# 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：** HW11

## 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：老师提供的关于 cocos2d-x_ui_调度器_帧动画的课件，TA 师兄的作业指导课件

网站：cocos2d-x 官方 API http://api.cocos.com/

进度条 http://blog.csdn.net/wwj_748/article/details/37819787

http://www.byjth.com/biji/32.html 等等

倒计时 http://blog.csdn.net/hsljz/article/details/40862663

此外，还有很多大大小小的问题，看了很多博客，这里不再一一列举。

## 二． 实验步骤

请在这里简要写下你的实验过程。

关键代码在第四部分中体现。

① 首先，学习课件中的知识点，主要是 cocos2d-x 中的 UI、调度器的使用和帧动画的实现。UI 包括 label、button、menu、slider、view 等；调度器分为三种：默认，自定义，单次；最后是序列帧动画的处理。

② 根据作业要求，可以先完成游戏的 UI 部分。主要包括游戏中出现的所有 label 及其字体大小，计时器（数字 180），而人物精灵和人物血条 TA 师兄已经写好，我自己加了一个游戏背景，所以这部分也就基本完成了。

③ 接下来是添加帧动画，根据 TA 师兄攻击动画的代码，我们可以依样画葫芦加载其他动画资源，例如 dead 动画，运动前进动画。添加帧动画的原理是利用一个 vector 把切割图片得到的每一帧动画存进去，使用 Animation 类描述一个动画，精灵显示动画的动作是一个 Animate 对象，动画动作 Animate 是精灵显示动画的动作，由动画对象创建，由精灵执行。

④ 接着实现按键 wasd 的功能，要实现上下左右移动，我们需要有一个回调函数，所以我设置了 wasd_move(Ref*, char)，其中 char 用来接收一个参数表示 wasd 中的一个，然后根据该参数实现上下左右移动的功能。

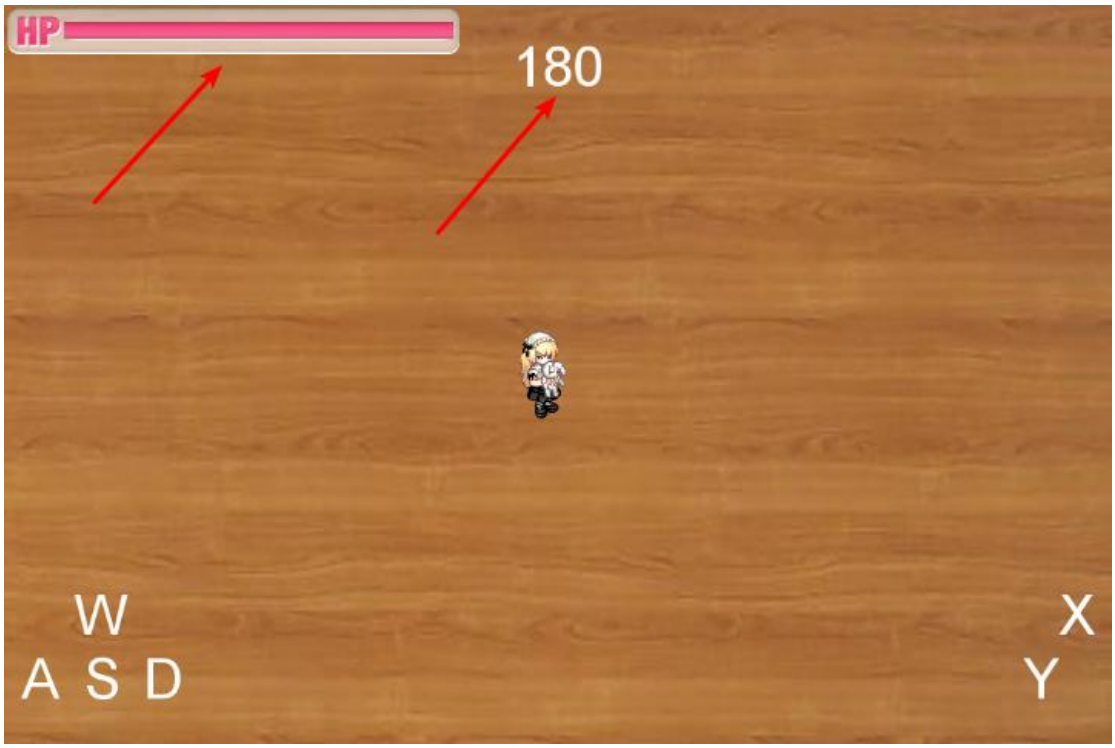
⑤ 实现按键 xy 的功能，和 wasd 的实现相同。我设置了 xy_action(Ref*, char) ,其中 char 用来接收一个参数表示 x 或者 y，然后根据该参数实现 dead 或者 attack 功能。

