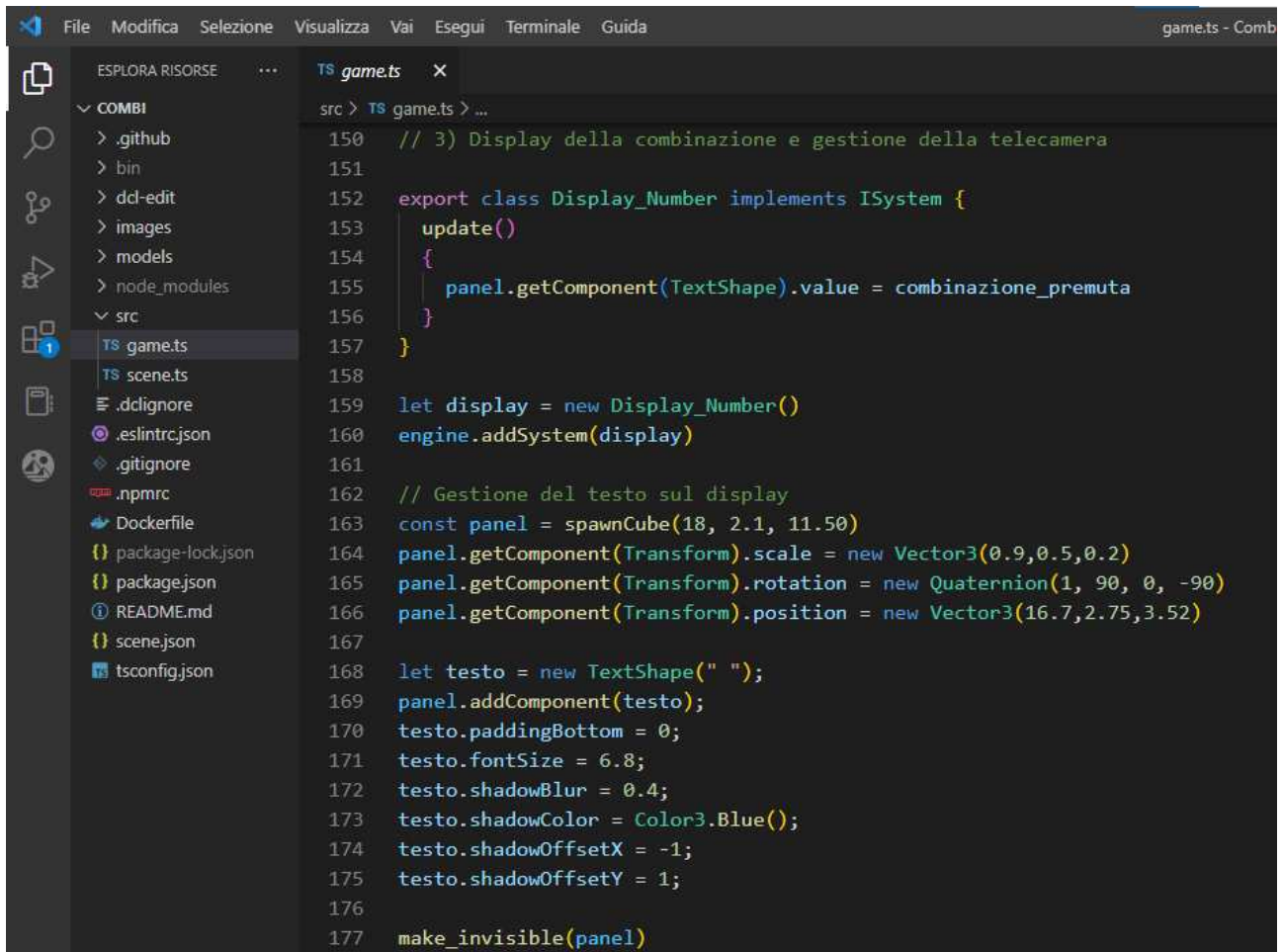
Essa, nello specifico, azzera la combinazione dell'utente inserita fino a quel momento così da permettere un nuovo inserimento.

```
110
111 const pulsante_1 = scene.pulsante1.entity
112 const pulsante_2 = scene.pulsante2.entity
113 const pulsante_3 = scene.pulsante3.entity
114 const pulsante_4 = scene.pulsante4.entity
115 const pulsante_5 = scene.pulsante5.entity
116 const pulsante_6 = scene.pulsante6.entity
117 const pulsante_7 = scene.pulsante7.entity
118 const pulsante_8 = scene.pulsante8.entity
119 const pulsante_9 = scene.pulsante9.entity
120 const pulsante_Conferma = scene.pulsanteconferma.entity
121 const pulsante_0 = scene.pulsante0.entity
122 const pulsante_Reset = scene.pulsantereset.entity
123
124 interactButton(pulsante_1,1)
125 interactButton(pulsante_2,2)
126 interactButton(pulsante_3,3)
127 interactButton(pulsante_4,4)
128 interactButton(pulsante_5,5)
129 interactButton(pulsante_6,6)
130 interactButton(pulsante_7,7)
131 interactButton(pulsante_8,8)
132 interactButton(pulsante_9,9)
133 resetButton(pulsante_Reset)
134 interactButton(pulsante_0,0)
135 confermaButton(pulsante_Conferma)
136
137 make_invisible(pulsante_1)
138 make_invisible(pulsante_2)
139 make_invisible(pulsante_3)
140 make_invisible(pulsante_4)
141 make_invisible(pulsante_5)
142 make_invisible(pulsante_6)
143 make_invisible(pulsante_7)
144 make_invisible(pulsante_8)
145 make_invisible(pulsante_9)
146 make_invisible(pulsante_Conferma)
147 make_invisible(pulsante_0)
148 make_invisible(pulsante_Reset)
149
```

Nelle seguenti linee di codice, sono stati importati i modelli dei pulsanti dalla scena creata con dcl-edit.

