まえがき

本書は、cocos2d-xを利用してiOSやAndroidの2Dゲームを作成したいと思っている人のためのレシピ本です。
特に、iOSとAndroidのアプリを同時に作成するクロスプラットフォーム開発を検討されている方にとっては、手放すことのできない一冊になるでしょう。
これまでcocos2d-xを学んでこられた方は、cocos2d-xだけで画面をデザインしたり、アニメーションを構成したり、物理エンジン用シェイプを作成したりしていたのではないでしょうか。
一般的にcocos2d-xだけでも2Dゲームを作ることができますが、本書では開発の手助けとなる便利なツールやクラスを紹介しているので、より簡単にゲームを作成することができるようになります。cocos2d-xでは、多くのツールをサポートしているので、それらを組み合わせてゲームを作成すると、より良いものができるでしょう。

本書は、cocos2d-xの基本を学んだことがある方を対象としています。とはいえ、全く知らない方のためにも必要最小限の説明は行っているので、本書のみでもcocos2d-xに触れることはできると思います。しかし、cocos2d-xの基本について詳しく学びたい場合は、cocos2d-xの入門書と併せて読むことをオススメします。
また、本書を読み終わった後、さらにcocos2d-xを学ぶための情報も載せているので、これをステップとしてcocos2d-xのスペシャリストになってください。cocos2d-xはオープンソースですので、cocos2d-xやC++に長けた人のコードを見て学ぶこともできるでしょう。いろいろな視点からcocos2d-xに接してみるといいでしょう。

2013年5月

清水 友晶

目　次　Contents

Chapter 5
実践的な画像表現

Chapter 6
ネイティブ間連携

Chapter 7
cocos2d-xをもっと知るために

Chapter 1

cocos2d-x について

1-1 cocos2d-x の概要

ここではcocos2d-xの歴史や簡単な構成を紹介します。cocos2d-xの基本的な情報を知ることで、今後のスムーズな開発が可能になるでしょう。

1-1-1 cocos2d-xの特徴

cocos2d-xは2Dゲームフレームワーク

cocos2d-xは2Dゲームフレームワーク――すなわち、2Dゲームを作成するために必要な機能が詰め込まれたアプリケーションの土台となるソフトウェアです。簡単に説明すると、キャラクターを表示する機能やアニメーションを行う機能などの集合体です。またC++では厄介なメモリ管理を意識しなくてもいい仕組みや、ゲームに限らずプログラミングする上で便利な機能やマクロなども含まれています。

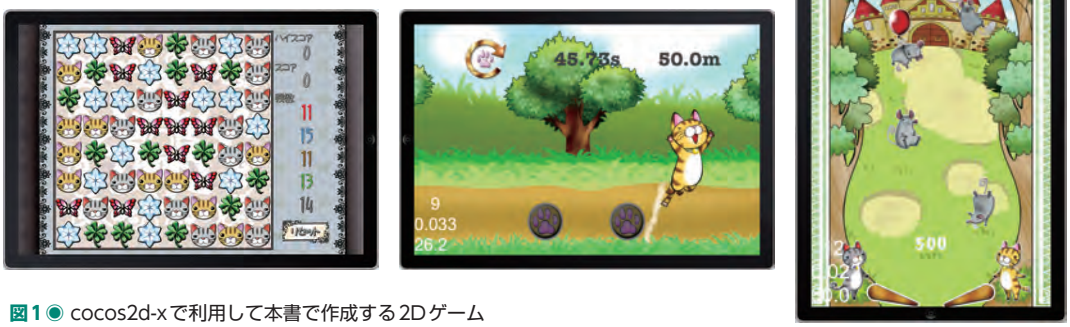

図1◉ cocos2d-xで利用して本書で作成する2Dゲーム

マルチプラットフォーム開発

cocos2d-xの最大の特徴といえば、クロスプラットフォーム開発が行える事です。iOSとAndroidはもちろんのこと、その他にも多くのプラットフォームをサポートしています。本章ではiOSとAndroidのクロスプラットフォーム開発の方法を紹介しますが、どのようにしてクロスプラッ

トフォーム開発を行うのでしょうか？

たとえば3Dゲームツールとして代表的なUnity 3Dでは、専用のツールによりマルチプラットフォーム開発を行います。cocos2d-xはフレームワークですので専用のツールはなく、iOSはXcodeで、AndroidはEclipseで開発を行います。ソースコードや画像などのリソースは特定のディレクトリに配置し、それぞれのプロジェクトから共通で利用されます。

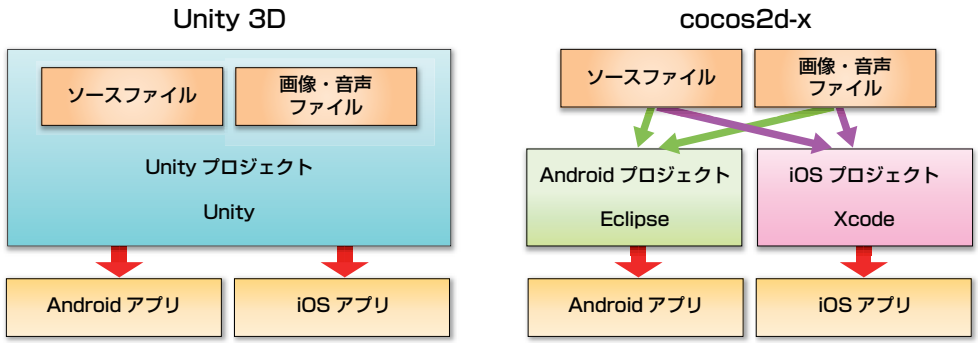

図2◉ Unity 3Dとcocos2d-xのクロスプラットフォーム開発の比較

オープンソース（MITライセンス）

cocos2d-xは、MITライセンスのオープンソースソフトウェアです。オープンソースソフトウェアとは、ソースコードが公開されているソフトウェアのことです。またMITライセンスにおいて皆さんが利用する上で関係することを挙げると、無償で利用できること、ソースコードの複製や改変が行えることです。無償ですので、初心者の方でも軽い気持ちで利用することができます。またソースコードの改変が行えるので、上級者の方が自分のためにカスタマイズすることもできます。ソースコードはGitHubで公開されており、開発元だけでなく一般ユーザが機能追加・修正を行うこともできます。

アプリ作成に必要なフレームワーク・ツール

先ほど紹介したように、cocos2d-xではプラットフォームごとにアプリ開発環境が必要になります。iOSアプリは基本的にMac上でしか作成できません。AndroidアプリはMacでもWindowsでも開発可能ですが、Windowsのほうが多くのものをインストールする必要があります。これらのインストール方法は、次のセクションで解説します。

	Mac	Windows
Android	cocos2d-x Eclipse Andorid SDK Android NDK	cocos2d-x Eclipse Andorid SDK Android NDK Java Cygwin Python
iOS	cocos2d-x Xcode	作成できません

図3◉ 開発に必要なソフトウェア

1-1-2 cocos2d-xの歴史

cocos2d-xはマルチプラットフォーム開発が可能な2Dゲームフレームワークとして有名ですが、cocos2d-xができるまでには長い歴史がありました。cocos2d-xの仕組みを学ぶ上で、知っておいた方が理解しやすいと思いますので紹介しておきましょう。

cocos2dは、2008年2月にオープンソース（MITライセンス）の2Dゲームフレームワークとして誕生しました。このときからマルチプラットフォーム開発が可能であり、Windows・Mac OS X・Linuxに対応したゲームをPythonで開発することができました。

誕生からわずか4ヶ月後の2008年6月に、cocos2d for iPhoneが誕生しました。cocos2dの機能を受け継いだほか、オープンソース（MITライセンス）という点も変わっていません。「for iPhone」とありますが、Mac OS Xの開発も可能です。ちょうど世界的にiPhoneの人気があった頃ですので、一気に普及しcocos2dを超える勢いで利用者が増加しました。今のcocos2dのブームを作り出したのは、このcocos2d for iPhoneといっても過言ではないでしょう。これがきっかけとなり、cocos2d for iPhoneから多くの派生が生まれました。

2008~2009年頃は、現在のスマートフォンを代表する2大OSであるiOSとAndroidがシェアを伸ばし始めた頃でした。iOS向けにはcocos2d for iPhoneがあったように、Android向けにもJavaで開発可能なcocos2d-androidと呼ばれるプロジェクトがありました。しかしこの開発が難航していたためcocos2d-android-1と呼ばれるプロジェクトまで立ち上がりました。とは言うものの、Javaでの開発は非常に困難であり、現在では共に開発がストップしています。

そしてcocos2d for iPhone の初版から2年半後の2010年11月にcocos2d-xが生まれました。もちろんいずれもオープンソース（MITライセンス）です。乱立する多くのOSに対応できるようプログラミング言語はC++で開発されており、対応OS数は10を超えています。デスクトッププラットフォームではWindows・Mac OS X・Linuxに対応しており、モバイルプラットフォームではiOS・Android・Blackberryなどの他マイナーなOSにも対応しています。cocos2d-xは、cocos2d for iPhoneの機能を有しているだけではなく、iOS用アプリケーションを構築するためのCocoaフレームワークの機能も一部取り入れています。そのためメモリ管理の考え方は非常に似ており、cocos2d-xにもretain関数やrelease関数などが存在します。そのおかげもあり、cocos2d-xはプログラミング言語がC++であるにも関わらず、基本的な範囲であればメモリを気にすることなく利用することができます。

