



Cocos2d-JS游戏开发

---工程结构及核心概念



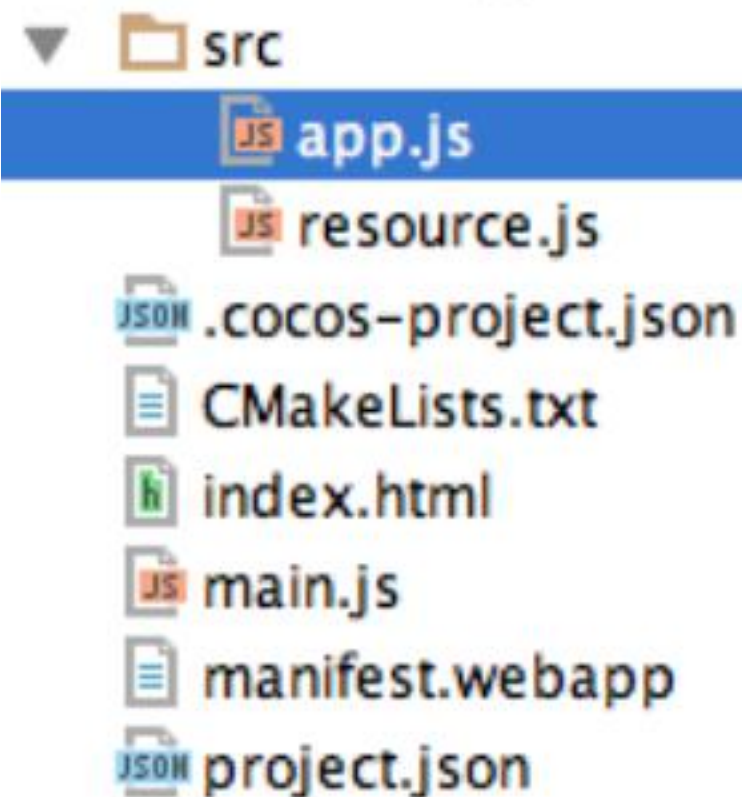
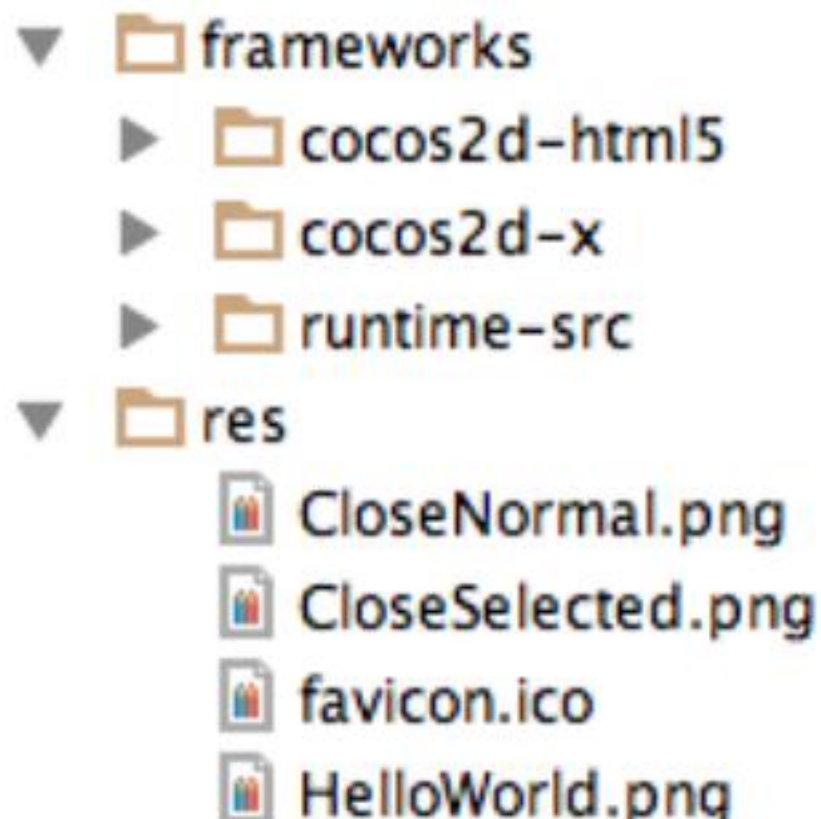
河北师范大学软件学院
Software College of Hebei Normal University

工程结构概述

- Frameworks (引擎框架)
cocos2d-html5、cocos2d-x、runtime-src
- res (资源目录)
- src (源代码)
- index.html、main.js (程序入口)
- project.json (配置文件)
debugMode、showFPS、frameRate
rendMode、engineDir、modules、jsList

