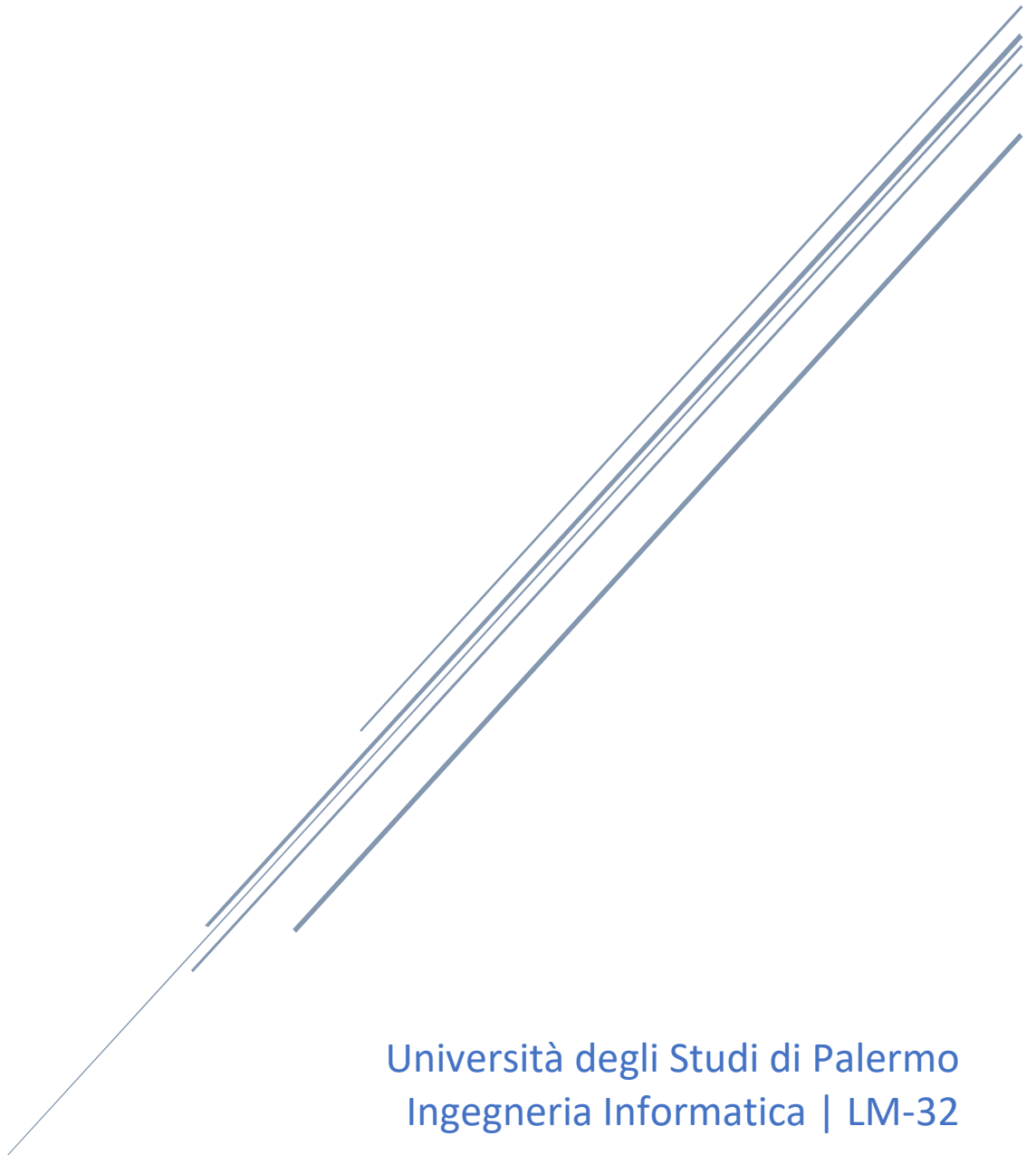


RESCUE BOTS

Progetto di Robotica ed Intelligenza Artificiale II



Università degli Studi di Palermo
Ingegneria Informatica | LM-32

*Ignazio Daniele Di Blasi
Domenico Giosuè
Chiara Lupo*

Sommario

INTRODUZIONE.....	1
DESCRIZIONE DEL PROGETTO	1
APPLICAZIONE REALE	1
AMBIENTE DI SVILUPPO.....	2
TOOLS ESTERNI	3
MODELLAZIONE DEL PROBLEMA	3
MODELLAZIONE DEL TERRENO	3
STRUTTURA DEL PROGETTO	4
GERARCHIA DELLE COMPONENTI	4
<i>Assets</i>	4
COMPONENTE XR	5
<i>Complete XR Origin Set Up</i>	5
COMPONENTI FISICHE	6
<i>Terrain</i>	6
<i>Tablet</i>	6
<i>Mine</i>	7
<i>Landmark</i>	7
COMPONENTI LOGICHE	8
ALTRE COMPONENTI.....	10
SCENE	11
<i>Start Scene</i>	11
<i>Primary Scene</i>	11
AUDIO	12
HARDWARE	13
MASTER ROBOT	13
<i>FSM</i>	13
<i>Ricerca</i>	15
SEEKER ROBOT	15
<i>Movimento</i>	15
<i>Comunicazione</i>	15
PROTOCOLLO DI COMUNICAZIONE	16
EVENTMANAGER	16
SICUREZZA	17
MESSAGGI	17
<i>EXCH_SP</i>	17
<i>EXCH_MS</i>	18
<i>AUTH_SP</i>	18
<i>AUTH_MS</i>	19
<i>MINE_SP</i>	19
SUITE CRITTOGRAFICA	20
<i>Algoritmo di Cifratura</i>	20
<i>Funzione Hash</i>	20
<i>Certificati</i>	20
ALGORITMI	21
RICERCA A*	21

<i>Utilità</i>	22
<i>Implementazione</i>	23
FILTRO DI KALMAN	23
<i>Utilità</i>	24
<i>Implementazione</i>	24
GUIDA AL DEPLOY	25
TEST	30
ATTREZZATURA	30
CONCLUSIONI	31

Introduzione

Descrizione del progetto

Ambiente immersivo, in Realtà Virtuale, che simula uno scenario di guerra in cui un operatore ha lo scopo, dato un punto **A** di partenza, di raggiungere un punto **B** obiettivo per l'evacuazione.

Lo scenario risulta interamente percorribile; tuttavia, questo presenta diverse mine antiuomo, generate in posizioni casuali a ogni avvio, che potrebbero ferire o uccidere l'operatore.

L'operatore, nella ricerca di un percorso sicuro, sarà assistito da una serie di Robot così categorizzati:

- Seeker/Spider Robots:

Robot cercatori che, dividendosi in punti differenti della mappa, esaminano il terreno e individuano la presenza di mine antiuomo mediante un sensore di rilevamento di metalli (Metal Detector Sensor).

Man mano che i Seeker Robots scansioneranno il terreno, questi manderanno delle informazioni sulle mine individuate al Master Robot.

- Master Robot:

Robot principale che, date le informazioni ricevute dai Seeker Robots, si occupa di generare il percorso migliore tenendo conto della distanza tra il punto **A** e il punto **B** e della presenza di mine, cercando di salvaguardare la vita dell'operatore.

Una volta che il Master Robot avrà selezionato il percorso migliore, questo comincerà a muoversi dando all'operatore la possibilità di seguirlo verso il punto di evacuazione.

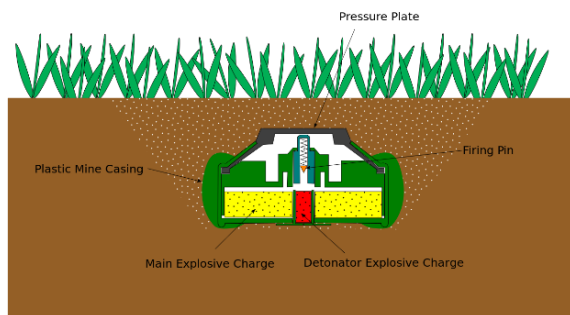
L'operatore, inoltre, potrà utilizzare un Tablet per consultare una mappa, aggiornata in tempo reale, che mostra la posizione del Master Robot, dei Seeker Robot e delle Mine rilevate.

Applicazione Reale

Questo progetto nasce con l'obiettivo di salvaguardare vite umane.

In luoghi dove la guerra purtroppo non è una realtà lontana, la realizzazione di robot in grado di perlustrare l'ambiente al fine di individuare mine antiuomo, progettate con lo scopo di ferire o uccidere persone, risulta di vitale importanza. Ottenute le informazioni riguardo le posizioni delle mine rilevate, il Master Robot calcolerà un percorso relativamente sicuro, provvedendo a guidare l'umano durante il tragitto dal punto di partenza al punto di arrivo e, al tempo stesso, perlustrando la zona da attraversare con l'ausilio di un sensore di rilevamento di metalli.

Anti-Personnel Mine (Blast Type) Components



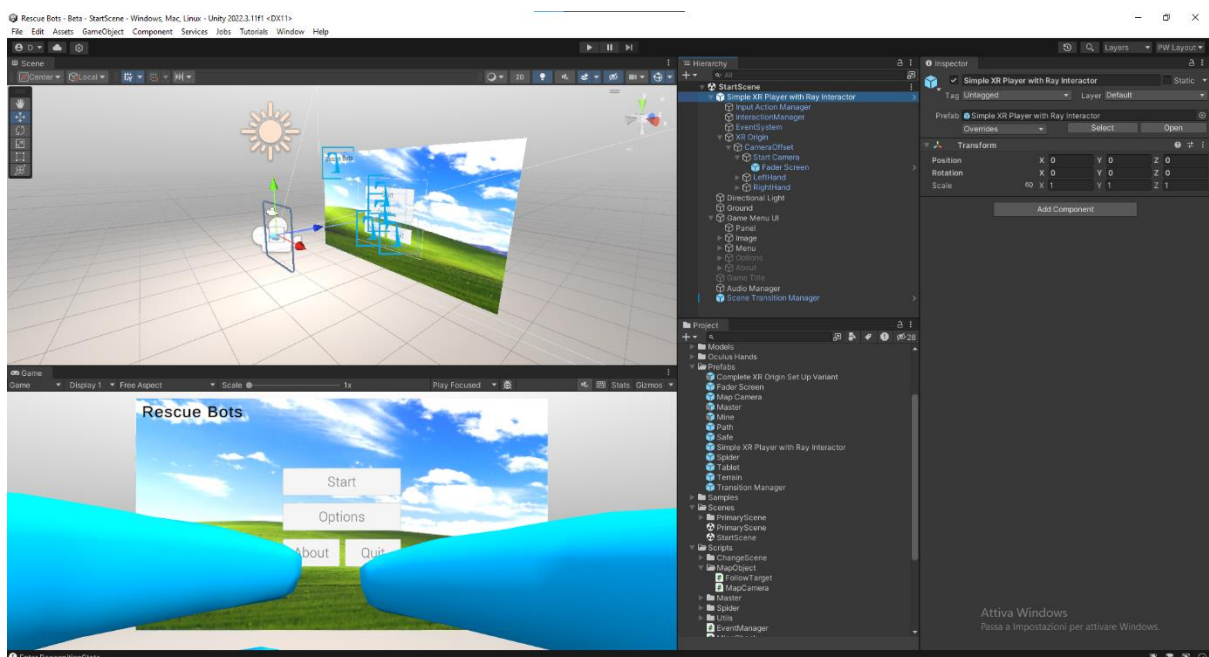
Le mine antiuomo (PMN-1) vengono attivate da una pressione esercitata da una massa di almeno 5,8kg. Per tale ragione, è preferibile utilizzare dei Seeker Robot, aventi massa nettamente inferiore a quella necessaria per avviare l'ordigno, per individuare le posizioni in cui sono presenti le mine.

