现代操作系统应用开发实验报告

学号： 14307004　　**班级　：** 15 软工 1 班

姓名： 蔡冠文　　**实验名称：** Homework13

一．参考资料

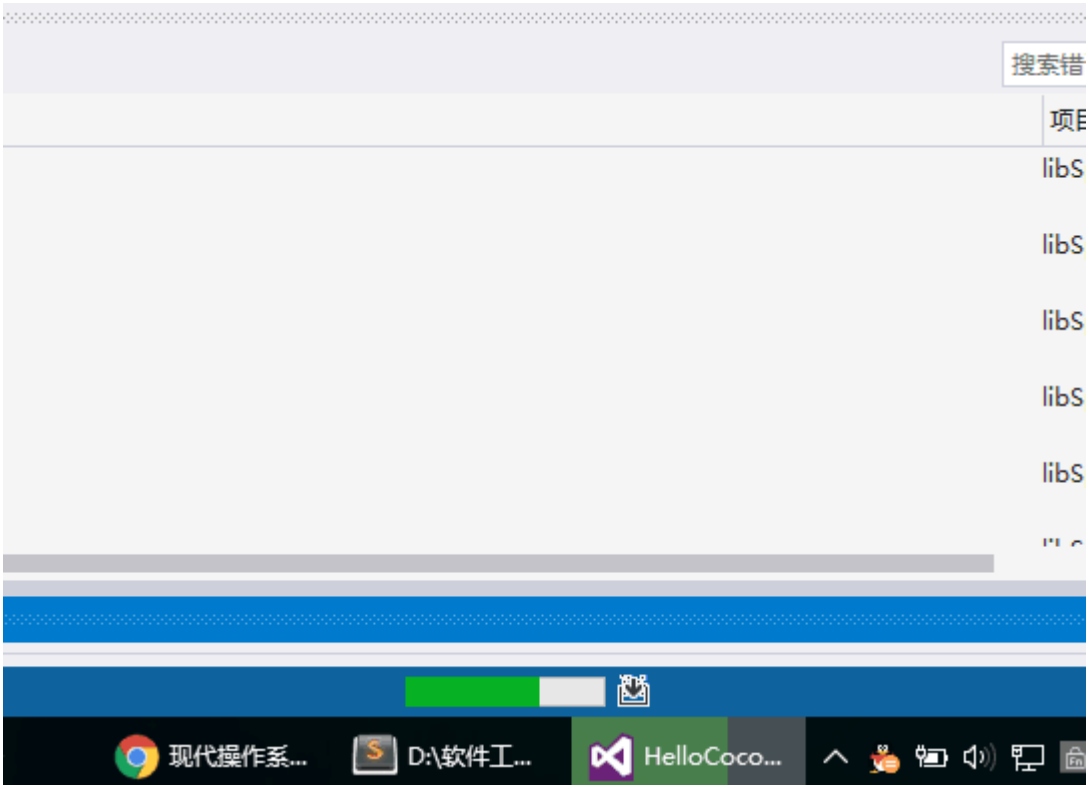
请在这里列出对本实验有帮助你所参考的资料或者网站。

课件，群聊记录，dalao 的指点

二．实验步骤

请在这里简要写下你的实验过程。

1. 生成解决方案



2. 仿照课件的方法预加载音乐文件，添加播放。

```
//预加载音乐文件
void Thunder::preloadMusic() {
    // Todo
    auto audio = SimpleAudioEngine::getInstance();
    audio->preloadBackgroundMusic("music/bgm.mp3");
    audio->preloadEffect("music/explore.wav");
    audio->preloadEffect("music/fire.wav");
}
```

3. 实现飞船的移动。movePlane 在 update 中调用，仿照上次作业改变坐标的方法写 update.

```
// 移动飞船
void Thunder::movePlane(char c) {
    // Todo
    auto currentPos = player->getPosition();
    MoveTo* moveTo;
    if (c == 'a' || c == 'A') {
        moveTo = MoveTo::create(0.5, Vec2(currentPos.x - 40, currentPos.y));
    }
    else if (c == 'd' || c == 'D') {
        moveTo = MoveTo::create(0.5, Vec2(currentPos.x + 40, currentPos.y));
    }
    else
    {}
    auto spawn = Spawn::create(moveTo, NULL);
    auto seq = Sequence::create(spawn, NULL);

    player->runAction(seq);
```

