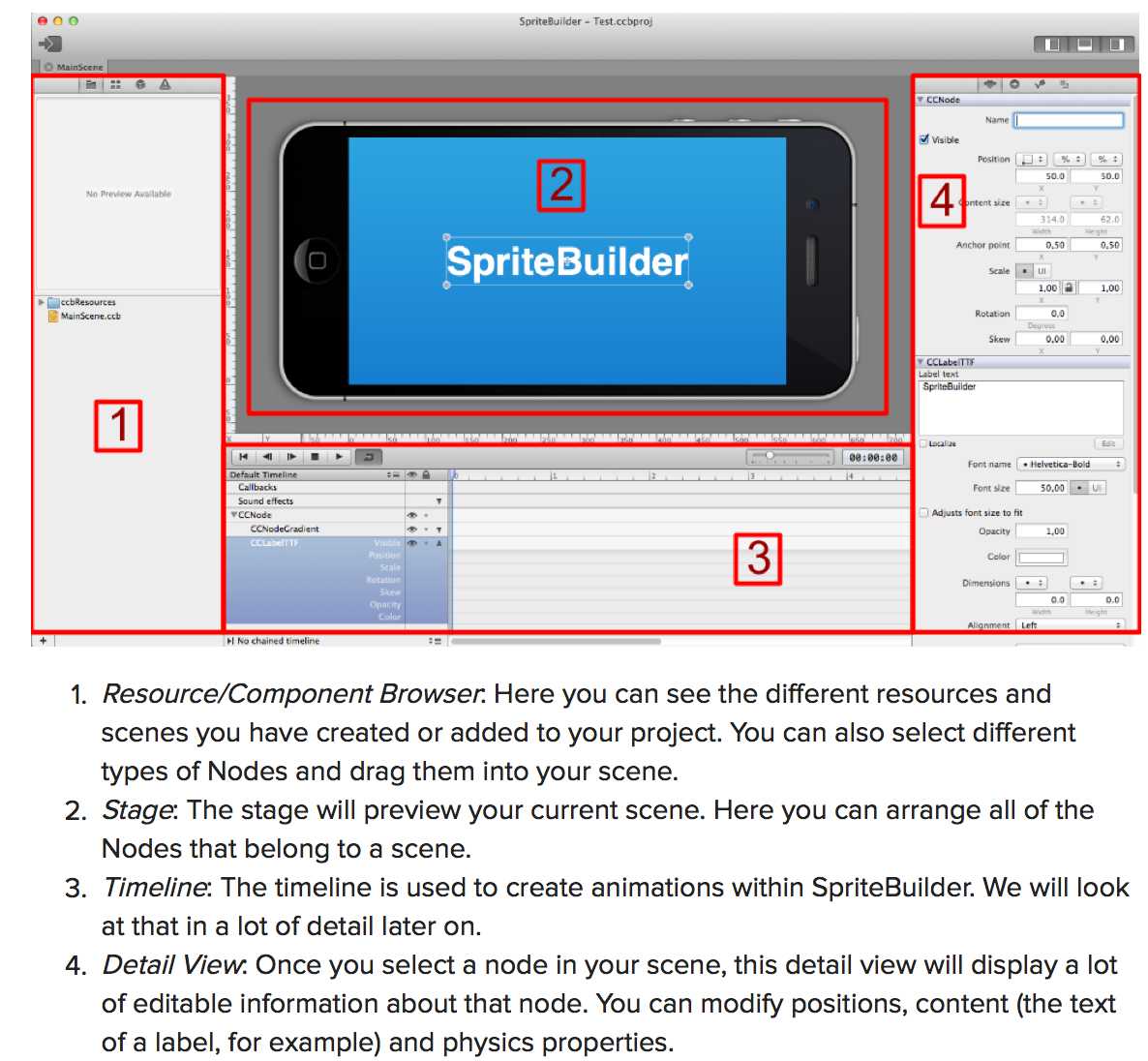
In general your workflow with SpriteBuilder will look like this:

