现代操作系统应用开发实验报告

学号： 15331046　　**班级　：** 晚上班

姓名： 陈志扬　　**实验名称：** HW12

一．参考资料

请在这里列出对本实验有帮助你所参考的资料或者网站。

资料：老师上课所用的 PDF 资料，TA 师兄提供的作业指导文件

网站：定时器的使用

http://blog.csdn.net/zhanghefu/article/details/38466801

官方 API 文档 http://api.cocos.com/

问题解决查找 https://www.baidu.com/

二．实验步骤

请在这里简要写下你的实验过程。

① 首先，学习课件中的知识点，主要是 cocos2d-x 中的数据结构、本地存储和 Tilemap。其中数据结构主要有三种 Vector、Map、Value;本地数据存储主要有两种形式:UserDefault 和 SQLite;最后是学习瓦片地图与 TileMap 的使用以及如何利用软件 Tiled 创建瓦片地图。

② 接着，根据作业要求，可先把瓦片地图加载到场景中，然后实现随机产生怪物的功能，由于 TA 提供的 demo 已经实现了很多

接口，例如创建怪物 createMonster 方法，所以在随机创建怪物时我们主要用到 cocos2d-x 已实现的 random 方法。

③ 接下来实现怪物碰到角色后，角色掉血，我们可使用 Rect 类中的 containsPoint 来做简单的碰撞检测，我所做的是怪物的坐标在角色的 Rect 中时，发生碰撞。当发生碰撞时，移除碰撞的怪物，并让角色掉血。这部分 TA 也已提供大部分代码，我们只需要编写判断怪物和角色是否碰撞的函数方法 collider 即可。

④ 然后，角色可攻击怪物，当角色和怪物碰撞时，这里我所做的角色可以攻击前后方水平方向 40 内的怪物，打到怪物后角色回血，怪物消失。

⑤ 移除怪物的实现思路 TA 师兄也已给出，照搬即可。

⑥ 然后，怪物可向角色所在位置移动，由于角色和怪物的坐标都是 Vec2（向量），我们做减法就可以获得从怪物到角色的方向。这部分 TA 也已给出。

⑦ 根据 TA 提供的翻转的思路，在原来 wasd 的基础上做简单修改即可实现。

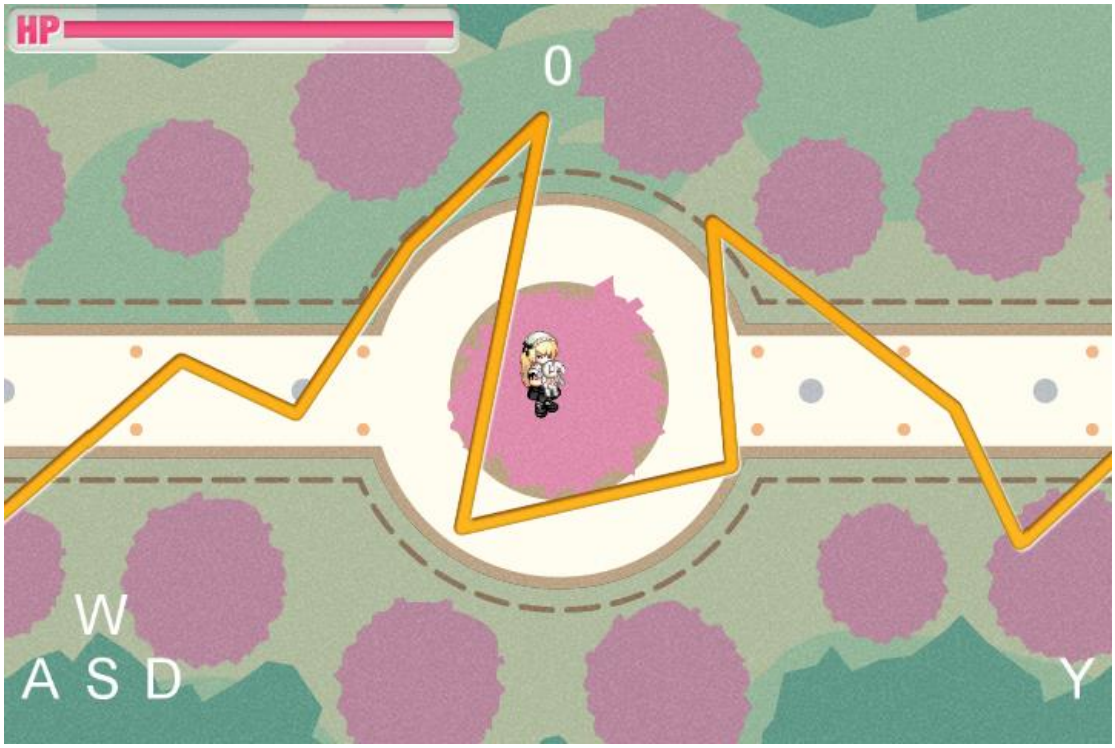
⑧ 实现本地数据的存储，这部分虽说是加分项，但其实非常简单，只需要简单几行代码即可实现，具体代码见第四部分。

