《基于扩展位置动力学方法的全身逆向动画技术研究》

说明文档 by曾可为

研究目的

游戏中，大多数虚拟角色模型都很复杂，无论是类人模型还是非类人模型（蛇、昆虫、多足动物、绳子）。这些模型包括许多关节，具有大量的自由度。为了游戏的真实性，它们往往需要满足许多约束。从前的游戏需要美术人员和游戏设计师根据关卡、环境、动作，手动调整所有的自由度和约束条件，从而让角色做出需要的动作，这是一个极其复杂和耗时的过程。因此大部分游戏往往选择简化角色的部分动作表现来节约时间和成本，例如，当玩家获取桌子上的物体时，不播放手臂抓取物体的动画，而是直接将物体放入玩家物品栏，又或者在玩家骑乘坐骑时，仅把坐骑作为一个整体建模，并用简单的刚体碰撞进行约束，而忽略不同地形条件下坐骑移动方式的不同。这种简化方式降低了游戏的真实性，牺牲了部分的游戏体验，并经常导致穿模现象的发生。除此以外，严格限定的动画表现也给游戏设计师带来了诸多不便，例如，门必须适应角色的高度不然角色会直接穿过门框；障碍物必须是固定的高度否则角色可能根本跳不过去；楼梯必须是固定的间距否则上楼的动作就会显得极其不自然。

因此，有必要寻找有效的方法来操纵由复杂的多关节模型组成的系统，GDC2016中，育碧的开发人员首次提出了在游戏中使用反向运动学（IK）这一技术来控制多关节模型的动画表现[1]。动画师或动作捕捉系统指定关节末端需要到达的位置（例如桌子上的物体位置），剩余肢体关节的位置则根据设计好的约束和反向动力学算法实时计算得到。这一技术可以为动画师节省大量的工作，同时实现游戏角色动画的环境自适应，增加玩家的游戏体验。自此之后，IK系统在游戏中得到了广泛的应用。

但是，主流 IK 技术仍局限于局部肢体的 IK 求解，例如腿部 IK、手部 IK、头部 IK 等，这些技术在局部有较为真实的表现，但是在匍匐、攀爬、游泳等需要手脚配合的运动中就会出现明显的不协调问题。针对这些常见的运动情况，人们开始研究全身逆向动画技术。全身逆向动画技术能够统一手脚的动作，并根据手脚运动带动身体的其它部位，使得运动过程更加协调自然。该技术被广泛应用于虚拟现实、虚拟人中。根据穿戴设备或是摄像头捕捉的手脚位置，就能够使用全身逆向动画技术解算出合理的身体动作。在游戏中，可以使用全身逆向动画技术控制角色全身的运动情况，模拟出角色各种状态的动画特征，使得角色动画更加真实和连贯。同时，通过使用全身逆向动画技术，也可以解决局部肢体 IK 的不协调问题，节省动画开发的时间和成本，实现更加创新和复杂的动画效果。因此，全身逆向动画已经成为现在程序化动画领域研究的热门问题之一。 虽然全身逆向动画有非常广阔的应用前景，但是该技术在除 VR 以外的商业游戏中并未得到广泛的应用。这主要是因为现有的全身逆向动画技术存在以下局限性： （1） 全身逆向动画求解方法现有的复杂度随关节数增加呈几何级数上升，在3-4 个关节的局部肢体情况下可以满足实时性需求，而对于约有 70 个关节总数的人体模型会导致复杂度过高，计算开销过大。简单的 VR 游戏可以将大部分计算性能分配给 FullBodyIK 求解，而对于更复杂的主机游戏则不可能满足实时性的要求。

（2） 现有的全身逆向动画求解方法并不稳定，仅仅对于几个关节的肢体都会存在扭曲和死锁等情况，而直接应用于人体模型会放大这种不稳定性，在很多情况下会导致整个人体呈现不自然的运动姿态。

