修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 版本 | 说明 | 作者 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 摘要

主要讲解在手机游戏的架构设计中的对象管理的设计，着重于架构，算法与数据结构。并使用当下流行的引擎Unity3D为和其推荐的编程语言C#，辅助讲解。其设计思想可以应用于其他引擎和语言。

# 概述

先用一张简要的模块图来帮助大家了解游戏架构设计，了解对象管理在架构中所处的位置和他的作用是。实际的模块图会比本图的内容详细，并且对模块的分割有其他的方式，这张图只是为了帮助大家了解对象管理，做了很多精简。

要想讲清楚对象管理，还得从下层的资源管理说起。

## 资源管理

资源管理主要有两个重要的功能：资源加载和资源释放。

资源加载指的是把需要的资源（如prefab，材质，贴图等，从磁盘加载到内存中），供上层模块调用。这些资源通常以某种形式进行打包。Unity3D通常是用AssetBundle打包。举例来说，想在游戏中显示一个角色需要的资源就是穿过多个模块后，最后达到资源管理模块，从这里获得。

资源释放是与其对应的，就是把不用的资源从内存中卸载。那问题来了，什么时候资源什么时候不用？通常有两种情形。场景切换和角色不在屏幕时。难道此时我们就应该把资源从内存中释放？这样做也不是不可以，但这样就意味着做出来的游戏是低效的。脑补下策划，制作人的表情:) 道理很简单，角色是非常常见的，如果每次不用了就从内存释放，那就意味着每次用的时候都要从磁盘加载。这样就引出了对象管理这个概念。

## 对象管理

对象管理是资源管理和上层模块的一个中间层，缓存。他主要为那些暂时不用的资源提供内存缓存作用。角色首次需要的时候确实从资源管理从获取，并且缓存在对象管理中，暂时不用时不从资源管理器释放，而是由对象管理来接管。当下次再用的时候，就从对象管理来取角色。都是在内存中，速度自然比从资源管理中快。

# 对象管理模块设计

既然作为缓存，那当然对速度有要求。比如场景中需要10个相同的NPC，那么我们缓存一个NPC显然是不够的，而是需要多个。不然对象管理的世界意义就会大打折扣。多个重复的对象（NPC），这种情形下，“对象池”出场了。对象池就是用于存储重复对象的。

NPC可能有多种，“对象池”就会有多个。既然是多个对象池，就需一个“对象池管理器”来管理这些“对象池”。比如NPC分为NPC1....NPCn，那么就会有n个对象池。我们不太可能遍列这些对象池。而是需要提供O(1)的查找效率，找到对应的对象池。在C#里采用Dictionary这个数据结构是最合适的了。他的key就是“对象池”的名字（或者是类型），value就是对象池的指针。而持有这个Dictionary的就是对象池管理器。

对象从哪个“对象池”取的问题貌似就解决了。那找到了对应的“对象池”，那怎么以O(1)的效率从“对象池”中找到对象呢？我们还是采用Dictionary这个数据结构。因为在客户端显示的NPC都是在服务器中有唯一编号的，EntityID（事实上Player也是）。他是一个唯一的编号，通常是个int或者无符号的int类型的值。是服务器和客户端双方都认可的，而且双方还约定了类型。也就是还有一个EntityType.这个EntityType就是拿来对应对象池名字的。通过EntityType，找到对应的对象池，然后通过EntityID找到对应对象。

无论是本图和上图都一个叫“缓存”的模块。这究竟是怎么回事？事实上是，整个对象管理除了“对象池管理器”和“对象池”之外，还有“缓存”。"对象池"和“缓存”都是存储重复对象的，但他们两点本质的区别。

第一，“缓存”中的对象是已经被用过的（分配了EntityID）,用于相同对象（EntityID）相同的对象可以重复使用。而“对象池”中的对象可能是被用过的，也可能没被用过。当上层逻辑暂时不用时，是把对象还给“对象池”，而避免下次使用同类型的对象从资管管理器中从磁盘读取。当对象被还给“对象池”时，对象上虽然有EntityID,但他是无效的。举例来说，当一个“坦克1”对象第一次被服务器发出指令出现在屏幕上，做出一个“攻击”动作。那么“坦克1”是从“对象池”取到“缓存”中，然后取出，执行“攻击”。如果服务器又发出指令要求“坦克1”做出一个“受击”动作。这时“坦克1”是直接从“缓存”中取出，做出受击动作。如果“坦克1”过一会儿，移动出屏幕了，这时需要把“坦克1”从“缓存”移除，并还回“对象池”。然后这时服务器要求“坦克2”做出执行“攻击”。这个“坦克2”除了EnityID和之前的“坦克1”不同之外，其他都一样。那么这时从“对象池”中取出的对象，很有可能是之前用过的“坦克1”。没关系，我们把他的EntityID重新赋值即可。“缓存”和“对象池”两者配合，就实现真正的对象重复使用。也可以简单理解，缓存中的对象是游戏中正在活动的对象，有身份证的（EntityID），而对象池中的对象是预先从资源管理中读取的对象，要么没有身份证，要么身份证已经过期。