三．实验结果截图

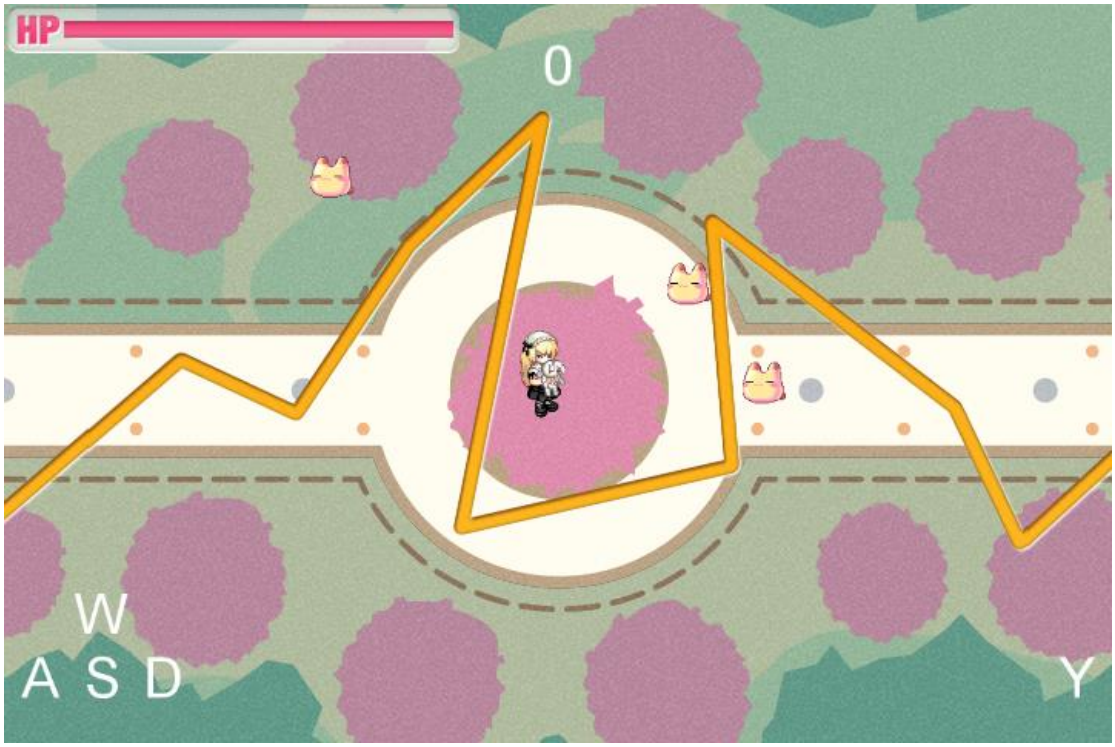
请在这里把实验所得的运行结果截图。

① 程序运行起始界面：可以看到还没怪物产生，因为我设置

schedule 区间为 3s，要在 3s 后才出现第一只怪物。

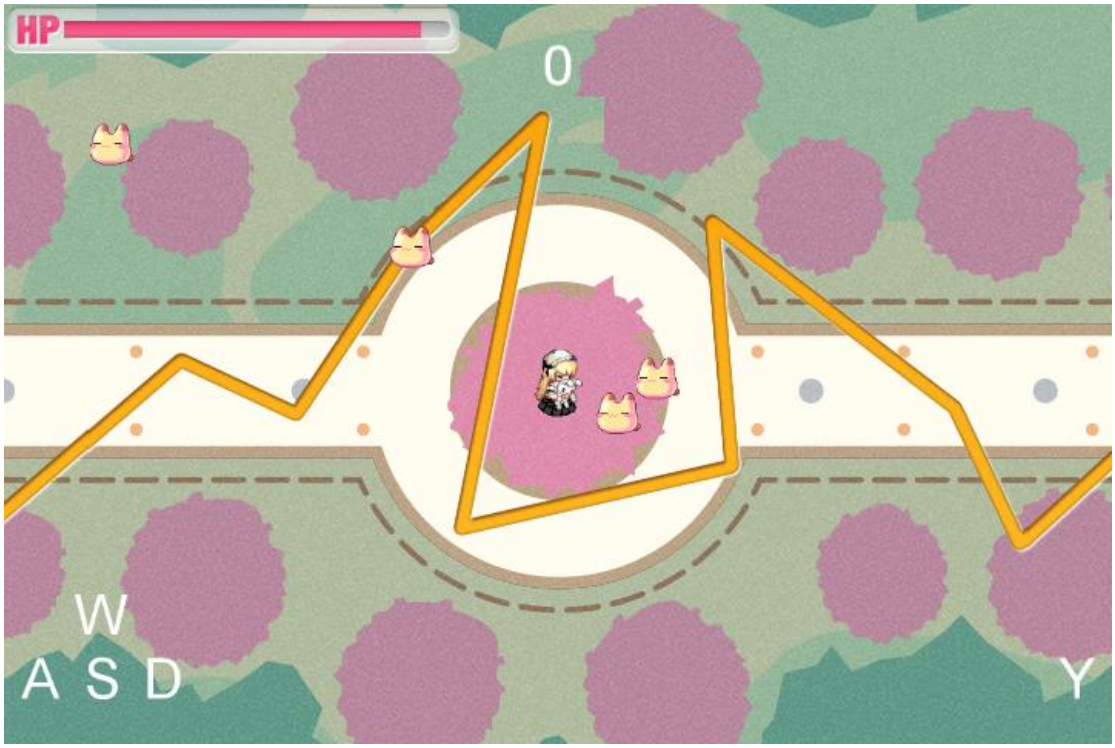


② 如图，随机产生怪物，还可看到怪物都移向角色，准备攻击角色

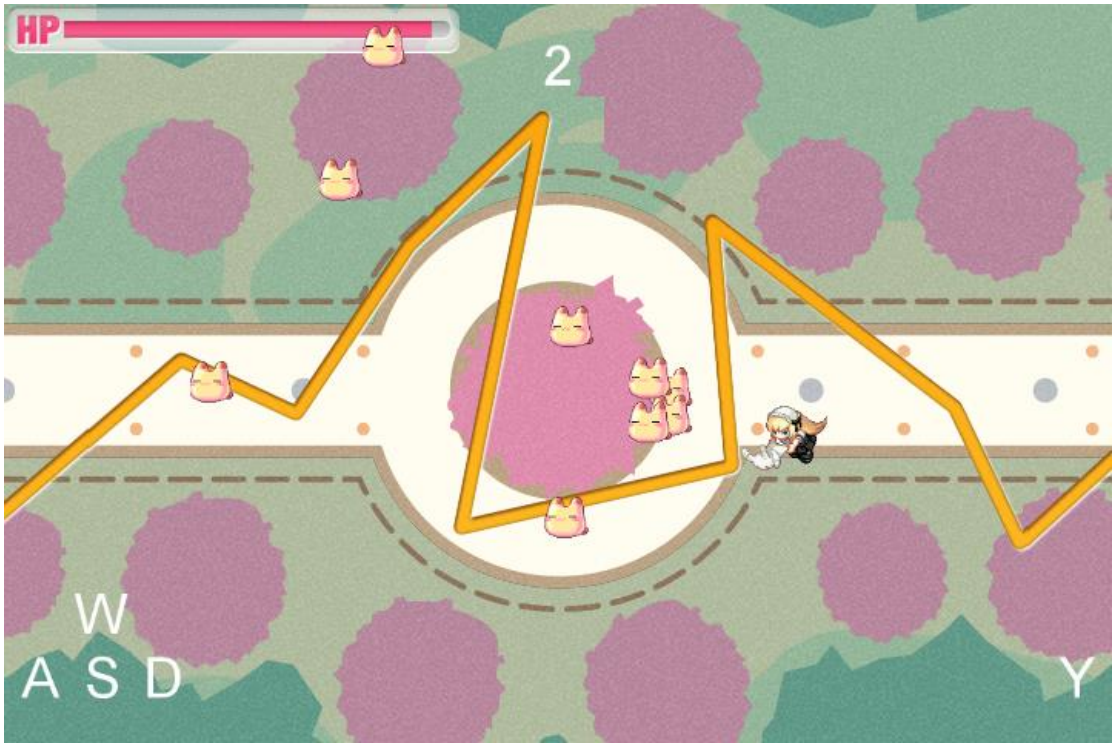


