1.场景加载

cc.director.loadScene('场景名称');//场景跳转

cc.director.preloadScene('场景名称');//预加载场景

cc.director.getScene();//获取当前场景

2.获取节点

var node = cc.find("Canvas/bg");//通过访问路径来获取节点

var a = this.node.getChildByName('name');//通过名字获取子节点

node.getComponent(cc.Label).string = 'abc';//获取节点上的组件值

var a = cc.find("Canvas/bg").getComponent(cc.Sprite);//通过访问路径来获取节点，及获取该节点的指定组件

this.node .getChildByName('节点名称').getComponent(cc.Label)//通过节点名获取子节点，获取该节点指定组件

var a = this.node;//获取当前脚本所在的节点

var a = this.node.parent;//获取父节点

var a = this.node.getChildByTag(1001);//通过标签获取子节点

var a = cc.find("bg/score",this.node);//通过指定节点下的路径获取节点

var a = this.node.children;//获取所有子节点

var a = this.node.childrenCount;//获取子节点数量

var a = cc.director.getScene();//获取场景主节点

var a = cc.instantiate(node);//克隆节点

this.node.parent = cc.find('Canvas');//绑定父节点

this.node.addChild(nodeName,zIndex,tag);//添加子节点,可设置层级和标签

this.node.removeChild(nodeName);//通过名字移除子节点

this.node.removeChildByTag (nodeTag);//通过标签移除子节点

this.node.destroy();//销毁节点

this.node.isValid;//判定节点是否可用

this.node.removeChild(newNode);//移除节点中指定的子节点

this.node.removeChildByTag(1001);//通过标签移除节点中指定的子节点

this.node.removeAllChildren();//移除所有子节点

this.node.destroyAllChildren();//销毁所有子节点

this.node.cleanup();//停止所有正在播放的动作和计时器

var sprites = this.node.getComponentsInChildren(cc.Label);//递归查找自身及所有子节点中指定类型的组件

3.获取节点位置，设置节点

var a = node.getPositionX();或 getPositionY() //获取节点的X轴或Y轴坐标

var a = node.getScaleX(); 或getScaleY() //获取节点的X轴或Y轴缩放比例

node.x = 100;//设置节点x轴坐标

node.y = 100;//设置节点y轴坐标

node.setPosition(x,y); //设置节点坐标

node.rotation = 90; //设置节点旋转角度

node.scaleX = 2; //设置节点x轴缩放倍数

node.scaleY = 2; //设置节点y轴缩放倍数

node.setScale(2); //设置节点整体缩放倍数

node.width = 100; //设置节点宽度大小

node.height = 100; //设置节点高度大小

node.setContentSize(100, 100); //设置节点宽高尺寸大小

node.anchorX = 1; //设置节点x轴锚点坐标

node.anchorY = 0; //设置节点y轴锚点坐标

node.setAnchorPoint(1, 0); //设置节点锚点坐标

node.opacity = 128; //设置节点透明度大小（0-255）

node.setOpacity(20); //设置节点透明度（0~255）

node.color = new cc.color(100,100,100,255); //设置节点颜色（R,G,B,透明度）

if (cc.isValid(this.label.node) ) //判定节点是否存在

node.destroy(); //销毁节点

this.cannons = [];

this.cannons = node.getChildren(); //获取所有子节点

this.cannons = node.getChildrenCount(); //获取子节点数量

node.active = false; //关闭节点(隐藏节点)

cc.game.addPersistRootNode(myNode); //常驻节点（全局变量）

cc.game.removePersistRootNode(myNode); //取消常驻节点

4.动作操作

cc.show()//立即显示

cc.hide ()//立即隐藏

cc.toggleVisibility()//显隐切换

cc.fadeIn(1)//渐显效果

cc.fadeOut(1)//渐隐效果

cc.delayTime(1)//等待1秒

node.runAction(cc.moveTo(1,0,0)); //移动到当前节点（时间（s），X轴坐标，Y 轴坐标）

node.runAction(cc.scaleTo(1,0.7,0.8));//缩放到当前倍数节点（时间（s），X轴倍数，Y 轴倍数）

node.runAction(cc.rotateTo(1,160,160));//旋转到指定角度（时间（s），X轴角度，Y 轴角度）

node.runAction(cc.skewTo(1,5,-5));//变化节点倾斜度（时间（s），X轴倾斜度，Y 轴倾斜度）

node.runAction(cc.fadeTo(2,0));//变化当前节点的透明度（时间（s），透明度）

node.runAction(cc.tintTo(2,255,255,0));//变化当前节点颜色（时间，R,G,B）

node.stopAllActions();//停止所有动作

//自定义动作

var action = cc.moveTo(2, 100, 100);// 创建一个移动动作

node.runAction(action);// 执行动作

node.stopAction(action);// 停止一个动作

cc.sequence(action1,action2); //按顺序连续执行，先action1，后action2

cc.spawn(action1，action2); //同时执行，action1和action2一起执行

cc.repeatForever(cc.sequence(action1,action2)); //一直重复括号里的动作

5.计时器

//只用1次的计时器,2秒后执行

this.scheduleOnce(function(){

//一条或多条执行语句

},2); //(function(){},时间（s）)

//每隔5秒执行1次

this.schedule(function(){

//一条或多条执行语句

},5);