⑥ 最后，使用调度器实现倒计时。这里需要注意 time 是一个 label，无法接收一个 int 的 dtime，所以需要对 int 的 dtime 进行字符串转换。

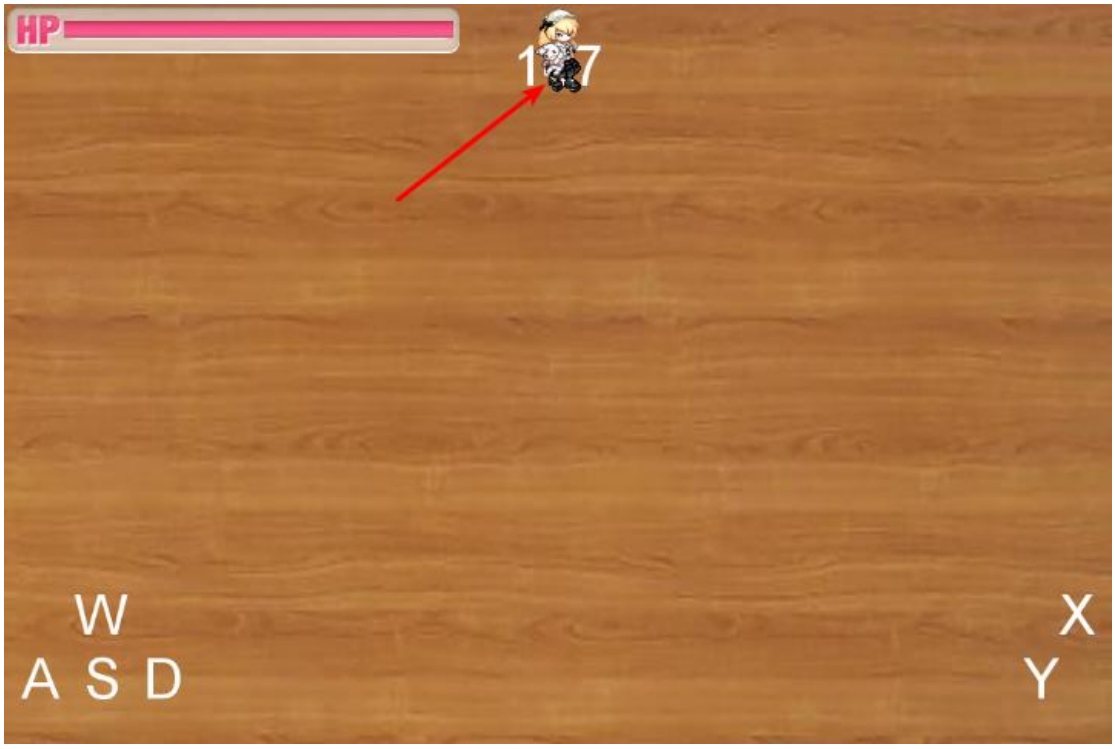
