# MUSEO DI ARCHEOLOGIA E SCIENZE NATURALI

**Progetto esame:**

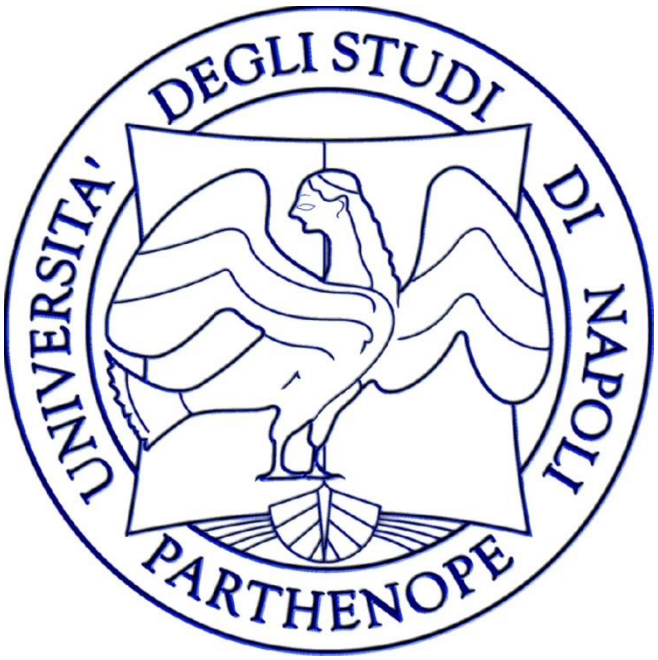
Realtà Virtuale

**Studente**:

Giovanni Castellano

**Matricola**:

0124001514




**Docenti**:

Francesco Camastra

Maurizio De Nino

**Anno Accademico**:

2019/2020

**Dispositivo**:

Oculus Go

## Legenda:


- Scopo dell'applicazione
- Sviluppo – UML
  - Primo UML – Camminata
  - Secondo UML – Oculus
  - Terzo UML – Interfacce
  - Quarto UML – Menù iniziale
  - Quinto UML – Impostazioni
  - Sesto UML – Animazioni fossili
  - Settimo UML – Audio
  - Ottavo UML – Video
- Sviluppo – Realizzazione finale
- Bibliografia – asset utilizzati

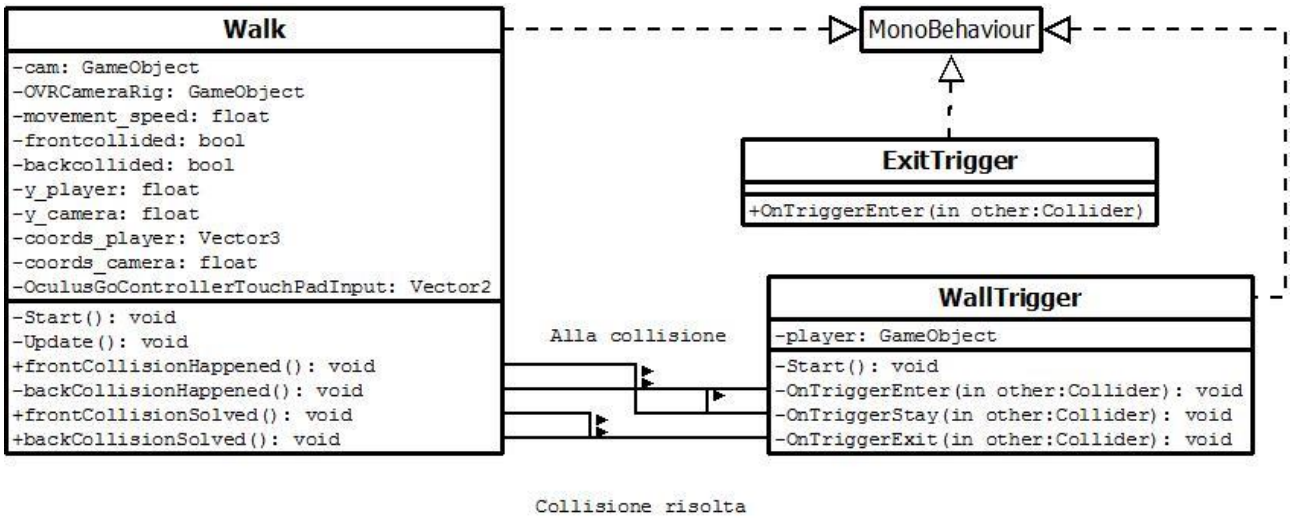

### Scopo dell'applicazione:

L'idea alla base dell'applicazione è far vivere all'utente un'esperienza in un museo virtuale diviso in due aree, una interna e una esterna. L'area esterna contiene svariati fossili di dinosauro ormai estinti in scala 1:1, potendoli osservare con luce naturale del sole, caso impossibile in un reale museo, dato che i fossili sarebbero esposti alle intemperie, ma che in un ambiente virtuale può essere realtà. Inoltre, verranno riportate delle informazioni sulla stazza dell'esemplare medio e una breve descrizione, la quale verrà letta da una voce nel mentre che si osserva il dinosauro, o, in alternativa, è possibile comunque leggerle su un pannello illustrativo. L'area interna invece si divide in due ulteriori aree, una presenta dei fossili posizionati sulle teche, interagibili in modo da poterli far fluttuare e ruotare ed analizzarli a tutto

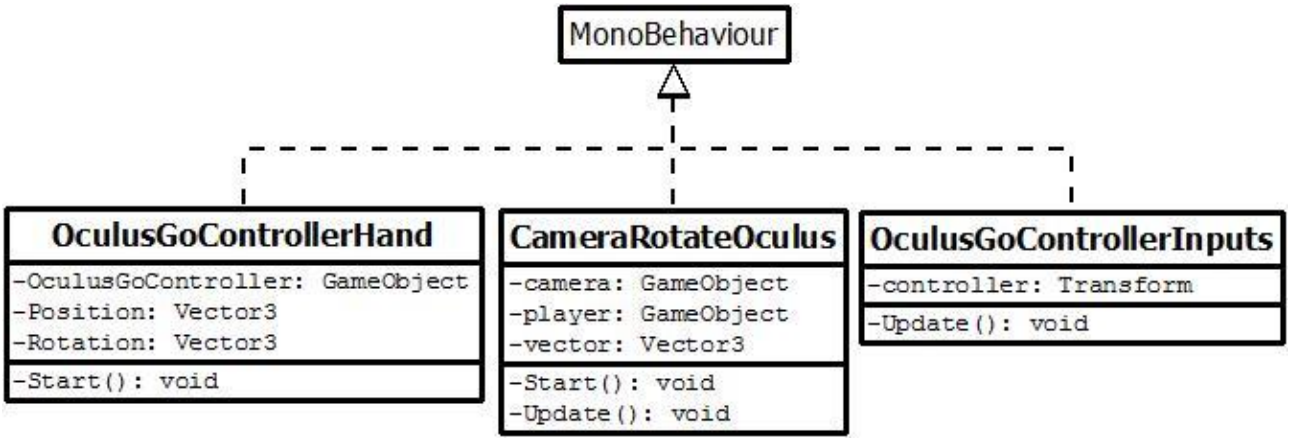
tondo, mentre l'altra area presenta una sala cinema, in cui sarà possibile visualizzare delle clip che mostrano, tramite modelli ed animazioni 3D, la riproduzione virtuale di fasi di vita di alcuni dinosauri, di fasi di caccia o della loro anatomia.

