修订历史记录

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

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。