UNIVERSITÀ DI PISA

Facoltà di Scienze Matematiche, Fisiche e Naturali
Corso di Laurea in Informatica

Relazione di tirocinio

Sviluppo di una piattaforma di Realtà Aumentata per iOS

Candidato

Stefano Piamonti

Matricola 253048

Tutor Accademico

Prof. Carlo Montangero

Tutor Aziendale

Ing. Domenico Loschiavo

Anno Accademico 2011/2012

A tutti quelli che ci hanno creduto

''Don't be misguided by the gimmicky marketing applications now. Look ahead, and pay attention to what the visionaries are talking about right now. Find the right idea, help build the team, fund them, and then sit back and watch the world change. Also, AR has long term implications for smart cities, green tech, education, entertainment, and global industry. This is serious business, but it has to be done right. [...] AR is not the same as last decade's VR.''

Robert Rice, Chairman of the AR Consortium [Shu]

Indice

Elenco delle figure

Capitolo 1

Introduzione

''Augmented reality (AR) is a live, direct or indirect, view of a physical, real-world environment whose elements are augmented by computer-generated sensory input such as sound, video, graphics or GPS data. It is related to a more general concept called mediated reality, in which a view of reality is modified (possibly even diminished rather than augmented) by a computer. As a result, the technology functions by enhancing one's current perception of reality. By contrast, virtual reality replaces the real world with a simulated one. Augmentation is conventionally in real-time and in semantic context with environmental elements, such as sports scores on TV during a match. With the help of advanced AR technology (e.g. adding computer vision and object recognition) the information about the surrounding real world of the user becomes interactive and digitally manipulable. Artificial information about the environment and its objects can be overlaid on the real world.'' [Wikb]

1.1 Cos'è la Realtà Aumentata?

Se si è sentito parlare di Realtà Virtuale (tecnologia molto di moda fino a qualche anno fa), si sa che ha l'obiettivo di circondare l'utente con un ambiente completamente virtuale. È facile capire perché la Realtà Virtuale è molto utilizzata nei simulatori di volo o nei videogiochi: con un sistema di Realtà Virtuale l'utente viene allontanato

dal mondo reale e proiettato in uno completamente generato dal computer.
A differenza della Realtà Virtuale che proietta la persona in un mondo immaginario, la Realtà Aumentata (AR) invece si propone di lasciare l'utente nel mondo reale e di ''aumentare'' la sua esperienza con elementi virtuali. Rilevata con precisione la posizione e l'orientamento dell'utente, la computer graphics allinea informazioni contestuali ad oggetti reali che ne vengono così arricchiti o ''aumentati''. Un determinato oggetto reale può essere aumentato o completamente sostituito, ma non rimosso.
Secondo Azuma [A+97] si definiscono di Realtà Aumentata quei sistemi che hanno le seguenti caratteristiche:

1. Combinano Mondo Reale e Realtà Virtuale
2. Interagiscono in tempo reale
3. Lavorano in tre dimensioni

La prima caratteristica implica che il mondo reale e quello virtuale vengano miscelati e si ottiene questo risultato ''appoggiando'' oggetti di computer graphic su un flusso video.
La seconda si traduce in una modalità di interazione e un tempo di aggiornamento/risposta dello stream video di almeno 5 volte al secondo[1].
La terza comporta che il sistema lavori in un mondo tridimensionale anche se il risultato finale viene applicato a uno schermo piatto, a due dimensioni.
Secondo Milgram [MK94], invece, il mondo reale e il mondo virtuale sono due estremi di un sistema continuo uniti da una regione chiamata Mixed Reality. La AR si colloca vicino all'ambiente reale per indicare che la percezione del mondo reale è predominante sul mondo virtuale. ''Augmented Virtuality'' è un espressione creata da Milgram per identificare sistemi che mappano alcune parti del mondo virtuale con oggetti reali (vedi figura 1.1 e figura 1.2).

[1] Un frame rate di 5 immagini al secondo è il minimo per una interattività accettabile, contro i 30 necessari per una animazione fluida [TI01]

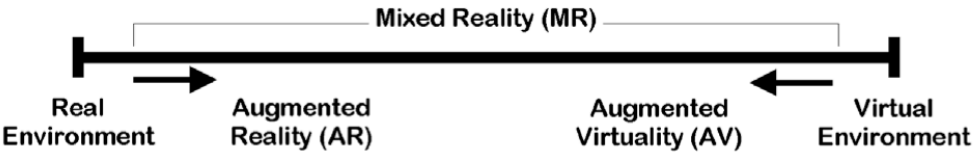


Figure 1 Milgrams Reality – Virtuality continuum

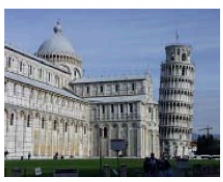


Figure 2 Real Environment

Courtesy Ericsson Medialab

Figure 3 AR

Courtesy Ericsson Medialab

Figure 4 AV

Courtesy Ericsson Medialab

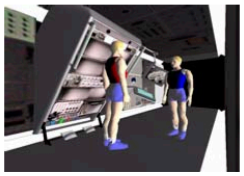


Figure 5 Virtual Environment

Courtesy Ericsson Medialab

Figura 1.1: La definizione di AR data da Milgram in [MK94].

Immagine da [Eke09].

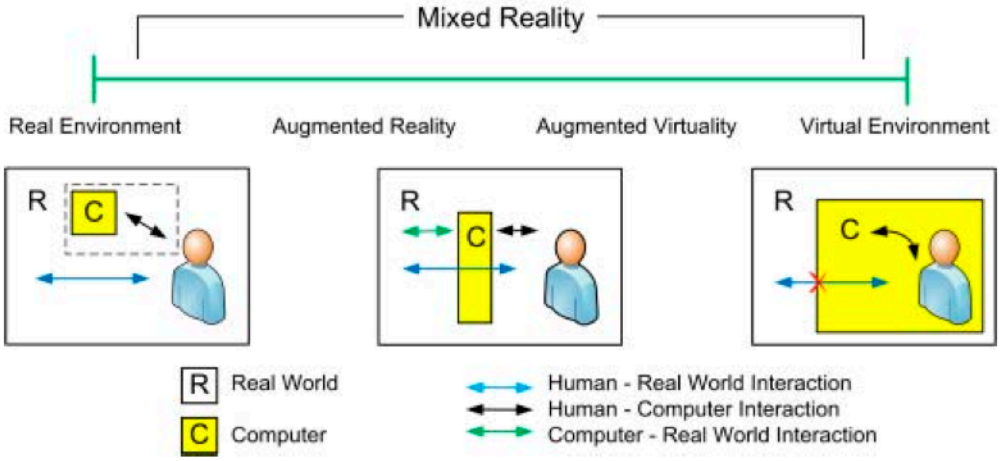


Figura 1.2: La definizione di AR data da Milgram in [MK94].

Immagine da [Hen07].

L'interesse verso la Realtà Aumentata è dovuto principalmente a due aspetti:

- il nuovo modo di fruire contenuti geolocalizzati, reali e non;
- ampliare le informazioni su ciò che si sta guardando e quindi aumentare la produttività (intelligence amplification).

Anche se ci si riferisce principalmente a elementi visuali, si può pensare anche ad ''aumentare'' suoni, oggetti concreti attraverso sensazioni tattili, ecc. In un recente documento sul futuro della tecnologia, IBM prevede dispostivi che aumentino tutti e cinque i sensi. I ricercatori di IBM immaginano, per esempio, che gli smartphone dei prossimi anni saranno in grado di offrire le sensazioni tattili degli elementi che si stanno visualizzando sullo schermo [IBM].

1.2 Passato, presente e futuro della Realtà Aumentata

L'idea della Realtà Aumentata nasce nel 1901: L. Frank Baum per primo ha l'idea di uno schermo elettronico che mostri dati sulla vita reale. Il primo sistema di AR però viene realizzato solamente 65 anni dopo: nel 1966 Ivan Sutherland della Harvard University crea il primo Head Mounted Display (HDM, vedi il § 2.1.2) per la realtà virtuale mostrato in figura 1.3 e ipotizza che quello che viene mostrato (semplici righe a schermo) possa essere fatto coincidere con elementi reali: ''[the virtual graphics could be made to] coincide with maps, desktops, walls or the keys of a typewriter'' [Loo07]. Negli anni '70 e '80 del secolo scorso tutto il lavoro di ricerca si sposta in ambito militare. Il progetto Super Cockpit della US Air Force per esempio permise di realizzare il primo Head-Up Display (HUD) sul casco dei piloti e quindi la prima vera applicazione della Realtà Aumentata [Hen07].

Il termine Realtà Aumentata viene usato per la prima volta all'inizio degli anni '90

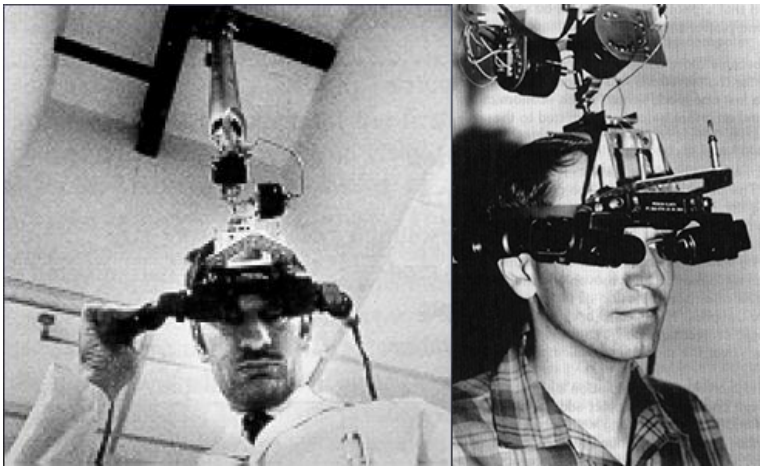


Figura 1.3: Ultimate display di Ivan Sutherland, 1965.

Immagine da [Loo07].

da Tom Caudell della Boeing per descrivere il casco utilizzato per far allenare i lavoratori ad installare cavi imbracati ai velivoli [Loo07] [Dop].
Il primo lavoro di ricerca sull'argomento viene presentato nel 1992 da Steven Feiner, Blair MacIntyre e Doree Seligmann alla Graphics Interface Conference e pubblicato l'anno successivo dalla Association for Computing Machinery (ACM).
Alla fine degli anni '90 si inizia ad utilizzare questa tecnologia durante la trasmissione di eventi sportivi televisivi e nel 1997 Steve Feiner e altri suoi collaboratori realizzano Touring Machine, il primo sistema di Realtà Aumentata Mobile a scopo civile [Dop]. Nel 1998 viene pubblicata la libreria ARToolKit che permette la creazione delle prime soluzioni a basso costo basate su un comune pc con webcam [Loo07].

La Realtà Aumentata è dunque un concetto noto da diversi anni e molto utilizzato, oltre che nei film di fantascienza (ad esempio Terminator), anche in ambito militare e nelle missioni spaziali: in questi casi si tratta di dispositivi creati ad hoc per compiti specifici in progetti con un alto budget. Per esempio i caschi dei soldati di terra sono oggi dotati di un display che fornisce informazioni su cosa sta succedendo intorno a loro e su cosa stanno guardando. Di un dispositivo molto simile sono dotati coloro

che fanno manutenzione di dispositivi molto complessi o impossibilitati a tenere un manuale a portata di mano, ad esempio gli astronauti sulla stazione spaziale.
Negli ultimi anni la diffusione di dispositivi tascabili potenti quasi come i classici calcolatori da scrivania, dotati di svariati sensori e dell'accesso a internet ha permesso di cominciare a portare la Realtà Aumentata anche in ambito civile e alla portata di tutti. La diffusione dei sistemi operativi iOS e Android, che forniscono agli sviluppatori gli strumenti necessari a creare applicazioni di qualità, e un modello di distribuzione e installazione del software molto semplice, ha dato la spinta definitiva alla nascita di applicazioni general purpose di Realtà Aumentata come Layar (`www.layar.com`), Metaios's Junaio (`www.junaio.com`) e Wikitude (`www.wikitude.com`). Queste applicazioni aggregando le informazioni provenienti da diversi sorgenti permettono di aumentare e migliorare l'esperienza della realtà. Ad esempio passeggiando in città un'attività commerciale può essere aumentata con i suoi recapiti, i suoi prezzi o i suoi orari (tutte informazioni reperibili sulla rete).

Un ambito di sicuro successo secondo gli analisti sarà quello dei videogiochi: la Realtà Aumentata apre moltissime possibilità per migliorare il livello di immersione del giocatore nello scenario di gioco, proiettandolo nell'ambiente che lo circonda e, con l'aiuto di dispositivi mobili, anche lontano dall'apparentemente irrinunciabile (almeno adesso) monitor/televisione.
Stanno nascendo molte applicazioni specifiche in campi come il design, la medicina, l'architettura e l'elettronica di consumo (vedi Appendice A).
La potenza di calcolo a disposizione e il prezzo basso dei moderni smartphone sta spingendo verso la loro adozione perfino in ambito militare.

L'interesse per la Realtà Aumentata è cresciuto negli ultimi anni come mostrato in figura 1.4, tanto che Time magazine l'ha inserita nella top ten delle trend technologies del 2010.

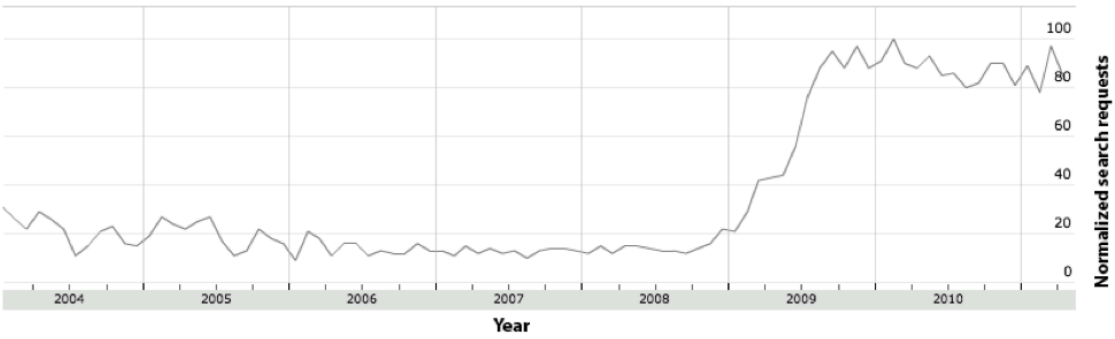


Figura 1.4: Andamento delle ricerche su Google per l'espressione augmented reality. Immagine da [PSPS11].

Nel mercato delle applicazioni per smartphone si sta diffondendo anche come funzionalità aggiuntiva per tutte le applicazioni che usano dati geolocalizzati. È un modo per un prodotto di emergere in alcune categorie ormai sature: ''Realtà Aumentata'' è infatti uno dei termini più ricercati sugli store per dispositivi mobili [RZ10].

Infine la Realtà Aumentata è al centro di quello che viene considerato il terzo grande passo evolutivo nella storia dei computer dopo i mainframe (un computer serve molte persone), e i personal computer (un computer per persona): l'Ubiquitus Computing (una persona ha a che fare con molti computer). Secondo questo paradigma, l'ambiente in cui viviamo sarà pieno di piccoli computer invisibili. Grazie alla possibilità di comunicare tra loro e di reagire ad eventi, grazie ai sensori di cui sono dotati, aggiungere questi dispositivi agli oggetti che ci circondano significa in qualche modo dotarli di intelligenza. In questa ipotesi i cosiddetti wearable computers hanno ruolo molto importante e fanno largo uso di Realtà Aumentata. Un esempio del futuro di questa tecnologia è in uno degli oggetti che hanno generato più interesse nell'ultimo anno: i Google Glass (`plus.google.com/+projectglass`). Project Glass è un programma di ricerca e sviluppo di Google con l'obiettivo di sviluppare un paio di occhiali dotati di realtà aumentata. Sono occhiali dotati di un

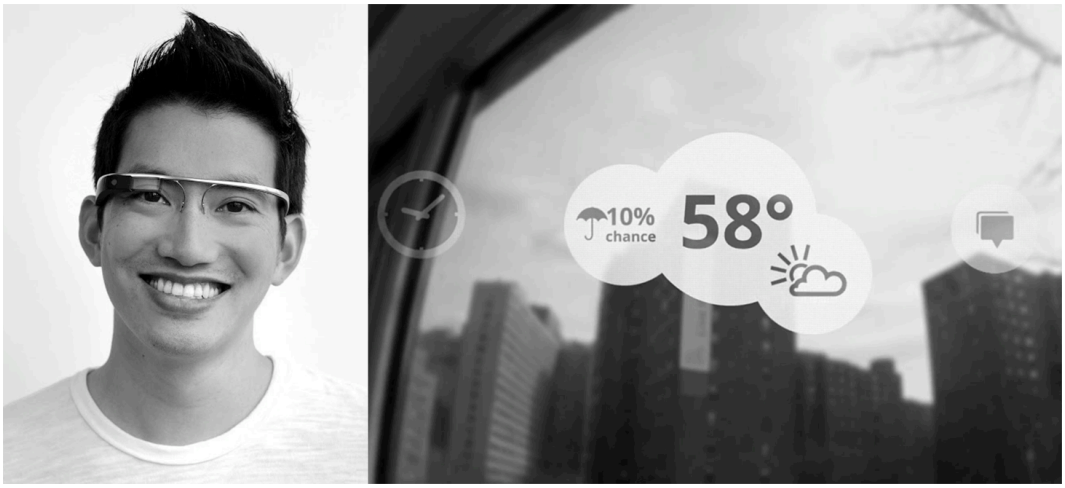


Figura 1.5: Google Project Glass.

