现代操作系统应用开发实验报告

学号： 14970011　　**班级：** 上午班

姓名： 宋思亭　　**实验名称：** homework13

一．参考资料

作业要求文档，课件 PPT，

官网：http://www.cocos.com/

Github：https://github.com/cocos2d/cocos2d-x

用户手册：http://www.cocos2d-x.org/wiki/Cocos2d-x

商店：http://store.cocos.com/

API:　http://api.cocos.com/

二．实验步骤

1. 阅读作业需求和课件 PPT，了解阅读课件内容以及作业要求，了解事件分发与响应，音乐与音效的设置。

2. 设置加载音乐文件并播放背景音乐，直接调用 SimpleAudioEngine 标准接口

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

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

3. 调度器调用 update 函数，运行 newEnemy 生成新陨石，先将所有陨石 y 坐

标下移 50，再创建新陨石并均匀放置在窗口顶端

```
// 陨石向下移动并生成新的一行(加分项)
void Thunder::newEnemy() {
    static int stonelist = 1;
    for (auto it = enemys.begin(); it != enemys.end(); ++it) {
        if (*it != NULL) (*it)->setPosition((*it)->getPosition() + Vec2(0, -50));
    }
    char enemyPath[20];
    sprintf(enemyPath, "stone%d.png", stonelist);
    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(visibleSize.width / 6.0 * j + 65, visibleSize.height - 50);
        enemys.push_back(enemy);
        addChild(enemy, 1);
    }
    stonelist = stonelist % 3 + 1;
}
```

4. 设置按 A 和 D 键，飞船左右移动，注意飞船不能移动到窗口外

```
// 移动飞船
void Thunder::movePlane(char c) {
    auto nowPos = player->getPosition();
    float min;
    switch (c) {
        case 'A':
            min = (player->getBoundingBox().getMinX() > 20.0) ? 20.0 : player->getBoundingBox().getMinX();
            player->runAction(MoveBy::create(0.2f, Vec2(-min, 0)));
            break;
        case 'D':
            min = (visibleSize.width - player->getBoundingBox().getMaxX() > 20.0) ?
                20.0 : visibleSize.width - player->getBoundingBox().getMaxX();
            player->runAction(MoveBy::create(0.2f, Vec2(min, 0)));
            break;
    }
}
```

5. 设置移除飞出屏幕外的子弹，注意使用回调函数将子弹从容器中移除，方便计数

```
bullet->runAction(Sequence::create(MoveBy::create(1.0f, Vec2(0, visibleSize.height)), CallFunc::create([this, bullet]() {
            bullet->removeFromParentAndCleanup(true);
            this->bullets.remove(bullet);
}), nullptr));
```

6. 切割爆炸动画帧，将 explosion.png 中的爆炸图像切割并添加到帧动画

```
// 切割爆炸动画帧
void Thunder::explosion() {
    auto textureCache = Director::getInstance()->getTextureCache()->addImage("explosion.png");
    for (int i = 0; i < 1000; i += 200) {
        explore.pushBack(SpriteFrame::createWithTexture(textureCache, CC_RECT_PIXELS_TO_POINTS(Rect(i, 0, 140, 140))));
    }
    for (int i = 0; i < 800; i += 200) {
        explore.pushBack(SpriteFrame::createWithTexture(textureCache, CC_RECT_PIXELS_TO_POINTS(Rect(i, 190, 140, 140))));
    }
}
```

7. 在 meet 函数中判断子弹与陨石边界距离小于 25 则击中爆炸，播放击中音效，

   注意 it2 动画的 lambda 函数中，不能直接移除 it2，需要用 temp 中转

```
// 判断子弹是否打中陨石并执行对应操作
  Sprite* temp;
  bool boom = false;
  for (auto it1 = bullets.begin(); it1 != bullets.end(); ) {
      boom = false;
      for (auto it2 = enemys.begin(); it2 != enemys.end(); ++it2) {
          if (*it1 && *it2 && (*it1)->getPosition().getDistance((*it2)->getPosition()) < 25) {
              temp = *it2;
              (*it1)->removeFromParentAndCleanup(true);
              (*it2)->runAction(Sequence::create(Animate::create(Animation::createWithSpriteFrames(explore, 0.05f, 1)), CallFunc::create([this, temp] {
                  temp->removeFromParentAndCleanup(true);
              }), nullptr));
              SimpleAudioEngine::getInstance()->playEffect("music/explore.wav", false);
              it1 = bullets.erase(it1);
              it2 = enemys.erase(it2);
              boom = true;
              break;
          }
      }
      if (!boom) ++it1;
  }
```

