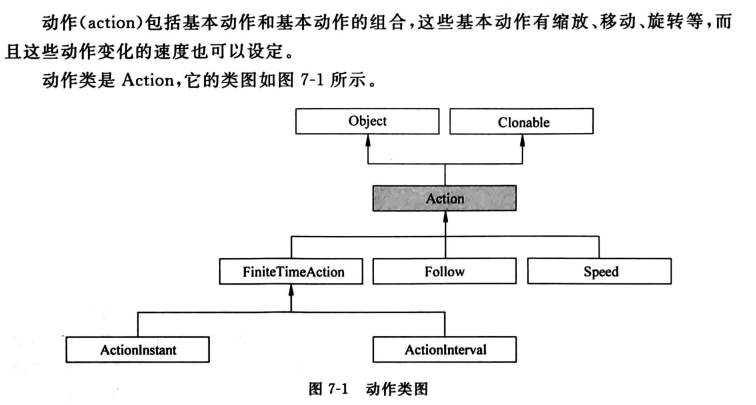
动作、特效和动画

1. 动作

基本动作包括缩放、移动、旋转等，它们的速度也可以设定。



动作类是Action.

①Action有三个子类，FiniteTimeAction是一种受时间限制的动作,Follow是一种允精灵跟随另一个精灵的动作，Speed是在一个动作运行时改变其运动速率。

②此外，FiniteTimeAction有两个子类：ActionInstant和ActionInterval,是两种不同风格的动作类，前者封装瞬时动作，后者封装间隔动作

③Node类有关动作的函数如下

1. Action \*runAction(Action \*action)//运行指定动作，返回值仍然是一个动作对象
2. viod stopAction(Action \*action)//停止指定动作
3. void stopActionByTag(int tag)//通过指定标签停止动作
4. void stopAllActions()//停止所有动作
5. 瞬时动作

瞬时动作就是不等待就立即执行的动作，基类是ActionInstant;

大致实现思路：建立一个场景A，B.A中设立一些文字菜单，并记录其Tag，在点击菜单的回调函数中切换到场景B。B中初始化一些菜单，根据A中菜单的Tag执行相应的动作。详情见Github.

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

注：1.在进行场景切换时，若想给场景传递参数，应使用A::create();若不传递参数可以调用A::createScene();

---------------在创建动作的时候记得在类中定义一个精灵对象sprite,并且可以在init函数中将其初始化到一个图片上

2.有关动作的函数

1. sprite->runAction(FlipX::create(true))//将精灵等Node对象水平方向翻转
2. sprite->runAction(FlipY::create(true))// 将精灵等Node对象垂平方向翻转
3. sprite->runAction(Hide::create())//执行Hide的动作，将精灵对象隐藏
4. sprite->runAction(Show::create())//执行Show动作，将精灵等Node对象显示出来
5. sprite->runAction(ToggleVisibility::create())//进行Node对象显示/隐藏切换
6. sprite->runAction(Place::create(p));将Node对象移动到p点

可以使用CCRANDOM\_0\_1()作为产生随机数的系数产生随机的点

如：Vec2 p=Vec2(CCRANDOM\_0\_1()\*size.eidth,CCRANDOM\_0\_1()\*size.height);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

2.间隔动作（代码示例详见Github）

可以设置duration属性来设置动作的执行时间，间隔动作的基类是

ActionInterval

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

在间隔动作中有很多都是XXXTo,XXXBy前者指运动到绝对位置，后者指运动到相对位置  
////CCMoveBy  创建一个移动的动作   参数1：移动到目标坐标所需的时间 参数2：目标坐标    支持reverse 可以获取其反向动作  
////    CCMoveTo  一样的  
  
