**Cocos2dx开发实验报告**

——刘亚辉 16340157

Cocos2dx第十一周实验报告

一、参考资料

1. PPT课件 (调度器，UI样式，序列帧动画)
2. <https://blog.csdn.net/wlk1229/article/details/55683358> （调度器）
3. <https://blog.csdn.net/natsu1211/article/details/37915207> （runAction回调）
4. <https://blog.csdn.net/decajes/article/details/38141121> (回调函数)
5. <https://blog.csdn.net/ac_huang/article/details/38072505> （ProgressTimer）

二、实验步骤

* 需求及流程：

1.添加倒计时、人物血条；

2.实现右侧X、Y按键播放不同的帧动画（X、Y不同时播放）；

3.实现左侧W、S、A、D四个虚拟按键使得角色可以移动；

4.X动画使得血条减少，Y动画使得血条增加。

* 详细实现过程：

1.计时器、人物血条的实现：

1. //定时器
2. schedule(schedule\_selector(HelloWorld::updateCustom), 1.0f, kRepeatForever, 0);
3. time = Label::createWithTTF(std::to\_string(dtime), "fonts/arial.ttf", 36);
4. time->setColor(Color3B(255,255,255));
5. time->setAnchorPoint(Vec2(0.5, 0.5));
6. time->setPosition(Vec2(origin.x+visibleSize.width/2, origin.y+visibleSize.height-50));
7. addChild(time);

利用自定义的调度器，设置回调函数，调度时间间隔为1秒，延迟为0。下方的代码是增加显示时间time的标签。

回调函数如下：

1. **void** HelloWorld::updateCustom(**float** dt)
2. {
3. dtime--;
4. log(dtime);
5. time->setString(std::to\_string(dtime));
6. **if** (dtime < 10)
7. time->setColor(Color3B(255,80,80));
8. **if** (dtime == 0)
9. unschedule(schedule\_selector(HelloWorld::updateCustom));
10. }

在回调函数中不断减少时间，并作更新显示。最终，当时间减少到0时，取消该调度器。

人物血条利用ProcessTimer实现：

1. //hp条
2. Sprite\* sp0 = Sprite::create("hp.png", CC\_RECT\_PIXELS\_TO\_POINTS(Rect(0, 320, 420, 47)));
3. Sprite\* sp = Sprite::create("hp.png", CC\_RECT\_PIXELS\_TO\_POINTS(Rect(610, 362, 4, 16)));
5. //使用hp条设置progressBar
6. pT = ProgressTimer::create(sp);
7. pT->setScaleX(90);
8. pT->setAnchorPoint(Vec2(0, 0));
9. pT->setType(ProgressTimerType::BAR);
10. pT->setBarChangeRate(Point(1, 0));
11. pT->setMidpoint(Point(0, 1));
12. pT->setPercentage(100);
13. pT->setPosition(Vec2(origin.x + 14 \* pT->getContentSize().width, origin.y + visibleSize.height - 2 \* pT->getContentSize().height));
14. addChild(pT, 1);
15. sp0->setAnchorPoint(Vec2(0, 0));
16. sp0->setPosition(Vec2(origin.x + pT->getContentSize().width, origin.y + visibleSize.height - sp0->getContentSize().height));
17. addChild(sp0, 0);

关于ProcessTimer的各个属性方法解释如下：

pT->setScaleX()：设置进度条的缩放大小；

pT->setPercentage(100)：设置进度条百分比，初始化为100%；

pT->setType()：设置进度条的形状radial，bar；

1. //设置进度条类型为条型
2. pT->setType(ProgressTimer::Type::BAR);

pT->setMidPoint()：设置进度条的起点位置；

对于radial形状的进度条，MidPoint即为中心点。对于bar形状的进度条：从左到右，需要设置MidPoint为Point（0，y），（x轴y轴范围为0-1）；从右到左，需要设置MidPoint为Point（1，y）；从上到下，需要设置MidPoint为Point（x，1）；从下到上需要设置MidPoint为Point（x，0）。上述代码中设置MidPoint为Point（0，1）表示进度条变化从左到右。

pT->setBarChangeRate()：设置x轴\y轴的改变；

Point（x，y）表示x轴与y轴的改变量。Point（1，0）表示竖直方向不变，水平方向改变100%。上述代码中设置为Point为Point（1，0）即表示竖直方向不变，水平方向改变100%。

2.实现右侧X、Y按键播放不同的帧动画（X、Y不同时播放）；

首先实现dead和attack动画：

1. // 攻击动画
2. attack.reserve(17);
3. **for** (**int** i = 0; i < 17; i++) {
4. auto frame = SpriteFrame::createWithTexture(texture, CC\_RECT\_PIXELS\_TO\_POINTS(Rect(113 \* i, 0, 113, 113)));
5. attack.pushBack(frame);
6. }
8. // 死亡动画(帧数：22帧，高：90，宽：79）
9. auto texture2 = Director::getInstance()->getTextureCache()->addImage("$lucia\_dead.png");
10. dead.reserve(22);
11. **for** (auto i = 0; i < 22; i++) {
12. auto frame = SpriteFrame::createWithTexture(texture2, CC\_RECT\_PIXELS\_TO\_POINTS(Rect(79 \* i, 0, 79, 90)));
13. dead.pushBack(frame);
14. }

