现代操作系统应用开发实验报告

学号：15331191　　**班级　　：**早上班

姓名：廖颖泓　　**实验名称：**Homework12

一．参考资料

1. 2017-homework12.pdf
2. 2017-cocos2dx 数据结构_本地存储和 tilemap.pdf
3. 博客 cocos2d-x.2.0 版本自适应屏幕分辨

   http://codingnow.cn/cocos2d-x/975.html

二．实验步骤

1. 根据 2017-homework12.pdf 提供的代码，在 Factory 类中的 createMonster 函数中产生单个 Monster 并在场景中用上节课学过的调度器随机产生 Monster。
2. 使用 Rect 类中的 containsPoint 做简单的碰撞测试，完成 hitByMonster 和 attackMonster 函数，然后用调度器定时检测碰撞，使得 Player 在被 Monster 攻击后能进行死亡动画和并血量扣减和攻击 Monster 后能进行攻击动画和血量恢复（这些动画在上次已经实现）。
3. 完成 Factory 类中的 moveMonster 函数 ,用 For each 使得每个 Monster 都会向 Player 的位置移动。
4. 根据 2017-homework12.pdf 提供的代码，用 player->setFlipX 函数使得 player

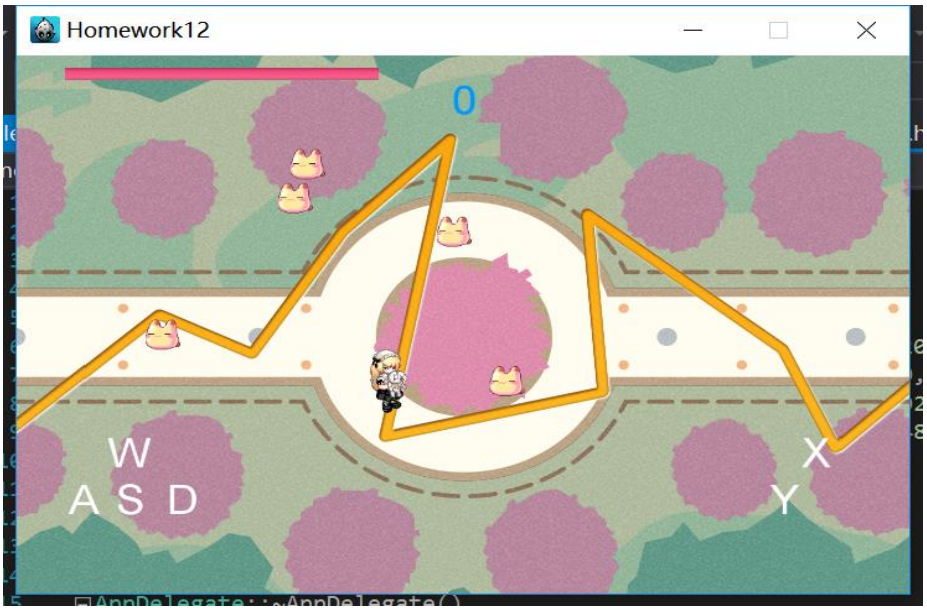
左右移动可以翻转。

5. 移除怪物。实现怪物死亡动画，并用 eraseObject 将场景中的怪物移除。

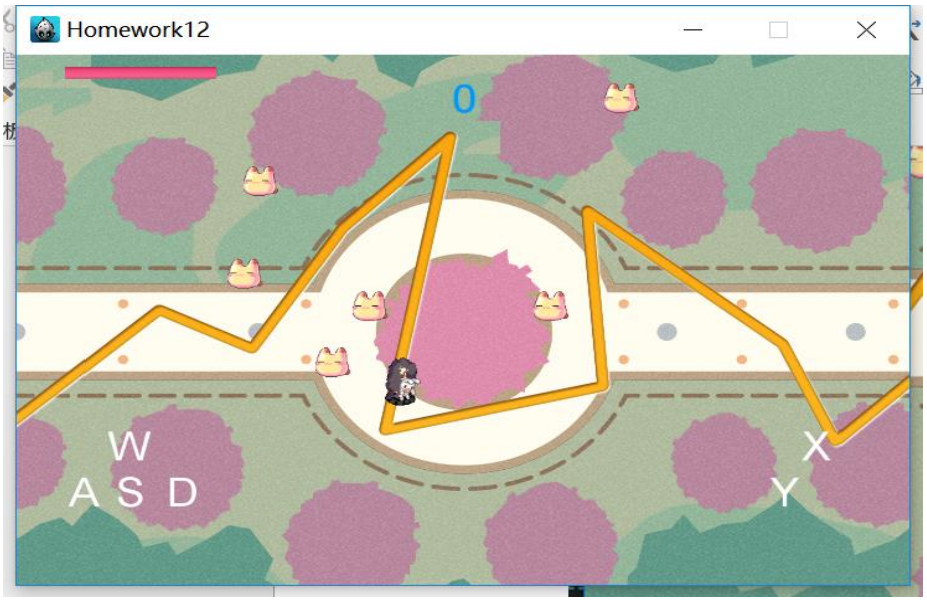
6. 加入地图，获取放大因子使得地图可以自适应窗口。

三．实验结果截图

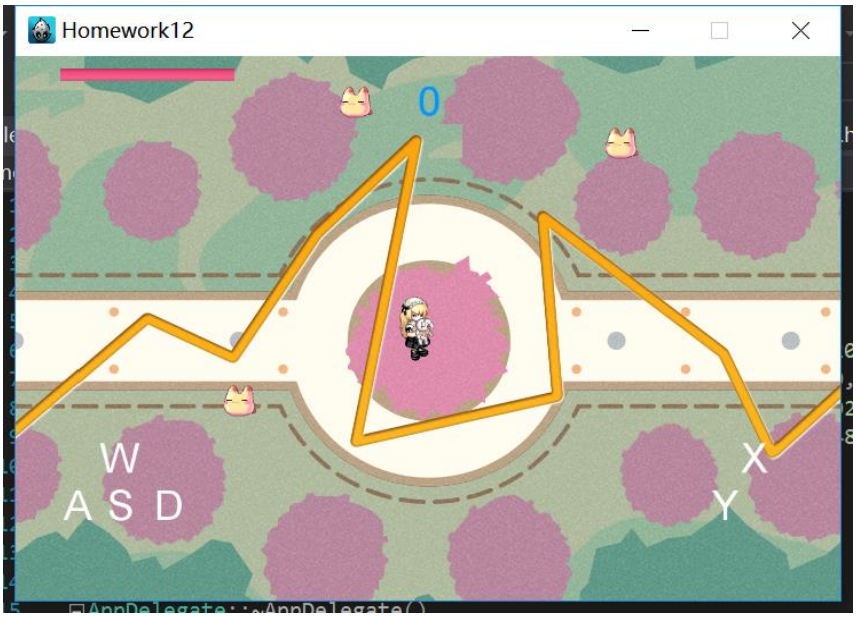
1. 随机产生怪物。

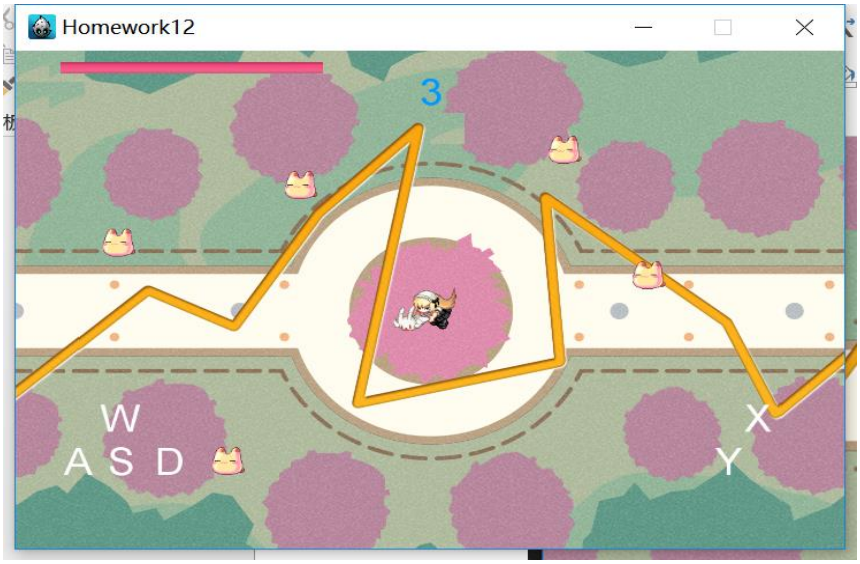


2. 怪物碰到角色后，角色掉血。



3. 角色可以攻击怪物。





4. 使用 TileMap 创建地图

```
//根据文件路径快速导入瓦片地图
TMXTiledMap* tmx = TMXTiledMap::create("map.tmx");
tmx->setPosition(visibleSize.width / 2, visibleSize.height / 2);
tmx->setAnchorPoint(Vec2(0.5, 0.5));
tmx->setScale(Director::getInstance()->getContentScaleFactor());
