ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

CAMPUS DI CESENA

SCUOLA DI SCIENZE

Corso di Laurea in Ingegneria e Scienze Informatiche

Sinergia tra Blender e Unity nella realizzazione di un Mondo Aumentato

Relazione finale in
Computer Graphics

Relatore:
Damiana Lazzaro

Presentata da:
Giulia Capacci

Sessione III
Anno Accademico 2016-2017

Dedico questa tesi
ai miei genitori, che mi insegnano qualcosa di nuovo ogni giorno,
a Pierluigi, la metà della mia anima,
a me stessa, che nonostante tutto riesco sempre a dimostrare di poter fare
qualunque cosa.

Indice

Elenco delle figure

Introduzione

L'obiettivo di questa tesi, come si evince dal titolo, è la realizzazione di un'applicazione per Android volta all'arricchimento della realtà con elementi tridimensionali virtuali creati al calcolatore al fine di simulare un "Mondo Aumentato". Nello specifico, questo Mondo punta a ricreare la sala di un *Museo Oceanico*, ricco di quadri - elementi tangibili della realtà - e ricco di oggetti sintetici - elementi dello strato virtuale con cui è possibile interagire.

Concretamente, la simulazione della stanza del "Museo Aumentato" è stata effettuata all'interno di un ambiente ampio, luminoso e dai toni un po' freddi in cui sono stati posizionati i *"marker"*, immagini dalle caratteristiche particolari in grado di mostrare, grazie all'impiego di dispositivi tecnologici come tablet e smartphone, modelli 3D virtuali di oggetti che non sono presenti nella scena fisica. La realtà, quindi, è stata "aumentata" ed arricchita per mezzo di questi semplici pezzi di carta stampata. Alle pareti, inoltre, a richiamo delle opere artistiche, sono stati appesi i quadri contenenti le scene stampate prodotte.

La sinergia di questo progetto, ovvero l'integrazione di più agenti che operano insieme per produrre un risultato non ottenibile singolarmente, consiste nella cooperazione di tre strumenti di carattere informatico, ossia *Blender* - il motore grafico con cui sono stati creati i modelli e le scene - , *Unity* - la piattaforma di sviluppo adibita all'unione degli elementi virtuali prodotti - e *Vuforia* - il supporto volto all'implementazione dei metodi e delle tecniche della Realtà Aumentata.

La Realtà Aumentata, quindi, mira ad amplificare le esperienze umane arricchendole con elementi sintetici ed interattivi che risiedono in quella che sembra

essere una dimensione parallela ma che in realtà è in grado di vivere e persistere come strato a se stante.

L'ambizioso progetto descritto in questo elaborato è nato come conseguenza della collaborazione tra l'Università e il Museo Marittimo di Cattolica (Museo Della Regina) e l'amore per le rappresentazioni artistiche di un pittore russo del XIX secolo. L'idea ha cominciato a prendere forma già verso la fine dell'anno 2017 e si è concretizzata appieno nel marzo 2018.

La tesi è così organizzata:

- **capitolo 1**: introduce il concetto cardine della tesi, ovvero la Realtà Aumentata;
- **capitolo 2**: illustra Blender in quanto motore grafico;
- **capitolo 3**: presenta Unity come piattaforma di sviluppo dell'applicativo;
- **capitolo 4**: spiega la tecnologia di Vuforia volta all'inserimento della Realtà Aumentata;
- **capitolo 5**: riguarda l'analisi delle fonti d'ispirazione per il progetto e la realizzazione dettagliata del lavoro;
- **capitolo 6**: evidenzia la sinergia di tutti gli elementi sopraccitati.

Capitolo 1

Realtà Aumentata

Questo primo capitolo è volto alla spiegazione dettagliata dell'argomento principale affrontato nel progetto di tesi, il cardine attorno cui ruotano tutti gli altri concetti: la Realtà Aumentata.

1.1 Definizione

Con il termine "Realtà Aumentata" si intende l'arricchimento del mondo fisico e della percezione sensoriale umana tramite informazioni quali immagini, suoni, animazioni e feedback che vengono manipolati al calcolatore al fine di offrire esperienze interattive che vanno al di là dello scibile quotidiano. Queste informazioni si sovrappongono al mondo reale rendendolo digitalmente interattivo. L'immersione dell'utente in quello che può essere definito *"Mondo Aumentato"* è resa possibile grazie all'impiego di dispositivi tecnologici quali smartphone, occhiali o tablet[1].

I primi esempi di realtà aumentata furono introdotti nel settore dell'aeronautica militare sotto forma di *Head-Up Display (HUD)*[2] - un visore a sovrimpressione - sugli aerei da combattimento per mostrare ai piloti informazioni sul volo come, per esempio, la quota e velocità del velivolo o la distanza dall'obiettivo, permettendo così di non distogliere sguardo e concentrazione. Questa tecnologia fu adottata anche dall'aviazione civile e marittima. Successivamente, nel 2009 Apple decise di introdurre nei suoi Iphone 3GS un software chiamato

Layar, ovvero un reality browser che, grazie ai dati su longitudine, latitudine ed accelerometro, consentiva di inquadrare con la fotocamera un particolare edificio o monumento per riceverne informazioni come, per esempio, il nome o la storia, oppure informazioni sui punti di interesse presenti nelle vicinanze.

La Realtà Aumentata - definita d'ora in avanti con l'acronimo AR - si differenzia dalla Realtà Virtuale (VR) in quanto quest'ultima prevede che l'intero ambiente sia costituito da elementi fittizi, sintetici che distaccano l'utente da ciò che lo circonda[3].

1.1.1 Tipologie

Esistono diverse tipologie di implementazione per l'AR[4], ma tutte seguono gli stessi principi e le stesse regole; un'applicazione di questo genere può quindi utilizzarle tutte oppure solo una:

- **Marker** : vengono utilizzate delle immagini come ad esempio QRCode, numeri seriali, pattern o addirittura stampe che, una volta registrate e memorizzate digitalmente, diventano le sorgenti degli elementi aumentati con cui l'utente può interagire; occorre quindi inquadrarle con il dispositivo per poter avviare l'esperienza.

- **Markerless** : non si sfruttano più le immagini digitalizzate, ma bensì le informazioni sulla posizione dell'utente, quindi latitudine, longitudine, altitudine, giroscopio ed accelerometro sono chiamate in causa al fine di aumentare la realtà.

- **Layered** : grazie alla tecnologia *"object recognition"* è possibile, in una prima fase, identificare lo spazio fisico e in una seconda apporvi gli elementi sintetici. Oggi è una delle tipologie più popolari: molte applicazioni permettono infatti di provare capi di abbigliamento virtuali, oppure di verificare se un pezzo d'arredamento è concorde al design della stanza.

- **Projected** : si proietta della luce artificiale su una superficie di grandi dimensioni, come ad esempio sulla facciata di un palazzo, per simulare una scena del tutto differente.

1.2 Tecnologie

Esiste una grande varietà di dispositivi fisici in grado di eseguire applicazioni di tipo AR:

- **Hardware** : processore, display, sensori, dispositivi di input,fotocamera e sensori MEMS (sistemi microelettromeccanici) come accelerometro, GPS e bussola.

- **HMD** : Head-Mounted Display (HMD), dispositivo indossabile già citato in precedenza che assomiglia ad un casco e che mette le immagini sia del mondo fisico che degli oggetti virtuali sul campo visivo dell'utente. I più moderni utilizzano sensori di monitoraggio che permettono al sistema di allineare le informazioni virtuali al mondo fisico e adattarsi di conseguenza ai movimenti della testa dell'utente. L'HMD è in grado di fornire agli utenti esperienze profonde ed immersive anche grazie alla sua azione sinergica con le *gesture*.

- **Occhiali** : esistono versioni che includono occhiali da vista che impiegano telecamere per scannerizzare il mondo reale e quindi mostrare gli elementi attraverso di essi ed esistono versioni in cui le immagini vengono proiettate o riflesse sulle lenti. GoogleGlass[5] e HoloLens[6] sono i nomi più conosciuti.

- **Lenti a contatto** : si tratta di vere e proprie lenti a contatto bioniche che aumentano la realtà incorporando gli elementi virtuali nella lente stessa, includendo addirittura circuiti integrati, LED ed antenne per la comunicazione wireless. La prima lente a contatto di questo tipo è stata segnalata nel 1999. Una versione delle lenti a contatto in fase di sviluppo

per l' esercito americano, funziona insieme agli smartglasses, consentendo ai soldati di concentrarsi sulle immagini AR vicine agli occhi e sugli oggetti lontani reali allo stesso tempo[7].

- **Dispositivi portatili** : si tratta di smartphone e tablet, ormai diffusi in qualunque casa/struttura.

1.3 Algoritmi impiegati

Per poter realizzare applicazioni in AR sono necessari algoritmi in grado di integrare realisticamente gli elementi virtuali con il mondo reale: il software deve derivare le coordinate del mondo reale, indipendentemente dalla fotocamera e dalle immagini acquisite. Questo processo è chiamato *Image Registration* e utilizza diversi metodi di Computer Vision principalmente legati al monitoraggio video, ereditati in parte dalla Visual Odometry.[8] Vengono eseguite due fasi: la prima consiste nel rilevare punti di interesse, marker o un flusso ottico nelle immagini della telecamera sfruttando funzioni di rilevamento come edge detection, blob detection, thresholding e altri metodi di elaborazione delle immagini; la seconda fase consiste nel ripristinare il sistema di coordinate del mondo reale dai dati ottenuti precedentemente. Altri metodi si basano sulla geometria degli oggetti conosciuti che sono già presenti nella scena. I metodi matematici impiegati comprendono la geometria proiettiva (epipolare), l'algebra geometrica, la rappresentazione di rotazione con mappe esponenziali, i filtri di Kalman e particelle, l'ottimizzazione non lineare, ecc. Il Linguaggio di Markup per la Realtà Aumentata (Augmented Reality Markup Language, ARML) è uno standard di dati sviluppato nell'ambito del Consorzio Geospaziale Aperto (Open Geospatial Consortium, OGC), che sfrutta la grammatica XML per descrivere la posizione e l'aspetto di oggetti virtuali nella scena, nonché le associazioni ECMAScript per consentire accesso dinamico alle proprietà degli oggetti virtuali. Per consentire un rapido sviluppo delle applicazioni AR, sono stati sviluppati alcuni kit di sviluppo software (SDK) come Vuforia, ARToolKit, Catchoom CraftAR, Mobinett AR, Wikitude, Blippar, Layar, Meta e ARLab.

1.4 Possibili applicazioni

Le applicazioni della Realtà Aumentata sono innumerevoli e spaziano dal campo medico a quello artistico.

- **Commercio** : l'AR è utilizzata per integrare la stampa e il marketing video. Il materiale di marketing stampato può essere progettato con alcune immagini che, una volta scansionate da un dispositivo abilitato, attivano un video promozionale del prodotto. Una grande differenza tra la realtà aumentata e il riconoscimento dell'immagine diretta è la possibilità di sovrapporre contemporaneamente più contenuti multimediali nella schermata di visualizzazione, come ad esempio i pulsanti di condivisione tramite social media L'AR può, quindi, migliorare le anteprime dei prodotti, può consentire a un cliente di vedere ciò che è all'interno dell'imballaggio di un bene ancora prima di aprirlo, oppure può aiutare nella selezione di prodotti provenienti da un catalogo.

- **Educazione** : utilizzata per integrare l'educazione standard, testo e grafica. Video e audio sono sovrapposti all'ambiente in tempo reale andando ad arricchire i libri di testo con marker o trigger che, una volta scansionati da un dispositivo, producono informazioni supplementari in un formato multimediale, come ad esempio delle simulazioni di eventi storici oppure ancora di visualizzare ed interagire con la struttura spaziale di una molecola.

- **Videogiochi** : la videogame industry ha abbracciato la tecnologia AR sviluppando giochi per ambienti interni aumentati, come AR hockey, Titans of Space, combattimenti collaborativi contro nemici virtuali, giochi da tavolo, biliardo, ecc. La realtà aumentata permette ai giocatori di sperimentare il gioco digitale in un ambiente reale. Le aziende e le piattaforme come Niantic e LyteShot sono emerse come maggiori creatori di gioco di realtà, la prima in particolare per la creazione della celebre applicazione *Pokemon Go*[9].

- **Arte** : anche nelle arti visive consente agli oggetti o ai luoghi di innescare esperienze artistiche e diverse interpretazioni della realtà. Oggi sono sempre più numerosi i musei interessati alla realizzazione di esperienze aumentate, questa stessa tesi ne vuole essere una chiara esemplificazione.

- **Sicurezza** : già nel 2009 due articoli della rivista *Emergency Management* hanno discusso il potere della tecnologia per la gestione delle emergenze facendo risaltare l'enorme aiuto che questi sistemi potrebbero portare, ad esempio nel rintracciamento di un escursionista feritosi o smarritosi fra le montagne.

- **Medicina** : sin dal 2005 è stato utilizzato un dispositivo chiamato *"near-infrared vein finder"* che registra le vene sottocutanee, le elabora e ne proietta l'immagine sulla pelle. La Realtà Aumentata fornisce ai medici chirurghi dati di monitoraggio del paziente, come pressione sanguigna e temperatura corporea. Altri esempi includono una visione a raggi X virtuali basata su una tomografia effettuata in precedenza o su immagini in tempo reale, oppure ancora la posizione di un tumore nel video di un endoscopio.

- **Archeologia** : si "aumentano" le caratteristiche archeologiche sul paesaggio moderno permettendo agli archeologi di ipotizzare possibili configurazioni dei siti archeologici da strutture pre-esistenti. I modelli generati al computer di rovine, edifici, paesaggi o addirittura persone antiche sono stati impiegati nelle prime applicazioni AR archeologiche.

Capitolo 2

Blender come Motore Grafico

Blender è un software libero e multipiattaforma di modellazione, rigging, animazione, compositing e rendering di immagini tridimensionali[10]. Dispone inoltre di funzionalità per mappature UV, simulazioni di fluidi, di rivestimenti, di particelle, simulazioni non lineari e creazione di applicazioni/giochi 3D. È disponibile per vari sistemi operativi: Microsoft Windows, macOS, GNU/Linux, FreeBSD, assieme a porting non ufficiali per BeOS, SkyOS, AmigaOS, MorphOS e Pocket PC. Blender è dotato di un robusto insieme di funzionalità paragonabili, per caratteristiche e complessità, ad altri noti programmi per la modellazione 3D come Cinema 4D, 3D Studio Max e Maya.

2.1 Blender Foundation

In origine, il programma è stato sviluppato come applicazione interna dallo studio di animazione olandese NeoGeo. L'autore principale, Ton Roosendaal, fondò la società Not a Number Technologies (NaN) nel 1998 per continuare lo sviluppo e distribuire il programma che inizialmente fu distribuito come software proprietario a costo zero (freeware) fino alla bancarotta di NaN nel 2002. I creditori acconsentirono a pubblicare Blender come software libero, sotto i termini della licenza GNU General Public License, per il pagamento una-tantum di euro 100.000,00. Il 18 giugno 2002 fu iniziata da Roosendaal una campagna di raccolta fondi e il 7 settembre 2002 fu annunciato che l'obiettivo era stato

raggiunto e il codice sorgente di Blender fu pubblicato in ottobre. Ora Blender è un progetto open source molto attivo ed è guidato dalla Blender Foundation, una corporazione non-profit che si occupa di migliorare i servizi che il software mette a disposizione provvedendo a rilasciare aggiornamenti e patch. La community di Blender è grande ed attiva: molti utenti non si limitano ad utilizzare il programma a scopo privato e/o pubblico ma condividono i propri lavori, le proprie esperienze e le proprie conoscenze al fine di migliorare l'esperienza di utilizzo.

2.2 Funzionalità

Blender richiede poco spazio per essere installato e può essere eseguito su molte piattaforme. Sebbene sia spesso distribuito senza documentazione o esempi, il software è ricco di caratteristiche tipiche di sistemi avanzati di modellazione[11]. Tra le sue potenzialità, possiamo ricordare:

- Supporto per una grande varietà di primitive geometriche, incluse le mesh poligonali, le curve di Bézier, le NURBS, le metaball e i font vettoriali[12].
- Conversione da e verso numerosi formati per applicazione 3D, come Wings 3D, 3D Studio Max, LightWave 3D e altri.
- Strumenti per gestire le animazioni, come la cinematica inversa, le armature (scheletri) e la deformazione lattice, la gestione dei keyframe, le animazioni non lineari, i vincoli, il calcolo pesato dei vertici e la capacità delle mesh di gestione delle particelle[13].
- Gestione dell'editing video non lineare.
- Caratteristiche interattive attraverso il Blender Game Engine, come la collisione degli ostacoli, il motore dinamico e la programmazione della logica, che permettono la creazione di programmi stand-alone o applicazioni real time come la visione di elementi architettonici o la creazione di videogiochi.

- Motore di rendering interno versatile ed integrazione nativa col motore esterno YafaRay (un raytracer open source)
- Motore di rendering unbiased Cycles disponibile internamente a partire da Blender 2.61.
- Sistema Sculpt per modellare gli oggetti 3D.
- Scripting in python per automatizzare e/o controllare numerosi aspetti del programma e della scena.

2.3 Caratteristiche

Blender permette di seguire tutta la PIPELINE lavorativa senza dover uscire dal programma. Bozze, modellazione, rendering, sculpting, animazione, montaggio video, compositig, realtime, VR: avviene tutto al suo interno.

2.3.1 Interfaccia Utente

Blender ha la fama di essere un programma difficile da imparare. Quasi tutte le funzioni possono essere richiamate con scorciatoie e per questo motivo quasi tutti i tasti sono collegati a numerose funzioni. Da quando è stato pubblicato come opensource, la GUI è stata notevolmente modificata, introducendo la possibilità di modificare il colore, l'uso di widget trasparenti, una nuova e potenziata visualizzazione e gestione dell'albero degli oggetti e altre piccole migliorie, come la scelta diretta dei colori. L'interfaccia di Blender si basa sui seguenti princìpi:

- **Modalità di modifica** : le due modalità principali sono la *Modalità Oggetto* (object mode) e la *Modalità modifica* (edit mode), ed è possibile passare dall'una all'altra per mezzo del tasto tab. La modalità oggetto può essere usata per manipolare oggetti singoli, mentre la modalità modifica è usata per modificare i dati di un oggetto. Per esempio, in una mesh poligonale, la modalità oggetto può essere usata per muovere, scalare e

ruotare l'intera mesh, mentre la modalità modifica è usata per modificare i singoli vertici della mesh.[14] Ci sono anche altre modalità, come la Pittura Vertici, la Modalità Scultura o la Pittura Pesi.

- **Scorciatoie da tastiera** : la maggior parte dei comandi è impartibile attraverso la tastiera. Fino alla versione 2.x e specialmente nella versione 2.3x, questo era il solo modo per impartire comandi, e questo è stato il principale motivo che ha dato a Blender la reputazione di essere un programma difficile da imparare e capire. Le nuove versioni hanno menù molto più completi, che permettono di usare in larga misura il mouse per impartire i comandi.

- **Spazio di lavoro completamente ad oggetti** : l'interfaccia di Blender è formata da una o più scene, ognuna delle quali può essere divisa in sezioni e sottosezioni che possono essere formate da una qualunque immagine o vista di Blender. Ogni elemento grafico delle viste di Blender può essere controllato nello stesso modo in cui si controlla la finestra 3D - si possono ad esempio ingrandire i pulsanti della barra dei menù nello stesso modo in cui si ingrandisce un'immagine nella finestra di anteprima. La disposizione delle componenti dell'interfaccia di Blender è modificabile dall'utente, che può così lavorare a compiti specifici su un'interfaccia personalizzata e nascondere le caratteristiche non necessarie.

