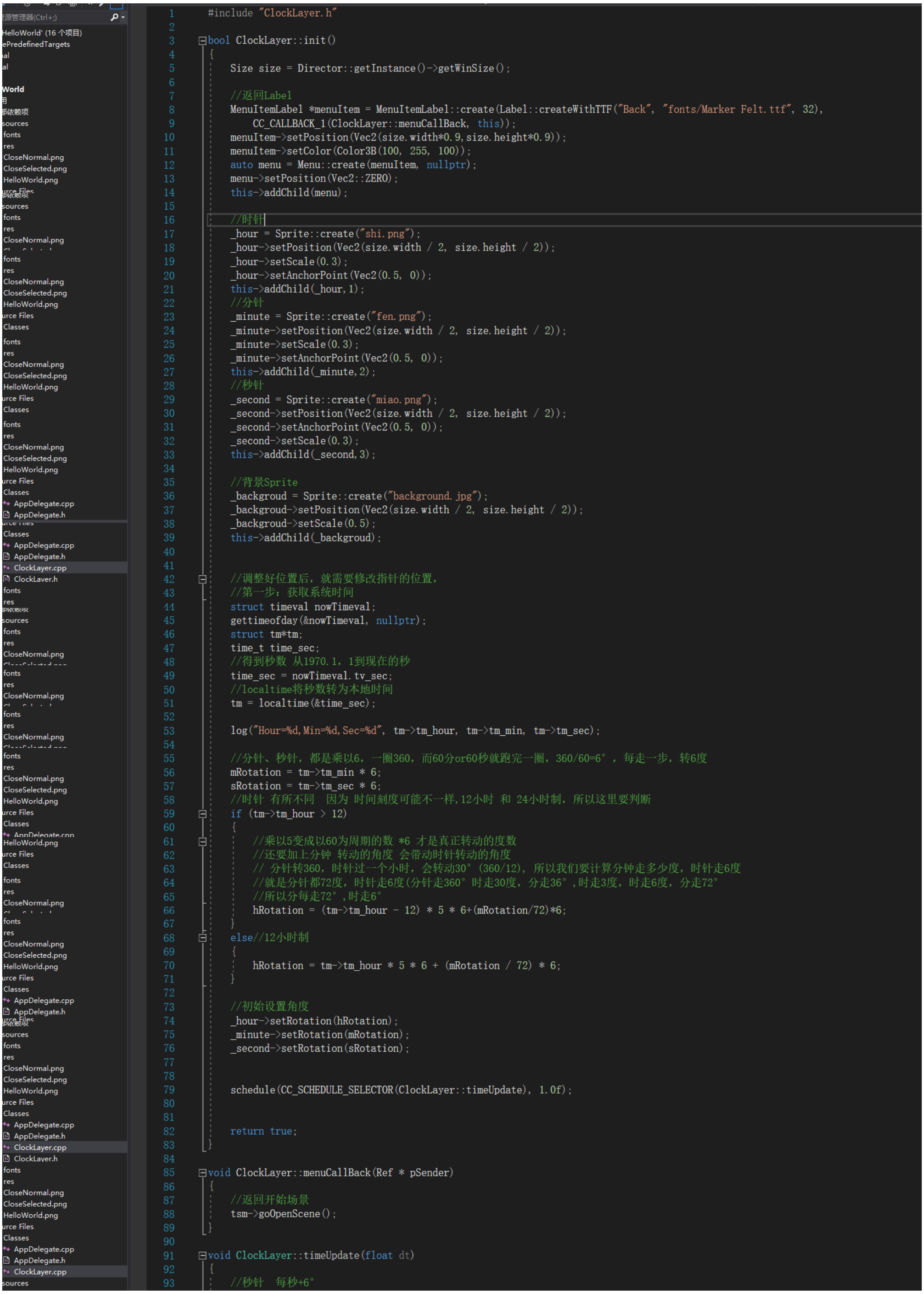
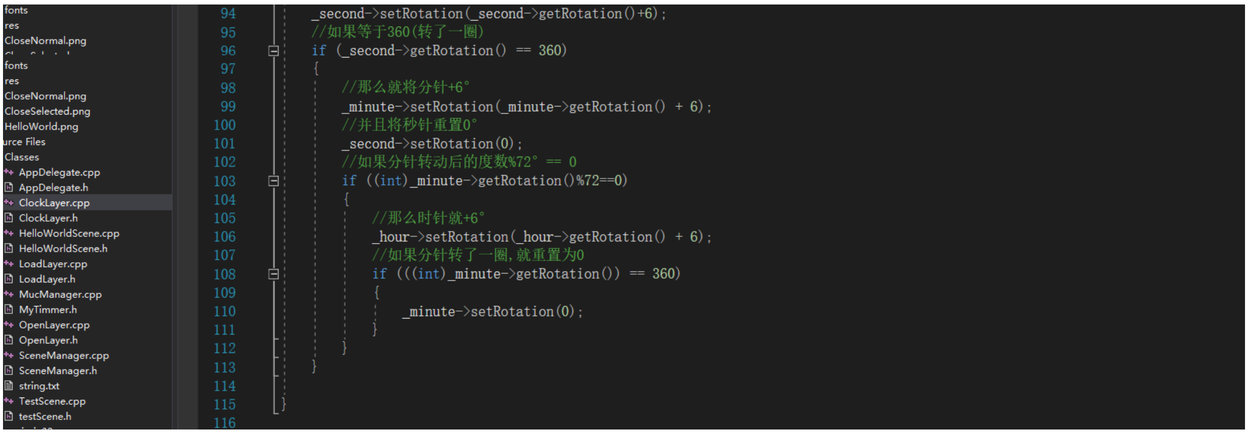
完整版

