CAPITOLO CINQUE

PROGRAMMAZIONE

2011-1-ES1-LEO05-35968

Indice

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

1. INTRODUZIONE A CORONA SDK: UNA PIATTAFORMA DI SVILUPPO

Introduzione

Corona SDK è un software eccellente per ogni tipo di sviluppatore o progettista di tecnologie mobili, dal principiante all'esperto. Questo tutorial è una guida all'uso di questa piattaforma e mostra come creare contenuti personalizzati.

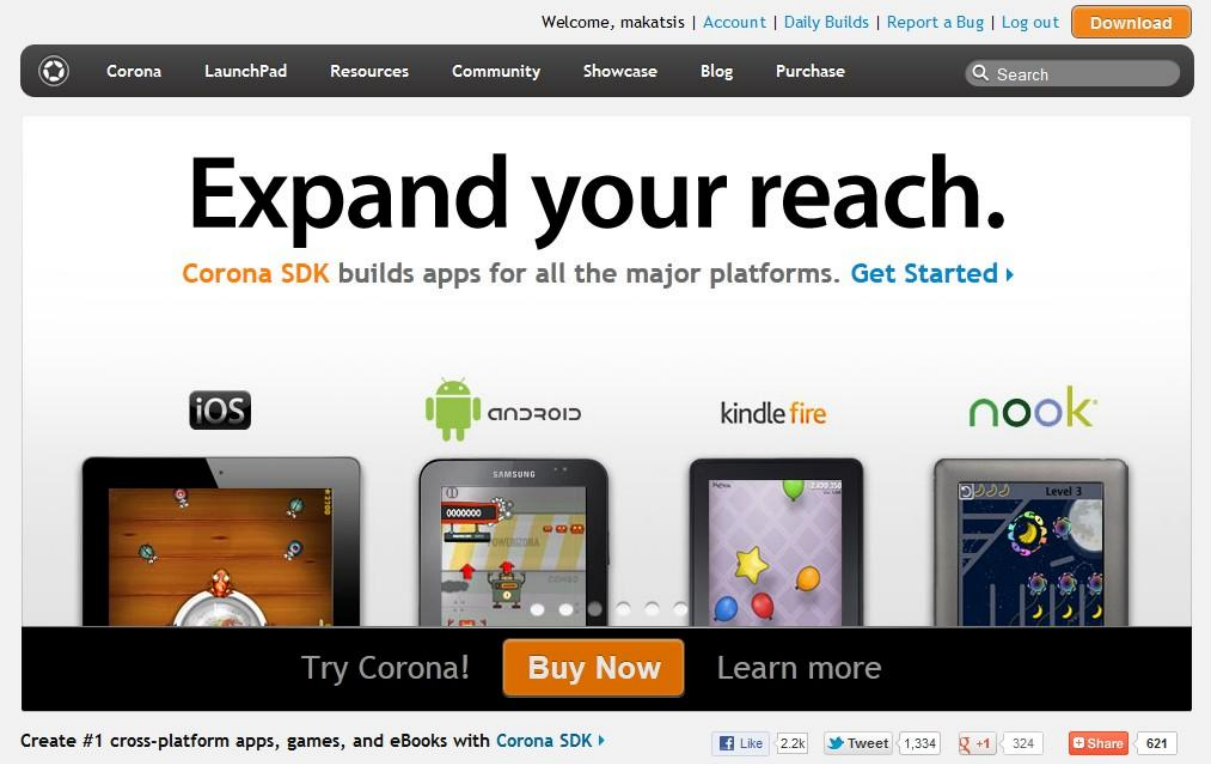


Figura 1: Sito ufficiale di Corona

Il **sito ufficiale di Corona** riporta quanto segue:

Corona è uno strumento facile e veloce da usare per realizzare giochi e applicazioni per iPhone, iPad e Android.

Le applicazioni che crea Corona arrivano a 30 fps in 300k, e il motore dedicato a grafica e animazioni sfrutta pienamente l'accelerazione hardware OpenGL.

Corona SDK è il primo dei prodotti della famiglia Ansca di Corona che crea applicazioni e giochi per iPhone, con alte caratteristiche multimediali e ricchezza grafica elevata. Con Corona è possibile, infatti, creare facilmente applicazioni per iPhone nel giro di poche ore. Non è, inoltre, richiesta la conoscenza del linguaggio C++.

Ansca è la compagnia che si trova dietro Corona, e SDK consente ai programmatori di creare applicazioni veloci e potenti che hanno accesso agli altri framework di API.

Corona SDK ha una molteplicità di variabili che lo rendono dotato di molte caratteristiche, pertanto è un software affidabile ed efficace per creare applicazioni. Alcune di queste caratteristiche sono:

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

- **Sviluppo di Native Application**: Corona è implementato al 100% tramite **Objective-C/C++**, perciò non è vincolato alle nuove regole dell'Apple iOS 5 sull'uso di strumenti esterni di sviluppo.
- **Integrazione automatica OpenGL-ES**: il software non ha bisogno di classi o funzioni estensive per creare semplici modifiche a schermo.
- **Sviluppo per ogni piattaforma**: Corona può creare app per **iOS** (iPhone, iPod Touch, iPad) e dispositivi **Android**.
- **Performance**: Corona è ottimizzato per usare hardware con caratteristiche avanzate per creare prodotti eccellenti nel campo di giochi e applicazioni.
- **Caratteristiche del dispositivo**: utilizza controlli e parti hardware del dispositivo come la fotocamera, l'accelerometro, il gps, ecc.
- **Facile da apprendere**: Corona usa il **linguaggio di programmazione Lua**, che è potente e facile da imparare.

Piattaforme supportate

Il più grande vantaggio di Corona consiste nel permettere all'utente di lavorare con un codice base e di realizzare prodotti per molti dispositivi differenti.

Specificamente, **Corona SDK** permette di creare applicazioni per tutti i dispositivi dotati di **iOS** e per i dispositivi **Android**.

Sviluppo tramite Lua

Figura 2: Il linguaggio di programmazione Lua

Corona si basa sul **linguaggio di programmazione Lua** per creare applicazioni. Lua è un linguaggio usato comunemente per sviluppare giochi. Tale linguaggio può, inoltre, essere utilizzato in modi

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

differenti. La sintassi di Lua può essere paragonata a linguaggi come **JavaScript** o **ActionScript 3**, molto facili da imparare.