Dopodiché, i pulsanti sono stati resi interattivi con la funzione `interactButton` e resi invisibili all'utente con la funzione `make_invisible`.

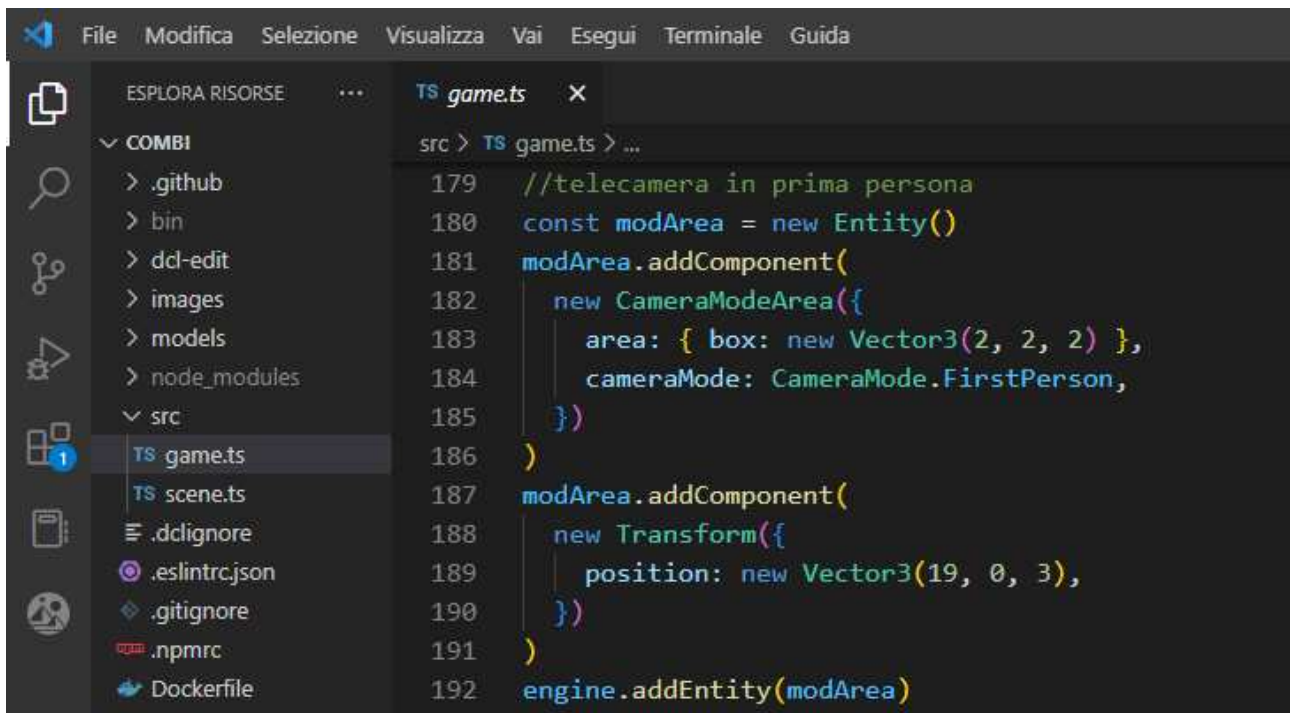
3) Display della combinazione e gestione della telecamera



```
File Modifica Selezione Visualizza Vai Esegui Terminale Guida game.ts - Comb  
ESPLORA RISORSE  
COMBI  
  > .github  
  > bin  
  > dcl-edit  
  > images  
  > models  
  > node_modules  
  > src  
    TS game.ts  
    TS scene.ts  
  .dclignore  
  .eslintrc.json  
  .gitignore  
  .npmrc  
  Dockerfile  
  package-lock.json  
  package.json  
  README.md  
  scene.json  
  tsconfig.json  
src > TS game.ts > ...  
150 // 3) Display della combinazione e gestione della telecamera  
151  
152 export class Display_Number implements ISystem {  
153   update()  
154   {  
155     panel.getComponent(TextShape).value = combinazione_premuta  
156   }  
157 }  
158  
159 let display = new Display_Number()  
160 engine.addSystem(display)  
161  
162 // Gestione del testo sul display  
163 const panel = spawnCube(18, 2.1, 11.50)  
164 panel.getComponent(Transform).scale = new Vector3(0.9,0.5,0.2)  
165 panel.getComponent(Transform).rotation = new Quaternion(1, 90, 0, -90)  
166 panel.getComponent(Transform).position = new Vector3(16.7,2.75,3.52)  
167  
168 let testo = new TextShape(" ");  
169 panel.addComponent(testo);  
170 testo.paddingBottom = 0;  
171 testo.fontSize = 6.8;  
172 testo.shadowBlur = 0.4;  
173 testo.shadowColor = Color3.Blue();  
174 testo.shadowOffsetX = -1;  
175 testo.shadowOffsetY = 1;  
176  
177 make_invisible(panel)
```

La componente di Sistema **Display_Number** si occupa di visualizzare sul display nella scena la combinazione premuta dall'utente.

La gestione del testo si occupa di posizionare correttamente il testo all'interno del display e di modificare differenti impostazioni sul testo come, ad esempio, dimensioni del font o colore del testo.



Nella scena, viene creata anche una piccola area invisibile all'utente.

Entrando in questa area, la telecamera dell'utente passerà automaticamente nella modalità in prima persona.

Quest'area è stata inserita vicino alla pulsantiera, così da permettere una maggiore precisione quando l'utente interagisce con i pulsanti.

