

UNIVERSITÀ DEGLI STUDI DI PADOVA

Facoltà di Scienze Matematiche Fisiche e Naturali

CORSO DI LAUREA MAGISTRALE IN INFORMATICA

Play With Eyes: uno strumento
per lo screening visivo in età
pediatrica

Relatore:

Prof.ssa Ombretta Gaggi

Laureando:

Alberto De Bortoli
Matr. 603482

Anno Accademico 2010/2011

*Ai miei genitori
per tutto il supporto che mi hanno
dato nel periodo universitario*



Università degli Studi di Padova
Facoltà di Scienze Matematiche Fisiche e Naturali
CORSO DI LAUREA MAGISTRALE IN INFORMATICA

**Play With Eyes: uno strumento per lo
screening visivo in età pediatrica**

Relatore: **Prof.ssa Ombretta Gaggi**

Laureando: **Alberto De Bortoli**

Anno Accademico 2010/2011

Indice

1	Introduzione	5
2	Stato dell'arte	9
1	I serious game	10
1.1	Definizione di <i>serious game</i>	10
1.1.1	Finalità	11
1.2	I <i>serious game</i> in campo medico	12
1.2.1	Scopo riabilitativo e diagnostico	13
2	Cenni clinici sull'Ambliopia	15
2.1	Cause dell'ambliopia	16
2.2	Diagnosi	17
2.3	Terapia	18
3	Lo screening visivo	19
3.1	Iniziative in ambito nazionale	19
4	Related works	21
4.1	Applicativi software commerciali	21
4.2	Applicativi software in ambito accademico	24
3	Scopo del progetto	25
1	Idea realizzativa del progetto	26
1.1	Barriere di ingresso per l'utilizzo del servizio	27
2	Il sistema	29
2.1	Attori	29
2.2	Componenti	29
3	Analisi dei requisiti	32
3.1	Requisiti non funzionali	32
3.2	Requisiti funzionali	33

3.3	Considerazioni sui requisiti	34
4	Background tecnologico	37
1	Sviluppo cross-platform	38
1.1	PhoneGap	38
1.2	Titanium framework	38
2	Considerazioni	41
2.1	Problemi inerenti allo sviluppo cross-platform	41
2.2	Motivazioni della scelta di sviluppo nativo	43
3	Sviluppo nativo	46
3.1	Ambiente di sviluppo	46
3.1.1	Xcode	46
3.1.2	Interface Builder	47
3.1.3	Simulator	47
3.1.4	Tool secondari	48
3.2	Linguaggio	50
3.2.1	Sintassi di invocazione dei metodi	50
3.2.2	Creazione di oggetti	51
3.2.3	Gestione della memoria	52
3.2.4	Protocolli e delegazione	54
3.3	Cocoa Framework	55
3.3.1	UIKit	56
3.3.2	Pattern MVC in Cocoa	57
5	Sviluppo del progetto “<i>Play With Eyes</i>”	59
1	Dettagli implementativi	60
1.1	Architettura	60
1.2	Strumenti ortottici utilizzati	60
1.3	Utilizzo di immagini tratte da cartoni animati	62
1.4	Tipologie di esercizi implementati	63
1.5	Modalità di gioco e interazione	64
1.6	Calcolo grandezza ottotipi	68
1.6.1	Basi teoriche	68
1.6.2	Implementazione	70
1.7	Connessione Wireless	72
1.7.1	Apple Bonjour Service	72

1.7.2	Connessione tramite GameKit	74
1.8	Comunicazione con display	76
1.9	Database	77
1.9.1	Struttura dei file JSON	79
2	Sviluppo agile	83
2.1	I principi della metodologia agile	84
3	Tool utilizzati	86
3.1	Hardware e licenza di sviluppo	86
3.2	Sistema di versionamento	86
3.3	Componenti aggiuntive	87
3.3.1	Libreria SBJSON	87
4	Problemi riscontrati	89
4.1	Comportamenti inattesi	89
4.2	Impossibilità di debugging	90
4.3	Dimensione display	91
5	Utilizzo del sistema	92
5.1	Utilizzo di iPadSight	92
5.2	Utilizzo di iPodSight	94
6	Test dello strumento	97
1	Test in fase di sviluppo	97
1.1	Test del codice	97
2	Test in scuola materna	99
2.1	Risultati ottenuti	101
2.2	Considerazioni sulla parte <i>serious</i>	101
2.3	Considerazioni sulla parte <i>game</i>	105
2.4	Tempi di esecuzione	105
2.5	Casi particolari	106
7	Conclusioni	109
1	Risultati ottenuti	110
2	Sviluppi futuri	111
A	SeGAH 2011 Short Paper	113
B	Glossario	119

Capitolo 1

Introduzione

Uno dei problemi principali dei medici che devono interagire con bambini è riuscire a ottenere la loro attenzione. I pediatri devono, in primis, poter contare sulla collaborazione del piccolo paziente per poter effettuare la diagnosi, la quale può essere di svariata natura. I medici oculisti sono una categoria che hanno una marcata necessità di catturare l'attenzione e la collaborazione del bambino per poter effettuare una diagnosi accurata dei suoi occhi. Non è realistico poter dare per assodata la collaborazione del paziente, in particolar modo se questo è molto giovane; i bambini infatti non possono essere trattati come piccoli adulti.

Si consideri il caso di una visita oculistica durante la quale il medico deve, tra i vari test, poter capire quanto il paziente è capace di vedere specifiche figure ad una determinata distanza per valutarne l'acutezza visiva. Ciò che viene chiesto al bambino è di indicare quali figure ottotipiche riesce a vedere durante la visita medica. Questo compito è spesso noioso e in pochi minuti il bambino è portato a rispondere frettolosamente ed in maniera erronea, in quanto non più motivato a collaborare.

Sono noti disturbi visivi che devono essere curati prima del raggiungimento di una certa età; se ciò non avviene tempestivamente essi possono evolvere in patologie croniche sulle quali non è più possibile intervenire. Il caso più noto è quello dell'ambliopia (più comunemente nota con il nome di “sindrome dell'occhio pigro”), ovvero un'alterazione della visione dello spazio che viene a manifestarsi inizialmente durante i primi anni di vita. È una condizione in cui la funzione visiva di un occhio è ridotta o assente senza che vi siano stati

dei danni oculari organici; ne è affetto circa il 2% di tutta la popolazione e il 4-5% dei bambini [33]. Uno dei problemi principali dell’ambliopia è la diagnosi in quanto il paziente non è in grado in genere di capire che è affetto da un disturbo visivo e non è consci del fatto che sta usando maggiormente un solo occhio. I trattamenti per la cura dell’ambliopia devono iniziare il più presto possibile appena il disturbo è individuato e prima che la vista sia completamente sviluppata, cosa che avviene a circa 10 anni. Cercare di curare l’ambliopia dopo lo sviluppo completo della vista può non portare risultati [20]. Alcuni studi [37] [33] mostrano che i migliori risultati si ottengono con una diagnosi precoce. Soggetti curati per ambliopia in età tra gli 8 e i 37 mesi hanno possibilità molto buone di riuscire ad essere corretti. L’evolversi naturale del deficit visivo dovuto ad ambliopia non prevede, infatti, alcun miglioramento spontaneo e, se non trattata, essa in età adulta incide negativamente sulla qualità di vita, aumenta i costi sociali e riduce la possibilità di accesso a varie carriere lavorative. Da ciò si comprende che una patologia del genere limita a vita la funzionalità visiva, mentre un banale controllo oculistico in età infantile avrebbe potuto evitare il problema. Con tali dati alla mano, l’esigenza di poter effettuare screening visivi in soggetti di giovane età, specialmente nelle scuole materne, risulta quindi essere ben motivata.

Questa tesi presenta “Play With Eyes”, un prodotto software con il quale è possibile testare l’acutezza visiva dei bambini attraverso un gioco. Utilizzando il paradigma dei *serious game* i bambini possono giocare in maniera divertente senza necessariamente sapere di essere sottoposti a un test con finalità differenti dal puro gioco [28]. L’architettura del sistema ha un comportamento server/client ed è composto da un dispositivo iPad®, un dispositivo iPod Touch® o iPhone®, e un monitor. Il bambino esaminato risponde sullo smartphone agli esercizi del test in base a ciò che riesce a vedere sul monitor posizionato alla distanza corretta. La scelta avviene nel modo più naturale possibile per un bambino, ovvero toccando l’interfaccia touch dello smartphone.

Per poter assegnare il termine “serious game” all’applicazione client è importante che il test assuma il più possibile le sembianze di un gioco, che mostri una complessità e un aspetto adatti per il target definito. Vengono

infatti utilizzate immagini di noti cartoni animati per catturare l'attenzione degli esaminandi, mostrate assieme a simboli ottotipici specifici per bambini piccoli. “Play With Eyes” permette di effettuare test non solo per valutare l’acuità visiva ma anche la visione dei colori.

Il progetto è nato con due finalità: essere utilizzato durante le visite in ambulatorio e per lo screening visivo scolastico senza la presenza di medici. Il secondo obiettivo è stato quello maggiormente trainante e ha visto lo svolgimento dei test nella scuola materna “Gianni Rodari” di Mogliano Veneto su un totale di 65 bambini. Tale screening è servito sia per testarne l’effettiva utilità nel riscontrare patologie visive, sia per valutarne le caratteristiche ludiche pensate per lo specifico target, con risultati positivi.

Il progetto ha visto la collaborazione del Dipartimento di Pediatria Salus Pueri [36] dell’Università degli Studi di Padova riscontrando interesse particolare da parte della Dott.ssa Luisa Pinello e della Dott.ssa Elisabetta Zannin.

Capitolo 2

Stato dell'arte

L'applicazione oggetto di questa tesi è sviluppata basandosi sul paradigma dei serious game. In questo capitolo viene introdotto tale concetto e vengono analizzate le modalità di utilizzo di questi giochi in ambiente medico.

Successivamente è mostrato uno studio sullo stato dell'arte riguardante i software attualmente in commercio, il cui scopo è simile al progetto presentato o in qualche modo accomunabile. Si analizzeranno tutte le tipologie di software riscontrate, da soluzioni professionali e commerciali ad applicativi gratuiti non professionali.

Infine sono affrontati l'argomento dello screening visivo e le relative iniziative che attualmente sono state intraprese in ambito nazionale.

1 I serious game

1.1 Definizione di *serious game*

Il serious game è una simulazione virtuale interattiva, talvolta con l'aspetto di un vero e proprio gioco. I serious game hanno scopi formativi, educativi, sociali e di marketing.

I primi serious game sono nati per scopi militari, puntando allo sviluppo da parte dell'utilizzatore di abilità e competenze da applicare nel mondo reale attraverso l'esercizio in un ambiente simulato e protetto: mediante la riproduzione di situazioni reali, i partecipanti devono raggiungere un obiettivo attraverso l'impiego di conoscenze specifiche e l'attuazione di strategie.

Il paradigma più generale a cui afferiscono i serious game è quello di proporre una situazione di gioco per nascondere le vere finalità della prova sottoposta o rendere più accattivante e divertente l'utilizzo del sistema. L'interesse per i serious game è cresciuto negli ultimi anni, tanto da vedere la nascita di siti ad essi dedicati [19] [13].

La grande varietà di obiettivi che i serious game si pongono permette loro di trovare applicazione in una vasta gamma di campi; titoli ben noti trovano già impiego, tra gli altri, in ambito militare (America's Army, gioco di guerra in modalità first-person shoot), nelle scuole (Sid Meier's Civilization, gioco di strategia che prevede la gestione di territori e la costruzione di città), nell'ambito governativo (Sim City, gioco di simulazione della creazione e gestione di società di persone virtuali) e in quello medico riabilitativo (Wii Fit, videogioco per l'allenamento fisico).

I serious game possono servire per reperire informazioni di diverso genere che vengono trattate in maniera automatica: un esempio è Google Image Labeler [18], un gioco che fa confrontare online due utenti per trovare parole comuni al fine di dare il tag ad una serie di immagini. In questo modo Google ha trovato un metodo intelligente per controllare che determinate parole chiave siano assegnate alle giuste immagini; tale operazione non è effettuabile senza l'intervento umano e serve per migliorare l'accuratezza del sistema informativo.

La varietà di impieghi dei serious game sta alla base della vertiginosa diffusione di questa tipologia di gioco e del rinnovato interesse accademico volto a fornire le risposte ai dubbi relativi a questo approccio didattico. Secondo

questo paradigma gli aspetti di “apprendimento” e “divertimento” diventano conciliabili.

1.1.1 Finalità

I serious game sono applicazioni videoludiche in grado di trasmettere al giocatore nozioni o messaggi durante l’atto di gioco, o di sviluppare le sue capacità, incrementandone la motivazione, facilitando l’apprendimento per mezzo del contesto significativo e piacevole in cui le informazioni vengono presentate. Sono giochi il cui fine ultimo non è l’intrattenimento fine a se stesso.

In ambito civile il training aziendale, le campagne di educazione e di sensibilizzazione, nonché le varie attività promozionali e campagne sociali, possono essere campi di applicazione del serious gaming. In contesti commerciali i serious game vengono solitamente utilizzati per simulare contesti di vendita telefonica o face-to-face, interazioni in occasione di colloqui o di riunioni e, in generale, tutte quelle situazioni in cui si renda necessaria un’esperienza diretta per assimilare contenuti e comportamenti. Per ciò che attiene agli usi militari, il serious game trova applicazione sia nell’addestramento del personale tecnico (quali operatori radar, personale di macchina a bordo delle navi, piloti di aerei ed elicotteri) sia nella simulazione di operazioni militari complesse come i cosiddetti War games.

Le informazioni e le sensazioni vissute rimangono fortemente impresse e permettono in questo modo al giocatore di affinare percezione, attenzione e memoria favorendo modifiche comportamentali attraverso il “learning by doing”, con il vantaggio però di agire in ambito protetto, quello, appunto, di un software di simulazione. Interiorizzare qualcosa che si è fatto di persona risulta molto più semplice rispetto all’apprendimento di contenuti veicolati durante lezioni frontali, apprendimento cosiddetto passivo. Rendendo la simulazione molto vicina alla realtà si riduce la paura del nuovo aumentando la fiducia dell’utente nel mettersi in gioco, e si accresce così l’esperienza pratica.

L’elemento ludico aumenta il coinvolgimento e permette di agire più spontaneamente, senza sentirsi giudicati. La possibilità di poter ripetere l’esercizio innumerevoli volte permette di giungere alla totale padronanza della dinamica “esplorata” conferendo maggiore serenità nell’utilizzo dello strumento e nell’affrontare la situazione reale.

1.2 I *serious game* in campo medico

Il benessere psico-fisico e la cura del corpo sono aspetti della vita quotidiana a cui le persone rivolgono sempre maggiore attenzione; per soddisfare le crescenti richieste di questo settore il mercato dei videogiochi si è adeguato progettando dei serious game in grado di allenare o sviluppare corpo e mente. Nascono quindi le prime startup che dei serious game per l'heathcare hanno fatto il loro business principale. È il caso ad esempio di Basis [32], società di recente formazione che promuove uno strumento hardware per il monitoring dell'attività fisica. Indossando questo strumento, dall'aspetto di un orologio, è possibile monitorare le calorie consumate, le ore di sonno, lo stress e l'attività fisica, portando il gioco ad un livello di coinvolgimento sociale tramite un social network che stimola l'utente a partecipare in maniera attiva e virtuosa. Il più famoso, gratuito e commerciale sistema Nike+ [29] è un ottimo esempio di serious game socialmente coinvolgente: gli utenti gareggiano tra di loro e in base ai risultati ottenuti su performance di resistenza e velocità di jogging acquisiscono punteggio.

I risultati in termini di incremento del benessere fisiologico e fisico sono provati, soprattutto se si gioca a concreti livelli di gioco e si seguono i percorsi consigliati dai personal trainer virtuali. Parlare di salute in generale risulta un po' vago, perché in un campo come questo ci si può riferire sia alla cura del corpo finalizzata all'estetica, sia alla cura finalizzata alla guarigione da patologie: i serious game non sono utilizzati solo per la cura effimera ma vengono utilizzati da molti medici e ospedali come coadiuvanti in certi trattamenti o per tranquillizzare le persone prima di un intervento chirurgico.

Uno studio sui serious game in campo medico [27] descrive le aree di applicazione di questo tipo di risorsa:

esercizi di fitness: si tratta di quella tipologia di videogioco che simula gli esercizi, attraverso dei sensori corregge eventuali posture sbagliate, e consiglia le tipologie di esercizio più adatte in base alle esigenze della persona. Solitamente questi giochi servono anche per motivare l'utente a seguire una dieta equilibrata e condurre una vita sana. Ne è un esempio il gioco WiiFit;

educazione alimentare: attraverso il gioco vengono spiegate quali sono le abitudini alimentari più corrette da tenere. Da diversi studi è emerso

che questa tipologia di gioco si è rivelata utile per correggere stili di vita alimentari sbagliati, ed è rivolta soprattutto a persone affette da obesità o da diabete. Ne è un esempio il sito myfoooddiary.com;

distrazione da terapie: alcuni serious game vengono utilizzati durante trattamenti terapeutici particolarmente fastidiosi per distrarre il paziente e per metterlo a proprio agio. Sono usati specialmente per aiutare i bambini malati a ridurre l'ansia e il malessere prima di un intervento chirurgico o durante terapie sgradevoli;

ricoveri e riabilitazioni: aiutano ad incrementare le capacità motorie in caso di processi riabilitativi o a spiegare ad un paziente come si svolgerà il proprio intervento chirurgico per renderlo più preparato;

simulazioni di interventi chirurgici: le simulazioni di intervento chirurgico dedicate ai medici hanno dimostrato che l'esperienza acquisita attraverso il gioco ha incrementato il successo degli interventi eseguiti in laparoscopia;

diagnosi e trattamento di malattie mentali: il gioco viene utilizzato per diagnosticare e trattare disturbi mentali come il deficit di attenzione, l'iperattività o il disturbo da stress post-traumatico;

funzioni cognitive: in commercio si trovano diversi giochi di questo tipo che attraverso quiz, rebus e indovinelli allenano la memoria e fanno sviluppare le proprie capacità analitiche e cognitive. Ne è un esempio il gioco "Who has the biggest brain?";

controllo: si tratta di quella tipologia di videogioco che grazie al monitoraggio delle funzioni vitali dà suggerimenti su come gestire al meglio le proprie reazioni e la propria emotività. Ne è un esempio il già citato strumento "Basis".

Considerata la vastità e la diversità di impieghi che trovano i serious game in questo ambito non c' è da stupirsi che gli stakeholders coinvolti in questo settore siano tra i più diversi, organizzazioni pubbliche e private, aziende, società di ricerca e sviluppo ecc.

1.2.1 Scopo riabilitativo e diagnostico

La riabilitazione medica è spesso complicata dalla presenza di strumenti inaccuracy. Essi sono spesso ingombranti e non adattivi, cosa che rende il loro

utilizzo scomodo e non sempre possibile. Il medico non ha sempre la possibilità di seguire costantemente il paziente, che si ritrova a dover eseguire degli esercizi senza che il fisioterapista possa controllarne l'esecuzione.

In [23], Kato esplora gli aspetti positivi nell'usare i videogiochi presenti in commercio per il benessere psicofisico e giochi sviluppati su misura per velocizzare la riabilitazione del paziente, che può essere fatta tramite giochi studiati e progettati appositamente. Un esempio di applicazione sviluppata su piattaforma Android, dal nome di “DroidGlove”, è diretta ai pazienti in cura fisioterapica per facilitare la riabilitazione dell'articolazione del polso [12]. L'applicazione sviluppata sfrutta un dispositivo mobile per venire incontro a questa problematica; il paziente può sfruttare questo strumento come oggetto riabilitativo ed eseguire diversi tipi di esercizi, consistenti in una serie di movimenti di rotazione dell'articolazione, tenendolo tra le mani. Le misurazioni inaccurate vengono così perfezionate tramite l'utilizzo dei sensori elettronici del dispositivo e la velocità di esecuzione, così come la precisione dei movimenti, vengono tenute sotto controllo dall'apparecchio. Un feedback immediato relativo ai miglioramenti o peggioramenti del paziente-giocatore viene generato, regolato da un sistema di punteggio.

Anche lo screening (affrontato nella sezione 3) può appoggiarsi all'utilizzo di serious game per risultare più accattivamente. Progetti come “*Screening per il controllo delle capacità visive nelle scuole dell'Isola Bergamasca*” [26] potrebbero fare utilizzo di giochi per ottenere una migliore collaborazione del paziente.

2 Cenni clinici sull'Ambliopia

L'ambliopia, in oftalmologia, è un'alterazione della visione dello spazio che viene a manifestarsi inizialmente durante i primi anni di vita. Il termine deriva dal greco e, più esattamente, da ops (che significa visione) e amblyos (che significa ottusa, pigra). Si tratta di una funzione visiva inferiore alla norma in uno o raramente in entrambi gli occhi, non giustificabile con patologie che siano evidenziabili dall'oculista. Comunemente l'occhio ambliope viene detto occhio pigro per indicare che non è un occhio che presenta una specifica patologia, ma che la sua funzione non si è completamente sviluppata. L'effetto principale è un comune deficit dell'acutezza visiva e si considera ambliope un occhio che abbia almeno una differenza di 3/10 rispetto all'altro, oppure un visus inferiore ai 3/10. Ne è affetto circa il 2% di tutta la popolazione e il 4-5% dei bambini; essa è considerata una delle prime cause di deficit visivo nei giovani sotto i 20 anni. Si definisce funzionale un'ambliopia in cui le strutture dell'occhio appaiono sane e funzionali. Infatti l'anomalia è legata a un non corretto sviluppo visivo e neuronale; usualmente, quando non si precisa il tipo di ambliopia, ci si riferisce proprio a questo. Infatti, il cervello disattiva le immagini che arrivano da un occhio perché, ad esempio, non riesce a combinarle con quelle provenienti dall'altro occhio, per cui non si viene a creare l'effetto della tridimensionalità. Si definisce organica, invece, quando la probabile causa è legata ad un'alterazione delle vie ottiche, ovverosia una lesione del globo oculare o delle vie visive cerebrali. La prima forma può essere trattata, l'ambliopia organica è invece spesso irrisolvibile.

L'ambliopia si può verificare solamente durante il periodo dello sviluppo dell'apparato visivo. Le stesse cause dell'ambliopia, se intervengono quando il sistema visivo è maturato, non sono in grado di determinarla. Queste cause sono pertanto efficaci nei primi 7-8 anni di vita; quanto più precocemente si attua una di queste cause tanto più difficile è il recupero visivo. Perciò quanto più precocemente si inizia la terapia tanto più rapidamente essa sarà efficace.

Si riportano alcune considerazioni del Dott. Savino D'Amelio, Primario Ospedale Oftalmico di Torino, Divisione di Oftalmologia Infantile e Unità Operativa di Ortottica e Strabismo e presidente della Associazione Italiana Pre-

venzione Ambliopia (AIPAM) [34].

« [...] L’ambliopia è una patologia curabile ma solo se diagnosticata e trattata nell’età infantile; si deve tenere presente che, indipendentemente dal tipo di ambliopia da cui è affetto, il paziente non guarirà da solo. Nella pratica quotidiana di chi opera nell’oftalmologia infantile e tratta un elevato numero di pazienti affetti da ambliopia, nulla è più triste del momento in cui debba dire ai genitori di un ragazzo che non c’è più niente da fare. Sono numerose le occasioni in cui si ha la riprova di ciò che anni di esperienza e di innumerevoli studi sull’argomento ci continuano a confermare: l’unica soluzione è la tempestività della diagnosi e del trattamento. [...]»

Gli individui affetti da ambliopia, se profonda, essendo monocoli funzionali, possono avere una vita difficile. Immaginate per esempio quanto la mancanza di una buona visione binoculare possa ostacolare il soggetto; a cominciare dall’età evolutiva, quando impedisce di realizzare un corretto sviluppo della localizzazione spaziale, della stereopsi, passando per l’età dell’apprendimento quando può limitare almeno in parte il rendimento scolastico, per arrivare all’età adulta, in cui tale handicap riduce le occasioni di inserimento lavorativo e la possibilità di ottenere patenti o brevetti che amplierebbero le occasioni d’impiego dell’individuo. [...] Numerosi studi al riguardo e precise esperienze in altri paesi, un esempio fra tutti la Svezia, hanno confermato che l’onere finanziario da dedicare ad un’intensa campagna d’informazione e prevenzione è di gran lunga inferiore a quello sostenuto per far fronte alle ricadute sociali di questa patologia se trascurata. Nell’ 800 von Graefe, uno dei padri dell’oftalmologia, definì l’ambliopia come la condizione in cui l’osservatore non vede nulla ed il paziente vede poco. Questa affermazione vuole enfatizzare la necessità che tutti i bambini siano sottoposti a visita da parte di personale esperto indipendentemente dai sintomi manifestati che tuttavia, quando presenti, non sono da sottovalutare »

2.1 Cause dell’ambliopia

L’ambliopia può essere causata da qualsiasi fattore che riduca l’input (stimolo) visivo in un solo occhio, se lo stesso fenomeno agisce su entrambi gli occhi non si manifesta l’ambliopia. Questi fattori possono essere divisi in tre categorie:

- Anisometropia (differenza di difetto refrattivo tra i due occhi).
- Strabismo (qualsiasi deviazione che non permetta una visione binoculare, cioè che le immagini dei due occhi non possano essere fuse in una unica e come tale percepite a livello cerebrale).
- Deprivazione sensoriale (qualsiasi ostacolo anatomico o patologia oculare che non permetta un input visivo normale in uno dei due occhi: per esempio cataratta congenita, ptosi palpebrale, opacità corneali, opacità vitreali, ecc.).

2.2 Diagnosi

La diagnosi precoce di ambliopia è fondamentale allo scopo di ottenere un recupero funzionale completo. Quale differenza di acuità visiva può essere considerata necessaria per fare diagnosi di ambliopia? Dal punto di vista pratico è sufficiente una differenza di due decimi di visus, dopo aver eseguito una accurata correzione del difetto refrattivo. In assenza di uno strumento opportuno, fare diagnosi di ambliopia sulla base di una valutazione del visus è possibile solo negli individui collaboranti, difficilmente nei bambini al di sotto dei 3-4 anni di età. Pertanto è necessario utilizzare segni indiretti per sospettare una ambliopia:

- Assenza di alternanza nello strabismo (il bambino fissa sempre con un occhio, quello non ambliope, e l'altro è deviato)
- Reazione del paziente alla copertura di un occhio (coprendo l'occhio non ambliope si scatena una reazione, per esempio il pianto, mentre non si verifica alcuna reazione alla copertura dell'occhio ambliope)
- Valutazione della refrazione (con schiasscopia in cicloplegia e con auto-refrattometria), la presenza di una anisometropia deve fare sospettare una ambliopia.

Se ne deduce che nei pazienti ambliopi è sempre importante ed interessante confrontare il visus ottenuto con simboli presentati isolati (un carattere per riga) e il visus ottenuto con più simboli vicini sulla stessa riga. In questi pazienti ambliopi si verifica il fenomeno dell'affollamento, cioè il visus è migliore con simboli presentati isolati rispetto a quello con più simboli per riga. Si verifica una incapacità di discriminare simboli ammassati strettamente. Un altro fenomeno che si verifica nell'ambliopia è la differenza di acuità visiva tra vicino e lontano, la prima è migliore rispetto alla seconda.

2.3 Terapia

La terapia dell'ambliopia si basa su due principali strade, entrambe necessarie per ottenere un sufficiente recupero visivo.

1. Individuazione e rimozione delle cause dell'ambliopia:

- fattori di depravazione visiva (cataratta congenita, ptosi palpebrale, ecc.);
- vizi di rifrazione e anisometropie;
- strabismo.

