3D 建模技术

第 3 章

CHAPTER 3

3.1 物体建模技术

虚拟世界通常由 3D 场景构成，需要运用 3D 建模技术对虚拟环境和角色进行建模。3D 建模是对现实中的对象或场景进行仿真的过程，对 3D 物体建模是构建整个虚拟世界的基础。

3.1.1 几何建模

几何建模是对物体几何信息的数字化表示。几何信息是指物体在欧式空间中的形状、大小和位置，其中最基本的几何元素是点、线、面。物体的几何建模是在虚拟环境中对物体的形状和外观进行建模，物体形状可由多边形、NURBS 曲线、边和顶点等确定，外观则由纹理、光照等确定。

1. 形状建模

形状建模通常采用如下方式实现。

1) 人工建模方法

基于人工的建模方法需要用户使用相关 3D 建模软件创建物体的 3D 模型，如有需要，还要具有一定的编程能力，故此对用户的要求较高。根据用户所需掌握的技能，可分为以下两类：

(1) 利用交互式建模软件进行物体建模，如 AutoCAD、Autodesk 3ds Max、Maya 等。用户可以在这些软件的可视化界面中进行交互式建模。这类方法提供便利的交互界面，能够快速对复杂物体建模。

(2) 利用现有的 3D 图形库通过编程的方式进行物体建模，如 OpenGL、Java 3D、VRML、DIRECT3D 等。这类方法可以从点、线、面对物体进行建模，特点是编程容易、效率较高、实时性较好，缺点是对复杂几何外形的物体建模能力不足。

2) 自动建模方法

当前自动化建模方法很多，最为常用的是利用 3D 扫描仪建模和基于图像的 3D 建模。

3D 扫描仪是人们专门为自动化建模发明的设备，它通过收集现实世界中物体的形状等信息对物体进行 3D 重建，完成该物体在虚拟世界中的数字化。3D 扫描仪可以分为两类：接触式和非接触式。接触式扫描仪需要与被扫描物体直接接触。该类扫描仪出现时间较早，虽精度较高，但体积巨大、造价高，如图 3.1(a)所示。非接触式 3D 扫描仪，顾名思义，指

不需要直接接触物体的 3D 描仪，通过激光、结构光等来收集被扫描物体的表面信息，如图 3.1(b)所示。

(a) (b)

图 3.1 接触式 3D 扫描仪和手持式非接触式 3D 扫描仪

3D 扫描仪以其高精度的优势得到广泛应用，但由于扫描设备空间容易受到场地限制，其传感器容易受到噪声干扰等因素，所以其使用范围受到一定限制。

基于图像的 3D 建模技术，是计算机图形学、计算机视觉和机器学习领域专家长期研究的一种 3D 建模方法。尤其是自 2012 年以来，得益于深度学习技术的发展，利用卷积神经网络进行基于图像的 3D 建模引起了广泛的关注，并取得了瞩目的成绩。基于图像的 3D 建模方法是从一幅或多幅二维图像中推断出物体的 3D 几何结构。与 3D 扫描仪相比，它只需使用普通的相机拍摄物体的单视角或多视角的照片，经过基于深度学习的 3D 重建网络，可以获取物体精准的 3D 模型，如图 3.2 所示。

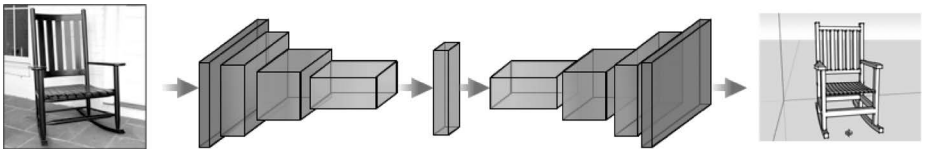


图 3.2 基于图像的 3D 物体建模示例

2. 外观建模

在 VR 场景中，为保证虚拟物体的真实感，除了精准的形状模型外，还需要处理物体模型的外观形态。虚拟物体外表的真实感主要取决于表面纹理、透明度反射、折射等特征。在 VR 系统中，纹理映射技术被广泛应用在外观建模中。纹理映射(Texture Mapping)，又称纹理贴图，是将真实世界中物体的纹理映射到虚拟世界 3D 物体的表面的过程。如图 3.3 所示，通过将植物纹理映射到 3D 植物模型上，可以实现外观颜色丰富的虚拟物体。

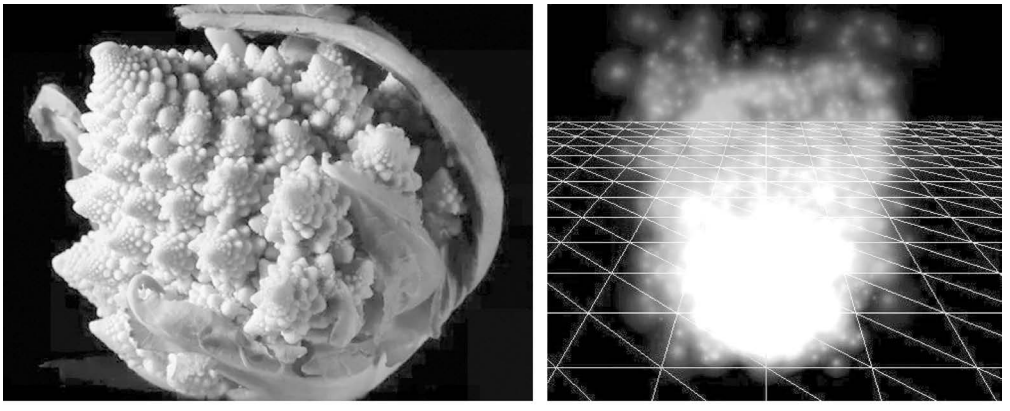
图 3.3 纹理映射

3.1.2　物理建模

VR 系统需要给用户沉浸式的逼真体验，在虚拟环境中，除了构建物体逼真的表面，还需要考虑物体的物理特性，如物体的质量、惯性、材质、硬度等。例如，用户在虚拟环境中观察到逼真的物体是几何建模的体现，而通过接触物体，能够感受其重量、软硬程度，这就是物理建模的体现。构建一个逼真的 VR 系统，需要将这些物理特性与几何建模相融合，这涉及计算机图形学和物理学领域知识。下面介绍两种典型的物理建模方法：分形技术和粒子系统。

1. 分形技术

分形技术可以用来描述具有自相似特征的物体，该类物体可以分成数个部分，且每一部分都（或者至少近似地）是整体缩小后的形状。自然界中，存在许多有着自相似特征的物体，整体复杂，而组成它们的部分近似于整体的缩小版，在各个尺度下都显得相似，如云、山脉、闪电、海岸线、雪片、植物根、多种蔬菜（如花椰菜和西蓝花），以及动物毛皮的图案等。如图 3.4(a)所示为具有螺旋纹路的西蓝花。在 VR 世界中，分形技术可以用来建模分形物体，通过无限递归生成复杂的不规则分形物体的建模。然而由于计算量太大，分形技术在 VR 中一般用于静态远景的建模。



(a)　　(b)

图 3.4　螺旋纹路的西蓝花和利用粒子系统建模的火焰

2. 粒子系统

粒子系统是一种典型的物理建模系统，由大量的粒子构成，用来表示或模拟某些复杂的现象或场景。这些粒子具有质量、位置、速度、颜色、寿命等属性，粒子系统初始化时会随机生成一些状态，之后根据模拟对象遵循的物理规律，计算更新粒子的状态，当一些粒子完成生命周期，便销毁这些粒子。在虚拟世界中，通常用粒子系统描述火焰、水流、雨雪、旋风、喷泉等现象。利用粒子系统建模火焰的效果如图 3.4(b)所示。

3.2　场景建模方法

观看视频

3.2.1　场景建模方法简介

场景建模旨在通过获取到的场景 3D 空间信息，结合计算机技术及数字化建模技术，最

终构建用于 VR 交互的场景,或为 VR 应用提供场景的数字化感知。场景建模主要面向 3D 应用环境,技术关键点在于面向 3D 空间信息的数字化和模型化技术。

场景建模技术的难点和关键在于,处于复杂环境下,针对环境中物体的空间几何属性、表面纹理属性、物体运动数据、非刚体形变数据等关键信息,如何获取并进行具备真实感的数字化模拟。在具体的应用上,各项属性的获取技术和数字化模拟技术往往与应用的需求场景密切结合,并随着硬件技术的进步而发展。而在这些技术中,物体空间几何位置关系的获取技术是最为核心的,而基于测量方式的不同,可以分为广泛使用的光学测量技术和更贴合专业应用的非光学测量技术。

随着技术的进步,光学测量技术已经广泛运用于 VR 应用等涉及场景和物体建模以及环境数字化理解的应用中,并可以根据光学信息的获取方式,进一步划分为主动光学扫描技术和被动多视角立体视觉技术。而对于作为辅助的非光学测量技术,则主要是通过电磁波、超声波等非光学介质,并使用相应的传感器等空间信息获取设备获得场景物体的空间位置关系。在应用中,非光学测量技术常与光学视觉纹理等多模态信息进一步结合,构建具有真实感的虚拟场景,例如在医学、自动驾驶等更为专业化的应用领域,一般会采用如 MRI、CT 等医学影像信息或 GPS 定位技术获得的地理坐标信息,实现对于空间位置信息更为精确的感知。

3.2.2 主动光学扫描技术

光学扫描是最为直接且精度较高的场景空间几何属性获取技术,部分光学扫描技术的精度已经能够小于 2mm,其中激光扫描技术和结构光技术较为成熟。在应用中不同精度和测量范围的激光扫描设备常与摄像机结合,实现空间关系和色彩纹理的进一步组合。而主动光学扫描技术的缺点在于容易受到物体表面反光的影响(见图 3.5)。

图 3.5 激光扫描仪对雕塑进行非接触性建模

主动光采集原理是基于飞行时间测距法(Time of flight,ToF),依靠主动光源设备将光线照射到采集对象上,进而根据相应反射光线的传播时间计算出主动光源与目标物体的距离,并以此获得场景内物体的空间位置信息。随着技术的进步,基于飞行时间的测量技术已被广泛应用于 VR 应用中的场景感知和建模应用中(见图 3.6)。

