**客户端优化文档**

# 优化目的

目前的WIN32操作系统中，应用程序的用户内存空间最多只能占到2GB，对于XP下客户端的内存占用还可以接受。但在WIN7下，由于好多原本在内核态下的东西被放到用户态下工作，以及D3D内部资源存储的机制，使得程序占用了更多的用户空间。这时WIN7下跑客户端时如果场景复杂度以及同屏玩家数过高时就会引起虚拟内存不足，内存分配失败而导致客户端崩溃。

为了解决崩溃的问题，需要对客户端内存占用进行优化。我们无法更改引擎里的代码，只能从客户端的资源管理逻辑入手。

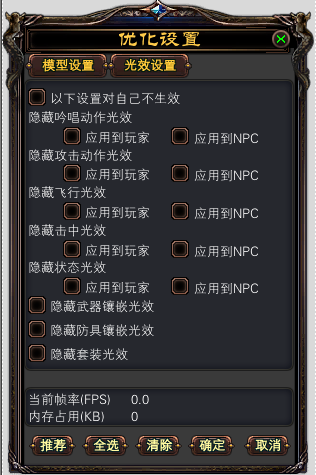
# 优化方案

## 精简方案

客户端内存占用较大的有三个地方：场景、模型、特效。

**场景部分**取决于引擎的优化或者美术简化场景的多边形数目以及纹理精度等。

**特效部分**包括场景特效、玩家动作特效、技能特效、装备特效（魂石特效、套装特效）、BUFF特效。这些地方可以提供一些开关，用于关闭这些特效，如此可降低对客户端内存的消耗。如下图所示：



**模型部分：**玩家主模型CECModel是一个容器，对外绘制的模型通过身体上各个部位的SKIN挂在CECModel上组成，SKIN有如下几种：

SKIN\_BODY\_INDEX

SKIN\_UPPER\_BODY\_INDEX

SKIN\_WRIST\_INDEX

SKIN\_LOWER\_INDEX

SKIN\_FOOT\_INDEX

SKIN\_HEAD\_INDEX

SKIN\_FASHION\_UPPER\_BODY\_INDEX

SKIN\_FASHION\_WRIST\_INDEX

SKIN\_FASHION\_LOWER\_INDEX

SKIN\_FASHION\_FOOT\_INDEX

玩家模型可以分为三个部分：裸体BODY+HEAD（角色默认头部SKIN）、装备SKIN、可以捏的自定义脸（CECFace）。

我们可以根据内存情况为每个玩家定义了四个级别，每个级别对玩家的模型作出相应的调整以减少客户端内存的占用。

MEMUSAGE\_NORMAL不作任何优化，玩家使用正常的模型和脸部。

MEMUSAGE\_NOFACE销毁玩家的自定义脸部，使用角色默认的头部SKIN。

MEMUSAGE\_SIMPLEMODEL销毁玩家的装备和时装SKIN，只显示美术新做的简单BODY和HEAD。

MEMUSAGE\_NOMODEL对于**非重要玩家**，我们销毁其整个模型，重要玩家则令其以SIMPLEMODEL模式渲染。

**非重要玩家**的概念：玩家进行游戏的时候，场景中存在着大量的其他玩家，这些玩家各自做自己的事情并没有与主玩家进行交互。在内存吃紧的时候我们大可将这些与主玩家没有交互的玩家的模型干掉以节省内存。这时候就需要动态的判断一个玩家是否为重要的。重要玩家的指定：被主玩家选中的玩家、相依相偎的对象、决斗的对方、夫妻、交易的对方、和自己有交互的（与自己私聊、加好友、加帮派等等发出各种请求的玩家）、等等,以后如果有需要还可以继续加。

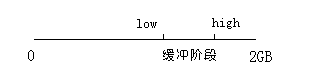
## 优化策略

为玩家提供了两种策略，手动设置和自动设置策略。如下图所示：



手动策略就是玩家通过界面使当前所有其他玩家的内存状态切换到指定的状态，同时对模型做出相应的精简。

自动策略则通过定期检测当前客户端的内存占用，按照策略来调整其他玩家们的内存精简状态。



将所有玩家按照其当前的内存状态MEMUSAGE分为四个组，MEMUSAGE\_NORMAL,

MEMUSAGE\_NOFACE, MEMUSAGE\_SIMPLEMODEL, MEMUSAGE\_NOMODEL; 相同状态的为一组，按照重要程度进行排序，如果重要程度相同则按照与主玩家的距离排序。