Gli Spider Robot mediante un protocollo sicuro mandano le informazioni al Master riguardo le mine che sono state individuate durante la perlustrazione dell'ambiente. L'operatore potrà visualizzare tali informazioni su un tablet messo a sua disposizione durante la fuga dalla zona minata.

Ambiente di Sviluppo

L'intero progetto ha visto il suo sviluppo all'interno del Game Engine **"Unity"** mediante il quale è stato possibile costruire l'intero ambiente di simulazione, le relazioni tra i GameObject, le meccaniche di gioco XR e, mediante una serie di Script in C#, l'intera logica dell'applicazione.

Il Game Engine in questione, tra le altre cose, offre una facile gestione delle associazioni tra Scripts e GameObject, consentendo uno sviluppo rapido ma al contempo stabile e preciso.



Tools Esterni

Oltre all'utilizzo di Unity, il progetto ha richiesto l'utilizzo di una serie di Software e Tools esterni per ciò che riguarda la produttività e la creazione (o modifica) degli Assets necessari per la realizzazione dell'intero ambiente di simulazione.

In particolare, il supporto alla realizzazione del progetto è stato fornito da:

- Visual Studio / Visual Studio Code
 - IDE utilizzati per lo sviluppo del codice sorgente in linguaggio C#, sono dotati di funzionalità come il debugging integrato e il controllo di versione.
- Blender
 - Software per la realizzazione di modelli 3D utilizzato per ottenere e modificare alcuni modelli utilizzati nel progetto.

Modellazione del Problema

L'idea progettuale nasce dalla volontà di realizzare un ambiente di simulazione che sia in grado di riprodurre uno scenario di guerra in cui un operatore, situato all'interno dello scenario stesso, si ritrovi ad avere la necessità di attraversare un campo minato per fuggire tempestivamente dal territorio.

L'ambiente è stato realizzato considerando che l'operatore umano possa percorrerlo e viverlo interamente; tuttavia, la percezione dell'ambiente da parte dei Robot è diversa rispetto a quella che ha l'operatore umano.

Per questo motivo, è stato necessario modellare il problema sotto un altro aspetto, consentendo ai Robot di discretizzare lo spazio attorno a essi, per permettergli di elaborare i percorsi da intraprendere.

Modellazione del Terreno

Il terreno è rappresentato come una matrice di dimensione 50 x 50, suddividendo così l'area in un totale di 2500 celle.

Le bombe sono disposte casualmente all'interno delle celle della matrice. Viene seguita una regola specifica: in media è presente 1 mina ogni 25 celle.

I punti di partenza e arrivo vengono scelti casualmente ai lati opposti della matrice. Intorno a questi punti, è presente una zona sicura di 3 x 5 caselle, che assicura la partenza e l'arrivo in un contesto protetto.

La casualità del punto di partenza e del punto di arrivo, insieme alla generazione casuale delle mine, fornisce una varietà alla simulazione nella ricerca del percorso sicuro.

Struttura del Progetto

Il Progetto comprende molteplici componenti che ne compongono la struttura, fornendo tutte le risorse, sia fisiche sia logiche, necessarie alla corretta esecuzione dell'ambiente di simulazione.

Gerarchia delle Componenti

Le componenti incluse all'interno della directory del progetto seguono un ordine ben preciso e vanno a formare una gerarchia tale per cui le componenti che hanno stessa natura, o che sono in qualche modo vincolate da uno stesso utilizzo, si trovano raccolte in sottodirectory specifiche.

Le due cartelle più importanti, direttamente connesse alla Root Directory del progetto, sono:

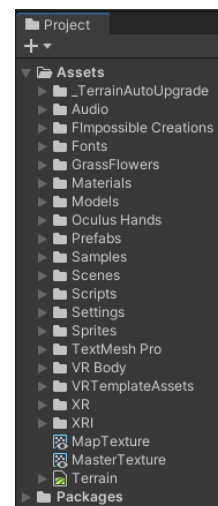
- **Assets**
 - Directory che comprende tutti gli elementi inclusi nel progetto e che vengono in qualche modo utilizzati e caricati durante la normale esecuzione di questo
- **Packages**
 - Comprende tutte le cartelle dove sono contenuti i modelli generali degli assets successivamente inclusi nel progetto

Assets

L'interesse principale riguardo la gerarchia delle componenti è rivolto verso la sottodirectory "Assets"; questa, come già anticipato, comprende tutte quelle componenti utili al corretto funzionamento del prodotto finale.

Guardando l'immagine a fianco, che mostra tutte le sottodirectory all'interno della directory "Assets", è possibile distinguere alcune cartelle fondamentali e tipicamente integrate in ogni progetto Unity:

- **Materials**
 - Comprende gli oggetti di tipo "Material" utilizzati per fornire un aspetto grafico agli oggetti in scena
- **Prefabs**
 - Comprende tutti i GameObject, con le opportune configurazioni logiche e grafiche, che dovranno prima o poi essere istanziati all'interno della scena
- **Scenes**
 - Comprende le diverse Scene del progetto, ognuna delle quali comprenderà i propri GameObject e la propria logica
- **Scripts**
 - Comprende tutti gli script per l'implementazione delle funzionalità logiche



Componente XR

Una delle componenti principali di questo progetto è la componente XR, la quale fornisce e gestisce il sistema di interazione in ambito VR/AR.

Nel caso specifico del progetto, la componente XR fornirà tutte le impostazioni necessarie alla gestione degli input XR comprese la logica relativa alle interazioni con le componenti di gioco e la logica relativa al movimento.

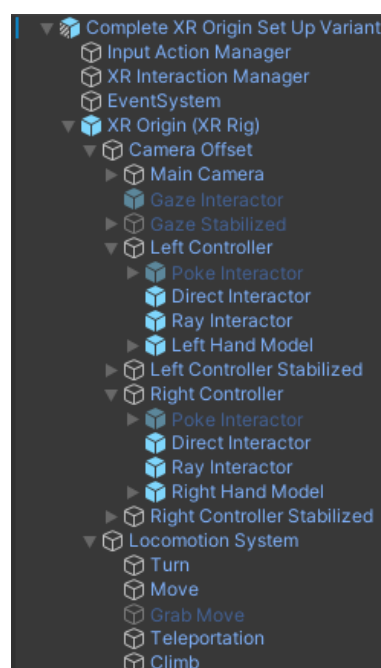
L'oggetto di gioco che implementa le suddette impostazioni è denominato come "Complete XR Origin Set Up Variant" e non è altro che una versione modificata del Set Up di default fornito da Unity.

Complete XR Origin Set Up

Il GameObject in questione è da considerarsi associato al giocatore vero e proprio; di fatto, questo comprende una serie di GameObject figli associati alle differenti funzionalità del giocatore e del suo sistema di interazione con l'ambiente circostante.

Tra i suddetti GameObject è bene dettagliare:

- XR Origin
 - Main Camera
 - Videocamera principale associata al POV del giocatore
 - Left/Right Controller
 - Comprendono, oltre che al modello 3D associati ai dispositivi di input del giocatore, tutti i possibili metodi di interazione con gli oggetti presenti nell'ambiente circostante
 - Locomotion System
 - Contiene dei GameObject a cui sono associate delle componenti logiche specifiche per i diversi sistemi di movimento possibili



Componenti Fisiche

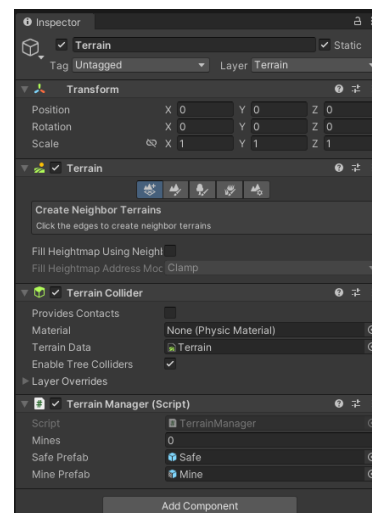
Un'altra categoria di componenti importante è relativa a tutte le componenti fisiche del progetto, queste intese come i GameObject graficamente renderizzati dal Game Engine.

Molte di queste componenti, come anticipato, sono memorizzate nel percorso *'Project/Assets/Prefabs'* e verranno istanziate nella scena al momento opportuno.

Terrain

Una delle principali componenti fisiche del progetto rappresenta il terreno ed è da considerarsi come il piano sul quale tutti i GameObject possono muoversi. Questa componente, in realtà, non è un vero e proprio piano; di fatto, questa integra una serie di dati relativi al terreno come, per esempio, l'altezza di questo in ogni punto.

Nel progetto, il GameObject "Terrain" è dotato di una componente logica specifica che consente di generare casualmente il punto **A** di partenza e il punto **B** di arrivo e, analogamente, di generare casualmente le posizioni in cui verranno generate le mine antiuomo.



Tablet

Un'altra componente fisica, realizzata appositamente per essere integrata in questo progetto, consiste in un Tablet che consente al giocatore, mediante un menù semplice e intuitivo, di:

- Visualizzare una Mappa della zona
- Visualizzare il POV della videocamera del Master
- Terminare la sessione di simulazione

Al Tablet è associato lo script "XR Grab Interactable" che consente al giocatore di interagire fisicamente con l'oggetto, utilizzare la funzione di "Grab" per prendere in mano l'oggetto e utilizzare la funzione di "Poke" per selezionare le opzioni a schermo.



Mine

La presenza delle Mine Antiuomo PMN-1 è essenziale ai fini della simulazione; dunque, è opportuno citare la presenza di tale componente, definendone la struttura e l'implementazione.

Le diverse Mine Antiuomo vengono generate, a ogni avvio, in posizioni casuali del terreno, utilizzando un valore costante per determinare la loro posizione verticale; di fatto, queste sono sempre posizionate sotto il terreno, in modo tale da essere poco visibili all'occhio umano.



Le Mine possono entrare in contatto con un Collider Component, denominato "MineCollider", avviando una detonazione che porta al fallimento della simulazione.

Landmark

L'utilizzo dei Landmark, come verrà illustrato più avanti, è di fondamentale importanza per l'implementazione del Filtro di Kalman.

Nell'ambito del progetto, i Landmark sono stati rappresentati da quattro cartelli segnaletici, posti su dei pali ai quattro angoli del terreno minato, che forniscono un'indicazione di pericolo, proprio dovuta alla presenza delle mine in tutto il territorio.



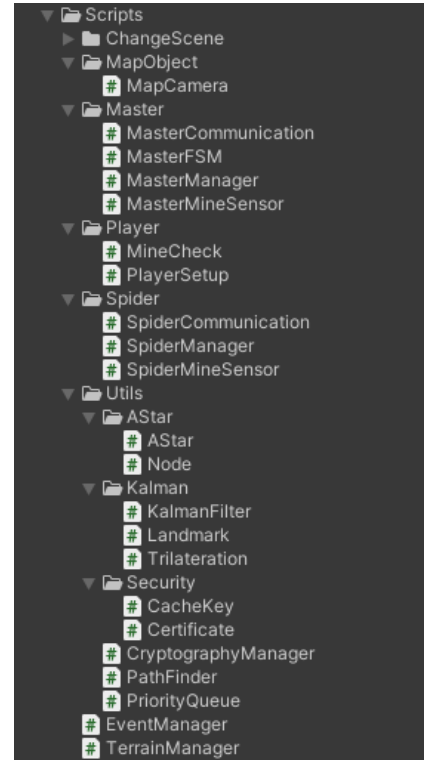
Componenti Logiche

Le componenti logiche del progetto sono rappresentate da tutti gli script, in linguaggio C#, che determinano le funzionalità necessarie per la corretta esecuzione dell'ambiente di simulazione.