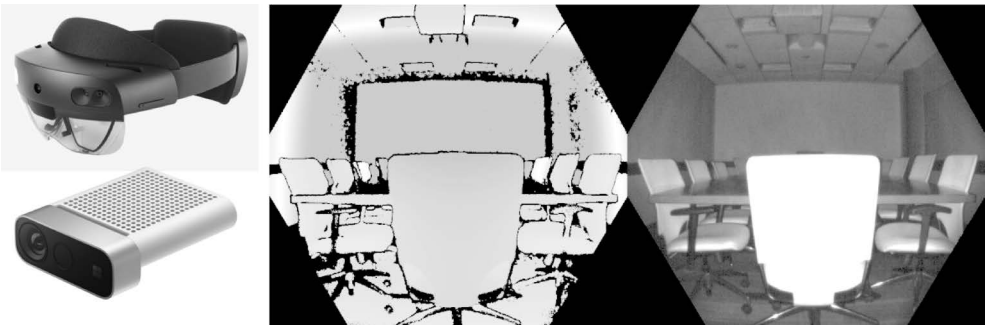


图 3.6　HoloLens2 与 Azure Kinect DK 使用 ToF 传感器进行会议室场景深度感知

结构光编码测量技术主要涉及对于光场在空间和时间上的调制，根据编码方式的差异可分为时域编码和空域编码，前者主要对光场的时间和频率进行调制，后者则主要对光场的振幅、相位和偏振进行调制，最终获得具有特定结构的特殊光线。结构光测量技术主要通过将特定的结构光信息投射到场景物体表面，之后通过摄像头采集获得相应结构光信号的位置变化和畸变，最后基于光学三角测量法获得物体的空间位置信息和表面结构信息。

3.2.3　被动多视角立体视觉方法

被动多视角立体视觉方法主要通过被动获取的多视角图像来构建空间位置关系。其中最为常用的是立体视差法，根据三角测量原理，利用不同视角图像内对应点的视差可以计算视野范围内测量点的 3D 位置信息(见图 3.7)。计算流程利用多个视角投影面信息和空间位置信息，利用采样点在各个视角投影内的坐标信息构建视线，最后通过计算视线的交点获得相应采样点的空间位置信息。立体视差法的优点在于能够同时获取空间位置信息和颜色纹理信息，但是对于弱纹理或重复复杂问题场景则容易配对困难。

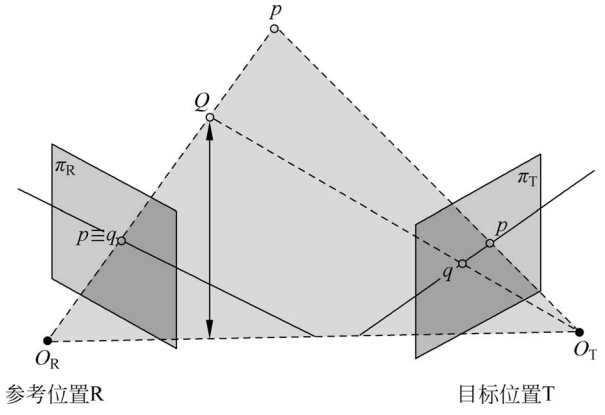


图 3.7　双视角立体视觉

3.2.4　其他场景测量方法

在具体应用中除了基于光学的场景测量和空间感知技术之外，通常结合专业测量技术为 VR 中的场景感知应用提供更为丰富且可靠的空间信息和纹理信息。例如在医学手术和

自动驾驶场景中,场景空间的建模和数字化理解不仅使用光学信息,而且一般需要结合用特定的专业设备(例如电磁导航设备、GPS 等)获取的空间信息。

在医学手术场景下,纯粹的术中视频并不能很好地反映病人体内的情况,而临床中则会使用如 CT、MRI、超声等医学造影数据来获得病灶的内部结构信息。通过将医学造影获得的关键组织空间信息与术中视频结合,构建基于 AR 技术的手术导航系统,能够有效地辅助医生进行手术操作,目前已经在部分手术场景上进行了临床实践(见图 3.8)。

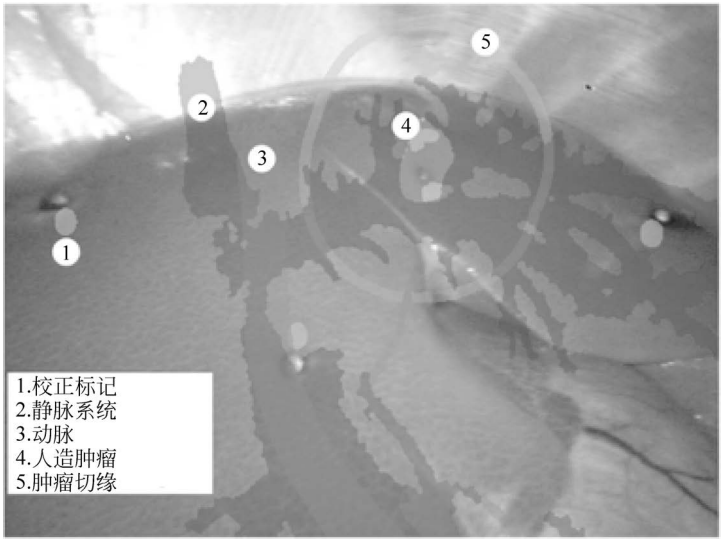


图 3.8 使用医学造影数据构建的 AR 手术导航系统

在自动驾驶场景中,除了多视角的摄像系统和激光雷达系统这些光学测量仪器之外,也会结合 GPS 信息来获取车辆自身的地理空间位置和运动信息,系统化地构建车辆周围的车辆、人员等位置信息,实现对于驾驶场景更为精确的数字化感知和建模,以支持自动驾驶算法的计算精度需求(见图 3.9)。

图 3.9 KITTI 自动驾驶车辆系统

3.2.5　多模态信息融合应用

在 VR 场景建模应用中，空间信息的获取会涉及不同模态的数据，而将不同模态的空间信息进行有机融合，能够实现许多贴近生活和自然场景的建模应用。例如智能手机上的房间建模应用，可以通过将摄像头视觉信息、陀螺仪空间位置信息、激光测距信息相结合，实现对于房间的高效建模和全景可视化，还可以结合构建的家具模型，实现对房屋装饰布局的模拟（见图 3.10）。

而在地图应用方面，也常将高精度 GPS 信息、多视角影像信息、空间遥感信息等结合，实现对于广场、大型建筑物的虚拟化建模，甚至使用更大规模的设备实现街道、城市、地形地貌的真实感建模，为用户提供传统二维地图之外的 3D 可视化场景（见图 3.11）。

图 3.10　3D Scanner 应用获得的房间建模

图 3.11　建筑物倾斜摄影建模

3.3　角色建模方法

观看视频

3.3.1　角色建模简介

VR 技术结合了人工智能、人机交互和计算机等相关技术。随着不断更新和飞速发展，VR 技术被广泛应用于工业、军事、医药、娱乐等诸多领域。其中游戏、电影以及广告创作领域，几乎离不开角色建模技术。在计算机技术发展、VR 技术支撑和影视娱乐行业需求增长的背景下，角色建模技术受到领域内越来越多的关注。

“角色”一词的涵盖范围十分宽广，通常是人、动物、机器人或神话生物。如在游戏中，角色可以是主角（如游戏玩家可控制的角色），也可以是次要角色（如与游戏玩家互动的角色），甚至可以单独存在于游戏外围。角色建模是将概念转变为 3D 模型的多阶段处理过程，第一步是构建模型的基础网格，通常从一个立方体开始，然后添加和减少多边形来创建所需的形状。一旦基础网格完成，就可以开始添加细节，如皱纹、疤痕、文身等。模型完成以后，就可以对它进行纹理处理，给模型添加颜色和阴影以使它看起来更逼真。3D 艺术家通常使用各种软件和工具来创建角色模型。

3.3.2　角色建模技术

当前角色建模主要利用 4 种技术来创建 3D 角色，即多边形建模、非均匀有理 B 样条建模、3D 雕刻建模和 3D 扫描建模。

1. 多边形建模

多边形建模也称 Polygon 建模,是指用多边形来表示物体的 3D 表面建模方式。多边形由基于顶点、边和面的几何体组成。其中顶点(Vertex)即指线段的端点,是构成多边形的最基本元素。边(Edge)指示一条连接两个多边形顶点的直线段。面(Face)就是由多边形的边所围成的一个面。法线(Normal)表示面的方向,法线朝外的是正面,反之是背面。顶点也有法线,均匀和打散顶点法线可以控制多边形的外观平滑度。

多边形建模快速可编辑的特点非常适合对精确性要求不是很高的物体进行建模,广泛用于视觉表现、影视,以及交互式视频游戏中的角色内容开发。利用多边形进行角色建模时,首先使一个对象转化为可编辑的多边形对象,然后通过对该多边形对象的各种子对象进行编辑和修改来实现建模过程。多边形建模可以进行复杂和简单的 3D 角色设计,并允许对形状进行控制。然而,多边形模型的一个缺点是它们可能需要大量的多边形来准确地表示物体的几何信息,甚至是简单的形状。

2. 非均匀有理 B 样条建模

非均匀有理 B 样条(Non-Uniform Rational B-Splines)缩写为 NURBS,也称为非均匀有理 B 样条曲线,由肯·弗斯普瑞尔(Ken Versprille)提出。具体解释如下。

(1) Non-Uniform(非均匀性):指一个控制顶点的影响力范围能够改变。

(2) Rational(有理):是指每个 NURBS 物体都可以用有理多项式来定义。

(3) B-Spline(B 样条):是指用路线构建一条曲线,在一个或更多的点之间以内插值来替换。

NURBS 是计算机图形学中常用的数学模型,用于产生和表示曲线及曲面。它为处理解析函数和模型形状提供了极大的灵活性和精确性。NURBS 建模的基础是一系列连通的控制点,这些控制点可以看作曲线或曲面的控制点。通过移动这些控制点,可以改变曲线或曲面的形状。NURBS 适合创建光滑的物体,能有效地控制物体表面的曲线度,从而创建出逼真、生动的角色模型。但是在这种由数学公式设置的模型中,个别部分很难编辑,无法在不破坏整个模型完整性的情况下完成(见图 3.12)。