**Sviluppo – UML:**

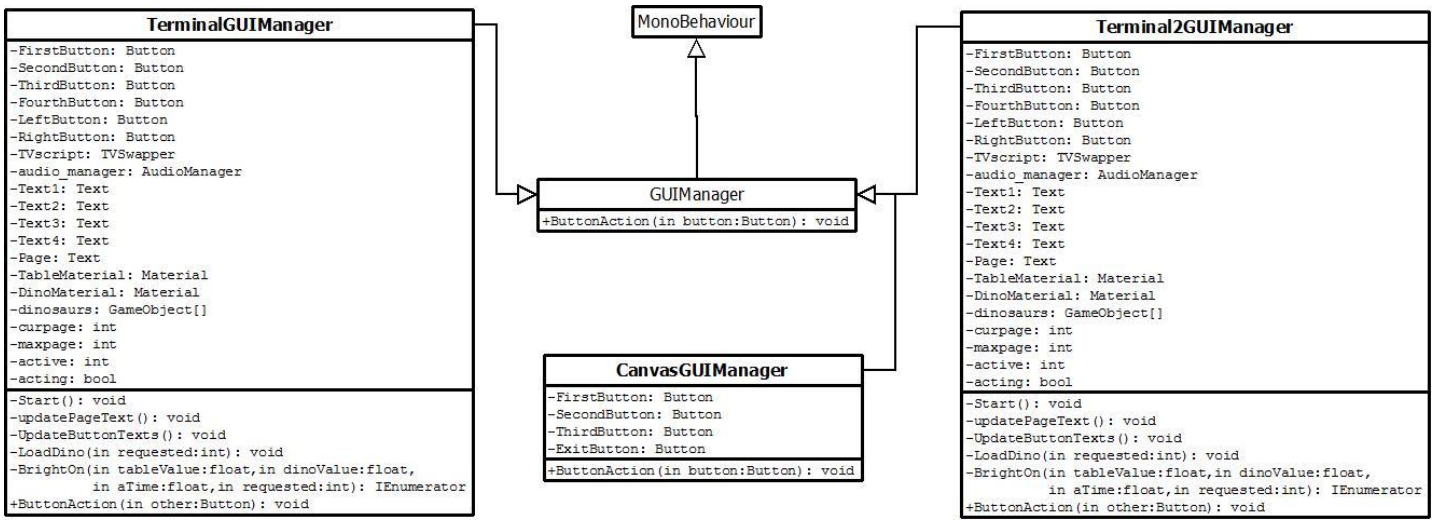
Di seguito verranno riportati i grafici UML delle varie classi utilizzate nel codice dell'applicazione, divise per associazioni tra di loro:




Innanzitutto, lo schema UML delle classi che si occupano della camminata dell'utente. Unity dovrebbe bloccare il movimento di due oggetti che collidono tra loro, laddove uno di essi sia ancorato, ma in questo caso non è così: lasciando l'utente camminare verso qualsiasi oggetto, egli avrebbe potuto attraversarlo senza problemi, sia esso stato un muro, un tavolo o un terminale, per cui ho creato un sistema di camminata in cui il player ha sia davanti che dietro di se due sfere che andranno a collidere con qualsivoglia oggetto prima che lo faccia il player; in questo modo, attiveranno un trigger su tutti gli oggetti che hanno lo script "WallTrigger" attivato, il quale bloccherà il movimento del player in avanti o indietro, in base a quale sfera (chiamate "Frontchecker" e "Backchecker") ha attivato il trigger. Una volta che il checker sarà uscito dal trigger, il che è possibile semplicemente ruotando la telecamera altrove, il movimento verrà nuovamente sbloccato. Infine, la classe "ExitTrigger" serve ad attivare un trigger posizionato su una porta d'uscita che, laddove attivato, chiude l'applicazione; sarà quindi possibile chiudere l'applicazione semplicemente camminando verso la porta d'uscita, uscendo una volta raggiunta.

