精灵

1. 精灵类

直接继承了Node类，具有Node类的全部属性，子类有Skin和PhysicsSprite

1. 创建Sprite精灵对象
2. static Sprite \*create()//常见一个精灵对象，纹理等属性需要在创建后设置
3. static Sprite \*create(const std::string&filename)//指定图片创建精灵
4. static Sprite \*create(const std::string&filename,const Rect &rect)//指定图片和裁剪的矩形区域来创建精灵
5. static Sprite \*createWithTexture(Texture2D\* texture)//指定纹理创建精灵
6. static Sprite \*createWithTexture(Texture2D \*texture,const Rect &rect,bool rotated=false)//指定纹理和裁剪的矩形区域来创建精灵，第三个参数指定是否旋转，默认不旋转
7. static Sprite \*createWithSpriteFrame(SpriteFrame\*pSpriteFrame)

//通过一个精灵帧对象创建另一个对象

1. static Sprite \*createWithSpriteFrameName(const std::string &spriteFrameName)

//通过指定帧缓存中精灵帧名创建精灵对象

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

构造的具体实现

auto knight = Sprite::create("knight.png", Rect(30, 40, 50, 60));

knight->setPosition(Vec2(150, 200));

this->addChild(knight, 0);

通过对图片进行裁剪获得结果，Rect类可以创建举行裁剪区，构造函数如下

Rect(float x,float y,float width,float height)//坐标在左上角

Texture2D\* cache = Director::getInstance()->getTextureCache()->addImage("coolguy.png");

//getTextureCache()获得TaxureCache实例，addImage()创建纹理对象

auto coolguy = Sprite::create();

coolguy->setTexture(cache);//设置纹理属性

coolguy->setTextureRect(Rect(30, 40, 50, 60));//设置裁剪区域

coolguy->setPosition(Vec2(200, 200));

this->addChild(coolguy, 0);

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

1. 精灵的性能优化

精灵的性能优化可以使用精灵表和缓存

1. 使用纹理图集（也称精灵表）

纹理图集将许多小的精灵图片组合到一张大图里面，有以下优点

减少文件读取次数，加速渲染，减少内存消耗

1. 使用帧缓存—纹理缓存，精灵帧缓存，动画缓存

精灵帧缓存，使用的类有SpriteFrame SpriteFrameCache

SpriteFrameCache::getInstance()->addSpriteFramesWithFile(“SpriteSheet.plist”);

auto mountain1 =Sprite::createWithSpriteFrameName(“mountain1.png”);

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

第一行是通过SpriteFrameCache创建精灵帧缓存对象，getInstance()可以获得

SpriteFrameCache单一实例，addSpriteFramesWithFile函数是将精灵帧添加到缓存中，其中SpriteSheet.plist是坐标文件第二行是通过Sprite的createWithSpriteFrameName函数创建精灵对象，其中mountain1.png是SpriteSheet.plist在坐标文件中定义的精灵帧名

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