现代操作系统应用开发实验报告

学号： 14970011　　**班级：** 2015 级教务 2 班

姓名： 宋思亭　　**实验名称：** homework11

一．参考资料

作业要求文档, 课件 PPT，

官网：http://www.cocos.com/

Github：https://github.com/cocos2d/cocos2d-x

用户手册：http://www.cocos2d-x.org/wiki/Cocos2d-x

商店：http://store.cocos.com/

API:　http://api.cocos.com/

cocos2d 无法打开包含文件：http://blog.csdn.net/cdamber/article/details/44700817

二．实验步骤

1. 阅读作业需求和课件 PPT，了解阅读课件内容以及作业要求，了解 cocos-2dx 中的基础界面元素动作 ,帧动画和调度器。学习进度条 ProgressBar 设置方法，动作 Aciton 使用方法，动作与 Repeat 动作重复结合，帧动画 SpriteFrame，自定义调度器(scheduler)的使用和取消等基础知识。
2. 写游戏类.h 文件

```
class HelloWorld : public cocos2d::Layer
{
public:
    static cocos2d::Scene* createScene();
    virtual bool init();
    void moveEvent(Ref*, char);
    void actionEvent(Ref*, char);
    void stopAc();
    void updateTime(float dt);
    CREATE_FUNC(HelloWorld);

private:
    cocos2d::Sprite* player;
    cocos2d::ProgressTimer* timer;
    cocos2d::Vector<SpriteFrame*> attack;
    cocos2d::Vector<SpriteFrame*> dead;
    cocos2d::Vector<SpriteFrame*> run;
    cocos2d::Vector<SpriteFrame*> idle;
    cocos2d::Size visibleSize;
    cocos2d::Vec2 origin;
    cocos2d::Label* time;
    int dtime;
    int hp = 100;
    bool actionOn = true;
    bool rotate = false;
    bool isEnd = false;
};
```

设计人物移动，人物动作函数，计时器函数；设计人物，计时器，动作等变量。

其中，actionOn 用于记录人物是否正在执行动作，rotate 记录人物是否转向，

isEnd 记录游戏是否结束。

3. 写 init 函数

```
// 设置背景
visibleSize = Director::getInstance()->getVisibleSize();
origin = Director::getInstance()->getVisibleOrigin();
float winw = visibleSize.width;  // 获取屏幕宽度
float winh = visibleSize.height; // 获取屏幕高度
auto bg = Sprite::create("background1.png");
bg->setPosition(Vec2(winw / 2 + origin.x, winh / 2 + origin.y));
float spx = bg->getTextureRect().getMaxX();
bg->setScaleX(winw / spx);//  背景缩放
bg->setScaleY(winw / spx);
this->addChild(bg, 0);

//创建一张贴图
auto texture = Director::getInstance()->getTextureCache()->addImage("$lucia_2.png");
//从贴图中以像素单位切割，创建关键帧
auto frame0 = SpriteFrame::createWithTexture(texture, CC_RECT_PIXELS_TO_POINTS(Rect(0, 0, 113, 113)));
//使用第一帧创建精灵
player = Sprite::createWithSpriteFrame(frame0);
player->setPosition(Vec2(origin.x + winw / 2, origin.y + winh /2));
addChild(player, 3);
```

显示计时器，调用 updateTime 函数：

```
//倒计时
time = Label::createWithTTF("03:00", "fonts/Marker Felt.ttf", 40);
//周期性调用调度器
schedule(schedule_selector(HelloWorld::updateTime), 1.0f, kRepeatForever, 0);
dtime = 180;
time->setPosition(Vec2(origin.x + visibleSize.width / 2, origin.y + visibleSize.height - time->getContentSize().height - 20));
addChild(time);
```

设计 hp 进度条：

```
//设置hp条progressBar
Sprite* sp0 = Sprite::create("hp.png", CC_RECT_PIXELS_TO_POINTS(Rect(0, 320, 420, 47)));
Sprite* sp = Sprite::create("hp.png", CC_RECT_PIXELS_TO_POINTS(Rect(610, 362, 4, 16)));
timer = ProgressTimer::create(sp);
timer->setScaleX(90);
timer->setAnchorPoint(Vec2(0, 0));
timer->setType(ProgressTimerType::BAR);
timer->setBarChangeRate(Point(1, 0));
timer->setMidpoint(Point(0, 1));
timer->setPercentage(100);
timer->setPosition(Vec2(origin.x + 14 * timer->getContentSize().width,origin.y + visibleSize.height - 2 * timer->getContentSize().height));
addChild(timer,1);
sp0->setAnchorPoint(Vec2(0, 0));
sp0->setPosition(Vec2(origin.x + timer->getContentSize().width, origin.y + visibleSize.height - sp0->getContentSize().height));
addChild(sp0,0);
```

