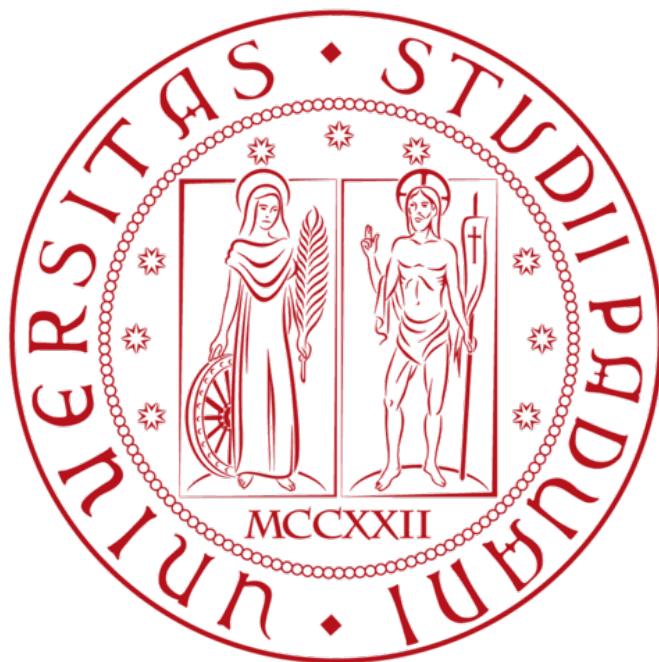


DIPARTIMENTO DI MATEMATICA

CORSO DI LAUREA IN INFORMATICA



Realizzazione di un configuratore di arredo in realtà aumentata

TESI DI LAUREA TRIENNALE

LAUREANDO

Davide Trivellato

RELATORE

Prof. *Tullio Vardanega*

ANNO ACCADEMICO 2014-2015

Reality exists in the human mind, and nowhere else.

Ringraziamenti

Desidero ringraziare prima di tutti i miei genitori Emanuela e Sergio che hanno sempre creduto in me e che mi hanno sempre supportato in ogni mia scelta, lasciandomi libero di esprimere la mia creatività in ogni sua forma e permettendomi di coltivare la mie passioni. Spero di non avervi deluso e che siate orgogliosi di me come io lo sono di voi.

Ringrazio mio fratello Gianluca, mia sorella Silvia e i miei parenti per avermi sempre fatto sentire l'affetto e il calore della famiglia.

Ringrazio il mio nipotino Tommaso per la sua simpatia e per avermi permesso di fargli da guida. Spero di aver fatto un buon lavoro e di non averlo mai deluso. Ringrazio le mie nipotine Tania, Gaia e Greta che anche se sono ancora piccole riescono sempre a mettermi il sorriso e il buon umore.

Ringrazio i miei amici di Bertipaglia: Matteo, Nicola, Gianmarco, Federico, Alberto, Filippo, Andrea. Mi sono sempre stati vicino e mi hanno supportato anche nei momenti più bui e difficili ricordandomi di non essere mai solo. A voi devo alcuni dei momenti più belli vissuti.

Vorrei ringraziare anche i miei amici da Rovigo: Marco e Massimo. Sono stati i miei compagni di "giochi" e mi hanno fatto scoprire una città molto bella e molto diversa da Padova inserendomi nel loro gruppo e permettendomi di conoscere tante nuove persone. Un grazie anche ai miei compagni di università: Maria Giovanna, Filippo, Matteo, Samuele, Enrico, Francesco. Con loro ho passato uno dei periodi più belli della mia carriera universitaria, fatto di gioie e dolori, di risate ma anche di tensioni. Il miglior team che potessi mai desiderare, siete fantastici.

Un grazie speciale va a Mauro, colui che mi sopporta dai tempi delle superiori. Lui è il mio Samvise Gamgee, gli devo tutto. Penso non abbia mai capito quanto importante sia stato per me in tutti questi anni di università ma senza di lui non so se sarei arrivato dove sono ora. È stato un bellissimo viaggio in tua compagnia che rifarei altre mille volte ancora!

Ringrazio Elena e Giulia, due Amiche con la "A" maiuscola. Possiamo non sentirci e non vederci per mesi ma la nostra amicizia non vacilla mai. Siete state il mio faro nelle notti più oscure.

Un ringraziamento va anche a quanti mi dicevano che non ce l'avrei mai fatta a laurearmi, perché mi hanno dato la motivazione per impegnarmi ancora di più.

Ringrazio tutte le persone che ci sono state e che ora non ci sono più, i nomi sarebbero troppi, ma grazie anche a loro per aver fatto parte della mia vita anche se per poco. Ogni batosta, ogni delusione è servita a farmi crescere e maturare facendomi diventare la persona che sono oggi.

Ringrazio il mio relatore, il prof. Vardanega, per essere stato cordiale e disponibile nei momenti di necessità.

Grazie a tutto il team di Experenti: Amir, Barbara, Gioele, Andrea, Nicolas, Veronica, Lucia ed Elisa. Mi hanno fatto sentire a casa, mi hanno fatto ridere ed emozionare, ma soprattutto, hanno creduto in me. Loro sono la mia seconda bellissima famiglia.

Grazie, infine a tutti quelli che mi hanno permesso di raggiungere questo piccolo traguardo, a voi prometto che mi impegnerò al massimo per fare sì che non sia l'ultimo.

Indice

1 L’azienda	1
1.1 Presentazione	1
1.2 Organizzazione aziendale	1
1.3 Prodotti e soluzioni offerte	2
1.3.1 Lo scenario contemporaneo	2
1.3.2 Il <i>trend</i>	3
1.3.3 Il <i>core business_g</i>	3
1.3.4 Il prodotto	4
1.3.5 Tipologie di prodotto	5
1.4 Processi aziendali	6
1.4.1 Modello di ciclo di vita <i>software_g</i>	6
1.4.2 Strumenti a supporto dei processi	9
1.5 Tecnologie utilizzate	10
1.5.1 Unity 3D	10
1.5.2 Vuforia SDK	11
1.6 Propensione all’innovazione	13
2 Strategia Aziendale	14
2.1 Motivazione dello stage	14
2.2 Obiettivo dello stage	14
2.3 Vincoli imposti	15
2.3.1 Vincoli tecnologici	15
2.3.2 Vincoli metodologici	19
2.3.3 Vincoli temporali	19
2.4 Prospettive	20
3 Resoconto dello stage	21
3.1 Pianificazione di progetto	21
3.1.1 Descrizione generale	21
3.1.2 Dettaglio delle attività	22
3.2 Studio delle tecnologie e strumenti	24
3.2.1 Unity 3D	24
3.2.2 Vuforia SDK	25
3.2.3 Photon Unity Networking	25
3.2.4 Esempio di contenuto in realtà aumentata a tema libero	27
3.3 Svolgimento delle attività	30
3.3.1 Il cliente	30
3.3.2 Analisi dei requisiti	31
3.3.3 Progettazione	34
3.3.4 Implementazione	41
3.3.5 Verifica e validazione	47
3.4 Livello di completezza raggiunto	49

4 Valutazione Retrospettiva	51
4.1 Soddisfacimento obiettivi	51
4.2 Conoscenze acquisite	52
4.3 Distanza tra università e lavoro	54
4.4 Valutazione personale	56
4.5 Screenshot finali	57
A Realtà Aumentata	60
Glossario	62
A	62
B	62
C	62
G	63
I	63
P	63
R	63
S	64
T	64
Bibliografia e Sitografia	65

Elenco delle figure

1	Logo di Experenti srl	1
2	Grafico <i>Hype Cycle</i> sui principali <i>trend</i> (2014)	3
3	Esempio di realta' aumentata - Planisfero	4
4	Esempio di realta' aumentata - Motore	5
5	Esempio di app configuratore	6
6	Flusso dei processi interni secondo la metodologia Agile	8
7	Esempio di <i>Kanban Board</i>	10
8	Interfaccia di Unity 3D	11
9	Architettura di Vuforia SDK e flusso di funzionamento delle operazioni di tracciamento delle immagini	12
10	Esempio sviluppato su Google Cardboard di AR/VR	13
11	Utilizzo dei sistemi operativi <i>mobile</i> nel secondo quarto del 2014 (dato fornito da Net <i>Applications</i>)	16
12	Versione base di Unity	18
13	Diagramma di Gantt _[g] delle attività	21
14	Implementazione in Unity del <i>tutorial</i> riguardante l'animazione di un <i>avatar</i> _[g] 3D di cui successivamente è stato eseguito il <i>porting</i> su Android	25
15	Architettura generale del <i>framework Photon Unity Networking</i>	27
16	Schermata iniziale di Call Of Toony con <i>Lobby</i>	28
17	Call Of Toony - <i>multiplayer</i> su device diversi	29
18	Call Of Toony - schermata di gioco	30
19	Diagramma dei casi d'uso generico relativo all'applicazione Cora' Parquet Live	32
20	Diagramma delle classi generico	35
21	<i>Layout</i> con menu' info aperto	38
22	<i>Layout</i> con menu inferiore aperto	38
23	<i>Animator Controller</i> del pannello Info laterale	40
24	<i>Animator Controller</i> dell' <i>avatar</i> _[g]	41
25	Presentazione iniziale	44
26	Applicazione visualizzata all'interno del Play Store	50
27	Presentazione iniziale - Ventaglio di legno in apertura	57
28	Presentazione iniziale - Avatar	57
29	Tutorial - Istruzioni	58
30	Configuratore - Menù inferiore aperto e Avatar in presentazione	58
31	Configuratore - Pannello info aperto	59
32	Configuratore - Menù dei Credits aperto	59
33	Esempio di realtà aumentata	60

Elenco delle tabelle

1	Tabella relativa alle ore dedicate per ciascuna attività	23
2	Tabella retrospettiva relativa alle ore dedicate per ciascuna attività	52

1 L'azienda

1.1 Presentazione

EXPERENTI srl nasce dalla brillante idea di sfruttare la tecnologia avanzata della realtà aumentata integrandola in un'ottica di *business_g* e *marketing* esperenziale. La *startup* è nata nel 2012 da una collaborazione tra l'Università di Padova e Mentis, società di consulenza strategica. Dopo due anni, è riuscita a crescere diventando, nel 2014, una vera e propria realtà aziendale ottenendo importanti investimenti che ne hanno permesso un rapido sviluppo ed una veloce espansione al punto da riuscire ad aprire una filiale a New York.



Figura 1: Logo di Experenti srl

1.2 Organizzazione aziendale

La filiale produttiva dell'azienda ha sede presso Massanzago (PD) in Via de Faveri 16, e conta nel suo organico due diversi *team*:

- un *team* con competenze relative al contesto commerciale
- un *team* focalizzato sulla parte tecnica e di gestione di progetto.

Il primo *team* si occupa degli aspetti commerciali, di *marketing* e di immagine dell'azienda. L'obiettivo principale del *team* è la ricerca e l'aggancio di nuovi clienti, nonché la fidelizzazione dei clienti già acquisiti. Per fare questo sono presenti diverse figure:

- CMO (*Chief Marketing Officer*): una figura con una preparazione in comunicazione e *marketing*, che conosce molto bene gli aspetti psicologici. Questa persona si occupa di capire come l'azienda viene percepita dall'esterno e cerca di costruirne un'immagine solida che ispiri fiducia e sicurezza nei possibili clienti. Un ulteriore compito è quello di fornire all'azienda la possibilità di partecipare ad eventi e fiere in ambito tecnologico e innovativo e di pubblicizzare l'azienda stessa.
- CEO (*Chief Executive Officer*): è l'amministratore delegato dell'azienda. Ne definisce le scelte strategiche, dal modello di *business_g* all'approccio al mercato. Segue direttamente le relazioni con i clienti chiave che generano oltre 150000 euro di *revenues_g*. Sviluppa le relazioni con investitori e *partner*.

Il secondo *team* si occupa della parte produttiva e della parte di ricerca e sviluppo. L'obiettivo primario del *team* è quello di soddisfare il cliente realizzando l'applicazione che più si avvicina alle sue aspettative e nel più breve tempo possibile, in modo efficace ed efficiente. Obiettivi secondari ma non di minore importanza sono la ricerca di nuove tecnologie e l'implementazione di metodologie che aumentino l'efficacia e l'efficienza con cui viene portato a termine il lavoro. Le principali figure della squadra sono:

- PM (*Project Manager*): il suo obiettivo essenziale è quello di raggiungere gli obiettivi di progetto, assicurando il rispetto dei costi, dei tempi e della qualità concordati e soprattutto il raggiungimento della soddisfazione del committente. Ha una forte preparazione economica.
- CTO (*Chief Technology Officer*): il suo ruolo è quello di monitorare le nuove tecnologie e valutarne il loro potenziale applicato ai prodotti e servizi; ma anche quello di supervisionare i progetti di ricerca per assicurare che portino valore aggiunto alla società.
- sviluppatore *software_g*: si prende cura di più aspetti del ciclo di vita del *software_g*, partendo dall'analisi, passando poi per progettazione e codifica e terminando con il *testing* e la validazione dell'applicazione.

Al momento l'azienda si compone di una decina di persone, ma è previsto un ampliamento dell'organico per fare fronte al crescente numero di progetti entranti.

1.3 Prodotti e soluzioni offerte

1.3.1 Lo scenario contemporaneo

Si può pensare che ormai è già stato tutto scoperto e inventato, ma non è così perchè ogni rivoluzione tecnologica apre un gigantesco universo di possibili applicazioni.

Quello in cui ci troviamo è un secolo che vede la più rapida espansione ed evoluzione dal punto di vista tecnologico nella storia dell'umanità. La crescita tecnologica si sviluppa ad una velocità tale che quello che oggi esce come una novità tra due mesi viene considerato vecchio e superato.

Inoltre, il modello capitalista adottato dai paesi più industrializzati e la globalizzazione impongono una continua ricerca ed aggiornamento dei propri prodotti e servizi per risultare più efficaci ed efficienti sul mercato. Per ottenere ciò, soprattutto negli ultimi decenni si sono cercate nuove vie di comunicazione per raggiungere il maggior numero di possibili clienti, investendo moltissimi soldi in campagne pubblicitarie.

Ci troviamo in un periodo temporale in cui siamo assuefatti da pubblicità di ogni genere, al punto da riconoscere un *brand* esclusivamente osservandone il *packaging*, oppure osservando il *design* di un prodotto. Ma la pubblicità tradizionale sta perdendo efficacia, soprattutto per il fatto che molti mezzi di comunicazione stanno sempre più cadendo in disuso. Questo è, per esempio il caso di riviste e televisori, che si sono viste superate da quello che web e dispositivi *mobile* stanno sempre più offrendo.

Lo scopo dell' AR (*Augmented Reality*) è quello di offrire nuove strade di comunicazione, da integrare ai dispositivi *mobile* (*smartphone*, *tablet* e visori).

1.3.2 Il trend

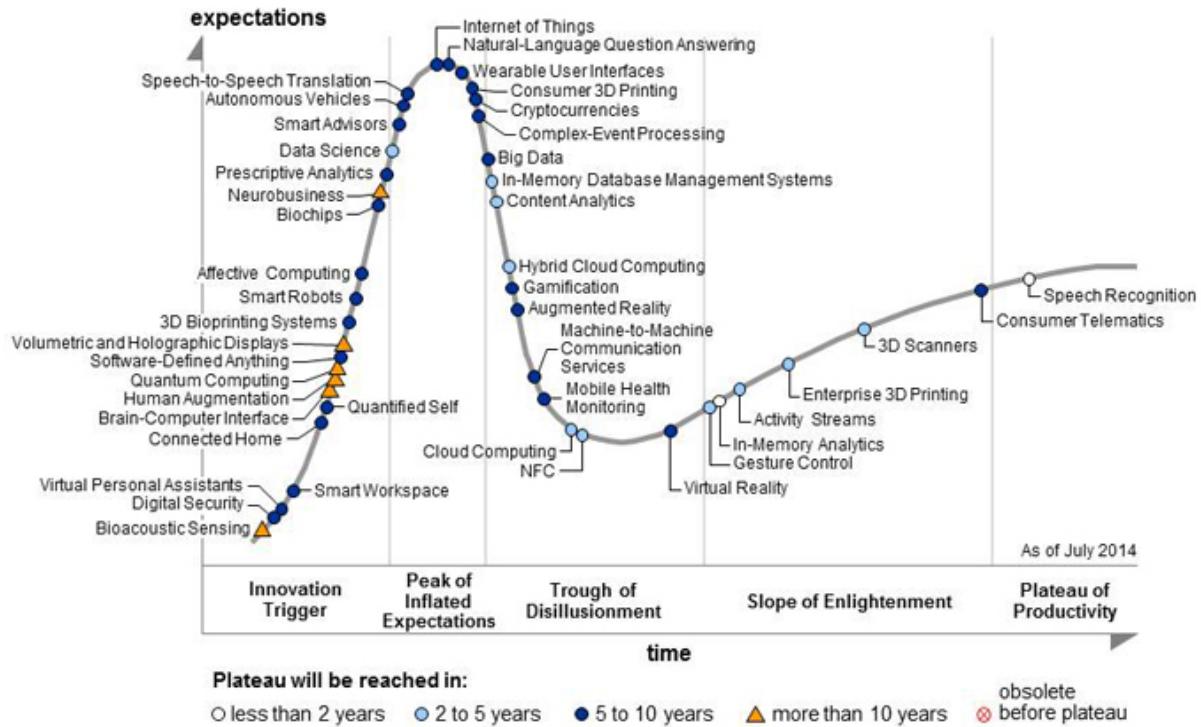


Figura 2: Grafico *Hype Cycle* sui principali trend (2014)

Quello che emerge da uno studio di Gartner Inc., multinazionale leader mondiale nel campo dell'*Information Technology*, è una interessante previsione sulla diffusione di alcune tecnologie emergenti. Lo strumento usato da Gartner è lo *Hype Cycle*, che consiste in una rappresentazione grafica che mette in scena il ciclo di vita di una tecnologia, dal suo concepimento, alla maturità, alla sua diffusione. Questo, inoltre, viene spesso utilizzato come punto di riferimento nel *marketing* e nel *reporting technology*, essendo allegati al grafico i rischi e le opportunità di queste tecnologie.

Nello *Hype Cycle* possiamo, per esempio, notare che per la realtà aumentata i tempi per un'adozione di massa sono stimati in un periodo compreso tra i 5 ed i 10 anni, e che quindi un investimento nel settore può portare a una crescita esponenziale dei ricavi dell'azienda.

1.3.3 Il core business_[g]

L'obiettivo che vuole raggiungere Experenti è quello di sfruttare la tecnologia della realtà aumentata per creare un servizio pubblicitario che faccia leva sulle emozioni dei consumatori e su quello che viene definito *wow factor* (espressione inglese che si riferisce a una qualità o una caratteristica che sorprende; letteralmente elemento sorprendente, o fattore sorpresa). L'obiettivo è quello di instaurare nell'utente finale un ricordo piacevole e associare quel ricordo a un marchio o a un prodotto.

1.3.4 Il prodotto

Il prodotto vero e proprio che viene creato dall'azienda e fornito ai committenti è una applicazione mobile che permette la visualizzazione e l'interazione con alcuni contenuti multimediali, come ad esempio video o modelli 3D, applicati a un "tag" riconosciuto tramite l'ausilio della fotocamera integrata nel device. Il funzionamento dell'applicazione prevede che, inquadrando con la fotocamera del proprio dispositivo *mobile* una particolare immagine o un oggetto 3D chiamati Tag, sia possibile visualizzare i contenuti multimediali associati, "aumentando" così le informazioni percepite attraverso nuovi canali informativi. I prodotti sviluppati da Experenti coprono la richiesta di diversi settori, tra i principali:

- architettura e arredamento;
- comunicazione ed editoria;
- turismo e cultura;
- istruzione e formazione;
- *smartcity* ed eventi.

Sono settori tendenzialmente distanti dal mondo tecnologico, per cui è di importanza cruciale un altissimo livello di usabilità dei prodotti. Per cui, il prodotto vero e proprio viene creato sulla base delle esperienze precedenti in termini di usabilità e di prestazioni (primo su tutti il risparmio energetico della batteria dei device), e sulla base di chi sarà l'utente finale dell'applicazione.



Figura 3: Esempio di realta' aumentata - Planisfero



Figura 4: Esempio di realta' aumentata - Motore

1.3.5 Tipologie di prodotto

L'azienda prevede la scelta fra tre diversi tipi di pacchetto, ognuna studiata per venire incontro alle diverse disponibilità di *budget* dei clienti:

- **inserimento di contenuti interni all'app Experenti:** la prima soluzione e quella più economica. Questa offerta consiste nell'inserimento di nuovi tag e nuovi contenuti associati all'interno di un visore di realtà aumentata già esistente avente come *brand* Experenti. Non è prevista la modifica della struttura dell'applicazione che funge da contenitore di elementi provenienti da diverse fonti e destinati a diversi utenti. L'applicazione è distribuita sia su dispositivi Android che su dispositivi iOS.
- **inserimento di contenuti interni ad un'app personalizzata:** è la soluzione intermedia, che prevede la creazione di un'app personalizzata sulla base della struttura che possiede l'app di Experenti. È quindi possibile cambiare nome, logo, palette di colori e contenuti andando a completare le fondamenta *standard* dell'app.
- **inserimento di contenuti interni ad un'app dedicata:** è la soluzione meno economica ma più completa tra quelle presentate. Consiste nella realizzazione di un'app fornendo completa libertà di personalizzazione e costruita su misura del cliente.

I prodotti tipicamente realizzati sono di due tipologie diverse:

- **app visore:** è l'idea su cui si basa l'app Experenti e cioè quella di fornire un semplice visualizzatore di contenuti, sia video che modelli 3D. È data la possibilità di interazione con i contenuti e vengono fornite le funzionalità per scattare *screenshot* e abilitare il *flash* della fotocamera.
- **app configuratore:** è un particolare tipo di applicazione che vede la sua migliore implementazione in ambito di architettura e arredo. Questo particolare tipo di app permette all'utente di "sfogliare" un catalogo di prodotti e di visualizzarli a grandezza naturale per avere una visione d'insieme all'interno del proprio locale o all'esterno. È possibile vedere i prodotti nelle proprie varianti e di confrontarli tra di loro. L'applicazione prevede tipicamente un menù inferiore per scorrere gli elementi e un pannello laterale per visualizzare le informazioni relative a ogni singolo oggetto. È fornita la possibilità di effettuare ricerche all'interno del catalogo tramite *keyword*.

