UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

FACOLTÁ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA INFORMATICA

PROVA FINALE

Realizzazione di un'applicazione per il recupero delle abilità linguistiche di bambini Late Producers

Pietro Buzzega

RELATORE

Prof. Nicola Bicocchi

ANNO ACCADEMICO 2016 - 2017

Indice

1. Introduzione

Nella letteratura internazionale, i bambini che nei primi anni di vita presentano un ritardo nel vocabolario espressivo, in assenza di patologie neurologiche e sensoriali, vengono definiti Parlatori Tardivi (PT). Nonostante si tratti di una definizione clinica alquanto generica e al cui interno cade un'ampia variabilità di profili evolutivi, la maggior parte dei bambini PT (circa il 70%; Bello et al., 2014) manifesta un ritardo circoscritto alle abilità espressive: questi ultimi sono detti Late Producers (LPs).
I bambini LPs, ed in generale i PT, risultano a rischio di Disturbi Specifici del Linguaggio (DSL) e dell'Apprendimento (DSA) in età successive; per tale ragione, costituisce una nuova sfida per ricercatori e clinici l'individuazione di interventi precoci e di prevenzione che favoriscano il recupero delle abilità linguistiche di questi bambini.
Ad oggi, gli studi a riguardo sono quasi del tutto assenti sul panorama internazionale, in particolare in riferimento alla sperimentazione di interventi informatizzati; il presente progetto intende procedere in questa direzione.

1.1 Late Producers

Come citato sopra, i bambini Late Producers sono bambini PT che manifestano un ritardo circoscritto alla componente espressiva del linguaggio: in particolare, ci si riferisce ad un ritardo "primario", ovvero non associato ad altri disturbi dello sviluppo (assenza di deficit neurologico, sensoriale, cognitivo e relazionale).

L'abilità di coarticolare i suoni in una sequenza fluida che genera una parola deriva dalla maturità delle capacità di rappresentazione fonologica e di programmazione motoria. Il livello di sviluppo di tali capacità dipende dalla frequenza d'uso della parola stessa (sia in termini di ascolto che di produzione) e dalla sua complessità, ma anche dalle capacità del bambino di cogliere rapidamente la struttura fonologica per immagazzinarla correttamente e poi recuperarla altrettanto velocemente (Pinton et al., 2014).
Il buon funzionamento di sistemi di memoria a breve (MBT) e lungo termine (MLT) appare indispensabile per assolvere a questi compiti. In particolare, una sotto-componente della MLT, la memoria procedurale, appare particolarmente importante nell'acquisizione e nell'uso di regole fonotattiche e nell'accesso automatico (quindi rapido) alle strutture fonologiche, processi alla base dell'espansione del vocabolario.
E' dunque ragionevole credere che un bambino con capacità di memoria limitata abbia

bisogno di un numero maggiore di esposizioni e ripetizioni. Inoltre, per potenziare i meccanismi di apprendimento fonologico, quindi lo sviluppo di nuove parole in bambini LPs, può essere utile esporre questi bambini ad un esercizio ripetuto con le parole, sia in percezione che in produzione, in cui si tenga conto delle caratteristiche fonetico-fonologiche delle parole stesse.

1.2 Interventi evidence-based esistenti

Gli studi volti a sperimentare l'efficacia di interventi in bambini LPs, e più in generale PT, sono ad oggi poco numerosi.

Un primo studio propone l'approccio definito "*watch and see*" (Paul, 1996) che consiste nell'offrire a bambini PT un attento monitoraggio dell'evoluzione spontanea del loro linguaggio tramite controlli regolari da parte del clinico (ogni 3-6 mesi in base alla gravità del ritardo linguistico). In questo caso non si tratta di un vero e proprio intervento, anche se in genere a questo tipo di monitoraggio si vanno ad aggiungere consigli ai genitori su come promuovere il linguaggio del proprio bambino.

Esistono poi programmi di intervento indiretto rivolti alle persone significative che si occupano del bambino (ad es. genitori, insegnanti) e che vanno sotto il nome di *General Language Stimulation*. Questi interventi mirano a favorire nell'adulto comportamenti responsivi, di attenzione congiunta, e comportamenti linguistici che modellano ed espandono la produzione verbale del bambino (Bonifacio, 2014).

Tra gli approcci che prevedono invece un intervento diretto al bambino ritroviamo la *Focused Language Stimulation* ed il *Milieu Teaching*. Questi interventi diretti possono essere erogati a casa o a scuola, dai genitori o da operatori scolastici addestrati da specialisti clinici (interventi mirati), oppure direttamente dallo specialista clinico (interventi specialistici). Essi comprendono programmi molto strutturati negli obiettivi e nelle attività, precisi target linguistici da stimolare e da imitare. In bambini PT con condizioni di ritardo linguistico più marcato, gli interventi diretti appaiono più promettenti rispetto ai precedenti. Tuttavia, gli studi a riguardo risultano ad oggi così poco numerosi da non poter trarre conclusioni certe circa i loro effetti a lungo termine.

In ambiente clinico è risaputo come il mancato intervento precoce in bambini che presentano ritardo del linguaggio può avere effetti negativi a cascata sullo sviluppo successivo. In letteratura sono sempre più descritti esiti a lungo termine di difficoltà linguistiche precoci, non solo a carico degli aspetti formali e del linguaggio ma anche

relativamente alle abilità di apprendimento e alle competenze relazionali e sociali in età scolare e nei giovani adulti. Tuttavia, in Italia, il Sistema Sanitario Nazionale riesce ad offrire un intervento specialistico immediato per bambini PT con ritardo di linguaggio associato a difficoltà cognitive e/o comunicativo-relazionali, ma per i bambini con ritardo linguistico primario (che ricordiamo essere circa l'80% dei PT) ciò solitamente non è possibile. Per questi ultimi viene utilizzato l'approccio *watch and see* descritto in precedenza, con talvolta interventi diretti sul genitore, ma la scarsità di risorse preclude interventi diretti al bambino. Di qui la necessità di trovare interventi diretti per bambini Late Producers, che siano efficaci ma al contempo sostenibili in termini economici.

1.3 Ipotesi di ricerca

A seguito della realizzazione dell'applicazione, parteciperanno alla sua sperimentazione due gruppi di bambini LPs: un gruppo sperimentale costituito da 50 LPs di 27-30 mesi che, in aggiunta all'approccio watch and see, riceverà l'intervento mirato con l'applicazione; un gruppo di controllo formato da 50 LPs di pari età a cui verrà offerto, come da prassi clinica, l'approccio watch and see.

I bambini saranno reclutati presso le Aziende-Unità-Sanitarie-Locali (AUSL) delle province limitrofe all'Università di Modena-Reggio Emilia.

Ci si attende che questa applicazione possa risultare un utile strumento sia nelle mani del clinico (logopedista, psicologo) che, a "costo zero", può istruire il genitore al suo utilizzo in ambiente domestico, sia nelle mani del genitore che da casa può usufruire di uno strumento per stimolare il vocabolario del proprio bambino. Vi è accordo sul fatto che gli interventi eseguiti in bambini con ritardo linguistico da genitori che ricevono un training di addestramento dai terapisti conducono a risultati simili rispetto a quelli eseguiti da logopedisti (Cable e Domsch, 2011; Law et al., 2004).

Si ipotizza che questo intervento mirato possa avere come effetto a breve termine, a 36 mesi, il potenziamento delle abilità espressive in misura più consistente nel gruppo sperimentale rispetto al gruppo di controllo; nel gruppo sperimentale si attendono inoltre effetti a lungo termine, vale a dire un maggior recupero del ritardo linguistico ed un minor esito in DSL a 48 mesi rispetto al gruppo di controllo.

1.4 Scopo e aspettative

Citando le parole di Bertelli e collaboratori (2014), nell'intervento con questi bambini occorre tenere presente che la competenza che intendiamo sollecitare rappresenta certamente l'obiettivo principale della terapia ma anche la componente più fragile del soggetto, della sua identità di parlante. Pensiamo che il fatto di chiedere al bambino la pronuncia di parole nel suo ambiente di vita e attraverso giochi su tablet possa rendere questo tipo di intervento una "proposta accattivante" e, per questo, anche particolarmente efficace.
Attraverso attività, giochi e storie, l'applicazione mira nello specifico a potenziare la componente percettiva e le abilità di discriminazione fonemica, a stimolare l'acquisizione di nuove parole e a favorirne la pronuncia corretta dal punto di vista fonetico-fonologico.

