Dipartimento di Architettura e Design

Corso di Laurea Triennale in Design e Comunicazione visiva

A.A. 2020-2021

REALTÀ AUMENTATA E REALTÀ VIRTUALE PER VALORIZZARE IL PATRIMONIO MUSEALE

Esperienze presso il Museo d'Arte Orientale

Relatrice:

Roberta Spallone

Correlatori:

Fabrizio Lamberti

Luca Maria Olivieri

Claudia Ramasso

Francesca Ronco

Candidato:

Lorenzo Castagna

Desidero ringraziare la professoressa Roberta Spallone per avermi accompagnato in questo lungo percorso, per avermi dato l'opportunità di intraprendere il progetto e per la sua disponibilità. Grazie all'architetto Francesca Ronco per aver coordinato le fasi di rilievo all'interno del museo e per essere stata un costante punto di appoggio; al professor Fabrizio Lamberti per i preziosi cosigli e per avermi introdotto e fatto appassionare all'argomento con le sue lezioni lo scorso anno. Un sentito ringraziamento alla dottoressa Claudia Ramasso e al professor Luca Maria Olivieri, per le numerose consulenze e per il contributo culturale che hanno dato al progetto, e al Museo d'Arte Orientale di Torino che ha permesso di realizzare questa esperienza. Infine ringrazio Filippo Gabriele Pratticò e Davide Calandra per l'intensa sessione di introduzione alla programmazione e per avermi supportato nella risoluzione di problemi tecnici.

Si chiude finalmente un percorso che tanto mi ha dato, ma che ha richiesto altrettanto da parte mia.
Grazie a Cecilia, Alessandro e Gianluca per essere stati ottimi e costanti compagni di gruppo.
In questi anni, tra le lunghe giornate e nottate di progetti, ho diviso gli spazi domestici con Lorenzo e Lorenzo, con cui, oltre ai nomi, ho potuto condividere alcuni tra i più bei ricordi che conservo. Grazie.

Indice

Capitolo 1

Tecnologie per i beni culturali

1.1 Digitalizzare i musei

Il trascorrere del tempo ha reso evidente l'abbondanza di tecnologia in cui siamo immersi oggi rispetto a pochi anni fa.
È dato per scontato che ogni individuo debba ogni giorno interfacciarsi con un dispositivo tecnologico di qualunque genere per svariate motivazioni, quali lavoro, svago, comunicazioni o commissioni.
Allo stesso modo anche il mondo si è evoluto, passando da un ambiente di contenuti analogici ad uno digitale. Di conseguenza anche le istituzioni legate alla cultura sono interessate in questo processo di trasformazione.
È infatti in corso una fase di digitalizzazione dell'opera, sia essa di qualunque tipo, un quadro, una scultura, musica, un testo, che comporta la creazione di una sua copia non fisica ma in grado di essere letta dai computer, in maniera da essere conservata e resa successivamente accessibile sul *web*.

Se è vero che a causa della pandemia globale di Covid-19, vista l'impossibilità da parte dei visitatori di recarsi all'interno dei musei, vi è stata una notevole spinta verso la produzione di contenuti digitali (ad esempio puntando sulle visite virtuali a distanza), è vero anche che l'Italia si trova piuttosto indietro in questo settore rispetto alla media europea.
In generale l'utilizzo da parte dei musei italiani dei *social network* e la presenza su un sito web personale è una pratica diffusa, rispettivamente al 65.9% e 43.7%, mentre solo uno su dieci ha la possibilità di offrire una visita virtuale[1].
Per colmare questo vuoto è stato dunque siglato il Piano Triennale per la Digitalizzazione e l'Innovazione dei Musei[2], che si prefigge di spingere sull'utilizzo delle nuove tecnologie per migliorare i servizi al pubblico da parte delle istituzioni culturali. Tra gli obiettivi principali è citata la volontà di definire processi di digitalizzazione, per la creazione di modelli 3D, esperienze di di realtà aumentata e di *gaming*.

Oltre alla distribuzione più efficace dei contenuti al passo con tecnologie attuali, alcune modalità di fruizione possono trasformare una semplice visita museale in un'esperienza interattiva, rendendo l'utente personaggio attivo della fase di acquisizione delle informazioni.
La migrazione verso modalità ricche di interazioni, che richiamano al mondo videoludico, riescono inoltre a coinvolgere anche le fasce della popolazione più giovane, non sempre interessata agli aspetti museali della cultura.
In particolare alcuni studi condotti a riguardo evidenziano come proprio la realtà aumentata rappresenti uno degli strumenti che si stanno rivelando più incisivi nel campo educativo.
La congiunzione del mondo reale

1. Il Giornale dell'Architettura - dati Istat 2018, *I musei e la sfida della digitalizzazione, https://ilgiornaledellarchitettura.com/2021/01/25/i-musei-e-la-sfida-della-digitalizzazione,* ultima consultazione 26 agosto 2021.

2. Mibact, *Piano Triennale per la Digitalizzazione e l'Innovazione dei Musei, 2019.*

e dei contenuti digitali provenienti dal dispositivo in uso (solitamente uno *smartphone*, strumento alla portata di tutti) influenza i processi cognitivi a tal punto che l'apprendimento risulta largamente accellerato. Le attività di memorizzazione vengono infatti facilitate dalle informazioni teoriche provenienti dall'esperienza, erogate attraverso una modalità pratico-sperimentale, che, rifacendosi a una costruzione del sapere più di tipo indipendente e autonomo, ottengono priorità dalle aree cerebrali relative rispetto alla trasmissione di conoscenza tradizionale.
Analogamente la realtà virtuale "catapulta" l'utente in un mondo nuovo, coinvolgendolo totalmente. Nell'ambito della cultura questo strumento presenta diverse possibilità di applicazione: permette di entrare in una dimensione parallela, viaggiando indietro nel tempo alla scoperta di luoghi restaurati oggi non più presenti, di entrare all'interno del museo senza entrarci fisicamente.

Strumenti e tecnologie 1.2

La realizzazione di esperienze di realtà aumentata e di realtà virtuale in ambito museale richiede la conoscenza e l'approfondimento di alcuni strumenti necessari.

Tecniche di acquisizione 1.2.1

La progettazione parte innanzitutto dall'attore protagonista, ovvero il manufatto storico.
Occorre che l'oggetto museale in questione attraversi una fase di digitalizzazione, avente come risultato finale la restituzione di un modello 3D che replichi perfettamente l'opera fisica, sia in termini di forma che di aspetto. Le modalità con cui si esegue il rilievo metrico digitale sono principalmente due: mediante fotogrammetria o attraverso la scansione laser.

La strumentazione che si serve del laser può agire sia corto che a lungo raggio, arrivando a scansionare oggetti posti fino a 1000 m di distanza dall'apparecchio. Gli scanner in circolazione utilizzano la tecnologia della triangolazione laser, per cui il raggio laser viene emesso dal dispositivo e arriva fino all'oggetto, colpendolo. La riflessione e la sua conseguente cattura da parte di un sensore apposito forniscono informazioni riguardo la precisa posizione del punto interessato[3]. La moltitudine dei raggi emessi restituiscono così informazioni riguar-

3. Girelli V. A., *Tecniche digitali per il rilievo, la modellazione tridimensionale e la rappresentazione nel campo dei beni culturali,* Tesi di Dottorato, Alma Mater Studiorum - Università di Bologna, 2007, relatore Bitelli G.

do le coordinate di moltissimi punti che vengono ricostruiti in ambiente digitale, risalendo alla superficie dell'oggetto. Questo metodo, servendosi di un comportamento attivo della luce, è adatto alla scansione di varie tipologie di superfici diverse, includendo anche oggetti metallici e riflettenti.

Una modalità più economica è invece la fotogrammetria, una tecnologia che si basa sull'analisi di varie fotografie dell'oggetto scattate da molteplici punti di vista differenti. È sufficiente avere a disposizione una macchina fotografica (o uno *smartphone* con fotocamera di alto livello) per procedere con la documentazione dell'oggetto.
In seguito le immagini vengono caricate in un *software* che procede alla creazione della nuvola di punti con la tecnica *Structure From Motion (SfM)*[4], per cui le diverse immagini, inquadrando punti dell'oggetto che si ripetono da una all'altra, forniscono dati al *software* relativi alla loro posizione nello spazio, così che possano essere collocate nella posizione esatta (*Figura 1.1*).
A questo punto le immagini, presentando sovrapposizioni, vengono elaborate per generare il modello 3D e la *texture*, derivante direttamente dagli scatti effettuati (la qualità dell'illuminazione è strettamente connessa al risultato finale).
Il limite di questo metodo è però la natura del materiale di cui è costituito il manufatto da acquisire.
Le superfici lucide e metalliche o trasparenti infatti presentano riflessioni che alterano la luce che arriva alla macchina fotografica. Pertanto lo stesso punto preso da due angolazioni differenti appare diverso in base all'illuminazione a cui e sottoposto, andando a rendere difficoltosa per il *software* la fase di allineamento delle immagini.

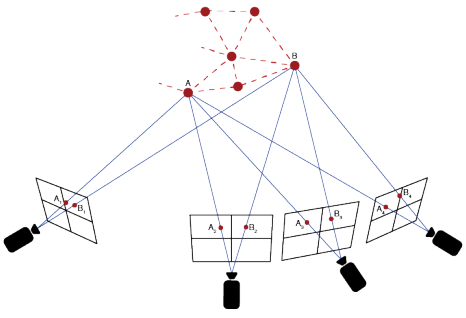


Figura 1.1 Schema di funzionamento del *Structure from Motion.* La posizione dei punti A e B all'interno delle immagini è diversa a seconda del punto di presa della fotocamera.

A disposizione esistono molti *software* utilizzabili per effettuare la fotogrammetria: Agisoft® Metashape, Autodesk® ReCap Photo, Meshroom®, 3DF Zephyr®, Pix4D-Mapper®.
Il principio di funzionamento è sempre lo stesso: l'elaborazione delle immagini genera una prima ricostruzione dell'oggetto costituita di punti singoli, che viene poi successivamente trasformata nella geometria vera e propria.
Tra questi è stato scelto di utilizzare Agisoft® Metashape, che ha un abbondante utilizzo in campo archeologico.

4. Horňák M., Novakovič P., Zachar J., *3D Digital recording of architectural and artistic heritage,* CONPRA Series, I, 2017.

1.2.2 *Modellazione 3D*

L'operazione di acquisizione dà origine a un file contenente la geometria dell'oggetto, che però spesso non è definitiva: è possibile che siano necessarie operazioni di correzione e di ottimizzazione per poterlo utilizzare in altri progetti. Inoltre può essere necessario ricostruire parti di sculture, edifici o altri elementi che si aggiungeranno all'esperienza finale.
Per questo motivo è necessario attraversare una seconda fase riguardante la modellazione tridimensionale.
Per le forme organiche di cui ci si occupa in questo ambito, la tipologia più adatta tra i vari metodi di modellazione è sicuramente la modellazione poligonale.

Mediante i *software* appositi è possibile creare e modificare la *mesh*, ovvero l'oggetto tridimensionale composto da vertici, lati e facce che ne definiscono la geometria.
L'aspetto della *mesh* può essere piuttosto diverso, si può utilizzare uno stile *low poly*, che utilizza poche facce per tenere basso il calcolo richiesto dal PC e originando oggetti più stilizzati, oppure aumentare il numero di vertici e rendere la *mesh* più realistica (*high poly*), a discapito della maggiore potenza richiesta (*Figura 1.2*).
Le modifiche alla *mesh* è possibile effettuarle anche attraverso operazioni di *sculpting*, simulando ciò che nel mondo reale è il processo di scultura, effettuando incisioni, aggiunta di materiale, smussamenti, con lo scopo di ottenere una forma organica.

Nonostante gran parte dell'industria dell'intrattenimento sia ancora legata ad alcuni software potentissimi, ma incentrati soprattutto nella modellazione o nello *sculpting*, come ad esempio Maya® e 3dsMax®, prodotti da Autodesk®, o Zbrush®, colosso per la scultura digitale, rilevante è la posizione all'interno di questo panorama di Blender®, uno strumento *open source* in grado di svolgere ad alto livello tutti gli step nel campo della computer grafica, a cui sono facilmente aggiungibili *plugin* creati dalla *community* per risolvere specifici compiti.

Figura 1.2 La stessa *mesh* modellata in uno stile *low poly* a sinistra, può crescere di risoluzione e divenire più realistica, con un approccio *high poly.*

Oltre alla già citata modellazione 3D e scultura è quindi utile per operazioni di animazione, di *rendering*, *texturing*, effetti speciali e montaggio video, tutto eseguito in un unico *software*, motivo per cui è stato selezionato per lo sviluppo dei progetti.

1.2.3 *Realtà aumentata e realtà virtuale*

Realtà aumentata e realtà virtuale (abbreviate AR e VR dall'inglese) sono tecnologie già presenti da svariato tempo, però solo negli ultimi anni si stanno avvicinando ad un pubblico più ampio, grazie ad alcune alternative che permettono di essere sperimentate anche su dispositivi economici, alla portata di tutti.

La realtà aumentata in particolare prevede la sovrapposizione di due mondi, quello reale, inquadrato dalla fotocamera del dispositivo, e quello digitale, generato dall'applicazione in esecuzione. Il principio su cui si basa la sovrapposizione dei due contenuti è il riconoscimento dello spazio situato attorno all'utente. Il dispositivo è in grado di capire i movimenti e la posizione degli oggetti semplicemente inquadrandoli, tracciando così l'ambiente circostante. Parallelamente riesce quindi a disporre i contenuti digitali e gestire i movimenti della fotocamera camera virtuale copiando gli spostamenti di quella fisica, rendendo la scena un ambiente unico.
Il protagonista del riconoscimento viene detto *marker*: si tratta di un oggetto specifico in relazione alla cui posizione si costruisce l'ambiente digitale. Può essere un'immagine o un oggetto tridimensionale oppure non esserci proprio (in quest'ultimo caso il riconoscimento avviene su tutte le superfici ed è l'utente a posizionare l'oggetto digitale sopra di esse).
Il riconoscimento del *marker,* indipendentemente dal *software* utilizzato per sviluppare l'esperienza, avviene grazie a un *Software Development Kit* (SDK). Uno dei più frequentemente utilizzati è Vuforia™, uno strumento gratuito che permette un buon riconoscimento in tempo reale di varie tipologie di *marker* su dispositivi mobili e garantisce una certa stabilità del tracciamento lungo tutta la durata dell'esperienza.

Tipicamente le applicazioni di realtà aumentata possono essere sperimentate su due tipologie di dispositivi.
La più economica riguarda l'utilizzo di *smartphone* e *tablet* sul quale viene direttamente installata l'app contenente l'esperienza.
I dispositivi devono soddisfare alcuni requisiti sia di *hardware* che di *software*. Prima di tutto il sistema operativo, sia esso Android o iOs, deve essere tra i più recenti disponibili per poter essere compatibile

con l'utilizzo di Vuforia™.
Inoltre è fondamentale, per la buona riuscita dell'esperienza, che vi sia installato un ulteriore SDK[5]: ARCore per quanto riguarda Android, ARKit per gli *smartphone* della Apple. Questi collaborano con Vuforia™ per il tracciamento stabile e memorizzano l'ambiente già mappato anche quando questo si trova fuori dal campo visivo della fotocamera. Pertanto per soddisfare questi requisiti è necessario che il modello di *smartphone* (e *tablet*) sia comunque di fascia medio/alta, dovendo in aggiunta essere dotato del sensore di movimento.

Una seconda possibilità è data invece da alcuni apparecchi chiamati *smartglasses*, tra cui figurano i Google Glasses e gli Hololens di Microsoft (*Figura 1.3*). Si presentano come un paio di occhiali da indossare in testa. Le due lenti sono costituite da un *display* ottico collegato con un processore incorporato nel dispositivo.
Le immagini che compaiono sulle lenti, che sono comunque trasparenti permettendo di vedere oltre, si sovrappongono quindi al mondo reale allo stesso modo di come lo fanno su uno *smartphone*, con la differenza che l'approccio in questo caso è *hand free*: gli *smartglasses* sono infatti dispositivi indipendenti e rispondono a comandi *touch* posti nulle bacchette laterali o a comandi vocali.

La realtà virtuale invece agisce in modo differente.
La differenza principale rispetto alla realtà aumentata è che in questo caso l'utente entra in una dimensione completamente fittizia, isolandosi dal mondo reale. All'interno di questa dimensione è possibile interagire con l'ambiente, spostarsi e compiere determinate azioni, a seconda del tipo di esperienza e su quale dispositivo viene avviata.
Lo strumento necessario per entrare nella realtà virtuale è il visore. Anche in questo caso si tratta di un apparecchio da montare sulla testa simile ad un paio di occhiali.
A differenza degli *smartglasses* però le lenti non proiettano immagini, ma servono per deformare e ingrandire un *display* posto dietro di esse:in questo modo agli occhi dell'utente il contenuto in riproduzione sul *display* si sostituisce perfettamente alla realtà.
Inoltre, per un migliore isolamen-

Figura 1.3 Un paio di *smartglasses* prodotto da Microsoft, gli Hololens

5. Developers Google Ar, *https://developers.google.com/ar/develop,* ultima consultazione 25 agosto 2021.

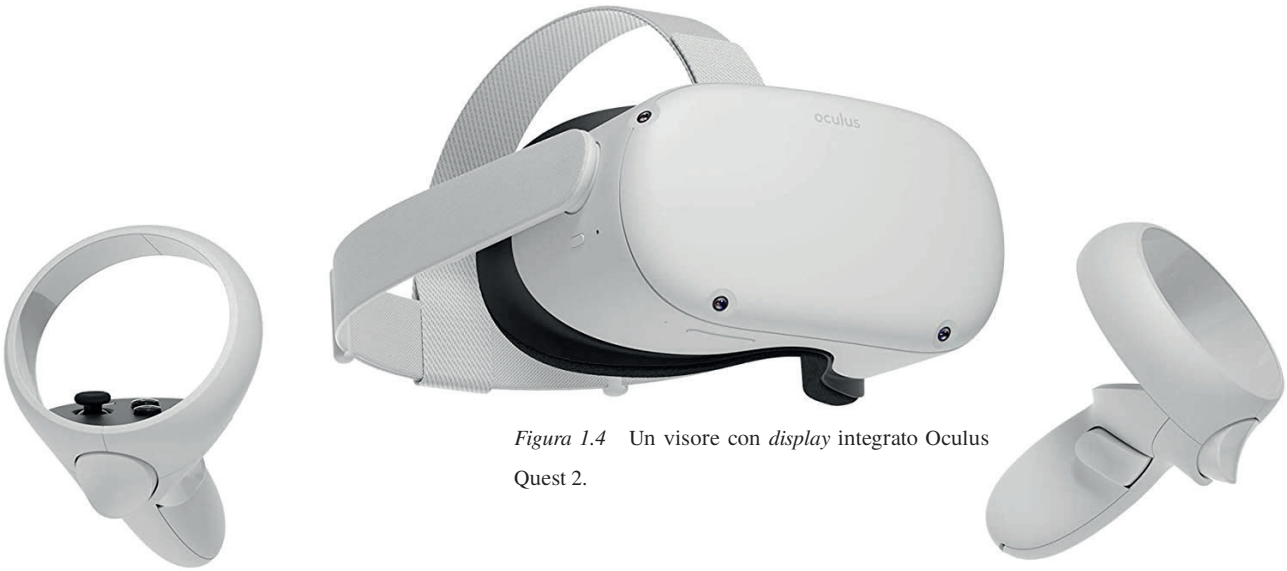


Figura 1.4 Un visore con *display* integrato Oculus Quest 2.

to, il visore presenta una forma tale che gli occhi non possano vedere altro al di fuori del visore stesso.

In commercio esistono due principali tipologie di dispositivi adatti a supportare esperienze di realtà virtuale, che si distinguono per le caratteristiche del *display*.
I più costosi e di alto livello sono dotati di display integrato, come ad esempio il celebre Oculus Quest (*Figura 1.4*), oltre a presentare anche due *controller* per avere accesso al movimento delle braccia nello spazio virtuale.
Anche se attualmente le aziende si stanno muovendo in direzione dei visori *All-in-One*, la distinzione principale all'interno della categoria è da fare tra quelli *stand alone*, che non necessitano di nessun altro dispositivo per funzionare, e i visori che si appoggiano a un PC o una *console*, connettendosi ad essi, riproducendo quindi videogiochi e contenuti installati esternamente.

Un gruppo a parte invece è costituito dai visore con *display* esterno (*Figura 1.5*).
In questo caso non si tratta di un vero e proprio dispositivo ma solamente di un accessorio dentro il quale si inserisce uno *smartphone*, il cui schermo diventa il *display* utilizzato per l'esperienza riproducendo contenuti digitali installati sotto forma di app.
Anche in questo caso ci sono ovviamente le due lenti prima citate, che ingrandiscono lo schermo dello *smartphone*, diviso in due sotto-*display*, ragion per cui, per godere di una qualità dell'esperienza sufficiente, la risoluzione e la dimensione dello schermo deve essere in partenza già piuttosto alta.
Ovviamente trattandosi di un accessorio senza sistema operativo e hardware per farlo funzionare il risparmio economico è evidente, a discapito però della qualità dell'immagine finale.

L'ultimo attore che prende parte alla realizzazione dell'esperienza è il *game engine*, la piattaforma al cui interno avviene la programmazione, disponendo i contenuti digitali in una scena tridimensionale e impostando le interazioni che l'utente effettuare con gli elementi a sua disposizione.

Tra i *game engine* più diffusi com-

Figura 1.5 Un visore con *display* esterno

paiono Unity® e Unreal Engine®. Entrambi sono dei motori grafici in tempo reale e permettono la realizzazione prodotti diversi, dal videogioco 3D fotorealistico al 2D, comprendendo ovviamente la produzione di contenuti AR e VR.
Ciò che accomuna i due *software* è la possibilità di poterli utilizzare gratuitamente registrandosi ai loro servizi online, avendo a disposizione comunque strumenti professionali in grado di restituire *release* per più piattaforme finali.

Sono disponibili inoltre molte preimpostazioni selezionabili in base al tipo di progetto che si vuole creare, così da poter iniziare da una base di partenza con alcune semplificazioni.

1.3 Casi studio

L'utilizzo delle tecnologie presentate in precedenza è già piuttosto diffuso nel panorama culturale come elemento di valorizzazione e attrazione.

1.3.1 *Sette musei di Varese*

A Varese sette musei sono stati oggetto di innovazione da parte di un bando della Regione Lombardia. Il Museo Civico d'Arte Moderna e Contemporanea del Castello di Masnago, i Musei Civici di Villa Mirabello, il Museo Baroffio e del Santuario del Sacro Monte sopra Varese, la Casa Museo Lodovico Pogliaghi, il Museo Castiglioni e il Centro Espositivo Monsignor Macchi[6] sono ora connesse grazie allo sviluppo di un'appposita applicazione per *smartphone*.
All'interno delle strutture è possibile fermarsi davanti a diversi manufatti e sperimentare la realtà aumentata, che in questo caso permette di ottenere informazioni dettagliate riguardo alla storia dell'opera specifica.
Lo schermo del dispositivo offre alcuni pulsanti con cui si può interagire con gli elementi digitali e scoprire lo stato originario di alcuni oggetti.

1.3.2 *Bone Hall - Smithsonian National Museum of Natural History*

A Washington, all'interno del National Museum, è collocata la Bone Hall, la "sala delle ossa", in cui dal 1881 è esposta una vasta collezione di scheletri, raggruppati per rappresentare ogni gruppo principale dei vertebrati (*Figura 1.7*).
Solitamente in questa tipologia di museo, a fianco di ogni scheletro

Figura 1.6 Utilizzo dell'applicazione di AR all'interno delle sale del Museo Castiglioni.

Figura 1.7 L'esposizione degli scheletri all'interno della Bone Hall.

6. In Italia Magazine, *Entrare nella storia con la realtà aumentata. Il progetto pilota dei musei di Varese, https://initalia.virgilio.it/,* ultima consultazione 29 agosto 2021.

7. Smithsonian National Museum of Natural History, Bone Hall, *https://naturalhistory.si.edu/exhibits/bone-hall,* ultima consultazione 25 agosto 2021.

Figura 1.8
L'applicazione Skin and Bone in funzione mentre riporta in vita lo scheletro di un pescespada.

sono presenti illustrazioni e fotografie per fare capire al visitatore come appare l'animale in vita.
Lo Smithsonian ha scelto di comunicare questo tipo di informazioni con una loro app di realtà aumentata, chiamata "Skin and Bones"[7], disponibile al momento solo nello *store* di iOs (*Figura 1.8*).
Il visitatore può così inquadrare uno dei tanti scheletri presenti nelle teche della sala e questi vengono "resuscitati" sullo schermo dello *smartphone* e grazie la tracciamento dell'ambiente circostante l'animale si sovrappone perfettamente alle ossa, in modo che si possa vedere in contemporanea la struttura ossea e i tessuti che la ricoprono.
Per alcuni degli animali inoltre è possibile anche avviare animazioni durante le quali gli scheletri prendono vita, permettendo di comprendere i movimente delle articolazioni delle varie specie presenti.