如果想要在一般游戏中广泛应用全身逆向动画技术，新的方法必须能够同时满足以下条件：

（1） 复杂度足够低：其复杂度必须是常数量级，每帧平均耗时不能超过 1ms，否则会导致动画线程拖慢游戏总体帧率。

（2） 稳定性足够好：不会出现频繁的抖动和死锁问题，在不同平台、不同配置下有相同且平滑的表现。

（3） 视觉表现合理：新方法的视觉表现必须足够真实，不能有明显的异常行为，其动作表现需要接近真人动捕数据。

在上述条件都满足的情况下，人们就可以完全用全身逆向动画取代局部肢体的反向运动学模型，从而同时控制角色全身的运动情况，模拟出角色各种状态的动画特征。因此为了使全身逆向动画能够在实际游戏中得到广泛使用，本文针对现有的全身逆向动画技术存在的问题，在实践过程中对全身逆向动画技术进行研究，提出了基于扩展位置动力学方法的全身逆向动画技术，对上述三个方面的问题进行了优化，证明了基于扩展位置动力学方法的全身逆向动画技术在实际 3A游戏中应用的可行性。

主要研究工作

一．骨骼扩展位置动力学方法（论文3.1）

通过结合人体骨骼特征，提出了骨骼扩展位置动力学方法，实现了基于扩展位置动力学方法的全身逆向动画技术。

因为在全身逆向动画求解时，一般不允许骨骼出现任何形变。因此可以将骨骼视为绝对刚体，做如下近似：

• 不同骨骼密度相同，骨骼质量 𝑚 正比于长度 𝑙。

• 骨骼形状近似于半径为 0 的细棒。

• 因为骨骼不允许形变，所以其刚度系数 𝛼 为 0。

同时，引入了约束和刚度系数对不同骨骼的运动特性进行区分，保证运动姿态的合理性。

二．针对角色骨骼动画的全身逆向动画迭代方法（论文3.2）

在经过上述调整后，基于扩展位置动力学方法的全身逆向动画技术在实际性能测试中，仍然不能满足游戏的实时性要求。因此需要对原有方法进行简化，尽可能减少迭代的复杂度。因为该方法应用于人类骨骼系统，所以本文尝试根据人体运动特征对原方法迭代公式做出简化，取得了立竿见影的效果。

因为运动过程中，目标位移总是驱使骨骼向目标点的方向旋转，导致骨骼自身方向和目标点方向的夹角变小，因此，本文尝试省去广义逆质量中的角速度相关部分，从而显著减少了计算资源消耗。

因为人体骨骼结构中，越到中间的骨骼质量越大，在质心计算中的相应权重也越大。因此可以将原有的公式简化为计算该骨骼位置和子骨骼质心位置平均值，因此不需要再重新计算质量。这样无论位于哪个层级的骨骼，计算质心都只需要计算一次，显著减少了计算量。

因为相邻的骨骼质量相差不大，因此可以根据固定骨骼和可动骨骼进行划分，如果均为可动骨骼，则均分动量和角动量，如果有一方是固定骨骼，则动量角动量全部由可动骨骼承担。

三．全身逆向动画与物理动画结合的程序化动画框架（论文3.3）

物理动画作为程序化动画的另一大分支，在游戏中有着广泛的应用。为了优化全身逆向动画体系的表现效果，增加动画的真实性与灵活性，本文将全身逆向动画与虚幻 4 中的物理动画结合，提出了全身逆向动画与物理动画结合的程序化动画框架。