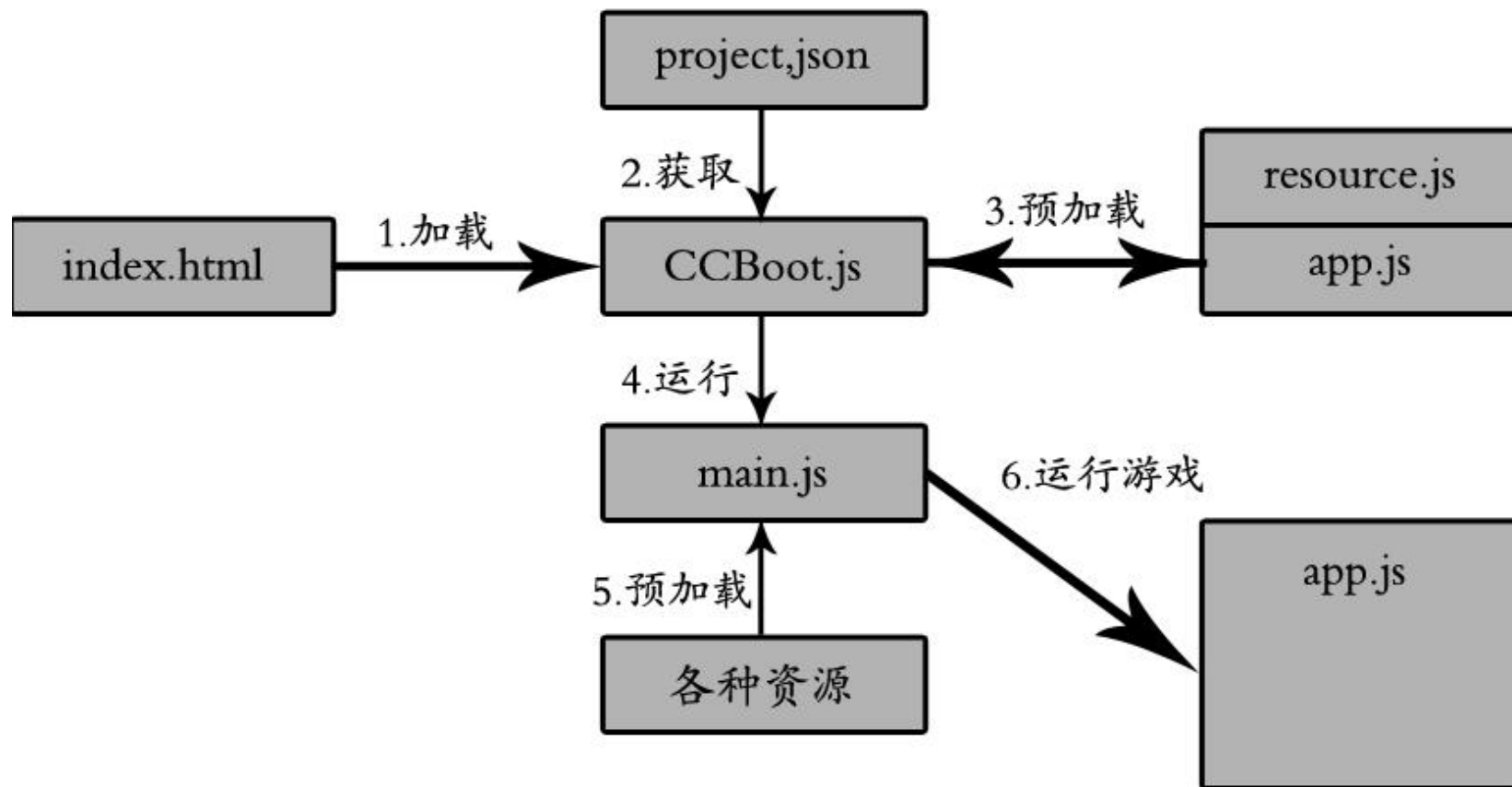
工程结构概述

- Cocos工程结构（frameworks、res、src等）



Cocos2d-JS游戏启动流程

- 工程启动及加载流程图



工程文件综述

- index.html (调取CCBoot.js获取project.json配置信息)

```
<html>
<head...>
<body>
  <!--<script src="res/loading.js"></script>-->
  <canvas id="gameCanvas" width="480" height="720"></canvas>
  <script...>
    <script src="frameworks/cocos2d-html5/CCBoot.js"></script>
    <script cocos src="main.js"></script>
  </body>
</html>
```