Tutte le componenti realizzate ad-hoc per il progetto sono memorizzate all'interno del percorso

'Project/Assets/Scripts' e prevedono una categorizzazione basata sulla parte del progetto su cui queste influiscono. In particolare, la gerarchia delle componenti logiche può essere descritta come segue:

- ChangeScene
 - Directory contenente una serie di script necessari al funzionamento della Scena di gioco contenente il menù iniziale
- MapObject
 - MapCamera
 - Script necessario al funzionamento del GameObject "Camera" che si occupa di fornire una vista dall'alto dei soli oggetti appartenenti ai Layer "Terrain" e "MapObject"
- Master
 - MasterCommunication
 - Script che si occupa di gestire il Protocollo di Comunicazione lato Master Robot
 - MasterFSM
 - Componente logica che implementa il funzionamento del Master Robot secondo un modello "Finiste State Machine"
 - MasterManager
 - Componente di gestione del Master Robot che inizializza alcuni parametri e che gestisce, mediante chiamate a funzioni degli altri moduli, il corretto funzionamento dell'automa
 - MasterMineSensor
 - Script che gestisce il funzionamento del metal detector installato nel Master Robot
- Player
 - MineCheck
 - Script che verifica la collisione tra il Player e le Mine
 - PlayerSetup
 - Script che si occupa di inizializzare la posizione del Player in base al punto di Start generato all'avvio del simulatore



- Spider
 - SpiderCommunication
 - Script che si occupa di gestire il Protocollo di Comunicazione lato Seeker Robot
 - SpiderManager
 - Componente di gestione del Seeker Robot che prevede l'inizializzazione dell'area di ricerca a cui l'automa sarà associato
 - SpiderMineSensor
 - Componente che gestisce il funzionamento del metal detector installato nel Seeker Robot
- Utils
 - AStar
 - Directory contenente gli script necessari all'implementazione dell'algoritmo A* per la ricerca del percorso ottimo da parte del Master Robot
 - Kalman
 - Directory contenente gli script necessari all'implementazione del Filtro di Kalman utilizzato dai Seeker Robot per individuare la loro posizione nell'ambiente e fornire al Master Robot delle coordinate quanto più precise possibili riguardo la posizione delle mine individuate
 - Security
 - Directory contenente gli script necessari per una corretta implementazione del Protocollo di Comunicazione Sicuro
 - CryptographyManager
 - Componente logica che implementa le funzionalità della Suite Crittografica utilizzata nel Protocollo di Comunicazione
 - PathFinder
 - Componente logica che, con l'ausilio dell'algoritmo A*, genera un percorso verso il punto obiettivo tenendo conto delle Mine registrate dal Master Robot
 - PriorityQueue
 - Implementazione della struttura dati "Coda con Priorità" utilizzata per memorizzare i Nodi da visitare durante la ricerca del percorso ottimo
- EventManager
 - Componente logica che si occupa della trasmissione dei messaggi
- TerrainManager
 - Script che inizializza tutte le informazioni relative alla posizione del punto di start, del punto di evacuazione e delle mine

Altre Componenti

Oltre le componenti principali, già illustrate nelle pagine precedenti, il progetto include ulteriori componenti e oggetti presenti all'interno delle sottodirectory di "Assets". La gerarchia di directory che contiene tali componenti può essere descritta come segue:

- Models
 - Master
 - Directory contenente Materiali e 3DObject relativi al modello 3D del Master Robot
 - Mine
 - Directory contenente Materiali e 3DObject del modello 3D della Mina Antiuomo
 - SciFi Tablet
 - Directory contenente Materiali e 3DObject del modello 3D del Tablet
- Oculus Hands
 - Directory, rilasciata da Oculus, contenente Modelli 3D e Animazioni per la corretta implementazione dei Prefab relativi alla mano destra e alla mano sinistra del giocatore
- VR Body
 - Animations
 - Animazioni dei modelli 3D delle mani relative alle azioni di "Grab" e "Poke"
 - AnimateHandOnInput
 - Script che si occupa di gestire l'animazione della mano in base al valore di input fornite dalle operazioni di "Grab" e "Poke"
- XR/XRI
 - Directory di default contenenti file di configurazione relativi alle interazioni XR
- MapTexture
 - Texture contenente l'output di visualizzazione della MapCamera che può essere utilizzato come Material per visualizzare tale output su un oggetto
- MasterTexture
 - Texture contenente l'output di visualizzazione della MasterCamera che può essere utilizzato come Material per visualizzare tale output su un oggetto
- MapObject
 - GameObject, non renderizzati dalla MainCamera, che vengono visualizzati dalla MapCamera per segnalare le diverse entità visibili dalla minimappa; in particolare, esisterà un MapObject specifico per le seguenti entità
 - Player
 - Master
 - Seeker
 - Path
 - Safe Zone
 - Mine

Scene

Il progetto è composto da due Scene, presenti nel percorso '*Project/Assets/Scenes*', che consentono di avviare e utilizzare l'ambiente di simulazione.

Start Scene

La scena iniziale prevede un menù contenente diversi pulsanti:

- Start
 - Avvia l'ambiente di simulazione passando alla scena "Primary Scene"
- Options
 - Apre un ulteriore menù dove è possibile configurare le impostazioni relative al volume nell'ambiente di simulazione
- About
 - Apre una sezione che mostra i creatori del progetto
- Quit
 - Termina l'applicazione

Primary Scene

La scena principale, dove è possibile utilizzare il simulatore, è composta dal vero e proprio ambiente di simulazione; in particolare, questa scena prevede:

- Terreno
 - Un manto erboso della dimensione di 50x50 celle
- XR Set Up
 - La Componente XR utilizzata dal giocatore
- Lighting
 - Componente di illuminazione
- MapCamera
 - Camera posta a un'altezza elevata che, seguendo il giocatore, fornisce le informazioni note dell'ambiente che verranno poi visualizzate sul Tablet
- Master
 - Master Robot, presente all'avvio della simulazione, che si occuperà poi di istanziare i GameObject relativi ai Seeker Robot
- DangerousSignal
 - Cartelli che rilevano e delimitano la presenza di un campo minato. Questi cartelli sono utilizzati dai Seeker Robot, come landmark, per l'applicazione del Filtro di Kalman

Audio

Nel progetto sono stati utilizzati audio per rendere la simulazione ancora più realistica.

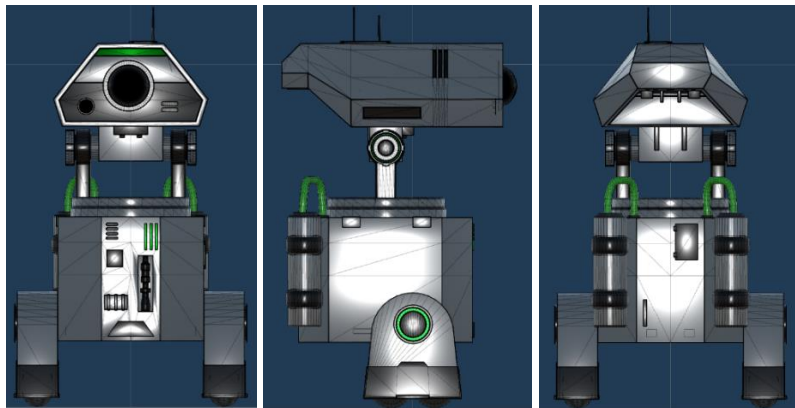
- Menù iniziale
 - uiclick: suono relativo al click di un bottone
 - uihoverenter: suono che viene emesso quando si punta un bottone
- Spider
 - toggle_004: indica che è stata rilevata una mina da uno spider
- Master
 - spaceEngine_000: indica che il Master è in azione per ricevere le informazioni sulle mine da parte degli Spider e può rilevare lui stesso le mine, non ancora individuate dagli Spider, lungo il percorso calcolato con le informazioni note
- Esplosione
 - Explosion: questo audio si attiva quando l'operatore, non seguendo il Master, entra in contatto con una mina. Questo comporta l'esplosione della mina e la fine della simulazione

Hardware

Trattandosi di un ambiente di simulazione, tutte le componenti Hardware sono state fino a ora menzionate sotto la struttura di GameObject; tuttavia, questi elementi potrebbero avere un corrispettivo fisico nel mondo reale e, per questo motivo, è bene menzionarne la struttura e le caratteristiche principali.

Master Robot

Il Master Robot riveste un ruolo fondamentale nella gestione delle operazioni strategiche e della sicurezza all'interno del progetto. La sua capacità di individuare un percorso sicuro verso la zona sicura di estrazione è cruciale in situazioni molto avverse con presenze di mine antiuomo/anticarro, fornisce infatti un vantaggio significativo nelle operazioni di salvataggio dell'operatore in pericolo.



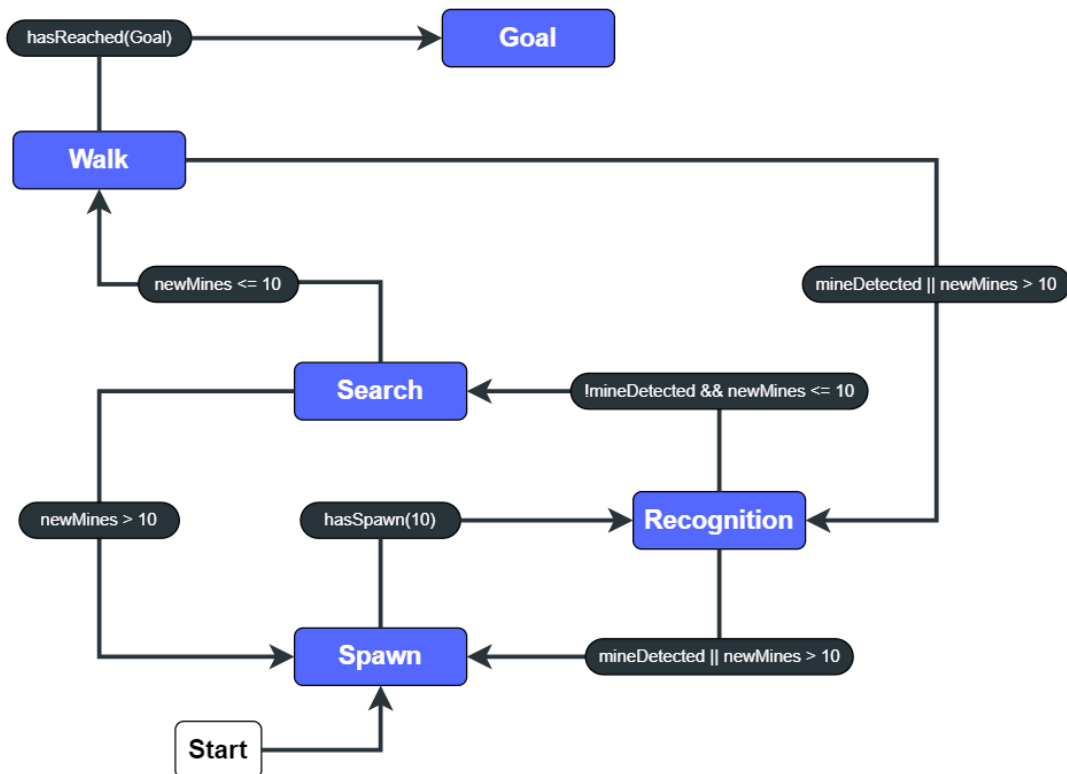
Il Master Robot è dotato di una videocamera, posta nella parte anteriore centrale della testa, che consente all'operatore di vedere in tempo reale ciò che vede il Robot; inoltre, questo è dotato di un metal detector, posto nella parte più bassa del corpo, che consente di individuare la presenza di mine antiuomo nelle vicinanze, anche se sottoterra.

