Corso di Programmazione ad Oggetti – A. A. 2013/2014
Corso di Laurea in Ingegneria e Scienze Informatiche – Università di Bologna

Alien Escape

Programmazione ad oggetti

Matteo Baldelli

01/02/16

1. Analisi del problema

Alien Escape è un classico gioco di tipo runner games, sviluppato in Java grazie l'uso della libreria gdx per renderlo multipiattaforma.
Il gioco è composto da un alieno, il personaggio del gioco, che corre per scappare dalla terra ma durante lungo il suo percorso troverà dei contadini e delle mucche che cercheranno di ostacolarlo nel suo tentativo di fuga, l'alieno dovrà cercare di saltare quanti più nemici possibili per continuare la sua fuga.
Lo scopo del gioco è sostanzialmente quello di saltare più ostacoli possibile, tramite un doppio touch sullo schermo, per accumulare punti e fare il punteggio più alto possibile.
Il gioco parte da un menu iniziale dove hai la possibilità di iniziare una nuova partita e dove si vede l'alieno pronto a iniziare la sua fuga, una volta avviata una nuova partita il protagonista inizierà la sua fuga, con un click sullo schermo il protagonista effettuerà un salto e può aumentare la distanza del salto con un altro click sullo schermo.
Ad ogni nemico evitato si conquisterà un punto, la somma dei punti viene visualizzato nella parte alta dello schermo, una volta che l'alieno inciamperà in un ostacolo il gioco viene fermato e appare una schermata con la visualizzino del punteggio record e del punteggio attuale.
Tale schermata a due tipi di visualizzazione:

- High Score: hai perso battendo il tuo record precedente
- Game Over: hai perso senza ribattere il tuo record precedente

Tutto questo è accompagnato da effetti sonori e grafici a seconda delle azioni del gioco.
Tale immagini sono state realizzate appositamente da me, mentre le tracce audio e i file font le ho preso su internet da altri progetti open source per poi integrarle nel progetto.
Volendo sviluppare questo gioco per multipiattaforma ho scelto di usare gdx, che mi permette di creare un progetto core per poi estenderlo alle piattaforme Android, iOS, HTML e applicazioni Desktop.

2. Progettazione

Ho cercato di sviluppare il progetto con una logica MVC (Model-View-Controller), ossia di tenere separata la logica del gioco racchiusa nei modelli, ossia le classi relative a tutti gli elementi del gioco, dalla parte grafica, ossia le classi per disegnare e animare gli elementi del gioco, questo grazie all'uso di controller, nel mio caso GameWorld, che mette assieme le logiche del gioco e i relativi effetti grafici.
Grazie a gdx mi è stato possibile gestire gli input per i vari tipi di dispositivi e la gestione delle collisioni degli elementi del gioco.
L'import delle librerie di gdx è gestito tramite gradle in modo da poter scaricare le librerie di cuoi hai bisogno al momento dell'importazione del progetto, ma purtroppo ho avuto diversi problemi al momento del download delle libreria per via del percorso del sdk di android (usata per il relativo progetto), per risolvere questo problema ho caricato il jar delle librerie usate al interno del progetto su bitbucket.
Non mi piace come soluzione, ma è stata l'unica in assenza del gradle a garantirmi un corretto import del progetto.
Ho diviso l'applicazione nei seguenti package che poi andremo ad analizzare:

- com.matteobaldelli.AEHelpers
- com.matteobaldelli.AlienEscape
- com.matteobaldelli.GameObjects
- com.matteobaldelli.GameWorld
- com.matteobaldelli.Screens
- com.matteobaldelli.TweenAccessors
- com.matteobaldelli.ui

3. Package e Classi

AEHelpers

Questo package contiene le classi che mi aiutano a gestire gli assets e gli input del gioco.

- **AssetLoader**: si occupa di caricarmi tutti gli asset del gioco, ovvero immagini, audio, il file per high score e i file per i font. Il particolare di questa classe è che per le immagini viene usata una singola immagine poi viene mappata tramite gli oggetti Texture per dividerla in più immagini.
- **InputHandler**: viene dichiarata come implementazione della classe InputProcessor di gdx, di questa classe non c'è molto da dire, viene usata semplicemente per intercettare le azioni degli utenti e richiamare gli oggetti coinvolti.

AlienEscape

Package principale del gioco, contiene una sola classe che viene chiamata per avviare il gioco.

- **AEGame**: estensione delle classe Game di gdx, vengono chiamati i metodi create e dispose per avviare il gioco.

GameObjects

Questo secondo me è il package più importante del gioco, qui ci sono tutte le classi che compongono tutti gli elementi del gioco che interagiscono tra di loro.