* Create a new project in SpriteBuilder
* Add images and other resources to your SpriteBuilder project
* Create multiple .ccb files for the different scenes and objects in your game
* Add code connections to extend the behavior of these scenes and objects
* *Publish* your project in SpriteBuilder. This will update the Xcode project that is linked to your SpriteBuilder project
* Run your game from Xcode



**Loading Image into SpriteBuilder**

Thanks to SpriteBuilder's autoscaling you only need to provide the image with the highest resolution and the lower resolution images are generated automatically. If you've worked with Cocos2d before this means **no more regular and -hd files!**

**Steps**

1. **Copy pictures’ folder into the same directory as MainScene.ccb**
2. **“Command + ;” to open project setting. Change default scaling**
3. **Right click image folder, and select “Make Smart Sprite Sheet” to Enable smart Sprite Sheet to save memory on images**

SpriteBuilder is set up by default to downsize assets from a 4x resolution (double resolution of retina images). The Peeved Penguins assets are provided as 2x assets (retina resolution) so we have to change this setting for our project. Open File > Project Settings and change Default scaling to 2x (phonehd):

**Create Keyframe based Animation Object**

1. “Command + N” to set up a new **node** type .cbb file (interface file)
2. Drag image(s) into .cbb node file.
3. Edit timeline
   1. Add keyframes
   2. Zoom in by dragging the slider on the left of the timecode a bit to the right. Select the arm of the bear and create three keyframes for rotation at 0, 1 and 2 seconds. Do this by pressing r (on your keyboard) when the time marker is set at the mentioned times. Update the rotation value at each to -15, 20, -15 respectively
   3. Chain animation to itself so it will repeat when playing

**Create Physics Object**

**Steps**

1. In Sprite Builder,
   1. “Command + N” to create new **sprite** type .ccb
   2. Click on *CCSprite* in Timeline Pane to reveal detail view and setting for the object in the Right Pane
   3. Under *Item Properties* tab, *CCSprite* section, select image from asset from *sprite frame* drop down menu.
   4. Under *Item Code Connection* tab, enter name into *Custom class* field the same name as your .ccb file. (The "custom class" property of a .ccb file creates a link between the .ccb file in your SpriteBuilder project and an Objective-C class in your Xcode project. The Objective-C class needs to be created in xCode.).
   5. For Item Physics, click *Enable physics* checkbox. Select physics shape.
   6. Publish to Xcode
2. In Xcode,
   1. pen xcodeproj, create **objective-C class** file. Make the Class name same as the .ccb file
   2. make it the subclass of CCSprite. Call it the same name as your .ccb file.
   3. Save it in the **Source** folder.

#import "Penguin.h"

@implementation Penguin

- (id)init {

self = [super init];

if (self) {

CCLOG(@"Penguin created");

}

return self;

}

@end

**Create Menu**

**Steps**

1. In Sprite Builder,
   1. Create **MainScene.ccb**
   2. Drag background image to MainScene.ccb and set position to 50% and 50 % relative to parent to ensure that it is always centered liked this regardless of screen size.
   3. Add the play button.
      1. Go to *Node library View* tab on the Left Pane and drag a *Button* to the stage.
      2. In Right Pane, *Item Properties* tab, *CCSprite* section, select images for *Normal State* and *Highlighted State* from *Sprite frame* dropdown menu.
      3. In *CCControl* section, select preferred size.
      4. Set *Title* of the *CCButton*
   4. Do the following to ensure that when “play” button is pressed, the **play**method of the **MainScene** class will get invoked.
      1. Select the button object. Under *Item Code* Connections tab, enter “play” for *Selector* to link a method to play button. Set *Target* to “Document root”
      2. Select *CCNode* on the *Timeline* (can also click on the background image), click on *Item Code Connections* tab on the Right Pane, and make *Custom class* **MainScene***.*
   5. Publish to xCode
2. In Xcode,
   1. Add **play** method to **MainScene.m**

**Create Gameplay Scene**

**Steps**

1. In Sprite Builder,
   1. Create a *Layer* type 960x320 file called **Gameplay.ccb**
   2. Hit the *publish* button
2. In xCode,
   1. Create an objective-c class **Gameplay.m**, as a subclass of CCNode. Store in **source** folder.
   2. In **MainScene.m**, edit the **play** method to load **Gameplay** when play button is pressed.