////    CCScaleTo   作用：创建一个缩放的动作  
////    参数1：达到缩放大小所需的时间  
////    参数2 ：缩放的比例  
//    CCActionInterval \* scaleto = CCScaleTo ::create(2, 2);  
//    sp->runAction(scaleto);  
//     
////    CCScaleBy  作用:创建一个缩放的动作  
////    参数1：达到缩放大小的所需时间  参数2：缩放比例  
//    CCActionInterval \* scaleby = CCScaleBy::create(2, 2);  
//    CCActionInterval \* actionbyback = scaleby->reverse();  
//    sp->runAction(actionbyback);  
//     
////    CCRotateTo  
////    作用创建一个旋转的动作  
////    参数1：旋转的时间  参数2：旋转饿角度  0 - 360  
//    CCActionInterval \* rotateto = CCRotateTo::create(2, 90);  
//    sp->runAction(rotateto);  
//     
////  CCSkewTo  
////   作用创建一个倾斜的动作  
////    参数1：倾斜到特定角度所需的时间  
////    参数2：x轴的倾斜角度  
////    参数3：y轴的倾斜角度  
//    CCActionInterval \* skewto = CCSkewTo::create(2, 10, 10);  
//    sp->runAction(skewto);  
//     
////    CCJumpTo  
////    作用：创建一个跳的动作  
////    参数1：跳到目标动作位子的所需时间  
////    参数2：目标位置  
////    参数3：跳的高度  
////    参数4跳到目标位置的次数  
//    CCActionInterval\* jumpto = CCJumpTo ::create(2, ccp(300, 200), 50, 4 );  
//    sp->runAction(jumpto);  
//     
////    CCJumpBy  
//    //    作用：创建一个跳的动作  
//    //    参数1：跳到目标动作位子的所需时间  
//    //    参数2：目标位置  
//    //    参数3：跳的高度  
//    //    参数4跳到目标位置的次数  
////    这个支持方向动作reverse  
//    CCActionInterval \* jumpby = CCJumpBy ::create(3, ccp(300, 200), 50, 4);  
//       CCActionInterval \* ac= jumpby->reverse();  
//    sp->runAction(ac);  
     
//    CCBezier  
//ccBezierConfig结构体     
//    ccBezierConfig bezierCon;  
//    bezierCon.controlPoint\_1=CCPointMake(200, 150);//控制点1  
//    bezierCon.controlPoint\_2=CCPointMake(200, 160);//控制点2  
//    bezierCon.endPosition =CCPointMake(340, 100);// 结束位置  
// CCBezierTo  
    //    创建一个贝塞尔曲线运动的动作  
    //    参数1：贝塞尔曲线运动的时间  
//    参数2 ：ccBezierConfig结构体    
//    CCActionInterval \* action = CCBezierTo::create(2, bezierCon);  
//    CCActionInterval \* action = CCBezierBy::create(3, bezierCon);//支持反向  
//    CCActionInterval \* action1 = action->reverse();  
//    sp->runAction(action1);

//    CCFadeIn  
//    作用：创建一个渐变出现的动作  
//    参数是时间  
//    CCActionInterval \* fadein = CCFadeIn::create(2);  
//    sp->runAction(fadein);

//    CCFadeOut  
    //    作用：创建一个渐变消失的动作  
    //    参数是时间  
//    CCActionInterval \* fadeout = CCFadeOut::create(2);  
//    sp->runAction(fadeout);

//CCTintTo  
//    作用：创建一个色彩变化的消失动作  
//    参数1：色彩变化的动作  
//    参数2 ：红色分量  
//    参数3：蓝色分量  
//    CCActionInterval \* tinto = CCTintTo ::create(3, 255, 255, 0);  
//    sp->runAction(tinto);  
   
     
//    CCTintBy  
    //    作用：创建一个色彩变化的出现动作  
    //    参数1：色彩变化的动作  
    //    参数2 ：红色分量  
    //    参数3：蓝色分量   但是家了reverse就是 反向的  
//    CCActionInterval \* tintby = CCTintBy::create(3, 0, 255, 255);  
//    CCActionInterval \* tintby1 = tintby->reverse();  
//    sp->runAction(tintby1);  
     
//    CCBlink  
//    作用 :创建一额闪烁的动作  
//    参数1：闪烁完成的时间  
//    参数2:闪烁的次数

//    CCActionInterval \* blink = CCBlink ::create(3, 10);  
//    sp->runAction(blink);  
     
     
    //    CCDelayTime  
//    创建一个延迟的动作  
//    参数  延迟的时间  
//    CCActionInterval \* delaytime = CCDelayTime::create(3);  
//    sp->runAction(delaytime);  
     
//    CCOrbitCamera  
//    作用：创建一个球面坐标轨迹进行旋转的动作  
//    参数1 ： 旋转轨迹的时间  
//    参数2 ：起始半径  
//    参数3：半径差  
//    参数4：起始z角  
//    参数5：旋转z角的差  
//    参数6：起始x角  
//    参数7：旋转x角的差  
//     CCActionInterval \* orbitcamera = CCOrbitCamera::create(3, 10, 0, 45, 180, 90, 0);  
//    sp->runAction(orbitcamera);

