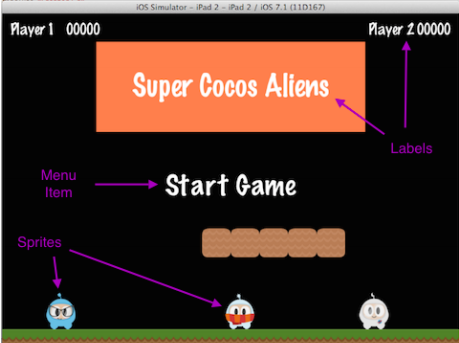
Cocos Basics

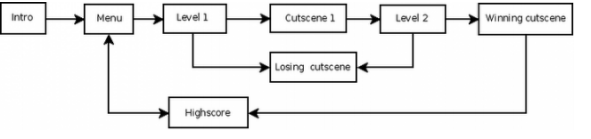
Objects:

* Scene,
* Node,
* Sprite,
* Menu
* Action

The below image is one Scene



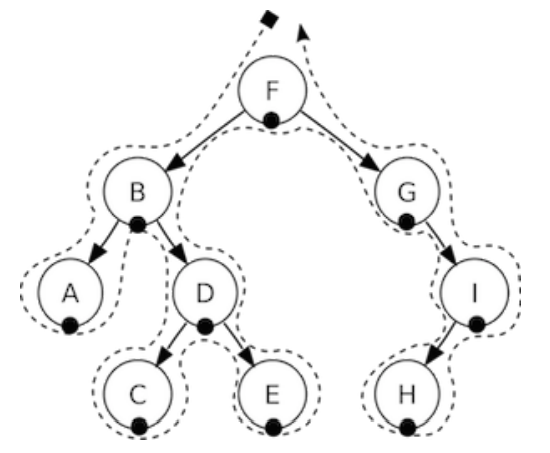
Workflow in Cocos:



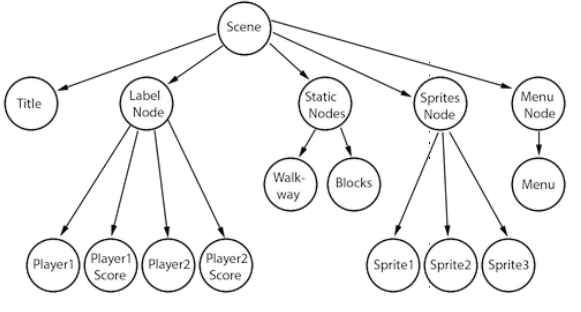
Data Structures:

Scene Graphs is a data structure which arranges graphical scene, cntains of Node OBjects in a Tree structure.

* This is decides which scene comes first then second and so on. (in-order walk)



The scene in Tree struture:



Scenes are consist of nodes, you decide the where they placed based on how you add them to the scane, (eg: Below 0 is left side above 0 is right side of tree, without number it will take 0 as position)

ex: scene.addChild(title\_node, -2);

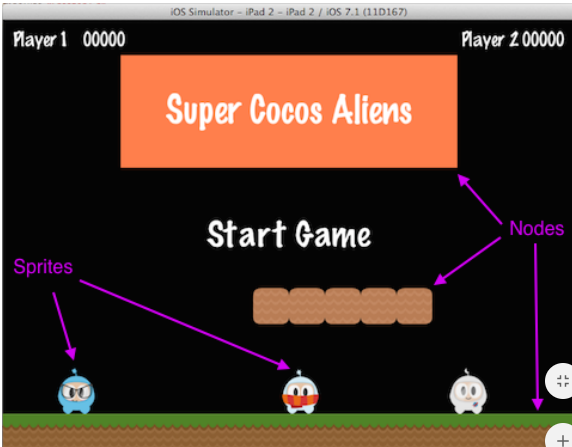
scene.addChild(label\_node);

scene.addChild(sprite\_node, 1);

**Sprites:**

Objects that are moved around the screen (Sprite is a node)

* Sprite is only sprite if you move em around
* Node if you dont move them around



Sprite Properties:

position, rotation, scale, opacity, color,

var mySprite = new cc.Sprite(res.mySprite\_png);

// this is how to change the properties of the sprite

mySprite.setPosition(cc.\_p(500, 0)); 13

mySprite.setRotation(40);

mySprite.setScale(2.0); // sets both the scale of the X and Y axis uniformly mySprite.setAnchorPoint(cc.\_p(0, 0));

**Actions:**

Allows the transofmarion of a node Object in time space.

Sprite is a subclass of a Node, you can make sequence with it if want to.

