Relazione di tirocinio: Progettazione e sviluppo dell'integrazione della simulazione della fisica in un framework di sviluppo di applicazioni multimediali per dispositivi mobili.

università degli studi di padova
Facoltà di ingegneria
Dipartimento di Ingegneria dellInformazione
Corso di Laurea Magistrale in Ingegneria Informatica
Laureando: Maroso Alessandro
Professore relatore: Congiu Sergio
Tutore Aziendale: De Bello Nicola

5 dicembre 2012

Alla mia famiglia, per avermi sostenuto e permesso di giungere fino a questo traguardo.

A Giulia, che mi è sempre vicina.

Ad Alberto, Luca, Stefano ed Umberto, per questi cinque anni di studio passati a studiare assieme.

Alessandro Maroso

Indice

Capitolo 1

Introduzione

Da circa 10 anni, cioè dalla nascita degli smartphone, il settore del software per dispositivi mobili è stato in costante crescita. Lo sviluppo tecnologico ha permesso di ottenere dispositivi sempre più piccoli e sempre più in grado di offrire funzionalità paragonabili a quelle dei personal computer, tanto che nel primo quarto del 2012 sono state registrate più vendite di smartphone e tablet rispetto ai personal computer. Una delle cause principali di questa tendenza è il grande parco applicazioni messo a disposizione degli utenti e supportato dagli store ufficiali delle piattaforme. Lo sviluppo di applicazioni mobile è agevolato dalla maturità delle piattaforme principali, iOS e Android, ed è incentivato dalla facilità di raggiungere il mercato a livello globale, ma daltra parte è altamente concorrenziale. Per questo motivo è necessario puntare sugli elementi di innovazione e di originalità affinchè un prodotto abbia successo, per differenziarsi dalle molte alternative che popolano il mercato.

1.1 Contesto

Il contesto aziendale nel quale si è svolto il tirocinio è quello di WARES ME. WARES ME è una giovanissima azienda padovana con la inconsueta missione di coniugare tecnologia e arte. Due tra i soci fondatori (Nicola De Bello e Michele Morbiato) hanno quindici anni di comune esperienza imprenditoriale negli ambiti tecnologici, avendo fondato e gestito assieme varie aziende di successo, poi acquisite da gruppi sia nazionali che internazionali. Ad essi si aggiunge il contributo della scrittrice e creativa Silvia Sorrentino, il cui compito è appunto quello di immaginare le possibili applicazioni dei nuovi, innovativi strumenti e veicoli tecnologici ad afflato e contenuti artistici, e di Stefano Trainito che cura la direzione artistica dellimpresa.

WARES ME è attualmente focalizzata sulle frontiere della pubblicazione digitale, sulle opportunità di auto-pubblicazione e sulle innovative potenzialità artistiche offerte dalle piattaforme mobili Apple (e non solo). WARES ME, ad esempio, offre a editori, autori e illustratori:

- servizi di supporto nei processi di conversione/pubblicazione digitale e/o di auto-pubblicazione.

- la trasformazione di libri per bambini (già esistenti come libri convenzionali) in applicazioni per dispositivi portatili interattive, con un modello di business che prevede per editori/autori solo revenue-sharing, e nessun costo up-front.

- la creazione di App interattive a partire da materiale nuovo, concepito apposta per esistere sia come App che come libro convenzionale

e mira a spingere e promuovere la produzione di artisti attraverso il proprio sito.

La prima uscita di un libro per bambini trasformato in App interattiva (nonchè il primo caso di successo del modello di business di cui sopra) proviene da un titolo di Kite Edizioni (Chissà, una storia di Marinella Barigazzi illustrata da Ursula Bucher), prestigioso editore specializzato, ed è frutto della partnership WARES ME-Altera. Altera è una piccola e dinamica azienda di Padova che nasce nel 1999 per produrre applicazioni web su misura. Negli anni ha allargato le proprie competenze dal web allo sviluppo di applicazioni per piattaforme Apple Mac OS X e iOS (iPhone e iPad) realizzando progetti di notevole rilievo per importanti aziende a rilevanza nazionale e diventando una piccola "boutique di eccellenza" nel settore software. Il lavoro svolto durante il tirocinio, riguardando la parte più tecnica dellofferta di WARES ME, è stato svolto in stretto contatto con Altera che grazie alla grande esperienza nellambito delle piattaforme Apple Mac OS X e iOS ha saputo dettare le linee guida del progetto.

1.2 Obiettivi del tirocinio

Lattività di tirocinio si inserisce nellattività di WARES ME di trasformazione di libri per bambini (già esistenti come libri convenzionali) in applicazioni per dispositivi portatili interattive. Il progetto, chiamato APP-KID, si avvale di un framework software per la produzione rapida di libri per bambini in formato App.

Il framework software è basato sul framework Sparrow, una libreria basata su UIKit di Apple. Il framework UIKit fornisce le classi necessarie per costruire e gestire l'interfaccia utente di un'applicazione per iOS, il sistema operativo Apple utilizzato in tutti i dispositivi mobili dellazienda (iPod, iPhone, iPad). La libreria gestisce lapplicazione in un singolo oggetto e inoltre fornisce oggetti e metodi per la gestione degli eventi, la visualizzazione dellinterfaccia, la gestione delle finestre, la gestione delle delle visuali, i controlli dellapplicazione ed è specificamente progettata per una interfaccia touch screen.

Il framework Sparrow consente di creare applicazioni interattive per la piattaforma iOS in maniera semplificata rispetto ad UIKit, automatizzando vari aspetti dello sviluppo di applicazioni multimediali e quindi aumentando lefficienza nello sviluppo di tali applicazioni. L'obiettivo principale del framework è la creazione di giochi 2D, ma Sparrow può essere utilizzato per tutte le applicazioni grafiche e multimediali come nel caso di APP-KID che non tratta veri e propri giochi 2D bensì applicazioni multimediali con elementi dinamici.

Il progetto APP-KID permette di semplificare ulteriormente lo sviluppo delle applicazioni multimediali e di standardizzarle secondo un modello di applicazione comune, in modo da poter effettuare la trasposizione di un libro illustrato dalla versione cartacea a quella multimediale con il minimo sforzo, pur mantenendo la possibilità di apportare le modifiche e le personalizzazioni richieste

da ogni autore. Le applicazioni multimediali di APP-KID hanno quindi una struttura comune e condividono le seguenti funzioni di base:

- Schermata iniziale WARE'S ME
- Schermata iniziale Editore, Autore(i) e Titolo
- Schermata Menu con le seguenti voci:
  - Lettura
  - Indice
  - Gioca
  - Opzioni
  - Crediti
- Indice grafico delle pagine del libro con la possibilità di accedere direttamente alla singola pagina. Nellapplicazione standard il numero massimo di pagine, intese come singola illustrazione (che può essere anche a doppia facciata nel libro cartaceo), è sedici.
- Lettura pagina per pagina con audio o senza.

In ogni pagina sono presenti le seguenti funzioni con i relativi pulsanti:

- Avanti
- Indietro
- Menu
- Indice
- Funzioni audio
- Un'animazione primaria attivata automaticamente allo sfogliare della pagina, senza un relativo pulsante
- Tre azioni o animazioni interattive

Inoltre ogni libro è dotato di alcune funzioni comuni:

- Due semplici giochi, come puzzle, giochi di colorazione o scopri le differenze.
- Funzione di registrazione e riproduzione audio per lutente, integrata nelle singole pagine.
- Multilingua per un massimo di 3 lingue supportate: italiano, inglese, francese.

Sfruttando APP-KID come base è possibile creare anche applicazioni multimediali più complesse e che differiscono dalla struttura standard appena presentata.

In seguito alla realizzazione delle prime applicazioni, visto il buon successo riscontrato nel pubblico, si è pensato di ampliare e migliorare lofferta approfondendo la parte dedicata ai giochi, che pur essendo semplici da realizzare

possono risultare poco avvincenti se confrontati con ciò che popola il mercato attualmente, proprio per la loro semplicità.

Per potenziare lofferta ludica di APP-KID si è pensato di dotare il framework di una funzione particolare che al giorno doggi risulta essere uno degli ingredienti vincenti nei videogiochi per dispositivi mobili: la simulazione della fisica. La simulazione della fisica viene comunemente realizzata tramite un software detto motore fisico. Il motore fisico è svincolato dal resto dellapplicazione, e si occupa solamente di simulare la fisica degli oggetti per dare loro un credibile un movimento realistico. Anche se possono essere utilizzati per altre applicazioni, i motori fisici nascono principalmente come librerie per l'uso nei videogiochi, ed i videogiochi costituiscono la maggior parte dei software che utilizzano tali software.

Lobiettivo primario del tirocinio è stato quello di selezionare il motore fisico più adatto al progetto APP-KID tra le varie soluzioni disponibili, e di integrarlo nel framework di sviluppo nel modo più funzionale possibile.

Capitolo 2

Analisi delle esigenze

Il progetto di integrazione del motore fisico allinterno del framework di sviluppo di APP-KID deve considerare varie esigenze imposte soprattutto dal contesto aziendale nel quale è inserito, dal mercato al quale è rivolto e dalla piattaforma per la quale è sviluppato.

Innanzitutto è necessario che le funzioni della libreria siano sufficientemente efficienti da non appesantire lesecuzione dellapplicazione finale in ognuno dei dispositivi supportati. Il problema dellefficienza non è affatto trascurabile in quanto le risorse a disposizione dellapplicazione sono quelle di dispositivi portatili come iPod, iPhone e iPad, che essendo di dimensioni contenute ed essendo condizionati dalla capienza limitata delle batterie non possono essere dotati della stessa potenza computazionale dei comuni personal computer. Oltre ad essere efficiente in termini di potenza computazionale, la libreria deve essere dotata di una gestione della memoria coerente con i limiti imposti dalla piattaforma. Inoltre la corretta gestione della memoria, che eviti i cosiddetti memory leaks, è un requisito critico per lutilizzo della libreria in applicazioni destinate alla vendita tramite il negozio online di applicazioni per dispositivi portatili Apple, perchè ogni applicazione, prima di essere messa in vendita, viene controllata nei contenuti, nel corretto funzionamento e nellefficienza in modo che non degradi la user experience generale del dispositivo. Infatti unerrata gestione della memoria potrebbe causare malfunzionamenti sia nel sistema che nelle altre applicazioni in uso nello stesso dispositivo.

In secondo luogo è importante che la libreria si inserisca nel framework di sviluppo senza comportare modifiche allo stato delle librerie attualmente utilizzate in modo da essere retro-compatibile con le applicazioni già esistenti. Per questo motivo è importante che la libreria si progettata sulla base del framework attuale senza creare ambiguità o duplicare funzioni già esistenti ma adattandosi alla sua struttura in modo da risultare coerente agli occhi dello sviluppatore.

Un altro requisito importante è quello di semplificare il più possibile lutilizzo del motore fisico da parte dello sviluppatore senza però rinunciare a nessuna delle funzioni garantite dalla simulazione. Il problema principale riscontrato all'utilizzo di un motore fisico 2D senza una adeguata integrazione con il framework consiste nella grande quantità di configurazioni da effettuare nellambiente di sviluppo dovute anche allutilizzo di diversi linguaggi di programmazione: mentre la maggior parte dei motori fisici sono scritti in linguaggi di basso livello come C e C++, lintera stack di framework utilizzati in APP-KID è scritta in Objective-C.

Lutilizzo contemporaneo di due linguaggi differenti, seppur compatibili, in una stessa applicazione è considerata una pratica prona ad errori, soprattutto nel caso specifico in analisi in quanto i due linguaggi utilizzati sono caratterizzati da una gestione della memoria completamente diversa tra loro. La semplificazione dellutilizzo del motore fisico consiste anche nel rendere più rapido il processo di definizione delle molteplici variabili che è necessario inizializzare quando si utilizza il motore fisico. Pur lasciando la possibilità di configurare a piacimento ogni oggetto del simulatore della fisica si è scelto di automatizzare parte della configurazione con dei valori comuni. Infine è importante che lutilizzo della libreria sia quanto più simile allutilizzo del framework preesistente in modo da rendere laggiunta delle nuove funzioni semplice e coerente con quanto è già stato creato e da rendere il codice più leggibile e quindi facile da mantenere.

Il framework di APP-KID si appoggia a vari livelli nella stack dei framework offerti dallarchitettura software. Per la parte di menù e configurazioni il framework può utilizzare direttamente UIKit che fornisce vari oggetti utili alla creazione grafica dei menù in maniera nativa, mentre per la parte multimediale dellapplicazione, dalla grafica allaudio e alla gestione degli input, utilizza Sparrow.

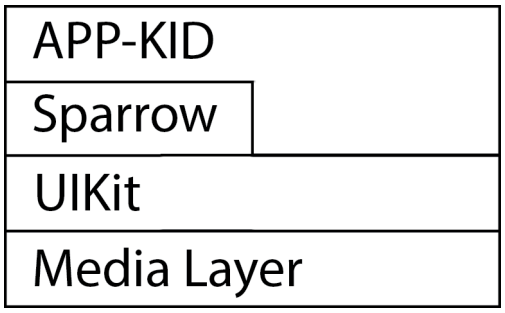
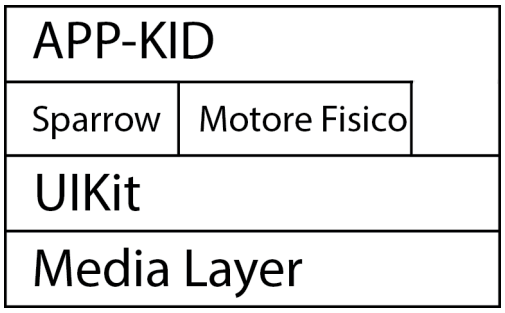


Figura 2.1:

Dopo aver analizzato le librerie sulle quali il progetto dovrà adattarsi è necessario scegliere il motore fisico da utilizzare. Una volta identificata la libreria più adatta a soddisfare le esigenze, è necessario capire come ed a che livello inserirla nella stack dei framework utilizzati, in modo che risulti coerente con il resto del progetto e soprattutto che non appesantisca il workflow di sviluppo delle applicazioni.

Sparrow è una libreria open source ed è semplice estenderla con nuove funzionalità, quindi risulta naturale integrare il motore fisico a questo livello dellarchitettura.

Figura 2.2:



Gli sviluppatori di Sparrow esortano la creazione di estensioni. Una delle priorità degli sviluppatori di Sparrow è quello di rimanere il più leggero e minimale possibile. Allo stesso tempo, non si desidera che gli utilizzatori di Sparrow ripetano operazioni comuni e sviluppino, da zero, elementi dell'architettura (o di uso comune) che già altri hanno sviluppato e testato. Per questo motivo sparrow supporta la creazione di estensioni che possono entrare a far parte del progetto open source. Per integrare bene una nuova estensione con il resto di Sparrow (e con altre estensioni), sono state identificate alcune linee guida da parte degli sviluppatori:

- Utilizzare prefissi personalizzati per i nomi delle classi. Se si pensa che la classe potrà essere integrata in una versione futura di sparrow, è bene evitare di usare il prefisso SP- utilizzato nelle classi del framework, perchè se venisse integrata ma se ne volesse modificare l'interfaccia violerebbe il codice di qualcuno che sta usando il proprio interno. Si consiglia il prefisso SX per le classi di estensione, ma si è liberi di utilizzare qualsiasi altro prefisso.

- Non cambiare il codice sorgente. UnEstensione non dovrebbe rendere necessario modificare il codice sorgente di Sparrow. Se fosse necessario modificare una classe Sparrow, sarebbe bene utilizzare le categorie di Objective-C. Le categorie di Objective-C forniscono un mezzo per aggiungere metodi a una classe. Tutti i metodi che si aggiungono in una categoria entrano a far parte della definizione della classe, quindi se si aggiunge un metodo alla classe qualsiasi istanza o sottoclasse avranno accesso a tale metodo. Inoltre permettono di sovrascrivere metodi già esistenti nella classe. A volte può risultare non sufficiente la possibilità di aggiungere o modificare qualche metodo. In questi casi, si può creare una mod del framework, ossia una versione alternativa di Sparrow che però non sarà totalmente compatibile con le applicazioni che utilizzano la versione ufficiale.

- Repository del codice sorgente. Se l'estensione è costituita da solo alcuni file, si consiglia di condividere semplicemente i files sorgenti tramite GIST. GIST è un servizio online che offre un metodo semplice e veloce per condividere piccole porzioni di codice che vengono gestite dal sistema in un repository git che automaticamente effettua un controllo della versione e rende il codice forkable. Un comodo archivio con il codice sorgente viene creato automaticamente dal sistema. Naturalmente, se l'estensione è troppo estesa, è possibile creare un vero e proprio repository e condividere i collegamenti ad esso.

- Creazione di una pagina Wiki. Quando lestensione è pronta, lo sviluppatore è invitato a creare una pagina contenente una descrizione dellestensione e le linee guida di utilizzo. Il sito web del framework Sparrow contiene uno spazio dedicato alle estensioni nel quale ognuno può aggiungere il proprio contributo.

Capitolo 3

Tecnologie utilizzate

3.1 iOS

iOS è il sistema operativo installato sulle famiglie di dispositivi mobili Apple iPhone, iPod touch, iPad. Il sistema operativo gestisce l'hardware del dispositivo e fornisce le tecnologie necessarie per realizzare applicazioni native. Il sistema operativo è dotato inoltre di diverse applicazioni di sistema come telefono, client di posta, web browser, che forniscono servizi di sistema standard per l'utente finale.

L'iOS Software Development Kit (SDK) contiene gli strumenti e le interfacce necessari per sviluppare, installare, eseguire, e testare le applicazioni native che appaiono sulla schermata iniziale di un dispositivo iOS. Le applicazioni native sono sviluppate utilizzando il framework del sistema iOS e il linguaggio Objective-C e sono eseguite direttamente su iOS, senza lutilizzo di virtualizzazioni o middleware tra lapplicazione e il sistema operativo. Diversamente dalle applicazioni web, le applicazioni native sono installate fisicamente su un dispositivo e sono quindi sempre a disposizione dell'utente, anche quando il dispositivo è completamente disconnesso sia dalla rete internet che dalla rete telefonica. Tali applicazioni risiedono accanto alle altre applicazioni di sistema e sia l'applicazione che tutti i dati utente sono sincronizzati al computer dell'utente tramite lapplicazione desktop iTunes.

Oltre alle applicazioni native, è possibile creare applicazioni web utilizzando una combinazione di HTML, fogli di stile CSS e di codice JavaScript. Le applicazioni web sono eseguite all'interno del browser web Safari e richiedono una connessione di rete internet per accedere al server web. Le applicazioni native, invece, vengono installate direttamente sul dispositivo e possono funzionare senza la presenza di una connessione di rete.

Il kit di sviluppo di iOS fornisce le risorse necessarie per sviluppare applicazioni native iOS, pertanto, la comprensione delle tecnologie e degli strumenti che compongono questo SDK può aiutare a fare scelte migliori in fase di progettazione e di implementazione delle applicazioni e dei framework che le supportano.

L'architettura del sistema operativo iOS si articola in più livelli sovrapposti. Al livello più basso, iOS agisce come intermediario tra l'hardware sottostante e le applicazioni che appaiono sullo schermo. Le applicazioni native non di siste-

ma è raro che comunichino direttamente con l'hardware del dispositivo, infatti esse comunicano con l'hardware attraverso un ben definito sistema di interfacce che proteggono le applicazioni dalle differenze di componenti che caratterizzano linsieme di dispositivi mobili per i quali sono sviluppate. Questa astrazione fa si che sia facile creare applicazioni che funzionino con in maniera uniforme e stabile su dispositivi con diverse capacità hardware.

Limplementazione delle tecnologie di iOS può anche essere vista come una serie di strati, che sono mostrati in figura 3.1. Ai livelli più bassi del sistema ci sono i servizi fondamentali e le tecnologie sulle quali si basano tutte le applicazioni mentre i livelli più alti contengono i servizi e le tecnologie più sofisticati.

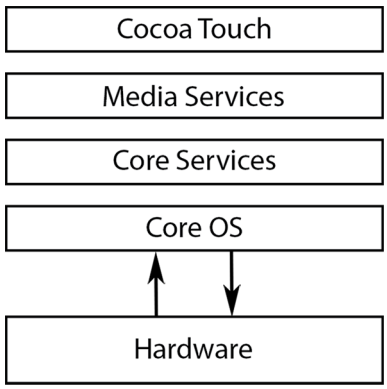


Figura 3.1:

Quando si sviluppa una applicazione, dove possibile, si deve prediligere l'uso di framework di alto livello piuttosto che dei servizi offerti dai livelli inferiori. Infatti i frameworks di livello superiore sono stati creati appositamente per fornire astrazioni orientate agli oggetti dei servizi e delle funzioni offerti dai costrutti dei livelli inferiori. Queste astrazioni, in genere, rendono molto più facile la scrittura di codice, perché riducono la quantità di righe di codice da scrivere e racchiudono caratteristiche potenzialmente complesse, come i socket per le connessioni e i thread. Sebbene forniscano una interfaccia per le tecnologie di livello inferiore, non si sovrappongono ad esse nascondendole al punto di vista dello sviluppatore: i frameworks di livello inferiore sono ancora disponibili per gli sviluppatori che preferiscono usarli o che vogliono utilizzare gli aspetti di tali strutture che non sono esposti dagli strati superiori.

Apple espone la maggior parte delle sue interfacce di sistema in pacchetti speciali chiamati framework. Un framework è una directory che contiene una libreria dinamica condivisa e le risorse (ad esempio, file di intestazione, immagini, applicazioni di supporto, ecc.) necessarie per a supporto di tale libreria. Per utilizzare i frameworks è sufficiente collegarli al progetto dellapplicazione proprio come si farebbe con qualsiasi altra libreria condivisa. Collegare un framework al progetto permette di accedere alle funzioni che espone e permette agli strumenti di sviluppo sapere dove trovare i file di intestazione e le risorse contenute nel framework.

I layers con i quali si interfacciano Sparrow e APP-KID sono principalmente i due più in alto nella stack, in quanto offrono completamente i servizi necessari a sviluppare una applicazione multimediale nativa.

3.1.1 Modello MVC

Una qualsiasi applicazione per iOS che utilizza gli strumenti messi a disposizione dai vari framework, se progettata correttamente, è basata su un modello detto Model View Controller (MVC). MVC è un vero e proprio pattern architetturale creato negli anni 80 per consentire la presentazione multipla (in diverse forme) di un oggetto in varie interfacce grafiche senza dover modificare o adattare loggetto da presentare. Infatti MVC è una applicazione del pattern Observer alle interfacce utente che non solo realizza la separazione tra dati e interfaccia ma svincola anche la logica di controllo dellapplicazione dai dati.

Il modello è basato sulla separazione dei compiti fra tre componenti software che interpretano tre ruoli principali:

- Model: gli oggetti che costituiscono il modello rappresentano particolari conoscenze e competenze, contengono i dati di un'applicazione e definiscono la logica che regola la manipolazione dei dati. Un'applicazione MVC ben progettata ha tutti i dati importanti incapsulati in oggetti del modello. Tutti i dati che fanno parte dello stato persistente dellapplicazione devono risiedere negli oggetti del modello una volta che i dati vengono caricati nell'applicazione. Idealmente, un oggetto del modello non ha alcun collegamento esplicito con l'interfaccia utente utilizzata per presentarlo e modificarlo. Tuttavia nella pratica la separazione dallinterfaccia non è sempre la cosa migliore, infatti cè un certo margine di flessibilità nel modello realizzato in iOS, ma in generale un oggetto del modello non dovrebbe occuparsi di come è realizzata linterfaccia.