A partire dello sviluppo delle versioni 2.5x, è stata introdotta una nuova interfaccia, oltre al cambiamento di alcune combinazioni di tasti, rendendo il tutto più intuitivo per chi è alle prime armi con il programma.

2.3.2 Gestione del filesystem

Blender è dotato di un file system interno in grado di confezionare più scene in un singolo file (di estensione *".blend"*), i cui dati vengono organizzati come vari tipi di "blocchi ", come oggetti, armature, lampade, scene, materiali, immagini e così via. Un oggetto in Blender è costituito da blocchi di dati multipli. Ad

esempio, ciò che l'utente descrive come rete di poligoni è costituito da un blocco di dati *Object*, uno di *Mesh*, uno di *Materials* ed altri collegati tra loro. Tutto ciò può anche essere collegato ad altri file .blend, grazie all'utilizzo di file .blend come librerie di risorse. Una vasta gamma di plugin e script di importazione-esportazione estendono le funzionalità di Blender (accesso ai dati degli oggetti tramite un'API interna), consentendo di interagire con altri strumenti 3D: infatti, il software supporta diversi formati 3D per l'importazione e l'esportazione, tra cui 3D Studio (3DS), Filmbox (FBX), Autodesk (DXF), SVG, STL (per la stampa 3D), VRML e X3D.

2.3.3 Modellazione

La Modellazione è l'insieme di strumenti e tecniche che consentono di modificare la forma degli oggetti di base di Blender (le "primitive") per ottenere le forme desiderate. All'interno di una scena di Blender 3D è possibile inserire vari tipi di oggetti, alcuni dei quali potranno essere visualizzati nel *rendering*, mentre altri serviranno da oggetti ausiliari o per impostare alcune caratteristiche della scena per il rendering (ad es.: l'illuminazione, con vari tipi di fonti luminose)[13]. A prescindere dal suo tipo, la creazione di un nuovo oggetto – e il suo inserimento nella scena 3D – avviene mediante l'utilizzo della funzione *ADD OBJECT*, che può essere richiamata da un menù, tramite shortcut o tramite codice Python. Quando si crea un nuovo oggetto, questo viene inserito nella scena con la sua *Origin* (un punto che lo identifica nella scena 3D, per dargli delle coordinate di riferimento) posizionata esattamente dove si trova il *3D Cursor*, ovvero un oggetto virtuale che non viene renderizzato, rappresentato mediante la forma di un "mirino" all'interno della *3D View*. Tale oggetto è provvisto di coordinate XYZ, per identificarne la posizione nell'universo virtuale 3D di Blender. È possibile posizionarlo in un altro punto della scena con un click del tasto sinistro nella *3D View*, oppure specificandone le coordinate nella sezione *"3D Cursor"* del pannello *Properties*, richiamabile con lo shortcut N quando il cursore del mouse si trova all'interno di una *3D View*. Se si desidera posizionare un oggetto in una coordinata specifica al momento della sua

creazione, quindi, si possono specificare le coordinate in *3D Cursor – Location* e procedere con l'inserimento dell'oggetto. Per ciascun oggetto presente nella scena è possibile impostare alcune caratteristiche di base nella scheda *Object* all'interno dell'editor *Properties*. Tra i campi presenti in questa scheda abbiamo, in cima, una casella di testo contenente il nome dell'oggetto, che può essere quindi rinominato semplicemente cliccando su tale casella e scrivendo un nuovo nome - è buona norma scegliere nomi significativi per gli oggetti. Le **Mesh** sono oggetti geometrici definiti da vertici, spigoli e facce. I vertici sono punti, dotati di coordinate 3D (XYZ) che ne definiscono la posizione nello spazio tridimensionale di Blender 3D. Gli spigoli ("edges") sono segmenti che collegano coppie di vertici. Le facce sono superfici di riempimento delimitate da tre o più spigoli; a seconda del numero di spigoli che delimitano una faccia, avremo dei TRIS (facce triangolari), QUADS (facce quadrangolari) o NGONS (poligoni con un numero N di lati). Tre o più spigoli possono formare una faccia senza per questo dover creare una faccia di riempimento; è possibile cancellare solo le facce o gli spigoli di una mesh mantenendo gli altri elementi geometrici della stessa. Le **Curve** e le **Superfici** sono forme particolari di Blender 3D che possono essere puramente bidimensionali (*Curves*) o tridimensionali (*Surfaces*). Si distinguono dalle geometrie di tipo Mesh in quanto queste sono definite mediante elementi discreti, i vertici, connessi tra loro da spigoli in modo da definire delle facce o superfici; le Curves e le Surfaces, di contro, presentano sì dei "punti-chiave" nello spazio (i cosiddetti punti di controllo), ma i collegamenti tra tali punti e le eventuali superfici di riempimento vengono realizzati implementando una formula matematica, il che consente di realizzare forme smussate, "dolci", con un numero teoricamente infinito di suddivisioni interne, controllabili però da un insieme molto ristretto di punti di controllo. Blender realizza quindi un'interpolazione tra i punti di controllo, "valutando" l'espressione matematica che rappresenta la curva o superficie, nascondendo all'utente la parte di calcolo e fornendo un set di strumenti semplici e intuitivi per lavorare su questi nuovi oggetti.

2.3.4 Animazione e Rigging

Il concetto di *Animazione* è semplice: modificare un oggetto nel tempo ed il concetto di tempo in Blender è espresso attraverso una Timeline: spostandosi su di essa è possibile far scorrere il tempo avanti e indietro. La tecnica più semplice per realizzare un'animazione è quella di utilizzare i *Keyframe*, ovvero dei fotogrammi che memorizzano le pose principali del modello. La parola *Rig* in inglese indica il sistema di fili e meccanismi che fanno muovere le marionette. Il concetto nel 3D è identico, solo che si hanno a disposizione (invece dei fili) le parentele e i Constraint tra gli oggetti, in particolare tra un tipo di oggetti detti "Armature" che sono composti da sotto oggetti chiamati "Bone". Creare un rig completo richiede diversi step:

- prima si crea il sistema di ossa dentro l'Armatura in Edit Mode;
- poi si crea un sistema di controllo che permette muovere le ossa con meno comandi possibili in Pose Mode;
- infine si aggancia l'Armatura all'oggetto da animare.

2.3.5 Rendering

Cycles è un motore di rendering Unibiased integrato all'interno di Blender. I motori come Cycles si differenziano dai motori come il Blender Render (Biased) principalmente perché utilizzano algoritmi per il calcolo dell'immagine finale basati sulla fisica reale della luce. Questo comporta una resa molto più realistica del rendering, ma tempi di rendering nettamente più lunghi. La potenza di Cycles rispetto ad altri motori Unibiased è la possibilità di utilizzare la scheda video (o le schede video) in modalità OPENCL (schede video ATI) o CUDA (schede video Nvidia) in modo da velocizzare notevolmente il calcolo rispetto all'utilizzo dei processori. Per la realizzazione delle scene di questa tesi sono state sfruttate le schede video di tipo CUDA. Cycles non è settato come motore di render principale. Per utilizzarlo bisogna obbligatoriamente cambiare, tramite un apposito pulsante posto in alto all'interfaccia, il tipo di Rendering (da

Blender Render a Cycles). Essendo Cycles un motore di rendering che utilizza algoritmi complessi per simulare il risultato della luce, anche i materiali devono avere alcune caratteristiche "reali" e ciò è possibile grazie all'utilizzo dei *Nodi* all'interno del Node Editor.

2.4 Licenza

Blender è rilasciato sotto la licenza GNU General Public Licence (the GPL), quindi:

- si ha il diritto di utilizzare il programma per qualsiasi scopo;
- si ha il diritto di modificare il programma e avere accesso ai sorgenti;
- si ha il diritto di copiare e modificare il programma;
- si ha il diritto di migliorare il programma, e rilasciare la propria versione.

In cambio di questi diritti, si hanno alcune responsabilità:

- distribuire la licenza GPL insieme al programma, in modo che l'utilizzatore sia a conoscenza dei suoi diritti garantiti dalla licenza;
- distribuire anche il codice sorgente, o fare in modo che sia liberamente accessibile;
- se si modifica il codice e si rilascia una nuova versione del programma, questa rimane sotto licenza GPL e bisogna accertarsi che il codice modificato sia liberamente accessibile (non è possibile utilizzare codice GPL in programmi proprietari).
- non è possibile restringere la licenza del programma entro i termini della GPL (non si possono limitare i diritti garantiti dalla licenza).

2.5 Impiego

La comunità di Blender[15] è composta da persone provenienti da tutto il mondo, artisti grafici principianti e professionisti, utenti occasionali e case commerciali[16], in particolare:

- hobbysti/studenti che vogliono esplorare il mondo della computer graphics (CG) e dell'animazione 3D[17]
- artisti 2D che producono singole immagini/poster o elaborano singole immagini
- artisti 2D o gruppi che producono animazioni per spot televisivi o corti (come "The Magic of Amelia")
- artisti 3D che lavorano soli o in team con altre persone per produrre brevi animazioni CG, a volte incorporando azione live (come "Suburban Plight")
- gruppi 3D che producono film d'animazione (come "Elephant's Dream", "Plumiferos", "Big Buck Bunny", "Sintel" e tanti altri).
- gruppi 3D che lavorano insieme per produrre film con azione live che include Computer Graphics[18].

Team 2D e 3D che producono film e animazioni sono spesso specializzati in alcuni aspetti della CG. Alcuni di questi lavori specifici che potrebbero includere l'uso di Blender sono:

- **Regista** - Definisce il contenuto di ciascuna scena, e l'azione (animazione) relativa a quella scena. Definisce i movimenti di camera all'interno della scena.
- **Modellatore** - Crea una realtà virtuale. Alcune specialità includono Character, Prop e modellatori di Panorami/Stage

- **Cameraman, Direttore della Fotografia**: imposta la telecamera e il suo movimento, filma l'azione live, crea il render di output
- **Pittore di materiali** - Dipinge il set, gli attori, e qualsiasi cosa che si muova. Se non si muove, la dipinge lo stesso
- **Animazione e Rigging** - Fa saltellare le cose usando armatures
- **Specialista di Luci e Colori** - Illumina il set, aggiusta i colori in modo che appaiano bene quando illuminati, aggiunge polvere e sporco ai materiali, alle scene e alle textures.
- **Talento specializzato** - Fluidi, Motion capture, Abiti, polvere, sporco, fuoco, esplosioni, insomma, le cose divertenti
- **Montatore** - Prende tutto il girato dal Direttore della Fotografia e crea le sequenze per un film gradevole. Taglia le cose superflue.

Capitolo 3

Unity come piattaforma di sviluppo

Unity[19] è uno strumento di authoring integrato multipiattaforma sviluppato da Unity Technologies per la creazione di videogiochi 2D e 3D o altri contenuti interattivi, quali visualizzazioni architettoniche o animazioni 3D in tempo reale.

3.1 Descrizione

L'ambiente di sviluppo Unity gira sia su Microsoft Windows sia su macOS e i giochi che produce possono essere eseguiti su PC, Mac, Linux, Xbox 360, PlayStation 3, PlayStation Vita, Wii, iPad, iPhone, Android, Windows Mobile e, nel corso dell'anno 2014, Playstation 4, Xbox One e Wii U. Può anche produrre giochi per browser web che utilizzano il plugin Unity web player, supportate su Mac e Windows. All'interno di giochi 2D, Unity consente l'importazione di sprites e di un render avanzato, mentre per i giochi 3D consente di specificare le impostazioni di compressione e risoluzione della texture per ogni piattaforma supportata dal motore di gioco e fornisce supporto per il bump mapping, reflection mapping, parallax mapping, occlusione ambientale dello spazio dello schermo (SSAO), ombre dinamiche usando shadow maps, render-to-texture ed effetti post elaborazione a schermo intero[20]. Come motore di gioco fornisce tutto il necessario per realizzarne uno, infatti, oltre a tutte le impostazioni gra-

fiche, si ha il completo controllo sulla fisica, sulle animazioni, sull'audio, sulla gestione dei video e della AI e fornisce anche un supporto interno alla realizzazione di titoli multiplayer, sia online sia in locale[21]. Una caratteristica extra che Unity offre è il pool di servizi dedicato agli sviluppatori: Unity Ads, Unity Analytics, Unity Certification, Unity Cloud Build, Unity Everyplay, Unity IAP, Unity Multiplayer, Unity Performance Reporting e Unity Collaborate[22].

3.2 Interfaccia Utente

Al primo impatto, Unity sembra essere ricco di feature e potenzialità, ma allo stesso tempo di difficile comprensione: è infatti necessario capire a fondo l'interfaccia grafica e familiarizzare con essa prima di mettersi al lavoro. La finestra principale dell'editor è costituita di tanti pannelli che possono essere riorganizzati, raggruppati, ancorati e distaccati. Ciò implica che l'aspetto dell'editor può essere differente da progetto a progetto, in base alle specifiche esigenze del momento.

3.2.1 Project Window

Mostra la libreria degli Assets[23] disponibili nel progetto; ogni volta che un nuovo package o asset viene importato, questo viene visualizzato nella Project Window. La parte a sinistra mostra la struttura del progetto, organizzata gerarchicamente ad albero; quando una cartella viene selezionata tramite click, il suo contenuto viene mostrato nel pannello immediatamente sulla destra. La finestra include un'opzione di ricerca per facilitare il drag and drop degli elementi nella scena.

3.2.2 Scene View

La Scene View permette di navigare visivamente nella scena e di modificarla; essa può mostrare una prospettiva 2D o 3D in base al tipo di progetto su cui si sta lavorando.Si tratta della parte interattiva dello sviluppo, in quanto permette di agire sugli oggetti e sul mondo che si stanno creando: ci si posizionano gli

scenari, i personaggi, le camere, le luci e qualsiasi altro immaginabile oggetto del tipo *Game Object*. Nell'angolo in alto a destra è presente il **Gizmo**, ovvero lo stumento che mostra l'attuale posizione degli assi cartesiani della camera. Il Gizmo permette di ruotare e cambiare vista semplicemente facendo click su uno degli assi. Per poter agire attivamente sugli oggetti presenti nella scena è possibile ricorrere alle icone apposite che si trovano in alto a sinistra, oppure alle shortcut da tastiera: esse includono operazioni di movimento, rotazione, scala e trasformazione, accessibili e modificabili anche nell'*Inspector*, il menu a destra della scena. Sono presenti, in alto nella parte centrale, anche dei bottoni che permettono di animare la scena/il gioco, al fine di rendersi conto di come effettivamente tutti gli elementi sono orchestrati.

3.2.3 Inspector Window

L'Inspector Window mostra tutte le informazioni dettagliate del Game Object selezionato al momento, inclusi tutti i *"Component"* ad esso associati, come ad esempio Mesh, Materials, Script e Animazioni.

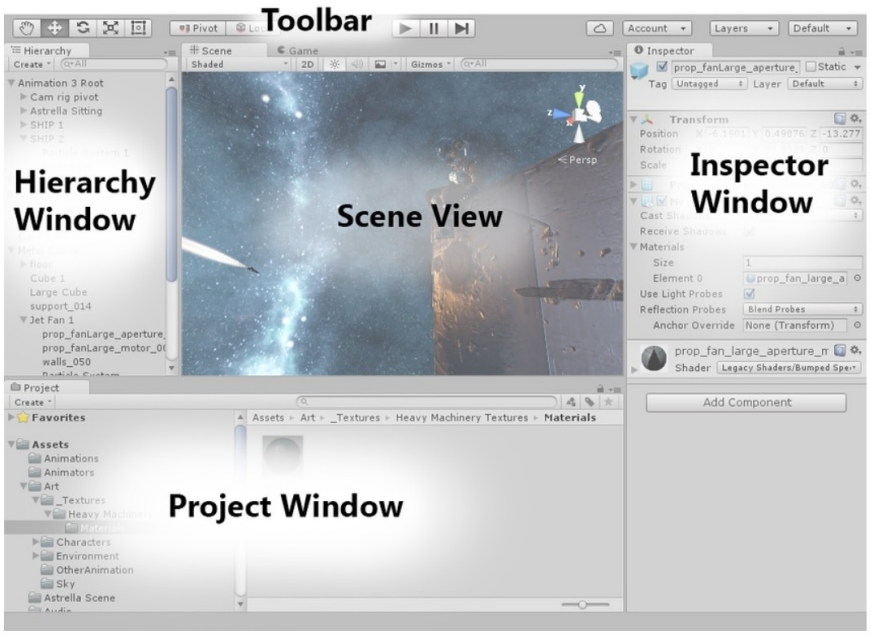

Figura 3.1: Rappresentazione dell'Interfaccia Utente di Unity

3.3 Scripting

Il codice sorgente è uno degli ingredienti fondamentali di qualsiasi gioco/applicazione. Per poter raccogliere input e generare conseguenti e logici output è necessario scrivere Script affinché essi siano correttamente gestiti. Unity supporta lo scripting in C# e JavaScript e ne permette la stesura tramite un editor integrato - *Mono* - oppure tramite Visual Studio (prerequisito per la corretta installazione del programma)[24].

3.4 Asset Store

L'Asset Store[23] è la homepage che permette l'accesso a tutte le librerie gratuite e a pagamento che possono essere scaricate ed importate direttamente da dentro Unity. Contiene modelli, mesh, shader, sistemi particellari, effetti grafici per l'illuminazione e la post-produzione, tracce audio e prefabs.

Capitolo 4

Vuforia come supporto alla Realtà Aumentata

Vuforia è un avanzato kit di sviluppo software (SDK) che consente la creazione di applicazioni di Realtà Aumentata. La sua licenza non è di tipo Open Source, ciò implica il pagamento di una quota al fine di utilizzare appieno le funzionalità che mette a disposizione nel caso in cui si è interessati alla commercializzazione e alla distribuzione dell'app. Se, al contrario, si desidera utilizzare questo SDK in licenza Personal Use o a scopo didattico è sufficiente registrarsi al portale di Vuforia per iniziare ad essere operativi, seppur con qualche lecita limitazione sul numero di marker caricabili.

4.1 Descrizione

Vuforia[25] utilizza la tecnologia della Computer Vision per riconoscere e tracciare in tempo reale le immagini piane (Image Targets/marker) e semplici oggetti 3D. Questa capacità di registrazione delle immagini consente agli sviluppatori di posizionare e orientare oggetti virtuali, come i modelli 3D e altri supporti, in relazione alle immagini del mondo reale quando vengono visualizzate attraverso la fotocamera di un dispositivo mobile. L'oggetto virtuale traccia quindi la posizione e l'orientamento dell'immagine e renderizza il modello virtuale sulla base di come è stato posizionato via software. L'SDK di Vuforia supporta una

varietà di tipi di target 2D e 3D che verranno trattati nella prossima sezione del capitolo. Le funzionalità aggiuntive dell'SDK comprendono la localizzazione di "pulsanti virtuali", la selezione di target di immagine a runtime e la possibilità di creare e riconfigurare i set di destinazione in modo programmato in fase di esecuzione. Vuforia fornisce interfacce di programmazione delle applicazioni (API) in C++, Java, Objective-C++ e .Net tramite un'estensione al motore di gioco Unity. In questo modo, l'SDK supporta sia lo sviluppo di iOS sia di Android, consentendo allo stesso tempo lo sviluppo di applicazioni AR in Unity che sono facilmente trasportabili su entrambe le piattaforme. Le applicazioni AR sviluppate usando Vuforia sono quindi compatibili con un'ampia gamma di dispositivi mobili, inclusi iPhone, iPad e Android e tablet che eseguono versione Android OS 2.2 o superiore e un processore ARMv6.

