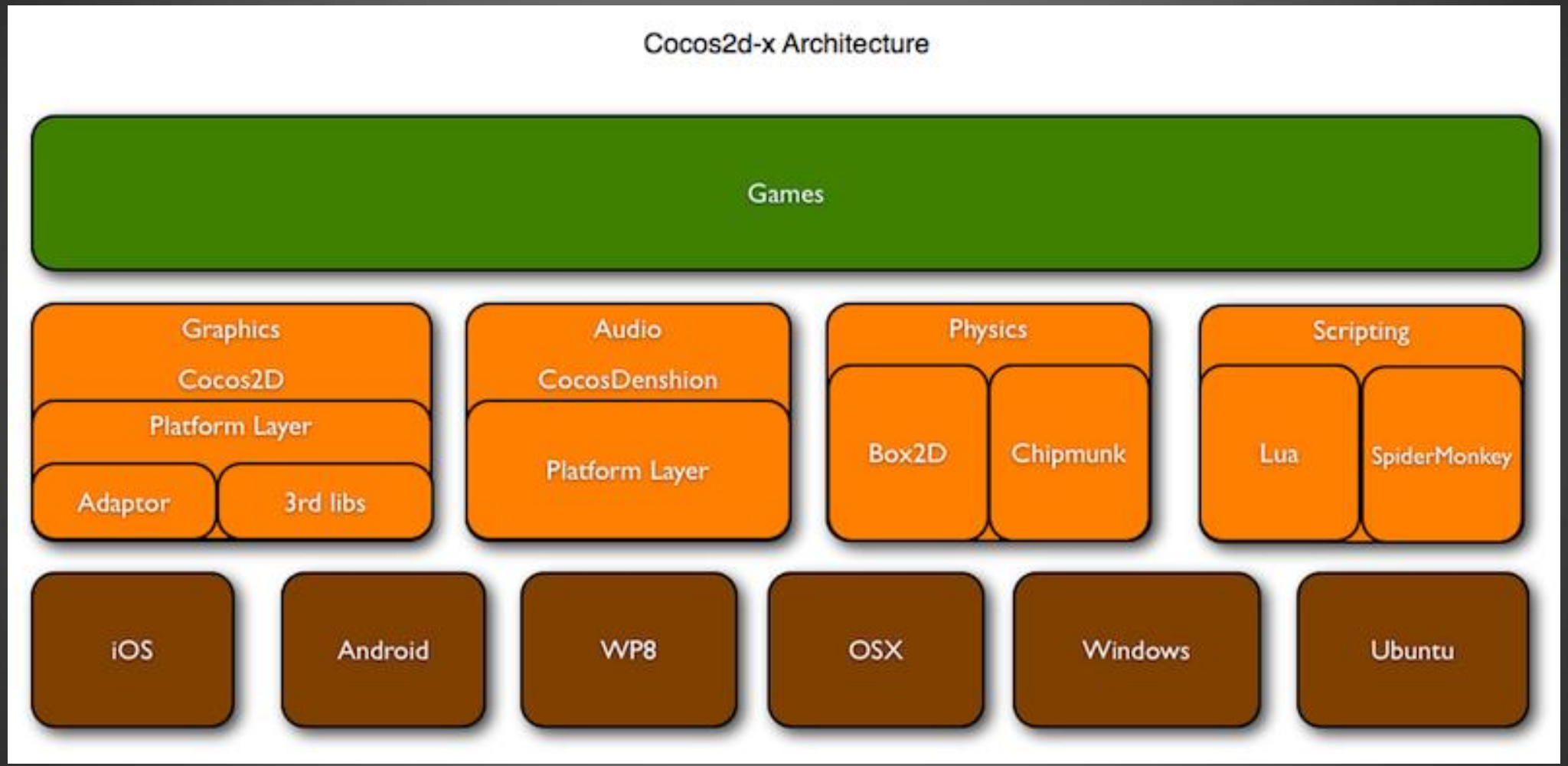
Cocos2dx 游戏开发教程

目 录 / contents

Cocos2d-x引擎架构



Cocos2d-x引擎架构

Cocos2d-x引擎主要由以下四个模块组成：
1.Graphics：图形模块是Cocos2d-x最重要的一个模块，该模块提供了Sprite，Action等等的实现。