Ci si aspetta innanzitutto che l'applicazione possa rispondere alla necessità del clinico di poter offrire a questi bambini un intervento mirato; inoltre, essa potrebbe essere la risposta alla richiesta che i genitori rivolgono spesso allo specialista, cioè di strumenti ed esercizi da poter utilizzare a casa che sollecitino lo sviluppo del linguaggio del proprio bambino.
Ci si attende di riscontrare un effetto di questo intervento informatizzato sia a breve che a lungo termine. Più precisamente, ci aspettiamo di evidenziare un maggior recupero del ritardo espressivo nel gruppo sperimentale rispetto a quello di controllo a 3, 6, 9 e 12 mesi dall'inizio dell'intervento ed una minore presenza di DSL a 4 anni.

2. Descrizione dell'applicazione

L'applicazione sarà strutturata in blocchi, dal più semplice al più complesso, in base alla complessità fonetico-fonologica delle parole (ad es. si andrà dal blocco dedicato all'occlusiva "p" fino all'ultimo livello dedicato alla vibrante "r").
Ciascuna sezione di ogni livello sarà strutturata "a gioco" e sarà ricca di rinforzi positivi sulla prestazione in modo da tener vive nel bambino attenzione e motivazione, facilitando così l'apprendimento.

2.1 Menu iniziale

All'apertura dell'applicazione l'utente verrà portato nella schermata di menu iniziale, dove troverà due pulsanti: "Esci" ed "Aggiungi bambino"; il primo, quando premuto, chiederà all'utente la conferma per uscire dall'applicazione, il secondo farà apparire uno spazio vuoto dove inserire il nome del bambino ed al suo fianco un bottone di conferma.
Una volta aggiunto il primo nome il menu cambierà nel seguente modo: verrà aggiunto un pulsante per rimuovere un bambino e tanti pulsanti quanti sono i bambini, con su scritto il loro nome. Premendo su "Rimuovi bambino", l'utente avrà a disposizione un menu a tendina dove potrà selezionare il nome da rimuovere e un tasto di invio. Una volta dato l'invio verrà richiesto di confermare l'operazione, quindi (se la conferma sarà positiva) verrà cancellato il bambino e l'utente sarà riportato alla schermata iniziale.

Fig. 2.1 - Menu iniziale (grafica provvisoria).

2.2 Scelta del mondo

Una volta che l'utente avrà scelto con che bambino iniziare a giocare, la schermata cambierà completamente e verrà generato un mondo con sfondo animato, che passerà dalla notte al giorno scorrendo verso destra.
Da qui si potrà accedere alle impostazioni tramite un tasto a forma di ingranaggio (in alto a sinistra) o tornare indietro grazie ad un altro bottone (in basso) a forma di freccia.
La scelta del mondo sarà legata al tipo di esercizio che si vuole eseguire con il bambino; scorrendo da sinistra a destra si troveranno vari bottoni, uno per ogni mondo, identificati da un'icona che li rappresenta.

Fig. 2.2 - Scelta del mondo (grafica provvisoria).

2.3 Primo mondo (fattoria)*

Dopo aver premuto l'icona della fattoria, si entrerà nel primo mondo. Come al livello precedente, l'utente avrà l'opportunità di scegliere se tornare indietro o modificare le impostazioni, utilizzando i due bottoni già citati. Inoltre, premendone un altro sulla sinistra dello schermo, apparirà un menu con i dettagli del lavoro svolto fino ad ora: il tempo totale trascorso all'interno delle sezioni sottostanti ed il numero totale di volte in cui il genitore

---

**si descriverà solamente il primo mondo poiché, ad oggi, è l'unico che è stato terminato.*

ha toccato la parte destra dello schermo (come si spiegherà in seguito, questo numero corrisponde al numero delle parole pronunciate dal bambino). La schermata appena descritta sarà utile allo specialista per capire quanto il mondo corrente è stato utilizzato.
Per scendere ancora di livello ed arrivare all'esercizio vero e proprio, l'utente dovrà premere uno dei sei bottoni disponibili, che si riferiscono ad animali (od oggetti) diversi.

Fig. 2.3 - Primo mondo, fattoria (grafica provvisoria).

2.4 Schermata di esercitazione

Premendo uno dei pulsanti rappresentati da icone di animali (od oggetti) si entrerà nell'ultimo livello dell'applicazione, dove si terrà la vera e propria esercitazione.
In quest'ultima schermata sarà rappresentato l'ambiente della fattoria, con un bambino sulla sinistra e un animale (od oggetto) all'interno della stalla; anche qui saranno presenti i bottoni per tornare indietro o modificare le impostazioni.
L'esercitazione funziona in questo modo: il genitore (o tutore) cercherà di far parlare il bambino, imitando il verso dell'animale o il rumore dell'oggetto. Appena il bambino si sforzerà per far uscire un suono, che assomigli a quello che ha pronunciato il tutore, quest'ultimo toccherà la parte più a destra dello schermo e l'animale (fino ad ora invisibile) uscirà dalla stalla con un balzo: questa animazione dovrebbe premiare il bambino per lo sforzo che ha fatto, invogliando a ripeterlo. L'esercizio andrà avanti allo stesso modo fino

alla terza volta: infatti, dopo tre volte che il bambino effettua uno sforzo per parlare, è necessario premiarlo con un'animazione più consistente.
In questo caso, l'animazione finale sarà costituita da un coniglio che, partendo dalla parte destra dello schermo (non visibile inizialmente) correrà verso il bambino, aspetterà che esso gli dia la carota che ha in mano, dopo di che tornerà indietro scappando verso il luogo dal quale era partito inizialmente. Nel frattempo, l'animale citato in precedenza salterà sul posto ed il bambino farà lo stesso, muovendo le braccia.

Fig. 2.4 - Attimo in cui il coniglio prende la carota (grafica provvisoria).

3. Descrizione tecnica

In questa sezione si procederà alla descrizione del progetto software, in riferimento agli strumenti e ai metodi di stesura del codice che sono stati utilizzati.