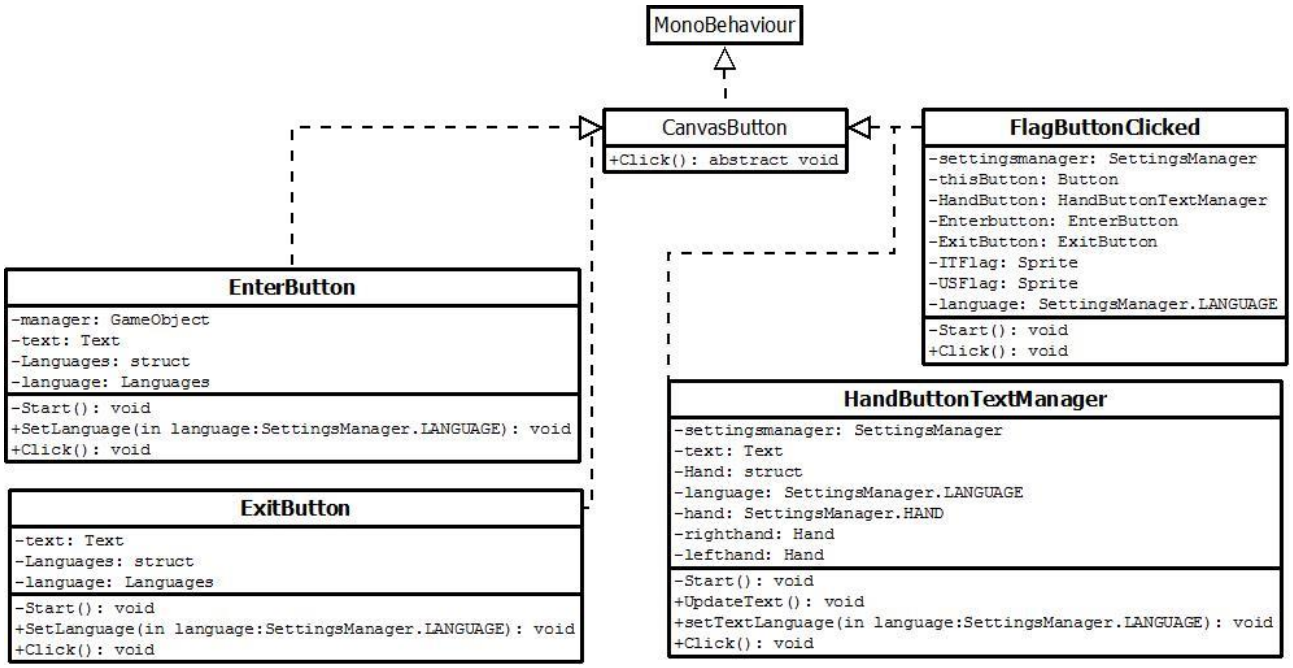



Il secondo grafico UML riguarda le classi necessarie a far funzionare l'applicazione su Oculus Go: la classe "OculusGoControllerHand" è una classe col solo metodo start che imposta il controller a destra o a sinistra dell'utente in base a se ha scelto di essere destrorso o mancino; la classe "CameraRotateOculus" si occupa di ruotare il corpo del player e i checker assieme alla telecamera, nella direzione in cui l'utente sta guardando: nonostante essi siano invisibili, è necessario che ruotino nella direzione della vista dell'utente dato che essa sarà anche la linea che seguirà lo script di movimento, e i checker devono essere ben posizionati per bloccare l'utente in caso di urto con qualsivoglia oggetto.

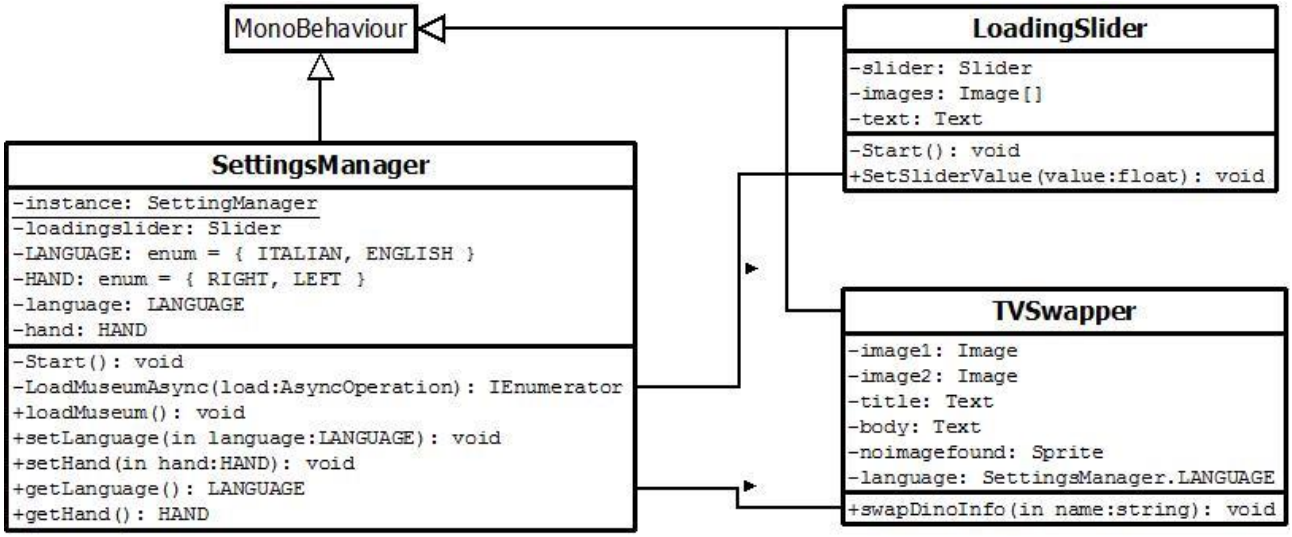



Questo UML riguarda l'interfaccia con cui l'utente può interagire, ed è composto da una classe astratta "GUIManager" che estende la classe di Unity "MonoBehaviour", e da altre due classi che estendono la classe astratta "GUIManager": "CanvasGUIManager" riguarda l'interfaccia del menù iniziale, e definisce un metodo ButtonAction come override dalla classe padre in cui definisce come attivare le funzioni dei Button cliccati, mentre la classe "TerminalGUIManager" riguarda l'interfaccia del terminale presente nella scena del museo, in cui i Button non hanno direttamente delle azioni

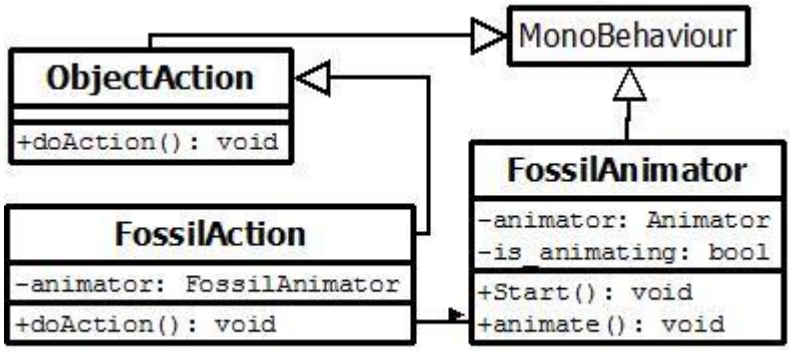
collegate da eseguire, ma ritornano semplicemente la stringa del loro testo allo script che esegue le operazioni appropriate di conseguenza.




Più precisamente, l'UML del menù inziale spiega meglio come i Button eseguono il loro compito, e tutti lo fanno tramite il metodo Click, definito in una classe astratta generica CanvasButton, dalla quale le altre classi, concrete, ereditano il metodo e eseguono l'override. L'"EnterButton" è lo script del Button per entrare nel museo, e si occupa di caricare la scena del museo, comunicandolo al SettingsManager. L'"HandButtonTextManager" è lo script del tasto che riguarda la mano scelta, e come semplice compito ha lo switch della stringa su di esso scritta con destrorso o mancino e la loro versione inglese, e di comunicare al SettingsManager quale opzione è stata scelta. Il "FlagButtonClicked" cambia la lingua del menù ad ogni click, e comunica la lingua selezionata al SettingsManager, infine, l'"ExitButton" chiude l'applicazione se cliccato.