Immagini da `plus.google.com/+projectglass`

processore, fotocamera, memoria integrata, ricevitori radio multipli e sensori quali il giroscopio, l' accelerometro e la bussola. Nonostante ciò, i Google Glass sono anche comodi da indossare: gli ultimi prototipi hanno un peso inferiore a quello di un paio di occhiali da vista e alcuni membri del team di Google hanno mostrato di averli indossati mentre facevano jogging e anche durante un salto con il paracadute da un aereo.

1.3 3logic

L'azienda dove ho sviluppato il progetto di tirocinio si chiama 3logic MK S.r.l. ed è nata a Pisa nel 2001. Si occupa principalmente di progettazione e integrazione di sistemi nei quali l'innovazione tecnologica ricopre un ruolo fondamentale.
Offre servizi rivolti alla grande e media impresa ed è da sempre impegnata in progetti di ampio respiro, sia con partner istituzionali (come CNR e Università) sia con altre aziende italiane.
3logic svolge attività principalmente in 3 aree:

- **Multimedia Library:** catalogazione contenuti e sistemi di fruizione multimediali;
- **Mobile Applications:** sviluppo di applicazioni mobili su dispositivi Windows Phone, iOS (iPhone/iPad) e Android;
- **Realtà Aumentata:** progettazione e sviluppo di sistemi digitali per musei e spazi espositivi, installazioni interattive anche di tipo immersivo.

Le ultime due in particolare riguardano questo lavoro in maniera molto diretta.

1.3.1 Mobile Applications

Negli anni, 3logic ha accumulato esperienza nello sviluppo di progetti per dispositivi mobili con particolare attenzione alla realizzazione di applicazioni per piattaforme mobili, quali Windows Phone, iOS e Android. Questo ambito ha portato la società ad affrontare le problematiche legate alla realizzazione e allo sviluppo di prodotti distribuiti tramite canali ufficiali, tra cui la certificazione dell'applicazione da parte del distributore e l'adozione di un processo produttivo standard per lo sviluppo e il test. Oltre a ciò, 3logic ha maturato esperienze nell'affrontare problematiche legate a prodotti utilizzabili da bacini di utenza molto ampi, senza particolari conoscenze informatiche, com'è quello degli utilizzatori dei dispositivi mobili.

1.3.2 Realtà Aumentata

3logic sperimenta varie tecnologie per la Realtà Aumentata: interfacce il più possibile ''naturali'', sensori il più possibile ''invisibili'', sistemi per la localizzazione ed il riconoscimento della posizione e dei gesti dei visitatori, software per il playback dei contenuti. Progetta inoltre sistemi interattivi per mostre, musei, eventi, spettacoli. Con queste tecnologie, 3logic ha realizzato applicazioni ad alto grado di interattività, sfruttando a fondo le possibilità offerte dall'hardware dei dispositivi (accelerometri, telecamera, gps, schermo multitouch) per arricchire l'esperienza dell'utente

nel loro utilizzo. Tutto ciò ha permesso anche di realizzare applicazioni per alcuni dei principali operatori mobili italiani, e di prendere parte a progetti di ricerca che fanno uso di Realtà Aumentata e visual computing soprattutto su iPhone ed Android OS.
In ambito turistico, 3logic si è occupata dello sviluppo della parte mobile all'interno del progetto VISITO Tuscany, focalizzato sullo sviluppo di tecnologie che permettono agli smartphone di riconoscere attraverso la fotocamera i monumenti che il turista sta visitando in modo interattivo. Nell'ambito di questo progetto si sono sviluppati i precedenti progetti di tirocinio sviluppati da 3logic con l'Università di Pisa e collegati a mobile [Mel09] e Realtà Aumentata [Bra10].

1.4 Obiettivi del tirocinio e struttura della relazione

Gli obiettivi di questo tirocinio sono:

- chiarire come viene affrontato il problema della realizzazione di un sistema di Realtà Aumentata in letteratura e quali sensori disponibili sui moderni smartphone (e in particolare quelli prodotti da Apple e quindi dotati di sistema operativo iOS) possono essere utilizzati per creare una applicazione che utilizza questa tecnologia (Capitolo 2);
- analizzare il sistema operativo iOS e le tecnologie a disposizione sul questa piattaforma e scegliere tra le varie implementazioni possibili la migliore in termini di prestazioni, manutenzione, flessibilità ed espandibilità futura del progetto (Capitolo 3);
- implementare una piattaforma di Realtà Aumentata per dispositivi con sistema operativo iOS (Capitolo 4).

Capitolo 2

Panoramica sulla Realtà Aumentata

Prima di cominciare a parlare del lavoro svolto è utile chiarire lo scenario in cui ci si inserisce, sia a livello del software e dei modi di affrontare il problema presenti in letteratura, sia al livello di hardware a disposizione che potrebbe essere utilizzato.

2.1 Software

La recente crescita di alcune discipline in ambito informatico come image processing, computer vision, edge detection, ecc. sono state fondamentali per l'accelerazione che ha avuto lo sviluppo della Realtà Aumentata in ambito civile e militare.
Anche se in questo tirocinio ci si concentrerà in particolare su applicazioni di AR per dispositivi mobile, quelle in cui tipicamente l'utente punta il suo smartphone in direzione dell'oggetto di suo interesse e lo visualizza ''aumentato'' sullo schermo del suo device, è utile vedere qual'è l'approccio al problema e le varianti descritte in letteratura.
Un sistema di Realtà Aumentata si articola in tre passi ripetuti continuamente, come mostrato in figura 2.1:

1. **Tracking:** fase nella quale il sistema rileva in tempo reale e in modo continuo la posizione e l'orientamento del device.

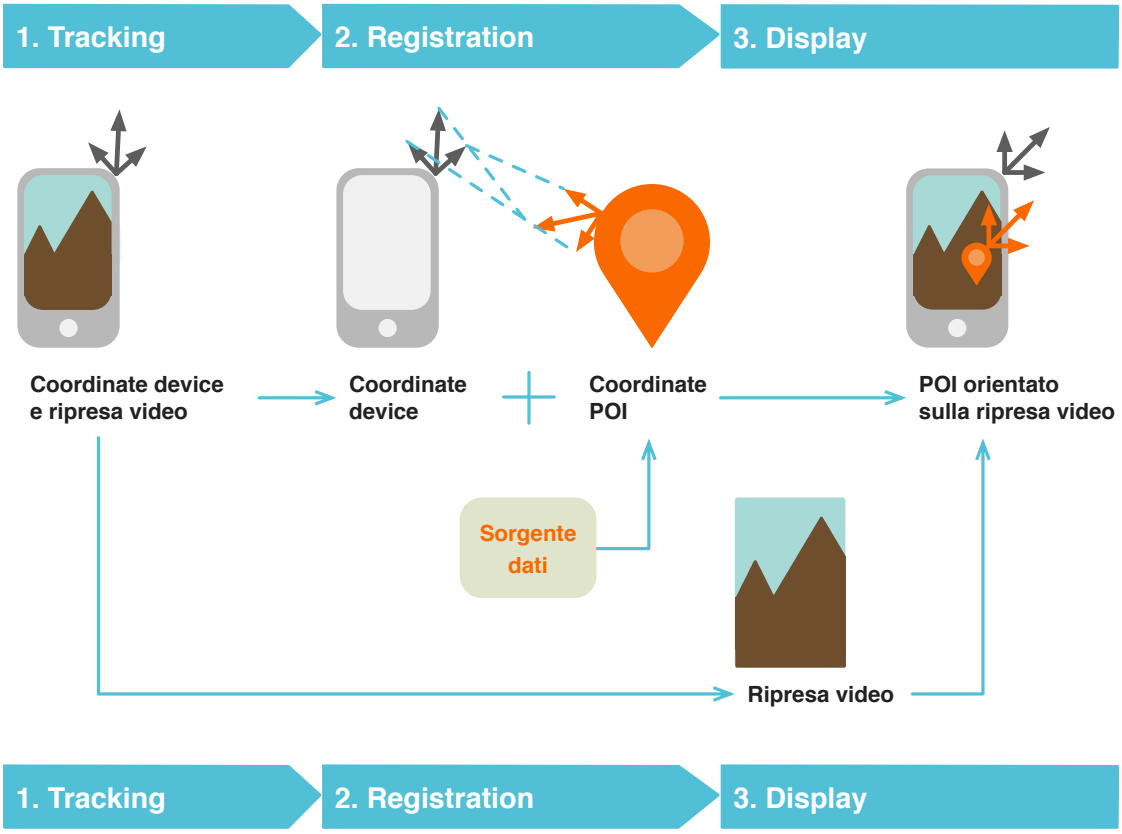


Figura 2.1: Ciclo di funzionamento.

2. **Registration:** allineamento tra le informazioni virtuali e il mondo reale.
3. **Display:** visualizzazione delle informazioni.

Inoltre, c'è la possibilità di permettere all'utente di interagire con la scena, aggiungendo altri parametri e modificando quindi ciò che viene mostrato. La velocità e l'efficienza della comunicazione tra i pezzi di software che gestiscono queste fasi è fondamentale per fornire un esperienza accettabile.

2.1.1 Tracking

Per rilevare quale sia l'oggetto da aumentare puntato dall'utente o, a seconda dei punti di vista, qual è l'orientamento del dispositivo, ci sono tre tipi di soluzioni che vengono in genere utilizzate. A seconda dell'applicazione e del dispositivo si possono fare scelte diverse, comprese soluzioni ibride:

- **Visual-based Marker Tracking:** rilevare con la fotocamera ben identificabili marker 2D come per esempio barcode o qrcode che contengono l'id dell'oggetto (o più raramente direttamente le informazioni da visualizzare). Ogni marker ovviamente è univoco e viene riconosciuto anche se ruotato. La principale controindicazione è che la ''scena'' deve essere preparata prima e la AR funzionerà solo in aree attrezzate allo scopo. Questa soluzione è l'ideale nelle seguenti condizioni:
  - interni o luoghi nei quali i sistemi di localizzazione non funzionano;
  - spazi di limitate dimensioni;
  - oggetti da aumentare particolarmente piccoli o vicini;
  - dispositivi general purpose in cui la potenza di calcolo a disposizione è molto limitata;
  - dispositivi non dotati di sensori ma solo di fotocamera.

  È il caso per esempio dell'applicazione Virtual Box Simulator mostrata in figura 2.2 e presente sul sito di USPS (United States Postal Service - `www.prioritymail.com`) che permette con il solo uso di una webcam e di un marker stampato di capire quale sarà la misura della scatola o della busta per l'oggetto che dovremo spedire da casa prima di recarsi all'ufficio postale.

- **Visual-based Natural Feature Tracking:** rilevare con la fotocamera elementi naturali o artificiali facilmente individuabili grazie a angoli e forme note. Il problema di questa modalità è l'altissima quantità di calcolo richiesto. Anche ipotizzando di spostare il carico su un server che fornisca la potenza di

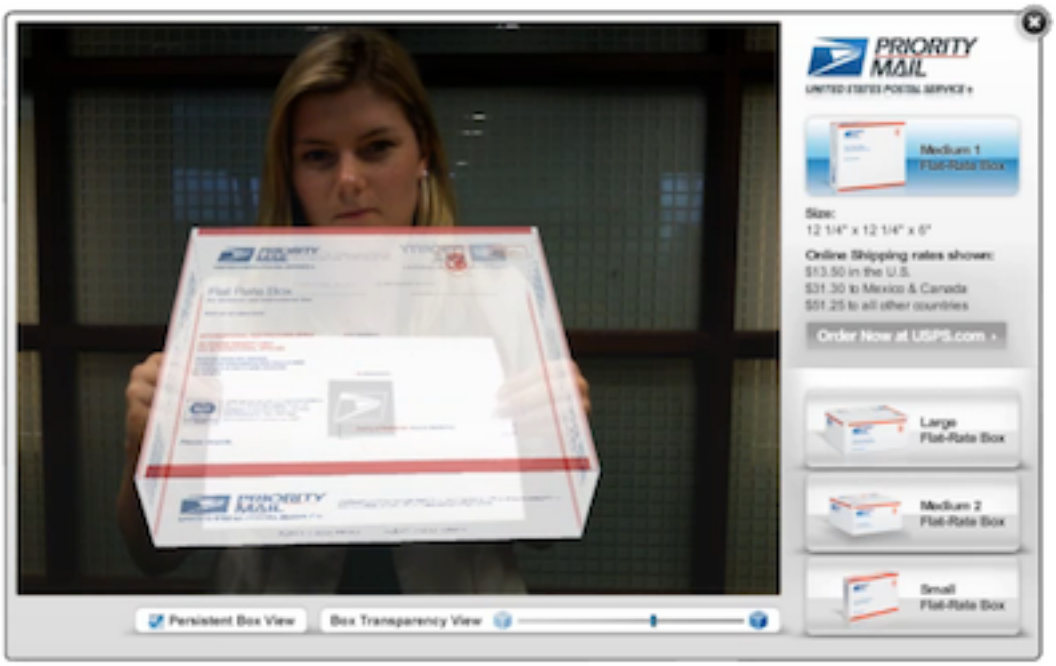


Figura 2.2: L'applicazione Virtual Box Simulator.

Immagine da `upload.wikimedia.org/wikipedia/commons/d/d4/USPS_virtual_box_application.png`.

Figura 2.3: Esempio di Realtà Aumentata basata su Natural Feature Tracking

L'applicazione Atol les opticiens utilizza il tracking basato sul riconoscimento dei tratti del viso per permettere all'utente di provare gli occhiali prima di recarsi in negozio.

Immagine da `itunes.apple.com/it/app/atol-les-opticiens/id410808066?mt=8`.

calcolo in remoto, un'applicazione mobile in questo momento non sarebbe in grado di fornire un'esperienza utente sufficientemente reattiva a causa dei tempi richiesti per scambiare grandi quantità di informazioni, come inviare un'immagine. Questa soluzione può essere ottimale in caso di:

- ampie aree da coprire;
- oggetti in movimento e/o non segnalabili con marker;
- presenza di hardware ad hoc particolarmente potente.

Viene adottata per esempio dai sistemi militari di Realtà Aumentata per le operazioni di terra che devono poter ''aumentare armi'' e mezzi nemici intorno a loro senza poterli segnare con un marker e senza sapere prima quale potrebbe essere la loro posizione.

- **Sensors Monitoring Tracking:** ricostruire la posizione dell'utente e l'orientamento del dispositivo a partire dai dati rilevati dai sensori presenti nel device (tipicamente uno smartphone) e utilizzare la fotocamera solo per proiettare le informazioni che aumentano l'oggetto puntato. Anche questa soluzione ha i suoi difetti: infatti i dati dei sensori possono essere non precisi o del tutto assenti (per esempio il GPS in ambienti chiusi), o sottoposti a interferenze (in particolare quelli che usano componenti magnetici).
  Dunque questa soluzione è conveniente nelle seguenti condizioni:
  - dispositivi dotati di vari sensori (come i moderni smartphone);
  - grandi spazi aperti che sarebbe impossibile contrassegnare manualmente con marker;
  - spazi a cielo aperto in cui i sensori di localizzazione possano funzionare.

  Queste condizioni sono alla base della gran parte delle applicazioni di Realtà Aumentata presenti su smartphone e che mirano a mappare il mondo reale a partire da grandi basi di dati geolocalizzati.