Attualmente molti linguaggi di programmazione sono disponibili per gli utenti, ma hanno un alto grado di complessità necessitando di centinaia di migliaia di linee per scrivere un programma. Per questo motivo, tali software offrono pacchetti, sistemi a caratteri complessi, molteplicità di costruzioni e migliaia di documenti da studiare per aiutare gli utenti nella scrittura di un programma.

Lua non aiuta gli utenti scrivendo programmi di centinaia di migliaia di linee. Lua, piuttosto, prova ad aiutarli a risolvere il problema in poche centinaia di linee, o anche meno. Per raggiungere questo obiettivo, Lua si basa sull'estensibilità, come molti altri linguaggi. Al contrario degli altri linguaggi, però, Lua è esteso facilmente non solo con il software con cui è scritto, ma anche con software scritti in altri linguaggi, come **C** e **C++**.

Lua è stato progettato, fin dall'inizio, per essere integrato con software scritti in linguaggio C e altri linguaggi convenzionali. Questo dualismo nei linguaggi porta molti benefici. Lua è un linguaggio ristretto e semplice, infatti, in parte anche perché non prova a fare ciò per cui già il linguaggio C è più che valido, come sheer performance, operazioni di basso livello, o particolari interfacce software. Lua si affida al linguaggio C per questi compiti. Ciò che offre Lua è qualcosa in cui C non è efficace: una giusta distanza dalla componente hardware, strutture dinamiche, nessuna ridondanza, facilità di test e debugging. Lua, perciò, ha un ambiente sicuro, una gestione automatica della memoria e grande facilità nella gestione di stringhe e altri tipi di dati di formato dinamico.

Più che essere un linguaggio estensibile, Lua è un linguaggio unificante. Lua supporta, infatti, un approccio basato sui componenti per lo sviluppo dei software, in cui si crea un'applicazione unendo insieme componenti di alto livello che già esistono. Solitamente questi componenti sono scritti in un linguaggio rigido, statico, come C o C++, Lua è la colla che può essere usata per unire le parti. Generalmente le componenti (o gli oggetti) rappresentano prodotti concreti (come widget e strutture di dati) che non sono soggetti a grandi cambiamenti durante lo sviluppo del programma e che occupano la maggior parte della CPU per il programma finale. Lua fornisce, dunque, la forma finale dell'applicazione, che probabilmente cambierà molto durante il ciclo di vita del prodotto. Al contrario delle altre tecnologie collanti, inoltre, Lua è un vero e proprio linguaggio. È possibile, pertanto, usare Lua non solo come collante, ma anche adattarlo e riformarlo, o addirittura adoperarlo per creare intere nuove componenti.

Lua, naturalmente, non è il solo linguaggio possibile. Ci sono altri linguaggi che si possono usare più o meno per gli stessi scopi, come **Perl**, **Tcl**, **Ruby**, **Forth**, e **Python**. Le seguenti caratteristiche di Lua lo collocano sopra gli altri linguaggi. Sebbene gli altri linguaggi condividano alcune di queste caratteristiche, infatti, nessun altro offre un simile profilo:

- **Estensibilità**: l'estensibilità di Lua è così notevole che molte persone lo considerano non come un linguaggio ma come uno strumento per costruire linguaggi con domini specifici. Lua è stato progettato partendo da un contesto iniziale ed espandendolo, attraverso il codice Lua e il codice esterno C. Implementa la maggior parte delle sue funzionalità, infatti, attraverso librerie esterne. Si interfaccia, dunque, molto facilmente con Lua e C/C++ e altri linguaggi come **Fortran**, **Java**, **Smalltalk**, **Ada**, e altri.

- **Semplicità**: Lua è un linguaggio semplice e ristretto. Ha pochi concetti (sebbene siano potenti).Questa semplicità rende Lua facile da apprendere e contribuisce alla sua implementazione. La sua completa distribuzione (il codice sorgente, il manuale e i codici binari per alcune piattaforme) entrano comodamente in un floppy disk.

- **Efficienza**: Lua ha un'implementazione molto efficiente. Benchmark indipendenti indicano Lua come uno dei più veloci linguaggi nell'ambito dei linguaggi di scripting.

- **Portabilità**: quando parliamo di portabilità, non ci riferiamo solo alla possibilità di utilizzare Lua sia su piattaforme **Windows** che **Unix**. Ci riferiamo all'utilizzo di Lua su tutte le piattaforme possibili: **NextStep**, **OS/2**, **PlayStation II (Sony)**, **Mac OS-9** e **OS X**, **BeOS**, **MS-**

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

DOS, **IBM mainframes**, **EPOC**, **PalmOS**, **RISC OS**, più, ovviamente, tutti i tipi di Unix e di Windows. Il codice sorgente per tutte queste piattaforme è virtualmente lo stesso. Lua, pertanto, non usa differenti codici per adattarsi alle varie piattaforme, ma lo standard **ANSI (ISO) C**. In questo modo non c'è bisogno di adattarsi a un nuovo ambiente.

La gran parte dell'efficienza di Lua deriva dalle sue librerie. Questo non avviene a caso. Una delle principali forze di Lua è, infatti, l'estensibilità attraverso nuovi caratteri e nuove funzioni. Ci sono molte caratteristiche che contribuiscono alla sua forza: i caratteri del software permettono di operare con un alto grado di polimorfismo, la gestione automatica della memoria semplifica le interfacce, perché non c'è bisogno di decidere riguardo la locazione e la dislocazione della memoria, il software, inoltre, consente l'utilizzo di funzioni versatili grazie a un alto grado di parametrizzazione che si viene a creare tramite funzioni di alto livello e funzioni anonime.

Lua dialoga con un numero di librerie standard. Quando si utilizza Lua in un ambiente fortemente limitato, come processori incorporati, è utile scegliere attentamente di quale libreria si necessita. Se le limitazioni sono elevate, inoltre, è possibile entrare facilmente all'interno del codice sorgente delle librerie e scegliere quali funzioni si devono tenere una a una. È necessario ricordare, comunque, che Lua è abbastanza piccolo (anche con tutte le librerie standard) e nella maggior parte dei sistemi è possibile usare l'intero pacchetto senza problemi.

Lua editors

Corona non implementa un editor esclusivo, ma ci sono alcuni grandi editor già disponibili per essere usati:

Gratis:

- **Eclipse**, usando il plugin Lua Eclipse.
- **LuaEdit**, LuaEdit è un IDE/Debugger/Script Editor progettato per la versione 5.1 di Lua.
- **NotePad++**, un editor dotato di codice sorgente gratis che supporta alcuni linguaggi di programmazione, incluso Lua.
- **TextWrangler**, un potente editor di testi dotato di strumenti di amministrazione di Unix e di server.

