现代操作系统应用开发实验报告

学号：153311117　　**班级　　：**教务 2 班

姓名：黄楠绚　　**实验名称：**homework12

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。

Cocos2d-x v3.3 中 UserDefault 保存的 XML 文件位置：http://www.tuicool.com/articles/Eb6NNz2

以及老师和 ta 给的 demo 和 ppt

二．实验步骤

请在这里简要写下你的实验过程。

在上次作业的基础上进行修改，去除 X 按钮，根据 ppt 的代码导入地图，并且把所有按钮和正上方计算打怪数量的 label 设为黑色：

```
grade->setColor(Color3B(0, 0, 0));
```

使用 userfaulet 进行本地数据储存打怪数量，如果是第一次调用 UserDefault::getInstance()，则初始化设置为 0：

```
if (!database->getIntegerForKey("grade")) {
    database->setIntegerForKey("grade", 0);
}
```

再用 UserDefalut 中的数据对 label 的文字进行赋值：

```
grade_counts = database->getIntegerForKey("grade");
char *grade_counts_string = new char[5];
sprintf(grade_counts_string, "%d", grade_counts);
```

设置两个调度器：

```
// 每隔5.0s产生和移动怪物
schedule(schedule_selector(HelloWorld::create_and_move_monster), 5.0f);
// 每隔0.1f检查一次怪物与player是否发生碰撞
schedule(schedule_selector(HelloWorld::hitByMonster), 0.1f);
```

```
// 检查是否与怪物碰撞
void HelloWorld::hitByMonster(float dt) {
    auto fac = Factory::getInstance();
    Sprite  *collision = fac->collider(player->getBoundingBox());
    if (collision != NULL) {
        fac->removeMonster(collision);
        dying();
    }
}
```

```
// 产生和移动怪物
void HelloWorld::create_and_move_monster(float dt) {
    auto m = Factory::getInstance()->createMonster();
    m->setPosition(random(origin.x, visibleSize.width), random(origin.y, visibleSize.height));
    addChild(m, 3);
    auto fact = Factory::getInstance();
    fact->moveMonster(player->getPosition(), 1.0f);
}
```

dying()函数是在上次作业的 X 按钮的回调函数上更改，在播放死亡动画前应终止当前所有动作：

```
void HelloWorld::dying() {
    // 先终止当前一切动作
    player->stopAllActions();
```

Y 按钮的攻击函数，使用 Monster.cpp 中的 collider 判断是否有怪物在 player 的攻击范围：

```
Rect playerRect = player->getBoundingBox();
// 攻击前后方的水平方向40内的敌人
Rect attackRect = Rect(playerRect.getMinX() - 40, playerRect.getMinY(),
    playerRect.getMaxX() - playerRect.getMinX() + 80, playerRect.getMaxY() - playerRect.getMinY());

auto fac = Factory::getInstance();
Sprite  *collision = fac->collider(attackRect);
```

攻击成功后血条增加且打怪数增加：

```
// 攻击成功
if (collision != NULL) {
    fac->removeMonster(collision);
    // 血条均匀增加
    float progressTo = pT->getPercentage() + 20;
    float progressFrom = pT->getPercentage();
    CCProgressFromTo *from_to = CCProgressFromTo::create(1.5, progressFrom, progressTo);
    pT->runAction(from_to);
    // 打怪数字增加
    ++grade_counts;
    // 保存到本地数据
    database->setIntegerForKey("grade", grade_counts);
    // 修改label
    char *grade_counts_string = new char[5];
    sprintf(grade_counts_string, "%d", grade_counts);
    grade->setString(grade_counts_string);
}
```

Monster.cpp 中：

removeMonster 移除怪物，先判断怪物是否在 vector 中，若存在则移除；

moveMonster 怪物移动，朝着 player 所在方向移动；

collider 判断怪物是否在所给的 rect 区域内，有则返回第一个符合条件的怪物，没有则返回 NULL：

```
Sprite* Factory::collider(Rect rect) {
    Vector<Sprite*>::iterator it = monster.begin();
    for (; it != monster.end(); it++) {
        if (rect.containsPoint((*it)->getPosition())) {
            return (*it);
        }
    }
    return NULL;
}
```