4) Gestione della fisicità della porta all'apertura

```
195 //4) Gestione della fisicità della porta all'apertura
196
197 function movimento(obj: Entity, time: number){
198
199     let start = new Vector3(16.7, 2, 6.2)
200     let end = new Vector3(15.7, 2, 5.40)
201
202     let start_r = Quaternion.Euler(0, 0, 0)
203     let end_r = Quaternion.Euler(90, 0, 90)
204
205     obj.addComponent(new utils.MoveTransformComponent(start, end, time))
206     obj.addComponent(new utils.RotateTransformComponent(start_r, end_r, 0.7))
207 }
208
209
210 //collider per la porta
211 let porta_col = new Entity()
212 porta_col.addComponent(
213     new Transform({
214         position: new Vector3(16.7, 2, 6.2),
215         scale: new Vector3(0.2, 1.7, 1.8)
216     })
217 )
218
219 let box_porta = new BoxShape()
220
221 box_porta.withCollisions = true
222 porta_col.addComponent(box_porta)
223 make_invisible(porta_col)
224 engine.addEntity(porta_col)
```

La porta, a differenza degli altri modelli, non possiede un collider fisso in quanto se l'utente inserisse la combinazione giusta la porta si aprirebbe cambiando posizione all'interno della scena ma il collider resterebbe sempre lì e non permetterebbe l'accesso all'utente.

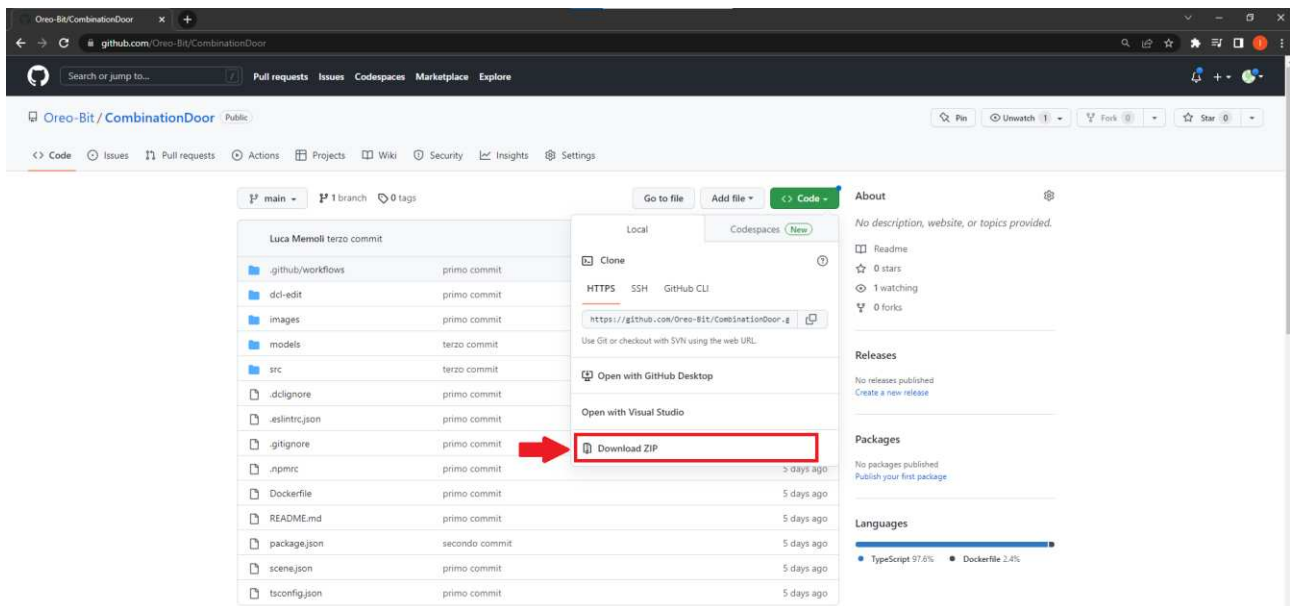
Per cui, è stata creata la funzione movimento che si occupa del coordinamento tra l'animazione dell'apertura della porta e il movimento del suo collider.

3.1.4 Risultato finale

Dopo aver analizzato le caratteristiche dello script presente nella scena, sarà possibile visualizzare un'anteprima del risultato finale.

Prima di tutto, occorre scaricare il progetto da GitHub:

