《现代操作系统应用开发》HW13 实验报告

姓名：羊伊 学号：15331349

一、参考资料：

homework13.pdf

week13_事件处理与音效.pdg

二、实验步骤：

1. 利用键盘事件实现飞船左右移动。

添加 EventListenerKeyboard，使其监听 KeyPressed, KeyRelease，并且调用相应的函数来响应。

```
void Thunder::addKeyboardListener() {
    auto KeyboardListener = EventListenerKeyboard::create();
    KeyboardListener->onKeyPressed = CC_CALLBACK_2(Thunder::onKeyPressed, this);
    KeyboardListener->onKeyReleased = CC_CALLBACK_2(Thunder::onKeyReleased, this);
    _eventDispatcher->addEventListenerWithFixedPriority(KeyboardListener, 2);
}
```

2. 利用键盘和触摸事件实现子弹发射。

调用 fire()。

键盘：

```
void Thunder::onKeyPressed(EventKeyboard::KeyCode code, Event* event) {
  switch (code) {                    159 enumeration values not handled in switch: 'KEY_NO
  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;
  case EventKeyboard::KeyCode::KEY_SPACE:
    fire();
    break;
  }
}
```

触摸：

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

在这里要注意的是要用 sequence 来实现先爆炸后移除的效果，否则的话没有爆炸效果。并且 temp 如果赋值为 iterator 的话有一定几率会出错。

```
void Thunder::meet(EventCustom * event) {
    auto explosion = Animate::create(AnimationCache::getInstance()->getAnimation("explosion"));
    for (auto it1 = bullets.begin(); it1 != bullets.end(); it1++) {
        for (auto it2 = enemys.begin(); it2 != enemys.end(); it2++) {
            if (*it1 == NULL || *it2 == NULL) {
                continue;
            } else if ((*it1)->getPosition().getDistance((*it2)->getPosition())<25) {
                auto temp1 = it1;
                it1 = bullets.erase(it1);
                (*temp1)->removeFromParentAndCleanup(true);
                auto temp = (*it2);
                (*it2)->runAction(Sequence::create(explosion, CallFunc::create([temp] {
                    temp->removeFromParentAndCleanup(true);
                }), NULL));
                it2 = enemys.erase(it2);
            }
        }
    }
```

4. 游戏过程中有背景音乐，发射子弹、击中陨石有音效。

预加载音乐，并播放背景音乐。

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

//≤•∑≈±≥∞"Ù¿÷
void Thunder::playBgm() {
    auto audio = SimpleAudioEngine::getInstance();
    audio->playBackgroundMusic("bgm.mp3", true);
}
```

5. 注意飞船、子弹的移动范围。

在 touchMoved 的响应中，只改变 X 的值，保证了飞机不会向前飞。

```
void Thunder::onTouchMoved(Touch *touch, Event *event) {
    Vec2 delta = touch->getDelta();
    auto moveBy = MoveBy::create(1, Vec2(delta.x, 0));
    player->runAction(moveBy);
}
```

6. 游戏结束飞船爆炸，移除所有监听器

这个很方便 ,调用_EventDispatcher 的 removeAllEventListeners()即可。

```
for (auto it = enemys.begin(); it != enemys.end(); it++) {
    if (player->getPosition().getDistance((*it)->getPosition())<25) {
        _eventDispatcher->removeAllEventListeners();
        player->runAction(Sequence::create(explosion, CallFunc::create([&] {
            player->removeFromParentAndCleanup(true);
        }), NULL));
        auto temp = *it;
        (*it)->runAction(Sequence::create(explosion, CallFunc::create([temp] {
            temp->removeFromParentAndCleanup(true);
        }), NULL));
        it = enemys.erase(it);
        break;
    }
}
```

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）

