Cocos2dx第14周实验报告

一、参考资料

1. <https://blog.csdn.net/natsu1211/article/details/37915207> （动作函数回调）
2. 课件ppt

二、实验步骤

**·**需求及流程：

1. 利用键盘事件实现飞船左右移动。
2. 利用键盘和触摸事件实现子弹发射。
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。
4. 游戏过程中有背景音乐，发射子弹、击中陨石有音效。
5. 注意飞船、子弹的移动范围。
6. 游戏结束飞船爆炸，移除所有监听器

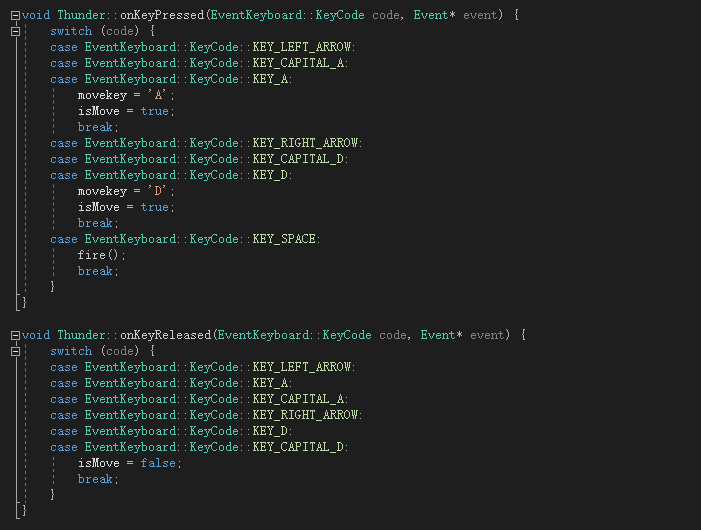
**·**详细实现过程：

1.移动飞船中，利用时间监听器监听当前按键触发的是哪个按钮以及是否可以移动（onKeyPress和onKeyRelease）；可以利用全局变量来进行传递信息；然后再定时器函数中判断是否可以移动，并调用移动函数。

增加键盘事件监听器：然后调用事件分发器分发处理事件：

1. // 添加键盘事件监听器
2. **void** Thunder::addKeyboardListener() {
3. // Todo
4. auto listenerKey = EventListenerKeyboard::create();
5. listenerKey->onKeyPressed = CC\_CALLBACK\_2(Thunder::onKeyPressed, **this**);
6. listenerKey->onKeyReleased = CC\_CALLBACK\_2(Thunder::onKeyReleased, **this**);
7. //auto dispatcher = Director::getInstance()->getEventDispatcher();
8. **this**->getEventDispatcher()->addEventListenerWithSceneGraphPriority(listenerKey, **this**);
9. }

onKeyPress和onKeyRelease函数源代码已给出如下：



然后使用movePlane函数即可：

1. // 移动飞船
2. **void** Thunder::movePlane(**char** c) {
3. // Todo
4. **if** (c == 'A') {
5. **if** (player->getPosition().x - 20 >= player->getContentSize().width / 2)
6. player->runAction(MoveBy::create(0.02f, Vec2(-20, 0)));
7. }
8. **else** **if** (c == 'D') {
9. **if**(player->getPosition().x + 20<=visibleSize.width-player->getContentSize().width/2)
10. player->runAction(MoveBy::create(0.02f, Vec2(20, 0)));
11. }
12. }

2.

键盘事件发射子弹：

再按下space按键时调用fire函数：其中在runAction中利用Sequence顺序执行一系列动作，并在动作结束之后利用CallFunc设置回调函数，移除当前bullet。

1. **void** Thunder::fire() {
2. auto bullet = Sprite::create("bullet.png");
3. bullet->setAnchorPoint(Vec2(0.5, 0.5));
4. bullets.push\_back(bullet);
5. bullet->setPosition(player->getPosition());
6. addChild(bullet, 1);
7. SimpleAudioEngine::getInstance()->playEffect("music/fire.wav", **false**);
8. // 移除飞出屏幕外的子弹
9. Sprite \*temp = bullet;
10. bullet->runAction(
11. Sequence::create(
12. MoveTo::create(1.0f, Vec2(bullet->getPosition().x, visibleSize.height+bullet->getContentSize().height)),
13. CallFunc::create([=] {
14. bullets.remove(temp);
15. //log("%d", bullets.size());
16. }),
17. nullptr
18. )
19. );
20. }

