Alma Mater Studiorum · Università di Bologna

SECONDA FACOLTÀ DI INGEGNERIA CON SEDE A CESENA
Corso di Laurea Triennale in Ingegneria Informatica

APPLICAZIONI AUGMENTED REALITY PER ANDROID

Tesi di Laurea in Ingegneria del Software

Relatore:
Chiar.mo Prof.
Antonio Natali

Presentata da:
Andrea Nicolò

Sessione Terza
Anno Accademico 2010/2011

"Reality can be beaten
with enough imagination."

— Mark Twain

Indice

Introduzione

Con il presente elaborato ho scelto di trattare l'argomento delle tecnologie *augmented reality* (ossia *realtà aumentata*, spesso abbreviato in *AR*), con particolare enfasi sulle applicazioni per la piattaforma Android™, non solo come overview delle sue caratteristiche in generale, ma anche come analisi degli strumenti disponibili per il progettista software interessato allo sviluppo di applicazioni AR su Android. Quest'ultima è complementata dalla descrizione di un progetto di esempio, denominato ARTC, progettato e implementato mediante uno dei tool analizzati (Layar).

Per augmented reality si intende, come spiegato più dettagliatamente all'interno della dissertazione al fine di addolcire la curva di apprendimento per il lettore non ancora familiare con tale tecnologia, la sovrapposizione di elementi virtuali, tipicamente generati da un calcolatore, al mondo reale, catturato ad esempio su video tramite una telecamera, con l'obiettivo di aggiungere informazione o elementi interattivi.

La scelta di focalizzare il lavoro di tesi sulle applicazioni augmented reality per Android, sistema operativo per dispositivi mobili quali smartphone e tablet, è dovuta principalmente al fatto che la piattaforma, nel Compatibility Definition Document (CDD) che ne specifica i requisiti hardware e software, già include l'hardware per esse necessario, il tutto integrato all'interno di dispositivi dalle dimensioni limitate, dai costi accessibili e dalla diffusione attualmente molto elevata ed in costante crescita. Una volta constatata la complessità intrinseca nello sviluppo da zero di applicazioni basate sull'AR, la disponibilità di svariati middleware per Android, progettati per semplificare ed automatizzare la gestione di interfacce augmented reality, ha consentito di cambiare punto di vista verso

quello dello sviluppatore interessato ad utilizzarle, e di seguire un approccio di tipo bottom-up, fissando la tecnologia ed i tool da utilizzare per implementare la business logic in modo semplice ed efficiente. Dopo aver analizzato diversi framework, la scelta dello strumento di cui servirsi per l'implementazione del progetto è caduta sul browser AR Layar, la cui infrastruttura, dettagliata nel quarto capitolo, si è rivelata particolarmente adatta per lo sviluppo di un'applicazione completa in tempi brevi.

Layar, infatti, usa un'architettura di tipo client-server ed automatizza, oltre alla gestione e al rendering della visuale AR, anche le funzionalità di comunicazione tra i due endpoint. Il lato client, eseguito sul dispositivo Android, è inoltre generato automaticamente tramite un semplice tool online. Ciò significa che la realizzazione di software (content layer) eseguito sulla piattaforma si riduce alla semplice implementazione della logica applicativa sul lato server, secondo le modalità descritte all'interno della dissertazione, il che comporta una sostanziale riduzione degli oneri a carico dello sviluppatore e, di conseguenza, tempi e costi inferiori.

Più nel dettaglio, il primo capitolo della tesi è dedicato ad un'introduzione intuitiva e generale dell'augmented reality come sovrapposizione di elementi reali e virtuali, seguita poi dall'elaborazione di una definizione più formale basata su quelle date da Milgram, Mann e Azuma tra 1992 e 2002, e da alcuni cenni sulla sua evoluzione storica a partire dagli anni Cinquanta. Durante la stesura sono state inoltre affrontate problematiche fondamentali quali la complessità comportata dalle trasformazioni in tempo reale tra sistemi di coordinate differenti, necessarie perchè tale sovrapposizione sia consistente, e alcuni possibili effetti indesiderati causati da hardware inadeguato (errori statici e dinamici), e sono stati inoltre descritti i principali dispositivi di input/output utilizzati nel campo.

L'overview generale delle caratteristiche dell'AR prosegue quindi con il secondo capitolo, dedicato all'analisi dei suoi domini applicativi principali. Svariati settori, dal campo medico all'ingegneria, passando per le applicazioni militari o di intrattenimento, possono infatti trarre beneficio dall'augmented reality, ed il capitolo ne approfondisce le applicazioni attuali e le potenzialità attraverso la descrizione di svariati esempi concreti.

Il terzo capitolo introduce la piattaforma Android, descrivendo l'architettura software del sistema operativo, la struttura ed il funzionamento delle applicazioni (scritte in Java ed eseguite su una macchina virtuale), ed introducendo le tipologie di componenti utilizzati da queste ultime (attività, servizi, content provider e broadcast receiver), la cui conoscenza è fondamentale per lo sviluppo di applicazioni sulla piattaforma.

Il quarto capitolo si concentra sulle applicazioni AR per Android, e incarna lo shift di prospettiva, già menzionato in precedenza, verso il punto di vista dello sviluppatore. Oltre a fornire un'overview sull'hardware utilizzato specificamente sulla piattaforma, e alcuni esempi di software AR già disponibile sull'Android Market, infatti, il capitolo descrive nel dettaglio la struttura ed il funzionamento di tre framework AR in particolare (NyARToolkit, AndAR e QCAR/Vuforia SDK), scelti rispettivamente per diffusione, semplicità e completezza, prima di introdurre il già citato browser AR general purpose Layar ed analizzarne le caratteristiche.

Il quinto capitolo conclude l'elaborato descrivendo il progetto ARTC, con l'obiettivo di evidenziare i passi da seguire e gli elementi principali di cui tenere conto durante lo sviluppo di content layer per la piattaforma Layar. Questi sono fondamentalmente indipendenti dalla logica applicativa del progetto in sè, in quanto quest'ultima dipende più che altro dalla natura o dall'origine dei dati in esame, ed i concetti introdotti durante il capitolo sono quindi altamente riutilizzabili in progetti futuri.

Capitolo 1

Augmented Reality

1.1 Cos'è l'augmented reality

Per augmented reality (AR) si intende la sovrapposizione di contenuti sensoriali generati da un sistema informatico, quali immagini o audio, a contenuti catturati dal mondo reale per mezzo, ad esempio, di una videocamera o una webcam, allo scopo di aggiungere informazione o elementi interattivi. Gli elementi aggiunti sono inesistenti nella realtà fisica (virtuali), e sono pertanto fruibili dall'utente esclusivamente attraverso l'uso di un dispositivo, come ad esempio lo schermo di un computer o di un telefono cellulare, oppure ancora occhiali stereoscopici.

Più in generale, l'augmented reality è un caso particolare di *Computer-mediated reality*, ovvero di realtà modificata (mediata) tramite un calcolatore con lo scopo di alterarne la percezione complessiva o la quantità di informazione presente. A differenza della forse più nota realtà virtuale, le modifiche si limitano ad affiancare la realtà fisica, integrandola con l'aggiunta di elementi virtuali ma senza sostituirla del tutto, in modo che la percezione da parte dell'utente rimanga comunque quella di un ambiente prevalentemente reale.

Tali modifiche avvengono tipicamente, ma non esclusivamente, in real time, e devono pertanto essere elaborate dal sistema ad una velocità tale da essere sovrapposte ad un flusso video catturato dal mondo reale senza introdurre ritardi ed in modo trasparente all'utilizzatore, per non infran-

gere la sua percezione della realtà, e per consentirgli una più intuitiva ed efficace interazione con essa.

Un semplice esempio di augmented reality è la sovraimpressione in tempo reale di testo o immagini al flusso video di una trasmissione televisiva, come l'interfaccia grafica che mostra il punteggio ed il tempo di gioco in una partita di calcio, o le immagini tridimensionali affiancate in chroma-key alle riprese in studio durante le previsioni meteorologiche. Tale sovrapposizione aumenta infatti la quantità di informazione disponibile all'utente tramite elementi originariamente non presenti nel flusso video catturato dalle telecamere, generati in tempo reale da un computer e visibili solo per mezzo di uno schermo.

La natura degli elementi aggiunti è fortemente dipendente dal contesto semantico della loro applicazione: esistono infatti svariati domini applicativi in cui l'augmented reality ha avuto un forte sviluppo negli ultimi anni, ed il continuo miglioramento delle tecnologie di computer vision ha consentito un progressivo ampliamento di tali domini. La possibilità di riconoscere degli oggetti fisici in un flusso video (object recognition) ha infatti consentito lo sviluppo di applicazioni in grado di elaborare contenuti il cui stato o le cui proprietà variano in tempo reale in funzione dello stato o delle proprietà di tali oggetti, rendendo possibile una interazione con il programma tramite la loro manipolazione fisica o semplicemente utilizzandoli come punti di riferimento nello spazio.

Un classico esempio è il riconoscimento di un *marker*, ossia un oggetto fisico il cui aspetto esteriore è stato progettato appositamente per essere riconosciuto e tracciato con facilità, e che viene inserito intenzionalmente nel flusso video come punto di riferimento per il software, e per consentire all'utente, che lo manipola al momento della cattura del video, di interagire con il programma stesso senza bisogno di ulteriori periferiche di input, visualizzandone i risultati in tempo reale.

E' facile intuire come attualmente sia proprio l'object recognition una delle componenti con maggiori potenzialità dell'augmented reality. Oggi esistono svariate piattaforme per lo sviluppo di applicazioni AR il cui fulcro sono proprio gli strumenti per la creazione dei marker ed il loro

riconoscimento, ed è verosimile che l'evoluzione delle tecnologie di computer vision continui ad essere una componente chiave per lo sviluppo di applicazioni augmented reality anche in futuro.

Ma prima di poter discutere tali applicazioni e le loro potenzialità, e prima di poter analizzare le tecnologie oggi utilizzate nel campo, è necessario dare una definizione più rigorosa di augmented reality ed analizzare la sua evoluzione storica.

1.2 Definizione formale

Il conio del termine augmented reality è attribuito a Thomas Caudell e David Mizell[1], ricercatori dell'azienda costruttrice di aeromobili Boeing e autori di *"Augmented reality: an application of heads-up display technology to manual manufacturing processes"* [CAU92]. In questo documento, l'AR è definita come sovrapposizione di materiale generato dal computer al mondo reale, e contrapposta al già esistente concetto di realtà virtuale come alternativa più economica e meno esosa di risorse computazionali, ideale per assistere i meccanici Boeing nelle operazioni di costruzione e manutenzione degli aeromobili mediante l'uso di *head-mounted display* (HMD) in grado di visualizzarne gli schemi progettuali in semitrasparenza.

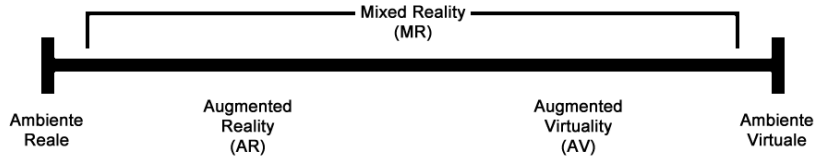

Figura 1.1: Milgram's Reality-Virtuality Continuum

La definizione di Caudell è poi approfondita da Paul Milgram[2] e Fumio Kishino[3] che, nella stesura di *"A Taxonomy of Mixed Reality Visual*

[1]T. P. Caudell, D. Mizell, D.W. Res. & Technol., Boeing Comput. Services, Seattle
[2]P. Milgram, Dept. of Industrial Engineering, University of Toronto
[3]F. Kishino, ATR Communication Systems Research Laboratories

Displays" [MIL94], introducono il *Milgram's Reality-Virtuality Continuum* (Continuum della Realtà-Virtualità di Milgram, figura 1.1), ossia l'intervallo continuo tra un ambiente puramente reale (R) ed uno puramente virtuale (V), i cui punti interni sono detti *mixed reality*.

La mixed reality, quindi, si frappone tra i due estremi del continuum e rappresenta una composizione tra di essi, mediata dal computer, in cui reale e virtuale si incontrano e si fondono. Il variare della predominanza dell'uno o dell'altro, ossia la percezione complessiva della composizione come ambiente reale modificato da elementi virtuali o, viceversa, come ambiente virtuale modificato da elementi reali, implica una classificazione per tale composizione come augmented reality nel primo caso, o come *augmented virtuality* nel caso opposto.

Inquadrata all'interno del continuum, l'augmented reality è quindi definibile come un'istanza di mixed reality percepita da un osservatore come ambiente prevalentemente reale modificato da elementi virtuali.

Tale definizione, però, non è sufficiente, in quanto si limita a considerare la prevalenza di elementi virtuali o reali senza tener conto di quanto essi siano poi simili o differenti dalla realtà. Due istanze di augmented reality in cui l'incidenza delle modifiche effettuate dal calcolatore è tutto sommato simile, possono essere percepite in modo completamente diverso dall'utente in base alla natura degli elementi coinvolti, e da quanto essi siano integrati o percepiti come distanti dal mondo reale a cui vengono sovrapposti. Inoltre, è anche possibile allontanarsi dalla realtà senza l'utilizzo di alcun elemento virtuale: prismi o sistemi ottici, ad esempio, possono distorcerne o alterarne la percezione senza che vi sia alcun tipo di composizione tra reale e virtuale. In quest'ultimo caso, in particolare, la tassonomia individuata da Milgram si rivela insufficiente, in quanto ciò che viene percepito è difficilmente classificabile come un ambiente puramente reale, e al tempo stesso non è classificabile come mixed reality per l'assenza di elementi virtuali.

Steve Mann[4], nella stesura di *"Mediated Reality with implementations for everyday life"* [MAN02], estende il Reality-Virtuality Continuum ad una

[4]S. Mann, Dept. of Electrical and Computer Engineering, University of Toronto

seconda dimensione (fig. 1.2) con l'aggiunta di un secondo asse chiamato *Mediality*, che introduce un concetto da lui definito come *mediated reality* (realtà mediata), ovvero una generalizzazione della mixed reality di Milgram che consente di classificare realtà modificate in modo differente, e non necessariamente dipendente, dalla semplice presenza di elementi virtuali, come il già citato esempio di un prisma ottico. La mediated reality consente inoltre di introdurre il concetto di *diminished reality* (realtà diminuita), ovvero la rimozione digitale di elementi reali dal flusso video catturato, non classificabile nel continuum di Milgram in quanto non vi è alcuna presenza di elementi virtuali.

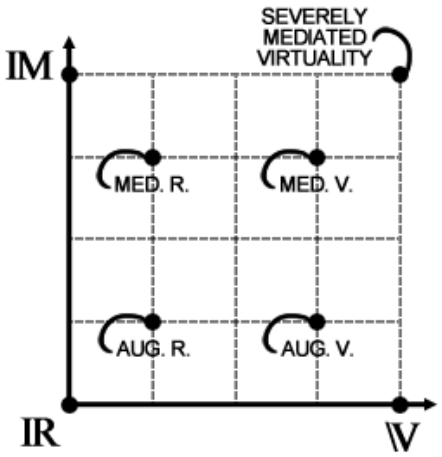

Figura 1.2: Tassonomia di Mann

Più nel dettaglio, la tassonomia di Reality, Virtuality e Mediality di Mann è descritta da un piano cartesiano con origine Reale (R), ovvero la realtà pura e non modificata, ed i cui assi sono Virtuality (V) per le ascisse e Mediality (M) per le ordinate.

Quest'ultimo indica il livello di mediazione, ovvero quantifica l'entità delle modifiche effettuate. Più ci si allontana dall'origine e più la crescente mediazione causa una percezione di complessivo allontanamento dalla realtà. Per valori sufficientemente grandi di M, le regioni di augmented reality e augmented virtuality sono classificate, per sottolineare la

rilevanza delle modifiche, come *mediated reality* e *mediated virtuality*.

Nel punto diagonalmente opposto all'origine, alla massima distanza sia lungo V che lungo M, in particolare, si ha una percezione di un mondo puramente virtuale così fortemente mediato da essere completamente diverso dalla realtà (*severely mediated virtuality*).

Va sottolineato che la tassonomia di Mann non è strettamente limitata al campo informatico e multimediale, ma è un concetto del tutto generale che, data la sempre crescente pervasività del virtuale nella vita di tutti i giorni dovuta alla rapida diffusione delle nuove tecnologie e di nuovi mezzi di comunicazione (e-mail, SMS), trova applicazioni anche in ambito sociologico e antropologico. Inoltre, come già detto in precedenza, visto che il concetto di Mediality consente di introdurre modifiche non necessariamente dovute alla presenza di elementi virtuali, non riguarda esclusivamente istanze di realtà mediata tramite l'utilizzo di un calcolatore. Per tale motivo spesso viene utilizzato il più specifico termine *Computer-mediated reality*, già usato nel paragrafo precedente, in modo da restringere la sua applicazione al campo delle modifiche effettuate da un computer.

Inquadrata all'interno di tale tassonomia, quindi, l'augmented reality è definibile come un'istanza di Computer-mediated reality percepita da un osservatore come ambiente prevalentemente reale modificato da elementi virtuali (definizione del tutto equivalente a quella nel continuum di Milgram) per valori sufficientemente piccoli di Mediality tali da mantenere una percezione di complessiva similitudine alla realtà.

Ronald T. Azuma[5], nella stesura di *"A Survey of Augmented Reality"* [AZU97], documento successivo a quello di Milgram ma precedente a quello di Mann, fornisce un'ulteriore definizione di AR tuttora largamente accettata, tramite l'individuazione di tre punti fondamentali:

1. *Combines real and virtual*

2. *Interactive in real time*

3. *Registered in 3-D*

[5]R. T. Azuma, Hughes Research Laboratories, Malibu

Il primo punto è fondamentalmente equivalente alla definizione di Milgram, ma il secondo e il terzo specificano due dettagli finora ignorati: l'AR deve offrire interattività in tempo reale, e gli elementi virtuali devono essere combinati a quelli reali (*registration*) in tre dimensioni.

Si tratta, quindi, di una definizione più rigorosa e stringente, che esclude svariati esempi citati in precedenza come la sovraimpressione di testo o interfacce grafiche ad una trasmissione televisiva, e che invece rientrano nella più lasca definizione di AR data da Milgram e Mann.

