现代操作系统应用开发实验报告

学号： 15331046　　**班级　　：** 晚上班

姓名： 陈志扬　　**实验名称：** HW3 Thunder

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。

资料：老师上课所使用的 PPT，TA 师兄提供的作业指导

网站：官方 API 文档 http://api.cocos.com/

按住鼠标拖动精灵参考博客：

http://m.blog.csdn.net/article/details?id=47083397

http://blog.csdn.net/luoyikun/article/details/48289979

STL list 链表的用法详细解析：

http://www.jb51.net/article/41525.htm 等等

二．实验步骤

请在这里简要写下你的实验过程。

① 首先，学习老师课件上的知识内容，主要是 cocos2d-x 事件处理和音效，事件处理即事件分发与响应，理解事件分发机制：事件监听器、分发器，编写事件响应函数。事件监听器主要分为 5 种类型：前四种类型有其对应的响应事件，自定义事件顾名思义是我们自己定义编写的事件。然后是音乐与音效，用到

SimpleAudioEngine 头文件和 cocos2d-x 自带的 CocosDension 库，可编写预加载、播放、暂停、停止的代码。

② 然后，根据 TA 师兄提供的 demo，按照作业要求理解一下逻辑，逐一编写各个函数。先实现“利用键盘事件实现飞船左右移动”的功能。这里师兄已经给出键盘事件分发器的编写，我们只需要添加键盘事件监听器的代码和键盘响应事件 movePlane 函数的编写即可。具体代码如下：

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

// 移动飞船
void Thunder::movePlane(char c) {
    switch (movekey) {
    case 'A':
        // 不让飞船移出左边界
        if (player->getPositionX() - 30 > Director::getInstance()->getVisibleOrigin().x)
            player->runAction(MoveBy::create(0.08, Vec2(-10, 0)));
        break;
    case 'D':
        // 不让飞船移出右边界
        if (player->getPositionX() + 30 < Director::getInstance()->getVisibleOrigin().x + visibleSize.width)
            player->runAction(MoveBy::create(0.08, Vec2(10, 0)));
        break;
    }
}
```

③ 先把音乐音效预加载，背景音乐加上，以便后面使用。

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

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

④ 切割爆炸动画帧

```
// 切割爆炸动画帧
void Thunder::explosion() {
    auto texture = Director::getInstance()->getTextureCache()->addImage("explosion.png");
    explore.reserve(8);
    for (int i = 0; i < 5; i++) {
        auto frame = SpriteFrame::createWithTexture(texture, CC_RECT_PIXELS_TO_POINTS(Rect(188.8 * i, 0, 188.8, 160)));
        explore.pushBack(frame);
    }
    for (int i = 5; i < 8; i++) {
        auto frame = SpriteFrame::createWithTexture(texture, CC_RECT_PIXELS_TO_POINTS(Rect(188.8 * (i - 5), 160, 188.8, 160)));
        explore.pushBack(frame);
    }
}
```

⑤ 接下来实现“利用键盘和触摸事件实现子弹飞射”的功能。代码如下，这里遇到一点小问题，留到第四部分再来简单谈谈。

```
// 添加触摸事件监听器
void Thunder::addTouchListener() {
    auto touchListener = EventListenerTouchOneByOne::create();
    touchListener->onTouchBegan = CC_CALLBACK_2(Thunder::onTouchBegan, this);
    touchListener->onTouchEnded = CC_CALLBACK_2(Thunder::onTouchEnded, this);
    touchListener->onTouchMoved = CC_CALLBACK_2(Thunder::onTouchMoved, this);
    this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(touchListener, this);
}

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

调用fire函数即可发射

```
//发射子弹
void Thunder::fire() {
    auto bullet = Sprite::create("bullet.png");
    bullet->setAnchorPoint(Vec2(0.5, 0.5));
    bullets.push_back(bullet);
    bullet->setPosition(player->getPosition());
    addChild(bullet, 1);
    SimpleAudioEngine::getInstance()->playEffect("music/fire.wav", false);

    // 移除飞出屏幕外的子弹
    bullet->runAction(
        Sequence::create(
            MoveBy::create(1.0f, Vec2(0, visibleSize.height)),
            CallFuncN::create(CC_CALLBACK_1(Thunder::removeBullet, this)),
            nullptr
        )
    );
}

// 移除子弹
void Thunder::removeBullet(Node* bullet) {
    bullet->removeFromParentAndCleanup(true);
    bullets.remove((Sprite*)bullet);
}
```