またcocos2d-xは、C++のほかにも（OSが限定されてしまいますが）LuaやJavascriptで開発することができます。今後はWindows8やWindows Phone 8にも対応する予定となっています。

1-2 cocos2d-x のインストールおよびプロジェクトの作成

最初にcocos2d-xをインストールしましょう。そしてcocos2d-xを適用したプロジェクトを作成します。プロジェクトの作成には便利なツールがあり、1コマンドで簡単にマルチプラットフォーム開発環境を構築してくれます。

1-2-1 cocos2d-xのインストール

cocos2d-xは以下の公式サイトのダウンロードページよりダウンロードできます。

図4◉ 公式サイトのダウンロードページ
http://www.cocos2d-x.org/projects/cocos2d-x/wiki/Download

本書ではcocos2d-x 2.1.2を使用します。ダウンロードページの「cocos2d-2.1rc0-x-2.1.2-hotfix.zip @ Apr.08, 2013」をクリックして、「cocos2d-2.1rc0-x-2.1.2.zip」ファイルをダウンロードしましょう。ダウンロードが完了すると任意のディレクトリに解凍してください。本書ではホームディレクトリに解凍しました。

/Users/Sumomo/cocos2d-2.1rc0-x-2.1.2 （ユーザ名がSumomoの場合）

ディレクトリ名の指定は特にありませんが、cocos2d-xはバージョンアップが頻繁に行われるので、バージョンがわかるように解凍したままのディレクトリ名とすることをおすすめします。

1-2-2 プロジェクトの作成

プロジェクトの作成にはcocos2d-xに用意されているプロジェクト作成ツールを使用します。このプロジェクト作成ツールは、1コマンドでマルチプラットフォーム開発環境を構築してくれる優れものです。そのツールはインストールしたcocos2d-xのディレクトリの

cocos2d-2.1rc0-x-2.1.2/tools/project-creator/create_project.py

にあります。拡張子が「py」ですのでpythonが必要になります。Macであれば標準でPythonがインストールされているので問題ないでしょう。Windowsの場合は以下のURLよりダウンロードしてください。

http://www.python.org/download/

バージョンについては特に明記されていませんが、cocos2d-xのWikiには2.7.3を使用している例があるので、Python 2.7系をダウンロードするといいでしょう。またPythonインストール後は、インストールしたディレクトリを環境変数の「PATH」へ追加しましょう。

それではコマンドによる操作を行いますので、Macであれば「ターミナル.app」、Windowsであれば「cmd.exe（コマンドプロンプト）」を起動してください。そして

cocos2d-2.1rc0-x-2.1.2/tools/project-creator/

のディレクトリに移動します。そしてPythonにより「create_project.py」を実行します。「create_project.py」のオプションは3つあります。

「project」オプションはプロジェクト名を設定します。「package」オプションはパッケージ名を設定します。「language」オプションはプログラミング言語を指定します。プログラミング言語は3種類あり、cpp（C++）・lua・JavaScriptから選択できます。また作成されるプロジェクトのプラットフォームはプログラミング言語によって異なります。

- C++の場合、iOS, Android, Win32, Mac, Blackberry, Linux, Marmaladeの7種類
- Luaの場合、iOS, Android, Win32, Blackberry, Linux, Marmaladeの6種類
- JavaScriptの場合、iOS, Android, Win32の3種類

オプションはすべて指定しないとエラーになります。それでは「create_project.py」を実行してみましょう。

● Mac

```
./create_project.py -project MyProject -package net.tks2.myproject -language cpp
```

● Windows

```
create_project.py -project MyProject -package net.tks2.myproject -language cpp
```

正常に処理されると次のように表示されます。

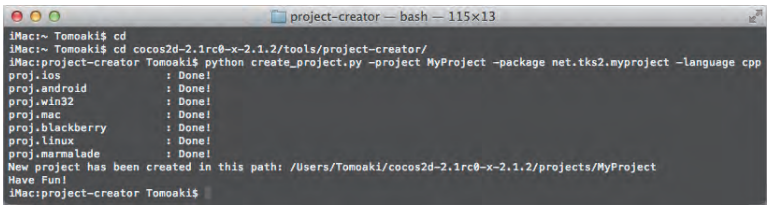

図5◉ create_project.pyの実行（Mac）

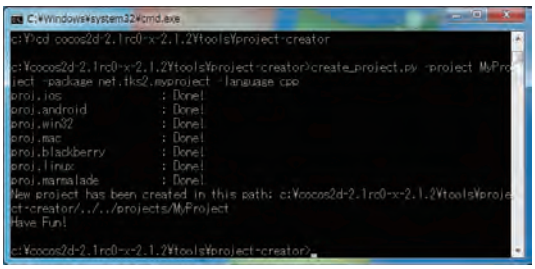

図6◉ create_project.pyの実行（Windows）

作成されたプロジェクトは、次のディレクトリに作成されました。

cocos2d-2.1rc0-x-2.1.2/projects/MyProject

少しディレクトリの中を覗いてみましょう。

cocos2d-xで重要な「Classes」ディレクトリと「Resources」ディレクトリがあります。基本的にヘッダーファイルやソースファイルは「Classes」ディレクトリに配置し、画像や音声などのリソースファイルは「Resources」ディレクトリに配置します。また「proj.xxx」ディレクトリがOSの数だけ存在します。各OSに対してビルドするときは、「proj.xxx」にあるファイルを用いますが、ソースは「Classes」を、リソースは「Resources」を読込みます。これを利用すれば1ソースで複数のOS向けにアプリを作成することができます。

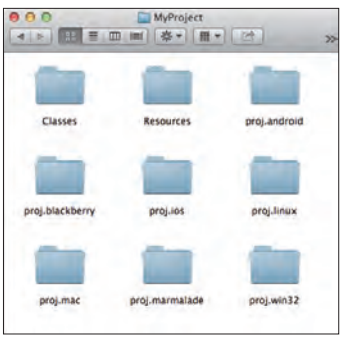

図7◉ プロジェクトのディレクトリの内容

1-3 Eclipse で Android アプリを作る (Win、Mac)

Eclipse上でAndroidアプリを開発する方法について紹介します。具体的には、Androidアプリ開発環境の構築からアプリを実行するまでの手順や、ファイルを追加する際の作法などについて説明します。

1-3-1 EclipseおよびAndroid SDKのインストール

一般的にAndroidアプリの開発には、EclipseとAndroid SDKが利用されます。cocos2d-xにおいても、Androidアプリを実行するときはEclipseを利用するのが一番お手軽だと思います。以前は、EclipseやAndroid SDKのインストールには手間が掛かっていましたが、現在ではEclipseとAndroid SDKを一緒にダウンロードすることができます。またEclipseにはADTプラグインが適用されている状態なので、わずらわしい設定などが必要ありません。

すでにEclipseとAndroid SDKがインストールされており、Androidアプリを作れる環境が整っているのであれば「**1-3-3** Android NDKのインストール」から読み始めても問題ありません。さらにAndroid NDKもインストール済みでC++での開発環境が整っているのであれば、Windowsの場合は「**1-3-4** Cygwinのインストール」、Macの場合は「**1-3-5** Eclipse上での設定」から読み始めても問題ありません。本節ではMacとWindows両方の説明を行いますので、各OSで必要な箇所をご覧ください。それでは次のURLよりAndroid SDK(ADT Bundle)をダウンロードしましょう。

図8◉ Android SDKのホームページ
http://developer.android.com/sdk/

ダウンロードが完了したら任意のディレクトリへ展開すると、インストールは完了です。ディレク

トリの中を見てみると「eclipse」と「sdk」ディレクトリがあります。またWindowsの場合であればSDK Manager.exeもあります。

前記だけでも最新のAndroidプラットフォームの対応ファイルがインストールされているため問題ありませんが、cocos2d-xはAndroid2.2以上に対応しています。必要に応じて、EclipseからSDK Managerを起動して、Android 2.2（API 8）のAndroidプラットフォームをインストールするといいでしょう。またWindowsであれば、SDK Manager.exeからもインストールできます。

1-3-2 Javaのインストール

Macの OS X 10.7までは標準でJavaがインストールされていましたが、OS X 10.8からはJavaがインストールされていません。しかし、Javaが必要な状況になると、次のようなダイアログが表示されるので簡単にインストールすることができます。