//    CCCardinalSpline  
//    作用：创建数组  点的数组  
    CCPointArray \* array = CCPointArray::create(20);  
    array->addControlPoint(ccp(0,0));  
     array->addControlPoint(ccp(210,0));  
     array->addControlPoint(ccp(210,240));  
     array->addControlPoint(ccp(0,160));  
    array->addControlPoint(ccp(0,0));  
//    CCCardinalSplineTo  
//    作用：创建一个样条曲线轨迹的动作  
//    参数1：完成轨迹所需的时间  
//    参数2：控制点的坐标数组  
//    拟合度  其值= 0 路径最柔和  
//    CCActionInterval  \* CardinalSplineTo=CCCardinalSplineTo::create(3, array, 0);  
//   sp->runAction(CardinalSplineTo);  
//CCCardinalSplineBy  
    //    作用：创建一个样条曲线轨迹的动作  
    //    参数1：完成轨迹所需的时间  
    //    参数2：控制点的坐标数组  
    //    拟合度  其值= 0 路径最柔和  
//    CCActionInterval \* CardinalSplineBy = CCCardinalSplineBy::create(3, array, 0);  
//    sp->runAction(CardinalSplineBy);  
     
//    CCCatmullRomTo  CCCatmullRomBY  
//    作用：创建一个样条插值轨迹  
//    参数1：完成轨迹的时间  
//    参数2：控制点的数组坐标  
//    CCActionInterval \* catmullRomTo = CCCatmullRomTo::create(3, array);  
//    sp->runAction(catmullRomTo);  
     
//    CCFollow  
//    作用：创建一个跟随动作  
//    参数1：跟随的目标对象  
//    跟随范围，离开范围就不再跟随  
      //创建一个参照物spT  
//    CCSprite \* spt = CCSprite::create("Icon.png");  
//    spt->setPosition(ccp(420,40));  
//    addChild(spt);  
//    sp->runAction(CCMoveTo::create(3, ccp(940,sp->getPositionY())));  
//     
//     CCFollow \* follow = CCFollow::create(sp,CCRectMake(0, 0, 960, 320));  
//    this-> runAction(follow);  
     
//    CCEaseBounceIn  
//    目标动作  
//    CCActionInterval\* move = CCMoveTo::create(3, ccp(300, sp->getPositionY()));  
////    让目标动作缓慢开始  
////    参数：目标动作  
//    CCActionInterval \* EaseBounceIn = CCEaseBounceIn::create(move);  
//    sp->runAction(EaseBounceIn);  
     
//CCEaseBounceOut  
//    作用：让目标动作赋予反弹力，且以目标动作结束位子开始反弹  
//    参数目标动作  
//    CCActionInterval \* easeBounceOut = CCEaseBounceOut ::create(move);  
//    sp->runAction(easeBounceOut);  
     
//    CCEaseBounceInOut  
//    作用：让目标动作赋予反弹力，且以目标动作起始与结束位子开始反弹  
//    CCActionInterval \* easeBounceInOut= CCEaseBounceInOut::create(move);  
//    sp->runAction(easeBounceInOut);  
     
//   CCEaseBackIn  
//    作用：让目标动作赋予回力 ， 且以目标动作起点位置作为回力点  
//    参数：目标动作  
//    CCActionInterval \* easeBackIn = CCEaseBackIn::create(move);  
//    sp->runAction(easeBackIn);  
     
//    CCEaseBackOut  
//    作用：让目标动作赋予回力 ， 且以目标动作终点位置作为回力点  
    //    参数：目标动作  
//    CCActionInterval \*easeBackOut = CCEaseBackOut::create(move);  
//    sp->runAction(easeBackOut);  
     
//    CCEaseBackInOut  
//    作用：让目标动作赋予回力 ， 且以目标动作起点和终点位置作为回力点  
    //    参数：目标动作  
//    CCActionInterval \* easeBackInOut = CCEaseBackInOut::create(move);  
//    sp->runAction(easeBackInOut);  
     
//    CCEaseElasticIn  
//    作用：让目标动作赋予弹性 ，且以目标动作起点位子赋予弹性  
//     参数：目标动作  
//    CCActionInterval \* easeElasticIn= CCEaseElasticIn::create(move);  
//    sp->runAction(easeElasticIn);  
     