4.2 Target

Con il termine *Target* si identificano tutte quelle immagini e quegli oggetti in grado di essere memorizzati, rilevati e tracciati dall'SDK al fine di renderizzare il modello 3D ad essi precedentemente associato.

4.2.1 Image Target

Gli Image Target sono costituiti da un'immagine .png o .jpg su scala RGB o Graysclae, di una dimensione inferiore ai 2MB e che contengono un'alta percentuale di quelle che vengono definite *"features"*, ovvero punti di alta tracciabilità. Per poter utilizzarli, è necessario prima registrarli al Vuforia Target Manager, ovvero il portale di gestione delle impostazioni per un'applicazione in Realtà Aumentata, affinchè ne venga calcolato e riconosciuto il *livello di aumentabilità*.

4.2.2 User Defined Target

Questi target definiti dall'utente sono creati in fase di esecuzione dai fotogrammi della fotocamera. Condividono la maggior parte delle funzionalità degli Image Targets standard, ad eccezione del fatto che non supportano i Virtual Buttons.

Per catturare le immagini che fungono da target, sono necessarie delle ottime condizioni di illuminazione, inoltre le immagini stesse devono essere ricche di dettagli, avere un buon contrasto e non avere pattern ripetitivi.

4.2.3 Model Target

I Model Target sono costituiti da oggetti fisici tridimensionali designati dall'utente e supportano il riconoscimento tramite l'individuazione della loro forma. Vuforia prevede delle caratteristiche anche per questo genere di target, ovvero:

- **staticità** : dopo essere stati rilevati devono rimanere nello stesso punto all'interno dell'ambiente;
- **colore** : oggetti colorati funzionano meglio;
- **geometria** : forme e profili particolari, insoliti e ricchi di dettagli sono più semplici da riconoscere e non rischiano di essere confuse con altre;
- **rigidità** : gli oggetti devono avere la stessa identica forma della loro controparte virtuale.

4.2.4 Multi Target

Si tratta di marker costituiti da più oggetti d'immagine in una disposizione geometrica definita. La posizione e l'orientamento di ogni Target all'interno di un Multi-Target è definito rispetto all'origine dello stesso, che si trova al centro volumetrico. Tutti i volti di un Multi-Target possono essere tracciati contemporaneamente perché dispongono di una posa predefinita relativa all'origine: ciò consente di monitorare l'intero Multi-Target quando uno dei suoi target figli è stato rilevato. I Multi-Target vengono creati definendo una relazione tra più campi di immagine esistenti utilizzando il Target Manager di Vuforia o manipolando direttamente il file XML di configurazione del dataset.

4.2.5 Cylinder Target

I Cylinder Targets sono cilindri che consentono di rilevare e monitorare le immagini avvolte in forme cilindriche e coniche. Essi supportano l'individuazione e il monitoraggio delle immagini sui piani superiori e inferiori del cilindro. È possibile crearli definendo le lunghezze laterali e i diametri del cilindro, aggiungendo le immagini che si desidera. I Cylinder Targets possono essere utilizzati per riconoscere e tracciare imballaggi dei prodotti approssimativamente cilindrici o conici come le lattine, le tazze e le bottiglie. Le caratteristiche delle immagini sono le stesse di quelle degli Image Targets, inoltre questi marker, essendo cilindrici,sono considerati modelli di occlusione, quindi permettono agli elementi virtuali creati durante l'esecuzione dell'applicazione di nascondersi dietro di essi.

4.2.6 VuMark

Vuforia ha ideato un tipo di codice a barre di nuova generazione chiamato VuMark: esso ammette la libertà di un design personalizzato e contemporaneamente permette la codifica dei dati agendo come target per la Realtà Aumentata. I disegni VuMark sono completamente personalizzabili, in modo da averne uno unico per ogni oggetto. I VuMark forniscono alcune delle stesse funzionalità degli ImageTargets in quanto possono essere riconosciuti individualmente e monitorate tramite l'SDK di Vuforia, ma esistono significative differenze che li rendono particolarmente utili per molte applicazioni aziendali e consumer: essi, infatti, possono rappresentare milioni di istanze univoche identificabili, possono codificare una grande varietà di formati di dati e consentono di distinguere tra prodotti apparentemente identici in base all'ID dell'istanza.

4.3 Extended Tracking

L'Extended Tracking utilizza le caratteristiche dell'ambiente circostante per migliorare le prestazioni di monitoraggio e mantenere il modello renderizzato anche

quando il marker/target non è più in vista, consentendo di mantenere gli oggetti virtuali attivi con un certo grado di persistenza. Anche se l'obiettivo scompare, Vuforia reperisce altre informazioni dall'ambiente per dedurne la posizione.

4.4 Smart Terrain

Lo Smart Terrain riconosce, traccia e ricostruisce gli oggetti fisici e le superfici: questi possono poi essere rappresentati come mesh 3D in una scena di Unity. La raccolta delle mesh della superficie e degli oggetti è denominata "Terrain" della scena, la quale avviene attraverso tre fasi:

- una fase di impostazione della scena creata dall'utente, aggiungendo gli oggetti reali e il target di inizializzazione;
- una fase di scansionamento in cui vengono acquisiti e ricostruiti la scena e gli oggetti utilizzati dal tracker dello Smart Terrain;
- un'ultima fase di monitoraggio in cui il terreno è aumentato in tempo reale dalla scena Unity sviluppata in precedenza.

Lo Smart Terrain è una funzionalità molto potente che però richiede grande potenza di calcolo per essere sfruttata appieno, motivo per cui è supportato solo su dispositivi con processori multicore.

Capitolo 5

Ispirazione e realizzazione delle scene

In questo capitolo della tesi vengono discussi gli elementi d'ispirazione che hanno permesso la nascita e lo sviluppo del progetto e viene analizzato nel dettaglio l'intero lavoro svolto.

5.1 L'arte di Aivazovsky come ispirazione

> *"L'artista che copia solo la natura ne diventa schiavo. I movimenti degli elementi naturali, dal vivo, sono impercettibili per un pennello: un fulmine, un colpo di vento o il tonfo di un'onda. L'artista deve memorizzarli. La trama dell'immagine è composta nella mia memoria, come quella di un poeta. Dopo aver fatto uno schizzo su un pezzo di carta, mi metto a lavorare e sto sulla tela fino a quando col mio pennello non ho detto tutto su di essa." Ivan Aivazovsky[26]*

Il mare suscita un grande fascino sugli uomini sin da tempi immemori, sia per la sua bellezza eterea in una notte di luna piena, sia per la sua furia incontrastata durante un pomeriggio di tempesta. Ivan Aivazovsky, colui che oggi viene ritenuto uno dei migliori artisti del mare del XIX secolo, se non addirittura il più grande, ne era ammaliato e dipingeva con passione la situazione di coloro che si trovavano tra le sue acque a bordo di barche di legno o grandi vascelli. Il

Romanticismo satura ogni sua pennellata di colore ed emerge ad ogni sguardo, rendendo quasi impossibile spostare gli occhi dai dipinti.[27]

Ed è proprio per l'ammirazione profonda di questo artista e per il magnetismo che suscitano le sue opere che i suoi quadri sono stati scelti come fonte di ispirazione per la realizzazione di questa tesi. Il lavoro prodotto e qui analizzato non vuole essere una mera e banale copia delle sue opere, ma un vero e proprio omaggio volto alla produzione - seppur semplice ed umile - di quello che potrebbe essere definito *"morceau d'arte digitale"*.

5.2 La collaborazione con il Museo della Regina di Cattolica come punto di partenza

Il Museo della Regina [28] è un museo di recente istituzione situato a Cattolica ed inaugurato nel 2000 dopo un importante intervento di rifunzionalizzazione di un vecchio edificio. Oggi conserva le tracce della tradizione navale del territorio ed ospita pezzi unici ed originali di modelli di imbarcazioni di varie dimensioni ed epoche storiche[29]. Il motivo per cui viene citata e brevemente descritta questa struttura risiede nel fatto che essa è stata uno dei punti di partenza per la realizzazione del progetto "Ocean Museum": infatti, a seguito della collaborazione tra la direttrice del museo ed il corso di studi di Ingegneria e Scienze Informatiche, si è sviluppato l'interesse per questa tematica. Nello specifico, la collaborazione era volta alla realizzazione di un evento innovativo caratterizzato da esperienze di Realtà Aumentata che celebrasse la prima uscita di un libro in formato ebook prodotto dal museo. Il ruolo ricoperto durante questa cooperazione è stato quello di modellista 3D di due tipologie di barche:

- **trabaccolo** : un'imbarcazione da pesca e/o da carico tipica del medio e alto Adriatico dotata di due alberi muniti di vela al terzo che esercitava il cabotaggio sin nello Ionio.

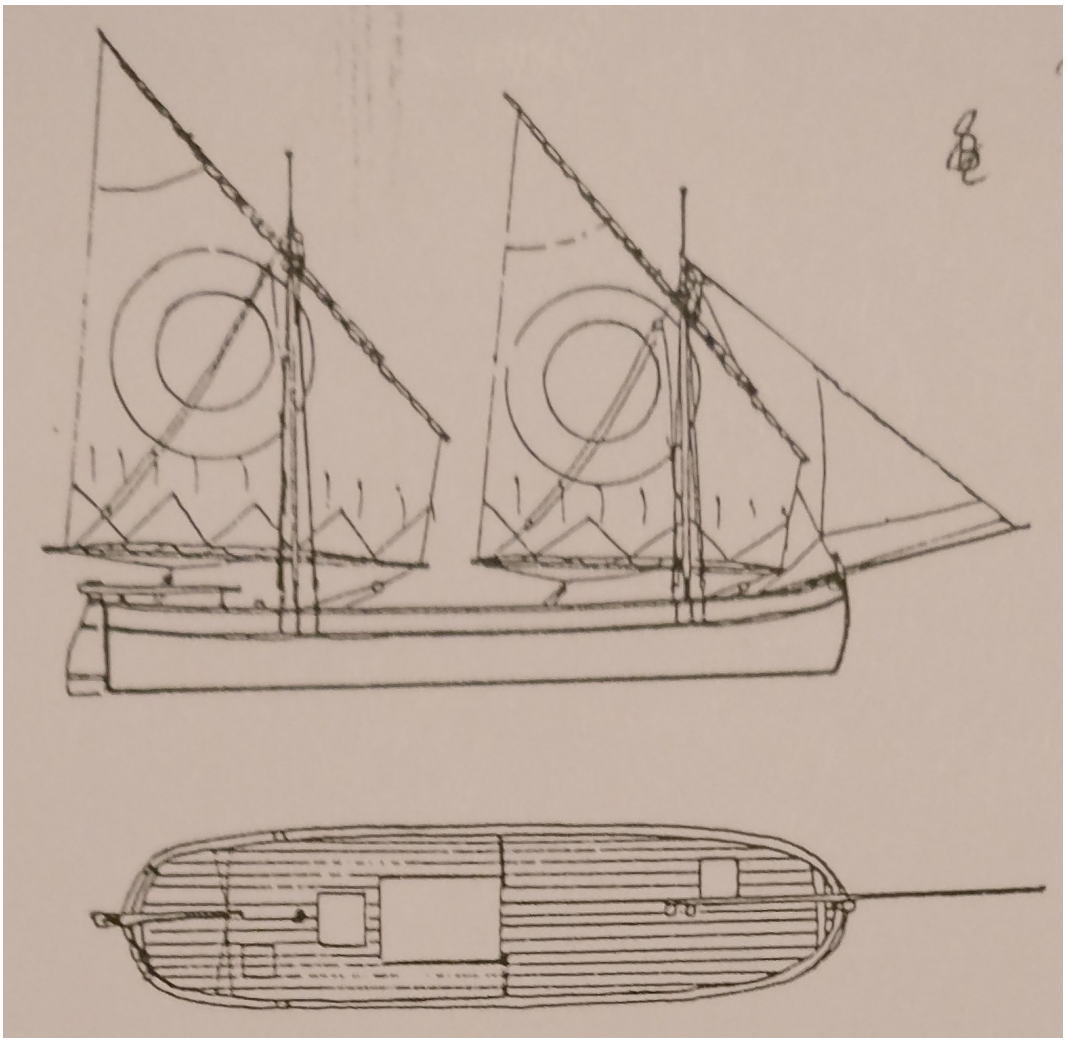

Figura 5.1: Modello di un trabaccolo

- **lancione** : una barca tradizionale caratteristica del litorale romagnolo destinata alla pesca, di lunghezza normalmente compresa tra i dieci e i dodici metri, che arma due alberi con vele al terzo, oltre ad un polaccone (vela triangolare simile a un fiocco) che viene invergato su una lunga asta buttafuori chiamata spigone.

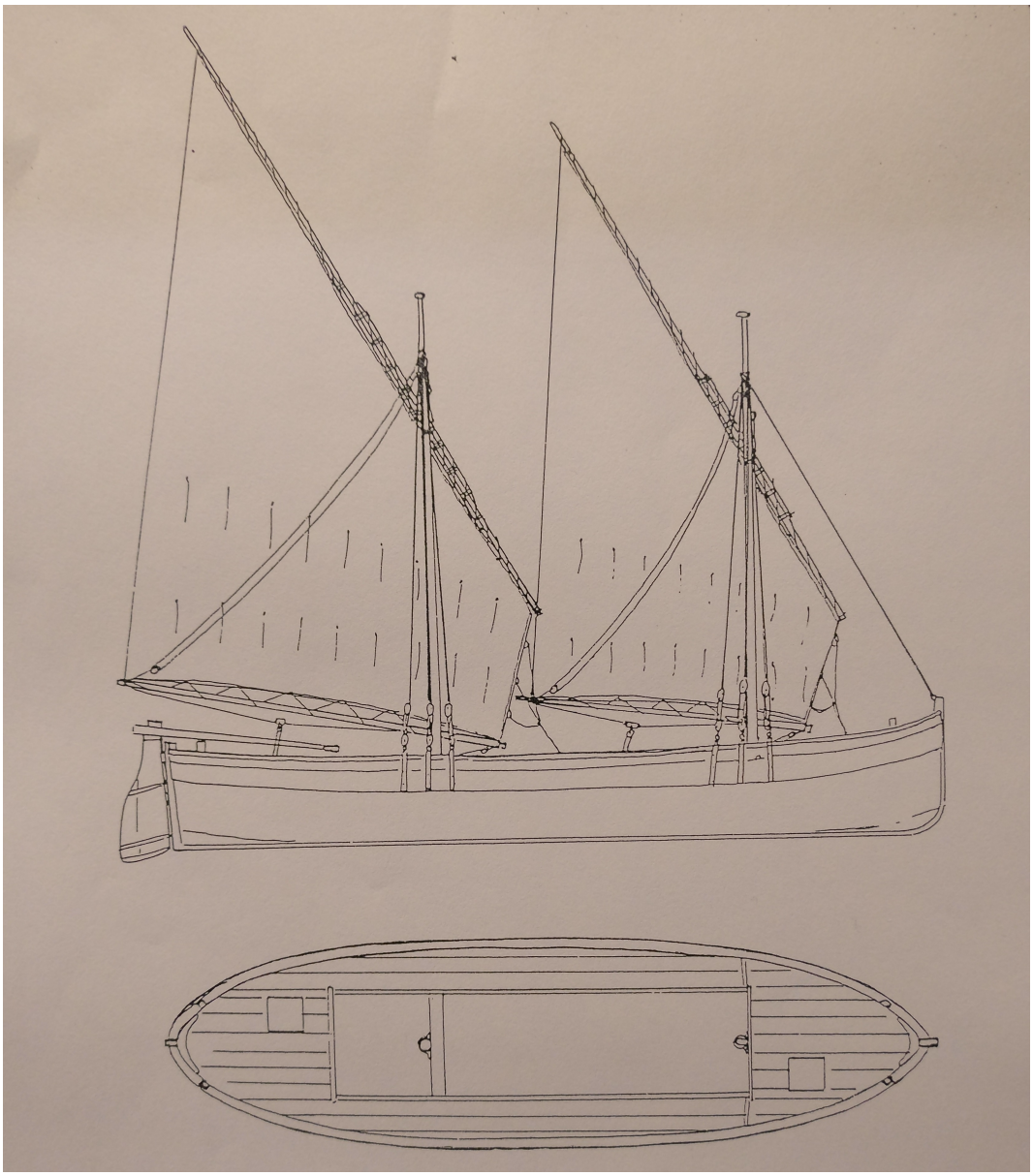

Figura 5.2: Modello di un lancione

L'esperienza vissuta unita all'amore per i quadri del pittore romantico hanno costituito il trampolino di lancio di questo progetto.

5.3 Progettazione del lavoro

Nel concreto, poiché l'obiettivo è quello di realizzare scene 3D in Blender in un numero consono a ricreare l'illusione di trovarsi nella sala di un museo, sono stati scelti quattro dipinti di Ivan Aivazovsky come riferimento. In particolare, i dipinti sono:

- **"Ship in the Stormy Sea"**

Figura 5.3: Dipinto "Ship in the Stormy Sea" di Ivan Aivazovsky

- **"Twelve Apostles"**

Figura 5.4: Dipinto "Twelve Apostles" di Ivan Aivazovsky

- **"Ship at Moonlight"**

Figura 5.5: Dipinto "Ship at Moonlight" di Ivan Aivazovsky

- **"Sea"**

Figura 5.6: Dipinto "Sea" di Ivan Aivazovsky

La quinta scena è stata realizzata senza tenere conto di un dipinto reale, ma è stata concretizzata semplicemente seguendo ciò che l'immaginazione proponeva.

Tutte le scene sono state ricreate prima su carta sotto forma di semplice bozza, poi al calcolatore.

5.4 Realizzazione pratica 3D

La modellazione 3D delle barche utilizzando il motore grafico è stata impegnativa e ha richiesto un numero considerevole di ore di lavoro. Sono state sfruttate tutte le conoscenze apprese durante il corso di Computer Graphics e tutte le nozioni acquisite online grazie a video-tutorial e articoli provenienti dalla community di Blender. Poiché è buona norma utilizzare *immagini di background* quando si realizzano modelli 3D, è stata scelta, dopo lunghe ricerche, il disegno tecnico della nave "Santa Maria", una delle tre imbarcazioni utilizzate da Cristoforo Colombo nel 1492.