FSM

Una FSM descrive il comportamento di un sistema mediante una serie di stati discreti e transizioni tra di essi. Ogni stato rappresenta una condizione specifica del sistema e le transizioni rappresentano le azioni che cambiano lo stato attuale.

Il Master Robot è stato modellato come una FSM (Finite State Machine) in modo tale da sfruttare i seguenti vantaggi:

- Chiarezza concettuale: La comprensione del comportamento del Robot è semplificata, in quanto il sistema viene suddiviso in stati distinti e le transizioni tra di essi sono chiare e concise.
- Semplicità di progettazione: La struttura a stati consente di progettare il software in modo organizzato e comprensibile. Ogni stato può gestire in modo autonomo un aspetto specifico del comportamento del Robot.



Gli stati e le transizioni illustrati, che vedono come punto di partenza lo stato “Spawn”, sono così progettati:

- **SpawnState:** Stato in cui il Master Robot provvede a generare 10 Spider che manderà ad esplorare il terreno avverso, questi si distribuiranno sul terreno in modo tale da inviare informazioni al Master Robot riguardo le posizioni in cui vengono rilevate le mine antiuomo
- **RecognitionState:** Stato in cui il Master Robot valuta la pericolosità del terreno in base alle informazioni acquisite sulle mine scoperte fino a quel momento, le nuove mine scoperte in un periodo di tempo di qualche secondo e la possibile presenza di mine vicine al Robot stesso
- **SearchState:** Stato in cui il Master Robot reputa di poter cominciare ad effettuare una ricerca del possibile percorso da percorrere per portare l’operatore in salvo. Il cammino trovato non importa che sia minimo, ciò che è importante è che sia sicuro. Inoltre, è possibile che il percorso trovato passi sopra qualche mina non ancora scoperta, ma in quel caso il metal detector del Robot permetterà di individuarla per farlo istantaneamente indietreggiare
- **WalkState:** Stato in cui il Master Robot, dopo aver determinato il percorso ottimo, comincia a muoversi verso il punto di arrivo obiettivo, scansionando il percorso di fronte a sé con l’utilizzo del metal detector al fine di rilevare eventuali mine non segnalate
- **GoalState:** Stato in cui il Master Robot ha raggiunto il punto di estrazione dell’operatore. Il Robot è riuscito nell’operazione a lui assegnata

Ricerca

La ricerca del percorso ottimo, effettuata durante la permanenza in 'SearchState', si basa su un'implementazione di A* che tiene conto, oltre che del percorso minimo, delle mine presenti nel territorio.

Tuttavia, i dettagli riguardo quest'implementazione verranno discussi nei capitoli seguenti.

Seeker Robot

Il Seeker Robot nasce con lo scopo di semplificare le operazioni svolte dal Master Robot, andando alla ricerca delle mine presenti sul territorio, segnalandole prontamente al Master che sarà così in grado di calcolare il percorso ottimo massimizzando le sue informazioni riguardo le zone pericolose presenti nel terreno.



I Seeker Robot (Spider Robot) sono dotati di otto zampe, che consentono loro di muoversi abilmente in qualunque terreno distribuendo equamente il peso dell'intera struttura, e di un metal detector, implementato direttamente nella struttura del corpo, che gli consente di individuare le mine presenti in un'area ristretta attorno al proprio sensore.

Movimento

Lo Spider Robot, quando viene spawnato dal Master Robot, riceve indicazioni sull'area da perlustrare. In quell'area provvederà a setacciare casualmente il terreno in modo da rilevare delle mine nascoste sotto di esso.

Si usano gli Spider Robot per perlustrare il terreno a causa del suo trascurabile peso che non attiverà le mine qualora ci passasse sopra.

Comunicazione

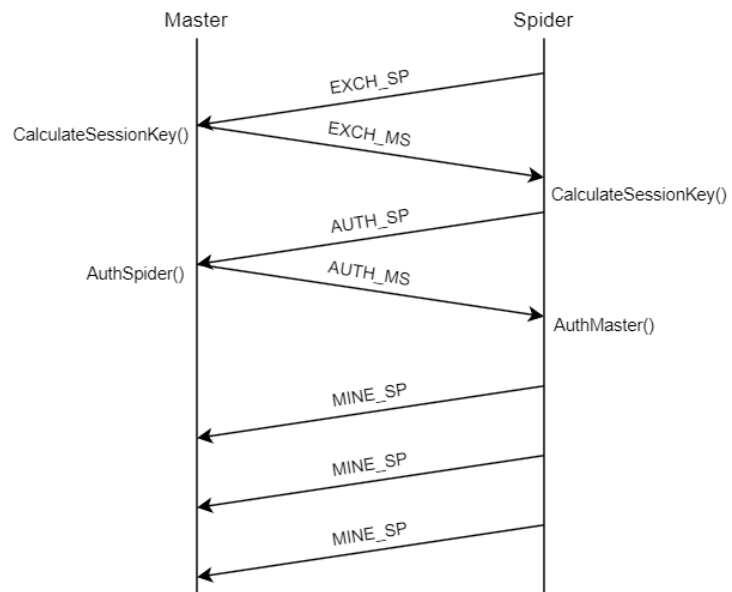
Ogni qualvolta lo Spider Robot rileva una mina, questo manderà una comunicazione al Master con la posizione in cui ha rilevato la mina.

La comunicazione viene protetta dalla cifratura con chiavi master e di sessione, opportunamente custodite/generate, in modo tale da non permettere ad un eventuale attacco informatico di segnalare false mine o non permettere in qualche modo al Master Robot di calcolare un percorso che porti l'operatore in una zona sicura.

Protocollo di comunicazione

La figura a fianco mostra un diagramma di sequenza relativo all'ordine cronologico dei messaggi scambiati secondo il formato imposto dal Protocollo di Comunicazione.

Di seguito saranno descritti i messaggi scambiati e la loro utilità ai fini della comunicazione.



EventManager

Un EventManager è un pattern di progettazione che facilita e gestisce la comunicazione tra gli oggetti di un'applicazione o di un gioco. Questo pattern viene spesso utilizzato in Unity per consentire la trasmissione di messaggi o eventi tra più GameObject senza che questi siano legati tra loro.

Questo approccio offre:

- Disaccoppiamento degli oggetti: Riduce la dipendenza diretta tra gli oggetti, rendendo il codice più semplice da mantenere.
- Organizzazione e centralizzazione: Fornisce un punto centrale per la gestione degli eventi per facilitare la manutenibilità e la leggibilità del codice.
- Modularità e scalabilità: Permette di gestire facilmente le comunicazioni anche in progetti complessi e consente l'aggiunta di nuove funzionalità senza alterare il codice esistente.

Sicurezza

Il Robot Master e il Robot Seeker condividono una chiave master di cifratura (MasterKey, MasterIV), che è unica per ogni Seeker. Questa chiave viene utilizzata per scambiare dati utili alla creazione di una chiave di sessione (SessionKey, SessionIV) da utilizzare per cifrare le successive comunicazioni riguardanti le posizioni delle mine rilevate dai Seeker.

Generare una chiave di sessione da una chiave master è una pratica comune in quanto è necessario considerare i seguenti aspetti:

- Unicità della chiave master: Nel caso in cui un attacco informatico riuscisse a recuperare la chiave master, a partire dai messaggi cifrati scambiati tra le parti, tutti i messaggi precedenti e successivi saranno inevitabilmente compromessi.
- Limitazione temporale: Le chiavi di sessione vengono generate con lo scopo di avere un periodo di tempo di validità. Nel caso in cui venga scoperta la chiave di sessione, verranno compromessi solo i messaggi scambiati in un breve periodo di tempo.

Messaggi

Di seguito saranno dettagliati i messaggi scambiati tra Master Robot e Seeker Robot al fine di stabilire i parametri necessari a raggiungere una comunicazione sicura.

EXCH_SP

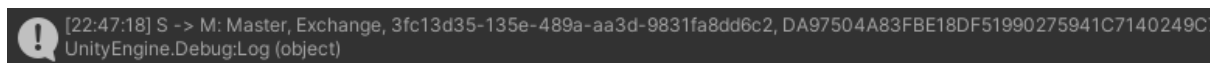
Il messaggio EXCH_SP viene inviato da uno Spider Robot per iniziare lo scambio di messaggi con il Master Robot finalizzato alla generazione della chiave di sessione e successiva autenticazione.

Questo messaggio può essere mandato quando viene creato lo Spider Robot oppure quando è necessario cambiare chiave di sessione.

Lo Spider Robot genera un nonce e lo manda al Master Robot in un messaggio composto come segue:

$$S \rightarrow M: Master || Exchange || ID_S || E[K_{MS}, ID_S || Nonces_S]$$

Alla ricezione del messaggio, il Master Robot controlla che il messaggio sia ben formato e che il messaggio decifrato con la chiave master contenga effettivamente l'ID dello Spider.



N.B.: I messaggi che contengono testo cifrato sono stati tagliati per motivi grafici, in quanto generando nonce di tipo uuid si gestiscono messaggi abbastanza lunghi (come mostrato in figura), ma il vantaggio sta nel fatto che la probabilità di generare due volte lo stesso nonce è praticamente nulla.

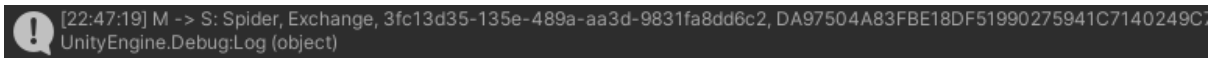
EXCH_MS

Il messaggio EXCH_MS viene inviato dal Master Robot dopo aver ricevuto un messaggio EXCH_SP da parte di uno Spider Robot.

Il Master Robot genera un nonce e lo manda allo Spider Robot in un messaggio composto come segue:

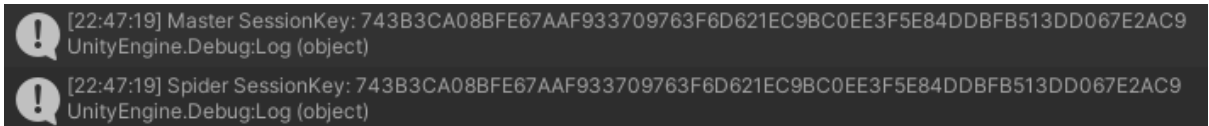
$$M \rightarrow S: Spider || Exchange || ID_S || E[K_{MS}, ID_S || Nonce_M]$$

Alla ricezione del messaggio, lo Spider Robot controlla che il messaggio sia ben formato e che il messaggio decifrato con la chiave master contenga effettivamente l'ID dello Spider.



[22:47:19] M -> S: Spider, Exchange, 3fc13d35-135e-489a-aa3d-9831fa8dd6c2, DA97504A83FBE18DF51990275941C7140249C7
UnityEngine.Debug:Log (object)

Successivamente agli scambi di EXCH_SP e EXCH_MS, sia il Master Robot sia lo Spider Robot calcolano la chiave di sessione (SessionKey, SessionIV) con cui verranno cifrati i prossimi messaggi.



