现代操作系统应用开发

期末 Project

[英雄战斗]

实验报告书

队　　长：肖雨蓓

队　　员：谢苑珍 吴锐红 吴若琳

一、项目分工

学号	名字	角色	班级	职责	贡献
15331324	肖雨蓓	组长	晚上班	主要负责网络访问，魔法攻击、部分文档	25%
15331315	吴锐红	组员	晚上班	主要负责音乐、开始结束、粒子系统、部分文档	25%
15331327	谢苑珍	组员	晚上班	主要负责人物动画、人物移动、文档排版整合	25%
15331316	吴若琳	组员	晚上班	主要负责 tilemap、随机炸弹、部分文档	25%

二、开发环境

Visual Studio 2015 + IntelliJ IDEA + Tiled

三、项目阐述

3.1.名称

英雄战斗

3.2.简介

该产品名称为“英雄战斗”，是一款双人实时对战游戏；

游戏界面中有男女两个英雄人物，双方各有 100 点血，并且有物理攻击和魔法攻击两种攻击功能，其中魔法攻击只能够有两次的试用机会，物理攻击需要靠近对方才能够攻击到对方，若攻击到对方则减，魔法攻击能够追踪敌人的位置进行攻击，若魔法攻击攻击到对方则减少 30 滴血，其中还会有随机炸弹这样的障碍物，如果不小心被随机炸弹砸伤，则会减少 20 滴血，谁先死亡则对方获胜，若最终时间到达，谁的血多则谁获胜，若一样则平局。

3.3.功能

两个玩家对战，每个玩家拥有物理和魔法两种攻击。

玩家一用 AWSD 控制人物移动，K 为魔法攻击，J 为物理攻击；

玩家二用方向键控制人物移动，shift 为魔法攻击，/为物理攻击。

同时人物所在场景会有随机的障碍“炸弹”，人物格斗的同时还要注意躲避这些“炸弹”的攻击。在规定时间内有一方没血，另外一方就胜利;过了规定时间的，以血多的一方为胜利。

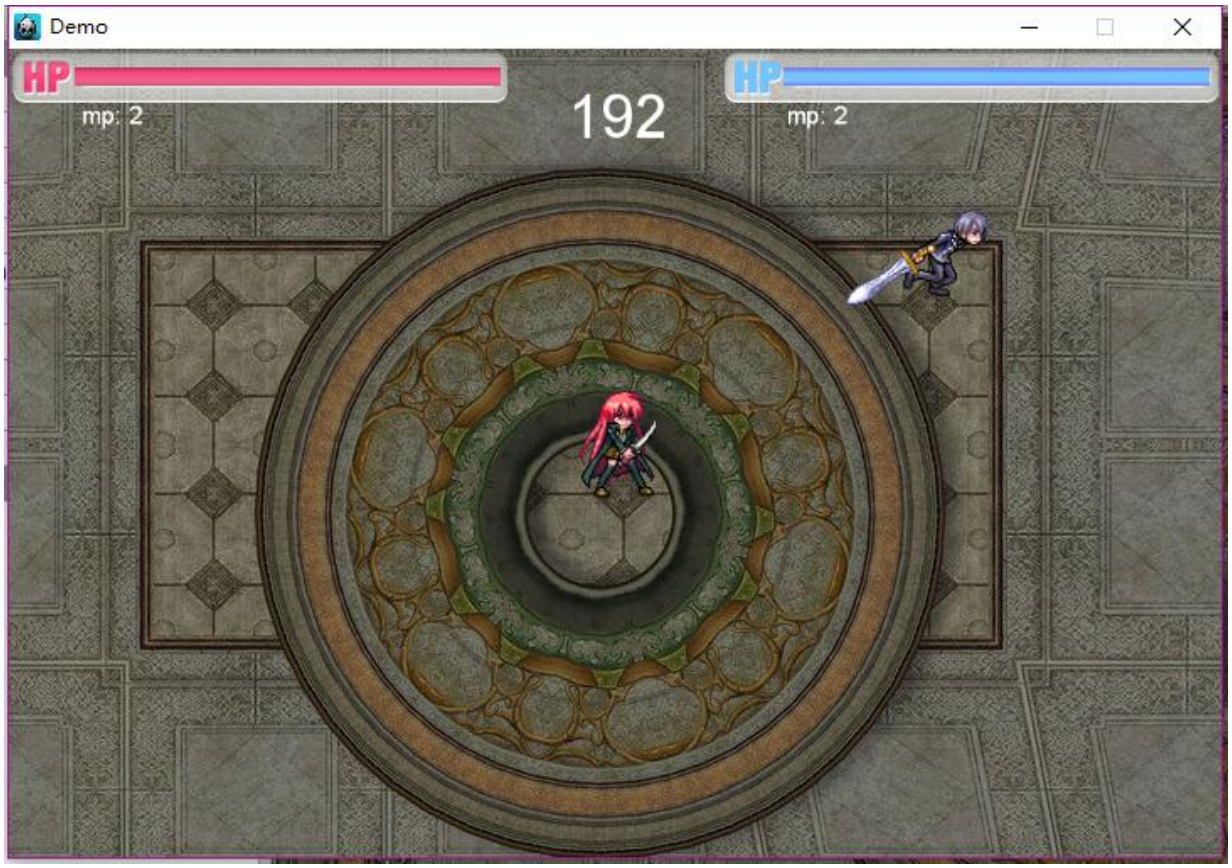
3.4.亮点

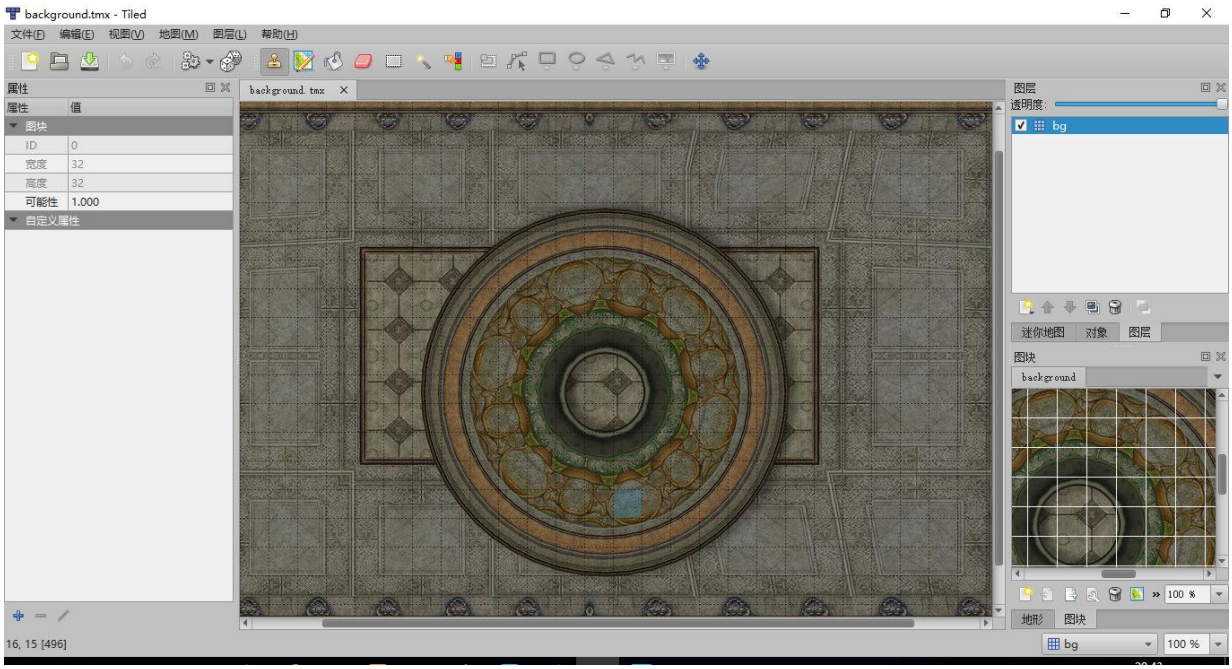
游戏形式为双人格斗游戏。玩家不仅有普通的物理攻击模式，还有强大的魔法攻击模式。玩家在游戏过程中，不仅要能攻能防，还要灵活避开随时会出现的炸弹。看谁能在有限的时间内获胜。

