PLAStiblob

Bernardi Lorenzo  
Bortolamiol Alessandro  
Dal Colle Gabriele  
Scognamiglio Simone

# Introduzione

Per questo progetto ci siamo affidati ad un’idea che avevamo avuto durante l’anno: quella di riuscire a coniugare un gioco dall’aspetto animato e divertente con un tema attuale e molto importante come la natura. Plastiblob è anche questo: è un progetto di cui andiamo molto fieri, soprattutto perché perché contiene molta passione.   
Speriamo che possa piacerle almeno un quarto di quanto piace a noi e speriamo riesca ad arrivare al livello finale, per gustare una piccola sorpresa.

Il progetto è stato creato e modificato online grazie alla piattaforma GitHub, ognuno di noi infatti poteva eseguire la sua modifica al progetto e renderla disponibile a tutti gli altri membri del gruppo. Tale scelta è stata presa per aver una maggiore flessibilità in termini di modifica del progetto ma anche di tempo.

Tutto il progetto è stato pensato per uno schermo **1280 x 720**, infatti le immagini di background sono di questa misura. All’interno del file di config del progetto, abbiamo però inserito tale riga:

scale = "zoomEven",

che andrà a ridimensionare gli elementi nello schermo in base alla grandezza del display del cellulare di chi sta utilizzando il gioco.

# Files del progetto

## menu.lua

Menu è la prima pagina su cui l’utente approda subito dopo aver cliccato sull’avvia applicazione. Questa semplice pagina presenta:

* Il logo del nostro gioco
* Un bottone collegato ad un listener per il suo relativo click
* Il logo dell’università di Udine

Notiamo subito che il bottone è rappresentato da un animazione sprite. All’interno di questo file abbiamo anche caricato una musichetta di sottofondo che rappresenta il tema portante del gioco.

## menu-levels

Questa pagina è un po’ più complicata della precedente, infatti contiene 4 immagini che permettono all’utente di scegliere il livello in cui andare. Per memorizzare i livelli sbloccati dall’utente abbiamo usato la libreria di corona che permette di interfacciarsi con sqlite3. Tramite questa implementazione siamo stati capaci di controllare e memorizzare dati utili a noi come il numero di liveli completati dall’utente e i punteggi ottenuti in ogni singolo livello.  
Se l’utente si interfaccia per la prima volta con l’applicazione, creeremo questa semplice tabella all’interno di un db ‘data’

local tableSetup = [[CREATE TABLE levels ( ID INTEGER PRIMARY KEY autoincrement, level, scoreLevel1, scoreLevel2, scoreLevel3, scoreLevel4);]]

la tabelle andrà a contenere un campo primario ID che conterrà un numero univoco per user. Ovviamente questo è superfluo nel nostro caso, infatti il db non si interfaccia con utenza online ma rimane in locale e quindi ci sarà un’unica riga all’interno di questa tabella. Oltre al campo di ID, abbiamo ‘level’ che conterrà un numero intero in base al livello che si deve affrontare: per esempio se è la prima volta che l’utente entrà nel gioco sarà 1, se ho correttamente passato il livello 1 aumenterà a 2. Gli altri campi possiedono al loro interno i vari punteggi ottenuti nei livelli, tali punteggi aumentano in base agli oggetti di plastica raccolti durante il percorso del blob.  
Dato che l’obiettivo del gioco è quello di **raccogliere più plastica possibile,** se l’utente arriverà a raccogliere il numero massimo di oggetti di plastica verrà premiato con 3 stelle, sennò a scendere con 2 stelle o 1 stella.

Il funzionamento di questo file lua non è immediato: abbiamo infatti 4 immagini distinte da cliccare per andare in ogni singolo livello: una per il livello 1, una per il livello 2, una per il livello 3 e una per il 4.   
L’utente, però, deve aver superato il livello precedente per passare al successivo e quindi bisogna rendere non cliccabile la foto del numero del livello che non è disponibile allo user. Per fare ciò abbiamo controllato il campo ‘level’ all’interno del db, e abbiamo reso disponibile al touch solamente i livelli <= al campo del database.   
Per esempio: nel caso il campo del db sia 2 🡪 è possibile andare a giocare all’interno dei livelli 1 e 2.

