现代操作系统应用开发实验报告

学号： 1433188 **班级 ：** 教务三班

姓名： 郑泽佳 **实验名称：** 实验 10

一．参考资料

http://www.tairan.com/archives/5831/

http://cocos2d-x.org/docs/api-ref/cplusplus/v3x/

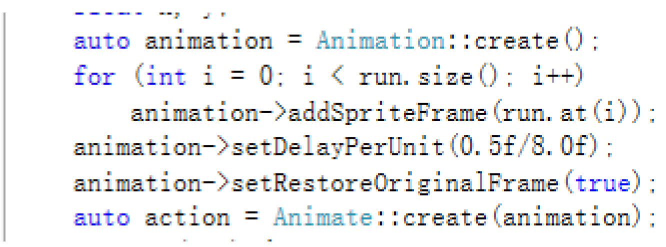
http://shahdza.blog.51cto.com/2410787/1546707

课件

二．实验步骤

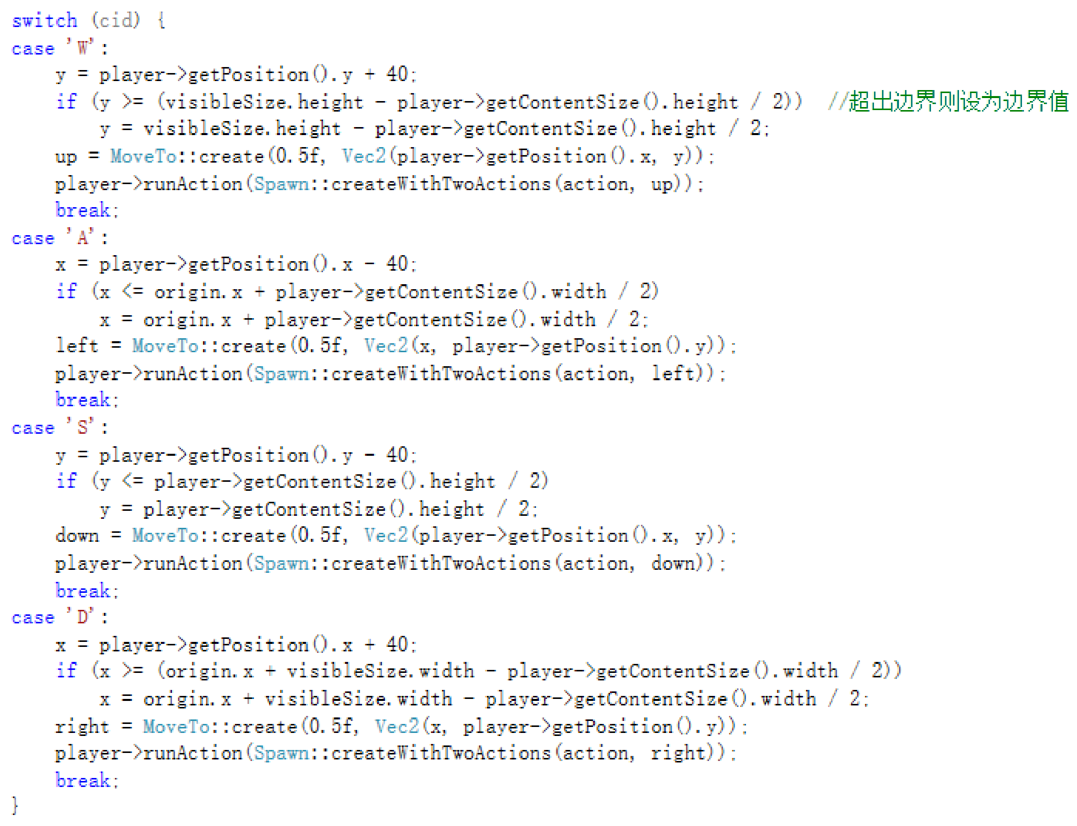
1、实现左边 wasd4 个虚拟按键能控制角色移动

a) 实现人物走动的动画，代码如下



```
auto animation = Animation::create();
for (int i = 0; i < run.size(); i++)
    animation->addSpriteFrame(run.at(i));
animation->setDelayPerUnit(0.5f/8.0f);
animation->setRestoreOriginalFrame(true);
auto action = Animate::create(animation);
```

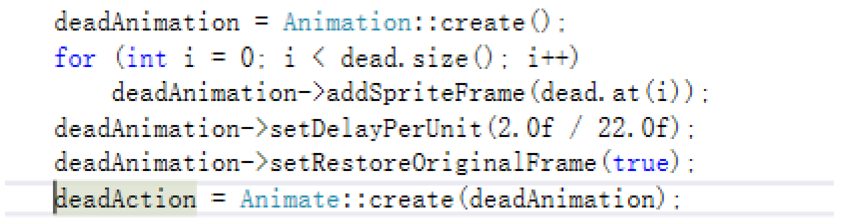
b) 通过完善 wasd 的触发事件来完成角色的上下左右移动，并添加判断条件使得角色不会跑出可视窗口