## 三． 实验结果截图

请在这里把实验所得的运行结果截图。

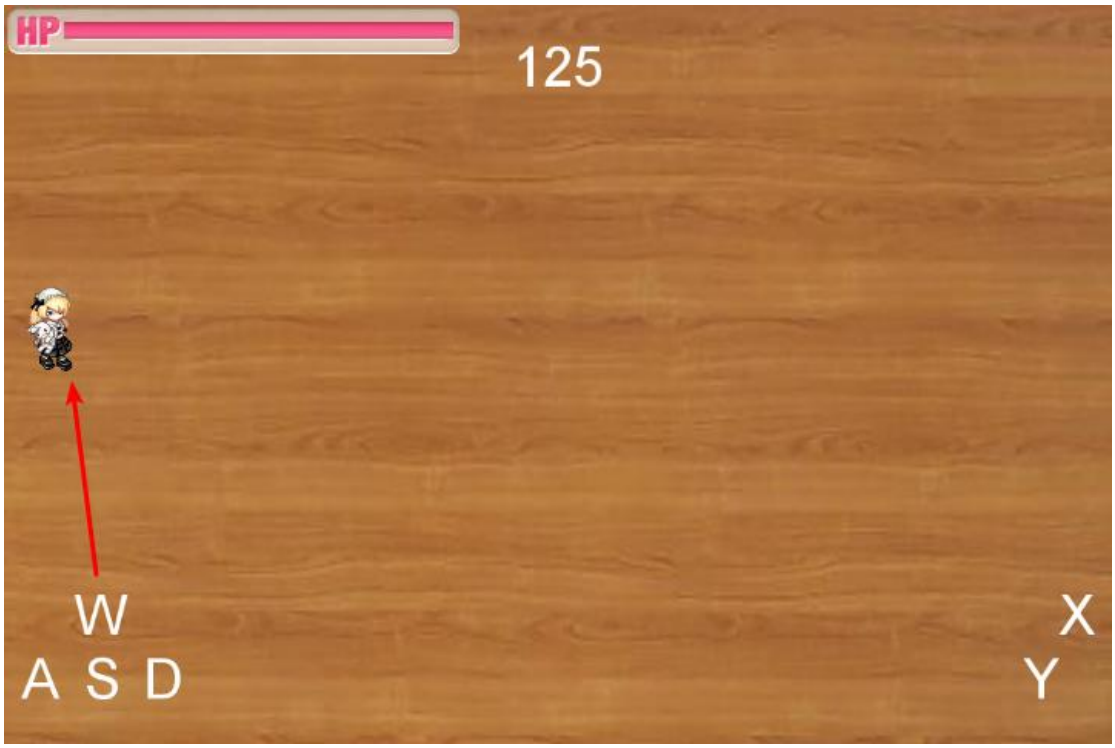
① 游戏初始界面：可以看到时间显示为 180，人物满血，WASD 和 XY 位置如图所示