⑥ 接下来，用自定义事件实现：子弹和陨石相距小于一定距离时，陨石爆炸，子弹消失。

```
// 分发自定义事件
EventCustom e("meet");
_eventDispatcher->dispatchEvent(&e);
```

```cpp
// 自定义碰撞事件
void Thunder::meet(EventCustom * event) {
    // 判断子弹是否打中陨石并执行对应操作
    for (list<Sprite*>::iterator b = bullets.begin(); b != bullets.end(); ) {
        bool isHit = false;// 标记打中
        for (auto enemy : enemys) {
            if ((*b)->getPosition().getDistance(enemy->getPosition()) < 25) {
                Sprite* temp = enemy;
                enemy->runAction(
                    Sequence::create(
                        Animate::create(Animation::createWithSpriteFrames(explore, 0.05f, 1)),
                        CallFunc::create([temp] {
                            temp->removeFromParentAndCleanup(true);
                        }),
                        nullptr
                    )
                );
                isHit = true;// 置为True
                SimpleAudioEngine::getInstance()->playEffect("music/explore.wav", false);
                enemys.remove(enemy);
                break;
            }
        }
        // true，打中，移除子弹
        if (isHit == true) {
            (*b)->removeFromParentAndCleanup(true);
            b = bullets.erase(b);
        }
        else {
            b++;
        }
    }
```

⑦ 加分项 1：利用触摸事件实现飞船移动（点击飞船后拖动鼠标）

```cpp
// 当鼠标按住飞船后可控制飞船移动 (加分项)
void Thunder::onTouchMoved(Touch *touch, Event *event) {
    Vec2 delta = touch->getDelta();
    double playerPositionX = player->getPositionX();
    double playerPositionY = player->getPositionY();
    double newPositionX = playerPositionX + delta.x;
    Vec2 newPosition = Vec2(newPositionX, playerPositionY);// 注意playerPositionY，表明飞船只能在水平方向移动
    // 控制飞船移动范围
    if (newPositionX - 30 > Director::getInstance()->getVisibleOrigin().x
        && newPositionX + 30 < Director::getInstance()->getVisibleOrigin().x + visibleSize.width) {
        player->setPosition(newPosition);
    }
}
```

⑧ 加分项 2：陨石向下移动并生成新的一行陨石

```
// 陨石向下移动并生成新的一行(加分项)
void Thunder::newEnemy() {
    // 遍历enemys，若不空，则向下移一行
    for (auto enemy : enemys) {
        if (enemy != nullptr)
            enemy->setPosition(enemy->getPosition() + Vec2(0, -50));
    }
    char enemyPath[20];
    sprintf(enemyPath, "stone%d.png", stoneType + 1);// 选择stone类型
    stoneType = (stoneType + 1) % 3;// 1,2,3循环
    double width = visibleSize.width / 6.0,
        height = visibleSize.height - 50;
    for (int j = 0; j < 5; ++j) {
        auto enemy = Sprite::create(enemyPath);
        enemy->setAnchorPoint(Vec2(0.5, 0.5));
        enemy->setScale(0.5, 0.5);
        enemy->setPosition(width * (j + 1) - 80, height);
        enemys.push_back(enemy);
        addChild(enemy, 1);
    }
}
```

⑨ 至于加分项 3，其实并不需要添加多少代码，只要注意在移除子弹或者陨石爆炸的时候注意 remove 或者 erase，然后在新增一行陨石的时候记得 push_back 就可以了。

