Laboratorio di Applicazioni Mobili

Applicazione in ambiente iOS

Giovanni Fazi
giovanni.fazi@studio.unibo.it

Alma Mater Studiorum - Università degli Studi di Bologna

September 1, 2018

Abstract

Lo scopo di questo progetto è la realizzazione del famoso rompicapo "Spaccaquindici" in ambiente iOS utilizzando il linguaggio Swift e l' ambiente di sviluppo Xcode. Il progetto base prevede che l' utente scelga una immagine che verrà poi ripartita in 16 caselle (di cui 1 vuota) secondo lo schema di una griglia 4x4, le caselle vengono poi disordinate secondo mosse casuali. Lo scopo del gioco è quello di riportare le caselle nella posizione originale. Il tempo trascorso per il riordinamento delle caselle viene misurato da un timer che si attiva a seguito della prima mossa dell' utente. Il progetto base è stato poi esteso aggiungendo funzionalità che migliorano l' esperienza di gioco.

Contents

1 Scopo dell' Applicazione

Il gioco del quindici o spaccaquindici è un rompicapo classico creato nel 1874 da Noyes Palmer Chapman. Il gioco consiste di una tabellina di forma quadrata, solitamente di plastica, divisa in quattro righe e quattro colonne (quindi 16 posizioni), su cui sono posizionate 15 tessere quadrate, numerate progressivamente a partire da 1. Le tessere possono scorrere in orizzontale o verticale, ma il loro spostamento è ovviamente limitato dall'esistenza di un singolo spazio vuoto. Lo scopo del gioco è riordinare le tessere dopo averle "mescolate" in modo casuale (la posizione da raggiungere è quella con il numero 1 in alto a sinistra e gli altri numeri a seguire da sinistra a destra e dall'alto in basso, fino al 15 seguito dalla casella vuota).
Lo scopo della applicazione è quello di realizzare una applicazione iOS che implementa il gioco del quindici. Diversamente dalla versione classica, che prevede l' uso di tessere numerate, i requisiti del progetto richiedono l' uso di una immagine ripartita in tessere. Inoltre deve anche essere presente un timer che si attiva a seguito della prima mossa effettuata dal giocatore e che tiene traccia del tempo trascorso per risolvere il rompicapo, fermandosi una volta che esso viene risolto.

2 Requisiti e Funzionalità previste

2.1 Progetto Base

- L' applicazione deve mostrare una finestra di scelta di almeno tre immagini.
- Dopo aver selezionato una immagine, una seconda vista deve mostrare la immagine divisa in NxN caselle disordinate, con N maggiore o uguale a 4.
- Il giocatore puó spostare le caselle nella posizione vuota toccando quelle adiacenti ad essa
- Deve essere presente un timer che misura il tempo trascorso per il riordinamento
- Per disordinare le caselle bisogna effettuare una serie di mosse casuali
- Deve essere presente almeno un Navigation Controller

2.2 Progetto Esteso

- Il giocatore deve poter scegliere il layout della griglia di gioco, le dimensioni possibili devono essere 4x4, 5x5, 6x6.
- L' applicazione deve prevedere un database di utenti dove, per ogni giocatore, vengono registrati i valori di: Nome, Punteggio, Obiettivi raggiunti, Miglior tempo di partita per ognuno dei tre layout possibili, Minor numero

di mosse utilizzate per riordinare un puzzle per ognuno dei tre layout possibili.

- Definire un elenco di sfide che ogni utente può completare. Devono essere previsti più livelli di difficoltà
- Creare una vista che mostri una top 5 di utenti con punteggio più alto, gli utenti con i tempi migliori di risoluzione di un puzzle per ognuno dei tre layout possibili, gli utenti con il numero di mosse minore impiegate per la risoluzione di un puzzle per ognuno dei tre layout possibili.
- Creare una vista che elenchi tutti i giocatori registrati nel database e che permetta, toccando il nome di un giocatore, di visualizzare in una nuova vista gli obiettivi raggiunti e quelli da raggiungere. In queste viste deve essere possibile rimuovere dal database ogni giocatore registrato o poter rimuovere un singolo giocatore una volta selezionato.
- L' utente deve essere in grado di poter aggiungere immagini con cui giocare. Le immagini possono essere prese dalla galleria o possono essere acquisite da fotocamera.
- Durante una partita deve essere possibile visualizzare la immagine originale, stampandola a schermo
- Durante una partita deve essere possibile mostrare il numero di ogni tessera in modo che si capisca l' ordine attuale delle tessere.
- Al completamento di un puzzle deve comparire un popup che mostra informazioni sulla partita giocata e che permette di inserire lo username del giocatore per aggiornarne i valori nel database o, nel caso il giocatore non sia registrato, inserirlo nel database.
- Alla fine di ogni partita l' utente deve essere in grado di condividere i propri risultati tramite social, sms, email... utilizzando un ActivityViewController.

