Politecnico di Torino

Ingegneria informatica

Tesi di laurea magistrale

Sviluppo di un'Applicazione di Realtà Virtuale Immersiva per l'Esplorazione Terrena

Relatori

Prof. Andrea Sanna

Candidato

Alberto Alessio

Aprile, 2018

Abstract

Il progetto di tesi si colloca nell'ambito della realtà virtuale e si pone come obiettivo principale lo sviluppo di un'applicazione stereoscopica 3D per l'esplorazione terrena, nello specifico rivolta all'esplorazione di terreni marziani.
I principali requisiti e obiettivi che si intende raggiungere per riuscire a fornire un supporto reale per le operazioni di esplorazione sono: semplificare la comprensione di dati scientifici attraverso una corretta visualizzazione e rappresentazione; riprodurre una simulazione realistica delle azioni compiute da un rover nello spostamento su un terreno; fornire le funzionalità che consentano una pianificazione di percorsi attraverso la definizione di una serie di waypoint; realizzare l'applicazione in modo tale che sia versatile e facilmente adattabile a diversi dati e funzionalità sviluppabili in futuro. Il progetto è realizzato con il software Unity3D, Blender e utilizzando la libreria GDAL per la gestione di file geospaziali.
Il primo passo affrontato nella realizzazione dell'ambiente virtuale è stato la generazione di un modello 3D per ogni terreno esplorabile, realizzato utilizzando i file DTM (Digital Terrain Model), ottenuti dal sito HiRISE, la cui peculiarità è costituita dall'elevata risoluzione.
La parte più complessa è costituita dalle funzionalità che sono state realizzate. All'utente vengono fornite le informazioni riguardanti: posizione del rover, elevazione del terreno in quel punto e rotazione del corpo del rover rispetto al sistema di riferimento globale. Con la finalità di consentire la visualizzazione di altre tipologie di informazioni che riguardano il sito che si sta visitando, è stata realizzata la funzione di selezione di uno specifico layer applicabile al terreno. Attraverso un caricamento dinamico effettuato durante l'esecuzione dell'applicazione, viene definito un elenco composto di alcune texture appositamente elaborate (es. mappe altimetriche, immagini orbitali) e modalità di visualizzazione (es. wireframe), dal quale è possibile selezionare il layer desiderato.
Una delle funzioni più importanti al fine di rendere possibile la pianificazione dei percorsi è quella che permette di specificare un target da far raggiungere al rover. Sono state definite due modalità: una che permette la definizione dell'obiettivo inserendo le coordinate in latitudine e longitudine, l'altra cliccando con il tasto destro del mouse su un punto del terreno. Attraverso la funzionalità appena descritta è possibile far raggiungere un singolo target al rover, mentre per la definizione di un percorso è stata realizzata una specifica funzione, che prevede che l'utente selezioni diversi punti, definiti POI (Point Of Interest). È possibile raggiungere i punti singolarmente, eliminarli o seguire l'intero percorso specificato, in questo caso il rover procederà in ordine di definizione dei POI e proseguirà fino a che anche l'ultimo obiettivo non viene raggiunto.
Una volta terminato lo sviluppo dell'applicazione è stato chiesto a un gruppo di dodici persone, comprendente esperti del settore e non, di fare una prova di tutte le funzionalità e di rispondere a un questionario. Nel complesso, i risultati ottenuti al completamento dello sviluppo dell'applicazione sono soddisfacenti, come dimostrato anche dai test effettuati; sicuramente sviluppi futuri, l'aggiunta di nuove funzionalità e il perfezionamento di quelle esistenti possono portare a miglioramenti che ne amplierebbero ulteriormente i campi d'applicazione.

Indice

Elenco delle figure

Elenco delle tabelle

Introduzione

La realtà virtuale, che fino a pochi anni fa era una tecnologia ancora in fase sperimentale, sta diventato sempre più di tendenza, passando di fatto dall'essere pura fantascienza a essere un'effettiva realtà. Questa crescita può esser attribuita allo sviluppo tecnologico, alla riduzione di costi di hardware e software e all'integrazione della tecnologia in diversi settori, che spaziano dalla telefonia mobile, alla medicina, all'ingegneria, all'industria aerospaziale. Un'analisi di mercato[1] effettuata nel gennaio 2018, prevede che il mercato globale per le tecnologie di virtual reality (VR) passerà dai 3.7 miliardi di dollari investiti nel 2017, ai 39.4 miliardi nel 2022, con una crescita annua del 60.5%, come è mostrato nel grafico in figura 1.
Nel settore aerospaziale la realtà virtuale è ormai diventata di uso comune per svariati scopi, quali:

- addestramento degli astronauti
- interpretazione di dati scientifici
- pianificazione di percorsi e rotte
- simulazioni realistiche di tali spostamenti

Diversi sono i fattori che influenzano l'addestramento e l'utilizzo di strumentazioni altamente tecnologiche con le quali si ha a che fare nell'ambiente dell'industria aerospaziale, i principali sono:

1. Il **costo**: per ridurre i costi è necessario minimizzare il tempo e le risorse spese nell'addestramento, mantenendone però alta la qualità. Talvolta, il costo elevato degli strumenti non consente di poterli utilizzare per le esercitazioni.

2. La **complessità**: attività sempre più complesse necessitano di un supporto migliore e immediato, che va aldilà della consultazione di manuali.

3. La **qualità**: con compiti sempre più complicati diventa fondamentale un approccio pratico ai problemi e alle procedure da eseguire.

4. Il **rischio**: un approccio pratico seppur migliori la qualità dell'addestramento, può risultare pericoloso, oltre che di difficile realizzazione.

5. La **valutazione**: i metodi di valutazione tradizionali possono non esser sufficienti in quanto vi è la necessità di testare in modo pratico le abilità che si vogliono valutare. Una semplice verifica teorica dei risultati può non bastare quando si ha a che fare con dispositivi dal costo così elevato.

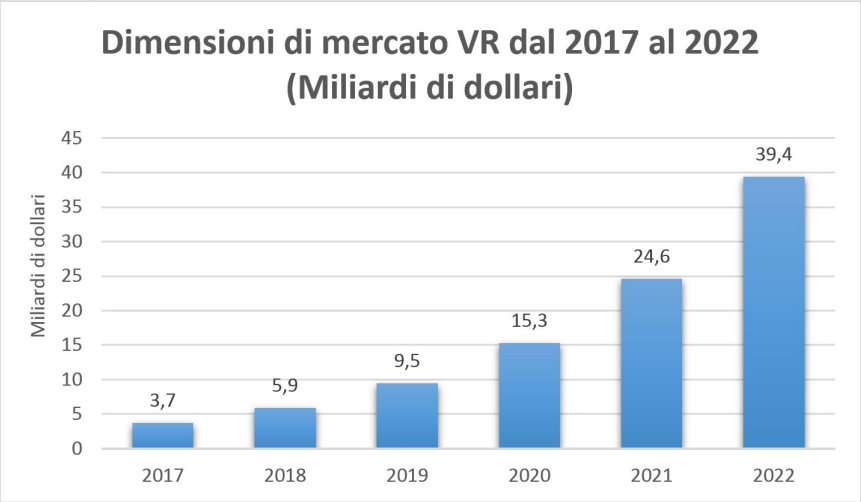


Figura 1: Previsione del mercato VR dal 2017 al 2022

L'utilizzo della realtà virtuale va incontro a tutte queste problematiche. La possibilità di creare un ambiente virtuale e di poter interagire in modo realistico con esso, permette di mantenere costi ridotti rispetto alla riproduzione di repliche a grandezza naturale della strumentazione necessaria. Con la creazione di svariati scenari e situazioni critiche, è possibile inoltre evitare danni ad apparecchiature e persone, permettendo di lavorare in condizioni di completa sicurezza e difficilmente riproducibili altrimenti. Un ruolo fondamentale per raggiungere questi obiettivi è il livello di realismo dell'esperienza virtuale e il senso d'immersione nell'ambiente simulato; è necessario riuscire a stimolare i sensi dell'utente in maniera adeguata, fornendogli riscontri sensoriali fedeli e in tempo reale.
Le applicazioni VR si rivelano estremamente utili anche per la visualizzazione di dati scientifici, facilitandone la comprensione a esperti del settore e non solo. Riscontri visivi accurati forniscono un'interpretazione chiara e immediata dei dati, rendendo più facili e intuitive operazioni come la pianificazione di azioni o spostamenti. Un discorso analogo può esser fatto per la verifica e la valutazione di risultati, rese più rapide e di semplice interpretazione tramite simulazioni precise che, se adeguatamente accurate, possono anche esser sfruttate come metodo di validazione.
In questo ambito si può collocare il progetto di tesi realizzato, il cui obiettivo principale è lo sviluppo di un'applicazione stereoscopica 3D che fornisca un supporto reale per le operazioni che riguardano l'esplorazione terrena, che sia il più possibile espandibile e adattabile a diversi dati e funzionalità. Nello specifico l'applicazione prevede l'utilizzo di dati riguardanti il pianeta Marte ma, con le opportune modifiche, può esser riadattata a qualsiasi tipo di terreno.
La tesi è composta di sei capitoli, di seguito ne viene fornita una breve descrizione. Nel primo capitolo vengono analizzati diversi progetti che riguardano la rappresentazione di dati planetari, l'interazione con applicazioni che permettono la visualizzazione di terreni, la pianificazione e simulazione di spostamenti. Vengono inoltre descritte alcune fra i più recenti software che hanno affrontato queste tematiche e viene fornita una prima descrizione del progetto realizzato.
Il secondo capitolo tratta l'argomento realtà virtuale, analizzando i concetti chiave e le diverse tipologie e dispositivi presenti sul mercato. È presente un approfondimento sulla stereoscopia 3D poiché è la tecnica che viene utilizzata nella realizzazione

dell'applicazione.
Nel terzo capitolo vengono delineati gli obiettivi e i requisiti del progetto di tesi, successivamente sono descritte le tecnologie hardware e software utilizzate.
Il quarto capitolo è focalizzato sullo sviluppo del progetto. In questa sezione viene fornita un'analisi dettagliata di come l'applicazione è stata realizzata, partendo dalla struttura e dal design, per arrivare alla descrizione degli elementi grafici e delle funzionalità disponibili.
Il quinto capitolo affronta, le problematiche e le criticità riscontrate durante lo sviluppo, viene inoltre fornita un'analisi statistica di test soggettivi effettuati sulle funzionalità, le prestazioni e l'utilità dell'applicazione.
Nel sesto capitolo vengono analizzati gli obiettivi raggiunti e tratte le conclusioni, fornendo degli spunti per possibili sviluppi futuri.

Capitolo 1

Stato dell'Arte

1.1 Rappresentazione di dati planetari

Fin dai primi anni '60 l'esplorazione marziana è stato un obiettivo importante per le agenzie aerospaziali di tutto il mondo. Numerose sono le sonde che ogni giorno inviano i dati acquisiti verso la Terra, con lo scopo di conoscere meglio il pianeta rosso; tali conoscenze potrebbero permettere agli scienziati di ottenere più informazioni riguardanti il passato, il presente e il futuro della Terra.
Nelle missioni per l'esplorazione marziana, un obiettivo che ci si è posti, è la possibilità di far raggiungere al rover un punto conosciuto del pianeta Marte, ma non visibile dal rover stesso. Uno studio[2] del 2007, che ha visto la partecipazione di ricercatori Nasa, ingegneri ambientali ed esperti di software, ha rivolto l'attenzione proprio su questo problema. La soluzione permette di facilitare e migliorare la navigazione a lungo raggio, generando mappe di terreni 3D. Più nello specifico sono generate delle mappe con risoluzione differente, utilizzando le diverse immagini che vengono acquisite da lander (il veicolo che permette al rover di atterrare sul pianeta), rover e satelliti: la mappa a massima risoluzione è ottenuta grazie alle immagini stereo acquisite dalle camere del rover, la mappa a risoluzione intermedia sfruttando immagini catturate dal lander durante la discesa o immagini stereo catturate da camere poste a distanza maggiore (wide-baseline stereo), la mappa a bassa risoluzione, ma che copre un'area maggiore, da immagini riprese durante la discesa ad alta quota o immagini orbitali. In figura 1.1 è possibile vedere la mappa altimetrica e un'ortoimmagine del sito di atterraggio di Mars Pathfinder, mentre in figura 1.2 il terreno 3D generato dai dati del rover.
Nonostante questa soluzione non sia pensata per l'integrazione con strumenti per VR, mette in risalto la necessità di facilitare e rendere più intuitiva l'interazione con i dispositivi per l'esplorazione terrena. La realtà virtuale è una tecnologia che si adatta perfettamente a queste esigenze, rendendo più semplice l'interpretazione e la visualizzazione dei dati scientifici, la simulazione, la pianificazione e il controllo delle azioni dei dispositivi.
Come riportato nell'articolo[3] presentato al *Extraterrestrial Mapping Symposium: Mapping of Mars*, il primo caso di utilizzo della realtà virtuale per analisi geologiche è avvenuto nella missione Mars Pathfinder, la prima ad aver portato un rover sul pianeta rosso. Il progetto consiste in un sistema interattivo di visualizzazione di terreni, in cui i modelli 3D della superficie marziana vengono riprodotti utilizzando le immagini stereo catturate dalla camera del lander, con un algoritmo chiamato

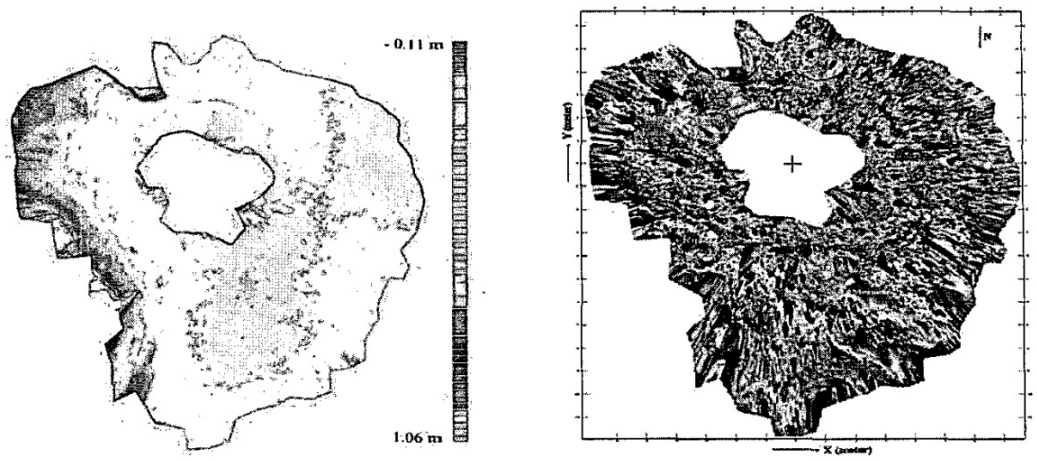


Figura 1.1: Mappa altimetrica e ortofoto del sito di atterraggio di Mars Pathfinder.

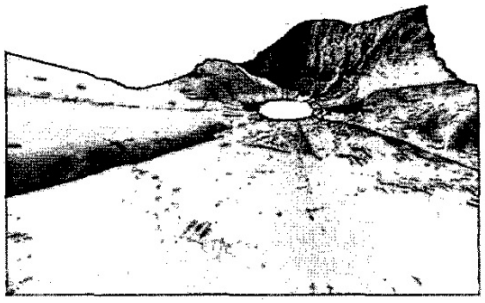


Figura 1.2: Vista 3D del terreno.

Ames Stereo Pipeline. Le immagini stereo consistono in una coppia di immagini che corrispondono a quella sinistra e destra che vedrebbero i nostri occhi. L'algoritmo sfrutta questo tipo di immagini per calcolare le posizioni in tre dimensioni dei vari punti e generare un modello 3D. È possibile visualizzare tali modelli con specifici occhiali stereoscopici o con un dispositivo HMD (head-mounted display) per godere di un'esperienza più immersiva. L'interazione è gestita tramite l'utilizzo di un normale mouse 2D. Vi è la possibilità di lavorare in diverse modalità che permettono di muoversi sul terreno a una certa altezza, oppure di ruotare o ingrandire il modello rispetto a un punto fissato.
Per il supporto e la visualizzazione di dati scientifici riguardanti la Terra, i ricercatori Suya You (University of Southern California, USA) e Charles. K. Thompson (Jet Propulsion Laboratory, USA), hanno sviluppato un sistema[4] che permette l'interazione con un modello di un terreno e la visualizzazione di informazioni scientifiche tramite l'utilizzo di mixed reality. Il prototipo dà la possibilità a più utenti, che siano in loco o in una sede distaccata, di lavorare insieme. Il suo funzionamento prevede l'elaborazione automatica di un video ripreso da un cellulare. Dopo questa fase è possibile:

1. rilevare gli elementi da visualizzare

2. stimare la posizione della camera

3. creare il contenuto virtuale

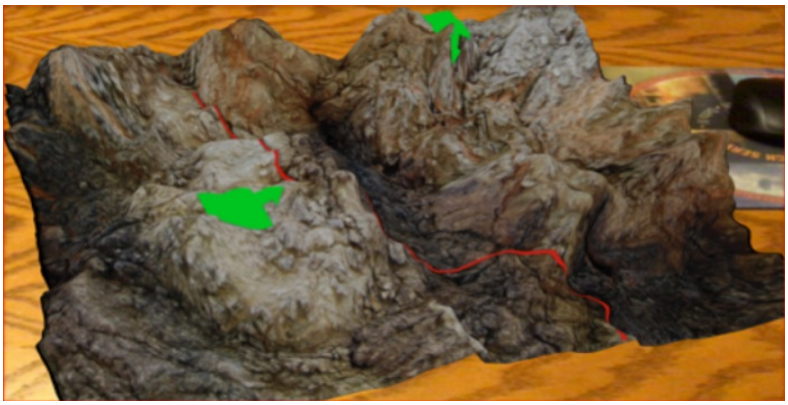


Figura 1.3: Rappresentazione 3D del modello di un terreno.

Quest'applicazione permette agli scienziati che si trovano realmente sul sito, di eseguire analisi e inviare informazioni ai collaboratori esterni, i quali possono accedere a un ambente virtuale immersivo e visualizzare un modello del terreno, in miniatura o a dimensioni reali. In figura 1.3 è possibile vederne un esempio. Un recente articolo[5], pubblicato dal *IEEE Transactions on Visualization and Computer Graphics journal*, tratta la rappresentazione di superfici planetarie, con l'obiettivo di rendere accessibili, anche ai non addetti ai lavori, informazioni e dati di difficile comprensione. Vi sono tre esempi che mostrano i risultati di questo lavoro, uno dei quali è proprio la rappresentazione di un terreno marziano. L'applicazione per realizzare un modello virtuale di una specifica area di Marte utilizza le immagini DTM (Digital Terrain Model) raccolte dal *Mars Reconnaissance Orbiter* (MRO)[6]. Il lavoro, realizzato con il software OpenSpace, permette di integrare diversi dispositivi per VR per rendere l'esperienza immersiva. In figura 1.4 si può osservare un esempio di riproduzione del suolo marziano.

Interessanti risultati derivano da uno studio[7] del 2014, effettuato da ricercatori delle università di Duke e Winchester, in cui vengono messi a confronto tre metodi per l'esplorazione e lo studio di terreni:

1. un CAVE a sei lati per la realtà virtuale immersiva
2. un'applicazione web 3D
3. un'applicazione standard 2D per desktop

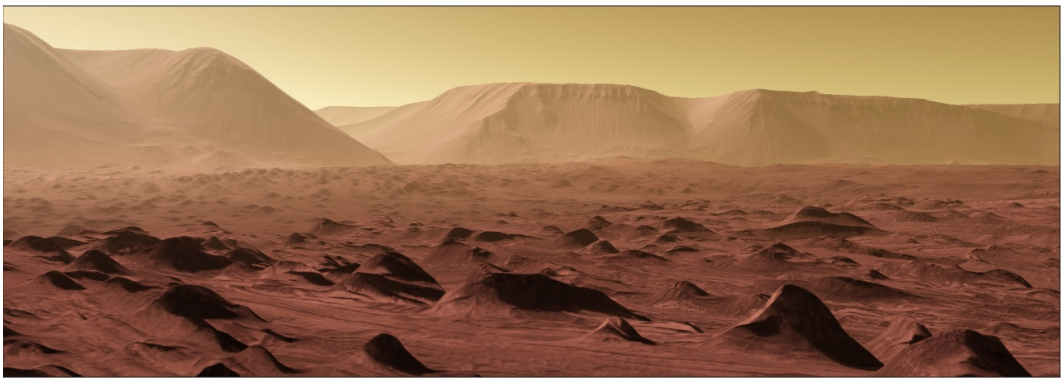


Figura 1.4: Riproduzione del terreno marziano ad alta risoluzione.

(a) GIS Desktop (b) 3D Web Application (c) CAVE-type immersive system

Figura 1.5: Diverse tipologie di applicazioni a confronto.

In figura 1.5 è possibile osservare i tre metodi a confronto. L'esperienza completamente immersiva è stata preferita dalla maggior parte dei partecipanti, soprattutto da non esperti del settore. Questo è dovuto alla maggior intuitività e alla più facile interazione che una soluzione per realtà virtuale garantisce. Nonostante la preferenza per soluzioni 3D e in modo specifico per l'esperienza fornita dal CAVE, non si ottiene un miglioramento sostanziale nell'esplorazione e nelle osservazioni archeologiche.
La tabella 1.1 riassume per ogni progetto sopracitato, il macro obiettivo, i dati che utilizza, le tecnologie e i metodi di interazioni supportati.