⑩ 其实至少得做加分项 2 才有游戏结束的可能（不做加分项 2 的话，打完所有陨石就 gameover），所以这部分我是留到最后才做的。

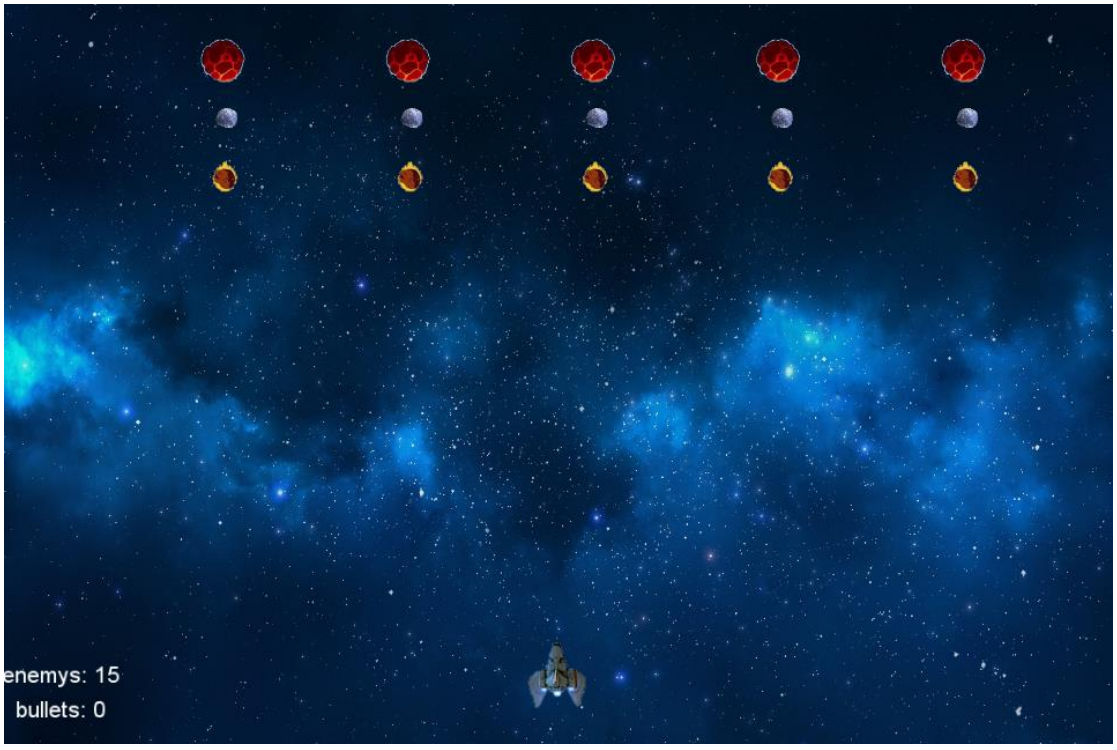
```
// 判断游戏是否结束并执行对应操作
for (auto enemy : enemys) {
    if (enemy->getPositionY() < 100) {
        auto gameOver = Sprite::create("gameOver.png");
        gameOver->setAnchorPoint(Vec2(0.5, 0.5));
        gameOver->setPosition(visibleSize.width / 2, visibleSize.height / 2);
        this->addChild(gameOver, 2);

        Sprite* temp = player;
        player->runAction(
            Sequence::create(
                Animate::create(Animation::createWithSpriteFrames(explore, 0.05f, 1)),
                CallFunc::create([temp] {
                    temp->removeFromParentAndCleanup(true);
                }),
                nullptr
            )
        );
        unschedule(schedule_selector(Thunder::update));// 停止定时器
        this->getEventDispatcher()->removeAllEventListeners();// 移除监听器
    }
}
```