Wessex Archaeology 1.3.3

Il team del Wessex Archaeology[8] sta sviluppando esperienze di realtà virtuale, consentendo al pubblico di prendere parte a una visita immersiva negli scavi sotto l'abbazia di Bath.
Durante la fasi degli scavi nel sottosuolo sono state effettuate operazioni di scansione 3D attraverso la fotogrammetria con la tecnica già vista in precedenza del *Structure from Motion*, il che ha permesso di ricostruire digitalmente tutto l'am-

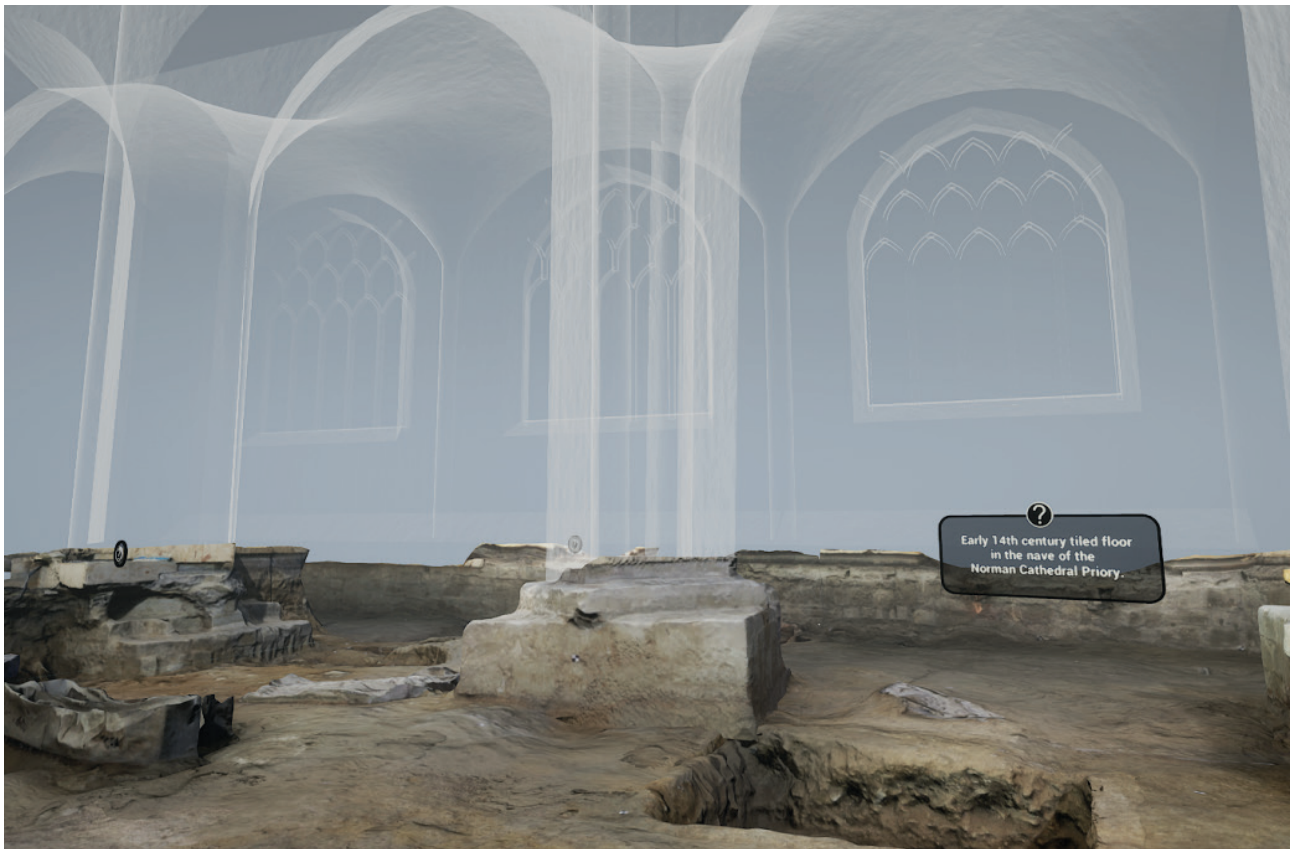


Figura 1.9
Una scena della visita virtuale negli scavi dell'abbazzia di Bath.

biente con precisione.
All'utente è quindi permesso di indossare il visore e iniziare il tour, servendosi di un navigatore e di una torcia virtuale per esplorare gli scavi (*Figura 1.9*), che in alcuni casi possono essere difficilmente accessibili nella realtà.
Oltre all'ambiente derivato dall'acquisizione, sono presenti ricostruzioni ipotetiche del vasto patrimonio culturale conservato nel tempo, tra cui resti romani e sassoni che restituiscono l'idea delle costruzioni andate distrutte.

La stessa azienda propone altre esperienze di questo tipo, offrendo la possibilità di viaggiare nel tempo alla riscoperta dello stile di vita di antiche popolazioni del Wessex.
Gli ambienti in questo caso non provengono da un'operazione di fotogrammetria, ma attraverso modellazione 3D e documentazione storica sono state ricostruite abitazioni e interni tipici dei Romano-Britanni e dei Sassoni (*Figure 1.10 e 1.11*).
Il visitatore entra nei panni di un cittadino dell'epoca, può interagire con alcuni oggetti afferrandoli e spostandoli, sperimentando in prima persona come poteva essere la vita all'interno di quelle società.

Figura 1.11 L'interno di un'abitazione nella società dei Sassoni.

Zoan Oy - National Museum of Finland 1.3.4

Nel più importante museo di Helsinki è conservato un quadro altamente rappresentativo del passato

Figura 1.10 L'interno di un'abitazione nella società dei Romano-Britanni ricostruita dal team di Wessex Archeology.

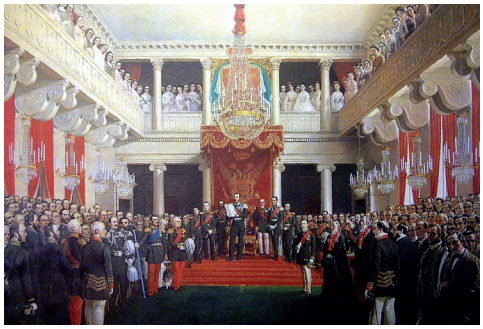


Figura 1.12 Il quadro "*L'imperatore Alessandro II dichiara l'apertura della Dieta del 1863*", realizzato da Robert Wilhelm Ekman.

8. Wessex Archaeology, Bath Abbey, *https://www.wessexarch.co.uk/our-work/virtual-reality-experience-excavation-below-bath-abbey,* ultima consultazione 23 agosto 2021.

9. VrFocus, *National Museum Of Finland Offers Virtual Time Travel, https://www.vrfocus.com/2018/02,* ultima consultazione 22 agosto 2021.

Figura 1.13 Un *frame* estratto dall'esperienza di realtà virtuale, in mezzo alla folla durante l'apertura della Dieta.

politico finlandese, rappresentante l'apertura della Dieta di Finlandia per opera di Alessandro II, l'organo legislativo durato per tutto il XIX secolo (*Figura 1.12*).
Lo studio Zoan Oy[9] ha trasformato un quadro tradizionale in una visita al suo interno.
Dopo aver osservato l'opera infatti inizia una sessione di realtà virtuale.
L'utente si trova inserito nella sala del Palazzo Imperiale durante la riunione del Governo. Similmente a quanto accade in un videogioco si ha la possibilità di interagire con i personaggi storici raffigurati nel dipinto e di spostarsi tra la folla, visitare la Sala degli Specchi e persino avere un dialogo con l'imperatore (*Figura 1.13*).
La realizzazione di questo tipo di esperienza richiede la collaborazione di esperti storici per i contenuti dei dialoghi e di *3d artist* che si occupino di donare movimenti realistici a tutti i modelli dei personaggi presenti, collaborando anche con attori in carne ed ossa che recitano in *motion capture*.

Capitolo 2

Elaborazione dei contenuti

2.1 La selezione delle opere

2.1.1 *Le statue dell'Asia Meridionale*

Per realizzare le esperienze di AR/VR sono state selezionate inizialmente quattro opere, tutte appartenenti al MAO, in particolare alla sezione dell'Asia Meridionale, curata dalla dottoressa Claudia Ramasso.
Si tratta di uno *Yaksha* (*Figura 2.1*), della testa di un *Buddha Kapardin* (*Figura 2.2*) e di due figure del *Buddha* stante (*Figure 2.3 e 2.4*).

Le statue sono state scelte sia per una questione di rilevanza culturale e artistica, sia perché rispondono perfettamente alle caratteristiche richieste dalle tecniche di acquisizione tridimensionale.

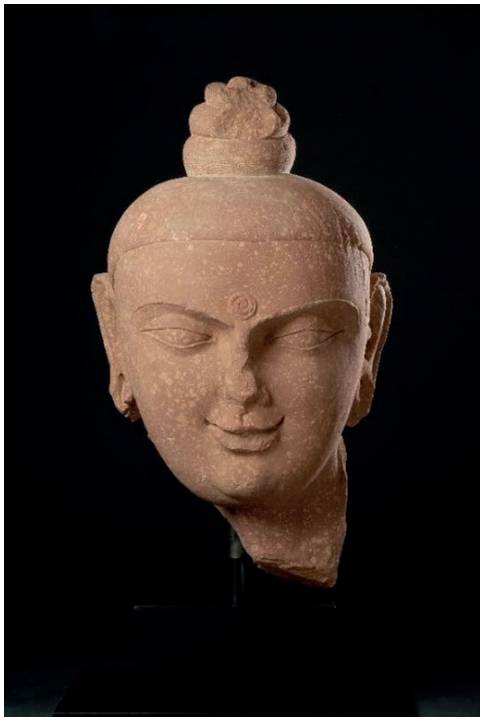


Figura 2.2 La testa del Buddha con la tipica acconciatura a *Kaparda*.

Figura 2.1 La rappresentazione dello Yaksha conservato al MAO, privo di braccia e testa.

Per procedere al rilievo fotogrammetrico attraverso la tecnica SfM (*Structure from Motion*), è necessario che la superficie dell'oggetto da rilevare presenti alcuni accorgimenti.
La scultura deve essere piuttosto ruvida, non metallica e opaca, in modo tale che le foto scattate all'opera non presentino riflessi e specchiature che possono essere fuorvianti per il *software* e andrebbero ad interferire con la costruzione del modello 3D.
Le opere prese in esame sono realizzate in arenaria rossa maculata e in scisto grigio, materiale che le rende adatte per la fotogrammetria.

Delle quattro statue ne sono state selezionate due, le due rappresentazioni del Buddha stante, che presentano caratteristiche di conservazione simili (ad entrambe mancano arti superiori e inferiori e parte dell'aureola) e appartengono ai due centri di produzione artistica principali del periodo: regno del *Gandhara* e scuola di *Mathura*.

Data la possibilità di reperire ulteriore materiale archeologico per ricostruire l'ambientazione storica grazie alla collaborazione del professore Luca Maria Olivieri, la statua del Buddha del Gandhara è la protagonista dell'esperienza di realtà virtuale, mentre sul Buddha del periodo Gupta viene realizzata un'esperienza di realtà aumentata.

2.1.2 *Il Buddha del periodo Gupta*

La prima opera selezionata è il Buddha stante del periodo *Gupta*.
Gupta[1] fu uno dei maggiori imperi dell'antica India, governato dalla dinastia omonima tra il IV e il VI secolo d.C.; il nucleo centrale dell'impero fu il regno del Magadha, con capitale Pataliputra.
Nella fase di massima espansione occupò la maggior parte dell'India settentrionale e degli attuali stati del Pakistan e del Bangladesh.

La scultura, nella tipica arenaria rossa maculata, è comparabile ai migliori esempi della scuola *Gupta* di *Mathurā*. La figura del Buddha stante, con le mani e gli arti inferiori spezzati, si impone per la compostezza della raffigurazione: le misurate proporzioni del corpo elegante sono esaltate dall'ampio nimbo rotondo che incornicia la testa.
L'immagine idealizzata del Buddha, vigorosa e aggraziata al contempo, si esalta nella forma sensuale del corpo, fasciato dalla tunica.
La veste presenta il sapiente panneggio della scuola di *Mathurā*, sensibile all'esperienza del *Gandhāra*, tuttavia innovatrice nella misurata distribuzione delle pieghe, dal tratto leggero ma finemente reso nel movimento dell'abito. Il cordone in vita segna il bordo dell'*antarīya* indossato sotto la tunica che, aderendo al corpo come un velo sottile, fa trasparire le forme.
Seppure coperti dell'abito si delineano così pienamente le gambe robuste e armoniose, il pube asessuato, i fianchi stretti, i muscoli pettorali torniti, le braccia proporzionate.
Il Buddha probabilmente mostrava il gesto dell'*abhayamudrā* con la mano destra e teneva appena sollevato un lembo della veste con la mano sinistra, secondo l'iconografia canonica del periodo.
La testa, dolcemente sporta in avanti in un gesto di incoraggiamento, si appoggia sul lungo collo segnato da tre pieghe di bellezza. I tratti del volto, segnato da un lieve sorriso, mostrano la sapienza artistica di questa scuola: gli occhi

1. I testi e le informazioni riguardanti le due statue sono concessi direttamente dalla dottoressa Claudia Ramasso.

Figura 2.3 Il Buddha stante del periodo Gupta in arenaria rossa.

socchiusi con lo sguardo rivolto verso il basso, le labbra carnose e il naso sottile riempiono delicatamente l'ovale del viso, incorniciato ai lati dalle orecchie dai lunghi lobi. L'alta *ushnīsha* svetta sul capo del Buddha, ricoperto da file regolari di piccoli riccioli a spirale.
Il viso ovale e pieno si inserisce al centro della grande aureola - di cui si conserva solo la porzione destra - decorata con motivi vegetali disposti secondo cornici concentriche che seguono la corolla di fiori di loto centrale. Il primo giro presenta un motivo di piccole foglie palmate affiancate; il secondo mostra un virgulto continuo dall'andamento sinuoso - in realtà composto da steli di fiori sbocciati susseguenti - entro le cui anse si inserisce una ricca decorazione vegetale; il terzo è occupato da una ghirlanda intrecciata e fermata da fascette decorate con un fiorellino.
L'ultima cornice, separata dalla precedente da una fila di piccoli grani, si distingue per la decorazione a brevi emicicli incisi lungo il bordo, che rendono l'idea dei raggi solari, segno del fulgore emanato dalla figura del Buddha.

2.1.3 *Il Buddha del Gandhara*

La seconda opera protagonista del progetto è sempre una rappresentazione stante del Buddha, originario però del *Gandhara*, un'antica regione situata fra il Pakistan settentrionale e l'Afghanistan orientale. Nei primi secoli della nostra era il vasto impero creato dalla dinastia centro-asiatica *Kushāna*, che si estendeva fino alla porzione occidentale della pianura gangetica, favorì un'intensa produzione artistica, incentrata particolarmente nella regione di *Mathurā* - dove è netta la componente autoctona indiana - e nell'area del *Gandhara*. È a partire da questo periodo che il *Gandhara* raggiunse il suo massimo splendore artistico, quale luogo d'incontro tra i contenuti dell'arte sacra buddhista e un linguaggio cosmopolita caratterizzato dalla compresenza di influssi classici (ellenistico-romani), indiani, iranici e centro-asiatici.

La figura si presenta con la gamba sinistra lievemente piegata nell'atto di avanzare un passo.
La statua manca dei piedi e delle mani e ha l'aureola fortemente compromessa; nondimeno, malgrado qualche scheggiatura, si conserva la buona fattura dell'opera.
Il Buddha indossa un *samghāti* appoggiato su entrambe le spalle, reso con un panneggio sapiente non privo di una certa gravità, che lascia tuttavia trasparire la fisionomia del corpo, alleggerendo nel complesso la raffigurazione.
Il volto, in ottimo stato di conservazione, ha tratti lievi: il mento affusolato compensa la fronte alta e la testa leggermente squadrata, con una voluminosa *ushnīsha* di for-

Figura 2.4 Il Buddha stante del Gandahara in scisto grigio.

ma globulare.
Poco sopra il centro delle sopracciglia si trova l'ūrnā a rilievo, particolarmente piatta e piccola, che quasi si perde sulla fronte spaziosa. Gli occhi sono allungati, con le palpebre carnose semichiuse sulle iridi incise: sebbene il Buddha sia raffigurato nell'accenno di un movimento, l'espressione dello sguardo appare rivolto verso il basso e assorto nell'atto della meditazione. Il naso aquilino e le labbra piene, ma non carnose, ingentiliscono il viso, incorniciato dai lobi allungati, ma in modo contenuto, delle orecchie, e dall'acconciatura dei capelli mossi, resa con onde a rilievo che partono dal centro della fronte e seguono il contorno della testa.
Il trattamento dell'*ushnīsha* risulta ambiguo, poiché le ciocche di capelli sembrano quasi coprire una reale protuberanza cranica, seguendone il profilo: il ciuffo in effetti non presenta elementi di legatura alla base. Le braccia spezzate non permettono di stabilire con precisione l'atteggiamento delle mani; se ricorriamo tuttavia alla comparazione con altre raffigurazioni simili si può supporre che il braccio sinistro trattenesse un lembo della veste e quello destro fosse in *abhayamudrā*.

2.2 Acquisizione delle opere

Per lo sviluppo delle applicazioni VR/AR è stato necessario prima di tutto avere dei modelli digitali delle statue.
Il processo è avvenuto in due fasi principali: una sessione all'interno del MAO coordinata dalla dottoressa Claudia Ramasso e dall'architetto Francesca Ronco in cui ci si è occupati della scatto delle fotografie necessarie; la ricostruzione dei modelli 3D attraverso il *software* Agisoft® Metashape.

2.2.1 *La fotogrammetria*

L'allestimento all'interno del museo di uno studio fotografico ha previsto l'impiego di apparecchiatura professionale.
Per lo scatto delle fotografie è stata utilizzata l'attrezzatura fornita dal ModLab Arch del Politecnico di Torino, ovvero una professionale Canon reflex EOS 6D installata su un treppiede e dotata di un telecomando per il controllo a distanza, per mantenere lo scatto il più fermo possibile.

Per l'illuminazione della scena sono stati installati quattro cavalletti, su due dei quali sono stati montati due ombrelli argentati per indirizzare la riflessione della luce, mentre gli altri due hanno retto due *soft-box*, delle "scatole" di tessuto che diffondono omogeneamente la luce emanata dalla lampadina al loro interno.
L'obiettivo del set di luci è quello di garantire un'illuminazione dell'opera il più omogenea possibile, cercando evitare zone d'ombra e di luce troppo contrastate.
Lungo tutta la sequenza di fotografie la luce deve essere la stessa per garantire una migliore ricostruzione del modello, pertanto, una volta posizionati, i punti di illuminazione sono stati mantenuti fissi.

Figura 2.5 L'allestimento del set fotografico all'interno delle sale del MAO.

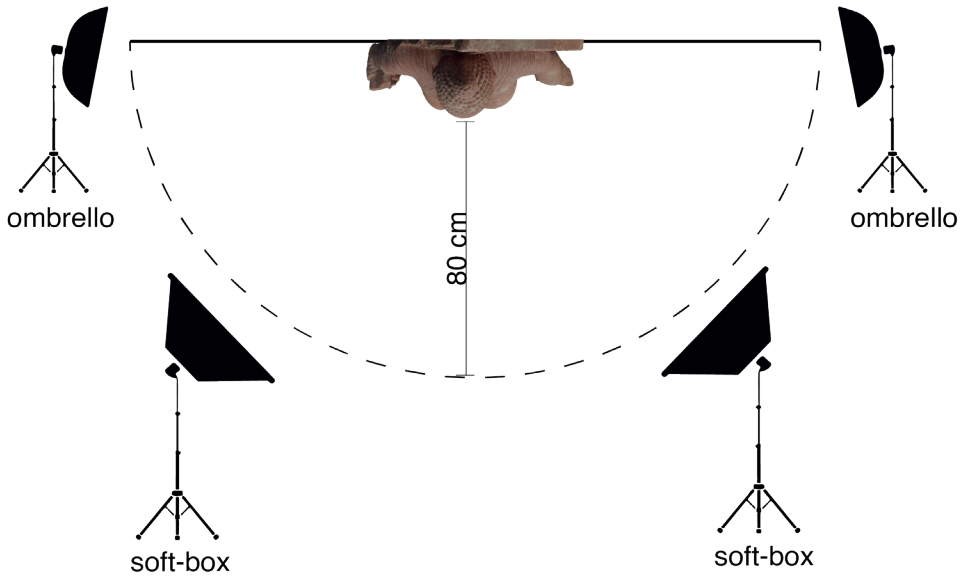


Figura 2.6 Schema della fotogrammetria eseguita sul Buddha Gupta.

Un altro aspetto fondamentale riguarda le impostazioni della camera (diaframma, ISO, tempi di esposizione), che devono essere mantenute anch'esse uguali dall'inizio alla fine della sessione.
Tendenzialmente il processo non presenta troppe difficoltà, se non alcune riguardanti la posizione delle statue: vista la conformazione e il posizionamento delle opere all'interno del museo, si è dovuto eseguire il lavoro di rilievo senza possibilità di spostare i Buddha in luoghi più comodi.

Perciò il rilievo del Buddha *Gupta*, che si presenta addossato a una parete, è avvenuto scattando fotografie in un campo visivo di 180° (*Figura 2.6*) e si sono riscontrate alcune problematiche per le zone laterali attaccate al muro. Nell'elenco finale ovviamente non sono presenti immagini relative al retro della statua, ma per gli obiettivi del progetto AR non sono necessarie.

Il Buddha del *Gandhara* invece si trova al centro della sala ed è quindi possibile osservarlo a 360° (*Figura 2.7*). Le difficoltà insorte sono da attribuire al piedistallo di 96 cm sul quale è posizionato, che hanno reso problematico lo scatto delle foto per la parte superiore della testa.
Si è ovviato servendosi di un tavolo e di una scala per innalzare ulteriormente il cavalletto.

I punti luce nel caso della statua *Gupta* sono stati posizionati con le due soft-box frontalmente a dare la maggior quantità di illuminazione e i due ombrelli sui lati, per eliminare ulteriori ombre createsi.
Il Buddha del *Gandhara* ha reso possibile l'illuminazione anche sul retro della statua, perciò le due soft-box sono state messe per dare

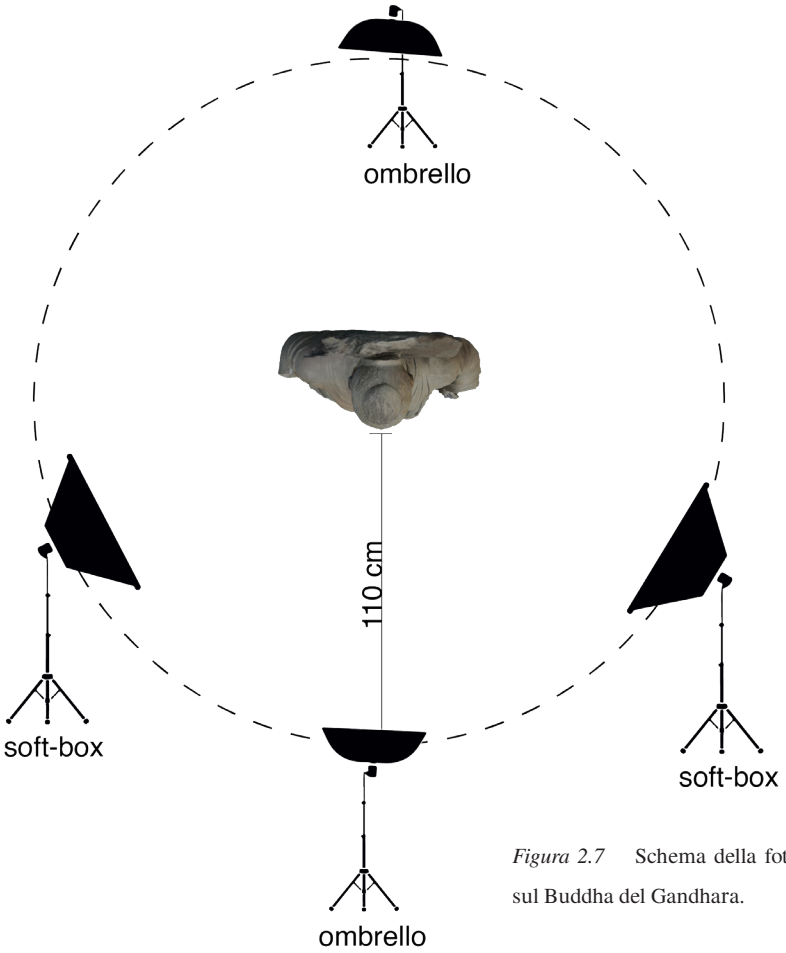


Figura 2.7 Schema della fotogrammetria eseguita sul Buddha del Gandhara.

luce di tre quarti sia da destra, sia da sinistra per cercare di essere più omogenei possibile, un ombrello frontale frontale per spalmare maggiormente le ombre e uno solamente sul retro, per un minimo di luce: essendo questa porzione dell'opera non rilevante quanto il fronte, anche se le foto non sono chiare (e quindi la conseguente ricostruzione del modello di qualità inferiore) si può sorvolare.

Affinché Agisoft® Metashape funzioni correttamente e restituisca un modello digitale fedele, le immagini da fornire devono presentare delle zone di sovrapposizione: in tal modo infatti il *software* per ogni immagine riesce a risalire facilmente alla posizione originale della camera, in relazione alle altre foto e ai punti di sovrapposizione.

Pertanto si è scelto di procedere con gli scatti inquadrando l'opera attraverso "fasce" (*Figura 2.8*).
Partendo dal basso, si fotografa la statua in posizione ravvicinata per tutte le angolazioni possibili (180° o 360°). Finita la prima sequenza, si alza il cavalletto della macchina fotografica salendo di qualche centimetro, avendo cura che la fascia successiva e quella precedente inquadrino una porzione della statua che presenti punti omologhi, così che ci siano le sovrapposizioni (*Figura 2.9*).

Al termine dell'operazione la statua del Buddha del *Gandhara* ha previsto lo scatto di 118 fotografie (quantità piuttosto alta, da attribuire alla possibilità di inquadrarlo a 360° e alla cattiva illuminazione di alcune zone del panneggio inferiore, che

Figura 2.8 Altezza da terra della fotocamera ad ogni cambio di "fascia".

hanno richiesto una luce diversa e uno zoom molto ravvicinato), mentre per il Buddha *Gupta* ne sono state scattate 96.
Un numero così alto di immagini per entrambe le statue è dovuto ai circa 20 scatti per fascia, con lo scopo di ottenere dei modelli digitali estremamenti precisi, sia in termini di *mesh*, sia per quanto riguarda la *texture*.

Figura 2.9 Esempio di due immagini appartenenti a fasce contigue.

Elaborazione su Agisoft® Metashape 2.2.2

Le due sequenze di immagini vengo dunque caricate all'interno del *software* Metashape.
Il *software* prevede un preciso *workflow* per la ricostruzione digitale di un modello.
Nel primo step si esegue l'upload delle foto scattate in formato *.jpeg*. Metashape in automatico provvede ad analizzare le immagini attraverso il *Structure from Motion* e, cercando i punti di sovrapposizione tra una foto e l'altra, risale alla posizione della fotocamera per ogni scatto e la posiziona all'interno dell'ambiente di lavoro tridimensionale (*Figura 2.10*).

Durante questa fase si può impostare un livello di precisione del tracciamento più o meno alto e, a seconda della scelta effettuata, alcune foto possono essere scartate

Figura 2.10 Fase di allineamento delle immagini ad opera di Metashape.

in automatico perchè difficili da posizionare. In questo caso si è scelto di prediligere una precisione massima, con l'aumento del tempo necessario per l'allineamento.

Il secondo passaggio porta alla formazione di una prima versione della cosiddetta nuvola di punti, chiamata *tie points*, da cui si inizia a vedere un'abbozzata ricostruzione della statua (*Figura 2.11*).