2. Trattamento riabilitativo con metodi che possono essere anche associati.

- Riduzione dell'input visivo dell'occhio fissante (occhio buono):

Atropinina: annebbiamento della visione specialmente per vicino instillando Atropina collirio;

Penalizzazione ottica: prescrizione di lenti per occhiali che determinano una sfuocatura dell'occhio fissante (occhio buono);

Settorizzazione: copertura di parte del campo visivo dell'occhio fissante con pellicole adesive traslucide sugli occhiali;

Occlusione: inibizione dell'occhio fissante applicando direttamente sulla pelle una benda opaca adesiva per un tempo variabile nella giornata in base all'entità dell'ambliopia.

- Stimolazione diretta dell'occhio ambliopico Questa stimolazione dell'occhio ambliopico può essere attuata con:

- Terapia pleottica secondo Cuppers o secondo Bangerter (attualmente poco usata);
- Apparati come il CAM o il GPG.

3 Lo screening visivo

Il termine screening è utilizzato in medicina per indicare una strategia di indagine diagnostica generalizzata, utilizzata per identificare una malattia in una popolazione standard, con un rischio medio di malattia che si reputa sufficientemente elevato da giustificare la spesa e lo sforzo per cercarla. A differenza dei test medici, le procedure dello screening prevedono che gli stessi test siano eseguiti a tappeto su tutta la popolazione.

Lo scopo dello screening è quello di identificare le malattie presenti in una comunità in una fase precoce, permettendo di ridurre i disagi derivati dalle malattie più diffuse e facilmente diagnosticabili. Tramite lo screening si intende diagnosticare una malattia prima di quando essa si manifesterebbe in assenza di esso. Senza lo screening la malattia potrebbe essere scoperta più tardivamente, una volta che i sintomi compaiono.

Le attrezzature mediche impiegate nei test di screening sono tipicamente differenti da quelle impiegate nei comuni test diagnostici; i test di screening vengono usati soltanto per indicare la possibilità o la probabilità di avere una malattia, una condizione lievemente patologica oppure una predisposizione. L'equipaggiamento medico viene usato per visite mediche accurate e specialistiche. Nonostante il test di screening possa sembrare poter portare solo dei vantaggi alla popolazione fornendo uno strumento di diagnosi precoce, in realtà nessun test è perfetto: problemi dovuti a risultati non corretti possono portare all'individuazione di falsi positivi o negativi.

3.1 Iniziative in ambito nazionale

In Italia sono nate alcune iniziative a scopi diagnostici per lo screening visivo, tutte circoscritte ad un'area geografica limitata e spesso patrocinate da enti privati. Tre province italiane che si sono distinte per sensibilità su questi temi sono Parma [15], Bergamo [26] e Belluno [6].

A Parma l'attività di screening ha avuto risultati più che soddisfacenti sin dall'inizio, ossia dal triennio "sperimentale" 1987-1990; l'équipe dello screening è composta da tre ortottisti-assistenti in oftalmologia, che eseguono il test di screening presso le Scuole Materne nei vari distretti della Provincia e

dal medico oculista che effettua le visite di secondo livello presso la Sezione di Oftalmologia.

A Bergamo ha riscosso successo lo “Screening per il controllo delle capacità visive nelle scuole dell’Isola Bergamasca” promossa da Lions Club San Pietro Isola. Essa è un’iniziativa gratuita per i bambini dell’ultimo anno delle scuole materne e del primo anno delle scuole elementari ed è circoscritta all’area di Bergamo Ovest. Ormai attiva dal 2004 può vantare di avere eseguito lo screening su migliaia di soggetti, per un numero di iscritti annuali sempre in crescita.

A Belluno l’attività di screening è mirata al riconoscimento dell’ambliopia. Un’ortottista dell’ULSS conduce un esame sui bambini delle scuole dell’infanzia del Cadore raccogliendo una serie di informazioni preliminari attraverso le quali verranno individuati i casi sospetti che saranno successivamente esaminati da un medico specialista dell’ospedale della Provincia.

Altri scenari prevedono uno screening visivo effettuato nelle scuole materne dalla figura del pediatra, soluzione tutt’altro che economica. L’attività di screening riveste un ruolo fondamentale nella tutela e nella prevenzione in campo sanitario; in Veneto ad esempio è demandato ai pediatri con un costo superiore ai 20€ a bambino come da dati della Camera dei Deputati [7]. Ciò viene fatto in modo non efficiente e a volte neppure efficace a detta degli stessi oculisti. Un’unificazione ed uniformità di procedure per arrivare alla stesura di un protocollo di screening visivo valido su scala nazionale ed accettato dalle autorità locali è quanto di più auspicabile.

4 Related works

4.1 Applicativi software commerciali

Esistono soluzioni software che permettono di effettuare test della vista sia per uso professionale che personale. Allo stato dell'arte le soluzioni professionali più famose reperite in rete, che possono essere adottate dai medici oculisti sono le seguenti:

- 20/20 Vision (Canela Software) [8]
- PVVAT - Precision Vision Visual Acuity Test (Precision Vision) [31]



Figura 2.1: Logo promozionale del prodotto software PVVAT

Il comportamento dei due software è molto simile: entrambi mostrano gli strumenti usati dagli oculisti (tabelle di Snellen, tavole di Ishihara, ottotipi di Lea, ecc.) su di un monitor che viene visto dal paziente. Lo schermo può essere il monitor del computer o un display esterno. Stando alla descrizione ufficiale dei prodotti, entrambi sono semplici sostituti di cartelli prestampati o del proiettore di dispositivi. Non sono focalizzati in maniera specifica nel testare pazienti in età pediatrica, sebbene sia possibile l'utilizzo degli ottotipi di Lea; inoltre queste soluzioni non si appoggiano al paradigma dei serious game per migliorare l'attenzione dei piccoli pazienti.

Negli ultimi anni la nascita del mercato di applicazioni mobile per smartphone ha visto l'apparizione di serious game per l'health care, ed alcune applicazioni permettono all'utente di testare la loro vista utilizzando direttamente il loro cellulare smartphone.

Le applicazioni presenti sul mercato App Store di Apple sono:

- Acuity (Intellicore)
- EyeChart HD (Dok LLC)
- Vistion Test (3 sides cube)
- FastAcuity Lite (KyberVision Consulting, RD)
- Vision (AppZap)
- Eyetest (Konrad Feiler)

mentre quelle presenti sul mercato Android Market di Google sono:

- Advanced Eye Charts (Ying Wen Techonologies)
- Advanced Vision Test (Kikapps)
- Mini Acuity (Hyrulean Productions)
- Visual Acuity Test (Advent Mobile Designs)

Tutte le sopracitate applicazioni per iPhone, iPod Touch, iPad o device Android, presentano pressoché le stesse feature senza differenze degne di nota.

Le feature implementate sono:

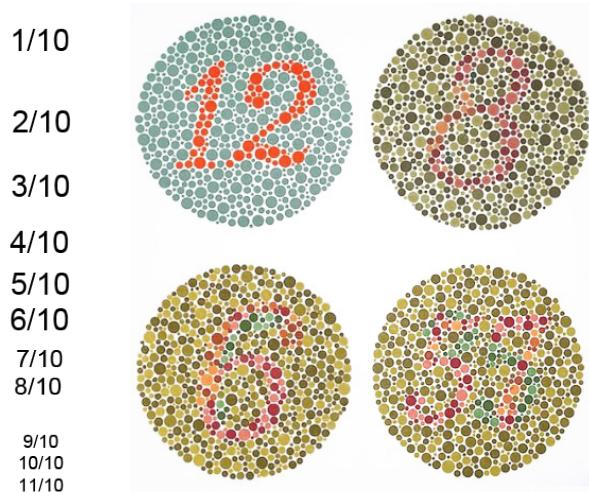
- test dell'acutezza visiva tramite tabella di Snellen
- test dei colori tramite tavola di Ishihara
- test dell'acutezza visiva tramite anelli di Landolt
- test dell'acutezza visiva tramite E orientate

La tabella di Snellen per l'acutezza visiva, gli anelli di Landolt e le tabelle di Ishihara sono strumento ortottici utilizzati dagli oculisti e mostrati in figura 2.2.

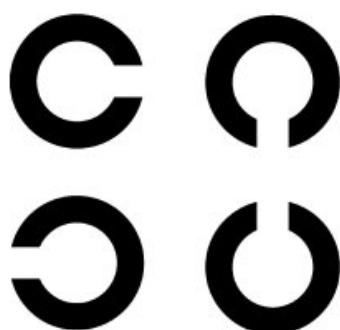
Applicazioni come queste sono spesso sviluppate in assenza di forte specifiche, non consentono niente più che semplici test auto somministrati senza implicazioni reali e non possono essere utilizzate in uno studio medico. Nessuna di queste applicazioni tiene in considerazione specifici requisiti accomunabili al paradigma dei serious game, mal si adattano ai bambini e non affronta il problema del mantenimento dell'attenzione di questi. Per questi motivi "Play With Eyes" vuole essere un tool necessario e innovativo.



(a) esempio di tabella di Snellen



(b) esempio di tavola di Ishihara



(c) esempio di anelli di Landolt



(d) esempio di E orientate

Figura 2.2: Alcuni strumenti ortottici

4.2 Applicativi software in ambito accademico

Tenere conto che gli utenti giocatori sono dei bambini ha un'importanza non trascurabile. Per rendere la simulazione accattivante e divertente, l'applicativo deve presentarsi sottoforma di gioco possibilmente corredando l'esperienza con caratteristiche che ne favoriscano il coinvolgimento ludico.

Molti articoli affrontano tale problema, [30] in particolare descrive un gioco che permette l'accesso all'informazione ai bambini in età prescolare. I bambini usano oggetti fisici (una bacchetta magica) per interrogare e navigare un database di immagini. Forlines et al [14] affrontano le differenze tra l'utilizzo del mouse e di interfacce touch, concludendo che le seconde, sebbene portino ad un peggioramento di performance nella velocità e nell'accuratezza, sono preferibili per la maggiore semplicità, la minore fatica che ne deriva e l'utilizzo della memoria spaziale.

In [22] viene presentato uno strumento che permette ai bambini di conoscere nuovi oggetti toccando oggetti fisici attraverso strumento appositamente progettato e con un'interfaccia RFID. Viene utilizzata la tecnologia RFID per l'identificazione degli oggetti e la comunicazione Bluetooth per trasmettere i dati al computer in cui il software del sistema è in esecuzione. Tre giochi sono stati implementati e consentono ai bambini di diverse età di beneficiare delle funzionalità del sistema incoraggiandoli nell'interazione.

Questi esempi rappresentano dei punti di partenza da tenere in considerazione.

Capitolo 3

Scopo del progetto

In questo capitolo viene descritta l'idea alla base del progetto con alcune specifiche emerse fin dalla fase di analisi iniziale. Vengono descritte le funzionalità basilari che il sistema deve presentare e viene fatta un'analisi dei requisiti necessaria per la chiarificazione di tali funzionalità.

1 Idea realizzativa del progetto

Nell'ultimo decennio in particolare lo sviluppo delle tecnologie informatiche ha permesso la ricerca in settori che concernono lo studio e la prevenzione di patologie di svariato tipo. Grazie all'evoluzione dei dispositivi mobili e a recenti tecnologie come RFID, Bluetooth, Wi-Fi, image recognition, augmented reality, accelerometro e giroscopio, è possibile sfruttare i device mobili, il cui uso tangibile porta alla creazione di servizi fino a pochi anni fa impensabili.

Per quanto descritto nei capitoli precedenti, nasce l'idea di un progetto di ricerca e sviluppo che vada a supportare, con gli strumenti e le tecnologie esposte, la prevenzione e lo screening precoce di una patologia come l'ambliopia per la quale al momento non esistono efficaci strumenti tecnologici. L'idea alla base è selezionare bambini con problemi visivi con un esame rapido, divertente, a basso costo, senza necessità di instillare colliri e sottoporli ad una visita medica mirata. L'esperienza ludica tramite serious game rende il bambino più partecipativo e permette di segnalare in tempo utile ai genitori l'eventuale difetto e attuare i protocolli necessari alla risoluzione di esso evitandone l'aggravamento. Il sistema ha quindi come scopo ultimo il reperimento di dati sull'apparato visivo dei bambini tramite un serious game.

Quando utilizzata per scopi di screening, l'applicazione deve mostrare un insieme di test preconfigurati per un'utilizzo immediato. Devono essere necessarie solo configurazioni minori con dati riguardanti i bambini; tali dati potranno essere inseriti prima di effettuare i test, mantenendo un database con le classi dei bambini già pronto per l'uso per permettere di testare un bambino dopo l'altro ottimizzando i tempi.

Permettere di effettuare i test in maniera veloce avendo già i dati sugli utenti caricati è voluto; per iniziare il test si vuole evitare di far attendere il bambino, il quale rischia di diventare impaziente durante l'attesa. L'acutezza visiva del bambino potrà essere quindi testata individualmente e il report derivante viene salvato in maniera persistente sul database. Tale report è indicativo della vista del bambino per quanto sia stato possibile rilevare.

Nel caso di utilizzo in laboratorio, l'applicazione rappresenta soltanto una

facilitazione per l'oculista per ottenere la cooperazione dai bambini. Più a lungo l'attenzione del bambino viene catturata e più a lungo il test può durare per avere maggiori informazioni su ciò che il bambino riesce a vedere portando a diagnosi più accurate.

L'acutezza visiva è testata anche tramite i simboli Lea [24], specifici per bambini in età prescolare per la loro semplicità: è facile infatti per i bambini riferirsi a loro anche verbalmente in maniera specifica. Non è necessario che abbiano appreso l'alfabeto perché non vi sono riferimenti a lettere; i simboli utilizzati sono quattro: una mela, una casa, un quadrato e un cerchio mostrati in figura 3.1.



Figura 3.1: Ottotipi di Lea: una mela, una casa, un quadrato, un cerchio

Quanto descritto finora si può ricondurre a uno schema a due livelli. Al primo livello avviene lo screening nelle scuole proposto con sistema che verrà sviluppato, mentre al secondo livello avviene la visita oftalmica specialistica se il soggetto risulta positivo allo screening. Gli utilizzatori per i quali il sistema verrà sviluppato e nei quali verrà testato saranno principalmente enti pubblici, in particolare asili, e studi oculistici. È da intendersi che l'utilizzo del sistema non sostituisce in nessun modo una visita oftalmica professionale ma fornisce dati che fungono da "cartina tornasole" per successive diagnosi.

1.1 Barriere di ingresso per l'utilizzo del servizio

Sono emerse fin da subito due criticità di cui si è dovuto tenere conto sia nella definizione dell'architettura tecnologica, sia nella fase di analisi e realizzazione degli strumenti di supporto ritenuti idonei. Tali criticità sono legate ad avere individuato come destinatario/utilizzatore degli strumenti gli enti pubblici, gli asili, le scuole d'infanzia, ecc. Dal punto di vista commerciale quindi le

barriere di ingresso per l'utilizzo del servizio presso gli enti interessati sono rappresentate da due fattori principali:

1. Il costo iniziale della tecnologia necessaria deve essere minimo, utilizzando il più possibile strumenti già in possesso degli utilizzatori onde evitare investimenti che possono limitare la possibilità di adozione dello strumento. In particolare, l'obiettivo iniziale era quello di utilizzare sistemi diffusi come personal computer e smartphone di diversi produttori mantenendo il più possibile la compatibilità con sistemi diversi.
2. Si deve tenere conto anche di limitate competenze tecnologiche del personale che configurerà e utilizzerà il sistema, che sarà più facilmente un educatore piuttosto di un operatore informatico.

Il primo punto rappresenta una barriera non indifferente; un'azione per ridurla è cercare di ammortizzare le spese prevedendo degli scenari di spesa derivati dalla situazione del mercato tecnologico. Nello sviluppo del prodotto la scelta della tecnologia hardware da utilizzare è stata fatta anche tenendo conto dei costi che gravano sugli utilizzatori finali.

Per cercare di mitigare la possibile riluttanza dell'utente finale nell'appoggiarsi al sistema è stato realizzato uno studio di usabilità mirato a fornire un'interfaccia di facile utilizzo per qualsiasi tipo di attore coinvolto nel sistema.

2 Il sistema

In questa sezione viene usata una notazione alfanumerica per tracciare componenti e attori. In figura 3.2 viene mostrata l'architettura pensata per il sistema.

2.1 Attori

Si prevede la presenza dei seguenti attori operanti nel sistema, quali:

A01 (utente configuratore): l'attore che ha il potere di modificare le impostazioni per permetterne un uso “ad hoc” all’utente giocatore. Nello scenario attuale si può pensare che tale attore sia il maestro della scuola che utilizza il prodotto o un medico oculista;

A02 (utente giocatore): l’attore che utilizza la parte di serious gaming del sistema per fornire dati che descrivono il comportamento e le capacità. Nello scenario attuale si può pensare che tali attori siano i soggetti dai quali occorre reperire informazioni, ovvero i bambini nelle scuole o i pazienti sottoposti a visite oculistiche;

A03 (utente valutatore): l’attore che, a seguito dei dati pervenuti dal sistema tramite la parte di serious gaming, analizza tali dati con conoscenze mediche ed eventualmente elabora tali dati con tecniche statistiche per catalogare ed estrarre informazioni rilevanti a scopi di catalogazione. Nello scenario attuale tale attore può coincidere con la figura del medico oculista o dello statista.

2.2 Componenti

L’architettura pensata per il sistema prevede la presenza di alcune componenti hardware:

HW01 (server): device tablet, l’utilizzatore è l’utente configuratore descritto di seguito;

HW02 (client): device smartphone, l’utilizzatore è l’utente giocatore descritto di seguito;

HW03 (display): componente collegata al server e dal quale riceve informazioni da presentare visivamente;

HW04 (cavo VGA): componente che collega il server al display.

Inizialmente si era ipotizzato l'utilizzo di un personal computer per il componente HW01; successivamente si è deciso per l'utilizzo di un tablet, in quanto un personal computer è di più difficile trasporto, le funzionalità sono superiori a quelle effettivamente necessarie e il costo è in genere superiore. Un tablet rappresenta inoltre uno strumento di nuova generazione; lo sviluppo di un'applicazione desktop sarebbe apparso non sfruttare le nuove tecnologie emergenti.

Viene prevista la presenza di due sole componenti software:

SW01 (Server): componente che gestisce i dati e configura il serious game, l'utilizzatore è l'utente configuratore descritto in seguito;

SW02 (Client): componente che presenta il serious game, l'utilizzatore è l'utente giocatore descritto in seguito.

Nell'applicazione mobile si prevede l'uso degli strumenti classici utilizzati dall'oftalmologo. Verranno usate delle figure semplici come simboli per ottotipi, in particolare quelli di Lea Hyvärinen [24] creati nel 1976: una mela, una cassetta, un quadrato e un cerchio, adatti a bambini molto piccoli. Questi simboli anche se sottendono una dimensione leggermente diversa tra di loro, sono stati studiati per presentare la medesima leggibilità e molti studi scientifici confermano la bontà di questi ottotipi. Gli ottotipi di Lea possono essere utilizzate con bambini a partire dai 3-4 anni e quindi risultano perfette per lo scopo.

Il sistema è stato progettato per permettere un tempo di gioco di 5-10 minuti per soggetto, sufficiente per testare l'acutezza visiva e la sua capacità di discernere i colori. Ogni sessione si svolge come segue:

1. Vengono proiettati i simboli ottotipici contornati di immagini tratte da cartoni animati (in maniera non fuorviante per il bambino) sul display (HW03);
2. Il bambino (A02), in base a ciò che è capace di vedere sullo schermo, sceglie la stessa figura riprodotta sul device mobile; ciò è possibile sfruttando le superfici touch e/o gli accelerometri presenti negli smartphone di recente produzione;

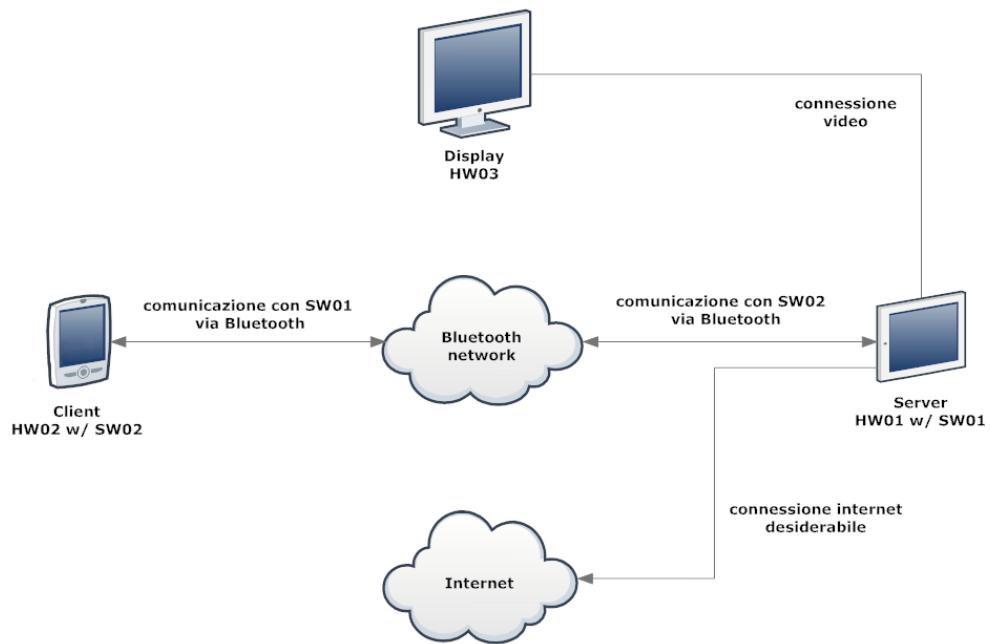


Figura 3.2: Architettura ipotizzata

3. Il server (SW01) riceve i dati e li salva in maniera persistente per analisi successive; la serie di output prodotta è ciò che nel primo livello di screening determina la positività o negatività al test.

3 Analisi dei requisiti

Le componenti descritte nella sezione precedente devono rispettare determinati requisiti. Per le componenti hardware tutti i requisiti si intendono essere non funzionali, per le componenti software invece si intendono essere funzionali in quanto è necessario fare assunzioni sui servizi che il sistema deve fornire e del comportamento a seguito di particolari input.

I requisiti sono a sua volta suddivisi in obbligatori, desiderabili e opzionali, utilizzando la seguente notazione in suffisso:

- “**o**” per indicare un requisito obbligatorio, irrinunciabile per il committente;
- “**d**” per indicare un requisito desiderabile, cioè non strettamente necessario, ma che genera un valore aggiunto riconoscibile;
- “**p**” per indicare un requisito opzionale, relativamente utile e contrattabile in seguito.

3.1 Requisiti non funzionali

Requisiti per la componente server hardware **HW01**:

RF_HW01_01o: HW01 deve presentare un sistema operativo sul quale poter installare applicazioni di terze parti sviluppate con un apposito SDK;

RF_HW01_02o: HW01 deve essere dotato di una scheda wireless Bluetooth;

RF_HW01_03o: HW01 deve essere dotato di qualche sorta di output video per il collegamento a un display esterno;

RF_HW01_04d: HW01 dovrebbe essere dotato in una scheda wireless Wi-Fi 802.11.

Requisiti per la componente client hardware **HW02**:

RF_HW02_01o: HW02 deve presentare un sistema operativo sul quale poter installare applicazioni di terze parti sviluppate con un apposito SDK;

RF_HW02_02o: HW02 deve essere dotato di una scheda wireless Bluetooth;

RF_HW02_03o: HW02 deve predisporre di una superficie touch capacitiva per un’interazione più semplice.

3.2 Requisiti funzionali

Vengono elencati di seguito i requisiti funzionali che gli applicativi devono rispettare. Requisiti per la componente server software **SW01**:

RF_SW01_01o: SW01 deve poter gestire la creazione di utenti, di classi di utenti, gestire la configurazione dei test e degli esercizi facenti parte di questi in maniera persistente;

RF_SW01_02o: SW01 deve adottare il protocollo wireless Bluetooth per comunicare con SW02;

RF_SW01_03o: la comunicazione tra SW01 a SW02 deve prevedere connessione e scambio dati riguardanti il serious game;

RF_SW01_04o: SW01 deve poter gestire un flusso video su display esterno;

RF_SW01_05o: SW01 deve poter memorizzare i dati ricevuti da SW02 in maniera persistente;

RF_SW01_06o: SW01 deve permettere la visualizzazione di ottotipi in grandezze diverse in base all'altezza dello schermo e alla distanza del bambini da esso;

RF_SW01_07d: SW01 dovrebbe permettere un meccanismo per l'invio dei risultati ottenuti tramite mail;

RF_SW01_08d: SW01 dovrebbe permettere un meccanismo per lo scambio di configurazioni (utenti e test) tramite computer desktop;

RF_SW01_09d: SW01 dovrebbe permettere un'utilizzo semplice ed intuitivo con interazioni di tipo touch.

Requisiti per la componente client software **SW02**:

RF_SW02_01o: SW02 deve adottare il protocollo wireless Bluetooth per comunicare con SW01;

RF_SW02_02o: la comunicazione tra SW01 a SW02 deve prevedere connessione e scambio dati riguardanti il serious game;

RF_SW02_03o: SW02 deve presentare un'interfaccia grafica pulita e di immediato utilizzo per un bambino;

RF_SW02_04d: SW02 dovrebbe sfruttare le gesture touch degli smartphone per rendere il serious game più coinvolgente.

3.3 Considerazioni sui requisiti

I requisiti RF_HW01_01o e RF_HW02_01o rappresentano la base di sistema e senza di essi non sarebbe possibile sviluppare nessun applicativo software. Allo stato dell'arte tutti i device smartphone e tablet di recente produzione presenti sul mercato portano con sè questa caratteristica.

I requisiti RF_SW01_02o, RF_SW02_02o, RF_HW01_02o e RF_HW02_02o permettono al sistema di lavorare in assenza di reti Wi-Fi, non necessariamente presenti in enti come scuole materne o studi oculistici. Inoltre, la pretesa di disponibilità di una rete Wi-Fi va oltre le reali necessità tecnologiche in quanto la semplice connessione Bluetooth è sufficiente per l'interazione tra le componenti HW01 e HW02 sia per la mole di dati scambiati sia per la distanza a cui tali componenti saranno situate. La specifica Bluetooth promette una connettività nel raggio di 10 metri, distanza più che accettabile prevedendo che in qualsiasi scenario i device saranno situati nella stessa stanza ad una distanza sicuramente non superiore ai 3 metri. Questi requisiti inoltre pongono minori restrizioni in termini di componentistica hardware richiesta.

Il requisito RF_HW01_04d vorrebbe la possibilità di connessione Wi-Fi per l'invio dei dati reperiti tramite mail secondo il requisito RF_SW01_07d, anch'esso desiderabile.

I requisiti RF_HW01_03o e RF_SW01_04o sono necessari per poter mostrare su un monitor esterno ciò che il bambino deve riuscire a vedere durante il test. Non tutti i device mobili disponibili sul mercato hanno questa caratteristica e anche in tale condizione non è scontata l'applicazione del requisito RF_SW01_04o.

Come detto, l'utente giocatore è un bambino che per interagire con il serious game ha necessariamente bisogno di un sistema di facile utilizzo con interazione tramite touch; per questo motivo il requisito RF_HW02_03o è obbligatorio. Ciò non rappresenta un ostacolo data la stragrande maggioranza di dispositivi touch screen presenti attualmente sul mercato. Stessa necessità concerne il server e il client con i requisiti desiderabili RF_SW01_09d e

RF_SW02_04d rispettivamente (in base all'utilizzo più o meno sapiente che si può fare dei meccanismi di touch disponibili).

Il requisito RF_SW01_06o è necessario in quanto non si può né si vuole fare alcuna assunzione sulla grandezza del display esterno o sulla distanza del bambino da esso. Questo aspetto è ciò che rende il sistema dinamico a differenza di tavole ottiche luminose o stampate che presentano una dimensione fissa.

I requisiti RF_SW01_07d e RF_SW01_08d prevedono l'invio di mail comprensive dei risultati sottoforma di file allegato e lo scambio di file di database e configurazione tramite computer desktop; le finalità sono il backup e lo scambio tra diversi dispositivi.