图 3.12 基于 NURBS 的角色建模

3. 3D 雕刻建模

3D 雕刻也称数字雕刻,是通过专业软件如 ZBrush 或 Maya 中的“捏拉”方法创建 3D 模型的过程,被艺术家广泛用于游戏和动画电影中,适合创建形状自然且线条圆润的超现实角色内容。

3D 雕刻过程和用黏土或石头等真实材料进行雕刻非常相似,建模过程中首先使用类似刷子的雕刻工具处理多边形网格,推、拉、扭转各个几何部分,增加额外的几何部分来模仿自然结构(见图 3.13)。与多边形建模相比,数字雕塑需要更多的艺术技巧,而且需要格外仔细,也更耗时。所以在多数情况下,这些方法会同时使用:首先对物体进行建模,然后发送给 3D 雕塑家进行细节处理,完成最终造型。3D 雕刻容许对最终产品有很大的控制权,艺术家可以对雕塑进行小的修改,而不必从头开始,从而节省大量的时间和精力。

图 3.13　基于 3D 雕刻的角色建模

4. 3D 扫描建模

3D 扫描建模技术通过记录真人面部表情数据并进行编码采集，再应用到 CG 角色中，为角色注入活力。3D 扫描建模技术不仅提供了高度的准确性和精确度，还提供了高度的灵活性。例如，当需要在创建模型之后对其进行更改，可以简单地重新扫描对象并进行必要的更改。

3D 扫描采用的技术不同，扫描过程也有所不同。常见产品类型包括结构光 3D 扫描仪、激光三角测量扫描仪、时差测距激光扫描仪等。现在的一些智能手机和平板电脑也可以作为扫描仪使用，这要归功于这些移动设备内置或附加的传感器。但无论使用何种技术，3D 扫描仪的最终效果都是一样的，即生成真实物体的 3D 模型，从而可用于多种场景，比如 CAD、逆向工程、质量检测、遗产保护、CGI 等（见图 3.14）。

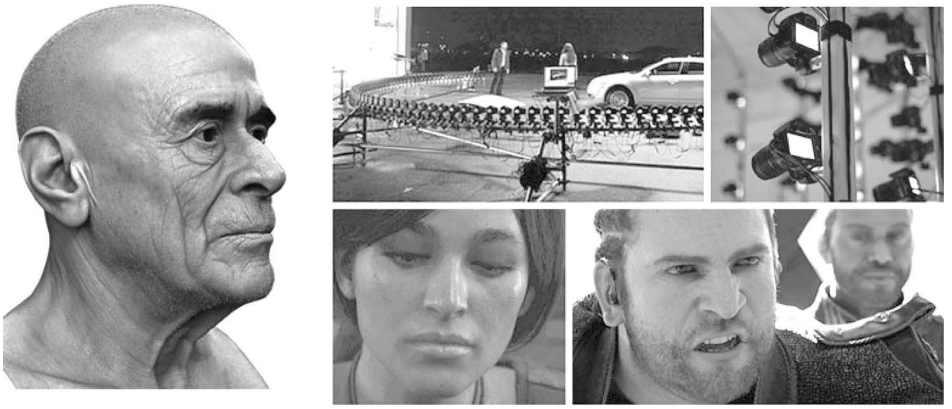


图 3.14　基于 3D 扫描的角色建模

3.3.3　角色建模软件

角色建模时根据需求选择合适的 3D 建模软件是非常重要的。目前市面上常用的 3D 角色建模软件有 Maya、3ds Max、ZBrush 和 Blender 等。

1. Maya

Maya 是世界顶级的 3D 动画软件，被视为 CG 的行业标准，拥有一系列齐全的工具和功能，擅长建模、纹理、灯光和渲染。它具有极其庞大的功能，包括粒子系统、头发、实体物理、布

料、流体模型和角色动画。它功能完善,工作灵活,易学易用,制作效率极高,渲染真实感强。

2. 3ds Max

3ds Max 是一款基于 PC 系统的 3D 动画渲染和制作软件,它强大的功能和灵活性是实现创造力的最佳选择。它拥有非常强大的 3D 建模工具,如流体模拟和毛发,以及角色操纵和动画。与 Maya 一样,它能使用直接操作和程序两套建模技术。

3. ZBrush

ZBrush 是一款专业数字雕刻、绘画软件,多被用在次世代美术的设计中。它以强大的功能和直观的工作流程彻底改变了整个 3D 行业。可以说,它的出现为 CG 艺术家提供了世界上最先进的 3D 工具。

4. Blender

Blender 是一款免费开源的 3D 图形图像软件,提供了从建模、动画、材质、渲染,到音频处理、视频剪辑等一系列制作动画短片所需的解决方案。Blender 拥有多种用户界面,方便多种工作场景,而且内置绿屏抠像、摄像机反向跟踪、遮罩处理、后期节点合成等高级影视解决方案。

3.3.4 角色建模流程

简而言之,角色建模就是 3D 艺术家根据角色设计师设计的角色原画,通过 3D 软件制作出原画的 3D 角色模型。具体而言,首先根据原画建立初始模型,再将模型进行高精度模型(高模)的雕刻和细节的优化。高模细节多,面片数多,对系统设备性能和引擎算法要求高,从而产生低精度模型(拓扑低模)的概念。低模完全符合各项要求和布线规则,而高模的作用就是通过 UV、烘焙和贴图把高模的细节投射到低模上达到更加极致的视觉效果。最后再对角色模型进行骨骼绑定和蒙皮操作,实现角色模型的操控,生成动画。

角色建模流程具体包含以下 5 个阶段。

第一阶段:角色模型设计

好的角色造型不仅要有视觉上的美感,还需要生动有趣,富有性格特征。因此在设计角色造型时,要将角色的性格特征通过外部造型表现出来。为了设计角色概念,美术师需要寻找概念创作、角色灵感、研究角色起草的来源。通常情况下,他们会先制作一个情绪板,然后画出角色模型的草图,包括主要的身体、面部特征和角色的轮廓。可以从简单的 2D 绘图开始,也可以在 3D 建模软件中绘制草图,包括轮廓和所有主要特征。完成轮廓后,将角色模型的几何图形放置在 X、Y 和 Z 平面上,获取角色模型正面图、侧面图/剖面图、背面图和俯视图(见图 3.15)。

第二阶段:角色建模

角色概念设计完成后,实际的建模过程就开始了,这是角色建模最关键的一环,主要包括网格模型、3D 雕刻和重拓扑,分别对应角色的中模、高模和低模表示。

1) 网格模型(角色中模)

建模的第一步是创建角色的基本 3D 网格。艺术家们可以用任何 3D 角色建模软件来完成,如 Maya、Blender 或 MakeHuman。创建一个基本的网格并不困难,但需要一些关于如何使用所选软件程序的知识。可以使用立方体、圆柱体或任何其他类型的基本几何对象创建角色模型的整体外形(见图 3.16)。

图 3.15　角色概念设计

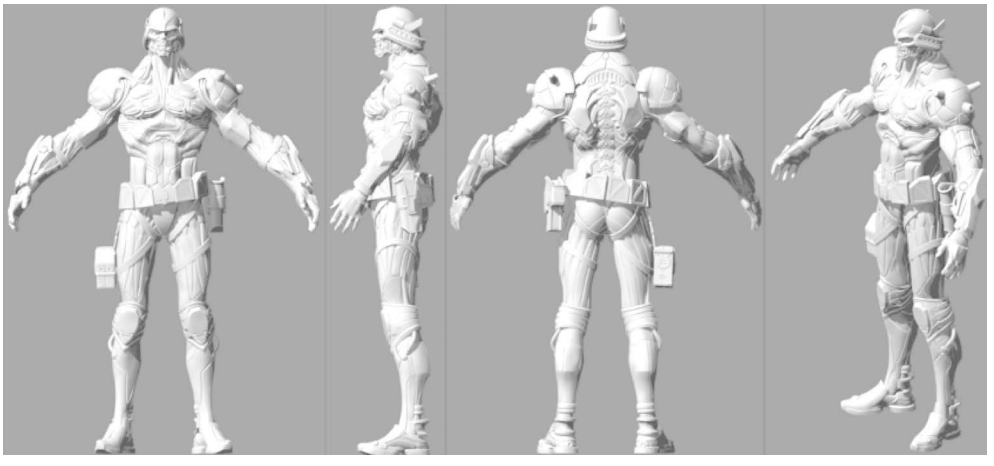


图 3.16　角色建模

2）3D 雕刻（角色高模）

在 3D 雕刻阶段，艺术家将基础模型输入到 3D 雕刻软件，如 ZBrush 中，利用类似刷子的工具操纵角色模型的多边形网格，逐步刻画出模型的细节，如头发、皮肤、皱纹等。通过雕刻，设计师可以在主体的纹理上达到令人难以置信的细节水平。由于网格很复杂，因此雕刻后需要重新拓扑，然后在后续程序中被使用（见图 3.17）。

3）重拓扑（角色低模）

重拓扑是在雕刻的模型基础上重新建模。由于雕刻的面数太高，很难应用在游戏与影视中，根据雕刻的模型细节然后重新建模即为重拓扑。拓扑低模有利于快速调整角色模型网格，实现更好的编辑。以游戏为例，越高的面数就会更加消耗计算机的硬件资源，也就是运行速度越慢。要保障游戏能够流畅地运行，同时保持丰富的细节，重拓扑显得尤为重要（见图 3.18）。