È prevista l'introduzione di una nuova tipologia di prodotto ancora in fase di sviluppo che è il **visore di video configurabile da web**. Tutte le altre applicazioni che non rientrano in queste tipologie di prodotto non sono ancora state standardizzate e rientrano nella tipologia di **applicazioni custom**.



Figura 5: Esempio di app configuratore

1.4 Processi aziendali

1.4.1 Modello di ciclo di vita *software_g*

Come modello di ciclo di vita *software_g*, l'azienda ha deciso di adottare una metodologia AGILE. Si riferisce a un insieme di metodi di sviluppo del *software_g* emersi a partire dai

primi anni 2000 e fondati su insieme di principi comuni, direttamente o indirettamente derivati dai principi del "Manifesto per lo sviluppo agile del *software_g*". Experenti ha scelto questo modello perché, lavorando in un ambiente altamente innovativo, necessita di prediligere le iterazioni con gli individui esterni e la collaborazione con il cliente oltre a una rapida reazione al cambiamento. Questo tipo di modello si prefigge di ottenere *software_g* funzionante tralasciando aspetti importanti ma non essenziali quali, per esempio, una documentazione completa. I principi su cui si basa una metodologia Agile che seguono i punti indicati dall'Agile Manifesto, sono quattro:

- le persone e le interazioni sono più importanti dei processi e degli strumenti (ossia le relazioni e la comunicazione tra gli attori di un progetto *software_g* sono la miglior risorsa del progetto);
- più importante avere *software_g* funzionante che documentazione (bisogna rilasciare nuove versioni del *software_g* ad intervalli frequenti, e bisogna mantenere il codice semplice e avanzato tecnicamente, riducendo la documentazione al minimo indispensabile);
- bisogna collaborare con i clienti oltre che rispettare il contratto (la collaborazione diretta offre risultati migliori dei rapporti contrattuali);
- bisogna essere pronti a rispondere ai cambiamenti oltre che aderire alla pianificazione (quindi il *team* di sviluppo dovrebbe essere pronto, in ogni momento, a modificare le priorità di lavoro nel rispetto dell'obiettivo finale).

La gran parte dei metodi agili tenta di ridurre il rischio di fallimento sviluppando il *software_g* in finestre di tempo limitate chiamate iterazioni che, in genere, durano qualche settimana.

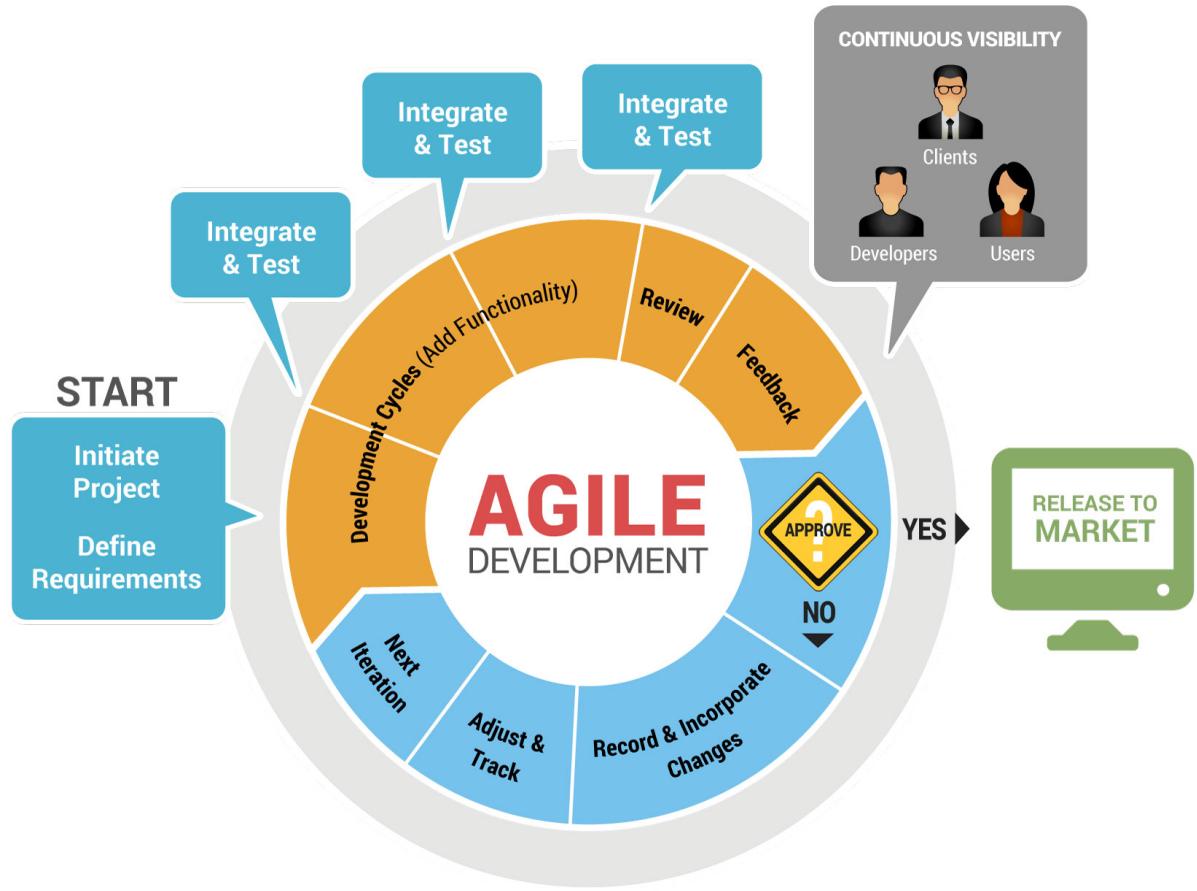


Figura 6: Flusso dei processi interni secondo la metodologia Agile

Il *software_g*, all'interno dell'azienda viene sviluppato in finestre di tempo limitate chiamate *iterazioni* che, in genere, durano dalle 2 alle 4 settimane. Ogni iterazione può essere considerata come un piccolo progetto a sé stante e deve contenere tutto ciò che è necessario per rilasciare un piccolo incremento nelle funzionalità del *software_g*: pianificazione (*planning*), analisi dei requisiti, progettazione, implementazione, test e documentazione. La comunicazione con il cliente avviene quotidianamente, fornendo da parte dell'azienda *screenshot* o video sulle funzionalità e ottenendo dal cliente *feedback* e nuove richieste. Anche se il risultato di ogni singola iterazione non ha sufficienti funzionalità da essere considerato completo deve essere rilasciato e, nel susseguirsi delle iterazioni, deve avvicinarsi sempre di più alle richieste del cliente. Alla fine di ogni iterazione il *team* rivaluta le priorità di progetto, viene eseguita una nuova pianificazione e una nuova progettazione in modo da ottenere un sostanziale incremento alla prossima iterazione, fino al completo soddisfacimento del cliente. Se in corso di progettazione in seguito a una richiesta di modifica dei requisiti ci si accorge che alcune funzionalità richiedono un numero troppo elevato di risorse rispetto a quanto preventivato ci si accorda con il cliente per trovare un compromesso in modo tale che non resti deluso.

I metodi agili preferiscono la comunicazione in tempo reale, preferibilmente faccia a faccia, a quella scritta (documentazione). Il team agile è composto da tutte le persone

necessarie per terminare il progetto *software_g*. Come minimo il *team* deve includere i programmatori ed i loro clienti (con clienti si intendono le persone che definiscono come il prodotto dovrà essere fatto: possono essere dei *product manager*, dei *business_g analysts*, o veramente dei clienti).

1.4.2 Strumenti a supporto dei processi

1.4.2.1 Sistemi operativi Il lavoro viene svolto in ambiente Microsoft Windows 8.1, anche se a fine agosto è iniziato l'aggiornamento di alcune macchine a Windows 10. Una parte del team di sviluppo, invece, lavora in ambiente MacOS principalmente per la compilazione e pubblicazione di app per iOS che ne rendono l'utilizzo necessario. I programmi utilizzati quali Photoshop, Gimp, Unity, sono *cross-platform_g* quindi non è un problema lo sviluppo su sistemi diversi. Per quanto riguarda il *server* aziendale vediamo la presenza di un sistema Linux così come per gli ambienti *cloud*, in quanto si presta molto bene a quello scopo. Per il *testing* delle applicazioni vengono utilizzati dispositivi Android aggiornati all'ultima versione 5.1.1 e dispositivi iOS aggiornati alla versione 8.4.1.

1.4.2.2 Gestione del versionamento Per la gestione del versionamento viene utilizzato internamente Tortoise SVN che è un *client* grafico *Subversion*. Si è scelto il suo utilizzo in quanto, oltre ad essere *open source*, è stato scritto per girare come estensione di Microsoft Windows e quindi perfettamente integrabile nel sistema operativo usato per lo sviluppo *software_g*. I progetti realizzati sono contenuti in un *repository* che risiede nei server interni e gestito dal reparto tecnico dell'azienda.

1.4.2.3 Enterprise Resource Planning Come sistema di gestione per integrare tutti i processi di *business_g* rilevanti, l'azienda ha scelto di utilizzare **Odoo**, ossia un *software_g* ERP *Open Source* maturo per la gestione di piccole e medie imprese. Odoo integra, tramite moduli, tutti i processi necessari all'impresa come:

- gestione della contabilità;
- gestione delle risorse umane;
- gestione di vendite e acquisti;
- gestione dei progetti;
- gestione documentale.

Odoo è noto per essere molto completo ed estremamente modulare, con più di 1000 moduli disponibili. È basato su una robusta architettura *Model-View-Controller*, con un *server* distribuito, *workflow* flessibili, una *GUI_g* dinamica e *report* personalizzabili. Le funzionalità principali per cui è stato scelto Odoo sono:

- **Kanban Board:** utilizzata per organizzare in modo ottimale il lavoro e avere una visione generale sullo stato dei singoli progetti. È molto usata soprattutto per il fatto che ci si trova ad agire seguendo un modello di sviluppo molto dinamico e

soggetto a continui cambiamenti. L'utilizzo della *Kanban Board* porta ad eliminare una classe di problemi e sprechi nell'attività produttiva attraverso un approccio sistematico ovvero creando un ambiente di lavoro che rende difficile commettere errori.



Figura 7: Esempio di *Kanban Board*

- **Gestione presenze e richiesta di permessi:** entrate e uscite sono gestite da Odoo così come la richiesta di permessi, in questo modo risulta semplice capire la disponibilità di personale a breve e lungo termine.
- **Calendarie note condivise:** grazie ai calendari e alle note condivise è possibile avere una visione di insieme altrimenti difficile da osservare.

L'utilizzo di Odoo si è rivelato di importanza fondamentale per gestire in modo efficiente il tempo del personale, ma anche per avere sempre sotto mano le priorità su un progetto piuttosto che un altro, oppure avere sempre un elenco descrittivo delle attività da svolgere durante la giornata.

L'utilizzo delle note condivise è usato soprattutto in ambito *bug fixing*, in quanto viene tenuta traccia della soluzione a un particolare *bug* riscontrato in una applicazione e quindi rintracciabile in futuro da altri che incontrano le stesse problematiche.

1.5 Tecnologie utilizzate

1.5.1 Unity 3D

Unity 3D è una potente e flessibile piattaforma di sviluppo utilizzata per la creazione di giochi 2D e 3D, oltre che per esperienze interattive. È un ecosistema completo per chiunque miri a costruire un *business_{lg}* sulla creazione di contenuti di alto livello. I punti forti che hanno portato alla scelta di questa piattaforma sono essenzialmente:

- **Migliore motore di gioco mobile:** la caratteristica principale è che Unity è il migliore motore di gioco per lo sviluppo di applicazioni mobile. Permette migliaia di ottimizzazioni per ridurre il peso degli *assets_g* e opzioni dedicate per ottenere la migliore resa grafica sui vari dispositivi *mobile*.
 - **Supporto al multiplatform:** Unity che permette la creazione dei contenuti una volta sola e la distribuzione su tutte le principali piattaforme *mobile*, *desktop* e *console*, semplicemente con un click.
 - **Partnership di successo:** Unity trae molti benefici dalla forte e positiva *partnership* instaurata con colossi di piattaforme e costruttori di *hardware* come Microsoft, Sony, Qualcomm, Intel, Samsung, Oculus VR e Nintento.

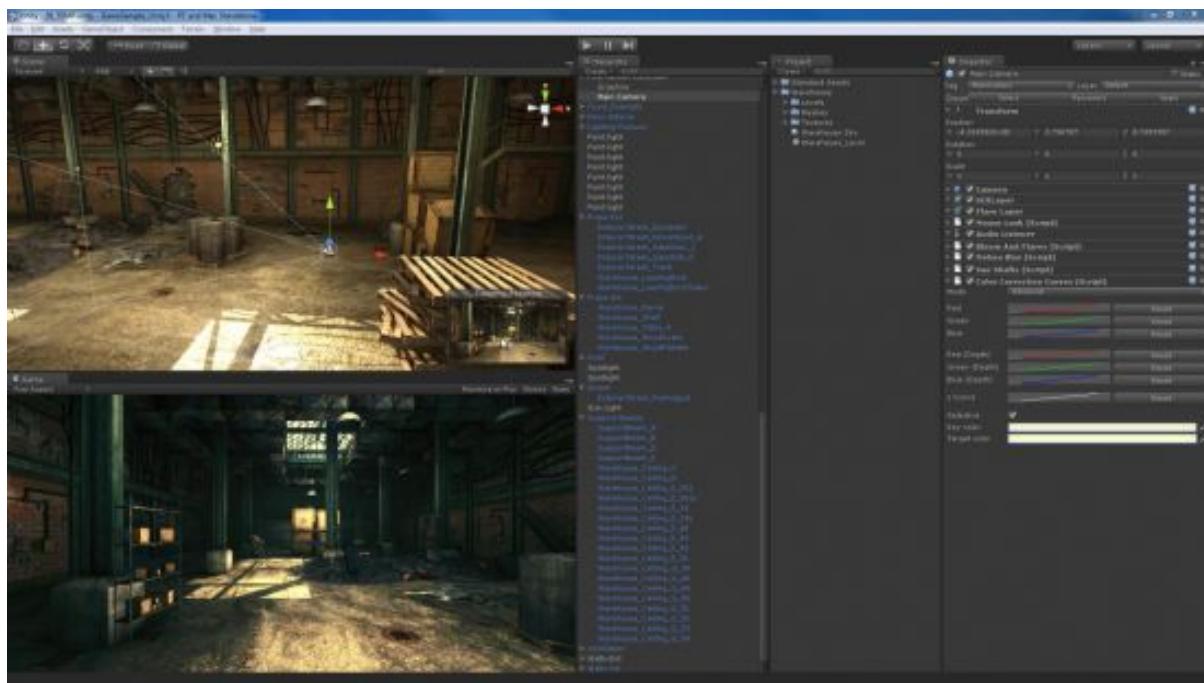


Figura 8: Interfaccia di Unity 3D

1.5.2 Vuforia SDK

Vuforia è la piattaforma *software*_[g] che consente le migliori e più creative esperienze di realtà aumentata attraverso gli ambienti del mondo reale, dando alle applicazioni *mobile* il potere di "vedere" attraverso la fotocamera del *device*. La piattaforma Vuforia utilizza un efficiente e stabile riconoscimento delle immagini basato sulla visione artificiale. La visione non è intesa solo come acquisizione di una fotografia bidimensionale di un'area ma soprattutto come l'interpretazione del contenuto di quell'area. È stato scelto l'utilizzo di Vuforia per il suo pieno supporto ad iOS, Android e Unity 3D.

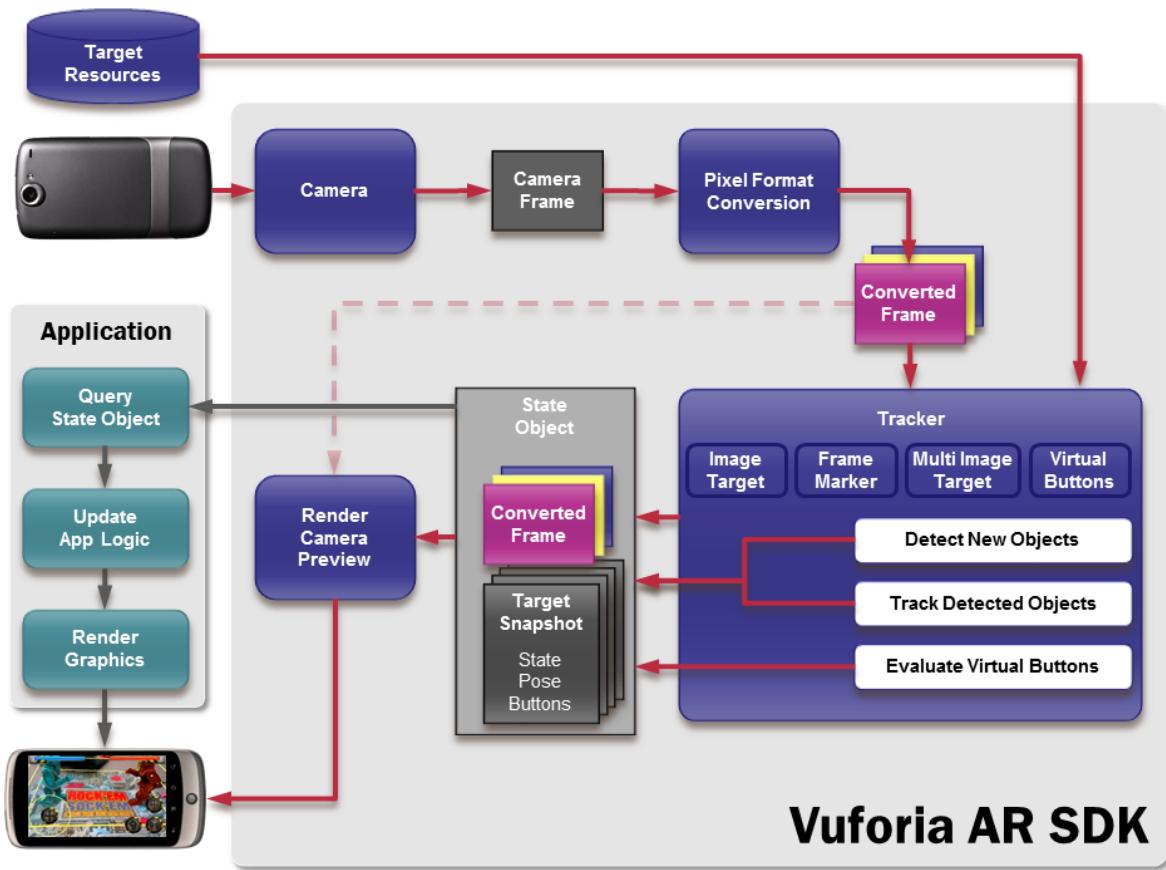


Figura 9: Architettura di Vuforia SDK e flusso di funzionamento delle operazioni di tracciamento delle immagini

Di seguito vengono spiegati i componenti dell'architettura di Vuforia SDK AR.

- **Application:** viene eseguita su *device*. In base ai dati di *input* avviene l'aggiornamento di stato di alcuni oggetti forniti da Vuforia che servono ad aggiornare l'*App Logic* e a renderizzare la grafica sullo schermo.
- **Camera:** assicura che tutti i frame della camera vengano passati al *Pixel Format Conversion* per ulteriori elaborazioni.
- **Pixel format Conversion:** converte i *frame* provenienti dal modulo *Camera* in modo da essere riconosciuti dal rendering e dal tracking dell'OpenGL ES. Questo modulo è necessario in quanto le fotocamere sono diverse da dispositivo a dispositivo e forniscono diversi formati di *frame*.
- **Target Detection:** Vuforia SDK fornisce il riconoscimento del tag in tre forme:
 - tag definiti dall'utente;
 - tag interni all'applicazione;

- tag gestiti in *cloud*.

I tag definiti dall'utente sono definiti usando un'algoritmo interno disponibile in Vuforia SDK. Per la seconda tipologia di tag, è necessario il caricamento delle immagini sul portale di sviluppo di Vuforia, per poi scaricare un file da utilizzare in Unity. L'ultima tipologia usa il riconoscimento ricercando i tag tra quelli caricati sul portale di sviluppo di Vuforia.

- **Tracker:** è il cuore di Vuforia SDK dove sono scritti tutti gli algoritmi di visione computerizzata per ogni tipologia di tag (immagini, cilindri, etc.). Il modulo si occupa di creare oggetti di stato, in base ai dati ricevuti, che verranno poi utilizzati dall'applicazione sviluppata.

1.6 Propensione all'innovazione

Come già accennato in precedenza, Experenti sta cavalcando l'onda di un *trend* molto caldo, e l'arrivo sul mercato dei visori di realtà virtuale e realtà aumentata sono un passo in avanti che l'azienda è pronta ad affrontare. Durante il mio periodo di stage ho potuto osservare come il *team* di sviluppo sia sempre aggiornato sulle nuove tecnologie e sulle *release* di nuove versioni di tecnologie già utilizzate. L'azienda ha già avviato ricerche e sperimentazioni per:

- l'utilizzo oggetti tridimensionali reali come tag;
- la creazione di applicazioni in realtà aumentata senza l'utilizzo di alcun tag;
- l'implementazione della realtà aumentata su visori AR/VR e quindi la creazione di applicazioni di realtà aumentata mista realtà virtuale.

L'azienda ha già portato avanti alcuni esempi di realtà aumentata sui Google Cardboard con discreti successi. Anche se l'innovazione è un aspetto molto importante per l'azienda, non può essere prioritaria per il fatto che le risorse sia umane che finanziarie sono ancora limitate e assegnate in primo luogo alla produzione.