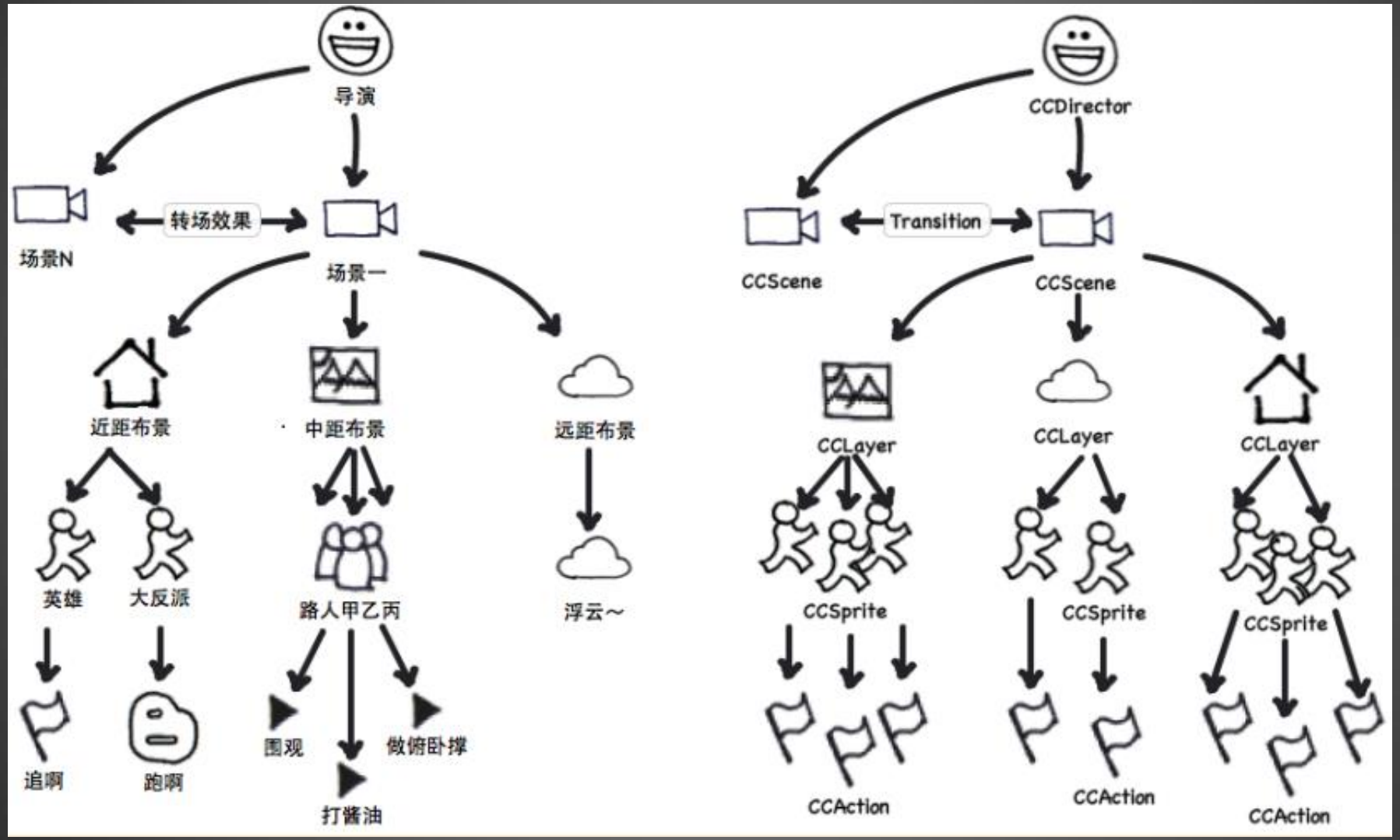
2.Audio：声音模块提供了游戏的音效支持。

cocos2d-x引擎架构

3.Physic：物理模块提供了物理引擎支持，底层类库有Chipmunk与Box2d两种方式

4.Scripting:脚本支持

Cocos2d-x游戏设计理念



Cocos2d-x游戏设计理念

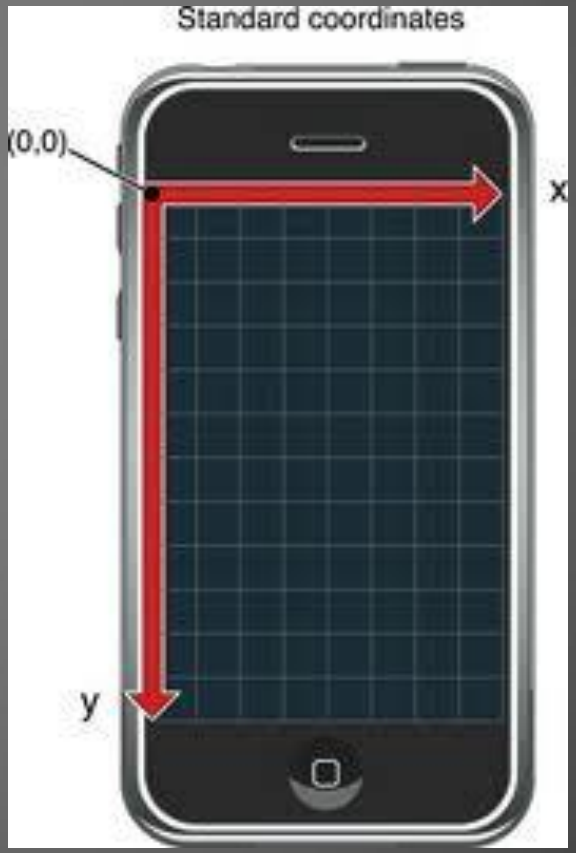
1.Director:是游戏的大脑，通过它来控制场景跳转，事件分发等功能。
2.Scene:通常一个场景是游戏的一个界面
3.Layer：一个场景包含多个图层，每个图层有各自的角色信息
4.Sprite：可以理解为游戏的每一个角色为Sprite，它时处理图片与动画的基本层
5.Action：角色可以通过Action实现运动，旋转等