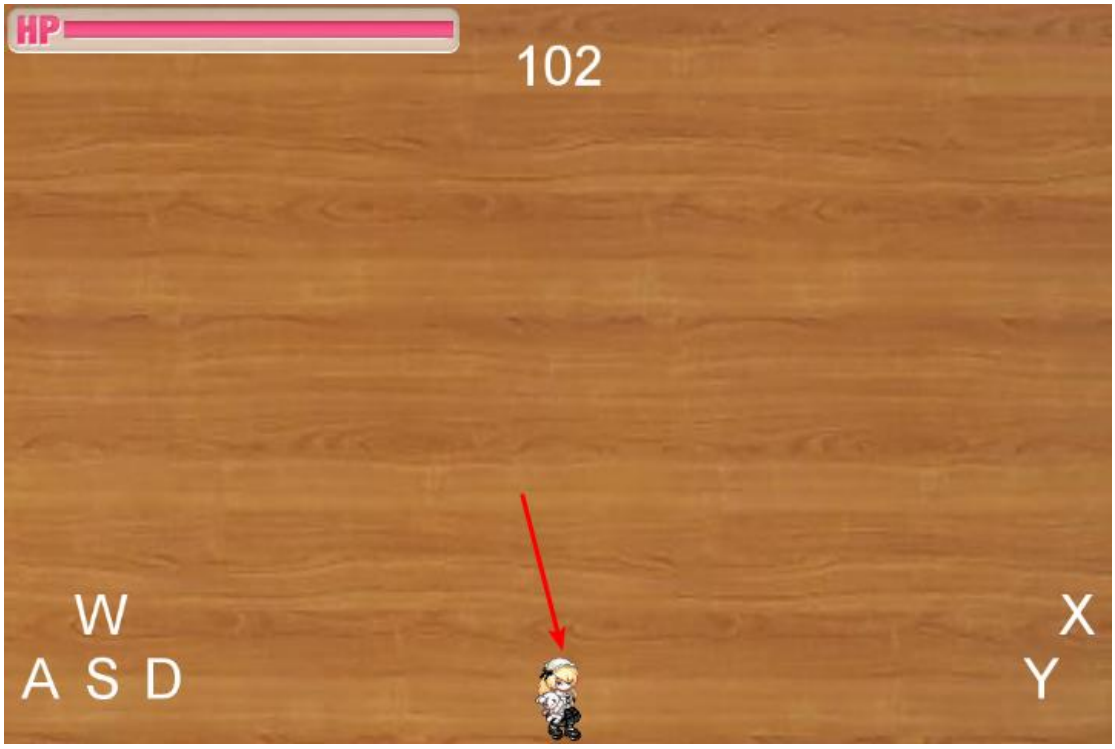

② 点击 W：可看到人物精灵向上移动，不断点击 W，当精灵到达游戏界面顶端时，则不再向上移动



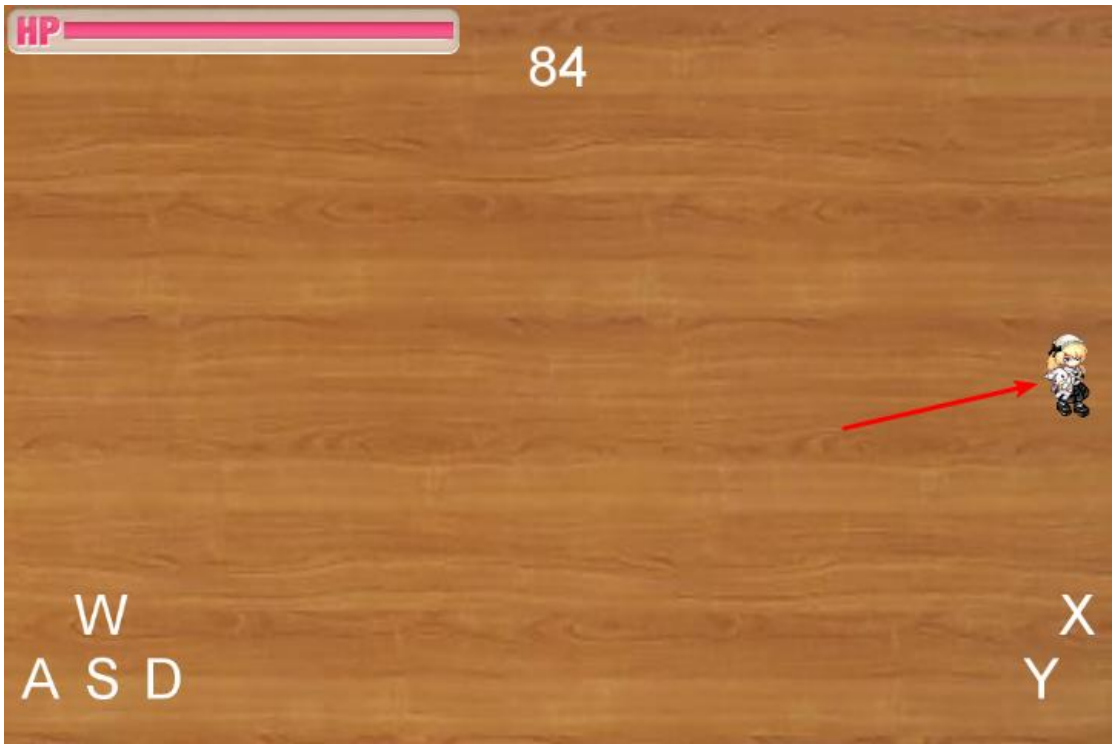
③ 点击 A：可看到人物精灵向左移动，不断点击 A，当精灵到达游戏界面左端时，则不再向左移动




