Cocos2dx 游戏开发教程

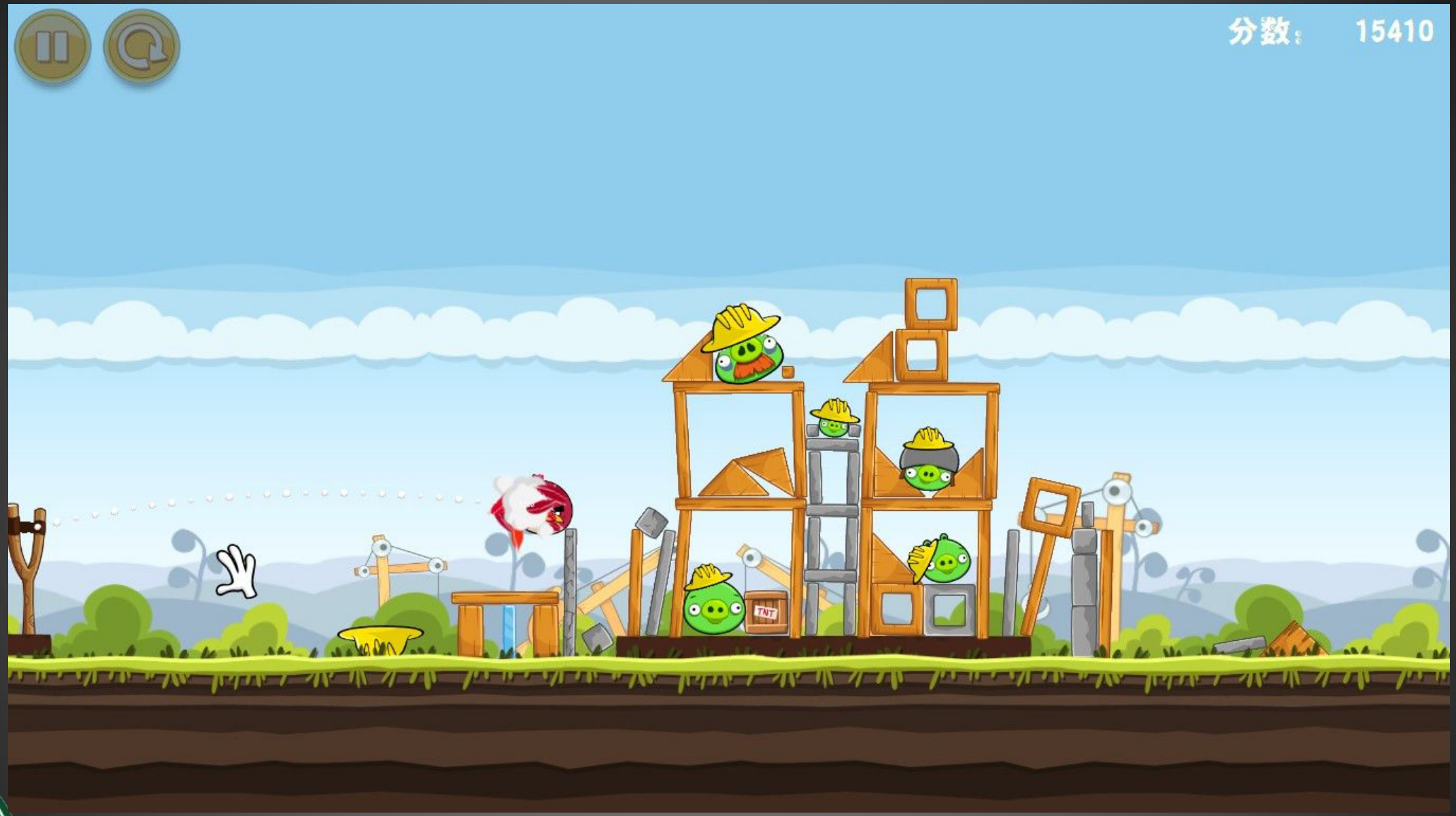
by 陈元仿

CONTENTS

CONTENTS

section1

Physics Engine，物理引擎


分数：
15410
TNT


中山大學
SUN YAT-SEN UNIVERSITY


COCOS2DX

蜡笔物理学


DRAW
A BOX
USE ESC TO ACCESS MENU


中山大學
SUN YAT-SEN UNIVERSITY


COCOS2DX

重力世界

基础-游戏物理引擎

物理引擎使用对象属性（动量、扭矩或者弹性）来模拟刚体行为，这不仅可以得到更加真实的结果，对于开发人员来说也比编写行为脚本要更加容易掌握。

好的物理引擎允许有复杂的机械装置，像球形关节、轮子、气缸或者铰链。有些也支持非刚性体的物理属性，比如流体。

基础-Physics integration

在Cocs2d-x 3.0 中有了改变，全新的Physics integration，把chipmunk和Box2D封装到引擎内部，游戏开发不用关心底层具体是用的哪个物理引擎，不用直接调用物理引擎的接口。

物理引擎和Cocos2d-x进行了深度融合：

物理世界被融入到Scene中，即当创建一个场景时，就可以指定这个场景是否使用物理引擎。

Node自带body属性，也就是sprite自带body属性。

Cocos2d-x 3.0对物理引擎的Body(PhysicsBody)，Shape(PhysicsShape)，Contact(PhysicsContact)，Joint(PhysicsJoint)，World(PhysicsWorld)进行了封装抽象，使用更简单。

基础-物理引擎重要元素

1、Body实体是物理引擎中可相互交互的基本对象。它包含了质量，质心位置和密度等属性。开发人员可以应用一个力，力矩或者脉冲到这个实体上。实体可以有静态的和动态的。

2、Shape形状定义了一个Body所应该拥有的外形特征，可以使用多边形，矩形或者圆形之中的任意一个。形状同时也用来定一个实体的质量。这一功能让开发人员能够通过给定密度，物理引擎将会计算出形状的面积从而求解出实体的质量属性。同时，形状还拥有摩擦力等属性。形状还可以带有碰撞过滤信息，能够防止某些特定的游戏物体发生碰撞。

3、Joints关节是两个刚体的链接部位，它可以影响和约束不同刚体之间的行为和运动方式。在游戏中的典型应用有滑轮，链条和合页等。通过各种巧妙的不同组合，各种关节能够拼凑出很多有趣的运动方式。

4、Contact碰撞是指物理世界中两个物体相互接触

基础-Cocos2d-x中使用物理引擎

- 第一步：创建一个物理世界

```