添加动画，采用 SpriteFrame 帧动画效果：

```
//静态动画
idle.reserve(1);
idle.pushBack(frame0);

//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);
}

//死亡动画
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);
}

//运动动画
auto texture3 = Director::getInstance()->getTextureCache()->addImage("$lucia_forward.png");
run.reserve(9);
for (int i = 0; i < 9; i++) {
    if (i < 8) {
        auto frame = SpriteFrame::createWithTexture(texture3, CC_RECT_PIXELS_TO_POINTS(Rect(68 * i, 0, 68, 101)));
        run.pushBack(frame);
    } else {
        auto frame = SpriteFrame::createWithTexture(texture3, CC_RECT_PIXELS_TO_POINTS(Rect(0, 0, 68, 101)));
        run.pushBack(frame);
    }
}
```

添加按钮：

```
//Label
auto menuLabel1 = Label::createWithTTF("W", "fonts/Marker Felt.ttf", 40);
auto menuLabel2 = Label::createWithTTF("S", "fonts/Marker Felt.ttf", 40);
auto menuLabel3 = Label::createWithTTF("A", "fonts/Marker Felt.ttf", 40);
auto menuLabel4 = Label::createWithTTF("D", "fonts/Marker Felt.ttf", 40);
auto menuLabel5 = Label::createWithTTF("X", "fonts/Marker Felt.ttf", 40);
auto menuLabel6 = Label::createWithTTF("Y", "fonts/Marker Felt.ttf", 40);
//menuItem
auto item1 = MenuItemLabel::create(menuLabel1, CC_CALLBACK_1(HelloWorld::moveEvent, this,'W'));
auto item2 = MenuItemLabel::create(menuLabel2, CC_CALLBACK_1(HelloWorld::moveEvent, this,'S'));
auto item3 = MenuItemLabel::create(menuLabel3, CC_CALLBACK_1(HelloWorld::moveEvent, this,'A'));
auto item4 = MenuItemLabel::create(menuLabel4, CC_CALLBACK_1(HelloWorld::moveEvent, this,'D'));
auto item5 = MenuItemLabel::create(menuLabel5, CC_CALLBACK_1(HelloWorld::actionEvent, this, 'X'));
auto item6 = MenuItemLabel::create(menuLabel6, CC_CALLBACK_1(HelloWorld::actionEvent, this, 'Y'));
//setPosition
item3->setPosition(Vec2(origin.x+item3->getContentSize().width,origin.y+item3->getContentSize().height));
item4->setPosition(Vec2(item3->getPosition().x + 3 * item4->getContentSize().width, item3->getPosition().y));
item2->setPosition(Vec2(item3->getPosition().x + 1.5*item2->getContentSize().width, item3->getPosition().y));
item1->setPosition(Vec2(item2->getPosition().x, item2->getPosition().y + item1->getContentSize().height));
item5->setPosition(Vec2(origin.x+visibleSize.width-item5->getContentSize().width,item1->getPosition().y));
item6->setPosition(Vec2(item5->getPosition().x-item6->getContentSize().width,item3->getPosition().y));

auto menu = Menu::create(item1, item2, item3, item4, item5, item6, NULL);
menu->setPosition(Vec2(0, 0));
addChild(menu, 1);
```

4. 写 moveEvent 函数，左边 wasd4 个虚拟按键控制角色移动时调用，并保证角色不会移动到可视窗口外

```
void HelloWorld::moveEvent(Ref*, char cid) {
    if (actionOn && !isEnd) {
        actionOn = false;
        auto s = Director::getInstance()->getWinSize();
        auto position = player->getPosition();
        auto animation_run = Animation::createWithSpriteFrames(run, 0.05f);
        auto animate_run = Animate::create(animation_run);
        int x = position.x;
        int y = position.y;
        switch (cid) {
        case 'W':
            y += 50;
            if (y > s.height - 20) {
                y = s.height - 20;
            }
            break;
        case 'A':
            x -= 50;
            if (x < origin.x + 20) {
                x = origin.x + 20;
            }
            if (!rotate) {
                player->setRotationY(180);
                rotate = true;
            }
            break;
```

```
        case 'S':
            y -= 50;
            if (y < origin.y + 20) {
                y = origin.y + 20;
            }
            break;
        case 'D':
            x = x + 50;
            if (x >= s.width - 20) {
                x = s.width - 20;
            }
            if (rotate) {
                player->setRotationY(0);
                rotate = false;
            }
            break;
        }
        FiniteTimeAction *runAction = MoveTo::create(0.5, Point(x, y));
        FiniteTimeAction *repeatRunAction = Repeat::create(animate_run, 1);
        auto stopAction = CallFunc::create(CC_CALLBACK_0(HelloWorld::stopAc, this));
        player->runAction(Sequence::create(Spawn::create(runAction, repeatRunAction, NULL), stopAction, NULL));
    }
}
```