//     CCEaseElasticOut  
    //    作用：让目标动作赋予弹性 ，且以目标动作终点位子赋予弹性  
    //     参数：目标动作  
//    CCActionInterval \*easeElasticOut = CCEaseElasticOut::create(move);  
//    sp->runAction(easeElasticOut);  
     
//    CCEaseElasticInOut  
    //    作用：让目标动作赋予弹性 ，且以目标动作起点和终点位子赋予弹性  
    //     参数：目标动作  
//    CCActionInterval \*easeElasticInOut = CCEaseElasticOut::create(move);  
//    sp->runAction(easeElasticInOut);

//    CCEaseExponentialIn  
//    让目标动作缓慢开始  
//    参数：目标动作  
//    CCActionInterval \* easeExponentialIn= CCEaseExponentialIn::create(move);  
//    sp->runAction(easeExponentialIn);  
     
    //    CCEaseExponentialOut  
    //    让目标动作缓慢中止  
    //    参数：目标动作  
//    CCActionInterval \* easeExponentialIn= CCEaseExponentialOut::create(move);  
//    sp->runAction(easeExponentialIn);  
     
//    CCEaseExponentialInOut  
    //    让目标动作缓慢开始和中止  
    //    参数：目标动作  
//    CCActionInterval \* easeExponentialInOut= CCEaseExponentialInOut::create(move);  
//    sp->runAction(easeExponentialInOut);  
     
//    CCEaseRateAction  
//    作用 ： 让目标动作设置速率  
//    参数1:目标动作  
//    参数2：速率  
//    CCActionInterval \* move = CCMoveTo::create(5, ccp(300,sp->getPositionY()));  
//    CCActionInterval \* easeRateAction = CCEaseRateAction::create(move, 3);  
//    sp->runAction(easeRateAction);  
     
//    CCEaseSineIn  
//    作用：动作由慢到快  
//      参数：目标动作  
//    CCActionInterval \* easeSineIn = CCEaseSineIn::create(move);  
//    sp->runAction(easeSineIn);  
     
    //    CCEaseSineOut  
    //    作用：动作由快到慢  
    //      参数：目标动作  
//    CCActionInterval \* easeSineOut = CCEaseSineOut::create(move);  
//    sp->runAction(easeSineOut);  
     
    //    CCEaseSineInOut  
    //    作用：动作由慢到快再快到慢  
    //      参数：目标动作  
//    CCActionInterval \* easeSineInOut = CCEaseSineInOut::create(move);  
//    sp->runAction(easeSineInOut);  
     
//    CCSpeed  
//    作用：让目标动作运行速度加倍  
//    参数1：目标动作  
//    参数2:倍速  
//        CCActionInterval \* move = CCMoveTo::create(10, ccp(300,sp->getPositionY()));  
//    CCSpeed \* speed =CCSpeed::create(move, 100);  
//    sp->runAction(speed);  
     
//    CCSpawn  
//  作用：让多个动作同时执行  
//    参数：目标动作的可变参数  
//    CCActionInterval \* move = CCMoveTo::create(10, ccp(300,sp->getPositionY()));  
//    CCActionInterval \* scale = CCScaleTo::create(2, 3);  
//    CCActionInterval \* rotate = CCRotateTo::create(4, 190);  
//    CCFiniteTimeAction \* spawn =CCSpawn::create(move,scale,rotate,NULL);  
//    sp->runAction(spawn);  
     
//    CCSequence  
    //    作用：让多个动作按照前后顺序逐一执行  
    //    参数：目标动作的可变参数  
//    CCActionInterval \* move = CCMoveTo::create(2, ccp(300, sp->getPositionY()));  
//    CCActionInterval \* scale = CCScaleTo::create(2, 3);  
//    CCFiniteTimeAction \* seq= CCSequence::create(move,scale,NULL);  
//    sp->runAction(seq);  
    //    扩展如果要对目标动作全部进行方向运动，可以使用如下形式操作  
    //    CCFiniteTimeAction \*seq=CCSequence::create(moveby,scaleby,...NULL);  
    //    CCFiniteTimeAction \* reverseseq = CCSequence::create(seq,seq->reverse(),NULL)  
    //    注意CCSequence中的所有动作都必须支持reverse函数，否则会出现异常  
//    CCActionInterval \* move = CCMoveBy::create(2, ccp(300, sp->getPositionY()));  
//    CCActionInterval \* scale = CCScaleBy::create(2, 3);  
//    CCFiniteTimeAction \* seq= CCSequence::create(move,scale,NULL);  
//    CCFiniteTimeAction \* reveseseq = CCSequence::create(seq,seq->reverse(),NULL);  
//    sp->runAction(reveseseq);  
     