- View: Un oggetto di tipo vista è in grado di visualizzare, e in alcuni casi di modificare, i dati del modello dell'applicazione. La view non dovrebbe essere responsabile per la memorizzazione dei dati che presenta, ma può memorizzare nella cache alcuni dati per motivi di prestazioni. Un oggetto della vista può essere responsabile della visualizzazione di solo una parte di un oggetto del modello, di un oggetto intero del modello o anche di molti oggetti del modello differenti. Le view sono disponibili in molte varietà diverse e tendono ad essere riutilizzabili e configurabili fornendo coerenza tra applicazioni diverse nello stesso sistema. Infatti il framework UIKit definisce un gran numero di oggetti di visualizzazione che possono essere riutilizzati assicurando lo stesso funzionamento in diverse applicazioni, garantendo un elevato livello di coerenza per aspetto e comportamento tra le applicazioni. Un oggetto di visualizzazione deve assicurarsi che i dati del modello siano visualizzati correttamente, di conseguenza ha bisogno di conoscere le modifiche apportate al modello.

- Controller: Poiché gli oggetti del modello non devono essere legati ad interfacce specifiche, hanno bisogno di un modo generico per segnalare il cambiamento. Un oggetto di controllo funge da intermediario tra gli oggetti della view dell'applicazione e gli oggetti del modello. I controller hanno spesso il compito di fare in modo che la view abbia accesso agli oggetti del modello che devono presentare e quello di agire come canale attraverso il quale linterfaccia sia aggiorna secondo le modifiche dei dati. Gli oggetti del controller possono anche configurare e coordinare le attività di una applicazione e gestire i cicli di vita di altri oggetti. In un tipico modello MVC, quando gli utenti immettono un valore o indicano una scelta

attraverso un oggetto di visualizzazione, il valore o la scelta viene comunicato a un oggetto di controllo. L'oggetto di controllo può interpretare l'input dell'utente in alcune applicazioni specifiche o può indicare ad un oggetto del modello cosa fare con questo ingresso. Sulla base dello stesso input dell'utente alcuni oggetti del controller potrebbero anche indicare ad un oggetto di visualizzazione di modificare il proprio aspetto o il proprio comportamento. Al contrario, quando un oggetto del modello cambia, il modello comunica il cambiamento al controller, che penserà ad aggiornare di conseguenza gli oggetti di visualizzazione.

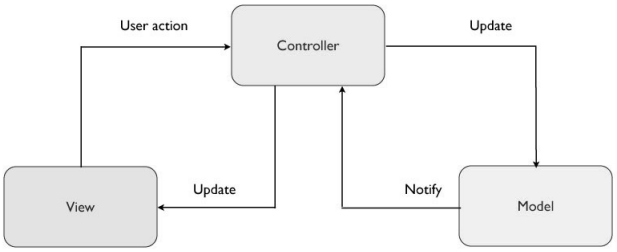


Figura 3.2:

Il modello MVC presenta alcuni aspetti positivi e negativi per lo sviluppo software. Le applicazioni progettate secondo questo modello godono di una separazione tra dati ed interfaccia molto solida essendo realizzata a livello progettuale, inoltre la separazione della logica dellapplicazione nel controllore garantisce una facile espandibilità e mantenibilità dellapplicazione. Il modello risulta vantaggioso anche nel caso in cui si sviluppino più applicazioni con elementi comuni perchè la separazione dei compiti in moduli di tre tipi permette una grande riusabilità delle classi. Daltra parte, il modello risulta svantaggioso nel caso in cui non ci sia la necessità di riutilizzare parti del programma, infatti loverhead dovuto alla progettazione piuttosto complessa e alla necessità di creare classi aggiuntive rispetto a quelle richieste da modelli più semplici può essere molto grande.

3.1.2 Media Layer

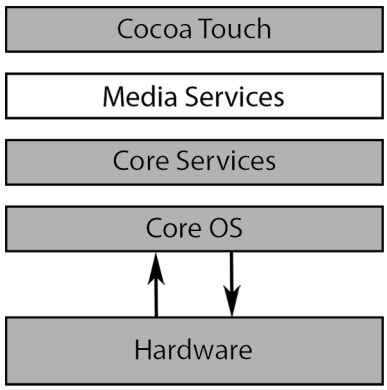


Figura 3.3:

Lo strato Media contiene le tecnologie di grafica, audio e video volte a creare la migliore esperienza multimediale possibile su un dispositivo mobile. Le tecnologie di questo livello sono state progettate per rendere più facile la creazione e la gestione degli elementi multimediali allinterno di una applicazione nativa.

3.1.2.1 Grafica

La grafica di alta qualità è una parte importante di tutte le applicazioni iOS. Il più semplice (e più efficace) modo di creare un'applicazione è quello di utilizzare le immagini prerenderizzate insieme agli elementi standard e ai controlli del framework UIKit e lasciare che il sistema si occupi di renderizzare linterfaccia. Tuttavia possono esserci situazioni in cui è necessario andare al di là della semplice grafica adatta ad applicazioni standard. In tali situazioni, è possibile utilizzare le seguenti tecnologie per la gestione dei contenuti grafici dellapplicazione: Il framework Core Graphics (noto anche come Quartz) gestisce il rendering nativo vettoriale 2D e basato sulle immagini attraverso le API di disegno 2D Quartz. Quartz è lo stesso motore di disegno vettoriale che viene utilizzato in OS X. Esso fornisce il supporto per di disegno basato su percorso, rendering con anti-aliasing, gradienti, immagini, colori, trasformazioni tra coordinate spaziali, e la creazione, visualizzazione, e il parsing di documenti PDF. Anche se l'API è basata su C, utilizza astrazioni basate sugli oggetti per rappresentare oggetti di disegno fondamentali, il che rende facile memorizzare e riutilizzare il contenuto grafico.

Il framework Core Animation fornisce supporto avanzato per animare elementi, immagini ed altri contenuti. Core Animation è contenuto nel QuartzCore framework e fornisce un alto livello di interfaccia Objective-C per la configurazione le animazioni e gli effetti che vengono poi renderizzati in hardware per migliorare le prestazioni. Core Animation è integrato in molte parti di iOS, comprese le classi UIKit come UIView, offrendo animazioni per molti comportamenti di sistema standard. È inoltre possibile utilizzare l'interfaccia Objective-C in questo contesto per creare animazioni personalizzate, infatti si possono creare interfacce utente dinamiche per le applicazioni senza dover utilizzare API grafiche di basso livello come OpenGL per ottenere animazioni senza un degrado delle prestazioni. Per semplificare il processo di creazione delle animazioni Core Animation mette a disposizione una serie di funzioni:

- Compositing ad alte prestazioni con un semplice e accessibile modello di programmazione.
- Un comodo livello di astrazione che permette di creare interfacce utente complesse con una gerarchia di oggetti a livelli.
- Una struttura di dati leggera. È possibile visualizzare e animare centinaia di strati contemporaneamente.
- Un'interfaccia di animazione astratta che permette di eseguire animazioni su un thread separato, indipendente dal ciclo di esecuzione dell'applicazione. Una volta che l'animazione è configurata e inizia, Core Animation si assume la piena responsabilità per sincronizzarla al frame rate.
- Miglioramento delle prestazioni delle applicazioni. Le applicazioni devono ridisegnare linterfaccia solo quando cambia il contenuto, infatti è richiesta

una interazione minima con lapplicazione per il ridimensionamento e la fornitura di servizi di layout.

- Un gestore del modello di layout flessibile e anche un gestore che permette il posizionamento e le dimensioni di un elemento a seconda del posizionamento degli elementi adiacenti di pari livello.

Il framework Core Image fornisce un supporto avanzato per la gestione di video e immagini fisse. È possibile utilizzare i filtri incorporati per tutto, dalle semplici operazioni (come ritoccare e correggere le foto) alle operazioni più avanzate (come il riconoscimento facciale e delle features). Il vantaggio di utilizzare questi filtri è che operano in modo non distruttivo in modo che le immagini originali non vengono mai modificate direttamente. Inoltre, Core Image si avvale della CPU disponibile e della potenza di elaborazione della GPU per garantire che le operazioni siano veloci ed efficienti. La classe CIImage fornisce l'accesso a un set standard di filtri che è possibile utilizzare per migliorare la qualità di una fotografia. Per creare altri tipi di filtri, è possibile creare e configurare un oggetto CIFilter per il tipo di filtro. In sintesi, le operazioni principali che vengono eseguite con il framework Core Image sono:

- Processare le immagini utilizzando filtri di immagine esistenti.
- Concatenare una serie di filtri e poi archiviarli per un uso successivo.
- Rilevare feature(come volti e occhi) in immagini fisse tracciare i volti nei filmati.
- Analizzare le immagini per ottenere un insieme di filtri di regolazione automatica.

I frameworks OpenGL ES e GLKit forniscono il supporto per il rendering 2D e 3D utilizzando interfacce di accelerazione hardware.

L'Open Graphics Library (OpenGL) è utilizzata per visualizzare i dati 2D e 3D. Si tratta di una libreria open-standard grafica multiuso dedicata alla creazione di contenuti digitali 2D e 3D, alla progettazione meccanica e architettonica, alla prototipazione virtuale, alla simulazione di volo, ai videogiochi e ad altro ancora. OpenGL permette agli sviluppatori di applicazioni di configurare una pipeline grafica 3D e inviare i dati ad essa. I vertici sono trasformati e illuminati, assemblati in primitive, e rasterizzati per creare un'immagine 2D. OpenGL è stata progettata per convertire le chiamate di funzione in comandi grafici che possono essere inviati all'hardware grafico sottostante. Poiché questo hardware sottostante è dedicato alla elaborazione di comandi di grafica e disegno OpenGL è in genere molto veloce.

OpenGL per sistemi embedded (OpenGL ES) è una versione semplificata di OpenGL che elimina le funzionalità ridondanti per fornire una libreria che sia più facile da imparare e più facile da implementare nellhardware grafico del quale sono forniti i comuni dispositivi mobili.

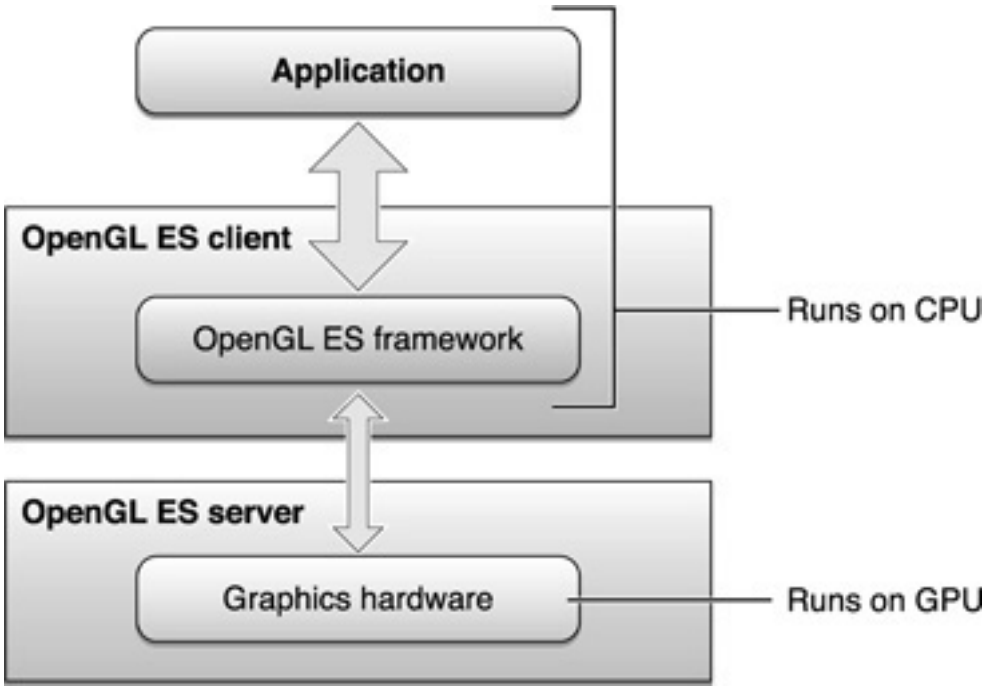


Figura 3.4:

Il framework GLkit contiene una serie di classi di utilità basate su Objective-C che semplificano il lavoro necessario per creare una applicazione OpenGL ES 2.0. GLKit fornisce il supporto per quattro aree chiave dello sviluppo di applicazioni con elementi di OpenGL ES: Il GLKView e le classi GLKViewController forniscono un'implementazione standard di una finestra OpenGL-ES e i loop di rendering ad essa associato. La finestra gestisce l'oggetto framebuffer sottostante che a sua volta gestisce la memoria buffer della scheda video nella quale vengono memorizzate le informazioni destinate all'output per la rappresentazione di un intero fotogramma per conto dellapplicazione; l'applicazione deve solo utilizzarlo, senza curarsi della renderizzazione.

La classe GLKTextureLoader mette a disposizione le routine per la conversione delle immagini e le routine di caricamento per l'applicazione, che permette di caricare automaticamente le texture e le immagini nel contesto dellapplicazione. E 'possibile caricare le texture in modo sincrono o asincrono. Quando si carica la texture in modo asincrono, l'applicazione fornisce un blocco del gestore di completamento da chiamare quando la texture viene completamente caricata nel contesto dellapplicazione.

Il framework GLKit fornisce implementazioni di vettori, matrici, e quaternioni così come operazioni su stack di matrici che coincidono con le funzionalità trovate in OpenGL ES 1.1.

Le classi GLKBaseEffect, GLKSkyboxEffect e GLKReflectionMapEffect forniscono shader grafici configurabili che implementano le operazioni grafiche di uso più comune. In particolare, la classe GLKBaseEffect implementa il modello di illuminazione e di materiale trovato nelle specifiche di OpenGL ES 1.1, semplificando lo sforzo richiesto per eseguire la migrazione di un'applicazione da OpenGL ES 1.1 a OpenGL ES 2.0.

Il framework Core Text fornisce un layout di testo e il relativo motore di rendering. Il motore di layout di Core Text è stato progettato appositamente per effetuare semplici operazioni di disposizione di testo con semplicità e per evitare effetti collaterali. Linterfaccia di programmazione dei font di Core Text è complementare al motore di layout di Core Text ed è progettata per gestire i font Unicode nativamente, unificando diverse strutture font di OS X in un'unica interfaccia di programmazione completa.

Il framework Image I/O fornisce interfacce di lettura e scrittura per la maggior parte dei formati di immagine. Originariamente parte del framework Core Graphics, Image I/O costituisce ora una framework indipendente per consentire agli sviluppatori di utilizzarlo in modo indipendente da Core Graphics (Quartz 2D). Image I/O fornisce il modo definitivo per accedere ai dati di immagine essendo altamente efficiente, consente un facile accesso ai metadati, e fornisce la gestione del colore.

Il framework Assets Library consente di accedere alle foto e video nella libreria fotografica dell'utente. La Assets Library fornisce un sistema basato su query per il recupero di foto e video dal dispositivo dell'utente. Con questo framework è possibile accedere alle stesse attività che normalmente vengono gestite dallapplicazione Photos, inclusi gli elementi degli album fotografici salvati dallutente e gli eventuali video e foto che sono stati importati sul dispositivo. È inoltre possibile salvare nuove foto e video torna all'album salvato la foto dell'utente.

3.1.2.2 Audio

Le tecnologie audio disponibili in iOS sono state progettate per aiutare a fornire una ricca esperienza audio per gli utenti. Questa esperienza include la possibilità di riprodurre audio di alta qualità, la registrazione audio di alta qualità, e di attivare la funzione di vibrazione sui dispositivi che ne sono dotati.

Il sistema fornisce diversi modi per riprodurre e registrare contenuti audio. I framework nella seguente lista sono ordinati dal livello più alto al più basso, con il framework Media Player che offre le interfacce più alte che è possibile utilizzare. Quando si sceglie quale framework audio utilizzare, è importante ricordare che i framework di livello più alto sono più facili da usare mentre quelli di livello inferiore offrono maggiore flessibilità e controllo, ma richiedono più lavoro.

Il framework Media Player offre un facile accesso alla libreria musicale iTunes dellutente e il supporto per la riproduzione di brani e playlist. La libreria è provvista di interfacce di accesso al sistema iPod che permette all'applicazione di riprodurre i brani di contenuti nel dispositivo di un utente, accedere agli audiolibri e ai podcast audio. La struttura della libreria rende la riproduzione di base molto semplice ma permette anche di sostenere la ricerca avanzata e il controllo della riproduzione. Lutilizzo della libreria iPod permette alle applicazioni iOS di avvalersi di una vasta gamma di miglioramenti relativi alla musica contenuta nel dispositivo.

Il framework AVFoundation fornisce un insieme di interfacce Objective-C di facile utilizzo per la gestione della riproduzione e della registrazione di audio in alta qualità. I servizi offerti da questo framework includono anche:

- Gestione degli asset multimediali

- Ritocco e montaggio dei contenuti multimediali
- Registrazione e riproduzione di filmati
- Gestione della traccia in riproduzione
- Gestione dei metadati per gli elementi multimediali
- Panning stereofonico
- Sincronizzazione precisa tra i suoni
- Un interfaccia Objective-C per determinare i dettagli di file audio, come ad esempio il formato dei dati, frequenza di campionamento e il numero di canali.

Da iOS 5, il framework AVFoundationinclude il supporto per lo streaming di contenuti audio e video su AirPlay utilizzando la classe AVPlayer. Il supporto per AirPlay è abilitato di default, ma le applicazioni possono rinunciarvi, se necessario. Il framework è una fonte unica per la registrazione e la riproduzione di audio e video in iOS e fornisce anche il supporto molto più sofisticato per la gestione e la gestione di elementi multimediali rispetto ai frameworks di livello superiore.

OpenAL fornisce una serie di interfacce cross-platform per la realizzazione di audio posizionale. Può essere utilizzato per implementare un sistema di audio posizionale ad alte prestazioni e ad alta qualità audio in giochi e altri programmi che lo richiedono. Inoltre, essendo OpenAL uno standard cross-platform, il codice scritto utilizando moduli OpenAL su iOS può essere portato su molte altre piattaforme con il minimo sforzo.

Il framework Core Audio offre interfacce per la riproduzione e la registrazione di contenuti audio sia in maniera semplice che con funzioni più sofisticate . È possibile utilizzare queste interfacce per la riproduzione di suoni di avviso di sistema, attivare la capacità di vibrazione di un dispositivo, gestire il buffering e la riproduzione di contenuti audio multicanale locali o in streaming. Il pacchetto è costituito da cinque framework:

- Il framework Core Audio (da non confondere con il framework Core Audio che lo contiene) dichiara i tipi di dati e le costanti utilizzate da altre interfacce Core Audio e comprende anche una serie di funzioni di convenienza.
- Il framework AudioToolbox fornisce interfacce per la registrazione, la riproduzione e il parsing stream audio e interfacce per la gestione di sessioni audio. Questo framework fornisce anche il supporto per la gestione dei file audio, riproduzione di suoni di avviso del sistema, e l'attivazione della capacità di vibrazione su alcuni dispositivi.
- Il framework AudioUnit fornisce servizi per l'utilizzo delle unità audio pre-costruite, che sono moduli di elaborazione audio a disposizione degli sviluppatori. Audio Unit fornisce plug-in audio che supportano l'elaborazione, la miscelazione, l'equalizzazione, la conversione di formato, e la gestione in tempo reale di input e output. È possibile caricare e utilizzare in modo dinamico questi potenti e flessibili plug-in, chiamati appunto unità audio, dall'applicazione nativa che si sta sviluppando. Le unità audio

solitamente svolgono le proprie funzioni nel contesto di un oggetto contenitore chiamato grafico di elaborazione audio, come mostrato in figura 3.5. Nellesempio, lapplicazione invia audio alle prime unità audio nel grafico per mezzo di una o più funzioni di callback ed esercita un controllo individuale su ogni unità audio. L'uscita dellunità I/O, l'ultima unità audio in questo e in qualsiasi grafico di elaborazione audio, collega direttamente all'uscita hardware.

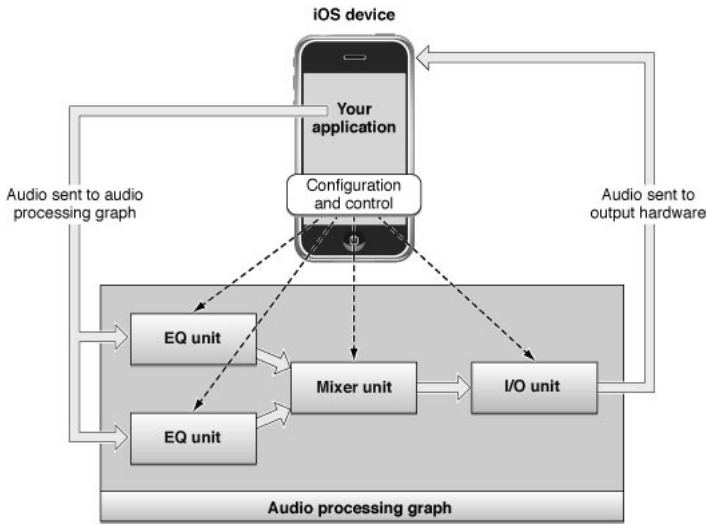


Figura 3.5:

Poiché le unità audio costituiscono lo strato più basso di programmazione dello stack audio iOS, per usarle in modo efficace è necessaria una comprensione più profonda di quella che serve per le altre tecnologie audio di iOS. A meno che non si richieda la riproduzione in tempo reale di suoni sintetizzati, a bassa latenza di input e output, o unità audio con funzioni specifiche, sarebbe più semplice utilizzare i frameworks multimediali di più alto livello come Media Player, AV Foundation, OpenAL o Audio Toolbox. Queste tecnologie, caratterizzate da un più alto livello di astrazione, utilizzano internamente le unità audio e sono dotate di importanti funzionalità aggiuntive. I due più grandi vantaggi di utilizzare direttamente le unità audio sono:

- Risposta eccellente. Avendo accesso a un thread dedicato con priorità in tempo reale attraverso una funzione di callback che renderizza una unità audio funzione di callback render unità, il codice audio è il più vicino possibile alla risposta di un dispositivo analogico. Attività come luso di strumenti musicali digitalizzati in tempo reale e la gestione di input e output simultanei della voce traggono i maggiori benefici dall'uso diretto di unità audio.

- Riconfigurazione dinamica. L'API del grafico di elaborazione audio, costruita intorno alla classe AUGraph, consente di assemblare, riconfigurare, e riorganizzare dinamicamente le catene complesse di elaborazione audio in maniera thread-safe, il tutto durante l'elaborazione audio.

Questa è l'unica API audio in iOS offrire questa funzionalità. Il ciclo di vita di una unità audio procede come segue:

- In fase di esecuzione, si ottiene un riferimento alla libreria che definisce lunità audio che si desidera utilizzare.
- Si crea un'istanza dell'unità audio.
- Si configura l'unità audio come richiesto per il tipo specifico di unità e per accomodare l'intento dellapplicazione.
- Inizializzare l'unità audio per prepararla a gestire l'audio.
- Avviare il flusso audio.
- Controllare l'unità audio.
- Al termine, rilasciare l'unità audio liberando la memoria.

Le unità audio forniscono caratteristiche individuali di grande utilità come lo stereo panning, miscelazione, controllo del volume, e la misurazione del livello audio. Inserire le unità audio nella propria applicazione consente di aggiungere tali funzionalità ma per ottenere tali risultati è necessario avere ben chiari una serie di concetti fondamentali, tra cui i formati di flussi audio, lutilizzo di funzioni di callback, e l'architettura dell'unità audio.

Loggetto che gestisce le varie unità audio è chiamato grafico di elaborazione audio ed è definito nella classe AUGraph. I metodi di AUGraph forniscono interfacce per la rappresentazione di un insieme di unità audio, i relativi collegamenti tra gli ingressi e le uscite e callback utilizzati per fornire gli input. La classe consente inoltre l'incorporamento di grafici figli per permettere una organizzazione logica e strutturata della catena di segnale. Un oggetto di un grafico di elaborazione audio può essere analizzato al suo interno per ottenere informazioni complete su tutte le unità audio che contiene. I vari oggetti, nodi (ognuno di tipo AUNode) del grafico, rappresentano ciascuno un apparecchio audio o un grafico figlio e possono essere aggiunti o rimossi, e le loro interazioni modificate. Lo stato di un oggetto grafico può essere manipolato sia nel thread di rendering che in altri thread. Di conseguenza, qualsiasi attività che influenzi lo stato del grafico viene gestita con dei lock e con un sistema di messaggiì tra qualsiasi thread chiamante e il thread su cui viene chiamata lunità audio di uscita dell'oggetto grafico (il thread di rendering). Un oggetto grafico avrà un singolo nodo testa ossia una unità di output. L'unità di output viene utilizzata sia per avviare che fermare le operazioni di rendering di un grafico, ed è il punto di controllo per la manipolazione sicura dello stato del grafico mentre è in funzione.