Nessuno degli attori del sistema ha necessariamente buone conoscenze riguardo l'uso dei computer ed è quindi importante predisporre un'interfaccia utente intuitiva e facile da usare, sfruttando se possibile paradigmi già consolidati. L'utilizzo di device che non hanno bisogno di cablaggio per il funzionamento e che possono comunicare attraverso l'etere è sicuramente un'agevolazione pratica nell'utilizzo finale.

Capitolo 4

Background tecnologico

In questo capitolo vengono affrontate le tecnologie che sono emerse essere allo stato dell'arte per lo sviluppo di applicazioni mobile. In una prima fase sono valutate tecnologie di sviluppo mobile cross-platform discutendone le limitazioni e le problematiche ad esse collegate. Successivamente si discute la scelta di focalizzazione su un solo tipo di device e di utilizzo di tecnologia Apple® sia hardware che software. Vengono discussi sia il linguaggio con le sue peculiarità sia l'ambiente di sviluppo utilizzato per lo sviluppo del progetto.

1 Sviluppo cross-platform

Tutte le tecnologie attualmente utilizzate in ambito commerciale per lo sviluppo cross-platform mobile si basano sull'idea di scrivere codice che tramite meccanismi ad hoc può essere eseguito su device di case produttrici distinte e con sistemi operativi differenti.

Ciò che tale paradigma vuole promuovere è la possibilità di scrivere codice in un linguaggio indipendente dagli specifici ambienti di sviluppo delle singole piattaforme. I linguaggi che si prestano bene per essere utilizzati per tale scopo sono i linguaggi nati per il web quali HTML, CSS, Ruby, Php, JavaScript.

1.1 PhoneGap

PhoneGap è il framework più famoso per lo sviluppo di applicazioni mobile non native. Esso permette l'esecuzione della stessa applicazione sulla stragrande maggioranza di sistemi operativi attualmente sul mercato: iOS, Android, Palm, Symbian, BlackBerry e Windows Mobile. È open source e utilizza i linguaggi JavaScript per la logica e HTML e CSS per la grafica.

La potenza che PhoneGap promette è qualcosa di molto allettante: poter scrivere codice che può eseguire potenzialmente su qualsiasi device mobile. In realtà ciò che avviene dietro le scene è la creazione di una vera e propria applicazione web che viene caricata dentro una webview sul device. PhoneGap riesce a simulare la veste grafica dei diversi tipi di dispositivi contribuendo a dare all'applicazione un “look and feel” tipico del device. L'applicazione non è quindi nativa per il sistema operativo ospitante, con il conseguente crollo prestazionale.

PhoneGap è installabile tramite plugin per i diversi ambienti di sviluppo e si integra con essi; fa uso di un simulatore ad hoc per i diversi tipi di dispositivi.

1.2 Titanium framework

Il framework Titanium è sviluppato da AppCelerator Inc; il progetto è nato nel dicembre 2008 ed è stato aggiornato costantemente con nuove feature; la versione attuale è la 1.7. Titanium permette di sviluppare sia per dispositivi mobili sia per piattaforme desktop.

A differenza di PhoneGap questo framework promette lo sviluppo di applicazioni native per il sistema operativo ospitante. Non vi è quindi un meccanismo di “mascheramento” di applicazioni web che vengono rese con un aspetto nativo; il codice finale generato è specifico per l’architettura del dispositivo. Attualmente i sistemi operativi supportati sono iOS e Android. Il meccanismo alla base di Titanium è complesso, ed è basato su un’architettura composta dai seguenti moduli:

Proxy: classe base che rappresenta il collegamento tra il codice JavaScript e il codice nativo; contiene il codice per gestire ogni metodo e ogni evento lanciato;

ViewProxy: specializzazione di Proxy, rappresenta il modello dei dati e agisce come un controller per gestire metodi e cambiamenti di una View;

View: rappresentazione visuale di un componente della UI, gestisce la logica per interagire con il framework grafico nativo;

Module: è uno speciale tipo di Proxy che descrive uno specifico insieme di API o namespace.

Sebbene Titanium sia formato da una moltitudine di moduli che mappano le funzionalità dei framework specifici nativi, non sono presenti moduli per ogni componente nativa. Scrivere un modulo significa creare una componente aggiuntiva che aggiunge funzionalità non ancora mappate ed equivale a fare il lavoro di progettazione e codifica degli sviluppatori di Titanium. L’operazione è molto complicata e sono necessarie ottime conoscenze sia del funzionamento dell’architettura Titanium, sia del linguaggio e ambiente di sviluppo della piattaforma sul quale si vuole eseguire tale modulo. Il concetto che sta alla base della creazione di un modulo è creare delle interfacce per comunicare con le classi del framework nativo e cercare di mappare più funzionalità possibile di queste in Titanium, che si comporta a tutti gli effetti come un wrapper. Per far fronte al vasto insieme di classi degli ambienti di sviluppo Android e iOS, marzo 2011 vede la nascita di Titanium+Plus, uno store online per la compravendita di moduli creati da sviluppatori terzi.

Il codice JavaScript scritto per Titanium mobile viene tradotto dal compilatore in Objective-C (per iOS) o in Java (per Android); può sembrare naïve ma ciò che avviene è una vera e propria compilazione da codice JavaScript a Objective-C/Java. Titanium supporta le interfacce di Android

e iOS e fornisce pieno supporto ai servizi come Twitter, Facebook, Yahoo, nonché funzionalità dei dispositivi come l'accelerometro, la geolocalizzazione e le mappe.

2 Considerazioni

2.1 Problemi inerenti allo sviluppo cross-platform

Negli ultimi due anni sono nate differenti scuole di pensiero, divisi tra i fautori della portabilità del codice a livello di cross-platformness e i puristi del codice nativo. Personalmente ritengo che la vera portabilità del codice avvenga quando vi è un'infrastruttura ad hoc per l'esecuzione di codice compilato in un linguaggio uguale per ogni piattaforma. L'esempio più calzante è il bytecode Java: avendo a disposizione la macchina virtuale capace di interpretarlo, questo esegue in maniera identica su ogni piattaforma. La promessa di portabilità che framework come PhoneGap fanno è solo un'illusione che comporta un deficit prestazionale dovuto al motore del browser oltre al tentativo poco elegante di simulazione di effetti grafici nativi del sistema operativo.

Sebbene Titanium abbia lo stesso scopo ultimo di PhoneGap, esso è raggiunto in maniera totalmente diversa e sicuramente più nobile in quanto il codice finale è specifico per la piattaforma. Purtroppo tale tecnologia, per quanto interessante e con ottime possibilità di successo commerciale, non è attualmente matura e può prestarsi più o meno bene a seconda dei requisiti del progetto che si deve sviluppare.

Vengono elencate di seguito le forti barriere che sono emerse dalla valutazione di questa tecnologia per lo sviluppo del progetto di tesi.

Collegamento a display esterno: un requisito fondamentale dell'applicazione del progetto di tesi è la possibilità di utilizzare l'uscita hardware video del server (dispositivo tablet) per riprodurre il segnale su monitor esterno. Questo in ambiente iOS avviene tramite l'utilizzo della classe UIScreen che al momento della valutazione della tecnologia non era stata ancora interfacciata da Titanium;

Interfacciamento con tablet: la possibilità di sviluppo su tablet quali iPad e GalaxyTab non è ancora resa possibile dal framework;

Editor grafico: la mancanza di un editor grafico per la creazione della GUI è un punto a sfavore di questo ambiente di sviluppo; progettare l'intera interfaccia grafica via codice è un'operazione ormai superata da qualsiasi ambiente di sviluppo e porta lo sviluppo stesso a ostacolare piuttosto che a facilitare lo sviluppatore;

Bonjour: un requisito fondamentale dell'applicazione di tesi è la connessione wireless. Ciò viene implementato tramite il protocollo Bonjour che al momento della valutazione di Titanium non era stato implementato per la parte desktop. Ciò avrebbe reso impossibile la comunicazione tra dispositivi mobili e fissi se fosse nata la necessità di sviluppare un'applicativo server desktop alternativo al tablet;

Funzionalità di logica: funzionalità base come la generazione di numeri casuali, rappresenta un problema in quanto da mesi erano presenti bug aperti sul forum degli sviluppatori Titanium indicando tale grave limitazione quale sintomo di una tecnologia non ancora solida;

Prestazioni: la compilazione su device è lenta ed evidentemente non ottimizzata. È possibile notare dalla shell del front-end che piccole modifiche a parti del codice causano la ricompilazione di intere parti del programma, causando attese non accettabili. Lo sviluppo del progetto di tesi, per sua natura, viene eseguito praticamente sempre su device fisici e non su simulatori software. Compilazioni su device impiegano dai 30 secondi fino ai 5 minuti con una macchina di prestazioni medie. Questo fattore rallenterebbe drasticamente i tempi di sviluppo;

Supporto: la documentazione, seppur chiara, spesso non basta e l'aiuto della comunità degli sviluppatori è fondamentale per qualsiasi ambiente di sviluppo. Essendo Titanium una tecnologia non ancora matura, non può contare sull'interazione di una community molto vasta.

Le mancanze sopra elencate sono state riscontrate nello sviluppo di un prototipo iniziale di "Play With Eyes". A valle di uno studio basilare del framework, in fase di progettazione di alto livello ma soprattutto in fase di codifica, la documentazione e la community hanno portato alla luce determinate impossibilità di implementazione. La tecnologia viene dunque valutata non matura per i motivi sopracitati e non adatta ad essere utilizzata.

Altre sono le motivazioni a sfavore dell'utilizzo di Titanium degne di nota:

Mancanza di standard: Titanium non è la metodologia standard per sviluppare applicazioni mobile: è solo un modo per interfacciarsi alle API (propriarie o meno) di specifiche piattaforme. Un framework che nasce con questo scopo non è detto abbia il nulla ostacolare commerciale da

parte delle case produttrici dei device, che spesso guardano con forte sospetto queste tecnologie che di fatto aggiungono un livello di indirezione nello sviluppo. Inoltre non tutte le classi native possono essere interfacciate da Titanium e ciò causa limitazioni nell'utilizzo (come è il caso della classe UIScreen).

Natura wrapper: l'involucro che Titanium crea per le classi native è un wrapper; il cambiamento delle API comporta la riscrittura o l'aggiornamento dei suoi moduli.

I fattori positivi che Titanium promuove, ovvero la velocità nella scrittura del codice e i servizi aggiuntivi che la casa produttrice offre a pagamento, sono mitigati dal fatto che non è una piattaforma solida e ufficiale, ma una facile scorciatoia per lo sviluppo veloce di progetti che difficilmente possono vantare manutenibilità ed estendibilità. Il pensiero comune alla cerchia degli sviluppatori professionisti è quello di non usare meccanismi semplicistici; come si può notare su portali di riferimento per il code hosting come GitHub [16] e CodeBase [10], i repository che presentano progetti mobile non nativi sono un numero irrisorio.

2.2 Motivazioni della scelta di sviluppo nativo

La scelta di usare tecnologia Apple è motivata dai seguenti fattori:

target ben definito: il target per il progetto di tesi è ben definito: gli utilizzatori finali saranno maestri degli asili, medici oculisti e ovviamente i bambini in età pediatrica. Le prime due categorie raramente hanno conoscenze tecnologiche tali da approcciarsi senza difficoltà a device di nuova generazione, quindi risulta saggio utilizzare prodotti che riducono al minimo il gap tecnologico. La terza categoria di utenti presenta la necessità di utilizzo immediato senza barriere evidenti per l'utilizzo medesimo: device dall'aspetto pulito sono adatti a bambini, e utilizzare device professionali (smartphone dedicati al mondo dell'Impresa o palmari) per l'interazione di un bambino risulterebbe una forzatura;

facilità d'utilizzo: la possibilità di interagire con gesture su schermi touch rappresenta per tutti i tipi di utilizzatori finali un vantaggio per l'abbattimento del gap tecnologico e per usabilità e accessibilità migliori tramite una GUI intuita e più efficiente nell'utilizzo;

tecnologia matura: in anni di innovazione tecnologica continua, i device Apple sono in commercio da un arco di tempo importante (2008 per iPod Touch e iPhone, 2010 per iPad), l'utilizzo di essi è entrato a fare parte della vita di molte persone; appoggiarsi a tecnologie conosciute dalle masse è un grande punto a favore;

documentazione: la documentazione ufficiale è molto completa e ricca di esempi, frutto di varie iterazioni nel corso degli ultimi anni di sviluppo del sistema operativo e del framework;

supporto: il mondo degli sviluppatori iOS è fitto e molto attivo: lo startup per lo sviluppo è ormai alla portata di qualsiasi sviluppatore che possiede le basi della programmazione. Non è difficile trovare soluzioni a problemi informatici semplicemente consultando le comunità di sviluppo presenti online, che spesso propongono svariate soluzioni a problemi banali come a quelli di più difficile soluzione.

L'altra tecnologia attualmente degna di nota per lo sviluppo mobile risulta essere il sistema operativo Android di Google. La piattaforma è basata sul kernel Linux, usa il database SQLite, la libreria dedicata SGL per la grafica bidimensionale e supporta lo standard OpenGL ES 2.0 per la grafica tridimensionale [17]. Le applicazioni vengono eseguite tramite la Dalvik virtual machine, versione di Java virtual machine adattata per l'uso su dispositivi mobili.

La varietà di device con sistema operativo Android viene valutato come fattore negativo per l'utilizzo di questa piattaforma; lo sviluppo porterebbe alla focalizzazione dell'uso di uno specifico modello (o limitati) di smartphone, vanificandone la scelta tecnologica. Ottenerne il medesimo comportamento di un'applicazione su differenti dispositivi non lo si può dare per assodato, anche in presenza dello stesso sistema operativo. Dispositivi Android, inoltre, mostrano un design più complesso con la presenza di un numero di tasti fisici che risulterebbe fuorviante per un bambino; la moltitudine di tasti aumenterebbe la probabilità di interazioni non volute, spesso non gestibili via software.

Gli aspetti elencati precedentemente come fattori positivi per lo sviluppo su piattaforma Apple® sono risultati più pregnanti per questa tecnologia piuttosto che per quella di Google.

Un fattore che ha indubbiamente influito sulla scelta è la, seppur basilare, conoscenza pregressa posseduta per lo sviluppo in ambiente iOS; ciò ha permesso di risparmiare giorni di lavoro che sarebbero stati altresì dedicati allo studio del framework.

3 Sviluppo nativo

3.1 Ambiente di sviluppo

I dispositivi mobili Apple presentano un sistema operativo proprietario che prende il nome di iOS. Lo sviluppo di applicativi nativi per tale sistema operativo prevede l'utilizzo dell'iOS SDK, kit di sviluppo utilizzabile solo in ambiente Macintosh. Al momento della stesura del documento la versione di iOS è la 4.3.3. Versioni precedenti eseguivano su device specifici; attualmente la stessa versione di iOS può essere installata sia su iPad che sui più piccoli device iPod Touch o iPhone grazie ad una sempre maggiore unificazione del kernel. Il kit di sviluppo si può scaricare gratuitamente previa registrazione al portale di Apple dedicato agli sviluppatori [5], anche se la tendenza che pare emergere è quella di portare l'ultima versione dell'SDK proprietario ad un livello più commerciale e a pagamento. È necessaria inoltre la sottoscrizione di una licenza di sviluppo iOS per avere la possibilità di installare le applicazioni sviluppate su un dispositivo fisico e per poter pubblicare e vendere i propri prodotti sull'App Store, il negozio virtuale di Apple. Tale licenza commerciale ha un prezzo di 79€ annui; il Dipartimento di Matematica dell'Università degli Studi di Padova ne è già dotata.

3.1.1 Xcode

L'IDE Apple prende il nome di Xcode. L'ultima versione 4.0 rilasciata a marzo 2011 introduce grandi cambiamenti che permettono allo sviluppatore una codifica molto più rapida rispetto alle versioni precedenti dell'IDE. Oltre ad ampliare le API con il supporto della nuova versione di iOS 4.2 rilasciata con Xcode4, questo porta l'integrazione dello strumento visuale di creazione delle interfacce grafiche in un'unico applicativo. Precedentemente alla versione 4.0, lo strumento grafico prendeva il nome di Interface Builder e l'utilizzo non era obbligatorio. Un IDE dovrebbe promettere un'integrazione totale dei tool di sviluppo (come la lettera 'I' dell'acronimo suggerisce) ma a differenza di ambienti di sviluppo per piattaforme diverse quali Qt o .NET, è singolare notare che solo nel 2011 Apple è riuscita a raggiungere un ambiente di sviluppo veramente integrato.

Xcode attualmente fornisce tutte le funzionalità necessarie per creare e gestire progetti, compilare, eseguire e fare il debug del codice. Da notare che

dalla versione 4 dell'IDE è possibile utilizzare l'infrastruttura di compilazione LLVM, che tra i vari vantaggi permette compilazioni fino a 3 volte più veloci di GCC e un'analisi sintattica attualmente allo stato dell'arte. A differenza di caratteristiche ormai palesi e comuni a tutti gli ambienti di sviluppo professionali come la syntax highlighting e l'autocompletamento del codice, le precedenti caratteristiche fanno di Xcode uno degli ambienti di sviluppo maggiormente apprezzati dagli sviluppatori.

3.1.2 Interface Builder

Interface Builder è uno strumento ormai integrato in Xcode che permette di creare l'interfaccia utente delle applicazioni in modo visuale tramite meccanismi di drag and drop e settaggio di proprietà senza costringere lo sviluppatore ad essere a conoscenza del codice del View. I componenti includono i controlli standard iOS della libreria grafica UIKit (label, text field, button ecc.). I file generati hanno estensione .xib e contengono tutte le informazioni.

Tramite un meccanismo di binding è possibile collegare metodi a determinati eventi: ciò è molto semplificato e intuitivo ed è possibile farlo in maniera totalmente grafica.

Facendo riferimento al pattern Model-View-Controller (MVC), vengono chiamati IBOutlet i binding che dal View fanno riferimento a oggetti definiti nel Controller, mentre prendono il nome di IBAction i collegamenti di metodi presenti nel Controller a specifici eventi scatenabili sul View, ovvero sull'interfaccia utente.

Il contenuto dei file è un dialetto xml che rispetta uno schema definito da Apple. Sebbene sia possibile per lo sviluppatore editare direttamente il contenuto dei file, non vi è necessità in quanto lo strumento visuale è ampiamente sufficiente.

3.1.3 Simulator

Per l'esecuzione delle applicazioni non è necessario un device fisico, ma è possibile simulare l'esecuzione dell'applicazione sull'iOS Simulator, un simulatore software dei device Apple. Fatta eccezione per la possibilità di accesso a risorse fisiche come la fotocamera o l'accelerometro, il kernel installato sul simulatore è lo stesso presente sui device.

La simulazione è molto accurata e il comportamento dell'applicazione è pressoché identico rispetto ai dispositivi reali. Dato che il simulatore esegue su un computer desktop e accede alle risorse hardware di questo, i tempi di esecuzione non sono paragonabili a quelli di un device, che ha performance minori. L'utilizzo del simulatore permette un notevole risparmio di tempo rispetto all'utilizzo dei dispositivi fisici dove la fase di compilazione è seguita dalla fase di installazione, molto onerosa in termini di tempo.

3.1.4 Tool secondari

Strumenti secondari per lo sviluppo risultano essere estremamente utili durante le varie fasi della codifica. Con Xcode e lo strumento Instruments è possibile infatti:

- eseguire analisi prestazionali;
- esaminare il comportamento dei processi ed eventuali interazioni tra processi concorrenti;
- fare il debug e controllare il contenuto degli oggetti utilizzati a run time;
- memorizzare informazioni relative a specifiche esecuzioni dell'applicazione;
- monitorare l'allocazione degli oggetti nel tempo e quindi l'utilizzo della memoria;
- controllare la presenza di leak (falle nella gestione della memoria) dovuti ad una codifica non corretta.

Tali strumenti sono facili e intuitivi da usare e fanno di Xcode un IDE completo che non tutti gli ambienti di sviluppo posso vantare. Controllare ad esempio l'allocazione degli oggetti e la presenza di leak a run time, permette di rilevare anomalie che porterebbero a situazioni di cattiva gestione della memoria. Se non corrette, tali anomalie potrebbero causare la chiusura da parte del sistema operativo dell'applicazione se questa richiedesse troppe risorse non correttamente gestite.

Una peculiarità di Xcode è l'introduzione dell'analizzatore sintattico Clang [9] che fa parte della struttura di compilazione LLVM il cui sviluppatore capo è Chris Lattner.

L'idea che sta alla base di un analizzatore sintattico è quella di scovare errori di programmazione senza eseguire l'applicazione, ma soltanto ispezionando sapientemente il codice sorgente. Senza analizzatore sintattico tali potenziali errori verrebbero rilevati soltanto a runtime.

L'analizzatore Clang è open source e analizza codice C e Objective-C, permette un utilizzo molto user friendly e tramite un paradigma visivo apprezzabile. In figura 4.1 è possibile risalire alla causa del seguente errore di programmazione. Se la variabile `x` vale 1 (secondo caso dello switch), alla terminazione del metodo `foo` non viene rilasciato l'oggetto allocato a riga 13.

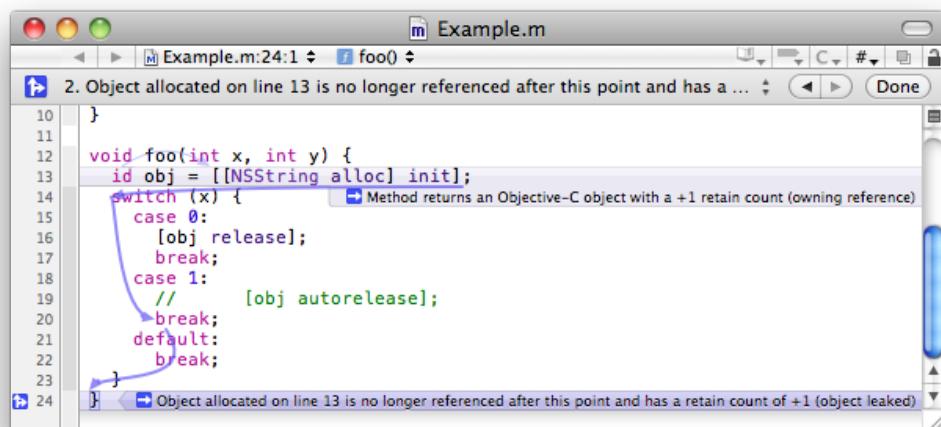


Figura 4.1: Analizzatore sintattico Clang

Gli analizzatori sintattici sollevano il programmatore da molti oneri di testing, ma non sono perfetti e non possono rilevare tutti i bug nel codice. L'istanza di un oggetto singleton, ad esempio, che viene allocata ma non viene mai deallocated, non supera il controllo dell'analizzatore statico che segnala tale situazione come un leak¹. Clang, di cui Xcode è l'unico IDE che ne fa uso, è attualmente la soluzione open source di più alto livello. Analizzatori sintattici commerciali permettono controlli più approfonditi del codice con risultati migliori anche se lo stato dell'arte è rappresentato dall'ampio settore di ricerca dedicato all'analisi statica. Xcode è attualmente l'unico IDE che fa uso di Clang.

¹Recentemente Apple ha apportato modifiche alla sua documentazione consigliando linee guida per la creazione di oggetti singleton che superano il controllo dell'analizzatore statico.

3.2 Linguaggio

Il linguaggio di programmazione usato per lo sviluppo di applicazioni per iPhone/iPad è Objective-C [3]. Tale linguaggio è un'estensione ad oggetti del linguaggio C e ne mantiene la completa compatibilità. Il modello di programmazione dell'Objective-C è basato sullo scambio di messaggi tra oggetti, similmente a quanto avviene in Smalltalk (al quale è ispirato). Il paradigma a cui Objective-C fa riferimento, seppur sempre mirato all'object-orientation, è differente dal C++ dal quale si distacca fortemente.

Brevi cenni storici sono in questo caso necessari per spiegare la scelta di questo linguaggio utilizzato nella pratica soltanto da Apple. Con la nascita di Smalltalk nel 1980 si cerca di soppiare alla mancanza dell'orientazione agli oggetti del C; dopo 3 anni nasce la prima versione di Objective-C che si ispira a Smalltalk per il paradigma di scambio messaggi. Con la nascita della società NeXT nel 1988, il linguaggio viene utilizzato per la creazione del sistema operativo NextStep in maniera del tutto parallela al filone di orientazione agli oggetti portato avanti da C++. Quando nel 1996 Apple acquisisce NeXT, inizia lo sviluppo del sistema operativo che ci accompagna fino ai presenti anni, MacOS, anch'esso scritto in Objective-C. In quanto il kernel di iOS è un sottoinsieme delle funzionalità del kernel di MacOS, l'intera infrastruttura software di Apple si fonda sul linguaggio Objective-C che gli sviluppatori sono forzati ad utilizzare senza alternative.

Siccome il linguaggio C++ ha avuto una storia molto più articolata di Objective-C e può essere considerato un linguaggio di programmazione di riferimento, si presta bene per un confronto. Nei seguenti paragrafi di cerca di evidenziare le maggiori differenze tra i linguaggi.

3.2.1 Sintassi di invocazione dei metodi

A differenza del C++, in Objective-C i metodi non vengono “chiamati”, ma vengono inviati dei messaggi all’oggetto che espone il metodo.

La sintassi del C++ prevede che un metodo venga chiamato nel seguente modo:

```
oggetto.metodo(parametro1, parametro2, ...)
```

l’equivalente in Objective-C è:

```
[oggetto metodoConPrimoParametro:parametro1 eSecondoParametro:parametro2]
```

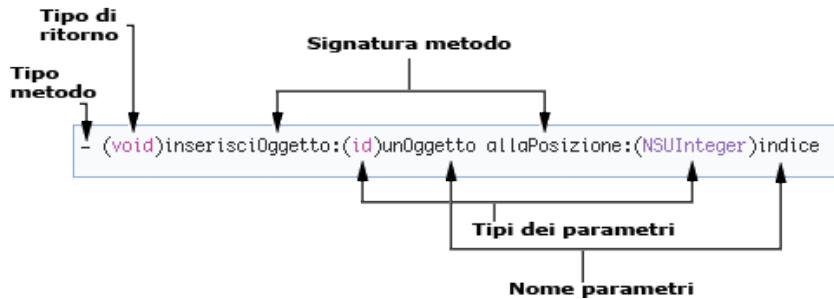


Figura 4.2: Signatura di un metodo in Objective-C

La figura 4.2 illustra la dichiarazione di un metodo in Objective-C. I metodi contraddistinti dal simbolo “-” sono di istanza, quelli contraddistinti dal simbolo “+” sono invece metodi di classe. Una particolarità interessante riguarda il nome del metodo che viene in pratica suddiviso per dare una definizione ad ogni parametro; questa caratteristica può rendere la lettura dei metodi più chiara rispetto alla tradizionale dichiarazione stile C.

Il paradigma di invio dei messaggi è molto potente e consente di scrivere codice utilizzando pattern specifici. Una caratteristica molto usata è la possibilità di inviare messaggi a `nil`; semplicemente l'azione non avrà effetti a runtime. Sono molti i pattern del framework Cocoa che fanno uso di questa caratteristica; il seguente codice mostra un utilizzo valido di invio di messaggi a `nil`.

```
id oggettoChePotrebbeValereNil = nil;
...
if ([oggettoChePotrebbeValereNil metodoCheRitornaUnAltroOggetto] == nil)
{
    // implementazione continua...
}
```

3.2.2 Creazione di oggetti

Objective-C prevede la creazione di un oggetto in 2 fasi distinte: allocazione ed inizializzazione. Tramite l'allocazione viene riservato all'oggetto il quantitativo di memoria necessario; questa operazione viene effettuata attraverso il metodo di classe `alloc`. Tramite l'inizializzazione vengono settati i valori iniziali dell'oggetto; l'istruzione utilizzata allo scopo è `init`. Usualmente entrambe le operazioni vengono eseguite in una singola linea di codice come mostrato di seguito:

```
id mioOggetto = [[ClasseMioOggetto alloc] init];
```

Si fa notare che durante la creazione di un oggetto, se il metodo di classe per l'allocazione fallisce (e viene ritornato il valore `nil`), per la legalità di mandare messaggi a `nil` il successivo metodo di istanza invocato non avrà effetto evitando crash a runtime.

