现代操作系统应用开发实验报告

学号： 14331388 **班级：** 教务三班

姓名： 郑泽佳 **实验名称：** 实验 12

一．参考资料

课件

http://www.2cto.com/kf/201502/377611.html

http://bbs.csdn.net/topics/390917952

二．实验步骤

1、利用键盘事件实现对飞船左右移动、发射子弹的控制

键盘监听事件监听器配置好后，在事件分发器中，添加键盘监听器及其相应的绑定 Sprite

```
void Thunder::addKeyboardListener() {
    // TODO
    //
    auto keyListener = EventListenerKeyboard::create();
    keyListener->onKeyPressed = CC_CALLBACK_2(Thunder::onKeyPressed, this);
    keyListener->onKeyReleased = CC_CALLBACK_2(Thunder::onKeyReleased, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(keyListener, player);


}
```

响应键盘事件控制 player 的移动，通过条件判断使 player 不走出边界范围

```
void Thunder::onKeyPressed(EventKeyboard::KeyCode code, Event* event) {
    switch (code) {
        case cocos2d::EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        case cocos2d::EventKeyboard::KeyCode::KEY_A:
            if (player->getPositionX() - 1 - player->getContentSize().width / 2> 0 &&
                player->getPositionX() -1 + player->getContentSize().width / 2< visibleSize.width)
                player->setPosition(player->getPositionX() - 1, player->getPositionY());
            move -= 5;
            // TODO
            break;
        case cocos2d::EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
        case cocos2d::EventKeyboard::KeyCode::KEY_D:
            if (player->getPositionX() + 1 - player->getContentSize().width / 2> 0 &&
                player->getPositionX() + 1 + player->getContentSize().width / 2< visibleSize.width)
                player->setPosition(player->getPositionX() + 1, player->getPositionY());
            move += 5;
            // TODO
            break;
        case cocos2d::EventKeyboard::KeyCode::KEY_SPACE:
            fire();
            break;
        default:
            break;
    }
}
```

```
void Thunder::update(float f) {
    // TODO
    if (player->getPositionX() + move - player->getContentSize().width / 2> 0 &&
        player->getPositionX() + move + player->getContentSize().width / 2< visibleSize.width)
        player->setPosition(player->getPosition() + Vec2(move, 0));
}
```

2、用自定义事件实现：子弹和蜜蜂（陨石）相距小于一定距离时判定为击中，子弹和蜜蜂（陨石）

自定义事件订阅

```
void Thunder::addCustomListener() {
    // TODO
    auto meetListener = EventListenerCustom::create("meet", CC_CALLBACK_1(Thunder::meet, this));
    _eventDispatcher->addEventListenerWithFixedPriority(meetListener, 1);
}
```

判断是否碰撞，自定义事件的消息传递与发布

```
    for (int j = 0; j < bullets.size(); j++) {
        if (bullets[j] != NULL && enemys[i] != NULL &&
            bullets[j]->getPosition().getDistance(enemys[i]->getPosition()) < 30) {
            bullets[j]->removeFromParentAndCleanup(true);
            bullets.erase(bullets.begin() + j);
            j--;
            EventCustom e("meet");
            e.setUserData(&i);
            _eventDispatcher->dispatchEvent(&e);
        }
    }
```

```
void Thunder::meet(EventCustom* event) {

    int index = *(int*)event->getUserData();
    auto audio = SimpleAudioEngine::getInstance();
    audio->playEffect("music/explore.wav");
    enemys[index]->removeFromParentAndCleanup(true);
    enemys[index] = NULL;
}
```

添加调度器更新状态

```
// add schedule
schedule(schedule_selector(Thunder::update), 0.01f, kRepeatForever, 0);
```

3、游戏过程中有背景音乐，发射子弹、击中蜜蜂（陨石）时有音效

加载音乐资源

```
void Thunder::preloadMusic() {
    // TODO
    auto audio = SimpleAudioEngine::getInstance();
    audio->preloadBackgroundMusic("music/bgm.mp3");
    audio->preloadBackgroundMusic("music/explore.wav");
    audio->preloadBackgroundMusic("music/fire.wav");


}
```

播放背景音乐

```
void Thunder::playBgm() {
    // TODO
    auto audio = SimpleAudioEngine::getInstance();
    audio->playBackgroundMusic("music/bgm.mp3", true);
}
```

