Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica

Play on your CPU:

la didattica giocosa dei principi di funzionamento dei processori tramite un'applicazione per sistemi mobili.

Relatore:
Chiar.mo Prof.
Renzo Davoli

Presentata da:
Giulia Naponiello

Sessione III
Anno Accademico 2013/2014

Indice

Elenco delle figure

Introduzione

"Play on your CPU" è un'applicazione mobile per ambienti Android e iOS. Lo scopo di tale progetto è quello di fornire uno strumento per la didattica dei principi di funzionamento dei processori a chiunque sia interessato ad approfondire o conoscere questi concetti.

Si cerca di approcciarsi all'utente attraverso un'esposizione degli argomenti il più interattiva e giocosa possibile. Sono state inserite parti di sperimentazione, vi è infatti una sezione che consente di costruire circuiti "virtuali" che possono essere testati su una *breadboard* fisica. Gli argomenti trattati sono presentati attraverso una sorta di libro interattivo, fornito di esempi, tabelle, spiegazioni e molto altro. È stata inserita un'ultima sezione ludica, con la speranza di avvicinare, attraverso il gioco, un pubblico che fosse il più vasto possibile.

L'applicazione è disponibile su tablet e cellulari, strumenti utilizzatissimi ogni giorno da chiunque. Per questo si è scelto di realizzare "Play on your CPU": per creare uno strumento disponibile a tutti, senza distinzioni di età e conoscenze pregresse.

Capitolo 1

Stato dell'arte

1.1 Informatica nel pensiero comune

Nel pensiero comune vi è un'errata concezione dell'informatica, spesso legata solamente ai computer e agli strumenti che vengono utilizzati per l'elaborazione dei dati. Frequentemente viene confusa con la mera tecnologia e non viene riconosciuta come una scienza. Questo deriva sicuramente dalla disinformazione verso ciò che l'informatica rappresenta. Si pensa spesso che specifici programmi o dispositivi siano identificabili con il termine "informatica".

Questa disinformazione è in parte causata da una carenza nell'insegnamento dell'informatica nelle scuole primarie e della poca importanza che viene attribuita a tale materia. Un altro fattore responsabile è il diffuso utilizzo passivo e poco consapevole della tecnologia presente, che viene frequentemente utilizzata in maniera scorretta e per fini di scarsa utilità e non come uno strumento utile all'apprendimento e al miglioramento delle nostre vite.

Lo scopo di questo progetto è di fornire uno strumento che possa far avvicinare all'informatica chiunque, a prescindere dall'età e dalle conoscenze pregresse. L'obiettivo è quello di spiegare argomenti a basso livello - dal

codice binario, alla struttura di un processore - attraverso uno strumento ad alto livello a tutti familiare: l'applicazione mobile. Tutti utilizzano un telefono cellulare, ma pochi sanno cosa esso sia veramente, come esso funzioni e quali siano i componenti al suo interno. Con questo lavoro si vuole quindi fornire agli utenti una maggiore consapevolezza sul funzionamento degli strumenti di elaborazione che usano quotidianamente.

Si vuole dimostrare che l'informatica è una scienza divertente e creativa, e che essere informatici significa avere vastissime competenze utili in moltissime discipline. Essa educa a risolvere i problemi (*problem solving*), a trovare le soluzioni ottime; insegna a pensare in maniera sistematica e a lavorare in gruppo. Questo è il motivo per cui vi è sempre più bisogno dell'insegnamento di questa scienza.

A tal fine si è scelto di costruire uno strumento utile per la didattica e utilizzabile da chiunque. Questo ha portato a una strutturazione diversificata in base a differenti approcci. Persone con gli interessi più disparati possono entrare in contatto con questa applicazione e comprendere le possibilità che l'informatica ha da offrirgli.

1.2 Scelta degli argomenti

È legittimo chiedersi perché si è scelto di affrontare questi specifici argomenti, piuttosto che altri più ad alto livello. Tale scelta deriva non soltanto da un interesse personale per la materia, ma anche dalla volontà di far riemergere nel lettore la curiosità e la fantasia tipiche di un'età fanciullesca, qualità che possono riaffiorare grazie all'informatica, disciplina dove non è fondamentale solamente l'interesse per le materie scientifiche, ma anche una discreta dose di creatività. È forse più facile avvicinare il lettore a uno strumento fisico e tangibile, uno strumento hardware con il quale può materialmente interagire e vedere i risultati che cambiano sotto i propri occhi, senza alcuna "magia" che non riesce a comprendere.

La trattazione dell'hardware e degli argomenti a basso livello è la più nascosta e quindi temuta. È frequente temere ciò che non si conosce e di conseguenza evitarlo. Alle volte la mancata comprensione di una materia deriva dallo scarso interesse per essa o dalla mancata conoscenza. Uno degli obiettivi di questa applicazione è quello di far familiarizzare l'utente con tali argomenti, stimolando il suo interesse attraverso uno strumento ludico.

1.3 Strumenti hardware già presenti

Vi sono presenti già moltissimi strumenti creati a scopo didattico e per rendere l'informatica un'esperienza di divertimento. Qui di seguito esporremo quelli che sono i mezzi hardware più famosi e apprezzati.

Essi sono tuttavia differenti da ciò che è stato creato in questo progetto, in quanto esso non è rappresentato da uno strumento fisico, ma da un'applicazione mobile, la quale fornisce anche un'emulazione di un hardware.

Gli strumenti che esporremo in seguito spesso non vengono utilizzati, più che per il loro prezzo di acquisto che risulta essere molto economico, a causa di una poca dimestichezza con l'hardware da parte dell'utente. Si è infatti spesso disorientati e un po' timorosi di fronte al loro utilizzo, tuttavia è da sottolineare la presenza sempre più vasta di documentazione sul web che cerca di eliminare tale scoglio.

1.3.1 Raspberry Pi

Il *Raspberry Pi* è un computer grande quanto una carta di credito, dal costo limitato (circa 35 euro), creato con lo scopo di fornire uno strumento utile per la programmazione e lo studio dell'informatica, anche nelle scuole. Oltre all'economicità, uno dei maggiori vantaggi di tale *board*, è quello di supportare un vero e proprio sistema operativo e fornire quindi un'interfaccia di

semplice utilizzo anche ai meno esperti. Il sistema operativo maggiormente utilizzato e consigliato è *Raspian*, una versione di *Debian* ottimizzata per il *Raspberry Pi*, ma non è il solo sistema disponibile[1].

Questa *board* è composta[2] da un circuito integrato *Broadcom* BCM2835[3], il quale comprende un processore ARM[4] da 700Mhz, una GPU[5] VideoCore IV e una RAM, condivisa con la GPU, da 512MB. Il *Raspberry Pi* prevede uno *slot* per la scheda microSD come supporto di memorizzazione. Per il collegamento alla rete è fornito di una porta Ethernet 10/100. Per l'output video è disponibile un'uscita composita e una HDMI, mentre per l'output audio è prevista un'uscita jack 3.5mm e HDMI. Dispone inoltre di 4 porte USB2.0, una porta per la videocamera opzionale e una per un display LCD. Gli elementi più interessanti del *Raspberry Pi* rimangono tuttavia i 26 pin chiamati GPIO (General Purpose Input/Output), essi permettono di dialogare con altri apparecchi e possono essere utilizzati come input o output a seconda delle necessità.

Il vantaggio di questo strumento è sicuramente quello di avere un bassisimo costo se commisurato con ciò che offre, questo permette a chiunque di ottenere un vero e proprio computer senza avere alcuna paura di sperimentare. Sul web è presente molto materiale riguardante i più svariati progetti realizzabili: dalla macchinina telecomandata, ad utili esperimenti di domotica. Risulta sicuramente un valido approccio per insegnare a chiunque, anche ai bambini, la programmazione e molto altro ancora, lasciando grande spazio alla creatività.

[1] `http://www.raspberrypi.org/downloads/`

[2] Tali specifiche si riferiscono al modello B+, per vedere le specifiche di tutti i modelli visitare:

`http://www.raspberrypi.org/documentation/hardware/raspberrypi/models/specs.md`

[3] `https://www.broadcom.com/products/BCM2835`

[4] Famiglia di microprocessori RISC a 32-bit.

[5] Graphics Processing Unit.

1.3.2 Arduino

Arduino è una scheda elettronica dotata di un semplice microcontrollore e fornisce un ambiente di sviluppo per scrivere software per tale *board*. Arduino può essere utilizzato per creare numerosi progetti in maniera rapida e per sviluppare oggetti interattivi, collegando led, sensori, pulsanti, motori e molto altro ancora.

Arduino comprende un'intera famiglia di *board*[6], ognuna con differenti specifiche. La *board* può essere assemblata a mano oppure è possibile acquistare il Kit di montaggio e saldare i componenti su circuito stampato. Infatti Arduino è un hardware *open source* ed è possibile costruirne una propria versione modificata grazie all'opportunità di scaricare lo schema elettrico e la lista di componenti necessari.

Numerosi fattori rendono Arduino vantaggioso rispetto ad altri microcontrollori, non soltanto per hobbisti e utenti amatoriali interessati, ma anche per scopi didattici e quindi per studenti e insegnanti.

- È più economico di qualsiasi altro microcontrollore, il suo prezzo è variabile in base alla *board*, ma in ogni caso resta sempre inferiore ai 50 euro.
- L'ambiente di sviluppo e di programmazione è molto semplice e quindi adatto anche a persone senza particolari competenze e preconoscenze.
- Arduino è compatibile con i sistemi operativi Linux, Mac OS X e Windows, al contrario della maggior parte dei microcontrollori, compatibili solamente con Windows.
- Come già detto, l'hardware è *open source* e allo stesso modo lo è anche il software di Arduino, il quale può essere esteso e pubblicato.