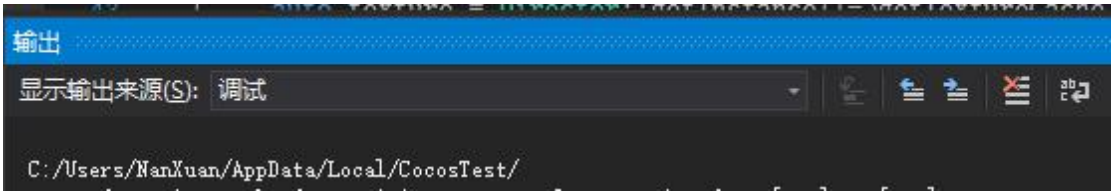
三．实验结果截图

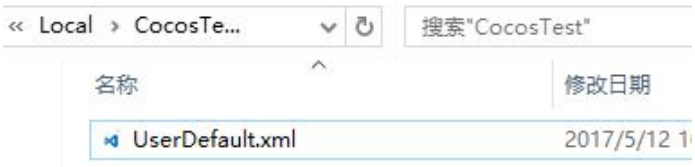
请在这里把实验所得的运行结果截图。

为了查看 UserDefault.xml，测试时在代码中写一句：

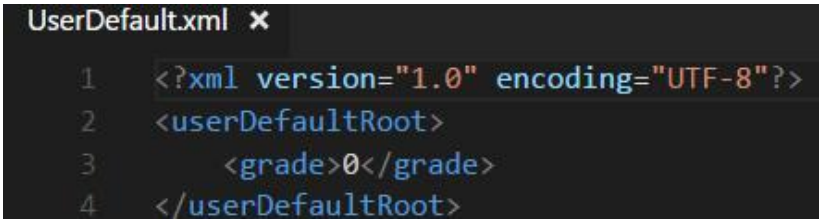
```
}
log("%s", FileUtils::getInstance()->getWritablePath().c_str());
```