工程文件综述

- project.json工程配置文件

- 调试模式、是否显示帧率、默认帧率设置、渲染模式
- 引擎模块、脚本索引列表

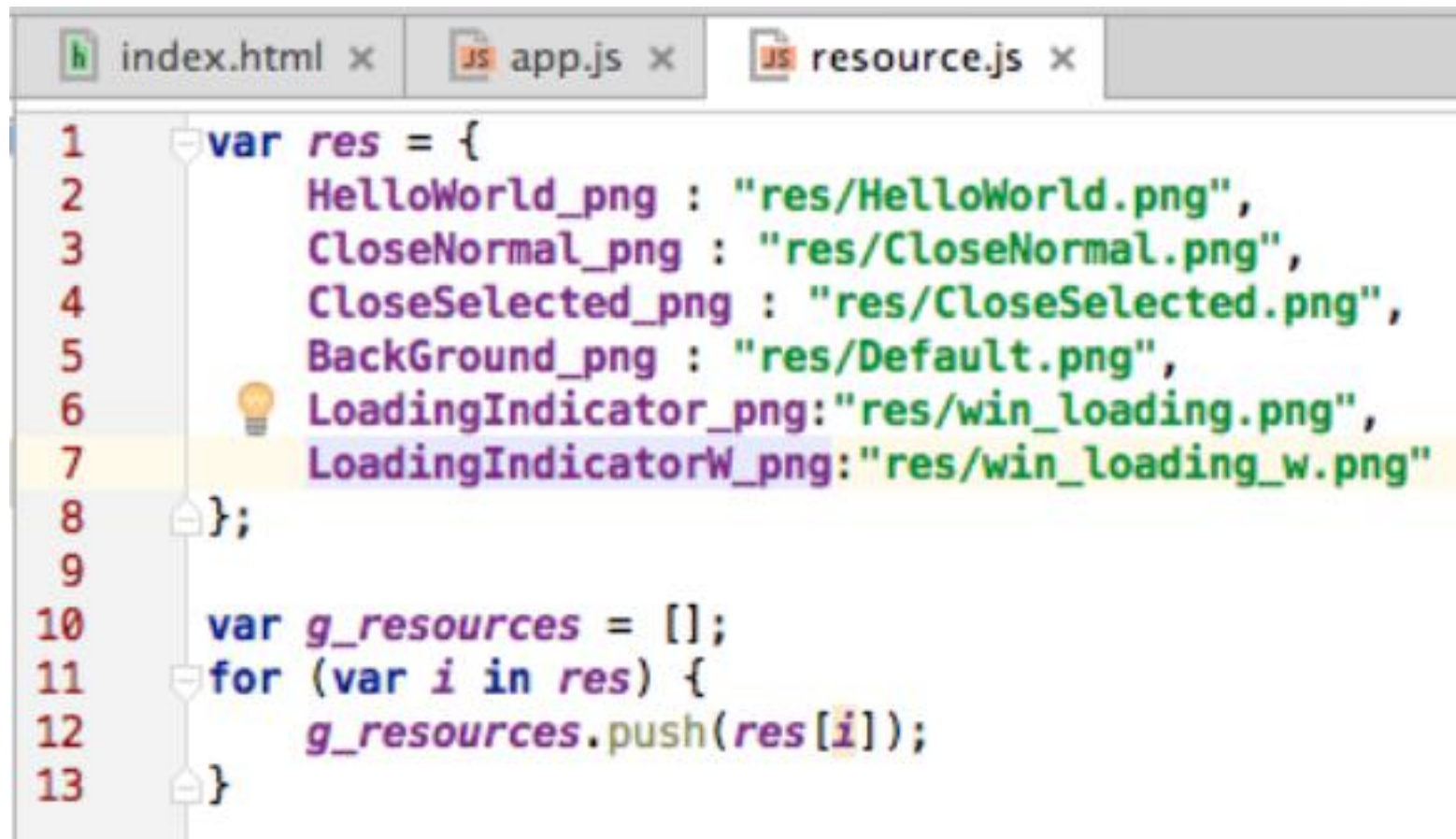
```
"project_type": "javascript",  
"debugMode" : 1,  
"showFPS" : true,  
"frameRate" : 60,  
"noCache" : false,  
"id" : "gameCanvas",  
"renderMode" : 0,
```

```
"engineDir": "frameworks/cocos2d-html5",  
"modules" : ["cocos2d", "ccui"],  
"jsList" : [  
    "src/resource.js",  
    "src/app.js"  
]
```



工程文件综述

- resource.js资源索引



```
1  var res = {  
2      HelloWorld_png : "res/HelloWorld.png",  
3      CloseNormal_png : "res/CloseNormal.png",  
4      CloseSelected_png : "res/CloseSelected.png",  
5      BackGround_png : "res/Default.png",  
6      LoadingIndicator_png:"res/win_loading.png",  
7      LoadingIndicatorW_png:"res/win_loading_w.png"  
8  };  
9  
10 var g_resources = [];  
11 for (var i in res) {  
12     g_resources.push(res[i]);  
13 }
```



工程文件综述

- 程序入口文件main.js

```
cc.game.onStart = function(){  
    if(!cc.sys.isNative && document.getElementById("cocosLoading")) //If reference  
        document.body.removeChild(document.getElementById("cocosLoading"));  
  
    // Pass true to enable retina display, disabled by default to improve performance  
    cc.view.enableRetina(false);  
    // Adjust viewport meta  
    cc.view.adjustViewPort(true);  
    // Setup the resolution policy and design resolution size  
    cc.view.setDesignResolutionSize(750, 1334, cc.ResolutionPolicy.FIXED_WIDTH);  
    // The game will be resized when browser size change  
    cc.view.resizeWithBrowserSize(true);  
    //load resources  
    cc.LoaderScene.preload(g_resources, function () {  
        cc.director.runScene(new Scene1());  
    }, this);  
};  
cc.game.run();
```



工程文件综述

- 起始场景案例app.js



```
1
2  var HelloWorldLayer = cc.Layer.extend({
3      sprite:null,
4      ctor:function () {...}
70  });
71
72  var HelloWorldScene = cc.Scene.extend({
73      onEnter:function () {
74          this._super();
75          var layer = new HelloWorldLayer();
76          this.addChild(layer);
77      }
78  });
```



Cocos2d-JS中的面向对象机制

- 在开源社区中John Resig在他的博客中提供了一种简单JavaScript继承方法。Cocos2d-JS使用的就是这种方法
- John Resig的简单JavaScript继承方法灵感来源于原型继承机制，它具有与Java等面向对象一样的类概念，并且他设计了所有类的根类Class

<http://ejohn.org/blog/simple-javascript-inheritance/>



实验一

- 创建一个非原生游戏工程
 - `cocos new -l js --no-native Demo_1`
- 熟悉工程相关文件（src、res等）
- 更改屏幕设计尺寸及适配方式（改为竖屏）
- 替换背景图、设置背景颜色
- 设置断点单步调试，熟悉程序流程

The background of the slide is decorated with numerous overlapping circles in various shades of green and yellow, scattered across the top and right sides.