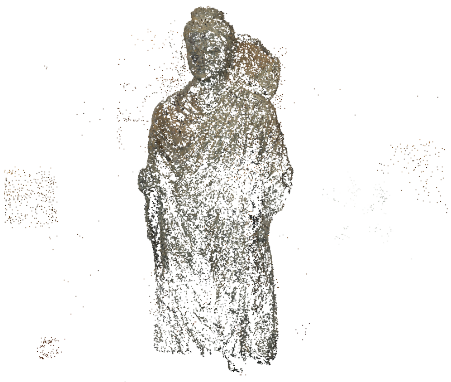


Figura 2.11 Creazione dei *tie points* durante il secondo step.

Con lo step successivo si crea la *dense cloud*, ovvero una nuvola di punti molto più densa e da cui successivamente verrà elaborata la geometria finale (*Figura 2.12*).
È dunque necessario fare attenzione alle impostazioni relative alla qualità del calcolo previsto per la dense cloud, in quanto con *Ultra High Quality* impostato (l'opzione presa nel nostro caso) si avrà di conseguenza un numero di punti all'interno della nuvola maggiore.

Figura 2.12 Costruzione della *dense cloud* nella terza fase.

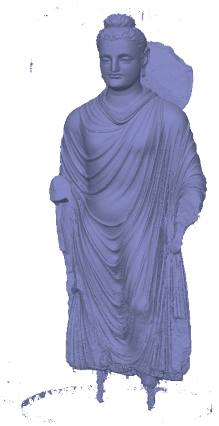


Figura 2.13 La *mesh* costruita nella quarta fase.

Figura 2.14 La scultura completata , con la *texture* applicata nell'ultimo passaggio

In seguito partendo dalla *dense cloud* si continua con la costruzione della geometria: il *software* procede a collegare tra loro i *points* generando vertici, lati e facce necessari per ottenere la *mesh* (*Figura 2.13*). Anche in questo caso alcune impostazioni possono variare il livello di precisione della *mesh*, l'opzione selezionata è comunque sempre in direzione di un modello 3D molto dettagliato.

Infine l'ultimo passaggio prevede la realizzazione della *texture* sotto forma di immagine *.jpeg*.
A partire dalle immagini allineate, Metashape è in grado di proiettarle sulla *mesh* appena costruita e di creare un'immagine unica per tutto l'oggetto scansionato, senza bisogno di preparare una mappatura UV manualmente (*Figura 2.14*).

Da Metashape il modello può essere esportato in formato *.obj*, un tipo di file facilmente apribile dalla maggior parte dei *software* che si occupano di 3D.

Operazioni di ottimizzazione 2.2.3

Pur esportando da Metashape in ottima qualità, le due statue 3D presentano alcuni piccoli problemi da sistemare prima di poterle utilizzare all'interno delle app.

Innanzitutto, quando sono state scattate le fotografie, inevitabilmente è stato inquadrato anche parte dell'ambiente circostante del museo.
Queste porzioni di spazio sono state anche processate su Metashape: ne risulta che nella *mesh* finale sono presenti parti di parete, tracce di altre opere e del podio espositivo. Perciò i modelli vengono prima importati su Blender® per eseguire un'operazione di pulizia della *mesh*, in modo da eliminare vertici e facce che non appartengono direttamente

alle statue.
Oltre a definire precisamente la geometria della statua, questa operazione è utile anche per alleggerire la dimensione del file e renderlo più maneggevole.

Per diminuire ulteriormente il peso si usa il modificatore *Decimate*, impostando il valore *Ratio* fino a quando le modifiche subite dalla statua sono percepili.
Questi due semplici passaggi hanno permesso per entrambe le opere di diminuire di circa 10 volte il numero di vertici, senza che il modello appaia diverso da come era stato esportato.

Dopo un'attenta analisi, alcune zone delle statue (le parti più nascoste, tra le pieghe delle vesti e della pelle) presentano la *texture* non corretta.
Su Blender® è possibile correggere questi errori dal momento che sono presenti tutti gli strumenti per il *texturing*.
A differenza di altri *software* di manipolazione di immagini, è possibile modificare una *texture* su Blender® in due modi: lavorando sulla foto in 2D (come fanno altri programmi, come Photoshop) oppure lavorando sull'immagine applicata al modello, osservando quindi direttamente le modifiche apportate

Figura 2.15 Il modello 3D definivo del Buddha del periodo Gupta, ottimizzato e con la *texture* migliorata.

Figura 2.16 Il modello 3D definivo del Buddha del Gandhara, ottimizzato e con la *texture* migliorata.

nello spazio tridimensionale.
Con lo strumento "timbro clone" si è velocemente riparata la *texture* nelle zone rovinate copiando e incollando altre parti simili dell'immagine.
Salvando la nuova immagine come *.jpeg* si ottiene la nuova *texture* pronta da usare, cambiando semplicemente il file nella gestione dei materiali.

2.3 Creazione dei contenuti aggiuntivi

2.3.1 *Ricostruzione delle parti mancanti*

Come già detto in precedenza, le due opere scelte sono prive degli arti superiori e inferiori e di una porzione dell'aureola.
Nella fase di selezione si è ragionato anche sulla destinazione d'uso dei modelli all'interno delle app di AR e VR.
L'applicazione più interessante è sembrata la ricostruzione 3D delle parti mancanti, in un'ottica di archeologia digitale: le parti ricostruite hanno bisogno di essere modellate realisticamente come se facessero parte della scultura, ma senza l'applicazione della *texture*, così che vi sia una netta distinzione tra l'opera originale e la ricostruzione.

La fase di ricostruzione ha richiesto un'intensa opera di ricerca guidata dalla dottoressa Ramasso e dal professor Olivieri.
Come riferimento sono state prese in considerazione opere simili (*Figure, 2.17, 2.18,2.19 e 2.20*) e dello stesso periodo provenienti da altri musei, selezionando statue del Buddha rappresentato nella stessa posa.

Dal confronto è emerso che molto probabilmente, in entrambe le

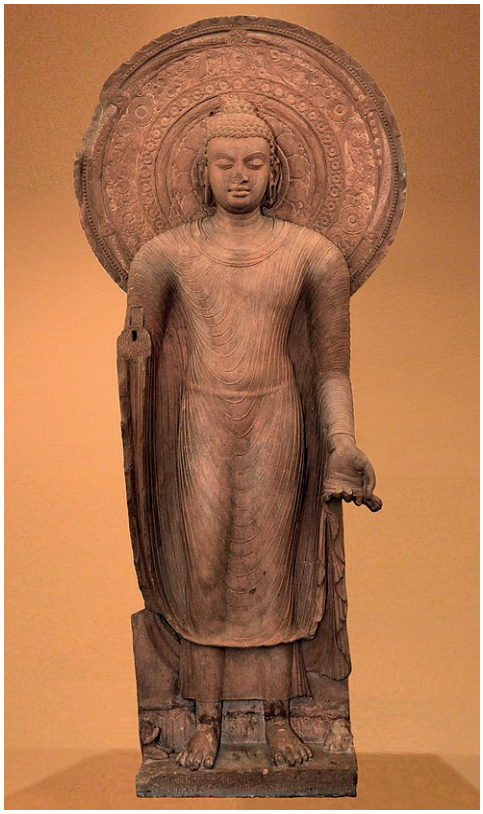


Figura 2.17 Buddha stante, Government Museum, Mathura, Uttar Pradesh, India.

Figura 2.18 Buddha stante, Freer Gallery of Art, Washington, DC, U.S.A.

Figura 2.19 Buddha stante, The Metropolitan Museum of Art, New York, U.S.A.

Figura 2.20 Buddha stante, Tokyo National Museum, Tokyo, Giappone.

sculture, la mano destra si presentava distesa, mostrando il gesto dell'*abhayamudrā*, mentre la sinistra era impegnata a tenere un lembo della veste, come già era supposto prima della ricerca.
Il paragone è stato di vitale importanza per quanto riguarda i piedi, le gambe e il podio su cui le statue poggiano.

Dopo aver compreso la direzione da prendere, inizia la fase di modellazione.
Esistono molti modi nel mondo 3D per avere dei manichini di base da cui partire per modellare il corpo umano.
Sicuramente uno dei migliori *software* è DAZ® Studio, un programma gratuito (con eventuali acquisti *premium*) utilizzato in larga misura da creatori di effetti speciali e videogiochi. Iscrivendosi sulla piattaforma è possibile scaricare sul proprio *account* una serie di pacchetti predefiniti: nel nostro caso è stato selezionato il pack "*Genesis8 Male*".
All'interno dell'app è possibile effettuare modifiche al *rig* del manichino e metterlo già in una posa specifica, operazione tralasciata in quanto questa azione può essere eseguita direttamente su Blender®.
Con un *add-on* il manichino può essere importato in Blender®: in automatico si presenta con il *rig* nella stessa posa con cui si è esportato da DAZ® Studio.
Entrando in *pose mode* si possono

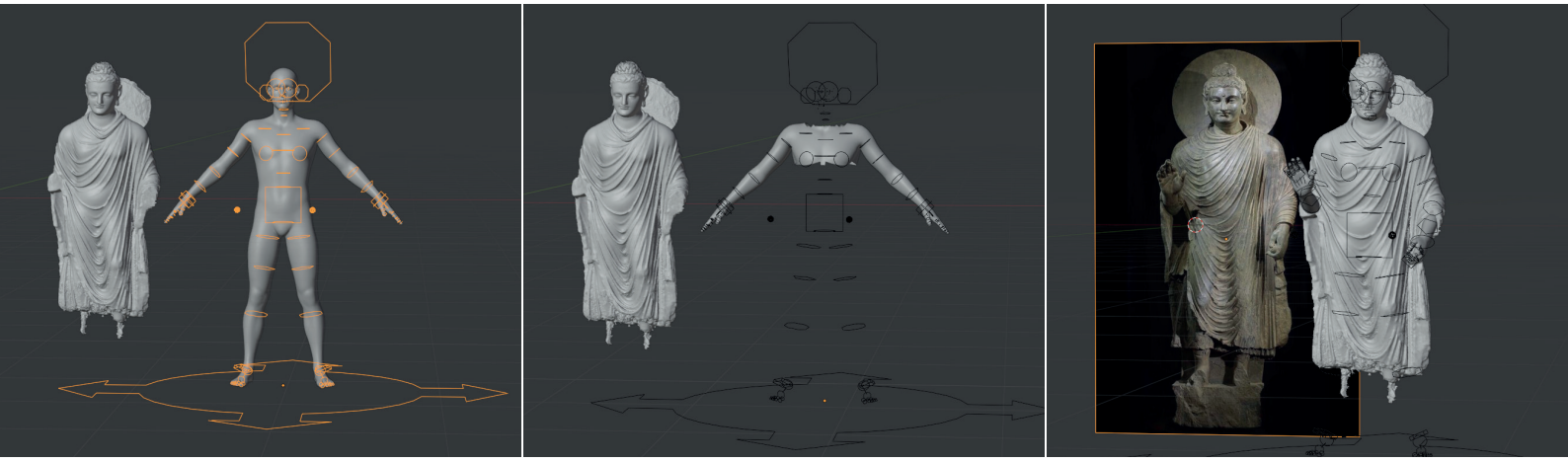


Figura 2.21 Le operazioni per fare prendere la posa al manichino.

muovere e ruotare i *bone*, fino a raggiungere la posizione che più si avvicina all'ipotetica statua originaria (*Figura 2.21*).
Per semplificare l'operazione si è scelto di eliminare le parti della *mesh* del manichino che non interessano, lasciando solo le braccia, e di inserire a fianco del rilievo della scultura un'immagine come *reference.*

Per i piedi e il podio del Buddha del Gandhara, con una ricerca sul web si è trovato un modello gratuito su Sketchfab, una piattaforma di condivisione di modelli 3D.
Il modello in questione è una statua del Gandhara del tutto simile a quella del MAO, proveniente dal Minneapolis Institute of Art.
Dopo averlo scaricato come file *.obj*, con un'operazione di pulizia *mesh* si elimina il resto della statua che non interessa.

Per il Buddha Gupta non è stato trovato nessun modello già pronto, per cui sia per le braccia che per le gambe ci si è serviti di DAZ® Studio come base di partenza.

Dopo aver posizionato i manichini ed eliminato le parti che non interessano, si procede con una fase di *sculpting*.
In Blender® è presente una sezione apposita: selezionando la *mesh* da scolpire ed entrando in *sculpt mode*, si possono selezionare diverse tipologie di pennello, ognuno dei quali è pensato per scopi diversi. Nel nostro caso ci si è serviti prevalentemente del *Clay Strips* per aggiunge-

Figura 2.22 La parte inferiore della statua del Minneapolis Institute of Art.

re materiale, *Grab* per deformare zone più ampie, *Pinch*, *Crease* e *Draw Sharp* per affilare gli spigoli o per fare delle sottili insenature e infine *Smooth* per rendere più liscia la superficie finale.
Con questi strumenti la superficie del manichino iniziale è stata resa più realistica e simile alle sculture. Oltre agli arti sono stati anche scolpiti i lembi delle vesti andati rovinati con il tempo e le due metà delle aureole mancanti (*Figura 2.23*).

In particolare, per quanto riguarda l'aureola del Buddha Gupta, si è scelto di non scolpire le decorazioni, ma di simularle attraverso l'uso di *bump map* e *normal map*.
Utilizzando Adobe®Photoshop si è ricostruita la vista frontale dell'intera aureola ruotando la porzione conservata attorno al centro.
Convertendo l'immagine in bianco e nero e lavorando sui livelli si trasforma l'immagine in una *bump map* che, proiettata correttamente sulla *mesh*, da l'illusione del rilievo. Infine viene eseguito il *baking* per ottenere una *normal map* (*Figura 2.24*) solo per la parte di aureola ricostruita, che ha lo stesso scopo

Figura 2.23 Dettagli e viste complessive dei risultati di ricostruzione dopo le operazioni di *sculpting*.

a

b

c

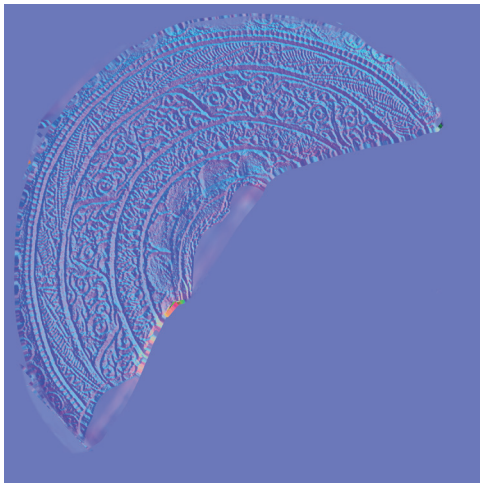


Figura 2.24 La trasformazione della *texture* iniziale (*a*) in *bump map* (*b*), convertita infine in *normal map* (*c*).

della *bump* ma è in formato a 3 colori, più facilmente leggibile da altri *software* di 3D.
Il *baking* viene eseguito anche per le restanti parti di entrambe le statue, così da ottenere alla fine dei modelli più leggeri e abbozzati facilmenti elaborabili dalle future app installate sugli *smartphone*, in cui il livello di dettaglio non manca ma è simulato dalle *normal map*.

Ricostruzione dell'ambientazione 2.3.2

Dal confronto con la dottoressa Ramasso e il professor Olivieri è emersa la possibilità di realizzare l'ambientazione dell'esperienza di realtà virtuale ispirandosi a *Gumbat*, un antico tempio risalente al regno del *Gandhara*.

Il sito archeologico è situato nella valle del fiume Swat nel Pakistan settentrionale, situato nei pressi dell'antica città di Bazira (oggi conosciuta come Barikot), uno dei centri abitati conquistati da Alessandro Magno posto più a Est.
Dal 1955 l'Italia ha condotto un'ininterrotta attività di scavi archeologici in questa zona grazie alla figura di Giuseppe Tucci e alla Missione Archeologica Italiana in Pakistan - MAIP[2].
Oggi il direttore della missione è lo stesso prof. Olivieri, grazie al quale si è potuto usufruire di documenti e immagini fondamentali per ricostruire correttamente l'edificio.

2. Missione Archeologica Italiana in Pakistan - MAIP, *https://www.ismeo.eu/portfolio_page/missione-archeologica-italiana-in-pakistan-maip,* ultima consultazione 30 agosto 2021.

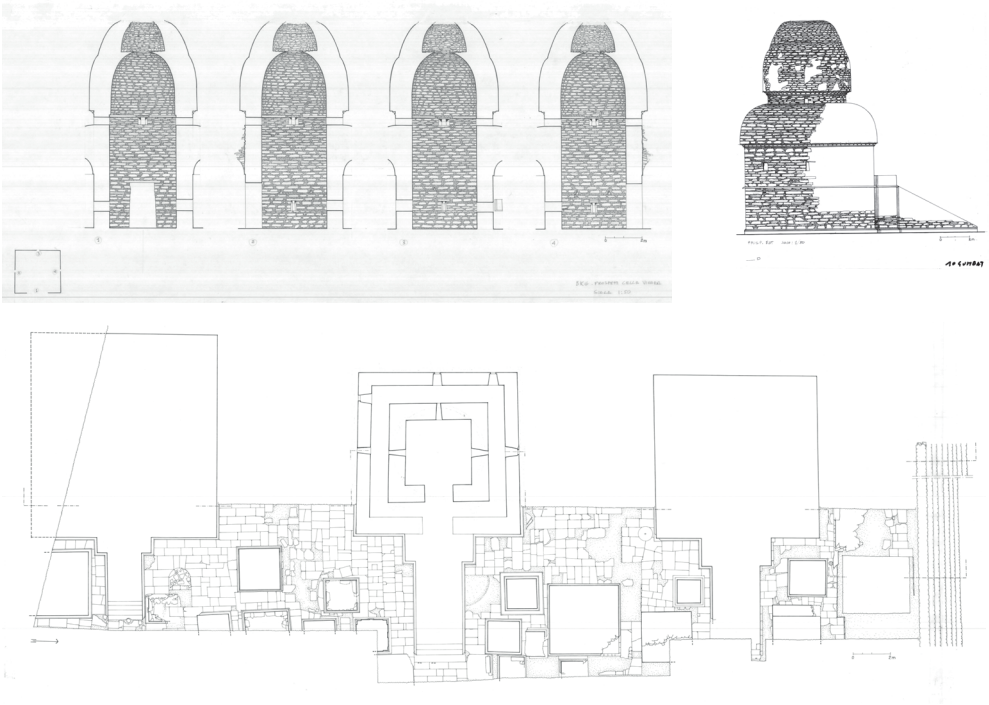


Figura 2.25 Alcune piante, sezioni e viste tra il materiale di *Gumbat* messo a disposizione.

Dopo aver allineato e ridimensionato tra loro le immagini fornite, su Autocad 2020 si è ricalcato il tempio in moda da ottenere le viste ortogonali e alcune sezioni.
Il documento ottenuto, esportato in *.pdf* è poi stato caricato in Blender®: ogni vista e sezione è stata allineata nello spazio tridimensionale per facilitare la modellazione 3D.
Partendo da queste linee è stato sufficiente estrudere e intersecare i volumi per creare le pareti, attivando l'opzione *snap* per essere sempre sicuri dei agganciarsi perfettamente alle linee di partenza, ed eseguire degli *spin* e *array* per la cupola e i piccoli contrafforti (*Figura 2.26*).

Posizionando la statua del *Gandhara* nella cella all'interno di *Gumbat* è apparso come le proporzioni tra i

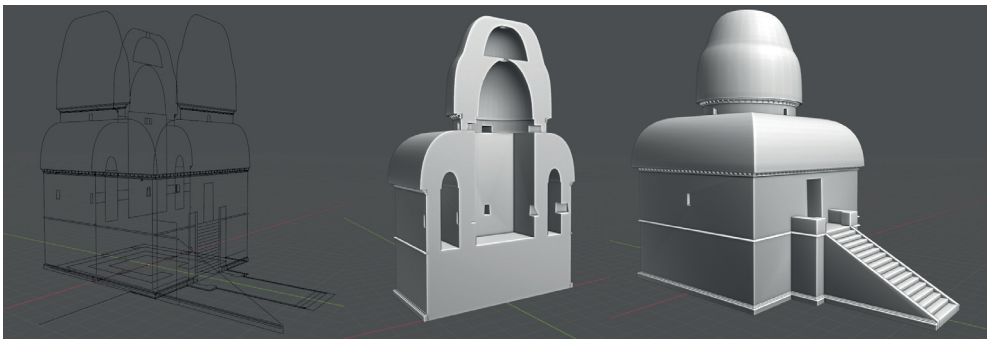


Figura 2.26 Le fasi di modellazione dell'edificio.

due siano troppo differenti, perciò si è scelto eseguire un adattamento dell'edificio per ridimensionarlo affinchè sia plausibile la presenza di una statua non molto grande (119 cm).
Le dimensioni sono state quindi scalate di un terzo (*Figura 2.27*), mantenendo alcuni elementi fissi, come le aperture e l'altezza del corridoio interno.
La rampa d'accesso ovviamente non è stata ridimensionata, ma sono stati tolti solamente alcuni gradini, mantenendo le caratteristiche proporzione delle scale del *Gandhara* (17 cm per l'alzato, 20 cm per la pedata).

Dopo la riduzione delle dimensioni si può quindi procedere con la creazione dei materiali. Originariamente il tempio si presentava

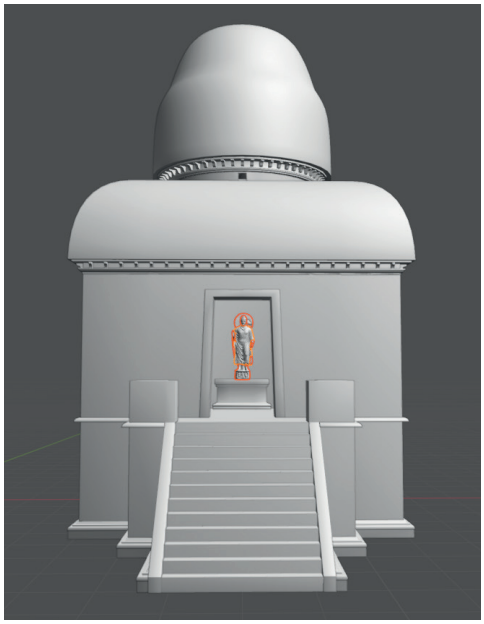


Figura 2.27 La statua posizionata all'interno del tempio con le dimensioni ridotte.

Figura 2.28 Un particolare della pavimentazione interna di *Gumbat*.

con le pareti e la cupola interamente intonacate, di un colore avorio piuttosto chiaro. C'è abbondante presenza di scisto grigio, di cui si costituisce l'architrave d'ingresso e la pavimentazione a lastre (*Figura 2.28*). Alcune foto dei sopralluoghi hanno permesso di avvicinarsi a un risultato vicino alla situazione originaria.

Alcuni accorgimenti sono stati presi per il materiale della scala, che presentava degli *stair-risers*, ovvero delle decorazioni di scene e simboli sacri legati alla vita del Buddha raffigurati in bassorilievo nell'alzato dei gradini. L'ultimo gradino presentava una fascia continua decorata con una girale di pianta di *pipal*, il resto della scala probabilmente dei simboli posti ai lati e al centro e ripetuti in sequenza. Su Photoshop CC 2021 si sono modificate le immagini per ottenere, similmente a quanto operato con l'aureola precedentemente, delle *texture color* e *normal*, per simulare il bassorilievo senza la necessità di scolpirlo per davvero (*Figura 2.29*).

Figura 2.29 Le *texture* ricostruite degli *stair-risers* presenti sui gradini della rampa d'accesso.

La creazioni dei materiali di *Gumbat* ha richiesto l'utilizzo di *texture* in alta qualità, scaricabili online da molti siti che offrono immagini gratuite, come AmbientCG e PolyHeaven (utilizzati entrambi scaricando texture in qualità 4K), e combinandole tra loro e insieme ad altri parametri all'interno dello *shader editor* di Blender®, uno strumento che utilizza il sistema dei *nodes* per realizzare gli *shader* (*Figura 2.33*).

Figura 2.30 Dettaglio dell'aspetto della scalinata.

Figura 2.31 Aspetto di *Gumbat* con le *texture* applicate.

Dopo aver impostato i materiali per ogni *mesh* dell'edificio, si procede anche in questo caso con il *baking*, in modo da ottenere solamente immagini senza il *set up* dei nodi, in quanto il *software* che verrà usato successivamente prevede un diverso controllo degli *shader* in cui è più conveniente trattare *texture* singole.

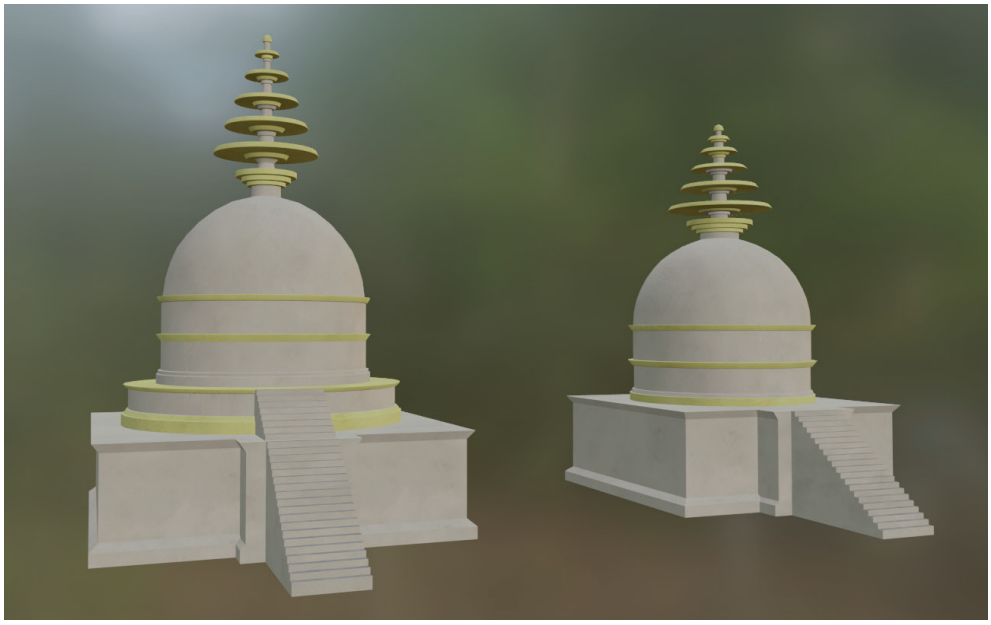


Figura 2.32 La ricostruzione dei due *stupa* con le *texture* applicate.

