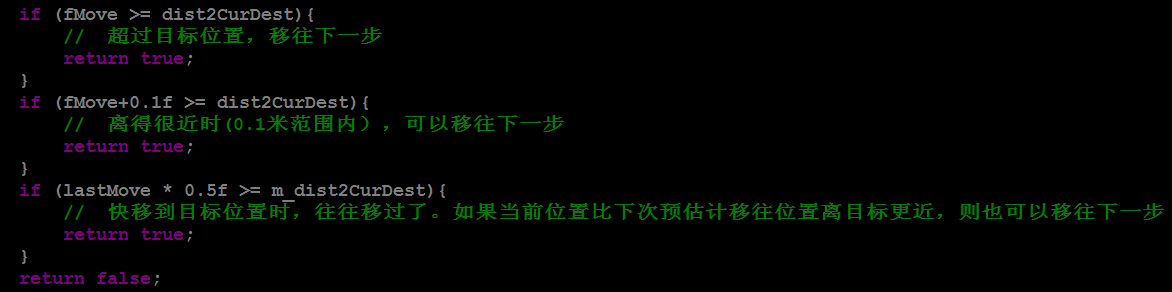
**以下内容基于集成 /GameServer\_MMORPG/AutoPFImp项目，对应支持多层自动寻路的模块。**

1. **路径优化可能：**

当玩家**移动速度较快**、又遇上斜坡时，容易出现朝向抖动，类似下面这张图（绿色为寻路路径节点，红色为玩家的移动距离，实际效果没图中严重）：



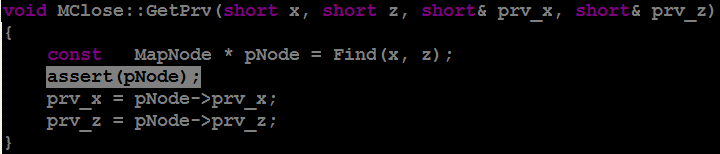
解决思路是，让玩家以路径上尽可能远的目标为移动终点。原先客户端集成代码里，当玩家移动距离**超过**到路径节点的距离时，才会在后续路径上寻找下一个合适节点，可参考下述思路改进：



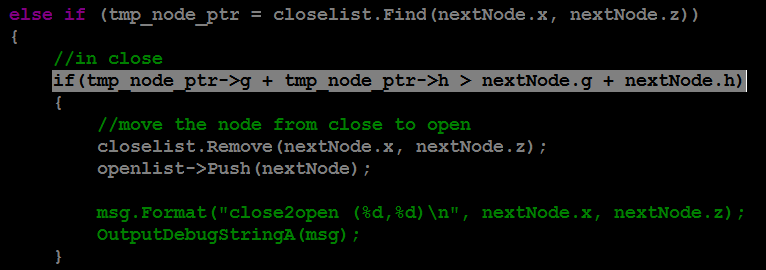
其中，fMove为这次移动距离，dist2CurDest为这次移动前的位置与当前寻径路径上的节点的距离。以前的做法是相当于只做了第一步，即检查了 fMove >= dist2CurDest。