Commerciali:

- **TextMate**, disponibile solo per Mac OS X.
- **BBedit**, un editor di testi professionale in HTML per Macintosh.
- **Decoda**, un ambiente di sviluppo professionale per il debugging degli script di Lua nelle applicazioni.

Creare un primo programma

Per approcciarsi al software Corona cominciamo con la classica applicazione "Hello World".

Apri il tuo editor preferito fra gli editor compatibili con Lua e scrivi il codice seguente: `print("Hello World!").`

Crea una nuova cartella chiamandola *HelloWorld* e salva al suo interno il file assegnandogli il nome *main.lua*. Lanceremo questa app nei prossimi passaggi.

Lifelong Learning Programme
Education and Culture DG
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

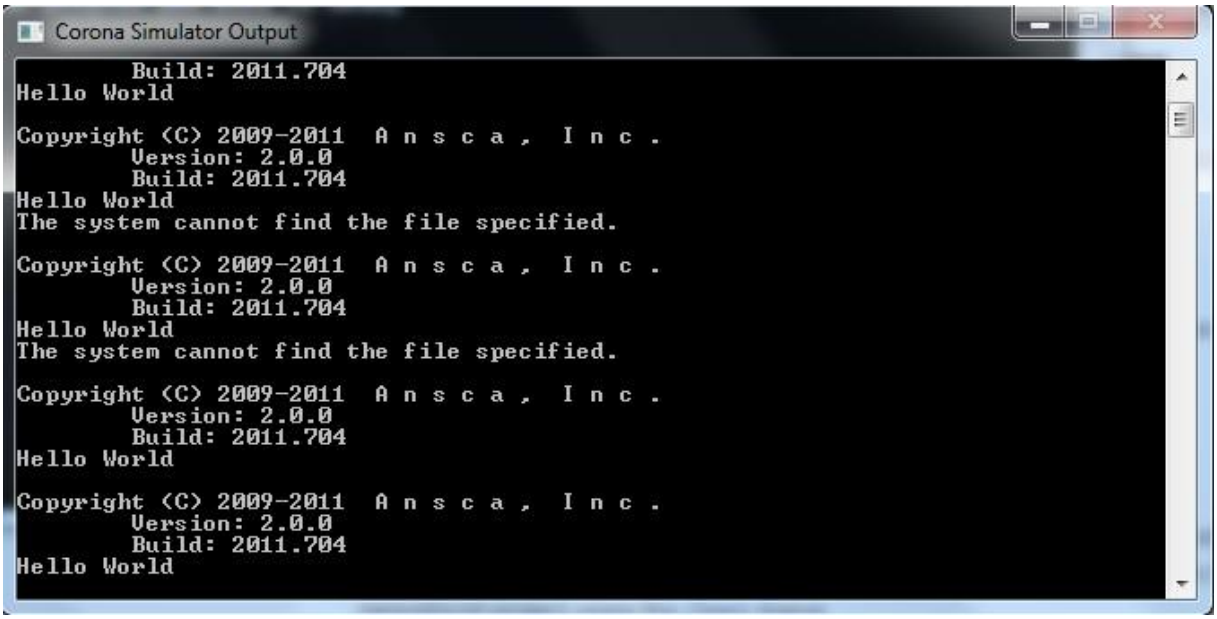


Figura 3: Terminale di Corona

In questo modo si aprirà anche il *Corona Simulator*.

Simulator

Per accedere al simulatore è necessario usare il **Corona specific API's**.

Nel tuo file main.lua scrivi questa stringa e apri nuovamente il programma:

```
local myTextField = display.newText("Hello World!", 1, 20, nil, 14);myTextField:setTextColor(255, 255, 255);
```

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Figura 4: Corona Simulator

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

2. CORONA SDK: COME CREARE UN'APP PER UN OROLOGIO ANALOGICO

Usando il Corona API's, creeremo un orologio analogico elementare. La grafica sarà esportata da un editor di immagini di tua scelta e, in seguito, implementata da Lua. In questo modo imparerai come testare la tua applicazione usando il simulatore e a costruire la tua app per il test dei dispositivi.

Selezionare il Target Device

La prima cosa che devi fare è selezionare la piattaforma in cui vuoi implementare la tua app, così facendo potrai scegliere la dimensione per le immagini che userai.

La piattaforma **iOS** ha tre formati:

- **iPad**: 1024x768px, 132 ppi
- **iPhone/iPodTouch**: 320x480px, 163 ppi
- **iPhone 4**: 960x640px, 326 ppi

Android, essendo una piattaforma open, funziona in maniera leggermente differente. Si possono infatti riscontrare delle differenze nella risoluzione dello schermo:

- **Nexus One**: 480x800px, 254 ppi
- **Droid**: 854x480px, 265 ppi
- **HTC Legend**: 320x480px, 180 ppi

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Interfaccia

Figura 5: Interfaccia dell'orologio

Codice

Background

La prima cosa da fare è aggiungere il background dell'orologio:

```
local background = display.newImage("background.png")
```

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Questa stringa crea il background locale variabile e usa il display API per aggiungere l'immagine specificata. Per default l'immagine è aggiunta a 0,0.

Inserimento delle lancette

Ripetiamo il processo con le lancette mettendole al centro:

```
local hourHand = display.newImage("hourHand.png", 152, 185)
local minuteHand = display.newImage("minuteHand.png", 152, 158)
local center = display.newImage("center.png", 150, 230)
local secondHand = display.newImage("secondHand.png", 160, 155)
```

Punto di riferimento

Per posizionare correttamente l'immagine, dobbiamo modificare il punto di riferimento in modo da spostare l'immagine dal fondo al centro:

```
hourHand:setReferencePoint(display.BottomCenterReferencePoint)
minuteHand:setReferencePoint(display.BottomCenterReferencePoint)
secondHand:setReferencePoint(display.BottomCenterReferencePoint)
```

Posizione iniziale

Qui impostiamo la posizione iniziale delle lancette. Poi imposteremo la rotazione.

```
local timeTable = os.date("*t")

hourHand.rotation = timeTable.hour * 30 + (timeTable.min * 0.5)
minuteHand.rotation = timeTable.min * 6
secondHand.rotation = timeTable.sec * 6
```

Memoria

La variabile *timeTable* sarà usata solo una volta al lancio dell'applicazione, così non c'è bisogno di tenerla in memoria. Per liberare la memoria usata dalla variabile (che è quasi niente) impostiamo il suo valore su *nil*, in questo modo viene cancellato:

```
timeTable = nil
```

Movimento delle lancette

Queste stringhe del codice servono per far ruotare le lancette. È lo stesso codice di prima, solo questa volta è coinvolto in una funzione che viene eseguita ogni secondo da un *Timer*.