//计算多次的计时器（1秒后，以0.1秒的执行间隔，执行10次）

this.schedule(function(){

//一条或多条执行语句

},0.1,10,1); //(function(){},间隔时间，次数，多久后开始)

this.unscheduleAllCallbacks(this);//停止某组件的所有计时器

//自定义定时器执行内容（相比常规使用的定时器优势是：方便随时开启或关闭）

var cb= function(){

//do something

};

this.schedule(cb,1);//启动定时器

this.unschedule(cb);//取消定时器

6.事件监听

(开始：'touchstart'，移动：'touchmove'，结束：'touchend'，取消：'touchcancel')

node.on('touchstart',function(event){

this.doSomething();

},this);

var a = event.getID();//获取触点的ID

var a = event.getLocationX();//获取触摸点的坐标X

var b = event.getLocationY();//获取触摸点的坐标Y

cc.eventManager.addListener({

event: cc.EventListener.KEYBOARD/TOUCH\_ONE\_BY\_ONE,myfunction},self.node);

7.定义全局变量

window.DEFAULT\_IP = "192.168.1.1";//任意脚本里可定义全局变量

//任意脚本里可定义全局变量

window.G = {

a: null,

b: null,

};

//任意脚本里可访问全局变量（切记定义全局变量的那个脚本已执行过）

G.a = 0;

G.b = 0;

var something = require（‘something’）；

cc.game.addPersistRootNode(myNode);//常驻节点,必须位于层级的根节点（也可算全局节点吧）

module.exports = {

config: 123

}

8.分辨率

//获得设备分辨率

var b = cc.director.getWinSizeInPixels()

var bx = b.width

var by = b.height

cc.view.getCanvasSize().width;//获得设备分辨率的宽度

cc.view.getCanvasSize().height;//获得设备分辨率的高度

cc.director.setDisplayStats(true);//显示帧数信息

9.音频控制

cc.audioEngine.playMusic(this.BGAudio,true);//播放音乐（true代表循环）

cc.audioEngine.stopMusic()//停止播放背景音乐

cc.audioEngine.playEffect(this.ClickAudio,false);//播放音效（false代表只播放一次）

cc.audioEngine.stopEffect(音效变量名);//停止指定音效（需要先把音效赋值给变量）

cc.audioEngine.AllEffects();//停止所有音效

cc.audioEngine.setMusicVolume(参数); //设置背景音乐的音量（该参数范围是0到1）

cc.audioEngine.setEffectsVolume(参数); //设置音效的音量（该参数范围是0到1）

10.存档操作

cc.sys.localStorage.setItem('存储标识名',变量名);//存储存档数据

var a = cc.sys.localStorage.getItem('存储标识名');//读取存档数据

cc.sys.localStorage.removeItem('存储标识名');//擦除存档数据

userData = {

name: 'Tracer',

level: 1,

gold: 100

};

cc.sys.localStorage.setItem('userData', JSON.stringify(userData));//存取复杂对象数据

var userData = JSON.parse(cc.sys.localStorage.getItem('userData'));//读取复杂对象数据

11.判断平台

cc.sys.isNative //是否是本地

cc.sys.isBrowser //是否是网页

cc.sys.isMobile //是否是移动系统

cc.sys.platform //正在运行的平台

cc.sys.language //当前运行系统的语言

cc.sys.os //当前正在运行的系统

cc.sys.OS\_IOS //是否是IOS系统

cc.sys.OS\_ANDROID //是否是android系统

cc.sys.OS\_WINDOWS //是否是windows系统

cc.sys.openURL('Http://www.baidu.com'); //打开网页

12.监听和发射事件

this.node.pauseSystemEvents(true);//暂停节点系统事件

this.node.resumeSystemEvents(true);//恢复节点系统事件

this.node.targetOff(this);//移除所有注册事件

触摸监听：开始'touchstart',移动'touchmove',结束'touchend',取消'touchcancel'

var pos = event.getLocation();//获取触摸点的坐标(包含X和Y)

var x = event.getLocationX();//获取触摸点的X坐标

var y = event.getLocationY();//获取触摸点的Y坐标

var a = event.getID();//获取触点的ID

鼠标监听：鼠标按下'mousedown',移入节点'mouseenter',节点中移动'mousemove',移出节点'mouseleave,'松开鼠标'mouseup'

var a = event.getScrollY();//获取滚轮滚动的 Y 轴距离，只有滚动时才有效

var a = event.getLocation();//获取鼠标位置对象，对象包含 x 和 y 属性

输入框监听：获得焦点'editing-did-began',文字变化'text-changed',失去焦点'editing-did-ended',按下回车'editing-return'

属性变化监听：位置'position-changed',宽高 'size-changed',旋转'rotation-changed',缩放'scale-changed'

ScrollView控件监听：滚动中'scrolling',停止滚动'scroll-ended'

用户自定义事件:

this.node.on('事件名',function,this);//注册监听

this.node.emit('事件名');//发送监听广播

this.node.off('事件名',function,this);//关闭监听

//注册带参数监听

this.node.on('事件名',function(event){

“具体方法函数内容”

},this);

//发送带参数的监听

this.node.emit('事件名',{id:1001});

cc.eventManager.addListener(listener, node);//添加事件

cc.eventManager.removeListener((listener);//移除事件

13.其他操作

cc.director.pause();//暂停

cc.director.resume();//继续

cc.director.end();//退出整个应用

cc.log(变量) 或 console.log(something);//输出想要的信息

let self = this;//锁定当前使用的this指向

node.getLocalZOrder();//层级获取

node.setLocalZOrder(1);//层级改变

cc.find('canvas/map' + num)//读取带变量的路径

cc.sys.openURL('Http://www.baidu.com');//打开网页