clockLayer.cpp

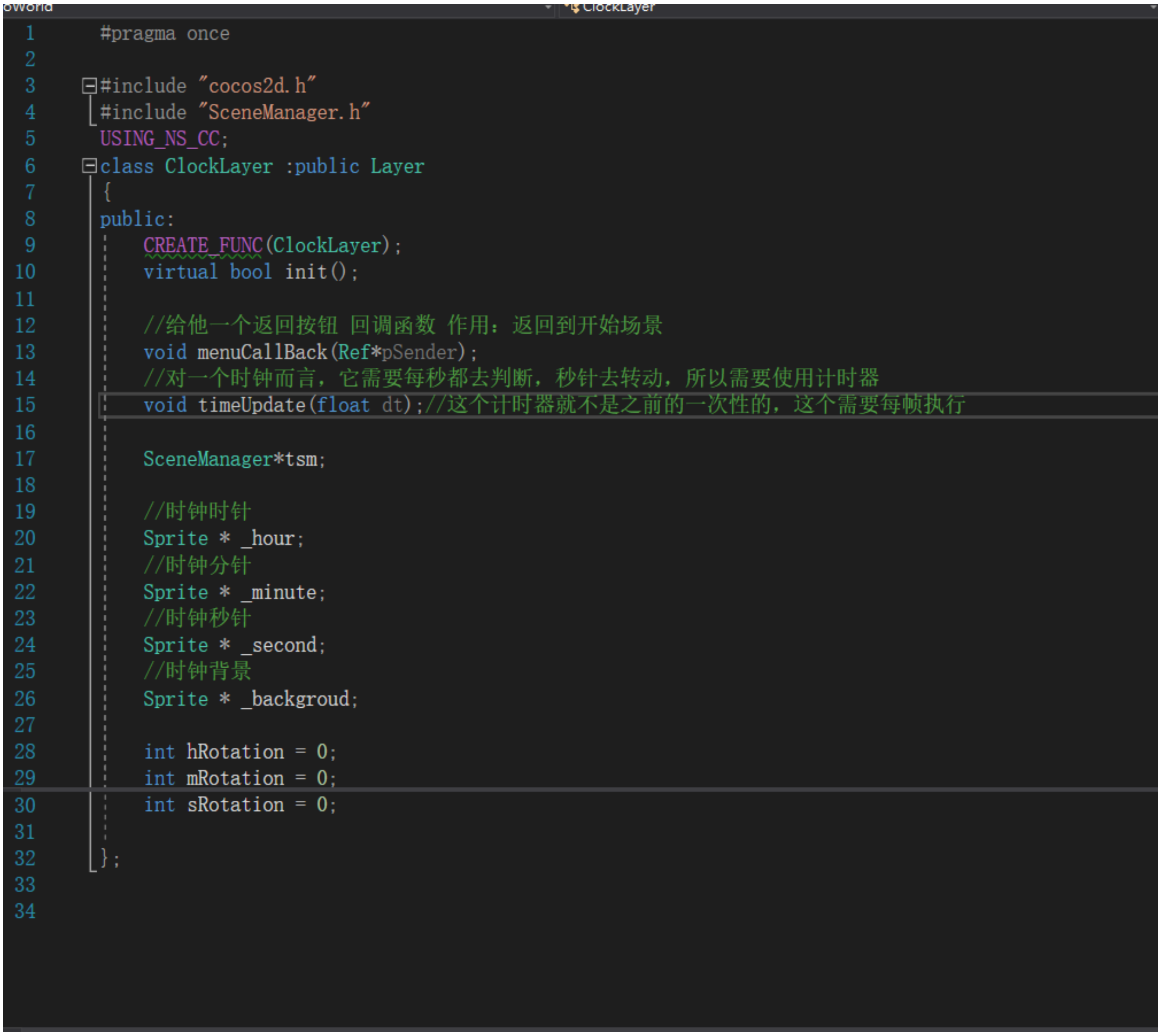


```cpp
#include "ClockLayer.h"

bool ClockLayer::init()
{
    Size size = Director::getInstance()->getWinSize();

    //返回Label
    MenuItemLabel *menuItem = MenuItemLabel::create(Label::createWithTTF("Back", "fonts/Marker Felt.ttf", 32),
        CC_CALLBACK_1(ClockLayer::menuCallBack, this));
    menuItem->setPosition(Vec2(size.width*0.9,size.height*0.9));
    menuItem->setColor(Color3B(100, 255, 100));
    auto menu = Menu::create(menuItem, nullptr);
    menu->setPosition(Vec2::ZERO);
    this->addChild(menu);

    //时针
    _hour = Sprite::create("shi.png");
    _hour->setPosition(Vec2(size.width / 2, size.height / 2));
    _hour->setScale(0.3);
    _hour->setAnchorPoint(Vec2(0.5, 0));
    this->addChild(_hour,1);
    //分针
    _minute = Sprite::create("fen.png");
    _minute->setPosition(Vec2(size.width / 2, size.height / 2));
    _minute->setScale(0.3);
    _minute->setAnchorPoint(Vec2(0.5, 0));
    this->addChild(_minute,2);
    //秒针
    _second = Sprite::create("miao.png");
    _second->setPosition(Vec2(size.width / 2, size.height / 2));
    _second->setAnchorPoint(Vec2(0.5, 0));
    _second->setScale(0.3);
    this->addChild(_second,3);

    //背景Sprite
    _backgroud = Sprite::create("background.jpg");
    _backgroud->setPosition(Vec2(size.width / 2, size.height / 2));
    _backgroud->setScale(0.5);
    this->addChild(_backgroud);


    //调整好位置后，就需要修改指针的位置，
    //第一步：获取系统时间
    struct timeval nowTimeval;
    gettimeofday(&nowTimeval, nullptr);
    struct tm*tm;
    time_t time_sec;
    //得到秒数 从1970.1，1到现在的秒
    time_sec = nowTimeval.tv_sec;
    //localtime将秒数转为本地时间
    tm = localtime(&time_sec);

    log("Hour=%d,Min=%d,Sec=%d", tm->tm_hour, tm->tm_min, tm->tm_sec);

    //分针、秒针，都是乘以6，一圈360，而60分or60秒就跑完一圈，360/60=6°，每走一步，转6度
    mRotation = tm->tm_min * 6;
    sRotation = tm->tm_sec * 6;
    //时针 有所不同  因为 时间刻度可能不一样,12小时 和 24小时制，所以这里要判断
    if (tm->tm_hour > 12)
    {
        //乘以5变成以60为周期的数 *6 才是真正转动的度数
        //还要加上分钟 转动的角度 会带动时针转动的角度
        // 分针转360，时针过一个小时，会转动30°(360/12)，所以我们要计算分钟走多少度，时针走6度
        //就是分针都72度，时针走6度(分针走360°时走30度，分走36°,时走3度，时走6度，分走72°
        //所以分每走72°,时走6°
        hRotation = (tm->tm_hour - 12) * 5 * 6+(mRotation/72)*6;
    }
    else//12小时制
    {
        hRotation = tm->tm_hour * 5 * 6 + (mRotation / 72) * 6;
    }

    //初始设置角度
    _hour->setRotation(hRotation);
    _minute->setRotation(mRotation);
    _second->setRotation(sRotation);


    schedule(CC_SCHEDULE_SELECTOR(ClockLayer::timeUpdate), 1.0f);


    return true;
}

void ClockLayer::menuCallBack(Ref * pSender)
{
    //返回开始场景
    tsm->goOpenScene();
}

void ClockLayer::timeUpdate(float dt)
{
    //秒针  每秒+6°
```