Articolo	**Macro obiettivo**	**Dati Utilizzati**	**Tecnologie supportate**	**Interazione**
Visual Terrain Mapping for Mars Exploration	Generazione mappe 3D di terreni	Immagini: stereo, wide-baseline stereo, di discesa e orbitali	VR non supportata	Mappe a diversa risoluzione
Visualizing Mars using virtual reality: A state of the art mapping tool used on Mars pathfinder	Visualizzazione e interazione con modelli 3D di terreni marziani	Immagini stereo	Occhiali stereoscopici e HMD	Mouse 2D
Mobile Collaborative Mixed Reality for Supporting Scientific Inquiry and Visualization of Earth Science Data	Interazione con un modello 3D di un terreno e visualizzazione di dati scientifici	Stream video di un cellulare	Mixed reality	Cellulare e tablet
Globe Browsing: Contextualized Spatio-Temporal Planetary Surface Visualization	Rappresentazione di superfici planetarie	Immagini orbitali e DTM	Immersive dome theater, interactive touch table, virtual reality headset	Schermi touch e VR controller
Comparison of interactive environments for the archaeological exploration of 3D landscape data	Confronto tra metodi per esplorazione terrena	—	CAVE a sei lati, 3D web application, 2D desktop application	—

Tabella 1.1: Confronto tra progetti di rappresentazione dati planetari

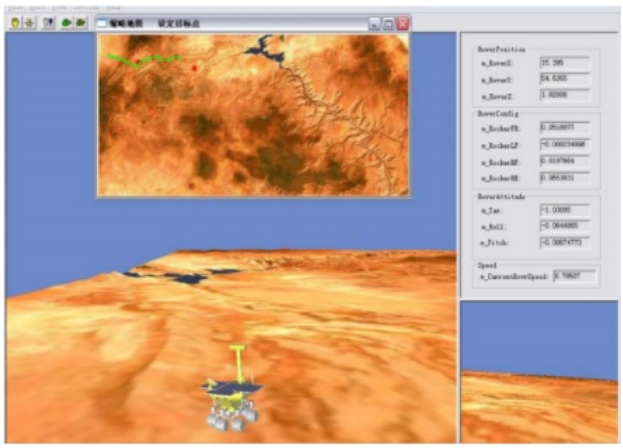


Figura 1.6: Interfaccia utente di VROS

1.2 Pianificazione e simulazione

Nel 2009, due ricercatori della University of Technology di Pechino e una professoressa della Surrey Space Centre University di Guildford, hanno sviluppato *VROS*[8] (Virtual Rover Operation Simulator), una piattaforma di simulazione che fornisce supporto alle missioni, permettendo di testare e validare diverse tipologie di rover sul suolo marziano. I modelli dei terreni marziani sono generati sfruttando i dati ottenuti da un DEM (Digital Elevation Model), mentre i rover sono dei modelli CAD a grandezza naturale dei prototipi reali. Il software è sviluppato come un'applicazione desktop e l'interazione è gestita tramite un mouse trakball. Sono gestite due diverse modalità di funzionamento:

- **Manuale**: l'utente può manovrare il rover con il mouse.
- **Automatica**: l'utente specifica un punto sul DEM e viene simulata la navigazione.

Una funzionalità che risulta importante per le missioni di esplorazione è la pianificazione dei percorsi che il rover deve seguire per raggiungere degli obiettivi. Un progetto[9] sviluppato nel 2017 da tre ricercatori spagnoli include questa funzione oltre a fornire una rappresentazione 3D in realtà virtuale della navigazione lungo il

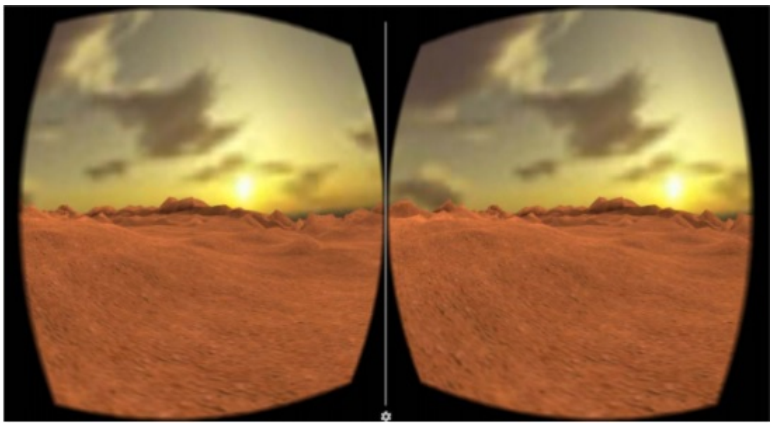


Figura 1.7: Esplorazione di un terreno attraverso un visore VR.

percorso stabilito. Il pianificatore di percorsi lavora con mappe di costo, calcolate in base alla morfologia del terreno, e con DTM ottenuti dal satellite MRO. Come è possibile vedere in figura 1.7, l'applicazione permette di visualizzare un modello del terreno in tre dimensioni, con l'utilizzo di visori VR, consentendo inoltre di poterlo esplorare in completa libertà.
Nella missione *Mars Exploration Rovers* (MER) è stata utilizzata un'applicazione come supporto all'esplorazione marziana che ha avuto grande successo[10]. Per realizzarla sono stati usati:

- *Viz*: un software per la visualizzazione 3D.

- *Ames Stero-pipeline*: l'algoritmo per la ricostruzione di modelli ad alta qualità dei terreni, già utilizzato per la missione Mars Pathfinder.

Viz è stato progettato per aiutare gli scienziati a comprendere una grande quantità di dati e fornire una maggior consapevolezza della situazione per le attività scientifiche, creando un ambiente di visualizzazione 3D interattivo, immersivo e facile da usare. Ames stereo-pipeline è invece in grado di generare mesh 3D partendo da immagini stereo, mentre immagini di discesa e orbitali vengono utilizzate per creare dei DEM. Viz permette la visualizzazione dell'ambiente in realtà virtuale utilizzando degli occhiali stereoscopici LCD per un'esperienza più immersiva.
Un'altro progetto[11] appartenente a questo settore che fornisce un'esperienza in realtà virtuale che permette agli utilizzatori di osservare l'intero ambiente da qualsiasi prospettiva si voglia, non solo da quella del rover, è stato sviluppato da Wang J. e Bennett K., due ricercatori della Washington University, a St. Louis.
La tabella 1.2, di seguito riportata, confronta obiettivi, dati utilizzati, tecnologie e metodi di interazione supportati dalle applicazioni di simulazione e navigazione terrena prese in considerazione.

Articolo	**Macro obiettivo**	**Dati Utilizzati**	**Tecnologie supportate**	**Interazione**
3D Virtual Reality Simulator for Planetary Rover Operation and Testing	Simulazione operazioni del rover su terreni 3D	Modelli CAD, DEM	Applicazione desktop	Spaceball mouse (6 DOF)
A Virtual Reality Mission Planner for Mars Rovers	Pianificatore di percorsi	DTM, immagini orbitali e di superficie	HMD	Dispositivi aptici
Photo-realistic Terrain Modeling and Visualization for Mars Exploration Rover Science Operations	Generazione ambiente immersivo e interattivo per supporto alle operazioni del rover	DTM, immagini stereo, orbitali e di discesa	applicazione desktop 3D, visalizzazione anaglifi	Mouse 2D, occhiali stereoscopici
A Virtual Reality Study on Santa Maria Crater on Mars	Creazione ambiente immersivo, esplorazione tramite rover	Immagini orbitali e DTM	Applicazione desktop	Mouse 2D

Tabella 1.2: Confronto tra progetti di simulazione e navigazione su terreni.

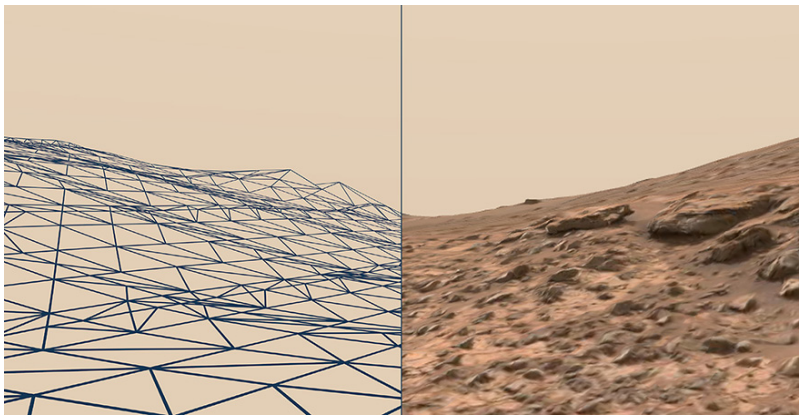


Figura 1.8: Modello 3D terreno marziano in Access Mars

1.3 Le applicazioni

Negli ultimi anni, Nasa e *Jet Propulsion Laboratory* (JPL) stanno sviluppando diversi progetti mirati a facilitare sempre più il lavoro di ricercatori e scienziati. Da una collaborazione tra JPL e Google, è nata un'applicazione unisce realtà virtuale e aumentata: *Access Mars*. Lo scopo è quello di riprodurre in modo fedele e realistico diversi siti marziani che sono stati cruciali per la missione *Mars Science Laboratory* della Nasa, nello specifico: il cratere Gale, sito di atterraggio del rover Curiosity, Marias Pass, luogo in cui Curiosity ha scattato una delle sue celebri foto, Pahrump Hills, dove il rover ha prelevato un campione di roccia e infine Murray Buttes, la Monument Valley del pianeta rosso. Come viene riportato nell'articolo presente sul sito ufficiale Nasa[12], grazie a quest'applicazione chiunque possieda una connessione a Internet può effettuare un vero e proprio tour del pianeta Marte, facendosi un'idea di come si presenta, scoprendo le conformazioni rocciose di particolare interesse per gli scienziati e camminando fianco a fianco a Curiosity. Access Mars è stato sviluppato per WebVR, uno standard che semplifica l'utilizzo di dispositivi per la realtà virtuale con i diversi software per la navigazione in Internet. In figura 1.8 è possibile vedere la ricostruzione del modello 3D del terreno con e senza la texture applicata.
Le funzionalità che offre Access Mars sono state ereditate dal software *OnSight*, anch'esso sviluppato da JPL con la collaborazione di Microsoft, che permette agli

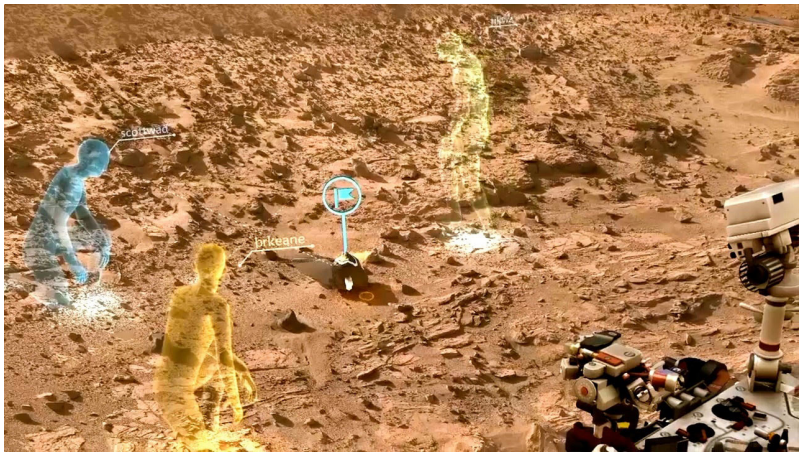


Figura 1.9: Collaborazione tra scienziati con OnSight

scienziati NASA di pianificare e seguire gli spostamenti del rover Curiosity e lavorare come se fossero su Marte utilizzando il dispositivo Microsoft HoloLens. Come si legge nell'articolo[13] pubblicato sul sito JPL, il software elabora i dati acquisiti da Curiosity per ricreare una riproduzione dell'ambiente marziano, dove gli scienziati, connessi da diverse parti del mondo, possono incontrarsi ed esaminare il sito del rover, pianificare attività e visualizzare i risultati ottenuti (figura 1.9); il tutto da una prospettiva in prima persona. L'innovazione che applicazioni come OnSight portano rispetto al passato, è costituita dalla percezione dello spazio e della profondità. La possibilità di esaminare in prima persona il suolo marziano e le attività di Curiosity, anche se solo in modo virtuale, aumentano notevolmente il senso d'immersione nell'ambiente riprodotto, aspetto che risultava estremamente carente in precedenza. Il software permette quindi di aiutare i ricercatori a conoscere in modo più approfondito l'ambiente che li circonda e sfruttare in modo ottimale le tecnologie a propria disposizione.

Grazie a un'altra collaborazione con Nasa è nata un'ulteriore applicazione in VR, chiamata Mars 2030. A scopo ludico e divulgativo, permette di impersonare un astronauta e di muoversi per il pianeta su una superficie di circa 40 km^2. Il terreno è modellato e mappato utilizzando i dati catturati dal satellite Mars Reconnaissance Orbiter. L'applicazione, sviluppata con il software Unreal Engine, permette di esplorare il pianeta ed effettuare svariate missioni, cercando di ricostruire un'esperienza che risulti essere più precisa e fedele possibile dal punto di vista scientifico.

La possibilità di esplorare gli ambienti in un modo più coinvolgente, sfruttando tutti i dati a disposizione, è vantaggiosa per professionisti e ricercatori, ma è anche al-

Figura 1.10: Foto scattata da Curiosity stesso nei pressi del cratere Gale

tamente consigliata e adatta per l'istruzione e la sensibilizzazione del pubblico. È per questo motivo che le società del settore aerospaziale stanno avviando progetti che sfruttano visori per la realtà virtuale a scopo educativo, che possano trovare spazio in musei, planetari o eventi dimostrativi e che permettano agli studenti di interagire in prima persona. L'ESA (European Space Agency), ha prodotto un'applicazione[14] che permette di visitare i diversi moduli che costituiscono la Stazione Spaziale Internazionale (ISS) e visualizzare informazioni e video.

1.4 Mars Navigation

Nell'applicazione realizzata, molta attenzione è stata posta sul realismo dell'ambiente virtuale, cercando di riprodurre effetti quali: la polvere, che viene sollevata dal rover quando è in movimento, e le tracce che lascia sul suolo.
Come per diversi progetti sopracitati sono state utilizzate immagini DTM per la realizzazione dei modelli dei terreni, mentre per il rover è stato utilizzato un modello di Curiosity, visibile in figura 1.10. Sono state sviluppate due modalità per l'interazione: una manuale, in cui è possibile manovrare il rover inviandogli dei comandi; un'altra automatica, che permette di impostare uno o più punti da raggiungere. L'interazione è gestita tramite un normale mouse 2D, con il quale si può osservare il sito marziano e Curiosity dall'esterno, oppure dalla prospettiva della camera montata sul rover.
Vengono visualizzate a video le informazioni relative alla posizione in cui ci si trova e all'altezza. È inoltre possibile impostare diverse immagini che, sovrapposte alla texture del terreno, permettono di visualizzare dati scientifici più accurati.
L'applicazione è predisposta per esser utilizzata con un proiettore a stereoscopia attiva, in un CAVE a schermo singolo.
Un importante obiettivo dell'applicazione è la versatilità che questa necessita di avere, la facile adattabilità a dati di diverso tipo e modalità di funzionamento.

Capitolo 2

La Realtà Virtuale

2.1 Cosa si intende per realtà virtuale

Trovare una definizione esatta di realtà virtuale è più difficile di quanto si possa pensare. La parola "virtuale" può esser definita come qualcosa che dia l'impressione di esistere, ma non lo sia realmente. La definizione di "realtà" può esser invece espressa come una qualità o una condizione di qualcosa che esiste effettivamente e concretamente.
Di fatto quindi, la realtà virtuale, si pone come obiettivo la creazione di un mondo completamente sintetico e simulato, che riesca però a far dimenticare all'utente di trovarsi in un ambiente artificiale, ma anzi, che gli faccia provare un senso di immersione totale in esso. Per ottenere tutto ciò è necessario coinvolgere il soggetto fornendogli diversi stimoli sensoriali.
I ricercatori W. Sherman e A. B. Craig, nel libro intitolato *Understanding Virtual Reality: Interface, Application, and Design*[15], definiscono quattro elementi chiave per la realtà virtuale:

- Mondo virtuale
- Senso d'immersione
- Stimoli sensoriali
- Interattività

Il mondo virtuale

Il mondo virtuale è un ambiente immaginario tridimensionale che spesso, ma non sempre, viene realizzato e visualizzato tramite un mezzo (display, render ad esempio). Include un insieme di oggetti e una serie di regole e relazioni che li governano. In un mondo virtuale, che venga riconosciuto come realistico dall'utente, è necessario che elementi quali la prospettiva, le risposte visive e i movimenti siano riprodotti in modo del tutto fedele alla realtà.

Il senso d'immersione

Può esser definito come la percezione di esser fisicamente presente in un mondo non fisico. Lo stato di immersione totale sussiste quando vengono stimolati abbastanza

sensi in modo da ricreare la sensazione di esser realmente presenti in un luogo che invece reale non è.
Nella realtà virtuale il senso di presenza è favorito da due tipologie diverse di immersione:

- **Immersione fisica o sensoriale**: con cui si intende la stimolazione, in maniera sintetica, dei sensi dell'essere umano.
- **Immersione mentale**: con cui si intende lo stato mentale che permette di sentirsi profondamente coinvolti nell'ambiente simulato, superando il senso di incredulità.

Gli stimoli sensoriali

Maggiore è il numero di sensi che vengono coinvolti nell'esperienza virtuale, migliore sarà il senso d'immersione. La vista è il senso dominante nell'essere umano, proprio per questo motivo il fotorealismo assume un ruolo fondamentale nelle soluzioni VR. Stimoli sensoriali per gli altri sensi sono riproducibili per mezzo di appositi dispositivi hardware. Nonostante spesso alcuni sensi vengano parzialmente trascurati, olfatto e tatto ne sono un esempio, è possibile sfruttare diversi fattori per ingannare il cervello e rendere l'esperienza virtuale completa ed efficacie. Un esempio è costituito dall'utilizzo di audio spaziale 3D che, sfruttando la capacità del cervello umano di capire distanza e direzione di un suono, permette di ricreare un'esperienza altamente immersiva.

L'interattività

L'interazione in applicazioni per la realtà virtuale è un elemento fondamentale per trasmettere agli utenti un naturale senso di coinvolgimento. La difficoltà nella gestione dei meccanismi d'interazione risiede nel fatto che il cervello umano sia molto sensibile a eventi e comportamenti che differiscono da quelli reali a cui è abituato, per questo anche piccole imprecisioni nelle risposte che forniscono l'ambiente e gli oggetti in esso presenti, portano a una diminuzione del senso d'immersione.

2.2 Le origini e le tecnologie

Del concetto di realtà virtuale si iniziò a parlare già nei primi anni cinquanta quando, Morton Heilig, scrisse del *Experience Theater*. Il cosiddetto "cinema d'esperienza" era in grado di coinvolgere tutti i sensi in maniera efficacie, immergendo lo spettatore nelle scene che osservava sullo schermo. Partendo da quest'idea, nel 1962, ideò un prototipo che chiamò Sensorama (figura 2.1). Lo strumento funzionava in modo meccanico e permetteva di vedere cinque cortometraggi che coinvolgevano diversi sensi.
Nel 1968, venne creato quello che viene considerato oggi come il primo visore per applicazioni immersive. Progettato dallo scienziato Ivan Sutherland con un suo studente, Robert Sproull, il dispositivo, molto grezzo dal punto di vista dell'interfaccia grafica e del realismo, era così pesante che, per esser indossato, necessitava di esser

appeso al soffitto. Proprio per il suo aspetto venne battezzato La Spada di Damocle. La nascita del termine VR, Virtual reality, risale agli anni '80 e si diffonde grazie a uno dei pionieri in questo campo, Jaron Lanier, il quale, dopo aver fondato l'azienda VPL Research nel 1895, sviluppò diversi dispositivi VR, quali: Data Glove, Eye Phone e Audio Sphere.
Negli anni '90 venne sviluppata da alcuni ricercatori dell'Electronic Visualization Laboratory, la prima stanza adibita per la realtà virtuale. Il Cave Automatic Virtual Enviroment, più noto con l'acronimo CAVE, consiste in una stanza cubica sulle cui pareti vengono proiettate le immagini che permettono all'utente di sentirsi parte di un mondo virtuale, aumentando notevolmente la libertà e il senso di immersione dell'utilizzatore. Le proiezioni possono avvenire su tutte le pareti della stanza oppure solamente su alcune.
Con l'arrivo del nuovo millennio si ha assistito a un moltiplicarsi delle applicazioni destinate alla realtà virtuale e di conseguenza anche delle tecnologie dedicate a questo scopo. HTC, Valve Corporation, Playstation VR e Oculus VR sono solo alcune delle società che hanno sviluppato visori sempre più all'avanguardia e dalle performance elevate, fino ad arrivare a includere in uno stesso dispositivo: due schermi differenti per occhio destro e sinistro ad alta definizione, la riproduzione di audio spaziale 3D e il tracciamento della posizione e dei movimenti della testa.

Figura 2.1: Sensorama, la macchina per il cinema d'esperienza

2.3 Tipi di realtà virtuale

Esistono diverse tipologie di realtà virtuale, che si differenziano a seconda del livello d'immersione che forniscono che, come visto in precedenza, è strettamente collegato a quanti sensi vengono stimolati e in che modo. Si possono definire delle macro categorie con cui identificare le applicazioni VR:

- **Non immersive** (figura 2.2a): solo una parte dei sensi vengono stimolati, mantenendo viva la consapevolezza del mondo esterno da parte dell'utente. Sono utilizzati schermi ad alta risoluzione, con strumenti per l'interazione convenzionali, come tastiere, mouse, trackball, o meno convenzionali, come SpaceBall o DataGlove.

- **Semi immersive** (figura 2.2b): forniscono un'esperienza migliore delle precedenti, in cui l'utente è immerso parzialmente nel mondo virtuale. Vengono utilizzate tecnologie più sofisticate: sistemi con prestazioni elevate, sistemi di proiezione o schermi multipli. Spesso, questo tipo di applicazioni vengono utilizzate per l'addestramento, ne sono un esempio i simulatori di volo.

- **Completamente immersive** (figura 2.2c): forniscono l'esperienza più diretta e coinvolgente. In simulazioni completamente immersive si utilizzano solitamente HMD e dispositivi che rilevano il movimento e permettono l'interazione e la stimolazione dei sensi dell'utente. La possibilità di avere alta risoluzione, ampio campo visivo, frame rate elevato permette di creare simulazioni molto realistiche.

(a) Non immersiva

(b) Semi immersiva

(c) Completamente immersiva

Figura 2.2: Tipi di realtà virtuale

2.4 Concetti e componenti base

Per fare in modo che l'utente si senta parte del mondo artificiale e lo consideri come reale, è necessario non solo che questo sembri reale all'aspetto, ma anche che venga percepito come tale. Di fondamentale importanza diventano quindi i dispositivi che stimolano i sensi dell'utente. I componenti base per un sistema di realtà virtuale sono:

- **Un computer, console o telefono cellulare**: che permetta di alimentare e gestire l'ambiente virtuale.
- **Un sistema di visualizzazione**: sistemi che permettano di vedere il mondo artificiale in 3D. Questi possono essere: HMD con schermi separati per occhio destro e sinistro, sistemi CAVE, schermi o proiettori stereoscopici.
- **Dispositivi di input**: che ricreino il senso di immersione permettendo l'interazione e fornendo stimoli sensoriali.