<https://github.com/Oreo-Bit/CombinationDoor>



Successivamente, estraiamo il contenuto del pacchetto CombinationDoor-main.zip in una cartella, apriamo il terminale ed eseguiamo il comando: **dcl start**

```
Seleziona C:\Windows\system32\cmd.exe
Microsoft Windows [Versione 10.0.19045.2604]
(c) Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Luca\Desktop\test\CombinationDoor-main>dcl start
✓ Latest SDK installation found.
✓ Checking decentraland libraries

> dcl-project@1.0.0 watch
> build-ecs --watch

> dev mode: true
> working directory: C:\Users\Luca\Desktop\test\CombinationDoor-main

> processing C:/Users/Luca/Desktop/test/CombinationDoor-main/src/game.ts
> bundling:
- node_modules@dcl\amd\dist\amd.js
- node_modules\decentraland-ecs\dist\src\index.js
- node_modules@dcl\ecs-scene-utils\dist\index.js
- src\game.ts
> writing C:/Users/Luca/Desktop/test/CombinationDoor-main/bin/game.js
> writing C:/Users/Luca/Desktop/test/CombinationDoor-main/bin/game.js.lib

The compiler is watching file changes...

<<< Initializing components >>>
<<< Wiring app >>>
<<< Starting components >>>
2023-02-27T21:07:21.849Z [LOG] (http-server): Listening 0.0.0.0:8000

Preview server is now running

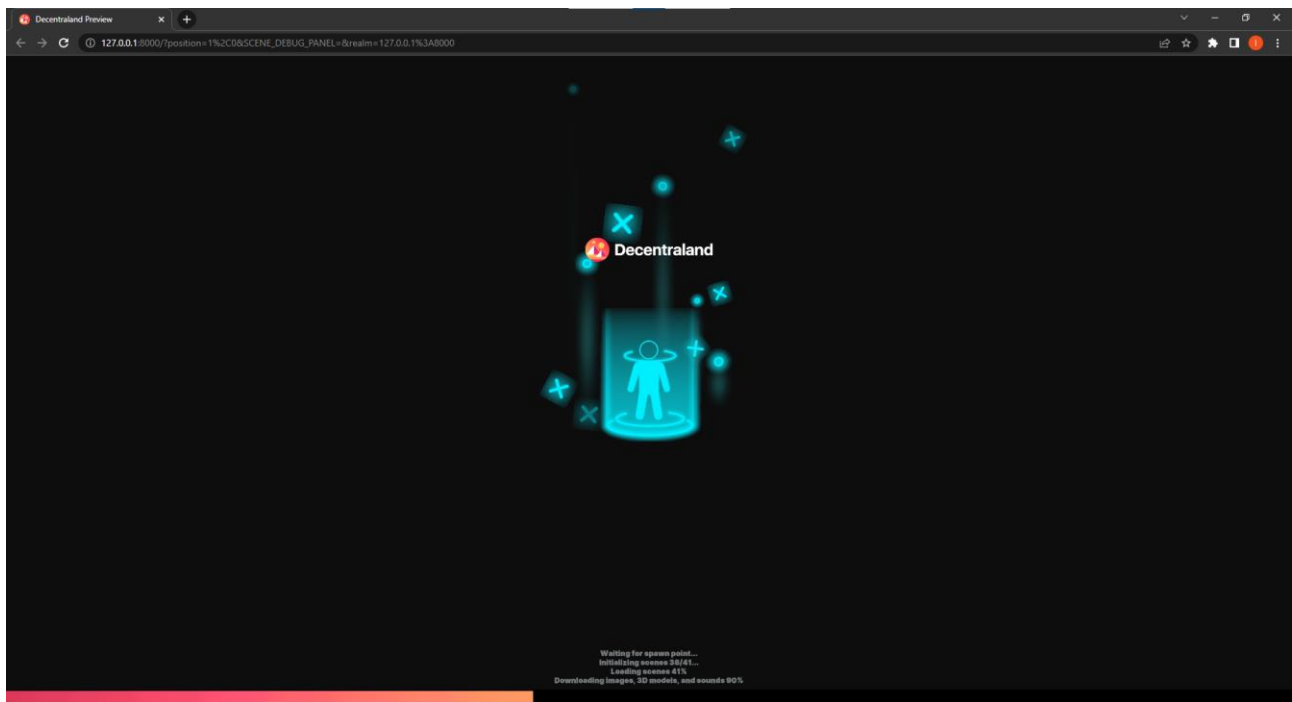
Available on:

  http://127.0.0.1:8000?position=0%2C0&SCENE_DEBUG_PANEL
  http://169.254.85.118:8000?position=0%2C0&SCENE_DEBUG_PANEL
  http://192.168.1.4:8000?position=0%2C0&SCENE_DEBUG_PANEL

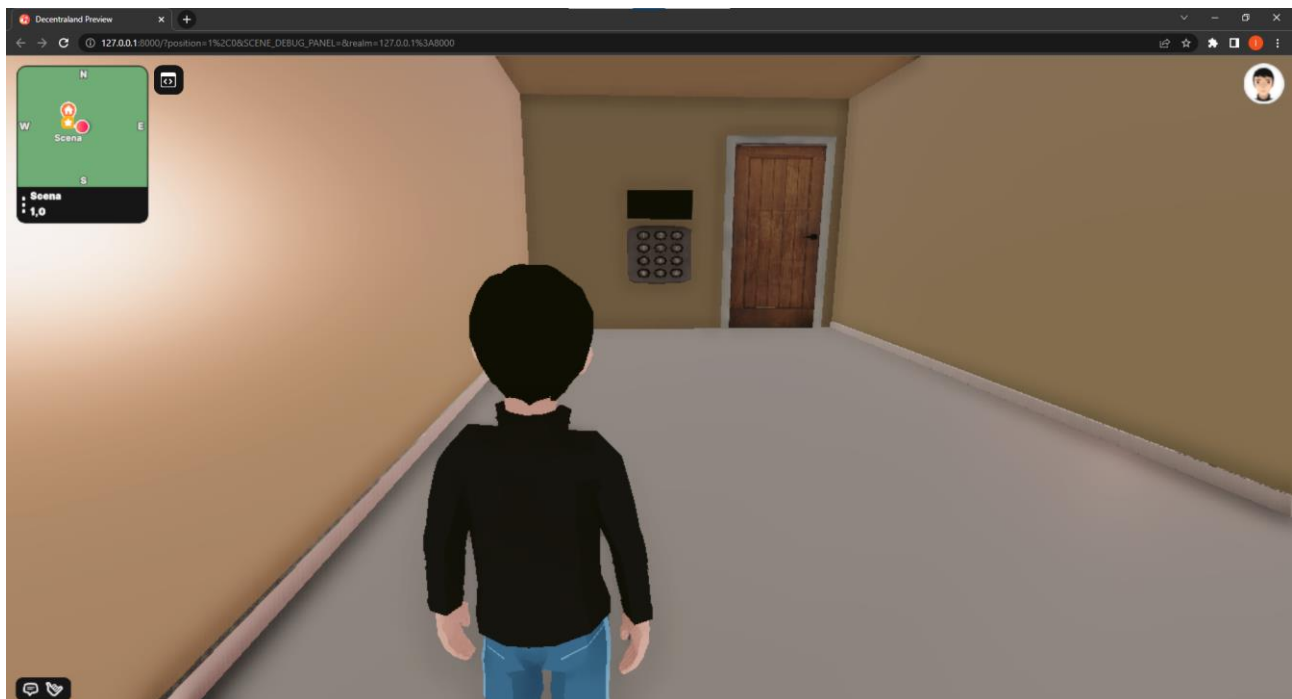
Details:

Press CTRL+C to exit
```