Il framework CoreMIDI fornisce servizi di MIDI di basso livello, ossia fornisce un metodo standard per comunicare con i dispositivi MIDI hardware, incluse le tastiere e sintetizzatori. È possibile utilizzare questo framework per inviare e ricevere messaggi MIDI e per interagire con le periferiche MIDI collegate a un dispositivo basato su iOS tramite il connettore dock o la rete.

Quando si creano i contenuti multimediali da inserire in una applicazione, è importante tenere conto di quali sono i formati multimediali supportati dalla piattaforma sulla quale sarà installata lapplicazione. Nel caso di iOS le tecnologie audio delle quali è dotato il sistema operativo supportano i seguenti formati audio:

- AAC
- Apple Lossless (ALAC)
- A-law
- IMA / ADPCM (IMA4)
- PCM lineare
- -law
- DVI / Intel ADPCM IMA
- Microsoft GSM 6.10
- AES3-2003

3.1.2.3 Video

iOS fornisce diverse tecnologie per riprodurre i contenuti video. Sui dispositivi con l'hardware appropriato, è anche possibile utilizzare queste tecnologie per registrare video e incorporarli nellapplicazione. Il sistema fornisce diversi metodi per riprodurre e registrare contenuti video che si possono scegliere a seconda delle esigenze. Quando si sceglie una tecnologia video, è bene ricordare che i frameworks di livello superiore possono semplificare notevolmente il lavoro da fare per supportare le più comuni funzionalità di cui si ha bisogno e sono in questi casi preferibili. I frameworks nel seguente elenco sono ordinati dal livello più alto al livello più basso, con il quadro Media Player che offre il livello di astrazione più alto che è possibile utilizzare.

La classe UIImagePickerController in UIKit fornisce un'interfaccia standard per la registrazione video su dispositivi con una fotocamera supportata. Un controller di tipo Image Picker gestisce le interazioni degli utenti e fornisce i risultati di tali interazioni a un oggetto delegato. Il ruolo e l'aspetto di un controller Image Picker dipendono dal tipo di sorgente che si assegna ad esso prima di visualizzarlo:

- Un tipo UIImagePickerControllerSourceTypeCamera fornisce un'interfaccia utente per scattare una nuova foto o un filmato.
- Un tipo di UIImagePickerControllerSourceTypePhotoLibrary o UIImagePickerControllerSourceTypeSavedPhotosAlbum fornisce un'interfaccia utente per scegliere tra le immagini salvate e filmati.

Per utilizzare un controller Image Picker contenente i controlli predefiniti, si può procedere secondo una procedura standard:

- Verificare che il dispositivo sia in grado di ricevere il contenuto dalla sorgente desiderata. È possibile fare ciò chiamando il metodo di classe isSourceTypeAvailable fornendo una costante che faccia parte dell'enumerazione "UIImagePickerControllerSourceType".

- Verificare quali tipi di formati multimediali sono disponibili per il tipo di sorgente che si sta utilizzando, chiamando il metodo availableMediaTypesForSourceType. Questo permette di distinguere tra un controller che può essere utilizzato per la registrazione video e uno che può essere utilizzato solo per le immagini fisse.

- Impostare il controller in modo che regoli linterfaccia utente in base ai tipi di supporto che si desidera rendere disponibili (immagini fisse, filmati o entrambi) impostando la proprietà mediaTypes.

- Presentare l'interfaccia utente. A seconda della piattaforma iPhone e iPod touch o iPad è possibile utilizzare la modalità a schermo intero oppure pop-up. Sui dispositivi più piccoli è sufficiente impostare come nuovo controller il controller Image Picker appena configurato, utilizzando il metodo presentViewController:. Sui dispositivi più grandi come gli iPad il modo corretto di presentare il controller dipende dal tipo di sorgente che si sta utilizzando. Se la sorgente è la fotocamera è necessario usare la modalità a schermo intero mentre se la sorgente è la galleria è necessario usare la modalità pop-up.

- Quando l'utente tocca un pulsante per selezionare l'immagine appena acquisita o per salvare un filmato o se annulla l'operazione, è necessario chiudere lImage Picker tramite lapposito oggetto delegato. I contenuti appena acquisiti, come foto e video, possono essere salvati dalloggetto delegato nella memoria del dispositivo.

Il framework Media Player fornisce una serie di interfacce semplici da usare per la riproduzione di filmati a schermo intero o in finestre.

L'AV Foundation framework fornisce un insieme di interfacce Objective-C per la gestione della registrazione e riproduzione di filmati.

Core Media descrive i tipi di dati di basso livello utilizzati dalle strutture di livello superiore e fornisce interfacce di basso livello per la manipolazione dei contenuti multimediali.

iOS supporta la riproduzione di file video con le estensioni mov, mp4, m4v, e .3gp e utilizzando i seguenti standard di compressione:

- Video H.264, fino a 1,5 Mbps, 640 x 480 pixel, 30 fotogrammi al secondo, Low-Complexity versione del Baseline Profile H.264 con audio AAC-LC fino a 160 Kbps, 48kHz, audio stereo nei formati. M4v, . mp4 e mov.

- Video H.264, fino a 768 Kbps, 320 x 240 pixel, 30 fotogrammi al secondo, Baseline Profile fino al livello 1.3 con audio AAC-LC fino a 160 Kbps, 48 kHz, audio stereo nei formati. M4v,. Mp4, e. mov

- Video MPEG-4, fino a 2,5 Mbps, 640 x 480 pixel, 30 fotogrammi al secondo, Simple Profile con audio AAC-LC fino a 160 Kbps, 48 kHz, audio stereo nei formati. M4v,. Mp4 e mov.

3.1.3 Cocoa touch

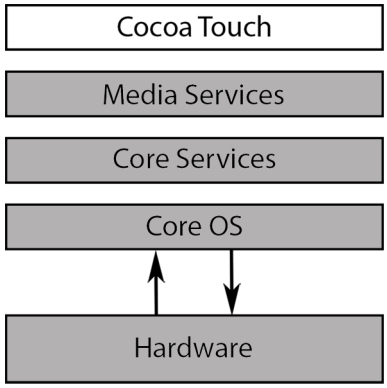


Figura 3.6:

Lo strato più alto di iOS è costituito dal livello Cocoa Touch. Cocoa Touch contiene i frameowrks chiave per la creazione di applicazioni iOS. Questo livello definisce l'infrastruttura di base e il supporto delle applicazioni per le tecnologie chiave come il multitasking, linput basato sul tocco, le notifiche push, e molti servizi di alto livello del sistema.

3.1.3.1 Layout Automatico

Introdotto nel iOS 6, layout automatico migliora il modello precedentemente utilizzato per disporre gli elementi di una interfaccia utente. Con il layout automatico è possibile definire regole che determinano la disposizione degli elementi dell'interfaccia utente. Queste regole esprimono una classe più ampia di relazioni e risultano più intuitive da usare rispetto al metodo di layout precedente.

Le entità utilizzate nel layout automatico sono oggetti Objective-C chiamati vincoli. Questo approccio offre una serie di vantaggi:

- Localizzazione attraverso un semplice scambio di stringhe.
- Mirroring di elementi dell'interfaccia utente per la scrittura da destra a sinistra in lingue come l'ebraico e l'arabo.
- Migliore stratificazione delle responsabilità tra gli oggetti negli strati della view e dei controller.

Un oggetto, solitamente, ha informazioni più dettagliate in merito alla sua dimensione normale, il suo posizionamento all'interno della superview, e il suo posizionamento rispetto al suo punto di vista di pari livello. Daltra parte, un controller può sostituire questi valori con valori non standard in caso di necessità particolari.

3.1.3.2 Storyboard

Introdotto in iOS 5, lo storyboard costituisce il nuovo metodo consigliato per progettare l'interfaccia utente dell'applicazione. A differenza dei file .nip usati precetendemente, gli storyboard consentono di progettare l'intera interfaccia

utente in un unico ambiente grafico in modo da poter vedere tutte le view e i controller e le loro connessioni. Una parte importante della storyboard è la capacità di definire transizioni da un controller ad un altro: lo sviluppatore può definire nelle proprie applicazioni tali transizioni tramite un comodo editor grafico integrato nellIDE Xcode oppure può crearle manualmente in codice. Queste transizioni consentono di controllare il flusso dell'interfaccia utente, oltre al contenuto. È possibile utilizzare un singolo file storyboard per memorizzare tutti i controller e le view di una applicazione, oppure è possibile utilizzare diversi storyboard per organizzare separatamente varie parti dellinterfaccia. In fase di compilazione, Xcode prende il contenuto del file storyboard e lo divide in parti distinte che possono essere caricate singolarmente per migliorare le prestazioni. L'applicazione non deve interfacciarsi direttamente con questi pezzi in cui è diviso lo storyboard ma ha a disposizione varie classi della libreria UIKit per accedere ai contenuti di uno storyboard dal codice.

3.1.3.3 Supporto documenti

Da iOS 5, il framework UIKit ha introdotto la classe UIDocument per la gestione dei dati associati a documenti dell'utente. Questa classe rende molto più facile l'implementazione di applicazioni basate sui documenti, in particolare le applicazioni che archiviano documenti su iCloud. Oltre a fornire un contenitore per tutti i dati relativi al documento, la classe UIDocument fornisce il supporto incorporato per la lettura e la scrittura asincrona dei dati dei file, salvataggio sicuro dei dati, il salvataggio automatico dei dati e il supporto per la rilevazione di conflitti iCloud. Per le applicazioni che utilizzano Core Data come modello di dati, è possibile utilizzare la sottoclasse UIManagedDocument per gestire gli archivi di dati.

3.1.3.4 Multitasking

Le applicazioni create utilizzando il kit di sviluppo iOS in versione 4.0 successiva (e in esecuzione su iOS 4.0 o versioni successive) non vengono terminati quando l'utente preme il tasto home, come avveniva nelle versioni precedenti, ma sono poste in un contesto di esecuzione in background. Il supporto multitasking definito da UIKit aiuta la transizione delle applicazioni da e verso lo stato di background senza problemi. Per preservare la durata della batteria, la maggior parte delle applicazioni sono sospese dal sistema poco dopo l'entrata nello stato di background. Una applicazione sospesa rimane in memoria ma non esegue alcun codice. Questo comportamento consente a un'applicazione di riprendere velocemente se viene rilanciata senza consumare la carica della batteria nel frattempo. Tuttavia, le applicazioni possono essere autorizzate a continuare l'esecuzione anche in background per i seguenti motivi:

- Un'applicazione può richiedere una quantità limitata di tempo per completare un compito importante.
- Un'applicazione può dichiararsi quale sostenitrice di servizi specifici che richiedono un tempo di durata fissa e a intervalli regolari di esecuzione in background.
- Un'applicazione può utilizzare le notifiche locali per generare avvisi degli utenti se l'applicazione è in esecuzione.

Indipendentemente dal fatto che l'applicazione è sospesa o continua a funzionare in background il supporto multitasking non richiede lavoro aggiuntivo da parte dello sviluppatore. Il sistema invia delle notifiche all'applicazione quando passa dallo stato attivo allo stato di background. Questi notifiche sono utili per capire quando svolgere importanti operazioni come il salvataggio dei dati utente.

3.1.3.5 Stampa

Introdotto in iOS 4.2, il supporto di stampa UIKit consente alle applicazioni di inviare contenuti in modalità wireless a stampanti vicine e dotate di connettività wireless. UIKit fornisce il supporto per la maggior parte delle operazioni relative alla stampa. Gli oggetti della libreria forniscono le interfacce di stampa, gli strumenti per eseguire il rendering del contenuto stampabile e gestiscono la programmazione e l'esecuzione dei job di stampa sulla stampante. I job di stampa inviati dall'applicazione verranno passati al sistema di stampa, che gestisce il processo di stampa vero e proprio. I job di stampa di tutte le applicazioni su un dispositivo vengono messi in coda e vengono stampati secondo una logica first-come/first-served. Gli utenti possono ottenere lo stato dei job di stampa dall'applicazione "Print Center" e possono anche utilizzare questa applicazione per annullare i processi di stampa. Tutti gli altri aspetti della stampa sono gestiti automaticamente dal sistema. La stampa wireless è disponibile solo su dispositivi che supportano il multitasking.

3.1.3.6 Conservazione dello stato dell'UI

Introdotto con iOS 6, la funzione di conservazione dello stato dellUI rende più facile per le applicazioni ripristinare la loro interfaccia utente allo stato in cui era l'ultima volta che lutente lha usata. Quando un'applicazione viene posta in uno stato di background, può salvare lo stato semantico della sua interfaccia. Al rilancio, l'applicazione utilizza questo stato salvato per ripristinare la sua interfaccia e far sembrare che l'applicazione non abbia mai smesso di essere eseguita. Il supporto per la conservazione dello stato è integrato in UIKit, che fornisce l'infrastruttura per salvare e ripristinare l'interfaccia delle applicazioni.

3.1.3.7 Servizio di notifica push

Introdotto in iOS 3.0, il servizio di notifiche push di Apple fornisce un modo per avvisare gli utenti di nuove informazioni o eventi, anche quando l'applicazione non è attiva in esecuzione. Utilizzando questo servizio, è possibile visualizzare le notifiche di testo, aggiungere un'icona dell'applicazione, o attivare gli avvisi acustici sui dispositivi degli utenti in qualsiasi momento. Queste notifiche consentono agli utenti di sapere che un qualche evento è accaduto e che possono aprire l'applicazione relativa allevento per ricevere le informazioni correlate. Dal punto di vista del design, ci sono due operazioni da fare per utilizzare le notifiche push nelle applicazioni iOS. In primo luogo, l'applicazione deve chiedere la consegna delle notifiche ed elaborare i dati della notifica una volta eseguita. In secondo luogo, è necessario fornire un processo sul lato server per generare le notifiche. Questo processo è eseguito sul server locale e utilizza il servizio Apple Push Notification per attivare le notifiche.

3.1.3.8 Notifiche locali

Introdotte in iOS 4.0, le notifiche locali completano l'attuale meccanismo di notifica push, dando un metodo alle applicazioni per la generazione delle notifiche a livello locale, invece di basarsi su un server esterno. Le applicazioni in esecuzione in background possono utilizzare le notifiche locali come un modo per attirare l'attenzione di un utente quando gli eventi importanti accadono. l vantaggio delle notifiche locali è che sono indipendenti dallapplicazione, infatti le applicazioni possono anche programmare lattivazione delle notifiche locali per un momento particolare che verranno attivate anche se lapplicazione non sarà in esecuzione.

3.1.3.9 Riconoscimento dei gesti

Introdotto in iOS 3.2, il riconoscimento dei gesti è realizzato da oggetti che si attaccano alle view e possono essere utilizzati per rilevare i comuni tipi di gesti, come "swip"e e "pinch". Dopo aver fissato un sistema di riconoscimento gesti per la view, si assegnano le azioni da eseguire quando si verifica il gesto. L'oggetto di riconoscimento registra gli eventi di tocco e applica le euristiche definite dal sistema per riconoscere il tipo di tocco che è stato effettuato. Fare tutto ciò senza il supporto delle librerie può essere molto complicato. UIKit include una classe UIGestureRecognizer che definisce il comportamento di base per tutti gli oggetti di riconoscimento gesti. È possibile definire sottoclassi personalizzate o utilizzare una delle sottoclassi definite in UIKit per gestire qualsiasi dei seguenti gesti standard:

- Tocco (qualsiasi numero di tocchi successivi)
- Pinch in e out (per lo zoom)
- Panning o trascinamento
- Swipe (in qualsiasi direzione)
- Rotazione (le dita in movimento in direzioni opposte attorno ad un punto)
- Pressione prolungata

3.1.3.10 Servizi peer-to-peer

Introdotto in iOS 3.0, con il framework Game Kit è possibile stabilire connessioni peer-to-peer via Bluetooth. È possibile utilizzare il peer-to-peer per avviare sessioni di comunicazione con dispositivi adiacenti e attuare molte delle caratteristiche che si trovano nei giochi multiplayer. Anche se usato principalmente nei giochi, è anche possibile utilizzare questa funzione in altri tipi di applicazioni.

3.1.3.11 View controller standard di sistema

Molti dei framework di Cocoa Touch contengono view controller standard per la presentazione delle interfacce di sistema più comuni. Si consiglia di usare questi view controller per presentare un'esperienza utente uniforme. Ogni volta che è necessario eseguire una delle seguenti operazioni, è necessario utilizzare un view controller dal framework corrispondente:

- Visualizzare o modificare le informazioni di un contatto: Address Book UI framework
- Creare o modificare il calendario eventi: Event Kit UI framework
- Comporre una e-mail o SMS: Message UI framework
- Aprire o visualizzare in anteprima il contenuto di un file: UIKit framework
- Scattare una foto o selezionare una foto dalla foto dell'utente: UIKit framework
- Riprodurre un video clip: UIKit framework.

3.1.3.12 Supporto display esterno

Introdotto in iOS 3.2, il supporto del display esterno permette ad alcuni dispositivi basati su iOS di essere collegati ad un monitor esterno attraverso una serie di cavi supportati. Quando è collegato, lo schermo associato può essere utilizzato dall'applicazione per visualizzare il contenuto. Le informazioni sullo schermo, comprese le sue risoluzioni supportate, sono accessibili attraverso le interfacce del framework UIKit. È inoltre possibile utilizzare questo framework per associare le finestre dellapplicazione con uno schermo o un altro.

Le varie funzioni appena descritte sono realizzate tramite alcuni frameworks contenuti in Cocoa Touch. Conoscere quali sono questi frameworks e come sono suddivise le funzioni è fondamentale per lo sviluppo delle applicazioni che utilizzano tale funzione.

3.1.3.13 Address Book UI Framework

Il framework Address Book UI è uninterfaccia Objective-C che consente di visualizzare le interfacce di sistema standard per la creazione di nuovi contatti e per la modifica e la selezione dei contatti esistenti. Questo framework semplifica il lavoro necessario per visualizzare le informazioni di un contatto nellapplicazione e garantisce, inoltre, che l'applicazione utilizzi le stesse interfacce delle altre applicazioni, in modo da garantire la coerenza nella la piattaforma.

3.1.3.14 Event Kit UI Framework

Introdotto in iOS 4.0, il framework Event Kit UI fornisce view controller per presentare le interfacce di sistema standard per la visualizzazione e la modifica di eventi correlati al calendario.

3.1.3.15 Game Kit Framework

Introdotto in iOS 3.0, il Game Kit framework consente di aggiungere funzionalità di rete peer-to-peer per le applicazioni e funzioni vocali in-game. Il framework fornisce funzionalità di rete attraverso un semplice (ma potente) insieme di classi costruite sulla base di Bonjour. Queste classi nascondono al programmatore molti dei dettagli nello sviluppo della rete, quindi per gli sviluppatori inesperti con la programmazione di rete, il framework permette loro di integrare funzionalità di networking nelle loro applicazioni con il minimo sforzo. Game Center

è un'estensione al framework introdotta in iOS 4.0 che fornisce il supporto per le seguenti funzionalità:

- Alias, per consentire agli utenti di creare il proprio profilo in linea. Gli utenti possono accedere a Game Center e interagire con gli altri giocatori in forma anonima attraverso i loro alias. I giocatori possono impostare i messaggi di stato e aggiungere persone specifiche come loro amici.
- Classifiche, per consentire all'applicazione di inviare i punteggi degli utenti di Game Center e recuperarli in un secondo momento. È possibile utilizzare questa funzione per mostrare i migliori punteggi tra tutti gli utenti dell'applicazione.
- Matchmaking, per consentire di creare giochi multiplayer, collegando i giocatori che hanno effettuato l'accesso in Game Center. I giocatori non devono necessariamente essere spazialmente vicini tra loro per partecipare ad una partita multiplayer.
- Obiettivi, per consentire di registrare il progresso che un giocatore ha fatto nel gioco.
- Sfide, permettono al giocatore di sfidare un amico per battere un risultato o punteggio.

In iOS 5 e versioni successive, è possibile utilizzare la classe GKTurnBasedMatch per implementare il supporto per i turni di gioco, che permette di creare giochi con partite persistenti il cui stato viene memorizzato in iCloud. Il gioco gestisce le informazioni di stato per la partita e determina quale giocatore deve agire per migliorare lo stato della partita.

3.1.3.16 iAd Framework

Introdotto in iOS 4.0, il framework iAd consente di visualizzare banner pubblicitari integrati nell'applicazione. Gli annunci sono incorporati nelle view standard che si integrano nellinterfaccia utente e vengono presentate quando si vuole. Il sistema è integrato con il servizio di annunci di Apple che gestisce automaticamente tutte le operazioni legate alla presentazione del contenuto degli annunci e alla reazione al tocco di tali annunci.

3.1.3.17 Map Kit Framework

Introdotto in iOS 3.0, il Map Kit framework fornisce un'interfaccia scorrevole di una mappa che è possibile integrare nelle view esistenti. È possibile utilizzare questa mappa per fornire indicazioni o punti di interesse. Le applicazioni possono impostare gli attributi della mappa o consentire all'utente di navigare la mappa liberamente. È inoltre possibile annotare la mappa con immagini personalizzate o contenuti. In iOS 4.0, la visualizzazione della mappa di base ha ottenuto il supporto per le annotazioni trascinabili e i moduli personalizzati. Le annotazioni trascinabili consentono di riposizionare una annotazione, a livello di codice o tramite interazioni con l'utente, dopo che è stata posta sulla mappa. Da iOS 6.0 è possibile creare una applicazione di routing, il cui compito è quello di fornire indicazioni agli utenti. Quando l'utente richiede indicazioni relative

alla navigazione, l'applicazione Mappe consente ora all'utente di scegliere l'applicazione da cui ricevere quelle direzioni. Inoltre, tutte le applicazioni possono chiedere l'applicazione Mappe per fornire indicazioni stradali e la visualizzazione di più punti d'interesse.

3.1.3.18 Message UI Framework

Introdotto in iOS 3.0, il framework Message UI fornisce il supporto per la composizione di messaggi ed e-mail nella posta in uscita in coda dell'utente. Il supporto per la composizione è costituito da un view controller che si può aggiungere allapplicazione. È possibile precompilare i campi di questo view controller per impostare i destinatari, l'oggetto, il contenuto, il corpo e gli eventuali allegati che si desidera includere con il messaggio. Dopo aver presentato il view controller, l'utente ha quindi la possibilità di modificare il messaggio prima di inviarlo. In iOS 4.0 e versioni successive, questo framework fornisce un view controller per presentare una schermata di composizione di SMS. È possibile utilizzare questo controller al fine di creare e modificare messaggi SMS senza lasciare l'applicazione. Come con l'interfaccia di composizione della posta, questa interfaccia offre all'utente la possibilità di modificare il messaggio prima di inviarlo.

3.1.3.19 UIKit Framework

Il framework UIKit è il più grande framework di iOS essendo responsabile di tutte le funzioni dellinterfaccia utente, dalla creazione delle finestre ai minimi componenti dellinterfaccia fino alla gestione degli input. Gran parte del successo della piattaforma iOS è dovuta a questo framework in quanto fornisce elementi per creare interfacce di ogni tipo in maniera ordinata e coerente tra applicazione e applicazione, semplificando luso della piattaforma in generale. Quando unapplicazione viene avviata, la sua funzione main() istanzia un oggetto di tipo UIApplication, che è la classe base per tutte le applicazioni iOS dotate di interfaccia grafica e che provvede laccesso a tutte le funzioni di alto livello viste fino ad ora e alle funzioni di controllo come sospensione, riavvio e conclusione.