MoveBy, Rotate, Scale

Example:

var mySprite = new cc.Sprite(res.mySprite\_png);

// Move a sprite 50 pixels to the right, and 10 pixels to the top over 2 seconds.

var moveBy = new cc.MoveBy(2, cc.\_p(50,10));

mySprite.runAction(moveBy);

// Move a sprite to a specific location over 2 seconds.

var moveTo = new cc.MoveTo(2, cc.\_p(50,10));

mySprite.runAction(moveTo);

**Sequences and spawns:**

Sequence is multiple action objects run in an order. Can work in both direction

Example:

var mySprite = new cc.Node();

// move to point 50,10 over 2 seconds

var moveTo1 = new cc.MoveTo(2, cc.\_p(50,10));

// move from current position by 100,10 over 2 seconds

var moveBy1 = new cc.MoveBy(2, cc.\_p(100,10));

// move to point 150,10 over 2 seconds

var moveTo2 = new cc.MoveTo(2, cc.\_p(150,10));

// create a delay

var delay = new cc.DelayTime(1);

mySprite.runAction(Sequence.create(moveTo1, delay, moveBy1, delay.clone(), moveTo2));

**Running multiple Sequences is a spawn: (eg: Multiple actions for the bosses at the same time)**

Sequences played at the same time, some of them shorter, some of them longer.

var myNode = new cc.Node();

var moveTo1 = new cc.MoveTo(2, cc.\_p(50,10));

var moveBy1 = new cc.MoveBy(2, cc.\_p(100,10));

var moveTo2 = new cc.MoveTo(2, cc.\_p(150,10));

myNode.runAction(Spawn.create(moveTo1, moveBy1, moveTo2));

**Logging:**

Logging is built in cocos use log() for example debugging;

// a simple string log

("This would be outputted to the console");

// a string and a variable

string s = "My variable"; log("string is %s", s);

// a double and a variable

double dd = 42; log("double is %f", dd);

// an integer and a variable

int i = 6; log("integer is %d", i);

// a float and a variable

float f = 2.0f; log("float is %f", f);

// a bool and a variable

bool b = true;

if (b == true)

log("bool is true");

else

log("bool is false");

And, as expected, if you prefer you can use std::cout in place of log(), however, log() might offer easier formatting of complex output.

// a simple string

cc.log("This would be outputted to the console");

// outputting more than a simple string

var pos = cc.\_p(sender.x, sender.y);

cc.log("Position x: " + pos.x + ' y:' + pos.y);

**General informations**

Sprites:

Define: 2D images with properties of rotation, position, scale, color, Can be created by Tools

eg: Cocos creator.

Formats: PNG, JPEG, TIFF

Code to creat the whole picture as Sprite:

var mySprite = new cc.Sprite(res.mySprite\_png);

Code to creat a partion of the picture as Sporite: (Starts at the top left corner)

var mySprite = new cc.Sprite(res.mySprite\_png, cc.rect(0,0,40,40));

**Sprite Sheet:**

All image can be placed ont he same Sprite sheet and save up resource.

Code for Load Sprite Sheet:

// load the Sprite Sheet

var spritecache = cc.SpriteFrameCache;

// the .plist file can be generated with any of the tools mentioned below spritecache.addSpriteFramesWithFile(res.sprites\_plist);

**Load an image from SpriteSheet:**

// Our .plist file has names for each of the sprites in it. We'll grab

// the sprite named, "Blue\_Front1" from the sprite sheet:

var mysprite = cc.Sprite.createWithSpriteFrameName(res.mySprite\_png);

**Image creation from Sprite Sheet:**

Loading a Sprite Sheet: (into the SpriteFrameCache)

// load the Sprite Sheet

var spritecache = cc.SpriteFrameCache;

// the .plist file can be generated with any of the tools mentioned below spritecache.addSpriteFramesWithFile(res.sprites\_plist);

**Create the image from SpriteFrameCache**

// Our .plist file has names for each of the sprites in it. We'll grab

// the sprite named, "Blue\_Front1" from the sprite sheet:

var mysprite = cc.Sprite.createWithSpriteFrameName(res.mySprite\_png);

**Creating a Sprite from a SpriteFrame.**

// this is equivalent to the previous example,

// but it is created by retrieving the SpriteFrame from the cache.

var newspriteFrame = cc.SpriteFrameCache.getSpriteFrameByName(res.sprites\_plist);

var newSprite = cc.Sprite.createWithSpriteFrame(newspriteFrame);

**Actions**

Actions are med a Node to perfomr a change to its own properties in run time. Any objects who’s base class is Node can have an Action Object.

**Move a Sprite to a position.**

// Move sprite to position 50,10 in 2 seconds.

var moveTo = new cc.MoveTo(2, cc.\_p(50, 10));

mySprite1.runAction(moveTo);

// Move sprite 20 points to right in 2 seconds

var moveBy = new cc.MoveBy(2, cc.\_p(20,0));

mySprite2.runAction(moveBy);

**Difference between By and To:**

By: Relative tot he current state of the node.