利用SpriteFrame的createWithTexture方法来读取图片或者图像帧保存在向量数组中。

1. //按钮X、Y
2. auto X = Label::createWithTTF("X","fonts/arial.ttf", 36);
3. auto XButton = MenuItemLabel::create(X, [&](cocos2d::Ref\* sender) {
4. **if** (flag) {
5. flag = 0;
6. auto decrease = ProgressFromTo::create(0.4, pT->getPercentage(), pT->getPercentage() - 20);
7. pT->runAction(decrease);
8. auto deadAnimation = Animation::createWithSpriteFrames(dead);
9. deadAnimation->setDelayPerUnit(0.12f);
10. deadAnimation->setRestoreOriginalFrame(**true**);
11. auto deadAction = Animate::create(deadAnimation);
12. player->runAction(Sequence::create(deadAction, CallFunc::create([&]{
13. flag = 1;
14. }),NULL));
15. }
16. });
17. XButton->setPosition(Vec2(origin.x+visibleSize.width-15, origin.y+50));
19. auto Y = Label::createWithTTF("Y", "fonts/arial.ttf", 36);
20. auto YButton = MenuItemLabel::create(Y, [&](cocos2d::Ref\* sender) {
21. **if** (flag) {
22. flag = 0;
23. auto increase = ProgressFromTo::create(0.4, pT->getPercentage(), pT->getPercentage() + 20);
24. pT->runAction(increase);
25. auto attackAnimation = Animation::createWithSpriteFrames(attack);
26. attackAnimation->setDelayPerUnit(0.12f);
27. attackAnimation->setRestoreOriginalFrame(**true**);
28. auto attackAction = Animate::create(attackAnimation);
29. player->runAction(Sequence::create(attackAction, CallFunc::create([&]{
30. flag = 1;
31. }), NULL));
32. }
33. });
34. YButton->setPosition(Vec2(origin.x + visibleSize.width - 50, origin.y+15));

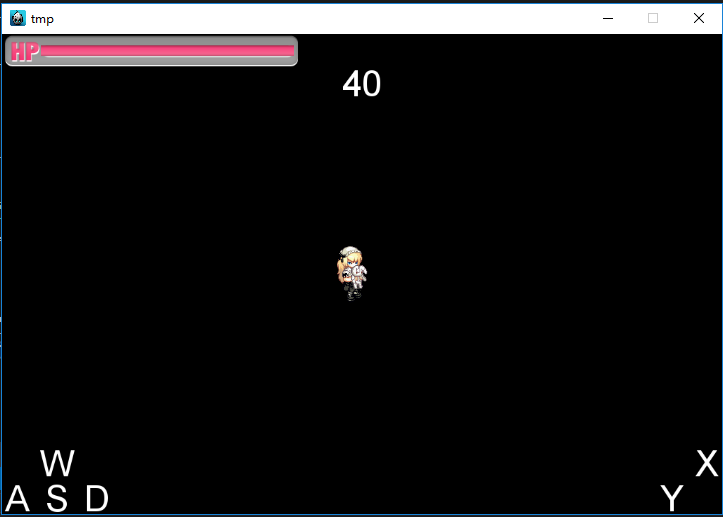
在按钮回调函数中，我使用lambda表达式实现，调用ProgressFromTo类的方法，实现血条的回复与减少；创建deadAnimation和attackAnimation动画对象及其各自动作对象，然后调用runAction使用动画动作。设置flag布尔型变量，该变量确保同一时间只有dead和attack一个动作可以执行，利用Sequence顺序动作对象，在执行完相应动作之后，执行一个回调函数来修改flag的值，使得dead与attack互斥。CallFunc：：create参数可以用CC\_CALLBACK\_0回调函数，效果相同。

1. 实现左侧W、S、A、D四个虚拟按键使得角色可以移动；
2. auto W = Label::createWithTTF("W", "fonts/arial.ttf", 36);
3. auto WButton = MenuItemLabel::create(W, [=](cocos2d::Ref\* sender) {
4. auto runAnimation = Animation::createWithSpriteFrames(run);
5. runAnimation->setDelayPerUnit(0.12f);
6. runAnimation->setRestoreOriginalFrame(**true**);
7. auto runAction = Animate::create(runAnimation);
9. **if** (player->getPosition().y + 30 < visibleSize.height - player->getContentSize().height) {
10. auto moveTo = MoveTo::create(0.5f, Vec2(player->getPosition().x, player->getPosition().y + 30));
11. player->runAction(Spawn::create(runAction, moveTo, NULL));
12. }
13. **else**
14. player->runAction(runAction);
15. });
16. WButton->setPosition(Vec2(origin.x + 55, origin.y + 50));

