《现代操作系统应用开发》HW11 实验报告

姓名：羊伊 学号：15331349

一、参考资料：

homework11.pdf

二、实验步骤：

1.界面所有字体要求：使用字体 arial.ttf，字体大小为 36。
用 TTFConfig 统一设置字体，并在 MenuItemLabel 创建的时候应用。

```
// 添加MenuItemLabel
TTFConfig ttfConfig;
ttfConfig.fontFilePath = "arial.ttf";
ttfConfig.fontSize = 36;
ttfConfig.outlineSize = 2;
auto X = MenuItemLabel::create(Label::createWithTTF(ttfConfig, "X"), CC_CALLBACK_1
    (HelloWorld::DeadMenuCallback, this));
auto Y = MenuItemLabel::create(Label::createWithTTF(ttfConfig, "Y"), CC_CALLBACK_1
    (HelloWorld::AttackMenuCallback, this));
auto W = MenuItemLabel::create(Label::createWithTTF(ttfConfig, "W"), CC_CALLBACK_1
    (HelloWorld::WMoveMenuCallback, this));
auto A = MenuItemLabel::create(Label::createWithTTF(ttfConfig, "A"), CC_CALLBACK_1
    (HelloWorld::AMoveMenuCallback, this));
auto S = MenuItemLabel::create(Label::createWithTTF(ttfConfig, "S"), CC_CALLBACK_1
    (HelloWorld::SMoveMenuCallback, this));
auto D = MenuItemLabel::create(Label::createWithTTF(ttfConfig, "D"), CC_CALLBACK_1
    (HelloWorld::DMoveMenuCallback, this));
X->setPosition(Vec2(origin.x + visibleSize.width-X->getContentSize().width,
                    origin.y+X->getContentSize().height+50));
Y->setPosition(Vec2(origin.x + visibleSize.width-Y->getContentSize().width-50,
                origin.y+Y->getContentSize().height));
W->setPosition(Vec2(origin.x + Y->getContentSize().width+50,
                    origin.y+Y->getContentSize().height+50));
A->setPosition(Vec2(origin.x + Y->getContentSize().width,
                    origin.y+Y->getContentSize().height));
S->setPosition(Vec2(origin.x + Y->getContentSize().width+50,
                    origin.y+Y->getContentSize().height));
D->setPosition(Vec2(origin.x + Y->getContentSize().width+100,
                    origin.y+Y->getContentSize().height));
auto menu = Menu::create(X,Y, W, A, S, D, NULL);
menu->setPosition(Vec2::ZERO);
this->addChild(menu, 2);
```

2. 左边 wasd4 个虚拟按键能控制角色移动
先设置”move”这个动作，并存在 AnimationCache 里以待调用。

```
// move

auto texture3 = Director::getInstance()->getTextureCache()->addImage("$lucia_forward.png");
run.reserve(8);
for (int i = 0; i < 8; i++) {
    auto frame = SpriteFrame::createWithTexture(texture3, CC_RECT_PIXELS_TO_POINTS(Rect(68*
        i,0,68,101)));
    run.pushBack(frame);
}
animation = Animation::createWithSpriteFrames(run, 0.05f);
AnimationCache::getInstance()->addAnimation(animation, "move");
```

以 W 上移为例，先算出要移动的位置，然后移动。

```
void HelloWorld::WMoveMenuCallback(Ref* pSender) {
    auto loc = player->getPosition()+Vec2(0, 25);
    if (!canAct())return;
    Move(loc);
}
```

在 Move 中判断要移到的位置是否在可视范围内，是则移动。移动的时候调用 AnimationCache 中的 move 的动作并且叠加 MoveTo 的动作。

```
void HelloWorld::Move(Vec2 loc) {
    if (loc.x < origin.x || loc.x > origin.x+visibleSize.width||
        loc.y < origin.y ||loc.y > origin.x+visibleSize.height)
            return;
    player->runAction(MoveTo::create(0.4, loc));
    Animate* _move = Animate::create(AnimationCache::getInstance()->getAnimation("move"));
    player->runAction(_move);
}
```

3. 右边 2 个虚拟按键 x，y 能控制角色播放不同的帧动画
和 move 一样，先设置动作然后放在 AnimationCache 中，在 X，Y 的回调函数中调用动作。

```
    //  ------------------------------------------------------------

    //  attack
    auto texture = Director::getInstance()->getTextureCache()->addImage("$lucia_2.png");
    auto frame0 = SpriteFrame::createWithTexture(texture, CC_RECT_PIXELS_TO_POINTS(Rect(0, 0,
        113, 113)));
    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);
    }
    auto animation = Animation::createWithSpriteFrames(attack, 0.1f);
    animation->setRestoreOriginalFrame(true);  //  回到第一帧
    AnimationCache::getInstance()->addAnimation(animation, "attack");

    //  ------------------------------------------------------------

    //  dead
    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,0,90,90)));
        dead.pushBack(frame);
    }
    animation = Animation::createWithSpriteFrames(dead, 0.1f);
    animation->setRestoreOriginalFrame(true);
    AnimationCache::getInstance()->addAnimation(animation, "dead");
```

```
void HelloWorld::DeadMenuCallback(Ref* pSender) {
    if (!canAct()) return;
    if(percent >= 20) pT->runAction(CCProgressTo::create(2, percent-20));
    percent -= 20;
    Animate* _dead = Animate::create(AnimationCache::getInstance()->getAnimation("dead"));
    player->runAction(_dead);

}
void HelloWorld::AttackMenuCallback(Ref* pSender) {
    if (!canAct()) return;
    if(percent < 100) pT->runAction(CCProgressTo::create(2, percent+20));
        percent += 20;
    Animate* _attack = Animate::create(AnimationCache::getInstance()->getAnimation("attack"));
    player->runAction(_attack);
}
```