[6] `http://arduino.cc/en/Main/Products`

Numerosissimi progetti sono realizzabili con Arduino, per esempio con Arduino Robot[7] è possibile riprodurre dei file musicali, scegliendo fra tre melodie attraverso i bottoni di controllo[8].

Un altro esempio di progetto[9] consiste nel leggere i dati da un sensore di pressione barometrica, utilizzando le *Serial Peripheral Interface*[10].

1.4 Metodologie già presenti

Vi sono presenti moltissime metodologie volte alla didattica dell'informatica e di seguito ne verranno esposte due particolarmente interessanti e innovative. Entrambe nascono con lo scopo di insegnare la programmazione ai bambini, quindi si tratta di un fine leggermente differente da quello preposto per il progetto relativo a questo elaborato. Tuttavia sono sicuramente approcci didattici efficaci e degni di nota.

1.4.1 Scratch

Scratch[11] è uno strumento che permette di programmare attraverso un'interfaccia grafica intuitiva e che consente di realizzare giochi, storie interattive e animazioni, attraverso il *drag and drop* di blocchi rappresentanti istruzioni. È un approccio divertente e molto adatto ai bambini, che insegna a pensare in maniera creativa e predispone a un tipo di pensiero adatto alla programmazione. Il target principale di Scratch è rappresentato quindi dalla fascia di età dagli 8 ai 16 anni, tuttavia è adatto e utilizzabile da persone di qualsiasi età. Forse l'obiettivo principale di questo strumento è quello di insegnare ai bambini non a programmare, ma a pensare in maniera sistematica e a saper risolvere problemi.

[7] `http://arduino.cc/en/main/robot`
[8] `http://arduino.cc/en/Tutorial/RobotDiscoBot`
[9] `http://arduino.cc/en/Tutorial/BarometricPressureSensor`
[10] `http://arduino.cc/en/Reference/SPI`
[11] `http://scratch.mit.edu/`

Scratch non è solo uno strumento per programmare, ma anche una comunità online che fornisce aiuto e tantissimi esempi di progetto per iniziare a realizzare animazioni, giochi, storie e tanto altro[12]. Tali progetti sono liberamente modificabili e utilizzabili. Questo è in linea con la filosofia di Scratch, secondo la quale la condivisione è un fattore importante e si prefissa inoltre di insegnare ai bambini a lavorare in maniera collaborativa.

È possibile utilizzare Scratch online, oppure offline attraverso la versione *desktop* disponibile per Linux, Mac OS X e Windows.

1.4.2 CoderDojo

I CoderDojo[13] sono dei club nati con lo scopo di insegnare la programmazione ai bambini. Gli incontri sono gratuiti, totalmente liberi e senza scopi di lucro. Vi sono presenti numerose "palestre" in tutto il mondo, di cui più di 40 in Italia. È stato creato infatti il CoderDojo Italia[14], ovvero la comunità di tutti i CoderDojo italiani.

In questa sede ai bambini vengono insegnati linguaggi come HTML, Javascript e CSS, inoltre gli viene insegnato come creare programmi con Scratch o piccoli progetti con Arduino.

I CoderDojo sono sempre più presenti all'interno delle scuole, con un incremento di contributo anche da parte degli insegnanti che diventano *mentor*, termine utilizzato dalla comunità del CoderDojo per indicare il ruolo di una guida per i bambini.

1.5 No more secrets in your CPU

Gli argomenti che sono stati trattati nello specifico sono quelli affrontati dal professor Renzo Davoli all'interno del suo progetto e documento "No

[12] `http://scratch.mit.edu/starter_projects/`

[13] `https://coderdojo.com/`

[14] `http://www.coderdojoitalia.org/`

more secrets in your CPU", disponibile sul wiki del RaspiBO[15]. Tali argomenti sono stati ritenuti i più adeguati per lo svolgimento di tale progetto, per la completezza e la semplificazione di concetti complessi riguardanti anche argomenti che esulano a volte dall'informatica, come ad esempio l'elettronica.

Tale documento affronta in primis come costruire, con componenti elettronici di basso costo, uno strumento del tutto simile a una normale CPU come tutti conosciamo[16]. Tale *tool* è composto da una *breadboard* ospitante 11 porte NAND, ognuna costruita attraverso 2 *transistor*, 3 resistenze e 2 *jumper*. È inoltre presente un *master clock* di 1Hz creato mediante 3 *transistor*, 5 resistenze, 2 condensatori e 4 *jumper*.

Viene poi mostrato come è possibile costruire moltissimi circuiti su tale *breadboard*, fondamentali per il funzionamento di un processore: dal *full adder*, alla ALU. In questo documento, di tali circuiti, non sono disponibili solamente le indicazioni per la costruzione, ma anche spiegazioni di come essi operino, tabelle di verità e disegno attraverso sole porte NAND.

[15] `http://www.raspibo.org/wiki/index.php/Pagina_principale`

[16] `http://www.raspibo.org/wiki/index.php/No_more_secrets_Part_2:_Let_us_build_the_experimental_board`

Capitolo 2

Play on your CPU

Play on your CPU è un'applicazione mobile per ambienti *Android* e *iOS*, realizzata attraverso il *framework open source Cocos2d-x*. Tale applicativo è stato diviso in tre macrosezioni, finalizzate all'apprendimento dell'utente dei contenuti proposti:

- Impara: trattasi di una sorta di libro interattivo all'interno del quale sono esposti la maggior parte degli argomenti presenti nel documento *No more secrets in your CPU*, con l'aggiunta di alcuni concetti di informatica di base. Tale trattazione viene presentata suddivisa in base a un crescente livello di difficoltà.

- Sperimenta: vi è presente un'emulazione di una *breadboard*, costruita secondo le specifiche indicate, anche in questo caso, nel documento *No more secrets in your CPU*. Lo strumento è stato disegnato e realizzato attraverso *Tile Map* e *Gimp*. In questa sezione è possibile interagire con la *breadboard* ed effettuare un certo numero di operazioni.

- Gioca: un gioco il cui funzionamento è ispirato a quello di una rete combinatoria, ovvero una rete il cui output dipende solamente dagli ingressi correnti. Nonostante tale premessa, esso è intrapprendibile da chiunque, a prescindere dalle effettive conoscenze informatiche. Tale

gioco dovrebbe funzionare da "esca" per attrarre utenti di qualsiasi tipo e creare in loro l'interesse di scoprire come tale sezione è stata realizzata e quali basi si fondano sulle meccaniche di *gameplay*.

Il target auspicato non presenta molti limiti, la speranza è infatti quella di rendere *Play on your CPU* disponibile ad utenti di qualsiasi età, genere e preconoscenze. Vuole essere uno strumento per dimostrare al pubblico quanto l'informatica possa essere interessante e divertente, anche possibilmente al pubblico femminile che sembra autoprecludersi l'interesse verso tale materia, apparentemente per un qualche infondato pregiudizio presente nella società contemporanea.

Infatti secondo le statistiche effettuate dal Ministero dell'Istruzione, dell'Università e della Ricerca[1] nell'anno accademico 2013/2014, le donne iscritte al corso di laurea triennale di Informatica dell'Università di Bologna sarebbero solo 39, contro i 321 uomini. Sempre nello stesso anno, le donne immatricolate sarebbero solamente 11, contro i 92 uomini[2].

È forse molto ambizioso voler raggiungere un insieme di persone così vasto, specialmente per quanto riguarda il tipo di utenza più giovane, ma l'obiettivo che quanto meno si brama di raggiungere, è quello di poter accendere un piccolo barlume di interesse, che possa far nascere, anche in un momento futuro, uno stimolo per una disciplina così mal compresa.

2.1 Strumenti per lo sviluppo

In questa sezione parleremo degli strumenti maggiormente importanti e maggiormente utilizzati per lo sviluppo di "Play on your CPU". Primo fra tutti *Cococs2d-x*, il *framework* utilizzato per scrivere il codice dell'applicazione. Gimp è invece il programma di fotoritocco utilizzato per costruire qualsiasi immagine presente in questo progetto.

[1] `http://statistica.miur.it/`

[2] `http://statistica.miur.it/scripts/IU/vIU1.asp`

2.1.1 Cocos2d-x

Cocos2d-x è un *framework open source* che permette di scrivere applicazioni multipiattaforma in linguaggio C++ (nello specifico per gli ambienti Android e iOS). *Cocos2d-x* permette di creare applicativi molto leggeri e veloci e fornisce un ottimo supporto per diverse tecnologie e tecniche. Vi è infatti a disposizione un motore fisico[3] grazie al quale si riesce ad attribuire a certi elementi un corpo fisico e simulare movimenti e collisioni in maniera del tutto realistica. Permette di utilizzare agilmente strumenti come l'accelerometro e fornisce un ottimo supporto per tecniche come le *Tile Map* (che approfondiremo in seguito).

Cocos2d-x fa parte di una grande famiglia di *framework open source*[4], infatti vi è, per esempio, una versione in Javascript (*Cocos2d-JS*), oppure *Cocos2D-Swift*, che può essere utilizzato per costruire giochi in 2D multipiattaforma con Xcode[5] e Objective-C[6]. Vi è anche una versione in Python, del tutto simile a quella in C++, funzionante su Linux, Mac OS X e Windows. Moltissime aziende specializzate nella realizzazione di videogiochi hanno utilizzato questo *framework* in passato; *Square Enix*, produttori del popolarissimo gioco *Final Fantasy*, ne è un esempio, merito sicuramente la facile portabilità del codice e la velocità col quale permette di scrivere applicazioni grazie alle moltissime librerie di supporto.