3.1 Strumenti utilizzati

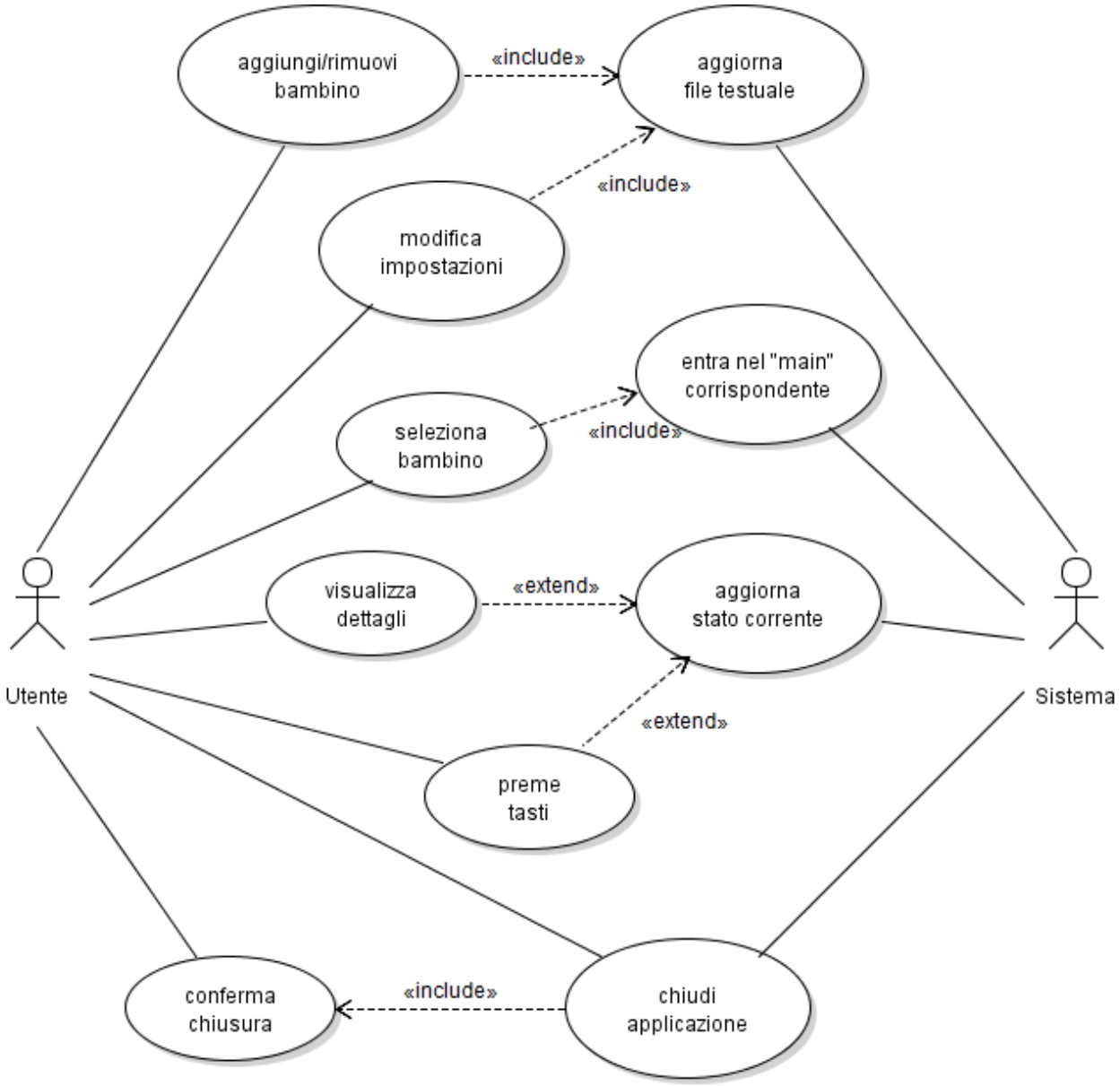
L'applicazione è interamente scritta in linguaggio Java, eccetto per le "Skin" scritte in JSON che si possono trovare online al seguente indirizzo: https://github.com/czyzby/gdx-skins. Come compilatore è stato utilizzato Android Studio, che insieme al plugin Gradle forma l'ambiente di sviluppo; il gioco è basato sul framework LibGDX.

3.2 Terminologia

- SpeakUp: classe principale dell'applicazione, estende ApplicationAdapter (che è una classe definita nella libreria).
- Super: classe astratta che definisce lo stato di gioco, quindi antenata di tutti gli stati.
- WorldManager (abbr. WM): classe che gestisce gli stati di gioco attraverso una pila.
- Init: classe che definisce lo stato di menu iniziale (vedi punto 2.1).
- Main: classe che definisce lo stato di "scelta del mondo" (vedi punto 2.2).
- Sub: classe che definisce lo stato di "mondo" (vedi punto 2.3).
- Last: classe che definisce lo stato di "esercitazione" (vedi 2.4).
- Animation: classe animazione (ad esempio, il coniglio che corre o il bambino che salta).
- Attore: qualsiasi elemento presente sulla schermata (ad esempio bottoni o scritte).
- Stage: classe di LibGDX; contenitore invisibile per gli attori, può gestirli con i suoi metodi.
- Table: classe di LibGDX; contenitore invisibile per gli attori che gestisce il loro inserimento con metodi che ne determinano automaticamente la posizione.
- ScrollPane: classe di LibGDX: contenitore scorrevole per gli attori.
- Label: classe di LibGDX; etichetta, qualsiasi scritta che non sia all'interno di un bottone.
- Livello: quanto ci si trova in profondità nel gioco (es. Init = liv. 0, Main = liv. 1 e così via).

3.3 Use Case Diagram

L'applicazione deve permettere le seguenti macro-funzionalità:



Per "stato corrente" si intende qualsiasi classe che estenda "Super" che stia venendo aggiornata nell'istante considerato.

Siccome i dati da mantenere ad applicazione chiusa non sono complicati, numerosi o sensibili, si è optato per conservarli in un file testuale non criptato. Vi sarà dunque un file per le impostazioni (ad esempio volume, effetti sonori o voce parlante) che conterrà anche il numero dei bambini registrati, il loro nome ed eventualmente il cognome.

Si considera chiave primaria la stringa inserita dall'utente al momento della registrazione del bambino; non può essere vuota.

Oltre ad esso sarà creato un file per ogni bambino registrato, contenente tutti i dettagli (es. tempo di gioco o numero di parole pronunciate).
Questo sistema di file è facilmente gestibile tramite la classe Preferences di LibGDX, che permette di inserire, eliminare o modificare valori tramite i suoi metodi; ogni valore ha una parola chiave che lo identifica unicamente.

```
<properties>
<entry key="musicVol">70.0</entry>
<entry key="child2">Giacomo Ferrari</entry>
<entry key="voiceVol">70.0</entry>
<entry key="child1">Giuseppe Verdi</entry>
<entry key="n_children">3</entry>
<entry key="child0">Mario Rossi</entry>
<entry key="soundVol">70.0</entry>
</properties>
```

Fig. 3.3 - Esempio di file testuale con tre bambini registrati.

3.4 Design pattern

Come accennato in precedenza, per il progetto corrente si è fatto uso del design pattern "State". Questa scelta è stata fatta poiché viene aggiornata soltanto una classe per volta e tutti gli stati hanno bisogno di essere aggiornati tramite le stesse funzioni.
La classe astratta Super funge da modello per tutti gli stati: al suo interno sono dichiarati i quattro metodi principali: handleInput(), render(), dispose() ed update().
La classe WorldManager è la classe di gestione degli stati, utilizza una pila; con il metodo set() posso liberare la memoria occupata dallo stato corrente ed istanziarne uno nuovo.
Inoltre, in WorldManager sono definiti altri metodi chiamati update() e render(); ad ogni aggiornamento, SpeakUp chiama wm.update() e wm.render(), che al loro interno chiamano a loro volta i metodi omonimi dello stato corrente.

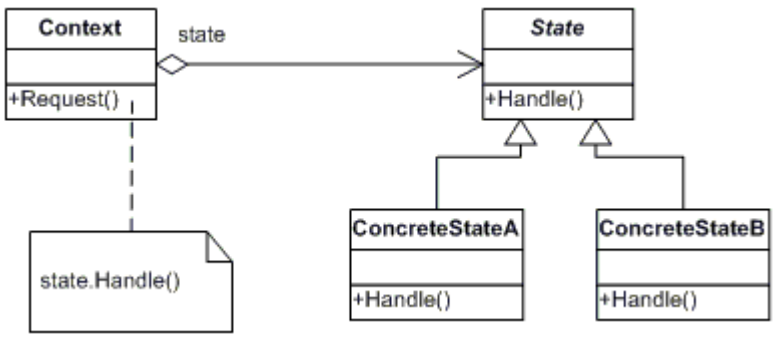


Fig. 3.4 - Generico class diagram del design pattern State.

