**西南大学**

**计算机与信息科学学院**

网络工程专业综合设计报告

**题 目：**  水果忍者游戏开发

**年级、专业：** 19 **级** 网络工程 **专业** 01 **班**

**学生姓名：**  金琦琦

**指导教师**  刘红俊

**提交日期：**  2022 **年** 6 **月** 1 **日**

|  |
| --- |
| **网络工程专业综合设计报告评定：** |
| **成绩：**  **指导教师（签字）：**  **年 月 日** |

目录

[1 游戏开发背景概述 1](#_Toc12573)

[2 水果忍者游戏效果需求 2](#_Toc8925)

[3 水果忍者游戏设计方案 2](#_Toc2025)

[3.1 游戏模块设计 2](#_Toc970)

[3.2 类的设计 3](#_Toc16707)

[1) PrepareScene类——继承CCLayer 4](#_Toc7926)

[2) GameStart类——继承CCLayer 5](#_Toc1845)

[3) FruitManager类——继承CCNode 5](#_Toc16223)

[4 切水果游戏具体实现描述 6](#_Toc9567)

[4.1 场景绘制模块实现 6](#_Toc25689)

[4.1.1 菜单界面的绘制 6](#_Toc16170)

[4.1.2 菜单界面实现触控效果 7](#_Toc26473)

[4.1.3 菜单界面的水果刀效果 9](#_Toc1221)

[4.1.4 菜单界面的菜单项选择 10](#_Toc27094)

[4.1.5 游戏场景的绘制 16](#_Toc17636)

[4.2 游戏控制模块 18](#_Toc4068)

[4.2.1 实现多种水果的创建 18](#_Toc27030)

[4.2.2 实现切水果 22](#_Toc32474)

[4.2.3 实现循环创建水果 25](#_Toc8367)

[4.2.4 实现计分机制 27](#_Toc31807)

[5 源代码 30](#_Toc13461)

[6 系统设计总结 62](#_Toc821)

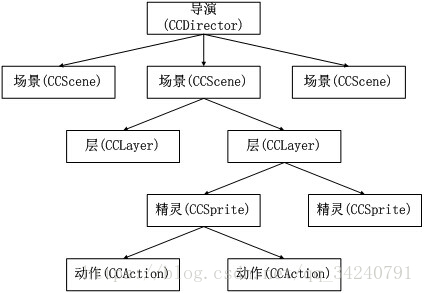
[7 参考文献 63](#_Toc13449)

1 游戏开发背景概述

游戏引擎是指一些已编写好的可编辑电脑游戏系统或者一些交互式实时图像应用程序的核心组件。游戏引擎最重要的一块就是游戏的渲染，美工会按照不同的面把材质贴图赋予模型，这相当于为骨骼蒙上皮肤，最后再通过渲染引擎把模型、动画、光影、特效等所有效果实时计算出来并展示在屏幕上。我的项目里用到的游戏引擎是Cocos2d-x2.2.3，Coco2d-x就是一个基于**OpenGL** 的游戏引擎，因此它的绘图部分完全由 OpenGL 实现。 OpenGL 是一个基于 C 语言的三维图形 API，基本功能包含绘制几何图形、变换、着色、光照、贴图等。该引擎最重要的优点是易于使用：游戏开发者可以把关注焦点放在游戏设置本身，而不必消耗大量时间学习晦涩难懂的OpenGL ES。开发者甚至可以在完全不懂 OpenGL ES，不懂计算机图形学的情况下，利用Cocos2d-x API开发出商业上成功的游戏作品。另外，关心Cocos2d-x的开发者自发建立了多个社区组织，可以方便的查阅各类技术资料。

cocos2d-x引擎主要有六个部分组成，分别为导演(CCDirector)、摄像机(CCCamera)、场景(CCScene)、布景(CCLayer)、精灵(CCSprite)、以及动作(CCAction)六个部分组成。

相互之间的关系框架如下图所示  
导演控制场景，场景控制图层，图层控制精灵，精灵控制动作。



Coco2d-x游戏引擎使用须知

（1）整个游戏一般只有一个导演。  
（2）一个游戏可以由多个场景组成（菜单界面、游戏界面、游戏结束界面等），但是每次导演只能指导一个场景的运作。  
（3）一个场景可以由多个图层组成（背景层、道具层、英雄层、怪物层等，每个图层都可以由多个精灵元素组成（标签、按钮、菜单等）。  
（4）对于图层上的每个精灵元素，一般都附带相应的动作，可以带多个动作。如挥刀、使用魔法、跑、跳，以及放大、缩小、旋转等动作。

2 水果忍者游戏效果需求

1. 实现两个场景并实现场景的切换（游戏的菜单界面和开始游戏界面）
2. 实现触屏事件，并显示水果刀效果。
3. 实现间断性的创建**多种**水果，并实现水果的上抛下落
4. 判断水果是否被切到，实现水果被切后的落体运动
5. 实现计分机制，通过计分机制判断是否结束游戏

3 水果忍者游戏设计方案

3.1 游戏模块设计

本次游戏设计主要分为两个模块：场景绘制模块和游戏控制模块。各个模块之间作用如下：

场景绘制模块：主要负责游戏中两个静态场景的绘制，加载场景中的各个节点和节点动作

游戏控制模块：主要负责游戏进行状态的控制，负责游戏中各种动画的创建，动画播放，动画事件，和动画销毁等。

当程序启动时绘制一个HelloWorld场景（这个场景是游戏引擎自带的，我只是稍微修改了HelloWorldScene类的代码，给这个场景换了一个背景图片），1.5秒后刷新界面，替换到制菜单界面，菜单界面准备好后会播放音频，并且可以选择菜单项。如果选择了开始游戏则替换到游戏场景，当游戏场景绘制完成时就由控制模块控制游戏的进行与结束，游戏结束会回到菜单界面，整体的流程图如下

游戏开始

开始

菜单界面

HelloWorld场景

游戏进行场景

游戏控制模块

程序结束

场景绘制模块

游戏结束

3.2 类的设计

下面是主要类的关系图

CCScene

CCNode

**PrepareScene**

**GameStart**

**FruitManage**

CCLayer

**Entity**

**Fruit**

CCSprite

CCTouchDelegate

游戏控制模块

场景绘制模块

关键类介绍：

1. **PrepareScene类——继承CCLayer**

主要功能：负责菜单界面的创建和绘制，实现菜单界面的触控和菜单项选择

主要成员函数：

static ::CCScene\* scene()：创建一个PrepareScene对象，把这个对象添加到临时成员变量scene中，并返回scene对象。即绘制一个PrepareScene层，用这个层来描绘一个静态场景

virtual bool ccTouchBegan(CCTouch\* , CCEvent\* )：用户触碰事件响应函数

virtual void ccTouchMoved(CCTouch\* , CCEvent\* )：用户触碰移动事件响应函数

virtual void ccTouchEnded(CCTouch\* , CCEvent\* )：用户放开事件响应函数

void loadlogo()：加载界面logo

void loadtitle()：加载游戏标题

void loadmenu()：加载菜单

void update(float)：默认定时器的刷新函数体，定时器每刷新一次就调用一次函数

void scle\_rotate(int )：设置旋转动作

void loadGame()：替换到游戏场景

主要成员变量：

CCArray\* m\_fruitBrokArr：数组类，存放被切的水果数组，以便后续对该数组内的水果进行操作

CCPoint \_point[2]：触碰点数组,记录前后两个触碰点用于计算角度

CCSize size：记录屏幕大小

CCPoint pt：记录屏幕触碰点位置

float angel：记录触屏移动时的角度

bool ifColli(int , CCPoint )：判断是否触碰到主界面的水果

bool ifstart：记录主界面状态是否已经准备好，准备好后播放音频并响应菜单选择事件

void Colli(int )：创建被切后的水果，并显示刀光

CCMotionStreak\* streak：拖尾效果

1. **GameStart类——继承CCLayer**

主要功能：实现游戏场景的创建和绘制，实现触控

主要成员函数：

static CCScene\* scene()：绘制一个GameStart层，用这个层来描绘一个静态场景，返回值是一个场景对象

virtual bool init()：初始化函数

virtual bool ccTouchBegan(CCTouch\* , CCEvent\* )

virtual void ccTouchMoved(CCTouch\* , CCEvent\* )

virtual void ccTouchEnded(CCTouch\* , CCEvent\* )

float ang：记录触屏滑动角度

CCSize size：记录屏幕大小

CCPoint \_point[2]：触碰点数组,记录前后两个触碰点用于计算角度

FruitManager\* fruitmanager：游戏主要控制类实例

CCParticleSystemQuad\* knifecolor：触屏点的粒子效果

CCMotionStreak\* streak：触屏拖尾效果

1. **FruitManager类——继承**CCNode

主要功能：控制水果的循环上抛下落和旋转动作、水果被切后的落体、果汁效果、刀光效果

主要成员函数

virtual void update(float dt)：默认定时器的刷新函数体

CCAction\* createAction(CCPoint )：创建水果斜抛并旋转动作

void setPt(CCPoint )：设置节点坐标

void setAngel(float ang)：设置节点旋转角度

void createFruit(float dt)：创建水果并添加到场景中

void actionEnd(CCNode\* )：水果上抛下落动作，刀光效果和果汁效果结束后执行的函数回调动作的函数体

void CollisionWithPoint(int )：水果被切到时，把切成两瓣的水果添加到场景中，并添加果汁效果

void gameOver()：游戏结束，替换到菜单界面

主要成员变量：

CCPoint pt：记录触屏点位置

CCSize size：屏幕大小

float angel：旋转角度

int notCutNum：记录未切到的水果个数

int score：记录分数

void notCut()：未切到水果个数加1，替换场景中的精灵图

CCArray\* m\_fruitArr：未被切的水果数组

CCArray\* m\_fruitBrokenArr：被切的水果数组

CCArray\* m\_systemArr：放射粒子系统数组

ShowNumberNode\* snn; //显示数字的类

4 切水果游戏具体实现描述

4.1 场景绘制模块实现

4.1.1菜单界面的绘制

菜单界面的元素位置的位置都是固定，添加元素到图层固定位置很简单，无非是先创建一个精灵，设置好在屏幕中的位置，然后就可以添加到图层中并执行动作了，只是每个精灵的位置和执行的动作有所不同。所有元素的加载都在以下几个函数中实现。

void loadlogo()：加载界面logo

void loadtitle()：加载游戏标题

void loadmenu()：加载菜单

这里挑一个看，例如游戏标题的添加



CCActionInterval是一个连续动作类，CCCallFunc是函数回调动作类，CCSequence是组合动作类，squen动作的含义是，先执行跳跃动作：从自身的初始位置0.8秒内经过两次跳跃20个单位的高度到达坐标（480,400）的位置；当跳跃动作执行完后紧接着执行函数回调动作，调用loadmenu函数

4.1.2 菜单界面实现触控效果

继承自触屏代理协议CCTouchDelegate的布景层类CCLayer可以检测触屏事件并调用回调函数。在布景层类中可以重写ccTouchesBegan等函数获得触屏的信息

virtual bool ccTouchBegan(CCTouch\* pTouches, CCEvent\* pEvent);

