UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

Dipartimento di Ingegneria Elettrica e delle Tecnologie dell'Informazione

Classe di Laurea in Ingegneria Elettronica, Classe n. L.8

Corso di Laurea in Ingegneria Elettronica

Elaborato di Laurea

Caccia al tesoro in realtà aumentata con Unity e Vuforia

Relatore:

Ch.mo Prof. Porfirio Tramontana

Candidato:

Gennaro Mariano Altobelli

Matr. N43001003

Anno Accademico

2020/2021

がんばれ,がんばれ! 石の上にも３年。

Ganbare,Ganbare! Ishi no ue ni mo sannen

Indice

Elenco delle figure

Introduzione

Nell'ultimo decennio, la realtà aumentata, si sta affermando in vari settori da quello industriale, all'ambito medico e all'entertainment. Il primo utilizzo di realtà aumentata, anche se si sta diffondendo solo recentemente, risale al 1968 da un lavoro di Ivan Sutherland, ma il suo utilizzo concreto, con visioni piu coerenti e chiare, si ha solo quando vengono introdotti miglioramenti tecnologici e grazie all'avvento dell'elettronica miniaturizzata. Si iniziano a trovare prodotti sul mercato solo alla fine del primo decennio del duemila, questo è dovuto anche all'affermazione sul mercato di occhiali per la realtà aumentata e di smartphone. Così come le altre tecnologie, anche questa è in continuo sviluppo, ed anche se esordisce con l'entertaiment sta trovando nell'infotainment la sua concreta vocazione grazie alla quantità e qualità di informazioni rese semplici,chiare ed immediate, riducendo cosi i costi produttivi.
Vediamo ora cos'è la realtà aumentata,le sue applicazioni nella realtà, ed infine la sua differenza con la realtà virtuale, dato che queste vengono spesso confuse.

Capitolo 1

Realtà Aumentata

1.1 Definizione

Secondo il sito Wikipedia :
*"Per realtà aumentata(in inglese augmented reality, abbreviato "**AR**"), o realtà mediata dall'elaboratore, si intende l'arricchimento della percezione sensoriale umana mediante informazioni, in genere manipolate e convogliate elettronicamente, che non sarebbero percepibili con i cinque sensi."* [1]
Quindi la normale realtà viene alterata ,attraverso un dispositivo, con l'aggiunta di informazioni virtuali con le quali è anche possibile interagire, da tale aggiunta deriva il nome realtà aumentata.

1.2 Come funziona?

La realtà aumentata, come già accennato in precedenza, può essere implementata attraverso diversi dispositivi:

- **Attraverso smathphone e tablet**: sicuramente il più diffuso, dato che i moderni smartphone sono dotati di giroscopi, accelerometro, GPS, fotocamera. Grazie alla fotocamera posso avere delle informazioni aggiuntive non presenti nella realtà sul display del mio dispositivo, questo può essere utile ad esempio per avere informazioni su un dato macchina-

rio, inquadrandolo , attraverso video interattivi, immagini o testi, posso capire il suo funzionamento,come effettuare la manutenzione, ecc...

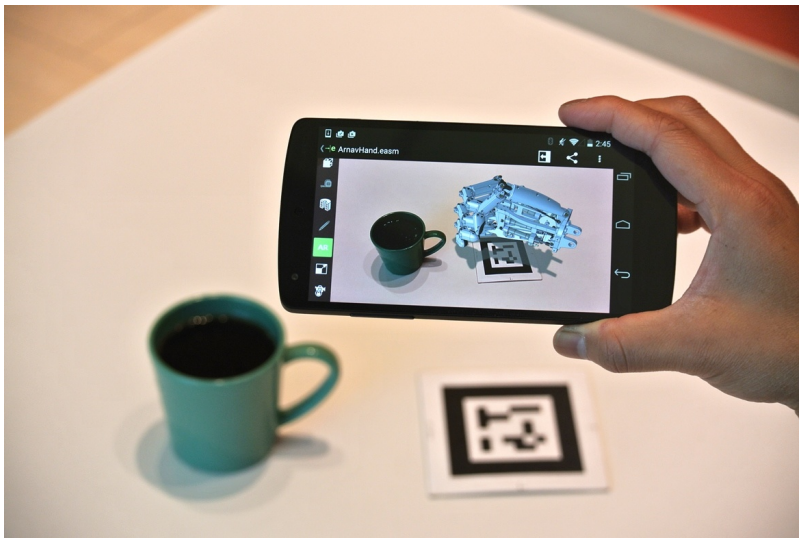


Figura 1.1: AR attraverso smarthphone [2]

- **Attraverso occhiali**: anche questu abbastanza diffusi soprattutto grazie ai Google Glass della XLab, anche se non sono i migliori in circolazione, tra i migliori troviamo sicuramente gli Hololens della Microsoft. Gli occhiali sono muniti di una fotocamera e mediante uno schermo posizionato sull'occhio destro, le informazioni vengono sovrapposte alla realtà.

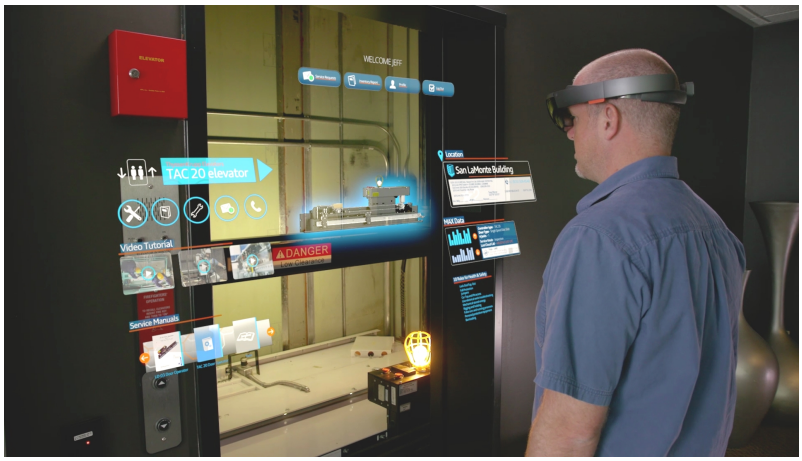


Figura 1.2: AR attraverso Hololens [3]

1.3 Applicazioni della Realtà Aumentata nei vari settori

La Realtà Aumentata viene utilizzata in diversi settori, varie aziende si stanno evolvendo in questo settore per avere dei benefici.
Ecco alcuni esempi di utilizzo:

- **Campo medico**: viene utilizzata per avere modelli 3D per studiare e per ricreare interventi chirurgici. Attraverso gli scanner AR è possibile ricreare tutte le possibili ramificazioni delle vene di un dato paziente e facilitare in questo modo i prelievi di sangue.

- **Educazione ed Apprendimento**: gli studenti possono benificiare di questa tecnologia attraverso l'uso di libri interattivi che godono anche di effeti sonori. Inoltre possono interagire maggiormente con l'ambiente che li circonda.

- **Navigazione**: grazie all'utilizzo del AR è possibile avere tutte le informazioni utili nella visuale del guidatore, in questo modo non viene distratto durante la guida.

- **Manutenzioni e riparazioni**: per verificare il corretto funzionamento di un macchinario o correggere i suoi eventuali errori, è possibile far ricorso all' AR che grazie a immagini o video 3D mi permette di risolvere in maniera facile e veloce. L'AR può essere utilizzata anche come un manuale.

- **Viaggi e turismo**: attraverso le mappe è possibile avere delle informazioni aggiuntive su un determinato luogo, monumento, su hotel, ristoranti con relative recensioni, quindi in generale indica tutti i luoghi d'interesse che ci circondano, arrichendo cosi l'esperienza di viaggio.

- **Videogiochi**: quando si parla di videogiochi AR, il più famoso è senza dubbio Pokemon GO, grazie alla fotocamera ed al GPS del proprio sma-

thphone, ha trasformato i giocatori in Ash Ketchum [1] alla ricerca dei Pokemon. In circolazione esistono molti altri giochi ma sono per lo più sparatutto e missioni tattiche.

- **Arte**: all'interno dei musei riesce a fornire un esperienza unica, grazie a contenuti aggiuntivi sugli oggetti esposti che diventano interattivi. Anche in italia ci sono alcuni musei che utilizzano l' AR , tra questi troviamo il Museo del Castello Sforzesco di Milano che utilizza un app, chiamata MusA-Museo Accessibile, che aiuta gli ipovedenti ingrandendo a piacimento le opere, modificando il contrasto e scegliere quale parte dell'opera farsi descrivere.

- **Militare**: permette di ridurre notevolente i costi per la formazione di un soldato, evitando di consumare armi monuso come bombe e proiettili e ridurre l'uso di aerei e quindi minor spreco di carburante. Con l'AR è possibile aumentare la visione dei militari, cosi possono avere numerose informazioni come la posizione dei nemici, la conformazione del territorio da raggiungere, miglioramento della mira.

In ogni settore la tecnologia è in continuo sviluppo per cercare di portare maggior benefici, riduzioni dei costi e un maggior numero di informazioni.

1.4 Realtà Virtuale

Secondo il sito Wikipedia:
*"Con il termine realtà virtuale (a volte abbreviato in "***VR***" dall'inglese virtual reality) si identificano vari modi di simulazione di situazioni reali mediante l'utilizzo di computer e l'ausilio di interfacce appositamente sviluppate."* [4]

La realtà virtuale crea un ambiente simile a quello reale, l'utilizzatore che si trova nel mondo virtuale ha la sensazione di trovarsi fisicamente lì.

[1] Protagonista dell'omonimo anime

Questo è dovuto anche ad interfaccie sofisticate, come visori nei quali è presente la scena e guanti per le sensazioni tattili e movimenti.
Questa tecnologia si è dimostrata particolarmente efficace nell'attuale pandemia attraverso la telemedicina.

"In situazioni catastrofiche, come l'attuale pandemia, il rischio infettivo limita l'accesso ai servizi ospedalieri ed ai dipartimenti di emergenza ed urgenza. In tali situazioni, l'utilizzo delle nuove tecnologie, quali la realtà virtuale, può facilitare l'accesso ad alcuni dei servizi del sistema sanitario tramite interventi di telemedicina. Le nuove tecnologie si sono dimostrate efficaci in situazioni di pandemia nel migliorare la qualità di vita dei pazienti, in particolare in coloro affetti da malattia cronica e negli anziani. Inoltre, gli interventi di telemedicina forniscono soluzioni per il monitoraggio e la valutazione dei pazienti da remoto, evitando il rischio di contagio" [5]

1.5 Differenza tra Realtà Aumentata e Virtuale

Figura 1.3: AR vs VR [6]

Una volta capito cosa sono AR e VR e facile comprendere che la principale differenza tra le due consta in come l'utente interagisce con la realtà, nell AR l'utente è ancora nella realtà che è arricchita di informazioni digitali, invece nella VR viene creata una nuova realtà e l'utente viene immerso in quel mondo, quindi è estraniato dal mondo fisico.

"L'elemento più importante che distingue AR e VR è, come abbiamo detto, il fatto di continuare a vedere il mondo fisico mentre si ha accesso alle informazioni digitali." [7]

Quindi la differenza consta nel grado di immersività poichè in una abbiamo l'utente che è legato al mondo reale nell'altra, invece, l'utente è totalmente immerso in mondo nuovo.

"Un'esperienza di realtà virtuale immerge completamente l'utente in un ambiente nuovo, che lo circonda e lo avvolge. L'esperienza è totale e assoluta. La grande magia della realtà virtuale è affascinante da vedere da fuori. Quando osserviamo qualcuno che indossa un visore ed entra nella scena, possiamo capire con chiarezza quando il suo cervello fa click e quella persona non è piu con noi, anche se la vediamo muoversi al nostro fianco." [7]

Questa completa immersività del VR porta l'utente ad escludersi dal mondo circostante ed a vivere un esperienza solitaria in un mondo che può avvertire solo lui. Oggi si cerca di creare un mondo virtuale in cui sono presenti più persone, anche se la progettazione è ancora allo stadio sperimentale.

Capitolo 2

Tools per AR

2.1 SDK per AR

La grande diffusione della reatà aumentata è dovuta anche alla diffusione degli SDK (Kit di sviluppo software, in italiano) che hanno permesso lo sviluppo di applicazioni AR per device mobili. Tali SDK sono semplici da utilizzare e in essi sono rinchiuse molte funzioni complesse, permettendo al sviluppatore di realizzare programmi per la realtà aumentata. Esistono diversi SDK e quelli piu diffusi sono tutti compatibili con i principali sistemi operativi, Android ed iOS, entrambi o solo uno dei due, ed alcuni sono compatibili anche con Windows. Vediamo ora quali sono i più diffusi:

- **ARkit**, compatibile solo con iOS;
- **ARCore**, compatibile sia con Android che iOS anche se con quest'ultimo limitatamente, sono presenti meno funzioni;
- **SimpleCV**, non è legato a nessuna piattaforma specifca e risulta compatibile con C++, Java e Python;
- **Kudan** ,compatibile con iOS ed Android e fornisce plug-in Unity;
- **Wikitude**, compatibile con Android, iOS, Windows e smart glasses. Le app possono essere sviluppate in JavaScript, Unity, Cordova, Xamarin, Flutter e Native API;

- **Vuforia**, compatibile con Android, iOS and UWP(Universal Windows Platform). Le App possono essere compilate con Unity, Android Studio, Xcode, e Visual Studio.

Menzione a parte per l'AR SDK di Pikkart, azienda italiana, che è compatibile con iOS, Android e Windows. La Pikkart è riuscita a sviluppare una tecnologia unica, alla stessa immagine posso associare informazioni diverse di realtà aumentata.