帧动画 run 动作与 Repeat 重复结合，使人物看起来在跑动。采用 origin 和 getWinSize 控制人物不会移动到可视窗口外，当人物左右移动时，判定是否需要 rotate 转向，setRotationY 控制人物翻转。

5. 写 moveEvent 函数，点击虚拟按键 x 或 y 时触发，点击 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

```
void HelloWorld::actionEvent(Ref*, char cid) {
    if (actionOn && !isEnd) {
        FiniteTimeAction *deadAction = Repeat::create(Animate::create(Animation::createWithSpriteFrames(dead, 0.1f)), 1);
        FiniteTimeAction *attackAction = Repeat::create(Animate::create(Animation::createWithSpriteFrames(attack, 0.1f)), 1);
        FiniteTimeAction *idleAction = Repeat::create(Animate::create(Animation::createWithSpriteFrames(idle, 0.1f)), 1);
        auto stopAction = CallFunc::create(CC_CALLBACK_0(HelloWorld::stopAc, this));
        switch (cid) {
            case 'Y':
                if (hp < 100) {
                    hp = hp + 20 >= 100 ? 100 : hp + 20;
                }
                actionOn = false;
                player->runAction(Sequence::create(attackAction, idleAction, stopAction, NULL));
                break;
            case 'X':
                if (hp > 0) {
                    hp = hp - 20 <= 0 ? 0 : hp - 20;
                }
                actionOn = false;
                player->runAction(Sequence::create(deadAction, idleAction, stopAction, NULL));
                break;
        }
        CCProgressTo* progress = CCProgressTo::create(2, hp);
        timer->runAction(progress);
    }
}
```

点击 Y 则实现女孩攻击，静止的动作序列。并且 hp 增加 20，如果超过满血条则保持不变；点击 X 则实现女孩死亡，静止的动作序列。并且 hp 减少 20，如果低于 0 血条则保持不变。最后执行血条变化的动画。

6. 写 updateTime 类，实现倒计时功能：

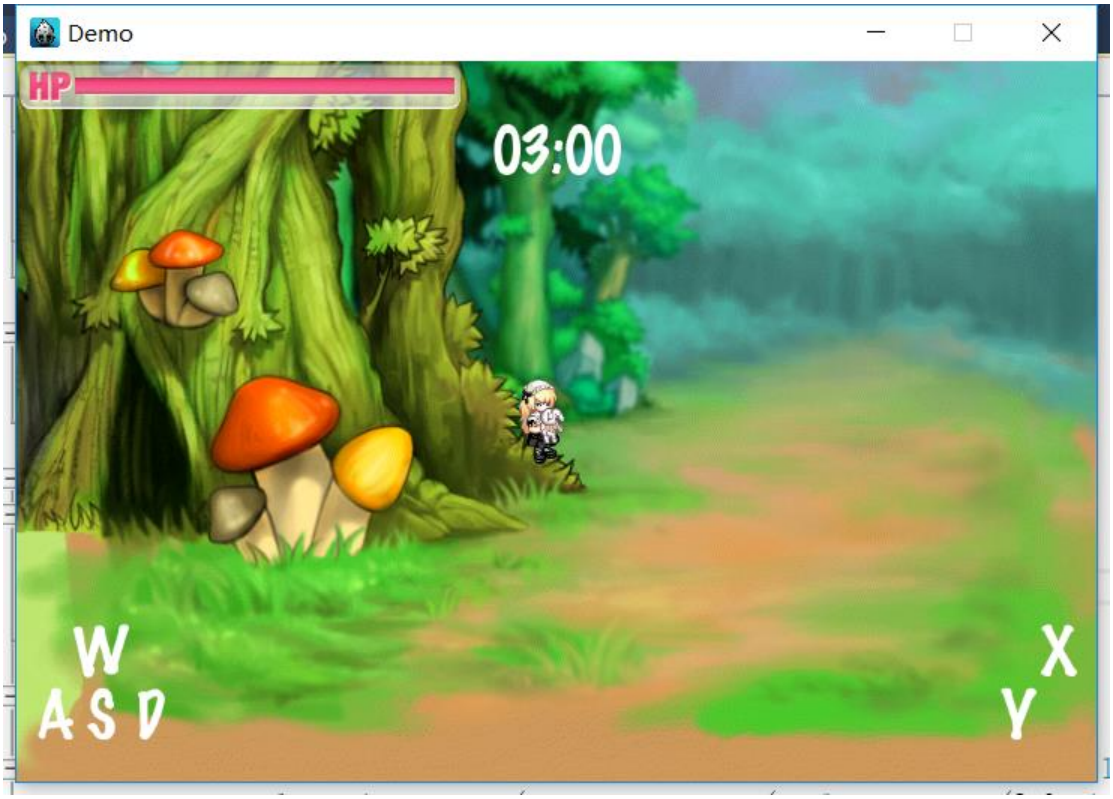
```
void HelloWorld::updateTime(float dt) {
    if (timer->getPercentage() == 0) {
        unschedule(schedule_selector(HelloWorld::updateTime));
        return;
    }
    --dtime;
    if (dtime < 0) {
        unschedule(schedule_selector(HelloWorld::updateTime));
        timer->runAction(Sequence::create(CCProgressTo::create(2, 0), CallFunc::create([this]() {
                player->runAction(Sequence::create(ScaleTo::create(2.0, 1.0), FadeOut::create(1.0), nullptr)); // 人物消失
                auto over = Sprite::create("over.png");
                float winw = visibleSize.width;  // 获取屏幕宽度
                float winh = visibleSize.height; // 获取屏幕高度
                over->setPosition(Vec2(winw / 2 + origin.x, winh / 2 + origin.y));
                float spx = over->getTextureRect().getMaxX();
                over->setScaleX(winw / spx);//  背景缩放
                over->setScaleY(winw / spx);
                this->addChild(over, 2);
                isEnd = true;
            }), nullptr));
        return;
    }
    string t = "0";
    char minute[5], second[5];
    _itoa(dtime / 60, minute, 10);
    _itoa(dtime % 60, second, 10);
    t += minute;
    t += ':';
    if (dtime % 60 < 10) t += '0';
    t += second;
    time->setString(t);
}
```