Scene* scene=Scene::createWithPhysics();
```

- 第二步：添加物理物体，设置属性包括刚体等信息

```
m_sprite->setPhysicsBody(
        PhysicsBody::createBox(m_sprite->getContentSize()));
```

基础-Cocos2d-x中使用物理引擎

- 第三步 设置监听事件

```
auto listener = EventListenerPhysicsContact::create();
listener->onContactBegin = CC_CALLBACK_1(GameScene::onConcactBegan, this);
listener->onContactPreSolve = CC_CALLBACK_1(GameScene::onConcactPreSolve, this);
listener->onContactPostSolve = CC_CALLBACK_1(GameScene::onConcactPostSolve, this);
Director::getInstance()->getEventDispatcher()->addEventListenerWithFixedPriority(listener, 1);
```

进阶-关节的使用

Joints关节是两个刚体的链接部位，它可以影响和约束不同刚体之间的行为和运动方式。在游戏中的典型应用有滑轮，链条和合页等。通过各种巧妙的不同组合，各种关节能够拼凑出很多有趣的运动方式。

Cocos2d内嵌物理引擎中提供的关节有

PhysicsJointFixed

PhysicsJointLimit

PhysicsJointPin

PhysicsJointDistance

PhysicsJointSpring

PhysicsJointGroove

PhysicsJointRotarySpring

PhysicsJointRotaryLimit

PhysicsJointRatchet

PhysicsJointGear

PhysicsJointMotor

Cocos2d-x中使用关节

第一步：创建关节

```
PhysicsJointSpring* springJoint = PhysicsJointSpring::construct(
    m_basketball->getPhysicsBody(), m_soccer->getPhysicsBody(),
    m_basketball->getAnchorPoint(), m_soccer->getAnchorPoint(),
    200, 0);
```