   判断陨石下边沿低于飞船上边沿，则游戏结束，播放游戏结束音效，飞船爆炸

并移除所有监听器

```
// 判断游戏是否结束并执行对应操作
  for (auto it = enemys.begin(); it != enemys.end(); ++it) {
      if (*it && (*it)->getBoundingBox().getMinY() <= player->getBoundingBox().getMaxY()) {
          temp = player;
          player->runAction(Sequence::create(Animate::create(Animation::createWithSpriteFrames(explore, 0.05f, 1)), CallFunc::create([this, temp] {
              temp->removeFromParentAndCleanup(true);
              auto gameover = Sprite::create("gameOver.png");
              gameover->setAnchorPoint(Vec2(0.5, 0.5));
              gameover->setPosition(visibleSize / 2);
              this->addChild(gameover, 1);
          }), nullptr));
          SimpleAudioEngine::getInstance()->playEffect("music/explore.wav", false);
          unschedule(schedule_selector(Thunder::update));
          this->getEventDispatcher()->removeAllEventListeners();
          return;
      }
  }
```

8. 添加监听器，自定义监听器监听 meet 函数中碰撞和游戏结束事件触发，键盘

   监听按键和释放，以及触摸/鼠标拖动监听

```
// 添加自定义监听器
void Thunder::addCustomListener() {
    auto customListener = EventListenerCustom::create("meet", CC_CALLBACK_1(Thunder::meet, this));
    _eventDispatcher->addEventListenerWithFixedPriority(customListener, 1);
}

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

```
// 添加触摸事件监听器
void Thunder::addTouchListener() {
    this->setTouchEnabled(true);
    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);
    this->getEventDispatcher()->addEventListenerWithSceneGraphPriority(touchListener, this);
}
```

9. 设置鼠标点击发射炮弹，若点击在飞船上，则可以按住拖动飞船，注意不能拖动飞船离开窗口

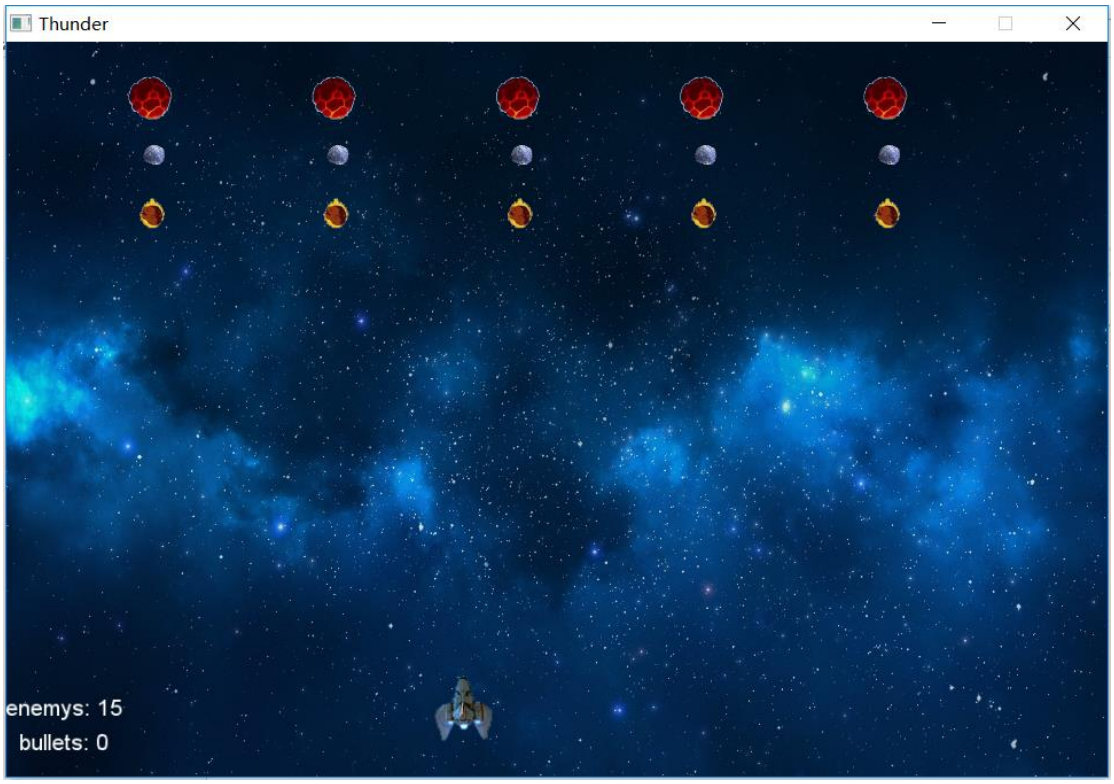
```
// 鼠标点击发射炮弹
bool Thunder::onTouchBegan(Touch *touch, Event *event) {
    isClick = player->getBoundingBox().containsPoint(touch->getLocation());
    fire();
    return true;
}