使用 string 变量记录改变的 dtime 时间并转化为时间形式表示，若时间到，显示人物消失的动画，和游戏结束的图片提示。

7. 调试项目

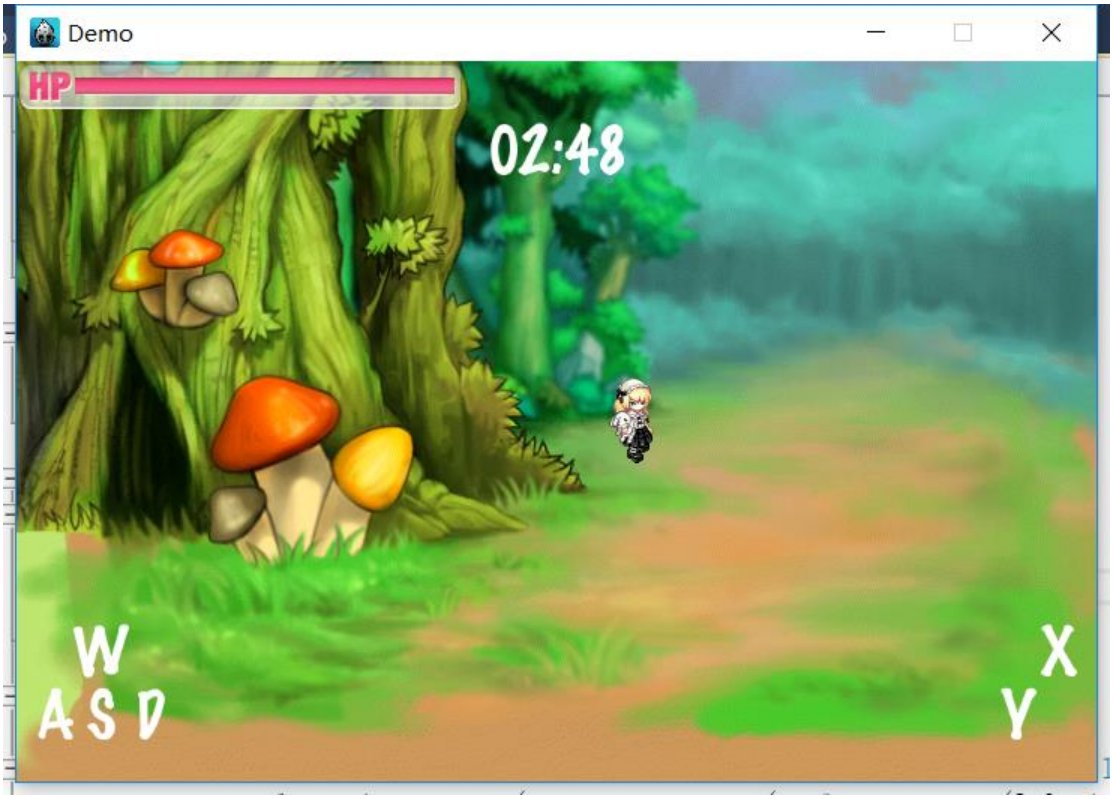
三．实验结果截图

1. 打开游戏

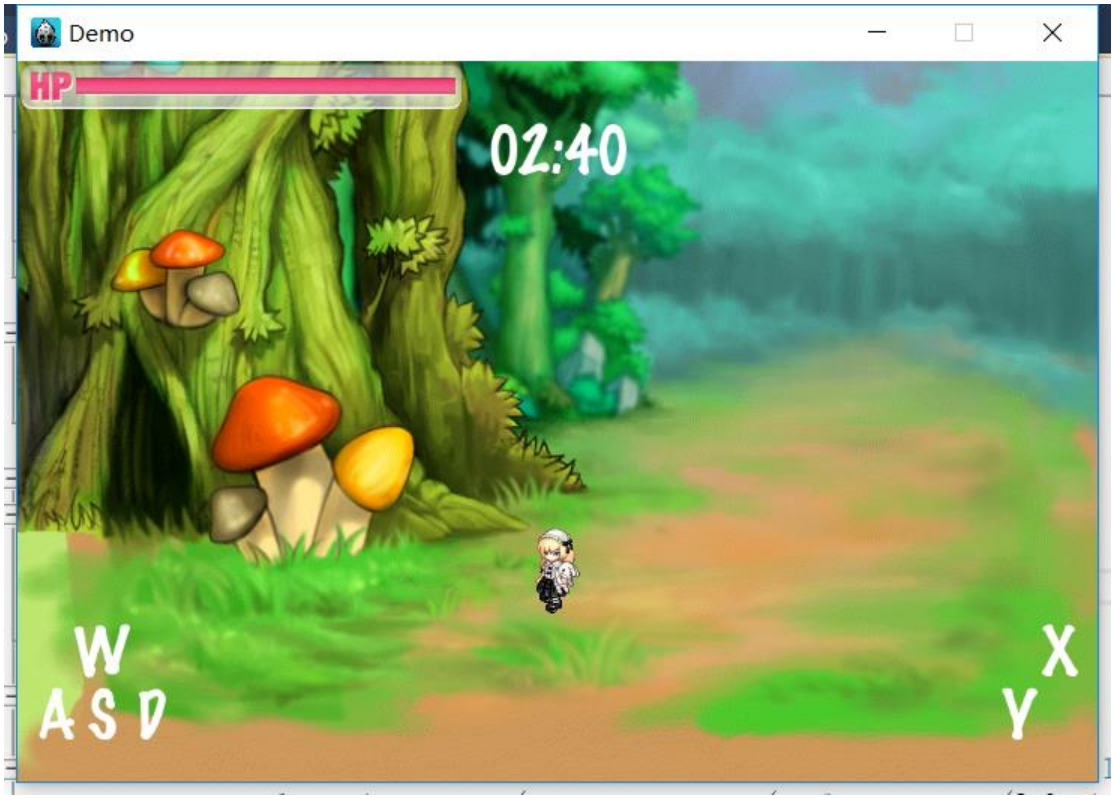


2. 人物走动

向右

