3D 游戏引擎构架及游戏动画渲染技术分析

文 / 黄竣

在当前科学技术迅速发展的背景下，对技术先进性依赖较强的游戏行业迎来较大的发展机遇，3D 游戏引擎自诞生以来即对游戏行业的发展格局带来较大影响，其当前在游戏领域内得到广泛应用。3D 游戏情节、关卡等核心内容均是建立 3D 游戏引擎基础上，再加上游戏动画渲染技术即可为玩家奉上感官盛宴。基于此，本文研究中将针对 3D 游戏引擎以及游戏动画渲染技术进行分析阐述。

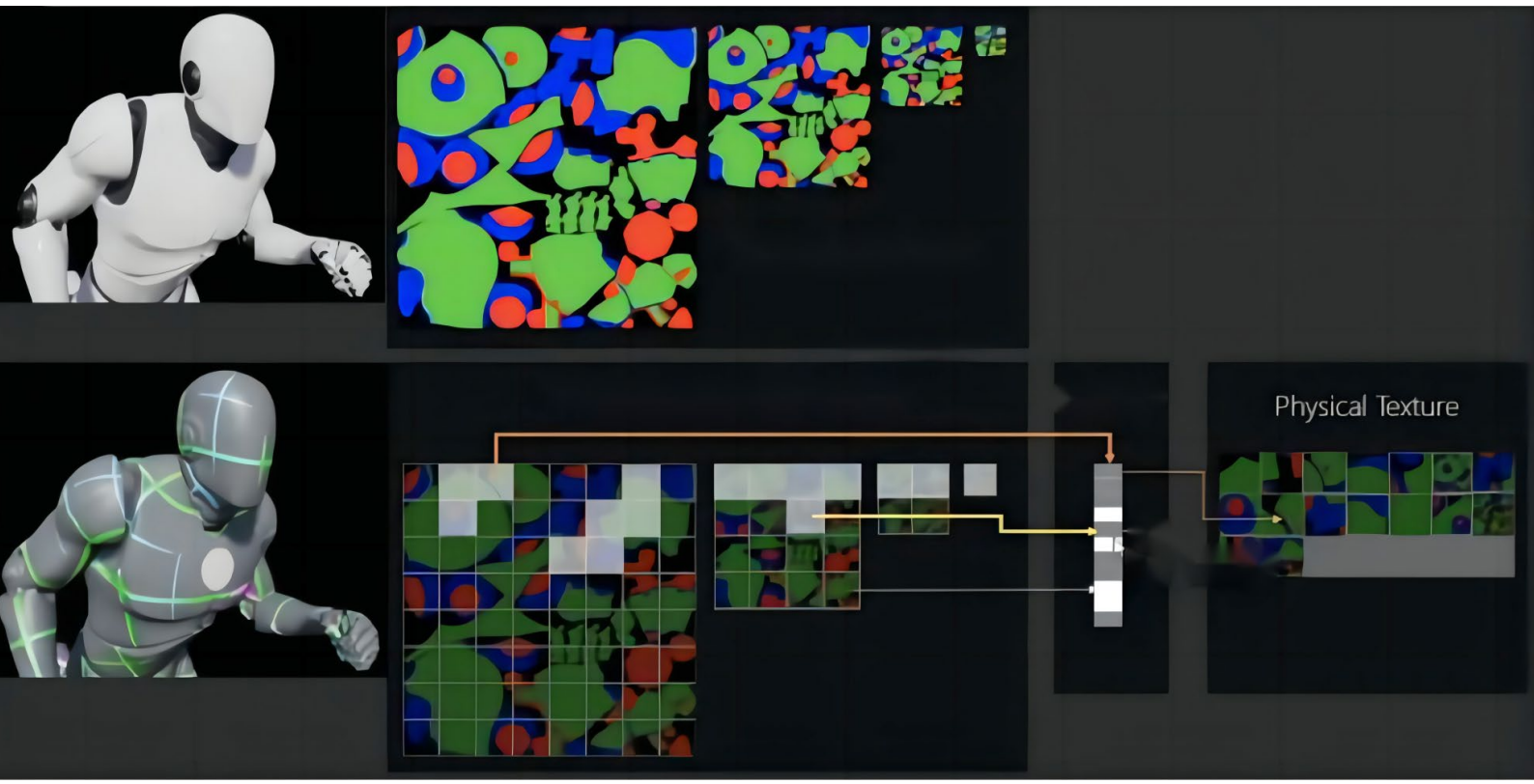

图 1 虚拟纹理映射示意图

从技术层面分析，游戏引擎本质即是基于通用技术细节整理与封装的面向游戏应用的程序接口函数，其主要功能在于替代开发人员完善底层技术实现细节，进而降低开发人员工作负荷并节省开发时间。引擎在实际应用过程中呈现出较为显著的驱动性、完整性以及独立性特征。而游戏场景模块则是游戏渲染过程中所面临的关键环节。场景模块作为游戏引擎核心环节，是游戏开发过程中调用频率最高的模块。为进一步提升游戏的开发效率，保障游戏引擎模块的高隐藏性以及可拓展性，对于加强游戏引擎的研究具有重要现实意义。

3D 游戏引擎构架概述

在当前科学技术高速发展背景下，各种新兴技术为游戏技术升级提供必要支持，3D 游戏即是当前新游戏引擎技术支持下诞生的产物，其与 2D 引擎支持下的游戏最为显著的差异在于游戏画面更加趋于立体化、画面更加精致，游戏中所需功能也有显著提升，比较有代表性的包括静态网络模型、粒子系统、动态、曲面、灯光等。而将相关新增功能进行梳理归纳，可将其划分为系统模块、控制台模块、数据存储模块、游戏接口模块、游戏插件模块以及底层渲染模块等。

系统模块

系统模块是引擎与机器之间进行信息交流的模块，也是在进行平台迁移过程中必须通过代码扩加的模块。其主要功能由输入端、定时端、图形端、配置以及声音 5 个分模组构成，所有分模组都可以在主控系统中被初始化、关闭或更新。其中，输入子模块功能在于统一控制、接收和处理游戏板、键盘以及游戏手柄等输入设备的输入触发，并通过向输入子模块发出请求来获得玩家的位置等信息，进而达到玩家进行自由切换的目的。定时器主要是对通过时间触发器来进行业务变更的支持。