四、项目展示

4.1.Tiled Map

游戏背景采用 Tiled 制作。



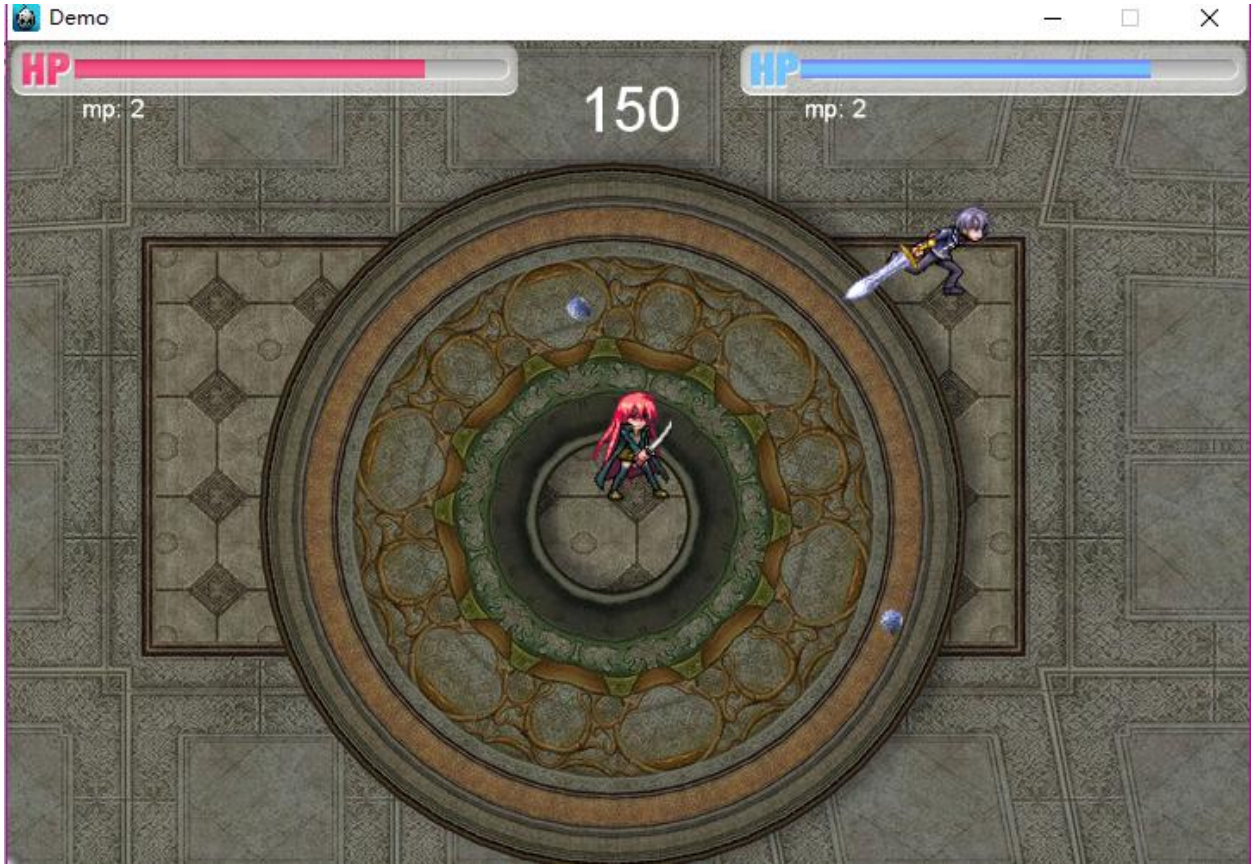


相应代码：

```
void HelloWorld::loadMap() {
    //导入瓦片地图
    TMXTiledMap* tmx = TMXTiledMap::create("map/background.tmx");
    //设置地图位置
    tmx->setPosition(visibleSize.width / 2, visibleSize.height / 2);
    //设置锚点
    tmx->setAnchorPoint(Vec2(0.5, 0.5));
    //适应大小
    tmx->setScale(Director::getInstance()->getContentScaleFactor());
    //添加到游戏图层中
    this->addChild(tmx, 0);
}
```

4.2.物理引擎随机产生炸弹

游戏会不定时产生炸弹，把炸弹放置在游戏边界，并给炸弹一个初速度。若玩家碰到炸弹，炸弹爆炸，玩家 HP 值减少。



相应代码，利用物理引擎刚体，设置掩码从而实现炸弹与玩家的碰撞：

```
void HelloWorld::addEnemy(Point loc) {
    auto stoneAnimation = Animation::createWithSpriteFrames(stoneFrames, 0.5);
    auto ani = Animate::create(stoneAnimation);
    auto stone = Sprite::create();
    stone->setTag(2);
    stone->runAction(RepeatForever::create(ani));
    addChild(stone, 3);

    stone = Sprite::create("stone2.png");
    stone->setPhysicsBody(PhysicsBody::createCircle(stone->getContentSize().height / 2));
    stone->setAnchorPoint(Vec2(0.5, 0.5));
    stone->setScale(0.5, 0.5);
    stone->setPosition(loc);

    //初速度
    stone->getPhysicsBody()->setVelocity(
        (Point(rand() % (int)(visibleSize.width - 100) + 50, rand() % (int)(visibleSize.height - 100) + 50) - loc));

    //撞击掩码 0110
    stone->getPhysicsBody()->setCategoryBitmask(0x06);
    stone->getPhysicsBody()->setContactTestBitmask(0xffffff);
    //设置随机炸弹的位置
    stone->getPhysicsBody()->setAngularVelocity(CCRANDOM_0_1() * 10);
    enemys.pushBack(stone->getPhysicsBody());
    addChild(stone);
}
```

