

UNIVERISTÀ DEGLI STUDI DI UDINE

---

# Progetto di Laboratorio di Game Programming

---

*Autore:*

Fedrico Mattia

Maestrutti Andrea


Mauro Luca

Not Simone 139032

*email:*

not.simone@spes.uniud.it

maestrutti.andrea@spes.uniud.it



uniud.png

## **Indice**

## 1 Il problema

Il progetto d'esame relativo all'insegnamento di Laboratorio di Tecnologie Audio-VideoInterattive (LGP) consiste nell'implementazione di un videogioco utilizzando il framework di sviluppo CoronaSDK. Il tema del videogioco è l'ambiente.

Ai fini dello sviluppo è stato utilizzato il linguaggio di programmazione "Lua", che è un linguaggio utilizzato prevalentemente per scripting.

## 2 Meccaniche di gioco

### 2.1 Spiegazione del gioco

Il gioco realizzato è intitolato Save The Ocean, ed ha come scopo quello di liberare il fondale marino dalla spazzatura lasciata dall'uomo a bordo di un sommergibile, preservano così il pianeta e le creature dell'oceano.

La spazzatura è rappresentata da bottigliette di plastica e barili pieni di petrolio.

La sconfitta si verifica in due casi distinti: ogni qualvolta non si raccolga la spazzatura, oppure quando il sommergibile va a sbattere contro uno scoglio. Nel primo caso, non raccogliendo gli oggetti la "vita del mare" diminuisce e quando quest'ultima raggiunge la fine la partita finisce (raccogliendo gli oggetti, però, la vita del mare aumenta); nel secondo caso, la partita termina immediatamente.

### 2.2 Cosa l'utente deve fare (da cambiare)

L'utente dovrà raccogliere tutti i rifiuti presenti nell'ambiente. Ad ognuno corrisponde un punteggio differente: con la raccolta delle bottigliette di plastica si guadagnano 50 punti; mentre con la raccolta dei barili 100 punti.

Il gioco, inoltre, tiene traccia del punteggio massimo che viene raggiunto nelle varie sessioni. Con il passare del tempo la velocità di gioco aumenta fino a raggiungere e mantenere la velocità massima. Con l'aumentare della velocità, aumenta anche la difficoltà quindi il giocatore dovrà essere il più abile possibile a raccogliere gli oggetti.

Quando la partita termina compare la scritta "Game Over", e pochi secondi dopo la scena viene reindirizzata alla schermata iniziale.

## 3 Implementazione

Il gioco è stato implementato utilizzando il framework coronaSDK, e scrivendo il codice in "Lua".

Il codice è stato diviso in moduli per avere una visione più completa del programma e per poter apportare modifiche più agevolmente.

### 3.1 Scene

Il gioco è strutturato in "scene", gestite utilizzando il **composer**, una libreria che fornisce agli sviluppatori un modo semplice per creare e passare da una scena all'altra. Il ciclo di vita di composer inizia all'interno di main.lua. Tuttavia, main.lua stesso non è una scena del composer: viene semplicemente utilizzato per inizializzarlo, quindi avvia la prima scena tramite composer.gotoScene(). In questa chiamata, si specifica il nome della scena (file) da caricare.

Quattro diverse funzioni del ciclo di vita gestiscono gli eventi generati dal Composer: : create, show, hide and destroy. La funzione "scene:create()" viene utilizzata per creare tutti gli oggetti, caricare in memoria e di settare i display object, inclusi pulsanti, testo, grafica e altri oggetti che dovrebbero essere visualizzati prima dell'effettiva comparsa sullo schermo. La funzione "scene:show()" ...

La funzione "scene:hide()" è composta da due fasi: una fase "will", quando la scena è sullo schermo (ma sta per uscire dallo schermo); ed una fase "did", immediatamente dopo che la scena esce dallo schermo. Se viene specificato un effetto di transizione o sovrapposizione per la nuova scena, la fase "will" viene inviata prima che l'effetto inizi l'esecuzione e la fase "did" viene inviata dopo che l'effetto è terminato. Quando si esce dalla scena la funzione che viene richiamata per eliminare gli oggetti è "scene:destroy".