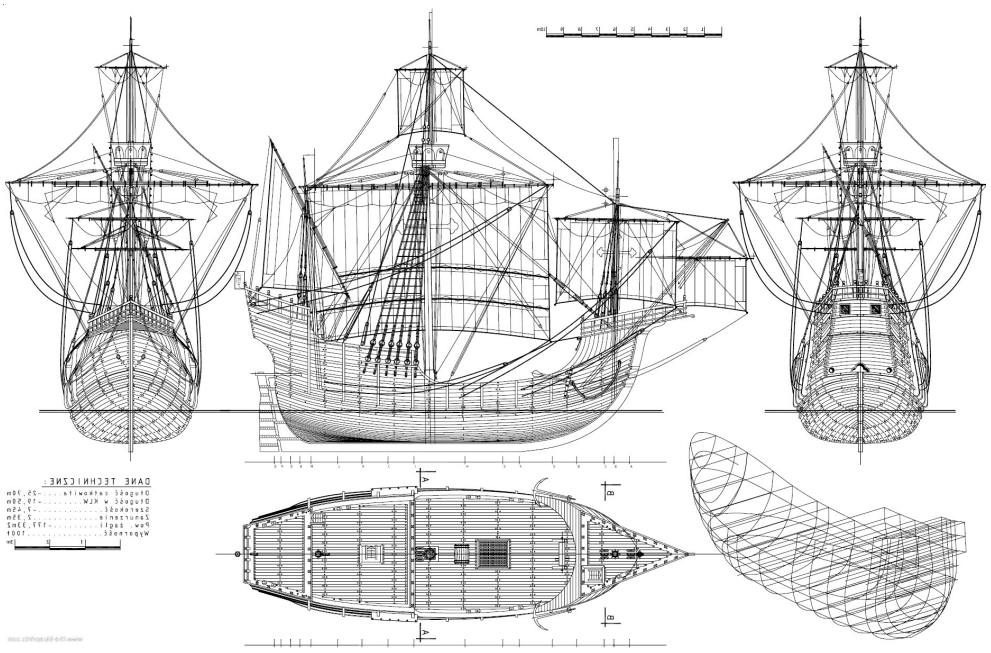

Figura 5.7: Blueprint della nave Santa Maria

L'immagine è stata reperita dal sito "The-Blueprints" (http://www.the-blueprints.com/it) ed è stata scelta in quanto molto simile al tipo di imbarcazione dipinta da Ivan Aivazovsky e poiché in possesso di tutte le caratteristiche per essere una buona guida alla modellazione: essa, infatti, comprende tutte le viste ortografiche, quindi la sua riproduzione 3D è da considerarsi più che fattibile.

5.4.1 Il galeone

La modellazione dello scafo della nave è stata, probabilmente, una delle parti più delicate dell'intero processo. I primi tentativi prevedevano la modellazione diretta di una mesh solida (cube) oppure piana (plane), ma non sono stati in grado di produrre risultati soddisfacenti. Dopo alcune ricerche e confronti con altri membri della community di Blender, si è optato per delineare l'intero profilo dello scafo mediante ciò che viene definita *"sottrazione di solidi"*: esattamente come uno scultore modella un blocco di creta eliminando tutti i pezzi che non fanno parte della sua statua, il Modifier *"bool"* ha permesso di sottrarre dalla

mesh principale - un parallelepipedo - le facce in eccesso, delineando così un profilo più dolce, realistico e con un numero adeguato di poligoni.

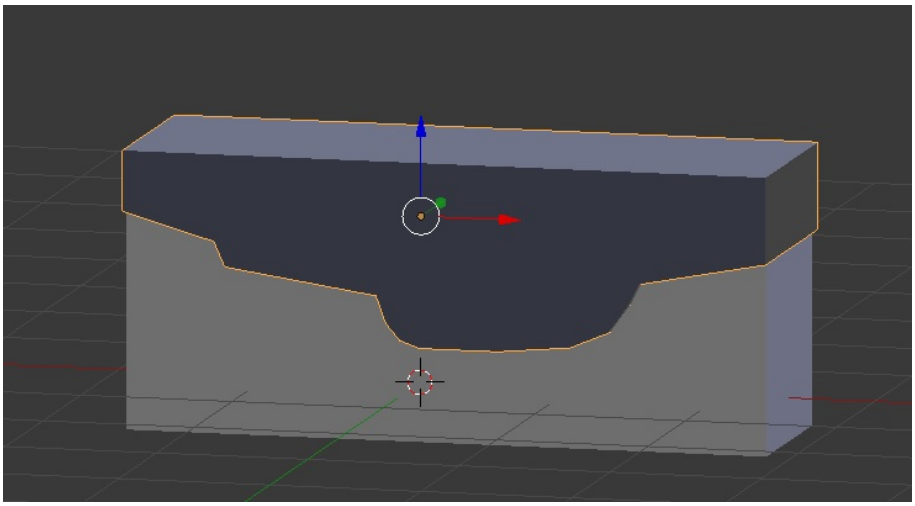

Figura 5.8: Sottrazione fra solidi - fase 1

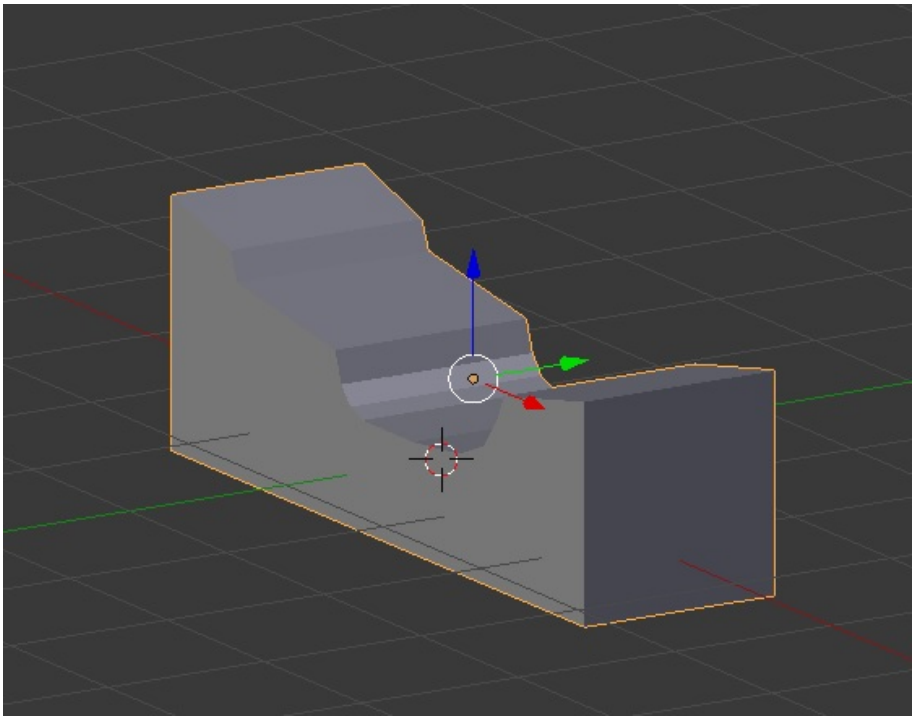

Figura 5.9: Sottrazione fra solidi - fase 2

Nella modalità Wireframe, ossia quella che permette di vedere e modificare gli oggetti come se fossero composti da fili, sono stati selezionati opportunamente i vertici per adagiarli al profilo mostrato dalla figura di riferimento.

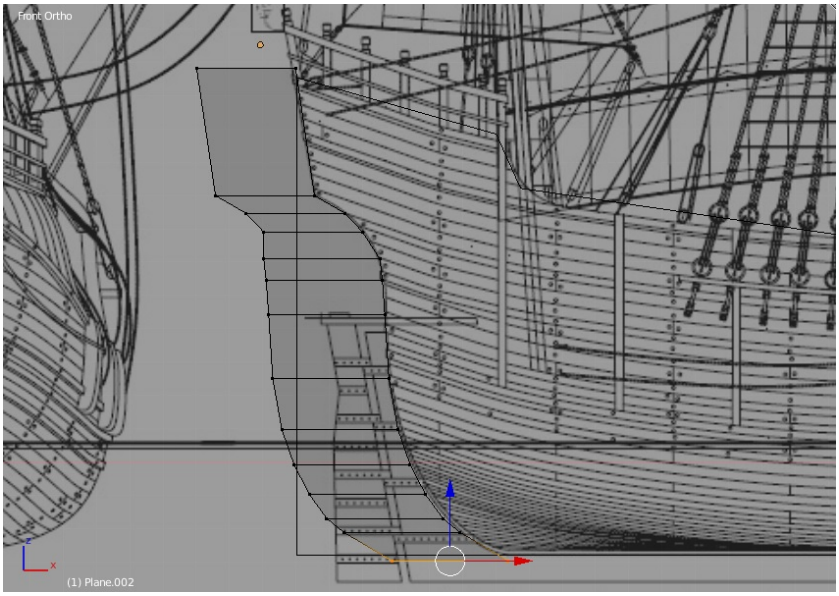

Figura 5.10: Sottrazione fra solidi - fase 3

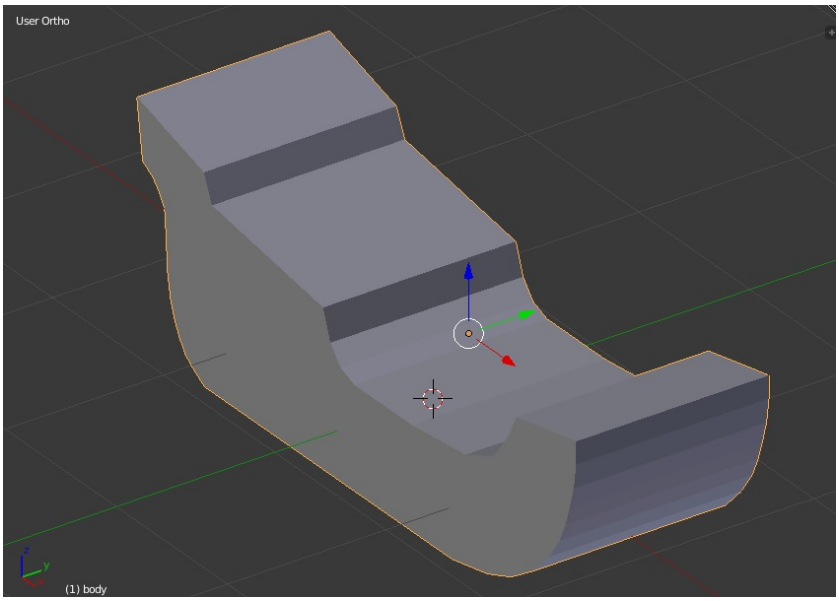

Figura 5.11: Sottrazione fra solidi - fase 4

L'aspetto dello scafo ultimato è il seguente:

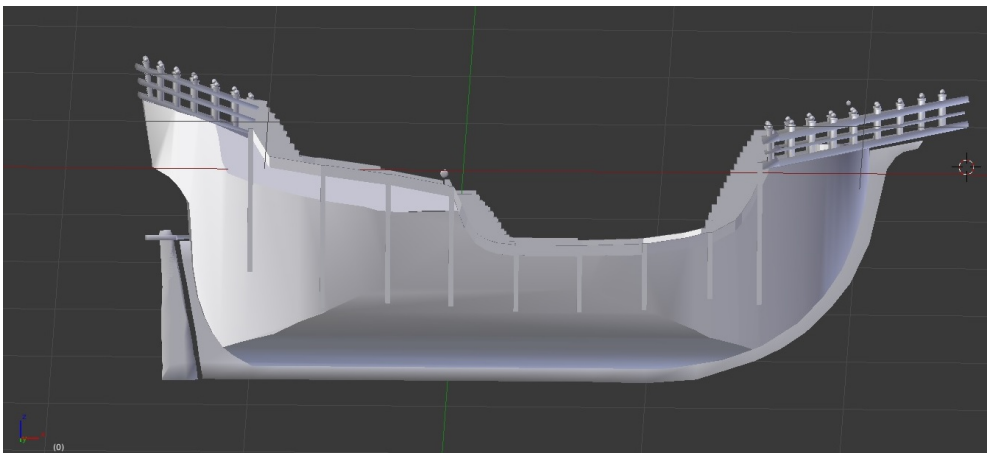

Figura 5.12: Vista laterale

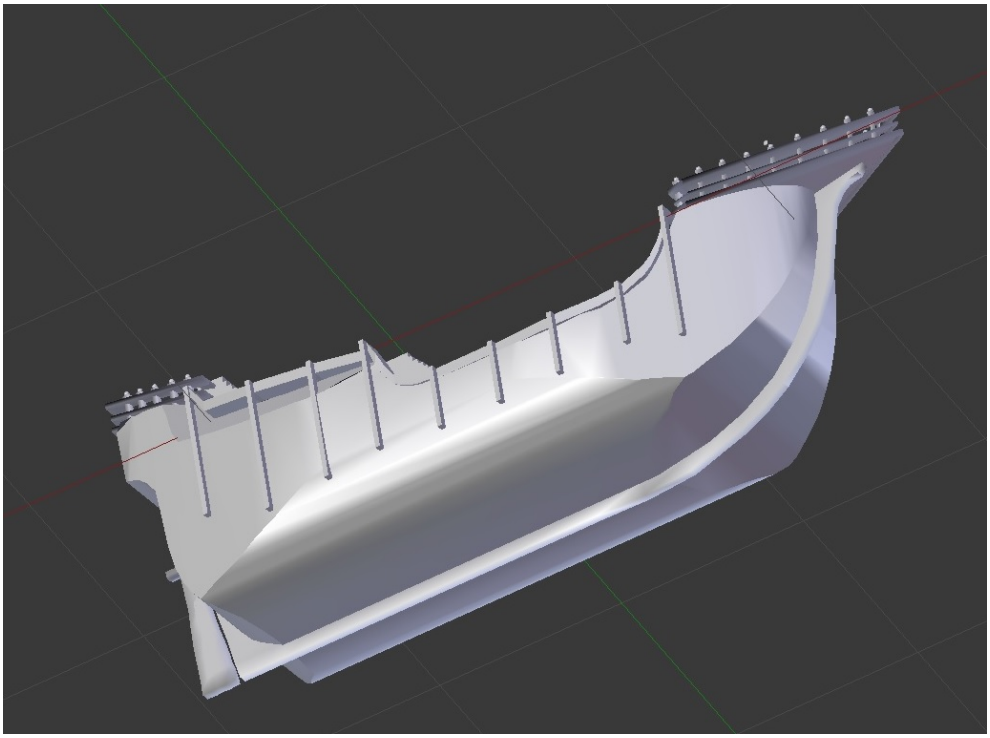

Figura 5.13: Vista obliqua dal basso

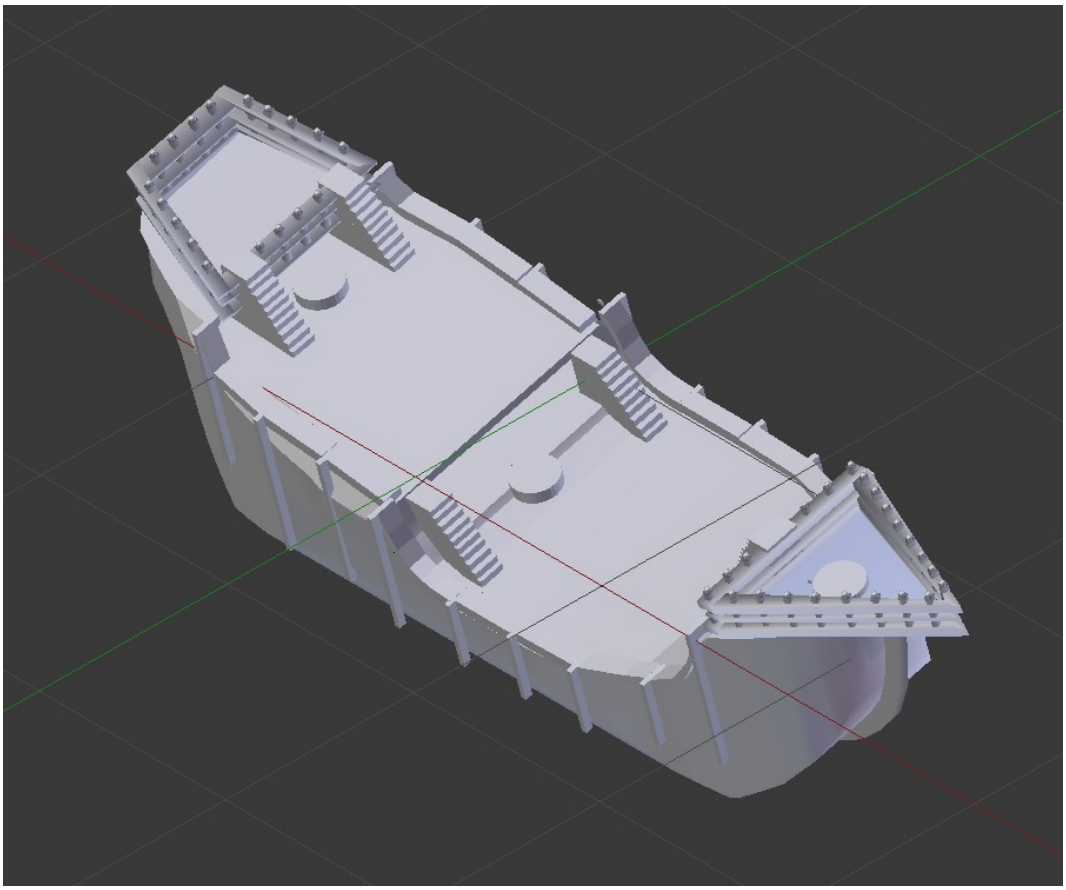

Figura 5.14: Vista obliqua dall'alto

5.4.2 Prua e poppa

I ponti a prua e a poppa sono stati modellati partendo da un semplice cubo che è stato poi opportunamente scalato ed estruso in modo da ottenere la forma desiderata per la base. Essa è stata poi duplicata e spostata sull'asse z due volte in modo da poter ottenere i bordi del ponte; per avere l'idea di "bordo" per entrambe le copie sono state eliminate le facce centrali. Le aste pomellate di legno sono state create a partire da un semplice cubo scalato, unito nella parte superiore alla mesh di una sfera. Dopo aver realizzato la prima, tutte le altre sono state duplicate.

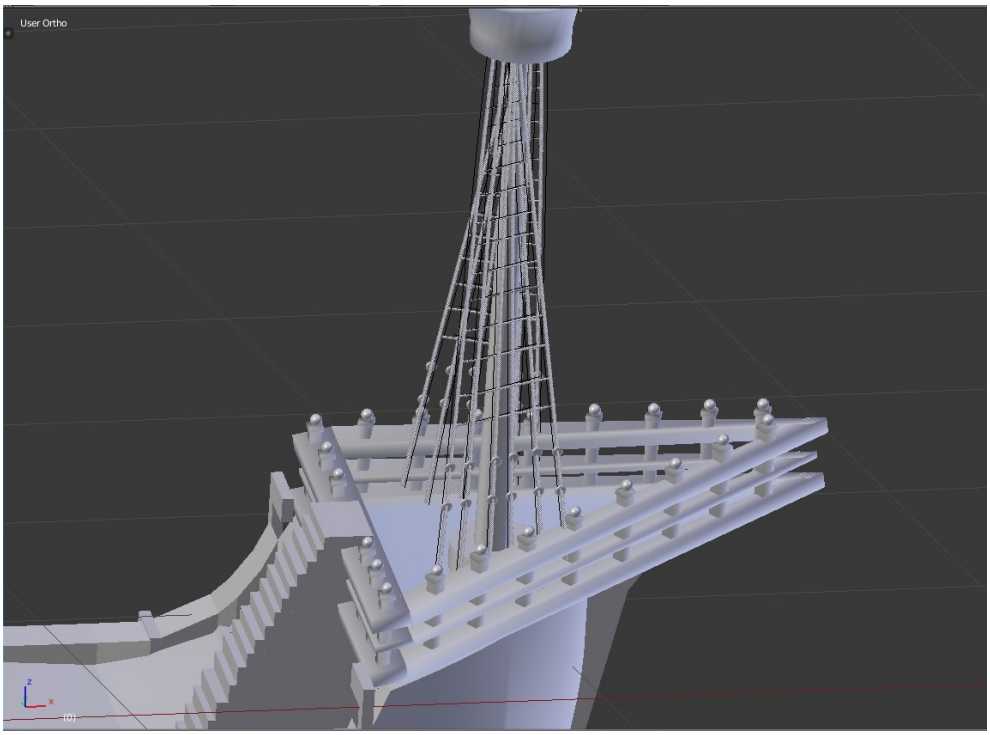

Figura 5.15: Ponte di prua

5.4.3 Le vele

Per poter riprodurre l'aspetto ricurvo che le caratterizza, le vele sono state realizzate a partire da una curva di Bezier; tramite il pannello *"Curve"* è stata scelta come *"Bevel Object"* una curva del tutto identica che, modificandone opportunamente i vertici di controllo e le tangenti, permette di "curvare" la mesh dando proprio l'idea di una vela mossa dal vento.

