现代操作系统应用开发实验报告

姓名：_____金汇丰_____

学号：_____16340097_____

实验名称：cocos2d-x数据结构，本地存储和tilemap（横版游戏）；
事件处理与音效（小蜜蜂）；
物理引擎与粒子系统（打砖块）；

一.参考资料

https://gfzheng.github.io/MOSAD/
https://blog.csdn.net/lin453701006/article/details/54409380
https://blog.csdn.net/qq_35131940/article/details/77584982
https://www.cnblogs.com/luorende/p/6684155.html

二.实验步骤

1.横版游戏

（1）产生怪物并向角色靠近

```
void HelloWorld::AddMonster(float dt){
    if (alive) {
        auto fac = Factory::getInstance();
        auto m = fac->createMonster();
        float x = random(origin.x,visibleSize.width);
        float y = random(origin.y,visibleSize.height);
        m->setPosition(x,y);
        addChild(m,3);
    }
}
void HelloWorld::MonsterMove(float dt){
    if(alive){
        auto fac = Factory::getInstance();
        fac->moveMonster(player->getPosition(), 4.0f);

    }
}
```

（2）被怪物攻击：根据血量判断是掉血还是死亡

```
void HelloWorld::hitByMonster(float dt){
    if (alive) {
        auto fac = Factory::getInstance();
        Sprite* collision = fac->collider(player->getBoundingBox());
        if (collision != NULL) {
            removeChild(collision);
            if(pT->getPercentage() <= 0){
                pT->setPercentage(0);
                alive = false;
                auto die_animation = Animation::createWithSpriteFrames(dead, 0.1f);
                die_animation->setRestoreOriginalFrame(true);
                AnimationCache::getInstance()->addAnimation(die_animation, "dieAnimation");
                auto action = Animate::create(AnimationCache::getInstance()->getAnimation("dieAnimation"));
                player->runAction(action);
            }else{
                schedule(schedule_selector(HelloWorld::subHp), 0.05f, 20, 0);
            }
            fac->removeMonster(collision);
        }
    }
}
```

（3）角色攻击怪物：设定一定的攻击范围，如果怪物在攻击范围内，将之移除

```
bool HelloWorld::attackMonster(){
    auto fac = Factory::getInstance();
    Rect playerRect = player->getBoundingBox();
    Rect attackRect = Rect(playerRect.getMinX() - 40, playerRect.getMinY(),
                           playerRect.getMaxX() - playerRect.getMinX() + 30, playerRect.getMaxY() - playerRect.getMinY());
    Sprite* collision = fac->collider(attackRect);
    if (collision != NULL) {
        removeChild(collision);
        fac->removeMonster(collision);
    }
    return collision != NULL;
}
```

攻击按键函数中：调用攻击怪物函数，如果击中加分并回血

```
    if (attackMonster()) {
        scoreNum++;
        database->setIntegerForKey("score", scoreNum);
        if(pT->getPercentage() >= 100){
            pT->setPercentage(100);
        }else{
            schedule(schedule_selector(HelloWorld::addHp), 0.05f, 20, 0);
        }
    }
```

（4）使用Tilemap创建地图

```
TMXTiledMap* tmx = TMXTiledMap::create("map1.tmx");
tmx->setPosition(visibleSize.width / 2, visibleSize.height / 2);
tmx->setAnchorPoint(Vec2(0.5, 0.5));
tmx->setScale(Director::getInstance()->getContentScaleFactor());
this->addChild(tmx, 0);
```

（5）bonus：本地存储
从本地存储读数据

```
scoreNum = database->getIntegerForKey("score");
```

击杀怪物更新本地存储

```
    scoreNum++;
    database->setIntegerForKey("score", scoreNum);
```

2.小蜜蜂（Thunder）

（1）背景音乐的预加载和播放

```
//预加载音乐文件
void Thunder::preloadMusic() {
    auto audio = SimpleAudioEngine::getInstance();
    audio->preloadEffect("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);

}
```