3.2.3 Gestione della memoria

In Objective-C la memoria può venir gestita tramite garbage collector (gestione automatica della memoria) o con la tecnica del reference counting, ovvero il semplice utilizzo di un contatore di riferimenti agli oggetti. La prima possibilità, introdotta con la versione 2.0 del linguaggio, è applicabile al solo sviluppo di applicazioni desktop; per applicazioni mobile l'efficienza gioca un ruolo fondamentale e viene quindi imposto l'utilizzo del reference counting. Dalla versione 4.2 di Xcode, non ancora disponibile, sarà possibile utilizzare una feature molto utile e molto attesa dagli sviluppatori: l'Automatic Reference Counting (ARC). Se utilizzata, essa solleverà in toto lo sviluppatore dall'onere della gestione della memoria: il compilatore inserirà sapientemente le invocazioni `retain`, `release` su qualsiasi oggetto e non sarà quindi più necessaria nemmeno l'implementazione dei distruttori (metodi `dealloc`). Per ora questa feature non è disponibile e viene quindi di seguito spiegato il meccanismo alla base della gestione della memoria.

Ogni oggetto allocato sullo heap dispone di un “retain count”, ovvero un contatore che memorizza il numero di puntatori che riferiscono ad esso. Alla creazione dell'oggetto il suo retain count è pari a 1. È possibile agire sui retain count invocando sull'oggetto i seguenti metodi preposti dal linguaggio:

retain: incrementa il retain count di 1

release: decremente il retain count di 1

La figura 4.3 mostra un esempio di ciclo di vita di un oggetto e il variare del suo retain count. All'allocazione

L'allocazione con inizializzazione di un oggetto imposta il suo retain count a 1, successive operazioni di `retain` e `release` da parte di altre entità causano la variazione di un'unità del retain count. Se un oggetto viene utilizzato da una entità, questa indica la volontà di usarlo effettuando la procedura di

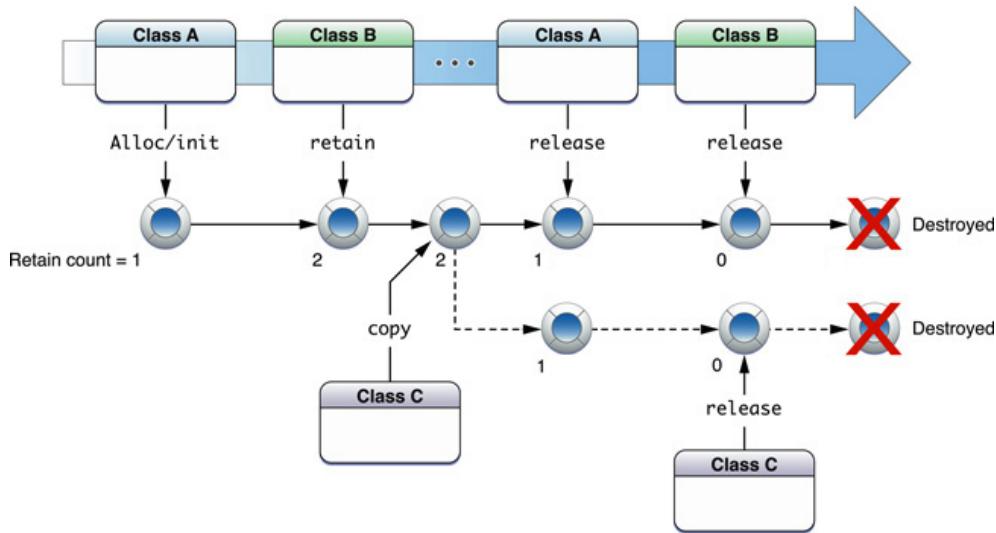


Figura 4.3: Gestione manuale della memoria in Objective-C

`retain`, e solo quando l'utilizzo dell'oggetto è stato completato potrà venir rilasciato per mezzo della `release`. Rispettando queste semplici regole, solo quando un oggetto non sarà più utilizzato da nessuno potrà raggiungere un retain count pari a 0. La copia profonda di un oggetto setta il retain count del nuovo oggetto a 1. Quando un oggetto con retain count pari a 1 viene rilasciato tramite `release`, esso non è più raggiungibile e il sistema operativo invoca la deallocazione (metodo `dealloc`) su di esso. Cercare di accedere ad un oggetto deallocated (situazione di dangling pointer) causa banalmente un crash dell'applicazione a runtime.

Tale gestione manuale della memoria è quanto di più grezzo si possa concepire in un linguaggio orientato agli oggetti. La scelta architettonale di non utilizzare forme di gestione della memoria più astratte, come i riferimenti inversi e la gestione gerarchica degli oggetti utilizzati nella libreria Qt o qualche forma di smart pointer per la gestione automatica del retain count, è motivata soltanto dal bisogno di estrema efficienza.

Un oggetto non più usato all'interno di un metodo ma sul quale non viene invocata una `release` dopo il suo utilizzo crea un memory leak (individuabile con l'analisi statica del codice). La necessità di utilizzare tale oggetto anche dopo l'esecuzione del metodo prevede l'utilizzo di un meccanismo chiamato “autorelease pool”.

Se in un frammento di codice è presente questa entità, gli oggetti sui qua-

li viene invocato il metodo `autorelease` vengono aggiunti a tale oggetto. Quando l'autorelease pool viene rilasciata con il comportamento classico di un qualsiasi altro oggetto, essa effettua il metodo `release` sugli oggetti in essa presenti.

In questo modo l'oggetto viene rilasciato dopo un determinato periodo di tempo (al termine del run loop corrente), delta temporale che non deve interessare al programmatore.

Con la versione 2.0 del linguaggio viene introdotto il garbage collector che però è disponibile solo per le applicazioni desktop. Tale scelta è dovuta al fatto di voler rendere le applicazioni mobile il più leggere e efficienti possibile seppur caricando il programmatore dell'onere di gestione della memoria.

3.2.4 Protocolli e delegazione

I protocolli sono un meccanismo per definire dei metodi che una classe deve implementare. Spesso sono (forse erroneamente) accomunati alle interfacce di Java; in realtà essi sono intrinsecamente legati al concetto di delegazione sul quale l'intero framework Cocoa è basato.

L'interfaccia, in senso stretto, di una classe espone il suo repertorio di metodi pubblici; le entità esterne alla classe interagiscono con essa rispettando questo contratto. Tanto quanto una classe espone metodi, così essa può anche rendere noto il fatto di invocare dei metodi facenti parte di un protocollo legato semanticamente ad essa. Tali metodi devono essere implementati altrove e la classe che promette di implementarli prende il nome di classe delegata. La dichiarazione di un protocollo ha una sintassi simile alla seguente:

```
@protocol MioProtocollo
- (void)primoMetodoDaImplementare:(id)oggetto;
- (void)secondoMetodoDaImplementare:(id)oggetto;
@end
```

Una classe adotta un protocollo quando ne implementa i metodi. La sintassi per dichiarare l'adozione di un protocollo nell'interfaccia e la successiva implementazione è la seguente:

```
@interface MiaClasseDelegata <MioProtocollo> {
    // variabili di istanza ...
}
@end
```

```
@implementation  
- (void)primoMetodoDaImplementare:(id)oggetto { ... }  
- (void)secondoMetodoDaImplementare:(id)oggetto { ... }  
@end
```

La classe delegante che utilizza un protocollo per comunicare con entità esterne ad essa presenta un campo dati con la seguente sintassi:

```
id <MioProtocollo> delegate;
```

Tale campo dati viene settato dandogli il riferimento all'oggetto delegato per rendere noto che sarà esso ad occuparsi dell'implementazione. Quando l'oggetto delegante invoca i metodi che sono stati delegati, ciò viene fatto sull'oggetto delegato che ha tipo statico `id` in quanto non vi sono vincoli sul tipo dinamico della classe che ne fornisce l'implementazione. Ciò avviene con la sintassi di invocazione già discussa:

```
[delegate primoMetodoDaImplementare:oggetto];
```

Il framework Cocoa fa uso profondo del delegation pattern e non è possibile progettare un'applicazione senza essere pienamente consci di tali meccanismi. Altri ambienti di sviluppo usano paradigmi diversi per l'interazione tra entità e oggetti: ad esempio le librerie Qt utilizzano un meccanismo di segnali e slot molto più pulito ed elegante, dove gli oggetti coinvolti non sono costretti a forme di riferimento tra di loro (Meta Object System). Due principali svantaggi della delegazione sono la complicazione del codice nella lettura e l'eccessiva lunghezza che ne deriva dall'utilizzo di tale pattern.

3.3 Cocoa Framework

Cocoa Touch è il macro framework orientato agli oggetti che permette lo sviluppo di applicazioni per il sistema operativo iOS. I framework che lo compongono sono Foundation e UIKit. Foundation raccoglie delle classi che estendono le funzionalità di Objective-C, mentre UIKit permette la gestione degli elementi grafici.

La figura 3.6 riassume visivamente la composizione di Cocoa Touch: dentro i riquadri che rappresentano Foundation e UIKit sono elencate le classi che li compongono a puro scopo di dare un'idea generale.

Foundation è un framework che arricchisce notevolmente il linguaggio Objective-C aggiungendo classi e funzionalità di logica. Questo framework venne

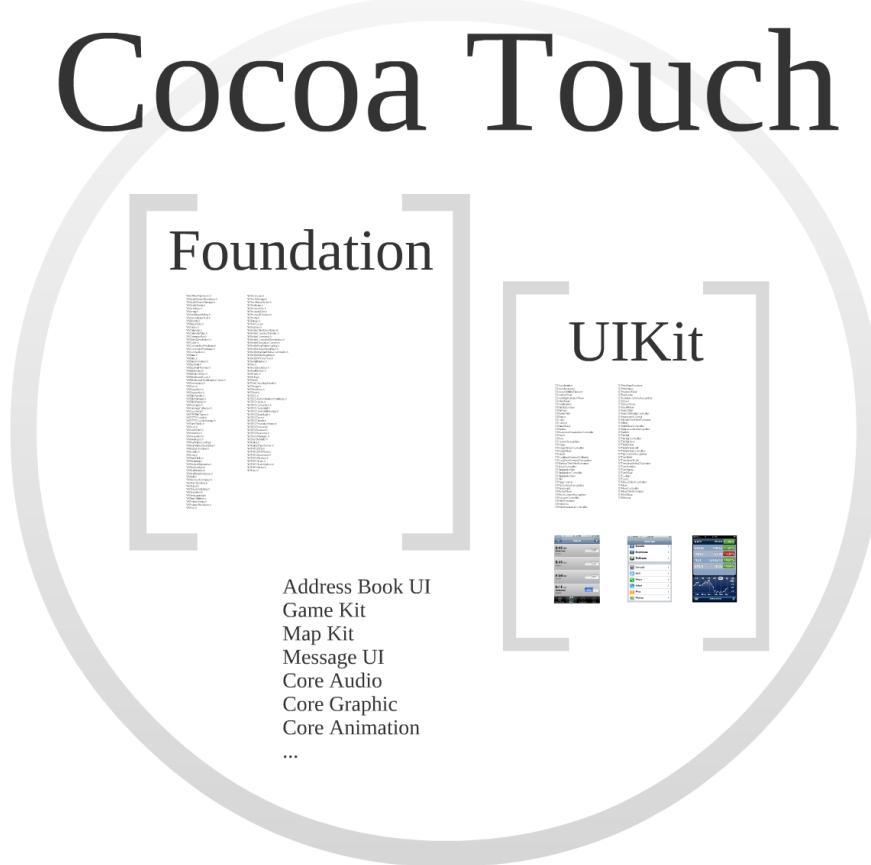


Figura 4.4: Rappresentazione visuale della composizione del framework Cocoa Touch

inizialmente sviluppato dalla società NeXT, come base del sistema operativo NeXTSTEP. Per retaggio storico tutte le classi di Foundation iniziano con le lettere NS, acronimo di NeXTSTEP.

3.3.1 UIKit

UIKit è il framework che permette di costruire e gestire l'interfaccia grafica di applicazioni iOS. UIKit rende disponibili una varietà di classi che consentono la gestione degli eventi, la rappresentazione grafica dei componenti ed i controlli relativi all'interfaccia utente dei dispositivi iOS. Tutte le classi facenti parte di UIKit presentano il prefisso UI. Alcune delle classi più utilizzate di questo framework sono:

UIWindow: gestisce tutte le schermate (dette view) e ne distribuisce gli

eventi; è indispensabile in quanto un oggetto di tale tipo dev'essere la radice della gerarchia di view di un'applicazione, rendendo possibile la loro visualizzazione;

UIView e UIViewController: classi che rappresentano una view dell'applicazione ed il suo controllo rispettivamente;

UITableView e UITableViewController: classi che rispettivamente rappresentano la visualizzazione di una tabella ed il suo controllo.

3.3.2 Pattern MVC in Cocoa

Il design pattern Model-View-Controller (MVC) è diventato negli ultimi anni qualcosa a cui ogni ambiente di sviluppo deve fare riferimento. Anche Xcode quindi è fortemente incentrato su di esso. Il pattern MVC permette di isolare la logica dell'applicazione dall'interfaccia utente (controlli e presentazione), permettendo uno sviluppo indipendente delle parti. L'utilizzo di tale design pattern rende quindi più semplice creare del codice manutenibile e riutilizzabile. Come suggerisce il suo acronimo, MVC prevede 3 ruoli che gli oggetti possono assumere: *model*, *view* e *controller*.

Model

Un oggetto di tipo *model* incapsula i dati e definisce la logica e le funzioni che agiscono su tali dati; non dovrebbe quindi occuparsi dell'interfaccia e della presentazione grafica. Le azioni degli utenti che vanno a creare o modificare i dati vengono comunicate attraverso l'oggetto *controller*, che invia gli aggiornamenti all'oggetto *model*. Quando un oggetto *model* cambia (ad esempio, perché riceve dei nuovi dati tramite una connessione di rete), notifica l'oggetto *controller* che aggiornerà in modo appropriato l'oggetto *view*. Oggetti di questo tipo fanno parte del framework Foundation e gestiscono interazioni logiche con il database o la forma di salvataggio dei dati utilizzata.

View

Un oggetto *view* rappresenta un elemento che l'utente potrà effettivamente vedere all'interno dell'applicazione. Gli oggetti *view* visualizzano, tramite l'interfaccia grafica, i dati presenti nell'oggetto *model*; possono inoltre rendere possibile la modifica di tali dati tramite eventi di interazione. Un oggetto *view*

viene aggiornato dei cambiamenti nel *model* dall'oggetto *controller* e può comunicare le modifiche attuate dagli utenti al *controller* che ha il compito di aggiornare il *model*. Oggetti di questo tipo appartengono al framework UIKit e sono gestiti interamente dall'editor grafico in file con estensione .xib.

Controller

Un oggetto *controller* fa da intermediario tra gli oggetti *view* e i rispettivi oggetti *model*. Un *controller* interpreta le azioni degli utenti attuate nell'oggetto *view* e comunica all'oggetto *model* i cambiamenti apportati ai dati. Quando è l'oggetto *model* che cambia, il *controller* comunica i nuovi dati all'oggetto *view* in modo che possano essere visualizzati. Oggetti di questo tipo fanno parte del framework Foundation e hanno tipicamente il suffisso “Controller” al nome della classe.

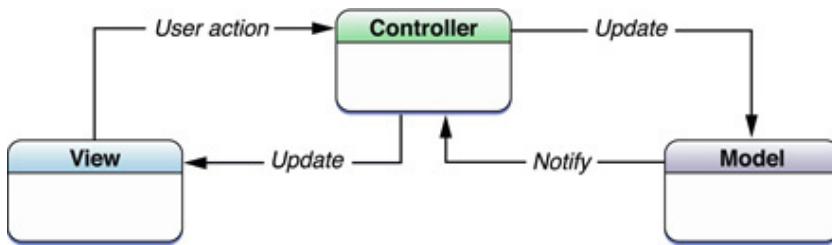


Figura 4.5: Design pattern Model-View-Controller

La figura 4.5 illustra il concetto del pattern MVC, evidenziandone le interazioni.

Capitolo 5

Sviluppo del progetto “*Play With Eyes*”

L’applicazione “Play With Eyes” è stata sviluppata con lo scopo di essere un serious game che possa essere utilizzato sia nella scuole dell’infanzia per effettuare lo screening visivo, sia in ambulatori per eseguire test della vista con un medico.

In questo capitolo vengono illustrate le scelte implementative effettuate durante la fase di sviluppo ed eventuali modifiche a ciò che è stato preventivato nell’idea realizzativa del progetto. Viene descritto il prototipo finale realizzato e le caratteristiche che presenta a valle di modifiche iterative necessarie per venire incontro alle esigenze nate durante le settimane di sviluppo. Viene discusso come è stato affrontato lo sviluppo del progetto di tesi dal punto di vista metodologico e tecnologico. Vengono inoltre mostrate le problematiche di diversa natura riscontrate durante lo sviluppo e la descrizione dell’utilizzo del sistema.

1 Dettagli implementativi

1.1 Architettura

L’architettura descritta nella sezione 2 del capitolo 3 è stata implementata utilizzando tecnologia Apple per le motivazioni discusse nel capitolo 4. La notazione utilizzata nella suddetta sezione viene ora sostituita con i seguenti nomi, scelti per le specifiche componenti:

iPad: componente hardware precedentemente riferita con il nome HW01;

iPod Touch/iPhone: componente hardware precedentemente riferita con il nome HW02;

iPadSight: componente software precedentemente riferita con il nome SW01 eseguite su iPad;

iPodSight: componente software precedentemente riferita con il nome SW02 eseguite su iPod Touch/iPhone;

display esterno: componente hardware precedentemente riferita con il nome HW03;

VGA adapter: componente hardware precedentemente riferita con il nome HW04.

L’adattatore VGA, prodotto da Apple, permette il collegamento di un cavo VGA alla presa 30 pin dell’iPad, indispensabile per la connessione col display esterno. In figura 5.1 si mostra l’architettura del sistema.

1.2 Strumenti ortottici utilizzati

Le figure ottotipiche utilizzate per la valutazione dell’acutezza visiva sono i simboli di Lea mostrati in figura 5.2 e le E orientate mostrate in figura 5.3. Per gli esercizi che valutano la visione dei colori sono state utilizzate le stesse figure in aggiunta alla figura stilizzata di un camaleonte; per tali esercizi vengono utilizzati i colori riportati in tabella 5.1.

I colori utilizzati sono quelli proposti da Lea in [24], dove vengono però utilizzate due varietà di colori: la prima a sedici colori, la seconda soltanto a sei. Per la natura di questo strumento non è possibile utilizzare la prima gamma di gradazioni e il motivo risiede nell’architettura del sistema. Dovendo il bambino rispondere su iPodSight in base a ciò che è capace di vedere sul

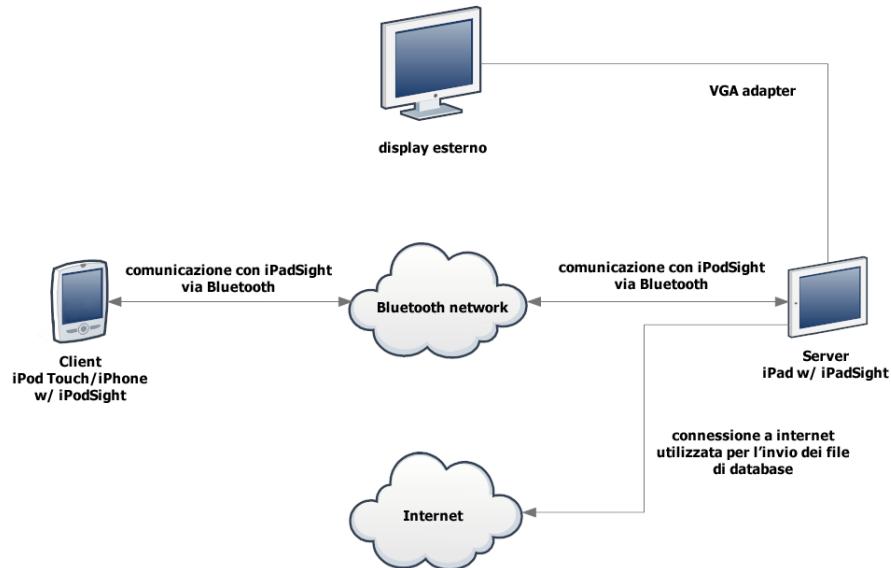


Figura 5.1: Architettura del sistema



Figura 5.2: Ottotipi di Lea [24]: una mela, una casa, un quadrato, un cerchio

E M A S W

Figura 5.3: E orientate

display, non si può avere la certezza che i colori visualizzati sul display vengano visualizzati sullo schermo del client con gli stessi valori di luminosità, contrasto e correzione gamma; gradazioni simili risulterebbero difficilmente distinguibili. Tale problema è arginabile parzialmente impostando la massima luminosità possibile sugli schermi e una corretta luminosità dell'ambiente per favorire la visione, ma di base appare irrisolvibile. Si utilizza quindi la

Nome	Valori RGB	Valori HEX	Esempio
Azzurro	[70, 146, 196]	#4692C4	
Acqua	[32, 156, 156]	#209C9C	
Verde	[40, 162, 89]	#28A259	
Ocra	[70, 146, 196]	#978255	
Rosa	[151, 130, 85]	#A86580	
Viola	[138, 122, 171]	#8A7AAAB	

Tabella 5.1: Colori consigliati da Lea [25]

seconda varietà di colori, sei appunto, per cercare di mitigare il problema dovuto alla tecnologia. I colori sono suggeriti da Lea in maniera tale da mettere in difficoltà soggetti affetti da daltonismo e incapaci quindi di distinguere le gradazioni proposte.

I simboli Lea sono coperti da copyright, la cui detentrice è Lea Hyvärinen. L'utilizzo deve essere quindi autorizzato. Si fa notare inoltre che il software commerciale PVVAT, di cui si è discusso nella sezione 4.1 del capitolo 3, utilizza dei simboli molto simili a quelli di Lea ma non esattamente gli stessi: le figure sono disegnate in maniera leggermente differente. Figure come il cerchio e il quadrato non possono essere protette da copyright e minime modifiche alle forme dei restanti ottotipi, la casa e la mela, sono sufficienti per aggirare il problema. Qualora tale strumento avesse un futuro in ambienti esterni all'Università, si dovrebbe provvedere alla sostituzione dei simboli usati con figure simili non protette da copyright ma con gli stessi vantaggi e proprietà di quelli originali o a contattare la detentrice del copyright per ottenere l'autorizzazione all'uso.

1.3 Utilizzo di immagini tratte da cartoni animati

Il serious game utilizza animazioni visive e contributi sonori per rendere l'esperienza di gioco più coinvolgente per il bambino. Il gioco è lasciato il più minimale possibile per evitare distrazioni da parte del bambino o risultare eccessivamente fuorviante e deconcentrante; allo stesso tempo sono però utilizzate immagini di contorno prese dal noto cartone animato *SpongeBob*

SquarePantsTM che vengono mostrate sia sul display sia su iPodSight. Le immagini usate si riferiscono ai personaggi principali e ben conosciuti del cartone animato.

Si pensa che l'utilizzo di immagini note per i bambini, come appunto quelle tratte dal cartone animato sopracitato, possano aiutare a catturare l'attenzione durante il test.

Come per i simboli di Lea, coperti da copyright, anche l'utilizzo di queste immagini deve essere autorizzato per scopi commerciali. Lo strumento è stato progettato e realizzato in modo tale che la modifica delle sole immagini o del tema sia un'operazione triviale.

1.4 Tipologie di esercizi implementati

Sono stati implementati quattro diversi tipi di esercizi:

Acutezza: viene mostrato un ottotipo ad una determinata grandezza in decimi e viene chiesto di riconoscere la figura mostrata;

Crowding: vengono mostrate più figure ottotipiche su una sola riga ad una certa distanza tra loro e ad una determinata grandezza in decimi; una freccia posta sopra un solo ottotipo indica quale bisogna riuscire ad individuare. Soggetti ambliopi hanno difficoltà nel riconoscere tra i vari simboli quello indicato [11];

Rotazione: viene mostrato un ottotipo, possibilmente una E direzionale, posizionato in una specifica direzione (0, 90, 180, 270 gradi) e viene chiesto di riconoscere che direzione presenta l'ottotipo mostrato;

Colore: viene mostrato un ottotipo o la figura stilizzata di un camaleonte con un determinato colore tra quelli mostrati in tabella 5.1 e viene chiesto di riconoscere quale colore è mostrato.

La difficoltà degli esercizi è espressa in termini di decimi di acutezza visiva. La presenza di tipologie diverse di esercizi è motivata anche dalla necessità di creare un serious game che possa apparire all'utente sottoforma di videogioco con diversi livelli da superare. I test mostreranno possibilmente esercizi di tipologia differente l'uno dall'altro con difficoltà via via crescente. La creazione di test secondo questo schema rappresenta soltanto una linea guida.

1.5 Modalità di gioco e interazione

La scelta del bambino può avvenire secondo due modalità a seconda delle capacità de soggetto di rapportarsi con il dispositivo fisico:

1. modalità di gioco tramite scelta touch
2. modalità di gioco tramite scelta dragging

La prima modalità prevede la scelta della risposta tramite semplice touch sulla figura corretta tra quelle mostrate. Questa modalità è pensata per essere immediata e semplice nell'utilizzo; presenta il livello minimo di astrazione e una richiesta minima di interazione da parte del bambino. La seconda modalità prevede la scelta della risposta tramite dragging o panning, ovvero trascinando e spostando una figura sulla superficie touch del dispositivo. La figura centrale del personaggio deve essere quindi trascinata sopra la risposta corretta, azione effettuabile con un solo dito. Per gli esercizi di rotazione la figura viene invece ruotata; quando si pensa sia posizionata correttamente la scelta avviene tramite la pressione dell'immagine di un personaggio che compare al termine dell'azione di rotazione. Il personaggio raffigurato rappresenta intenzionalmente una figura negativa dell'universo del cartone animato *SpongeBob SquarePants™*. Questa modalità è pensata per essere più interattiva senza aggiungere complessità eccessiva nell'utilizzo.

In figura 6.2 vengono mostrati screenshot dell'applicazione iPodSight nella modalità di gioco tramite scelta touch, mentre in figura 5.5 viene mostrata la modalità di gioco tramite scelta dragging.

Gli esercizi per testare la visione dei colori vengono effettuati in maniera del tutto simile a quelli di acutezza visiva, con la differenza che il bambino non deve riconoscere una forma ma individuare il colore proiettato sul monitor. Ottotipi di Lea o la figura di un camaleonte di dimensioni sufficientemente grandi vengono proposte in diversi colori. Nella tipologia di esercizio che prevede il riconoscimento della direzione degli ottotipi, sebbene sia possibile utilizzare i simboli di Lea (fatta eccezione per il cerchio), è preferibile l'uso delle E orientate mostrate in figura 5.3.

Il display, gestito tramite iPadSight, visualizza le schermate mostrate in figura 5.6 a seconda della tipologia di esercizio scelta.