发射子弹和子弹击中时播放特效音

```
void Thunder::fire() {

    auto bullet = Sprite::create("bullet.png");
    bullet->setPosition(player->getPosition());
    addChild(bullet);
    bullets.insert(bullets.begin(), bullet);

    auto audio = SimpleAudioEngine::getInstance();
    audio->playEffect("music/fire.wav");
}
```

```
void Thunder::meet(EventCustom* event) {

    int index = *(int*)event->getUserData();
    auto audio = SimpleAudioEngine::getInstance();
    audio->playEffect("music/explore.wav");
    enemys[index]->removeFromParentAndCleanup(true);
    enemys[index] = NULL;
}
```

4、注意飞船、子弹的移动范围

飞船的移动范围

```
if (player->getPositionX() + move - player->getContentSize().width / 2> 0 &&
    player->getPositionX() + move + player->getContentSize().width / 2< visibleSize.width)
    player->setPosition(player->getPosition() + Vec2(move, 0));
```

子弹的移动范围

```
if (bullets[j] != NULL) {
    bullets[j]->setPosition(bullets[j]->getPositionX(), bullets[j]->getPositionY() + 5);
```

5、实现用触摸事件控制飞船的移动和子弹发射

触摸监听器

```
void Thunder::addTouchListener(){
    //
    auto touchListener = EventListenerTouchOneByOne::create();
    touchListener->onTouchBegan = CC_CALLBACK_2(Thunder::onTouchBegan, this);
    touchListener->onTouchMoved = CC_CALLBACK_2(Thunder::onTouchMoved, this);
    touchListener->onTouchEnded = CC_CALLBACK_2(Thunder::onTouchEnded, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(touchListener, player);
}
```

判断触摸点在不在 player 上

```
bool Thunder::onTouchBegan(Touch *touch, Event *unused_event) {
    //
    auto touchPos = touch->getLocation();
    auto playerPos = player->getPosition();
    auto playerSize = player->getContentSize();
    auto rect = Rect(playerPos.x - playerSize.width / 2, playerPos.y - playerSize.height / 2,
        playerSize.width, playerSize.height);
    if (rect.containsPoint(touchPos))
        return true;
    return false;
}
```

拖动飞机

```
void Thunder::onTouchMoved(Touch *touch, Event *unused_event) {
    //
    auto touchPos = touch->getLocation();
    player->setPosition(touchPos);
}
```

松开即开火

```
void Thunder::onTouchEnded(Touch *touch, Event *unused_event) {
    //
    fire();
}
```

6、 demo 只能同时存在一颗子弹，实现同时存在多颗子弹的功能

将子弹用 vector 存储

```
std::vector<Sprite*> bullets;
```

```
bullets.reserve(10);
```

调度器中更新子弹状态

```
for (int j = 0; j < bullets.size(); j++) {
    if (bullets[j] != NULL) {
        bullets[j]->setPosition(bullets[j]->getPositionX(), bullets[j]->getPositionY() + 5);
        if (bullets[j]->getPositionY() > visibleSize.height - 10) {
            bullets[j]->removeFromParentAndCleanup(true);
            bullets.erase(bullets.begin() + j);
            j--;
        }
    }
}
```

```
for (unsigned i = 0; i < enemys.size(); i++) {
    if (enemys[i] != NULL) {
        for (int j = 0; j < bullets.size(); j++) {
            if (bullets[j] != NULL && enemys[i] != NULL &&
                bullets[j]->getPosition().getDistance(enemys[i]->getPosition()) < 30) {
                bullets[j]->removeFromParentAndCleanup(true);
                bullets.erase(bullets.begin() + j);
                j--;
                EventCustom e("meet");
                e.setUserData(&i);
                _eventDispatcher->dispatchEvent(&e);
            }
        }
```

添加子弹

```
void Thunder::fire() {

    auto bullet = Sprite::create("bullet.png");
    bullet->setPosition(player->getPosition());
    addChild(bullet);
    bullets.insert(bullets.begin(), bullet);

    auto audio = SimpleAudioEngine::getInstance();
    audio->playEffect("music/fire.wav");
}
```

7、demo 中只实现了陨石左右移动，添加代码，使得陨石会在每次来回后向玩家移动一定距离。实现上述效果后，再用自定义事件等方式实现当陨石和玩家小于一定距离时，飞船爆炸，游戏失败