図9◉ Javaインストール（Mac）

Windowsの場合は、Javaのインストールが必要になります。次のJavaダウンロードのページよりJDKをダウンロードしてください。

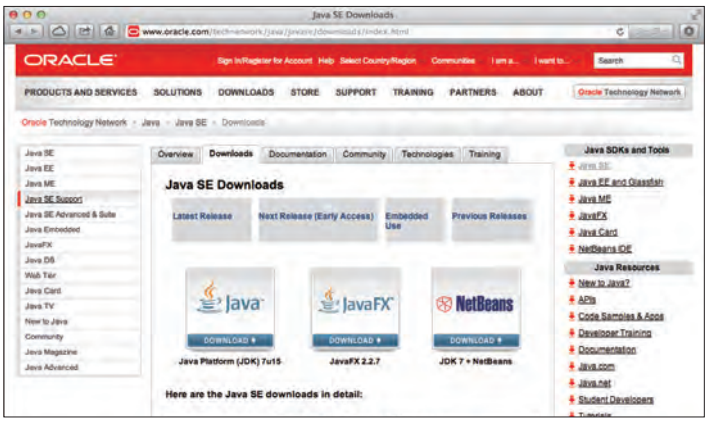

図10◉ Javaダウンロードのページ
http://www.oracle.com/technetwork/java/javase/downloads/

ダウンロードが完了したら、ダウンロードされた実行ファイルをダブルクリックしてインストーラーを起動し、画面の指示に従ってJDKをインストールしてください。

1-3-3 Android NDKのインストール

cocos2d-xを利用した場合、C++のソースがAndroid NDKによりライブラリ化されます。そのためAndroid NDKをインストールしましょう。Android NDKは、次のページよりダウンロードすることができます。適切なOSのファイルをダウンロードしてください。

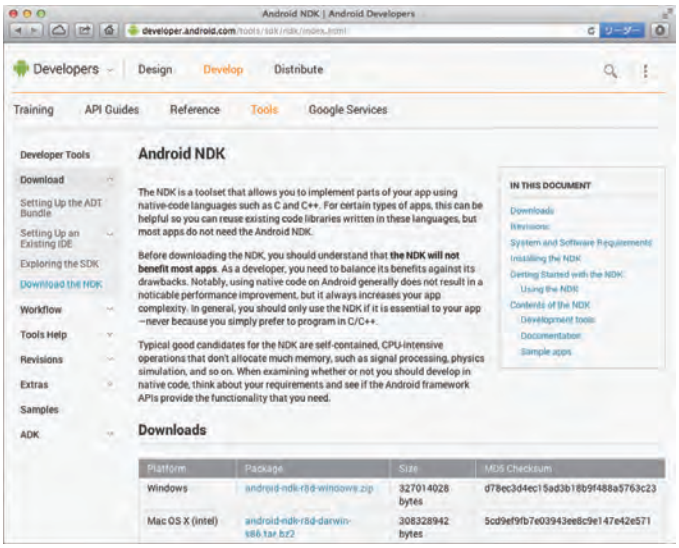

図11 Android NDKダウンロードページ
http://developer.android.com/tools/sdk/ndk/

ダウンロードが完了したら、任意のディレクトリへ展開しましょう。

1-3-4 Cygwinのインストール（Windowsのみ）

Android NDKを利用してC++のソースをライブラリ化するときは、Eclipse上で「proj.android」ディレクトリにある「build_native.sh」のシェルが実行されます。Macであれば問題なく実行できるのですが、Windows標準ではシェルを実行することができません。そこでCygwinを使用しビルドを行うので、Cygwinをインストールしましょう。Cygwinの次のページの「setup.exe」のリンクから「setup.exe」をダウンロードしてください。

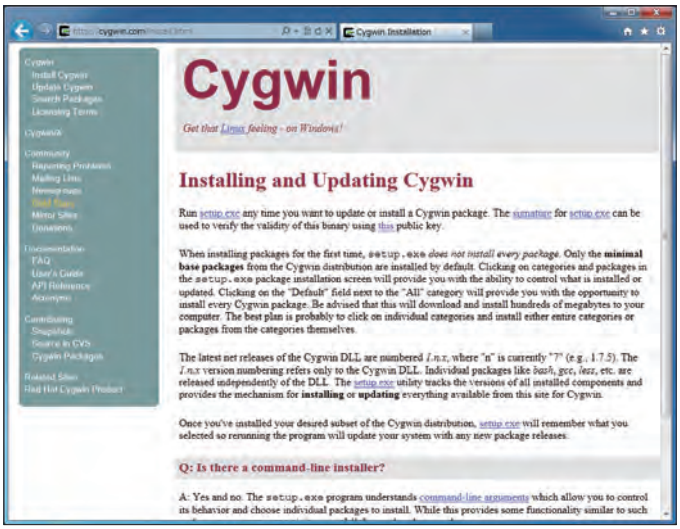

図12◉ Cygwinダウンロードページ
http://cygwin.com/install.html

ダウンロードが完了したら「setup.exe」を実行します。メッセージに沿って適切にインストールを行ってください。途中「Select Packages」の画面でパッケージを追加インストールすることができます。

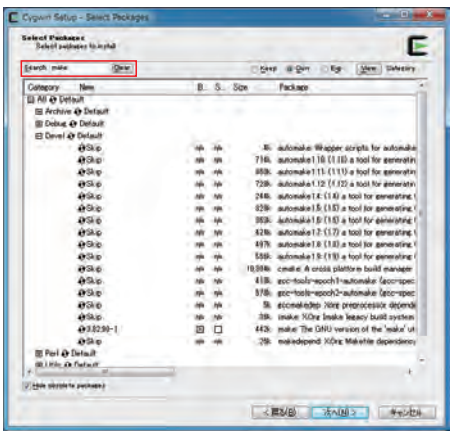

図13◉ Cygwinインストール中の「Select Packages」

ここで**図13**のように「make」もインストールするようにしてください。パッケージ数が多く見つけるのが困難かと思いますが、上にある検索ボックスより「make」で検索すると見つけやすくなります。パッケージ一覧の「Devel」の中から見つけることができます。

1-3-5 Eclipse上での設定

必要なものはすべてインストールできましたので、次はEclipseの設定を行いましょう。それではEclipseを起動して下さい。そして「Preferences」画面を表示します。Macの場合は、メニューから［ADT(Eclipse)］→［環境設定...］を選択します。Windowsの場合は、メニューから［Window］→［Preferences］を選択します。

表示された「Preferences」画面の左ペインから、［General］→［Workspace］→［Linked Resources］を選択して下さい。すると右ペインに「Linked Resources」画面が表示されます。

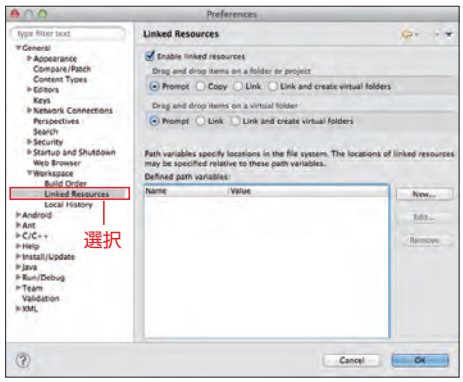

図14◉ Linked Resources（Mac）

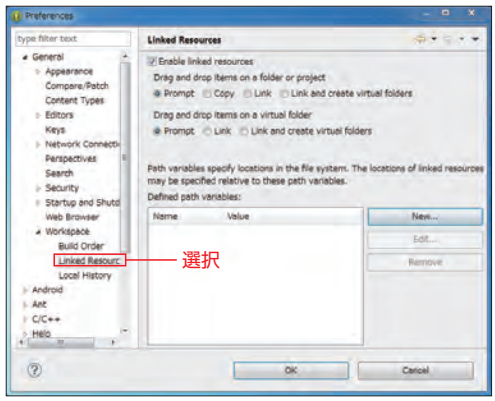

図15◉ Linked Resources（Windows）

「Linked Resources」画面の下には、「Defined path variables:」一覧があります。ここにcocos2d-xをインストールしたディレクトリを設定します。一覧の右に［New］ボタンがあるので、クリックしてください。すると「New Variable」画面が表示されます。ここでは［Name］に「COCOS2DX」を、［Location］に「(cocos2d-xをインストールしたディレクトリ)」を入力してください。

図16◉ New Variable（Mac）

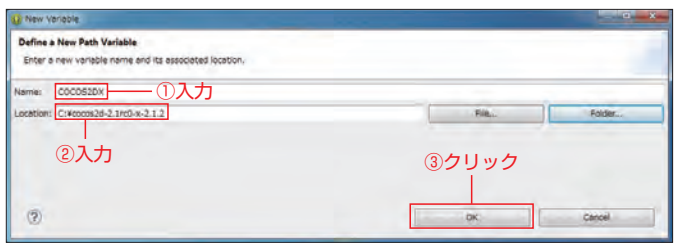

図17◉ New Variable（Windows）

入力が完了したら「OK」ボタンをクリックしましょう。すると「Linked Resources」画面の「Defined

path variables:」一覧に入力した値が登録されました。

次は「Preferences」画面の左ペインから、[C/C++] → [Build] → [Environment] を選択して下さい。すると右ペインに「Environment」画面が表示されます。