③ 角色受到怪物攻击后“死亡”：角色做出死亡动画，血条减少，

同时与之碰撞的怪物消失：

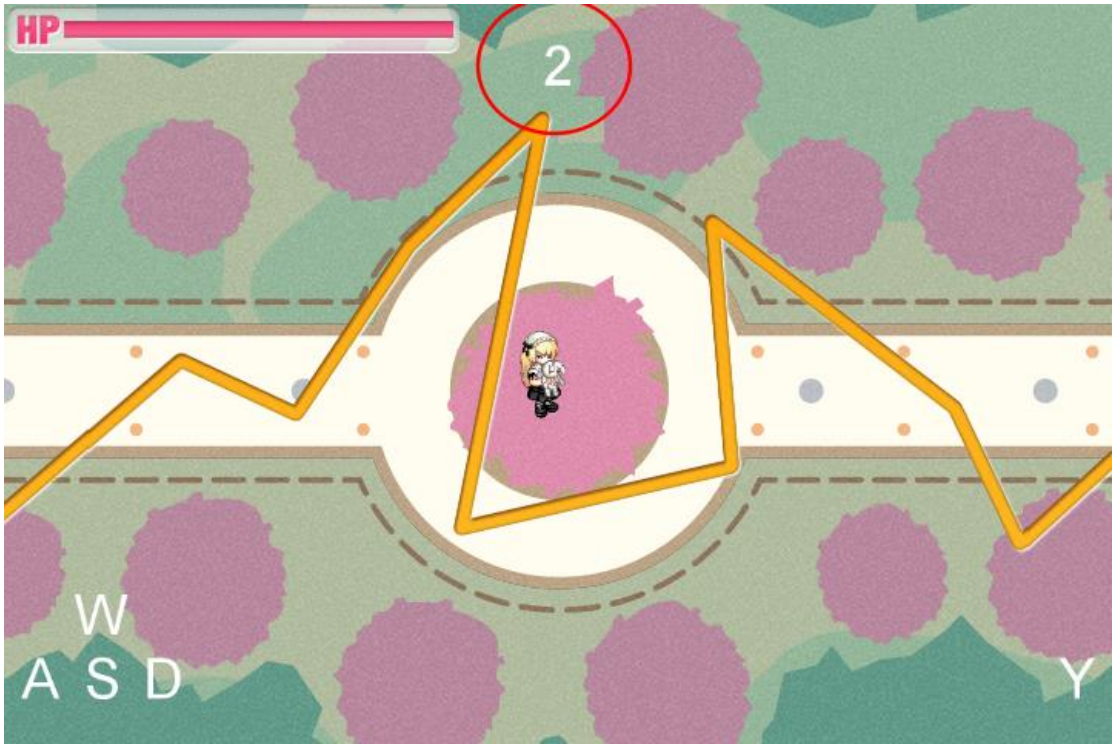


④ 角色攻击怪物：角色做出攻击动画，血条回升，上方数字显示已攻击怪物的数量，同时在攻击范围内的怪物消失：



⑤ 本地数据存储：在上图攻击数量为 2 的状态下关闭程序，重新

运行程序，可看到上方数字显示为 2



四．实验过程遇到的问题

请在这里写下你在实验过程中遇到的问题以及解决方案。

由于 TA 师兄给了相当多的代码，所以这次实验总的来说不是很难，下面我主要根据一点实验步骤给出相关代码。

① 利用 TMX 文件加载瓦片地图，设置成背景并自适应屏幕

```
// 利用TMX文件设置背景并自适应屏幕（根据放大因子来设置）
TMXTiledMap* tmx = TMXTiledMap::create("map.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);
```