Nella trattazione di questa tesi, si farà riferimento principalmente alla definizione data da Azuma, inquadrata all'interno della tassonomia di Mann come caso particolare di Computer-mediated reality, con qualche sporadico excursus su applicazioni non propriamente rientranti in tale definizione, ma comunque definibili come AR in modo lasco.

1.3 Cenni storici

Nonostante l'introduzione del termine augmented reality sia, come già detto, risalente al 1992, il primo dispositivo AR viene realizzato negli anni Sessanta ed un primo passo verso tale direzione, una radice in comune con la realtà virtuale, risale già a metà degli anni Cinquanta.

Nel 1955 Morton Heilig, regista e cineoperatore oggi considerato uno dei padri della realtà virtuale, pubblica il saggio *"Cinema of the Future"*, in cui descrive il cinema del futuro come in grado di catturare la realtà per ognuno dei cinque sensi, oltre a semplicemente vista e udito [HEI55].

L'impossibilità di ottenere fondi per realizzare la sua visione in un contesto cinematografico spinge Heilig a sviluppare in proprio una macchina, che battezzerà *Sensorama* e che completerà solo nel 1962, in grado di offrire allo spettatore un'esperienza multisensoriale. E' proprio Sensorama, macchina meccanica dall'aspetto simile ad un cabinato per videogiochi da bar, che proietta video in 3D stereoscopico, riproduce audio in stereo, offre feedback tattile e consente all'utente di percepire aromi, ad essere considerata un primo rudimentale esempio di realtà virtuale: pur non essendo generati da un computer, i contenuti prodotti da Sensorama di fatto mira-

no a sostituire quelli della realtà fisica, grazie ad una struttura che avvolge l'utente limitando le sue percezioni dell'ambiente esterno.

Nel 1965 Ivan Sutherland, docente associato di Ingegneria Elettrica ad Harvard e già popolare per i risultati ottenuti nel campo della computer grafica e dell'interfaccia uomo-macchina con Sketchpad[6] due anni prima, descrive l'*ultimate display* [SUT65], una finestra che si affaccia su una realtà generata e controllata dal computer, come lo specchio che l'Alice di Lewis Carroll deve attraversare per entrare in un paese delle meraviglie matematico, in cui le leggi della fisica non funzionano necessariamente come nel mondo reale ed in cui è possibile effettuare simulazioni che coinvolgano elementi inesistenti nella realtà, come ad esempio oggetti con massa negativa o in grado di diventare trasparenti quando necessario. Similmente alle teorie di Heilig poi concretizzatesi in Sensorama, Sutherland cita la possibilità di coinvolgere anche gli altri sensi, e di sfruttare tecnologie di head tracking per tracciare la direzione in cui l'utente è rivolto e mostrargli di conseguenza contenuti congruenti con ciò che è in grado di vedere.

Insieme ad uno dei suoi studenti, Bob Sproull, Sutherland realizza tale schermo nel 1968, creando così uno dei primi prototipi di head-mounted display. Troppo grande e pesante per essere semplicemente indossato, viene invece appeso al soffitto, cosa che guadagna al sistema il nome di *The Sword of Damocles* (La Spada di Damocle) [SUT68]. Il display, binoculare, è in grado di visualizzare semplici modelli in wire-frame sovrapposti in semitrasparenza alla realtà circostante grazie ad un sistema di head tracking. Si tratta, a tutti gli effetti, del primo sistema AR.

Nel 1975 l'artista Myron Krueger[7] realizza *Videoplace*, una serie di stanze dotate di videocamere e proiettori in cui gli utenti possono interagire con silhouette di oggetti virtuali, o persino di utenti in una stanza adiacente, sfruttando la silhouette del proprio corpo. Grazie a tecniche di image recognition, il sistema è in grado di registrare collisioni bidimensionali tra le silhouette, e di sfruttarle per offrire interattività, simulando un contatto fisico tra di esse. Krueger definisce questo ambiente simulato artificial

[6] Software di grafica, seminale per interfacce grafiche e object oriented programming
[7] M. Krueger, Space Science and Engineering Center, University of Wisconsin-Madison

reality (realtà artificiale) ma è facile notare come, essendo le silhouette una limitazione di natura tecnologica saggiamente sfruttata in modo artistico, si tratti di un'istanza di augmented reality interattiva: il proiettore mostra un flusso video catturato dal mondo reale a cui sono sovrapposti in real time degli elementi virtuali interattivi.

La popolarità di tale tecnologia esplode poi negli anni Novanta, quando successivamente alle ricerche dei già citati Caudell, Milgram e Mann segue una rapida successione di esperimenti e nuove applicazioni.

Vengono introdotti i marker, inizialmente color-coded ed in seguito basati su matrici in bianco e nero simili a codici a barre bidimensionali (QR Code), ed i nuovi dispositivi augmented reality diventano quindi in grado di identificare elementi della realtà catturata.

Grazie alle nuove tecnologie, diventa inoltre finalmente possibile progettare dispositivi AR effettivamente indossabili dall'utente, come il *Mobile Augmented Reality System* (MARS) di Steven Feiner[8].

L'evoluzione dell'augmented reality in questi anni non si limita all'hardware ma coinvolge anche il software. Nel 1999, Hirokazu Kato[9] e Mark Billinghurst[10] introducono *ARToolKit*, libreria open source per il tracking dei marker scritta in C e rilasciata sotto licenza GNU GPL. Grazie anche ai numerosi porting per svariate piattaforme, ARToolKit è tuttora una delle librerie più utilizzate nel settore.

Tali sviluppi consentono una rapida espansione dei domini applicativi per le applicazioni AR (discussi nel dettaglio nel prossimo capitolo) negli anni successivi, e l'introduzione di smartphone dotati di sensori e avanzate caratteristiche multimediali, in grado di eseguire software non necessariamente limitato alle basiche funzionalità di telefonia, rende questi ultimi, per la loro estrema portabilità e per l'integrazione di default in molti modelli dell'hardware necessario, piattaforme ideali per lo sviluppo di applicazioni augmented reality.

Nonostante la larga diffusione di tale hardware ed una vasta scelta di

[8] S. Feiner, Dept. of Computer Science, Columbia University, New York
[9] H. Kato, Faculty of Information Sciences, Hiroshima City University
[10] M. Billinghurst, Human Interface Technology Laboratory, University of Washington

applicazioni software liberamente accessibili, però, ancora oggi la popolarità di questa tecnologia è piuttosto limitata presso il grande pubblico. La recente introduzione di hardware con applicazioni augmented reality preinstallate, come il browser AR *Layar* (Layar, 2009) su alcuni smartphone Android oppure *AR Games* (Nintendo, 2011) sulla console portatile per videogiochi Nintendo 3DS, potrebbe tuttavia essere nei prossimi anni un trampolino di lancio verso il mercato di massa, e segnare una nuova importante svolta per la tecnologia e le sue applicazioni.

1.4 Unire reale e virtuale

Una problematica fondamentale di ogni applicazione AR propriamente detta consiste nel combinare in modo consistente elementi reali e virtuali come se fossero parte di un unico ambiente, in modo da fornire all'utente l'illusione che gli elementi virtuali facciano parte del mondo reale.

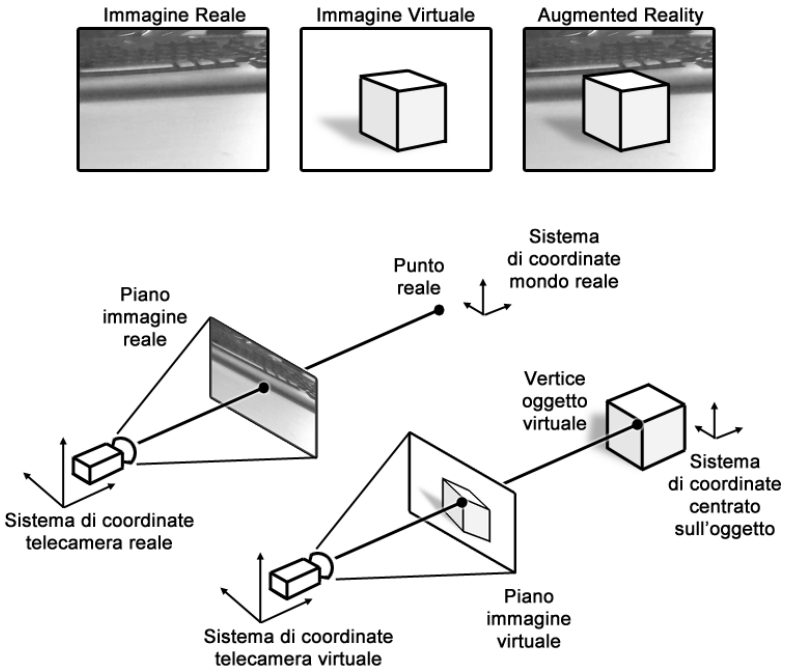

Figura 1.3: Sistemi di coordinate in augmented reality

Come visibile in figura, tale sovrapposizione implica una serie di trasformazioni in tempo reale tra differenti sistemi di coordinate. [VAL98]

Per comporre una scena in augmented reality, è necessario specificare innanzitutto la posizione e l'orientamento dell'oggetto virtuale rispetto al sistema di coordinate reale, effettuando una prima trasformazione basata sulla relazione tra oggetto e mondo reale. Occorre inoltre definire le coordinate della telecamera rispetto a quest'ultimo, mediante una seconda trasformazione, ed infine occorre proiettare la scena tridimensionale catturata dalla telecamera su un piano bidimensionale, ovvero il piano che compone l'immagine finale. L'utilizzo di marker, dalle dimensioni note, come punto di riferimento può rivelarsi molto utile per la composizione, in quanto le loro dimensioni all'interno dell'immagine, e la loro posizione e angolazione, possono tradursi direttamente in una trasformazione equivalente da applicare all'oggetto virtuale ad essi associato.

Perchè l'effetto sia convincente e l'utente abbia l'impressione di una sovrapposizione consistente, è necessario che le trasformazioni siano accurate, e che la relazione tra oggetto virtuale e mondo reale sia ben definita geometricamente. In caso di inconsistenze, è possibile classificare due tipi diversi di errore: statici e dinamici.

Gli errori statici sono percepiti dall'utente come un posizionamento incorretto dell'oggetto virtuale all'interno della scena. Ad esempio, un cubo virtuale posizionato sopra un tavolo reale potrebbe apparire invece oltre i bordi del tavolo con una diversa inquadratura. Una delle cause principali di errori statici è tipicamente l'imprecisione dei sensori utilizzati.

Gli errori dinamici sono invece dovuti a lacune nella sincronizzazione con il mondo reale, tipicamente causate dai ritardi introdotti dai limiti hardware del sistema AR in esame. Tali limiti possono essere di natura computazionale, sensoriale o di controllo, e possono essere parzialmente aggirati utilizzando tecniche predittive per stimare la posizione dell'oggetto virtuale quando essa non è rilevabile con precisione. Spesso, però, tali tecniche si rivelano insufficienti, e l'effetto percepito dall'utente è di un oggetto virtuale che cambia rapidamente posizione in modo innaturale per alcuni fotogrammi, per poi tornare alla posizione corretta.

Sistemi AR dotati di dispositivi di output differenti, come ad esempio un head-mounted display in grado di visualizzare 3D stereoscopico oppure periferiche con feedback tattile (*haptic feedback*) presentano problematiche aggiuntive. Nel primo caso, il sistema AR dovrà renderizzare due immagini differenti, una per ogni occhio, e utilizzare due telecamere reali e due virtuali, la cui distanza deve essere il più possibile simile alla distanza intra-oculare dell'utente per garantire una visione corretta. Nel secondo caso, occorre effettuare un'ulteriore trasformazione di coordinate tra dispositivo e mondo reale, in modo da generare una sensazione tattile consistente con la scena osservata.

Entrambi gli esempi incidono sui costi computazionali, introducendo ritardi che, come già detto, possono condurre a degli errori dinamici.

1.5 Hardware utilizzato

Dal punto di vista hardware, i componenti di un tipico sistema AR coinvolgono un'unità di elaborazione e dei dispositivi di input e output. Dispositivi portatili come laptop, tablet e smartphone tipicamente integrano tutto l'hardware necessario per l'esecuzione di un largo numero di applicazioni AR in svariati domini applicativi, e di conseguenza si prestano facilmente a tali scopi.

Perchè il flusso video formato da tali immagini sia percepito fluidamente, è necessaria una frequenza di refresh minima pari a circa 15-20 Hz, con valori superiori (almeno pari a 30 Hz) ovviamente preferibili.

1.5.1 Dispositivi di input

I dispositivi di input possono includere hardware di vario genere, a seconda del dominio applicativo e delle necessità della specifica applicazione. Il più importante e comunemente utilizzato è ovviamente la telecamera, o due telecamere nel caso di sistemi AR in grado di generare immagini in 3D stereoscopico, tipicamente basata su tecnologia CMOS.

A seconda dell'applicazione, possono poi essere utilizzati dispositivi classici come mouse e tastiera, touch screen, oppure meno convenzionali come guanti, puntatori o il corpo dell'utente in applicazioni che usino tecniche di computer vision e object recognition, sia con che senza l'utilizzo di marker. Ulteriori dispositivi di input, soprattutto in sistemi AR mobili o in applicazioni che necessitino di dati sulla posizione fisica dell'utente o sul suo orientamento nello spazio, possono includere sensori quali accelerometri o giroscopi, tipicamente basati su tecnologia MEMS, oppure ricevitori GPS e magnetometri.

La qualità e la precisione dei dispositivi di input utilizzati influisce sia sulla presenza di errori statici nel caso di sensori imprecisi, come ad esempio telecamere a risoluzione insufficiente o sensori MEMS con letture non sufficientemente accurate, che sulla presenza di errori dinamici se tali dispositivi introducono dei ritardi durante l'elaborazione.

1.5.2 Dispositivi di output

Così come per i dispositivi di input, anche quelli di output dipendono fortemente dalla specifica applicazione. E' fondamentale, comunque, che il sistema disponga di uno o più display per visualizzare il flusso video in uscita. Oltre ai comuni monitor per computer ed agli schermi per dispositivi portatili, sono spesso utilizzati head-mounted display di due categorie: optical see-through e video see-through. Mentre i primi utilizzano un sistema di specchi per sovrapporre in semitrasparenza gli elementi virtuali alla realtà circostante, come il già citato The Sword of Damocles o il display AR suggerito da Caudell, gli HMD di tipo video see-through utilizzano uno o due schermi opachi, del tutto simili a dei monitor, per visualizzare immagini AR complete.

Il vantaggio principale degli HMD è di non obbligare l'utente a rimanere rivolto nella direzione del monitor, o a tenere lo schermo in mano nel caso di un portatile, e spesso tali dispositivi possono integrare sensori aggiuntivi, per consentire ad esempio l'head tracking. Gli svantaggi inclu-

dono un maggiore costo, la necessità di avere un HMD per ogni persona, ed una limitazione nel campo visivo dell'utente.

Un ulteriore dispositivo di output comunemente utilizzato per visualizzare scene AR è il proiettore. Mediante uno o più proiettori digitali è possibile proiettare immagini aumentate su oggetti reali, tecnica denominata *spatial augmented reality* (SAR), consentendo a più di una persona alla volta di usufruire del programma senza incorrere nelle limitazioni dei display tradizionali e nei costi di un numero elevato di HMD. Inoltre, utilizzare oggetti reali per le proiezioni consente di ottenere feedback tattile passivo semplicemente toccandoli, ossia senza bisogno di periferiche e risorse computazionali. Gli svantaggi includono la necessità di disporre di superfici reali per le proiezioni, e la sensibilità di queste ultime nei confronti delle condizioni di illuminazione ambientale.

Oltre ai display descritti finora, la categoria dei dispositivi di output utilizzati nei sistemi AR comprende anche una serie di interfacce il cui scopo è fornire all'utente ulteriori tipi di feedback, come il già citato feedback tattile. Tali dispositivi, denominati interfacce aptiche, hanno forma, struttura e tecnologia fortemente dipendenti dalla specifica applicazione, ed utilizzano motori o attuatori per fornire all'utente sensazioni di tipo tattile quali pressione, forze o vibrazioni. Esempi possono includere un joystick con force feedback in grado di offrire resistenza fisica ai movimenti dell'utente, o display braille.

Nello specifico, in campo AR vengono utilizzate interfacce aptiche in grado di fornire, mediante force feedback, la sensazione di toccare fisicamente gli elementi virtuali. Esempi includono touch screen, guanti dotati di attuatori, esoscheletri e braccia robotiche. A causa dei costi e, talvolta, delle dimensioni, l'utilizzo di tali dispositivi al di fuori dell'ambito accademico o di un numero ristretto di laboratori è ancora oggi piuttosto ridotto. La recente introduzione di nuovi touch screen con feedback aptico, in grado di fornire all'utente la sensazione di toccare oggetti con consistenze o texture differenti, potrebbe però cambiare tale situazione non appena i suoi costi raggiungeranno il mercato consumer.

Capitolo 2

Domini applicativi

In questo capitolo verrà analizzata, senza pretesa di esaustività dato il rapido e continuo sviluppo delle tecnologie coinvolte, una serie di domini applicativi fondamentali per le applicazioni augmented reality.

Come già accennato in precedenza, infatti, il miglioramento delle tecnologie di computer vision e l'introduzione di sensori più precisi o di periferiche di input/output sempre più evolute e specializzate, hanno contribuito sensibilmente al rapido ampliamento di tali domini.

In generale, qualsiasi campo abbia a che fare con l'interpretazione di contenuti sensoriali catturati dalla realtà, soprattutto se di natura visuale, può trarre beneficio da applicazioni augmented reality che ne aumentino la quantità di informazione contenuta, ma perchè esse siano effettivamente viabili è ovviamente necessario che i benefici non si rivelino inferiori ai costi di implementazione ed agli investimenti in ricerca e sviluppo.

2.1 Applicazioni mediche

La rilevanza e la pervasività delle tecnologie di imaging nel campo medico rendono quest'ultimo un terreno fertile per l'utilizzo dell'augmented reality, tanto che al momento si tratta di uno dei domini applicativi più importanti ed in rapida espansione per tale tecnologia, con investimenti nel settore in costante aumento.