实现陨石的左右移动

```
static double count = 0;
static int dir = 1;
count += f;
if (count > 1) { count = 0.0; dir = (dir + 1) % 3; }
```

```
for (unsigned i = 0; i < enemys.size(); i++) {
    if (enemys[i] != NULL) {
        if (dir == 0)
            enemys[i]->setPosition(enemys[i]->getPosition() + Vec2(1, 0));
        else if (dir == 1)
            enemys[i]->setPosition(enemys[i]->getPosition() + Vec2(-1, 0));
        else
            enemys[i]->setPosition(enemys[i]->getPosition() + Vec2(0, -1));
```

实现当陨石和玩家小于一定距离时，飞船爆炸，游戏切换到 GameOver 的场景

```
    if (enemys[i] != NULL && player->getPosition().getDistance(enemys[i]->getPosition()) < 30) {
        EventCustom f("meet");
        f.setUserData(&i);
        _eventDispatcher->dispatchEvent(&f);
        _eventDispatcher->removeAllEventListeners();
        auto sence = Gameover::createScene();
        Director::getInstance()->replaceScene(sence);
    }
```

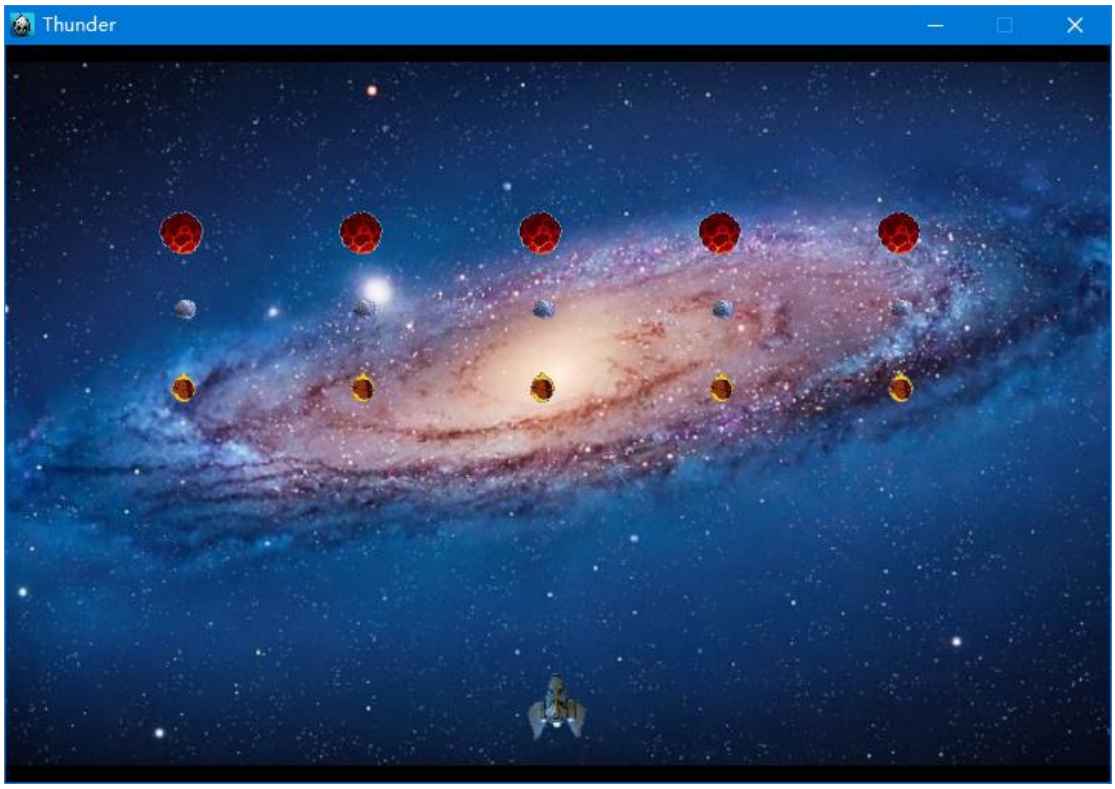
自定义 Gameover 场景

```
#pragma once
#ifndef _GAMEOVER_H_
#define _GAMEOVER_H_
#include "cocos2d.h"
#include "SimpleAudioEngine.h"
#include <vector>

USING_NS_CC;

class Gameover : public cocos2d::Layer
{
public:
    // there's no 'id' in cpp, so we recommend returning the class instance pointer
    static cocos2d::Scene* createScene();

    // Here's a difference. Method 'init' in cocos2d-x returns bool, instead of returning 'id' in cocos2d-iphone
    virtual bool init();

    // implement the "static create()" method manually
    CREATE_FUNC(Gameover);
};
```