Diversi sono i concetti di cui bisogna tener conto quando si ha a che fare con la realtà virtuale. Di seguito verranno analizzati i più importanti, che possono avere un forte impatto sulla qualità delle applicazioni VR.

Campo visivo

Identificato con l'acronimo FOV (Field of View o Field of Vision), è uno degli aspetti più importanti per riuscire a fornire una percezione realistica del mondo virtuale. Consiste nell'estensione dell'ambiente che è possibile osservare, viene misurato in gradi e, nell'essere umano, se ne possono distinguere due tipi differenti. Uno è il campo visivo monoculare, che di norma varia tra i 170° e i 175°, l'altro è il campo visivo binoculare, che consiste nella sovrapposizione dei due campi monoculari, destro e sinistro. L'area visibile è data dalla combinazione di queste due aree e misura circa 200° - 220°, ma, quella in cui si ha la percezione della tridimensionalità, è solo l'aera in cui i campi monoculari si sovrappongono, di circa 114°.
I comuni visori hanno un FOV di 100° - 110°, sufficiente per la maggior parte dei contenuti virtuali; alcuni dispositivi raggiungono anche ampiezze superiori a 200°.

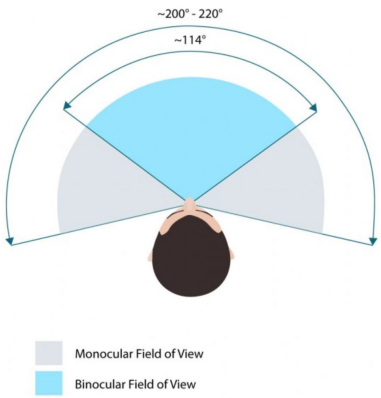


Figura 2.3: Campo visivo dell'essere umano

Frame rate

Corrisponde alla frequenza con la quale vengono riprodotti i fotogrammi. Il numero di immagini riprodotte in un secondo viene identificato dall'acronimo FPS (Frames Per Second). Per una normale riproduzione video si ha un valore di circa 30 FPS, per applicazioni VR, invece, è necessario che il frame rate sia elevato per ottenere un risultato fluido, occorre avere quindi un valore di almeno 60 FPS. Per evitare il manifestarsi di effetti indesiderati nell'utente, come nausea o mal di testa, l'obiettivo per sviluppatori di applicazioni VR è di mantenere un valore di circa 90 FPS.

Latenza

Per latenza si intende un ritardo che intercorre tra un segnale in input e la rispettiva risposta in output. La latenza del sistema è data dalla combinazione della latenza di sensori di input, del sistema di tracking, della pipeline di rendering, del display e del sistema di sincronizzazione.In generale, una latenza inferiore ai 100 ms non viene percepita direttamente, ma una scena in movimento da l'impressione di esser instabile. Perché un'esperienza in realtà virtuale sia realistica e fluida, è quindi necessario che la latenza sia inferiore ai 20 ms. Un valore superiore causerebbe nell'utente nausea e altri effetti indesiderati come giramenti di testa e disorientamento.

Audio

La componente sonora, rispetto a quella visiva, potrebbe non esser considerata come fondamentale, ma anch'essa è importante per raggiungere un senso d'immersione completo. L'audio utilizzato in applicazioni VR è detto audio 3D o binaurale. Il cervello umano è in grado di identificare la posizione di emissione di un suono, capacità che, se sfruttata, può migliorare nettamente l'esperienza VR. Sono tre i fattori che danno informazioni sulla direzione di emissione di un'onda sonora:

1. Il ritardo di ricezione del suono tra le due orecchie.
2. La differenza di volume e intensità del segnale per le attenuazioni della testa.
3. Filtraggio del suono legato all'interazione acustica con tronco, testa e orecchio esterno. Il filtraggio delle diverse frequenze dipende dalla direzione di incidenza del suono.

Tracking

I dispositivi di tracking hanno il compito fondamentale di capire l'orientamento e la posizione dell'utente nel mondo virtuale. Esistono diversi dispositivi di tracking che utilizzano tecnologie differenti, i principali sono:

- **Head tracking**: tramite un sistema detto a sei gradi di libertà, 6DOF (Degrees of Freedom), è possibile tracciare i movimenti della testa in termini dei tre assi (x, y, z). Si possono quindi distinguere spostamenti della testa avanti/indietro, su/giù, sinistra/destra, combinati con rotazioni lungo i tre assi perpendicolari, anche detti imbardata (*yaw*), beccheggio (*pitch*), rollio (*roll*). Per tracciare questi movimenti sono utilizzati sensori di diverso tipo, tra i quali giroscopi, accelerometri e magnetometri.

- **Motion tracking**: permette di utilizzare come mezzo d'interazione il proprio corpo, l'esempio più comune sono le mani. Per riuscire a gestirne il movimento si utilizzano dispositivi di vario genere (controller, joystick o appositi guanti) tramite i quali è possibile riconoscere determinati gesti o movimenti.

- **Eye tracking**: questo sistema è ancora in evoluzione, la possibilità di riconoscere il movimento degli occhi e di poter interagire con essi è probabilmente l'ultimo passo da compiere per rendere un'esperienza per VR completamente immersiva. Un grande vantaggio che l'eye tracking può portare è la possibilità di riprodurre l'effetto di sfocatura causato dalla percezione della profondità: nella realtà infatti, se guardiamo un oggetto in primo piano, l'ambiente e gli altri oggetti in lontananza risulteranno non a fuoco, e viceversa. Con tutti gli oggetti presenti nel mondo virtuale a fuoco, è più facile incorrere in problemi di malessere generale come nausea e giramenti di testa poiché l'occhio e il cervello umano si accorgono di questa differenza.

2.5 Stereoscopia 3D

Il sistema visivo umano combina le due visioni prospettiche bidimensionali che vedono occhio destro e sinistro, ottenendo così un'immagine tridimensionale, con associate le informazioni su profondità e distanze. La percezione della profondità nelle immagini, può esser rinforzata da una serie di indizi, ad esempio:

- **Occlusioni**: un oggetto più vicino nasconde un oggetto più lontano che si trova sulla stessa linea di vista.

- **Dimensioni relative**: oggetti lontani risultano più piccoli rispetto a oggetti vicini.

- **Ombre**: forniscono informazioni sulla posizione di oggetti rispetto a una sorgente di luce.

- **Parallasse di movimento**: oggetti più vicini sembrano muoversi più velocemente di quelli distanti.

La stereoscopia è una tecnica che permette di trasmettere un'illusione di tridimensionalità, analoga a quella che il nostro cervello genera grazie alla visione binoculare o, per l'appunto, stereoscopica. In questo modo si possono realizzare e vedere immagini, filmati e più in generale qualsiasi contenuto riproducibile su uno schermo, in tre dimensioni.
L'idea di base per è quella di ricreare due diverse immagini, corrispondenti a ciò che vedrebbe l'osservatore con l'occhio destro e sinistro.
I sistemi più diffusi sono basati sull'utilizzo di specifici occhiali e possono essere:

- **Head mounted**: due schermi diversi posizionati davanti agli occhi, è questo il caso dei più famosi HMD.

- **Multiplexed**: visione stereoscopica riprodotta su un'unica superficie di visualizzazione.

Mentre per i sistemi head mounted vengono visualizzate due immagini destra e sinistra ognuna sullo schermo posizionato davanti al rispettivo occhio, per i sistemi multiplexed, in cui lo schermo è unico, è necessario utilizzare apposite tecniche.
Di seguito verranno descritte le principali tecniche di stereoscopia, soffermandosi in particolare sulla proiezione 3D essendo la strumentazione utilizzata per il progetto di tesi.

2.5.1 Stereoscopia passiva

Viene definita stereoscopia *passiva* perché gli permette la visualizzazione di immagini tridimensionali con occhiali che non sono dotati di alcuna tecnologia elettronica. La separazione delle immagini può avvenire tramite filtri polarizzatori o cromatici. Con quest'ultima tecnica, l'immagine, detta anaglifo, è data dalla sovrapposizione di immagine destra e sinistra, alle quali vengono applicati filtri di colori complementari. È possibile visualizzare il contenuto in 3D di un anaglifo utilizzando degli appositi occhiali filtrati con i medesimi colori. Il costo di questa soluzione è molto basso, ma anche la qualità non è elevata.
Per quanto riguarda la proiezione 3D passiva realizzata tramite polarizzazione della luce, si utilizza un doppio proiettore o un solo proiettore attivo con l'aggiunta di un polarizzatore elettronico. Nel primo caso è necessario utilizzare dei filtri ottici polarizzati che abbiano una polarizzazione opposta, tale da consentire la separazione dei due canali di proiezione sovrapposti. Nel secondo caso, il proiettore è sincronizzato con il filtro del polarizzatore elettronico che alterna polarizzazioni opposte a seconda del fotogramma che viene proiettato.
La luce può esser polarizzata in modi differenti (figura 2.4):

- **Lineare**: i filtri sono orientati a 90° l'uno rispetto all'altro. Ha il grande svantaggio che, inclinando la testa, si perde la percezione della tridimensionalità, perché la luce non viene più filtrata correttamente.
- **Circolare**: luce polarizzata a spirale di senso opposto per immagine destra e sinistra. Ha il grande vantaggio che rotazioni o inclinazioni della testa non fanno perdere l'effetto stereoscopico.

2.5.2 Stereoscopia attiva

Il termine *attiva* deriva dal fatto che gli occhiali per la visualizzazione sono dotati di una tecnologia elettronica che permette loro di adattarsi alle immagini da visualizzare.
La stereoscopica attiva viene realizzata tramite monitor o proiettori. Per quanto riguarda la proiezione 3D attiva viene utilizzato un solo proiettore che alterna nel tempo la proiezione di immagine destra e sinistra.
Gli occhiali sono costituiti da due otturatori elettronici a cristalli liquidi che si sincronizzano con il proiettore, permettendo all'osservatore di vedere correttamente immagine destra e sinistra con i rispettivi occhi. In sostanza, gli otturatori LCD possono esser resi alternativamente opachi o trasparenti, in modo tale che ciascun occhio riesca a vedere solo l'immagine a lui designata.
È di fondamentale importanza che gli occhiali siano sincronizzati con il proiettore, solitamente la sincronizzazione è realizzata tramite un emettitore a infrarossi o in

radio frequenza.
Le immagini sono quindi elaborate in modo separato per occhio destro e sinistro, per ottenere un risultato fluido occorre quindi avere un frame rate elevato, che generalmente è di circa 120Hz (60Hz per occhio).

2.5.3 Proiezione attiva e passiva a confronto

La proiezione 3D passiva che garantisce una buona qualità finale è quella che utilizza filtri polarizzatori. Come già descritto in precedenza, la polarizzazione lineare ha il grande inconveniente che, con l'inclinazione della testa, si perde l'effetto stereoscopico. La polarizzazione circolare risolve questo problema, ma non è comunque priva di difetti; in particolare, è particolarmente soggetta a quello che viene definito effetto *ghosting*, che consiste nel vedere con un occhio parte del fotogramma destinato all'altro occhio. Per utilizzare questa tecnica, è necessario, inoltre, avere una potenza luminosa molto elevata, che se da un lato è un vantaggio perché migliora la chiarezza delle immagini, dall'altro, è anche un fattore che influisce sul costo delle apparecchiature. A questo, si deve anche aggiungere la necessità di utilizzare uno schermo per proiezione non depolarizzante, che sia in grado di restituire i raggi luminosi con la medesima polarizzazione. Il costo di schermi di questo tipo è molto elevato e la loro qualità influisce molto sul risultato della proiezione stereoscopica. Il grande vantaggio riguarda invece gli occhiali che vengono utilizzati che, non essendo dotati di alcuna tecnologia elettronica, hanno un costo molto basso e non necessitano di particolare manutenzione. A ciò va aggiunto che l'assenza di *flickering*, lo sfarfallio che si ha negli occhiali 3D attivi dovuto all'alternarsi di apertura e chiusura degli otturatori elettronici, permette una visione prolungata nel tempo senza incorrere in malessere o affaticamento degli occhi.
Per quanto riguarda la proiezione 3D attiva, un vantaggio è determinato dalla versatilità; non è infatti richiesto l'utilizzo di schermi specifici per la proiezione, fattore che porta a una riduzione dei costi oltre che a una maggior flessibilità di utilizzo. Questa tecnica però porta con sé alcuni difetti: gli occhiali infatti, oltre ad avere un costo nettamente maggiore di quelli passivi, hanno il problema del flickering già

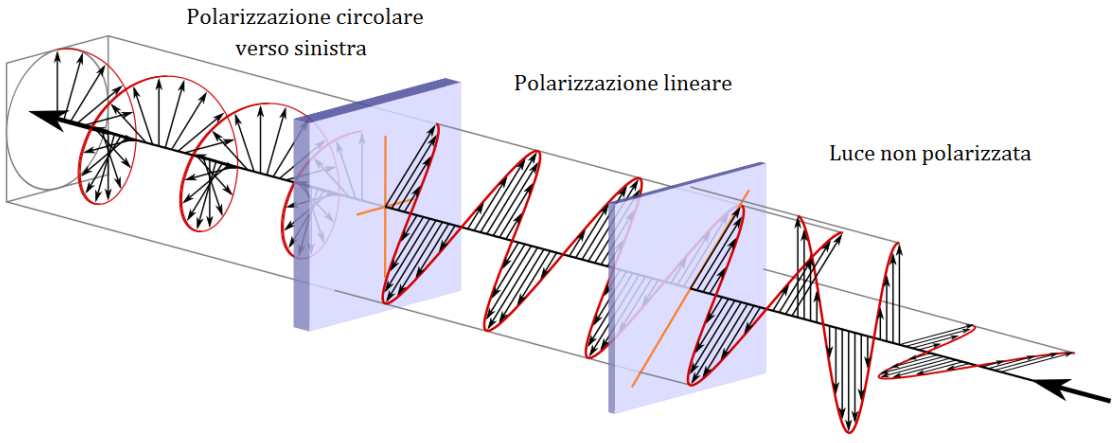


Figura 2.4: Tipi di polarizzazione della luce

citato in precedenza, sono più ingombranti e pesanti e necessitano di un sistema di alimentazione e di una manutenzione più accurata.

2.6 Problemi della realtà virtuale

Non si può pensare che la realtà virtuale abbia solo aspetti positivi, nel suo utilizzo infatti, si va inevitabilmente incontro a problemi e fattori negativi, che non le permettono ancora di diventare una tecnologia di massa per scopi ludici o in ambito tecnico/scientifico, analizziamo i principali:

- **Costo**: nonostante sia un aspetto in miglioramento grazie alla diffusione di dispositivi low cost, il costo dei componenti necessari a usufruire di un'applicazione VR non è alla portata di tutti.
- **Malessere**: la realtà virtuale può portare all'utente un senso di malessere, affaticamento visivo, mal di testa e in alcuni casi anche nausea. Questo è dovuto al fatto che i nostri occhi nel mondo reale convergono in modo naturale e mettono a fuoco un punto nello spazio, mentre nell'ambiente virtuale gli occhi cercano di metter a fuoco qualcosa che sembra lontano ma che in realtà è distante pochi centimetri.
- **Sicurezza**: oltre all'invasività di alcune interfacce, uno dei pericoli che legati alla realtà virtuale è la perdita della percezione di ciò che sta succedendo intorno. Questa è una problematica che si riscontra utilizzando i visori virtuali, che se da un lato migliorano il senso di immersione, dall'altro escludono completamente l'utente dal mondo reale.
- **Dispositivi di input**: in alcuni casi i dispositivi utilizzati necessitano di esser collegati tramite cavo a computer o console, che possono esser d'intralcio oltre a limitare i possibili movimenti.

L'interazione, il realismo, le sensazioni di malessere e la sicurezza sono aspetti che sicuramente necessitano di esser migliorati per riuscire a sfruttare a pieno una tecnologia che, seppur ancora in via di sviluppo e in continua evoluzione, può portare a grandi benefici nei più svariati ambiti.

Capitolo 3

Requisiti di Progetto

3.1 Obiettivi

L'obiettivo del progetto di tesi è la realizzazione di un'applicazione immersiva in realtà virtuale per la navigazione terrena che sia di supporto per operazioni come la visualizzazione di dati planetari, l'esplorazione di un terreno e la pianificazione di percorsi. Il principale software utilizzato è stato *Unity3D*, si veda il paragrafo 3.2 per una descrizione delle tecnologie utilizzate, software e hardware.
Di seguito vengono descritti i principali requisiti del progetto, indicandone motivazioni e finalità.

- **Ricostruzione realistica**: l'immediatezza visiva è uno degli obiettivi prefissati che risulta più importante. Operazioni come lo studio della conformazione di un terreno o la valutazione di particolari aree, risultano nettamente semplificate se supportate da ricostruzioni 3D in grado di fornire riscontri visivi veloci, chiari e intuitivi, comprensibili da chiunque, non solo da esperti del settore. Mantenere un livello di realismo elevato non è solo fondamentale per ricreare un'esperienza in realtà virtuale soddisfacente, ma lo è anche al fine di fornire un supporto che sia veramente efficacie. Per questo motivo l'applicazione deve poter utilizzare ricostruzioni 3D di terreni marziani reali, ottenute da appositi dati.

- **Visualizzazione dati scientifici**: l'intuitività che può dare una ricostruzione 3D è sufficiente a fornire un'idea di come si presenta una determinata area, ma non lo è se se ne vogliono effettuare delle valutazioni più tecniche; è quindi importante fornire un supporto scientifico e più dettagliato che può esser dato dalla visualizzazione di dati orbitali e derivati (immagini orbitali, mappe delle pendenze, altimetriche o infrarossi) o da informazioni legate al terreno o al veicolo utilizzato.

- **Simulazione movimenti del rover**: un altro requisito del progetto consiste nel ricreare in modo realistico i movimenti di un rover che possa spostarsi sul suolo marziano e permetta di ottenere una simulazione fisica affidabile. Un dato che è di fondamentale importanza visualizzare e mantenere sempre aggiornato, è la posizione del rover sul terreno, in termini di latitudine, longitudine e altezza a cui si trova.

- **Pianificazione**: la pianificazione di percorsi, la definizione di punti di interesse e obiettivi da raggiungere è una funzionalità che necessita di esser realizzata in un'applicazione di questo tipo. L'utente deve poter specificare uno o più obiettivi da raggiungere e andare a definire dei punti di interesse sul terreno. È inoltre utile poter visualizzare gli spostamenti effettuati dal rover, in modo da controllare il percorso fatto; a volte è necessario che segua una certa traiettoria per effettuare analisi del sottosuolo, la possibilità di visualizzare a video una traccia degli spostamenti che il rover ha compiuto permette di verificarne la correttezza.

- **Predisposizione per stereoscopia 3D attiva**: la realizzazione di un'applicazione in realtà virtuale necessita dell'utilizzo di un dispositivo adatto. Il progetto di tesi prevede l'utilizzo di un cave con schermo singolo e deve supportare la stereoscopia 3D attiva, per rendere l'esperienza dell'utente immersiva.

- **Adattabilità**: viste le numerose funzionalità che è possibile realizzare è necessario che l'applicazione sia resa il più facilmente espandibile e adattabile. Questo per facilitare l'integrazione di differenti tipologie di file o dati e di rendere più semplice la modifica di alcune funzionalità in base al loro utilizzo.

3.2 Tecnologie utilizzate

3.2.1 Software

Blender

Blender[16] è un software libero e multipiattaforma di modellazione, animazione e rendering di immagini tridimensionali. Dispone inoltre di funzionalità per mappature UV, simulazioni di fluidi, di particelle, altre simulazioni non lineari e per la creazione di applicazioni 3D.
In questo progetto è stato utilizzato per la creazione dei modelli 3D dei terreni marziani. A tale scopo è stato utilizzato il plugin *HiRISE DTM from PDS IMG*[17] di HiRISE che permette di creare un modello 3D a partire da un DTM generato dalle immagini provenienti dalla sonda della NASA Mars Reconnaissance Orbiter. Il plugin legge le informazioni in formato PDS (Planetary Data System) dal file, decodifica il raster delle altezze e genera il modello 3D. È possibile impostare diverse risoluzioni e diverse scale di ingrandimento. I modelli sono poi stati esportati in formato FBX, così da permetterne la gestione con il software Unity.
Approfondimenti sul funzionamento e su come sono stati gestiti i modelli saranno trattati nel capitolo successivo, al paragrafo 4.2.

Unity

Unity[18] è un motore grafico tramite il quale è possibile creare applicazioni 2D e 3D per un gran numero di piattaforme. È dotato di un motore fisico integrato, *PhysX*[19], che permette di gestire simulazioni realistiche, modificando le forze che agiscono su oggetti e sull'ambiente.
Unity fornisce anche supporto per sviluppo di applicazioni in realtà virtuale e anche,

più nello specifico, per la stereoscopia 3D rivolta a dispositivi non head-mounted, come nel caso del progetto di tesi. Risulta quindi possibile gestire i parametri che regolano l'effetto stereoscopico, come la distanza interpupillare (IPD) e il punto di convergenza.
Con questo software è stata realizzata sia la parte grafica dell'applicazione che la parte funzionale, tramite script in linguaggio C#.
I componenti utilizzati e le soluzioni adottate saranno descritte nel capitolo 4, in cui verrà trattata la parte relativa allo sviluppo vero e proprio dell'applicazione.

GDAL

GDAL[20], *Geospatial Data Abstraction Library*, è una libreria Open Source che permette l'elaborazione di file geospaziali, supportando tutti i formati disponibili.
Della libreria, nativa C++, è stata utilizzata una versione C#, andando a realizzare alcune funzioni non disponibili per questa versione.
Con questa libreria è stato possibile effettuare le trasformazioni nei diversi sistemi di riferimento per ottenere le informazioni necessarie a localizzare un punto sul terreno digitale e fornirne tutti i dati richiesti. Anche in questo caso, i dettagli tecnici saranno trattati nel capitolo 4.

3.2.2 Hardware

Per creare l'effetto stereoscopico è stato utilizzato un proiettore *Barco RLM-W14* [21] a stereoscopia attiva che utilizza la tecnologia DLP (*Digital Light Processing*). La frequenza di visualizzazione dei frame è di 120Hz e viene utilizzata una risoluzione di 1920x1200 pixel.
Gli gli occhiali stereoscopici attivi sono dotati di schermi LCD e la sincronizzazione con il proiettore avviene tramite infrarossi.
L'applicazione è pensata per esser eseguita su una macchina *Dell Precision Tower 7810* che monta due processori *Xeon E5-2650 v3*, ognuno con 10 core e frequenza base 2.30 GHz, scheda grafica *NVIDIA Quadro K5200*.