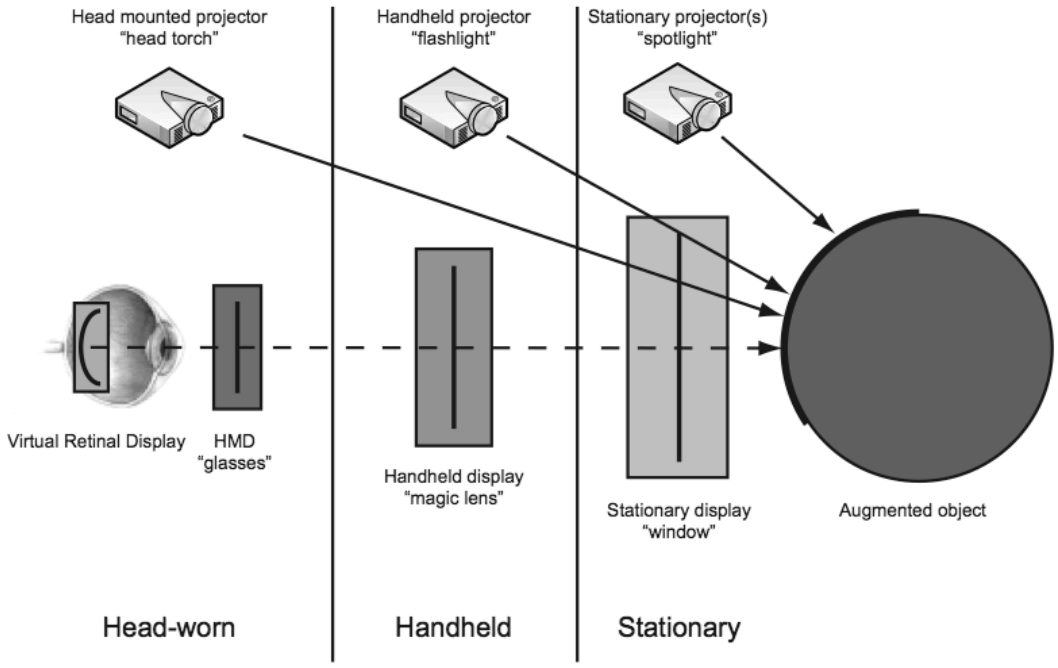


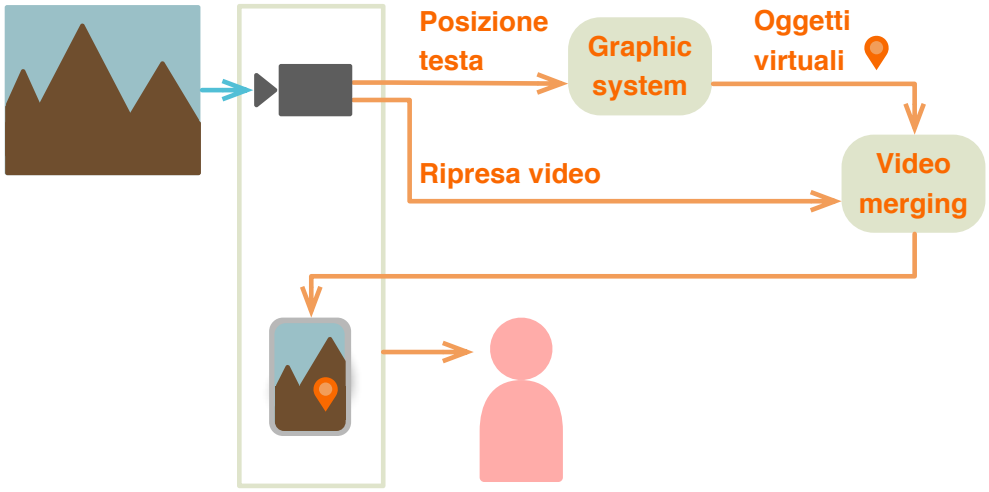
Figura 2.4: Le possibili metafore per implementare la fase Display.

Immagine da [Hen07].

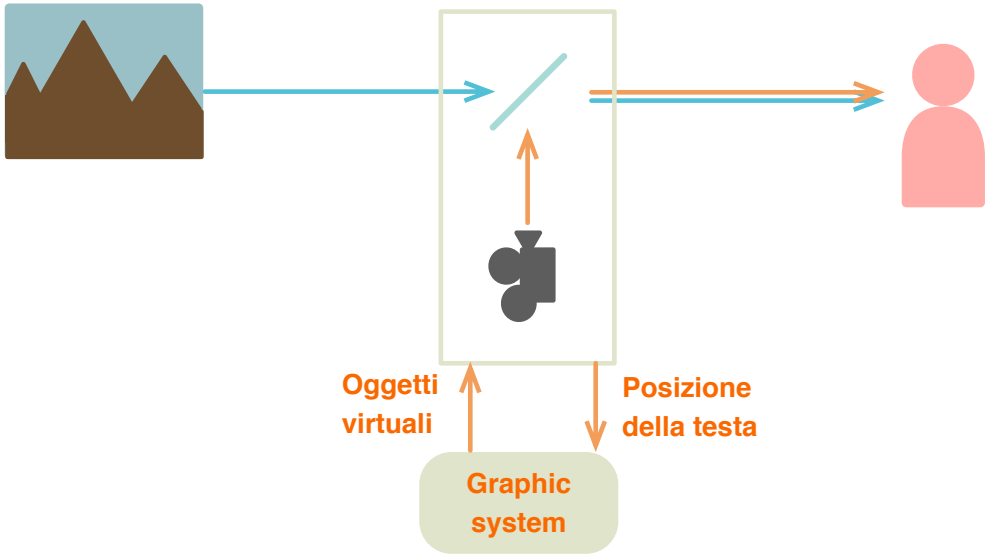
2.1.2 Display

Ci sono vari modi di realizzare la fase di Display, come descritto in figura 2.4.
Il modo più vecchio e diffuso per creare un sistema di Realtà Aumentata è quello di montare uno schermo sulla testa dell'utente attaccandolo a un casco o a delle cuffie (**HMD: Head-Mounted Display** o **HUD: Head Up Display**). Lo schermo può essere:

- **opaco:** se l'utente vede uno schermo su cui viene riportata la ripresa di una telecamera (in genere sulla sua testa) con le informazioni contestuali aggiunte:



- **trasparente:** nel caso in cui l'utente vede la realtà attraverso lo schermo e su questo vengono proiettati gli elementi di computer graphics, come nei Google Glass (vedi § 1.2):



Un'altra modalità possibile di implementazione di questa metafora, ma ancora ad un livello embrionale di sviluppo è il Virtual Retinal display che prevede di proiettare le informazioni direttamente sulla retina (per approfondire vedi [TJMI]).

L'approccio più comune usa invece la metafora chiamata **Magic Lens** mostrata in

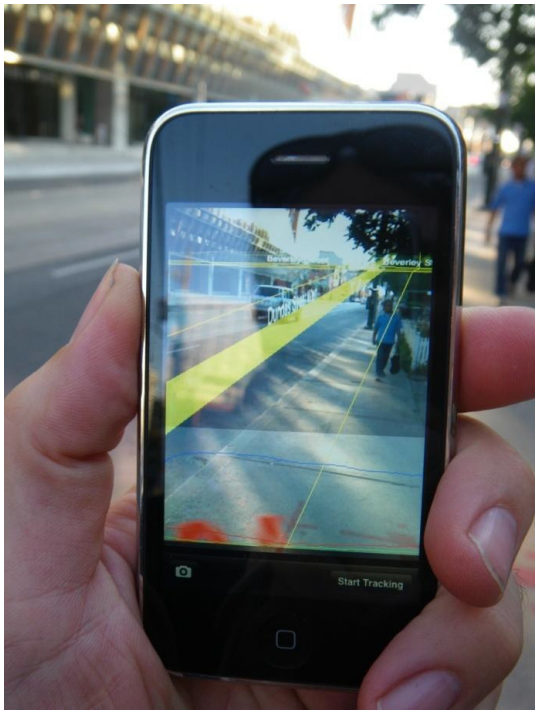


Figura 2.5: Tipico uso della metafora Magic Lenses in una applicazione di Realtà Aumentata per iPhone.

Il software mostra i nomi delle strade inquadrate sovrapponendoli all'immagine ripresa dalla fotocamera.
Immagine da [Wikb].

Figura 2.6: Mirracle, sistema di Realtà Aumentata in campo medico, utilizza la metafora magic mirror.

Immagine da `http://mirracle.de`.

figura 2.5. Le Magic Lens sono elementi di interfaccia o filtri 2D trasparenti o semi-trasparenti che forniscono visioni alternative degli oggetti visualizzati attraverso di essi. Possono essere utilizzate come delle normali lenti di ingrandimento: coprono solo una parte dello spazio visivo e mostrano l'effetto applicato solo su quella parte. Il risultato che si ottiene è lo spontaneo collegamento tra oggetti reali e virtuali aumentati. In questo caso la video camera si trova dietro lo schermo utilizzato per mostrare le informazioni. Il modo classico è quello di utilizzare uno smartphone come una lente di ingrandimento che invece di ingrandire l'oggetto visualizzato ne aumenta le informazioni.
In alternativa si può realizzare un sistema di Realtà Aumentata con la metafora detta **Magic Mirror**, mostrata in figura 2.6, una tecnica che prevede che veda se stesso nel display come fosse uno specchio che aumenta la realtà. In questo caso in genere si posiziona una videocamera sopra lo schermo e rivolta verso l'utente; anche la fotocamera frontale di uno smartphone potrebbe essere utilizzata a questo scopo (come nell'esempio riportato in figura 2.3).

Infine esiste la metafora detta **Spatial** (o anche Stationary) **Augmented Reality** che non prevede l'utilizzo di schermi da parte dell'utente perché i contenuti aggiuntivi sono proiettati direttamente sugli oggetti reali (vedi [BRI05] e [MTUK95]).

2.2 Hardware

I dispositivi portatili attuali sono oggetti perfetti per realizzare l'AR perché hanno le seguenti caratteristiche hardware:

- **Processing:** gli smartphone hanno potenza sufficiente per algoritmi di computer vision, decodifica video e grafica 3D attraverso schede video dedicate che permettono prestazioni grafiche avanzate e calcolo in virgola mobile;
- **Imaging:** questi dispositivi sono dotati di uno schermo di dimensioni gene-

rose capaci di visualizzare milioni di colori. Inoltre sono dotati di una o più fotocamere che permettono di fare video e foto con una qualità paragonabile ad una macchina fotografica compatta;

- **Positioning:** molti dispositivi sono inoltre dotati di sensori che permettono di conoscere con precisione la posizione e l'orientamento del dispositivo (indispensabili per utilizzare la tecnica del Sensors Monitoring per la fase di Tracking).

Per questo i requisiti hardware minimi per l'AR sono un display, una fotocamera per il tracciamento oppure dei sensori di posizione per la posizione e una cpu. I sensori a cui si fa riferimento sono principalmente: GPS, accelerometro, giroscopio e bussola.

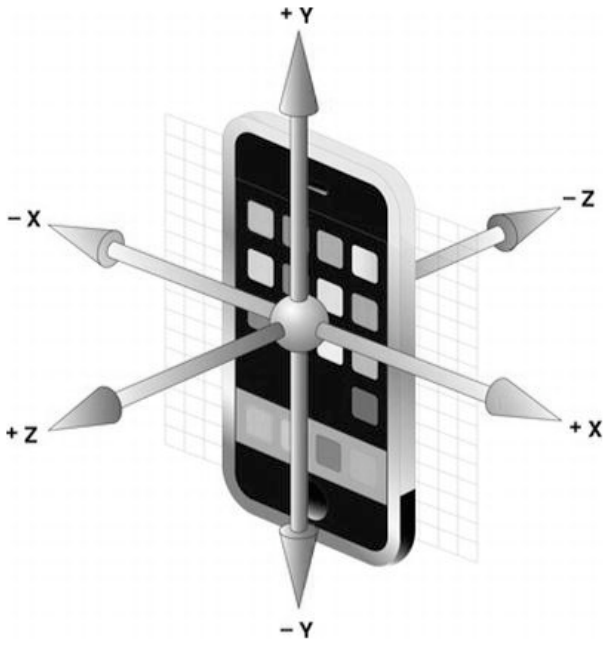


Figura 2.7: Gli assi X, Y e Z misurati dall'accelerometro.

Immagine da [Roc11].

L'accelerometro (vedi figura 2.7) misura su tre assi l'orientamento di una piattaforma stazionaria relativamente alla superficie della terra. Se cade o ha accelerazione

costante indica 0 perché non riesce a distinguerlo dall'accelerazione di gravità. Il giroscopio invece misura il cambiamento dell'angolo di rotazione degli assi e lavora generalmente su sei assi, ma quello incluso nei dispositivi usati per lo sviluppo ne ha solo tre. In pratica il primo misura l'accelerazione sugli assi, il secondo misura e mantiene l'orientamento. Il magnetometro o bussola misura la forza del campo magnetico intorno al dispositivo. Ipotizzando di non avere altri campi magnetici quello misurato sarà il campo magnetico terrestre. Il sensore ovviamente rileverà il nord magnetico e non il nord geografico (comunque sempre calcolabile a partire dal primo), indicando in quale direzione è rivolto il dispositivo.

Capitolo 3

Tecnologie utilizzate

In questo capitolo sono analizzate e descritte le tecnologie utilizzate, sia quelle imposte dall'azienda cioè il sistema operativo e i dispositivi da utilizzare, sia quelle scelte e il perchè.

3.1 iOS e Objective-C

iOS (precedentemente chiamato iPhone OS) è un sistema operativo sviluppato da Apple per i suoi dispositivi multitouch cioè iPhone, iPod touch e iPad. E' disponibile solo per dispositivi Apple con processori ARM. Come Mac OS X è una derivazione di UNIX (famiglia BSD) e usa un microkernel XNU Mach basato su Darwin OS. I due sistemi operativi di Apple inoltre condividono i linguaggi di programmazione ''ufficiali'' (C/C++ e Objective-C), molte API e l'ambiente di sviluppo XCode (vedi § 3.1.1).

Il linguaggio di programmazione utilizzato durante il tirocinio è appunto Objective-C, linguaggio orientato agli oggetti, sviluppato da Brad Cox alla metà degli anni ottanta presso la Stepstone Corporation e estensione a oggetti del linguaggio C (con cui mantiene la completa compatibilità). La sua diffusione è principalmente legata al framework OpenStep di NeXT e al suo successore Cocoa, presente nel sistema operativo Mac OS X di Apple.

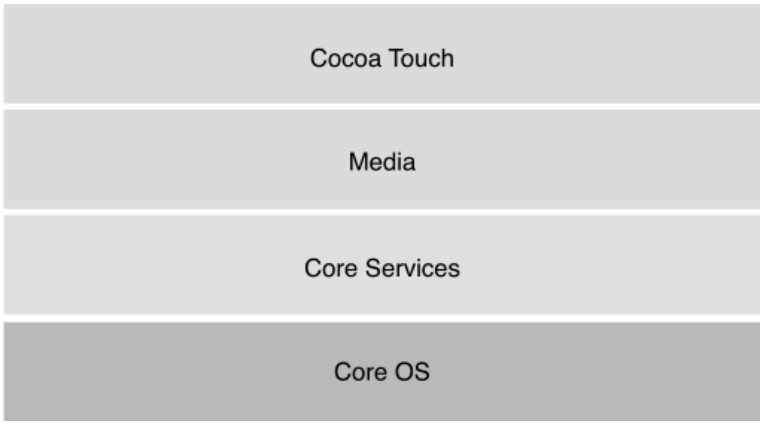


Figura 3.1: I layer del sistema operativo iOS.
Immagine da [Appf].

Sul kernel poggiano una serie di servizi ed API organizzati per livelli di astrazione. Come si vede in figura 3.1 al livello più basso troviamo lo stack di derivazione BSD e funzioni per IO, threading, gestione processi, sqlite, ecc. Dal livello Core Services in poi troviamo tecnologie sviluppate da Apple, in particolare Core Foundation che, come indica il nome, è la base su cui sono sviluppate molte delle API di livello più alto.
Media contiene principalmente API C (Core Audio, Quartz, OpenGL ES etc etc) con la notevole eccezione di Core Animation, un motore avanzato per la gestione di animazioni sviluppato in Objective-C. Al livello più alto troviamo infine Cocoa Touch, API completamente Objective-C composta a sua volta da Foundation e UIKit.
Foundation contiene la definizione di NSObject, la root class da cui ereditano tutte le classi presenti in Cocoa, le classi legate alla gestione/manipolazione/storage dei dati, wrapper object-oriented per alcuni dei livelli più bassi ed in genere tutto quello che non riguarda l'interazione con l'utente o la visualizzazione.
Questi due aspetti sono di competenza di UIKit, la prima fra le API citate fino ad ora ad essere specifica di iOS: qui troviamo infatti le classi legate alla gestione del multitouch ed alla renderizzazione su schermo, oltre ad una serie di widget grafici legati

all'UI.

In iOS, la contrario di Mac OS X, non è disponibile un garbage collector. Apple consiglia in sua sostituzione un sistema statico che funziona a tempo di compilazione chiamato ARC (Automatic Reference Counting). Infine iOS utilizza OpenGL ES 1.1 e OpenGL ES 2.0 per la grafica avanzata e dalla versione 4.0 il sistema supporta il multitasking.