I livelli non disponibili andranno marcati usando una foto diversa con il lucchetto

## Storylevel

Ogni livello ha bisogno di una piccola premessa, tramite un fumetto infatti andremo a raccontare l'avventura the last ones appunto questo file permette di visualizzare le varie pagine del fumetto andando a girare pagina tramite un click. All'interno di questo file è presente un funzionale meccanismo di scorrimento immagini, ad ogni click si andrà ad aumentare una variabile denominata n e se andrà a richiamare la funzione che visualizzerà l'immagine nel percorso creato in questo modo:

   imgpath = "immagini/livello-"..leveltarget.."/storia/"..n..".png" --creo il percorso dell'immagine

Come possiamo notare abbiamo usato due variabili, una e leva il target e contiene il livello di cui stiamo leggendo la storia appunto leva il target viene valorizzata appena si entra nella scena grazie alla tabella pa ramses. Tale tabella di pa ramses viene passata dal file chiamante nel nostro caso da me New levels che andrà a indicare quante immagini si andrà a mostrare all'utente durante la lettura del fumetto. Ovviamente per rendere possibile ciò abbiamo dovuto rinominare ogni singola immagine del fumetto con un numero progressivo, componendo quindi un percorso di questo tipo: “immagini/livello-1/storia/1.png”.

per esempio con la stringa indicata appena qui sopra andremo a leggere la prima pagina del fumetto riguardante il primo livello del nostro gioco .

## tutorial

questo file waw permette la visualizzazione di un semplice straight con grandezza 1280 x 720, grazie all animazione di tale spray tine si andranno a vedere le semplici istruzioni necessarie per riuscire a giocare pompa. Grazie ad un semplice clic dello schermo riusciremo ad andare al livello. Il numero del livello in cui andremo a puntare lo prendiamo impasto grazie alla solita tabella dei parametri che viene passata dal chiamante virgola in questo caso ci chiamerà la schermata story level che a sua volta veniva chiamata dal menù dei livelli punto si andrà quindi a delineare la perfetta gerarchia della chiamata di un livello vesc

cliccando sulla foto del primo livello si andrà al fumetto di quest'ultimo, Passando lì il parametro uno (numero del livello a cui andare). successivamente dopo avere visionato il fumetto si andrà al tutorial del livello da cui siamo partiti nel nostro caso il livello uno. Dope avere visionato anche il tutorial costruiremo un percorso di questo tipo:

local levelTo = "levels.level"..leveltarget --punto al livello in base al paramettro passato dalla scena precedente

             composer.gotoScene(levelTo, "fade", 500)

Come possiamo prenotare allinterno della variabile e leva il target e possibile trovare il numero del livello a cui andare.

## level1

Il primo livello è quello più semplice di tutti. Al suo interno avviene uno scrolling del background infinito e ogni tot secondi vengono richiamati dei nemici grazie a funzioni simili a questa qui sotto

gameLoop = timer.performWithDelay( time\_speed\_min, loop, 0 ) --loop del gioco in cui fa muovere gli sprite

Questi timer infatti andranno a creare e mettere in loop il movimento dei nemici che piano piano scrolleranno verso la parte sinistra dello schermo.

Allinterno della scena vengono definiti tra i gruppi: un gruppo per il background di sfondo, un gruppo per gli elementi che scrollano nello sfondo e un ultimo che contiene i bottoni per tornare alla home. La gerarchia che abbiamo pensato di utilizzare vede come gruppo di fondo il background, sopra di esso tutti gli elementi come ISP right, e sopra di quest altro gruppo ancora si trova il gruppo dei bottoni.

La durata del livello è decisa a priori grazie ad una variabile “TimeToPlay”, tale variabile viene valorizzata all’entrata della scena.  
All’interno di quest’ultima abbiamo inserito una funzione per il conteggio dei secondi che andrà a comparare ‘timePlayed’ (variabile che ogni secondo si incrementa di 1) con ‘TimeToPlay’: se tali risultano uguali allora l’utente avrà passato il livello e il suo punteggio verrà salvato all’interno del database e il livello 2 verrà reso disponible ad essere giocato.

All'inizio del file abbiamo dichiarato un numero massimo di elementi di plastica da raccogliere, tali elementi verranno fatti comparire nello schermo attraverso la funzione qui sotto:

callingPlasticbag = timer.performWithDelay( (timeToPlay/plasticToCatch)\*1000, plasticbagLoop, plasticToCatch) --chiama gli oggetti di plastica ogni 10 secondi

Vediamo come la variabile time to play cambia i vari settaggi di tempo del nostro gioco rendendolo più lento o più veloce in base alla nostra scelta. Un valore che ci sembra più consono all utilizzo dell utente durante l'esperienza di gioco e un valore tra i 60 secondi e il 70.