3 Progettazione e Scelte Implementative

L' applicazione è stata progettata per iOS 11 tramite Xcode 9.3.1 (9E501). È stata testata su iPhone X tramite simulatore.
Il design pattern utilizzato è MVC (Model View Controller). Ogni View è gestita da una classe Controller. La progettazione è iniziata creando il Model, ovvero la logica del gioco.

3.1 Model: Board e Tile

La struct Tile rappresenta i valori di una tessera di gioco:
Ogni tessera è composta da un identificativo univoco e le coordinate x,y che indicano la posizione sulla Board di gioco. Per generare gli identificativi è stata usata una variabile statica incrementale. Per le coordinate è stata creata una struct "TilePosition" che contiene due variabili x ed y, più due extension per la uguaglianza o disuguaglianza fra due "TilePosition".

La struct Board è l' equivalente della "tabellina di plastica" usata per giocare, essa infatti contiene un array di Tiles, le coordinate della tessera vuota e la lunghezza di un 'lato' della griglia. Infatti per calcolare il numero di tessere totali in un layout è sufficiente conoscere quante ne sono presenti in una riga o colonna. Inoltre è anche presente un array di Tiles disposti in ordine risolto, questo array viene utilizzato per controllare se il puzzle è stato risolto.
I metodi principali della struct Board sono:

- Init: Viene passata la dimensione del lato della griglia e vengono create tutte le tessere in ordine
- isAdjacentToEmpty e moveTile: per muovere un tile nella posizione vuota viene effettuato un semplice swap di posizioni fra coordinate del tile e posizione vuota, ovviamente non tutti i tile si possono muovere e quindi si effettua un controllo per stabilire se un tile è "movable". Il controllo viene fatto confrontando il tile richiesto con tutti i tile adiacenti in verticale o orizzontale rispetto alla posizione vuota. Se qunidi il tile che si vuole spostare supera il controllo viene effettuato lo swap di coordinate.

3.2 Model: Achievements

Sono state definite 24 sfide che un giocatore può affrontare. Nel database le sfide sono memorizzate come una stringa contenente solo "0" ed "1". La stringa ha lunghezza 24 ed il carattere n-esimo corrisponde alla n-esima sfida la cui descrizione è salvata in un array con tutte le descrizioni.
I metodi principali sono due:

- getPoints: prende in input i valori di una partita (numero di mosse, tempo, dimensione griglia) e calcola il punteggio secondo la formula: 50 * numero sfide che sono state superate nella partita
- getAchievements: prende in input i valori di una partita (numero di mosse, tempo, dimensione griglia) e costruisce una stringa di lunghezza 24 dove il carattere n-esimo è "1" se è stata superata la sfida n-esima, "0" altrimenti.

3.3 Controller: Menu

Prima view del navigation controller. Contiene tre UIButtons con tre segues: Uno mostra le opzioni di partita, uno mostra la leaderboard, l' ultimo mostra gli utenti registrati e le varie sfide superate

3.4 Controller: LeaderBoard e Achievements

Sono state usate delle semplici TableView per rappresentare i dati dei giocatori.

Nella leaderboard si trovano varie sezioni contenenti i record per categoria. I dati rappresentati vengono acquisiti effettuando un filtraggio sul database tramite NSPredicate.

Gli achievements vengono rappresentati in due tableViews: La prima più generale mostra tutti gli utenti registrati. Anche qui i dati vengono acquisiti dal database tramite Core Data.
In questa view è presente uno UIButton che permette di cancellare tutti gli utenti nel database (previo messaggio di warning). Nel caso in cui si scelga di cancellare l' intero database, si effettua una pop nello stack del navigation controller e si ritorna quindi al menu iniziale; questa scelta è stata fatta per evitare di riaggiornare la table view con il nuovo contenuto del database; inoltre, visto che il database è vuoto, non c'è motivo di mostrare una tableView vuota all' utente.
Toccando un elemento nella prima tableView viene mostrata una seconda tableView più specifica che mostra le varie sfide superate da un utente. Le descrizioni delle sfide sono precedute dal prefisso "Done:" se la sfida è stata superata, "To do:" altrimenti. Inoltre le sfide superate hanno il testo colorato di verde, quelle To do sono colorate di rosso.
In questa View è presente un button che permette di cancellare l' utente dal database (previo messaggio di warning).