② 设置程序启动时延迟产生怪物，这里延迟并不是设置 schedule 函数的最后一个参数，因为 demo 是每隔三秒左右产生一个怪物，所以我们修改第二参数“区间”即可。随机产生怪物的代

码也随之付出，写于 update 函数中。

```
schedule(schedule_selector(HelloWorld::update), 3.0f, kRepeatForever, 0);

void HelloWorld::update(float dt)
{
    // 获取工厂，生成怪物，放置在场景中
    auto fac = Factory::getInstance();
    auto monster = fac->createMonster();
    float monsterx = random(origin.x, visibleSize.width);
    float monstery = random(origin.y, visibleSize.height);
    monster->setPosition(monsterx, monstery);
    addChild(monster, 1);

    fac->moveMonster(player->getPosition(), 1.0f);

}
```

③ 判断怪物和角色是否碰撞的函数如下：遍历数据结构 Vector（整个 monster），根据参数 rect 是否包含怪物的位置返回符合条件的怪物。

```
Sprite* Factory::collider(Rect rect) {
    for (int i = 0; i < monster.size(); i++) {
        Vec2 position = monster.at(i)->getPosition();
        if (rect.containsPoint(position)) {
            return monster.at(i);
        }
    }
    return NULL;
}
```

④ 角色受到攻击

```
void HelloWorld::hitByMonster()
{
    auto fac = Factory::getInstance();
    //Rect playerRect = player->getBour
    //Rect hitRect = Rect(playerRect.ge
    Sprite* collision = fac->collider(p
    if (collision != NULL) {
        xy_action(this, 'X');
        fac->removeMonster(collision);
    }
}
```

⑤ 角色攻击怪物

```
bool HelloWorld::attackMonster()
{
    Rect playerRect = player->getBoundingBox();
    // 攻击前后方水平方向40内的敌人
    Rect attackRect = Rect(playerRect.getMinX() - 40
    auto fac = Factory::getInstance();
    Sprite* collision = fac->collider(attackRect);
    if (collision != NULL) {
        score++;
        char* myScore = new char[10];
        sprintf(myScore, "%d", score);
        time->setString(myScore);
        fac->removeMonster(collision);
        database->setIntegerForKey("value", score);
        return true;
    }
    return false;
}
```

⑥ 怪物移向角色的代码：

```
void Factory::moveMonster(Vec2 playerPos,float time){
    for (int i = 0; i < monster.size(); i++) {
        Vec2 position = monster.at(i)->getPosition();
        Vec2 direction = playerPos - position;
        direction.normalize();
        monster.at(i)->runAction(MoveBy::create(time, direction * 30));
    }
}
```

⑦ 移除怪物的代码：

```
void Factory::removeMonster(Sprite* sp) {
    Animation* animation = Animation::createWithSpriteFrames(monsterDead, 0.1f);
    Animate* animate = Animate::create(animation);
    Sequence* seq = Sequence::create(animate, CallFunc::create(CC_CALLBACK_0(Sprite::removeFromParent, sp)), NULL);
    sp->runAction(seq);
    monster.eraseObject(sp);
}
```

⑧ 按 A 或 D 角色翻转方向的代码：

```
char lastOption = NULL;
switch (option)
{
    case 'A':
    {
        if (position_x <= origin.x + 45) break;
        // 改变角色的方向（向左）
        if (lastOption != 'A') {
            player->setFlipX(true);
        }
        lastOption = 'A';
        auto moveBy = MoveBy::create(0.3, Point(-30, 0));
        auto seq = Sequence::create(begin, runAnimate, end, NULL);
        auto spawn = Spawn::createWithTwoActions(seq, moveBy);
        player->runAction(spawn);
        break;
    }
```

```
case 'D':
{
    if (position_x >= visibleSize.width + origin.x - 45) break;
    // 改变角色的方向（向右）
    if (lastOption != 'D') {
        player->setFlipX(false);
    }
    lastOption = 'D';
    auto moveBy = MoveBy::create(0.3, Point(30, 0));
    auto seq = Sequence::create(begin, runAnimate, end, NULL);
    auto spawn = Spawn::createWithTwoActions(seq, moveBy);
    player->runAction(spawn);
    break;
}
```