第二步：将关节添加到物理世界

```
m_world->addJoint(springJoint);
```

进阶-碰撞过滤

物理引擎中刚体的碰撞检测是一个复杂的过程，需要耗费大量资源。PhysicsBody中有位_categoryBitmask, _collisionBitmask, contactTestBitmask, _group四个属性。通过合理地设置这四个属性值，可以过滤不必要的碰撞检测，减小开销。

```
int     _categoryBitmask;
int     _collisionBitmask;
int     _contactTestBitmask;
int     _group;
```

进阶-碰撞过滤

如下是cocos2d-x中判断是否进行碰撞检测的源码

```
// bitmask check
if ((shapeA->getCategoryBitmask() & shapeB->getContactTestBitmask()) == 0
    || (shapeA->getContactTestBitmask() & shapeB->getCategoryBitmask()) == 0)
{
    contact.setNotificationEnable(false);
}

if (shapeA->getGroup() != 0 && shapeA->getGroup() == shapeB->getGroup())
{
    ret = shapeA->getGroup() > 0;
}
else
{
    if ((shapeA->getCategoryBitmask() & shapeB->getCollisionBitmask()) == 0
        || (shapeB->getCategoryBitmask() & shapeA->getCollisionBitmask()) == 0)
    {
        ret = false;
    }
}
```

进阶-碰撞过滤

- categoryBitmask 类别标示掩码，用来标记我是谁
- contactTestBitmask 接触测试掩码
- collisionBitmask 碰撞掩码
- group 组别

进阶-碰撞过滤

- categoryBitmask标明物体的类别，标明我是谁。碰撞与接触检测需要建立在双方都对彼此“感兴趣”的基础上，即双方的类别掩码作与运算不为0
- contactTestBitmask标明该物体对和谁的接触检测感兴趣，若双方接触掩码相与不为0，则系统会传达接触信息
- collisionBitmask 标明该物体对和谁的碰撞感兴趣，若双方碰撞掩码相与不为0，则会产生碰撞效果
- group标明物体的组别，group的优先级比collisionBitmask高。当两个对象的group相等时，如果都为正，则必然碰撞，如果都为负，则必然不碰撞。若不相等，再根据collisionBitmask处理

进阶–迭代速率

- scene->getPhysicsWorld()->setSubsteps(10);
- scene->getPhysicsWorld()->setUpdateRate(10);

