Cocos2d-x基础概念

骆铭涛

游戏案例：捕鱼达人

点击控制大炮的方向，发射子弹

- 点击屏幕触发触屏响应事件，监听器返回触摸点的坐标（Touch坐标）
- Touch坐标是触摸点OpenGL坐标系中的点坐标，也就是在Cocos2d-x坐标系位置

Coco2d-x坐标系


(0, 0)
x
标准屏幕坐标系
y
y
Cocos2d-x坐标系
GL verts: 264
GL calls: 3
53.1 / 0.007
x
(0, 0)

点击控制大炮的方向，发射子弹

- Cocos2d中的元素是有父子关系的层级结构，我们通过Node的setPosition设定元素的位置使用的是相对与其父节点的本地坐标，最后在绘制屏幕的时Cocos2d会把这些元素的本地坐标映射成世界坐标系坐标

世界坐标与本地坐标

- 世界坐标系也叫做绝对坐标系。世界坐标系和GL坐标系一致，原点在屏幕左下角。
- 本地坐标系也叫做物体坐标系，是和特定物体相关联的坐标系。每个物体都有它们独立的坐标系，当物体移动或改变方向时，和该物体关联的坐标系将随之移动或改变方向。

点击控制大炮的方向，发射子弹

- 触摸点的世界坐标转换为精灵所在层的本地坐标
- 计算精灵坐标与触摸点坐标的关系，做出相应的反应

本地坐标与世界坐标的相互转换

1.CCPoint convertToNodeSpace(const CCPoint& worldPoint);
2.CCPoint convertToWorldSpace(const CCPoint& nodePoint);
3.CCPoint convertToNodeSpaceAR(const CCPoint& worldPoint);
4.CCPoint convertToWorldSpaceAR(const CCPoint& nodePoint);

参考链接：

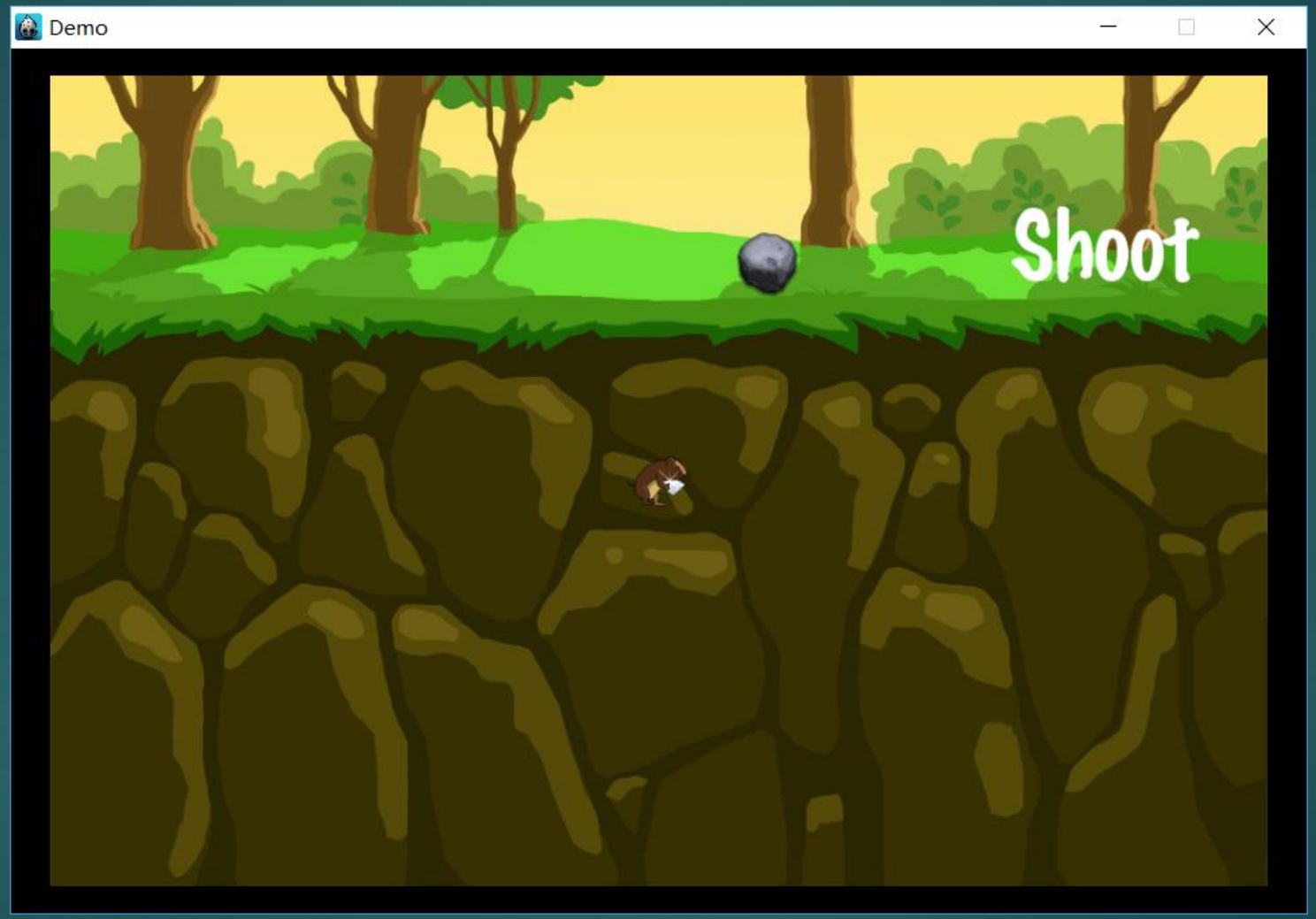
http://blog.163.com/zjf_to/blog/static/201429061201292193855498/