In particolare, in campo chirurgico l'AR offre strumenti avanzati di *image-guided surgery* (IGS): linee guida o immagini generate al computer partendo da TAC, ecografie o risonanze magnetiche possono essere sovrapposte in 3D in tempo reale al corpo del paziente in modo da fornire al chirurgo maggiori informazioni visive, incluse dettagliate ricostruzioni tridimensionali della specifica anatomia dello stesso, che consentono di osservarne gli organi interni come se il corpo del paziente fosse semitrasparente, con effetto simile ad una visione a raggi X. Tali ricostruzioni sono visibili già in fase preoperativa, ed oltre a facilitare il lavoro dell'equipe chirurgica durante l'intervento vero e proprio consentono quindi di pianificarlo con maggiore dettaglio e di ridurne l'invasività e la durata, con conseguente riduzione dei rischi per il paziente.

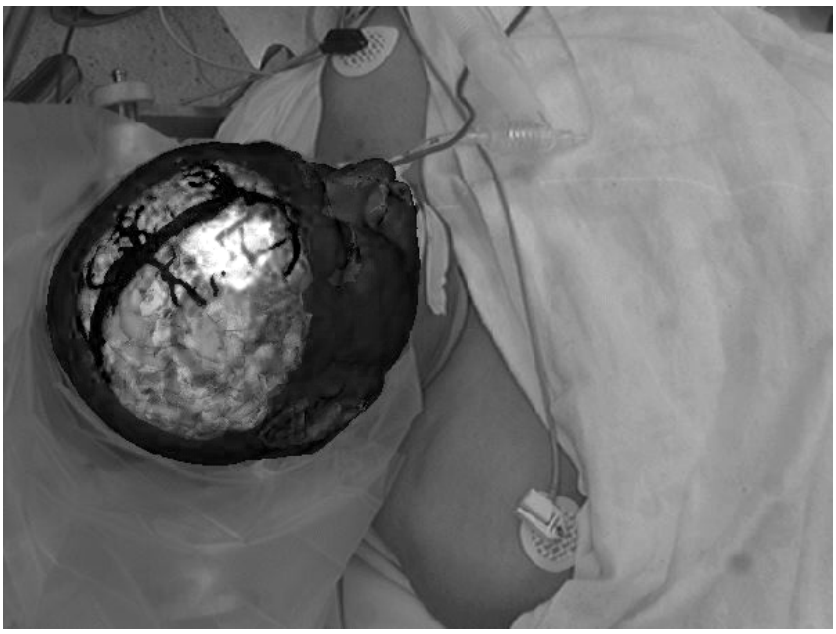

Figura 2.1: Utilizzo di un sistema AR per visualizzare l'anatomia intercraniale e la patologia in esame (tumore, evidenziato in bianco) [JOL97]

In particolari tecniche chirurgiche, come ad esempio la radiochirurgia stereotassica per il trattamento di tumori al cervello, l'AR consente di eliminare la necessità di utilizzare ingombranti frame meccanici intorno all'area interessata, snellendo l'intervento in modo significativo. In tecni-

che poco invasive quali la laparoscopia, tipologia di intervento chirurgico addominale effettuata senza l'apertura della parete per minimizzare le incisioni ed il trauma subito dal paziente, l'AR complementa la limitata visione degli organi interni da parte del chirurgo e può consentire una visualizzazione selettiva (o arricchita di informazioni) degli stessi, cosa non possibile utilizzando tecniche di imaging tradizionali.

Ulteriori applicazioni dell'augmented reality in campo chirurgico includono biopsie e interventi per la rimozione dei tumori, e la tecnologia si presta facilmente per lo sviluppo di simulazioni educative di vario genere, utilizzabili dal personale per far pratica con interventi particolarmente delicati senza mettere a rischio pazienti reali.

Come facilmente intuibile, l'utilizzo di applicazioni AR in campo medico richiede assoluta precisione e affidabilità nella registrazione dei modelli. Imprecisioni anche minime nell'allineamento e nella sincronizzazione di elementi reali e virtuali, dovute a errori statici o dinamici, possono avere conseguenze molto gravi per il paziente. Pertanto, le apparecchiature mediche basate su tecniche di augmented reality sono spesso molto costose, e conseguentemente di diffusione ancora limitata.

2.2 Applicazioni militari

L'applicazione forse più immediata dell'augmented reality in campo militare è l'utilizzo di HMD, prevalentemente di tipo optical see-through, in dotazione ai singoli soldati. Soprattutto in ambienti complessi, quali campi di battaglia urbani, tali sistemi AR consentono di migliorare la *situational awareness* (SA) delle forze dispiegate sul territorio e garantire una maggiore efficienza fornendo informazioni addizionali sull'ambiente circostante, quali planimetrie degli interni di un edificio o informazioni aggiornate in tempo reale sulla posizione di cecchini nemici, su elementi di particolare interesse strategico, sulle condizioni e la posizione dei compagni di squadra o sulla direzione da seguire per raggiungere un determinato obiettivo.

A differenza di mappe tradizionali, che distolgono l'attenzione del soldato, che non consentono di rappresentare intuitivamente la natura tridimensionale del terreno e sono difficilmente aggiornabili e sincronizzabili con rapidità tra gruppi differenti, le informazioni fornite da un sistema AR possono includere contenuti multimediali di vario genere, sono integrate con il cono visivo dei singoli soldati, e le informazioni sono direttamente gestibili e sincronizzabili anche tra gruppi separati da grandi distanze mediante trasmissioni wireless dirette o via satellite.

Un esempio concreto di sistema militare basato su AR è *iARM* (Intelligent Augmented Reality Model), attualmente in via di sviluppo da parte di Tanagram Partners per conto dell'agenzia governativa statunitense DARPA (Defense Advanced Research Projects Agency). Il sistema utilizza geolocation e triangolazione per fornire immagini AR sincronizzate e aggiornate in tempo reale a soldati dotati di HMD e schierati sul campo di battaglia, coerentemente con il loro cono visivo, aumentando la loro SA collettiva e facilitando lo scambio di informazioni tattiche.

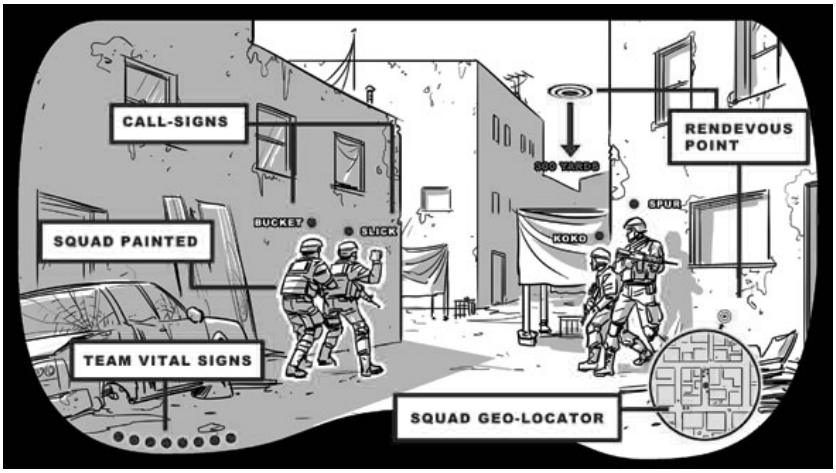

Figura 2.2: Concept art che mostra la visione aumentata della realtà da parte di un soldato dotato di HMD grazie a iARM [TAN10]

Come visibile in figura, iARM fornisce informazioni aggiuntive quali la posizione e lo stato di salute dei compagni di squadra (le cui silhouette sono riconosciute ed evidenziate in tempo reale), un indicatore per la posi-

zione dell'obiettivo (in questo caso, un punto di rendezvous), ed una mappa della zona circostante che comprende sia la posizione dei compagni di squadra che quella dell'obiettivo.

Il sistema si occupa inoltre di evidenziare con colori diversi le silhouette di civili e soldati nemici, visualizza planimetrie bidimensionali o tridimensionali e, tramite comandi vocali e gesture recognition, consente di evidenziare parti dello scenario per segnalare pericoli o punti di interesse ai compagni di squadra. Ulteriori informazioni possono inoltre essere ricevute in tempo reale dal *TOC* (tactical operations center), o ricavate mediante l'uso di droni, aeromobili a pilotaggio remoto di piccole dimensioni in grado di fornire immagini aeree della zona.

Sistemi basati su AR con l'obiettivo di aumentare la SA dei soldati sono inoltre in via di integrazione su diversi modelli di aerei militari, sfruttando il parabrezza del cockpit come display oppure il casco del pilota come HMD. Nel secondo caso, si parla più propriamente di *helmet-mounted display*. Il caccia monoposto Lockheed Martin F-35 Lightning II, ad esempio, integra sensori in grado di tracciare la posizione degli aerei nemici, anche quando essi si trovano al di fuori del campo visivo del pilota, e ne visualizza la direzione in semitrasparenza sulla visiera del casco, consentendo allo stesso di mirare verso il bersaglio da qualsiasi angolazione.

Ulteriori applicazioni dell'AR nel settore includono sistemi utilizzati per assistere meccanici dotati di HMD nella riparazione e nella manutenzione di apparecchiature e veicoli militari. Esempio concreto di tale applicazione è il progetto *ARMAR* (Augmented Reality for Maintenance and Repair), sviluppato dal Computer Graphics & Interface Research Lab della Columbia University, dipartimento diretto dal già citato Steven Feiner [FEI06]. ARMAR sovrappone etichette e schematiche alle apparecchiature, e consente di visualizzare informazioni in maniera sensibile al contesto grazie a tecniche di gesture recognition ed alla sovrapposizione di pulsanti virtuali interattivi. Questi ultimi emulano un'interfaccia di tipo touch screen tracciando la posizione delle mani dell'utente e fornendo feedback aptico passivo in modo visuale, ovvero mostrando graficamente l'interazione con i pulsanti senza richiedere periferiche dedicate.

2.3 Applicazioni ingegneristiche

Le applicazioni dell'augmented reality nell'ingegneria sono molteplici, ed applicabili ad un numero elevato di discipline ingegneristiche nonostante l'elevata differenziazione e specializzazione delle stesse.

Soprattutto in fase di progettazione, infatti, la possibilità di visualizzare un modello tridimensionale del progetto, sovrapposto alla realtà circostante, consente di eliminare i costi ed i tempi necessari per la costruzione di prototipi. Ciò è particolarmente utile per favorire la collaborazione di gruppi distribuiti, i cui membri si trovano fisicamente in luoghi differenti: il prototipo virtuale è aggiornato in tempo reale ed ogni modifica effettuata da un progettista viene istantaneamente trasmessa all'intero team. Il funzionamento del prototipo può inoltre essere dimostrato al management mediante opportune simulazioni AR, i cui elementi virtuali sono visualizzati tramite HMD o proiettori digitali (utilizzando le tecniche di Spatial AR descritte brevemente nel primo capitolo), senza richiedere la costruzione o l'impiego di ulteriori dispositivi fisici.

Un potente strumento utilizzato nella progettazione sono gli ambienti *CAVE* (acronimo ricorsivo per CAVE Automatic Virtual Environment), ovvero stanze di forma prevalentemente cubica sulle cui pareti vengono proiettate, con tecniche SAR, immagini generate dal computer. Il nome CAVE è anche un riferimento al mito della caverna di Platone (La Repubblica, libro VII), in quanto le immagini visualizzate al suo interno, così come le ombre percepite dai prigionieri incatenati nell'allegoria del filosofo, non sono effettivamente oggetti reali ma semplicemente una loro proiezione.

Per la loro natura di ambiente limitato e controllato, tali ambienti si prestano bene per la realizzazione di simulazioni, sia puramente in realtà virtuale che in augmented reality e SAR. L'utente tipicamente indossa un HMD per visualizzare in 3D stereoscopico gli elementi virtuali non proiettati in SAR, ed il sistema ne traccia i movimenti con tecniche di motion capture basate sull'utilizzo di sensori elettromagnetici presenti nell'ambiente, in modo da generare immagini consistenti con la posizione ed il campo visivo dello stesso. Altoparlanti posizionati ad angolazioni differenti e

controllati dal software, inoltre, consentono di fornire all'utente feedback audio in 3D consistente con la sua posizione e lo stato della simulazione. Ovviamente, data la precisione necessaria per tracciare l'utente, è spesso necessario calibrare i sensori prima dell'inizio delle sessioni vere e proprie.

Figura 2.3: Utente dotato di HMD all'interno di un ambiente CAVE

L'ambiente imseCave [IMS05] è un esempio di CAVE a basso costo ed elevate performance sviluppato dal dipartimento di Industrial and Manufacturing Systems Engineering dell'Università di Hong Kong, utilizzato per progettazione ed esecuzione di simulazioni di vario genere grazie alla dotazione di diversi dispositivi di input/output.

Una delle applicazioni principali di imseCave è *MagicPad*, framework tangibile per interfacce utente che consiste semplicemente di un notepad, su cui vengono proiettate immagini AR, ed una penna ad infrarossi, mediante la quale è possibile manipolare liberamente il contenuto di tali immagini in uno spazio 3D. Non sono necessari HMD in quanto il notepad agisce da display, e la sua posizione e angolazione nello spazio sono tracciate in tempo reale da un sistema ottico per garantire la correttezza delle

proiezioni, dando l'illusione di un'immagine effettivamente fissata su di esso indipendentemente dai suoi movimenti. Notepad e penna sono strumenti leggeri e facilmente trasportabili, utilizzabili in modo intuitivo anche da utenti che non dispongono di specifiche conoscenze in campo AR, il che garantisce risultati positivi in termini di curva di apprendimento. L'utilizzo di proiettori, tuttavia, limita l'effettiva risoluzione delle immagini visualizzabili sul notepad, soprattutto ad angoli di incidenza elevati. Può inoltre comportare la visualizzazione di immagini sfocate a seconda della distanza dal proiettore, e la presenza di eventuali ostacoli, incluso talvolta il corpo dell'utente stesso, può occludere parti della proiezione.

2.4 Marketing

Nel campo del marketing, se si considera la definizione in senso lato di AR data nel primo capitolo, le applicazioni sono molteplici. Qualsiasi tipo di sovraimpressione, di immagini sovrapposte in green screen o persino di effetti speciali generati al computer, infatti, rientra in tale definizione. Restringendo il campo alle applicazioni dell'AR in senso stretto compatibilmente con la definizione di Azuma, tuttavia, sebbene il numero di applicazioni attuali si riduca notevolmente, lo stesso non avviene alla loro efficacia come strumento di marketing.

Semplici applicazioni AR come visualizzatori di modelli tridimensionali, infatti, possono tradursi in ottimi strumenti pubblicitari se utilizzati per visualizzare, sovrapposti al mondo reale, prodotti come automobili, o persino personaggi di fantasia nel caso di brand indirizzati ai bambini. L'utente può osservare il modello 3D da ogni angolazione, con tanto di eventuali animazioni o semplici possibilità di interazione, ed è intuibile come far comparire di fronte all'utente degli elementi virtuali che sembrano reali, soprattutto se non esistenti nella realtà come nel caso dei personaggi di fantasia, possa lasciare facilmente impressionati.

La possibilità di visualizzare animazioni sulle copertine di libri o CD musicali, di vedere carte collezionabili far comparire creature fantastiche sulla propria scrivania, o di guardare concerti della propria band preferita

all'interno di un pacchetto di patatine (come nella campagna pubblicitaria *Doritos Late Night*, che ha legato la band americana dei Blink 182 al produttore di tortilla chips Doritos), infatti, è di evidente appeal e di semplice implementazione grazie alle tecnologie attuali.

Figura 2.4: Bug Juice, esempio di videogame AR promozionale [BUG10]

Bug Juice è un progetto del Qualcomm Augmented Reality Game Studio del Georgia Tech, una collaborazione sovvenzionata da Qualcomm tra l'Augmented Environments Lab del Georgia Tech ed il programma di Interactive Design and Game Development del Savannah College of Art and Design. Si tratta di un semplice videogame realizzato come esempio di applicazione AR promozionale per accompagnare un differente prodotto di consumo, in questo caso del latte. Il programma, sviluppato con le librerie AR di Qualcomm ed il motore grafico Unity3D, viene eseguito su un cellulare Android ed è in grado di riconoscere l'intera confezione del latte come marker tridimensionale, per sovrapporre su di essa i componenti del gioco: del fogliame, da cui emergono piccoli insetti che il giocatore deve incenerire prima che possano raggiungere l'apertura della confezione, rovinandone il contenuto. Il consumatore che si trova al supermercato è quindi incoraggiato ad acquistare il prodotto per poter usufruire del videogame, e la possibilità di utilizzare tale gioco per intrattenere i figli du-

rante la spesa, ancor prima dell'acquisto, può essere un ulteriore incentivo per la scelta del prodotto al posto di un marchio concorrente.

2.5 Intrattenimento

Il campo dell'intrattenimento rivela molti punti in comune con il marketing. Come già menzionato nel paragrafo precedente, infatti, le applicazioni sono molteplici - e tutto sommato simili - se si considera una definizione più lasca di AR, ma rimangono ugualmente efficaci, seppur inferiori in numero, utilizzando una definizione più stringente. Similmente, visualizzatori di modelli tridimensionali possono essere utilizzati per arricchire l'esperienza delle simulazioni nei parchi dei divertimenti, o per dare vita ad action figure e farle combattere tra di loro, e diverse tecnologie AR possono essere incorporate con facilità nei videogames.

E sono proprio i videogiochi ad essere probabilmente l'applicazione più diffusa, e più in rapida espansione, dell'AR in questo campo. Il primo videogame AR di successo è probabilmente *EyeToy: Play* (2003) del Sony Computer Entertainment (SCE) London Studio per la console Sony PlayStation 2, ovvero una raccolta di semplici giochi che utilizzavano una telecamera digitale USB, denominata *EyeToy*, per sovrapporre elementi virtuali al flusso video, ed utilizzavano algoritmi di computer vision per riconoscere il giocatore nel video e permettergli di interagire con tali elementi. EyeToy soffriva tuttavia di rilevanti limitazioni tecnologiche, soprattutto la risoluzione del sensore limitata a 320x240 pixel e la necessità di giocare in un ambiente ben illuminato. Tali limitazioni sono state poi alleviate nel successore di tale periferica, *PlayStation Eye*, lanciato nel 2007 insieme al videogame *The Eye of Judgement*, gioco di carte collezionabili alle quali venivano sovrapposti modelli tridimensionali delle creature fantastiche rappresentate da ogni carta. Il gioco tracciava le carte posizionate fisicamente su una griglia, consentendo al giocatore di utilizzarle come unico input per sfidare il computer, o altri avversari online.

