Frank Di Natale and David Parker

Contents

- Student Bios
- What is it?
- History
- Why use Cocos2d?
- Features
- Installation
- Creating a new project
- Classes
- Descriptions of various classes
- OO Principles with Cocos2d
- The Big Picture
- OO Suggestions for Cocos2d
- Number of games
- Other ports
- Future
- Other frameworks
- Resources

Frank Di Natale and David Parker

Frank is a PhD student in Computer Science ('16).

He enjoys gaming, coding, and drawing... and being a gangster.

He wants to get his PhD in Computer Architecture and work for Intel.

Parker is a dual MS in Computer Science and MBA student ('13)

He enjoys coding, running, bboying, sleeping, eating, and Scotch.

He wants to solve the world's problems one program at a time.

What is it?

- Cocos2d is a framework for building 2-dimensional (2D) applications (mostly games) for iOS
- It is most commonly used for game development
- It provides a wrapper to OpenGL ES which is already on the iOS device

History

- Based on Cocos2d, written in Python
  - Started march 2008
  - Originally named Los Cocos
- Cocos2d-iPhone
  - Quickly became Cocos2d as the iOS version overcame the Python version
  - iPhone version started in April 2008
  - iPhone v0.1 released in July 2008
- By end of 2008, over 40 games in the App Store made with Cocos2d

Why use Cocos2d?

- Easy to Use
  - Familiar, simple API and many examples
- Fast
  - Uses OpenGL ES best practices
- Flexible
  - Easy to use, easy to integrate with 3rd party libs
- Free
  - OSS, closed and open source compatible
- Community support
  - Big, active, friendly community (Forum and IRC)
- App Store approved
  - 2500 App Store approved games use it

Features (I)

- Scene Management (workflow)
- Transitions between scenes
- Sprite and Sprite sheets
- Effects: Lens, ripple, liquid, etc
- Actions (behaviors):
  - Transformations: Move, Rotate, Scale
  - Composable: Sequence, Repeat
  - Ease: Exp, Sin
- Menus and Buttons
- Integrated physics engines (Box2d and Chipmunk)

Features (II)

- Particle System
- Text Rendering
- Texture Atlas Support
- Tile Map Support
  - Orthogonal, Isometric, Hexagonal
- Parallax Scrolling Support
- Sound Support
- Streak Motion Support
- Render Texture Support
- Many, many more...

Installation

Download the latest (stable) version at:
http://www.cocos2d-iphone.org/download

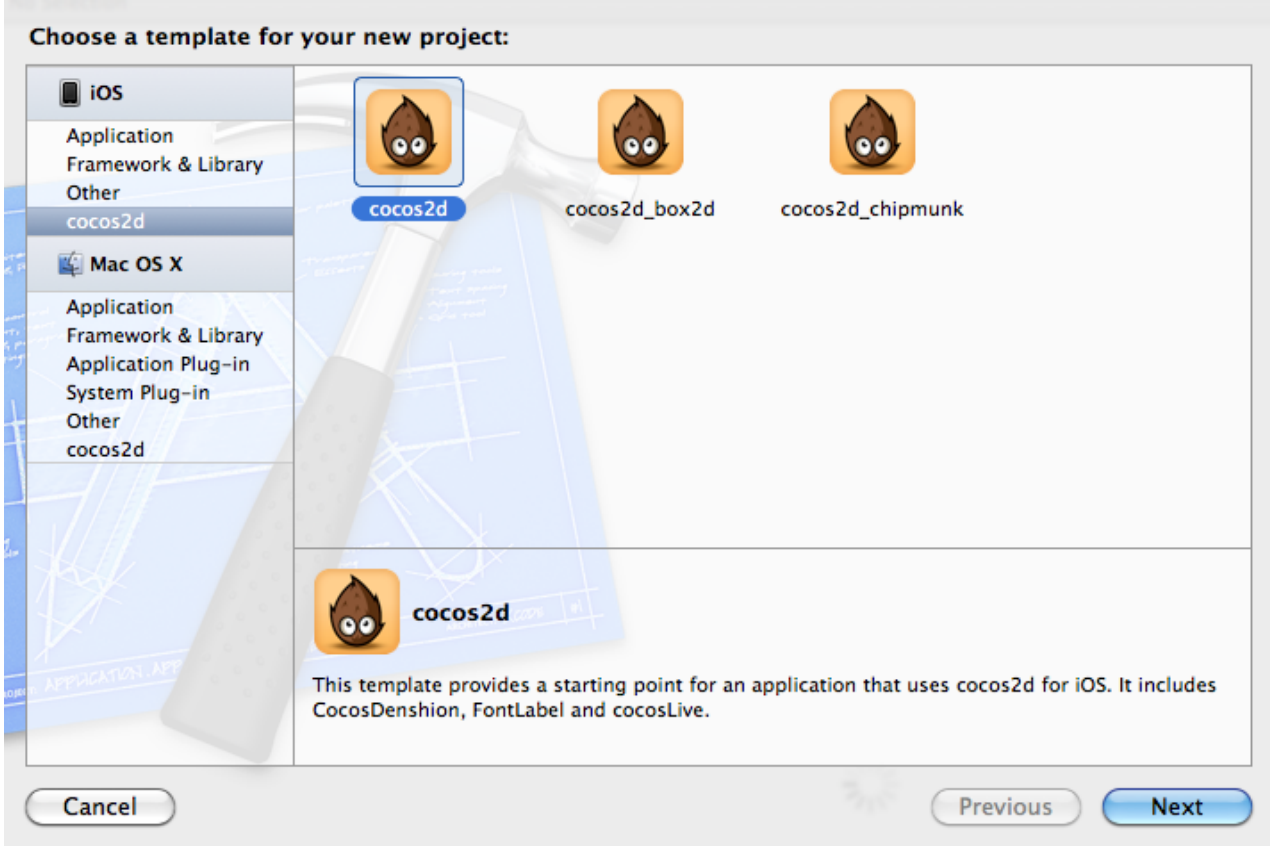
Install the templates by running:
./install-templates.sh -u -f

```
Installing Xcode 3 cocos2d Mac template
-------------------------------------------------

...creating destination directory: /Users/dparker/Library/Application Support/Developer/Shared/Xcode/Projec
2d 1.0.1/cocos2d Application - Mac/
...copying template files
...copying cocos2d files
...copying CocosDenshion files
done!
...copying file templates
...creating destination directory: /Users/dparker/Library/Application Support/Developer/Shared/Xcode/File T
1.0.1/
```