Scelta del framework

La scelta di *Cocos2d-x* è stata dettata principalmente dalle seguenti cause:

- Prima e più importante è stata la volontà di rendere questo progetto disponibile al più vasto pubblico possibile. Questo ha portato a scegliere

[3] `https://chipmunk-physics.net/`

[4] `http://cocos2d.org/`

[5] Un IDE per lo sviluppo di software per Apple.

[6] Linguaggio di programmazione a oggetti per applicazioni iOS.

uno strumento che potesse essere ospitato sui più diffusi sistemi operativi per cellulari (Android e iOS). Il *framework*[7] scelto permette di creare un'applicazione multipiattaforma attraverso un codice, scritto in C++, identico e immutato per entrambi gli ambienti.

- L'applicazione che è stata sviluppata racchiude i contenuti e utilizza le tecniche più disparate. Si è voluta realizzare non solo una parte didattica, in stile libro interattivo, con elementi grafici piacevoli ed accattivanti, ma anche una parte di *gaming* e un'ultima sezione di emulazione e simulazione.

- Da non sottovalutare il fatto che si tratti di un *framework open source*, fattore che ha sicuramente portato allo sviluppo di una comunità online, che ha reso disponibile una vasta gamma di tutorial e di documentazioni, utili anche a chi si interfaccia per la prima volta con *Cocos2d-x*.

Tile Map

Due delle tre sezioni presenti nel progetto realizzato, sono state interamente costruite grazie alle *Tile Map*: mappe suddivise in celle (*tile*). A ogni cella, o insieme di celle, è possibile attribuire specifiche proprietà che verrano riconosciute all'interno del codice. Grazie a queste proprietà potremo stabilire determinati comportamenti in base al tipo di cella con cui stiamo interagendo.

Per generare tali mappe è stato utilizzato l'editor *Tiled*[8], che fornisce un file salvato in formato tmx, supportato da *Cocos2d-x*. Inizialmente *Tiled* permette di scegliere la grandezza delle singole celle e del numero di celle della mappa. Successivamente fornisce la possibilità di creare la mappa

[7] D'ora in avanti, ogni volta che nel testo verrà utilizzato il termine "il framework", è da intendersi traducibile come *Cocos2d-x*.

[8] `http://www.mapeditor.org/`

attraverso un *tileset*, che potremmo identificare come un'immagine in cui sono presenti tutti i differenti tipi di celle che potranno essere inserite in maniera ripetuta o singola.

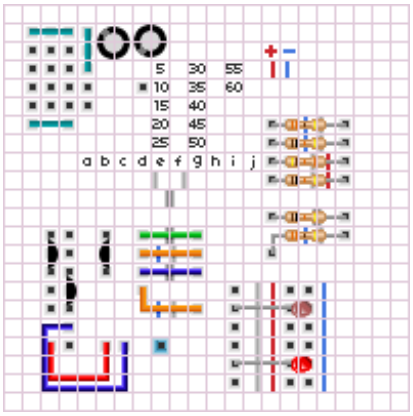

Figura 2.1: *Tileset* creato per realizzare la *breadboard*

Il *software Tiled* genera una la mappa, le cui celle sono accedibili da *Cocos2d-x* attraverso un sistema di coordinate che forma una griglia. La cella nell'angolo in alto a sinistra viene identificata con la coordinata (0,0), mentre quella posizionata nell'angolo in basso a destra avrà coordinate (numero colonne - 1, numero righe - 1). Attraverso questo sistema viene reso possibile il riferimento a una qualsiasi cella, in una qualsiasi posizione.

Il *framework* fornisce un ampio insieme di librerie che permettono di interagire con la *Tile Map* creata. Vi sono infatti funzioni in grado di modificare le celle, attribuendo un differente disegno alla mappa. Altri metodi permettono di riconoscere le proprietà attribuite alle *tile*. È possibile inserire differenti livelli di una mappa, che possono essere resi visibili o meno a *runtime*. Si possono inoltre aggiungere oggetti (come ad esempio il *player* inserito nel gioco sviluppato) e farli interagire con la mappa.

Nonostante questa sia una tecnica principalmente utilizzata per i videogiochi, si è rivelata estremamente adatta anche al tipo di interfaccia per

l'emulazione della *breadboard*, come avremo modo di vedere in seguito.

2.1.2 Gimp

Gimp[9] (*GNU Image Manipulation Program*) è un *software open source* per il fotoritocco e la manipulazione di immagini. È stato largamente utilizzato per la realizzazione di tale progetto. La scelta è stata dettata principalmente dal fatto di possedere già una conoscenza pregressa di tale programma. Sicuramente, un altro fattore che ha aiutato in tale scelta, è che si tratti di un *software open source*, senza considerare una delle migliori qualità di Gimp: la sua leggerezza, attributo solitamente atipico per quanto riguarda i programmi di grafica.

Gimp permette di creare immagini complesse in molteplici formati. Il modo migliore di lavorare in tale applicativo è quello di generare diverse trasparenze, chiamate livelli, i quali possono essere modificati, spostati, ruotati, opacizzati e molto altro ancora. Tali livelli verranno fusi nel caso del salvataggio nei comuni formati quali, per esempio, jpg o png. Formato nativo di Gimp è xcf, che mantiene invece tali trasparenze anche con il susseguirsi delle chiusure e riaperture del programma, permettendo all'utente di continuare a modificare l'immagine, come l'aveva salvata, senza compressioni e senza perdita di informazioni di alcun tipo.

Esso fornisce i più svariati tipi di strumenti, fra i più importanti possiamo citare la *selezione rettangolare* che permette appunto di selezionare un'area di forma rettangolare, modificando eventualmente la sfumatura dei bordi o il grado di arrotondamento degli angoli. Vi è presente anche la *selezione ellittica* e *a mano libera*. Altri strumenti degni di nota sono il *pennello* e la *matita* grazie ai quali è possibile disegnare tratti dai bordi più morbidi o più secchi. Gimp offre oltre a questi e a molti altri strumenti, la possibilità

[9] `http://www.gimp.org/`

di lavorare attraverso tracciati, maschere di livello[10], canali alfa[11] e anche *plugin* integrativi.

È molto comune leggere di dibattiti comparativi fra Gimp e Adobe Photoshop[12]. In seguito illustriamo le principali differenze fra i due programmi:

- Gimp è in grado di supportare il formato nativo di Adobe Photoshop (psd), al contrario di quest'ultimo che non supporta gli xcf;
- Gimp è maggiormente limitato di Adobe Photoshop per quanto riguarda la gestione dei colori;
- Adobe Photoshop contiene maggiori funzionalità, apparentemente utili soprattutto per un tipo di grafica di livello professionale;
- Gimp è disponibile per la maggior parte dei sistemi operativi, fra cui Linux, Mac OS X e Windows, al contrario di Adobe Photoshop che è disponibile solamente per Windows e Mac OS X;
- Gimp appare essere più vantaggioso rispetto ad Adobe Photoshop a livello di specifiche hardware richieste, necessitando di minori risorse sia a livello di RAM, che a livello di spazio su disco;
- Gimp è *open source* e completamente *free*, mentre Adobe Photoshop è *software* proprietario e a pagamento.

Si lascia al lettore la considerazione di quale software sia migliore, in quanto la comparazione non sembra essere basata su una questione qualitativa, ma piuttosto una questione di necessità prettamente personale.

[10] Consentono di definire parti opache o trasparenti in un livello.

[11] Canale utile per la trasparenza.

[12] Software proprietario per il fotoritocco prodotto dalla Adobe Systems.

2.2 Tre macrosezioni

L'idea di creare una suddivisione in tre differenti sezioni, con diversi approcci didattici, è spinta dalla convinzione che attraverso uno strumento ludico e la memoria visiva sia più semplice apprendere concetti complessi o verso i quali siamo più avversi. Per questo si è cercato un risultato il più interattivo possibile.

L'utente all'apertura dell'app è portato a scegliere quale sia il metodo che egli ritiene più adeguato al proprio metodo di apprendimento. Nonostante siano differenti, tutti e tre gli approcci si basano sulle stesse tematiche.

Figura 2.2: Situazione all'avvio dell'applicazione

Si è scelta tale suddivisione pensando in special modo alle diverse preparazioni e competenze del lettore. Ogni persona presenta differenti abilità e differenti approcci allo studio, nonché prerequisiti diversi. Essendo un'app destinata a tutti, si rende possibile a chiunque di orientarsi all'interno di essa.

Le tre sezioni possono essere identificate anche in base a differenti livelli di complessità. La sezione di gioco è infatti molto semplice e affrontabile da tutti. La sezione rappresentata dal "libro interattivo" tratta di alcuni argomenti comprensibili da chiunque e altri più complessi, che potrebbero non essere chiari a utenti dalle scarse conoscenze informatiche. Arrivati al termine degli argomenti trattati in tale sezione dovrebbero essere stati acquisiti tutti i requisiti per affrontare la parte forse più complessa dell'applicazione, ovvero quella di sperimentazione.

2.2.1 Impara

Quando l'utente sceglie tale sezione deve nuovamente decidere fra 3 proposte, le quali identificano il livello di complessità degli argomenti che verranno esposti. Essi sono:

- Principiante: gli argomenti qui contenuti sono i più semplici proposti e non richiedono grandi conoscenze pregresse. Per facilitare l'apprendimento sono stati inseriti esempi interattivi per aiutare la comprensione dei concetti meno banali.

