现代操作系统应用开发实验报告

学号：153311117　　**班级　　：**教务 2 班

姓名：黄楠绚　　**实验名称：**homework

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。

cocos2d-x 中该如何实现键盘的按住事件？：

https://segmentfault.com/q/1010000000578296

二．实验步骤

1. 利用键盘事件实现飞船左右移动：

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);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(keyboardListener, this);   // 注册分发器
}
```

根据左右键移动飞船：

```
void Thunder::onKeyPressed(EventKeyboard::KeyCode code, Event* event) {
  switch (code) {
  case EventKeyboard::KeyCode::KEY_LEFT_ARROW:
  case EventKeyboard::KeyCode::KEY_CAPITAL_A:
  case EventKeyboard::KeyCode::KEY_A:
    movekey = 'A';
    isMove = true;
    break;
  case EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
  case EventKeyboard::KeyCode::KEY_CAPITAL_D:
  case EventKeyboard::KeyCode::KEY_D:
    movekey = 'D';
    isMove = true;
    break;
```

```cpp
void Thunder::movePlane(char c) {
    if (isMove == true) {
        switch (movekey) {
        case 'A':
            if (player->getPosition().x - 10 > origin.x) {
                auto move_by = MoveBy::create(0.04f, Vec2(-10, 0));
                player->runAction(move_by);
            }
            break;
        case 'D':
            if (player->getPosition().x + 10 < origin.x + visibleSize.width) {
                auto move_by = MoveBy::create(0.04f, Vec2(10, 0));
                player->runAction(move_by);
            }
            break;
        }
    }
}
```

2.利用键盘和触摸事件实现子弹发射。

触摸事件发射子弹：

```cpp
// 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->onTouchMoved = CC_CALLBACK_2(Thunder::onTouchMoved, this);
    touchListener->onTouchEnded = CC_CALLBACK_2(Thunder::onTouchEnded, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(touchListener, player);
}
```

```cpp
bool Thunder::onTouchBegan(Touch *touch, Event *event) {
    fire();
```

键盘事件发射子弹：

在 onKeyPressed 函数中：

```cpp
case EventKeyboard::KeyCode::KEY_SPACE:
    fire();
    break;
```

实现发射子弹：

加入回调函数在子弹飞出边界后自动移除：

```cpp
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(this, callfuncN_selector(Thunder::remove_bullet)),
            nullptr
        )
    );
}
```

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

3.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。

meet 函数中：

```
for (list<Sprite*>::iterator itor = bullets.begin(); itor != bullets.end(); ) {
    bool is_explore = false; // 判断是否相撞
    for (list<Sprite*>::iterator e_itor = enemys.begin(); e_itor != enemys.end(); e_itor++) {
        if ((*itor)->getPosition().getDistance((*e_itor)->getPosition()) < 25) {
            // 陨石爆炸
            Sprite *enemy = (*e_itor);
            enemy->runAction(
                Sequence::create(
                    Animate::create(
                        Animation::createWithSpriteFrames(explore, 0.05, 1)
                    ),
                    CallFunc::create([enemy] {
                        enemy->removeFromParentAndCleanup(true);
                    }),
                    nullptr
                )
            );
        );
        // 移除陨石
        enemys.erase(e_itor);
        // 播放爆炸音效
        SimpleAudioEngine::getInstance()->playEffect("music/explore.wav", false);
        is_explore = true;
        break;
    }
}
// 若陨石爆炸则移除该子弹
if (is_explore == true) {
    (*itor)->removeFromParentAndCleanup(true);
    itor = bullets.erase(itor);
}
else {
    itor++;
}
```

4.游戏过程中有背景音乐，发射子弹、击中陨石有音效。

背景音乐：

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

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

发射子弹、击中陨石音效已在前面的代码有提到

5.注意飞船、子弹的移动范围：

子弹利用回调函数在飞出边界后自动移除；飞船移动范围在一个范围内；均在前面代码有提过

6.游戏结束飞船爆炸，移除所有监听器

```
for (list<Sprite*>::iterator itor = enemys.begin(); itor != enemys.end(); itor++) {
    // 陨石位置变化
    if ((*itor)->getPosition().y < 50) {
        // 出现game over
        auto game_over = Sprite::create("gameOver.png");
        game_over->setPosition(Vec2(origin.x + visibleSize.width / 2,
                                    origin.y + visibleSize.height / 2));
        this->addChild(game_over, 2);
        // 飞船爆炸
        Sprite *temp = player;
        player->runAction(
            Sequence::create(
                Animate::create(
                    Animation::createWithSpriteFrames(explore, 0.05, 1)
                ),
                CallFunc::create([temp] {
                    temp->removeFromParentAndCleanup(true);
                }),
                nullptr
            )
        );
        unschedule(schedule_selector(Thunder::update));  // 取消调度器
        _eventDispatcher->removeAllEventListeners();     // 移除监听器
    }
}
```

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）