触摸时间与此类似：

首先增加触事件监听器，注册监听函数：

1. // 添加触摸事件监听器
2. **void** Thunder::addTouchListener() {
3. auto touchListener = EventListenerTouchOneByOne::create();
4. touchListener->onTouchBegan = CC\_CALLBACK\_2(Thunder::onTouchBegan, **this**);
5. touchListener->onTouchMoved = CC\_CALLBACK\_2(Thunder::onTouchMoved, **this**);
6. touchListener->onTouchEnded = CC\_CALLBACK\_2(Thunder::onTouchEnded, **this**);
8. **this**->getEventDispatcher()->addEventListenerWithSceneGraphPriority(touchListener, **this**);
9. }

然后实现鼠标点击发射炮弹的代码（当鼠标点击范围距离飞机较远时触发发射子弹事件）：

1. // 鼠标点击发射炮弹
2. **bool** Thunder::onTouchBegan(Touch \*touch, Event \*event) {
4. **if** (touch->getLocation().getDistance(player->getPosition()) <= 30)
5. isClick = **true**;
6. **else**
7. fire();
8. **return** **true**;
9. }
10. 判断子弹与陨石的撞击事件，在该函数中，遍历整个子弹向量与陨石向量，当二者距离过小时(<25)，认为两者发生碰撞，陨石在调用爆炸动画之后移除场景，子弹在碰撞时直接移除场景；并将二者从相应的向量数组中移除：
11. **void** Thunder::meet(EventCustom \* event) {
12. // 判断子弹是否打中陨石并执行对应操作
13. // Todo
14. **bool** flag = **false**;
15. **for** (auto bulletIter = bullets.begin(); bulletIter != bullets.end(); bulletIter++) {
16. **for** (auto enemyIter = enemys.begin(); enemyIter != enemys.end(); enemyIter++) {
17. **if** ((\*bulletIter)->getPosition().getDistance((\*enemyIter)->getPosition()) < 25) {
18. Sprite \*tempBullet = \*bulletIter;
19. Sprite \*tempEnemy = \*enemyIter;
20. tempEnemy->runAction(
21. Sequence::create(
22. Animate::create(
23. Animation::createWithSpriteFrames(explore, 0.05f, 1)
24. ),
25. CallFunc::create([=] {
26. SimpleAudioEngine::getInstance()->playEffect("music/explore.wav", **false**);
27. SimpleAudioEngine::getInstance()->resumeBackgroundMusic();
28. tempEnemy->removeFromParentAndCleanup(**true**);
29. }),
30. nullptr
31. )
32. );
33. tempBullet->removeFromParentAndCleanup(**true**);
34. bullets.remove(tempBullet);
35. enemys.remove(tempEnemy);
36. flag = **true**;
37. **break**;
38. }
39. }
40. **if** (flag) **break**;
41. }
42. }

4.

首先预加载音乐，使用SimpleAudioEngine类来加载背景音乐和音效：

1. //预加载音乐文件
2. **void** Thunder::preloadMusic() {
3. // Todo
4. auto audio = SimpleAudioEngine::getInstance();
5. audio->preloadBackgroundMusic("music/bgm.mp3");
7. audio->preloadEffect("music/explore.wav");
8. audio->preloadEffect("music/fire.wav");
9. }

然后播放背景音乐，其中playBackgroundMusic函数中第二个参数true表示循环播放：

1. //播放背景音乐
2. **void** Thunder::playBgm() {
3. // Todo
4. auto audio = SimpleAudioEngine::getInstance();
5. audio->playBackgroundMusic("music/bgm.mp3", **true**);
6. }

之后在发射子弹和击中陨石时调用playEffect函数即可：

1. SimpleAudioEngine::getInstance()->playEffect("music/fire.wav", **false**);

5.

在之前实现的飞船移动的函数中，判断移动边界即可保证飞船处于游戏界面中，键盘判断效果如下：

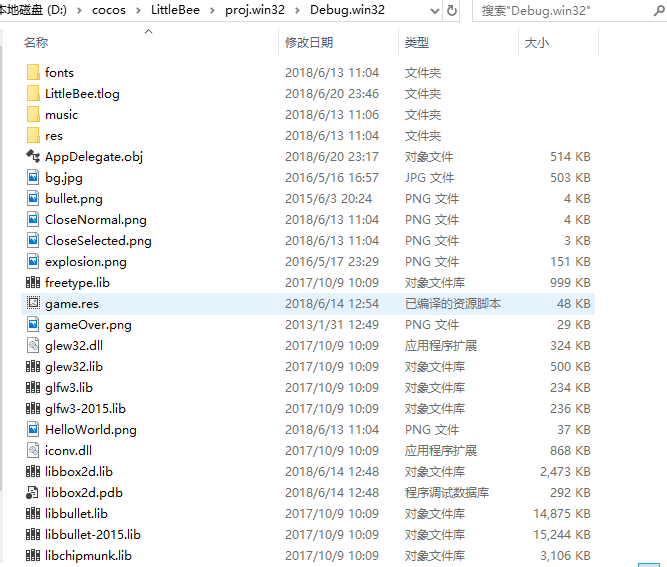
1. **if** (c == 'A') {
2. **if** (player->getPosition().x - 20 >= player->getContentSize().width / 2)
3. player->runAction(MoveBy::create(0.02f, Vec2(-20, 0)));
4. }
5. **else** **if** (c == 'D') {
6. **if**(player->getPosition().x + 20<=visibleSize.width-player->getContentSize().width/2)
7. player->runAction(MoveBy::create(0.02f, Vec2(20, 0)));
8. }