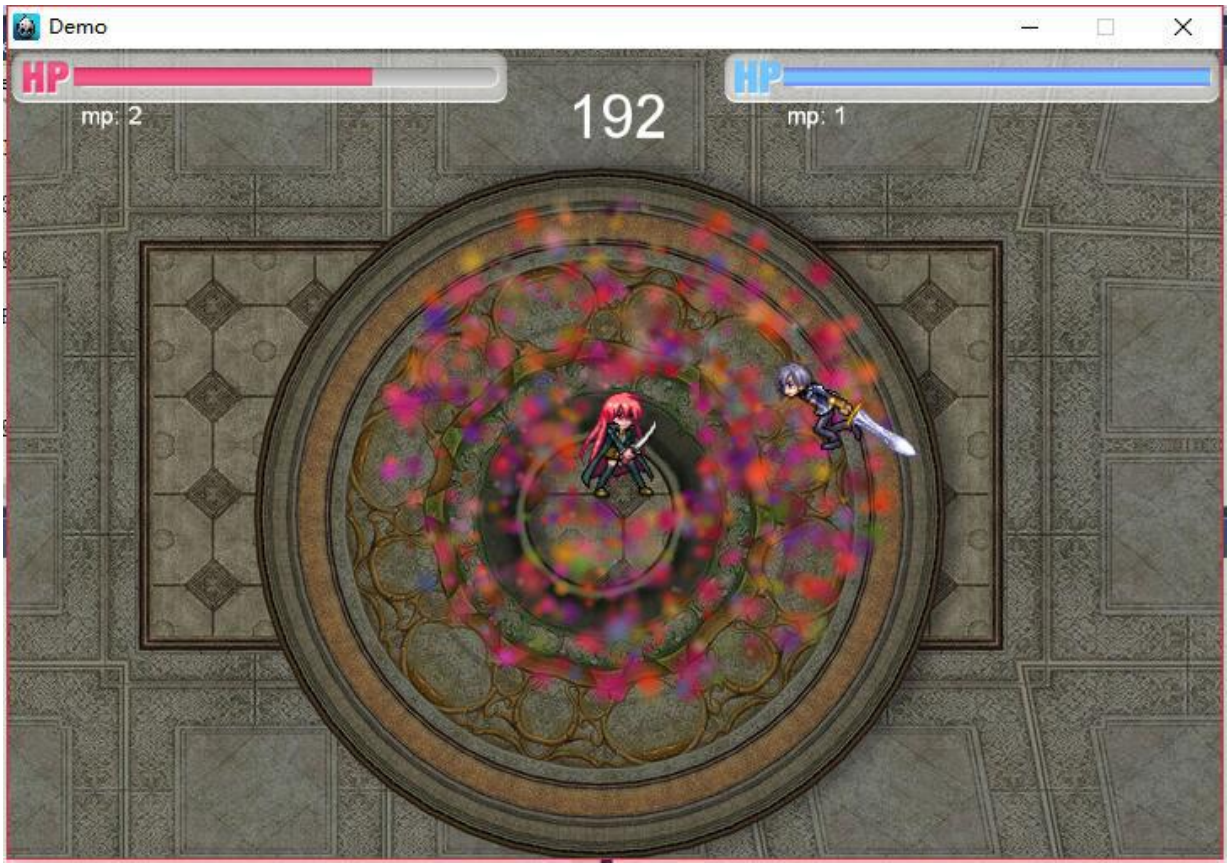
4.3.人物动画

人物帧动画切割

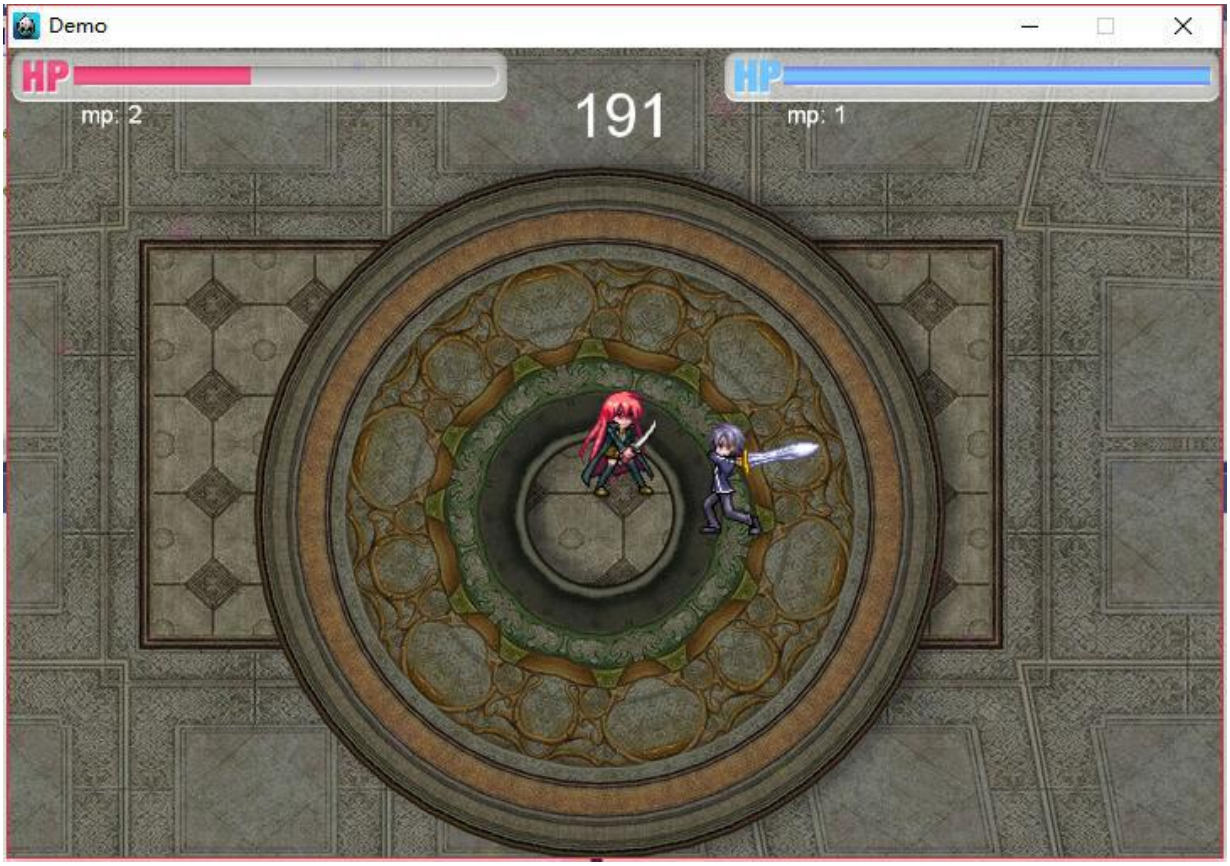
```
//设置第二个精灵动画
//创建一张贴图
auto texture2 = Director::getInstance()->getTextureCache()->addImage("$hero.png");
//从贴图中以像素单位切割，创建关键帧
auto frame2 = SpriteFrame::createWithTexture(texture2, CC_RECT_PIXELS_TO_POINTS(Rect(0, 0, 64, 66)));

//攻击动画
attack2.reserve(8);
for (int i = 0; i < 8; i++) {
    auto texture2_temp = Director::getInstance()->getTextureCache()->addImage("$hero_beat.png");
    auto frame = SpriteFrame::createWithTexture(texture2_temp, CC_RECT_PIXELS_TO_POINTS(Rect(143* i, 0, 143, 113)));
    attack2.pushBack(frame);
}

//运动动画
auto texture3 = Director::getInstance()->getTextureCache()->addImage("$hero_forward.png");
run2.reserve(8);
for (int i = 0; i < 8; i++) {
    auto frame = SpriteFrame::createWithTexture(texture3, CC_RECT_PIXELS_TO_POINTS(Rect(85 * i, 0, 85, 67)));
    run2.pushBack(frame);
}

//静止
auto standbyAnimation = Animation::createWithSpriteFrames(run2, 0.1f);
auto standbyAnimate = Animate::create(standbyAnimation);
player2->runAction(RepeatForever::create(standbyAnimate));
```

创建人物移动动画

```
//player1移动的动画
auto moveAnimationPlayer1 = Animation::createWithSpriteFrames(run, 0.1f);
auto moveAnimate = Animate::create(moveAnimationPlayer1);

//player2移动的动画
auto moveAnimationPlayer2 = Animation::createWithSpriteFrames(run2, 0.1f);
auto moveAnimate2 = Animate::create(moveAnimationPlayer2);
```



创建攻击动画效果



4.4.事件处理人物移动逻辑

利用键盘监听事件来实现人物移动逻辑；玩家一通过 AWSD 安吉按键进行左上下右移动，而玩家二通过方向键进行移动。引入状态和调度器来进行人物移动的判断——用数字 0 1 2 3 4（4 为无效状态） 表示人物当前的状态。数字 0 表示它按下按钮；如果玩家松开按钮状态变为无效状态（4）；如果状态为 0 的时候设置人物一直移动。