④ 点击 S：可看到人物精灵向下移动，不断点击 S，当精灵到达游戏界面底部时，则不再向下移动

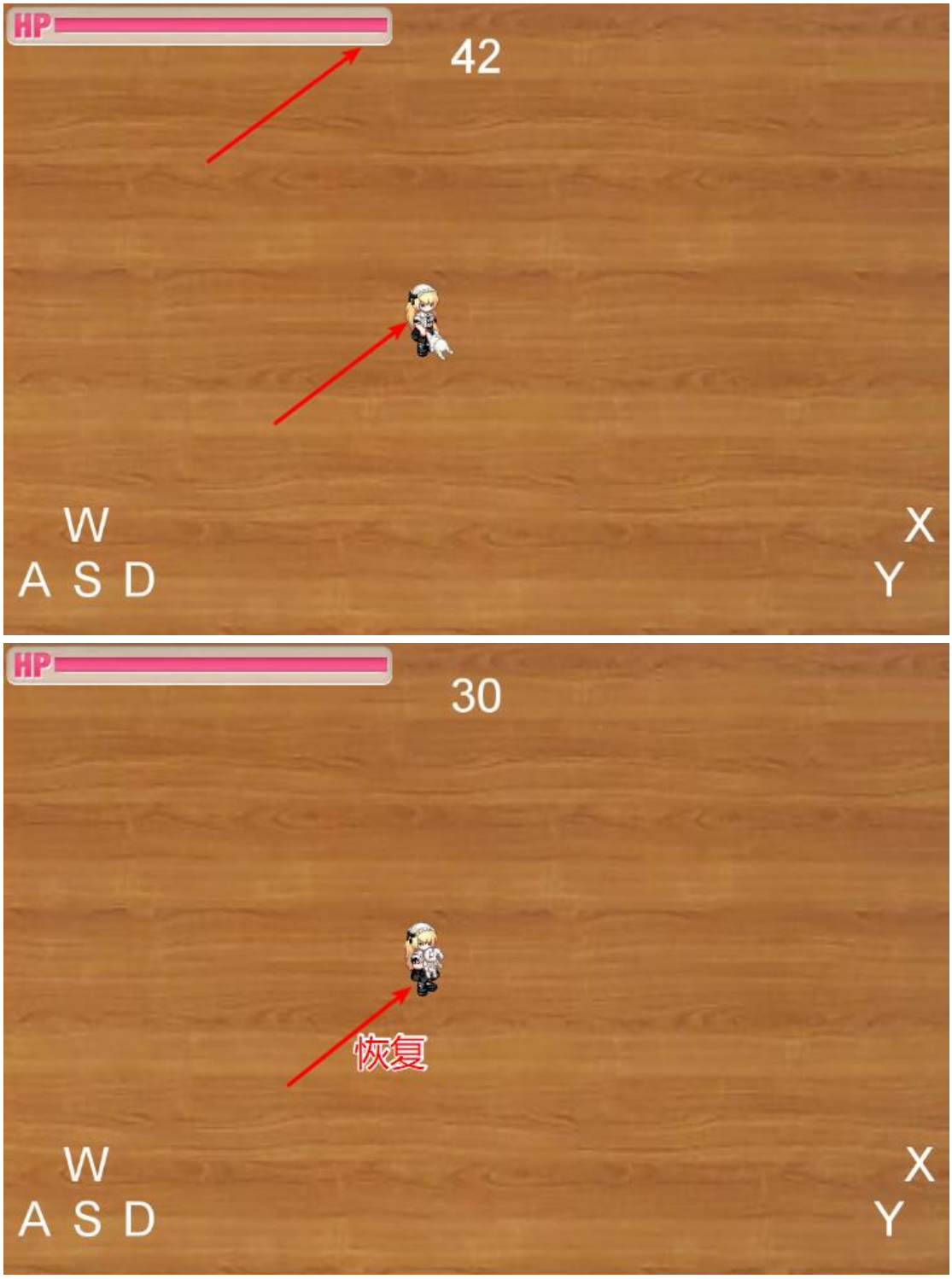


⑤ 点击 D：可看到人物精灵向右移动，不断点击 D，当精灵到达游戏界面右端时，则不再向右移动

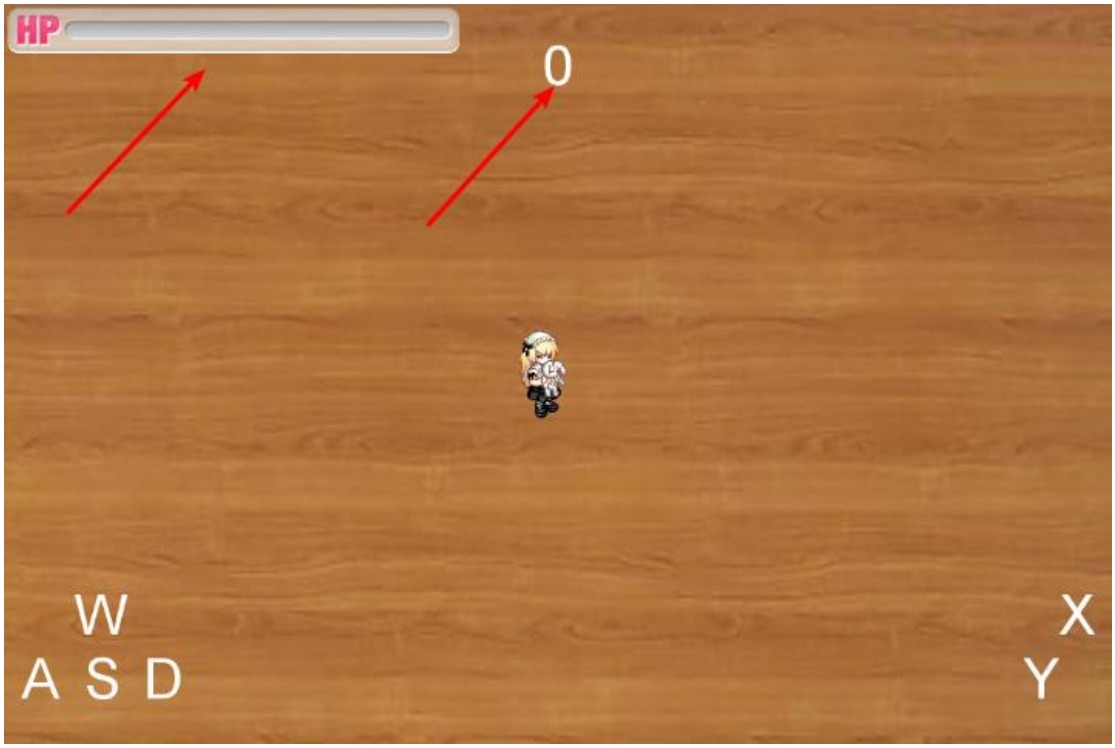



⑥ 点击 X：可看到人物精灵死亡，同时人物血条减掉 1/5，在该过程中点击其他按钮（包括 Y、WASD）均不会做出反应，死亡动画完毕后，恢复初始人物精灵动画


⑦ 点击 Y：可看到人物精灵攻击，同时人物血条加 1/5，在该过程中点击其他按钮（包括 X、WASD）均不会做出反应，攻击动画完毕后，恢复初始人物精灵动画




⑧ 人物血条的显示（0），时间显示（0）




## 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。

① 首先是一个简单的问题，就是调节各个 Label 的位置问题。在这方面一直只能盲试，而自己又是个“强迫症”患者，所以花的时间比较多。由于这是时间问题，谈不上技术，简单贴一下代码作为例子（以 A 为例子）就好。用的是 MenuItemLabel 类。

```
// add the "a" label with MenuItemFont Class
auto a = Label::createWithTTF(ttfConfig, "A");
auto a_item = MenuItemLabel::create(a, CC_CALLBACK_1(HelloWorld::wasd_move, this, 'A'));
a_item->setPosition(Vec2(origin.x + a_item->getContentSize().width, origin.y + a_item->getContentSize().height));
// add the a menu
auto a_menu = Menu::create(a_item, NULL);
a_menu->setPosition(Vec2::ZERO);
this->addChild(a_menu, 1);
```

② 时间显示的问题。因为时间 dtime 是一个 int 类型，而 Label 类的 createWithTTF 函数的第二个形参是 string 类型，所以需要把 int 转换成 string 类型，所以我采用 sprintf 来实现，

而调度器实现倒计时可根据 PPT 的指导来编写，具体代码如下：

```
// 时间显示
dtime = 180;
char* strTime = new char[4];
sprintf(strTime, "%d", dtime);
time = Label::createWithTTF(ttfConfig, strTime);
time->setColor(Color3B(255, 255, 255));
time->setPosition(Vec2(origin.x + visibleSize.width / 2, origin.y + visibleSize.height - time->getContentSize().height));
addChild(time);
// 定时器
schedule(schedule_selector(HelloWorld::update), 1.0f, kRepeatForever, 0);
```

```
void HelloWorld::update(float dt)
{
    if (dtime == 0) {
        unschedule(schedule_selector(HelloWorld::update));
    }
    else {
        dtime = dtime - dt;
        char* strTime = new char[4];
        sprintf(strTime, "%d", dtime);
        time->setString(strTime);
    }
}
```

