Alma Mater Studiorum · Università di Bologna · Campus di cesena

FACOLTÀ DI SCIENZE MATEMATICHE FISICHE E NATURALI

Corso di Laurea in Scienza e Tecnologie Informatiche

Analisi e sperimentazione della piattaforma Goblin XNA per applicazioni di realtà aumentata

Tesi di Laurea in Scienze e Tecnologie Informatiche

Relatore:
Chiar.mo Prof.
Dario Maio

Presentata da:
Thomas Trapanese

Sessione Prima
Anno Accademico 2012/2013

"Reality leaves a lot to the imagination."

John Lennon

Indice

Elenco delle figure **1**

Elenco dei codici **5**

1 Introduzione **5**
1.1 Realtà aumentata: cenni storici 5
1.2 Realtà aumentata: definizione 7
1.3 Tipologie di realtà aumentata 8

2 Applicazioni **13**
2.1 Applicazioni mediche 13
2.2 Applicazioni ludiche 14
2.3 Applicazioni informative 15
2.4 Applicazioni ingegneristiche 16

3 Framework e librerie **17**
3.1 Open Source Computer Vision 17
3.2 BaZar 18
3.3 ARToolKit 18
3.4 ALVAR 21
3.5 OpenSpace3D 21
3.6 SSTT 21
3.7 Vuforia 22
3.8 Goblin XNA 22
3.9 D'Fusion 23
3.10 POPCODE/Zappar 23
3.11 Altri framework 24

4 Applicazione **31**
4.1 Scelta del framework 31
4.2 Struttura di GoblinXNA 32
4.2.1 Tools 33
4.3 Struttura di un progetto Goblin 34
4.4 Struttura di una Scena 36
4.5 Struttura dell'applicazione 39
4.5.1 ArApp 40

4.5.2 Engine 44
4.5.3 libBox 45
4.5.4 libShip 47
4.5.5 libActionButton 50

Conclusioni **53**

Bibliografia **55**

Elenco delle figure

1.1 Bozzetto allegato al brevetto di sensorama 6
1.2 Utente che utilizza sensorama 6
1.3 Reality-Virtuality continuum 6
1.4 Vista di ARQuake attraverso HDR 7
1.5 Hardware ARQuake 7
1.6 Evoluzione temporale dei marker 9
1.7 Esempio di riconoscimento dei bordi 10

2.1 Sovrapposizione di immagini sui pazienti 14
2.2 ArGames per Nintendo 3DS 15
2.3 EyePet per Sony Playstation 3 15
2.4 Informazioni aggiunte al panorama da Layar 16
2.5 Funzione di navigazione satellitare di Wikitude 16
2.6 Framework metaio utilizzato nella semplificazione di processi industriali 16

3.1 Riconoscimento facciale tramite OpenCV 18
3.2 Riconoscimento Marker tramite OpenCV 18
3.3 Esempio realizzato con ARMedia 19
3.4 Esempio realizzato tramite AndAR 19
3.5 Esempio relizzato tramite la libreria LinceoVR 20
3.6 Videocameta Wowee Rovio, supportata da LinceoVR 20
3.7 Applicazioni di esempio prese dalla documentazione di Vuforia 22
3.8 Applicazioni realizzate tramite POPCODE 24
3.9 Applicazioni realizzate tramite Aurasma 25
3.10 Trasformazione applicata a una lattina tramite grazie alle primitive di Obvious-Engine 26
3.11 Tracking aereo con Robocortex 27
3.12 Applicazione industriale per ipad 27

4.1 Marker ALVAR 33
4.2 Struttura della scena in GoblinXNA 36
4.3 Vuzix iWear VR920 39
4.4 Vuzix Wrap 920 39
4.5 Diagramma dei componenti 39

4.6 Sistema di riconoscimento per il caricamento dinamico in ArApp 43
4.7 Diagramma delle classi del progetto Engine 45
4.8 Visualizzazione dell'oggetto generato dal progetto libBox . . 47
4.9 Rendering dell'oggetto Ship con e senza DebugMode attivo . 49
4.10 Diagramma delle classi del progetto libActionButton 51
4.11 Pulsante generato nel progetto libActionButton 52
4.12 Risultato finale dell'applicazione 52
4.13 Prototipo dei Google Glass di prossima uscita 53
4.14 Progetto di retina-display con celle solari 53

Codici

4.1 Esempio di file xml per i marker di ALVAR 33
4.2 Scheletro di un'applicazione GoblinXNA 35
4.3 Inizializzazione della sorgente video 40
4.4 Inizializzazione del tracker 41
4.5 Caricamento degli oggetti 42
4.6 Gestione degli eventi della tastiera 43
4.7 Gestione rotazione del cubo in libBox 46
4.8 Metodo CreateObject() della classe Ship 48
4.9 Metodo goOn() della classe Movement 49
4.10 Metodi goUp() e goDown() della classe Movement 49
4.11 Metodi turnLeft() e turnRight() della classe Movement . . . 50
4.12 Funzione isClick() della classe Button 51

Capitolo 1

Introduzione

In questo capitolo introduttivo sono mostrati i principali fatti storici che hanno portato alla nascita e alla diffusione della realtà aumentata.Inoltre, il capitolo contiene un'analisi dell'evoluzione temporale del concetto di realtà aumentata e a corredo di ciò è illustrato l'hardware che caratterizza un moderno sistema AR.

1.1 Realtà aumentata: cenni storici

Il primo approccio verso un ambiente virtuale risale al 1957 quando Morton Helig realizzò "Sensorama", un dispositivo per riprodurre filmati. Grazie all'utilizzo di varie tecnologie, dalle immagini 3D al feedback tattile, esso riusciva a stimolare i sensi dell'utilizzatore creando una sensazione di immersione mai realizzata prima (Figura ?? e Figura ??). Sensorama creava delle sensazioni statiche, nel senso che l'utente era immobile e ad esso venivano forniti stimoli esterni.

Il progetto non riscosse un grande successo e fu presto abbandonato, ma i risultati raggiunti illustravano già allora i concetti di realtà virtuale e stimoli sensoriali generati artificialmente che rappresentano ancora oggi le caratteristiche fondanti della realtà aumentata.

Una prima versione dinamica di ambiente virtuale si ebbe grazie all'invenzione degli Head Mounted Display (HDM) ad opera di Ivan Sutherland [1] che nel 1968 realizzò il primo prototipo di casco per la realtà aumentata.

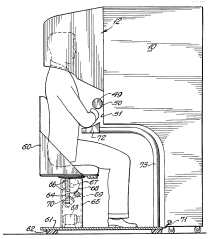

FIGURA 1.1: Bozzetto allegato al brevetto di sensorama

FIGURA 1.2: Utente che utilizza sensorama

"The Sword of Damocles" era un dispositivo in grado di riprodurre un ambiente virtuale in un display binoculare in cui la prospettiva si adattava all'orientamento dello sguardo dell'utilizzatore. Le immagini riprodotte erano limitate a dei semplici wire-frame[1].

Per il primo esempio di interazione occorre attendere il 1975, quando Myron Krueger creò "Videoplace" un ambiente di realtà virtuale che sfruttava proiettori, videocamere e schermi al fine di realizzare l'interazione con l'utente.

Nel 1990 Tom Caudelle coniò il termine "Realtà aumentata" mentre lavorava per la Boing a un progetto atto ad aiutare gli ingegneri nella costruzione e nella manutenzione dei veicoli. L'idea alla base del progetto di Caudelle era quella di utilizzare degli HDM con gli schemi elettrici precaricati che visualizzassero la posizione di determinati cavi all'interno dell'aereo e aiutassero il loro corretto posizionamento [2].

Nel 1994 Paul Milgram introdusse il concetto di realtà mista (Figura ??) vista per indicare il passaggio graduale da un ambiente reale ad un ambiente completamente; virtuale[3]

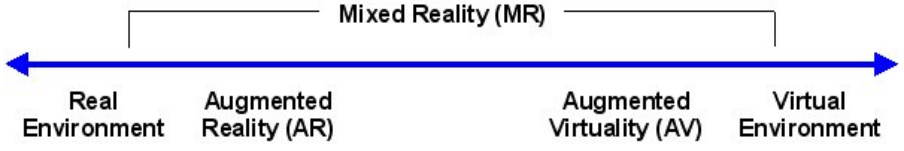

FIGURA 1.3: Reality-Virtuality continuum

tra questi due estremi troviamo la realtà aumentata, nella quale le informazioni virtuali vengono integrate nel mondo reale, e la virtualità aumentata

[1] rappresentazione grafica di oggetti tredimensionali che si limita a visualizzare i bordi delle immagini che risultano trasparenti

in cui, al contrario, informazioni reali vengono introdotte in un ambiente completamente virtuale.

Nel 2002 Bruce H. Thomas realizzò il primo videogioco in realtà aumentata per utilizzo esterno; si trattava di ARQuake (Figura 1.4 e Figura 1.5), il cui nome derivava dal suo omonimo per computer[4].

I giocatori, tramite degli HDR, visualizzavano immagini del videogame muovendosi nel mondo reale. ARQuake utilizzava un sensore GPS, un sensore di orientamento abbinati a un comune laptop posizionato in uno zaino e un controller per creare l'esperienza di gioco.

FIGURA 1.4: Vista di ARQuake attraverso HDR

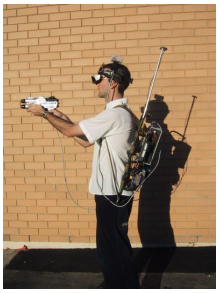

FIGURA 1.5: Hardware ARQuake

1.2 Realtà aumentata: definizione

L'espressione *augmented reality* compare nella letteratura scientifica sin dai primi anni 40 ma assume un significato concreto solo alla fine degli anni 80 quando, come già accennato,

Tom Caudelle ne fornì una prima definizione formale in relazione all'utilizzo di Head-Mounted Display.

Nel 1997 Azuma estese il termine a una più vasta tipologia di applicazioni su dispositivi differenti definendola in base a tre caratteristiche:

1. combinazione di reale e virtuale;
2. interattività in real-time con l'ambiente;
3. visualizzazione tre-dimensionale degli oggetti virtuali.

Una definizione ancor più estesa venne fornita nel 2005 da Paolo Magrassi, un tecnologo italiano che nel suo libro 2015 Weekend nel futuro valutava le future evoluzioni di questa tecnologia definendola come:

> *"L'arricchimento della percezione sensoriale umana mediante informazioni, in genere manipolate e convogliate elettronicamente, che non sarebbero percepibili con i cinque sensi".*

In questo caso il concetto di realtà aumentata non è più limitato al solo senso della vista ma coinvolge l'intera sfera sensoriale umana, limitandosi a definire genericamente l"'aumento" come un'aggiunta di un certo tipo di informazione elaborata da un computer, a quella che sarebbe la normale percezione della persona.

1.3 Tipologie di realtà aumentata

Al fine di classificare le varie tipologie di realtà aumentata, possiamo distinguere tre macro categorie:

- realtà aumentata basata sull'utente;
- realtà aumentata basata sui marker;
- realtà aumentata basata sulla posizione.

La realtà aumentata basata sull'utente è il tipo più semplice, in quanto nelle sue applicazioni la realtà serve solo da sfondo passivo e non partecipa alle interazioni con l'utente. Un esempio di questa tipologia di realtà aumentata è data da ARQuake; le immagini aumentate vengono generate da algoritmi che sfruttano i sensori per conoscere la direzione del punto di vista ed elaborare le immagini di conseguenza.

Nella realtà aumentata basata sui marker, invece, l'obiettivo è quello di riconoscere, in un flusso di video, determinati oggetti di geometria nota chiamati appunto marker. Una volta riconosciuto l'elemento è possibile determinarne orientamento, posizione e scala e questo consente di aggiungere informazioni coerenti con la prospettiva.

I marker hanno subito una notevole evoluzione negli ultimi anni(Figura 1.6), passando da marker mono-dimensionali quali i semplici codici a barre a versioni sempre più avanzate. I QR-code (Quick Response code) sono codici a barre bidimensionali appositamente studiati per il riconoscimento tramite fotocamera e la memorizzazione di piccole quantità di informazioni. A causa dell'eccessiva sensibilità al movimento dell'algoritmo utilizzato per il rilevamento dei QR-Code sono stati introdotti marker appositamente studiati per la realtà aumentata e per il riconoscimento all'interno di un flusso video in tempo reale. Oggi si cerca di sviluppare algoritmi per il riconoscimento di oggetti del mondo reale da utilizzare come marker naturali per aumentare il livello di realismo e le possibilità di utilizzo. Il riconoscimento delle immagini è progredito molto negli ultimi anni e, anche grazie all'aumento della potenza di calcolo disponibile, oggi è possibile rilevare un volto umano, una mano, o anche un oggetto all'interno di un'immagine. L'utilizzo di questi algoritmi per la definizione di marker naturali è attualmente limitato dalla potenza di calcolo necessaria per eseguire il riconoscimento in real-time.