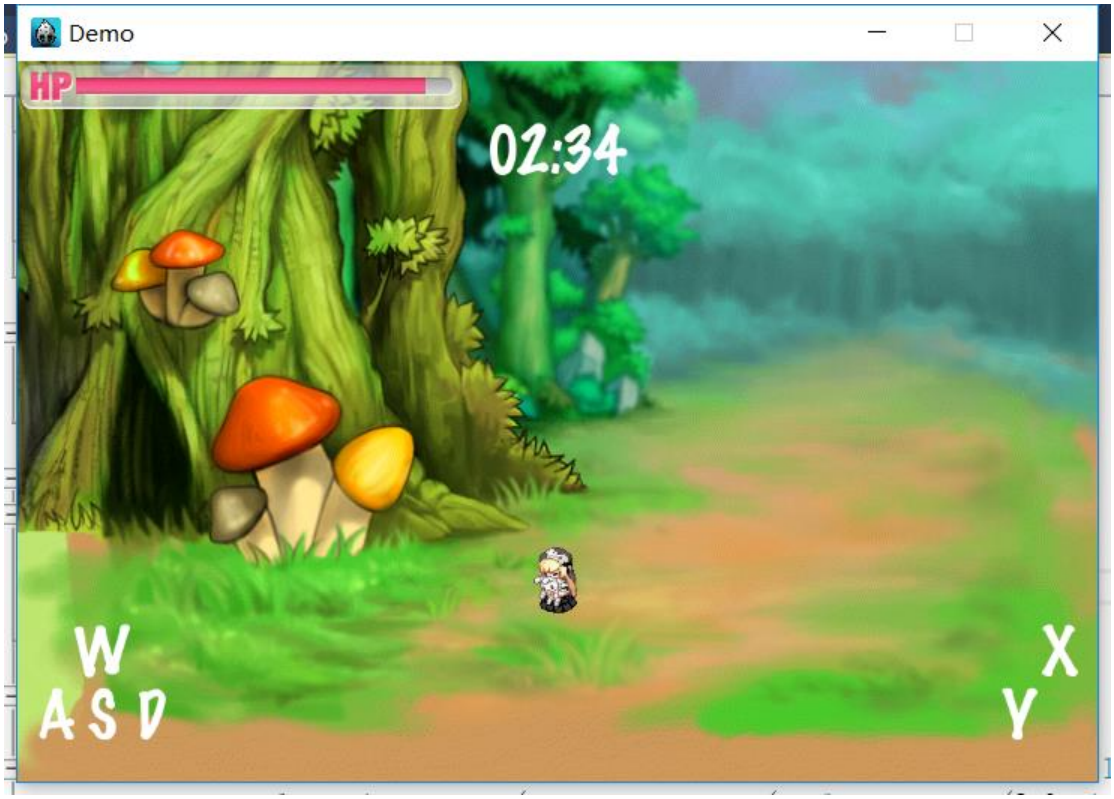


向左

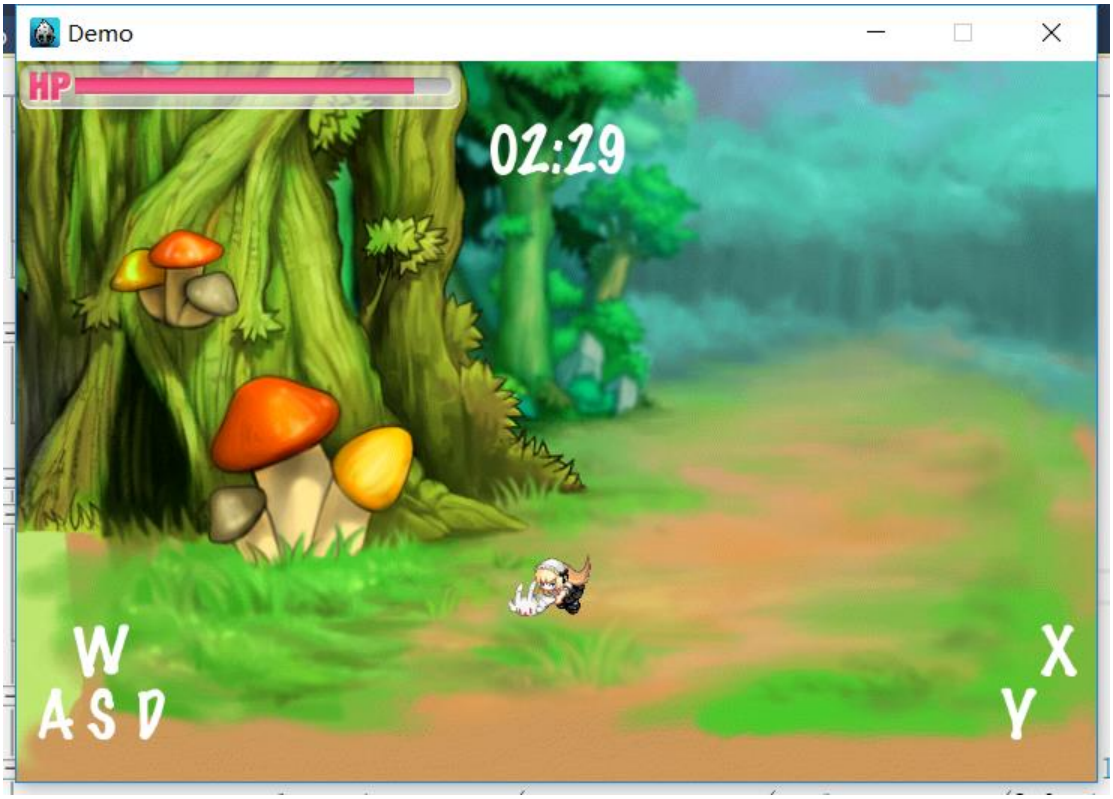


3. 人物动作

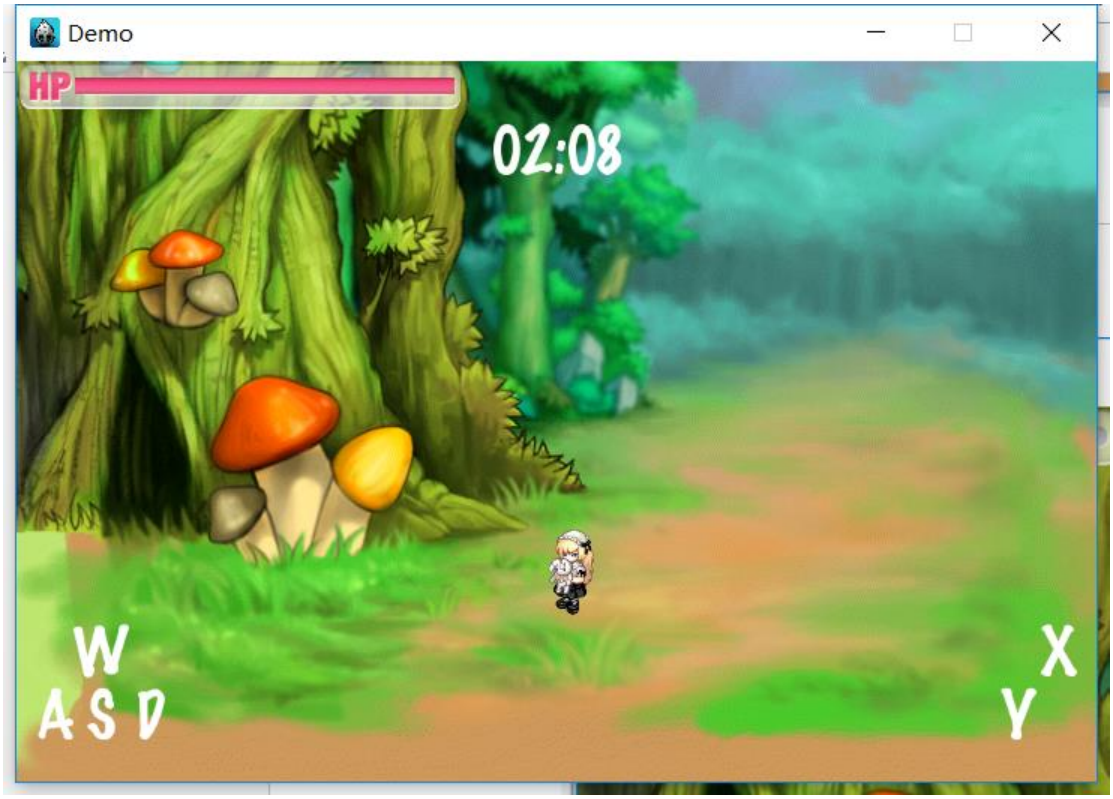
点击 X，执行死亡动作



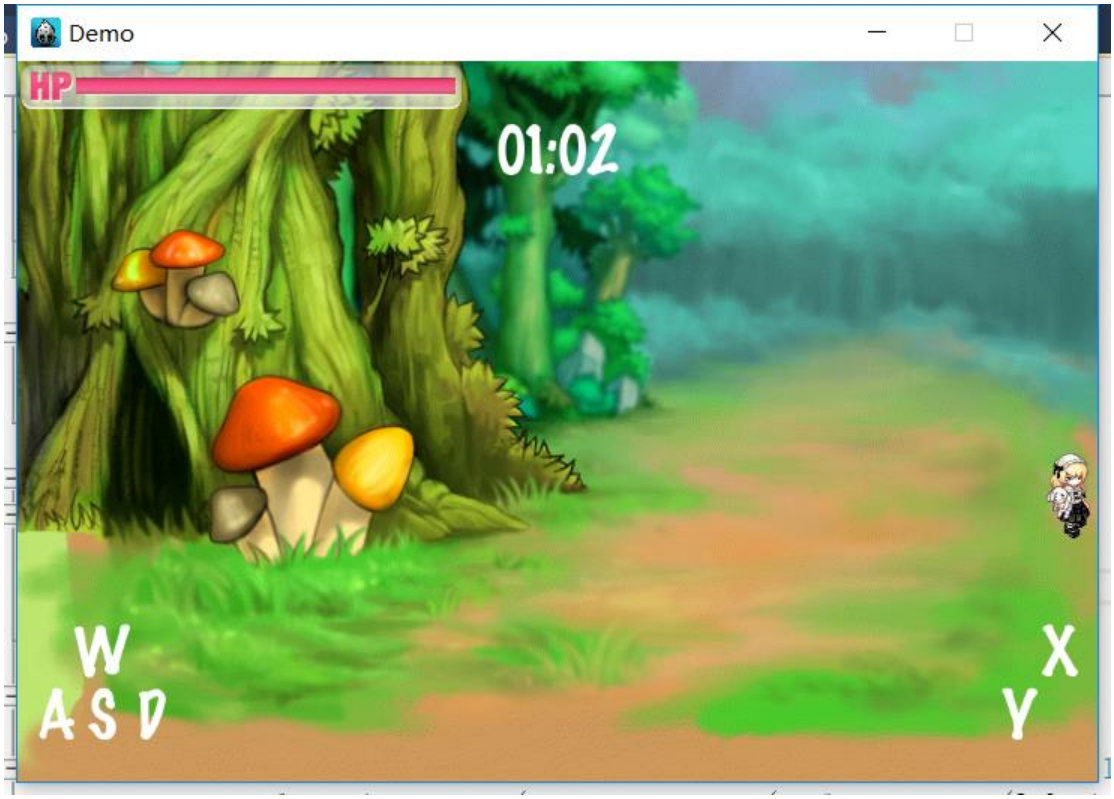
点击 Y，执行攻击动作