  Il primo argomento affrontato è il sistema binario; chiaramente, assimilare e capire tale concetto, risulta fondamentale e indispensabile per affrontare gli argomenti successivi. Seguono, a tale spiegazione, quella relativa alla nozione di bit significativo, tabelle di verità e insieme funzionalmente completo. Nella sezione relativa alle tabelle di verità vengono fornite tutte le tabelle relative ai principali *gate*, ovvero NOT, AND, NAND, OR, NOR, XOR, NXOR, mostrando anche la re-

lativa porta logica. Nella parte riguardante gli insiemi funzionalmente completi è presente la costruzione dei circuiti logici delle porte sopra citate attraverso soli NAND, in quanto tale *gate* da solo costituisce un insieme funzionalmente completo al pari del *gate* NOR.

- Medio: in questa parte si sono inseriti i concetti indispensabili per comprendere il funzionamento dei componenti presenti sulla *breadboard* che si è costruita e per capire lo scopo di tale strumento. Infatti vi è una sezione dove viene illustrato cos'è un processore e come il suo funzionamento sia paragonabile a quello della *breadboard* del documento *No more secrets in your CPU*. Vi sono inoltre le indicazioni necessarie per costruire tale strumento. Altri concetti illustrati, in maniera molto sintetica, sono: transistor, resistenze e condensatori.

- Esperto: vi sono presenti i circuiti trattati nel documento *No more secrets in your CPU*, i quali corrispongono ai circuiti di maggior rilievo nelle reti logiche. Alcuni esempi sono: decoder, encoder, multiplexer, demultiplexer, full adder, flip flop.

La difficoltà maggiore riscontrata nella realizzazione di tale sezione è stata quella di comprimere così tanti argomenti all'interno dello spazio limitato di un'applicazione mobile. Per aggirare tale problematica si è cercato di rendere i contenuti presentati il più concisi e sintetici possibile, al fine di non rendere la trattazione tediosa per l'interlocutore. I concetti sono stati spesso accompagnati da esempi e immagini per una lettura più piacevole e per fornire un'interfaccia e un approccio maggiormente visivo.

Tale sezione, trattandosi di una sorta di libro interattivo, è stata creata attraverso un susseguirsi di scene. Una scena è un oggetto di *Cocos2d-x* chiamato *Scene*, sottoclasse di *Node*, il quale è il più importante elemento del *framework*, infatti, quasi ogni altro elemento presente è sottoclasse di

Node[13]. Possiamo identificare una scena come il signolo *schermo* dell'applicazione in un preciso istante. Un'applicazione può avere più scene, ma solamente una può essere attiva in un solo momento. In *Cocos2d-x* una *Scene* può essere composta da uno o più *Node* che vengono aggiunti come *child* della scena. La scena è quindi un contenitore di elementi come *Sprite* o *Label* e vi è necessario che ve ne sia almeno una per avviare l'applicazione.

Per creare una scena è sufficiente utilizzare questo codice:

```
auto myScene = Scene::create();
```

Invece per passare da una scena all'altra vi sono due metodi:

- *pushScene*: la vecchia scena rimane sullo stack e sarà possibile ritornare ad essa non appena la nuova scena avrà terminato, utilizzando il metodo *popScene*.

  Di seguito vediamo il codice per la creazione di una nuova scena:

  ```
  auto myScene = myNewScene::createScene();
  Director::getInstance()->pushScene(myScene);
  ```

- *replaceScene*: la scena sostituita non rimane allocata sullo stack, ma viene rimossa; quindi non sarà più possibile farvi ritorno.

È possibile invece tornare alla scena precedente in tal modo:

```
Director::getInstance()->popScene();
```

[13] http://www.cocos2d-x.org/reference/native-cpp/V3.0alpha0/d3/d82/classcocos2d_1_1_node.html

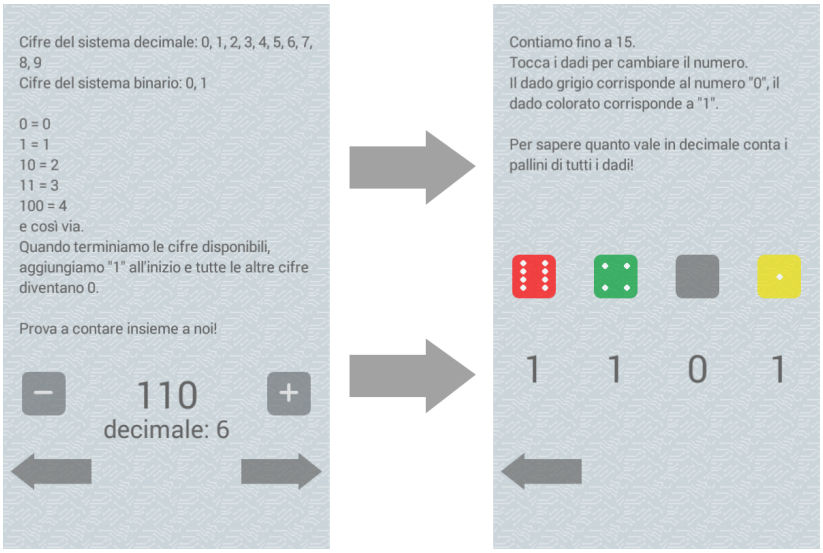

Figura 2.3: Passaggio da una scena all'altra

Il *Director*, che abbiamo visto in uno stralcio di codice precedente, è l'oggetto di *Cocos2d-x* responsabile della navigazione fra le scene. Egli è a conoscenza di quale sia la scena attualmente attiva e permette di cambiarla. È quindi un suo compito quello di mettere in pausa le scene e di riesumarle.

2.2.2 Sperimenta

In questa sezione viene data la possibilità di interagire con un'emulazione della *breadboard* di cui abbiamo già parlato e sperimentare vari circuiti attraverso gli 11 *gate* NAND disponibili. La *breadboard* è stata realizzata attraverso la tecnica di *Tile Map* grazie al software *Tiled* ed è stata interamente disegnata con il programma di grafica Gimp.

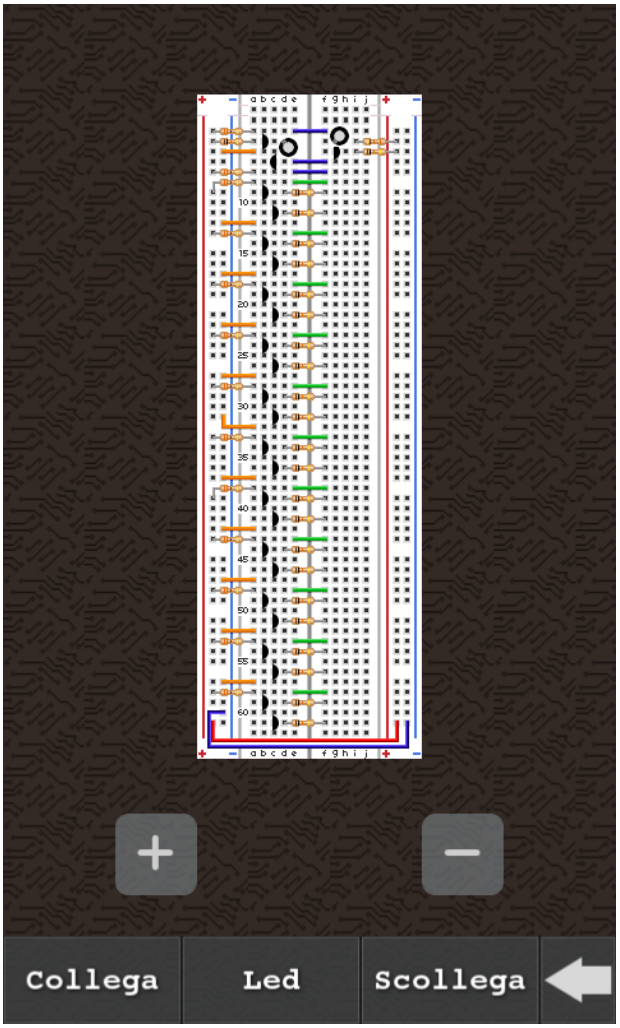

Figura 2.4: Schermata all'avvio della sezione "Sperimenta"

Oltre alla rappresentazione grafica creata per fornire un *feedback* visivo all'utente di ciò che sta avvenendo, vi è una parte implementativa dove è presente una matrice 11x2 - dove 11 è il numero di *gate* e 2 sono gli input della porta NAND - responsabile di tenere traccia dei valori degli input dei vari *gate*. Similmente alla matrice, vi è un array contenente il valore di ogni *gate*, ovvero dei loro output.

Gli elementi all'interno della matrice avranno un valore compreso fra 0 e

12, corrispondente al valore di un *gate* o equivalente al valore 0 o 1. Infatti l'input di un *gate* potrà essere equivalente al valore dato in output da un altro *gate*, in maniera tale da connettere i vari NAND fino a formare un circuito. È stato creato inoltre un secondo array resposabile di memorizzare in quali input e output è connesso un led, in maniera tale da poter risalire a tale informazione in qualsiasi momento. Gli elementi di tale array, come quelli del vettore relativo agli output, potranno assumere i soli valori 0 e 1. In particolare, gli elementi del vettore tenente traccia dei led, avranno valore 0 quando il led non è connesso e 1 in caso contrario.

Ogni input viene inizializzato a 0 all'avvio della scena, in quanto inizialmente non è ancora stato effettuato alcun collegamento. Tutti gli output di ogni *gate* sono invece inizializzati a 1, infatti, come è ben noto, settando a 0 entrambi gli input di una porta NAND, essa fornirà in output il valore 1.

Sono state create diverse costanti attraverso la direttiva per il preprocessore *define*, le più importanti sono quelle aventi come nome le prime 11 lettere dell'alfabeto, in maniera tale da poter inserire all'interno della matrice degli input il riferimento al collegamento di un output e un input.