三．实验结果截图

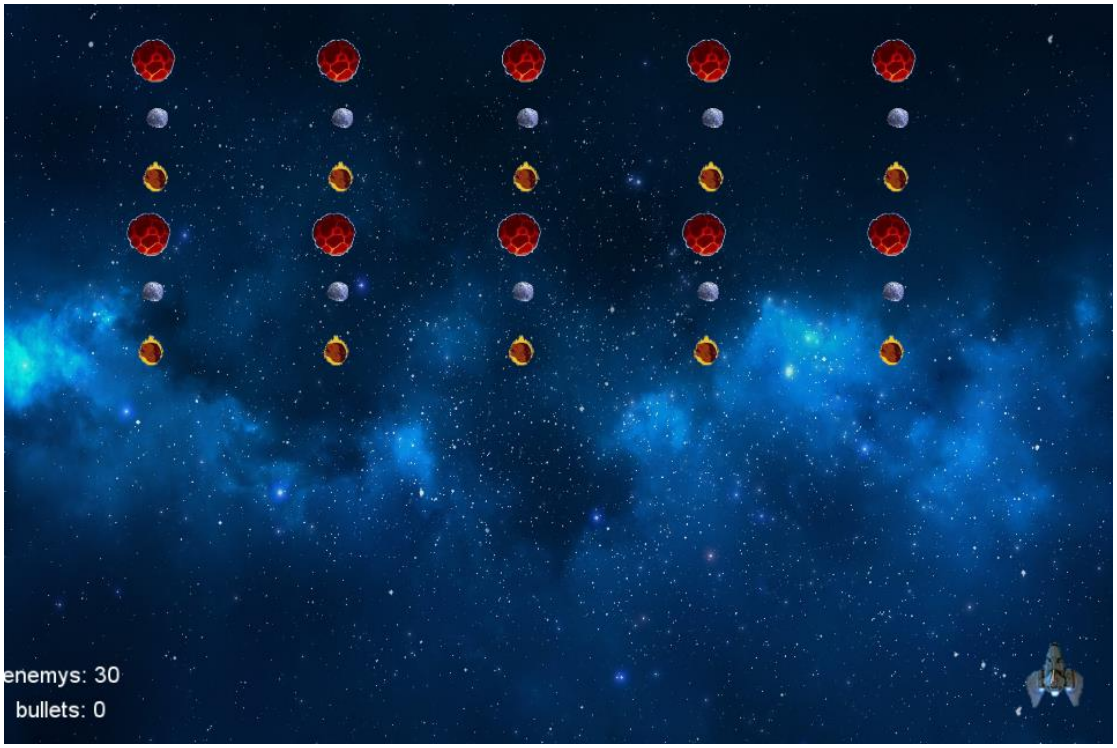
请在这里把实验所得的运行结果截图。

① 游戏运行起始界面：

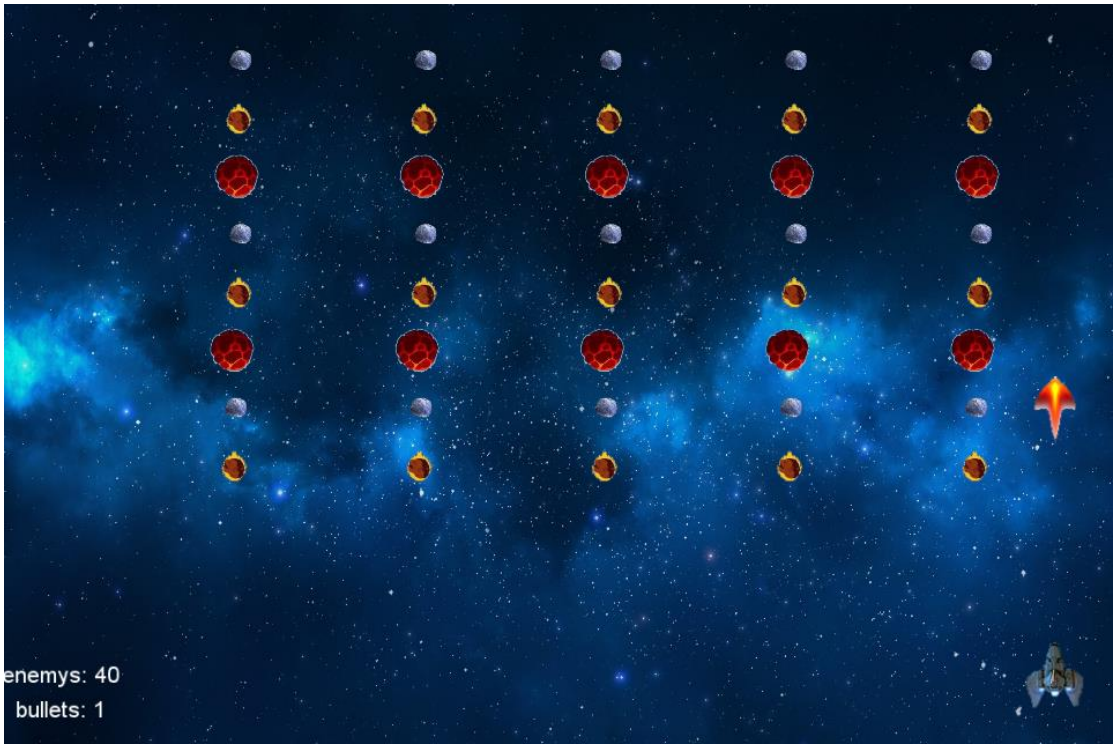


② 测试键盘事件（A、D、左箭头、右箭头）移动飞船，飞船不会

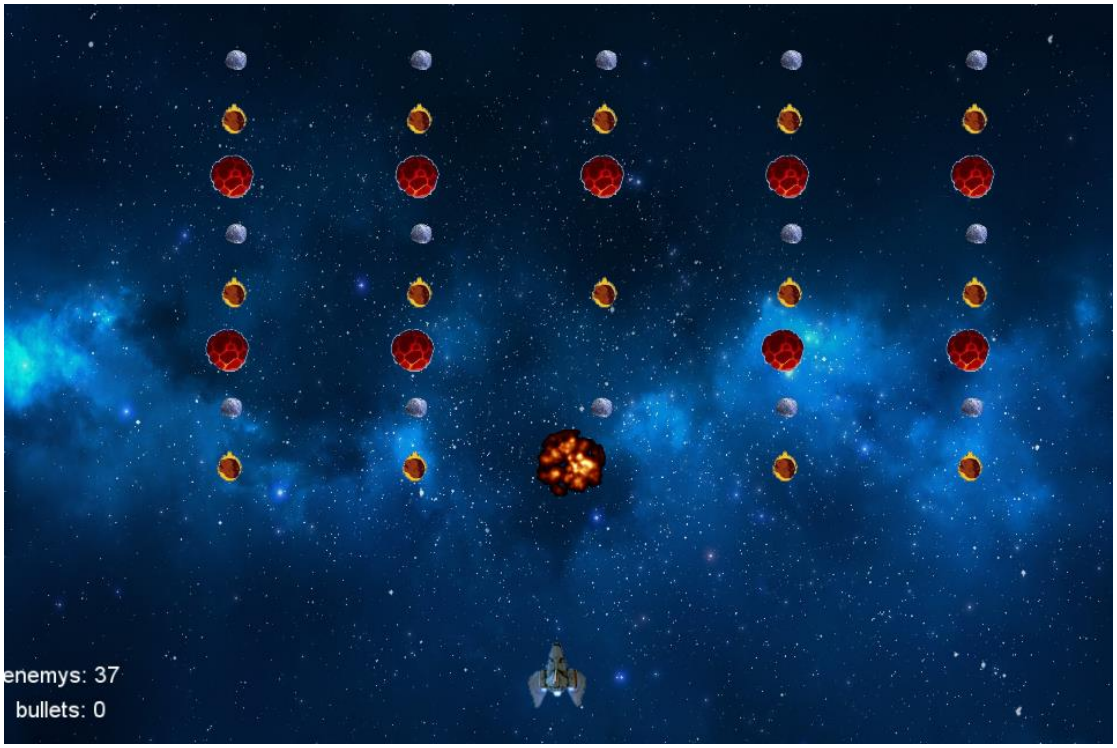
移出边界，该功能实现



③ 测试键盘事件（空格）和触摸事件（点击屏幕），可实现发送子弹，子弹飞出，移除子弹（包括事件方法）



④ 当子弹与陨石小于一定距离时，陨石爆炸，子弹消失



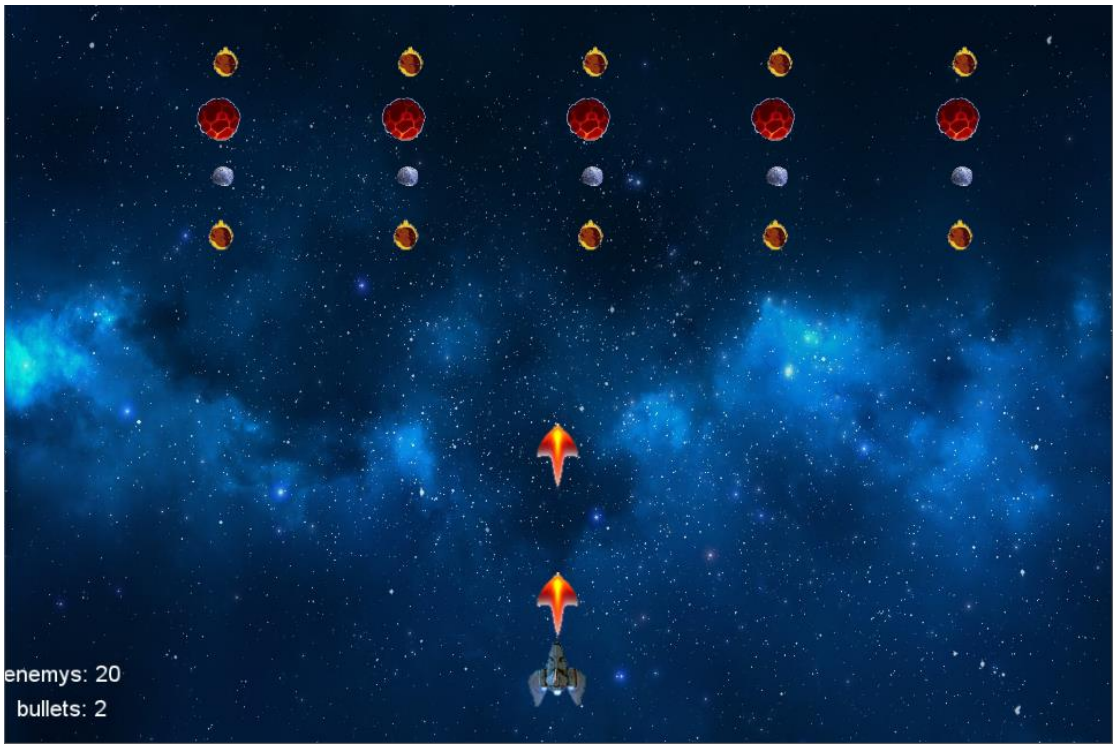
⑤ 当陨石落到底部，游戏结束，移除飞船，移除所有监听器

⑥ 加分项 1：利用触摸事件实现飞船移动（点击飞船后拖动鼠标），无法截图，略过，请 TA 谅解。

⑦ 加分项 2：陨石向下移动并生成新的一行陨石，由上面的截图

可知该功能已实现

⑧ 加分项 3：左下方 enemys 和 bullets 显示正确



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 发射子弹后移除飞出屏幕外的子弹遇到的问题：

removeFromParentAndCleanup 函数和 remove 函数调用位置，一开始我写在 fire()函数里，明知道这样一发射就会被移除掉，所以把这两个方法放在一个 Sequence 里面，但是好像无法直接使用，最后上网查资料，利用 CallFuncN 和 CC_CALLBACK_1 可实现

```
static cocos2d::CallFuncN *cocos2d::CallFuncN::create(const std::function<void (cocos2d::Node *)> &func)
* Creates the action with the callback of type std::function<void()>.
 This is the preferred way to create the callback.
 * @param func A callback function need to be executed.