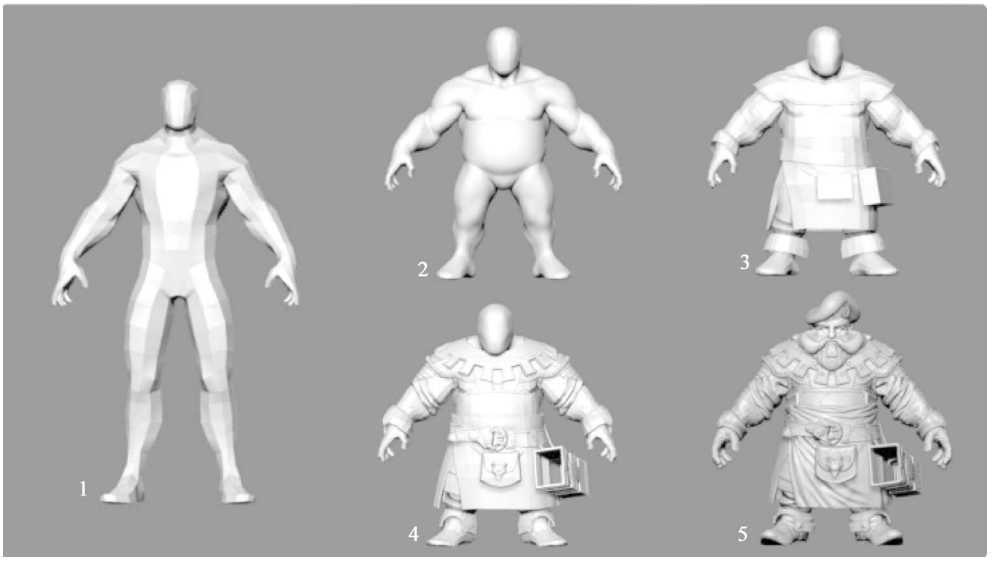


图 3.17　角色雕刻

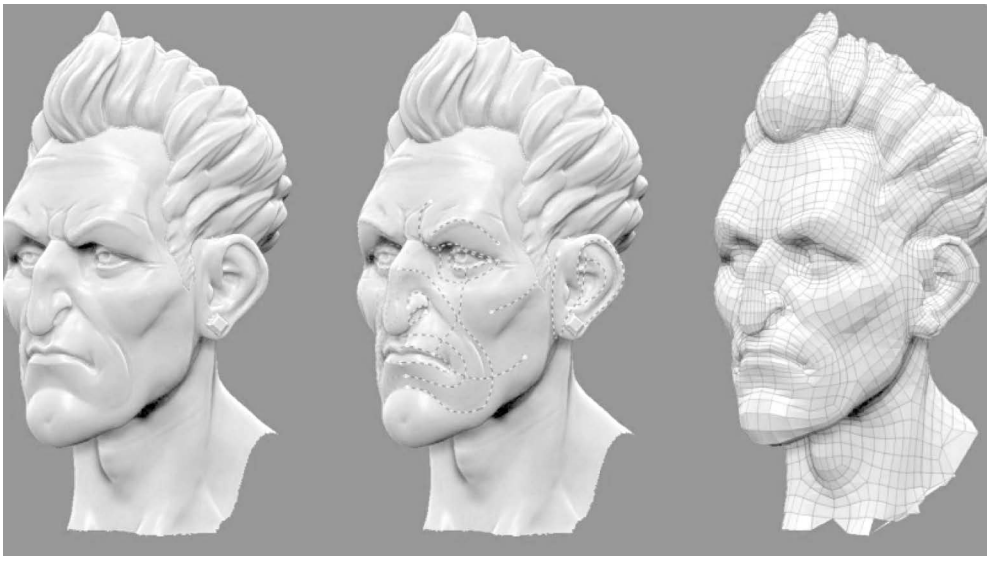


图 3.18　角色重拓扑

第三阶段：角色纹理

创造一个可信的角色需要赋予其真实的纹理。有许多不同的方法来生成角色的纹理，如使用照片，手绘纹理，甚至扫描纹理。纹理确定以后，需要将纹理映射到角色上，该过程包含 UV 展开、烘焙和纹理贴图三个步骤。

1）UV 展开

低模制作完成后，还需要对模型拆分 UV 贴图，UV 贴图通常是指贴图对应模型的坐标。当我们将 3D 的模型拆开变成 2D 平面时，每个平面对应 3D 模型的具体位置都是通过 UV 进行计算的，UV 能够使贴图精准地对应到模型表面。一般来说，最合理的 UV 分布取决于纹理类型、模型构造、模型在画面中的比例、渲染尺寸等因素，但有一些基本的原则要注

意：UV 应展开并放置在 UV 格子里，不能有拉扯和重叠的 UV，同时注意 UV 块的大小比例尽量符合模型中的相应大小比例（见图 3.19）。

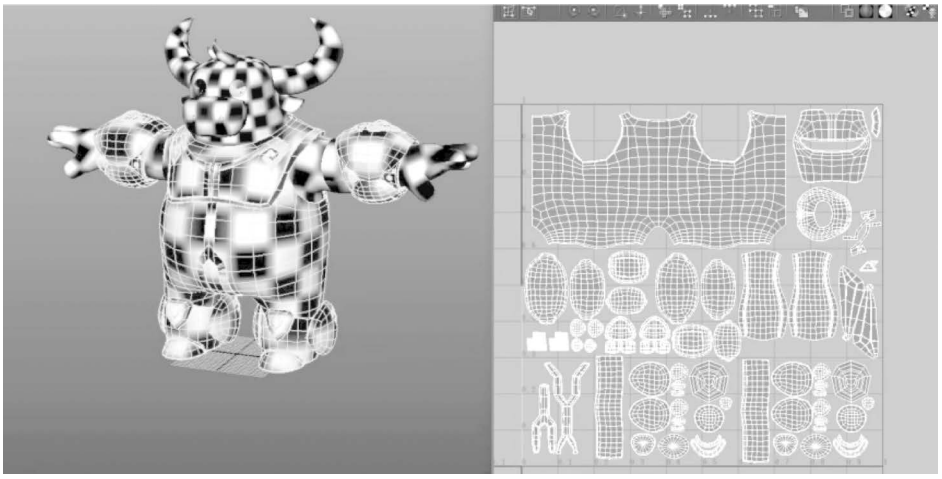


图 3.19　UV 展开

2）烘焙

烘焙是将高模上的细节信息从 3D 网格保存到纹理文件位图的过程。烘焙过程一般涉及两个网格：高模多边形网格和低模多边形网格。高模多边形网格有许多多边形（通常可达数百万个），而低模多边形网格的多边形较少（通常有数千个）。烘焙纹理可以获得高模多边形网格的高水平细节信息和低模多边形网格的低模型成本。在烘焙过程中，高模多边形网格的信息被传递到低模多边形网格并保存到纹理中。

3）纹理贴图

纹理贴图是构建 3D 模型的最重要步骤之一。在这个阶段，使用烘焙过程中生成的贴图，给角色模型表面的各个部分，如衣服上的褶皱、面部及身体上的皱纹、阴影、面部特征等，提供各自的材料物理属性，构建形象和逼真的角色模型。当然，在这个空间里还可以做很多其他的事情，如颜色、反射、粗糙度、排列细节、每种材料的镜面效果等。常见的烘焙贴图有法线贴图、OCC 或 AO 贴图、转换贴图、高光贴图、固色贴图等。将贴图赋予模型，可以得到一种很真实的效果。

第四阶段：角色绑定与蒙皮

通常，角色的运动是通过骨骼来带动的，而骨骼又是通过肌肉的收缩来控制的，即肌肉带动骨骼，骨骼再带动肢体运动。但艺术家不会直接去操纵骨骼，所以需要对骨骼添加控制器，相当于人体肌肉来带动骨骼运动。为 3D 角色模型创建一个虚拟骨骼的过程叫作角色绑定。由于骨骼与模型是相互独立的，为了让骨骼驱动模型产生合理的运动，通常把模型绑定到骨骼上的过程叫作蒙皮（skinning）。蒙皮可以将骨骼控制模型的形态节点达到合理的绑定效果，保证模型能顺利且正确地动起来（见图 3.20）。

第五阶段：制作角色动画

动画是角色建模流程的最终步骤。动画化可以让角色模型的身体动起来，创造面部表情，唤起情感，让它尽可能接近真实。通常使用特殊的工具和技术（如运动捕捉）来创建所有动作，并操控角色不同的身体部位。

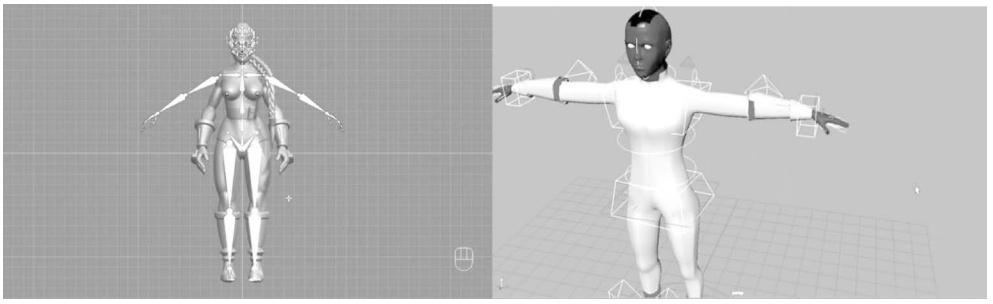


图 3.20　角色绑定(左)和蒙皮(右)

3.4　大规模城市建模方法

3.4.1　智慧城市简介

随着经济水平的提升和信息科技的发展,智慧城市建设应运而生。智慧城市的概念起源于传媒领域,是指在城市规划、设计、建设、管理与运营等领域中,通过物联网、云计算、大数据、空间地理信息集成等智能计算技术的应用,使得城市管理、教育、医疗、房地产、交通运输、公用事业和公众安全等城市组成的关键基础设施组件和服务更互联、高效和智能,从而为市民提供更美好的生活和工作服务,为企业创造更有利的商业发展环境,为政府赋予更高效的运营与管理机制。

近年来,作为智慧城市主要内容之一,虚拟城市相关软硬件技术快速发展,以 VR、计算机图形学为基础的大规模城市 3D 场景可视化技术在多个行业领域内得到推广应用。例如,城市 3D 仿真与可视化系统可以辅助政府部门进行市政规划工作,从而节约设计成本,提高工作效率; 对于公共安全、消防、医疗等机构,在虚拟城市中进行专业训练、应急指挥、预案推演,以及相关的地理信息服务系统构建; 在生活方面,基于城市仿真与可视化技术的 2D 与 3D 地图、导航、模拟驾驶等技术的应用给民众生活增加了许多便利。同时,社交娱乐应用程序中也越来越多地使用虚拟城市 3D 场景来提高真实感和沉浸感,提供更好的用户体验。