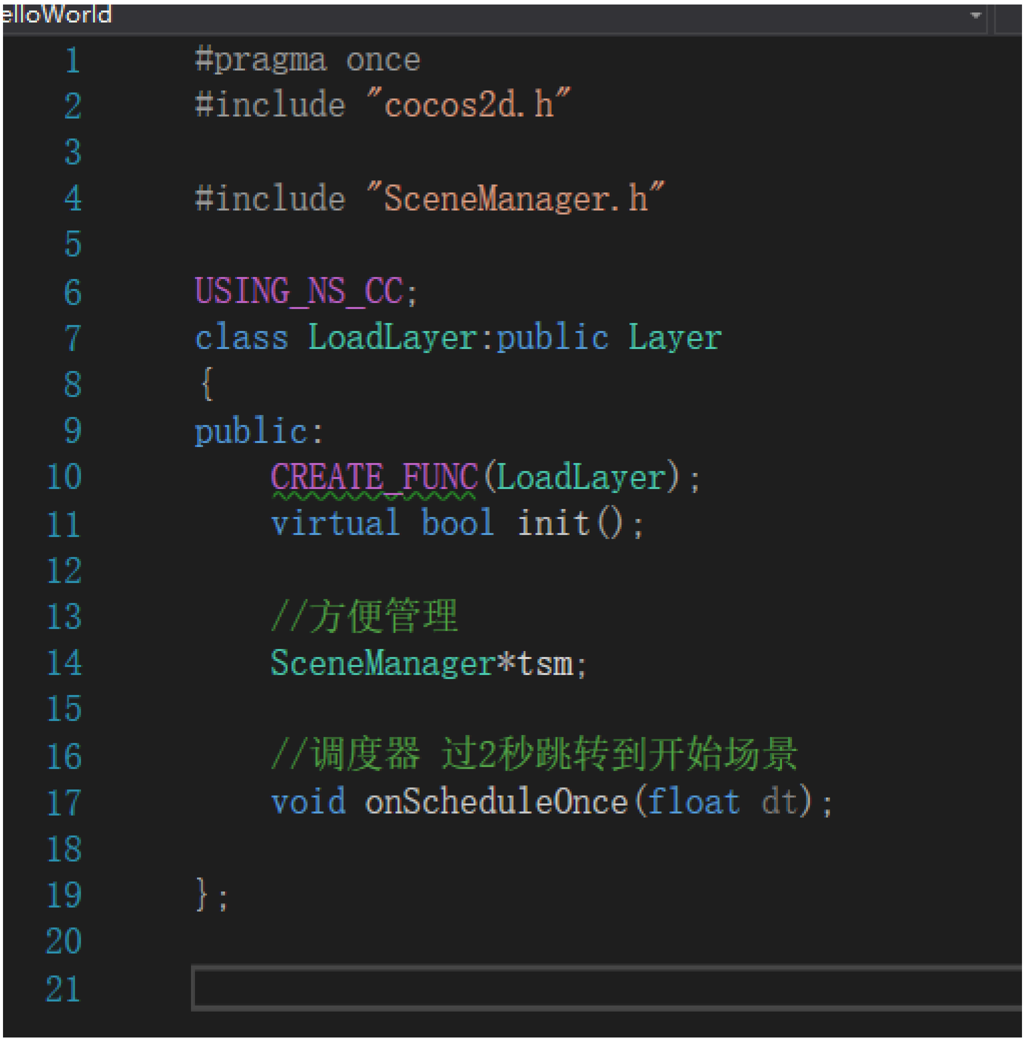
```cpp
    _second->setRotation(_second->getRotation()+6);
    //如果等于360(转了一圈)
    if (_second->getRotation() == 360)
    {
        //那么就将分针+6°
        _minute->setRotation(_minute->getRotation() + 6);
        //并且将秒针重置0°
        _second->setRotation(0);
        //如果分针转动后的度数%72° == 0
        if ((int)_minute->getRotation()%72==0)
        {
            //那么时针就+6°
            _hour->setRotation(_hour->getRotation() + 6);
            //如果分针转了一圈,就重置为0
            if (((int)_minute->getRotation()) == 360)
            {
                _minute->setRotation(0);
            }
        }
    }

}
```