4. 添加发射子弹的动作。这里涉及到触摸事件监听器，仿照课件的方法，回调点击和松开两个事件，在点击的时候调用 fire()发射子弹

```
// 添加键盘事件监听器
void Thunder::addKeyboardListener() {
    // Todo
    auto keyboardListener = EventListenerKeyboard::create();
    auto keyboardListener2 = EventListenerKeyboard::create();
    keyboardListener->onKeyPressed = CC_CALLBACK_2(Thunder::onKeyPressed, this);
    keyboardListener2->onKeyReleased = CC_CALLBACK_2(Thunder::onKeyReleased, this);
    this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(keyboardListener, this);
    this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(keyboardListener2, this);
}
```

```
// 鼠标点击发射炮弹
bool Thunder::onTouchBegan(Touch *touch, Event *event) {
    isClick = true;
    fire();
  return true;
}

void Thunder::onTouchEnded(Touch *touch, Event *event) {
  isClick = false;
}
```

5. 添加自定义事件 meet。产生爆炸效果

```
void Thunder::meet(EventCustom * event) {
  // 判断子弹是否打中陨石并执行对应操作
  // Todo
    for (list<Sprite*>::iterator it1 = bullets.begin();it1!=bullets.end();)
    {
        bool isHit = false;
        for (auto &it2 :enemys)
        {
            Sprite* temp = it2;
            if ((*it1)->getPosition().getDistance(it2->getPosition())< 25)
            {
                it2->runAction(Sequence::create(Animate::create(
                    Animation::createWithSpriteFrames(explore,0.05f,1)),
                    CallFunc::create([temp] {
                    temp->removeFromParentAndCleanup(true);
                }), nullptr));
                isHit = true;
                enemys.remove(it2);
                break;
            }
        }
        if (isHit == true) {
            (*it1)->removeFromParentAndCleanup(true);
            it1 = bullets.erase(it1);
        }
        else it1++;
    }
```