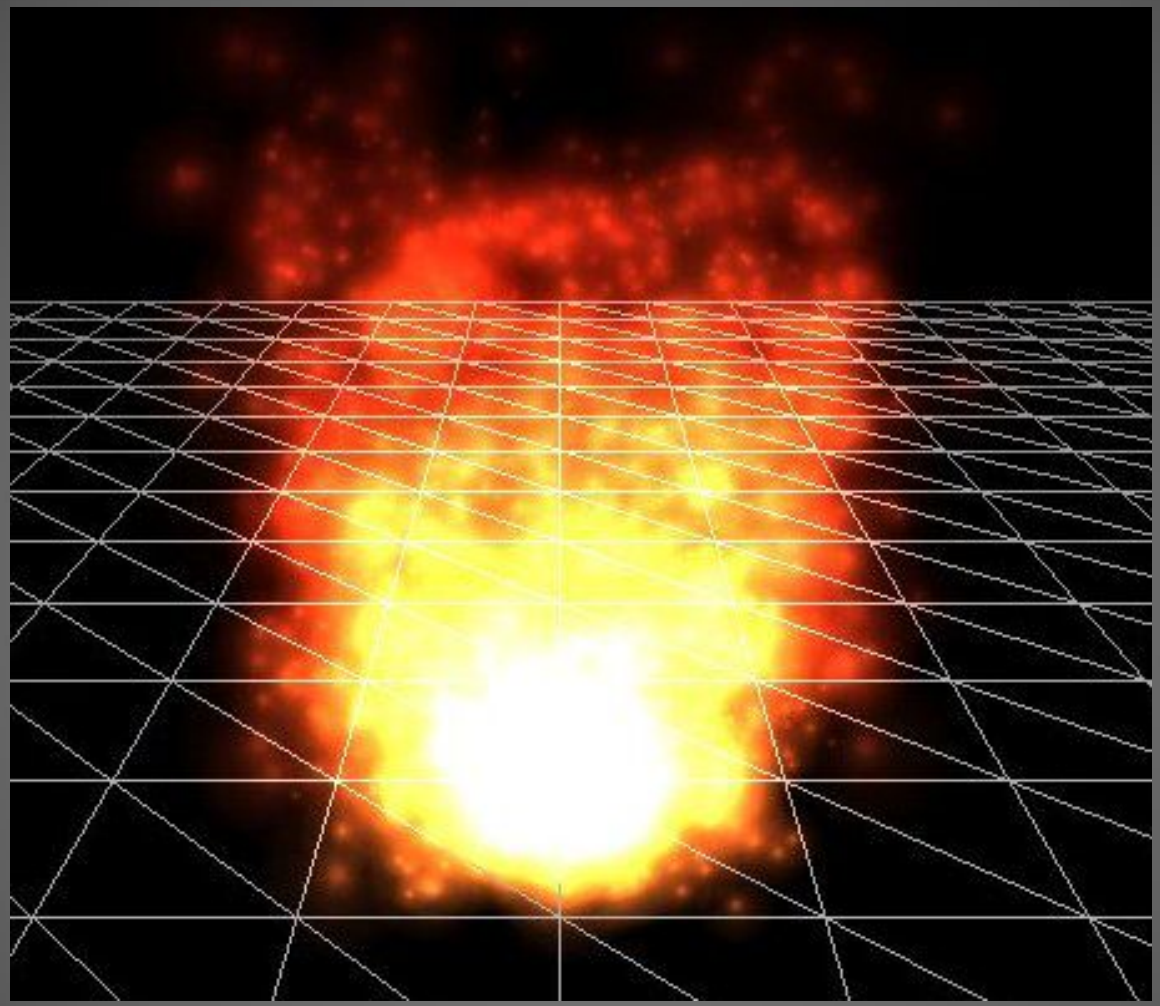
```
else
{
    _updateTime += delta;
    if (++_updateRateCount >= _updateRate)
    {
        const float dt = _updateTime * _speed / _substeps;
        for (int i = 0; i < _substeps; ++i)
        {
            cpSpaceStep(_cpSpace, dt);
            for (auto& body : _bodies)
            {
                body->update(dt);
            }
        }
        _updateRateCount = 0;
        _updateTime = 0.0f;
    }
}
```