添加监听事件：

```
void HelloWorld::addKeyboardListener() {

    auto listenKB = EventListenerKeyboard::create();
    listenKB->onKeyPressed = CC_CALLBACK_2(HelloWorld::onKeyPressed, this);
    listenKB->onKeyReleased = CC_CALLBACK_2(HelloWorld::onKeyReleased, this);
    auto dispatcher = Director::getInstance()->getEventDispatcher();
    dispatcher->addEventListenerWithSceneGraphPriority(listenKB, player);
}
```

玩家一人物移动按键设置：（玩家二类似）

```
void HelloWorld::onKeyPressed(EventKeyboard::KeyCode command, Event* event) {
    //player1
    lastKey = 'D';
    lastKey2 = 'D';
    switch (command) {
    case cocos2d::EventKeyboard::KeyCode::KEY_A:
        if (lastKey != 'A') {
            player->setFlipX(true);
        }
        lastKey = 'A';
        player->setPosition(player->getPositionX() - 1, player->getPositionY());
        status = 0;
        break;
    case cocos2d::EventKeyboard::KeyCode::KEY_W:
        player->setPosition(player->getPositionX(), player->getPositionY() + 1);
        status = 1;
        break;
    case cocos2d::EventKeyboard::KeyCode::KEY_S:
        player->setPosition(player->getPositionX(), player->getPositionY() - 1);
        status = 3;
        break;
    case cocos2d::EventKeyboard::KeyCode::KEY_D:
        player->setFlipX(false);
        lastKey = 'D';
        player->setPosition(player->getPositionX() + 1, player->getPositionY());
        status = 2;
        break;

    case cocos2d::EventKeyboard::KeyCode::KEY_J:
        playerAttack();
        break;
    case cocos2d::EventKeyboard::KeyCode::KEY_K:
        fire1();
        break;
```

键盘放开设置标志位为 4——无效位

```
void HelloWorld::onKeyReleased(EventKeyboard::KeyCode code, Event* event) {
    status = 4;
    status2 = 4;
}
```

周期性调度：

```
//周期性更新人物移动
schedule(schedule_selector(HelloWorld::continueKeyUpdate), 0.05f, kRepeatForever, 0);
```

人物移动逻辑：

```
switch (status) {

    //左
case 0:
    if (player->getPosition().x - 25 > origin.x) {
        player->setPosition(player->getPositionX() - 8, player->getPositionY());
        player->runAction(Sequence::create(moveAnimate, NULL));
    }
    break;
    //上
case 1:
    if (player->getPosition().y + 25 < origin.y + visibleSize.height) {
        player->setPosition(player->getPositionX(), player->getPositionY() + 8);
        player->runAction(Sequence::create(moveAnimate, NULL));
    }
    break;
    //右
case 2:
    if (player->getPosition().x + 25 < origin.x + visibleSize.width) {
        player->setPosition(player->getPositionX() + 8, player->getPositionY());
        player->runAction(Sequence::create(moveAnimate, NULL));
    }
    break;
    //下
case 3:
    if (player->getPosition().y - 25 > origin.y) {
        player->setPosition(player->getPositionX(), player->getPositionY() - 8);
        player->runAction(Sequence::create(moveAnimate, NULL));
    }
    break;
default:
    break;
}
```

4.5.音乐与音效

加载背景音乐和音效，预加载可以加快运行时候的效率。

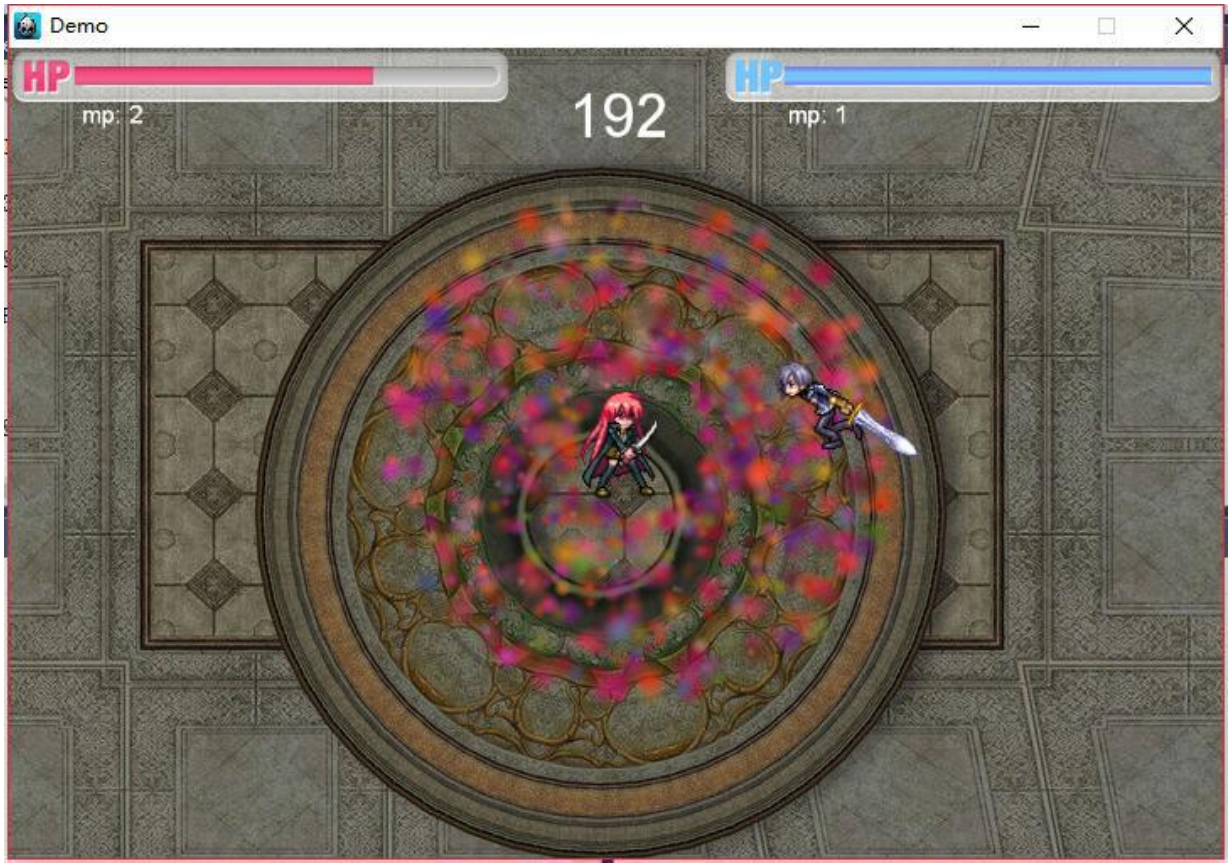
```
//导入音乐和音效
preloadMusic();
playBGM();
```

```
void HelloWorld::preloadMusic() {
    auto audio = SimpleAudioEngine::getInstance();

    audio->preloadBackgroundMusic("music/playing_BGM.mp3");
    audio->preloadEffect("music/shoot.mp3");
    audio->preloadEffect("music/boom.mp3");
    audio->preloadEffect("music/beat.mp3");
}
void HelloWorld::playBGM() {
    SimpleAudioEngine::getInstance()->playBackgroundMusic("music/playing_BGM.mp3", true);
}
```

4.6.粒子系统

爆炸效果 使用了系统自带的爆炸效果

```
//爆炸特效
ParticleExplosion* exp = ParticleExplosion::create();
exp->setPosition(bullet->getPosition());
exp->setDuration(0.3f);
addChild(exp);
```