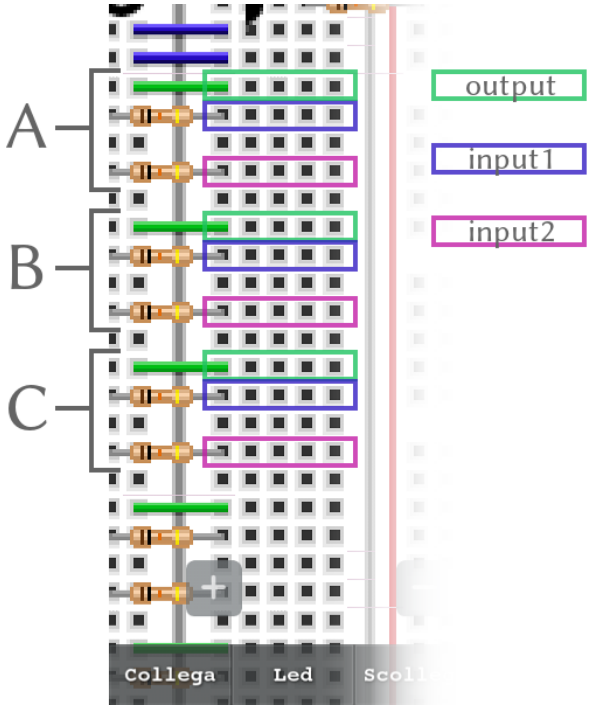

Figura 2.5: Posizionamento dei *gate* sulla *breadboard*

All'utente viene data la possibilità di svolgere varie operazioni. La prima è quella di connettere dei fori fra loro attraverso un *jumper*. Per farlo egli dovrà prima di tutto toccare il bottone "Collega", in maniera tale da esprime all'applicazione la propria volontà di creare un collegamento. A questo punto l'utente si ritrova davanti a due sole operazioni considerate valide:

- Creazione di un collegamento fra il ramo positivo e l'input di un *gate*: con questa azione si abilita uno degli input del *gate*, in quanto ad esso verrà fornita la corrente dal ramo positivo. Con questo collegamento l'input verrà aggiornato al valore di 1.
- Creazione di un collegamento fra l'output di un *gate* e l'input di un altro *gate*: con questa azione l'output di un *gate* verrà fornito in input a un altro *gate* e nella matrice degli input verrà memorizzato tale collegamento.

Qualsiasi altro tipo di collegamento non è consentito in quanto non produrrebbe alcun risultato e quindi non ne è stata resa possibile l'effettuazione. In seguito al tentativo di eseguire una mossa non considerata valida, l'utente dovrà ripetere l'intera operazione, ovvero toccare nuovamente il bottone "Collega" in quanto l'azione precedente si considera annullata.

Al tocco del primo foro che si vuole collegare, verrà mostrato un *feedback* grafico, evidenziando con un'aurea colorata il foro effettivamente toccato. Nell'eventualità del tentato compimento di un collegamento non consentito, l'area colorata verrà eliminata e il foro riassumerà l'aspetto iniziale.

Vediamo lo stralcio di codice dove avviene il cambiamento dell'immagine della cella, nel momento in cui l'operazione deve essere annullata e il contorno colorato del foro deve essere rimosso e avviene il ripristino della cella originaria della mappa.

Ripristino immagine originaria della cella

```
1 void Sperimenta::fixMap(Point tofix){
2     unsigned int originale = _original->
3                         tileGIDAt(ccp((float)firstToConnect.x,
                              (float)firstToConnect.y));
4     _background->setTileGID(originale, ccp((float)firstToConnect.x,
          firstToConnect.y));
5 }
```

Per memorizzare il disegno originario della mappa è stato inserito un livello nella *Tile Map* che non viene mai modificato. Questo livello permette di risalire alle celle come erano prima che avvenisse una qualsiasi modifica. Viene quindi estrapolata la cella originaria dal livello e poi sostituita nel livello principale che è l'unico ad essere visibile e su cui avvengono tutti i cambiamenti. Ogni altro cambiamento d'aspetto delle celle avviene in maniera similare.

Al momento dell'effettivo collegamento, l'applicazione dovrà occuparsi

di aggiornare i valori dei *gate* in cascata. Infatti se vi sono presenti *gate* che forniscono il proprio output in input ad altri *gate*, allora anche tali input andranno aggiornati e corrispondentemente sarà aggiornato il valore della porta in caso di un cambiamento di valore. Vi è quindi un metodo responsabile di effettuare tale controllo e apportare ogni modifica. Esso viene richiamato ogni qualvolta viene creato o rimosso un collegamento.

Un'ulteriore operazione consentita è quella di inserimento di un led. Per compiere tale azione è sufficiente toccare prima il bottone "Led", comunicando quindi all'applicazione la propria volontà effettuare tale operazione, e successivamente toccare il punto in cui si desidera inserirlo. È stata aggiunta la possibilità di connettere un led sia nelle righe di output che nelle righe di input, in maniera tale da consentire all'utente di rendersi conto di tali valori. Anche in questo caso non è stato reso possibile collegare led all'infuori delle righe valide.

Nel caso in cui si tocchi un punto non considerato adeguato a tale operazione, il led non verrà connesso e quindi non sarà mostrato. Tuttavia l'operazione non verrà annullata e nel caso in cui successivamente si tocchi un input o un output, il led sarà collegato. Dopo aver toccato il bottone "Led", la volontà di inserire un led verrà annullata toccando il bottone "Collega" o il bottone "Scollega".

Quella di scollegare due *jumper* è l'ultima operazione effettuabile. Vi è appunto un bottone che permette all'utente di esprimere all'applicazione la propria volontà di rimuovere un collegamento già esistente. L'utente poi toccherà prima una delle due estremità del jumper e poi anche l'altra. Nel caso in cui il collegamento non sia effettivamente presente, o non siano stati toccati correttamente due punti collegati fra loro, l'operazione non verrà ritenuta valida e l'utente dovrà ripetere l'intera azione premendo nuovamente il bottone relativo. L'operazione verrà annullata anche nel caso in cui un altro dei bottoni venga toccato.

Per capire meglio il funzionamento di tale sezione vediamo un esempio di un circuito che è stato testato su tale *breadboard*: il decoder. In questo caso vedremo un decoder 2:4, ovvero con 2 ingressi (che chiameremo in1 e in0) e 4 uscite (che chiameremo out0, out1, out2, out3), la cui tabella di verità è la seguente:

in1	in0	out0	out1	out2	out3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Tabella 2.1: Tabella di verità del decoder 2:4

Il decoder, in base all'input, setta in output un solo valore a 1 e tutti gli altri a 0. L'output con valore 1 è quello corrispondente all'input abilitato. Come è infatti intuibile dalla tabella di verità, con input 00, ad essere attivato è l'out0; con input 01 viene attivato l'out1; con input 10 (2 in decimale) è attivato l'out2; con input 11 (3 in decimale), l'output settato a 1 è out3.

Un decoder 2:4 è realizzabile attraverso 10 *gate* NAND e di seguito vediamo il circuito relativo.

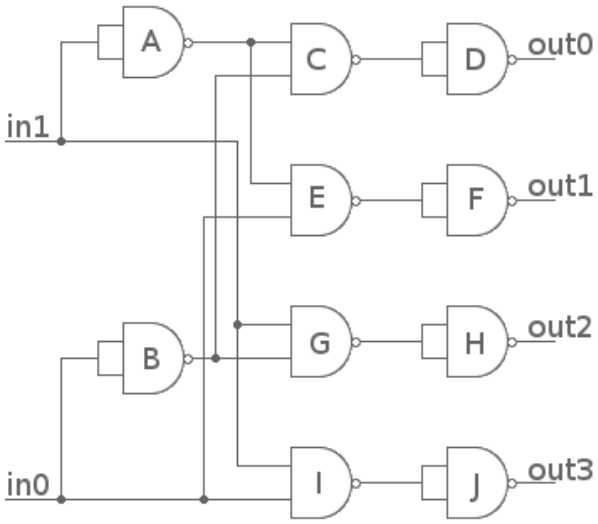

Figura 2.6: Circuito del decoder 2:4

I *gate* sono stati identificati con una lettera dell'alfabeto similmente a come sono stati ospitati su *breadboard*. Evitiamo di scendere nei dettagli più particolari[14], e mostriamo la realizzazione all'interno di "Play on your CPU":

[14] `http://www.raspibo.org/wiki/index.php/No_more_secrets_Part_4:_Encode_and_Decode...`

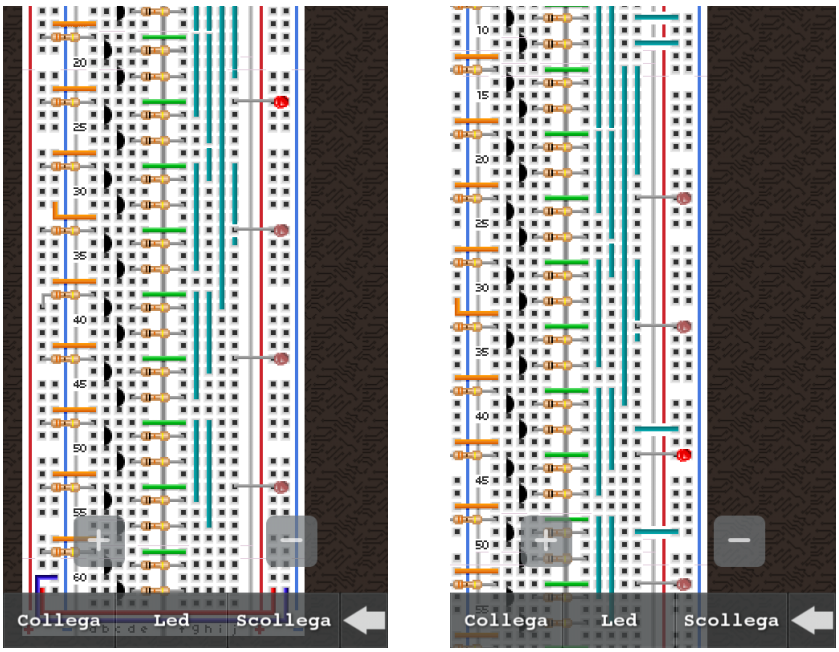

Figura 2.7: Decoder 2:4 con input 00 (a sinistra) e input 10 (a destra)

Nella prima immagine (a sinistra) la configurazione di ingresso è 00 - non vi è infatti nessun foro collegato con il ramo positivo - e si può infatti notare che gli output (in corrispondenza dei led) sono tutti settati a 0 ad eccezione del primo, il cui led è infatti acceso. Nella seconda immagine (a destra) la configurazione di ingresso è 10, è quindi stato settato a 1 l'input in1, che infatti viene ricevuto in ingresso dai *gate* A (entrambi i suoi input), G e I, come si può notare dalla Figura 2.6. Con tali ingressi gli output sono tutti settati a 0 ad eccezione del terzo, infatti è possibile vedere in figura il terzo led acceso, al contrario degli altri che risultano spenti.