（2）飞船左右移动

```
// 移动飞船
void Thunder::movePlane(char c) {

    if (c == 'A') {
        if(player->getPosition().x - 10 >= 0){
            auto moveby = MoveBy::create(0.1f, Vec2(-10, 0));
            player->runAction(moveby);
        }
    }else if(c == 'D'){
        if(player->getPosition().x + 10 <= visibleSize.width){
            auto moveby = MoveBy::create(0.1f, Vec2(10, 0));
            player->runAction(moveby);
        }
    }

}
```

OnkeyPressed中

```
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;
```

(3)添加键盘事件监听器

```
void Thunder::addKeyboardListener() {
    // Todo
    auto keyBoardListener = EventListenerKeyboard::create();
    keyBoardListener->onKeyPressed = CC_CALLBACK_2(Thunder::onKeyPressed, this);
    keyBoardListener->onKeyReleased = CC_CALLBACK_2(Thunder::onKeyReleased, this);
    _eventDispatcher->addEventListenerWithSceneGraphPriority(keyBoardListener, player);
}
```

（4）发射子弹并播放音效

```
//发射子弹
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
    // Todo

}
```

子弹的飞行与移除（update函数中）

```
Sprite* bulletOut;
for (Sprite* s : bullets) {
    if (s != NULL) {
        if (s->getPosition().y > visibleSize.height) {
            bulletOut = s;
        } else {
            s->setPosition(s->getPosition() + Vec2(0, 30));
        }
    }
}
bullets.remove(bulletOut);
```

（5）爆炸动画

```
// 创建爆炸动画帧
void Thunder::explosion() {
    auto texture = Director::getInstance()->getTextureCache()->addImage("explosion.png");
    explore.reserve(8);
    for (int i = 0; i < 8; i++) {
        auto frame = SpriteFrame::createWithTexture(texture, CC_RECT_PIXELS_TO_POINTS(Rect(199 * (i % 5), 200 * (i / 5),
            199, 200)));
        explore.pushBack(frame);
    }

}
```

（6）判断子弹打中陨石

```
void Thunder::meet(EventCustom * event) {
    // 判断子弹是否打中陨石并执行对应操作
    // Todo
    for (auto b = enemys.begin(); b != enemys.end(); b++) {
        for (auto s = bullets.begin(); s != bullets.end(); s++) {

            if ((*b)->getPosition().getDistance((*s)->getPosition()) < 25) {

                auto audio = SimpleAudioEngine::getInstance();
                audio->playEffect("music/explore.wav", false);
                auto temp1 = (*b);
                auto temp2 = (*s);
                (*s)->runAction(Sequence::create(Animate::create(Animation::createWithSpriteFrames(explore, 0.05f,
                    1)),CallFunc::create([temp1, temp2] {
                    temp1->removeFromParentAndCleanup(true);
                    temp2->removeFromParentAndCleanup(true);

                }),    nullptr));
                enemys.erase(b);
                bullets.erase(s);
                return;
            }
        }
    }
}
```

添加监听器

```
void Thunder::addCustomListener() {
    auto meetEventListener = EventListenerCustom::create("meet", CC_CALLBACK_1(Thunder::meet, this));
    _eventDispatcher->addEventListenerWithFixedPriority(meetEventListener, 1);

}
```

（7）游戏停止

```
void Thunder::stopAc() {

        Sprite* gameover = Sprite::create("gameOver.png");
        gameover->setPosition(visibleSize.width / 2, visibleSize.height / 2);
        addChild(gameover, 5);
        this->_eventDispatcher->removeAllEventListeners();
        Director::getInstance()->pause();

}
```

陨石移动到底部，飞船爆炸并停止（update中）

```
//陨石左右移动
for (Sprite* s : enemys) {
    if (s != NULL) {
        s->setPosition(s->getPosition() + Vec2(dir, 0));
        if (s->getPosition().y < 88) {
            if (stop == false) {
                auto audio = SimpleAudioEngine::getInstance();
                audio->playEffect("music/explore.wav", false);
                audio->pauseBackgroundMusic();
                auto explosion = Animate::create(Animation::createWithSpriteFrames(explore, 0.05f, 1));
                explosion->setTag(666);
                player->runAction(explosion);
                stop = true;
            }else{
                if (player->getActionByTag(666) == nullptr) {
                    stopAc();
                }
            }

        }
    }
}
```