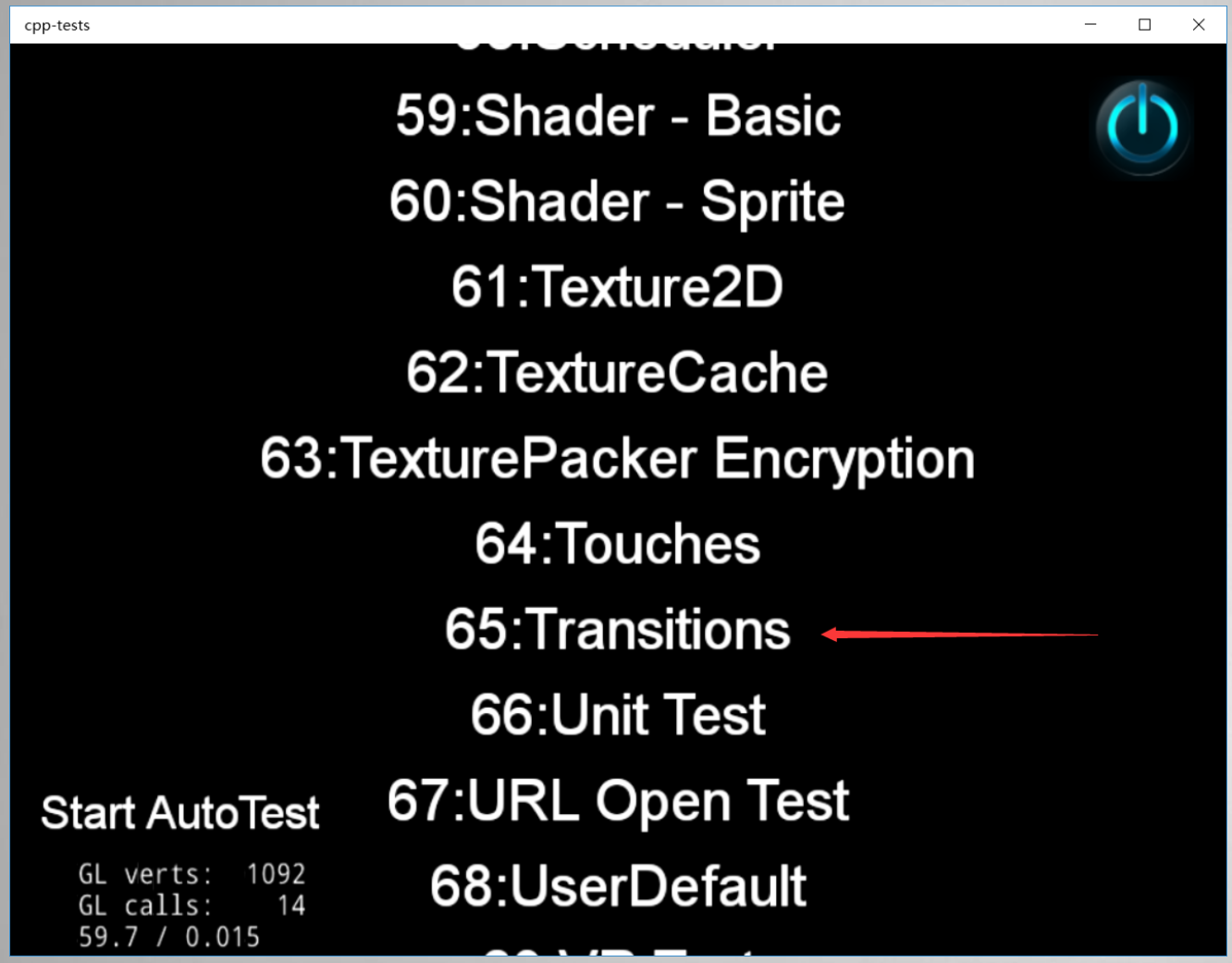
http://www.cnblogs.com/lyout/p/3292702.html