Negli anni successivi, l'introduzione di dispositivi portatili dotati di webcam e sensori di movimento integrati ha contribuito allo sviluppo di

un numero crescente di progetti basati su tecnologie AR, ma non ha ancora consentito ad esse di raggiungere realmente il mercato di massa. La presenza della suite *AR Games* di Nintendo tra il software preinstallato sulla (recentemente introdotta) console portatile Nintendo 3DS, tuttavia, potrebbe facilitare il raggiungimento di tale obiettivo. Come successore del Nintendo DS, piattaforma in grado di raggiungere una base installata di 149 milioni di unità (dato aggiornato al 30 settembre 2011), il 3DS ha, come già detto, il potenziale per far giungere a breve termine applicazioni augmented reality nelle mani di un numero elevato di utenti.

Figura 2.5: Effetti di deformazione in tempo reale in AR Games [NIN11]

AR Games, in particolare, consiste in una serie di minigiochi che utilizzano dei marker simili a carte da gioco per sovrapporre elementi virtuali alla realtà. Alcuni di essi implementano degli effetti di deformazione in tempo reale alla scena catturata. La scrivania mostrata in figura può trasformarsi in uno stagno dalla superficie increspata dalle onde, in una catena montuosa, oppure ancora può mostrare dei buchi non presenti nella realtà, ed al cui interno sono nascosti oggetti virtuali. La scena è ripresa mediante le due telecamere presenti sul lato esterno del Nintendo 3DS, e visualizzata in 3D, senza bisogno di indossare occhiali di alcun genere, sullo schermo autostereoscopico a parallax barrier della console.

2.6 Altre applicazioni consumer

In generale, l'AR ben si presta all'integrazione in semplici applicazioni rivolte ad un'utenza generale di tipo consumer, in quanto consente di aumentare la quantità di informazione presente, di semplificare la presentazione dei dati tramite interfacce utente intuitive, e di migliorare complessivamente l'user experience.

Il mercato per tali applicazioni è in rapida crescita, ma prevalentemente limitato a dispositivi mobili come smartphone, tablet e console per videogiochi. Le piattaforme iOS e Android, in particolare, offrono un ampio portfolio di software AR, sia gratuito che a pagamento. Nei prossimi capitoli, dopo un'introduzione generale alla struttura ed alle caratteristiche della piattaforma Android, ci si soffermerà dettagliatamente sulle tecnologie e sulle applicazioni AR disponibili per quest'ultima.

Capitolo 3

Introduzione ad Android

Figura 3.1: Android Robot, logo del sistema operativo

Android è un sistema operativo per dispositivi mobili, come tablet e smartphone, progettato da Android Inc., azienda specializzata nello sviluppo di software per il mercato mobile acquisita da Google nel 2005. Il 5 Novembre 2007, Android viene annunciato da Google insieme alla fondazione di Open Handset Alliance (OHA), consorzio di (attualmente) 84 compagnie di tecnologia e telecomunicazioni formato al fine di sviluppare standard aperti per i dispositivi mobili e di, citandone la missione aziendale, accelerare l'innovazione nel campo mobile ed offrire ai consumatori un'esperienza più ricca, meno costosa, e in generale migliore. [OHA07]

I sorgenti di Android sono rilasciati interamente sotto licenze open source. Il kernel, derivato da Linux 2.6 e modificato architetturalmente al punto di perdere compatibilità con applicazioni e librerie Linux standard,

è rilasciato sotto licenza GPLv2. Il resto del codice, strutturato a stack con vari livelli di astrazione, è invece rilasciato sotto licenza Apache License 2.0. Tipicamente lo sviluppo delle porzioni di codice GPL è pubblico e gestito da OHA, mentre lo sviluppo delle porzioni Apache License è gestito privatamente e rilasciato al pubblico insieme alle release principali del sistema operativo. Al momento della stesura di questa tesi, l'ultima release principale di Android è la 4.0 (denominata Ice Cream Sandwich).

Sebbene l'OS sia open source, per poter rilasciare commercialmente un dispositivo che usi il trademark Android è necessario che questo soddisfi le specifiche, sia dal punto di vista hardware che software, definite nel Compatibility Definition Document (CDD) periodicamente rilasciato da Google in occasione di ogni release principale. Tali specifiche, che includono la compatibilità con le API e gli standard supportati, e la presenza di applicazioni fondamentali e componenti hardware di base, garantiscono la conformità ad un set di caratteristiche standard per ogni cellulare o tablet Android, e permettono di considerare tali dispositivi come una piattaforma hardware e software unica, con software applicativo cross-compatibile e potenzialità simili nonostante le differenze tra i vari modelli, rilasciati da produttori differenti e destinati a diverse fasce di mercato.

3.1 Architettura software

Lo stack software che funge da struttura per il sistema operativo, i cui componenti principali sono dettagliati in figura, include il kernel Linux al livello di astrazione più vicino all'hardware, per servizi di sistema fondamentali quali sicurezza, gestione dell'energia, dei processi e della memoria a basso livello, stack di rete e driver delle periferiche di input/output.

Al livello di astrazione immediatamente superiore, si trovano le librerie principali. Le librerie grafiche 2D (custom) sono denominate SGL, mentre quelle 3D sono basate su OpenGL ES, 1.0 nelle prime versioni e 2.0 in quelle successive, con accelerazione hardware opzionale. L'archiviazione dei dati è gestita mediante un database relazionale SQLite, ed il rendering del testo, sia bitmap che vettoriale, mediante FreeType. Le altre librerie di

base includono il supporto al protocollo SSL per la sicurezza, un'implementazione delle librerie di sistema C (libc), derivata da BSD e ottimizzata per sistemi embedded basati su Linux, e librerie multimediali per la riproduzione e lo streaming di file multimediali in diversi formati.

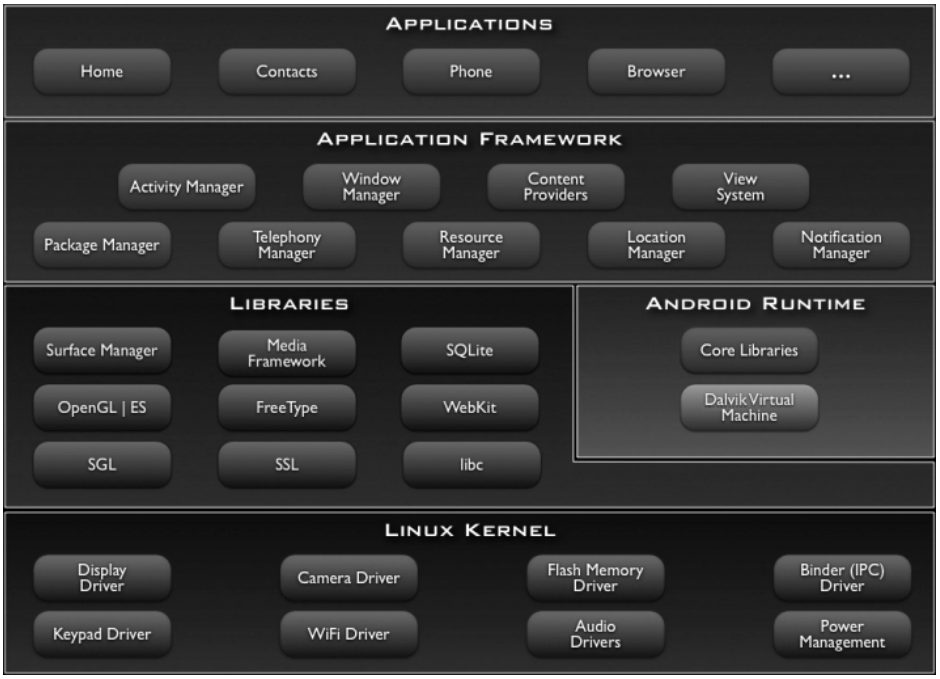

Figura 3.2: Architettura del sistema operativo [GOO07]

Il livello di astrazione denominato framework applicativo (*application framework*), immediatamente superiore alle librerie, si frappone tra queste ultime ed il livello più elevato, le applicazioni vere e proprie, ed offre un'interfaccia per consentire al software, eseguito concorrentemente in multitasking reale, di accedere alle funzionalità di livello inferiore. Android include, come già detto, una serie di applicazioni fondamentali per dispositivi mobili (tra cui un browser basato su WebKit, una rubrica, un sistema di messaggistica e un calendario) e consente all'utente di installare opzionalmente software addizionale, gratuito o a pagamento.

3.2 Esecuzione delle applicazioni

Mentre i livelli sottostanti sono scritti in C, le applicazioni sono scritte in Java ed eseguite mediante una macchina virtuale open source, Dalvik, con architettura register-based e compilazione just-in-time. Il bytecode contenuto nei file .class viene convertito in formato eseguibile .dex (*Dalvik Executable*), ottimizzato per l'esecuzione su dispositivi con limitazioni in termini di memoria e prestazioni, mediante un tool denominato *dx*. Il codice ed eventuali risorse o dati aggiuntivi vengono poi compressi in un unico file .apk (*Android package*), utilizzato per l'installazione.

Ogni applicazione è eseguita come processo separato con una propria istanza della macchina virtuale, grazie al ridotto footprint della stessa e dell'eseguibile, in modo da garantire una completa separazione tra processi concorrenti. Tale approccio è denominato *security sandbox*: le applicazioni sono isolate e indipendenti le une dalle altre, e hanno accesso a un sottoinsieme limitato e controllato di dati e risorse, per garantire sicurezza nell'accesso a dati sensibili e tolleranza ai guasti in caso di malfunzionamento. Ciò è possibile grazie al supporto multiutente del kernel Linux: ogni applicazione è gestita come se fosse un utente diverso a cui è associato un proprio user ID, accessibile esclusivamente dal sistema operativo e non conosciuto dall'applicazione stessa, ed i permessi di ogni file da essa utilizzato vengono settati in modo che solo ed esclusivamente tale user ID possa accedervi. Il processo, inoltre, viene avviato soltanto quando la sua esecuzione è necessaria e termina immediatamente non appena smette di esserlo, o quando il sistema ha bisogno di recuperare memoria per gli altri processi eventualmente in esecuzione. In questo modo Android implementa il *principio del privilegio minimo*, ovvero rende disponibili ad ogni applicazione esclusivamente i componenti immediatamente necessari al proprio funzionamento e impedisce l'accesso alle parti del sistema per cui essa non dispone di privilegi sufficienti.

La condivisione di dati e risorse tra più applicazioni differenti è comunque possibile tramite meccanismi che consentono ad esse di condividere lo stesso user ID, e quindi gli stessi privilegi sui file. In tal caso, le

applicazioni possono essere eseguite con lo stesso processo Linux e condividere la stessa macchina virtuale, ma purchè ciò sia possibile è necessario che esse siano tutte firmate digitalmente con lo stesso certificato. Inoltre, i dati di sistema del dispositivo (che includono i contatti della rubrica, i messaggi SMS, i dati immagazzinati nelle memorie di massa, i file multimediali, eccetera) sono accessibili alle applicazioni soltanto dietro esplicita autorizzazione da parte dell'utente al momento della loro installazione.

3.3 Struttura a componenti delle applicazioni

A differenza delle normali applicazioni Java, su Android non esiste un singolo entry point, ovvero un punto di ingresso equivalente al metodo main, per l'avvio e l'inizializzazione del software.