Con le stesse procedure, a partire dalle piante provenienti dagli scavi archeologici, sono stati ricostruiti anche i due *stupa* posti originariamente ai lati di *Gumbat*, due edifici la cui funzione principale era quella di conservare reliquie e di ricordare la vita terrena del Buddha.

Ricostruzione del paesaggio 2.3.3

Per creare un'esperienza maggiormente immersiva è stato ricreato anche il paesaggio in cui sono collocati i tre edifici.
Utilizzando l'*add on* gratuito "*Blender GIS*" è possibile aprire *Google Maps* all'interno di Blender® e selezionare la porzione di planisfero che si preferisce, impostando un

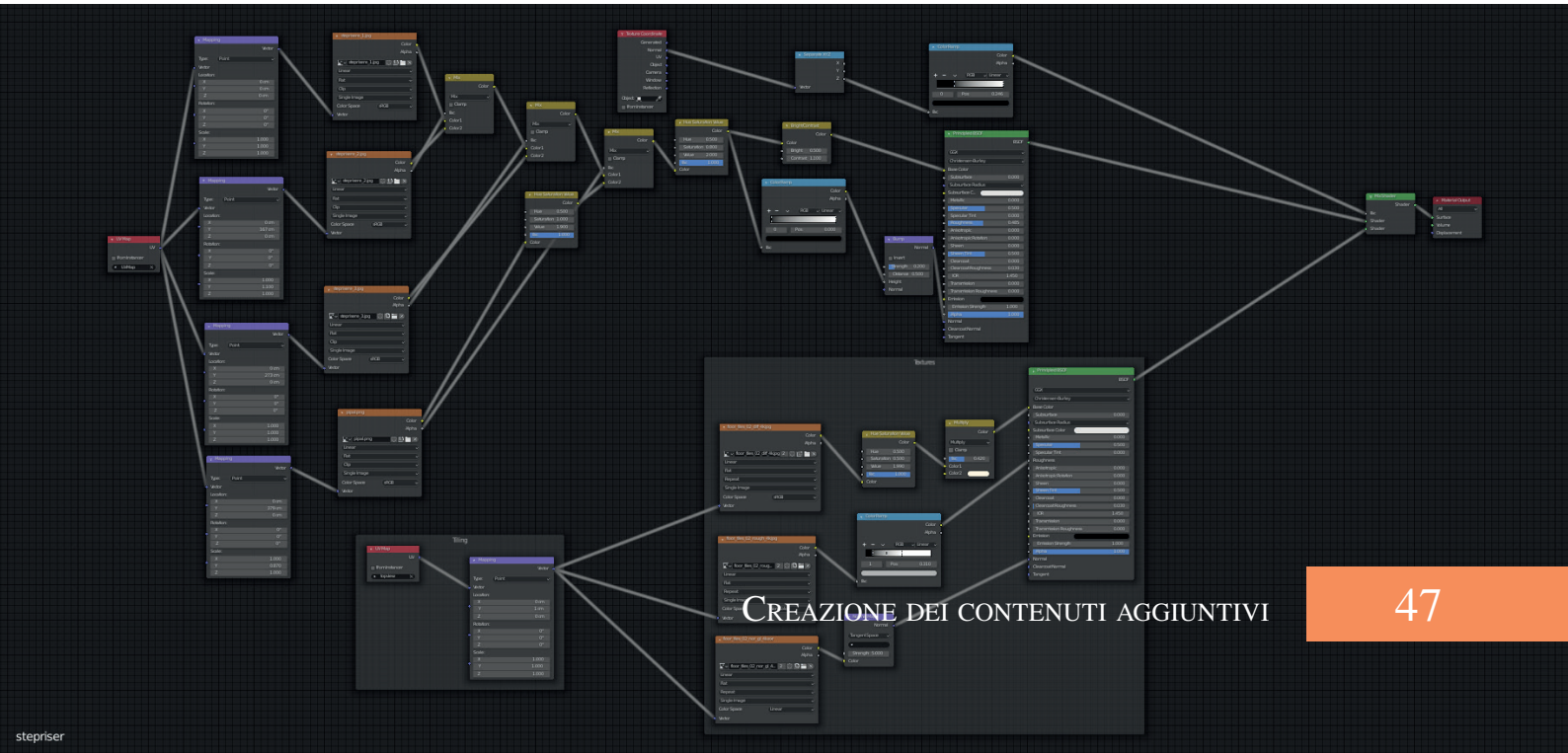


Figura 2.33 Il *setting* dei *nodes* per realizzare lo *shader* della rampa di scale.

sufficiente livello di zoom per ottenere una *texture* a risoluzione più alta (*Figura 2.34*).
L'immagine ottenuta viene trasformata con un apposito comando in una versione tridimensionale, grazie alla presenza dei dati GIS (*Geographic Information System*), ovvero le informazioni relative alla conformazione del terreno.
I dati sono infatti trasformati in *Displacement map* e automaticamente inserite in un *modifier* che deforma la *Image Plane* di partenza, applicando nel frattempo la *texture* (*Figura 2.35*).
La zona selezionata ovviamente è l'area che racchiude le montagne che circondano il sito archeologico di Gumbat.

Figura 2.34 L'area del planisfero selezionata.

Dato che la *mesh* generata presenta molti vertici e riducendone il numero si ridurrebbe il realismo del terreno, si è scelto di convertire la scena in un'immagine HDRI a 360°.
Le immagini HDRI vengono utilizzate nei *software* di computer grafica per due motivi principali: illuminare la scena, grazie alle caratteristiche del tipo di file che appunto presenta un intervallo dinamico più ampio rispetto alle foto tradizionali (da cui *High Dynamic Range Imaging)*, e servire da sfondo.
Siccome gumbat si trova piuttosto distante la soluzione è più che opportuna. Con un'apposita impostazione della camera di Blender, impostata su *Panoramic Equirectangular*, si ottiene un render a 360° (*Figura 2.36*).

Il terreno su cui poggiano fisicamente gli edifici è modellato con gli strumenti di *sculpting*, cercando di

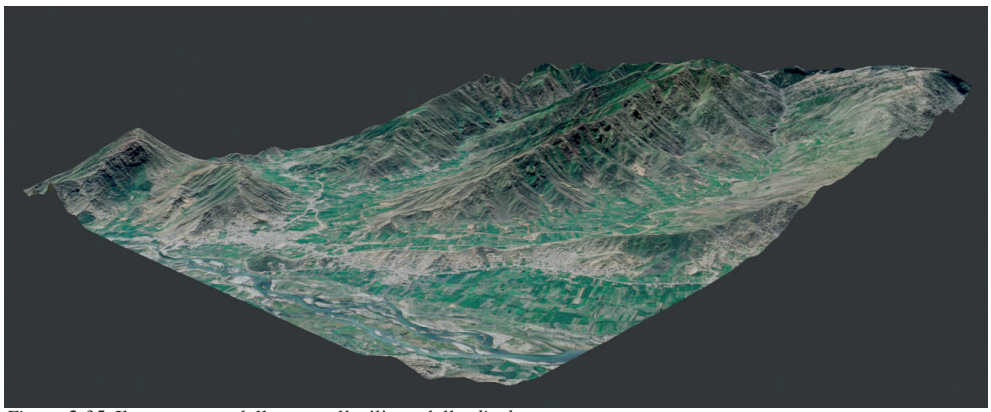


Figura 2.35 Il terreno modellato con l'utilizzo della *displacement map.*

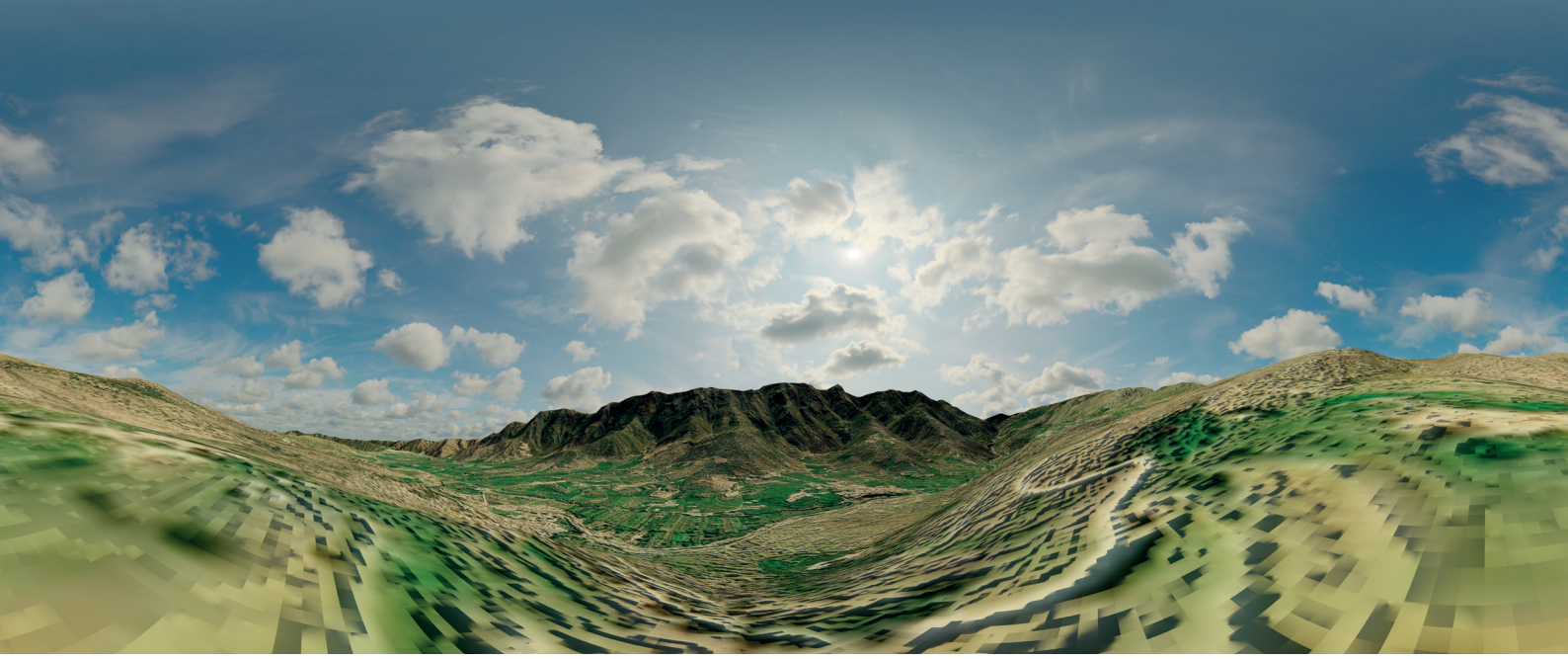


Figura 2.36 L'immagine HDRI a 360° utilizzata come sfondo. La parte pixelata vicino alla camera viene poi coperta da altre *mesh*.

riprodurre il paesaggio reale, documentato dalle fotografie fornite dal prof. Olivieri.

La realizzazione dei gruppi di pietre poste sui rilievi dietro il tempio ha richiesto l'utilizzo di tecniche di *scattering*, per disporre velocemente e in maniera casuale oggetti su una certa superficie.
Viene utilizzato uno strumento di Blender® chiamato *Geometry Nodes*: si modifica la geometria di partenza della superficie aumentandone i vertici e sostituendoli con degli altri oggetti, in questo caso le *mesh* di alcune rocce. Ogni vertice viene poi *randomizzato* per variare nella rotazione e nella dimensione, facendo apparire la distribuzione più realistica (*Figura 2.37*).

L'ambientazione di contesto per il progetto VR è così conclusa e si procede quindi alla creazione vera e propria delle esperienze.

Figura 2.37 Vista delle *mesh* del terreno con le rocce distribuite.

Capitolo 3

Il progetto di realtà aumentata

3.1 Operazioni preliminari

3.1.1 *Creazione del Target*

Prima di procedere alla realizzazione dell'esperienza di realtà aumentata su Unity®, è necessario capire quale metodo di riconoscimento della statua si vuole utilizzare.

Uno dei migliori e più diffusi SDK riguardanti la realtà aumentata è Vuforia™, uno strumento che lavora molto bene in collaborazione con Unity® e che permette il tracciamento di piani e oggetti tridimensionali in tempo reale.
Aprendo un profilo gratuito nel portale apposito, è possibile registrarsi come sviluppatori e avere accesso agli strumenti di generazione dei *Target*.
Il *Target* è un oggetto prestabilito che il motore di Vuforia™ riconosce nella scena reale e ne esegue il tracciamento nello spazio.
Le due tipologie di target più comuni sono *Single Image* e *3D Object*.

Le *Single Image* sono immagini bidimensionali che vengono spesso utilizzate come *marker*, ovvero dei punti di riferimento posti nei pressi dell'oggetto reale come sostituti, in quanto più facilmente tracciabili.
Nel nostro caso si vuole che l'oggetto da tracciare sia direttamente la statua posta nella sala del museo, perciò bisogna avanzare con la creazione di un *Target 3D Object*.

Sono richiesti alcuni prerequisiti affinché il modello virtuale possa essere trasformato in un *Target*: non devono esserci componenti flessibili nell'oggetto reale, come tessuti, la geometria deve essere articolata in maniera che Vuforia™ la possa distinguere con precisione dagli oggetti circostanti e allo stesso modo il materiale di cui è costituito deve mostrare alcuni aspetti che lo caratterizzano.
La statua del Buddha Gupta presenta tutti questi requisiti, perciò il tracciamento si può ipotizzare funzioni correttamente senza difficoltà.

All'interno di Model Target Generator (un *software* fornito da Vuforia™) è possibile quindi eseguire

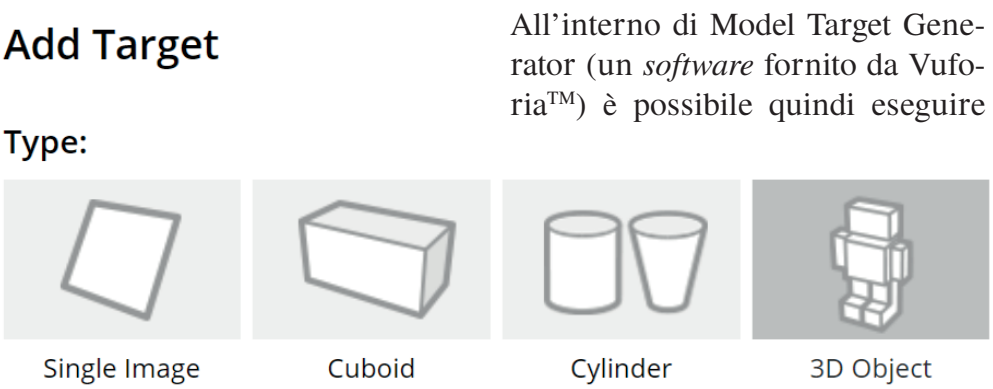


Figura 3.1 Le opzioni di selezione della tipologia di *Target* all'interno del portale di Vuforia™.

l'*upload* del modello digitale della statua, dopo che è stata precedentemente esportata da Blender® in formato *.obj*.
È importante che il file sia in scala reale, in quanto il motore di Vuforia™ esegue un migliore tracciamento se l'oggetto reale e il *Target* si presentano con le stesse identiche dimensioni.

Durante il procedimento per la creazione del *Target*, Model Target Generator richiede alcune impostazioni.
Innanzitutto confermare l'orientamento del file, perchè il sistema di assi cartesiani utilizzato in Blender® è diverso da quello utilizzato in Vuforia™ e Unity®: nel primo caso l'asse verticale è l'asse *z*, nel secondo si tratta dell'asse *y*, pertanto si deve correggere l'orientamento in fase di esportazione oppure all'interno del Generator.
Successivamente viene richiesto di selezionare l'unità di misura, che è già corretta se il modello è in scala 1:1. In seguito si può ottenere la *Guide View*, un file immagine che ricalca in maniera stilizzata l'immagine del modello 3D, inquadrato dalla stessa posizioni in cui si vuole che venga inquadrato al momento dell'utilizzo dell'app (*Figura 3.2*).
Si potrebbe anche impostare l'*Advance View*, utile se il *Target* andasse visto da più angolazioni, ma essendo il Buddha in questione addossato alla parete è sufficiente un riconoscimento solo frontale.
Si può ancora impostare un margine di errore di angolazione entro cui il riconoscimento avviene lo stesso senza che la sovrapposizione tra *Guide View* e statua inquadrata sia precisa al massimo, impostato a 5° di rotazione sia del piano *xy* che del piano *xz*.
Una volta terminata la procedura, il *Model Target* viene esportato con l'opzione apposita in un file *.unitypackage*, pronto per essere importato nel progetto su Unity®.

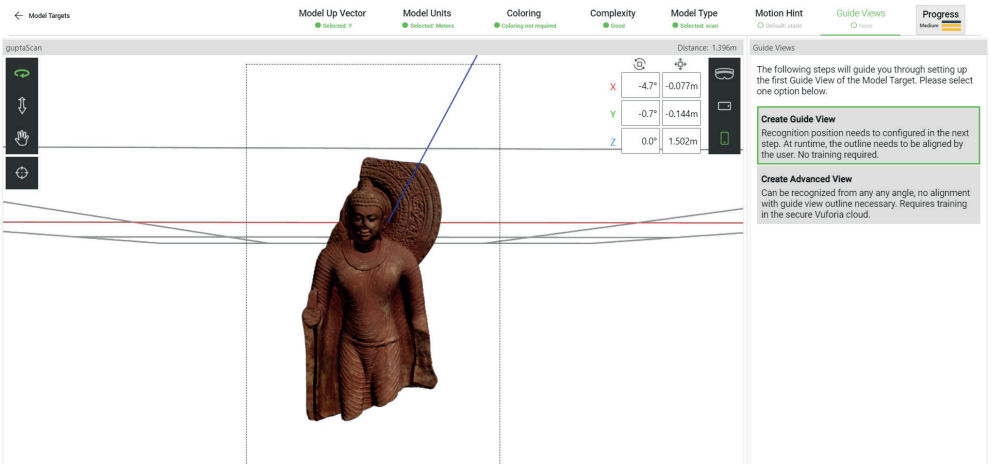


Figura 3.2 Una schermata del Model Target Generator, nella fase di creazione della *Guide View*.

Figura 3.3 L'interfaccia di Unity® 2020.3 alla prima apertura di un nuovo progetto.

3.1.2 *Avviamento di Unity®*

La realizzazione vera e propria dell'esperienza avviene grazie a Unity®, un *game engine* gratuito largamente utilizzato. Al suo interno è possibile combinare vari elementi come il *Target* e oggetti tridimensionali e programmare la loro interazione.

Il primo passo da compiere, dopo essersi registrati sul portale di Unity®, è fare il *download* della versione del *software* adatta ai nostri scopi. Siccome l'unico vincolo per questo progetto è dato dal supporto di Vuforia™, il quale si aggiorna anch'esso frequentemente, si è scelto di utilizzare l'ultima versione disponibile al momento, la 2020.3, una LTS (*long term support*).
Dal sito di Vuforia™ si scarica il file *.unitypackage* che permette di aggiungere il corrispondente SDK al progetto e si installa attraverso il *package manager* di Unity®, sostitutivo del vecchio *asset manager*.
Di base infatti Unity® ha alcuni pacchetti installati di base per mantenere ogni file di progetto più leggero. In base alle necessità di ognuno vanno dunque installati alcuni componenti, solitamente gratuiti, scaricabili direttamente da Unity® o, come nel nostro caso, da altre piattaforme.

Creato un nuovo progetto URP, l'interfaccia di Unity® si presenta divisa in finestre personalizzabili (*Figura 3.3*). Al centro si trova la *Scene*, l'insieme degli oggetti tridimensionali e bidimensionali, delle luci e della camera posizionati nello spazio (di default è già presente una scena d'esempio).
Alla sinistra la *Hierarchy* mostra la gerarchia degli oggetti nel progetto, classificandoli a seconda di quale *item* è "*child*" e quale "*parent*".
La finestra a destra contiene l'*In-*

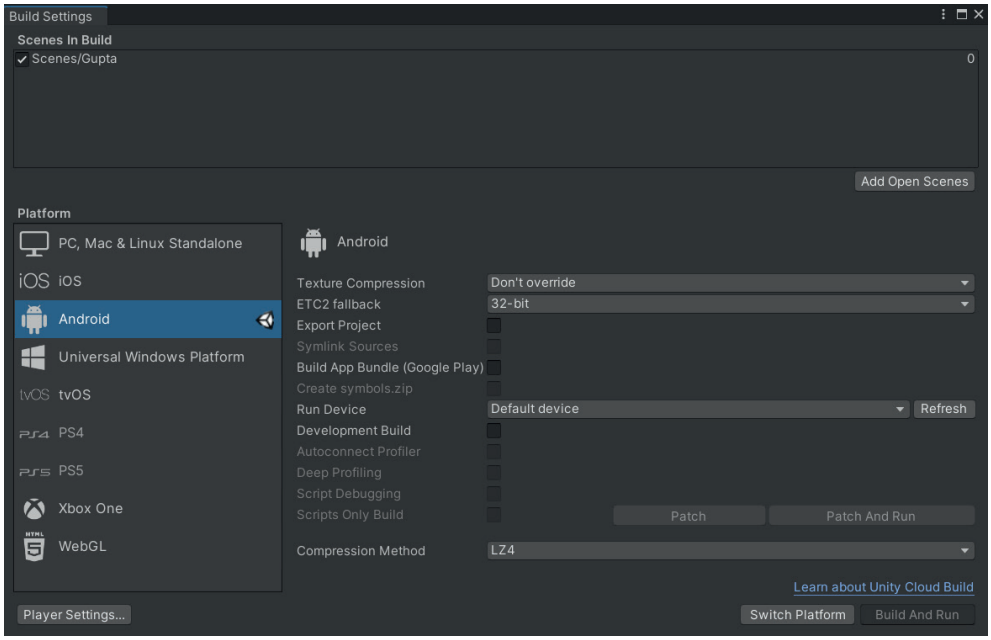


Figura 3.4 La finestra contenente le impostazione per le *build.*

spector, un pannello diverso per ogni oggetto, nel quale si possono trovare impostazioni e componenti a seconda della tipologia selezionata.
Infine in basso si trovano le cartelle contenenti i file da cui Unity® attinge per creare il progetto, in cui si possono caricare gli *asset* creati in precedenza, e gli strumenti di animazione, ovvero la *timeline* con il *dope sheet* e l'*Animator,* di cui discuteremo in seguito.

Un'operazione preliminare da eseguire è l'impostazione del dispositivo sul quale verrà testata l'esperienza. Aprendo la finestre del *Build Setting* (*Figura 3.4*) si trovano varie impostazioni, tra cui la possibilità di cambiare tra sistemi operativi. Per il nostro progetto, utilizzando un Redmi Note 10 Pro prodotto da Xiaomi, è quindi necessario selezionare Android e cliccare su *Switch Platform*, in modo che il progetto cambi le impostazioni per adattarsi al sistema operativo di arrivo.
Dalla stessa finestra è possibile in seguito realizzare le *build*, ovvero delle versione dell'app finali pronte per essere installate sullo *smartphone* per essere testate.

3.2 Realizzazione dell'esperienza

3.2.1 *Inserimento del Target*

Il primo passo da compiere è la creazione della camera. È necessario eliminare completamente la scena di default, perchè la camera utilizzata per esperienze di realtà aumentata è di una tipologia particolare, diversa nelle impostazioni da quella base utilizzata da Unity®.

Dopo aver aggiunto al progetto il *package* di Vuforia™, nel menu dei *game object* esso compare tra le categorie di oggetti, presentando tra le voci *AR Camera* (*Figura 3.5*).
La camera in questione agisce come la camera base di Unity®, quindi è lo strumento attraverso cui si osserva la scena nella *Game View.*
A differenza di quest'ultima è però collegata alla camera fisica del dispositivo su cui viene installata l'app e viene comandata da essa in base al tracciamento del *Model Target*, ovvero viene ruotata e traslata in maniera tale che oggetto reale e modello digitale coincidano perfettamente da ogni angolazione.
Nell'*Inspector* di Vuforia™, accessibile attraverso quello dell'*AR Camera*, è quindi necessario impostare il funzionamento del tracciamento solo in presenza di AR Core, che, come spiegato in precedenza, garantisce sui dispositivi Android il funzionamento della realtà aumentata.

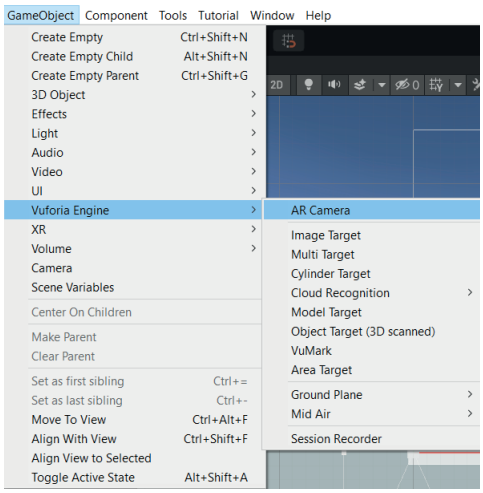


Figura 3.5 L'inserimento della ARCamera come camera principale.

È il momento di inserire il *Model Target* creato con il Generator.
Incorporato con l'oggetto entrato nella *Hierarchy* è presente la *Guide View* che si è scelto di impostare. Al momento dell'avvio dell'app e fino al riconoscimento del *Target*, sullo schermo compare solamente la *Guide View* e ciò che sta al di fuori della gerarchia del *Target.*

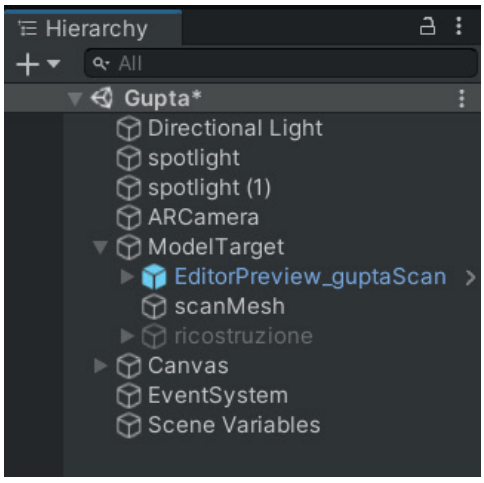


Figura 3.6 La gerarchia degli elementi ineriti nella scena.