（8）bouns：向下移动添加新的陨石

```
void Thunder::newEnemy() {
    // Todo
    for (auto s : enemys) {
        s->setPosition(s->getPosition() + Vec2(0, -50));
    }
    char enemyPath[20];
    int i = rand() % 3;
    sprintf(enemyPath, "stone%d.png", 3 - i);
    double width = visibleSize.width / 6,
    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), height);
        enemys.push_back(enemy);
        addChild(enemy, 1);

    }
}
```

```
++ct;
if (ct == 120)
    ct = 40, dir = -dir;
else if (ct == 80) {
    dir = -dir;
    newEnemy();  // 陨石向下移动并生成新的一行(加分项)
}
```

鼠标控制飞船移动，点击发射子弹，并添加监听器

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

}

// 鼠标点击发射炮弹
bool Thunder::onTouchBegan(Touch *touch, Event *event) {
    if (touch->getLocation().getDistance(player->getPosition()) <= 30)
        isClick = true;
    // Todo
    if (!isClick) {
        fire();
    }
    return true;
}

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

// 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
    if (isClick) {
        Vec2 delta = touch->getDelta();
        player->setPosition(player->getPosition() + Vec2(delta.x, 0));
    }

}
```

3.打砖块

（1）板子的移动、蓄力、发射

```
case cocos2d::EventKeyboard::KeyCode::KEY_LEFT_ARROW:
        player->getPhysicsBody()->setVelocity(Vec2(-300,0));

  break;
case cocos2d::EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
  // ϕÛ"""Δ∂0
  // Todo
        player->getPhysicsBody()->setVelocity(Vec2(300,0));
  break;

case cocos2d::EventKeyboard::KeyCode::KEY_SPACE: // ø™ º–Ó¡¶
        spHolded = true;
```

```
void HitBrick::onKeyReleased(EventKeyboard::KeyCode code, Event* event) {
  switch (code) {
  case cocos2d::EventKeyboard::KeyCode::KEY_LEFT_ARROW:
  case cocos2d::EventKeyboard::KeyCode::KEY_RIGHT_ARROW:
    // ÕÉ+π'Àð0
    // Todo
          player->getPhysicsBody()->setVelocity(Vec2(0,0));
    break;
  case cocos2d::EventKeyboard::KeyCode::KEY_SPACE:    // –Ó¡¶Ð· ¯£'–°»ÛΣ¢…%
          launch();
    break;
```

```
void HitBrick::update(float dt) {
    if (spHolded) {
        spFactor += 5;
    }
}
```

```
void HitBrick::launch(){
    if (onBall) {
        m_world->removeJoint(joint1);
        ball->getPhysicsBody()->setVelocity(Vec2(0,spFactor));
        onBall = false;
    }
}
```

(2) 生成砖块，设置刚体

```
void HitBrick::BrickGenerated() {

  for (int i = 0; i < 3; i++) {
    int cw = 20;
    while (cw <= visibleSize.width) {
        auto box = Sprite::create("box.png");
        // Œ™$®ªÊ…Ë+√Π'ÃÅ Ù–'
        // Todo
        auto boxBody = PhysicsBody::createBox(box->getContentSize(),PhysicsMaterial(100.0f,1.0f,0.0f));
        boxBody->setCollisionBitmask(0x03);
        boxBody->setCategoryBitmask(0x02);
        boxBody->setContactTestBitmask(0x02);
        boxBody->setDynamic(false);
        box->setPhysicsBody(boxBody);
        box->setPosition(Vec2(cw,visibleSize.height - 30 * i - 15));
        box->setTag(2);
        this->addChild(box,2);
        cw += 50;