UIKit mette a disposizione alcuni elementi di base per lo sviluppo di interfacce e applicazioni grafiche:

- Windows e View: sono le classi base per creare una qualsiasi interfaccia grafica. Una finestra rappresenta uno spazio geometrico sullo schermo mentre una view rappresenta un contenitore per altri oggetti. I più piccoli componenti dellinterfaccia utente sono tutti collegati ad una view che a sua volta è collegata ad una finestra. Una finestra può contenere una sola view, ma allinterno di una view possono essere contenute varie view. La finestra è implementata nella classe UIWindow mentre la view nella classe UIView. Le due classi sono strettamente interconnesse e sono entrambe necessarie per visualizzare qualsiasi cosa a schermo.

- Text View: Sono semplici classi derivate dalla classe UIView che permettono di visualizzare del testo modificabile.

- Barre di navigazione: linterfaccia utente di iOS tratta le varie schermate come se fossero pagine di un libro e utilizza le barre di navigazione per

passare da una view allaltra o per attivare particolari funzioni della view corrente.

- Transizioni: le transizioni permettono di visualizzare il passaggio da una finestra ad unaltra tramite unanimazione selezionabile tra quelle messe a disposizione dal framework.

- Alert sheets: sono finestre che compaiono al centro dello schermo per notificare allutente dellavvenimento di un qualche evento o per chiedere conferma di qualche operazione.

- Tabelle: sono oggetti che permettono la visualizzazione di liste di files, messaggi o altri tipi di collezioni. Sono comode per selezionere uno o più elementi di una lista e sono molto flessibili permettendo allo sviluppatore di definire come ogni cella deve essere rappresentata e e di definirne il comportamento.

- Manipolazione della barra di stato: la barra di stato è un elemento parte del sistema operativo che compare nel lato più alto dello schermo e che visualizza informazioni come lora, la potenza del segnale della rete telefonica e la durata residua della batteria. La barra può essere configurata a piacimento nella posizione e nellaspetto grafico ed è possibile aggiungere immagini alla barra per notificare allutente di qualche evento o di qualche operazione in corso.

Risulta di particolare importanza per il funzionamento del framework Sparrow la possibilità di creare classi derivate dalla classe UIView. Quando si crea una classe derivada da UIView è necessario sovrascrivere almeno due metodi di UIView. Il metodo initWithFrame viene chiamato quando la view viene istanziata per la prima volta e si occupa dellinizializzazione delloggetto view utilizzando le coordinate dellarea rettangolare che la view andrà ad occupare. Il metodo dealloc viene utilizzato quando la view viene eliminata, in esso qualsiasi risorsa precedentemente allocata va rilasciata in modo da liberare la memoria occupata dagli oggetti non più necessari.

3.2 Sparrow Framework

Il framework Sparrow è costituito da un insieme di classi che consentono di creare applicazioni multimediali interattive 2D e giochi 2D per la piattaforma iOS di Apple. È progettato in modo da massimizzare prestazioni e facilità duso, è gratuito, open-source ed è, per migliorare la pulizia del codice e la velocità di esecuzione, scritto esclusivamente nel linguaggio madre della piattaforma di destinazione di programmazione: Objective-C.

Essendo il progetto APP-KID dedicato solo alla piattaforma iOS, è naturale la scelta di un framework nativamente compatibile con il sistema e costruito sulle API rese disponibili da UIKit per massimizzare la compatibilità, le prestazioni e la mantenibilità del codice. Sparrow è dotato di una API particolarmente intuitiva e pulita che riprende in molti aspetti ActionScript, nota per essere particolarmente apprezzata dalla comunità di sviluppatori di applicazioni multimediali 2D in quanto semplifica molte operazioni comuni nello sviluppo di tali

software, come ad esempio le animazioni, che con altre API risultano macchinose e complicate da realizzare. Diversamente da ActionScript, Sparrow non pecca sul lato delle performance perché essendo basato sulle librerie native di iOS garantisce il massimo dellefficienza. In aggiunta, Sparrow è un framework completamente open source, e gode quindi di molti vantaggi. In primo luogo il framework è testato e controllato da unintera community, che avendo accesso completo al codice può aiutare gli sviluppatori a migliorarlo e a correggerne gli errori. Un altro vantaggio è quello di essere completamente configurabile a seconda delle esigenze dellutilizzatore e inoltre può essere modificato in ogni sua parte per aderire in ogni caratteristica alle esigenze del proprio progetto. Infine utilizzando Sparrow per il proprio progetto è possibile avvalersi di un grande numero di estensioni sviluppate dalla comunità che gravita attorno al framework, che spesso semplificano operazioni comuni e ripetitive. In alcuni casi queste estensioni vengono integrate nelle versioni successive del framework, ma la tendenza è quella di mantenere le librerie più leggere e minimali possibile. Come visto in precendenza, iOS consente di utilizzare i chip grafici dei dispositivi con OpenGL e la loro scheda audio attraverso OpenAL. Utilizzare queste librerie può essere utile in alcuni casi, ma nella maggior parte dei casi sono sufficienti le funzioni offerte da Sparrow. Tuttavia, se per esigenze particolari o per motivi prestazionali è necessario utilizzare funzioni a livello OpenGL, accedere a tali funzioni allinterno di una applicazione già sviluppata con Sparrow non solo è possibile ma è anche semplificato dal framework: Sparrow espone diversi punti di accesso che consentono di accedere allo stack OpenGL direttamente inserendo codice C che è perfettamente compatibile con Objective-C, essendone parte.

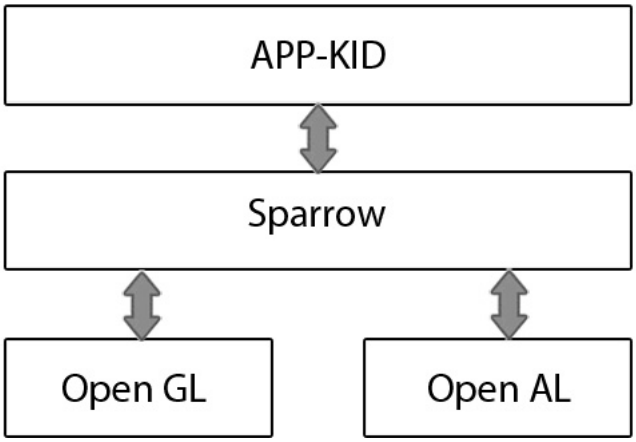


Figura 3.7:

In genere una qualsiasi applicazione grafica ha bisogno di una qualche forma di struttura gerarchica che permette di raggruppare gli oggetti insieme per poterli gestire a gruppi a seconda delle necessità. Questo non solo è adatto ad una progettazione orientata agli oggetti e aiuta a mantenere gestibile il codice del gioco, ma anche solleva lo sviluppatore da altre mansioni scomode come il monitorare e aggiornare attivamente la posizione e la rotazione di tutti gli oggetti grafici uno per uno. All'interno di Sparrow questa struttura è chiamata display tree. Ogni elemento di visualizzazione che è visibile o che interagisce con

l'utente deve risiedere all'interno del display tree. Ciò significa che se si vuole visualizzare una immagine a schermo è necessario non solo caricare l'immagine in memoria, ma anche collegarla al display tree nella posizione corretta.

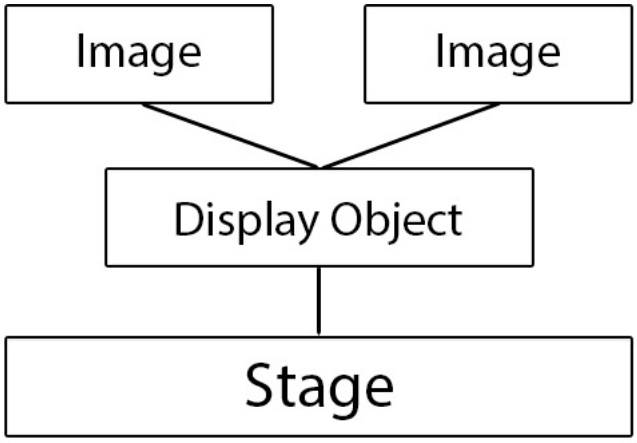


Figura 3.8:

Il display tree è utile anche per un altro motivo. Infatti Sparrow offre un potente sistema di eventi che utilizza proprio la struttura ad albero del display tree per determinare a chi devono essere visibili determinati eventi. In Sparrow, proprio come in ActionScript, qualsiasi oggetto di visualizzazione all'interno della struttura ad albero può generare eventi. Una volta che gli eventi vengono lanciati da un oggetto, si muovono verso il basso lungo i rami del display tree e visitano ogni nodo correlato fino a raggiungere il nodo radice.

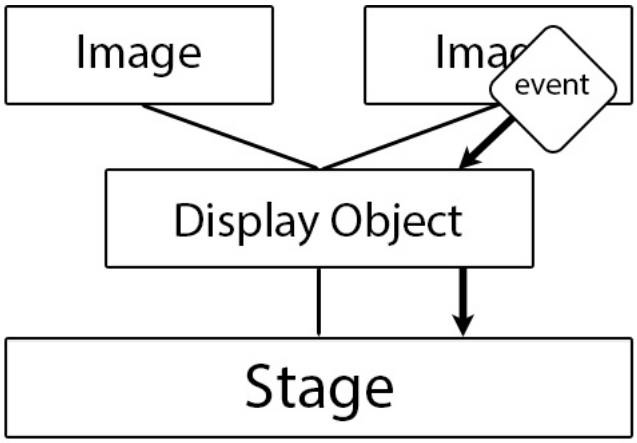


Figura 3.9:

Ci sono vari tipi diversi di eventi in Sparrow e inoltre si è liberi di creare qualsiasi tipo di evento personalizzato necessario. Alcuni eventi comunemente utilizzati, come quello corrispondente ad un tocco da parte dellutente sullo schermo del dispositivo, sono già implementati nel framework e pronti ad essere utilizzati.

Un altro potente strumento offerto da Sparrow sono i Tween. Un tween è un oggetto in grado di creare una transizione in qualsiasi proprietà numerica, dal loro valore iniziale a un valore finale definito all'interno di un arco di tempo stabilito. Il tween è una classe leggera che si occupa solo del progresso della transizione della proprietà che gestisce. Non avendo conoscenza del programma cè bisogno di un altro oggetto che gestisca il tween a seconda degli intervalli di tempo con i quali avverrà la visualizzazione. Loggetto che svolge questo compito è detto Juggler. La combinazione di questi due strumenti permette di creare animazioni, anche complesse, in maniera ancora più semplice del metodo offerto dai framework grafici di iOS, sui quali, peraltro, si basano queste meccaniche.

3.3 Simulazione della fisica

Lesigenza principale e lobiettivo primario dellesperienza di tirocinio è stato quello di aggiungere al framework esistente la funzionalità di simulazione della fisica per gli oggetti grafici presenti nelle applicazioni. Sviluppare una funzione simile può essere semplice se ci si accontenta di un risultato rudimentale, ma volendo ottenere un risultato allo stato dellarte e che sia sufficientemente efficiente da essere utilizzato su dispositivi mobili è ben più complesso e richiede molto tempo. Inoltre esistono varie librerie che offrono funzioni di simulazione della fisica e utilizzano sofisticati algoritmi per rendere la simulazione più semplice possibile da calcolare, permettendo simulazioni di scenari più complessi a parità di piattaforma. Questi software sono chiamati motori fisici. Un motore fisico è un software che fornisce una simulazione approssimativa di alcuni sistemi fisici, come la dinamica dei corpi rigidi (incluso il rilevamento di collisione), la dinamica dei corpi morbidi e la dinamica dei fluidi, impiegato generalmente nei settori della computer grafica, videogiochi e cinema. Un sistema software per la simulazione di fenomeni fisici ad uso scientifico necessita di un più basso livello di approssimazione ma non avendo bisogno della simulazione in tempo reale può regolare la precisione della simulazione a seconda di quanto tempo si ha a disposizione. Nel caso dei videogiochi o di una qualsiasi applicazione interattiva la velocità di simulazione è più importante della precisione della simulazione. Questa necessità porta a progettare motori fisici che producono i risultati in tempo reale, ma simulano la fisica del mondo reale solo per casi semplici e in genere la simulazione è orientata a fornire una approssimazione "percettivamente corretta", piuttosto che una vera e propria simulazione. I motori fisici di questo tipo in genere hanno due componenti fondamentali:

- Un sistema di collision detection.
- Una componente di simulazione della dinamica dei corpi, responsabile della risoluzione delle forze che influenzano gli oggetti simulati.

I più moderni motori fisici possono contenere anche simulazioni di fluidi, sistemi di controllo di animazione e strumenti di integrazione delle attività. Ci sono tre principali paradigmi per la simulazione fisica dei solidi:

- Metodi di penalità, dove le interazioni sono comunemente modellate come sistemi massa-molla (oscillatori). Questo paradigma è utilizzato per simulare oggetti deformabili detti o soft body.

- Metodi basati su vincoli, in cui sono risolte equazioni di vincolo che stimano le leggi fisiche.
- Metodi basati su impulsi, in cui vengono applicati impulsi di interazione tra gli oggetti.
- Metodi ibridi che combinano aspetti degli altri paradigmi.

Il limite primario al realismo dei motori fisici è la precisione dei numeri che rappresentano le posizioni e le forze che agiscono sugli oggetti. Quando la precisione è troppo bassa gli errori di arrotondamento influenzano i risultati, e anche le piccole fluttuazioni dei valori che non sono modellate nella simulazione possono cambiare drasticamente i risultati previsti: gli oggetti simulati possono comportarsi in modo imprevisto e trovarsi in posizioni e rotazioni errate.

Considerando la difficoltà di creare da zero un software con queste funzioni e la maturità delle soluzioni offerte dal mercato è conveniente scegliere la soluzione migliore in base alle esigenze. Le esigenze principali che il motore deve soddisfare sono:

- Deve essere gratuito (possibilmente open source).
- Deve essere sufficientemente performante da funzionare su dispositivi mobili.

Dalla lunga lista di motori fisici esistenti che soddisfano queste esigenze sono stati esclusi i motori fisici tridimensionali perchè il framework Sparrow supporta solo lo sviluppo di applicazioni bidimensionali, quindi è preferibile utilizzare un motore fisico progettato per la simulazione della fisica in due dimensioni. Le soluzioni rimanenti sono le seguenti:

- Box2D
- Chipmunk Physics
- Farseer Physics Engine
- Physics2D.Net

Farseer Physics Engine e Physics2D.Net essendo entrambi scritti in linguaggio C# e progettati per essere utilizzati con il framework XNA di Microsoft non sono utilizzabili allinterno di un framework scritto interamente in linguaggio Objective-C.

Le due scelte rimanenti, Box2D e Chipmunk Physics, scritti rispettivamente in C++ e C, si equivalgono sotto vari punti di vista e quindi è stato necessario osservarli più approfonditamente evidenziando i pro e i contro di ogni per poter operare una scelta tra i due.

3.3.1 Box2D

Box2D è un motore fisico 2D ricco di funzionalità, scritto in C++ da Erin Catto. Box2D è sviluppato su piattaforma Microsoft Windows utilizzando Visual C++, ma vari porting sono disponibili anche per Flash, Java, C#, Python.

Un importante aspetto positivo del motore è che non solamente è gratuito ma è anche open source e utilizza la licenza zlib che permette luso del software alle seguenti condizioni:

- Non si deve fornire informazione ingannevole sull'origine del software.
- Versioni modificate del codice sorgente devono essere chiaramente identificate come tali.
- L'informativa non deve essere rimossa od alterata da alcuna distribuzione del codice sorgente.

La licenza non richiede quindi che il software che utilizza questa libreria renda disponibile il codice sorgente.

Il fatto che Box2D sia nato come software open source ha determinato la nascita di una grande community di sviluppatori attorno al progetto che utilizzano la libreria per vari scopi. Il progetto è supportato da un forum ufficiale in cui è facile risolvere eventuali dubbi sullutilizzo della libreria. Oltre che dalla community, lutilizzo della libreria è supportato dallampia documentazione che accompagna il codice e dal manuale della versione più recente. Essendo scritto in C++, il software è fortemente orientato agli oggetti ed è quindi adatto ad essere integrato in un framework che utilizza gli stessi paradigmi. Per gestire tali oggetti è stata creata una comoda gestione interna della memoria che solleva il programmatore che utilizza la libreria dai compiti di allocare e liberare la memoria per ogni oggetto che si utilizza, delegando il compito ad una classe dedicata a tale scopo che viene utilizzata ogniqualvolta si crea o si distrugge un oggetto. Un altro aspetto positivo della libreria è che è costantemente corretta, aggiornata e dotata di nuove funzionalità. Un effetto collaterale di questo aspetto può essere che se si utilizza lultima versione disponibile della libreria è probabile trovare caratteristiche che non sono documentate nel manuale. Le nuove funzionalità vengono aggiunte al manuale dopo che sono state dichiarate mature, che sono state ampiamente testate e quando un punto di rilascio di una nuova versione è imminente. Tuttavia, tutte le principali caratteristiche aggiunte a Box2D sono accompagnate da codice di esempio nelle classi di prova, utilizzate per testare la funzionalità e mostrare l'utilizzo previsto agli utenti che così possono anche capire come implementare le nuove funzioni nei propri progetti.

3.3.2 Chipmunk Physics

Chipmunk Physics è un motore fisico 2D, scritto in C99, da Scott Lembcke con alcune parti pubblicate sotto la licenza MIT. Chipmunk è gratuito nella sua versione di base scritta in C99, ma non è totalmente open source, comunque la licenza MIT permette di utilizzare la libreria anche per scopi commerciali senza rilasciare il codice sorgente del proprio prodotto. La libreria è scritta in C99 per essere veloce, facile da ottimizzare, per trovare e correggere facilmente gli errori, e per essere facilmente utilizzabile allinterno di altri linguaggi. Essendo Chipmunk scritto in C99 si può compilare facilmente come codice C, C++, Objective-C e Objective-C + +, che derivano tutti da C99, il che rende la libreria facile da integrare in progetti che utilizzano tali linguaggi. Daltra parte, lutilizzo di C99 preclude lutilizzo di alcune funzioni offerte dai linguaggi più moderni come ad esempio la possibilità di sovrascrivere gli operatori *,+e -, ricorrendo alle normali funzioni rendendo il codice un po più difficile da leggere. Un altro problema del linguaggio C è la mancanza della restrizione di accesso. Ci sono molte strutture, i campi e le funzioni Chipmunk che pur essendo accessibili servono

solo per uso interno e non sono documentate, è quindi compito del programmatore trattare con cura tali parti di codice. Un linguaggio procedurale non orientato agli oggetti risulta anche meno naturale da integrare allinterno di un framework fortemente basato sugli oggetti. Anche la gestione della memoria è in stile C e la libreria non offre una gestione della memoria semplice e centralizzata ma lascia al programmatore tutte le operazioni di riserva e di liberazione della memoria. La documentazione che accompagna la libreria è completa di tutte le informazioni relative allultima versione del software ed è affiancata da un forum di discussione nel quale si possono risolvere dubbi e ricevere consigli sullutilizzo dell motore fisico, pur non essendo supportato da una comunità grande come quella di Box2D. Oltre alla versione gratuita esiste la versione Chipmunk Pro che oltre a varie migliorie prestazionali comprende "Objective-Chipmunk", un wrapper Objective-C per tutte le funzionalità di Chipmunk. In questo modo la gestione della memoria è molto più semplice (soprattutto se combinata con la funzione di reference counting di Objective-C), si ha una sintassi orientata agli oggetti e varie comode classi per la gestione degli input multi-touch.

Considerate le varie caratteristiche dei due motori, Chipmunk, nella versione pro, risulterebbe più completo e compatibile con il framework Sparrow soprattutto in virtù della presenza del wrapper per Objective-C. Viste però le esigenze principali espresse precedentemente, ha più senso confrontare le versioni gratuite delle librerie: in questo caso è preferibile Box2D in quanto è in grado di offrire le stesse funzionalità di Chipmunk ma in un ambiente ad oggetti, con una comoda gestione della memoria e soprattutto con il supporto di una grossa comunità di sviluppatori che può essere molto di aiuto nello sviluppo dellintegrazione nel framework Sparrow.

Capitolo 4

Progettazione e sviluppo

4.1 Analisi Sparrow Framework

Il framework Sparrow è costruito principalmente sulla base di due classi di Cocoa Touch: NSObject e UIView.

NSObject è la classe radice della maggior parte delle gerarchie di classi in Objective-C. Attraverso NSObject, gli oggetti ereditano una interfaccia di base per il sistema di runtime e la capacità di comportarsi come oggetti Objective-C. Da NSObject discendono quasi tutte le classi di Sparrow.

La classe UIView, invece, definisce un'area rettangolare sullo schermo e le interfacce di gestione del contenuto in quella zona. In fase di esecuzione, un oggetto UIView svolge tre funzioni principali allinterno dellapplicazione:

- Disegna e anima i contenuti nellarea rettangolare utilizzando tecnologie quali UIKit, Core Graphics, e OpenGL ES.
- Layout e gestione delle sub-view: una view può contenere zero o più sub-view. Ogni view definisce il proprio comportamento di ridimensionamento di default in relazione alla sua vista primaria. Una view può definire la dimensione e la posizione delle sue sub-view come necessario.
- Gestione degli eventi: Una view può gestire eventi di tocco e altri eventi definiti dalla classe UIResponder.

La classe UIView funge quindi da ponte tra il sistema di visualizzazione nativo di iOS e la gestione dei contenuti di Sparrow. Per inserire tali contenuti si può derivare una sottoclasse da UIView e implementare laggiunta di contenuti e la gestione degli eventi a mano. Dalla classe UIView deriva solamente una classe di Sparrow: SPView.

4.1.1 SPView

Un SPView non è altro che un oggetto UIView, da cui eredita, che Sparrow utilizza per renderizzare al suo interno tutto il contenuto. Una SPView può essere aggiunta alla gerarchia degli oggetti UIKit di una applicazione come una qualsiasi altra view. Per avviare lesecuzione di una applicazione che utilizza Sparrow è sufficiente collegare questa classe con la sottoclasse di SPStage che

definisce le caratteristiche della scena e chiamare il metodo -(void)start. Quando l'applicazione termina o si sposta in secondo piano, è necessario chiamare il metodo -(void)stop. La classe è dotata inoltre di 3 proprietà principali:

- isStarted: indica se la visualizzazione è stata avviata o meno.
- frameRate: assegna il framerate voluto. Solo i divisori di 60 sono consentiti.
- stage: loggetto SPStage che verrà processato.

Le altre classi di Sparrow, che derivano da NSObject, servono a caricare, gestire e utilizzare tutti i contenuti e gli eventi che andranno a costituire lapplicazione vera e propria, che verrà solo alla fine visualizzata tramite la classe SPView. La libreria è suddivisa in 6 pacchetti principali, come si vede in figura.