"Durante il roadshow di AWE 2016, il ciclo di fiere inerenti la Realtà Aumentata e tecnologie affini più importante al mondo, Pikkart ha lanciato l'AR SDK proprietario, la soluzione ideale per sviluppatori, web/marketing agency e fornitori hw/sw che desiderano: realizzare in autonomia innovative AR App per dispositivi mobile; integrare all'interno di App preesistenti funzioni avanzate di Realtà Aumentata." [8]

2.1.1 Vuforia

L'SDK da me scelto è Vuforia, l'SDK più utilizzato per lo sviluppo delle realtà aumentata, fornisce un tracking dell'immagine veloce ed accurato. Lo schermo del tuo dispositivo viene utilizzato come lente e sovrappone elementi virtuali alla realtà fisica in modo da farli sembrare reali.
Mi permette di inserire pulsanti virtuali e creare effetti di sfondo.
In Vuforia è possibile creare un database con i propri targets. Una volta iscritti sul sito in maniera gratuita, cliccando su Target Manager possiamo aggiungere i nostri targets che possono essere:

- **Single Image**
- **Cuboid**
- **Cylinder**
- **3D Object**

Figura 2.1: Inserimento di un nuovo target in un database di Vuforia

Bisogna inoltre inserire la larghezza ed il nome che esso avrà nel database. Successivamente è possibile fare il download del proprio database per le seguenti piattaforme di sviluppo: Android Studio, Xcode, Visual Studio ed Unity Editor.

Vuforia, però, è gratuita solo in fase di sviluppo , se si vuole mettere l'applicazione sui market di Apple ed Android bisogna tener conto che bisogna pagare 500€ all'anno per ogni applicazione. Per informazioni specifiche e piu dettagliate sul suo utilizzo è possibile consultare la libreria di Vuforia [9].

2.2 Game Engine

"Il motore di gioco ,detto anche motore grafico, è il nucleo software di un videogioco o di qualsiasi altra applicazione con grafica in tempo reale. Esso fornisce le tecnologie di base, semplifica lo sviluppo e spesso permette al gioco di funzionare su piattaforme differenti come le console o sistemi operativi per personal computer". [10]

La funzionalità di base fornita tipicamente da un motore grafico include:

- Un motore di rendering [1] per grafica 2D e 3D;
- Un motore fisico realistico;
- Supporto audio;
- Scripting;
- Assoluta flessibilità con le animazioni degli oggetti di scena;
- Gestione intelligenza artificiale;
- Networking;
- Gestione automatica delle performance del computer per garantire fluidità;

Tra gli Engine, più diffusi, che rispecchiano queste caratteristiche troviamo Unity ed Unreal Engine. Entrambi non sono usati solo per videogiochi ma anche per realtà interattive, simulazioni e nelle produzioni cinematografiche, tra queste troviamo Mandalorian e Westworld.

[1] Un motore di rendering, in informatica ed in particolare nella computer grafica, è un componente hardware o software che interpreta delle informazioni in ingresso codificate secondo uno specifico formato e le elabora creandone una rappresentazione grafica. [11]

2.2.1 Unity

Il game engine da me scelto è Unity. Possiamo trovare sia una versione perennemente gratuita che una a pagamento che di diverso ha soltanto il colore delle interfaccie più qualche piccola ottimizzazione. Nonostante sia gratuito, è uno strumento potentissimo, basti pensare che alcuni giochi importanti come Hearthstone della Blizzard Entertainment e Among Us della Innersloth sono stati realizzati con questo game engine. Installabile su Windows, su macOS e su Linux, i giochi, le applicazioni o in generale, quello che si vuole realizzare tramite Unity, sono compatibili con Mac, Linux, Windows, Android, iOS, PS3, PS4, PS5, xbox one ed altri ancora [Vedi Figura 2.2].Come per Vuforia anche qui è richiesta la registrazione al sito. In figura 2.3 è possibile vedere l'interfaccia di Unity in cui è presente un GameObjet [2], in particolare un 3D Object, il Cubo.

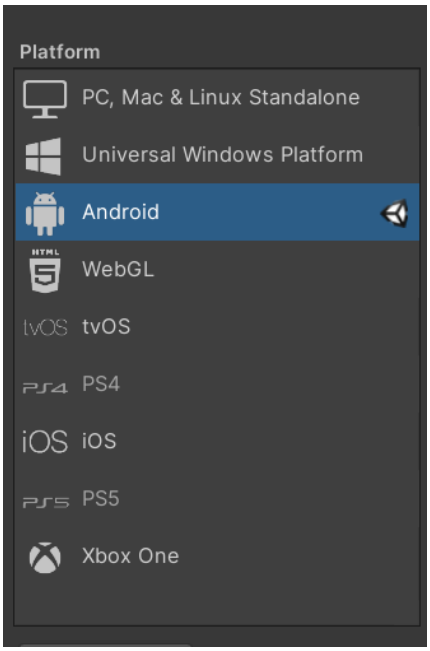


Figura 2.2: Piattaforme sulle quale posso realizzare giochi o altre applicazioni.

[2]In Unity sono gli oggetti che vengono utilizzati all'interno di una scena, quindi altro non sono che gli oggetti che ritroviamo nel nostro gioco. Essi, però, posso essere sia presenti all'avvio del gioco sia spawnati durante esso, quindi instanziati successivamente.

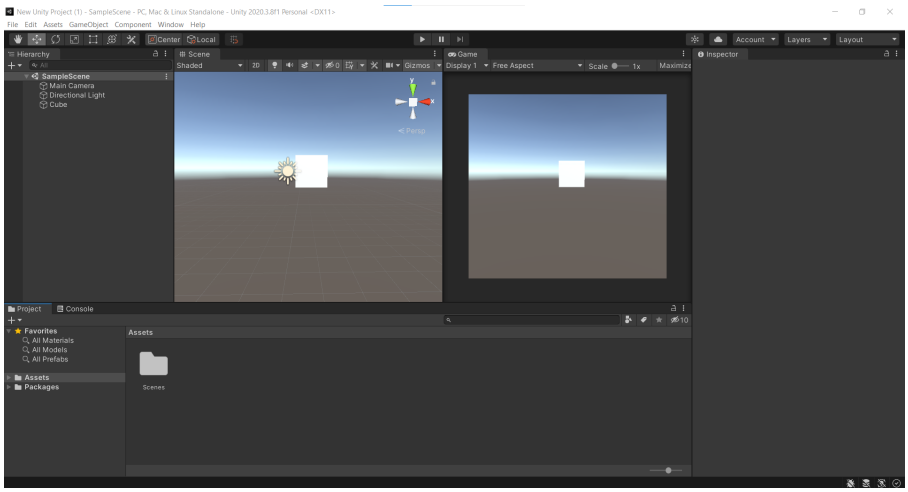


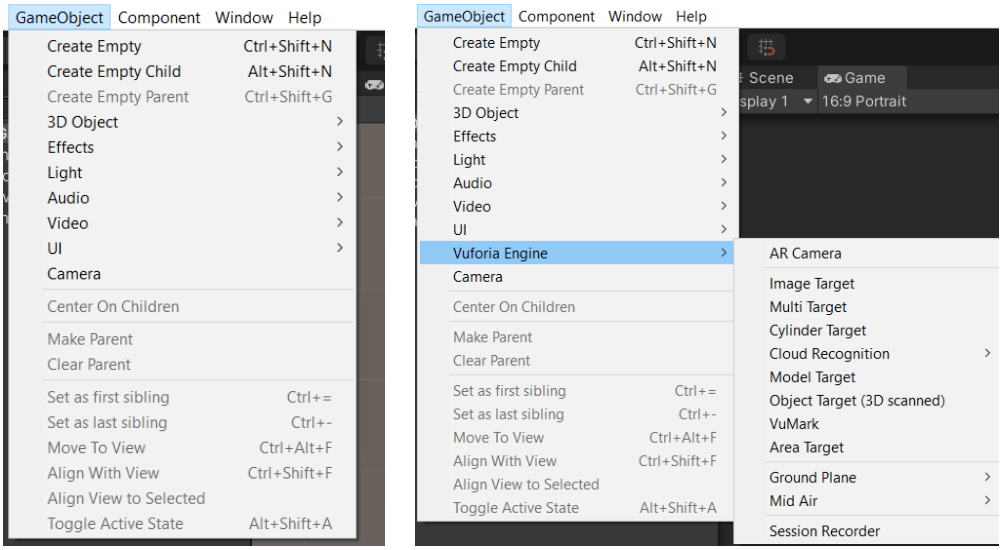
Figura 2.3: Interfaccia di Unity.

Quando si apre un nuovo progetto si apre una scena [3] in cui è presente solo una telecamera ed una luce, posso avere un numero qualsiasi di scene.

2.3 Vuforia in Unity

Una volta scelto sia il game engine che l'SDK da utilizzare bisogna vedere come integrare entrambi. Andando sul sito di Vuforia [13] è possibile scaricare il pacchetto Vuforia Engine per Unity, è disponibile anche per Android, iOS e UWP. Una volta scaricato il pacchetto bisogna importarlo nel nostro progetto Unity. In figura 2.4 possiamo vedere che una volta importato il pacchetto abbiamo delle funzionalità in più che ci permettono di realizzare diverse esperienze in realtà aumentata. La versione di Unity che ho utilizzato nel mio progetto è 2020.3.5f1 poichè quando ho iniziato ad usarlo (26 aprile 2021) era la più recente che potevo scaricare,il liguaggio di programmazione usato per gli script è il C#, invece la versione di Vuforia importata è la 9.8.5 per lo stesso motivo precendente. Infine dato che il mio cellurare è Android, questa è stata la piattaforma utilizzata.

[3] Le scene sono il luogo in cui si lavora con i contenuti in Unity. Sono risorse che contengono tutto o parte di un gioco o di un'applicazione. Possiamo sia costruire un gioco semplice in una singola scena sia uno complesso usando piu scene ognuna con i propri ambienti, personaggi, ostacoli, decorazioni e UI. [12]



(a) *Senza Vuforia Engine* (b) *Con Vuforia Engine*

Figura 2.4: GameObject di Unity: senza il pacchetto Vuforia Engine (a) e con il pacchetto Vuforia Engine (b)

L'intero progetto è stato salvato in GitLab.

2.4 Git

Git è un software per il controllo di versionamento, open source e multipiattaforma. In pratica è un programma che vi permette di salvare sia sul vostro computer che su un server più versioni di uno stesso file o progetto che si può modificare anche successivamente. Mi permette di lavorare con più persone allo stesso progetto senza doversi passare i file, utile per i backup, per il confronto tra due versioni dello stesso file, quindi le modifiche apportate. Così come gli altri software di versionamento salva i cambiamenti tra una versione e l'altra del file, ma a differenza degli altri gitfa delle foto, delle istantanee di quella particolare versione, se è cambiata rispetto alla versione precedente allora me li salva altrimenti riprende il file e lo riporta nella versione nuova. Esso è disponibile sia per Windows, sia per Mac che per Linux, per scaricarlo basta cercare su Google 'git download' e cliccare il primo risultato disponibile [14].

2.4.1 GitLab

GitLab è uno dei principali servizi di repository, ovvero servizi che mi permettono di gestire progetti che usano git.Per usarlo bisogna registrarsi gratuitamente al sito [15]. Una volta fatto il login, cliccare su New Project, dargli un nome e se si vuole una descrizione. Si può scegliere anche il grado di visibilità, privato, richiede registrazione, o pubblico, aperto a tutti. Successivamente creo, o apro, una cartella sul mio pc, click su tasto destro e selezionare Git Bash Here , copiare i comandi uno alla volta presenti sotto la voce ' Create a new repository', una volta fatto ciò avremo all'interno della nostra cartella una nuova cartella con il nome del progetto al cui interno è presente il file di testo, vuoto, denominato 'README'. Vediamo come caricare ulteriori file sul server. Per prima cosa bisogna aggiungere i file all'interno della cartella che ha il nome del progetto. Grazie al comando 'git status' posso verificare se ci sono stati dei cambiamenti tra quelli che ho in locale e quelli presenti sul server. Se ci sono delle modifiche devo usare il comando 'git add', in particolare uso 'git add -A .' questo comando dice a git di vedere tutti i file che sono stati cancellati,modificati e aggiunti all'interno della cartella. Il prossimo comando da utilizzare è 'git commit -m "messaggio che indica le modfiche apportate", questo mi permette di dire a git di prendere tutti i file che lui sta guardando, dopo il comando precedente, e chiuderli in un pacchetto al quale attacco il messaggio. Infine per mandare gli aggiornamenti al server bisogna utilizzare il comando 'git push'.

Una volta visto cosa sia l' AR e gli strumenti per implementarla possiamo vedere il progetto da me realizzato, una caccia al tesoro in AR.

