【完美国际】移动施法

**文档撰写时间：**2014年9月29日

**关键词：Angelica 2.0，VC6**

**策划需求：**针对本次资料片新种族胧族的两大职业之一月仙的部分技能，要求实现边移动边施法，具体内容约定为：

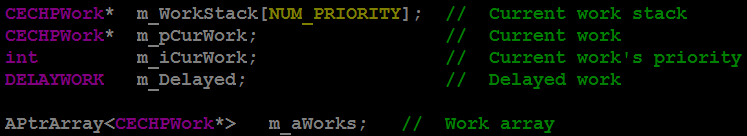
* 针对不可以移动施法的技能，当点击施法时，移近目标到可以施法的距离、吟唱施法；
* 针对可以移动施法的技能：
  + 如果当前没有移动，点击施法时，移近目标到可以施法的距离、吟唱施法；
  + 如果当前正在移动且离目标距离够近，则边移动边施法、且身体不背向目标而头部朝向目标；
  + 如果当前正在移动但离目标距离不够近，则忽略施法而继续移动。

**实现难点：**有两个难点：

* 客户端实现移动的类CECHPWorkMove和实现施法的类CECHPWorkSpell同时只能存在一个，要实现边移动边施法，必须允许两者同时存在，因此需修改CECHPWorkMan中的管理机制、并修改客户端当前同时运行的只有一个CECHPWork子类（如CECHPWorkMove、CECHPWorkSpell）的假定；
* 移动施法过程中需要实现上身播放攻击动作、下身保持行走，并保持头部朝向目标，且随时与普通动作之间的切换。

**多个CECHPWork同时存在的实现：**

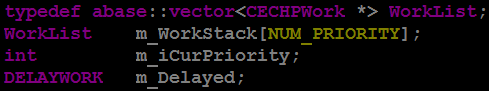
CECHPWorkMan原先的结构：



m\_WorkStack是优先级栈，m\_pCurWork等价于m\_WorkStack[m\_iCurWork]指向当前运行的CECHPWork子类。m\_Delayed指向被暂时延迟生效的CECHPWork子类。

由于要允许多个CECHPWork子类存在，m\_pCurWork变量必须删除，而m\_WorkStack中每个元素要修改为一个列表，指向同时运行的CECHPWork子类序列。m\_iCurWork涉及到优先级调度，可以保留，而m\_Delayed是不易理解的设计，但由于与玩家习惯绑定，也保留下来。

CECHPWorkMan修改后的结构：



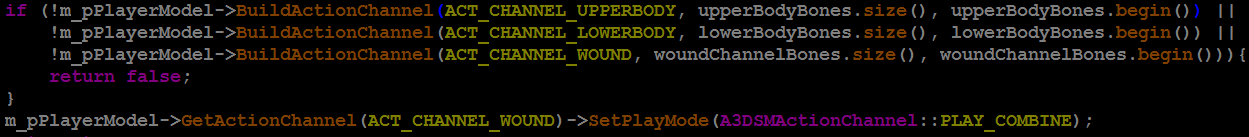
原来结构中的m\_aWorks保留了CECHPWork:: NUM\_WORK 个对象，用于m\_WorkStack使用，可能用于减少内存碎片。新的结构中去掉了，原因是担心这个假定在多个CECHPWork子类同时运行时有麻烦。

与这个结构相配合，原有的CECHPWorkMan中的Get方法GetCurWork、GetCurWorkID等及FinishCurWork都必须删除，替换以FinishRunningWork、IsWorkRunning等

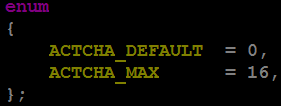
**动作播放和头部转向实现：**

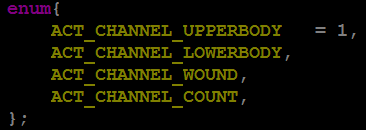
* 【**上半身攻击、下半身移动实现**】

支持移动攻击动作表现的职业的模型，需要将全身骨骼分为上、下半身两个通道，通过CECPlayerActionController::BuildChannelForCastSkillWhenMove函数调用CECModel:: BuildActionChannel(int iChannel, int iNumJoint, int\* aJoints)建立通道：



ACT\_CHANNEL\_WOUND是受伤动作播放的通道，包含人身上所有骨骼，其动作会与现有所有动作合成，有特殊逻辑，是特殊通道。

引擎支持的通道最多为16条：

排除第0条通道（默认通道），我们使用了编号为1、2、3条的通道，分别为上半身通道、下半身通道、伤害通道：

对生成通道的模型，播放动作时，需要上下半身同时播放：



ecm模型上组合动作需要设置通道优先级，

1. 定义可移动施法的技能动作为A类动作；
2. 定义非A类动作中可与移动施法技能动作合成的动作（如奔跑）为B类动作；
3. 定义其它动作（如站立、攻击类动作、不可移动施法的技能动作等）为C类；

则通道优先级的规则为：

* + A类动作上半身通道优先级设置为1，其它通道优先级为默认值0；
  + B类C类等其它所有动作下半身通道优先级设置为1，其它通道优先级设置为默认值0。

这样可以做到：

* + A类动作之间，或B、C类B、C类动作之间切换时，由于动作通道优先级设置相同，会产生覆盖行为，达到后播放动作替换之前动作的目的；
  + 播放B类动作时再播放A类动作（即移动中施法），A类上半身动作优先级高而进入最终表现，B类下半身动作优先级高而进入最终表现，达到动作合成目的；播放A类时再播放B类也一样（施法过程中移动）；如果B类动作中途结束，剩下的A类动作下半身动作会自动进入最终表现，A类动作中途结束也类似；
  + 从C类往A类切换时，为避免产生合成效果，需清除原有C类动作；A类往C类切换时，也需清除原有A类动作；

