PLASTIBLOB

BERNARDI LORENZO

BORTOLAMIOL ALESSANDRO

DAL COLLE GABRIELE

SCOGNAMIGLIO SIMONE

LINK AL GIOCO

https://drive.google.com/open?id=1pypyBr5S_td9JDYz30V1VPTcZrgB-WMA

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 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.

STORIA

La storia di Plastiblob nasce con la voglia di unire il tema di questo esame, ovvero ambiente/natura, e la nostra parte creativa per l'adattamento al mondo di mutanti nel quale si svolgerà il gioco. Il protagonista, uno scienziato che lavora per un'azienda ecosostenibile, sta lavorando alla sua ultima creazione quando qualcosa va storto generando un'esplosione con effetti devastanti su di lui (che si trasforma in un blob) e sul mondo, mutando geneticamente chiunque entri in contatto con le "radiazioni".

Da qua comincia l'avventura: il protagonista per raggiungere l'azienda e mettere fine a tutto partirà dal mare e affronterà diverse sfide divise in base ai luoghi e ai livelli, incontrando numerosi mutanti di ogni tipo.

Il suo obiettivo non è solo quello di fermare l'espansione e porre rimedio ai danni causati dall'esplosione, Plastiblob infatti vuole ripulire il mondo intero e renderlo più eco-sostenibile e privo di inquinamento.

Questa storia, infatti, simboleggia in modo distopico un futuro che andrà incontro a grandi conseguenze negative se non iniziamo noi ogni giorno a provare ad apportare il cambiamento che vogliamo vedere.

NOTE DI PROGETTAZIONE

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 ottenere una maggiore flessibilità in termini di modifica del progetto ma anche di tempo.

Il gioco è stato sviluppato e testato sempre su un simulatore Samsung Galaxy S5 con uno schermo 16:9 e risoluzione 1080 x 1920.

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 sullo schermo in base alla grandezza del display del cellulare di chi sta utilizzando il gioco.

È importante menzionare che tutti i disegni e musiche sono state create da noi, rendendo Plastiblob un gioco perfettamente originale in ogni aspetto.

FILES DEL PROGETTO

MENU.LUA

Menu è la prima pagina su cui l'utente approda subito dopo aver cliccato sull' avvio dell'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.LUA

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**.(database)
Tramite questa implementazione siamo stati capaci di controllare e memorizzare dati utili a noi come il numero di livelli completati dall'utente e i punteggi ottenuti in ogni singolo livello.

Se l'utente si dovesse interfacciare 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 tabella "Levels" andrà a contenere un campo primario **ID** che conterrà un numero univoco per ogni 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 entrerà nel gioco, tale campo sarà 1. Se ho correttamente passato il livello 1, tale campo di tabella

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 verrà premiato con un numero pari a 2 stelle o fino a scendere a 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 'level' del db sia 2, sarà possibile andare a giocare all'interno dei livelli 1 e 2.

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

STORYLEVEL.LUA

Ogni livello ha bisogno di una piccola premessa, tramite un fumetto infatti andremo a raccontare l'avventura di Laston. Questo file permette di visualizzare le varie pagine del fumetto andando a girare pagina tramite il touch. All'interno di questo file è presente un funzionale meccanismo di scorrimento immagini: ad ogni touch si andrà ad aumentare una variabile denominata n e si andrà a richiamare la funzione che visualizza 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 è leveltarget che contiene il livello di cui stiamo leggendo la storia. Questa variabile viene valorizzata appena si entra nella scena grazie alla tabella params. Tale tabella di params viene passata dal file chiamante (nel nostro caso da menulevels) 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, giocando per la prima volta, con la stringa indicata qui sopra andremo a leggere la prima pagina del fumetto riguardante il primo livello del nostro gioco.

TUTORIAL.LUA

questo file permette la visualizzazione di un semplice sprite con grandezza 1280 x 720. Grazie all'animazione di tale sprite si andranno a visualizzare le semplici istruzioni necessarie per riuscire a giocare. Grazie ad un semplice clic sullo schermo riusciremo ad andare al livello e conseguentemente a giocarci.

Il numero del livello in cui andremo a puntare lo sapremo grazie ad una tabella dei parametri che viene passata dal file chiamante, in questo caso ci chiamerà la schermata storylevel che a sua volta è chiamata dal menu dei livelli. Si andrà quindi a delineare la perfetta gerarchia della chiamata di un livello.

cliccando sulla foto del primo livello si andrà al fumetto di quest'ultimo, Passandogli il parametro '1' (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). Dopo aver visionato anche il tutorial costruiremo un percorso di questo tipo:

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

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

Come possiamo ben notare all'interno della variabile 'leveltarget' è 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.

All'interno della scena vengono definiti tre gruppi:

- Un gruppo per il background di sfondo,
- Un gruppo per gli elementi che scrollano nello sfondo (personaggio, nemici e oggetti di plastica)
- Un ultimo che contiene i bottoni per tornare alla home.

La gerarchia di visualizzazione è quella elencata in ordine qui sopra, sotto di tutto ci sarà la foto di sfondo che scrollerà in maniera infinita, sopra ci saranno i vari sprite e sopra ancora i bottoni per tornare alla home.