[22:47:19] Master SessionKey: 743B3CA08BFE67AAF933709763F6D621EC9BC0EE3F5E84DDBFB513DD067E2AC9
UnityEngine.Debug:Log (object)
[22:47:19] Spider SessionKey: 743B3CA08BFE67AAF933709763F6D621EC9BC0EE3F5E84DDBFB513DD067E2AC9
UnityEngine.Debug:Log (object)

AUTH_SP

Il messaggio AUTH_SP viene inviato dallo Spider Robot dopo aver ricevuto un messaggio EXCH_MS da parte del Master Robot.

Dopo aver generato la chiave di sessione, lo Spider Robot manda al Master Robot un messaggio composto come segue:

$$S \rightarrow M: Master || Auth || ID_S || E[K_{SS}, ID_S || Nonce_M]$$

Lo Spider Robot, con il suddetto messaggio, persegue i seguenti obiettivi:

- Dimostrare la sua identità in quanto ha potuto decifrare correttamente il ticket cifrato con la chiave master nel messaggio EXCH_MS. N.B.: lo Spider Robot è l'unico possessore della chiave master oltre il Master Robot.
- Dimostrare di aver calcolato correttamente la chiave di sessione in quanto, se il Master Robot riuscirà a decifrare il ticket contenuto in questo messaggio sarà sicuro che lo Spider Robot ha calcolato la stessa chiave di sessione.



[22:47:20] S -> M: Master, Auth, 3fc13d35-135e-489a-aa3d-9831fa8dd6c2, CF7B54116F76E59414D8B9458297C8979E8200C57
UnityEngine.Debug:Log (object)


AUTH_MS

Il messaggio AUTH_MS viene inviato dal Master Robot dopo aver ricevuto un messaggio AUTH_SP da parte dello Spider Robot.

Dopo aver generato la chiave di sessione, il Master Robot manda allo Spider Robot un messaggio composto come segue:

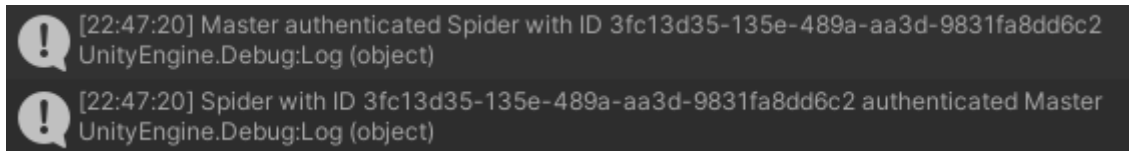
$$M \rightarrow S: Spider || Auth || ID_S || E[K_{SS}, ID_S || Nonce_S]$$

Gli obiettivi perseguiti dal suddetto messaggio sono i medesimi del messaggio AUTH_SP a parti inverse.



[22:47:20] M -> S: Spider, Auth, 3fc13d35-135e-489a-aa3d-9831fa8dd6c2, CF7B54116F76E59414D8B9458297C8979E8200C57
UnityEngine.Debug:Log (object)

Successivamente agli scambi di AUTH_SP e AUTH_MS, sia il Master Robot sia lo Spider Robot si autenticano scambiandosi l'nonce generato dalla controparte cifrato con la SessionKey.



[22:47:20] Master authenticated Spider with ID 3fc13d35-135e-489a-aa3d-9831fa8dd6c2
UnityEngine.Debug:Log (object)
[22:47:20] Spider with ID 3fc13d35-135e-489a-aa3d-9831fa8dd6c2 authenticated Master
UnityEngine.Debug:Log (object)

Da questo momento in poi la comunicazione tra Master e Spider è stabilita e quest'ultimo può inviare messaggi al Master riguardanti le posizioni delle mine rilevate.

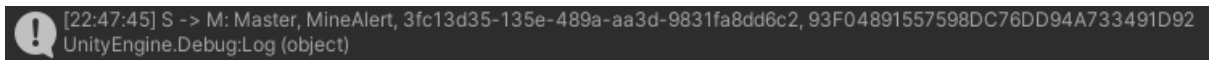
MINE_SP

Il messaggio MINE_SP viene inviato dallo Spider Robot ogni volta che rileva una mina mediante l'uso del suo sensore.

Le informazioni contenute in questo messaggio servono al Master Robot per costruire una conoscenza del terreno che dovrà percorrere per portare l'operatore a destinazione.

Il messaggio è composto come segue:

$$S \rightarrow M: Master || MineAlert || ID_S || E[K_{SS}, Position]$$



[22:47:45] S -> M: Master, MineAlert, 3fc13d35-135e-489a-aa3d-9831fa8dd6c2, 93F04891557598DC76DD94A733491D92
UnityEngine.Debug:Log (object)

Alla ricezione del messaggio, il Master Robot provvederà a decifrare il ticket con la chiave di sessione e ad aggiornare la sua conoscenza del terreno.

Suite Crittografica

La suite crittografica su cui si basa il Protocollo di Comunicazione Sicuro comprende l'utilizzo di:

- Algoritmo di Cifratura Simmetrico
- Funzione Hash
- Certificati

Algoritmo di Cifratura

L'algoritmo di cifratura utilizzato dal protocollo è 'AES-256' che necessita, grazie a un'adeguata dimensione della chiave, fornisce un certo livello di robustezza nella cifratura.

Inoltre, l'algoritmo prevede l'utilizzo di un Vettore di Inizializzazione della dimensione pari a 16 Byte.

Funzione Hash

Per quanto riguarda la Funzione Hash, utilizzata per la generazione della Chiave di Sessione, è stato scelto di utilizzare l'algoritmo 'SHA-256' che genera un Digest di dimensione pari a 256 bit.

La scelta di una funzione hash così robusta è dovuta al fatto che se un attaccante individuasse una chiave di sessione scaduta, questo dovrebbe sostenere un costo computazionalmente troppo elevato per ricostruire la Chiave Master a partire dalla quale è stata generato il Digest.

Certificati

Al fine di strutturare adeguatamente l'associazione tra un Seeker ID e i suoi parametri crittografici, sono stati implementati dei Certificati, posseduti da ogni Seeker Robot, che contengono:

- Spider ID
 - Identificatore univoco (UUID) del Seeker Robot a cui è associato il Certificato
- Master Key
 - Chiave Master condivisa da quel particolare Seeker Robot e il Master Robot
- Master IV
 - Vettore di Inizializzazione condiviso da quel particolare Seeker Robot e il Master Robot

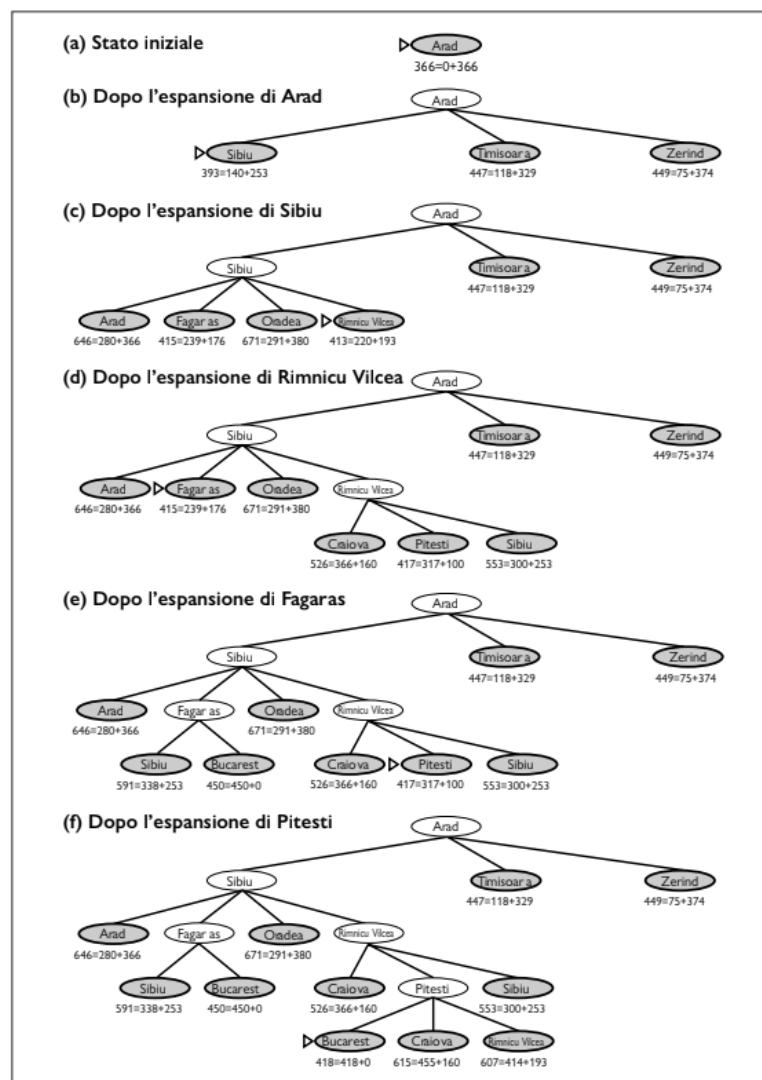
Algoritmi

Alcune delle principali funzionalità logiche hanno presentato la necessità di implementare degli algoritmi specifici; in particolare, gli algoritmi illustrati nelle righe seguenti consentono ai Robot di operare secondo approcci legati all'area dell'Intelligenza Artificiale e della Robotica Probabilistica.

Ricerca A*

L'algoritmo di ricerca A* è un metodo di ricerca ampiamente utilizzato per trovare il percorso ottimale tra due punti in un grafo. Combina la precisione della ricerca informata con l'efficacia della ricerca greedy, valutando il costo effettivo fino a quel momento e una stima del costo rimanente per raggiungere la destinazione.

Figura 3.18
Passi di una ricerca A* di un itinerario verso Bucarest. I nodi sono etichettati con i valori $f = g + h$. I valori h sono distanze in linea d'aria verso Bucarest, prese dalla Figura 3.16.



Le performance e l'ottimalità dell'algoritmo sono influenzate dalla scelta euristica della funzione h . Se l'euristica è ammissibile (ovvero, non sovrastima mai il costo reale per raggiungere la destinazione), allora l'algoritmo A* garantisce che il percorso trovato è il percorso ottimo, quindi con il costo minimo.