③ 运动前进动画的问题。TA 的 demo 并不是完整地截取前进动画的 8 帧，而只是前三帧（目测），所以我也只是截了前三帧。处理好这个后，还有一个要求——不让精灵移动到可视窗口外，需要调整精灵的当前位置和可视窗口的关系。代码如下：

```
// 运动动画(帧数：8帧，高：101，宽：68)
auto texture3 = Director::getInstance()->getTextureCache()->addImage("$lucia_forward.png");
run.reserve(3);
for (int i = 0; i < 3; i++) {  前三帧
    auto frame = SpriteFrame::createWithTexture(texture3, CC_RECT_PIXELS_TO_POINTS(Rect(68 * i, 0, 68, 101)));
    run.pushBack(frame);
}

// 创建一个Animation，参数：SpriteFrame*的Vector容器，每一帧之间的间隔
auto runAnimation = Animation::createWithSpriteFrames(run, 0.1f);
//runAnimation->setRestoreOriginalFrame(false);
// 使用animation创建一个animate，animate继承了ActionInterval，可以当做动作来使用
auto runAnimate = Animate::create(runAnimation);
```

```
switch (option)
{
    case 'A':
    {
        if (position_x <= origin.x + 45) break;
        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 'S':
    {
        if (position_y <= origin.y + 48) break;
        auto moveBy = MoveBy::create(0.3, Point(0, -30));
        auto seq = Sequence::create(begin, runAnimate, end, NULL);
        auto spawn = Spawn::createWithTwoActions(seq, moveBy);
        player->runAction(spawn);
        break;
    }
```

④ 按键 XY 的实现方法与 WASD 类似，主要注意把初始帧动画加入到 Vector 中即可，让人物精灵执行完 dead 或者 attack 动画后恢复到初始形态。

```
// 攻击动画
attack.reserve(17);
for (int i = 0; i < 17; i++) {
    auto frame = SpriteFrame::createWithTexture(texture, CC_RECT_PIXELS_TO_POINTS(Rect(113*i,0,113,113)));
    attack.pushBack(frame);
}
attack.pushBack(frame0);
```

将初始帧加入到vector，恢复到初始形态

```
// 可以仿照攻击动画
// 死亡动画(帧数：22帧，高：90，宽：79)
auto texture2 = Director::getInstance()->getTextureCache()->addImage("$lucia_dead.png");
dead.reserve(22);
for (int i = 0; i < 22; i++) {
    auto frame = SpriteFrame::createWithTexture(texture2, CC_RECT_PIXELS_TO_POINTS(Rect(79 * i, 0, 79, 90)));
    dead.pushBack(frame);
}
dead.pushBack(frame0);
```

```
switch (option)
{
    case 'X':
    {
        // 创建一个Animation，参数：SpriteFrame*的Vector容器，每一帧之间的间隔
        auto deadAnimation = Animation::createWithSpriteFrames(dead, 0.1f);
        //deadAnimation->setRestoreOriginalFrame(true);
        // 使用animation创建一个animate，animate继承了ActionInterval，可以当做动作来使用
        auto deadAnimate = Animate::create(deadAnimation);
        auto seq = Sequence::create(begin, deadAnimate, end, NULL);
        player->runAction(seq);
        // 以1/5的速度减血条
        if (pT->getPercentage() >= 20)
            pT->runAction(CCProgressFromTo::create(1.5f, pT->getPercentage(), pT->getPercentage() - 20));
        else
            pT->runAction(CCProgressFromTo::create(1.5f, pT->getPercentage(), 0));
        break;
    }
    case 'Y':
    {
```