根据输出的路径找到 xml：

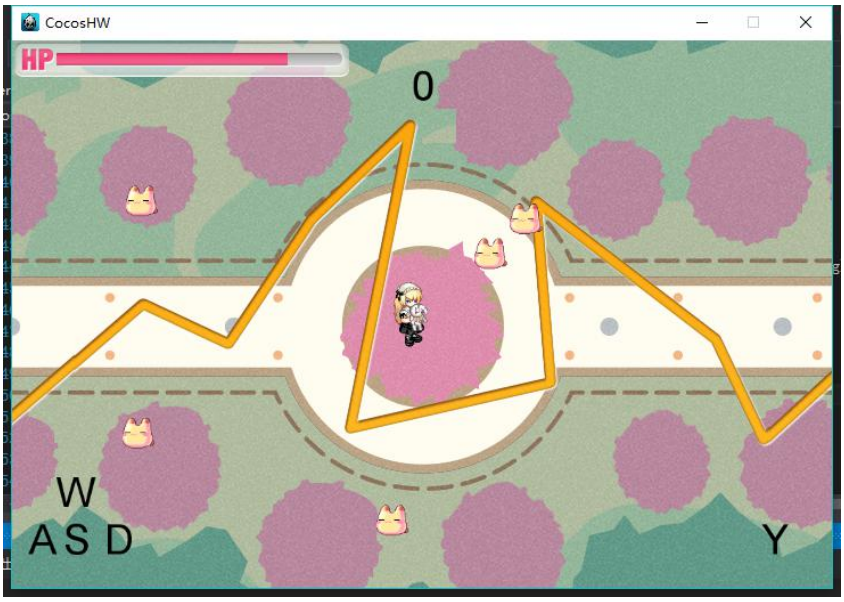




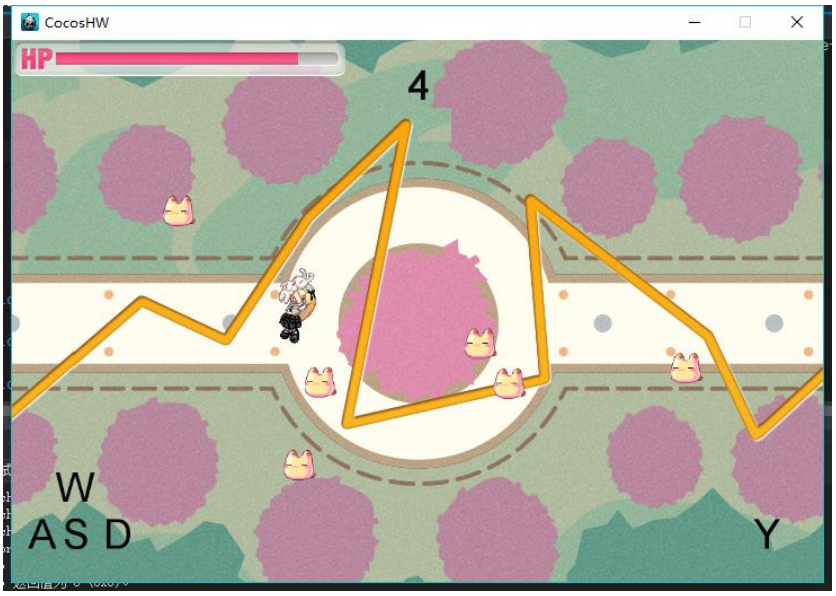
程序刚运行的状态下，查看了一下 xml 文件的内容，可以看到数据为 0：



程序刚开始运行，怪物随机产生：



攻击怪物，正上方的得分增加，血条增加：

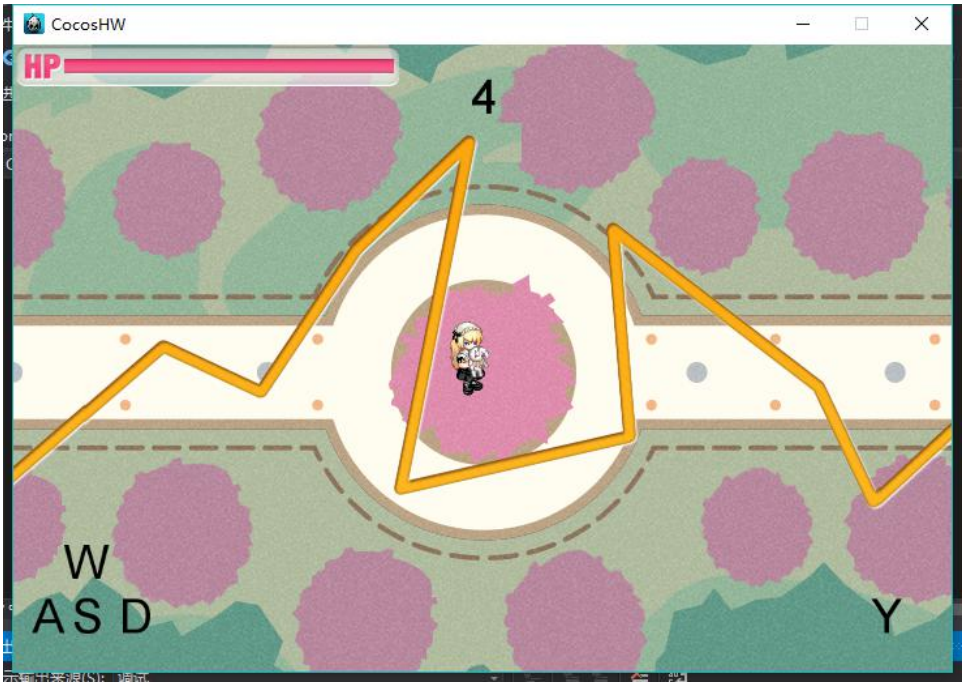


查看 xml，xml 的数据已经更新：

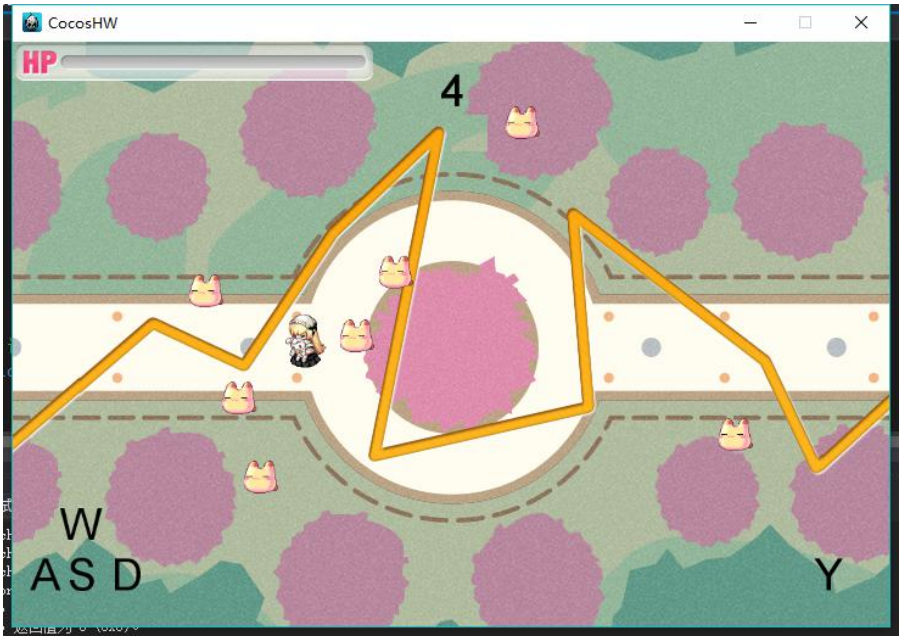
UserDefault.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<userDefaultRoot>
    <grade>4</grade>
</userDefaultRoot>
```

重新运行程序，可以看到得分可以恢复成上次关闭时候的数据：



与怪物碰撞时血条减少，死亡画面播放：



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。

问题：一开始在 player 连续执行 Y 攻击动作的时候，player 与怪物碰撞，怪物消失但不会播放死亡画面

解决方法：让怪物与 player 碰撞的时候，player 执行死亡画面之前停止一切动作。

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。

在这次实验中遇到了一些细节问题，在代码上不断做出调整和改进才能尽可能与 demo 相似的效果。

Ps: 这次实验的 ppt 给的代码稍微有点多，基本依靠 ppt 就可以完成代码，虽然作业可以很快完成，但希望可以的话 ppt 少一点代码提示，可能更可以锻炼动手能力。

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。