### 3.2 Cocos2d-x中的粒子系统

**【本节主要知识点】**

* **粒子系统中常用的概念**
* **如何在Cocos2d-x中使用粒子系统**
* **在Cocos2d-x中内置了哪些粒子效果，如何使用**
* **如何使用粒子编辑器**

**（1）什么是粒子系统**

前面的动画（如：人物行走），使用的是帧动画的形式。这样的方法会产生两个问题：

第1：帧动画需要大量的图片素材，这会给系统的存储和渲染带来极大的负担，尤其是在手机游戏中，系统的性能本来就有限，如果再游戏中大量的使用帧动画，显然不是一个好的方案。

第2：一些效果是要多次重复出现的，比如植物随风抖动或者湖面的波纹。这些通常会采用类似瓦片地图的方式将相同的动画素材拼接在一起。这样不免被玩家发现许多重复的地方，影响游戏的整体美感。

粒子系统通过无数细小的粒子进行相对随机的有规律的运动，来实现复杂的动画效果。

粒子效果的原理：通过对发射出的粒子的速度、颜色还有形状等属性进行控制，开模拟自然界中的火焰、暴雪、水流、云雾等特效。

在粒子系统中，虽然粒子整体有固定的方向，但对于单个粒子来说，它们的行为是随机的，因此从宏观上看去，基本上不会出现重复的样子。所以，粒子系统在游戏中使用会有很好的效果。

除此之外，由于在粒子系统中的单位都是一个个微粒，因此需要的素材资源都比较小，相对于帧动画来说节约了大量的资源。

在一款游戏引擎中，粒子系统通常包含以下特点：

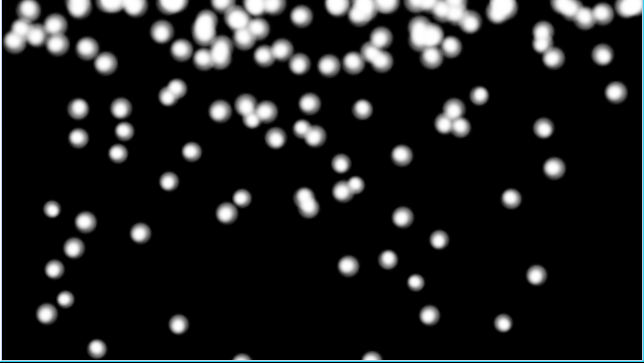
* 粒子性：在渲染时包含了大量的粒子对象
* 宏观性：每一个粒子都符合一定的物理规律，比如受重力或是向着某个方向移动等。
* 微观性：虽然每一个粒子都要符合共同的规律比如向着某个方向运动，但是却会有运动的先后以及相对的快慢等差异，每一个粒子都有自己的独特性和随机性。
* 过程动态性：每一个粒子单位都是动态的，在粒子运动的过程中是不断变化的。

一套完整的粒子系统应当包含了3个部分：

* 粒子本身
* 粒子发射器。粒子系统的控制器
* 粒子系统形成的动态效果。用来控制粒子的生成，以及赋予它们不同属性的部分。

**（2）粒子系统的使用**

使用粒子系统实现雪花飘落效果的粒子。运行程序后，可以看到有雪花状的粒子不断落下而每一个雪花，虽然都在下落，但它们下落的速度与方向又都有着某些随机性，就像在自然界中的雪花落下一样。



**【代码解释】**

**m\_emitter是一个ParticleSystemQuad类对象。ParticleSystemQuad类继承自类PraticleSystem，是一种粒子系统类型。**

**m\_emitter->initWithTotalParticles(900);//900个粒子对象**

**对粒子系统进行初始化，其中的参数定义了场景中可以存在的粒子的数量。一般来说，一个好的粒子系统都应当能够用尽量少的粒子来实现相同的效果。这样有利于提升系统的效率。**

**m\_emitter->setTexture(TextureCache::sharedTextureCache()->addImage("snow.png"));**

**用于选择粒子所使用的图像。**

**m\_emitter->setDuration(-1);**

**粒子系统持续的实现。但参数为-1时，表示粒子系统持续执行。**

**m\_emitter->setGravity(Point(0, -40));**

**设置重心方向。**

**m\_emitter->setPosition(Point(640,360));**

**m\_emitter->setPosVar(Point(640, 0));**

**定义粒子发射器的位置；粒子发射器的范围（上面表示横向640范围）**

**m\_emitter->setLife(4);**

**m\_emitter->setLifeVar(2);**

**定义雪花的生命周期。定义粒子被产生多久后销毁。**

**在学习过程中，我们可以通过自己去修改各个方法中的参数，体会个方法的功能和参数的作用。**

【代码】

auto \*m\_emitter = new ParticleSystemQuad();

m\_emitter->initWithTotalParticles(900);//900个粒子对象

m\_emitter->setTexture(TextureCache::sharedTextureCache()->addImage("snow.png"));

//设置发射粒子的持续时间-1表示一直发射，0没有意义，其他值表示持续时间

m\_emitter->setDuration(-1);

//设置重心方向,这个店是相对发射点，x正方向为右，y正方向为上

m\_emitter->setGravity(Point(0, -40));

//设置角度，角度的变化率

m\_emitter->setAngle(0);

m\_emitter->setAngleVar(360);

//设置径向加速度，径向加速度的变化率

m\_emitter->setRadialAccel(50);

m\_emitter->setRadialAccelVar(0);

