**目录**

[我的小记录： 2](#_Toc7336)

[Cocos2d js 帧动画和补间动画的播放 2](#_Toc9970)

[代码控制播放帧动画 2](#_Toc8624)

[Cocos2d js 骨骼动画播放 4](#_Toc5591)

[Skeleton层上的node节点运行其上的action。播放基本同cocostudio上帧动画的播放。 4](#_Toc14511)

[代码加载骨骼动画的方法 4](#_Toc3308)

[Cocostudio 骨骼层Skeleton层的骨骼编辑 5](#_Toc27100)

[Button的相关API 6](#_Toc22089)

[Cocostudio上的Armature控件 6](#_Toc5819)

[在浏览器和手机上友好输出的函数改造 7](#_Toc21994)

[依据平台添加相应的事件类型 7](#_Toc8633)

[定时器：只能依赖节点触发 7](#_Toc16739)

[取消定时器： 7](#_Toc15252)

[暂停/恢复定时器; 7](#_Toc23077)

[定时器改造 8](#_Toc19699)

[遮罩节点的使用 9](#_Toc26420)

[分批处理--SpriteBatchNode 9](#_Toc26083)

[缓存池pool的使用 9](#_Toc20789)

[位图缓存--BakeLayer 10](#_Toc10767)

[视察滚动背景（ParallaxNode） 10](#_Toc7229)

[瓦片地图--cc.TMXTiledMap 10](#_Toc10226)

[粒子系统 编辑器ParticleEditor 10](#_Toc5322)

[SQLite实现更安全、更强大的数据存储 11](#_Toc14927)

**cocos2d js 知识点归纳**

**我的小记录：**

1. ccui.helper.doLayout（widget） 主动调用 实现子节点偏移 ,是相对父节点的，其实是为了控件自适应父节点大写而做的一个操作 。

1. cocos2d js 热更新：

IMG_256http://www.cnblogs.com/Siegel/p/5927299.html

1. 文字移动的思路是：每次都update公告的CCLabelTTF的坐标，为了让它从右往左进行移动，右边栏出来，左边栏消失，需要设置一下CCLabelTTF的可显示区域，CCLabelTTF::setTextureRect函数正是设置Label的可显示区域，因此左右边界需要特殊处理.

Cocos2d js 帧动画和补间动画的播放

Cocostudio制作的帧动画和补间动画的播放（包括骨骼动画），都是ccs.load()加载的UI层的节点node属性 运行 （runAction()）动作action属性，播放方法：它的action属性调用play（）函数。

//cocostudio 制作的帧动画播放  
var mmNode = ccs.load("res/emmAnimationNode.json");  
this.addChild(mmNode.node);  
  
(mmNode.node).runAction(mmNode.action);  
mmNode.action.gotoFrameAndPlay(0, 60, true);  
  
// var action = cc.sequence(this.mainHall.action, this.mainHall.action.reverse()).repeatForever();  
  
this.mainHall.node.runAction(this.mainHall.action);  
//通过设置开始帧，结束帧，是否循环播放等进行播放动画  
// this.mainHall.action.gotoFrameAndPlay(0, 50, true);  
//通过play函数播放指定名称的动画  
this.mainHall.action.play("animation0", false);  
//this.mainHall.action.play("animation1", true);

代码控制播放帧动画

在这之前，一定要先把图片加载到内存当中去；其实你也可以不加，但是需要换另外一个函数就行；

//将plist问价加载到内存当中

cc.spriteFrameCache.addSpriteFrames(res.play\_plist);

第一种方法:

创建一个空的精灵

var sp = new cc.Sprite();

sp.setTag(1000);

sp.setPosition(500,400);

this.addChild(sp,1);

//定义一个数组 后面加中括号

var allFrame = [];

//for语句载入5个动画图片

for(var i = 1; i < 4; i++){

//加载针动画，rect四个参数，前两个X,Y的坐标，默认0就OK,

//后面两个参数传图片的宽度和高度

var str = "enemyLeft1\_"+i+".png";

var allf = cc.spriteFrameCache.getSpriteFrame(str);//new cc.SpriteFrame()

if(!allf){

cc.log("@@@@@@@");

}

allFrame.push(allf);

}

//每隔0.03秒切换一张图片

var animation = new cc.Animation(allFrame,0.1);

//把所有的动画连接起来进行播放

var animate = new cc.Animate(animation);

//重复的执行摸个动作

var action = animate.repeatForever();//new cc.RepeatForever(animate)

//用精灵来执行针动画，让针动画跑起来

sp.runAction(action);

第二种方法：

同样要创建一个空的精灵

var sp = new cc.Sprite();

sp.setTag(1000);

sp.setPosition(500,400);

this.addChild(sp,1);

for(var i = 1;i<=4;i++){

var frameName = "enemyLeft1\_"+i+".png";

animation.addSpriteFrame(frameName);

}

animation.setDelayPerUnit(0.1);

animation.setRestoreOriginalFrame(true);

var action = cc.animate(animation).repeatForever();

sp.runAction(action);