Caratteristica utilizzata estensivamente per la comunicazione ad eventi è l'uso di protocolli. Un protocollo è un insieme di dichiarazioni di metodi che possono essere implementati dalle classi. Questi metodi definiscono un'interfaccia da utilizzare per comunicare che è avvenuto un evento. Quando una classe implementa i metodi di un protocollo, si dice che è conforme a tale protocollo. In Objective-C, i protocolli sono usati per creare oggetti delegate (delegati), cioè oggetti che agiscono in coordinamento o per conto di un altro oggetto. Se per esempio si è interessati a tracciare gli spostamenti dell'utente, sarà sufficiente implementare il metodo locationManager:didUpdateLocations: e indicare la propria classe come delegato del protocollo CLLocationManager per far sì che il metodo in questione venga chiamato ogni qual volta la posizione cambia.

Particolare è il modello di distribuzione del software di terze parti scelto da Apple, quello di uno store di applicazioni, che ha lanciato una moda: ogni sistema operativo moderno, mobile e non, ha un suo store. Ogni sviluppatore o azienda iscritti al programma a pagamento, può inviare la propria applicazione compilata ad Apple insieme ad una scheda con descrizione e qualche schermata del prodotto. Apple vaglia l'applicazione in base ad alcuni criteri contenuti nell'accordo fatto firmare allo sviluppatore al momento dell'iscrizione e infine lo pubblica nella sua ''vetrina virtuale''.

3.1.1 XCode

XCode (mostrato in figura **??**) è l'ambiente di sviluppo integrato (Integrated development environment, IDE) sviluppato da Apple per agevolare lo sviluppo di software per Mac OS X e iOS ed è l'IDE utilizzato per realizzare questo progetto di tirocinio. È fornito gratuitamente da Apple in bundle con i sistemi operativi desktop o scaricabarile dal Mac App Store.
XCode include due compilatori, il classico GCC e il più moderno LLVM, che sono in grado di compilare codice C, C++, Objective-C/C++ e Java e altri linguaggi meno comuni. Una delle caratteristiche interessanti è il supporto per la compilazione distribuita sulle macchine presenti nella stessa rete locale e configurate a questo scopo.
XCode contiene un tool grafico per realizzare interfacce grafiche (chiamato Interface Builder), un simulatore che virtualizza i sistemi operativi mobili, vari strumenti di test e dalla versione 3.1 è anche lo strumento per sviluppare le applicazioni native per iOS.

3.1.2 Come funziona una applicazione iOS

Il ciclo di vita di una applicazione su iOS è mostrato in figura 3.2. Come in ogni sistema multitasking è caratterizzato dalla transizione tra differenti stati in cui si può trovare una applicazione. Il passaggio tra i diversi stati è regolato dal sistema operativo in risposta agli eventi utente o a eventi di sistema. Nel caso di una applicazione iOS questi stati sono:

- **Not Running:** l'applicazione non è ancora stata lanciata oppure è stata terminata dal sistema operativo o da un errore che ha causato un crash.
- **Active:** l'applicazione è in esecuzione e sta ricevendo eventi.
- **Inactive:** l'applicazione è in esecuzione, ma non sta ricevendo eventi a causa di un evento bloccante generato dal sistema operativo come la ricezione

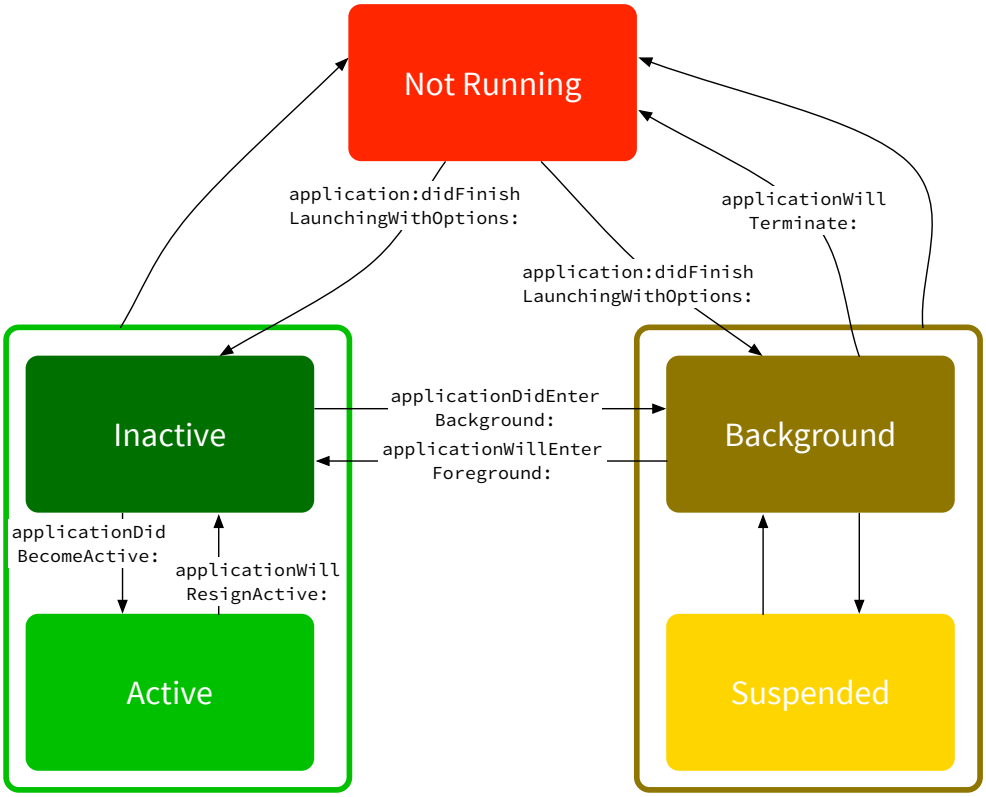


Figura 3.2: Ciclo di vita di una applicazione iOS.

I nomi vicino alle frecce si riferiscono ai nomi degli eventi utente, intercettabili a livello applicativo. Le frecce non accompagnate da alcun evento indicano transizioni dovute ad eventi di sistema e non gestibili dall'applicazione.

di una telefonata o di un SMS oppure quando il dispositivo ha lo schermo bloccato.

- **Background:** l'applicazione è in background e non riceve eventi utente. Può eseguire codice.
- **Suspended:** l'applicazione è in background ma non può eseguire codice

In iOS una sola applicazione alla volta può essere in foreground, ovvero negli stati Active o Inactive.

Inizialmente ogni applicazione si trova nello stato Not Running. Selezionando l'icona dell'applicazione il sistema operativo carica l'immagine di apertura e poi lancia la funzione main() che chiama a sua volta la funzione UIApplicationMain() che si occupa di creare il singleton UIApplication, caricare il file dell'interfaccia e instanziare il delegato dell'applicazione. A questo punto l'applicazione passa nello stato Inactive. Viene avviato l'event loop, che riceve gli eventi e si occupa di inviarli a chi li deve gestire. L'applicazione diventa quindi Active e può ricevere gli eventi e aggiornare l'interfaccia grafica.
Se l'utente preme il tasto home l'applicazione torna prima nello stato Inactive e poi passa nello stato Background. Infine l'applicazione passa allo stato Suspended in cui è sostanzialmente congelata.
Quando l'utente rilancia l'applicazione, il sistema la sposta prima nello stato Inactive e poi rapidamente nello stato Active.
Quando un'applicazione che si trova nello stato Active viene interrotta a causa della ricezione di una telefonata, di un SMS di un avviso di calendario, viene temporaneamente spostata nello stato Inactive. Rimarrà in questo stato fino a quando l'utente non decide di ignorare o accettare l'interruzione. Se l'utente ignora l'interruzione, l'applicazione torna nello stato Active. Se invece accetta l'interruzione, l'applicazione viene spostata nello stato Background.

Attraverso il delegato dell'applicazione possiamo essere informati delle transizioni tra i vari stati che un'applicazione può assumere durante il suo ciclo di vita. In particolare, il delegato dell'applicazione può implementare questi metodi:

- **application:didFinishLaunchingWithOptions:** E' il metodo che viene chiamato subito dopo che l'applicazione è stata lanciata, cioè nel passaggio da Not Running a Inactive.
- **applicationWillResignActive:** E' il metodo che viene chiamato appena prima del passaggio dallo stato Active a Inactive

- **applicationDidBecomeActive:** E' il metodo che viene chiamato subito dopo il passaggio dallo stato Inactive a Active.
- **applicationDidEnterBackground:** E' il metodo che viene chiamato non appena l'applicazione è entrata nello stato Background.
- **applicationWillEnterForeground:** E' il metodo che viene chiamato appena prima del passaggio dallo stato Background a Inactive.
- **applicationWillTerminate:** E' il metodo che viene chiamato quando l'applicazione è nello stato Background (non Suspendend) e il sistema necessita di terminarla per qualche motivo.

Si noti che le transizioni tra Background e Suspended avvengono senza che venga chiamato alcun metodo sul delegato. Stesso discorso quando l'applicazione è in background e l'applicazione viene terminata dall'utente o dal sistema operativo.

3.1.3 Ciclo di vita del UIViewController

La classe UIViewController fornisce il modello per la gestione del modello MVC (Model View Controller) nella applicazioni iOS. Anche il ciclo di vista di View Controller è gestito in base agli eventi del suo ciclo di vita e si articola nei seguenti passi, riportati in figura 3.3:

- **ViewDidLoad** Viene chiamato quando si crea una classe. E' il momento ideale in cui impostare e creare oggetti che rimarranno nella vista per tutto il ciclo di vita del UIViewController.
- **ViewWillAppear:** - Questo metodo invece viene chiamato prima che la vista venga mostrata all'utente, ogni volta che sta per essere mostrata, dato che ci si potrebbe muovere tra le viste e visualizzala o meno. E' l'ideale per gestire tutti quelli oggetti che devono essere ripristinati o mostrati/nascosti ad ogni visualizzazione.

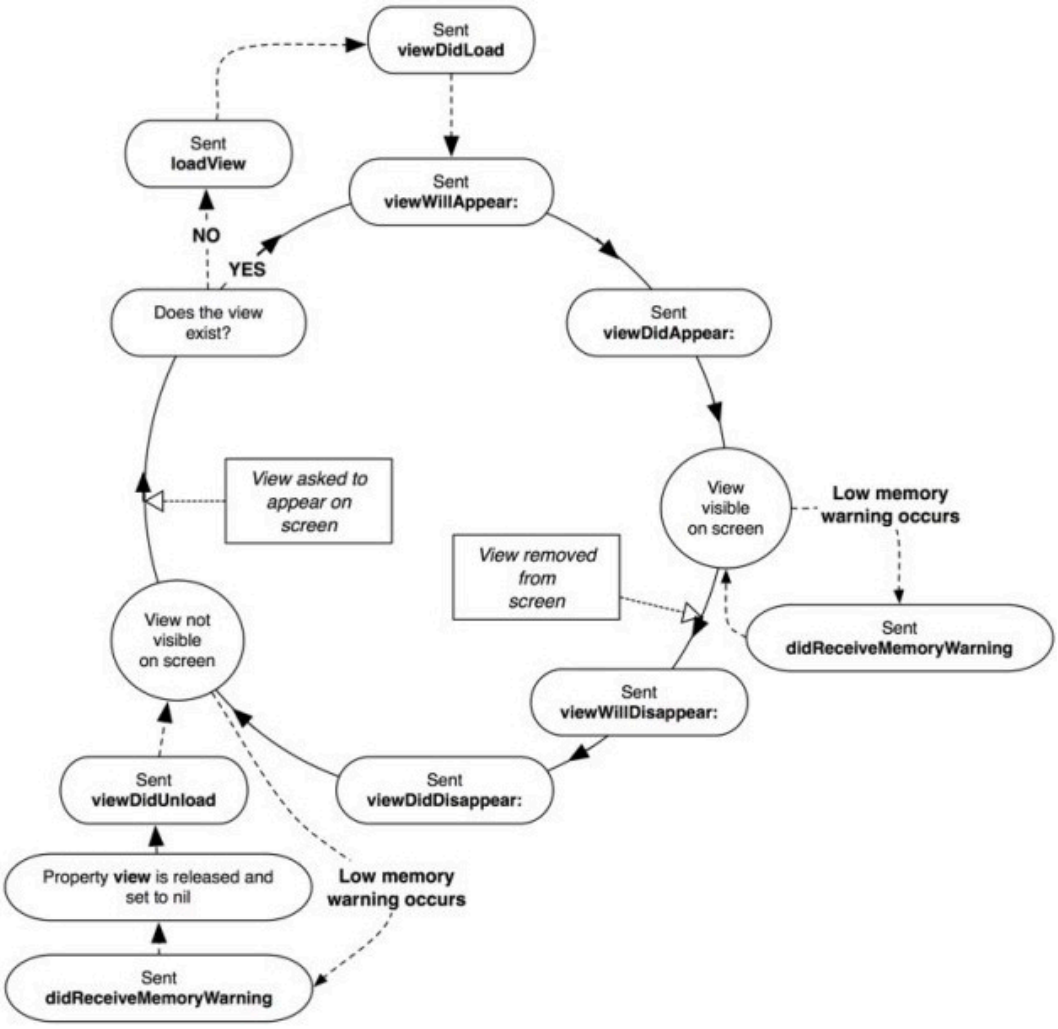


Figura 3.3: Ciclo di vita di un UIViewController.

Immagine da `blog.libingcun.com/images/post_images/UIViewController_lifecycle.gif`

- **ViewDidAppear:** - quest'altro invece viene eseguito dopo che la vista è apparsa, ed è il luogo ideale per far partire una animazione o caricare dati da una sorgente esterna.
- **ViewWillDisappear:**/**ViewDidDisappear:** - Specularmente agli ultimi due descritti esistono due metodi che vengono eseguiti ogni volta prima o dopo che

la vista scompaia.

- **ViewDidUnload**/**Dispose** - Questo metodo viene chiamato quando la vista viene dismessa o ed è il luogo ideale dove inserire operazioni di pulizia della memoria.
- **didReceiveMemoryWarning** - Questo metodo viene chiamato quando il sistema operativo notifica una mancanza di memoria e il programmatore deve rilasciare tutti gli oggetti non critici che ha in memoria.

3.2 Librerie di AR disponibili per iOS

Al momento di scegliere quale tecnologia utilizzare per implementare la nostra piattaforma di Realtà Aumentata ci si sono aperte tre possibilità:

- utilizzare framework già pronti dedicati alla Realtà Aumentata come ARToolkit, iPhone ARkit, String, Qualcomm QCAR, ecc.;
- utilizzare una libreria per la creazione di videogiochi come cocos2D o Unity;
- utilizzare le librerie base fornite dal sistema operativo.

Dopo un'attenta analisi dei vantaggi e degli svantaggi di ogni soluzione si è deciso di utilizzare quest'ultima soluzione. Passiamo in rassegna i pro e i contro di ogni soluzione.

3.2.1 Framework di terze parti per AR

Sono disponibili decine di librerie per implementare la Realtà Aumentata. La più diffusa è sicuramente ARToolkit (`http://www.hitl.washington.edu/artoolkit/`). Un confronto delle principali soluzioni disponibili è reperibile in [com].
Hanno il vantaggio di aiutare lo sviluppatore coprendo le fasi di Tracking e Registration (e qualche volta anche di Display). Includono spesso openCV (libreria per computer vision) e permettono di creare app con poche righe di codice, molto spesso

sono multipiattaforma e quindi portabili facilmente su altri sistemi operativi.
Questa soluzione ha diversi difetti: essendo una libreria che viene aggiunta al sistema operativo non avrà le stesse prestazioni di una soluzione nativa e aumenterà la dimensione del pacchetto compilato, inoltre non essendo ufficiale ma sviluppata da terze parti potrebbe avere dei problemi con gli aggiornamenti del sistema operativo o dell'ambiente di sviluppo (come smettere di funzionare o dare problemi a tempo di esecuzione e/o di compilazione) e costringere quindi ad aspettare un nuovo rilascio da parte di chi mantiene la libreria; infine non c'è nessuna certezza sul fatto che venga sviluppata e manutenuta sul lungo periodo.

3.2.2 Librerie per videogiochi

Nell'unico libro ad oggi pubblicato e dedicato alla Realtà Aumentata su iOS ([Roc11]) viene utilizzato cocos2D come libreria realizzare la fase di Display. Unity (`http://unity3d.com/`) e cocos2D (`http://www.cocos2d-iphone.org/`) sono i principali framework presenti sulla piattaforma iOS per la creazione di videogiochi e applicazioni interattive.
I vantaggi di questa soluzione sono sicuramente la maggiore facilità con cui si possono creare animazioni e oggetti di computer graphics. Entrambe le soluzioni sono multipiattaforma e quindi garantiscono portabilità verso altre piattaforme sia mobili come Android, Windows Phone, Bada, sia Desktop come Linux e Windows. Questa soluzione ha sostanzialmente gli stessi difetti della precedente.