Utilità

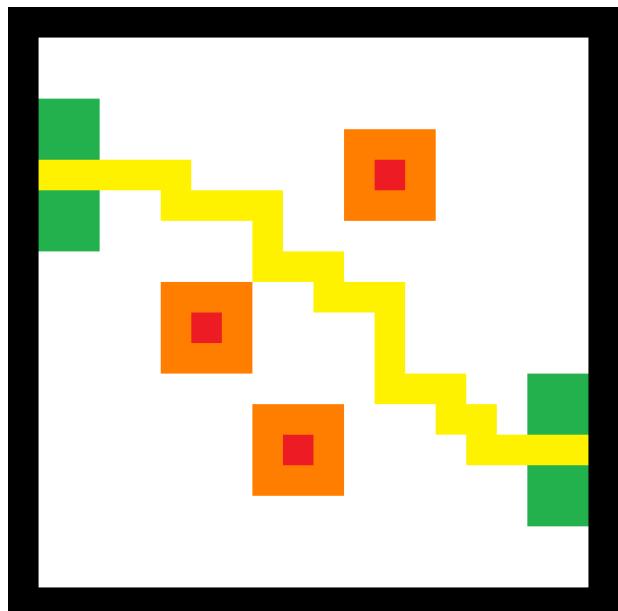
In questa ricerca, i punti fondamentali sono:

- Trovare un percorso sicuro
 - L'operatore non deve essere mai esposto ad un pericolo elevato perché, nonostante ci sia il Master Robot a guidarlo, le azioni umane sono imprevedibili e servono dei margini di errore
- Portare via l'operatore dal pericolo nel più breve tempo possibile
 - Il percorso trovato deve essere il più breve possibile per minimizzare il tempo di attraversamento
 - Il percorso deve essere calcolato dal Master Robot nel più breve tempo possibile

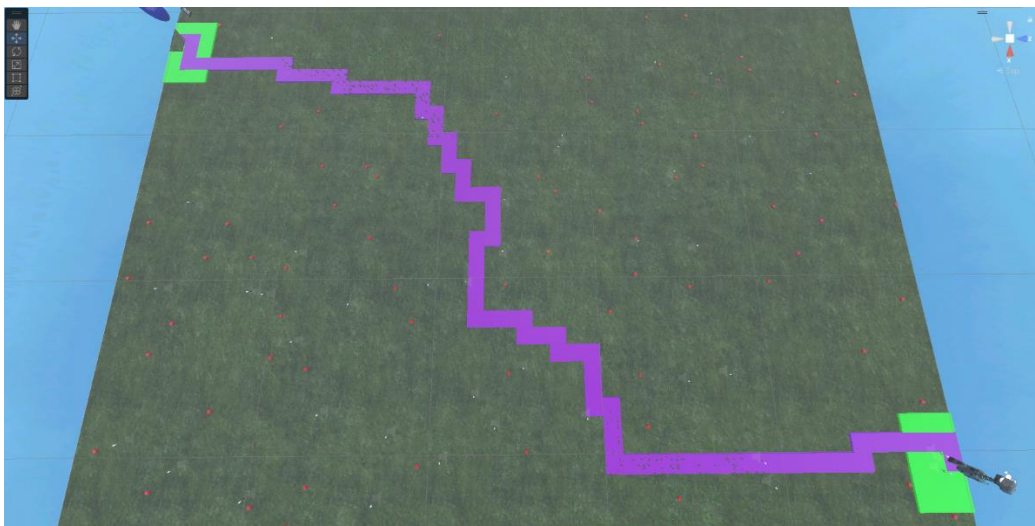
In questo scenario è stata utilizzata l'euristica Manhattan, che consente di identificare il cammino minimo, in quanto è ammissibile, abbastanza rapidamente.

Lo schema in figura mostra l'idea implementativa:

- Safe Zones (Verde)
 - Delimitano le zone sicure in cui sono situati il punto di inizio e il punto di evacuazione
- Alert Zones (Arancione)
 - Delimitano l'area circostante le minee individuate
- Path (Giallo)
 - Indica il percorso, suddiviso in celle, che è stato generato dal Master Robot



Il risultato effettivo è il seguente:



È possibile osservare (effettuando uno zoom) che in rosso sono presenti le mine, in verde ci sono le safe zones e in viola il percorso effettuato dal Master Robot.

Implementazione

L'algoritmo A* è stato implementato con l'utilizzo di due script:

- Node
 - Con questa classe viene modellato il concetto di Nodo utile ad effettuare la ricerca A* su grafo.
 - I Nodi contengono informazioni sulla posizione, i costi (g, h, f) e il riferimento al Nodo padre.
- AStar
 - I dati necessari per la ricerca A* sono la mappa e la grandezza di essa.
 - Per ottimizzare la ricerca è stata implementata la PriorityQueue con la struttura dati Heap Binario.

Pseudocodice:

- lista_aperta = []
- lista_chiusa = []
- Aggiungi start a lista_aperta
- Finché lista_aperta non è vuota:
 - nodo_corrente = nodo estratto da lista_aperta con f minimo
 - Aggiungi nodo_corrente a lista_chiusa
 - Se nodo_corrente è goal:
 - Restituisci il percorso
 - Per ogni nodo_adiacente a nodo_corrente:
 - Se nodo_adiacente non è in lista_chiusa:
 - Aggiorna i costi di nodo_adiacente
 - Aggiungi nodo_adiacente a lista_aperta

Filtro di Kalman

Il Filtro di Kalman è un filtro ricorsivo che determina lo stato di un sistema dinamico considerando delle misurazioni soggette a rumore.

Nell'ambito della Robotica questo filtro può essere utilizzato, per esempio, per determinare la posizione di un Robot in seguito a un'azione di movimento, stimando la nuova posizione sulla base della distanza (afflitta da errore) tra il Robot e più oggetti denominati "landmark".

A partire dalla nuova misura, calcolata sulla base dell'azione compiuta dal Robot, e dalla misurazione riguardante la distanza dai landmark, il Filtro di Kalman è in grado di ricostruire una misura di posizione più accurata come prodotto delle Gaussiane rappresentanti le due stime di partenza.

Utilità

Nell'ambito del progetto, il Filtro di Kalman è stato utilizzato per fare in modo che i Seeker Robot, una volta individuata una mina, fossero in grado di segnalare la propria posizione con una certa precisione, al fine di evitare segnalazioni con margine di errore troppo ampi.

In particolare, i Seeker Robot sfruttano come landmark dei cartelli di pericolo, che indicano la presenza di un campo minato, posti ai quattro angoli del terreno e a partire dai quali è possibile effettuare una triangolazione per determinare una posizione quanto più precisa possibile.

Implementazione

Il filtro di Kalman è stato implementato suddividendo i tre passaggi fondamentali:

- Inizializzazione
 - Definizione delle matrici di stato, transizione, misurazione e covarianza.
- Predizione
 - Utilizzo delle equazioni del filtro di Kalman per predire lo stato successivo e la covarianza.
- Correzione
 - Aggiornamento dello stato e della covarianza.

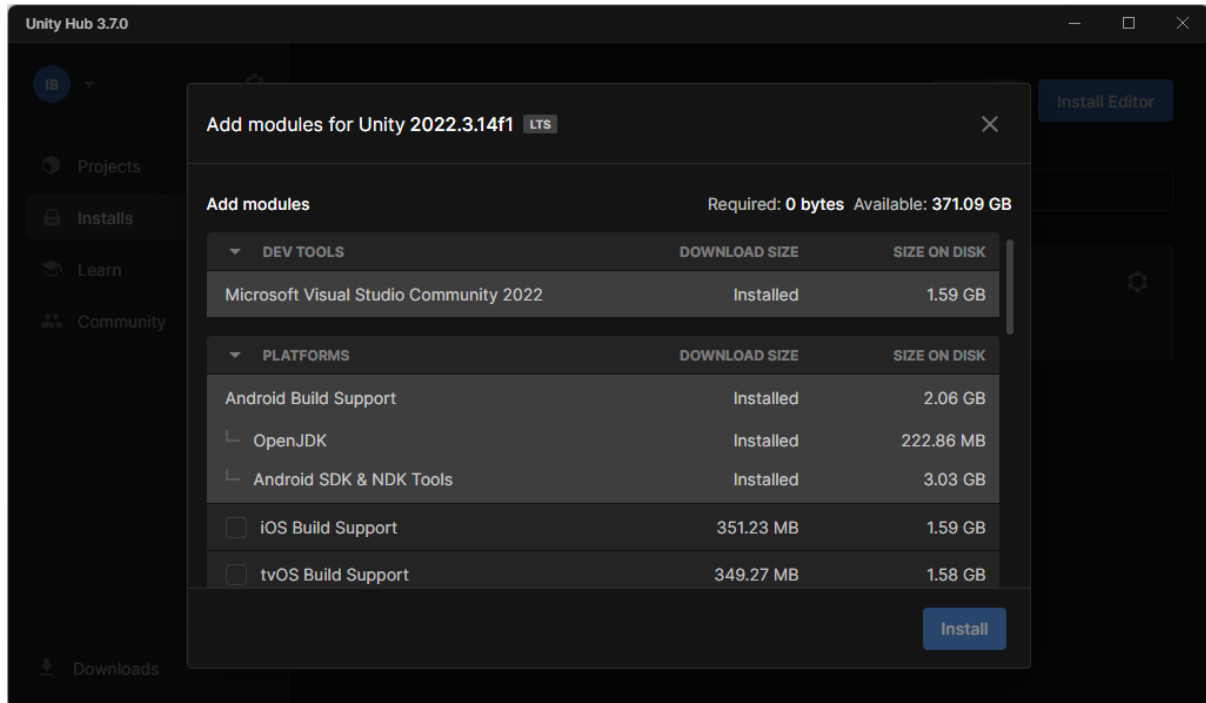
Inoltre, si è deciso di modellare la matrice di stato in modo tale da non tener conto dell'altezza (in quanto siamo in presenza di un terreno piatto) e dell'accelerazione all'interno dello stato.

Pseudocodice:

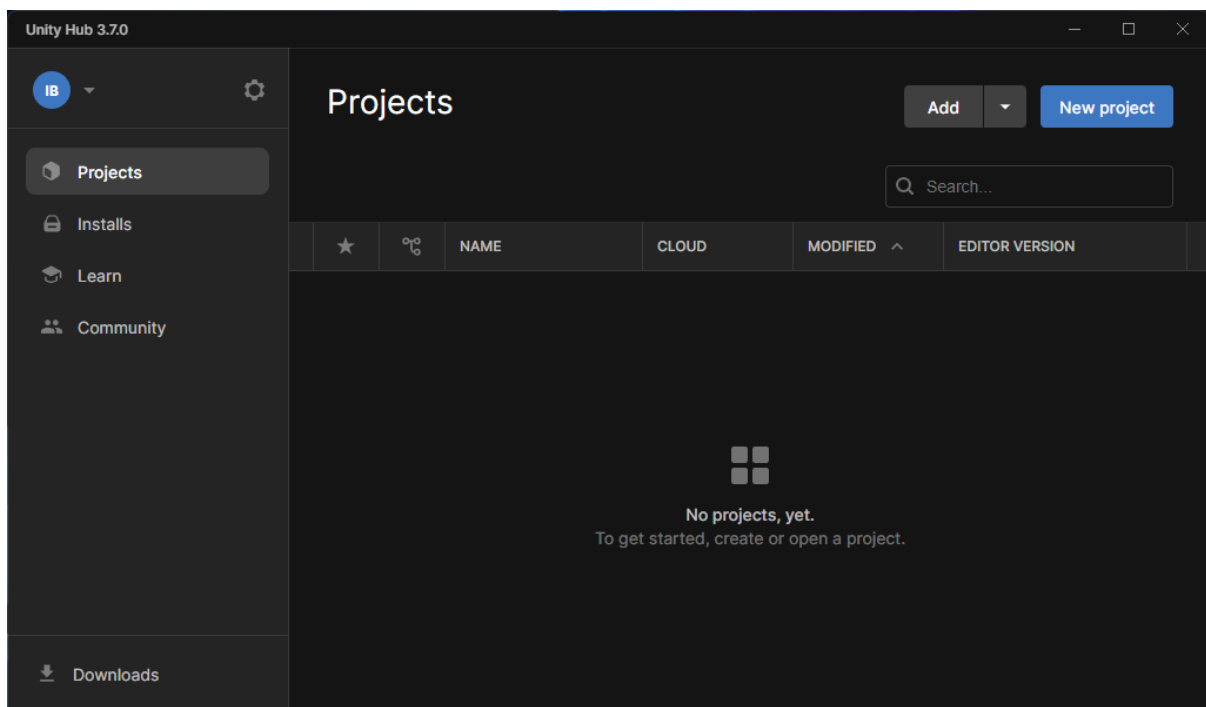
- Inizializzazione:
 - $x = \text{stato iniziale}$
 - $S = \text{covarianza iniziale}$
 - $A = \text{matrice di transizione}$
 - $C = \text{matrice di misurazione}$
 - $R = \text{covarianza del processo}$
 - $Q = \text{covarianza di misurazione}$
- Predizione:
 - $x_{pred} = Ax$
 - $S_{pred} = ASA^T + R$
- Correzione:
 - $K = S_{pred}C^T(CSC^T + Q)^{-1}$
 - $x = x_{pred} + K(z - Cx_{pred})$
 - $S = (I - KC)S_{pred}$