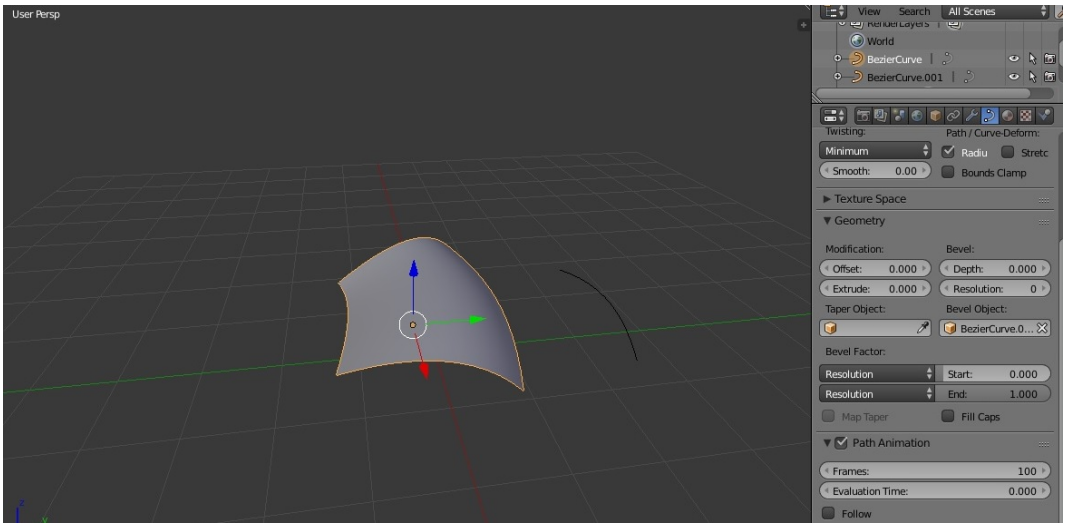

Figura 5.16: Modellazione della vela tramite curva

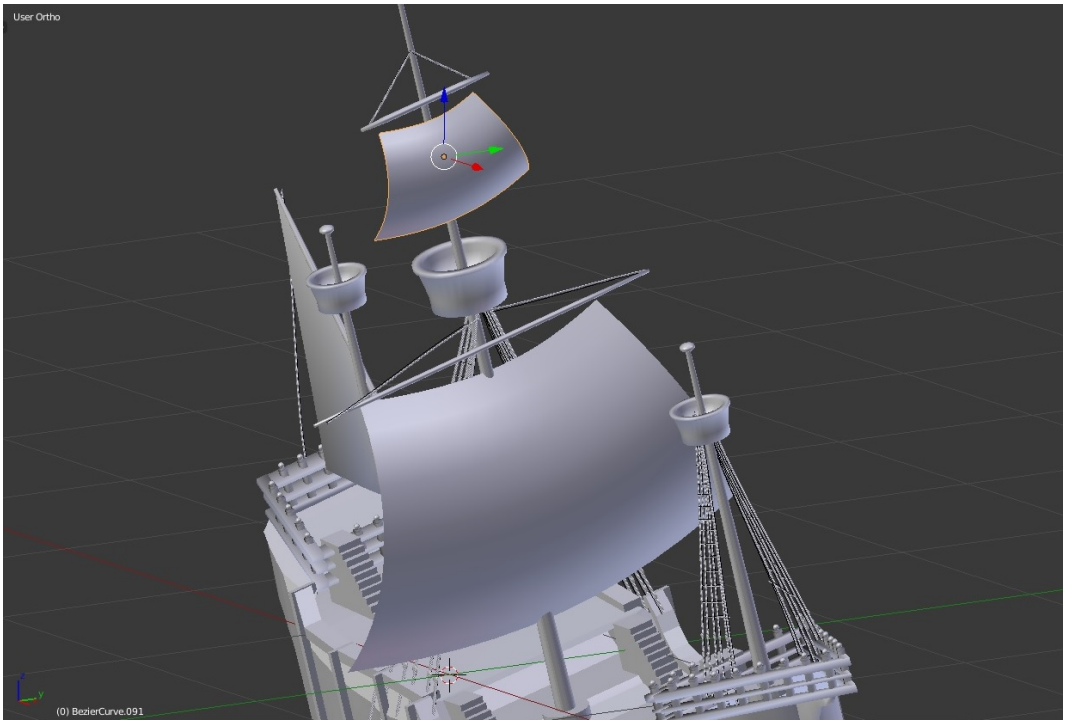

Figura 5.17: Esempi di vele

5.4.4 Le corde

L'aspetto realistico delle corde è stato ottenuto tramite la manipolazione del Modifier *"Screw"* applicato a quattro mesh di tipo *"Circle"* unite al centro.

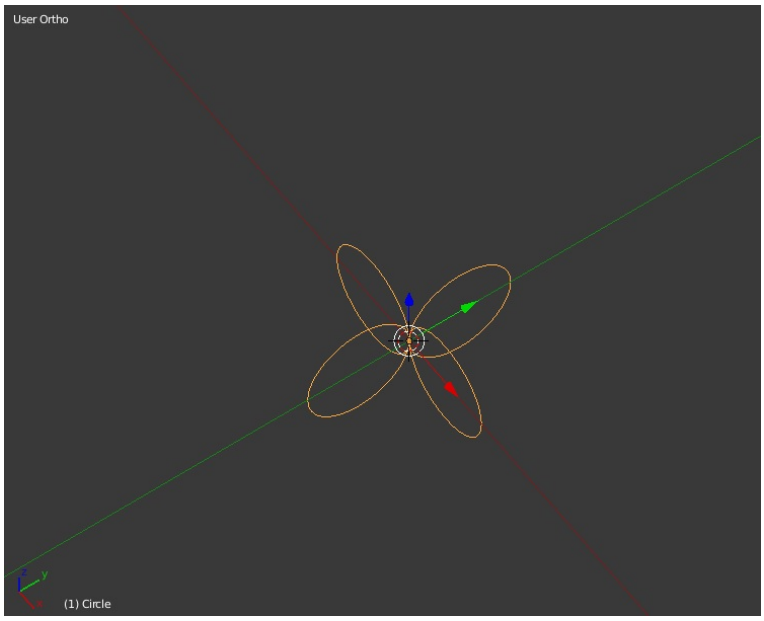

Figura 5.18: Modellazione della corda - fase 1

L'asse su cui sviluppare il Modifier è stato impostato su Y e il parametro *"Screw"* incrementato fino a 8.100.

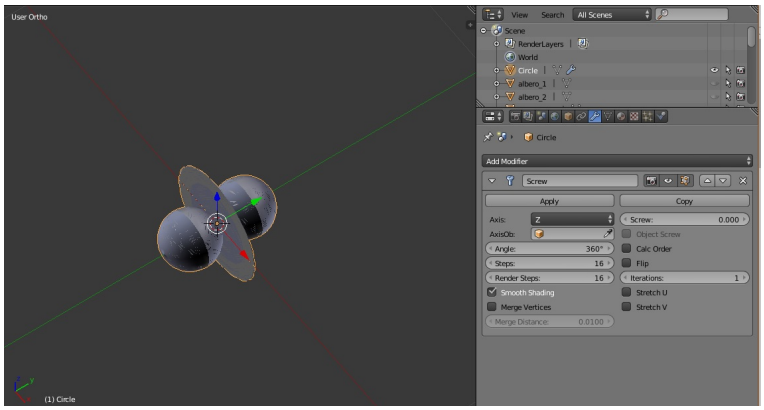

Figura 5.19: Modellazione della corda - fase 2

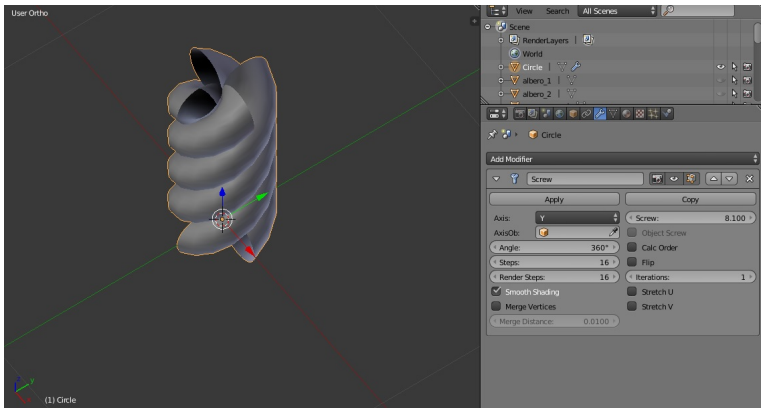

Figura 5.20: Modellazione della corda - fase 3

Per poter creare il profilo della corda è stata utilizzata una curva di Bezier "ancorata" ad un ulteriore Modifier su di essa, ovvero *"Curve"*.

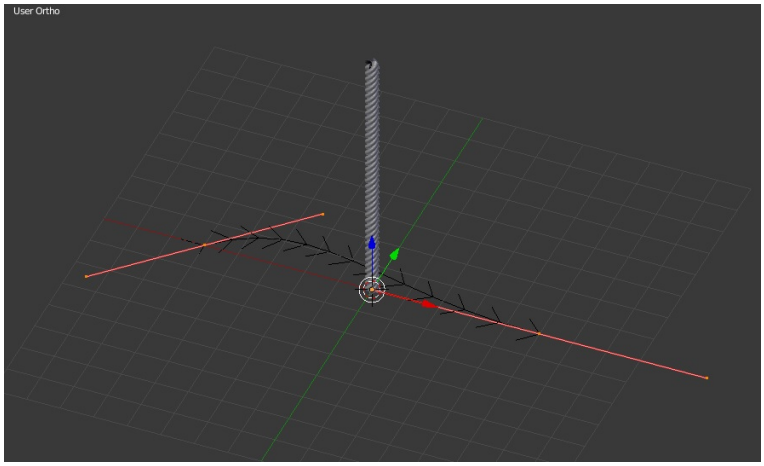

Figura 5.21: Modellazione della corda - fase 4

Per poterla muovere è sufficiente spostare i vertici di controllo della Bezier, mentre per aumentare la lunghezza effettiva della mesh si agisce sul parametro *"iteration"* incrementandolo a piacere.

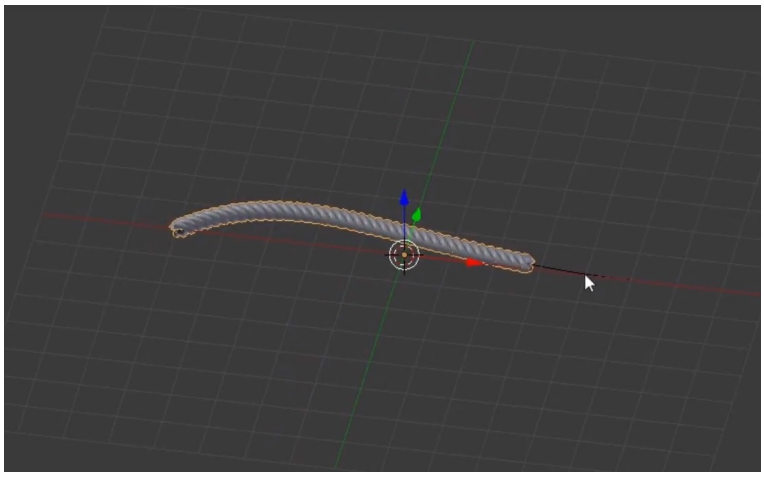

Figura 5.22: Modellazione della corda - fase 5

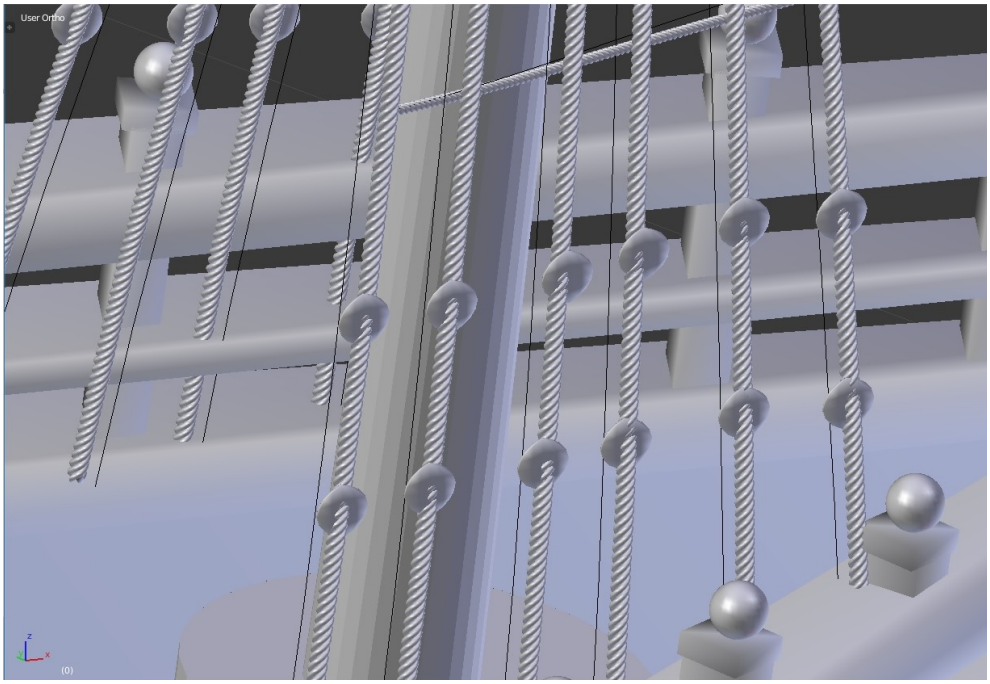

Figura 5.23: Esempi di corde

5.4.5 Gli alberi e le vedette

Gli alberi sono stati realizzati rapidamente scalando un cilindro prima sull'asse delle z,poi su quello delle y per dare snellirne l'estremità. Come per le aste

del ponte, è stata aggiunta una sfera in cima. Alla base degli alberi sono stati inseriti dei semplici supporti cilindrici per mostrare l'ancoraggio al ponte della nave. Anche le vedette sono state modellate a partire da un cilindro, estruso e scalato per ricreare i punti in cui si restinge e si allarga. La faccia superiore centrale è stata mossa verso il basso sull'asse delle z per realizzare lo spazio all'interno dell'oggetto.

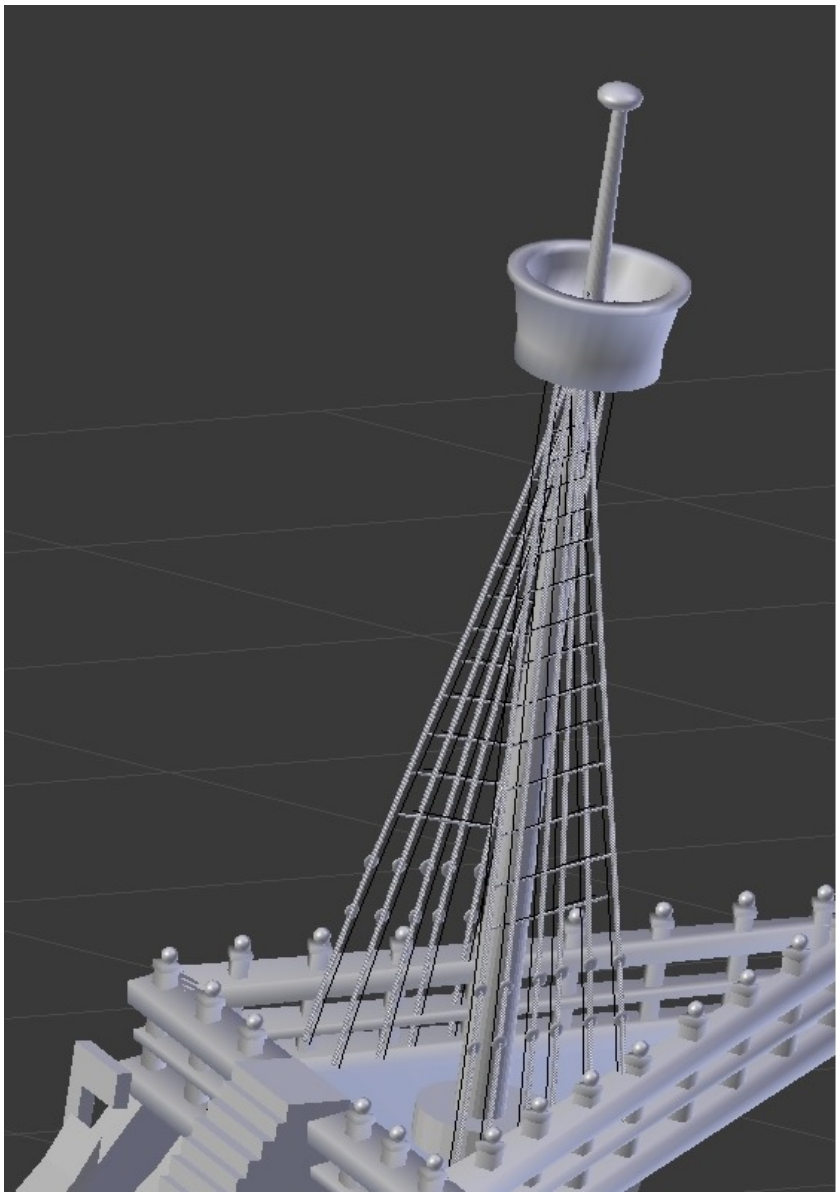

Figura 5.24: Esempio di albero e vedetta

5.4.6 Il mare

Modellare il mare è stato semplice e divertente in quanto Blender mette a disposizione un Modifier ad hoc per questo genere di oggetti: "Ocean". Applicato ad un oggetto geometrico come il *"Plane"*, il modificatore permette di agire su una serie di parametri che vanno a modificare la superficie del piano, donandogli l'aspetto della superficie di un vero e proprio mare. I parametri a cui bisogna fare riferimento sono:

- **"Geometry"** : definisce il modo con cui viene simulato il mare ed il suo comportamento; può essere di tipo *"generate"* o *"displace*. Utilizzando la prima geometria, il simulatore crea delle tiled mesh che corrispondono alla risoluzione dei dati e che vanno a sostituire la geometria già esistente, cioè l'oggetto iniziale, con una griglia. Con la seconda geometria, invece, si usa il piano per la creazione di vertici che vengono poi spostati sull'asse verticale. In questo progetto si è preferito l'utilizzo della geometria *"generate"* che ha permesso una lavorazione più immediata e più semplice.

- **"Depth"** : va a modificare la tipologia di mare simulato e la velocità delle onde. Per valori bassi del parametro si rappresentano acque basse, aventi onde più piccole e velocità minore.

- **"Resolution"** : è quello che va ad influenzare maggiormente la qualità e i costi, di conseguenza la velocità, della simulazione. Questo parametro, infatti, indica la risoluzione della griglia 2D generata dalla simulazione, cioè la dimensione dei dati contenuti in essa, indicata come potenza del due la cui base è il valore del parametro. Ovviamente, la qualità della simulazione e il costo computazionale sono direttamente proporzionali al valore di tale parametro.