该框架使用融合曲线来控制两种动画体系的融合权重，根据应用场景动态切换主导动画体系和辅助动画体系。当角色处于攀爬、行走和悬挂状态时，需要确保运动的真实性，因此以全身逆向动画体系为主导。全身逆向动画技术用于控制肢体末端运动，以保证角色肢体运动的真实性。同时，物理动画技术作为辅助，仅在角色攀爬或奔跑时的跳跃过程中启用，用于模拟角色身体剧烈运动过程中的惯性晃动和环境交互。在运动过程中，物理动画的融合权重曲线随着运动的进行先从 0 增大到最大值，后逐渐减小，当接近地面的时候重新变为 0。同时，角色四肢物理动画权重要低于角色躯干的物理动画权重，相应的，四肢全身逆向动画权重要高于躯干的全身逆向动画权重。

当角色处于受击、倒地和死亡状态时，需要确保和环境交互的灵活性和多样性，因此以物理动画体系为主导。物理动画技术用于模拟角色身体与敌人和环境的交互，根据力的大小、方向、地形呈现出丰富的动画效果。同时，扩展位置动力学技术作为辅助，在角色倒地或死亡时启用，用来确保角色不会因为原有的倒地和死亡动画和地面或是物件发生穿插。在倒地过程中，全身逆向动画的权重从0 开始逐渐增大，在角色着地后达到最大值。随着角色着地，提前将会穿插的臂或是腿部关节向上拉抬，使得角色倒地过程中始终保持躯干先着地，四肢后着地，避免四肢和环境或身体碰撞导致的剧烈抖动问题。

四．结合运动预测和弹簧臂插值的全身逆向动画技术管线（论文3.4）

为了提高复杂环境下的实时性和稳定性，本文将运动预测和弹簧臂插值引入了全身逆向动画求解过程，实现了新的全身逆向动画技术管线。在角色运动时，先通过运动预测得到前方地形情况，再使用弹簧臂插值的方式计算下一帧四肢末端的目标位置，最后，使用全身逆向动画求解计算出角色合理的动画姿态，并以动画融合的方式影响角色实际的动画表现。

当角色运动时，首先通过运动预测，根据动画信息，获取前方地形高度差，得到高度曲线图。再通过弹簧臂插值，根据高度曲线图，以插值的方式得到肢体末端的运动曲线。之后，全身逆向动画根据末端运动曲线，驱动末端运动，进而带动身体其余骨骼随之运动。因为角色运动前就已经得到了一个运动周期中的运动轨迹，因此运动过程会显得连贯且稳定，同时不会因为特殊或极端地形发生突变。同时，根据运动状态的不同，按照对应的融合曲线与物理动画融合，以提供灵活、多样的动画表现。最后，将全身逆向动画技术管线的输出和固定的帧动画结合，就可以得到视觉效果好、稳定性强、真实性好的游戏角色动画。

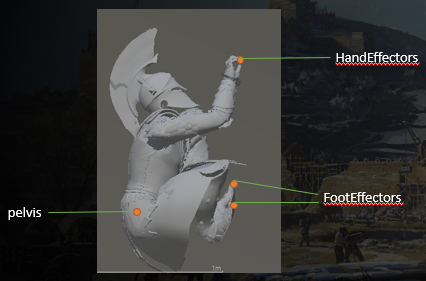
五．虚幻4引擎使用（论文第四章）

开发的工具以插件的形式嵌入到UE4动画系统中，主要复写了插件系统中的插件名.Plugin文件和插件名.Build.cs文件，前者用于描述插件的基本信息，例如版本号码、插件依赖的模块、作者名称、图标等，后者用于描述该模块和其它模块的依赖关系，相对路径与其它各种编译设定。在实际使用中，需要注意插件名.Build.cs文件中的依赖模块只能包含运行时模块，不能包含编辑器模块，否则在手机端会报错。因为UE4插件本身独立于游戏源码，可以随时开启与关闭，不需要重新编译，因此具有很强的复用性。用这种方式导入到游戏引擎，能够使我们的工具方便地应用到不同游戏场景。