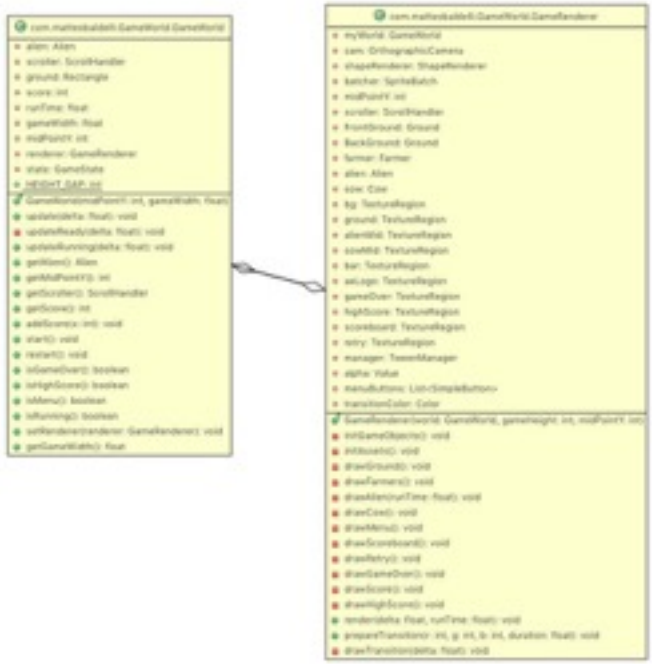
- **Alien**: questa è la classe che gestisce il personaggio principale del gioco, alla sua inizializzazione viene dichiarata l'altezza, la larghezza, la posizione e la velocità del personaggio.
  Possiamo vedere qui il metodo update che ha il compito di aggiornare la posizione e le velocità del personaggio, il metodo onClick che si occupa di far saltare il personaggio e controllare che non abbia già effettuato due salti, slide che permette al personaggio di restare sul terreno, die e onRestart che si occupano di finire e far ripartire il gioco.
- **Cow**: estensione della classe Scrollable, la classe si occupa di instanziare una mucca che corre incontro all'alieno per ostacolarlo, aggiornando la sua posizione e controllando se effettua o no una collisione con esso, una volta che il personaggio supera l'ostacolo viene ricreato in una posizione e con una velocità random.
- **Farmer**: estensione della classe Scrollable, la classe si occupa di instanziare un contadino che aspetta l'arrivo del alieno per ostacolarlo, aggiornando la sua posizione e controllando se effettua o no una collisione con esso, una volta che il personaggio supera l'ostacolo viene ricreato in una posizione random.
- **Ground**: estensione della classe Scrollable per il terreno, niente di speciale da segnalare, da una base all'alieno su cui correre e ai nemici per interromperlo.
- **Scrollable**: classe base per tutti gli oggetti che scorrono lungo il percorso dell'alieno e fornisce metodi standard per le altre classi.

- **ScrollHandler**: Uso questa classe per gestire tutti gli oggetti che scorrono luna il percorso dell'alieno, in pratica tutti gli oggetti che sono estensione della classe "Scrollable", uso l'update di questa classe per gestire tutte le altri, come per lo stop e il restart.
  Mettono collides usato per controllare se il personaggio ha superato l'ostacolo e che non sia andato a sbattere.

GameWorld

Questo package contiene le due classi principali che fanno da unione per tutti gli altri elementi, ovvero il centro del controller della mia applicazione.

- **GameRender**: in questa classe è presenta tutta la logica della views della applicazione, ovvero vengono disegnati gli asset caricati dalla classe AssetLoader nelle posizioni degli oggetti corretti. Sono stati creati funzioni per disegnare ogni elemento del gioco in modo da poter ben separare ogni elemento.
- **GameWorld**: qui vengono richiamati tutti gli elementi del gioco per unirli secondo la logica di esso, il metodo centro è updateRunning, dove ci sono i vari eventi che l'alieno può causare una volta in corsa.



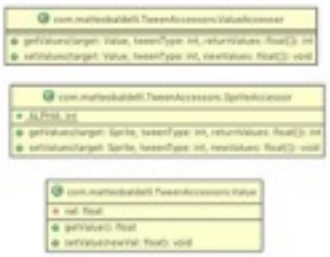
Screens

Package usato per gestire lo schermo di gioco

- **GameScreen**: implementazione delle classe Screen di gdx, si prende carico di gestire lo schermo di dove verrà eseguito il gioco per dare riferimenti alle dimensioni di tutti gli altri elementi presenti in gioco.
- **SpalshScreen**: anche questa implementazione delle classe Screen di gdx, viene usata solo per la SplashScreen iniziale.

TweenAccessors

Questo package contiene le classi per l'uso della libreria Tween Engine, una libreria che ho usato per gestire gli effetti di transizioni. Non scrivo in dettaglio come ho sviluppato queste classi in quanto le ho copiate dal sito dello sviluppatore della libreria.

