现代操作系统应用开发实验报告

学号： 15331248　　**班级　：** 上午班

姓名： 潘承远　　**实验名称：** Cocos2d(HW13)

一.参考资料

- http://www.cnblogs.com/findumars/p/6256856.html 如何用 Enigma Virtual Box 打包生成 exe 文件
- http://blog.csdn.net/nanshan_hzq/article/details/39961237 重新复习了调度器，知道如何在游戏结束的时候消除调度器
- 《事件处理与音效》了解了 cocos2d 事件处理的几种类型，以及如何给自己的游戏加上音效
- 《hw13》TA 给的作业说明中已经提供了很多思路，比如使用自定义事件来处理子弹遇到岩石的问题

二.实验步骤

- 仔细阅读课件与相关文档
- 完成要求 1“利用键盘事件实现飞船左右移动”。这个部分比较简单，onKeyPressed 和 onKeyReleased 的代码已经写好，我们只需要完成添加键盘事件监听器的函数和 movePlane 函数。U

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

```
// 移动飞船
void Thunder::movePlane(char c) {
  // Todo
    auto position = player->getPosition();
    if (c == 'A') {
        if (position.x >= 40) {
            auto movet = MoveBy::create(0.08, Vec2(-20, 0));
            player->runAction(movet);
        }
    }
    else if (c == 'D') {
        if (position.x <= visibleSize.width - 40) {
            auto movet = MoveBy::create(0.08, Vec2(20, 0));
            player->runAction(movet);
        }
    }
}
```

●完成要求 2“利用键盘和触摸事件实现子弹发射”，键盘部分已经提供了，只需写触摸事件。

```
// 添加触摸事件监听器
void Thunder::addTouchListener() {
    // Todo
    EventListenerTouchOneByOne* touchlistener = EventListenerTouchOneByOne::create();
    touchlistener->onTouchBegan = CC_CALLBACK_2(Thunder::onTouchBegan, this);
    touchlistener->onTouchMoved = CC_CALLBACK_2(Thunder::onTouchMoved, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(touchlistener, this);
}

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

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

●完成要求 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”。这个部分是整个实验陷阱最多的地方，必须要移除飞出屏幕外的子弹，否则使用 getDistance 遍历检查是否有子弹离岩石距离很近的时候就会有 bug。此外还要注意将爆炸后的岩石从 enemys 列表中删除，否则在 update 函数中左右移动会访问 enemys 列表的每一项，这时候会出错（list 中有该岩石，而实际上我们已经调用 removeFromParentAndCleanup 函数将该岩石从屏幕中删除了）。

```
    it2 = enemys.erase(it2);
```

```
// 切割爆炸动画帧
void Thunder::explosion() {
    // Todo
    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(0 * i, 212, 210, 180)));
        explore.pushBack(frame);
    }
    for (int i = 5; i < 8; i++) {
        auto frame = SpriteFrame::createWithTexture(texture, CC_RECT_PIXELS_TO_POINTS(Rect(0 * i, 0, 210, 180)));
        explore.pushBack(frame);
    }
}
```

●完成要求 4“游戏过程中有背景音乐，发射子弹、击中陨石有音效”，这个部分比较简单，只需要看老师提供的 pdf 就能解决。

●完成要求 5“注意飞船、子弹的移动范围。”如下添加 if 语句来判断

```
// 移动飞船
void Thunder::movePlane(char c) {
    // Todo
    auto position = player->getPosition();
    if (c == 'A') {
        if (position.x >= 40) {
            auto movet = MoveBy::create(0.08, Vec2(-20, 0));
            player->runAction(movet);
        }
    }
    else if (c == 'D') {
        if (position.x <= visibleSize.width - 40) {
            auto movet = MoveBy::create(0.08, Vec2(20, 0));
            player->runAction(movet);
        }
    }
}
```

●完成要求 6“游戏结束飞船爆炸，移除所有监听器”