3.2.3 Librerie di iOS

L'utilizzo di librerie di terze parti è stato fondamentale fino all'uscita della versione 3 di iOS. Dalla questa versione in poi infatti Apple ha cominciato ad includere una serie di API che hanno facilitato molto il lavoro degli sviluppatori. L'evoluzione delle API è mostrato in tabella 3.1. In particolare a partire da iOS 4 è stato possibile iniziare a lavorare con i dati grezzi che arrivavano dalla fotocamera e dalla versione 5

è possibile avere una rappresentazione dell'orientamento del dispositivo senza dover manipolare e incrociare i dati dei sensori (vedi il § 3.3.3).
Questo approccio ribalta sostanzialmente i pro ed i contro degli altri due: le librerie native hanno prestazioni migliori, non occupano spazio e garantiscono una compatibilità completa con il sistema operativo, al costo di una maggiore complessità. Si è optato per questa soluzione per la sua maggiore longevità e il suo supporto da parte del produttore.

	iOS 3	iOS 4	iOS 5
Libreria per Cattura Video	UIImagePickerController	AVFoundation	AVFoundation
Libreria per Orientamento	UIAccelerometer delegate	Core Motion (Accelerometer, Gyros)	Core Motion (CMDeviceMotion, CMAttitude)
Libreria per Localizzazione	LocationManager (GPS, Compass)	LocationManager (GPS, Compass)	LocationManager (GPS, Compass)

Tabella 3.1: Evoluzione delle librerie utili per la realtà aumentata in iOS

3.3 Framework utilizzati

Come indicato nel paragrafo 3.2 si è deciso di affidarsi esclusivamante a librerie di sistema. I framework utilizzati nella versione definitiva del codice sono descritti nei paragrafi seguenti.

3.3.1 AVFoundation

AV Foundation [Appa] è uno dei framework del sistema operativo che può essere usato per esaminare, creare, editare e codificare contenuti multimediali time-based oppure gestire e riprodurre flussi multimediali in input e manipolarli in tempo reale (vedi figura 3.4). Nello specifico gli oggetti da istanziare per gestire il flusso dall'input all'output (vedi figura 3.5):

- un'istanza di **AVCaptureSession** per gestire e coordinare flussi input e output;
- un'istanza di **AVCaptureDevice** che rappresenta l'input device (fotocamera, microfono, ...);
- un'istanza di una sottoclasse concreta di **AVCaptureInput** per configurare le porte dal device di input;
- un'istanza di una sottoclasse concreta di **AVCaptureOutput** per gestire l'output in un file o in flusso video oppure un'istanza di **AVCaptureVideoPreviewLayer**.

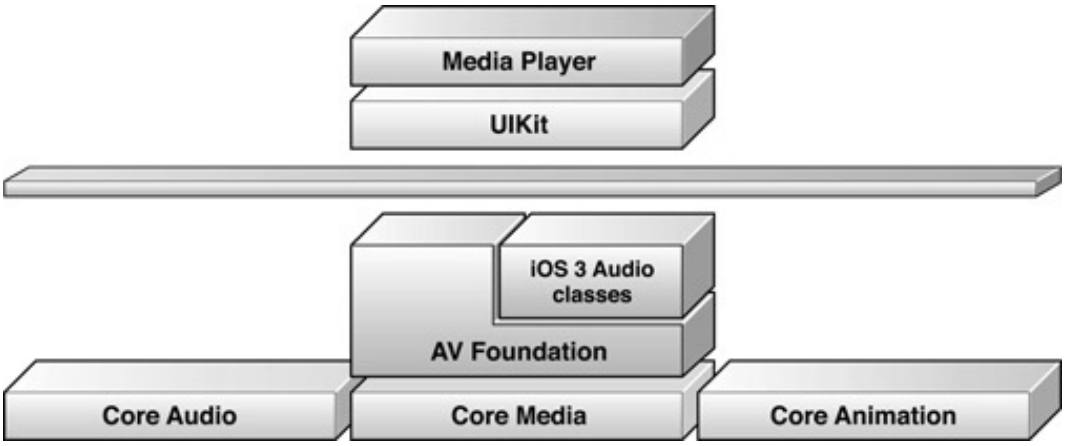


Figura 3.4: AVFoundation stack.

Immagine da [Appa].

3.3.2 Core Location

Il framework Core Location [Appc] permette di determinare la posizione e la direzione del dispositivo utilizzando rispettivamente il GPS e la bussola. Il framework utilizza i sensori disponibili per fornire le informazioni più precise possibili, indipendentemente da quale sia il dispositivo in uso creando così un layer che permette allo

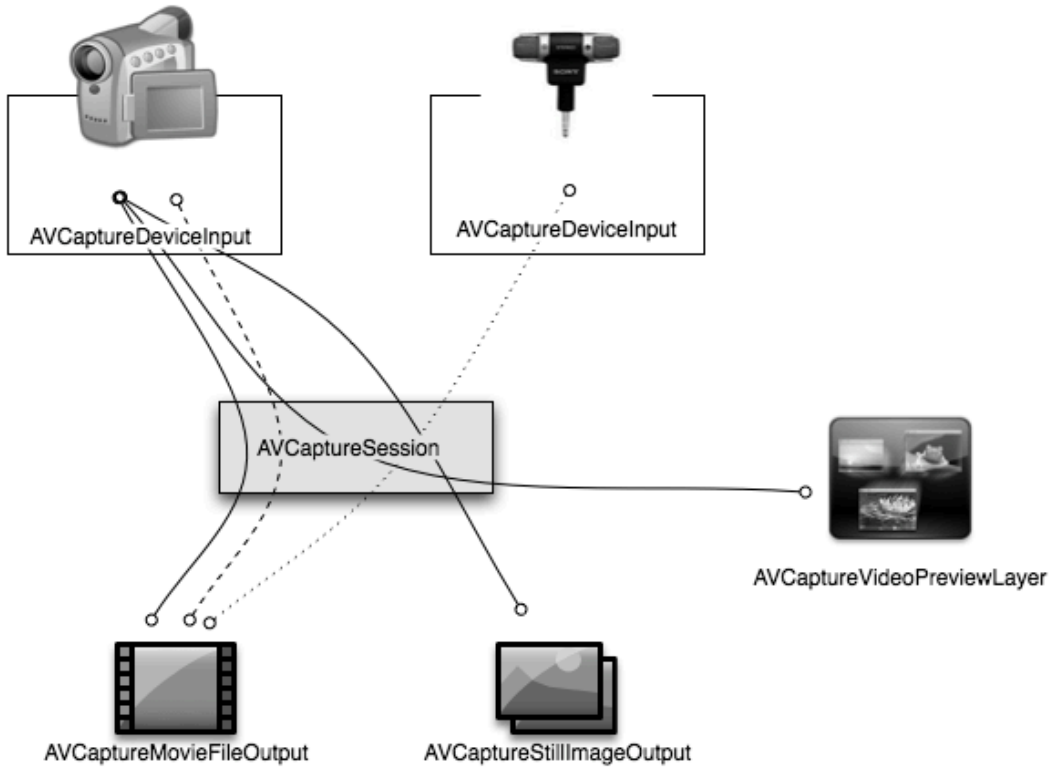


Figura 3.5: AVFoundation Media Capture.

Immagine da [Appb].

sviluppatore di non doversi preoccupare delle differenze hardware dei vari dispositivi. Gli oggetti che rappresentano la posizione generati da CLLocation Manager si chiamano CLLocation e contengono le coordinate geografiche, l'altitudine, la precisione della misurazione e il timestamp della rilevazione. Inoltre in iOS contengono anche la velocità istantanea e l'angolazione della bussola.

3.3.3 Core Motion

Il framework Core Motion [Appd] permette di ottenere i dati di spostamento del device e di elaborarli. Utilizza accelerometro e giroscopio. Attraverso la classe CMMotionManager si può scegliere di ricevere i dati dei singoli sensori oppure avere

una rappresentazione dell'orientamento del dispositivo ottenuto dalle informazioni combinate dei vari sensori. Come mostrato in figura 3.6 Core Motion definisce una classe manager CMMotionManager e tre classi le cui istanze includono varie misurazioni di movimento:

- **CMAccelerometerData** incapsula una struttura dati che mantiene la misura dell'accelerazione sui tre assi, in pratica fornisce i dati dell'accelerometro;
- **CMGyroData** incapsula una struttura dati che consente di conoscere il grado di rotazione del dispositivo sui tre assi. Sono i dati derivati dal giroscopio;
- **CMDeviceMotion** invece mantiene i dati di movimento del device processati a partire da entrambi i sensori. L'algoritmo si occupa di fondere i dati provenienti dalle due sorgenti fornendo una misura recisa dell'orientamento del dispositivo, un grado di rotazione assoluto sugli assi, la direzione della gravità e l'accelerazione eventualmente impressa al dispositivo dall'utente. L'oggetto **CMAttitude** incluso contiene proprietà che forniscono diverse misure

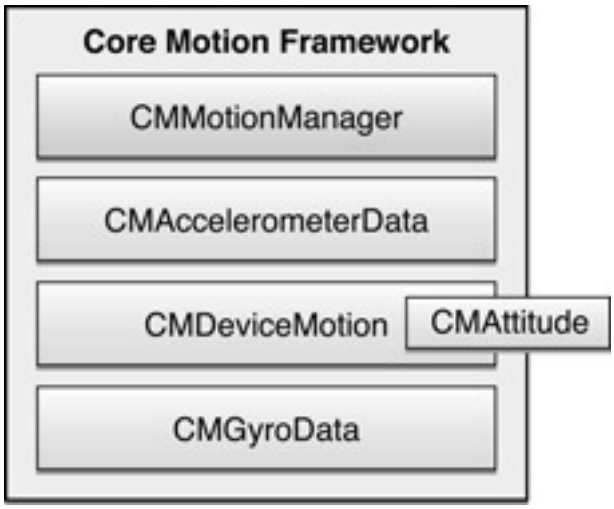


Figura 3.6: Core Motion classes.

Immagine da [Appd].

della posizione, compresa quella che utilizza gli angoli di Eulero (o rpy, vedi Appendice B) e quella con i quaternioni.

Anche in questo caso la classe definisce un protocollo ma si può decidere di ricevere gli aggiornamenti non in modalità ''ad evento'' (push) ma su richiesta (pull). In genere si usa la prima modalità in caso di applicazioni che devono collezionare dati senza perderne nessuno, il secondo in caso di situazioni best-effort dove il real time è più importante di perdere qualche informazione.

Capitolo 4

Progettazione e realizzazione

In questo capitolo è descritto il processo che ha portato alla realizzazione della piattaforma di Realtà Aumentata per iOS.

4.1 Progettazione

La richiesta da parte dell'azienda è stata di costruire una piattaforma di Realtà Aumentata, cioè non un'applicazione specifica ma un modulo software sufficientemente indipendente e flessibile da poter essere inserito in varie applicazioni. In merito alle varianti descritte nei capitoli precedenti, il prodotto creato fa riferimento ai seguenti modelli:

- **Tracking** effettuato con la modalità Sensors Monitoring Tracking (vedi § 2.1.1);
- **Display** che usa la metafora Magic Lens (vedi § 2.1.2).

Per prima cosa si è cominciato definendo un modello dei dati e un modo per interpretarli (§ 4.2).
Poi si è creata una prima versione dell'applicazione servita anche alla scoperta e alla conoscenza del linguaggio e della piattaforma (§ 4.3). In un secondo momento si è proceduto migliorandone alcuni aspetti (§ 4.4). Prendendo in considerazione

quanto presente in letteratura e descritto nel Capitolo 2, il lavoro svolto in entrambi i casi è stato diviso nelle seguenti fasi:

- **fase di tracking** (§ 4.3.1 e 4.4.1): recuperare la posizione e l'orientamento del dispositivo attraverso i sensori;
- **fase di registration** (§ 4.3.2 e 4.4.2): allineare i punti di interesse alla ripresa della fotocamera;
- **fase di display** (§ 4.3.3 e 4.4.3): mostrare la ripresa della fotocamera in tempo reale aumentata con le informazioni contestuali.

4.2 Recupero dati

La prima cosa da fare dunque è stata creare la lista di elementi da visualizzare con la Realtà Aumentata. Ognuno di questi Punti di Interesse (o POI, Point of Interest in inglese) ha delle proprietà come nome, coordinate, foto, ecc. Per la semplicità di gestione e l'universalità del formato, è stato scelto il Document Type Definition (DTD) definito da Apple per questo tipo di utilizzi e chiamato **plist** (abbreviazione di Property List).
Un esempio di documento contenente come unico punto di interesse la Torre di Pisa è riportato di seguito.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.
     apple.com/DTDs/PropertyList-1.0.dtd">
3 <plist version="1.0">
4    <array>
5       <dict>
6          <key>name</key>
7          <string>Torre di Pisa</string>
8          <key>lat</key>
9          <real>43.72306</real>
```

```
10          <key>long</key>
11          <real>10.39639</real>
12          <key>alt</key>
13          <integer>18</integer>
14          <key>image</key>
15          <string>leaningtower.png</string>
16          <key>description</key>
17          <string>La Torre di Pisa</string>
18          <key>url</key>
19          <string>http://www.opapisa.it/</string>
20       </dict>
21    </array>
22 </plist>
```

Il file XML contenente i Punti di interesse può essere incluso nel progetto o scaricato da Internet.
Quando l'applicazione ha bisogno dei dati è sufficiente, utilizzando un parser XML, creare oggetti a partire dalle proprietà contenute nel file plist.

4.3 Realizzazione della prima versione

4.3.1 Tracking

Per la fase di tracking si è proceduto a scrivere il codice necessario a gestire i vari sensori: GPS (§ 4.3.1.1), bussola (§ 4.3.1.2) e accelerometro (§ 4.3.1.3). Si noti che in tutti i casi in cui è stato necessario si sono inseriti controlli che verifichino la presenza o meno dell'hardware necessario.

4.3.1.1 GPS e localizzazione

Per poter utilizzare il GPS si è aggiunto il framework Core Location al progetto e si è importato l'header file necessario. Inoltre si è indicata la classe che si sta scrivendo conforme al protocollo del location manager:

```
#import <CoreLocation/CoreLocation.h>
@interface FirstViewController : UIViewController <
    CLLocationManagerDelegate>
```

Poi si è creato un metodo che avvia il location manager impostando le condizioni relative alla frequenza, cioè si è chiesto di essere avvertiti solo di variazioni della misurazione superiori a una costante DISTANCE_FILTER espressa in metri, e alla precisione degli aggiornamenti, nel codice seguente si è scelto una precisione intorno al chilometro. Si noti che si sono verificate due condizioni fondamentali per la riuscita della misurazione: il fatto che il sensore sia abilitato e che il dispositivo sia dotato di magnetometro.

```
- (void)startLocationUpdates {
  CLLocationManager *locationManager = [[CLLocationManager alloc]
      init];
  locationManager.delegate = self;
  locationManager.desiredAccuracy = kCLLocationAccuracyKilometer;
  locationManager.distanceFilter = DISTANCE_FILTER;
    if ([CLLocationManager locationServicesEnabled] && [
        CLLocationManager headingAvailable]) {
         [locationManager startUpdatingLocation];
    }
    else {
    NSLog(@"Error starting location updates");
    }
}
```

Il protocollo CLLocationManagerDelegate invia informazioni di aggiornamento sui dati rilevati attraverso il metodo delegato locationManager:didUpdateToLocation: fromLocation:. Se invece viene rilevato un errore viene chiamato il metodo delegato locationManager:didFailWithError:.

Inoltre è importante verificare la data della rilevazione: nella documentazione della libreria si raccomanda di controllare sempre il timestamp dell'informazione relativa prima di utilizzarla perché potrebbero essersi verificati ritardi nella chiamata del

metodo. L'oggetto NSTimeInterval che specifica un intervallo di tempo è espresso in secondi.

Di seguito un'implementazione del metodo delegato in cui si è stampato latitudine e longitudine nella console del debugger.

```
1 - (void)locationManager:(CLLocationManager *)manager
    didUpdateToLocation:(CLLocation *)newLocation fromLocation:(
    CLLocation *)oldLocation {
2   NSDate *eventDate = newLocation.timestamp;
3   NSTimeInterval howRecent = [eventDate timeIntervalSinceNow];
4   if (abs(howRecent) < TIME_INTERVAL) {
5     NSLog(@"latitude %+.6f, longitude %+.6f\n", newLocation.
          coordinate.latitude, newLocation.coordinate.longitude);
6   }
7   else {
8     NSLog(@"Update was old");
9   }
10 }
```

Il risultato stampato nella console sarà:

```
1 latitude +41.926676, longitude +12.512543
```

4.3.1.2 Magnetometro o bussola

Per utilizzare il Magnetometro si utilizza il framework CoreLocation come nella localizzazione (§ 4.3.1.1). Si è però impostato la proprietà headingFilter che regola la soglia richiesta per ricevere una chiamata sul metodo delegato: nel codice che segue si riceve un aggiornamento solo se il cambiamento rispetto al rilevamento precedente è maggiore di una costante HEADING_FILTER espressa in gradi.