L'utilizzo dei marker naturali comporta alcuni problemi, visto che gli algoritmi di riconoscimento attuali non sono in grado di distinguere gli oggetti reali dalle loro rappresentazioni. Ad esempio un volto viene riconosciuto indipendentemente dal fatto che esso sia realmente presente in una scena o che sia semplicemente rappresentato in una foto stampata. Superare questo limite rappresenta una delle frontiere del riconoscimento delle immagini e permetterà l'utilizzo di marker reali nelle applicazioni di realtà aumentata.

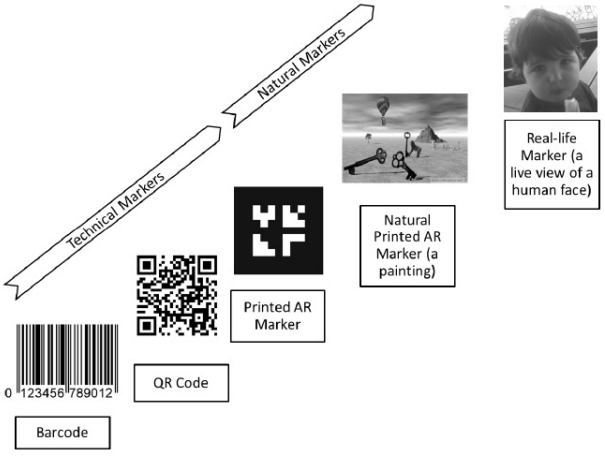

FIGURA 1.6: Evoluzione temporale dei marker

La terza tipologia di applicazioni di realtà aumentata sfrutta dati di posizione e orientamento ottenuti tramite vari sensori per sovrapporre informazioni sugli oggetti del mondo circostante. GPS (Global Position System), accelerometri e giroscopi vengono utilizzati per stabilire posizione e orientamento del dispositivo utilizzato (generalmente uno smartphone o un tablet). Generalmente queste applicazioni sfruttano un database di POI (Point Of Interest) che contiene le coordinate geografiche dei punti da evidenziare. In aggiunta, alcune di queste applicazioni sfruttano algoritmi di riconoscimento dei margini per ottenere un posizionamento più accurato (Figura 1.7).

FIGURA 1.7: Esempio di riconoscimento dei bordi

Componenti Hardware

In questa sezione sono illustrati le principali componenti hardware che vengono comunemente utilizzate per la realizzazione della maggior parte dei sistemi in realtà aumentata.

Global Positioning System (GPS) Il GPS è un sistema di geo-localizzazione che sfrutta una serie di satelliti in orbita per fornire latitudine, longitudine e altitudine. Pur non essendo l'unico sistema di posizionamento satellitare, esso è sicuramente il più utilizzato per via del numero di satelliti di cui dispone.

Viene utilizzato da tutte quelle applicazioni che si basano sulla posizione geografica per fornire informazioni.

Giroscopio Il giroscopio è un dispositivo che mantiene un asse orientato in una direzione fissa ed è utilizzato nei dispositivi portatili come bussola per indicare il nord.

E' spesso utilizzato insieme al GPS per fornire l'orientamento per la rappresentazione delle immagini in realtà aumentata

Accelerometro L'accelerometro è un dispositivo che permette di misurare l'accelerazione. I dispositivi moderni vengono forniti con accelerometri in grado di misurare l'accelerazione lungo i tre assi consentendo di calcolare gli angoli di inclinazione rispetto alla verticale.

Head-Mounted Display (HDM) Gli HDM sono dei display montati su un casco all'altezza degli occhi, vengono utilizzati per visualizzare le informazioni in realtà aumentata. Ne esistono di due tipologie, ottici e video. Gli HDM ottici sono costituiti da schermi semitrasparenti che permettono di vedere attraverso il display e questo permette di visualizzare solo le informazioni generate. La seconda tipologia elabora le immagini di una videocamera e visualizza su un display opaco il risultato della sovrapposizione delle immagini della videocamera e di quelle generate digitalmente.

Smartphone/Tablet In tempi recenti questi dispositivi si sono diffusi a tal punto da diventare il punto di riferimento per la realtà aumentata. Smartphone e tablet dispongono di una moltitudine di sensori (accelerometri, GPS, giroscopi ecc..) e questo ha portato alla nascita di molte applicazioni in grado di sfruttarli per generare un ambiente in realtà aumentata e alla nascita di molti SDK mirati a semplificare la creazione di applicazioni in AR. Nel Capitolo 3 Ilustrerò i framework più conosciuti e alcuni esempi di applicazioni che li sfruttano.

Proiettori Un tipo particolare di realtà aumentata è quello che sfrutta dei proiettori per visualizzare le informazioni aggiuntive. A differenza di quanto accade con HDM e dispositivi portatili in questo caso non sono

presenti monitor ma le informazioni vengono proiettate direttamente nell'ambiente che circonda l'utente. Un esempio di applicazione che sfrutta dei proiettori è Videoplace

Capitolo 2

Applicazioni

In questo capitolo è presente una panoramica dei principali campi di applicazione della realtà aumentata mostrando alcuni esempi particolarmente rilevanti.

2.1 Applicazioni mediche

La realtà aumentata può essere estremamente utile in campo medico, sia per agevolare l'apprendimento sia assistendo i chirurghi durante le operazioni di chirurgia poco invasiva. La video chirurgia ha permesso di realizzare interventi chirurgici anche complessi nei quali il chirurgo opera solo piccole incisioni e si muove basandosi su immagini acquisite da microcamere inserite nelle incisioni stesse e visualizzate su appositi monitor. Un evoluzione naturale di questa tecnologia è rappresentata dall'utilizzo della realtà aumentata attraverso la quale il medico può visualizzare direttamente sul paziente in semitrasparenza dati derivanti da TAC, radiografie e risonanza magnetica(Figura 2.1). L'aggiunta di informazioni direttamente sul paziente rende più naturale il lavoro del chirurgo e riduce le possibilità di errore[5].

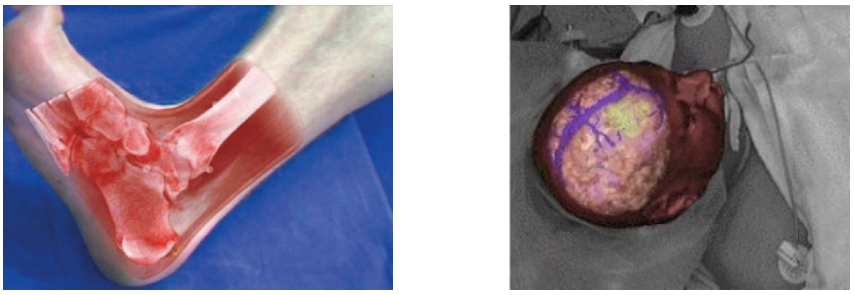

FIGURA 2.1: Sovrapposizione di immagini sui pazienti

2.2 Applicazioni ludiche

Sin dai primi esperimenti di realtà aumentata furono evidenti le potenzialità ludiche di questa tecnologia. Già Sensorama, pur avendo una funzione più che altro dimostrativa, mirava a colpire per le sue capacità di intrattenimento e così pure ARQuake che fù uno dei primi videogiochi in realtà aumentata. Oggi il campo dei videogiochi è estremamente prolifico e le grandi case produttrici utilizzano concretamente la realtà aumentata anche a livello commerciale, a dimostrazione della maturità raggiunta da questa tecnologia.

Recentemente la Nintendo ha lanciato due giochi in realtà aumentata per la propria console portatile, in Nintendo 3DS: ArGames (Figura 2.2) e Face Raider. Il primo è un videogioco che sfrutta delle carte, utilizzate come marker, per animare dei personaggi attraverso la videocamera del dispositivo e include diverse modalità di gioco nelle quali l'interazione avviene attraverso lo schermo touch-screen. Il secondo sfrutta la tecnologia di riconoscimento facciale e i sensori del dispositivo (accelerometro e giroscopio) per realizzare uno shooter game nel quale il giocatore dovrà colpire dei palloncini con dei volti disegnati sopra, il tutto usando come sfondo le immagini riprese dalla videocamera posteriore.

Un altro gioco per console che utilizza ampiamente la realtà aumentata è EyePet per Sony Playstation 3 (Figura 2.3) il cui scopo è quello di interagire con una scimmietta virtuale. Attraverso una carta utilizzata come marker, il sistema riconosce il piano di appoggio sul quale "dar vita" all'animale, grazie alla tecnologia di riconoscimento delle immagini. L'interazione con l'animale virtuale avviene tramite dei gesti delle mani, e in questo modo diventa possibile accarezzare (virtualmente) l'animaletto che reagirà al tocco

muovendosi di conseguenza. Il gioco risponde inoltre anche ai rumori acquisiti grazie al microfono ed è in grado di riconoscere (e animare) particolari disegni realizzati dall'utente su un pezzo di carta.

FIGURA 2.2: ArGames per Nintendo 3DS

FIGURA 2.3: EyePet per Sony Playstation 3

2.3 Applicazioni informative

Le applicazioni informative rappresentano una fetta importante di quello che è oggi il mercato delle applicazioni per dispositivi mobile. Tra queste si distinguono diversi browser in realtà aumentata, applicazioni che permettono di visualizzare in sovrapposizione alle immagini riprese dalla videocamera informazioni sugli edifici che ci circondano, sui luoghi di interesse, sugli eventi nelle vicinanze, indicando caratteristiche e distanza di ognuno(Figura 2.4 e Figura 2.5). Attualmente le principali applicazioni che si contendono il mercato sono Layar e Wikitude, cioè browser in realtà aumentata che sfruttano i dati provenienti da accelerometri, GPS e giroscopio al fine di filtrare e visualizzare le informazioni ottenute interrogando dei web services che mettono a disposizione liste di Point Of Interest. Il punto di forza di questo tipo di applicazioni è l'espandibilità cioè la possibilità di sviluppare i propri web service. Sia Layar che Wikitude, infatti, mettono a disposizione il loro SDK a gli sviluppatori e ad oggi layar conta più di 3000 layer differenti per la maggior parte sviluppati da terzi. La società proprietaria del marchio si occupa solamente della validazione e della pubblicazione dei nuovi strati[6]. Wikitude è stato invece eletto da diverse testate giornalistiche come la miglior applicazione in realtà aumentata del 2012[7]; va fatto notare a tal proposito che nella lista delle migliori 5 applicazioni in realtà aumentata stilata dal The Telegraph tre sono web browser, a dimostrazione dell'interesse che suscitano queste tipologie di servizio.

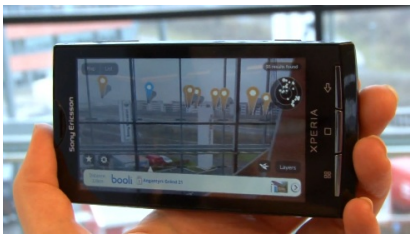

FIGURA 2.4: Informazioni aggiunte al panorama da Layar

FIGURA 2.5: Funzione di navigazione satellitare di Wikitude

2.4 Applicazioni ingegneristiche

Le applicazioni in campo ingegneristico sono di notevole interesse; ad esempio, attraverso applicativi ad-hoc, è possibile assistere i lavoratori nella fase di costruzione di apparecchiature complesse sovrapponendo gli schemi progettuali direttamente sopra l'oggetto. La realtà aumentata in questo caso permette anche di valutare rapidamente lo stato di avanzamento dei lavori di costruzione e di valutare l'effettiva corrispondenza con il progetto.

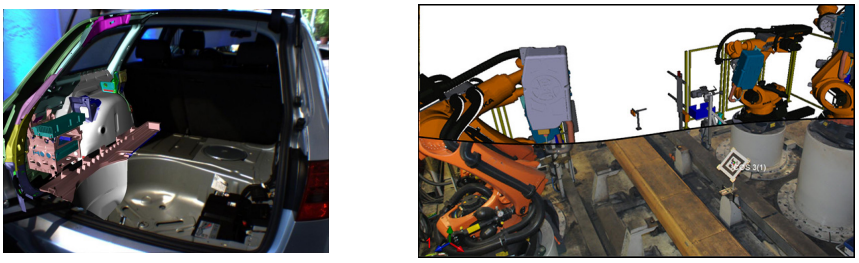

FIGURA 2.6: Framework metaio utilizzato nella semplificazione di processi industriali

Un altro punto di forza è rappresentato dalla semplificazione del lavoro di manutenzione, si pensi ad esempio alla possibilità di visualizzare in semitrasparenza tutto lo schema elettrico di un edifico, o di un veicolo. Per rispondere a queste esigenze sono nati molti framework di tipo commerciale che mirano ad automatizzare processi industriali. Ne è un esempio Metaio, un framework nato per le industrie che desiderano implementare sistemi di controllo e assistenza basati sulla realtà aumentata [8]. In figura 2.6 si può vedere un esempio di applicazione.