To: Absolute, ignores the current state of the node.

var mySprite = new cc.Sprite(res.mysprite\_png);

mySprite.setPosition(cc.\_p(200, 256));

// MoveBy - lets move the sprite by 500 on the x axis over 2 seconds

// MoveBy is relative - since x = 200 + 200 move = x is now 400 after the move

var moveBy = new cc.MoveBy(2, cc.\_p(500, mySprite.y));

// MoveTo - lets move the new sprite to 300 x 256 over 2 seconds

// MoveTo is absolute - The sprite gets moved to 300 x 256 regardless of

// where it is located now.

var moveTo = new cc.MoveTo(2, cc.\_p(300, mySprite.y));

// Delay - create a small delay

var delay = new cc.DelayTime(1); var seq = new cc.Sequence(moveBy, delay, moveTo); mySprite.runAction(seq);

**Basic Actions and how to run them:**

Its a singular action, thus accomplishing a single objective.

var mySprite = new cc.Sprite(res.mysprite\_png);

// Move a sprite to a specific location over 2 seconds.

var moveTo = new cc.MoveTo(2, cc.\_p(50, 0));

mySprite.runAction(moveTo);

// Move a sprite 50 pixels to the right, and 0 pixels to the top over 2 seconds.

var moveBy = new cc.MoveBy(2, cc.\_p(50, 0));

mySprite.runAction(moveBy);

**Rotate:**

Rotate a Node clockwise over 2 seconds.

var mySprite = new cc.Sprite(res.mysprite\_png);

// Rotates a Node to the specific angle over 2 seconds

var rotateTo = new cc.RotateTo(2.0, 40.0);

mySprite.runAction(rotateTo);

// Rotates a Node clockwise by 40 degree over 2 seconds

var rotateBy = new cc.RotateBy(2.0, 40.0);

mySprite.runAction(rotateBy);

**Scale:**

var mySprite = new cc.Sprite(res.mysprite\_png);

// Scale uniformly by 3x over 2 seconds

var scaleBy = new cc.ScaleBy(2.0, 3.0);

mySprite.runAction(scaleBy);

// Scale X by 5 and Y by 3x over 2 seconds

var scaleBy = new cc.ScaleBy(2.0, 3.0, 3.0);

mySprite.runAction(scaleBy);

// Scale to uniformly to 3x over 2 seconds

var scaleTo = new cc.ScaleTo(2.0, 3.0);

mySprite.runAction(scaleTo);

// Scale X to 5 and Y to 3x over 2 seconds

var scaleTo = new cc.ScaleTo(2.0, 3.0, 3.0);

mySprite.runAction(scaleTo);

**Fade In/Out:**

It modifies the opacity from 0 to 255. The reverse of this action is FadeOut var mySprite = new cc.Sprite(res.mysprite\_png);

// fades in the sprite in 1 seconds

var fadeIn = new cc.FadeIn(1.0);

mySprite.runAction(fadeIn);

// fades out the sprite in 2 seconds

var fadeOut = new cc.FadeOut(2.0);

mySprite.runAction(fadeOut);

Tint:

Its a Node which implements a NodeRGB protocol to make a custom Tint

var mySprite = new cc.Sprite(res.mysprite\_png);

// Tints a node to the specified RGB values

var tintTo = new cc.TintTo(2.0, 120.0, 232.0, 254.0);

mySprite.runAction(tintTo);

// Tints a node BY the delta of the specified RGB values.

var tintBy = new cc.TintBy(2.0, 120.0, 232.0, 254.0);

mySprite.runAction(tintBy);

**Animate:**

It makes possible to make a Flipbook Animation with your sprite. (Replacing Frames at a set of interval time)

var mySprite = new Sprite(res.mysprite\_png);

// now lets animate the sprite we moved.

var animFrames;

animFrames.push(new cc.SpriteFrame(res.Blue\_Front1\_png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Front2.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Front3.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Left1.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Left2.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Left3.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Back1.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Back2.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Back3.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Right1.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Right2.png), cc.Rect(0,0,65,81)));

animFrames.push(new cc.SpriteFrame(res.Blue\_Right3.png), cc.Rect(0,0,65,81)));

// create the animation out of the frames

var animation = cc.Animation.createWithSpriteFrames(animFrames, 0.1);

var animate = new cc.Animate(animation);

// run it and repeat it forever

mySprite.runAction(cc.RepeatForever(animate));

**Easing:**

It makes the animation smoother by modifying the acceleration. Good for fake Physic in your game. Also good for animate Menues and Buttons in your game.