clockLayer.h:



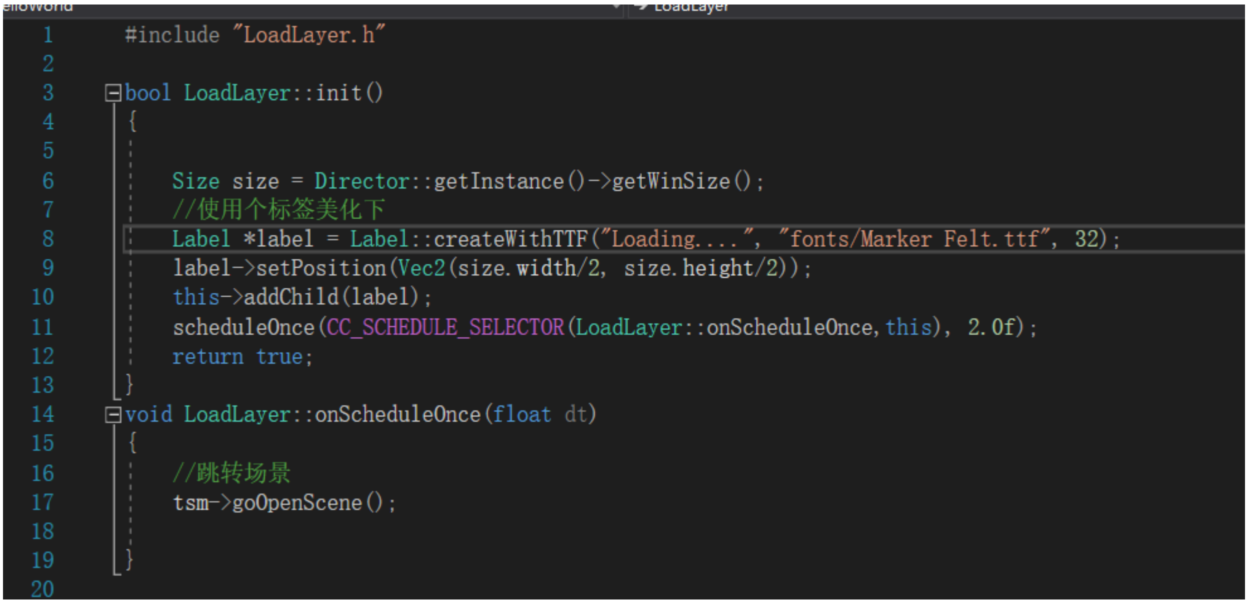
```cpp
#pragma once

#include "cocos2d.h"
#include "SceneManager.h"
USING_NS_CC;
class ClockLayer :public Layer
{
public:
    CREATE_FUNC(ClockLayer);
    virtual bool init();

    //给他一个返回按钮 回调函数 作用：返回到开始场景
    void menuCallBack(Ref*pSender);
    //对一个时钟而言，它需要每秒都去判断，秒针去转动，所以需要使用计时器
    void timeUpdate(float dt);//这个计时器就不是之前的一次性的，这个需要每帧执行

    SceneManager*tsm;

    //时钟时针
    Sprite * _hour;
    //时钟分针
    Sprite * _minute;
    //时钟秒针
    Sprite * _second;
    //时钟背景
    Sprite * _backgroud;

    int hRotation = 0;
    int mRotation = 0;
    int sRotation = 0;

};