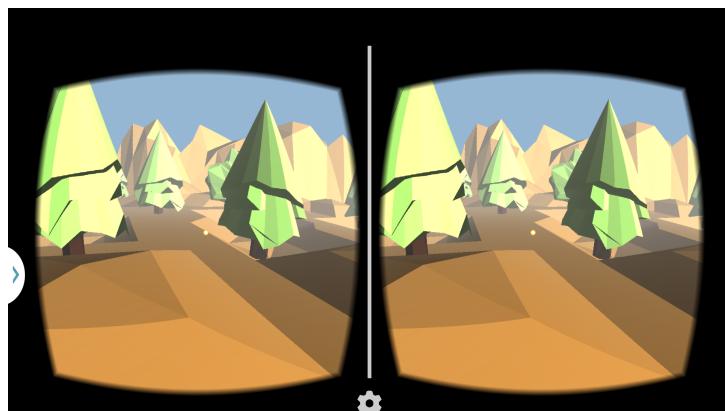


Figura 10: Esempio sviluppato su Google Cardboard di AR/VR

2 Strategia Aziendale

2.1 Motivazione dello stage

Lo stage ha potuto svolgersi grazie all'evento STAGE-IT 2015 che ha permesso l'incontro tra le imprese e gli studenti che sarebbero entrati a breve in stage nel mondo del lavoro con specifico riferimento al settore ICT. L'evento ha favorito un'occasione di conoscenza reciproca mediante colloqui individuali.

Experenti sta vivendo un momento di forte crescita, e ha visto nell'ultimo periodo un aumento del numero di progetti in ingresso. Per fare fronte alla richiesta, l'azienda ha deciso di espandere il suo organico anche in una possibile ottica di inserimento post-stage. Il team di Experenti richiedeva un laureando in Informatica che possedesse un'ottima capacità di programmazione ad oggetti, la conoscenza di C# e una propensione per la parte di progettazione propedeutica al *coding* vero e proprio. Inoltre, era apprezzata una qualche esperienza con modellazione, *rendering* 3D e con il motore grafico Unity 3D.

Per una azienda avviata da soli due anni, è di fondamentale importanza gestire in modo ottimale le risorse, soprattutto quelle finanziarie. Per cui, l'azienda ha valutato positivamente il fatto di poter prendere uno stagiaire a tempo limitato senza obbligo di retribuzione, in modo da avere a disposizione ulteriori forze nell'immediato, per gestire il notevole numero di progetti entranti in quel periodo.

Non è stata una scelta dettata esclusivamente dalla necessità di manodopera, però, in quanto il tempo di formazione dello stagiaire comportava un dispendio iniziale di risorse, dato che era necessario l'affidamento di un tutor aziendale per l'insegnamento delle metodologie, dell'utilizzo degli strumenti e delle *best practices* presenti in azienda.

2.2 Obiettivo dello stage

Lo stage prevedeva la suddivisione delle attività in due parti: la prima prettamente formativa, ha occupato circa il 60% del periodo di stage, mentre la seconda, che ha occupato il successivo tempo restante, si è concentrata sulla parte produttiva dell'attività aziendale, in particolar modo sulla parte orientata alla realizzazione di progetti destinati ai clienti esterni.

Come **obiettivo minimo** era richiesto di sviluppare almeno un singolo contenuto complesso in realtà aumentata (ovvero: non video AR semplice e non 3D statico AR) da inserire all'interno di un'app commissionata da un cliente esterno.

Mentre, come **obiettivo massimo** era richiesto di sviluppare un'intera app visore di AR, completa di tutti i suoi contenuti semplici e complessi e della propria grafica, dalla fase di accettazione dei materiali in entrata fino alla fase di consegna della beta finale al cliente. Il progetto che avrei dovuto seguire non era stabilito sin da subito, ma è stato concordato a stage già avviato, in seguito all'ingresso di un progetto commissionato da Corà Divisione Parquet, di cui parlerò in seguito.

Il progetto consisteva nella realizzazione di un configuratore di arredo in realtà aumentata e nella gestione di un *avatar_{lg}* 3D che effettuasse una presentazione iniziale e si occupasse di seguire l'utente durante l'utilizzo dell'app con una spiegazione sulle varie categorie di prodotto.

Entrando nel dettaglio, era richiesto di partire da un configuratore di prodotto di base, che consiste in un'applicazione tramite la quale gli utenti possono scegliere un modello di prodotto e le caratteristiche desiderate, e una volta definiti possono mandare una e-mail di richiesta preventivo oppure essere rimandati al sito web.

Il prodotto di base da cui bisognava partire era un configuratore, già realizzato, di stufe comprensivo di menù inferiore per la selezione delle categorie e dei prodotti, ed un pannello laterale mostrante la descrizione di ogni prodotto. Nel configuratore, inoltre, era già presente uno script per gestire l'*auto-focus* della *camera* del *device* e un mirino con un piccolo pulsante per scaricare il tag nel caso non fosse già disponibile all'utente.

Definita la base di partenza, la prima parte del progetto, era la creazione della GUI personalizzata partendo da una grafica in formato PSD. Successivamente, bisognava inserire i primi prodotti all'interno del configuratore e quindi gestire i singoli dati riguardanti un prodotto in modo da fornire, in futuro, l'eventuale possibilità di effettuare ricerche tramite l'inserimento di *keyword* in un'apposita casella di *input* testuale.

Si richiedeva, inoltre, l'implementazione di una *feature* nuova, per fornire la *gesture* di *pinch-to-zoom* realizzata ad-hoc per una pavimentazione irregolare che è quella del parquet. Era richiesta un'estensione della superficie coperta dal parquet in concomitanza ad una particolare *gesture* (in questo caso un "pizzico" sullo schermo del device) applicata all'oggetto tridimensionale che si stava visualizzando.

La seconda parte del progetto consisteva nella gestione di un *avatar_g* umanoide in realtà aumentata e di un modello tridimensionale rappresentante un ventaglio in legno da usare come sipario per la comparsa dell'*avatar_g*. L'*avatar_g* doveva effettuare una presentazione iniziale dell'applicazione e delle tipologie di prodotto, e si voleva poterlo richiamare successivamente, tramite un apposito pulsante, per richiedere spiegazioni riguardanti particolari categorie di prodotto concordate con il committente.

2.3 Vincoli imposti

Per tutta la durata di svolgimento dello stage sono state imposte alcune condizioni da rispettare, spiegate di seguito, divise per tipologia.

2.3.1 Vincoli tecnologici

2.3.1.1 Android Android è un sistema operativo personalizzabile per dispositivi mobili sviluppato da Google Inc. basato su kernel Linux. È stato progettato principalmente per *smartphone* e *tablet*, con interfacce utente specializzate per televisori (Android TV), automobili (Android Auto), orologi da polso (Android Wear), occhiali (Google Glass), e altri. È per la quasi totalità *Free and Open Source Software*, ed è distribuito sotto i termini della licenza libera Apache 2.0.

Android dispone di una vasta comunità di sviluppatori che realizzano applicazioni con l'obiettivo di aumentare le funzionalità dei dispositivi. Queste applicazioni sono scritte soprattutto in linguaggio di programmazione Java.

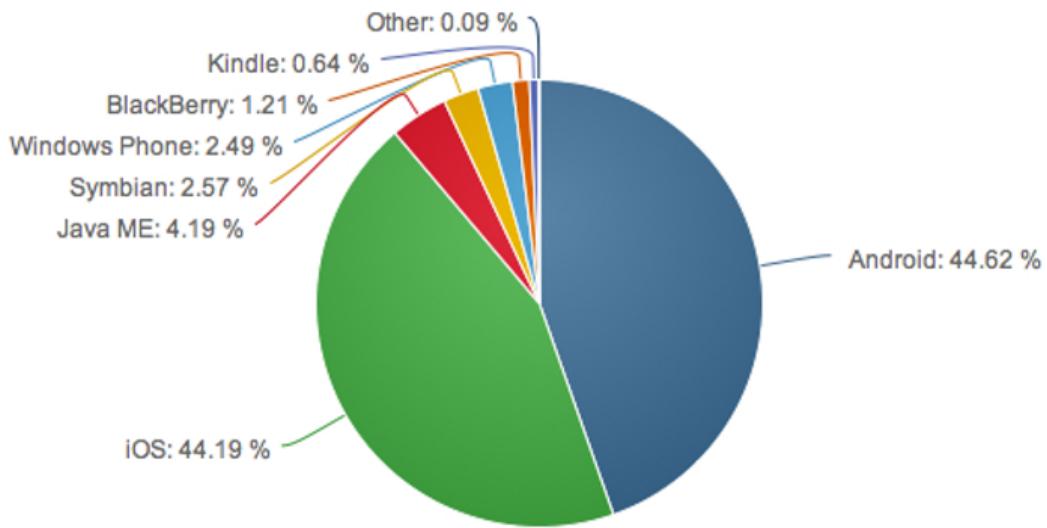


Figura 11: Utilizzo dei sistemi operativi *mobile* nel secondo quarto del 2014 (dato fornito da *Net Applications*)

Si è scelto di sviluppare il progetto, in prima istanza, per i dispositivi Android su richiesta del committente (anche se la release dell'app è avvenuta anche su iOS) e soprattutto perché, mentre Apple vive in un ambiente chiuso e ben definito, per quanto riguarda Android ci si trova a dover fare i conti con la diversità di *hardware* dei dispositivi e diverse risoluzioni degli schermi. Per lo sviluppo dell'app si predilige Android, in quanto possono sorgere più problematiche, e dato che Unity, comunque, offre la possibilità di sviluppare l'app solo una volta e distribuirla a più dispositivi diversi.

2.3.1.2 Unity 3D Come già spiegato in precedenza, Unity 3D è un ambiente di sviluppo per la creazione di giochi. Come prima cosa fornisce un potente *engine* di supporto. Il motore in questione offre il supporto completo per tutti gli aspetti quali *rendering* grafico, effetti di luce, creazione di *terrain*, simulazioni fisiche, implementazione dell'audio, funzionalità di rete e, cosa più importante, un sistema di *scripting*. Unity semplifica di molto la vita del programmatore con piccole feature di qualità, come per esempio la *"live preview"*, che consente di vedere dal vivo quello che si sta creando, con un solo click. Inoltre, mette a disposizione un sistema di gestione delle risorse da usare nel progetto efficace ed intuitivo. I formati di file supportati dall'*engine* sono molteplici. Per i modelli 3D si ha il pieno supporto ai file generati da:

- Maya;
- 3D Studio Max;
- Cinema4D;
- Blender;
- SketchUp;
- Cheetah;
- Lightwave;
- XSI;
- Carrara;

- Wavefront Obj.

Mentre per quanto riguarda le immagini, i formati supportati sono:

- | | | |
|---------|--------|---------|
| • JPEG; | • BMP; | • PICT; |
| • PNG; | • TGA; | |
| • GIF; | • IFF; | |

Anche a livello di audio il sistema si difende piuttosto bene:

- | | |
|---------------|---------|
| • MP3; | • AIFF; |
| • Ogg Vorbis; | • WAV. |

Infine, per i video si ha supporto a:

- | | | |
|--------|--------|---------|
| • MP4; | • AVI; | • ASF; |
| • MPG; | • MOV; | • MPEG. |

Uno dei punti di forza di Unity 3D è il fatto di essere gratuito in quasi tutte le sue *features*. Tuttavia, per particolari scopi o esigenze di pubblicazione, ci sono alcuni strumenti o parti del sistema che sono a pagamento. Unity è la versione base che viene rilasciata gratuitamente con quasi tutte le funzionalità più importanti a disposizione. Unity Pro, la versione a pagamento, consente allo sviluppatore di usufruire di diverse *features* non presenti nella versione normale; su tutte, l'assenza dello *splash screen* di Unity, cioè della schermata iniziale mostrante il logo di Unity. Per usi commerciali, l'azienda necessita l'utilizzo della versione Pro, anche per il fatto che questa versione mette a disposizione utilissimi effetti di *render* su *texture_g*, di *post processing* ed effetti su luci e ombre.

Nella figura 12 si può osservare una tipica schermata di lavoro di Unity nella versione base (non Unity Pro) composta da cinque sezioni ben distinte:

1. **Scene**: La scena è lo spazio di lavoro in cui vengono posizionati gli oggetti di gioco e in cui è possibile avere una visione "grezza" di come apparirà la nostra applicazione. In questa zona è possibile effettuare azioni sugli oggetti, quali modificare la scala, cambiare la loro posizione sugli assi, ruotarli o modificare le loro ancore e pivot.
2. **Game**: questa sezione è dove viene mostrata la "*live preview*" dell'applicazione. Schiacciando il pulsante "*play*", infatti, sarà possibile avviare l'applicazione e interagire con essa, con la possibilità di poter usufruire di tutte le *gesture mobile* tramite l'uso di tastiera e mouse. Per simulare la *camera* del dispositivo viene usata una webcam collegata al computer.
3. **Hierarchy**: è la gerarchia di oggetti di gioco istanziati. Qui si possono tenere sotto controllo tutti gli oggetti presenti nella scena e modificarne le parentele.
4. **Project**: nella sezione *Project* vengono inserite tutte le risorse che si vuole rendere disponibili nel progetto come *texture_g*, *script*, modelli 3D, etc. Qui vengono inseriti anche gli oggetti non istanziati nella scena, ma che devono essere istanziati *runtime*.

5. **Inspector:** l'*inspector* è una sezione importantissima in cui si possono modificare tutti i parametri di ogni oggetto di gioco e agganciare nuove componenti come per esempio *animator*, *collider*, etc.

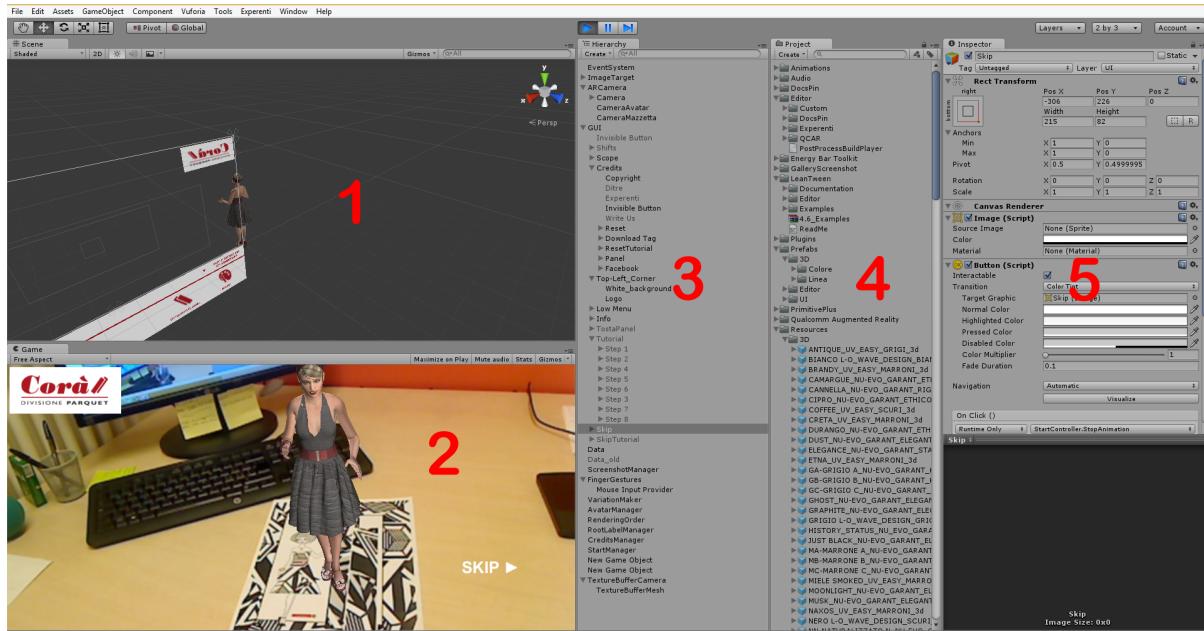


Figura 12: Versione base di Unity

Anche il *layout* delle sezioni da utilizzare in Unity mi è stato imposto, in quanto più persone lavorano sullo stesso progetto ed è necessario non rimanere disorientati da una diversa disposizione dell'interfaccia.

2.3.1.3 OpenGL ES OpenGL ES è uno *standard* industriale per la programmazione grafica 3D su dispositivi *mobile*. Khronos Group, un conglomerato che include marchi come ATI, NVIDIA ed Intel si preoccupa di definire ed estendere lo *standard*. Attualmente esistono molte versioni diverse delle specifiche OpenGL ES. La versione 1.0 venne ricalcata sulla versione 1.3 di OpenGL, mentre la versione 1.1 si basa su OpenGL 1.5 e la 2.0 è definita in relazione a OpenGL 2.0. La versione di OpenGL ES utilizzata era la 2.0 poichè, ormai, la gran parte dei dispositivi Android supporta quel tipo di libreria grafica.

2.3.1.4 Vuforia SDK Come già descritto in precedenza, è stato imposto il vincolo di utilizzo di Vuforia SDK, rispetto al suo principale *competitor* Metaio. La scelta è stata fatta alle origini dell'azienda, ed è stata fatta sulla base del supporto fornito in quanto documentazione, supporto e *tutorial*.

Vuforia è una piattaforma completa che offre *features* di spessore, quali:

- Rilevamento istantaneo dei tag locali;

- Riconoscimento *cloud* fino a 1 milione di tag simultanei;
- Riconoscimento e *tracking* di testo stampato;
- *Tracking* contemporaneo fino a 5 tag;
- Risultati eccezionali con condizioni ambientali sfavorevoli come tag semi-coperti e carenza di luce.

Rispetto a Metaio, Vuforia SDK presenta un *tracking* migliore e una migliore integrazione con Unity 3D, che ne hanno dettato la scelta rispetto all'utilizzo di Metaio e altri competitor minori.

2.3.1.5 C# Per la scrittura del codice è stato imposto l'utilizzo di C#, in quanto è un linguaggio molto simile al Java, di cui avevo già una buona preparazione. L'alternativa all'utilizzo di C# sarebbe stata Javascript, un linguaggio molto potente ma che avrebbe richiesto ulteriore tempo di formazione. Essendo che più persone collaborano sullo stesso progetto, si è ritenuto indispensabile la piena comprensione da parte di tutti i membri del team di sviluppo del linguaggio utilizzato.

2.3.2 Vincoli metodologici

Durante tutta l'attività di stage, è stato imposta la gestione del versionamento tramite l'utilizzo di TortoiseSVN, inserendo le componenti in un *repository* interno al *server* locale dell'azienda. Non mi è stato fornito alcun vincolo su quando effettuare i *commit*, la cui gestione è stata lasciata a me.

2.3.3 Vincoli temporali

Il progetto prevedeva che lo stagiaire svolgesse un totale di 320 ore di attività presso l'azienda ospitante, suddivise in circa 40 ore settimanali soggette a possibili variazioni nel caso di scadenze aziendali o di impegni di varia natura da parte dello studente. Tali ore si dovevano svolgere internamente all'orario d'ufficio, dal lunedì a venerdì dalle 9:00 alle 13:00 e dalle 14:30 alle 18:30.

Le prime date concordate di inizio e fine stage sono state, rispettivamente, 2015-07-09 e 2015-09-09. In seguito a problematiche sorte da parte del tutor aziendale a ridosso della data di inizio stage si è deciso di riconcordare nuovamente le date di inizio e fine stage, rispettivamente, 2015-07-13 e 2015-09-11.

Il periodo concordato è stato suddiviso in due sezioni temporali della stessa dimensione:

- la prima parte prettamente formativa, consisteva nello studio delle tecnologie utilizzate e nella realizzazione di un piccolo progetto per l'assimilazione dei concetti appresi;
- la seconda parte, iniziata in conclusione della parte formativa, consisteva nella realizzazione del progetto vero e proprio.

2.4 Prospettive

Come descritto in precedenza, Experenti vuole ampliare il suo organico, inserendo nel *team* alcune nuove figure con una preparazione in ambito informatico e grafico. Il numero sempre crescente di progetti entranti rendono necessario l'inserimento di sviluppatori Android e iOS che seguano attivamente i progetti dalla fase di raccolta dei materiali, alla fase di progettazione, codifica e *testing*.

Il settore altamente innovativo e appena nato della realtà aumentata, comporta la possibilità di lavorare su progetti all'avanguardia in ambito AR. Basta osservare il lavoro svolto nel mio stage per accorgersi delle continue migliorie che gli si possono applicare. Experenti presta molta attenzione nella ricerca di clienti che possano offrire motivazione nel cercare soluzioni per il raggiungimento di un grado di innovazione sempre maggiore e provare, quindi, ad essere sempre un passo avanti rispetto ai *competitor*.

3 Resoconto dello stage

3.1 Pianificazione di progetto

3.1.1 Descrizione generale

Come già detto in precedenza, lo stage è stato suddiviso in due parti: la prima orientata alla formazione su strumenti e tecnologie e la seconda parte orientata alla realizzazione di progetti destinati ai clienti esterni. Come tale, l'attività di formazione è stata opportunamente orientata all'apprendimento, da parte mia, delle meccaniche e delle norme vigenti internamente per lo sviluppo di tali progetti, oltre che alla normale parte di formazione tecnica prevista per portare a termine in maniera opportuna le attività dei progetti stessi. L'obiettivo finale dello stage è stato quindi quello di inserirmi come parte integrante del *team* di sviluppo per i progetti esterni, attribuendomi responsabilità e compiti adeguati al mio ruolo e orientati alle attività di produzione, *testing* e *delivery* di app *mobile* di Realtà Aumentata; la valutazione finale da parte del tutor aziendale è stata quindi effettuata sulla base sia della qualità sia della quantità delle attività portate a termine nella fase produttiva finale, oltre che alla capacità di lavorare correttamente in squadra con l'obiettivo comune di consegnare un prodotto finale nei tempi e nelle modalità stabilite.

Nel periodo antecedente l'inizio dello stage, insieme al tutor aziendale, sono state concordate le attività principali che avrei dovuto svolgere durante il periodo seguente della durata di 2 mesi. Nella descrizione delle attività, riportata nella sezione successiva, è stata fornita una descrizione molto generica per quanto riguarda il progetto principale che avrei dovuto seguire, in quanto non era ancora chiaro a priori se ci sarebbe stata o meno la possibilità di seguire un progetto commissionato dall'esterno.

