Cocos2d-js3.0

项目创建：cocos new -l js ProjectName

项目发布：cocos compile -p web -m release//进入项目目录

文件说明：

resource.js----图片，声音资源文件加载

project.json------加载程序的js文件和模块开启。如激活Chipmunk，

"modules" : ["cocos2d","chipmunk"],

Main.js-----入口文件

//判断是手机还是浏览器来采取不同适应手段适应页面

**var** mode = ***cc***.sys.**isMobile** && ***window***.**navigator**.**userAgent**.indexOf(**"MicroMessenger"**) != -1 ? ***cc***.**ResolutionPolicy**.**FIXED\_HEIGHT** : ***cc***.sys.**isMobile** ? ***cc***.**ResolutionPolicy**.**FIXED\_WIDTH** : ***cc***.**ResolutionPolicy**.**SHOW\_ALL**;

**//手机采用：*cc***.**ResolutionPolicy**.FIXED\_WIDTH

***//浏览器采用：cc***.**ResolutionPolicy**.SHOW\_ALL

***cc***.view.setDesignResolutionSize(640, 831, mode);

***cc***.**LoaderScene**.preload(***g\_resources***, **function** () {  
 ***cc***.director.runScene(**new *MainMeunSence***());  
}, **this**);

***cc***.**LoaderScene**.preload：可以自定义加载页面

***g\_resources：资源文件数组***

***cc***.director.runScene(**new *MainMeunSence***());资源加载完成后启动Sence

**知识点**

.添加精灵

**this**.**\_sptLogo** = **new *cc***.**Sprite**(***res***.**mm\_logo\_png**);//加载资源图片

**//** **new *cc***.**Sprite**(***res***.**mm\_mune\_png**, ***cc***.rect(252, 0, 126, 33));可以用矩形裁剪，参数为(x,y,width,height)

//指定位置，也可以用**this**.**\_sptLogo.x，还可以**back.setPosition(***Const***.**centre**);  
**//锚点默认0.5,0.5（比例，即中心），**

**this**.**\_sptLogo**.attr({  
 **x**: ***Const***.**w**/2,  
 **y**: ***Const***.**h**/2+60,

**anchorX**:1  
});

//可以设置属性

**this**.**player**.setScale(0.5);//—缩小0.5

**this**.**player**.setRotation(90);//旋转90度

*//在0之后加载，在0层的上面----默认值0* **--this**.addChild(**this**.**\_sptLogo**);**this**.addChild(**this**.**\_sptLogo**,1);

**.精灵表添加精灵（相同图片的精灵用精灵表高效）**

*var tex = cc.textureCache.addImage(res.runner0);//把图片添加到缓存  
var frame1=new cc.SpriteFrame(tex,rect);//一定要rect。。不然会报错*

**//简单几张图**

*var rect = cc.rect(0, 0, 62, 56);  
var tex = cc.textureCache.addImage(res.runner0);//把图片添加到缓存  
var tex1 = cc.textureCache.addImage(res.runner1);  
var tex2 = cc.textureCache.addImage(res.runner2);  
var moveFrames=[];//移动精灵帧  
var frame1=new cc.SpriteFrame(tex,rect);//一定要rect。。不然会报错  
var frame2=new cc.SpriteFrame(tex1,rect);  
var frame3=new cc.SpriteFrame(tex2,rect);  
moveFrames.push(frame1);  
moveFrames.push(frame2);  
moveFrames.push(frame3);  
this.player = new cc.Sprite(res.runner0);  
this.player.attr({  
 anchorX : 0.5,  
 anchorY : 0.5,  
 x : Const.w/2,  
 y : Const.h/2  
});  
this.addChild(this.player,5);  
var animation = new cc.Animation(moveFrames, 0.2);  
var animate = cc.animate(animation);  
var action = animate.repeatForever();//永远执行  
this.player.runAction(action);*

***//精灵表绘制方块---绘制同一图片且多数量时效率高*this**.**batch** = **new *cc***.**SpriteBatchNode**(***res***.**Block\_png**);//定义精灵表  
**this**.addChild(**this**.**batch**);  
**for**(**var** i=0;i<***Storeblock***.**length**;i++){  
 **if**( ***Storeblock***[i].**s**==0)  
 ***Storeblock***[i].**block**=**new *cc***.**Sprite**(**this**.**batch**.**texture**,***BLOCK2\_RECT***);  
 **else  
 *Storeblock***[i].**block**=**new *cc***.**Sprite**(**this**.**batch**.**texture**,***BLOCK1\_RECT***);  
 ***Storeblock***[i].**block**.attr({  
 **x**: ***Storeblock***[i].**x**,  
 **y**: ***Storeblock***[i].**y** });  
 **this**.**batch**.**addChild**(***Storeblock***[i].**block**);*//添加到精灵表中*}