Cocos2d js 骨骼动画播放

Skeleton层上的node节点运行其上的action。播放基本同cocostudio上帧动画的播放。

//骨骼动画的播放  
var skeleton = ccs.load("res/Skeleton.json");  
this.addChild(skeleton.node);  
播放方法1：skeleton.node.runAction(skeleton.action);  
skeleton.action.gotoFrameAndPlay(0, 70, true);

代码加载骨骼动画的方法

Cocostudio自己导出的骨骼动画的加载方式：

ccs.armatureDataManager.addArmatureFileInfo("res/heroArmature/Hero0.png", "res/heroArmature/Hero0.plist", "res/heroArmature/Hero.json");  
  
var armature =new ccs.Armature("Hero");  
  
this.addChild(armature);  
armature.getAnimation().setSpeedScale(24 / 60);  
armature.getAnimation().play("run");  
armature.x = 80;  
armature.y = 480;  
armature.setScale(0.5);

ccs.armatureDataManager.addArmatureFileInfo

("res/dragonbones/skeleton.png",

"res/dragonbones/skeleton.plist",

"res/dragonbones/skeleton.json");

var armature = new ccs.Armature("Dragon");//动画名字

This.addchild(armature);

armature.getAnimation().play("stand");

armature.getAnimation().setSpeedScale(24/60);

1. ccs.armatureDataManager.addArmatureFileInfo(res.COwboy\_png0, res.CowBoy\_plist0, res.CowBoy\_exportjson);//动画文件信息放到动画数据管理器里面
2. **var** armature = ccs.Armature.create("Cowboy");//从动画数据管理器里面创建名叫Cowboy的动画对象
3. armature.getAnimation().play("Walk");//播放Cowboy里面名叫Walk的动画
4. armature.x = size.width \* 0.1;
5. armature.y = size.height / 2;
6. armature.setScale(0.5);

**this**.\_sprite.getAnimation().setMovementEventCallFunc(**this**.callBackEndAttack,**this**);

Cocostudio 骨骼层Skeleton层的骨骼编辑**：**

1. 在形体模式下，将资源拖到渲染区，并摆好位置。
2. 开始创建骨骼：选中快捷菜单栏中创建骨骼的按钮，在角色身体的各部位单击左键并拖动鼠标画出骨骼。
3. 将资源素材与骨骼绑定。
4. 当绑定好所有素材和骨骼后，右击骨骼选择“绑定父关系”，再选择该骨骼的父骨骼（选择的骨骼同样会高亮显示），这样依次根据身体部位的“父子”层级关系设定好。
5. 切换到动画模式，选中相应控件，添加关键帧，修改属性。

这里我们通过ArmatureManager动画数据管理器来加载动画。ArmatureManager本身是一个单例，它管理整个场景中的Armature。而Armature则封装了我们播放动画需要的Animation。由此可见，这是一个三层缔结的层次结构。其中ArmatureManager最大，然后是Armature，最后是Animation。

ArmatureDataManager::getInstance()->addArmatureFileInfo("NewAnimation0.png","NewAnim

Armature \*armature = Armature::create( "NewAnimation");

// 播放动画

armature->getAnimation()->play("walk");

**this**->addChild(armature);

可以像上面代码一样指定动画名来播放动画，也可以通过指定动作编号来播放动画，如下：

|  |  |
| --- | --- |
| 1 | armature->getAnimation()->playWithIndex(0); |

Button的相关API

//Button的相关功能调用  
var btn = mo.sceneHelper.MySeekWidgetByName(this.mainHall.node, "Button\_charge");  
btn.setBright(false); //按钮明亮/灰暗变化  
// btn.setTouchEnabled(false);//按钮禁用  
btn.setPressedActionEnabled(false);//是否 按下效果  
cc.log(btn.getDescription());//获取控件类名

Cocostudio上的Armature控件

**在cocostudio上可以直接拖拽添加Armature控件，可以导入xxx.ExportJson的文件。对其播放的代码控制如下：**

*//Armature控件---人物移动*this.renwu = mo.sceneHelper.MySeekWidgetByName(this.mainHall.node, "ArmatureNode\_2");  
*//Armature控件 播放控制* ***todo。。****//this.renwu.getAnimation().play("stand");*this.renwu.getAnimation().stop();

在浏览器和手机上友好输出的函数改造**：**

Var trace=function(){

Cc.log(Array.prototype.join.call(arguments,”,”));

};

利用了JS动态参数的特性，可以接受任何个数的参数，最终按数组的方式输出信息。

依据平台添加相应的事件类型**：**

**mouse touches keyboard accelerometer**

If(‘mouse’in cc.sys.capabilities)

{

Cc.eventManager.addListener{

event:cc.EventListener.MOUSE,

onMouseDown:function(event){

var pos=event.getLocation();

var target=event.getCurrentTarget();

if(event.getButton()===cc.EventMouse.BUTTON\_RIGHT)

{}

}

...