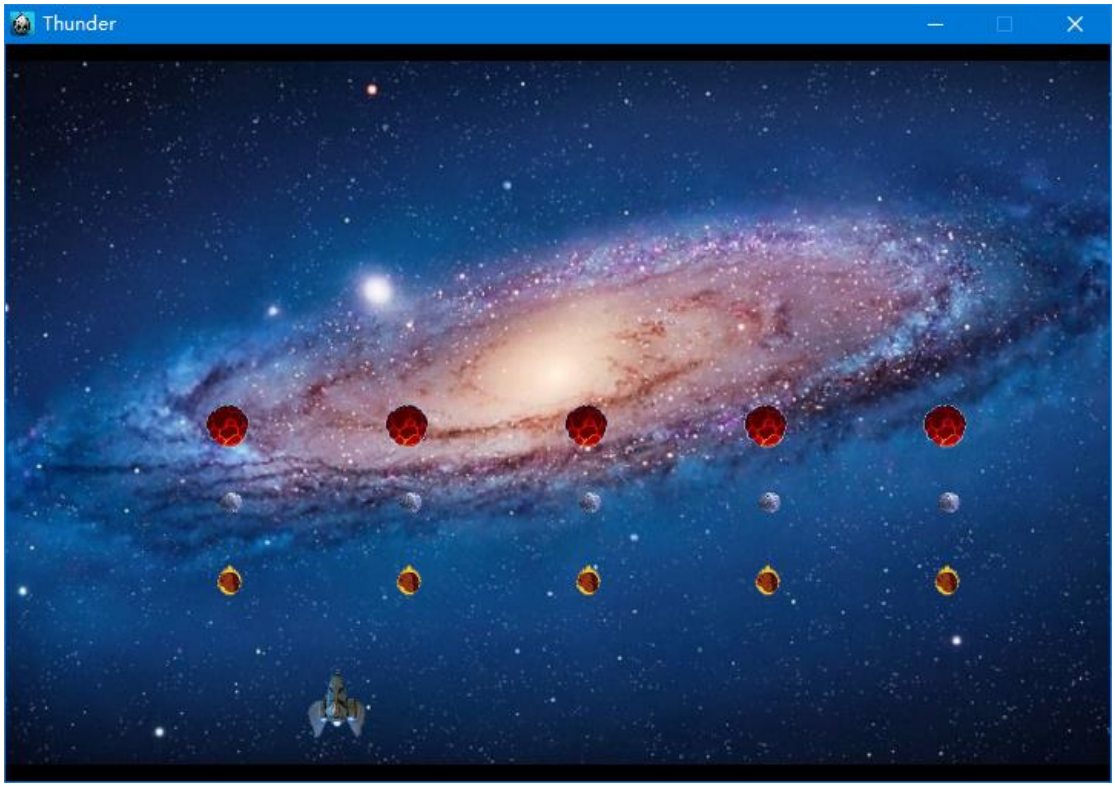
```
bool Gameover::init()
{
    if (!Layer::init())
        return false;
    Size visibleSize = Director::getInstance()->getVisibleSize();
    auto bg = Sprite::create("gameover.jpg");
    bg->setPosition(Vec2(visibleSize.width / 2, visibleSize.height / 2));
    addChild(bg, 0);
    auto startItem = MenuItemImage::create("start-0.png",
        "start-1.png",
        [](Ref* sender) {
        auto sence = Thunder::createScene();
        Director::getInstance()->replaceScene(sence);
    });
    startItem->setPosition(Vec2(visibleSize.width  - 180,  220));
    auto menu = Menu::create(startItem, NULL);
    menu->setPosition(Point(0, 0));
    this->addChild(menu, 1);

    return true;
}
```