进阶-刚体设置

CONTENTS

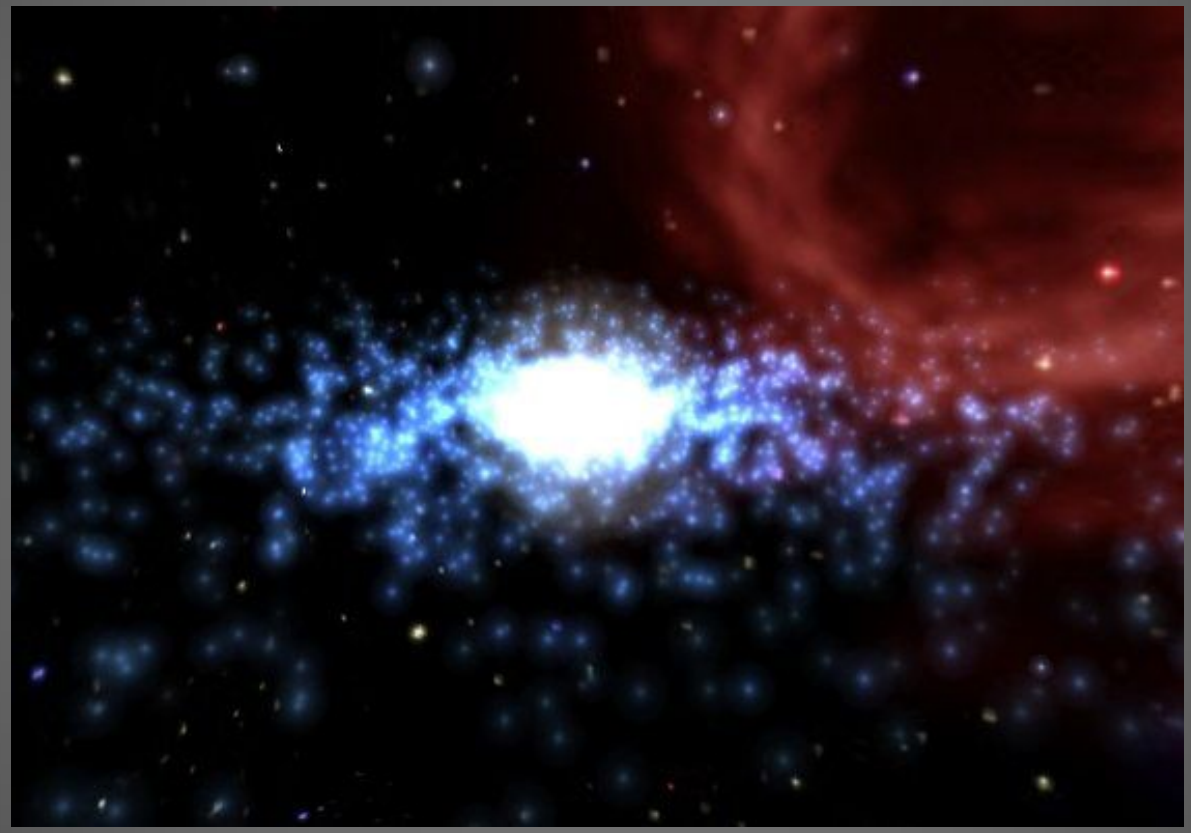
section2

Particle System，粒子系统




中山大學
SUN YAT-SEN UNIVERSITY


COCOS2DX




中山大學
SUN YAT-SEN UNIVERSITY


COCOS2DX

粒子系统表示三维计算机图形学中模拟一些特定的模糊现象的技术，而这些现象用其它传统的渲染技术难以实现的真实感的游戏图形。经常使用粒子系统模拟的现象有火、爆炸、烟、水流、火花、落叶、云、雾、雪、尘、流星尾迹或者发光轨迹这样的抽象视觉效果等等。