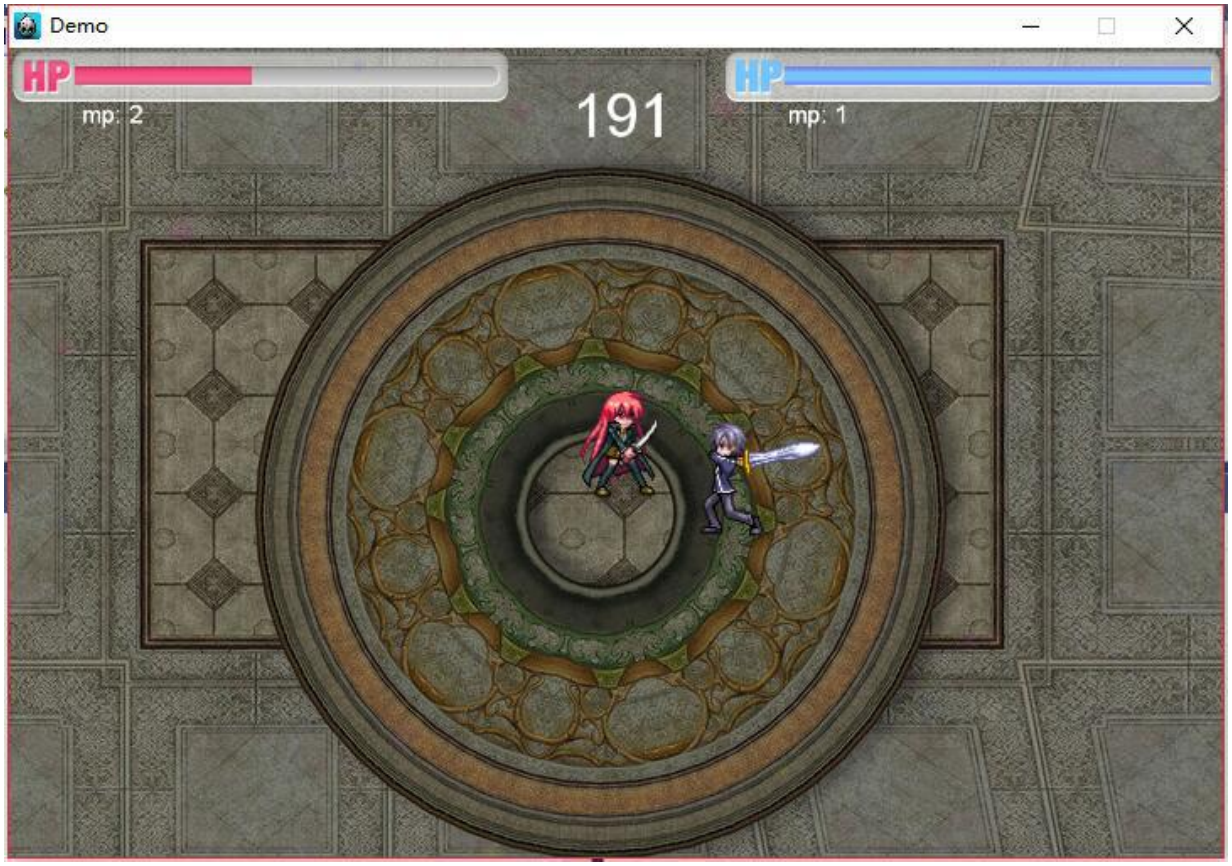
4.7.人物互相攻击

先判断是否在攻击范围内

```
//判断两个人物是否在攻击范围
bool HelloWorld::canAttack() {

    //定义攻击范围
    Rect Player_Rect = player->getBoundingBox();
    Rect attack_Rect = Rect(Player_Rect.getMinX() - 40, Player_Rect.getMinY(), \
        Player_Rect.getMaxX() - Player_Rect.getMinX() + 80, Player_Rect.getMaxY() - Player_Rect.getMinY());

    if (attack_Rect.containsPoint(player2->getPosition())) {
        return true;
    }
    return false;
```

进行攻击，同时播放音效

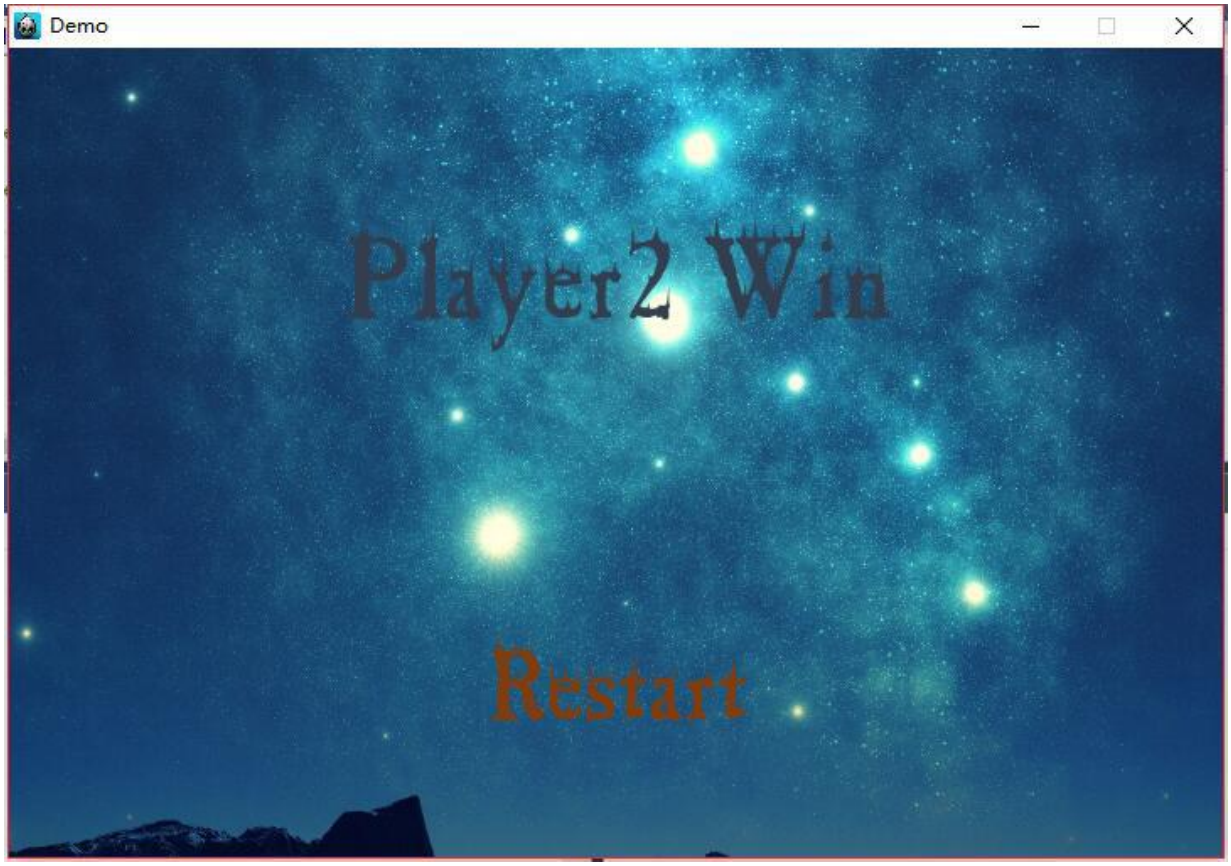
```
void HelloWorld::playerAttack() {

    //音效

    playBeatM();


    auto Attack = Animation::createWithSpriteFrames(attack, 0.1f);

    auto Animate = Animate::create(Attack);

    player->runAction(Sequence::create(Animate, NULL));

    if (canAttack()) {

        pT2->runAction(ProgressTo::create(0.5f, pT2->getPercentage() - 10));

    }
```