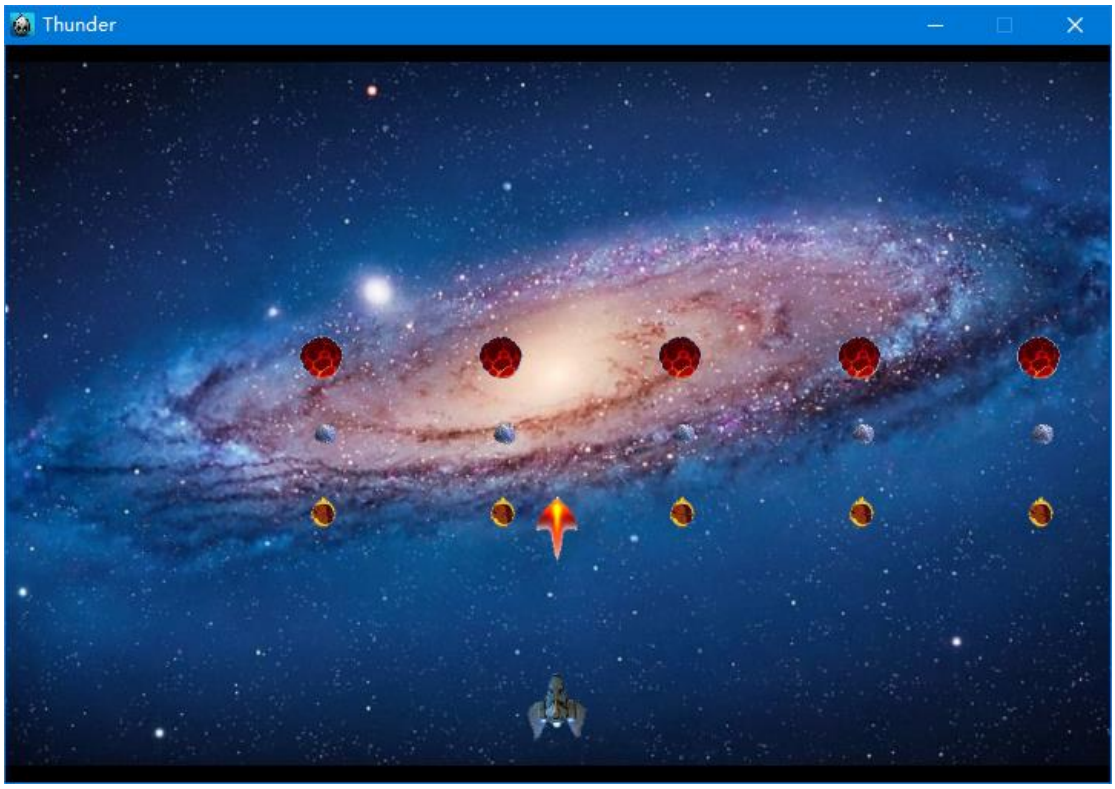
三．实验结果截图

运行界面：

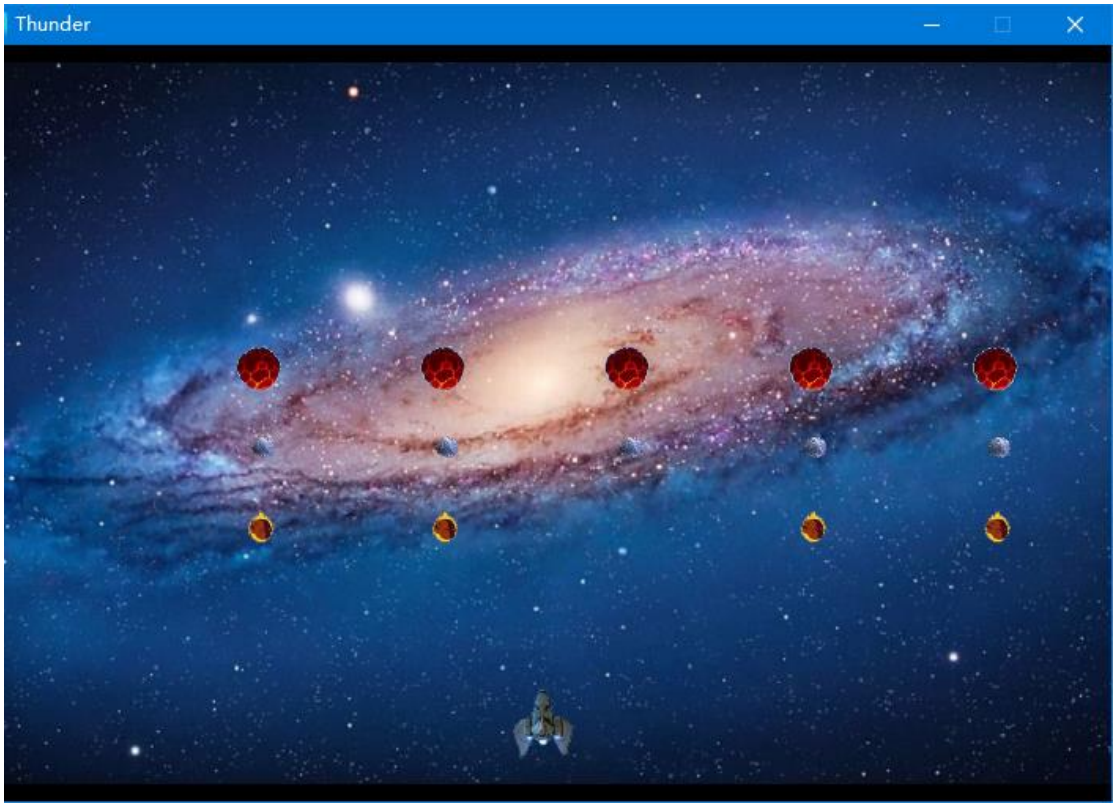


键盘控制

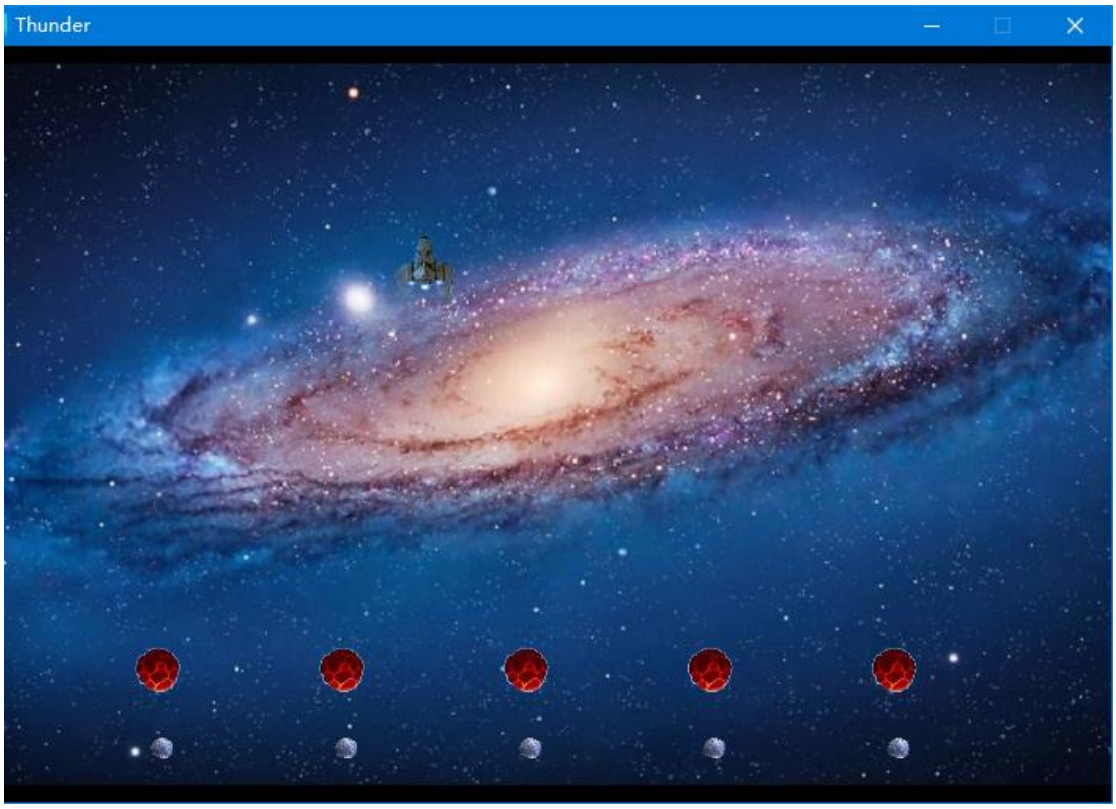




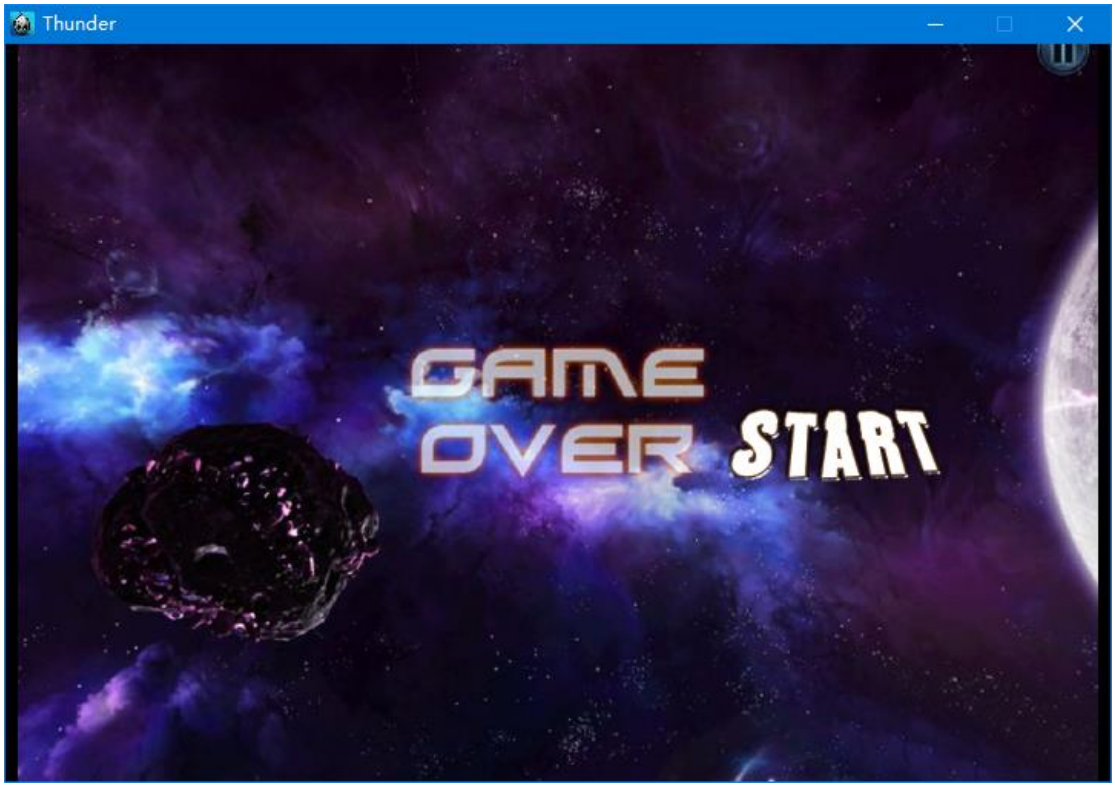
击中后



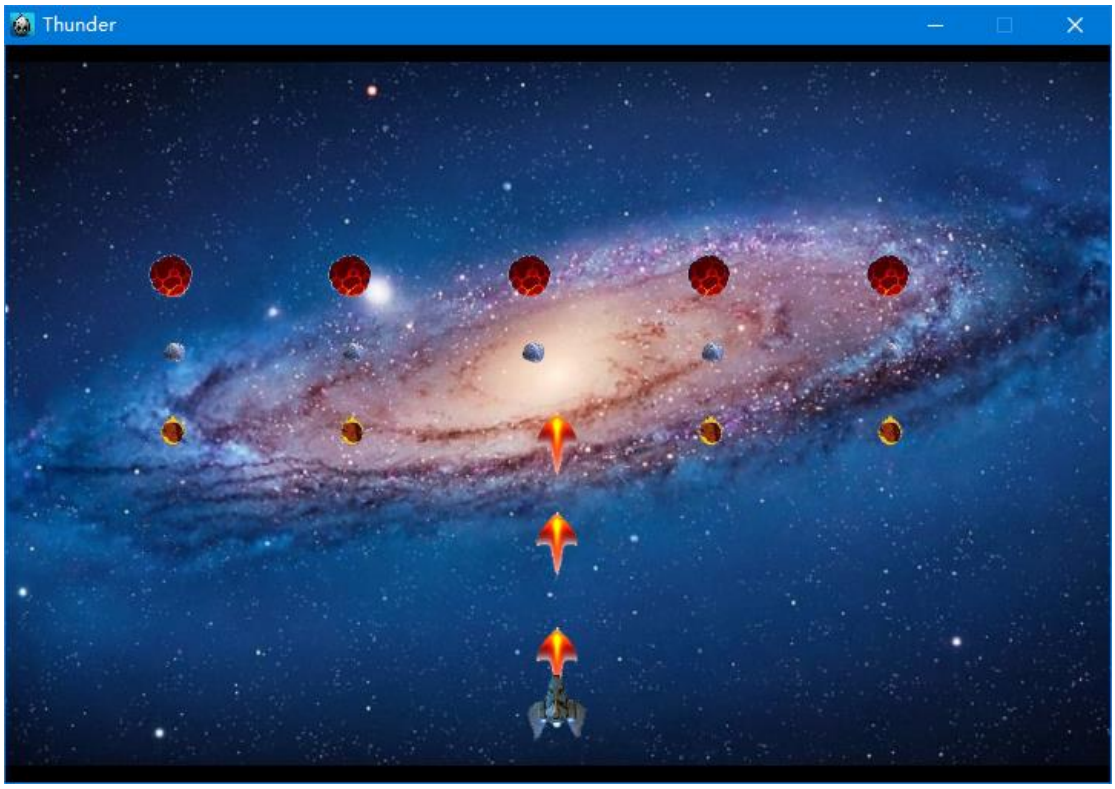
触摸控制



游戏结束



点击 start 重新开始，发射多枚子弹



四．实验过程遇到的问题

问题一：游戏运行时很卡

原因：刚开始游戏的特效音都是使用 playBackgroundMusic,这样占用资源很大，改为 playEffect 后解决问题

问题二：发射多枚子弹时偶尔内存会报错

原因：在两枚子弹几乎同时碰到怪物时，会出现怪物被删除而仍对怪物进行处理的情况，刚开始以为这样没有问题

```
if (enemys[i] != NULL) {
    if (dir == 0)
        enemys[i]->setPosition(enemys[i]->getPosition() + Vec2(1, 0));
    else if (dir == 1)