同时，使用了UE4提供的骨骼控制(Control Rig)蓝图，该蓝图是UE4.26添加的新特性，其主要目的就是用于运行时的全身姿态控制，并且提供骨骼关系和骨骼位置接口，因此只需要对每个骨骼分别创建对应的Control Rig蓝图，之后在蓝图内部添加全身逆向动画系统节点，便可以实时控制角色姿态表现。

主要功能

主要以攀爬的形式实现，也可以用于行走和瞄准等需要IK的场景。以攀爬场景举例：



攀爬场景使用示例

首先，在角色手脚处建立四个虚拟骨骼用于标记Idle状态手脚稳定位置（图中的HandEffectors、FootEffectors）。这四个骨骼保持与根骨骼(pelvis)相对位置不变，并跟随根骨骼运动。

在跳跃过程时，从这四个虚拟骨骼出发，向前方发出射线检测。射线将会返回第一个碰撞物体的位置和法线方向。我们计算碰撞物体位置和手脚当前位置的差，得到手脚偏移量。之后将偏移量以弹簧臂插值的形式逐渐添加到运动过程中。

同时，根据碰撞物体的法线方向，可以计算身体的俯仰角旋转，让身体与攀爬表面平行。因为旋转后手脚位置会发生相应的偏移，因此旋转后还需要平移身体，使得手部位置和原本的射线检测位置相同。需要注意这样做并不能完全解决脚部的偏移，但考虑到手部的悬空比脚部更加明显，因此优先保证手部偏移的准确性。

除此以外，因为攀爬过程往往与跳跃过程结合，而跳跃过程我们不希望受到逆向动画的影响，因此需要对跳跃动画进行标记，在其中某个时间段内禁用逆向动画，或是对逆向动画的最大偏移量做出限制。

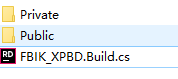
最后，我们将计算出的偏移量输入全身逆向动画系统，作为质心偏移量进行计算，按照前面公式的步骤，迭代求解全身姿态。

如果应用的是角色行走或是瞄准，只需要更改末端位置和射线检测方向，再将检测出的目标位置输入求解器即可。

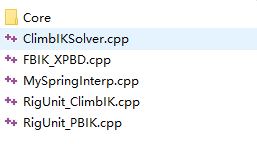
代码结构和模块说明

一．模块介绍

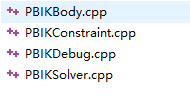
如图所示，Private文件夹中包含代码的.cpp部分，而Public文件夹中包含代码的.h头文件部分，这两部分的代码是一一对应的。PBIK.Build.cs文件使用C#语言，包含UE4的依赖库信息，需要注意的是代码依赖UE4的ControlRig插件，所以务必在UE4中启用该插件。



接下来以Private文件夹为例说明主要模块和对应功能，如图所示：



Core文件夹包含XPBD算法核心代码，其结构如图所示：



各代码模块对应的功能如下：

PBIKBody：该模块用于存储身体骨架的相关信息，包括子骨骼、父骨骼、质心、骨骼质量（长度）、末端骨骼等。按照XPBD算法实现往骨骼添加力后不同部位的位移和角度变化计算，并提供了供其余模块使用的接口。

PBIKConstraint：该模块用于存储骨骼之间的约束信息，包括关节间约束和关节末端约束。同样按照XPBD算法实现关节约束和末端约束的求解，并根据每次求解结果更新位置、角度以及约束信息。

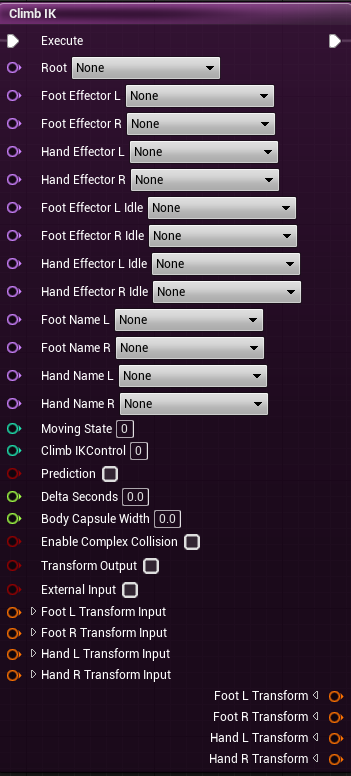
PBIKDebug：该模块用于绘制XPBD算法求解前后骨骼位置变化情况，一般用于调试使用。