Capitolo 3

AR TREASURE HUNT

La realizzazione del progetto si è diviso in due parti, la prima riguardante la realizzazione del gioco in casa e la seconda per quello fuori casa, ho realizzato, quindi, prima tutte le scene per la versione 'HOME' e successivamente quelle per la versione 'OUTDOOR'. Innanzitutto è stata creata la scena della schermata iniziale, nella quale troviamo:

- **Un Quad**, in cui ho inserito l'immagine di sfondo;
- **Due Button**, uno 'Play' per iniziare il gioco, l'altro 'Director' per aprire la scena con le istruzioni per l'organizzatore della caccia al tesoro;
- **Uno script** associato ai due Button, Play.cs, solo che usano funzioni diverse. Il pulsante 'Play' usa la funzione void OnPlay(), invece il pulsante 'Director' la funzione void OpenDirector();

La scena successiva è quella che riguarda la scelta della modalità di gioco. Qui troviamo:

- **Due Quad e un Image**, nei due quad sono state inserite due immagini, una è lo sfondo l'altra chiede di confermare la scelta.Nell' Image ,invece, c'è l'immagine di caricamento;
- **Sei Button**, i button 'Outdoor' e 'At home' servono a far comparire l'immagine per la conferma, il button 'No' mi ricarica la schermata attuale, il pulsante 'Yes', compare solo se ho scelto la versione a casa e mi

avvia il gioco, 'Yes1' invece compare solo se ho scelto la versione fuori casa e mi fa partire il gioco. Il pulsante 'Return' , invece, mi ricarica la schermata iniziale;

- **Quattro script**, ConfirmChoose.cs associato ai due pulsanti 'Outdoor' e 'At home', Play.cs associato ai due diversi pulsanti 'Yes', ma utilizzano due funzioni diverse, void OnPlay() ,per avviare la versione a casa, void OnContOut() per quella fuori casa. Invece al pulsante 'No' è associato lo script No.cs. Infine, DataManager.cs, in particolare la funzione void ClickReturn() , viene associato al pulsante 'Return';

Per realizzare la prima versione del gioco , ho importato prima la cartella delle immagini, contenente gli indizi, per trovare il marker successivo, e gli sfondi per le varie schermate. Successivamente dato che ho aggiunto anche la musica, ho importato gli audio da associare alle varie scene. Per quanto riguarda i pulsanti,invece , li ho scaricati da internet ed inseguito importati su Unity. La prima scena che ritroviamo in questa versione è quella riguardante le istruzioni da eseguire. in questa scena è presente:

- **Un Quad**, in cui ci sono le istruzioni per il gioco;
- **Un Button**, 'Continue';
- **Uno script**, Play.cs, la funzione associata al pulsante precedente è void OnPlay(), va avanti nella scena successiva;

La prima versione è composta da sei scene indizio. In tali scene troviamo:

- **Due Quad**, in cui ho inserito l'immagine dell'indizio, e la richiesta di conferma;
- **Tre Button**, uno per continuare,'Continue', uno per confermare, 'Yes', e l'altro per tornare ricaricare la schermata corrente, 'No';
- **Due AudioSource**, uno in cui è presente un audio di sottofondo con il piano e l'altro per l'applauso;

- **Tre Script** ognuno per ogni pulsante. ConfirmChoose.cs è associato a 'Continue', fa comparire la scritta di conferma insieme ai due button 'Yes' e 'No'. yes.cs è associato a 'Yes' e mi carica la scena successiva. no.cs invece è associato a 'No' e mi ricarica la schermata attuale;

Il giocatore deve decifrarli per arrivare al prossimo marker, ma avere l'indizio non è semplice in quanto bisogna dapprima risolvere un piccolo gioco in realtà aumentata, in cui si deve indovinare la giusta sequenza di pulsanti. Ci sono due versioni di questo gioco, due scene in cui sono presente tre pulsanti ed altre tre scene in cui sono presenti quattro pulsati. Nelle scene in cui sono presenti tre pulsanti ritroviamo:

- **ARCamera** al posto della MainCamera, per utilizzare Vuforia bisogna utilizzare questo tipo di Camera;
- **ImageTarget**, in cui metto il Marker, ed al suo interno quello che voglio far comparire in realtà aumentata, in questo caso i tre 'Button';
- **Tre Button**, ognuno di colore diverso per distinguerli tra loro;
- **Tre Script** per i Button, ognuno serve a definire la sequenza, quindi l'ordine con cui deve essere premuto. Al primo associo button13.cs, al secondo button23.cs ed al terzo button33.cs;
- **Due Text**, compaiono solo a fine sequenza. Uno mi indica che devo riprovare, l'altro che ho indovinato il primo pulsante della sequenza;
- **Tre AudioSource**, associati ai pulsanti, quando questi vengono cliccati fa un suono;

Quella con quattro pulsanti è simile alla precedente in cui troviamo:

- **Un pulsante in più**
- **Un text in più**, questo mi dice se ho indovinato i primi due pulsanti della sequenza;

- **Quattro script** associati ai pulsanti che sono diversi da quelli precedenti. Per il primo abbiamo Button1.cs, per il secondo Button2.cs, per il terzo Button3.cs e per il quarto Button4.cs;
- **Un AudioSource in più**, ha la stessa funzioni degli altri;

L'ultimo gioco in realtà aumentata è diverso da quelli precedenti, infatti attraverso le animazioni, muovo il pulsante a destra e sinistra. Quindi prima ci sono nuove regole, e la scena presenta:

- **Due Quad**, uno per le istruzioni nuove e l'altro per confermare la scelta di continuare. Dato che il Quad è dotato di Audiosource ,il suono è stato inserito direttamente al suo interno, in questo caso nel primo quad, quello in cui ci sono le istruzioni ;
- **Tre button**, 'Continue', 'yes' e 'No';
- **Tre script**, ConfirmChoose.cs, che mi fa apparire il quad che chiede la conferna e i due button 'Yes' e 'No', al primo dei due è associato lo script yes.cs ed utilizza la funzione void OnContinue2(), mentre al secondo è associato lo script No.cs;

Nell'ultimo gioco, essendo in realtà aumentata, ritroviamo l'ARCamera al posto della MainCamera. In questa scena ritroviamo:

- **Otto Button** di quattro colori diversi, ognuno con la sua animazione;
- **Due script**, Wrong.cs che mi attiva il suono che ho sbagliato,Yes.cs che mi porta nella scena successiva;
- **Otto animazioni** che modificano la PosX del Rect Trasform del pulsante;

Ho realizzato, infine, la scena del tesoro , comune ad entrambe le versioni, in cui ho scaricato ed importato dall'Asset store sia il forziere che i fuochi d'artificio. In tale scena troviamo:

- **Due text**, uno mi invita ad aprire il forziere, l'altro mi indica che il gioco è stato completato attraverso la scritta vittoria.
- **Due forzieri**, chest,scaricati dall'Asset Store di Unity, uno aperto e uno chiuso
- **Le monete**, coins, scaricati insieme ai forzieri;
- **I Fuochi d'artifico**, fireworks, scaricati sempre dall'Asset Store di Unity;
- **Due Button**, 'replay' e 'exit', il primo per rigiocare il secondo per uscire dal gioco;
- **Due script** per i button citati sopra, entrambi usano lo script 'Play';
- **Due script**, uno, openchest.cs, per far comparire il forziere aperto, con monete, i fuoci d'artificio , il suono della tromba, la scritta victory ed i due button sopra citati, invece l'altro script, coins.cs, serve a far scoparire monete e fuochi d'artificio se il marker non è più tracciato;

Una volta appurato che questa versione funzionasse ho realizzato le scene per la versione 'OUTDOOR', ho importato nuove immagini, regole nuove e la freccia per indicare la direzione da seguire per ritrovare il marker, e sono state messe sempre nella stessa cartella utilizzata per la versione precedente. Come per la scena delle istruzioni precedenti ritroviamo:

- **Un Quad**, in cui è presente l'immagine delle istruzioni;
- **Un Button**, 'Continue';
- **Uno script**, associato al button precedente, Play.cs, in particolare per andare avanti nelle scene viene utilizzata la funzione void OnContOut();

Oltre alle istruzioni del gioco troviamo due tipi di scene, una scannerizza il marker mentre l'altra ti indica dove trovare il marker, attraverso una freccia e la distanza che c'è tra te ed il marker. Nelle prime troviamo:

- **ARCamera** al posto della MainCamera, dato che usiamo Vuforia;
- **Un Text**, in cui si invita ad inquadrare il marker;
- **ImageTarget**, in cui viene inserito il Marker;
- **Un Image**, presente nell'ImageTarget, che dice al player di star facendo un buon lavoro, trovando i marker;
- **Due Button**, uno, 'Return', per tornare alla schermata precedente l'altro, 'Continue' che è inserito all'interno dell'ImageTarget, per proseguire il gioco;
- **Uno Script**, Play.cs, associato ad entrambi i button, solo che il button 'Return' utilizza la funzione void Onreturn(), ed il button 'Continue' void OnContOut();

Nelle seconde invece troviamo:

- **Due Image**, una contenente la freccia, l'altra chiede conferma della scelta fatta;
- **Due Text**, uno per indicare la distanza che mi separa dal marker, l'altro invece indica l'accuratezza con cui viene calcolata;
- **Tre Button**, 'Scan', 'Yes' e 'No'
- **Uno script** associato ai tre button. Scan utilizza la funzione void OnConfirm(), per far apparire l'immagine che chiede la conferma e gli altri due button. No, invece, utilizza void OnNo() per farli scomparire. Infine, Yes, utilizza la funzione void OnYes() per andare nella scena successiva in cui c'è Vuforia;
- **Uno script** ,UpdateText.cs, associato ai due Text, serve ad aggiornare la distanza e la sua accuratezza;
- **Uno script**, Compass.cs, associato all'Image della freccia. Mi indica che direzione seguire;

- **Due script**, uno che calcola la distanza che mi divide dal marker, Distance.cs, e l'altro GPS.cs, invece calcola la posizione GPS in real time del mio dispositivo;

Infine ho realizzato delle scene per le istruzioni per chi organizza la caccia al tesoro. Nella prima scena che riguarda il Director troviamo:

- **Due text**, uno mi chiede dove si vuole organizzare il gioco, nell'altro c'è semplicemente scritto 'or';
- **Un Image** che è l'immagine di caricamento;
- **Tre Button** , 'At home' , 'Outdoor' e 'return';
- **Due script**, Play.cs associato ai pulsanti 'At home' e 'Outdoor', in particolare il primo utilizza la funzione void OpenUrlOnBrowser() che lo porta al mio GitLab, in cui sono presenti tutte le istruzioni ed i marker da utilizzare, il secondo invece void OnPlay() che lo porta alla scena successiva. L'altro script è DataManager.cs ed è associato al pulsante 'return' che utilizza la funzione void Onreturn();

Nel caso si scelga la versione outdoor abbiamo una nuova scena in cui è presente:

- **Un Text**, che mi invita a leggere le istruzioni prima di inserire le coordinate GPS dei marker che bisognerà cercare;
- **Tre Button**, 'Instructions', 'Coordinates', 'Return';
- **Due script**, Play.cs associato ai pulsanti 'Instructions' e 'Coordinates', in particolare il primo utilizza la funzione void OpenUrlOnBrowser() che lo porta al mio GitLab, il secondo invece void OnPlay() che lo porta alla scena successiva. L'altro script è DataManager.cs ed è associato al pulsante 'return' che utilizza la funzione void Onreturn();
- **Un Image** che è l'immagine di caricamento;

Nell'ultima scena dedicata al 'Director' troviamo la scena in cui egli puo inserire le coordinate GPS dei marker da trovare. In questa scena troviamo:

- **Quattordici InputField**, sette per inserire la latitudine ed altri sette per inserire la longitudine;
- **Sette Text**, in due di questi c'è l'attuale latitudine e longitudine del dispositivo, in modo da facilitare l'inserimento delle coordinate GPS del marker, sopra questi due c'è il testo che mi specifica che è l'attuale posizione, altri due text mi indicano a chi appartengono la latitudine e longitudine, ad esempio Lat1, indica la latitudine del primo marker. Uno dei testi mi invita ad inserire le coordinate, ed il testo di sotto mi indica a quale marker mi riferisco, il primo, il secondo etc...;
- **Un Image** che è l'immagine di caricamento;
- **Due Button**, 'Save' e 'Return';
- **Due script**, lo script UIPlace.cs, associato ai due button, il pulsante 'Save' usa la funzione void ClickSave(), invece, il pulsante 'Return' la funzione void ClickReturns(). Lo script DataManager.cs viene utilizzato dallo script precedente per creare un file di salvataggio '.txt', salvare i dati all'interno di essi e caricarli quando richiesto;

Vediamo ora nel dettaglio come è stato realizzato e come funziona l'applicazione.

3.1 Inizio Progetto

Volendo realizzare un app Android bisogna fare lo 'switch platform' su Unity, dato che di default è impostato su PC, cliccando in alto a sinistra su **file->build setting** ,scegliere la piattaforma Android e poi cliccare su 'switch platform',Fig 3.1.

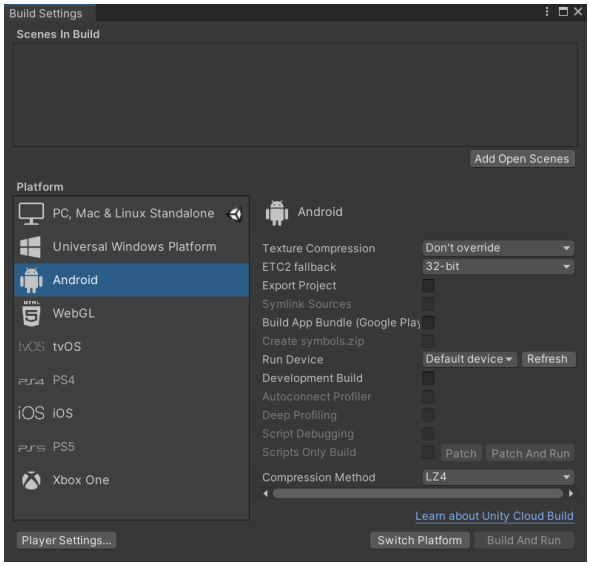


Figura 3.1: Switch platform

Il progetto conta 34 scene, alcune sono per l'organizzatore della caccia al tesoro, che ho chiamato "Director" ed altre sono per i giocatori. Il Director oltre al posizionamento dei vari marker, nel caso venga scelta la versione del gioco fuori casa deve inserire anche le coordinate GPS dei vari marker.
Come ogni caccia al tesoro, l'obiettivo è quello di trovare il tesoro attraverso degli indizi. In questo gioco gli indizi mi portano a trovare i vari marker fino ad arrivare proprio al tesoro. Gli indizi forniti sono dati in maniera differente a seconda di dove scegliamo di giocare, questo però sarà più chiaro quando verranno spiegate le due versioni del gioco, 'Outdoor' e 'At Home', rispettivamente fuori casa ed in casa. Appena apro l'applicazione devo dare il permesso per utilizzare la fotocamera ed il GPS, quest'ultimo soprattutto se

voglio giocare fuori casa. Vediamo ora nelle specifico alcune scene partendo da quelle di gioco.