**// plist图片和精灵表一起使用**

***cc***.spriteFrameCache.addSpriteFrames(***res***.**runner\_plist**);//添加到缓存  
**this**.**spriteSheet** = **new *cc***.**SpriteBatchNode**(***res***.**runner\_png**);*//精灵表***this**.addChild(**this**.**spriteSheet**);  
**var** animFrames = [];  
**for** (**var** i = 0; i < 8; i++) {  
 **var** str = **"runner"** + i + **".png"**;  
 **var** frame = ***cc***.spriteFrameCache.getSpriteFrame(str);*//获取精灵帧* animFrames.push(frame);  
}  
**var** animation = **new *cc***.**Animation**(animFrames, 0.1);  
**this**.**runningAction** = **new *cc***.**RepeatForever**(**new *cc***.**Animate**(animation));  
**this**.**sprite** = **new *cc***.**Sprite**(**"#runner0.png"**);*//从缓存获取图片***this**.**sprite**.attr({**x**:80, **y**:85});  
**this**.**sprite**.runAction(**this**.**runningAction**);  
**this**.**spriteSheet**.**addChild**(**this**.**sprite**);

.添加文字

//内容+字体+大小

**var** label = **new *cc***.**LabelTTF**(**"继续刷屏！\n"**+**"6步推倒我的小羊驼\n打败99%朋友圈的人！ "**, **"宋体"**, 20);  
label.**x** = w/2;  
label.**y** = h/4+5;  
label.**textAlign** = ***cc***.**LabelTTF**.**TEXT\_ALIGNMENT\_CENTER**;  
label.**width** = w;//宽度  
label.setString(**"hahahahah"**);//更改文字内容

label.**color** = ***cc***.color(0, 0, 0); *//面板上面加文字***this**.**winPanel**.**addChild**(label,1);

.音乐播放

***cc***.audioEngine.playEffect(***res***.**gp\_shipDestroyEffect\_mp3**);//播放音效

***cc***.audioEngine.playMusic(***res***.**mm\_bgMusic\_mp3**, **true**);//播放背景音乐，循环播放

**var** audioEngine = ***cc***.audioEngine;  
**if**(***Const***.**SOUND\_ON**){  
 **if** (***cc***.audioEngine.isMusicPlaying()){  
 **return**;  
 }  
 ***cc***.audioEngine.playMusic(***res***.**mm\_bgMusic\_mp3**, **true**);//循环播放,false只播放一次  
}  
**else**{  
 audioEngine.stopMusic();//背景音乐  
 audioEngine.stopAllEffects();//音效  
}

.场景切换特效

//特效- **new *cc***.**TransitionFade***更多效果，参考引擎包samples/js-tests下的Transitions Test*

***cc***.director.runScene(**new *cc***.**TransitionFade**(1.2, **new *MainMeunSence***()));//1.2s完成切换

.文字按钮

**var** label = **new *cc***.**LabelTTF**(**"Go back"**, **"Arial"**, 24);  
*//字体按钮***var** back = **new *cc***.**MenuItemLabel**(label, **this**.onBackCallback);//响应函数  
**var** menu = **new *cc***.**Menu**(back);//创建菜单  
menu.**x** = ***Const***.**w**/2;  
menu.**y** = 40;  
**this**.addChild(menu);

