Criando animações com Lua e Corona SDK para Android

29. maio 2011 Experiências, Lua, Mobile

Em meu último post eu compartilhei minhas últimas descobertas (pois é, estou meio atrasado) do mundo do desenvolvimento mobile, onde falei a respeito de Lua, uma ótima linguagem de script para codificar para dispositivos móveis e sobre o Corona SDK, sem sombra de dúvida o melhor kit de desenvolvimento mobile multiplataforma (se é que existe outro). Continuando meus testes com ambos, a linguagem e a plataforma, obtive alguns resultados bem interessantes, que eu trago hoje em primeira mão. A idéia é mostrar aos leitor do blog como desenvolver uma animação simples de um caça-recompensas do oeste americano galopando seu alazão pelo deserto selvagem. Na verdade é apenas o Cormano do game Sunset Riders correndo à be dos trilhos de trem da fase 2 do game...

Criando o Projeto

Primeiro de tudo você deve ter baixado na sua máquina o Lua e o Corona SDK. O primeiro pode ser encontrado em Lua.org e o segundo emcoronalabs.com. Uma vez que tenha ambos instalados em sua máquina, você deve criar uma pasta para seu projeto com o nome que mais lhe agrada. Eu pus o nome de SunsetRiders, e dentro desta pasta, deve-se criar o arquivo main.lua, que conterá todo o código de nossa aplicação. Este arquivo deve ser aberto com o Corona Simulator para ver como anda sua aplicação. A título curiosidade, o Corona SDK assume aparência de Android caso rode no Windows e a aparência de iPhone rode em Mac. Essas aparências são ajustáveis no menu Windows->View as, porém no Windows se limitam apenas a dispositivos com Android (como NexusOne e Milestone). Isso não muda nada para nós neste momento.

Configurando a Aplicação

Como nossa aplicação será um cavaleiro galopando, necessitaremos que a animação rode na horizontal. Existem dua smaneiras de fazer isso: uma delas é indo no menu Hardware e mandando o Corona Simulato rodar para direita e a outra é criando um arquivo de configuração de compilação (build.settings). A segunda opção é mais bacana pois fará com que o Simulator já fique na posição certa sempre que der um Ctrl + R p recompilar o projeto. Crie um arquivo build.settings e cole o seguinte conteúdo dentro:

```
settings = {
    orientation = { default = "landscapeRight" }
}

```

Você precisará também de um arquivo de configuração para que sua animação rode sempre em tela cheia. Você consegue isso criando um arquivo config.lua e adicionando o código a seguir, que determina a resolu da animação e se ela usará zoom ou não.

```
application =
{
content =
{
width = 320,
height = 480,
scale = "zoomEven"
},
}
```

Pronto, sua animação está devidamente configurada e prontapara ser codificada. It's time to pay!

Sprite Sheet

Antes de codificar qualquer coisa, eu tive de sair à cata dos sprites do Cormano, para poder animá-lo via Lu Para quem não sabe o que é uma sprite sheet ou sequer uma sprite (não, não é a bebida de limão da Coca cola), aqui vai uma breve introdução: sprites são os chamados "bonequinhos" dos games 2D, e sprite sheet (folha de sprites) nada mais é do que uma imagem única contendo todos os movimentos do bonequinho. O seja: ele caminhando para um lado, para o outro, atirando, etc. Tudo depende do game. No meu caso, quer Cormano galopando no deserto, então tratei de catar na net a sprite sheet dele à cavalo. O Corona permite animar sprites em tamanhos diferentes, mas tudo fica mais simples se cada sprite for do mesmo tamanho, isso dei uma editada nas imagens menores para que todas ficassem com a mesma largura. Usei o software Fireworks CS5 para isso, mas qualquer software de edição de imagens serve, não esquecendo que o fundo deve ser transparente e a extensão deve ser PNG. O resultado da minha sprite sheet segue abaixo:

Animando seu 1º sprite

Uma vez que você tenha a spritesheet em mãos, com fundo transparente, extensão .PNG e imagens com mesmo tamanho, você pode começar a animá-la usando Lua. É tudo muito simples e o código abaixo most

como fazê-lo, considerando que o nome da minha spritesheet é cormano.png.

```
require "sprite"<br><br>-- cormano galopando
local sheet1 = sprite.newSpriteSheet( "cormano.png", 95

local spriteSet1 = sprite.newSpriteSet(sheet1, 1, 6)
sprite.add( spriteSet1, "cormano", 1, 6, 500, 0 ) -- rc

local instance1 = sprite.newSprite( spriteSet1 )
instance1.x = display.contentWidth / 4 + 40
instance1.y = 225--distância do cavalo ao chão
instance1.xScale = 1.2
instance1.yScale = 1.2

instance1:prepare("cormano")
instance1:play()
```