```

LoadLayer.h



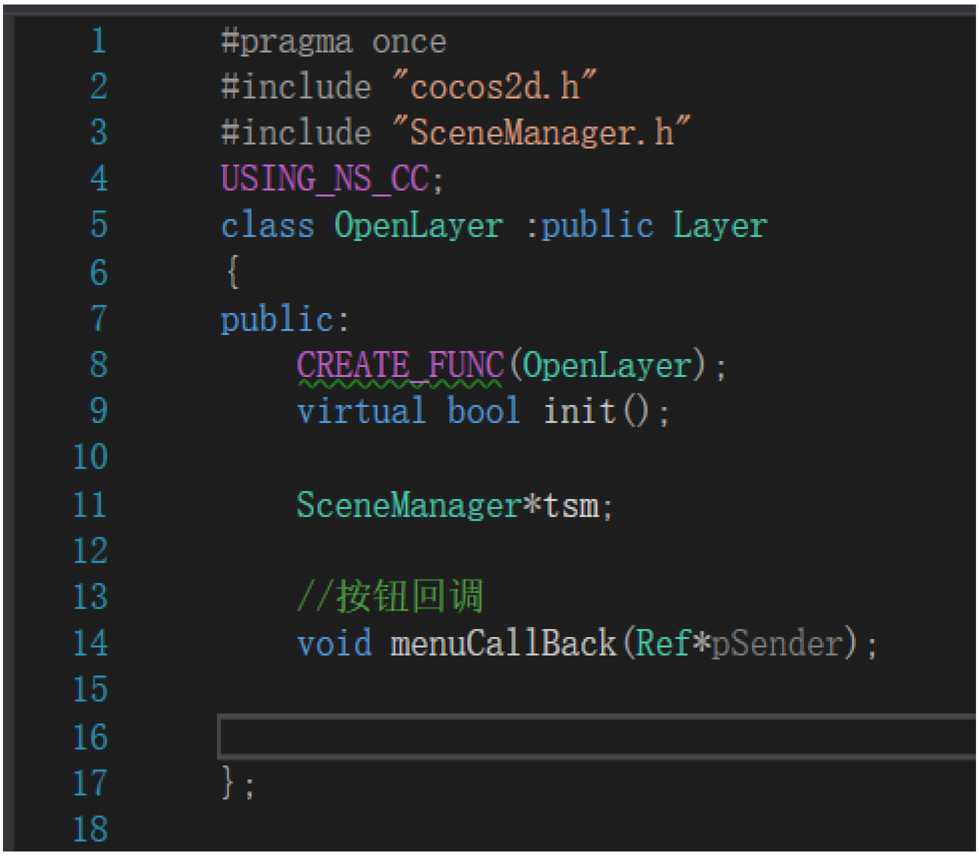
```
#pragma once
#include "cocos2d.h"

#include "SceneManager.h"

USING_NS_CC;
class LoadLayer:public Layer
{
public:
    CREATE_FUNC(LoadLayer);
    virtual bool init();

    //方便管理
    SceneManager*tsm;

    //调度器 过2秒跳转到开始场景
    void onScheduleOnce(float dt);

};
```

LoadLayer.cpp



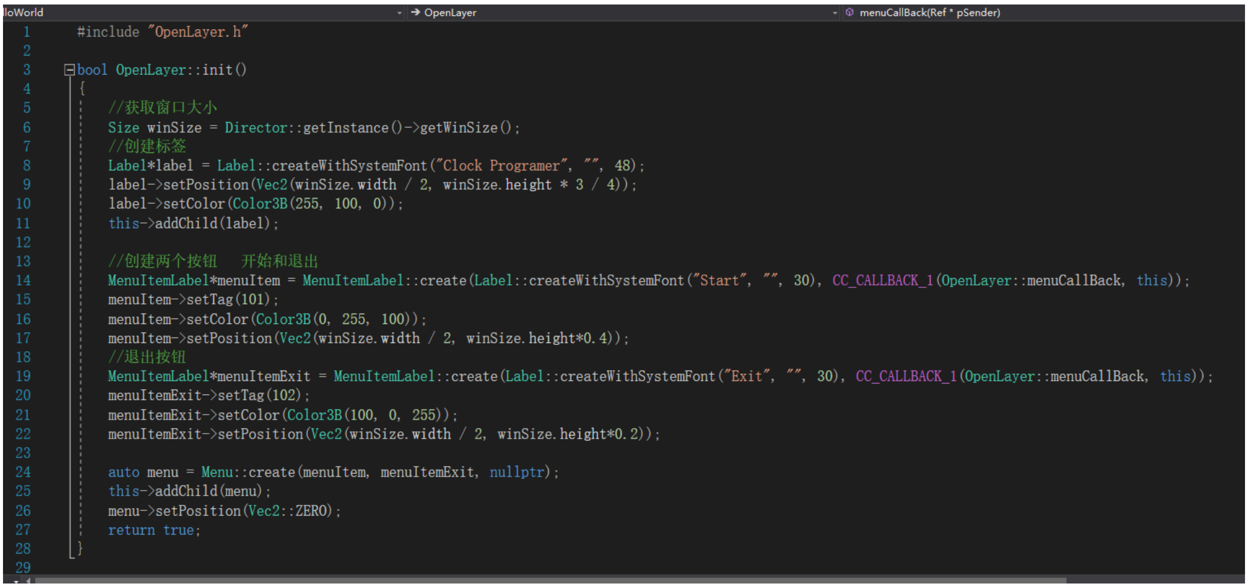
```
#include "LoadLayer.h"

bool LoadLayer::init()
{

    Size size = Director::getInstance()->getWinSize();
    //使用个标签美化下
    Label *label = Label::createWithTTF("Loading....", "fonts/Marker Felt.ttf", 32);
    label->setPosition(Vec2(size.width/2, size.height/2));
    this->addChild(label);
    scheduleOnce(CC_SCHEDULE_SELECTOR(LoadLayer::onScheduleOnce,this), 2.0f);
    return true;
}
void LoadLayer::onScheduleOnce(float dt)
{
    //跳转场景
    tsm->goOpenScene();

}
```