Ogni oggetto posizionato come *child* del *Target* invece è automaticamente disattivato in partenza.
Per l'esperienza che si vuole costruire perciò si posizionano in questa sezione gli *asset* riguardanti la scansione 3D della statua e le parti ricostruite, mentre al di fuori come oggetti indipendenti trova posto il *Canvas*, l'*Event System* e l'illuminazione della scena (*Figura 3.6*).

Il *Canvas* è una sorta di contenitore al cui interno vengono organizzati tutti gli elementi riguardanti l'interfaccia utente, come scritte, istruzioni e pulsanti. Con la sua creazione viene in automatico aggiunto un *Event System*, un oggetto che non appare nella scena ma che si occupa di gestire le interazioni del *Canvas* e dei suoi elementi.

Il *Canvas* creato è stato pensato diviso in varie sezioni a seconda del momento dell'esperienza. I comandi si troveranno nella parte bassa dello schermo, come è comune alla maggior parte delle app in circolazione. Per mettere maggiormente in risalto i pulsanti che compariranno in questa zona, è presente un rettangolo di sfondo, denominato *backgroud*, riempito con una campitura bianca con *alpha* a 60, così da rendere visibili i comandi e allo stesso tempo lasciare intravedere lo spazio inquadrato dalla fotocamera reale (*Figura 3.7*).
Gli altri elementi inseriti all'interno del *Canvas* per comodità sono raggruppati in più blocchi: per ogni schermata diversa si crea un *Empty*, un oggetto vuoto ma presente nello spazio, che funziona come raccoglitore per controllare allo stesso tempo i suoi "figli".
Nella schermata iniziale oltre al *background* è attivo l'*empty start*, che contiene al suo interno il testo con le istruzioni iniziali per l'utente.

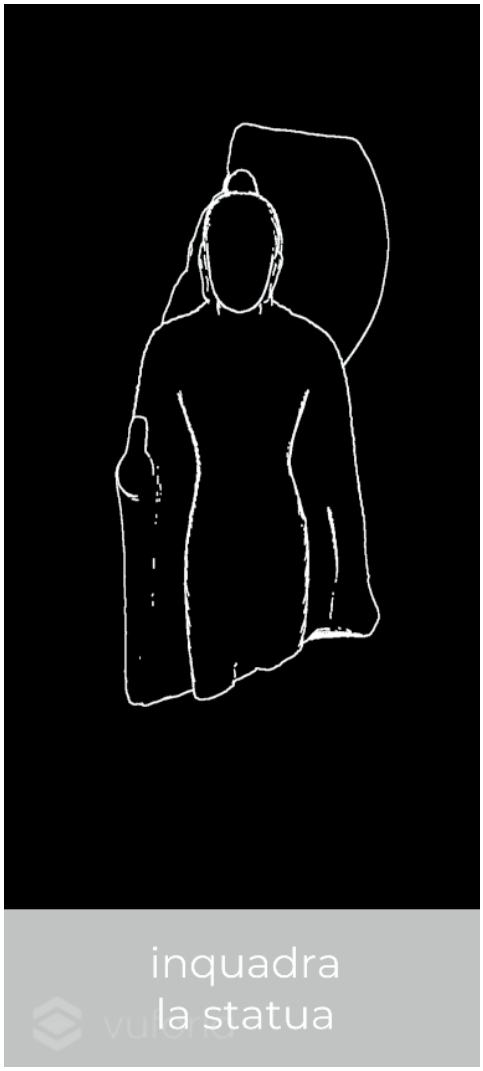


Figura 3.7 La *Guide View* e il *background* nella schermata iniziale.

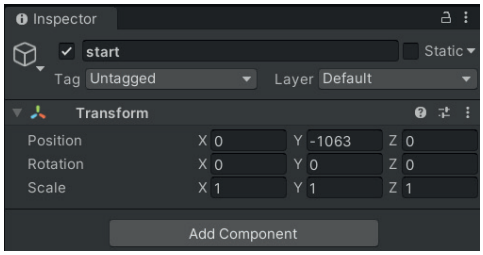


Figura 3.8 La finestra dell'*Inspector* con il *checkbox* in alto a destra per l'attivazione dell'oggetto.

Per attivare e disattivare un oggetto, nell'*Inspector* relativo si seleziona o deseleziona il *checkbox* posto in alto, tenendo presente che l'attivazione di un oggetto porterà all'attivazione di tutti i suoi figli (*Figura 3.8*).

Alcuni *script* all'interno di Unity® sono già preimpostati: solitamente riguardano le azioni attivate dai bottoni dell'interfaccia. In questo caso il *package* di Vuforia ha preimpostato una serie di azioni possibili in due momenti, quando il *Target* viene riconosciuto e quando viene perso.
Sotto la voce *On Target Found* viene quindi specificato che l'*Empty start* (e tutto ciò che esso comprende) venga disattivato, grazie al comando *Set Active*, e parallelamente venga attivato un altro *Empty* appartenente al *Canvas* con l'interfaccia della seconda schermata (*Figura 3.9*).
Il comando *Set Active* risponde a una variabile di tipo *booleano*, che può presentare due valori, uno *true* e uno *false*.
Perciò quando il *checkbox* è spuntato corrisponde al valore *true* (e l'oggetto viene attivato), quando è deselezionata corrisponde a *false*, con la disattivazione dell'*Empty*.
L'opzione all'inizio denominata *Consider target as visible if its status is* permette di decidere se attivare il riconoscimento del *Target* anche se non è completamente visibile attivando la voce *Tracked* or *Extended Tracked*.

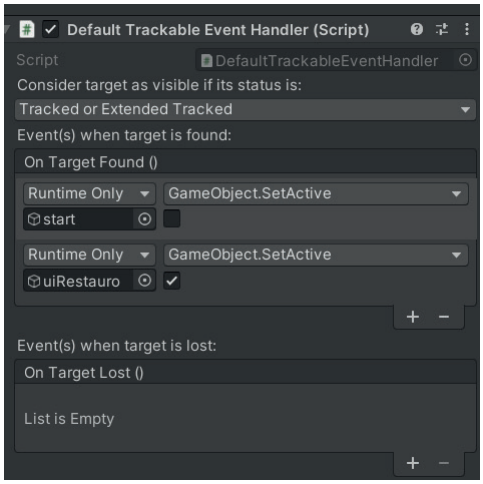


Figura 3.9 Il componente relativo al riconoscimento del *Target*.

Il layout 3.2.2

Ad eccezione del passaggio dalla schermata iniziale con l'istruzione al secondo step dell'esperienza, che avviene semplicemente quando il motore di Vuforia™ riesce a individuare il *Target*, per la navigazione tra le varie fasi c'è bisogno di premere dei comandi, al quale si danno delle istruzioni.
Pur essendoci delle immagini di default presenti su Unity® che posso-

Figura 3.10 La barra di navigazione

no essere utilizzate come pulsanti, si può utilizzare un qualsiasi programma di grafica vettoriale per realizzare le proprie icone personalizzate.
Su Adobe® Illustrator sono state quindi disegnati dei tracciati seguendo delle forme piuttosto standard, comprensibili da chiunque, che vanno a costituire la barra di navigazione (*Figura 3.10*).
Gli elementi presenti sono il tasto play, il tasto pausa, un freccia per tornare indietro, una freccia verticale (con la corrispettiva verso il basso) per aprire un pannello aggiuntivo e un pulsante rettangolare per far iniziare la ricostruzione (*Figura 3.11*).
È presente solamente la freccia per tornare indietro in quanto, una volta iniziata l'esperienza, le schermate rimanenti a parte le due iniziali si cambiano tra loro attraverso altri comandi discussi più avanti.

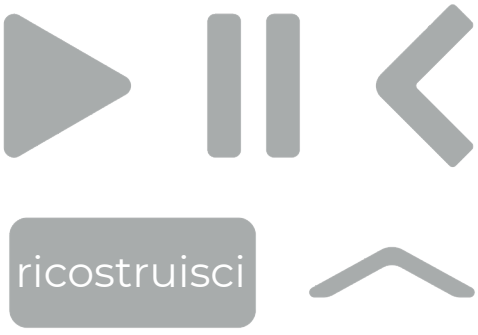


Figura 3.11 Le icone per la navigazione all'interno dell'app.

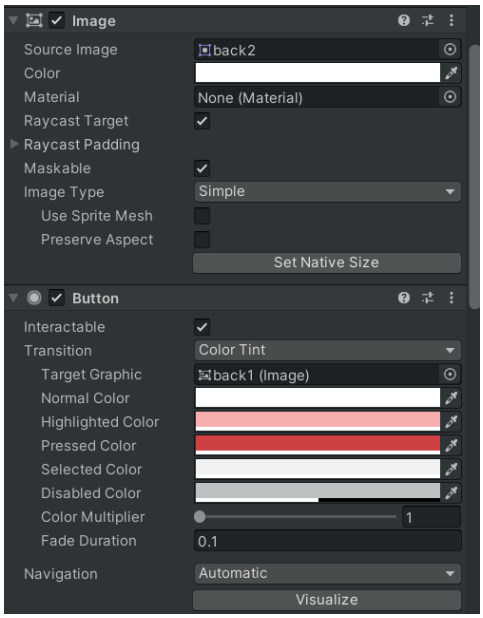


Figura 3.12 La configurazione dei colori di un *button* durante i vari momenti dell'interazione.

Le immagini vengono aggiunte agli asset di Unity® attraverso una specifica azione: per essere riconosciute come pulsanti, e quindi ottenere il tracciamento dell'area cliccabile dall'utente, bisogna aggiungerle come immagini *sprite*, che può essere indicato nell'*Inspector* relativo.
Nel menu a tendina, sotto la voce *GameObject - UI*, si può dunque inserire un oggetto *Button*, nel cui *Inspector* si possono trovare alcune impostazioni aggiuntive oltre al classico *Rect Transform*, che serve per posizionarlo all'interno del *Canvas*.
Nella casella *Source Image* si inse-

risce l'immagine *sprite* che si vuole appaia sullo schermo, mentre all'interno del *Component Button* si possono impostare i colori che questo avrà nei diversi momenti dell'interazione dell'utente.
Essendo un pulsante che richiede semplicemente un tocco sul *touchscreen*, sono impostati due colori, uno bianco per la situazione a riposo, uno rosso per il momento in cui viene premuto.
In aggiunta si imposta anche un terzo colore, l'*Highlighted Color*, nel caso ci fosse uno *smartphone* che utilizza un cursore al posto del touch, per evidenziare la preselezione del tasto (*Figura 3.12*).
Le stesse opzioni sono state quindi selezionate per tutti i *Button* inseriti.

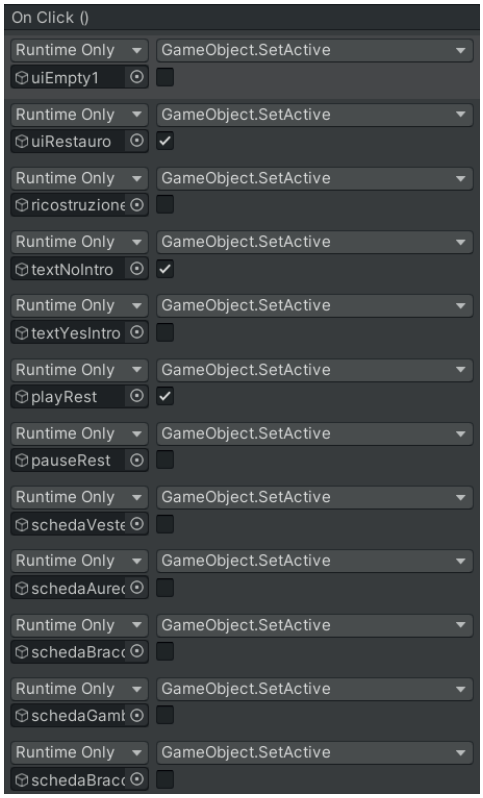


Figura 3.13 La configurazione dei colori di un *button* durante i vari momenti dell'interazione.

Similmente a quanto fatto in precedenza con il rilevamento del *Target,* anche per i *Button* esiste la possibilità di attivare un'azione quando premuti.
Nella finestra *OnClick* di ogni pulsante si seleziona su quale *GameObject* eseguire l'azione: per tutti i button si è usato prevalentemente il comando *Set Active*, per attivare o disattivare l'oggetto di riferimento (*Figura 3.13*).
In questo modo ogni volta che si clicca un bottone, esso stesso o il suo *Empty* di riferimento scompaiono per fare spazio all schermata successiva (o precedente). È importante fare attenzione, nel caso dell'alternanza tra *Play* e *Pause*, che quando appare una nuova schermata essa si presenti sempre allo stesso modo, anche quando l'ultima volta che è stata aperta era diversa. Perciò ogni step partirà sempre con il tasto *Pause* attivo e il tasto *Play* disattivato.
Allo stesso modo si imposta il pannello con le informazioni aggiuntive: all'apertura di ogni finestra esso sarà sempre nascosto inizialmente.

L'inserimento del testo all'interno del pannello viene eseguito attraverso uno *scroll rect*.
È infatti abitudine ormai di tutti quando si utilizza un'applicazione che prevede la lettura di un testo

Figura 3.14 La configurazione del pannello di testo aperto.

su un dispositivo *touchscreen* di far scorrere il paragrafo verso il basso per continuare la lettura.
Oltre a una questione di comodità di utilizzo è anche necessario inserirlo per permettere di non occupare tutto lo schermo dello *smartphone* e tenere una dimensione del *font* comunque leggibile senza difficoltà.

Si procede quindi all'inserimento di un'*Image* della dimensione della finestra di testo che si vuole appaia, denominata *Scroll Area*.
Su di essa viene aggiunto un componente *Scroll Rect* e un componente *Mask*, da attivare entrambi in seguito.
Mask fa in modo che solo la porzione di testo compresa nella *scrollArea* sia visibile, mentre *scroll rect* si occupa dello scorrimento del testo con il semplice trascinamento verso il basso (e anche verso l'alto per tornare indietro). Come *child* della *scrollArea* viene inserito il contenitore del testo, il *textContainer*, al cui interno è imparentato il testo vero e proprio (*Figura 3.15*).
Dopo aver regolato la dimensione del *textContainer* in modo che tutti i caratteri siano inclusi, si può attivare la *checkbox* del componente *Mask*.
Per disattivare il pannello ovviamente va premuta nuovamente la freccetta orientata verso il basso, nella cui sezione *OnClick* è programmato lo spegnimento della *scrollArea* e dei suoi *children*.

Figura 3.15 La gerarchia per la costruzione del testo scorrevole.

3.2.3 *Le animazioni*

Come detto in precedenza, il modello derivato dalla scansione 3D e la relativa ricostruzione delle parti mancanti realizzata su Blender® vengono inserite nella *Hierarchy* come *children* del *Model Target*, così da attivarsi solamente al riconoscimento del *Target* da parte di Vuforia™.
L'intenzione dell'esperienza è che, dopo una descrizione iniziale contenenti le informazioni generali dell'opera, vi sia la possibilità per l'utente di esplorare uno ad uno gli elementi più importanti della statua, per approfondire la visita e poter osservarla interga come poteva essere un tempo.
Dopo alcuni ragionamenti si è pensato che la maniera più efficace per accedere ad ognuno degli elementi fosse direttamente cliccare su di esso.

Per fornire un'ulteriore informazione all'utente su quale sia la *mesh* attiva in un certo momento è utile che questa presenti un materiale differente rispetto a quelle inattive.
Si parte quindi da uno *shader* di base comune tra tutte le *mesh* (ad eccezione del modello originale che presenta la *texture* della scansione). Per rendere più visivamente interessante l'utilizzo, il cambio di colore può essere effettuato con un'animazione, rendendo così la transizione dei materiali più graduale.
Per preparare le animazioni sono necessari due strumenti situati nella parte inferiore dell'interfaccia di Unity®: *Animation* e *Animator*.
La finestra *Animation* è dove si prepare concretamente la sequenza da animare, selezionando la *mesh* desiderata e creando una nuova clip. Per i nostri scopi bisogna deselezionare l'opzione di default che rende attivo il *loop* della clip animata: si vuole infatti che la transizione avvenga una volta sola e non riparta immediatamente da capo.
Nella *Timeline* dell'*Animation* si può scegliere la proprietà della *mesh* di cui si vuole eseguire l'animazione. In questo caso si tratta del colore dello *shader*, presente nell'*Inspectror* come *Base Map*. Aggiungendo un *keyframe* all'inizio e uno alla fine della durata desiderata, il *software* provvede automaticamente all'in-

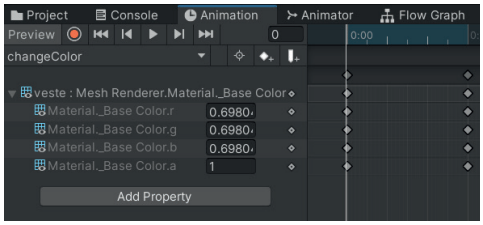


Figura 3.16 La finestra dell'*Animation* con la visualizzazione *dope sheet*.

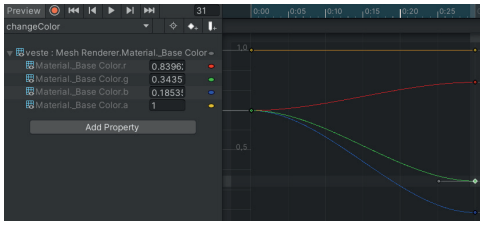


Figura 3.17 La finestra dell'*Animation* con la visualizzazione delle curve.

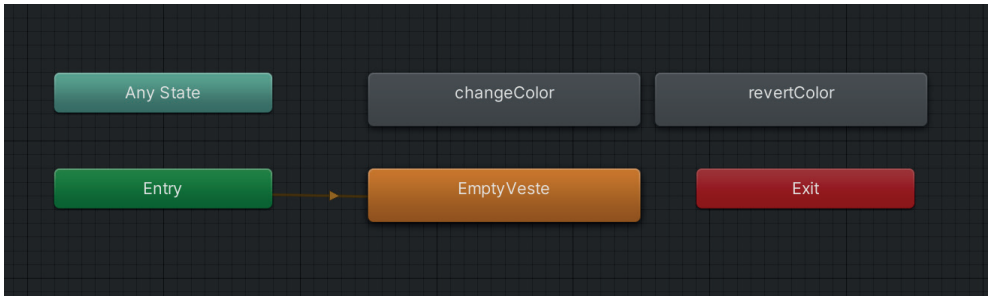
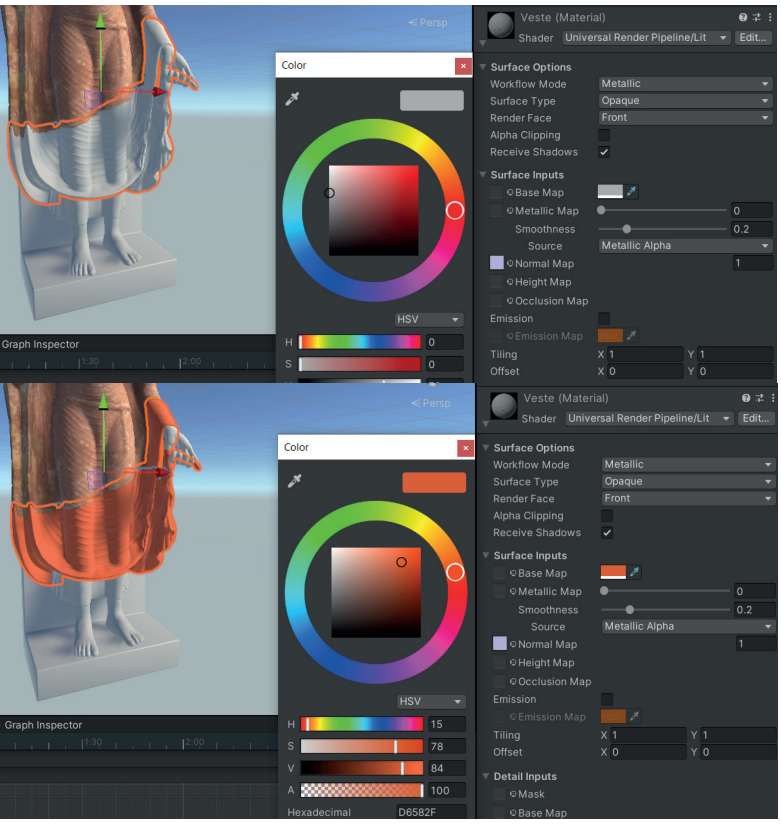


Figura 3.18 La finestra dell'*Animator* con la visualizzazione delle connessioni tra le clip.

terpolazione tra questi due valori secondo una curva molto morbida. Oltre alla finestra in cui sono presenti esclusivamente i *keyframe* (*Figura 3.16*) è possibile cambiare per visualizzare le curve create ed eventualmente modificarle per cambiare l'andamento della transizione tra i due valori della *Base Map* (*Figura 3.17*).

Figura 3.19 I pannelli con le impostazioni dei due *shader*.



Per ognuna delle *mesh* sono state create due animazioni, una che passa dal grigio chiaro iniziale al rosso (colore che indica la selezione) e una che esegue la transizione opposta ritornando al colore iniziale (*Figura 3.19*).
Entrambe le clip sono molto brevi perché come per il cambio di colore dei *button* deve essere appena percettibile la sfumatura intermedia, per questo la durata è di soli 0.5 secondi. Il progetto su Unity® è impostato a 60 *fps*, perciò per raggiungere la durata desiderata l'animazione deve essere lunga 30 *frame*.

Nella finestra dell'*Animator* vengono automaticamente inserite le clip create sotto forma di blocchi.
In questa sezione si possono collegare tra di loro e programmare grossolanamente, indicando ad esempio l'avvio automatico di una clip all'attivazione del suo *GameObject*.
Siccome vogliamo che all'inizio le varie *mesh* rimangano statiche, è necessario creare un *Empty state* per ognuna, ovvero una clip vuota che si imposta come *Layer Default*

State, riconoscibile dal colore arancio (*Figura 3.18*).

La freccia di collegamento tra *Entry* e l'*Empty* indica che all'attivazione della *mesh* parte la clip *Empty*, perciò non accade nulla.
In questo modo sono presenti le clip che potranno essere gestite successivamente con degli *script* personalizzati, per ottenere un maggior controllo su di esse.

3.2.4 *Le interazioni con gli oggetti*

Per rendere cliccabile un oggetto tridimensionale all'interno di Unity® c'è bisogno che abbia una parte di superficie predisposta alla collisione. Infatti il meccanismo dietro il *click* è gestito dal *raycast*: dal punto sul *touchscreen* in cui si clicca parte un "raggio" invisibile che prosegue fino a quando non incontra lungo la traiettoria un oggetto specifico, dotato appunto del *component Collider*, che è l'unica caratteristica che rende il *gameObject* visibile al *raycast*.
Pertanto nell'*Inspector* delle *mesh* viene aggiunto il *Component Collider 3D*.
Ne esistono di diversi tipi, partendo da alcuni solidi base come cubo, sfera e capsula oppure *Mesh Collider*. Quest'ultimo, che è quello utilizzato nel progetto, crea un *collider* a partire dalla geometria dell'*object,* perciò qualunque punto della *mesh* può essere intercettato dal *ray*.

Predisposti tutto l'occorrente per far interagire gli elementi, si passa alla creazione degli *script*.
Unity® utilizza il linguaggio di programmazione C#, un linguaggio che prevede l'utilizzo di classi, funzioni e variabili. È possibile studiarlo in funzione dell'utilizzo che se ne può fare all'interno di Unity® grazie a dei corsi disponibili gratuitamente sulla piattaforma Unity Learn.

Per coloro che non provengono dal mondo dell'informatica e della programmazione (come in questo caso) e sono più orientati ad apprendere visivamente, è disponibile un *tool* gratuito che permette di semplificare i lavori e procedere alla programmazione senza doverla studiare in maniera approfondita.
Lo strumento in questione, chiamato Bolt, sostanzialmente trasforma le righe di codice più frequentemente utilizzate in blocchi di nodi, connettibili e combinabili tra di loro similmente a quanto si fa nello *Shader Editor* di Blender®.
Questo metodo di programmazione è detto *visual scripting* e consente di effettuare tutte le operazioni della programmazione manuale, vedendo in tempo reale (durante il *play mode* della scena) il *flow* degli *input* ed eventualmente correggendo gli errori mentre questo è in funzione.

Per preparare lo *script* si deve aggiungere all'oggetto desiderato il *Component Flow Machine*, impo-

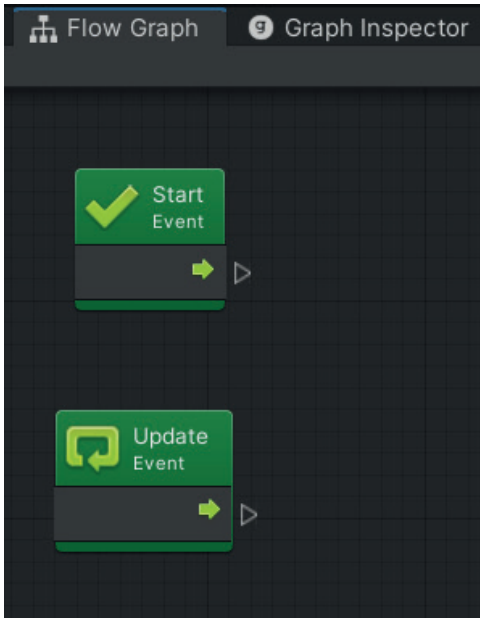


Figura 3.20 La finestra del *Flow Graph* con i blocchi di default.

standolo su *Embed*.
All'apertura del *Flow Graph*, ovvero la finestra in cui si gestiscono i nodi, sono già presenti due unità, *Start* e *Update* (*Figura3.20*).
Il primo si attiva una sola volta all'attivazione dell'oggetto, mentre *Update* si aggiorna a ogni *frame*.
Siccome si vuole che dopo il riconoscimento del *Target* non accada nulla se non la semplice attivazione delle parti ricostruite, il blocco start può essere eliminato.

Per indicare che il *ray cast* si avvii solamente quando avviene il tocco sullo schermo si può utilizzare il blocco *Branch*, che utilizza un *input booleano* per restituire due possibili risultati (e quindi *flow* differenti).
L'*input* che questo richiede è denominato *GetMouseButton*, impostato sullo 0, ovvero il *click* sinistro del mouse, che diventa il semplice tocco del *touchscreen*.
Quando c'è il tocco, e quindi l'*input* restituisce il valore *booleano True*, un altro *Branch* fa in modo che il *ray cast* si avvii solo per la parte superiore dello schermo in cui non c'è la barra di navigazione del *Canvas*, così che un singolo tocco non inneschi azioni sull'interfaccia e sulla scena contemporaneamente.
Avverate queste due condizioni si avvia il *ray cast*, partendo dal punto sullo schermo in cui è stato toccato, grazie al comando *ScreenPointToRay* (*Figura 3.21*).