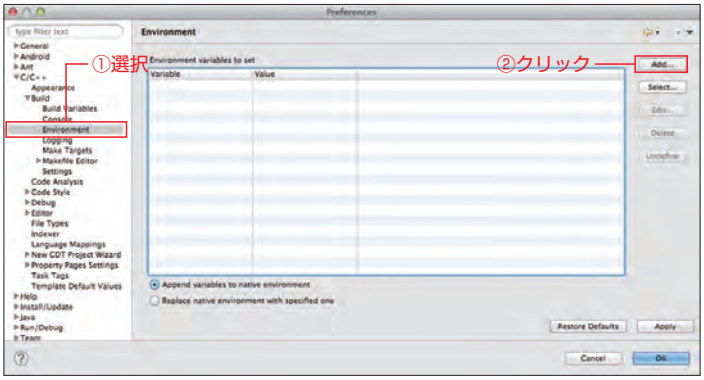

図18◉ Environment（Mac）

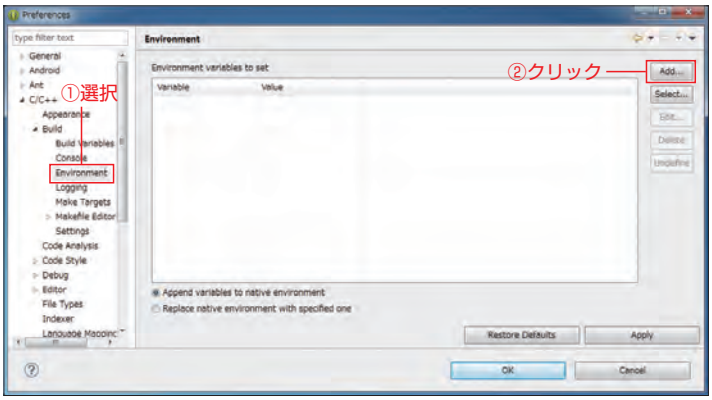

図19◉ Environment（Windows）

「Environment」画面には一覧がありますが、ここにAndroid NDKをインストールしたディレクトリを登録します。また、Windowsの場合は、Cygwinとシェルオプションの設定も行います。一覧の右に［Add...］ボタンがあるので、クリックして下さい。すると「New variable」画面が表示されました。ここでは［Name］に「NDK_ROOT」、［Value］に「(Android NDKをインストールしたディレクトリ)」を入力してください。「Value」項目の右に「Variables」ボタンがありますが、こちらからは入力できないので「Value」項目に直接入力を行ってください。

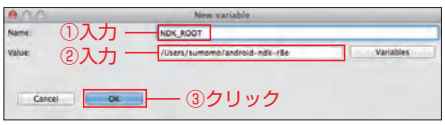

図20◉ New Variable（Mac）

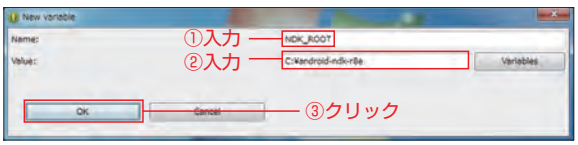

図21◉ New Variable（Windows）

入力が完了したら［OK］ボタンをクリックしましょう。すると「Environment」画面の一覧に入力した値が登録されました。Windowsの場合は、同様の手順で「Name:CYGWIN、Value:nodosfilewarning」と「Name:SHELLOPTS、Value:igncr」を入力してください。すべての入力が完了すると「Environment」画面の一覧には、Macの場合は1つ、Windowsの場合は3つの値が登録されました。

次は「Preferences」画面の左ペインから、［C/C++］→［Code Analysis］を選択して下さい。すると右ペインに「Code Analysis」画面が表示されます。

［Problems］一覧から［Syntax and Semantic Errors］のチェックを外して下さい。

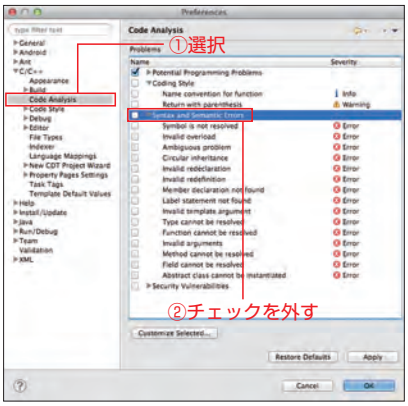

図22◉ Code Analysis（Mac）

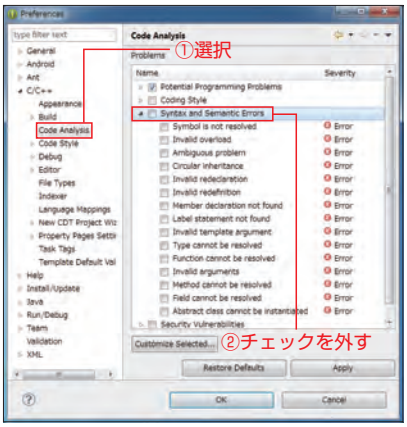

図23◉ Code Analysis（Windows）

次はEclipseでAndroid向けcocos2d-xライブラリプロジェクトである「libcocos2dx」を開きましょう。まずはEclipseを立ち上げてください。そしてメニューから［File］→［New］→［Project...］を選択しましょう。

「Select a wizard」の画面が表示されるので、一覧から［Android］にある［Android Project from Existing Code］を選択し［Next >］ボタンをクリックしてください。

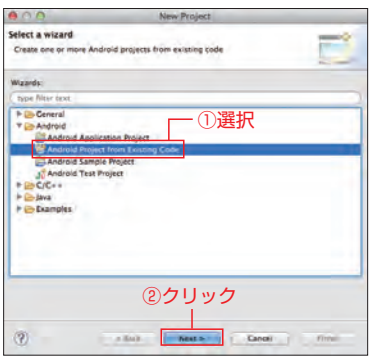

図24◉ Select a wizard（Mac）

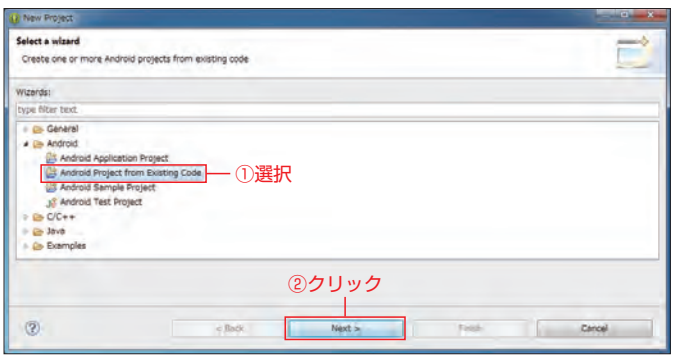

図25◉ Select a wizard（Windows）

「Import Projects」の画面が表示されるので、[Root Directory] にAndroid向けcocos2d-xライブラリプロジェクトである次のディレクトリを指定します。