OpenLayer.h



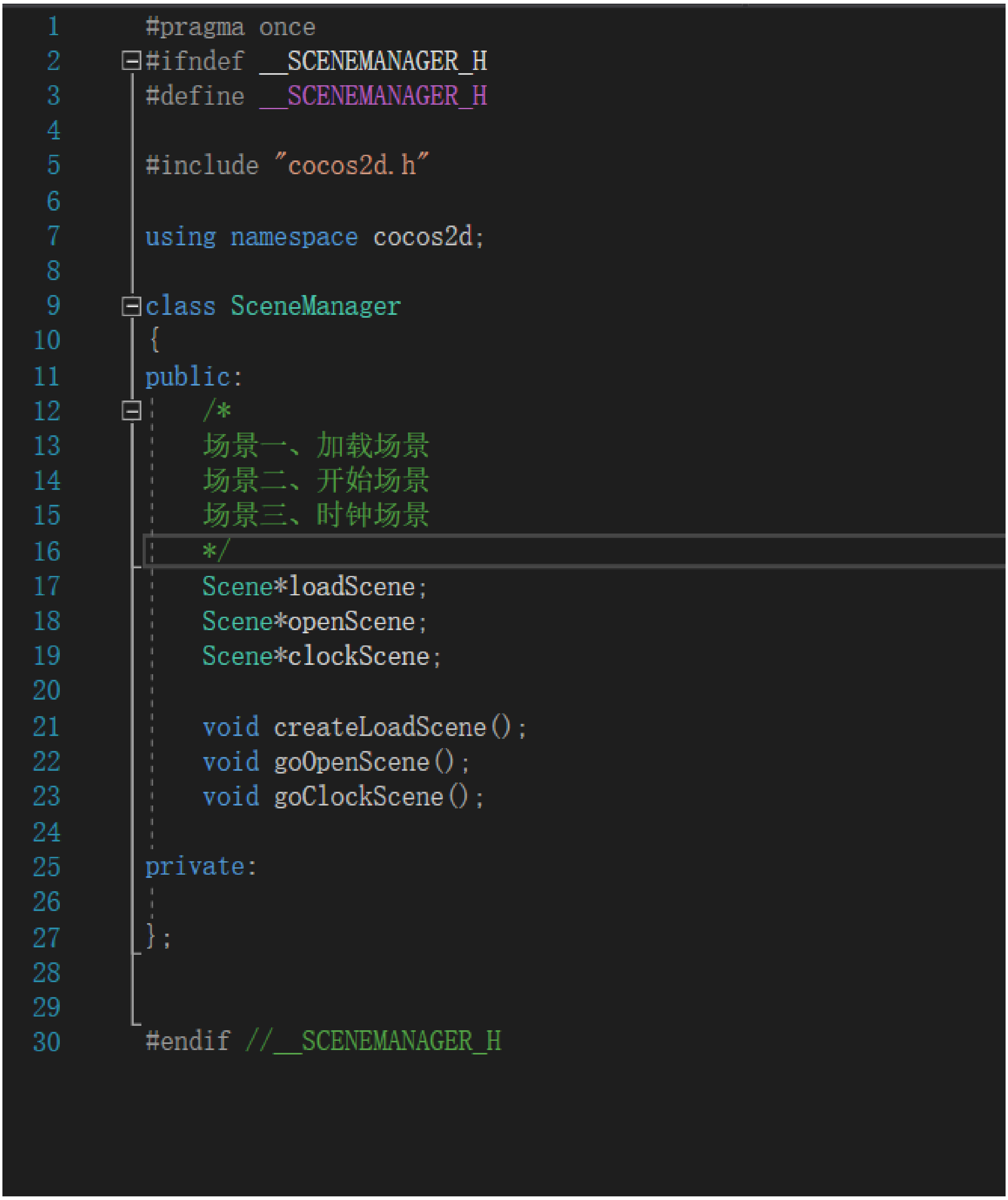
```cpp
#pragma once
#include "cocos2d.h"
#include "SceneManager.h"
USING_NS_CC;
class OpenLayer :public Layer
{
public:
    CREATE_FUNC(OpenLayer);
    virtual bool init();

    SceneManager*tsm;

    //按钮回调
    void menuCallBack(Ref*pSender);


};