图形子模块则是通过 OpenGL、Direct3D 等技术，提取出“图形层”并放在 API 执行层上，从而增强软件兼容性和性能。配置位于顶端，其职责是阅读命令行参数，对记录文件进行配置或对设定进行更改。在系统初始化和正式运行的过程中，配置要求与所有的子模块进行信息互动，包括游戏的载入、声音支持选项、按钮的定义、图片分辨率调节和色彩深度调节都可以使用。声音模块负责下载和播放游戏中的背景音乐以及各种音效。

底层渲染模块

在 3D 游戏引擎架构中，底层渲染模块可以分为：冲突检测和反馈、可视化裁剪、动态几何、静态几何、颗粒系统、成像、节点着色、照明、输出、网格、布告板和天空本体。底层渲染模块是几个单独的工具库所组成，这些工具可以被上层引擎重复使用，并且可以为最基本的呈现函数，通过从控制台获取的信息来调用相应的工具 API 来实现呈现。通常情况下，在使用底层渲染模块时，首先要对其进行包装，然后按照 OpenGL 库、Direct3D 库，从业务呈现提交程序中进行拆解，也可以使用多边形裁剪算法（Sutherland-Hodgman）、基于 Voronoi 区域的冲突算法或基于 BSP 引擎的冲突检测算法，来直接进行处理。

数据存储模块

数据存储模块包含 3D 游戏引擎中的所有数学元素、文件载入器、存储管理器、数据容器等。游戏引擎中主要以曲面、点、矩阵为主要的数学要素。资料储存模块则是负责设定资料的格式，以及资料组织方式。比如游戏文件内数据存储组织、游戏运行阶段内存数据组织、数据读取以及保存历程等。3D 游戏中有大量数据结构类库，它们之间有各自的传输和保存通道。比如，在对游戏运行阶段的内存数据进行组织时，可以利用指针、链表来处理数据结构的类库关系，以此来实现数据快速传递和即时保存。

控制台模块

作为 3D 游戏引擎框架的核心部分，其主要功能是利用命令行、变量调节等功能，实现玩家无需重新启动就可以随意更改游戏引擎设定的功能。就本质层面而言，在 3D 引擎的安装和调试中需要逐个使用控制台模块。比如，在开发控制台过程中，如果要对一系列的命令行变数值进行测试，那么开发者就必须把这些数据输入到控制台，进而实现对可变参数的快速检测，提高系统开发效率。在 3D 游戏引擎运行过程中如果发生问题，则可以通过简单的控制“disable”的方式来解决，而不用马上退出程序，从而大大提升系统调试效率。