        enemys[i]->setPosition(enemys[i]->getPosition() + Vec2(-1, 0));
    else
        enemys[i]->setPosition(enemys[i]->getPosition() + Vec2(0, -1));
    for (int j = 0; j < bullets.size(); j++) {
        if (bullets[j] != NULL &&
            bullets[j]->getPosition().getDistance(enemys[i]->getPosition()) < 30) {
            bullets[j]->removeFromParentAndCleanup(true);
            bullets.erase(bullets.begin() + j);
            j--;
            EventCustom e("meet");
            e.setUserData(&i);
            _eventDispatcher->dispatchEvent(&e);
        }
```

但是只在 bullet 循环外面加 enemys[i] != NULL 是不够的，需要在 bullet 的循环也加入判断

```
    for (int j = 0; j < bullets.size(); j++) {
        if (bullets[j] != NULL && enemys[i] != NULL &&
            bullets[j]->getPosition().getDistance(enemys[i]->getPosition()) < 30) {
            bullets[j]->removeFromParentAndCleanup(true);
            bullets.erase(bullets.begin() + j);
            j--;
            EventCustom e("meet");
            e.setUserData(&i);
            _eventDispatcher->dispatchEvent(&e);
        }
```

问题三：场景切换中再切换到 Thunder 场景时，子弹一碰到 enemys 内存就报错

原因：使用 replaceScene 过程中没有先取消订阅消息，导致之前场景资源没有全部释放，

所以在切换场景之前将所有监听器删除就可以解决问题

```
if (enemys[i] != NULL && player->getPosition().getDistance(enemys[i]->getPosition()) < 30) {
    EventCustom f("meet");
    f.setUserData(&i);
    _eventDispatcher->dispatchEvent(&f);
    _eventDispatcher->removeAllEventListeners();
    auto sence = Gameover::createScene();
    Director::getInstance()->replaceScene(sence);
}
```

五．思考与总结

心得体会：

这周的 demo 看似不难，但是做的时候出现很多问题，而且还都是很隐蔽的问题，所以基本上把时间都花在如何解决前面提到的问题上了。

通过这次实验我总结出以下三点：

一：学习门技术绝对要有耐心，能坚持下去。

二：解决要学会利用多渠道，可以请教他人上网查询等总之达到高效化；

三：因为大部分参考资料是英文的，所以还要提高语水平

四：要提高自主学习能力，学会查看 API 文档