4.8.游戏结束

判断哪一方获胜并返回

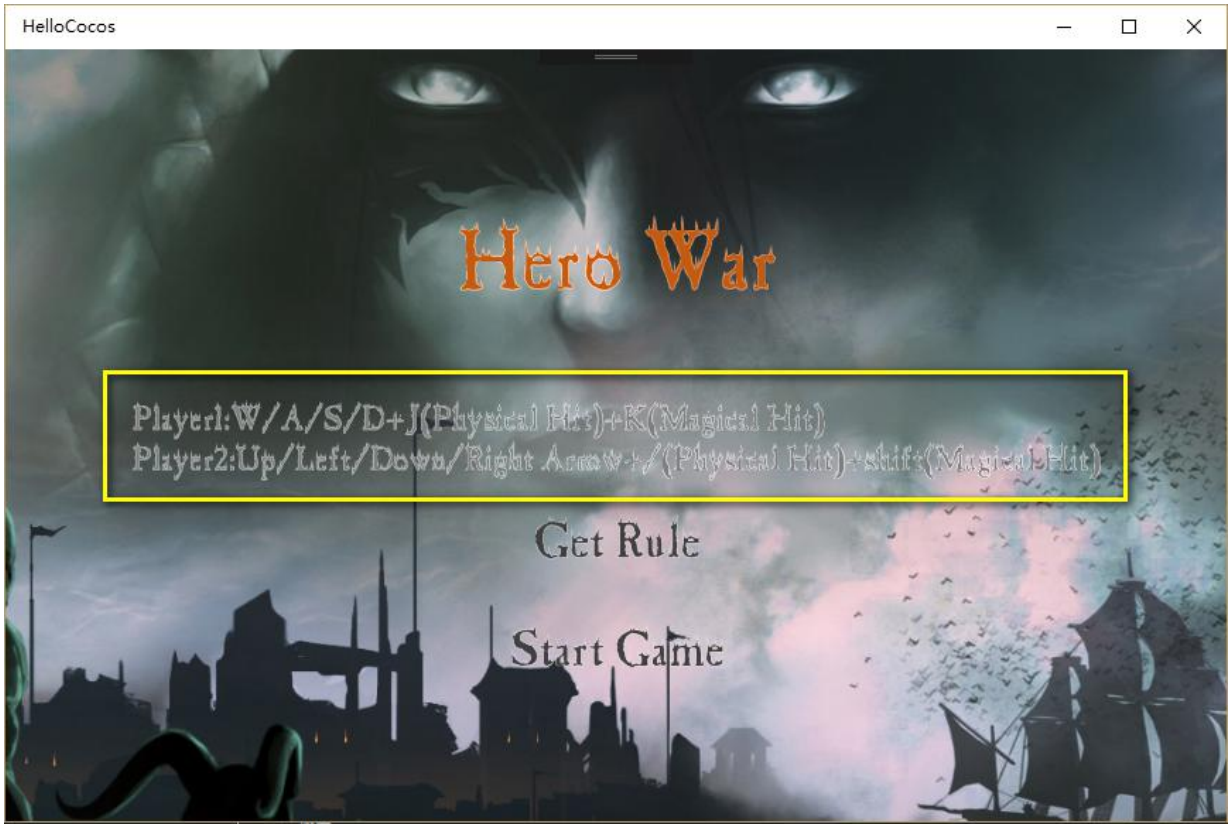
```
void Global::setWinner(int a) {
    if (a == 0) winner = "Player2";
    else if (a == 1) winner = "Both";
    else winner = "Player1";
}

string Global::getWinner() {
    return winner;
}
```



4.9.网络访问：

点击下图指向的 Get Rule，程序将会从服务端请求规则：

规则请求到，传回程序，程序进行处理，将规则显示在中间，如下图：



4.10.网络访问的知识点：

在 cocos2d 中的主要函数如下，向本地服务端发送请求，请求得到 rule，并且将 onHttpComplete

设置为回调函数：

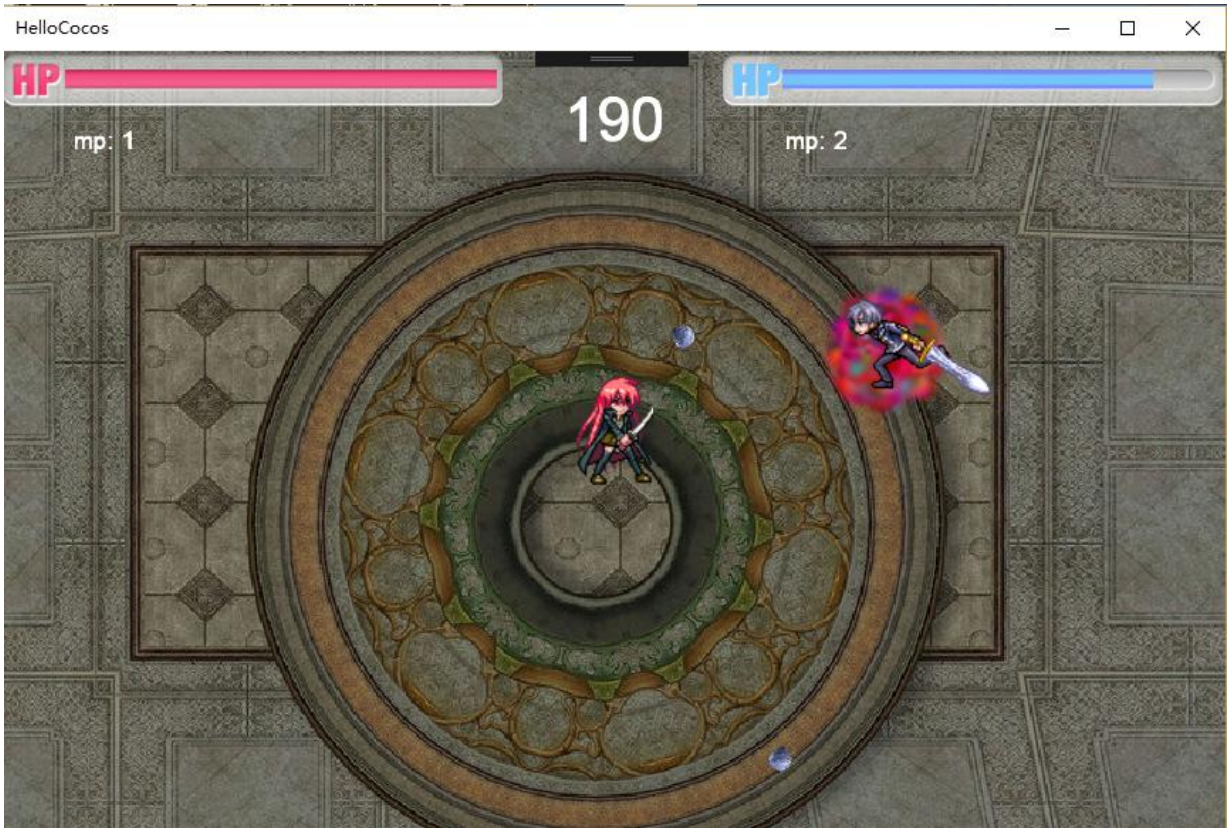
```
void MenuSence::getRuleListener(Ref *pSender) {

    HttpRequest* request = new HttpRequest();
    request->setRequestType(HttpRequest::Type::GET);

    int number = rand();
    std::stringstream ss;
    ss << number;
    std::string numberToString;
    ss >> numberToString;
    std::string url = "http://localhost:8080/rule";
    request->setUrl(url.c_str());
    request->setResponseCallback(CC_CALLBACK_2(MenuSence::onHttpComplete, this));

    std::vector<std::string> headers;
    headers.push_back("Content-Type: application/x-www-form-urlencoded; charset=UTF-8");
    request->setHeaders(headers);

    cocos2d::network::HttpClient::getInstance()->send(request);
    request->release();
}
```