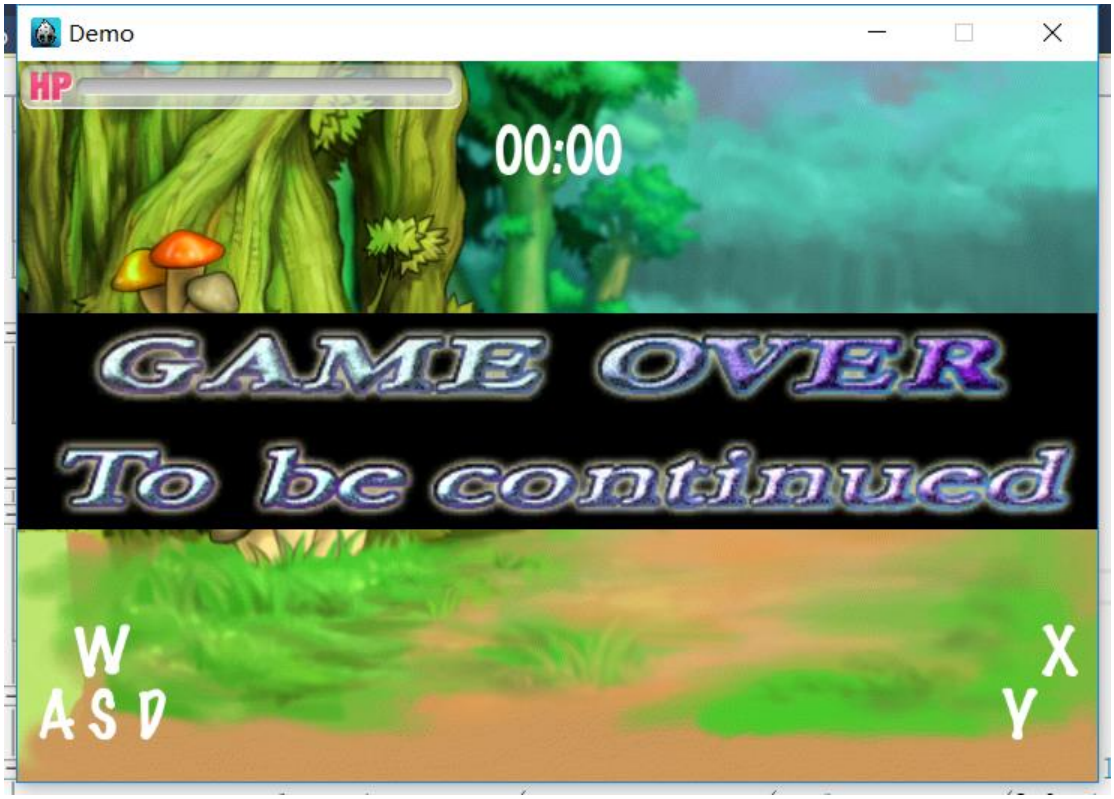
执行后血量增加



4. 人物不能离开可视界面



5. 时间到游戏结束



四．实验过程遇到的问题

1. 报错：无法解析的外部符号 public: static class cocos2d::Scene * __cdecl

MenuSence。

根据博客中的方法：在你自己的头文件中加入#include “extensions/cocos-ext.h"，使用命名空间 USING_NS_CC_EXT;，选中工程右键“属性” ->"配置属性 “->"c/c++"->"常规” ->"附加包含目录 “中添加$(EngineRoot)、$(EngineRoot)cocos\editor-support、$(EngineRoot)cocos解决。

2. time 的值需要赋值为 string 类型，不能直接赋 dtime 的值。
3. 为了保证每次动作执行完之后才能执行下一个动作 ,添加一个精灵动作停止的判断条件：

```
void HelloWorld::stopAc() {
    actionOn = true;
}
```

每个动作执行之前判断人物是否正在执行动作 ,判断标志 true 表示这个动作执行结束下一个动作可以执行。

五．思考与总结

1. 真看似简单的一个小游戏，实现起来还是遇到很多问题，需要通过网络搜索，经过这次作业，对帧动画和调度器有了更深的了解。