```
// 判断游戏是否结束并执行对应操作
  list<Sprite*>::iterator it = enemys.begin();
  for (; it != enemys.end(); it++) {
      if ((*it)->getPosition().y < player->getPosition().y + 40) {
          SimpleAudioEngine::getInstance()->playEffect("music/explore.wav", false);
          Sprite* temp2 = player;
          player->runAction(
              Sequence::create(
                  Animate::create(Animation::createWithSpriteFrames(explore, 0.05f, 1)),
                  FadeOut::create(0.001f),
                  nullptr)
          );  //  end of runAction
          auto gameover = Sprite::create("gameOver.png");
          gameover->setPosition(visibleSize.width / 2, visibleSize.height / 2);
          this->addChild(gameover, 1);
          unschedule(schedule_selector(Thunder::update));
          _eventDispatcher->removeAllEventListeners();
          break;
      }
  }
// Todo
```

●完成加分项 1“利用触摸事件实现飞船移动。(点击飞船后拖动鼠标)”，这个部分比较简单，完成 onTouchMoved 函数即可。

```
// 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) {
  // Todo
    auto loc = touch->getLocation();
    auto pos = player->getPosition();
    if ((pos.x - 40 <= loc.x&&loc.x <= pos.x + 40)&&(pos.y - 40 <= loc.y&&loc.y <= pos.y + 40))
    player->setPositionX(loc.x);
}
```

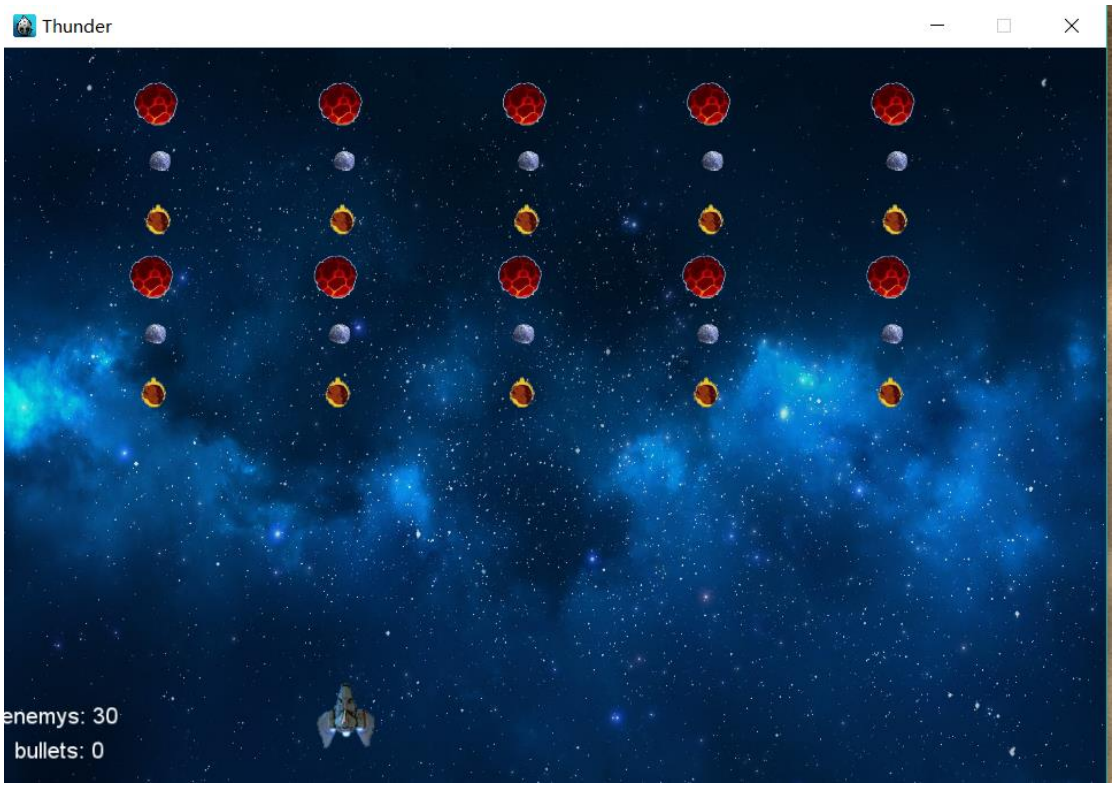
●完成加分项 2“陨石向下移动并生成新的一行陨石”。我是将原来处在屏幕上的岩石都向下移动，然后在最上面添加新的岩石行。此外我添加了 row 私有变量用于记录总共添加的岩石的行数，这样就能够在 3 种岩石中来回切换添加。