PBIKSolver：该模块提供XPBD算法入口。首先，根据模型情况初始化身体和骨骼。之后，读取设置好的约束参数，当前姿态和目标末端位置，根据这三者计算骨骼位置变化，并根据计算结果更新末端位置。之后求解关节约束情况，将末端位置的变化传递到全身，该过程会迭代进行两次。最后输出计算后的身体骨骼和末端位置。

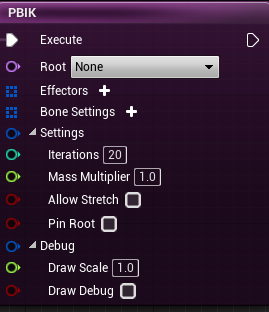
ClimbIKSolver.cpp：该模块用于角色攀爬使用。取角色双手双脚为末端效应器，以末端效应器当前位置为基准，向前方发射射线。射线命中的位置即为末端需要到达的目标位置。同时包含旋转、线与面相交算法等接口。

FBIK\_XPBD：该模块用于以插件的形式在UE4中注册一个命名空间，属于UE4格式上的需求，本身没有其它作用。

RigUnit\_ClimbIK：该模块用于构建一个角色攀爬节点。主要包含的参数有双手双脚的效应器位置、名称、各种控制参数等。该模块接收这些输入参数，然后调用求解器模块的算法进行计算求解，最后将求解结果输出。如图所示：



RigUnit\_PBIK：该模块用于构建一个PBIK节点。主要包含参数有：效应器骨骼、每个骨骼的约束设置、迭代次数以及各种控制参数和Debug参数等。该模块接收这些输入参数，并输出处理后的身体姿态。如图所示:



MySpringInterp:该模块为弹簧臂插值算法模块，其基本原理是以正弦函数形式实现的非线性插值方法。（远离平衡点斜率大，靠近平衡点斜率小）

二．模块间的依赖和调用关系



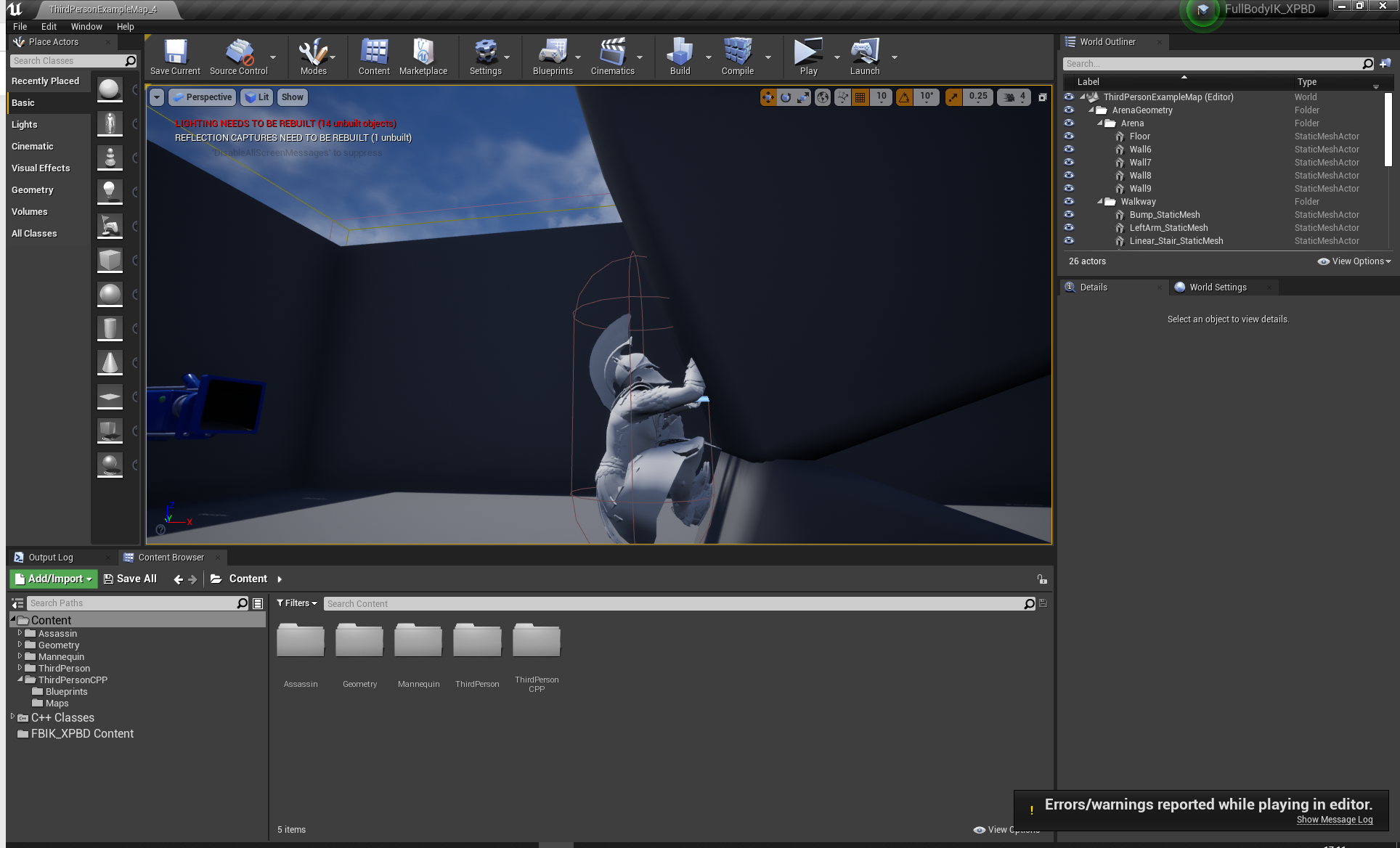
使用方法

一．Demo中的使用实例

Demo文件夹中的FullBodyIK\_XPBD压缩文件中有使用示例。首先，解压FullBodyIK\_XPBD压缩文件，打开项目文件（需要安装好4.26版本的UE引擎）：

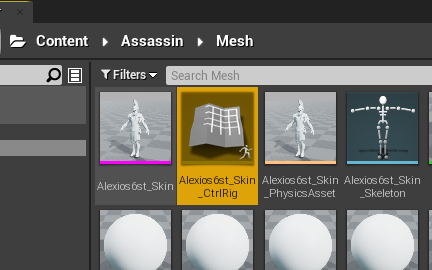


打开后，会进入开始界面，如图所示：

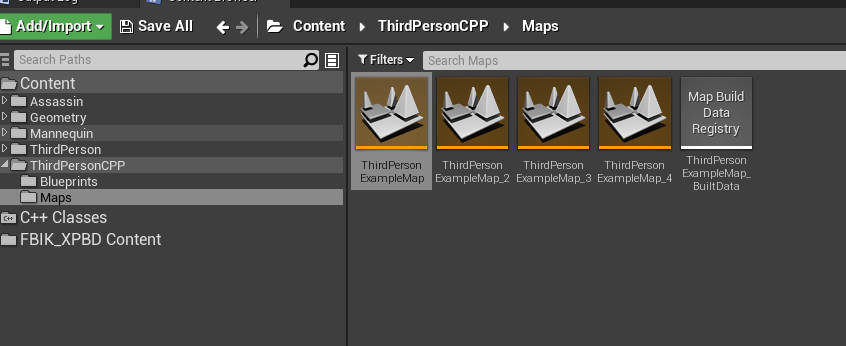


点击左上角的Play之后，按住空格便可以看到人物跳跃姿态变化。

控制人物姿态的蓝图是位于Content/Assassin/Mesh目录下的Alexios6st\_Skin\_CtrlRig蓝图。双击打开可以调整IK配置。如图所示：