Durante il gioco, la velocità di scorrimento dei nemici andrà ad aumentarsi fino ad arrivare ad un determinato massimo imposto da noi. La bravura dell’utente sarà quella di riuscire a saltare ogni singolo nemico e allo stesso tempo riuscire a raccogliere quanta più plastica possibile in modo da ripulire il mondo da una situazione a tratti post-catastrofica.  
Ogni personaggio all’interno del gioco possiede un corpo fisico, dotato di un suo peso, densità, trasparenza. Ogni corpo fisico contiene al suo interno anche un id, grazie a queste idee in fase di collisione riusciremo a capire chi sia scontrato con chi. Durante il nostro gioco, infatti, può capitare di scontrarsi con un elemento piuttosto che un altro: c'è il caso in cui lo spray tine si possa collidere con il sacchetto di plastica virgola in questo caso andrà aumentato il ponteggio e il gioco continuerà in maniera normalissima. Nel caso in cui il nostro personaggio si dovesse scontrare con un nemico si andrà a richiamare una funzione game over che richiamerà un'altra schermata per permettere all'utente di ricominciare il livello oppure di tornare al menù della selezione dei livelli.

## Gameover

Visto che sopra abbiamo parlato della fase di game over , andiamo a parlare del file che ci permette di ricominciare il livello oppure di andare alla home per sceglierne un altro punto esiste un unico file di game over, il chiamante della schermata del game over andrà a passare a quest'ultima una tabella dei parametri contenente:

* la transazione con questo andrà alla schermata di game over
* i millisecondi della transazione
* il nome di quale livello ha richiamato la schermata di game over

quest'ultimo è molto importante nel funzionamento della schermata di game over, infatti quando cliccheremo nel bottone di ricomincia andremo a consultare quest'ultimo parametro per creare il percorso del file adatto a tornare Nel file da cui siamo partiti e abbiamo richiamato il game over Ehi. Oltre a questo caso, si può anche richiamare la schermata dei menu livelli tramite un semplice bottone composto da una casetta.

## level2

Il livello due se andrà a sbloccare solamente dopo il completamento del primo livello, il funzionamento è simile a quello del primo livello ma con l'aggiunta di un secondo nemico: una pozza di acido. L'obiettivo del livello è sempre quello di raccogliere più plastica possibile riuscendo ad evitare tutti i tuoi tipi di nemici presenti nello schermo. Un'ulteriore aggiunta di questo livello e la possibilità dello spray te di sparare al nemico in modo da toglierlo completamente dal display e riuscire ad andare avanti con maggiore facilità. Le munizioni del personaggio vanno ad aumentare in base al numero di oggetti di plastica raccolti durante il suo percorso, se dovesse utilizzare una munizione il punteggio andrebbe adesso è diminuito in quanto andrebbe a rendere più facile l'esperienza del livello. Lo sparo e uno spray te uguale a tutti gli altri presenti nel livello , con l'unica differenza che lo scroll avviene verso destra. Per permettere all'utente di sparare abbiamo diviso lo schermo del cellulare in due parti: una prima metà di sinistra e una seconda metà di destra. tramite un ascoltatore di daccia riusciamo a capire da dove proviene il tocco del dito e sappiamo se richiamare la funzione del salto oppure la funzione della creazione di un proiettile .

Anche in questo livello i nemici vengono chiamati tramite un timer. Vediamo un esempio qui sotto:

 callingEnemies[1] = timer.performWithDelay (1500, enemiesLoop, 1 )

Per comodità abbiamo inserito tutti i timer per la chiamata dei nemici allinterno di apposite tabelle, ogni indice contiene al suo interno un nuovo timer che viene richiamato ogni tot secondi. Nel momento in cui se dovesse uscire dalla scena, tutte queste tabelle andranno ad essere il suo Tate tramite una funzione di reset, che in base al numero di elementi allinterno della tabella ne eliminerà i timer e ascoltatori. Anche in questo file uos è presente il salvataggio dei dati attraverso il database che avviene in maniera totalmente uguale a quella del primo livello, solamente con la differenza che si andrà ad aumentare il punteggio acquisito nel livello due .

## level3

in questo terzo livello vengono aggiunte anche le piattaforme che permettono allo spray tine di saltarci sopra e riuscire a raccogliere più oggetti di plastica. il funzionamento del livello è pressoché uguale a quello dei precedenti: gli amici che ci si sposta coleranno durante il percorso sono sempre due ma verranno richiamati più spesso prendendo livello più complicato all'utente.

## level4

Il quarto livello e l'ultimo presente allinterno del gioco. Abbiamo composto il quarto livello prestando una minuziosa attenzione al posizionamento di ogni singolo nemico in modo da rendere difficile la passare. È necessario più di un tentativo per riuscire a oltrepassare questo livello, oltre a questo è necessario anche sparare ha uno o più nemici in modo da liberare la strada. alla fine di questo livello si potra affrontare il boss del gioco, rappresentato da un cumulo di blob.