La durata del livello è decisa a priori grazie ad una variabile "TimeToPlay", tale variabile viene valorizzata all'entrata della scena.

All'interno della scena 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 disponibile ad essere giocato.

All'inizio del file abbiamo dichiarato un numero massimo di elementi di plastica da raccogliere, tali elementi verranno fatti comparire sullo 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 è un valore tra i 60 e i 70 secondi.

Durante il gioco, la velocità di scorrimento dei nemici andrà ad aumentare 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 e a volte trasparenza. Ogni corpo fisico contiene al suo interno anche un ID, grazie a questo ID, riusciremo in fase di collisione a capire chi si sia scontrato con chi. Durante il nostro gioco, infatti, può capitare di scontrarsi con un elemento piuttosto che con un altro: c'è il caso in cui lo sprite si possa collidere con il sacchetto di plastica, in questo caso andrà aumentato il punteggio 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 permetterà all'utente di ricominciare il livello oppure di tornare al menù della selezione dei livelli.

GAMEOVER.LUA

Visto che sopra abbiamo parlato del game over, andiamo a parlare del file che lo rende possibile. 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 cui si andrà alla schermata di game over
- i millisecondi della transazione
- il nome del livello che ha richiamato la schermata di game over

quest'ultimo è molto importante nel funzionamento del file di game over, infatti quando clicchiamo sul bottone 'retry' andremo a consultare quest'ultimo parametro per creare il percorso del file adatto a tornare nel livello da cui siamo partiti.. Oltre a questo caso, si può anche richiamare la schermata dei menu livelli tramite un semplice bottone composto da una home.

LEVEL2

Il livello due si 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 mutante. L'obiettivo del livello è sempre quello di raccogliere più plastica possibile riuscendo ad evitare tutti i tipi di nemici presenti sullo schermo. Un'ulteriore aggiunta di questo livello è la possibilità dello sprite di sparare al nemico in modo da toglierlo completamente dal display e riuscire ad andare avanti con maggiore facilità. Le munizioni del personaggio aumentano in base al numero di oggetti di plastica raccolti durante il suo percorso, ogni munizione usata riduce il punteggio finale del livello (così da rendere il gameplay più difficile). Lo sparo è uno sprite 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 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 (funzione di riferimento a riga 639 di level2).

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 all'interno di apposite tabelle, ogni indice contiene al suo interno un nuovo timer che viene richiamato ogni tot secondi. Nel momento in cui si esce dalla scena, tutte queste tabelle vengono eliminate tramite una funzione di reset, che in base al

numero di elementi all'interno della tabella ne elimina timer e ascoltatori. Anche in questo file 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 aumenta il punteggio acquisito nel livello due.

LEVEL3.LUA

In questo terzo livello vengono aggiunte anche le piattaforme che permettono allo sprite di saltarci sopra e riuscire a raccogliere più oggetti di plastica. Il funzionamento del livello è pressoché uguale a quello dei precedenti: I nemici sono sempre due ma verranno richiamati più spesso rendendo il livello più complicato all'utente.

LEVEL4.LUA

Il quarto e ultimo livello presente all'interno del gioco. Per rendere la parte di shoot & run più difficile abbiamo inserito manualmente i timer dei vari nemici così da avere uno schema di chi apparisse e a che punto del livello. Il posizionamento dei nemici è fatto in maniera tale che in alcune parti sia indispensabile sparare. È necessario più di un tentativo per riuscire a oltrepassare questo livello!

In aggiunta però, questo livello ha anche una seconda parte dove si potrà affrontare il boss del gioco, il boss dell'azienda tramutatosi in un gigante plasticoso.

FINAL.LUA

Tale file contiene il livello finale del gioco nominato qui sopra.
Il funzionamento è simile a quello dei vecchi giochi arcade che tanto ci facevano divertire da piccoli: il blob nemico, alla destra del nostro 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 togliergli una piccola quantità di vita.

La vita del nemico è rappresentata da due rettangoli: uno rosso e uno bianco. Il rettangolo rosso è il rettangolo di fondo, quello bianco invece viene visualizzato sopra di esso.
Ogni volta che il proiettile del nostro personaggio si scontrerà con con il corpo nemico, il rettangolo bianco andrà a diminuire la sua lunghezza rendendo quindi facile capire la quantità di vita rimasta al blob nemico.
Anche il blob nemico possiede un corpo fisico, tale da renderci disponibili i riconoscimenti della collisione.
All'interno 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 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
```

Dentro l'istanza numero 3 della tabella 'table_loop' è infatti presente un timer che richiama ogni tot millisecondi la funzione di creazione di un nuovo proiettile che ci verrà addosso. Solo dopo aver battuto il nemico potremmo arrivare all'ultima schermata della storia, che ci andrà a narrare la conclusione dell'avventura di Laston.

VICTORY.LUA

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. Successivamente abbiamo anche voluto inserire una nostra caricatura che ci vede alzare la tazza del programmatore che speriamo di poter vincere grazie a Plastiblob.