3.4.2　大规模城市建模简介及研究现状

传统的城市场景可视化技术基于卫星遥感影像、区域色斑图等基础数据构建城市场景的数字化表示。一般表现为瓦片化的二维地图系统,如 Google Map、ArcInfo、Skyline 以及国内的 Super Map、天地图等。但由于其在数据模型方面只是城市场景的二维正射投影,用户获取的有效信息非常有限,所以目前城市场景的可视化研究和产业应用已逐步向 3D 方向转移。大规模城市 3D 场景可视化是以 GIS、3D 仿真技术为支撑,对城市空间环境进行建模与可视化表达,并进一步进行分析应用的过程。由于大面积、超高分辨率、可展示信息多样等特点,城市 3D 可视化场景更接近人们对城市环境客观的认知感受,具备更高的真实感、沉浸感。特别是近年来由于可视化技术的发展,城市 3D 场景可视化技术取得了巨大的进步,也涌现出了大量的研究与应用成果。

如图 3.21 所示为谷歌公司推出的 Google Earth 和微软公司推出的 Virtual Earth 系

统，它们都是集地形、影像、3D建筑等多种数据于一体的数字地球平台。这些平台都与地理信息、社交网络服务进行了紧密结合。此外，苹果公司的3D地图系统也提供了从手机、平板电脑到工作站的多终端城市3D导航服务访问。截至目前，美国已经实现了纽约、旧金山、西雅图、洛杉矶、迈阿密等数十个大中城市的3D可视化及相关应用系统的构建。

图3.21　Google Earth(左)与Virtual Earth(右)

在我国，解放军信息工程大学在地理信息仿真领域进行了大量的研究工作。武汉大学测绘遥感信息工程国家重点实验室也对数字城市构建及其在地理信息系统中的应用进行了大量的工作。这些研究有利于促进城市地理信息资源的整合，提升城市地理信息资源的共享层次和共享效率，缩小目前地理本体的实际应用与理论研究之间存在的鸿沟。浙江大学CAD&CG国家重点实验室对虚拟城市的高真实感仿真技术还进行了一系列的深化探索，这些研究主要包括大规模复杂场景渲染、虚拟城市3D模型自动化生产与可视化关键技术等方面。清华大学计算机图形学与计算机辅助设计研究中心则重点针对基于工程图的城市3D模型重建方面进行研究。该研究立足于基于工程图的3D重建问题，分类总结了形体重建方法的研究成果，着重介绍了典型的重建算法。在比较典型算法的适用范围的基础上，剖析了目前形体重建研究所面临的问题。最后，指出了基于工程图的3D重建算法进一步的研究方向。北京大学地理信息系统研究所对数字城市和智慧城市建设过程中涉及的一些关键技术问题进行了分析与研究。该研究分析了数字城市的产生背景，提出了数字城市的研究体系：数字城市的基础研究、数字城市的实现技术，以及数字城市的工程研究，最后论述了数字城市实现的关键性技术。此外，北京航空航天大学VR技术与系统国家重点实验室研究并实现了可用于城市仿真的VR系统：BH_GRAPH。该系统是一个面向视景仿真类应用系统而开发，支持实时3D图形开发与运行的基础软件平台，能提供可扩展的软件体系结构、标准化的场景管理机制、高效率的场景处理方法和方便易用的应用程序接口，为3D图形应用系统的快速开发及高效运行提供完整的技术支持。该团队还在人机交互和沉浸式多通道立体显示方面进行了大量的创新。目前在中国，已经有一百多个城市建设了城市规划管理信息系统与空间地理基础信息系统。北京、上海、青岛、武汉、广州、深圳、宁波、长沙等城市已经率先开始了3D数字城市及相关的虚拟地理信息系统的示范应用并取得了一定的实用成果。

3.4.3　大规模城市3D模型的生成方法

在城市3D场景可视化技术中，建模是一项基本工作。传统的城市建模中，往往采用如

3ds Max、CAD 等传统建模软件。该方法建模速度慢,通常需要消耗大量的人力物力资源。因此,为加快建模效率,在建模时一般会借助其他如影像、数字高程模型(DEM)等数据。城市 3D 研究的未来方向应该是自动建模,激光雷达、倾斜摄影技术以及过程式建模等都是近年来的研究热点。

1. 激光雷达方法

激光雷达(LiDAR)是 20 世纪 90 年代在国外兴起的一种遥感主动观测技术,是以发射激光束探测目标的位置、速度等特征量的雷达系统。它具有自动化、生产效率高、精度高、不易受外界环境条件干扰等特点。随着 LiDAR 设备的发展,其价格也逐步降低。目前,在城市 3D 建模中应用广泛的就是无人机雷达系统,并可分为机载、车载和地面系统,如图 3.22 所示。

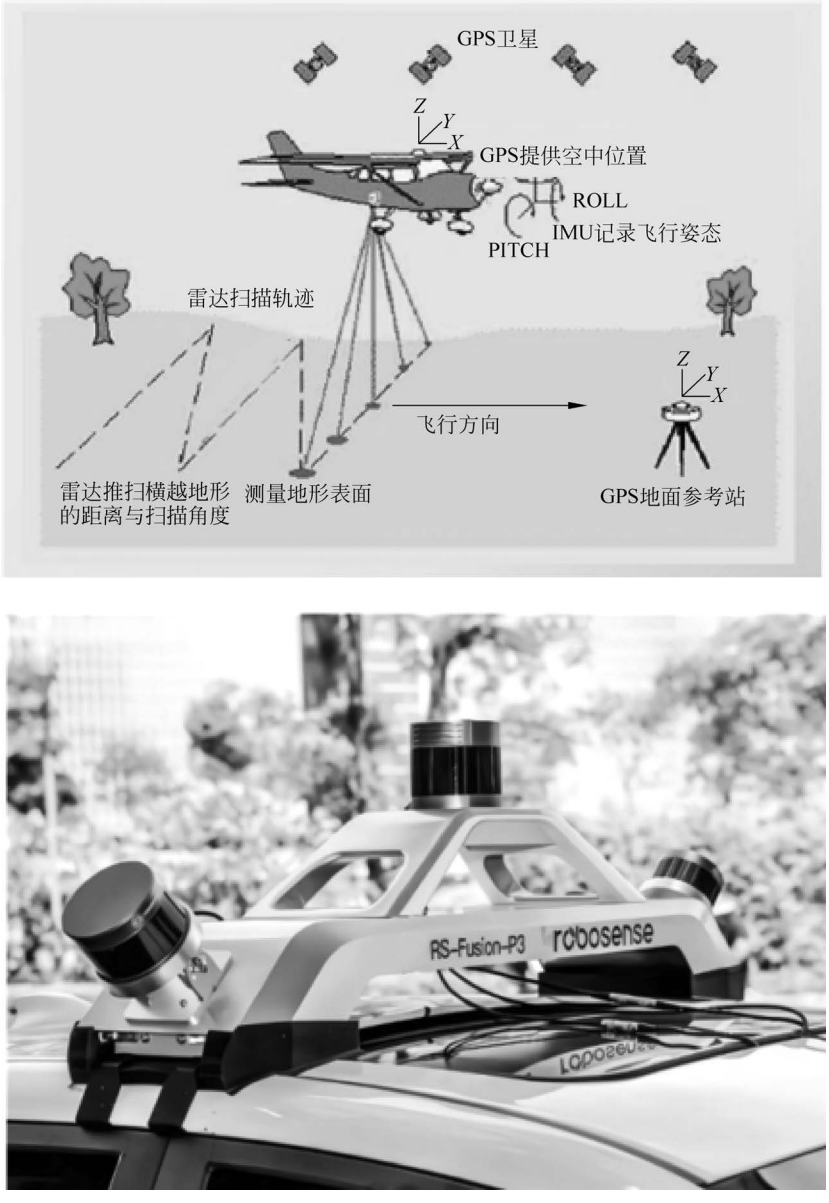


图 3.22 机载(上)与车载(下)激光雷达系统

机载 LiDAR 系统主要用来生成大范围的数字地面模型(DTM)和数字表面模型(DSM)。车载 LiDAR 可以用来获取道路与建筑物的侧面结构。地面 LiDAR 倾向于获取高精度的几何数据。利用 LiDAR 技术进行城市 3D 建模具有独特的优势,逐渐成为测绘领域的研究热点。骆钰波等针对亚热带环境条件下森林树高、胸径自动化提取精度较低、单木形态模拟较为困难的问题,提出基于地面激光雷达点云数据提取森林树高、胸径及重建森林场景 3D 模型的方法。为增加激光雷达对扫描点云进行分割的准确性,王张飞等提出一种基于深度投影的点云目标实时分割方法,可有效地识别并判断物体空间位置关系,提升碰撞识别的准确性。熊友谊等提出了基于线特征的半自动配准方法纠正图像,将鱼眼图像和 LiDAR 点云投影为透视成像的柱面全景图像,采用 Hough 变换提取图像直线特征,并利用修正迭代 Hough 变换方法,实现在鱼眼全景图像视点约束下与离散激光点云的 3D 对齐。试验表明,该方法能在较少的人工干预下实现 2D 到 3D 数据对齐。

2. 倾斜摄影技术

倾斜摄影技术是国际测绘领域近些年发展起来的一项高新技术。它颠覆了以往正射影像只能从垂直角度拍摄的局限,通过在同一飞行平台上搭载多台传感器,同时从一个垂直与四个倾斜共五个不同角度采集影像,将用户引入了符合人眼视觉的真实直观世界(见图 3.23)。该技术将不同视角采集到的影像进行处理,从而生成精度高、视觉效果好的 3D 城市模型。整个处理过程全自动,不需要人工参与,效率有了很大提升。

图 3.23　倾斜摄影技术

近年来,倾斜摄影相关软硬件技术发展迅速,出现了如天宝的 AOS、徕卡的 RCD30 等知名的倾斜摄影相机系统。梁志滔等通过分析无人机倾斜摄影在城市 3D 建模的应用情况,找出了无人机倾斜摄影在 3D 城市建模过程和成果中的优缺点,探索总结了一套无人机倾斜摄影城市 3D 建模的改进方法,将倾斜摄影和 MAX 等技术有效结合进行城市 3D 建模。李鹏等通过融合 LiDAR 点云及倾斜摄影测量进行城市级 3D 建模的技术方法和生产路线,综合运用两种技术各自优势,对城市区域和重要地物构建高精度,精细化 3D 单体化模型进行技术探索。赵飞等以倾斜设计技术作为论述基础,建立 3D 模型并将 3ds Max 插件引入渲染模型,快速构建城市 3D 场景。利用共线方程确定影像和建筑之间的几何关系,将数据对象向高度的 MESH 对象进行相应的转换,实现重构几何体。按照纹理法则获取