CCTouch触点类封装了触摸点的信息，包括触摸点的横纵坐标值和触摸点的ID号，作为在函数中传递的参数在回调函数中，通过一系列操作获得我们需要的坐标值

先重写registerWithTouchDispatcher函数，在此函数中通过导演类获得分发触摸类，并调用**触摸代理addTargetedDelegate函数允许本布景层（PrepareScene）接受触摸事件**。（导演类中封装了一个m\_pTouchDispatcher 成员变量，是绑定导演类的触屏调度对象）

addTargetedDelegate函数的参数：加入代理的对象（当前层）、触摸的优先级、是否拦截触摸点。

bool PrepareScene::init()

{

……

//得到触屏调度对象并注册触屏侦听事件，拦截触碰事件

CCDirector::sharedDirector()->getTouchDispatcher()->addTargetedDelegate(this, 1, true);

……

return true;

}

在PrepareScene类重载触控事件函数 ，三个重载函数分别在用户触碰到屏幕时、触碰点移动时和用户放开时调用，通过触控事件的重载函数获得触碰点的坐标和触碰移动时的轨迹，并记录在pt、angel属性中

class PrepareScene : public CCLayer

{ public:

**……**

//触碰事件的重载函数

virtual bool ccTouchBegan(CCTouch\* pTouches, CCEvent\* pEvent);

virtual void ccTouchMoved(CCTouch\* pTouches, CCEvent\* pEvent);

virtual void ccTouchEnded(CCTouch\* pTouches, CCEvent\* pEvent);

**……**

private:

CCPoint \_point[2]; //触碰点数组,记录前后两个触碰点用于计算角度

CCPoint pt; //记录屏幕触碰点位置

float angel; //记录触屏移动时的角度

**……**

};

触控事件重载函数

//处理用户按下事件

bool PrepareScene::ccTouchBegan(CCTouch\* pTouches, CCEvent\* pEvent)

{

CCPoint point = pTouches->getLocation(); //获得触碰点的坐标（包含x,y属性）

\_point[index++] = point;

return true;

}

//处理Touch Move事件

void PrepareScene::ccTouchMoved(CCTouch\* pTouches, CCEvent\* pEvent)

{

CCPoint point = pTouches->getLocation();

pt = point; //记录触碰点位置

if (index > 1)

{

index = 0;

}

\_point[index++] = point;

angel = atan((\_point[0].y - \_point[1].y) / (\_point[0].x - \_point[1].x)) \* 180 / 3.14; //计算移动角度

}

//处理用户放开事件

void PrepareScene::ccTouchEnded(CCTouch\* pTouches, CCEvent\* pEvent)

{

pt = ccp(500, 500);

}

4.1.3 菜单界面的水果刀效果

在PrepareScene类中添加私有数据成员CCMotionStreak\* streak;并在构造函数中初始化streak对象。

PrepareScene::PrepareScene()

{

index = 0;

angel = 0;

ifstart = false;

pt=ccp(500,500);

streak = CCMotionStreak::create(0.5f,4.0f,10.0f,ccc3(251,4,207),"knife2.png");

this->addChild(streak,2); //拖尾效果设置在第二层

}

第一个参数是设置显示时间，第二个参数是最小单元段（值设置大了转动的时候会有菱角的感觉），第三个参数是宽度，第四个是背景色，

第五个参数是设置背景图片

在ccTouchMoved触屏事件中，设置拖尾效果的位置为触碰点的位置

//处理Touch Move事件

void PrepareScene::ccTouchMoved(CCTouch\* pTouches, CCEvent\* pEvent)

{

CCPoint point = pTouches->getLocation();

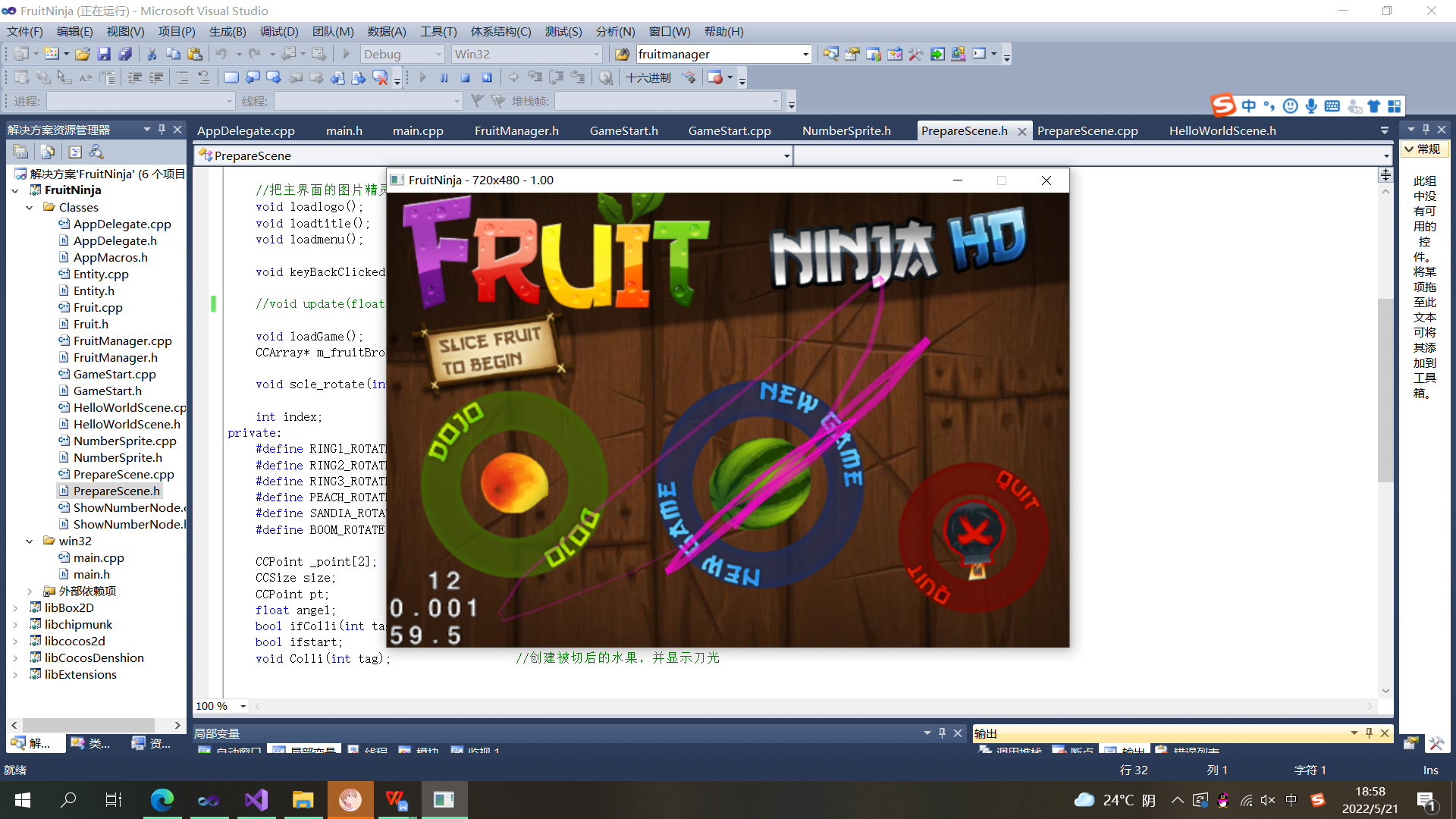
**streak->setPosition(point);**

……

}

4.1.4 菜单界面的菜单项选择

菜单界面的菜单项是三个水果如下图，这里需要作出的效果是当水果刀划到菜单项时，水果被切开并下落，在延迟1.5秒后切换到游戏场景



判断触碰点是否碰到某个菜单图片这一功能在PrepareScene类的私有成员函数ifColli(int tag,CCPoint point)中完成，ifColli的两个参数分别是精灵编号和触碰点位置。所以在开始写ifColli函数之前先要给三个菜单的精灵打上编号，编号在精灵添加到场景前标上

void PrepareScene::loadmenu()

{

**……**

peach->setTag(4); //桃子4

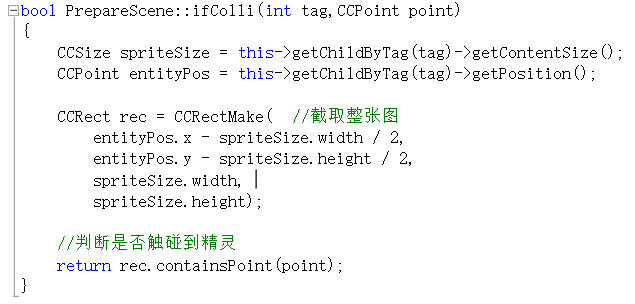
sandia->setTag(5); //西瓜5

boom->setTag(6); //炸弹6

**……**

}

ifColli函数的实现。创建一个矩形对象，得到编号为tag的精灵占据的位置，判断传入的坐标类实例point是否触碰到矩形的边框

  
CCRect是一个矩形类。包含：起始坐标（左下角坐标）CCPoint、矩阵的尺寸大小CCSize两个属性。

height

width

x,y

值得注意的是CCRect类中：  
intersectsRect函数，可以用作两个CCRect矩形是否相交，即碰撞检测。  
containsPoint 函数，可以用作判断点CCPoint是否在CCRect矩形中。

交点落在矩形的边上

则判断为切到水果

为了实时判断触屏点是否划到菜单上，需要不断执行ifColli函数。这里通过默认定时器的刷新函数体来实现持续的执行ifColli。

在PrepareScene::init()中启用默认定时器并在PrepareScene类中重载刷新函数体，每刷新一帧就调用一次刷新函数体update

void PrepareScene::update(float delta)

{

if (ifstart && ifColli(4, pt)) {……}

if (ifstart && ifColli(5, pt)) {……}

if (ifstart && ifColli(6, pt)) CCDirector::sharedDirector()->end(); //结束游戏

}

水果被切到之后设置水果的下落。落体运动和时间有关（和时间的平方成比例），所以我们仍然需要用update函数实现，update函数的delta参数就是两次帧刷新时间的延迟，把时间累加起来就有计时器的功能，可以计数水果下落时间

**位移**与时间的关系 

创建被切后的两瓣水果，并用一个数组来保存，以便我们对被切的水果进行后续的操作

class PrepareScene : public CCLayer

{ **……**

CCArray\* m\_fruitBrokArr; //被切的水果数组

void Colli(int tag); //创建被切后的两瓣水果，并显示刀光

**……**

};

水果被切之后的过程如下图所示：

在update函数中设置两瓣水果的运动轨迹

15 \* t + 10 \* t\* t

水果

判断到水果被切

1. 显示刀光
2. 消除完整的水果
3. 创建两瓣水果
4. 计时器清零

**T=0**

计时器为t时的一帧

**T=t**

**T=1.5**时切换场景

1. 第一个过程由Colli函数实现。先移除被切的水果，添加被切后的两瓣水果，如果是桃子，就用两瓣桃子的图片分别创建两个精灵，设置这两个精灵的位置还是在原来完整桃子的位置上，设置刀光的角度为触屏点滑动的角度，桃子切开的角度为刀光的角度，把两瓣水果添加到场景中和添加到m\_fruitBrokArr数组中。

水果被切到后需要清零计时器，从零开始计时下落时间

void PrepareScene::Colli(int tag)

{ ifstart = false;

this->removeChild(this->getChildByTag(tag)); //从场景中移除被碰到的精灵

CCSprite\* sp3 = CCSprite::create("flash.png"); //刀光

if (tag == 4)