游戏接口模块

游戏接口模块大多是 3D 游戏引擎、游戏开发人员的连接端口，开发人员可以通过这个界面，使用 3D 游戏引擎的部分或全部功能。比如，游戏接口模块可以为游戏渲染模块的每个子模块提供接口，从而与系统相关属性配置模块（设置、方向和位置）连接，进而根据 3D 游戏引擎每个部分的动态属性进行灵活改变。同时，游戏接口模组还负责将操作界面和系统界面进行连接，方便操作界面模组对输入操作、音效播放、摄像等进行实时修正。通常情况下，游戏接口模块大多采用的是对象 + 图形界面的形式，还可以利用关卡编辑器来设计一种逻辑脚本语言，以此方便开发人员在更高层上编写游戏事件与方法。

3D 游戏渲染技术概述

游戏动画渲染技术，就是使用类手工绘制的图形来替代计算机绘制的图形，从而达到去真实感的目的。通常情况下，需要由专业人士使用专用的动画渲染着色器来使游戏图像、动画以及漫画达到与动画相似并且简洁清晰的效果。游戏动画绘制是图像技术发展的“副产物”，其可利用 .FX 文件对游戏关卡进行灵活渲染，使实际画面与游戏运行阶段完

全一致，避免由于绘制软件频繁切换而造成的画面帧率降低等问题。也可以将前期制作的呈现效果库作为依据，通过添加 C++ 代码来引用呈现程序模块，将呈现程序中的参数编码到模型、纹理以及顶点中，从而提升呈现速率。通过对相关技术进行梳理，渲染三维场景涉及以下基本步骤：

第一，描述虚拟场景，通常是根据以某种数学形式表示的 3D 表面。

第二，虚拟相机被定位和定向以产生所需的场景视图。通常，相机被建模为理想化的焦点，成像表面悬停在它前面的一小段距离，由与目标显示设备的图像元素（像素）对应的虚拟光传感器组成。

第三，定义了各种光源。这些光源提供所有光线，这些光线将与环境中的物体相互作用并从中反射，并最终到达虚拟相机的图像感应表面。

第四，描述了场景中表面的视觉属性。这定义了光应该如何与每个表面相互作用。

第五，对于成像矩形内的每个像素，渲染引 | 擎计算通过该像素会聚在虚拟相机焦点上的光线的颜色和强度。这称为求解渲染方程（也称为着方程）。

纹理映射技术

纹理映射技术是当前应用频率较高的 3D 游戏动画渲染技术，其也是近年来发展最为快速、使用最为广泛的技术，被广泛地用于 3D 真实感图像的生成和显示。3D 游戏动画渲染技术中的纹理映射技术可以为制作真实感图像带来许多便利条件，不需要游戏设计者在物体表面细节处理上耗费过多时间，通过对 3D 对象进行二维参数化，让游戏用户可以体验到更为真实的游戏。在进行 3D 对象的二维参数化时，首先要在 3D 对象表面上选取任意一个二维参数值（u, v），再对二维参数值（u, v）进行分析得到其质地值，最后在 3D 图形表面上生成质地图案。在平滑表面上加入纹理图形时，最关键的问题在于如何将其转换成简单坐标系，从而为平滑表面纹路设计提供更好的途径。在进行纹理图案设计时，通常会有两个映射，一类是从纹理空间到景物空间的转换，此方面属于曲面参数化问题。另外一类是从场景到屏幕转换，其也被称为取景转换。部分情况下，开发人员在实际设计纹理图案过程中会将两种映射方式合并为一个映射变化。

面缓存技术

面缓存技术也是 3D 游戏动画渲染技术的一个重要环节，其在 3D 游戏设计中起着非常重要的作用。面缓存技术主要是存储纹理光照映射计算结果，当一个游戏房间带映射纹理数量达到 10 个，而玩家又不会走出游戏房间，此时就必须对相应纹理光照映射进行运算，通过运用高新计算机技术将运算的成果进行缓存，以备下一次运算时所需。如此在进入下次游戏时，因为先前图像亮度图的计算结果被缓冲，所以不需要再次进行图像亮度图的计算，同时其也会随着情节发展而不断地进行运算更新。