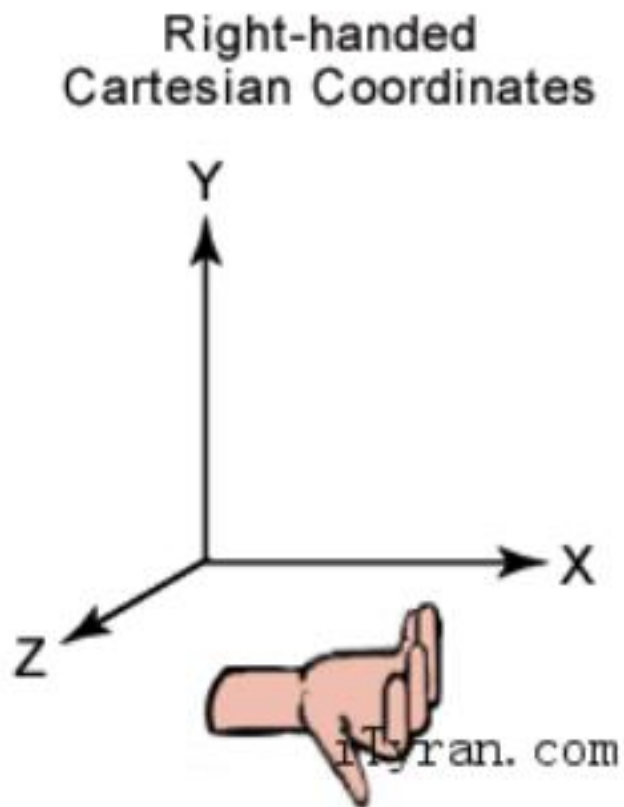
Have a break!

核心概念

- 坐标系、节点 (Node)
- 定时器 (Schedule)
- 标签及菜单 (Label、Menu、MenuItem)
- 场景与层 (Scene、Layer)
- 导演 (Director)
- 精灵 (Sprite)
- 动作与动画 (Action、Animation)

坐标系

- 笛卡尔（右手）坐标系，原点为左下角



Cocos2d-JS坐标系

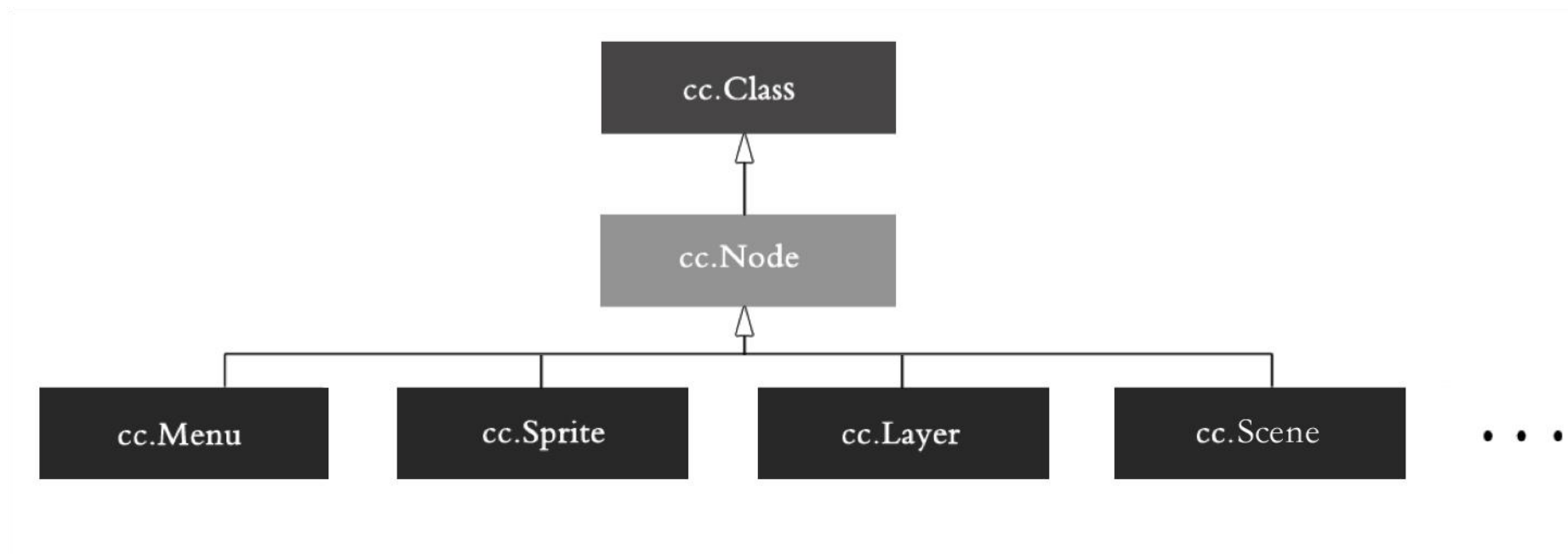
- Cocos2d采用的是笛卡尔坐标系



- UI坐标系与Cocos2d(OpenGL)坐标系

节点 (Node)