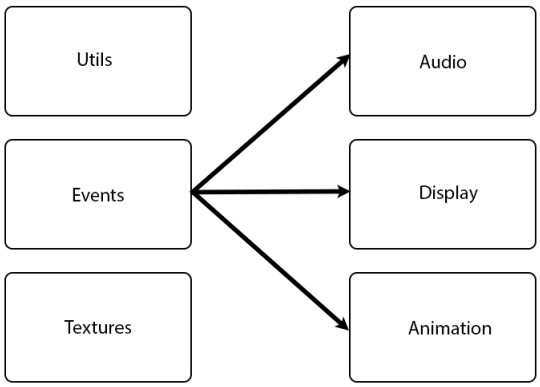


Figura 4.1:

4.1.2 Display

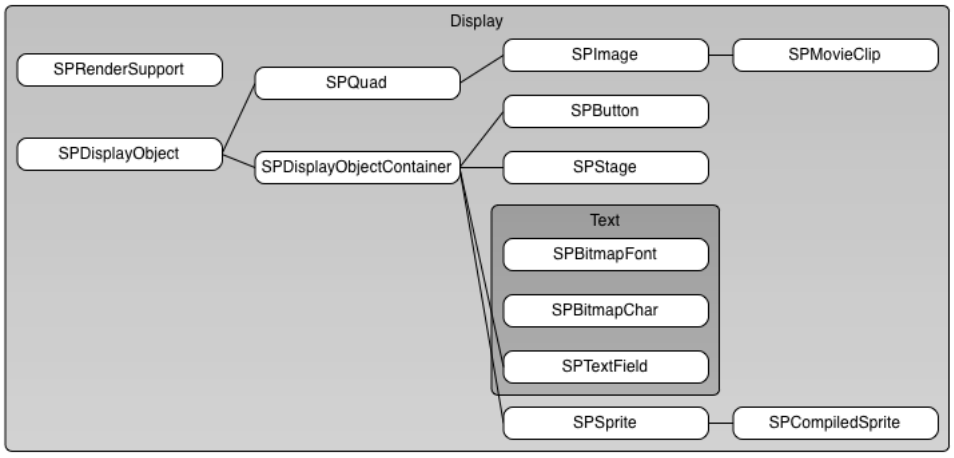


Figura 4.2:

La parte principale del framework sparrow è costituita dalle classi che gestiscono la visualizzazione degli oggetti grafici. La classe SPDisplayObject è la classe base per tutti gli oggetti che vengono visualizzati sullo schermo. In Sparrow, tutti gli oggetti visualizzabili sono organizzati in una struttura ad albero detta display tree. Solo gli oggetti che fanno parte della struttura del display tree vengono visualizzati nella fase di rendering. Il display tree è composto da nodi foglia (SPImage, SPQuad), che verranno visualizzati direttamente sullo schermo, e di nodi contenitore (sottoclassi di SPDisplayObjectContainer, come SPSprite). Un contenitore è semplicemente un oggetto di visualizzazione che ha dei nodi secondari che possono a loro volta essere sia i nodi foglia che altri nodi contenitori. Alla radice dell'albero di visualizzazione, vi è un oggetto derivato da SPStage, che è anchesso un contenitore. Per creare un'applicazione Sparrow, si lascia che la classe principale erediti da SPStage, e si costruisce il proprio albero di visualizzazione da lì.

SPDisplayObject Un SPDIsplayObject dispone di proprietà e di metodi che tutti gli oggetti di visualizzazione condividono. Le caratteristiche principali rappresentate dalle proprietà sono:

- nome
- posizione (x, y)
- dimensione (width, height)
- limiti (bounds) relativi al sistema di coordinate delloggetto genitore.
- fattore di scala (scaleX, scaleY)
- pivot (pivotX, pivotY)
- rotazione (rotation)
- opacity (alpha)
- toccabile (touchable)
- visible (visibility)
- oggetto
- genitore (parent)

Ogni oggetto di visualizzazione può essere bersaglio di eventi di tocco. Se non si desidera che un oggetto sia toccabile è possibile disattivare la proprietà touchable. Quando è disattivata, né l'oggetto né i suoi discendenti nel display tree saranno più raggiunti dagli eventi di tocco.

Tutte le dimensioni e le distanze sono misurate in punti, piuttosto che in pixel. La conversione da punti a pixel dipende dalla contentScaleFactor delloggetto SPStage. Su un dispositivo a bassa risoluzione (fino ad iPhone 3GS), un punto equivale ad un pixel. Sui dispositivi con display retina, un punto può equivalere a 2 pixel.

All'interno della struttura ad albero, ogni oggetto ha un proprio sistema di coordinate locali. Se si ruota un contenitore, si ruota tale sistema di coordinate

e quindi tutti gli oggetti figli delloggetto contenitore ruotano con esso. A volte è necessario sapere dove si trova un certo punto rispetto ad un altro sistema di coordinate. A questo scopo esiste il metodo transformationMatrixToSpace. Ricevendo in input un SPDisplayObject creerà una matrice che rappresenta la trasformazione di un punto dal sistema di coordinate delloggetto chiamante al sistema di coordinate delloggetto ricevuto in input. La rotazione di un oggetto di visualizzazione è sempre eseguita rispetto al punto di pivot. Il pivot, ossia il punto di rotazione di un oggetto, noto anche come anchor point, definisce lorigine del suo sistema di coordinate. Per default, il punto di pivot è a (0, 0), in una immagine, che è la posizione in alto a sinistra.

Poichè SPDisplayObject è una classe astratta, non è possibile istanziarla direttamente, ma è necessario utilizzare una delle sue sottoclassi. Come si può vedere dalla figura esistono già varie classi derivate da SPDisplayObject che realizzano gli oggetti di visualizzazione più comuni e nella maggior parte dei casi sono sufficienti. Tuttavia, è anche possibile creare sottoclassi personalizzate. Questo è particolarmente utile quando si desidera creare un oggetto con una funzione di rendering personalizzata o con qualche comportamento particolare. Quando si estende la classe SPDisplayObject in una nuova sottoclasse è necessario implementare i seguenti metodi:

- (void) render: (SPRenderSupport *) support : renderizza l'oggetto con l'aiuto di un oggetto di supporto.
- (SPRectangle *) boundsInSpace: (SPDisplayObject *) targetCoordinateSpace: Restituisce un rettangolo che racchiude completamente l'oggetto così come appare in un altro sistema di coordinate corrispondente a quello delloggetto ricevuto in input.

Le principali sottoclassi che implementano SPDisplayObject utilizzate nel progetto sono le seguenti.

SPStage Un SPStage è la radice del display tree di Sparrow, e rappresenta l'area di rendering dellapplicazione. Per creare un'applicazione di Sparrow, è necessario creare una classe che eredita da SPStage e popolarne lalbero di visualizzazione. Lo stage consente di accedere all'oggetto SPView sul quale sta renderizzando i contenuti grafici e, inoltre, è possibile modificare il framerate in cui viene eseguito il rendering del contenuto. È possibile accedere allo stage di una applicazione da qualsiasi punto del codice con il metodo statico [SPStage mainStage]. Uno stage contiene anche un oggetto juggler di default che, come spiegato in seguito, è possibile utilizzare per gestire le animazioni e che è gestito automaticamente dallo stage.

SPImage Un SPImage è un display object che visualizza un quadrato con una texture mappata su di esso. Sparrow utilizza la classe SPTexture per rappresentare le texture. Per visualizzare una texture, è necessario mapparla su un quadrato. SPImage eredita da SPQuad ed indirettamente da SPDisplayObject, quindi può essergli assegnato un colore. Per ogni pixel, il colore risultante sarà il risultato della moltiplicazione del colore della texture con il colore del quadrato. In questo modo, si può facilmente colorare la texture con un determinato colore. Inoltre, SPImage permette la manipolazione delle coordinate della texture. In

questo modo, è possibile spostare una texture all'interno di un'immagine senza modificare le coordinate dei vertici del quadrato.

SPSprite Un SPSprite è la classe contenitore di oggetti grafici più semplice. Uno sprite si può utilizzare come un semplice mezzo di raggruppamento di oggetti in un unico sistema di coordinate. SPSprite estende direttamente la classe astratta SPDisplayObjectContainer, che rappresenta un insieme di oggetti di visualizzazione. Mantenendo un elenco ordinato di oggetti figli, definisce i livelli di sovrapposizione degli oggetti secondo la struttura ad albero. La larghezza e laltezza del contenitore dipendono quindi dagli oggetti contenuti, e cambiare la scala delloggetto cambierà conseguentemente le dimensioni dei figli. La classe definisce i metodi che consentono di aggiungere o rimuovere gli oggetti. Quando si aggiunge un oggetto figlio, verrà aggiunto in posizione più avanzata, occludendo eventualmente un oggetto già presente.

4.1.3 Animation

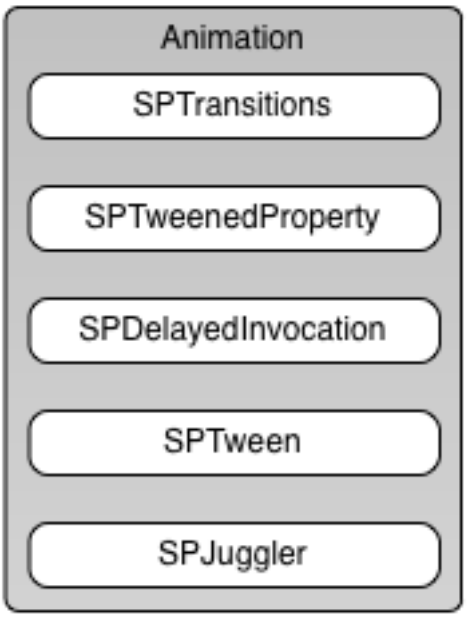


Figura 4.3:

Le animazioni sono una parte fondamentale di ogni applicazione multimediale. Sparrow offre un metodo semplice per realizzare animazioni. In generale esistono due tipi di animazioni. In un primo tipo di animazioni si sa fin dallinizio qualè lo stato iniziale e lo stato finale delloggetto grafico, e si tratta solo di applicare la transizione corretta. Un secondo tipo di animazioni, più dinamico, considera un movimento che può cambiare in ogni fotogramma, in quanto può dipendere dallinput dellutente.

SPEnterFrameEvent Nella maggior parte dei framework per la realizzazione di applicazioni grafiche esiste un ciclo principale chiamato run-loop. Si tratta

di un ciclo infinito che costantemente aggiorna tutti gli elementi della scena. In Sparrow, a causa dell'architettura a display tree, un ciclo di esecuzione di questo tipo non avrebbe senso. Avendo separato il gioco in numerosi diversi elementi di visualizzazione personalizzati, ognuno dovrebbe sapere da sé cosa fare quando un certo intervallo di tempo è passato. Per questo motivo esiste il SPEnterFrameEvent. Questa classe rappresenta un evento che viene notificato a qualsiasi oggetto di visualizzazione, e che viene attivato una volta in ogni fotogramma. Il metodo "onEnterFrame:, impostato come listener per tale evento, viene chiamato una volta per fotogramma, ed ha a disposizione il tempo che è passato dall'ultimo fotogramma, dato che il frame-rate è variabile, per poter effettuare una animazione a velocità costante.

SPTween Le animazioni predefinite sono molto più comuni e per questo motivo Sparrow ha un approccio molto semplice ma potente per questo tipo di animazioni. In sostanza, è possibile animare qualsiasi proprietà di qualsiasi oggetto, a patto che sia una proprietà numerica di tipo int, float o double. Queste animazioni sono descritte in un oggetto chiamato "SPTween". Il termine "tween" deriva dalle animazioni disegnate a mano, in cui un illustratore capo disegnava i fotogrammi chiave, mentre il resto dei disegnatori aveva il compito di riempire con i fotogrammi mancanti lo spazio tra un frame e laltro. Il sistema funziona esattamente in questo modo: è sufficiente specificare il valore finale della variabile da animare e in quanto tempo deve essere raggiunto per creare una animazione. Inoltre è possibile cambiare il modo in cui viene eseguita l'animazione, cioè in maniera non lineare ma con accelerazioni e decelerazioni. Ciò è possibile semplicemente specificando il tipo di animazione desiderata tra quelle disponibili, come in figura 4.4.

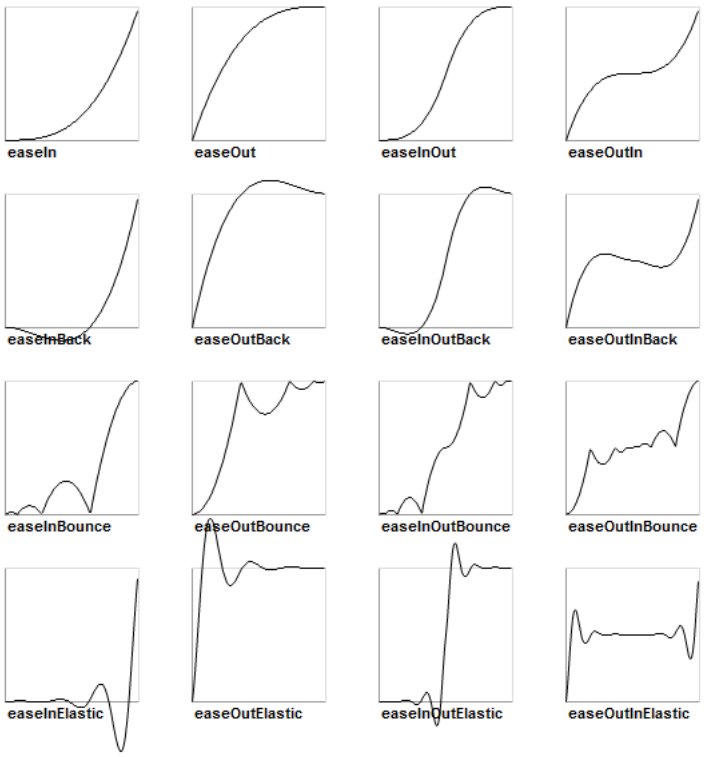


Figura 4.4:

Un oggetto SPTween descrive solamente l'animazione, ma non la esegue. Per fare ciò esiste loggetto SPJuggler. Il juggler prende tutto ciò che può essere animato (vale a dire, tutto ciò che implementa il protocollo SPAnimatable), ed esegue l'animazione. C'è sempre un juggler predefinito disponibile in ogni SPStage. Il modo più semplice per eseguire un'animazione è aggiungere loggetto SPTween al juggler con il metodo addObject: . Mentre nella maggior parte dei casi questo approccio semplice funziona, a volte si desidera raggruppare le animazioni in diversi contesti ad esempio separando le animazioni di gioco da quelle dellinterfaccia, in modo da poterle bloccare mentre il gioco è in pausa. In questo caso è sufficiente creare un juggler personalizzato che affianchi quello di default. Quando si crea un juggler personalizzato, è necessario chiamare il suo metodo "advanceTime:" ad ogni fotogramma.

Infine esiste un altro tipo di animazione meno comune degli altri due. A volte, si vuole fare in modo che le conseguenze di un qualche evento si verifichino con un certo ritardo dallevento stesso. Il juggler può essere utile anche in questo caso infatti, a tale scopo, è sufficiente effettuare una chiamata di questo tipo: [juggler delayInvocationAtTarget:self byTime:3.0f] metodoDaChiamare];

La sintassi può sembrare strana ma il metodo delayInvocationAtTarget è come se attendesse il tempo specificato prima di ritornare loggetto target sul quale verrà chiamato il metodo del quale si voleva ritardare la chiamata.

4.1.4 Events

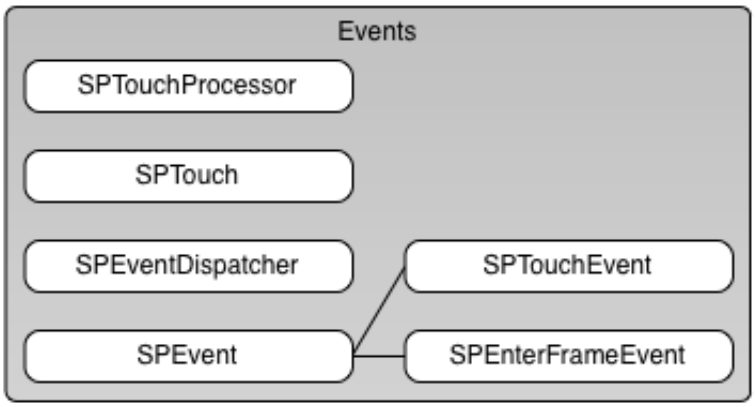


Figura 4.5:

In Sparrow qualsiasi oggetto di visualizzazione, quindi derivante da SPDisplayObject, all'interno della struttura ad albero può generare eventi, perchè SPDisplayObject deriva da SPEventDispatcher. Una volta che un evento si verifica la sua notifica percorre i rami dell'albero verso la radice notificando ogni nodo correlato fino al nodo SPStage.

La classe SPEventDispatcher è la base per tutte le classi che inviano eventi. Il meccanismo degli eventi è una caratteristica fondamentale dell'architettura di Sparrow. Gli oggetti possono comunicare tra loro tramite gli eventi. Un dispatcher di eventi può inviare eventi (oggetti di tipo SPEvent o una delle sue sottoclassi) agli oggetti che si sono registrati in qualità di listener. Una stringa, con il nome del tipo di evento, viene utilizzata per identificare eventi diversi. Per inviare un evento è sufficiente utilizzare il metodo dispatchEvent, mentre per ricevere la notifica di un evento, un oggetto deve registrarsi come listener per quellevento ed indicare che metodo lanciare al verificarsi di tale evento, che ovviamente deve essere implementato. Data la struttura ad albero, un listener può registrarsi per un tipo di evento, non solo per l'oggetto che lo spedisce, ma su qualsiasi oggetto che sia un genitore diretto o indiretto del dispatcher.

La classe SPEvent contiene dati che descrivono un evento. Un oggetto dispatcher crea istanze di questa classe e le invia ai listener registrati. Un evento contiene informazioni che caratterizzano un evento, soprattutto il tipo di evento e se levento deve risalire il display tree fino alla radice. Il target di un evento è l'oggetto che lo ha spedito. Per alcuni tipi di eventi, queste informazioni sono sufficienti, mentre per altri eventi potrebbero essere utili ulteriori informazioni da notificare al listener. In questo caso, è possibile creare sottoclassi di SPEvent e aggiungere proprietà con tutte le informazioni necessarie. Di fondamentale importanza è la classe SPToucheEvent, che tra le proprietà aggiuntive gestisce oggetti di tipo SPTouch. Un oggetto SPTouch contiene informazioni sulla presenza o sul movimento di un dito sullo schermo, cioè su un singolo tocco. Ogni tocco, normalmente, si muove attraverso tre fasi della sua vita cioè inizio, movimento, fine. Informazioni sulla fase attuale sono contenute nelloggetto SPTouch. Inoltre, un tocco può entrare in una fase stazionaria. Questa fase non attiva un evento di tocco, e può avvenire solo quando la funzione multitouch è

attivata. In una situazione in cui si muove un dito e l'altro è stazionario, sarà inviato un evento di tocco solo per l'oggetto sotto il dito che si muove. Nell'elenco dei tocchi di quell'evento, si trova il secondo tocco nella fase stazionaria. È possibile ottenere la posizione corrente e l'ultima posizione sullo schermo con le proprietà corrispondenti, nel sistema di coordinate globale. Per questo motivo, ci sono metodi che convertono le coordinate dei tocchi nel sistema di coordinate locale di qualsiasi oggetto.

4.1.5 Textures

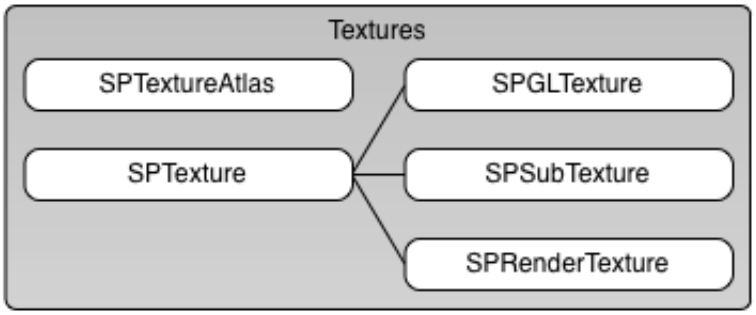


Figura 4.6:

Una texture memorizza le informazioni che rappresentano un'immagine. Non può essere visualizzata direttamente, ma deve essere mappata su un oggetto di visualizzazione. In Sparrow, tale oggetto di visualizzazione è costituito dalla classe SPImage.

Sparrow supporta diversi formati di file per le texture. I formati più comuni sono PNG, che contiene un canale alfa, e JPG (senza un canale alfa). È inoltre possibile caricare i file in formato PVR (compresso o non compresso). PVR è un formato speciale del chip grafico di dispositivi iOS che è molto efficiente.

Inoltre, Sparrow permette lo sviluppo di applicazioni in risoluzioni multiple, vale a dire la creazione di una applicazione che funzioni allo stesso tempo per i display normali e a risoluzione maggiore. Se il supporto per le texture ad alta definizione viene attivato (tramite il metodo setSupportHighResolutions) e si carica una texture da un file chiamato ad esempio image.png, Sparrow verificherà se trova un file chimato image@2x.png e, in caso positivo, caricherà questultimo a condizione che l'applicazione sia in esecuzione su un dispositivo ad alta definizione. L'oggetto SPTexture restituirà i valori di larghezza e altezza pari al numero originale di pixel diviso 2 (impostando il loro fattore di scala a 2.0). In questo modo, si lavora sempre con gli stessi valori di larghezza e altezza indipendentemente dal tipo di dispositivo. È anche possibile commutare texture a seconda del dispositivo sulla quale viene eseguita l'applicazione. La convenzione è quella di aggiungere un modificatore per dispositivo (~ipad o ~iphone) per il nome dell'immagine, direttamente prima l'estensione del file e dopo il modificatore di scala, se presente.

Sparrow consente di creare grafica personalizzata direttamente in fase di esecuzione utilizzando le API Core Graphics. È possibile accedere alle API di disegno con uno speciale metodo di inizializzazione di SPTexture, che prende un parametro block di Objective-C che si può riempire con il codice di disegno.

La proprietà frame di una texture permette di definire la posizione in cui la texture appare all'interno di un oggetto SPImage. Il rettangolo è specificato nel sistema di coordinate della texture. La classe texture di per sè non fà alcun uso dei dati di frame. Tocca a classi che utilizzano SPTexture supportare tale funzione.

4.1.6 Audio

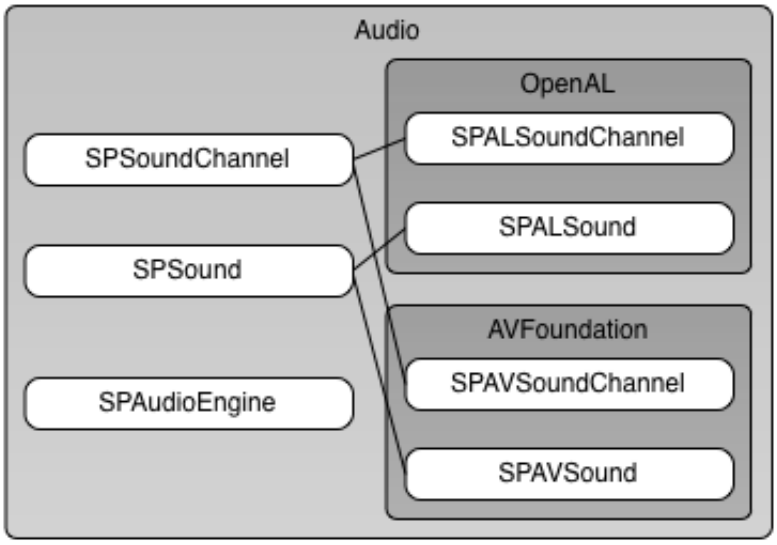


Figura 4.7:

Il motore audio, ossia loggetto che gestisce tutti i suoni di una applicazione Sparrow, deve essere inizializzato all'avvio dellapplicazione, e fermato prima che lapplicazione finisca. Tale oggetto è definito dalla classe SPAudioEngine, che prepara il sistema per la riproduzione audio e controlla il volume globale. Prima di riprodurre i suoni, si dovrebbe avviare una sessione audio. Il tipo di sessione audio definisce il modo in cui iOS si occuperà dellelaborazione audio e come il resto dellaudio di sistema si mescola con l'audio dellapplicazione.

La classe SPSound contiene dati audio che pronti per la riproduzione. Proprio come SPTexture contiene dati di immagine, SPSound contiene dati audio. La classe è in grado di caricare i file audio in diversi formati e li di mantenerli in memoria (o di effettuare uno stream, se supportato). È possibile utilizzare SPSound per riprodurre un suono direttamente, ma questa modalità non permette di controllare la riproduzione. Il vantaggio di usare questa classe è che se si avesse bisogno di riprodurre un suono più volte si può creare l'oggetto SPSound solo una volta e tenerlo in memoria, quindi utilizzarlo ogniqualvolta si desideri riprodurre il suono. I suoni verranno automaticamente messi in pausa quando l'applicazione viene interrotta (ad esempio da una telefonata), e continueranno la riproduzione da dove si erano fermati. La classe SPSound sceglierà autonomamente la tecnologia appropriata per la riproduzione: per i file non compressi utilizzerà OpenAL, mentre per i suoni compressi utilizzerà AVAudioPlayer.