Nel blocco *Mesh Collider Raycast* si inserisce il componente *Collider* che si intende colpire ed eventualmente la distanza massima che il raggio deve percorrere.
L'*output* dell'unità presenta anch'esso un valore *booleano*: se il raggio colpisce il *Collider* in questione il valore è *True*, se invece colpisce una *mesh* diversa o non colpisce niente diventa *False*.
Il *flow* in questo caso continua solo quando il valore restituito è *True*.

Da qui in avanti le operazioni eseguite da Bolt sono molto semplici: si tratta del comando *Set Active* già visto precedentemente per i *Button* del *Canvas* e della gestione delle animazioni.
Il blocco *AnimatorPlay* richiede un input di tipo *string*, cioè una serie di caratteri uguali identici a quello che

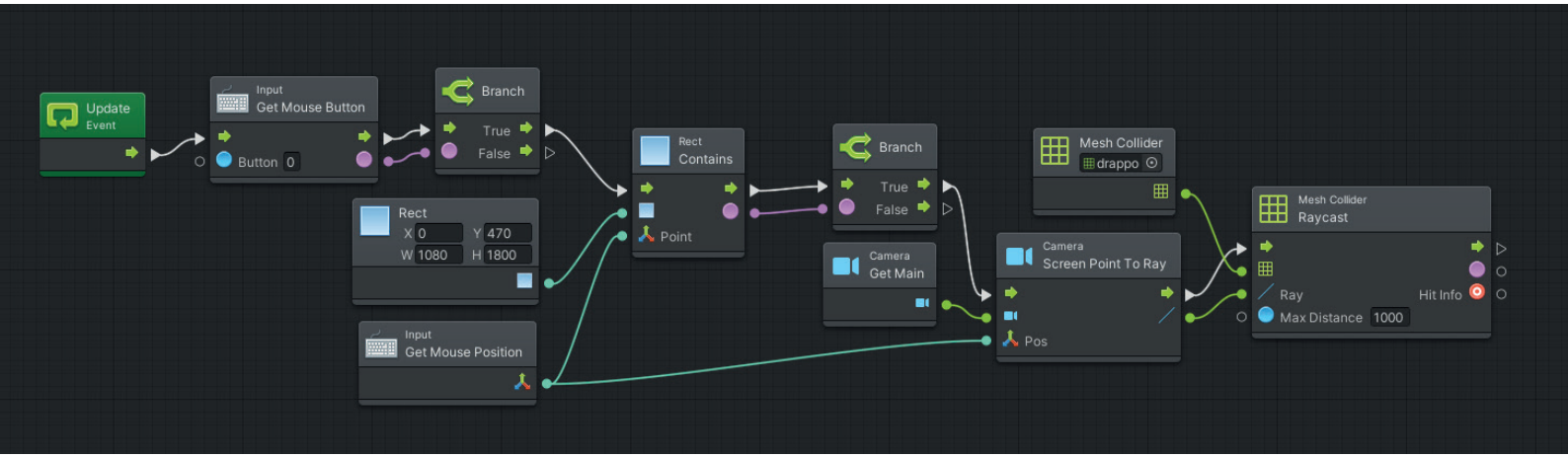


Figura 3.21 La prima parte dello *script* che attiva il *raycast*.

si vuole richiamare. In questo caso l'*input* deve avere la stessa dicitura della denominazione della clip animata che si vuole riprodurre (Figura 3.22).

La gestione di questi ultimi elementi è così organizzata: alla collisione del raggio con il *Collider*, si avvia l'animazione che porta la *mesh* in questione dal grigio al rosso e allo stesso tempo partono tutte le altre che invece fanno tornare gli altri oggetti al colore originario.
Parallelamente si attiva la scheda informativa relativa a quel dettaglio e si disattivano le altre. In questo modo facendo disattivare anche ciò che era già disattivato e avviando animazioni per tutti gli elementi, agli occhi dell'utente appare che cambi solamente la *mesh* cliccata e quello selezionata appena prima. Per lo sviluppo dello *script* invece è di gran comodità perché si può eseguire semplicemente una serie di copia e incolla per tutti i dettagli, sostituire i nomi delle stringhe da attivare e cambiare alcune *checkbox*, così da completare i *Flow Graph* di tutti con lo stesso schema ripetuto (*Figura 3.23*).

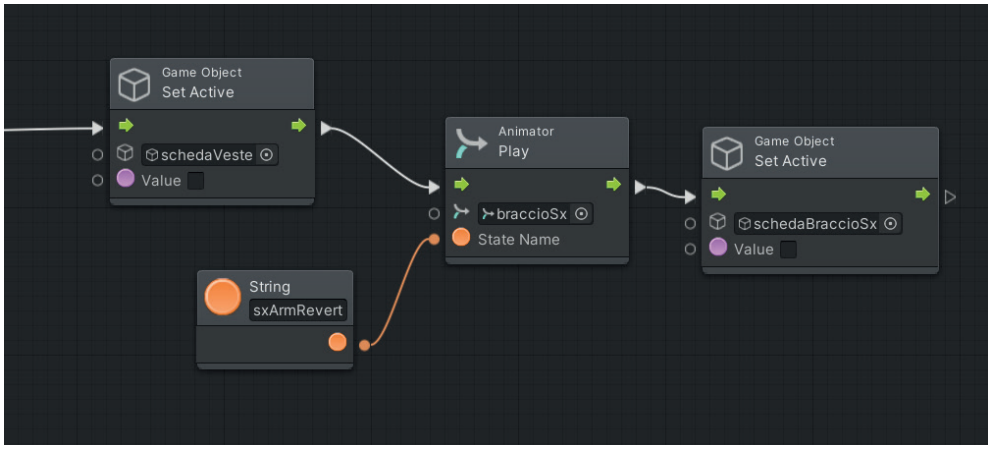


Figura 3.22 Il collegamento con la *string* che avvia l'animazione desiderata.

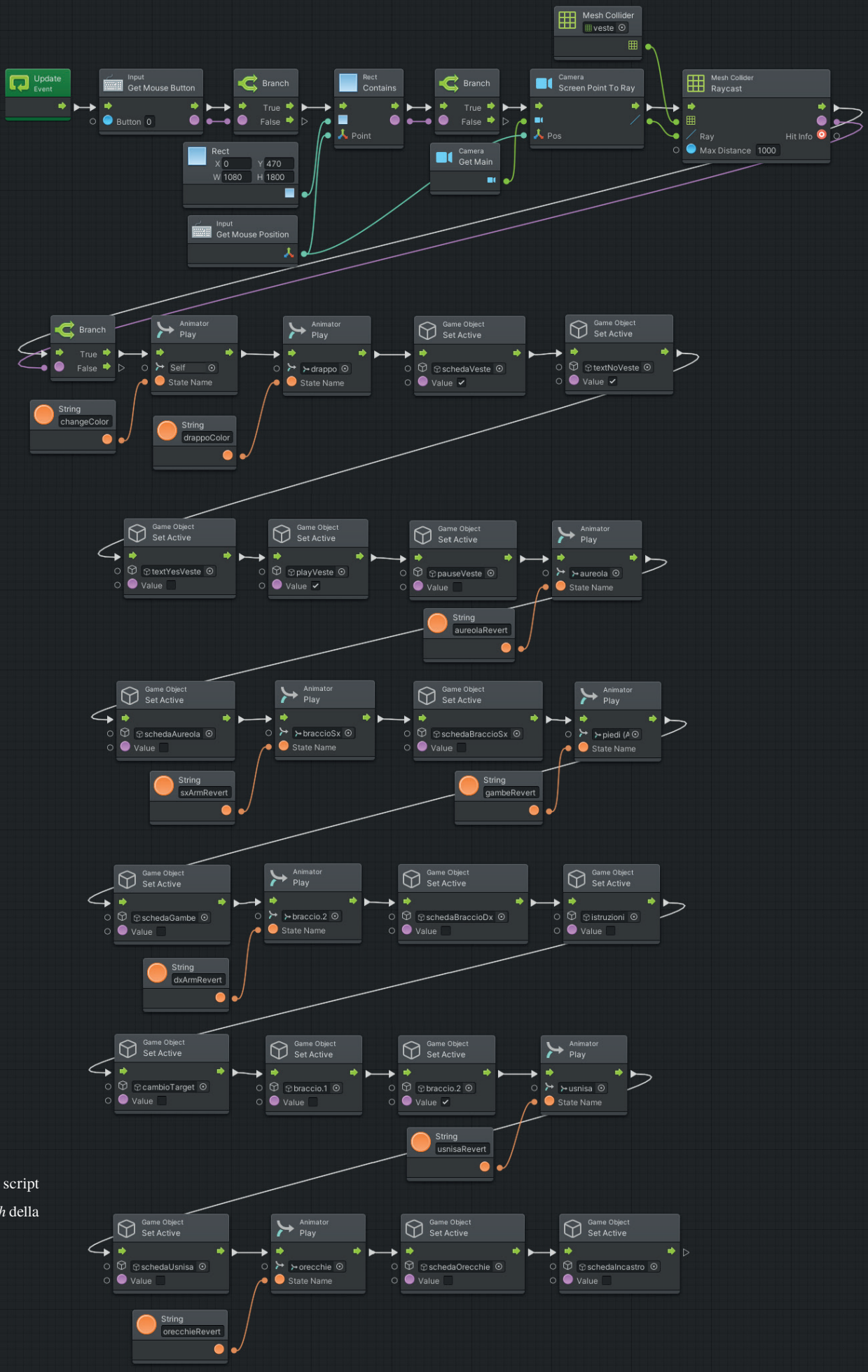


Figura 3.23 Lo script completo della *mesh* della veste.

Per rendere il progetto più interessante le informazioni contenute nelle schede di ogni step vengono anche lette da una voce sintetizzata. In rete sono disponibili più strumenti che consentono di scrivere un testo e selezionare la tipologia di timbro che si preferisce.
L'audio si esporta in formato *.mp3* e si può aprire su un *software* per la manipolazione della traccia audio, come ad esempio Adobe® Audition, per correggere alcune imprecisioni che può commettere un sintetizzatore automatico.
In seguito, agli *Empty* contenenti i testi informativi viene aggiunto per ognuno il *component AudioListener*, al cui interno è possibile specificare quale traccia audio si deve riprodurre (*Figura 3.24*). È importante che la voce *PlayOnAwake* sia spuntata, così che ogni volta che l'*object* a cui è agganciato si attiva, allo stesso tempo parta la riproduzione della traccia.

Ai vari pulsanti *Play* e *Pause* presenti all'interno del *Canvas* si aggiunge quindi, sotto la finestra *OnClick*, l'opzione *AudioSourcePlay* e

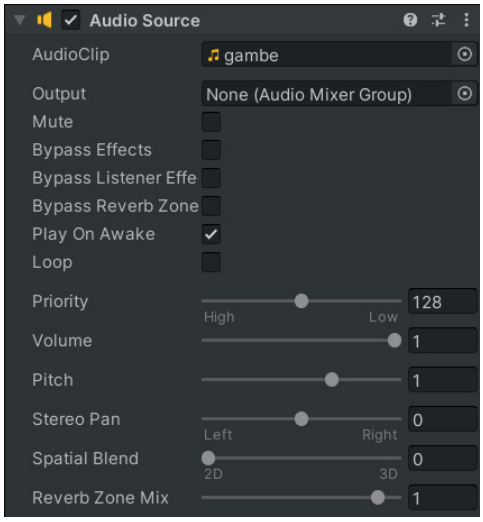


Figura 3.24 Il pannello con le impostanzioi dell'*AudioListener.*

Infine è stato aggiunto un ulteriore *script* per fare in modo che quando si conclude la riproduzione della traccia audio, compaia il tasto *Play* al posto di *Pause* e sia possibile quindi farla ripartire da capo (*Figura 3.25*).
Bolt dispone infatti dell'unità *IsPlaying* che, associata a un *AudioSource*, è in grado di capire se la riproduzione è attiva o no e di conseguenza restituire il valore *booleano* corrispondente.

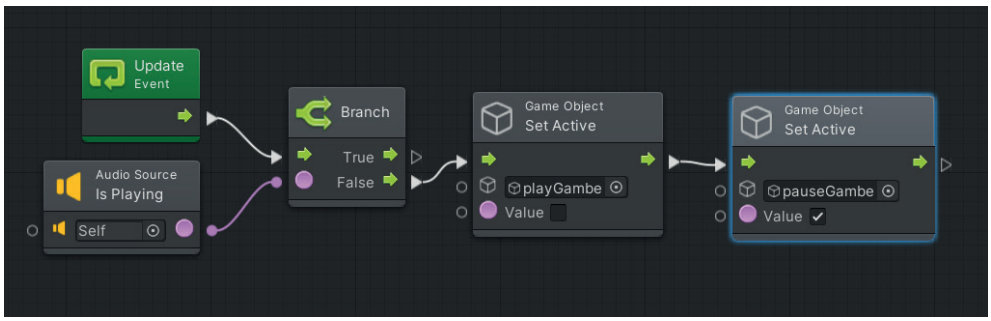


Figura 3.25 Lo *script* che regola i pulsanti alla fine della riproduzione audio.

Durante la fase di ricerca per la ricostruzione delle parti mancanti è emerso che il braccio destro poteva avere più forme. Sempre mostrando il gesto dell'*abhayamudra*, la conformazione della statua lascia ipotizzare che ci potesse essere un incastro, per cui l'autore dell'opera avrebbe scolpito l'arto separatamente e poi una volta terminato questo sarebbe stato agganciato al resto della scultura.
Si è scelto quindi di mostrare durante l'esperienza entrambe le possibilità.

a

b

Figura 3.26 Due momenti durante il cambio delle configurazioni del braccio destro.

All'avvio della ricostruzione compare solamente il braccio unito al resto del corpo. Nello *script* corrispondente, a differenza degli altri elementi, viene aggiunto che allo scadere della traccia audio compare un nuovo *GameObject*, raffigurante un pulsante per cambiare tra le due diverse configurazioni del braccio.

Il pulsante, che non appartiene al *Canvas* bidimensionale ma è a tutti gli effetti un oggetto 3D, è dotato anch'esso di un *Collider* che lo rende un oggetto cliccabile (*Figura 3.26 a*).
Al pulsante è collegato il solito *visual script*, al cui tocco sul *touchscreen* si disattiva la prima ricostruzione del braccio e compare la seconda ipotesi.
Per fare comprendere meglio all'utente come poteva avvenire il processo di incastro, il secondo braccio possiede una clip nell'*Animator* che si avvia all'attivazione dell'oggetto. L'animazione, realizzata nella finestra *Animation* inserendo *keyframe* di posizione e di rotazione, mostra il braccio completamente staccato dalla statua mentre si avvicina per unirsi (*Figura 3.26 b*).

Al progetto si aggiungono ancora degli *object* che servono per illuminare la scena. Si trovano nel menu a tendina *GameObject* sotto la voce *Light* e ne esistono di diverse tipologie.
Per cercare di simulare la stessa il-

luminazione con cui la statua viene illuminata all'interno del museo, si sono usate una *Directional Light*, che illumina tutta la scena allo stesso modo a secondo della sua inclinazione, e due *Point Light*, che invece sono dei punti luce sferici che influenzano l'oggetto a seconda della distanza e della potenza, utilizzati per togliere alcune zone d'ombra troppo scure.

Terminata l'aggiunta di alcuni dettagli, come alcuni pannelli con didascalie e istruzioni d'uso, il progetto di Unity® può finalmente essere esportato.
Aprendo la finestra *Build Setting* sono presenti alcune impostazioni per la realizzazione della *build*, che possono essere lasciate come appaiono. Cliccando sul comando *Build* e specificando il percorso di destinazione sul PC, si crea un file *.apk*, un pacchetto leggibile dai dispositivi Android che permette l'installazione dell'app contenuta al suo interno.

3.3 L'utilizzo dell'app

A - Istruzioni iniziali

B - Descrizione generale

C - Istruzioni di interazione

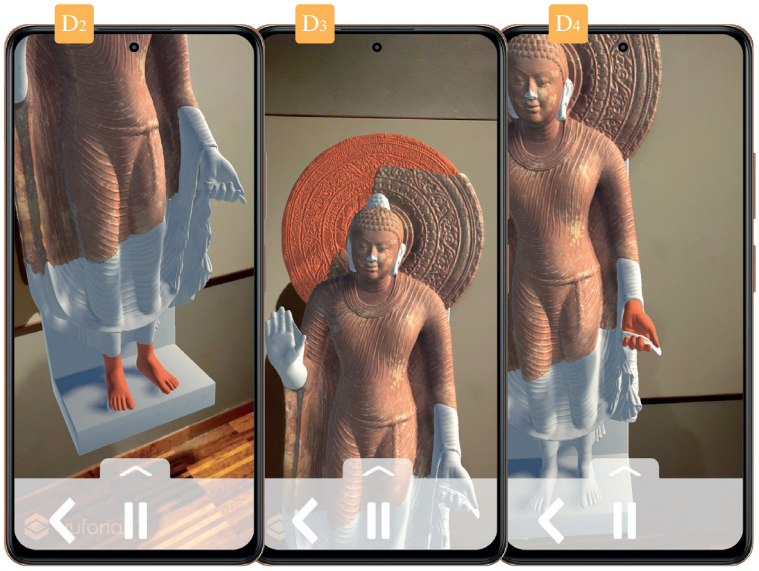
Per ricostruire la statua

Per tornare indietro

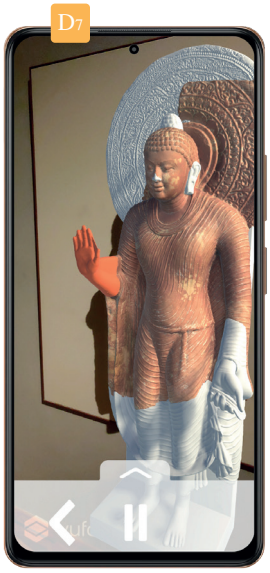
Per aprire il testo

Per mettere in pausa

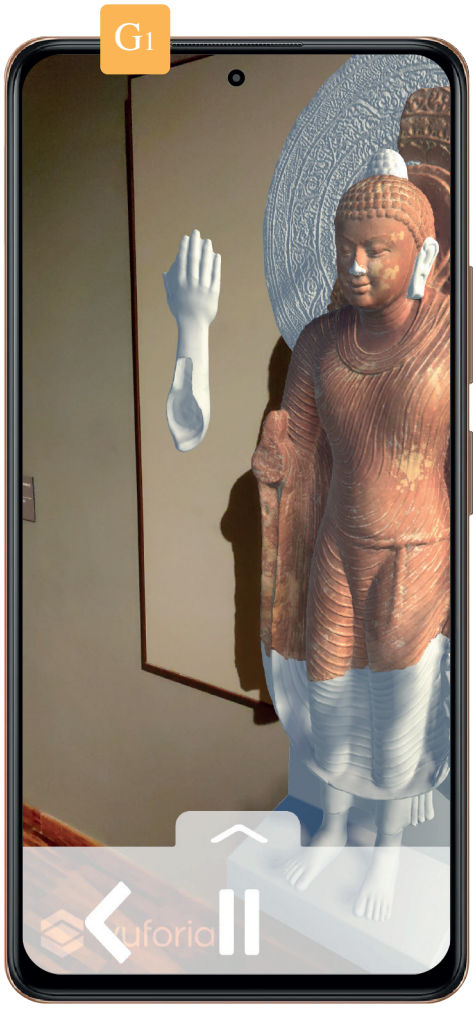
E - Lettura del testo

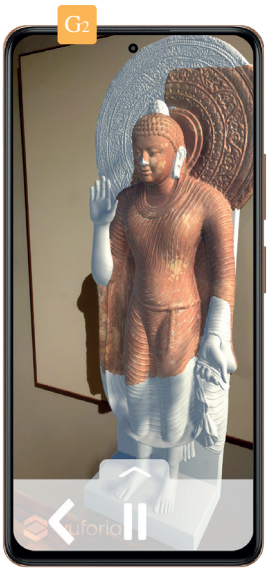


D - Descrizione della parte selezionata



F - Cambio di tipologia di braccio





G - Animazione dell'incastro

Figura 3.27 Il test del funzionamento all'interno del MAO.

Capitolo 4

Il progetto di realtà virtuale

4.1 Il visore e le impostazioni iniziali

Lo sviluppo dell'esperienza di realtà virtuale richiede innanzitutto di decidere quale tipologia di visore si intende utilizzare, in maniera da creare un progetto su Unity® ottimizzato appositamente.
Dei tanti disponibili in commercio ne esiste una tipologia economica e facilmente reperibile ideata da Google, il *Cardboard*.
La scelta è avvenuta dopo una serie di riflessioni rispetto alla fruizione dell'esperienza. La scelta di visori più complessi (e costosi), come ad esempio quelli prodotti da Oculus, avrebbe reso il progetto più completo e interessante sicuramente.
Va detto però che l'utente ne avrebbe quasi solamente fruito all'interno delle sale del museo, dal momento che si tratta di un prodotto costoso che difficilmente si trova nelle case di ognuno. Inoltre ci sarebbe la necessità di istituire una stanza atta ad ospitare i visitatori, per evitare possibili incidenti.
L'utilizzo di Google Cardboard rende invece l'esperienza indipendente dalla visita al museo, nel senso che può avvenire successivamente anche a distanza di tempo.
Google infatti, oltre a vendere direttamente il prodotto, mette a disposizione di tutti alcuni istruzioni per poterlo costruire in casa: è sufficiente quindi comprare i componenti, cartoncino, lenti di Plexiglass e delle cinghie di velcro, seguire le indicazioni e installare l'app sul proprio *smartphone*.

Figura 4.1 Il visore Cardboard selezionato per l'esperienza.

Alcune aziende, seguendo il progetto di Google, mettono in vendita alcuni visori già assemblati.

Per questo progetto ne è stato scelto uno che presenta anche un tasto in cartoncino rivestito in alluminio, che trasmette la carica elettrica del dito allo schermo del dispositivo, simulando così il tocco sul *touchscreen* (*Figura 4.1*).

Google mette inoltre a disposizione alcuni pacchetti creati appositamente per rendere più semplice la creazione di contenuti destinati all'utilizzo su Cardboard.
Ne esistono due pensati come plugin da installare su Unity®. È sta-

to scelto di utilizzare *GoogleVr for Unity*, leggermente più vecchio ma preferibile in quanto molte guide e tutorial *online* non sono aggiornate per l'ultimo disponibile.
La versione Unity 2020.3 LTS utilizzata in precedenza non supporta questo *plugin*, perciò è necessario utilizzare la versione precedente, la 2019.4.
Dopo aver creato quindi un nuovo progetto con la versione corretta, si procede quindi all'installazione del *package* attraverso il *Package Manager*. Con questa operazione diventano fruibili degli oggetti all'interno degli *asset* che semplificheranno il lavoro.

Il passo successivo prevede la preparazione dell'ambiente di lavoro predisponendolo alla realtà virtuale. Nel manuale di utilizzo di Cardboard sono presenti alcune indicazioni riguardanti delle specifiche impostazioni da impostare all'interno della finestra *Player Setting*.
Prima di tutto è necessario all'interno di *XR Setting* attivare la spunta a fianco della voce *Virtual Reality Supported* e aggiungere il *Virtual Reality SDK* che si intende utilizzare, in questo caso *Cardboard* (*Figura 4.2*).
Si decide l'orientamento dello schermo all'avvio dell'app, selezionando *Landscape Left* per fare in modo che lo *smartphone* si orienti sempre orizzontalmente verso sinistra.
Infine all'interno di *Other Setting* si trovano alcune impostazioni legate al reparto grafico.
La realtà virtuale genera due scene che poste all'interno del visore vengono elaborate dall'occhio umano come una scena unica.
Il lavoro del motore grafico del dispositivo deve però renderizzare in tempo reale due scene diverse: gli elementi presenti sono gli stessi, ma il punto di vista cambia e perciò svolge il doppio del lavoro.
Per ottimizzare la scena cercan-

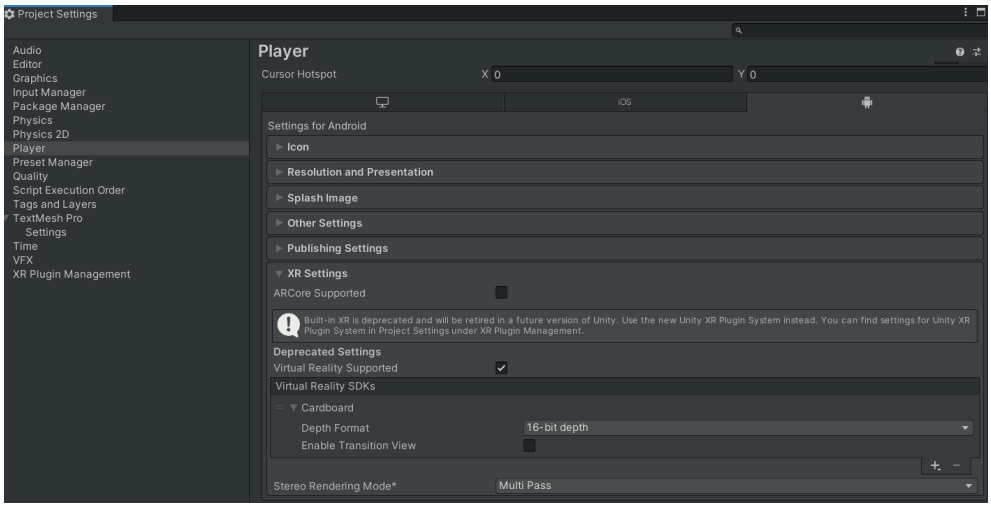


Figura 4.2 L'aggiunta del SDK di Cardboard al progetto.

do di ridurre lo sforzo per la GPU quindi si elimina Vulkan dall'elenco degli *Auto Graphics API*. Inoltre si imposta il *Minimum API Level* su Android 9.0. I dispositivi precedenti non saranno in grado di utilizzare l'app, ma ciò fa in modo che la scena vengo ottimizzata per una ristretta cerchia di *smartphone*, riducendo la quantità di lavoro.

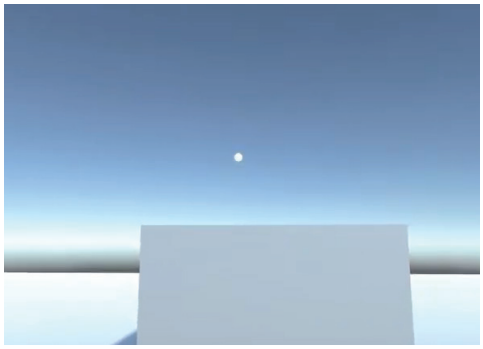


Figura 4.3 Il puntatore standar inserito dal *GvrReticlePointer*.