3.2 Schermata di Avvio

La scena iniziale è composta da due 'button', un'immagine di sfondo ed un suono. Il pulsante 'Play' mi permette di avviare il gioco, invece il pulsante 'The Director' mi apre una schermata che permette di ricevere le informazioni necessarie all'organizzatore per pianificare la caccia al tesoro. Quello che fanno i due pulsanti altro non è che cambiare scena attraverso uno script che ho chiamato Play. Un esempio di comando all'interno dello script è possibile vederlo in figura 3.2.

```
SceneManager.LoadScene("choose");
```

(a) *comando per il tasto Play*

```
SceneManager.LoadScene("Director");
```

(b) *Comando per il tasto The Director*

Figura 3.2: Script per cambiare scena

Nella prossima figura possiamo vedere come si presenta l'App una volta avviata.

Figura 3.3: schermata iniziale

Infine è stato aggiunto un suono, questo è possibile aggiungendo alla scena un GameObject chiamato AudioSource.

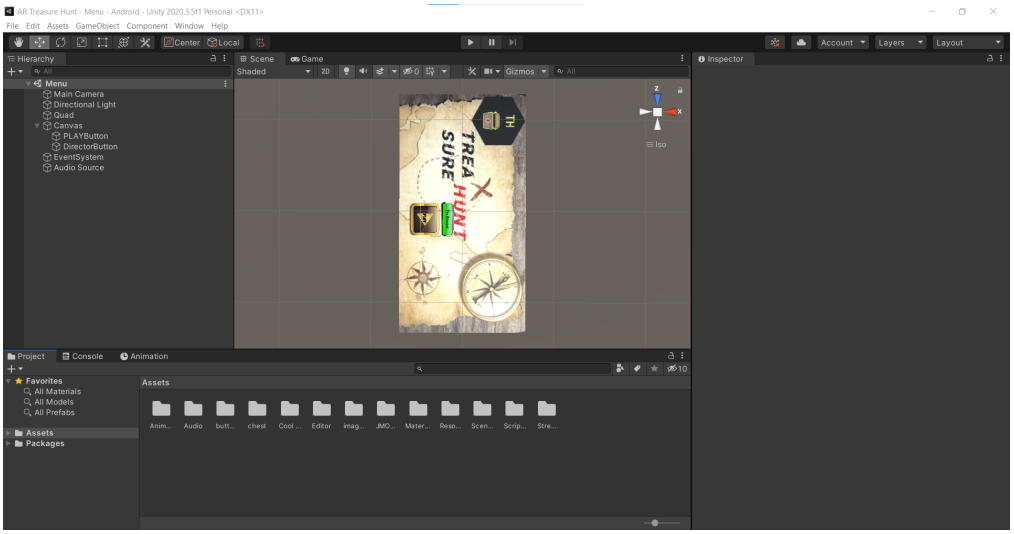


Figura 3.4: scena iniziale in unity

In figura 3.4, sulla sinistra, è possibile vedere i GameObject che sono stati inseriti per realizzare tale scena che sono stati menzionati precedentemente. Il GameObject 'Quad' è un quadrato al quale ho cambiato le dimensioni e successivamente assegnata la foto della schermata iniziale, vedi figura 3.3, senza i due pulsanti 'Play' e 'The Director'.

Anche se in realtà è possibile inserire direttamente l'immagine cliccando, in alto, **GameObject->UI->Image**.
Il Canvas è dove vanno inseriti gli elementi dell'interfaccia utente, in questo caso i due button possono essere inseriti cliccando **GameObject->UI->Button**.
L'EventSystem, compare quando viene inserito il canvas quindi gli elementi UI, ed è responsabile dell'elaborazione e della gestione degli eventi in una scena Unity. AudioSource è utilizzato per inserire un audio.
Vediamo ora cosa accade una volta cliccato 'Play' e seguiamo il gioco cosi come lo vede un giocatore.

3.3 Scelta del luogo di gioco

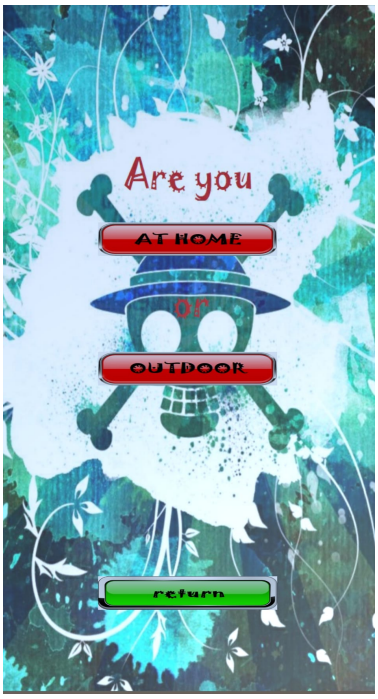


Figura 3.5: Schermata per la scelta del luogo in cui giocare

Dopo avere cliccato il pulsante per avviare il gioco ci troviamo in una nuova scena, che ho chiamato choose. In questa scena il giocatore può scegliere se giocare a casa o fuori casa, a seconda della scelta il gioco sarà differente, come vedremo più avanti. Questa scena è molto simile alla precedente, in quanto come prima troviamo dei pulsanti ed un immagine. In realtà però sono presenti alcuni GameObject che non si vedono se non accade un particolare evento, premere su uno dei due pulsanti, 'At home' o 'Outdoor'. Questo mi permette di instanziare oggetti anche successivamente all'avvio della scena. In questo caso appare un immagine, che mi chiede conferma della mia scelta, e due 'button', per confermare o meno la scelta, vedi Figura 3.6. Nel primo caso utilizzando lo stesso script precedente, Play, vado in una nuova scena a seconda se io abbia scelto di giocare a casa o fuori, invece nel secondo caso viene ricaricata la scena in cui mi trovo, in questo caso choose, dove l'immagine e i due pulsanti sono resi nuovamente invisibili.

Figura 3.6: Conferma la scelta

Vediamo nella figura 3.7 come avere un GameObject non attivo, quindi non presente in scena.

(a) *GameObject attivo* (b) *GameObject inattivo*

Figura 3.7: Come impostare un Game Object inattivo nell'interfaccia di unity

Quindi bisogna togliere la spunta se voglio togliere il GameObject dalla scena. Attraverso script però è possibile rifarlo comparire. Nel mio caso ai pulsanti 'At home' e 'Outdoor' è stato assegnato lo script 'Confirmchoose' ed il comando che mi permette di mostrare i GameObject è il seguente:

```
images.SetActive(true);
yes.SetActive(true);
no.SetActive(true);
```

Figura 3.8: Comando per attivare un GameObject

Nel caso io voglia disattivarlo, attraverso script, dovrò inserire false al posto di true. Prima di questo comando bisogna dichiarare le variabili GameObject images, yes e no. Dobbiamo dichiararle public in modo da poter interagire con esse Vedi Figura 3.9.

```
public GameObject images;
public GameObject yes;
public GameObject no;
```

Figura 3.9: Dichiarazione delle variabili GameObject

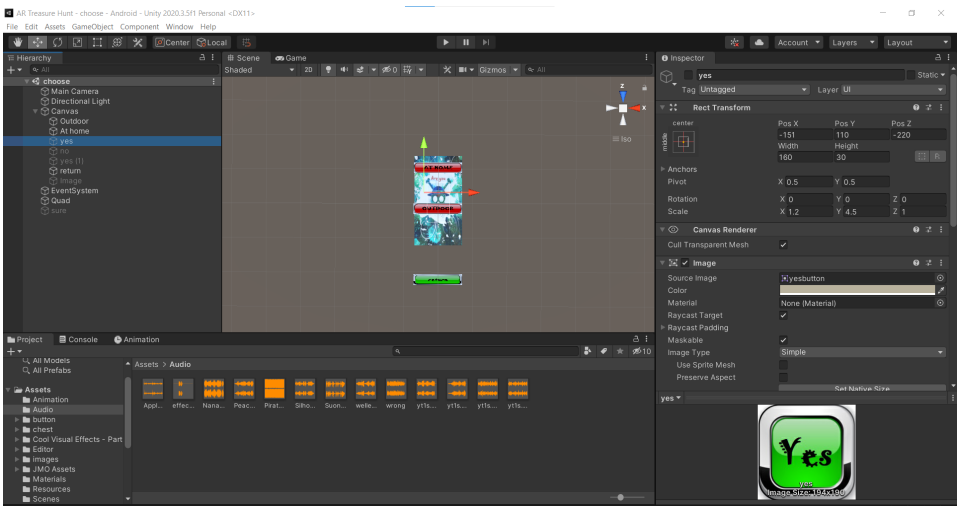


Figura 3.10: Ecco come appare la scena 'choose' in unity

In figura 3.10, possiamo vedere come, sulla sinistra, le scritte dei GameObject disattivati non siano evidenziati in bianco. Come già detto, quelli che non compaiono sono i due pulsanti 'Yes' e 'No', il Quad in cui è inserita l'immagine, qui chiamata 'sure' , dove c'è la scritta che mi chiede se sono sicuro della scelta da me fatta. Qui però non è presente solo un pulsante yes ma bensi due, infatti uno compare se clicco 'At home' mentre l'altro se clicco 'Outdoor', in modo da condurmi nelle due scene diverse. Infine troviamo un GameObject Image, non il Quad, in questa immagine troviamo la scritta loading con un simbolo che indica il caricamento, questa appare quando clicco il pulsante 'return' che mi porta nella scena iniziale. Ho

inserito questa immagine,tra i due pulsanti per la scelta del luogo, poichè in alcuni casi per cambiare scena può richiedere qualche secondo in più e quindi viene segnalato al giocatore che il pulsante è stato premuto correttamente e che si sta caricando la scena, vedi Figura 3.11.

Figura 3.11: Caricamento scena

3.4 At Home

Vediamo prima il gioco nella versione 'At home'. Quindi abbiamo cliccato sull'omonimo pulsate e confermato la scelta per trovarci nella nuova scena, 'Rules'.

3.4.1 Regole

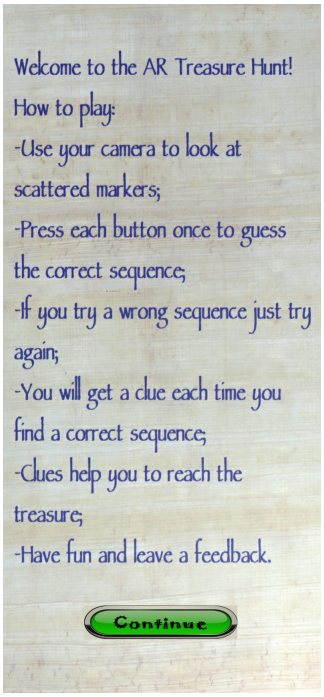


Figura 3.12: Regole per il gioco versione 'At Home'

In questa scena vengono spiegate le regole per giocare, in sottofondo c'è una canzone. Le regole sono abbastanza semplici, una volta inquadrato il marker, appaiono dei pulsanti, bisogna cliccare i pulsanti ed avremo l'indizio solo se indovineremo la sequenza esatta, altrimenti dovremo ripetere, finchè non la troviamo, ricordando che ogni pulsante può essere premuto solo una volta e devono essere premuti tutti.

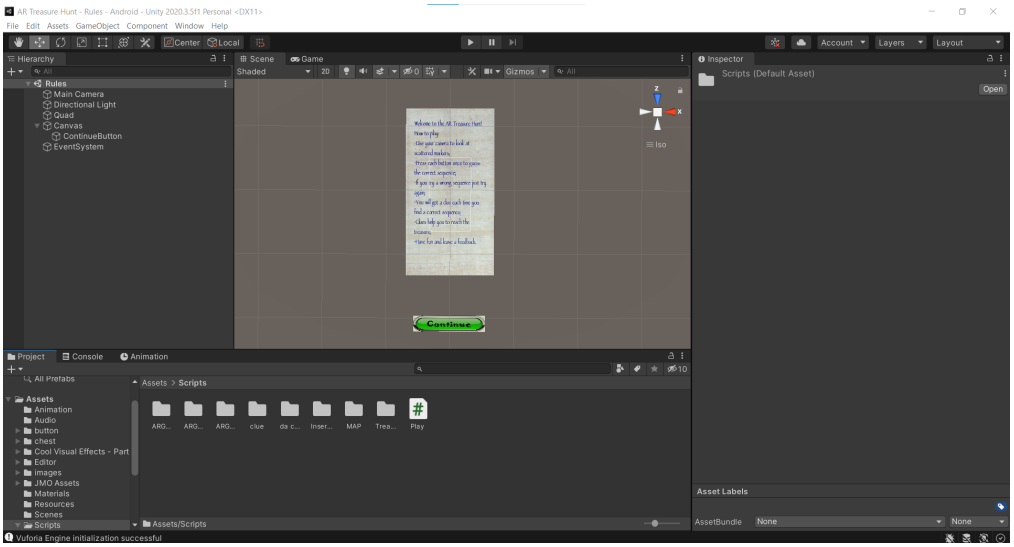
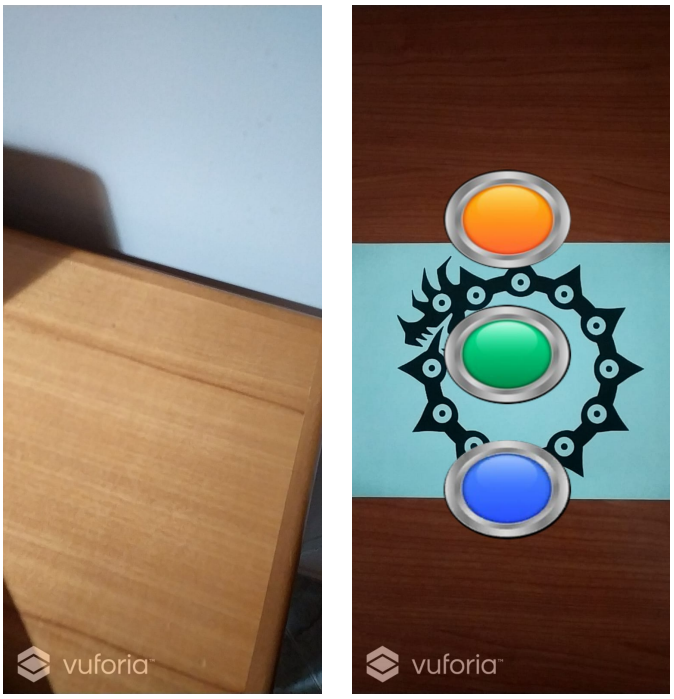


Figura 3.13: Regole viste da Unity

In figura 3.13, vediamo che per creare la scena è stato utilizzato solamente un Quad ed un button. Nel Quad è stata inserita l'immagine in cui sono scritte le regole e, visto che ha un proprio AudioSource, è stato inserito anche l'audio. Il pulsante continua semplicemente cambia scena, mi apre la prima scena con vuforia che dopo aver inquadrato il marker mi mostra pulsanti AR.