```
  // 陨石向下移动并生成新的一行(加分项)
  void Thunder::newEnemy() {
    // Todo
      int picnum = 1 + row % 3;
      row++;
      list<Sprite*>::iterator it = enemys.begin();
      for (; it != enemys.end(); it++) {
          auto x = (*it)->getPosition().x;
          auto y = (*it)->getPosition().y - 50;
          (*it)->setPosition(x, y);
      }
      char enemyPath[20];
      sprintf(enemyPath, "stone%d.png", picnum);
      double width = visibleSize.width / 6, height = visibleSize.height - 50;
      for (int i = 0; i < 5; i++) {
          auto enemy = Sprite::create(enemyPath);
          enemy->setAnchorPoint(Vec2(0.5, 0.5));
          enemy->setScale(0.5, 0.5);
          enemy->setPosition(width * (i + 0.5), height);
          enemys.push_back(enemy);
          addChild(enemy, 1);
      }
  }
```

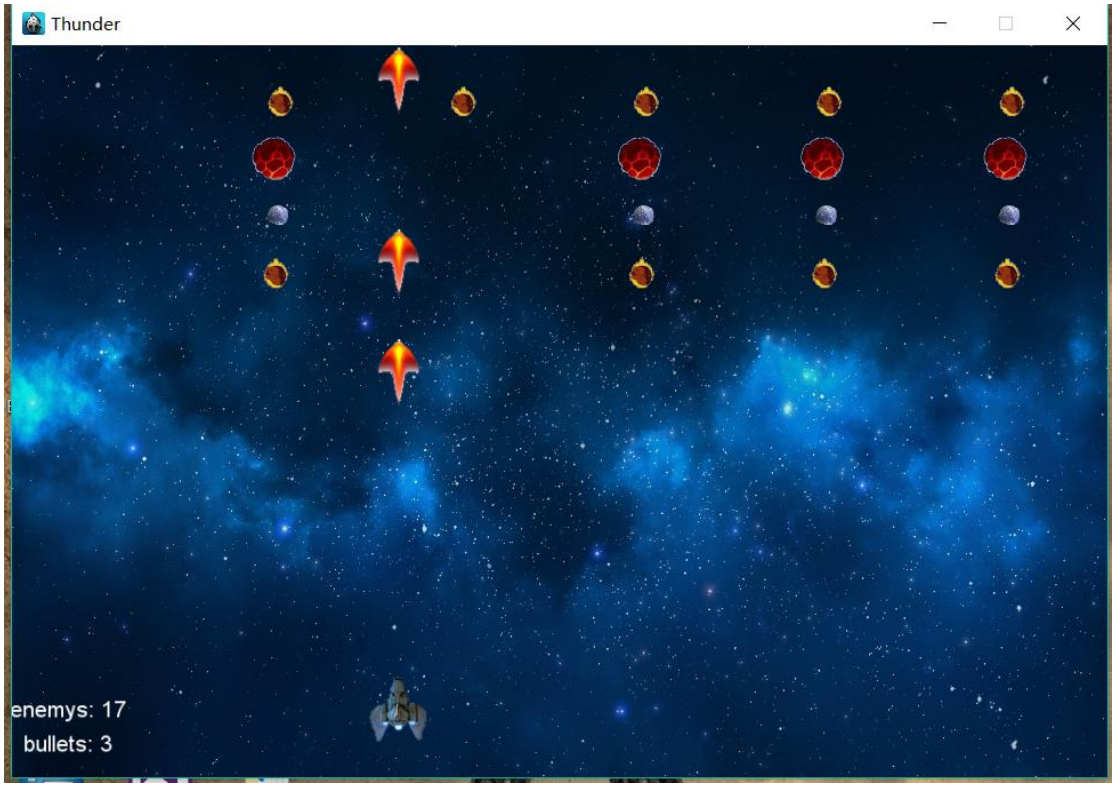
●完成加分项 3“子弹和陨石的数量显示正确”，只要在合适的地方给 bullets 和 enemys 列表添加或删除元素，就能够保证子弹和陨石的数量显示正确。

三.实验结果截图

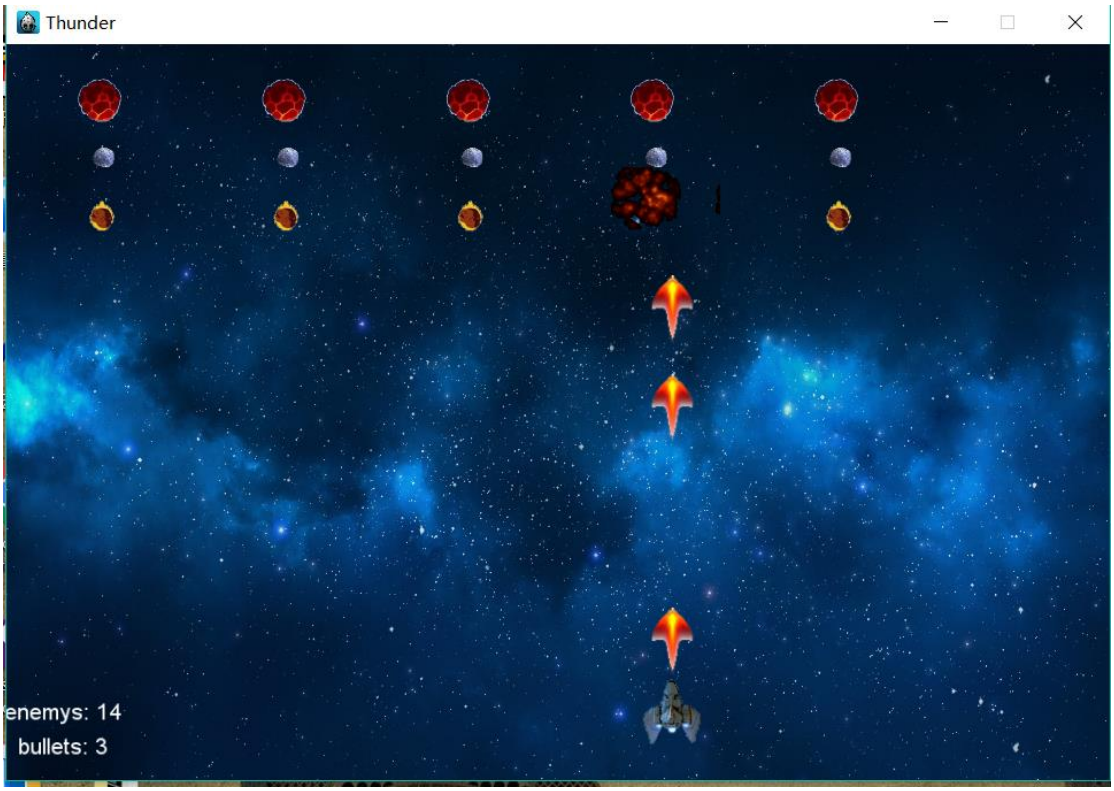
1. 利用键盘左右移动飞机，此外也可点击飞机拖动鼠标来左右移动飞机



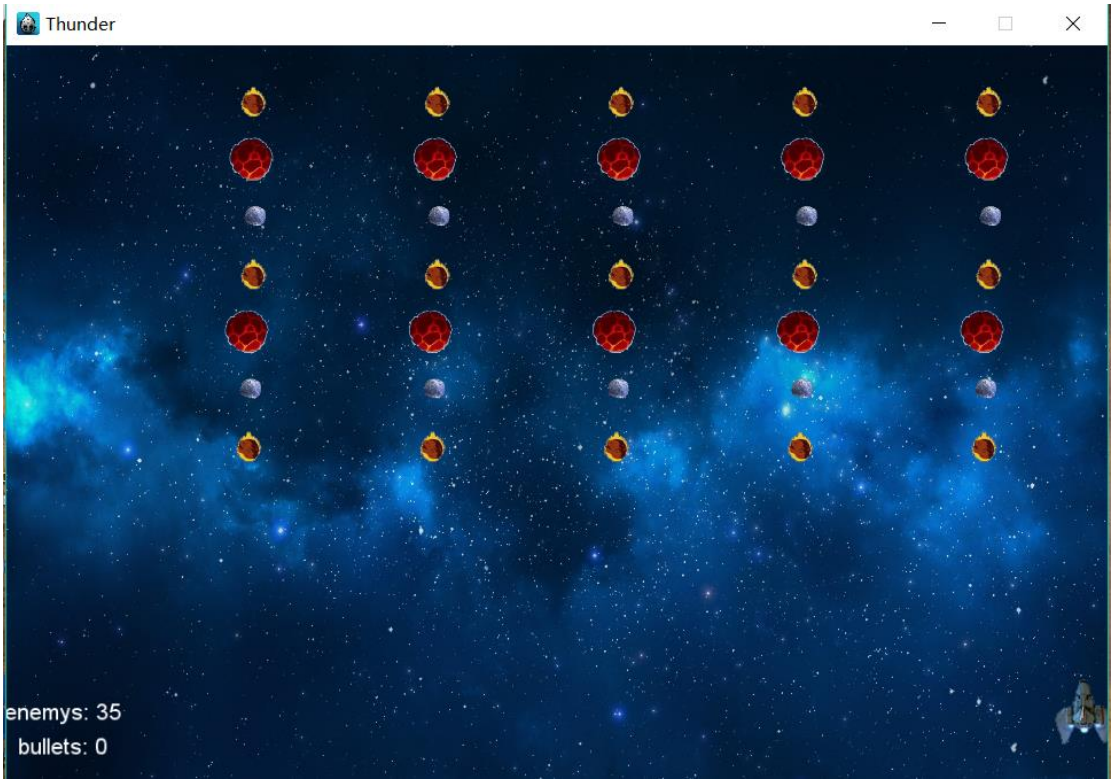
2. 利用键盘和触摸事件实现子弹发射



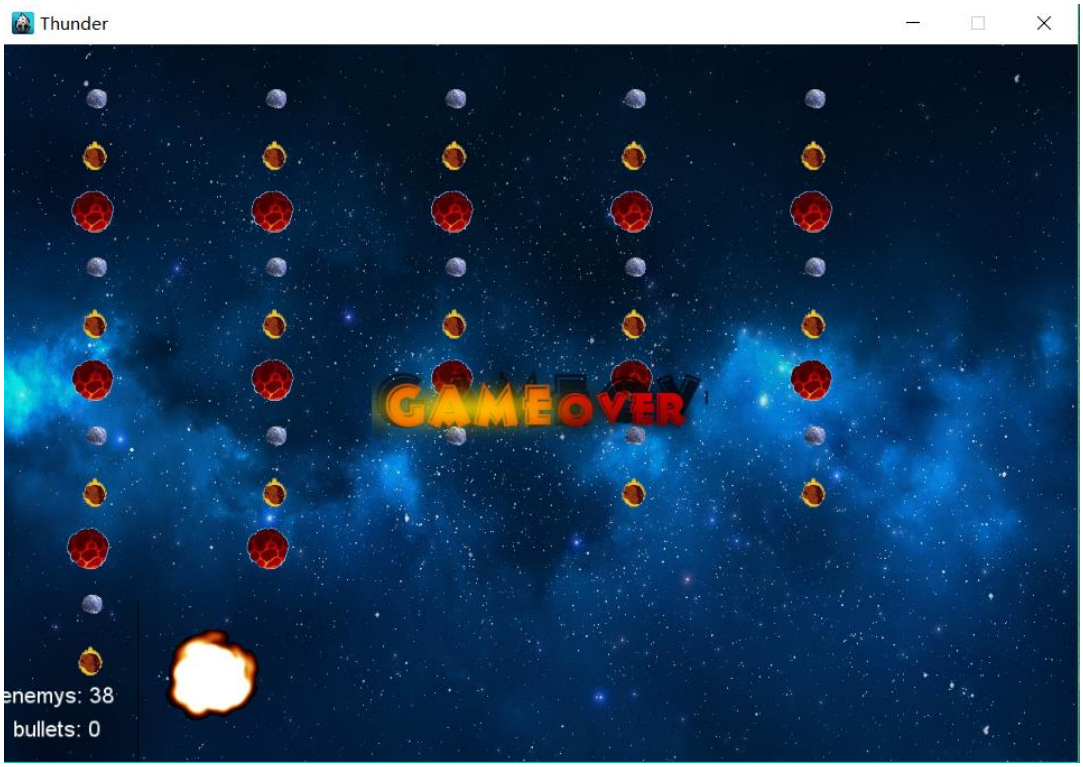
3. 子弹和陨石相距小于一定距离时，陨石爆炸，子弹消失