鼠标移动判断效果如下:

1. Vec2 delta = touch->getDelta();
2. //log("%f %f", delta.x, delta.y);
3. **float** x = player->getPosition().x + delta.x;
4. **if** (x < player->getContentSize().width / 2)
5. x = player->getContentSize().width / 2;
6. **if** (x > visibleSize.width - player->getContentSize().width / 2)
7. x = visibleSize.width - player->getContentSize().width / 2;
8. player->runAction(MoveTo::create(0, Vec2(x, player->getContentSize().height)));
9. 游戏结束之后，飞船调用爆炸动画，移除所有监听器和定时器：
10. player->runAction(Animate::create(
11. Animation::createWithSpriteFrames(explore, 0.05f, 1)
12. ));
14. unschedule(schedule\_selector(Thunder::update));
15. **this**->getEventDispatcher()->removeAllEventListeners();

7. 导出可执行文件exe

首先，打开工程文件夹下的/proj.win32/Debug.win32文件夹：



然后，使用ctrl+A全选，右键添加压缩文件LittleBee.zip：

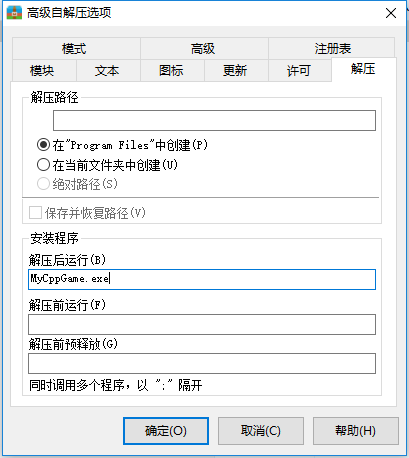
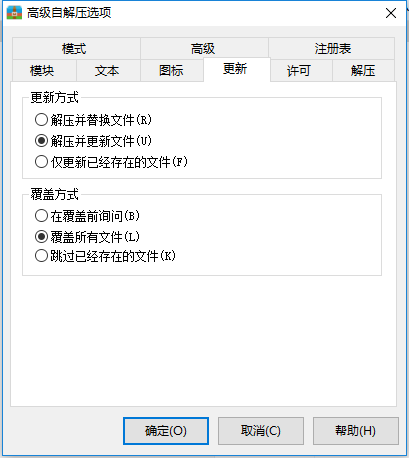