Attraverso il software *Tiled*, ad ogni cella collocata all'interno di una riga di output è stata assegnata la proprietà "output". Allo stesso modo, ogni cella riguardante un input è stata contrassegnata con la proprietà "input1" o "input2", a seconda che si tratti della prima riga di input del *gate* o della

seconda. Similmente ad ogni *gate* è stata assegnata una proprietà dal valore corrispondente a una lettera dell'alfabeto in maniera crescente da A fino a K. Il primo *gate* sarà quindi identificato dalla lettera A, il secondo da B, il terzo da C e così via fino a K. Un'altra proprietà è stata assegnata a tutti i fori del ramo positivo, attribuendogli valore "positive".

Attraverso questo sistema di proprietà è quindi possibile riconoscere il tipo di cella toccata dall'utente e agire di conseguenza. Questo è fondamentale per creare un legame fra la versione grafica ed implementativa.

Di seguito lo stralcio di codice che identifica il riconoscimento di una certa proprietà di una specifica cella.

Riconoscimento di un foro del ramo positivo

```
1
2  auto tileGid = _positive->getTileGIDAt(firstToConnect);
3  if(tileGid){
4      auto properties =
           _tileMap->getPropertiesForGID(tileGid).asValueMap();
5      if(!properties.empty()){
6                  auto positive = properties["positive"].asString();
7                  if(positive == "true"){
8                          //ramo positivo
9                  }
10         }
11 }
```

Questo codice viene eseguito nel momento in cui l'utente desidera collegare o scollegare un jumper e quindi è necessario verificare se uno degli estremi della congiunzione sia il ramo positivo. Attraverso la funzione getTileGIDAt otteniamo l'identificatore globale (GID, Global Identifier) della cella, relativa al livello riguardare il ramo positivo, presente alle coordinate

firstToConnect, rappresentante il punto relativo al primo tocco dell'utente. Si verifica che vi sia una proprietà in tale cella e successivamente si controlla che sia effettivamente quella cercata e che abbia il valore desiderato. Se tutte queste condizioni sono soddisfatte allora abbiamo trovato la cella di cui avevamo bisogno.

2.2.3 Gioca

Tale sezione è rappresentata da un gioco che nasce con lo scopo di attrarre utenti anche apparentemente disinteressati all'informatica, permettendogli di comprendere come esso sia collegato agli argomenti presenti nelle altre due parti dell'applicazione. L'intento è di rendere appetibile "Play on your CPU" a un target più esteso. Tutti siamo stimolati a giocare, commisuratamente alla nostra età e ai nostri interessi. Ogni persona necessita e percepisce il bisogno di uno svago e tale applicazione vuole fornire uno strumento ludico per sopperire a tale esigenza.

In tale progetto si è sfruttato il concetto di *gamification of learning*, ovvero un metodo di insegnamento che utilizza il gioco per permettere agli studenti di imparare in maniera più semplice e divertente. Lo scopo di tale approccio è quello di dimostrare che l'apprendimento è divertente, proprio come un gioco. Raph Koster, sviluppatore di videogiochi, nel suo libro "A Theory of Fun for Game Design" scrive: "Fun is just another word for learning" ed espone che spesso l'apprendimento risulta noioso per molti studenti perché il metodo di insegnamento utilizzato è sbagliato.

Si prospetta che il pubblico che si appresterà a scaricare "Play on your CPU" sarà di due categorie:

- Utenti interessati agli argomenti trattati, i quali non dimostreranno particolare interesse per la sezione di gioco, in quanto più propensi a interagire con la parte di apprendimento e di sperimentazione.

- Utenti non esperti degli argomenti trattati e probabilmente anche disinteressati ad essi. La speranza è che attraverso il gioco possa nascere una curiosità per i meccanismi che hanno portato al suo sviluppo e un interesse verso il suo funzionamento. Se anche solo una piccola parte di questi utenti si interesserà alle altre due sezioni, il gioco proposto sarà da considerare come un successo.

Tale sezione è rappresentata da un gioco, il cui scopo è quello di riuscire a portare al termine dello scenario il *player*. Per interagire con il *player* l'utente ha a disposizione due tasti. Il primo, quello di sinistra, fornisce al *player* l'abilità di volare. Toccando il secondo tasto, quello di destra, il *player* si alza in volo; quando tale bottone viene rilasciato, il *player* perde quota fino a toccare il suolo. Rilasciando il primo tasto il *player* non sarà in grado di volare e qualsiasi tocco del secondo bottone non produrrà alcun risultato.

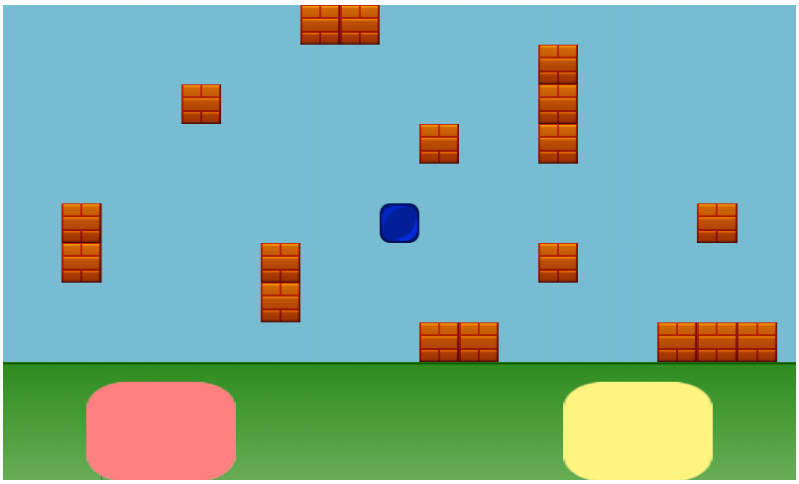

Figura 2.8: Screenshot del gioco

Il *player* è in continuo avanzamento e deve evitare gli ostacoli che trova sulla via alzandosi in volo o abbassandosi. Il gioco termina nel momento

in cui il *player* riesce ad arrivare al termine dello scenario. Alla fine o in caso di *game over* il giocatore potrà ripetere il gioco oppure tornare al menù principale.

Dato il complesso *gameplay*, prima che l'utente inizi a giocare, gli viene data la possibilità di poter consultare un breve tutorial, dove viene spiegata la dinamica di gioco e il suo funzionamento.

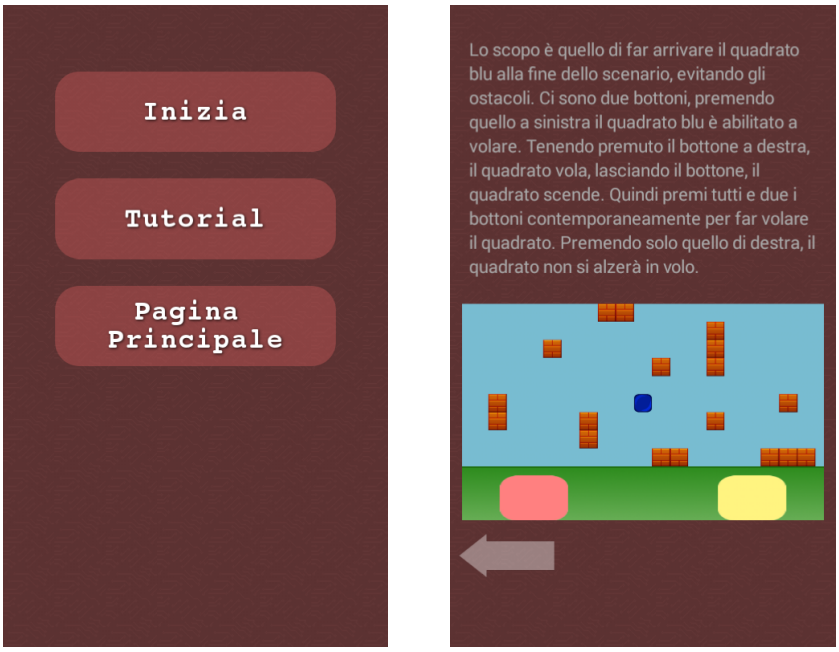

Figura 2.9: Schermata iniziale della sezione gioca e il tutorial del gioco

Tale gioco si ispira a una rete combinatoria dove gli input sono rappresentati dai due bottoni, mentre l'output non è altro che il comportamento del *player*, ovvero la sua posizione sullo schermo. Sono perciò presenti due input, che chiameremo in1 e in2, e un output, che chiameremo out. Quando in1 vale 0, l'output varrà sempre 0 e in2 verrà infatti completamente ignorato. Quando in1 assume valore 1, in2 verrà copiato in output, il quale assumerà perciò il valore di in2 per tutto l'intervallo di tempo in cui in1 è settato a 1.

in1	in2	out
0	0	0
0	1	0
1	0	0
1	1	1

Tabella 2.2: Tabella di verità relativa al comportamento del gioco

Di seguito proponiamo un esempio di forma d'onda:

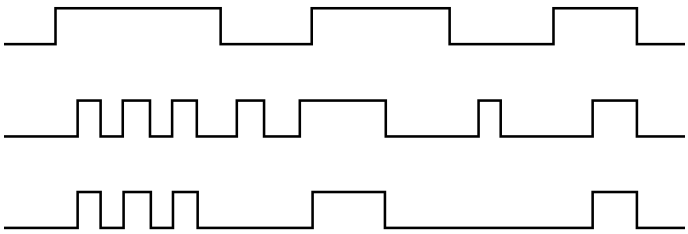

Figura 2.10: Esempio forma d'onda del gioco

Se si osserva con più attenzione il funzionamento di tale rete, si può notare che essa rispecchia perfettamente il comportamento del *gate* AND, ovvero l'output assume valore 1, nel solo caso in cui entrambi gli input valgano 1. In caso contrario l'output sarà sempre 0. Come sappiamo, è possibile realizzare un AND con un circuito di soli due NAND.