作业

- 新版黄金矿工游戏，共有两个界面：主界面与游戏界面
- 主界面：在demo代码基础上完善场景，添加开始按钮（MenuItem），点击进入游戏界面。
- 游戏界面：两个Layer，StoneLayer锚点位于左下角，坐标设为(0,0)，其上有一石头精灵，初始坐标为（560，480）；MouseLayer锚点位于左下角，坐标设为(0,屏幕高度的一半)，其上有一老鼠精灵，初始坐标为（屏幕宽度的一半，0）。有一个Label，作为shoot按钮。
- 游戏要求：游戏开始后，点击屏幕任意位置，在该位置添加一块奶酪，老鼠跑到该位置吃掉奶酪；点击shoot按钮，石头发射到老鼠所在的位置，老鼠跑开，留下钻石。
- 加分项：尝试添加一两个动画
- 作业提交：提交实验报告（文档），Classes（文件夹），Resources（文件夹）。实验报告要求有截图。


Demo
GOLD MINER
START


Demo
Shoot


cpp-tests
59:Shader - Basic
60:Shader - Sprite
61:Texture2D
62:TextureCache
63:TexturePacker Encryption
64:Touches
65:Transitions
66:Unit Test
67:URL Open Test
68:UserDefault
Start AutoTest
GL verts: 1092
GL calls: 14
59.7 / 0.015

场景切换

触屏响应

```
// 单点触摸事件监听器
EventListenerTouchOneByOne* listener = EventListenerTouchOneByOne::create();

// 给触摸监听函数设置吞没事件，使得触摸上面的层的时候事件不会向下传递
listener->setSwallowTouches(true);

// 设置回调函数
listener->onTouchBegan = CC_CALLBACK_2(GameSence::onTouchBegan, this);

// 使用EventDispatcher注册事件监听器
// _eventDispatcher->addEventListenerWithSceneGraphPriority(listener, this);
Director::getInstance()->getEventDispatcher()->addEventListenerWithSceneGraphPriority(listener, this);
```

触屏响应

- 有三个函数onTouchBegan，onTouchMove和onTouchEnded可供重写，本次作业重写onTouchBegan方法即可

```
bool GameSence::onTouchBegan(Touch *touch, Event *unused_event) {

    return true;
}
```

序列帧动画

AppDelegate.cpp中预先加载动画资源

```
// load game resource
SpriteFrameCache::getInstance()->addSpriteFramesWithFile("general-sheet.plist");
char totalFrames = 3;
char frameName[20];
Animation* legAnimation = Animation::create();

for (int i = 0; i < totalFrames; i++)
{
    sprintf(frameName, "miner-leg-%d.png", i);
    legAnimation->addSpriteFrame(SpriteFrameCache::getInstance()->getSpriteFrameByName(frameName));
}
legAnimation->setDelayPerUnit(0.1);
AnimationCache::getInstance()->addAnimation(legAnimation, "legAnimation");
```

序列帧动画

```
<key>miner-leg-0.png</key>
<dict>
    <key>frame</key>
    <string>{{150,804},{101,154}}</string>
    <key>offset</key>
    <string>{2,3}</string>
    <key>rotated</key>
    <false/>
    <key>sourceColorRect</key>
    <string>{{4,0},{101,154}}</string>
    <key>sourceSize</key>
    <string>{105,160}</string>
</dict>
```

序列帧动画

使用动画资源

```
auto leg = Sprite::createWithSpriteFrameName("miner-leg-0.png");
Animate* legAnimate = Animate::create(AnimationCache::getInstance()->getAnimation("legAnimation"));
leg->runAction(RepeatForever::create(legAnimate));
leg->setPosition(110 + origin.x, origin.y + 102);
this->addChild(leg, 1);
```