Se si desidera controllare la riproduzione e il volume del suono, è necessario creare un oggetto SPSoundChannel. Un SPSoundChannel rappresenta

una sorgente audio. I sound channel sono creati con il metodo createChannel. Permettono un controllo sulla riproduzione tramite tre semplici metodi (play, pause, stop) e sulle proprietà come il volume o se il suono deve essere ripetuto ciclicamente. Inoltre, quando finisce la riproduzione di un suono la classe invia automaticamente eventi che notificano tale avvenimento. Prima di eliminare un canale, è una buona abitudine chiamare stop o rimuovere eventuali listener di eventi collegati ad esso.

4.1.7 Utils

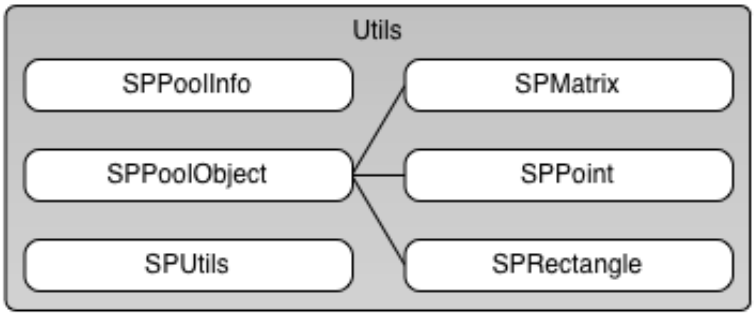


Figura 4.8:

La classe SPUtils contiene metodi di utilità per scopi diversi come il caricamento dei files o la generazione di numeri random.

La classe SPPoolObject è un'alternativa alla classe NSObject come base per gestire un gruppo di oggetti. Le sottoclassi di SPPoolObject non rilasciano le istanze degli oggetti quando il contatore dei puntatori raggiunge lo zero, invece gli oggetti rimangono in memoria e vengono riutilizzati quando una nuova istanza dell'oggetto viene richiesta. In questo modo, l'inizializzazione oggetto viene accelerata. È possibile comunque rilasciare la memoria da tutti gli oggetti riciclati in qualsiasi momento chiamando il metodo purgePool.

Sparrow utilizza questa classe per SPPoint, SPRectangle e SPMatrix, in quanto sono classi utilizzate molto spesso come oggetti di supporto.

4.2 Analisi Box2D

La libreria Box2D è composta da tre moduli: Common, Collision, e Dynamics. Il modulo common ha il codice per la gestione della memoria, le funzioni matematiche e le impostazioni. Il modulo Collision definisce le forme e le funzioni per la simulazione delle collisioni. Infine, il modulo Dynamics offre lambiente di simulazione, i corpi, le fixture, e i joint.

4.2.1 Common

La parte di impostazioni della libreria di questo modulo contiene:

- Tipi di dato: Box2D definisce vari tipi, come float32, int8, per rendere più facile determinare la dimensione delle strutture.

- Costanti: Box2D definisce diverse costanti, tutte documentati in b2Settings.h. Normalmente non è necessario modificare queste costanti. La libreria utilizza calcoli in virgola mobile per la verifica delle collisioni e la simulazione. A causa degli errori di arrotondamento sono definite alcune tolleranze numeriche, certe sono assolute e altre sono relative.

- Wrapper di allocazione: Il file di impostazioni definisce b2Alloc e b2Free per le allocazioni di memoria di grandi dimensioni. Si può utilizzare queste chiamate nel proprio sistema di gestione della memoria.

- Il numero di versione della libreria è contenuto in una struttura chiamata b2Version, che può essere interrogata in fase di esecuzione

Nella parte di gestione della memoria, molte scelte di progettazione di Box2D sono basate sulla necessità di un uso rapido ed efficiente della memoria. Box2D tende ad allocare un numero elevato di oggetti di piccole dimensioni (circa 50-300 bytes). Utilizzare la heap di sistema attraverso malloc o new per gli oggetti di piccole dimensioni è inefficiente e può causare la frammentazione della memoria. Molti di questi piccoli oggetti possono avere una vita breve, come i contatti, ma comunque possono persistere per vari intervalli di tempo. Quindi è necessario un sistema che possa provvedere una efficiente gestione della memoria heap per questo tipo di oggetti. la soluzione offerta da Box2D è quella di utilizzare uno Small Object Allocator(SOA) chiamato b2BlockAllocator. Il SOA mantiene un certo numero di pool di memoria variabili in dimensioni. Quando viene effettuata una richiesta per la memoria, il SOA restituisce il blocco di memoria che meglio si adatta alle dimensioni richieste. Quando il blocco viene liberato, viene restituito al pool. Entrambe queste operazioni sono veloci e causano poco traffico di memoria heap. Dal momento che si usa il b2BlockAllocator, non si dovrebbe mai chiamare new o malloc per creare un oggetto della libreria. Tuttavia, è necessario avere un modo per creare tali oggetti a piacimento. A questo scopo la classe b2World, che gestisce lambiente di simulazione, fornisce metodi per la creazione di tutti gli oggetti utili a chi utilizza la libreria, permettendo di utilizzare il SOA di Box2D ma creando gli oggetti in maniera semplice e intuitiva. Per questo motivo non bisogna tentare di liberare la memoria manualmente. Durante l'esecuzione di un intervallo di tempo, Box2D necessita di una porzione di memoria temporanea per effettuare i propri calcoli. Per questo scopo utilizza un allocatore sulla stack chiamato b2StackAllocator per evitare allocazioni su heap durante lelaborazione di un intervallo temporale, perchè se così fosse lesecuzione ne sarebbe rallentata.

Lultima parte del modulo Common comprende un piccolo e semplice modulo dedicato a vettori e matrici. Questo modulo è stato reso più semplice possibile per rendere la libreria facilmente portabile e più facile da mantenere.

4.2.2 Collision

Il modulo Collision contiene le forme e le funzioni che operano su di esse. Il modulo contiene anche un albero dinamico e un algoritmo di broad-phase per il calcolo delle collisioni in sistemi complessi. Il modulo di collisione è progettato per essere utilizzabile al di fuori del sistema dinamico di simulazione della fisica, ad esempio per rilevare la sovrapposizione tra due oggetti grafici.

Le forme, genericamente descritte nella classe b2Shape, rappresentano la geometria di collisione e possono essere utilizzate in modo indipendente di simulazione fisica. La classe di base b2Shape definisce le funzioni di:

- Verificare la sovrapposizione di uno specifico oggetto con la forma.
- Eseguire un ray-cast sulla forma.
- Calcolare la AABB della forma.
- Calcolare le proprietà di massa della forma.

Inoltre, ogni forma ha un tipo e un raggio, che nei poligoni è approssimato.

Forme Circolari Le forme circolari, definite in b2CircleShape hanno una posizione e un raggio. Le forme circolari sono solamente piene, cioè non è possibile creare una forma circolare cava. Tuttavia, è possibile creare catene di segmenti utilizzando forme poligonali per approssimare un cerchio vuoto.

Forme poligonali Le forme poligonali sono poligoni convessi esclusivamente pieni. Intuitivamente, un poligono è convesso quando tutti i segmenti che collegano due punti interni qualsiasi del poligono non attraversano alcun bordo del poligono.

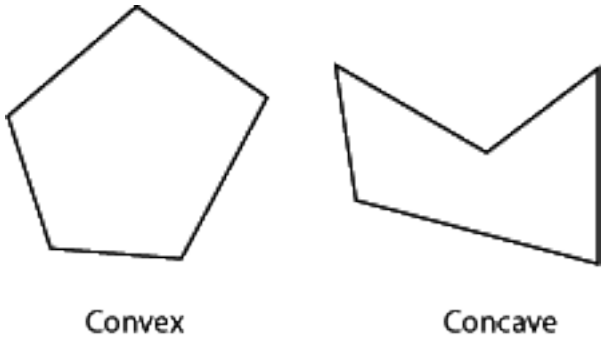


Figura 4.9:

Un poligono deve avere 3 o più vertici. Quando si definisce un poligono è necessario specificare vertice per vertice, in senso anti-orario, rispetto ad un sistema destrorso di coordinate con l'asse z che punta fuori dal piano. Questo fatto potrebbe creare problemi quando si usa la libreria in congiunzione con altre librerie come Sparrow che utilizzano un diverso orientamento degli assi. Per creare un poligono è necessario utilizzare le funzioni di inizializzazione che eseguono automaticamente la convalida dei vertici inseriti. La dimensione massima dellarray di vertici che si utilizza nellinizializzazione è controllata dalla costante b2_maxPolygonVertices che ha un valore di default di 8, che è sufficiente nella maggior parte dei casi.

La forma poligonale ha alcune funzioni di inizializzazione personalizzate per creare forme di parallelepipedi, specificando solamente altezza e larghezza del parallelepipedo, oppure aggiungendo anche informazioni sul centro e sullangolo di rotazione rispetto agli assi globali.

Le forme poligonali ereditano da b2Shape la variabile raggio. Il raggio specifica lo spessore dello strato che circonda il poligono come una pelle, che viene utilizzato in scenari che comprendono pile di poligoni uno sopra allaltro per mantenere poligoni leggermente separati. Questo permette il funzionamento della collisione continua con un poligono di base. Il raggio del poligono aiuta a prevenire leffetto di tunneling mantenendo i poligoni separati. Ciò si traduce nella presenza di piccoli spazi tra le forme, che possono essere nascosti dallutilizzo di immagini leggermente più grandi delle forme sottostanti.

Forme-bordo Le forme-bordo, descritte nella classe b2EdgeShape, sono sostanzialmente dei segmenti. Questo tipo di forme è fornito per aiutare a creare un ambiente statico di forme irregolari e non semplicemente un piano. Una limitazione importante di questo tipo di forme è che possono entrare in collisione solo con forme circolari e poligonali, ma non con se stesse. Gli algoritmi utilizzati da Box2D per rilevare le collisioni richiedono che almeno una delle due forme collidenti abbia un volume. Forme di questo tipo, non avendo volume, non possono collidere tra loro. In molti casi, un ambiente di simulazione viene creato collegando diverse forme-bordo tra loro facendo combaciare le loro estremità. Ciò può dar luogo ad un comportamento imprevisto quando un poligono scivola lungo la catena di spigoli. Nella figura seguente si vede una forma poligonale collidere con un vertice interno.

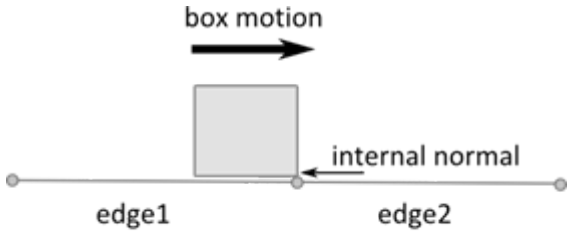


Figura 4.10:

Queste collisioni, chiamate ghost collision, sono generate durante la collisione della forma poligonale con un vertice del bordo generando una collisione. Se il bordo1 non esistesse questa collisione sembrerebbe normale, ma con il bordo1 presente, la collisione sembra un bug. La classe b2EdgeShape fornisce un meccanismo per eliminare le ghost collision memorizzando quali sono i vertici adiacenti che non devono generare collisioni.

Forme a catena Solitamente unire le forme-bordo una dopo laltra in questo modo è un problema semplice ma lungo da svolgere a mano. Per risolvere questo problema è stata creata la forma a catena, definita nella classe b2ChainShape. Questa forma rappresenta un modo efficiente di collegare molti bordi assieme per costruire scenari irregolari. Definendo una forma a catena è possibile eliminare automaticamente le ghost collision e fornire le collisioni solo nelle due estremità. Per creare tale forma è sufficiente specificare in successione i punti che costituiscono i vertici della catena ed automaticamente verranno eliminate le ghost collision nei vertici interni. Si possono inoltre collegare tra loro più catene semplicemente come si collegavano due forme-bordo, quindi specificando

i vertici di collisione. Infine è possibile chiudere automaticamente una catena creando un loop semplicemente chiamando la funzione CreateLoop. Nel creare tali cicli è importante che non si generino auto-intersezioni tra i lati della catena in quanto strutture di questo tipo non sono supportate. La simulazione potrebbe funzionare ma lalgoritmo utilizzato non garantisce un corretto funzionamento in casi di questo tipo.

Funzioni di test Tra le funzioni offerte dal modulo Collision cè la possibilità di verificare se un punto è in sovrapposizione con una forma. È sufficiente utilizzare la funzione TestPoint che riceve come parametri la posizione del punto e un oggetto b2transform, che contiene informazioni sulla posizione e sulla rotazione della forma. Applicando questo test alle edge shape, e quindi anche le chain shape, si ha sempre esito negativo, anche se la catena è un ciclo.

Per verificare un possibile contatto con una di queste forme è possibile utilizzare il ray-cast. Questo metodo consiste nel simulare la proiezione di un raggio verso una forma per ottenere il primo punto di intersezione con essa e il vettore normale su quel punto. Affinchè questo metodo funzioni è necessario che la sorgente del raggio sia posta allesterno della forma. Nel caso di una chain shape verrà restituito anche lindice di quale bordo è stato colpito dal raggio.

Funzioni bilaterali Il modulo Collision contiene funzioni bilaterali che prendono un coppia di forme in input e calcolano alcuni risultati.

- Sovrapposizione
- Contact Manifold
- Distanza
- Tempo di impatto

È possibile verificare la sovrapposizione tra due forme utilizzando la funzione b2TestOverlap. Anche in questo caso è necessario fornire indici delle estremità per il caso delle forme a catena.

Box2D è dotato di varie funzioni per calcolare i punti di contatto nella sovrapposizione di forme. Se si considerano collisioni cerchio-cerchio o cerchio-poligono, si possono ottenere solo un punto di contatto e una normale. Nel caso di collisioni poligono-poligono possiamo ottenere invece due punti. Questi punti condividono lo stesso vettore normale così Box2D li raggruppa in una struttura b2Manifold. Lalgoritmo di risoluzione dei contatti può sfruttare questa struttura per migliorare la stabilità della collisione. Normalmente non è necessario calcolare queste strutture direttamente, tuttavia è probabile che si utilizzino i risultati ottenuti dalla simulazione. La struttura b2Manifold contiene un vettore normale e fino a due punti di contatto. I punti e le normali e sono espressi in coordinate locali. I dati memorizzati in b2Manifold sono ottimizzati per l'uso interno della libreria, se si avesse bisogno di questi dati può essere preferibile utilizzare la struttura b2WorldManifold per generare le coordinate globali.

La funzione b2Distance può essere usata per calcolare la distanza tra due forme. La funzione di distanza necessita che entrambe le forme siano convertite in oggetti di tipo b2DistanceProxy, che è un oggetto di convenienza per lalgoritmo di calcolo della distanza. Essendo una funzione di uso molto comune è

supportata da un sistema di caching per migliorare le prestazioni nelle chiamate ripetute.

Se due forme si muovono molto velocemente luna contro laltra, si può verificare il cosiddetto effetto di tunneling, ossia che si attraversano senza generare nessun contatto. La funzione b2TimeOfImpact viene utilizzata per determinare il momento in cui due forme in movimento collidono. Questo è chiamato il tempo di impatto (TOI, time of impact). Lo scopo principale di b2TimeOfImpact è la prevenzione delleffetto tunnel. In particolare, è progettato per impedire ad oggetti in movimento di fuoriuscire dalla geometria del livello statico. Questa funzione tiene conto sia della rotazione che traslazione di entrambe le forme, ma se le rotazioni sono abbastanza grandi la funzione può non registrare una collisione. La b2TimeOfImpact identifica un asse di separazione iniziale e si assicura che le forme non attraversino tale asse. Questo approccio potrebbe non registrare alcune collisioni ma è molto veloce e adeguato per prevenire il tunneling. Un approccio più robusto potrebbe consistere nel limitare lampiezza delle rotazioni in un tempo di frame, ma risulta difficile stabilire tale limite senza influire sullusabilità della libreria. Inoltre possono esserci casi in cui alcune collisioni non vengono registrate a causa di piccole rotazioni, ma essendo piccole non influiscono molto nella resa dellapplicazione. Infine è possibile utilizzare rotazioni fisse per eseguire uno shape-cast, lequivalente di un ray-cast ma con la proiezione di una intera forma . In questo caso la funzione b2TimeOfImpact non perderebbe eventuali collisioni.

Albero dinamico La classe b2DynamicTree viene utilizzata da Box2D per organizzare un grande numero di forme in modo efficiente. La classe non utilizza oggetti derivati da b2Shape perché utilizza esclusivamente gli AABB (Axis-Aligned-Bounding-Box) delle forme. L'albero dinamico è un albero gerarchico di AABB. Ogni nodo interno nella struttura ad albero può avere due figli. Un nodo foglia è un singolo AABB. L'albero utilizza rotazioni per mantenersi in equilibrio, anche nel caso di un input degenere.

La struttura ad albero consente di effettuare ray-cast e shape-cast in maniera efficiente. Ad esempio, avendo centinaia di forme nella scena, sarebbe possibile eseguire un ray-cast sulla scena in brute-force su forma. Questo metodo sarebbe però inefficace, perché non approfitta delle informazioni sulla distribuzione delle forme. Invece, è possibile mantenere un albero dinamico ed eseguire il ray-cast sullalbero per poi eseguire un ray-cast sulle forme relative agli AABB incrociati dal primo raggio.

Similmente, una region query utilizza l'albero per trovare tutte le AABB foglia che si sovrappongono una AABB di input, escludendo a priori le forme non comprese in questa ricerca preliminare.

Normalmente non si utilizza lalbero dinamico direttamente. Piuttosto, si utilizzano i metodi forniti dalla classe b2World per ray-cast e region query.

Broad-phase Lelaborazione delle collisioni in uno step della simulazione della fisica può essere diviso in due fasi, narrow-phase e broad-phase. Nella narrow-phase si calcolano i punti di contatto tra coppie di forme. Immaginando di avere n forme, utilizzando un algoritmo di forza bruta dovremmo eseguire la narrow-phase per n * n / 2 coppie. La classe b2BroadPhase riduce questo carico utilizzando un albero dinamico per la gestione delle coppie, che riduce notevolmente

il numero di chiamate della narrow-phase . Normalmente non si interagisce con la broad-phase direttamente, che viene creata e gestita internamente da Box2D.

4.2.3 Dynamics

Il modulo Dynamics è la parte più complessa di Box2D ed è la parte con la quale si interagisce maggiormente nello sviluppo di applicazioni. Il modulo Dynamics utilizza i moduli Common e Collision. Il modulo Dynamics contiene tutte le classi che si utilizzano nello sviluppo di una applicazione dotata di simulazione della fisica, dallambiente di simulazione ai singoli corpi rigidi.

4.2.3.1 World

La classe b2World contiene i body e i joint che li uniscono. Gestisce tutti gli aspetti della simulazione e consente lesecuzione di query asincrone (come query region-query e ray-cast). La creazione di un b2World è abbastanza semplice. È sufficiente fornire un vettore di gravità e un valore booleano che indica se i corpi contenuti possono essere disattivati. Di solito si crea e si distrugge il mondo con new e delete, diversamente da tutti gli altri oggetti. La classe b2World contiene metodi detti factory per la creazione e la distruzione di tutti gli oggetti che animano la scena.

La classe b2World è utilizzata per gestire la simulazione ad intervalli discreti di tempo. Per simulare un intervallo temporale si utilizza la funzione Step che, dato in input un valore di tempo in secondi, simula levoluzione della scena in quellintervallo temporale. Chiamando ripetutamente questa funzione si ottiene un effetto di movimento.

Dopo ogni step è possibile esaminare body e joint per ottenere le informazioni su come sono cambiati. Loperazione più comune è quella di leggere la posizione e la rotazione degli oggetti in modo da poter aggiornare gli oggetti grafici a cui corrispondono e che stanno simulando. È possibile eseguire il passo temporale in qualsiasi punto del ciclo di esecuzione, ma è importante ricordare che verranno simulati solo i corpi che sono già stati creati, quindi è buona norma creare gli oggetti prima dellesecuzione dello step. Il parametro timestep che si utilizza nella funzione è necessario che sia costante. Utilizzando un passo temporale più grande è possibile migliorare le prestazioni in scenari più complessi, ma in generale è necessario utilizzare un passo temporale non superiore a 1/30 di secondo. Con un passo temporale 1/60 di secondo di solito si ottiene una simulazione di alta qualità. Un altro parametro della funzione Step è il conteggio delle iterazioni, che controlla il numero di volte che il metodo risolutore dei vincoli cicla su tutti i contatti e i joint in tutto il mondo. Effettuare più iterazioni produce sempre una migliore simulazione, a scapito però delle performance. Valori standard prevedono una frequenza dello step di 60Hz con 10 iterazioni.

Il b2World è un contenitore per tutti gli oggetti della simulazione, e come tale ha a disposizione le liste di tutti gli oggetti che contiene tramite metodi come GetBodyList().

A volte si vuole determinare quali forme sono presenti in una determinata regione in una regione. La classe b2World ha a disposizione un metodo veloce (log(n)) per questa funzione che utilizza la struttura dati broad-phase. Per utilizzare tale metodo si fornisce un AABB in coordinate globali e un'implementazione di b2QueryCallback, così loggetto b2World chiama tramite il callback

la classe che lo implementa una volta per ogni AABB di un oggetto che si sovrappone con lAABB specificato in input.

Per controllare la presenza di forme su una linea si possono utilizzare i ray-cast. È possibile eseguire un ray-cast mediante l'implementazione di una classe di callback e fornendo il punto iniziale e finale della linea. La classe b2World chiama la classe che implementa la b2RayCastCallback, precedentemente creata, ad ogni oggetto incontrato dalla linea. La callback riceve informazioni riguardanti la forma, il punto di intersezione, il vettore normale sulla superficie della forma, e la distanza frazionaria lungo il raggio. Le forme che incrociano il raggio non chiamano la callback in ordine di distanza. È possibile controllare la prosecuzione del raggio in seguito alla chiamata di una callback restituendo un valore convenzionale. La restituzione di una valore pari a zero indica che il ray-cast deve fermarsi, mentre la restituzione di un valore paria a uno indica che il raggio deve continuare come se non avesse incontrato alcuna forma. La restituzione di un valore pari la valore del parametro fraction che la callback calcola automaticamente, fa in modo che il raggio si fermi al punto di intersezione corrente e trovi solo gli incroci con le forme precedenti. Così si può applicare un ray-cast che attraversi tutte le forme, che si fermi alla prima o che si fermi ad una forma specifica. A causa di errori di arrotondamento i ray-cast possono passare attraverso piccoli spazi tra i poligoni dellambiente statico se si utilizzano poligoni troppo piccoli.

Tramite b2World, è possibile applicare forze, momenti, impulsi ad un corpo, con il metodo ApplyForce(const b2Vec2& force, const b2Vec2& point). Quando si applica una forza o di un impulso, è necessario fornire il punto in cui viene applicato, e questo si traduce spesso in una momento attorno al centro di massa. Per trovare i punti corretti su cui applicare le forze, la classe b2Body ha alcune funzioni di utilità per aiutare a trasformare punti e vettori tra lo spazio locale di un corpo e globale.

4.2.3.2 Body

I body sono oggetti definiti nella classe b2Body e sono dotati di posizione e velocità. È possibile applicare su di essi forze, momenti, e impulsi. I body possono essere statici, cinematici, o dinamici a seconda del valore type impostato:

- b2_staticBody: Un corpo statico non si sposta sotto simulazione e si comporta come un corpo a massa infinita. Internamente, Box2D mantiene a zero il valore della massa, come valore convenzionale. I corpi statici possono essere spostati manualmente dall'utente cambiando le variabili di posizione. Un corpo statico ha velocità zero e non può collidere con altri corpi di tipo statico o cinematico.

- b2_kinematicBody: Un corpo cinematico si muove sotto simulazione secondo la sua velocità, senza rispondere alle forze che gli vengono applicate. Questo tipo di corpo, infatti, può essere spostato solo manualmente dall'utente, impostando la sua velocità o modificandone le variabili di posizione. Un corpo cinematico si comporta come se ha avesse massa infinita, tuttavia come per i corpi statici Box2D mantiene a zero il valore della massa. Similmente ai corpi staici, i corpi cinematici non collidono con quelli di tipo statico o cinematico.