La dislocazione temporale delle attività è stata rappresentata graficamente in un Diagramma di Gantt_[g] che mi ha aiutato ad avere sempre una visione accurata sullo stato del mio stage, in particolare su eventuali ritardi. Rispetto al diagramma concordato nel piano di lavoro, il mio stage è partito dopo 2 giorni rispetto a quanto concordato a causa di un'indisponibilità del tutor aziendale, per cui è stata rifatta la pianificazione tenendo conto di questo ritardo.

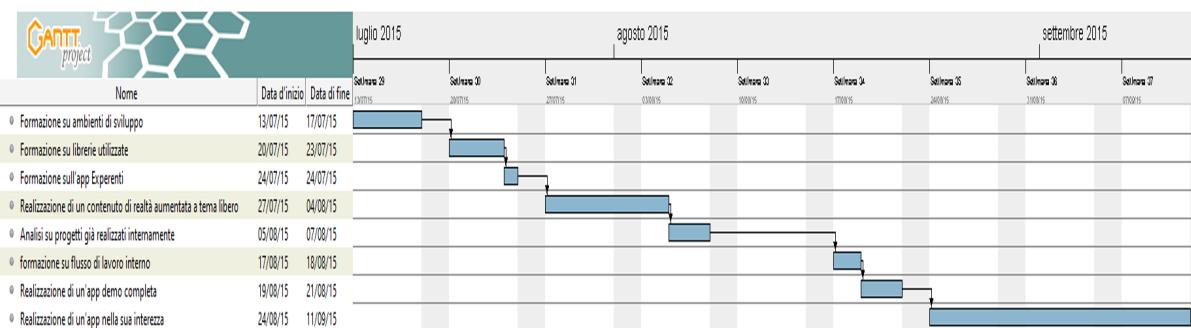


Figura 13: Diagramma di Gantt_[g] delle attività

Il Diagramma di Gantt_{|g|} riportato in figura 13 mostra piuttosto fedelmente quanto svolto durante il periodo in azienda ed eventuali anticipi sulla tabella di marcia sono stati riempiti con approfondimenti sulle tecnologie e sessioni di ricerca e sviluppo su visori Google Cardboard. Il tempo guadagnato dal periodo di formazione ha permesso di iniziare prima il progetto principale. Non ci sono stati, invece, ritardi su quanto preventivato.

3.1.2 Dettaglio delle attività

Di seguito vengono elencate in dettaglio le attività svolte durante il periodo di stage presso l'azienda ospitante Experenti. Un approfondimento per le principali.

1. Formazione sulle tecnologie utilizzate internamente per lo sviluppo, quali *framework* e SDK. In particolare:
 - (a) Ambiente di sviluppo (IDE_{|g|}) utilizzato (Unity3D) e fondamenti dei sistemi operativi mobile (Android e iOS);
 - (b) Formazione sulle librerie utilizzate internamente per l'elaborazione delle immagini per la realtà aumentata e per il successivo riconoscimento delle stesse in ambiente mobile;
 - (c) Formazione sull'app Experenti: nascita del progetto, funzionamento attuale, obiettivi di sviluppo. Formazione sulle procedure *standard* applicate internamente.
2. Realizzazione di un esempio di contenuto in Realtà Aumentata a tema libero. Questo contenuto, il cui sviluppo è stato necessario alla comprensione del flusso di lavoro interno e all'individuazione di determinate problematiche relative all'ambito AR mobile, ha particolari caratteristiche, quali animazioni e/o movimenti di parti specifiche, un certo grado di interattività e prevede parti semplici di grafica GUI_{|g|} (su schermo, in modalità HUD). È stata richiesta, inoltre, l'individuazione di un tag adatto al riconoscimento dalle fotocamere *mobile*, possibilmente legato alla tematica che è stata sviluppata.
3. Analisi di casi di studio e app varie già realizzate internamente. Focus particolare sui progetti base già realizzati e sulla loro struttura: progetto base demo, progetto base visore AR, progetto base configuratore. In questa fase è avvenuta la formazione sul flusso di lavoro standard interno all'azienda e sul normale iter di un progetto commissionato da un cliente, dalla ricezione dei materiali fino alla fase di distribuzione (sia essa una distribuzione ad hoc o una distribuzione pubblica tramite *Store mobile*) ed è iniziato l'affiancamento al *Project Manager* nelle fasi di accettazione materiali.
4. Realizzazione di un'app demo completa. Per app demo si intende un'app a distribuzione solitamente ad hoc (non pubblicata sugli *Store*) resa disponibile dall'azienda per i propri clienti o *reseller*, comprendente un numero solitamente limitato di contenuti semplici (3D o video) fruibili dall'utente in realtà aumentata attraverso l'uso di un tag fornito dal cliente stesso. L'app possiede, inoltre, una GUI_{|g|} minimale

3 RESOCONTO DELLO STAGE

ma personalizzata con il logo del cliente stesso, nonchè un'icona e una *splashscreen* anch'esse personalizzate allo stesso modo. Richiesto l'affiancamento al *Project Manager* fin dalla fase iniziale di ricezione materiali, e prosecuzione poi in autonomia nella fase di sviluppo fino alla fase di rilascio e consegna (previa verifica del risultato prodotto da parte del tutor aziendale). L'entità dell'app demo è stata stabilita dal *Project Manager* aziendale alcuni giorni prima dell'inizio di questa fase e si è data preferenza, alla produzione di una demo per un cliente esterno.

5. Inserimento effettivo nel *team* di sviluppo per i progetti esterni. In questa fase, inizia l'affiancamento al *team* di sviluppo per i progetti commissionati dai clienti esterni; è iniziato quindi il coordinamento dal *Project Manager* aziendale nell'assegnazione di task appositi comprendenti le fasi di sviluppo e testing di intere app semplici o parti di app complesse; si è preferito assegnare la realizzazione di almeno un'app semplice nella sua interezza commissionata da un cliente esterno. L'assegnazione delle attività è stata effettuata attraverso il sistema di *ticketing* utilizzato internamente all'azienda, attraverso il quale è stato anche richiesto di rendicontare le proprie attività in termini di tempo utilizzato per ciascuna di esse, mentre l'assegnazione dei singoli *task* è stata effettuato dal *Project Manager* aziendale in collaborazione con il tutor aziendale. È stato valutato positivamente in questa fase la capacità di attenersi alle tempistiche date e il livello di dettaglio fornito nella successiva rendicontazione delle ore, oltre ovviamente alla qualità intrinseca del risultato prodotto.

Sezione	Descrizione	Ore di lavoro
1.1	Formazione su ambienti di sviluppo	40
1.2	Formazione su librerie utilizzate	28
1.3	Formazione sull'app Experenti	12
2	Realizzazione di un contenuto di realtà aumentata a tema libero	56
3	Analisi su progetti già realizzati internamente e formazione su flusso di lavoro interno	40
4	Realizzazione di un'app demo completa	24
5	Inserimento nel <i>team</i> di sviluppo e realizzazione di un'app nella sua interezza	120
TOTALE		320

Tabella 1: Tabella relativa alle ore dedicate per ciascuna attività

3.2 Studio delle tecnologie e strumenti

In questa sezione, vengono spiegate le attività di apprendimento svolte per imparare l'utilizzo delle nuove tecnologie e degli strumenti usati. La maggior parte delle tecnologie apprese mi era prima di allora sconosciuta. Grazie al progetto didattico svolto per il corso di Ingegneria del Software, avevo una solida base per quanto riguarda la tecnologia Android e non ho avuto particolari problemi nell'apprendimento autonomo delle tecnologie sconosciute.

3.2.1 Unity 3D

Unity, come già detto in precedenza è un sistema *cross-platform* per lo sviluppo di giochi composto da un *game engine* e da un IDE integrato. Unity viene usato internamente all'azienda per lo sviluppo di app *mobile* distribuite su Android e iOS.

Unity nel suo sito fornisce un grosso supporto agli sviluppatori rendendo disponibile una documentazione completa e una sezione ben fornita di *tutorial* testuali e video suddivisi per categoria.

Inizialmente, ho dovuto seguire una parte di video *tutorial* riguardanti l'interfaccia di Unity, lo *scripting*, la gestione della fisica, animazioni e gestione della GUI. Questo primo periodo si è svolto integrando, oltre alla visione, anche la prova diretta sull'*editor* in modo da assimilare meglio i concetti appresi.

Nel caso in cui avessi voluto approfondire un argomento oppure non lo avessi ritenuto abbastanza chiaro, avevo sempre la possibilità di ottenere una spiegazione da parte del *tutor* aziendale, il quale si è dimostrato sempre molto disponibile anche nel ripetere più volte lo stesso concetto.

In questa parte di formazione, dopo aver seguito e implementato un *tutorial* riguardante l'animazione di un *avatar*, di mia iniziativa, ho effettuato il porting dell'applicazione su Android gestendo *touch* e *multi-touch* sullo schermo e impostando i movimenti dell'*avatar* basandoli sull'accelerometro del dispositivo *mobile*.

3 RESOCONTO DELLO STAGE

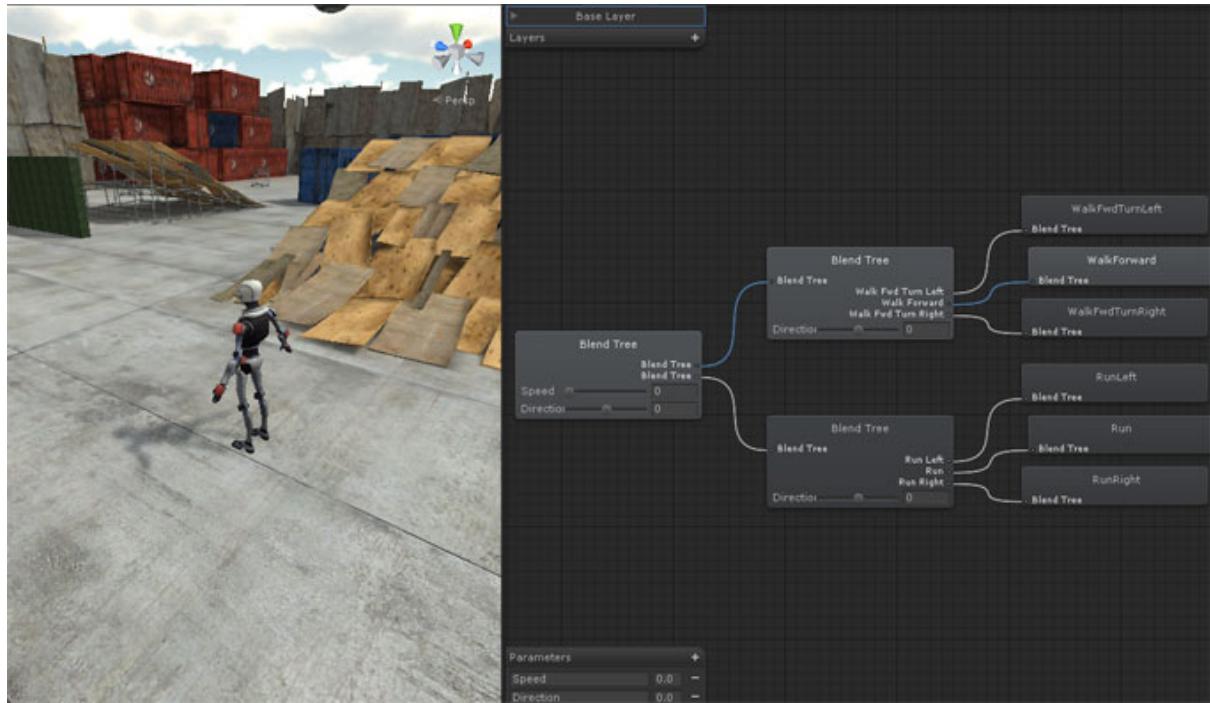


Figura 14: Implementazione in Unity del *tutorial* riguardante l’animazione di un *avatar_g* 3D di cui successivamente è stato eseguito il *porting* su Android

3.2.2 Vuforia SDK

Terminata la parte di formazione su Unity 3D, è iniziata la parte di preparazione relativa a Vuforia SDK, l’SDK utilizzato dall’azienda per l’implementazione della realtà aumentata. Nello specifico, il *team* tecnico si è occupato di spiegarmi come funziona l’SDK, e come funziona nello specifico il *plugin* di Unity, grazie alla quale è possibile operare all’interno di un unico ambiente di lavoro. Le attività principali svolte in questo lasso di tempo sono state l’implementazione di modelli 3D e di video associati ai tag, e lo studio sul riconoscimento e la creazione di tag ottimali.

3.2.3 Photon Unity Networking

Photon Unity Networking (PUN) è un *framework* di Unity per l’implementazione del *multiplayer realtime* nei giochi o nelle applicazioni sviluppate. Le applicazioni sviluppate con Photon vengono eseguite su un *server cloud* proprietario. Quindi, le operazioni di *scaling* e di *service hosting* sono gestite interamente da PUN, permettendo allo sviluppatore di concentrarsi puramente sulla costruzione dell’applicazione. Tutti i prodotti Photon Cloud sono basati su un’architettura *client-server_g*, che è la soluzione ottimale per il *gaming online* rispetto ad una connessione *peer-to-peer*.

Photon è un *package* scaricabile dall’*Asset Store* (negozi di Unity *online* in cui comprare o scaricare gratuitamente *asset_g* come *script*, modelli 3D e 2D, etc.), e nella sua versione gratuita prevede l’accesso concorrente fino a 20 utenti sulla stessa stanza.

Di seguito vengono riportati esempi di codice per mostrare la semplicità di utilizzo del *framework*.

- **Connessione al *server*:** La connessione al *server* si basa sul passaggio di una stringa contenente la versione dell'applicazione. Può essere usata per dividere gruppi di *client*.

```
1 PhotonNetwork.ConnectUsingSettings("1.0");
```

- **Accesso a una stanza:** Per prendere parte a una partita esistente basta la seguente riga di codice specificando il nome della stanza in cui si vuole entrare.

```
1 PhotonNetwork.JoinRoom("RoomName");
```

- **Creare una stanza:** per creare una stanza basta fornire il nome, dare la possibilità o meno di essere trovata da altri utenti, fornire la possibilità agli altri di entrare, e il numero massimo di giocatori.

```
1 public void OnConnectedToMaster() {  
    2     PhotonNetwork.CreateRoom("RoomName", true, true, 4);  
3 }
```

Lo studio di *Photon Unity Networking* non mi è stato imposto dall'azienda, ma è stato un approfondimento che ho voluto fare di mia iniziativa per lo sviluppo dell'esempio di contenuto in realtà aumentata a tema libero di cui parlerò successivamente.

PUN si è dimostrato uno strumento molto potente e relativamente di facile utilizzo. Creare uno scambio di dati tra diversi *client* è risultato piuttosto semplice. Il livello di difficoltà è salito quando ho cercato di aumentare il numero di informazioni passate e il numero di oggetti da "osservare". Essendo uno studio non richiesto dall'azienda, ho preferito non spendere troppo tempo in approfondimenti, ma piuttosto avere un'idea chiara del funzionamento di base.

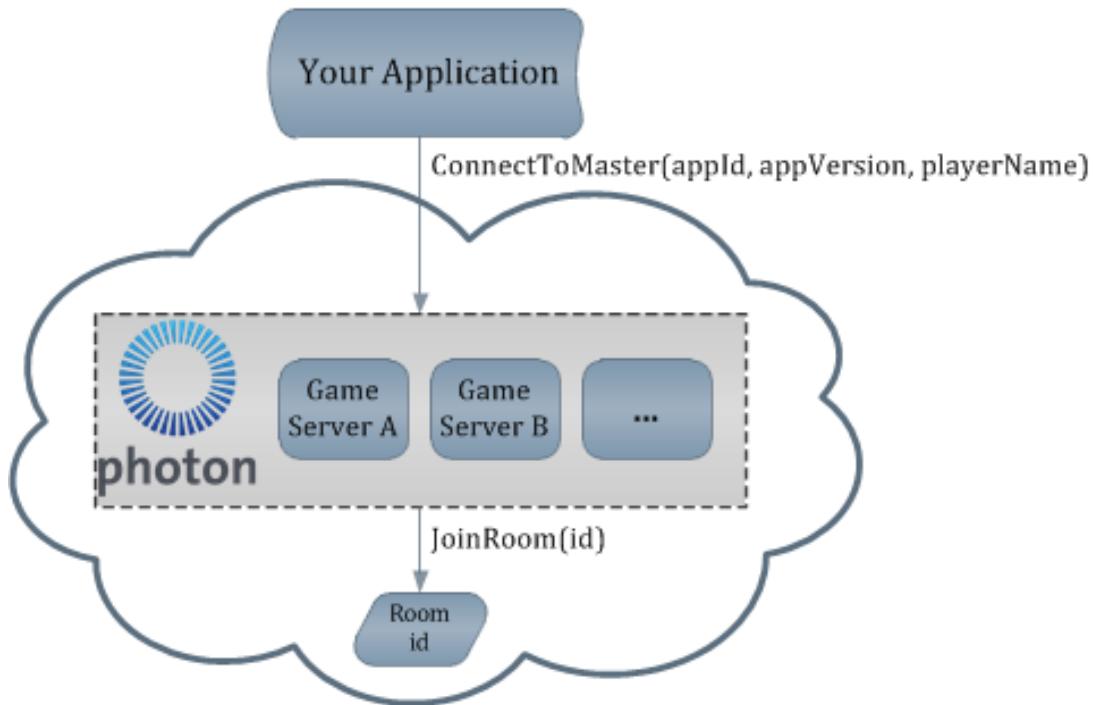


Figura 15: Architettura generale del *framework Photon Unity Networking*

3.2.4 Esempio di contenuto in realtà aumentata a tema libero

Come ultimo passo a compimento del percorso di preparazione tecnologica e strumentale, e prima di iniziare la realizzazione del progetto principale di stage, mi è stato chiesto di sviluppare un'applicazione completa con dei contenuti non banali in realtà aumentata. Ciò che mi è stato reso disponibile per la realizzazione dell'app oltre a Unity 4.6.3 e Vuforia SDK 4, sono stati anche tutti i contenuti gratuiti disponibili sull'*Asset Store*. Ho quindi proceduto con un'accurata scansione dei contenuti scaricabili trovando ciò che fosse più utile per dare luce all'idea in stato embrionale che avevo in mente.

Quello che stavo cercando erano dei modelli 3D in stile *"cartoon"* per lo sviluppo di un videogioco sparatutto *multiplayer* in realtà aumentata.

L'idea che volevo implementare era dare la possibilità a più utenti di interagire sullo stesso tag, e rendere tali interazioni visibili ad utenti che si trovassero dall'altra parte del mondo. La prima idea che mi era venuta, è stata quella di gestire un oggetto 3D e dare la possibilità di modificarne la struttura, il colore e la scala agli utenti, in modo che i cambiamenti fossero visibili a tutti gli utenti in osservazione su quel dato tag. Questo, però, non mi è bastato, in quanto volevo vedere fino a che punto si poteva spingere un *device* Android con il *rendering* di contenuti in realtà aumentata. Per cui la realizzazione di un gioco mi sembrava perfetta per testare questi due aspetti.

Come già detto, ho effettuato uno studio di *Photon Unity Networking* per l'implementazione del *multiplayer* e successivamente ho proceduto con una semplice progettazione architettonica e con l'implementazione vera e propria.

La prima cosa che ho fatto è stata la realizzazione della mappa di gioco, cercando di renderla il più simmetrica possibile in modo da non sfavorire nessuno dei due giocatori

e creando delle barriere invisibili per non permettere ai giocatori di poter uscire dallo scenario.

Il passo successivo è stato quello di creare dei punti di *respawn* e di gestire la nascita dei giocatori in modo casuale sulla mappa. Avendo trovato nell'*Asset Store* un modello 3D già in possesso di animazioni e *script* che simulassero un soldato *cartoon* vero e proprio, non ho dovuto occuparmidella gestione dell'*avatar_g*.

Una volta creati e gestiti i punti di *respawn*, ho iniziato il lavoro di configurazione della *lobby* (contenitore di stanze) in primo luogo, e delle stanze successivamente, fornendo la possibilità ai giocatori di creare la propria stanza e di vedere le stanze create dagli altri giocatori. Per la *lobby* ho creato una schermata iniziale apposita di scelta dello *username* e di creazione nuova stanza o accesso a una stanza già creata.

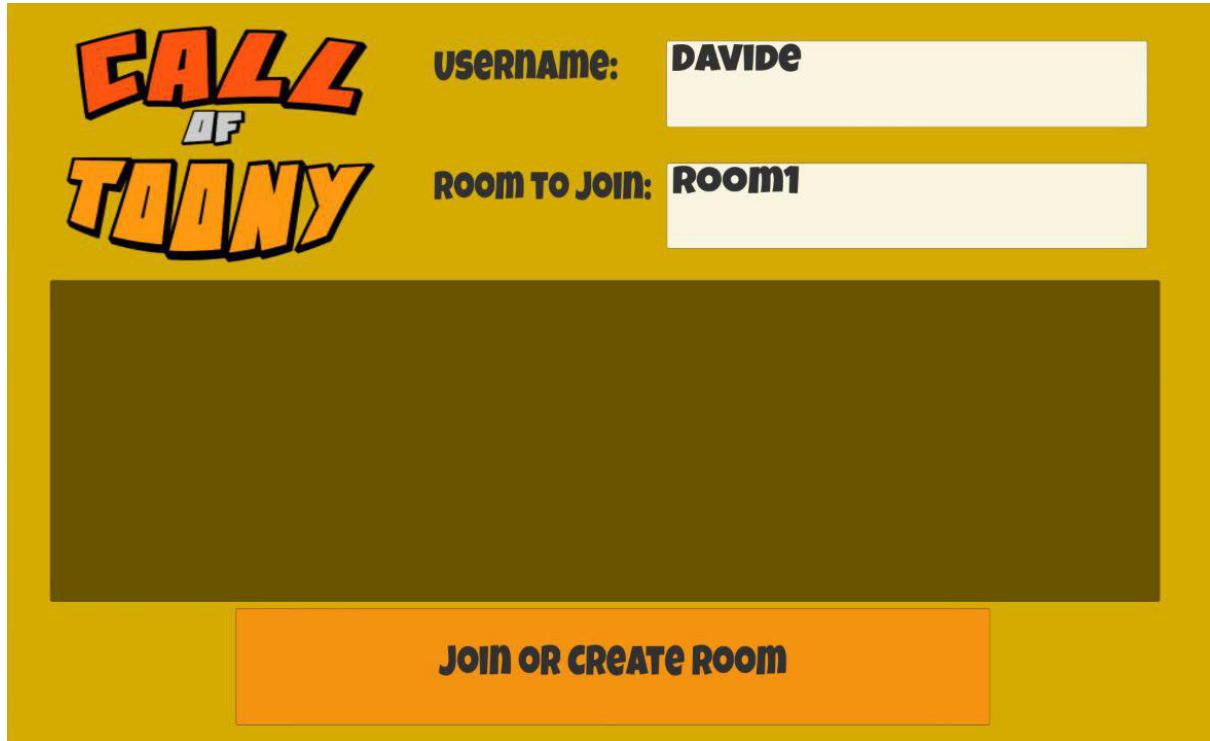


Figura 16: Schermata iniziale di Call Of Toony con *Lobby*

In seguito, sono stati gestiti gli *script* relativi al passaggio dei dati attraverso la rete. In particolare, i dati passati sono:

- posizione e rotazione dei personaggi rispetto alla mappa;
- percentuale di salute rimasta sulla *health bar*;
- bersaglio colpito e chi ha colpito il bersaglio.