Capitolo 3

Framework e librerie

Nel corso degli ultimi anni si sono moltiplicati i framework disponibili per tutte le piattaforme e tutti i linguaggi, alcuni dei quali gratuiti e molti altri a pagamento. nel corso di questo capitolo saranno illustrate le caratteristiche dei principali framework che valutati in questo lavoro di tesi come base dell'applicativo che sarà illustrato nel prossimo capitolo.

3.1 Open Source Computer Vision

OpenCV è una libreria nata nei laboratori Intel nel 1999 e successivamente rilasciata sotto licenza open - source. Essa rappresenta uno dei principali contributi nell'ambito della visione artificiale e data la sua natura open ha visto nel corso degli anni una notevole crescita di funzionalità. Oggi integra funzioni che vanno dal riconoscimento facciale (Figura 3.1) alla realtà aumentata al riconoscimento di oggetti (Figura 3.2).

OpenCV è scritta in linguaggio C/C++e ciò le garantisce un'elevata efficienza oltre al fatto di essere multi-piattaforma. Oggi sono disponibili wrapper della libreria per molti linguaggi e molte piattaforme; nello specifico è stato valutato l'utilizzo della libreria per lo sviluppo su piattaforma android.

Le caratteristiche sin qui elencate fanno si che OpenCV sia utilizzata come base da diverse altre librerie[9].

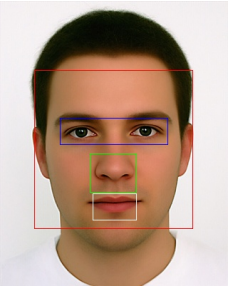

FIGURA 3.1: Riconoscimento facciale tramite OpenCV

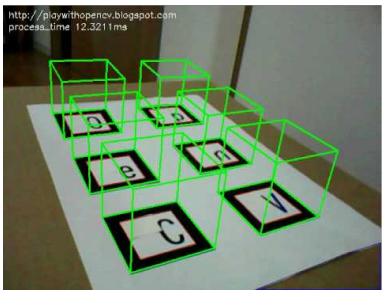

FIGURA 3.2: Riconoscimento Marker tramite OpenCV

3.2 BaZar

BazAR è una libreria per la visione artificiale che pone particolare interesse verso il riconoscimento di oggetti. Essa mette a disposizione le primitive per realizzare applicazioni in grado di riconoscere marker naturali e viene utilizzata come base da molti framework per realtà aumentata. [10].

3.3 ARToolKit

ARToolKit è una libreria nata appositamente per la realizzazione di applicazioni in realtà aumentata. Grazie a una videocamera e al riconoscimento di specifici marker consente di sovrapporre in maniera semplice oggetti generati al computer che seguono l'orientamento della videocamera, è disponibile per tutte le principali piattaforme sotto licenza sia open-source che commerciale [11].

Esistono molti sotto progetti correlati che si basano sulla sua libreria, alcune esempi notevoli sono:

ARMedia É un plugin per Google SketchUp, un ambiente di modellazione 3D, che aggiunge la possibilità di visualizzare i modelli in realtà aumentata, consente la realizzazione di modelli stand-alone che possono essere visualizzati tramite un apposito player (Figura 3.3). L'obiettivo della libreria è quello di consentire lo studio e l'analisi di modelli 3d in scala in ambienti di realtà mista [12].

AndAR AndAR è l'estensione di ARToolkit per il sistema operativo android, è un progetto open-source che fornisce un interfaccia java per l'accesso alle primitive di ARToolkit [13](Figura 3.4).

FIGURA 3.3: Esempio realizzato con ARMedia

FIGURA 3.4: Esempio realizzato tramite AndAR

Unity ARToolkit Un'insieme di plugin per Unity 3D, un game engine che consente lo sviluppo di videogiochi 3d per diverse piattaforme, che estendono le funzionalità dell'engine consentendo lo sviluppo di applicazioni in realtà aumentata . [14] UART si divide in tre plugin:

VideoWrapper: Permette di selezionare la videocamera e fornisce degli script in c# per catturare ed elaborare i frame aggiungendo delle texture

TrackerWrapper: Fornisce delle primitive per collegare degli elementi basati sul marker-tracking direttamente agli elementi di unity

VRPNWrapper: Consente di istanziare i traker sia lato client che server

LinceoVR É una soluzione che garantisce il supporto completo a Wowee Rovio (WiFi remote controlled AR Robots, Figura 3.6), una videocamera robotizzata controllata tramite wi-fi e a iWear di Vuzix (Figura 4.3), ochiali per la realtà aumentata con supporto per immagini tre-dimensionali.

LinceoVR dispone anche di un plugin per Microsoft PowerPoint che consente di aggiungere la l'esperienza della realtà aumentata alle proprie presentazioni, il plugin consente di accedere direttamente dalle proprie slide a un'interfaccia con LinceoVR[15] (Figura 3.5).

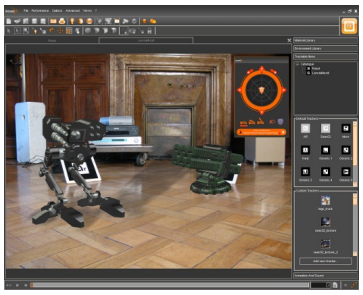

FIGURA 3.5: Esempio relizzato tramite la libreria LinceoVR

FIGURA 3.6: Videocameta Wowee Rovio, supportata da LinceoVR

osgART É una libreria multi-piattaforma scritta in c++, poggi su diverse librerie per la visione in realtà aumentata tra cui ARToolkit e BazAR, in combinazione con la libreria grafica 3D OpenSceneGraph ha come obbiettivo la semplificazione dello sviluppo fornendo un alto livello di astrazione[16].

Win AR É una piattaforma di sviluppo per la realtà aumentata su sistemi operativi windows, mette a disposizione diverse funzionalità utili agli sviluppatori tra cui la possibilità di importare modelli 3D in formato STL e 3DS ed effettuarne il rendering sulla scena in tempo reale. Win AR oltre a mettere a disposizione tutte le funzionalità di ARToolkit sfrutta la libreria BazAR che fornisce primitive di riconoscimento delle immagini che consentono il traking marker-less di oggetti in tempo reale[17].

SLARToolkit Creata da ARToolWorks, stessa casa produttrice di ARToolKit, è l'estensione della libreria per Silverlight[18].

FLARToolKit Anch'esso realizzato dalla casa madre di ARToolKit fornisce l'interfaccia per Flash Actionscript alle componenti della libreria, supporta tutte le i principali engine grafici per Flash (Papervision3D, Away3D, Sandy, Alternativa3D)[19].

3.4 ALVAR

ALVAR è sviluppato dal VTT Technical Research Centre of Finland. Anche in questo caso si tratta di un prodotto open-source.

Come riportato sul sito ufficiale del progetto, ALVAR è stato concepito per essere il più flessibile possibile, si basa sulla libreria OpenCV, offre tool di alto livello che consentono in poche righe di codice lo sviluppo di applicazioni performanti e di sicuro impatto visivo, inoltre fornisce anche un'interfaccia per le funzioni di basso livello delle OpenCV che permette all'utente di utilizzare i propri algoritmi per le funzioni fondamentali.

Tra le funzionalità più interessanti rientrano la possibilità di rimuovere un marker dalla scena in maniera dinamica, il riconoscimento di più marker contemporaneamente e la possibilità di utilizzare delle immagini come marker[20].

3.5 OpenSpace3D

É una soluzione Open Source per lo sviluppo di applicazioni interattive 3D real-time che sfrutta la libreria ALVAR , il suo punto di forza è rappresentato dalla facilità di utilizzo, consente di realizzare scene 3D limitando al minimo la scrittura di codice in quanto tutte le principali funzionalità sono già fornite dall'sdk. OpenSpace3D punta al web fornendo supporto alla tecnologia Flash. Il progetto è in continua evoluzione e mira a fornite il maggior supporto alla realtà aumenta, riconoscimento del parlato e alla computer grafica[21].

3.6 SSTT

Simplified Spatial Target Tracker è una libreria per il tracking di marker orientata alla realtà aumentata, offre supporto ai marker naturali ed è disponibile in varie forme per la maggior parte dei dispositivi, le specifiche indicano la compatibilità con Windows Mobile 5 e 6, Android ,Ios, Maemoo e symbian per il comparto mobile e per quando riguarda i sistemi desktop

il supporto comprende windows, linux e MacOs. La libreria sfrutta una versione ottimizzata delle OpenCV che aggiunge il supporto alle estensioni SIMS dei moderni processori, la libreria è scritta in c++ ma espone metodi di accesso in c standard al fine di garantire la massima portabilità. Per come è strutturata SSTT permette l'utilizzo con qualsiasi motore grafico 3D[22].

3.7 Vuforia

Vuforia è il framework per realtà aumentata realizzato da Qualcomm, si tratta di un prodotto molto potente che permette di realizzare applicazioni professionali, supporta XCode per lo sviluppo Ios e Eclipse per gli sviluppatori android tramite NDK. La libreria è realizzata in c++, permette la realizzazione di applicazioni di traking marker-less (Figura 3.7) e la gestione di complessi ambienti 3D, inoltre permette di integrare elementi di interfaccia nelle proprie applicazioni. Il sito ufficiale mette a disposizione degli sviluppatori una documentazione molto vasta con diverse guide e applicazioni di esempio, inoltre possiede una vasta community che garantisce supporto costante per ogni problema. Tra i framework visti fin'ora Vuforia è senz'altro quello meglio documentato[23].

FIGURA 3.7: Applicazioni di esempio prese dalla documentazione di Vuforia

3.8 Goblin XNA

Goblin XNA è una piattaforma di sviluppo per applicazioni 3D Open Source, come suggerisce il nome Goblin estende il framework XNA di Microsoft nato per agevolare lo sviluppo di videogiochi per le piattaforme Microsoft (Windows, Xbox, Windows Phone).

Goblin fornisce una vasta gamma di interfacce utente, non presenti nel framework Microsoft, da utilizzare nelle applicazioni 3D. Goblin poggia su diverse librerie esterne per agevolare il più possibile lo sviluppo di complessi ambiente 3D interattivi.

Per la fisica si ha la possibilità di sfruttare le librerie Newton o Havok entrambe Open Source. Goblin garantisce il supporto alla realtà aumentata tramite ALVAR (vedi 3.4) verso cui fornisce un'interfaccia in c# [24].

3.9 D'Fusion

D'Fusion è una delle soluzioni commerciali per la realtà aumentata più utilizzate al mondo, oltre a fornite un sdk fornisce anche una suite di prodotti che agevolano lo sviluppo, garantisce il supporto ad android/ios oltre che ad Adobe Flash e Unity 3D[25].

La suite si compone di 4 prodotti:

D'Fusion Mobile Appositamente studiata per il settore mobile offre il supporto per android e ios

D'Fusion @Home É studiata per la produzione via web o su supporti ottici, inoltre fornisce la possibilità di integrazione con Facebook.

D'Fusion Pro Utilizzato per la realizzazione di applicazioni professionali, aggiunge il supporto a video HD, videocamere multiple, videocamere infrarossi e sensori specifici.

D'Fusion Studio É la soluzione gratuita che utilizza la versione free dell'sdk, consente la creazione di modelli 3D e la loro integrazione nell'ambiente tramite l'utilizzo di tracker.

3.10 POPCODE/Zappar

Popcode è costituito da un framework e da una applicazione per la realtà aumentata, consente il tracking marker-less (Figura 3.8), i contenuti multimediali sono indicati dalla presenza del logo di POPCODE che, ripreso

attraverso l'applicazione dedicata, viene riconosciuto, i dati relativi all'implementazione vengono scaricati usando le informazioni presenti nel logo[26].

Il progetto non è più in sviluppo, il team si è spostato su un nuovo progetto Zappar[27] dalle caratteristiche simili ma esplicitamente orientato all'intrattenimento.