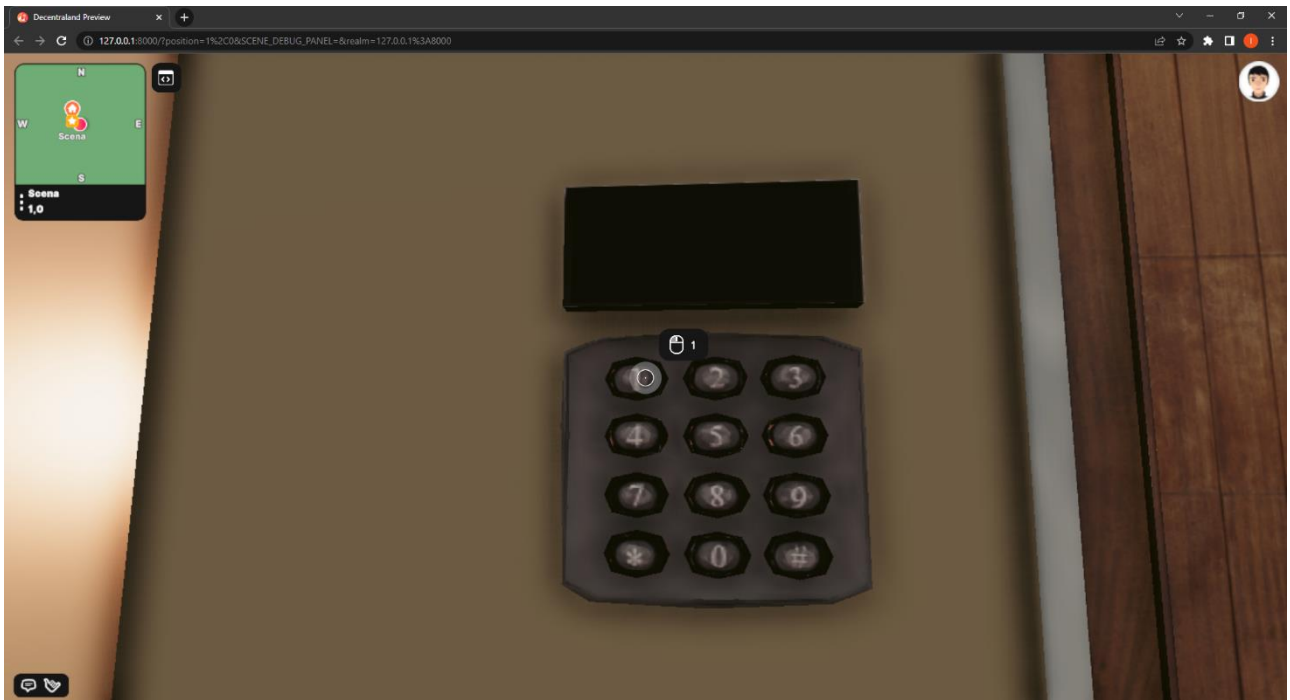
Si aprirà una finestra sul nostro browser predefinito e occorrerà attendere il corretto caricamento della scena.



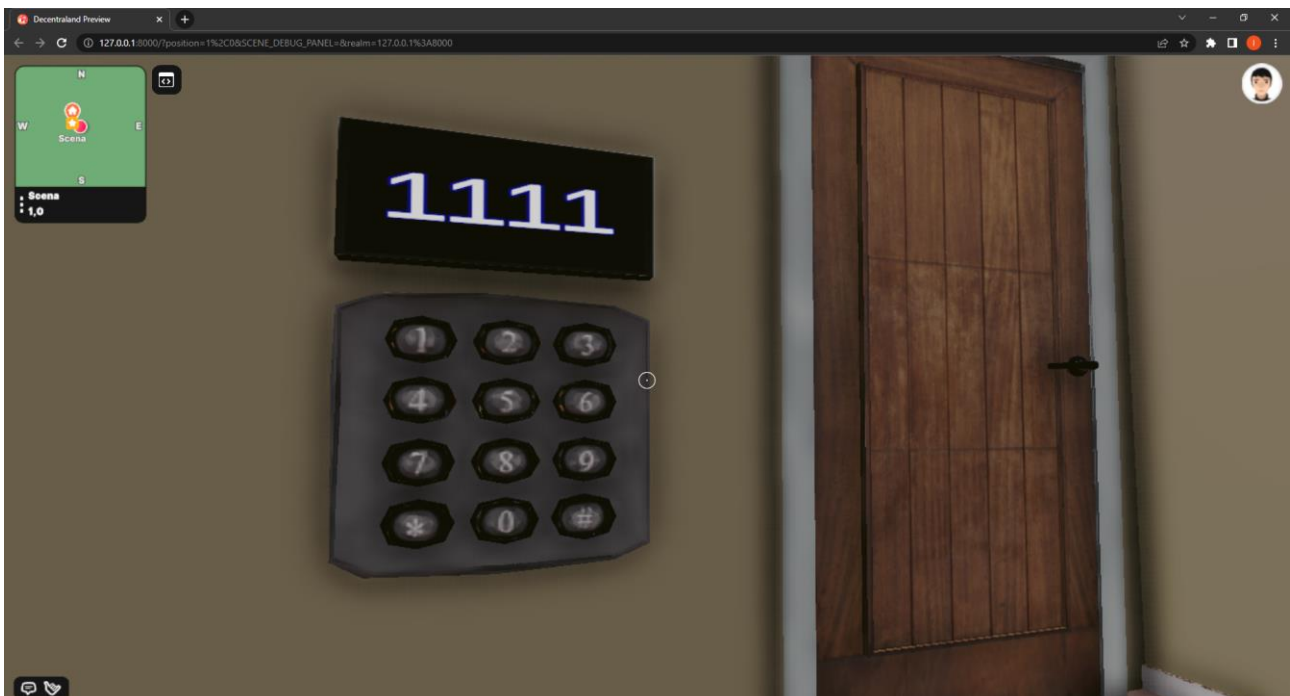
Al termine, l'utente si troverà nel corridoio e davanti a lui sarà presente la porta chiusa.