3.5 Controller: GameSetup

Questa vista serve per selezionare i parametri di gioco. L' utente può scegliere una immagine tramite una pickerView. Questa pickerView è stata ruotata orizzontalmente ed inoltre presenta un effetto di scrolling infinito, tutto ciò serve per migiorare la navigazione nel menù rendendo tutto più intuitivo e comodo per l' utente.
La scelta della griglia di gioco avviene tramite un SegmentedControl dove le scelte contengono una breve descrizione (ad esempio la scelta 4x4 è descritta come "classic", mentre la 6x6 è descritta con "Hard" per indicare la difficoltà).
In alto a destra sulla barra di navigazione è presente un pulsante per aggiungere immagini alla pickerView. Per comodità è stata usata la libreria YPImagePicker per la gestione dell' acquisizione di immagini da galleria o fotocamera. Questa libreria permette inoltre di effettuare crop delle immagini scelte, funzionalità importante dato che sono ammesse immagini di dimensioni 480x480. La scelta delle dimensioni fisse per le immagini non è limitante, la libreria permette di ridimensionare adeguatamente, inoltre si evita di dover fare resize forzati di immagini rettangolari che portano a spiacevoli effetti grafici dovuti al nuovo aspect ratio non ottimale.
In fondo alla view si trova il buttonPlay che permette di passare alla view di gioco. I metodi principali di questa classe sono:

- AddImage: chiama la libreria YPImagePicker impostandone la configurazione. Se l' utente aggiunge con successo una immagine, essa viene salvata nella galleria in un album apposito chiamato "Spaccaquindici". La libreria restituisce quindi una immagine che viene aggiunta alla PickerVIew; per migliorare la user experience ogni immagine appena aggiunta viene selezionata automaticamente dalla PickerView in modo che l' utente non debba cercarla.

- pickerView: come già accennato la pickerView è stata routata di 90 gradi per creare un effetto di scroll più naturale e comodo. Questa funzione di inizializzazione della PickerView stabilisce che gli elementi da presentare siano UIImage (anch' essi ruotati come la pickerView).

- fetchAlbumPhotos: Le immagini custom aggiunte si trovano nell' album della galleria chiamato "Spaccaquindici", questa funzione usa PHAsset e predicati NSPredicate per caricare le immagini dentro un array da cui la pickerView attinge le UIImage da mostrare.

- viewDidLoad: chiama tutte le funzioni precedenti in modo da coordinare adeguatamente la pickerView e il fetching di immagini dalla galleria

- prepare: effettua un segue passando i valori di dimensione griglia ed immagine scelta alla view di gioco successiva.

3.6 Controller: GameView

Questo controller comunica con il model Board e la View di gioco. I tasselli di gioco sono dei buttons a cui viene assegnata una porzione della immagine scelta precedentemente.

- viewDidLoad: Per prima cosa si calcolano le dimensioni della Board di gioco, poi si generano tutti i buttons in ordine, assegnando ad ognuno la porzione di immagine corretta ed effettuando lo smussamento dei bordi per imitare la forma dei tasselli reali. Dopo aver assegnato le target action ai vari buttons, si effettuano 130 mosse casuali per mischiare le tessere di gioco.
  Lo scrambling viene effettuato scegliendo a caso uno dei tile "movable" che sono adiacenti al tile vuoto; per aumentare l' entropia delle mosse random si è fatto in modo che a seguito di una mossa casuale non sia possibile effettuare la mossa "inversa", ovvero dallo stato che si ottiene a seguito della mossa non è possibile ritornare direttamente allo stato precedente. Questa politica evita che si crei una sorta di backtracking sprecando mosse casuali.