I primi elementi da aggiungere alla scena sono dei *prefab* di Google, caricati automaticamente all'interno degli *asset* nel momento dell'installazione del SDK di Cardboard.
Solitamente si aggiungono tre prefab nella *Hierarchy*: *GvrEditorEmulator, GvrEventSystem e GvrReticlePonter.*
Il primo è uno strumento molto utile in fase di sviluppo, in quanto consente di emulare ciò che accadrà sul dispositivo finale, vedendo la scena da un unico punto di vista, simulando il tocco sul *touchscreen* e la rotazione della testa dell'utente.
Il *GvrEventSystem* è ciò che regola le interazioni tra i *prefabs* e gli elementi nella scena, agisce in un certo senso come l'*EventSystem* ma in un progetto VR.
Il *GvrReticlePointer* è invece un oggetto che aggiunge davanti alla camera un elemento bidimensionale, un puntatore circolare (*Figura 4.3*).

Dato che il visore Cardboard non presenta altri *input* se non il giroscopio che fa ruotare la camera e il tocco sullo schermo, per interagire con la scena si può procedere in due direzioni.
Il puntatore in entrambi i metodi si evidenzia diventando più grande nel momento in cui passa sopra a un *object* dotato di *Collider*.
L'interazione più classica prevede che il tocco sullo schermo quando il *pointer* è sopra un oggetto faccia partire l'azione prestabilita.
Un'altra via percorribile prevede l'utilizzo del cosiddetto *Gaze Control*: il puntatore passa sopra l'oggetto con cui deve interagire e parte un timer; se il *Pointer* non viene distolto dall'oggetto per un certo tempo (che può essere di 1 o 2 secondi ad esempio), l'app riconosce l'attesa come se fosse un *click*.

Si è scelto di procedere con la prima opzione, in quanto il Cardboard a disposizione è dotato del pulsante per effettuare il tocco sullo schermo, semplificando anche i movimenti della testa che non si devono preoccupare di soffermarsi su punti specifici.

Volendo utilizzare il *visual scripting* per mezzo di Bolt, che male si sposa con gli *script* tradizionali già presenti nei *prefab* di Google, è stato necessario eliminare il *GvrReticlePointer* e di ricrearne uno con gli stessi funzionamenti, ma realizzato con i blocchi di nodi.
Prima però occorre disporre nell'ambiente tridimensionale tutti gli elementi utili.

Realizzazione dell'esperienza 4.2

Completamento degli asset nella scena 4.2.1

Si carica così l'adattamento del tempio Gumbat, i due *stupa* posizionati ai suoi lati, l'ambientazione circostante e la statua, sia la porzione conservata al museo, sia le parti ricostruite.
Alcune ricerche hanno fatto emergere che il piccolo podio scolpito in bassorilievo, su cui poggia direttamente la scultura, poggiava a sua volta su un altare più grande, per tenere la statua in posizione più elevata. Perciò con una rapida modellazione si è aggiunto anch'esso alla scena (*Figura 4.4*).

Figura 4.4 L'altare aggiunto all'interno della cella.

Per l'illuminazione, allo stesso modo di come fatto per il progetto di AR, si utilizza una *Directional Light* principale accompagnata da una *Spot Light*.
La *Directional* è orientata e dimmerata in maniera tale da simulare un sole di metà pomeriggio, così che la luce passi ancora attraverso il foro della cupola ma allo stesso tempo

illumini anche attraverso le fessure poste ai fianchi del tempio.
La *Spot Light*, che simula il comportamente a cono delle luci da palcoscenico, è stata posizionata dall'alto per evidenziare la scultura e rendere la scena più teatrale.

Siccome non si può controllare il movimento della camera con un *gamepad*, offrendo così la possibilità di muoversi liberamente nello spazio, si sono istituiti alcuni punti fissi in cui ci si può spostare.
L'esperienza comincia fuori dal tempio ai piedi della rampa di scale, si sposta sul pianerottolo di fronte alla porta d'ingresso, arriva davanti alla statua e si muove nel corridoio interno compiendo la *pradakshina*, la circumambulazione rituale in senso orario attorno all'oggetto sacro.
All'interno del corridoio ci si ferma inoltre in due punti in corrispondenza delle feritoie, attraverso cui è possibile osservare la scultura lateralmente.
Per avviare gli spostamenti sono inserite nella scena delle frecce che danno la possibilità di essere cliccate per procedere nella direzione indicata (*Figura 4.5*). Per ognuna di esse c'è poi anche la corrispettiva che fa tornare indietro al punto da cui si è partiti.

Per ogni freccia sono state realizzate delle clip animate per far capire meglio all'utente quando il puntatore si sofferma su di esse.

Figura 4.5 Le frecce che regolano lo spostamento dell'utente.

Le proprietà inserite all'interno della clip riguardano il colore del materiale, che passa dal giallo all'arancione scuro, e la dimensione, diventando più grandi. Allo stesso modo viene realizzata la clip inversa per fare ritornare la freccia alle condizioni iniziali (*Figura 4.6*).

Figura 4.6 Cambio di condizione della freccia.

4.2.2 *Organizzazione dello script*

La creazione dello *script* ha seguito un processo differente rispetto al progetto AR. In questo caso si è fatto un tentativo per avere un unico *script* molto grande che comanda tutti gli oggetti insieme.

L'*object* a cui è collegato lo *script* è il player, un *Empty* a cui è imparentata la *MainCamera*, a cui a sua volta è imparentato un altro *Empty* denominato *RayCaster* (*Figura 4.7*). In questo modo si ottengono due oggetti differenti raggruppati sotto il *Player*, ma che avranno sempre la stessa posizione e rotazione, in quanto comandando la *MainCamera* con i movimenti della testa si comanda anche il *RayCaster*.

A questo punti si impostano nella finestra *Blackboard* di Bolt due variabili.
Le variabili nel *visual scripting* possono essere di più tipologie a secondo della diffusione che si vuole all'interno del progetto. Dato che si crea un unico *script* è sufficiente crearle all'interno della sezione *Object*: in tal modo solo quel preciso *Object* (il *Player*) avrà a disposizione queste variabili.
Si creano dunque la variabile *GameObject rayCaster*, che richiama all'*Object RayCaster*, e la variabile *String hitObjectTag*, con valore *Null* (*Figura 4.8*).
In questo progetto è importante che ogni elemento all'interno della scena abbia un *tag* specifico e unico, così da permettere il funzionamente del *Get Collider*, e anche il *Component Mesh Collider*.
Il *Tag* compare nell'*Inspector* e vi è la possibilità di aggiungerne e di assegnarne agli *object* designati.
Ad esempio per la freccia che si occuperà del primo spostamento si è assegnato il *Tag Teleport1*, che per comodità ha la stessa dicitura del proprio *gameObject* (*Figura 4.9*).

La prima parte dello script quindi prevede l'utilizzo del blocco *Raycast*. Si vuole che il raggio parta dall'*Empty* imparentato alla *Main-*

Figura 4.7 La gerarchia del *Player*.

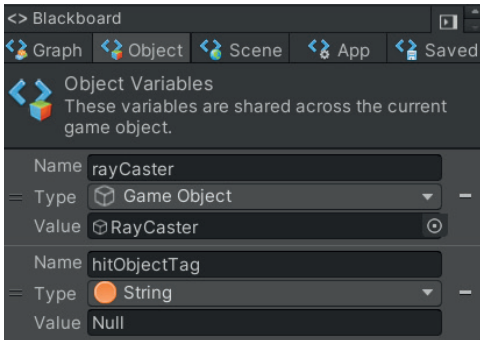


Figura 4.8 La finestra *Blackboard* in cui si creano le variabili per il *visual scripting*.

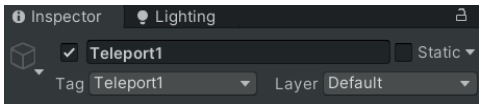


Figura 4.9 L'assegnazione del *Tag*.

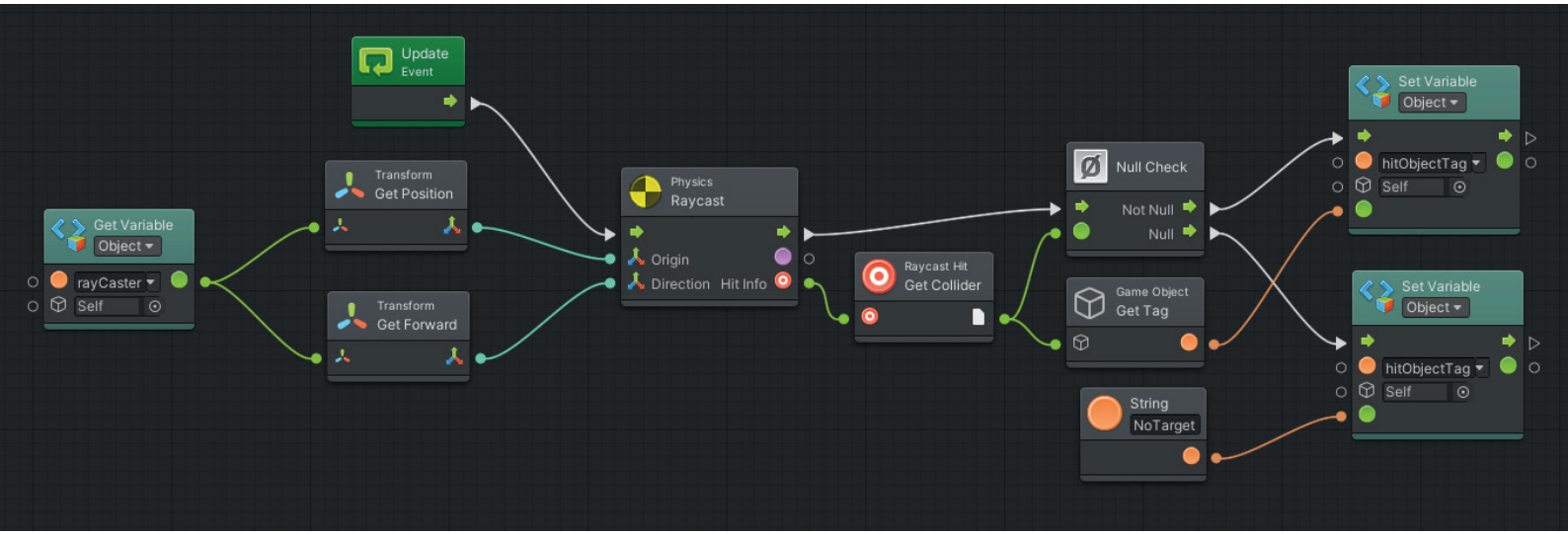


Figura 4.10 I primi blocchi dello script.

Camera, perciò, grazie alla variabile impostata precedentemente collegata alle unità di *GetPosition* e *GetForward*, il *raycaster* è in grado di far partire il raggio sempre dalla esatta posizione della *MainCamera* e nella stessa direzione in cui è direzionato.
Il *flow* continua arrivando al blocco *Null Check*. Il *GetCollider* si occupa di indicare al *NullCheck* se è stato colpito o no un oggetto dotato di *Mesh Collider*. In caso negativo il flusso continua sulla variabile impostata prima, restituendo il valore *NoTarget* (utile solamente in fase di sviluppo) e terminando le operazioni.
Se invece il *NullCheck* riceve un *input* positivo, si procede a indicare alla variabile il *tag* dell'oggetto colpito (*Figura 4.10*).

Prima di procedere con la programmazione, si crea il sostituto del *pointer* di Google.

Importata un'immagine creata su Adobe® Illustrator, si inserisce all'interno di un *Canvas*. Dato che i punti di vista saranno poi due, la scelta obbligata da Unity® è di scegliere il *RenderMode* del *Canvas* come *ScreenSpace*.
L'immagine quindi rimane attaccata alla *MainCamera*, ma è possibile regolare la distanza dalla stessa.
È fondamentale che il *Pointer* sia posizionato perfettamente al centro della camera, dato che la sua posizione e quella del *RayCaster* devono coincidere.
Si sono create due clip animate per distinguere il puntatore dalla situazione di riposo a quella in cui punta verso un oggetto cliccabile, passando dal bianco al rosso e aumentando di dimensioni (*Figura 4.11*).
Alcune animazioni sono aggiunte anche per il Player, facendo in modo che ognuno di esse faccia compiere il movimento da una posizione a quella successiva.

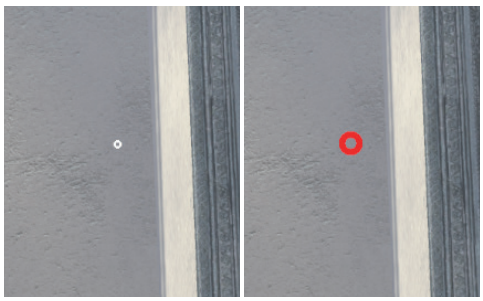


Figura 4.11 Le due configurazioni del *pointer*.

Predisposti questi elementi si può procedere con il completamento dello *script*.
Il blocco *CompareTag* permette di inserire una *String* con il *Tag* che si vuole confrontare e di agganciare il *GetCollider* prima creato.
In questo modo ci sarà un paragone tra il *Tag* dell'oggetto colpito e il *Tag* espresso nella *String*: il blocco restituisce un valore *booleano*, *True* se i due *Tag* coincidono e *False* se sono differenti (*Figura 4.12*).
L'unità *Branch* a questo punto divide i due casi. Nel caso di valore *booleano* positivo il flusso continua facendo avviare due animazioni, quella della freccia e quella del puntatore, in cui entrambi gli oggetti diventano rossi e aumentano di dimensioni.

Viene inserito un blocco *WaitForFlow* che prevede vi siano due *input* per far proseguire il flusso (*Figura 4.13*).
Siccome si è optato inizialmente per il controllo attraverso il tocco dello schermo (consentito dal tasto di alluminio del Cardboard), il secondo *input* richiesto dal blocco è il *click* del tasto sinistro del *mouse*, che si trasforma nel tocco sul dispositivo finale. Perciò se allo stesso tempo si verifica l'*input* del *mouse* e la corrispondenza tra i due *Tag*, si avviano una serie di blocchi, prevalentemente di tipo *SetActive*, per attivare e disattivare oggetti nella scena.

Nel caso delle frecce, dopo l'*input* del *mouse* sono presenti anche le rispettive animazioni del *Player* per passare da una zona all'altra. Il flusso in questo caso è ritardato successivamente da un'unità chiamata *Cooldown*, una specie di timer che ritarda della durata della clip animata l'avviamento dei blocchi che lo seguono.

Nel caso in cui il *Branch* prima citato riceva un *booleano* negativo si

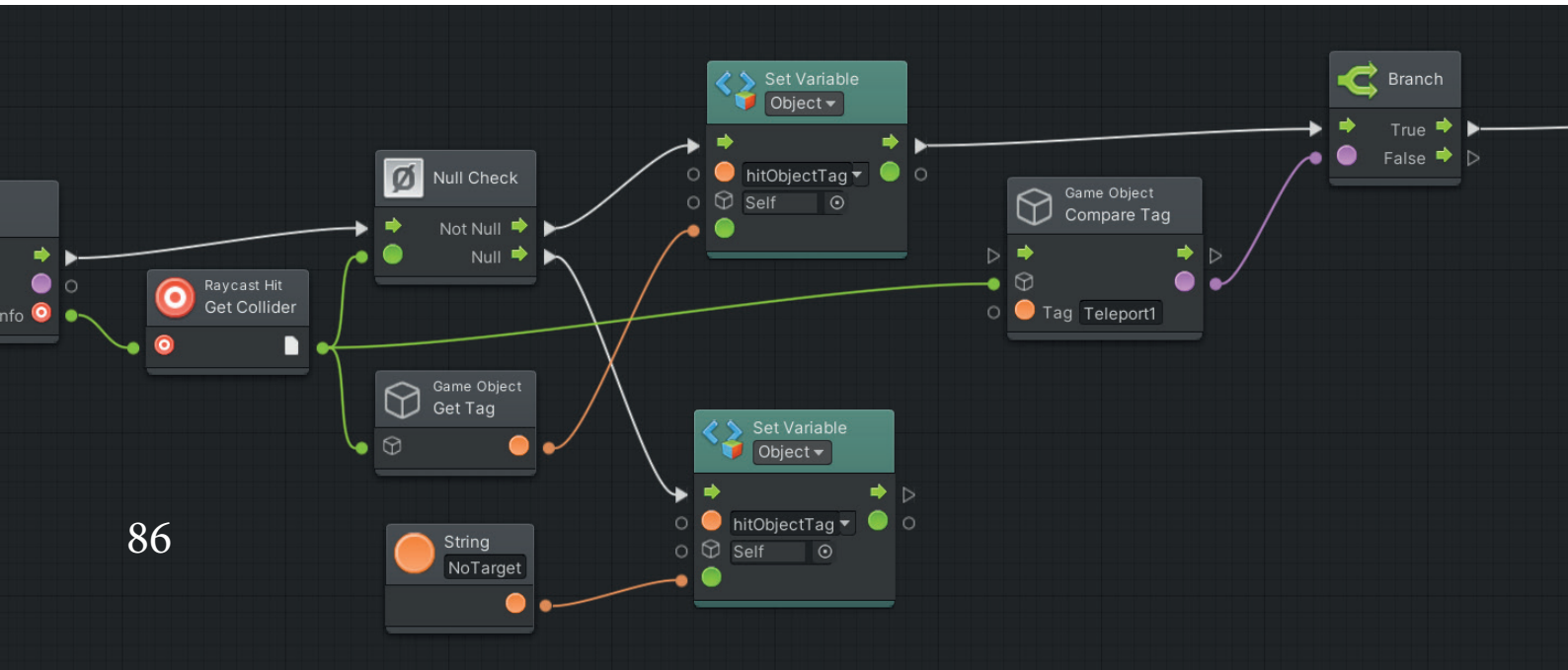


Figura 4.12 L'utilizzo del blocco *CompareTag*.

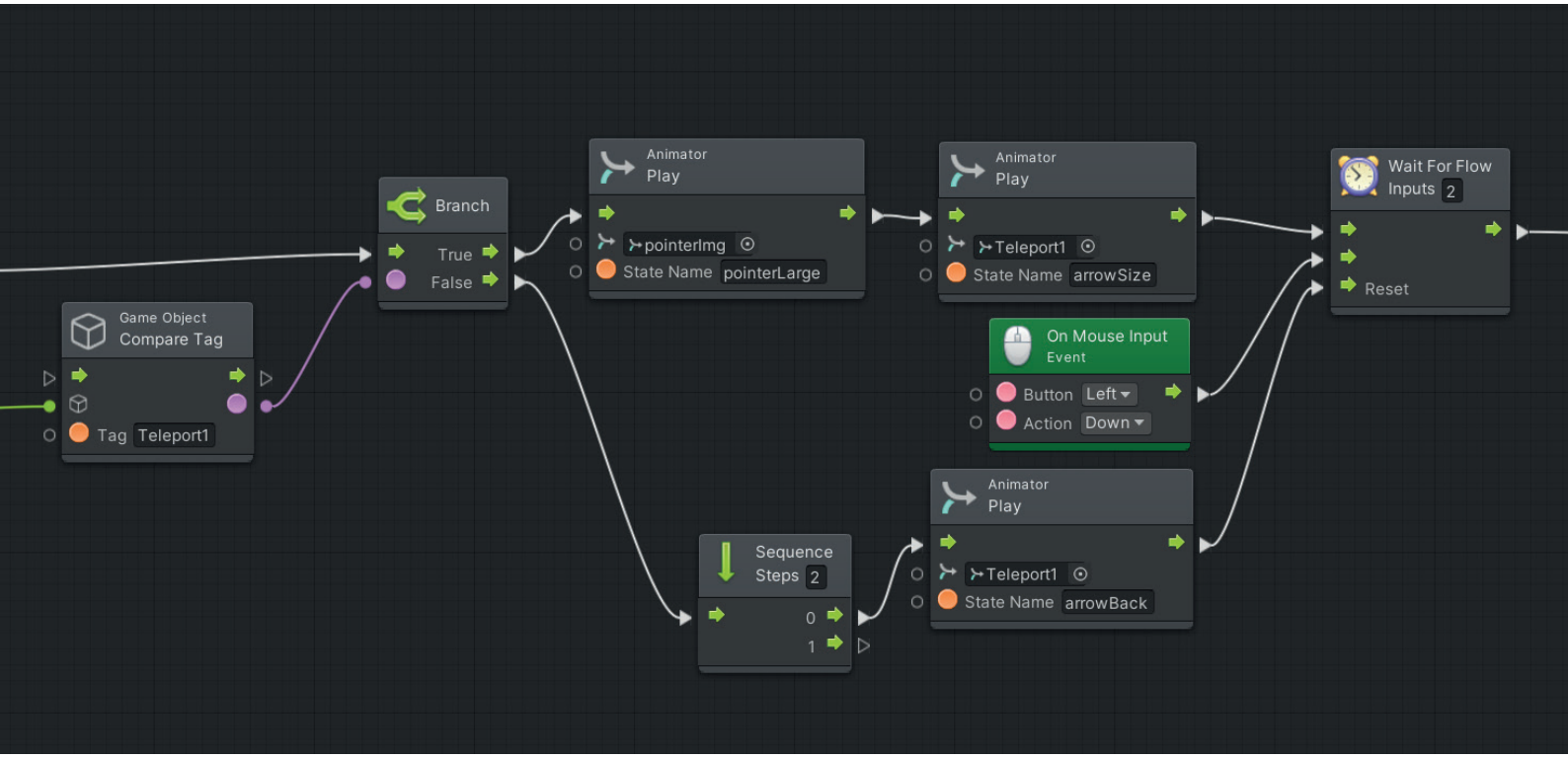


Figura 4.13 L'utilizzo del blocco *WaitForFlow* e il *MouseInput*.

procede con un *SequenceSteps*, che contemporaneamente presenta più *output*.
Uno dei due fa avviare l'animazione che riporta la freccia allo stato originario e successivamente effettua il reset del *WaitForFlow*, così che in futuro sono richiesti nuovamente da capo i due *input* necessari per attivarlo.
Il secondo *output* del *SequenceSteps* porta il flusso verso la freccia successiva, che a sua volta ripete le stesse operazioni.

Si genera dunque una serie di sette linee di blocchi, una per ogni freccia, che funzionano a "cascata": ogni volta che il *Branch* di una linea ottiene come valore *False*, il *flow* passa al *Branch* successivo, fino a quando uno di questi riceve un input *True* (che corrisponde al posizionamento del *Pointer* sull'oggetto corrispondente) (*Figura 4.14*).

Nella terza posizione che si può andare a occupare durante l'esperienza ci si trova di fronte alla statua. Analogamente a quanto fatto per la statua del Buddha Gupta, si vuole

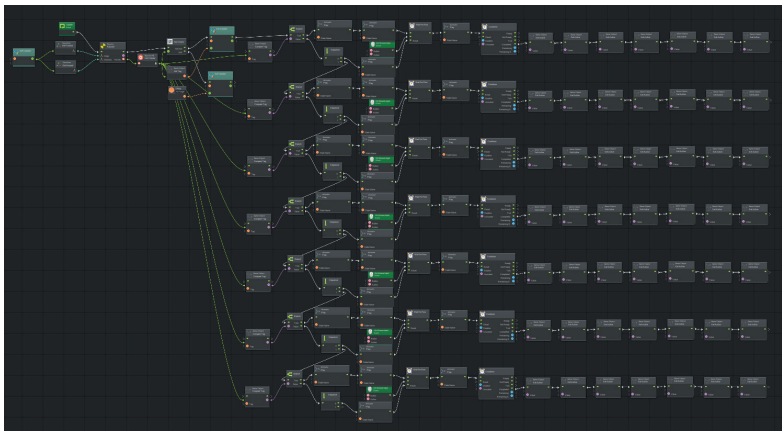


Figura 4.14 Vista complessiva della gestione dei movimenti del *Player*.

Figura 4.15 Il *Pointer* interagisce con le *mesh* della statua.

dare anche in questo caso la possibilità all'utente di interagire con essa, andando a cliccare le varie parti dell'opera per approfondire i dettagli.
Perciò ogni *mesh* presenta un materiale di partenza grigio chiaro e due animazioni, che lo fanno diventare rosso e lo portano indietro al colore di partenza.
Come per gli altri oggetti il cambio di colore serve per evidenziare la geometria sul quale il puntatore si è posato e capire quale sia al momento quello attivo (*Figura 4.15*).
Il metodo con il quale programmare il *visual script* è dunque lo stesso effettuato per gli spostamenti, con la differenza che successivamente al blocco *WaitForFlow* si attivano o disattivano un serie di *Empty* a cui sono legati dei Component *AudioSource*, con l'opzione *PlayOnAwake*. Cliccando quindi su una porzione di statua si avvia una traccia audio in cui la voce sintetizzata spiega il dettaglio corrispondente.

La cascata del gruppo relativo agli spostamenti del *Player* si connette quindi al gruppo dei blocchi della statua, connettendo l'ultimo *Branch* negativo del primo raggruppamento al primo *Branch* del secondo.

Per evitare che un singolo tocco del *touchscreen* attivi più elementi sovrapposti si è scelto di attivare il *Mesh Collider* delle parti della statua solo quando il *Player* si trova all'interno del tempio e di disattivarli una volta uscito: in questo modo quando l'utente si trova al di fuori dell'edificio non rischia di interagire inavvertitamente con la scultura.
Come descrizione aggiuntiva di alcuni particolari sono presenti degli oggetti di testo tridimensionali che indicano la parte interessata (*Figura 4.16*).
Questi si attivano al tocco sulla relativa *mesh* e appaiono con un'animazione di trasparenza, effettuata

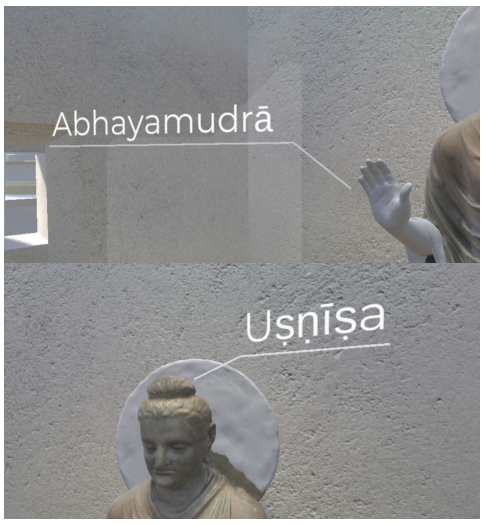


Figura 4.16 Le scritte 3D di supporto alla spiegazione audio.

attraverso il canale *alpha* della *Base Map*.
Date alcune caratteristiche del tempio si è scelto di rendere interattiva anche la rampa di scale e la cornice dell'apertura d'ingresso. Il funzionamento della meccanica è sempre lo stesso degli elementi precedenti.