{ sp1 = CCSprite::create("peach-1.png");

sp2 = CCSprite::create("peach-1.png");

sp1->setPosition(ccp(120, 180)); sp1->setRotation(angel);

sp2->setPosition(ccp(350, 180)); sp2->setRotation(angel);

sp3->setPosition(ccp(350, 180)); sp3->setRotation(180 - angel);

}

if (tag == 5)

{……}

sp1->setTag(7); sp2->setTag(8); sp3->setTag(9);

this->addChild(sp1); this->addChild(sp2); this->addChild(sp3);

//刀光消失动画

CCActionInterval\* rotate = CCRotateBy::create(0.5f, 90.0f, 0);

CCSequence\* action = CCSequence::create(rotate, NULL);

sp3->runAction(action);

m\_fruitBrokArr->addObject(sp1);

m\_fruitBrokArr->addObject(sp2);

ti = 0; //充当计时器计时，水果被切到时开始计时

}

Colli函数是在水果被切后立即调用的函数，在update函数中判断水果被切到后立马调用

void PrepareScene::update(float delta)

{ ti += delta;

if (ifstart && ifColli(4, pt)) Colli(4);

if (ifstart && ifColli(5, pt)) Colli(5);

……

}

做完以上内容我们可以看看效果，我切的是西瓜，中间的西瓜的确被切开了，但是仍然在原来的位置，且没有其他动作



1. 接下来就要在update函数中对m\_fruitBrokArr数组中存放的水果进行操作，使它实现落体了。当然，当m\_fruitBrokArr中没有被切开的水果时，for循环内的代码块也不会被执行。

void PrepareScene::update(float delta)

{

……

//设置被切的水果的落体运动

for (int i = 0; i < m\_fruitBrokArr->count(); i++)

{

CCSprite\* sp = (CCSprite\*)m\_fruitBrokArr->objectAtIndex(i);

if (sp->getTag() == 7)

{

sp->setPositionX(sp->getPositionX() - 3);//向左边下落

sp->setPositionY(sp->getPositionY() - (15 \* ti + 10 \* ti \* ti));

}

if (sp->getTag() == 8)

{……}//同向左下落的代码

if(ti==0)this->scheduleOnce(schedule\_selector(PrepareScene::loadGame),1.5f);

}

}

在for循环中有一个一次性定时器，1.5秒后会执行loadGame函数体，这个函数的作用是替换到游戏场景，那么当水果下落了1.5秒后，场景就会切换

void PrepareScene::loadGame(float dt)

{

this->unscheduleUpdate(); //取消默认定时器

this->cleanup(); //停止所有动作和回调函数

CocosDenshion::SimpleAudioEngine::sharedEngine()->pauseEffect(ID);

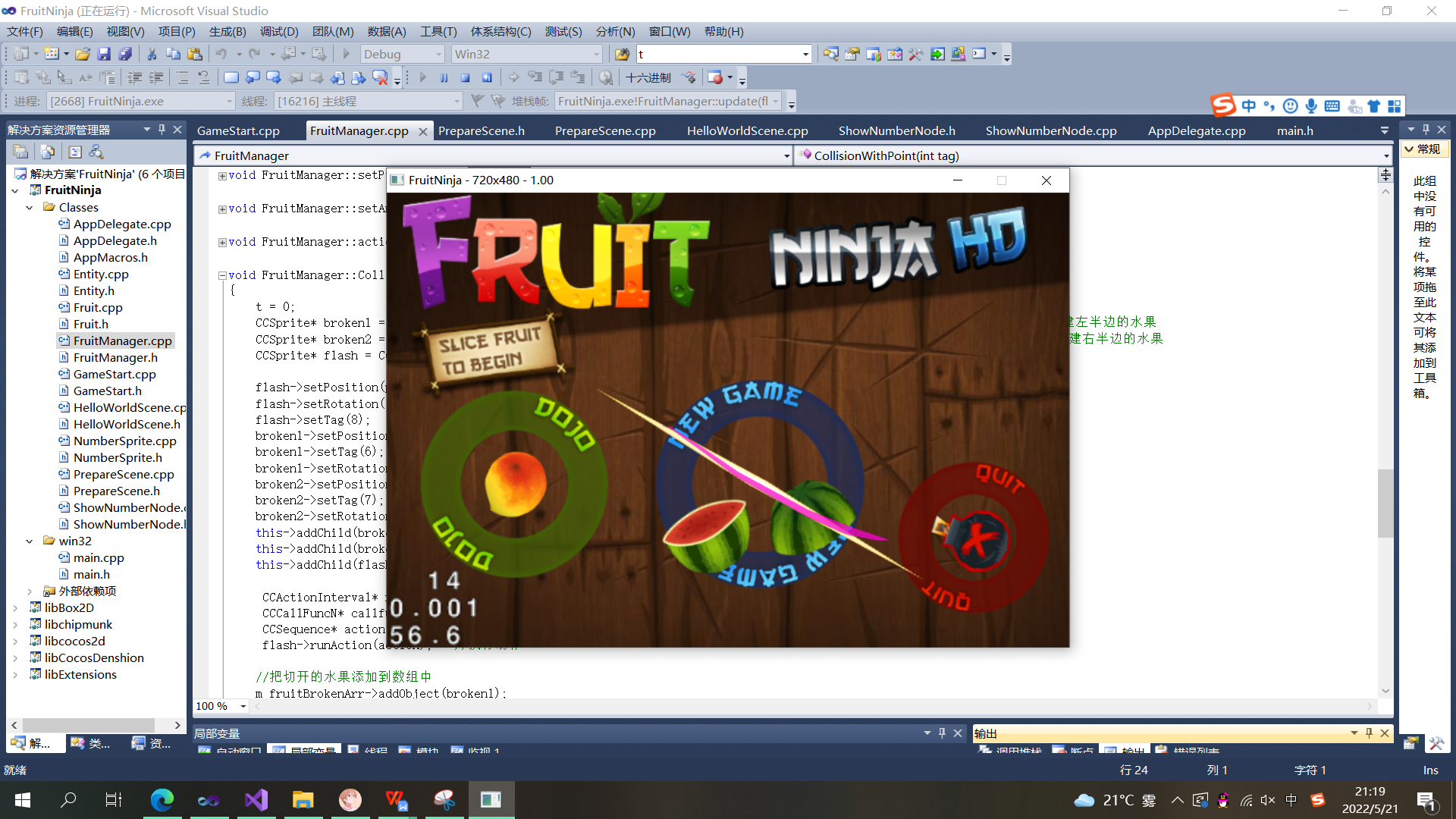
CCScene\* scene = GameStart::scene(); //创建GameStart类场景

CCTransitionCrossFade\* pTranslateScene = CCTransitionCrossFade::create(0.5f, scene); //场景切换类

CCDirector::sharedDirector()->replaceScene(pTranslateScene); //0.5秒内慢慢淡化到scene场景

}

看一下效果，可以实现下落并在1.5秒后切换场景



4.1.5 游戏场景的绘制

游戏场景主要有三个任务，1.图层绘制，2.实现触控效果，3.把触控点信息传给GameSart图层类中的FruitManege对象

图层绘制不用多说，添加的都是些固定位置的精灵，且这些精灵元素都没有动作，很好实现，像贴图一样粘到固定位置就行。另外触控的实现和菜单界面的代码是一样的，为了实时传输触控点的信息，在触控事件重载函数中传输给FuitManage对象

GameSart图层类的初始化函数，添加五个精灵元素，分别是背景图片，左上角分数图片，和右上角三个小图图片，允许GameStart层接收触屏事件

  bool GameStart::init()

{

size = CCDirector::sharedDirector()->getWinSize(); //获取屏幕大小

CCSprite\* background = CCSprite::create("background.jpg");

background->setPosition(ccp(size.width / 2, size.height / 2));

this->addChild(background); //设置游戏背景

CCSprite\* x = CCSprite::create("x.png");

CCSprite\* xx = CCSprite::create("xx.png");

CCSprite\* xxx = CCSprite::create("xxx.png");

CCSprite\* score = CCSprite::create("score.png");

x->setPosition(ccp(562, 440));

xx->setPosition(ccp(588, 440));

xxx->setPosition(ccp(620, 440));

score->setPosition(ccp(40, 410));

score->setScale(1.5); //控制整体缩放

this->addChild(x);

this->addChild(xx);

this->addChild(xxx);

this->addChild(score);

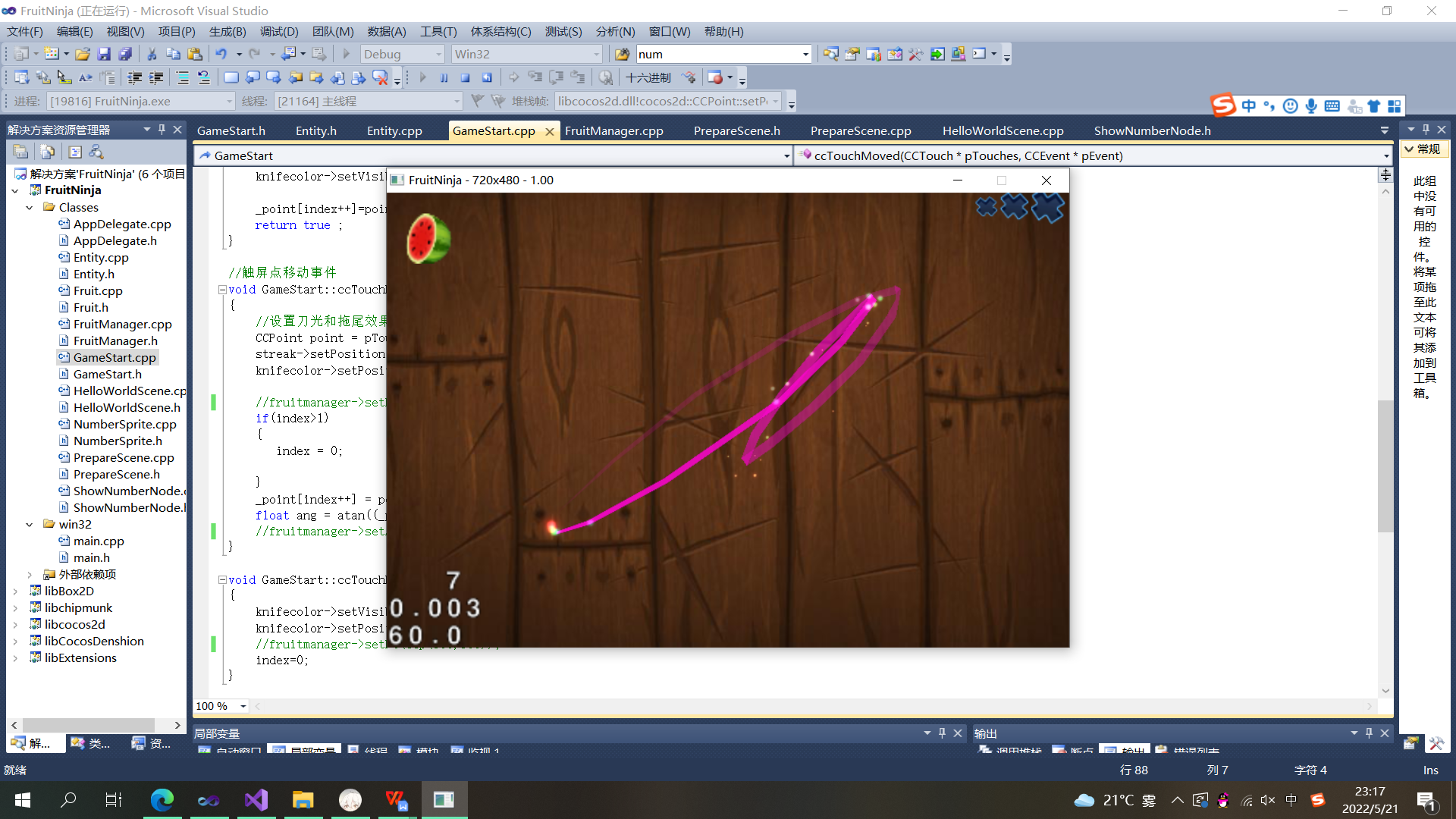
//注册触碰侦听事件

CCDirector::sharedDirector()->getTouchDispatcher()->addTargetedDelegate(this, 1, true);

return true;