Example code only in C++ by official Programmers Guide of Cocos

// create a sprite

auto mySprite = Sprite::create("mysprite.png");

// create a MoveBy Action to where we want the sprite to drop from.

auto move = MoveBy::create(2, Vec2(200, dirs->getVisibleSize().height -

newSprite2->getContentSize().height));

auto move\_back = move->reverse();

// create a BounceIn Ease Action

auto move\_ease\_in = EaseBounceIn::create(move->clone() );

// create a delay that is run in between sequence events

auto delay = DelayTime::create(0.25f);

// create the sequence of actions, in the order we want to run them

auto seq1 = Sequence::create(move\_ease\_in, delay, move\_ease\_in\_back,

delay->clone(), nullptr);

// run the sequence and repeat forever.

mySprite->runAction(RepeatForever::create(seq1));

**Sequences and how to run them:**

Series of Actions, can be any amount of Action objects, Functions or Sequences. In sequence you can call Functions important to remember that.

Jump -> callbackJump() -> Rotate -> callbackRotate() is the workflow of the below code.

auto mySprite = Sprite::create("mysprite.png");

// create a few actions.

auto jump = JumpBy::create(0.5, Vec2(0, 0), 100, 1);

auto rotate = RotateTo::create(2.0f, 10);

// create a few callbacks

auto callbackJump = CallFunc::create([](){

log("Jumped!");

});

auto callbackRotate = CallFunc::create([](){

log("Rotated!");

});

// create a sequence with the actions and callbacks

auto seq = Sequence::create(jump, callbackJump, rotate, callbackRotate, nullptr);

// run it

mySprite->runAction(seq);

**Spawn:**

Similar to Sequence, All Sequence will run at the same time. Can contain any number of Action objects or another Spawn. Same result as running multiple runAction(). Mostly used for special effect, otherwise not really achivable.

Example, given:

// create 2 actions and run a Spawn on a Sprite

auto mySprite = Sprite::create("mysprite.png");

auto moveBy = MoveBy::create(10, Vec2(400,100));

auto fadeTo = FadeTo::create(2.0f, 120.0f);

Using a Spawn:

// running the above Actions with Spawn.

auto mySpawn = Spawn::createWithTwoActions(moveBy, fadeTo);

mySprite->runAction(mySpawn);

Run Action statements:

// running the above Actions with consecutive runAction() statements.

mySprite->runAction(moveBy);

mySprite->runAction(fadeTo);

**Clone:**

You can apply an Action to multiple Nodes, by using Clone(), The reason to have clone is that the Actions have internal states during runing the actions the properties of the Node objects are changing, Without clone you wont know the properties of an action. (Basically doble the current state to store it).

**Example:**

Let’s hash through an example, say you have a heroSprite and it has a position of (0,0). If you

run an Action of:

MoveBy::create(10, Vec2(400,100));

This will move heroSprite from (0,0) to (400, 100) over the course of 10 seconds. heroSprite

now has a new position of (400, 100) and more importantly the Action has this position in it’s

internal state. Now, say you have an emenySprite with a position of (200, 200). If you were

to apply this same:

MoveBy::create(10, Vec2(400,100));

to your enemySprite, it would end up at a position of (800, 200) and not where you thought it

would. Do you see why? It is because the Action already had an internal state to start from

when performing the MoveBy. Cloning an Action prevents this. It ensures you get a unique

version Action applied to your Node.

**Correctly, using clone()!:**

// create our Sprites

auto heroSprite = Sprite::create("herosprite.png");

auto enemySprite = Sprite::create("enemysprite.png");

// create an Action

auto moveBy = MoveBy::create(10, Vec2(400,100));

// run it on our hero

heroSprite->runAction(moveBy);

// run it on our enemy

enemySprite->runAction(moveBy->clone()); // correct! This will be unique

**Reverse:**

It can rune an action in reverse/backwards mode its also manipulating the properties of Sequences or Spawns. Most Action and Sequences are reversible.

// reverse a sequence, spawn or action

mySprite->runAction(mySpawn->reverse());

Most Action and Sequence objects are reversible!

It’s easy to use, but let’s make sure we see what is happening. Given:

// create a Sprite

auto mySprite = Sprite::create("mysprite.png");

mySprite->setPosition(50, 56);

// create a few Actions

auto moveBy = MoveBy::create(2.0f, Vec2(500,0));

auto scaleBy = ScaleBy::create(2.0f, 2.0f);

auto delay = DelayTime::create(2.0f);

// create a sequence

auto delaySequence = Sequence::create(delay, delay->clone(), delay->clone(),

delay->clone(), nullptr);

auto sequence = Sequence::create(moveBy, delay, scaleBy, delaySequence, nullptr);

// run it

newSprite2->runAction(sequence);

// reverse it

newSprite2->runAction(sequence->reverse())

http://www.cocos2d-x.org/docs/ProgrammersGuide.pdf