Le applicazioni sono infatti strutturate in componenti pilotati ad eventi (*event driven*), e in base alla natura della specifica applicazione è possibile avere entry point diversi in base all'evento catturato dal sistema. In generale, il funzionamento dell'applicazione è definito dall'insieme dei suoi componenti, ognuno dei quali esiste come entità indipendente, ha un proprio ciclo di vita, e svolge un ruolo ben specifico.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
    <application android:icon="@drawable/app_icon.png" ... >
        <activity android:name="com.example.project.ExampleActivity"
                  android:label="@string/example_label" ... >
        </activity>
        ...
    </application>
</manifest>
```

Figura 3.3: Dichiarazione di un componente nel file Manifest

I componenti vengono dichiarati all'interno di AndroidManifest.xml (il file *Manifest* dell'applicazione), come visibile nella figura di esempio tratta dalla documentazione ufficiale Android. Dopo la dichiarazione, è possibile avviare tali componenti mediante un oggetto *Intent*, ovvero un

messaggio asincrono che li lega insieme a runtime, definisce le azioni che essi devono effettuare e, più in generale, ne consente l'interazione. Ad esempio, un'applicazione può inviare un intent per richiedere ad un componente i dati di un contatto della rubrica, e quest'ultimo restituirà a sua volta un intent contenente un URI (uniform resource identifier) che punta ai dati relativi al contatto desiderato.

E' possibile classificare i componenti delle applicazioni Android in quattro tipologie, ciascuna con ruoli e caratteristiche differenti:

3.3.1 Attività

Un'attività rappresenta una singola schermata dotata di un'interfaccia utente. Un'applicazione che consiste di più schermate diverse, come ad esempio la lista dei contatti ed i dettagli di ogni singolo contatto in un'applicazione che gestisce la rubrica del cellulare, utilizzerà un'attività distinta, e indipendente, per ciascuna schermata. Le applicazioni esterne sono in grado di avviare specifiche attività in base alla loro funzione: nel già citato esempio della rubrica, un'applicazione esterna potrà quindi avviare l'attività corrispondente alla lista dei contatti oppure quella corrispondente ai dettagli di una singola persona, in base alle necessità effettive, e riceverne i dati come se l'attività facesse parte dell'applicazione stessa, in maniera del tutto trasparente all'utilizzatore.

3.3.2 Servizi

Un servizio è un componente eseguito in background per operazioni di lunga esecuzione, o per l'esecuzione di processi remoti, che non dispone di un'interfaccia utente e non è pertanto direttamente visibile all'utilizzatore. Esempi di servizi includono la ricezione dei pacchetti nelle comunicazioni di rete senza causare attese per l'utente, che nel frattempo può continuare ad interagire con l'attività principale dell'applicazione, oppure la riproduzione di file musicali in background anche quando l'utente sta visualizzando l'interfaccia di un'altra applicazione in esecuzione concorrente. I servizi possono essere avviati da altri componenti, principal-

mente attività, e se necessario queste ultime possono legarsi ad essi per continuare a interagirvi, e scambiare dati, anche dopo la loro esecuzione.

3.3.3 Content Provider

I content provider (provider di contenuti) gestiscono in lettura e scrittura un set condiviso di dati applicativi, memorizzati all'interno del file system, in un database SQLite, sul web, o in qualsiasi altra locazione accessibile dalle applicazioni. Si occupano quindi di fornire alle applicazioni i dati necessari alla loro esecuzione e, se queste ultime dispongono di privilegi sufficienti ed il provider lo consente, di modificarli e salvarli. Nel precedente esempio di gestione della rubrica, il sistema operativo mette a disposizione dell'applicazione un content provider che si occupa di gestire i dati dei contatti, e che può fornire ad essa, se autorizzata, le informazioni necessarie per visualizzarli o modificarli. I content provider possono anche essere specifici per l'applicazione, e limitarsi a gestire i dati da essa utilizzati senza condividerli esternamente. In tal caso, sarà esclusivamente l'applicazione di cui essi fanno parte a potervi accedere. Per consentire l'utilizzo di un content provider da parte di applicazioni esterne, è necessario implementare un apposito set standard di API per gestire tali transazioni. Per garantire maggiore sicurezza, inoltre, i content provider non ricevono intent direttamente da altri componenti. Questi ultimi devono invocare i metodi di un oggetto separato, denominato *ContentResolver*, che funge da livello di astrazione aggiuntivo e si occupa di gestire le transazioni al loro posto.

3.3.4 Broadcast Receiver

Un broadcast receiver (ricevitore di broadcast) è un componente che si occupa di rispondere a particolari eventi, chiamati *broadcast announcement*, non destinati ad una singola applicazione ma a tutte quelle attualmente in esecuzione. Esempi di broadcast announcement includono lo spegnimento dello schermo, un basso valore di carica per la batteria dell'apparecchio, oppure lo scatto di una fotografia, e sono concettualmente

simili a degli interrupt. Anche le singole applicazioni possono iniziare dei broadcast announcement, ad esempio per segnalare il completamento di un'operazione o l'avvenuto download di un file. Sebbene i broadcast receiver non dispongano di una propria interfaccia grafica, questi possono generare delle notifiche sulla status bar del sistema, ed avviare attività o servizi per la logica necessaria alla gestione dell'evento ricevuto.

3.4 Software development kit

Lo starter package dell'SDK (software development kit) per Android, liberamente scaricabile dal sito ufficiale per gli sviluppatori, include una serie di tool fondamentali, accessibili via script e linea di comando, oppure tramite un plugin denominato *ADT* (Android Development Tools) per Eclipse, l'IDE Java consigliata per sviluppare sulla piattaforma. E' ovviamente possibile utilizzare qualsiasi altro IDE o editor di testo, mantenendo l'accesso all'intero feature set del kit di sviluppo.

Una volta installato lo starter package, il tool *Android SDK and AVD Manager* (invocabile come *android* da linea di comando) consente il download e la gestione di componenti aggiuntivi, che includono le diverse versioni della piattaforma Android, ulteriori tool, progetti di esempio e documentazione. Il package di base include inoltre *Android Emulator*, tool in grado di emulare un dispositivo Android per testare l'applicazione come se questa fosse eseguita su un dispositivo reale, e *Android Debug Bridge*, ovvero un'interfaccia verso l'emulatore (o un dispositivo reale), per installare applicazioni o invocare comandi. Per ulteriori dettagli sull'SDK, si rimanda alla documentazione ufficiale.

Capitolo 4

Applicazioni AR per Android

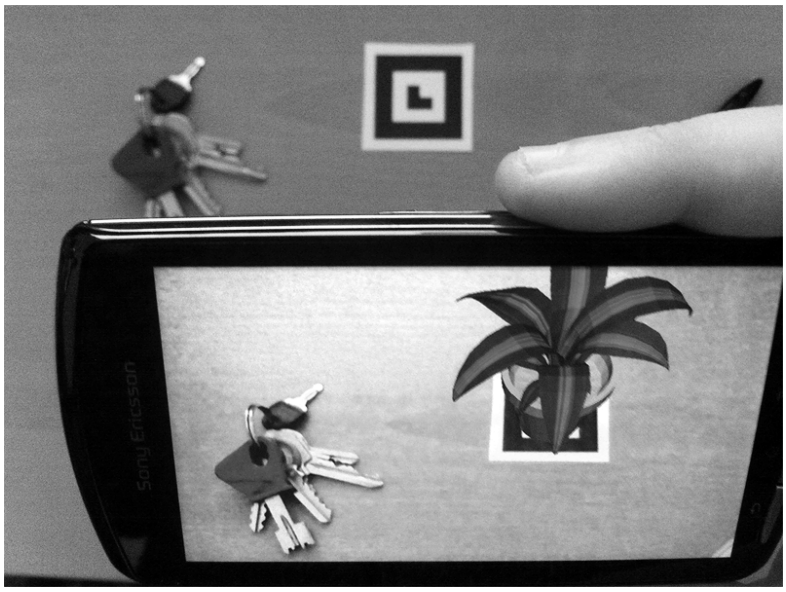

Figura 4.1: Augmented reality su Android

Una volta definite le caratteristiche, le applicazioni e le potenzialità delle tecnologie basate su augmented reality, ed una volta introdotta la piattaforma Android, è adesso possibile descrivere le applicazioni di tali tecnologie per quest'ultima, analizzandone l'hardware necessario, i framework utilizzati per semplificare lo sviluppo di software applicativo AR, ed alcuni esempi di tali applicazioni già disponibili per la piattaforma. In particolare, ci si soffermerà sul browser augmented reality Layar, la cui

struttura general purpose consente di sviluppare estensioni di vario genere e con finalità differenti, mediante semplici tecnologie web quali PHP e MySQL. Si proporrà inoltre un esempio concreto di estensione, con tanto di indicazioni per una sua possibile implementazione come applicazione stand-alone tramite gli strumenti descritti in questo capitolo.

4.1 Hardware utilizzato su piattaforma Android

Nonostante sia stata già fatta una panoramica generale sull'hardware comunemente utilizzato per le applicazioni AR nel primo capitolo, i dispositivi presenti nel Compatibility Definition Document meritano senza dubbio un approfondimento particolare per comprendere appieno il funzionamento del software, su cui si concentreranno i paragrafi successivi.

Per questo elaborato, si è scelto di prendere come riferimento la seconda revision del CDD di Android 4.0 [GOO12], datata 25 gennaio 2012.

4.1.1 Display

Il display è usato sia come dispositivo di input, in quanto touchscreen (normalmente di tipo capacitivo, dotato di funzionalità multitouch), che come dispositivo di output predefinito su cui mostrare sistema operativo e applicazioni in esecuzione. Da specifiche, ha dimensione minima di 2.5 pollici, densità di pixel compresa tra 120 e 320 dpi, ed aspect ratio (rapporto tra larghezza e altezza) variabile tra 4:3 e 16:9 (widescreen). A seconda dell'orientamento verticale del dispositivo può funzionare in modalità portrait (verticale) o landscape (orizzontale), e le applicazioni possono supportare solo una o entrambe le modalità.

4.1.2 Telecamera

Il dispositivo deve includere una telecamera rivolta esteriormente, e può includerne una seconda rivolta interiormente. La prima deve avere una risoluzione almeno pari a 2 MPixel e auto-focus hardware, la seconda

una risoluzione almeno pari a 640x480px (VGA). Nelle applicazioni AR, la telecamera esterna è quella solitamente usata per catturare il video, in quanto la telecamera interna è rivolta verso l'utente stesso. Per utilizzare la telecamera è necessario richiedene i permessi nel file Manifest.

4.1.3 Sensori

Android 4.0 include API per interagire con gli input forniti da diversi tipi di sensori. L'inclusione dei principali sensori normalmente usati nelle applicazioni AR, descritti qui di seguito, è fortemente consigliata nel CDD.

- *Accelerometro*, in grado di percepire l'accelerazione sui tre assi XYZ, aggiornato con frequenza pari o superiore a 50 Hz e in grado di registrare accelerazioni almeno pari a 2g su ogni asse. Fornisce tre valori interi, uno per asse, all'interno di un array *SensorEvent.values*, e viene usato per tracciare il movimento traslazionale del dispositivo.

- *Giroscopio*, in grado di percepire la rotazione intorno ai tre assi XYZ, aggiornato con frequenza pari o superiore a 100 Hz e in grado di registrare rotazioni almeno pari a 5.5π radianti al secondo. Fornisce tre valori interi, uno per asse, all'interno di un array *SensorEvent.values*, e viene usato per tracciare il movimento rotazionale del dispositivo.

- *Ricevitore GPS*, in grado di fornire le coordinate di latitudine e longitudine del dispositivo, aggiornate periodicamente ed accessibili dalle applicazioni tramite un *LocationProvider*. Se il ricevitore è presente nel sistema, è fortemente consigliata l'inclusione di una modalità *Assisted GPS*, che sfrutti una connessione dati per minimizzare i tempi di lock-on GPS e ridurre le imprecisioni dovute alla presenza di ostacoli verticali, come ad esempio edifici, per il segnale GPS.

- *Magnetometro*, in grado di percepire l'orientamento nei tre assi XYZ, aggiornato con frequenza pari o superiore a 10 Hz e in grado di reagire correttamente al campo magnetico terrestre. Fornisce tre valori

interi, uno per asse, all'interno di un array *SensorEvent.values*, e viene usato come bussola digitale. In congiunzione al ricevitore GPS, consente di tracciare la posizione e l'orientamento del dispositivo, e quindi dell'utente, nello spazio. Questa funzionalità è particolarmente utile per svariate applicazioni AR, ed è accessibile tramite le API fornite dal servizio di sistema *LocationManager*.

4.2 Framework AR principali

Esistono numerosi framework progettati per semplificare lo sviluppo di applicazioni augmented reality su piattaforme Android, tramite librerie e interfacce di base per il riconoscimento dei marker, il rendering e la sovrapposizione di elementi virtuali al video, o persino comprendenti engine tridimensionali completi e dotati di simulazioni fisiche per gestire realisticamente tali elementi all'interno di una scena 3D complessa. Molti di questi framework, però, si trovano ancora in uno stadio di sviluppo piuttosto primitivo, sono spesso porting incompleti di progetti rilasciati precedentemente per altre piattaforme, possono soffrire di bug anche rilevanti, e sono generalmente piuttosto carenti in termini di documentazione, progetti di esempio e supporto per lo sviluppatore. Ai fini di questa tesi, dunque, si è scelto di soffermarsi su tre esempi in particolare, che si distinguono rispettivamente per diffusione, semplicità d'uso e completezza.

4.2.1 NyARToolkit for Android

NyARToolkit [NYA12] è un'implementazione del già citato framework ARToolKit (basata sulla versione 2.72.1) a cura dello sviluppatore giapponese Nyatla, e consiste di una serie di librerie utili per lo sviluppo di applicazioni AR vision-based, ovvero che usino tecniche di object recognition per il riconoscimento di marker all'interno di un flusso video. Sono inoltre incluse funzionalità per il rendering 3D degli oggetti virtuali, e per l'integrazione in tempo reale di questi ultimi con il video catturato.

Distribuito sotto licenza gratuita GPLv3 e, su richiesta, anche sotto licenza commerciale, il framework è disponibile anche per C++, C#, Java, Processing, Silverlight e ActionScript 3. NyARToolkit for Android, nello specifico, è scritto in Java ed il progetto è attualmente mantenuto su SourceForge dalla comunità di sviluppatori nata intorno alla libreria.

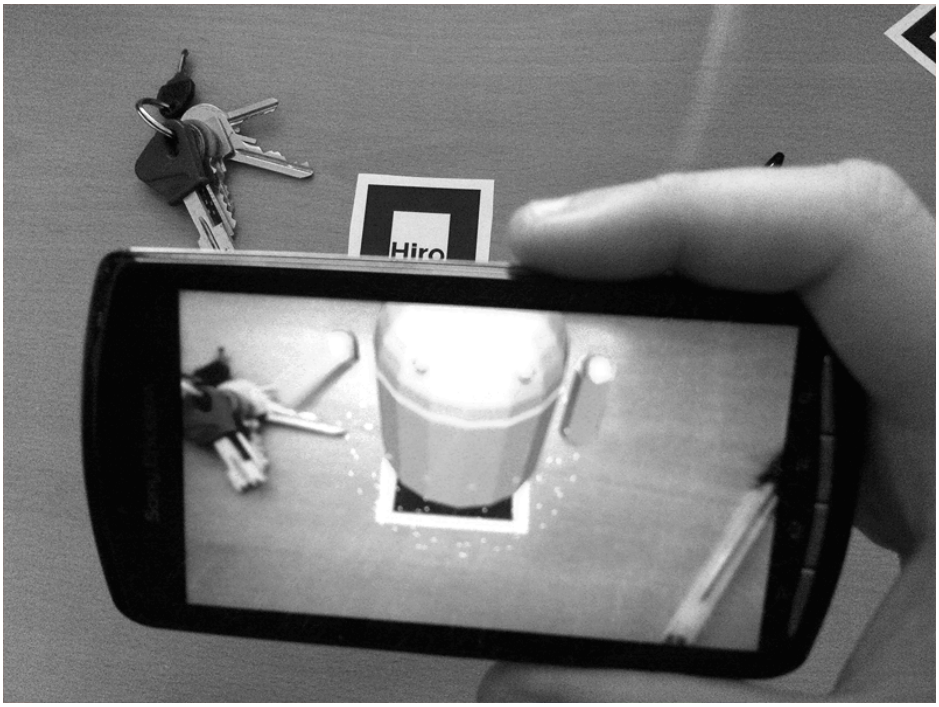

Figura 4.2: Sovrapposizione in tempo reale di un modello 3D ad un marker bidimensionale riconosciuto tramite NyARToolkit

I marker utilizzati sono bidimensionali, di forma quadrata, in bianco e nero, e caratterizzati da spessi bordi neri intorno all'immagine da riconoscere (tipicamente testo, pattern di pixel equivalenti a codici a barre bidimensionali, o ideogrammi giapponesi), in modo da facilitarne l'identificazione all'interno di scene complesse, che possono includere pattern similari. La documentazione ed i commenti all'interno dei sorgenti della libreria sono quasi interamente in giapponese, ma ciò, insieme alla popolarità della tecnologia in terra nipponica, non ha impedito a NyARToolkit di diventare rapidamente uno dei framework AR più diffusi per Android.

4.2.2 AndAR - Android Augmented Reality

Il progetto AndAR [DOM09], sviluppato dal tedesco Tobias Domhan e parte del programma MFG WisTa della divisione MFG Innovation della Media and Film Society del Baden-Württemberg, è anch'esso basato sul framework ARToolKit, e consente di riconoscere marker all'interno di un flusso video. Similmente, la libreria è rilasciata sotto licenza gratuita GPL, offre la possibilità di acquistare una licenza commerciale, ed è scritta in Java. A differenza di NyARToolkit, però, AndAR è disponibile esclusivamente per piattaforme Android, comprende un maggior numero di progetti di esempio e offre documentazione in lingua inglese (inclusi i commenti all'interno dei sorgenti). Pertanto, risulta più facilmente approcciabile, ed utilizzabile in modo efficiente, per gli sviluppatori occidentali.

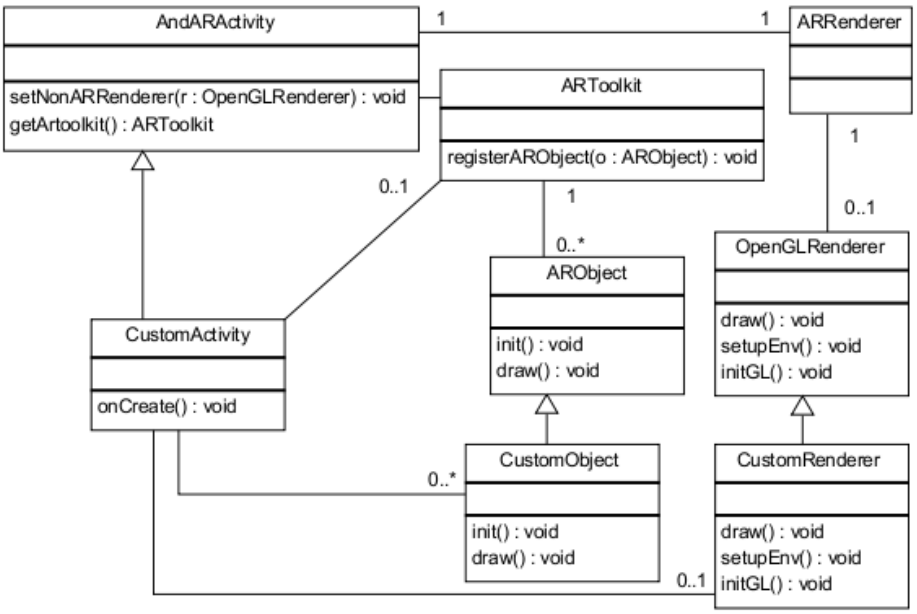

Figura 4.3: Architettura semplificata di un'applicazione basata sul framework AndAR [DOM09]

I marker utilizzati sono bidimensionali e sostanzialmente identici a quelli di NyARToolkit, ma AndAR offre un tool (mk_patt) per consentire l'identificazione di marker personalizzati. Il tool consente di creare un

file che descrive le caratteristiche del marker, da inserire all'interno del progetto, e di istruire la telecamera per il suo riconoscimento.

Come visibile in figura, il diagramma delle classi di un'applicazione basata su AndAR ha una struttura sorprendentemente semplice. L'attività *AndARActivity*, parte della libreria, si occupa già di gestire tutti gli elementi correlati all'augmented reality (gestione della videocamera, riconoscimento dei marker, visualizzazione del video), e può essere facilmente estesa dallo sviluppatore a seconda delle necessità dell'applicazione. Per associare un oggetto ad un marker, è necessario definire l'oggetto in questione come sottoclasse di *ARObject* e registrarlo in un'istanza di *ARToolkit* (reperibile tramite AndARActivity). A questo punto l'applicazione sarà già in grado di riconoscere il marker e sovrapporgli l'oggetto associato seguendone le coordinate. E' inoltre possibile personalizzare il rendering degli oggetti virtuali mediante un renderer personalizzato che implementi l'interfaccia di libreria *OpenGLRenderer*, ma si tratta di una procedura del tutto opzionale, la cui utilità dipende dai fini dell'applicazione.

4.2.3 QCAR (Qualcomm AR)/Vuforia SDK

QCAR SDK [QUA10], recentemente ribattezzato commercialmente con il nome *Vuforia* ma tuttora denominato come tale all'interno del codice e della documentazione ufficiale, è un framework AR per Android e iOS sviluppato dalla compagnia di telecomunicazioni americana Qualcomm, produttrice di chipset utilizzati da numerosi modelli di cellulari e tablet Android, e rappresenta al momento la soluzione forse più avanzata e completa per lo sviluppo di applicazioni AR destinate a dispositivi mobili.

A differenza dei framework analizzati in precedenza, QCAR non si limita a riconoscere marker bidimensionali, ma implementa una superclasse più generale (denominata *Trackable*) di elementi individuabili e tracciabili (target) all'interno del flusso video. Da essa derivano tre sottoclassi:

- *Frame Marker*, ovvero dei marker bidimensionali simili ai precedenti, ma a colori e con un ID codificato tramite un pattern binario disposto intorno al bordo. Per il riconoscimento del marker, quindi,

è necessario che questo sia inquadrato interamente dalla telecamera. A differenza delle altre tipologie di target, non è possibile generare frame marker personalizzati: tutte le combinazioni binarie che codificano i 512 diversi ID utilizzabili sono già fornite insieme alla libreria, ed sistema è in grado di tracciare fino a cinque marker contemporaneamente.

- *Image Target*, ovvero delle immagini bidimensionali a colori, non limitate a semplici pattern o con bordi di specifici colori come i marker già esaminati, ma ugualmente identificabili anche all'interno di scene complesse. Una volta riconosciuta l'immagine, è possibile continuare a tracciarla anche se quest'ultima non è inquadrata completamente dalla telecamera. Come per i frame marker, è possibile tracciare fino a cinque image target contemporaneamente.

- *Multi Target*, ovvero oggetti tridimensionali le cui facce sono costituite da più image target bidimensionali. Grazie all'esistenza di relazioni spaziali fisse tra le facce, una volta riconosciuto un image target sarà possibile tracciare anche gli altri, in quanto la loro posizione ed il loro orientamento nello spazio saranno noti. L'applicazione è quindi in grado di riconoscere oggetti reali solidi (purchè con un numero di facce limitato), e ciò consente all'utente di interagire con l'applicazione manipolandoli direttamente, o di occludere realisticamente oggetti virtuali posizionati spazialmente alle spalle del multi target.

Regioni rettangolari degli image target possono inoltre essere utilizzate come pulsanti virtuali (Virtual Buttons), per consentire un'interazione più intuitiva. Ciò non avviene mediante il riconoscimento delle mani dell'utente ed il tracciamento della loro posizione nello spazio per determinare l'avvenuta pressione, ma semplicemente tramite occlusione: se il pulsante virtuale risulta coperto da un ostacolo, il sistema lo considererà premuto. Ciò consente di utilizzare altri oggetti reali per premere, o tenere premuti, pulsanti virtuali. Per limitare le eventualità di pressioni accidentali, il sistema ignora le pressioni registrate quando la telecamera è

in movimento, quando i pulsanti non sono inquadrati completamente, e quando sono presenti movimenti rapidi all'interno del flusso video.

Figura 4.4: In basso: Frame Marker con diversi ID. Da sinistra: Multi Target, Image Target, Image Target con pulsanti virtuali [QUA10]

E' possibile generare image target e multi target personalizzati utilizzando il *Target Management System*, una web application disponibile sul sito della Qualcomm. Per utilizzare l'applicazione, è necessario effettuare il login con un account per sviluppatori.

QCAR mette a disposizione dello sviluppatore quattro componenti di libreria, tutti singleton. *Camera* si occupa di gestire la telecamera, e consente di avviare o arrestare la cattura del flusso video. *Image Converter* si occupa di convertire i frame catturati in un formato adatto al rendering con OpenGL (ad esempio, da YUV12 a RGB565), può intervenire su alcune caratteristiche del video, come la luminanza, per facilitare il tracking dei target, e organizza i frame catturati in stack che comprendono versioni dello stesso fotogramma a risoluzioni differenti, in base alle necessità dell'applicazione ed alle caratteristiche hardware del dispositivo. *Tracker* contiene gli algoritmi di computer vision utilizzati per riconoscere e tracciare i target e lo stato dei pulsanti virtuali. I dati ottenuti dal componente sono uniti ai frame già convertiti, e memorizzati in un oggetto (*State Object*) che

rappresenta lo stato della scena analizzata, accessibile dal renderer e dal codice dell'applicazione. *Video Background Renderer*, infine, si occupa del rendering delle immagini processate contenute nello State Object.

Una volta inizializzati i componenti all'interno dell'applicazione, ad ogni frame analizzato verrà aggiornato lo State Object, ed invocato conseguentemente il Video Background Renderer. Lo sviluppatore dovrà quindi, per ogni frame, semplicemente interrogare lo State Object, utilizzarne i dati per aggiornare la logica dell'applicazione, ed occuparsi del rendering dell'overlay AR, ossia degli elementi virtuali da sovrapporre.

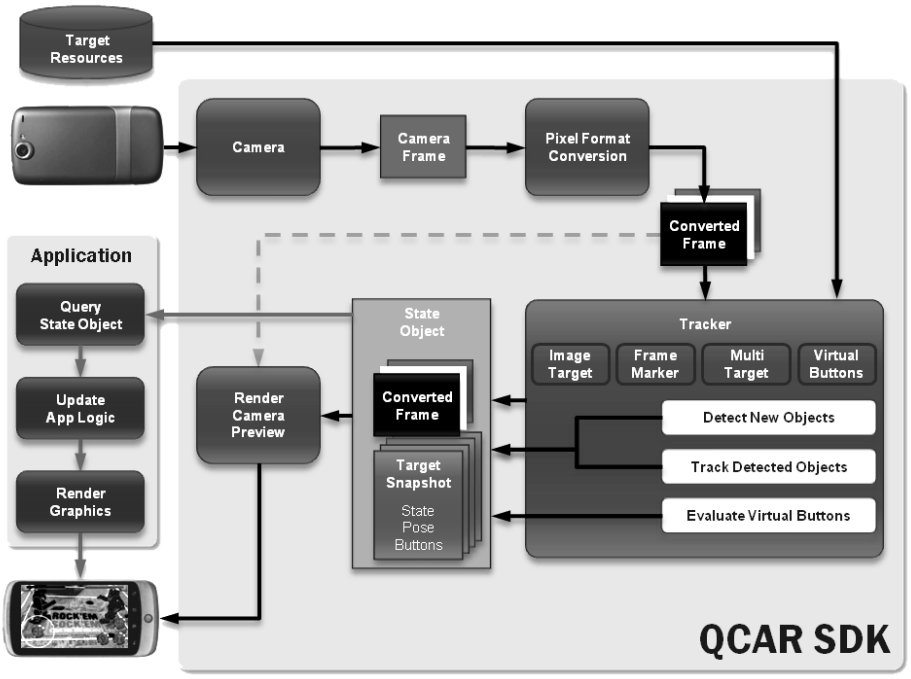

Figura 4.5: Architettura di un'applicazione basata su QCAR [QUA10]

Per semplificare ulteriormente i compiti dell'applicazione, soprattutto per quel che riguarda il rendering, QCAR SDK fornisce un'estensione per l'engine tridimensionale multipiattaforma Unity3D di Unity Technologies, a cui aggiunge funzionalità per il riconoscimento ed il tracking dei target. Mediante l'IDE di Unity3D, un avanzato editor WYSIWYG, è possibile sviluppare l'applicazione in modo visuale e gestirne la logica mediante

script, utilizzando linguaggi come C# e JavaScript. L'engine automatizza l'interrogazione dello State Object e la gestione degli elementi virtuali legati ai target riconosciuti nel video, integra l'engine fisico nVidia PhysX per consentire simulazioni fisiche realistiche, ed offre funzionalità grafiche avanzate per il rendering della scena. Queste ultime sono particolarmente utili se l'applicazione richiede la gestione di elementi virtuali complessi o dall'aspetto realistico, come personaggi animati o persino intere ambientazioni virtuali sovrapposte alla realtà. Sebbene Unity3D sia disponibile in versione gratuita e supporti nativamente una larga varietà di piattaforme, l'estensione di QCAR SDK è compatibile esclusivamente con Android e iOS, e il deployment su di esse richiede una licenza a pagamento.

4.3 Software AR disponibile per Android

Come già detto, il portfolio di software applicativo AR disponibile su Android è ampio e in rapida crescita. La vastità della base installata e la semplicità di accesso ai kit di sviluppo hanno stimolato, in generale, la creazione di un largo volume di software destinato a target di riferimento anche completamente disgiunti, in grado di soddisfare esigenze differenti e di esplorare le potenzialità di diverse tecnologie o tipologie di interfaccia. Ciò rimane vero per le applicazioni dotate di funzionalità augmented reality, i cui domini applicativi variano dall'intrattenimento alla produttività o all'educazione, e possono essere destinate ad un'utenza di tipo generale oppure, in alcuni casi, richiedere conoscenze più specializzate.

Ulteriori fattori responsabili per la diffusione di tale tecnologia includono l'integrazione dell'hardware per essa necessario (telecamera, sensori e display) nel Compatibility Definition Document, la disponibilità, come già visto, di un largo numero di framework e librerie AR gratuite o con licenze a pagamento dai costi comunque accessibili per sviluppatori non professionisti, e l'overlap generale tra gli sviluppatori software e gli appassionati di nuove tecnologie. Qualcomm, inoltre, organizza concorsi annuali per lo sviluppo, mediante QCAR SDK, di concept originali per

applicazioni AR mobili, e promuove iniziative in partnership con diversi college americani per favorire la diffusione del proprio framework.

Le applicazioni attualmente disponibili sull'Android Market consentono all'utente di sovrapporre alla realtà articoli di Wikipedia dotati di geotagging, in modo da visualizzare informazioni su monumenti e landmark semplicemente inquadrandoli, oppure le orbite di satelliti geostazionari, di corpi celesti, o persino di visualizzare le strutture ed i nomi delle costellazioni puntando il dispositivo verso di esse. Consentono di individuare percorsi per certe destinazioni, ad esempio per ritrovare la propria auto parcheggiata o la locazione di reti Wi-Fi note, di individuare la posizione aggiornata in tempo reale dei propri amici, di sostituire del testo in lingua straniera con la sua traduzione direttamente all'interno dell'immagine catturata, di distorcere la realtà tramite caleidoscopi virtuali, di simulare la visione di una persona affetta da daltonismo, o di giocare vari tipi di videogiochi con oggetti virtuali sovrapposti al mondo reale.

Figura 4.6: Overlay AR di *Wikitude Drive* [WIK09]

Wikitude Drive, ad esempio, è un sistema di navigazione per Android dotato di funzionalità augmented reality. Utilizza sensori standard quali il ricevitore GPS incorporato nel dispositivo per la localizzazione, ed

il magnetometro come bussola per individuare la direzione verso cui esso è rivolto. In questo modo, è possibile visualizzare sul display mappe e indicazioni con una prospettiva in prima persona del tutto simile a un qualsiasi navigatore GPS stand-alone. La modalità AR integra in overlay i percorsi suggeriti, più una serie di informazioni aggiuntive come velocità attuale e distanze, con il flusso video catturato dalla telecamera. Una volta posizionato il dispositivo su uno stand fisso all'interno del veicolo, ciò consente al guidatore di usufruire di tali informazioni senza distogliere lo sguardo dalla realtà circostante, in modo da non perdere di vista nemmeno per un attimo eventuali ostacoli, come pedoni o altre automobili, impossibili da tracciare mediante un sistema di navigazione tradizionale.