}

实现效果



4.2 游戏控制模块

控制模块需要实现的功能有一次创建多种类的水果、实现所有创建出的水果的上抛和下落动作、切水果和水果落体效果、循环创建水果、实现计分

4.2.1 实现多种水果的创建

每次游戏开始后抛出的水果种类和数量都是随机的，我们需要完成的功能就是创建不同的水果。不同的水果是用不同的图片创建的精灵，都是精灵类。

**我们先创建一个实体类，**实体类中有一个精灵数据类型的成员，实现两个方法：（1）给精灵数据成员赋值（2）返回当前精灵数据成员的指针

class Entity :public CCNode

{public:

CCSprite\* getSprite(); //返回当前对象指针

void bindSprite(CCSprite\* sprite); //给精灵绑定具体的元素

private:

//设置一个精灵对象

CCSprite\* m\_sprite;

};

创建实体类的目的是为了方便之后创建不同种类的实体，例如水果、蛋糕或者炸弹。

接下来要创建的就是水果类

class Fruit : public Entity

{ public:

// Fruit();

CREATE\_FUNC(Fruit); //create();函数

virtual bool init(); //init();函数在create();中被调用

bool IsCollideWithPoint(CCPoint point); //判断水果是否被切到

};

这里的init函数和coco2d-x中的大多数类的初始化函数一样设置为虚函数，当继承这个类的派生类想要修改Init函数功能时提供了方便（实现多态）。IsCollideWithPoint方法的实现和PrepareScene类中的ifColli方法的实现一致。

创建一个fruitmanage类，实现对水果类对象的管理,控制游戏的进行状态

class FruitManager : public CCNode

{public:

FruitManager(); //构造函数

CREATE\_FUNC(FruitManager);

virtual bool init(); //初始化函数

virtual void update(float dt); //默认定时器的刷新函数体

CCAction\* createAction(CCPoint point); //创建水果斜抛并旋转动作

CCPoint pt; //记录触屏点位置

CCSize size; //屏幕大小

float angel; //旋转角度

void setPt(CCPoint p); //设置节点坐标

void setAngel(float ang); //设置节点旋转角度

void createFruit(float dt);//创建水果精灵并添加到场景中

CCArray\* m\_fruitArr; //未被切的水果数组

CCArray\* m\_fruitBrokenArr; //被切的水果数组

CCArray\* m\_systemArr; //放射粒子系统数组

ShowNumberNode\* snn; //显示数字的类

void actionEnd(CCNode\* sender);

void CollisionWithPoint(int tag); //水果被切到时，把切成两瓣的水果添加到场景中

private:

int notCutNum; //记录未切到的水果个数

int score; //记录分数

void notCut(); //未切到水果个数加1，替换场景中的精灵图

void gameOver();

};

实现不同种类的水果创建是在createFruit(float dt)函数中完成。这里我们要用到精灵帧缓存，精灵帧缓存主要用来缓存多张小图合并后的精灵表单，精灵表单中包含多张小图，通过CCSpriteFrameCache缓存后通过CCSpriteFrame引用其中一个小图片。可以用TexturePacker工具来生成精灵表单和它对应的.plist文件（用来捆绑信息的一种文件格式，数据格式是.xml）.plist文件指定了每个独立的精灵帧在这张“大图”里面的位置和大小

CCSpriteFrameCache\* framecache = CCSpriteFrameCache::sharedSpriteFrameCache();

//寻找工程目录下名为fruit.png的图片，并把这个图片加入精灵帧缓存池

framecache->addSpriteFramesWithFile("fruit.plist", "fruit.png");//通过fruit.plist文件创建多个精灵帧，fruit.png纹理将与被创建的精灵帧结合，



精灵帧缓存池CHICCSprite::createWithSpriteFrameName(

池

CCSprite::createWithSpriteFrameName(“5”)

取出缓存池中名字是5的精灵帧创建一个精灵

peach

2

1

21

22

11

12

4

3

5

当把所有需要用的精灵帧都添加到了精灵帧缓存池中，就可以利用精灵帧缓存池来创建精灵了，为了方便通过精灵帧的名字来创建精灵，在制作精灵表单的时候我把所有的纹理图片都用数字来命名，完整的水果分别是1-5，半边水果就是11、12、21、22、31……这样，创建随机种类的水果问题就变成了取一个1-5的随机数

rand%5



banana





peach

1

2

3

4

5

void FruitManager::createFruit(float dt)

{

if (notCutNum >= 3) return; //未切到的水果个数大于3个，函数返回

Fruit\* fruit = NULL;

for (int i = 0; i <= rand() % 4; i++) //随机创建1-4个水果

{ int index = rand() % 5 + 1;

//const\*char 转CCString,CCString转char

fruit = Fruit::create(); fruit->bindSprite(CCSprite::createWithSpriteFrameName(CCString::createWithFormat("%d.png", index)->getCString()));

fruit->setTag(index); //设置节点的Tag编号，为随机的1-5

CCPoint p = ccp(rand() % 640, -50); //创建横坐标为0-640之间任意值，纵坐标为-50的坐标点(起始坐标点在界面的左下角)

fruit->setPosition(p); //设置水果的初始位置为p坐标

this->addChild(fruit); //把水果添加到场景中

CocosDenshion::SimpleAudioEngine::sharedEngine()->playEffect(m4); //播放音频4

m\_fruitArr->addObject(fruit); //把当前水果实例添加到数组中

fruit->runAction(createAction(p)); //执行动作

}

}

凡是创建出来的水果初始位置都在屏幕外，需要执行createAction动作跳到屏幕的显示区域。createAction动作主要实现了水果上抛的同时旋转，上抛动作用CCJumpuTo来实现。

jump = CCJumpTo::create(2.0f, ccp(rand() % 200 + 400, -20), 340.0f, 1);

第一个参数是动作经过的时间，第二个参数是目标位置，第三个参数是跳跃的高度，第四个参数是跳跃次数。

以上代码的意思就是2秒内经过一次跳跃，到达ccp(rand() % 200 + 400, -20)屏幕右下角向下超出屏幕20个单位的位置，跳跃高度为340个单位

水果

跳跃的目的坐标

一次跳跃的高度340

屏幕显示区域

4.2.2 实现切水果

因为水果是创随机创建的，为了方便判断这些随机产生的水果是否被切到，我们需要一个数组类来存放完整的水果，每个创建出的水果都会被添加到这个数组中。同样的也需要一个m\_fruitBrokenArr数组来存放切开的水果

水果被切后的第一个流程创建两瓣水果显示刀光效果计时器清零在CollisionWithPoint函数中实现，代码和PrepareScene类中代码的相似，这里就不赘述

主要的不同点要判断**所有**被抛上来的水果是否被切到，在FruitManage类的默认定时器刷新函数体中写一下代码，所有被创建的水果都在m\_fruitArr数组中，遍历数组就可以判断每个水果是否被切到，如果被刀切到就从数组中移除

createFruit

fruit

m\_fruitArr

IsCollideWithPoint

m\_**fruitBrokenArr**

1 4 3

11 12

update

4 3

m\_fruitArr

m\_**fruitBrokenArr**

IsCollideWithPoint?

11 12

m\_**fruitBroke**数组中的半边水果的下落过程和PrepareScene图层类中的一样，当它们运动到屏幕外不可见区域的时候从数组中和场景中消除，不消除会占用资源

在update函数中设置两瓣水果的运动轨迹

15 \* t + 10 \* t\* t

水果

判断到水果被切

1. 显示刀光
2. 消除完整的水果
3. 创建两瓣水果
4. 计时器清零

**T=0**

计时器为t时的一帧

**T=t**

掉出屏幕可见区域，

从数组中移除

void FruitManager::update(float delta)

{

t += delta;

CCObject\* obj = NULL;

Fruit\* fruit = NULL;

//遍历水果数组，判断是否被切到

for (int i = 0; i < m\_fruitArr->count(); i++)

{

fruit = (Fruit\*)m\_fruitArr->objectAtIndex(i);

if (fruit->IsCollideWithPoint(pt)) //被刀切到

{

m\_fruitArr->removeObject(fruit); //把水果从水果数组中移除

CollisionWithPoint(fruit->getTag()); //把切成两瓣的水果添加到场景中 this->removeChild(fruit); //场景中消除完整的水果

}

}

//遍历被切开的水果数组，描绘被切开的水果的运动

for (int i = 0; i < m\_fruitBrokenArr->count(); i++)

{

CCSprite\* sp = (CCSprite\*)m\_fruitBrokenArr->objectAtIndex(i);

if (sp->getTag() == 6)

{

sp->setPositionX(sp->getPositionX() - 2); //屏幕每刷新一次，左边的水果就左移动2个单位

sp->setPositionY(sp->getPositionY() - 20 \* t \* t); //屏幕每刷新一次，水果以-10的加速度向下移动

}

if (sp->getTag() == 7)

{

sp->setPositionX(sp->getPositionX() + 2);

sp->setPositionY(sp->getPositionY() - 20 \* t \* t);

}

}

}

4.2.3 实现循环创建水果

上面我们实现的效果只能执行一次水果的上抛，水果创建一次后就不再创建；而且创建出来的水果只有有两种结果，第一种是被刀切到然后从场景和数组中移除，另一种是执行完完整的CCJumpTo动作，运动到了屏幕外，但是并未从场景和数组中移除。那么可以根据水果数组是否为空来判断要不要执行创建水果的函数。

等所有未被切到的水果运动到固定的位置后，就从水果数组中抹除，当判断到水果数组中没有元素时，就开始创建水果

水果

水果

目标位置

水果

remove

remove

m\_fruitArr

createFruit

Is NULL

在update函数中判断水果数组是否为空,当判断到为空时，延迟2秒钟执行reateFruit函数

if(m\_fruitArr->count()==0)

this->scheduleOnce(schedule\_selector(FruitManager::createFruit),2.0f);

那么运动到屏幕外指定位置的水果怎么从数组中消除呢，这要用到函数回调动作，当水果执行完跳跃运动后就执行函数回调动作，函数体为actionEnd(CCNode\* sender)，参数是执行动作的对象，在函数actionEnd中实现从场景和数组中消除水果。 createAction函数中写了下面这一段，下面这一段完整的动作只有未被切到的水果才会执行；对于被切到的水果节点，已经从图层中移除，自然也就没有了动作