```
1 - (void)startLocationUpdatesWithHeading {
2   CLLocationManager *locationManager = [[CLLocationManager alloc]
        init];
3   locationManager.delegate = self;
4   locationManager.headingFilter = HEADING_FILTER;
```

```
5   if ([CLLocationManager locationServicesEnabled] && [
        CLLocationManager headingAvailable]) {
6     [locationManager startUpdatingHeading];
7     [locationManager startUpdatingLocation];
8   }
9   else {
10    NSLog(@"Error starting location updates");
11  }
12 }
```

Bisogna inoltre introdurre una correzione nel caso in cui l'applicazione possa prevedere altri orientamenti del dispositivo perchè il nord rilevato è sempre da riferirsi alla parte alta del device. Il codice riportato corregge la rilevazione nel caso in cui cambi l'orientamento dell'applicazione:

```
1 - (float)heading:(float)heading fromOrientation:(UIDeviceOrientation
    )orientation {
2   float correctedHeading = heading;
3   switch (orientation) {
4     case UIDeviceOrientationPortrait :
5       break;
6     case UIDeviceOrientationPortraitUpsideDown :
7       correctedHeading = heading + PI_GRECO;
8       break;
9     case UIDeviceOrientationLandscapeLeft :
10      correctedHeading = heading + PI_GRECO/2;
11      break;
12    case UIDeviceOrientationLandscapeRight :
13      correctedHeading = heading - PI_GRECO/2;
14      break;
15    default :
16      break;
17  }
18  return correctedHeading % PI_GRECO*2;
19 }
```

Il metodo delegato che riceve gli aggiornamenti della bussola sarà implementato come descritto nel prossimo blocco di codice. Prima di utilizzare i dati si verifica la precisione della rilevazione: headingAccuracy restituirà un valore negativo nel caso in cui i dati non siano disponibili. Infine si stampa il valore del nord magnetico (magneticHeading) e quello del nord geografico (trueHeading) nella console.

```
1 - (void)locationManager:(CLLocationManager *)manager
    didUpdateHeading:(CLHeading *)newHeading {
2   if (newHeading.headingAccuracy > 0) {
3     UIDevice *device = [UIDevice currentDevice];
4     float magneticHeading = [self heading:newHeading.magneticHeading
          fromOrientation:device.orientation];
5     float trueHeading = [self heading:newHeading.trueHeading
         fromOrientation:device.orientation];
6     NSLog(@"magneticHeading: %f - trueHeading: %f", magneticHeading,
          trueHeading);
7   }
8 }
```

Il risultato stampato nella console sarà:

```
1 magneticHeading: 141.705276 - trueHeading: 144.180298
```

Bisogna infine tenere conto del fatto che potrebbe essere necessaria una calibrazione del sensore per rilevare la differenza tra le due misurazioni. La rilevazione del sensore viene fatta sempre sul nord magnetico e poi viene calcolato lo spostamento. È previsto un metodo delegato locationManagerShouldDisplayHeadingCalibration che permette di impostare se chiedere all'utente di muovere il device in una figura ad 8 fino a che non riesca a rilevare quale sia il valore dello spostamento.

4.3.1.3 Accelerometro

Dopo aver reso la classe conforme al protocollo UIAccelerometerDelegate per prima cosa è necessario scegliere i tempi di aggiornamento impostando la proprietà updateInterval di UIAccelerometer in secondi.

```
1 - (void)startAccelerometerUpdates {
2   UIAccelerometer *accelerometer = [UIAccelerometer
       sharedAccelerometer];
3   accelerometer.updateInterval = ACCELEROMETER_UPDATE_INTERVAL;
4   accelerometer.delegate = self;
5 }
```

Il metodo delegato chiamato dal protocollo ad ogni aggiornamento è accelerometer:didAccelerate:. L'oggetto di tipo UIAcceleration restituito ha tre proprietà x, y e z che contengono il valore dell'accelerazione sui tre assi. È stato necessario inoltre filtrare i dati dei sensori di un fattore kFilteringFactor ignorando così i minimi movimenti. Per farlo ho salvato i dati della rilevazione in un array float accel[3], per averlo disponibile al rilevamento successivo. Si sono stampato i dati rilevati come di consueto nella console:

```
1 - (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate
     :(UIAcceleration *)acceleration {
2   accel[0] = acceleration.x * kFilteringFactor + accel[0] * (1.0 -
       kFilteringFactor);
3   accel[1] = acceleration.y * kFilteringFactor + accel[1] * (1.0 -
       kFilteringFactor);
4   accel[2] = acceleration.z * kFilteringFactor + accel[2] * (1.0 -
       kFilteringFactor);
5   NSLog(@"%@%f %@%f %@%f", @"X: ", accel[0], @"Y: ", accel[1], @"Z:
       ", accel[2]);
6 }
```

Il risultato sarà il seguente:

```
1 X: 0.695961 Y: 0.182876 Z: -0.468010
```

4.3.2 Registration

La fase di registration è, lo ricordiamo, quella in cui si allineano le informazioni reali con quelle virtuali grazie, nel caso del Sensor Monitoring Tacking (§ 2.1.1), ai dati di

localizzazione provenienti dai sensori e quelli associati agli oggetti da aumentare raccolti durante la fase di tracking.
La matematica di base necessaria per una semplice Realtà Aumentata è mostrata in figura 4.1: il punto (x_0, y_0) rappresenta la posizione dell'utilizzatore del dispositivo (latitudine e longitudine), mentre il punto (x_1, y_1) rappresenta la posizione del POI che si vuole aumentare. Considerando la distanza tra l'utente e il punto d'interesse

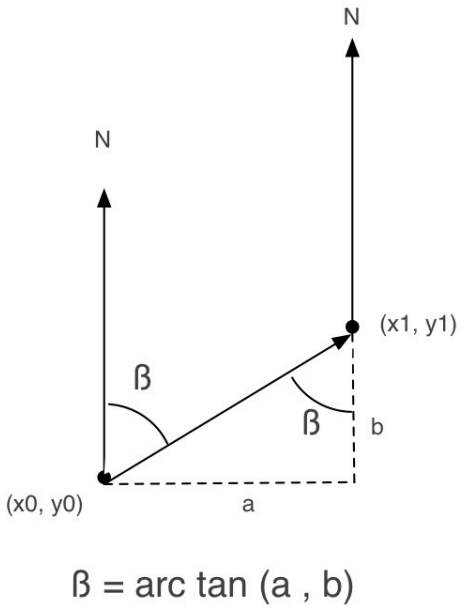


Figura 4.1: Matematica di base per la fase di Registration

Immagine da [Par11]

molto minore della distanza tra il dispositivo e il polo nord, si può approssimare che le due linee che collegano l'utente e il POI con il polo nord (per la precisione il nord magnetico) siano parallele. Sappiamo inoltre grazie alla geometria che i due angoli β mostrati in figura sono congruenti. Per questo l'angolo β, che rappresenta, nella prospettiva dell'utente, l'angolo tra l'oggetto osservato e il nord, è facile da calcolare conoscendo le proprie coordinate e quelle del POI, con la formula trigonometrica:

$$\beta = arctan(a, b) \tag{4.1}$$

Ma β è anche l'angolo restituito dalla bussola quando si è rivolti verso il POI e questo si trova al centro dello schermo, quindi se ipotizziamo di utilizzare lo smartphone in modalità orizzontale, rivolto in direzione del POI le coordinate dell'oggetto visualizzato sullo schermo saranno:

$$x = \frac{w}{2} \tag{4.2}$$

$$y = \frac{h}{2} \tag{4.3}$$

dove w e h sono la larghezza e l'altezza in pixel dello schermo del dispositivo.
La posizione del POI dovrà cambiare quando muoveremo il dispositivo verso destra o verso sinistra, ma le sue coordinate sullo schermo rimangono facilmente calcolabili considerando che sono proporzionali alla differenza tra β e l'angolo restituito dalla bussola. La posizione orizzontale sarà:

$$x = \frac{w}{2} + \delta * \frac{w}{\rho/2} \tag{4.4}$$

dove δ è la differenza tra il β calcolato e l'orientamento ottenuto dalla bussola muovendo il dispositivo, ρ è l'angolo di visuale orizzontale della fotocamera (circa 60 gradi nel nostro caso).
Dunque è stato sufficiente, nel metodo delegato che aggiorna su cambiamenti della bussola (locationManager:didUpdateHeading:), inserire il seguente codice per far apparire un immagine (imageView) in direzione del Punto d'Interesse.

```
1  float poiLon = POI_LONGITUDE;
2  float poiLat = POI_LATITUDE;
3  float a = poiLon - userLon;
4  float b = poiLat - userLat;
5  float alpha = PI_GRECO * atan2(a, b) / M_PI;
6  if (alpha < 0.0)
7      alpha += PI_GRECO*2;
8  else if (alpha > PI_GRECO*2)
9      alpha -= PI_GRECO*2;
10 CLLocationDirection theHeading = newHeading.trueHeading;
```

```
11 float deltaOrient = alpha - theHeading;
12 CGFloat X = SCREEN_WIDTH/2 + (deltaOrient * SCREEN_WIDTH /
      HORIZONTAL_VIEWING_ANGLE);
13 CGFloat Y = SCREEN_HEIGHT/2;
14 [self.imageView setCenter:CGPointMake(X, Y)];
```

4.3.3 Display

Per quanto riguarda la fase di Display, quella in cui si ''appoggiano'' oggetti sulla ripresa della telecamera si è proceduto passo passo fino a ottenere il risultato cercato.

Per prima cosa si è utilizzata la ripresa della fotocamera e ci si è appoggiata sopra una semplice etichetta con un stringa. Il primo metodo riportato nel seguente blocco di codice viene lanciato durante la fase viewDidLoad, il secondo durante la viewDidAppear:. Si noti che 1:1.12412 è il rapporto tra la larghezza e la lunghezza dello schermo del dispositivo.

```
1 - (void)createImagePicker {
2   picker = [[UIImagePickerController alloc] init];
3   picker.sourceType = UIImagePickerControllerSourceTypeCamera;
4   picker.cameraViewTransform = CGAffineTransformScale(picker.
       cameraViewTransform, 1.0f, 1.12412f);
5 }
6 - (void)startImagePicker {
7   [self presentModalViewController:picker animated:NO];
8   [label setText:@"Hello augmented world!"];
9   picker.cameraOverlayView = label;
10 }
```

In seguito si è proceduto ponendo una riga al centro dello schermo che indicasse l'orizzonte terreste cioè che mostrasse la rotazione del dispositivo verticalmente e orizzontalmente rispetto ad esso mantenendo sullo schermo una linea che lo indicasse. Il risultato è stato ottenuto monitorando la rotazione del dispositivo sull'asse

z e sull'asse x dell'accelerometro, secondo le seguenti formule:

$$\alpha = -atan2(acceleration.y, acceleration.x) - \frac{\pi}{2} \tag{4.5}$$

$$\phi = -atan2(acceleration.y, acceleration.z) - \frac{\pi}{2} \tag{4.6}$$

$$y = \frac{h}{2} - (\lambda * 10) * sin(\phi) \tag{4.7}$$

Dove α è l'angolo di rotazione dell'orizzonte artificiale rispetto al piano terrestre, ϕ è l'angolo verticale e y è la posizione del centro della barra calcolato in base all'altezza dello schermo, all'angolo di visuale verticale della fotocamera λ (circa 53 gradi nel caso in oggetto) e a ϕ. Si noti che nella formula 4.6 la misura dell'angolo è espressa in radianti, nella 4.7 in gradi, quindi è necessaria una conversione. $atan2$ è una variazione della funzione trigonometrica arcotangente che prende in ingresso due parametri: presi gli argomenti reali x e y non nulli, $atan2(y, x)$ indica l'angolo in radianti tra l'asse positivo delle X in un piano e un punto di coordinate giacente su di esso.

Si è dunque modificato nel modo seguente i metodi startImagePicker: e accelerometer:didAccelerate:

```
1  - (void) startImagePicker {
2      [self presentModalViewController:picker animated:NO];
3      CGRect viewFrame = self.view.frame;
4      viewFrame.origin.x = viewFrame.origin.y = 0.0;
5      UIView *overlayView = [[UIView alloc] initWithFrame:viewFrame];
6      UIImage *overlayGraphic = [UIImage imageNamed:@"
           artificialHorizon.png"];
7      overlayGraphicView = [[UIImageView alloc] initWithImage:
           overlayGraphic];
8      overlayGraphicView.frame = CGRectMake(30, 100, 260, 200);
9      [overlayView addSubview:overlayGraphicView];
10     picker.cameraOverlayView = overlayView;
11     [overlayView release];
12     [[UIAccelerometer sharedAccelerometer] setDelegate:self];
13 }
```

```
14 - (void)accelerometer:(UIAccelerometer *)accelerometer didAccelerate
     :(UIAcceleration *)acceleration {
15   double angle = -atan2(acceleration.y, acceleration.x) - M_PI/2;
16   double angleInDeg = angle * PI_GRECO / M_PI;
17   double vertAngle = -atan2(acceleration.y, acceleration.z) - M_PI
        /2;
18   double vertAngleInDeg = vertAngle * PI_GRECO / M_PI;
19   [label setText:[NSString stringWithFormat:@"H-Angle: %5.1f V-Angle
        : %5.1f", angleInDeg, vertAngleInDeg]];
20   CGPoint overlayCenter = [overlayGraphicView center];
21   overlayCenter.y = SCREEN_HEIGHT - VERTICAL_VIEWING_ANGLE * sin(
        vertAngle);
22   [overlayGraphicView setCenter:overlayCenter];
23   overlayGraphicView.transform = CGAffineTransformMakeRotation(angle
        );
24 }
```

Il risultato è mostrato in figura 4.3.

4.4 Realizzazione della seconda versione

Si è deciso di migliorare le fasi di Tracking e Registration utilizzando le novità di CoreMotion disponibili a partire dalla versione 5 di iOS. L'idea sulla nuova fase di Tracking e sulla quella di Registration è stata suggerita dal codice di esempio **pARk** [Appg] disponibile all'interno della documentazione Apple per sviluppatori iOS nella sezione relativa alle API di CoreMotion. Inoltre si è cambiata la fase di display passando da UIImagePicker a AVCaptureSession di AV Foundation introdotto con iOS 4.

4.4.1 Tracking con Core Motion

Pur mantenendo quando descritto in precedenza per la localizzazione (§ 4.3.1.1), non si sono utilizzati i dati dell'accelerometro e della bussola per tracciare i movimenti ma la matrice di rotazione associata al dispositivo. Questo ha permesso di

evitare le approssimazioni del modello presenti nel primo prototipo.
Nel seguente codice si avvia CMMotionManager e si imposta la possibilità di richiedere una calibrazione, un tempo di aggiornamento pari alla costante MOTION_UPDATE_INTERVAL espressa in secondi e il nord geografico invece di quello magnetico come misura di riferimento.

```
1 - (void)startDeviceMotion {
2   motionManager = [[CMMotionManager alloc] init];
3   motionManager.showsDeviceMovementDisplay = YES;
4   motionManager.deviceMotionUpdateInterval = MOTION_UPDATE_INTERVAL;
5   [motionManager startDeviceMotionUpdatesUsingReferenceFrame:
        CMAttitudeReferenceFrameXTrueNorthZVertical];
6 }
```

Le misurazioni verranno continuamente aggiornate da CoreMotion e andremo a leggerne i dati ad ogni refresh dello schermo, come spiegato nella descrizione della nuova fase di display (§ 4.4.3).

4.4.2 Registration utilizzando matrici di proiezione

In questa sezione è descritto come sfruttare la matrice di rotazione per proiettare elementi sullo schermo e utilizza diverse funzioni di utilità matematiche per calcoli sulle matrici. Una di queste è quella che durante la fase di inizializzazione crea una matrice di proiezione dati l'angolo verticale del campo visivo, la proporzione tra le l'altezza e la larghezza dello schermo e l'area di due piani perpendicolari.

```
1 createProjectionMatrix(projectionTransform, VERTICAL_VIEWING_ANGLE*
    DEGREES_TO_RADIANS, SCREEN_WIDTH*1.0f / SCREEN_HEIGHT,
    NEAR_PLAN_AREA, FAR_PLAN_AREA);
```

Ad ogni cambiamento dei dati dei sensori viene chiamata una funzione che realizza effettivamante la fase di Registration, cioè allinea i POI nelle posizioni relative alla posizione dell'utente. Per prima cosa vengono preparati i punti per essere utilizzati: latitudine e longitudine vengono trasformate in coordinate ECEF e ENU (vedi appendice C), i POI vengono ordinati in ordine di distanza, di modo che stampandoli

in ordine quelli più vicini siano in primo piano, quindi vengono disposti nello spazio nella loro posizione relativa.