Un altro esempio notevole è *TagWhat*, applicazione orientata al social networking che utilizzava una visuale AR, rimossa per motivi non chiari dalla nuova versione del programma, per aggiungere informazione alla realtà circostante a beneficio di altri utenti. Sebbene non sia più considerabile a tutti gli effetti come tale, le sue caratteristiche la rendono comunque un buon esempio delle potenzialità delle applicazioni AR su dispositivi mobili. TagWhat consente ad ogni utente di associare *tag*[1], ovvero pop-up informativi contenenti testo, immagini o video, al luogo in cui egli si trova al momento (individuato dal sistema, similmente a Wikitude Drive, tramite ricevitore GPS e magnetometro). I tag possono contenere informazioni sull'elemento che complementano, oppure semplicemente le opinioni di chi lo ha taggato, o file multimediali come fotografie e brevi video, e possono inoltre essere condivisi, sotto forma di cartoline virtuali, via e-mail o attraverso i principali siti di social networking, come Facebook. Prima della rimozione della modalità AR dal programma, questi venivano sovrapposti in overlay al video catturato dalla telecamera, e orientando il dispositivo in varie direzioni nello spazio era possibile visualizzare i tag intorno all'utente, direttamente sovrapposti agli elementi ad essi associati. La nuova versione, tuttavia, si limita a mostrarli all'interno di un menu bidimensionale, indipendentemente dall'orientamento del dispositivo.

[1] Il processo di associare un elemento a delle informazioni addizionali, o ad altre entità come persone, nel gergo dei social network, è comunemente definito *taggare*.

Come facilmente intuibile dalla varietà degli esempi precedenti, il marketplace di Android offre prodotti dotati di funzionalità augmented reality per numerosi tipi di esigenze differenti, il che si traduce in elevate potenzialità di sviluppo per il futuro all'interno di svariati domini applicativi. La molteplicità di possibili applicazioni per tale tecnologia sulla piattaforma, inoltre, ha portato alla nascita di una nuova categoria di software AR: i browser augmented reality general purpose. L'esempio più popolare è senza dubbio Layar, descritto dettagliatamente nel prossimo paragrafo.

4.4 Layar: Augmented Reality Browser

Layar [LAY09], anche denominato *Layar Browser* o *Layar Reality Browser*, è, come già detto, un browser AR general purpose multipiattaforma, sviluppato dalla compagnia omonima e disponibile gratuitamente per Android e iOS, in grado di aumentare la realtà con diversi tipi di informazione in base alle esigenze del momento e agli interessi degli utenti.

4.4.1 Content layer

Ciò è possibile tramite estensioni sviluppate da terze parti, denominate *content layer* e specializzate per fini e domini applicativi anche radicalmente differenti. I content layer, concettualmente equivalenti a plugin caricati ed eseguiti a runtime, vengono selezionati dall'utente direttamente all'interno dell'interfaccia grafica del programma (esiste infatti un'attività dedicata al catalogo dei layer), e sovrappongono al video catturato dalla telecamera elementi come testo, file multimediali o oggetti 3D, il cui significato semantico dipende dallo specifico layer attualmente in esecuzione. La necessità di eseguire software aggiuntivo, sviluppato mediante tool e API standard, per implementare una logica applicativa vera e propria, consente di classificare Layar come una piattaforma general purpose.

I content layer sono quindi considerabili come dei subset di informazione sovrapposti al flusso video, in maniera simile a più fogli di plastica

trasparenti posizionati a strati uno sopra l'altro, i cui contenuti aumentano la realtà circostante e incorporano la logica del programma.

Sebbene molti dei content layer disponibili utilizzino la posizione del dispositivo (ottenuta sempre tramite ricevitore GPS e magnetometro) per visualizzare informazioni associate agli elementi circostanti e registrarle in 3D in loro corrispondenza, in maniera simile alla modalità AR di TagWhat, ciò non è obbligatorio. E' infatti possibile sviluppare anche dei layer vision-based per il riconoscimento di immagini, piuttosto che location-based, utilizzando *Layar Vision* (libreria descritta in seguito). In entrambi i casi, gli elementi sensibili riconosciuti dal layer intorno all'utente sono denominati POI (*Point of Interest*), ed oltre alla normale modalità AR possono anche essere visualizzati in una lista bidimensionale, o disposti su una mappa (mediante l'attività Maps di Android).

Figura 4.7: A sinistra: Layer Gallery, il catalogo di content layer. A destra: registrazione in 3D intorno all'utente, tramite un apposito layer, di articoli di Wikipedia dotati di geolocation.

4.4.2 Creazione ed esecuzione di content layer

I layer sono scritti utilizzando comuni tecnologie web quali PHP, Java e MySQL, e sono composti da una parte client, eseguita sui dispositivi di ogni utente, ed una parte server, denominata *web service* nella documentazione ufficiale, eseguita su un server remoto gestito dallo sviluppatore.

La parte client è molto semplice e viene generata attraverso un pannello di controllo presente sul Publishing Site, un portale web accessibile esclusivamente mediante un account da sviluppatore dopo essersi registrati sul sito ufficiale di Layar. E' possibile specificare, oltre al nome del layer, anche la sua tipologia, l'URL dell'endpoint lato server, e dettagli aggiuntivi quali descrizione e screenshot che ne descrivono il funzionamento. Il client comunica con il web service tramite HTTP con una chiamata a *GetPOIs Request*, contenente la posizione dell'utente e dettagli aggiuntivi come la lingua utilizzata o il raggio entro cui cercare dei POI, e si aspetta di ottenere una *GetPOIs Response* che contenga i dati necessari per la logica del programma, e una lista dei punti di interesse all'interno del raggio specificato. La comunicazione tra parte client e parte server è mediata da un server proxy di Layar, che si occupa di validare i dati scambiati, ma il trasferimento di risorse multimediali (immagini, modelli 3D, link ipertestuali, eccetera), invece, è diretto tra i due endpoint.

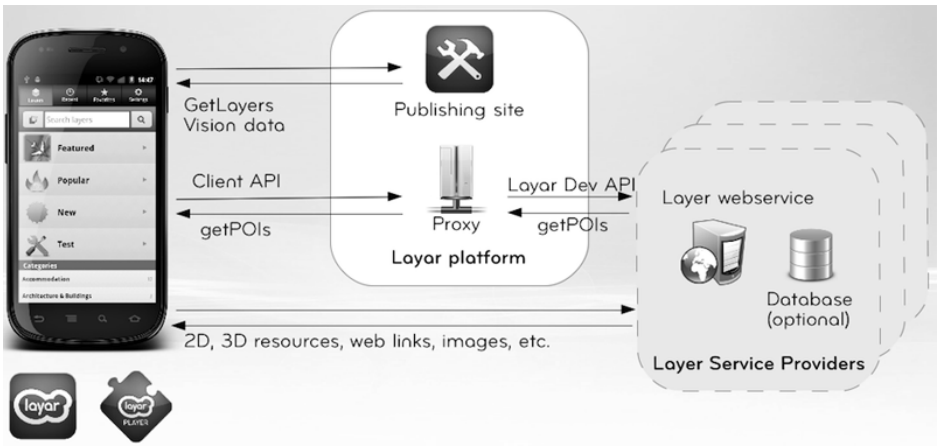

Figura 4.8: Comunicazione tra client e server mediata da Layar [LAY09]

Il web service utilizza un database, tipicamente basato su MySQL, per immagazzinare i dati sui POI. Questi ultimi, a seconda della tipologia di layer, possono essere classificati come *Geo POI* (punti di interesse geografici, definiti da coordinate GPS) o *Vision POI* (dati sulle immagini riconoscibili tramite Layar Vision). Una volta ricevuta una richiesta dal client, il server interpreta i parametri HTTP ed interroga il database per restituire una lista, formattata in JSON, dei POI necessari. Il server può inoltre restituire delle *action*, ovvero una serie di possibili comandi eseguibili dall'utente quando il POI a cui sono associate è evidenziato. Esempi di azioni includono l'apertura di una pagina web, la riproduzione di file multimediali, l'avvio di una chiamata vocale, o l'invio di un'e-mail.

La posizione dei POI nella visuale AR sullo schermo dei client è tipicamente rappresentata tramite delle icone dall'aspetto personalizzabile, sempre rivolte in direzione dell'utente indipendentemente dalla sua posizione e angolazione (modalità usualmente denominata billboard). E' tuttavia possibile sostituire tali icone con immagini 2D, video o modelli tridimensionali ricevuti dal server. Tali oggetti possono essere dotati di trasformazioni (definite da matrici di traslazione, rotazione e scaling) e persino animazioni, ed insieme alle action consentono di aggiungere interattività all'applicazione: diventa infatti possibile utilizzare le action per interagire in vari modi con gli elementi virtuali circostanti. Ad esempio, per un semplice videogioco, il server potrebbe restituire dei POI rappresentati tramite modelli 3D di soldati nemici, modificandone le coordinate relativamente a quelle dell'utente (ricevute come parametro della GetPOIs Request) per farli apparire localizzati intorno a lui, e fornire all'utente un'azione per ucciderli. In questo modo, l'utente può sconfiggere i nemici puntando il dispositivo nella loro direzione ed eseguendo l'action, e lo schermo del dispositivo diventa una finestra verso il mondo del videogioco, attraverso cui osservare e interagire con elementi non visibili nella realtà.

Grazie alla comunicazione tra client e server effettuata tramite HTTP, è possibile utilizzare il protocollo per richiedere autenticazione da parte dell'utente per accedere al layer o a parti di esso, filtrando i POI restituiti a meno che non sia stato effettuato il login. Uno dei parametri per il

client, settabile tramite il Publishing Site, è la richiesta di un cookie valido all'avvio: se *true*, Layar controllerà la presenza o meno sul sistema di un cookie per l'URL associato (una pagina web sullo stesso dominio del web service utilizzata per il login, tipicamente con protocollo HTTPS per garantire maggiore sicurezza). In caso questo sia assente, Layar lancerà automaticamente un'attività contenente il browser web per consentire il login oppure, se possibile, la registrazione di un nuovo account.

I layer possono inoltre essere avviati utilizzando un *Layar intent*, ovvero un URL in uno dei tre possibili formati indicati qui di seguito:

- `layar://<layername>`
- `http://m.layar.com/open/<layername>`
- `layarshare://<layername>/?<parameters>`

Il primo formato può essere utilizzato all'interno di Layar, ad esempio come link ipertestuale per un layer specifico o per aprire una pagina web (come nel caso del login), oppure all'interno dei browser web di Android e iOS se Layar è installato sul dispositivo. Il secondo formato può essere utilizzato anche esternamente, in qualsiasi applicazione o sito web, e consente di avviare direttamente il layer specificato oppure, se Layar non dovesse essere installato sul sistema, redirige verso un link per scaricare il programma. Il terzo formato può essere utilizzato esclusivamente all'interno di Layar, e consente all'utente di condividere link al layer tramite siti di social network come Facebook e Twitter.

Layar mette inoltre a disposizione una serie di tool di terze parti per facilitare la creazione e la gestione dei web service che fungono da backend per il layer, tramite *CMS* (Content Management System) che forniscono interfacce grafiche per l'aggiunta dei POI al database e la gestione delle funzionalità lato server. Per utilizzare tali servizi, sono richiesti account a pagamento. E' inoltre disponibile per il download *PorPOISe* (Portable Point Of Interest Server), un server open source che già implementa le funzionalità di comunicazione con il client e la formattazione in JSON, e supporta file XML o database SQL per lo storage dei dati.

4.4.3 Layar Vision

Layar Vision è un'estensione per Layar che consente di definire Vision POI, in modo da implementare funzionalità AR vision-based all'interno dei layer. A differenza dei framework AR tradizionali non utilizza marker ma, similmente agli image target ed ai multi target di QCAR, è in grado di riconoscere e tracciare in tempo reale elementi di qualsiasi genere.

Tali oggetti, denominati *target object*, possono essere creati caricando sul Publishing Site un'immagine di riferimento, dalle caratteristiche principali il più possibile chiare e distinte. Il server di Layar analizza l'immagine e ne ricava una *visual fingerprint*, ovvero un set di caratteristiche base equivalente a delle impronte digitali univoche, che consente l'identificazione del target object all'interno del flusso video. L'algoritmo che genera la visual fingerprint necessita di un certo numero di caratteristiche uniche e sufficiente contrasto, per cui immagini sfocate, pattern ripetitivi o marker bidimensionali simili a quelli usati da ARToolKit non consentono di ottenere risultati soddisfacenti. Una volta riconosciuta la visual fingerprint nel video, un oggetto 2D o 3D denominato *virtual augment* viene sovrapposto al target object per aumentare la scena, e l'utente può eseguire le action ad esso associate per la logica applicativa del layer.

Figura 4.9: Layer che consente di avviare un link per acquistare biglietti di un concerto dopo averne identificato un manifesto (target object) [LAY09]

I dati relativi alle visual fingerprint sono conservati all'interno dei server di Layar. Una volta ottenuta una GetPOIs Request, il web service fornisce al server un identificatore (una stringa denominata *image key*), e quest'ultimo include i dati necessari all'interno della GetPOIs Response. In questo modo, il client riceve il Vision POI ed è in grado di riconoscere il target object, e renderizzare quindi l'augment ad esso associato. A causa del passaggio attraverso i server Layar per il download dei Vision POI, l'utilizzo di Layar Vision all'interno dei propri layer richiede il pagamento, da parte dello sviluppatore, di una quota mensile variabile, in modo da coprire i costi di bandwidth associati al processo.

4.4.4 Layar Player

Layar Player, al momento disponibile esclusivamente per piattaforme iOS ma di prossima uscita anche su Android, consente di avviare i propri layer all'interno di applicazioni esterne, senza dover eseguire o installare Layar. Ciò consente allo sviluppatore di integrare le funzionalità AR descritte in precedenza con il proprio codice, senza richiedere all'utente di eseguire un'applicazione (il browser AR Layar, appunto) separata. Per avviare un layer con Layar Player è sufficiente scaricare l'SDK e aggiungere poche righe di codice al proprio software per la sua inizializzazione.

Capitolo 5

ARTC - Progetto di content layer

Figura 5.1: Mockup di ARTC in esecuzione

Per questa tesi si è scelto di sviluppare come progetto un esempio concreto di content layer, denominato ARTC, che sfrutti le caratteristiche e le potenzialità della piattaforma Layar, per arrivare poi a suggerirne una possibile implementazione alternativa come applicazione stand-alone.

La scelta di vincolare il progetto alla tecnologia Layar, piuttosto che agli altri framework esaminati nel capitolo precedente o ad un'implementazione indipendente già in partenza, è principalmente dovuta al fatto che la piattaforma già include un'infrastruttura completa di tipo client-server,

che consente al client Android di ricevere i dati necessari per la logica applicativa da una sorgente esterna e di visualizzarli o eseguirli indipendentemente dalla logica stessa e dalla natura o dall'origine dei dati, massimizzando quindi le potenzialità per il riuso e lo sviluppo di progetti similari anche in domini applicativi molto differenti. Con gli altri framework, o partendo da zero, sarebbe infatti necessario implementare anche tale infrastruttura, con costi e tempi di sviluppo ovviamente superiori.

Grazie a Layar, inoltre, una porzione rilevante della fase di progettazione si riduce alla semplice strutturazione dei dati e del web service, in quanto il lato client è generato automaticamente dopo aver impostato alcuni semplici parametri nel Publishing Site, e Layar stesso si occupa dell'interpretazione dei dati ricevuti dal server, del rendering su schermo dei POI e dell'esecuzione delle action ricevute. Ciò richiede ovviamente l'adesione ai formati utilizzati per la comunicazione tra i due endpoint, ovvero i parametri e i tipi di codifica usati da GetPOIs Request e GetPOIs Response, ma l'architettura di Layar fornisce una certa libertà per la logica del web service (si veda l'esempio dato in precedenza per il videogame in cui le coordinate dei POI, che rappresentano i nemici, sono generate in base alle coordinate del client, ovvero il giocatore, e non realmente estratte da un database), ed è proprio questo a fornire la flessibilità necessaria per l'utilizzo della piattaforma in vari domini applicativi.

Il progetto ARTC, descritto nei prossimi paragrafi, evidenzia i passi da seguire e gli elementi principali di cui tenere conto durante lo sviluppo di un content layer. E' inoltre basato su requisiti intenzionalmente laschi, in modo da offrire spunti per l'integrazione di funzionalità aggiuntive ove necessario, e, sebbene l'esempio sia rivolto in particolare ad una specifica azienda operante nel settore dei trasporti pubblici, il progetto è ovviamente adattabile con facilità anche a contesti radicalmente differenti, in quanto problematiche quali la ricezione e la visualizzazione di punti di interesse localizzati intorno all'utente, oppure l'associazione di action a ciascuno di essi per implementare le funzionalità richieste dal committente, sono molto comuni nell'ambito delle applicazioni AR, e la loro risoluzione favorisce evidentemente il riutilizzo del codice all'interno di progetti successivi.

5.1 Requisiti

Si intende realizzare un applicativo software per dispositivi mobili Android basato su tecnologie augmented reality e denominato ARTC, da proporre ad agenzie di trasporti pubblici, quali *ATC - Azienda Trasporti Città* di Bologna, per la distribuzione al pubblico sotto forma di content layer eseguito all'interno del browser augmented reality Layar.

ARTC è in grado di fornire all'utente informazioni sulla posizione delle fermate e sugli orari degli autobus per il bacino di Bologna (gestito da ATC stessa). Il software utilizza una visuale AR per mostrare la posizione delle fermate relativamente all'utente, ed è in grado di visualizzare orari aggiornati per ognuna delle linee che vi afferiscono, in base alle informazioni più recenti messe a disposizione dalla compagnia.

5.2 Analisi del problema

Il software da sviluppare è destinato al mercato consumer e fornisce su richiesta informazioni utili, indirizzate al singolo utente in base alla sua locazione. Le specifiche, dunque, definiscono, indipendentemente dalla tecnologia effettivamente utilizzata, un'applicazione AR di tipo location-based, ovvero che sfrutti le coordinate dell'utente per reperire, all'interno di un determinato raggio di ricerca, coordinate e dati sugli elementi virtuali da registrare al flusso video, e li posizioni coerentemente con la locazione e l'orientamento del dispositivo che esegue l'applicazione.

Le entità in gioco sono l'*utente*, le *fermate*, le *linee* degli autobus e gli *orari*. Il software effettua, all'avvio, una *ricerca*, caratterizzata da un *raggio* e contenente le *coordinate* dell'utente, ed ottiene come risultato una lista di fermate, caratterizzate dalle loro coordinate e da una lista di orari per ciascuna delle linee che vi afferiscono. All'utente non è consentito modificare i dati o interagirvi in alcun modo, egli può infatti soltanto *selezionare* una fermata all'interno del raggio di ricerca e visualizzare sul proprio dispositivo gli orari relativi ad ogni linea.

Questa è l'unica azione per l'utente presente nei requisiti, in modo da mantenere la struttura del software quanto più semplice possibile. Tuttavia, per poter fornire ulteriori esempi sulle action di Layar, verranno poi descritti alcuni esempi di possibili azioni aggiuntive. I requisiti non funzionali, inoltre, specificano l'utilizzo di tecnologie ben precise, per cui l'approccio utilizzato in fase di progettazione e di implementazione dovrà necessariamente tenere conto delle loro caratteristiche e limitazioni.

5.3 Progettazione

I requisiti impongono un'implementazione sotto forma di content layer per Layar, il che implica, come già visto in precedenza, un'architettura di tipo client-server, con un client molto semplice eseguito dall'utente sul dispositivo Android dopo averlo selezionato all'interno della Layar Gallery, il catalogo integrato nel browser AR. Il client si connette ad un web service eseguito su un server remoto per scaricare una lista di punti di interesse (POI) e le eventuali action ad essi relative, e l'interfaccia grafica di Layar consente all'utente di eseguire le action associate al POI evidenziato semplicemente toccando il pulsante su schermo che le rappresenta.

Utilizzare Layar per il progetto significa servirsi di strumenti già esistenti, come il pannello di controllo del Publishing Site per il lato client, e tecnologie web come PHP e MySQL per il lato server. Pertanto, più che sulla strutturazione delle classi (non esistenti in questo contesto) e sull'applicazione di design pattern, la fase di progettazione verterà su quali strumenti forniti da Layar utilizzare, su quali parametri dei metodi usati per la comunicazione tra i due endpoint (ovvero le già citateGetPOIs Request e GetPOIs Response) gestire, e su come strutturare i dati di POI e action necessari per implementare la logica applicativa desiderata.

5.3.1 Progettazione del client

Per quel che riguarda il client, il Publishing Site permette di impostare tutti i parametri necessari, e non è necessario (nè comunque possibile)

scrivere una singola riga di codice. Il layer non richiede l'utilizzo di icone particolari per marcare la posizione delle fermate circostanti, ma per garantire una maggiore usabilità è possibile sostituire tali icone con immagini bidimensionali che contengano, ad esempio, i numeri degli autobus che le utilizzano, in modo da fornire all'utente tali informazioni a colpo d'occhio, senza richiedere che le evidenzi una alla volta per trovare gli autobus a cui è interessato. Per utilizzare tali immagini occorrerà pertanto selezionare *3D and 2D Objects in 2D Space*, piuttosto che *Generic (2D)*, come tipo di layer, in modo da abilitare la ricezione dal server di file multimediali.

Bisognerà, inoltre, specificare un URL per l'endpoint del server (tipicamente il file .php che risponde alle GetPOIs Request). Non è necessario un login ad alcun sito o servizio, per cui non verrà richiesta la presenza di un cookie valido all'avvio. Layar Vision non è utilizzato, per cui la casella corrispondente alla sua abilitazione verrà deselezionata. Una volta creato il layer, il Publishing Site consentirà inoltre di settare un valore di default per il raggio di ricerca, e di impostare icone, descrizioni e screenshot in modo da personalizzarne la presentazione all'interno della Layer Gallery.

5.3.2 Progettazione del server

Per il funzionamento del layer, è necessario che il web service disponga di dati aggiornati, sia per gli orari che per l'effettiva posizione delle fermate. Sarà quindi necessario avere accesso ad un database MySQL, i cui dati si suppone siano gestiti ed aggiornati con regolarità dalla compagnia, possibilmente anche tenendo conto del traffico, di imprevisti nella viabilità, e di corse saltate per via di guasti o problemi tecnici, in modo da risultare più precisi e affidabili degli orari già presenti ad ogni fermata.

Nel caso di ARTC, i POI sono evidentemente le fermate circostanti, e la visualizzazione degli orari relativi alla fermata evidenziata è l'unica action richiesta dalle specifiche. Quindi, il database dovrà contenere una tabella *POI* per le fermate, ed una tabella *POIAction* per le action associate. Entrambe le tabelle dovranno contenere dei campi specifici, evidenziati in figura, per poter interagire correttamente con le API Layar.

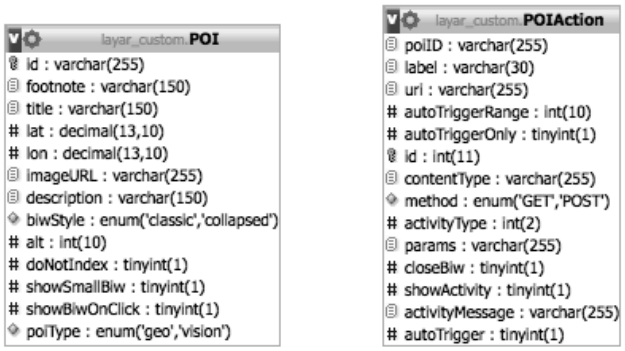

Figura 5.2: Campi delle tabelle POI e POIAction usati da Layar [LAY09]

La tabella POI deve contenere, oltre a un *id* univoco per identificare la fermata ed ai valori di *lat* e *lon* (latitudine e longitudine) per le coordinate, anche una descrizione testuale ed un campo *imageURL* contenente un link all'immagine usata per marcarne la posizione all'interno della visuale AR. Deve essere inoltre specificato il tipo di POI tramite un enum, i cui possibili valori sono *geo* e *vision*, nel campo *poiType*. In questo caso, essendo il content layer in esame di tipo location-based, si utilizzerà il primo.

Data la necessità di mostrare gli orari su uno schermo di dimensioni potenzialmente ridotte, e data l'impossibilità di interazione da parte dell'utente con i dati scaricati dal server, si suppone di visualizzare questi ultimi all'interno di una pagina web opportunamente formattata in modo da garantire leggibilità, accessibile tramite un link ipertestuale avviato dall'action tramite l'attività browser web standard di Android. Per estendere le feature di ARTC, è possibile progettare ulteriori action aggiuntive, non previste nei requisiti iniziali, come un link al sito ufficiale della compagnia o la possibilità di comunicare eventuali disservizi via e-mail, in modo da migliorare l'user experience complessiva dell'applicazione.

La tabella POIAction, quindi, utilizzerà un campo *uri* per specificare l'URL della pagina web da visualizzare, oppure il numero di telefono da comporre o l'indirizzo a cui inviare e-mail nel caso delle action aggiuntive. Devono inoltre essere presenti un *poiID* per determinare a quale POI l'action è associata, un *contentType* che specifichi il tipo di contenuto dell'URI (ad esempio *text/html* nel caso di un link a una pagina web, oppure

application/vnd.layar.internal per numeri di telefono o indirizzi e-mail). Possono infine essere specificate funzionalità aggiuntive supportate da Layar, come ad esempio l'esecuzione automatica delle action quando l'utente si trova all'interno di un certo raggio dal POI (*autoTriggerRange*).

Dopo aver progettato la struttura dei database, il passo successivo è lo sviluppo del web service. Quest'ultimo può essere basato sul già citato server open source PorPOISe, che già implementa buona parte delle caratteristiche necessarie (comunicazione con il client, collegamento al database, codifica in JSON della risposta), oppure può essere scritto da zero in PHP o Java. Una volta avviato (ad esempio con Apache su una macchina locale per semplificarne il testing), il web service rimane in attesa di GetPOIs Request da parte dei client, da cui estrarre i parametri HTTP necessari per elaborare la risposta. Per il progetto in esame, i parametri realmente utili saranno *layerName* per assicurarsi che la richiesta sia effettivamente relativa al layer ARTC, *lat* e *lon* per le coordinate del client, e *radius* per il raggio di ricerca. Sebbene la Request consenta di specificare ulteriori parametri, come *countryCode* o slider e filtri aggiuntivi per i POI da restituire in risposta, questi non sono rilevanti per il caso di ARTC.

Ricevuta la richiesta e verificati i parametri in essa contenuti, il server interroga il database tramite un metodo *getHotspots()*, per elaborare la risposta da inviare al client tramite una GetPOIs Response. Quest'ultima verrà quindi codificata in JSON, e includerà una chiave di tipo stringa chiamata *layer*, contenente il nome del layer a cui la risposta è diretta per garantire di non ricevere sul client, in ritardo, risposte relative a layer precedentemente in uso. La Response conterrà inoltre un array di POI chiamato *hotspots* contenente i dati su POI e relative action ricavati da getHotspots(), un intero chiamato *errorCode* contenente l'esito della richiesta (con valore zero se quest'ultima ha avuto successo, o un codice numerico di errore altrimenti), ed un'altra stringa chiamata *errorString* per mostrare eventuali messaggi di errore all'interno della visuale AR. Una volta ricevuta la Response, il client renderizzerà le immagini scaricate dal server in corrispondenza dei POI, e consentirà quindi all'utente di eseguire le action descritte in precedenza, toccandole, per interagire con il programma.

5.4 Implementazione

Come già detto in precedenza, l'implementazione del lato client consiste semplicemente nell'utilizzare il Publishing Site per impostare alcuni parametri di base (descritti nel paragrafo precedente). Il client, infatti, non esegue codice scritto dallo sviluppatore, ma si limita a caricare layer pubblicati nella Layer Gallery, con logica applicativa interamente a carico del web service e delle action restituite con la GetPOIs Response.

Per quel che riguarda il lato server, invece, una volta inizializzati e popolati i database, il passo successivo è l'implementazione del web service vero e proprio. Quest'ultimo dovrà utilizzare come entry point un file con lo stesso URL indicato in precedenza sul client come endpoint remoto a cui inviare GetPOIs Request, la cui struttura è simile alla seguente:

Listing 5.1: Formato GetPOIs Request ricevuta dal server