In prossimità della pulsantiera, la telecamera passerà automaticamente da terza a prima persona.



Se l'utente inserisce una combinazione errata, la porta resterà chiusa e cliccando il tasto reset sarà possibile inserire nuovamente la combinazione e riprovare.



Se l'utente, invece, inserisce la combinazione corretta, la porta si aprirà e quest'ultimo potrà accedere alla stanza chiusa.



Inoltre, attraverso la piattaforma Render.com è stato possibile eseguire il caricamento e il deploy della scena.

Per visionare l'anteprima, occorre collegarsi al seguente link da browser desktop:

<https://combinationdoor.onrender.com/>

3.2 Scena Spatial

3.2.1 Analisi del problema

Un'insegnante desidera mostrare alla propria classe alcune delle funzionalità che il metaverso ha da offrire.

Così decide di svolgere la lezione del giorno in un'aula virtuale, creata sulla piattaforma Spatial, in cui sarà possibile mostrare il materiale didattico ai propri studenti.

3.2.2 Modellazione della scena

Nella scena sono stati importati i seguenti modelli:

- aula: <https://skfb.ly/ozNVM>
- albero: <https://skfb.ly/6tQ6l>

Tuttavia, sono stati importati anche altri modelli e texture con lo scopo di rendere più realistica la scena.

In particolare, sono stati modificati alcuni elementi dei modelli, come ad esempio, i colori oppure la forma, al fine di renderli più coerenti con il contesto scolastico.

Tutte le modifiche ai modelli sono state effettuate utilizzando l'editor di Unity.

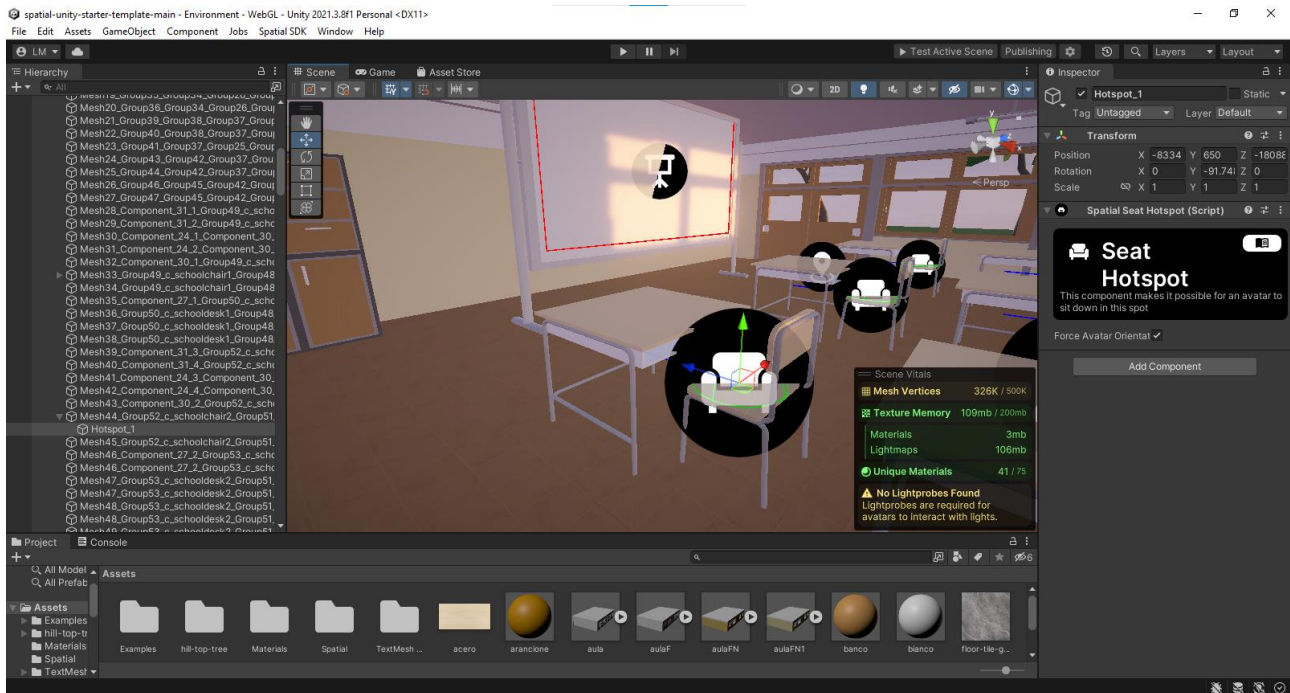
3.2.3 Fase di sviluppo

Dopo aver posizionato correttamente i vari modelli con modifiche di elementi e texture all'interno dell'ambiente, sono state utilizzate delle componenti presenti nel plugin Spatial SDK:

- **Spatial Entrance Point:** questa componente specifica l'area di spawn degli avatar degli utenti all'interno della scena;



