**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 帧动画播放

帧动画播放，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);

ccs.armatureDataManager.addArmatureFileInfo(res.wolf\_animation);

var wolfAnimation = new ccs.Armature("wolf");

wolfAnimation.setScale(scale);

wolfAnimation.setPosition(size.width / 2 + 150, size.height / 2);

wolfAnimation.animation.play("Animation1");

this.addChild(wolfAnimation);

**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();