```
- (void)updatePlacesOfInterestCoordinates {
  if (placesOfInterestCoordinates != NULL) {
    free(placesOfInterestCoordinates);
  }
  placesOfInterestCoordinates = (vec4f_t *)malloc(sizeof(vec4f_t)*
      placesOfInterest.count);
  int i = 0;
  double myX, myY, myZ;
  latLonToEcef(USER_COORDINATE_LATITUDE, USER_COORDINATE_LONGITUDE,
      0.0, &myX, &myY, &myZ);
  typedef struct {
    float distance;
    int index;
  } DistanceAndIndex;
  NSMutableArray *orderedDistances = [NSMutableArray
      arrayWithCapacity:placesOfInterest.count];
  for (Place *poi in [[self placesOfInterest] objectEnumerator]) {
    double poiX, poiY, poiZ, e, n, u;
    latLonToEcef(POI_LATITUDE, POI_LONGITUDE, 0.0, &poiX, &poiY, &
        poiZ);
    ecefToEnu(USER_COORDINATE_LATITUDE, USER_COORDINATE_LONGITUDE,
        myX, myY, myZ, poiX, poiY, poiZ, &e, &n, &u);
    placesOfInterestCoordinates[i][0] = (float)n;
    placesOfInterestCoordinates[i][1]= -(float)e;
    placesOfInterestCoordinates[i][2] = 0.0f;
    placesOfInterestCoordinates[i][3] = 1.0f;
    DistanceAndIndex distanceAndIndex;
    distanceAndIndex.distance = sqrtf(n*n + e*e);
    distanceAndIndex.index = i;
    [orderedDistances insertObject:[NSData dataWithBytes:&
        distanceAndIndex length:sizeof(distanceAndIndex)] atIndex:i
        ++];
  }
```

```
27   [orderedDistances sortUsingComparator:(NSComparator)^(NSData *a,
        NSData *b) {
28     const DistanceAndIndex *aData = (const DistanceAndIndex *)a.
          bytes;
29     const DistanceAndIndex *bData = (const DistanceAndIndex *)b.
          bytes;
30     if (aData->distance < bData->distance) {
31       return NSOrderedAscending;
32     } else if (aData->distance > bData->distance) {
33       return NSOrderedDescending;
34     } else {
35       return NSOrderedSame;
36     }
37   }];
38   for (NSData *d in [orderedDistances reverseObjectEnumerator]) {
39     const DistanceAndIndex *distanceAndIndex = (const
          DistanceAndIndex *)d.bytes;
40     Place *poi = (Place *)[placesOfInterest objectAtIndex:
          distanceAndIndex->index];
41     [self addSubview:poi.associatedView];
42   }
43 }
```

Il resto viene fatto da drawRect:, il metodo che viene chiamato ogni volta che si marca la vista per essere ridisegnata, dove inizialmente si crea la matrice di proiezione della fotocamera utilizzando la matrice di proiezione creata in precedenza e la matrice che descrive la posizione della fotocamera ottenuta con CoreMotion (vedi § 4.4.3). La moltiplicazione tra queste due matrici crea la matrice di proiezione della fotocamera che corrisponde ad un tronco di piramide.

```
1 mat4f_t projectionCameraTransform;
2 multiplyMatrixAndMatrix(projectionCameraTransform,
     projectionTransform, cameraTransform);
```

Come mostrato in figura 4.2, questo tronco di piramide definisce l'area in cui si devono trovare i POI per essere visualizzati.

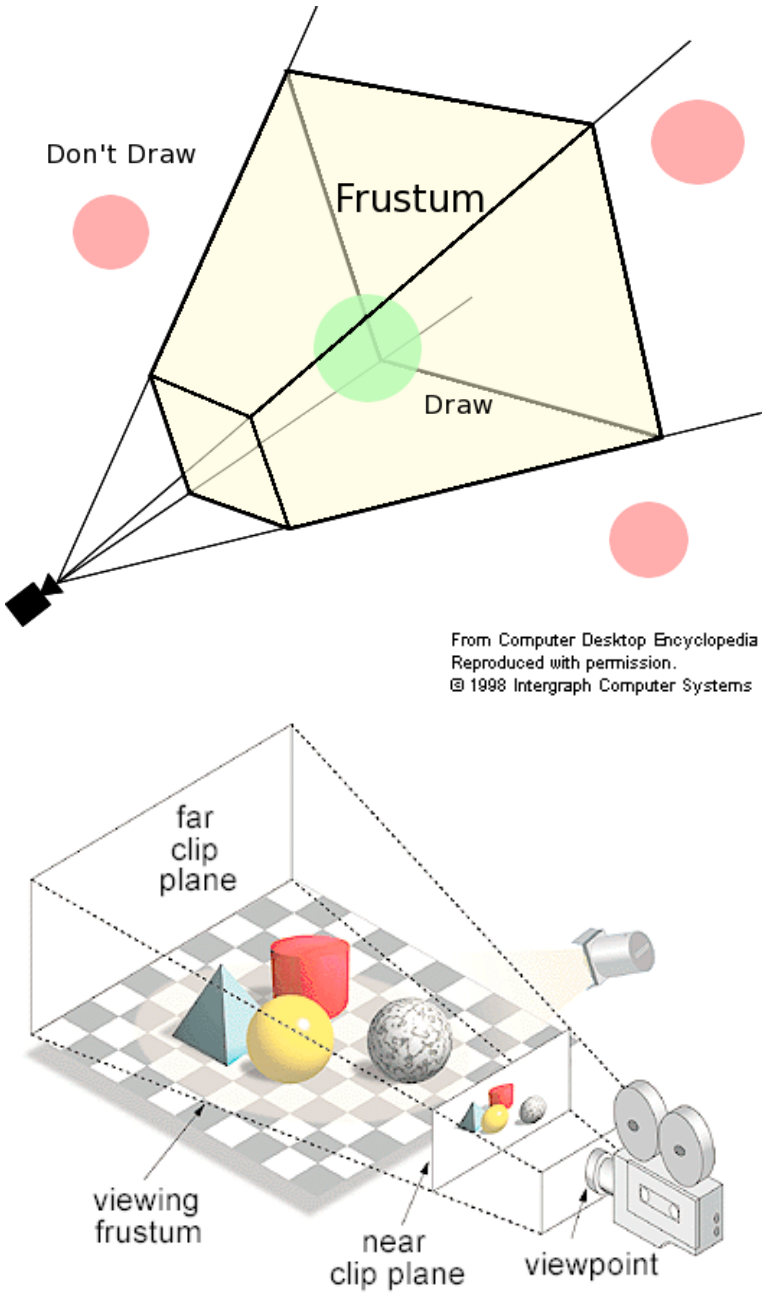


Figura 4.2: Il tronco di piramide descritto dalla matrice di proiezione.

Immagini da `http://blog.jaxgl.com/wp-content/uploads/2012/06/frustum-culling.png` e `http://img.tfd.com/cde/_FRUSTUM.GIF`.

Quindi per ogni punto di interesse la sua posizione è moltiplicata per la matrice di proiezione. Questo permette di proiettare appunto il punto di interesse nel area visibile alla fotocamera, applicandogli rotazione e prospettiva. Inoltre vengono fatti alcuni calcoli che permettono di riportare in due dimensioni e nel sistema di riferimento della grafica, le posizioni degli oggetti eliminando quelli troppo vicini o troppo lontani.

```
1 for (PlaceOfInterest *poi in [placesOfInterest objectEnumerator]) {
2         vec4f_t v;
3         multiplyMatrixAndVector(v, projectionCameraTransform,
              placesOfInterestCoordinates[i]);
4         float x = (v[0] / v[3] + 1.0f) * 0.5f;
5         float y = (v[1] / v[3] + 1.0f) * 0.5f;
6         if (v[2] < 0.0f) {
7             poi.view.center = CGPointMake(x*self.bounds.size.width,
                  self.bounds.size.height-y*self.bounds.size.height);
8             poi.view.hidden = NO;
9         } else {
10            poi.view.hidden = YES;
11        }
12        i++;
13    }
```

4.4.3 Miglioramento della fase di display

Anche la fase di display è stata migliorando da UIImagePicker di iOS 3 a AVCaptureSession di iOS 4. Il primo passo è creare un'istanza di AVCapureSession e visualizzarlo in un AVCaptureVideoPreviewLayer per poterci poi disegnare sopra oggetti in overlay. Inoltre si è creata un'istanza di AVCaptureDevice specificando la fotocamera posteriore del dispositivo:

```
1 AVCaptureSession* session = [[AVCaptureSession alloc] init];
2 AVCaptureVideoPreviewLayer *previewLayer = [
     AVCaptureVideoPreviewLayer layerWithSession:session];
```

```
previewLayer.frame = previewView.frame;
[previewView.layer addSublayer:previewLayer];
AVCaptureDevice* camera = [AVCaptureDevice
    defaultDeviceWithMediaType:AVMediaTypeVideo];
NSError *error=nil;
AVCaptureInput* cameraInput = [[AVCaptureDeviceInput alloc]
    initWithDevice:camera error:&error];
AVCaptureVideoDataOutput* videoOutput = [[AVCaptureVideoDataOutput
    alloc] init];
[session setSessionPreset:AVCaptureSessionPresetMedium];
[session addInput:cameraInput];
[session addOutput:videoOutput];
[session startRunning];
```

Infine, come accennato in precedenza, con displayLinkWithTarget:selector: si ottiene che il metodo di lettura dei sensori venga chiamato ad ogni refresh dello schermo e che questo marchi la vista per essere ridisegnata, cioè cancelli tutti gli oggetti presenti e ridisegni i punti di interesse nella eventuale nuova posizione relativa.