备选影像,将其裁切成适当面积矩形提取最小纹理信息,再以纹理映射和分配原理作为基础,将实体与模型的具体坐标建立起映射。该技术能够很好地应用于城市 3D 建模中,不但将建筑顶部细节全面地展示出来,而且能大幅提升建模效率。

3. 过程式建模

过程式建模是计算机图形学领域近十年来发展最迅速的 3D 场景构建和表示技术之一。使用该方法进行大规模的基于有限规则的 3D 城市场景建模时,无须烦琐的人工建模操作与交互,只需简单地重复套用规则语法,就能快速有效地生成大规模的 3D 模型。过程式建模基于城市场景中模型具有相似性的特点,采用形状语法描述模型的结构规则,进而重建复杂的模型,最终实现大范围场景的自动构建生成。当建模数量达到一定级别时,基于规则的过程式建模可以有效地提高效率并降低成本。过程式建模的实现相对简单,只需通过对参数进行调整,就可以构建一些复杂的模型对象,并且也是用于场景 3D 模型提取、特征描述、联系分析的一种有效理论和技术方法。在近期的研究过程中,大量的过程式建模研究都集中在传统的领域,如植被模拟、噪声生成、粒子系统以及分型算法。在最近几年,过程式建模相关的研究中出现了新的热点领域,即虚拟城市建模与可视化。过程式建模开始用于虚拟城市建模,并且与物理模型、草图绘制技术、动画建模紧密结合。王丽英等提出了一种基于过程式方法构建城市模型的建模系统,该系统根据用户的设计意图生成城市的 2D 蓝图和 3D 模型。蓝图设计主要采用道路规则、布局模板驱动和约束布局优化三种方法,重点是控制城市整体结构,表现道路和建筑的空间风格。3D 造型根据蓝图提供的 2D 信息,通过重用并组装模型库的基本模型来自动生成。该系统能够在数十分钟内高效可控地生成视觉可信的城市模型。王元等对过程式建模中路网生成方法的研究进行了梳理与总结,并指出今后对于城市过程式建模,路网生成与城市环境的结合将是一个重要的研究方向。

3.5 动画仿真方法

3.5.1 角色仿真概述

基于生物体运动的角色仿真在影视制作、计算机游戏、广告动画设计、VR 等诸多领域都有着重要的应用。角色仿真不是一项独立的工作,而是多项工作的融合。在造型、动画、绘制、合成四项过程中,角色造型是角色仿真的首要问题,是构建角色真实感的前提。绘制与合成继承了定格动画创造运动错觉的特点,通过每秒改变 24 次或 30 次造型来模拟运动过程,最终将绘制的 3D 模型合成投影到显示器屏幕上。

早在 20 世纪 70 年代,最早的角色仿真技术只能生成手、脸等人体局部的简短序列。随着技术的发展,3D 角色仿真更广泛地被观众接受。3D 动画使用计算机技术建立模型并制作动画,在空间方面有明显的优势,通过输入参数即可自动计算出发生透视变化的效果。在现代影视、游戏中高超的角色仿真技术,不仅能够模拟出真实生物的运动细节,还能制作出感官真实的幻想生物,完成实景拍摄难以达到的视觉特效。正如著名动画公司梦工厂的那句名言:“我们所做的一切都是关于纯粹的想象力。”

角色仿真不仅在影视工业中十分重要,在计算机图形学研究中也十分重要。其中影响力最大的学术会议是 SIGGPRAPH,他们每年都会刊出数篇关于角色仿真技术的前沿研究

论文，它们通常是游戏、影视界与大学高校的合作成果。SIGGRAPH（Special Interest Group on Computer Graphics and Interactive Techniques）是由 ACM SIGGRAPH 组织的计算机图形学年会，始于 1974 年。其主要会议在北美举行，SIGGRAPH Asia 是每年举行的第二次会议，自 2008 年以来一直在亚洲各国举行。会议内容既包括学术报告，也包括行业贸易展览会。

3.5.2　角色造型

角色仿真的首要问题就是构建出逼真的角色造型，并考虑角色的运动特征。造型主要反映角色的外形，主要由生物体的骨骼结构和附着在骨骼上的肌肉运动决定。在运动过程中，受肌肉伸展与收缩的控制，骨骼发生旋转，同时皮肤发生形变（见图 3.24）。在形体塑造过程中，为了角色的感官真实性，往往需要参考大量解剖学资料来模仿生物体的运动姿态。通常可以获得的参考资料有标本、模型、图片、书籍等。随着硬件发展，动作捕捉方式也逐渐流行。根据动作捕捉采集的数据，形成初步角色动作模型，其后再根据艺术创意进一步调整修改。专业动作捕捉需要专用空间、设备、软件和技术人员等支持，具有一定成本和技术门槛。

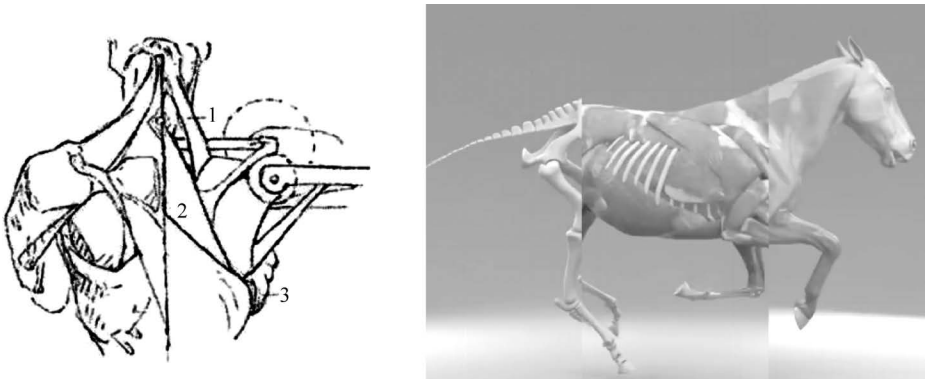


图 3.24　人体（左）及马（右）的解剖特征

绑定与蒙皮。骨骼绑定与表面蒙皮是主流的角色造型方法，将角色分为抽象化的骨骼绑定与表面蒙皮两部分。表面蒙皮是指由若干多边形面片拼接形成的几何表面，用于角色的表面呈现方式。蒙皮确切地定义了骨骼绑定移动时表面的变形方式，以便动画师通过简单的骨骼运动产生角色动画。骨骼绑定是指为一组相互连接的部分的层次结构设置蒙皮变形参数的过程。骨架位于角色内部，准确定义角色的移动方式以及对应限制。有别于生物解剖的骨架，绑定骨架不一定真实，但符合艺术表现的需求。绑定与蒙皮通常用于驱动人类等生物体，使动画过程更为直观，并且可以使用相同的技术来控制任何物体的变形，甚至包含门、勺子、建筑物等拟人动画。当动画对象比人形角色更一般时，骨骼集可能不是分层或互连的，而只是表示它正在影响的网格部分（见图 3.25）。

3.5.3　角色动画

动画导演诺曼·迈凯伦曾说："动画不是移动的绘画艺术，而是绘制动作的艺术。"角色动画师类似于电影或舞台导演，能够为角色注入生命，创造思想、情感和个性。角色动画师

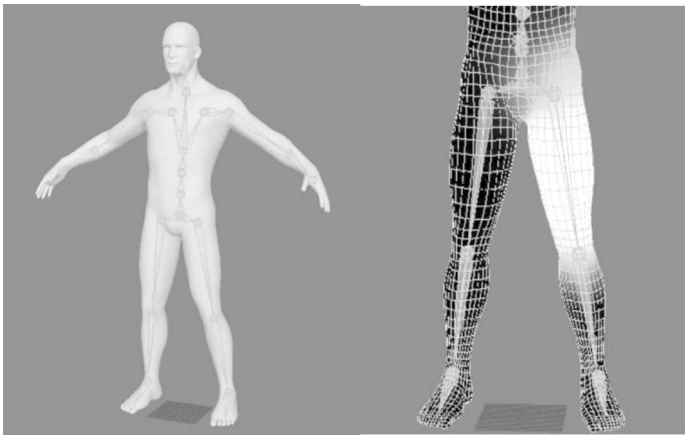


图 3.25　人体的骨骼绑定(左)与表面蒙皮(右)

通常会在关键帧上控制角色部位的变化,如四肢、眼睛、嘴巴、衣服等。在动画和电影制作中,关键帧是定义平滑过渡的起点和终点的绘图或镜头。这些被称为帧,因为它们在时间上的位置是以胶片上的帧或数字视频编辑时间轴上的帧来衡量的。关键帧之间的外观差异可由计算机在称为补间或变形的过程中自动计算。

在角色仿真中,状态机常用来驱动角色运动。动画通常由一系列零散的预录制的运动片段产生,它们靠状态机联系起来,提供模块化的组织形式。用户定义角色或骨架蒙皮可能存在的一系列不同状态,反映出不同的运动模式。进入、退出这些状态的流程图构成了运动的状态机。游戏开发者构造很多的动画片段,然后通过状态机判断当前角色的状态,然后选择播放某一个动画片段(见图 3.26)。状态机动画作为一项成熟的技术,因其简单易实现而应用广泛。但在动画切换过渡时容易失真,且状态数量随转换次数增加而增长,扩展性受限。运动匹配在状态机动画的基础上,通过额外算法选择动画状态,避免手工设置的劳动,同时能缓解切换的失真现象。

在传统动画制作流程中,每一帧都必须手工编辑。现在动画师可以通过使用数字关键帧动画来识别图形的不同元素并选择这些元素如何随时间移动或变化,从而减少重复劳动。要在数字动画序列中创建动作,首先需要定义该动作的起点和终点,这些标记称为关键帧,它们用作所有不同类型动画程序中动作的锚点(见图 3.26)。关键帧动画根据关键帧对象不同,产生关键帧之间的内插值。拟合关键帧之间的曲线是提高运动平滑度的关键因素,常见的方法有线性拟合与样条拟合。前者通过线性函数近似中间帧的连续变化,后者使用高阶多项式逼近运动函数。另外,可以通过添加状态参数实现更精确的运动描述,或是利用大数据集提升采样精度以提高插值帧率。