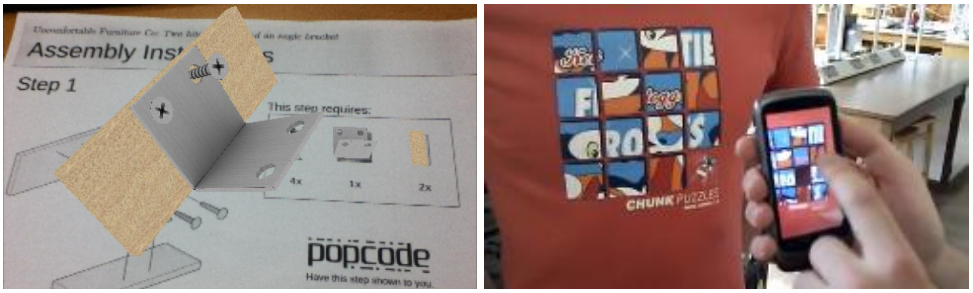

FIGURA 3.8: Applicazioni realizzate tramite POPCODE

3.11 Altri framework

Il campo della realtà aumentata è molto vasto, quelli sin ora elencati rappresentano le principali e più documentate soluzioni disponibili ma non rappresentano certamente le uniche. Di seguito verranno elencati alcuni framework "minori" con delle brevi descrizioni.

ARLab É un sdk ancora acerbo, il sito ufficiale offre una panoramica delle funzioni disponibili che attualmente coprono solo la possibilità di realizzare browser in realtà aumentata e il riconoscimento delle immagini. Sul sito ufficiale del progetto sono evidenziate altre funzionalità attualmente in sviluppo tra cui il rendering di oggetti 3D e il supporto a elementi di interfaccia. ARLab è disponibile in vari pacchetti per ios e android[28].

Aurasma Aurasma è un'applicazione/sdk per dispositivi mobile, offre supporto ad android e ios, si tratta di un prodotto maturo che permette di sviluppare applicazioni avanzate(Figura 3.9), utilizza il riconoscimento delle immagini come marker. Aurasma offre un browser espandibile tramite l'sdk

ceh permetti di visualizzare le "Aure", animazioni, oggetti 3D immagini e contenuti digitali.

La "A" del logo di Aurasma su riviste/giornali indica la presenza di contenuti che possono essere visualizzati tramite il browser. Aurasma fornisce anche una mappa che consente di visualizzare le Auree nelle vicinanze[29].

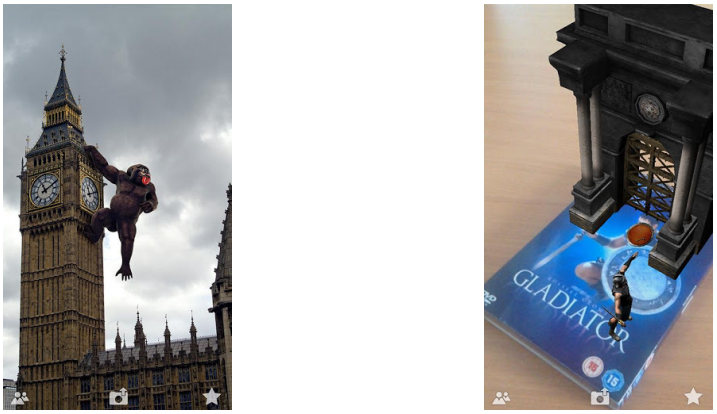

FIGURA 3.9: Applicazioni realizzate tramite Aurasma

Google Goggles L'estensione per la ricerca in realtà aumentata del famoso motore di ricerca, riconosce testi, immagini e loghi che utilizza per effettuare ricerche[30].

IN2AR IN2AR è un sdk con supporto alle piattaforme Adobe Flash, Adobe AIR e Unity 3D, consente lo sviliuppo di applicazioni multipiattaforma in realtà aumentata in grado di utilizzare il tracking di innagini utilizzate come marker[31].

AR23D É un sdk orientato al mondo mobile, offre supporto a ad android, ios, blackbarry os e symbian, consente il tracciamento di immagini e dispone di funzionalità di riconoscimento facciale. É disponibile sia in licenza gratuita che a pagamento per fini commerciali[32].

Cortexica É una delle applicazioni leader nel settore mobile per il riconoscimento e la ricerca visuale delle immagini, il funzionamento è semplice, si scatta una fotografia e l'applicazione si occupa di effettuare la ricerca su una base dati preconfigurata o sulla rete tramite google[33].

Obvious Engine Disponibile solo in versione commerciale, la libreria fornisce supporto per andorid, ios e Unity 3D. Obvious è in grado di utilizzare marker naturali (Figura 3.10) ai quali permette di applicare degli effetti grafici[34].

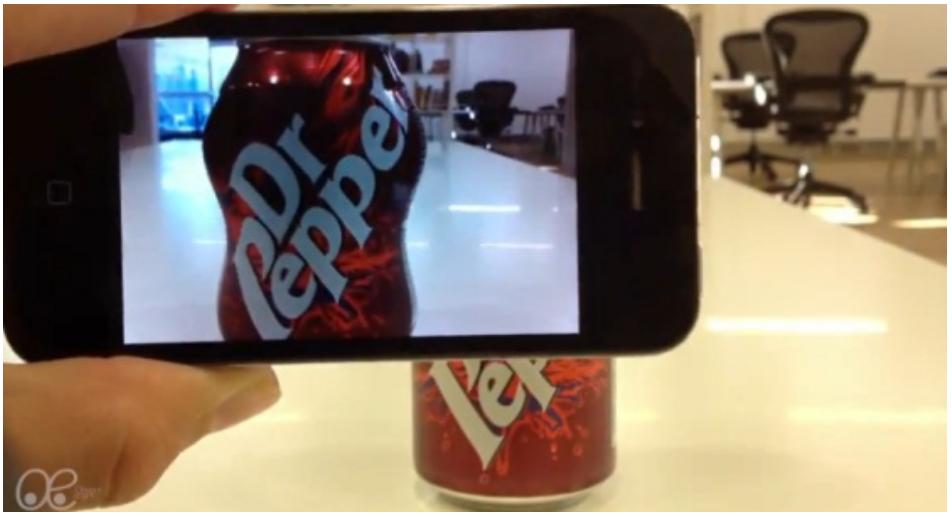

FIGURA 3.10: Trasformazione applicata a una lattina tramite grazie alle primitive di Obvious-Engine

Mixare mix Augmented Reality Engine, è una browser per la realtà aumentata Open Source. Disponibile per android e ios è strutturato come applicazione totalmente autonoma, consente di realizzare le proprie applicazioni standalone[35] a differenza di altri browser che hanno un controllo più centralizzato (Wikitude, Layar vedi 2.3).

NyARToolkit Soluzione Open Source, fornisce librerie di supporto per i linguaggi più comuni, c#, Java, Unity 3D e android. Si tratta certamente di una libreria che copre tutti gli aspetti principali della realtà aumentata, purtroppo si tratta di un progetto giapponese che attualmente dispone solo di documentazione in lingua originale[36].

3DAR - PanicAR Sono due librerie dalle caratteristiche molto simili, entrambe sono studiate specificatamente per l'utilizzo in ios, consentono di sostituire la widget per le mappe di default con una personalizzata che include un browser in realtà aumentata, il suo punto di forza di queste soluzioni è nell'immediatezza dell'implementazione anche se il supporto è limitato alla solo piattaforma Apple[37][38].

Robocortex É un framework per realtà aumentata con uno sguardo particolare alle aziende, molto professionale, fornisce primitive per effettuare il tracking di oggetti al fine della videosorveglianza aerea (Figura 3.11 e Figura 3.12), la tecnologia Robocortex viene utilizzata anche per la realizzazione di autoveicoli senza pilota, un automezzo senza pilota riesce, grazie all'sdk, a tracciare i movimenti di un altro veicolo e a seguirlo[39].

FIGURA 3.11: Tracking aereo con Robocortex

FIGURA 3.12: Applicazione industriale per ipad

ArUco si tratta di una libreria minimale scritta in c++ che poggia sulle openCv fornendone un più alto livello di astrazione. ArUco permette di gestire il riconoscimento di marker sia singoli che composti in tavole, nasce per essere il più possibile portabile, a tal fine sfrutta le OpenGL come libreria grafica di riferimento per la visualizzazione di modelle tre-dimensionali[40].

Awila Soluzione improntata alla realizzazione di applicazioni geo-spaziali, è costituita da un SDK e un Content Management Solutions. É realizzata in java con supporto anche per dispositivi android, fornisce un hosting per pubblicare i propri dati e consentirne la visualizzazione all'interno dell'applicativo[41].

String Framework per ios, rende possibile realizzare applicazioni in realtà aumentata che materializzano e animano oggetti tre-dimensionali in sovrapposizione a marker naturali[42].

XLoudia É una libreria che sfrutta il cloud computing per il riconoscimento delle immagini, si vanta di avere ottimi tempi di risposta e un

altissimo livello di affidabilità, i dati presenti sul sito ufficiale indicano un tempo di risposta di 0.2 sec e una livello di affidabilità nel riconoscimento del 99,5%.É disponibile per android e ios e non dispone di una versione gratuita[43].

PointCloud Questo framework consente sia il traking di oggetti che la funzionalità di browser in realtà aumentata, la libreria nasce per ios ma è disponibile anche una versione web che sfrutta javascript e html5 per fornire un esperienza di realtà aumentata via browser. É disponibile gratuitamente per fini non commerciali[44].

Nome	Piattaforma	Marker	Licenza
3DAR	Ios	No	Free + Opzione Commerciale
ALVAR	Ios, Android, Flash, Silvertligth	Sì anche naturali	OpenSource
AndAR	Android	Sì	OpenSource
AR23	Ios, Android	Sì	Free + Opzione Commerciale
ARLab	Ios, Android	Qr-Code	Free + Opzione Commerciale
ARMedia	Ios, Android, Windows, Mac Os, Linux	No	Free + Opzione Commerciale
ARToolkit	Ios, Android	Sì anche naturali	Free + Opzione Commerciale
ArUco	Ios, Android, Windows, Mac Os, Linux	No	OpenSource
Aurasma	Ios, Android	Sì anche naturali	Free + Opzione Commerciale
Awila	Ios	No	OpenSource
BaZar	Ios, Android, Windows, Mac Os, Linux	Offre primitive per implementarli	Free + Opzione Commerciale

Cortexica	Ios, Android, Windows Mobile	No	Free
D'Fusion	Ios, Android, Flash	Sì anche naturali	Free + Opzione Commerciale
FLARToolkit	Flash	Sì	OpenSource
GoblinXNA	Windows, Windows Mobile, XBox	Sì	Free + Opzione Commerciale
Gooogles	Ios, Android	No	Free + Opzione Commerciale
IN2AR	Flash	Sì anche naturali	Free + Opzione Commerciale
Layar	Ios, Android	Sì anche naturali	Free + Opzione Commerciale
LinceoVR	Plugin per: PowerPoint, 3DStudio Max, Rhino	No	OpenSource + SDK Commerciale
Mixare	Ios, Android	No	OpenSource
NyARToolkit	Android, Windows	No	OpenSource
Obvious Engine	Ios, Android	No	Commerciale
OpenSpace 3D	Flash	Sì	OpenSource
OpenCV	Ios, Android, Windows, Mac Os, Linux	Offre primitive per implementarli	OpenSource
osgART	Ios, Android, Windows, Mac Os, Linux	Sì anche naturali	OpenSource
PanicAR	Ios	No	Free + Opzione Commerciale
PointCloud	Ios	No	Free + Opzione Commerciale
POPCODE	Ios, Android	Sì, marker proprietario	Commerciale

Robocortex	Ios, Android, Windows, Mac Os, Linux	Sì anche naturali	Free + Opzione Commerciale
SLARToolkit	Windows Mobile	Sì	OpenSource
SSTT	Ios, Android, Windows Mobile, Windows, Mac Os, Linux	Sì anche naturali	OpenSource
String	Ios	Sì	Free + Opzione Commerciale
Unity ARToolkit	Unity3D	No	Free
Vuforia	Ios, Android	Sì anche naturali	Free + Opzione Commerciale
WinAR	Windows	Sì anche naturali	Free + Opzione Commerciale
Xloudia	Ios, Android, Unity3D	Sì anche naturali	Commerciale
Wikitude	Ios, Android, BlackBarry OS	Sì anche naturali	Free + Opzione Commerciale
Zappar	Ios, Android	Sì, marker proprietario	Commerciale

Tabella 3.1: Tabella riassuntiva dei framework AR

Capitolo 4

Applicazione

4.1 Scelta del framework

Nella scelta del framework da utilizzare sono stati valutati diversi aspetti:

I - Tipoogia di applicazioni Come illustrato precedentemente (1.3) le applicazioni in realtà aumentata appartengono a diverse tipologie, nella realizzazione dell'applicativo si è preferito un orientamento verso applicazioni di riconoscimento dei marker, questo esclude diversi candidati tra quelli illustrati nel capitolo 2.