```
// 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) {
    auto target = static_cast<Sprite*>(event->getCurrentTarget());
    Vec2 delta = touch->getDelta();
    Vec2 currentPosition = target->getPosition();
    Vec2 newPosition = Vec2(delta.x + currentPosition.x, currentPosition.y);
    if (newPosition.x > origin.x && newPosition.x < origin.x + visibleSize.width)
        target->setPosition(newPosition);
    }
}
```

2. 陨石向下移动并生成新的一行陨石

增加一个类成员变量：

```
int enemy_type; // 用于增加新陨石，不同种类的陨石
```

```
// 陨石向下移动并生成新的一行(加分项)
void Thunder::newEnemy() {
    // 遍历所有的陨石并让它们移动
    for (Sprite* s : enemys) {
        if (s != NULL) {
            s->setPosition(s->getPosition() + Vec2(0, -50));
        }
    }
    // 生成新陨石
    char enemyPath[20];
    sprintf(enemyPath, "stone%d.png", enemy_type + 1);
    double width = visibleSize.width / (5 + 1.0), 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) - 84, height);
        enemys.push_back(enemy);
        addChild(enemy, 1);
    }
    enemy_type = (enemy_type + 1) % 3;
}
```

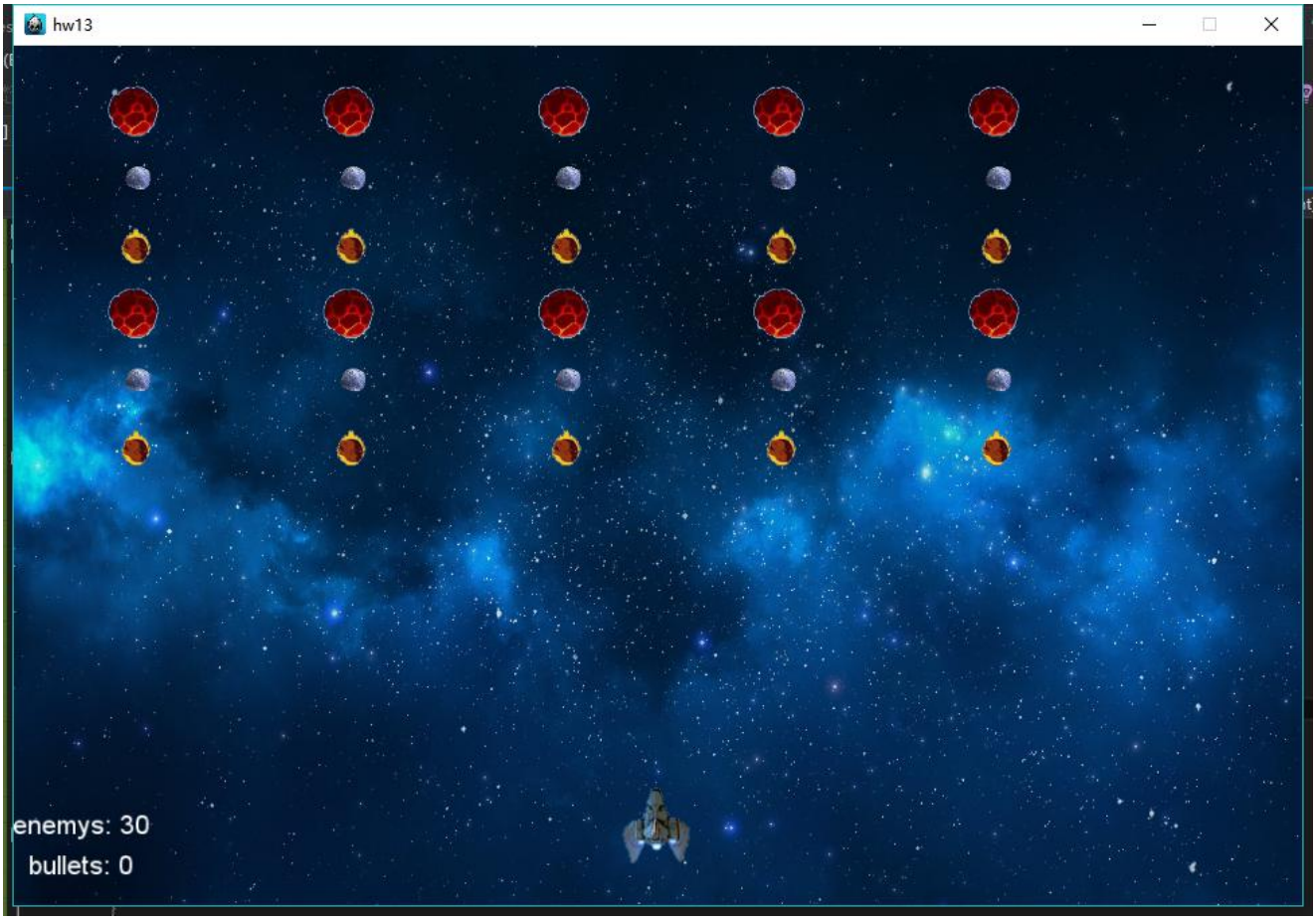
3. 子弹和陨石的数量显示正确

在前面的代码中已经实现了子弹发射、飞出，和陨石碰撞或产生后，左下角的 label 显示的数字均正确

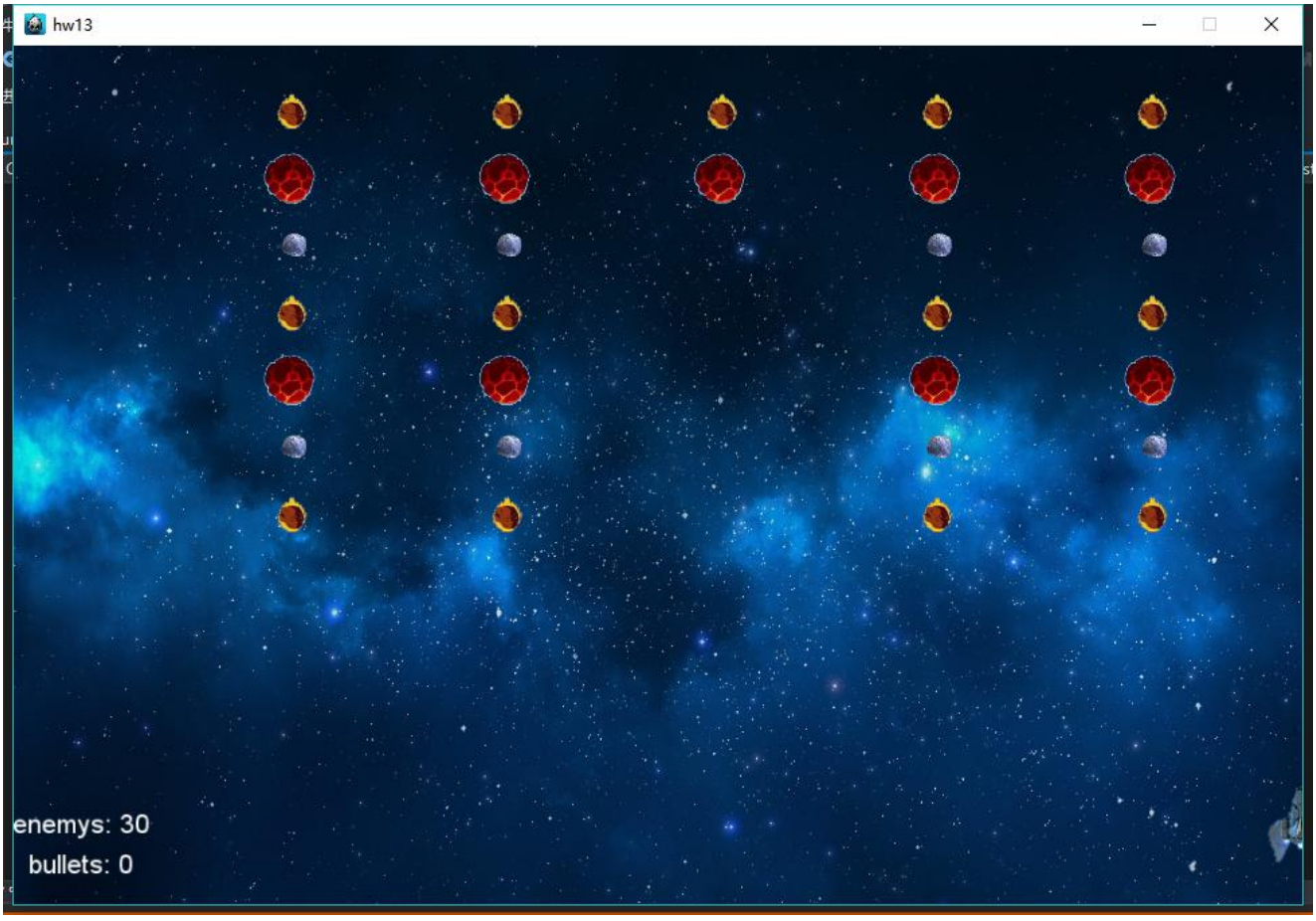
三．实验结果截图

请在这里把实验所得的运行结果截图。

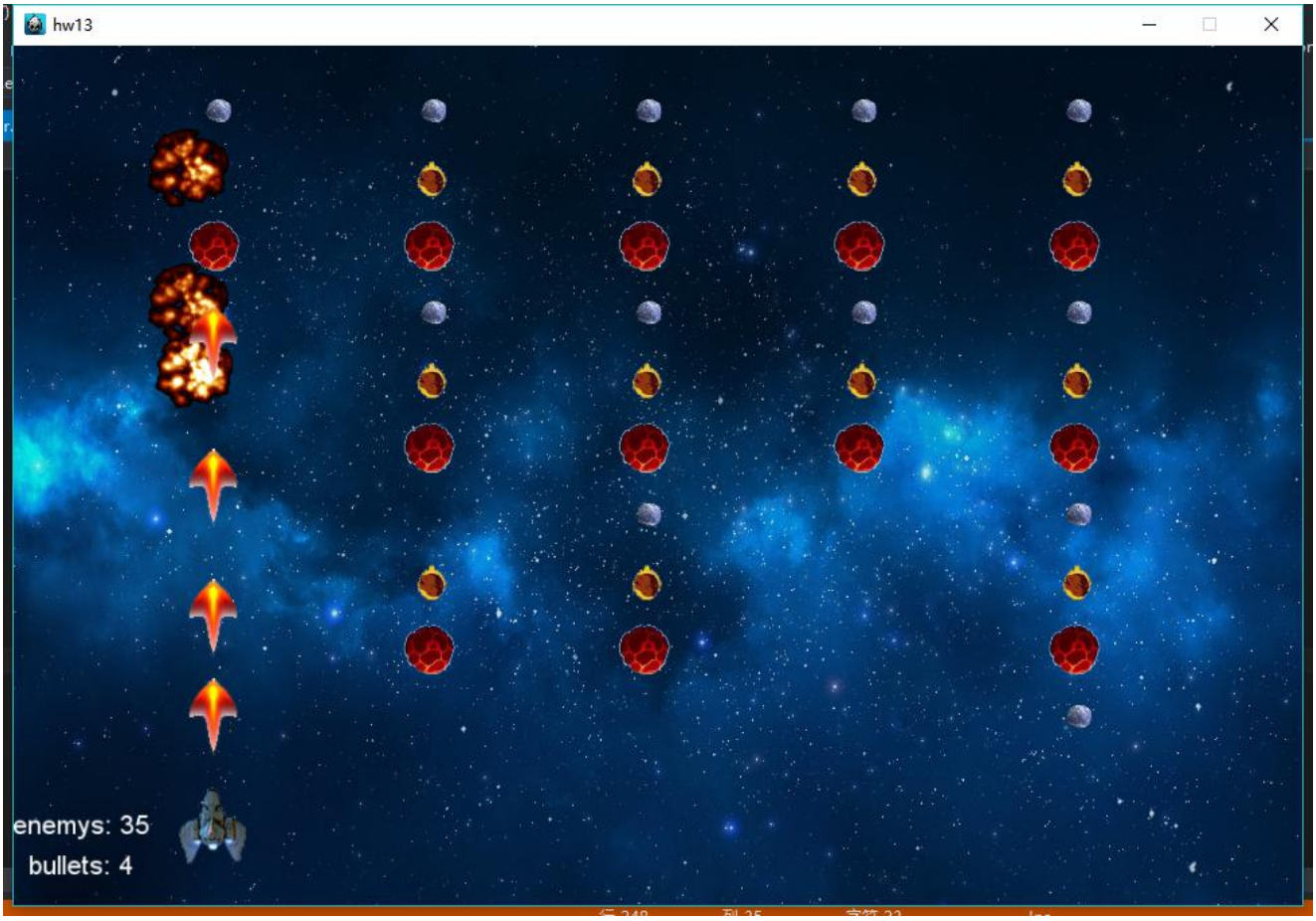
陨石可以增加：