3 RESOCONTO DELLO STAGE

La gestione di questi dati ha permesso la creazione di un gioco basilare, a cui, successivamente, ho aggiunto un'area in cui rappresentare i *log* della partita su un angolo di schermo, quali: *spawn* dei giocatori e le uccisioni avvenute. Inoltre, localmente ho reso disponibile il numero di morti del proprio personaggio, mentre non c'è stato abbastanza tempo per gestire anche il numero di uccisioni.

Il risultato è stato un gioco fluido e ben strutturato, in grado di gestire il *multiplayer* ad una latenza bassissima e in grado di coinvolgere il giocatore in un'esperienza nuova e unica. I problemi riscontrati sono dovuti unicamente ai *device*, i quali a lungo termine presentano surriscaldamento, consumo elevato di batteria e un calo di *frame* per secondo. Il progetto è piaciuto molto al *team* tecnico, che non ha escluso la possibilità, dopo un'accurata ottimizzazione, di un inserimento del contenuto in realtà aumentata all'interno dell'app Experenti.

Il nome del gioco è **Call Of Toony**, in richiamo a titoli videoludici del genere ben più noti.

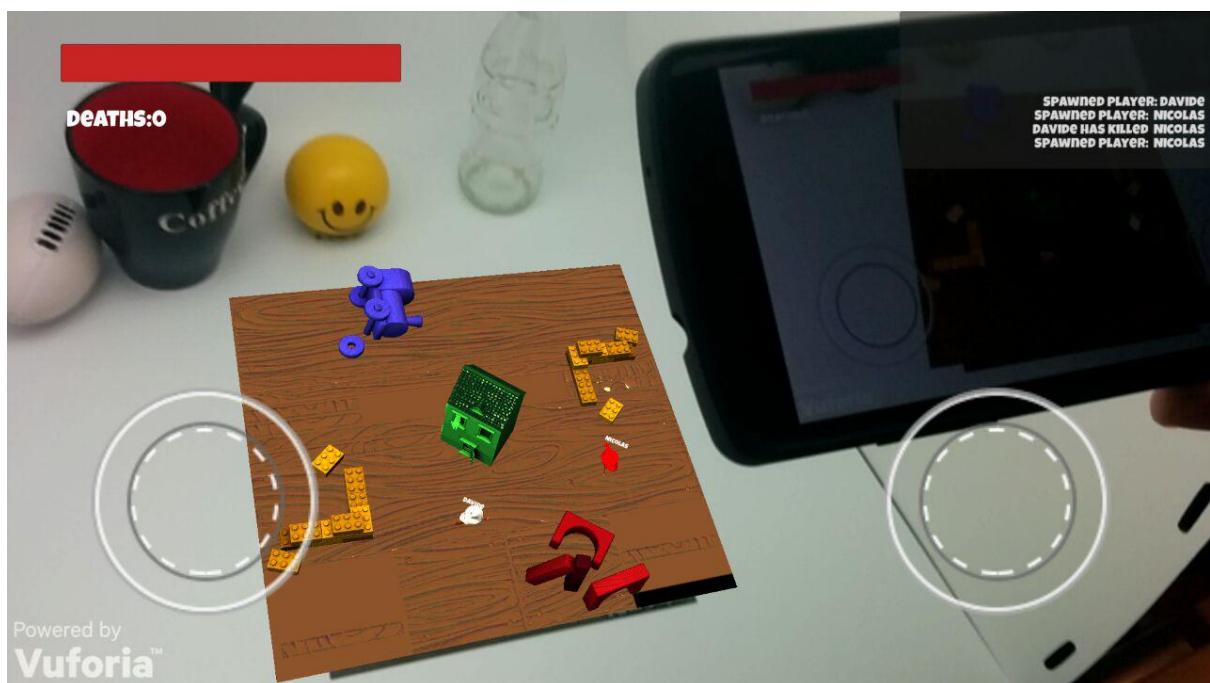


Figura 17: Call Of Toony - *multiplayer* su device diversi

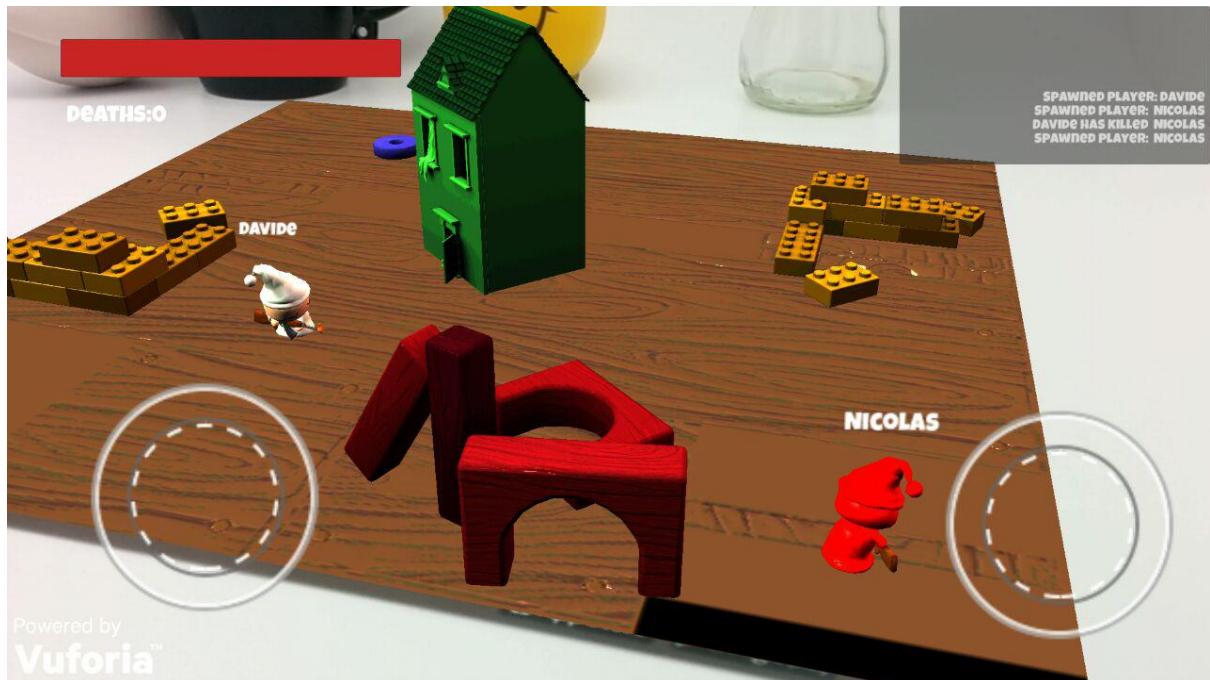


Figura 18: Call Of Toony - schermata di gioco

3.3 Svolgimento delle attività

Inizia ora la stesura di quanto svolto nel progetto vero e proprio, ripercorrendo tutte le fasi del ciclo di vita del *software* sviluppato fino al suo rilascio sui vari *store*. Per lo svolgimento del progetto sono state previste circa 120 ore inserite all'interno dell'orario di lavoro aziendale, dal lunedì al venerdì.

Il nome dell'applicazione da sviluppare è **Corà Parquet Live**.

Lo studio di fattibilità è stato fatto dal *Project Manager* insieme ad un collega del reparto tecnico, ed io non ne ho potuto prendere parte.

3.3.1 Il cliente

Prima di procedere con gli aspetti tecnici è bene avere una panoramica su chi è il cliente che ha commissionato la realizzazione del progetto, quale *target* di pubblico punta a raggiungere e quali sono le sue aspettative.

Il cliente in questione è **Corà Divisione Parquet**, una divisione dell'azienda **Corà Legnami**, nata nel 1919. Corà Parquet è specializzata nella realizzazione di pavimenti in legno e realizza pavimentazioni per ambienti interni, ambienti esterni e spazi pubblici. L'azienda, *leader* nel settore, punta a raggiungere i suoi clienti attraverso canali innovativi grazie alla spinta al ringiovanimento voluta da Ettore Corà, amministratore delegato presso Corà Domenico & Figli SpA.

La richiesta del cliente era la realizzazione di un configuratore in realtà aumentata di pavimentazioni in legno, in grado di essere utilizzato inizialmente a scopo fieristico e suc-

3 RESOCONTO DELLO STAGE

cessivamente utilizzabile dai clienti finali dell'azienda in modo da avere uno strumento in grado di fare provare le varie linee di prodotto direttamente nelle case dei possibili clienti.

Uno degli obiettivi era quello di fare scaturire nel cliente ciò che viene chiamato "fattore wow", lasciando sbalordito l'utente e consolidando nella sua mente la propensione dell'azienda verso l'innovazione e il miglioramento. Il secondo obiettivo è dato dal fatto che un'applicazione di un tale livello innovativo inevitabilmente produce un passaparola dagli utenti verso chi ancora non ha visto l'applicazione, generando quindi pubblicità e maggiore visibilità all'azienda.

Per generare questo "effetto wow", Corà ha deciso di farsi realizzare un modello tridimensionale di una pin-up con un vestito di legno. Questo *avatar_{gl}* avrebbe dovuto presentare le varie linee di prodotto ed eseguire una introduzione all'applicazione. Il modello e le animazioni non sono state realizzate internamente all'azienda ma sono state delegate a terzi.

Infine, per quanto riguarda il *target* dell'app, ci si voleva rivolgere all'utente medio, in grado di poter stampare autonomamente il tag a casa propria. Visto che l'utente finale avrebbe dovuto essere la cosiddetta "signora Maria", ossia un utente senza particolare propensione per tecnicismi informatici e tecnologia in generale, l'applicazione doveva avere un elevato grado di usabilità e doveva guidare l'utente passo dopo passo.

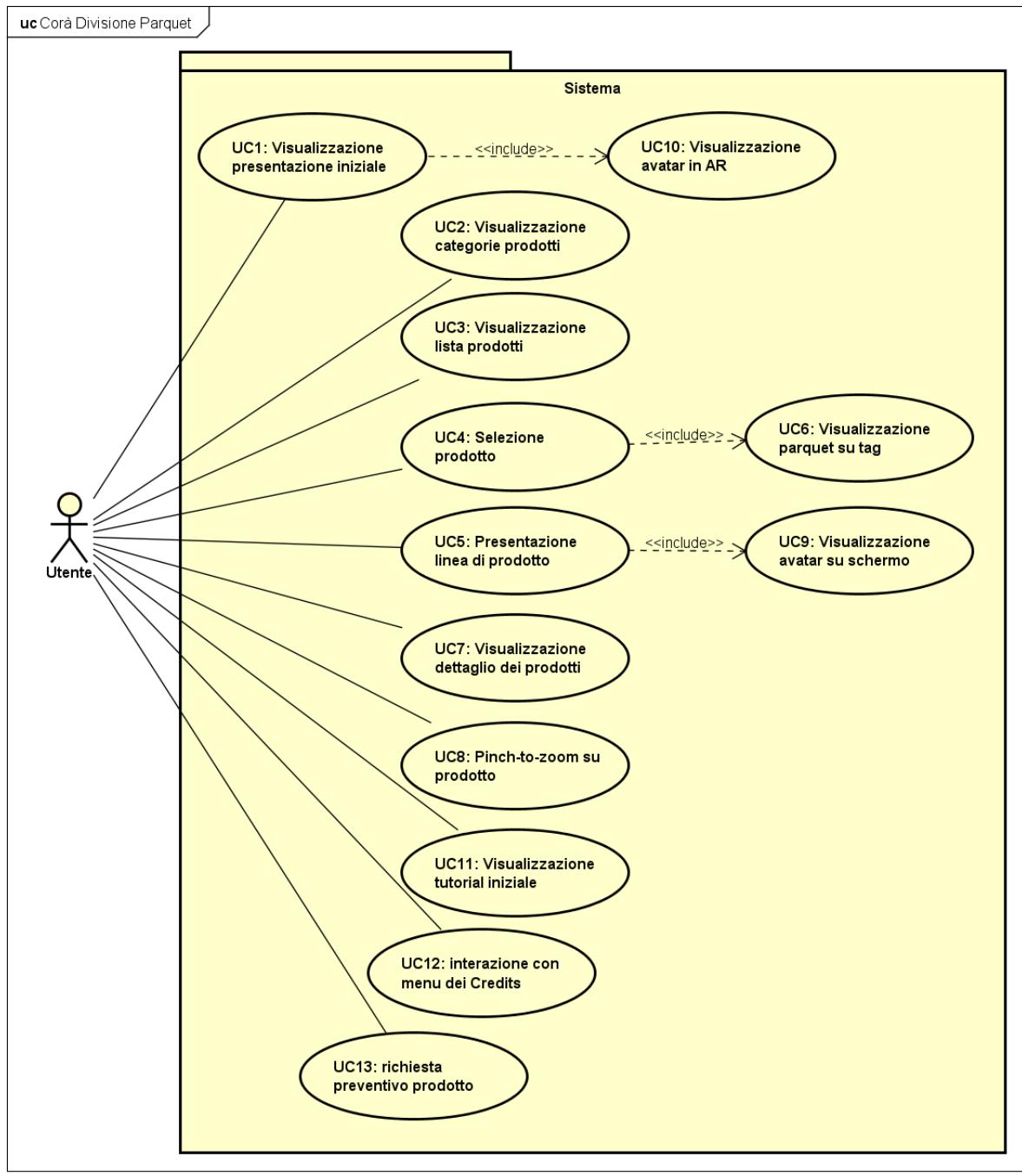
3.3.2 Analisi dei requisiti

Nella metodologia Agile, l'insieme totale dei requisiti è ottenibile esclusivamente nelle ultime iterazioni, o comunque a progetto quasi ultimato. Questo perchè l'obiettivo principale del modello Agile è quello di ottenere il massimo soddisfacimento del cliente nei tempi e nei costi preventivati.

Alcuni aspetti di rifinitura sono stati quindi concordati in iterazioni intermedie, e sono stati accettati anche se cambiavano molto la struttura del progetto, purchè comunque non si avesse un totale stravolgimento di quanto fatto fino a quel punto.

Dopo un'attenta analisi iniziale, sono stati estratti i casi d'uso principali dalle richieste del committente, da uno studio sull'utenza finale e basandosi sulla *user story*.

L'analisi dei requisiti è stata svolta insieme al *Project Manager*, il quale a mantenuto i contatti con il responsabile d'azienda esterno per tutta la durata del progetto. Insieme, abbiamo stilato una lista di funzionalità prioritarie e una lista, invece, di requisiti desiderabili e opzionali.



powered by Astah

Figura 19: Diagramma dei casi d’uso generico relativo all’applicazione Cora’ Parquet Live

Per quanto riguarda i requisiti, essi sono stati suddivisi in 3 categorie: requisiti obbligatori, requisiti opzionali e requisiti desiderabili.

Per quanto riguarda i requisiti opzionali, essi sono stati resi obbligatori dall’applicazione dei vincoli metodologici derivati dalle pratiche in uso aziendali.

Segue ora una lista descrittiva dei principali requisiti. Il presente documento vuole ga-

3 RESOCONTO DELLO STAGE

rantire un discreto livello di dettaglio senza annoiare il lettore, per cui verranno elencate solo le funzionalità principali.

Requisiti obbligatori:

- L'applicazione deve eseguire una presentazione iniziale dell'*avatar_g*, il quale deve comparire dietro un ventaglio di legno che si apre e deve fare un breve discorso di apertura.
- L'applicazione deve rendere disponibile un *tutorial* iniziale che spieghi tutte le funzionalità dell'app.
- L'applicazione deve dare la possibilità all'utente di sfogliare una lista di categorie di prodotto, suddivise alla radice in "Linea di prodotto" e "Colori", che permetteranno di accedere agli stessi prodotti in due modi diversi. La lista deve essere navigabile in entrambe le direzioni (da padre a figlio e da figlio a padre).
- Quando una linea di prodotto viene selezionata deve essere data la possibilità all'utente di avviare una breve presentazione eseguita dall'*avatar_g* sulla linea di prodotto scelta. Questo comporta l'apparizione dell'*avatar_g* a schermo.
- L'applicazione deve dare la possibilità all'utente di visualizzare i prodotti appartenenti a una certa categoria e deve dare la possibilità di selezionarne uno da visualizzare in realtà aumentata.
- L'utente deve avere la possibilità di selezionare un prodotto da una lista di prodotti e di visualizzarlo in realtà aumentata agganciato al tag. Alla selezione del prodotto deve, inoltre, essere disponibile la possibilità di visualizzare la descrizione del prodotto scelto in un pannello appositamente creato.
- L'applicazione deve permettere, quando un prodotto è selezionato e visibile in realtà aumentata, di effettuare modifiche sulla superficie coperta. In particolare, si vuole rendere disponibile all'utente la funzionalità di *pinch-to-scale* sull'oggetto parquet.
- l'applicazione rende disponibile un menù di "Credits" in cui è possibile riavviare la presentazione iniziale, rivedere il *tutorial*, oppure collegarsi alla pagina di *download* tag, o alla pagina Facebook ufficiale di Corà Divisione Parquet.
- L'applicazione deve dare la possibilità all'utente di richiedere un preventivo *online* e di scattare uno screenshot della schermata visualizzata.
- L'applicazione deve contenere almeno i primi 30 prodotti inviati dall'azienda.

Requisiti desiderabili:

- L'applicazione deve contenere i 21 prodotti inviati in seconda istanza.
- L'applicazione deve prevedere le gesture di "Swipe" per aprire e chiudere i menù.

- Le $texture_{|g|}$ dei prodotti devono utilizzare uno *shader* speculare per ottenere un'effetto più lucido.
- La barra del menù di navigazione deve essere ingrandita rispetto alla versione base del configuratore.
- Gli *screenshot* devono catturare solo la schermata principale con il prodotto in realtà aumentata, il pannello descrittivo del prodotto e il logo di Corà Divisione Parquet, niente altro.

Come già descritto in precedenza, i requisiti hanno visto modifiche anche sostanziali ad ogni iterazione, questo perchè la pratica adottata è stata un'analisi iniziale seguita da una breve progettazione e da un intenso periodo di implementazione. Al termine di tale periodo, avveniva prima una verifica completa e poi compilata una demo ed inviata al cliente. Il cliente visionava l'app e inviava di ritorno al *Project Manager* un *feedback* con le criticità e le migliorie da apportare. Seguiva, quindi un'ulteriore analisi dei requisiti ed iniziava un nuovo ciclo iterativo.

3.3.3 Progettazione

Nella metodologia Agile, la fase di progettazione è la più importante e difficile da realizzare. Serve infatti molta esperienza per lavorare con una metodologia di questo tipo, in quanto il progetto si ritrova soggetto a continui cambiamenti. Risulta quindi difficile una progettazione solida e duratura, e non si ha il tempo necessario per entrare nel dettaglio, in quanto le ore previste per questa attività sono poche e frequenti.

Come già detto in sezioni precedenti del documento, il configuratore da realizzare doveva partire da un modello di configuratore di base già realizzato e implementante un *design pattern* architetturale *Model View Controller*, per quanto Unity ne permetta un'implementazione limitata.

I *design pattern* sono strumenti che aiutano a risolvere un certo tipo di problema comune. Quindi, è necessaria la presenza di un problema per applicare un *design pattern* che lo risolva.

Bisogna sottolineare il fatto che Unity si basa pesantemente attorno alle componenti o ai più noti "GameObject", ovvero gli oggetti di gioco e non c'è modo di aggirare questa cosa.

Segue ora una descrizione di come è stato progettato il configuratore comprensivo delle funzionalità di base offerte da un configuratore già realizzato e delle parti da me costruite.