cocos2d-2.1rc0-x-2.1.2/cocos2dx/platform/android/java

[Projects:] のリストにプロジェクトが存在することを確認し [Finish] ボタンをクリックしてください。

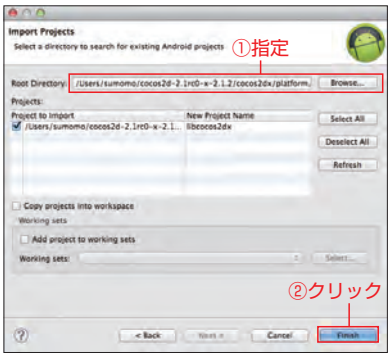

図26◉ Import Projects（Mac）

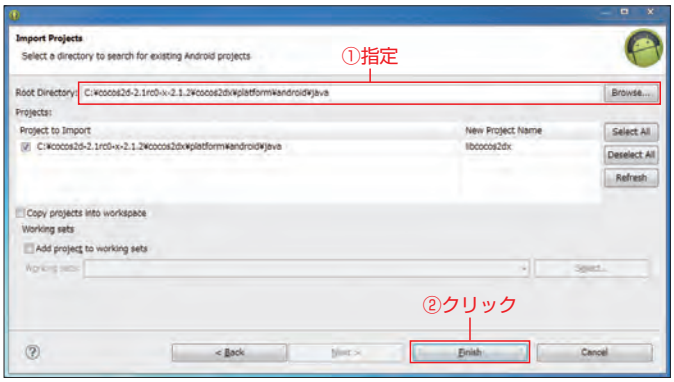

図27◉ Import Projects（Windows）

Eclipseの左ペインに「libcocos2dx」プロジェクトが表示されました。

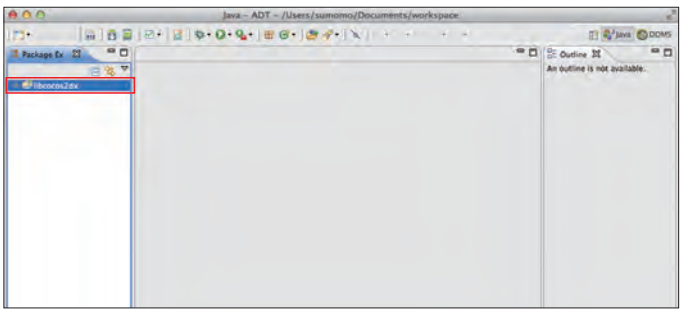

図28◉ libcocos2dxインポート（Mac）

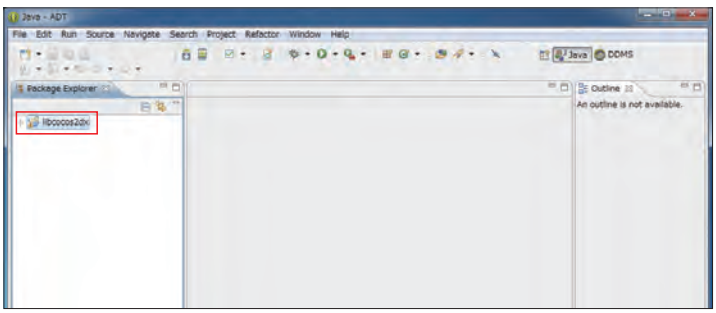

図29◉ libcocos2dxインポート（Windows）

ここまでの手順は、初回に一度だけ行えばいい作業です。ここからはプロジェクトを作成するたびに行う作業になります。「libcocos2dx」を取り込んだ手順と同じ方法で、作成したプロジェクトの「proj.android」ディレクトリを選択します。例えばプロジェクト名が「MyProject」の場合は、次のディレクトリを選択します。

cocos2d-2.1rc0-x-2.1.2/projects/MyProject/proj.android

正常にプロジェクトが取り込まれると、Eclipseの左ペインに「MyProject」プロジェクトが表示されます。

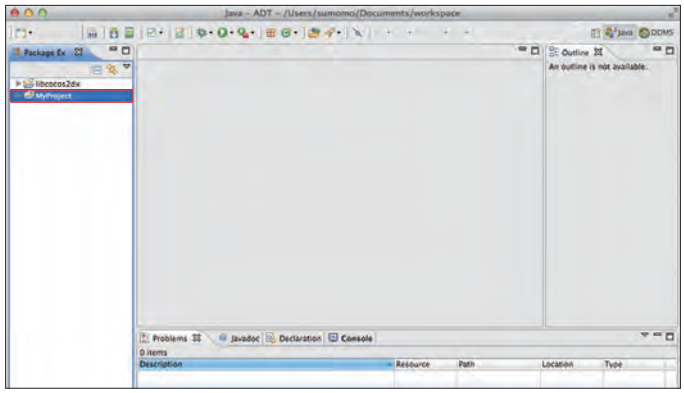

図30◉ libcocos2dxインポート(Mac)

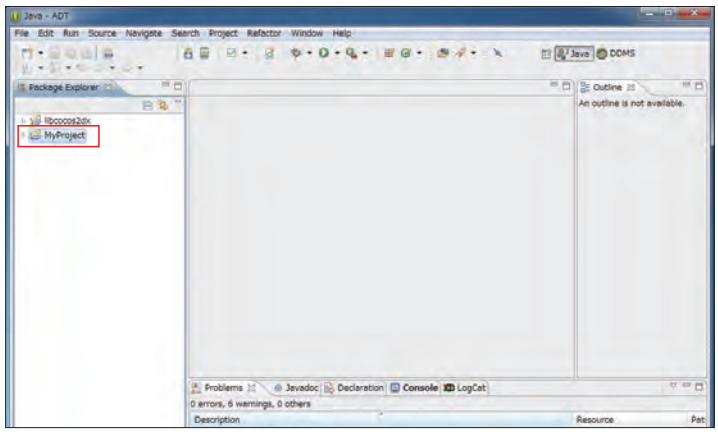

図31◉ libcocos2dxインポート(Windows)

以上でプロジェクトを取り込む作業が終わりました。Eclipseに特別な設定を行っていない場合、プロジェクトを取り込むと同時にビルドが開始されます。このビルドはPCのスペックにもよりますが、5~10分の時間を要します。Eclipseのステータスバーにビルドの状態がプログレスバーで表示されるので、それが消えるまで待って下さい。初回は「libcocos2dx」のビルドに時間がかかっているので、次回以降はすぐに終わります。

1-3-6 アプリの実行

Androidの動作確認は実機で行うといいでしょう（cocos2-x 2.1.3よりAndroidシミュレータで動作可能です）。

アプリの起動方法は、一般的なAndroidアプリと同じです。左ペインよりプロジェクトを選択し、メニューより［Run］→［Run］を選択してデバッグすることが可能です。ただし、C++のコードはライブラリ化されているため、Eclipse上でC++のコードに対してブレークポイントを設置できません。注意してください。

Eclipse以外でソースコードやリソースを変更した場合は、Eclipse上でリフレッシュを行いましょう。メニューの［File］→［Refresh］を選択すればリフレッシュできます。

また、APKの書き出しも、一般的なAndroidアプリと同じです。左ペインよりプロジェクトを選択し、コンテキストメニューを表示します。コンテキストメニューの［Android Tools］→［Export Signed Application Package...］を選択し、適切に署名を行いましょう。

1-3-7 ソースファイルの追加方法

これまではプロジェクト作成時に用意されたソースを用い、ライブラリのビルドを行いました。cocos2d-xを利用する上で、ソースファイルを追加するときの作法がありますので、その説明を行います。

基本的な流れは、以下の通りです。

1. ソースファイルを「Classes」ディレクトリへ配置する。
`cocos2d-2.1rc0-x-2.1.2/projects/MyProject/Classes`

↓

2. Makefileを編集する。
`cocos2d-2.1rc0-x-2.1.2/projects/MyProject/proj.android/jni/Android.mk`

↓

3. Eclipse上でプロジェクトのリフレッシュを行い、実機で確認する。

1.において、ソースファイルを「Classes」ディレクトリ以外へ配置する場合も、**2.**のMakefileを変更する必要があります。それでは**1.**と**2.**の説明のため、Android.mkを覗いてみましょう。

Android.mk

```
01 LOCAL_PATH := $(call my-dir)
02
03 include $(CLEAR_VARS)
04
05 LOCAL_MODULE := cocos2dcpp_shared
06
07 LOCAL_MODULE_FILENAME := libcocos2dcpp
08
09 LOCAL_SRC_FILES := hellocpp/main.cpp ¥
10                    ../../Classes/AppDelegate.cpp ¥
11                    ../../Classes/HelloWorldScene.cpp
12
13 LOCAL_C_INCLUDES := $(LOCAL_PATH)/../../Classes
14
15 LOCAL_WHOLE_STATIC_LIBRARIES += cocos2dx_static
16 LOCAL_WHOLE_STATIC_LIBRARIES += cocosdenshion_static
17 LOCAL_WHOLE_STATIC_LIBRARIES += box2d_static
18 LOCAL_WHOLE_STATIC_LIBRARIES += chipmunk_static
19 LOCAL_WHOLE_STATIC_LIBRARIES += cocos_extension_static
20
21 include $(BUILD_SHARED_LIBRARY)
22
23 $(call import-module,cocos2dx)
24 $(call import-module,cocos2dx/platform/third_party/android/prebuilt/libcurl)
25 $(call import-module,CocosDenshion/android)
26 $(call import-module,extensions)
27 $(call import-module,external/Box2D)
28 $(call import-module,external/chipmunk)
```

実装ファイルを追記

ヘッダーファイルのフォルダを指定

ここで注目するのは9行目のLOCAL_SRC_FILESと13行目のLOCAL_C_INCLUDESです。LOCAL_SRC_FILESは、実装を行う拡張子「cpp」のファイルを追記します。OSにより見え方が異なりますが、Macの場合は文末に「\」（バックスラッシュ）をつけることで改行でき、Windowsの場合は「¥」（エンマーク）をつけることで改行できます。

たとえば、「GameScene.cpp」のファイルを追加するときは、

```
LOCAL_SRC_FILES := hellocpp/main.cpp ¥
                   ../../Classes/AppDelegate.cpp ¥
                   ../../Classes/HelloWorldScene.cpp ¥
                   ../../Classes/GameScene.cpp
```

のように追加します。ここに追加するパスは、絶対パスでも相対パスでもどちらもOKです。

LOCAL_C_INCLUDESは、ヘッダーファイルとなる拡張子「h」のファイルを指定しますが、すでに「Classes」ディレクトリがセットされているので、特に追記は必要ありません。もし「Classes」ディレクトリ以外にあるヘッダーファイルを読込む場合は、LOCAL_C_INCLUDESに追記すればOKです。

編集後は**3.**を行い、ビルドエラーが発生しないことを確認しましょう。

1-3-8 画像・音声ファイルなどリソースファイルの追加方法

cocos2d-xでは画像・音声ファイルなどのリソースファイルの追加も作法があります。
基本的な流れは、次の通りです。

1. リソースファイルを「Resources」ディレクトリへ配置する。
`cocos2d-2.1rc0-x-2.1.2/projects/MyProject/Resources`

