### 2.3 定时器的原理及使用

在游戏中实现运动的另一种方式。如在一些游戏中，主角永远在屏幕的中央，但通过背景的不断平移看起来就像主角在前进一样。

定时器就是系统所提供的一个专用对象，可以在每一帧自动地调用回调函数，通过回调函数的重写达到控制游戏的效果。

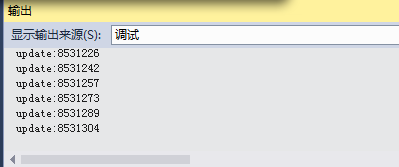
**【本节知识点】**

* **定时器的基本用法**
* **定时器的缺陷及改进**
* **定时器的注册和取消**
* **利用定时器的特点实现自己的触发器**

（1）定时器的基本使用方法

Cocos2d-x中提供了一个对象schedule，可被简单看做是一个定时器，通过它可以按照一定的频率不断地调用某个函数，从而达到观察并掌控游戏中一切的目的。

本节示例的运行结果如下图。在输出面板显示以毫秒为单位的时间戳。



**代码解释：**

**GetCurrentTime方法的作用是获得以毫秒为单位的时间戳。**

**scheduleUpdate方法的作用是不断调用update方法输出当前的时间戳。**

代码：

bool HelloWorld::init()

{

//////////////////////////////

// 1. super init first

if ( !Layer::init() )

{

return false;

}

scheduleUpdate();

return true;

}

void HelloWorld::update(float dt)

{

CCLOG("update:%d", GetCurrentTime());

}

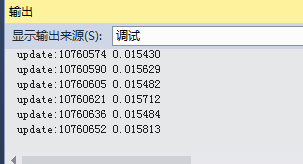
如果将上面update方法实现，修改为如下形式。使用update中的dt参数，它的作用就可以显示上一次调用update方法与本次调用的时间间隔。输出结果如下图。可以看到每次调用的时间间隔基本使相同的。

void HelloWorld::update(float dt)

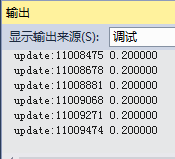
{

CCLOG("update:%d\t%f", GetCurrentTime(), dt);

}



可否在程序代码中设定每次调用update的时间间隔内。下面的代码可以实现。输出结果如下图。可以看到每次调用的时间间隔为0.2。



**代码解释：**

**schedule(schedule\_selector(HelloWorld::myupdate),0.2f);**

**schedule方法中有一个参数为0.2f，就是设置的调用回调函数的时间间隔。第一个参数则是用于确定将被调用的回调函数是哪一个。**

代码：

bool HelloWorld::init()

{

//////////////////////////////

// 1. super init first

if ( !Layer::init() )

{

return false;

}

schedule(schedule\_selector(HelloWorld::myupdate),0.2f);

return true;

}

void HelloWorld::myupdate(float dt)

{

CCLOG("update:%d\t%f", GetCurrentTime(), dt);

}

**（2）定时器的注册与取消**

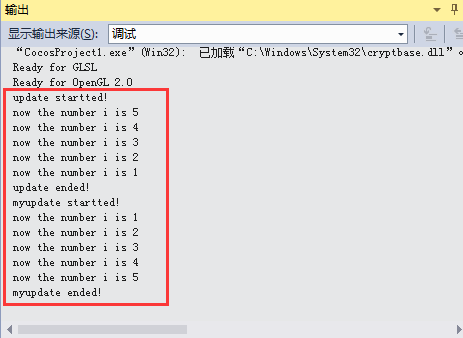
前面学习了使用定时器的方法，如：

**scheduleUpdate();**

**schedule(schedule\_selector(HelloWorld::myupdate),0.2f);**

在实际应用中，往往还需要进行一些别的操作，比如需要在定时器执行10次操作后停止，那就需要一个取消定时器的操作。

本节示例加入update和myupdate两个方法，引入一个整形变量i，实现效果如下图所示。



**代码解释：**

**在代码中可以使用schedule和unschedule方法实现定时器的注册与注销**

代码：

bool HelloWorld::init()

{

if ( !Layer::init() )

{

return false;

}

i = 5;

scheduleUpdate(); **//注册update定时器**

CCLOG("update startted!");

return true;

}

void HelloWorld::update(float dt)

{

if (i > 0)

{

CCLOG("now the number i is %d",i);

i--;

}

else

{

unscheduleUpdate(); **//注销update定时器**

CCLOG("update ended!");

schedule(schedule\_selector(HelloWorld::myupdate), 0.2f); **//注册myupdate定时器**

CCLOG("myupdate startted!");

}

}

void HelloWorld::myupdate(float dt)

{

if (i < 5)

{

i++;

CCLOG("now the number i is %d", i);

}

else

{

unschedule(schedule\_selector(HelloWorld::myupdate)); **//注销myupdate定时器**

CCLOG("myupdate ended!");

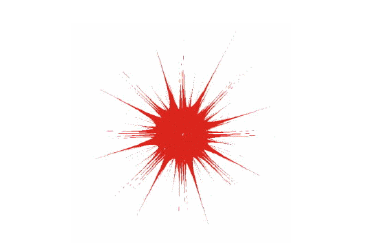
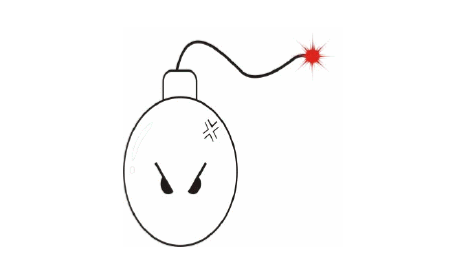
}

}

**（3）使用定时器实现触发器**

本节学习另一种（使用scheduleOnce方法）来注册触发器的方法。

本节示例的运行结果如下图。初始显示如左图，5秒后显示右图。



代码解释：

在代码中，自定义方法bang仅被调用一次。在该方法中调用了scheduleOnce方法来注册触发器。其中2个参数就是触发器将会在多少秒后被调用。

代码：

bool HelloWorld::init()

{

if ( !Layer::init() )

{

return false;

}

auto\* background = LayerColor::create(ccc4(255, 255, 255, 255));

addChild(background);

bomb = Sprite::create("bomb.png");

bomb->setPosition(320, 180);

addChild(bomb);

scheduleOnce(schedule\_selector(HelloWorld::bang),5.0f);

return true;

}

void HelloWorld::bang(float dt)

{

auto\* m\_frameCache = SpriteFrameCache::getInstance();

m\_frameCache->addSpriteFramesWithFile("bang.plist", "bang.png");

Vector<SpriteFrame\*> frameArray;

for (int i = 1; i < 11; i++)

{

auto\* frame = m\_frameCache->getSpriteFrameByName(String::createWithFormat("bang%d.png", i)->getCString());

frameArray.pushBack(frame);

}

Animation\* animation = Animation::createWithSpriteFrames(frameArray);

animation->setLoops(1);//表示无限循环播放

animation->setDelayPerUnit(0.1f);//每两张图片的时间隔，图片数目越少，间隔最小就越小

//将动画包装成一个动作

auto\* action = Animate::create(animation);

bomb->runAction(action);

}

**（4）实验2-6 实现气球定时爆炸效果**

**（5）实验5-3 利用定时器实现人物行走**

**（6）实验5-4 实现回合制游戏**