II - Il livello di astrazione Il framework deve fornire metodi che permettano di realizzare applicazioni grafiche in realtà aumentata senza richiedere l'implementazione diretta di algoritmi per il riconoscimento delle immagini da parte dello sviluppatore, deve fornire i metodi fondamentali per l'interfacciamento con l'hardware e per il riconoscimento di marker e per il rendering di oggetti tre-dimensionali.

III - Linguaggio di programmazione Questo rappresenta un aspetto fondamentale nella scelta di un framework indipendentemente da quale sia il suo campo di utilizzo, la scelta di un linguaggi di programmazione nel quale si possiede esperienza permette di ridurre il tempo di apprendimento dei metodi messi a disposizione dal framework.

IV - Solidità del sistema Il framework scelto dovrà essere ben testato e collaudato, questo esclude diversi framework attualmente in sviluppo o in fase di testing.

Dopo aver valutato i vari framework la scelta è ricaduta su GoblinXNA in quanto:

- Consente di sviluppare applicazioni in grafica 3D con riconoscimento in real-time di marker.
- Per il riconoscimento delle immagini si appoggia ad ALVAR e OpenCV di cui fornisce un alto livello di astrazione; questo esenta lo sviluppatore dall'implementare i molteplici algoritmi di elaborazione delle immagini necessari.
- É un basato scritto in c#, un linguaggio fortemente orientato agli oggetti molto potente e che mette a disposizione un IDE (Visual Studio) tra i più evoluti del mercato, inoltre sfrutta librerie native in c/c++ (ALVAR e OpenCV), questo limita il calo di prestazioni dovuti alla macchina virtuale .NET
- GoblinXNA è attualmente alla versione 4.1, è molto utilizzato e testato, poggia su infrastrutture consolidate come il framework XNA di Microsoft e questo fornisce una grande robustezza al sistema.

4.2 Struttura di GoblinXNA

Goblin XNA viene fornito con una soluzione .net contenente 18 progetti, ognuno di questi illustra una funzionalità messa a disposizione. L'installazione del pacchetto risulta difficoltosa, Goblin sfrutta molte librerie open source quali ALVAR, OpenCV, Havok, Newton, ognuna di queste librerie deve essere scaricata separatamente dal relativo sito ufficiale e includa nella soluzione di esempio, ovviamente è richiesta anche l'installazione del framework .net con l'estensione XNA. Sul sito è comunque possibile reperire la guida dettagliata di installazione.

4.2.1 Tools

Goblin viene fornito con una serie di tools, di questi i più interessanti ai fini della realtà aumentata sono il tool di calibrazione della videocamera e il tool di generazione dei marker che non sono altro che wrapper in c# per le funzioni di ALVAR.

Il tool di calibrazione permette di calibrare la videocamera utilizzando un marker di default (preso dalla guida ufficiale di ALVAR), al termine della calibrazione viene generato un file xml che può essere utilizzato nei propri progetti Goblin per inizializzare la videocamera.

Molto importante per gli scopi di questa trattazione è il tool per la realizzazione di marker personalizzati.

FIGURA 4.1: Marker ALVAR

Per riconosce un marker ALVAR sfrutta le informazioni contenute in un file xml (Codice 4.1) che ne rappresenta la struttura, ad esempio il marker in figura 4.1 è rappresentato dal file xml:

```
<?xml version="1.0" encoding="utf-8" standalone="no"?>
<multimarker markers="1">
  <marker index="0" status="2">
    <corner x="0" y="-10,8" z="0" />
    <corner x="10,8" y="-10,8" z="0" />
    <corner x="10,8" y="0" z="0" />
    <corner x="0" y="0" z="0" />
  </marker>
 </multimarker>
```

CODICE 4.1: Esempio di file xml per i marker di ALVAR

Come si può intuire dalla struttura del file xml sono supportati i multi-marker. Più marker, dalla struttura simile a quello mostrato in precedenza, possono essere assemblati in modo da migliorare il riconoscimento e ridurre gli errori.

Alcuni test effettuati hanno dimostrato che il giusto compromesso tra dimensioni complessive del marker e facilità di riconoscimento si ottiene con dei multi-marker 4x4.

Il tool di Goblin genera automaticamente le informazioni riguardanti il marker; per farlo sfrutta degli elementi precaricati di geometria nota (come l'esempio mostrato in precedenza Fig. 4.1) e si limita ad assemblare le informazioni in un file png che rappresenta il multi-marker da stampare e un file xml che ne contiene la descrizione e che verrà utilizzato in seguito per inizializzare il traking.

Un altro tool che può rivelarsi utile è lo SceneGraphDisplay, questo tool verrà utilizzato più che altro a scopo di debug, è costituito da una libreria che se aggiunta al progetto consente di visualizzare in un form la struttura gerarchica della scena renderizzata, questo aspetto verrà illustrato più nel dettaglio quando mostrerò la struttura della scene in Goblin.

Oltre a quelli elencati esiste anche il tool StereoCameraCalibration, che viene utilizzato per calibrare particolati tipologie di videocamere hardware che consento la visione a 360°.

4.3 Struttura di un progetto Goblin

La struttura di un progetto Goblin deriva direttamente da quella di XNA; una nuova soluzione consiste di due progetti, uno sarà il progetto principale nel quale andrà inserita tutta la logica dell'applicazione, l'altro progetto funge da contenitore per tutte le risorse quali oggetti grafici 3d, file audio ecc..

La classe di avvio del progetto estende *Microsoft.Xna.Framework.Game* da cui segue l'implementazione dei metodi

Initialize() Come lascia intuire il nome qui va inserita tutta la logica da eseguire in fase di inizializzazione, generalmente in questa funzione non vengono inizializzati gli oggetti grafici. La funzione Inizialize esegue l'override dalla classe padre, deve quindi richiamare il metodo base dopo l'inserimtento della nostra logica di inizializzazione

LoadContent() Viene richiamata solo una volta ad ogni avvio, qui va eseguito il caricamento di tutti i contenuti da rendere disponibili, tra le altre cose qui vengono caricate le texture degli oggetti 3d, definiti nuovi oggetti a partire dalle primitive ecc..

UnloadContent() Esattamente l'inverso di LoadContent, anche questa funzione viene richiamata solo una volta e subito prima della chiusura dell'applicazione, qui vengono liberate tutte le risorse caricate in LoadContent.

Update() Viene richiamata in un ciclo continuo alternata a Draw(), qui viene inserita la logica di aggiornamento delle applicazioni come ad esempio la rilevazione degli imput.

Draw() Viene richiamata in un ciclo continuo alternata ad Update(), qui viene ridisegnata la scena.

Viene inoltre dichiarato un oggetto, il *GraphicsDeviceManager* che è la classe che permette l'interfacciamento con l'hardware grafico All'interno del costruttore va definita la posizione della cartella Content, di default alla creazione di un nuovo progetto viene assegnata automaticamente alla sotto-cartella *"Content"* della directory del progetto.

Lo scheletro di un nuovo progetto risulterà:

```
public class Game1 : Microsoft.Xna.Framework.Game
{
  GraphicsDeviceManager graphics;
  public Game1()
  {
    graphics = new GraphicsDeviceManager(this);
    Content.RootDirectory = "Content";
  }
  protected override void Initialize()
  {
    base.Initialize();
  }
```

```
  protected override void LoadContent(){}
  protected override void UnloadContent(){}
  protected override void Update(GameTime gameTime)
  {
    base.Update(gameTime);
  }
  protected override void Draw(GameTime gameTime)
  {
    GraphicsDevice.Clear(Color.CornflowerBlue);
    base.Draw(gameTime);
  }
}
```

CODICE 4.2: Scheletro di un'applicazione GoblinXNA

4.4 Struttura di una Scena

La struttura di una applicazione Goblin è ad albero (Figura 4.2), la scena è l'albero dell'applicazione che è costituito da nodi di diversa natura.

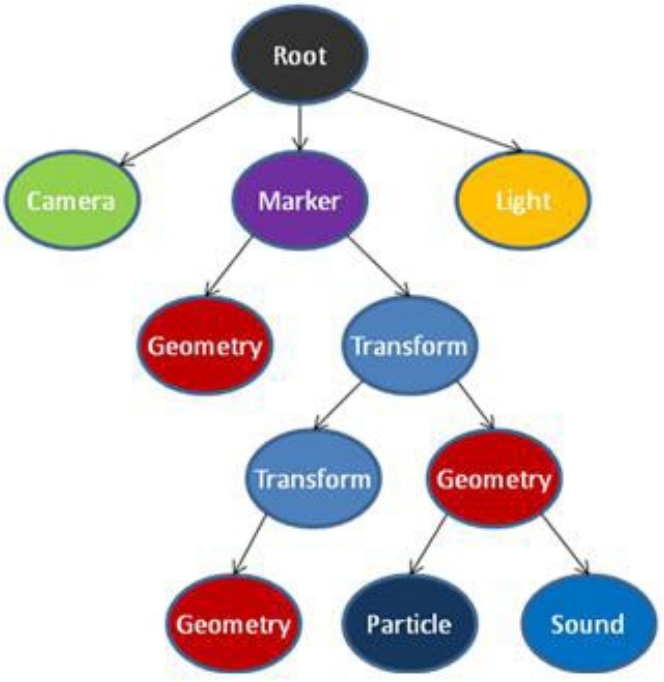

FIGURA 4.2: Struttura della scena in GoblinXNA

I nodi una volta inizializzati possono essere aggiunti alla scena con la sintassi

```
Scene.RootNode.AddChild(Node);
```

In maniera analoga è possibile associare i nodi tra di loro per formare la struttura

```
Node.AddChild(Node);
```

Attualmente il framework supporta 10 tipi di nodi:

Geometry Questo tipo di nodo contiene un modello geometrico da renderizzare sulla scena. Può contenere modelli 3d caricati da file con estensione .x e .fbx o modelli generati dinamicamente come ad esempio un cilindro, un cubo, una sfera ecc..

Goblin offre la possibilità di impostare il materiale degli oggetti generati definendone il colore, le texture e altre informazioni che garantiscono grandi possibilità di personalizzazione.

A i nodi di tipo Geometry è inoltre possibile assegnare proprietà fisiche che verranno utilizzate nelle simulazioni.

Transform I nodi Transform permettono di applicare delle trasformazioni a tutti i nodi figli. Questo permette di applicare traslazioni, rotazioni e trasformazioni di scala a tutti i nodi figli del transform node. Le trasformazioni possono essere applicate singolarmente richiamando gli appositi metodi, in questo caso il nodo ci occupa di calcolare la matrice di trasformazione risultante, oppure si può impostare direttamente la matrice di trasformazione.

Light I nodi Light contengono sorgenti di luce (Oggetti LightSource del framework) utilizzati per illuminare i modelli.

Le sorgenti di luce si possono essere di tre tipi:

Luci Direzionali è una luce orientata la cui sorgente si assume essere infinitamente lontana, si può pensare alla luce solare come a una sorgente di luce direzionale

Punto di luce è un punto con una posizione specifica che irradia luce in ogni direzione. L'intensità della luce emessa diminuisce all'aumentare della distanza

SpotLight - Riflettore è una sorgente luminosa che possiede una posizione, una direzione e un tronco di cono che ne identifica la diffusione.

I nodi Light possono essere definiti locali nel qual caso avranno effetto solo sui nodi fratelli e figli, o globali e avranno effetto sull'intera scena

Camera i nodi Camera definiscono la posizione del punto di vista della scena, nel caso della realtà aumentata il punto di vista coincide con quello della sorgente video.

Particle Contengono uno o più effetti particellari come le fiamme, fumo, esplosioni ecc...

Marker I nodi marker hanno un funzionamento molto simile ai transform node con la differenza che le trasformazioni vengono applicate in maniera automatica in relazione alla posizione di un marker

Questi nodi sono fondamentali nelle applicazioni di realtà aumentata

Sound Contiene informazioni riguardo sorgenti sonore 3D come la posizionee la velocità

Switch Viene utilizzato per visualizzare solo uno dei figli

LOD(Level Of Detail) è usato per selezionare un solo modello da renderizzare da una lista di modelli in cui ognuno è assunto avere un differente livello di dettaglio questo nodo consente anche di abilitare la selezione automatica del livello di dettaglio in base alla distanza dell'oggetto in modo da massimizzare le prestazioni

Tracker è molto simile al nodo marker, ma viene utilizzato per dispositivi hardware che offrono 6 gradi di libertà, come gli HDM. Da specifiche sono supportati Vuzix iWear VR920 (Figura 4.3) e il Vuzix Wrap 920 (Figura 4.4).