Le varie scene sono rappresentate da dei file.lua, che verranno chiamati mediante la funzione gotoScene() citata in precedenza, in particolare:

- menu.lua

- game.lua

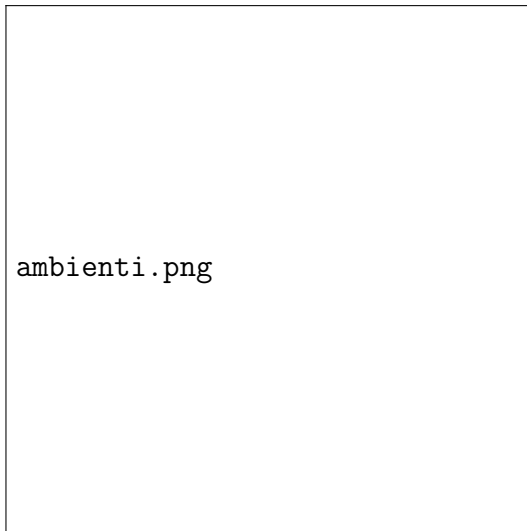
La schermata iniziale del gioco raffigura il menù, composto da tre pulsanti principali: il pulsante "play" che avvia la partita vera e propria; il pulsante "scores" che mostra i vari punteggi ottenuti; ed infine il pulsante "about" che mostra le informazioni sugli sviluppatori ed il link al git se si volesse contribuire al progetto.



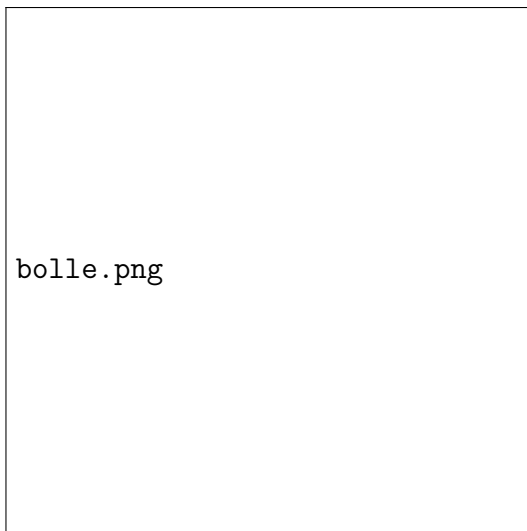
Ci sono inoltre tre bottoni secondari: uno relativo alla scelta delle diverse skin del sottomarino;



uno che mostra i vari ambienti di gioco selezionabili;



e uno che permette di scegliere il colore delle bolle che usciranno dal sottomarino.



## 4 Game.lua

Dopo aver premuto il pulsante play si passa al file "game.lua", in cui vengono realizzati tutti gli oggetti ed il codice relativo alla realizzazione della logica del gioco.

### 4.1 create()

Nella funzione create(), inizialmente vengono settate le variabili legate al composer: "startTime" è utilizzata per calcolare la velocità dello schermo in base al tempo; "gameSpeed", imposta la velocità iniziale di scorrimento dello schermo (inizialmente gameSpeed = 1). Tale velocità viene aggiornata ogni 1000 ms tramite un listener che chiama la funzione updateGameSpeed. La scelta del valore 1000ms non è casuale, ma è stato inserito in modo tale da non accumulare troppo l'incremento in quel secondo.

Le collisioni tra sottomarino ed oggetti da raccogliere sono gestite a tempo di esecuzione mediante un listener, in questo modo ogni volta che avviene una collisione viene chiamata la funzione onCollision().

Il punteggio viene rappresentato mediante un "display object" in cui è stato utilizzato il font di default salvato in una variabile locale in modo tale da renderlo più accessibile in tutte le parti del programma.

La vita del mare viene creata usando la funzione newProgressView in cui vengono inseriti 3 parametri:

- percentuale della vita: rappresentata da un valore all'interno dell'intervallo [0,1] in cui 1 rappresenta il 100 e 0 lo 0. La barra viene creata inizialmente al 100.
- posizione della barra a seconda dell'asse X: inizialmente posta al centro dell'asse X (contentCenterX)
- posizione della barra a seconda dell'asse Y:

La percentuale di progresso della barra viene gestita da un metodo setProgress inserito all'interno della funzione newProgressView.