场景位于Content/ThirdPersonCPP/Maps目录下（默认打开的是第一个场景），其余场景中摆放不同的攀爬物件：

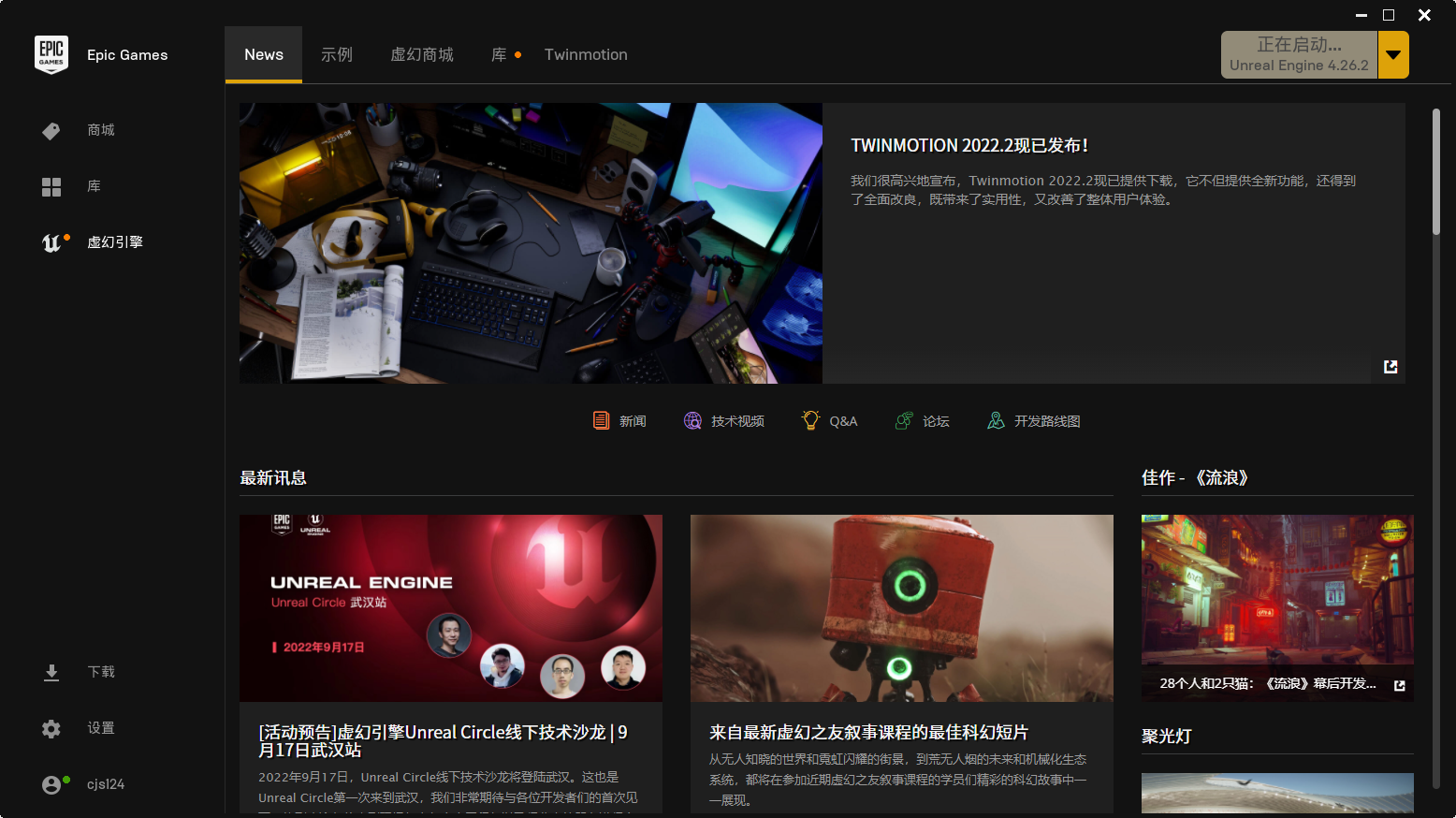


在场景中可以任意组合攀爬物体，可以观察不同攀爬地形IK动作的变化。

二．如何在新项目中使用该插件

如果是从零开始创建新的项目，使用该插件的方法如下：

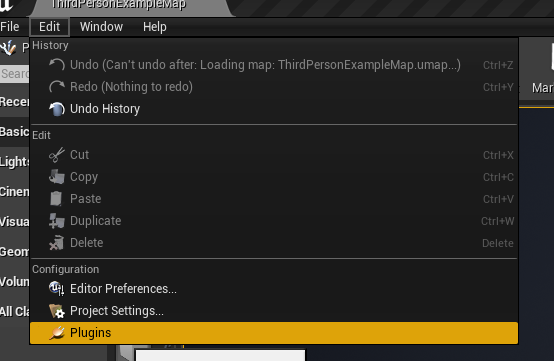
首先，点击界面右上角启动UE4.26引擎。在新项目中选择游戏。在模板中选择ThirdPerson（第三人称）。

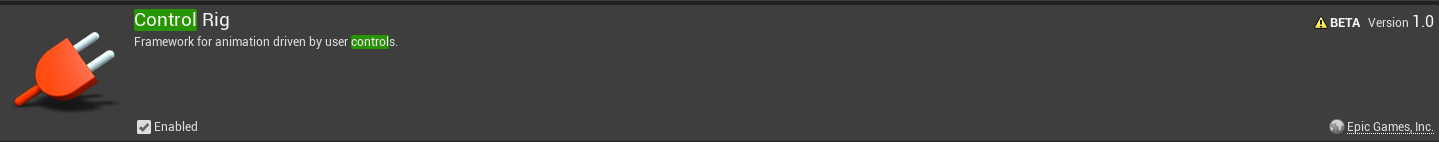




项目设置中，左上角将蓝图切换成C++项目，下面选择路径合名称，点击右下角Create Project。