Creating a New Project

Open Xcode > New Project > cocos2d



Classes (I)

- CCAction
- CCActionCamera
- CCActionEase
- CCActionGrid
- CCActionGrid3d
- CCActionInstant
- CCActionInterval
- CCActionManager
- CCActionPageTurn 3d
- CCActionProgressTimer
- CCActionTiledGrid
- CCActionTween
- CCAnimation
- CCAnimationCache
- CCAtlasNode
- CCBlockSupport
- CCCamera
- CCConfiguration

Classes (II)

- CCDirector
- CCDrawingPrimitives
- CCGrabber
- CCGrid
- CCLabelAtlas
- CCLabelBMFont
- CCLabelTTF
- CCLayer
- CCMenu
- CCMenuItem
- CCMotionStreak
- CCNode
- CCParallaxNode
- CCParticleSystem
- CCParticleSystemPoint
- CCParticleSystemQuad
- CCProgressTimer

Classes (III)

- CCRenderTexture
- CCRibbon
- CCScene
- CCScheduler
- CCSprite
- CCSpriteBatchNode
- CCSpriteFrame
- CCSpriteFrameCache
- CCTexture2D
- CCTextureAtlas
- CCTextureCache
- CCTexturePVR
- CCTileMapAtlas
- CCTMXLayer
- CCTMXObjectGroup
- CCTMXTiledMap
- CCTMXXMLParser

Classes (IV)

- CCTransition
- CCTransitionPageTurn
- CCTransitionRadial

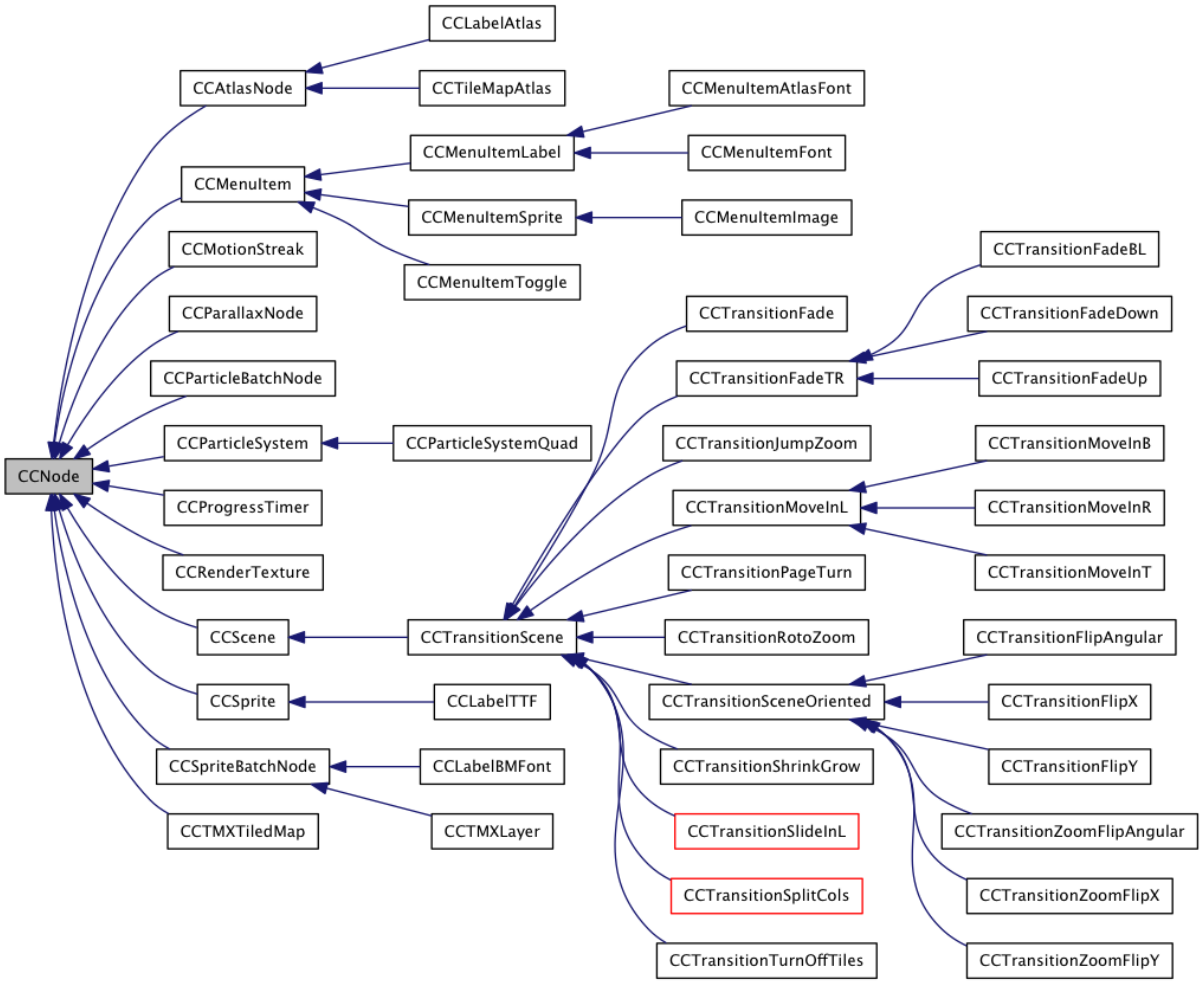
Due to the high number of classes used, this presentation will only cover "key" classes used to make a game with Cocos2d.