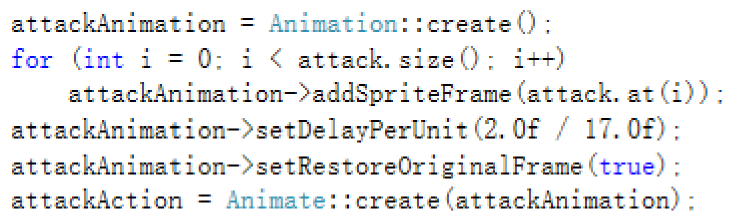
```
switch (cid) {
case 'W':
    y = player->getPosition().y + 40;
    if (y >= (visibleSize.height - player->getContentSize().height / 2))  //超出边界则设为边界值
        y = visibleSize.height - player->getContentSize().height / 2;
    up = MoveTo::create(0.5f, Vec2(player->getPosition().x, y));
    player->runAction(Spawn::createWithTwoActions(action, up));
    break;
case 'A':
    x = player->getPosition().x - 40;
    if (x <= origin.x + player->getContentSize().width / 2)
        x = origin.x + player->getContentSize().width / 2;
    left = MoveTo::create(0.5f, Vec2(x, player->getPosition().y));
    player->runAction(Spawn::createWithTwoActions(action, left));
    break;
case 'S':
    y = player->getPosition().y - 40;
    if (y <= player->getContentSize().height / 2)
        y = player->getContentSize().height / 2;
    down = MoveTo::create(0.5f, Vec2(player->getPosition().x, y));
    player->runAction(Spawn::createWithTwoActions(action, down));
    break;
case 'D':
    x = player->getPosition().x + 40;
    if (x >= (origin.x + visibleSize.width - player->getContentSize().width / 2))
        x = origin.x + visibleSize.width - player->getContentSize().width / 2;
    right = MoveTo::create(0.5f, Vec2(x, player->getPosition().y));
    player->runAction(Spawn::createWithTwoActions(action, right));
    break;
}
```

2、实现.右边 2 个虚拟按键 x，y 能控制角色播放不同的帧动画

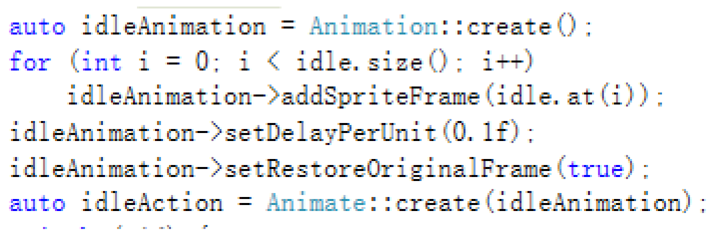
a) 添加角色死亡动画、攻击动画和闲置动画



```
deadAnimation = Animation::create();
for (int i = 0; i < dead.size(); i++)
    deadAnimation->addSpriteFrame(dead.at(i));
deadAnimation->setDelayPerUnit(2.0f / 22.0f);
deadAnimation->setRestoreOriginalFrame(true);
deadAction = Animate::create(deadAnimation);
```



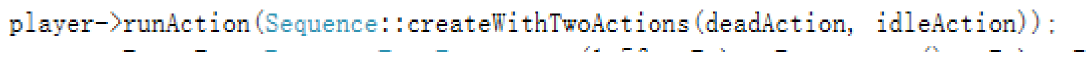
```
attackAnimation = Animation::create();
for (int i = 0; i < attack.size(); i++)
    attackAnimation->addSpriteFrame(attack.at(i));
attackAnimation->setDelayPerUnit(2.0f / 17.0f);
attackAnimation->setRestoreOriginalFrame(true);
attackAction = Animate::create(attackAnimation);
```



```
auto idleAnimation = Animation::create();
for (int i = 0; i < idle.size(); i++)
    idleAnimation->addSpriteFrame(idle.at(i));
idleAnimation->setDelayPerUnit(0.1f);
idleAnimation->setRestoreOriginalFrame(true);
auto idleAction = Animate::create(idleAnimation);
```

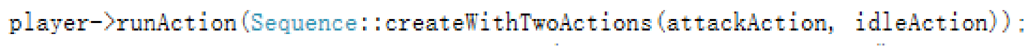
b) 实现 x、y 按键控制角色播放动画

Case ‘X‘:



```
player->runAction(Sequence::createWithTwoActions(deadAction, idleAction));
```

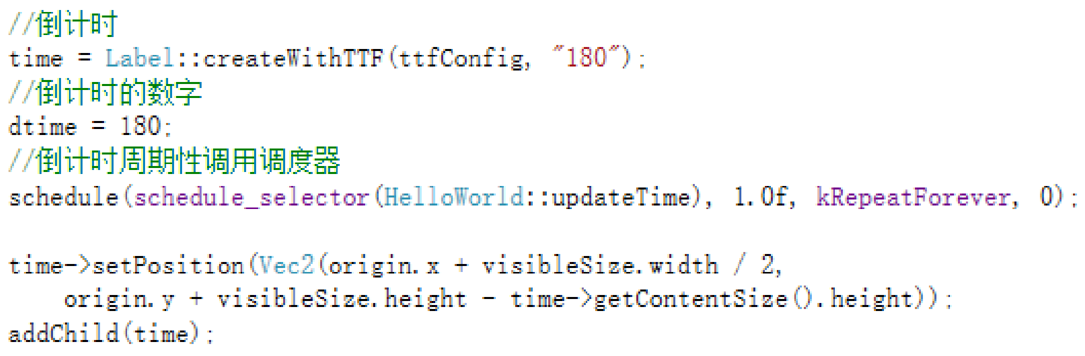
Case ‘Y‘:



```
player->runAction(Sequence::createWithTwoActions(attackAction, idleAction));
```