下面就是回调函数处理 response 的数据，将规则显示到游戏界面中：

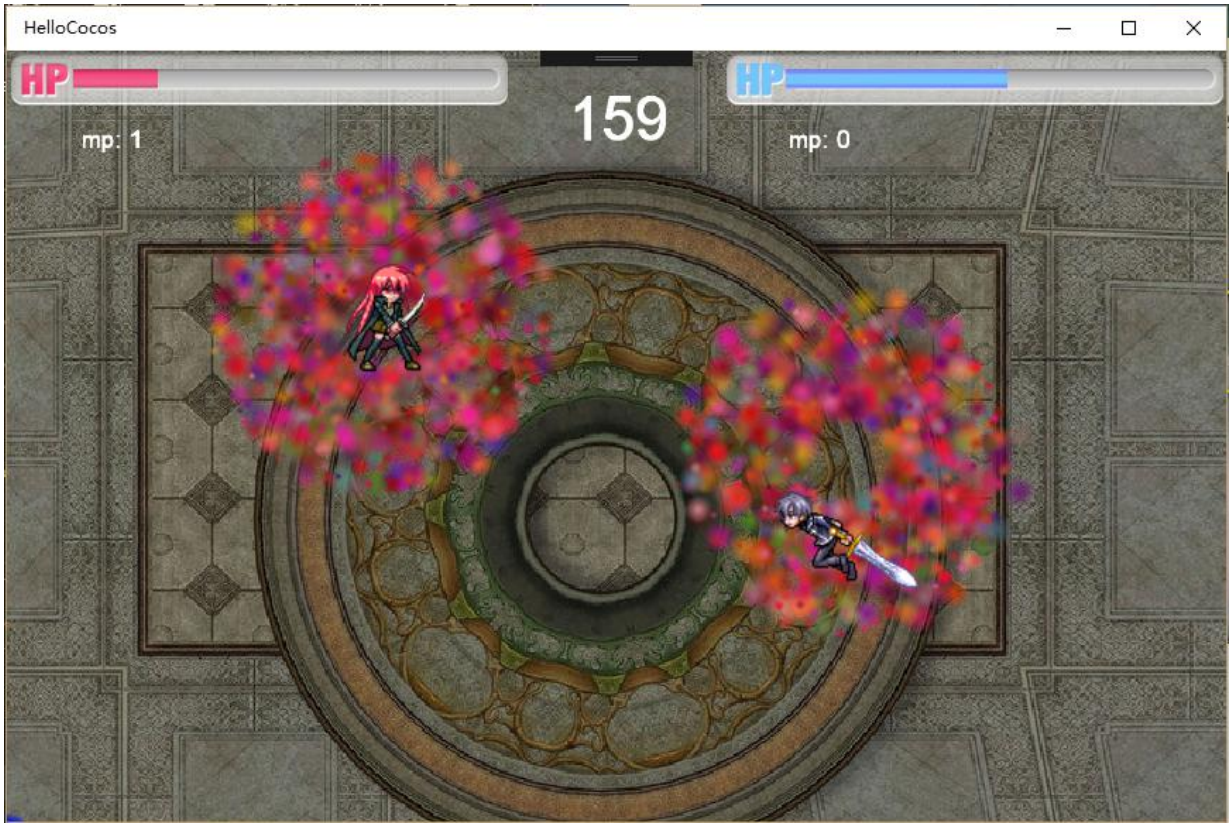
```
void MenuSence::onHttpComplete(HttpClient *sender, HttpResponse *response) {

    Size visibleSize = Director::getInstance()->getVisibleSize();
    Vec2 origin = Director::getInstance()->getVisibleOrigin();
    if (rule_field != NULL) {
        rule_field->removeFromParent();
        rule_field = NULL;
    }
    std::vector<char> *buffer = response->getResponseData();
    string rule = Global::toString(buffer);
    rule_field = TextField::create(rule, "fonts/Pan.ttf", 30);
    rule_field->setPosition(Size(visibleSize.width / 2, visibleSize.height / 2+10));
    rule_field->setFontSize(20);
    rule_field->setTextColor(ccc4(34, 139, 34, 255));
    this->addChild(rule_field, 2);

}
```

4.11.魔法攻击的编写

玩家一按下 K 键会向玩家二的方向发出魔法攻击，成功攻击到玩家二，玩家二的血量减 30，并且魔法攻击可以使用的次数在图中 mp 处显示：



玩家二按下 shift 键会向玩家一的方向发出魔法攻击，成功攻击到玩家一，玩家一的血量减 30，并且魔法攻击可以使用的次数在图中 mp 处显示：



4.12.调度器、粒子效果、音效在魔法攻击中的使用

玩家点击 K 或者 shift 发动魔法攻击：

```
case cocos2d::EventKeyboard::KeyCode::KEY_SHIFT:
    fire2();
    break;
```

添加魔法攻击 bullet 到场景中：

```
void HelloWorld::fire2()
{
    if (fire_count2 <= 0 || bullet2 != NULL) return;
    //音效
    playShoot();

    bullet2 = Sprite::create("bullet.png");
    bullet2->setPosition(player2->getPosition());
    addChild(bullet2, 1);
    fire_count2--;
}
```

周期性调度魔法攻击的函数：

```
//倒计时周期性调用调度器
schedule(schedule_selector(HelloWorld::hitByMagic), 0.05f, kRepeatForever, 0);
```

如果魔法攻击存在并且击中了目标，则发生爆炸，并且目标血量减少，以下是部分截图：

```
void HelloWorld::hitByMagic(float dt) {
    //发动魔法攻击
    if (bullet != NULL) {
        bullet->setPosition(bullet->getPosition() + (player2->getPosition() - player->getPosition())*0.1);
        //击中目标
        if (bullet->getPosition().getDistance(player2->getPosition()) < 20) {
            //爆炸特效
            ParticleExplosion* exp = ParticleExplosion::create();
            exp->setPosition(bullet->getPosition());
            exp->setDuration(0.3f);
            addChild(exp);
            //音效
            playBoomM();
            removeChild(bullet);
            bullet = NULL;
```

```
    if (bullet != NULL && (bullet->getPositionX() > visibleSize.width || bullet->getPositionX() < 0 || bullet->getPositionY() > visibleSize.height || bullet->getPositionY() < 0)) {
        removeChild(bullet);
        bullet = NULL;
    }
}
```

在魔法攻击后更新魔法攻击的数量，以下是部分截图：

```
// 更新魔法攻击数量
string str = "mp: ";
if (fire_count == 2) str = str + "2";
else if (fire_count == 1) str = str + "1";
else str = str + "0";
count_field = Label::create(str, "Arial", 15);
count_field->setPosition(Size(visibleSize.width / 10 - 10, visibleSize.height - 40));
this->addChild(count_field, 1);
```