// 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) {
    if (isClick) {
        float x = touch->getDelta().x + player->getPositionX();
        if (x < 0) x = 0;
        if (x > visibleSize.width) x = visibleSize.width;
        player->setPosition(x, player->getPositionY());
    }
}
```

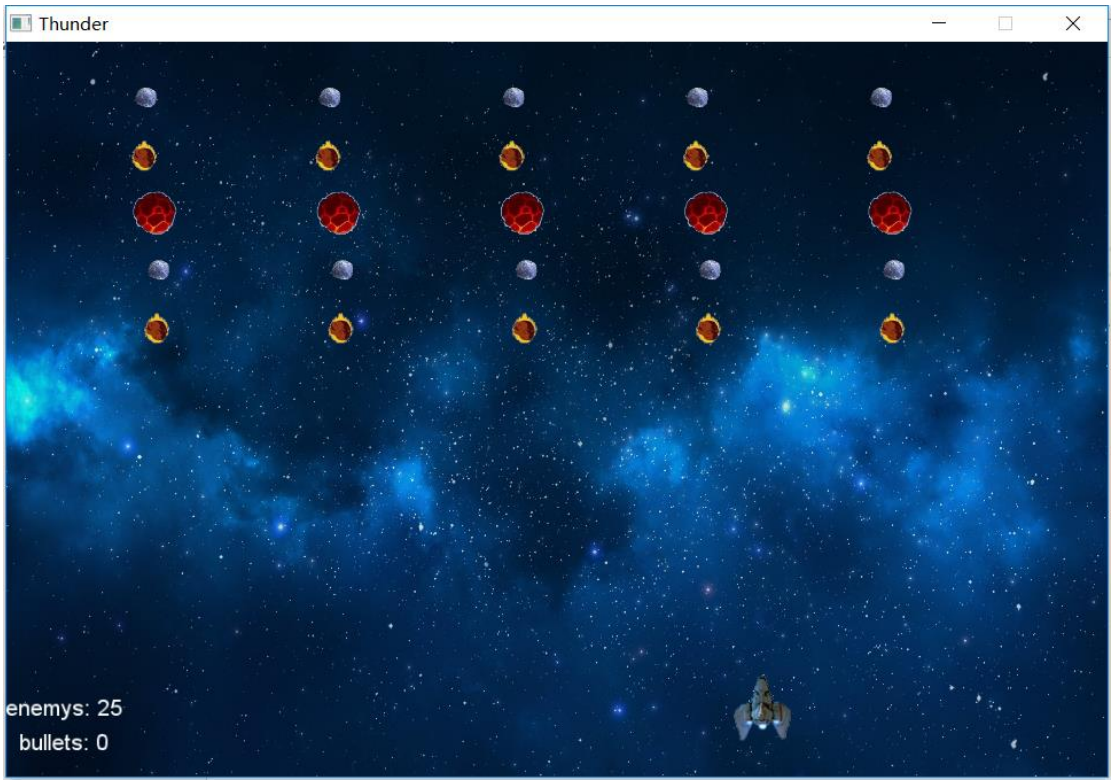
10. 调试项目

三．实验结果截图

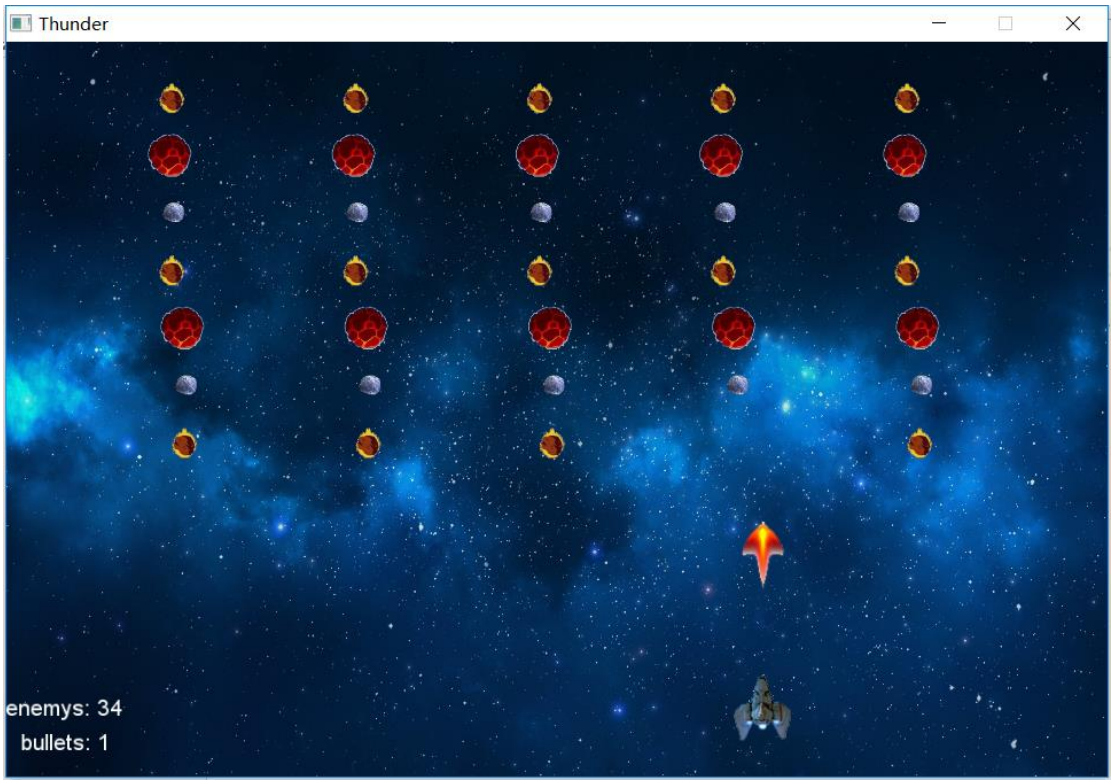
1. 打开窗口



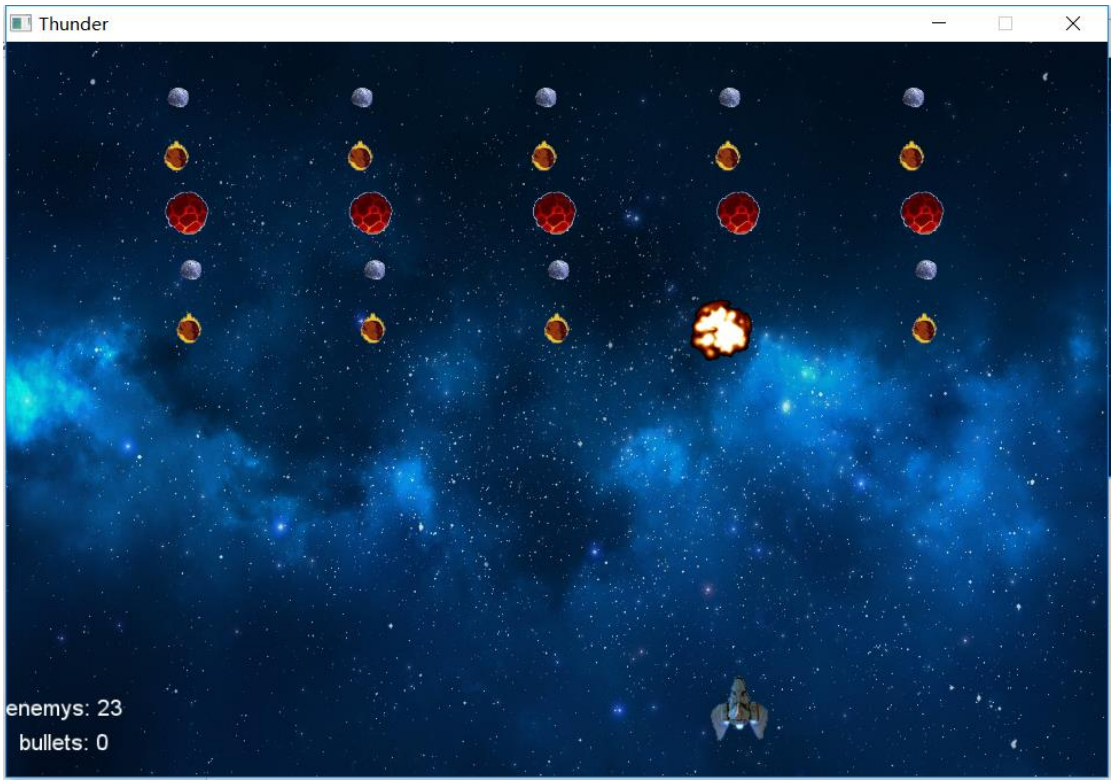
2. 按左右键/AD 键/鼠标拖动飞船左右移动，陨石向下移动并生成新陨石



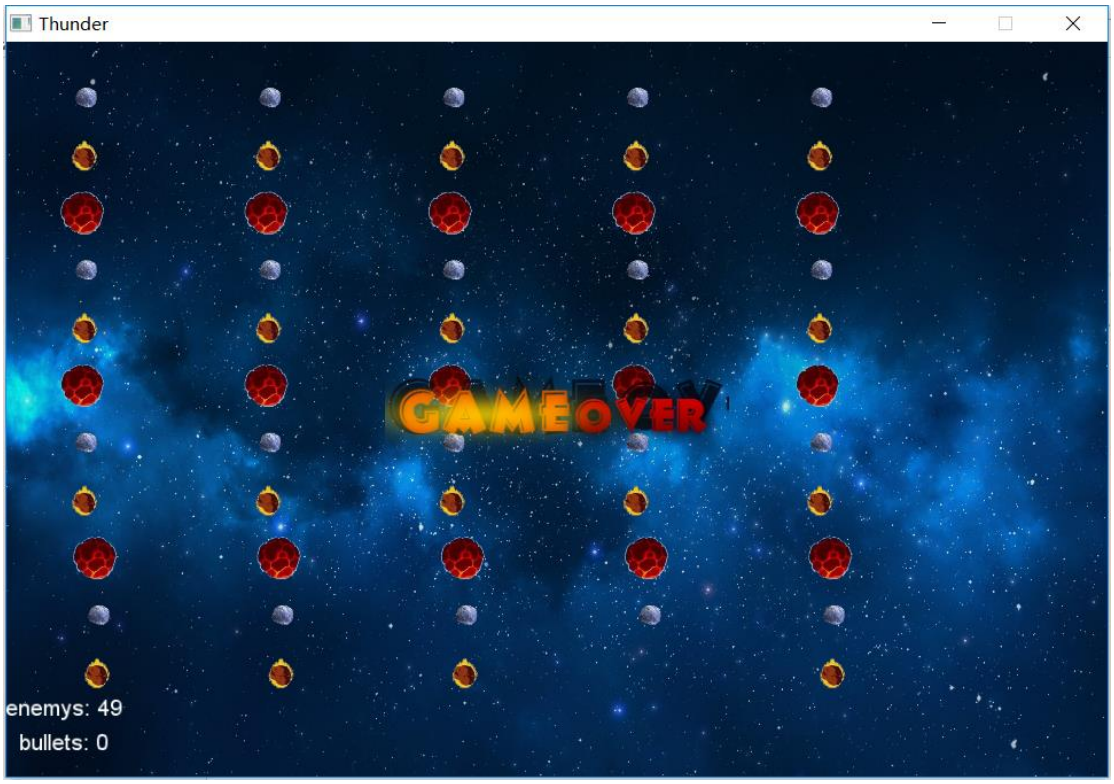
3. 点击鼠标/按空格键发射子弹，左下角显示实时子弹和陨石数量



4. 子弹击中陨石后爆炸消失



陨石下落到飞船高度时游戏结束



四．实验过程遇到的问题

1. 判定子弹击中陨石时，删除迭代器中的元素返回下一个元素而不需要再执行 it++，否则循环会出错

五．思考与总结

1. 真看似简单的一个小游戏，实现起来还是遇到很多问题，需要通过网络寻找答案，经过这次作业，对事件分发与响应，音乐与音效的设置有了更深的了解。
2. 把程序分解成一个个小的部分，分而治之，更有效率而且更容易排错。
3. 学习理论之后直接去实践很有用，虽然途中遇到了很多 bug 甚至是很难找出的错误的断点。但是只要肯花时间，一定会解决眼前的困难的。