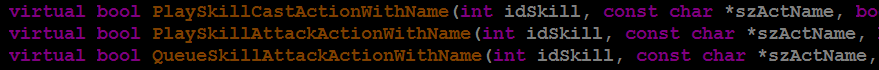
由于需要区分A、B、C三类动作，因此动作播放接口不再是统一接口，而拆分为如下三大类：

* + 非技能动作播放接口（定义为X类接口）：

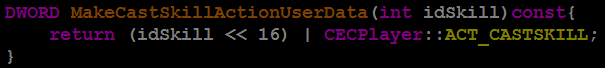


将参数中iAction与即将播放的动作关联（通过CECModel::PlayActionByName参数中的dwUserData），并随时查询当前播放动作是否为B类动作；

* + 技能动作播放接口（定义为Y类接口）：



将参数中的idSkill与即将播放的动作关联（通过CECModel::PlayActionByName参数中的dwUserData），dwUserData计算方式如下：



此计算方式不但能与非技能动作播放接口区分，还包含技能ID,以供随时查询当前播放动作是否为A类动作；

* + 伤害动作播放接口（定义为Z类接口）：



此类动作播放使用单独通道ACT\_CHANNEL\_WOUND，且直接通过CECModel::GetA3DSkinModel()->PlayActionByName播放，并与当前播放动作合成。

根据上述接口，当使用X类接口时，需结合正在播放的动作是否为B类动作、和正要播放的动作是否为A类动作进行处理；使用Y类接口时也类似。另外，还要考虑到不支持移动攻击的角色模型，具体实现见EC\_PlayerActionController.h及EC\_PlayerActionController.cpp，对模型和技能动作是否支持移动攻击的判断和通道生成所需骨骼的统计工作统一在EC\_CastSkillWhenMove.h及EC\_CastSkillWhenMove.cpp中。

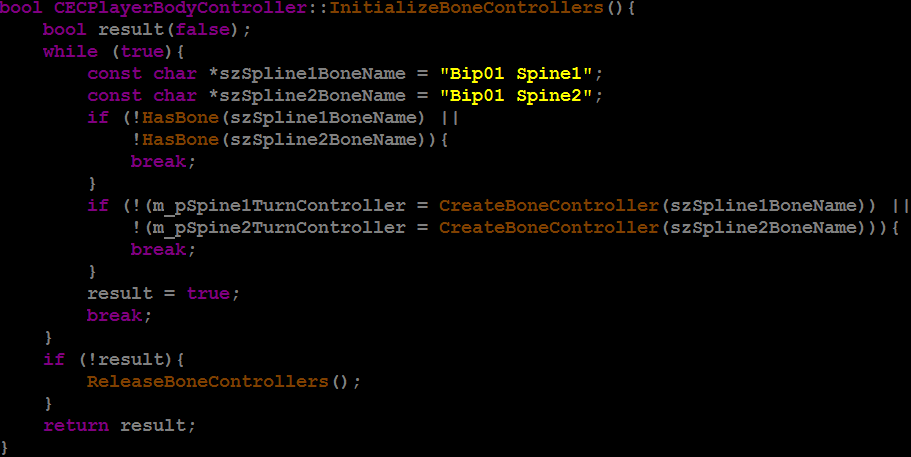
* 【**移动中头部始终朝向目标实现**】

转向逻辑为：

* + 非移动施法时，模型朝向由外界控制，一般与移动方向一致，但空中、水中转向期间会不一致；
  + 移动施法时，如果从玩家到目标的向量与玩家移动方向夹角小于90度，则玩家模型朝向与移动方向一致；否则，玩家模型朝向与移动方向相反，即退着移动。两种情况下，头部都能借助小于90度的转向正对目标（通过控制部分骨骼的转动）。

转向需要设置部分骨骼的额外旋转角度，引擎对此的支持是为每个需要设置额外旋转角度的骨骼创建一个A3DBoneBodyTurnController对象，通过A3DBoneBodyTurnController::SetTurnDeg设置额外旋转角度，每个骨骼转向一定值，累积到头部转向目标。

游戏内仅对2个骨骼创建了A3DBoneBodyTurnController：



如果创建了更多A3DBoneBodyTurnController，则当模型偏离（0，0，0）位置较远时（如在8\*11的大地图的左下角位置），由于浮点数(float)计算误差累积，末端骨骼（人物头部）会出现抖动现象，无法接受；若采用引擎升级浮点数为double改动也很大，故放弃。

转向实现的控制类CECPlayerBodyController基于以下逻辑：

* + 没有朝向目标时，即静止或无目标位置移动时（如按住WSAD），模型朝向由外界控制；
  + 朝向某目标未移动时，模型面向目标位置；
  + 朝向某目标移动、但非移动施法时，模型朝向与移动方向保持一致；
  + 朝向某目标移动施法时，模型朝向与移动方向相同或相反，再通过小于90度的转向角使头部朝向目标；
  + 以上模型朝向与目标或移动方向关联而不是由外界控制时，Y轴需置0以保持上身直立，否则空中或水下攻击时，会出现身体歪的情况；其它情况，如相依相偎时，女方模型朝向完全由男方挂接处骨骼控制，强行修改会出现头部与身体分离的BUG；
  + 转向角只在移动施法时生效，当然，状态切换是平滑过渡的。

转向实现在EC\_PlayerActionController.h及EC\_PlayerActionController.cpp中。