链滴

6. 读取动画

作者：xu365082218
原文链接：https://ld246.com/article/1500906847595
来源网站：链滴

流星的动画架构还是相当的精炼的。他共用了一部分动画。然后每个角色还有自身的特色动画。br>
要说动画，至少有 4 个组成部分
1，骨骼
2，模型与骨骼绑定的蒙皮数据-骨骼权重
3, 骨骼的帧
4，动画定义 多少帧-多少帧 是否循环
骨骼 就是 p0.bnc 文件 bnc 文件，保存的是 TPOSE 下，每根骨骼的姿势
模型与骨骼绑定的权重 p0.skc 文件 此文件即第一章里加载的，只是那时候是读取静态文件，而这里需要读取骨骼权重
骨骼的帧 p0.amb character.amb 文件，前者是角色特有的动画，后者是招式动画，招式动画是全角公用的
动画定义 p0.pose 文件
有了这四者，那么动画就可以出来，无法使用 u3d 的动画格式（要使用就必须自己写 max 脚本转换之前写过一个脚本提取了 character.amb 里的 17071 帧动画，电脑开了一天跑那个 max 脚本），以直接用代码实现自己的动画读取播放等，虽然简陋。
有几点基础要首先讨论
1 骨骼动画的坐标系，原流星是右手坐标系，要换到左手坐标系，不光模型要换，连骨骼的 位置，旋，也需要相应的转换
这里的转换原则是
位置 转换后的坐标(x,y,z) = 转换前的坐标(x, z, y) y 和 z 互换
旋转 转换后的四元数(w,x,y,z) = 转换前的四元数(w, -x, -z, -y) x = -x ;y = -z ;z = -y
2 动画间的过渡
说动画间的过渡，其实意思应该是 2 个关键帧之间插入一些普通帧，让 2 个关键帧看起来不是瞬间就生了变化，这部分也是有一定麻烦的
因为插入多少帧，多了很平滑但是动作看起来缓慢，少了普通帧缺少细节，类似防御，刚一点，就从 Ile 切换到了 Defence 姿态，中间的普通帧都没有了
还有一点是插入的普通帧姿态怎么算的，一般位移就用 vector.lerp，旋转就用 quaternion.slerp 来从 2 个关键帧中按播放了多少普通帧读取
3 动画的位移
有些动画，是带位移的，并且播放完之后自身的位移就是动画的位移后的位置了，类似匕首的大招，个位移，在原本 U3D 内的动画系统可设置的时候，点一个 apply root motion 就可以了。可是这里得自己去实现。
而且有些动画，类似大刀和双刺的绝招，我不知道他是怎么实现的，好像在动画中还可以参照敌人的置，进行一个修正，导致后续的招式会不断的朝目标去攻击，这一点是目前没有做到的（可能原系统是这么做的）
4 角色的重力
重力这部分涉及到角色跳跃的一些物理反馈，类似在墙壁上飞檐走壁时，有一个向下的加速度，导致色不能无限飞檐走壁，这部分比较复杂，也还未解决
5 角色的阻挡
带位移的动画会遇见有阻碍的情况下，应该是无法穿透阻碍的，比如正对着墙壁，使用匕首的绝招，该是抵着墙壁发出大招，而不会穿越墙壁，由于动画没办法得自己写弄的异常麻烦，这里直接使用 characterController 控制角色，当动画发生位移的时候，调用
characterController 的 move 函数，即可被阻挡
代码是在太多无法一一讲解，还是贴视频吧
<a href="http://v.youku.com/v_show/id_XMjkxNjgxMTUzNg==.html">点击查看武器绝招动画<a>
源工程就不贴了，这部分内容需要兴趣，这里只提供各种文件格式的解释，如果有兴趣也可以给我留，实在需要源工程，之后会提供。
amb 文件
5 字节文件头
4 字节骨骼数量 bones
4 字节虚拟体数量 dummy
4 字节帧数 frames

4 字节 FPS（估计）
//对每一帧有如下代码读取
for (int i = 1; i < frames; i++)
{
4 字节标志位
4 字节帧序号
4 字节 x float 型
4 字节 z float 型
4 字节 y float 型
//这个 pos 是每一帧根骨骼的 localposition 其他骨骼在动画帧中是只有旋转，而无移动的。
//循环中嵌套循环
//每一帧的每一个骨骼
for (int j = 0; j < bones; j++)
{
//四元数的开始
4 字节 w float 型
4 字节 x float 型 取负，坐标系转换为左手
4 字节 z float 型 取负，坐标系转换为左手
4 字节 y float 型 取负，坐标系转换为左手
//读取完了后 Quaternion = new Quaternion(x,y,z,w)
}
//每一帧的每一个虚拟体
for (int j = 0; j < dummy; j++)
{
流从当前位置往后移 5 字节
4 字节 x float
4 字节 z float
4 字节 y float
4 字节 w float
4 字节 x float 取负
4 字节 z float 取负
4 字节 y float 取负
localposition = new vector3(x,y,z)
localrotation = new Quaternion(w,x,y,z)
}
}
其他文件都是文本行文件
实际上创建一个空对象 meteorUnit 放到原点，并且置 0（位置圆点，旋转 Quaternion.identity,缩放 v3.one） 加入 skinmeshrenderer 组件然后把骨骼和虚拟体从 bnc 文件里读出来并且创建到一个树录上(树的根是 meteorUnit)，让 skinmeshrenderer 的 bone 指定到这个骨骼数组，不包括虚拟体<r>
然后通过 skc 加载顶点数据 mesh 以及贴图和 uv，并且设置好权重，之后把所有骨骼的 worldtolocamatrix 取出得到 bindpose 数组,也即世界到本地变换矩阵
如果不用 skinmeshrenderer 就没有这些东西了，也就是一个顶点，受到数个骨骼影响，每个的权重 其旋转矩阵 × 顶点坐标=经过骨骼影响后顶点的坐标，没有太仔细写过细节，大致就是顶点经过几矩阵 每一个矩阵代表一个骨骼，1-4 个骨骼，每个不同的权重，就可以计算出某一个帧下，全部顶点坐标，这样看起来就是模型动起来了
然后读取 pose 文件，pose 文件类似一个文本的动画列表文件，其没有名称只有序号，内部定义了哪一帧开始到哪一帧结束，而且指定了源文件是 从 p0.amb 里读还是 character.amb 里读，source 0 表示 character.amb source 1 表示 p0.amb
类似 p0.pose 中的
Pose 0
{
source 1

Start 1
End 65
LoopStart 1
LoopEnd 65
}
这个指的是 Idle 动画 使用 p0.amb 文件，从第一帧开始 到第六十五帧结束 而且是循环类型的动画。
br>
如此，就可以读取动画了</p>