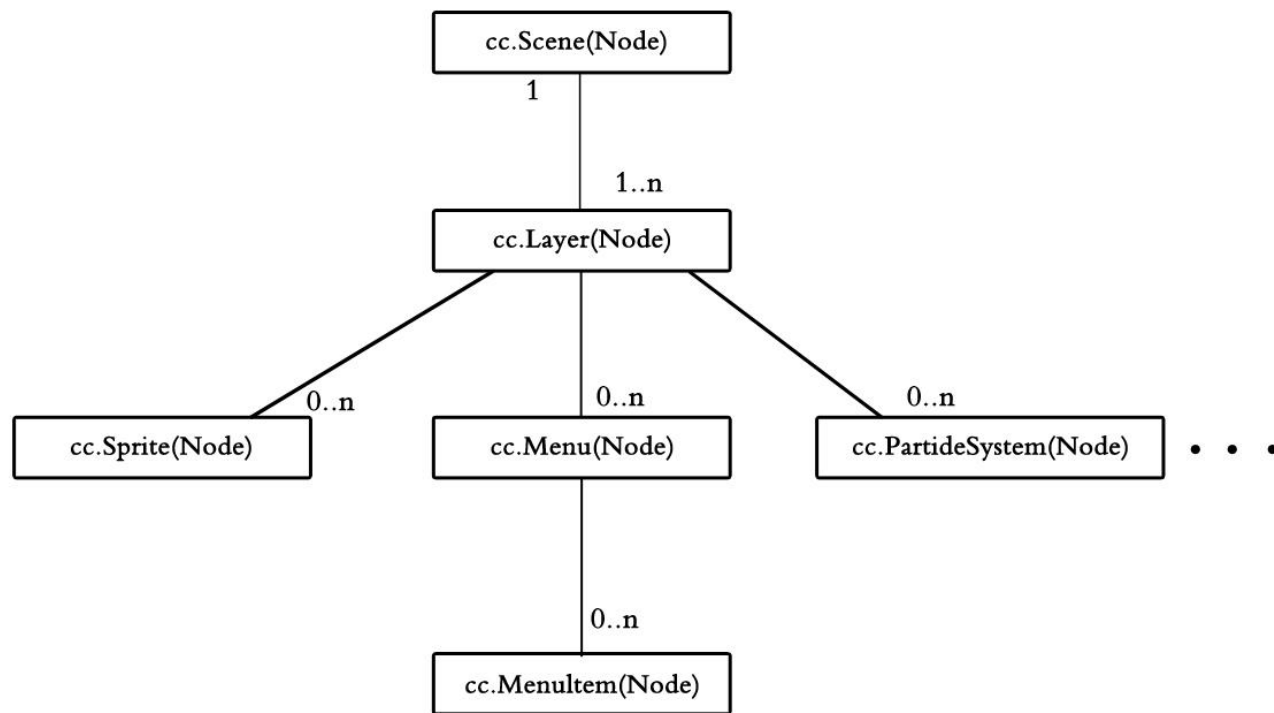
- cc.Node类是所有可视化组件类的根类



- 现实中的例子（教室、乐高、游戏等）
- 注：一般常用Node子类来实例化对应的可视化组件

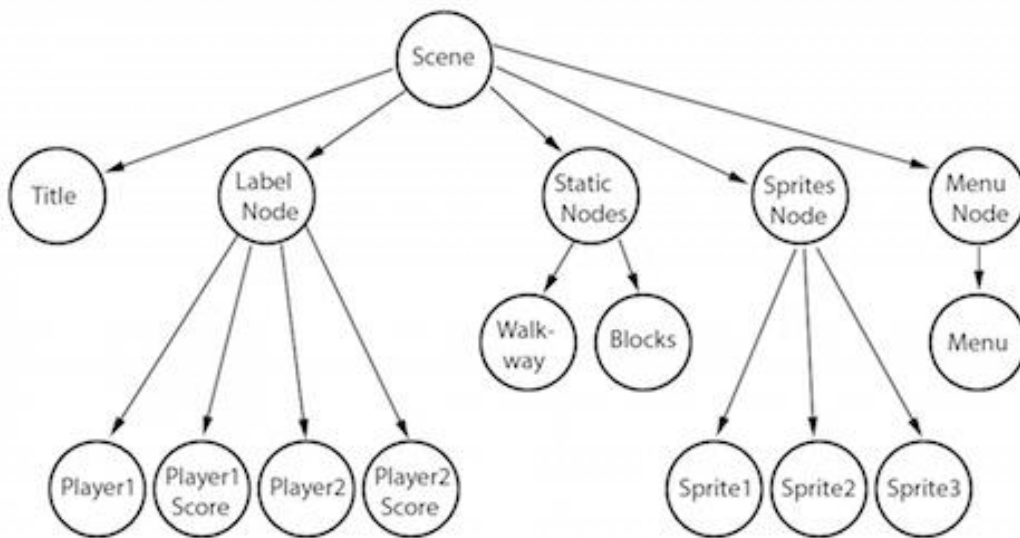
节点（树形结构管理）

- Cocos2d-JS采用层级（树形）结构来管理场景、层、精灵、菜单等节点（Node）



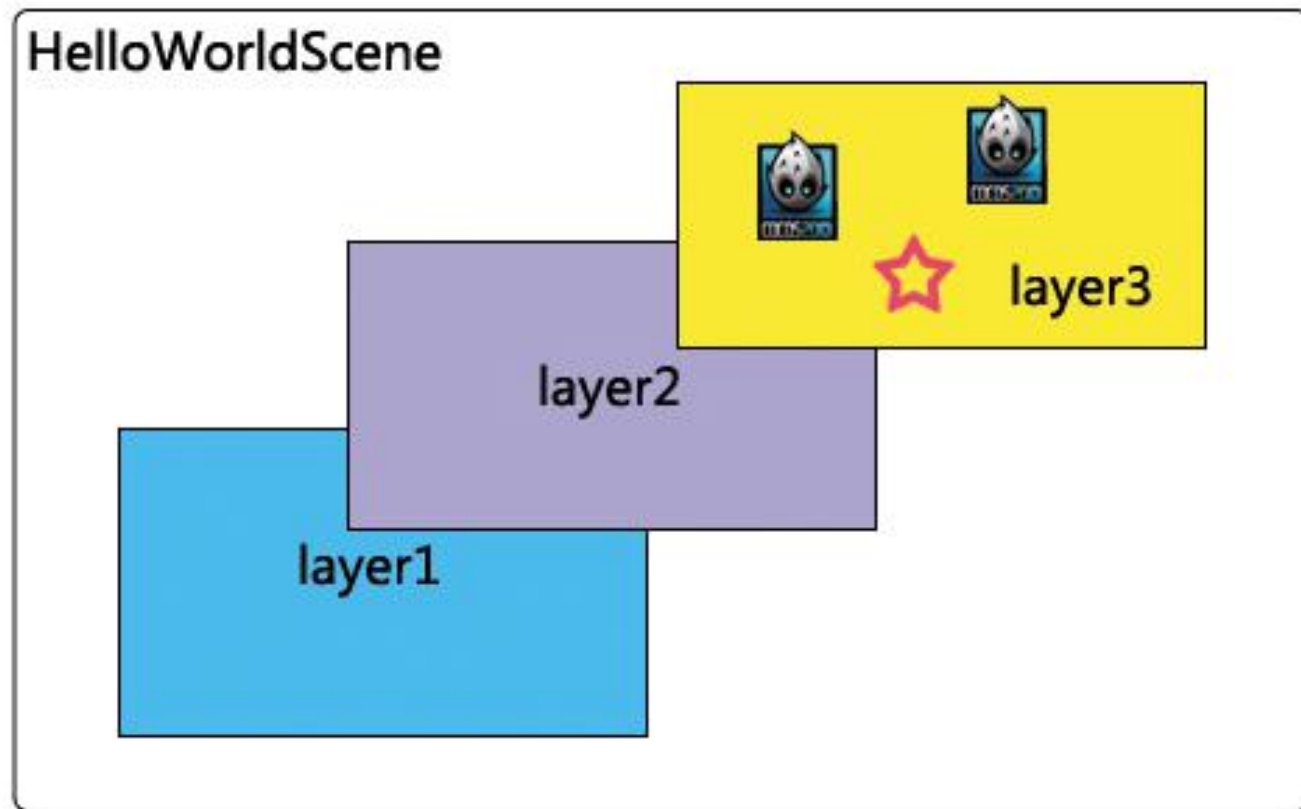
Cocos2d世界中的树状结构

- 节点组合实例