CCSpawn\* jumpandrota = CCSpawn::create(jump, rotate, NULL); //组合动作，使水果上抛的同时旋转

CCCallFuncN\* callfun = CCCallFuncN::create(this, callfuncN\_selector(FruitManager::actionEnd)); //函数回调动作，回调函数actionEnd

CCSequence\* action = CCSequence::create(jumpandrota, callfun, NULL); // 组合动作，使动作按顺序执行

void FruitManager::actionEnd(CCNode\* sender)

{

CCSprite\* sprite = (CCSprite\*)sender;

if (sprite->getTag() < 6) //判断调用该函数的对象，未被切到的水果

{

notCut();

m\_fruitArr->removeObject(sprite);

this->removeChild(sprite);

if (m\_fruitArr->count() == 0)

createFruit(0.0f);

}

else

this->removeChild(this->getChildByTag(9)); //刀光动作消失后，消除粒子效果

}

4.2.4 实现计分机制

当切到水果时需要记录分数，未切到水果时需要记录未切到个数，当超过三个就结束游戏；

当切到水果时，分数score+1，FruitManage类中有一个ShowNumberNode类对象snn，ShowNumberNode是一个显示数字图片的类,它有一方法f\_ShowNumber可以修改显示的数字，当score变化时，用这个方法就可以动态的显示score的值。

现在FruitManage的构造函数中初始化snn对象

FruitManager::FruitManager()

{

pt = ccp(500, 500); //创建坐标

angel = 0; //属性赋值

notCutNum = 0;

score = 0;

snn = ShowNumberNode::CreateShowNumberNode("menu\_num.png", 999, 22, 30);

snn->f\_ShowNumber(score);

snn->setPosition(ccp(95, 410));

this->addChild(snn, 2, 2);

}

CreateShowNumberNode("menu\_num.png", 999, 22, 30)：第一个参数是用来创建snn对象的数字图片（如下），第二个参数是该对象最大能显示的值的大小，第三和第四个参数分别是下面图片中单个数字的长和宽

menu_num

在update函数中实时更新score的显示

void FruitManager::update(float delta)

{ ……

//显示得分

**snn->f\_ShowNumber(score);**

for (int i = 0; i < m\_fruitArr->count(); i++)

{

fruit = (Fruit\*)m\_fruitArr->objectAtIndex(i);

if (fruit->IsCollideWithPoint(pt)) //被刀切到

{ ……

score++;

}

}

……

}

回调函数actionEnd中有noCut函数，这个函数是当有一个水果未被切到时，需要替换游戏场景右上角的精灵图片，用红色叉覆盖原来的精灵，这一段的代码也很简单

void FruitManager::notCut()

{

notCutNum++; //未切到水果记录+1

//创建红叉图片精灵，并设置精灵的位置

CCSprite\* xf = CCSprite::create("xf.png");

CCSprite\* xxf = CCSprite::create("xxf.png");

CCSprite\* xxxf = CCSprite::create("xxf.png");

xf->setPosition(ccp(562, 440));

xxf->setPosition(ccp(588, 440));

xxxf->setPosition(ccp(620, 440));

//把精灵图添加到场景中

if (notCutNum == 1)

{

this->addChild(xf);

}

if (notCutNum == 2)

{

this->addChild(xf);

this->addChild(xxf);

}

if (notCutNum == 3)

{

this->addChild(xf);

this->addChild(xxf);

this->addChild(xxxf);

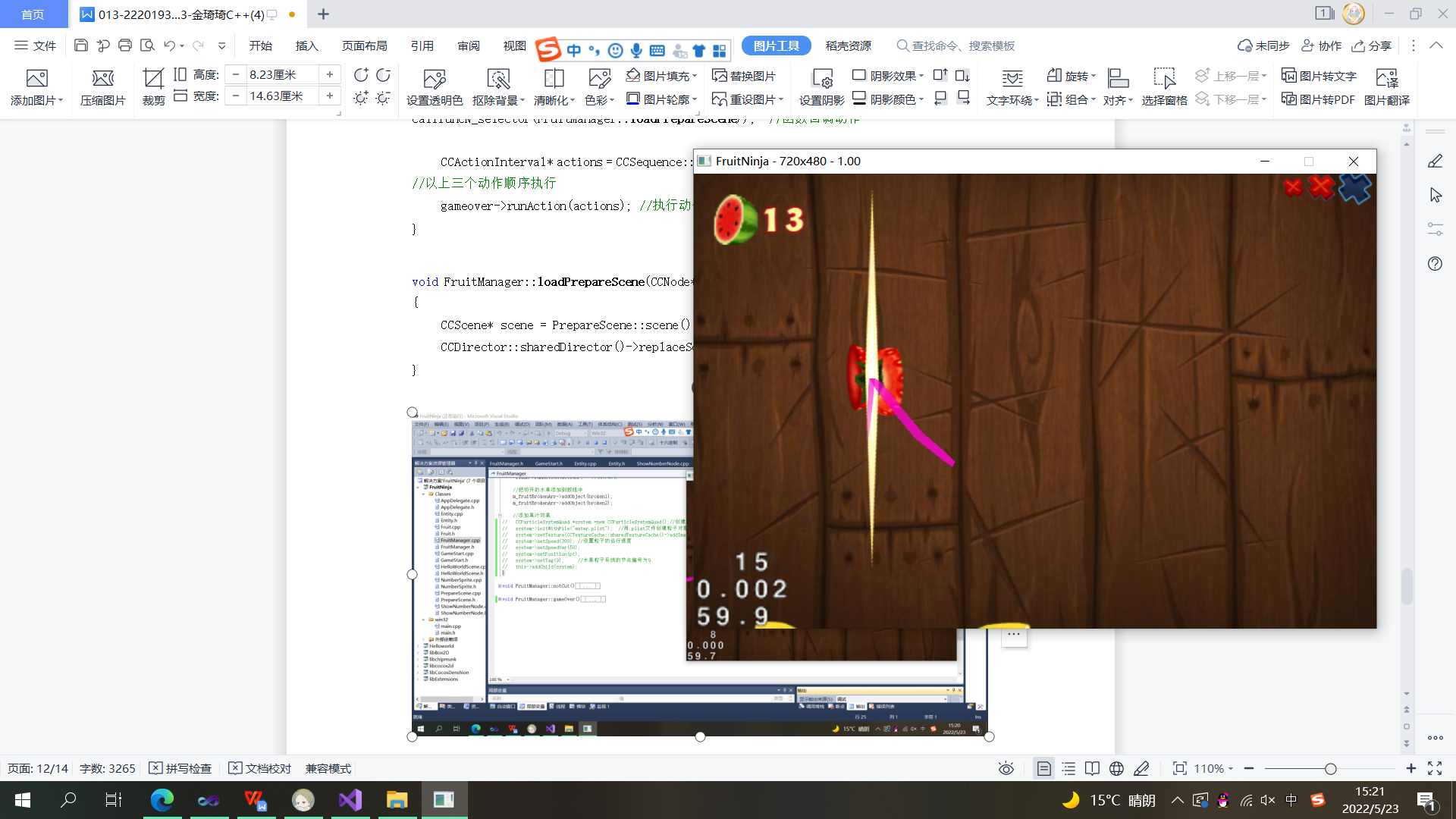
gameOver();

}

}

当未切到水果达到三个的时候，就结束游戏，替换场景返回到菜单界面通过gameOver函数实现

实现后就是下面的效果，左上角显示切到13个水果，右上角显示未切到两个水果



5 源代码

**PrepareScene.h**

#ifndef \_preparescene\_

#define \_preparescene\_

#include "cocos2d.h"

using namespace cocos2d;

USING\_NS\_CC;

class PrepareScene : public CCLayer

{

public:

PrepareScene();

virtual bool init();

static cocos2d::CCScene\* scene();

CREATE\_FUNC(PrepareScene);

//触碰事件的重载函数

virtual bool ccTouchBegan(CCTouch\*, CCEvent\*); //用户触碰事件响应函数

virtual void ccTouchMoved(CCTouch\* pTouches, CCEvent\* pEvent); //用户触碰移动事件响应函数

virtual void ccTouchEnded(CCTouch\* pTouches, CCEvent\* pEvent); //用户放开事件响应函数

//把主界面的图片精灵添加到场景中

void loadlogo();

void loadtitle();

void loadmenu();

void keyBackClicked();

void update(float delta); //默认定时器的刷新函数体

void loadGame(float dt); //替换成游戏场景

CCArray\* m\_fruitBrokArr; //被切的水果数组

void scle\_rotate(int i); //设置精灵旋转

int index;

private:

#define RING1\_ROTATE 1

#define RING2\_ROTATE 2

#define RING3\_ROTATE 3

#define PEACH\_ROTATE 4

#define SANDIA\_ROTATE 5

#define BOOM\_ROTATE 6

CCPoint \_point[2]; //触碰点数组,记录前后两个触碰点用于计算角度

CCSize size; //记录屏幕大小

CCPoint pt; //记录屏幕触碰点位置

float angel; //记录触屏移动时的角度

bool ifColli(int tag, CCPoint point); //判断是否触碰到主界面的水果

bool ifstart; //记录主界面状态，是否已经准备好

void Colli(int tag); //创建被切后的水果，并显示刀光

unsigned int ID; //记录界面背景音效的ID

CCMotionStreak\* streak; //拖尾效果

};

#endif

**PrepareScene.cpp**

#include "PrepareScene.h"

#include "HelloWorldScene.h"

#include "GameStart.h"

#include "math.h"

float ti = 0;

char\* pm1 = "/sound/menu.mp3";

char\* pm2 = "/sound/splatter.mp3";

PrepareScene::PrepareScene()

{

index = 0;

angel = 0;

ifstart = false;

pt = ccp(500, 500);

streak = CCMotionStreak::create(1.0f, 4.0f, 10.0f, ccc3(251, 4, 207), "knife2.png");

this->addChild(streak, 2); //拖尾特效在第二层

}

CCScene\* PrepareScene::scene()

{

CCScene\* scene = CCScene::create();

PrepareScene\* layer = PrepareScene::create();//创建PrepareScene类层对象

scene->addChild(layer); //把PrepareScene类层添加到场景中

return scene;

}

bool PrepareScene::init()

{

m\_fruitBrokArr = CCArray::create();

m\_fruitBrokArr->retain();

size = CCDirector::sharedDirector()->getWinSize(); //获取屏幕大小

CCSprite\* backsp = CCSprite::create("background.jpg"); //用图片创建精灵backsp

backsp->setPosition(ccp(size.width / 2, size.height / 2)); //把锚点设置在屏幕正中央

this->addChild(backsp, -1); //背景图片添加到最底层，负数代表最高优先级

//得到触屏调度对象并注册触屏侦听事件，拦截触碰事件

CCDirector::sharedDirector()->getTouchDispatcher()->addTargetedDelegate(this, 1, true);

this->setKeypadEnabled(true); //使能够使用返回键

loadlogo();

this->scheduleUpdate(); //默认定时器

//预处理音效文件

CocosDenshion::SimpleAudioEngine::sharedEngine()->preloadEffect(pm1);

CocosDenshion::SimpleAudioEngine::sharedEngine()->preloadEffect(pm2);

return true;

