现代操作系统应用开发实验报告

学号： 15331242　　**班级　　：** 周四早上班

姓名： 明友芬　　**实验名称：** homework12

一．参考资料

http://www.bkjia.com/Androidjc/820936.html

http://www.jellythink.com/archives/752

http://www.cocos.com/doc/tutorial/show?id=2455

二．实验步骤

（1）新建一个项目将上一次的程序拷贝过来，到初始状态，更改一下第一次做的内容，将上次的程序中的 init()函数更改一下，将 item 的实现部分内容重新放置到 createItem()函数里面，将设置各种层的程序放到 createLayer()里面；

（2）将 Monster.cpp 和 Monster.h 两个文件导入到新的项目里面；

（3）实现 Monster.cpp 里面的各个函数，关于每个函数的实现只用考虑到怪物就可以了，不用考虑主函数那些；

（4）在 HelloWorld.cpp 里面实现创建怪物，被怪物攻击以及攻击怪物等函数；

（5）修改对应的 actionEvent 函数，以及一些需要修改的其它函数；

（6）Debug 程序。

三．实验结果截图

（1）本次实验的初始界面如图 1 所示：

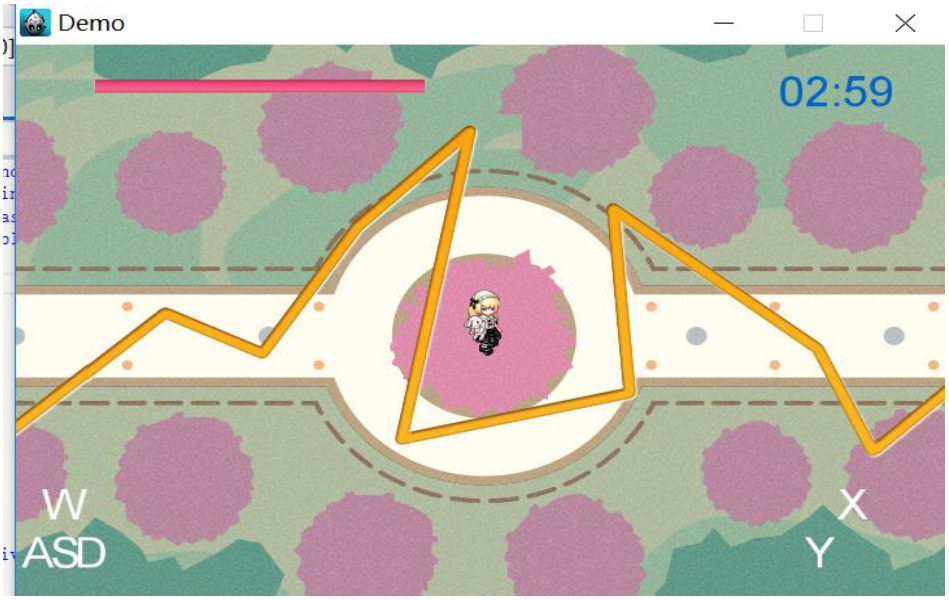

图 1

（2）随机产生角色，运行截图如下图 2 所示， 实现代码如下图 3 所示：

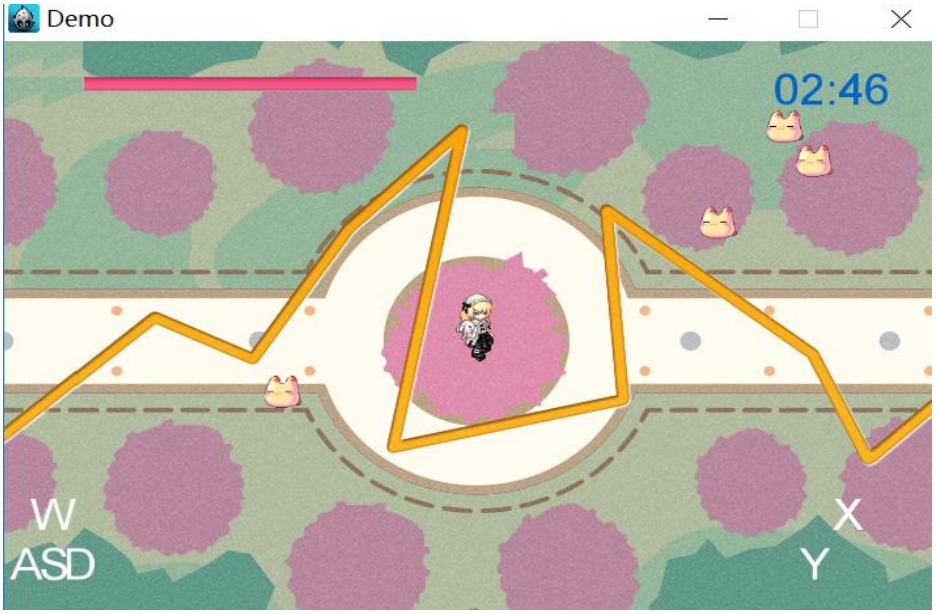

图 2

```
void HelloWorld::createMonster(float flo) {
    auto fac = Factory ::getInstance();
    auto m = fac->createMonster();
    //将怪物随机的放在屏幕上的一个位置上
    m->setPosition(random(origin.x, visibleSize.width), random(origin.y, visibleSize.height));
    this->addChild(m, 1);
    fac->moveMonster(player->getPosition(), 2.0f);
```