之后，将该压缩包剪切到工程文件目录下（不更改目录也可以）

双击打开该压缩包，选择自解压：



修改一下信息（解压后运行的文件选择相应名称的exe）：

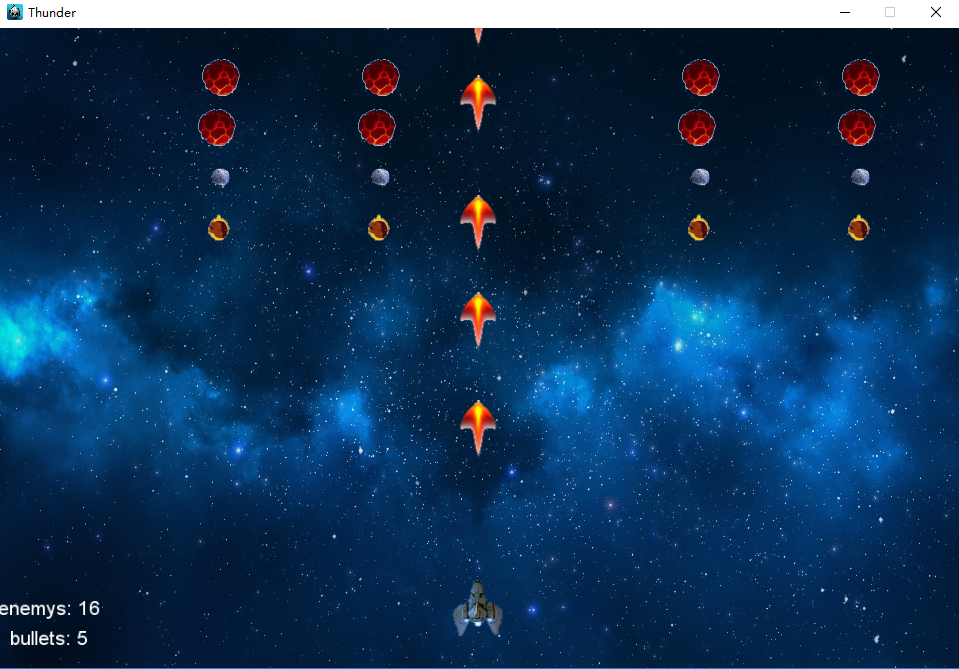




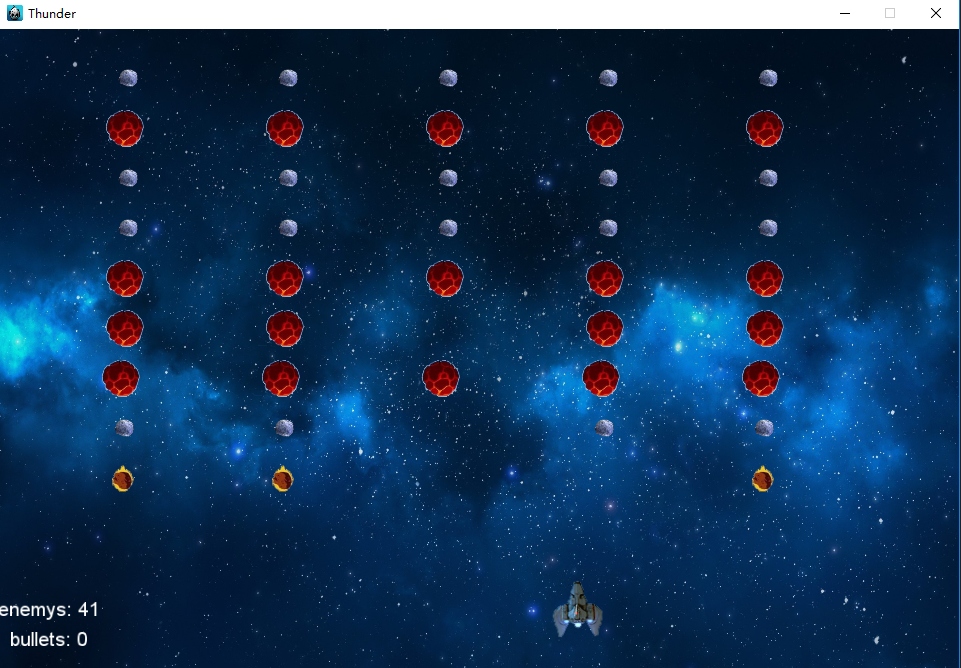
之后点击确定就得到了可以运行的exe文件。

三、关键步骤截图

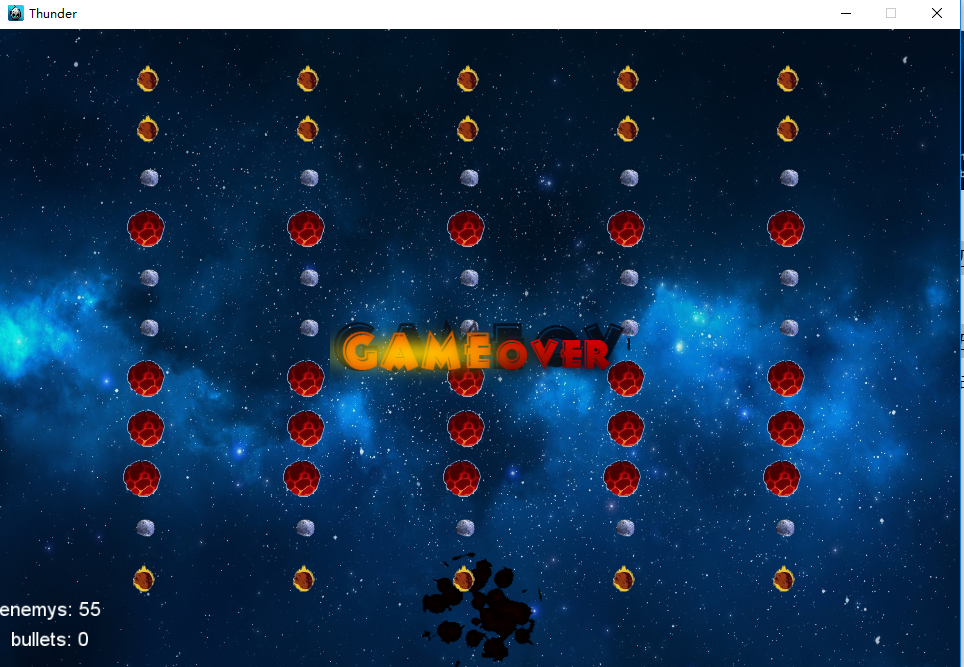
当前界面中存在五个子弹，16个陨石：



当前页面中有41个



游戏结束界面：



四、亮点与改进（可选）

鼠标触摸实现飞机移动，利用变量isClick判断当前是否处于鼠标触摸事件：

1. // 当鼠标按住飞船后可控制飞船移动 (加分项)
2. **void** Thunder::onTouchMoved(Touch \*touch, Event \*event) {
3. // Todo
4. **if** (isClick) {
5. Vec2 delta = touch->getDelta();
6. log("%f %f", delta.x, delta.y);
7. **float** x = player->getPosition().x + delta.x;
8. **if** (x < player->getContentSize().width / 2)
9. x = player->getContentSize().width / 2;
10. **if** (x > visibleSize.width - player->getContentSize().width / 2)
11. x = visibleSize.width - player->getContentSize().width / 2;
12. player->runAction(MoveTo::create(0, Vec2(x, player->getContentSize().height)));
13. }
14. }

陨石向下生成一排陨石，首先让已存在的陨石向下移动，若触碰到底部则结束游戏；之后生成一排新的陨石，添加到enemys向量数组中，并在界面上加以显示即可：

1. **void** Thunder::newEnemy() {
2. // Todo
3. **for** (auto it = enemys.begin(); it != enemys.end(); it++) {
4. **if** ((\*it)->getPosition().y - 50 < (\*it)->getContentSize().height / 2 || (\*it)->getPosition().getDistance(player->getPosition())<35) {
5. stopAc();
6. **return**;
7. }
8. (\*it)->setPosition((\*it)->getPosition().x, (\*it)->getPosition().y-50);
9. }
10. **char** enemyPath[20];
11. sprintf(enemyPath, "stone%d.png", rand()%3+1);