}

//处理用户按下事件

bool PrepareScene::ccTouchBegan(CCTouch\* pTouches, CCEvent\* pEvent)

{

CCPoint point = pTouches->getLocation(); //获得触碰点的坐标（包含x,y属性）

\_point[index++] = point;

return true;

}

//处理Touch Move事件

void PrepareScene::ccTouchMoved(CCTouch\* pTouches, CCEvent\* pEvent)

{

CCPoint point = pTouches->getLocation();

streak->setPosition(point);

pt = point; //记录触碰点位置

if (index > 1)

{

index = 0;

}

\_point[index++] = point;

angel = atan((\_point[0].y - \_point[1].y) / (\_point[0].x - \_point[1].x)) \* 180 / 3.14; //计算移动角度

}

//处理用户放开事件

void PrepareScene::ccTouchEnded(CCTouch\* pTouches, CCEvent\* pEvent)

{

pt = ccp(500, 500);

}

void PrepareScene::loadlogo()

{

//添加主界面的横幅

CCActionInterval\* move = CCMoveTo::create(0.4f, ccp(size.width / 2, 387.5)); //持续动作，0.4秒内移动到屏幕中间，高387.5的位置

CCActionInterval\* action = CCEaseOut::create(move, 2.0f); //变速动作，由快到慢，加速率2

CCCallFunc\* callfun = CCCallFunc::create(this, callfunc\_selector(PrepareScene::loadtitle)); //回调函数动作，函数体loadtitle

CCSequence\* seque = CCSequence::create(action, callfun, NULL); //组合动作，顺序执行action和callfun

CCSprite\* sprite = CCSprite::create("home-mask.png"); //以图片home-mask.png创建精灵

sprite->setPosition(ccp(size.width / 2, 550)); //把刚刚创建的精灵锚点设置在屏幕中间的顶上

this->addChild(sprite); //把精灵添加到场景中

sprite->runAction(seque); //执行动作，并接着调用loadtitle函数

//添加主界面logo

CCSprite\* splgo = CCSprite::create("logo.png"); //以图片logo.png创建精灵

splgo->setPosition(ccp(150, 550));

this->addChild(splgo);

CCActionInterval\* move2 = CCMoveTo::create(0.4f, ccp(160, 410));

CCActionInterval\* action2 = CCEaseOut::create(move2, 3.0f); //变速动作，由快到慢

splgo->runAction(action2);

}

void PrepareScene::loadtitle()

{

CCSprite\* title = CCSprite::create("ninja.png");

title->setPosition(ccp(480, 400));

this->addChild(title);

CCActionInterval\* jump = CCJumpTo::create(0.8f, ccp(480, 400), 20.0f, 2);

CCActionInterval\* action = CCEaseIn::create(jump, 2.0f); //变速动作，由慢到快

CCCallFunc\* callfun = CCCallFunc::create(this, callfunc\_selector(PrepareScene::loadmenu));

CCSequence\* squen = CCSequence::create(action, callfun, NULL);

title->runAction(squen); //执行动作，并接着调用loadmenu函数

CCSprite\* title2 = CCSprite::create("home-desc.png");

title2->setPosition(ccp(-80, 300));

this->addChild(title2);

CCActionInterval\* action2 = CCMoveTo::create(0.5f, ccp(100, 300));

title2->runAction(action2);

}

void PrepareScene::loadmenu()

{

CCSprite\* scle1\_dojo = CCSprite::create("dojo.png");

CCSprite\* scle2\_new = CCSprite::create("new-game.png");

CCSprite\* scle3\_quit = CCSprite::create("quit.png");

CCSprite\* peach = CCSprite::create("peach.png");

CCSprite\* sandia = CCSprite::create("sandia.png");

CCSprite\* boom = CCSprite::create("boom.png");

scle1\_dojo->setPosition(ccp(120, 180));

scle1\_dojo->setTag(1); //第一个圆圈

scle2\_new->setPosition(ccp(350, 180));

scle2\_new->setTag(2);

scle3\_quit->setPosition(ccp(550, 130));

scle3\_quit->setTag(3);

peach->setPosition(ccp(120, 180));

peach->setTag(4); //桃子4

sandia->setPosition(ccp(350, 180));

sandia->setTag(5); //西瓜5

boom->setPosition(ccp(550, 130));

boom->setTag(6); //炸弹6

this->addChild(scle1\_dojo);

this->addChild(scle2\_new);

this->addChild(scle3\_quit);

this->addChild(peach);

this->addChild(sandia);

this->addChild(boom);

//旋转效果

scle\_rotate(RING1\_ROTATE);

scle\_rotate(RING2\_ROTATE);

scle\_rotate(RING3\_ROTATE);

scle\_rotate(PEACH\_ROTATE);

scle\_rotate(SANDIA\_ROTATE);

scle\_rotate(BOOM\_ROTATE);

ifstart = true; //记录主界面已经准备好

ID = CocosDenshion::SimpleAudioEngine::sharedEngine()->playEffect(pm1, true);

}

void PrepareScene::keyBackClicked() //按返回键时替换场景

{

CCScene\* scene = HelloWorld::scene();

CCDirector::sharedDirector()->replaceScene(scene);

CCDirector::sharedDirector()->getTouchDispatcher()->removeDelegate(this); //注销触屏侦听事件

}

void PrepareScene::update(float delta)

{

ti += delta;

if (ifstart && ifColli(4, pt)) { Colli(4); CocosDenshion::SimpleAudioEngine::sharedEngine()->playEffect(pm2); }

if (ifstart && ifColli(5, pt)) { Colli(5); CocosDenshion::SimpleAudioEngine::sharedEngine()->playEffect(pm2); }

if (ifstart && ifColli(6, pt)) CCDirector::sharedDirector()->end(); //结束游戏

//设置水果被切后的运动

for (int i = 0; i < m\_fruitBrokArr->count(); i++)

{

CCSprite\* sp = (CCSprite\*)m\_fruitBrokArr->objectAtIndex(i);

if (sp->getTag() == 7)

{

sp->setPositionX(sp->getPositionX() - 3);

sp->setPositionY(sp->getPositionY() - (15 \* ti + 12 \* ti \* ti));

}

if (sp->getTag() == 8)

{

sp->setPositionX(sp->getPositionX() + 3);

sp->setPositionY(sp->getPositionY() - (15 \* ti + 1 \* ti \* ti));

}

this->scheduleOnce(schedule\_selector(PrepareScene::loadGame), 1.5f);

}

}

void PrepareScene::scle\_rotate(int i)

{

CCRotateBy\* rotate;

switch (i)

{

case RING1\_ROTATE: rotate = CCRotateBy::create(9.0f, 360.0f, 360.0f); break; //相对动作，9秒后旋转360度

case RING2\_ROTATE: rotate = CCRotateBy::create(12.0f, -360.0f, -360.0f); break; //12秒钟逆时针旋转360度

case RING3\_ROTATE: rotate = CCRotateBy::create(15.0f, 360.0f, 360.0f); break;

case PEACH\_ROTATE: rotate = CCRotateBy::create(12.0f, -360.0f, -360.0f); break; //相对动作，9秒后旋转360度

case SANDIA\_ROTATE: rotate = CCRotateBy::create(7.0f, 360.0f, 360.0f); break; //12秒钟逆时针旋转360度

case BOOM\_ROTATE: rotate = CCRotateBy::create(12.0f, -360.0f, -360.0f);; break;

default:break;

}

CCRepeatForever\* repeatForever = CCRepeatForever::create(rotate); //无限重复动作

this->getChildByTag(i)->runAction(repeatForever);

}

void PrepareScene::loadGame(float dt)

{

this->unscheduleUpdate(); //取消默认定时器

this->cleanup(); //停止所有动作和回调函数

CocosDenshion::SimpleAudioEngine::sharedEngine()->pauseEffect(ID);

CCScene\* scene = GameStart::scene(); //创建GameStart类场景

CCTransitionCrossFade\* pTranslateScene = CCTransitionCrossFade::create(0.5f, scene); //场景切换类

CCDirector::sharedDirector()->replaceScene(pTranslateScene); //0.5秒内慢慢淡化到scene场景

}

bool PrepareScene::ifColli(int tag, CCPoint point)

{

CCSize spriteSize = this->getChildByTag(tag)->getContentSize();

CCPoint entityPos = this->getChildByTag(tag)->getPosition();

CCRect rec = CCRectMake( //截取整张图

entityPos.x - spriteSize.width / 2,

entityPos.y - spriteSize.height / 2,

spriteSize.width,

spriteSize.height);

//判断是否触碰到精灵

return rec.containsPoint(point);

}

void PrepareScene::Colli(int tag)

{

ifstart = false;

this->removeChild(this->getChildByTag(tag)); //从场景中移除被碰到的精灵

CCSprite\* sp1 = NULL;

CCSprite\* sp2 = NULL;

CCSprite\* sp3 = CCSprite::create("flash.png");

if (tag == 4)

{

sp1 = CCSprite::create("peach-1.png");

sp2 = CCSprite::create("peach-1.png");

sp1->setPosition(ccp(120, 180));

sp1->setRotation(angel);

sp2->setPosition(ccp(120, 180));

sp2->setRotation(angel);

sp3->setPosition(ccp(120, 180));

sp3->setRotation(180 - angel);

}

if (tag == 5)

{

sp1 = CCSprite::create("51.png");

sp2 = CCSprite::create("52.png");

sp1->setPosition(ccp(350, 180));

sp1->setRotation(angel);

sp2->setPosition(ccp(350, 180));

sp2->setRotation(angel);

sp3->setPosition(ccp(350, 180));

sp3->setRotation(180 - angel);

}

sp1->setTag(7);

sp2->setTag(8);

sp3->setTag(9);

this->addChild(sp1);

this->addChild(sp2);

this->addChild(sp3);

//刀光消失动画

CCActionInterval\* rotate = CCRotateBy::create(1.0f, 90.0f, 0);

//CCCallFunc\* callfun = CCCallFunc::create(this,callfunc\_selector(PrepareScene::loadGame));

CCSequence\* action = CCSequence::create(rotate, NULL);

sp3->runAction(action);

m\_fruitBrokArr->addObject(sp1);

m\_fruitBrokArr->addObject(sp2);

ti = 0; //充当计时器计时，水果被切到时计时开始

//this->scheduleOnce(schedule\_selector(PrepareScene::loadGame),1.0f);

}

**GameStart.h**

#ifndef \_gamestart\_

#define \_gamestart\_

#include "cocos2d.h"

#include "FruitManager.h"

#include "SimpleAudioEngine.h"

using namespace cocos2d;

USING\_NS\_CC;

class GameStart : public CCLayer

{

public:

GameStart();

static CCScene\* scene();

virtual bool init();

CREATE\_FUNC(GameStart);

virtual bool ccTouchBegan(CCTouch\* pTouches, CCEvent\* pEvent);

virtual void ccTouchMoved(CCTouch\* pTouches, CCEvent\* pEvent);

virtual void ccTouchEnded(CCTouch\* pTouches, CCEvent\* pEvent);

//CCPoint getPoint();

//CCPoint point;

float ang;

CCSize size;

void keyBackClicked();

//void addfruit(float dt);

CCPoint \_point[2];

int index;

FruitManager\* fruitmanager;

CCParticleSystemQuad\* knifecolor; //粒子效果

CCMotionStreak\* streak; //拖尾效果

};