3.4.2 AR Game



(a) *Visuale Vuforia senza Marker*

(b) *Visuale Vuforia dopo aver trovato il Marker*

Figura 3.14: Scena AR Game con (a) e senza (b) marker

Una volta cliccato su continua si avvia il gioco vero e proprio, aprendosi Vuforia, vedi Figura 3.14(a). Come accenato prima una volta inquadrato il marker appaiono i pulsanti, vedi Figura 3.14(b), e devo indovinare la sequenza, il click dei pulsanti è accompagnato da un effetto sonoro. Quando sbagli la sequenza oltre alla scritta 'Try again', se hai indovinato il primo pulsante questo viene segnalato con la scritta 'Guessed the first', il tutto è abbinato ad un suono che mi indica l'errore, vedi Figura 3.15. Se invece indovino vado in una nuova scena in cui è presente l'indizio, in cui è stato aggiunto un breve effetto sonoro dell'applauso. Sono presenti due scene di questo tipo. Una volta trovato il terzo marker ed avviato l'AR Game, sono presenti quattro pulsanti e non piu tre. Quindi viene aumentata la difficoltà, ma in caso di errore mi viene indicato anche se indovino i primi due pulsanti della sequenza, per il resto è uguale al precendente,vedi Figura 3.15. Sono state inserite tre scene di questo tipo.

(a) *Nessun pulsante indovinato*

(b) *Indovinato solo il primo pulsante*

Figura 3.15: AR Game a tre pulsanti

(c) *Indovinato il primo*

(d) *Indovinati i primi due*

Figura 3.15: AR Game a quattro pulsanti

Vediamo ora come si mostra in Unity:

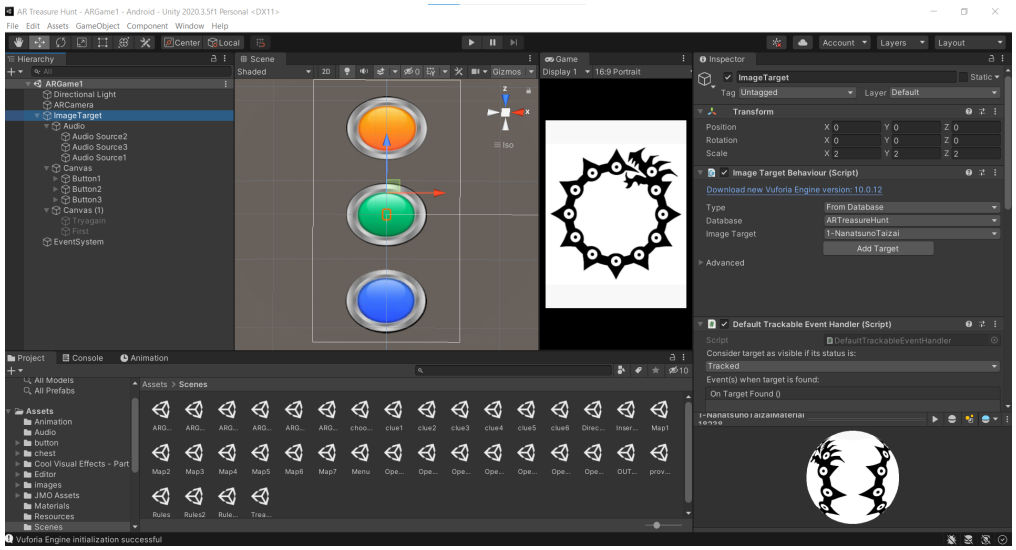


Figura 3.16: AR Game a tre pulsanti visto in 'Unity'

La prima differenza con le scene precedenti è la mancanza di una MainCamera, sostituita dall'AR Camera, GameObject presente in Unity solo dopo aver importato Vuforia. Un altro elemento di Vuforia presente nella scena è l'ImageTargets in cui si inserisce uno dei marker presenti nel database che si creato sul sito Vuforia e importato in Unity. Una volta inquadrato questo marker avrò la mia scena AR, in questo caso l'inserimento di tre 'button', i suoni relativi ai pulsanti ed due GameObject Text, uno che diventa attivo ogni volta che sbaglio, invitando il giocatore a ritentare, l'altro presente solo se il primo pulsante cliccato è quello esatto, dicendo appunto che il primo pulsante della sequenza è stato indovinato. Ad ogni pulsante è associato uno script a seconda che questo debba essere il primo, il secondo o il terzo. Per verificare la giusta sequenza sono state dichiarate due variabili pubbliche di tipo bool , una per lo script del button1,b13, e l'altra per lo script del button2,b23. Quella associata al primo pulsante da premere diventa true solo se l'altra è false ed il numero di click è pari a 0, quella associata al secondo, diventa true solo se il primo è true ed il numero di click è uguale a uno, il contatore parte da zero. Per ogni sequenza,ogni tre click e quindi numero di click pari a due, il numero di click viene azzerato, nel caso

che venga sbagliata la sequenza tutte le variabili booleane ridiventano false, se invece è quella corretta, l'ultimo pulsante mi porta al cambio di scena. Per tenere traccia che è stato indovinato il primo pulsane della sequenza ho dichiarato un'altra variabile booleana pubblica,ind13, essa diventa true quando diventa true la variabile b13 ma mentre questa se sbaglio il secondo pulsante diventa false l'altra resta true fino a quando non viene segnalato l'errore della mia sequenza. Cliccando il terzo pulsante in caso di errore attivo anche i Text 'TryAgain' e 'First', il primo ogni volta che sbaglio, il secondo solo se ind 13 è true. In base a quale pulsante associo gli script posso cambiare la sequenza.

Molto simile è la scena con quattro pulsanti, bisogna aggiungere un nuovo button nel canvas, ma gli script sono diversi, vengono aggiunte altre due variabili booleane una che diventa true quando sono stati indovinati i primi tre pulsanti, l'altra tiene traccia che sono stati indovinati i primi due pulsanti. In oltre è presente un nuovo Text che mi segnale che ho indovinato i primi due pulsanti della sequenza, vedi Figura 3.17.

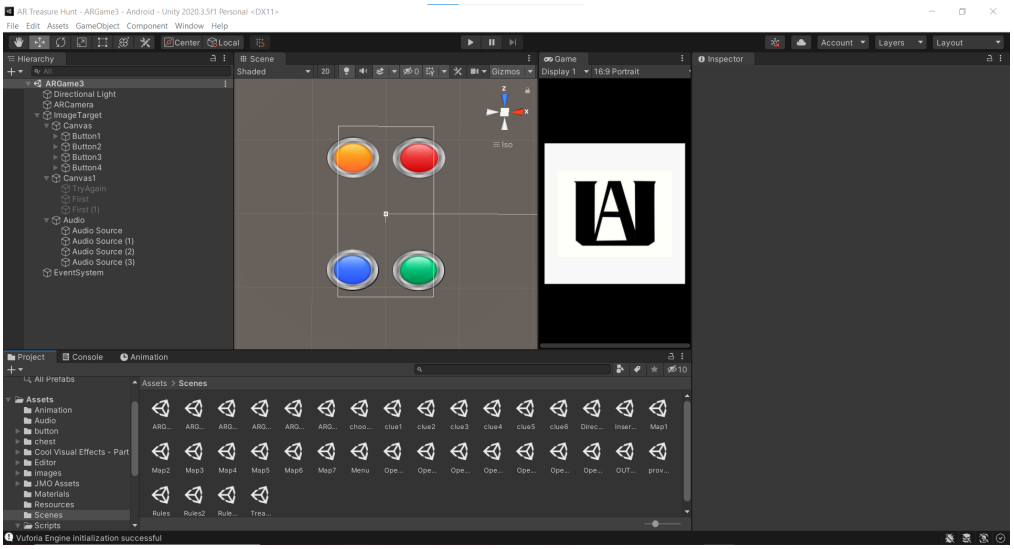
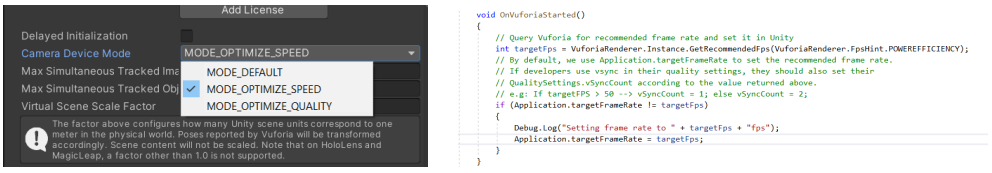


Figura 3.17: AR Game a quattro pulsanti visto in 'Unity'

Nel tentativo di ridurre i consumi quando viene utilizzato Vuforia, quindi eccessivo surriscaldamento, nell'inspector dell' ARCamera , ho aperto 'Vuforia Configuration' e settato 'MODE_OPTIMIZE SPEED' in 'Camera Device Mode', in modo da minimizzare l'impatto di Vuforia sul sistema. Una volta impostata la modalità di prestazione, un ulteriore passo è quello di limitare gli FPS, quindi imposto un framerate piu basso. Per fare ciò, allo script per il tracciamento, DefaultTrackableEventHandler.cs, è stata aggiunta una funzione ,void OnVuforiaStarted(), ed è stato scelto il flag, FPSHINT_POWER_EFFICIENCY, che mi permette di ridurre l'accumulo di calore e aumentare la durata della batteria, anche se a spese delle prestazioni dell'applicazione.



(a) *Impostazione della modalità delle prestazioni in Unity*

(b) *Impostazione del framerate in Unity*

Figura 3.18: Framerate e ottimizzazione delle prestazioni

3.4.3 Indizi

Quando indovino la sequenza esatta viene caricato un nuovo indizio. Dopo il breve applauso iniziale parte una musica di sottofondo suonata al piano. Scena molto simile a quella delle regole. L'indizio indica l'ubicazione del prossimo marker, il giocatore deve risolverlo per continuare il gioco.

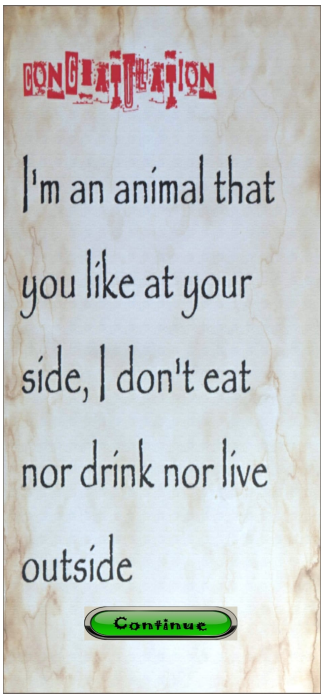
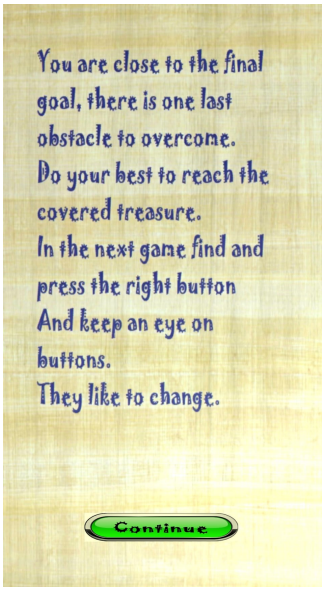


Figura 3.19: Un'indizio per trovare il prossimo marker nella versione 'At Home'

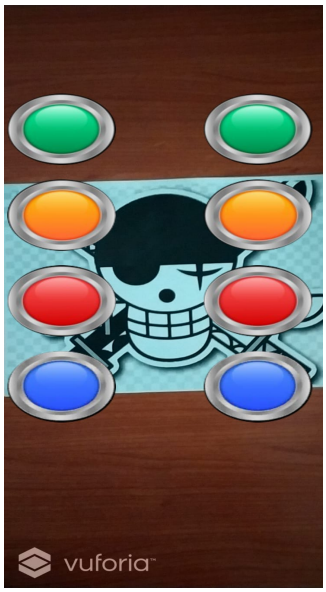
In Unity la scena è uguale a quella già vista per per regole, cambia solo l'immagine inserita nel Quad.

ARGame2

L'ultimo AR Game è diverso dagli altri, abbiamo quattro pulsati a sinistra e quattro a destra, e si muovono in direzioni opposte, quelli di destra verso sinistra e tornano indietro, viceversa quelli di sinistra verso destra e tornano indietro. Per superare questo gioco ed arrivare all'ultimo indizio, bisogna cliccare solo sul pulsante corretto, se clicchi quello sbagliato ci sarà un effetto sonoro che te lo fa capire. Prima di questo gioco però sono presenti delle nuove istruzioni, per preparare il giocatore a questo nuovo gioco.



(a) *nuove regole*



(b) *nuovo gioco*

Figura 3.20: ultimo AR Game

Dopo è presente la scena finale con il tesoro, ma ne parlerò in seguito in quanto comune ad entrambe le modalità di gioco, sia 'At home' che 'Outdoor'.