三．实验结果截图

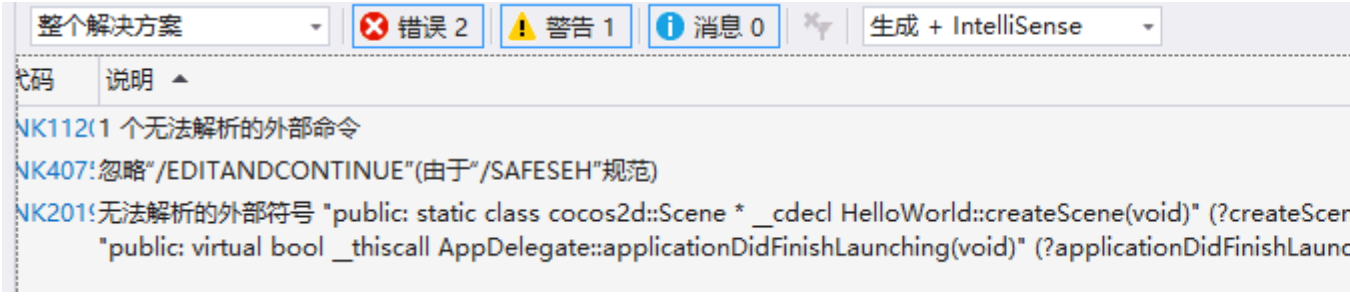
请在这里把实验所得的运行结果截图。

录了一个小视频，放在文件夹里

四．实验过程遇到的问题

请在这里写下你在实验过程中遇到的问题以及解决方案。

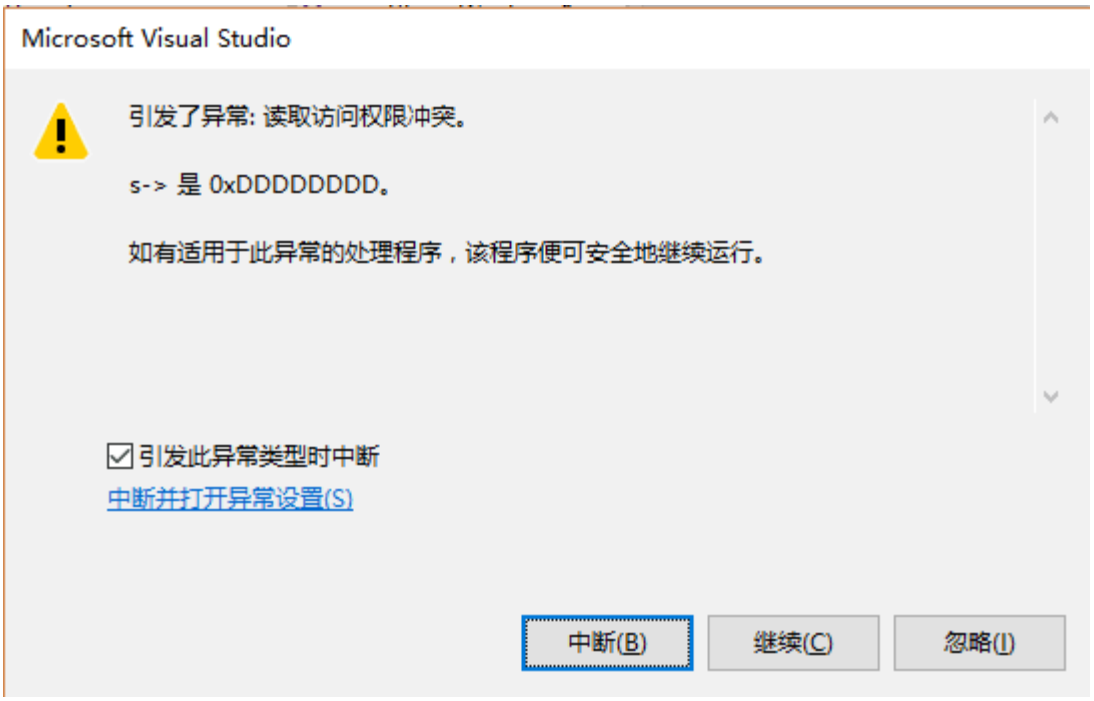
1. 编译错误

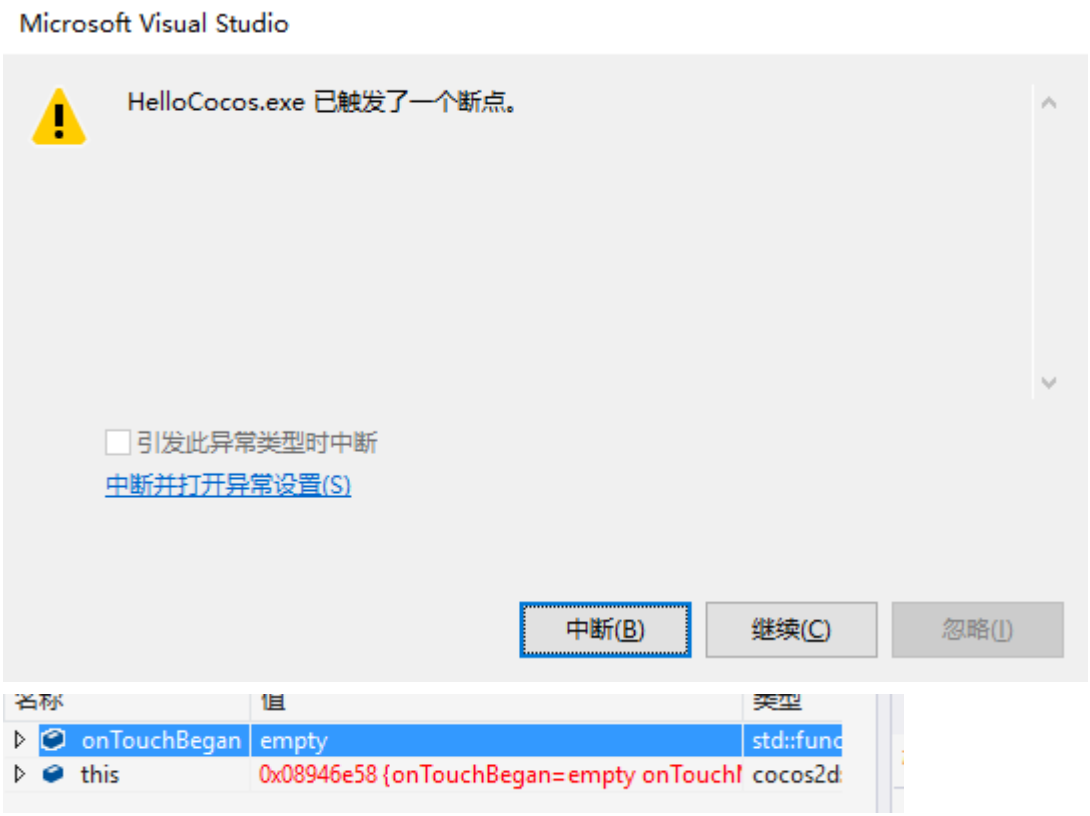


原因：将 helloWorldScene 和 thunder 一起生成运行

解决方案：从解决方案中删去了 helloWorldScene，重新编译生成解决方案

2.





问题：子弹碰到陨石就崩溃了，弹出报错

造成这两个报错的原因是类似的，没有对 list 的内容做好管理，仅仅是调用了 removefromparentandcleanup，而没有对 list 的内容进行 remove，所以访问了已删除的项。

解决方案：在 TA 大神的基础上，加一句

```
        }), nullptr));
        isHit = true;
        enemys.remove(it2);
        break;
    }
}
```

3. 子弹连续轰掉了几个陨石

原因：没有及时把子弹处理掉。需要注意的是，子弹 list 的遍历和陨石 list 的遍历稍有不同。陨石列表的遍历是内层循环，可以用 for(auto &)的形式，删除之后可以直接跳出循环。但是子弹列表在外循环，删除了子弹之后还要继续循环（不停发射子弹）。如果也采取这种方式，删除了之后就不能接着遍历了。所以这里用到了迭代器 iterator 负责遍历指向下