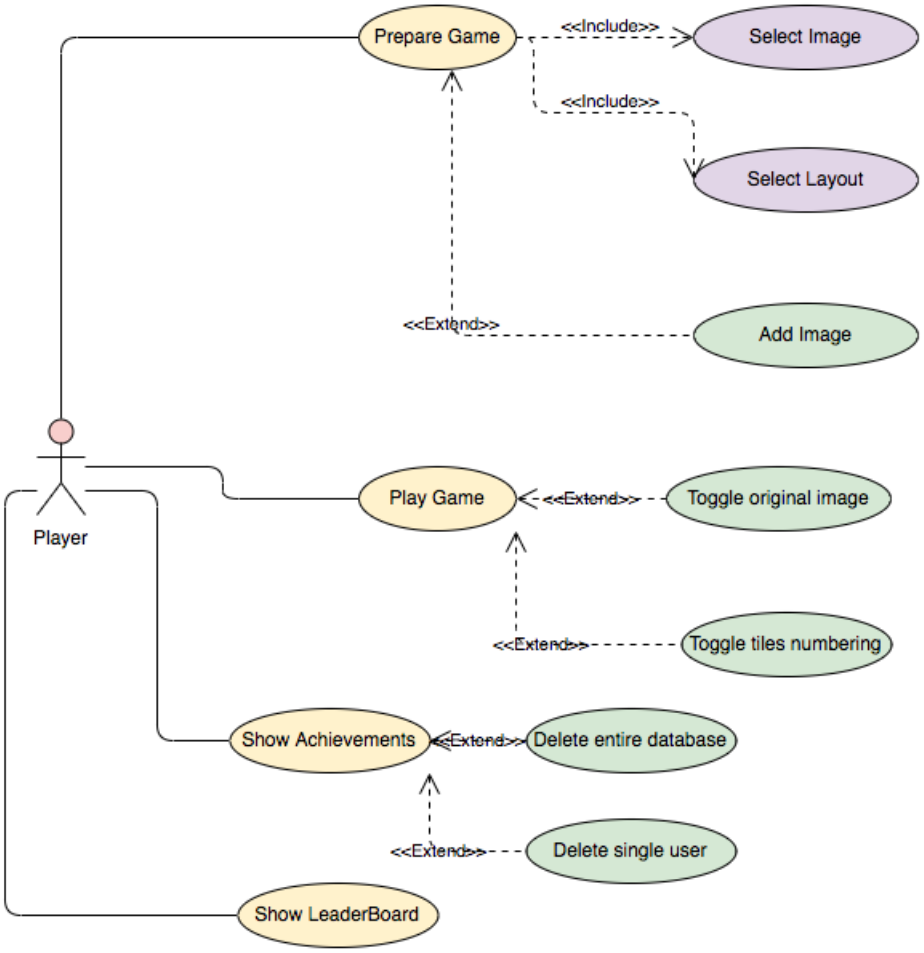
- buttonDidTouch: muove il button scelto nella view aggiornando di conseguenza il Model. Inoltre è responsabile per la attivazione del timer e del conteggio delle mosse (entrambi memorizzati in variabili e presentati a video tramite labels). Infine ha il compito di controllare alla fine di ogni mossa se il puzzle è risolto, in tale caso ferma il timer ed elimina tutte le target action dei buttons (in modo che non si possano muovere) e poi mostra a video il pop up finale.

- makePopUpAppear: compare tramite una animazione ed attiva un effetto blur nel background rendendo il popup maggiormente visibile. Nel popup è possibile vedere i dati della partita ed inserire un nome utente per registrarsi sul database.

- closePopup: una volta che si preme il pulsante "done" sul popup se è stato inserito un nome utente si effettua la registrazione nel database o l' update dei valori della partita nella tabella del giocatore. Un nuovo giocatore ha sempre tutti i campi popolati con valori di default, questo per evitare di usare valori nulli nel database.