第二，两者的数据结构不同。“缓存”中的对象是存在以EntityID为key存储的Dictionary中。 而“对象池”不能用Dictionary存储。因为“对象池”种对象必须是当前没有使用的。相当于有个状态。而且这个状态只有两个取值，要么正在被用，要么没有用。那他到底采用什么样的数据结构呢？先买个关子，下篇再讲。下面是对象数据的时序图，比上面的流程图更容易看懂整个过程。重要的是先访问“缓存”再访问“对象池”。

对象管理

缓存池pool

预先分配固定数量的对象（譬如坦克模型）。主要对外两个功能，一是申请对象，从未使用中选择一个，并将其标识为已使用；二是归还对象，将其标识为未使用，可供下次申请使用。可以用QObjPool或者DFPool。都是我自己写的pool实现，申请和归还都是O(1)的效率。目前使用的前者，前者代码实现代码简单些，容易读懂。

PoolManager

存在多个Pool就应该个Manager。这里显然是多个，坦克，士兵。。。。还有各种攻击特效。这里有两个PoolManager。一个是用于角色（坦克，士兵。。。），另一个用于特效。内部都是map。前一个的key是角色类型，另一个的key是unity的GameObject的实例ID.

缓存cache

为实现对对象重复操作提供O(1)的查找效率。比如之前收到服务器消息EntityID=1的坦克对象进行移动操作。过一段时间又收到服务器消息对齐进行攻击操作。这是需要从cache中取出之前的对象。基本是跟实际需求去做一个map的数据结构。在这里key=EntityID。

需要说明的是，特效的cache是做在CharObj上的，作用主要是为了释放机制。

pool中的对象什么时候可以删除?

分两种情况，还和删。

因为不能在runtime中无限制的增加不删除，所以应该把不用的对象及时“还”给pool（比如不在视线范围内的角色对象）。

当场景切换的时候，需要删除pool中的对象。

如果在runtime中无限制的增加不删除obj，结果是pool分配出来的obj越来越多，obj从pool出来是放在cache中。由于pool是队列，而cache是map，因此结果就是obj越来越多，cache越来越大。

做了好还给pool的机制，实际上就控制了obj的数量。因为pool的对象是可以重复使用的。这也是pool存在的价值。

很多介绍QuadTree（四叉树QTree）,这里采用另外一种思路。把屏幕的4个点，投影到4个世界坐标上的点，如果Obj的位置不在这4个点的范围内，就还给pool。

meshbaker 使用经验

（1）保证种子的Animator上的Culling modle 为Always Animate，否则不能执行动画

（2）meshbaker的Renderer要改成 SkinMeshRenderer，否则不能执行动画

（3）官方和网上的代码合不成功怎么办：

官方代码是2的版本，网上很多也是，但也有版本3的。无论如何，按别人的来都不成功，各种报错。我怀疑是我当前使用的版本有问题。不过依然通过以下方式完成了。

首先要使用一次正常的手工合并，保存4样东西。合并后的材质，材质合并结果，隐藏后的种子，整个meshbaker。手工合并后自动生成，后两者手工保存成prefab

其次程序中，通过资源加载的方式读出以上4种，其中前两样，要设置到加载的meshbaker中。然后再按官方代码合并。

（4）合并好后，通过种子进行移动，动画等操作。但如果位置超出合并后的SkinMeshRenderer的bound就看不见了。这时可以人为的在合并前将4个种子放到最大地图的4个角，这样合并出来的bound的extend就是那4个角。

缓存池

pool和creator的关系：pool聚合creator

creator可以抽象出来，for meshbaker也可以for简单的加载资源，实例化n个。

目前的代码所有类型都合并到1个mesh中,最终会超过上限，爆掉，这个需要解决。每次合并单独的生成器？

已解决。每次合并时，创建新的meshbaker对象。

队列的Peek到底是个什么东西。如果队列为空，不能取的。现在加了个总数控制。

生成出来的模型不要挂在一个对象下面，这样会爆掉

Mono的初始化最好写Awake中，之前CharController写在Start中，当创建完一批对象，再次创建就会出现异常的情况，大概是感觉Start里的函数没被执行。没有深入查，简单改成AWake就好了。

需要一个插件来获取机器配置，根据配置来动态调整一些runtime参数。比如对象生成器一次生成的数量之类的。

如何提高meshbaker的速度？提高不了的，可以预先在场景中合并几个，单独管理。这样启动就没了合并时间。但有额外的操作，比如也要控制大小，对象资源改变时，要改场景。具体问题实际中再去思考解决。

角色资源如何释放？除了模型还有被挂在上面的特效，甚至还有头顶血条的UI之类。应该和模型一起释放，先释放特效什么的，最后才释放模型，不然会导致泄露。

角色的Commond类应该被抽象，然后搞个带参数的工厂，这样就可以坦克和士兵共用CharController，并能使用不同的Commond类

有什么办法减少协程开销？坦克开炮前的等待是用协程里等待。

为什么有粒子残影？

修改测试方案，证明了CharObj播放2次特效后，再次取出有特效残影的问题不是特效逻辑的问题。目前只能采用把GmaeObject的active设置成false的方式。