一位。

解决方案：

```
if (isHit == true) {
    (*it1)->removeFromParentAndCleanup(true);
    it1 = bullets.erase(it1);
}
else it1++;
```

五．思考与总结

请在这里写下你本次试验的心得体会以及所思所想。

这次实验做了一个类似于雷电的 2d 游戏。运用到的知识包括了 cocos2dx 中音频声效，还有事件处理。难点主要是事件处理，涉及到回调函数，lambda 表达式，事件监听器等的知识运用。

我觉得我对事件处理的理解还不算到位，基本上靠着课件和 dalao 在群里的聊天记录一点点磨出来的。下面说说我的理解，cocos2dx 对于游戏中输入输出事件的响应模式，采用了订阅者模式。订阅者模式的核心有两个，事件监听器和事件分发器。前者相当于订阅者的角色，封装了事件处理的代码，负责响应事件的处理。后者相当于发布者的角色，当事件发生时通知对应事件的监听器

在这次作业中，事件监听器包括触摸事件监听器（处理点击发射这个响应事件），键盘事件监听器（负责按键后的响应事件），自定义监听器（负责碰撞后的响应处理）。以飞船移动为例，过程大概如下：

点击 AD 实现飞船移动 -> 通知键盘事件监听器 keyboardListener –> 事件分发-> 响应事件处理的两份代码 KeyPressed 和 KeyRelease，改变了 isMove 的值 -> update 中根据 isMove 的值选择是否调用 move 函数。