Cocos2d世界中的树状结构

- 节点组合实例（参见初始工程）



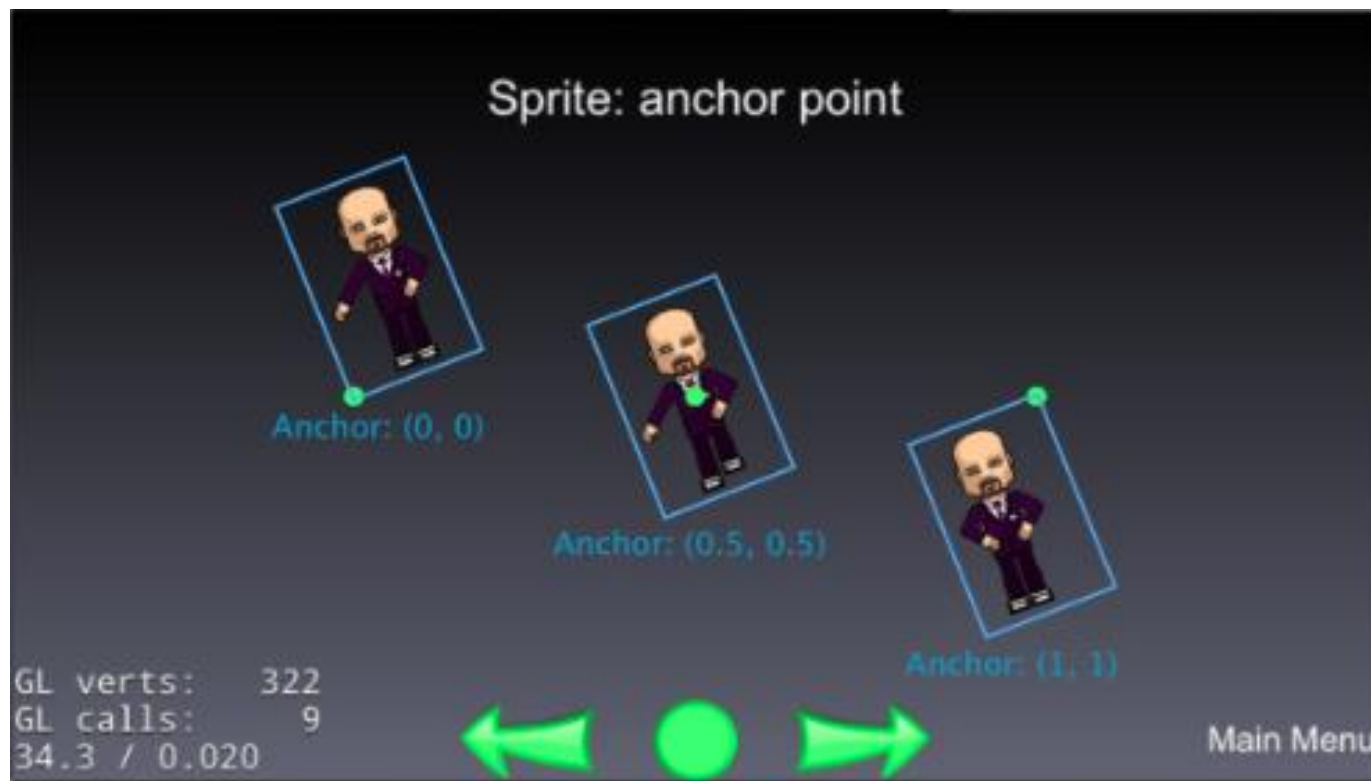
节点（属性、方法）

- 常用方法：addChild、getChildByTag、removeChild、setZOrder、setScale、setPosition、setVisible、**pause**、**onEnter**、**onExit**、**schedule**、**update** ...
- 常用属性：_visible、tag、_parent、_scheduler、_running、_localZOrder ...