 * @return An autoreleased CallFuncN object.

+ 1 重载
```

代码如下：

```
//发射子弹
void Thunder::fire() {
    auto bullet = Sprite::create("bullet.png");
    bullet->setAnchorPoint(Vec2(0.5, 0.5));
    bullets.push_back(bullet);
    bullet->setPosition(player->getPosition());
    addChild(bullet, 1);
    SimpleAudioEngine::getInstance()->playEffect("music/fire.wav", false);

    // 移除飞出屏幕外的子弹
    bullet->runAction(
        Sequence::create(
            MoveBy::create(1.0f, Vec2(0, visibleSize.height)),
            CallFuncN::create(CC_CALLBACK_1(Thunder::removeBullet, this)),
            nullptr
        )
    );
}

// 移除子弹
void Thunder::removeBullet(Node* bullet) {
    bullet->removeFromParentAndCleanup(true);
    bullets.remove((Sprite*)bullet);
}
```

② 自定义碰撞事件：判断子弹是否打中陨石并执行对应操作（陨石爆炸、子弹消失），采用遍历子弹和陨石的方法，根据设定的两者距离判断是否碰撞，这里主要是 list 链表的操作不太熟悉，导致 remove 或者 erase 后访问为 null 的问题。还有一开始我利用 continue，当碰撞后 continue 跳出内层循环但是还是会出现 list 访问问题，而 cocos2d-x 的错误信息输出感觉很不友好，跳到某个代码文件看都看不懂它提示什么问题出错，这里浪费了很多时间。最后还是添加了一个陨石被打中的标记，利用 break 来实现。具体代码如下：

```
// 判断子弹是否打中陨石并执行对应操作
for (list<Sprite*>::iterator b = bullets.begin(); b != bullets.end(); ) {
    bool isHit = false;// 标记打中
    for (auto enemy : enemys) {
        if ((*b)->getPosition().getDistance(enemy->getPosition()) < 25) {
            Sprite* temp = enemy;
            enemy->runAction(
                Sequence::create(
                    Animate::create(Animation::createWithSpriteFrames(explore, 0.05f, 1)),
                    CallFunc::create([temp] {
                        temp->removeFromParentAndCleanup(true);
                    }),
                    nullptr
                )
            );
            isHit = true;// 置为True
            SimpleAudioEngine::getInstance()->playEffect("music/explore.wav", false);
            enemys.remove(enemy);
            break;
        }
    }
    // true，打中，移除子弹
    if (isHit == true) {
        (*b)->removeFromParentAndCleanup(true);
        b = bullets.erase(b);
    }
    else {
        b++;
    }
}
```