Class: CCNode

Source Code: https://github.com/cocos2d/cocos2d-iphone/blob/develop/cocos2d/CCNode.m
Documentation: http://www.cocos2d-iphone.org/api-ref/1.0.0/interface_c_c_node.html

- CCNode is the base element in Cocos2D.
  - Anything that can be drawn uses the **CCNode** object.
- CCNode Features:
  - They can contain other **CCNode** nodes (addChild, getChildByTag, removeChild, etc)
  - They can schedule periodic callback (schedule, unschedule, etc)
  - They can execute actions (runAction, stopAction, etc)
  - They can be translated, scaled, rotated, skewed, and moved.

Class: CCNode
How it uses OO?



- Some **CCNode** nodes provide extra functionality for them or their children.
- The most popular **CCNodes** are: **CCScene**, **CCLayer**, **CCSprite**, **CCMenu**.
- Subclassing a **CCNode** usually means (one/all) of:
  - overriding init to initialize resources and schedule callbacks
  - create callbacks to handle the advancement of time
  - overriding draw to render the node

Class: CCSprite

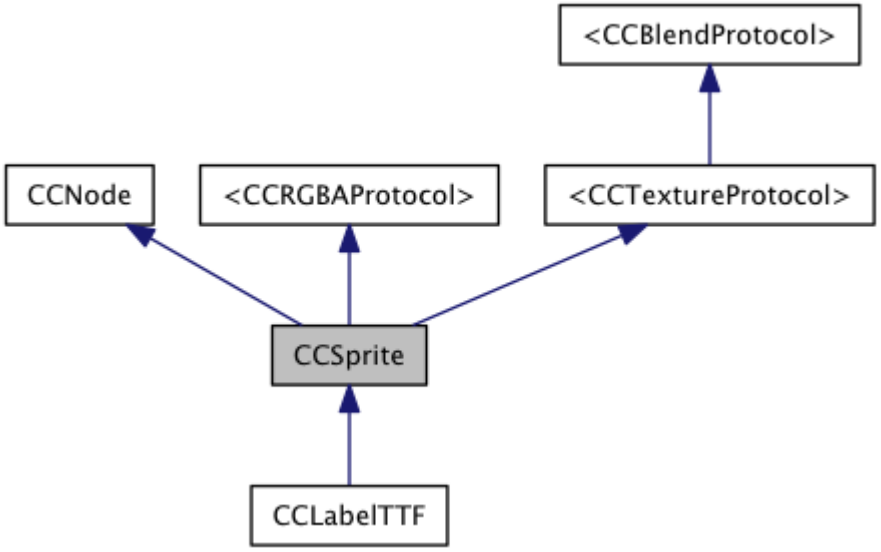
Source Code: https://github.com/cocos2d/cocos2d-iphone/blob/develop/cocos2d/CCSprite.m
Documentation: http://www.cocos2d-iphone.org/api-ref/1.0.0/interface_c_c_sprite.html

- **CCSprites** are derived from the **CCNode** class and have the same features.
- **CCSprite** is used to denote a **CCNode** that has an image that can be displayed to the user.
  - Supports blending functions
  - Supports aliasing/anti-aliasing

Class: CCSprite

- **Example of sprites from Nintendo's Pokemon Black on the Nintendo DS**

Class: CCSprite
How it uses OO?