Gli ultimi elementi inseriti sono alcuni pannelli iniziali.
Il primo compare subito all'avvio dell'esperienza e fornisce una serie di informazioni su come funzioni l'app (*Figura 4.17*). Dopo questo passaggio compaiono altri due pulsanti che permettono di avere informazioni più generali riguardanti l'ambientazione, con una descrizione sul *Gandhara*, una sul *Gumbat* e una sugli *stupa* posti ai suoi fianchi (*Figura 4.18*).
Terminate le tracce audio, è possibile premere sul pulsante *"Prosegui"* per fare comparire la prima freccia e iniziare con gli spostamenti.
Pur essendo immagini bidimensionali è possibile dotare questi elementi di un *Collider*, in modo da utilizzare lo stesso *workflow* all'interno di Bolt per programmarne l'interazione.

Per rendere l'esperienza più coinvolgente infine è stata aggiunta una traccia audio di sottofondo che riproduce una tipica melodia antica proveniente dalle zone in questione.
Il *Player* è quindi dotato del componente *AudioSource* con la colonna sonora, che si avvia fin dall'inizio, con l'opzione *PlayOnAwake*. Per regolare meglio il volume e fare sì che la voce sintetizzata non venga coperta dalla musica, parallelamente allo *script* che regola le interazioni si crea una sezione di blocchi destinati esclusivamente alle impostazioni audio, grazie al già utilizzato *SequenceSteps*, che permette la gestione di due flussi in contemporanea.
Anche qui vi è una "cascata" di blocchi *Branch*, in cui ognuno di essi riceve un valore *booleano* da parte di un'unità *IsPlaying* (*Figura 4.19*).
Se la traccia in questione è attiva (ovvero la voce sintetizzata sta parlando) il valore *True* si trasmette al *Branch* che continua il flusso arrivando al blocco *SetVolume* collegato alla melodia, abbassando il livello del volume da 1 a 0.15.

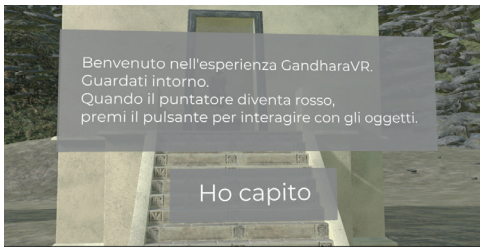


Figura 4.17 Il pannello di introduzione con le istruzioni iniziali.

Figura 4.18 I due pulsanti per le descrizioni generali introduttive.

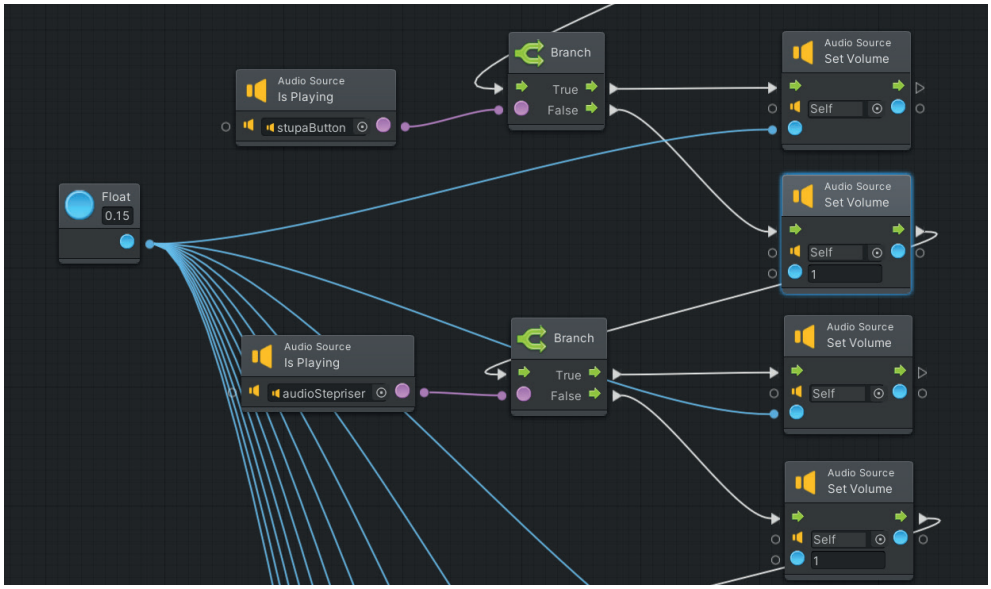


Figura 4.19 Lo *script* per la gestione dei volumi audio.

In caso di *booleano* negativo invece si attiva un altro *SetVolume* che lascia l'impostazione a 1. Quest'ultimo a sua volta procede al *Branch* successivo, fino a passare tutte le tracce audio.
Nel caso quindi in cui nessuna sia in riproduzione, il *flow* arriva al fondo della cascata e il volume rimane 1. In caso contrario si abbassa come già detto.

L'ultimo accorgimento da prendere per completare il progetto riguarda la messa a fuoco della scena.
L'essere umano può osservare il mondo che lo circonda attraverso la visione stereoscopica: i due occhi infatti vedono da due punti di vista leggermente sfalsati che generano nel cervello due immagini differenti. Quando ci si concentra su un punto specifico e lo si mette a fuoco, le due immagini si sovrappongono in quel punto particolare, mettendo a fuoco tutto ciò che si trova alla stessa distanza, mentre il resto dell'immagine appare "sdoppiata".
La stessa situazione avviene all'interno di un esperienza di realtà virtuale. Le due immagini generate infatti non sono perfettamente identiche, ma ci sono alcune piccole differenze che ci fanno percepire la profondità della scena.
Dal momento che il puntatore si trova immediatamente davanti alla lente della *MainCamera*, esso risulta essere sempre sdoppiato: le due immagini del *Pointer* si allontanano maggiormente se mettiamo a fuoco qualcosa di molto ravvicinato e viceversa.
L'unica situazione in cui potrebbero sovrapporsi richiederebbe di tenere gli occhi esattamente perpendicolari all'asse che li unisce,

rendendo però impossibile mettere a fuoco il resto della scena.
Questa condizione rende difficile all'utente l'utilizzo corretto del *Pointer* in quanto, vedendolo sdoppiato, non riesce con precisione a capire cosa sta puntando.

È perciò necessario aggiungere un terzo script parallelo a quello principale per correggere il problema.
Grazie all'unità *GetPoint*, si ottiene in tempo reale l'esatta posizione espressa in coordinate del punto che in un certo momento è colpito dal *RayCaster*.
Con il blocco *Distance* si può risalire alla distanza tra questo punto ed il punto di origine del *RayCaster* (*Figura 4.20*).
È sufficiente quindi impostare questa distanza come valore di distanza per il *SetPlaneDistance* del *Canvas*.
In questo modo il *Canvas* si troverà sempre alla stessa distanza dell'oggetto su cui punta il *Pointer* (a cui è sottratto uno 0.15 per tenersi leggermente più vicini).
Dato che in quel momento l'utente sta mettendo a fuoco proprio la porzione della scena che sta fissando, le due immagini del puntatore si trovano così sovrapposte con precisione e il cervello le elabora quindi come un'immagine unica collocata nello spazio.

Alla termine della programmazione aver creato uno script unico si è rivelato più difficoltoso nella gestione della moltitudine dei blocchi (ottimizzabile attraverso la creazione di sottogruppi, *Figura 4.21*), ma allo stesso tempo molto vantaggioso per tenere sotto controllo gli errori generati dal flusso in tempo reale e in un'unica finestra.
Infine si può dunque preparare la *build* del progetto e installare il pacchetto *.apk* sul proprio *smartphone*, poi dopo avere avviato l'app si inserisce il dispositivo all'interno del visore Cardboard e l'esperienza è pronta per essere testata.

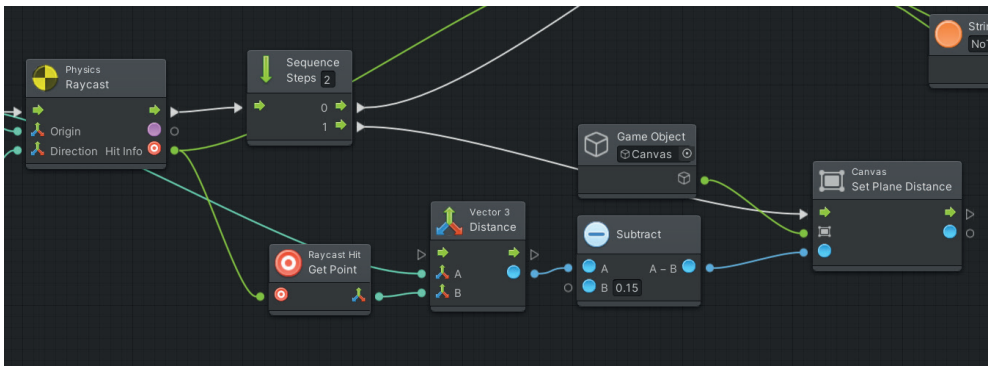


Figura 4.20 I blocchi per la correzione della distanza del *Canvas*.

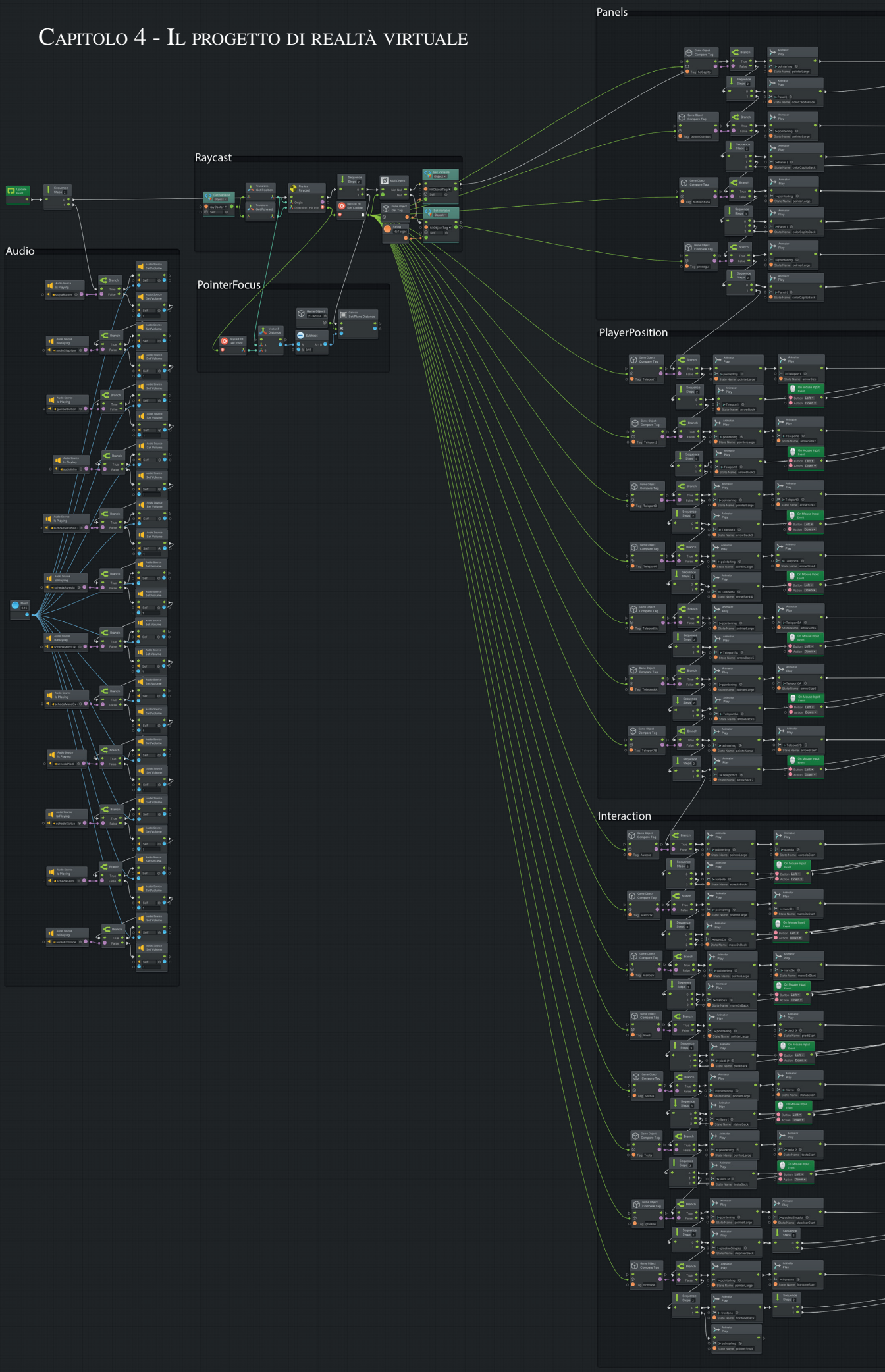
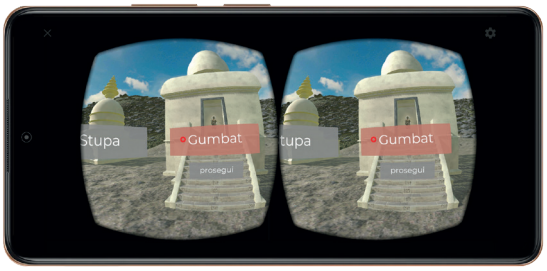
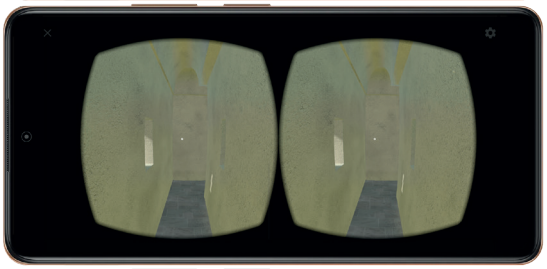

Panels
Raycast
Audio
PointerFocus
PlayerPosition
Interaction

Figura 4.21 Il *visual script* completo dell'esperienza VR.

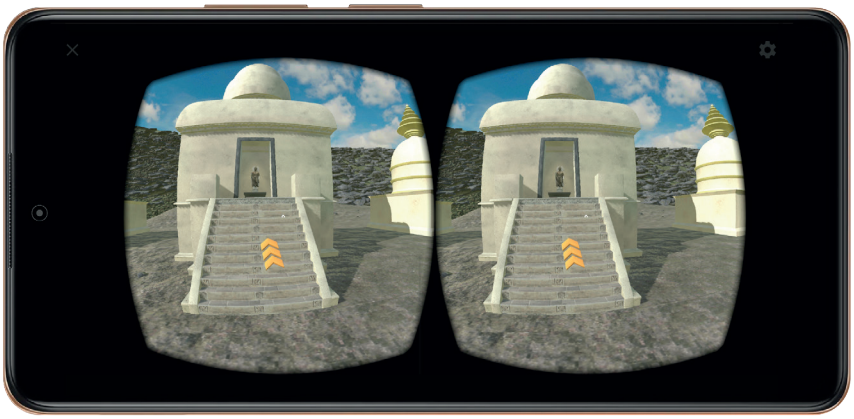
4.3 L'utilizzo dell'app



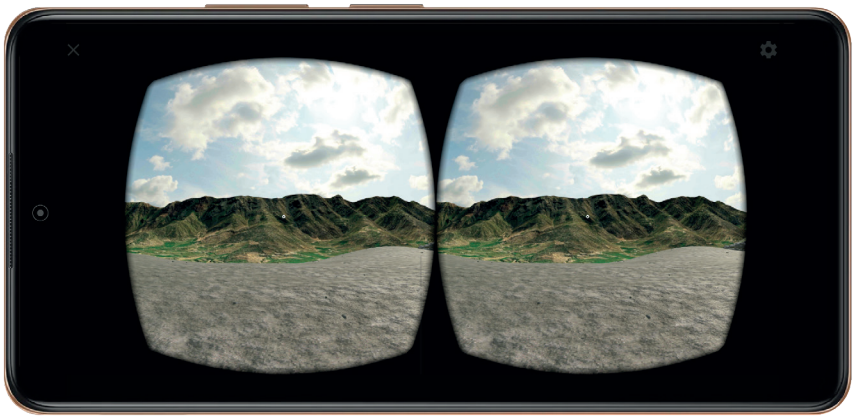
I pannelli iniziali

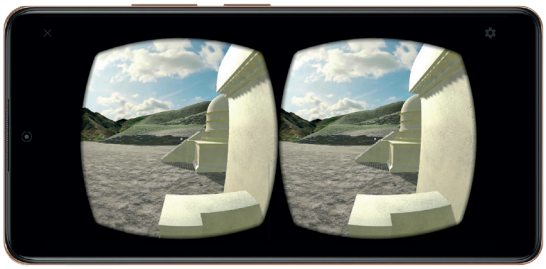


Gli spostamenti nell'ambiente



Le interazioni con gli oggetti





Il paesaggio circostante

Bibliografia e sitografia

Bibliografia

Arduini G., *La realtà aumentata e nuove prospettive educative*, in «Education sciences & society», 2012.

Barrile V., Bilotta G. et al., *Computer vision / structure for Motion per la diffusione dei beni culturali*, XIX Conferenza Nazionale ASITA, Lecco, Settembre 2015.

Barrile V., Fotia A., Candela G., Bernardo E., *Geomatics Techniques for Cultural Heritage Dissemination in Augmented Reality: Bronzi di Riace Case Study*, in «Heritage», II,, 2019.

Basu C., Behrendt K et al., *South Asian Religions and Visual Forms in their Archaeological Context*, Vincent Lefèvre, Breple, vol. 2, 2012.

Betto F. et al., *Mont'e Scan: Effective Shape and Color Digitization of Cluttered 3D Artworks*, in «ACM Journal on Computing and Cultural Heritage», vol. 8, n.1, 2015.

Bianconi F. et al., *Comparison between two non-contact techniques for art digitalization*, XXIV A.I.VE. LA. Annual Meeting, Brescia, 27-28 Ottobre 2016.

Bianconi F., Filippucci M., *La fotomodellazione per il rilievo archeologico*, in «Archeologia e calcolatori», All'insegna del giglio, Firenze, n.30, 2019.

Bianconi F., Filippucci M., Catalucci S., T*ratto e punti. Analisi critica nell'evoluzione del rilievo archeologico in trent'anni di esperienze in Umbria*, in «Disegnare con», volume X, n.19, 2017.

Böhler W., Marbs A., *3D scanning and photogrammetry*
for heritage recording: A comparison, Proceedings of the 12th International Conference on Geoinformatics: Geospatial Information Research: Bridging the Pacific and Atlantic, University of Gävle, Sweden, 7-9 June 2004.

Bozzelli G., Raia A.,, Ricciardi S. et al., *An integrated VR / AR framework for user-centric interactive experience of cultural heritage: The ArkaeVision project*, in «Digital Applications in Archaeology and Cultural Heritage», XV, 2019.

Coomaraswamy A. K., *Elements of Buddhist Iconography*, Munshiram Manoharlal, 2004.

Faccenna D., Filigenzi A., *Repertorio terminologico per la schedatura delle sculture dell'arte gandharica*, ISIAO, Reports and Memoirs, Roma, 2007.

Girelli V. A., *Tecniche digitali per il rilievo, la modellazione tridimensionale e la rappresentazione nel campo dei beni culturali*, Tesi di Dottorato, Alma Mater Studiorum - Università di Bologna, 2007, relatore Bitelli G

Horňák M., Novaković P., Zachar J., *3D Digital recording of architectural and artistic heritage*, CONPRA Series, I, 2017.

Iori E., Olivieri L. M., *Monumental Entrance to Gandharan Buddhist Architecture Stairs and Gates from Swat*, in «Annali di Ca' Foscari. Serie orientale», vol. 57, Giugno 2021.

Jasink A. M., Dionisio G., (a cura di), *MUSINT 2. Nuove esperienze di ricerca e didattica nella museologia interattiva*, Firenze University Press, Firenze, 2016.

Khandalavala K. J., *The Golden age - Gupta art - empire, province, and influence*, Marg Publications, 1991.

Luigini A., Panciroli C., *Ambienti digitali per l'educazione all'arte e al patrimonio*, FrancoAngeli s.r.l., Milano, 2018.

Machidon M. Octavian et al., *Virtual humans in cultural heritage ICT applications: A review*, in «Journal of Cultural Heritage», 33, 2018.

Meschini A., *Tecnologie digitali e comunicazione dei beni culturali. Stato dell'arte e prospettive di sviluppo*, in «Disegnare con», IV, n.8, 2011.

Mibact, *Piano Triennale per la Digitalizzazione e l'Innovazione dei Musei*, 2019.

Myer Prudence R., *Bodhisattvas and Buddhas - Early Buddhist Images from Mathurā*, Artibus Asiae Publishers, Vol. 47, n. 2, 1986.

Obradović M., Vasiljević et al., *Virtual Reality Models Based on Photogrammetric Surveys - A Case Study of the Iconostasis of the Serbian Orthodox Cathedral Church of Saint Nicholas in Sremski Karlovci (Serbia),* in «Applied sciences», X, 2020.

Oddicini C., Olivieri L. M., *Valle dello Swat, Memorie di un viaggiatore,* in «Archeologia Viva», XXXIX, n. 202, 2020.

Olivieri L. M., *A short note on contexts and chronology of the materials from Saidu Sharif, Amluk-dara, Gumbat and Barikot (Swat)*, in «Restauro Archeologico, Rivista del Dipartimento di Architettura dell'Università degli Studi di Firenze», 2019.

Quintanilla S. R., *History of Early Stone Sculpture at Mathura, CA. 150 BCE-100 CE (Studies in Asian Art and Archaeology)*, Brill Academic Publishers, 2007.

Rhi J. H., *From Bodhisattva to Buddha: The Beginning of Iconic Representation in Buddhist Art,* Artibus Asiae Publishers, vol. 54, n. 3/4, 1994.

Spallone R., Bertola G., Ronco F., *SfM and Digital Modelling for Enhancing Architectural Archi-*

ves Heritage, Proceedings of 2019 IMEKO TC-4 International Conference on Metrology for Archaeology and Cultural Heritage, Firenze, Dicembre 2019.

Verdiani G., *Retroprogettazione. Metodologie ed esperienze di ricostruzione 3D digitale per il Patrimonio Costruito,* Dida press, Firenze, 2017.

Wallace C., *Photogrammetry in Mediterranean Archaeology*, Master's Thesis, University of Waterloo, 2016, supervisor Deadman P.

Sitografia

Agisoft Metashape Helpdesk Portal, *https://agisoft.freshdesk.com/support/home*, ultima consultazione 15 marzo 2021.

AmbientCG, *https://ambientcg.com,* ultima consultazione 24 agosto 2021.

Blender Community, *https://blender.community*, ultima consultazione 24 agosto 2021.

Britannica, Gandhara Art, *https://www.britannica.com/art/Gandhara-art*, ultima consultazione 26 luglio 2021.

Daz 3D, *https://www.daz3d.com,* ultima consultazione 30 marzo 2021.

Developers Google Ar, *https://developers.google.com/ar/develop,* ultima consultazione 25 agosto 2021.

Developers Google Vr, h*ttps://developers.google.com/cardboard/develop/unity/quickstart?hl=en*, ultima consultazione 28 agosto 2021.

Getting started with Bolt in Unity, *https://www.youtube.com*, ultima consultazione 17 giugno 2021.

Il Giornale dell'Architettura, *I musei e la sfida della digitalizzazione, https://ilgiornaledellarchitettura.com/2021/01/25/i-musei-e-la-sfida-della-digitalizzazione,* ultima consultazione 26 agosto 2021.

In Italia Magazine, *Entrare nella storia con la realtà aumentata. Il progetto pilota dei musei di Varese, https://initalia.virgilio.it/entrare-nella-storia-con-la-realta-aumentata-il-progetto-pilota-dei-musei-di-varese-16786,* ultima consultazione 29 agosto 2021.

Introduction to VR in Unity, *https://www.youtube.com,* ultima consultazione 17 giugno 2021.

Minneapolis Institute of Art 3D Models, *https://sketchfab.com/artsmia*, ultima consultazione 30 marzo 2021.

Missione Archeologica Italiana in Pakistan – MAIP, *https://www.ismeo.eu/portfolio_page/missione-archeologica-italiana-in-pakistan-maip*, ultima consultazione 30 agosto 2021.

Museo d'Arte Orientale, *https://www.maotorino.it/it,* ultima consultazione 30 agosto 2021.

Museum Next, V*irtual Reality is a big trend in museums, but what are the best examples of museums using VR?, https://www.museumnext.com/article/how-museums-are-using-virtual-reality,* ultima consultazione 25 agosto 2021.

Reddit, Unity 3D, *https://www.reddit.com/r/Unity3D,* ultima consultazione 24 agosto 2021.

Smithsonian National Museum of Natural History, Bone Hall, *https://naturalhistory.si.edu/exhibits/bone-hall*, ultima consultazione 25 agosto 2021.

TresessantaStudio, *Come la realtà aumentata viene utilizzata nei musei, hhttps://www.tresessantastudio.it/realta-aumentata-nei-musei*, ultima consultazione 25 agosto 2021.

Unity, *https://unity.com*, ultima consultazione 1 settembre 2021.

Unity Bolt VR Controller From Scratch, *https://www.* ultima consultazione 17 giugno 2021.

Unity Learn, *https://learn.unity.com,* ultima consultazione 31 agosto 2021.

VrFocus, *National Museum Of Finland Offers Virtual Time Travel, https://www.vrfocus.com/2018/02/national-museum-of-finland-offers-virtual-time-travel/*, ultima consultazione 22 agosto 2021.

Vuforia Developer Portal, *https://developer.vuforia.com,* ultima consultazione 26 agosto 2021.

Wessex Archaeology, Bath Abbey, *Virtual Reality experience an excavation below Bath Abbey, https://www.wessexarch.co.uk/our-work/virtual-reality-experience-excavation-below-bath-abbey*, ultima consultazione 23 agosto 2021.

Wessex Archaeology, *Virtual Reality experience Saxon homelife, https://www.wessexarch.co.uk/our-work/virtual-reality-experience-saxon-homelife,* ultima consultazione 23 agosto 2021.