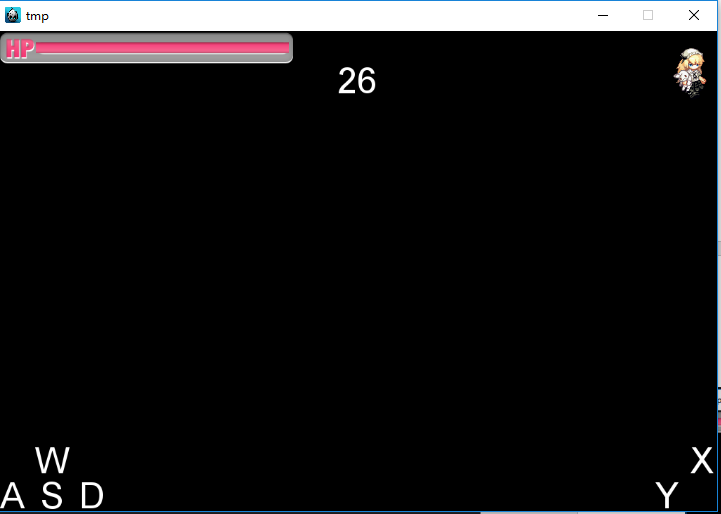
WButton按钮实现如上，首先创建runAnimation动画对象及其runAnimate动作对象，然后尝试让Sprite移动，判断是否超出界面范围，如果不超过，则利用Spawn类型，使得runAction和moveTo动作对象同时执行；否则只执行runAction对象。其他按钮实现与此类似。

三、关键步骤截图

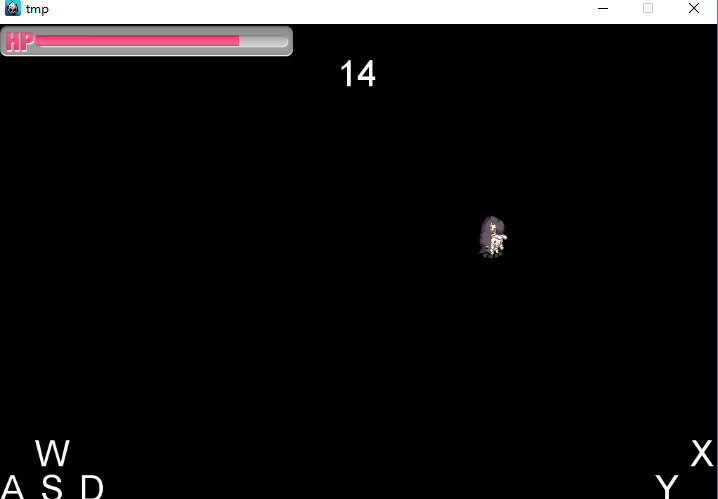
初始化界面：



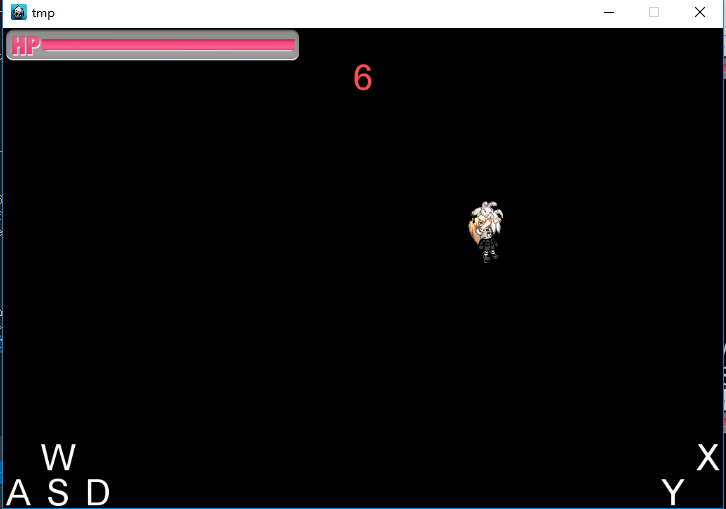
运动到边界：



点击X：



点击Y：



四、亮点与改进

实现血条的增加与减少。

五、遇到的问题

X、Y互斥执行runAction的回调实现：

通过设置互斥量flag，在动作执行之前判断flag是否为1，为1的话，表示当前Sprite处于空闲状态，否则的话，表示当前Sprite正在执行X、Y动作，则不响应该点击事件；若进入执行该动作，将flag置为0，表示Sprite为动作状态，然后在runAction中利用Sequence顺序动作对象实现回调函数，在相应动作执行完后，在回调函数中修改flag为1，将Sprite置为空闲状态。

W、S、A、D移动不出边界：

判断如果出边界的话，不应该执行moveTo方法。之前我在实现时，在不出边界的情况对x和y也赋予相应的值（即当前位置），此时若调用moveTo，在瞬间多次点击移动按钮的时候，会出现Sprite离开边界的情况。