国际采用上述修改后某次测试中，上述3个if判断的成功概率是 131:174:37，中间一种情况更常见。

1. **内存占用优化可能：**
2. COptimizePath::m\_LookUp 类型为short[m\_MapWidth\*m\_MapHeight] ，对1024\*1024地图占用内存为2MB，对国际8\*11\*1024\*1024占用为176MB。但实际使用时只有路径上的少数节点有效。可以修改为 abase::hash\_map<int, short>
3. CluCommon.cpp中加载寻路图的函数bool LoadCluGraph(CGGraph \* g, AFile& fileimg)里可以大量使用reserve，减少CGGraph中包含所有CGNode的vector、包含所有CGEdge的vector、以及每个CGNode所连接的CGEdge的vector中因resize策略带来的内存增加。
4. CGNode中类型为vector<labelValue>的变量m\_Labels，此处没必要使用vector，直接使用原生数组即可。
5. CGEdge中类型为vector<labelValue> 的变量 m\_Labels，此处没必要使用vector，直接以一个浮点型weight代替即可、表示权重。
6. CGEdge中涉及寻路代价的weight浮点类型保存类型可以由double修改为float，CGNode中保存的weight也同时修改，可以少量减少文件大小，运行时能减少部分内存。此修改将**导致文件格式变化**（具体涉及CLU\_FILE\_VERSION、MLU\_FILE\_VERSION两个定义），兼容已有文件需要写部分代码。运算精度的修改，也更容易**导致一个崩溃BUG重现**（见下）。
7. **集成时发现的BUG：**
   1. 寻路过程代价计算和比较从浮点double修改为float后，很容易导致崩溃（在MClose::GetPrv中，经由CMAStar::\_GeneratePath调用），直接原因为寻径成功后按顺序提取路径上结点过程中找不到中间结点了（下面截图中pNode为空），因为在搜索过程中因BUG被删除了：

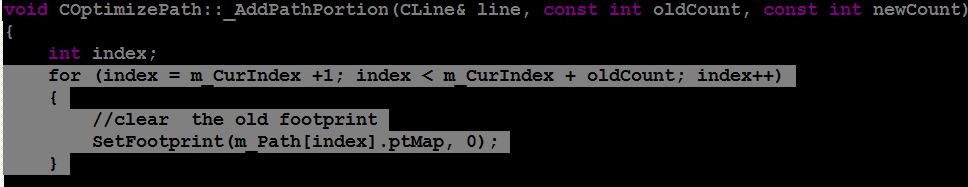


罪魁祸首在CMAStar::Search中的这句（下面截图中高亮）:

****

发生问题时此处tmp\_node\_ptr和nextNode实际内容指向同一节点，但浮点计算误差而使if语句结果为true，导致closelist.Remove等语句被调用。

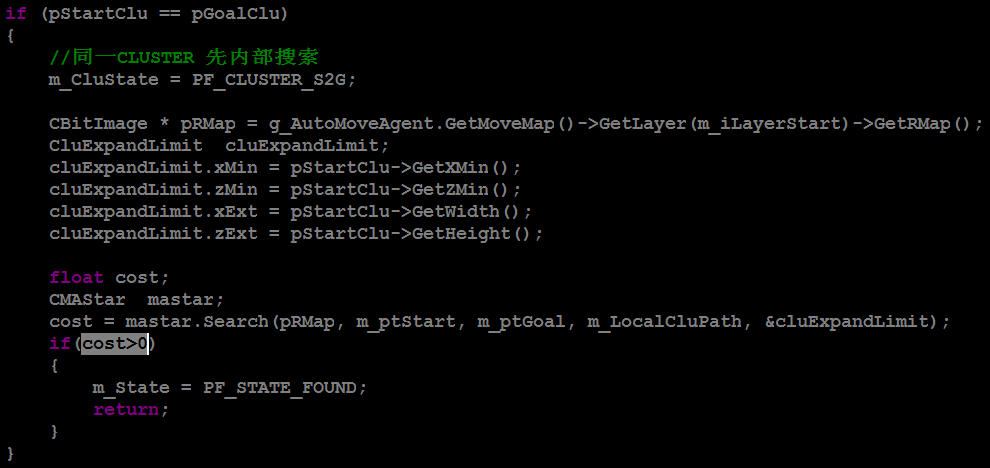
* 1. COptimizePath::\_AddPathPortion中：



上述index<应改为<=，需要处理m\_CurIndex+oldCount。

在这个函数的末尾，应对m\_Path中新增加的PathNode调用SetFootPrint(\*,1)，对应m\_Path下标范围 [m\_CurIndex+1, m\_CurIndex+newCount]。

* 1. CPfClu::SetupSearch 当 pStartClu == pGoalClu 时，if (cost > 0) 修改为 if (cost >= 0)，以正确处理待寻径两点在同一搜索位置的情况：



* 1. bool CMoveAgent::Search 实现有漏洞，未考虑 CMoveAgent::SetStartEnd 调用后状态：