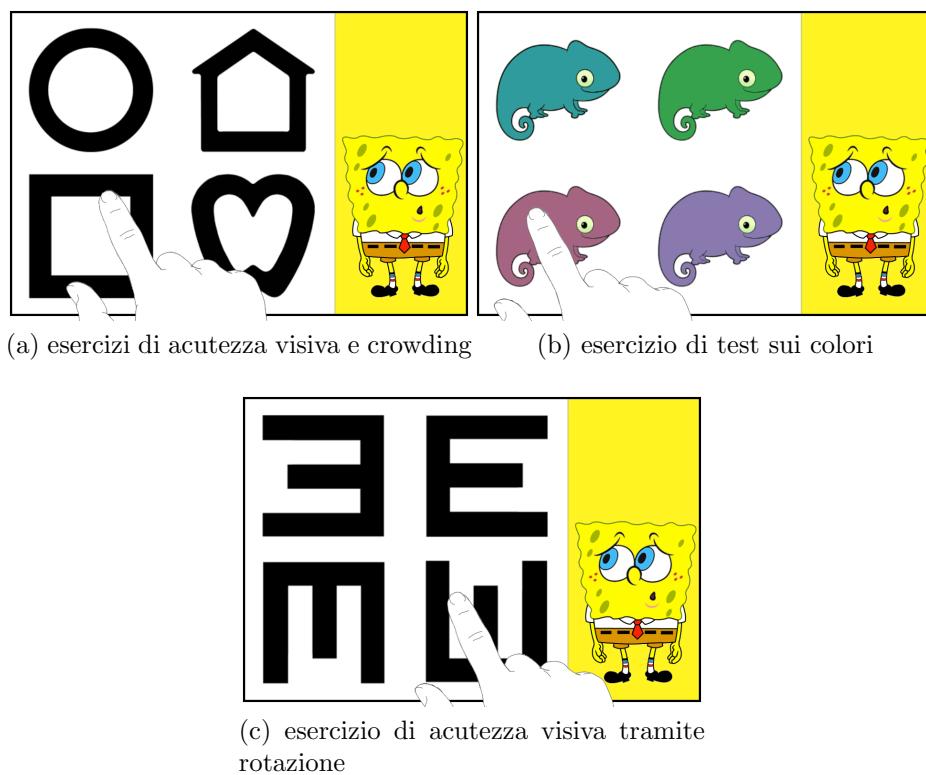


Figura 5.4: Modalità di gioco con scelta tramite touch

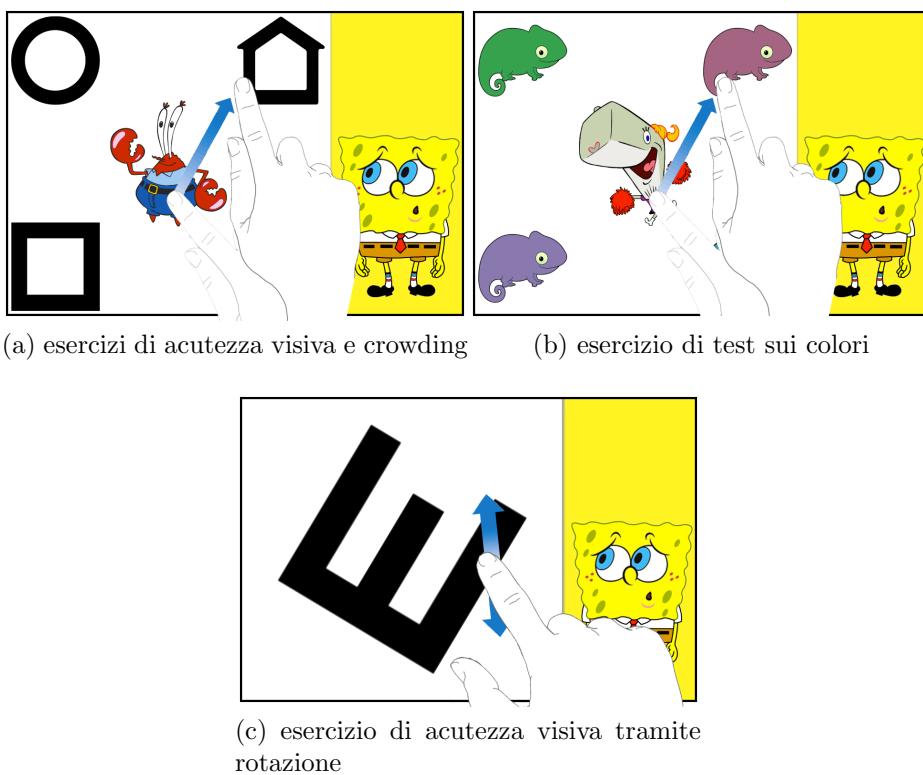


Figura 5.5: Modalità di gioco con scelta tramite dragging

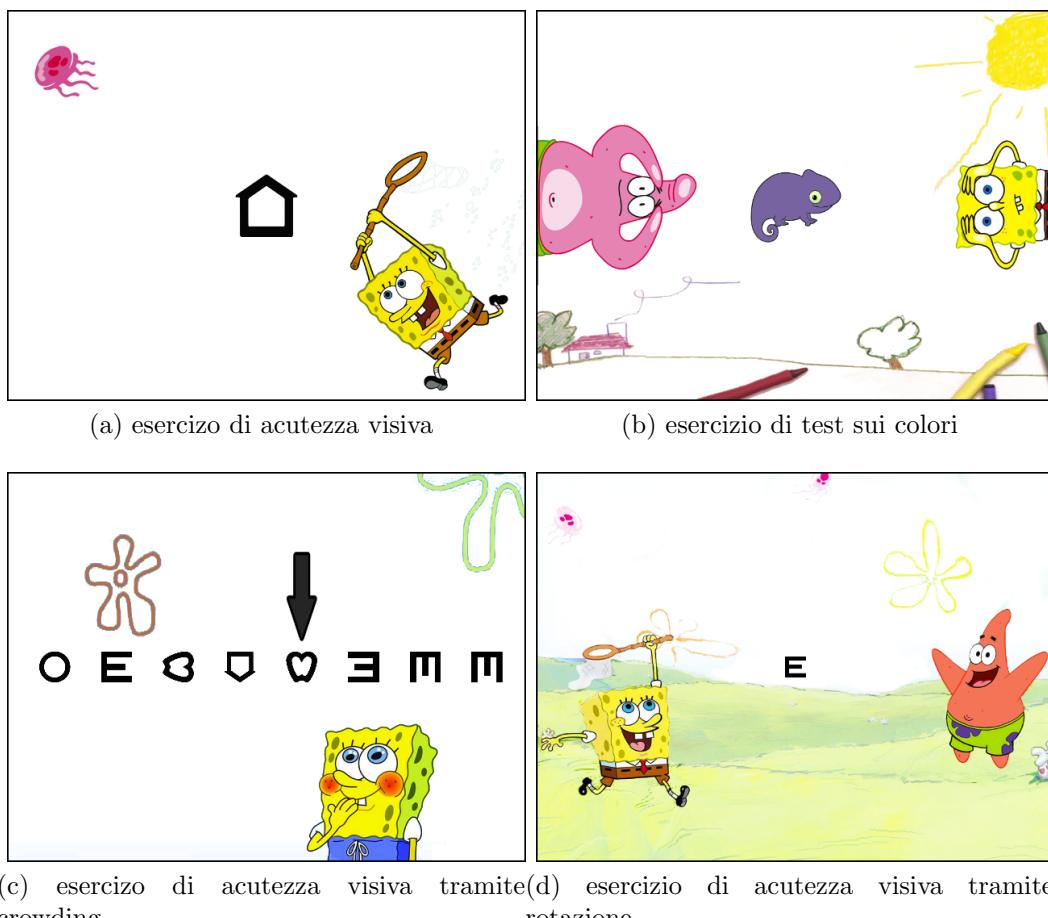


Figura 5.6: Schermate proiettate da iPadSight sul display

Si fa notare l'utilizzo dei personaggi del cartone animato *SpongeBob SquarePants™* in maniera limitata e non invasiva ma sufficiente per favorire il coinvolgimento, come descritto nella sezione 1.3.

1.6 Calcolo grandezza ottotipi

1.6.1 Basi teoriche

L'acutezza visiva è definita come l'ampiezza dell'angolo formato dalle rette congiungenti l'occhio con due punti distanti tra loro nella minima quantità che permetta di distinguere separatamente. Gli ottotipi creati da Snellen possono essere inscritti in una griglia di 5x5 i cui singoli tratti compongono le diverse lettere come mostrato in figura 5.7; secondo Snellen la “visione normale” è la capacità dell'occhio umano di riconoscere un ottotipo quando esso sottende 5 minuti d'arco e quindi discriminare un singolo tratto della dimensione di 1 minuto d'arco.

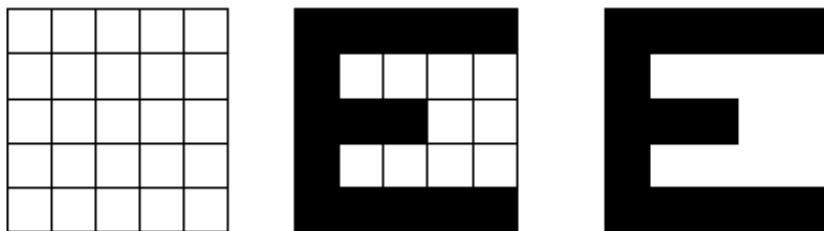


Figura 5.7: Ottotipo disegnato in una griglia 5x5

L'acutezza visiva è definita come la distanza a cui il test viene eseguito diviso la distanza a cui il dettaglio della lettera del test sottende un angolo di 1 minuto d'arco (indicato con 1'), ovvero:

$$AV = \frac{D_{effettiva}}{D_{1'}}$$

Osservando la formula, teoricamente sarebbe sufficiente una singola lettera e variare la distanza di esecuzione del test per ottenere angoli di visione differenti, ovvero mantenere fisso il denominatore e variare il numeratore. Questa procedura all'atto pratico risulta scomoda e richiede spazi ampi per poter raggiungere diversi valori di acutezza visiva; inoltre il soggetto dovrebbe continuamente variare la distanza di esecuzione del test. È più utile variare la

grandezza delle lettere e mantenere fissa la distanza di esecuzione del test. Nonostante ciò, la formula per il calcolo dell'acutezza visiva rispetto alle dimensioni del singolo ottotipo rimane la medesima e pertanto è possibile utilizzarla a patto di calcolare per ogni ottotipo la distanza a cui l'ottotipo stesso sottende 5 minuti d'arco.

I calcoli da effettuare sono molteplici. Si consideri la seguente notazione:

- **H1** = altezza dell'ottotipo
- **H2** = altezza del tratto dell'ottotipo
- **D** = distanza di osservazione
- α = angolo in primi sotteso dal tratto dell'ottotipo

Valgono le seguenti formule:

$$AV = \frac{1}{\alpha}$$

$$\tan(\alpha) = \frac{H2}{D}$$

e l'inversa:

$$\alpha = \arctan \frac{H2}{D}$$

Un ottotipo è uguale a 5 volte il suo singolo tratto:

$$5H2 = H1$$

Quindi il singolo ottotipo sottende un angolo di 5' e unendo le formule otteniamo:

$$AV = \frac{1}{60 \arctan \frac{H1}{5D}}$$

È possibile calcolare l'acutezza visiva di un ottotipo di qualsiasi dimensione H1 ad una particolare distanza D. Allo stesso modo con una formula inversa è possibile calcolare l'altezza dell'ottotipo per creare tabelle ottotipiche per qualsiasi distanza:

$$H1 = 5D \tan \frac{1}{60AV}$$

Si fa notare che AV vale 10/10, in quanto si sta calcolando la visione "normale". È utile mostrare la formula nel seguente modo:

$$\frac{H1}{5 \tan \frac{1}{60AV}} = D$$

Sapendo che:

$$\frac{1}{5 \tan \frac{1}{60}}$$

è una costante e vale 687,5, otteniamo la seguente formula:

$$H1 = \frac{D}{687,5 AV}$$

Avendo ottenuto la costante 687,5 per il calcolo della “visione normale”, possiamo calcolare la grandezza di un ottotipo in base alla distanza di osservazione e all’acutezza visiva che si vuole misurare.

Si mostrano alcuni esempi. A 3 metri un ottotipo visto con acutezza visiva normale (10/10) è alto 4,36 millimetri per il seguente calcolo:

$$\frac{3000mm}{687,5} = 4,36mm$$

mentre a 5 metri risulta dover essere alto:

$$\frac{5000mm}{687,5} = 7,27mm$$

Volendo calcolare invece l’altezza di un ottotipo visto con acutezza visiva di 3/10 ad una distanza di 3 metri, otteniamo:

$$\frac{3000mm}{687,5 * \frac{3}{10}} = \frac{3000mm * 10}{687,5 * 3} = 14,54mm$$

1.6.2 Implementazione

Al fine di ottenere l’altezza di ogni ottotipo (H) da mostrare sul display, possiamo utilizzare dunque la formula per il calcolo dell’altezza dell’ottotipo ricavata nella sezione precedente:

$$H = \frac{D}{687,5 AV}$$

dove D è la distanza di osservazione e AV è l’acutezza visiva. È quindi possibile sapere quanto grande deve essere l’intera figura di un ottotipo per ogni acutezza visiva ed è quindi possibile derivare l’altezza in pixel necessaria per

visualizzare l'ottotipo sul display.

Si consideri il seguente problema: si vuole disegnare un ottotipo per valutare l'acutezza visiva di 2/10 a una distanza di 4 metri; si calcoli il numero di pixel p necessari in altezza per disegnare l'ottotipo conoscendo l'altezza del display e la risoluzione utilizzata. Si dispone di una risoluzione di 1024x768 e di un display 15'.

Utilizzando la notazione:

- \mathbf{A} = altezza del display (immagine utile visualizzata)
- \mathbf{R} = numero di pixel dell'altezza della risoluzione
- \mathbf{D} = distanza di osservazione
- \mathbf{H} = altezza dell'ottotipo
- \mathbf{p} = numero di pixel per il disegno dell'ottotipo in altezza
- \mathbf{f} = numero di volte in cui è possibile disegnare l'ottotipo in altezza sul display

R vale 768, mentre A vale circa 270 mm (considerando un monitor 15').

L'altezza dell'ottotipo per quanto detto finora è:

$$H = \frac{4000}{687,5 * \frac{2}{10}} = \frac{4000 * 10}{687,5 * 2} = 29,1$$

Il numero di volte in cui è possibile disegnare l'ottotipo in altezza sul display:

$$f = \frac{A}{H} = \frac{270}{29,1} = 9,28$$

Il numero dei pixel p è quindi dato da:

$$p = \frac{R}{f} = \frac{768}{9,28} = 82,7$$

Sia R , A , D , e AV sono variabili gestite a software. L'impostazione di questi valori permette in primis l'utilizzo dello strumento con display di differenti dimensioni e in secundis la scelta della distanza a cui effettuare il test, conferendo di fatto caratteristiche di dinamicità al software. Come prova pratica di correttezza, prima dell'inizio del test, viene mostrato sul display il disegno

di un righello riportante le misure dei centimetri e dei pollici. Se la risoluzione del display viene rilevata correttamente e viene inserita nel software l'altezza corretta dello schermo, la dimensione del righello disegnato è confrontabile con uno fisico. Ciò permette di essere sicuri riguardo alla corretta altezza degli ottotipi disegnati per effettuare il test alla distanza stabilita.

1.7 Connessione Wireless

La comunicazione tra device iOS avviene tramite il protocollo Bonjour, implementazione Apple di Zeroconf, protocollo per la configurazione dinamica dei nodi di una rete utilizzante il protocollo IP. Le API permettono un utilizzo totalmente trasparente del protocollo; di seguito si descrivono le basi di Bonjour.

1.7.1 Apple Bonjour Service

Zeroconf (protocollo standard dell'IETF) è un insieme di tecniche che permettono di creare in maniera automatica una rete IP senza operazioni manuali di configurazione. Esso permette ai device di collegarsi alla rete automaticamente e di riconoscere altre periferiche disponibili in rete. Il servizio Bonjour, implementazione di Apple di Zeroconf rappresenta un livello di astrazione ulteriore a quest'ultimo.

La soluzione proposte da Bonjour copre tre aree descritte in seguito:

1. **addressing**: assegnazione di un indirizzo IP ai dispositivi;
2. **naming**: utilizzo di nomi propri invece di indirizzi IP per riferirsi ai dispositivi;
3. **service discovery**: individuazione automatica di altri servizi disponibili nella rete.

Con Bonjour non è necessario assegnare indirizzi IP o inserire il nome del servizio che si desidera utilizzare. All'utente è reso disponibile un meccanismo per la ricerca dei servizi disponibili nella rete e la scelta del servizio desiderato da una lista. Le applicazioni possono individuare automaticamente i servizi di cui hanno bisogno, individuare altre applicazioni con cui poter interagire o effettuare connessioni e scambio di dati in maniera trasparente per l'utente. Tali caratteristiche sono ideali per creare connessione e comunicazione tra iPadSight e iPodSight.

Addressing Gli indirizzi vengono assegnati ai dispositivi tramite il link-local addressing. Un link-local address è un indirizzo di rete usato solo per comunicazioni appartenenti ad un segmento di una rete locale (un link) o ad una connessione point-to-point a cui un dispositivo è connesso. La tecnica del link-local addressing sfrutta solo un certo range di indirizzi riservati alla rete locale. Il funzionamento prevede l'assegnazione un indirizzo IP random che viene immediatamente testato. Se tale indirizzo risulta non essere già utilizzato, sarà assegnato al dispositivo, in caso contrario si ripete il passo precedente fino a trovare un indirizzo disponibile.

Naming Con naming si intende l'assegnazione di un identificativo univoco ad un servizio, per poterlo poi pubblicare nella rete locale. Quando un servizio viene pubblicato nella rete significa che si rende visibile agli altri dispositivi e disponibile all'utilizzo. Quando un servizio vuole individuare altri servizi nella rete, ma anche verificare se il nome (o l'indirizzo) che vuole utilizzare è disponibile, usa la tecnica del Multicast DNS (mDNS): vengono spedite contemporaneamente delle richieste nella rete ad un range di IP. Tali richieste hanno lo scopo di individuare tutti i servizi che corrispondono ad un certo tipo o appartengono ad un dato dominio. Quando un servizio può rispondere ad una richiesta i cui parametri corrispondono ai suoi, risponde al mittente fornendo il suo indirizzo.

Service Discovery Il service discovery permette alle applicazioni di individuare tutti i servizi a disposizione nella rete mantenendo una lista di nomi. Ogni applicazione traduce tali nomi in un indirizzo IP e in una porta che serviranno ad utilizzare il servizio. Questa lista di servizi disponibili presenta un enorme vantaggio: elimina la necessità di notificare tutti i dispositivi della rete nel caso in cui un dato servizio cambi, ad esempio, il suo indirizzo IP. L'indirizzo IP e la porta di un dato servizio vengono recuperati nel momento in cui l'utente seleziona il servizio dalla lista; anche se l'utente avesse selezionato un servizio il cui indirizzo è mutato, l'operazione di traduzione (resolving) fornirebbe l'indirizzo corretto. C'è quindi allocazione dinamica dei servizi e le applicazioni ne salvano solamente i nomi. Come spiegato nella sezione Naming viene inviata una richiesta di tipo mDNS allo scopo di individuare tutti i servizi di un certo tipo ed appartenenti ad un dato dominio. Ogni servizio che corrisponde alle richiesta risponderà fornendo il proprio

nome. Il risultato finale è una lista di tutti i servizi disponibili da cui sarà possibile scegliere il servizio desiderato.

1.7.2 Connessione tramite GameKit

Il framework GameKit permette di creare applicazioni ludiche e tra le varie funzionalità presenta una implementazione di Bonjour su rete Bluetooth. L’implementazione è verbosa per il numero di metodi delegati necessari da implementare ma logicamente risulta semplice. La documentazione [4] è molto esaustiva e ricca di esempi specifici per creare connessione ed esporre servizi rispettando i passaggi del protocollo Bonjour.

Ciò su cui si vuole porre accento è la semplicità data all’utente per stabilire la connessione tra iPadSight e iPodSight. In figura 5.8 si mostrano le semplici view con le quali interagire tramite poche azioni touch per stabilire la connessione. Si supponga di avere due dispositivi denominati A e B, il flusso è come descritto di seguito. La ricerca di servizi disponibili da entrambi i device visualizza la schermata 5.8a. I servizi reperiti vengono mostrati in una lista come in figura 5.8b. Selezionando un servizio dalla lista sul device A, B visualizza la richiesta di connessione da A come in figura 5.8c, mostrando la possibilità di accettare o rifiutare la connessione.

Purtroppo la connessione Bluetooth non è supportata dal simulatore iOS, di conseguenza tutti i test del progetto sono stati effettuati su dispositivi fisici con il conseguente rallentamento dei tempi di sviluppo.

È possibile implementare uno dei due scenari mostrati in figura 5.9: il primo prevede l’utilizzo di un server puro¹ e di un client; il secondo prevede invece l’utilizzo di due peer, ovvero dare la possibilità ad entrambi i dispositivi di accettare, richiedere connessioni ed esporre servizi. Sebbene l’architettura descritta presenti concettualmente un server puro e un client, per semplicità viene implementato il secondo scenario a 2 peer. Così facendo si ha un importante vantaggio: non è importante da quale dispositivo si richieda connessione. Non sfruttando un servizio che viene esposto solo da iPadSight o

¹Per server puro si intende un server che accetta richieste ed eroga servizi, ma non intraprende azioni verso altre entità.

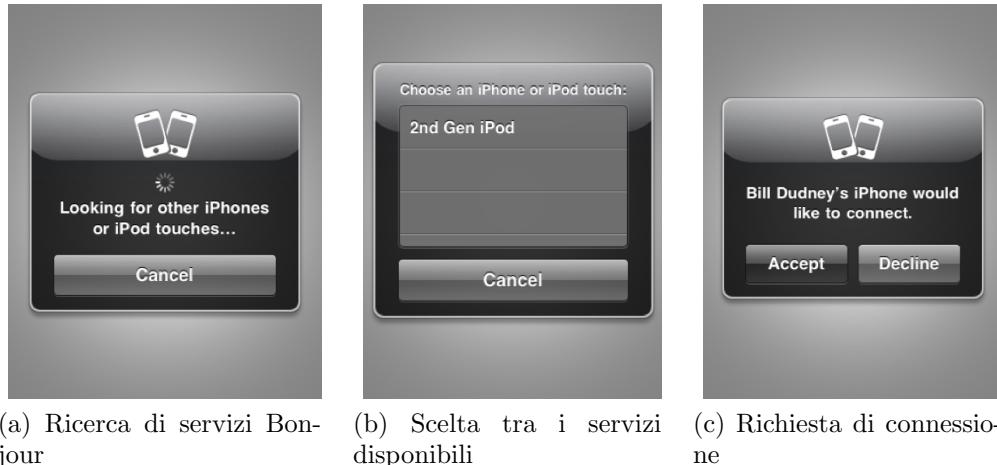


Figura 5.8: Connessione Bonjour tramite GUI

iPodSight, ma richiedendo solo connessione tra dispositivi per scambio messaggi, non è concettualmente sbagliato permettere ciò; In modalità peer inoltre si può sfruttare la metafora grafica mostrata in figura 5.8 che consente la scelta dei servizi da una lista. L'implementazione dello scenario server/client non avrebbe reso possibile l'uso del componente grafico per la presenza di un solo server esponente il servizio.



(a) Server/Client



(b) Peer

Figura 5.9: Modalità di connessione

Una volta stabilita la connessione, iPadSight e iPodSight salvano a vicenda il riferimento all'altro dispositivo. L'invio di messaggi avviene in maniera bana-

le e la struttura del protocollo di comunicazione a livello applicativo rimane a discrezione del programmatore: sono state usate strutture dati tipiche del linguaggio C, ovvero degli **struct**.

1.8 Comunicazione con display

La connessione tra iPadSight e il display avviene con il VGA adapter. A livello software, schermi esterni sono gestibili con l’oggetto **UIScreen** e il riconoscimento del collegamento o scollegamento fisico del cavo avviene tramite la gestione di eventi scatenati dal sistema operativo.

Di seguito si mostra come è possibile fornire l’implementazione dei metodi che vengono invocati al momento di connessione e disconnessione con la registrazione presso il servizio di notifiche di iOS (oggetto **NSNotificationCenter**).

```
- (void)screenDidConnectNotification:(NSNotification *)notification { ... }
- (void)screenDidDisconnectNotification:(NSNotification *)notification { ... }

[[NSNotificationCenter defaultCenter] addObserver: self
                                         selector:@selector(screenDidConnectNotification:)
                                         name: UIScreenDidConnectNotification
                                       object: nil];

[[NSNotificationCenter defaultCenter] addObserver: self
                                         selector:@selector(screenDidDisconnectNotification:)
                                         name: UIScreenDidDisconnectNotification
                                       object: nil];
```

Di seguito si mostra come è possibile assegnare una finestra (oggetto **UIWindow**) ad uno schermo fisico esterno che è stato rilevato.

```
1 if ([[UIScreen screens] count] > 1) {
2     UIScreen *externalScreen =
3         [[[UIScreen screens] objectAtIndex:1] retain];
4     screenModes = [externalScreen availableModes retain];
5
6     UIScreenMode *desiredMode = [screenModes objectAtIndex:<index>];
7     externalScreen.currentMode = desiredMode;
8
9     UIWindow *externalWindow;
10    externalWindow.screen = externalScreen;
11 }
```

Alla riga 1 viene controllato che vi sia più di uno schermo collegato: l'array statico `screens` dell'oggetto `UIScreen` presenta sempre almeno un oggetto in posizione 0, ovvero lo schermo del dispositivo. Se uno schermo è collegato, questo sarà rappresentato da un oggetto `UIScreen` in posizione 1 dell'array. L'array `screenModes` contiene le risoluzioni disponibili per il dispositivo e a riga 6 viene settato il `desiredMode` a un determinato indice dell'array contenente le risoluzioni. A riga 7 viene impostata la risoluzione scelta sull'oggetto mentre a riga 10 l'oggetto `externalScreen` viene settato come schermo della finestra `externalWindows`. Tutto ciò che viene disegnato su `externalWindows` dalla riga 10 in poi verrà visualizzato sul monitor esterno.

La possibilità di usare schermi esterni è resa disponibile solo dalla versione 4.0 del sistema operativo con l'oggetto `UIScreen`; versioni precedenti non permettevano l'utilizzo di tali API (private) e nemmeno la funzione di mirroring dello schermo era permessa.

1.9 Database

Per gestire gli utenti, le classi e i test in maniera persistente (requisito RF_SW01_01o) si è dovuto adottare un meccanismo per la gestione di un database. La scelta della tipologia del database è stata dettata in particolar modo dalla richiesta di permettere un meccanismo per l'invio dei risultati ottenuti tramite mail (requisito RF_SW01_07d) e un meccanismo per lo scambio di configurazioni (utenti e test) tramite computer desktop (requisito RF_SW01_08d). Si è giunti all'utilizzo di formati aperti per la rappresentazione dei dati. La scelta che è sembrata più consona inizialmente è stata l'utilizzo di un OODBMS, ovvero un database in cui l'informazione è rappresentata sottoforma di oggetti tipici della programmazione object-oriented. Un OODBMS aiuta i programmatore a rendere gli oggetti creati in un linguaggio di programmazione a comportarsi come un oggetto di database.

Per Objective-C non è consigliato l'utilizzo di un OODBMS, sia per l'assenza di tool sviluppati, sia per la presenza del noto sistema DBMS di Apple Core Data. L'uso di Core Data prevede conoscenze molto buone di tematiche affini ed è risaputo non essere immediato nell'utilizzo.

La base di dati, fin dalla fase di analisi, non è apparsa di complessità tale da richiedere l'utilizzo di un database relazionale; la gestione delle classi,

degli utenti e dei test è stata gestita a livello di application logic. In figura 5.10 vengono mostrate le classi wrapper concettuali per la rappresentazione dei dati a livello applicativo, in figura 5.11 vengono invece mostrate le classi singleton per gestire la persistenza della base dati su file in formato JSON.