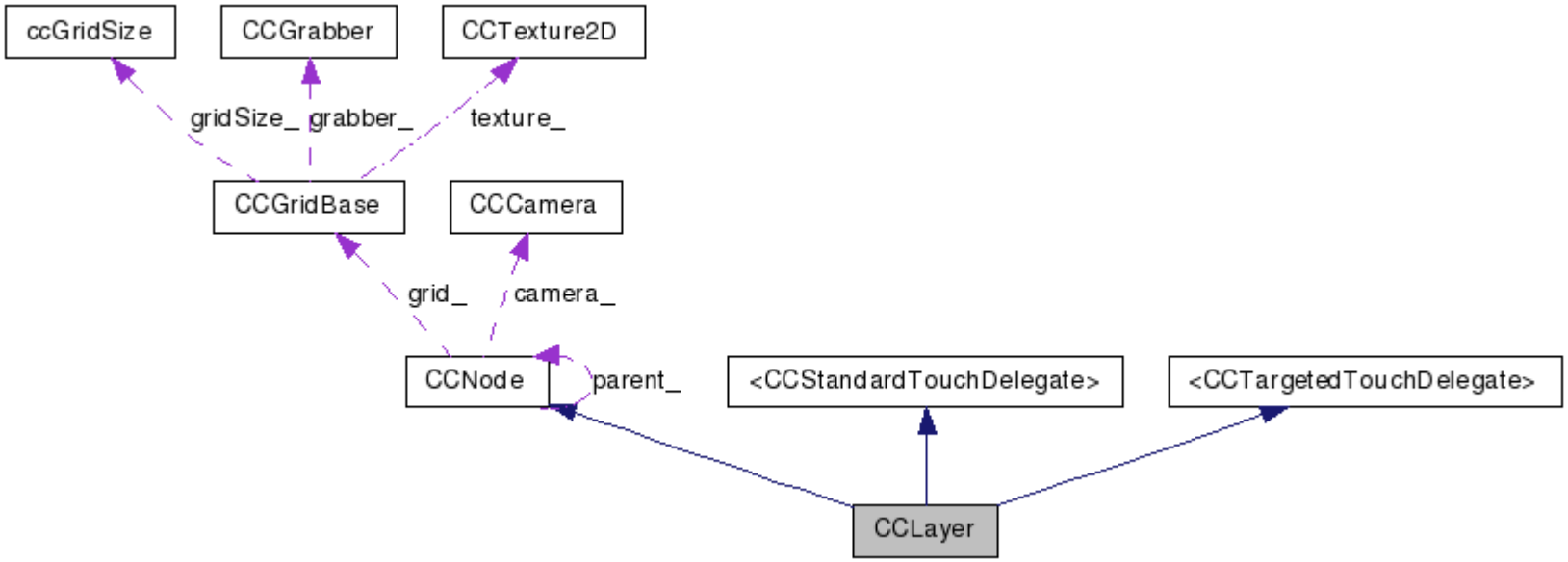
- A CCSprite is used to manage an image to be rendered to the output screen.
- Use of the CCSprite allows for batch rendering using the **CCSpriteBatchNode**.
  - Each CCSprite requires a single OpenGL call to be rendered.
  - Using batch rendering reduces the number of OpenGL calls made to render objects to the screen.

Class: CCLayer

Source Code: https://github.com/cocos2d/cocos2d-iphone/blob/develop/cocos2d/CCLayer.m
Documentation: http://www.cocos2d-iphone.org/api-ref/0.99.2/interface_c_c_layer.html

- **CCLayer** is a subclass of **CCNode** that implements the TouchEventsDelegate protocol.
- All features from **CCNode** are valid, plus the following new features:
  - It can receive iPhone Touches.
  - It can receive Accelerometer input

Class: CCLayer
How it uses OO?



- A CCLayer is no different than a CCNode, with the only exception being its implementation of touched interfaces.
  - The use of interfacing allows for developers to modify how their own game layers respond to touch (or disable it altogether).

Class: CCScene

Source Code: https://github.com/cocos2d/cocos2d-iphone/blob/develop/cocos2d/CCScene.m
Documentation: http://www.cocos2d-iphone.org/api-ref/1.0.0/interface_c_c_scene.html

- **CCScene** is a subclass of **CCNode** that is used only as an abstract concept.
  - **CCScene** an **CCNode** are almost identical with the difference that **CCScene** has it's anchor point (by default) at the center of the screen.
  - For the moment **CCScene** has no other logic than that, but in future releases it might have additional logic.
  - It is a good practice to use and **CCScene** as the parent of all your nodes.

Class: CCScene
How it uses OO?

- While the CCScene object is basically a node, there is the notion of a scene stack that controls the ordering of scenes.
  - The basic operation consists of a scene replacing the current scene, which does not require the use of the stack. The scene stack, however, is useful for layering multiple scenes on top of other background scenes.
  - The stack is useful for scenes where it is possible to have a menu appear over the current game screen (such as pause menus, inventory screens, etc.)
  - Another use is overlaying cutscenes onto the game screen, which would allow the player to resume where they left off one the scene is popped.

Class: CCDirector

Source Code: https://github.com/cocos2d/cocos2d-iphone/blob/develop/cocos2d/CCDirector.m
Documentation: http://www.cocos2d-iphone.org/api-ref/1.0.0/interface_c_c_director.html

CCDirector creates and handles the main Window and manages how and when to execute scenes.

The **CCDirector** is also responsible for:

- Initializing the OpenGL ES context
- Setting OpenGL pixel format (default is RGB565)
- Setting OpenGL buffer depth (default is 0-bit)
- Setting projection (default one is 3D)
- Setting orientation (default one is Portrait)

Class: CCDirector Singleton Goodness

CCDirector uses the **Singleton** Design Pattern.

The Singleton DP is used throughout Cocos2d.

The CCDirector instance is created within the AppDelegate.