Guida al Deploy

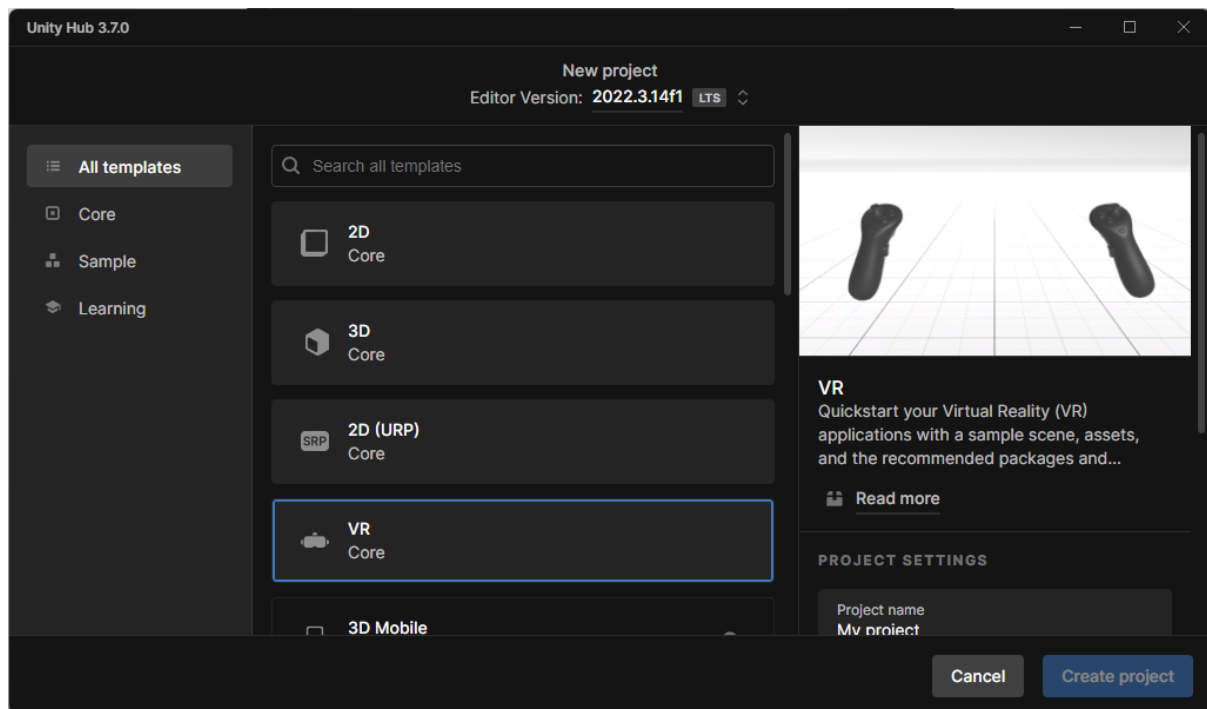
Nel caso in cui si volesse deployare il progetto su un visore VR, il prerequisito è aver installato il modulo **Android Build Support**. Per farlo basta andare nella tab **Installs**, cliccare sull'icona dei Settings riguardanti l'Editor e selezionare **Add modules**.



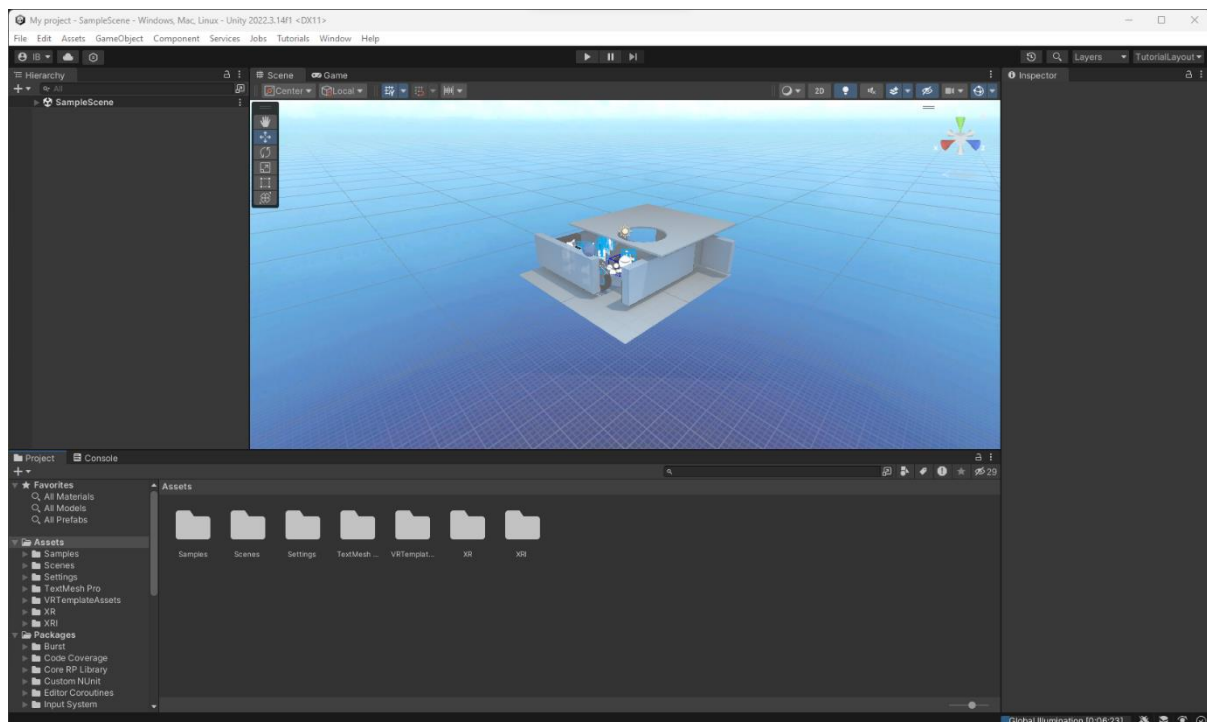
Il deploy del progetto è molto semplice, serve prima di tutto creare un progetto VR su Unity cliccando su **New project**.



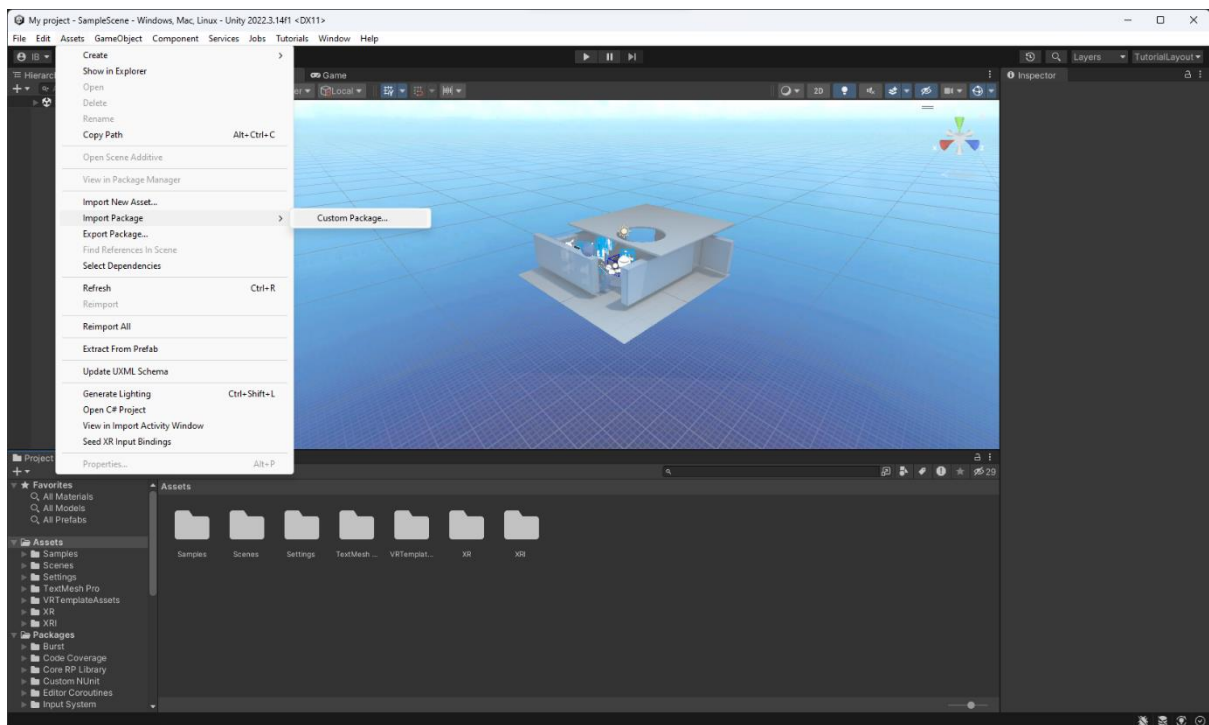
Selezionare il template VR, il nome del progetto e la sua location. Se non si è mai creato un progetto VR dovrà essere fatto il download del template tramite l'apposito pulsante che apparirà a destra. In fine, cliccare su **Create project** per creare il template del progetto.



Il progetto impiegherà qualche minuto per crearsi, il template si presenterà come mostrato di seguito.

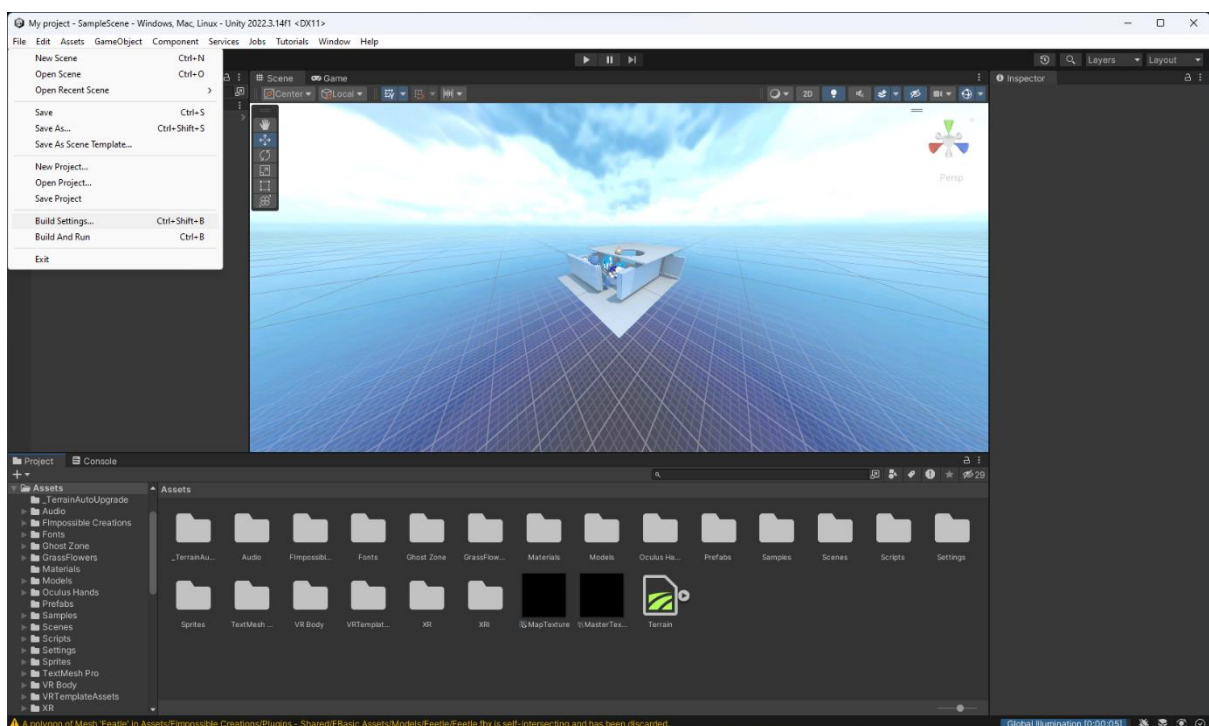


Per importare il progetto basterà seguire il seguente passaggio, si aprirà il selettore di file di Windows per selezionare il pacchetto Unity fornito.



