**现代操作系统应用开发实验报告**

——刘亚辉 16340157

Cocos2dx第十周实验报告

1. 参考资料
2. PPT课件 (Scene类，Sprite类，Action类，序列帧动画，坐标系，内存管理)
3. <http://blog.163.com/zjf_to/blog/static/201429061201292193855498/> (本地坐标与世界坐标)
4. <http://www.cnblogs.com/lyout/p/3292702.html> (本地坐标与世界坐标)
5. 实验步骤

* 需求及流程：

1. 完善主界面场景，增加开始按钮，点击切换到游戏界面。

2. 游戏界面中，实现mouse和stone及其layer的创建及布局，并且创建一个shoot按钮。

3. 点击游戏界面，放置一块奶酪，老鼠可以跑过去将其吃掉，点击shoot，石头发射到老鼠所在位置，老鼠跑开留下钻石。

* 详细实现过程：

1.场景转化：

GameScene和MenuScene场景类继承了Scene类型，在其成员函数createScene()中分别返回GameScene::create()和MenuScene::create()即可。

在MenuScene场景类中，完善场景，增加START图片按钮，goldMinerTitle以及menuStartGold精灵。其中START按钮回调函数如下：通过调用导演类的replaeScene方法即可完成场景的替换，可以利用TransitionFadeUp方法来实现场景切换的样式。

1. **void** MenuScene::menuStartCallback(cocos2d::Ref\* pSender) {
2. Director::getInstance()->replaceScene(TransitionFadeUp::create(1.0f, GameScene::createScene()));
3. }

2.游戏界面制作：

动画的制作：

Animation类描述的是动画，Sprite显示动画的动作是一个Animate对象。动画动作Animate是Sprite显示动画的动作，由动画对象Animation创建，由Sprite执行。

在本次作业中，采用的是手动初始化Animation类：利用SpriteFrameCache加载xml/plist（plist文件中保存了精灵的动画帧）。然后通过图片名读取SpriteFrameCache中的动画帧，调用Animation的接口addSpriteFrame添加动画。Mouse移动的动画如下：

1. // Mouse resource
2. SpriteFrameCache::getInstance()->addSpriteFramesWithFile("level-sheet.plist");
3. Animation\* mouseAnimation = Animation::create();
5. **for** (**int** i = 0; i < 8; i++) {
6. std::string mouseFramesName = "";
7. mouseFramesName = mouseFramesName+"gem-mouse-" + **char**(i+'0') + ".png";
8. mouseAnimation->addSpriteFrame(SpriteFrameCache::getInstance()->getSpriteFrameByName(mouseFramesName));
9. }
10. mouseAnimation->setDelayPerUnit(0.1);
11. AnimationCache::getInstance()->addAnimation(mouseAnimation, "mouseAnimation");

然后调用该动画：

1. mouse = Sprite::createWithSpriteFrameName("gem-mouse-0.png");
2. Animate\* mouseAnimate = Animate::create(AnimationCache::getInstance()->getAnimation("mouseAnimation"));
3. mouse->runAction(RepeatForever::create(mouseAnimate));
4. mouse->setAnchorPoint(Vec2(0.5,0.5));
5. mouse->setPosition(Vec2(visibleSize.width/2, 0));

利用Sprite的createWithSpriteFrameName方法来创建Sprite实例，然后创建动画动作Animate实例，利用之前加入到AnimationCache中的动画对象实例，通过getAnimation和名字即可获取到该实例，作为Animate的create参数即可创建动画动作实例。调用mouse的runAction方法使用该动画。剩下动画的创建与此类似。

3.鼠标点击放置奶酪事件、shoot点击事件：

（1）鼠标点击放置奶酪事件：

1. // 单触摸事件监听器
2. EventListenerTouchOneByOne\* listener = EventListenerTouchOneByOne::create();
3. listener->setSwallowTouches(**true**);
4. listener->onTouchBegan = CC\_CALLBACK\_2(GameScene::onTouchBegan, **this**);
5. Director::getInstance()->getEventDispatcher()->addEventListenerWithSceneGraphPriority(listener, **this**);

设置事件监听器，EventListenerTouchOneByOne为单触摸事件监听器，设置setSwallowTouches为true，则该触摸事件不会被其他比他优先级更高的监听器所捕获（捕获优先级由小到大），然后设置触摸事件回调函数，该回调函数为CC\_CALLBACK\_2类型，有两个占位的参数；之后调用导演类的事件分派对象的addEventListener-WithSceneGraphPriority方法，将监听者添加到事件派发管理器中，这样当你点击到被监听者时，就会触发相应的事件，该方法注册的优先级为0。将该场景的触摸事件分配给定义的listener，由其监听。其中回调函数定义如下：