帧动画也能实现特效效果，但相对于粒子系统缺点：

1. 画面效果弱
2. 细节不自然
3. 修改不方便

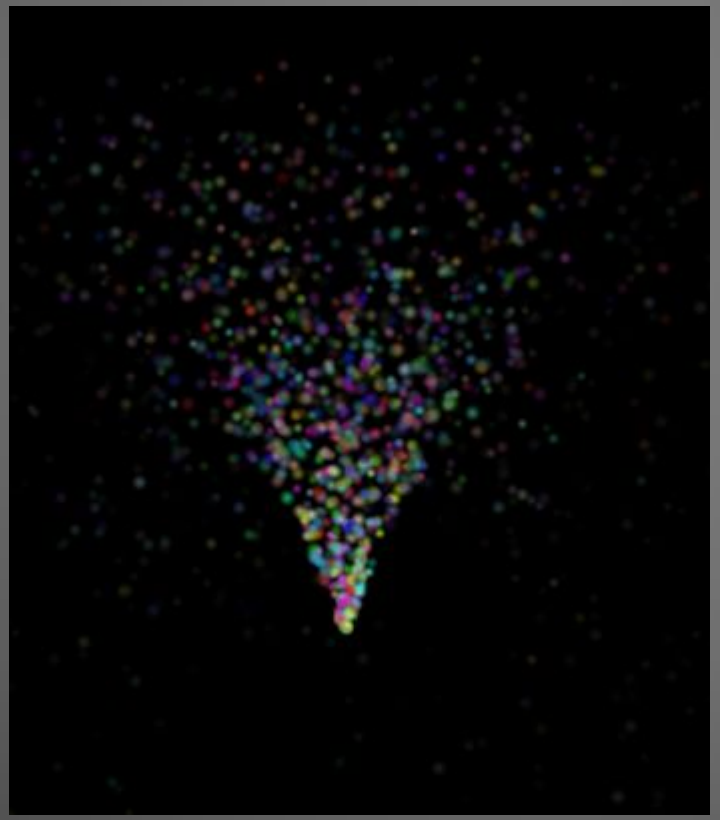
使用cocos2d-x内置粒子系统

cocos2d-x内嵌了如下的11种粒子系统：

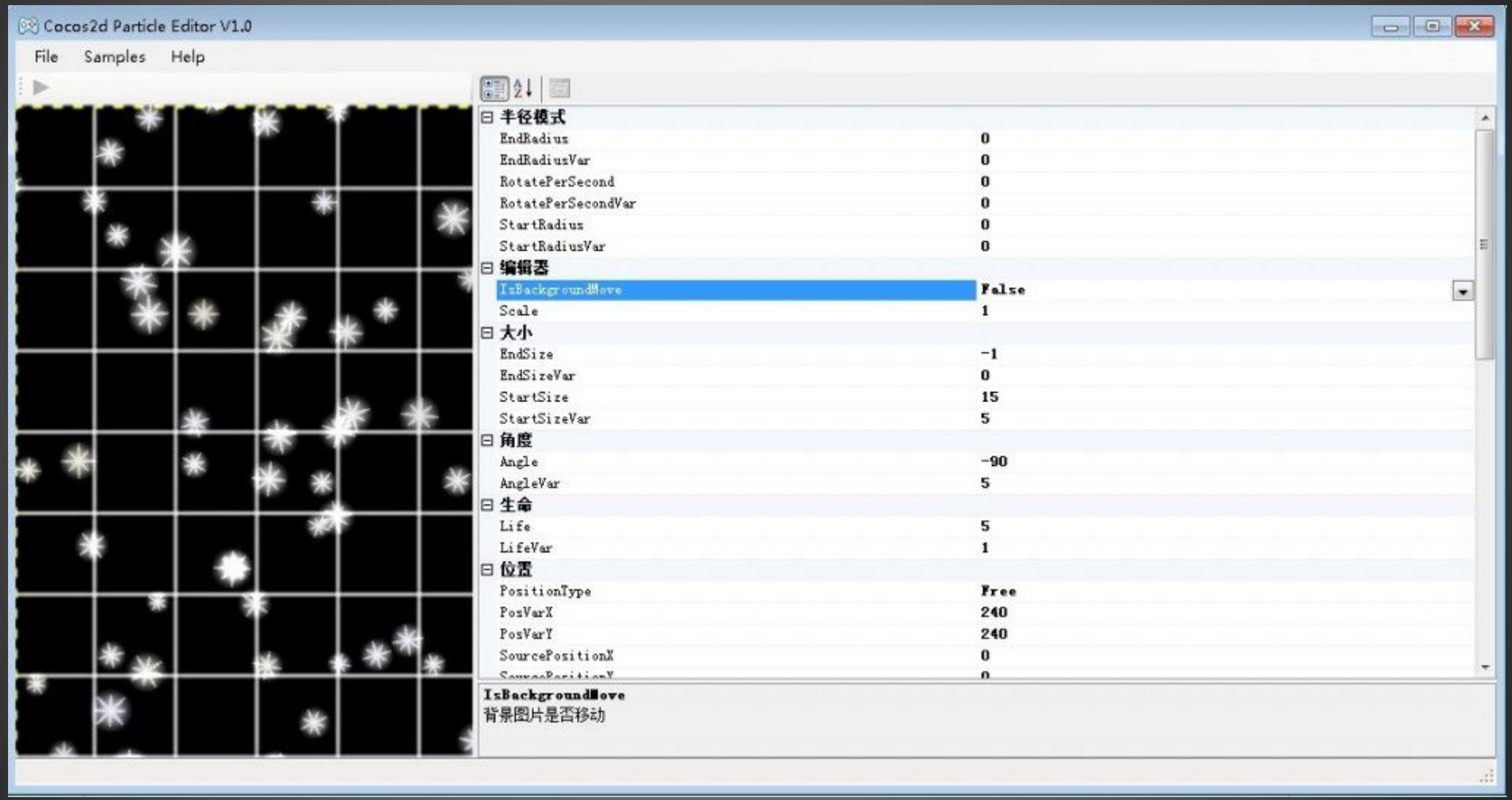
ParticleExplosion。爆炸粒子效果，属于半径模式。
ParticleFire。火焰粒子效果，属于重力径模式。
ParticleFireworks。烟花粒子效果，属于重力模式。
ParticleFlower。花粒子效果，属于重力模式。
ParticleGalaxy。星系粒子效果，属于半径模式。
ParticleMeteor。流星粒子效果，属于重力模式。
ParticleSpiral。漩涡粒子效果，属于半径模式。
ParticleSnow。雪粒子效果，属于重力模式。
ParticleSmoke。烟粒子效果，属于重力模式。
ParticleSun。太阳粒子效果，属于重力模式。
ParticleRain。雨粒子效果，属于重力模式。

