Cocos2D-Swift v4.0

Planning for v4.0 is mostly done, and some preliminary work has begun! This document may change as development continues, but hopefully it gives people a good idea about what sorts of changes to expect in the next few months.

When?

It's way too early to guess when it will be done, but you are always welcome to follow along with the progress or contribute on GitHub!
https://github.com/cocos2d/cocos2d-swift/tree/develop
https://github.com/spritebuilder/SpriteBuilder/tree/develop

Planned New Features

- CCFileUtils - Many improvements and simplifications. See the end for more information.
    - Better support for working across many device types without requiring several copies of assets at different sizes.
    - Device specific assets will be optional, not required.
    - Works with CCImage to automatically rescale images when loading them.
    - SpriteBuilder output will change to use the CCFileUtils so that SB projects won't require a separate non-default configuration.
- CCFile - A common class for working with files. Works with compressed files and files loaded from remote servers.
- CCImage - A class for loading and handling images.
- CCViewPortNode - A node type that uses a camera to render to a certain part of the screen. Supports panning, zooming, 3D effects and more!
- New CCTexture features.
    - Cube maps to use with CCEffects or custom shaders.
    - Integration with CCImage and better initializers to make it easier to load non-cached textures.
    - Public API support for configuring filtering or wrapping.
- Better multi-threading support. Cocos has never been very thread safe which makes it difficult to create scenes or load CCB files on a background thread.

v4.0 API Changes

API Deprecations

Deprecated APIs should continue to work, but will cause warnings in your code. Deprecated APIs may be removed in future versions.

- CCNode.isRunningInActiveScene - Replaced by CCNode.active
- CCNode.*Transform - Replaced by CCnode.*Matrix properties that return GLKMatrix4 instead.

- ccColor3B, ccColor4B, ccColor4F, ccVertex2F, ccVertex3F, ccTex2F - Use GLKMath types instead.
- ccBlendFunc - Use CCBlendMode instead.
- Vertex, triangle, and quad types (ex: ccV2F_C4B_T2F) - Use CCVertex or create local/private structs instead.
- CCTexture.pixelWidth, CCTexture.pixelHeight - Use CCTexture.pixelSize instead.
- [CCTexture initWithCGImage:contentScale:] - Use [CCTexture initWithImage:options:] instead.

API Changes

- CCFileUtils - See below for more information.
- CCConfiguration renamed to CCDeviceInfo for clarity. (Should we add a deprecated alias?)
- CCRenderableNode - Rendering properties on CCNode (blend mode, shaders, etc) have been moved into a new abstract subclass.
- CCTiledMapLayer.tiles was replaced with a readonly CCTileMapLayer.tileData property that returns an NSData.

API Removals

- Most, if not all, ivars will be made private - ivars in 3.x are public for legacy reasons, and have been the causes of many bugs.
- CCSpriteBatchNode, CCParticleBatchNode - Deprecated in 3.1 and no longer necessary.
- [CCNode setOpacityModifyRGB:], [CCNode doesOpacityModifyRGB] - Had been deprecated and unused since 3.1.
- CCBlendProtocol.blendFunc - Had been deprecated since 3.1. Use CCBlendProtocol.blendMode instead.
- CCNodeMultiplexer - A poorly supported node type with little to no use.
- CCParallaxNode - A poorly supported node type with unfixable problems. Use a parallax projection with CCViewPortNode instead.
- CCParticleSystemBase - This was merged with CCParticleSystem.
- CCTexturePixelFormat - Textures loaded from bitmaps must all be RGBA8 in v4. If you need to load packed texture formats, use PVR files.
- [CCTextureinitWithData:pixelFormat:pixelsWide:pixelsHigh:contentSizeInPixels:contentScale:] - Use [CCTexture initWithImage:options:] instead.
- CC_MAC_USE_DISPLAY_LINK_THREAD, CC_MAC_USE_OWN_THREAD - These modes had race conditions present for input that commonly resulted in crashes. Use the threaded renderer instead.
- CC_SPRITE_DEBUG_DRAW - Not very useful, not fully implemented since 3.1.
- CC_PROFILER* - Use Apple's profiling tools instead.
- CCParticleSystemExamples - These were hard-coded particle system definitions. They are still included with the tests if you *really* want them back.

- CCAccelerometerDelegate - This had not been used by Cocos2D since before 3.0.
- TGA support - Use PNG instead.
- “ziputils” - Functions for working with gzip compressed data. Use CCFile to load data from compressed files instead.
- CCSpriteFrameCache - This will be made private.

Uncertain Changes

- Position types, transform delegates - Massive code cleanup is required in CCNode. This will affect the CCNode.*Type methods for sure.
- CCRenderTexture - This will likely be deprecated or removed and replaced by a much simpler subclass of CCTexture. We have a proposal at the end of this document.
- CCView, CCDirector - Allow multiple Cocos views at once, and make it play nicely with UIKit. This will have a lot of effects on CCDirector as well. The current plan is to make CCDirector a regular object instead of a singleton. So far this is working well. The [CCDirector sharedDirector] will be deprecated and replaced with a [CCDirector currentDirector] method that does more or less the same thing. Most code that relies on the director should continue to work as it always has.
- CCSpriteFrame and CCSprite texture rect coordinates were previously undocumented and very inconsistent. These will be made explicit to support automatic texture scaling, but which ones will change remains to be seen.

Resolution Handling

The biggest change is going to be how Cocos2D handles screen resolutions and asset loading. Cocos2D’s current scheme grew over many years as support for devices was added one at a time. This has made a bit of a mess, requiring too many images at different sizes with inflexible suffixes. We’ve promised a few times to add better image loading support to Cocos2D, and have failed to do so without requiring changes to the existing API. We couldn’t do it without requiring some changes to your existing code, so we decided to wait for the next major version. We are very excited that v4 will finally be able to offer this feature as it should make creating art that works across many devices easier than any other competing framework!

Here is the workflow we are trying to optimize for: You want a sprite in your game that is 100x100 points in size. The maximum content scale you intend to support is 4x, so you draw the image at 400x400 pixels and save it as Hero.png. No extra files, no suffixes, no hassles.

It’s ingrained in every Cocos2D developer that devices have a “content scale”. That 100x100 point sprite will be a different number of pixels on different devices. **So how does it load the 400x400 pixel Hero.png image correctly on all the different devices? It will rescale the image while it’s being loaded.** Every device has a content scale. This is, and will continue to be configurable, but there will be reasonable defaults. For instance an iPhone 4/5/6 or an iPad 2 will have a 2x content scale, an iPhone 6+ will have a 3x content scale, a retina iPad will have a

4x content scale, etc. When Hero.png is loaded on the iPhone 4, it will be rescaled to 200x200 pixels, and you don't have to do anything.

You will also be able to use optional suffixes such as "-2x" to explicitly tag a file's content scale. This can be useful to bypass the automatic scaling (because you want more control), or because you only need a low resolution image. Maybe you have a blurry background image that doesn't need more resolution than 1x on a retina iPad, or perhaps you have a texture for an effect shader that you don't want rescaled.

Upgrading a project to v4 may require renaming or deleting a lot of files. We might be able to help with this using some sort of compatibility mode, but it's too early to say for sure. SpriteBuilder projects should "just work" since it manages all the files for you.

If you want to read more, we have detailed proposals for CCFileUtils here, and resolution setups here. (Including information on Mac and Android)