锚点

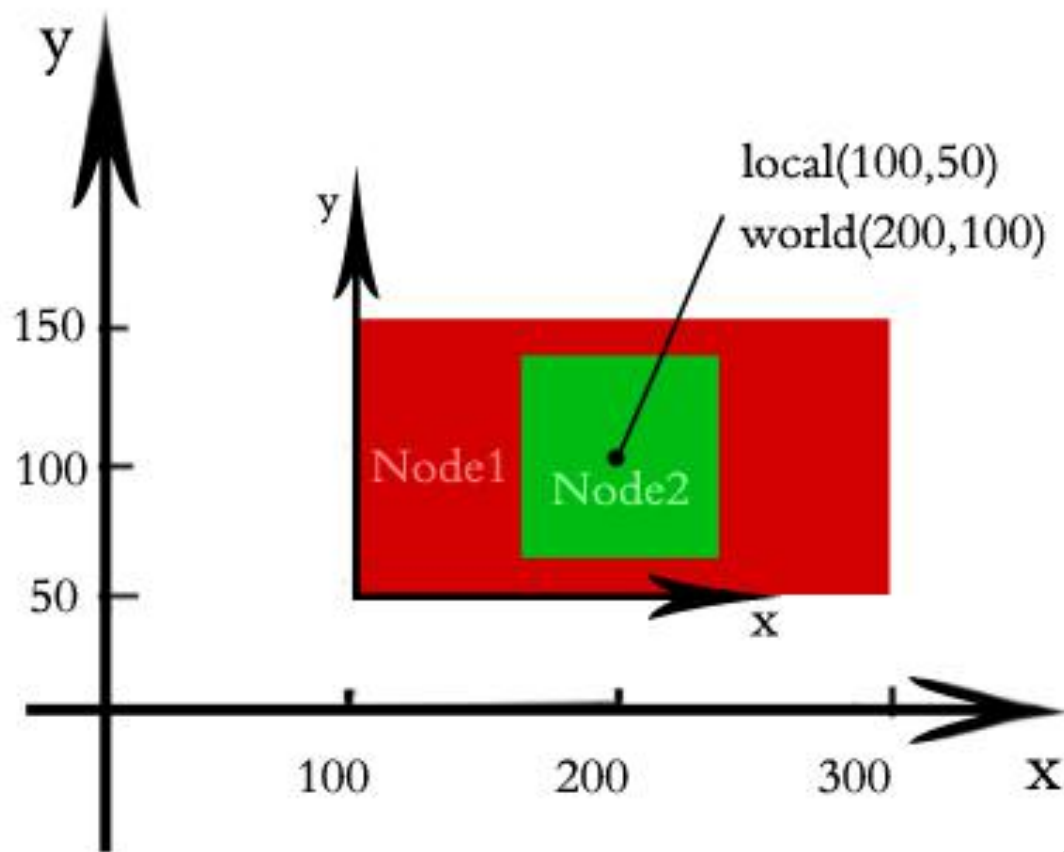
- 锚点就是给节点定位和仿射变换的基准点

取值范围 $[0, 1]$ ，层的默认锚点为 $(0, 0)$ ，其他节点为 $(0.5, 0.5)$ ，可设置锚点忽略



本地坐标系VS世界坐标系、z轴

- 本地坐标系 (100, 50) , 世界坐标系 (200, 100)



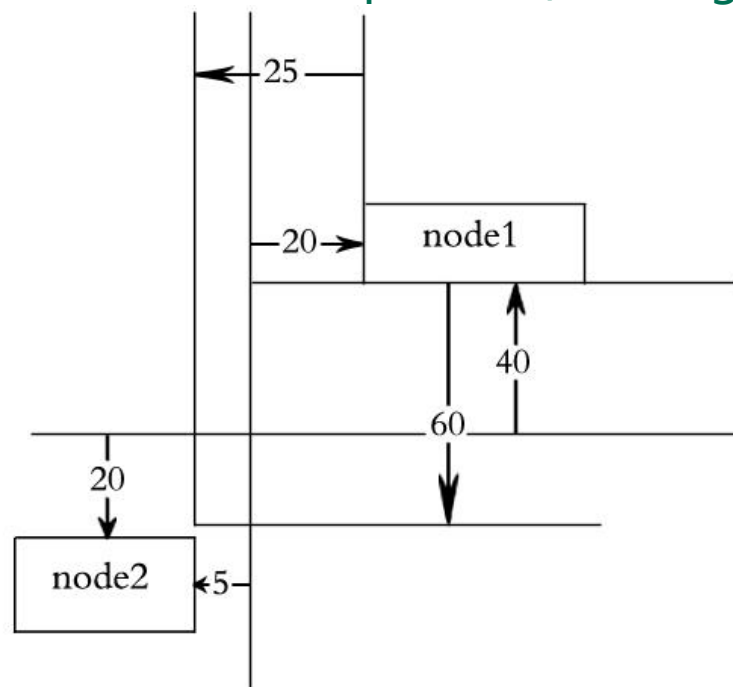
坐标系实验

本地坐标系VS世界坐标系、z轴

- node1的位置(20,40), 锚点(0,0), node2位置(-5,-20), 锚点是(1,1)

```
var point = node1.convertToNodeSpace(node2.getPosition());
```

```
var point = node1.convertToNodeSpaceAR(node2.getPosition());
```