使用cocos2d-x内置粒子系统

```
ParticleFireworks* fireworks = ParticleFireworks::create();
fireworks->setPosition(500, 300);
addChild(fireworks);
```



使用cocos2d-x编辑器生成粒子系统



使用cocos2d-x编辑器生成粒子系统

```
auto paopao = ParticleSystemQuad::create("lizhi_qipao.plist");
paopao->setPositionType(ParticleSystemQuad::PositionType::RELATIVE);
paopao->setPosition(500, 300);
paopao->setScale(2.0f);
addChild(paopao);
```

Heomework

完成一个割绳子游戏：割绳子后苹果下落，砸中小猫则小猫消失并在屏幕中央播放烟火粒子系统特效。


My2048


中山大學
SUN YAT-SEN UNIVERSITY


COCOS2DX


My2048


中山大學
SUN YAT-SEN UNIVERSITY


COCOS2DX

Q:如何让小猫，砖块在地板上站立？

A：在地面位置设置一个static的线段刚体（static刚体不受重力影响），
用以支托小猫，砖块

```
Node* ground = Node::create();
ground->setPhysicsBody(PhysicsBody::createEdgeSegment(Vec2(0, 29), Vec2(winSize.width, 29)));
ground->getPhysicsBody()->setDynamic(false);
addChild(ground);
```

Q:如何绳子绕着一个点旋转？

A：绳子旋转通过PhysicsJointPin关节实现，可以定义一个ropeBase刚体（ropeBase是static刚体），通过PhysicsJointPoint将rope与ropeBase连接，然后将旋转点定位ropeBase的位置，则rope就会围绕着ropeBase旋转。

Q:如何让苹果依附在绳子上

A：苹果依附通过PhysicsJointFixed实现，通过Fixed关节，可以将苹果与绳子固定在一起。

Q:割到绳子后如何让苹果掉落

A：移除之前用来绑定苹果与绳子的PhysicsJointFixed即可

Q: 如何判定碰撞的两个物体类型

A：可以通过Node::setTag()方法给不同类型的物体设置tag，然后碰撞后通过getTag（ ）方法判断碰撞的两者是什么物体。

Q:为什么设置完PhysicsListener，碰撞并没有回调设置的函数

A：创建时PhysicsBody的testContactMaskbit默认为0x00000000，即任何与之碰撞的物体都不会回调contact函数。要修改testContactMaskbit的值

Thanks !