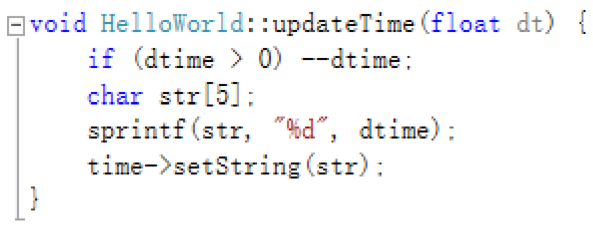
3、添加倒计时

a)添加倒计时Ｌａｂｅｌ，并添加调度器



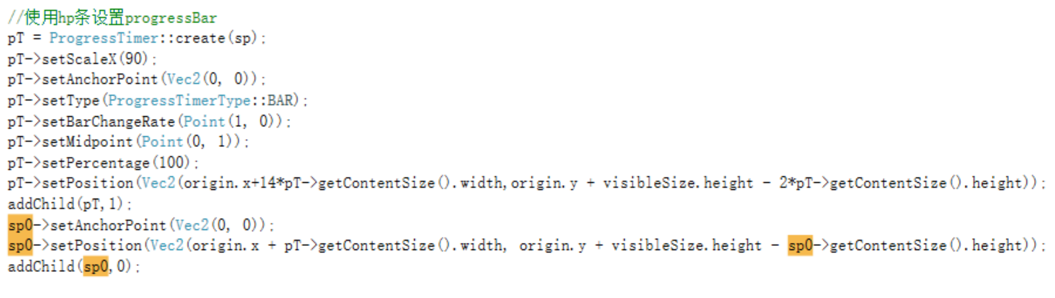
```
//倒计时
time = Label::createWithTTF(ttfConfig, "180");
//倒计时的数字
dtime = 180;
//倒计时周期性调用调度器
schedule(schedule_selector(HelloWorld::updateTime), 1.0f, kRepeatForever, 0);

time->setPosition(Vec2(origin.x + visibleSize.width / 2,
    origin.y + visibleSize.height - time->getContentSize().height));
addChild(time);
```

b)实现调度器，时间计数



```
void HelloWorld::updateTime(float dt) {
    if (dtime > 0) --dtime;
    char str[5];
    sprintf(str, "%d", dtime);
    time->setString(str);
}
```