3 RESOCONTO DELLO STAGE

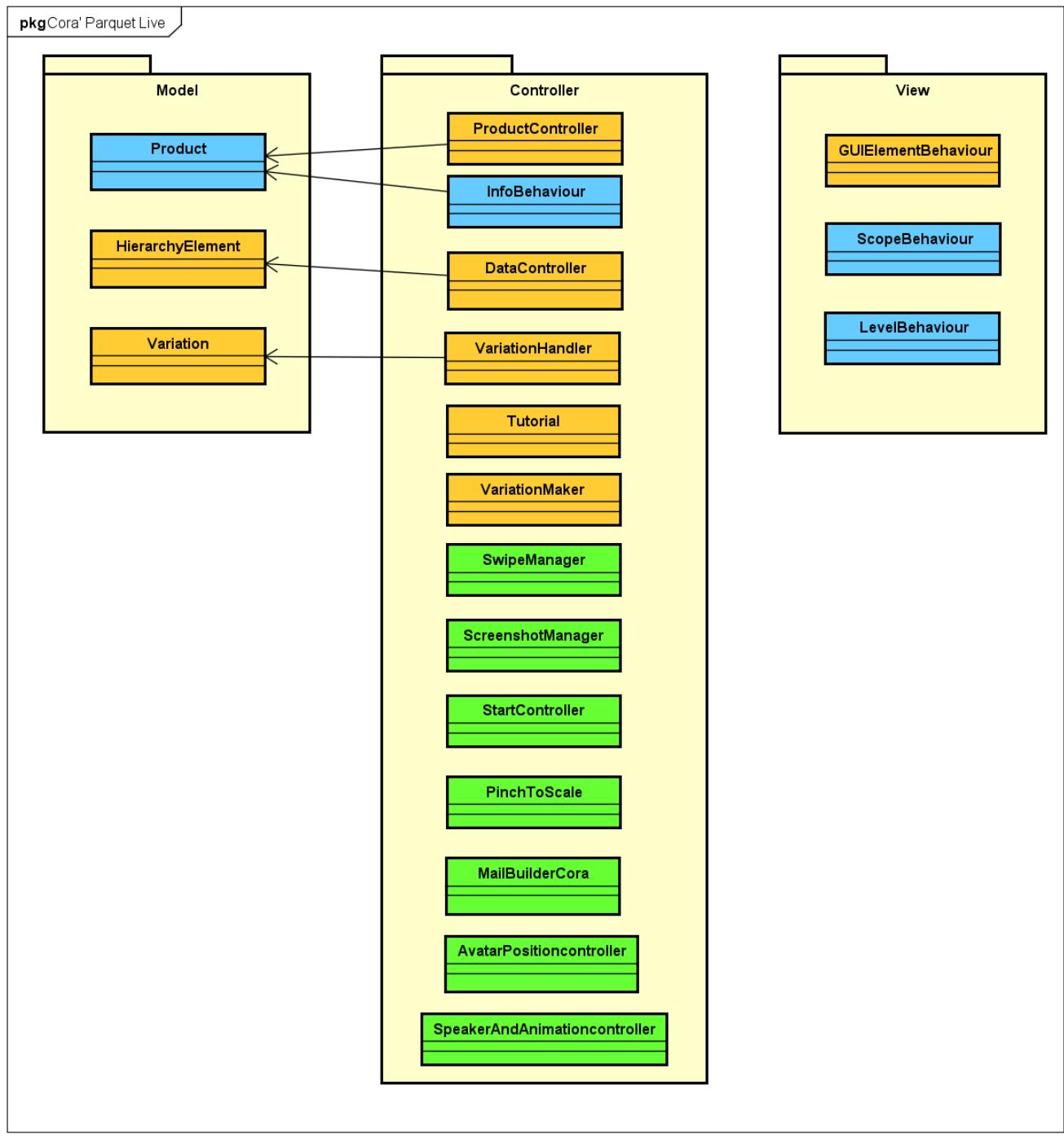


Figura 20: Diagramma delle classi generico

In figura 20 si può notare come sia stata organizzata l’architettura dell’applicazione sulla base di ciò che era già stato realizzato. In particolare, necessita una spiegazione l’etimologia dei colori del diagramma:

- **Giallo**: le classi colorate in giallo rappresentano classi già presenti e realizzate dai colleghi del reparto tecnico a cui non sono state apportate modifiche.
- **Azzurro**: Le classi colorate di azzurro rappresentano le classi già implementate a cui però è stata applicata una modifica non banale.
- **Verde**: in verde le classi create da zero.

Come è facile notare dal diagramma, il modello di gestione dei prodotti del configuratore era già implementato in una sorta di MVC. Il mio compito è stato quello di adattare il codice già sviluppato perché si prendesse carico dell’inserimento e della gestione di un nuovo tipo di contenuto.

Inoltre, il mio compito principale riguardava la gestione dell’ $avatar_{|g|}$ all’interno dell’applicazione e la gestione delle *gesture*.

3.3.3.1 Architettura generale Da subito sono state definite quelle che avrebbero dovuto essere le entità principali all’interno dell’applicazione, ovvero:

- **Product**: Classe rappresentante il modello dati di un prodotto, avente il compito di memorizzare tutte le informazioni relative a un parquet, quali: nome, linea, categoria, finitura, codice, descrizione, essenza, scelta, superficie e dimensioni. Inoltre contiene il nome del padre del prodotto e uno sprite di dimensioni 300x300(px) da usare come icona.
- **ProductController**: Questa classe si occupa di istanziare l’oggetto 3D rappresentante il prodotto vero e proprio una volta che un prodotto viene selezionato. Il modello 3D viene caricato e istanziato *runtime* dalle risorse disponibili in base al nome dell’oggetto scelto, che quindi dovrà essere univoco tra i vari prodotti e contenere la stringa ”_3d” in coda.
- **HierarchyElement**: Classe che si occupa di gestire genitore e figli del prodotto preso in considerazione.
- **DataController**: Questa è la classe centrale su cui si basa tutto il funzionamento del configuratore. Il compito di questa classe è quello di gestire la gerarchia completa di prodotti, permettendo di impostare il nome e l’icona delle varie categorie, e di impostare i genitori e le icone di ogni prodotto. Tutto questo può essere gestito come *plugin* di Unity semplicemente attaccando lo *script* ad un *GameObject* vuoto.
- **AvatarPositionController**: Classe principale di gestione e controllo dell’ $avatar_{|g|}$. Qui viene gestito il posizionamento dell’ $avatar_{|g|}$: all’avvio dell’applicazione l’ $avatar_{|g|}$ viene posizionato sopra il tag ad una grandezza fissata, successivamente, una volta terminata la presentazione iniziale, l’ $avatar_{|g|}$ viene spostato e fissato a schermo in una posizione laterale e non troppo ingombrante. Quando l’ $avatar_{|g|}$ è fissato a

schermo ha una scala posta a zero, mentre viene ingrandito una volta richiamato tramite apposito pulsante. Questo *script* si occupa anche di gestire la posizione e l'apparizione del ventaglio iniziale per poi invocare il metodo dedicato alla presentazione vera e propria dell' $avatar_{|g|}$.

- **StartController:** È la classe che si occupa di gestire la presentazione iniziale dell' $avatar_{|g|}$. Qui i file audio vengono caricati e temporizzati con l'inizio delle animazioni. La classe si occupa di gestire sia l'avvio sia il termine (anche richiesto dall'utente) dell'applicazione.
- **SpeakerAndAnimationController:** Classe realizzata con lo scopo di gestire l' $avatar_{|g|}$ in modo da fargli presentare le categorie tramite animazioni e file audio. Non era conosciuto a priori il numero di animazioni che si sarebbero dovute implementare per cui questa classe doveva essere progettata in modo da gestire un numero indefinito di animazioni e file audio.
- **PinchToScale:** Questa classe non si occupa di riconoscere la *gesture*, compito dedicato a una classe sviluppata da terzi, ma si occupa di definire il comportamento quando viene rilevata una particolare *gesture* sul modello 3D del prodotto.
- **SwipeManager:** Questa classe, come la precedente, non è un riconoscitore di *gesture* ma un gestore che si occupa di definirne il comportamento. Lo SwipeManager deve aprire e chiudere i menù in base al tipo di *swipe* effettuato.
- **ScreenshotManager:** Si occupa di gestire la cattura e il salvataggio della schermata visualizzata sul *display*. Lo *script* si affida all'utilizzo di classi dedicate sviluppate da terzi e ha il compito di gestire cosa deve essere visualizzato nello *screenshot*.
- **CoraMailBuilder:** È lo *script* che si occupa della creazione della e-mail di richiesta preventivo su un determinato prodotto.
- **LevelBehaviour:** Questa classe si occupa di popolare il menù inferiore dell'applicazione, creando i pulsanti per ogni categoria e i pulsanti relativi ai prodotti.

Questi sono gli *script* principali utilizzati nella realizzazione del progetto. Come già detto, sono stati creati altri *script* più dedicati e di rifinitura, la cui spiegazione in dettaglio non farà parte di questo documento per non appesentirne la lettura.

3.3.3.2 Layout e GUI_{|g|} Per quanto riguarda la GUI_{|g|} dell'applicazione, essa è stata progettata da terzi per conto di Corà Divisione Parquet e mi è stata inviata tramite un file PSD_{|g|} contenente le grafiche da utilizzare e il *layout* a cui adeguarsi.

Il *layout* è stato scelto sulla base dei configuratori precedentemente utilizzati ed è stato adattato per il progetto specifico.



Figura 21: *Layout con menu' info aperto*



Figura 22: *Layout con menu inferiore aperto*

3 RESOCONTO DELLO STAGE

Di seguito vengono descritti i componenti principali della GUI_{|g|}.

- **Logo:** il logo è un pulsante cliccabile che apre il menù dei *Credits* nel centro dello schermo;
- **Menù inferiore:** contiene la lista delle categorie, con eventualmente la relativa icona, e la lista dei prodotti come figli della propria categoria madre. Il menù è composto da una *breadcrumb* navigabile, e da una sequenza di bottoni assegnati ognuno ad una categoria, e nei livelli più profondi ai prodotti.
- **Pannello Info:** è un pannello laterale che compare dopo la pressione del pulsante "i". Il pulsante in questione diventa visibile solo quando un prodotto è selezionato e permette di aprire la sezione relativa alla descrizione del prodotto. Il pannello si compone anche di un pulsante per l'acquisizione di *screenshot* e un pulsante per la richiesta di preventivo del prodotto visualizzato.
- **Pulsante animazione:** è un pulsante a forma di fumetto situato nella parte a est dello schermo, e diventa visibile ogni volta che l'*avatar_{|g|}* ha qualcosa da dire. Nello specifico, compare quando viene selezionata una categoria interna alle "linee di prodotto", e scompare negli altri casi. La comparsa è seguita da un'animazione di notifica per richiamare l'attenzione dell'utente. Una volta premuto il pulsante, compare l'*avatar_{|g|}* nella parte ovest dello schermo iniziando l'animazione dedicata.
- **Scope:** mirino posizionato al centro dello schermo e utilizzato per fornire brevi istruzioni all'utente su cosa fare e dare la possibilità all'utente di scaricare il tag, oltre al fatto di eseguire la sua funzionalità principale di supporto per mirare il tag.

L'apertura del menù, dei credits e del pannello laterale non è mutualmente esclusiva, in quanto tutte e tre le componenti possono essere visibili in stato aperto contemporaneamente senza interferire tra di loro.

In questo capitolo ho avuto voce, in quanto ho potuto esprimere il mio dissenso su alcune scelte relative all'usabilità dell'applicazione mantenendo un certo tipo di *layout*. Queste osservazioni e richieste di modifica della grafica sono state concordate prima con *Project Manager* e successivamente con il cliente. Le modifiche osservate e accolte sono state:

- Grandezza del pannello delle informazioni laterali. Essendo troppo piccolo risultava inutilizzabile da *smartphone*.
- Stile dell'icona dell'applicazione che risultava poco leggibile.
- Posizione del pulsante animazione. Inizialmente posto nella parte superiore dello schermo, risultava difficile da raggiungere con un'impugnatura *standard* del *device*.

Inoltre, ho proposto l'utilizzo di un tag di grandezza A3, al posto del foglio A4 utilizzato per ottenere un migliore risultato di agganciamento del tag e di *tracking*. Questa richiesta non è stata accolta in quanto si presuppone la stampa domestica del tag in assenza di possibilità di stampare fogli A3.

3.3.3.3 Animator Controller Gli *Animator Controller* sono i gestori delle animazioni di un *GameObject* in Unity. Generalmente, ogni oggetto che si vuole animare ha attaccato un *Animator* con associato un *Animator Controller* che ne definisce il comportamento. In questo progetto, tali componenti hanno avuto un ruolo fondamentale, per cui è stata necessaria un'attenta progettazione di queste componenti.

In questo paragrafo verranno tralasciate le spiegazioni degli *Animation Controller* banali.

- **Info Panel Controller:** controller dedicato al pannello laterale per gestire le animazioni entrata, uscita e di oscurazione del pulsante "i". Esempio utile per capire come ogni oggetto della $\text{GUI}_{|g|}$ è stato gestito in modo simile. Ogni animazione viene fatta partire al cambio di stato di un valore booleano oppure all'azionamento di un particolare *trigger*.

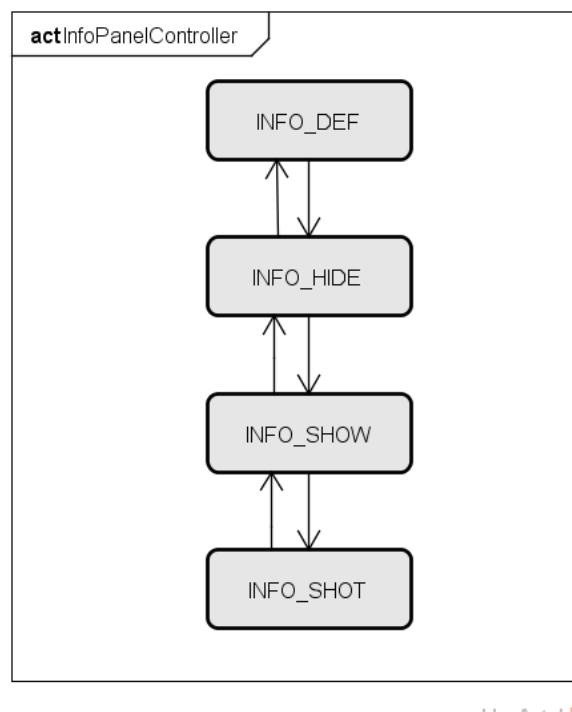


Figura 23: *Animator Controller* del pannello Info laterale

Nel dettaglio: `INFO_DEF` nasconde sia pannello che pulsante, `INFO_HIDE` mostra il pulsante "i", `INFO_SHOW` apre il pannello su schermo, `INFO_SHOT`, rimuove gli elementi della $\text{GUI}_{|g|}$ indesiderati per l'esecuzione di uno *screenshot*.

- **Avatar Speaker Controller:** dedicato alla gestione della presentazione iniziale da parte dell' $\text{avatar}_{|g|}$ e alle animazioni relative alla spiegazione delle categorie.

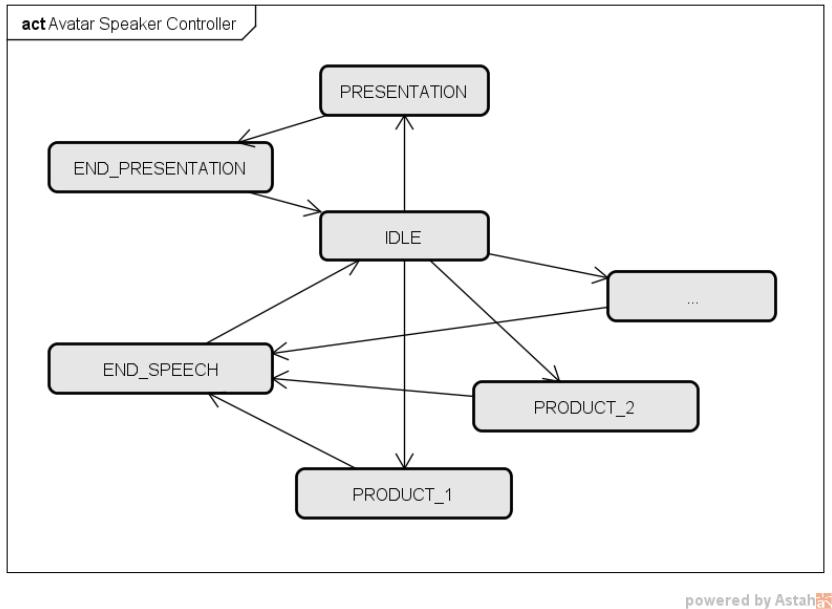


Figura 24: *Animator Controller* dell’ avatar_{g}

Nel dettaglio: IDLE è lo stato iniziale, a cui l’ avatar_{g} ritorna alla fine di ogni presentazione. Lo stato PRESENTATION è un particolare stato in cui l’*animator* viene portato nel momento in cui si vuole avviare la presentazione iniziale. Questo stato termina poi con il passaggio a END_PRESENTATION che è uno stato necessario per effettuare alcuni controlli da codice, come controllare che la presentazione sia terminata. Lo stesso viene fatto con le n categorie di prodotti di cui si ha la presentazione.

Questo *animator* poteva essere implementato in modo più efficiente caricando l’animazione da fare partire di volta in volta in base ad una stringa passata. Avendo tempo limitato ho preferito procedere per una strada di facile implementazione come questa.

3.3.4 Implementazione

L’attività di implementazione e di codifica si è svolta secondo quanto previsto dalla metodologia Agile. Le fasi di codifica non sono avvenute in un unico *round*, ma in brevi e frequenti blocchi di tempo. Per prima cosa, sono state implementate le parti principali del sistema, cioè le parti che quasi sicuramente non avrebbero avuto modifiche nelle successive iterazioni.

Ho proceduto, in primo luogo, con l’adattamento di quanto disponibile nel configuratore base. In particolare ho creato una classe *Product* in grado di contenere tutte le informazioni necessarie per ogni prodotto.

3.3.4.1 Pinch-To-Scale È stata richiesta la possibilità di espandere il parquet mostrato in realtà aumentata direttamente *runtime* tramite l’applicazione di una *gesture*

dedicata.

Bisogna premettere che le $texture_{|g|}$ dei parquet sono state recapitate simulandone la grandezza reale per cui, data anche l’irregolarità di alcuni parquet, era difficile fare il *tiling* (ripetizione della $texture_{|g|}$ su uno o più assi) della $texture_{|g|}$.

Su suggerimento del CTO ho deciso di procedere mantenedo la dimensione reale della $texture_{|g|}$, senza *tiling*, e applicando una maschera ridimensionabile all’oggetto in modo tale da visualizzarne solo una parte. La realizzazione è stata affidata completamente a me.

Ho creato una maschera a forma di cornice, unendo più oggetti tridimensionali prefabbricati. A tale maschera ho, poi, applicato un particolare *shader* in grado di rimuovere il *rendering* delle parti di $texture_{|g|}$ coperte dalla maschera.

Di seguito è illustrata la codifica relativa allo *script* *PinchToScale*.

1. PinchToScale.cs

```

2  public class PinchToScale : MonoBehaviour
3  {
4      public Vector3 scaleWeights = Vector3.one;
5      public float minScaleAmount = 0.4f;
6      public float maxScaleAmount = 4.0f;
7      public float sensitivity = 0.3f; // augment for a bigger scaling
8      public float smoothingSpeed = 12.0f; // set to 0 to disable smoothing
9      public GameObject mask;
10
11     private float idealScaleAmount = 0.4f;
12     private float scaleAmount = 0.4f;
13     private Vector3 baseScale = Vector3.one;
14
15     public float ScaleAmount
16     {
17         get { return scaleAmount; }
18
19         set
20         {
21             value = Mathf.Clamp( value, minScaleAmount, maxScaleAmount ); // check
22             if the value is between limits
23
24             if( value != scaleAmount )
25             {
26                 scaleAmount = value;
27
28                 Vector3 s = scaleAmount * baseScale;
29                 s.x *= scaleWeights.x;
30                 s.y *= scaleWeights.y;
31                 s.z *= scaleWeights.z;
32
33                 mask.transform.localScale = s; // resize the mask scale
34             }
35         }
36     }
37
38     public float IdealScaleAmount
39     {

```

3 RESOCONTO DELLO STAGE

```
40     get { return idealScaleAmount; }
41     set { idealScaleAmount = Mathf.Clamp( value , minScaleAmount ,
42                                         maxScaleAmount ); }
43   }
44
45   void Start()
46   {
47     baseScale = transform.localScale;
48     IdealScaleAmount = ScaleAmount;
49   }
50
51   void Update()
52   {
53     if (smoothingSpeed > 0) {
54       ScaleAmount = Mathf.Lerp (ScaleAmount , IdealScaleAmount , Time.deltaTime
55                               * smoothingSpeed);
56     } else {
57       ScaleAmount = IdealScaleAmount;
58     }
59   }
60
61   void OnPinch( PinchGesture gesture )
62   {
63     IdealScaleAmount += sensitivity * gesture.Delta.Centimeters();
64   }
```

Il codice, all'avvio dello *script*, memorizza la scala locale dell'oggetto, e imposta il valore di *scaling* ideale allo *scaling* di base. Ad ogni *frame* viene aggiornato, internamente al metodo *Update()*, lo *scaling* attuale, sia nel caso sia presente un effetto di *smooth* sia altrimenti. Quando viene catturata una *gesture* di *Pinch*, ne viene presa l'ampiezza in centimetri e moltiplicata per la sensibilità che gli si vuole dare allo *scaling* rispetto alla *gesture*. Questo valore calcolato viene memorizzato in *IdealScaleAmount* che nel successivo *Update* sarà memorizzato come nuovo valore di *ScaleAmount*. *ScaleAmount* ogni volta che viene modificato si occupa di applicare lo *scaling* impostato alla maschera passata come parametro da Unity.

3.3.4.2 Inserimento prodotti Successivamente ho proceduto con l'inserimento dei primi 20 prodotti, in quanto il modello e la gestione delle categorie e dei prodotti era già implementata. Per ogni prodotto, prima, ho dovuto importare negli *asset_g* la relativa *texture_g* e icona, poi, dopo aver attaccato uno *script* *Product* al prodotto comprensivo del modello 3D, questo veniva processato in uno script (*VariationMaker.cs*) che creava 2 oggetti: un *GameObject* contenente le informazioni del prodotto da utilizzare nello *script* di gestione dei dati (*DataController.cs*) e un *GameObject* con il modello 3D.

3.3.4.3 Gestione dell'*avatar_g* Come già accennato, il grosso del lavoro consisteva nell'implementazione di un *avatar_g* all'interno dell'applicazione. Per prima cosa ho

gestito la presentazione iniziale composta da 4 file di animazione e 4 file audio. È stata usata la tecnica del *lip sync* per ottenere un effetto realistico. Di ciò se ne è occupato lo *StartController* sviluppato appositamente.

Lo *StartController* è stato implementato utilizzando il *design pattern Singleton* in quanto era necessario avere al più un'istanza della classe.



Figura 25: Presentazione iniziale

All'avvio dello *StartController* vengono caricati in memoria gli audio file relativi alla presentazione. Durante ogni ciclo di *Update*, invece, viene controllato lo stato dell'animazione corrente e il numero contenuto all'interno di un particolare contatore. In questo modo è possibile eseguire nell'ordine corretto i vari pezzi ottenendo, così, l'animazione completa, senza che venga ciclata più volte essendo che il contatore procede verso un'unica direzione incrementale e viene ripristinato solo quando si vuole riavviare la presentazione.