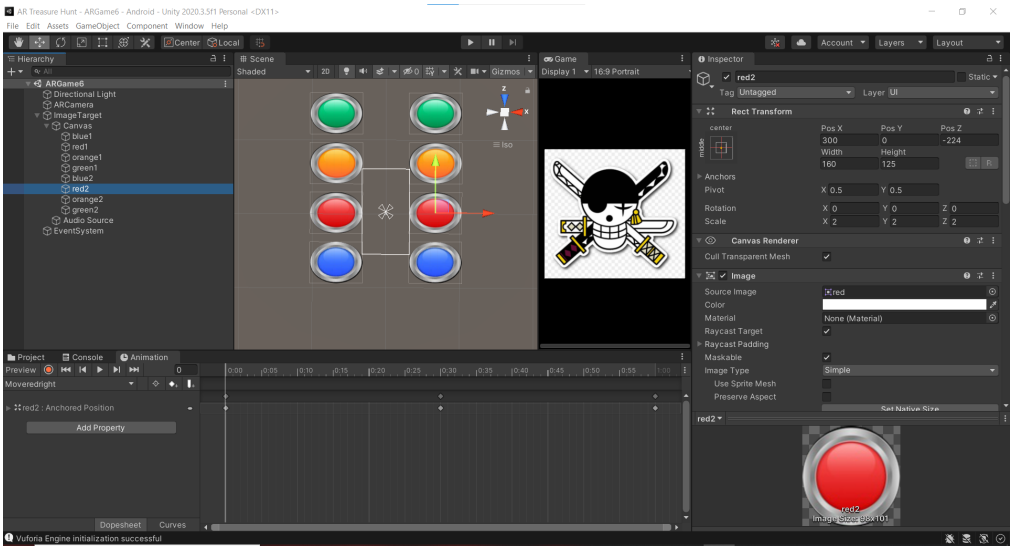


Figura 3.21: AR Game2 pulsanti mobili visto in 'Unity'

In figura 3.21, posso vedere l'ARGame2 visto in Unity. In Unity è possibile animare i GameObject anche senza script con l'Animation.

Per capire come funzionino le animanizioni base aprire il link `https://www.youtube.com/watch?v=RCakXN3RAFM&t=2s`.

Mentre nel video modifica la rotazione, qui viene modificato Position del Rect Tranform del button, in particolare PosX. Solo ad un pulsante viene dato lo script per passare alla scena successiva "SceneManager.LoadScene("clue6")" gli altri attivano solo il suono che indica l'errore.

3.5 Outdoor

Ora vediamo la versione 'Outdoor', quindi ho cliccato l'omonimo pulsante nella schermata che mi chiede dove voglio giocare. Una volta data la conferma si apre la schermate delle regole.

3.5.1 Regole

Queste sono diverse rispetto alla versione vista precedentemente. Qui infatti bisogna avere il GPS attivo e seguire le indicazioni date dalla freccia per raggiungere il prossimo marker, e ridurre la distanza che separa il giocatore dal marker, vedi Figura 3.22

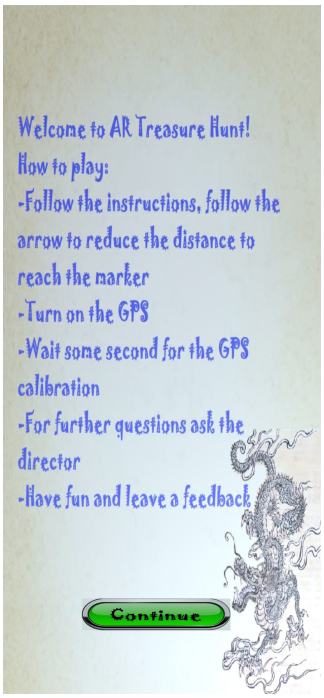


Figura 3.22: Regole per la versione 'Outdoor'

3.5.2 Scan Marker

(a) *Prima di inquadrare il marker*

(b) *dopo aver inquadrato il marker*

Figura 3.23: ultimo AR Game

Una volta lette e capite le istruzioni, si clicca il pulsante 'Continue' che mi apre l'AR Camera di Vuforia. In rosso è segnata l'istruzione da seguire 'Frame the marker, quindi bisogna inquadrare quest'ultimo per accedere alla scena successiva. Troviamo anche il pulsante 'return' che mi riporta nella scherma della scelta, Figura 3.5. Nell'app possiamo trovare 7 scene simili, con l'unica differenza, oltre al marker, che il tasto return non mi apre sempre la stessa scena ma quella che mi permette di localizzare il marker in questione, vedi sottoparagrafo successivo.

Ecco come si presenta in unity:

Figura 3.24: Scan Marker visto in unity

3.5.3 Localizzazione Marker

Figura 3.25: Indicazioni per ritrovare il marker

Una volta trovato il marker giusto, si apre la scena che mi da informazioni sulla distanza tra il giocatore ed il marker, ed anche in che direzione esso debba muoversi attraverso la freccia al centro, che è in grado di ruotare in base alla direzione da prendere.

In questa scena è presente:

- Lo script 'Compass', originariamente utilizzato come da nome come bussola, poi è stato modificato affinchè invece di puntare al Nord, punti verso la direzione del marker. Quindi viene aggiunto un angolo alla direzione per il Nord, questo viene calcolato nello script 'Distance';

- Lo script 'Distance', calcola la distanza e la direzione tra il player e la posizione del marker, questa è inserita da quello che io ho chiamato 'Director'.

  Per calcolare la distanza è stato utilizzata la seguente formula:

  $d(A,B) = R * arccos(sin(latA) * sin(latB) + cos(latA) * cos(latB) * cos(lonA - lonB))$

  Invece per la direzione sono state utilizzate le seguenti equazione:

  $\Delta\phi = ln(tan(latB/2 + \pi/4)/tan(latA/2 + \pi/4))$

  $\Delta lon = abs(lonA - lonB)$

  direzione : $\theta = atan2(\Delta lon, \Delta\phi)$

  In figura 3.26 è possibile vedere come queste equazioni vengono tramutate in linguaggio C# all'interno dello script 'Distance'.

```
/* Defines constants and variables */
const double R = 6371;
latA = GPS.Instance.latitude;
lonA = GPS.Instance.longitude;

double lat_alfa, lat_beta;
double lon_alfa, lon_beta;
double deltalon,deltaphi,theta,deg;
double p,d1;
/* Converts degrees to radians */
lat_alfa = Math.PI * latA / 180; // Player latitude
lat_beta = Math.PI * latB / 180; // Marker latitude
lon_alfa = Math.PI * lonA / 180; // Player longitude
lon_beta = Math.PI * lonB / 180; // Marker longitude
/* Calcola l'angolo compreso fi */
deltalon = lon_alfa - lon_beta;
/* Calculate the third side of the spherical triangle*/
p = Math.Acos(Math.Sin(lat_beta) * Math.Sin(lat_alfa) +
  Math.Cos(lat_beta) * Math.Cos(lat_alfa) * Math.Cos(deltalon));
/* Calculate the distance on the surface
raggio terrestre R = ~6371 km */
d1 = p * R * 1000; // distance in meters
d =(int) d1:// to get integer
```

(a) *comandi per trovare la distanza*

```
deltaphi = Math.Log(Math.Tan(lat_beta / 2 + Math.PI / 4) /
           Math.Tan(lat_alfa / 2 + Math.PI / 4));
theta = Math.Atan2(deltalon,deltaphi);
deg = theta * 180 / Math.PI;
if (deg < 0)
{
    deg = -deg;
}
else
{
    deg = 360 - deg;
}
degrees = (float) deg;
```

(b) *comandi per trovare la direzione*

Figura 3.26: Distance Script

Questi sono inseriti all'interno del metodo Update{}[1], dato che la posizione deve essere aggiornata di frequente dato che il giocatore si muove, e quindi cambia costantemente.

- Lo script 'GPS', questo mi da la posizione del giocatore aggiornata continuamente ed anche l'accuratezza della rilevazione.

```
private void Update()
{
    Input.location.Start(0.5f,1f);//Starts location service updates
    //Input.location.Start(float desiredAccuracyInMeters, float updateDistanceInMeters);
    latitude = Input.location.lastData.latitude; //Last measured device latitude.
    longitude = Input.location.lastData.longitude;//Last measured device longitude.
    accuracy = Input.location.lastData.horizontalAccuracy;//Horizontal accuracy of the location.
}
```

Figura 3.27: Comandi per trovare il GPS del proprio device

All'interno della scena troviamo anche il pulsante 'Scan' che serve ad aprire vuforia ed inquadrare il marker,vedi paragrafo 3.5.2.
In Unity si presenta così:

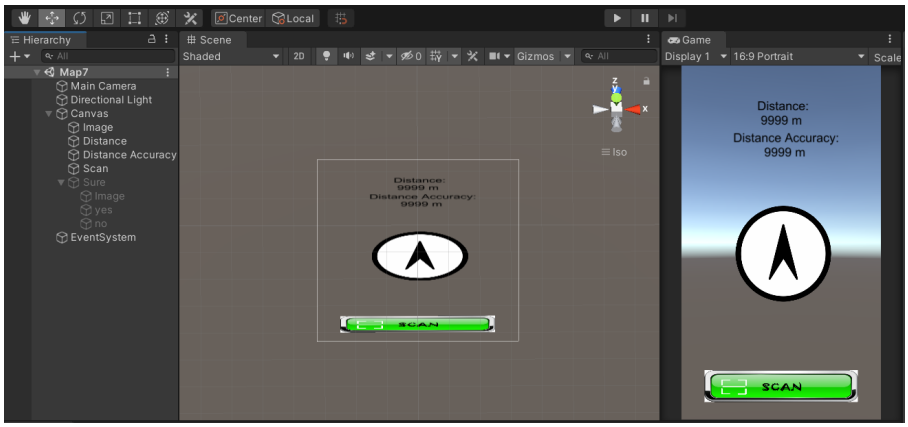


Figura 3.28: la scena della localizzazione del marker in Unity

[1]Il metodo Update viene usato in Unity per creare quello che viene definito game loop: una ripetizione ciclica di controlli e di operazioni, che avviene una volta per frame e che determina cosa avviene in scena.[16]

3.6 Tesoro

(a) *Forziere chiuso che gira* (b) *Forziere aperto con il tesoro*

Figura 3.29: Scena del tesoro

Appena arrivati in questa nuova scena troviamo il forziere ancora chiuso che gira, Figura 3.29(a). Il giocatore deve aprire il forziere e questo può farlo cliccando sul catenaccio. Una volta aperto mi ritrovo in figura 3.29(b), oltre ai fuochi d'artificio è presente il suono di una tromba e la scritta cambia in 'Victory'. Sono presenti , infine due pulsanti, replay che mediante il solito comando per cambiare scena,come in figura 3.2, mi riporta nella schermata iniziale, e il tasto exit che mi fa uscire dal gioco tramite il comando seguente:

```
public void OnExit()
{
    Application.Quit();
}
```

Figura 3.30: Comando per chiudere un applicazione

Infine è presente un ultima funzione del gioco, se il marker non viene più inquadrato e poi inquadrato una seconda volta, il tesoro scompare e così anche i fuochi d'artificio, vedi figura 3.31.

Figura 3.31: Scena Treasure quando inquadro la seconda volta il marker

Vediamo ora come è stata creata questa scena in Unity. Innanzitutto ho dovuto importare e scaricare dall'assetstore di Unity sia il forziere con il tesoro [17], sia i fuochi d'artificio [18].

Ecco come si presenta in Unity :

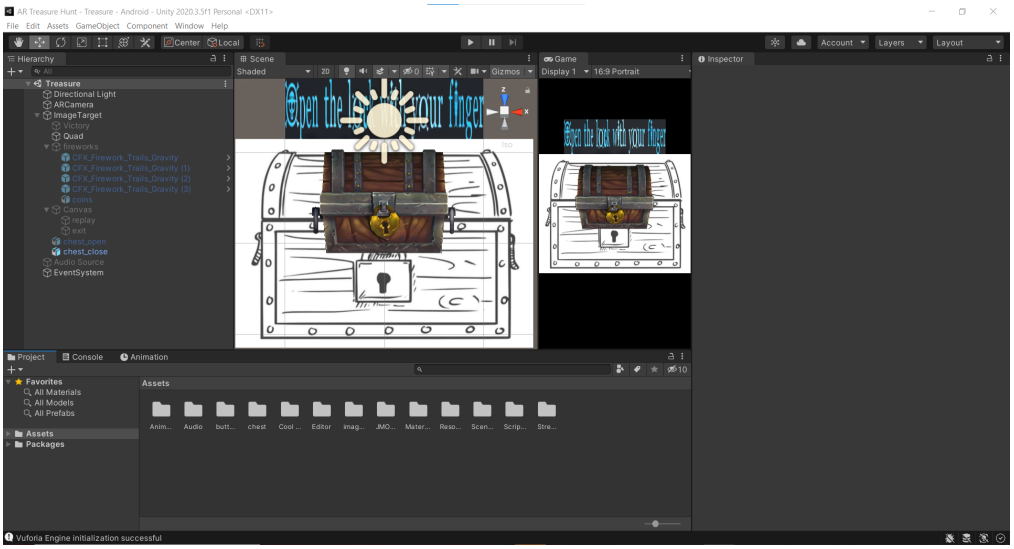


Figura 3.32: Scena Treasure in Unity

Quando viene importato il forziere dall' assetstore di Unity, all'interno troviamo due forzieri 3D, chest_close e chest_open, quindi uno aperto e l'altro chiuso. Nella figura 3.32, si vede solo quello chiuso ma all'interno della scena come si puo vedere a sinistra sono presenti entrambi solo che quello aperto è stato disattivato. Al chest_close è stato dato un Box Collider[2] è stato rimpiciolito per avere all'incirca le dimensioni del catenaccio ed è stato posizionato davanti a esso. Una volta fatto ciò attraverso script posso definire qual'è l'evento scatenante e cosa deve succedere. In questo caso l'evento scatenante è il touch sul catenaccio, e questo viene gestito dallo script 'OpenChest',vedi figura 3.33

[2]In Unity, il Collider è quello che si occupa di gestire le collisioni fra oggetti. Esistono diversi tipi di Collider in base alla forma: BoxCollider, SphereCollider, CapsuleCollider e MeshCollider. Quando IsTrigger è abilitato è utilizzato per attivare eventi.

```
void OnMouseDown()
    {
        close.SetActive(false);
        open.SetActive(true);
        fireworks.SetActive(true);
        text.SetActive(false);
        victory.SetActive(true);
        aSource.SetActive(true);
        button.SetActive(true);
    }
```

Figura 3.33: Script 'OpenChest'

OnMouseDown viene chiamato quando l'utente ha premuto il pulsante del mouse mentre si trova sul Collider, nel caso di dispositivi mobili equivale al touch sul Collider, però non supporta il multitouch, quindi non va bene quando devo cliccare in più punti contemporaneamente sul mio dispositivo mobile. Ora vediamo cosa accade quando tocco il catenaccio del forziere. Riferendoci alla figura 3.33, possiamo notare che la variabile GameObject close [3] viene messa su 'false' cosi come anche text, essa è la scritta che invita il giocatore ad aprire il lucchetto con il proprio dito. Invece quelle che vengono attivate sono open, che è la figura del mio forziere aperto senza tesoro, fireworks, che come indica il nome contiene i fuochi d'artificio ma ho inserito anche le monete d'oro e questo sarà chiarito a breve, victory, che è l'omonima scritta, aSource, è l'AudioSource in cui è presente il suono della tromba ed infine, button, in cui sono presenti i pulsanti di 'replay' ed 'exit'. La scelta di inserire coins all'interno di fireworks è stata fatta solo per comodità dato che compaiono insieme e allo stesso modo scompaiono insieme. Per farli scomparire, cosi come anche il suono della tromba, una volta che il marker non è più inquadrato viene utilizzato lo script 'Coins', anche se riguarda anche i fireworks. Tale script ha le funzionalità simili a quelle di 'Default Trackable Event Handler' di Vuforia, solo che quando non avviene piu il tracciamento del marker disattiva il GameObject fireworks ed AudioSource.