//设置粒子的切向加速度，切向加速度的变化率

m\_emitter->setTangentialAccel(30);

m\_emitter->setTangentialAccelVar(0);

//设置粒子的位置，位置的变化率

m\_emitter->setPosition(Point(640,360));

m\_emitter->setPosVar(Point(640, 0));

//设置粒子声明，生命的变化率

m\_emitter->setLife(4);

m\_emitter->setLifeVar(2);

//设置粒子开始的自旋转速度，开始自旋转速度的变化率

m\_emitter->setStartSpin(90);

m\_emitter->setStartSpinVar(120);

//设置结束的时候的自旋转以及自旋转的变化率

m\_emitter->setEndSpin(90);

m\_emitter->setEndSpinVar(160);

ccColor4F cc;

cc.a = 1.0f;

cc.b = 255.0f;

cc.g = 255.0f;

cc.r = 255.0f;

ccColor4F cc2;

cc2.a = 0;

cc2.b = 0;

cc2.g = 0;

cc2.r = 0;

//设置开始的时候的颜色以及颜色的变化率

m\_emitter->setStartColor(cc);

m\_emitter->setStartColorVar(cc2);

//设置结束的时候的颜色以及颜色的变化率

m\_emitter->setEndColor(cc);

m\_emitter->setEndColorVar(cc2);

//设置开始时候粒子的大小，以及大小的变化率

m\_emitter->setStartSize(30);

m\_emitter->setStartSizeVar(0);

//设置粒子结束的时候的大小，以及大小的变化率

m\_emitter->setEndSize(20.0f);

m\_emitter->setEndSizeVar(0);

//设置每秒钟产生粒子的数目

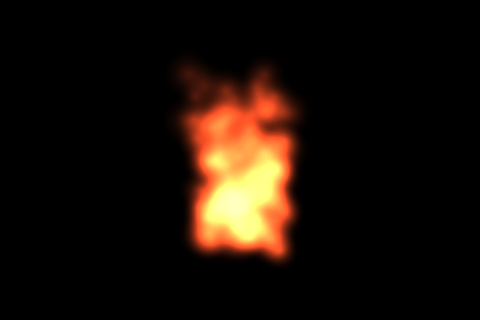
m\_emitter->setEmissionRate(100);

addChild(m\_emitter);

可将代码中的图片snow.png修改为snow.png1，但是实际运行程序之后，感觉更像是星星而不是雪花。但是六边形的图案能够更加方便地观察到该粒子系统的某些特性，比如可以在场景中看到雪花自身的旋转效果。

**（3）使用粒子系统创建火焰**

Cocos2d-x中提供了一些现成的类，来帮助开发者方便地实现某些常用效果。比如下面的示例，就实现了火焰的效果。



**【代码说明】**

**除了创建效果之外，还可以修改火焰的效果，比如下面代码中注释的内容**

【代码】

auto\* fire = ParticleFire::create();

fire->setPosition(320, 100);

addChild(fire);

//fire->setAngle(45);

//fire->setEndSize(200);

return true;

在Cocos2d-x中，除了火焰，还提供了一些其他的类来生成相应的效果，这些类的使用方法与上面代码相似。

表：Cocos2d-x中可以直接使用的粒子相关类

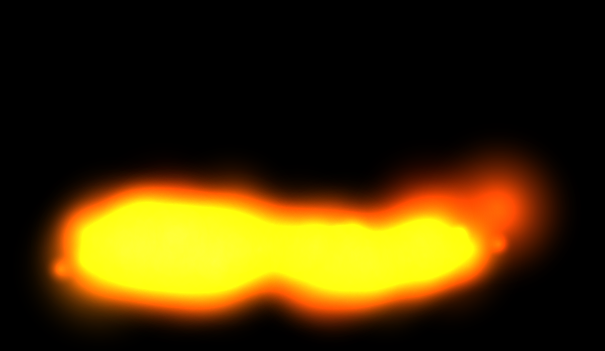
|  |  |
| --- | --- |
| 类名 | 作用 |
| ParticleExplosion | 爆炸粒子效果 |
| ParticleFire | 火焰粒子效果 |
| ParticleFireworks | 烟花粒子效果 |
| ParticleFlower | 花粒子效果 |
| ParticleGalaxy | 星系粒子效果 |
| ParticleMeteor | 流行粒子效果 |
| ParticleSpiral | 旋涡粒子效果 |
| ParticleSnow | 雪粒子效果 |
| Particlemoke | 烟粒子效果 |
| ParticleSun | 太阳粒子效果 |
| ParticleRain | 雨粒子效果 |

**（4）使用粒子编辑器**

通过上一节的学习，我们已经可以实现一些特殊效果了，比如火焰或太阳。但是在实际的开发中，仅仅会这几个效果还是不够的。就拿火焰来说，ParticleFire类可以生成一簇火焰，但是实际使用时，可能需要的是星火燎原的效果。

想通过修改各种参数来实现想要的效果，难度无疑十分巨大，那么久需要借助某些工具：如粒子编辑器。--ParticleEditor。

在Cocos2d-x中使用粒子编辑器生成的粒子效果如下图：



auto\* fire = ParticleSystemQuad::create("fire.plist");

fire->setPosition(250, 100);

fire->retain();

addChild(fire);

**（5）实验3-4 使用粒子系统实现下雪效果**

**（6）实验3-5 使用粒子系统创建火焰**