addChild(tmx, 0);
```

5. 使用本地数据存储，记录打到的怪物数量，并将倒计时改为显示打倒数量。


```
//检测xml文件是否存在（非必须）
if (!database->getBoolForKey("isExist"))
{
    database->setBoolForKey("isExist", true);
    database->setStringForKey("killNum", "0");
}

database->setStringForKey("killNum", "0");
```

四．实验过程遇到的问题

attackMonster 函数一开始我没搞清楚碰撞检测的原理，直接使用了 containsPoint 函数，但是行不通后来才发现 attackRect 可以放入 collider 中，这样才确保 player 可以在一定范围内攻击怪物。

```
auto fac = Factory::getInstance();
Rect playerRect = player->getBoundingBox();
Rect attackRect = Rect(playerRect.getMinX() - 40, playerRect.getMinY(), playerRect.getMaxX()
    - playerRect.getMinX() + 80, playerRect.getMaxY() - playerRect.getMinY());
Sprite* collision = fac->collider(attackRect);
if (collision != NULL)
{
    fac->removeMonster(collision);
    killNum++;
    if (pT->getPercentage() != 100)
    {
        pT->runAction(CCProgressTo::create(1.8f, pT->getPercentage() + 20));
    }
    char ct[10];
    _itoa(killNum, ct, 10);
    killNumLabel->setString(ct);
    database->setStringForKey("killNum", std::string(ct));
```

五．思考与总结

本次实验运用了课上讲的数据结构、本地存储和 Tilemap，对上个星期的小游戏进行了进一步的改进，人物事件更加丰富，趣味性十足。本地存储可以在一定程度上避免使用复杂的数据库，对游戏速度的优化有很大的意义。