4. 飞机在右下角，此时它不能够再向右移动，同理左边一样



5. 游戏结束飞船爆炸，移除所有监听器



6. 成功将游戏打包（Alvin）

四.实验过程遇到的问题

● 在完成要求 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。”的时候花的时间最多。当我把 addCustomListener()和 meet 函数都写好之后运行，发现只有我发射的第一颗子弹能够造成陨石的爆炸，而以后发射的子弹都不能造成陨石的爆炸。我首先怀疑是不是我的自定义事件写错了，上网查了很多，用了很多种不同的形式来注册监听器，分发监听器，最后发现还是同样的毛病。最后是翻了 Q 群以后，有人说要移除屏幕外的子弹，这时候我才恍然大悟，移除之后就把 bug 轻松解决。因为我有使用 getDistance 函数来判断子弹和岩石的位置（遍历两个 list），必须要将屏幕外的子弹移除，不然会错。

● 此外还有一个问题是 vs 告诉我左右移动陨石那部分代码可能有问题，这不是已经写好的代码么？按道理不会有错，后来才发现这是因为在处理陨石爆炸的时候我将爆炸的陨石从屏幕中移除，但是没有在 enemys 移除它，这样左右移动陨石那部分代码需要遍历 enemys，发现 enemys 有该岩石，而“屏幕”中没有，自然会错。