'CCProgressTo' is deprecated

'CCProgressTo' is deprecated

4. 添加人物血条
demo 已给出

5. 角色不会移动到可视窗口外
在 Move()中判断。
6. X、Y 播放的动画不能同时播放
判断现在是否有在运行的动作，若有，则不能有新的动作。

```
bool HelloWorld::canAct() {
    return (getActionManager()->getNumberOfRunningActionsInTarget(player) < 1);
}
```

7.添加倒计时
用自定义的调度器，设置间隔为 1 秒。

```
    dtime = 30;
    String* temp = String::createWithFormat("%d", dtime);
    time = Label::createWithTTF(ttfConfig, temp->_string);
    time->setPosition(Vec2(origin.x+visibleSize.width/2, origin.y+visibleSize.height-time->
        getContentSize().height));
    addChild(time, 1);
    schedule(schedule_selector(HelloWorld::update), 1.0f, kRepeatForever, 0);
```

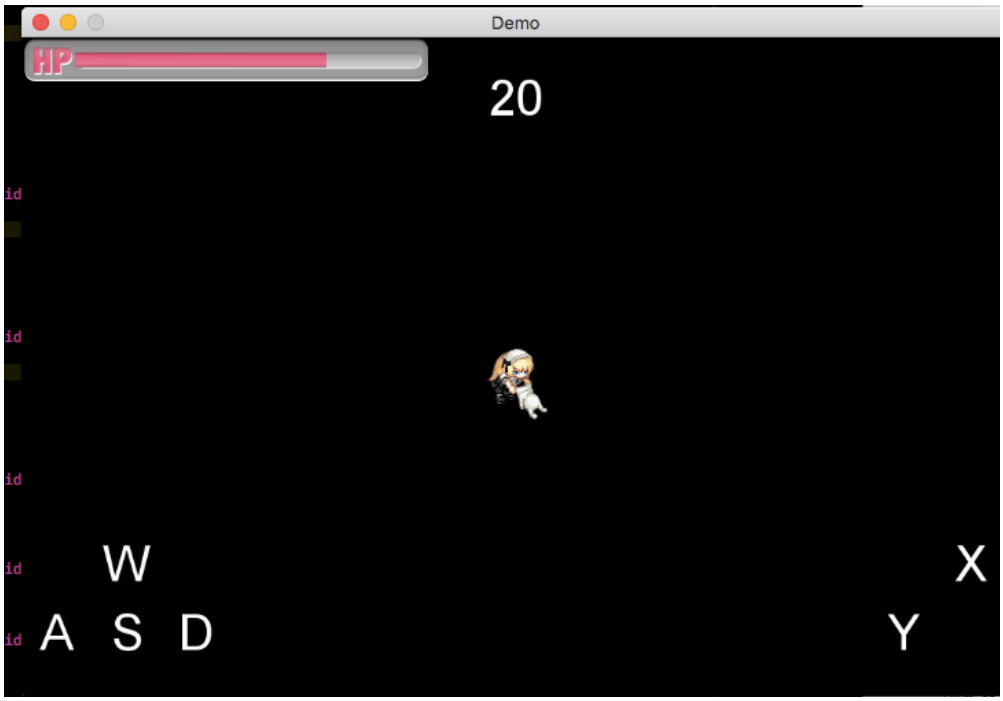
重载 update，更新倒计时。倒计时为零时 Pause，停止所有操作。

```
void HelloWorld::update(float dt) {
    if (dtime > 0) dtime--;
    else Director::getInstance()->pause();
    String* temp = String::createWithFormat("%d", dtime);
    time->setString(temp->_string);
}
```

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（加分项）
让血条 pT 根据 X, Y 按键，增加/减少血条。

```
if(percent >= 20) pT->runAction(CCProgressTo::create(2, percent-20));
percent -= 20;
```

三、效果截图



四、收获与感受

本次实验的重点是 Action 这部分，主要是将图按帧存到 Vector 里并设置间隔时间，来形成动画效果，并存在 AnimationCache 中。在某个菜单按钮的回调函数中调用动画，来形成响应，达到很好的交互效果。虽然我对游戏开发不太感兴趣，但看着一个小人能受控制动来动去还挺有成就感的。