    }

  }
```

（3）关节固定球和板子

```cpp
void HitBrick::setJoint() {
    joint1 = PhysicsJointPin::construct(ball->getPhysicsBody(), player->getPhysicsBody(), player->getAnchorPoint());
    m_world->addJoint(joint1);
}
```

（4）设置物理属性

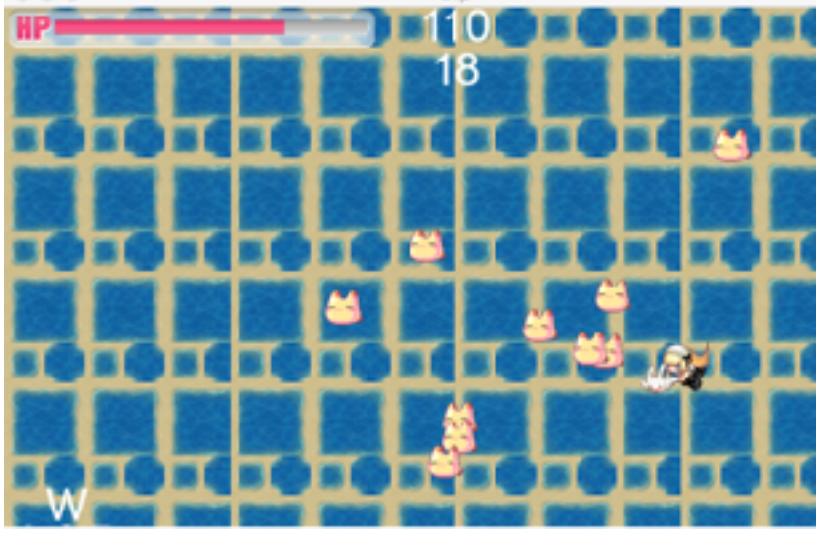
```cpp
  auto playerBody = PhysicsBody::createBox(player->getContentSize(),PhysicsMaterial(100.0f,1.0f,0.5f));
    playerBody->setCategoryBitmask(0x01);
  playerBody->setCollisionBitmask(0x01);
  playerBody->setContactTestBitmask(0x01);
  playerBody->setDynamic(false);
  player->setPhysicsBody(playerBody);


  this->addChild(player, 2);

  ball = Sprite::create("ball.png");
  ball->setPosition(Vec2(xpos, player->getPosition().y + ball->getContentSize().height*0.1f - 25));
  ball->setScale(0.1f, 0.1f);
    // ‚Ê÷√«Ùµ╔∏'ÄÄ Ù;-'
  // Todo
    auto ballBody = PhysicsBody::createBox(ball->getContentSize(),PhysicsMaterial(100.0f,1.0f,0.5f));
    ballBody->setCategoryBitmask(0x03);
    ballBody->setCollisionBitmask(0x03);
    ballBody->setContactTestBitmask(0x03);
    ballBody->setGravityEnable(false);
    ballBody->setRotationEnable(false);
    ball->setPhysicsBody(ballBody);
```

（5）消砖头、游戏结束、（bonus）结束产生粒子效果

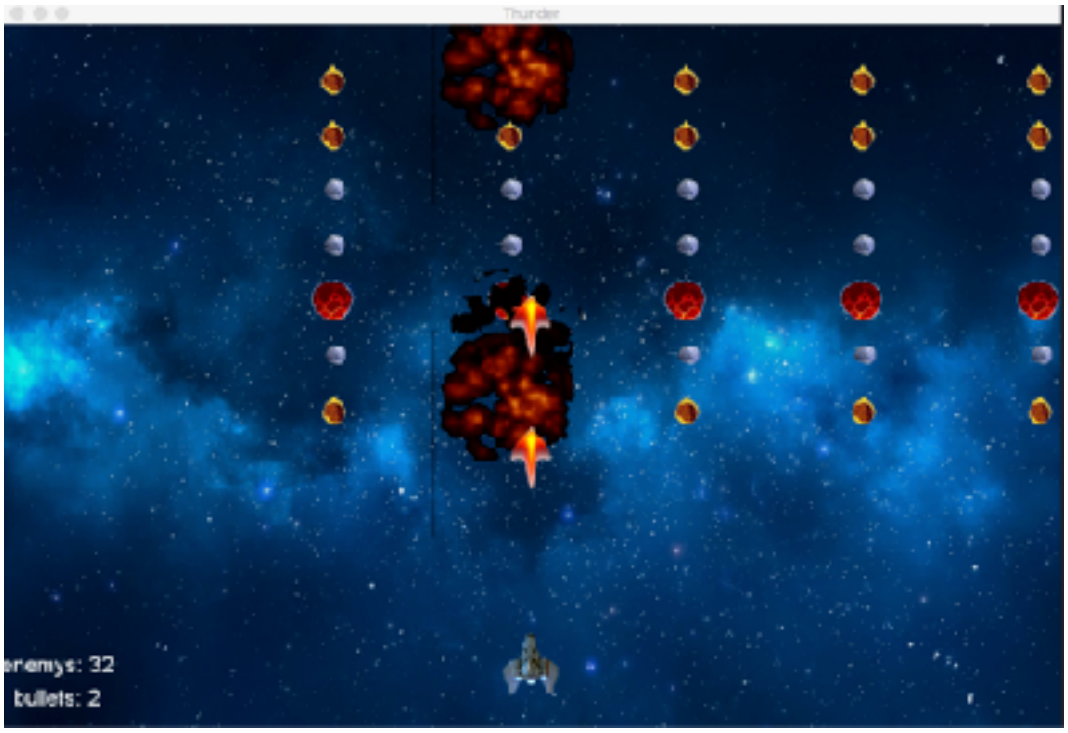
```cpp
// Todo
bool HitBrick::onConcactBegin(PhysicsContact & contact) {
    auto c1 = contact.getShapeA()->getBody()->getNode();
  auto c2 = contact.getShapeB()->getBody()->getNode();