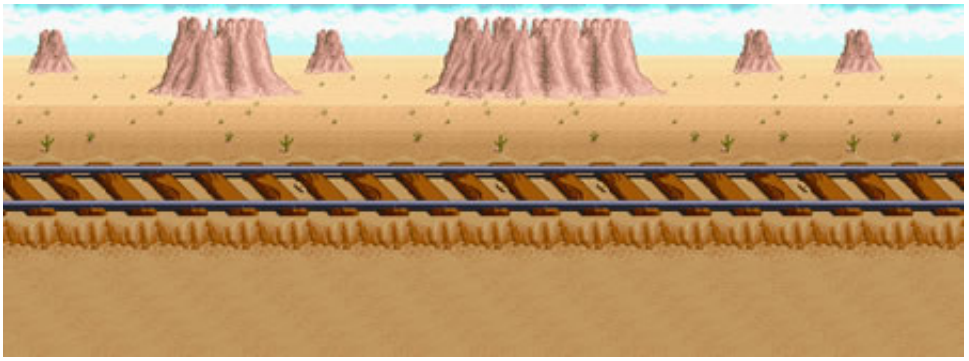
Note que a primeira linha de código é a mais importante. É nela onde definimos qual é nossa spritesheet, q a largura e qual a altura de cada uma das sprites dentro da spritesheet (por isso que devem ter o mesmo tamanho). Na segunda linha de código, dizemos qual o índice da primeira sprite da animação e qual o índic segunda. Na terceira linha de código, dizemos quantos frames (quadros de sprite) vamos animar a cada período de tempo. Neste caso, optei por realizar toda a animação a cada meio segundo (500ms). Se você diminuir o tempo, o galope ficará mais intenso.

No próximo parágrafo defini o posicionamento do Cormano na tela do dispositivo e aumentei um pouco o se tamanho (em 20%). E por fim, no último parágrafo, botei Cormano a galopar. Se você abrir este projeto no Corona Simulator, verá Cormano galopando em uma tela preta. É hora de colocar um cenário ao fundo!

Animando o Cenário

Já fizemos um grande avanço colocando o cowboy mexicano a correr pela tela de seu dispositivo, agora va colocar o famoso cenário de velho oeste presente na segunda fase do game, onde Cormano e seus amigos correm ao lado dos trilhos de trem para encontrar Hawkeye, um foragido da justiça. Antes que eu me empo falando deste clássico da minha infância, vamos ao que interessa. Busquei no Google Imagens uma image do cenário de Sunset Riders, onde aparecem os trilhos, o deserto e as montanhas ao fundo. Como essa imagem deve ficar correndo ao fundo da animação para dar a impressão de que Cormano está galopando p algum lugar, usaremos um recurso bastante comum nos games 2D de corrida, onde o cenário fica se move

repetidamente com a mesma imagem de fundo, dando a impressão de movimento. A imagem que utilizei p
o fundo é esta:



Como essa imagem correrá da esquerda para direita incessantemente, eu dei uma editada nela para que o
lado direito, quando "colado" ao lado esquerdo, não houvesse quebras de continuidade, parecendo que é u
imagem com width infinito. O código para que a imagem fique correndo da esquerda para direita é bem sim
e consiste apenas em criar duas imagens e juntá-las. Criei um evento de atualização da cena onde o x da
imagem é incrementado e quando a mesma termina, inicia-se o processo novamente do zero, alteranando
objetos de imagem entre o original e a cópia. É mais dificil de explicar do que de fazer, então segue o códig

```
require "sprite"
1
2  local baseline = 280
3
4  -- Trilhos
5  -- O dobro de trilhos para fazer a animação
6  -- Quando um dos trilhos termina, aparece o outro
7  local trilhos = display.newImage( "trilhos.jpg" )
8  trilhos:setReferencePoint( display.CenterLeftReferenceP
9  trilhos.x = 0
10 trilhos.y = baseline-115
11 trilhos.xScale = 1.3
12 trilhos.yScale = 1.3
13 local trilhos2 = display.newImage( "trilhos.jpg" )
14 trilhos2:setReferencePoint( display.CenterLeftReference
15 trilhos2.x = 610
16 trilhos2.y = baseline-115
17 trilhos2.xScale = 1.3
18 trilhos2.yScale = 1.3
19 -- cormano galopando
20
```