3D 动画渲染技术要点

Unity 参与制作的 CF 大动画

CF 动画是一款中文名字叫《穿越火线》的 FPS 游戏，其中某些环节与内容都是由 Unity 引擎来完成的，通过 Unity 来完善传统的动画流程，从而建立起一套完整的动画流程。与传统动画制作时对特效、角色以及动画模块进行划分的模式相比较，Unity 引擎所参与制作的 CF 动画属于数字资源的组织，Unity 二进制文件在同时实现方便操作的难度比较大，此时就必须从提高效率的角度来考虑，贯彻平行分工原则，由导演、执行导演分别使用 Cinemachine、Timeline 制作镜头以及白模 Layout 场景。在“向外射击”这一幕中，把“目标”“绑”在一颗子弹上，确保了画面随着子弹平稳地移动。当高模进入最后阶段时，由于体型、关节点已经确定，因此可以绑定 3DMax 拓扑低模，进行 UV 划分，并由材料师在 Unity 中模拟布料，进行 ID 贴图，再由地编师利用 Unity 将各种主界面、副本场景打包，将其放入 Loading 图中，使一系列游戏对象在 Hierarchy 层级窗口中显示，完成初始场景的搭建，并分层输出序列帧。比如，在海港的镜头中，就需要将时间线分成三个等级，分别是“组员”、“组长”和“导演”，来达到镜头的整体效果。同时，选取拍摄场景的主光，分割区域，并装载多区光的统一场景。其具体流程如图 3 所示。

基于 Cartoon CG 卡通渲染的 3D 游戏动画渲染技术应用要点

Cartoon CG 卡通特指，由计算机和 3D 模拟引擎制作而成的三维动画短片，其特点是画面华丽、场景逼真，比如暴雪公司制作的《魔兽争霸 3》和《星际争霸 2》开场 CG。在 CG 动画制作过程中，要使用 3D 动画软件进行虚拟角色装载，在实现完美手绘表达的同时，通过卡通渲染、材质贴图的方式将三维立体影像渲染为单一颜色、线条展现的二维画面，减少手工绘制工作量。最后，再对绘制出的线段进行人工修正，从而得到具有仿手绘风格的个性化图形。Cel-Shading 等具有艺术风格的光线渲染算法是 Cartoon CG 中常见的工具，其可以减少色阶数量，用单调平面色彩（白灰双色阶）来取代具有丰富色彩的渐变色彩。在此基础上，拟采用法线贴图等多层梯度采样贴图替代编码的振幅色彩，进一步丰富 UV 图像的细节信息，并采用分片光照加权（单位法矢量，单位光源方向矢量）来实现 UV 图像的采集。以此为基础，利用贴图技术对纸张进行贴图处理，以得到具有类似漫画书、卡通效果的纸张贴图。以街头篮球题材的 Cartoon CG 动画为例，针对其动感、影视视听表达和手绘动画等特性，开发人员可首先采用 Low-polygon Model 对低面数模型进行建模，将“面”