```
http://www.dominio.it/ARTC.php
  ?lang=it            // lingua del client
  &countryCode=IT     // codice nazione del client
  &layerName=ARTC     // nome del layer
  &lat=44.495155      // latitudine del client
  &lon=11.340997      // longitudine del client
  &radius=1000        // raggio di ricerca
  &accuracy=100       // precisione del GPS
  &userId=<hash ID utente>
  &developerId=1337
  &developerHash=<hash ID sviluppatore>
  &version=4.0
  &timestamp=1330014079
```

I parametri necessari per il caso di ARTC sono, come già detto nel paragrafo precedente, soltanto *layerName*, *lat*, *lon* e *radius*. Servirà quindi un metodo denominato *getRequestParams()* per estrarne i valori. In PHP, i parametri della richiesta HTTP sono contenuti nella variabile globale `$_GET`, un dizionario con i nomi dei parametri come chiavi, per cui il processo di estrazione si limita semplicemente ad una copia dei parametri desiderati all'interno dell'array `$params`, tramite un array di chiavi `$keys`.

Listing 5.2: Estrazione dei parametri dalla GetPOIs Request

```
// $keys contiene le chiavi dei parametri
// da copiare in $params
foreach($keys as $key)
{
  if (isset($_GET[$key]))
  {
    $params[$key] = $_GET[$key];
  }
  else
  {
    throw new Exception($key .' Parametro non presente nella
        GetPOIs Request.');
  }
}
return $params;
```

Una volta recuperato l'array dei parametri e verificato che il valore di layerName sia corretto, è possibile effettuare la connessione al database per interrogare la tabella POI tramite una select SQL, e ricavare tutte le fermate all'interno del raggio specificato in radius. Si utilizzerà quindi un metodo denominato *getHotspots()* per interagire con il database e strutturare i dati ricevuti all'interno di un array `$hotspots`.

Lo spezzone di codice riportato qui di seguito, tratto da uno degli esempi presenti nella documentazione ufficiale Layar e parte del metodo getHotspots(), mostra l'SQL necessario per ottenere le fermate desiderate, ordinate per distanza (crescente) e con un limite massimo di 50 risultati (il massimo numero di POI supportati contemporaneamente da Layar). L'algoritmo utilizzato per ricavare le distanze tra utente e fermate, inoltre, è basato sulla formula di Haversine, comunemente usata in navigazione per ricavare le distanze tra punti situati su una superficie sferica in base alle loro coordinate. Infine, il codice si occupa anche di assicurarsi che i POI restituiti siano esclusivamente di tipo Geo, in quanto, come già detto, Layar non permette di utilizzare sia Geo che Vision POI all'interno dello stesso content layer.

Listing 5.3: Select SQL usata per interrogare la tabella POI