```

OpenLayer.cpp



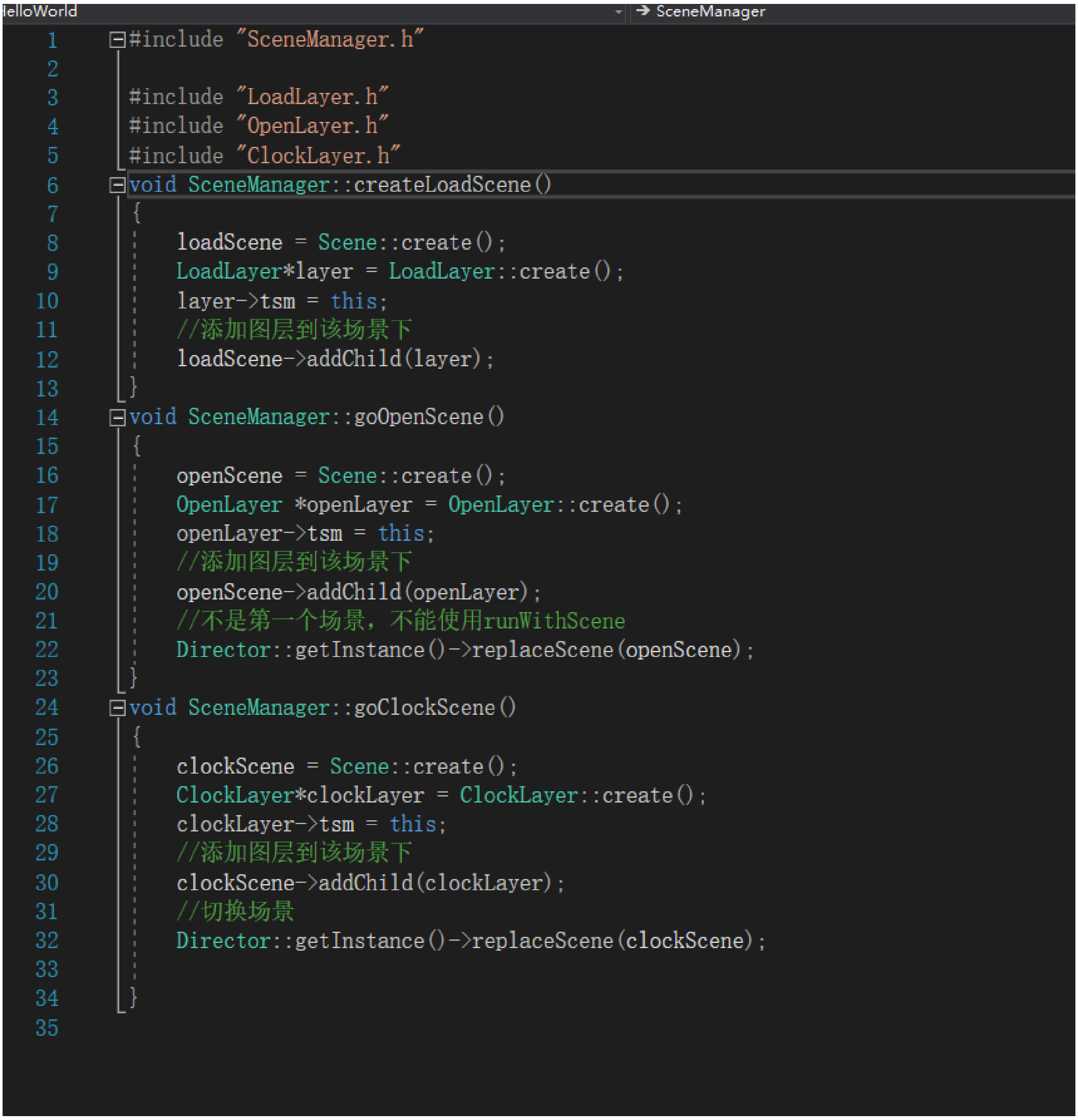
```cpp
#include "OpenLayer.h"

bool OpenLayer::init()
{
    //获取窗口大小
    Size winSize = Director::getInstance()->getWinSize();
    //创建标签
    Label*label = Label::createWithSystemFont("Clock Programer", "", 48);
    label->setPosition(Vec2(winSize.width / 2, winSize.height * 3 / 4));
    label->setColor(Color3B(255, 100, 0));
    this->addChild(label);

    //创建两个按钮   开始和退出
    MenuItemLabel*menuItem = MenuItemLabel::create(Label::createWithSystemFont("Start", "", 30), CC_CALLBACK_1(OpenLayer::menuCallBack, this));
    menuItem->setTag(101);
    menuItem->setColor(Color3B(0, 255, 100));
    menuItem->setPosition(Vec2(winSize.width / 2, winSize.height*0.4));
    //退出按钮
    MenuItemLabel*menuItemExit = MenuItemLabel::create(Label::createWithSystemFont("Exit", "", 30), CC_CALLBACK_1(OpenLayer::menuCallBack, this));
    menuItemExit->setTag(102);
    menuItemExit->setColor(Color3B(100, 0, 255));
    menuItemExit->setPosition(Vec2(winSize.width / 2, winSize.height*0.2));

    auto menu = Menu::create(menuItem, menuItemExit, nullptr);
    this->addChild(menu);
    menu->setPosition(Vec2::ZERO);
    return true;
}