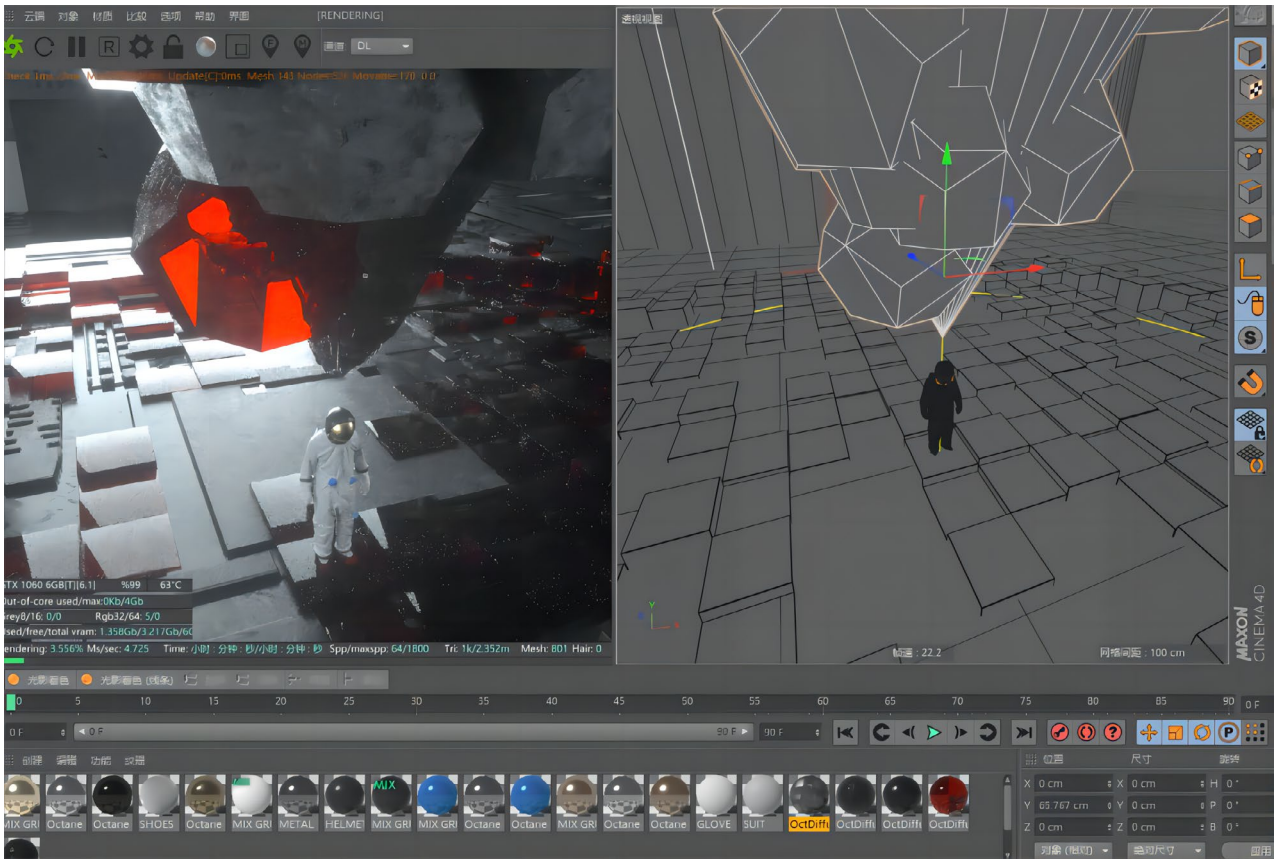

图 2 静帧渲染

的数量尽量压缩，实现对其形态结构的精确刻画。其次，利用天然运动规律对角色骨骼进行调整，使其具备做出显示无法达成的夸张动作能力。最后，采用 Cel-Shading 绘制技术，结合 3D 电脑软件中的虚拟摄像机位置和视点操控，为模型赋予动画材料，在模型上增加合适的粗细边缘，绘制出高质量的动画场景，突出“线”的艺术性。

目前，除 Low-polygon Model、Cel-Shading 绘制技术之外，在 Cartoon CG 动画渲染中，Unity 也是比较常用的引擎工具。其实际应用过程中，首先要通过使用步长函数或者 RampMap 映射对兰伯特照明进行修正。其中，步进函数对照明模式的修正主要是采用直接给出值方法，实现照明效果在两个已知值间的平稳过渡。给定数值标准是，以相应数值为参照，大于该数值则给定为 1，反之则给定为 0；而 RampMap 贴图方法，就是把兰伯特灯光作为映射，然后通过 ramp 图表中的颜色转换（LUT 查询表）来获取相光照。其次，通过对光照模型进行修正，利用 Unity “ShaderGraph”算法对目标点进行计算，得到底层的边缘光。普通动画的基本边线有软、硬两层边线线相互交织的趋势。所谓深度偏移，就是在移动模型顶点之后，再进行一次样本深度的调整，让模型的深度偏离原来的深度，达到一定的数值范围（边界），与《原神》中角色边缘亮边系统相似。第三，通过对 NdotL 值控制，实现像素色彩改变，并在法线 N 的范围内对 NDotL 进行二值化阴影、毛发投影和人脸阴影等运算，将这些运算与 NDotL 结合在一起。同时，在动画渲染中加入高光、毛发和线型光，确保色彩混合效果。最后，采用 Unity 中的“Sobel 算子”对图像进行边缘提取，在提取出图像中存在的冗余信息后，再对图像中不同颜色块中的单色区域进行填充，从而得到清晰图像边界。另外，由于动画绘制的特征牵涉到很多的非物理特性，因此在具体的应用中，要根据项目具体情况来选择不同特征参数。

综上所述，在当前科技高速发展的背景下，对高新技术依赖性较强的游戏行业要积极抓住机遇，大力推进游戏引擎技术发展，为游戏项目开发奠定坚实基础，技术人员在实际工作中应继续加强对引擎中五大子模块的开发研究力度，同时充分认识到渲染的重要作用，实际工作中积极结合游戏类别以及其他因素差异，选择适配 3D 游戏引擎框架的渲染技术，切实保障游戏项目开发质量，同时也为推动行业健康可持续发展奠定坚实基础。（本文作者为上海剑圣网络科技有限公司项目总监）科