FIGURA 4.3: Vuzix iWear VR920

FIGURA 4.4: Vuzix Wrap 920

4.5 Struttura dell'applicazione

L'obiettivo principale nella realizzazione dell'applicativo di esempio è quello di mostrare come sia possibile interagire con oggetti 3D in realtà aumentata attraverso l'utilizzo di GoblinXNA. Inoltre si è voluto mostrare come fosse possibile effettuare il riconoscimento di più marker in contemporanea e come fosse possibile far interagire oggetti legati a marker differenti. Secondariamente si è cercato di rendere l'applicazione "dinamica" nel senso che oggetti e interazioni possono essere aggiunti a run-time con un meccanismo che verrà illustrato in seguito.

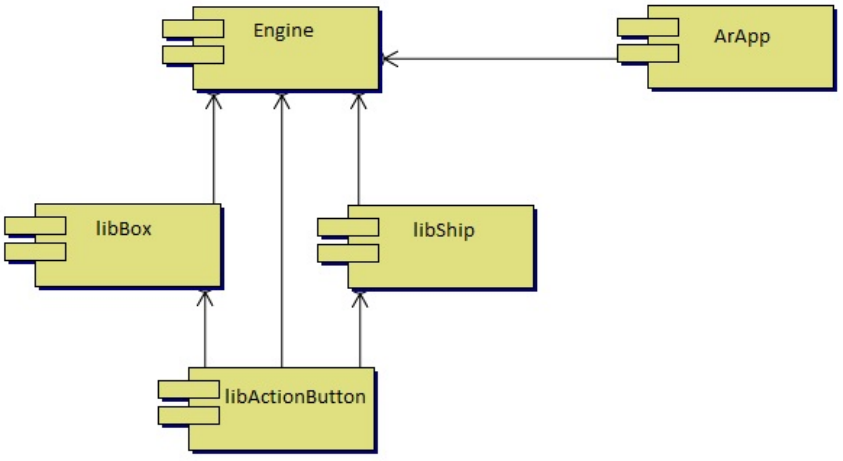

FIGURA 4.5: Diagramma dei componenti

L'applicazione che è divisa in 5 progetti (Vedi diagramma in figura 4.5) così composti:

ArApp É il progetto principale, da qui viene inizializzato l'hardware, gestiti gli aggiornamenti degli oggetti e gestita l'interfaccia utente

Engine Progetto che contiene le interfacce e le classi base per l'implementazione di oggetti da utilizzare all'interno del progetto ArApp

libShip Una libreria che rappresenta un'implementazione di Engine, visualizza un'astronave che può essere comandata da tastiera

libBox Altra implementazione di Engine, visualizza un cubo che ruota seguendo la posizione del mouse

libActionButton Altra implementazione di Engine, visualizza un pulsante che può essere premuto con il mouse, inoltre presenta 2 interazioni, al click del pulsante, se presente, l'astronave viene parcheggiata e il cubo viene riportato nella posizione originale.

La struttura così articolata permette di illustrare vari aspetti dell'interazione tra oggetti in GoblinXNA in maniera indipendente l'uno dall'altro.

4.5.1 ArApp

Il progetto ArApp è il progetto di avvio. La prima cosa che viene fatta all'avvio del progetto, dopo l'inizializzazione delle variabili, è la creazione di un nodo di luce globale che viene agganciato alla root della scena, successivamente si impostano la videocamera e il tracker:

```
captureDevice = new DirectShowCapture();
captureDevice.InitVideoCapture( 0,
                FrameRate._30Hz,
                Resolution._800x600,
                GoblinXNA.Device.Capture.ImageFormat.B8G8R8A8_32,
                false);
scene.AddVideoCaptureDevice(captureDevice);
```

CODICE 4.3: Inizializzazione della sorgente video

Viene inizializzata la `captureDevice`, si hanno due opzioni per l'inizializzazione, `DirectShowCapture()` e `DirectShowCapture2()` la prima richiama delle api di basso livello estremamente efficienti, la seconda è una versione più datata e meno efficiente ma che supporta un maggior numero di dispositivi. Successivamente viene inizializzata la sorgente video, è possibile definire il framerate, la risoluzione, il formato di colore e un valore booleano

che indica se l'acquisizione deve essere in scala di grigi, un'errata configurazione comporta il crash dell'applicazione in fase di avvio. Sono state testate 3 differenti videocamere, una integrata nel portatile e due esterne, in tutti e tre i casi non sono stati riscontrate difficoltà di configurazione previa installazione dei driver.

```
tracker = new ALVARMarkerTracker();
tracker.MaxMarkerError = 0.02f;
tracker.InitTracker(captureDevice.Width, captureDevice.Height, "calib.xml", 30f);
scene.MarkerTracker = tracker;
```

CODICE 4.4: Inizializzazione del tracker

Il tracker può essere inizializzato da qualunque classe che implementi l'interfaccia `IMarkerTracker`, alla versione attuale (4.1) Goblin supporta solo il tracker ALVAR mediante la classe `ALVARMarkerTracker` che viene utilizzata in questo progetto. É possibile impostare l'errore massimo nel riconoscimento del tracker attraverso la property `MaxMarkerError`. La funzione di inizializzazione `InitTracker` accetta in ingresso: la dimensione dell'input video (larghezza e altezza), un file di calibrazione, che è quello ottenuto dal tool messo a disposizione da Goblin(Sez. 4.2.1) e un valore di tipo `Float` che rappresenta la dimensione che verrà associata al tracker, questo valore verrà utilizzato in seguito per dimensionare gli oggetti renderizzati.

Il caricamento degli oggetti e delle interazioni avviene a run-time, all'interno della cartella di progetto sono presenti due sottocartelle `xml` e `library`, all'avvio viene inizializzata la classe `EngineLoader` che si occupa di caricare gli oggetti all'interno dell'applicazione.

All'interno del metodo `LoadContent` viene richiamata la funzione `CreateObject` (Codice 4.5) che richiama l'`EngineLoader` che a sua volta esegue il controllo di tutti i file dll all'interno della sotto-cartella `library` e, se i file contengono classi istanziabili come `Object` o `Interaction` (definiti nel progetto `Engine`), ne esegue il caricamento. A i file dll è associato anche un file xml che contiene la definizione del marker associato Il caricamento dinamico è possibile grazie alla `Reflection` disponibile nel framework .NET.

```
private InteractionEngine mIntEngine;
private List<Tesi.Engine.Object> mObject;
private List<Tesi.Engine.Interaction> mInteraction;
private LibraryChecker check;
private EngineLoader mEL;

private void CreateObject()
{
  List<Tesi.Engine.Object> mTempObject = new List<Engine.Object>();
  List<Tesi.Engine.Interaction> mTempInteraction = new List<Interaction>();

  mTempObject.AddRange(mEL.ObjectList);
  mTempInteraction.AddRange(mEL.InteractionList);

  List<Tesi.Engine.Object> mObjectToAdd = mTempObject.Except(mObject,
                                          new ObjectComparer()).ToList();
  List<Tesi.Engine.Interaction> mInteractionToAdd = mTempInteraction.Except(mInteraction,
                                          new InteractionComparer()).ToList();

  mObject.AddRange(mObjectToAdd);
  mIntEngine.addAllObject(mObjectToAdd);
  mIntEngine.addAlInteraction(mInteractionToAdd);

  mIntEngine.start();

  foreach (Tesi.Engine.Object obj in mObjectToAdd)
  scene.RootNode.AddChild((MarkerNode)obj.getTransformNode());
}
```

CODICE 4.5: Caricamento degli oggetti

Gli oggetti caricati vengono salvati all'interno di due liste nella classe main, `mObject` e `mInteraction`, l'associazione tra gli oggetti e le interazioni viene risolta dall'`InteractionEngine` il quale controlla che tutti gli oggetti richiesti da ogni interazione siano già stati caricati prima di effettuare l'associazione.

All'interno del metodo `Update()` viene effettuato l'aggiornamento degli oggetti. Ogni oggetto è indipendente dagli altri ed espone un proprio metodo update pubblico, avendo salvato precedentemente tutti gli oggetti in una lista è sufficiente un ciclo `For Each` sulla lista che richiami il metodo per tutti gli oggetti.

Nel metodo update si trova anche la logica di gestione degli eventi a livello di applicazione (ogni oggetto gestirà i propri eventi in maniera interna). La gestione della tastiera avviene utilizzando il tipo `KeyboardState` che rappresenta uno stato della tastiera, la classe statica messa a disposizione

da Goblin `Keyboard` espone il metodo `GetState()` che ritorna lo stato attuale della tastiera, in questo modo se si desidera scatenare un evento alla pressione di un tasto è sufficiente aggiornare una variabile di classe con lo stato della tastiera ad ogni update, se il pulsante X nello stato precedente non era premuto mentre nello stato attuale lo è si scatena l'evento, nel Listato seguente viene mostrato un esempio applicativo:

```
private KeyboardState oldState;
protected override void Update(GameTime gameTime)
{
  KeyboardState currState = Keyboard.GetState();

  //Controlla lo stato del tasto T
  if (oldState.IsKeyUp(Keys.T) && currState.IsKeyDown(Keys.T))
  {
    Azione(); //Eseguo una qualche azione
  }
  oldState = currState; //Aggiorno lo stato della tastiera
  scene.Update(gameTime.ElapsedGameTime, gameTime.IsRunningSlowly, this.IsActive);
}
```

CODICE 4.6: Gestione degli eventi della tastiera

Caricamento Dinamico Il caricamento degli oggetti avviene sfruttando una libreria esterna, `ThoughtWorks.QRCode.dll` che espone dei metodi per la gestione dei QRCode (Sezione 1.3), l'idea è che insieme al marker su cui generare l'oggetto venga fornito anche un qr-code contenente le informazioni per scaricare la dll da caricare (Figura 4.6).

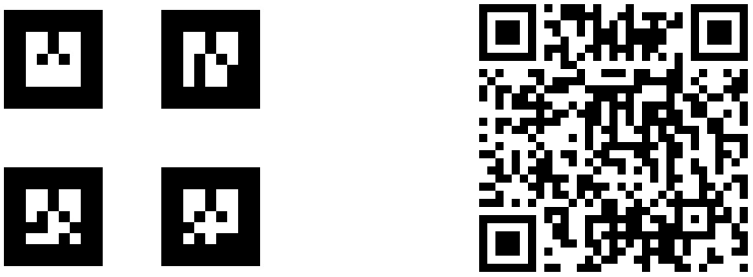

FIGURA 4.6: Sistema di riconoscimento per il caricamento dinamico in ArApp

Nell'applicazione il QR-Code contiene semplicemente un percorso relativo all'interno del file-system, ma nulla vieta di estendere il concetto introducendo un web-services in maniera analoga a quelli utilizzati da Wikitude e Layar (Sez. 2.3)

Come illustrato in precedenza (Sezione 1.3) i qr-code non sono adatti al riconoscimento in real-time in un flusso video, per questo motivo in ArApp il riconoscimento viene avviato alla pressione del tasto "T", il riconoscimento prosegue effettuando un tentativo ogni 0.5 secondi per un totale di 10 secondi, se non viene rilevato alcun qr-code il riconoscimento viene disabilitato, in caso contrario viene richiamato un metodo della classe `LibraryChecker` che si occupa di importare il file dll ed eventualmente il relativo xml con le informazioni di tracciamento e posizionarli nelle apposite sotto-cartelle. Infine viene richiamato l'`EngineLoader` che ricarica le librerie.

4.5.2 Engine

Il progetto Engine contiene tutte le classi astratte necessarie per implementare gli oggetti e le interazioni in modo che possano essere caricate in ArApp. Engine contiene 3 classi fondamentali (Vedi diagramma UML in Figura 4.7):

EngineElement É una classe astratta che implementa solo una property `Name` utilizzata per identificare univocamente ogni oggetto all'interno del sistema tramite il proprio nome.

Object Fornisce la base per l'implementazione degli oggetti all'interno di ArApp. Una funzionalità introdotta è quella del "debug-mode". Il debug di un'applciazione 3D senza punti di riferimento può risultare difficoltoso, `Object` espone il metodo `protected void addToDebug(Node node)` che permette a tutte le classi che ereditano da `Object` di definire quali nodi debbano essere soggetti alla modalità di debug.

La modalità debug non fa altro che visualizzare gli assi associati al sistema di riferimento di un nodo, richiamando una classe interna vengono costruiti gli assi x,y e z utilizzando le primitive messe a disposizione da Goblin

salvate in un nodo che viene agganciato al nodo fornito come parametro alla funzione `addToDebug`.