↓

2. Eclipse上でプロジェクトのリフレッシュを行い、実機で確認する。

cocos2d-xにおいては、リソースを「Resources」ディレクトリへ配置しました。しかし、Androidにおいてリソースは「Assets」ディレクトリへ追加するのが一般的です。

cocos2d-2.1rc0-x-2.1.2/projects/MyProject/proj.android/assets

Eclipse上でビルドが行われるとき、Androidプロジェクトに含まれる「build_native.sh」が実行されます。この「build_native.sh」は、リソースを「Resources」ディレクトリから「Assets」ディレクトリへコピーします。「build_native.sh」の該当する箇所を見てみましょう。

build_native.sh

```
01  # make sure assets is exist
02  if [ -d "$APP_ANDROID_ROOT"/assets ]; then
03      rm -rf "$APP_ANDROID_ROOT"/assets
04  fi
05
06  mkdir "$APP_ANDROID_ROOT"/assets
07
08  # copy resources
09  for file in "$APP_ROOT"/Resources/*
10  do
11  if [ -d "$file" ]; then
14      cp -rf "$file" "$APP_ANDROID_ROOT"/assets
15  fi
16
17  if [ -f "$file" ]; then
18      cp "$file" "$APP_ANDROID_ROOT"/assets
19  fi
20  done
```

「Assets」ディレクトリを削除

ファイルをコピー

2~4行目で「Assets」ディレクトリを削除しています。そして9~20行目で「Resources」ディレクトリから「Assets」ディレクトリへディレクトリとファイルをコピーしています。ここで注意すべき点は、「Assets」ディレクトリが一旦完全削除されることです。「Assets」ディレクトリのファイルだけを書き換えたとしても、削除されてしまうので意味がありません。また、バージョン管理にSubversionを使用している場合、管理ディレクトリである「.svn」まで削除されてしまいます。もし「Assets」ディレクトリが削除されては困る場合は、「build_native.sh」の前記の箇所を修正するかコメントアウトし、自分で管理する必要があります。

1-4 Xcode で iOS アプリを作る (Mac)

Xcode上でiOSアプリを開発する方法について紹介します。具体的には、iOSアプリ開発環境の構築からアプリを実行するまでの手順や、ファイルを追加する際の作法などについて説明します。

1-4-1 Xcodeのインストール

XcodeのインストールはApp Storeよりインストールを行います。すでにXcodeがインストールされており、iOSアプリを作れる環境が整っているのであれば「**1-4-2** プロジェクトの起動」から読み始めても問題ありません。

「App Store.app」を起動してください。Xcodeは[カテゴリ]タブの[開発ツール]の項目にあります。もし見つからない場合は、画面右上にある検索ボックスから「Xcode」を検索してください。Xcodeは無料でダウンロードできます。

図32◉ Xcodeのインストール

ダウンロードするファイルサイズが大きいので時間がかかりますが、ダウンロードからインストールまで自動で行われます。インストール完了後も特別な設定はありません。

1-5 cocos2d-x の重要なクラス

cocos2d-xを扱っていく上で知っておいた方が良いクラスを紹介します。何度もお世話になるクラスですので、自由に扱えるようになっておきましょう。

1-5-1 CCDirectorクラス

CCDirectorクラスは、cocos2d-xを統括するクラスになります。画面に表示するOpenGLのビューを指定したり、デバッグ情報の表示有無を指定したり、フレームレートを指定したりなど、画面表示に関する重要な役割を担っています。そのため、1つのアプリにCCDirectorクラスのインスタンスは1つしか存在しません。画面（シーン）の出し入れなど画面制御も担当しています。

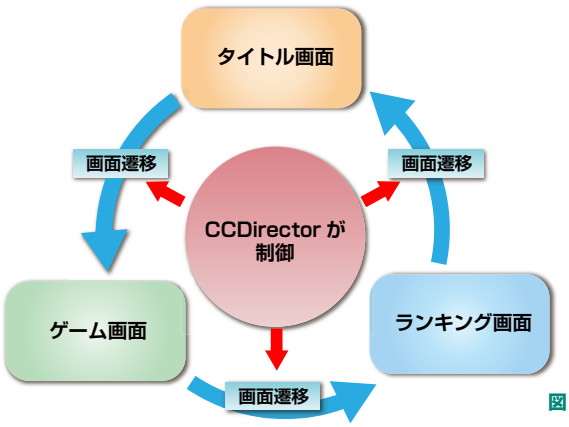

図42● CCDirectorによる画面遷移

1-5-2 CCSceneクラス

cocos2d-xでは、1つの画面が1つのシーン（CCSceneクラス）になります。**図42**で言うところの、タイトル画面・ゲーム画面・ランキング画面それぞれが1つのシーン（CCSceneクラス）です。た

とえば、初回に起動する画面を指定する場合もCCSceneクラスのインスタンスを利用しますし、画面遷移のアニメーションを行うときもCCSceneクラスのインスタンスを指定します。とはいうものの、CCSceneクラスのソースコードを見るとわかりますが、特別な機能を有しているクラスではありません。実際の利用では、次に紹介するCCLayerクラスのインスタンスを1つだけ持つ器ですので難しく考える必要はないでしょう。

1-5-3 CCLayerクラス

CCLayerクラスは、表示される画面の1つのレイヤーを表します。CCSceneクラスで紹介したように、基本的にCCSceneクラスのインスタンスは1つのレイヤーを持っていますが、そのレイヤーに対して複数のレイヤーを追加することも可能です。設計次第ではありますが、画像などのキャラクターを表示するだけのレイヤーや、タップ処理のみを受け付けるタップイベント取得レイヤーなどレイヤーのあり方は様々です。もちろん1つのレイヤーにすべての機能を詰め込んでも問題ありません。例えばChapter2で作成するゲームは、**図43**のように1つのシーン上に背景レイヤーを表示し、その上にコマとなるCCSpriteクラスやボタンとなるCCMenuItemクラスを配置しています。

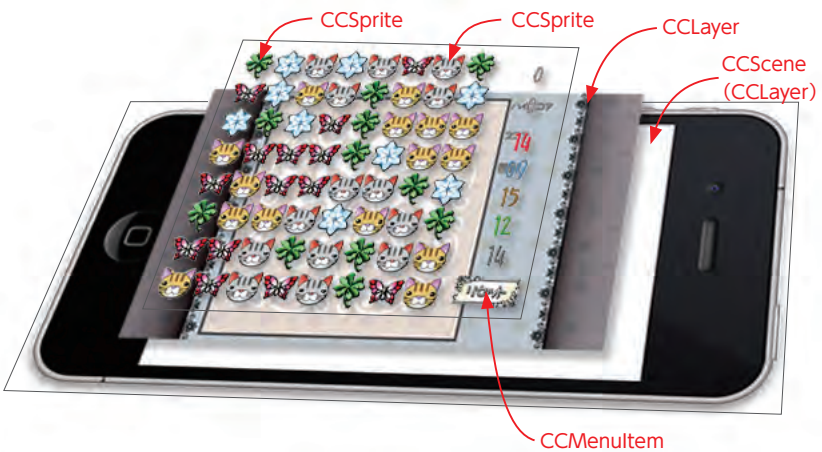

図43◉ CCLayerの利用

1-5-4 CCSpriteクラス

初心者であれば、CCSpriteクラスはキャラクターなどの画像を表示するクラスと覚えていればいいでしょう。ゲームの中で、画像によるメモリ消費は非常に大きいものです。そのためcocos2d-xでは、メモリ効率のいいテクスチャ表示方法なども用意されています。

このあたりについてはChapter5で紹介しているので、メモリにやさしいアプリを作る挑戦も行ってみてください。

1-5-5 CCMenuItemクラス

CCMenuItemクラスは、一般的なボタンに利用します。ただし、CCMenuItemクラスは単体では動作せず、CCMenuクラスと合わせて利用する必要があります。この利用方法はChapter2でも紹介しているので、そちらをご覧頂くといいでしょう。

1-5-6 CCActionクラス

CCActionクラスは、キャラクターを表すCCSpriteクラスなどに動きを与えるアクションとなります。時間とともにキャラクターが移動・回転・拡大縮小するなど多くのアクションが用意されています。またその他にも、アクション同士を組み合わせるアクションや、関数を呼び出すアクションなどもあります。これらのアクションがなければ、キャラクターに動きが見られないのでゲームがつまらなくなってしまいます。必ず使いこなせるようになっておきましょう。

Chapter 2
パズルゲームをつくる「にゃんがめ」

Chapter 2

この章で作るゲーム

みなさんは「さめがめ」というゲームをご存知でしょうか？ 1985年に森辺訓章氏がルールを考案してから、WindowsをはじめMacやUnix、さらにはニンテンドーDS用のゲームにもなりました。筆者も学生時代にMac版である「まきがめ」に夢中で、時間を忘れて友達と点数を競い合っていたものです。