Cocos2d-x坐标系介绍

1. UI坐标系

IOS/Android/Windows SDK中的通用UI坐标系：

- 起点坐标(x=0, y=0)位于左上角
- X轴从屏幕最左边开始，由左向右渐增
- Y轴坐标从屏幕最上方开始，由上向下渐增



注意：Cocos2d-x中触摸返回的坐标为UI坐标

Cocos2d-x坐标系介绍

2. Cocos2d-x坐标系

Cocos2d-x使用的是Opengl坐标系：

- 起点坐标(x=0, y=0)位于左下角
- X轴从屏幕最左边开始，由左向右渐增
- Y轴坐标从屏幕最下方开始，由下向往上增

注意：Cocos2d-x中setPosition使用的是Cocos2d-x坐标系

Cocos2d-x坐标系介绍

1. 世界坐标系

与GL坐标系相同，是屏幕全局坐标

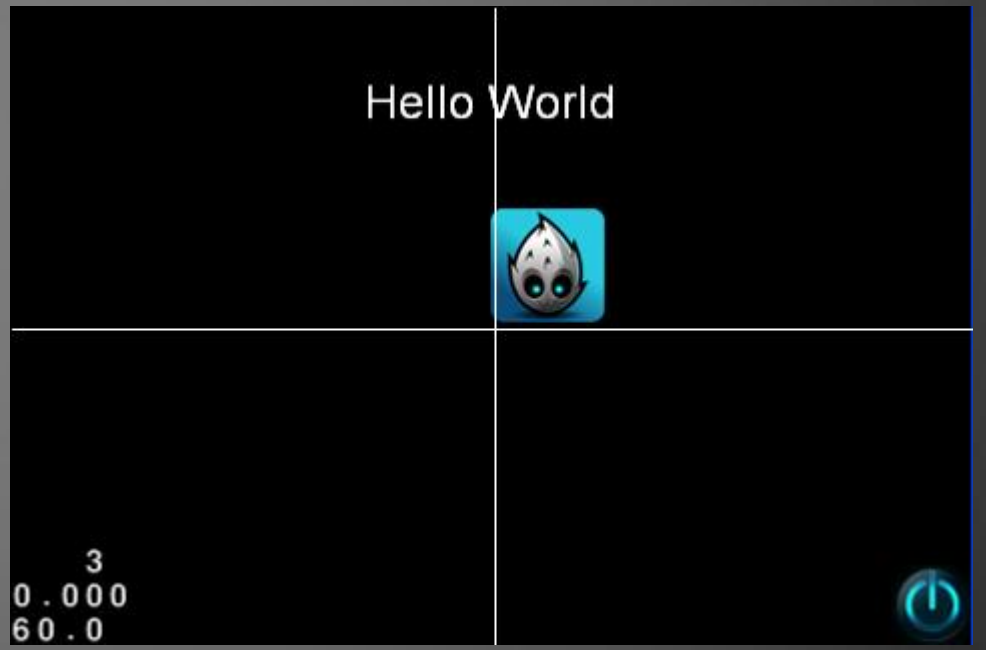
2. 本地坐标系：

是节点（CCNode）的坐标系，原点在节点左下角，x轴向右，y轴向上。cocos2d中的元素是有父子关系的层级结构，我们通过Node的position设定元素的位置使用的是相对与其父节点的本地坐标系而非世界坐标系。最后在绘制屏幕的时候cocos2d会把这些元素的本地坐标映射成世界坐标系坐标。