.菜单按钮

**var** aboutNormal = **new *cc***.**Sprite**(***res***.**mm\_mune\_png**, ***cc***.rect(252, 0, 126, 33));

*三个菜单项，并且指定菜单项点击所会执行的函数* **var** newGame = **new *cc***.**MenuItemSprite**(  
 newGameNormal,//精灵图片—正常状态  
 newGameSelected,//点击状态  
 newGameDisabled,//消失状态  
 **function**(){*//点击触发事件*  
 **this**.onButtonEffect();  
 **this**.flareEffect(flare, **this**, **this**.onNewGame);  
 }.bind(**this**)  
 );  
 **var** about = **new *cc***.**MenuItemSprite**(  
 aboutNormal,  
 aboutSelected,  
 aboutDisabled,  
 **this**.onAbout,*//如果this.onAbout()会自动跳转到这个页面* **this** );  
*// 菜单。 对应三者关系：菜单里面有菜单项，菜单项中绑定要执行的方法，并且需要图片去显示。图片就是精灵* **var** menu = **new *cc***.**Menu**(newGame, gameSettings, about);//加载3个按钮...参数可以多个  
 menu.alignItemsVerticallyWithPadding(10);//垂直摆放  
 menu.**x** = ***Const***.**w**/2;  
 menu.**y** =***Const***.**h**/2 - 80;  
 **this**.addChild(menu, 1, 2);//2—tag，标志  
 },

**.点击变换按钮或文字**

**this**.**music**=**new *cc***.**MenuItemToggle**(  
 *// new cc.Sprite("#music.png")错误  
 //变化的文字用--- new cc.MenuItemFont("On"),* **new *cc***.**MenuItemImage**(**"#music.png"**),  
 **new *cc***.**MenuItemImage**(**"#music\_sel.png"**)  
);  
*// 设置函数回调***this**.**music**.setCallback(**this**.MusicCallback );  
*//设置显示图片  
//（条件表达式）？（条件为真时的表达式）：（条件为假时的表达式）***this**.**music**.setSelectedIndex(***Const***.**SOUND\_ON** ? 0 : 1);*//文字显示选择--0.1下标，****Const***.**SOUND\_ON为声音状态****var** menu = **new *cc***.**Menu**(**this**.**music**);  
menu.**x**=***Const***.**w**-**this**.**music**.**width**;  
menu.**y**=**this**.**logo**.**y**;  
**this**.addChild(menu);

**回调函数**

MusicCallback:**function**()  
{  
 ***Const***.**SOUND\_ON** = !***Const***.**SOUND\_ON**;  
 ***cc***.log(***Const***.**SOUND\_ON**);  
}

**.慢慢显示ProgressTimer**

**this**.**line** = **new *cc***.**ProgressTimer**(**new *cc***.**Sprite**(***res***.**line\_png**));//要显示的图片  
**this**.**line**.**type** = ***cc***.**ProgressTimer**.**TYPE\_BAR**;//线性显示

*you want a left to right then set the midpoint all the way to cc.p(0,y)*<br/>  
*you want a right to left then set the midpoint all the way to cc.p(1,y)*<br/>  
*you want a bottom to top then set the midpoint all the way to cc.p(x,0)*<br/>  
*you want a top to bottom then set the midpoint all the way to cc.p(x,1)*</p>  
**this**.**line**.**midPoint** = ***cc***.p(0, 1);// *top to bottom*  
**this**.**line**.**barChangeRate** = ***cc***.p(0, 5);//速率,x速率0，y为5  
**this**.**line**.**percentage** = 0;//当前显示0%  
**this**.**line**.setPosition(***cc***.p(***cc***.winSize.**width** / 2 - 30, ***cc***.winSize.**height** - **this**.**line**.**height** / 2));  
**this**.addChild(**this**.**line**);  
//  
**this**.**line**.runAction(***cc***.progressTo(3,20));//3秒增加20

**.定时器**

1.

**this**.scheduleUpdate();*//时时处理--会调用本类的update方法---每一帧都调用*

*//取消定时器*

unschedulerUpdate()

2.

**this**.schedule(**this**.scoreCounter, 2);*//响应函数。。每隔2秒*

scoreCounter*:function(dt)//dt就是时间2s  
//取消定时器  
this.unschedule(this.* scoreCounter*,2);*

3.

单次调度器:scheduleOnce(SEL\_SCHEDULE selector, float delay)//只执行一次

取消：unschedule(SEL\_SCHEDULE selector, float delay)

Js自带定时函数

setInterval(function(){

this.addChild(sprite);

}.bind(this),200);//每隔200ms执行一次，要bind(this)，不然函数里的this不是所要的this

*.***活动—Action**

**每个活动都有自己的活动管理**

\*\*\*To: 意味着运动到指定的位置。

\*\*\*By:意味着运动到按照指定癿 x、y 增量的位置。(x、y 可以是负值)

参考：<http://cn.cocos2d-x.org/article/index?type=cocos2d-x&url=/doc/cocos-docs-master/manual/framework/native/v3/action/zh.md>