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

```
local function moveHands(e)
	local timeTable = os.date("*t")
	hourHand.rotation = timeTable.hour * 30 + (timeTable.min * 0.5)
	minuteHand.rotation = timeTable.min * 6
	secondHand.rotation = timeTable.sec * 6
end
```

Timer

Il *Timer* è in esecuzione ogni secondo e deve svolgere una funzione specifica, cioè la M*oveHands* function che abbiamo creato nello scorso passaggio.

This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

2011-1-ES1-LEO05-35968

Figura 6: Orologio analogico realizzato con Corona Simulator

3. APPLICAZIONE : ACCELEROMETRO

Usando **Corona API's**, creeremo un'applicazione che registra il movimento del dispositivo muovendo un oggetto sullo schermo.

Selezionare il Target Device

La prima cosa che devi fare è selezionare la piattaforma su cui vuoi che la tua app venga implementata, in questo modo potrai scegliere la grandezza delle immagini che userai.

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

La piattaforma **iOS** ha tre formati:

- **iPad**: 1024x768px, 132 ppi
- **iPhone/iPodTouch**: 320x480px, 163 ppi
- **iPhone 4**: 960x640px, 326 ppi

Android, essendo una piattaforma open, funziona in maniera leggermente differente. Si possono infatti riscontrare delle differenze nella risoluzione dello schermo:

- **Nexus One**: 480x800px, 254 ppi
- **Droid**: 854x480px, 265 ppi
- **HTC Legend**: 320x480px, 180 ppi

Interfaccia

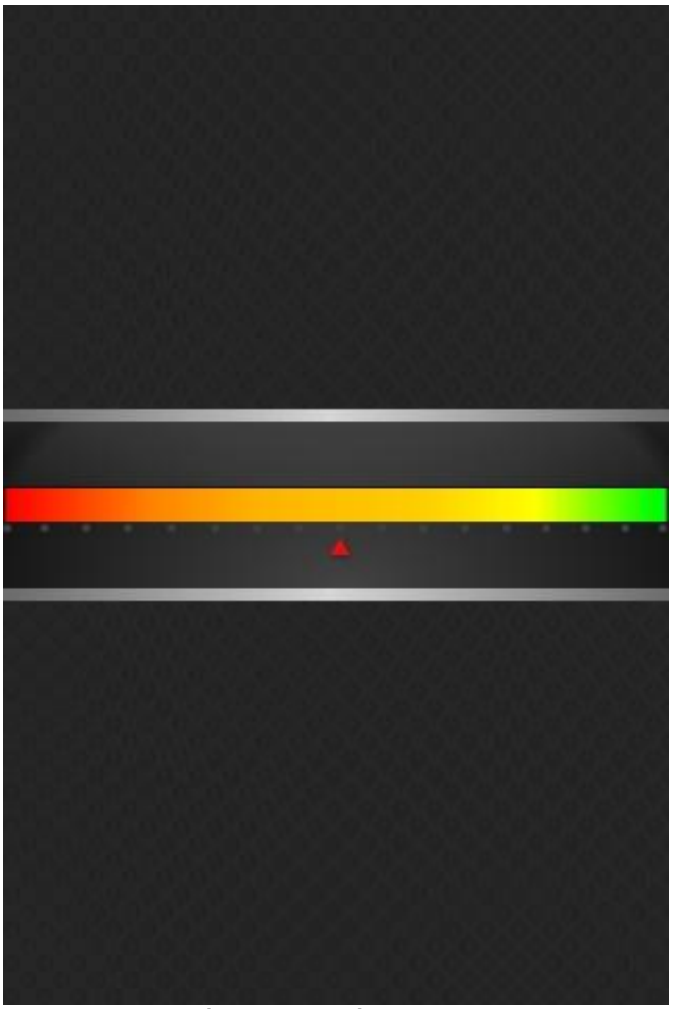


Figura 7: Accelerometero

Questa è l'interfaccia grafica che useremo, include un triangolo che servirà come punto di riferimento per la posizione.

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

Codice

Nascondere la barra dello stato

Per prima cosa, nascondiamo la barra dello stato, cioè la barra in cima allo schermo che mostra il tempo, i segnali e altri indicatori.

```
display.setStatusBar(display.HiddenStatusBar)
```

Background

Adesso aggiungiamo la app per lo sfondo.

```
local background = display.newImage("background.png")
```

Questa stringa crea la variabile local background e usa il display API per aggiungere l'immagine specifica alla scena. Per impostazioni di default, l'immagine è aggiunta a 0,0 e sfrutta l'angolo superiore sinistro come punto di riferimento.

Indicatore

Ripetiamo il processo con la posizione dell'indicatore di immagine, mettendolo al centro dello schermo.

```
local indicator = display.newImage("indicator.png")

indicator:setReferencePoint(display.CenterReferencePoint)
indicator.x = display.contentWidth * 0.5
indicator.y = display.contentWidth * 0.5 + 100
```

Variabili

Le seguenti variabili vengono usate nella gestione dell'accelerometro.

- **acc**: una tabella che verrà usata come ricevitore dell'accelerometro.
- **centerX**: registra i valori dell'immagine.

```
local acc = {}
local centerX = display.contentWidth * 0.5
```

Funzione accelerometro

Questa funzione usa la tabella *acc* per creare un registratore dell'accelerometro, la proprietà *xGravity* (parte dell'acceleremetro) e la variabile *center X* muovono l'indicatore di posizione in base alla posizione calcolata.

```
function acc:accelerometer(e)
	indicator.x = centerX + (centerX * e.xGravity)
```


This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

```
end
```


Questo bilancerà il nostro indicatore quando cambia l'inclinazione del dispositivo, la proprietà *xGravity* gestirà i movimenti laterali, la proprietà *yGravity* l'inclinazione in alto e in basso.

Registratore dell'accelerometro

Gli eventi dell'accelerometro sono basati su runtime, dunque dobbiamo usare la parola chiave Runtime per aggiungere il registratore.

```
Runtime:addEventListener("accelerometer", acc)
```

4. SVILUPPARE UN'APPLICAZIONE : LA SFERA MAGICA

Uaando lo *Shake* Event presente nel **Corona API**, creeremo un'applicazione che genera un risultato casuale partendo da parole predefinite. Impareremo, inoltre, a creare semplici animazioni usando i metodi *transition*.