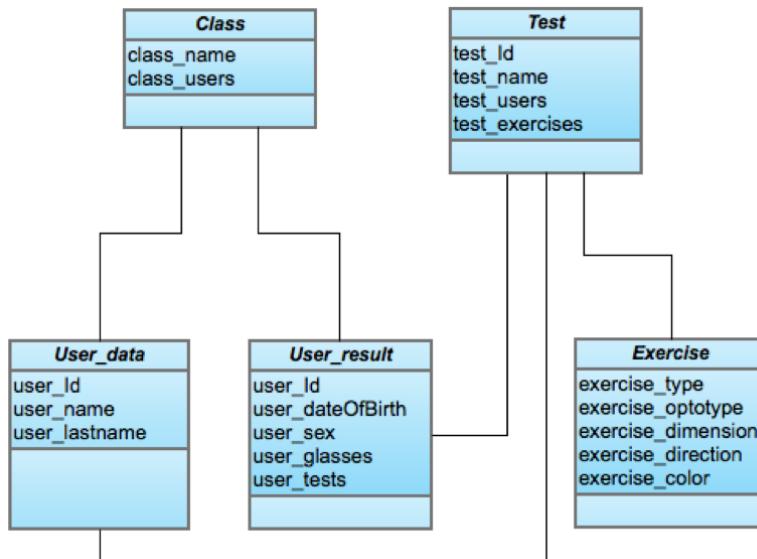


Figura 5.10: Diagramma delle classi wrapper

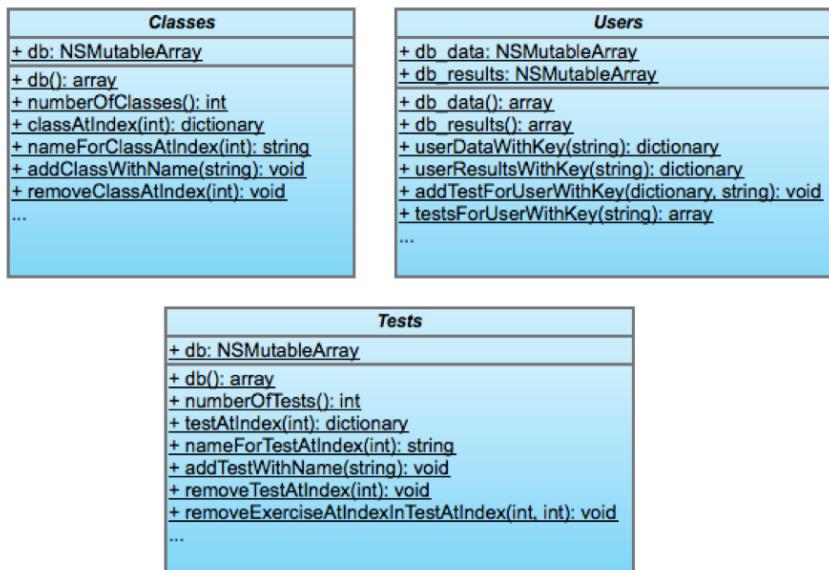


Figura 5.11: Diagramma delle classi singleton

L'utilizzo del formato JSON (utilizzando la libreria SBJSON) invece di un dialetto XML (Apple supporta il formato *plist* per il salvataggio di file xml) è stato motivato dal voler avere un formato semplice la cui modifica manuale non comportasse una sintassi eccessivamente verbosa. Permettere il salvataggio in formato XML, se necessario, è un'operazione triviale dato il supporto già presente nel framework Foundation.

Privacy nel trattamento dei dati Per garantire la privacy è stato necessario separare i dati quali nome e cognome degli utenti da quelli che vengono effettivamente utilizzati a fini statistici come illustrato in figura 5.10. Effettuare questa divisione ha reso possibile la persistenza dei dati su file separati, consentendo di fatto l'esportazione dei soli dati riguardanti i risultati dei test (*User_result*) con il conseguente rispetto della privacy secondo la legge L. 675/1996.

1.9.1 Struttura dei file JSON

Si mostra la struttura dei dati salvati nei file JSON utilizzando, oltre alla notazione JSON standard [35], il formalismo tipico delle grammatiche formali in quanto calzante in questo contesto. La raffigurazione delle strutture secondo la notazione JSON mostrata in [35] sarebbe stata troppo onerosa e non necessaria. Tale notazione (che descrive i valori STRING e NUMBER) prevede la separazione degli elementi di array e dizionari tramite il carattere virgola. Nella seguente notazione mista inoltre, l'asterisco indica la presenza di zero o più predicati precedenti il simbolo e il carattere pipe indica la scelta (operazione di OR).

Le strutture dati in figura 1.9 sono descritte con le seguenti entità:

```
CLASSES:= [ CLASS* ]
USERS_DATA:= [ USERS_DATA_ENTRY* ]
USERS_RESULTS:= [ USERS_RESULTS_ENTRY* ]
TESTS:= [ TEST* ]
```

utilizzando la seguente grammatica definita:

```
CLASS:= {
    "users": USERS_KEY,
```

```

    "name": STRING
}
USERS_KEY:= [ NUMBER* ]
DECIMO_VALUE:= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10
USERS_DATA_ENTRY:= {
    "name": STRING,
    "surname": STRING,
    "key": NUMBER
}
USERS_RESULTS_ENTRY:= {
    "tests": TESTS_USER_ENTRY,
    "key": NUMBER,
    "sex": STRING,
    "dateOfBirth": STRING,
    "glasses": BOOL
}
BOOL:= true | false
TESTS_USER_ENTRY:= [ TEST_USER_ENTRY* ]
TEST_USER_ENTRY:= {
    "ref": NUMBER,
    "note": STRING,
    "eye": STRING,
    "duration": NUMBER,
    "answers": ANSWERS
}
EXERCISES:= [ EXERCISE* ]
EXERCISE:= EX_ACUITY | EX_ROTATION | EX_CROWDING | EX_COLOUR
ANSWERS:= [ ANSWER* ]
ANSWER:= ANSWER_ACUITY | ANSWER_ROTATION |
    ANSWER_CROWDING | ANSWER_COLOUR
OPTOTYPE_VALUE_1:= "casa" | "mela" | "quadrato" | "cerchio"
OPTOTYPE_VALUE_2:= "casa" | "mela" | "quadrato" | "cerchio" |
    "camaleonte"
OPTOTYPE_VALUE_3:= "casa" | "mela" | "forchetta"
DIRECTION_VALUE:= "su" | "giù" | "destra" | "sinistra"

```

```
COLOUR_VALUE:= "azzurro" | "acqua" | "verde" | "rosa" |
    "ocra" | "viola"
ANSWER_ACUITY:= {
    "type": "acuity",
    "direction": DIRECTION_VALUE,
    "optotype": OPTOTYPE_VALUE_1
}
ANSWER_ROTATION:= {
    "type": "rotation",
    "rotation": DIRECTION_VALUE
}
ANSWER_CROWDING:= {
    "type": "crowding",
    "optotype": OPTOTYPE_VALUE_1
}
ANSWER_COLOUR:= {
    "type": "colour",
    "colour": COLOUR_VALUE
}
EX_ACUITY:= {
    "type": "acuity",
    "direction": DIRECTION_VALUE,
    "optotype": OPTOTYPE_VALUE_1,
    "dimension": DECIMO_VALUE
}
EX_ROTATION:= {
    "type": "rotation",
    "rotation": DIRECTION_VALUE,
    "dimension": DECIMO_VALUE,
    "optotype": OPTOTYPE_VALUE_3
}
EX_CROWDING:= {
    "type": "crowding",
    "optotype": OPTOTYPE_VALUE_1,
    "detail": DECIMO_VALUE
}
```

```
    }  
EX_COLOUR:= {  
    "type": "colour",  
    "colour": COLOUR_VALUE,  
    "optotype": OPTOTYPE_VALUE_2  
}  
TEST:= {  
    "users": USERS_KEY,  
    "name": STRING,  
    "key": NUMBER,  
    "exercises": EXERCISES  
}
```

Si fa notare che la chiave `ref` di `TEST_USER_ENTRY` fa riferimento al valore della chiave `key` di un `TEST` effettivamente esistente. I valori delle chiavi `users` di `TEST` e di `CLASS` fanno riferimento a valori delle chiavi `key` dei predicati `USERS_RESULTS_ENTRY` e `USERS_DATA_ENTRY` effettivamente esistenti.

2 Sviluppo agile

L'oggetto della tesi abbraccia diversi ambiti disciplinari ai quali è stato necessario approcciarsi con lo studio di tematiche specifiche, perlopiù in campo medico. È risultato quindi naturale procedere con una metodologia di sviluppo agile che non fa assunzioni forti sulle fasi di sviluppo da seguire.

Nell'ingegneria del software, per metodologia agile si intende un particolare metodo per lo sviluppo del software che coinvolge quanto più possibile il committente, come è di fatto avvenuto per questo progetto, ottenendo in tal modo una elevata reattività alle sue richieste.

Il relatore, ha agevolato la possibilità di utilizzo di questa metodologia leggera con una comunicazione stretta e un coinvolgimento continuo; per la natura del progetto non sarebbe stato pensabile adottare un atteggiamento di tipo "hands off". Proprio il coinvolgimento da parte del committente ha permesso consegne frequenti del software per valutazioni e fasi di testing continue.

Sono stati fatti test continui con utenti reali ottenendo un feedback costante sulla parte di serious game. A causa del target del progetto, non è stato possibile chiedere direttamente ai bambini come è stata l'esperienza di gioco ma è stato possibile vedere e giudicare l'interazione.

Altro aspetto tipico della metodologia agile è la fase di refactoring effettuato dopo l'aggiunta di ogni componente funzionante e l'applicazione del TDD (Test Driven Development) sia come metodologia di codifica del codice, sia per il controllo che il refactoring non avesse comportato situazioni anomale.

I metodi agili tentano di ridurre il rischio di fallimento sviluppando il software in finestre temporali limitate chiamate sprint che, in genere, durano qualche settimana. Ogni sprint è un piccolo progetto che contiene tutto ciò che è necessario per rilasciare un percepibile incremento nelle funzionalità del software. Anche se il risultato di ogni singolo sprint non ha sufficienti funzionalità da essere considerato completo, il software viene rilasciato e nel susseguirsi delle iterazioni, deve avvicinarsi sempre più alle richieste del cliente. Alla fine di ogni sprint il team rivaluta le priorità di progetto. Questo modo di procedere è esattamente ciò che è stato necessario fare per il progetto in atto.

I metodi agili inoltre preferiscono la comunicazione in tempo reale, preferibilmente faccia a faccia, a quella scritta (documentazione). Il team agile è

composto da tutte le persone necessarie per terminare il progetto software; deve includere almeno i programmatori ed i loro clienti (ovvero le persone che definiscono come il prodotto finale dovrà essere fatto).

Nella fattispecie il team è stato formato dal relatore nella figura di committente/cliente e dal tesista nella figura del progettista/programmatore.

Per sviluppare in maniera agile è necessaria una conoscenza non indifferente (o almeno non nulla) di best practice e del dominio di sviluppo del software; la conoscenza pregressa dello sviluppo mobile e l’esperienza maturata durante gli anni universitari si sono rivelati fondamentali. Per quest’ultimo e per i motivi precedentemente discussi, l’adozione di una metodologia leggera è risultata naturale.

2.1 I principi della metodologia agile

La formalizzazione dei principi su cui si basano le metodologie leggere è stata oggetto del lavoro di un gruppo di progettisti software e guru dell’informatica riunitisi nell’Agile Alliance [1]. Il documento finale di questo lavoro, il manifesto dello sviluppo agile, è stato poi sottoscritto da un nutrito gruppo di questi professionisti.

L’obiettivo è la piena soddisfazione del cliente e non solo l’adempimento di un contratto. L’uso di queste metodologie, inoltre, serve ad abbattere i costi di sviluppo del software. Tale metodologia è esplosa in concomitanza con la crisi successiva al boom di Internet prendendo spunto dai metodi applicati in piccole software house.

In senso lato questo termine indica tutte quelle metodologie di sviluppo che rivoluzionano i vecchi sistemi di ingegneria del software (modello a cascata, modello a spirale, ecc.), basati sulla raccolta di specifiche e sulla strutturazione sequenziale dello sviluppo software.

In definitiva, i principi su cui si basa una metodologia leggera che seguia i punti indicati dall’Agile Manifesto [2] sono quattro:

1. le persone e le interazioni sono più importanti dei processi e degli strumenti;
2. è più importante avere software funzionante che documentazione;
3. bisogna collaborare con i clienti al di là del contratto;

4. bisogna essere pronti a rispondere ai cambiamenti più che aderire al progetto.

Col primo punto si intende dire che le relazioni e la comunicazione tra gli attori di un progetto software sono la miglior risorsa; meeting frequenti anche su tematiche estranee al progetto portano indirettamente valore aggiunto al prodotto e alle conoscenze del team.

Il secondo punto prevede il rilascio di nuove versioni del software ad intervalli frequenti, mantenendo il codice semplice, avanzato tecnicamente e cercando di ridurre la documentazione al minimo indispensabile.

Il terzo e il quarto punto riferiscono alla sfera del cliente: la collaborazione diretta offre risultati migliori dei rapporti contrattuali e il team di sviluppo dovrebbe essere autorizzato a suggerire modifiche al progetto in ogni momento.

3 Tool utilizzati

3.1 Hardware e licenza di sviluppo

Per lo sviluppo è stata utilizzata la seguente strumentazione hardware e software concessa in prestito dal Dipartimento di Matematica Pura e Applicata dell’Università di Padova:

- iPhone 3GS 16Gb
- iPad 16Gb
- VGA adapter
- licenza iOS Developer

I due device sono stati tenuti in prestito per l’intero periodo di sviluppo, fatta eccezione del dispositivo iPad e il connettore VGA dei quali il Dipartimento si è fornito solo dopo due mesi dall’inizio dello sviluppo del progetto. Per tutto il periodo di sviluppo e precedentemente alla disponibilità del dispositivo iPad, è stato utilizzato in maniera massiccia il simulatore iOS che per le funzionalità non riguardanti la connessione Bluetooth e la visualizzazione su display esterno si è rivelato più che sufficiente.

Il Dipartimento di matematica Pura e Applicata è provvisto di una licenza di sviluppatore iOS; per l’installazione del software sui device è stato necessario l’utilizzo dei certificati digitali che consentono la firma delle applicazioni per mezzo di chiave privata.

3.2 Sistema di versionamento

Per lo sviluppo del progetto è stato utilizzato un repository privato rispettando le basilari norme di sviluppo software. La scelta finale della tipologia di repository da utilizzare, variata tra Git, Hg e Subversion (in quanto sono i sistemi di versionamento più utilizzati) è ricaduta su Git. Per lo sviluppo di un progetto in maniera autonoma senza il contributo di altri utenti sarebbe stato sufficiente l’utilizzo di un sistema Subversion non distribuito; la scelta di usare Git è motivata dai seguenti fattori:

- il sistema è integrato in Xcode;
- il sistema presenta caratteristiche particolari di distribuzione come la staging area e lo stashing dei file;

- il sistema era conosciuto in quanto già utilizzato per svariati progetti universitari.

La scelta sembrata meno adatta è stata l'uso di Hg per la mancanza di integrazione nell'IDE a differenza degli altri sistemi di versionamento.

La totalità dei commit sul repository a fine sviluppo ha superato il numero di 170 con frequenza di aggiornamento quasi giornaliera. Il versionamento del progetto si è rivelato in alcune circostanze estremamente utile per risolvere situazioni anomale come quella di “invalid summary” descritta successivamente (vedi paragrafo 4).

3.3 Componenti aggiuntive

3.3.1 Libreria SBJSON

Il framework Foundation è molto ricco di funzionalità ma non privo di mancanze; una tra le più eclatanti è forse l'assenza di un parser JSON che ormai molti ambienti di sviluppo presentano per il vasto utilizzo del formato. Soluzioni per ovviare a questa mancanza sono di terze parti, reperibili da repository pubblici e rappresentano ormai uno standard de facto. Le librerie per manipolare file JSON in Objective-C sono solo due e ben note nella community degli sviluppatori:

1. SBJSON² (conosciuto anche con il nome “json-framework”)
2. TouchJSON³
3. MTJSON⁴

Tutte permettono di mappare le strutture dati del formato JSON, ovvero dizionari e array reperiti sottoforma di stringhe, in oggetti di Foundation (funzionalità di parser) e viceversa (funzionalità di writer). La conversione del parser avviene quindi tra i seguenti oggetti:

```
NSString -> NSArray  
NSString -> NSDictionary
```

²Reperibile all'indirizzo: <https://github.com/stig/json-framework/>

³Reperibile all'indirizzo: <https://github.com/atomicbird/TouchJSON>

⁴Reperibile all'indirizzo: <https://github.com/mysterioususers/MTJSON>

Il writer invece permette le inverse conversioni:

```
NSArray -> NSString  
NSDictionary -> NSString
```

Le librerie sono state valutate consultando le documentazioni e la scelta è ricaduta sull'uso di SBJSON senza motivazioni particolari in quanto tutte presentano le medesime funzionalità.

La necessità di poter manipolare strutture dati JSON nasce a seguito delle scelte riguardanti il database utilizzato (si veda la sezione 1.9). I file di configurazione dei test e della base dati degli utenti vengono infatti salvati in questo formato facilmente trattabile da uno sviluppatore.

4 Problemi riscontrati

4.1 Comportamenti inattesi

Un problema molto grave è stato riscontrato nella fase finale di codifica. A seguito di un'attenta analisi del comportamento del codice si è potuto evincere che tale problema è causato dall'ambiente di sviluppo.

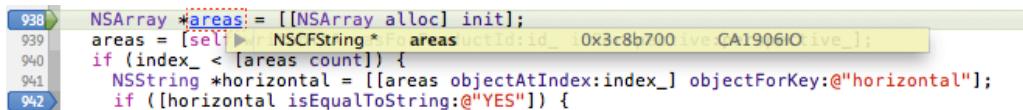
Il problema è di difficile descrizione e riguarda la gestione dei puntatori in memoria che dovrebbe essere gestita dal compilatore a basso livello. Tale problema si è verificato in maniera aleatoria su zone di codice circoscritte durante l'esecuzione dell'applicazione. Si sono rilevati comportamenti anomali come il seguente. La variabile booleana settata a TRUE alla riga 1 del frammento di codice riportato, assume erroneamente valore FALSE quando l'esecuzione raggiunge la riga 2, rendendo impossibile l'esecuzione del relativo ramo *if*.

```
1 BOOL isOn = TRUE;
2 if (isOn) {
3     // L'implementazione del ramo if non è mai raggiunta
4     // il debugger segnala che la variabile isOn a riga 1
5     // vale TRUE ma a riga 2 assume valore FALSE
6 }
```

La situazione appena descritta è avvenuta in assenza di multithreading. Nelle figure 5.12 e 5.13 si mostra l'esecuzione di due istanze successive dello stesso pezzo di codice con gli stessi input e precondizioni. Si può notare l'allocazione e l'inizializzazione di un array (oggetto NSArray di Foundation) che assume valori casuali: in 5.12 contiene una stringa (oggetto NSString di Foundation) che è possibile ispezionare ed accedere da riga 938; in 5.13 contiene un valore non identificato (invalid summary).

Nel secondo caso ogni tentativo di accesso all'oggetto, successivo alla sua dichiarazione, ha causato crash a runtime in quanto il puntatore all'oggetto punta all'indirizzo esadecimale 0x1 (si ricorda che, pur non essendo questo il caso, l'invio di messaggi a nil, indirizzo 0x0, è consentito dal linguaggio).

La comunità degli sviluppatori attribuisce il problema alla versione 4 di Xcode, che assieme a questo problema presenta ancora numerosi bug. Test con compilatori differenti supportati da Xcode quali GCC 4.2, LLVM GCC 4.2

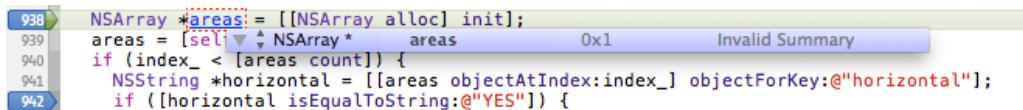


```

938 NSArray *areas = [[NSArray alloc] init];
939 areas = [sel] ► NSString *areas objectID: 0x3c8b700 live CA190610 live_;
940 if (index_ < [areas count]) {
941     NSString *horizontal = [[areas objectAtIndex:index_] objectForKey:@"horizontal"];
942     if ([horizontal isEqualToString:@"YES"])

```

Figura 5.12: Errore rilevato con debugging



```

938 NSArray *areas = [[NSArray alloc] init];
939 areas = [sel] ▼ NSArray * areas 0x1 Invalid Summary
940 if (index_ < [areas count]) {
941     NSString *horizontal = [[areas objectAtIndex:index_] objectForKey:@"horizontal"];
942     if ([horizontal isEqualToString:@"YES"])

```

Figura 5.13: Errore di invalid summary rilevato con debugging

e l’LLVM compiler 2.0 non hanno mutato la situazione. Data la rarità del problema, prima di giungere alla conclusione che questo bug si presenta solo con tale versione dell’IDE, è stata fatta un’analisi del problema a livello di codice rivelatasi inutile e non mirata. Lo stesso progetto infatti, compilato con la versione precedente dell’IDE (versione 3.2), non ha presentato problemi e il debugger durante l’esecuzione non ha mai mostrato situazioni come quelle in figure 5.12 e 5.13.

Al momento dello sviluppo, tra la comunità di sviluppatori è ancor presente molta diffidenza verso Xcode 4, rilasciato in versione ufficiale nel mese di marzo 2011. Indagare la natura di queste situazioni aleatorie è difficile; ciò che può aver forse agevolato la presenza di “inconsistenze” è la creazione del progetto con la versione precedente dell’IDE, e l’aver effettuato la migrazione alla quella nuova, di fatto corrompendo i file di progetto. Un progetto ex novo è stato creato con l’ultima versione dell’IDE. Questa operazione ha, di fatto, risolto il problema.

4.2 Impossibilità di debugging

Il tool di debugging è stato usato in maniera continuativa durante lo sviluppo, sia su device che sul simulatore software; tuttavia in certe circostanze non è stato possibile utilizzarlo. Quando l’applicazione iPadSight mostra output su display esterno, la porta fisica a 30 pin di cui dispone l’iPad è utilizzata dall’adattatore VGA e il device non è quindi collegabile al computer per eseguire il debugging. Xcode rende possibile simulare a software un display esterno solo se l’applicazione è in modalità portrait; tale limitazione è dovuta

a un bug noto non ancora chiuso riguardante il simulatore iOS. L'applicazione iPadSight è in modalità landscape e non è stato quindi possibile utilizzare lo strumento di simulazione del display esterno.

4.3 Dimensione display

Nella fase iniziale descritta nel capitolo 3 si era preventivato l'uso di un display per la proiezione degli ottotipi del serious game, ma non erano state fatte assunzioni né sulla dimensione né sulla risoluzione da adottare; nella sezione 1.6 di questo capitolo si mostra invece come calcolare quanti pixel da utilizzare per il disegno degli ottotipi.

Inizialmente si era ipotizzato l'utilizzo di un proiettore, strumento hardware che tra l'altro avrebbe comportato una complessità maggiore nel disegno architettonico del sistema e l'uso di una parete sgombra per la proiezione. La qualità della lampada dello strumento inoltre avrebbe rappresentato un problema dovuto alla visualizzazione corretta dei colori.

Il fattore che comporta l'impossibilità di utilizzo di un proiettore è legato alla risoluzione. Proiettori comuni supportano risoluzioni di 640x480, 800x600 e 1024x768. Anche prendendo come risoluzione di riferimento quella di 1024x768 è facile dedurre che a distanze ragionevoli per effettuare il test, gli ottotipi sfruttarebbero un numero insufficiente di pixel per essere disegnati con una definizione accettabile.

Minore è la dimensione della proiezione e maggiore è la risoluzione utilizzata, maggiore sarà quindi la definizione ottenuta nel disegno degli ottotipi. Tale aspetto è emerso ingenuamente solo in una fase successiva a quella di analisi. Piuttosto che un problema effettivo, utilizzare un semplice monitor da computer per il display, di dimensioni quindi ridotte rispetto all'immagine proiettata da un proiettore, è risultato una semplificazione pratica. Per la connessione inoltre si utilizza lo stesso tipo di cablaggio VGA.

5 Utilizzo del sistema

L'utilizzo del sistema dovrebbe risultare immediato grazie alla GUI autoesplorativa e all'uso di paradigmi di usabilità Apple noti. Come per la stragrande maggioranza delle applicazioni mobile, non è necessario un manuale utente: lo sviluppo agile, come detto, preferisce software funzionante piuttosto che documentazione. È presente un help inline sia su iPodSight sia su iPadSight.

5.1 Utilizzo di iPadSight

In questa sezione si mostrano le figure delle sole schermate necessarie all'esecuzione di un test in quanto quelle di configurazione della base dati sono pensate per essere intuitive nell'immediato.

Sono solo tre le macro viste dell'applicazione iPadSight selezionabili dalla tab bar mostrata in figura 5.14 e riferite come segue:

- **V1:** esecuzione test (screening)
- **V2:** gestione utenti e classi
- **V3:** gestione test e esercizi

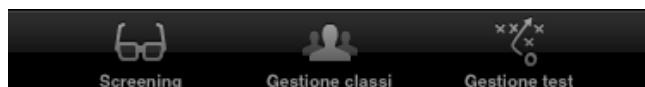


Figura 5.14: Tab bar di iPadSight

Le viste V2 e V3 mostrano un meccanismo di gestione dei dati e configurazione tramite scroll di tabelle e picker che non ha bisogno di descrizioni ulteriori.

La vista V1 presenta la possibilità di inviare via mail i risultati o di passare alla modalità di esecuzione test, la cui schermata è mostrata in figura 5.15. Premendo il pulsante con la lettera “i” in alto a sinistra viene visualizzato un help inline; la prassi per poter effettuare un test è mostrata nella barra in alto ed è la seguente:

1. **Cerca dispositivi mobili:** permette di cercare i dispositivi eseguenti l'applicazione iPodSight in rete locale; la connessione avviene con poche interazioni touch secondo quanto descritto nella sezione 1.7.2;

2. **Configura test:** permette di scegliere quale utente valutare e con quale test, entrambi dati preconfigurati in V2 e V3; è possibile configurare la distanza di osservazione e l'altezza dello display utilizzato;
3. **Cerca schermi collegati:** permette di effettuare la connessione di iPadSight al display con un meccanismo che consente di rilevare il monitor a livello fisico e scegliere la risoluzione da utilizzare tra quelle supportate dal display.

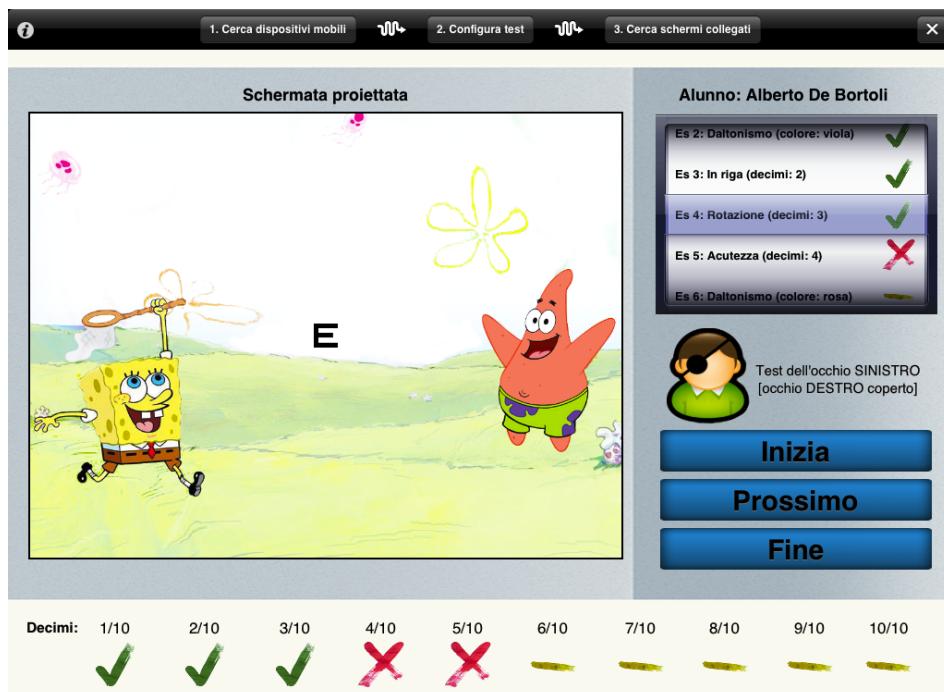


Figura 5.15: Schermata di esecuzione test in iPadSight

Sulla parte sinistra viene mostrata un'anteprima di ciò che viene proiettato sul display, mentre a destra sono presenti dei controlli per l'esecuzione dinamica del test. Il componente picker permette di scegliere quale esercizio del test presentare senza essere vincolati all'ordine prestabilito. Se per qualche motivo si vuole saltare un esercizio (che risulta troppo difficile o non adatto) si può passare al successivo con il pulsante “Prossimo”. Il test può essere terminato tramite la pressione del pulsante “Fine” portando alla richiesta di salvataggio dei risultati ottenuti.