之后，在打开的项目窗口，选择Edit/Plugins，在里面找到ConrtrolRig插件，选择启用。Control Rig是骨骼控制插件，用于提供读写骨骼位置的接口。然后关闭编辑器窗口。

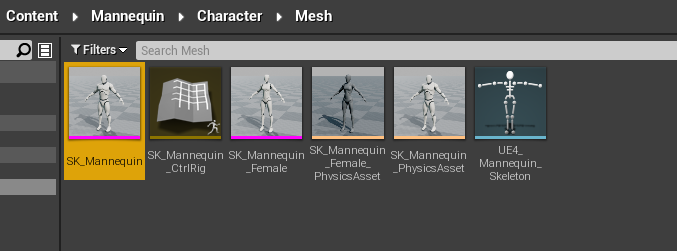




在之后，将附件项目中的Plugins（内含代码和插件配置文件）拷贝到新项目目录下，然后打开项目VS工程文件，编译。完成后再次打开项目文件，即可以使用。



如果想使用在角色身上，首先找到角色Mesh，路径如图所示：



右键选择Create Control Rig，这样就在右边创建了一个用Mesh命名的CtrlRig文件。双击打开该CtrlRig文件，右键打开蓝图选单。在其中搜索ClimbIK和PBIK即可使用，如图所示：