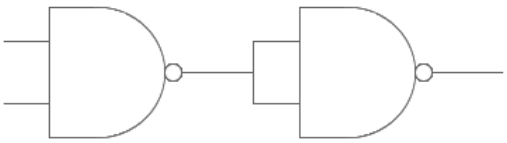

Figura 2.11: AND realizzato attraverso sole porte NAND

Lo scopo principale di questa sezione, oltre che a fornire un divertimento

e intrattenimento per l'utente, è quello di spiegare come queste dinamiche di gioco siano collegate all'informatica. Per questo motivo viene fornita all'utente la spiegazione di come tale gioco sia del tutto similare al comportamento del *gate* AND. Vi è quindi una scena in cui sono presenti tali contenuti e dai quali si può raggiungere il punto della sezione "Impara" dove viene presentato e illustrato il funzionamento della porta AND.

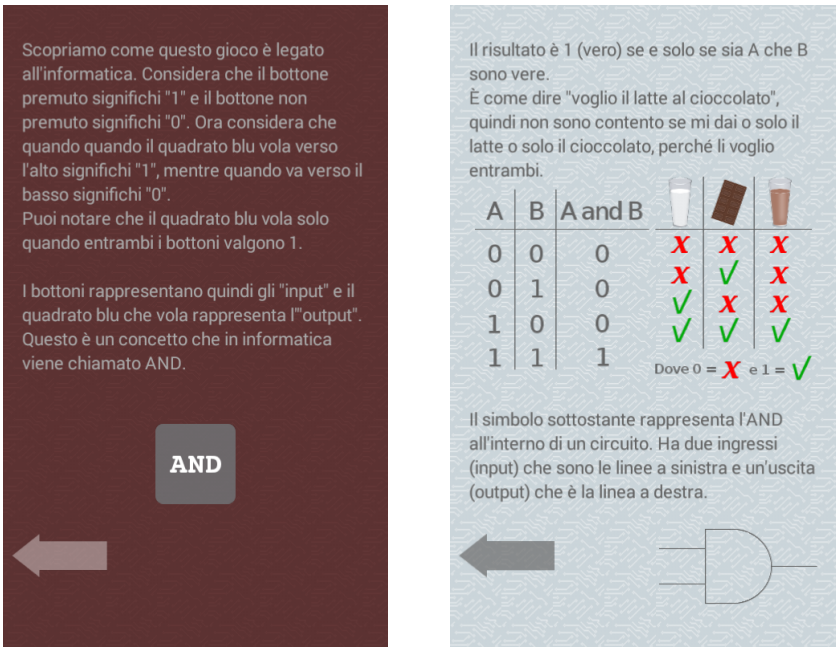

Figura 2.12: Spiegazione del funzionamento del gioco

Anche questa sezione è stata realizzata attraverso la tecnica delle *Tile Map*, attribuendo agli ostacoli una specifica proprietà che permettesse di determinare l'avvenuta collisione del *player* con l'ostacolo.

Sono state create tre differenti classi, una classe "Gioca" responsabile della gestione del gioco e le classi "ButtonSx" e "ButtonDx" responsabili, rispettivamente, della gestione e del tocco del bottone sinistro e di quello destro. I due bottoni, come è già stato detto, rappresentano gli input della rete combinatoria e quindi al momento del tocco essi varieranno il loro valore.

Nel momento in cui il bottone sarà premuto, l'input relativo assumerà valore 1 e tornerà a 0 solo nel momento in cui esso verrà rilasciato.

È stato scelto di inizializzare le due classi relative ai bottoni come sottoclasse di *Sprite*. La *Sprite* è un oggetto di *Cocos2d-x* ed è l'elemento maggiormente utilizzato all'interno dei giochi. Rappresenta un'immagine che è possibile muovere, toccare, modificare, traslare e molto altro ancora. È stato scelto di creare i bottoni come *Sprite* perché questo tipo di oggetti fornisce una gestione del tocco decisamente più efficiente rispetto a qualsiasi altro tipo di oggetto presente nel *framework*.

Le dinamiche di gioco, per come sono state progettate, prevedono che, in specifici momenti, l'utente tocchi entambi i bottoni contemporaneamente. Per questo motivo, è stato necessario abilitare e gestire il *multitouch*. Questa *feature*, in *Cocos2d-x*, è già abilitata di default per Android, a differenza di quanto succede in iOS dove è necessario effettuare delle modifiche[15]. Il *multitouch*, dopo essere stato quindi correttamente abilitato, viene gestito attraverso tre metodi fondamentali:

- OnTouchBegan: fornisce il punto in cui è stato toccato lo schermo e permette di effettuare delle operazioni al momento di inizio del tocco. In questo caso specifico, non appena l'utente tocca il bottone, l'input relativo assumerà valore 1. Verrà inoltre modificato il bottone con un colore più intenso, in maniera tale da dare un *feedback* all'utente dell'effettiva avvenuta del tocco.

- OnTouchMoved: questo metodo viene invocato nel momento in cui l'utente sposta il dito mentre sta effettuando il tocco. Fornisce il punto in cui lo schermo viene toccato e quindi permette di implementare facilmente operazioni tipo *drag and drop* o *slide*, gesti frequenti nelle applicazioni mobili. In questo specifico caso non è stato utilizzato questo metodo, in quanto non necessario.

[15] `http://www.cocos2d-x.org/wiki/How_to_Enable_Multi-Touch`

- OnTouchEnded: fornisce il punto di fine del tocco e permette di effettuare operazioni nel momento in cui il dito viene sollevato dallo schermo. In questo specifico caso, nel metodo viene risettato a 0 il relativo input e viene ripristinata l'immagine originale del bottone, sempre per poter dare un *feedback* all'utente.

Vediamo lo stralcio di codice relativo alle operazioni effettuate all'inizio del tocco e quindi l'implementazione del metodo OnTouchBegan.

Riconoscimento del tocco di un bottone

```
1  touchListener->onTouchBegan = [&](cocos2d::Touch* touch,
       cocos2d::Event* event){
2          auto target = event->getCurrentTarget();
3
4          Point locationInNode =
               target->convertToNodeSpace(touch->getLocation());
5          Size s = target->getContentSize();
6          Rect rect = Rect(0, 0, s.width, s.height);
7
8          if (rect.containsPoint(locationInNode)){
9                  _sxbutton->setTexture (TextureCache ::
                       sharedTextureCache()->
                       addImage("game_buttonsx-on.png"));
10                 input1 = 1;
11                 return true;
12         }
13 };
```

touchListener rappresenta un *listener*, ovvero un oggetto responsabile di rilevare le interazioni dell'utente con la *Sprite* e quindi permette di riconoscere il momento del tocco. Il metodo getCurrentTarget[16] restituisce

[16] http://www.cocos2d-x.org/reference/native-cpp/V3.0alpha0/d4/d7a/classcocos2d_1_1_event.html

il target corrente di un evento, il quale verrà poi convertito in un punto - che corrisponderà a quello che l'utente ha toccato - attraverso il metodo convertToNodeSpace.

Successivamente viene creato un rettangolo contenente l'area di tocco ed avviene una verifica dei punti dell'area toccata. Rilevato quindi il tocco avviene la modifica dell'immagine e l'input relativo viene settato a 1. Un codice similare è presente in entrambe le classi relative ai bottoni, nonché nella funzione OnTouchEnded che è stata introdotta precedentemente.

2.3 Problematiche e difficoltà riscontrate

Una delle principali difficoltà riscontrate è stata quella di dover creare un'interfaccia grafica dell'applicazione, in special modo per la sezione "Sperimenta", ricca di elementi grafici. Si è cercato di fornire una rappresentazione di una *breadboard* fisica che fosse il più reale possibile. Questo ha portato a riscontrare una certa difficoltà dovuta al fatto di dover disegnare elementi in base al tocco dell'utente, si ci riferisce in special modo alla creazione dei *jumper*. Non era infatti possibile prevedere graficamente ogni singola combinazione di collegamenti o ogni possibile inclinazione e lunghezza dei *jumper* utilizzando le *Tile Map*. Infatti sarebbe stato necessario creare un *tileset* troppo grande da gestire.