}

}

定时器：只能依赖节点触发

1. node.scheduleOnce(callback,delay)

故定时器的逻辑应该放到节点的子类中实现。

b.node.schedule(callback,interval,repeat,delay)

delay表示第一次触发前的延迟时间，秒为单位

无限循环：cc.REPEAT\_FOREVER

取消定时器：

node.unscheduleUpdate()-->scheduleUpdate()

node.unschedule()-->scheduleOnce()和schedule()

暂停/恢复定时器;

node.pause() node.resume()

定时器改造

**schedule的实现基于帧频，当帧频降低时，误差问题越来越严重。所以，schedule只能用于动画的控制，不可以用于逻辑处理或者网络同步的计算上。**

var BetterScheduleLayer = cc.Layer.extend({  
  
 ctor: function () {  
 this.\_super();  
  
 var startTime = Date.now();  
 var count = 0;  
 this.schedule2(function () {  
 var timePass = Date.now() - startTime;  
 count++;  
 var delta = timePass - (count \* 100);  
 cc.log("timepass", timePass, "total delta", delta, "count", count);  
 }, 1);  
  
 *//this.scheduleUpdate();* },  
 schedule2: function (callback, interval) {  
 var then = Date.now();  
 interval = interval \* 1000;  
 this.schedule(function () {  
 var now = Date.now();  
 var delta = now - then;  
 if (delta > interval) {  
 then = now - (delta % interval);  
 callback.call(this);  
 }  
  
 }.bind(this), 0, cc.REPEAT\_FOREVER, 0);  
 }  
 *// update: function (dt) {  
 // for (var i = 0; i < 10000000; i++) {  
 // b = 1 / 0.22222;  
 // }  
 // }*});

遮罩节点的使用

指定范围的内容显示，超出范围的内容隐藏。

DrawNode灵活的节点，可以自由绘制图案。

var clipnode = new cc.ClippingNode();  
this.addChild(clipnode);  
var npanel = new cc.Layer();  
npanel.x = 100;  
npanel.y = 100;  
clipnode.addChild(npanel);  
  
var nsprite = new cc.Sprite(res.HelloWorld\_png);  
nsprite.attr({  
 x: size.width / 2+150,  
 y: size.height / 2+100  
});  
npanel.addChild(nsprite);  
  
var stencil = new cc.DrawNode();  
stencil.drawRect(cc.p(480, 320), cc.p(700, 600), cc.color(255, 255, 255), 5, cc.color(255, 0, 0));  
clipnode.stencil = stencil;

分批处理--SpriteBatchNode

**需要手机的浏览器支持webGL，否则无用武之地。**

new cc.SpriteBatchNode(res.png);可以传spriteSheet大图片路径。

把SpriteSheet加载到游戏中，使用：

cc.spriteFrameCache.addspriteFrames(res.plist);

new cc.Sprite(“#spriteFrameName”);

*//---------spriteBatchNode---*cc.spriteFrameCache.addSpriteFrame("res/candy.plist");  
var batchnode = new cc.SpriteBatchNode("res/candy.png");  
this.addChild(batchnode);  
var ball = new cc.Sprite("#spriteFrameName");  
batchnode.addChild(ball);

缓存池pool的使用

建立一个可以回收的类，该类中必须有reuse和unuse方法的实现。

位图缓存--BakeLayer

bake方法只在Canvas2d环境才有效，例如Android或ios7以下的手机浏览器，而在手机原生版本（JSB）和WebGL环境都是无效的。

视察滚动背景（ParallaxNode）

背景运动遵循现实的移动情况。

*//-------------视察滚动背景-----*var parallaxnode = new cc.ParallaxNode();  
this.addChild(parallaxnode);  
*//parallaxnode.addChild(bgSprite,z-index,ratio.offset);*parallaxnode.addChild(bgSprite1,1,cc.p(0.1,0),cc.p(200,200));  
parallaxnode.addChild(bgSprite2,2,cc.p(0.3,0),cc.p(200,200));  
parallaxnode.addChild(bgSprite3,3,cc.p(0.5,0),cc.p(200,200));  
var act=cc.moveBy(1,200,0);  
parallaxnode.runAction(act);

方法二：使用scheduleUpdate做每帧变化。移动层。

瓦片地图--cc.TMXTiledMap

var map=new cc.TMXTiledMap(“map.tmx”);

粒子系统 编辑器ParticleEditor

停止播放：

paiticalSystem.stopSystem();

this.removeChild(paiticalSystem);

stopSystem方法主要用于无限循环播放的粒子系统，而对于播放有次数限制的粒子系统，我们可以让它自生自灭，具体使用setAutoRemoveOnFinish方法。

var particleSystem=new cc.ParticleSystem(res.plist);

particleSystem.setAutoRemoveFinish(true);

1. **使用什么工具生成文件的MD5值？**

**2.HTTP 短连接，一次通信后关闭连接；Socket是长连接，类似于通电话。cocos2d js 都支持两种通信方式，在html5和jsb上。**

**3.发布游戏时还需要接入一些系统：用户系统，广告系统，支付系统，分享系统和统计系统。同时接入多个渠道，使用官网的AnySDK，提供接入和发布功能，提供JS版本的API，在原生版本可以无缝对接。**

**4.plist文件编辑工具，plistEditor**

SQLite实现更安全、更强大的数据存储