参数为要转换的坐标

坐标系变换实验一

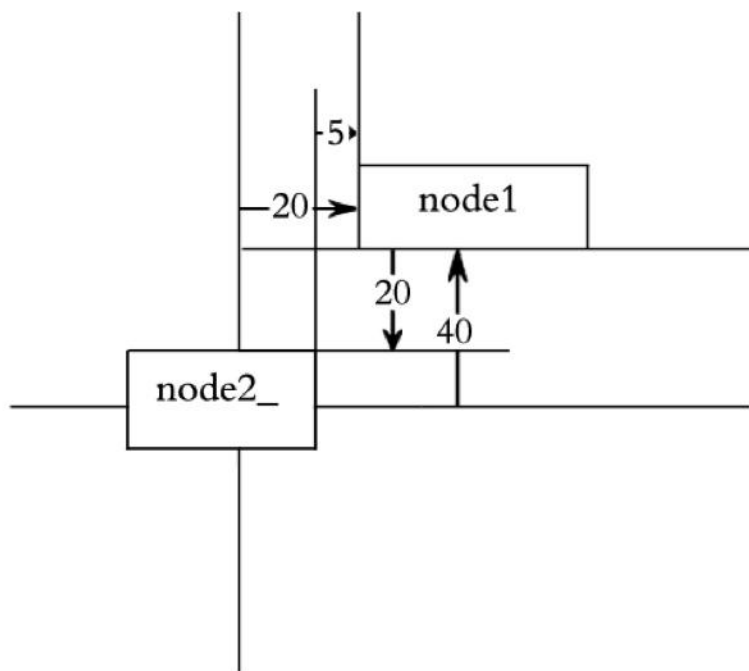
node2世界坐标转换为node1本地坐标系中的坐标, point等于(-25,-60)

本地坐标系VS世界坐标系、z轴

- Node1为根节点位置(20,40)，锚点(0,0)，node2_子节点位置(-5,-20)，锚点是(1,1)

```
var point = node1.convertToWorldSpace(node2_.getPosition());
```

```
var point = node1.convertToWorldSpaceAR(node2_.getPosition());
```

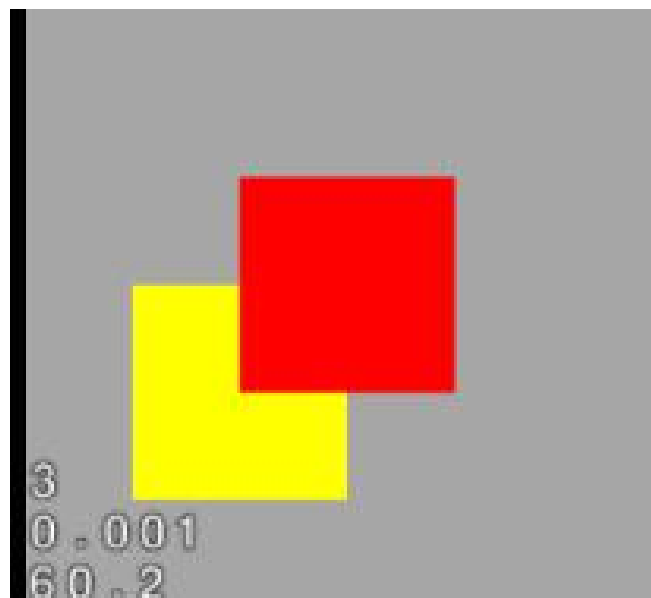
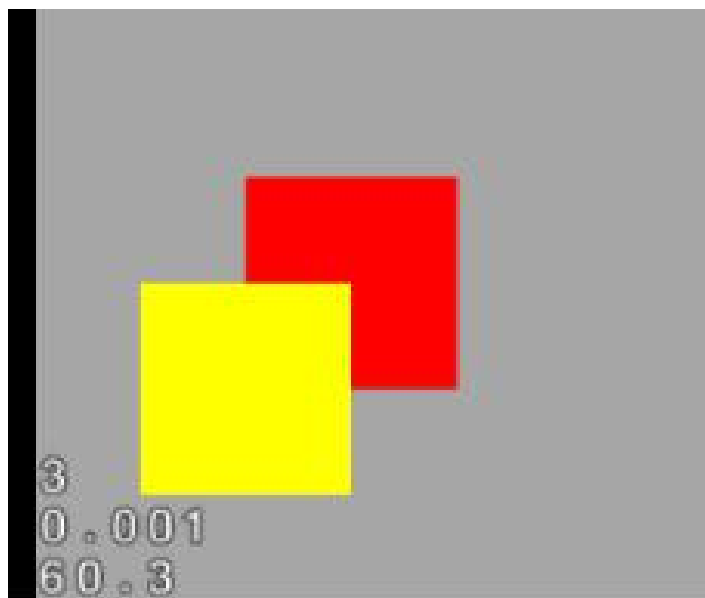


坐标系变换实验二

node2_在node1本地坐标系中的坐标转换为世界坐标，其结果是point等于(15,20)

z轴

- LocalZOrder
- addChild方法中第二个参数



核心概念

- 坐标系、节点 (Node)
- 定时器 (Schedule)
- 标签及菜单 (Label、Menu、MenuItem)
- 场景与层 (Scene、Layer)
- 导演 (Director)
- 精灵 (Sprite)
- 动作与动画 (Action、Animation)

Node节点定时器

/*开启定时器*/

- scheduleUpdate();//update方法循环调用
- schedule(callback,interval,repeat,delay);
- scheduleOnce();

/*停止定时器*/

- unscheduleUpdate();
- unschedule(callback);
- unscheduleAllCallback();

定时器实验

The background of the slide is decorated with numerous overlapping circles in various shades of green and yellow, scattered across the top and right sides.

Thank You!