13. **double** width = visibleSize.width / (5 + 1.0),
14. height = visibleSize.height - 50;
16. **for** (**int** i = 0; i < 5; i++){
17. auto enemy = Sprite::create(enemyPath);
18. enemy->setAnchorPoint(Vec2(0.5, 0.5));
19. enemy->setScale(0.5, 0.5);
20. enemy->setPosition(width \* (i + 1)-80, height);
21. enemys.push\_back(enemy);
22. addChild(enemy, 1);
23. }
24. }

子弹和陨石的数量显示，在update定时器回调函数中，时刻使显示的子弹和陨石数量与相对应的向量数组大小保持一致：

1. **char** str[15];
2. sprintf(str, "enemys: %d", enemys.size());
3. enemysNum->setString(str);
4. sprintf(str, "bullets: %d", bullets.size());
5. bulletsNum->setString(str);

五、遇到的问题

问题一：

移除废除屏幕的子弹无效：

1. bullet->runAction(
2. Sequence::create(
3. MoveTo::create(1.0f, Vec2(bullet->getPosition().x, visibleSize.height+bullet->getContentSize().height)),
4. CallFunc::create([&] {
5. bullets.remove(bullet);
6. //log("%d", bullets.size());
7. }),
8. nullptr
9. )
10. );

上述代码错误的原因是因为，回调函数是异步执行的，当CallFunc内的lambda还为执行时，fire()已经运行结束，导致局部变量bullet被销毁，所以异步执行的回调函数中的bullet已不存在，所以删除失败。

1. Sprite \*temp = bullet;
2. bullet->runAction(
3. Sequence::create(
4. MoveTo::create(1.0f, Vec2(bullet->getPosition().x, visibleSize.height+bullet->getContentSize().height)),
5. CallFunc::create([=] {
6. bullets.remove(temp);
7. //log("%d", bullets.size());
8. }),
9. nullptr
10. )
11. );

上面为正确执行的代码，在lambda中采用=捕获bullet的值，这样在fire()执行完之后，该回调函数依然可以执行正确的结果。

问题二：

运行exe文件提示找不到可以执行的文件，该问题是因为在之前设置的解压后运行的文件名书写错误。如果是压缩的整个Debug.win32文件夹，那么加压后运行的文件应该写为./Debug.win32/…exe。