注意：Cocos2d-x中setPosition使用的是对于父节点的本地坐标

Cocos2d-x坐标系介绍

锚点（AnchorPoint）：锚点可认为是一个对象的中心点，
父对象通过把子对象的锚点放到position上来实现布局。



Cocos2d-x坐标系介绍

本地坐标与世界坐标的相互转换：

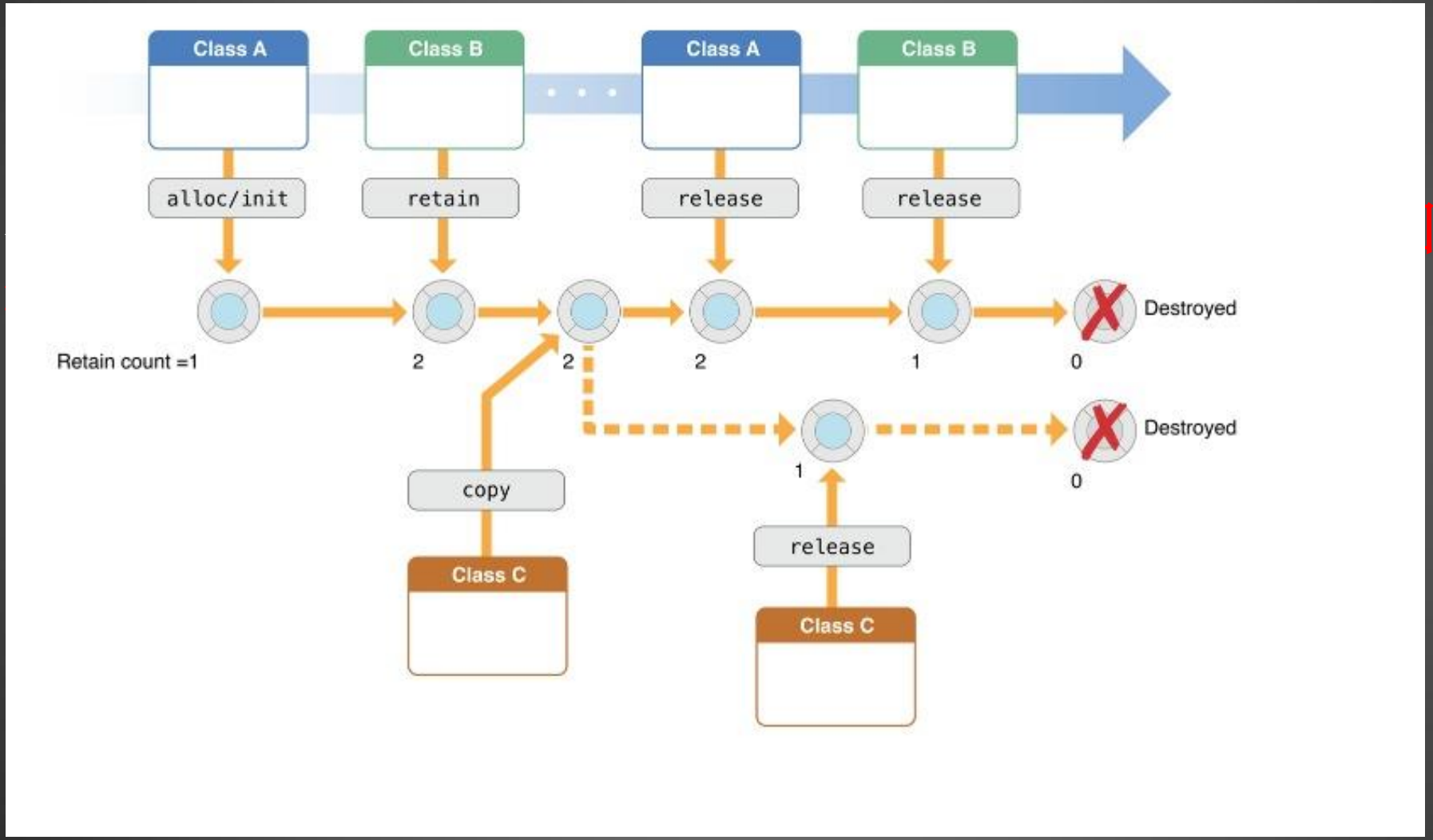
- CCPoint CCNode::convertToNodeSpace(const CCPoint& worldPoint);
- CCPoint CCNode::convertToWorldSpace(const CCPoint& nodePoint);
- CCPoint CCNode::convertToNodeSpaceAR(const CCPoint& worldPoint);
- CCPoint CCNode::convertToWorldSpaceAR(const CCPoint& nodePoint);