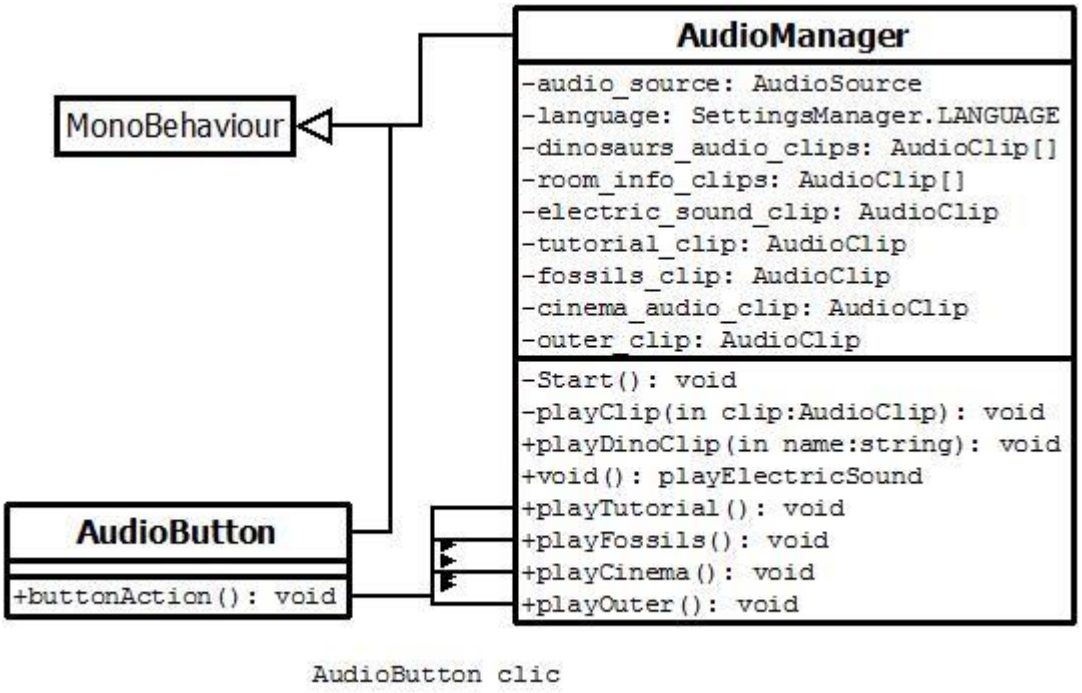



Il SettingsManager è l'unico oggetto che non viene distrutto dal menù iniziale e viene conservato anche nella seconda scena: ha un ruolo importantissimo, dato che conserva le informazioni dettate dal menù riguardo la mano del controller e la lingua scelta, e queste informazioni saranno utili sia per posizionare il controller a destra o a sinistra dell'utente, sia per, come mostrato anche dall'UML, cambiare la lingua dei testi del pannello informativo. Infine, si occupa anche di caricare la scena del museo in modo asincrono, tale che non si abbia una sensazione di blocco dell'applicazione, e nel mentre che carica aggiorna anche lo slider che indica la percentuale di caricamento. Lo script TVSwapper si occupa di caricare, ad ogni cambio di fossile visualizzato, il testo e le due immagini inerenti al dinosauro caricato, e comunica ogni volta con il SettingsManager per leggere quale lingua deve utilizzare.



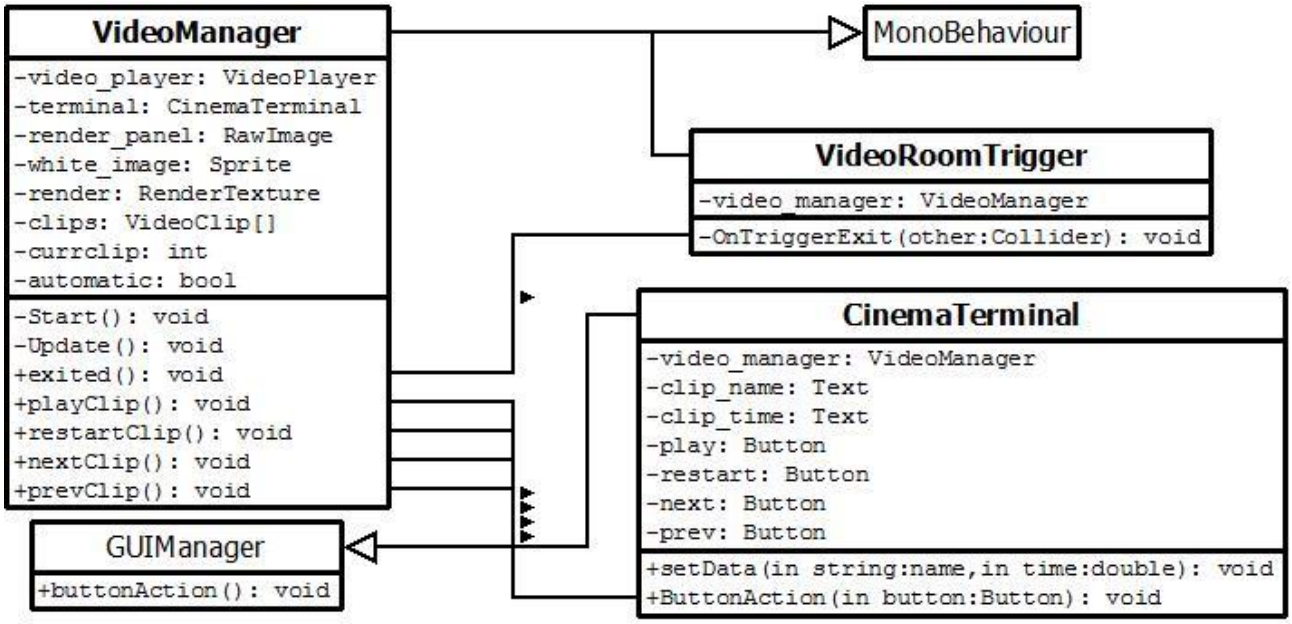

La sala dei fossili consente di visualizzarli e animarli, facendoli fluttuare e ruotare, in modo da poter essere osservati a tutto tondo. Le animazioni sono definite tramite l'Animation di Unity, ma gli script che consentono l'interazione utente-fossile sono descritti in questo diagramma. Il FossilAnimator fa partire l'animazione di inizio se è la prima interazione, e fa partire l'animazione di fine in caso di una seconda interazione, e l'interazione è gestita dal

FossilAction, classe figlia dell'ObjectAction, la quale vede attivarsi il suo metodo doAction nel caso si sia cliccato il fossile su cui è applicata. La classe astratta ObjectAction è necessaria nello script ButtonCollider, dato che definisce una firma di metodo chiamato virtualmente da quest'ultimo, e nel caso dei fossili, il metodo concreto è definito da FossilAction.