4、添加人物血条

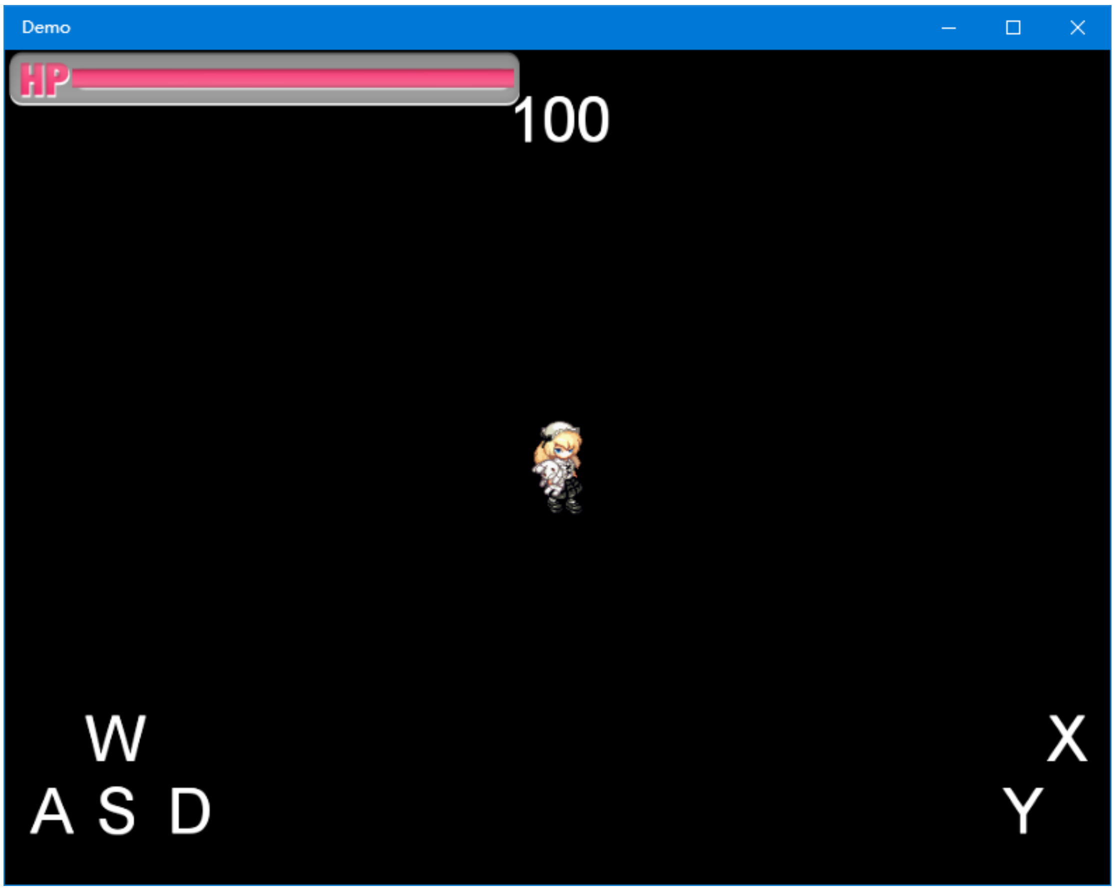


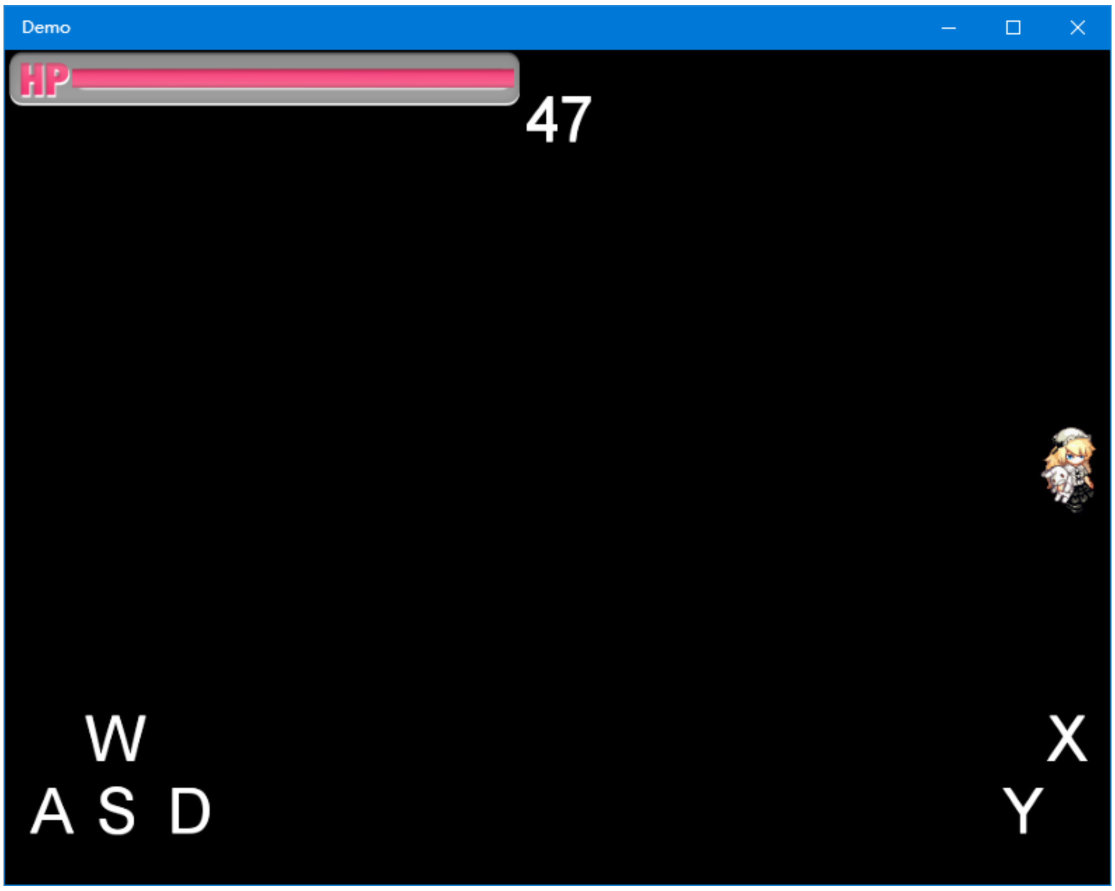
```
//使用hp条设置progressBar
pT = ProgressTimer::create(sp);
pT->setScaleX(90);
pT->setAnchorPoint(Vec2(0, 0));
pT->setType(ProgressTimerType::BAR);
pT->setBarChangeRate(Point(1, 0));
pT->setMidpoint(Point(0, 1));
pT->setPercentage(100);
pT->setPosition(Vec2(origin.x+14*pT->getContentSize().width,origin.y + visibleSize.height - 2*pT->getContentSize().height));
addChild(pT,1);
sp0->setAnchorPoint(Vec2(0, 0));
sp0->setPosition(Vec2(origin.x + pT->getContentSize().width, origin.y + visibleSize.height - sp0->getContentSize().height));
addChild(sp0,0);
```