内存管理机制

C++内存管理机制

由程序员控制，谁分配谁释放

内存管理机制

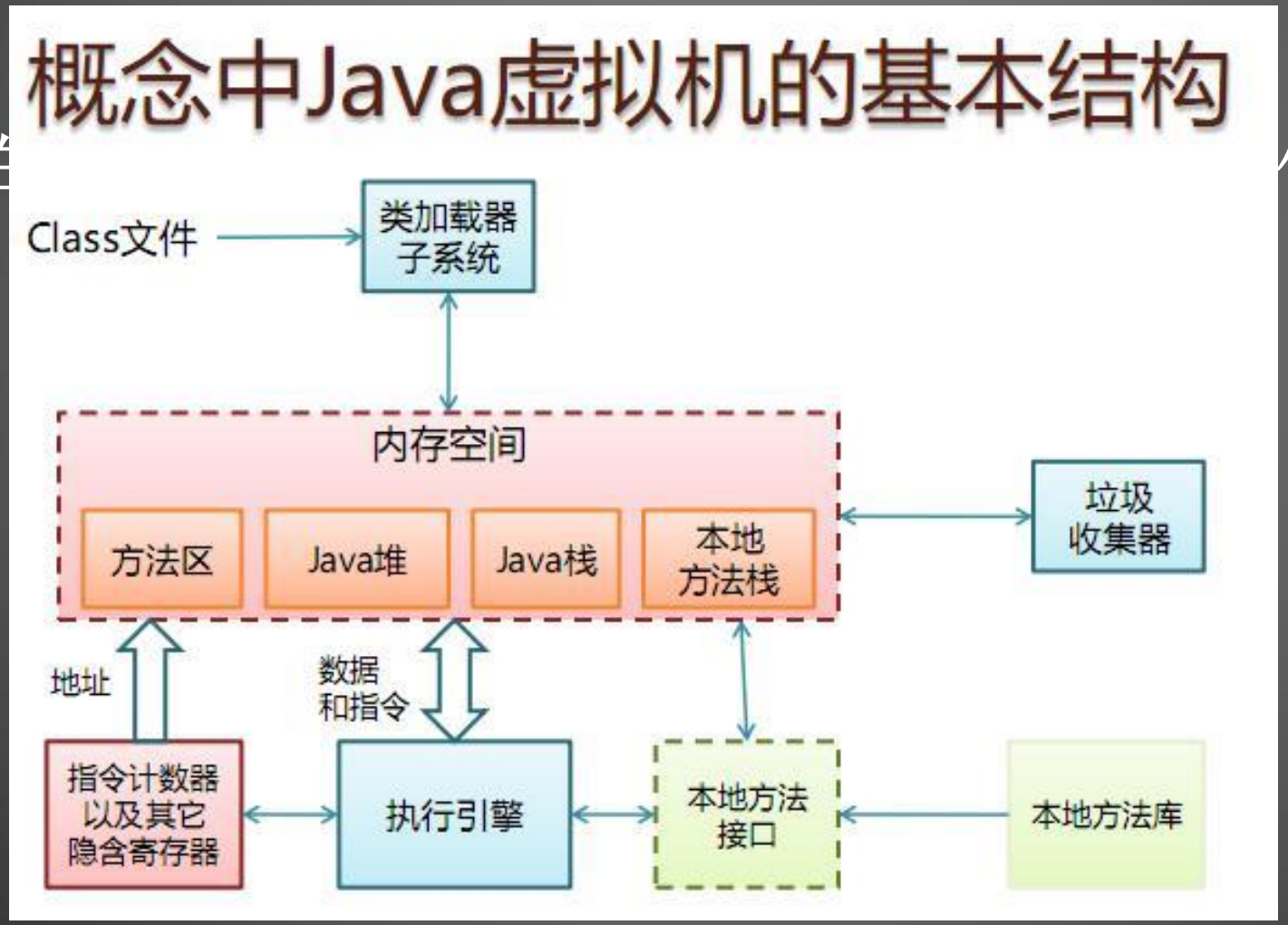

Class A
Class B
Class A
Class B
alloc/init
retain
release
release
Retain count =1
2
2
2
1
0
Destroyed
copy
Class C
1
0
Destroyed
release
Class C

收

内存管理机制

Java内存管理机制

自动的 使用的内存



COCOS2DX

内存管理机制

Cocos2d-x内存管理机制

虽然Cocos2d-x是用C++实现，但Cocos2d-x采用的是Object-C的内存管理方式。通过给每个对象维护一个引用计数器，记录该对象当前被引用的次数。当对象增加一次引用时，计数器加1；而对象失去一次引用时，计数器减1；当引用计数为0时，标志着该对象的生命周期结束，自动触发对象的回收释放。

内存管理机制

Cocos2d-x内存管理机制

retain： 引用计数加1；

release： 引用计数减1

autorelease： 引用次数并没有减1，只是告诉自动释放池，该对象对内存释放由自动释放池管理

```
Ref* Ref::autorelease()
{
    PoolManager::getInstance()->getCurrentPool()->addObject(this);      // 交给自动释放池管理
    return this;
}
```

课后作业

模拟捕鱼达人场景，共有主界面与游戏界面两个界面：
主界面：有一个开始按钮，点击进入游戏界面
游戏界面：游戏界面有两个Layer。WeaponLayer锚点位于左下角，坐标设为（0,0）。其上有一个weapon精灵，初始位置（屏幕中间，50）。FishLayer锚点位于左下角，坐标为（0， 屏幕中间）。其上有一个fish精灵，初始位置（屏幕中间， 0）。点击屏幕任意位置，fish精灵游动到改位置。点击Shoot按钮，weapon发射至当前fish精灵位置。

主界面


My Game
Start
GL verts: 6
GL calls: 1
59.9 / 0.001


中山大學
SUN YAT-SEN UNIVERSITY


COCOS2DX

游戏界面

自适应屏幕

在AppDelegate中设置图片资源大小以达到自适应屏幕效果

```
glview->setDesignResolutionSize(800, 480, ResolutionPolicy::EXACT_FIT);
```

加载游戏资源

在AppDelegate中预先加载

```