Durante il test, le risposte corrette, errate o non pervenute sono visualizzate con le immagini in figura 5.16 dentro al picker. Viene usata la stessa notazione

in basso per tracciare le risposte in base all’acutezza visiva (misurata in decimi).



Figura 5.16: Legenda utilizzata per la valutazione delle risposte

5.2 Utilizzo di iPodSight

In figura 5.17 si mostra la schermata iniziale che permette la ricerca di dispositivi eseguenti l’applicazione iPadSight in rete locale. Al primo avvio dell’applicazione, con la pressione del pulsante “Inizia test” si avvierà la ricerca finalizzata alla connessione con iPadSight. La connessione avviene con poche interazioni touch secondo quanto descritto nella sezione 1.7.2. Esecuzioni successive possono presentare già una connessione col dispositivo remoto, in tal caso non verrà effettuata la ricerca ma verrà direttamente presentata la schermata di gioco. Prima di iniziare il test è possibile scegliere quale modalità di gioco adottare: tramite scelta dragging (selezionata di default) o tramite scelta touch.



Figura 5.17: Schermata iniziale di iPodSight

La schermata di gioco e le modalità di interazione sono state esplicate nella sezione 1.5 e non vi è necessità di una descrizione ulteriore. Si fa notare l’u-

nico aspetto che necessita di spiegazione: per uscire dalla schermata di gioco (figure 6.2 e 5.5) interrompendo il test, occorre fare doppio touch sull'area gialla in alto a sinistra. La mancanza di un pulsante visibile è voluta in quanto la presenza potrebbe complicare l'interfaccia grafica e distrarre il bambino dal test o invogliarlo all'interazione con quell'elemento. Ciò comporterebbe di fatto l'uscita dalla schermata di gioco e quindi dal test, azione che l'utente configuratore invece, consci del fatto, può effettuare.

Capitolo 6

Test dello strumento

1 Test in fase di sviluppo

Al termine delle varie iterazioni (sprint) della fase di sviluppo del software, sono stati effettuati test da un numero contenuto di bambini di età talvolta più giovane (18-36 mesi) rispetto al target del progetto. Ciò è servito per adattare comportamenti del software e giungere ai test fatti nelle scuole con una consapevolezza maggiore delle potenzialità del software.

1.1 Test del codice

Test sul codice sono stati effettuati in maniera continuativa durante lo sviluppo utilizzando i tool necessari.

Test di unità

Lo *unit testing* è una procedura usata per verificare singole parti di un codice sorgente e per unità si intende genericamente la minima parte testabile. I test di unità verificano il corretto funzionamento di parti di programma permettendo così una precoce individuazione dei bug e semplificando le modifiche e l'integrazione dei pezzi di codice testati.

Xcode mette a disposizione un comodo meccanismo per effettuare test di unità tramite il framework OCUnit. Tramite macro definite dal framework come `STAssertEquals`, `STAssertEqualObjects` o `STAssertFail` si possono scrivere dei test specifici per testare piccole parti di codice e metodi specifici. Il progetto Xcode predisponde una classe per effettuare test ed è necessario

soltanto scrivere il codice per testare le unità interessate. L'esecuzione dei test avviene con una semplice esecuzione lanciata da Xcode e i risultati si possono visualizzare sulla console dell'IDE.

Sono noti altri due tool di unit testing di terze parti per Xcode, ovvero **GHUnit**¹ e **OCMock**² che promettono caratteristiche migliori, ma l'utilizzo di OCUnit è apparso sufficiente e utile solo in determinate occasioni.

Analisi statica

Come già visto nel capitolo 4 si è fatto uso continuo del tool di analisi statica presente in Xcode per controllare l'eventuale presenza di leak. La gestione della memoria avviene manualmente ed effettuare tali controlli rappresenta un aspetto fondamentale di qualsiasi progetto scritto in Objective-C.

Performance

Il tool Instruments è risultato estremamente utile sia per l'identificazione di leak non rilevati dall'analisi statica, sia in svariati test sulle performance dell'applicazione. Principalmente è stato controllato che l'allocazione della memoria non superasse i limiti prefissati e che il carico computazionale gravante sulla CPU non causasse alcun tipo di problema.

¹Reperibile all'indirizzo <http://gabriel.github.com/gh-unit/>

²Reperibile all'indirizzo <http://www.mulle-kybernetik.com/software/OCMock/>

2 Test in scuola materna

La presentazione di un progetto come quello di “Play With Eyes” perderebbe di valore se il software sviluppato non venisse effettivamente testato in una delle sedi previste per il suo utilizzo. Nelle giornate del 17, 18 e 19 maggio 2011 sono stati effettuati test su un totale di 65 bambini presso la Scuola dell’Infanzia “Gianni Rodari” di Mogliano Veneto. Ogni giorno è stato effettuato lo screening in una delle tre classi presenti, per una media di 21 screening giornalieri. Nella seguente figura si mostrano i numeri e le percentuali di adesione allo screening.

	3° anno (5 anni)			2° anno (4 anni)			1° anno (3 anni)			Totale		
	maschi	femmine	totale	%	maschi	femmine	totale	%	maschi	femmine	totale	%
totalità bambini	8	17	25		8	13	21		9	10	19	
visitati	8	17	25	100,00%	7	13	20	95,24%	9	9	18	94,74%
rifiutati	0	0	0	0,00%	1	0	1	4,76%	0	1	1	5,26%
									25	40	65	
									24	39	63	97%
									1	1	2	3%

Figura 6.1: Adesioni allo screening

È stata fatta una presentazione del funzionamento del gioco ad ogni classe in maniera da non impaurire i bambini sottolineando la natura ludica del test. Successivamente è stato effettuato il test di screening ad ogni bambino in una stanza predisposta con l’attrezzatura necessaria.

Lo screening per ogni bambino è avvenuto come segue:

1. È stato ripetuto al bambino come funziona il gioco in caso non avesse capito qualcosa durante la spiegazione a tutta la classe;
2. Sono stati fatti indossare al bambino degli occhiali coprenti l’occhio destro;
3. È stato fatto eseguire il test per l’occhio sinistro;
4. Sono stati fatti indossare al bambino degli occhiali coprenti l’occhio sinistro;
5. È stato fatto eseguire il test per l’occhio destro.

Ogni bambino quindi ha affrontato due test simili, uno per ogni occhio. Per agevolarlo durante il gioco si è cercato di metterlo a proprio agio nell’utilizzo di iPodSight e di aiutarlo nel caso non riuscisse a interagire con facilità.



(a)



(b)



(c)



(d)

Figura 6.2: Esecuzioni del test di screening

2.1 Risultati ottenuti

In figure 6.3 e 6.4 sono mostrati i risultati dei test effettuati su entrambi gli occhi. A seguito di questi risultati, è possibile fare delle considerazioni generali per apportate le migliorie necessarie.

2.2 Considerazioni sulla parte *serious*

I test effettuati hanno evidenziato che gli esercizi di crowding sono risultati molto difficili in generale. Il fatto che gli esercizi venissero molto spesso sbagliati anche a decimi molto bassi, cioè con immagini di grandi dimensioni, ha fatto pensare ad un problema nello strumento e non nella vista dei bambini. In un successivo consulto con la Dott.ssa Pinello è emerso che la distanza tra gli ottotipi mostrati era troppo bassa e ciò ha causato difficoltà nella distinzione. Tale miglioria è stata implementata successivamente. Test successivi alla correzione con soggetti esterni all'ambiente scolastico non hanno più presentato tale problema derivato da specifiche mediche iniziali poco chiare. Nel grafico in figura 6.4 si può infatti notare un calo della percentuale di risposte corrette per l'esercizio numero nove e in figura 6.5 si mostrano le percentuali di errore e mancata risposta per questo esercizio che, paradossalmente, risultano superiori a quelle dell'esercizio che valuta i 10/10.

Un altro punto degno di nota e che gli esercizi di rotazione con l'utilizzo di E orientate hanno causato incertezza nella distinzione tra il verso destro e sinistro, mentre lo stesso ottotipo mostrato nelle altre direzioni non ha rappresentato problemi seppur più piccolo. Tale problema ha afflitto soprattutto i bambini delle prime due classi portando a pensare essere un problema dovuto alla difficoltà di distinguere la destra dalla sinistra, tipico della giovane età, a cui quindi sembrano adattarsi meglio gli ottotipi di Lea.

D'altra parte, questi ottotipi sono utilizzabili solo con alcuni accorgimenti, perché le figure ottotipiche della mela e della casa sono state spesso scambiate rispettivamente per la figura del cerchio o del quadrato. Ciò è avvenuto quando sono state usate tali figure per valutare la visione superiore ai 6/10. Nel grafico in figura 6.4 si nota un drastico calo della percentuale di rispo-

Esercizio	Tipo	3° anno (5 anni, nati nel 2005)			2° anno (4 anni, nati nel 2006)			1° anno (3 anni, nati nel 2007)		
		corretti	sbagliati	saltati	corretti	sbagliati	saltati	corretti	sbagliati	saltati
1/10	acutezza	100,00%	0,00%	0,00%	100,00%	0,00%	0,00%	92,80%	7,14%	0,00%
2/10	crowding	100,00%	0,00%	0,00%	94,12%	5,88%	0,00%	85,71%	14,29%	0,00%
3/10	rotazione	93,75%	6,25%	0,00%	97,06%	2,94%	0,00%	64,29%	21,43%	14,29%
4/10	acutezza	93,75%	6,25%	0,00%	91,18%	5,88%	2,94%	71,43%	28,57%	0,00%
5/10	rotazione	100,00%	0,00%	0,00%	88,29%	14,71%	0,00%	92,88%	7,14%	0,00%
6/10	acutezza	62,50%	25,00%	12,50%	59,82%	29,41%	11,76%	7,14%	64,29%	28,57%
7/10	rotazione	75,00%	18,75%	6,25%	97,06%	2,94%	0,00%	78,57%	7,14%	14,29%
8/10	acutezza	68,75%	12,50%	18,75%	64,71%	14,71%	20,59%	50,00%	35,71%	14,29%
9/10	rotazione/crowding	31,25%	12,50%	56,25%	85,29%	5,88%	8,82%	28,57%	35,71%	35,71%
10/10	acutezza	81,25%	0,00%	18,75%	88,24%	5,88%	57,14%	14,29%	28,57%	46,15%
Cobre 1	cobre	87,50%	12,50%	0,00%	91,18%	8,82%	0,00%	85,71%	14,29%	7,14%
Cobre 2	cobre	87,50%	12,50%	0,00%	91,18%	8,82%	0,00%	78,57%	14,29%	100,00%

Figura 6.3: Risultati dei test effettuati su entrambi gli occhi

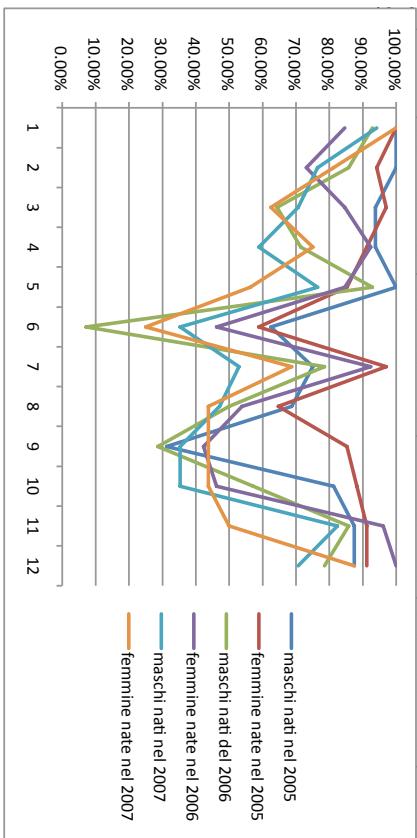


Figura 6.4: Percentuale di esercizi corretti per ogni insieme di bambini

ste corrette per l'esercizio numero sei, che presentava l'ottotipo della casa orientato a destra, spesso confuso con la figura del quadrato. In figura 6.5 si possono notare le percentuali di errore e mancata risposta per tale esercizio, nettamente superiori alla media, anche per decimi di acutezza superiori che dovrebbero risultare in un numero maggiore di errori. Si attribuisce la causa in parte alla risoluzione del monitor ma soprattutto alle forme degli ottotipi: le curve della mela possono infatti ricordare un cerchio e gli spigoli di tre lati della casa ricordare la figura del quadrato. In conclusione quindi, i test fatti sembrano suggerire di restringere i set di ottotipi da utilizzare alle E orientate e alle figure del quadrato e del cerchio per la valutazione di valori alti di acutezza visiva. L'utilizzo di ottotipi diversi dalle E orientate erano stati previsti inizialmente per enfatizzare l'elemento ludico, ma il coinvolgimento del paziente ottenuto è tale da poter restringere il set a ottotipi meno "divertenti".

In linea di principio, poter disporre di risoluzioni maggiori sul display esterno aiuterebbe il disegno più definito degli ottotipi. Inoltre, per quanto detto nella sezione 1.6.1 del capitolo 5, ogni ottotipo ha bisogno di almeno una griglia di 5x5 pixel per essere disegnato in maniera (almeno in linea teorica) sufficiente. Tenendo conto di eventuali interpolazioni effettuate dal framework Foundation sulle immagini degli ottotipi, maggiore è il numero di pixel a disposizione e maggiore sarà il dettaglio ottenuto. Il massimo output video dei device iPad e iPad2 mostra la risoluzione di 1024x768, ma la prossima versione dell'iPad supporterà molto probabilmente risoluzioni HD, eliminando quindi nel futuro questa limitazione.

Per testare problemi visivi legati al riconoscimento dei colori è necessario un test più mirato. È stato infatti verificato un caso di falso positivo, discusso successivamente, che è stato un chiaro segnale sull'inadeguatezza del test. Inoltre i problemi tecnologici già discussi (sez. 1.2, cap 5), ovvero la rappresentazione differente dei colori tra il display esterno e il display del device, la scarsa gradazione di tonalità e la luminosità dell'ambiente, non hanno facilitato la resa di tale esercizio.

Alcuni esercizi presentavano uno sfondo colorato visualizzato sul display esterno che è risultato in un contrasto non sufficiente con la figura dell'ottoti-

po, rendendone più difficile la visualizzazione. Sfondi chiari e molto luminosi, possibilmente bianchi, sono stati suggeriti della Dott.ssa Pinello e le modifiche sono state apportate al progetto.

Su iPodSight era prevista la possibile presenza dello stesso ottotipo mostrato in direzioni diverse per le varie scelte. Ad esempio, se veniva mostrata una mela sul display, le possibili scelte generate sul dispositivo in mano al bambino prevedevano sia lo stesso ottotipo con orientazioni differenti sia gli altri ottotipi³. In particolare bambini piccoli, senza difetti visivi rilevati, hanno avuto qualche difficoltà a capire la dinamica del gioco, perché tendevano a cercare il simbolo senza curarsi del giusto orientamento. riconoscere il simbolo, ma hanno avuto molte difficoltà a riconoscere la giusta orientazione se l'ottotipo veniva mostrato più volte. Tale scelta randomica è stata rimossa proponendo solo i quattro ottotipi di Lea per gli esercizi di crowding e di acutezza a seguito dell'esperienza avuta in scuola, scelta confermata anche dalla Dott.ssa Pinello.

Esercizio	Tipo	Totalità					
		maschi			femmine		
		corretti	sbagliati	saltati	corretti	sbagliati	saltati
1/10	acutezza	95,66%	4,34%	0,00%	94,87%	5,13%	0,00%
2/10	crowding	87,39%	10,64%	1,96%	82,81%	13,02%	4,17%
3/10	rotazione	76,21%	17,07%	6,72%	81,39%	11,88%	6,73%
4/10	acutezza	74,67%	23,37%	1,96%	86,16%	8,69%	5,15%
5/10	rotazione	89,78%	8,26%	1,96%	75,39%	20,45%	4,17%
6/10	acutezza	34,98%	45,45%	19,57%	43,33%	27,43%	29,24%
7/10	rotazione	68,84%	18,43%	12,73%	86,04%	6,43%	7,53%
8/10	acutezza	55,27%	23,91%	20,82%	54,10%	17,08%	28,82%
9/10	rotazione/crowding	31,71%	19,99%	48,30%	57,12%	9,17%	33,71%
10/10	acutezza	57,90%	10,64%	31,46%	59,38%	11,26%	29,36%
Colore 1	colore	85,19%	12,85%	1,96%	79,11%	16,72%	4,17%
Colore 2	colore	78,89%	10,89%	10,22%	92,89%	2,94%	4,17%

Figura 6.5: Risultati dei test effettuati sulla totalità dei bambini

³Esercizi con utilizzo degli ottotipi di Lea

2.3 Considerazioni sulla parte *game*

Per ogni bambino è stato segnato il grado di divertimento durante il gioco chiedendogli verbalmente se il gioco era piaciuto e osservandolo durante l'utilizzo dello strumento. A parte un paio di casi isolati, il gioco ha riscosso un grande successo e tutti i bambini hanno detto che si sono divertiti molto. La complessità del gioco è risultata quindi adeguata per l'età dei bambini.

L'età ha influito invece sulle capacità di interazione in maniera abbastanza evidente: i bambini del secondo e terzo anno della scuola non hanno avuto difficoltà a giocare con la modalità d'interazione di tipo dragging, mentre i bambini del primo anno hanno dovuto usare la modalità touch per difficoltà nel tenere in mano o nel tenere fermo appoggiato sul tavolo il device.

Talvolta la collaborazione del bambino è stata molto buona e questo iniziava a giocare senza aver bisogno di ulteriori spiegazioni. Più spesso invece, soprattutto per i bambini più giovani, è stato necessario rispiegare le regole del gioco o eseguire un esercizio per metterlo a proprio agio nell'ambiente e non intimorirlo.

Un bambino ha espresso il suo stupore nel vedere che l'interazione con il device comportava l'avanzamento dell'esercizio sul monitor senza che vi fosse un collegamento fisico. L'assenza di cablaggio è stata sicuramente una facilitazione d'uso.

Come ultima nota finale, bambini che avevano già fatto visite oculistiche hanno espresso una forte preferenza per questa modalità di indagine rispetto a quella tradizionale.

2.4 Tempi di esecuzione

Ogni test di screening è stato misurato anche in termini di tempo. In figura 6.6 vengono mostrati i tempi medi dell'esecuzione di un test. I tempi per la valutazione dell'occhio sinistro sono superiori in quanto è stato il primo occhio valutato e il bambino ha impiegato più tempo per capire come approcciarsi allo strumento. In genere, il test successivo sull'occhio destro ha visto tempi di esecuzione più brevi in quanto il meccanismo di gioco era stato appreso. Per i bambini del primo anno è apparso subito essere troppo complicata la modalità di gioco tramite dragging ed è stata utilizzata quella touch che richiede sicuramente meno tempo.

Con tali dati è possibile notare che è sufficiente un tempo medio inferiore agli 8 minuti per effettuare lo screening su entrambi gli occhi. Una visita ortottica dura in genere dai 15 ai 20 minuti con la presenza dei genitori che aiutano a mantenere la collaborazione dei bambini. Le maestre della scuola hanno riportato che altre iniziative simili svolte nella scuola, come lo screening audiologico, sono durate mesi, interferendo negativamente con l'attività scolastica.

	1° anno (3 anni)*	2° anno (4 anni)	3° anno (5 anni)
Primo occhio esaminato	4'27	5'13	3'42
Secondo occhio esaminato	3'02	3'06	3'15
Totale	7'29	8'19	6'57

* test in modalità touch

Figura 6.6: Tempi per l'esecuzione dello screening

2.5 Casi particolari

La scuola è stata messa al corrente a priori che il test non ha nessun valore medico e per legge i risultati ottenuti non possono essere né divulgati né resi noti ai genitori dei bambini.

Sono stati riscontrati però tre casi particolari degni di nota. Due bambini hanno riconosciuto tutti (o quasi) i simboli con un occhio, mentre con l'altro occhio hanno avuto difficoltà a riconoscere ottotipi con valori superiori ai 3/10. Eseguendo il test più volte per conferma, è apparso chiaro che si era in presenza di ambliopia, anche per il fatto che i bambini cercavano in continuazione di togliere l'occhiale che copriva l'occhio dominante.

Si fa presente che altri due bambini che hanno effettuato lo screening erano già stati curati per ambliopia e i genitori hanno confermato, in via del tutto uffiosa, che i valori di acutezza visiva visti dal bambino durante la visita medica specialistica sono gli stessi di quelli evidenziati durante lo screening.

Un altro caso particolare è rappresentato però da un falso positivo. Un bambino non ha avuto problemi di acutezza visiva ma ha sbagliato i test sui colori pur ripetendolo, per altro rispondendo in maniera molto sicura. Utilizzando le tavole di Ishihara e cercando la collaborazione del bambino a fine test, è apparso però non avere problemi di daltonismo. Per la presenza di questo

caso, gli esercizi sui colori sembrano non essere mirati come precedentemente detto nella sezione 2.2.

Si può quindi concludere che, mentre per il test dell'acutezza visiva lo strumento ha raggiunto ottimi risultati e ha ottenuto l'approvazione di alcuni specialisti, la valutazione di eventuali disfunzioni cromatiche deve essere ulteriormente approfondito.

Capitolo 7

Conclusioni

In questa tesi è stato presentato lo strumento “Play With Eyes”, un serious game utilizzabile sia per effettuare uno screening visivo a partire dall’età prescolare sia per essere usato nelle visite oculistiche specialistiche. Lo strumento è pensato per essere usato nelle scuole dell’infanzia da parte di personale non medico, abbattendo i costi ma allo stesso tempo consentendo uno screening che permetta di rilevare problemi evidenti legati all’apparato visivo, in particolare l’ambliopia.

Lo strumento presenta un’architettura hardware il più minimale possibile: un iPad per la gestione e configurazione dei test e degli utenti, un iPod Touch o iPhone per mostrare il serious game e un display per la raffigurazione degli ottotipi. Alcune problematiche sono state affrontate durante lo sviluppo del software, tra le altre la gestione di connessione Bluetooth in un’architettura client/server, la non immediata visualizzazione su un display esterno, il disegno su schermo di figure con grandezze fisiche specifiche, la gestione di un database e la gestione di interazioni touch.

Il gioco prevede che il bambino utilizzi il dispositivo smartphone e risponda tramite operazioni touch (le più naturali possibili per un bambino) in base a ciò che vede sul display posizionato alla corretta distanza, similmente a quanto avviene durante una visita oculistica. La distanza di esecuzione del test, la grandezza del display disponibile e la modalità di gioco sono configurabili conferendo di fatto dinamicità allo strumento. I risultati dello screening, che contengono informazioni utili riguardo la visione del bambino, vengano salvati per successive analisi.

Lo sviluppo di questa tesi ha richiesto lo studio di tematiche riguardanti

l’ottica e l’oculistica, sia per capire le basi teoriche riguardanti il calcolo di specifiche misure, sia per capire quali sono gli strumenti tipici utilizzati dall’oculista e poterli adattare adeguatamente per soddisfare le specifiche esigenze. Uno studio continuo dell’usabilità del software con utenti reali inoltre ha permesso di soddisfare più consciamente alcune necessità.

Lo sviluppo ha previsto due fasi di studio preventive: la prima dedicata al framework cross-platform Titanium per valutarne le caratteristiche e l’effettiva possibilità di utilizzo, la seconda dedicata all’ambiente di sviluppo di Apple. L’analisi di Titanium ha evidenziato mancanze dovute alla poca maturità del framework che ne hanno reso impossibile l’utilizzo per lo scopo del progetto; la scelta infatti è ricaduta sulla seconda opzione per la facilità di utilizzo dei dispositivi da parte dell’utente finale, e per la non completa maturità della tecnologia che permette uno sviluppo cross-platform.

Lo strumento è stato testato presso la Scuola dell’Infanzia “Gianni Rodari” di Mogliano Veneto, effettuando test su un totale di 65 bambini. I risultati sono stati positivi sia per l’aspetto ludico della parte di gioco, sia per i risultati derivati dallo screening.

Il progetto ha visto la collaborazione del Dipartimento di Pediatria Salus Pueri dell’Università degli Studi di Padova riscontrando interesse particolare da parte della Dott.ssa Luisa Pinello e della Dott.ssa Elisabetta Zannin.

1 Risultati ottenuti

Malgrado conoscenze nulle in campo oculistico ci si è approcciati all’argomento studiando il dominio applicativo necessario per lo sviluppo del progetto. Lo strumento realizzato ha riscosso approvazioni su diversi fronti: in primis è stato apprezzato dai medici che lo hanno visionato, in secundis la parte ludica ha coinvolto i bambini durante i test.

L’iniziativa di screening svolta in scuola ha riscosso molto successo da parte di genitori e maestre e ha permesso l’individuazione di due casi di presunta ambliopia sottolineando la reale ed effettiva utilità dello strumento.

Durante lo sviluppo dello strumento sarebbe stato desiderabile avere maggior coinvolgimento di medici oculisti o comunque figure professionali capaci di dare un reale contributo al progetto sia in termini di analisi dei requisiti

iniziale sia di feedback a seguito di test del prodotto. Purtroppo ottenere tale collaborazione verbale è complicato da parte del personale medico in quanto poco prono all'utilizzo di strumenti informatici per la medicina; forse anche per questo motivo la nascita di questi ultimi è spesso meno agevole. Infine, parte del lavoro presentato in questa tesi è stato presentato e accettato su la IEEE 1st International Conference on Serious Games for Health, SeGAH 2011 [21]. L'articolo *"PlayWithEyes: a new way to test children eyes"*, è allegato in appendice.

2 Sviluppi futuri

Lo strumento sviluppato non rappresenta un punto di arrivo, ma bensì un punto di partenza molto interessante nel campo dei serious game per l'health-care. Estensioni ulteriori di *Play With Eyes*, tramite l'aggiunta di funzionalità e esercizi visivi, apporterebbero valore aggiunto al software; per permettere ciò il codice è stato reso open source.

In particolare, è attualmente allo studio, in collaborazione con il Dipartimento di Pediatria Salus Pueri, la possibilità di utilizzare questo strumento su bambini affetti da Cerebral Visual Impairment (in acronimo CVI, disfunzione visiva dovuta a problemi cerebrali e non all'apparato visivo) allo scopo di riabilitare le loro capacità di fissazione dell'attenzione. *Play With Eyes*, presentandosi sotto forma di gioco, si presta molto bene per esercizi riabilitativi che devono essere ripetuti più volte e che, senza un paradigma di gioco, rischiano di diventare noiosi e di essere abbandonati dai bambini sotto cura. Sarebbe particolarmente indicata l'implementazione, supervisionata da un oculista, di un'intera gamma di esercizi visivi ben definitivi. Gli obiettivi più auspicabili ai quali mirare sono i seguenti: la certificazione da parte di un medico, l'utilizzo effettivo presso enti pubblici e la commercializzazione.