受机器学习与神经网络技术的影响,状态机动画与关键帧动画得以更加智能化与自动化。通过记录下一帧动画的标签,由人工智能直接输出后续帧,从而实现运动预测、合成与控制等高级任务(见图 3.27)。数据驱动的动画从预录制的数据集产生,从而提高数据集的复用性与模型的可扩展性。

通常而言,数据集是静态的,而数据驱动产生的动画能够动态变化。一种名为运动图的方式能够将数据帧投影到特征空间,并在特征空间通过帧间的连接与过渡产生动画。基于网络的方法往往根据数据来学习帧的上下文关系,从而根据当前帧预测未来帧的具体数值。

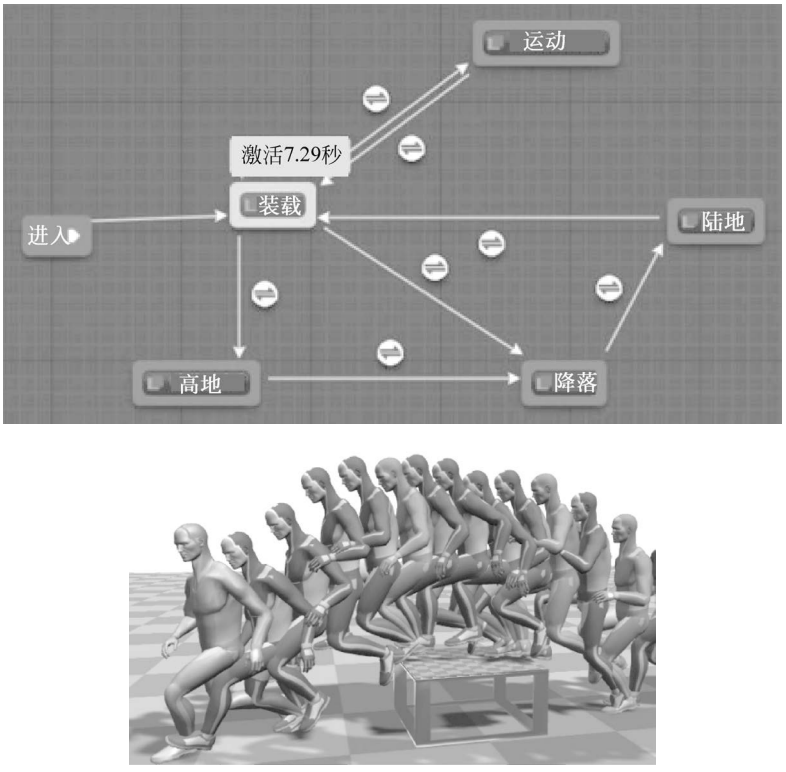


图 3.26　虚幻引擎下的状态机动画图示(上)及关键帧动画技术(下)

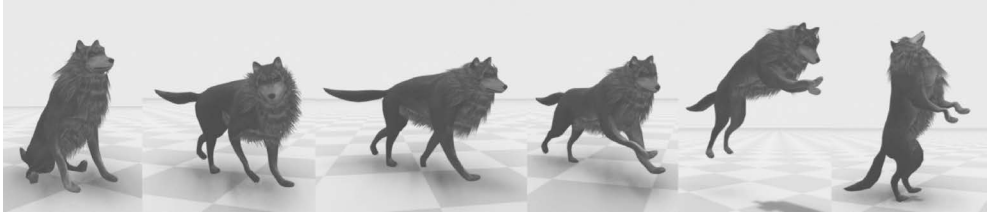


图 3.27　数据驱动的四足动物仿真

数据驱动的方法充分利用了大数据集的优势,同时尽可能对数据进行复用与推测,以产生观感真实的运动(见图 3.28)。

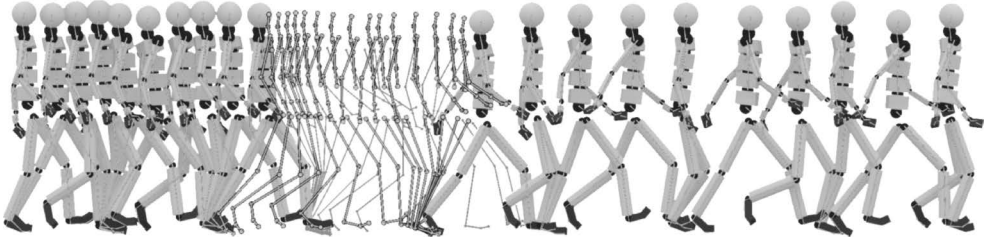


图 3.28　数据驱动的数字人仿真

对于表演与动作捕捉中太过危险的运动,可以通过计算运动物理特性和阻力来产生动画。根据虚拟解剖学属性,例如肢体重量、肌肉反应、骨骼强度和关节约束,物理动画能够实

现逼真的弹跳、翻滚、打击等效果。骨骼动画则能直接对关节位置赋值来改变运动姿态(见图 3.29)。物理动画类似在关节处添加电机,让整个动作符合物理定律,如同控制机器人。确定物理过程的方式包括基于搜索和基于强化学习的方法。其中基于搜索的方法通过前向模拟与评估成本函数来找到最优的运动过程,即通过优化运动的控制参数,而非运动姿态本身来构造控制器,以提升模型的稳健性。基于强化学习的方法通过优化代理来选择时间回报最大的行为,并根据回报函数与状态不断更新。强化学习的方法更适用于复杂运动。

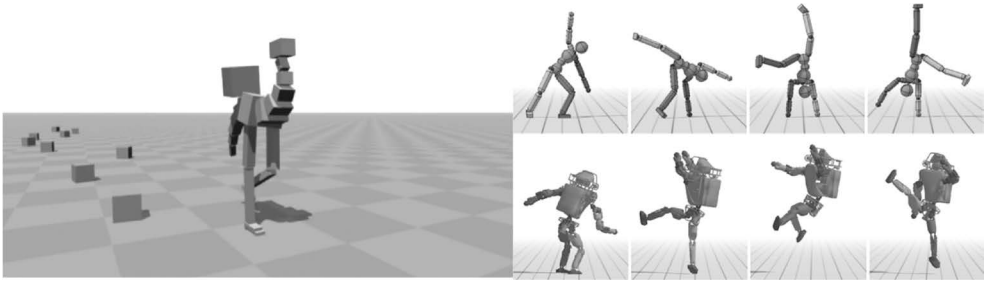


图 3.29 双足动物物理仿真及数字人物理仿真

观看视频

3.6 实践环节——玩家角色建模

用户的虚拟化身,即玩家角色,往往是某些应用中最重要的模型。创建一个完美的玩家角色主要依赖于商业软件 Autodesk 3ds Max、Maya、Zbrush 等(如 3.3 节所述)。因受篇幅限制,本章将假设这个玩家角色模型已经被创建好了,更多关注开发流程中这个模型如何被导入 VR 应用中。本案例中用到的玩家角色模型随教材对外公开,读者可以从教材附带的电子资源中获取。

图 3.30 展示了玩家模型加载和初始化的流程,这个流程衔接了之前的游戏启动流程,同时为后面实现玩家对模型的控制和玩家技能做好了准备。

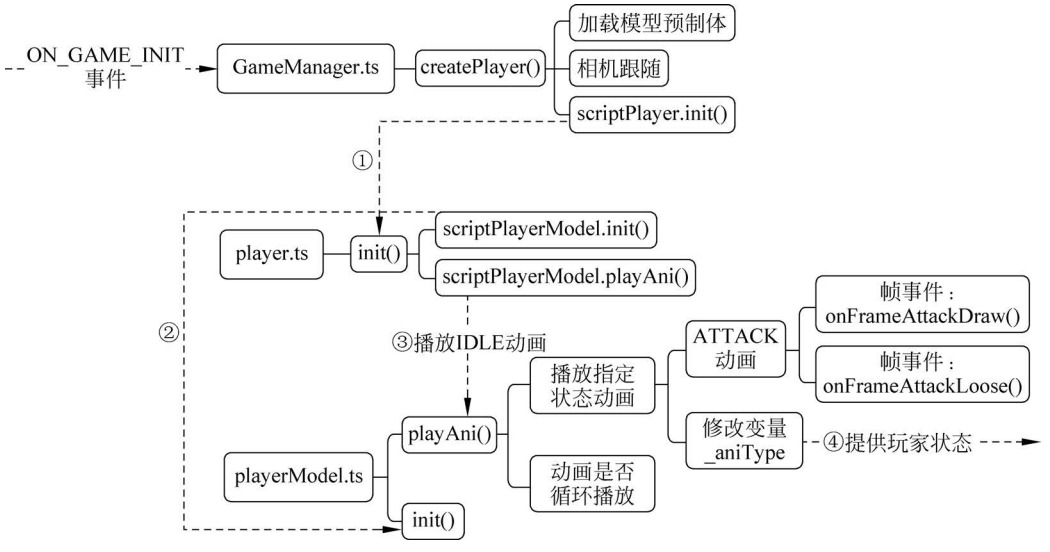


图 3.30 玩家模型加载和初始化

3.6.1　玩家模型资源导入

在本项目中，主角模型为预制件，附带骨骼动画以及相关脚本等属性。和往常一样，我们将本讲的资源导入。注意不要让文件夹相互覆盖，如果无法自动合并同名文件夹，则需要将本讲资源按照目录层级一一自行拖入。导入后资源管理器内的层级如图 3.31 所示。

图 3.31　课程资源导入

我们在 gameManager 脚本中实现将玩家模型加载到 fight 场景的功能。首先，声明准备用于存储玩家模型的节点：public static ndPlayer：Node＝null!；//玩家节点，这之后，创建加载玩家模型为刚刚声明的 ndPlayer 节点的函数（如果编译错误，记得检查代码顶部是否导入了依赖的框架文件）：

```
1.  private _createPlayer () {
2.    ResourceUtil.loadModelRes("player/player01").then((pf: any) =>{
3.      GameManager.ndPlayer = PoolManager.instance.getNode(pf, this.node) as Node;
4.    })
5.  }
```