- **"Choppiness"** e **"Scale"**: sono due dei parametri più importanti per la definizione delle onde e, quindi, per una loro realistica e plausibile simulazione. Il primo permette, assumendo valori nell'intervallo [0,1], di modificare lo spostamento delle onde e i loro picchi. Blender pone un

limite superiori pari a quattro, tuttavia per valori superiori a uno in combinazione con la scala si possono generare buchi nella mesh. Per choppiness pari a zero, il mare viene spostato lungo l'asse verticale, mentre per valori maggiori, vengono simulati anche gli spostamenti relativi agli assi orizzontali. Il secondo, invece, modifica l'ampiezza delle onde, andando a scalare tutti gli aspetti della simulazione dallo spostamento sugli assi alla schiuma del mare.La combinazione di choppiness e scala permette la simulazione di mari relativamente calmi a mari molto mossi.

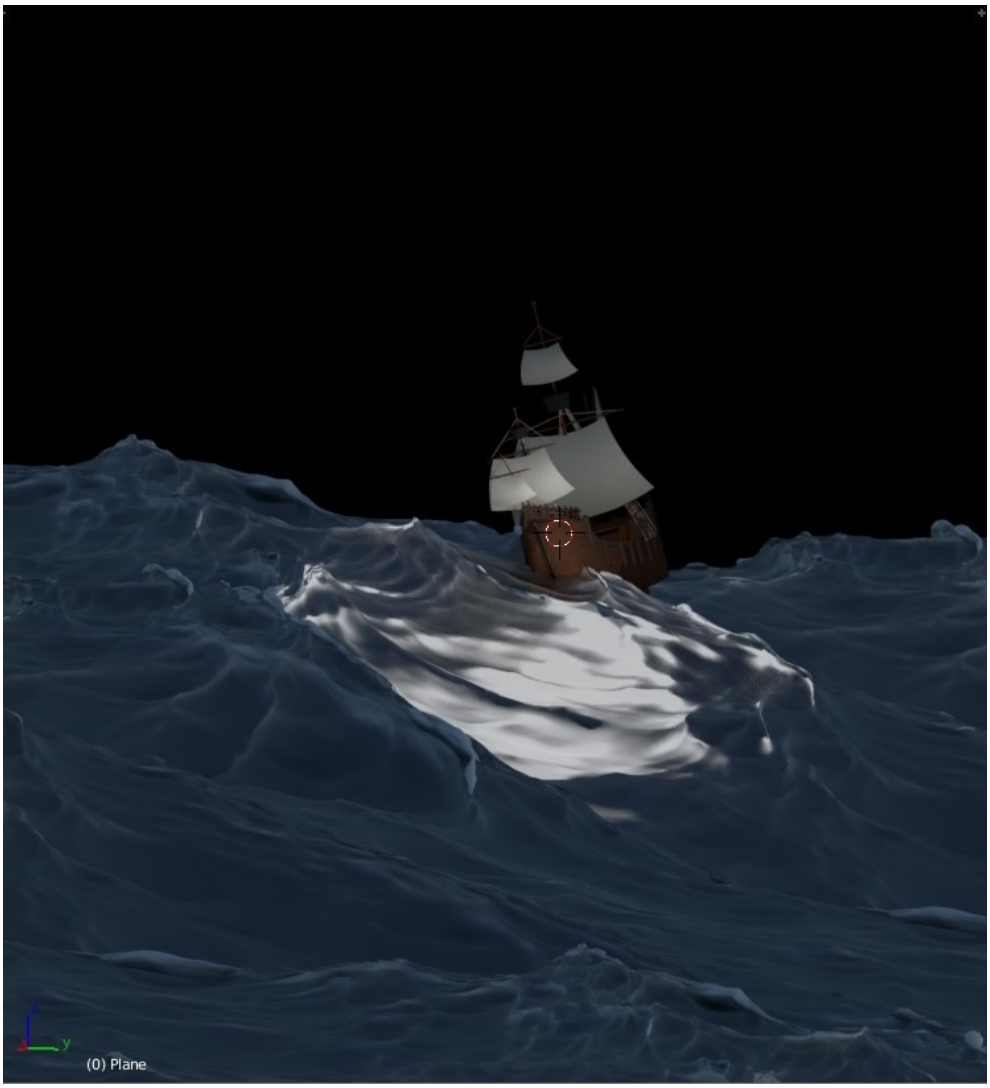

Figura 5.25: Esempio di mare

5.5 Texturing

Poiché tutti gli elementi del modello sono stati realizzati separatamente senza effettuare alcun join sul padre, l'operazione di texturing del modello è risultata molto rapida e semplice. Il texturing, come già anticipato in uno dei capitoli precedenti, consiste nell'apporre delle immagini alle mesh in modo da colorarle; il menù che se ne occupa è l'**UV Editor**.

5.5.1 Reperimento delle immagini

Per prima cosa, sono state raccolte tutte le texture necessarie a colorare i modelli, prevedendo quindi anche le diverse colorazioni delle vele. Le principali fonte di materiale sono state:

- **Textures.com** (https://www.textures.com/)
- **Blender Cloud** (https://cloud.blender.org/p/textures/)
- **TextureMate** (http://www.texturemate.com/category/key-words/blender)

Tutti i siti contengono un pool di texture gratuite ed utilizzabili senza limiti di licenza; in tutto ne sono state scaricate circa una decina.

5.5.2 UV Mapping

Prima di applicare le texture dall'apposita interfaccia utente è necessario realizzare l'UV Mapping di tutte le parti del modello. Questa pratica consiste nello "scucire" la mesh evidenziandone tutte le facce in modo da poterle "mappare" sulla texture desiderata. Blender mette a disposizione diverse tipologie di UV Mapping, ma per la realizzazione dei modelli di questa tesi sono stati impiegati:

- **Mapping a mano libera** che consiste nel selezionare le facce nell'ordine stabilito dal modellatore e poi mapparle direttamente sulla texture;
- **Smart UV**, una sorta di mapping automatizzato ed intelligente effettuato dal programma stesso.

In entrambi i casi, si accede al menù du Unwrap premendo il tasto "U" sulla tastiera dopo aver selezionato la mesh/le facce.

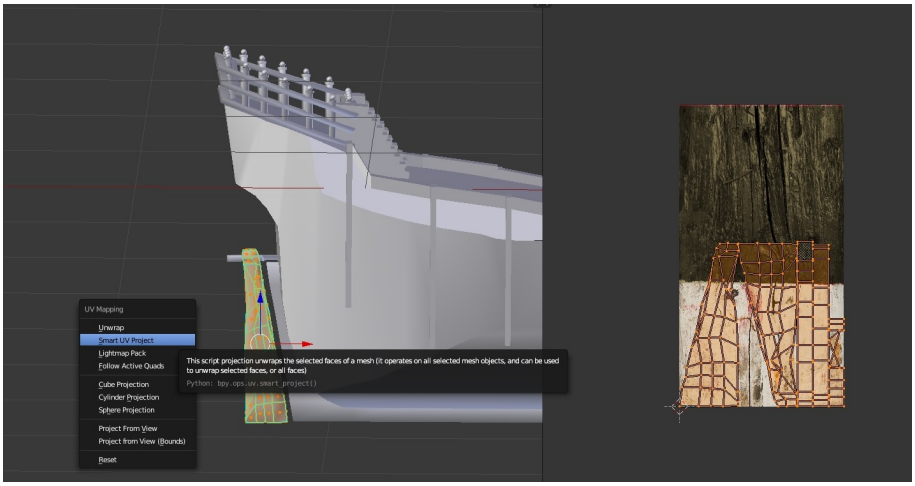

Figura 5.26: Esempio di UVMapping applicato alla pala del motore

5.5.3 Applicazione delle texture

Prima di procedere, occorre caricare le texture di interesse in modo da renderle disponibili all'interno dell'editor: si accede tramite *UV Editor* alla navigazione del file system e si selezionano le immagini. La texture, ora, può essere applicata sul modello semplicemente scegliendola dall'elenco. A questo punto, il modello può essere osservato anche tramite la Viewport Textured.

5.5.4 Assegnazione dei materiali

Poiché l'obiettivo finale di questa tesi prevede la realizzazione di immagini opportunamente renderizzate, occorre settare anche i *Materials* delle mesh in quanto in Cycles Render non è possibile visualizzare le texture. I Materials sono elementi più complessi rispetto alle texture perché non si limitano a "dipingere" gli oggetti a cui vengono applicati, ma includono anche proiezioni di luce che ne determinano la superficie opaca, traslucida o riflettente. Un materiale può essere creato e poi gestito tramite il Node Editor; una volta assegnato il material a ciascuna parte del modello è possibile osservare l'aspetto che avrà

in fase di rendering: basta spostarsi dalla Viewport principale (Solid) a quella Rendered.

5.5.5 Applicazione dello sfondo

Ora che i modelli (galeone e mare) sono stati realizzati è possibile dedicarsi allo sfondo della scena: precedentemente, sono state scaricate delle immagini di background in HD dal sito *Deviantart* (https://www.deviantart.com/ photography/nature/sky/) rappresentanti cieli con grandi nuvole, il più possibile somiglianti agli sfondi dei quadri di Aivazovsky. Per inserirle nella scena è necessario ancora una volta ricorrere al Node Editor impostando questa configurazione di nodi e caricando dal file system l'immagine desiderata. Congiuntamente, occorre eliminare dal pannello delle *Properties* la prima immagine di background, quella utilizzata come linea guida durante la fase di modelling e verificare che la voce Transparent nel menù di Render non sia spuntata.

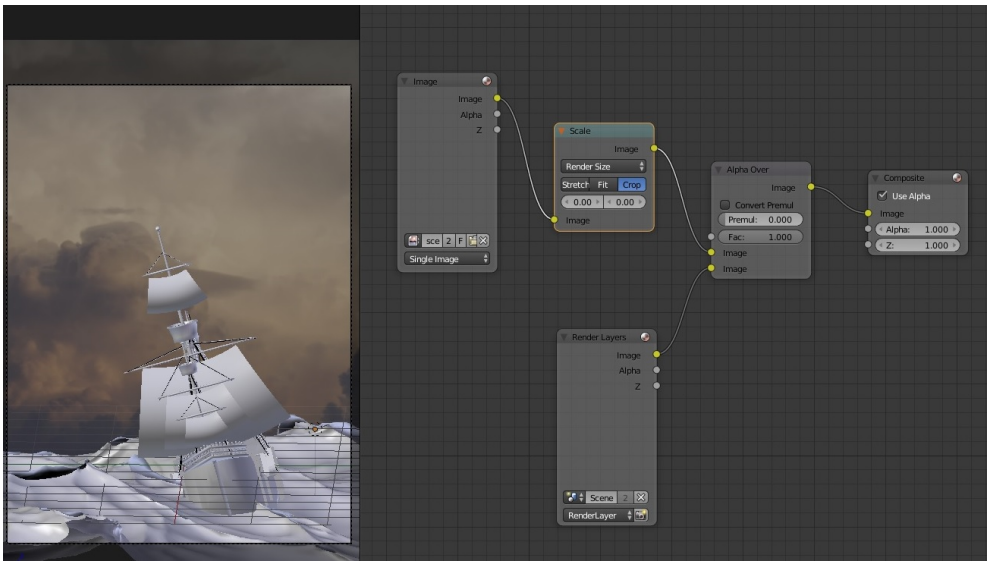

Figura 5.27: Inserimento dell'immagine di sfondo tramite nodi

5.6 Rendering

La fase di Rendering è quella che mette più a dura prova le componenti fisiche utilizzate durante l'intero iter di modellazione, in quanto essa cerca di sfruttare al massimo le potenzialità della GPU in modo da accelerare i tempi.

5.6.1 Posizionamento della Camera

Prima di renderizzare è necessario regolare la posizione e l'ampiezza della vista dell'oggetto "Camera" in quanto essa rappresenta il piano di viewport che verrà effettivamente realizzato per la scena.

5.6.2 Illuminazione

Allo stesso modo, si posizionano più entità luminose di tipo "Point" in modo da creare giochi di luci e ombre per tutta la scena. Esse sono luci omnidirezionali, quindi proiettano la luminosità in ogni direzione, che dispongono di un selezionatore di falloff per la gestione dell'ombreggiatura.

5.6.3 Denoising

Il Denoising è quella pratica che permette di eliminare il *"rumore"* causato dalle luci nella scena, ovvero quella specie di tremolio, talvolta definito come "fireflies", di tipo puntiforme che infesta alcuni tiles dell'immagine. Il metodo adottato per cercare di ridurlo al minimo senza però allo stesso tempo dilatare troppo i periodi di attesa per il rendering è stato quello di aumentare il numero di frame e sampling all'interno del menù di Render. Così facendo, l'immagine ha guadagnato in nitidezza richiedendo in cambio solo cinque o al massimo otto minuti in più di rendering.

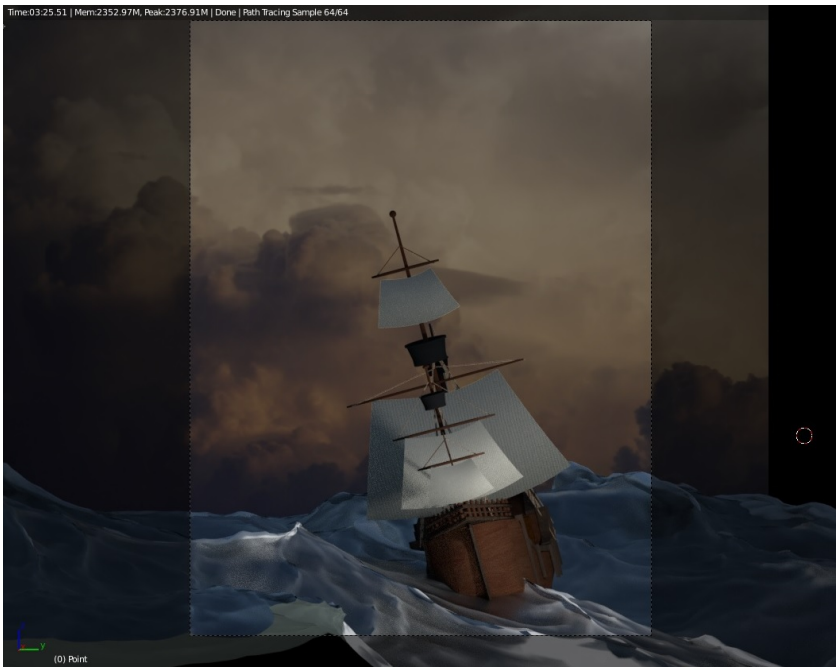

Figura 5.28: Esempio di denoising a 64 samples preview e 50 render

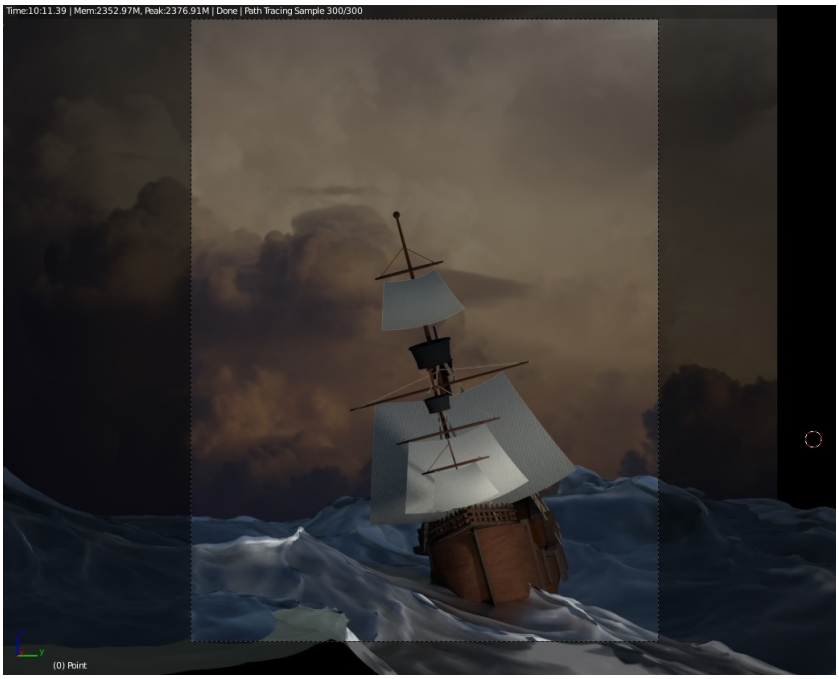

Figura 5.29: Esempio di denoising a 300 samples preview e 300 render

5.7 Scene ultimate

In questa sezione vengono allegate tutte e cinque le immagini prodotte a seguito della fase di rendering.

5.7.1 Scena 1 - "Ship in the Stormy Sea"

Figura 5.30: Scena 1 - "Ship in the Stormy Sea"

5.7.2 Scena 2 - "Twelve Apostles"

Figura 5.31: Scena 2 - "Twelve Apostles"

5.7.3 Scena 3 - "Ship at Moonlight"

Figura 5.32: Scena 3 - "Ship at Moonlight"

5.7.4 Scena 4 - "Ship Details"

Figura 5.33: Scena 4 - "Ship Details"

5.7.5 Scena 5 - "Sea"

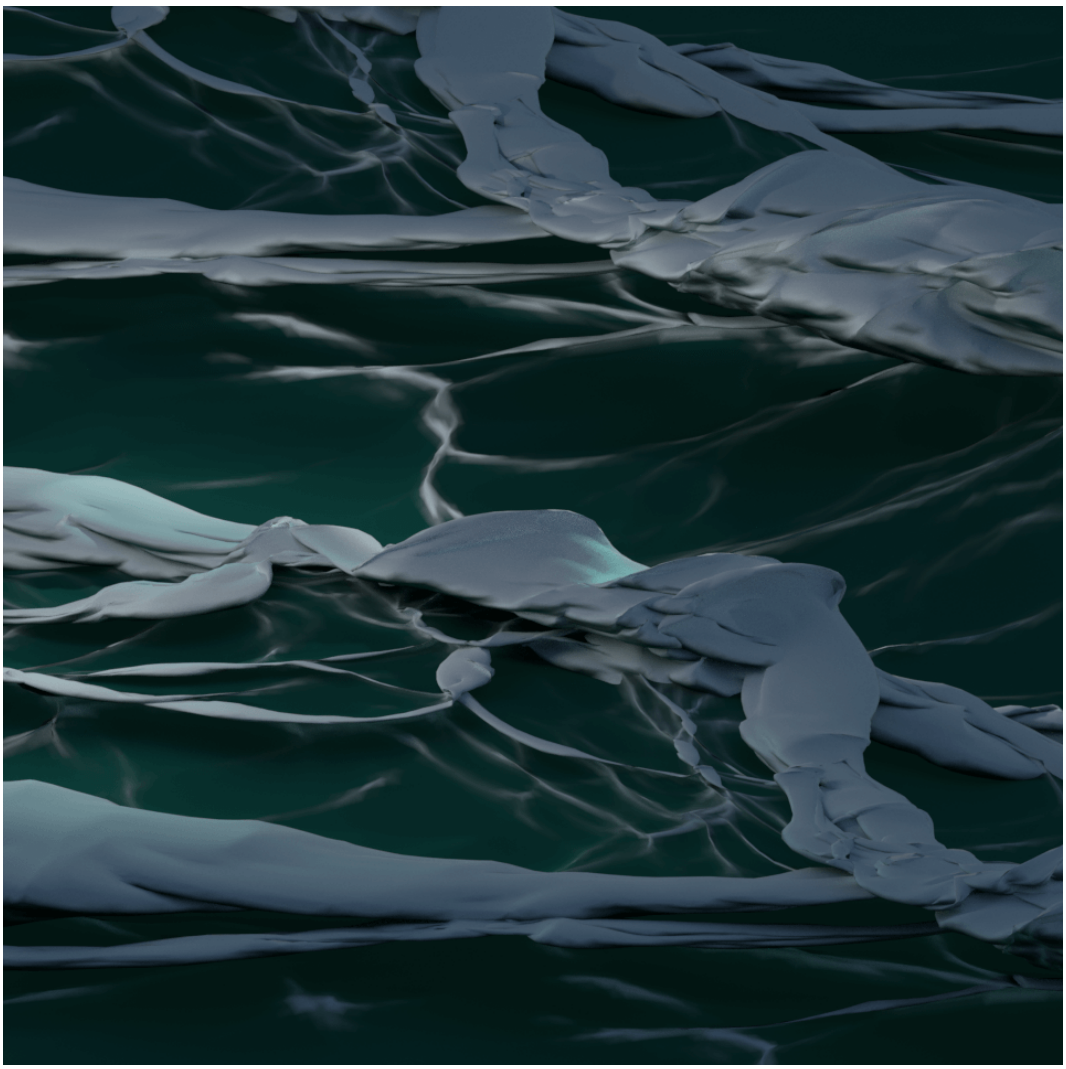

Figura 5.34: Scena 5 - "Sea"

Capitolo 6

Sinergia

La sinergia, termine greco che significa "lavorare insieme", in generale può essere definita come l'interazione di due o più agenti che lavorano insieme per produrre un risultato non ottenibile singolarmente. Il concetto è applicato in tutte le discipline, dalla biologia alla sociologia. Questa tesi è l'esempio di come essa sia applicabile anche in ambito informatico per raggiungere un obiettivo più grande che si colloca fuori dal mondo digitale e si ritaglia spazio in quello reale. Nel capitolo viene spiegato, passo dopo passo, il congiungimento di tutte le risorse impiegate nel progetto.