    if (c1->getTag() == 2) {
        c1->removeFromParentAndCleanup(true);
    }else if(c2->getTag() == 2){
        c2->removeFromParentAndCleanup(true);
    }else if(c1->getTag() == 3){
        ParticleFireworks* fireworks = ParticleFireworks::create();
        auto pos = c1->getPosition();
        fireworks->setPosition(Vec2(pos.x,pos.y+40));
        this->addChild(fireworks,2);
        GameOver();
    }else if(c2->getTag() == 3){
        ParticleFireworks* fireworks = ParticleFireworks::create();
        auto pos = c1->getPosition();
        fireworks->setPosition(Vec2(pos.x,pos.y+40));
        this->addChild(fireworks,2);
        GameOver();
    }
```

三.关键步骤截图

1.横版游戏



2.小蜜蜂



3.打砖块

四.亮点与改进（可选）

1.横版游戏

增加了本地的存储，代码部分以及效果截图参见上文。

2.小蜜蜂

添加新的陨石
鼠标移动飞船、点击发射子弹
显示敌人与子弹数目正确
代码部分和效果截图参见上文

3.打砖块

在游戏结束时添加粒子效果，代码和效果截图参见上文

五.遇到的问题

1.Tilemap的导入失败

基本就是路径的问题，解决方法就是把对应的素材放进resources文件夹中并生成，重新导入就没有问题了

2.小蜜蜂结束爆炸动画不能在pause前执行完毕

解决方法是设置tag并且get用来检测是否在执行爆炸动画，执行完毕后在pause。

3..打砖块游戏无法结束

细致分析，其实是ship的tag设置在了刚体上，所以一开始没有get到，解决方法在ship的node上重新设置了一个tag。

六.思考与总结

这周主要学习了调度帧、本地存储、监听事件、音效、物理引擎、粒子系统等方面的知识。作业布置的小游戏对于所学的内容起到了很好的思考与巩固作用，现在基本上已经可以使用cocos2d做一些有趣的小游戏了，无论是音效、监听，还是动画之类的，基本上需要的知识已经趋近于完备了。cocos我觉得相对比unity和之前学的uwp，作业更容易做一些。不过临近期末，真的很想吐槽一下这门课程相对于整个学期作业量还是挺大的。每周都有实验作业，三周交一次报告，期中期末还分别有两个项目。对于课程繁忙的本学期也是在做的很辛苦，不过收获还是满满的，这也能够让我欣慰一些。