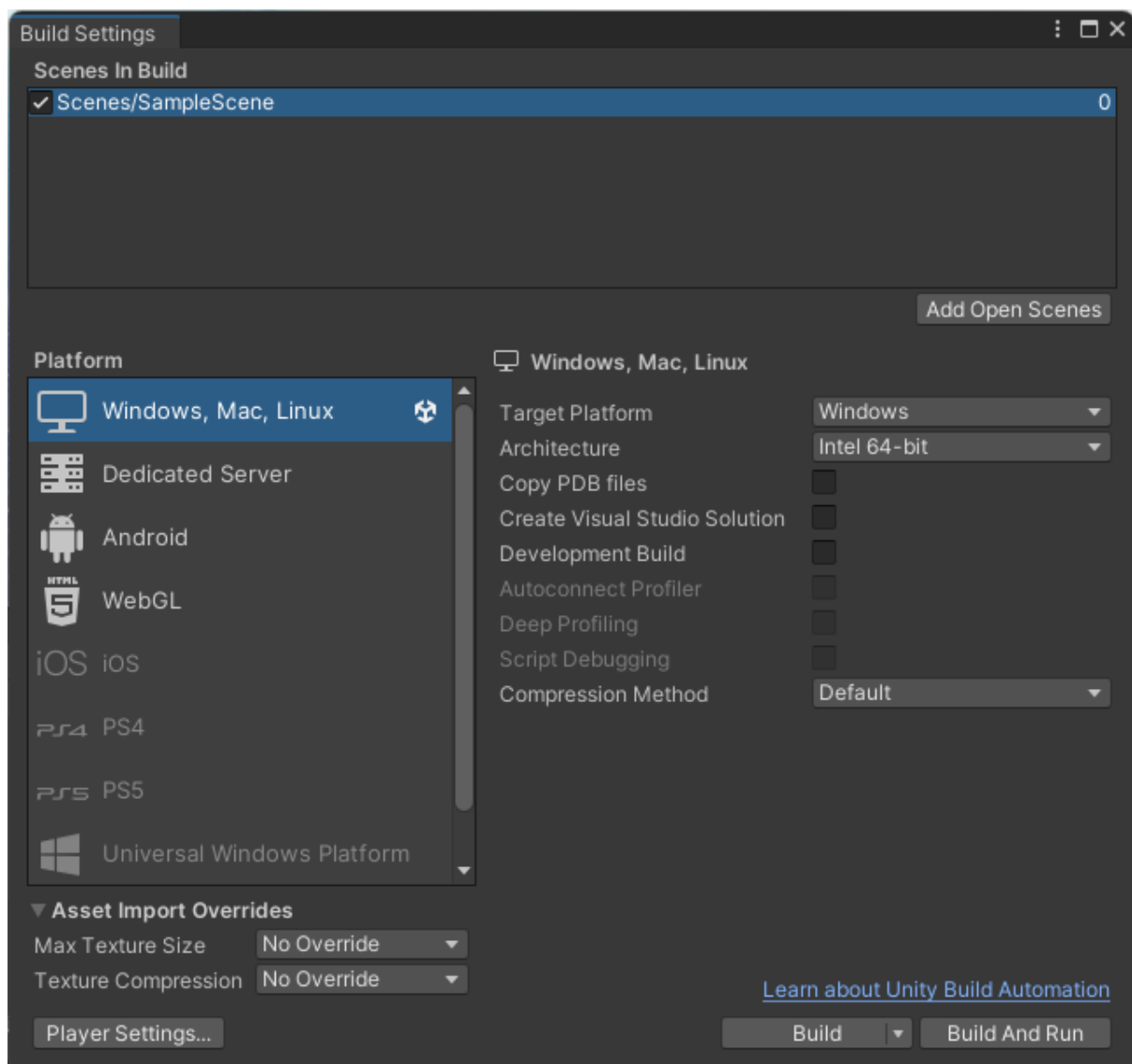
Dopo aver importato tutti gli asset, Unity potrebbe causare un crash, ma l'import sarà stato effettuato con successo. Quindi qualora dovesse succedere quanto descritto, basta riaprire il progetto.

Per ultima cosa, serve selezionare correttamente le scene del progetto andando nella seguente sezione.



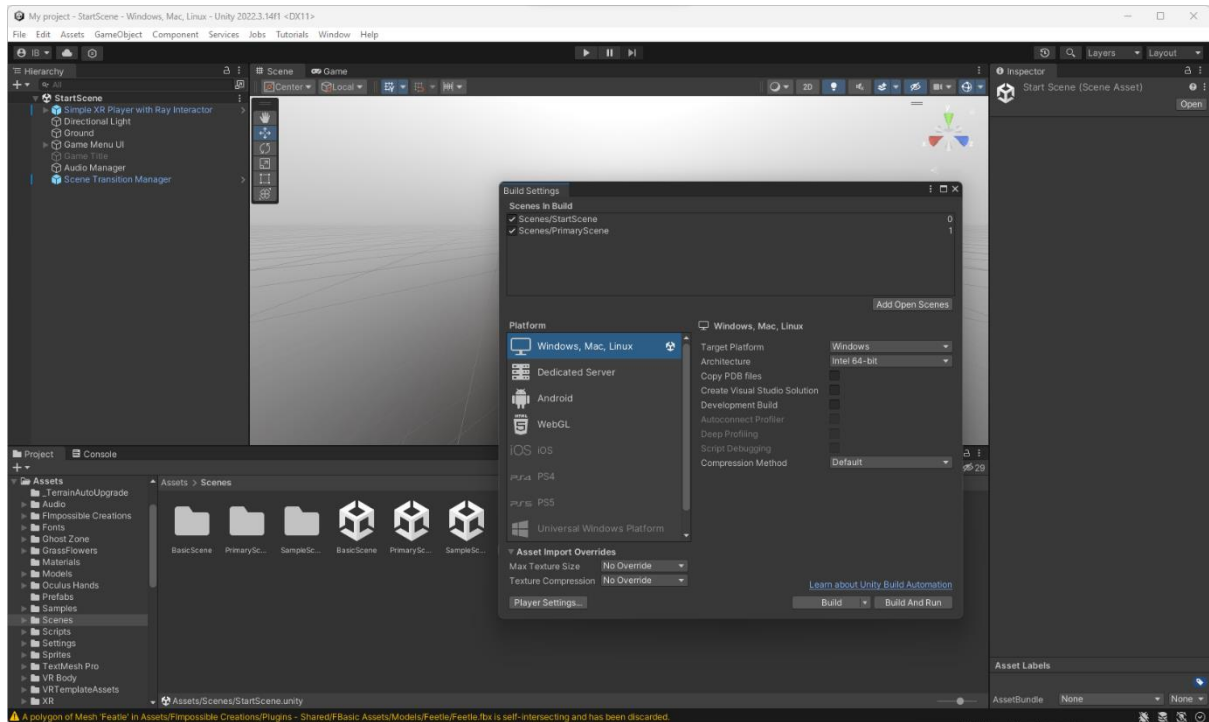
Si aprirà il seguente menù con “SampleScene” che va eliminata semplicemente selezionandola e cliccando il tasto Canc. Adesso i passaggi da seguire sono i seguenti:

- Aprire StartScene (directory Assets/Scene)
- Riaprire Build Settings e cliccare “Add Open Scenes”
- Aprire PrimaryScene
- Riaprire Build Settings e cliccare “Add Open Scenes”
- Nel caso in cui si voglia deployare su un visore, effettuare la Build impostando come “Platform” la voce “Android”, se no non è necessaria
- Infine, tornare su StartScene



Il risultato finale deve essere il seguente:

- StartScene aperta
- Build Settings:
 - StartScene 0
 - PrimaryScene 1



Adesso è possibile chiudere Build Settings e testare il progetto direttamente all'interno di Unity.

Per quanto riguarda ciò, è tuttavia necessario specificare che per alcune funzionalità è necessario utilizzare un dispositivo VR; per esempio, Unity non consente di uscire dall'applicazione mediante il tasto "Exit" del menù principale.

Test

Le fasi di Test sono state fondamentali, durante tutta la realizzazione del progetto, al fine di:

- Verificare la correttezza delle funzionalità implementate
- Migliorare la User Experience
- Determinare possibili migliorie da apportare

Tutte le componenti del progetto, di fatto, sono sempre state testate in due modi:

- Headset Simulator
 - Opzione offerta da Unity che consente di utilizzare Mouse e Tastiera per simulare la presenza di un Headset VR al fine di effettuare test veloci
- Headset VR
 - Le stesse operazioni di test condotte con l'utilizzo dell'Headset Simulator sono poi state condotte nuovamente, una volta raggiunto un certo grado di soddisfacimento riguardo l'implementazione delle funzionalità, con l'utilizzo di un Headset VR dedicato; di fatto, Unity offre la possibilità di interagire con l'applicazione in fase di progettazione direttamente con l'attrezzatura VR, rendendo i test semplici e veloci

Attrezzatura

L'attrezzatura utilizzata per condurre i test può essere categorizzata come segue:

- Software
 - Unity
 - Utilizzato per avviare l'applicazione in fase di sviluppo interfacciandola, se necessario, con l'Headset VR
 - Oculus
 - Applicazione rilasciata da Meta per offrire la possibilità di interfacciare l'Headset VR con il Computer
- Dispositivi
 - Headset VR
 - L'Headset VR utilizzato è l'Oculus Quest 2, il quale può operare in modalità stand-alone o mediante collegamento a Computer esterni
 - Controller
 - Per quanto riguarda le interazioni nell'ambiente VR, queste avvengono mediante l'utilizzo dei controller "Oculus Touch" la cui scelta, rispetto al Tracking delle Mani, è dovuta alla possibilità di eseguire il movimento del personaggio mediante levetta analogica

Conclusioni

Facendo un breve resoconto, lo sviluppo del progetto è stato terminato nei tempi stimati e con il raggiungimento di tutti gli obiettivi principali prefissati; inoltre, la collaborazione del gruppo ha portato all'implementazione e al miglioramento di alcune componenti logiche non inizialmente considerate.

L'intera fase di implementazione, inoltre, ha dato a ognuno di noi la possibilità di esplorare un ambiente sconosciuto, consentendo l'apprendimento di una metodologia di sviluppo del software completamente nuova.

Infine, la scelta dello sviluppo del progetto in un ambiente VR è stata principalmente spinta dall'interesse comune a esplorare, anche se in un ambiente ristretto, le possibilità che il Metaverso e, più in generale, le applicazioni XR potrebbero offrire.



Uno speciale ringraziamento da parte del Team di Sviluppo di "Rescue Bots"

Ignazio Daniele Di Blasi,
Domenico Giosuè,
Chiara Lupo