Selezionare il Target Device

La prima cosa che devi fare è selezionare la piattaforma per cui vuoi implementare la tua app, in questo modo potrai scegliere la dimensione delle immagini che userai.

La piattaforma **iOS** ha tre formati:

- **iPad**: 1024x768px, 132 ppi
- **iPhone/iPodTouch**: 320x480px, 163 ppi
- **iPhone 4**: 960x640px, 326 ppi

Android, essendo una piattaforma open, funziona in maniera leggermente differente. Si possono infatti riscontrare delle differenze nella risoluzione dello schermo:

- **Nexus One**: 480x800px, 254 ppi
- **Droid**: 854x480px, 265 ppi
- **HTC Legend**: 320x480px, 180 ppi

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Interfaccia

Figura 8: Sfera Magica

Questa è l'interfaccia grafica che useremo, include un triangolo che servirà come ottaedro della Sfera Magica.

Codice

Per prima cosa, nascondiamo la barra dello stato, cioè la barra in cima allo schermo che mostra il tempo, i segnali e altri indicatori.

```
display.setStatusBar(display.HiddenStatusBar)
```

Background

Adesso aggiungiamo l'applicazione per lo sfondo.

```
local background = display.newImage("background.png")
```

Questa stringa crea la variabile local background e usa il display API per aggiungere l'immagine specifica alla scena. Per impostazioni di default, l'immagine è aggiunta a 0,0 e sfrutta l'angolo superiore sinistro come punto di riferimento.

Ottaedro

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

Ripetiamo, quindi, il processo con l'immagine dell'ottaedro, posizionandola al centro dello schermo.

```
local octohedron = display.newImage("octohedron.png", 110, 186)
octohedron.isVisible = false
```

L'ottaedro sarà invisibile come impostazione di default e apparirà non appena il dispositivo verrà scosso.

Campo di testo (TextField)

La seguente stringa crea il campo di testo centrale (TextField) che mostrerà la frase casuale quando il dispoitivo verrà scosso.

```
local textfield = display.newText("", 0, 0, native.systemFontBold, 14)

textfield:setReferencePoint(display.CenterReferencePoint)
textfield.x = display.contentWidth * 0.5
textfield.y = display.contentHeight * 0.5
textfield:setTextColor(255, 255, 255)
```

Variabili necessarie

Le seguenti variabili sono necessarie per gestire l'applicazione nel momento della scossa:

- **shake**: una tabella che sarà usata con funzione di registratore.
- **options**: un memorizzatore di parole che possono essere mostrate dalla Sfera Magica.

```
local shake = {}
local options = {"Probably Not", "No.", "Nope", "Maybe", "Yes", "Probably", "It's Done", "Of Course"}
```

Funzione shake

Questa funzione recepisce le scosse e mostra l'ottaedro con le frasi.

```
function shake:accelerometer(e)
	if(e.isShake == true) then
		octohedron.isVisible = true
		transition.from(octohedron, {alpha = 0}) -- Show octohedron
		textfield.text = options[math.random(1, 8)]
		transition.from(textfield, {alpha = 0})
	end
end
```

2011-1-ES1-LEO05-35968

Registratore dell'accelerometro

Gli eventi dell'accelerometro sono basati su runtime, dunque dobbiamo usare la parola chiave Runtime per aggiungere il registratore.

```
Runtime:addEventListener("accelerometer", shake)
```

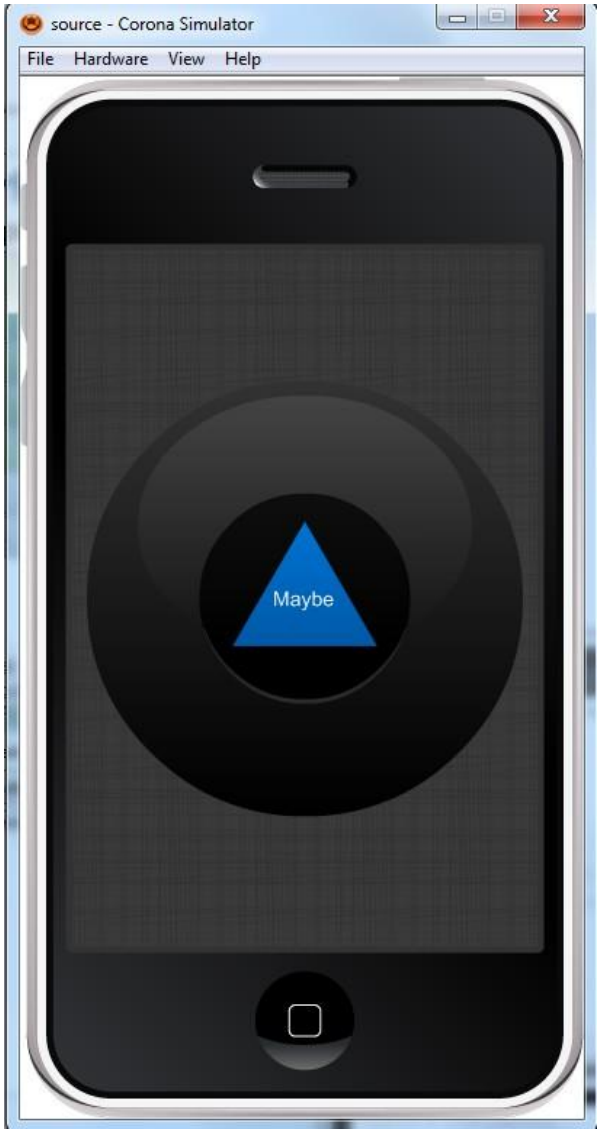


Figura 9: Sfera Magica – Risultato Finale

5. LAVORARE CON GLI ALERT

Un Alert è un sistema predefinito per mostrare informazioni all'utente. Gli alert sono comunemente usati per mostrare brevi messaggi e possono includere una o più opzioni per determinare un'azione posteriore.

Selezionare il Target Device

La prima cosa che devi fare è selezionare la piattaforma per cui vuoi implementare la tua app, in questo modo potrai scegliere la dimensione delle immagini che userai.