五、项目难点及解决方案

5.1 人物图片切割——人物动画不连贯

人物动画切割，可能不是很上手，还有找到的素材图片大小有差异导致人物动画不连贯。

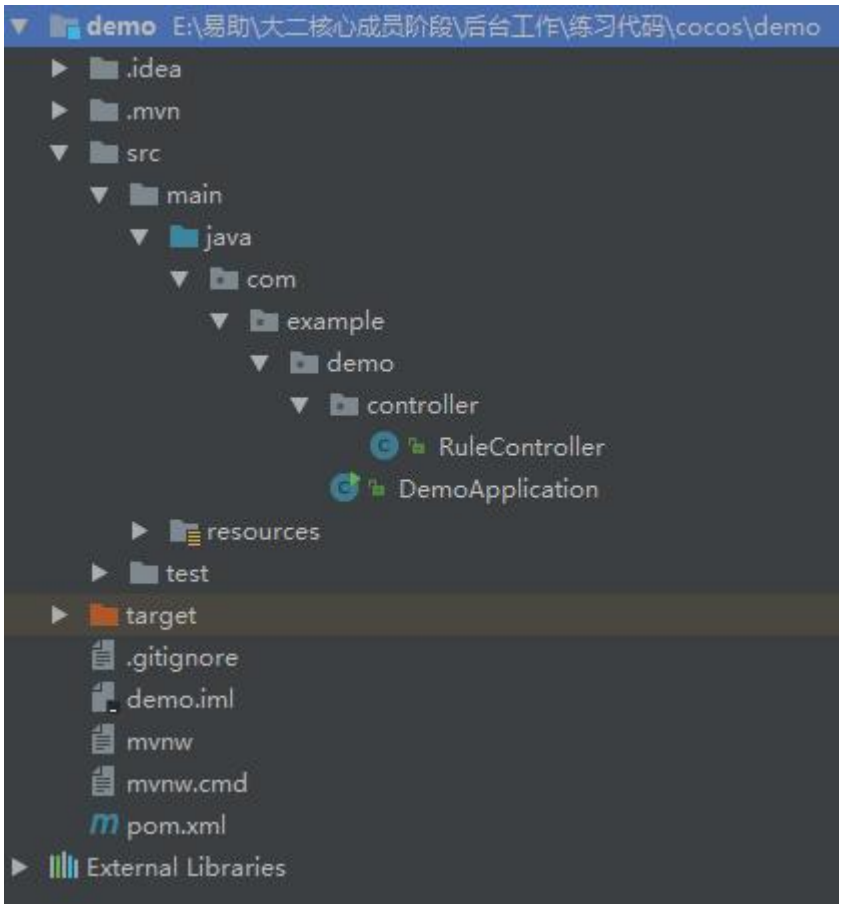
5.2.人物攻击

这个算一个重点，主要在判断的阶段，先获取要攻击角色的 boundingbox，再根据这个范围 合理设置攻击范围的 boundingbox，最后判断被攻击角色的位置是否包含在攻击范围的 boundingbox 里面。

5.3.网络访问

网络访问部分在 cocos2d 上代码不难，难点主要是用 spring boot 框架写了一个简单的 java 服务端，学习了一下怎么写 spring boot 框架的服务端，其他的部分就能够比较快地解决了：

下图是 Spring boot 项目的结构：



Spring boot 创建的服务端，就一个业务逻辑处理函数，相应/rule 的 url 的 get 请求，直接返回规则：

```
@RestController
public class RuleController {
    @RequestMapping("/rule")
    public String index() {
        return "Player1:W/A/S/D+J(Physical Hit)+K(Magical Hit)\nPlayer2:Up/Left/Down/Right Arrow+/(Physical Hit)+shift(Magical Hit)";
    }
}
```

六、项目总结

总的来说，由于经历了期中项目，有了一次团队合作开发一个小项目的经验，我们期末项目进展得相当快，从期末项目任务挂到网站上，我们就开始划分了编写代码的阶段，以及分配了各个阶段的任务，同时依旧采用码云的代码托管平台，团队高效协作，这一次期末项目将所有学过的 cocos2d 知识点均涵盖在其中，得到了最终的成品，具体的实现方式在项目展示部分已经展示，以下是团队成员各自的总结：

谢苑珍：

期末项目已经告一段落，通过这次项目，温习了下半学期所学的 cocos2d 游戏开发相关知识点，学会了协作开发流程，在上次期中项目的基础上更加熟悉了 git 的相关操作。感觉 git 遇到的坑还是比较多的，经常出现 push 不成功的情况。遇到这些情况的时候不能着急，对应错误点去网上查找对应的解决方法，一般都能解决。

有了第一次期中项目的基础，我们任务安排上出得很快。在刚开始学习 cocos2d 制作游戏得时候，我们就开始讨论了一些打算在期末制作的小游戏，当时考虑的有双人对战，跑酷等。综合考虑到我们学过的知识以及时间等原因，选择了双人对战游戏。整理好整体思路之后，算是确定了应用的目的已经相关的主要功能。在这个的基础上，组长进行了第一阶段的分工。分工的目的是并行，从完成时间上看，并行减少了周期长度，也让每个人都实时参与其中，深刻了解应用的每一细节，为后续的协作开发打下了坚实的基础。

在第二阶段——界面实现+需求实现方式明确中，我们讨论了许久需求具体实现方式，之前定下的许多实现方式都被推。当实际操作的时候才发现有一些实现方式过于复杂，有一些则完全不能如此实现。这个点也给我们未来的合作编程提供了需求实现方式确定的思路，必须先了解所使用的语言以及环境等能够提供的函数等，以免后续开发阶段过多修改需求导致思路紊乱。

在编程方面自己主要负责人物行走，人物精灵动画，人物攻击等。在实现上没有遇到特别的坑，不过素材找了好久是真的。找人物帧图片就找了整整一个晚上。在切割动画上根据大小和切割就好。整个分工合作有条不紊，很大程度上依赖于开发流程的合理性。这个项目能够按时按质完成，离不开大家的努力。

在这次项目中，自己深刻感受到遇到问题时不能过于急躁，百度了许久无果的东西其实如果能静下烦乱之心去思考，解决方法呼之欲出。有时候太过于依赖工具，反而被束缚了手脚，难以施展。勤思考，多动手，敢尝试，拨云见日或许就在前方。对自己在团队中的表现较为满意，算是圆满完成了任务。

吴若琳：

总的来说，此次项目我负责的开发部分没有太大的难题，基本都是沿用学过的知识。游戏开发给我最大的体会还是，游戏界面的设计、配乐和游戏美观程度很大程度上决定了该游戏的吸引力。因此，游戏素材的选取要更加用心，以提升游戏的总体质量。

吴锐红：

利用码云来存储代码版本，方便大家并行操作。分工较合理，各个成员负责的模块也清晰，整合的时候也更好整合。

肖雨蓓：

在这一次实验中，我负责的部分是最初的代码框架搭建，这个主要就是用之前写过的那个单人游戏进行地搭建，然后就是网络访问部分以及魔法攻击的编写部分，其中网络访问部分涉及到的知识点就是 cocos 网络 http 请求响应等的处理，然后还有用 spring boot 框架写了一个简单的服务端，至于魔法攻击部分，主要就是用调度器每隔一段时间调用魔法攻击判断是否有魔法攻击，其中魔法攻击是玩家点击 K 或者 shift 按钮自己发出的，如果发出了魔法攻击，则有相应的音效、爆炸效果等等；

因为很多之前都写过所以大家写的部分也都比较轻松，我写的部分也比较轻松，就只是涉及到了一个服务端编写的问题，当初在想应该如何实现这个网络访问，能够有效地融入游戏的网络免费 api 太少了，比如查询天气字典查询等等，完全和这个不沾边，因此我就想着自己写一个服务端，由于有 spring boot 框架的帮助，所以写的很快，就是看一下如何用 spring boot 构建项目就好了，而写的代码部分就一个小小的函数；

因此在这一次项目中，温习了之前的知识，也学习了一点新的东西，同时这一次我们团队效率很高，从任务布置下来就开始了工作，零零碎碎用了差不多两三周的时间基本上完成了，感觉很不错，都没有耽误复习，在后期心理也会比较放松，因为没有这个项目的压力压在那儿了。

同时在团队协作方面，我们因为经过了期中项目，所以还是利用期中项目的方法，在码云上进行代码托管，效率很高，我们也更加熟练！