活动执行**this**.\_ship.runAction(action);// this.ship执行对象，action-执行的活动

//移动

**var** easeMove = ***cc***.moveBy(3,***cc***.p(200,-200));*//，每秒钟往y移动-200，x移动200-----速率*

**var** easeMove = ***cc***.moveTo(3,***cc***.p(200,-200));*//3秒钟移动点(200,-200)*

*var faneaseMove=easeMove.reverse();//取反动作*

*//按顺序执行一组动作  
var seqAction = cc.sequence(easeMove,faneaseMove);*

*var reAction=seqAction.repeatForever();*//活动用于执行

//缩放

**var** biggerEase = ***cc***.scaleBy(0.7, 1.2, 1.2)//0.7s x和y放大1.2

//旋转

**var** rotateEase = ***cc***.rotateTo(2.5, 90)

//延时

***cc***.delayTime(0.2),//停止0.2s

//回调  
***cc***.callFunc(**this**.onGameOver, **this**)//活动执行完回调函数---响应函数**this**.onGameOver

//活动组合---顺序

**var** action = ***cc***.sequence(***cc***.delayTime(0.2),***cc***.callFunc(**this**.onGameOver, **this**));  
**this**.\_ship.runAction(action);//运行活动---this.ship执行对象

*//同时执行一组动作***var** action = ***cc***.spawn(seqAction, easeMove, rotateEase, bigger);

**.触屏事件**

//点击监听

***cc***.eventManager.addListener({  
 **event**: ***cc***.**EventListener**.**TOUCH\_ALL\_AT\_ONCE**,  
 **onTouchesEnded**: **this**.hit *//触屏函数*}, **this**);

*/\*\*  
 \* 触屏函数里面不能用this。。用 var target = event.getCurrentTarget();//获取当前对象代替  
 \** ***@param*** *touches  
 \** ***@param*** *event  
 \*/*hit:**function** (touches, event) {  
 **var** target = event.getCurrentTarget();*//获取当前对象* **var** touch = touches[0];  
 *//获取点击的坐标* **var** pos = touch.getLocation();//(pos.x,pos.y)

}

*// 触屏事件* ***cc***.eventManager.addListener({  
 **event**: ***cc***.**EventListener**.**TOUCH\_ONE\_BY\_ONE**,  
 **swallowTouches**: **true**,// *事件穿透，*  
 **onTouchBegan**: **this**.**onTouchBegan**,  
 **onTouchMoved**: **this**.**onTouchMoved**,  
 **onTouchEnded**: **this**.**onTouchEnded**,  
 **onTouchCancelled** : **this**.**onTouchCancelled** }, **this**);

**onTouchMoved**: **function** (touch, event) {  
 **var** target = event.getCurrentTarget();

var position= touch.getLocation()//位置  
 **return true**;  
 },

**常用函数**

cc.log();//输出函数

**克隆动作和监听**

action.clone()

**是否点击中精灵---击中返回true**

cc. rectContainsPoint(this.sprite.boundingBox(), Point);// boundingBox()精灵范围(所在矩形)，Point为需要判断的点

**判断两个精灵是否相撞—相撞返回true**

this.sprite.boundingBox().intersectsRect(this.sprite1.boundingBox())//this.sprite1.boundingBox()获取一个精灵所在矩形

***//获取屏幕高宽--cc.winSize*var** size = ***cc***.winSize;  
***Const***.**h**=size.**height**;  
***Const***.**w**=size.**width**;

**//获取精灵宽高**

***buttonW***=**this**.**notyfy**.getContentSize().**width**;  
***buttonH***=**this**.**notyfy**.getContentSize().**height**;

**停止精灵所有动作**

this.sprite.stopAllAction();

**//layer,精灵移除**

***target.initLayer.removeFromParent();****//在触控函数中，target代替this*

**target.removeAllChildren()**;*//直接移除所有节点*

***this.initLayer.player. removeFromParent()****可以多层调用*

**//获取缓存图片**

**var** frame0 = ***cc***.spriteFrameCache.getSpriteFrame(**"ship01.png"**);//精灵帧直接名字

**this**.**sprite** = **new *cc***.**Sprite**(**"#runner0.png"**);*//从缓存获取图片，加#*

*var tex = cc.textureCache.addImage(res.runner0);//把图片添加到缓存*

*var frame1=new cc.SpriteFrame(tex,rect);//一定要rect。。不然会报错。图片生成精灵帧移动要rect*