Capitolo 4

Sviluppo del Progetto

4.1 La struttura

Il progetto è costituito da tre scene: una iniziale, una di loading e una scena per l'esplorazione. Queste sono gestite da un primo script chiamato *SceneManagerScript*, che ha il compito di eseguire le operazioni di inizializzazione, e da un secondo script, *SceneLoader*, che controlla il passaggio da una scena all'altra. Il diagramma in figura 4.1 delinea in modo semplificato il flusso d'esecuzione dell'applicazione.

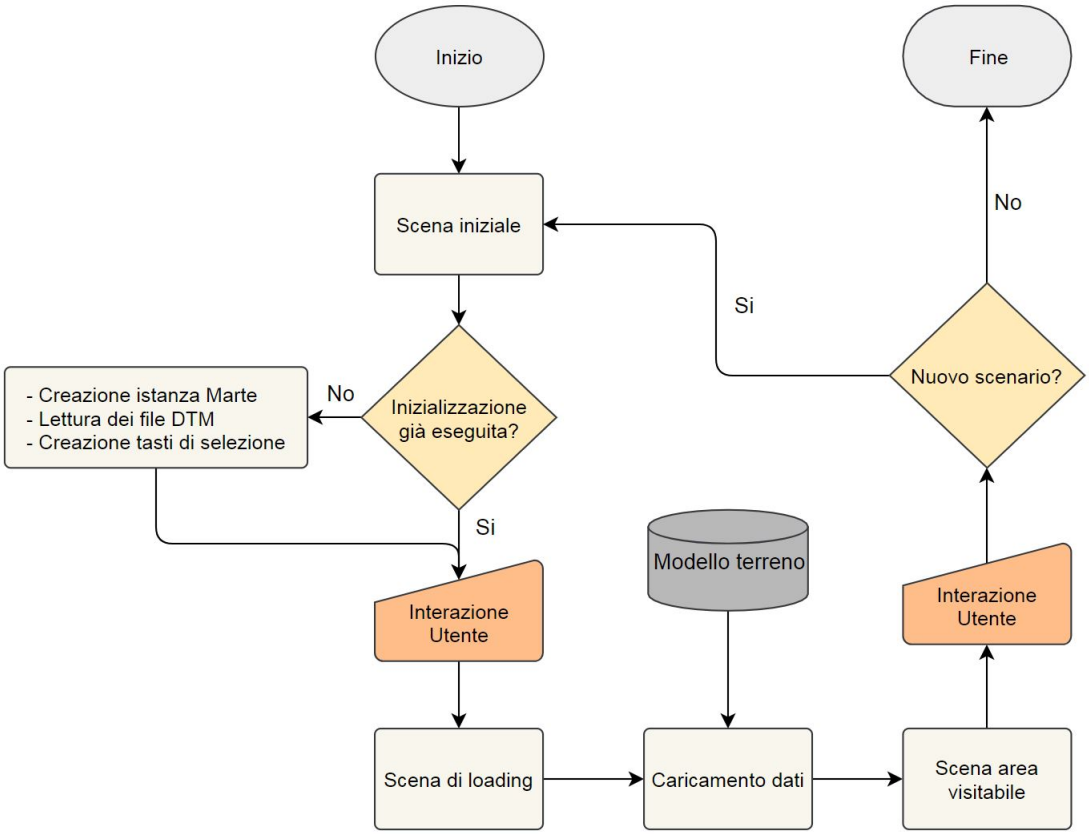


Figura 4.1: Diagramma di flusso dell'applicazione.

4.1.1 La scena iniziale

Nella scena iniziale l'utente può osservare il pianeta Marte nella sua interezza e selezionare una delle aree esplorabili disponibili da visitare.
Il pianeta è realizzato come una sfera perfetta a cui è stata applicata una texture e una normal map, ottenute dal sito *The Celestia Motherlode*[22]. Per ogni area visitabile, viene aggiunto un oggetto *Button* nell'esatta posizione in cui è situato il corrispondente terreno su Marte, in termini di latitudine e longitudine.
Vi è una fase di inizializzazione della scena in cui, all'avvio del programma, vengono svolte alcune operazioni preliminari:

1. Generazione dell'oggetto Marte: caricato in fase di runtime utilizzando la funzione *Resources.Load*.

2. Lettura dei file DTM: per ogni area visitabile viene mantenuto, nella cartella di progetto StreamingAssets, il corrispondente file DTM, con estensione *.IMG. Questi vengono letti tramite la libreria GDAL e ne vengono memorizzate le informazioni utili in oggetti di tipo *MarsTerrain*.

3. Generazione dei tasti di selezione: per ogni file letto viene caricato un oggetto Button e posizionato sul pianeta alla corrispondente latitudine e longitudine. A questi è assegnato un *Listener* che, una volta scatenato l'evento, permette di eseguire la funzione che effettua il cambio della scena.

In figura 4.2 è possibile vedere l'elenco di proprietà e metodi della classe MarsTerrain.

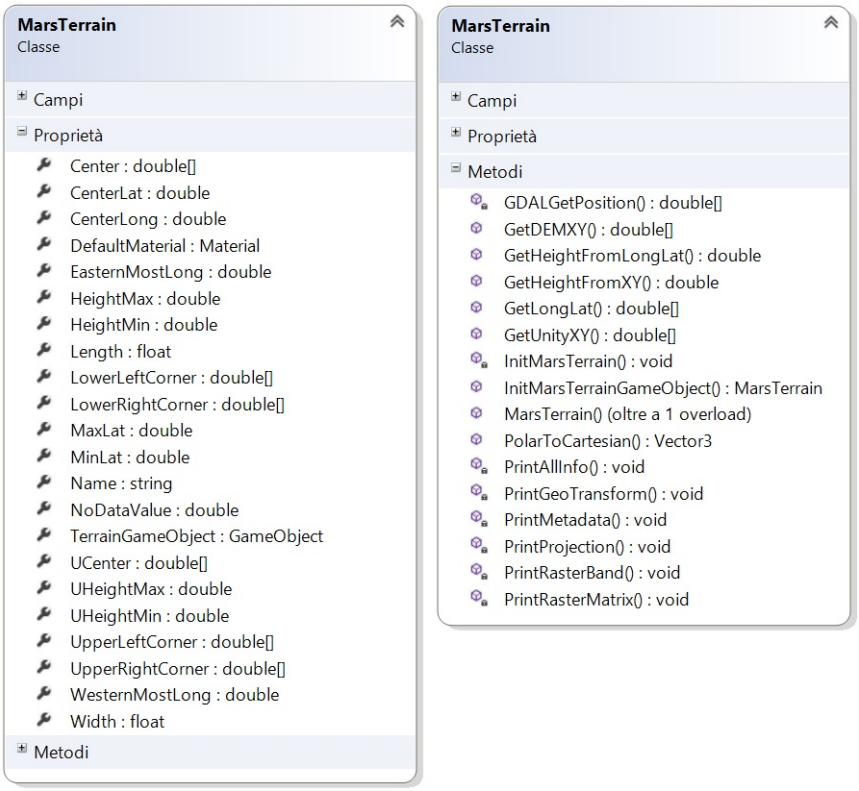


Figura 4.2: Diagramma della classe MarsTerrain.

Ogni oggetto MarsTerrain viene memorizzato in un dizionario, associato a una chiave rappresentata da un identificatore univoco: l'ID dell'oggetto Button corrispondente. La gestione del dizionario viene effettuata tramite la classe *TerrainDictionary*, un singleton[1] che permette di evitare di elaborare i file DTM a ogni cambio di scena, operazione che richiede uno sforzo significativo dal punto di vista computazionale.

4.1.2 La scena di loading

Il passaggio da una scena all'altra è stato gestito tramite la funzione di caricamento scena asincrona fornita da Unity: *SceneManager.LoadSceneAsync*.
La fase di inizializzazione della scena richiede un lasso di tempo di circa 10s, durante il quale viene caricato il terreno dell'area visitabile. Fintanto che la scena non è pronta viene visualizzata, nella scena di loading, una schermata di caricamento e viene impedita l'interazione dell'utente.

4.1.3 La scena dell'esplorazione

La scena dedicata all'esplorazione è unica per tutte le aree visitabili, il modello del terreno selezionato e tutti i suoi componenti vengono caricati in fase di runtime una volta che il cambio scena è stato avviato. Questa soluzione impatta sul tempo d'attesa per l'avvio che, come detto in precedenza, è di circa 10s, ma fornisce all'applicazione una maggior flessibilità. È possibile infatti apportare modifiche agli oggetti di scena una sola volta, senza il bisogno di replicarle per ogni area visitabile, inoltre, l'aggiunta di ulteriori zone esplorabili risulta notevolmente più semplice.
All'avvio della scena sono eseguiti due script, *TerrainLoader* e *InitTerrainScript*, che permettono di impostare il terreno, selezionato dall'utente nella prima scena, come l'oggetto MarsTerrain attivo, sul quale verranno eseguite tutte le operazioni richieste.
È la scena più complessa, comprende la gestione dell'ambiente virtuale, del modello del rover e la simulazione fisica dei suoi movimenti, la gestione di diverse inquadrature e tipologie di camere e quella delle funzionalità realizzate. Una descrizione dettagliata verrà trattata nel paragrafi successivi.

4.2 L'ambiente marziano

È stata già descritta in precedenza l'importanza di avere un ambiente virtuale che sia il più vicino possibile alla realtà; va quindi sottolineato come, per avere una simulazione fisica realistica, la forza di gravità utilizzata dal motore fisico di Unity sia stata impostata pari a quella che si trova sul pianeta Marte, ossia 3,711 m/s^2.
Il primo passo affrontato nella creazione del mondo virtuale è stato la generazione di un modello 3D di terreni marziani. Per realizzare tali modelli sono stati utilizzati alcuni file DTM presi dal catalogo di HiRISE[23]. La generazione di tali file viene effettuata utilizzando immagini sorgente a risoluzione molto elevata, permettendo così di ottenere DTM in cui si riesce a rappresentare circa 1 metro ogni pixel e a fornire una precisione di una decina di centimetri in termini di elevazione.
Il modello 3D del terreno viene creato sotto-campionando il file DTM, generando

[1]Un singleton è una classe che permette di creare una sola istanza della classe stessa.

un solo valore corrispondente alla media di ogni regione di dimensione 6x6 del file. Quest'operazione è necessaria per ridurre le dimensioni del modello 3D e migliorarne la gestione. Viene poi esportato in formato FBX e importato in Unity.
Il modello, a causa delle sue dimensioni, viene suddiviso in modo automatico in diverse geometrie (*Mesh*), ognuna delle quali necessita di avere un *MeshCollider*, un componente che permette di rilevare le collisioni sulla mesh a cui è applicato. Sono proprio i collider a far incrementare notevolmente la complessità computazionale della scena, ma sono indispensabili per attribuire agli oggetti delle proprietà fisiche in modo da ricreare una simulazione realistica.
Dalle numerose foto di Marte, scattate da satelliti, sonde e rover, si può notare come il suolo sia composto da un'alternanza di roccia e sabbia, è possibile vederne un esempio in figura 4.3. Per ottenere quest'effetto, è stato utilizzato uno shader[24] che permette di combinare più texture o materiali insieme. Avendo a che fare con modelli molto grandi, è stato necessario apportare delle modifiche allo shader riguardanti il modo in cui vengono applicate le texture (dimensioni, offset, intensità). La presenza di sabbia o roccia viene gestita utilizzando come maschera un'immagine in scala di grigi, ottenuta dal sito HiRISE, in cui in nero son raffigurate le zone di minor elevazione, sabbiose, mentre in bianco zone di maggior elevazione, rocciose.

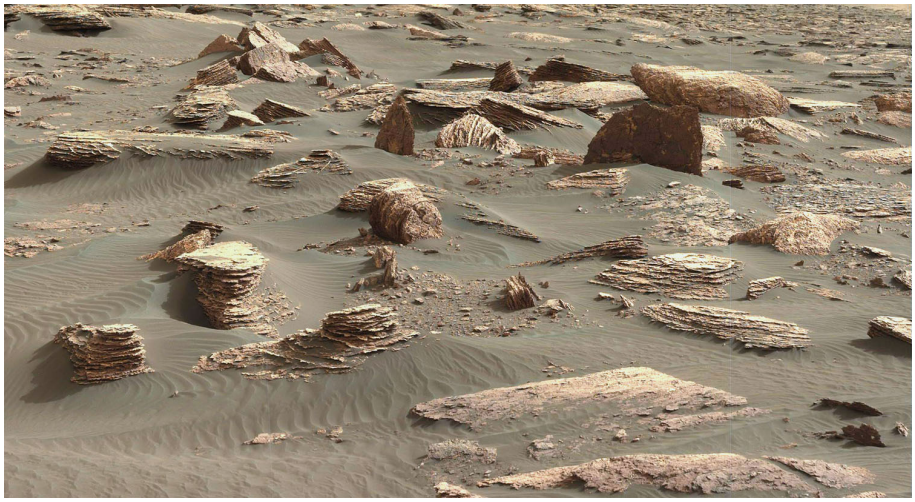


Figura 4.3: Immagine scattata da curiosity il 26 marzo 2017.

4.3 Il rover

Il rover utilizzato nell'applicazione è un modello di Curiosity, ottenuto dal sito[25] della NASA, che fornisce modelli 3D di diversi dispositivi aerospaziali. Il rover ha sei ruote, ognuna delle quali è dotata di un motore elettrico. Il sistema di sospensione che utilizza Curiosity è chiamato *rocker-bogie*, ed è composto da due braccia che fungono da bilancieri, connessi tra di loro e con il corpo da un differenziale. Questo particolare sistema permette al rover di superare ostacoli per oltre una volta e mezza il diametro delle ruote e, contemporaneamente, di mantenere sempre tutte le ruote a contatto con il terreno. In figura 4.4 è mostrato il sistema di sospensione del modello di Curiosity utilizzato. Ciò che rende complicato ricreare questo sistema in Unity è il numero di variabili che entrano in gioco. Numerosi sono infatti i vincoli che devono esser rispettati e spesso sono dipendenti gli uni dagli altri.

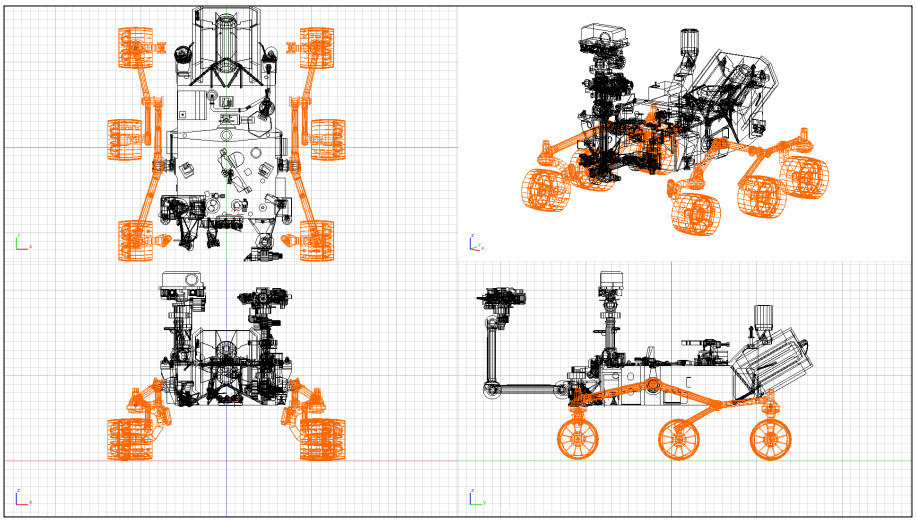


Figura 4.4: Sistema di sospensione rocker-bogie del modello di Curiosity utilizzato.

I problemi principali riscontrati sono stati due:

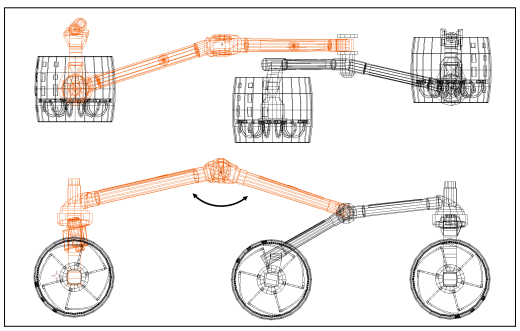
1. Fornire stabilità al rover
2. Riprodurre i reali movimenti del rover

Di seguito vengono analizzati i problemi e le soluzioni adottate.

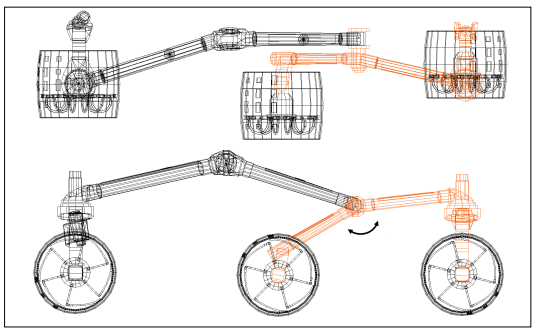
4.3.1 La stabilità

Unity mette a disposizione dei componenti, detti *Joint*, che permettono di collegare due oggetti controllati dal motore fisico, a cui sia stato assegnato un *Rigidbody*. Esistono diversi tipi di Joint che applicano restrizioni differenti.
Per ricreare il sistema di sospensione di Curiosity è stato innanzitutto necessario modificare il modello del rover, separando le singole parti che lo compongono. Ognuna delle due sospensioni, destra e sinistra, sarà quindi composta da due bilancieri:

- Anteriore: la parte evidenziata in arancione in figura 4.5a
- Posteriore: mostrato in figura 4.5b



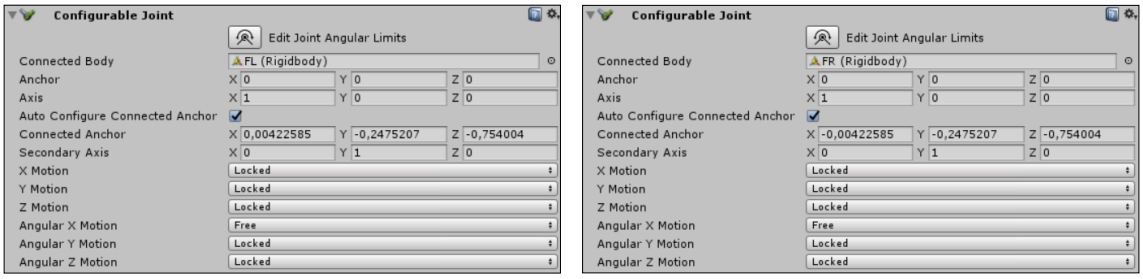
(a) Bilanciere anteriore del sistema di sospensione.



(b) Bilanciere posteriore del sistema di sospensione.

Figura 4.5: Sospensione sinistra del modello di Curiosity.

Per riprodurre il movimento dei bilancieri anteriori sono stati utilizzati due *Hinge Joint*, collegandoli al corpo del rover e specificando la posizione del perno attorno a cui ruotare e l'asse di rotazione. Con questo componente è possibile impostare dei limiti di rotazione, l'utilizzo di molle o di forze da applicare all'oggetto a esso connesso, nel caso in questione però nessuna di queste funzioni viene utilizzata, permettendo al bilanciere un movimento completamente libero attorno al perno. Questa scelta è stata fatta per riuscire a mantenere tutte e sei le ruote a contatto col terreno, rispettando così il comportamento del rover reale.
Per quanto riguarda i bilancieri posteriori è stato riprodotto il comportamento di un Hinge Joint attraverso un *Configurable Joint* che permette di agire su un numero maggiore di parametri. Oltre alle già citate funzioni per l'impostazione di limiti, molle e forze, è possibile specificare diversi assi di movimento e di rotazione del giunto. Come si può vedere in figura 4.6a e 4.6b, il movimento è stato bloccato in tutti e tre le direzioni, mentre per quanto riguarda la rotazione si è mantenuta solo quella lungo l'asse X.



(a) Configurable Joint per il bilanciere sinistro.

(b) Configurable Joint per il bilanciere destro.

Figura 4.6: Componenti Configurable Joint utilizzati.

Ci sono altri fattori che influiscono sulla stabilità che avrà il rover nell'applicazione. Uno è costituito dal valore che si assegna alla massa di Curiosity, o per esser più precisi al componente Rigidbody a esso assegnato. Il valore utilizzato equivale al peso che ha il rover della NASA nella realtà, vale a dire 900 Kg (In Unity questo valore è adimensionato, ma avendo utilizzato una scala 1:1 per i diversi componenti del progetto, viene considerato in Kg).
Un altro fattore che ha una notevole rilevanza per evitare strani comportamenti durante la simulazione è rappresentato da due coefficienti che intervengono quando si utilizzano i Joint, questi sono:

- *Mass Scale*: fattore di scala da applicare all'inverso della massa e al tensore inerziale del Rigidbody stesso.

- *Connected Mass Scale*: fattore di scala da applicare all'inverso della massa e al tensore inerziale del Rigidbody a cui è connesso il Joint.