3.4.1 Codice base di Super:

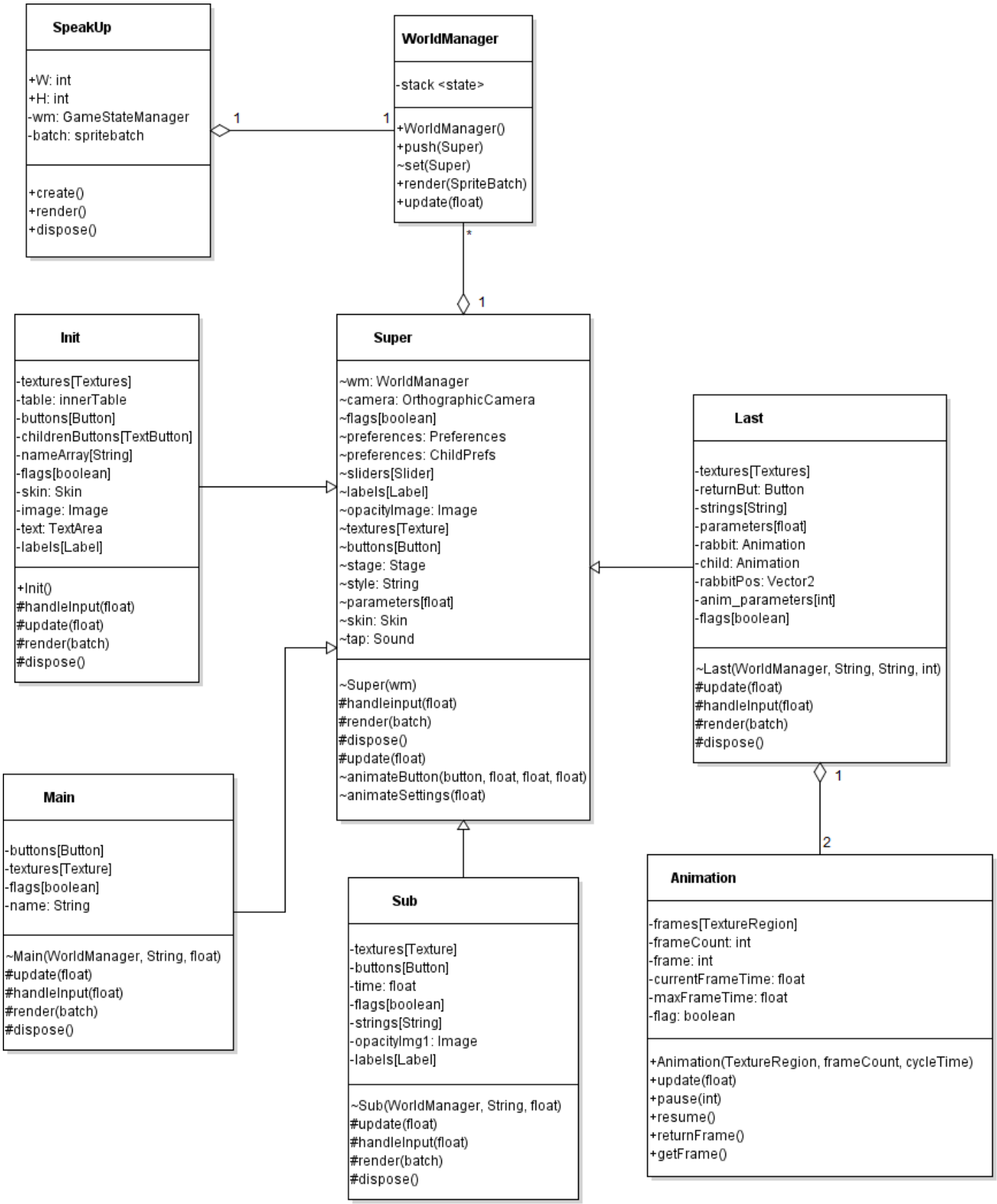
```
abstract class Super {

    //camera and world manager
    OrthographicCamera cam;
    WorldManager wm;

    Super(WorldManager wm) {
        this.wm = wm;

        //initialize camera
        cam = new OrthographicCamera();

    }
    //abstract methods to model classes below this one
    protected abstract void handleInput(float dt);

    protected abstract void update(float dt);

    protected abstract void render(SpriteBatch sb);

    protected abstract void dispose();

}
```

3.4.2 Codice di WorldManager:

```
public class WorldManager {
    //states are organized in a stack
    private Stack<Super> worlds;

    public WorldManager() {
        //creating a new stack of states
        worlds = new Stack<Super>();
    }

    public void push(Super world) {
        //pushing a state into the stack
        worlds.push(world);
    }

    void set(Super world) {
        //popping and free memory of the previous state
        worlds.pop().dispose();
        worlds.push(world);
    }

    public void update(float delta) {
        //update current state
        worlds.peek().update(delta);
    }

    public void render(SpriteBatch sb) {
        //render current state
        worlds.peek().render(sb);
    }
}
```

3.5 Class Diagram

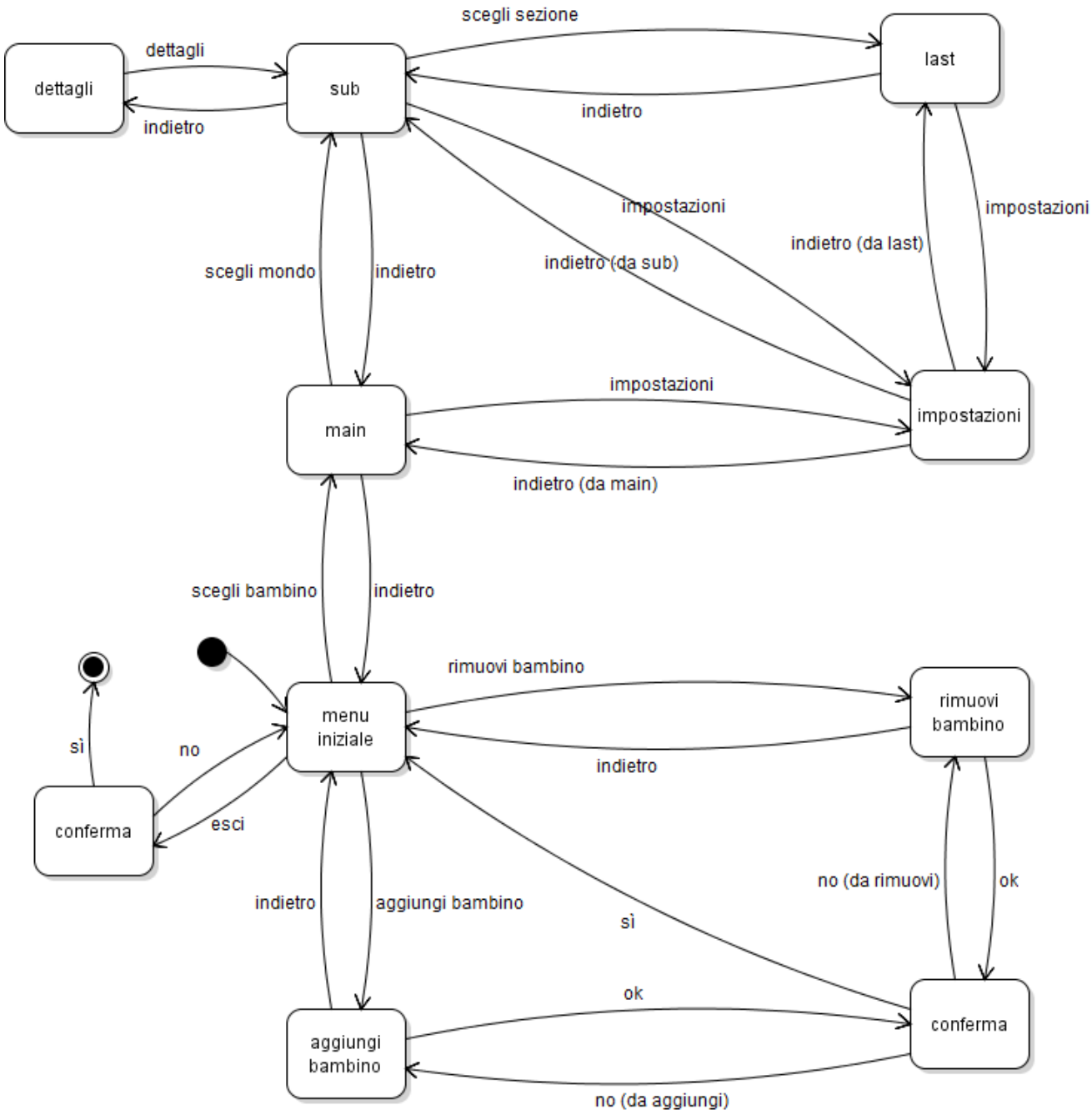
Si riporta di seguito lo schema secondo cui sono state organizzate le classi:



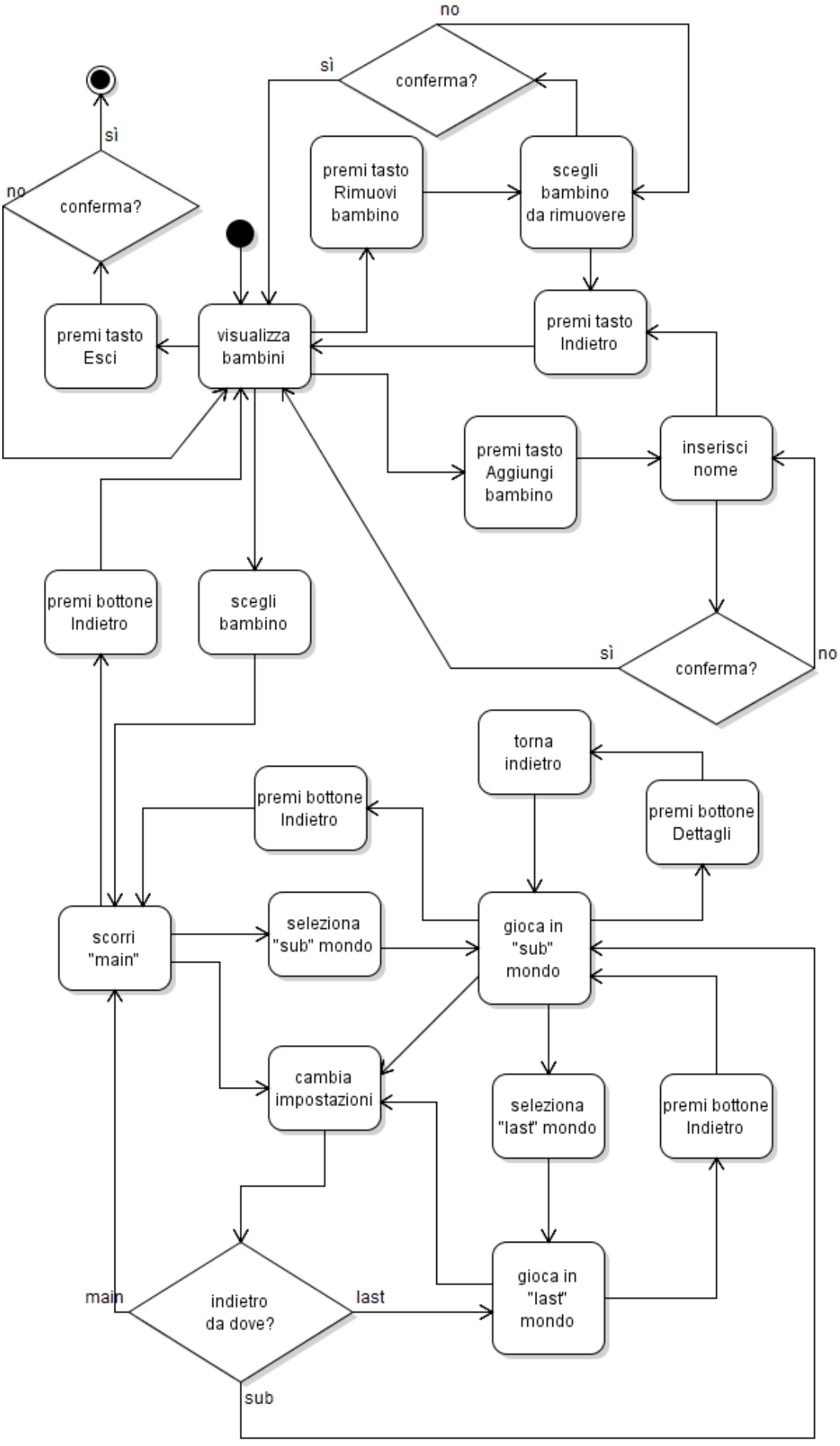
3.6 State diagram

Si noti che tutti gli attori delle impostazioni sono definiti in Super, quindi vengono creati all'inizio di ogni stato e sono raggiungibili da tutti gli stati. La scrittura corretta del diagramma prevedrebbe di considerare "impostazioni" come tre stati differenti, ma si è preferito disegnarlo in questo modo per questioni di chiarezza.

Lo stato di "conferma", raggiungibile da "rimuovi bambino" e aggiungi bambino", presenta lo stesso problema, quindi si è optato per la medesima soluzione.

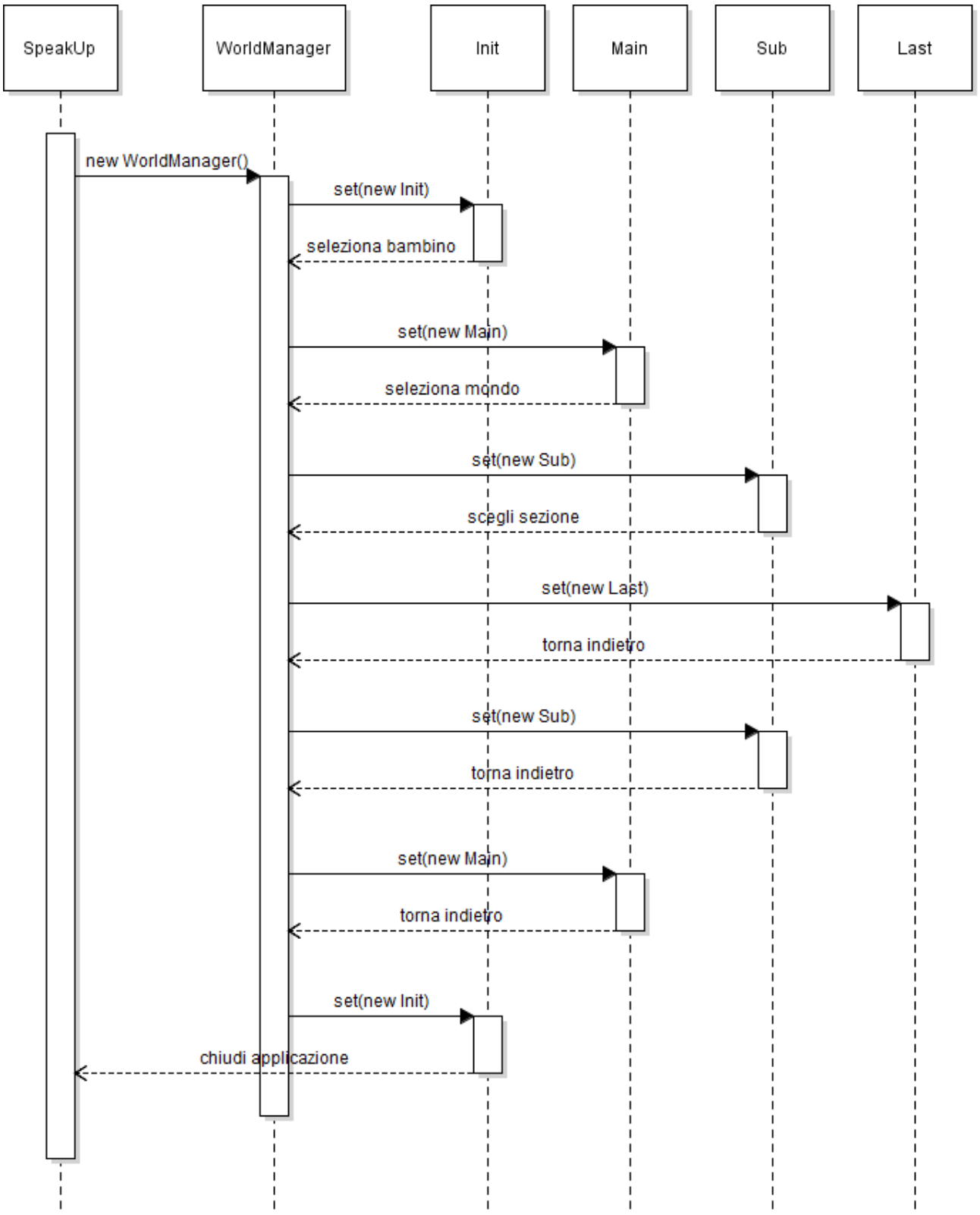


3.7 Activity diagram



3.8 Sequence diagram

Di seguito si riporta il sequence diagram dell'applicazione per un utente che, partendo dal menu iniziale, discende tutti i livelli per esercitarsi in una sezione di un mondo, dopo di che torna indietro e chiude l'applicazione.