AudioButton clic

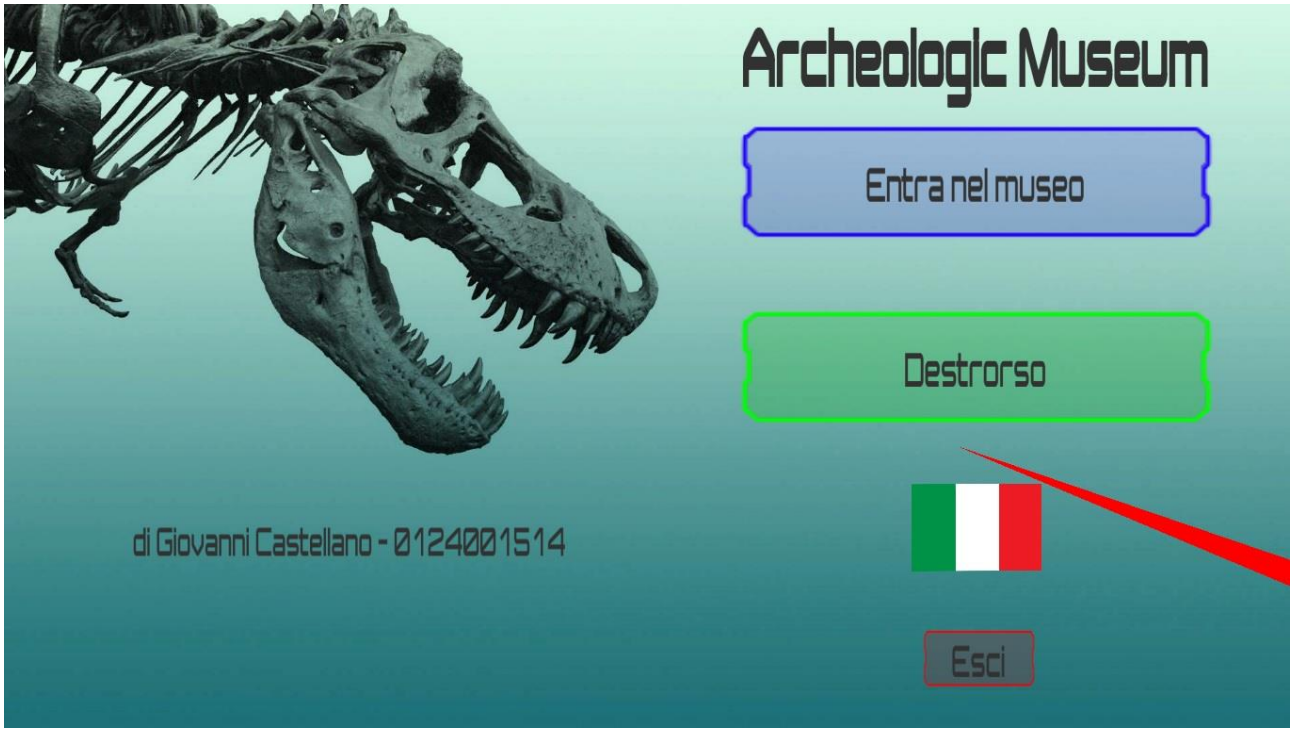
Questo diagramma analizza le classi utili al comparto audio, quindi tutto ciò che ha a che fare con i pannelli audio e con l'audio ascoltabile al cambio di dinosauro visualizzato. L'AudioManager si occupa di tutto l'audio presente nell'applicazione, caricandolo nello Start e mandandolo in play all'occorrenza. L'AudioButton è una classe che identifica il pulsante dei pannelli audio, riconoscibile dallo script ButtonCollider. AudioButton si occupa di far partire la clip audio del pannello cliccato.



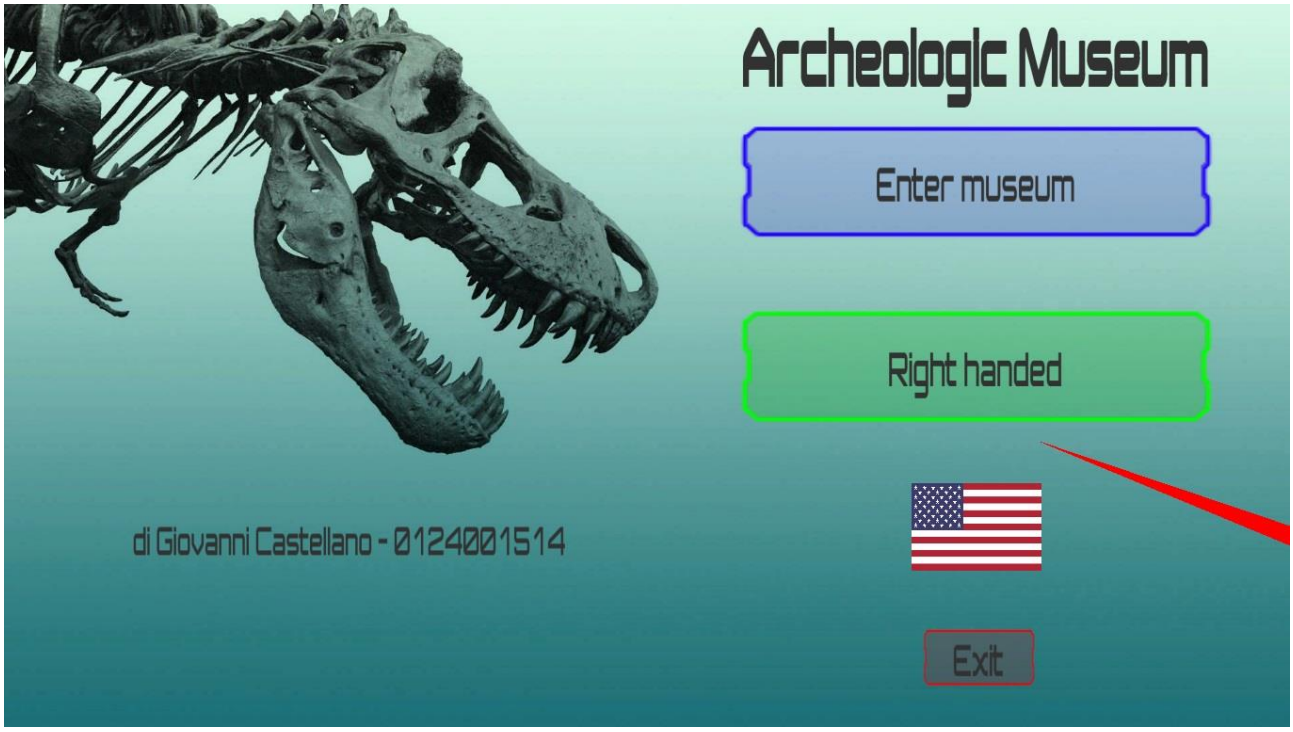

La sezione video dell'area cinema è divisa in tre classi, la VideoManager è la classe principale, che si occupa di caricare le clip video e di far funzionare il cinema, proprio come un video player, mentre la classe CinemaTerminal si occupa dell'interfaccia del terminale del cinema, con cui è possibile interagire e che fa da tramite per poter avviare clip, metterle in pausa, farle ricominciare e caricare la prossima o la precedente clip. Nel caso una clip finisca, la classe VideoManager si occupa anche di caricare ed avviare la successiva automaticamente, e può essere interrotto mettendo pausa o caricando la clip precedente o la successiva. Infine, la classe VideoRoomTrigger si occupa del trigger presente nell'area cinema, per cui nel caso l'utente dovesse lasciare la stanza nel mentre che il video è in play, il trigger si occuperà di mettere il video in pausa, e resterà in attesa che venga manualmente fatto ripartire dal terminale.