#endif

**GameStart.cpp**

#include "GameStart.h"

#include "PrepareScene.h"

#include "math.h"

float uti = 0;

GameStart::GameStart()

{

index = 0;

//创建CCMotionStreak 对象并添加到场景中

streak = CCMotionStreak::create(0.5f, 4.0f, 10.0f, ccc3(251, 4, 207), "knife.png");

this->addChild(streak, 2); //在第二层

knifecolor = new CCParticleSystemQuad(); //水果刀的粒子效果

knifecolor->initWithFile("knifecolor.plist"); //用.plist文件创建粒子对象的内容

knifecolor->setTexture(CCTextureCache::sharedTextureCache()->addImage("5color.plist.png"));//用5color.plist.png图片描绘粒子纹理

this->addChild(knifecolor, 2); //添加在第二层中

knifecolor->setVisible(false);

}

CCScene\* GameStart::scene()

{

CCScene\* scene = CCScene::create();

GameStart\* layer = GameStart::create();

scene->addChild(layer); //把GameStart类层添加到场景中

return scene;

}

bool GameStart::init()

{

size = CCDirector::sharedDirector()->getWinSize(); //获取屏幕大小

CCSprite\* background = CCSprite::create("background.jpg");

background->setPosition(ccp(size.width / 2, size.height / 2));

this->addChild(background); //设置游戏背景

CCSprite\* x = CCSprite::create("x.png");

CCSprite\* xx = CCSprite::create("xx.png");

CCSprite\* xxx = CCSprite::create("xxx.png");

CCSprite\* score = CCSprite::create("score.png");

x->setPosition(ccp(562, 440));

xx->setPosition(ccp(588, 440));

xxx->setPosition(ccp(620, 440));

score->setPosition(ccp(40, 410));

score->setScale(1.5); //控制整体缩放

this->addChild(x);

this->addChild(xx);

this->addChild(xxx);

this->addChild(score);

//注册触碰侦听事件

CCDirector::sharedDirector()->getTouchDispatcher()->addTargetedDelegate(this, 1, true);

this->setKeypadEnabled(true); //能够使用返回键

fruitmanager = FruitManager::create();

this->addChild(fruitmanager);

return true;

}

//用户按下事件

bool GameStart::ccTouchBegan(CCTouch\* pTouches, CCEvent\* pEvent)

{

//从触摸点获取到在OpenGL坐标系中的坐标

CCPoint point = pTouches->getLocation();

knifecolor->setPosition(point);

knifecolor->setVisible(true);

\_point[index++] = point;

return true;

}

//触屏点移动事件

void GameStart::ccTouchMoved(CCTouch\* pTouches, CCEvent\* pEvent)

{

//设置刀光和拖尾效果的位置

CCPoint point = pTouches->getLocation();

streak->setPosition(point);

knifecolor->setPosition(point);

fruitmanager->setPt(point);

if (index > 1)

{

index = 0;

}

\_point[index++] = point;

ang = atan((\_point[0].y - \_point[1].y) / (\_point[0].x - \_point[1].x)) \* 180 / 3.14;

fruitmanager->setAngel(ang);

}

void GameStart::ccTouchEnded(CCTouch\* pTouches, CCEvent\* pEvent)

{

knifecolor->setVisible(false);

knifecolor->setPosition(ccp(500, 500));

fruitmanager->setPt(ccp(500, 500));

index = 0;

}

void GameStart::keyBackClicked()

{

this->cleanup(); //停止所有运行的动作和回调函数

CCScene\* scene = PrepareScene::scene();

CCDirector::sharedDirector()->replaceScene(scene);

}

**FruitManage.h**

#ifndef \_fruitmanager\_

#define \_fruitmanager\_

#include "cocos2d.h"

#include "ShowNumberNode.h"

using namespace cocos2d;

class FruitManager : public CCNode

{

public:

FruitManager(); //构造函数

CREATE\_FUNC(FruitManager);

virtual bool init(); //初始化函数

virtual void update(float dt); //默认定时器的刷新函数体

CCAction\* createAction(CCPoint point); //创建水果斜抛并旋转动作

CCPoint pt; //记录触屏点位置

CCSize size; //屏幕大小

float angel; //旋转角度

void setPt(CCPoint p); //设置节点坐标

void setAngel(float ang); //设置节点旋转角度

void createFruit(float dt);//创建水果精灵并添加到场景中

CCArray\* m\_fruitArr; //未被切的水果数组

CCArray\* m\_fruitBrokenArr; //被切的水果数组

CCArray\* m\_systemArr; //放射粒子系统数组

ShowNumberNode\* snn; //显示数字的类

void actionEnd(CCNode\* sender);

void CollisionWithPoint(int tag); //水果被切到时，把切成两瓣的水果添加到场景中，并添加果汁效果

private:

int notCutNum; //记录未切到的水果个数

int score; //记录分数

void notCut(); //未切到水果个数加1，替换场景中的精灵图

void gameOver();

};

#endif

**FruitManage.cpp**

#include "FruitManager.h"

#include "Fruit.h"

#include "GameStart.h"

#include "PrepareScene.h"

#include "ShowNumberNode.h"

float t = 0;

char\* m2 = "/sound/over.mp3";

char\* m3 = "/sound/start.mp3";

char\* m4 = "/sound/throw.mp3";

char\* m5 = "/sound/splatter.mp3";

FruitManager::FruitManager()

{

pt = ccp(500, 500); //创建坐标

angel = 0; //属性赋值

notCutNum = 0;

score = 0;

snn = ShowNumberNode::CreateShowNumberNode("menu\_num.png", 999, 22, 30);

snn->f\_ShowNumber(score);

snn->setPosition(ccp(95, 410));

this->addChild(snn, 2, 2);

}

bool FruitManager::init()

{

//SpriteFrame（精灵帧）只加载一次，后续一直保存在 SpriteFrameCache 中

CCSpriteFrameCache\* framecache = CCSpriteFrameCache::sharedSpriteFrameCache(); //创建一个精灵帧缓存池

//寻找工程目录下名为fruit.png的图片，并把这个图片加入精灵帧缓存池

framecache->addSpriteFramesWithFile("fruit.plist", "fruit.png");

m\_fruitArr = CCArray::create(); //创建m\_fruitArr数组内容

m\_fruitArr->retain(); //保留资源不被回收（自动释放内存）

m\_fruitBrokenArr = CCArray::create(); //创建m\_fruitBrokenArr数组内容

m\_fruitBrokenArr->retain();

//一次性定时器，延迟2秒后执行createFruit函数

this->scheduleOnce(schedule\_selector(FruitManager::createFruit), 2.0f);

//默认定时器，刷新间隔为一帧(屏幕每刷新一次，就执行一次update()函数)

this->scheduleUpdate();

//预处理音效文件

CocosDenshion::SimpleAudioEngine::sharedEngine()->preloadEffect(m2);

CocosDenshion::SimpleAudioEngine::sharedEngine()->preloadEffect(m3);

CocosDenshion::SimpleAudioEngine::sharedEngine()->playEffect(m3); //

CocosDenshion::SimpleAudioEngine::sharedEngine()->preloadEffect(m4);

CocosDenshion::SimpleAudioEngine::sharedEngine()->preloadEffect(m5);

return true;

}

void FruitManager::createFruit(float dt)

{

if (notCutNum >= 3) return; //未切到的水果个数大于3个，函数返回

Fruit\* fruit = NULL;

for (int i = 0; i <= rand() % 4; i++) //随机创建1-4个水果

{

int index = rand() % 5 + 1;

//创建一个水果，(通过名字去访问精灵帧缓存中的精灵帧(const\*char 转CCString,CCString转char))

fruit = Fruit::create();

fruit->bindSprite(CCSprite::createWithSpriteFrameName(CCString::createWithFormat("%d.png", index)->getCString()));

fruit->setTag(index); //设置节点的Tag编号，为随机的1-5

CCPoint p = ccp(rand() % 640, -50); //创建横坐标为0-640之间任意值，纵坐标为-50的坐标点(起始坐标点在界面的左下角)

fruit->setPosition(p); //设置水果的初始位置为p坐标

this->addChild(fruit); //把水果添加到场景中

CocosDenshion::SimpleAudioEngine::sharedEngine()->playEffect(m4); //播放音频4

m\_fruitArr->addObject(fruit); //把当前水果实例添加到数组中

fruit->runAction(createAction(p)); //执行动作

}

}

void FruitManager::update(float delta)

{

t += delta;

CCObject\* obj = NULL;

Fruit\* fruit = NULL;

//显示得分

snn->f\_ShowNumber(score);

//遍历水果数组，判断是否被切到

for (int i = 0; i < m\_fruitArr->count(); i++)

{

fruit = (Fruit\*)m\_fruitArr->objectAtIndex(i);

if (fruit->IsCollideWithPoint(pt)) //被刀切到

{

CocosDenshion::SimpleAudioEngine::sharedEngine()->playEffect(m5);

m\_fruitArr->removeObject(fruit); //把水果从水果数组中移除

CollisionWithPoint(fruit->getTag()); //把切成两瓣的水果添加到场景中，并添加果汁效果

this->removeChild(fruit); //场景中消除完整的水果

score++;

}

}

//遍历被切开的水果数组，描绘被切开的水果的运动

for (int i = 0; i < m\_fruitBrokenArr->count(); i++)

{

CCSprite\* sp = (CCSprite\*)m\_fruitBrokenArr->objectAtIndex(i);

if (sp->getPositionY() < 0)

{

m\_fruitBrokenArr->removeObject(sp); //从数组中移除，释放资源

this->removeChild(sp); //从场景中移除

continue;

}

if (sp->getTag() == 6)

{

sp->setPositionX(sp->getPositionX() - 2); //屏幕每刷新一次，左边的水果就左移动2个单位

sp->setPositionY(sp->getPositionY() - 20 \* t \* t); //屏幕每刷新一次，水果以-10的加速度向下移动

}

if (sp->getTag() == 7)

{

sp->setPositionX(sp->getPositionX() + 2);

sp->setPositionY(sp->getPositionY() - 20 \* t \* t);

}

}

if (m\_fruitArr->count() == 0)

this->scheduleOnce(schedule\_selector(FruitManager::createFruit), 2.0f);

}

CCAction\* FruitManager::createAction(CCPoint point)

{

CCActionInterval\* jump; //持续动作类

if (point.x <= 320) //向右上斜抛，并下落

//创建动作jump：两秒内，经过一次跳跃340单位(上抛),到达右下纵坐标为-20的位置(下落：跳跃完成后到达指定位置)

{

jump = CCJumpTo::create(2.0f, ccp(rand() % 200 + 400, -20), 340.0f, 1);

}

if (point.x > 320) //向左上斜抛，并下落

{

jump = CCJumpTo::create(2.0f, ccp(rand() % 100, -20), 340.0f, 1);

}

CCActionInterval\* rotate = CCRotateBy::create(2.0f, 180.f, 180.f); //创建动作rotate：两秒后x、y轴分别旋转180度

CCSpawn\* jumpandrota = CCSpawn::create(jump, rotate, NULL); //组合动作，使水果上抛的同时旋转

CCCallFuncN\* callfun = CCCallFuncN::create(this, callfuncN\_selector(FruitManager::actionEnd)); //函数回调动作，回调函数actionEnd