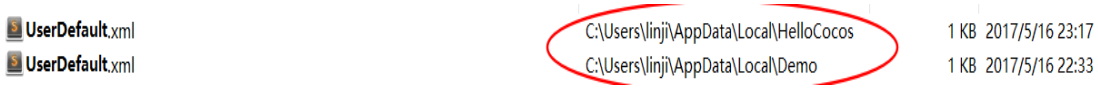
⑨本地数据存储的代码，攻击怪物更新数据库的值，以及 UserDefault.xml 文件在电脑中的位置：

```
#define database UserDefault::getInstance()
```

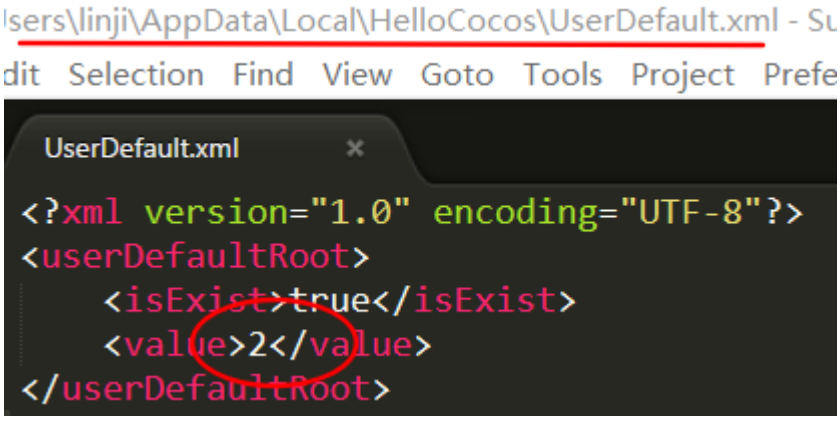
```
score = database->getIntegerForKey("value");
if (!database->getBoolForKey("isExist")) {
    database->setBoolForKey("isExist", true);
}
```

```
bool HelloWorld::attackMonster()
{
    Rect playerRect = player->getBoundingBox();
    // 攻击前后方水平方向40内的敌人
    Rect attackRect = Rect(playerRect.getMinX() - 40, play
    auto fac = Factory::getInstance();
    Sprite* collision = fac->collider(attackRect);
    if (collision != NULL) {
        score++;
        char* myScore = new char[10];
        sprintf(myScore, "%d", score);
        time->setString(myScore);
        fac->removeMonster(collision);
        database->setIntegerForKey("value", score);
        return true;
    }
    return false;
}
```

UserDefault.xml 文件在电脑中的位置：第二个是 TA 提供的 demo 的 UserDefault.xml 位置：



打开第一个文件，对应于我的数据库，可以看到对应第三部分截图中消灭 2 个怪物：



五．思考与总结

请在这里写下你本次试验的心得体会以及所思所想。

本次实验作业较为简单，原因有以下几点：

① TA 师兄提供的作业指导代码很多已给出，照搬即可。

② 不需要像前面几次作业那样调 Label 位置的要求。真的节省了很多时间。

③ 在 HW11 的基础上做修改即可，降低了作业难度，只需要实现几个函数方法即可。

④ 功能逻辑简单易懂，容易实现。

……

尽管如此，由于上周刚刚结束期中项目，精力憔悴，最终还是实现了一个自认为还 ok 的项目，但对现操作业莫名感到拒绝，还好作业不难，谢天谢地！

接下来还是要好好学习 cocos2d-x，很快就要做期末项目了！加油！

1. 实验报告提交格式为 pdf。
2. 实验内容不允许抄袭，我们要进行代码相似度对比。如发现抄袭，按 0 分处理。