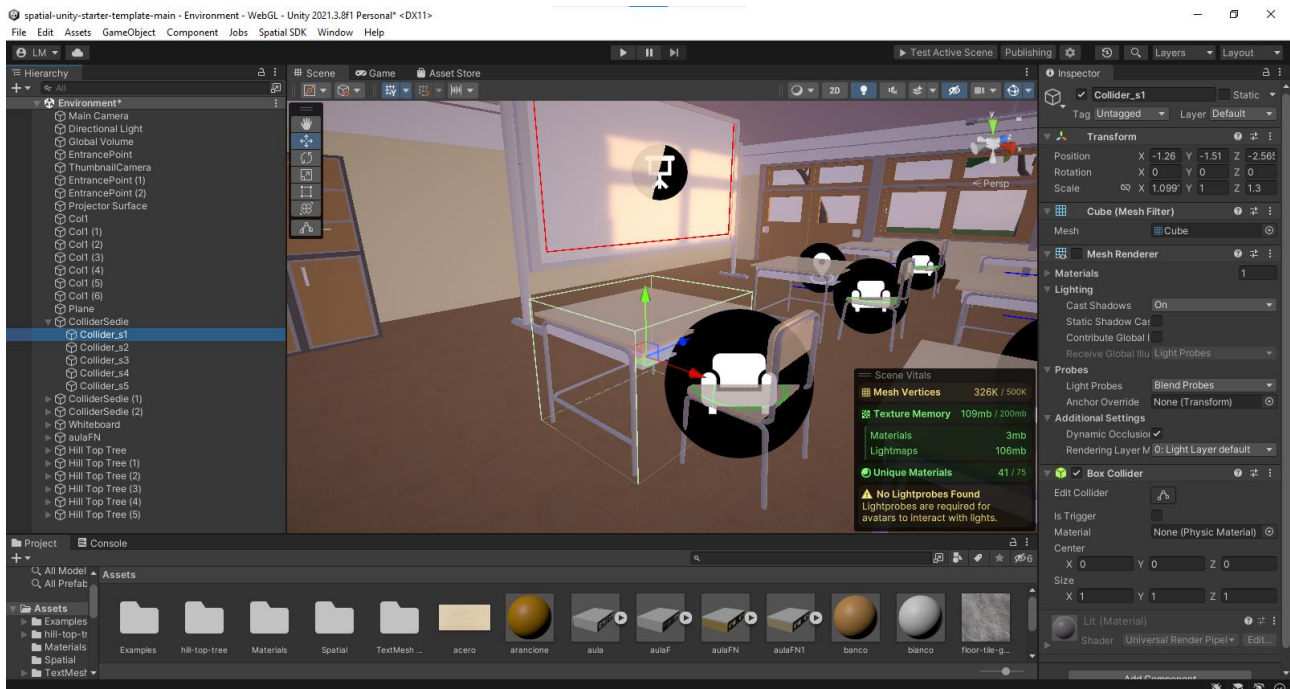
- **Spatial Seat Hotspot:** questa componente permette agli avatar di sedersi;



- **Spatial Projector Surface:** questa componente permette di gestire la posizione del riquadro nel momento in cui l'admin eseguirà lo share screen.



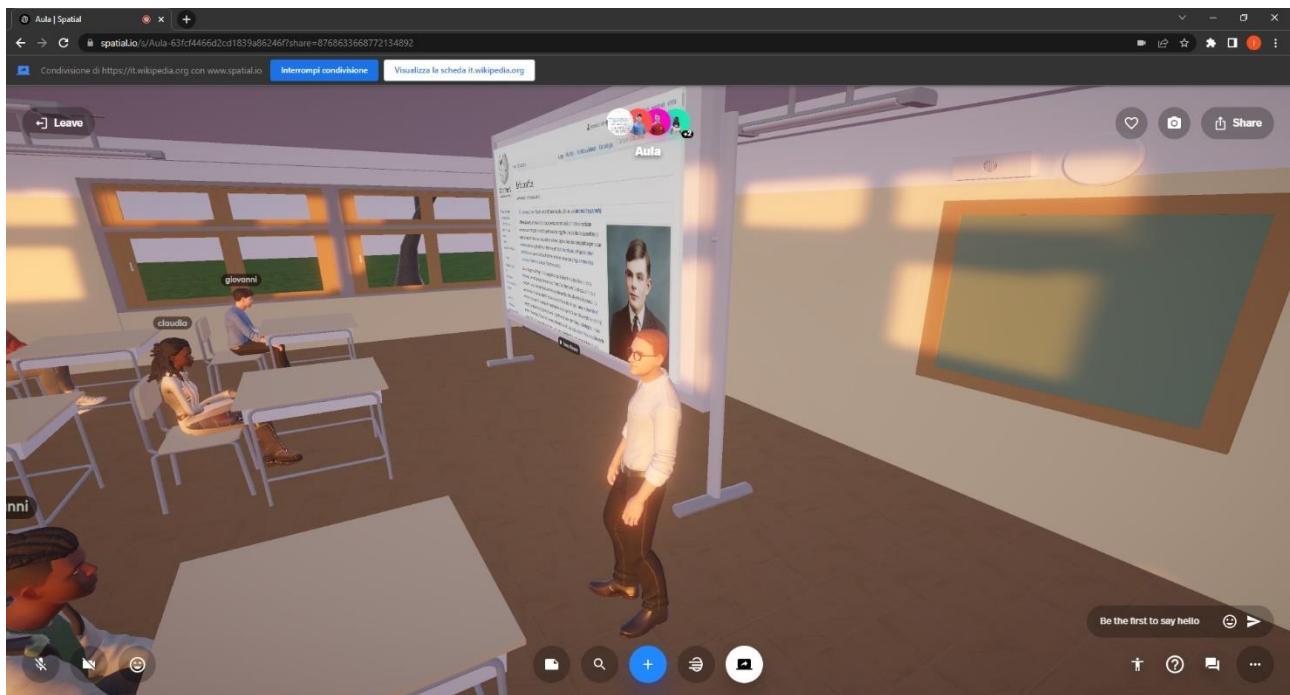
In particolare, per rendere la scena più realistica sono stati introdotti i collider per alcuni elementi della scena, come ad esempio, le mura dell'aula oppure i banchi degli alunni.