```

SceneManager.h



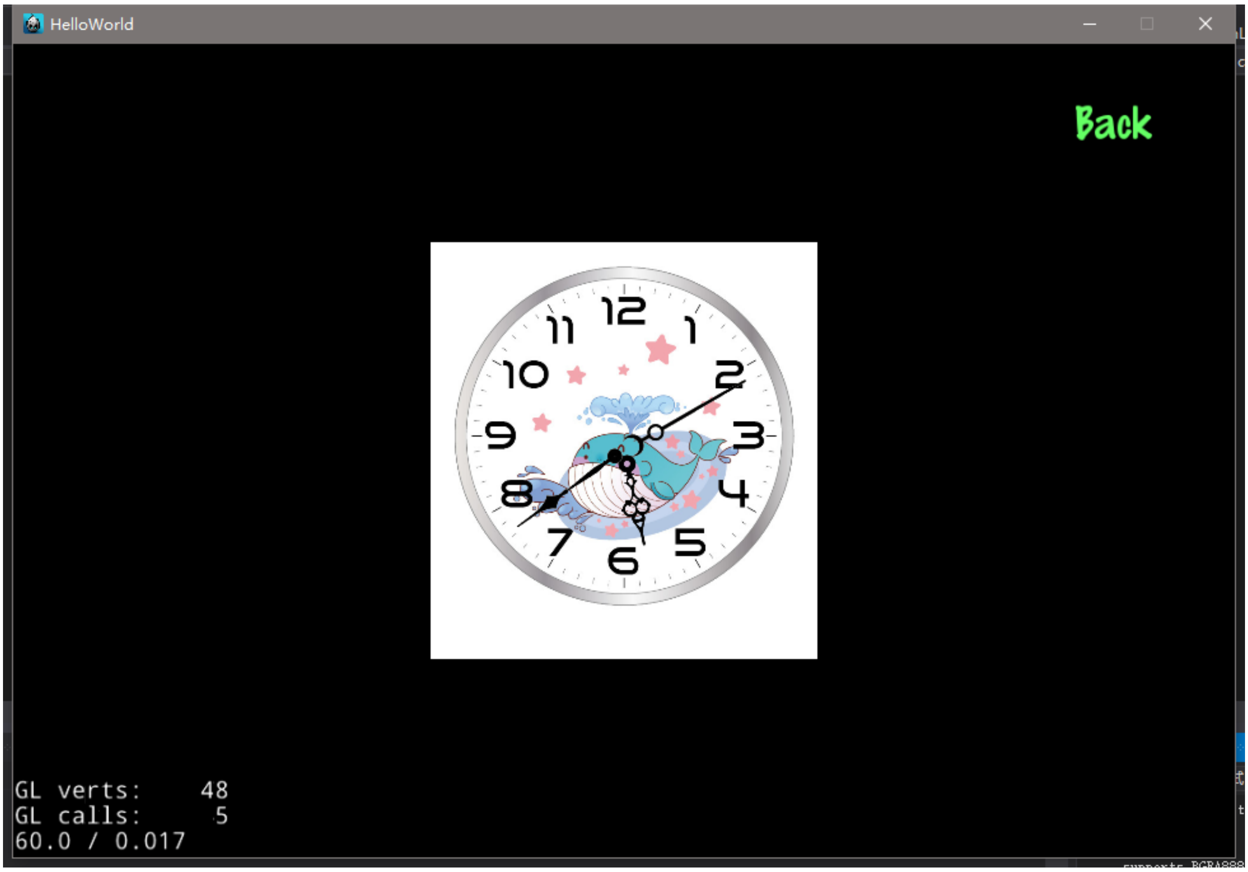
```
#pragma once
#ifndef __SCENEMANAGER_H
#define __SCENEMANAGER_H

#include "cocos2d.h"

using namespace cocos2d;

class SceneManager
{
public:
    /*
    场景一、加载场景
    场景二、开始场景
    场景三、时钟场景
    */
    Scene*loadScene;
    Scene*openScene;
    Scene*clockScene;

    void createLoadScene();
    void goOpenScene();
    void goClockScene();

private:

};


#endif //__SCENEMANAGER_H
```

SceneManager.cpp



```cpp
#include "SceneManager.h"

#include "LoadLayer.h"
#include "OpenLayer.h"
#include "ClockLayer.h"
void SceneManager::createLoadScene()
{
    loadScene = Scene::create();
    LoadLayer*layer = LoadLayer::create();
    layer->tsm = this;
    //添加图层到该场景下
    loadScene->addChild(layer);
}
void SceneManager::goOpenScene()
{
    openScene = Scene::create();
    OpenLayer *openLayer = OpenLayer::create();
    openLayer->tsm = this;
    //添加图层到该场景下
    openScene->addChild(openLayer);
    //不是第一个场景，不能使用runWithScene
    Director::getInstance()->replaceScene(openScene);
}
void SceneManager::goClockScene()
{
    clockScene = Scene::create();
    ClockLayer*clockLayer = ClockLayer::create();
    clockLayer->tsm = this;
    //添加图层到该场景下
    clockScene->addChild(clockLayer);
    //切换场景
    Director::getInstance()->replaceScene(clockScene);

}

```

output:



```
GL verts:     48
GL calls:      5
60.0 / 0.017
```