La piattaforma **iOS** ha questi formati:

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

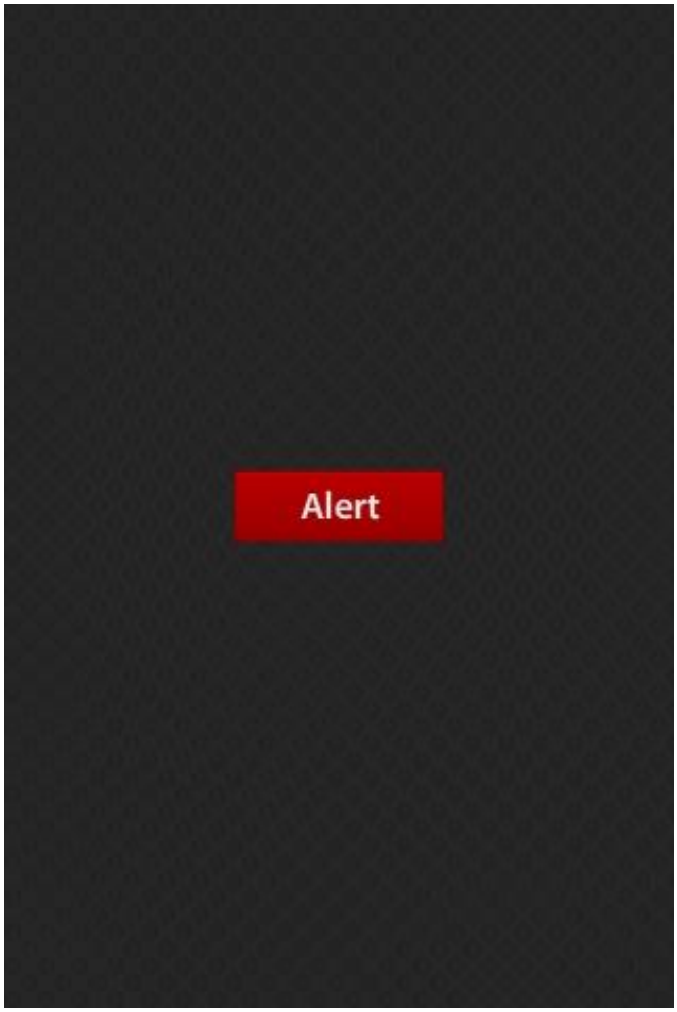
2011-1-ES1-LEO05-35968

- **iPad**: 1024x768px, 132 ppi
- **iPhone/iPodTouch**: 320x480px, 163 ppi
- **iPhone 4**: 960x640px, 326 ppi

Android, essendo una piattaforma open, funziona in maniera leggermente differente. Si possono infatti riscontrare delle differenze nella risoluzione dello schermo:

- **Nexus One**: 480x800px, 254 ppi
- **Droid**: 854x480px, 265 ppi
- **HTC Legend**: 320x480px, 180 ppi

Interfaccia



Codice

Nascondere la barra dello status

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Per prima cosa, nascondiamo la barra dello stato, cioè la barra in cima allo schermo che mostra il tempo, i segnali e altri indicatori.

```
display.setStatusBar(display.HiddenStatusBar)
```

Background

Questa stringa crea la variabile local background e usa il display API per aggiungere l'immagine specifica alla scena. Per impostazioni di default, l'immagine è aggiunta a 0,0 e sfrutta l'angolo superiore sinistro come punto di riferimento.

```
local background = display.newImage("background.png")
```

Pulsante Alert

Ripeti il processo con l'immagine del bottone, mettendola al centro dello schermo. La funzione del bottone sarà creata più avanti nel codice.

```
local alertButton = display.newImage("alertButton.png")

alertButton:setReferencePoint(display.CenterReferencePoint)
alertButton.x = 160
alertButton.y = 240
```

Alert Click

Quando l'utente clicca su qualunque pulsante delle opzioni nell'Alert viene mostrato un *clicked event*. Dobbiamo controllare nell'*index* dei bottoni cliccati per sapere quale opzione è stata selezionata. Un alert consente l'inserimento di un massimo di sei pulsanti, il suo indice è definito dall'ordine in cui è scritto nell'alert call.

```
local function onClick(e)
	if e.action == "clicked" then
		if e.index == 1 then
			-- //Some Action
			elseif e.index == 2 then
				system.openURL( "http://www.ebusiness-lab.gr" )
		end
	end
end
```

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Display Alert

Questa funzione verrà eseguita quando il pulsante Alert sarà premuto. Tale funzione usa il metodo *native.showAlert()* per mostrare l'alert. L'alert sarà connesso a una variabile che svolgerà la funzione di alert ID, in questo modo può essere localizzata, respinta o rimossa dal metodo *native.cancelAlert()*.

```
function alertButton:tap(e)
      local alert = native.showAlert("TEI Messolonghi", "Mobile Development at TEI Mesolonghi", {"OK", "Learn More"}, onClick)
end
```

Questo metodo ha quattro parametri, esaminiamoli:

- **titolo**: il testo in cima all'alert.
- **messagio**: il corpo dell'alert.
- **pulsanti**: una tabella che contiene i bottoni che saranno mostrati dagli alert, fino a un massimo di sei pulsanti.
- **registratore**: una funzione che registra gli eventi.

Pulsante di registrazione alert

La stringa seguente imposta il registratore:

```
alertButton:addEventListener("tap", alertButton)
```

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Figura 10: Simulazione di un messaggio di alert

6. CREARE UN SEMPLICE GIOCO DI BASKET CON CORONA

Il motore fisico che che presenta Corona è uno strumento incredibilmente potente e facile da usare. In questo tutorial esamineremo la creazione di un semplice gioco di basket usando questa fantastica tecnologia.

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

TO05-35968

Figura 11: Il progetto finale, visualizzato sul simulatore per l'iPhone

Step 1: Configurare il Motore Fisico

```
1. display.setStatusBar(display.HiddenStatusBar)
2.
3. local physics = require "physics"
4. physics.start()
5. physics.setGravity(9.81, 0)  -- 9.81 m/s*s in the positive x direction
6. physics.setScale(80)  -- 80 pixels per meter
7. physics.setDrawMode("normal")
```

La prima cosa che facciamo (come in molti programmi) è nascondere la barra di stato in cima allo schermo. Successivamente compiliamo la lista dei comandi necessari e memorizziamo il risultato nella variabile chiamata "physics". Nella stringa cinque impostiamo l'accelerazione di gravità. Solitamente la gravità è impostata con il valore di 9.8 m/s*s positivo nella direzione di y, ma in queso caso vogliamo che la gravità sia positiva nella direzione di x. Impostiamo, inoltre, la scala proporzionale a 80 pixel per metro. Questo numero può variare in base alle dimensioni degli oggetti nella tua figura, e tu devi fare delle prove per torvare le dimensioni più adatte. Io ho scelto 80 pixel al metro (px/m) perché voglio avere circa 15 feet di spazio in verticale sullo schermo. Una volta che hai capito di quanto spazio hai bisogno sullo schermo, con una semplice conversione puoi conoscere i pixel che ti sono necessari.