// load game resource
SpriteFrameCache::getInstance()->addSpriteFramesWithFile("fish.plist");
char totalFrames = 9;
char frameName[20];
Animation* fishAnimation = Animation::create();
for (int i = 1; i <= totalFrames; i++)
{
    sprintf(frameName, "fish13_0%d.png", i);
    fishAnimation->addSpriteFrame(SpriteFrameCache::getInstance()->getSpriteFrameByName(frameName));
}
fishAnimation->setDelayPerUnit(0.1);
AnimationCache::getInstance()->addAnimation(fishAnimation, "fishAnimation");
```

COCOS2DX

使用预加载资源

```
//create a fish sprite and run animation
m_fish = Sprite::createWithSpriteFrameName("fish13_01.png");
Animate* fishAnimate = Animate::create(AnimationCache::getInstance()->getAnimation("fishAnimation"));
m_fish->runAction(RepeatForever::create(fishAnimate));
m_fish->setPosition(visibleSize.width / 2, 0);
m_fishLayer->addChild(m_fish);
```

触屏响应

```
//add touch listener
EventListenerTouchOneByOne* listener = EventListenerTouchOneByOne::create();
listener->setSwallowTouches(true);
listener->onTouchBegan = CC_CALLBACK_2(HelloWorld::onTouchBegan, this);
Director::getInstance()->getEventDispatcher()->addEventListenerWithSceneGraphPriority(listener, this);
```

THE END

THANKS FOR WATCHING