```
- (void)startDisplayLink {
  displayLink = [[CADisplayLink displayLinkWithTarget:self selector:
      @selector(onDisplayLink:)] retain];
  [displayLink setFrameInterval:1];
  [displayLink addToRunLoop:[NSRunLoop currentRunLoop] forMode:
      NSDefaultRunLoopMode];
}
- (void)onDisplayLink:(id)sender {
  CMDeviceMotion *d = motionManager.deviceMotion;
  if (d != nil) {
    CMRotationMatrix r = d.attitude.rotationMatrix;
    transformFromCMRotationMatrix(cameraTransform, &r);
    [self setNeedsDisplay];
  }
}
```

Il fatto che le misurazioni vengano lette ad ogni aggiornamento dello schermo invece che ad ogni cambio di informazioni del sensore stesso, rende più fluida la visualizzazione e lo spostamento degli elementi di computer graphics. Il risultato è mostrato in figura 4.4.

Figura 4.3: Screenshot dal primo prototipo

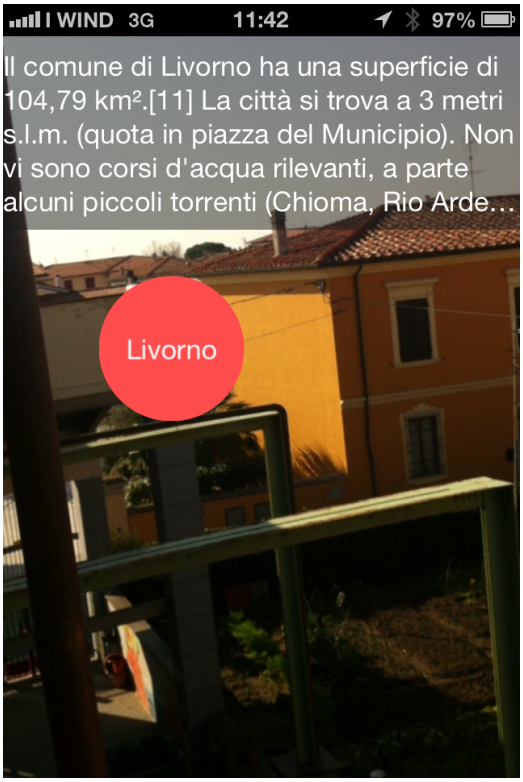


Figura 4.4: Screenshot dal secondo prototipo

Capitolo 5

Conclusioni e sviluppi futuri

5.1 Obiettivi Raggiunti

Anche se per motivazioni non collegate al tirocinio e all'azienda ospitante, il tempo necessario a terminare il tirocinio è stato più lungo del previsto.
Tema del tirocinio era Realtà aumentata su iPhone ma il progetto realizzato è più ampio in quanto ha previsto la realizzazione di una piattaforma (e non quindi di una singola applicazione) ed inoltre è stata realizzata per tutti i dispositivi iOS (non solo iPhone). Il titolo è dunque diventato: Sviluppo di una piattaforma di Realtà Aumentata per iOS.
Sono dunque stati raggiunti e superati gli obiettivi previsti dal progetto di tirocinio: è stata creata una piattaforma espandibile e riutilizzabile per applicazioni di Realtà Aumentata su dispositivi iOS. I test sui dispositivi reali hanno dimostrato un'ottima reattività e buone prestazioni.

5.2 Competenze teoriche e pratiche acquisite

Al momento dell'assegnamento del tirocinio, pur da studente lavoratore, non avevo mai lavorato con le tecnologie che ho incontrato in questo tirocinio, anche se da

tempo mi incuriosivano. Durante lo svolgimento del progetto ho imparato ad utilizzare proficuamente strumenti e tecnologie che mi erano nuovi, raggiungendone una ottima padronanza.
L'ingresso in 3logic attraverso questa esperienza è stato un trampolino che mi ha portato a entrare come dipendente della stessa prima del termine del progetto. L'ambiente stimolante e il confronto con persone con più esperienza hanno stimolato l'analisi e la soluzione dei problemi incontrati pur nella totale autonomia decisionale. L'ambiente giovane e rilassato ha sicuramente favorito queste dinamiche.
Inoltre la possibilità di mettere in campo e miscelare molte delle conoscenze acquisite durante il percorso universitario mi ha permesso di vedere in una nuova luce l'importanza della parte teorica della formazione di un informatico. Per esempio aver studiato sintassi dei linguaggi di programmazione e semantica operazionale ha reso veloce ed indolore imparare un linguaggio di programmazione nuovo, pur con le sue peculiarità. Il progetto è stato particolarmente interessante perché ha permesso di toccare vari aspetti dello sviluppo di una applicazione su iOS, dall'elaborazione grafica alla gestione di dati geolocalizzati, passando per l'utilizzo di documenti in xml, e molto altro.

5.3 Pubblicazione e futuro del codice e dell'applicazione

Durante lo sviluppo sono state ipotizzate numerose migliorie come un radar che mostri la presenza di punti di interesse a 360 gradi evidenziando al contempo il campo visivo del device.
Il lavoro svolto durante il tirocinio è stato inserito in un progetto che molto probabilmente sfocerà in un prodotto commerciale. Con i dirigenti dell'azienda inoltre si è ipotizzato l'inserimento del modulo software creato all'interno di altri progetti, sia preesistenti che nuovi.

Appendici

Appendice A

Esempi di utilizzo della realtà aumentata

La Realtà aumentata ha molte applicazioni e molte aree posso beneficiare dell'apporto di questa tecnologia, non solo le già citate aree militari o di intrattenimento. Ecco qualche esempio:

- **Architettura:** il principale utilizzo in questo campo è quello di visualizzare una costruzione che non esiste ancora nella posizione che assumerà una volta costruita, vedi per esempio [IPBB07].
- **Arte:** sono molti gli artisti che negli ultimi anni hanno realizzato opere d'arte che utilizzano la Realtà Aumentata. Friedbert Schulze (`http://www.friedbertschulze.de`) per esempio ha creato una mostra in cui le opere erano visibili solo attraverso una particolare applicazione di Realtà Aumentata.
- **Commercio:** la Realtà Aumentata viene spesso usata, è il caso dell'ultimo catalogo IKEA mostrato in figura 1, nei cataloghi per mostrare un modello 3D del prodotto che si vuole proporre al cliente, magari permettendogli di personalizzare alcuni parametri come il colore o le varianti. In altri casi viene usata per permettere all'acquirente di vedere all'interno di una confezione senza aprirla.

- **Educazione:** la Realtà Aumentata può essere usata per arricchire l'esperienza didattica. Per esempio l'obiettivo dell'azienda ARDL (`http://augmentedrealitydevel com`)è quello di creare ambienti virtuali in cui gli studenti possano interagire e visualizzare in maniera tridimensionale ciò che studiano sui libri di scuola. Una delle applicazioni più apprezzate è quella in cui gli studenti possono sezionare virtualmente il corpo umano ed i suoi organi, scoprendone il funzionamento.

- **Design:** La AR può aiutare a definire l'esperienza e il design di un oggetto prima di completarlo oppure per confrontare un disegno preparatorio con il prototipo realizzato da esso per verificare eventuali inconsistenze [PSPS11].

- **Medicina:** può mostrare sugli occhiali del chirurgo i parametri vitali dell'operato senza che debba ritrarre lo sguardo, oppure permettere una sorta di raggi x virtuali. Un interessante utilizzo è mostrato in [NPSA05].

- **Automotive:** Le informazioni dei navigatori che utilizzano la Realtà Aumentata cenno proiettate sullo schermo del veicolo insieme alle informazioni sull'auto, il meteo o il traffico, permettendo all'utente di avere a disposizione tutte le informazioni necessarie senza distogliere lo sguardo dalla strada.

- **Turismo:** le informazioni sul monumento inquadrato con lo smartphone possono sostituire la guida, inoltre è possibili mostrare all'utente immagini o ricostruzioni storiche di ciò che sta vedendo. Oppure possono suggerire servizi e itinerari di pubblica utilità come farmacie, uffici o altro.

- **Traduzioni:** Esistono applicazioni di Realtà Aumentata che traducono in tempo reale le scritte da una lingua ad un'altra.

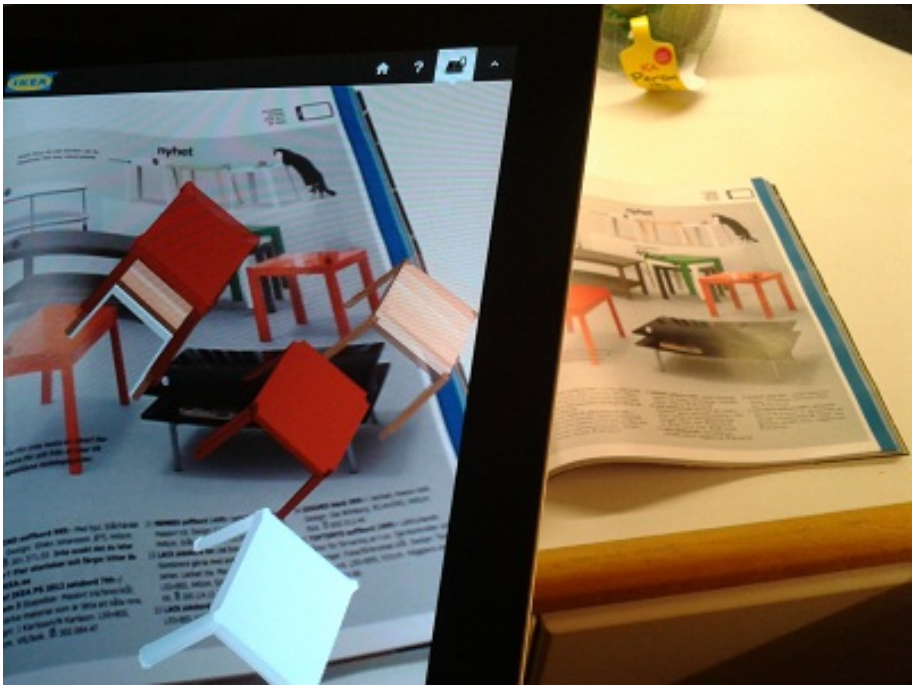


Figura 1: Realtà Aumentata nel catalogo IKEA 2013.

Immagine da
`beantin.se/post/30095128278/ikea-catalogue-augmented-reality`

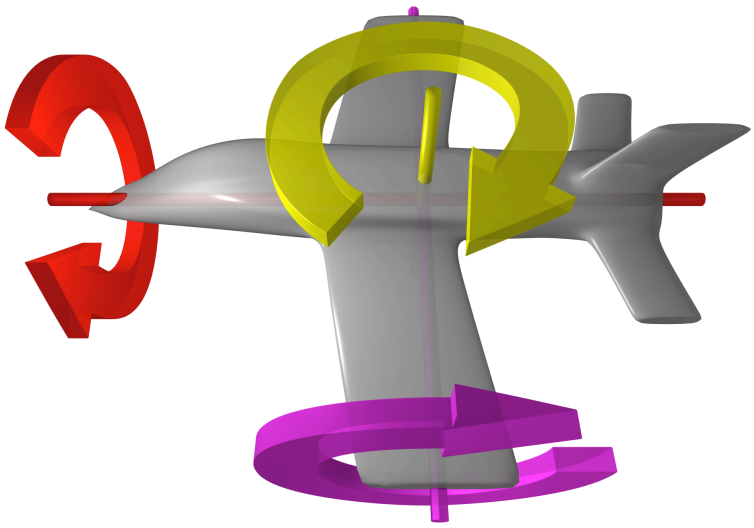


Figura 2: Rotazioni attorno agli assi corpo rigido.

Nomi delle rotazioni: beccheggio (viola), imbardata (giallo), rollio (rosso)
Immagine da [Wikd]

Appendice B

Come misurare l'orientamento nello spazio: le rotazioni rpy

Un movimento nello spazio tridimensionale si definisce a sei gradi di libertà (in inglese 6DOF, Six Degrees Of Freedom) e si riferisce all'abilità di muoversi liberamente avanti/indietro, sù/giù, sinistra/destra, misurando cioè la rotazione lungo tre assi perpendicolari. Poiché il movimento lungo ognuno di questi assi è indipendente sia per gli assi di traslazione che per gli assi di rotazione, il movimento ha quindi sei gradi di libertà. Per descrivere queste rotazioni si usano le rotazioni rpy mostrate in figura 3 che descrivono l'insieme delle rotazioni di un corpo rigido rispetto agli assi del sistema di riferimento cartesiano (le misure sono in radianti):

- una rotazione intorno all'asse y che attraversa il device da un lato all'altro è detta **beccheggio** (in inglese **pitch**);
- una rotazione intorno all'asse x che attraversa il device dall'alto al basso è detta **rollio** (in inglese **roll**);
- una rotazione intorno all'asse verticale z passante per il baricentro del mezzo è detta **imbardata** (in inglese **yaw**).

Appendice C

Coordinate ECEF e ENU

ECEF, acronimo di Earth-Centered Earth-Fixed, è un sistema di coordinate cartesiane geocentrico. Il punto (0,0,0) denota il centro della terra (da cui il nome Earth-Centered) ed il sistema ruota in maniera solidale con la Terra. Il piano X-Y è coincidente con il piano equatoriale; l'asse Z ortogonale a questo piano punta nella direzione del Polo Nord. Le coordinate sono rappresentate in metri. Le coordinate ECEF

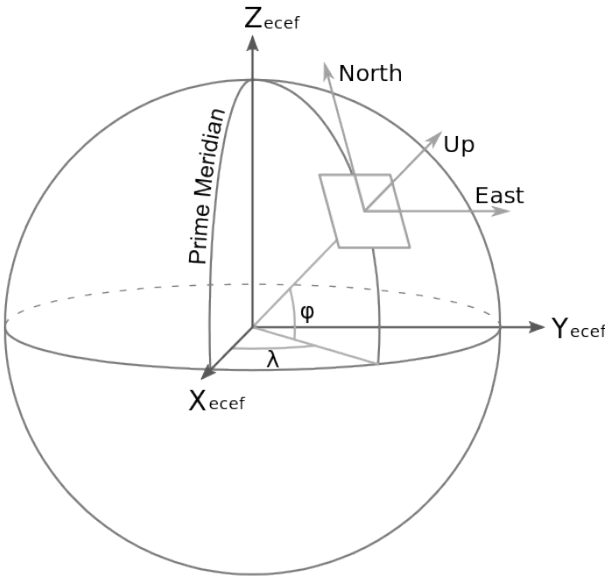


Figura 3: Coordinate ECEF e ENU.

Immagine da `http://en.wikipedia.org/wiki/Geodetic_datum`

sono utilizzate nel sistema di posizionamento GPS, in quanto considerate sistema di riferimento terrestre convenzionale.

Quello delle coordinate ENU(Local east, north, up) invece è un sistema di coordinate cartesiano considerato più intuitivo e più pratico di quello ECEF e Geodetico. Le coordinate locali ENU sono indicate dalla coordinate su un piano tangente la terra in un punto specifico. I tre assi per convenzione sono appunto est, nord e su.

Bibliografia

[A+97] R.T. Azuma et al. A survey of augmented reality. *Presence-Teleoperators and Virtual Environments*, 6(4):355--385, 1997.

[Appa] Apple. About the AV Foundation Framework. `http://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/AVFoundationPG`.

[Appb] Apple. AVFoundation Media Capture. `https://developer.apple.com/library/ios/#documentation/AudioVideo/Conceptual/AVFoundationPG/Articles/04_MediaCapture.html`.

[Appc] Apple. Core Location Framework Reference. `http://developer.apple.com/library/ios/#documentation/CoreLocation/Reference/CoreLocation_Framework/_index.html`.

[Appd] Apple. Event Handling Guide for iOS. `http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/MotionEvents/MotionEvents.html`.

[Appe] Apple. iOS Developer Library online. `http://developer.apple.com/library`.

[Appf] Apple. iOS Technology Overview. http://developer.apple.com/library/ios/documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/iPhoneOSTechOverview.pdf.

[Appg] Apple. pARk sample code. http://developer.apple.com/library/ios/#samplecode/pARk.

[App10] Apple. iphone human interface guidelines - user experience. http://developer.apple.com/library/ios/#documentation/userexperience/conceptual/mobilehig/, 2010.

[aug10] Has Augmented Reality Peaked? http://artimes.rouli.net/2010/04/has-augmented-reality-peaked.html, 2010.

[Bra10] Paolo Brandi. Sviluppo di un'applicazione in Realtà Aumentata per dispositivi mobili Android. Master's thesis, Università di Pisa, 2010.

[BRI05] Oliver Bimber, Ramesh Raskar, and Masahiko Inami. *Spatial augmented reality*. AK Peters, 2005.

[BS11] Jonathan Blocksom and Jonathan Saggau. iPhone Augmented Reality. Presented at the iPad Devcon, Boston, 2011.

[com] Augmented Reality SDKs - Social Compare. http://socialcompare.com/en/comparison/augmented-reality-sdks.

[DA11] Bill Dudney and Chris Adamson. *Sviluppare applicazioni con iPhone SDK*. Apogeo, 2011.

[Dop] Christian Doppler. History of Mobile Augmented Reality. https://www.icg.tugraz.at/~daniel/HistoryOfMobileAR/.

[Dut] Valerio Dutto. Il ciclo di vita delle applicazioni iOS. http://www.devapp.it/wordpress/l017-il-ciclo-di-vita-delle-applicazioni-ios.html.

[Eke09] Björn Ekengren. Mobile augmented reality. Master's thesis, Royal Institute of Technology, School of Computer Science and Communication, Stockholm, Sweden, 2009.

[Fox98] D. Fox. Composing magic lenses. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 519--525. ACM Press/Addison-Wesley Publishing Co., 1998.

[Gal] Giuliano Galea. iPhone OS è il ''parente povero'' di OS X? http://www.appuntidigitali.it/9204/iphone-os-e-il-parente-povero-di-os-x/.

[GZWS10] J.B. Gotow, K. Zienkiewicz, J. White, and D.C. Schmidt. Addressing challenges with augmented reality applications on smartphones. *Mobile Wireless Middleware, Operating Systems, and Applications*, pages 129--143, 2010.

[Hen07] A. Henrysson. *Bringing augmented reality to mobile phones*. PhD thesis, Linköping, 2007.

[IBM] IBM. The IBM 5-in-5, Innovations that will change our lives in the next five years. http://www.ibm.com/smarterplanet/us/en/ibm_predictions_for_future/ideas/.

[ios] Augmented reality on the iPhone with iOS4.0. http://cmgresearch.blogspot.it/2010/10/augmented-reality-on-iphone-with-ios40.html.

[IPBB07] J.L. Izkara, J. Pérez, X. Basogain, and D. Borro. Mobile augmented reality, an advanced tool for the construction sector. In *Proceedings of the 24th CIB W78 Conference*. Citeseer, 2007.

[iPh] An Overview of the iOS Architecture. `http://silicon-news.com/news/2012/06/15/ios-hardware-architecture/`.

[Lie09] G. Liestøl. Situated simulations: A prototyped augmented reality genre for learning on the iphone. *International Journal of Interactive Mobile Technologies (iJIM)*, 3:pp--24, 2009.

[Loo07] J.C.A. Looser. Ar magic lenses: Addressing the challenge of focus and context in augmented reality. 2007.

[Man] Steve Mann. Eye Am a Camera: Surveillance and Sousveillance in the Glassage. `http://techland.time.com/2012/11/02/eye-am-a-camera-surveillance-and-sousveillance-in-the-glassa`

[Mel09] Alessio Melani. Progetto e realizzazione di un sistema informativo georeferenziato per dispositivi mobili. Master's thesis, Università di Pisa, 2009.

[MK94] P. Milgram and F. Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77(12):1321--1329, 1994.

[MTUK95] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. Augmented reality: a class of displays on the reality-virtuality continuum. In *Proceedings of SPIE*, volume 2351, page 282, 1995.

[New10] Ben Newhouse. Developing Augmented Reality Applications On The Iphone. Presented at the Voices That Matter: iPhone Developers Conference 2010, Philadelphia, 2010.

[NPSA05] S. A. Nicolau, X. Pennec, L. Soler, and N. Ayache. A complete augmented reality guidance system for liver punctures: first clinical evaluation. In *Proceedings of the 8th international conference on Medical Image Computing and Computer-Assisted Intervention - Volume Part I*, MICCAI'05, pages 539--547, Berlin, Heidelberg, 2005. Springer-Verlag.

[Par11] Geppy Parziale. A simple Augmented Reality engine for iOS devices. `http://www.invasivecode.com/blog/archives/1435`, 2011.

[PK11] D. Prochazka and T. Koubek. Augmented reality implementation methods in mainstream applications. *arXiv preprint arXiv:1106.5569*, 2011.

[Pow11] Jeff Powers. Computer Vision and Augmented Reality on iOS: The rapidly evolving role of the camera. Presented at the Voices That Matter: iPhone Developers Conference 2011, Seattle, 2011.

[PP09] D. Pilone and T. Pilone. *Head First Iphone Development: A Learner's Guide to Creating Objective-C Applications for the Iphone*. O'Reilly Media, 2009.

[PSPS11] David Prochazka, Michael Stencl, Ondrej Popelka, and Jiri Stastny. Mobile augmented reality applications. *Proceedings of Mendel 2011: 17th International Conference on Soft Computing*, pages 469--476, 2011.

[Qua] Paolo Quadrani. Augmented reality. Presented at Advanced iPhone Programming Workshop 2010, Bologna.

[Roc11] Kyle Roche. *Pro iOS 5 Augmented Reality*. Apress, 2011.

[Rös09] A. Rösler. *Augmented Reality Games on the iPhone*. PhD thesis, Blekinge Institute of Technology, 2009.

[RZ10] Lucio Riccardi and Stefano Zingarini. Augmented Reality su iPhone: Il caso Meteo360. Presented at the WhyMCA conference, Milan, Italy, 2010.

[Shu] Tish Shute. Augmented Reality -- Bigger than the Web: Second Interview with Robert Rice from Neogence Enterprises. http://www.ugotrade.com/2009/08/03/augmented-reality-bigger-than-the-web-second-interview-with-

[TCD+02] Bruce Thomas, Ben Close, John Donoghue, John Squires, Phillip De Bondi, and Wayne Piekarski. First person indoor/outdoor augmented reality application: Arquake. *Personal and Ubiquitous Computing*, 6(2), 2002.

[TI01] J.C. Tang and E. Isaacs. Why do users like video? studies of multimedia-supported collaboration. *Sun Microsystems Laboratories The First Ten Years*, page 1, 2001.

[TJMI] Michael Tidwell, Richard S. Johnston, David Melville, and Thomas A. Furness III. The Virtual Retinal Display - A Retinal Scanning Imaging System. http://www.hitl.washington.edu/publications/p-95-1/.

[Val] Jim Vallino. Augmented Reality Page, Department of Software Engineering, Rochester Institute of Technology. http://www.se.rit.edu/~jrv/research/ar/index.html.

[VK10] E.E. Veas and E. Kruijff. Handheld devices for mobile augmented reality. In *Proceedings of the 9th International Conference on Mobile and Ubiquitous Multimedia*, page 3. ACM, 2010.

[Wal08] G. Waloszek. Magic Lens. http://www.sapdesignguild.org/goodies/visualization/controls/MagicLens.htm, 2008.

[Wei91] M. Weiser. The computer for the 21st century. *Scientific American*, 265(3):94--104, 1991.

[Wika] Wikipedia. 6dof. http://it.m.wikipedia.org/wiki/6dof.

[Wikb] Wikipedia. Augmented Reality. http://en.wikipedia.org/wiki/Augmented_reality.

[Wikc] Wikipedia. Objective-C. http://it.wikipedia.org/wiki/Objective-C.

[Wikd] Wikipedia. Rotazioni rpy. http://it.wikipedia.org/wiki/Rotazioni_rpy.

[Wike] Wikipedia. XCode. http://en.wikipedia.org/wiki/Xcode.