Since the CCDirector is a singleton, the standard way to use it is by calling:

```
[[CCDirector sharedDirector] methodName];
```

Class: CCDirector Usage

The main two things the CCDirector ends up being used for is screen size and scene management:

```
CGSize size = [[CCDirector sharedDirector] winSize];

[[CCDirector sharedDirector] replaceScene:[LevelSelectScene scene]];
```

For scene management, the methods regularly used are:

- `runWithScene`, `replaceScene`, `pushScene`, and `popScene`.

Class: CCAction

Source Code: https://github.com/cocos2d/cocos2d-iphone/blob/develop/cocos2d/CCAction.m
Documentation: http://www.cocos2d-iphone.org/api-ref/1.0.0/interface_c_c_action.html

Base class for all **Action** classes

- Keeps track of target of action
- Action will affect the target based on appropriate tag

Action subclasses are generally in one of two categories:

- Instant actions - no duration
- Interval actions - takes place within period of time

Class: CCAction

Instant Actions (less common):

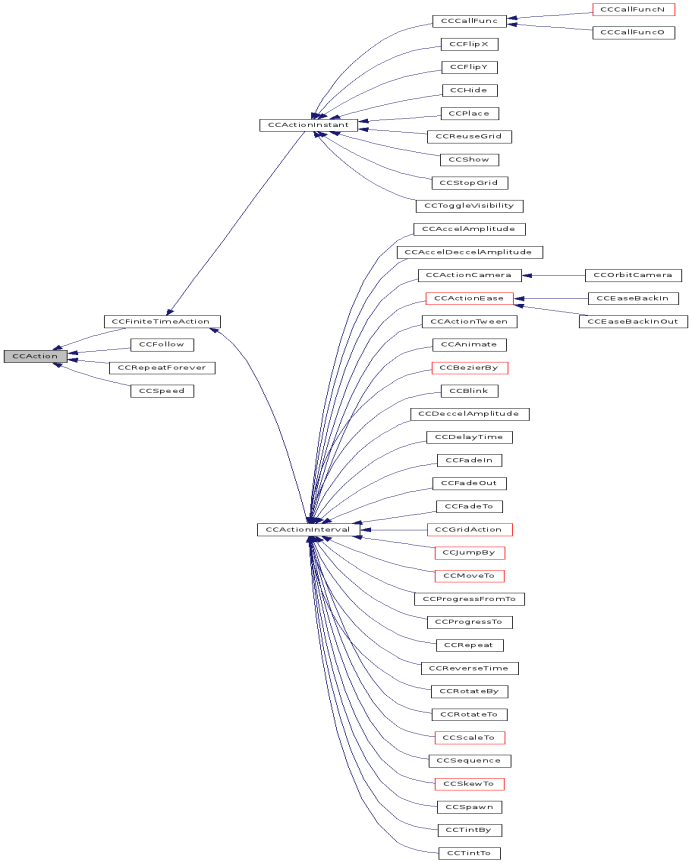
- Flipping
- Hiding
- Showing
- Placing
- Toggling Visibility

Interval Actions (more common):