6.1 Preparazione di Vuforia

Per poter inserire elementi utili alla realizzazione di applicazioni di Realtà Aumentata in Unity è necessario preparare l'ambiente all'integrazione con l'*SDK di Vuforia*. Per fare questo, occorre, innanzitutto - se non è già stato fatto in precedenza e se si usa una versione precedente di Unity 3D - effettuarne il download. Successivamente, si procede con la creazione di un nuovo progetto Unity di tipo 3D e si inseriscono nella scena il Vuforia Game Object principale, ossia la **AR Camera**. Inoltre è necessario attivare Vuforia dal **Player Settings** spuntando l'opzione *"Vuforia Augmented Reality Supported"* presente nell'Inspector sotto la voce **XR Settings**.

6.1.1 License Key

Per poter concretamente sviluppare questo genere di applicazioni occorre, però, inserire la License Key del progetto, reperibile presso il *Vuforia Developer Portal* tramite il *License Manager*. La licenza è unica per ogni progetto e va inserita nell'Inspector di Unity all'interno delle *Vuforia Configuration*, voce *App License Key*.

6.1.2 Marker Database

L'altro elemento fondamentale allo sviluppo è il Database contenente i marker registrati e digitalizzati tramite l'apposito *Target Manager*, raggiungibile sempre tramite il *Vuforia Developer Portal*[30].

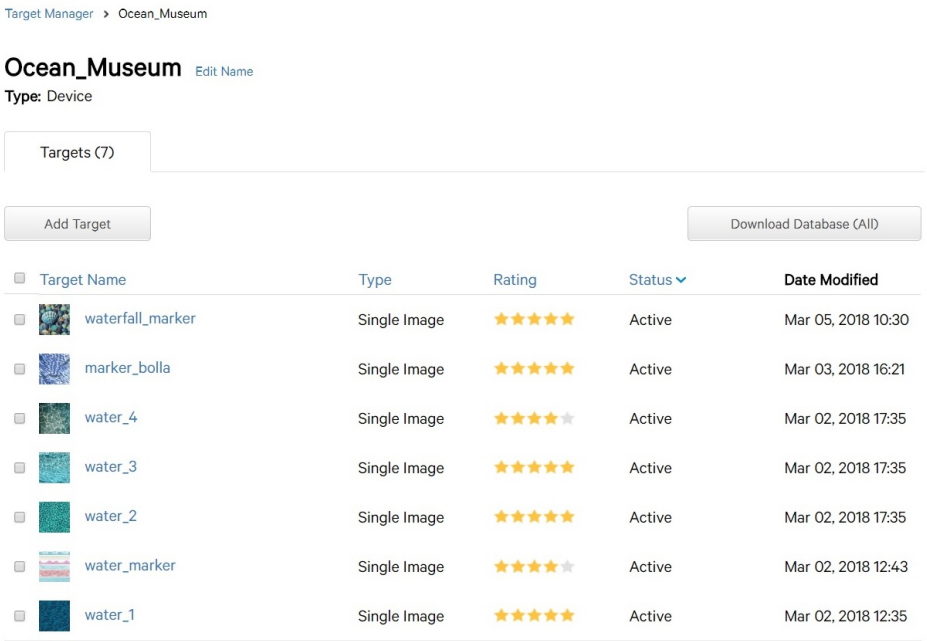

Figura 6.1: Marker Database view con star rate

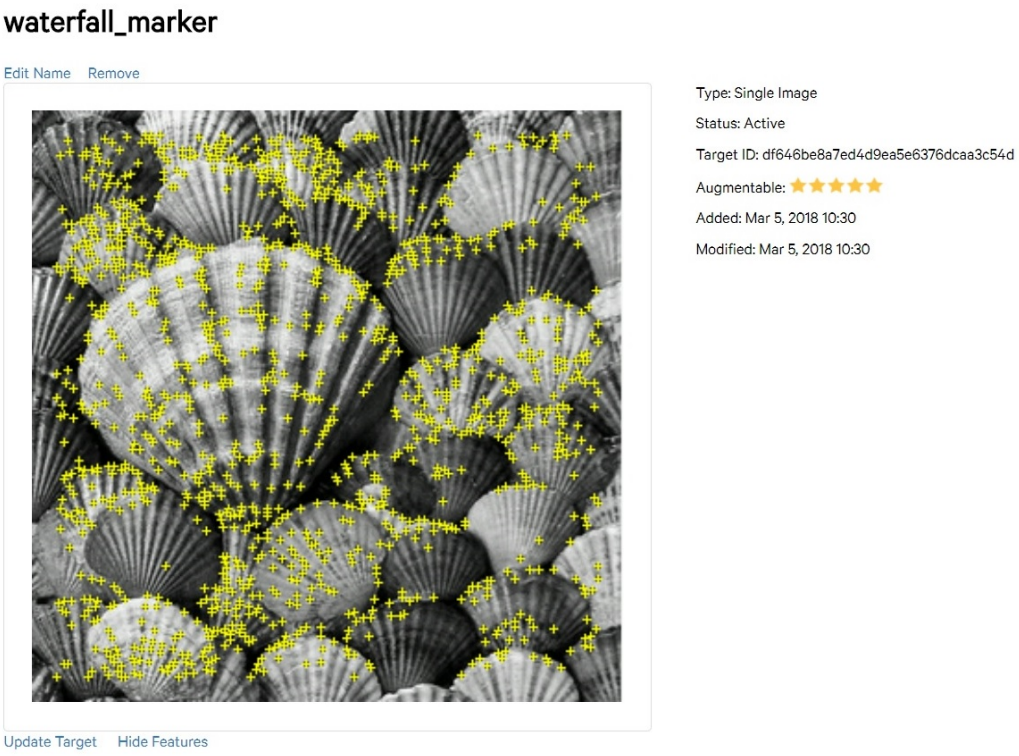

Figura 6.2: Esempio di analisi di un marker rappresentante le sue features

Il DB, una volta scaricato selezionando accuratamente l'ambiente di destinazione, contiene tutte le immagini scelte come marker e i loro metadati. Come la licenza, deve essere inserito all'interno di Unity e per farlo basta importarlo come se fosse un nuovo Asset, poi lo si attiva inserendo una spunta nelle Vuforia Configuration. A questo punto, è possibile inserire nella scena i marker come se fossero oggetti del tipo *"Vuforia Image"* ed è possibile scegliere l'immagine del target semplicemente scegliendolo dal menù a tendina nell'Inspector.

6.2 Preparazione dei modelli

Per poter aggiungere i modelli 3D creati in Blender alla scena in Unity occorre innanzitutto esportarli dal motore grafico ed importarli in quello di gioco.

6.2.1 Esportazione in un formato accettabile

Non tutti i formati disponibili sono accettati, quindi occorre scegliere con cura l'opzione più adatta, che nel caso di questa tesi - ma in generale nel caso di tutti i progetti con Unity come ambiente di destinazione - si tratta del formato **.FBX** (Filmbox). Per fare questo basta selezonare, da Blender, il menù *File -> Export -> .FBX* e scegliere il nome del nuovo file e la destinazione. Si può decidere anche di esportare solo il modello senza Camera e senza Luci, basta deselezionarle dal menù in basso a destra prima di confermare l'export.

6.2.2 Sistemazione dei Materiali

Poiché ancora non è supportato l'export diretto del modello comprensivo di texture e materials occorre sistemare il tutto a mano dall'interno di Unity. Nel caso specifico di questo progetto è stato sufficiente rimappare tutte le texture utilizzate sui rispettivi Materials dopo aver impostato il modo in cui reperirli, ovvero selezionando le opzioni:

- Use external materials (Legacy)
- Model name + Model's Material
- Recursive-Up

6.2.3 Scripting

La parte di Scripting presenta semplici comportamenti che non necessitano di grandi spiegazioni in quanto si occupano di effettuare rotazioni e traslazioni dei modelli, sia per ciò che riguarda i modelli a cui sono associati dei Virtual Button, sia per i modelli che eseguono una semplice animazione. Tutti utilizzano le funzioni base *transform.localPosition* e *transform.Rotate*.

```
using UnityEngine;

public class MoveShipAround : MonoBehaviour
{
    public float timeCounter = 0;
    public float delta = 1f;
    public float radius;
    public float centerx = -0.25f;
    public float centerz = 0.25f;
    public float angle_offset = 105;

    // Initialization
    void Start()
    {
        radius = 0.4f;
    }

    // Update is called once per frame
    void Update()
    {
        timeCounter -= delta;
        if (timeCounter <= -360)
        {
            timeCounter = 0;
        }
        float x = centerx + Mathf.Sin(Mathf.Deg2Rad * (timeCounter + angle_offset)) * radius;
        float z = centerz + Mathf.Cos(Mathf.Deg2Rad * (timeCounter + angle_offset)) * radius;
        transform.localPosition = new Vector3(x, transform.localPosition.y, z);
        transform.Rotate(new Vector3(0, -delta, 0));

    }
}
```

Figura 6.3: Script dell'animazione - C#

```
using UnityEngine;
using Vuforia;

public class RotateShip : MonoBehaviour, IVirtualButtonEventHandler
{
    private GameObject VB;
    private GameObject ship;
    public bool rotate = false;

    // Use this for initialization
    void Start()
    {
        VB = GameObject.Find("VB");
        VB.GetComponent<VirtualButtonBehaviour>().RegisterEventHandler(this);
        ship = GameObject.Find("BoxShip");
    }

    public void OnButtonPressed(VirtualButtonBehaviour vb)
    {
        Debug.Log("Button Pressed");
        rotate = true;

    }

    public void OnButtonReleased(VirtualButtonBehaviour vb)
    {
        Debug.Log("Button Released");
        rotate = false;
    }

    public void Update()
    {
        if (rotate)
        {
            ship.transform.Rotate(new Vector3(0, -100, 0) * Time.deltaTime);
        }
    }
}
```

Figura 6.4: Script per ruotare la nave - C#

```
1
2  var scrollSpeed_X = 0.5;
3  var scrollSpeed_Y = 0.5;
4  function Update() {
5  var offsetX = Time.time * scrollSpeed_X;
6  var offsetY = Time.time * scrollSpeed_Y;
7  GetComponent.<Renderer>().material.mainTextureOffset = Vector2 (offsetX,offsetY);
8  }
```

Figura 6.5: Script per lo scroll della texture - JavaScript

6.2.4 Associazione Marker-Model

La parte di associazione Marker-Model è estremamente semplice da realizzare in quanto è sufficiente scegliere il modello 3D che si vuole far apparire, scalarlo in modo opportuno se le dimensioni non sono quelle desiderate, posizionarlo sopra al marker ed infine renderlo "figlio" dell'oggeto ImageTarget utilizzando il pannello a sinistra che rappresenta la struttura del progetto.

6.3 Build su un dispositivo Android

Per poter eseguire l'applicazione su un dispositivo Android[31] occore effettuare una banale procedura di scelta della piattaforma selezionando dal menù *File* l'opzione *Build Settings*; dalla schermata che appare basta scegliere la piattaforma di interesse - in questo caso Android - ed impostare le API minime del dispositivo, poi fare click su *Switch Platform*. A questo punto, per far girare l'applicazione su uno smartphone Android basta collegare il dispositivo al computer e avviare la build: dopo un paio di minuti il software è pronto per essere eseguito.

6.4 Allestimento del Museo Aumentato

La simulazione della stanza del "Museo Aumentato" è stata effettuata all'interno di un ambiente ampio, luminoso e dai toni un po' freddi. Sono stati disposti alcuni mobili scuri per creare contrasto su cui sono stati sistemati i marker avvolti e protetti da cornici dal carattere minimalista. Sono stati inseriti oggetti

"di scena", vale a dire conchiglie di grandezze e tipologie diverse, una bottiglia contenente una barca di vetro e una specie di ampolla dal profilo rotondeggiante volta a contenere uno dei marker. Alle pareti sono stai appesi i quadri raffiguranti le cinque scene realizzate in blender, precedentemente stampati su fogli di dimensione A3.

6.5 Ocean Museum - l'applicazione AR conclusa

Tutti gli elementi trattati finora hanno permesso la realizzazione di un'esperienza di Realtà Aumentata concretizzata in un'applicazione.

6.5.1 Logo

Il logo - l'icona - è costituito due linee curve bianche rappresentanti le onde del mare e da un triangolo dello stesso colore che rappresenta la vela di una barca. Lo sfondo è di un colore azzurro chiaro tendente al pervinca (il suo esadecimale è *85abee*). La realizzazione è stata effettuata tramite *Android Asset Studio - Icon Launcher Generator* (https://romannurik.github.io/AndroidAssetStudio/icons-launcher.html), un tool gratuito online che permette di creare l'icona di un'applicazione in tutte le dimensioni supportate dagli schermi dei dispositivi.

Figura 6.6: Logo dell'applicazione

6.5.2 Foto

In questa sezione verranno inserite alcune foto del *Mondo Aumentato* al fine di rendere più completa ed esplicativa la trattazione dell'argomento.