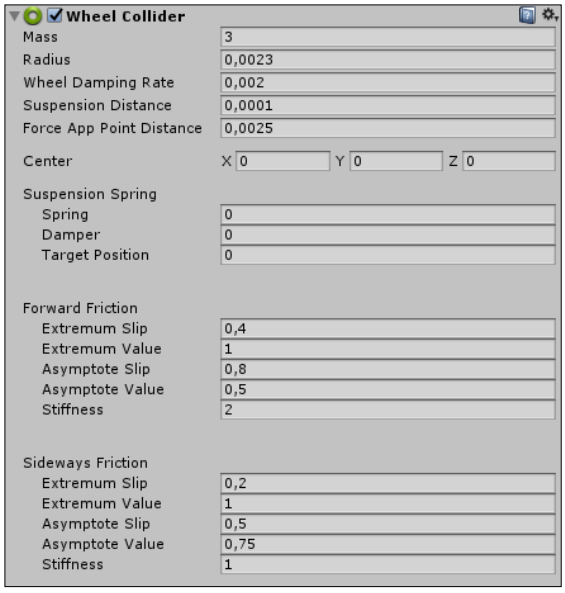
Questi coefficienti influenzano il modo in cui i due oggetti collegati tramite un Joint interagiscono. Infatti, applicando fattori di scala tali da ottenere masse e tensori inerziali risultanti simili, la serie di corpi connessi tramite Joint risulterà meno nervosa e gli oggetti individuali meno distanti tra loro. I valori utilizzati che permettono

di ottenere la stabilità desiderata di Curiosity sono:

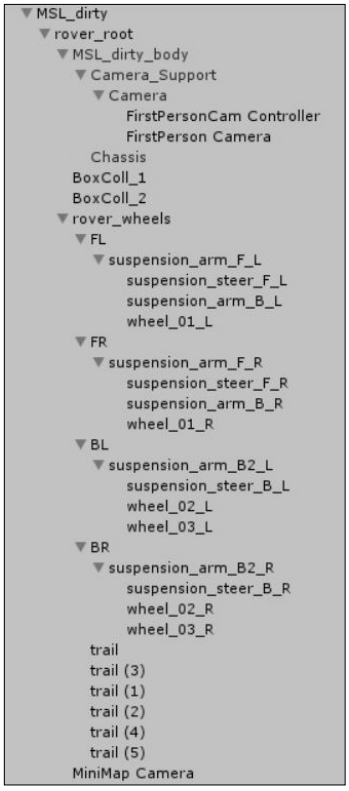
- Bilanciere anteriore:
  - Massa del Rigidbody: 10
  - Massa del Rigidbody connesso (rover): 900
  - Mass Scale: 1
  - Connected Mass Scale: 100
- Bilanciere posteriore:
  - Massa del Rigidbody: 1
  - Massa del Rigidbody connesso (bilanciere anteriore): 10
  - Mass Scale: 1
  - Connected Mass Scale: 1

4.3.2 Il movimento

Per ricreare il comportamento di un veicolo con ruote, che si sposta su terra, in Unity è necessario utilizzare un componente chiamato *Wheel Collider*, che integra diverse funzioni: rilevamento di collisioni, fisica e attrito delle ruote.



(a) Wheel Collider.



(b) Gerarchia del rover (*MSL_dirty*) in Unity.

Figura 4.7: Componenti Unity.

Diversi sono i parametri che permettono di influenzare le sue funzionalità, ad esempio: la massa, il centro e il raggio della ruota, la massima estensione della sospensione, il punto in cui vengono applicate le forze alla ruota, parametri per le sospensioni e la forza d'attrito.
Nello specifico i valori assegnati ai diversi parametri sono mostrati in figura 4.7a. Per quanto riguarda la massa delle ruote è stato approssimato in eccesso il peso delle ruote di Curiosity a 3 Kg. I parametri per le sospensioni sono nulli perché il sistema di sospensione è stato ricreato ad-hoc come visto in precedenza. I valori per la frizione in avanti e quella laterale sono stati calcolati sperimentalmente.
La gerarchia dell'oggetto rover utilizzato in Unity, mostrata in figura 4.7b viene descritta nel dettaglio di seguito:

- *rover_root*: è una empty, padre di tutti gli altri elementi.
- *MSL_dirty_body*: la mesh del rover, comprende il corpo e la camera (gli oggetti *FirstPerson Camera*, *FirstPersonCam Controller* e *MiniMap Camera* verranno analizzate in dettaglio nel capitolo 4.4 a essi dedicato).
- *BoxColl_1* e *BoxColl_2*: sono i Box Collider del corpo del rover, che permettono di rilevarne le collisioni.
- *rover_wheels*: empty padre del sistema di sospensione del rover e delle ruote. Questi elementi sono ulteriormente divisi in:
  - *FL* (Front Left): bilanciere anteriore sinistro
  - *FR* (Front Right): bilanciere anteriore destro
  - *BL* (Back Left): bilanciere posteriore sinistro
  - *BR* (Back Right): bilanciere posteriore destro
- *trail*: 6 empty posizionate nel punto di conttatto tra le ruote e il terreno, utilizzate per renderizzare i segni lasciati dalle ruote sul suolo. Per maggiori dettagli verdere il paragrafo 4.7.

Le ruote, a cui i Wheel Collider sono applicati, sono controllate andando ad agire via script su tre variabili che permettono di: generare una forza per far muovere il veicolo (*motorTorque*), generare una forza frenante (*brakeTorque*) e impostare un angolo di sterzata (*steerAngle*). Per interagire con il motore fisico è necessario utilizzare la funzione *FixedUpdate* che, a differenza della *Update* normale, viene eseguita con un framerate fisso.
Lo script che gestisce il movimento è chiamato *MSLController_v2*, in figura 4.8 è possibile vedere l'elenco dei parametri che possono essere impostati da Unity.
Nel metodo Start, utilizzato per l'inizializzazione, viene eseguita una funzione chiamata *ConfigureVehicleSubsteps*, a cui sono passati i valori assegnati ai tre parametri del secondo riquadro rosso in figura 4.8. A ogni esecuzione della funzione FixedUpdate, la simulazione suddivide ulteriormente il Δt fisso in intervalli minori, per ognuno dei quali vengono calcolate le forze che agiscono sulle ruote; le forze risultanti vengono integrate e applicate al veicolo. Utilizzando la funzione ConfigureVehicleSubsteps è possibile impostare il numero di sotto passaggi che verranno eseguiti dalla simulazione sotto e sopra il valore di *speedTreshold*.

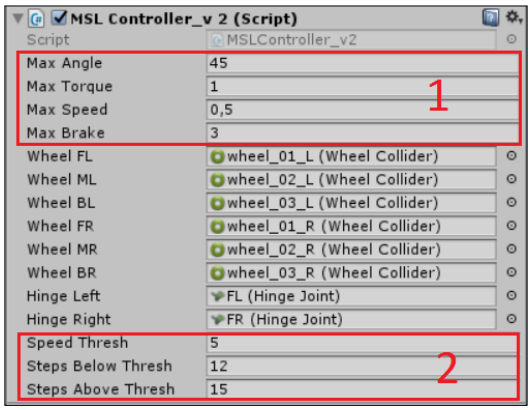


Figura 4.8: Pannello di MSLController_v2 in Unity.

Nel primo riquadro rosso invece vengono assegnati i valori a diverse variabili che definiscono:

- *Max Angle* $[deg]$: massimo angolo di sterzata
- *Max Torque* $[Nm]$: massima forza torcente
- *Max Speed* $[m/s]$: massima velocità raggiungibile
- *Max Brake* $[Nm]$: massima forza frenante

L'utente ha la possibilità di inviare degli input ai motori di destra e ai motori di sinistra, consentendo così al rover di procedere avanti o indietro, in linea retta o curvando, oppure di ruotare sul posto. Le ruote anteriori e posteriori sono dotate di un meccanismo che permette loro di sterzare fino al massimo angolo di sterzata, definito precedentemente, consentendo così al rover di ruotare sul posto.
Nel riprodurre i movimenti di Curiosity in modo fedele occorre far ruotare anche questi meccanismi, ossia gli elementi *suspension_steer_*_** che troviamo in figura 4.7b. Se questi facessero parte delle ruote, girerebbero con esse anche quando il rover procede avanti o indietro; per far loro seguire il movimento delle ruote solo quando a queste viene applicato un angolo di sterzata, è stato necessario calcolare la differenza tra l'attuale angolo di sterzata delle ruote e quello che avevano in precedenza, ed eseguire una rotazione di tale ampiezza dei meccanismi sul proprio asse verticale; avendo cura di eseguire anche l'operazione inversa e di non superare il massimo angolo di sterzata.
Un altro accorgimento che è stato preso riguarda la velocità massima che Curiosity può raggiungere. La velocità a cui il rover può muoversi nella simulazione è di circa 50 cm/s, dieci volte la reale velocità massima. Questa scelta è stata fatta sia per rendere più fluida la simulazione, sia perché velocità molto basse non sono gestite in maniera ottimale dal motore fisico di Unity.
La soluzione adottata per riuscire a mantenere la velocità del rover al di sotto della velocità massima impostata prevede diverse operazioni, che si differenziano a seconda degli input esterni. Innanzitutto viene eseguito un controllo sull'attuale velocità di Curiosity, e viene salvato in una variabile il risultato, che indicherà se è stata superata o meno. Fintanto che la velocità massima non viene superata, gli input dell'utente vengono eseguiti normalmente, altrimenti si è creato un sistema auto-frenante, che consente di invertire il verso della forza che viene applicata per

Input motore DESTRO	**Input motore SINISTRO**	**Velocità massima NON SUPERATA**	**Velocità massima SUPERATA**	**AZIONE RISULTANTE**
Avanti (Indietro)	Avanti (Indietro)	Forza motrice applicata a ENTRAMBI i motori nel verso di marcia	Forza motrice applicata a ENTRAMBI i motori nel verso opposto a quello di marcia	Il rover procede in avanti (indietro) senza superare la velocità massima
Avanti (Indietro)	Fermo	Forza motrice applicata al motore DESTRO nel verso di marcia	Forza motrice applicata al motore SINISTRO nel verso opposto a quello di marcia	Il rover procede in avanti (indietro) curvando verso sinistra senza superare la velocità massima
Fermo	Avanti (Indietro)	Forza motrice applicata al motore SINISTRO nel verso di marcia	Forza motrice applicata al motore DESTRO nel verso opposto a quello di marcia	Il rover procede in avanti (indietro) curvando verso destra senza superare la velocità massima
Avanti (Indietro)	Indietro (Avanti)	Forza motrice applicata a ENTRAMBI i motori nello stesso verso degli input	Forza motrice applicata a ENTRAMBI i motori in verso opposto agli input	Il rover ruota sul posto in senso antiorario (orario)
Fermo	Fermo	—	—	Il rover resta fermo e frenato

Tabella 4.1: Funzionamento del rover a seguito di input esterni.

mantenere la velocità sotto il limite massimo. In tabella 4.1 viene spiegato in modo dettagliato questo funzionamento, andando a descrivere per ogni combinazione di input da parte dell'utente l'azione risultante e le operazioni che vengono eseguite quando la velocità del rover è sotto o sopra il limite massimo impostato.
L'ultimo punto da analizzare riguarda l'inclinazione del corpo del rover che, con il sistema di sospensione Rocker-Bogie, è regolata da un differenziale meccanico. Per riprodurre questo comportamento sono state valutate diverse opzioni:

1. La prima configurazione prevedeva di demandare la simulazione di questo comportamento al motore fisico, andando a impostare un opportuno centro di massa. Si è però subito dimostrata una soluzione poco efficacie poiché soggetta a numerose oscillazioni causate dai cambi di direzione e dalla combinazione di diverse manovre del rover.

2. Poiché bloccare la rotazione del rover lungo il proprio asse trasversale non è una soluzione accettabile, in quanto non modella correttamente il comportamento del sistema di sospensione di Curiosity, la seconda configurazione prevede una combinazione delle due soluzioni. Si valuta l'angolo dei due Joint frontali:

   - Se entrambi gli angoli sono maggiori di un range prefissato: il rover è andato incontro a una salita/discesa, l'inclinazione del corpo va quindi modificata e viene utilizzata la prima configurazione.
   - Se entrambi gli angoli sono minori di un range prefissato: l'inclinazione del corpo ha già raggiunto una posizione stabile, in cui è, in modo approssimato, parallelo al terreno, viene quindi bloccata la rotazione sull'asse trasversale per evitare ondulazioni eccessive.

   Anche questa soluzione, seppur migliore della precedente non si è rivelata adatta alla qualità della simulazione desiderata.

3. La terza configurazione prevede che, ogni volta che viene eseguita la funzione FixedUpdate, venga effettuata una rotazione del corpo del rover lungo l'asse

trasversale di un angolo che corrisponde alla media tra l'angolo del bilanciere anteriore sinistro e l'angolo del bilanciere anteriore destro. Questa configurazione approssima in maniera corretta il comportamento del differenziale del sistema di sospensione rocker-bogie ed è la soluzione definitivamente adottata.

4.4 Le camere

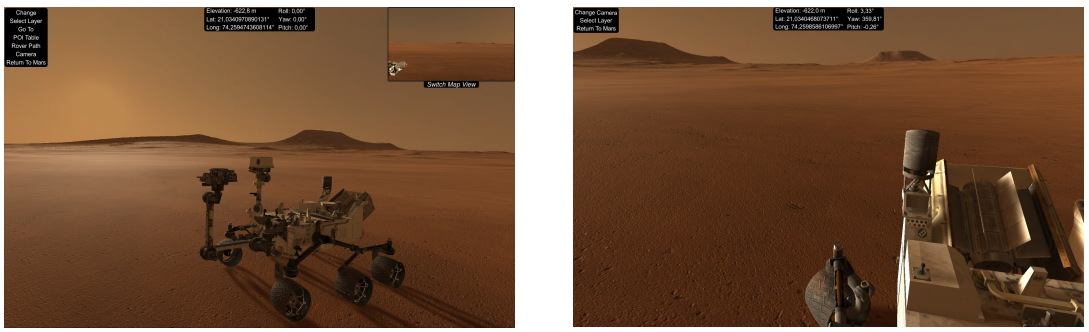
Le scene utilizzano diverse camere, fornendo svariate prospettive e diverse modalità d'interazione. L'applicazione è utilizzabile in modalità stereoscopica o no, in base a questa scelta si avranno impostazioni differenti per le camere. Di seguito vengono descritte le diverse configurazioni e le inquadrature disponibili, successivamente si tratterà la parte specifica per la stereoscopia.

4.4.1 La configurazione delle camere

Nella scena iniziale è presente un'unica inquadratura ed è permessa l'iterazione dell'utente tramite un comune mouse 2D, gestita con uno script. Tenendo premuto il tasto sinistro del mouse e spostando il cursore è possibile far ruotare la camera attorno al pianeta, consentendo di vedere tutta la superficie di Marte; è inoltre possibile effettuare uno zoom avanti o indietro. Sono impostati dei limiti sia nella distanza minima dal pianeta che nel massimo angolo di rotazione attorno all'asse orizzontale. Per rendere la rotazione più fluida viene utilizzata la funzione *Quaternion.Lerp*, che effettua un'interpolazione lineare tra due parametri, la rotazione attuale e quella desiderata, e infine normalizza il risultato.
Nella scena per l'esplorazione terrena sono presenti diverse camere con cui l'utente può interagire. La gestione di queste e i cambi d'inquadratura sono affidati a uno script chiamato *StereoCameraController*.

Main Camera

È la camera principale della scena che fornisce una visione periferica, in terza persona, del rover e del terreno marziano o un'inquadratura da una delle on-board camera di Curiosity, la cosiddetta MastCam.
La gestione delle diverse inquadrature è affidata a due controller che consistono in



(a) Prospettiva dal ThirdPersonCam Controller.

(b) Prospettiva dal FirstPersonCam Controller.

Figura 4.9: Immagini tratte da Mars Navigation da prospettive differenti.

due empty. L'oggetto *Main Camera* viene fatto coincidere, in termini di posizione e rotazione, con il controller che gestisce l'inquadratura selezionata; viene poi impostato un vincolo di parentela, grazie al quale ogni traslazione o rotazione applicata al padre (il controller) verrà applicata anche al filgio (la camera).

- *ThirdPersonCam Controller*: posizionato a una certa distanza dal rover, fornisce una visione ampia sul terreno e Curiosity (figura 4.9a). Il controller, e di conseguenza la camera a esso imparentata, è rivolto sempre verso un punto di riferimento, che consiste nel rover stesso e che sarà sempre al centro dell'inquadratura, permettendo all'utente di girarci attorno ed effettuare zoom in avanti o indietro. Da quest'inquadratura è possibile interagire con il pannello che racchiude tutte le funzionalità dell'applicazione.
- *FirstPersonCam Controller*: posizionato in corrispondenza della MastCam di Curiosity, fornisce una visione in prima persona dalla camera on-board (figura 4.9b). Da quest'inquadratura è possibile accedere a un numero ristretto di funzionalità, alcuni elementi dell'interfaccia grafica vengono nascosti per permettere all'utente di avere una visione più dettagliata dell'ambiente.

MiniMap Camera e FirstPerson Camera

Sono due camere le cui immagini sono renderizzate su una texture, visualizzata a video in un riquadro in alto a destra dello schermo (figura 4.10).

- *MiniMap Camera*: camera posizionata sull'asse verticale del rover, a 10 unità Unity di distanza, corrispondenti a 10 metri. Fornisce una visione dall'alto del rover e non prevede nessun tipo d'interazione da parte dell'utente.
- *FirstPerson Camera*: come per il FirstPersonCam Controller, questa camera si trova in corrispondenza della MastCam di Curiosity ed è utilizzata per avere la visuale della camera on-board anche nell'inquadratura esterna. A differenza della MiniMap Camera, è possibile interagire con essa ruotandola in ogni direzione, con un limite sull'angolo di rotazione dell'asse trasversale.

4.4.2 La modalità stereoscopica

Per fare in modo che la stereoscopia attiva funzioni è necessario che la scheda video riconosca il flusso di dati in output come un flusso video stereoscopico. In Unity per sistemi Linux e MacOSX l'API grafica predefinita è OpenGL Core, mentre per sistemi Windows è necessario impostarla manualmente; un'ulteriore impostazione da abilitare è il supporto per le funzionalità specifiche per la Virtual Reality. Questi due passaggi permettono di sviluppare in Unity un'applicazione VR e di fare accesso a impostazioni specifiche.
Per esser in grado di funzionare in stereoscopia attiva l'applicazione deve produrre un immagine per l'occhio destro e un'immagine per l'occhio sinistro dell'utente. Non è però necessario inserire due camere differenti grazie alla possibilità di specificare, per la singola camera, se le immagini riprese sono destinate a un singolo occhio oppure a entrambi, come in ne caso in questione. In questo modo la singola camera si comporta come se riprendesse due frame distinti, uno per l'occhio destro e l'altro per il sinistro.

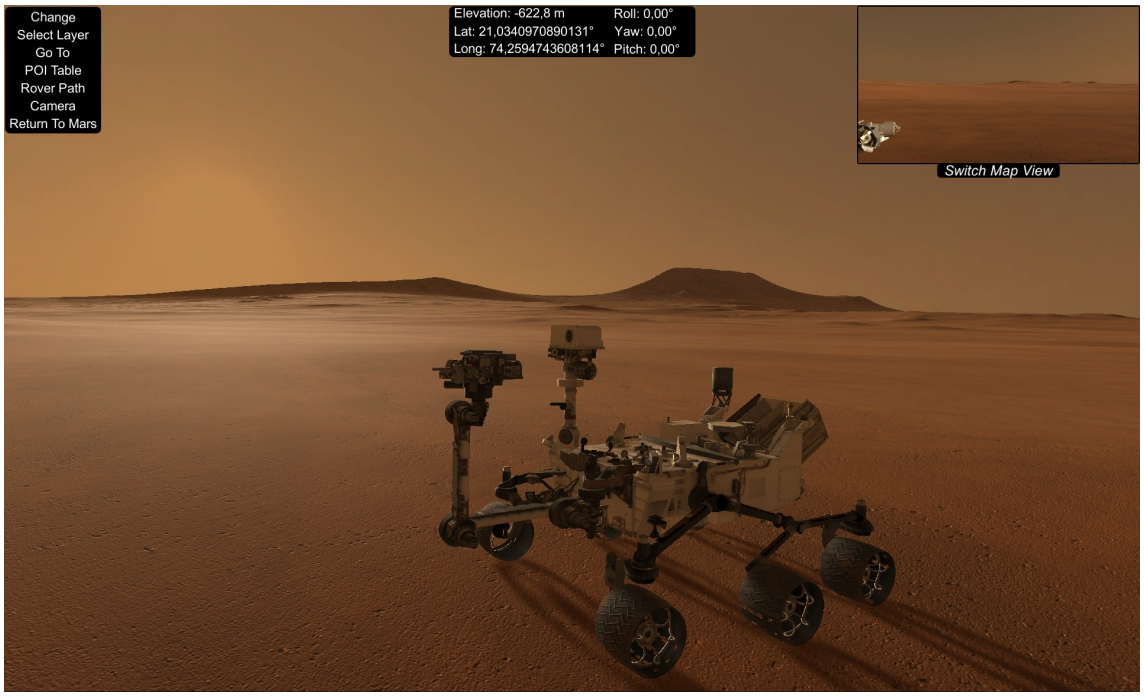


Figura 4.10: Immagine di Mars Navigation.

A influenzare la stereoscopia sono altri due parametri che è necessario impostare:

- *Stereo Separation*: è la distanza tra gli occhi, impostata a 0.032 unità Unity, corrispondenti a una distanza interpupillare di 64mm.
- *Stereo Convergence*: distanza a cui gli occhi convergono, impostata a 10m.

4.5 Le funzionalità

In Mars Navigation sono state realizzate diverse funzionalità che permettono all'utente di ottenere informazioni riguardanti rover e terreno e di interagire con essi. In questa sezione vengono analizzate una per una andando a descriverne il funzionamento e il modo in cui sono state realizzate.

4.5.1 Posizione, elevazione e rotazione

In un pannello posto in alto in posizione centrale rispetto alla schermata (figura 4.10), vengono visualizzate le informazioni riguardanti:

- La posizione del rover espressa in latitudine e longitudine.
- L'elevazione del terreno alla posizione in cui si trova il rover.
- La rotazione del rover espressa in gradi rispetto ai tre assi: longitudinale (roll), trasversale (pitch) e verticale (yaw).

Posizione

In figura 4.11 è possibile vedere uno schema che raffigura le diverse conversioni tra sistemi di riferimento necessarie per ottenere le informazioni riguardanti la posizione del rover.
Nel file DTM, il centro del terreno è identificato da una coppia di coordinate (x, y) che possono assumere valori anche molto elevati. In Unity esiste però un problema legato al fatto che il sistema di coordinate utilizzato è espresso in floating point, le coordinate X, Y e Z avranno quindi una limitazione di 7 cifre significative. Questo sta a significare che più ci si allontana dall'origine (x = y = z = 0.000000) più l'imprecisione aumenta. Ad esempio, assumendo che 1u (unità Unity) corrisponda a 1m, un oggetto posizionato in 1.234567 avrà una precisione di 6 cifre decimali, mentre un oggetto posizionato in 765432.1 avrà una precisione di una sola cifra decimale. Per questo motivo il modello 3D del terreno viene posizionato con il proprio centro nell'origine del sistema di riferimento di Unity, memorizzando in due variabili l'offset che si ha con il centro reale del terreno (quello nel DTM).
La conversione A indicata in figura 4.11 consiste quindi nel sommare gli offset calcolati precedentemente alle coordinate X e Z di Unity, tralasciando l'asse verticale Y.
La conversione B necessita dell'utilizzo di alcune funzioni che la libreria GDAL fornisce. La classe *OSGeo.OSR.CoordinateTransformation* mette a disposizione il metodo *CreateCoordinateTransformation(SpatialReference src, SpatialReference dst)*, che consente di creare un oggetto CoordinateTransformation a partire da due sistemi di riferimento spaziale, definiti dalla classe *OSGeo.OSR.SpatialReference*. Applicando la trasformazione ottenuta a un punto nel sistema di riferimento sorgente (quello del DTM) si ottiene il corrispondente punto nel sistema di riferimento destinazione (quello geografico).