- b2_dynamicBody: Un corpo dinamico è un corpo completamente simulato. I corpi di questo tipo possono essere spostati manualmente dall'utente, ma normalmente si muovono secondo forze e le collisioni della simulazione. Un corpo dinamico, infatti, può collidere con tutti i tipi di corpo ed ha sempre un valore di massa finito e diverso da zero. Se si tenta di impostare la massa di un corpo dinamico a zero, verrà automaticamente impostata a uno.

I corpi sono la base per applicare le shape definite nel pacchetto Collision alla simulazione. Attraverso le fixture, i body possono spostare e simulare le forme allinterno delloggetto b2World. I corpi sono sempre corpi rigidi in Box2D, cioè due forme appartenenti ad uno stesso corpo non possono muoversi uno in relazione allaltro.

Prima di creare un oggetto b2Body è necessario creare una definizione di corpo con la struttura b2BodyDef, che contiene i dati necessari per creare e inizializzare un corpo. Box2D copia i dati dalla definizione senza mantenere un puntatore alla struttura, quindi è possibile riciclare una definizione di corpo per creare più corpi uguali tra loro. Solitamente si mantiene un puntatore per ogni body creato, in questo modo è possibile leggere le posizioni del corpo per aggiornare le posizioni delle corrispondenti entità grafiche. Si mantengono i puntatori ai corpi in modo da poterli eliminare quando non servono più. Quando si crea una b2BodyDef si possono specificare alcuni parametri che definiscono alcune caratteristiche del corpo. Oltre al parametro type già visto, la definizione di corpo dà la possibilità di inizializzare la posizione del corpo alla sua creazione. Utilizzare questo parametro è importante perchè creando un corpo all'origine e spostandolo solo successivamente si rischia di sovrapporlo ad altri corpi creati con la stessa tecnica, degradando le prestazioni della simulazione.

Un corpo ha due principali punti di interesse. Il primo punto è l'origine del corpo. Le fixtures e i joint vengono attaccati al corpo rispetto alla sua origine. Il secondo punto di interesse è il centro di massa. Il centro di massa di un corpo è determinato dalla distribuzione di massa delle forme collegate ad esso, oppure è impostata in modo esplicito con b2MassData. Gran parte dei calcoli interni a Box2D utilizzano la posizione del centro di massa. Quando si crea la definizione del corpo, si può non sapere dove si trovi il centro di massa, pertanto si specifica la posizione del punto di origine. Si può anche specificare la rotazione del corpo in radianti, che non è influenzata dalla posizione del centro di massa. Se successivamente alla creazione si modifica la posizione del centro di massa del corpo, la posizione di origine non cambia e le forme e i joint collegati al corpo non si muovono.

Un altro parametro importante nella definizione di un corpo è quello relativo allo smorzamento, chiamato damping. Il damping viene utilizzato per ridurre progressivamente la velocità dei corpi, ma in maniera diversa dallattrito perché l'attrito si verifica solo con il contatto. Il damping non è un sostituto per dell'attrito e gli effetti dei due metodi devono essere usati assieme. I valori del parametro di smorzamento devono essere compresi tra 0 e infinito, con 0 che significa nessuno smorzamento, e infinito per smorzamento totale. Normalmente si utilizza un valore di smorzamento tra 0 e 0,1 per dare un effetto simile allattrito dellaria. Smorzamento è approssimata per la stabilità e le prestazioni. A piccoli valori di smorzamento l'effetto di smorzamento è in gran parte indipendente dallo step temporale, mentre con grandi valori di smorzamento, l'effetto

varierà con il variare del timestep, quindi è consigliato utilizzare un timestep costante.

Per migliorare le prestazioni, esiste un parametro chiamato allowSleep che indica la possibilità di un corpo di non prendere parte alla simulazione. Quando Box2D disattiva un corpo impostando a false il suo parametro isAwake, il corpo entra in uno stato di sospensione, che libera la CPU da molto carico. Se un corpo è attivo e si scontra con un corpo disattivato, allora il corpo disattivato si attiva. È inoltre possibile riattivare manualmente un corpo. Volendo creare un corpo che non partecipi alla simulazione dallinizio si può impostare il parametro active a false. Quando un corpo è in questo stato non viene attivato in casi di collisione con un altro corpo rimanendo totalmente fuori dalla simulazione fino a quando non viene riattivato manualmente.

I joint possono essere collegati a corpi inattivi, ma in questo modo non saranno simulati.

A volte può essere necessario che un corpo abbia un orientamento fisso. Tale corpo non deve quindi essere soggetto a rotazioni. È possibile utilizzare il parametro fixedRotation per ottenere questo effetto.

Come detto in precedenza, la simulazione genera una sequenza di stati che vengono calcolati uno dopo laltro secondo un certo intervallo temporale. Questo metodo è chiamato simulazione discreta. In una simulazione discreta, corpi rigidi possono spostarsi di molto durante un intervallo temporale. Se un motore fisico non tiene conto di questo fatto, è possibile che si verifichi leffetto tunneling a cui si è accennato in precedenza. Per evitare questo fenomeno, Box2D utilizza il rilevamento continuo di collisione (CCD) per impedire che corpi dinamici passino attraverso corpi statici. Il CCD simula il movimento dei corpi senza cambiarne la posizione, e cerca nuove collisioni calcolandone il tempo di impatto (TOI). Una volta effettuata la simulazione i corpi vengono spostati al primo TOI trovato. Normalmente il CCD non viene utilizzato tra corpi dinamici perché degraderebbe troppo le prestazioni. In alcuni scenari è necessario che alcuni corpi dinamici utlizzino il CCD, quindi è stata aggiunta la possibilità di aggiungere questa funzione ponendo a true il parametro bullet, che influisce solo sui corpi dinamici.

Infine, il parametro userData è utile per collegare gli oggetti di visualizzazione creati con librerie diverse ai rispettivi b2Body, essendo un generico puntatore a void.

Una volta inizializzata la struttura b2BodyDef il corpo può essere creato e distrutto con il metodo factory offerto dalla classe b2World che gestisce efficientemente la memoria e aggiunge automaticamente il corpo alla simulazione. Per creare corpi statici complessi è più veloce di collegare diverse forme in un unico corpo statico invece di creare diversi corpi statici dotati di una singola forma ciascuno.

Quando la simulazione finisce o comunque quando loggetto b2World deve essere eliminato, Box2D permette di evitare di eliminare i corpi manualmente, infatti b2World si occupa automaticamente della liberazione della memoria dei vari oggetti che utilizza per la simulazione. Tuttavia, è necessario annullare i puntatori ai b2Body che si utilizzavano durante la simulazione. Alleliminazione di un corpo tutti i joint e le fixture collegati vengono automaticamente distrutti. Ciò ha importanti implicazioni per il modo di gestire i puntatori alle shape e ai joint.

Dopo aver creato un corpo, ci sono molte operazioni che è possibile eseguire sul corpo come modificarne le proprietà e applicarci delle forze. Normalmente le proprietà di massa di un corpo vengono create automaticamente quando le forme sono aggiunte al corpo. È possibile comunque regolare la massa di un corpo in fase di esecuzione con il metodo SetMassData. Dopo aver impostato la massa di un corpo direttamente, si può reimpostare la massa naturale dettata dalle forme con il metodo ResetMassData. È possibile accedere anche alla posizione e alla rotazione di un corpo così come reimpostarle a piacimento tramite il metodo SetTransform. Infine, è possibile accedere alla posizione del centro di massa in coordinate locali e globali. Gran parte della simulazione interna in Box2D utilizza il centro di massa, tuttavia di norma non è necessario per accedervi, mentre di solito si lavora con il transform del corpo, ossia la coppia di vettori costituita dai vettori di posizione e rotazione.

4.2.3.3 Fixture

Box2D fornisce la classe b2Fixture per collegare le forme ai corpi. Una fixture può collegare una singola forma ad un corpo assegnando alla forma alcune proprietà che non sono necessariamente condivise con le altre forme che costituiscono il corpo. Come per i corpi, per creare una fixture è necessario prima inizializzare una struttura chiamata b2FixtureDef che può essere riutilizzata per la creazione di più fixture, e poi utilizzare un metodo factory offerto dalloggetto b2Body, chiamato CreateFixture. Quando si definisce una fixture si possono impostare quattro parametri che ne definiscono il comportamento:

- density: la densità è utilizzata per calcolare le proprietà di massa del corpo principale. La densità può essere zero o positiva. Si consiglia di utilizzare in genere un valore di densità simile per tutte le fixture.

- friction: lattrito è usato per fare in modo che gli oggetti scivolino gli uni sugli altri in maniera realistica. Box2D supporta sia lattrito statico che quello dinamico, ma utilizza lo stesso parametro per entrambi. Lattrito è simulato in modo accurato Box2D e la forza di attrito è proporzionale alla forza normale. Il parametro di attrito è solitamente impostato tra 0 e 1, ma può essere qualsiasi valore non negativo. Un valore di attrito 0 disattiva la forza di attrito e un valore di 1 rende lattrito molto forte. Quando la forza di attrito è calcolata tra due forme, Box2D combina i parametri di attrito delle due fixture cui appartengono calcolandone la media.

- restitution: la restituzione è usata per fare in modo che gli oggetti rimbalzino. Il valore di restituzione di solito è impostato per essere compreso tra 0 e 1. Un valore di zero corrisponde ad una collisione anelastica. Un valore di 1 darà luogo ad una collisione perfettamente elastica. Quando una forma è soggetta a più contatti, la restituzione viene simulata approsimativamente per evitare drastici cali di performance.

- filter: il filtro delle collisioni consente di evitare collisioni tra le fixture. Box2D implementa un filtro delle collisioni utilizzando categorie e gruppi: esistono 16 categorie di collisione , e per ogni fixture è possibile specificare a quali categorie appartiene e con quali altre categorie può entrare in

collisione tramite una semplice maschera a 16bit. I gruppi di collisione invece consentono di specificare un indice del gruppo al quale appartiene la fixture. Si possono avere gruppi di fixture che collidono tra loro, con indici positivi, o che non collidono tra loro, con indici negativi. Le collisioni tra i fixture di gruppi differenti vengono filtrati in base alle categorie, quindi i gruppi hanno precedenza sulle categorie. A volte potrebbe essere necessario cambiare i filtri dopo che una fixture è già stata creata, ciò è possibile tramite gli appositi metodi della classe b2Fixture, ma è bene tenere conto che le modifiche avranno effetto solo dallo step successivo.

Una volta creata una fixture con una shape possibile contrassegnarla come sensore tramite una sua proprietà. I sensori, che possono essere statici o dinamici, registrano le sovrapposizioni delle forme senza generare punti di contatto, e possono essere applicati ai corpi come ogni altra fixture.

4.2.3.4 Joint

I joint vengono utilizzati per vincolare corpi allambiente o per vincolarli luno all'altro, e possono essere combinati tra loro in molti modi diversi per creare movimenti più complessi.

Ogni tipo di joint ha una struttura per la sua definizione che deriva da b2JointDef. Tutti i joint costituiscono un collegamento tra due corpi diversi di qualsiasi tipo, ma generalmente almeno uno dei due è dinamico, altirmenti il joint non avrebbe alcun effetto. È possibile specificare per qualsiasi tipo di joint un oggetto grafico corrispondente nella variabile userData, ed è possibile evitare automaticamente che i corpi attaccati collidano luno con laltro, semplicemente specificando il parametro collideConnected. Molte definizioni di joint richiedono di fornire alcuni dati geometrici: spesso un joint sarà definito da punti di ancoraggio, cioè i punti di collegamento con i corpi annessi. Box2D richiede che questi punti siano specificati in coordinate locali, perché in questo modo il joint può essere specificato anche quando la posizione iniziale del corpo viola un vincolo del joint. Inoltre, alcuni joint necessitano di conoscere l'angolo relativo iniziale che separa i due corpi, in modo che possano applicare correttamente eventuali vincoli nella rotazione. Il resto dei dati di definizione dipendono dal tipo di joint.

Come i body, i joint vengono creati e distrutti utilizzando i metodi factory offerti da b2World che, come nel caso dei corpi rigidi, necessitano di una struttura di definizione. Diversamente dai corpi rigidi però, esistono vari tipi di joint, ma tutti derivano dalla stessa classe b2joint, ed è quindi sufficiente crearli con il metodo CreateJoint che riceve come arametro una generica struttura b2JointDef.

Una volta creati, i joint sono simulati autonomamente, ma è possibile in ogni istante leggerne i valori di posizione e le forze ed i momenti che intervengono su di essi. È possibile quindi utilizzare le forze di reazione ed i momenti per innescare particolari eventi.

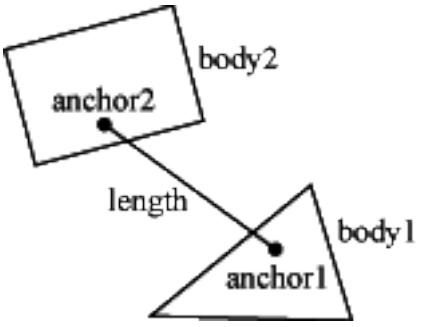


Figura 4.11:

Distance Joint Uno dei più semplici joint è quello di distanza, che impone che la distanza tra due punti su due corpi sia costante. Quando si specifica una distanza i due corpi dovrebbero già essere al loro posto, quindi si specificano i punti di ancoraggio in coordinate assolute. Il primo punto di ancoraggio è collegata al corpo 1, ed il secondo punto di ancoraggio è collegato al corpo 2. Questi punti implicano la lunghezza del vincolo di distanza, che non viene specificata con un parametro. Il joint di distanza può essere anche reso flessibile per creare un effetto molla, specificando i due parametri di frequenza e smorzamento. La frequenza è come la frequenza di un oscillatore armonico ed è specificata in Hertz. Tipicamente la frequenza dovrebbe essere inferiore alla metà della frequenza dello step di simulazione. Il valore di smorzamento è un valore adimensionale ed è compreso tra 0 e 1 e rappresenta lo smorzamento delloscillazione. Con un valore di smorzamento a 1 lo smorzamento delloscillazione è istantaneo.

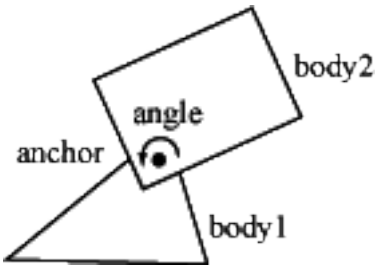


Figura 4.12:

Revolute Joint Un joint di rotazione, detto revolute joint, forza i due corpi a condividere un comune punto di ancoraggio. Il joint ha un solo grado di libertà: la rotazione relativa dei due corpi, chiamato joint angle. Per specificare un joint di rotazione è necessario fornire due corpi e un punto di ancoraggio singolo espresso nello spazio globale. La funzione di inizializzazione presuppone che i corpi siano già nella posizione corretta. Per convenzione l'angolo di rotazione è zero quando il joint viene creato, indipendentemente dalla rotazione corrente dei due corpi, mentre è positivo quando il secondo corpo ruota in senso antiorario attorno al punto di ancoraggio.

In alcuni casi si potrebbe voler controllare langolo di rotazione, per questo il revolute joint può, facoltativamente, simulare un limite massimo di rotazione comune e o un motore di rotazione. Un limite forza il langolo di rotazione tra

un limite inferiore e superiore, che deve includere lo zero, altrimenti causerebbe errori allinizio della simulazione. Un motore consente di specificare la velocità di rotazione, che può essere negativa o positiva, e una forza massima che può essere applicata ai corpi per raggiungere tale velocità. Il motore, raggiunta tale velocità, la mantiene fino a quando la coppia massima richiesta per mantenerla non eccede il limite massimo. Quando la coppia massima viene superata, la rotazione rallenta e può arrivare a invertire il proprio verso. È possibile utilizzare un motore anche per simulare l'attrito della rotazione, infatti basta impostare la velocità zero e impostare la coppia massima ad un valore piccolo, ma significativo. Il motore cercherà così di impedire la rotazione.

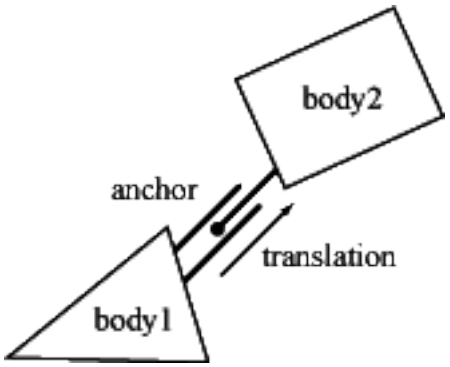


Figura 4.13:

Prismatic Joint Un prismatic joint consente la traslazione relativa dei due corpi lungo un asse specificato. Un prismatic joint impedisce la rotazione relativa, quindi ha un solo grado di libertà. La definizione è simile alla precedente, sostituendo la traslazione con l'angolo e la forza con il momento. Usando questa analogia è facile capire come funzionano in questo caso i parametri di limite, di velocità e di forza massima. In questo caso invece di specificare langolo di rotazione relativo iniziale si specifica un asse di traslazione iniziale lungo il quale potranno scorrere i due corpi. Come nel revolute joint, la traslazione iniziale è pari a zero quando il joint viene creato quindi è importante specificare i limiti massimo e minimo in modo che comprendano lo zero.

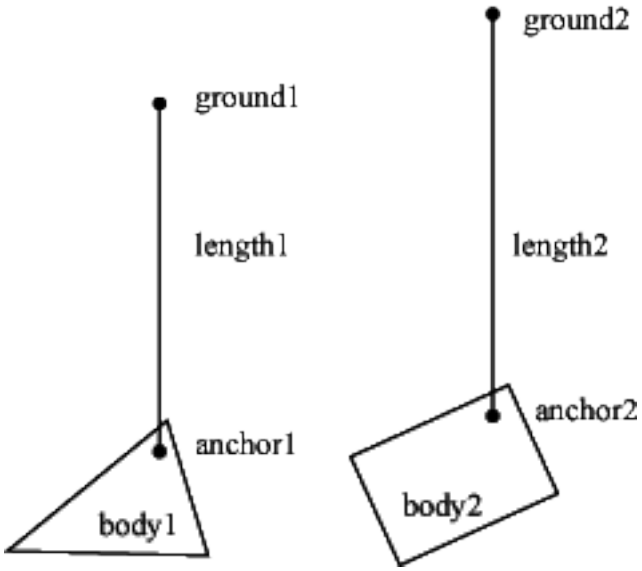


Figura 4.14:

Pulley Joint Un pulley joint viene utilizzato per creare una funzione simile a quella della carrucola. Il joint collega due corpi allo sfondo e indirettamente l'uno con l'altro. Al salire di un corpo l'altro scende mantenendo costante la somma delle distanze tra i punti di ancoraggio dei corpi ed i punti di collegamento con lo sfondo. È possibile fare in modo che la costante non sia la somma delle due distanze ma che sia ottenuta sommando la prima distanza alla seconda moltiplicata per un rapporto definibile tramite un parametro, detto ratio. Questo rapporto fai in modo che una delle distanze della carrucola si estenda più velocemente dell'altra. Al tempo stesso la forza di un vincolo sarà inferiore su un lato rispetto all'altro. I pulley joint può creare problemi quando un lato è completamente esteso, infatti il lato opposto avrà lunghezza zero creando problemi nelle equazioni di vincolo. È necessario configurare le forme per evitare che ciò accada.

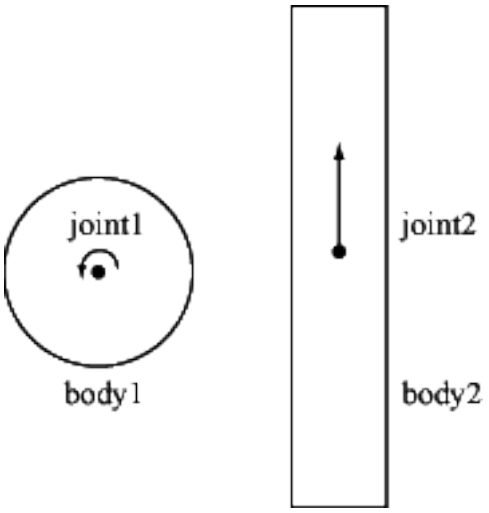


Figura 4.15:

Gear Joint Se si desidera simulare il funzionamento degli ingranaggi, in linea di principio è possibile crearli utilizzando dei corpi con forme che, composte, costituiscono degli ingranaggi. Questo approccio non è molto efficiente e potrebbe essere lungo da implementare. Per questo motivo è stato creato il gear joint. In un gear joint è possibile collegare tra loro revolute e prismatic joint, anche specificando un rapporto di trasmissione, che può essere anche negativo. È importante ricordarsi di eliminare sempre i gear joint prima dei joint che collegano, perché in caso contrario, lapplicazione si arresterebbe a causa di puntatori orfani. Si dovrebbe anche eliminare il gear joint prima di eliminare i corpi interessati.

Mouse Joint Il mouse joint viene utilizzato per manipolare i corpi con il mouse, tentando di guidare un punto del corpo verso la posizione corrente del cursore senza imporre restrizioni sulla rotazione. La definizione del mouse joint ha un target, una forza massima, una frequenza e un valore di smorzamento. Il punto target coincide inizialmente con il punto di ancoraggio del corpo. La forza massima è usata per prevenire reazioni violente quando più corpi dinamici interagiscono, mentre la frequenza e il valore di smorzamento sono utilizzati per creare un effetto simile a quello del distance joint.

Rope Joint Il rope joint limita la distanza massima tra due punti. Ciò può essere utile per prevenire che catene di corpi collegati tra loro si stirino, anche sotto potenti forze che tendono a dividerli.

Friction Joint Il friction joint è utilizzato per simulare lattrito in tutte le sfaccettature, dallattrito nelle traslazioni all'attrito angolare.

4.3 SPPhysics - libreria di integrazione

Prima di progettare una libreria che integri la libreria Box2D allinterno del framework Sparrow è necessario sapere come utilizzare congiuntamente le due

librerie. Uno degli oggetti indispensabili da utilizzare è SPView, cioè la view nella quale Sparrow disegna lapplicazione. La SPView è collegata ad uno stage e gestisce la visualizzazione sul dispositivo di quello che loggetto SPStage renderizza. Loggetto SPStage è la radice del display tree e rappresenta la rendering area dellapplicazione e ne gestisce il loop. Quando si crea una applicazione con Sparrow è necessario innanzitutto creare una classe di base, detta Game.h, come sottoclasse di SPStage, e collegarla ad un oggetto SPView per la visualizzazione del contenuto del display tree. Allinterno di questa classe va inserito loggetto b2World di Box2D. Come visto in precendeza, loggetto b2World contiene tutti i body e i joint e gestisce tutta la simulazione della fisica. I vari oggetti di simulazione della fisica sono collegati agli oggetti grafici di Sparrow con le funzioni Get/SetUserData(). Infatti con queste funzioni si possono chiamare sui body e i joint creati nel b2World per collegarli ai vari sprite inseriti nello stage, così da muoverli allunisono frame per frame, visualizzarli e eliminarli.

È importante tenere conto del fatto che mentre lorientamento del sistema di coordinate di Box2D equivale a quello del normale sistema di assi cartesiano, nel sistema di riferimento di Sparrow il verso dellasse y è invertito. È importante quindi fare attenzione a convertire le componenti dei movimenti e delle posizioni sullasse y.

Unaltra conversione che è necessario fare quando si utilizzano congiuntamente le due librerie è quella relativa alle unità di misura. Mentre Box2D utilizza i metri, Sparrow utilizza i punti (indirettamente i pixel) ed è necessario quindi mantenere una costante di conversione, nel nostro caso chiamata PTM_RATIO.

Infine utilizzare separatamente le librerie comporta la necessità di inizalizzare a mano tutti i dati geometrici dei corpi, dei joint e delle fixture. Uno dei vantaggi di utilizzare una integrazione della libreria è anche quello di snellire la creazione di oggetti di uso comune con valori di default. È comunque importante adattare i parametri ai casi specifici di ogni applicazione per rendere la simulazione più robusta e realistica.

La libreria di integrazione, chiamata SPPhysics, è strutturata in tre classe principali, più altre classi secondarie che aggiungono funzionalità particolari, organizzate come in figura.