- Easing
- Fading
- Moving
- Rotation
- Scaling
- Tinting
- Many more

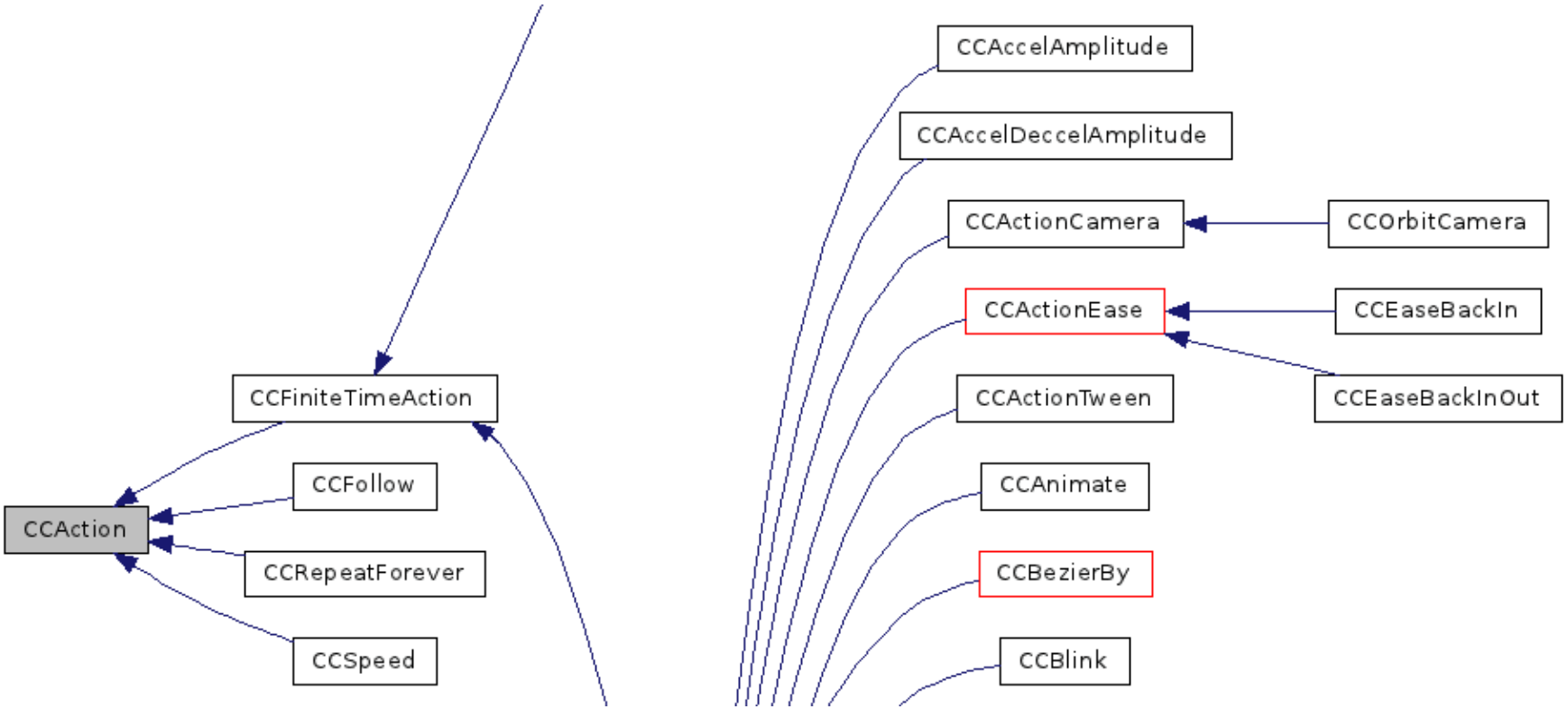
Class: CCAction

That's one **HUGE** Inheritance Tree:



Class: CCAction

Huge Inheritance Tree (a little closer):



Class: CCAction Usage

Create the specific type of action you want:

```
CCFiniteTimeAction* tint1 = [CCTintTo actionWithDuration:1 red:255 green:0 blue:0];
```

In this example, we're creating a Tint action.

We can create multiple, then add them to a **Sequence**, which we can then **Repeat** if we want. In this example, we attach the **RepeatForever** Action to the menu:

```
CCFiniteTimeAction* tint1 = [CCTintTo actionWithDuration:1 red:255 green:0 blue:0];
CCFiniteTimeAction* tint2 = [CCTintTo actionWithDuration:1 red:0 green:255 blue:0];
CCFiniteTimeAction* tint3 = [CCTintTo actionWithDuration:1 red:0 green:0 blue:255];
CCSequence* sequence = [CCSequence actions:tint1, tint2, tint3, nil];
CCRepeatForever* repeat = [CCRepeatForever actionWithAction:sequence];
[playButton runAction:repeat];
```

This will end performing a tint on the playButton, which will tint between red, green, and blue indefinitely.

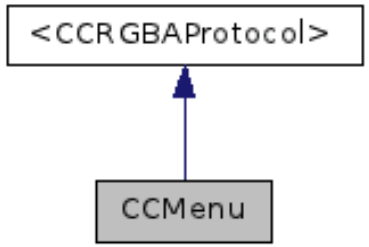
Class: CCMenu

Source Code: https://github.com/cocos2d/cocos2d-iphone/blob/develop/cocos2d/CCMenu.m
Documentation: http://www.cocos2d-iphone.org/api-ref/1.0.0/interface_c_c_menu.html

Allows **easy addition** of a **menu** to game

Features and Limitation:

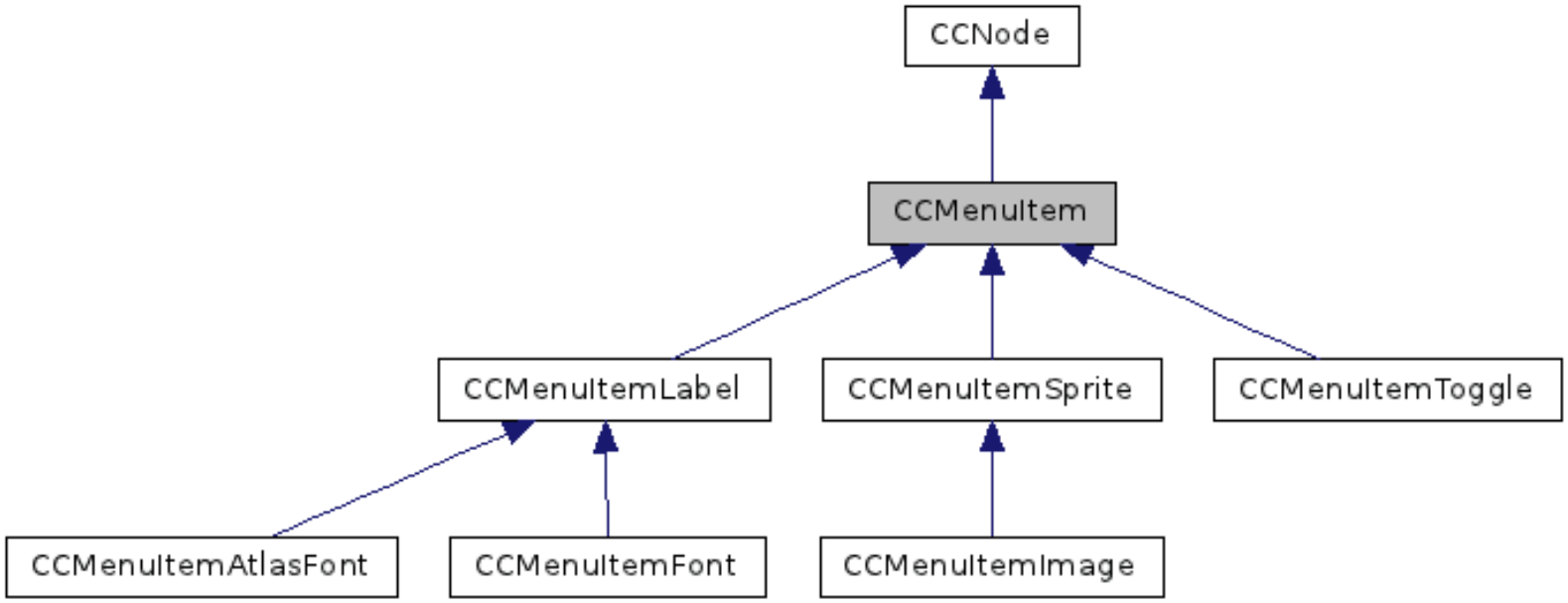
- You can add MenuItem objects in runtime using addChild:
- But the only accepted children are MenuItem objects



