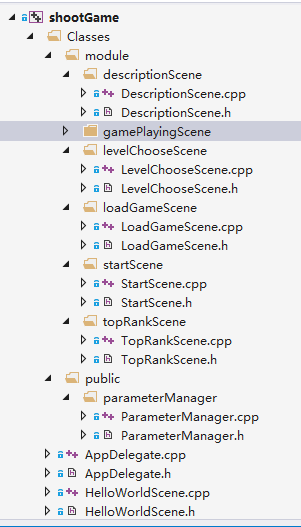
框架说明文档

一、框架介绍：

搭了一个不三不四的框架 ：） 为了方便分工，分了一些模块：



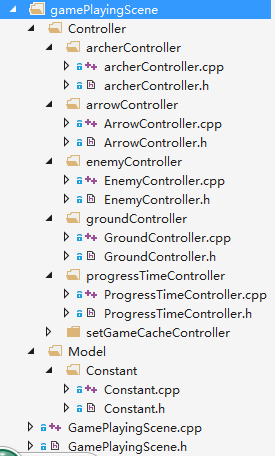
1.其中HelloWorldScene是入口，AppDelegate你懂的（其实不知道怎么描述）。

2.Public文件夹里面存放全局的公共变量或函数，例如ParameterManager就是管理公共的参数。

3.module文件夹里放置多个模块，

HelloWorldScene界面有4个按钮，分别进入DescriptionScene游戏说明模块，StartScene开始游戏模块，LoadGameScene读档模块，TopRankScene高分排名模块。进入StartScene模块里点击“Skip”按钮就进入LevelChooseScene选关模块，选关后就进入GamePlayingScene战斗场景模块。以上每个模块都是继承于Scene的类。

4.GamePlayingScene战斗场景模块里面又有一些模块：



这里也为了分(qiang)工(xing)方(zhuang)便(B)，模仿了一些MVC。

（1）GamePlayingScene可以理解为一个总的Controller控制器，因为感觉在cocos2d的模式下要把View分割出来似乎不容易。

（2）之后为每个角色设置一个Controller，archer（弓箭手）、Ground（地板）、ProgressTime（进度条，就是显示蓄力程度的）都是单例；arrow（射出来的箭）、enermy（敌人）不是单例。

（3）setGameCache用来预加载图片、帧、动画进cache。

5.Resources文件夹：

（1）image/map文件夹：放地图tmx文件

（2）image/texture文件夹：放纹理图（合并的大纹理图最好生成plist文件）

（3）image/particle文件夹：放粒子效果的文件（图与plist）

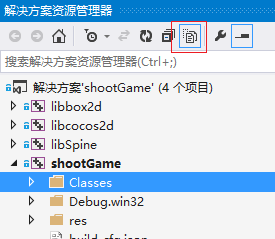
（4）其他文件正常放。

二、框架使用

1.win32中导入文件夹：

（1）在新建的项目中打开proj.win32目录下的sln文件。在VS中把原有的src目录中的文件移除。

（2）把zip解压后的Classes文件夹放进proj.win32下，VS中解决方案资源管理器中点击“显示所有文件”：

如图红框。

（3）之后对Classes文件夹右键->“包括在项目中”。导入完成。

（4）Resources文件夹可以直接替换原来的。

2.类的写法：

（1）.h文件用来声明，注意继承于什么类，是不是私有变量，是不是静态的函数or变量。头文件一般加上cocos2d.h和USING\_NS\_CC就够了。

（2）.cpp文件用来实现变量与函数，每一个的声明都要有对应的实现。Static，public等关键字不用写。但变量和函数名前一定要加“类名：：”

（3）继承类的写法：

一般的写法大家都懂，但是创建类的时候不能只写create（如Scene::create()）这样返回值是Scene，不是你继承的类。要再在.h文件中加上宏（CREATE\_FUNC(yourClassName)），这样返回值就是你的类。另外这样调用create()的时候会调用init()，所以我们只需要重写init函数就可以往我们这个类中addChild了。（可参照DescriptionScene）

但是特殊情况下这种方法不能用。例如GamePlayingScene中需要createWithPhysics（带屋里引擎的），没有对应的宏可以使用（当然你可以自己建）。这时候我们就要调用底层代码，点击createWithPhysics函数，按F12，跳到函数定义。可见这个函数关键只有三行代码：new、initWithPhysics、autorelease。因此我们只需要搬过来，并把new Scene()改为new自己的类就可以了。（可参照GamePlayingScene）。你也可以不用方法一直接用这种，B格瞬间提高。