CCSequence\* action = CCSequence::create(jumpandrota, callfun, NULL); // 组合动作，使动作按顺序执行

return action;

}

void FruitManager::setPt(CCPoint P) //记录触屏点坐标

{

pt = P;

}

void FruitManager::setAngel(float ang) //设置角度

{

angel = ang;

}

void FruitManager::actionEnd(CCNode\* sender)

{

CCSprite\* sprite = (CCSprite\*)sender;

if (sprite->getTag() == 10) //游戏结束

{

CCScene\* scene = PrepareScene::scene(); //创建PrepareScene类对象

CCDirector::sharedDirector()->replaceScene(scene); //替换为场景

}

this->removeChild(sprite); //移除以图片game-voer创建的精灵

if (sprite->getTag() < 6) //未被切到的水果

{

notCut();

m\_fruitArr->removeObject(sprite);

this->removeChild(sprite);

if (m\_fruitArr->count() == 0)

createFruit(0.0f);

}

else

this->removeChild(this->getChildByTag(9)); //刀光动作消失后，消除粒子效果

}

void FruitManager::CollisionWithPoint(int tag)

{

t = 0;

CCSprite\* broken1 = CCSprite::createWithSpriteFrameName(CCString::createWithFormat("%d1.png", tag)->getCString()); //创建左半边的水果

CCSprite\* broken2 = CCSprite::createWithSpriteFrameName(CCString::createWithFormat("%d2.png", tag)->getCString()); //创建右半边的水果

CCSprite\* flash = CCSprite::createWithSpriteFrameName("flash.png"); //创建刀光

flash->setPosition(pt); //设置刀光位置在触屏点的位置

flash->setRotation(180 - angel); //设置顺时针旋转180-angel度

flash->setTag(8); //给刀光节点设置编号为8

broken1->setPosition(pt);

broken1->setTag(6); //左半边水果的节点编号设置为6

broken1->setRotation(-angel);

broken2->setPosition(pt);

broken2->setTag(7); //右半边水果的节点编号设置为7

broken2->setRotation(-angel);

this->addChild(broken1); //添加到场景中

this->addChild(broken2);

this->addChild(flash);

CCActionInterval\* rotate = CCRotateBy::create(0.3f, 90.0f, 0); //创建旋转动作，0.3秒后x方向旋转90度,即让刀光消失

CCCallFuncN\* callfun = CCCallFuncN::create(this, callfuncN\_selector(FruitManager::actionEnd));//创建函数回调动作

CCSequence\* action = CCSequence::create(rotate, callfun, NULL); //使旋转和actionEnd按顺序执行

flash->runAction(action); //执行动作

//把切开的水果添加到数组中

m\_fruitBrokenArr->addObject(broken1);

m\_fruitBrokenArr->addObject(broken2);

//添加果汁效果

CCParticleSystemQuad\* system = new CCParticleSystemQuad();//创建放射性粒子系统

system->initWithFile("water.plist"); //用.plist文件创建粒子对象的内容

system->setTexture(CCTextureCache::sharedTextureCache()->addImage("LavaFlow.plist.png")); //用LavaFlow.plist.png图片描绘粒子纹理

system->setSpeed(300); //设置粒子的运行速度

system->setSpeedVar(50);

system->setPosition(pt);

system->setTag(9); //水果粒子系统的节点编号为9

this->addChild(system);

}

void FruitManager::notCut()

{

notCutNum++; //未切到水果记录+1

//创建红叉图片精灵，并设置精灵的位置

CCSprite\* xf = CCSprite::create("xf.png");

CCSprite\* xxf = CCSprite::create("xxf.png");

CCSprite\* xxxf = CCSprite::create("xxf.png");

xf->setPosition(ccp(562, 440));

xxf->setPosition(ccp(588, 440));

xxxf->setPosition(ccp(620, 440));

//把精灵图添加到场景中

if (notCutNum == 1)

{

this->addChild(xf);

}

if (notCutNum == 2)

{

this->addChild(xf);

this->addChild(xxf);

}

if (notCutNum == 3)

{

this->addChild(xf);

this->addChild(xxf);

this->addChild(xxxf);

gameOver();

}

}

void FruitManager::gameOver()

{

CCSprite\* gameover = CCSprite::create("game-over.png");

gameover->setPosition(ccp(320, 240));

gameover->setTag(10);

this->addChild(gameover);

//创建缩放动作

CCActionInterval\* scaleLarge = CCScaleTo::create(1.0f, 1.5f, 1.5f); //1秒后等比放大1.5倍

CCActionInterval\* scaleSmall = CCScaleTo::create(1.0f, 0.5f, 0.5f); //1秒后等比缩小1.5倍

CCCallFuncN\* callFunc = CCCallFuncN::create(this, callfuncN\_selector(FruitManager::actionEnd)); //函数回调动作

CCActionInterval\* actions = CCSequence::create(scaleLarge, scaleSmall, callFunc, NULL); //以上三个动作顺序执行

gameover->runAction(actions); //执行动作

CocosDenshion::SimpleAudioEngine::sharedEngine()->playEffect(m2);

}

**Entity.h**

//继承结点类

class Entity :public CCNode

{

public:

Entity(); //构造函数

~Entity(); //析构函数

//

CCSprite\* getSprite(); //返回成员变量的指针

void bindSprite(CCSprite\* sprite); //给精灵绑定具体的元素

private:

//设置一个精灵对象

CCSprite\* m\_sprite;

};

**Entity.cpp**

Entity::Entity()

{

//初始化私有变量m\_sprite

m\_sprite = NULL;

}

Entity::~Entity() {}

CCSprite\* Entity::getSprite()

{

//返回成员变量指针

return this->m\_sprite;

}

void Entity::bindSprite(CCSprite\* sprite)

{

this->m\_sprite = sprite; //私有变量赋值

//把这个精灵添加到场景中

this->addChild(m\_sprite);

}

**Fruit.h**

class Fruit : public Entity

{

public:

// Fruit();

//创建对象的代码，会自动创建对象

//如果失败会自动回收内存，如果回收内存也失败就会自动删除对象

CREATE\_FUNC(Fruit); //create();函数

virtual bool init(); //init();函数在CREATE\_FUNC();中被调用

bool IsCollideWithPoint(CCPoint point); //判断水果是否被切到

};

**Fruit.cpp**

bool Fruit::init()

{

return true;

}

bool Fruit::IsCollideWithPoint(CCPoint point)

{

//获取元素的原始宽高

CCSize spriteSize = getSprite()->getContentSize(); //继承自父类Entity的方法

//获取元素锚点，在图片的中心点位置

CCPoint entityPos = getPosition(); //继承自CCNode

//截取水果的范围

CCRect rec = CCRectMake(

entityPos.x - spriteSize.width / 2, //小图的左下角横坐标

entityPos.y - spriteSize.height / 2, //小图的左下角纵坐标

spriteSize.width, //小图宽

spriteSize.height); //小图高

//判断point是否包含在矩形内或四条边上

return rec.containsPoint(point); //刀刃是否滑到了图片边缘

}

简要的使用说明



一个三个菜单项，选择两个水果就会开始游戏，选择炸弹退出程序，游戏场景就是切水果，切到计分，未切到达三个结束游戏，返回菜单界面

6 系统设计总结

（1）资料寻找困难。并不是没有相关的资源，而是网上的资源太多了，很多的资源并不是项目需要的，因此找齐所有的元素需要花费大量的时间。

（2）自学新的知识困难。学习新的游戏引擎不知道从哪一步开始，跟着网上的博客内容学的很零散，需要把这些零散的知识应用到项目里，并且学一个新知识需要学很多拓展的内容，需要看的东西太多太杂。

（3）没有相关的开发经验。虽然有非常清楚的设计思路，但是开发过程的涉及到的技术比较生疏，必须花费大量的时间去看论文，学习算法。这使得工程比预期花费更多的时间。

（4）另外通过这个项目，我也学到会了更合理的构建项目的结构。项目中每个类都有不同的功能，很清晰的划分了功能模块

7 参考文献

blog.csdn.net/qq\_34240791/category\_7645225.html Cocos2dx学习笔记

www.cnblogs.com/zilongshanren/archive/2011/09/27/2192928.html Cocos2d-x win7 + vs2010 配置图文详解

blog.csdn.net/qq\_34240791/article/details/80222832 Cocos2D-X 和 VS2010 环境配置

blog.csdn.net/qq\_34240791/article/details/80223236 cocos2dx第一个程序HelloWorld

blog.csdn.net/yu97271486/article/details/89852754 创建Sprite方法 详解

blog.csdn.net/qq\_34240791/article/details/80284946 cocos2dx[3.2](4) ——节点类Node

blog.csdn.net/qq\_34240791/article/details/80252677 cocos2dx[2.x](16)--基本动作CCAction

blog.csdn.net/qq\_34240791/article/details/80254239 cocos2dx[2.x](21)--场景切换

blog.csdn.net/qq\_34240791/article/details/80238393 cocos2dx[2.x](3)--定时器更新

blog.csdn.net/u013435551/article/details/38579747 cocos2dx3.2学习笔记之Director（导演类）

www.cnblogs.com/atong/p/3284847.html Cocos2d-x学习笔记（二十四）之 触屏事件

# /www.yisu.com/zixun/107577.html cocos2dx基础篇(9)——触碰事件Touch

blog.csdn.net/lusic01/article/details/51332292 keyBackClicked事件侦听

www.cnblogs.com/atong/p/3269929.html Cocos2d-x学习笔记（四）之 导演类CCDirector

blog.csdn.net/luojie1990go/article/details/9453937 cocos2d-x--CCMotionStreak拖尾效果

blog.51cto.com/u\_9634496/3070429 cocos2dx进阶学习之CCEGLView

blog.csdn.net/andy1515/article/details/113915545 cocos2dx中世界坐标，节点坐标，opengl坐标，屏幕坐标理解

blog.csdn.net/qq\_34240791/article/details/80254556 cocos2dx[2.x](22)--简单碰撞检测

www.cnblogs.com/learn-program/p/9643554.html cocos2d-x学习之路（三）——精灵与动作

blog.csdn.net/Meloor/article/details/101479021 cocos2dx 精灵的创建、纹理缓存、精灵帧缓存

www.cnblogs.com/gongyan/p/4539406.html#:~:text=精灵帧缓存主要用来缓存多张小图合并后的图片，一张图中包含多张小图，通过CCSpriteFrameCache缓存后通过CCSpriteFrame引用其中一个小图片。

blog.csdn.net/qq\_34240791/article/details/80225563 cocos2dx浅析几个重要类

blog.csdn.net/nanshan\_hzq/article/details/42125835 cocos2d-x类型转换(CCstring int string char UTF-8互转)以及字符串详解

www.cnblogs.com/pengyingh/articles/2436648.html CCSpriteFrameCache的用法

blog.csdn.net/tianxiawuzhei/article/details/44493511

# blog.csdn.net/farsa/article/details/9430155 cocos2d-x 图片数字显示

www.cnblogs.com/xiaomaomi-199207/p/4918052.html cocos2d-x CCParticleSystem粒子系统

blog.csdn.net/zjwfisheep/article/details/9155173 Cocos2d-x Scene生命周期 pushScene和replaceScene

# blog.csdn.net/qq\_34240791/article/details/80254139 cocos2dx[2.x](20)--粒子系统CCParticleSystem