## final

Tale file contiene il livello finale del gioco nominato sopra. Il funzionamento è simile a quello dei vecchi giochi arcade che tanto ci facevano divertire da piccoli. Il blob, alla destra del vostro schermo, sparerà in continuazione dei proiettili che il nostro personaggio dovrà riuscire a saltare ed evitare. Allo stesso tempo il nostro personaggio può sparare in continuazione al nemico riuscendo a togliere una piccola quantità di vita. La vita del nemico è rappresentata grazie a due rettangoli: uno rosso e uno bianco. Il rettangolo rosso e un rettangolo di fondo, quello bianco viene visualizzato sopra di quello menzionato appena prima. Ogni volta che il proiettile del nostro personaggio con l'idea con con il corpo nemico il rettangolo bianco andrà a diminuire la sua lunghezza rendendo quindi facile all'occhio capire la quantità di vita rimasta Al Blog nemico. Anche il blog nemico possiede un corpo fisico, tale da renderci disponibili i riconoscimenti della collisione. Allinterno di questo livello è presente un ascoltatore capace di aumentare la difficoltà del gioco, infatti andrà ad aumentare i pixel di scorrimento dei proiettili durante il gioco: più questa velocità di scorrimento sarà alta e più i proiettili ci arriveranno addosso più velocemente. Oltre a questo incremento abbiamo aumentato anche il ritmo dei proiettili sparati dal nemico. Tramite la linea di codice qui sotto:

table\_loop[3].\_delay =  table\_loop[3].\_delay - secondsPlayed \* 3

Siamo andati a diminuire il di lei della chiamata dei proiettili. Dentro la stanza numero tre della tabella table up è infatti presente un timer che richiama ogni tot millisecondi la funzione di creazione di un nuovo proiettile che ci verrà addosso appunto. Solo dopo aver battuto il nemico potremo arrivare all'ultima schermata della storia virgola che ci andrà a narrare la conclusione di questa avventura di Laston.

## victory

questo file Contiene una schermata di riconoscimenti: possiamo notare una foto di background che scorre assieme ai nostri nomi. Oltre ai nostri nomi abbiamo voluto inserire il logo dell'università e il nome della materia che ci ha permesso di creare questo gioco appunto successivamente abbiamo anche voluto inserire una nostra caricatura che ci vede alzare la tazza del programmatore che spero potremo vincere grazie a questo gioco.

### menu-levels.lua

Questo file è la landing page del menu dove l’utente arriverà appena cliccato il bottone ‘play’ nella schermata di home. Questa pagina ha il dovere di controllare a che livello è arrivato l’utente all’interno del gioco tramite un controllo del database *Sqlite3.*   
Prima di tutti, quindi, controlliamo se è la prima volta che l’utente entra nel gioco in modo da sapere se bisogna creare un nuovo database o dovremmo utilizzare uno già esistente. Questo si farà con questa istruzione:

local checkifdbexists = [[SELECT \* from levels]]

    local dbexists = db:exec( checkifdbexists )

    print(dbexists)

Il print potrà ritornare due valori:

* **0:** l’esecuzione della query è andata a buon fine, ciò significa che la tabella è già istanziato all’interno della memoria e non serve crearlo
* **1:** in questo caso, la query non è andata a buon fine. Ciò significa che la tabella *levels* non esiste in memoria, andrà quindi creato.

Questa parte di codice invece è il fulcro della ppagina, infatti costituisce gran parte del funzionamento di essa. All’interno della variabile *livellicompletati* ci sarà il numero dei livelli completati dall’utente che è venuto precedentemente preso dalla tabella levels del database. In base a questo numero renderò disponibili al click o meno le varie immagini del display del cellulare.

* local levels={}
* for i=1, 8 do
* local impath
* --controllo se ho già passato il livello nell'identificativo su cui è posizionata 1
* if (tonumber(livellicompletati) >= i) then
* --assegno al percorso dell'immagine l'immagine corrispondente al livello in modalità SBLOCCATA
* impath = "immagini/menu/livelli/"..i..".png"
* else
* --assegno al percorso dell'immagine l'immagine corrispondente al livello in modalità BLOCCATA
* impath = "immagini/menu/livelli/locked"..i..".png"
* end
* print("percorso: ", impath)
* levels[i] = display.newImageRect( menu, impath, 200, 200 )
* levels[i].anchorX = 0
* levels[i].anchorY = 0
* levels[i].x = checkImagePositionX(i)
* levels[i].y = checkImagePositionY(i)
* end