```
void Thunder::onTouchMoved(Touch *touch, Event *event) {
    Vec2 delta = touch->getDelta();
    auto moveBy = MoveBy::create(1, Vec2(delta.x, 0));
    player->runAction(moveBy);
}
```

2. 陨石向下移动并生成新的一行陨石

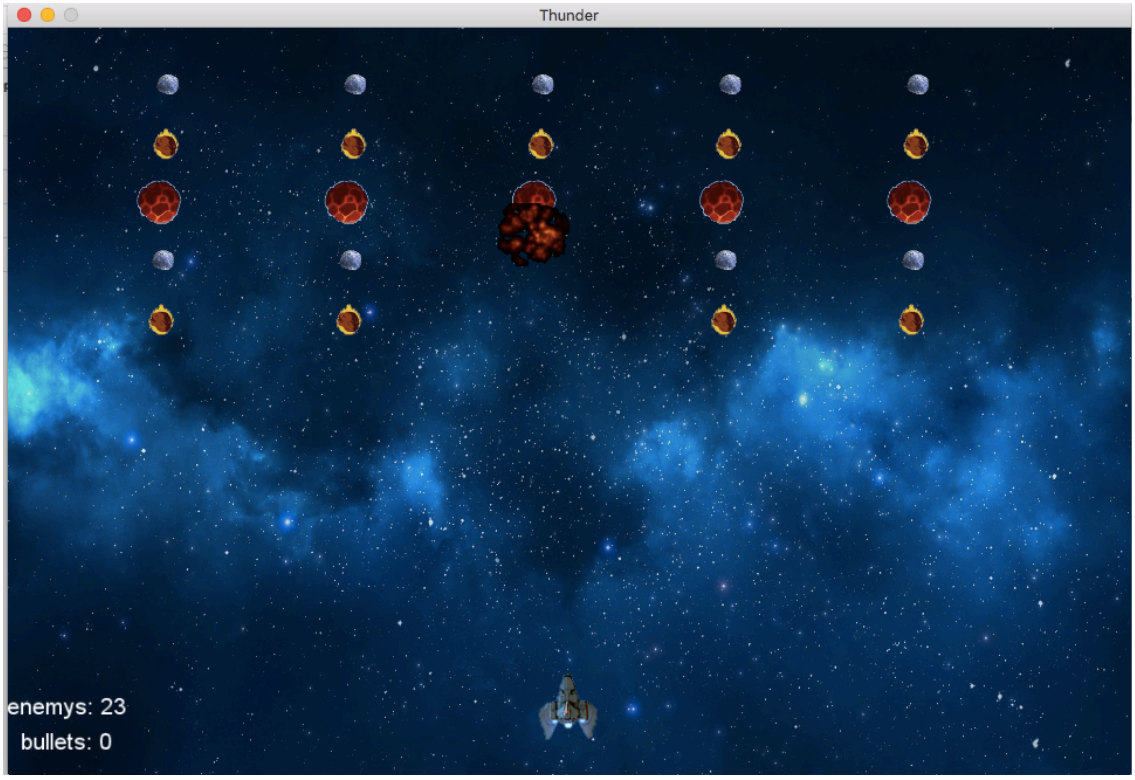
先向下移动现有的陨石，然后生成一行新的陨石并添加进来。这里用一个 static int 使得按顺序添加 3 种不同的陨石。

```
void Thunder::newEnemy() {
    for (Sprite* s : enemys) {
        if (s != NULL) {
            s->setPosition(s->getPosition() + Vec2(0, -50));
        }
    }
    static int i = 0;
    char enemyPath[20];
    sprintf(enemyPath, "stone%d.png", i+1);
    i = (i+1)%3;
    double width = visibleSize.width / (5.0+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+0.5), height);
        enemys.push_back(enemy);
        addChild(enemy, 1);
    }
}
```

3. 子弹和陨石的数量显示正确

这个也没做什么，但确实显示正确了。

三、效果截图



四、收获与感受

通过本次学习我还是收获了很多的。这次的重点是添加事件处理，这应该是一个游戏最核心的部分。

Cocos2d-x 对于事件的处理就使用了订阅者模式，即——由订阅者与发布者组成，游戏开发中，订阅者向事件发布器（发布者）注册监听器以监听某个事件的发生，当此事件发生时，事件发布器根据事件 ID 向监听器分发事件，然后监听器进行事件响应。

三个部分：事件监听器：即订阅者，封装了事件处理的代码，负责响应事件的处理。事件分发器：即发布者，当事件发生时通知对应事件的监听器。事件对象：包含了关于事件的信息。

对应到代码上就是 EventListenerXXX，_eventDispatcher 和响应的函数。掌握了这个概念，这次作业就能很好的完成了。