#import "MainScene.h"

@implementation MainScene

- (void)play {

CCLOG(@"play button pressed");

CCScene \*gameplayScene = [CCBReader loadAsScene:@"Gameplay"];

[[CCDirector sharedDirector] replaceScene:gameplayScene];

}

@end

**Make Game**

**Steps**

1. In Sprite Builder,
   1. Drag **background.png** onto **Gameplay.ccb**. Set *position*and *Anchor Point* all to zero.
   2. Go to *Node Library View* tab on Left Pane, drag a *Sub File* node onto **Gameplay.ccb**, and select **Bear.ccb** as the the *CCB File* in the CCBFile Section of the *Item Properties.*
   3. Drag a *Physics Node* in **Gameplay.ccb** and select **Penguin.ccb** as the *CCB File*.
   4. Select root node (CCNode) from Timeline, under Item *Code Connections* tab*,* name *Custom class* “Gameplay” to establish code connection with **Gameplay.m**.
   5. Select the Physics Node object. In *Item* Properties, select “Doc root var” from the dropdown menu and enter “\_physicsNode”. (This will link this physics node to a variable called **\_physicsNode** in Gameplay.m.
   6. Similarly, create code connection for catapultarm.
   7. Hit the *publish* button
2. In xCode,
   1. Create a firing method.

#import "Gameplay.h"

@implementation Gameplay {

CCPhysicsNode \*\_physicsNode;

CCNode \*\_catapultArm;

}

// is called when CCB file has completed loading

- (void)didLoadFromCCB {

// tell this scene to accept touches

self.userInteractionEnabled = TRUE;

}

// called on every touch in this scene

- (void)touchBegan:(UITouch \*)touch withEvent:(UIEvent \*)event {

[self launchPenguin];

}

- (void)launchPenguin {

// loads the Penguin.ccb we have set up in Spritebuilder

CCNode\* penguin = [CCBReader load:@"Penguin"];

// position the penguin at the bowl of the catapult

penguin.position = ccpAdd(\_catapultArm.position, ccp(16, 50));

// add the penguin to the physicsNode of this scene (because it has physics enabled)

[\_physicsNode addChild:penguin];

// manually create & apply a force to launch the penguin

CGPoint launchDirection = ccp(1, 0);

CGPoint force = ccpMult(launchDirection, 8000);

[penguin.physicsBody applyForce:force];

}

@end

//////////////////////////////

//Print Statement Debugging

if (self) {

CCLOG(@"HEYYYYYYYYYY");

}

//////////////////////////////

(lldb) breakpoint set --file foo.c --line 12

**Create Game Level**

**Steps**

1. In Sprite Builder,
   1. Drag **ground.png** onto **Gameplay** scene (make it a *static* physics object) and make it a child of *CCPhysicsNode* (do this in timeline). **Every physics object need to be a child of *CCPhysicsNode***.
   2. Hit the *publish* button
   3. Create a level node:
      1. Drag a *Node* under *CCPhysicsNode*. In *Code Connections*, assign this node to “Root doc var” and call it **\_levelNode**.
      2. Create new folder called **Levels**. Create new *Layer* type ccb file called **Level1.ccb**. Make the size (490, 275)
2. In xCode,

**Remember**

* In xCode, the objective-C class is always a subclass of the class of file in SpriteBuilder