4. Struttura ed organizzazione del codice

Il codice è stato scritto cercando di minimizzare il numero di classi, pur mantenendo la leggibilità. Sebbene sia stato terminato solo il primo mondo, per adesso si è riusciti nell'intento, avendo soltanto una classe per livello.
Init e Main sono unici, ma i mondi e le sezioni sono vari e potrebbero dover contenere immagini di sfondo o bottoni diversi: si è risolto il problema ponendo in input ad ogni mondo una stringa che lo caratterizza (es. "farm" per la fattoria) e, allo stesso modo, un numero per ogni sezione. All'interno di ogni istanza, le immagini vengono determinate dalla combinazione stringa-numero; non solo le immagini ma anche altri parametri possono cambiare in relazione a questi dati (es. dimensione o posizione dell'animale).

4.1 Super

La classe Super funge da modello per tutti gli stati di gioco. Essa definisce tutti gli attori che formano le impostazioni con la relativa animazione; definisce anche il metodo per animare ogni bottone nel gioco, in modo tale che sia utilizzabile in ogni sua classe figlia.
Le impostazioni sono sempre presenti in ogni stato, ma si trovano oltre il bordo sinistro dello schermo quindi non sono visibili; ogni volta che si preme il pulsante a forma di ingranaggio esse "entrano" nella schermata.

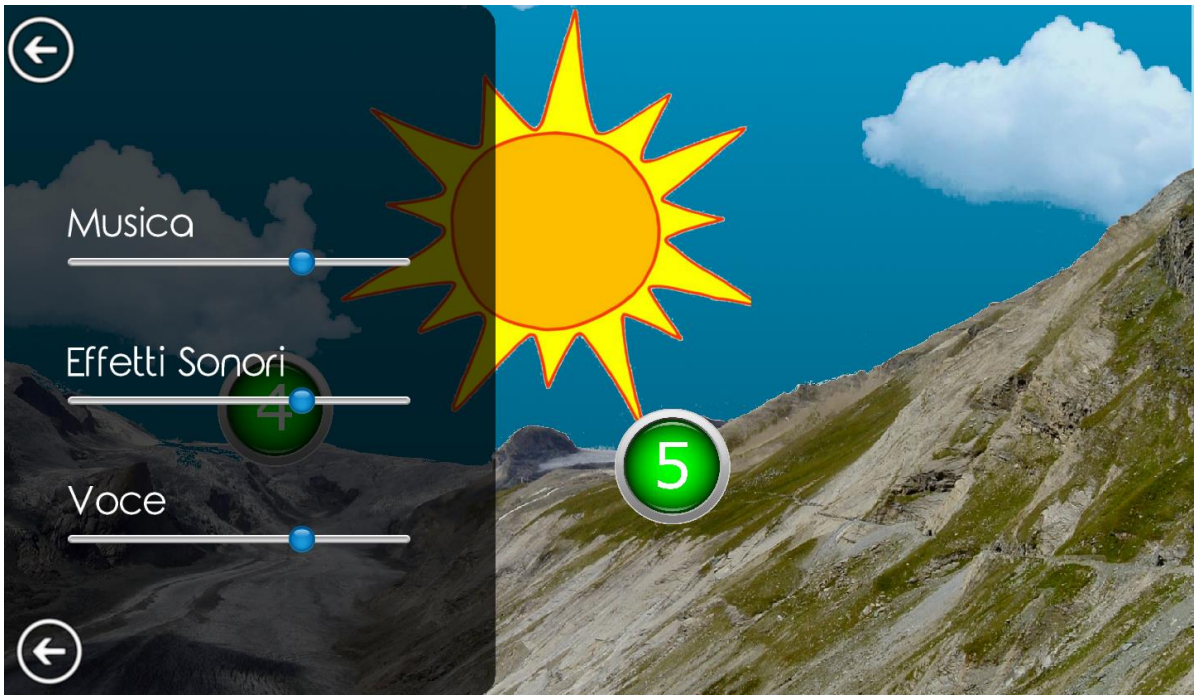


Fig. 4.1 - Impostazioni aperte dal Main (grafica provvisoria).

4.2 Init

Il menu iniziale deve chiedere all'utente la scelta del bambino con cui si vuole cominciare l'esercitazione e permettere la gestione del file dove si conservano i nomi; si è optato per dedicare un bottone testuale ad ogni bambino ma è sorto immediatamente un problema: potrebbe non essere sufficiente una sola schermata.

Vengono quindi definiti una Table interna, che conterrà tutti i bottoni (con i relativi nomi), uno ScrollPane, che conterrà la Table, e una seconda Table all'esterno che servirà per contenere lo ScrollPane. In questo modo, quando i bottoni diventeranno troppi per essere contenuti in una sola schermata, si potrà utilizzare il dito (o il mouse, da desktop) per scorrere verso destra e visualizzarli.

Non tutti i bottoni devono essere scorrevoli, ad esempio il pulsante "Esci" e tutti gli attori dei menu "Aggiungi bambino" e "Rimuovi bambino" devono rimanere fissi; si utilizza dunque una Stage, distaccata dalle Table menzionate in precedenza, che contiene tutti gli attori che non cambiano la loro posizione a seconda dello scorrimento.

I menu appena citati sono costruiti semplicemente aggiungendo un'immagine opaca alla Stage, grande quanto lo schermo, che rende quindi impossibile all'utente premere i bottoni che sono stati aggiunti prima di essa. Dopo averlo fatto si aggiungerà ogni altro pulsante necessario. Allo stesso modo, i menu di conferma aggiungono un'ulteriore opacità alla schermata bloccando tutto ciò con cui si poteva interagire prima.

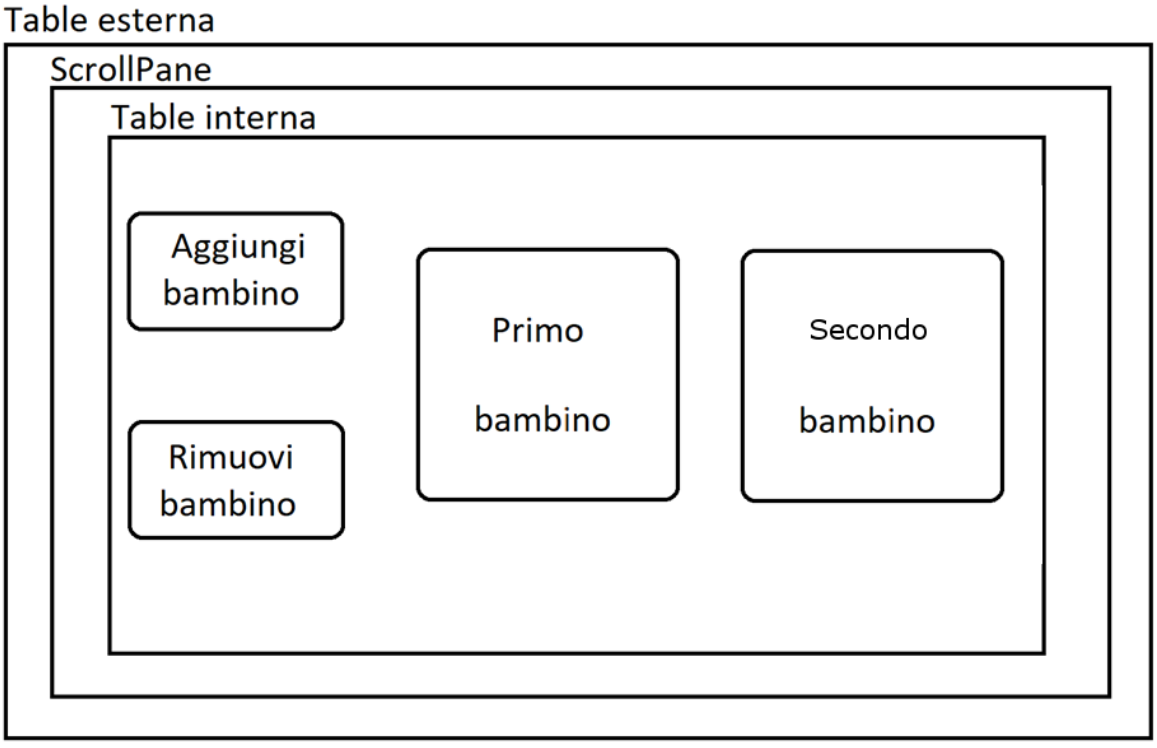


Fig. 4.2 - Schema esplicativo di Init (bottoni scorrevoli).