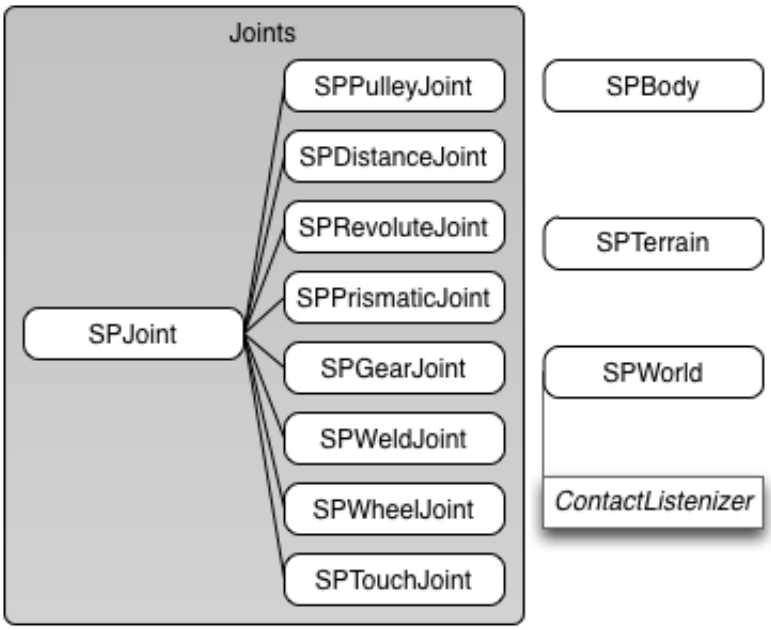


Figura 4.16:

4.3.1 SPWorld

La classe SPWorld è una classe pensata per funzionare come ambiente di base per una applicazione che integra in Sparrow la simulazione della fisica. SPWorld è sottoclasse di SPSprite, quindi può essere usata come un normalissimo sprite, ma è dotata di alcune funzioni aggiuntive.

Innanzitutto la classe contiene al suo interno un puntatore ad un oggetto b2World, necessario per la simulazione dellambiente fisico. La classe definisce anche una costante molto importante per poter nascondere a chi utilizza la libreria il fatto che Box2D utilizza come unità di misura i metri: PTM_RATIO, ossia il rapporto pixel su metro, impostato di default a 32 che è usato molto spesso internamente. Le altre proprietà della classe sono quelle relative al vettore di gravità, che specifica direzione e verso della forza di gravità, e ai parametri necessari per definire quante iterazioni della simulazione eseguire ad ogni frame di Sparrow (di default, 1).

Essendo sottoclasse di SPSprite, la classe SPWorld può ridefinire il metodo update, che viene eseguito ad ogni frame di Sparrow, e lo fa per chiamare ad ogni frame lesecuzione di uno step della simulazione della fisica.

Oltre ad estendere SPSprite, la classe implementa un protocollo chiamato ContactListenizer. Il protocollo ContactListenizer non è altro che una interfaccia che definisce labilità di ricevere gli eventi legati ai contatti, ossia collisione, separazione, sovrapposizione. Affinchè una classe implementi questo protocollo è sufficiente che siano definiti tre metodi: onOverlapBody, onSeparateBody, onCollideBody. Per ricevere gli eventi legati ai contatti, la classe SPWorld contiene al suo interno un oggetto chiamato ContactRelay. Questo oggetto implementa la classe astratta b2ContactListener di Box2D ed è loggetto che riceve gli eventi legati ai contatti avvenuti nella simulazione e li notifica ad una qualsiasi classe

che implementi il protocollo ContactListenizer, e ad i corpi protagonisti della collisione. Oltre a notificare levento, il ContactRelay calcola le forze normali e di frizione da applicare ai corpi che sono entrati in contatto, e le notifica loro, che al prossimo ciclo di esecuzione potranno applicarle su se stessi. In questo modo è possibile creare un collegamento tra la simulazione di Box2D e la classe SPWorld, che implementando il protocollo è in grado di ricevere gli eventi dal relay. Estendendo la classe SPWorld è possibile sovrascrivere i metodi relativi alla collisione per definire il comportamento desiderato nella propria applicazione.

Lutilizzo di SPWorld è piuttosto semplice e totalmente trasparente per chi è abituato ad usare un semplice SPSprite. Una volta allocata la memoria ed inizializzato loggetto si può impostare un vettore gravità a piacimento, o semplicemente lasciare il valore di default. Successivamente si può utilizzare loggetto come un SPSprite, cioè aggiungendo un qualsiasi oggetto grafico, ossia un oggetto che deriva da SPDisplayObject, con il metodo addChild. Il fatto di aggiungere un oggetto dotato di simulazione della fisica o meno è totalmente trasparente alla classe SPWorld.

4.3.2 SPBody

La decisione di scegliere la classe SPSprite come superclasse per SPWorld è naturale visto che, semanticamente, SPWorld è per definizione un contenitore di oggetti e quindi è un oggetto affine a SPSprite che a sua volta è per definizione un contenitore di oggetti grafici.

Nel caso di SPBody la scelta non è così naturale e ci possono essere più soluzioni possibili:

- Un SPBody come oggetto derivato da SPImage, in modo da poter caricare una immagine e fare in modo che si muova secondo la fisica. Questo approccio implica che un qualsiasi SPBody sia dotato di una componente grafica, essendo un particolare tipo di SPImage, ma un corpo deve poter essere simulato anche senza la controparte grafica, ad esempio per creare i muri invisibili che circondano la schermata per fare in modo che gli oggetti non escano dalla visuale.

- Un SPBody come oggetto derivato da un SPDisplayObject può essere una soluzione più ragionevole, perché dà la possibilità di estendere a piacimento una classe astratta che definisce un oggetto visualizzabile che è una delle proprietà che SPBody deve avere, mentre le altre proprietà possono essere aggiunte a piacimento.

- Un SPBody come oggetto derivato da un SPDisplayObjectContainer è probabilmente la soluzione semanticamente più corretta. Infatti, come visto precedentemente, un corpo di Box2D può essere composto da più forme che possono anche essere aggiunte e rimosse durante lesecuzione dellapplicazione. Quindi, un SPBody non è semplicemente un singolo oggetto ma può essere la composizione di più oggetti, sia fisici che grafici. la possibilità di essere composto da più oggetti fisici è data dal b2Body di Box2D, mentre la possibilità di contenere più oggetti grafici è data dalla natura della classe SPDisplayObjectContainer.

Un oggetto SPBody contiene un oggetto b2Body e tutte le proprietà corrispondenti alle variabili interne del b2Body. È importante che le proprietà delloggetto

SPBody siano mantenute in sincronia con quelle delloggetto b2Body in ogni metodo della classe. Replicare le variabili può sembrare uno spreco di memoria ma ha una ragione ben precisa: un SPBody può essere creato e inizializzato anche in assenza di un SPWorld, o quando un SPWorld non è ancora stato creato, e dato che un oggetto b2Body può essere creato solamente con il metodo factory di un b2World (non volendo gestire manualmente la memoria) è necessario mantenere i valori di inizializzazione fino alla effettiva creazione del corpo. La creazione delloggetto b2Body avviene solamente quando un oggetto SPBody viene aggiunto ad un SPWorld con il generico metodo addChild, viene quindi aggiunto come un qualsiasi altro oggetto di visualizzazione. Quando un SPBody viene inizializzato si registra come listener per due eventi:

- SP_EVENT_TYPE_ADDED, lanciato quando un oggetto di visualizzazione viene aggiunto ad un altro oggetto come figlio.
- SP_EVENT_TYPE_REMOVED, lanciato quando un oggetto di visualizzazione viene rimosso da un oggetto padre.

Con questo meccanismo, quando un SPBody viene aggiunto ad un SPWorld può ottenere un riferimento a tale oggetto e può eseguire del codice di inizializzazione delle proprie variabili. Di conseguenza, loggetto può creare una struttura di definizione di un corpo ed utilizzare il riferimento alloggetto SPWorld per creare un b2Body con il metodo apposito offerto dalloggetto b2World.

Essendo un contenitore di oggetti, è possibile aggiungere al SPBody oggetti grafici, oggetti fisici e oggetti dotati di grafica e fisica contemporaneamente, sia prima che dopo il collegamento al SPWorld:

- Per aggiungere un oggetto grafico è sufficiente utilizzare il metodo ereditato dalla superclasse, addChild, ed il comportamento sarà esattamente lo stesso.
- Per aggiungere un oggetto fisico è possibile utilizzare addBoxWithName, che permette di aggiungere al body una shape quadrata specificandone le dimensioni in punti, che saranno trasformate automaticamente in metri dalla libreria. In alternativa è possibile utilizzare addCircleWithName, che permette di aggiungere al body una shape circolare specificandone le dimensioni del raggio. Infine è possibile utilizzare addPolygonWithName per aggiungere al body una shape poligonale specificando un vettore di vertici, in ordine antiorario, che definiscono la forma del poligono da aggiungere.
- Per aggiungere un oggetto sia fisico che grafico esistono le combinazioni dei metodi visti in precedenza, cioè addChild:withBoxNamed, addChild:withCircleNamed e addChild:withPolygonNamed. Nei primi due casi, se non viene specificata la dimensione della forma che si vuole aggiungere essa può essere calcolata automaticamente dal metodo in base alle dimensioni delloggetto grafico che si sta aggiungendo. Nel terzo caso invece è necessario specificare i vertici manualmente. Futuri sviluppi della libreria potrebbero comprendere la generazione automatica dei poligoni che costituiscono il profilo di un oggetto grafico.

Se questi metodi vengono utilizzati prima dellentrata in un SPWorld, le forme vengono aggiunte in una apposita lista di b2ShapeData, realizzata con un

dizionario. Allaggiunta del SPBody ad un SPWorld, viene creato il b2Body e vengono create ed aggiunte tutte le b2Shape ed i rispettivi puntatori salvati in una lista, realizzata anchessa con un dizionario che utilizza come chiave il nome assegnato in fase di definizione. Un SPBody ha dei valori di default per density, friction e bounce, ma è comunque possibile definire valori diversi per ogni shape tramite gli appositi metodi.

Infine, una volta avviata la simulazione, è possibile agire sui corpi applicando forze e momenti con i metodi applyForce e applyTorque e le loro varianti. Per comodità, ma senza perdita di generalità, il punto di pivot di un SPBody è posto al centro delloggetto grafico, ricalcando la convenzione utilizzata in Box2D, mentre negli oggetti grafici di Sparrow di default lorigine è posta nellangolo in alto a sinistra. Lorientamento degli assi cartesiani invece rispetta la convenzione utilizzata da Sparrow, perché la libreria è stata progettata per essere utilizzata con il minor sforzo possibile allinterno di questo framework.

4.3.3 SPJoint

Come visto in precedenza, i joint sono degli elementi che uniscono e simulano linterazione tra due body. Teoricamente non hanno bisogno di essere visualizzati come gli SPBody, quindi potrebbe essere sensato derivare la classe da SPEvent-Dispatcher, per avere la possibilità di lanciare eventi. Daltra parte però è necessario che loggetto SPJoint possa appartenere al display tree quindi deve per forza discendere dalla classe SPDisplayObject, altrimenti sarebbe impossibile ricavare lSPWorld al quale il joint si riferisce senza comunicarlo esplicitamente. Quindi si è deciso di estendere uno degli elementi grafici più semplici e che è sempre un nodo foglia nei display tree, SPImage.

Ogni joint presente in Box2D è definito in una classe dedicata che deriva da b2joint. E bene replicare questa struttura anche nel codice di SPPhysics. La classe SPJoint corrisponderà quindi a b2joint e da essa deriveranno le classi dei joint particolari. La classe b2joint contiene i puntatori ai 2 corpi che unisce e una variabile che stabilisce se i due body devono collidere tra loro. I valori particolari di ogni joint sono invece di 3 tipi:

- Punti di ancoraggio
- Limiti del joint
- Motori del joint

Una volta create queste variabili devono essere modificabili dallesterno in modo da poter variare la scena durante lesecuzione, quindi è bene esporre tutte queste variabili con delle proprietà.

Un generico SPJoint viene utilizzato in maniera simile ad un SPBody: si crea con lapposita funzione, si inizializzano i valori specifici della scena che si sta creando, curandosi di specificare i due SPBody collegati dal joint, e lo si aggiunge al SPWorld come un semplice oggetto di visualizzazione. Internamente, le sottoclassi dei joint specifici si curano di inizializzare solamente la parte specifica della propria classe. Quando loggetto viene aggiunto al display tree viene il metodo della classe specifica crea la struttura di definizione e la inizializza nelle parti di sua competenza, dopodichè chiama il metodo della superclasse che si occupa di inizializzare la struttura nelle parti comuni a tutti i tipi di joint e di

creare il joint con il metodo factory offerto dallSPWorld. Infatti, allaggiunta del joint al display tree, loggetto SPJoint può ottenere un puntatore del nodo padre a cui è collegato, in questo caso proprio loggetto SPWorld. Una volta creato il joint ognuno dei body collegati salva internamente un riferimento al joint con il metodo addJoint, perché, in caso di eliminazione del body, è importante che il joint sia eliminato per primo, altrimenti causerebbe un errore irreparabile nellapplicazione. Di conseguenza, quando si elimina un SPJoint, non bisogna mai eliminare i body che collega prima di eliminare il joint stesso.

I joint di Box2D sono stati riproposti in tutte le loro funzionalità nella libreria SPPhysics, ma il MouseJoint, data lassenza di un mouse nei dispositivi mobili, è stato adattato allutilizzo con il tocco delle dita sullo schermo.

4.3.4 SPTerrain

La classe SPTerrain, come suggerito dal nome, è stata creata per simulare i terreni irregolari con i quali possono intergaire gli oggetti fisici. Internamente la classe gestisce un b2Body come la classe SPBody, ma in maniera semplificata e limitata alle funzioni necessarie ad un corpo statico. Infatti lunico tipo di forma che è possibile aggiungere a questo oggetto è una b2ChainShape che, come visto in precedenza, può essere utilizzata solo in corpi statici. Delle proprietà è stata mantenuta la proprietà di attrito che è particolarmente importante per le funzioni di questo oggetto.

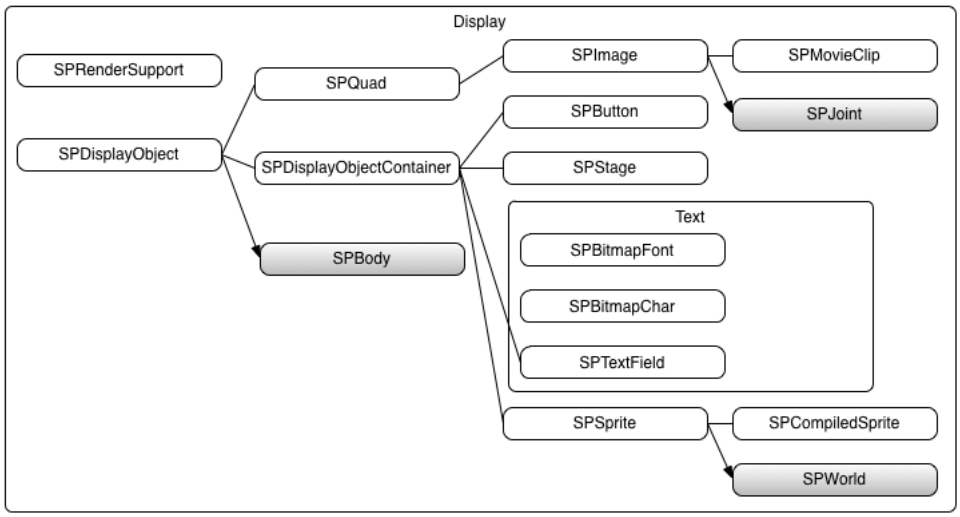


Figura 4.17:

Capitolo 5

Testing ed esempi d'uso

La libreria SPPhysics è stata testata in ogni sua classe fin dallinizio dello sviluppo. Non sono stati richieste particolari misurazioni delle prestazioni ma solamente di verificare che in semplici scenari di utilizzo la libreria non degradi le prestazioni dellapplicazione. Di seguito è presentato del codice di esempio commentato, per mostrare la semplicità di utilizzo della libreria.

5.1 Creare un world

Per crere una prima, semplice, applicazione che utilizzi SPPhysics, è necessario innanzitutto partire dal codice di base fornito da Sparrow per la creazione di una applicazione. Il codice è strutturato in tre classi principali:

- Un "main.m", che avvia il controller dell'applicazione.
- Un oggetto AppDelegate, che costituisce il controller dell'applicazione, ossia gestisce il lancio, la chiusura, l'attivazione e la disattivazione dell'applicazione. Il controller gestisce anche la creazione dell'oggetto SPView, dedicato alla renderizzazione dei contenuti, e la creazione di un oggetto Game.
- Un oggetto Game, derivante da SPStage, gestisce tutto il contenuto dell'applicazione.

Nel metodo di inizializzazione di questo oggetto Game è possibile creare tutti gli elementi grafici che popoleranno la SPView, definendone anche il comportamento. Essendo SPWorld un oggetto di visualizzazione deve essere creato all'interno di questo metodo.

Per creare l'oggetto è sufficiente procedere come con un semplice SPSrite:

```
SPWorld* world = [[SPWorld alloc] init];
```

Una volta creato l'oggetto, è necessario specificare il vettore di gravità che influenzerà gli oggetti inseriti nell'ambiente di simulazione, ricordando che l'orientamento degli assi cartesiani negli oggetti della libreria rispetta la convenzione utilizzata in Sparrow.

```
world.gravity = CGPointMake(0, 9.8 * PTM_RATIO);
```

Impostata la gravità, l'oggetto è pronto per esseree "riempito" di oggetti fisici.

(...)

È importante, infine, ricordare di liberare la memoria occupata dall'oggetto alla chiusura dell'applicazione.

```
[world dealloc];
```

Il metodo dealloc non deve essere chiamato all'interno del codice di inizializzazione bensì nel metodo di dealloc dell'oggetto Game, ricordando di chiamare successivamente il dealloc del genitore per non causare un memory leak.

```
[super dealloc];
```

5.2 Creare un body

Per creare un SPBody è sufficiente utilizzare il metodo fornito dalla classe, senza occuparsi di allocare manualmente spazio di memoria.

```
SPBody* box = [SPBody body];
```

Una volta creato l'oggetto SPBody è necessario aggiungere del contenuto a tale oggetto. Supponendo di avere a disposizione tra le risorse un file immagine di un quadrato colorato chiamato "box_square.png" è necessario caricarlo in un oggetto di visualizzazione di tipo SPImage con un semplice metodo di inizializzazione.

```
SPImage *boxImage = [SPImage imageWithContentsOfFile:@"box_square.png"];
```

Successivamente è possibile aggiungere l'immagine appena creata all'oggetto SPBody, creando contemporaneamente la forma fisica corrispondente ad esso, senza dover specificare manualmente dimensioni o parametri del quadrato, ma in maniera completamente automatica.

```
[box addChild:boxImage withBoxNamed:@"Box"];
```

Infine, affinchè il quadrato sia visualizzato, è necessario aggiungerlo all'ambiente di simulazione dell'oggetto SPWorld precedentemente creato. È bene ricordare che l'oggetto fisico corrispondente al quadrato viene creato solamente quando il quadrato viene aggiunto al world, quindi prima di tale evento è impossibile accedervi.

```
[world addChild:box];
```

Essendo la memoria gestita internamente alla libreria, non è necessario gestire leliminazione del body.

5.3 Creare un joint

Per creare un SPJoint tra due body, ad esempio un SPRevoluteJoint, bisogna innanzitutto creare l'oggetto con l'apposito metodo fornito dalla classe. Come per i body, non è necessario allocare manualmente lo spazio di memoria.

```
SPRevoluteJoint* joint = [SPRevoluteJoint revoluteJoint];
```

Supponendo di aver già creato due SPBody, box1 e box2, come nell'esempio precedente e di averli posizionati alle giuste coordinate per essere collegati, è necessario assegnarli al joint.

```
joint.bodyA = box1;
joint.bodyB = box2;
```

Una volta assegnati i due body è necessario impostare i punti di ancoraggio. Di default l'origine delle coordinate è posta al centro dell'oggetto e rispetto a tale origine è necessario specificare i punti di ancoraggio del joint in coordinate locali secondo l'orientamento degli assi di Sparrow. Con i dati in esempio il punto di ancoraggi osarà impostato nell'angolo in basso a destra del quadrato.

```
joint.localAnchorA = CGPointMake(box1.width/2, box1.height/2);
```

Assegnando questi valori al punto di ancoraggio del secondo body, sarà impostato al centro.

```
joint.localAnchorB = CGPointMake(0, 0);
```

Opzionalmente è possibile attivare un motore, in questo caso di rotazione, e impostarne la velocità di rotazione da raggiungere in radianti al secondo e la coppia massima del motore.

```
joint.enableMotor = YES;
joint.motorSpeed = SP_D2R(360);
joint.maxMotorTorque = 30;
```

Infine, come per ogni altro oggetto di visualizzazione, è necessario aggiungere il joint come oggetto figlio di SPWorld.

```
[world addChild:joint];
```

Essendo la memoria gestita internamente alla libreria, non è necessario gestire leliminazione del joint.

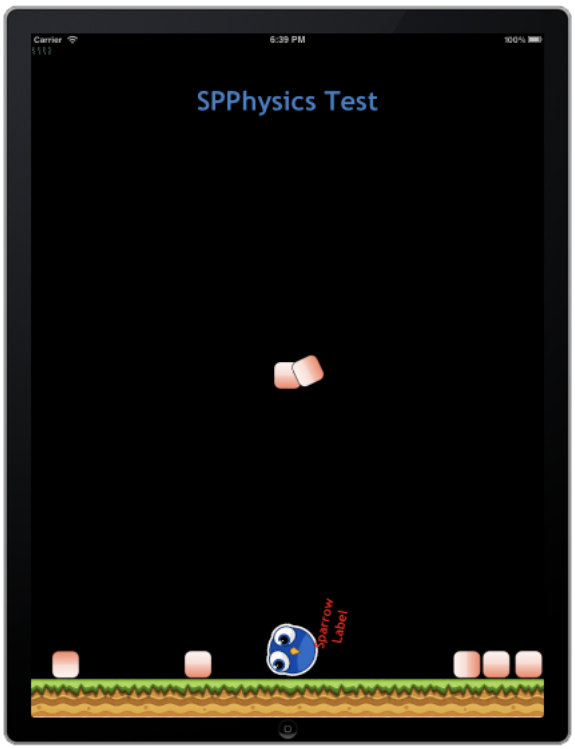


Figura 5.1:

Capitolo 6

Conclusioni

Come verificato nella fase di testing, la libreria funziona correttamente, e tutte le funzionalità offerte dal motore fisico Box2D e necessarie al framework APP-KID sono state implementate. In futuro, in seguito allutilizzo in qualche progetto, sarà possibile rendere lutilizzo della libreria ancora più semplice adattandola al workflow di APP-KID e in base alle esigenze che emergeranno. Inoltre è possibile aggiungere una funzione di riconoscimento automatico delle forme dalle immagini per snellire ulteriormente il processo di produzione.

Con SPPhysics, lazienda ha aggiunto un asset interessante al proprio framework di sviluppo che può rendere più interessanti e divertenti le applicazioni e fornire spunti per idee più originali nellambito dei mini-giochi inseriti.

Infine, durante il tirocinio, lo studente ha acquisito molte competenze che spaziono anche al di fuori del singolo progetto realizzato: una conoscenza del linguaggio di programmazione Objective-C, della piattaforma di sviluppo iOS, del framework Sparrow, e la capacità di gestire un progetto di discrete dimensioni, di curare la progettazione del software, limplementazione ed il testing.

Bibliografia

[1] Stevenson S. (2010), Cocoa and Objective-C: up and running, OReilly, Sebastopol.

[2] Zdziarski J. (2008), Iphone Open Application Development, OReilly, Sebastopol.

[3] Apple Inc. (2012), iOS Technology Review, Apple Inc., Cupertino.

[4] Sparrow Framework Reference, Sparrow Framework, 2012, http://-doc.sparrow-framework.org .

[5] Box2D v2.2.0 User Manual, Box2D, 2012, http://www.box2d.org/manual.html .