Nota: è importante provare a collegare gli oggetti riprodotti con quelli del mondo reale. Più le misure sono corrispondenti alla realtà, più la tua applicazione sembrerà realistica.

Alla fine di queste poche stringhe impostiamo su "normale" il draw mode. Questa stringa rende più facile, in seguito, passare alla modalità debug. Impostandola sul valore normale si possono disegnare le forme così come le vedrà l'utente nel gioco finale.

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

-ES1-LEO05-35968

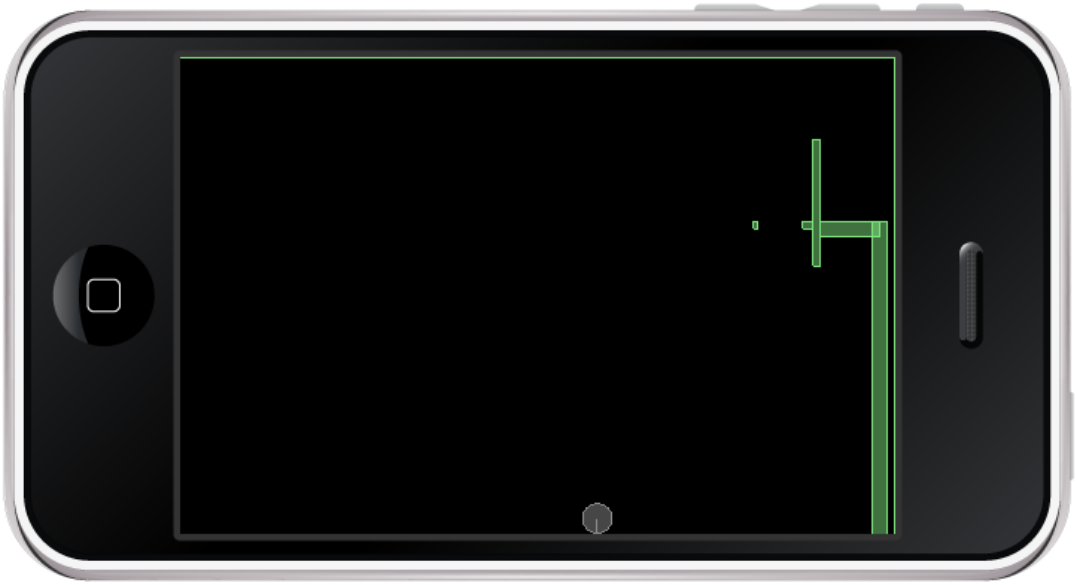


Figura 12: Il progetto finale visto nella modalità debug.

Step 2: Creare l'Arena

```
1. local background = display.newRect(0,0,display.contentWidth,display.contentHe
   ight)
2. local score = display.newText("Score: 0", 50, 300)
3. score:setTextColor(0, 0, 0)
4. score.rotation = -90
5. score.size = 36
6. local floor = display.newRect(320, 0, 1, 480)
7. local lWall = display.newRect(0, 480, 320, 1)
8. local rWall = display.newRect(0, -1, 320, 1)
9. local ceiling = display.newRect(-1, 0, 1, 480)
10.
11. staticMaterial = {density=2, friction=.3, bounce=.4}
12. physics.addBody(floor, "static", staticMaterial)
13. physics.addBody(lWall, "static", staticMaterial)
14. physics.addBody(rWall, "static", staticMaterial)
15. physics.addBody(ceiling, "static", staticMaterial)
```

Questo blocco stabilisce i confini dell'arena e le proprietà di tutti gli oggetti statici nell'applicazione. Aggiungiamo adesso un semplice rettangolo allo sfondo (bianco di default). Dentro al rettangolo sull'immagine di sfondo posizioniamo una casella di testo in cui indicare il punteggio. L'applicazione avrà un'inquadratura panoramica, pertanto è necessario fare degli aggiustamenti per il movimento di rotazione. La palla deve restare sempre nella porzione di schermo visibile, per fare ciò ci serviamo di quattro rettangoli statici (pavimento, parete di destra, parete di sinistra, soffitto) posizionati fuori dalla visuale.

In seguito inseriamo nuovamente i parametri nell'equazione, ma invece di digitare ancora la tabella delle proprietà fisiche per ogni oggetto, creiamo una tabella da usare per ogni muro e ogni obiettivo. Ho scelto valori standard per queste proprietà, sebbene consigli di variarli e provare differenti valori. C'è uno step in più che dobbiamo fare, cioè comunicare a Corona che questi oggetti devono essere considerati in condizioni fisiche realistiche e lo facciamo usando la funzione degli oggetti fisici addBody. Questa funzione riguarda:

1. L'oggetto
2. Un modificatore opzionale
3. Una tabella di proprietà fisiche

Lifelong Learning Programme
Education and Culture DG
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Abbiamo già determinato le proprietà e gli oggetti, così tutto ciò che rimane da impostare è il modificatore opzionale. Noi usiamo "static" per considerare la gravità e non qualsiasi altra forza al di fuori dalle nostre mura!

Figura 13: Lo sfondo bianco, il punteggio e le pareti invisibili

Step 3: Aggiungere una palla e un canestro

```
1.  -- Create the goal
2.  local vertPost = display.newRect(110, 5, 210, 10)
3.  vertPost:setFillColor(33, 33, 33)
4.  local horizPost = display.newRect(110, 10, 10, 40)
5.  horizPost:setFillColor(33, 33, 33)
6.  local backboard = display.newRect(55, 50, 85, 5)
7.  backboard:setFillColor(33, 33, 33)
8.
9.  physics.addBody(vertPost, "static", staticMaterial)
10. physics.addBody(horizPost, "static", staticMaterial)
11. physics.addBody(backboard, "static", staticMaterial)
12.
13. --Create the Ball
14. local ball = display.newCircle(50, 200, 10)
15. ball:setFillColor(192, 99, 55)
16.
17. physics.addBody(ball, {density=.8, friction=.3, bounce=.6, radius=10})
```

In un colpo solo, così facendo, creiamo i rimanenti elementi visuali dell'app. Ciò dovrebbe essere adesso abbastanza familiare. Ci sono solo due aspetti che vorrei sottolineare. Il primo è che alcuni dei valori per posizionare il canestro possono sembrare assenti. Questo è dovuto alla visione panoramica che abbiamo impostato, il canestro apparirà infatti solamente quando il dispositivo è orientato verso quel lato. Il secondo è di assicurarsi di includere nella tabella delle proprietà un raggio massimo in cui la palla può agire, in modo tale che si comporti verosimilmente.