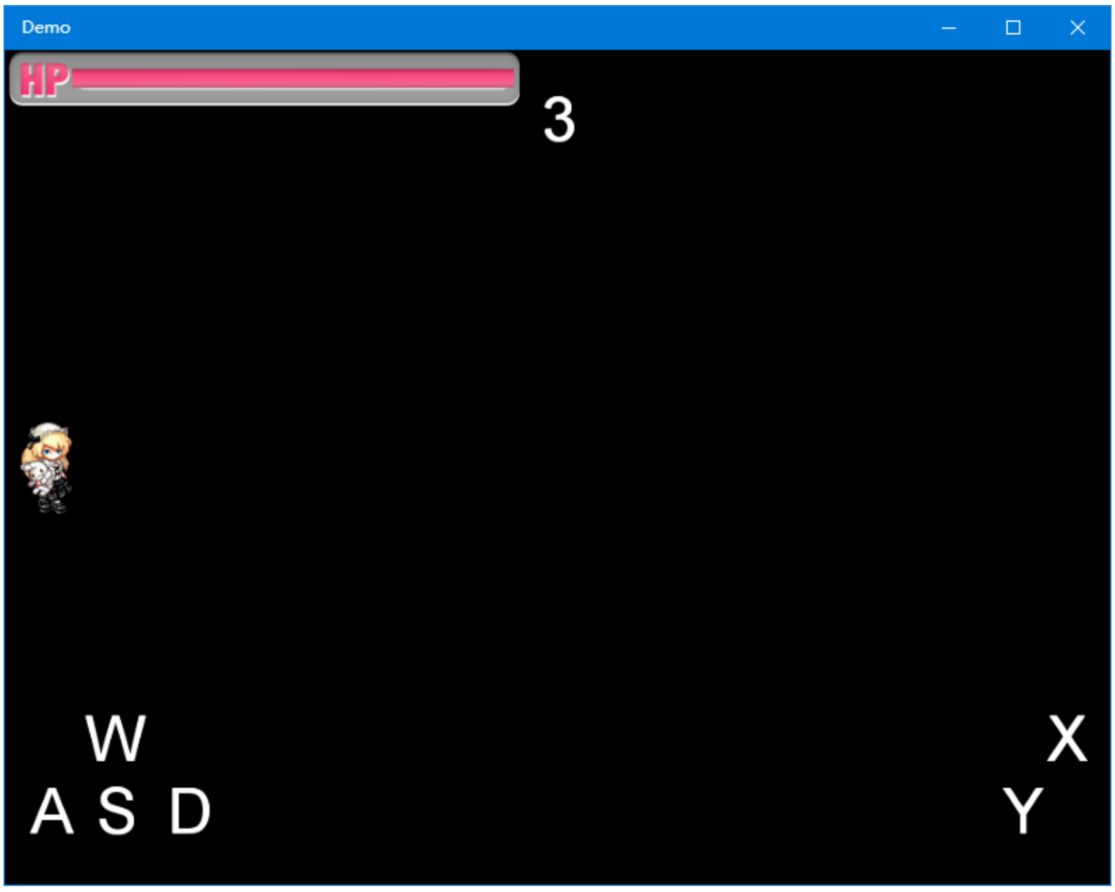
5、点击虚拟按键 x 播放帧动画并让血条减少,点击 y 播放帧动画并让血条增加

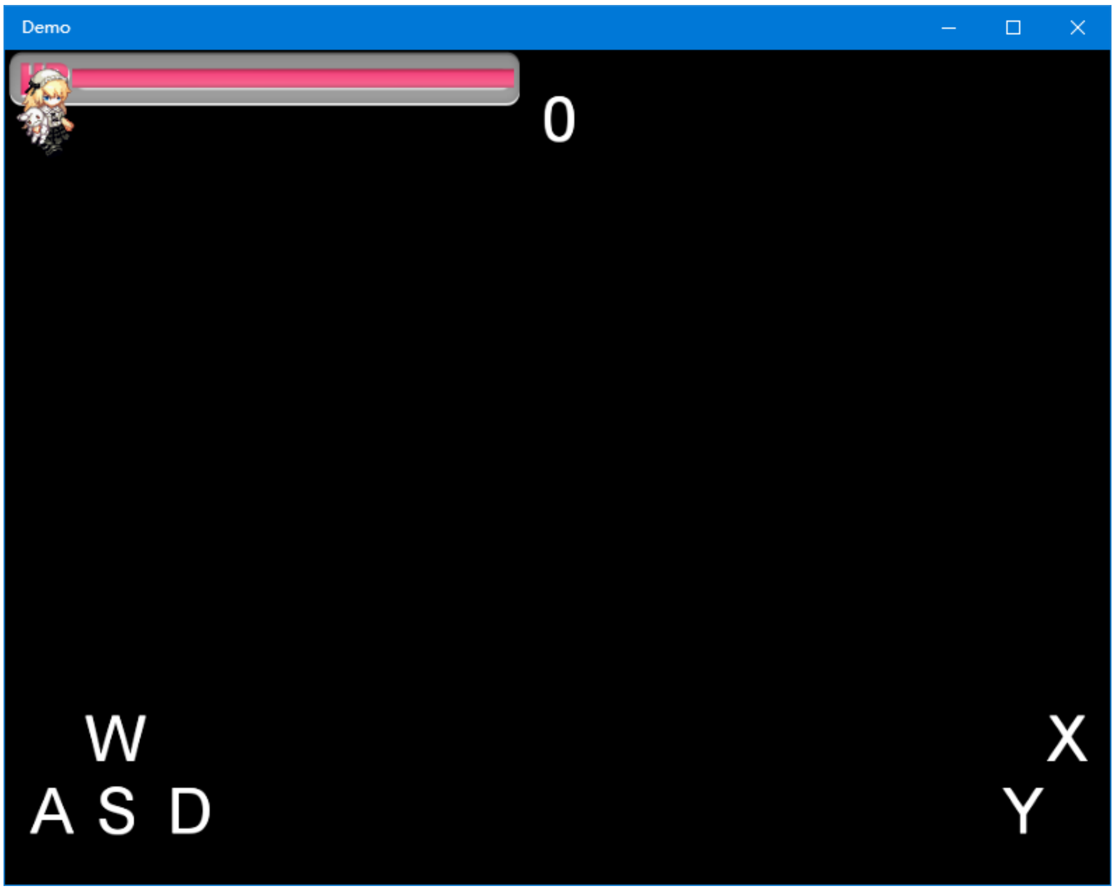
```
switch (cid) {
case 'X':
    deadAnimation = Animation::create();
    for (int i = 0; i < dead.size(); i++)
        deadAnimation->addSpriteFrame(dead.at(i));
    deadAnimation->setDelayPerUnit(2.0f / 22.0f);
    deadAnimation->setRestoreOriginalFrame(true);
    deadAction = Animate::create(deadAnimation);

    player->runAction(Sequence::createWithTwoActions(deadAction, idleAction));
    progressFromtTo = ProgressFromTo::create(1.5f, pT->getPercentage(), pT->getPercentage() - 20);
    pT->runAction(progressFromtTo);
    break;
case 'Y':
    attackAnimation = Animation::create();
    for (int i = 0; i < attack.size(); i++)
        attackAnimation->addSpriteFrame(attack.at(i));
    attackAnimation->setDelayPerUnit(2.0f / 17.0f);
    attackAnimation->setRestoreOriginalFrame(true);
    attackAction = Animate::create(attackAnimation);

    player->runAction(Sequence::createWithTwoActions(attackAction, idleAction));
    progressFromtTo = ProgressFromTo::create(1.5f, pT->getPercentage(), pT->getPercentage() + 20);
    pT->runAction(progressFromtTo);
    break;
}
```