[3] tutte le variabili sono state dichiarate public GameObject

3.7 Director

Il Director è colui che ha il compito di organizzare la caccia al tesoro, quindi di posizionare i marker ed inserire le coordinate GPS nel caso della versione 'Outdoor', nel gioco è stata assegnata una sezione che gli è utile a fare ciò. Per ricevere istruzioni per prima cosa nella schermata iniziale, vedi figura 3.3. Una volta cliccata apparirà una schermata, vedi figura 3.34, in cui può scegliere quale caccia al tesoro organizzare, 'At home' o 'Outdoor'.

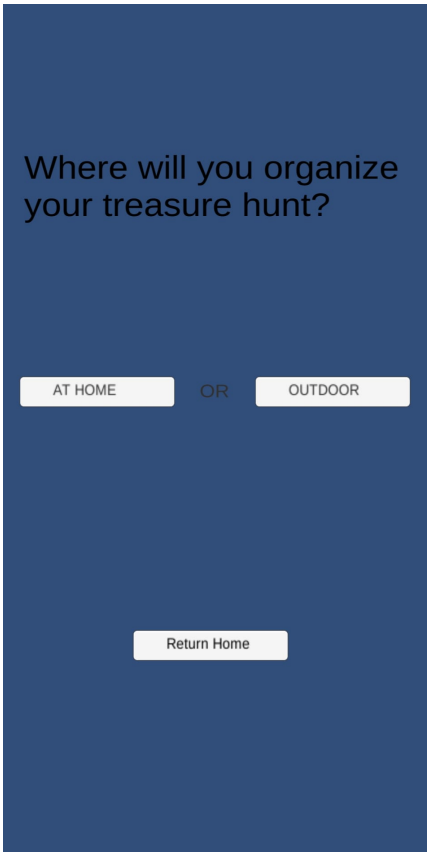


Figura 3.34: Scelta delle istruzioni in base alla versione desiderata

Qui è impostato lo sfondo della scena di unity blu, ed aggiunto dei 'button' uno mi porta alle istruzioni per la versione 'At Home' l'altro per la versione 'Outdoor'.

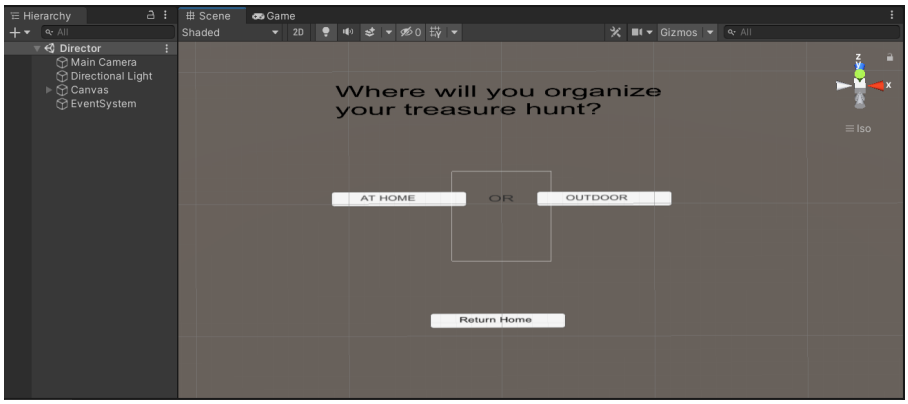


Figura 3.35: Visto in Unity

3.7.1 At home

Se la scelta ricade su 'at home' viene aperto Gitlab[19], attraverso script come possiamo vedere nella figura seguente.

```
public void OpenUrlOnBrowser()
{
    Application.OpenURL(@link);
}
```

Figura 3.36: comando per aprire una pagina web con il click di un pulsante

Il sito web che si desidera aprire può essere inserito nell'inspector del pulsante, all'interno dello script avendo dichiarato una variabile public string.

Figura 3.37: inspector del pulsante 'At Home'

In GitLab abbiamo sia i marker per la versione At home che i marker per la versione 'Outdoor', in piu ci sono le istruzione per il 'Director' , sempre per entrambi le versioni, e il tutto può essere scaricato.

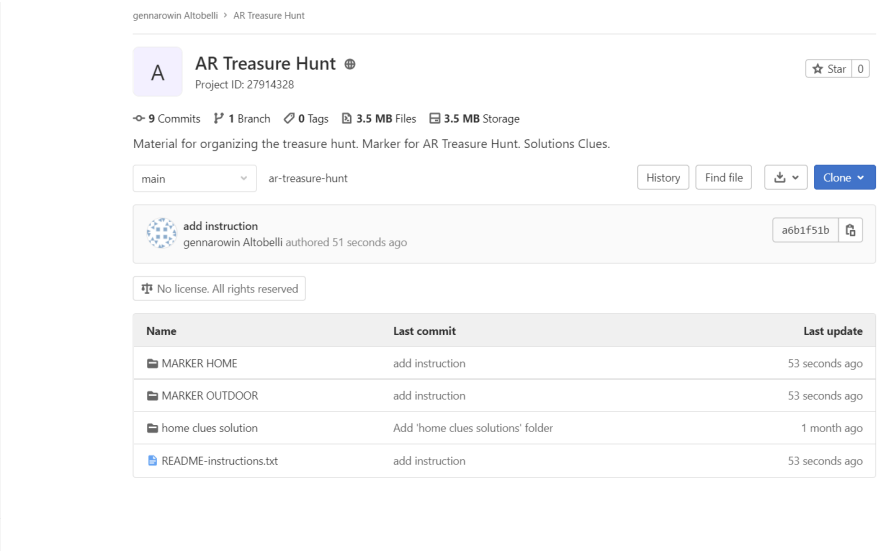


Figura 3.38: Istruzioni e marker in GitLab

3.7.2 Outdoor

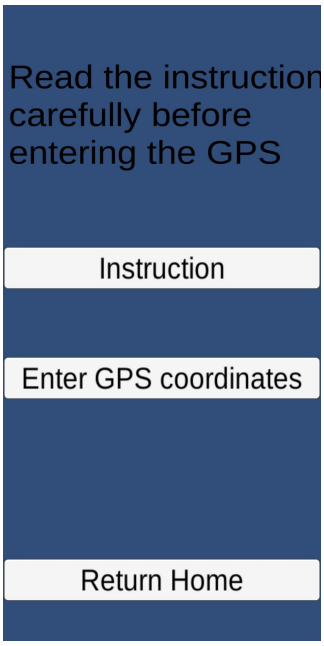


Figura 3.39: scelta per la versione outdoor

In questa scena troviamo due pulsanti uno che mi apre lo stesso GitLab precedente e quindi sono consultabili istruzioni e marker, l'altro mi porta in una nuova scena, in cui è possibile inserire le coordinate GPS dei vari marker. Un terzo pulsante mi porta nella schermata iniziale.
In Unity:

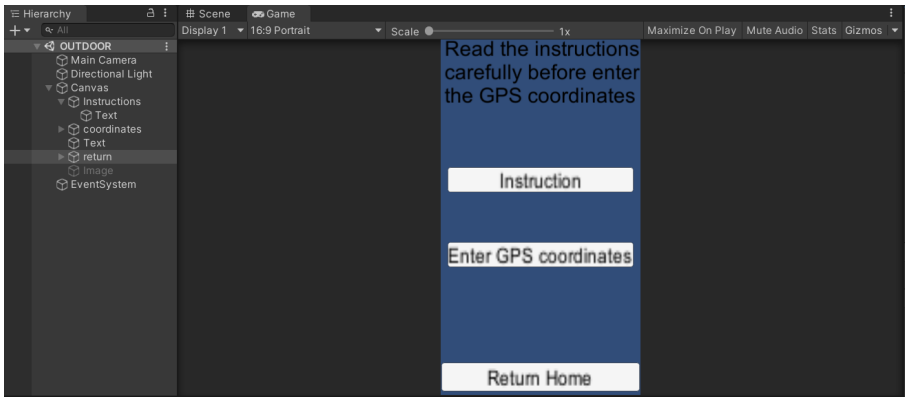


Figura 3.40: scelta per la versione outdoor

Possiamo notare che c'è un ulteriore GameObject, image, questa come detto in precedenza è l'immagine di caricamento, loading.

3.7.3 Inserimento coordinate

Figura 3.41: Inserimento coordinate GPS dei marker

Qui il Director può inserire latitudine e longitudine di ogni marker. In alto a sinistra viene aiutato segnalando qual'è la sua attuale posizione. Quindi, quando va a posizionare il marker, ha le sue coordinate GPS e può inserirle. Il pulsante 'return' mi riporta nella scena iniziale.

In Unity, invece, la scena si mostra cosi:

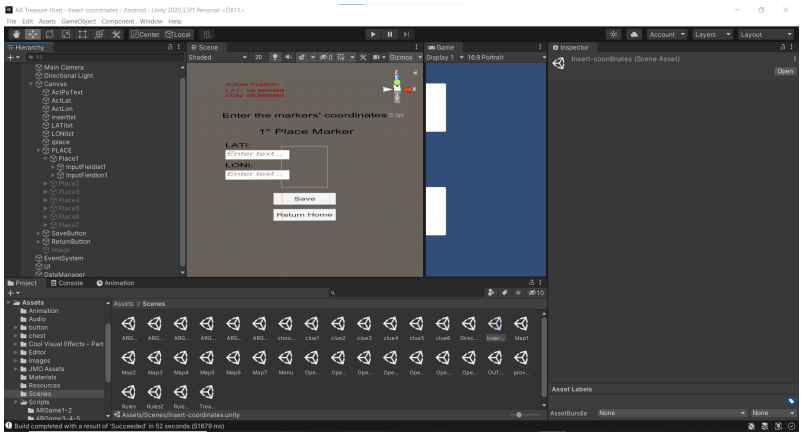


Figura 3.42: Inserimento coordinate GPS dei marker visto in Unity

Per le coordinate è stato utilizzato sempre lo script GPS. Per i campi in cui scrivere la latitudine o la longitudine è stato utilizzato il GameObject InputField, presente nelle UI. Per ogni inserimento, quindi ogni volta che clicco il pulsante Save, cambio anche gli InputField, disattivando i precendenti ed attivando i successivi, in modo da inserire tutte le coordinate, sette in tutto, questo viene gestito dallo script UI. I due InputField, uno per la latitudine e uno per la longitudine, sono inseriti all'interno di un GameObject Empty, chiamato Place.
Quindi disattivo ed attivo questi Gameobject durante il salvataggio. Per il salvataggio ho usato degli script presi da un tutorial su youtube [20]. Questi mi permettono di salvare i contenuti in un file di testo che ho chiamato "Savefiles".

3.8 Flessibilità

Nella versione 'Outdoor' , il 'Director' sceglie lui le coordinate GPS e può cambiarle di continuo, quindi ogni volta può creare una caccia al tesoro diversa, ma inserendo sempre sette coordinate. Invece nella versione 'At home' gli indizi sono fissi e non è possibile nemmeno cambiare l'ordine che rimane quello prestabilito in fase di progettazione. Per questo mentre la prima può essere giocata ogni volta la seconda solo una volta.

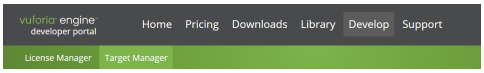
Capitolo 4

Modifiche al progetto

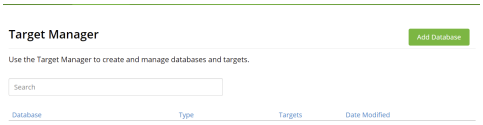
Se si intende apportare delle modifiche al progetto, bisogna prima scaricare il progetto dal mio Gitlab [19], successivamente aprire Unity andare su **File-> Open Project** e selezionare la cartella precedentemente scaricata. Una volta finite le modifiche si passa a fare il build e si genera il nuovo APK.