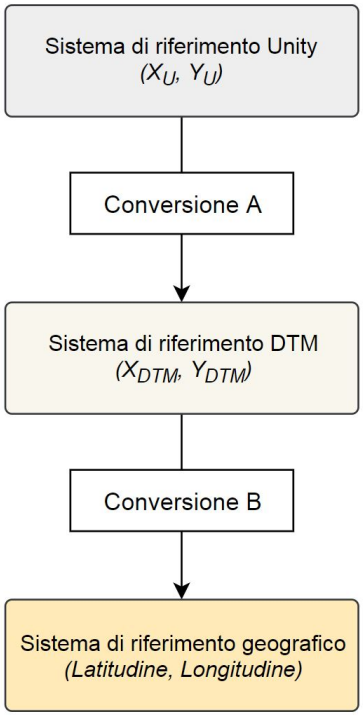


Figura 4.11: Schema di conversione tra sistemi di riferimento.

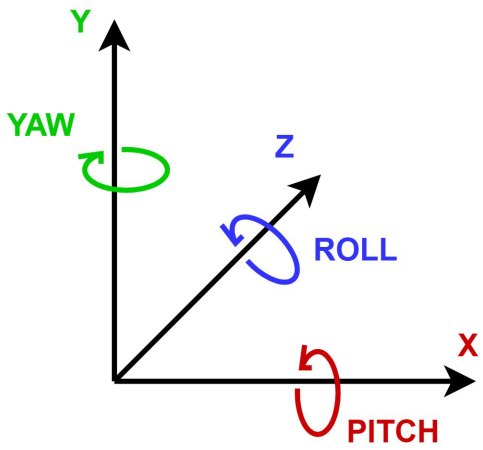


Figura 4.12: Sistema di riferimento di Unity.

Elevazione

Per identificare una posizione esatta è necessario un terzo dato oltre a latitudine e longitudine: l'elevazione. L'altezza del punto del terreno su cui si trova il rover viene calcolata secondo la formula:

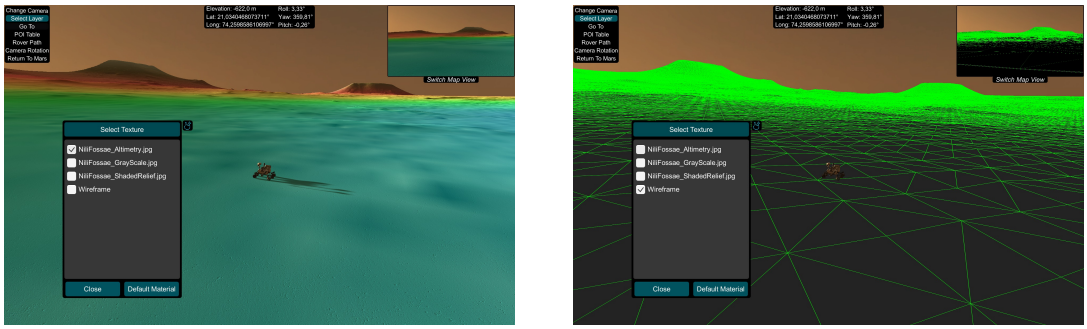
$$h_{\mathrm{DTM}} = Min_{\mathrm{U}} + \frac{(h_{\mathrm{U}} - Min_{\mathrm{U}})(Min_{\mathrm{DTM}} - Min_{\mathrm{DTM}})}{(Max_{\mathrm{U}} - Min_{\mathrm{U}})} \tag{4.1}$$

Dove:

- h_{DTM} è il valore dell'altezza del terreno nel DTM, da calcolare
- h_{U} è il valore dell'altezza del terreno in Unity
- Min_{U} è l'altezza minima del terreno nel sistema di riferimento di Unity
- Max_{U} è l'altezza massima del terreno nel sistema di riferimento di Unity
- Min_{DTM} è l'altezza minima del terreno nel sistema di riferimento del DTM
- Max_{DTM} è l'altezza minima del terreno nel sistema di riferimento del DTM

Rotazione

In figura 4.12 è mostrato il sistema di riferimento utilizzato da Unity, in cui viene riportata la corrispondente rotazione associata agli assi. Per una miglior comprensione, le informazioni sugli angoli di rotazione del rover rispetto agli assi roll e pitch vengono riportati in un range compreso tra -180° e 180°, mentre per l'asse yaw il range va da 0° a 360°. Tali angoli corrispondono alla rotazione del corpo del rover rispetto al sistema di riferimento globale; vengono calcolati attraverso gli angoli di Eulero e convertiti successivamente in gradi.



(a) Mappa altimetrica.

(b) Modalità wireframe.

Figura 4.13: Immagine dell'area di Nili Fossae a cui vengono applicati layer differrenti.

4.5.2 Selezione dei layer

Oltre alla visualizzazioni di dati relativi al rover è importante poter visualizzare altre tipologie di informazioni che riguardano il sito che si sta visitando. Con questa finalità è stata realizzata una modalità che permette la selezione di diversi layer visualizzabili sul terreno. Come è possibile vedere in figura 4.13, quando viene abilitata questa funzionalità si può scegliere da un elenco una texture appositamente elaborate o una modalità di visualizzazione da applicare al terreno, come ad esempio:

- Mappa altimetrica (figura 4.13a)
- Immagini orbitali
- Modalità wireframe (figura 4.13b)
- Etc.

Le immagini per poter esser visualizzate in modo corretto sul terreno, devono esser adeguatamente elaborate, sono perciò stati utilizzati dei file ottenuti dal sito di HiRISE e specifici per i terreni disponibili.
Per quanto riguarda la modalità wireframe è stato utilizzato uno shader che permette di visualizzare a video gli spigoli del modello del terreno 3D.
Il caricamento dell'elenco di tali file è dinamico, è quindi possibile andare ad aggiungere un'immagine successivamente all'avvio dell'applicazione che verrà caricata durante l'esecuzione. Ogni volta che viene richiamata la funzione di selezione layer, l'applicazione accede a una specifica cartella, per ogni file corrispondente a un'immagine (estensione: jpg, jpeg, png, gif, bmp, tiff) viene creato un elemento *Toggle* e aggiunto a una *ScrollView*[2].
Gli elementi Toggle hanno uno stato associato a essi che permette di capire se sono stati selezionati o meno (stato On/Off), nel caso in questione, la modalità di selezione è esclusiva: solo un Toggle alla volta può esser in stato On. A ogni cambiamento di stato, viene scatenato un evento a cui è stato associato un *Listener* che carica l'immagine da file. Il tratto di codice che gestisce questo processo è descritto in modo schematico diagramma in figura 4.14.

[2]Una ScrollView è una lista di un numero potenzialmente elevato di elementi, che è possibile scorrere senza che vengano rappresentati tutti quanti insieme. Per creare una ScrollView in Unity, è necessario utilizzare il componente *Scroll Rect* associato con una maschera che consente di visualizzare solo parte degli elementi effettivamente contenuti.

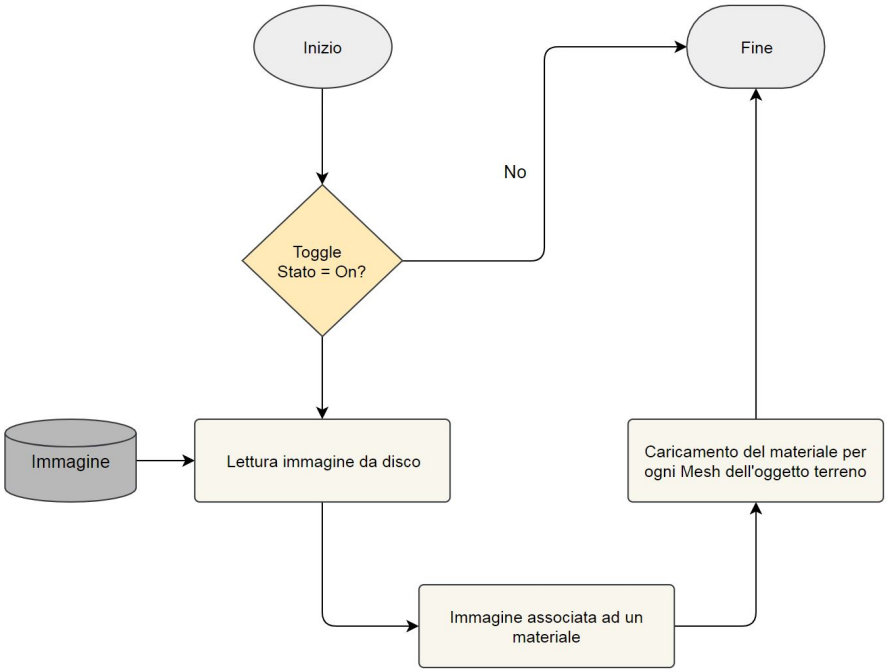


Figura 4.14: Diagramma di flusso per il caricamento del layer selezionato.

4.5.3 Scelta di un target

Una delle funzionalità principali di Mars Navigation è la possibilità di selezionare una posizione sul terreno marziano da far raggiungere a Curiosity, l'immagine 4.15 ritrae il rover che avanza verso l'obiettivo. Di seguito verranno analizzate le due modalità realizzate per la selezione del target e il modo in cui il rover raggiunge l'obiettivo.

Modalità Go To

In questa modalità la posizione da far raggiungere al rover viene specificata indicando una coppia di coordinate (latitudine, longitudine). Abilitando la funzione *Go To*, viene attivato un pannello, visibile in figura 4.15, in cui è possibile inserire le coordinate desiderate in due *InputField*. All'inserimento dei due valori si verifica che la coordinata immessa sia compresa tra il rispettivo valore di massimo e minimo. Un secondo controllo di questo tipo viene effettuato quando viene inviato il comando al rover di raggiungere il punto specificato.
Una volta inserite delle coordinate ritenute corrette, si effettua una conversione inversa a quella descritta nel paragrafo 4.5.1:

1. Viene creata una *CoordinateTrasformation*, tramite la funzione *CreateCoordinateTransformation*, che sia in grado di convertire dal sistema di riferimento geografico a quello del DTM.

2. Tramite la funzione *TransformPoint* si ottiene un punto identificato dalla coppia di coordinate $(X_{\mathrm{DTM}}, Y_{\mathrm{DTM}})$.

3. Viene effettuata una conversione in coordinate Unity e ritornato il risultato.

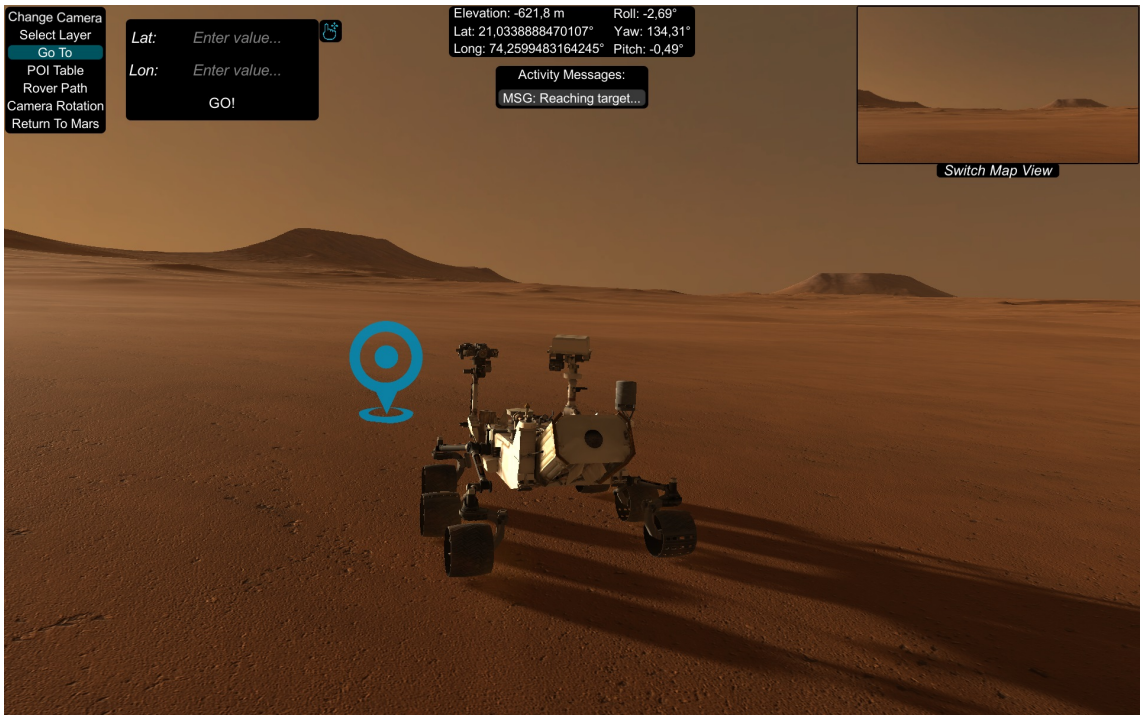


Figura 4.15: Immagine di Curiosity in avvicinamento verso il target selezionato.

Dopo queste operazioni è necessario calcolare l'altezza del punto da raggiungere, ottenuta accedendo al raster del DTM e applicando la formula 4.1 per calcolare il valore di h_{U} dato h_{DTM}. Infine viene visualizzato a video un marker che identifica l'obiettivo sul terreno.

Modalità Click-&-Go

In questa seconda modalità la posizione viene specificata cliccando con il tasto destro del mouse su un punto del terreno. Al click dell'utente, viene tracciato un raggio che parte dal piano della camera e passa attraverso una posizione indicata dalle coordinate (x, y) di un pixel sullo schermo, ignorando la coordinata z. La funzione che permette di eseguire quest'operazione è *Camera.ScreenPointToRay(Vector3 position)*, a cui viene passato come parametro la posizione del cursore del mouse, e viene ritornato un oggetto *Ray*.
Tramite la funzione *Physics.Raycast(Ray ray, out RaycastHit hitInfo)* è possibile verificare se questo raggio interseca un collider e, nel caso in cui questo avvenga, identificare il punto in cui vi è l'intersezione.
Una volta individuato il target che l'utente vuole raggiungere, viene visualizzato il marker per identificarlo.

Raggiungere il target

Lo script che gestisce la definizione di un nuovo target e che avvia il processo di avvicinamento del rover a tale obiettivo è chiamato *PropertiesAndCoroutines*, è collegato all'oggetto rover ed è possibile impostare alcuni parametri che ne modificano il comportamento:

- *Dist Range*: la distanza limite rispetto alla quale il target viene considerato raggiunto
- *Rot Range*: range di rotazione entro il quale il rover si considera rivolto verso l'obiettivo
- *Forward*: vettore di tre elementi che specifica la direzione di riferimento di Curiosity

Prima di analizzare in modo approfondito lo script è necessario introdurre il concetto di *Coroutine*. Una funzione normale, ritorna solo una volta che è stata eseguita completamente. Ciò significa che ogni azione che si svolge in una funzione deve esser completata durante l'aggiornamento di un singolo frame, una chiamata a funzione non può quindi eseguire una sequenza di eventi che perdura nel tempo. Una coroutine è una funzione che può mettere in pausa la sua esecuzione e passare il controllo a Unity, ma ha anche la capacità di riprendere, da dove aveva interrotto l'esecuzione, al frame successivo.
Il file PropertiesAndCoroutines espone la proprietà pubblica *Vector3 Target* che consiste in un vettore di tre elementi corrispondenti alle coordinate (x, y, z) che definiscono una posizione nello spazio. Ogni volta che viene assegnato un valore alla proprietà, vengono fermate le coroutine attualmente in esecuzione, inizializzate alcune variabili di controllo e avviata la coroutine *Movement(Vector3 target)* che riceve come parametro un target e dà inizio alle operazioni necessarie per raggiungere il punto selezionato.
Questa coroutine gestisce solamente le chiamate ad altre due, che controllano le azioni che deve eseguire il rover:

- *Rotation(Vector3 target)*: permette al rover di girarsi verso il target fissato, secondo il vettore forward definito in fase di inizializzazione. In questo caso non viene considerata l'altezza del punto da raggiungere, coordinata y, ma solamente la posizione rispetto alle coordinate (x, z) di Unity.
  Una volta calcolata la direzione verso cui il rover deve girarsi, è necessario effettuare il movimento di spin del rover. Nello script MSLController_v2 (paragrafo 4.3.2), è presente una funzione pubblica tramite la quale viene definito il verso di rotazione e di conseguenza applicata la corretta forza alle ruote di Curiosity in modo da iniziare lo spin.
  Fintanto che la direzione in cui punta il rover non rientra nel range di rotazione definito in fase di inizializzazione, impostato a 10°, la coroutine non ritorna completamente.
  Una volta terminata la rotazione, la coroutine di rotazione si conclude e viene avviata la successiva.

- *Position(Vector3 target)*: coroutine successiva a quella di rotazione, fa avanzare il rover fino a che il target non viene raggiunto. Anche in questo caso è necessario interfacciarsi con lo script MSLController_v2. Fintanto che la distanza dall'obiettivo non è inferiore al range prefissato (impostato a 1m), viene aggiornata la direzione verso cui sta andando il rover e, nel caso fosse necessario, viene corretta riducendo o aumentando la forza applicata ai motori destro o sinistro.
  Una volta che il controllo sulla distanza è soddisfatto, viene segnalato che l'obiettivo è stato raggiunto e terminano tutte le coroutine.

Essendo il movimento del rover gestito tramite il motore fisico di Unity, la sua traiettoria viene influenzata anche da fattori esterni, ad esempio dalla conformazione del terreno. Ipotizzando che solo le ruote a destra incontrino un ostacolo, come può esser una roccia, il veicolo tenderà a curvare verso destra, poiché le ruote da quel lato verranno rallentate. Verrà quindi effettuata una modifica alle forze che vengono fornite alle ruote per correggere la traiettoria. La correzione effettuata nella coroutine *Position* in alcuni casi non risulta sufficiente. È quindi stato realizzato un controllo aggiuntivo che verifichi che il rover non superi mai il target specificato e, nel caso in cui questa condizione non venga più soddisfatta, avvia da capo la procedura di avvicinamento all'obiettivo, andando a eseguire nuovamente tutte le coroutine e permettendo a Curiosity di arrivare sul punto indicato in maniera corretta.
Durante la fase di avvicinamento all'obiettivo, il marker che lo identifica inizia a lampeggiare per segnalare all'utente che il rover si sta spostando verso tale punto. Quest'operazione consiste di fatto in un decremento e incremento del valore del canale alfa del marker, anch'esso eseguito tramite coroutine. Una notifica viene anche inviata all'apposito pannello che registra le attività in esecuzione e visibile nella sezione centrale dello schermo in figura 4.15.

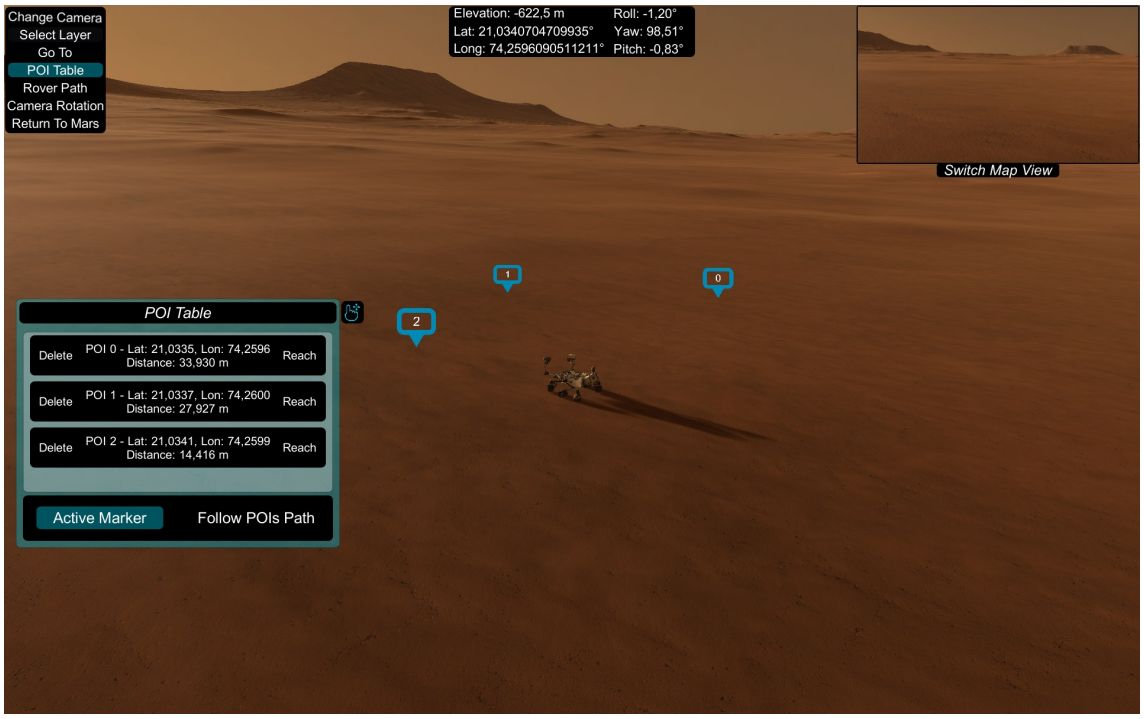


Figura 4.16: Immagine di Mars Navigation che mostra diversi POIs selezionati.