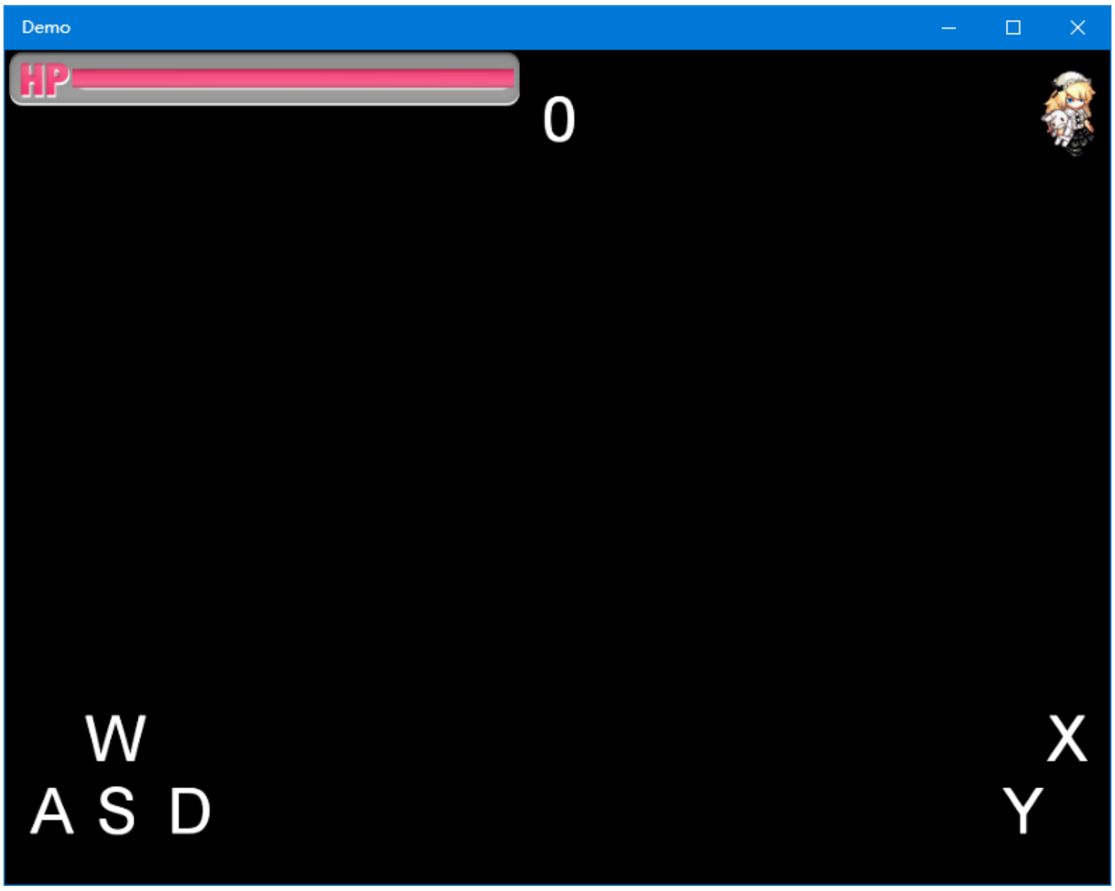
三．实验结果截图

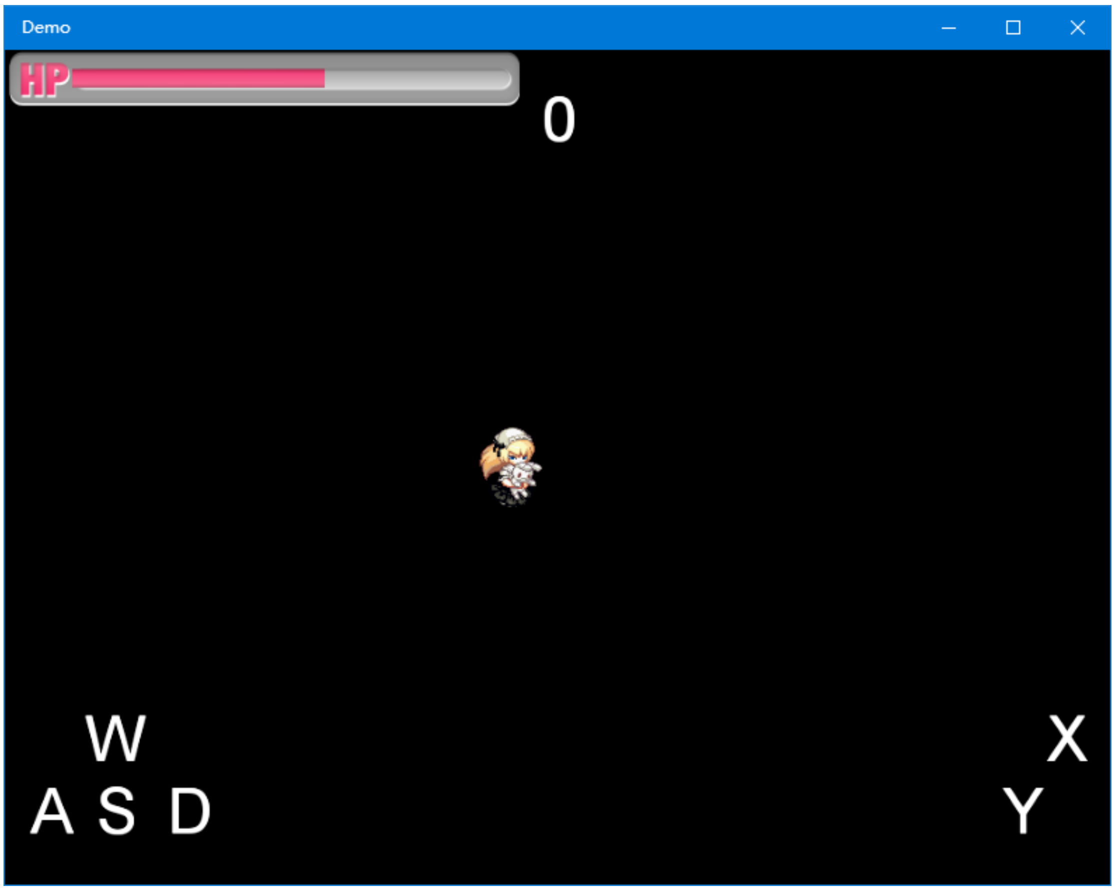


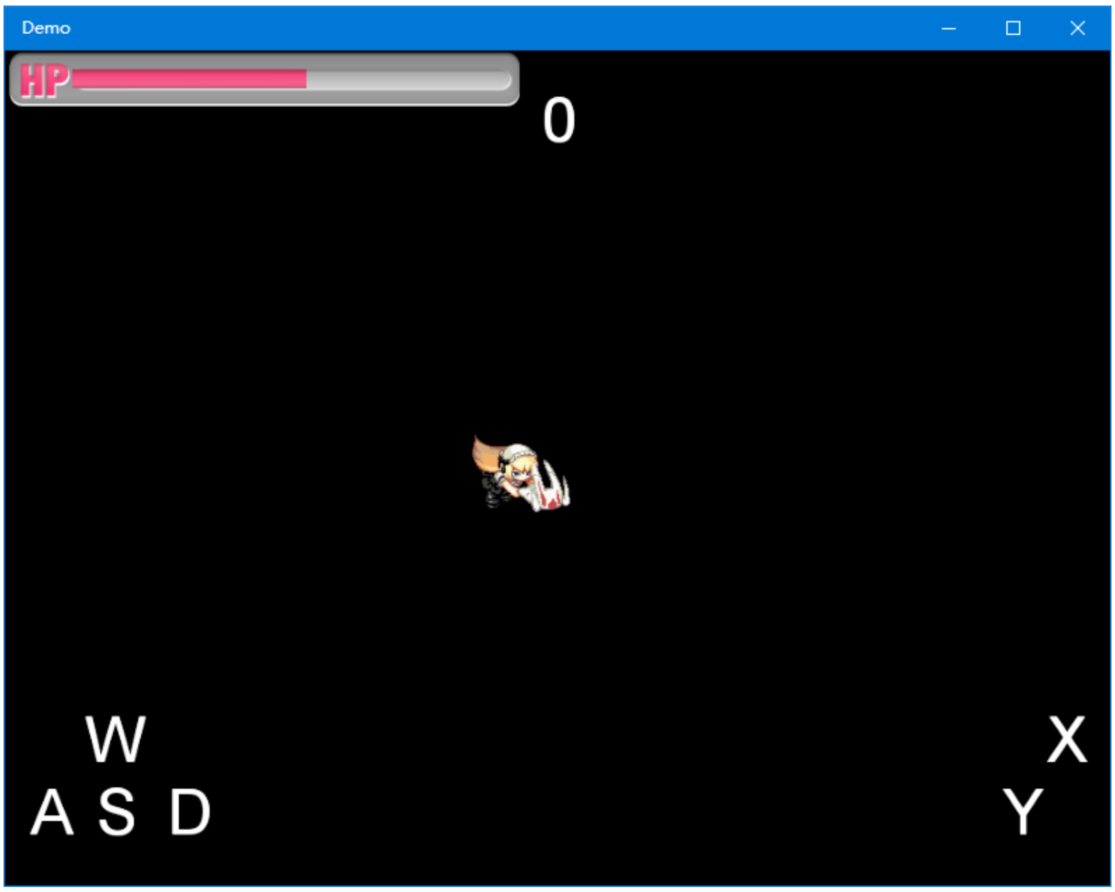

Demo
HP
47
W
A S D
X
Y

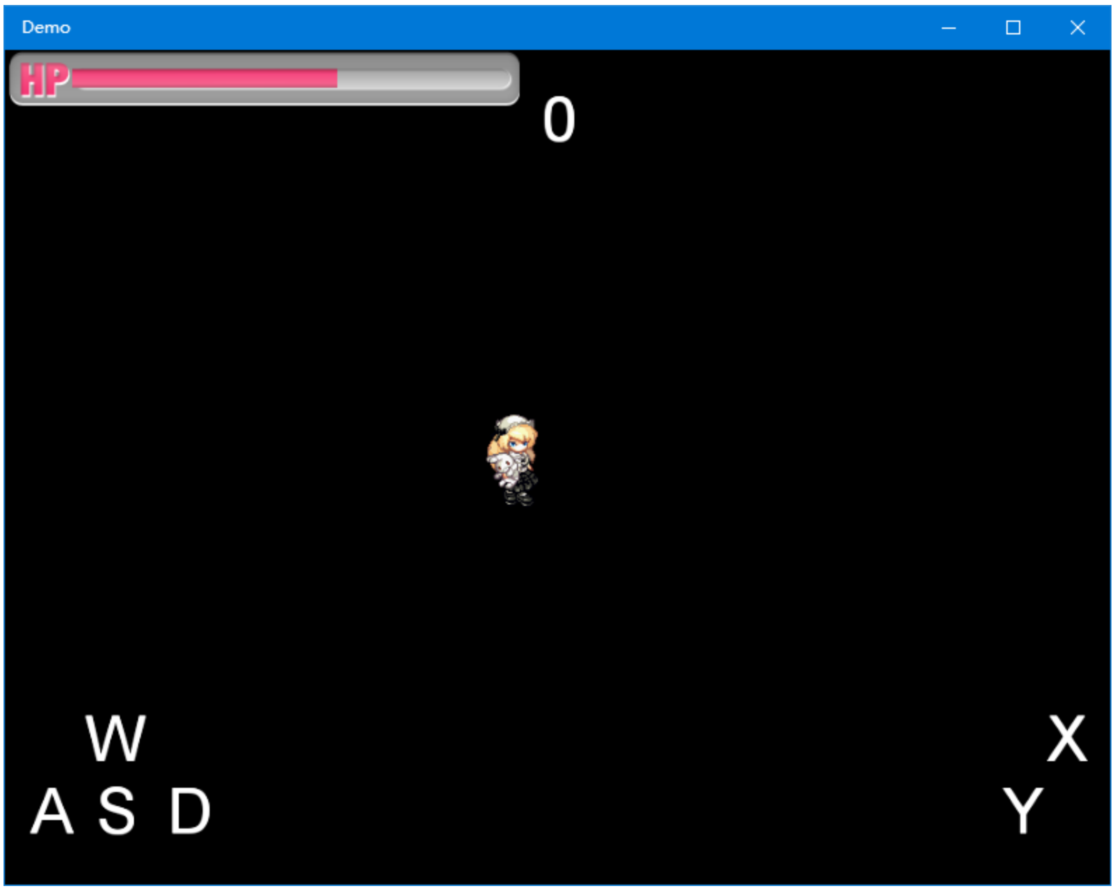

Demo
HP
3
W
A S D
X
Y


Demo
HP
0
W
A S D
X
Y


Demo
HP
0
W
A S D
X
Y


Demo
HP
0
W
A S D
X
Y


Demo
HP
0
W
A S D
X
Y



四．实验过程遇到的问题

问题一：精灵向前后左右走动的不同步，到达到目的还在原地走动

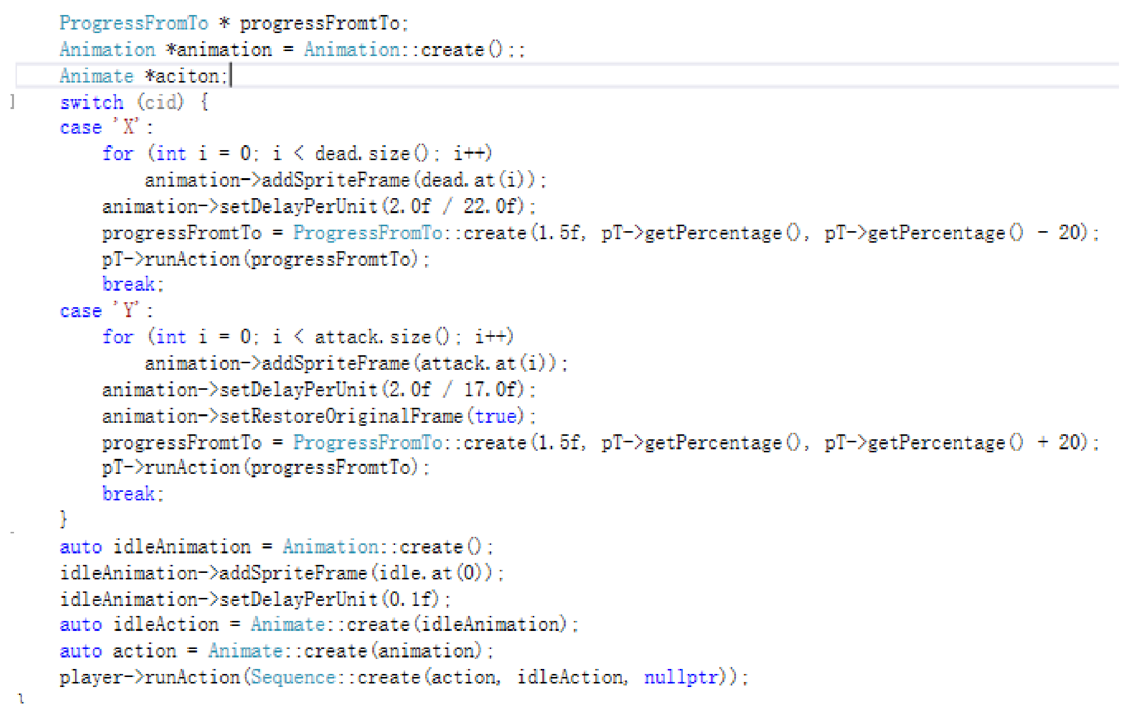
解决方法：因为走动的动画总共用了８帧图片，走到位置的时间为０．５ｆ，所以０．５／８即可算出平均每帧图片所需时间，这样就实现了同步

问题二：连续点击精灵向某一方向移动时，精灵在到达边界处会有点来回走动，但正常点击则不会出现该问题

出错原因:连续点击使精灵向某一方向移动时在获取精灵位置是会出现延迟现象，导致不同步。暂时只能将移动距离减少，使来回走动不那么明显。

问题三：连续点击使精灵进行死亡和攻击动画后，精灵无法恢复到闲置状态，而会停留在死亡和攻击动画中某一帧画面，同样正常点击不会出现该问题