1. **bool** GameScene::onTouchBegan(Touch \*touch, Event \*unused\_event) {
3. //location是世界坐标，将cheese至于世界坐标下
4. auto location = touch->getLocation();
5. auto cheese = Sprite::create("cheese.png");
6. cheese->setPosition(location);
7. cheese->setAnchorPoint(Vec2(0.5, 0.5));
8. **this**->addChild(cheese);
10. //然后将cheese的坐标转化为mouse的相对坐标，然后调用MoveBy即可
11. auto relativeLocation=mouse->convertToNodeSpaceAR(cheese->getPosition());
12. auto moveBy = MoveBy::create(1, relativeLocation);
13. mouse->runAction(moveBy);
15. auto fadeOut = FadeOut::create(4);
16. cheese->runAction(fadeOut);
17. **return** **true**;
18. }

在回调函数中，参数touch表示当前的点击位置，调用其getLocation方法获得世界坐标，然后将cheese置于该坐标，并设置颜色变淡的动作。对于mouse，我们需要将cheese的坐标转化为mouse的相对坐标，调用mouse->convertToNodeSpaceAR方法，然后定义MoveBy对象，实现移动的动作。

（2）shoot点击事件：

和一般的label按钮点击事件一致，设置按钮的回调函数：解释见注释。

1. **void** GameScene::ShootCallback(cocos2d::Ref\* pSender) {
2. auto visibleSize = Director::getInstance()->getVisibleSize();
4. //Create a new stone
5. auto newStone = Sprite::create("stone.png");
6. newStone->setPosition(Vec2(560, 480));
7. newStone->setAnchorPoint(Vec2(0.5, 0.5));
8. stoneLayer->addChild(newStone);
10. //首先将mouse的position转化为相对于mouseLayer的世界坐标（即该Sprite在哪个layer上，就相对于那个layer做转化），
11. //然后将该世界坐标转化为相对于stoneLayer的相对坐标，这时，此坐标与stone位于同一个坐标系下，就可以使用moveTo动作来完成任务
12. auto relativeLocation = stoneLayer->convertToNodeSpaceAR(mouseLayer->convertToWorldSpaceAR(mouse->getPosition()));
13. auto moveTo = MoveTo::create(1, relativeLocation);
14. newStone->runAction(moveTo);
15. auto fadeOut = FadeOut::create(4);
16. newStone->runAction(fadeOut);
18. // Add a diamod on the mouseLayer that the mouse left
19. // diamond Animate
20. auto diamond = Sprite::createWithSpriteFrameName("pulled-diamond-0.png");
21. Animate\* diamondAnimate = Animate::create(AnimationCache::getInstance()->getAnimation("diamondAnimate"));
22. diamond->runAction(RepeatForever::create(diamondAnimate));
23. diamond->setAnchorPoint(Vec2(0.5, 0.5));
24. diamond->setPosition(mouse->getPosition());
25. mouseLayer->addChild(diamond, 0);
27. //首先获取一个mouse可以移动到的地方， 通过以下方法获得坐标即为世界坐标，
28. //然后相对mouseLayer做一个坐标转化，就可以将此坐标与mouse至于同一个坐标系下，通过调用MoveTo动作即可完成任务
29. auto targetWorldPosition = Vec2(visibleSize.width\*random<**float**>(0, 1), visibleSize.height\*random<**float**>(0, 0.6));
30. relativeLocation = mouseLayer->convertToNodeSpaceAR(targetWorldPosition);
31. moveTo = MoveTo::create(1, relativeLocation);
32. mouse->runAction(moveTo);
34. //如下：也可以直接将世界坐标转化为相对于mouse的相对坐标，然后调用moveBy动作即可
35. //relativeLocation = mouse->convertToNodeSpaceAR(targetWorldPosition);
37. //auto moveBy = MoveBy::create(1, relativeLocation);
38. //mouse->runAction(moveBy);
39. }
40. 关键步骤截图

界面效果如下：



场景切换，点击鼠标，放置奶酪，老鼠跑向奶酪：



点击shoot：石头射向老鼠，老鼠抛开，留下钻石



1. 亮点与改进

增加老鼠移动的动画和钻石闪烁的动画。

1. 遇到的问题

本地坐标与世界坐标转化的问题：首先本地坐标是相对与所指定的参照对象的，如果时两个layer（layer1、layer2）上的两个sprite（sprite1，sprite2）坐标转化，需要先将prite1的坐标转化为相对layer1的世界坐标，然后通过该世界坐标获得相对于layer2的相对坐标，此时sprite1和sprite2的坐标相对于同一参考系，所以利用MoveTo动作即可从sprite1到达sprite2。或者将得到的世界坐标相对于sprite2做一个坐标转化，然后利用MoveBy动作可以完成相同的任务。

在设置位置时，通过设置锚点可以使得位置的设置更加精确。