Il primo obiettivo darebbe allo strumento un riconoscimento tangibile in ambiente medico. Il secondo porterebbe all'utilizzo pratico di quanto sviluppato e i conseguenti vantaggi sociali descritti nel corso della tesi. La commercializzazione del prodotto potrebbe avvenire a seguito della realizzazione dei precedenti obiettivi.

Appendice A

SeGAH 2011 Short Paper

PlayWithEyes: a new way to test children eyes

Short Paper

Alberto De Bortoli and Ombretta Gaggi

Dept. of Pure and Applied Mathematics, University of Padua
via Trieste, 63, 35121 Padua, Italy
adeborto@studenti.math.unipd.it, gaggi@math.unipd.it

Abstract—One of the major problems of doctors when dealing with children is to capture and maintain their attention. This is particularly true for oculists that must test children eyes, especially if they are very young. In this paper we present *PlayWithEyes*, a serious game that aims at testing the children eyes while they are having fun playing with Lea symbols and images taken from popular cartoons, using a touch interface. *PlayWithEyes* was created to perform screening of visual acuity of very young children, since early diagnosis is very important for some sight defects like amblyopia (*lazy eye*).

Index Terms—serious games; game-based diagnosis; assistive technologies; multimedia applications; mobile devices;

I. INTRODUCTION

No doctor can perform a good diagnosis without the patient's collaboration. The problem is that doctors cannot always take for granted patients' cooperation, especially when dealing with children. One of the major problems, in this case, is to capture and maintain their attention: children cannot be considered like little adults. This is particularly true for oculists that have to test children eyes, asking them to look at some projected letters and to recognize their orientation. This task is so boring for children that sometimes, especially for very young children, their responses become wrong, or not accurate, since they are not longer interested in the task. There exists some eye diseases for which an early diagnosis is very important, *amblyopia*, commonly known as *lazy eye*, is one of them.

Amblyopia, is the eye condition denoted by reduced vision not correctable by glasses or contact lenses which is not due to any eye disease. The brain, for some reason, does not fully acknowledge the images seen by the amblyopic eye. It is estimated that three percent of children under six have some form of amblyopia. Moreover, amblyopia causes more visual loss in the under 40 group than all the injuries and diseases combined in this age group [1]. One of the major problems of this eye disease is its diagnosis, since the patient usually does not understand to have a visual disease since he/she is not aware that he/she is using only one eye.

Treatment for amblyopia begins as soon after diagnosis as possible and an early treatment usually can reverse the condition. Treatment is best started before age 6 and should begin before child's vision has fully developed, which is around age 9 or 10, after amblyopia can be hard to correct. The younger the child is when treatment begins, the better his

or her chances for having good vision are. Some studies [2], [3] have shown that better results can be reached with earlier diagnosis.

Children who received orthoptic screening for amblyopia between the ages of 8 and 37 months have better outcomes from their treatment at 7.5 years than children who are screened only at 37 months. Williams et al [2] randomised 3490 children to receive either intensive or once only orthoptic screening. They found that those children given intensive screening had a lower prevalence of amblyopia and better visual acuity in the worse seeing eye, supporting the hypothesis that early treatment leads to better outcomes.

For this reason it is very important to allow orthoptic screening for very young children.

In this paper we present *PlayWithEyes*, an application which aims at testing children eyes through the use of a game. We ask to preschool children to perform a Lea vision acuity test [4] by recognizing symbols projected on a wall and pointing them in a touch interface, i.e., an Apple iPod Touch, which displays the Lea symbols. In order to obtain their collaboration, we use the *serious game paradigm*, so that children can have fun playing a game without understanding that they are testing their eyes. Children are engaged and encouraged with audio message, e.g., an applause in case of correct answer. Images taken from popular cartoons are projected together with Lea symbols.

The children can choose and touch a symbol on the interface or change the orientation of the device to match the orientation of the symbol. This possibility provides a very friendly interaction, even for younger patients. Moreover, the use of the serious game paradigm helps to improve the diagnosis of very young children, because the eye test may last longer since they have fun, allowing the doctor to better observe the little patient.

PlayWithEyes allows to test not only visual acuity but also color blindness.

II. RELATED WORKS

The “*serious game*” paradigm aims to engage users into an activity, usually that produces a common good, concealing it into a game designed for a primary purpose other than pure entertainment [5]. The idea is that the user does something useful, which normally would not, because he/she enjoys doing it. Serious games can be used to develop new knowledge

or skills. An example is Google Image Labeler [6], a game that challenges two users to find a common keyword to tag an image, i. e., a clever way for Google to ensure that its keywords are matched to correct images. This operation is not possible without human intervention.

Recently the serious game paradigm begins to be used in health care. In [7], Kato explores the positive aspects of using existing commercial video games for health improvements or surgical training, and tailor-made game for particular disease group in order to improve recovery and rehabilitation of patients.

Since our system is specifically intended for preschool children, we must pay particular attention to the interface. Many papers address the problem of interface for children. Among others, [8] describes a game which allows preschool children to access information. Children use magic objects to query and navigate an image database. Forlines et al [9] investigate the differences between mouse and direct touch input, both in terms of quantitative performance and subjective preference. They conclude that touch interfaces, even if they may not lead to greater performance, especially for speed and accuracy, are preferable for other considerations like fatigue, spatial memory and simplicity.

A. Existing Software

A certain number of software products that provide vision tests exist, both for professional and personal use. At the state of the art the main professional solutions that can be adopted by ophthalmologists are listen below:

- 20/20 Vision (Canela Software) [10]
- PVVAT™ (Precision Vision) [11]

Both products are desktop applications for professional use. The software shows the tools used by oculists (eye charts, Snellen tables, etc.) on the desired screen to test patients. The screen can be either a computer monitor or an external display. Both products do not focus on testing children patients specifically, even though Lea optotypes can be used. Moreover, these solutions do not use a game paradigm to improve users' attention.

Given the proliferation of mobile phones, another set of interesting software solutions are applications that allow users to self test their visual acuity with a cell phone or a portable device. The relevant applications are:

- Acuity (Intelllicore)
- EyeChart HD (Dok LLC)
- Vistion Test (3 sides cube)
- FastAcuity Lite (KyberVision Consulting, RD)
- Vision (AppZap)
- Eyetest (Konrad Feiler)

All the listed applications for iPhone, iPod Touch or iPad, behave almost in the same way with non perceptible differences. They propose daltonism tests based on Snellen tables and acuity tests based on eye charts and Landolt rings. Applications like these, often developed with no strong specifications, provide nothing more than simple self-made

tests with no real implications. None of these applications considers children specific requirements or makes use of a game to test eye, therefore they do not solve the problem to maintain children attention.

III. DESCRIPTION OF THE GAME “PLAYWITHEYES”

PlayWithEyes is a serious game which can be used both in kinder gardens and doctor's surgeries with different implications. In the first case, the game uses a set of preconfigured tests, studied by a group of oculists, so the teacher can easily test children simply by inserting children names. Children can be inserted in the database and classroom can be created before the test, in order to lower the waiting time between a child and the next one. This is particular important because we do not want the children get impatient while waiting. Each child is tested individually and the report, obtained while playing, indicates if the child need a specific visual examination by an oculist.

If used by a doctor, our system is just a facility for the oculist to obtain a better cooperation from the children, because as long as they have fun, they continue playing, thus testing their visual acuity. This helps to obtain a more precise sight examination, since the eye test may last longer, thus improving the possibility to perform a correct diagnosis for very young children.

Visual acuity is tested through standard Lea tests [4]. Lea symbols are specifically defined for preschool children because of their simplicity: children are able to refer to them without ambiguity even if they cannot read letters. The original¹ symbols are four: a house, an apple, a square and a circle. Images taken from popular cartoons are used to maintain the children concentration on Lea symbols. For example, the game shows in sequence apples and circles. The child must recognize the symbol and put the apples on a basket in the hands of Winnie The Pooh, and the circle in a basketball basket. To improve entertainment, some non misleading animations are shown on the wall.

A very simple interaction paradigm is provided: children give their answer by touching the right symbol on the device, or changing the device's orientation in order to match the orientation of the symbol.

Daltonism tests show cartoon characters with parts of the body colored differently from each other in order to test the children ability to recognize the right color. The child must choose the correct color on a palette depicted on his/her device and touch it.

The serious game on *iPodVision* uses accelerometer feature, animations and sounds, e. g., an applause in case of correct answer, to enhance the game experience. The game is intentionally left skinny to avoid any kind of negative interferences with the examination during the game play. Additional kinds of exercise will be implemented according to specific ophthalmology advices.

¹Some minor variants with additional optotypes are also used in ophthalmology.

IV. SYSTEM ARCHITECTURE AND DEVELOPMENT

Before presenting the architecture, some considerations on the chosen devices are discussed. The actors involved in the usage of the product are kinder garden teachers, oculists and children. None of them are supposed to have any special knowledge about the use of computers, therefore an important target to achieve is to provide a simple and easy-to-use product with a friendly interface for final users.

To this end, some aspects play a significant role, like the preference of easy-to-use devices with minimal configurations, the rising market of smart phones and tablets and the preference of interaction with touch interfaces among traditional ones. The first aspect is desirable in order to avoid problems during the use of *PlayWithEyes*; the third one is much more important since we have to deal with non-expert users and with children that do not understand non trivial input interfaces. According to discussion in Section II, we choose a touch interface, since we think it is best suited for children, that are naturally inclined to interact with the outside world through touch. For this reason, we use an Apple® iPad for the device in use by the teacher and at least one iPod Touch or iPhone for the device given to the children. Moreover, a projector is also needed to execute the vision screening tests.

PlayWithEyes has a classic server-client behavior, based on the following components:

Server: used by the teacher to configure the vision tests.

The server runs on an iPad. We call this component “*iPadVision*”.

Client: used by the children to play the game, i. e., to perform the preconfigured tests. The client may be execute on an iPod Touch or an iPhone. We call this component “*iPodVision*”.

Both server and client make use of the typical gesture of touch and tablet devices. Using an iPad as a server is not a common choice, but it represents the best solution in our scenario. In fact, the server is used both to project the eye test on a wall, and to interact with the doctor or teacher to select tests or create new ones (in the case of doctor), or to input children data (name, surname, age, etc.). Given the choice to use a touch interface in order to keep the interaction easy and simple, the next goal to reach was to minimize the needed hardware components. For this reason, the entire server resides on the iPad, therefore its development has taken into account efficiency issues since it should be used in a scenario of low resources in term of CPU, network bandwidth and disk space. Figure 1 shows the system architecture.

It could be noticed that an iPad is not usually present in doctors surgeries or in kinder gardens, while the presence of a personal computer is certainly more likely, especially in kinder gardens. Even if market analysis ([12]) have shown that the market share for smartphones and tablets increased over the last year, so these devices are rapidly becoming common available, to support also this situation, we implement also a web application running on a traditional computer as alternative to the *iPadVision* which can be used when the cost

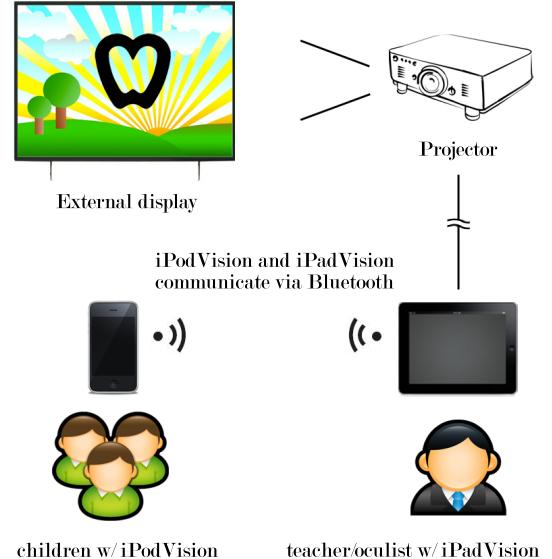


Fig. 1. Main System Architecture of *PlayWithEyes*

of an iPad is not affordable. In this case of the usability of the tool degrades a little, but not its accessibility, since the eye tests can still be performed.

Given the described architecture, the following requirements were reached:

- 1) *iPadVision* must manage users and tests;
- 2) *iPadVision* and *iPodVision* must adopt the Bluetooth® wireless protocol to communicate with each other;
- 3) *iPadVision* must connect to a projector and manage what to display on the external screen;
- 4) *iPodVision* is the interface of the serious games for kids that must be easy-to-use and funny.

The first requirement allows to create and edit users, classroom and tests, but it is also necessary to keep track of the screening results in order to transmit data to oculist for diagnosis, in the case of screening in a kinder garden, and to permit statistic analysis. The Bluetooth protocol (requirement #2) allows the system to work even in absence of a wireless network: once more, we minimize the system requirements in term of hardware components. Moreover, the development of the system has shown that Apple devices allow Bluetooth communications in a very simple way.

During eye test, children have to respond interacting with the *iPodVision*, in accordance to what they are able to see projected on a wall at a given distance. The distance may vary between different executions of tests in different places, depending on the dimensions of the projected image on the wall and the projector settings. The system must help the teacher to configure itself, asking the teacher the distance from the wall and the projector settings and automatically projecting the tests in the correct size.

A. Technology details

The entire project relies on Apple products, therefore the software has been developed on iOS operating system. This choice has been made after considering many factors. At the beginning, we tried to abstract from the actual platform using a framework which allows us to develop an application using web technologies (HTML, CSS, Javascript, etc.), and to translate it into device native language (Objective-C for Apple or Java for Android). Titanium framework [13] has been tested with a basic prototype, but the experience made has shown that this framework is not mature and it's not suitable for our purpose, since Bluetooth connectivity and display on external screen are not supported yet. Moreover, the same problem affects also Android devices, for which a standard way to manage external screens does not exist. For these reasons, using the Apple SDK for developing the software appeared to be the right way to go through, and *PlayWithEyes* completely relies on Apple products.

Recent versions of the Cocoa framework adopted by Apple expose an API that fulfills the previously listed requirements. The communication is supplied by the *GameKit* framework, whose main focus is to wrap the well-known *Bonjour* implementation of the *Zeroconf* protocol in order to establish a communication channel between two devices without asking users any configuration parameters. The *UIScreen* class provided by Cocoa in the *UIKit* framework enables the possibility to handle an external screen and to decide what to display on it.

V. CONCLUSION

In this paper we have presented *PlayWithEyes*, a serious game designed to allow eyes test for very young children. *PlayWithEyes* can be used for screening children in kinder gardens, aged from 3 to 6 years old, but it can be used also in doctor's surgeries with younger children. The use of the serious game paradigm helps to obtain their cooperation because they have fun, so the eye test may last longer, thus improving the possibility to perform a correct diagnosis: this is particularly important for some sight defects like amblyopia.

PlayWithEyes was provide a touch interface both for children and teachers/doctors since the use of this technology improves the accessibility and usability of interfaces for very young, not expert, users. At the time of writing, only a little tests set has been done successfully, but our system will be deeply tested in three kinder gardens by about 200 children and 18 teachers in May, 2011. Moreover, the gameplay of the system is currently under evaluation of a group of experts in children care and ophthalmology.

Future works will be devoted to include new type of tests, i. e., new games, to improve the user interface and to better investigate the adaptivity of the system to the set of available resources (e. g., in the case of the server which runs as a web application).

ACKNOWLEDGMENT

Partial financial support for this work is provided by the University of Padua through its research funding program.

REFERENCES

- [1] J. Cooper and R. Cooper, "All about amblyopia (lazy eye)," Optometrists Network. Strabismus., Tech. Rep., 2010.
- [2] C. Williams, K. Northstone, R. Harrad, J. Sparrow, and I. Harvey, "Amblyopia treatment outcomes after screening before or at age 3 years: follow up from randomised trial," *BMJ*, vol. 324, pp. 1549–1551, June 2002.
- [3] R. Chou, T. Dana, and C. Bougatsos, "Screening for Visual Impairment in Children Ages 15 Years: Update for the USPST," *Pediatrics*, vol. 127, no. 2, pp. 442–479, 2011.
- [4] Lea Test Ltd, "Lea Vision Test System," <http://www.lea-test.fi/index.html>.
- [5] D. R. Michael and S. L. Chen, *Serious Games: Games That Educate, Train, and Inform*. Muska & Lipman/Premier-Trade, 2005.
- [6] Google, Inc., "Google Image Labeler," <http://images.google.com/imagelabeler/>.
- [7] P. M. Kato, "Video Games in Health Care: Clogging the Gap," *Review of General Psychology*, vol. 14, no. 2, pp. 113–121, 2010.
- [8] F. Pittarello and R. Stecca, "Querying and Navigating a Database of Images With the Magical Objects of the Wizard Zurlino," in *9th international Conference on interaction Design and Children*, 2010, pp. 250–253.
- [9] C. Forlines, D. Wigdor, C. Shen, and R. Balakrishnan, "Direct-Touch vs. Mouse Input for Tabletop Displays," in *Proceedings of ACM CHI 2007 Conference on Human Factors in Computing Systems*, May 2007, pp. 647–656.
- [10] Canelas Software, "20/20 Vision," <http://canelasoftware.com>.
- [11] Precision Vision, "PVVAT," <http://www.precision-vision.com/index.cfm/feature/20/pvvat.cfm>.
- [12] PHYSOrg.com, "Mobile sales up 32 percent last year: Gartner," <http://www.physorg.com/news/2011-02-mobile-sales-percent-year-gartner.html>, February 2011.
- [13] Appcelerator, "Titanium framework," <http://www.appcelerator.com>.

Appendice B

Glossario

A

Acutezza visiva: o acuità visiva o visus è una delle abilità visive principali del sistema visivo. È la capacità dell'occhio di risolvere e percepire dettagli fini di un oggetto e dipende direttamente dalla nitidezza dell'immagine proiettata sulla retina;

Agile: metodologia leggera per lo sviluppo del software che coinvolge quanto più possibile il committente, ottenendo in tal modo una elevata reattività alle sue richieste;

Ambliopia: in oftalmologia, è un'alterazione della visione dello spazio che viene a manifestarsi inizialmente durante i primi anni di vita. Il termine deriva dal greco e, più esattamente, da ops (che significa visione) e -amblyos (che significa ottusa, pigra). È volgarmente nota con il nome “sindrome dell'occhio pigro”;

API: acronimo di “Application Programming Interface”, insieme di procedure disponibili al programmatore, di solito raggruppate a formare un set di strumenti specifici per un determinato compito. È un metodo per ottenere astrazione tra software a basso ed alto livello;

B

Beta: versione di prova di un software non definitivo che viene messo a disposizione di utenti non necessariamente sviluppatori, confidando nelle loro azioni imprevedibili che potrebbero portare alla luce bug o incompatibilità del software;

Beta testing: fase di prova e collaudo di un software non ancora pubblicato, con lo scopo di trovare eventuali errori;

Bluetooth: specifica industriale per reti personali senza fili. Fornisce un metodo standard, economico e sicuro per scambiare informazioni tra dispositivi diversi attraverso una frequenza radio sicura a corto raggio;

D

Daltonismo: difetto di natura prevalentemente genetica che altera in diverse modalità la percezione dei colori. Può insorgere anche in seguito a danni agli occhi, ai nervi o al cervello e persino in seguito all'esposizione ad alcuni composti chimici;

DataBase: insieme di dati riguardanti uno o più argomenti correlati tra loro in maniera strutturata in modo da permetterne l'uso da parte di applicazioni software;

G

GUI: acronimo di “Graphical User Interface”, paradigma di sviluppo che mira a consentire all'utente di interagire col computer manipolando graficamente degli oggetti, svincolandolo dall'obbligo di imparare una sintassi come invece avviene con le interfacce testuali command line interface;

I

IDE: acronimo di “Integrated Development Environment”, ambiente integrato per lo sviluppo software. In questo documento viene riferito a Xcode;

iOS: sistema operativo mobile sviluppato da Apple Inc. per iPhone, iPod Touch e iPad. La prima versione è stata rilasciata nel giugno 2007;

J

JSON: acronimo di “JavaScript Object Notation”, formato adatto per lo scambio dei dati in applicazioni client-server descritto nell’IETF RFC 4627;

O

Ottotipo: rappresentazione grafica di lettere o simboli, utilizzati per determinare l’acutezza visiva. La nascita dell’ottotipo si deve al fisico tedesco Hermann von Helmholtz, che stabilì che la dimensione delle figure rappresentate dovessero sottendere alla grandezza di un minuto d’arco;

R

RFID: tecnologia per l’identificazione automatica di oggetti, animali o persone basata sulla capacità di memorizzare e accedere a distanza ai dati usando dispositivi elettronici;

S

SDK: acronimo di “Software Development Kit”, insieme di strumenti per lo sviluppo e documentazione di software;

Serious game : applicazioni videoludiche in grado di trasmettere al giocatore nozioni o messaggi durante l'atto di gioco, di sviluppare le sue capacità, di incrementandone la motivazione facilitando l'apprendimento per mezzo del contesto significativo e piacevole in cui le informazioni vengono presentate. Sono giochi il cui fine ultimo non è l'intrattenimento fine a se stesso;

Screening : strategia di indagine diagnostica generalizzata. Viene utilizzata per identificare una patologia in una popolazione standard con un rischio medio di malattia sufficientemente elevato da giustificare la spesa e lo stress di cercarla;

Sprint : blocco di lavoro rapido alla fine del quale avvengono la consegna di una versione del software e un meeting del team di sviluppo per definire i dettagli del lavoro da svolgere nell'immediato futuro;

T

TDD: metodo usato per sviluppare l'architettura del software in maniera incrementale, guidata dai test;

Titanium: framework per lo sviluppo di applicazioni cross-platform. La versione mobile utilizza il linguaggio web JavaScript;

X

Xcode: ambiente di sviluppo integrato sviluppato da Apple Inc. per agevolare lo sviluppo di software per i sistemi operativi Mac OS X e iOS;

XML: acronimo di eXtensible Markup Language, metalinguaggio di markup che definisce un meccanismo sintattico che consente di estendere o controllare il significato di altri linguaggi marcatori;

Bibliografia

- [1] Agile Alliance. <http://www.agilealliance.org>, accessed 2011.
- [2] Agile Alliance. <http://www.agilealliance.org/the-alliance/the-agile-manifesto/>, accessed 2011.
- [3] Apple, Inc. *The Objective-C Programming Language*, 2010.
- [4] Apple, Inc. *Game Kit Programming Guide ('Peer-to-Peer Connectivity' and 'Finding Peers with Peer Picker' sections)*, 03 2011.
- [5] Apple, Inc. Apple Developer Portal. <http://developer.apple.com>, accessed 2011.
- [6] Azienda U.L.S.S. n°1 Belluno, Unità Locale Socio Sanitaria. *Progetto di prevenzione dell'ambliopia nel distretto del Cadore*. <http://www.ulss.belluno.it/index.php?pagina=sezione&cat=10&vedi=102>, accessed 2011.
- [7] Camera dei Deputati. Atto Camera, Interrogazione a risposta scritta 4/03632. http://banchedati.camera.it/sindacatoispettivo_15/showXhtml.asp?highLight=0&idAtto=11891&stile=6, accessed 2011.
- [8] Canela Software. 20/20 Vision. <http://canelasoftware.com>, accessed 2011.
- [9] Chris Lattner. Clang Static Analyzer, LLVM Project. <http://clang-analyzer.llvm.org/>, accessed 2011.
- [10] CodeBaseHQ. <http://www.codebasehq.com/>, accessed 2011.

- [11] Dennis M. Levi, Stanley A. Klein and A. P. Aitsebaomo. *Vernier acuity, crowding and cortical magnification*. *Vision Res*, September 1984. University of Houston, University Park, College of Optometry, 4901 Calhoun Boulevard, Houston, TX 77004, U.S.A.
- [12] Dario Deponti, Dario Maggiorini, and Claudio E. Palazzi. *DroidGlove: An android-based application for wrist rehabilitation*. In *ICUMT*, 2009.
- [13] ESSEGI. Portale sui serious games. <http://www.seriousgames.it/>, accessed 2011.
- [14] C. Forlines, D. Wigdor, C. Shen, and R. Balakrishnan. *Direct-Touch vs. Mouse Input for Tabletop Displays*. In *Proceedings of ACM CHI 2007 Conference on Human Factors in Computing Systems*, pages 647–656, May 2007.
- [15] Francesca Manzotti, Jelka G. Orsoni. *Lo screening per l'ambliopia: risultati e problemi da risolvere: l'esperienza di Parma dopo 21 anni di attività*. http://www.aipam.it/ambliopia_oculista_02.pdf, accessed 2011. Sezione di Oftalmologia del Dipartimento di Scienze Otorino-Odonto-Oftalmologiche e Cervico-Facciali dell'Università di Parma.
- [16] GitHub. <http://www.github.com/>, accessed 2011.
- [17] Google, Inc. Android documentation. <http://developer.android.com/>, accessed 2011.
- [18] Google, Inc. Google Image Labeler.
<http://images.google.com/imagelabeler/>, accessed 2011.
- [19] I-maginary. Portale sui serious games. <http://www.i-maginary.it/>, accessed 2011.
- [20] IAPB International Agency for the Prevention of Blindness. <http://www.iapb.it/news2.php?id=676&ia=40>, accessed 2011.
- [21] IPCA - Instituto Politécnico do Cávado e do Ave. SeGAH, IEEE 1st International Conference on Serious Games and Applications for Health. <http://www.ipca.pt/segah2011>, accessed 2011.

- [22] Ali Karime, M. Hossain, A. Rahman, Wail Gueaieb, Jihad Alja'am, and Abdulmotaleb El Saddik. *RFID-based interactive multimedia system for the children*. *Multimedia Tools and Applications*, accessed 2011.
- [23] Pamela M. Kato. *Video Games in Health Care: Closing the Gap. Review of General Psychology*, 14(2):113–121, 2010.
- [24] Lea Test Ltd. Lea Vision Test System.
<http://www.lea-test.fi>, accessed 2011.
- [25] Lea Test Ltd. Lea Vision Test System, sezione “Color game”.
<http://www.lea-test.fi/games/color/pv3.html>, accessed 2011.
- [26] Lions Club. *Screening per il controllo delle capacità Visive nelle scuole dell’Isola Bergamasca*. <http://www.lionsclub-pontesanpietro-isola.it/LionsPSP3105/Screening.htm>, accessed 2011.
- [27] Ludovica Munari. *E-Democracy e Serious Game: l’esperienza del consiglio regionale del Veneto*. Master’s thesis, Università degli Studi di Padova, 2009/2010.
- [28] David R. Michael and Sandra L. Chen. *Serious Games: Games That Educate, Train, and Inform*. Muska & Lipman/Premier-Trade, 2005.
- [29] Nike. Nike+. <http://nikerunning.nike.com/>, accessed 2011.
- [30] F. Pittarello and R. Stecca. *Querying and Navigating a Database of Images With the Magical Objects of the Wizard Zurlino*. In *9th international Conference on interaction Design and Children*, pages 250–253, 2010.
- [31] Precision Vision. PVVAT.
<http://www.precision-vision.com/index.cfm/feature/20/pvvat.cfm>, accessed 2011.
- [32] Pulsetracer, Inc. <http://mybasis.com/>, accessed 2011.
- [33] Roger Chou and Tracy Dana and Christina Bougatsos. *Screening for Visual Impairment in Children Ages 15 Years: Update for the USPST*. *Pediatrics*, 127(2):442–479, 2011.

- [34] Savino D'Amelio. *L'Ambliopia... questa sconosciuta.* <http://www.aipam.it/ambliopia.asp>, accessed 2011. Ospedale Oftalmico di Torino, Divisione di Oftalmologia Infantile e Unità Operativa di Ortottica e Strabismo, Associazione Italiana Prevenzione Ambliopia.
- [35] The Internet Society. Introducing JSON. <http://json.org/>, accessed 2011.
- [36] Università degli Studi di Padova. Dipartimento di Pediatria Salus Pueri. <http://www.pediatria.unipd.it/>, accessed 2011.
- [37] C. Williams, K. Northstone, R.A Harrad, J.M. Sparrow, and I. Harvey. *Amblyopia treatment outcomes after screening before or at age 3 years: follow up from randomised trial.* *BMJ*, 324:1549–1551, June 2002.