Il debug è attivato dalla pressione del tasto "D" che viene gestito in maniera analoga a quanto visto in precedenza (Codice 4.6).

Interaction La classe `Interaction` è piuttosto semplice, espone 2 metodi astratti da implementare nelle classi figle, `public abstract void start(Object[] mObj)` che accetta in ingresso un array di `Object` che rappresentano gli oggetti che intervengono nell'interazione e`public abstract string[] getObjectsName()` che deve tornare i nomi, che si ricorda essere univoci, degli oggetti che la classe fa interagire. In seguito verranno presentate alcune implementazioni.

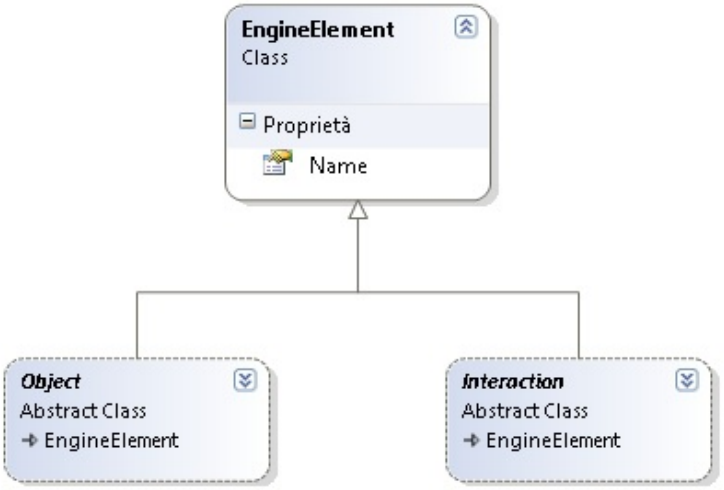

FIGURA 4.7: Diagramma delle classi del progetto Engine

4.5.3 libBox

Questo progetto contiene la prima vera e propria implementazione della classe Engine, visualizza a schermo un cubo ovviamente renderizzato in sovrapposizione a uno specifico marker(Figura 4.8), che può essere ruotato tramite l'utilizzo del mouse.

La gestione del mouse in Goblin avviene in maniera analoga a quella della tastiera, i metodi `KeyboardState` e `Keyboard` hanno la loro versione per il mouse in `MouseState` e `Mouse`. Gli oggetti di tipo `MouseState` contengono la posizione del mouse, in coordinate bi-dimensionali relative alla finestra dell'applicazione e lo stato dei pulsanti.

```
protected override void update()
{
  MouseState currentState = Mouse.GetState();
  if (previus.LeftButton == ButtonState.Pressed)
  {
    rotate(currentState.X - previus.X, currentState.Y - previus.Y);
  }
  previus = currentState;
}

private void rotate(float dX, float dY)
{
  float tollerance = 0.5f;
  float mX = dX * tollerance;
  float mY = dY * tollerance;
  boxTransNode.Rotation *= Quaternion.CreateFromAxisAngle(new Vector3(0, 1, 0),
  MathHelper.ToRadians(mX));
  boxTransNode.Rotation *= Quaternion.CreateFromAxisAngle(new Vector3(0, 0, 1),
  MathHelper.ToRadians(mY));
}
```

CODICE 4.7: Gestione rotazione del cubo in libBox

Va fatto notare che la libreria Goblin espone diversi metodi che agevolano la gestione delle trasformazioni, il metodo `Quaternion.CreateFromAxisAngle` crea una rotazione rappresentata come la rotazione di un certo angolo rispetto a una direzione, il valore di rotazione viene impostato sulla proprietà `Rotation` del nodo. Nel caso specifico di `libBox` è stata definita una funzione `rotate(float dX, float dY)` che esegue una rotazione lungo gli assi x e y del cubo, `dX` e `dY` rappresentano la variazione di coordinate operate con il mouse quando è premuto il tasto sinistro, per concatenare le due rotazioni si effettua una moltiplicazione dei due oggetti `Quaternion`, uno che rappresenta la rotazione su x e l'altro la rotazione su y e poiché le modifiche sono incrementali moltiplico il nuovo valore di rotazione con quello già salvato nel nodo. Goblin mette a disposizione la classe `MathHelper` che espone dei metodi utili nella gestione della geometria tre-dimensionale come ad esempio la conversione degli angoli in radianti utilizzata nel codice sovrastante.

Infine la classe `libBox` espone un metodo pubblico `Reset()` che riporta il cubo alla posizione di partenza, in seguito verrà mostrato come questo metodo venga utilizzato per realizzare delle interazioni tra oggetti.

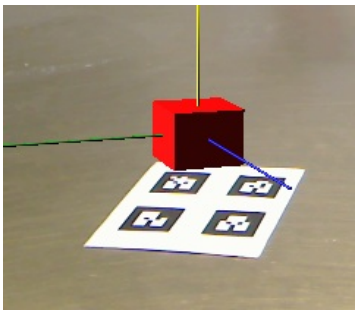

FIGURA 4.8: Visualizzazione dell'oggetto generato dal progetto libBox

4.5.4 libShip

Questo progetto è quello che rappresenta la gestione più complessa delle interazioni tra quelle che verranno mostrate. L'oggetto generato consiste in un'astronave, caricata da un modello 3d (Figura 4.9) che può essere controllata tramite la tastiera, l'utente può interagire con la nave facendola muovere in avanti (freccia su), facendola ruotare a destra (freccia destra) o a sinistra(freccia sinistra) e facendola alzare (A) e abbassare (Z).

La classe effettua l'override del metodo `CreateObjects()` ereditato dalla classe padre, all'interno del quale avviene il caricamento dei nodi:

- Inizialmente si crea un MarkerNode che effettua il tracking del marker relativo al progetto.
- Successivamente viene caricato il modello della nave presente negli assets del progetto e creato un nuovo nodo di tipo geometry al quale si associa il modello caricato.
- Dato che il modello scelto racchiude al suo interno la definizione dei materiali il loro utilizzo viene impostato nel modello.
- Utilizzando il valore `MarkerSize` fornito in fase di inizializzazione è possibile ridimensionare la nave per renderla proporzionale alla dimensione del marker.
- Si impostano la posizione e la rotazione iniziale e si inizializza la classe `Movement` che gestirà i movimenti
- Infine viene costruita la struttura ad albero dell'oggetto associando i nodi tra loro.

Di seguito viene illustrato il codice relativo alla creazione dell'oggetto "Ship"

```
protected override void CreateObjects()
{
  markerNode = new MarkerNode(this.MarkerTracker, MARKER_FILE);

  // Carico il modello della nave
  shipModel = (Model)new ModelLoader().Load("", MODEL_ASSET_NAME);

  // Creo un geometry node per il modello caricato
  shipNode = new GeometryNode("Ship");
  shipNode.Model = shipModel;

  // Impongo di utilizzare i materiali definiti nel modello
  ((Model)shipNode.Model).UseInternalMaterials = true;

  // Scalo il modello
  dimension = Vector3Helper.GetDimensions(shipModel.MinimumBoundingBox);
  shipTransNode = new TransformNode();
  float scale = MarkerSize * 0.7f / Math.Max(dimension.X, dimension.Z);
  shipTransNode.Scale = new Vector3(scale);

  //Imposto angolazione e posizione iniziali
  shipTransNode.Rotation = Quaternion.CreateFromAxisAngle(new Vector3(0, 0, 1),
  MathHelper.ToRadians(270))//
  * Quaternion.CreateFromAxisAngle(new Vector3(1, 0, 0),
  MathHelper.ToRadians(90));
  shipNode.Physics.Pickable = true;


  superTransNode = new TransformNode();

  shipTransNode.AddChild(shipNode);
  superTransNode.AddChild(shipTransNode);
  markerNode.AddChild(superTransNode);
  //Inizializzo la classe Movement
  mMov = new Movement(ref superTransNode);

  //Aggiungo al debug la nave e il marker
  addToDebug(superTransNode);
  addToDebug(markerNode);
}
```

CODICE 4.8: Metodo CreateObject() della classe Ship

La classe Ship espone anche il metodo `public void parcheggia()` che avvia il pilota automatico e riporta la nave alla posizione iniziale, questo metodo verrà utilizzato nella creazione di eventi personalizzati all'interno del progetto libActionButton.

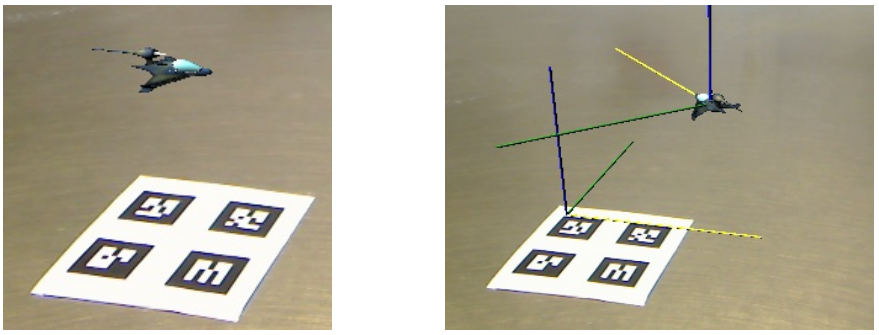

FIGURA 4.9: Rendering dell'oggetto Ship con e senza DebugMode attivo

Classe Movement

La gestione dei movimenti è demandata alla classe esterna `Movement`, la classe espone i metodi che consentono di eseguire movimenti atomici in una direzione:

```
public void goOn()
{
  mPosition.X += velocity * (float)Math.Cos(MathHelper.ToRadians(Rotation));
  mPosition.Y += velocity * (float)Math.Sin(MathHelper.ToRadians(Rotation));
  updatePosition = true;
}
```

CODICE 4.9: Metodo goOn() della classe Movement

Il metodo `goOn()` effettua un movimento atomico in avanti rispetto alla direzione dell'oggetto, nella variabile privata `Rotation` si trova l'angolo di rotazione, in gradi sessagesimali, rispetto all'asse z, risulta quindi che l'avanzamento dell'oggetto lungo gli assi x e y può essere facilmente ricavato dalla trigonometria. La variabile `velocity` è di tipo `Float` e contiene la dimensione della distanza atomica, poiché l'aggiornamento della posizione avviene ad ogni update dell'applicazione si ha che la distanza atomica rappresenta anche la velocità dell'oggetto, da cui il nome della variabile.

```
public void goUp()
{
  mPosition.Z += upVelocity;
  updatePosition = true;
}
public void goDown()
{
  if (mPosition.Z > 0)
  {
    mPosition.Z -= upVelocity;
    updatePosition = true;
  }
}
```

CODICE 4.10: Metodi goUp() e goDown() della classe Movement

Il funzionamento di questi metodi è analogo al precedente, con la differenza che il movimento è assunto essere uni-assiale nel senso che l'oggetto si muove nel piano x-y e l'asse z fornisce solo l'altezza, non è possibile quindi muovere l'oggetto lungo i piani x-z e y-z se non combinando le azioni delle funzioni `goOn()` e `goUp()` o `goDown()`. Anche in questo caso la variabile `upVelocity` rappresenta la velocità di movimento ascensionale e discensionale dell'oggetto.

```
public void turnRigth(){
  Rotation -= rotationVelocity;
  updateDirection = true;
}
public void turnLeft(){
  Rotation += rotationVelocity;
  updateDirection = true;
}
```

CODICE 4.11: Metodi turnLeft() e turnRight() della classe Movement

Questi due metodi modificano l'orientamento dell'oggetto nel piano x-y, la variabile `Rotation` contiene angolo di rotazione rispetto l'angolo di partenza, `rotationVelocity` indica la velocità di rotazione in gradi per update.

Nei metodi appena esposti sono presenti anche due valori booleani `updateDirection` e `updatePosition` che indicano le trasformazioni da applicare al prossimo update dell'oggetto.

Un altro importante metodo esposto dalla classe `Movement` è `public bool moveTo(Vector3 vect)` che esegue un movimento atomico in direzione di un punto specificato da un vettore richiamando i metodi esposti precedentemente. In seguito questo metodo verrà utilizzato per implementare un'azione di movimento automatica.

4.5.5 libActionButton

Questo è l'ultimo dei progetti realizzati per questa trattazione. Realizza, utilizzando le primitive messe a disposizione da Goblin un pulsante che se premuto scatena degli eventi.