簡単にルールを説明しましょう。

- 最初に、コマがランダムに配置されます。
- 同じ種類のコマが2つ以上並んでいるコマをタップすると、そのコマが消えます。
- 穴が空いたマスを埋めるように、上にあったコマが落ちてきます。
- もしコマを消すことで縦一列が消えてなくなると、右にあるコマが穴を埋めるように左へ移動します。
- コマを消すと点数が入ります。(消したコマ数 - 2)の2乗
- 隣り合う同じ種類のコマがなくなるまで、コマを消し続けます。

コマを消すだけの単純なゲームですが、同時に消すコマの数が多いほど点数は高くなるので、いかに隣り合うコマを増やすかが攻略の最大のポイントです。

単純ですが奥の深い「さめがめ」を作ってみましょう。本書ではネコのキャラクターを使うので、ゲームを「にゃんがめ」と命名しています。

2-1 利用するツールの説明

早速「にゃんがめ」を作りたいところですが、ゲームを飾る上で便利なツールと標準のcocos2d-xにはない新しいアクションを紹介しましょう。

2-1-1 GlyphDesigner（Mac）

ラベルの概要

cocos2d-xにはラベル表示にCCLabelTTFクラス、CCLabelAtlasクラス、CCLabelBMFontクラスの3種類が用意されています。

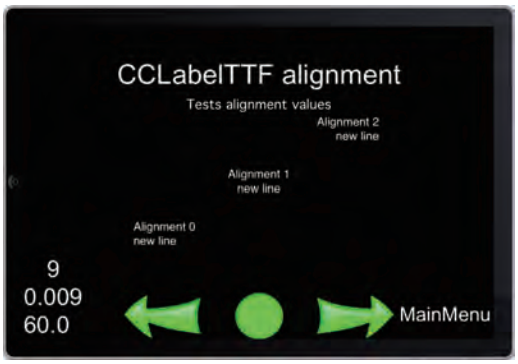

図1◉ CCLabelTTFクラスのサンプル

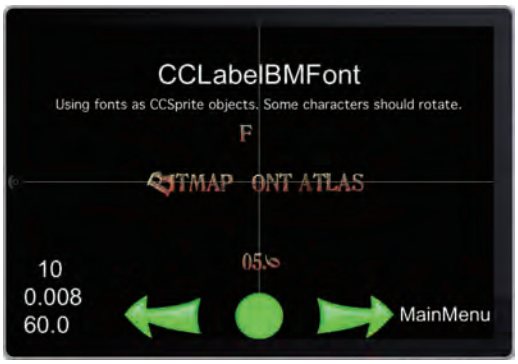

図2◉ CCLabelBMFontクラスのサンプル

この中でも一番簡単に利用できるのがCCLabelTTFクラスです。**図1**のように非常にシンプルですので説明文などにはいいと思いますが、たとえばスコア表示などには華やかさがなく向いていません。またOSにより使用可能なフォントが異なること、OSや端末により実際に表示されるサイズが異なったりもします。そんなときに利用するのがCCLabelAtlasクラスやCCLabelBMFontクラスです。**図2**のように画像化されたフォントを利用するため、ゲーム自体を飾ることができます。しかしテクスチャや設定ファイルが必要であり、それ相応の知識が求められます。そのために用意されたのが、簡単にテクスチャや設定ファイルを生成できるGUIツールです。

cocos2d-x公式サイトのWiki*でも紹介されていますが、Windows用であればドネーション（寄付）ウェアとして「Bitmap Font Generator」があります。またMac用には有償ではありますが、「GlyphDesigner」「bmGlyph」「TinyFont」があります。さすがに有償だけあって機能面では非常に優れています。自分に合ったツールを利用するといいでしょう。

＊ http://www.cocos2d-x.org/projects/cocos2d-x/wiki/Text_Labels#CCLabelBMFont

ここでは筆者が非常に便利だと感じた「GlyphDesigner」を紹介します。有償ではありますが試用が可能ですので、一度利用してみてはいかがでしょうか？　また本書では、サンプルとしてCCLabelBMFontクラスで利用可能なフォントを用意していますので、実装のときはそちらをサンプルとして利用しても結構です。

「GlyphDesigner」利用方法

「GlyphDesigner」の一般的な利用手順をご紹介します。

1. 「GlyphDesigner」をダウンロードしましょう。「GlyphDesigner」のホームページより［Free Download］ボタンを押下しダウンロードします（**図3**）。またインストールは一般的な方法ですので、手順に沿ってインストールを行ってください。
   http://www.71squared.com/glyphdesigner

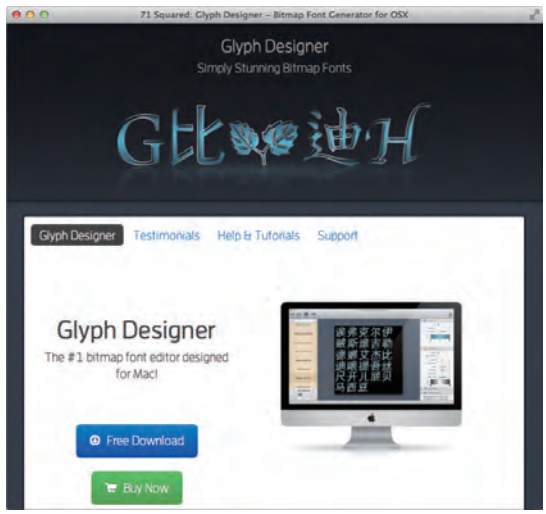

図3◉「GlyphDesigner」ホームページ

図4◉「GlyphDesigner」アイコン

2. インストールしたディレクトリよりアプリアイコンをダブルクリックして、「GlyphDesigner」を起動してください（**図4**）。「GlyphDesigner」は次のような画面構成になっています（**図5**）。

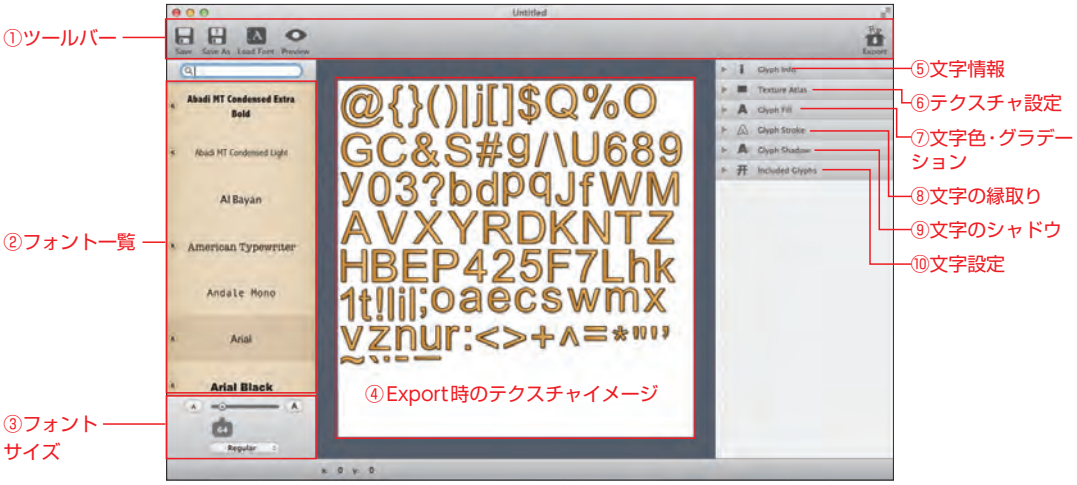

図5◉「GlyphDesigner」起動時の画像

3. ②の一覧には、Macにインストールされているフォント一覧が表示されます。日本語のフォントを作成したい場合は、ここから使用したい日本語フォントを選択すれば利用可能となります。また③によりフォントサイズを変更できるので、適切なサイズに変更してください。

4. ⑩より必要な文字を入力します（**図6**）。英数字のみが必要であればデフォルトのまま利用するのがいいでしょう。また②で日本語フォントを選択していると、日本語にも対応できます。［Update］ボタンを押下すると、変更した文字が④に反映されます。

5. ⑦より文字色の選択を行います（**図7**）。「Color Type: Gradient」を選択すると文字にグラデーションをかけることができます。

6. ⑧より文字の縁取りを行います（**図8**）。「Width」を変更することにより縁取りの幅が変わります。

図6◉入力文字の選択

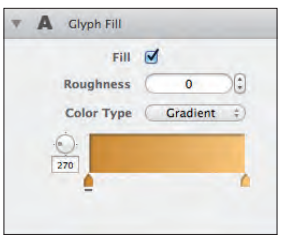

図7◉文字色の変更

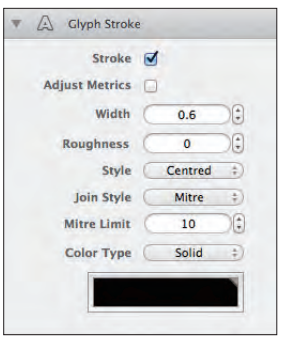

図8◉文字の縁取り

7. ⑨より文字の影を調整します（**図9**）。右にある「A」と書かれている文字の真ん中にある黄色の丸を移動することで簡単に調整することができます。また影の色を指定することもできます。