//    CCRepeat  
//    作用：对目标动作进行重复运动（目标动作可以是CCSequence ，CCSpawn）  
//    参数1：目标动作  
//    参数2：重复次数  
//    CCActionInterval \* move = CCMoveTo::create(2, ccp(300, sp->getPositionY()));  
//    CCActionInterval \* move2 = CCMoveTo::create(2, ccp(100,100));  
//    CCFiniteTimeAction\*seq =CCSequence::create(move,move2,NULL);  
//    CCFiniteTimeAction \*repeat = CCRepeat::create(seq, 3);  
//    sp->runAction(repeat);  
     
//    CCRepeatForever  
//    作用：对目标动作进行永久性的重复运动（目标动作可以是CCSequence ，CCSpawn）  
//    参数：目标动作  
//    CCActionInterval \* move = CCMoveTo::create(1, ccp(300, sp->getPositionY()));  
//    CCActionInterval \* move1 = CCMoveTo::create(1, ccp(100,100));  
//    CCFiniteTimeAction\* seq = CCSequence::create(move,move1,NULL);  
//    CCActionInterval \* repeatForever =CCRepeatForever::create((CCActionInterval\* )seq);  
//    sp->runAction(repeatForever);  
     
//    CCCallFunc  
//    作用：创建一个回调动作（调用不带参数的回调方法）；  
//    参数1：目标动作  
//    参数2：目标回调函数  
//    CCActionInterval \* move = CCMoveTo::create(1, ccp(300, sp->getPositionY()));  
//    CCCallFunc \* funcall= CCCallFunc::create(this, callfunc\_selector(HelloWorld::callbackC));  
//    CCFiniteTimeAction \* seq = CCSequence::create(move,funcall,NULL);  
//    sp->runAction(seq);  
     
//    CCCallFuncN  
    //    作用：创建一个回调动作（调用 带一个参数的回调方法）；  
    //    参数1：目标动作  
    //    参数2：目标回调函数  
//    CCActionInterval \* move = CCMoveTo::create(1, ccp(300, sp->getPositionY()));  
//    CCCallFuncN \* funcall= CCCallFuncN::create(this, callfuncN\_selector(HelloWorld::callbackN));  
//    CCFiniteTimeAction \* seq = CCSequence::create(move,funcall,NULL);  
//    sp->runAction(seq);  
     
//    CCCallFuncND  
    //    作用：创建一个回调动作（调用 带两个参数的回调方法）；  
    //    参数1：目标动作  
    //    参数2：目标回调函数

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

3.组合动作

组合动作包括以下几类：顺序、并列、有限次数重复、无限次数重复、反动作和动画

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

注：（1）不是所有的动作都有反动作，一般来说XxxTo无反动作，XxxBy有反动作//反动作：FiniteTimeAction\* ac1 = (FiniteTimeAction\*)MoveBy::create(2, Vec2(200, 0));

//auto ac2=ac1->reverse();

（2）几个常用的组合动作类：

①Sequence//顺序执行动作，派生于ActionInterval

sprite->runAction(sequence::create(ac0,ac1,NULL));//类似于menu的实现

其中ac0是FiniteTimeAction类，需要进行强制类型转换例如

auto\* ac2 = (FiniteTimeAction\*)sprite->runAction(RotateTo::create(2,40));

//上述的sprite是这一个场景类的一个成员变量

②Spawn//实现并列动作，派生于ActionInterval

sprite->runAction(Spawn::create(ac1,ac2,NULL));

③Repeat//动作的重复使用，派生于ActionInterval，

ActionInterval \*seq=Sequence::create(ac1,ac2,ac3,NULL);

sprite->runAction(Repeat::create(seq,3));//第二个参数代表重复3次

④RepeatForever//动作的永久重复

ActionInterval \*seq =Sequence::create(ac3,ac4,NULL);

sprite->runAction(RepeatForever::create(seq));

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*4.动作速度控制(详见github)