La scelta che è stata fatta per risolvere tale problematica, è stata quella di non disegnare mai i *jumper* in maniera obliqua, ma solamente in maniera parallela agli assi cartesiani. Questo significa che ogni volta che l'utente prova a collegare due punti non allineati secondo tale criterio, l'applicazione li allineerà automaticamente.

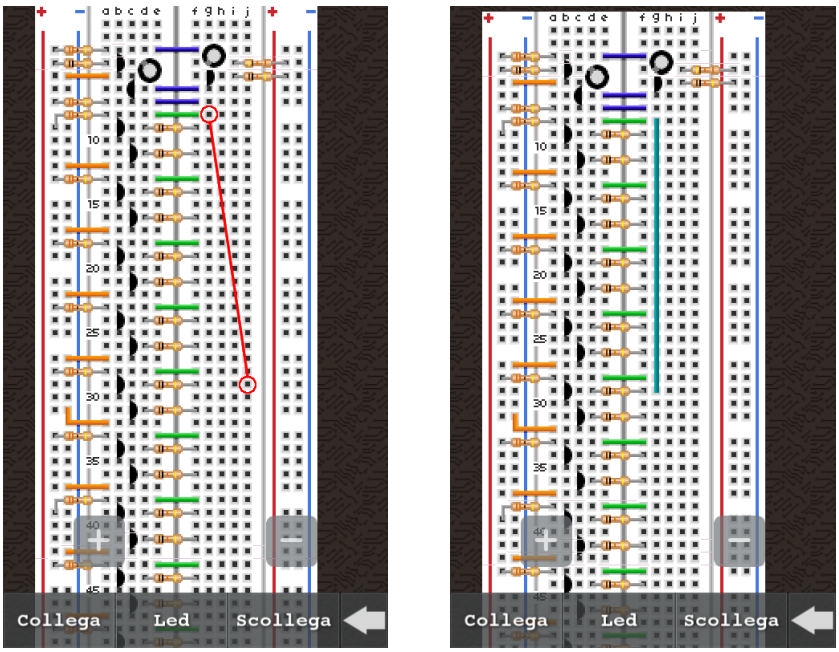

Figura 2.13: Allineamento dei *jumper*

In figura, i punti di tocco da parte dell'utente sono stati cerchiati di rosso. Nella seconda immagine (quella di destra) si puè vedere come viene effettuato in realtà il collegamento.

Questa è stata considerata una soluzione accettabile non soltanto perché non cambia in alcun modo il risultato dei valori in input e output delle porte logiche, lasciando quindi immutate le scelte fatte dall'utente, ma anche perché fornisce una rappresentazione molto ordinata della *breadboard*, evitando eventuali incroci dei *jumper*.

Un'ulteriore difficoltà riscontrata è stata quella di dover implementare un'applicazione funzionante su tutti i tipi di dispositivi ed era quindi indipensabile che fosse adattata a ogni risoluzione di schermo.

La soluzione che è stata ritenuta più adatta è stata quella di utilizzare

un'API di *Cocos2d-x* per il supporto della multirisoluzione[17]. Il funzionamento di tale metodo prevede la divisione delle risorse dell'applicazione in diverse *directory*, contenenti tutte le stesse risorse, ma con grandezze differenti. L'applicazione riconoscerà poi a *runtime* la risoluzione del dispositivo e caricherà le risorse contenute nella directory relativa alla grandezza dello schermo.

Reindirizzamento delle risorse

```
typedef struct tagResource{
    cocos2d::CCSize size;
    char directory[100];
}Resource;

static Resource smallResource = { cocos2d::CCSizeMake(320, 480),
    "small" };
static Resource mediumResource = { cocos2d::CCSizeMake(480, 800),
    "medium" };
static Resource largeResource = { cocos2d::CCSizeMake(768, 1024),
    "large" };
static Resource superLargeResource = { cocos2d::CCSizeMake(1536,
    2048), "superlarge" };
static cocos2d::CCSize designResolutionSize =
    cocos2d::CCSizeMake(480, 800);
```

Come è comprensibile dal codice sopra riportato, sono presenti quattro differenti cartelle nella *directory Resources*:

- small: contiene le risorse adatte alla risoluzione 320x480;
- medium: contiene le risorse adatte alla risoluzione 480x800;
- large: contiene le risorse adatte alla risoluzione 768x1024;

[17] `http://www.cocos2d-x.org/wiki/Multi_resolution_support`

- superlarge: contiene le risorse adatte alla risoluzione 1536x2048;

Le risoluzioni sono presentate in questo formato in quanto l'applicazione è stata interamente sviluppata in *portrait* e non in *landscape*. Viene poi specificata la risoluzione con cui l'app è stata testata e progettata. Successivamente verrà rilevata la grandezza dello schermo e caricata la *directory* corretta.

Attualmente, nonostante l'applicazione sia disponibile su qualsiasi dispositivo, risulta ottimizzata solamente per la risoluzione di uno schermo 480x800, in quanto è stato possibile testare "Play on your CPU" solamente su un cellulare con tale risoluzione.

Capitolo 3

Conclusioni e sviluppi futuri

È stata realizzata un'applicazione mobile funzionante e ricca di contenuti. La speranza è che possa essere uno strumento guida per chi si avvicina o vuole approfondire i concetti dell'informatica. Nonostante questa premessa, è sicuramente possibile apportare migliorie. Elenchiamo di seguito alcune aggiunte che si prevedono di effettuare:

- Il più importante e significativo dei cambiamenti è sicuramente quello di offrire all'utente la possibilità di salvare un certo numero di sessioni relative alla sezione "Sperimenta", in maniera tale che l'utente possa continuare a lavorare su un circuito precedentemente creato. Inoltre sarebbe interessante poter caricare certi prestabiliti circuiti, nello specifico, quelli illustrati nella sezione "Impara".

- Si voglio apportare migliorie grafiche, soprattutto per quanto riguarda la sezione "Sperimenta", in maniera da fornire maggiori *feedback* all'utente e guidandolo all'interno della creazione dei circuiti.

- Per fornire una trattazione dei concetti più esaustiva e comprensibile si vogliono aggiungere più esempi interattivi, in modo da fornire un apprendimento più interessante e stimolante per l'utente.

- Come già detto, l'applicazione è eseguibile su qualsiasi dispositivo, tuttavia risulta ottimizzata solamente per la risoluzione di schermo 480x800. L'intenzione futura è quella di riuscire a testare l'applicazione su più dispositivi e adattare "Play on your CPU" a dispositivi di qualsiasi grandezza.

- Vi è l'intenzione di creare un'esperienza di gioco che sia il più piacevole possibile per l'utente. Per questo motivo non solo si desidera migliorare il gioco a livello grafico, ma anche introdurre differenti livelli di difficoltà ed eventualmente un *gameplay* più particolareggiato.

Bibliografia

[1] Tim Bell, Ian H. Witten, Mike Fellows, *Computer Science Unplugged*, 2005 http://csunplugged.org/

[2] Andrew S. Tanenbaum, *Architettura dei Calcolatori un approccio strutturale*, quinta edizione, Pearson Prentice Hall, 2006

[3] Raph Koster, *A Theory of Fun for Game Design*, seconda edizione, O'Reilly Media, 2014

[4] M. Morris Mano, Charles R. Kime *Reti Logiche*, Prentice Hall, 2008

[5] "Raspibo", raspibo.org http://www.raspibo.org/wiki/index.php/Pagina_principale

[6] "Cocos2d-x", 2010 - 2015 Cocos2d-x.org http://cocos2d-x.org/

[7] "Sonar Systems", 2015 http://www.sonarlearning.co.uk/

[8] "Ray Wenderlich", 2015 Razeware LLC. http://www.raywenderlich.com/

[9] "Raspberry Pi", Raspberry Pi Foundation http://www.raspberrypi.org/

[10] "Arduino", 2015 Arduino http://www.arduino.cc/

[11] "Tiled Map Editor", 2008-2014 Thorbjørn Lindeijer http://www.mapeditor.org/

[12] "Scratch", Lifelong Kindergarten Group https://scratch.mit.edu/

[13] "CoderDojo", CoderDojo Foundation https://coderdojo.com/

[14] "MIUR", 2009 http://www.istruzione.it/

Ringraziamenti

Ringrazio i miei genitori, per avermi permesso di studiare e per aver avuto fiducia in me. Li ringrazio per avermi cresciuta in questo modo, per avermi insegnato ad essere onesta e rispettosa, e li ringrazio per avermi sempre permesso di fare ciò che amavo, senza pormi limiti.

Ringrazio la mia amica Lisa, per aver letto questa tesi e per essermi stata accanto, soprattutto in questi ultimi mesi. Sono fortunata ad aver avuto te ad aiutarmi. Non sei così male come pensi a dare consigli.

Ringrazio le mie migliori amiche: Francy (grazie anche per aver letto la mia tesi), Flavia e Mela. Grazie per le serate passate a ridere fino ad avere le lacrime agli occhi. Grazie per esserci nei momenti bui e grazie perché non mi giudicate mai. Siete le amiche migliori che io possa sperare di avere.

Ringrazio tutti i miei amici (che non nomino per non dimenticare nessuno) per i momenti in cui mi siete stati vicini in questi tre anni di università. Grazie per non aver fatto caso ai miei momenti di sclero e chiedo scusa per ciò che ho detto durante il periodo di esami.

Ringrazio il professor Renzo Davoli, per avermi supportato e sopportato. Per essere stato una guida fondamentale e per avermi appoggiato in questo progetto. Lo ringrazio per la sua disponibilità e pazienza senza confini e per aver sempre saputo trovare una risposta alle mie domande.

Ci sono altre persone che probabilmente dovrei ringraziare, ma mi limiterò a un "grazie" generale, dedicato a quelle persone che mi hanno aiutato ad essere quello che sono.

Giulia Naponiello