La definizione del pulsante fa uso di due primitive, un cilindro e un toro entrambe fornite da Goblin (Figura 4.11). Per verificare la pressione del pulsante viene richiamata la funzione `bool isClick(MouseState state)`,

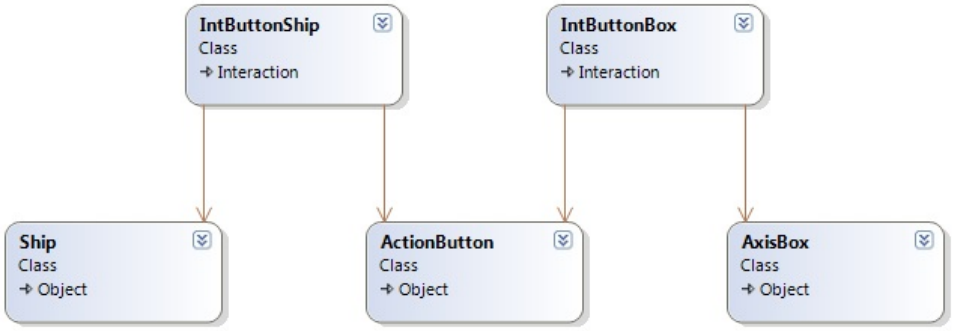

FIGURA 4.10: Diagramma delle classi del progetto libActionButton

descritta di seguito, che ritorna `True` se il pulsante è stato premuto, `False` altrimenti.

```
private bool isClick(MouseState state){
  if (state.LeftButton == ButtonState.Pressed)
  {
    //Proietto il click del mouse su due assi
    Vector3 nearSource = new Vector3(state.X, state.Y, 0);
    Vector3 farSource = new Vector3(state.X, state.Y, 1);
    //Moltiplico la matrice del punto di vista per la matrice di trasformazione del MarkerNode
    Matrix viewMatrix = markerNode.WorldTransformation * State.ViewMatrix;
    //Traspongo le coordinate del mouse nel sistema di riferimento del MarkerNode
    Vector3 nearPoint = this.Graphics.GraphicsDevice.Viewport.Unproject(nearSource,
                    State.ProjectionMatrix, viewMatrix, Matrix.Identity);
    Vector3 farPoint = this.Graphics.GraphicsDevice.Viewport.Unproject(farSource,
                    State.ProjectionMatrix, viewMatrix, Matrix.Identity);
    //Sfruttando il PhysicsEngine Newton ottengo la lista di oggetti selezionati
    List<PickedObject> pickedObjects = ((NewtonPhysics)this.GoblinScene.PhysicsEngine)
            .PickRayCast(nearPoint, farPoint);
    if (pickedObjects.Count > 0){
      pickedObjects.Sort();
      //Se e' stato cliccato il pulsante ritorno true
      string clickedName = ((GeometryNode)pickedObjects[0].PickedPhysicsObject.Container).Name;
      if (clickedName.Equals(cylinderNode.Name) || clickedName.Equals(torusNode.Name))
        return true;
    }
  }
  return false;
}
```

CODICE 4.12: Funzione isClick() della classe Button

Il progetto contiene anche due classi che estendono `Interaction` del progetto Engine, queste realizzano due interazioni tra oggetti (Vedi diagramma UML in Figura 4.10):

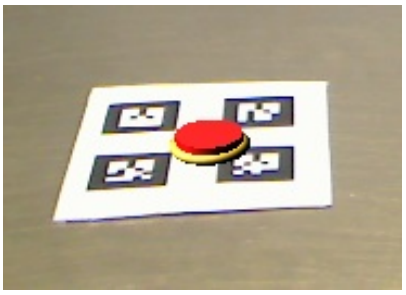

FIGURA 4.11: Pulsante generato nel progetto libActionButton

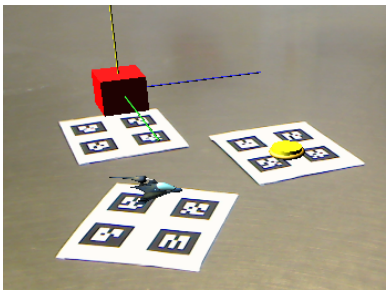

FIGURA 4.12: Risultato finale dell'applicazione

IntButtonBox Come lascia intuire il nome questa classe definisce un'interazione tra l'oggetto Button e l'oggetto Box, nello specifico alla pressione del pulsante viene richiamato il metodo `reset()` della classe Box che azzera la rotazione riportando il cubo nella posizione iniziale (sez. 4.5.3)

IntButtonShip Questa classe inizializza un'interazione tra l'oggetto Button e l'oggetto Ship, alla pressione del pulsante viene lanciato il metodo `parcheggia()` della classe Ship avvia il pilota automatico parcheggiando la nave al punto di partenza (sez. 4.5.4)

Il codice delle interazioni risulta abbastanza semplice e non richiede ulteriori spiegazioni, rimando al codice sorgente per approfondimenti.

In figura 4.12 è possibile vedere il risultato finale del rendering degli oggetti.

Conclusioni

La realtà aumentata rappresenta oggi un settore in forte sviluppo. La potenza di calcolo disponibile in dispositivi dalle dimensioni contenute quali smartphone e tablet consente l'elaborazione di immagini in real-time con algoritmi sempre più esosi e permettono di creare animazioni impensabili fino a qualche anno fa.

Come mostrato nel corso della trattazione il settore è molto in fermento e ogni giorno nascono nuovi framework che offrono nuove potenzialità agli sviluppatori, sono coperte tutte le principali piattaforme anche se risulta evidente l'orientamento verso il settore mobile che rappresenta senza dubbio la frontiera della realtà aumentata.

FIGURA 4.13: Prototipo dei Google Glass di prossima uscita

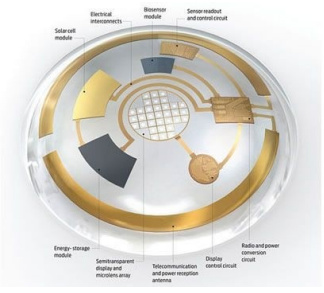

FIGURA 4.14: Progetto di retina-display con celle solari

L'evoluzione futura della realtà aumentata vedrà probabilmente il ritorno degli HDM, ad oggi il mercato attende con trepidazione l'uscita dei "Google Glass" (Figura 4.13) che dovrebbero rappresentare l'evoluzione dei classici HDM; a differenza degli esempi mostrati prima (vedi Figura 4.4 e Figura 4.3) i Glass sfruttano un prisma all'altezza dell'occhio destro che proietta l'immagine, generata da un micro-computer presente nell'asticella degli occhiali, direttamente sulla retina, questo permette di avere una notevole

luminosità delle immagini generate e riduce notevolmente l'utilizzo della batteria; oltre alla sovrapposizione di immagini in realtà aumentata i glass supportano l'interazione tramite riconoscimento vocale, dispongono di videocamera e cuffie integrate che permetteranno di sviluppare la realtà aumentata anche sul fronte sonoro.

Mentre fino ad oggi l'utilizzo degli "occhiali" per la realtà aumentata era riservato ad un pubblico di nicchia, con i glass e con le alternative che verranno introdotte dagli altri produttori, la realtà aumentata sarà alla portata di tutti e come insegna la storia dell'informatica quando un prodotto giunge alla portate delle masse questo subisce evoluzioni inaspettate, si pensi ad esempio all'evoluzione subita dai computer dalla nascita del primo pc commerciale.

Volendosi sporgere più avanti si possono analizzare alcuni degli studi attualmente in corso mirati alla realizzazione di "retina display", lenti a contatto per la realtà aumentata, un recente studio del MIT [45] illustra come sia possibile montare dei led su delle lenti a contatto realizzate con un materiale speciale sviluppato appositamente che unisce grafene e nanofili d'argento. Un altro interessante studio al riguardo è quello svolto dal professor Babak A. Parviz dell'università di Washington che in un articolo pubblicato nel 2009 [46] illustra come sia possibile realizzare lenti a contatto computerizzate che mostrano informazioni con la realtà aumentata e allo stesso tempo si ricaricano col sole (Figura 4.14).

In definitiva si può affermare che la realtà aumentata rappresenta oggi uno dei settori di punta del mercato hi-tech e che la sua evoluzione sia tutt'altro che conclusa, esistono interessanti studi tutti mirati alla miniaturizzazione dei componenti necessari ad un sistema AR i cui sviluppi sono tutt'altro che imprevedibili, d'altronde, solo 20 anni fa, chi avrebbe previsto che oggi avremmo avuto a disposizione smartphone accessibili a tutti e con una potenza di calcolo molto superiore a quella utilizzata dal progetto apollo per mandare un equipaggio sulla luna?

Bibliografia

[1] Ivan E Sutherland. A head-mounted three dimensional display. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, pages 757–764. ACM, 1968.

[2] Thomas P Caudell and David W Mizell. Augmented reality: An application of heads-up display technology to manual manufacturing processes. In *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, volume 2, pages 659–669. IEEE, 1992.

[3] Paul Milgram and Fumio Kishino. A taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 77 (12):1321–1329, 1994.

[4] 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 (1):75–86, 2002.

[5] N. Navab, J. Traub, T. Sielhorst, M. Feuerstein, and C. Bichlmeier. Action- and workflow-driven augmented reality for computer-aided medical procedures. *IEEE Computer Graphics and Applications*, 27 (5):10–14, September/October 2007.

[6] Layar official website, May 2013. URL http://www.layar.com/tools/.

[7] Wikitude official website, May 2013. URL http://www.wikitude.com.

[8] Metaio sdks official website, May 2013. URL https://research.cc.gatech.edu/uart/.

[9] Opencv official website, May 2013. URL http://www.opencv.org/.

[10] Bazar official website, May 2013. URL http://cvlab.epfl.ch/software/bazar/index.php.

[11] Artoolkit official website, May 2013. URL http://www.artoolworks.com/.

[12] Armedia official website, May 2013. URL http://www.inglobetechnologies.com/.

[13] Andar official website, May 2013. URL http://code.google.com/p/andar/.

[14] Uart official website, May 2013. URL https://research.cc.gatech.edu/uart/.

[15] Linceovr official website, May 2013. URL http://www.seac02.it/.

[16] osgart official website, May 2013. URL http://osgart.org/.

[17] Winar official website, May 2013. URL http://r2m.nus.edu.sg/cos/o.x?c=/r2m/license_product&ptid=5730&func=viewProd&pid=22.

[18] Slartoolkit official website, May 2013. URL http://slartoolkit.codeplex.com/.

[19] Flartoolkit official website, May 2013. URL http://www.libspark.org/wiki/saqoosha/FLARToolKit/en.

[20] Alvar official website, May 2013. URL http://www.vtt.fi/multimedia/alvar.html.

[21] Openspace3d official website, May 2013. URL http://www.openspace3d.com/.

[22] Sstt official website, May 2013. URL http://technotecture.com/augmentedreality.

[23] Vuforia official website, May 2013. URL http://developer.qualcomm.com/dev/augmented-reality.

[24] Goblin xna official website, May 2013. URL http://goblinxna.codeplex.com/.

[25] D'fusion official website, May 2013. URL http://www.t-immersion.com/.

[26] Popcode official website, May 2013. URL http://www.popcode.info/.

[27] Zappar official website, May 2013. URL http://www.zappar.com/.

[28] Arlab official website, May 2013. URL http://www.arlab.com/.

[29] Aurasma official website, May 2013. URL http://www.aurasma.com/.

[30] Google goggles official website, May 2013. URL http://www.google.com/mobile/goggles/.

[31] In2ar official website, May 2013. URL http://www.in2ar.com/.

[32] Ar23d official website, May 2013. URL http://www.ar23d.com/.

[33] Aurasma official website, May 2013. URL http://www.cortexica.com/.

[34] Obvious engine official website, May 2013. URL http://obviousengine.com/.

[35] Mixare official website, May 2013. URL http://www.mixare.org/.

[36] Nyartoolkit official website, May 2013. URL http://nyatla.jp/nyartoolkit/wp/.

[37] 3dar official website, May 2013. URL http://3dar.us/.

[38] Panicar official website, May 2013. URL http://www.dopanic.com/solutions/panic_ar.html.

[39] Robocortex official website, May 2013. URL http://www.robocortex.com/.

[40] Aruco official website, May 2013. URL http://www.uco.es/investiga/grupos/ava/node/26.

[41] Awila official website, May 2013. URL http://www.awila.co.uk.

[42] String official website, May 2013. URL http://www.poweredbystring.com/y.

[43] Xloudia official website, May 2013. URL http://xloudia.com/.

[44] Pointcloud official website, May 2013. URL http://www.pointcloud.io/.

[45] High-performance, transparent, and stretchable electrodes using graphene–metal nanowire hybrid structures, May 2013. URL http://pubs.acs.org/doi/abs/10.1021/nl401070p.

[46] Augmented reality in a contact lens, September 2009. URL http://spectrum.ieee.org/biomedical/bionics/augmented-reality-in-a-contact-lens.