图 3

（3）角色去攻击怪物的实现截图如下图 4 所示，实现代码如图 5 所示：

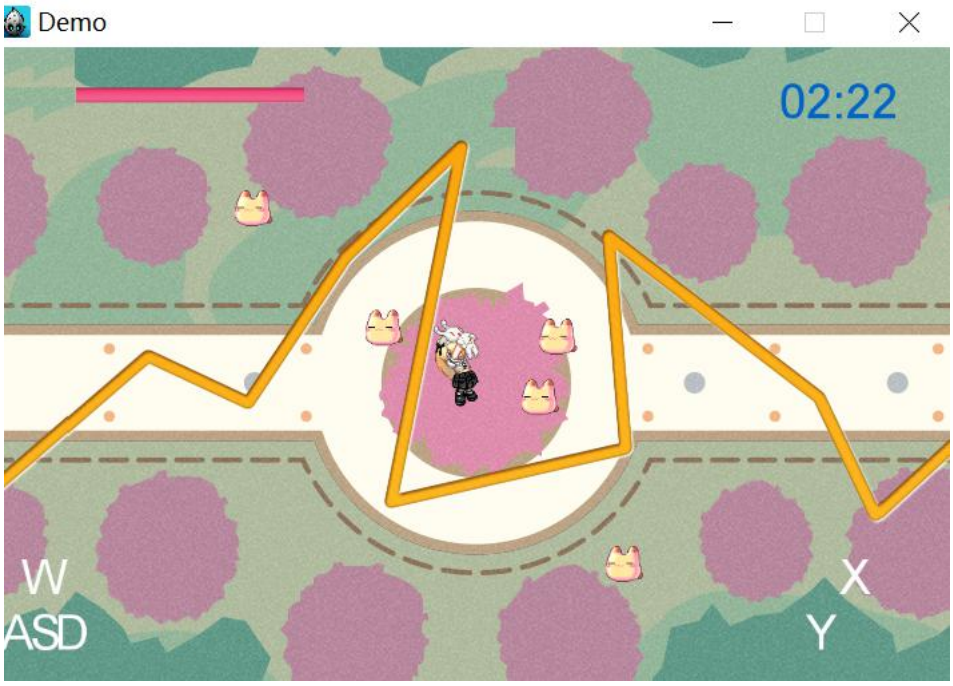

图 4

```
void HelloWorld::hitMonster() {
    auto fac = Factory::getInstance();
    //查看周围是否有怪物
    Sprite* collison = fac->hitMonster(player->getBoundingBox());
    if (collison != nullptr) {
        //将怪物移走
        fac->removeMonster(collison);
        if (pT->getPercentage() != 100) {
            //只要血量没有达到最大就充血
            pT->runAction(CCProgressTo::create(1.8f, pT->getPercentage() + 20));
        }
    }
}
```