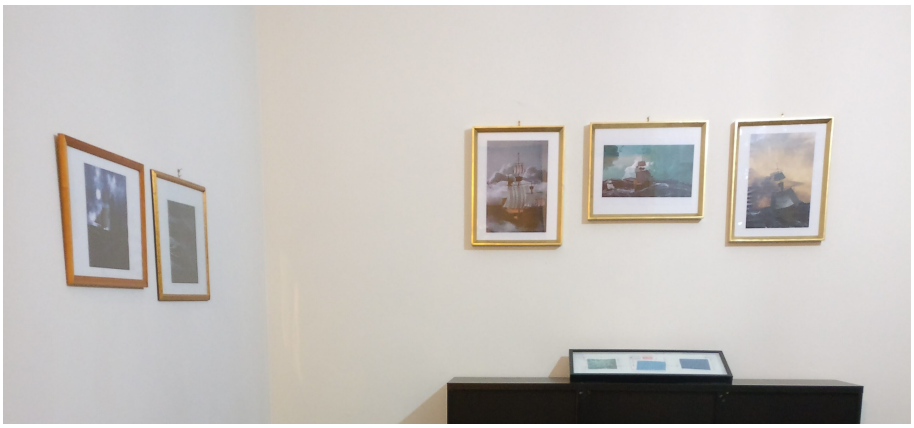

Figura 6.7: Quadri allestiti

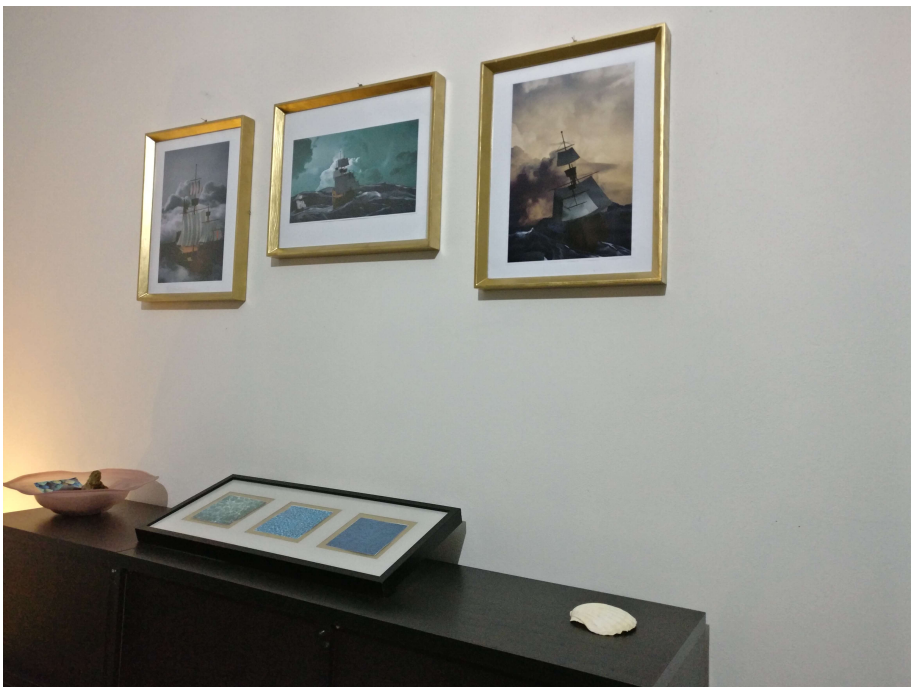

Figura 6.8: Museo

Figura 6.9: Vista frontale con quadri ed ampolla di vetro

Figura 6.10: Quadro aumentato

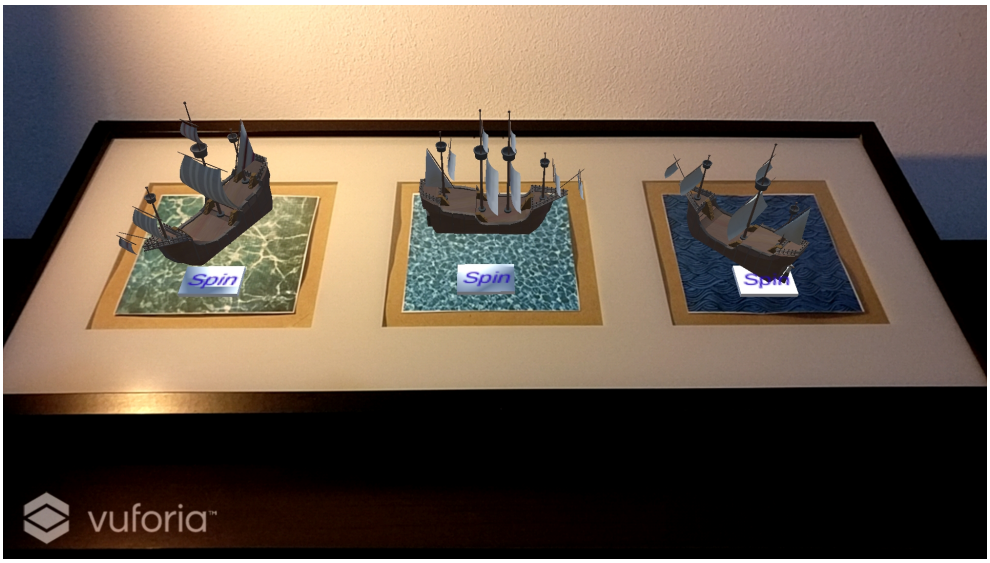

Figura 6.11: Navi aumentate

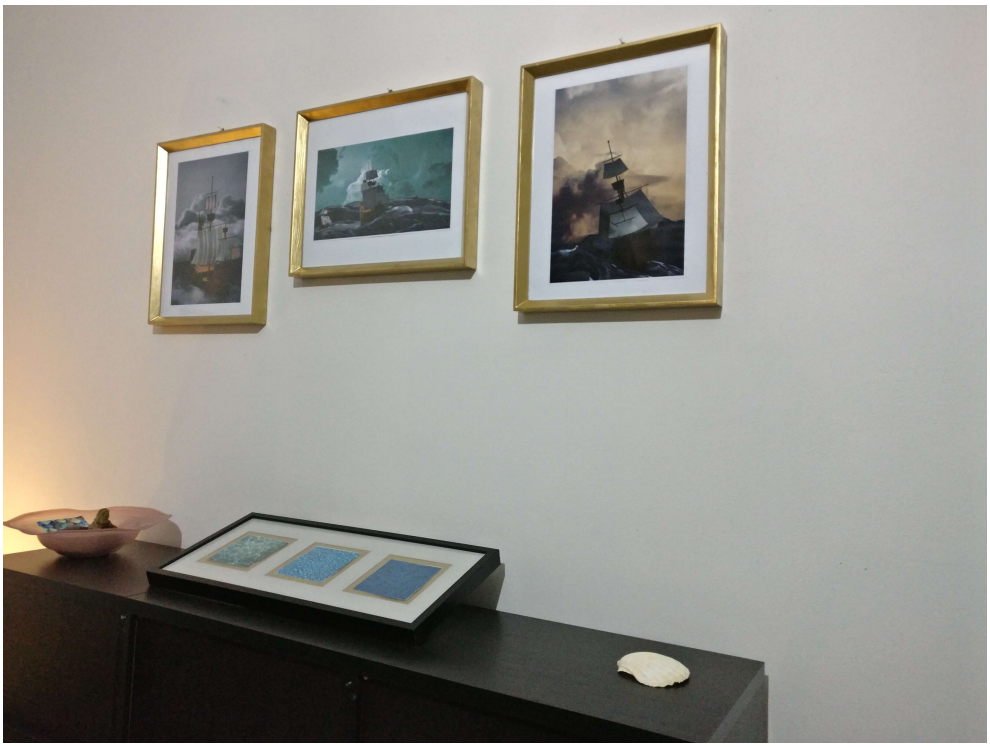

Figura 6.12: Parete frontale con quadri e marker

Figura 6.13: Ampolla con nave aumentata - device view

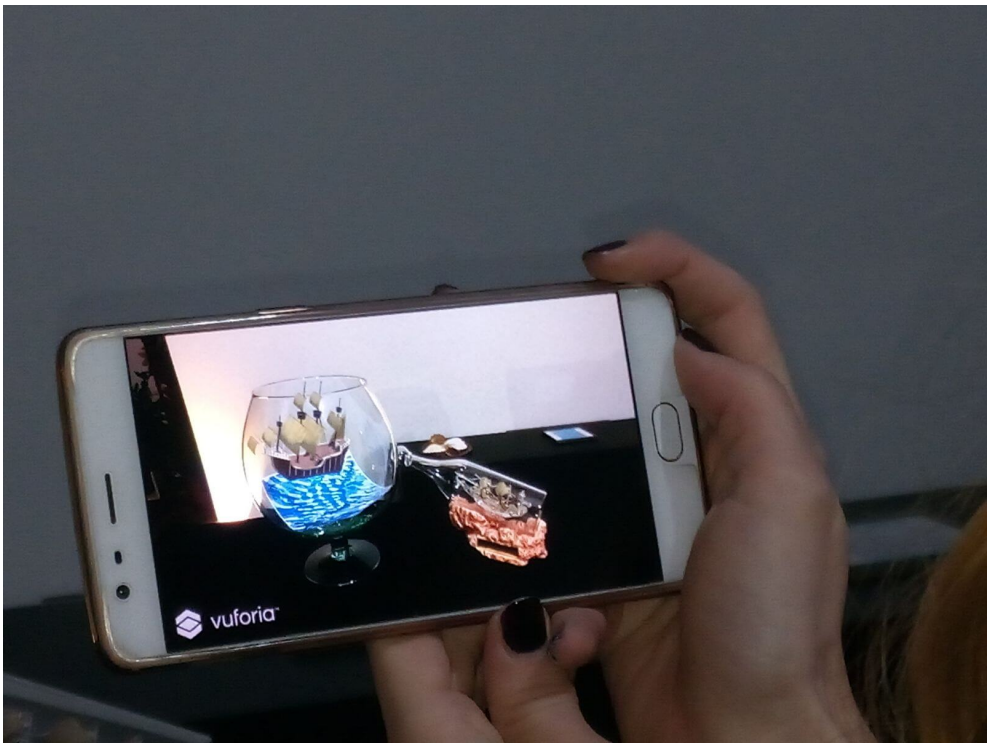

Figura 6.14: Ampolla con nave aumentata - device handled

Figura 6.15: Donna al museo

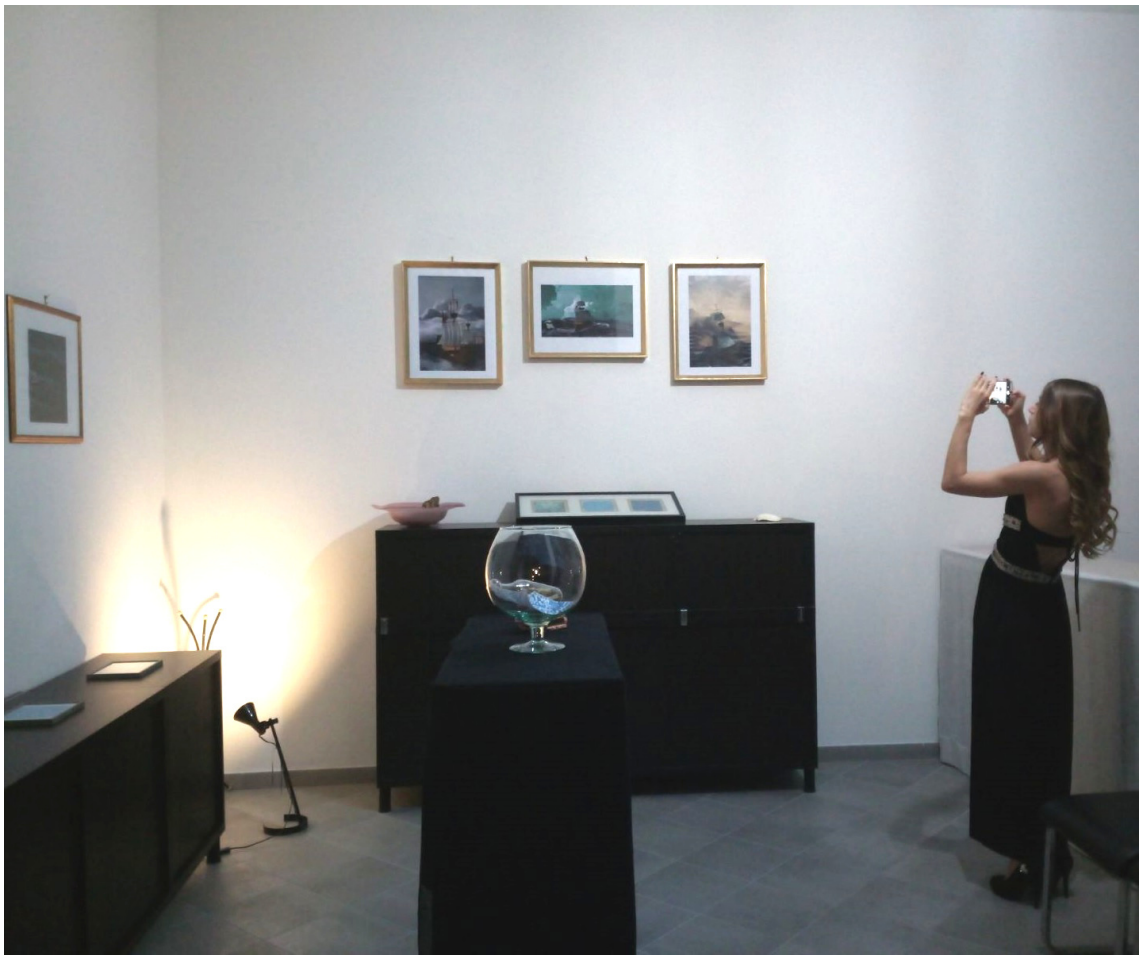

Figura 6.16: Donna al museo - con device

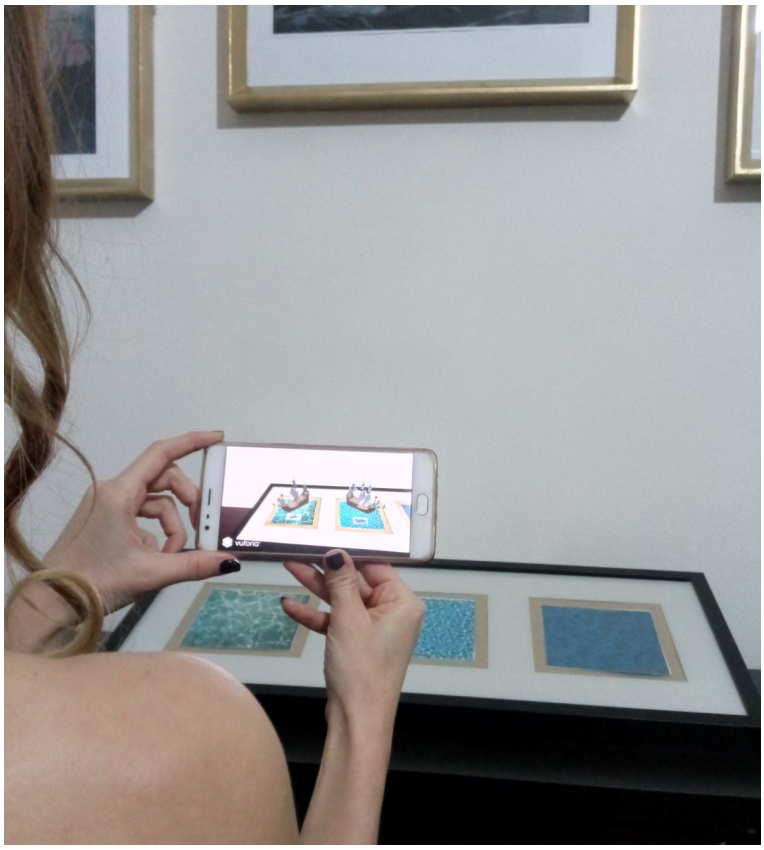

Figura 6.17: Navi aumentate - device handled

6.5.3 Video completo

Il video è stato realizzato filmando la scena e registrando direttamente lo schermo del dispositivo. La scelta di sperimentare entrambe le opzioni è stata presa in quanto l'obiettivo ultimo era quello di mostrare l'esperienza di Realtà Aumentata in prima e terza persona. Per registrare lo schermo è stata scelta l'applicazione *AZ Screen Recorder* in quanto tutte le riprese sono state fatte con un cellulare. Per il montaggio, invece, si è utilizzato il programma *Free Make* (http://www.freemake.com).

Il video completo è reperibile al seguente indirizzo: https://www.youtube.com/channel/UCxB-ETpn1pYDa6i6MnqyFow

Capitolo 7

Conclusioni

Il risultato finale di questo progetto attesta con chiarezza che il mondo informatico, pur essendo composto principalmente da dati numerici in formato digitale, è in grado di sposarsi con il mondo artistico-culturale andando a formare un connubbio meraviglioso avvolto in quella che sembra essere una leggera aura di magia e incredulità.

Esperienze come quella qui simulata rappresentano un futuro che non è troppo lontano dal nostro presente, ma che al contrario si appresta a diventare parte integrante della nostra quotidianità. Già oggi, sparse per il mondo, esistono piccole e grandi sfere in cui si utilizzano sinergicamente elementi virtuali e reali volti alla realizzazione di esperienze nuove.

Musei, scuole ed università, ospedali, strutture alberghiere, fabbriche, industrie, siti archeologici, luoghi di interesse, ristoranti, negozi: tutte queste realtà sono ricche di potenziale che può essere ampiamente sviluppato ed accresciuto grazie alla tecnologia dell' Augmented Reality.

Sta a noi cogliere l'occasione e, come Demiurghi, plasmare a nostro piacimento la materia fondendo insieme reale e virtuale.

Bibliografia

[1] Jonathan Steuer. *Defining Virtual Reality: Dimensions Determining Telepresence.* Stanford University, 1993.

[2] J; Biocca F; Hamza-Lup F; Yanggang H; Martins R Rolland. *Development of Head-Mounted Projection Displays for Distributed, Collaborative, Augmented Reality Applications.* Massachusetts Institute of Technology, 2005.

[3] Rachel Metz. *Augmented Reality Is Finally Getting Real.* Stanford University, 2012.

[4] Kerry Maxwell. *Augmented Reality.* Macmillan Dictionary Buzzword, 2010.

[5] Steve Mann. *GlassEyes: The Theory of EyeTap Digital Eye Glass.* IEEE Technology and Society, 2012.

[6] Kelly Sheridan. *Microsoft HoloLens Vs. Google Glass: No Comparison.* InformationWeek, 2013.

[7] Larry Greenemeier. *Computerized Contact Lenses Could Enable In-Eye Augmented Reality.* Scientific American, 2011.

[8] Sebastian Klepper. *Augmented Reality – Display Systems.* Romanelli and Feldman, 2013.

[9] Ariel Borge. *The story behind 'Pokemon Go's' impressive mapping.* Mashable, 2011.

[10] Roland Hess. *Blender Foundations: The Essential Guide to Learning Blender 2.6*. Focal Press, 2010.

[11] Pierre-Armand Nicq Romain Caudron. *Blender 3D By Example*. Packt, 2014.

[12] Brian R. Kent. *3D Scientific Visualization with Blender (Iop Concise Physics)*. Morgan and Claypool, 2017.

[13] Oliver Villar. *Learning Blender: A Hands-On Guide to Creating 3D Animated Characters*. Addison-Wesley Professional, 2017.

[14] John M Blain. *The Complete Guide to Blender Graphics: Computer Modeling and Animation, Fourth Edition*. A K Peters CRC Press, 2017.

[15] Blender org. `https://www.blender.org/`.

[16] Blender stack exchange. `https://blender.stackexchange.com/`.

[17] Blender italia. `https://www.blender.it/`.

[18] Blender today. `https://blender.community/`.

[19] Unity. `https://unity3d.com/`.

[20] Patrick Felicia. *Unity From Zero to Proficiency (Foundations): A step-by-step guide to creating your first game with Unity)*. Independently published, 2018.

[21] Jonathan Linowes. *Unity Virtual Reality Projects: Explore the world of virtual reality by building immersive and fun VR projects using Unity 3D*. Packt Publishing, 2015.

[22] Mike Geig Ben Tristem. *Unity Game Development in 24 Hours, Sams Teach Yourself*. Sams Publishingd, 2015.

[23] Unity asset store. `https://store.unity.com/`.

[24] Joe Hocking. *Unity in Action: Multiplatform Game Development in C# with Unity 5*. Manning Publications, 2015.

[25] Vuforia. `https://www.vuforia.com/`.

[26] Art Masters. *Ivan Aivazovsky*. Createspace Independent Pub, 2016.

[27] Ivan Samarine. *Light, Water and Sky: The Paintings of Ivan Aivazovsky*. Laurence King Pub, 2013.

[28] Museo della regina di cattolica. `http://www.cattolica.info/itinerari/itinerario-turistico-di-cattolica/museo-della-regina-a-cattolica/`.

[29] Vele di famiglia. `http://www.cattolica.net/retecivica-citta-di-cattolica/sites/default/files/statici/museo/veledifamiglia.pdf`.

[30] Vuforia developer. `https://developer.vuforia.com/`.

[31] Building unity and vuforia app on android. `https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity-2017-2-beta.html`.

Capitolo 8

Ringraziamenti

Dopo tanto tempo e tanti sforzi, finalmente è arrivato il mio giorno; scrivere il capitolo sui *ringraziamenti* è il tocco finale di questa tesi e la prova che davvero sono riuscita a farcela, a completare un percorso di studi meraviglioso e stimolante, nonostante le perplessità di chi all'inizio mi ha visto intraprendere questa strada. Si è trattato di un periodo molto intenso, ricco di picchi emozionali, nuove idee e sfide, sia a livello pratico sia a livello personale. Vorrei quindi ritagliare uno spazio in cui ringraziare tutte le persone che mi hanno sostenuto ed aiutato in questo folle periodo.

Innanzitutto, vorrei ringraziare la professoressa **Damiana Lazzaro**, relatore di questa tesi di laurea, per l'aiuto fornitomi in tutto questo tempo e la grande conoscenza che mi ha donato, ma soprattutto per la disponibilità e precisione dimostratemi durante il periodo di stesura e realizzazione del progetto. Senza di Lei questo lavoro non avrebbe preso vita!

Ringrazio anche il professor **Alessandro Ricci** e il dottorando **Angelo Croatti** per avermi coinvolto nel progetto *"Porti Aumentati"*, svolto in collaborazione con il Museo della Regina di Cattolica, e per avermi dato le dritte necessarie all'approfondimento di alcune tematiche, tra cui proprio la Realtà Aumentata. Unitamente, voglio ringraziare anche *Maria Luisa Stoppioni*, direttrice, mente e cuore del Museo, per le conoscenze specifiche e tecniche for-

nitemi sulle barche del ’900 .

Vorrei ringraziare i miei genitori, **Franco** e **Maria**, per sostenermi e supportarmi da quando ho memoria e per avermi assistito come "entourage" tecnica nella realizzazione e nell’allestimento pratico del mio "Ocean Museum" .

Un ringraziamento speciale a **Pierluigi**, che da oltre tre anni e mezzo è al mio fianco: stiamo crescendo insieme e stiamo per intraprendere una nuova avventura, mano nella mano. E io non vedo l’ora!

Voglio ringraziare le mie amiche e i miei amici, in particolare **Nicole, Carolina, Martina, Marina, Giulia, Alberto** e tutti quelli del **gruppo dell’università** con cui ho condiviso i successi e i momenti di sconforto conseguenti a quest’avventura.

Infine, ringrazio il **corriere della Dell** (anche se non ci conosciamo di persona) che ha consegnato il nuovo PC di Gigi con una bellissima scheda GTX 1060 Max-Q ben 3 giorni prima della data di prevista consegna, permettendomi di lavorare con un hardware adeguato alle mie esigenze grafiche, evitando quindi la mia furia e il mio odio per il mio PC che proprio prima di iniziare a modellare ha deciso di abbandonarmi uccidendo brutalmente le prestazioni. Senza gli strumenti giusti (e un po’ di fortuna mista a karma positivo), questo progetto avrebbe fatto molta fatica a prendere il volo.

Ancora una volta, grazie a tutti voi!

Giulia Capacci