ActionEase和其子类Speed能够实现精灵的非匀速或非线性速度运动，其中

ActionEase类是ActionInterval的子类

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

常用的速度控制函数：

PS：首先建立一些动作

FiniteTimeAction\* ac1 = (FiniteTimeAction\*)MoveBy::create(2, Vec2(200, 0));

FiniteTimeAction\* ac2 = ((FiniteTimeAction\*)ac1)->reverse();

ActionInterval\* ac = Sequence::create(ac1, ac2, NULL);

建立动作ac

1. sprite->runAction(EaseIn::create(ac,n))//以n倍速由慢至快
2. sprite->runAction(EaseOut::create(ac,n))//以n倍速由快至慢
3. sprite->runAction(EaseInout::create(ac,n))//以n倍速由慢至快再由快至慢
4. sprite->runAction(EaseSineIn::create(ac))//采用正弦变换速度由慢至快
5. sprite->runAction(EaseSineOut::create(ac))//采用正弦变换速度由快至慢
6. sprite->runAction(EaseSineInOut::create(ac))//采用正弦变换速度由慢至快再至慢
7. sprite->runAction(EaseExponentialIn::create(ac))//采用指数变换速度由慢至快
8. sprite->runAction(EaseExponentialOut::create(ac)) //采用指数变换速度由快至慢
9. sprite->runAction(EaseExponentialInOut::create(ac))//采用指数变换速度由慢至快再至慢

（10）sprite->runAction(Speed::create(ac,(CCRANDOM\_0\_1()\*5)))//随机设置变换速度

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

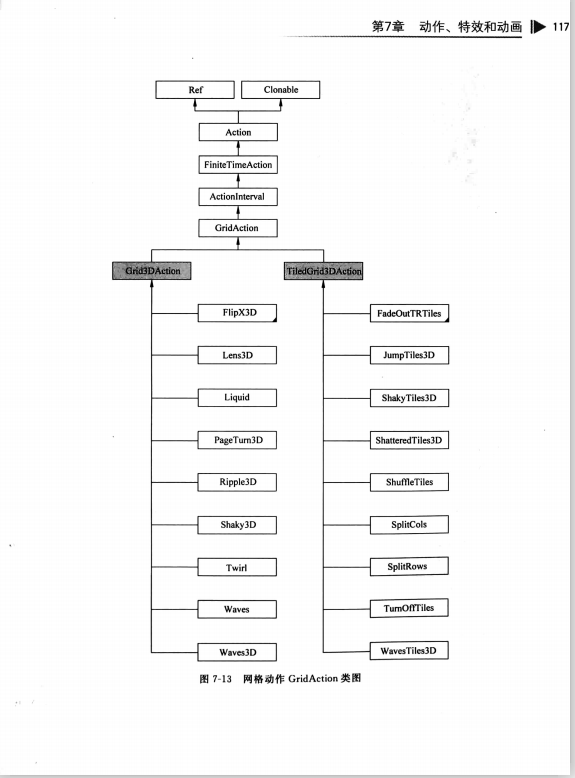
1. 特效

特效事实上属于间隔动作。特效类GridAction类也称为网格动作，

GridAction有两个主要的子类Grid3DAction和TiiledGridAction

注意：特效会给内存和CPU造成巨大的压力和负担，必要时应启用OpenGL

的深度缓冲



/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

创建特效的步骤：

1. 在新的场景类中定义一个特效对象

如：NodeGrid \* gridNodeTarget;//在后续的操作中将会利用它实现特效，

同时可以创建一个精灵sprite实现动作

同时在.cpp中的init()函数中定义并添加这个特效

如：gridNodeTarget=NodeGrid::create();

addChild(gridNodeTarget);

1. 处理一些细节，如向层中添加背景图片或精灵时，

gridNodeTarget->addChild(background);

gridNodeTarget->addChild(sprite);//将特效添加到相应的背景与精灵上

若不添加特效，直接this->addChlid()即可

1. 特效的实现

//    常见的22种特效

//    CCShaky3D  
//作用：创建一个3d晃动的特效  
//    参数1：晃动的范围  
//    参数2：是否有z轴晃动  
//    参数3：  网格大小  
//    参数4：特效持续的时间  
//    CCActionInterval \* shaky3D= CCShaky3D::create(15, false, ccg(15, 10), 4);  
//    sp->runAction(shaky3D);  
      