⑤ 血条的加减问题。一开始一直想着可不可以像 uwp 那样搞个数据绑定来实现（真的学傻了），后来顿然醒悟，这怎么可能？查了 ProgressTimer 相关的资料，发现 cocos2d-x 可以用 CCProgressFromTo 或者 CCProgressTo 来实现（两者的不同就是有没有初始值而已）我选了 CCProgressFromTo 来实现，而且还发现了 ProgressTimer 类中有 setPercentage 和 getPercentage 两个函数，水到渠成，解决了该问题。代码如下：

```
case 'X':
{
    // 创建一个Animation，参数：SpriteFrame*的Vector容器，每一帧之间的间隔
    auto deadAnimation = Animation::createWithSpriteFrames(dead, 0.1f);
    //deadAnimation->setRestoreOriginalFrame(true);
    // 使用animation创建一个animate，animate继承了ActionInterval，可以当做动作来使用
    auto deadAnimate = Animate::create(deadAnimation);
    auto seq = Sequence::create(begin, deadAnimate, end, NULL);
    player->runAction(seq);
    // 以1/5的速度减血条
    if (pT->getPercentage() >= 20)
        pT->runAction(CCProgressFromTo::create(1.5f, pT->getPercentage(), pT->getPercentage() - 20));
    else
        pT->runAction(CCProgressFromTo::create(1.5f, pT->getPercentage(), 0));
    break;
}
case 'Y':
{
    // 创建一个Animation，参数：SpriteFrame*的Vector容器，每一帧之间的间隔
    auto attackAnimation = Animation::createWithSpriteFrames(attack, 0.1f);
    //attackAnimation->setRestoreOriginalFrame(true);
    // 使用animation创建一个animate，animate继承了ActionInterval，可以当做动作来使用
    auto attackAnimate = Animate::create(attackAnimation);
    auto seq = Sequence::create(begin, attackAnimate, end, NULL);
    player->runAction(seq);
    // 以1/5的速度加血条
    if (pT->getPercentage() <= 80)
        pT->runAction(CCProgressFromTo::create(1.5f, pT->getPercentage(), pT->getPercentage() + 20));
    else
        pT->runAction(CCProgressFromTo::create(1.5f, pT->getPercentage(), 100));
    break;
}
```

⑥ 实现动画不能同时播放，这个过程就比较辛苦，耗费时间了。一开始知道要弄个 bool 值来记录是否处于播放动画中，但是只是简单地添加在判断语句中，但无论怎么弄都无法实现。最后，通过请教同学，原来是需要设置动作开始时和结束时的 bool 值，用到 CallFuncN 类，将 bool 值加入到动作序列中，即动作序列中以 begin，animate，end 为顺序执行动画动作。这里我添加了 wasd 和 xy 均不能同时进行，关键代码如下：

```
void HelloWorld::wasd_move(Ref * pSender, char option)
{
    if (isAction) return;
    auto begin = CallFuncN::create([&](Ref* sender) {
        isAction = true;
    });
    auto end = CallFuncN::create([&](Ref* sender) {
        isAction = false;
    });

    // 创建一个Animation，参数：SpriteFrame*的Vector容器，每一帧之间的间隔
    auto runAnimation = Animation::createWithSpriteFrames(run, 0.1f);
    //runAnimation->setRestoreOriginalFrame(false);
    // 使用animation创建一个animate，animate继承了ActionInterval，可以当做动作来使用
    auto runAnimate = Animate::create(runAnimation);

    auto position_x = player->getPositionX();
    auto position_y = player->getPositionY();

    switch (option)
    {
        case 'A':
        {
            if (position_x <= origin.x + 45) break;
            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;
        }
```

```
void HelloWorld::xy_action(Ref * pSender, char option)
{
    if (isAction) return;
    auto begin = CallFuncN::create([&](Ref* sender) {
        isAction = true;
    });
    auto end = CallFuncN::create([&](Ref* sender) {
        isAction = false;
    });
    switch (option)
    {
        case 'X':
        {
            // 创建一个Animation，参数：SpriteFrame*的Vector容器，每一帧之间的间隔
            auto deadAnimation = Animation::createWithSpriteFrames(dead, 0.1f);
            //deadAnimation->setRestoreOriginalFrame(true);
            // 使用animation创建一个animate，animate继承了ActionInterval，可以当做动作来使用
            auto deadAnimate = Animate::create(deadAnimation);
            auto seq = Sequence::create(begin, deadAnimate, end, NULL);
            player->runAction(seq);
            // 以1/5的速度减血条
            if (pT->getPercentage() >= 20)
                pT->runAction(CCProgressFromTo::create(1.5f, pT->getPercentage(), pT->getPercentage() - 20));
            else
                pT->runAction(CCProgressFromTo::create(1.5f, pT->getPercentage(), 0));
            break;
        }
```

## 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。

这次实现一个横版游戏，涉及到 cocos2d-x 中的 UI、调度器和序列帧动画知识，是对 cocos2d-x 的进一步学习，通过这次作业，学习了 Label、MenuItemLabel、ProgressTimer...等等类，还学习了调度器和回调函数的知识。

我也看到了平时我们玩的简单游戏其实并不简单，它包含了游戏开发者的设计，细节的完善，毫不夸张地说一个游戏蕴含了开发者的“思想”。我想接下来的游戏作业一定会更加有趣，也许难度会大点，但是真正锻炼了我们，能够真正学到知识，加油！