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

Figura 14: Grafica con palla e canestro

Step 4: Creare un supporto per trascinare la palla

```
1.  local function drag( event )  -- create a function to execute when the ball
    is touched
2.      local myball = event.target   -- event.target is "who" is capturing the
    event
3.
4.      local phase = event.phase -- event.phase describes the touch sequence:
    "began", "moved", "canceled", etc...
5.      if "began" == phase then
6.          display.getCurrentStage():setFocus( myball )  -- by setting focus to
    the ball we instruct the system to deliver all future hit events to the same
    object, so that we can drag the ball around (the "moved" phase, below)
7.
8.          -- store initial position: we store the horizontal & vertical
    difference between the ball and the touch positions, so that we can drag the
    ball from anywhere on it's surface
9.          myball.x0 = event.x - myball.x
10.         myball.y0 = event.y - myball.y
11.
12.         -- avoid gravitational forces
13.         event.target.bodyType = "kinematic"
14.
15.         -- stop current motion, if any
16.         event.target:setLinearVelocity( 0, 0 )
17.         event.target.angularVelocity = 0
18.
19.     else
20.         if "moved" == phase then
21.             myball.x = event.x - myball.x0
22.             myball.y = event.y - myball.y0
23.         elseif "ended" == phase or "cancelled" == phase then
24.             display.getCurrentStage():setFocus( nil )  -- clear focus, user
    can touch any other objects after this
25.             event.target.bodyType = "dynamic"  -- re-enable gravity
26.         end
```


This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

2011-1-ES1-LEO05-35968

```
    end

    return true  -- when an event handler returns true, no other handlers get executed after this one
end
myball:addEventListener("touch", drag) -- make ball listen to the touch event and reply with the drag function
```

Questa funzione ci fornisce un supporto per trascinare molto semplice. Un parametro molto importante da impostare è il bodyType della palla in relazione alla cinematica, in modo tale che la gravità non possa togliere la palla dalle mani dell'utente (**Nota: assicurati di rimetterlo sul parametro dynamic una volta finito**). Le stringhe qui sotto sono altrettanto importanti. Qui viene fermato il movimento della palla quando è toccata per evitare lo stesso problema della gravità.

Se apri l'app così come è ora, probabilmente noterai che la palla perde tutto il movimento non appena la tocchi. Per evitare questo problema dobbiamo creare una funzione per tracciare la velocità della palla e, successivamente, impostare la velocità della palla in seguito a un impatto.

```
local speedX = 0
local speedY = 0
local prevTime = 0
local prevX = 0
local prevY = 0

function trackVelocity(event)
    local timePassed = event.time - prevTime  -- time is given in msec
    prevTime = prevTime + timePassed

    speedX = (myball.x - prevX)/(timePassed/1000)  -- velocity is counted at pixels/sec
    speedY = (myball.y - prevY)/(timePassed/1000)

    prevX = myball.x
    prevY = myball.y
end

Runtime:addEventListener("enterFrame", trackVelocity)  -- trackVelocity gets executed everytime the screen is redrawn
```

Creiamo trackVelocity come un registro dell'evento enterFrame. Il compito che svolge questa funzione è trovare il cambiamento della velocità nell'arco di tempo ed esprimere la velocità della palla in pixel al secondo. Aggiungi la seguente stringa alla funzione di trascinamento per impostare la velocità lineare della palla.

```
myball:setLinearVelocity(speedX, speedY) -- when myball is released it gets the computed velocity
```

Step 5: Creare il cerchio e il meccanismo del punteggio

Il codice seguente crea il cerchio del canestro. Nota che la parte centrale dell'anello non sarà una parte del sistema fisico perché noi vogliamo che la palla ci possa passare liberamente attraverso.

```
local rimBack = display.newRect(110, 55, 5, 7)
rimBack:setFillColor(207, 67, 4)
local rimFront = display.newRect(110, 92, 5, 3)
rimFront:setFillColor(207, 67, 4)
local rimMiddle = display.newRect(110, 62, 5, 30)
rimMiddle:setFillColor(207, 67, 4)

physics.addBody(rimBack, "static", staticMaterial)
physics.addBody(rimFront, "static", staticMaterial)  -- both the back and front of the rim should have a static body
```

Education and Culture DG
Lifelong Learning Programme
This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.
Programa de aprendizaje permanente
APEE Organismo Autónomo Programas Educativos Europeos

2011-1-ES1-LEO05-35968

Adesso abbiamo bisogno di un modo per sapere quando la palla passa attraverso il canestro. Il modo più semplice è creare una piccola patch dello schermo vicino all'anello come una "zona di punteggio". Ogni volta che la palla è nella zona possiamo incrementare il punteggio. Per evitare che il punteggio venga erratamente aumentato quando la palla si ferma sull'anello, dobbiamo tenere conto del tempo trascorso dall'ultimo canestro, e assicurare che ci sia una separazione temporale adeguata tra un canestro e l'altro. Un secondo è un lasso di tempo sufficiente.

```
1.  scoreCtr = 0
2.  local lastGoalTime = 1000
3.
4.  function monitorScore(event)
5.      if event.time - lastGoalTime > 1000  then  -- allow execution only after
    1 second
6.          if ball.x > 103 and ball.x < 117 and ball.y > 62 and ball.y < 92 then

7.              scoreCtr = scoreCtr + 1
8.              print(score.text)
9.              lastGoalTime = event.time
10.             score.text = "Score: " .. scoreCtr
11.         end
12.     end
13. end
14. Runtime:addEventListener("enterFrame", monitorScore)   -- scoring is monitored
    everytime the screen is redrawn
```

Conclusione

Corona si occupa dei compiti più difficili, lasciandoti più tempo per concentrarti sul contenuto e sul modo in cui giocare al videogame che sviluppi.

(Fonte: Carter Grove, November 2010, mobile tuts+ http://mobile.tutsplus.com/tutorials/corona/corona-sdk_game-development_basketball)

Basketball Game – Codice completo

Link al file

This project has been funded with support from the European Commission.
This material reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.