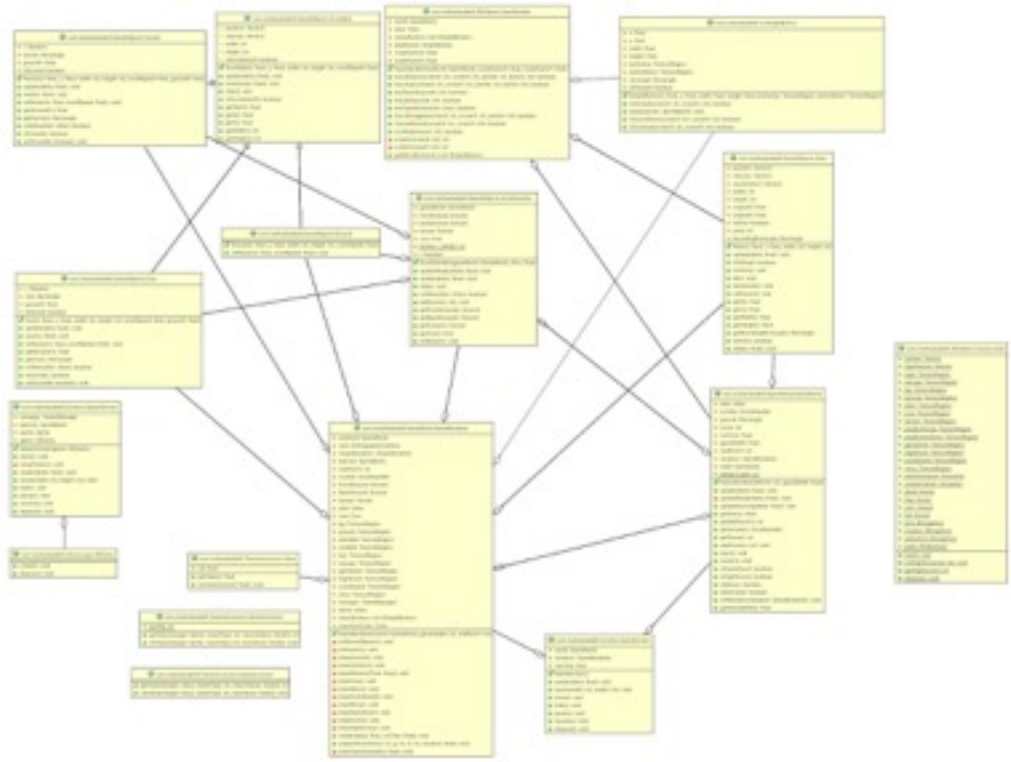


Ui

Package usato per creare gli elementi della ui del gioco.

- SimpleButton: Classe usato per gestire tutti i pulsanti presenti nel gioco.

Infine il diagramma UML di tutte le classi



5. Conclusioni

Il progetto è stato sviluppato nelle 100 ore previste, la parte in cui ho speso la maggior parte del tempo è stata per imparare a gestire gli elementi di gdx, ossia l'alieno, il contadino e la mucca. Infatti per questi oggetti ho dovuto controllare le loro collisioni e darli velocità e posizioni diverse e aggiornarle nel corso della partita.
Ho notato che con l'inserimento della mucca ogni tanto il gioco scatta, ma sinceramente non ho capito da cosa è dovuto, in quanto mi succedeva pochissime volte durante le prove dello sviluppo, potrebbe essere stata la batteria scarica del computer.
Come spiegato in precedenza sono dispiaciuto di non essere riuscito ad usare gradle per l'import delle librerie, infatti i jar che ho caricato nel progetto sono molto pensanti.
Anche se ho messo i progetti per tutte le piattaforme l'unico eseguibile è quello desktop, per quello android come ho detto ho avuto problemi con sdk, per quel-

lo iOS mi è stata chiesta una chiave che non era in mio possesso, sarei stato invece contento di aver fatto partire la versione html, in quanto sviluppatore web, per capire in che modo funzionava ma a quanto ho letto sulla documentazione di libGdx mi mancavano degli strumenti su eclisse per poterlo configurare.
Purtroppo le immagini del gioco non mi sono venute molto bene, infatti la grafica non è il mio forte, ma in quanto non parte della programmazione spero che non mi vengano valutate.
Concludo dicendo che sono felice del mio risultato dopo le 100 ore di programmazione ma mi sono trovato a disagio a usare eclipse come Ide e mercurial per bitbucket, sono abituato da un paio di anni ad usare bitbucket con git (anche se è quasi uguale a mercurial) e Ide come PHPStorm e Android Studio che secondo il mio parare sono molto più avanti come funzioni.

Bibliografia

- https://libgdx.badlogicgames.com/
- http://www.aurelienribon.com/blog/projects/universal-tween-engine/