Di seguito viene mostrata la codifica del metodo *Awake()* che si occupa del caricamento dei file audio.

2. StartController.cs - Metodo Awake()

```

1 void Awake() {
2     _instance = this;
3     pt1 = (AudioClip)Resources.Load ("Speakeraggio/01_Presentazione1",
4         typeof( AudioClip));

```

3 RESOCONTO DELLO STAGE

```
5     pt2 = (AudioClip)Resources.Load ("Speakeraggio/02_Presentazione2",  
6     typeof( AudioClip));  
7     pt3 = (AudioClip)Resources.Load ("Speakeraggio/03_Presentazione3",  
8     typeof( AudioClip));  
9     pt4 = (AudioClip)Resources.Load ("Speakeraggio/04_Presentazione4",  
10    typeof( AudioClip));  
11    }  
12 }
```

E il metodo *Update()* che si occupa della gestione delle animazioni e della sincronizzazione con i file audio. Inoltre, si può notare come venga reso disponibile un pulsante "Skip" per saltare la presentazione.

3. StartController.cs - Metodo Update()

```
1 void Update(){  
2     if (avatar\glossController.GetCurrentAnimatorStateInfo (0).IsName ("01  
3     _Presentazione1") && startLock == 1) {  
4         presentationAudio.clip = pt1;  
5         presentationAudio.Play();  
6         skipButton.SetActive(true);  
7         skipButton.GetComponent<Animator> ().SetBool ("Show", true);  
8         startLock++;  
9     }  
10    if (avatar\glossController.GetCurrentAnimatorStateInfo (0).IsName ("02  
11    _Presentazione2") && startLock == 2) {  
12        presentationAudio.clip = pt2;  
13        presentationAudio.Play();  
14        startLock++;  
15    }  
16    if (avatar\glossController.GetCurrentAnimatorStateInfo (0).IsName ("03  
17    _Presentazione3") && startLock == 3) {  
18        presentationAudio.clip = pt3;  
19        presentationAudio.Play();  
20        startLock++;  
21    }  
22    if (avatar\glossController.GetCurrentAnimatorStateInfo (0).IsName ("04  
23    _Presentazione4") && startLock == 4) {  
24        presentationAudio.clip = pt4;  
25        presentationAudio.Play();  
26        startLock++;  
27    }  
28    if (avatar\glossController.GetCurrentAnimatorStateInfo (0).IsName ("  
29    EndPresentation") && startLock == 5) {  
30        EndPresentation ();  
31        startLock = 1;  
32    }  
33 }
```

Il passo di implementazione successivo è stato la gestione della posizione dell'*avatar_g*. All'inizio doveva risultare agganciato al tag per poi spostarsi e rimanere agganciato alla

parte ovest dello schermo, comparendo solo su richiesta.

Questo *script* è quello che più ha usufruito della metodologia Agile, e in particolare quello che ha subito più incrementi nelle varie iterazioni.

Questo perchè il cliente, non essendo un esperto di realtà aumentata e, soprattutto di applicazioni Android, non aveva un'idea precisa di come e dove dovesse apparire l'*avatar_g*. Per cui sono state richieste numerose prove per ottenere il risultato che più rendesse felice il cliente.

In iterazioni intermedie avevo previsto un'interessante funzionalità per cui il *device* calcolava l'altezza rispetto al tag e modificava la grandezza dell'*avatar_g* per una visione "da alzati" e una "da seduti". Questa funzionalità da me realizzata è piaciuta molto al *team* di sviluppo ma è stata bocciata dal cliente finale che, alla fine, è stato convinto dal *Project Manager* ad adottare la soluzione ottimale consistente nell'*avatar_g* di grandezza fissa.

Questo *script* è fondamentale perchè gestisce le animazioni dei vari componenti che agiscono nella presentazione iniziale e le sincronizza per ottenere il miglior effetto visivo. Anche per l'implementazione di questo *script* è stato utilizzato un *design pattern* Singleton per permettere al più una sola istanza disponibile.

L'ultimo *script* realizzato per il controllo dell'*avatar_g* è lo *script* che si occupa di gestire le presentazioni delle varie categorie, chiamato *SpeakerAndAnimationController*. Anche in questo *script* è stato usato il *design pattern* Singleton.

Nel dettaglio, il pulsante di una categoria invoca, alla pressione, un metodo chiamato *SetPresentation(string)* che si occupa di caricare in memoria l'audio corrispondente effettuando una ricerca per nome negli *asset_g*. Nel caso la ricerca termini correttamente viene mostrato il pulsante animazione (pulsante a forma di fumetto), altrimenti viene nascosto, se già visibile, o non mostrato, se già invisibile.

Alla pressione del pulsante animazione viene invocato un metodo dedicato allo start e allo stop della presentazione. In particolare, se esiste già un'audio e un'animazione in esecuzione da parte dell'*avatar_g*, e viene premuto tale pulsante, l'animazione viene fermata. Se invece non è presente alcuna presentazione in esecuzione, viene fatto partire l'audio in memoria con la relativa animazione, facendo comparire l'*avatar_g* con un effetto di *scaling out*.

È utile osservare il funzionamento del metodo *Update()* per capire come avviene la sincronizzazione tra audio e animazione.

4. SpeakerAndAnimationController.cs - Metodo Update()

```

1 void Update() {
2     if (avatar\glossController.GetCurrentAnimatorStateInfo (0).IsName (""
3         EndSpeech")) {
4         StartCoroutine(scaleIn( new Vector3 (100f, 100f, 100f), new Vector3(0f
5             ,0f,0f) , 1.5f));
6         hasAudioBeenPlayed = false;
7         instance.resetAnimation (active);
8     }
9     if (avatar\glossController.GetCurrentAnimatorStateInfo (0).IsName ("Idle"
10 ) && active != "") {
11         AudioClip audio = (AudioClip)Resources.Load ("Speakeraggio/" + active ,

```

3 RESOCONTO DELLO STAGE

```
9     typeof(AudioClip));
10    activeAudio.clip = audio;
11  }
12  if (!hasAudioBeenPlayed && !string.IsNullOrEmpty(active) && avatar \
13      glossController.GetCurrentAnimatorStateInfo(0).IsName(active)) {
14      hasAudioBeenPlayed = true;
15      activeAudio.Play();
}
```

Come si può notare, viene fatto un controllo sullo stato in esecuzione dell'*animator controller*. In particolare, se ci si trova in uno stato di fine presentazione, viene data la possibilità di far partire un nuovo audio e viene nascosto l' $avatar_{|g|}$. Se l' $avatar_{|g|}$ si trova nello stato iniziale, viene caricato l'audio corrente, mentre se l'animazione relativa all'audio è attiva e l'audio può essere eseguito viene effettuato un play, che avviene in modo sincronizzato con l'animazione.

3.3.4.4 Gestione della GUI_{|g|} La GUI_{|g|} è l'aspetto del progetto che più ha subito modifiche. Inizialmente, per trovare la resa grafica migliore a livello estetico e, successivamente, per garantire il più alto livello possibile di usabilità. Il problema principale è stato identificare il contesto di utilizzo dell'applicazione e decidere se ottimizzare l'interfaccia per *smartphone* o per *tablet*, in quanto la creazione di due interfacce ad-hoc avrebbe avuto un costo aggiuntivo per il cliente. Si è deciso di ottimizzare l'app per *smartphone*, in quanto tecnologia più reperibile. Lo sviluppo della GUI_{|g|} è avvenuto seguendo come linea guida un file PSD_{|g|} contenente le varie schermate e da cui era possibile prendere le grafiche (icone, pulsanti, etc.)

3.3.4.5 Implementazione del tutorial Come ultimo passo di implementazione, ho proceduto con la creazione del tutorial, prendendo le grafiche da una sezione del file PSD_{|g|} dedicato e modificando del codice già creato per configuratori precedentemente realizzati. Il *tutorial* è stato pensato in modo tale da procedere con l'avanzamento grazie ad un *tap* sullo schermo ed è stata prevista una funzionalità per saltare il *tutorial* ed andare direttamente alla parte del configuratore.

3.3.5 Verifica e validazione

L'attività di verifica è stata svolta durante tutto il periodo di sviluppo del progetto e, in particolare, al termine di ogni iterazione, secondo quanto previsto dall'implementazione della metodologia Agile.

Come già detto, nella metodologia Agile, le risorse dedicate alla scrittura di documentazione e alla attività di progettazione sono poche. Per cui, anche le attività di verifica sono state svolte relativamente al codice, tralasciando gli aspetti secondari. Ciò significa che durante la fase di codifica ho eseguito, inizialmente, una breve ma mirata analisi statica composta da:

- **Analisi di flusso di controllo:** ho controllato che il codice scritto fosse eseguito nella sequenza specificata e mi sono accertato del fatto che non ci fosse codice non raggiungibile, oppure segmenti di esecuzione non terminanti.
- **Analisi di flusso dei dati:** ho controllato che tutte le variabili fossero inizializzate correttamente e che non ci fossero variabili prive di valore.
- **Analisi di limite:** ho verificato che tutti i dati da me usati restassero entro i limiti del loro tipo e della precisione desiderata.

Altri tipi di analisi statica sarebbero stati difficili da implementare, in quanto, essendo Unity un motore di gioco risulta molto più semplice ed economico, dal punto di vista delle risorse utilizzate, effettuare un'analisi dinamica del codice.

Ho proceduto quindi, in secondo luogo, con l'applicazione di tecniche di analisi dinamica, al termine di ogni significativa attività di codifica producente materiale testabile. Con l'aiuto di un *plugin* di Vuforia per Unity, è stato possibile utilizzare una webcam per simulare il comportamento della fotocamera del *device* ed evitare, quindi, lunghe attese dovute alla compilazione dell'applicazione.

Per l'analisi dinamica mi sono servito, per prima cosa, dell'utilizzo di *log*, in modo da verificare il corretto stato di ogni variabile e la corretta esecuzione dei metodi. Questo, quando possibile è stato verificato tramite l'opzione "Play" di Unity che permette l'esecuzione dell'applicazione direttamente sul PC. Per testare moduli richiedenti particolari funzionalità presenti esclusivamente su *device*, invece, ho proceduto sempre con l'utilizzo di *log*, ma compilando l'applicazione ed eseguendola direttamente su *device*.

Grazie all'utilizzo di Cygwin ho potuto visualizzare i *log* di sistema di Android e i *log* da me creati. Questa metodologia è stata utilizzata principalmente per la verifica della corretta implementazione delle *gesture*, quali il *Pinch-To-Scale* e lo *Swipe*.

Cygwin è una distribuzione di *software* libero che consente a diverse versioni di Microsoft Windows di svolgere alcuni compiti esteticamente e funzionalmente simile ad un sistema Linux.

Alla fine di ogni iterazione, infine, alcuni componenti del *team* tecnico si occupavano di verificare e validare l'applicazione per confermare il soddisfacimento dei requisiti richiesti per tale revisione. Questi *test* venivano svolti tramite un'esecuzione dell'applicazione su diversi tipi di device andando a testare il corretto funzionamento delle componenti implementate.

I *device* utilizzati per il *testing* dell'applicazione sono stati:

- Google Nexus 5 (Android);
- Samsung Note 4 (Android);
- Samsung Galaxy S3 (Android);

Una volta arrivati ad un buon livello di completamento del progetto abbiamo proceduto, io e la squadra tecnica, con una validazione interna dell'applicazione e, vista l'assenza di *bug*, con un rilascio in beta al cliente.

Il cliente ha effettuato i propri *test* fornendoci *feedback* e le ultime modifiche richieste per permettendoci di mandare in *release* dell'app sui vari *store* verso la metà del mese di Settembre.

3.4 Livello di completezza raggiunto

In definitiva, è stata realizzata un'applicazione configuratore in realtà aumentata dedicata alla visualizzazione di parquet in legno in un contesto domestico. L'applicazione si compone di un menù inferiore dedicato alla ricerca e selezione dei prodotti in base alla loro categoria o in base al loro colore. Un prodotto selezionato viene reso visibile agganciato al tag e viene reso disponibile un menù laterale in cui è possibile leggere le informazioni del prodotto, scattare *screenshot* e richiedere un preventivo relativo al parquet selezionato.

Il menù inferiore risulta composto da due sezioni:

- **Breadcrumb:** posizionata sulla parte superiore del menù, rappresenta il percorso seguito all'interno della gerarchia delle categorie. Ogni *step* della *breadcrumb* è un pulsante cliccabile che permette il ritorno al livello selezionato.
- **Slider:** la parte inferiore è una lista orizzontale di categorie o prodotti, in base al livello della gerarchia in cui ci si trova. Questa lista è composta da pulsanti cliccabili aventi un nome significativo e, per tutti i prodotti e alcune categorie, anche un'icona rappresentativa.

Il pannello laterale, che compare solo alla selezione di un prodotto, è composto invece da:

- **Icona:** posta nella parte superiore, consiste in una rappresentazione grafica del parquet selezionato;
- **Descrizione:** seguono una lista di informazioni riguardanti: nome, categoria, linea, essenza, scelta e dimensioni;
- **Pulsanti utili:** in fondo al pannello sono presenti due pulsanti: uno per la cattura di uno *screenshot* e uno per la richiesta di preventivo via e-mail;

Al centro dello schermo è presente un mirino che scompare quando si inquadra il tag, il cui scopo è quello di guidare l'utente con alcuni *tips* e di aiutare ad inquadrare correttamente il tag. È presente anche un pulsante che permette il *download* del tag, posto a nord-est del mirino.

Premendo il logo, posto nella parte superiore dello schermo, si accede al menù dei *Credits*. Questo menù, oltre ad alcune informazioni riguardanti lo sviluppo dell'applicazione, rende disponibili alcuni pulsanti utili:

- un pulsante che serve a rivedere la presentazione iniziale;

- un pulsante per avviare il *tutorial* iniziale;
- un pulsante per scaricare il tag;
- un pulsante dedicato all'apertura della pagina Facebook di Cora' Divisione Parquet.

L'applicazione si completa con una presentazione iniziale da parte dell'*avatar_{g1}* (il cui nome è Arte) e con la possibilità di richiamarlo su schermo ogni volta che ha qualcosa da dire tramite un apposito pulsante posto nella parte laterale dello schermo.

Alla fine del tempo di stage si può stimare un completamento del progetto quasi totalitario, in quanto l'ultima richiesta da parte del cliente prima della pubblicazione dell'app è stata quella di implementare il *follow* di occhi e testa dell'*avatar_{g1}* rispetto alla telecamera. Questo ultimo aspetto è stato sviluppato interamente da un collega del reparto tecnico dopo la conclusione del mio stage.

L'app è stata pubblicata sullo store Android in data 18 Settembre ed è stata fatta la richiesta di pubblicazione sullo store di Apple, la quale è stata accolta rapidamente.

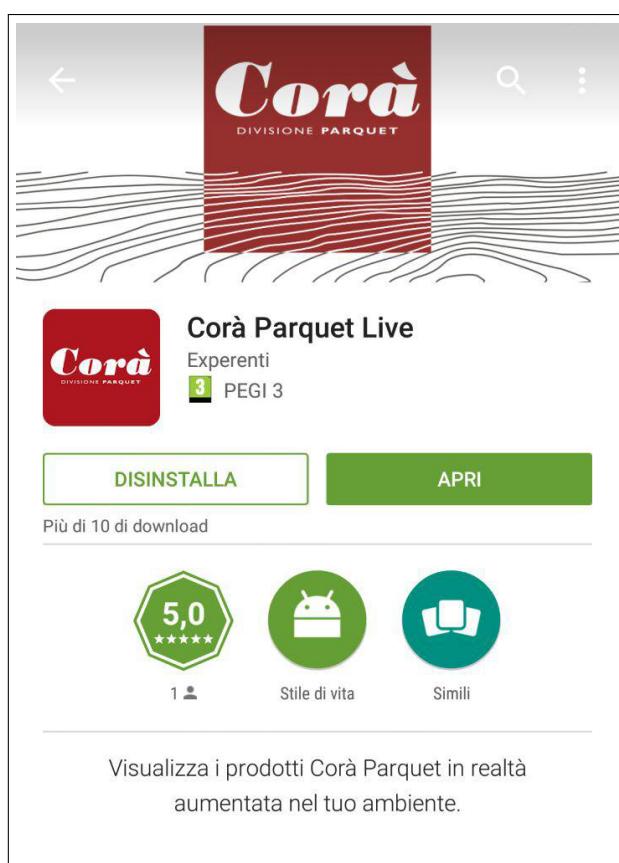


Figura 26: Applicazione visualizzata all'interno del Play Store

4 Valutazione Retrospettiva

Segue ora una valutazione personale sul periodo di stage trascorso.

4.1 Soddisfacimento obiettivi

Lo stage si è svolto nel completo rispetto di tutti i vincoli imposti, concludendosi in data 11/09/2015.

Per quanto riguarda gli obiettivi dello stage, essi sono variati durante lo svolgimento in quanto non era garantita la disponibilità di seguire un progetto commissionato dall'esterno.

Come obiettivo massimo era stato richiesto di sviluppare un'intera app visore di AR, completa di tutti i suoi contenuti semplici e complessi e della propria grafica, dalla fase di accettazione dei materiali in entrata fino alla fase di consegna della beta finale al cliente. Questo obiettivo è stato pienamente raggiunto, anzi si è quasi arrivati al release dell'applicazione sugli store, avvenuto pochi giorni dopo il termine del mio stage.

Tutte le funzionalità richieste sono state implementate e l'interfaccia consegnata è risultata completamente funzionante, questo ha permesso di ottenere un giudizio positivo da parte del tutor e del committente.

È stata soddisfatta la totalità dei requisiti obbligatori e la quasi totalità di quelli desiderabili. Non ho potuto seguire alcune richieste da parte del cliente che sono giunte gli ultimi giorni di stage.

Ho rispettato tutti i vincoli, sia tecnologici, sia quelli di metodo che temporali. Va evidenziato, però, che il numero di ore preventivate per ogni attività risulta diverso da quello stabilito nel piano di progetto.

Sezione	Descrizione	Ore di lavoro
1.1	Formazione su ambienti di sviluppo	32
1.2	Formazione su librerie utilizzate	20
1.3	Formazione sull'app Experenti	8
2	Realizzazione di un contenuto di realtà aumentata a tema libero	56
3	Analisi su progetti già realizzati internamente e formazione su flusso di lavoro interno	32
4	Realizzazione di un'app demo completa	12
5	Inserimento nel team di sviluppo e realizzazione di un'app nella sua interezza	160
TOTALE		320

Tabella 2: Tabella retrospettiva relativa alle ore dedicate per ciascuna attività

Come si evince dalla tabella, le ore effettivamente impiegate per la parte di formazione sono inferiori rispetto a quelle preventivate. Questo perchè su alcuni argomenti trattati, quelli cioè riguardanti Android, ero già abbastanza preparato, in quanto è stata una tecnologia ampiamente utilizzata durante il progetto didattico di Ingegneria del *Software*. Inoltre, non ho avuto particolari problemi di apprendimento dai *tutorial* seguiti, guadagnando qualche ora per ogni attività svolta in ambito formativo.

Il tempo guadagnato mi ha permesso di dedicare più tempo alla realizzazione del progetto, fornendo un grado di completezza maggiore rispetto a quanto preventivato. Infatti, dagli obiettivi, si vede che era richiesta una demo, mentre la demo è stata consegnata una settimana prima della fine dello stage, permettendomi di lavorare su alcune delle ultime richieste del cliente prima della *release*.

4.2 Conoscenze acquisite

Le conoscenze acquisite durante lo stage sono di due caratteri: tecnologiche e metodologiche.

A livello tecnologico ho potuto arricchirmi di strumenti e tecnologie innovative e all'avanguardia che sicuramente saranno sempre più richieste a livello curricolare con la prossima uscita di device portatili di ultima generazione e visori di AR/VR. In particolare:

- **Unity:** Strumento che sin da subito ha saputo mostrare il suo grandissimo potenziale nella realizzazione di applicazioni *cross-platform*. Dall'inizio dello stage sono

state utilizzate versioni di Unity dalla 4.6.0 alla 5.1.2, mostrandomi l'evoluzione che questo strumento ha subito per agevolare sempre di più lo sviluppo di applicazioni. Le principali nozioni apprese sono state:

- **Fisica**: Ho imparato a gestire la fisica degli oggetti e la loro interazione con il mondo che li circonda. Ho imparato ad applicare forze agli oggetti in modo da spostarli o lanciarli via.
 - **Collider_{|g|}**: Imparare a gestire e interagire con i collider è stata una delle cose più utili apprese, in quanto i collider servono per gestire le interazioni degli oggetti tra di loro e con l'esterno.
 - **Raycast**: La gestione dei raycast è fondamentale per permettere l'interazione dell'utente con gli oggetti virtuali tramite i tap su schermo.
 - **Animazioni e Animator**: Sono due aspetti di Unity che ho imparato a conoscere in profondità, in quanto largamente utilizzati all'interno del progetto.
 - **Creazione di GUI_{|g|}**: creare una buona GUI_{|g|} è di importanza fondamentale, non solo internamente a Unity ma per qualsiasi tipo di applicazione. Ho compreso come gestire testi, immagini, pulsanti, pannelli e *slider*.
- **Vuforia**: Ciò che ho appreso di Vuforia è stata principalmente la gestione di un singolo tag, ma anche una base per il *multi-target*. Le principali nozioni apprese sono state:
 - **Gestione Tag**: Ho appreso come gestire il caricamento dei tag sul portale di Vuforia e come riconoscere i tag ottimali e quelli pessimi. Ho imparato a utilizzare i tag internamente a Unity grazie al *plugin* fornito da Vuforia SDK e ad assegnargli dei contenuti.
 - **Modelli 3D**: I contenuti utilizzati per la realtà aumentata sono stati principalmente modelli 3D semplici, ma ho imparato anche a gestire modelli 3D complessi, quali sono gli *avatar_{|g|}*.
 - **Video**: Oltre a contenuti 3D ho appreso come collegare i video a un tag e come implementare la funzionalità di video *follow*, che permette a un video in esecuzione di disancorarsi dal tag e di seguire il device quando viene perso il *tracking*.
 - **Extended Tracking**: Ho appreso quando è meglio usare questa funzionalità permessa da Vuforia SDK e come viene applicata. Inoltre, ho imparato come ottimizzare al meglio gli oggetti che sfruttano questa opzione del tag.
 - **C#**: È un linguaggio di programmazione utilizzato per lo scripting delle componenti interne a Unity. Il linguaggio è risultato molto simile al Java e al C++, ma con meno simbolismi e meno elementi decorativi ma comunque orientato agli oggetti in modo nativo. Non ci sono stati problemi nell'apprendimento di questo linguaggio, in quanto la mia preparazione in Java e C++ è stata più che sufficiente.