解决方法：在函数最后才添加动作而不是判断中添加动作，改进代码如下：



```
    ProgressFromTo * progressFromtTo;
    Animation *animation = Animation::create();;
    Animate *aciton;
    switch (cid) {
    case 'X':
        for (int i = 0; i < dead.size(); i++)
            animation->addSpriteFrame(dead.at(i));
        animation->setDelayPerUnit(2.0f / 22.0f);
        progressFromtTo = ProgressFromTo::create(1.5f, pT->getPercentage(), pT->getPercentage() - 20);
        pT->runAction(progressFromtTo);
        break;
    case 'Y':
        for (int i = 0; i < attack.size(); i++)
            animation->addSpriteFrame(attack.at(i));
        animation->setDelayPerUnit(2.0f / 17.0f);
        animation->setRestoreOriginalFrame(true);
        progressFromtTo = ProgressFromTo::create(1.5f, pT->getPercentage(), pT->getPercentage() + 20);
        pT->runAction(progressFromtTo);
        break;
    }
    auto idleAnimation = Animation::create();
    idleAnimation->addSpriteFrame(idle.at(0));
    idleAnimation->setDelayPerUnit(0.1f);
    auto idleAction = Animate::create(idleAnimation);
    auto action = Animate::create(animation);
    player->runAction(Sequence::create(action, idleAction, nullptr));
```

五．思考与总结

本次实验很简单，资料非常详细，demo 也已经需要大部分代码，只需要在 demo 中添加要实现功能即可。但是对于连续点击出现的问题个人还是不是很清楚该如何解决，动作之间关系还不是很了解，所以还是需要进一步学习。