4.5.4 Definizione dei Point(s) Of Interest

Durante l'esplorazione di un'area è importante poter definire alcuni punti rilevanti. È stata realizzata una classe apposita che definisce i cosiddetti POI: *PointOfInterest*; di seguito vengono brevemente descritte le proprietà accessibili tramite questa classe (figura 4.17):

- *Height*: altezza del punto sul terreno identificato dal POI, espressa in metri
- *ID*: intero univoco e sempre crescente che identifica il POI
- *Lat*: latitudine del punto sul terreno identificato dal POI
- *Lon*: longitudine del punto sul terreno identificato dal POI
- *PoiMarkerGameObject*: oggetto utilizzato come marker del POI per visualizzarlo sul terreno
- *PoiMarkerImage*: immagine associata al marker del POI
- *PoiMarkerText*: testo associato al marker del POI
- *PoiTextInfo*: testo contenente le informazioni associate al POI
- *Position*: posizione in coordinate Unity del POI

I POI vengono gestiti utilizzando una variabile globale *Dictionary<int, PointOfInterest> poiList*, che permette di associare ogni oggetto PointOfInterest con il proprio ID, un numero che viene incrementato ogni volta che un POI viene definito. Questa struttura dati permette di mantenere una lista sempre aggiornata dei punti di interesse definiti e di potervi fare accesso utilizzando come chiave l'identificativo del punto, facilitandone l'utilizzo.
Quando l'utente abilita la funzionalità di definizione dei POI, si apre il pannello *POI Table*, visibile in figura 4.16 a sinistra. In questo pannello è presente una ScrollView, in cui si trova l'elenco dei POI definiti, rappresentato da un GameObject contenente:

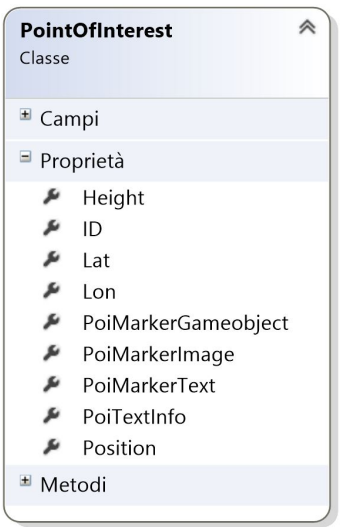


Figura 4.17: Elenco delle proprietà della classe *PointOfInterest*.

un testo con le informazioni rilevanti, un tasto per eliminare il POI e uno per farlo raggiungere dal rover.
Le informazioni da visualizzare sono inserite nella proprietà PoiTextInfo, ciò che viene rappresentato sono i valori di: ID del POI, latitudine, longitudine e distanza dal rover in linea d'aria, aggiornata a ogni spostamento di Curiosity.
Alla definizione di un punto d'interesse vengono eseguite le seguenti operazioni:

1. Creazione e inizializzazione dell'oggetto da inserire nella POI Table
2. Creazione e inizializzazione dell'oggetto PointOfInterest
3. Creazione e inizializzazione del marker del POI che lo identifica sul terreno
4. Inserimento del POI nel dizionario
5. Definizione dei Listener, associati ai tasti per cancellare o raggiungere il POI, tramite funzioni Lambda[3]

Il metodo per identificare il punto in cui inserire il POI sul terreno è lo stesso utilizzato per la definizione del target, facendo un Raycast dalla posizione del cursore del mouse. Il marker che viene inserito comprende anche l'ID che ne consente l'identificazione.
Quando viene selezionato il tasto *Delete* viene rimosso il POI corrispondente a quell'ID dal dizionario e vengono distrutti l'elemento nella POI Table e il marker.
Il tasto *Reach* permette invece di raggiungere il singolo punto d'interesse selezionato, identificandolo come nuovo target del rover; il corrispondente marker inizierà a lampeggiare e, fintanto che il rover sarà in movimento, verranno aggiornate le distanze da tutti i POI presenti sul terreno.

4.5.5 Navigazione lungo un percorso

L'identificazione di una serie di POI permette l'utilizzo di una delle funzionalità più importanti: la definizione di un percorso.
Per la creazione di un percorso si sfruttano i punti d'interesse sopra descritti che, selezionabili tramite l'ID, definiscono i waypoint del cammino da far seguire al rover.
Lo script PropertiesAndCoroutines, oltre alla proprietà Target già vista nel paragrafo 4.5.3, espone una seconda proprietà pubblica: *List<int> Targets*. Tale proprietà permette di avviare la coroutine che gestisce l'avvicinamento del rover a un obiettivo.
È stato quindi creato un Overload[4] di questo metodo, che non riceverà più come parametro un singolo punto da raggiungere, ma una lista di interi, corrispondenti agli ID dei POI definiti in precedenza. Come descritto nell'algoritmo 1 (pseudo-codice) l'overload della coroutine *Movement* permette di raggiungere i POI in successione, seguendo l'ordine specificato.

[3] "Un'espressione lambda è una funzione anonima, tramite la quale è possibile scrivere funzioni locali che possono esser passate come argomento o restituite come valore delle chiamate di funzione. [...] Le espressioni lambda possono fare riferimento alle variabili esterne presenti nell'ambito del metodo che definisce la funzione lambda oppure nell'ambito del tipo che contiene l'espressione lambda. Le variabili acquisite in questo modo vengono archiviate per poter essere utilizzate nell'espressione lambda anche se le variabili diventano esterne all'ambito."[26]

[4] Due membri appartenenti alla stessa tipologia e aventi lo stesso nome, ma con elenco di parametri differenti.

Anche in questo caso viene inviata una notifica che comparirà nel registro delle attività in esecuzione e il marker del POI che il rover sta raggiungendo diventerà intermittente.

Algorithm 1 Principali istruzioni della coroutine Movement.

```
function MOVEMENT(targetIds)
    for all id in targetIds do
        if id in poiList.Keys() then ▷ id è presente tra le chiavi di ricerca dei POI
            targetPoi = poiList.GetValue(id);
            LerpRotation(targetPoi.Position);    ▷ Avvia la coroutine di rotazione
            LerpPosition(targetPoi.Position);    ▷ Avvia la coroutine di posizione
```

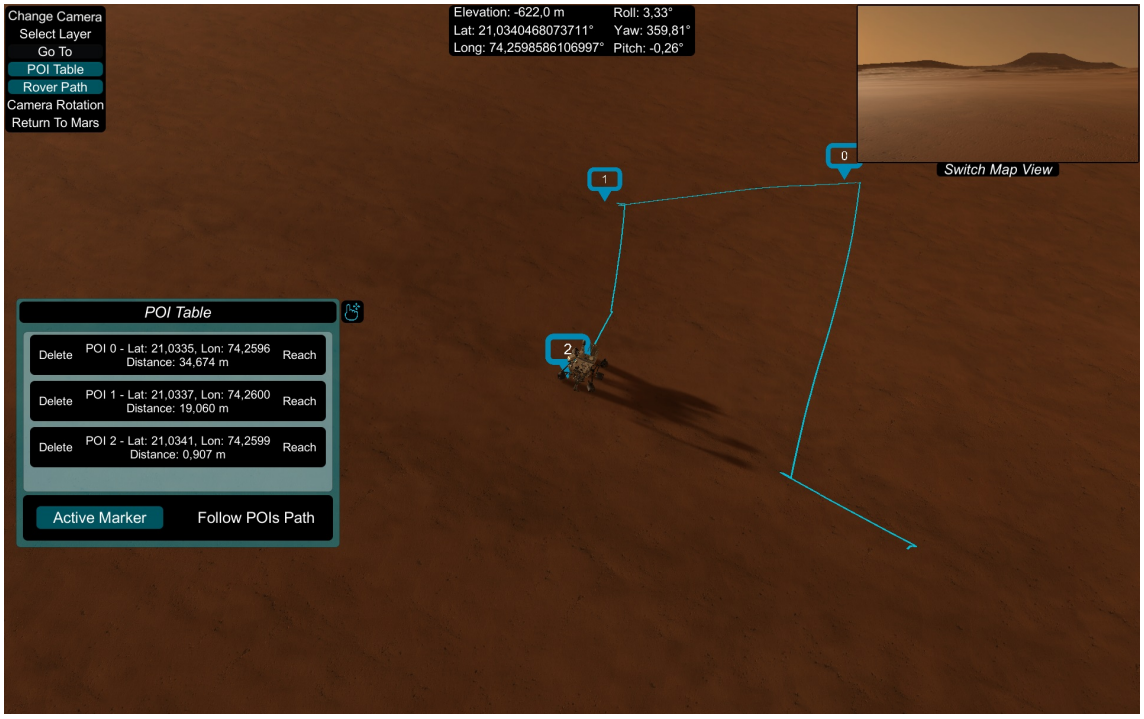


Figura 4.18: Immagine di Mars Navigation che mostra diversi POIs selezionati.

4.6 L'interfaccia utente

La modalità stereoscopica rende complicata la visualizzazione dell'interfaccia utente, poiché questa generalmente non risiede effettivamente nella scena. La UI (User Interface) viene gestita in un Canvas che può esser renderizzato con in tre diverse modalità:

1. *Screen Space - Overlay*: gli elementi della UI vengono renderizzati sullo schermo, davanti alla scena.

2. *Screen Space - Camera*: il canvas viene posizionato a una certa distanza da una specifica camera. Gli elementi della UI sono renderizzati da questa camera, saranno quindi influenzati dalle sue impostazioni, come ad esempio il FOV.

3. *World Space*: il canvas si comporta come ogni altro oggetto della scena.

In applicazioni non per VR, l'interfaccia grafica è spesso renderizzata utilizzando la prima delle tre modalità descritte. Viene definita come *non-diegetic UI* e non esiste effettivamente nel mondo Unity. Questo schema non è però utilizzabile per applicazioni VR, perché non supportato; diventa perciò necessario inserire gli elementi grafici all'interno della scena. Un'interfaccia utente che risiede all'interno della scena è detta *spatial UI* e permette agli occhi dell'utente di mettere a fuoco gli elementi dell'interfaccia grafica.
In Mars Navigation, essendo un'applicazione stereoscopica per cave singolo, è stata utilizzata la seconda modalità di render, che consente di mantenere l'UI sempre davanti alla camera. Il fatto di avere gli elementi della UI all'interno della scena porta a un nuovo problema: la sovrapposizione con altri oggetti. Un qualsiasi oggetto posizionato a una distanza dalla camera inferiore rispetto a quella della UI, la occluderà bloccandone anche l'interazione. La soluzione a questa problematica è data dall'utilizzo di uno shader che permette di renderizzare ogni elemento a cui è applicato in primo piano rispetto al resto.

4.6.1 Elementi della UI riposizionabili

Alcuni elementi della UI hanno dimensioni che possono infastidire con l'attività dell'utente. Problema che aumenta se vengono aperti più pannelli contemporaneamente. È pertanto indispensabile, oltre a poterli nascondere quando risultano non essenziali, poterne riorganizzare la disposizione rispetto allo schermo. I pannelli che possono esser spostati dall'utente sono definiti *draggable* ed è stato associato loro lo script *DraggablePanelController*. In questo script sono definite le funzioni che permettono di gestire tre eventi:

1. *Begin Drag*: evento a cui è associato un metodo che imposta a vero una variabile booleana globale, che indica che viene eseguita l'azione di drag su un elemento della UI. Tramite un controllo su questa variabile è possibile distinguere un drag effettuato per muovere una camera da uno per spostare un pannello grafico.

2. *Drag*: evento che richiama una funzione che riposiziona il pannello sullo schermo in base al movimento del mouse.

3. *End Drag*: evento a cui è associato un metodo che gestisce la fine dell'azione di riposizionamento.

4.6.2 Pannelli grafici

Gli elementi dell'interfaccia utente sono divisi in pannelli:

- *Action Panel*: posizionato in alto a sinistra, contiene tutti i comandi che permettono di accedere alle diverse funzionalità.

- *Info Panel*: pannello contenente le informazioni sulla posizione e rotazione del rover.

- *Interaction Panel*: non è un vero e proprio pannello grafico ma è una empty, padre di tutti i pannelli riposizionabili e che danno accesso ad altre funzionalità.
- *MiniMap Panel*: posizionato nell'angolo in alto a destra, mostra la camera on-board di Curiosity o una vista dall'alto.
- *Activity Panel*: compare nel momento in cui è necessario inviare qualche notifica all'utente, è l'unico pannello riposizionabile esterno alla empty Interaction Panel.

4.7 Effetti grafici

Per aumentare il realismo della simulazione sono stati introdotti diversi effetti grafici.
Tramite dei sistemi particellari associati alle singole ruote di Curiosity, è stato ricreato l'effetto della polvere che si solleva da terra su terreni sabbiosi. I *Particle System* sono componenti di Unity che permettono di gestire un gran numero di parametri e impostazioni. L'effetto dei sistemi particellari sono attivati automaticamente con il movimento delle ruote del rover; La forma e il colore delle particelle sono stati definiti assegnando loro una texture e un materiale simile al terreno, sono state aggiunte altre impostazioni che ne modificano il numero in base alla distanza, la rotazione e la dimensione in base al tempo di vita.
Un altro effetto associato al rover, sono le orme che vengono lasciate sul terreno quando questo si sposta. Per generare tali tracce, sono state create 6 empty, posizionate nel punto di contatto tra le ruote e il terreno. È stato associato loro un componente, chiamato *Trail Renderer*, che consente di generare delle orme con una certa texture e un materiale, creati appositamente utilizzando una maschera che avesse la stessa trama presente sulle ruote di Curiosity. La generazione delle tracce avviene solo al movimento delle empty a cui sono collegati i componenti, la durata complessiva di ogni singola texture che compone la traccia completa è di 50s.
Attraverso uno script chiamato *FadeToSkybox* e l'effetto *Fog* applicato in fase di post-processing, è stato ricreata la nebbia che è possibile vedere in lontananza in diverse immagini scattate dai rover che si trovano su Marte.

Capitolo 5

Valutazione del Lavoro

5.1 Criticità riscontrate

Durante lo sviluppo dell'applicazione, al fine di raggiungere tutti gli obiettivi prefissi, numerosi sono stati i problemi che è stato necessario affrontare. Di seguito vengono analizzati i principali problemi e le criticità che ne sono derivate.

5.1.1 Gestione modelli del terreno

Uno dei problemi per il quale sono state provate diverse soluzioni prima di adottarne una definitiva riguarda la gestione del modello 3D del terreno. L'ottima risoluzione dei file DTM consente la generazione di modelli molto dettagliati, costituiti da un gran numero di vertici. È proprio la quantità di vertici che necessitano di esser elaborati a far crescere la complessità dell'applicazione in termini di calcolo computazionale; file di grandi dimensioni, come i DTM in questione, impattano invece sulla memoria utilizzata.
Ciò che influisce maggiormente nella gestione dei terreni in Unity non sono però i modelli in sé, quanto più i collider che è necessario aggiungere affinché vengano rilevate le collisioni dal motore fisico. I modelli dei terreni sono suddivisi in mesh, che consistono di fatto in strisce di terreno, ognuna delle quali ha collegato un MeshCollider, componente che costruisce il collider seguendo l'esatta forma della mesh. Questo migliora il risultato delle simulazioni fisiche, poiché permette una rilevazione delle collisioni molto più accurata, andando però a incrementare notevolmente la complessità di calcolo.
Sono state valutate diverse soluzioni per riuscire a gestire il caricamento del terreno nel miglior modo:

1. La prima soluzione valutata prevede di caricare, nella fase iniziale, solo un sottoinsieme di MeshCollider, tre per l'esattezza. Supponendo che il rover si trovi sulla striscia n, vengono caricati i collider delle strisce: $(n-1)$, n, $(n+1)$. Quando il rover passa alla striscia successiva, la $(n+1)$, tutto l'insieme di collider viene spostato con esso, andando di fatto a eliminare il MeshCollider della striscia di terreno $(n-1)$ e aggiungendolo alla $(n+2)$. Con questa soluzione il caricamento iniziale risulta decisamente ridotto ma, al momento della creazione del nuovo collider per spostare avanti l'insieme si riscontra un

ritardo nella visualizzazione del flusso video dell'applicazione (*lag*), a causa della complessità del MeshCollider.

2. La seconda soluzione che è stata testata prevede il caricamento di tutti i MeshCollider nella fase iniziale, abilitandone però solamente tre, gli stessi dell'insieme descritto nel punto precedente. Questa soluzione è però risultata la peggiore in quanto il tempo di calcolo della fase iniziale non è diminuito e anche il lag dovuto all'attivazione del collider è rimasto.

3. La terza soluzione prevede il caricamento dell'intero modello, generando e abilitando tutti i MeshCollider nella fase di loading della scena, impiegando un tempo di circa 10s, come visto nel paragrafo 4.1.3.

La scelta è quindi ricaduta sulla terza soluzione che permette di ottenere la simulazione più fluida, pagando il costo del caricamento iniziale di tutti i collider e mantenendoli tutti attivi durante l'esecuzione dell'applicazione.

5.1.2 Perdita di precisione nelle conversioni

Una volta ottenuto il valore di un punto del terreno espresso in coordinate geografiche attraverso le dovute conversioni spiegate nel dettaglio nel capitolo 4.5.1, è stata eseguita un'operazione di verifica dei risultati ottenuti. L'analisi è stata effettuata utilizzando quattro punti di coordinate conosciute dei terreni, corrispondenti ai quattro angoli. Da questi test è stato individuato un errore variabile che influisce sulla precisione con cui vengono identificati i punti. L'errore massimo percentuale per la latitudine risulta dello 0.030%, dello 0.004% per la longitudine.
L'imprecisione nel calcolo può derivare dalla proiezione utilizzata per la conversione[27] nel sistema di riferimento geografico, in quanto viene fatta un'approssimazione della geometria dello sferoide di Marte.

5.1.3 Interfaccia utente stereoscopica

Uno dei problemi a cui si va incontro quando si ha a che fare con applicazioni stereoscopiche 3D riguarda la visualizzazione degli oggetti che si trovano ai lati dello schermo. Oggetti che dovrebbero esser posizionati davanti al piano dello schermo ma che vengono tagliati poiché si trovano agli estremi destro o sinistro, fanno perdere la percezione del 3D causando anche dei disturbi nell'utente. Questo problema può esser generalizzato nel caso dell'interfaccia utente che, come spiegato nel capitolo 4.6, viene renderizzata davanti a qualsiasi oggetto della scena per esser sempre visibile e accessibile all'utente. Gli elementi della UI sono posizionati a una distanza fissa dalla camera, se gli oggetti che dovrebbero occluderli, poiché a una distanza inferiore, vengono invece renderizzati dietro, il cervello riceve degli stimoli opposti che causano fastidio nell'utente. Questa situazione si verifica solamente quando la distanza del rover dalla camera è molto piccola, per limitare quest'effetto sgradevole, dalla prospettiva della camera on-board sono state nascosti alcuni pannelli e elementi dell'interfaccia considerati meno utili in questa modalità.

5.2 Test di valutazione soggettivi

L'applicazione, una volta ultimata, è stata fatta provare a un gruppo di dodici persone per avere un riscontro sull'usabilità nel suo complesso e per ottenere una valutazione delle funzionalità realizzate; l'età dei partecipanti al test varia da 18 a 44 anni. Per rendere più attendibile i risultati dei test, tra gli utenti selezionati sono presenti sia esperti del settore, i quali hanno già avuto a che fare con tecnologie simili o conoscono in modo dettagliato i temi che vengono trattati, sia persone con competenze molto differenti, alle prime esperienze con la realtà virtuale e le tecniche di pianificazione. Tra i soggetti sono inoltre presenti portatori di occhiali e non, per valutare in modo più completo i possibili disturbi che la stereoscopia può portare.
A seguito dell'utilizzo dell'applicazione è stato chiesto agli utenti di compilare un questionario[28], per poter effettuare un'analisi statistica dei risultati ottenuti.
Il questionario è composto da due parti: una da compilare precedentemente all'utilizzo dell'applicazione, in cui, oltre a domande di carattere generale (es. età, sesso, ecc.), i soggetti sono interrogati sulla propria esperienza con il mondo della realtà virtuale; l'altra da compilare successivamente, in cui viene valutata effettivamente l'applicazione Mars Navigation nel suo complesso. Nei grafici che seguono è possibile osservare, per ogni caratteristica presa in considerazione, la valutazione in percentuale attribuita dagli utenti, tenendo conto che la scala di valutazione per le domande riguardanti l'usabilità e l'utilità va da 1 a 5 e ha il seguente significato:

- 1 - Scadente - Inutile
- 2 - Accettabile - Poco utile
- 3 - Abbastanza buona - Abbastanza utile
- 4 - Buona - Utile
- 5 - Eccellente - Molto utile

Nonostante la metà degli utenti abbia dichiarato di non aver mai fatto uso di alcun dispositivo di realtà virtuale o stereoscopico, nessuno ha dimostrato particolari difficoltà nell'utilizzo dell'applicazione. Come è possibile vedere nel grafico in figura

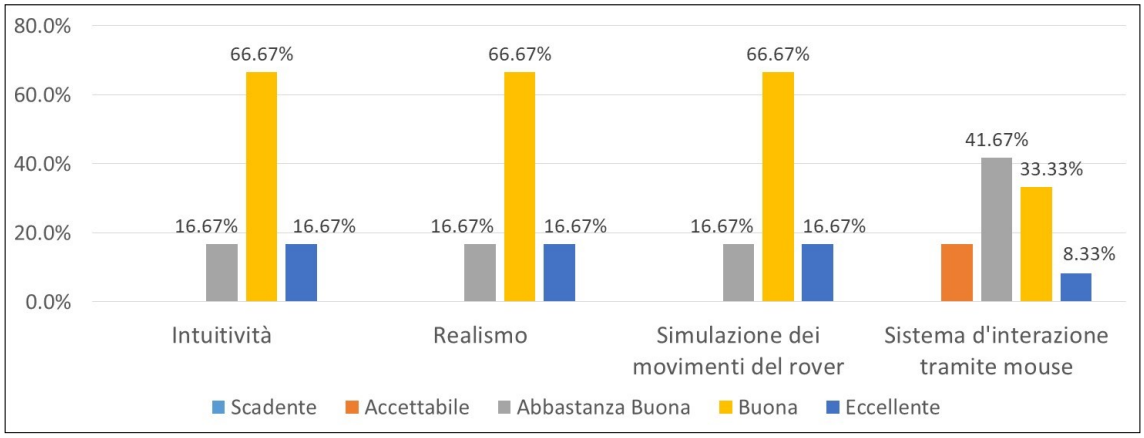


Figura 5.1: Grafico riassuntivo per la valutazione dell'intuitività, del realismo, della simulazione dei movimenti del rover e del sistema d'interazione di Mars Navigation.