```
$sql = $db->prepare( "
              SELECT id, imageURL, title, description,
                 footnote, lat, lon,
                     (((acos(sin((:lat1 * pi() / 180)) * sin((
                        lat * pi() / 180)) +
                        cos((:lat2 * pi() / 180)) * cos((lat *
                            pi() / 180)) *
                        cos((:long  - lon) * pi() / 180))
                       ) * 180 / pi()
                      )* 60 * 1.1515 * 1.609344 * 1000
                     ) as distance
                FROM POI
               WHERE poiType = "geo"
              HAVING distance < :radius
           ORDER BY distance ASC
              LIMIT 0, 50 " );
```

Gli hotspot restituiti dal metodo devono includere le action ad essi relativi, quindi il corpo di getHotspots() dovrà includere, per ogni POI recuperato dal database, anche una chiamata a una funzione denominata *getPOIActions()*, che, similmente al metodo precedente, si occupi di reperire e strutturare i dati necessari. In questo caso, la select sarà molto più semplice, in quanto si limiterà a cercare, all'interno della tabella POIAction, gli elementi con lo stesso *poiID* del POI a cui sono associati. Quindi, l'array `$hotspots` conterrà tutti i dati richiesti dal client, e sarà possibile costruire la GetPOIs Response con i valori visti in fase di progettazione, da codificare in JSON per la restituzione (il formato è supportato nativamente da PHP 5.2 o superiori tramite il metodo *json_encode()*).

Listing 5.4: Preparazione e restituzione di GetPOIs Response

```
// Assegnamento del nome del layer alla chiave layer
$response['layer'] = $params['layerName'];

// Assegnamento degli hotspot restituiti da getHotspots()
$response['hotspots'] = getHotspots($database, $params);
```

```
// Messaggio di errore in caso di array POI vuoto
if ($response['hotspots'])
{
    $response['errorCode'] = 20;
    $response['errorString'] = "Nessun POI individuato nel
       raggio di ricerca.";
}
else
{
    $response['errorCode'] = 0;
    $response['errorString'] = "ok";
}

// Codifica in JSON della risposta
$jsonResponse = json_encode($response);

// Dichiarazione del content type nell'header HTTP
header( "Content-type: application/json; charset=utf-8" );

// Restituzione della risposta in JSON
echo $jsonResponse;
```

A questo punto è possibile iniziare a testare il layer e, una volta ultimato il progetto, sottoporlo a Layar per la pubblicazione nella Gallery.

5.5 Note su Layar Vision

In alternativa all'approccio basato su POI geografici descritto finora, se i requisiti non imponessero di mostrare in visuale AR le fermate circostanti sarebbe anche possibile utilizzare Layar Vision per riconoscere dei target object dal design unico, stampati dalla compagnia e posizionati ad ogni fermata, a cui è associata un'action per mostrare gli orari degli autobus ad essa relativi (più, eventualmente, ulteriori action aggiuntive). Questo approccio, comunque, avrebbe dei costi associati superiori dovuti alle tariffe sull'utilizzo di Layar Vision.

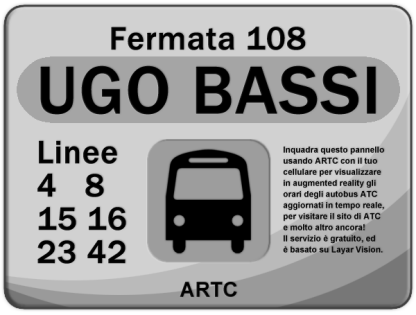

Figura 5.3: Esempio di reference image da cui ricavare un target object per una versione alternativa di ARTC che utilizzi Layar Vision

5.6 Implementazione alternativa stand-alone

Nonostante le potenzialità e la semplicità d'uso di Layar, un'applicazione AR indipendente può certamente godere di maggiore visibilità sul mercato, il che consente potenzialmente di raggiungere un pubblico più vasto. Pertanto, si è deciso di fornire alcune indicazioni per una possibile implementazione stand-alone del progetto descritto in precedenza.

In caso di una precedente implementazione sotto forma di content layer, l'obiettivo sarà raggiungibile con molta facilità successivamente al rilascio di Layar Player per Android: basterà infatti utilizzare l'SDK per creare una propria applicazione wrapper che avvii il layer pre-esistente.

In caso contrario, i framework esaminati in precedenza potranno comunque semplificare il lavoro relativo alla visuale AR. Un'applicazione vision-based, che riconosca un marker posizionato ad ogni fermata e mostri in visuale AR un pulsante virtuale da premere (toccando le coordinate del touchscreen corrispondenti) per ottenere un link ad una pagina web contenente gli orari desiderati, aggiornati dal database MySQL della compagnia, è infatti di implementazione piuttosto semplice grazie a tali strumenti. Con AndAR, ad esempio, basterà definire il pulsante come un ARObject, registrarlo in un'istanza di ARToolkit, ed estendere l'attività AndARActivity per lanciare l'attività del browser web quando l'utente preme il pulsante. QCAR, inoltre, implementa nativamente dei pulsanti virtuali: come già detto in precedenza, infatti, basterà toccare le regioni

dell'image target ad essi dedicate per registrare una pressione. Pertanto, implementare funzionalità equivalenti alle action viste in precedenza significa doversi occupare quasi esclusivamente della logica applicativa.

In teoria, a differenza di quanto visto con Layar, che non consente di utilizzare sia Geo POI che Vision POI contemporaneamente, sarebbe possibile visualizzare su schermo anche le fermate circostanti, dato che la posizione dei marker sarebbe nota a priori e basterebbe dunque mostrare oggetti aggiuntivi in coordinate relative ad essi. In pratica, però, ciò comporterebbe una pesante limitazione: sarebbe infatti necessario inquadrare il marker tutto il tempo, e perderlo di vista significherebbe perdere tutti i punti di riferimento. Il problema sarebbe evidentemente ovviabile tracciando le coordinate dell'utente, ma questo renderebbe il marker stesso fondamentalmente ridondante, in quanto non più necessario come punto di riferimento. Quindi, un'applicazione puramente location-based come quella sviluppata per il progetto soddisferebbe meglio i requisiti, ma richiederebbe un maggiore lavoro da parte dello sviluppatore, in quanto i framework in esame non supportano nativamente questo tipo di applicazione AR, ed un'intera infrastruttura equivalente a quella fornita da Layar andrebbe quindi, come già detto, implementata manualmente.

Conclusioni

La redazione del presente elaborato ha consentito di approfondire l'argomento dell'augmented reality in generale e delle sue applicazioni concrete nei vari domini esaminati, e nello specifico la scelta di focalizzare il lavoro di tesi su Android e sullo studio dei framework AR disponibili per la piattaforma ha consentito di prendere confidenza con essi, con la loro struttura e con le loro caratteristiche. Da ciò, in particolare per quel che riguarda Layar (soprattutto grazie all'esperienza pratica ottenuta tramite lo sviluppo del progetto), è possibile trarre una serie di conclusioni specifiche, per ricondurle poi alle tecnologie AR in generale.

Prima di tutto, la scelta di seguire un approccio bottom-up, fissando prima gli strumenti e progettando il sistema di conseguenza, si è rivelata fruttuosa, in quanto è stato possibile progettare un'applicazione concreta, e implementarne un prototipo, in tempi molto brevi. Va notato come, alla fine, lo sviluppo del progetto non abbia richiesto effettivamente conoscenze tecniche specifiche in campo AR, in quanto Layar automatizza del tutto la gestione di tale tecnologia, e come buona parte del codice sia effettivamente riutilizzabile in altri progetti similari semplicemente modificando il significato semantico dei dati inviati al client o la loro origine. Non è detto, ad esempio, sia davvero necessario un database, in quanto (come già visto per l'esempio del videogame citato all'interno del testo) i dati sui POI possono benissimo venire generati sul momento dal server.

Inoltre, la compatibilità di Layar con altre piattaforme simili come iOS, insieme alla generazione del tutto automatica del lato client e all'utilizzo dello standard HTTP per la comunicazione con il server, consentono di estendere il pubblico a cui tali applicazioni possono rivolgersi anche oltre i

confini dell'userbase di Android. Ciò significa che lo stesso codice lato server, senza alcuna modifica, può raggiungere un numero molto più elevato di utenti rispetto ad un'applicazione sviluppata da zero per Android.

Nonostante Layar possa quindi in questo momento sembrare quasi una panacea, una soluzione universale ad ogni problema relativo alla creazione di applicazioni AR per i dispositivi mobili supportati, l'esperienza di progetto ne ha anche portato alla luce le ovviamente esistenti limitazioni.

Prima di tutto, Layar impedisce come già detto di utilizzare sia POI visivi che geografici all'interno dello stesso layer, il che impedisce di realizzare applicazioni che sfruttino contemporaneamente sia vision-based che location-based AR. Inoltre, l'architettura client-server implica la necessità di una connessione di rete, che non è detto sia accessibile ovunque e comporta comunque dei costi per l'utente, e la comunicazione tutto sommato limitata fra client e server (una volta ricevuti i POI questi vengono aggiornati solo manualmente, tramite action o allo scadere di un timeout), insieme all'impossibilità di eseguire codice lato client, limitano fortemente le possibilità di interazione in tempo reale. Citando ancora una volta l'esempio del videogame, anche se i modelli 3D usati per rappresentare i nemici possono essere animati e possono reagire all'action fornita al giocatore per ucciderli, non potranno effettivamente muoversi o essere dotati di algoritmi di intelligenza artificiale vera e propria. E' proprio l'inaccessibilità da parte dello sviluppatore al lato client, nonostante questo garantisca compatibilità con le altre piattaforme, a rivelarsi il limite più grande di Layar, se i requisiti dell'applicazione da sviluppare richiedono feature non già offerte dalla sua infrastruttura di base oppure un'interfaccia grafica radicalmente differente da quella, comunque personalizzabile, di default.

Ovviamente gli altri framework esaminati consentono di introdurre feature più avanzate, e nonostante la mancanza di supporto nativo per AR location-based essi forniscono ugualmente allo sviluppatore una flessibilità notevolmente superiore. Come già detto, però, la mancanza di un'infrastruttura equivalente li rende meno efficienti per lo sviluppo in tempi ridotti del subset di applicazioni AR comunque realizzabile in modo soddisfacente con gli strumenti offerti da Layar (come ad esempio ARTC).

In termini generali, quindi, è possibile concludere che, una volta individuata una combinazione di hardware e software compatibile con gli obiettivi da raggiungere, i framework già esistenti possono senza dubbio aiutare lo sviluppatore, riducendo i tempi di sviluppo e le conoscenze tecniche richieste. Ovviamente, un'oculata scelta degli strumenti da utilizzare rimane fondamentale per il successo del progetto, in quanto framework diversi possono avere caratteristiche notevolmente differenti e possono quindi non comportare effettivi vantaggi in caso di limitazioni troppo stringenti o incompatibilità con i requisiti dell'applicazione.

Android, nel caso della scelta effettuata per il presente lavoro di tesi, si è rivelato una piattaforma molto fertile per l'augmented reality, come dimostrato anche dalla vastità di applicazioni già disponibili prese in esame nel quarto capitolo, e, in generale, l'elevata flessibilità offerta dagli strumenti hardware e software presenti consente di immaginare con facilità un futuro roseo per la tecnologia, soprattutto se (come auspicabile) tali strumenti continueranno a migliorare in efficienza e semplicità d'uso.

Ringraziamenti

I miei ringraziamenti vanno, innanzitutto, al Prof. Antonio Natali per avermi offerto l'opportunità di sviluppare il presente lavoro di tesi e per la fiducia mostrata nei miei confronti durante la sua redazione.

Ringrazio il Dr. Ivan Sutherland per avermi fornito personalmente un chiarimento su Sword of Damocles durante le mie ricerche sugli albori dell'augmented reality. Ringrazio inoltre il Dr. Ferenc A. Jolesz, Professore di Radiologia presso Brigham and Women's Hospital e Harvard Medical School, e Joseph Juhnke, Presidente e CEO di Tanagram Partners, per avermi dato il permesso di utilizzare alcune delle immagini pubblicate nel secondo capitolo dell'elaborato.

Ringrazio i miei genitori, Aldo e Margherita, per avermi aiutato, con tenacia e spirito di sacrificio, a crescere sia dal punto di vista intellettuale che umano, e per avermi dato la possibilità di arrivare a questo traguardo. Ringrazio il resto della mia famiglia, in particolare mio fratello Riccardo e mia sorella Luisa, per essermi stati sempre vicini, durante tutta la vita.

Ringrazio la mia fidanzata, Adele, per essere rimasta sempre al mio fianco negli ultimi otto anni, nei momenti più belli e nelle difficoltà, e per avermi reso completo come persona. Ringrazio i miei amici di Reggio Calabria, in particolare Antonio, Giuseppe e Raffaele, ed i miei amici di Bologna, per tutte le esperienze che abbiamo condiviso insieme, e che hanno contribuito a formarmi e rendermi quello che sono adesso.

Infine, ringrazio tutte le altre persone che, per brevità, non mi è possibile nominare individualmente in questa pagina, ma che sono state, o sono tuttora, parte della mia vita. A tutti voi, grazie di cuore.

Bibliografia

[CAU92] T. P. Caudell, D. Mizell, *Augmented reality: an application of heads-up display technology to manual manufacturing processes*, Proceedings of Hawaii International Conference on System Sciences, Kauai, Hawaii, 1992, Vol. 2, pp. 659-669.

[MIL94] P. Milgram, F. Kishino, *A Taxonomy of Mixed Reality Visual Displays*, IEICE Transactions on Information and Systems, E77-D(9), 1994, pp. 1321-1329.

[MAN02] S. Mann, *Mediated Reality with implementations for everyday life*, Presence Connect, PRESENCE: Teleoperators and Virtual Environments, 2002.

[AZU97] R. T. Azuma, *A Survey of Augmented Reality*, PRESENCE: Teleoperators and Virtual Environments, 1997, Vol. 6, pp. 355-385.

[HEI55] M. Heilig, *El cine del futuro: The cinema of the future*, Espacios, ristampato in PRESENCE: Teleoperators and Virtual Environments, 1992, Vol. 1, pp. 279-294.

[SUT65] I. E. Sutherland, *The Ultimate Display*, Proceedings of IFIP Congress, 1965, Vol. 2, pp. 506-508.

[SUT68] I. E. Sutherland, *A Head-Mounted Three Dimensional Display*, Proceedings of AFIPS Fall Joint Computer Conference, 1968, Vol. 33, pp. 757-764.

[VAL98] J. R. Vallino, *Interactive Augmented Reality*, University of Rochester, New York NY, 1998.

[JOL97] F. A. Jolesz M.D., *Image-guided Procedures and the Operating Room of the Future*, Harvard Medical School, Boston MA, 1997.

[TAN10] Tanagram Partners, *iARM Final Report*, Pagina web: `http://spill.tanagram.com/downloads/iARM_Final_Report.pdf`

[FEI06] S. Feiner, *ARMAR*, Pagina web: `http://graphics.cs.columbia.edu/projects/armar/index.htm`

[IMS05] University of Hong Kong, *imseCave*, Pagina web: `http://www.imse.hku.hk/intellisyslab/facilities/imseCAVE.htm`

[BUG10] Qualcomm AR Game Studio, *Bug Juice*, Pagina web: `http://ael.gatech.edu/argamestudio/2010/12/09/bug-juice/`

[NIN11] Nintendo, *AR Games*, Pagina web: `http://www.nintendo.com/3ds/built-in-software/#/4`

[OHA07] Open Handset Alliance, *Sito ufficiale*, Pagina web: `http://www.openhandsetalliance.com/`

[GOO07] Google, *Documentazione ufficiale Android*, Pagina web: `http://developer.android.com`

[GOO12] Google, *Android 4.0 CDD Rev. 2*, Pagina web: `http://source.android.com/compatibility/4.0/android-4.0-cdd.pdf`

[NYA12] Nyatla, *Sito ufficiale del progetto NyARToolkit*, Pagina web: `http://nyatla.jp/nyartoolkit/wp/`

[DOM09] T. Domhan, *Sito ufficiale del progetto AndAR*, Pagina web: `http://code.google.com/p/andar/`

[QUA10] Qualcomm, *Sito ufficiale di QCAR/Vuforia SDK*, Pagina web: `http://ar.qualcomm.com/qdevnet/sdk`

[WIK09] Wikitude, *Sito ufficiale Wikitude Drive*, Pagina web: `http://www.wikitude.com/tour/wikitude-drive`

[LAY09] Layar, *Sito ufficiale*, Pagina web: `http://www.layar.com`

Elenco delle figure