③ 加分项 1：利用触摸事件实现飞船移动（点击飞船后拖动鼠标），这部分主要问题是无法像 demo 一样只有点击飞船才能拖动，我尝试过 containsPoint 方法，判断 touch 初始位置是否包含在 player->getBoundingBox() 里面，但是那样做的话会出现丢失触摸位置，需要重新点击才能拖动，所以我放弃用这个方法。下面的代码是点击屏幕任意一处即可拖动飞船移动。

```
// 当鼠标按住飞船后可控制飞船移动 (加分项)
void Thunder::onTouchMoved(Touch *touch, Event *event) {
    Vec2 delta = touch->getDelta();
    double playerPositionX = player->getPositionX();
    double playerPositionY = player->getPositionY();
    double newPositionX = playerPositionX + delta.x;
    Vec2 newPosition = Vec2(newPositionX, playerPositionY);// 注意playerPositionY，表明飞船只能在水平方向移动
    // 控制飞船移动范围
    if (newPositionX - 30 > Director::getInstance()->getVisibleOrigin().x
        && newPositionX + 30 < Director::getInstance()->getVisibleOrigin().x + visibleSize.width) {
        player->setPosition(newPosition);
    }
}
```

④ 加分项 2：陨石向下移动并生成新的一行陨石，这里遇到的问

题是怎么按 1、2、3 的顺序读取图片创建陨石精灵的问题，我想在不添加一个私有成员变量的情况下实现，但综合考虑还是添加了一个私有成员变量 stoneType。

```
int stoneType;// stone的类型，在生成新的一行时选择（1,2,3）

 stoneType = 0;// 在生成新的一行时作为选择标记（1,2,3）

// 陨石向下移动并生成新的一行(加分项)
void Thunder::newEnemy() {
    // 遍历enemys，若不空，则向下移一行
    for (auto enemy : enemys) {
        if (enemy != nullptr)
            enemy->setPosition(enemy->getPosition() + Vec2(0, -50));
    }
    char enemyPath[20];
    sprintf(enemyPath, "stone%d.png", stoneType + 1);// 选择stone类型
    stoneType = (stoneType + 1) % 3;// 1,2,3循环
    double width = visibleSize.width / 6.0,
        height = visibleSize.height - 50;
    for (int j = 0; j < 5; ++j) {
        auto enemy = Sprite::create(enemyPath);
        enemy->setAnchorPoint(Vec2(0.5, 0.5));
        enemy->setScale(0.5, 0.5);
        enemy->setPosition(width * (j + 1) - 80, height);
        enemys.push_back(enemy);
        addChild(enemy, 1);
    }
}
```

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。

这次实验的代码逻辑很清楚，TA 师兄声明了各个方法以及私有成员，为我们整个作业的完成提供了一个合理的框架，大大节省了我们的时间。

这次实验主要是学习事件分发与响应，通过事件监听器、分发器以及事件响应函数来实现一些交互功能；另外也学习了背景音乐音效的知识；在本次实验中，还涉及到数据结构 list 链表的操作，特别要注意在 remove 和 erase 后为 null 的问题，避免入坑。

这次实验 TA 师兄也教我们将 Cocos2d-x 项目代码打包成.exe 文件，我自己在经过一番摸索后也写了个教程，希望能给大家一点帮助。博客链接：利用 WinRAR 的自解压格式将 Cocos2d-x 项目代码打包成.exe 文件

这周已经是十三周的作业了，还有两三周，继续加油，最后是期末项目，Fighting！