4.3 Main

La classe Main rappresenta una schermata scorrevole, in particolare uno sfondo animato che porta l'utente attraverso un mondo disegnato, dove alcuni disegni sono bottoni che possono essere premuti per scegliere quale mondo visitare.

Si tenga presente che il termine "background" è riferito al cielo, con tutti i suoi elementi, mentre il termine "foreground" indica il disegno dell'ambiente vero e proprio (le montagne, nel caso della grafica provvisoria utilizzata). Inoltre, si immagini che tutto ciò che si vede sullo schermo sia visto da una telecamera: per "camera" si intende punto di vista.

Fin tanto che l'utente scorre tenendo premuto il dito sullo schermo (o il bottone sinistro del mouse da desktop) il foreground darà l'impressione di essere trascinato da esso (in realtà è la posizione della camera che varia). Nel momento in cui l'utente smette di scorrere, staccando il dito, viene data una velocità di scorrimento alla camera sull'asse delle ascisse. Questa velocità sarà calcolata come spostamento diviso tempo, dove lo spostamento è dato dalla differenza tra le ultime due posizioni del cursore ed il tempo (in secondi) corrisponde agli attimi che sono trascorsi dall'ultimo aggiornamento; viene ottenuto grazie al metodo Gdx.graphics.getDeltaTime(). Oltre alla velocità sarà necessario implementare una forza di attrito che vi si opponga, in modo tale che questa velocità diminuisca nel tempo.

L'implementazione del movimento è stata effettuata seguendo di pari passo la fisica reale, dove si considera la velocità derivata prima (ds/dt) e accelerazione derivata seconda (ds/dt^2) dello spazio rispetto al tempo. Nel nostro caso dt sarà il tempo trascorso dall'ultimo aggiornamento, dato da getDeltaTime(); quindi, ogni volta che l'applicazione si aggiorna, si dovrà sottrarre la decelerazione moltiplicata per dt al modulo della velocità, poi sommare la velocità moltiplicata per dt alla posizione della camera.

```
//translate camera
cam.translate(velocity * dt, 0);

//translating button
button1.setX(button1.getX() - velocity * dt);
button2.setX(button2.getX() - velocity * dt);
button3.setX(button3.getX() - velocity * dt);

//decreasing velocity by a certain friction
if (velocity > 0) {
    velocity = velocity - deceleration * dt;
    if (velocity < 0)
        velocity = 0;
}
if (velocity < 0) {
    velocity = velocity + deceleration * dt;
    if (velocity > 0)
        velocity = 0;
}
```

4.3.1 Animazione del background

La posizione di ogni elemento del background è legata alla posizione della camera sull'asse delle ascisse. Il cielo è semplicemente un'immagine con un gradiente che va dal nero all'azzurro, larga quanto lo schermo, che parte dall'altezza zero sull'asse delle ordinate. Essa si muove verso il basso quando la camera si muove verso destra e verso l'alto quando quest'ultima procede verso sinistra. Sulla stessa immagine del cielo sono disegnate le stelle, mentre gli altri elementi sono contenuti in immagini separate perché hanno una velocità diversa (sole e luna) o devono poter passare davanti al sole (nuvole). Il sole e la luna si muovono più lentamente rispetto al foreground (sulle ascisse) ed al background (sulle ordinate), ciò viene ottenuto moltiplicando la posizione della camera per uno scalare.

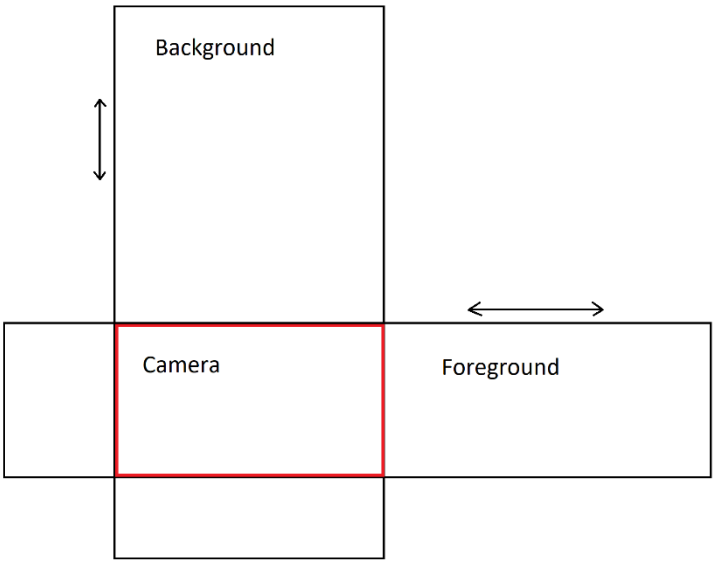


Fig. 4.3.1 - Schema di animazione del background.

4.4 Sub

Nella schermata dei dettagli, il tempo deve essere visualizzato in ore, minuti e secondi, perciò si arrotonderà il valore presente nel file testuale (variabile float che indica i secondi totali) con un casting per farlo diventare intero, poi si stamperà aggiungendo una "s" al termine del numero.

Se il valore sarà compreso tra 60 e 3600, prima della stampa si effettuerà la divisione intera per 60 dove il quoziente saranno i minuti (m) ed il resto i secondi.

Se il valore sarà maggiore di 3600 si dividerà per 3600, il quoziente saranno le ore (h) ed il resto sarà diviso per 60 e trattato come nel caso precedente.

Fig. 4.4 - Esempio di stampa del tempo trascorso.

4.5 Last

Per visualizzare le immagini (es. sfondo, animale) sullo schermo si utilizza il metodo draw() di SpriteBatch (classe di LibGDX). Utilizzando questo metodo è possibile sovrapporre le immagini, quindi per poter "far uscire" l'animale dalla stalla si è optato per "nasconderlo" dietro all'immagine di essa. Funziona esattamente come se i disegni fossero fatti su fogli di carta: si ha un foglio grande che funge da sfondo, su di esso vengono posti prima un foglio con il disegno dell'animale poi un altro con disegnata la porta della stalla; in questo modo, l'animale inizialmente non si vede ma può apparire se modifichiamo la sua posizione.
Nell'animazione di rinforzo, il coniglio ed il bambino che si muovono sono formati da una successione di frame. Per ottenere questo effetto è stata creata una classe ad hoc (Animation) il cui costruttore richiede una TextureRegion (classe di LibGDX, immagine che contiene tutti i frame affiancati), un numero intero (numero dei frame) e un numero float che indica, in secondi, quanto dovrà durare l'animazione. Dividendo la durata per il numero di frame si ottiene il "tempo di vita" di ogni singolo frame.

Fig. 4.5 - TextureRegion del coniglio che corre verso il bambino (grafica provvisoria).

Animation ha un metodo update() che deve essere chiamato ad ogni aggiornamento per permettere all'animazione di muoversi, un metodo stop() che accetta un parametro numerico per poter fermare l'animazione in un determinato istante ed un metodo getFrame() che ritorna al chiamante l'immagine del frame corrente (utilizzato per disegnarlo sullo schermo tramite la SpriteBatch nel metodo render() di Last).
Il metodo update(), se il tempo trascorso dall'ultimo cambio di frame è maggiore del tempo di vita, impone che il frame successivo diventi il frame corrente.
Per far sì che il movimento del coniglio sembri reale, dato che deve essere più veloce nel momento del salto e più lento quando tocca terra, si è optato per dargli una velocità sinusoidale in funzione del tempo. La funzione seno è stata opportunamente modificata cambiandone il periodo ed imponendovi una traslazione positiva sull'asse delle ordinate (necessaria altrimenti la velocità media sarebbe stata nulla). Periodo e traslazione sono parametri fissi e sono stati determinati in maniera sperimentale.
Per simulare lo scambio della carota tra il bambino ed il coniglio si utilizzano TextureRegion diverse, cambiandole in base alla posizione dei due oggetti.

4.6 Launcher

L'applicazione è dotata di diversi "launcher" (classi predeterminate dalla libreria), uno per ogni piattaforma sulla quale può essere compilata. Queste classi hanno un solo metodo main(), ed è da qui che viene chiamato il costruttore di SpeakUp.
In DesktopLauncher e HtmlLauncher è necessario definire le dimensioni della schermata; è possibile farlo riferendosi a SpeakUp.W e SpeakUp.H poiché questi parametri sono statici e vi si può avere accesso anche prima che la classe venga istanziata (il valore 0 corrisponde alla massima grandezza possibile sullo schermo in uso).