Class: CCMenuItem

Source Code: https://github.com/cocos2d/cocos2d-iphone/blob/develop/cocos2d/CCMenuItem.m
Documentation: http://www.cocos2d-iphone.org/api-ref/1.0.0/interface_c_c_menu_item.html

Super class for creating **Menu Items**



Class: CCMenu & CCMenuItem Usage

Create a menu item that you want to use.

```
CCSprite* lsNormal = [CCSprite spriteWithFile:@"play_button.png"];
CCSprite* lsSelected = [CCSprite spriteWithFile:@"play_button.png"];
CCMenuItemSprite* playButton = [CCMenuItemSprite itemFromNormalSprite:lsNormal
    selectedSprite:lsSelected target:self selector:@selector(playButtonTouched:)];
```

Then create a menu with that menu item, or attach it after creation.

```
CCMenu* menu = [CCMenu menuWithItems:playButton, nil];
[menu addChild:optionButton];
```

Now the button is added:

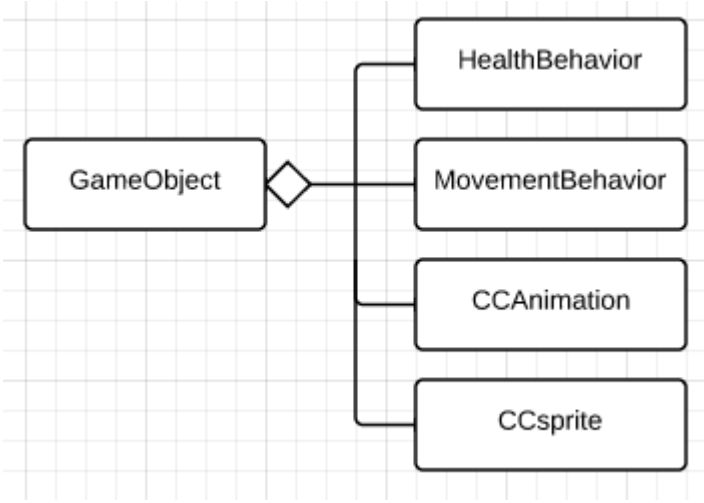
OO Principles with Cocos2d

- It is possible with the basic classes mentioned previously to create a set of generic objects that are used to create the main gameplay mechanics.
- GameObject
  - The base representation of all objects within the game.
  - Aggregates a few objects:
    - CCAnimation (subclass of CCAction) for animations
    - CCSprite to set its rendering image
    - HealthBehavior for tracking object life points (custom object)
    - Movement behavior and methods

OO Principles with Cocos2d

- GameCharacter
  - Derived from the GameObject class because a GameCharacter **IS-A** GameObject
  - Used to create in-game NPCs
  - Overloads movement, animations, and other methods as needed.
- You can additionally derive from GameCharacter to make a PlayerCharacter.
- Other objects that can be made from the existing Cocos2d libraries include specialized scenes (main menu scene, inventory scene, etc.)