8. ①の［Preview］ボタンを押下すると、プレビュー画面が表示されます（**図10**）。これで実際のイメージを確認しましょう。

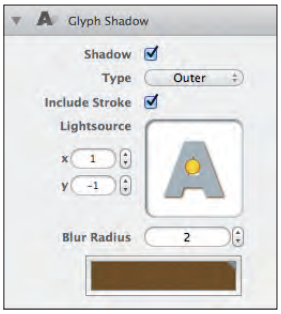

図9◉文字のシャドウ

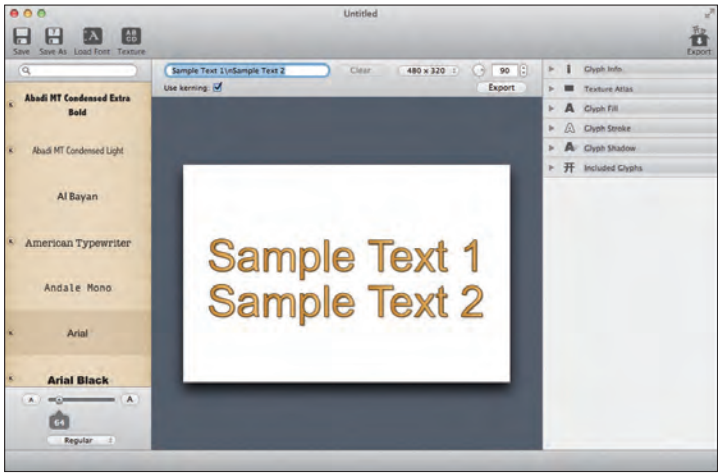

図10◉プレビュー画面

9. 問題なければ①の［Export］ボタンを押します。

図11◉ファイル名および保存先選択画面

10. ファイル名および保存先選択画面が表示されるので、それぞれ入力および選択します（**図11**）。
ここで「Export type」は必ず「.fnt (Cocos2d Text)」を選択してください。

11. [Save] ボタンを押下すると、指定した場所にfntファイルとpngファイルが作成されます。

以上が「GlyphDesigner」の一般的な利用手順となります。**3.**の説明の通り日本語にも対応できますが、必要な文字を全て登録する必要があるため、説明文のように長い文章にも使おうとすると、テクスチャファイルだけでかなり大きなサイズとなってしまうので注意しましょう。

このようなGUIツールだけでも、十分見栄えのいいフォントを作成することができますが、このGUIツールで生成したテクスチャファイルをペイントツールなどで更に加工して、他にはないフォントを生み出す方もいるようです。

cocos2d-x上での利用方法は、出力されたfntファイルとpngファイルをプロジェクトへ追加し、

```
CCLabelBMFont* label = CCLabelBMFont::create("hogehoge","Sample.fnt");
```

のようにCCLabelBMFontクラスのcreate関数でインスタンスを生成します。create関数の第1引数は表示したい文字列、第2引数はフォントファイル名を渡します。表示したい文字列の中の文字がフォントファイルに存在しない場合は、アプリが強制終了するので注意してください。実際の使い方は「にゃんがめ」の中で説明します。

2-1-2 自作アクションクラスの作成

cocos2d-xには、キャラクターに動きを与えるための多くのアクションが用意されています。cocos2d-xにおけるアクションとは、キャラクターとなる画像を時間とともに移動したり回転したりなど、キャラクターに動きを与える動作を示します。

たとえば

- CCMoveTo, CCMoveBy（移動）
- CCRotateTo, CCRotateBy（回転）
- CCScaleTo, CCScaleBy（スケール）
- CCSkewTo, CCSkewBy（変形）

などがあります。

2-2 「にゃんがめ」の作り方

それでは「にゃんがめ」を作りましょう！ 最初はプロジェクトの用意から背景・コマの表示、コマの消去までを見てみましょう。ここでは基本的なcocos2d-xの利用方法を学ぶことになります。

2-2-1 「にゃんがめ」の作り方の概要

今回の「にゃんがめ」の作成は次の手順で進めます。

- プロジェクトの作成・準備
- 背景の表示
- コマの表示
- コマの消去
- コマの削除アニメーション
- コマの移動アニメーション
- コマの残数表示
- スコア表示
- ゲーム終了処理
- ハイスコア表示
- リセットボタン表示

1つ1つ順番を追って見ていきます。説明に使用する画像や効果音は、本書サポートサイト（P002参照）よりダウンロードできるChapter2のサンプル（Resourcesディレクトリ）を利用しています。必要な場合はあらかじめダウンロードしてください。また今回のプロジェクトは1種類の画像のみを利用して、異なる解像度に対応します。

2-2-2 プロジェクトの作成・準備

最初にプロジェクトを作成しましょう。プロジェクトの作成方法は**1-2-2**を参照してください。ここではプロジェクト名を「nyangame」としました。この先の説明で「nyangame」と出てきた場合は、あなたが作成したプロジェクト名に置き換えて進めてください。

プロジェクトを作成するとHelloWorldクラスの「HelloWorldScene.h」「HelloWorldScene.cpp」がありますが、これらは利用しないので削除しても問題ありません。ゲームの画面はGameSceneクラスとするため「GameScene.h」「GameScene.cpp」を作成してください。クラスの作成方法は**1-3-7**（Android）または**1-4-4**（iOS）を参照してください。また、自動生成されるコードはすべて削除しています。

それでは追加したクラスに必要最小限のコードを入力しておきましょう。まずはクラスの宣言を行うために「GameScene.h」を編集します。

GameScene.h

```
01 #ifndef __GAMESCENE_H__
02 #define __GAMESCENE_H__
03
04 #include "cocos2d.h"
05
06 class GameScene : public cocos2d::CCLayer
07 {
08 public:
09     virtual bool init();
10     static cocos2d::CCScene* scene();
11     CREATE_FUNC(GameScene);
12 };
13
14 #endif // __GAMESCENE_H__#endif // __GAMESCENE_H__
```

「GameScene.h」には、CCSceneクラスとCCLayerクラスが出てきます。CCSceneクラスは、タイトル画面やゲーム画面など1つの画面（シーン）を表します。CCLayerクラスは、画面に表示する1つのレイヤーを表します。シーンのクラスは、6行目のとおりCCLayerクラスを継承します。CCSceneクラスではないので注意してください。CREATE_FUNCはマクロ定義ですが、その中身はcreate関数が宣言とともに実装されています。シーンのクラスは基本的に、init関数、scene関数、create関数があれば大丈夫です。

create関数はマクロで実装されたので、残りのinit関数、scene関数を実装しましょう。実装は「GameScene.cpp」に行います。こちらも必要最小限の次のコードを入力します。

GameScene.cpp

```
#include "GameScene.h"
#include "SimpleAudioEngine.h"

using namespace cocos2d;
using namespace CocosDenshion;
using namespace std;

CCScene* GameScene::scene()
{
    CCScene* scene = CCScene::create();
    GameScene* layer = GameScene::create();
    scene->addChild(layer);
    return scene;
}

// 初期化
bool GameScene::init()
{
    if (!CCLayer::init())
    {
        return false;
    }

    return true;
}
```

「SimpleAudioEngine.h」のincludeも行っていますが、後ほど音声周りを扱うため先に用意しています。またusing namespaceも同様に「CocosDenshion」と「std」を準備しています。以上でGameSceneクラスの準備は整いました。

次にアプリ起動時にこのGameSceneが実行されるようにしましょう。AppDelegate.cppの最初に「cocos2d.h」をincludeしている箇所があるので、そこにある「HelloWorldScene.h」を「GameScene.h」に変更します。

AppDelegate.cpp

```
#include "AppDelegate.h"
#include "GameScene.h"
```

```
04  USING_NS_CC;
```

次にAppDelegateクラスのapplicationDidFinishLaunching関数内で、HelloWorldクラスが生成されていたので、これをGameSceneクラスのインスタンスを生成するように変更します。またコードを見やすくするため、自動生成されたコメントは削除しています。

AppDelegate.cpp

```
01  bool AppDelegate::applicationDidFinishLaunching()
02  {
03      CCDirector* pDirector = CCDirector::sharedDirector();
04      CCEGLView* pEGLView = CCEGLView::sharedOpenGLView();
05      pDirector->setOpenGLView(pEGLView);
06      pDirector->setDisplayStats(true);
07      pDirector->setAnimationInterval(1.0 / 60);
08
09      CCScene *pScene = GameScene::scene();
10      pDirector->runWithScene(pScene);
11
12      return true;
13  }
```

以上で、プロジェクトの準備は完了です。これでアプリを動かすことも可能ですが、GameSceneクラスには最小限の実装しか行っていないため、真っ黒の画面が表示されるだけです。

2-2-3 背景の表示

近年多く販売されているAndroid端末を見ると、画面が正方形のものから長細いものまであり、アスペクト比も多種多様です。2012年冬に発売されたスマートフォンは解像度1280×720ピクセルのものが多く、このアスペクト比は16:9です。またタブレット端末として代表されるiPadのアスペクト比は4:3です。基本的にはこの辺りのアスペクト比をカバーできるといいでしょう。

図12◉背景に使用する画像