图 5

（4）角色被怪物攻击的运行截图如下图 6 所示，实现代码如图 7 所示：

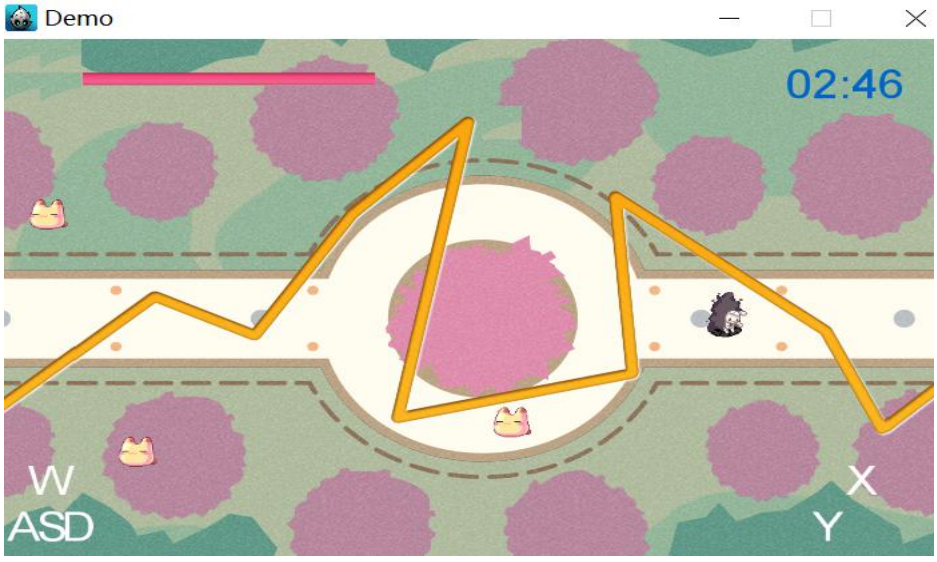

图 6

```
void HelloWorld::hitByMonster(float flo) {
    //isKeyDown();
    bool deaded = dead.contains(player->getSpriteFrame()) && player->getSpriteFrame() != frame0;
    bool attacked = attack.contains(player->getSpriteFrame()) && player->getSpriteFrame() != frame0;
    if (deaded == false && attacked == false) {
        //当攻击和死亡者两个事件有一个在进行的时候就不进行攻击事件
        auto fac = Factory::getInstance();
        //判断是否有碰撞
        Sprite* clloison = fac->collider(player->getBoundingBox());
        if (clloison != nullptr) {
            //在发生撞击事件之前要停止正在进行的所有操作
            player->stopAllActions();
            //将怪物移出屏幕
            fac->removeMonster(clloison);
            //在移出怪物的同时，精灵本身也要出血
            this->actionEvent(this, 'X');
        }
    }
```

图 7

（5）使用 tilemap 创建地图，如下图 8 所示：

```
//设置背景地图
TMXTiledMap* tmx = TMXTiledMap::create("map.tmx");
tmx->setPosition(visibleSize.width / 2, visibleSize.height / 2);
tmx->setAnchorPoint(Vec2(0.5, 0.5));
tmx->setScale(visibleSize.height / tmx->getContentSize().height);
```

图 8

（6）血条消失后，精灵也就一直处于死亡的状态，如下图 9 所示：

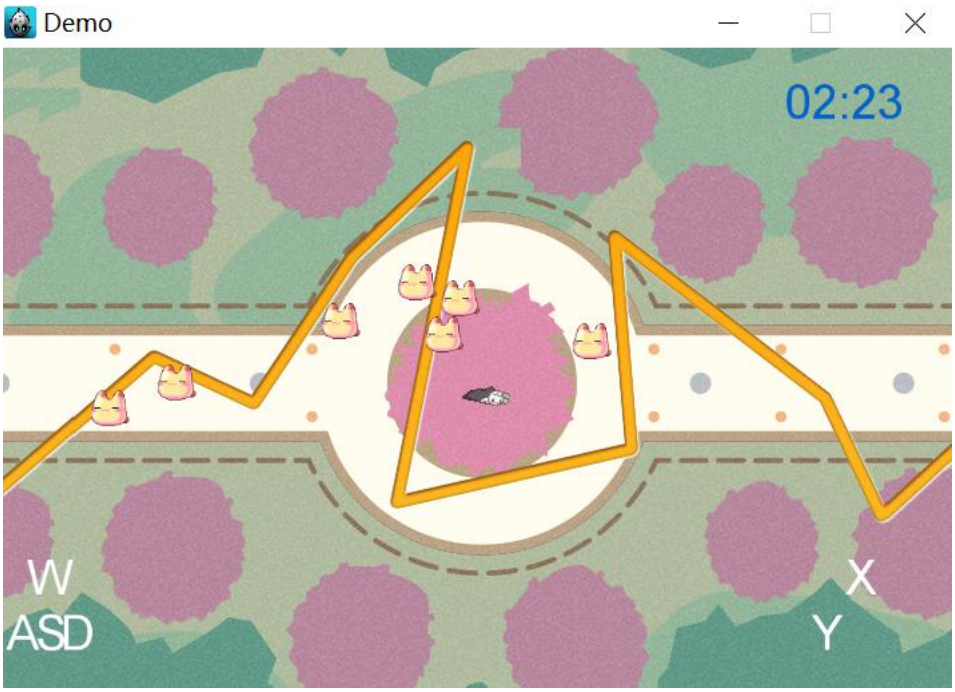

图 9

四．实验过程遇到的问题

（1）程序写完运行的时候发现不会自动产生怪物，在 HelloWorld.cpp 里面已经实现了 createMonster（）函数，后来经过查找，发现是因为在初始化函数里面没 有调用函数，所以将如下图 10 所示的程序添加进去就可以自动

产生怪物了。

```
schedule(schedule_selector(HelloWorld::createMonster), 3.0f, kRepeatForever, 0);
schedule(schedule_selector(HelloWorld::hitByMonster), 0.05f, kRepeatForever, 0);
```

图 10

（2）在实现的过程中发现，在攻击的时候也可以调用死亡函数，按理来说在同一个时间只能进行攻击事件，或者是死亡事件，后来在 hitByMonster 函数里面增加了一个判断函数，如下图 11 所示，程序就能比较正常的运行了。

```
void HelloWorld::hitByMonster(float flo) {
    //isKeyDown();
    bool deaded = dead.contains(player->getSpriteFrame()) && player->getSpriteFrame() != frame0;
    bool attacked = attack.contains(player->getSpriteFrame()) && player->getSpriteFrame() != frame0;
    if (deaded == false && attacked == false) {
        //当攻击和死亡者两个事件有一个在进行的时候就不进行攻击事件
```

图 11

五．思考与总结

本次实验学会了一个打怪兽的游戏，相对来说有些繁琐，首先要仔细看 demo，理清思路，然后在实现的时候要学会讲一个复杂的任务化为一个个比较小的任务，这样才不会自乱阵脚，在实现的时候思路比较清晰，而且这样的话，在调试程序的时候也才比较好调试。