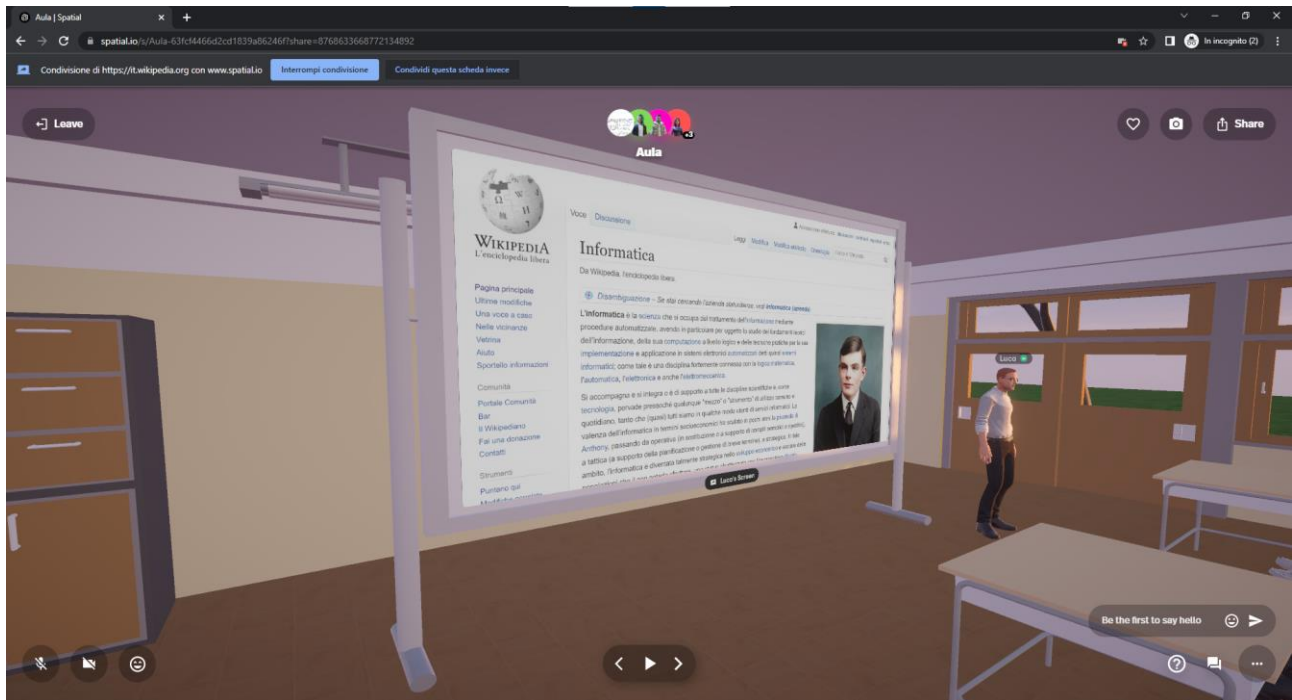
3.2.4 Risultato finale

Per avere una panoramica più chiara della scena, definiamo le due tipologie di utenti:

- L'insegnante: che si occupa del caricamento del materiale didattico sulla piattaforma e della gestione generale dell'aula virtuale.



- Lo studente: che assiste alla lezione e all'occorrenza può richiedere al docente il caricamento di un proprio elaborato.



Infine, è possibile provare la scena collegandosi al link:

<https://www.spatial.io/s/Aula63fcf4466d2cd1839a86246f?share=8768633668772134892>

4 Conclusioni e sviluppi futuri

Dallo studio condotto è emerso che sono innumerevoli le possibilità offerte dal metaverso, come abbiamo visto, pensiamo ad esempio alla possibilità di visitare luoghi altrimenti irraggiungibili o alla possibilità di sperimentare esperienze uniche come pilotare un'astronave.

Oltre a sperimentare nuove esperienze, gli utenti potranno condividere quest'ultime insieme ad amici, parenti, o chiunque essi desiderino.

Tuttavia, per la realizzazione di ambienti più complessi su queste piattaforme sono richieste conoscenze, sia in campo creativo (nella modellazione in 3D) che in campo informatico (nella scrittura di script per rendere interattiva la scena).

Entrambe le piattaforme attualmente sono ancora in via di sviluppo e presentano dei limiti nel panorama delle funzionalità e dei dispositivi supportati.

Inoltre, uno dei grandi vantaggi offerti da piattaforme di questo tipo è la possibilità di abbattere le distanze, in quanto permettono all'utente di sperimentare un'esperienza remota più realistica rispetto al passato.

Ciò è alimentato dal recente interesse nello sviluppo di applicazioni per i visori di realtà aumentata, i quali permettono una maggiore immersività all'interno di questi ambienti virtuali.

In futuro, lo sviluppo in questo settore permetterà la creazione di scenari sempre più realistici per gli utenti, accorciando sempre di più la linea di confine tra ciò che è reale e ciò che è virtuale

Riferimenti

- <https://www.digital4.biz/executive/metaverso-cos-e-possibili-applicazioni/>
- <https://www.techtarget.com/whatis/feature/The-metaverse-explained-Everything-you-need-to-know>
- <https://en.wikipedia.org/wiki/WebGL>
- https://it.wikipedia.org/wiki/Application_programming_interface
- <https://www.geekandjob.com/wiki/javascript>
- <https://it.wikipedia.org/wiki/TypeScript>
- <https://en.wikipedia.org/wiki/GITF>
- [https://it.wikipedia.org/wiki/Unity_\(motore_grafico\)](https://it.wikipedia.org/wiki/Unity_(motore_grafico))
- [https://en.wikipedia.org/wiki/Unity_\(game_engine\)](https://en.wikipedia.org/wiki/Unity_(game_engine))
- [https://it.wikipedia.org/wiki/Blender_\(programma\)](https://it.wikipedia.org/wiki/Blender_(programma))
- <https://docs.decentraland.org/creator/>
- <https://dcl-edit.com/>
- <https://builder.decentraland.org/>
- https://it.wikipedia.org/wiki/Visual_Studio_Code
- <https://it.wikipedia.org/wiki/Node.js>
- <https://render.com/>

- <https://support.spatial.io/hc/en-us>
- <https://spatialxr.notion.site/Getting-Started-with-the-Starter-Template-9b12fe1b51f848d48e16468998a10052>
- <https://it.wikipedia.org/wiki/Informatica>