**Sviluppo – Realizzazione finale:**

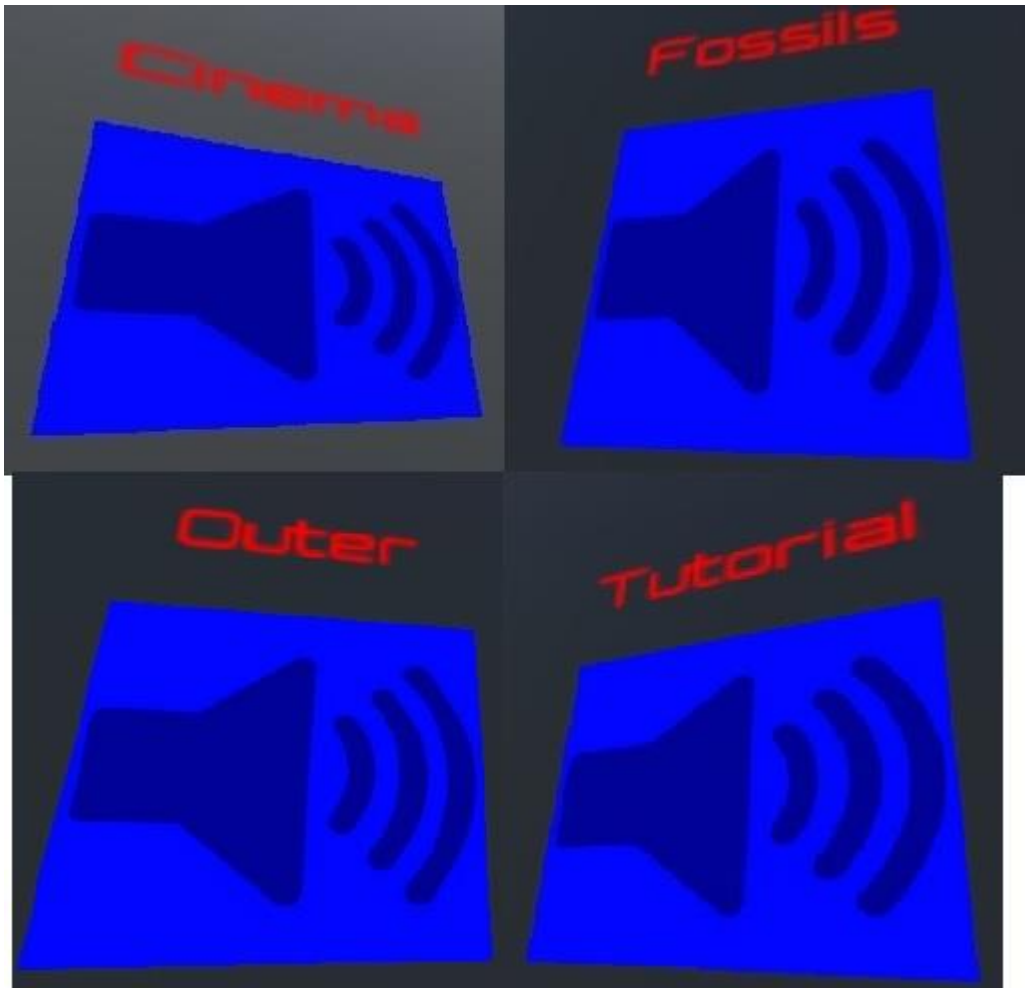
In seguito saranno mostrate varie immagini illustrative della realizzazione finale:



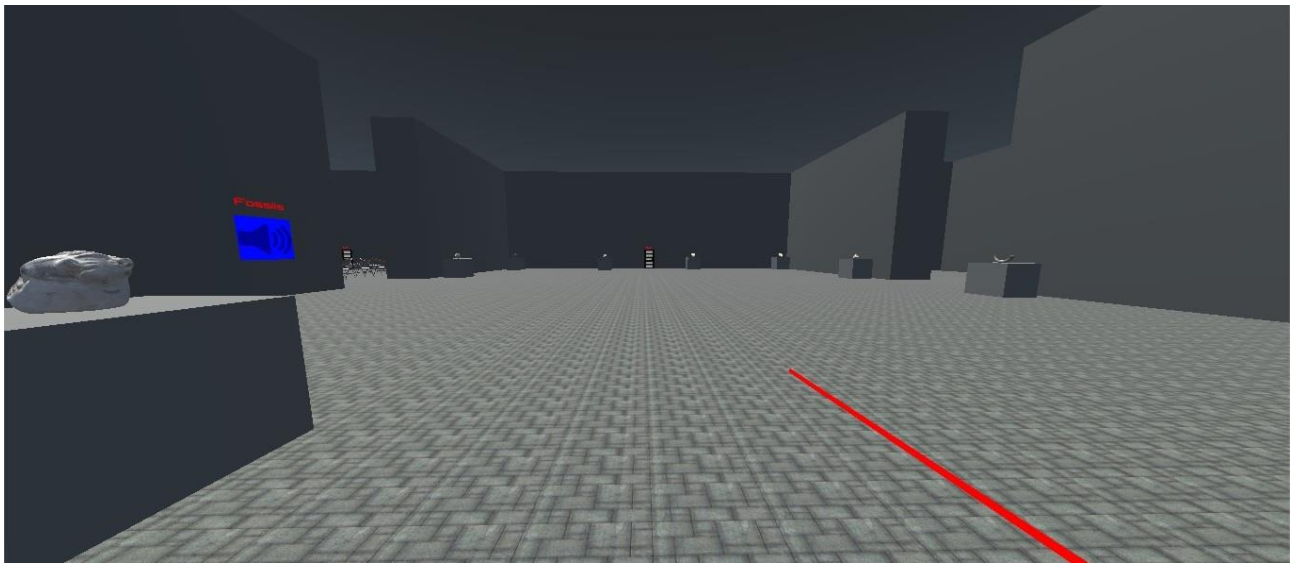

Menù iniziale, in lingua italiana



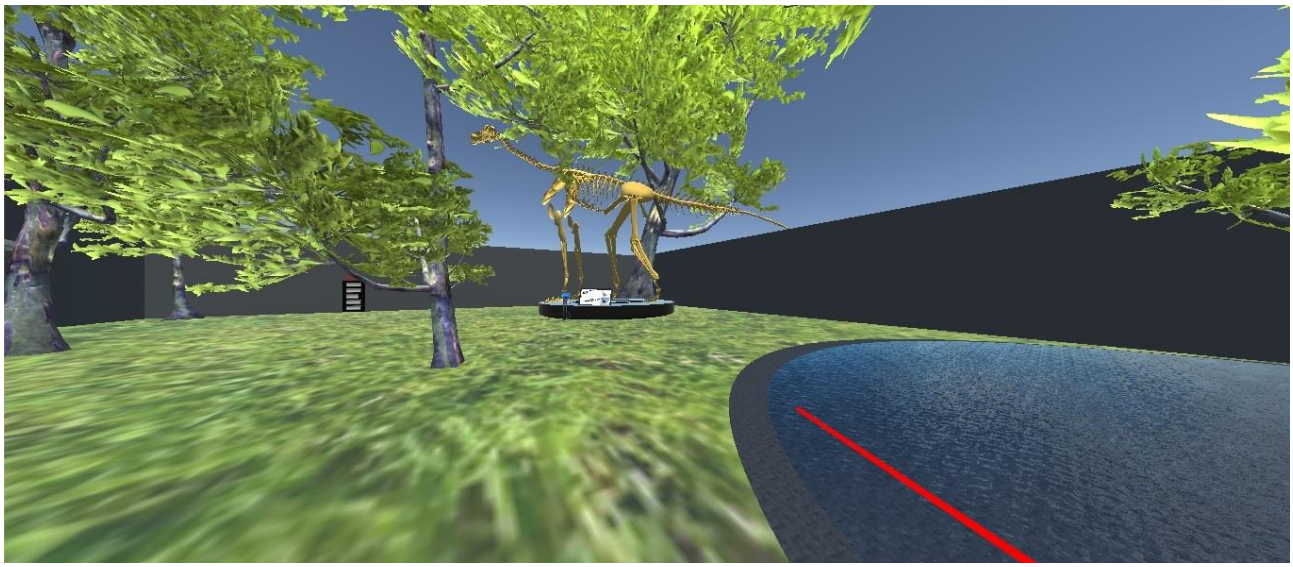

Menù iniziale, in lingua inglese



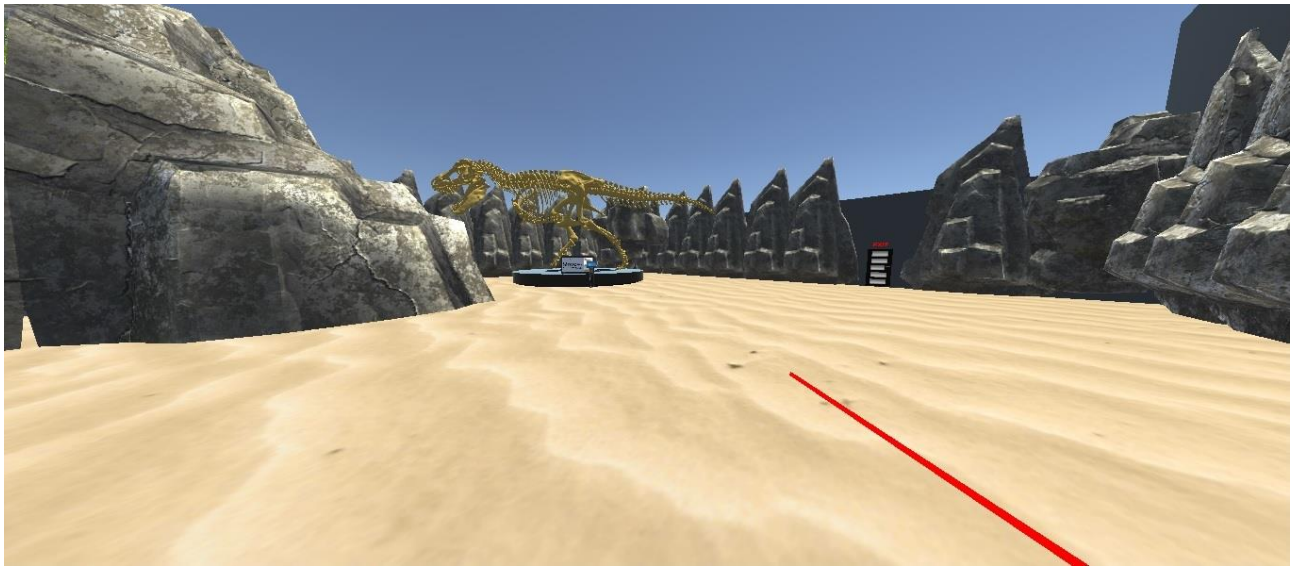

Pannelli audio presenti nelle varie sale, che riproducono audio in italiano o inglese