4.7 Stage

Tutti i dispositivi Android sono dotati di un tasto "BACK" che, se premuto, chiude l'applicazione in uso. Questo tasto può però essere gestito e risulta particolarmente comodo per tornare indietro nei menu; per farlo, è necessario modificare la classe Stage di LibGDX (la cui istanza controlla l'input dell'utente) in questo modo:

```
public class Stage extends com.badlogic.gdx.scenes.scene2d.Stage {

    public Stage(Viewport viewport) {
        super(viewport);

        //catch back button (otherwise it would close the application)
        Gdx.input.setCatchBackKey(true);
    }

    @Override
    public boolean keyDown(int keyCode) {
        return keyCode == Input.Keys.BACK;
    }
}
```

Fig. 4.7 - Codice della classe Stage utilizzata nell'applicazione.

5. Appendice

In questa sezione verranno raccolti in modo schematico i requisiti funzionali. Ognuno di essi sarà diviso in quattro parti:

- Descrizione: breve descrizione del processo.
- Input: informazioni in ingresso.
- Elaborazione: sequenza di azioni eseguita dall'utente.
- Output: risultato finale.

5.1 Inserimento di un bambino

Descrizione

Inserimento di un nuovo bambino.

Input

Nome del bambino.

Elaborazione

Quando l'utente preme il bottone "Aggiungi bambino" deve apparire una schermata contenente una casella di testo che permette di inserire un nome. Dopo aver scritto il nome e dato l'ok, la schermata sparirà e si farà ritorno al menu iniziale, che avrà un bottone in più con il nome del nuovo bambino inserito. Se al momento della conferma il nome è vuoto o è già stato utilizzato, dovrà apparire un messaggio di errore (diverso nei due casi) che blocca la procedura e richiede un nome diverso.

Output

Viene creata una nuova entry nel file testuale e viene aumentato il numero dei bambini. Viene creato un nuovo file dedicato al bambino inserito, destinato a contenere i suoi dati.

5.2 Rimozione di un bambino

Descrizione

Rimozione di un bambino già presente nel file e cancellazione dei dati relativi.

Input

Nome del bambino.

Elaborazione

Premendo il tasto “Rimuovi bambino” deve aprirsi una schermata con un menu a tendina che mostri tutti i bambini registrati. Scegliendo un nome e premendo il tasto “ok” apparirà una schermata di conferma che chiederà all’utente se vuole davvero eliminare quel bambino. Se la risposta sarà negativa si tornerà alla schermata precedente, altrimenti si farà ritorno al menu iniziale, che dovrà essere aggiornato e quindi non contenere il bottone relativo al bambino che è appena stato rimosso.

Output

Viene rimosso il nome dal file testuale e decrementato il numero di bambini.
Il file personale del bambino rimosso viene eliminato.

5.3 Scelta del bambino

Descrizione

Scelta del bambino con cui ci si vuole esercitare.

Input

Nome del bambino.

Elaborazione

L’utente, che ha appena aperto l’applicazione, trova davanti a sé una serie di bottoni con i nomi dei bambini inseriti nel file; ne sceglie uno.

Output

Chiamata del costruttore di Main e conseguente cambio di stato.

5.4 Chiusura dell’applicazione

Descrizione

Chiusura dell’applicazione.

Input

Nessuno.

Elaborazione

Nel menu iniziale deve essere presente un tasto “Esci” che, se premuto, chiede all’utente la conferma per uscire. Se viene annullata l’operazione si tornerà al menu iniziale; se invece viene confermata, l’applicazione si chiuderà.

Output

De-allocazione della memoria e chiusura dell’applicazione.

5.5 Impostazioni

Descrizione

Modifica delle impostazioni.

Input

I tre valori dei tre livelli di volume (suoni, musica e voce).

Elaborazione

Premendo il pulsante a forma di rotella da qualsiasi stato del gioco (eccetto il menu iniziale) l’utente accede alla schermata di impostazioni, che appare sul lato sinistro occupando soltanto una parte dello schermo. Vi saranno al suo interno tre slider con valori percentuali, utilizzando i quali l’utente potrà modificare il volume.

Output

Inserimento dei valori aggiornati nel file testuale.

5.6 Scelta del mondo

Descrizione

L’utente sceglie il mondo nel quale si eserciterà.

Input

Stringa che identifica il mondo.

Elaborazione

Scorrendo nella schermata di Main, l'utente potrà visualizzare tutti i bottoni relativi ai mondi esplorabili e sceglierne uno. Una volta premuto il bottone si dovrà cambiare schermata entrando nel mondo selezionato.

Output

Chiamata del costruttore di Sub e conseguente cambio di stato.

5.7 Dettagli

Descrizione

Visualizzazione dei dettagli del lavoro svolto fino ad ora nel mondo corrente.

Input

Stringa corrispondente al mondo in cui ci si trova e nome del bambino.

Elaborazione

Premendo il pulsante dei dettagli, verrà visualizzato il tempo totale trascorso nelle sezioni del mondo ed il numero di animazioni che sono state fatte partire. Se l'utente preme nuovamente lo stesso bottone si uscirà da questa schermata.

Output

Nessuno.

5.8 Scelta della sezione

Descrizione

Scelta della sezione di mondo in cui ci si trova.

Input

Numero che identifica la sezione.

Elaborazione

L'utente sceglie uno dei bottoni che rappresentano le sezioni del mondo. Una volta premuto si dovrà cambiare schermata, entrando nella sezione selezionata.

Output

Chiamata del costruttore di Last e conseguente cambio di stato.

5.9 Esercitazione

Descrizione

Esercitazione del bambino.

Input

Nessuno.

Elaborazione

Quando l'utente tocca la parte all'estrema destra dello schermo deve partire un'animazione che prèmi il bambino per aver provato a pronunciare la parola. Questo avverrà per le prime due volte; la terza volta invece partirà un'animazione di rinforzo, più duratura ed accattivante. Al termine di quest'ultima animazione si tornerà indietro allo stato precedente. Dovrà essere calcolato il tempo trascorso ed il numero di animazioni richieste.

Output

Chiamata del costruttore di Sub e conseguente cambio di stato.
Scrittura dei dati aggiornati (tempo e numero tocchi) sul file testuale del bambino.

6. Conclusione

Grazie a questa esperienza ho assunto più consapevolezza sul significato di progettazione del software e ho potuto applicare molti degli insegnamenti e dei consigli raccolti in tre anni di studio all'Università. Penso che progettare e realizzare la struttura basilare di un'applicazione che dovrà essere terminata da altri in futuro sia un compito piuttosto delicato; è necessario scrivere il codice in maniera che sia facilmente comprensibile da altre persone, che probabilmente ragionano in modo diverso; inoltre è importante curare l'aspetto progettuale, pensando per esempio alle possibili modifiche che potranno essere richieste in futuro. In questo modo sarà possibile risparmiare tempo e risorse quando verrà il momento di effettuarle.

Andare in prima persona a parlare con il cliente e discutere dei requisiti è stata un'esperienza formativa anche da un punto di vista lavorativo; mi ha aiutato a capire cosa potrebbe significare lavorare in questo ambiente in futuro, mettendomi faccia a faccia con il problema senza poter contare su aiuti esterni. Penso che la parte più ardua sia stata proprio il dialogo: capire esattamente le specifiche richieste e far capire i limiti e le convenienze tecniche non è sempre scontato, in particolare se il cliente ha una conoscenza limitata in ambito informatico.

Infine, svolgere questo lavoro mi ha permesso di approfondire la conoscenza del linguaggio Java, in particolare applicato al framework LibGDX che considero un ottimo strumento di programmazione, sia per la portabilità che per la documentazione facilmente reperibile online.

Bibliografia

La parte introduttiva di questo elaborato fa riferimento al seguente articolo non ancora pubblicato:

Benassi E. (2017). Realizzazione di una App per il recupero delle abilità linguistiche di bambini late producers e sperimentazione della sua efficacia.