当内存使用量大于HIGH时，我们按照MEMUSAGE\_NORMAL, MEMUSAGE\_NOFACE…的顺序去选取玩家进行精简，精简的原则是使他们的MEMUSAGE增加一个级别。例如：假设当前有50个玩家在场景中，其中有48个玩家位于MEMUSAGE\_NORMAL组，2个玩家位于MEMUSAGE\_NOFACE组，MEMUSAGE\_SIMPLEMODEL和MEMUSAGE\_NOMODEL组的玩家数目都为0。这是我需要优先取MEMUSAGE\_NORMAL组的玩家，将他们的自定义脸删除，此时他们的内存状态被增加了一个级别，同时该玩家对象的指针被移到了MEMUSAGE\_NOFACE组。依次为之，直到NORMAL组的玩家数目为0，所有玩家跑到了NOFACE组，此时再在NOFACE组里面取玩家精简，玩家会被删除装备只余BODY和HEAD，精简后的玩家就会跑到SIMPLEMODEL组里；由此一直精简下去，最终所有玩家都会到达MEMUSAGE\_NOMODEL状态。当内存小于HIGH大于LOW，介于缓冲阶段之间时，上述精简过程停止。

当内存使用量小于LOW时，我们按照与上述相反的顺序恢复精简的玩家。即NOMODEL –> SIMPLEMODEL –> NOFACE -> NORMAL的顺序。当内存大于LOW小于HIGH时，恢复精简的过程停止。

通过上述两个精简和恢复精简的过程，我们始终让客户端内存介于LOW和HIGH的缓冲阶段之间，减少内存占用降低客户端因内存用尽而崩溃的概率。

# 遇到并解决的问题

1). 卸载非重要玩家的模型时，要注意需要先下坐骑，否则依然存在的PetModel在TICK的时候，会TICK玩家模型（此时是PetModel的子模型），由于玩家模型已经被销毁，所以引发了崩溃。

2). 变身中的玩家解除相依相偎时，需在ApplyShapeModelChange中调用一次PlayAction确保玩家播放正确的站立动作。

3). 相依相偎时，男方SetPos, 因为女方模型是男方模型的子模型，所以如果男方PlayerModel为NULL, 女方模型就不会被TICK到，此时我们需要TICK一次以确保女方的世界变换矩阵是正确的。

4). NoModel模式下，有些摆摊玩家的摆摊模型看不见。原因：ChangeEquipments函数中负责加载摆摊模型，但被NoModel和非重要玩家的条件给过滤了。解决：把加载摆摊模型的代码提到NoModel和重要玩家判定的前面。

5). 提交给后台线程后的资源加载请求，没法撤销。例如：刚刚进入游戏时，会为发出模型加载请求，然后马上将全局状态切换到无模型状态，但此时加载线程已经开始了资源的加载。解决方法在SetPlayerLoadResult函数中通过当前内存精简状态去筛选我们想要的加载

结果，如在SimpleModel模式下舍弃装备SKIN，只取BODY和HEAD。

6). 有时候日志中会报SKIN和SKELETON不匹配，经查原因为当玩家没有取到服务器发来的PlayerBaseInfo数据时，由于缺少职业性别信息，所以不会立即创建模型和Skeleton对象，而此时如果通过内存状态切换可以使玩家加载错误的简单模型SKIN（此时的职业，BODYID等信息都是错的），等到收到PlayerBaseInfo, 装备, 自定义脸部等数据后，客户端开始为玩家创建模型，此时正确的Skeleton和错误的Skin绑定，就会触发上述错误。解决方法是确保当玩家在没有收到服务器数据前内存切换时引发的模型精简操作不会发生。

7). 当玩家的加载请求正在进行中的时候，如果这时强令玩家切到SimpleModel模式下，最终加载的BODYSKIN是错误的，不是策划新添的简单模型SKIN。原因：SwitchSimpleModel函数里会调用LoadBodySkin去加载简单模型，此函数在主线程中被调用，所以会马上获得BODYSKIN结果并存到m\_aSkins数组中；当之前发出的请求在后台完成加载的时候，SetPlayerLoadResult函数会将新的BODYSKIN把SwitchSimpleModel得到的BODYSKIN覆盖掉。解决办法：SwitchSimpleModel的时候，确保MajorModel为空的时候不发生LoadBodySkin。

8). 妖精变成狐狸后被拥抱，模型卸载后恢复模型显示时，如果男方模型为空则相依相偎恢复失败。解决办法：ApplyShapeModelChange中，DetachBuddy只RemoveChildModel，BuddyID等数据还保留。