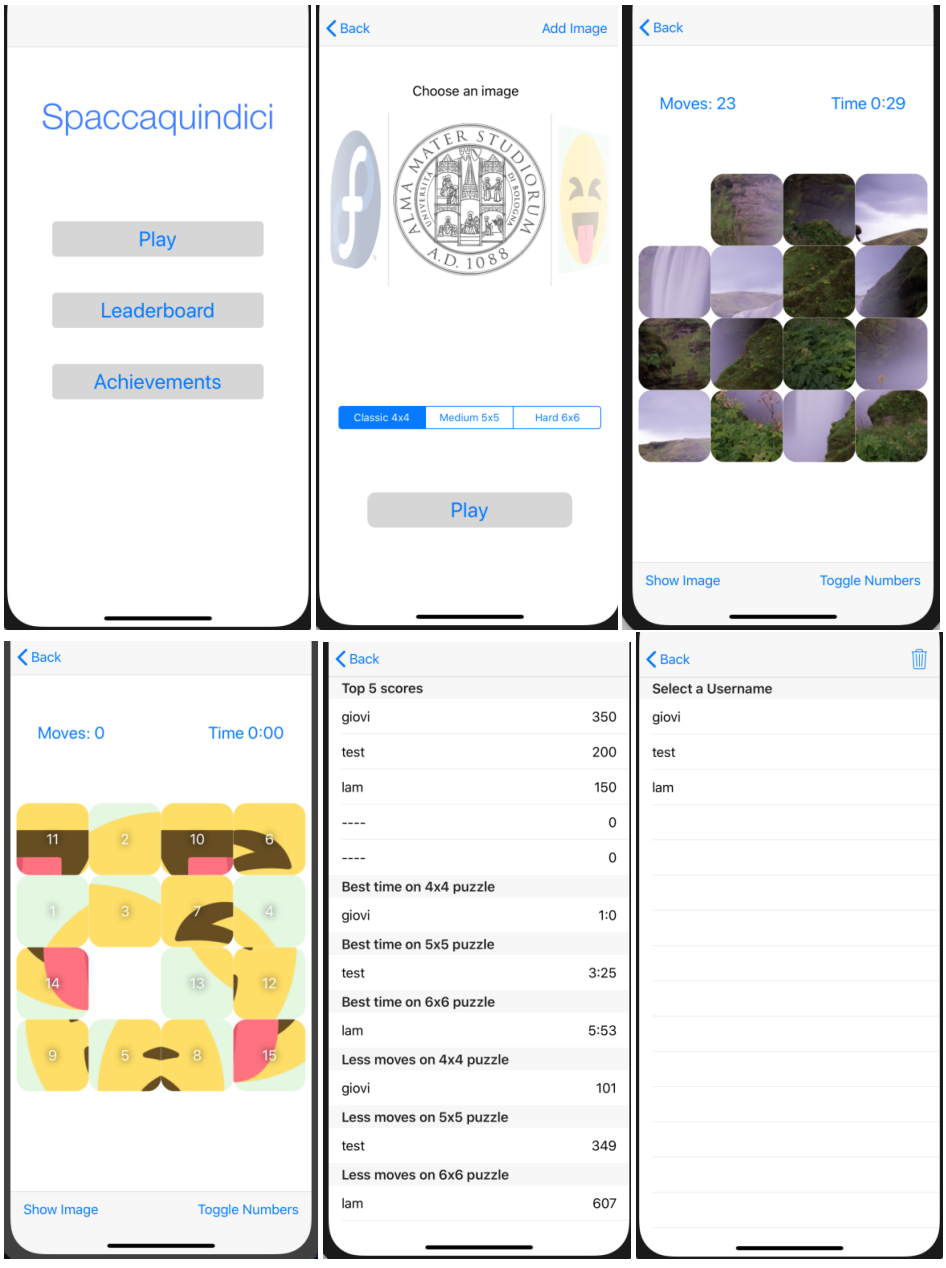
- ToggleNumbers e ShowImage: sono action di due buttons che permettono rispettivamente di mostrare all' utente gli id dei buttons e per mostrare la immagine originale. Entrambe le funzionalità servono per aiutare a risolvere un puzzle difficile, in particolare nel puzzle 6x6 capita spesso che due tiles siano indistinguibili fra loro (si pensi ad esempio ad una immagine monocolore) e quindi queste funzionalità risultano particolarmente utili.

4 Casi d'uso


Prepare Game
<<Include>>
Select Image
<<Include>>
Select Layout
<<Extend>>
Add Image
Player
Play Game
<<Extend>>
Toggle original image
<<Extend>>
Toggle tiles numbering
Show Achievements
<<Extend>>
Delete entire database
<<Extend>>
Delete single user
Show LeaderBoard

5 View e design

Per la parte grafica è stato usato un design minimale con colori bianco e grigio.



6 Idee per estensioni

- Aggiungere layout non quadrati (esempio 3x5)
- Aggiungere nuove sfide per i nuovi layout
- creare una leaderboard online in modo che gli utenti possano competere via internet
- aggiungere una modalità player vs player che effettua pairing di due (o piu) giocatori, poi mostra lo stesso puzzle. I giocatori gareggiano a risolvere il puzzle per primi
- aggiungere sezione tutorial che spiega le regole per i principianti