OO Principles with Cocos2d - GameObject



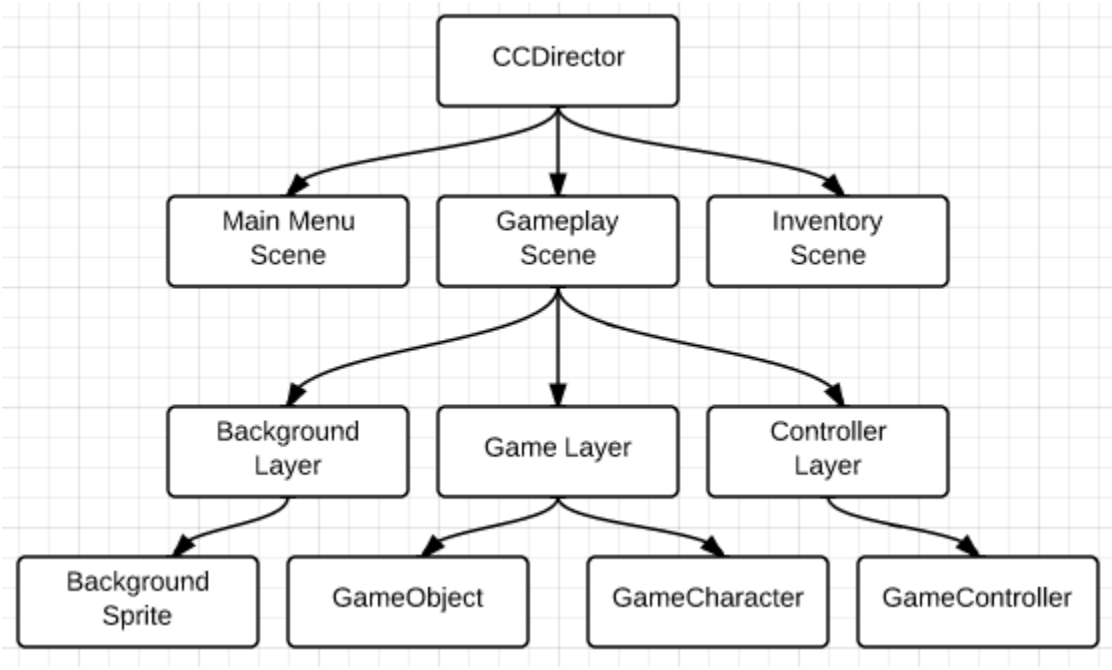
- The GameObject simply aggregates a CCSprite and other property classes such as CCAction, CCAnimation, etc.
  - Favoring aggregation over inheritance helps to keep the number of objects manageable because GameObjects exhibit varying behavior.
  - This design also allows new behaviors to be developed without affecting other classes, promoting loose coupling.

OO Principles with Cocos2d - GameObject

- From the GameObject, the GameCharacter class is a simple derivation with the following:
  - Overrides for default movement behavior. Assuming the GameObject would assume stationary objects, the new behavior would provide movement.
  - Health behavior could continue to be varied based on the type of GameCharacter (NPC, enemy, etc.)
  - The GameCharacter object would be the base object that NPCs, enemies, the player would inherit from.

The Big Picture

- The CCDirector is the controller to all of the other scenes in the game, coordinating timing and appearance.
  - The Gameplay Scene has been expanded to show how a scene is broken down into layers by function.
  - Each layer is then constructed of the necessary sprites and methods.



OO Suggestions for Cocos2d?

Cocos2d suffers greatly from using inheritance over using composition.

- For example, see the CCAction inheritance tree above.
- As opposed to having an CCActionBehavior, which can be assigned and set on the fly, the framework uses inheritance.
  - Instead, it should be using the Strategy pattern to allow different behaviors to be set.

Otherwise, Cocos2d is pretty well designed.

Number of games using Cocos2d

4,289 games listed on http://www.cocos2d-iphone.org/games/
as of October 17, 2012.

Probably hundreds (or thousands) more not listed!

Other ports

Besides Cocos2d, there are several other ports:

- Cocos2d-x
  - C++ Implementation of Cocos2d
- Cocos2d-html5
  - JavaScript port
  - Targets the Cocos2d-x API
- Cocos2d-android
  - Java port for Android

Future (I)

In active development:

https://github.com/cocos2d/cocos2d-iphone

- 2,340 users "starred" Cocos2d
- 521 forks
- 1000+ commits

Last commit (as of Nov 10):

- v1.X - Oct 21st
- v2.X - Nov 9th

Future (II)

Current stable build:

- v1.1.0

Next version build:

- v2.1-beta3 released November 7, 2012
- Allows targeting JavaScript (Cocos2d-x API)
- Atwood's Law:
  - "Any application that *can* be written in JavaScript, *will* eventually be written in JavaScript."

Other Frameworks

- Cocos2d-x
  - http://www.cocos2d-x.org/
- Corona
  - http://www.coronalabs.com/products/corona-sdk/
- Unity
  - http://unity3d.com/
- Sparrow
  - http://gamua.com/sparrow/
- OpenGL ES (the metal)
  - http://www.khronos.org/opengles/

Resources (I)

Framework and documentation:

- http://www.cocos2d-iphone.org/
- http://www.cocos2d-iphone.org/wiki/doku.php/
- http://www.cocos2d-iphone.org/wiki/doku.php/faq

Wikipedia:

- http://en.wikipedia.org/wiki/Cocos2d

Resources (II)

Issues tracker:

- http://code.google.com/p/cocos2d-iphone/issues/list

Source code:

- https://github.com/cocos2d/cocos2d-iphone/

Resources (III)

List of games using:

- http://www.cocos2d-iphone.org/games/

Very active forum:

- http://www.cocos2d-iphone.org/forum/

Resources (IV)

History:

- http://www.cocos2d-iphone.org/forum/topic/5653
- http://www.scribd.com/doc/88493987/Cocos2d-past-present-and-future

Resources (V)

Objective-C references:

- http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/objectivec/

Objective-C protocols:

- http://developer.apple.com/library/mac/#documentation/cocoa/conceptual/objectivec/chapters/ocProtocols.html

Resources (VI)

Cocos2d Books:

Learning Cocos2d: A hands on guide to building iOS games with Cocos2d, Box2d, and Chipmunk

By Ray Wenderlich

Learn iPhone and iPad cocos2d Game Development

By Steffen Itterheim