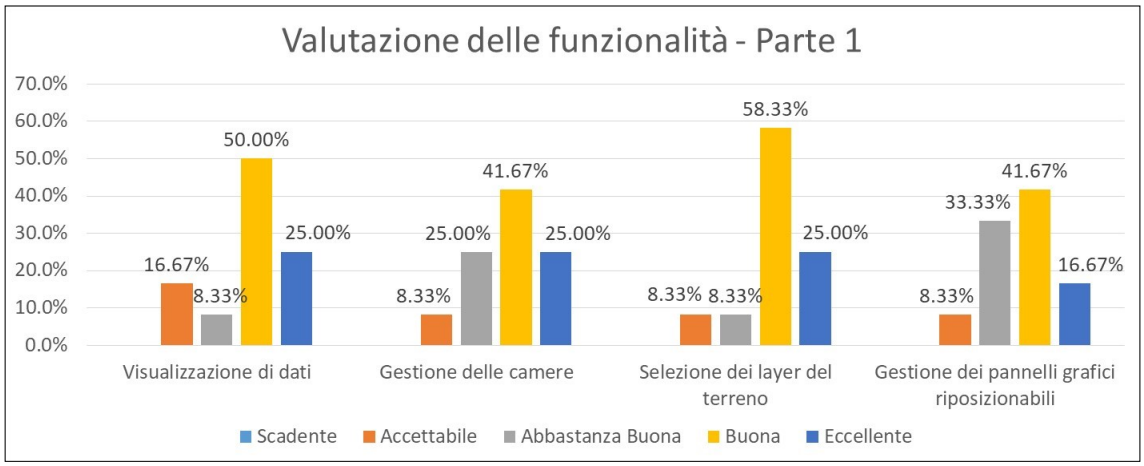


Figura 5.2: Grafico riassuntivo per la valutazione delle funzionalità di Mars Navigation (parte 1).

5.1 infatti, oltre il 66% ha valutato buona l'intuitività di Mars Navigation.
Nello stesso grafico si può vedere come anche il realismo e la simulazione dei movimenti del rover sono stati valutati in modo positivo.
Ciò che è risultato meno efficacie è invece il metodo d'interazione tramite mouse, che viene ritenuto abbastanza buono dalla maggior parte dei soggetti, ma che la quasi totalità giudica migliorabile come è stato inserito in un'apposita sezione di suggerimenti.
Successivamente sono state valutate le impressioni degli utenti sulle funzionalità realizzate, per ognuna delle quali era possibile esprimere un giudizio di usabilità. Nei due grafici riportati in figura 5.2 e 5.3 si possono osservare i risultati ottenuti.
La visualizzazione dei dati, la gestione delle camere e la selezione dei layer sono state valutate per lo più buone; la gestione dei pannelli grafici riposizionabili ha ottenuto risultati molto simili, mentre alla funzionalità di selezione del singolo target sono state date valutazioni più discordanti: alcune molto positive, mentre altri utenti l'hanno valutata solamente accettabile.
Le funzionalità che hanno ricevuto le migliori valutazioni sono quelle più complicate

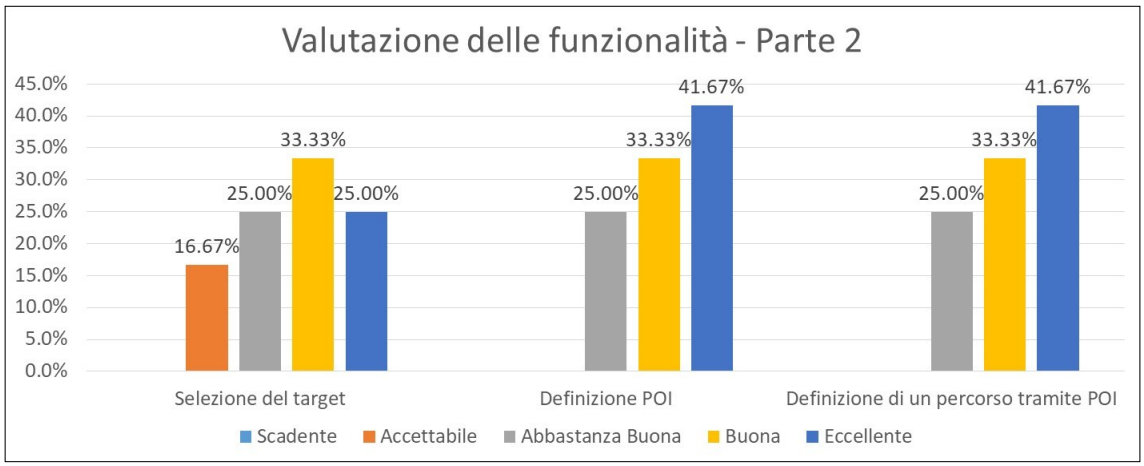


Figura 5.3: Grafico riassuntivo per la valutazione delle funzionalità di Mars Navigation (parte 2).

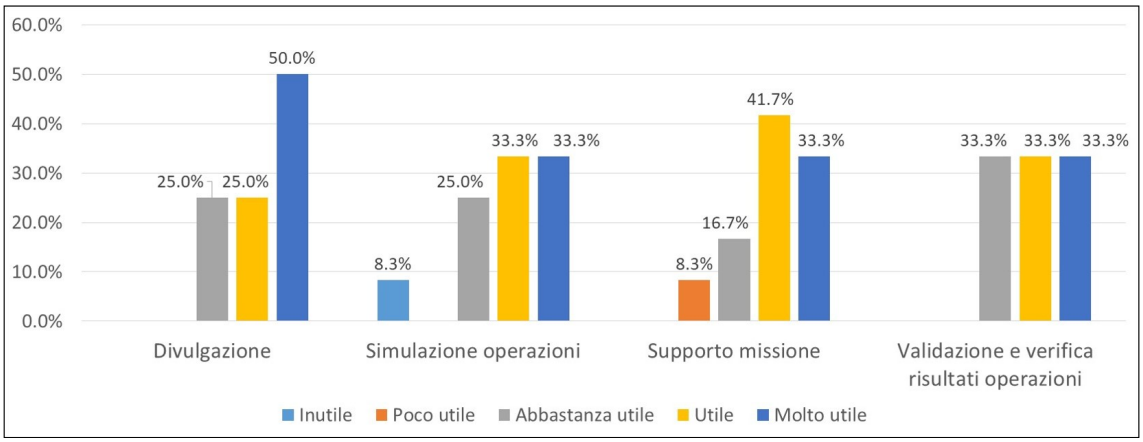


Figura 5.4: Grafico riassuntivo per la valutazione dei possibili ambiti di utilizzo di Mars Navigation (parte 2).

dal punto di vista del funzionamento e della realizzazione: la definizione dei POI e la definizione di un percorso da far seguire al rover. In figura 5.3 si può notare come più del 41% degli utenti ha valutato le due funzionalità come eccellenti.
Le ultime domande del questionario hanno l'intento di valutare l'utilità che gli utenti ritengono l'applicazione possa avere nell'immediato o in futuro, analizzando anche la validità di alcuni possibili sviluppi indicati. Alcuni suggerimenti sono più generici e riguardano la gestione delle camere, il sistema d'interazione o l'interazione con la finestra della mappa, altri invece sono più dettagliati descrivono alcune funzionalità o informazioni che potrebbe essere utile aggiungere.
L'applicazione, secondo i risultati del test effettuato visibili in figura 5.4, risulta molto efficacie se utilizzata a scopo divulgativo; indicazioni positive sono state ottenute anche per il supporto missioni e per un suo utilizzo volto a fornire una verifica e validazione dei risultati di operazione di vario genere. L'unico riscontro negativo, in termini di utilità (8.3% delle risposte), riguarda l'uso di Mars Navigation per la simulazione di attività di vario genere.
Le risposte fornite in merito a eventuali funzionalità aggiuntive sono osservabili nei grafici in figura 5.5 e 5.6, in cui si evidenziano indicazioni maggiormente favorevoli per aspetti più tecnici come ad esempio: l'analisi di ostacoli, per la quale il 58.3% degli utenti ha indicato la massima utilità, la definizione di Stay-Out-Areas o l'analisi di dati scientifici. Ritenute comunque utili o abbastanza utili le altre funzionalità.
Un altro aspetto che è stato valutato è il disturbo che la stereoscopia 3D può portare. Gli utenti hanno risposto in modo molto positivo: una piccola percentuale ha riscontrato un leggero affaticamento della vista, mentre nessuno ha avuto problemi più seri come di giramento di testa, nausea o altri disturbi.

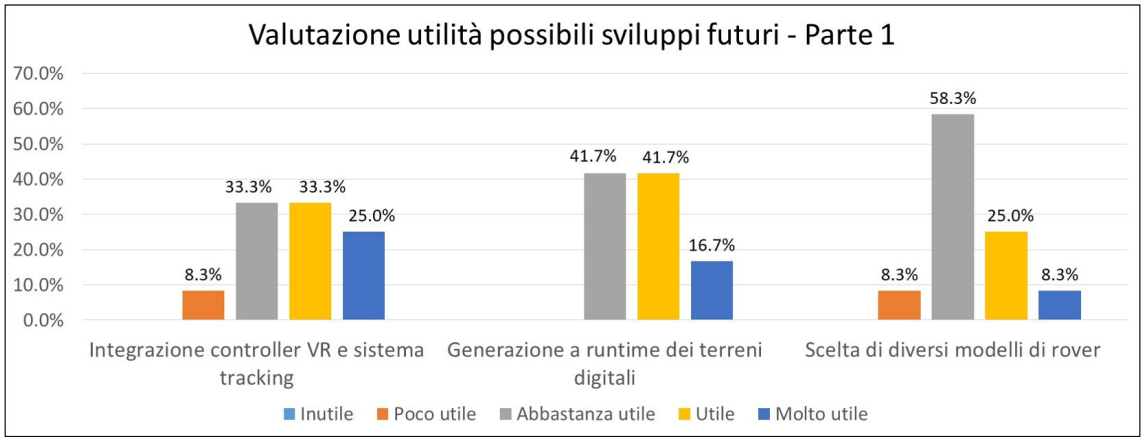


Figura 5.5: Grafico riassuntivo per la valutazione delle funzionalità di Mars Navigation (parte 1), in cui sono presenti le valutazioni riguardanti: l'integrazione di controller VR, mouse 3D e sistema di tracking; la generazione a runtime dei modelli dei terreni; la scelta di diversi modelli di rover.

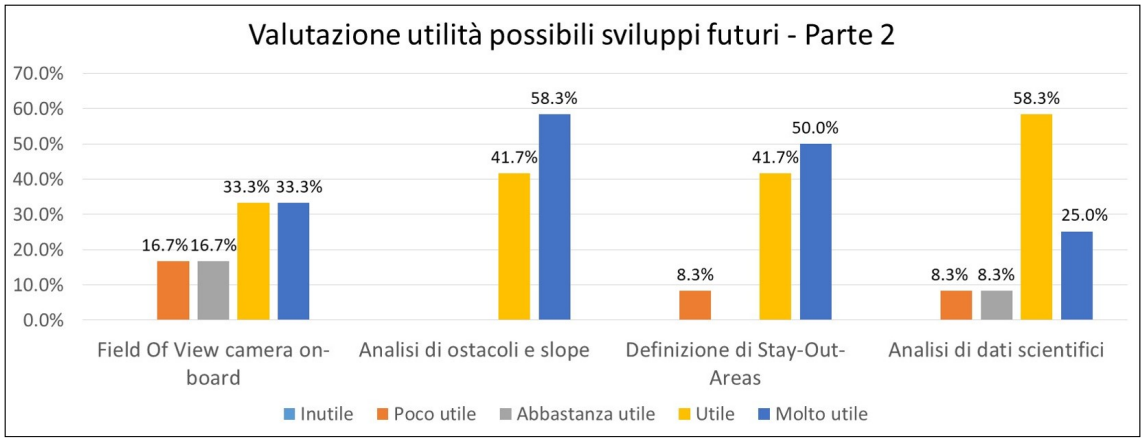


Figura 5.6: Grafico riassuntivo per la valutazione delle funzionalità di Mars Navigation (parte 2), in cui sono presenti le valutazioni riguardanti: la visualizzazione del FOV delle camere on-board; l'analisi di ostacoli e slope; la definizione di aree non raggiungibili (Stay-Out-Areas); l'analisi di dati scientifici (dati del sottosuolo).

Capitolo 6

Conclusioni

Il progetto di tesi presentato ha portato alla realizzazione di un'applicazione in realtà virtuale in grado di fornire un supporto per l'esplorazione terrena, integrando le più rilevanti funzionalità utili per un'attività di pianificazione, e in grado di presentarne una simulazione realistica.
Dal lavoro di tesi svolto e dalle analisi effettuate è emerso che gli obiettivi prefissati sono stati raggiunti e sono stati ottenuti risultati soddisfacenti, come dimostrato dai test effettuati.
Aspetto rilevante del progetto è costituito dalla versatilità ed espandibilità che questo presenta, tali da garantire una miglior integrazione di funzionalità aggiuntive e di dati differenti.
Possibili sviluppi futuri possono portare a miglioramenti che aumenterebbero il campo d'applicazione. Facendo riferimento alla definizione di percorsi, diversi sono i punti che è possibile sviluppare per rendere più completa la pianificazione, ad esempio:

- l'aggiunta di ulteriori informazioni (es. distanze intermedie tra POI consecutivi, aggiunta della direzione rispetto al rover, ecc.);
- l'aggiunta di un parametro che permetta di specificare l'orientamento desiderato del rover al punto d'arrivo;
- la possibilità di modificare il percorso definito, riposizionando i waypoint;
- l'utilizzo algoritmi specifici per il calcolo di cammini minimi.

Uno dei problemi riscontrati durante i test riguarda il metodo d'interazione tramite mouse e tastiera, il quale è risultato essere sicuramente migliorabile. L'utilizzo dell'applicazione in modalità stereoscopica, con un CAVE singolo, la rende sicuramente adatta all'utilizzo di controller VR o mouse 3D tramite i quali è possibile non solo migliorare il sistema d'interazione, ma rendere anche più efficienti le funzioni di pianificazione. Aggiungendo un sistema di tracciamento è possibile aumentare il senso d'immersione dell'utente, migliorandone l'esperienza virtuale.
L'introduzione di una nuova camera, che permetta di effettuare una navigazione svincolata dal rover, è sicuramente un perfezionamento che garantirebbe una maggior libertà di movimento, aiutando l'utente in fase di pianificazione e permettendogli di condurre analisi più dettagliate sul terreno.
Come è emerso dai test effettuati, una delle funzionalità aggiuntive, che risulterebbe più utile inserire nel progetto, riguarda l'analisi di ostacoli presenti sul terreno.

Tramite l'identificazione di rocce o lo studio dei pendii, si sarebbe infatti in grado di rilevare zone non accessibili dal rover, definite Stay-Out-Areas.
Una delle più importanti modifiche che consentirebbe di fare un grande passo avanti sull'aspetto dell'espandibilità e versatilità dell'applicazione riguarda la gestione dei modelli terreni. Questi infatti, vengono caricati in fase d'esecuzione dell'applicazione, ma generati precedentemente attraverso un software esterno. L'integrazione del processo di creazione del modello 3D nell'applicazione ne consentirebbe una miglior gestione, permettendo di definire ciò che in Unity è noto con il nome di LOD (Level Of Detail). L'utilizzo di questa tecnica permette di renderizzare un oggetto della scena con livelli di dettaglio differenti in base alla distanza che questa ha dalla camera, riducendo di fatto il numero di triangoli che vengono renderizzati. Questa tecnica permette di ottimizzare in maniera importante il calcolo computazionale, specialmente se il livello di dettaglio degli oggetti è molto elevato come nel caso dei modelli dei terreni. Per realizzarla è però indispensabile costruire il terreno in modo tale che non sia identificato come un singolo oggetto, ma come la composizione di diverse fasce, visualizzate con risoluzioni differenti in base alla distanza dal rover o dalla camera.
Con lo studio effettuato viene messo in evidenza come la realtà virtuale sia una tecnologia che va affermandosi sempre più in ambiti scientifici e possa portare a numerosi vantaggi. È necessario continuare a investire in tempo e risorse per portare miglioramenti negli aspetti ancora carenti, ma i risultati raggiunti fin'ora si dimostrano rilevanti e ne delineano un grande potenziale.

Bibliografia

[1] URL: https://www.reportlinker.com/p05308313/Virtual-Reality-Technologies-Global-Market-to.html (visitato il 09/02/2018).

[2] Clark J. Olson et al. «Visual Terrain Mapping for Mars Exploration». In: (gen. 2004).

[3] C. Stoker et al. *Visualizing Mars using virtual reality: A state of the art mapping tool used on Mars pathfinder*. 1999. URL: https://mars.jpl.nasa.gov/mgs/sci/fifthconf99/6037.pdf (visitato il 29/01/2018).

[4] Suya You e Charles. K. Thompson. «Mobile Collaborative Mixed Reality for Supporting Scientific Inquiry and Visualization of Earth Science Data». In: (mar. 2017).

[5] K. Bladin et al. «Globe Browsing: Contextualized Spatio-Temporal Planetary Surface Visualization». In: *IEEE Transactions on Visualization and Computer Graphics* (gen. 2018).

[6] R. W. Zurek e S. E. Smrekar. «An overview of the Mars Reconnaissance Orbiter (MRO) science mission». In: *Journal of Geophysical Research: Planets* (2007).

[7] Rebecca Bennet, David J. Zielinski e Regis Kopper. «Comparison of interactive environments for the archaeological exploration of 3D landscape data». In: (nov. 2014).

[8] Chang-qing Yu, He-hua Ju e Yang Gao. «3D virtual reality simulator for planetary rover operation and testing». In: (mag. 2009).

[9] F Ropero et al. «A Virtual Reality Mission Planner for Mars Rovers». In: (set. 2017).

[10] L. Edwards et al. «Photo-realistic Terrain Modeling and Visualization for Mars Exploration Rover Science Operations». In: (ott. 2005).

[11] Jue Wang e Keith J. Bennett. «A virtual reality study on Santa Maria Crater on Mars». In: (mar. 2013).

[12] Andrew Good. *Take a Walk on Mars – in Your Own Living Room*. A cura di Martin Perez. 2017. URL: https://www.nasa.gov/feature/jpl/take-a-walk-on-mars-in-your-own-living-room (visitato il 29/01/2018).

[13] Guy Webster e Veronica McGregor. *NASA, Microsoft Collaboration Will Allow Scientists to 'Work on Mars'*. A cura di Dwayne Brown. 2015. URL: https://www.jpl.nasa.gov/news/news.php?feature=4451 (visitato il 29/01/2018).

[14] URL: http://esamultimedia.esa.int/multimedia/virtual-tour-iss/ (visitato il 09/02/2018).

[15] W. Sherman e A.B. Craig. *"Understanding Virtual Reality: Interface, Application, and Design"*. Elsevier Inc, Morgan Kauffman Pub, 2002, pp. 6–13.

[16] URL: https://www.blender.org/ (visitato il 14/02/2018).

[17] URL: https://wiki.blender.org/index.php/Extensions:2.6/Py/Scripts/Import-Export/HiRISE_DTM_from_PDS_IMG (visitato il 14/02/2018).

[18] URL: https://unity3d.com/ (visitato il 14/02/2018).

[19] URL: http://www.nvidia.it/object/nvidia-physx-it.html (visitato il 27/03/2018).

[20] URL: http://www.gdal.org/ (visitato il 14/02/2018).

[21] URL: http://www.barco.com/en/product/rlm-w14 (visitato il 14/02/2018).

[22] URL: http://www.celestiamotherlode.net/catalog/mars.php (visitato il 15/02/2018).

[23] URL: https://www.uahirise.org/dtm/ (visitato il 15/02/2018).

[24] URL: https://assetstore.unity.com/packages/vfx/shaders/grit-buildup-shader-96981 (visitato il 15/02/2018).

[25] URL: https://nasa3d.arc.nasa.gov/detail/curiosity-dirty (visitato il 15/02/2018).

[26] URL: https://docs.microsoft.com/it-it/dotnet/csharp/programming-guide/statements-expressions-operators/lambda-expressions (visitato il 21/03/2018).

[27] URL: http://spatialreference.org/ref/sr-org/iau200049902mars-2000-planetocentric/ (visitato il 26/03/2018).

[28] URL: https://docs.google.com/forms/d/e/1FAIpQLScifcNEBj2BVpwvqXhp7RxebpSZ-yCPWHe5iLMhi89MpuaXEQ/viewform?usp=sf_link (visitato il 28/03/2018).

Ringraziamenti

Giunto alla conclusione del percorso di studi intrapreso, desidero ricordare tutti coloro che mi hanno supportato e aiutato nel raggiungimento di questo traguardo.
Vorrei ringraziare innanzitutto il Prof. Andrea Sanna per avermi dato la possibilità di svolgere questo progetto in collaborazione con Altec SpA, sottolineando la grande disponibilità nel fornirmi i preziosi consigli e le osservazioni che mi hanno aiutato nella stesura della tesi.
Un ringraziamento particolare va a Eugenio Topa e Carlo Vizzi, per avermi fatto da guida durante lo sviluppo del progetto, fornendomi tutte le informazioni e gli strumenti necessari a una sua buona riuscita, ma soprattutto per l'infinita disponibilità e pazienza dimostrata.
Un grazie a tutte le persone conosciute in Altec, in particolare ai colleghi tesisti che non mi hanno mai fatto mancare il loro supporto, ma soprattutto per aver condiviso insieme i momenti più belli e divertenti di questo percorso.
Grazie anche a tutti i colleghi di università con cui ho affrontato gli anni di studio, per ogni momento di difficoltà o gioia passato insieme.
Agli amici di una vita e ai compagni Mooskins va il mio ringraziamento per avermi fatto distrarre e sfogare nei momenti di crisi, così da superarli serenamente.
Un grazie speciale a Sara, che forse più di tutti mi ha seguito in questo percorso, incoraggiandomi e facendomi rigare dritto quando era necessario, in particolare per essermi stata vicina in ogni momento e avermi sopportato durante tutti questi anni.
Infine, il grazie più grande va a tutta la mia famiglia, specialmente a mia mamma Beatrice, che mi ha fornito tutti i mezzi per arrivare a questo risultato, altrimenti impossibile da raggiungere, e che spero di rendere orgogliosa. In modo particolare il mio pensiero e un grande abbraccio va ai miei nonni, Rita e Delio, che, nonostante probabilmente ancora non sappiano cosa riguardi il corso di laurea appena concluso, sono state le persone che più di tutte mi sono state vicine, informandosi tutti i giorni sui progressi fatti a "scuola". A loro vorrei dedicare questo lavoro.