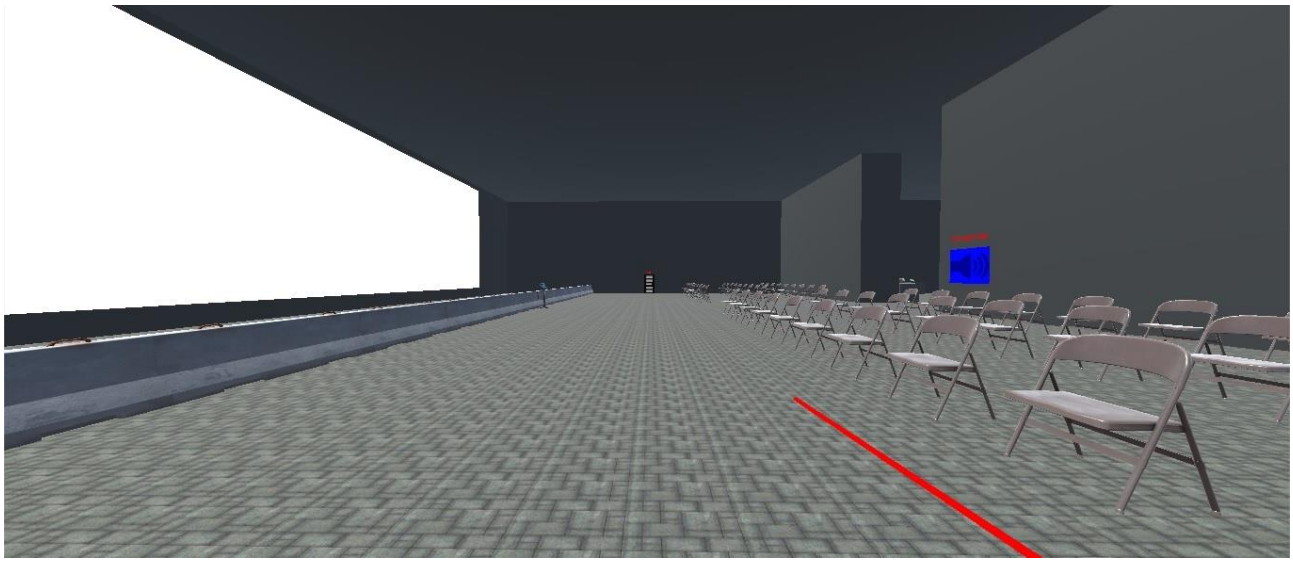

Area interna dedicata ai fossili



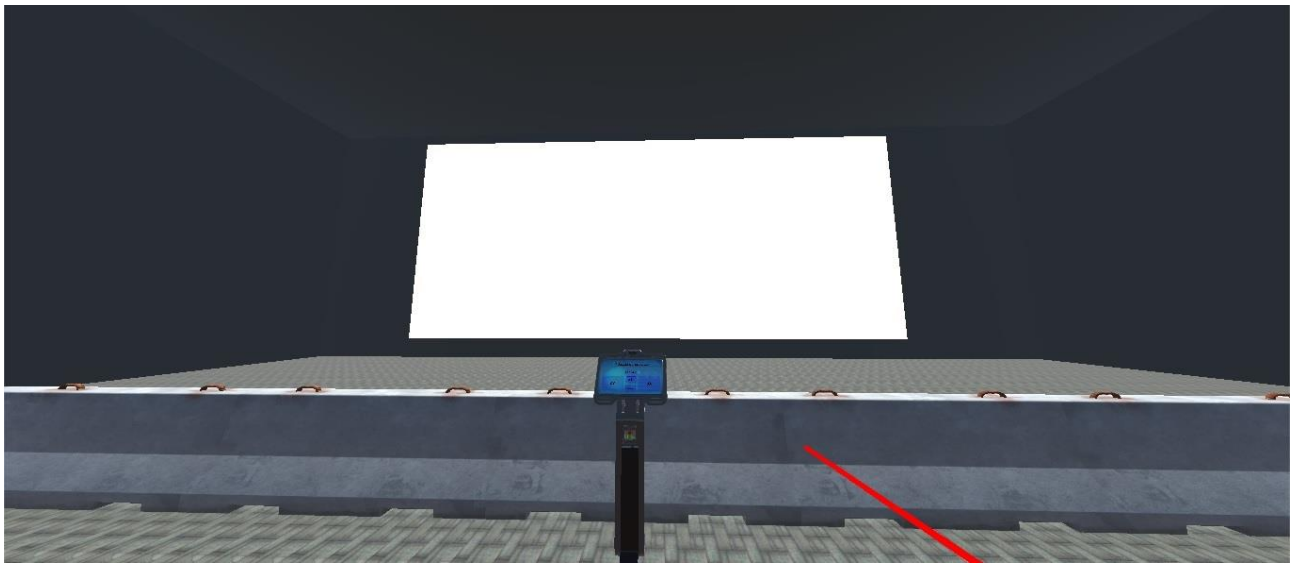
Area esterna dedicata agli erbivori e piscivori



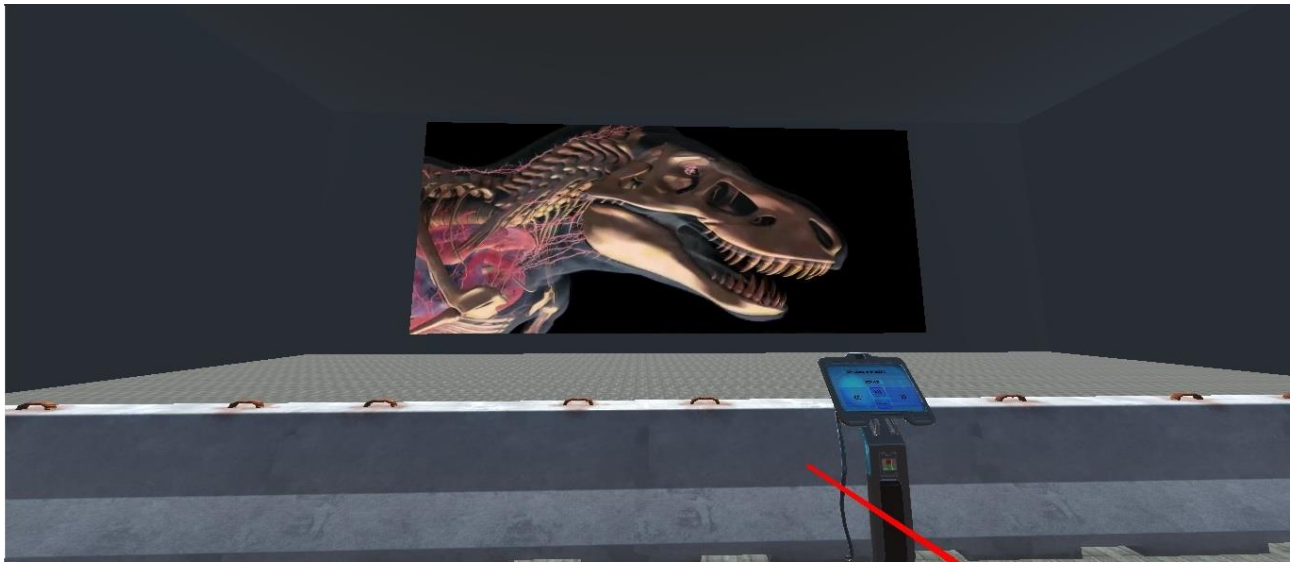
Area esterna dedicata ai carnivori



Sala cinema, con telo su cui riprodurre i video



Telo e terminale della sala cinema



Video in riproduzione nella sala cinema

**Bibliografia – asset utilizzati:**

Unity Asset Store (asset gratuiti):

- Oculus Integration – OCULUS
- Unity Samples: UI – UNITY TECHNOLOGIES
- PBR Folding Chairs – WIREFRAME WAREHOUSE
- Barrier & Traffic Cone Pack – SABRI AYES
- Water Shader – NVJOB
- Realistic Tree 9 – RAKSHI GAMES
- Free Rocks – TRIPLEBRICK

Ulteriori modelli 3D gratuiti sono stati scaricati dai seguenti siti web:

- Sketchfab
- 3D WareHouse