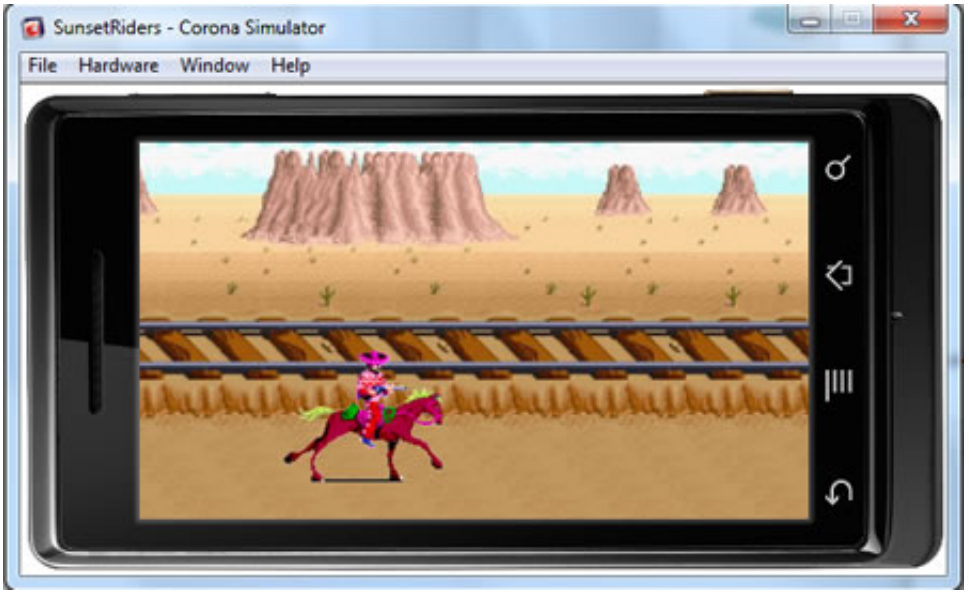
Note que este código vai depois do require("sprite") (que importa as bibliotecas de manipulação de sprites d
Corona e antes do código do Cormano galopando que fizemos anteriormente. Isso porque a ordem em que

objetos são instanciados define a sobreposição dos mesmos. Se colocarmos o cenário de fundo depois do Cormano, ele será sobreposto e não aparecerá na tela do seu iCoisa. Bom, o código acima não anima o cenário, apenas o apresenta na tela. Para que ele fique correndo da esquerda para direita, criei uma função move que adiciono ao listener enterFrame, que é executado quando a aplicação aparece na tela do iCoisa. Nesta função, que é apresentada abaixo, realizo um cálculo com base no tempo de execução do sistema (copiei este cálculo de um outro exemplo do Corona, então não sei realmente como funciona) e incremento posição dos trilhos, dando a impressão de movimento. Note que este código de movimento, deve ser colocado depois do código do Cormano galopando, para encerrar seu main.lua com chave de ouro.

```lua
-- movendo os elementos
local tPrevious = system.getTimer()
local function move(event)
    local tDelta = event.time - tPrevious
    tPrevious = event.time

    local xOffset = ( 0.2 * tDelta )

    trilhos.x = trilhos.x - xOffset
    trilhos2.x = trilhos2.x - xOffset

    if (trilhos.x + trilhos.contentWidth) < 0 then
        trilhos:translate( 610 * 2, 0)
    end
    if (trilhos2.x + trilhos2.contentWidth) < 0 then
        trilhos2:translate( 610 * 2, 0)
    end
end

-- faz tudo acontecer
Runtime:addEventListener( "enterFrame", move );
```

Concluindo

Se você fez tudo certo até agora (ous e simplesmente baixou os fontes disponíveis no final deste post) você deverá ver nuestro amigo Cormano galopando faceiro pelo deserto do Wild West americano. Sim, é mais uma aplicação boba, mas já dá pra ter uma idéia do poder do Corona SDK e da facilidade de lidar com sua API usando Lua. Além disso é bem mais bacana que os exemplos da Ansca Mobile onde tem o puma correndo atrás do ET... Até o próximo post!



Download do código completo - SunsetRiders.zip (76,77 kb)