```
216 void Thunder::meet(EventCustom * event) {
217     // 判断子弹是否打中陨石并执行对应操作
218     list<Sprite*>::iterator it1 = bullets.begin();
219     list<Sprite*>::iterator it2 = enemys.begin();
220     bool flag = false;
221     for (; it1 != bullets.end();) {
222         for (; it2 != enemys.end();) {
223             if ((*it1)->getPosition().getDistance((*it2)->getPosition()) < 50) {
224                 SimpleAudioEngine::getInstance()->playEffect("music/explore.wav", false);
225                 Sprite* temp = (*it2);
226                 Sprite* bullet = (*it1);
227                 (*it2)->runAction(
228                     Sequence::create(
229                         Animate::create(Animation::createWithSpriteFrames(explore, 0.05f, 1)),
230                         CallFunc::create([temp] {temp->removeFromParentAndCleanup(true);}),
231                         nullptr)
232                 );  //  end of runAction
233                 (*it1)->runAction(
234                         CallFunc::create([bullet] {bullet->removeFromParentAndCleanup(true); })
235                 );  //  end of runAction
236                 it2 = enemys.erase(it2);
237                 it1 = bullets.erase(it1);
238                 flag = true;
239                 break;
240             }
241             else it2++;
242         }
243         if (!flag) it1++;
244         flag = false;
245     }
```

五.思考与总结

本次实验个人感觉要比前面几次的陷阱多，必须要考虑周全才能保证运行的正确。除开要求 3 花费了我比较多的时间（主要是没有移除屏幕外的子弹和将爆炸的岩石与子弹从对应的 list 中会删除），其它的要求都比较简单，只需要看老师和 TA 提供的资料就能够解决。