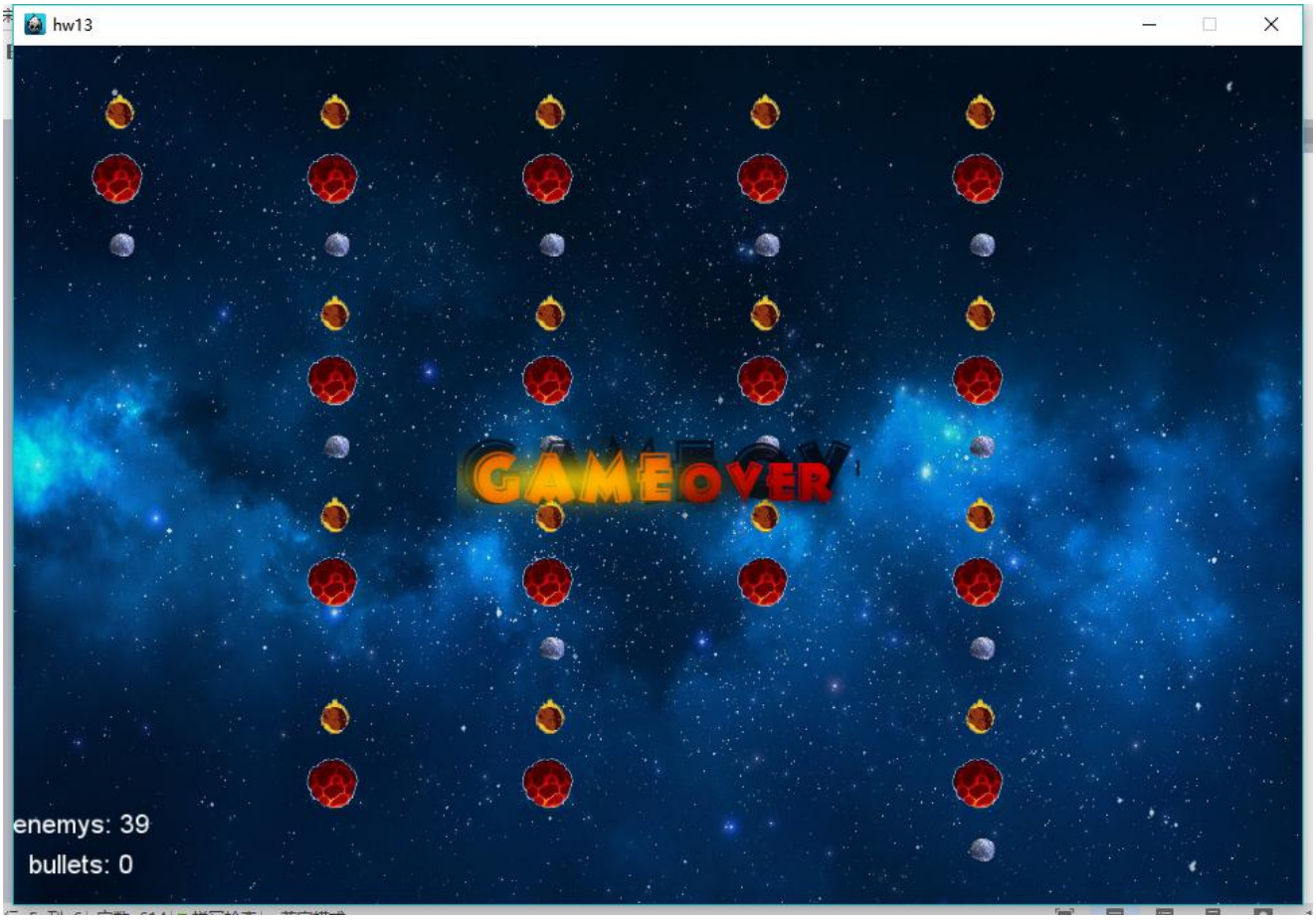
左右键移动和鼠标移动不出界：



发射子弹：



游戏结束：



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。

在判断子弹与陨石相遇的情况时迭代出了错误，结果导致死循环

解决方法：理清逻辑，解决了 bug

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。

这次代码用到了迭代器，有段时间没有用过迭代器，对于 List 遍历删除也不太熟悉，出了一些 bug，对本次代码架构不太熟悉，导致在遍历过程中出现死循环等 bug。认识到自己代码量的不足，以后应该多写代码练手。

1. 实验报告提交格式为 pdf。
2. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。