Ho potuto, inoltre, imparare a utilizzare metodologie e strumenti all'avanguardia, come:

- **Metodologia Agile:** Durante tutto il periodo di stage ho potuto apprendere i meccanismi su cui si basa la metodologia Agile, comprendendone le priorità e le tempistiche di lavoro. Ho potuto, inoltre, capire il grado di esperienza necessario per la migliore attuazione di questa metodologia e la sinergia interna al gruppo richiesta per il raggiungimento degli obiettivi finali.
- **Kanban Board:** Ho imparato a usare questo strumento per la gestione dei progetti, trovandolo fondamentale in ambito Agile, in quanto permette di quantificare il tempo dedicato ad ogni attività e permette a tutti i membri del *team* di sviluppo di conoscere lo stato dei progetti e sapere a cosa stanno lavorando i colleghi.
- **Repository e SVN:** L'uso di *Repository* e del versionamento era una pratica già conosciuta e utilizzata che ho potuto consolidare nel periodo di stage. Inoltre, ho imparato a usare uno strumento di *subversion* quale Tortoise SVN prima sconosciuto. Mentre per i progetti didattici svolti il livello di dettaglio nei *commit* era relativamente importante, mi sono trovato a dover fornire un livello considerevolmente maggiore dato che un progetto può essere ripreso anche dopo mesi dalla sua ultima modifica.

Infine, la mia crescita personale è avvenuta anche grazie a un miglioramento nelle mie capacità in fatto di:

- **Comunicazione:** Sia interna alla squadra di sviluppo, sia con gli altri componenti di Experenti. Ho imparato a comunicare in modo preciso riguardo sia a tematiche tecniche, sia per quanto riguarda la comunicazione esterna con il cliente avvenuta tramite il *Project Manager*. È stato fondamentale imparare a collaborare di squadra, utilizzando una metodologia Agile che impone frequente comunicazione sia interna che esterna.
- **Responsabilità:** Durante lo svolgimento del progetto mi è stata data grande responsabilità sulla buona riuscita del prodotto. Ho avuto campo libero per agire, richiedere modifiche progettuali sull'usabilità, e per gestire al meglio il mio tempo. Il piano di lavoro riguardante il progetto, infatti, è stato stilato insieme al *Project Manager* secondo le tempistiche da me comunicate.
- **Quantificazione del lavoro:** Ho potuto apprendere come quantificare al meglio il tempo necessario per un determinato lavoro. Ovviamente, questa abilità è stata ottenuta solo verso la fine dello stage al ripetersi di attività già svolte in precedenza.

4.3 Distanza tra università e lavoro

La distanza tra università e mondo del lavoro è ancora ampia, ma il corso di studi seguito ha permesso di avvicinare queste due realtà. In particolare, grazie a:

- **Seminari:** Durante l'ultimo anno del corso di studi ho potuto prendere parte a numerosi seminari, tecnologici e non, che mi hanno permesso di gettare le basi per l'apprendimento delle nuove tecnologie utilizzate e di capire le metodologie di

lavoro aziendali. Seminari di questo genere sono serviti per avere una visione della tipologia di figure ricercate da parte delle aziende, e delle conoscenze richieste. Il seminario tenuto dai ragazzi che lavorano a Google mi ha permesso, inoltre, di capire meglio come funzionano i colloqui in un'azienda esigente che lavora con un alto tasso innovativo.

- **Corsi mirati:** Sempre durante la mia carriera universitaria ho potuto seguire corsi obbligatori e opzionali, mirati all'avvicinamento con il mondo del lavoro. Questo è l'esempio del corso di Ingegneria del *Software* che grazie a un complesso progetto mi ha permesso di avere un assaggio del lavoro in *team* in condizione di risorse limitate con tempi stringenti.

Un altro corso di fondamentale importanza per la mia formazione è stato quello di Gestione di Imprese Informatiche, che mi ha mostrato come è composta un'azienda e soprattutto come si crea un'azienda. Il corso, inoltre, prevedeva un progetto di simulazione di creazione di *startup* in un team di 4 persone.

- **Eventi:** L'università prepara eventi di incontro tra studenti e aziende come STAGE-IT, un evento che mi ha permesso di vedere numerose realtà aziendali e di incontrare Experenti.

Oltre a questo, la mia partecipazione ad altri eventi quali, per esempio SMAU, mi hanno permesso di farmi conoscere dalle varie aziende del settore ICT e a mia volta di capire dove puntano le aziende e che tipo di figure cercano.

D'altra parte, alcuni elementi fanno da barriera per un passaggio agevolato da università a lavoro. I principali sono:

- **Tecnologie obsolete:** I corsi tenuti in ambito di programmazione vengono tenuti usando linguaggi di programmazione ormai obsoleti e caduti in disuso. Questo è l'esempio del linguaggio Perl usato nel corso di Tecnologie Web. Per avvicinare al mondo del lavoro è richiesta almeno una base solida sui più moderni linguaggi e framework quali: Ruby on Rails, Angular.js, Javascript.
- **Corsi fortemente teorici:** Un elemento molto importante dovrebbe essere la maggiore presenza di corsi mirati all'integrazione nel mondo lavorativo. Questo avviene in troppi pochi corsi di cui molti opzionali.

Gli elementi che rendono ancora ampio questo divario non sono esclusivamente colpe dell'Università ma anche richieste troppo esigenti da parte delle aziende.

Durante i numerosi colloqui svolti mi sono trovato in molte situazioni in cui l'azienda richiedeva la conoscenza di decine di tecnologie di cui alcune altamente innovative. La maggior parte delle aziende, quindi, richiede figure già formate, cosa che l'Università non può procurare visto il settore in cui lavoriamo in continua evoluzione.

4.4 Valutazione personale

Per quanto riguarda la mia personale esperienza di stage posso ritenermi pienamente soddisfatto. Tramite l'evento STAGE-IT ho avuto modo di conoscere varie realtà aziendali, partendo dalle startup formate da 4-5 persone arrivando ad aziende con all'attivo più di 80 dipendenti. Ho avuto modo di effettuare diversi colloqui e di visitare in sede diverse aziende. Tutte le realtà che ho visto mi hanno incantato. Ho potuto visitare l'ambiente sobrio di startup visitate all'interno di centri di ricerca, assaporare uno stile indipendente e altamente innovativo, ma ho potuto vedere gli ampi e ben arredati spazi di una grande azienda, della serietà con cui lavorano e la voglia che hanno di crescere e migliorare continuamente.

Fortunatamente, ho potuto osservare quanta speranza è ancora riposta in noi giovani, in un settore in cui le persone e soprattutto il valore di una persona viene ancora riconosciuto come una risorsa fondamentale per la crescita di un'azienda.

Alla fine, la mia scelta di svolgere lo stage ad Experenti è stata ampiamente ripagata dalle numerose soddisfazioni ottenute e dal modo in cui sono stato accolto all'interno dell'azienda, trovando un ambiente accogliente e persone aperte nei miei confronti.

Un'esperienza sicuramente appagante che mi ha fatto scoprire le mie vere capacità, rendendole disponibili anche agli altri. Tutto il team di Experenti si è dimostrato professionale e disponibile nel supportarmi nella mia formazione, fornendomi spiegazioni dedicate e rispondendo ad ogni mio dubbio. Sono stati molto propensi alle mie proposte, sia contestualmente al progetto da svolgere, sia per l'integrazione di nuovi strumenti di supporto. Concludo affermando la mia soddisfazione per il corso di studi scelto, visto non solo in ottica di nozioni apprese ma in una visione di maturazione a livello personale e professionale.

4.5 Screenshot finali



Figura 27: Presentazione iniziale - Ventaglio di legno in apertura



Figura 28: Presentazione iniziale - Avatar

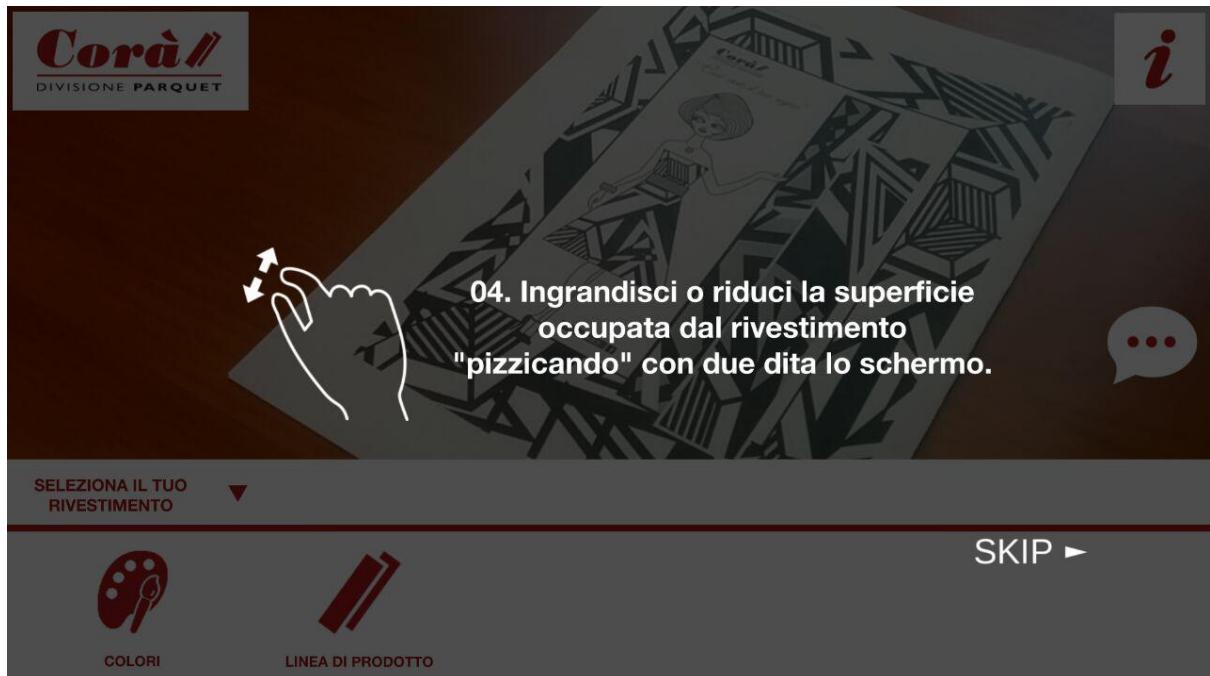


Figura 29: Tutorial - Istruzioni

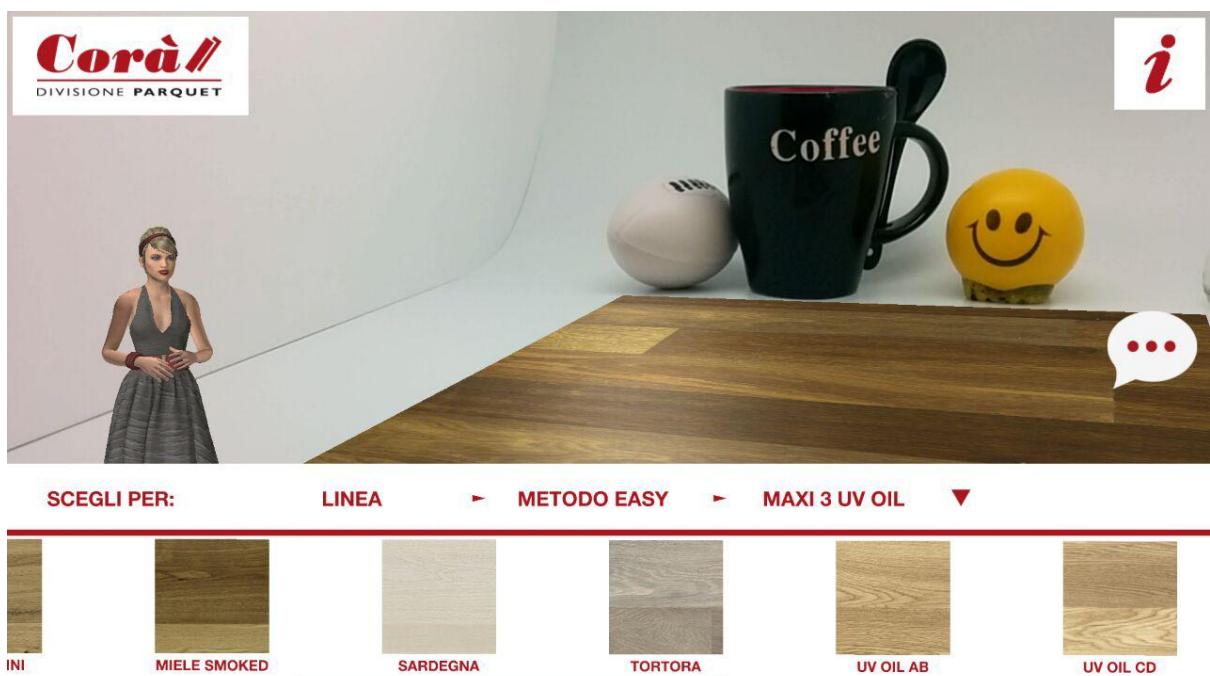


Figura 30: Configuratore - Menù inferiore aperto e Avatar in presentazione



Figura 31: Configuratore - Pannello info aperto



Figura 32: Configuratore - Menù dei Credits aperto

A Realtà Aumentata

La realtà aumentata consiste nell'arricchimento della percezione sensoriale umana mediante informazioni, in genere manipolate e convogliate elettronicamente, che non sarebbero percepibili con i cinque sensi. Gli elementi che "aumentano" la realtà possono essere aggiunti attraverso un dispositivo mobile, come uno *smartphone*, con l'uso di un PC dotato di webcam o altri sensori, con dispositivi di visione (per es. occhiali a proiezione sulla retina), di ascolto (auricolari) e di manipolazione (guanti) che aggiungono informazioni multimediali alla realtà già normalmente percepita.

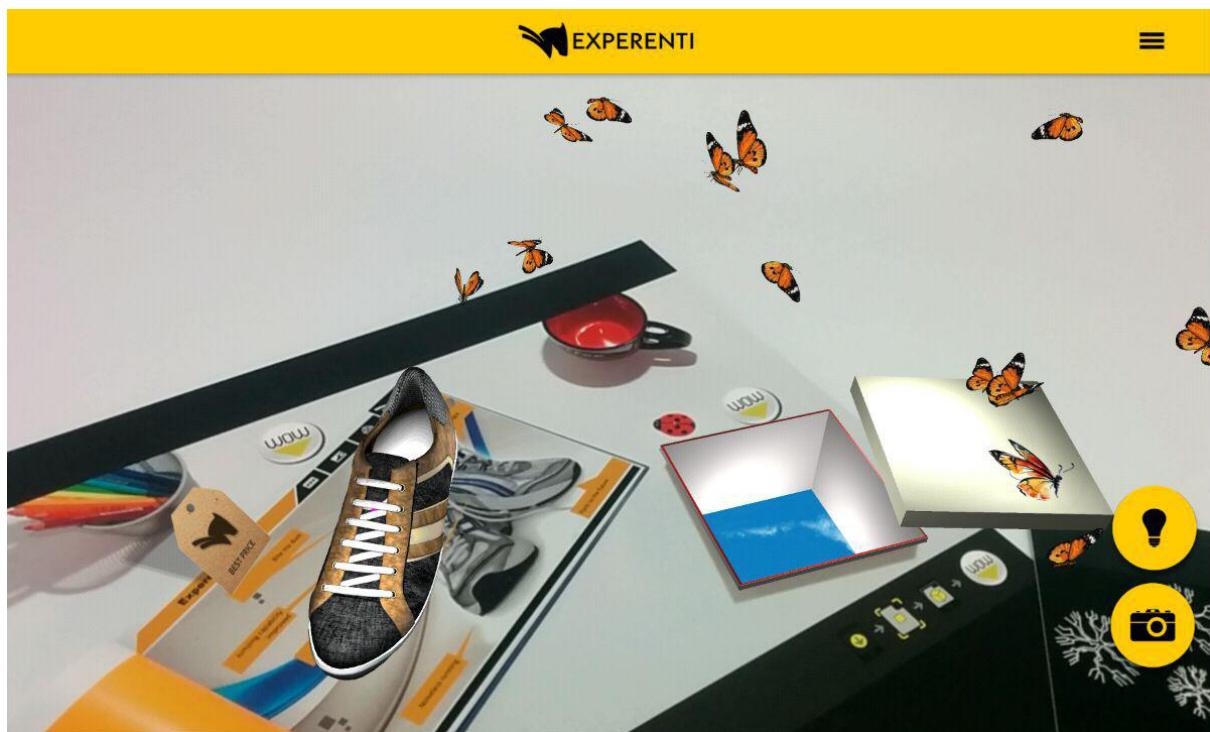


Figura 33: Esempio di realtà aumentata

Le informazioni "aggiuntive" possono, però, consistere anche in una diminuzione della quantità di informazioni normalmente percepibili per via sensoriale, sempre al fine di presentare una situazione più chiara o più utile o più divertente. Anche in questo caso si parla di AR.

Nella realtà virtuale (*virtual reality*, VR), le informazioni aggiunte o sottratte elettronicamente sono preponderanti, al punto che le persone si trovano immerse in una situazione nella quale le percezioni naturali di molti dei cinque sensi non sembrano neppure essere più presenti e sono sostituite da altre. Nella realtà aumentata (AR), invece, la persona continua a vivere la comune realtà fisica, ma usufruisce di informazioni aggiuntive o manipolate della realtà stessa.

Le informazioni circa il mondo reale che circonda l'utente possono diventare interattive e manipolabili digitalmente. Le informazioni che "aumentano" la realtà possono essere presenti nella memoria del dispositivo utilizzato, oppure possono essere ricavate da inter-

net in tempo reale.

Prima di essere impiegata in ambito mobile, con applicazioni per *smartphone* e *tablet* o visori da indossare, la realtà aumentata è stata introdotta in ambiti specifici come quello della ricerca, della medicina o nel settore militare. Basti pensare, ad esempio, agli *head-up display* (HUD) equipaggiati sugli aerei da combattimento, che mostrano al pilota informazioni come la distanza dall'obiettivo o l'inclinazione del velivolo, permettendogli di mantenere lo sguardo fisso su ciò che ha di fronte. In tempi recenti una delle prime app mobile a sfruttare questo approccio è stata Layar. Si tratta di un software che, sfruttando le informazioni di geolocalizzazione fornite dal modulo GPS del dispositivo, e accoppiandole con l'orientamento dello schermo individuato da accelerometro o giroscopio, permette all'utente di inquadrare attraverso la fotocamera l'ambiente circostante, visualizzando icone relative ai punti di interesse presenti nelle vicinanze, esattamente nella direzione in cui si trovano. Questo può risultare utile quando si cerca un ristorante, per capire che strada percorrere per raggiungerlo, oppure in modo da sapere in tempo reale la posizione di altre persone nei dintorni.

La realtà aumentata è una tecnologia applicabile a molti contesti diversi (contrariamente alla realtà virtuale che trova le sue principali applicazioni in ambito gaming e multimediale). I principali campi in cui può essere implementata spaziano dall'*advertising* al *gaming*, dall'edilizia all'arte e all'istruzione.

Glossario

A

Asset: È una risorsa utilizzata in Unity. Un *asset* può essere, per esempio, un modello 3D, uno sprite, una *texture*, o qualsiasi oggetto di gioco (*game object*).

Avatar: È un modello 3D o un'immagine scelta per rappresentare un'entità in ambientazioni virtuali.

B

Business: Insieme delle attività che contribuiscono maggiormente alla produzione del fatturato.

C

Client-server: La presenza di un *server* permette ad un certo numero di *client* di condividerne le risorse, lasciando che sia il server a gestire gli accessi alle risorse per evitare conflitti di utilizzazione.

Collider: Componente di Unity che serve a gestire le collisioni dell'oggetto con gli altri oggetti di gioco. *Collider* è la classe base da cui ereditano tutti i vari tipi di *collider*, ognuno dei quali ha una forma diversa.

Cross-Platform: Si riferisce ad un linguaggio di programmazione, ad un'applicazione *software* o ad un dispositivo *hardware* che funziona su più di un sistema o appunto, piattaforma.

G

Gantt: Un diagramma di Gantt permette la rappresentazione grafica di un calendario di attività, utile al fine di pianificare, coordinare e tracciare specifiche attività in un progetto dando una chiara illustrazione dello stato d'avanzamento del progetto rappresentato.

GUI: È un tipo di interfaccia utente che consente all'utente di interagire con il *device* controllando oggetti grafici convenzionali.

I

IDE: Sta per *Integrated Development Environment* ed indica un ambiente di sviluppo integrato, ovvero un *software* utilizzato dai programmatore per sviluppare il codice sorgente di un programma.

P

PSD: È un formato di file nativo di Adobe Photoshop per il salvataggio di immagini con le differenti caratteristiche gestite dal programma.

R

Revenue: introiti che l'azienda riceve dalla sua normale attività di *business*

S

Software: È un termine generico che definisce programmi e procedure utilizzati per far eseguire al computer un determinato compito.

T

Texture: Immagine di qualsiasi tipo utilizzata per rivestire la superficie di un oggetto virtuale, tridimensionale o bidimensionale, con un apposito programma di grafica.

Bibliografia e Sitografia

Siti consultati

- <http://www.odoo-italia.org/>
- <http://www.agilemanifesto.org/iso/it/>
- <http://www.math.unipd.it/~tullio/IS-1/2014/>
- <http://docs.unity3d.com/Manual/index.html>
- <https://developer.vuforia.com/>
- <http://doc.photonengine.com/en/pun/current/getting-started/>

So long, and thanks for all the fish.