//    CCShakyTiles3D  
//    作用：创建一个3d瓷砖晃动的特效  
    //    参数1：晃动的范围  
    //    参数2：是否有z轴晃动  
    //    参数3：  网格大小  
    //    参数4：特效持续的时间  
//    CCActionInterval \* shakyTiles3D = CCShakyTiles3D::create(15, true, ccg(15, 10), 4);  
//    sp->runAction(shakyTiles3D);  
      
//    CCWaves  
//    创建一个波动特效  
//    参数1：波动的速率  
//    参数2：振幅  
//    参数3:水平波动  
//    参数4：垂直波动  
//    参数5：网格大小  
//    参数6：特效持续时间  
//    CCActionInterval \* waves = CCWaves::create(10, 20, true, true, ccg(16, 12), 4);  
//    sp->runAction(waves);  
      
//    CCWaves3D  
    //    创建一个3D波动特效  
    //    参数1：波动的速率  
    //    参数2：振幅  
    //    参数3：网格大小  
    //    参数4：特效持续时间  
//    CCActionInterval \* waves3d= CCWaves3D::create(10, 20, ccg(16, 12), 4);  
//    sp->runAction(waves3d);  
      
//    CCWavesTiles3D  
    //    创建一个3D瓷砖波动特效  
    //    参数1：波动的速率  
    //    参数2：振幅  
    //    参数3：网格大小  
    //    参数4：特效持续时间  
//    CCActionInterval \* wavesTiles3D = CCWavesTiles3D::create(10, 20, ccg(16, 12), 4);  
//    sp->runAction(wavesTiles3D);  
      
//    CCFlipX3D  
//    作用：x轴3D反转特效  
//    参数：特效持续的时间  
//    CCActionInterval \*flipX3D= CCFlipX3D::create(4);  
//    sp->runAction(flipX3D);  
      
//    CCFlipY3D  
    //    作用：Y轴3D反转特效  
    //    参数：特效持续的时间  
//    CCActionInterval \* flipY3D = CCFlipY3D::create(4);  
//    sp->runAction(flipY3D);  
      
//    CCLens3D  
//    作用：  凸镜特效  
//    参数1：凸镜中心点  
//    参数2：凸镜半径  
//    参数3：网格大小  
//    参数4：网格持续时间  
//    CCActionInterval \* lens3D= CCLens3D::create(CCPointMake(size.width/2, size.height/2), 240, ccg(15, 10), 4);  
//    sp->runAction(lens3D);  
  
  
//    CCRipple3D  
//    作用：水波特效  
//    参数1：起始位子  
//    参数2：半径  
//    参数3：速率  
//    参数4：振幅  
//    参数5：网格大小  
//    参数6：特效持续的时间  
//    CCActionInterval \* ripple3D= CCRipple3D::create(CCPointMake(size.width/2, size.height/2), 240, 4, 160, ccg(15,10), 4);  
//    sp->runAction(ripple3D);  
      
//    CCLiquid  
//    作用：液体特效  
//    参数1：速率  
//    参数2：振幅  
//    参数3：网格大小  
//    参数4：特效持续时间  
//    CCActionInterval \* liquid = CCLiquid::create(4, 20, ccg(16, 12), 4);  
//    sp->runAction(liquid);  
      
//    CCTwirl  
//    作用：扭曲旋转特效  
//    参数1：扭曲旋转中心点  
//    参数2：扭曲旋转的次数  
//    参数3：振幅  
//    参数4：网格大小  
//    参数5：特效持续时间  
//    CCActionInterval  \* twirl = CCTwirl::create(CCPointMake(size.width/2, size.height/2), 2, 2.5f, ccg(12, 8), 3);  
//    sp->runAction(twirl);  
      
//    CCShuffleTiles  
//    作用：瓷砖的洗牌特效  
//    参数1：随机数  
//    参数2：网格大小  
//    参数3：特效持续的时间  
//    CCActionInterval \*shuffleTiles = CCShuffleTiles::create(16, ccg(16, 12), 2);  
//    sp->runAction(shuffleTiles);  
      
//    CCShatteredTiles3D  
//    作用：破碎的3D瓷砖特效  
//    参数1：范围  
//    参数2：是否开启z轴  
//    参数3：网格大小  
//    参数4：特效持续的时间  
//    CCActionInterval \* shatteredTiles3D= CCShatteredTiles3D::create(25, true, ccg(16, 12), 4);  
//    sp->runAction(shatteredTiles3D);  
  