4.1 Cambio Marker Vuforia

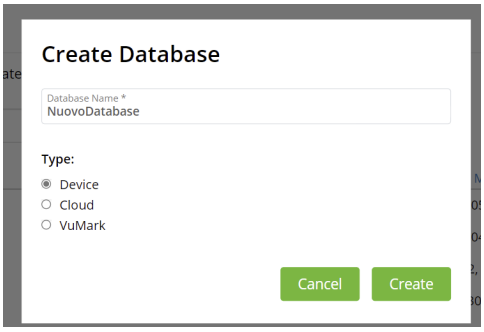
Per poterlo fare bisogna essere iscritti a Vuforia, che è gratuita se non si intende mettere l'App negli store. Una volta loggato cliccare su Develop in alto, successivamente Target Manager->Add Database,vedi figura 4.-2(a) e (b), dargli un nome e cliccare create, vedi figura 4.-2(c). Cliccare sul database da voi appena creato, vedi figura 4.-2(d), una volta aperta la nuova pagina aggiungere i target, vedi figura 4.-2(e) e (f), cercando di metterli con un rating di almeno tre stelle per un tracciamento migliore. Una volta inseriti tutti cliccare su Download Database(All), scegliendo come development platform Unity Editor, vedi figura 4.-2(g). Una volta scaricato il database, basta cliccare e in Unity vi verrà chiesto di importarlo. Per cambiare i Targert delle varie scene basta andare su 'ImageTarget' della scena in cui si vuole cambiare il target, e nell'Inspector scegliere come database quello da voi appena creato, e come immagine il target che desiderate per quella particolare scena, vedi figura 4.-2(h).



(a)

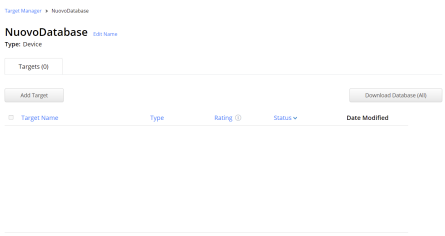


(b)

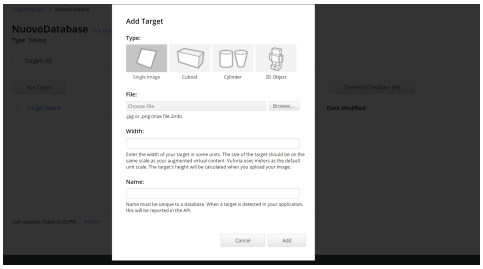


(c)

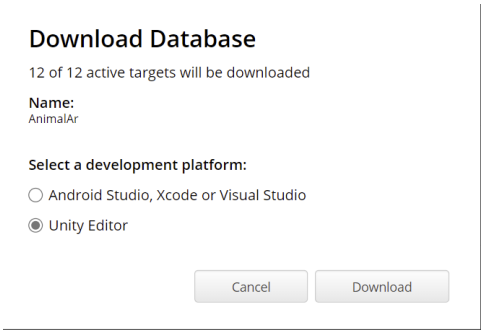
(d)



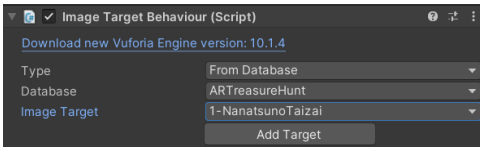
(e)



(f)



(g)



(h)

Figura 4.-2: Cambio Targets

4.2 Cambio indizi

Per cambiare indizi, invece, basta creare un immagine contenente l'indizio nuovo, trascinare queste immagini all'interno di Unity ad esempio nella cartella 'images'. Una volta fatto ciò basta andare nelle scene contenenti gli indizi e cambiare immagine al Quad o trascinando l'immagine su di esso o nel suo Inspector, all'interno di Mesh Renderer, cambiare 'Element 0' scegliendo l'immagine con l'indizio desiderato.

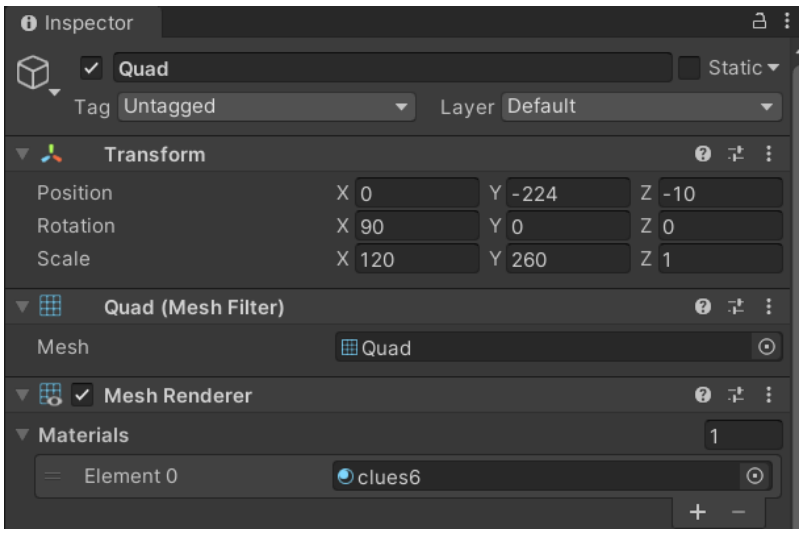


Figura 4.-1: Cambio immagine nel Quad

4.3 Cambio numero indizi

Nel gioco, il numero degli indizi è sempre sei, se si desidera aumentarli o diminuirli bisogna, nel primo caso ricreare nuove scene e modificare gli script, nel secondo caso solo script.

4.3.1 Diminuire il numero di indizi

In questo caso' bisogna modifcare lo script 'yes.cs','button33.cs','button4.cs'. In particolar modo, bisogna modificare l'if per farlo andare nella scena che si desidera, eliminando quelle superflue.

```
public void OnContinue2()
{
    activeAudio.reload = false;
    if(SceneManager.GetActiveScene().name=="clue1")
    SceneManager.LoadScene("ARGame2");
    else if(SceneManager.GetActiveScene().name == "clue2")
     SceneManager.LoadScene("ARGame3");
    else if (SceneManager.GetActiveScene().name == "clue3")
        SceneManager.LoadScene("ARGame4");
    else if (SceneManager.GetActiveScene().name == "clue4")
        SceneManager.LoadScene("ARGame5");
    else if (SceneManager.GetActiveScene().name == "clue5")
        SceneManager.LoadScene("Rules2");
    else if (SceneManager.GetActiveScene().name == "Rules2")
        SceneManager.LoadScene("ARGame6");
    else if (SceneManager.GetActiveScene().name == "ARGame6")
        SceneManager.LoadScene("clue6");
    else if (SceneManager.GetActiveScene().name == "clue6")
        SceneManager.LoadScene("Treasure");
}
```

(a)

```
if (SceneManager.GetActiveScene().name == "ARGame1")

    SceneManager.LoadScene("clue1");
else
    SceneManager.LoadScene("clue2");
```

(b)

```
if (SceneManager.GetActiveScene().name == "ARGame3")

    SceneManager.LoadScene("clue3");

else if(SceneManager.GetActiveScene().name == "ARGame4")
    SceneManager.LoadScene("clue4");
else if (SceneManager.GetActiveScene().name == "ARGame5")
    SceneManager.LoadScene("clue5");
```

(c)

Figura 4.-1: Cambiare l'if per modificare il numero degli indizi

Infine prima di fare il build, deselezionare le scene, che si vogliono eliminare, in 'Scenes in Build' che è all'interno di 'Build Setting'.

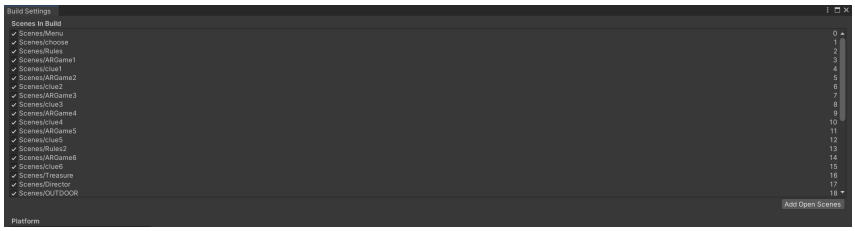


Figura 4.0: Deselezionare la spunta prima di fare il build

4.3.2 Aumentare il numero di indizi

In questo caso devo creare nuove scene clues e altre scene ARGame. Per farlo è possibile duplicare una scena, quindi apro prima la scena che voglio duplicare, poi clicco su edit nella barra menu in alto ed infine duplicate, senza dimeticare poi di dare un nuovo nome. Una volta create nuove scene devo modificare gli script precedenti aggiungendo nuovi else if. Infine bisogna aggiungere le scene

create all'interno di 'Build Setting', o trascinandole o tenerle aperte e cliccare su 'Add Open Scenes'.

4.4 Cambio sequenza AR Game

Se si desidera avere una sequenza di pulsanti diversa da quella stabilita, allora bisogna assegnare ai diversi pulsanti script button[1] diversi. Questo vale solo per quelli in cui devo indovinare la sequenza ed ho i pulsanti fissi,invece nell'ultimo se voglio cambiare il pulsante devo assegnare lo script yes.cs al pulsante da indovinare, basta trascinarlo nell'inspector del pulsante, poi trascirare lo script anche in Onclick, che si trova sempre nell'inspector del pulsante cliccare su No Function, scegliere yes e poi Finalclick1().

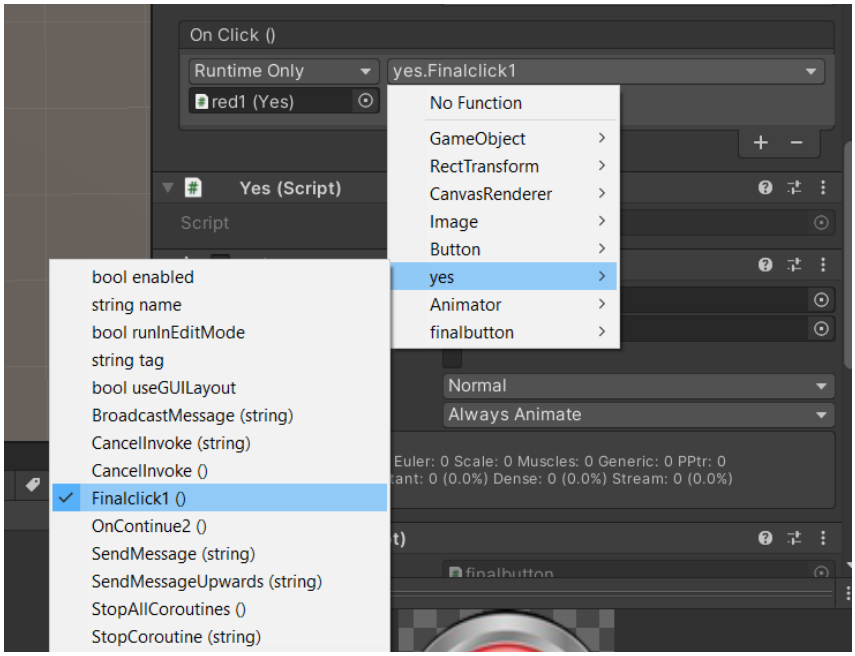


Figura 4.1: Assegnazione script al click del button

[1]Intendo i vari script button13.cs, button23.cs, button33.cs, button1.cs, ecc...

Conclusione

Tramite Unity e Vuforia si è riusciti a realizzare una caccia al tesoro in realtà aumentata per Android. Dato che non avevo mai utilizzato Unity, non è stato semplice ed immediato riuscire ad utilizzarlo, ma grazie ai tutorial, alcuni anche in italiano, su YouTube ma sopratutto grazie alla vasta community di Unity, sono riuscito a capire e risolvere molti problemi, dato che molti altri hanno avuto le mie stesse difficoltà e le risposte a queste erano molteplici, confrontanto le varie soluzioni sono riuscito a risolvere man mano ogni problema e completare l'applicazione.

Bibliografia e Sitografia

[1] URL: https://it.wikipedia.org/wiki/RealtÃă_aumentata.

[2] URL: https://creatoridifuturo.it/comunicazione/comunicazione-non-convenzionale/realta-aumentata-quello-che-ogni-marketer-dovrebbe-sapere/.

[3] URL: https://automazione-plus.it/la-realta-aumentata-di-microsoft-per-la-manutenzione-degli-ascensori-thyssenkrupp_86309/.

[4] URL: https://it.wikipedia.org/wiki/RealtÃă_virtuale.

[5] Marta Matamala-Gomez, Roberto De Icco e Giorgio Sandrini. «Telemedicina e realtà virtuale ai tempi della pandemia da Covid-19». In: *Confinia Cephalal Neurol* 30 (2020), pp. 79–83.

[6] URL: http://www.digitalmosaik.com/blog/differenza-ar-vr.

[7] Amir Baldissera. *Realtà virtuale e realtà aumentata per il business: Applicazioni pratiche con la Mixed Reality*. HOEPLI EDITORE, 2020.

[8] URL: https://www.pikkart.com/servizi/notizie/notizie_fase02.aspx?ID=1797.

[9] URL: https://library.vuforia.com/getting-started/overview.html.

[10] URL: https://it.wikipedia.org/wiki/Motore_grafico.

[11] URL: https://it.wikipedia.org/wiki/Motore_di_rendering.

[12] URL: https://docs.unity3d.com/Manual/CreatingScenes.html.

[13] URL: https://developer.vuforia.com/downloads/sdk.

[14] URL: https://git-scm.com/downloads.

[15] URL: https://about.gitlab.com/.

[16] URL: https://www.html.it/pag/45529/monobehaviour-gli-eventi-di-unity/.

[17] URL: https://assetstore.unity.com/packages/3d/props/interior/treasure-set-free-chest-72345.

[18] URL: https://assetstore.unity.com/packages/vfx/particles/cartoon-fx-free-109565.

[19] URL: https://gitlab.com/Icegennaro7/ar-treasure-hunt.

[20] URL: https://www.youtube.com/watch?v=ii31ObaAaJo.