3.GamePlayingScene的介绍：

（1）所有参数、路径、名称都保存在Constant类中，方便修改。例如archer图片的位置是"image/texture/archer.png"，我们每次只要修改在constant中的值就可以了，其他文件中调用Constant::getArcherPath()来获取。不然当文件一多，就很难维护。

（2）



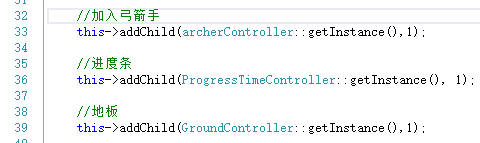
用来设置重力，重力的设置在ParameterManager中；如果还需要设置其他初始参数请写在这行下面，并把具体参数值保存在ParameterManager或Constant中（ParameterManager用来保存可能要在其他Scene中用到的参数，Constant中只保存只在GamePlayingScene中用到的参数）。

（3）



图片帧、动画的预加载写在这里

（4）



这三个是单例，getInstance会先检测单例是否被创建，没有就创建并init，有的话就直接返回该单例。

如果需要添加四周的墙壁，仿照GroundController就可以了，或者把GroundController设为不是单例。

（5）



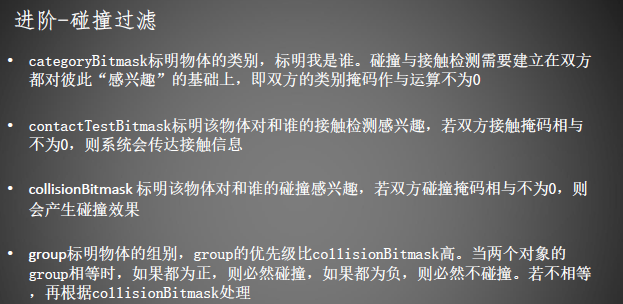
暂时设置是用schedule产生敌人，若需要敌人一开就在固定位置出现，就直接createAnEnemy：



就可以了。

（6）碰撞检测：

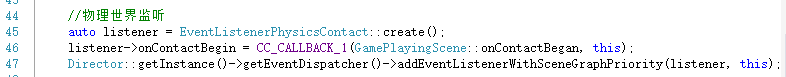
首先每个物体都要添加PhysicsBody才会进入物理世界。之后要设置



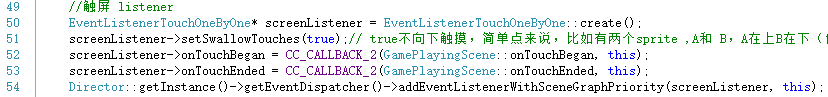
这些来规定要检查哪些碰撞。

暂时设置的是直接测箭与敌人、敌人与射击手、箭与地板的碰撞，所以设计的掩码是射击手（0x0000000F）,地板（0x00000FF0）,弓箭（0x0000F0F0）,敌人（0x0000F00F）。

碰撞监听：



（7）触屏监听：



其中onTouchBagan中，使用schedule不断地刷新ProgressBar的值，直到onTouchEnded，才进行unschedule，并把见按角度（放鼠标位置的角度）、力度（按鼠标时间的长短）射出。

4.GamePlayingScene的使用与修改：

记住保持模块化的思想就行了。就是在GameplayingScene这几个模块中，负责一个模块的同学不用改动其他模块（可以get，但不能set），ProgressTimeController除外（因为不参与碰撞检测）。这样方便管理代码，毕竟我们是这么多人写一份。

（1）例如要在archer中添加属性，就增加属性变量，并提供get和set接口。

（2）模块之间的参数传递：尽量不要模块之间直接传参。

有先后执行顺序的，先执行的可以先把变量存放在Constant中，后执行的程序再提取变量。

也可以在GamePlayingScene里面设置函数给模块之间传参。可以自己设置自定义监听。

5.代码管理：

使用Github：

https://github.com/mozi030/shootGame

6.分工安排：

负责某一模块的同学，需要负责包括从设计、选图，到动作效果、音效等细节以及与其他模块交互等关于模块的圈套工作。当然设计、选图最好与本小组的总设计师罗斯与美工担当可冰配合。模块间的交互可找莫子商量如何适当地交互

（1）罗斯：产品总设计，特别是战斗界面的设计(从需要哪些角色、道具，属性值，到每一个角色道具的像素大小)

（2）冬贤：选关模块、不同关卡的战斗场景的设计（与罗斯一同），射击手、弓箭。

（3）可冰：美工、开始界面模块、游戏说明模块、背景介绍模块

（4）刘超：数据库、存档模块（包括关卡间的存档）、高分排名模块、战斗场景中的敌人模块

（5）高佬：制作地图、商店、道具界面

（6）莫子：协助模块间的交互，handle GamePlayingScene模块与公共模块，协助个人完成任务。