  
//    CCFadeOutTRTiles  
//    作用：部落格效果 ， 从左下角到右下角  
//    参数1：网格大小  
//    参数2：特效持续的时间  
//    CCActionInterval \* fadeOutTRTiles = CCFadeOutTRTiles::create(ccg(16, 12), 3);  
//    sp->runAction(fadeOutTRTiles);  
  
  
//    CCFadeOutBLTiles  
    //    作用：部落格效果 ， 从右下角到左下角  
    //    参数1：网格大小  
    //    参数2：特效持续的时间  
//    CCActionInterval \* fadeOutBLTiles = CCFadeOutBLTiles::create(ccg(16, 12), 3);  
//    sp->runAction(fadeOutBLTiles);  
  
  
//    CCFadeOutUpTiles  
//    作用：折叠效果 从下到上  
    //    参数1：网格大小  
    //    参数2：特效持续的时间  
//    CCActionInterval \* fadeOutUpTiles = CCFadeOutUpTiles::create(ccg(16, 12), 3);  
//    sp->runAction(fadeOutUpTiles);  
      
//    CCFadeOutDownTiles  
    //    作用：折叠效果 从上到下  
    //    参数1：网格大小  
    //    参数2：特效持续的时间  
//    CCActionInterval \* fadeOutDownTiles = CCFadeOutDownTiles::create(ccg(16, 12), 3);  
//    sp->runAction(fadeOutDownTiles);  
    
//    CCTurnOffTiles  
//    作用：方块消失特效  
    //    参数1：网格大小  
    //    参数2：特效持续的时间  
//    CCActionInterval \* turnOffTiles = CCTurnOffTiles::create(ccg(16, 12), 3);  
//    sp->runAction(turnOffTiles);  
  
  
//    CCJumpTiles3D  
//    作用：跳动的方格特效  
//    参数1：跳动的次数  
//    参数2：网格的大小  
//    参数3：特效持续的时间  
//    CCActionInterval \* jumpTiles3D = CCJumpTiles3D::create(1, 30, ccg(15, 10), 2);  
//    sp->runAction(jumpTiles3D);  
      
//    CCSplitRows  
//    作用：分多行消失特效  
//    参数1：行数  
//    参数2：特效持续的时间  
//    CCActionInterval \*splitRows= CCSplitRows::create(20, 2);  
//    sp->runAction(splitRows);  
  
  
//    CCSplitCols  
    //    作用：分多列消失特效  
    //    参数1：列数  
    //    参数2：特效持续的时间  
//    CCActionInterval \* splitCols = CCSplitCols::create(20, 2);  
//    sp->runAction(splitCols);  
      
//    CCPageTurn3D  
//    作用：3D翻页特效  
//    参数1：网格大小  
//    参数2：特效持续的时间  
    CCActionInterval \* pageTurn3D = CCPageTurn3D::create(ccg(15, 10), 3);  
    sp->runAction(pageTurn3D);

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

1. 动画

动画分为场景过渡动画和帧动画，场景过渡动画以精彩之前提到过

帧动画

1. 帧动画相关概念

帧动画涉及两个类：Animation和Animate.前者是动画类，它保存有很多的动画帧，后者是动作类，它继承于ActionInterval类，属于间隔动作类，其作用是将前者定义的动画转换成动作进行执行，从而达到动画播放的效果

2.帧动画的使用（具体代码见github）

<https://www.cnblogs.com/luorende/p/5998893.html>超级详细的介绍

示例代码如下：

//创建一个跑酷的精灵

 auto sprite = Sprite::create("1.png");

//设置精灵的坐标

sprite->setPosition(Vec2(visibleSize.width/2,visibleSize.height/2));

//添加到当前层

this->addChild(sprite);

//创建序列帧动画

auto animation = Animation::create();

//设置动画名字数组的长度

char nameSize[20] = {0};

//动画的循环 12张图片

for (int i =1; i<13; i++)

{

//循环遍历

sprintf(nameSize, "%d.png",i);

 //添加到序列帧动画

  animation->addSpriteFrameWithFile(nameSize);

}

//设置动画帧的时间间隔

animation->setDelayPerUnit(0.02f);

//设置播放循环 一直播放 为-1

animation->setLoops(-1);

//设置动画结束后恢复到第一帧

animation->setRestoreOriginalFrame(true);

//创建动画动作

auto animate = Animate::create(animation);

//播放动画

sprite->runAction(animate);