最后，在_onGameInit()函数的末尾，将_createPlayer()加入到读取“loading 界面”的代码的回调内。这里的调用我们已经在 4.2 节中完成，在加载 loading 界面完成时会通过框架文件 UIManager.ts 中的 showDialog 函数，将_createPlayer()作为回调传入给 loading.ts 中的 show()函数，最终在界面加载完成后调用。

按之前的启动流程加载场景，就能在 3.6.1 节的编辑器内部预览时在层级管理器看到相应的场景。在进入战斗界面后，gameManager 节点下出现了新的子节点 player01。至此，gameManger.ts 脚本的代码如下：

```
1.  // 省略库文件的导入
2.
3.  @ccclass('GameManager')
4.  export class GameManager extends Component {
5.    public static ndPlayer: Node = null!;                    //玩家节点
6.
7.    onEnable () {
8.      ClientEvent.on(Constant.EVENT_TYPE.ON_GAME_INIT, this._onGameInit, this);
9.    }
10.
11.   onDisable () {
12.     ClientEvent.off(Constant.EVENT_TYPE.ON_GAME_INIT, this._onGameInit, this);
13.   }
14.
15.   private _onGameInit () {
16.     let level = PlayerData.instance.playerInfo.level;
17.     UIManager.instance.showDialog("loading/loadingPanel", [() =>{
18.       this._createPlayer();
19.     }]);
20.   }
21.
22.   private _createPlayer () {
23.     ResourceUtil.loadModelRes("player/player01").then((pf: any) =>{
24.       GameManager.ndPlayer = PoolManager.instance.getNode(pf, this.node) as Node;
25.     })
26.   }
27. }
```

3.6.2 骨骼动画资源导入

在层级管理器中找到预制体 player01 并双击打开,可以看到主角模型的具体层级。在 body 节点中,我们可以看到 cc.SkeletalAnimation 组件,即骨骼动画,可以直接理解为绑定到玩家模型上的一组动画,可以在动画编辑器中查看,但无法编辑。其他与我们之前学过的关键帧动画几乎没有区别。骨骼动画组件的属性如表 3.1 所示。

表 3.1 SkeletalAnimation 组件属性

属 性	描 述
Clips、DefaultClip、PlayOnLoad	与动画组件的属性功能一致,详情请参考动画组件属性
Sockets	用于将某些外部节点挂到指定的骨骼关节上,属性的值表示挂点的数量。详情请参考“挂点系统”部分的内容
useBakedAnimation	该项用于切换使用预烘焙骨骼动画或实时计算骨骼动画,详情请参考“骨骼动画系统”部分的内容

与之前的节一样,玩家模型身上不只具备一组骨骼动画。我们需要获取到玩家模型身上的 cc.SkeletalAnimation 组件,通过其播放指定的动画。首先在 script/fight 文件夹下新建脚本 playerModel.ts,挂载到 player01 下的 body 节点上(和 cc.SkeletalAnimation 同样的节点)。

首先声明用来存储骨骼动画组件的变量,并在 body 节点的属性检查器中将 body 节点自身拖入:

```
@property(SkeletalAnimationComponent)
public aniComPlayer: SkeletalAnimationComponent = null!; //骨骼动画组件
```

然后我们定义用于播放指定骨骼动画的函数，目前只实现根据动画名称播放指定骨骼动画的功能，同时根据 isLoop 判断是否循环播放动画。

```
public playAni (aniType: string, isLoop: boolean = false) {
    this.aniComPlayer?.play(aniType);

    if (this._aniState) {
    if (isLoop) {
      this._aniState.wrapMode = AnimationClip.WrapMode.Loop;
    } else {
      this._aniState.wrapMode = AnimationClip.WrapMode.Normal;
    }
}
```

最后，在 start 函数内播放指定的动画状态：

```
start() {
    this.playAni(Constant.PLAYER_ANI_TYPE.IDLE, true);
}
```

记得根据代码提示导入以上代码所依赖的库文件。此时开始演示，完成战斗界面的加载后，就可以看到玩家模型处于 idle 动画状态，也可以在 start()函数内选择其他动画播放查看效果，如 ATTACK、RUN 等。下方代码框内为 Constant.ts 内有关玩家动画类型的定义：

```
//玩家动画类型
public static PLAYER_ANI_TYPE = {
    IDLE: "idle",                                    //待机
    RUN: "run",                                      //向前跑
    ATTACK: "attack",                                //攻击
    DIE: "die",                                      //死亡动作,后仰倒地
    REVIVE: "revive",                                //复活
}
```

动画化事件也是事件系统的一种。通过在动画时间轴的指定帧调用动画事件，函数可以更好地充实动画剪辑。在动画时间轴某一帧上添加事件帧后，动画系统将会在动画执行到该帧时，根据事件帧中设置的触发函数名称去匹配动画根节点中对应的函数方法并执行。

3.6.3　事件脚本编写

需要用脚本定义指定动画帧所调用的动画时间和内容，这与之前定义 Button 组件触发事件的思路类似。本节以攻击动画 ATTACK 为例，在拉弓时显示箭，在箭射出时隐藏玩家模型身上的箭。

首先，增加声明如下变量，并在 body 节点的属性编辑器界面将其子节点 arrow 拖入 ndArrow 属性：

```
@property(Node)
public ndArrow: Node = null!;                        //攻击时候展示的箭矢
```

然后如下定义回调函数：

```
public onFrameAttackDraw () {
    this.ndArrow.active = true;
}

public onFrameAttackLoose () {
    this.ndArrow.active = false;
}
```

此外,定义初始函数 init(),玩家模型在一开始隐藏箭:

```
public init () {
    this.ndArrow.active = false;
}
```

并在 player.ts 的 init()函数中播放 idle 动画后调用(init 相关这两步只是为了让箭的出现和隐去过程完整,与动画帧事件无关)。

```
import { _decorator, Component, Node } from 'cc';
import { Constant } from '../framework/constant';
import { playerModel } from './playerModel';
const { ccclass, property } = _decorator;

@ccclass('player')
export class player extends Component {
    @property(playerModel)
    public scriptPlayerModel: playerModel = null!;          //玩家动画组件播放脚本

    start() {

    }

    update(deltaTime: number) {

    }

    public init() {
        this.scriptPlayerModel.playAni(Constant.PLAYER_ANI_TYPE.IDLE, true);
        this.scriptPlayerModel.init();
    }
}
```

选中 body 节点后打开动画编辑器,正如之前所说,我们无法在其中看到骨骼动画的细节,但仍可以在左上角的 clips 属性切换不同的动画,为 ATTACK 动画添加事件。

将时间控制线拖动到 0～15 帧,单击菜单工具栏中的插入帧事件按钮,添加事件。右键选中出现的图标(绿色框内,见图 3.32)。

单击编辑后,会看到图 3.33 所示页面:

(1) 添加新的触发函数;

(2) 保存事件函数;

(3) 填写需要触发的函数名称;

(4) 删除当前事件函数;

(5) 添加传入的参数(目前支持 String、Number、Boolean 三种类型)。

直接在 function Name 中填入 onFrameAttackDraw 添加即可。按相同方法为第 0～25

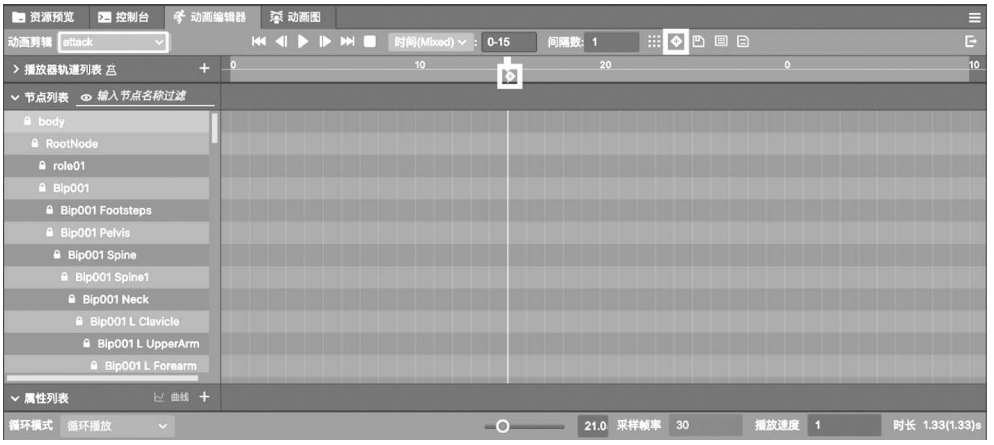


图 3.32　编辑动画帧事件

图 3.33　动画帧绑定函数

帧添加事件回调 onFrameAttackLoose。

不同骨骼动画的播出对应着玩家模型的不同状态。我们可以在播放骨骼动画时通过所要播放的动画名检测玩家模型所处的状态。

首先，在 playerModel.ts 中增加如下变量的声明：

```
1.  private _aniType: string = "";                          //动画类型名
2.  private _aniState: SkeletalAnimationState = null!;      //动画播放状态
3.  public isAniPlaying: boolean = false;                   //当前动画是否正在播放
```

接着，重新对 playAni()函数的内容做如下修改：

```
1.  public playAni (aniType: string, isLoop: boolean = false){
2.    this._aniState = this.aniComPlayer?.getState(aniType) as SkeletalAnimationState;
3.
4.    if (this._aniState && this._aniState.isPlaying) {
5.        return;
6.    }
7.
8.    this._aniType = aniType;
9.
10.   this.aniComPlayer?.play(aniType);
```

```
11.     this.isAniPlaying = true;
12.   }
```

这里主要是增加了依据输入的动画类型名修改表示模型所处状态的变量,修改表示模型是否正在播放动画的变量 isAniPlaying,同时获取模型相应的骨骼动画状态。这样,我们就可以根据_aniType 和 isAniPlaying 判断模型所处状态,作为属性接口提供给外界了。

在 playerModel.ts 中增加如下代码:

```
1.   //是否正在运行
2.   public get isRunning () {
3.     return this._aniType === Constant.PLAYER_ANI_TYPE.RUN && this.isAniPlaying === true;
4.   }
5.
6.   //是否待机
7.   public get isIdle () {
8.     return this._aniType === Constant.PLAYER_ANI_TYPE.IDLE && this.isAniPlaying === true;
9.   }
10.
11.  //是否正在攻击
12.  public get isAttacking () {
13.    return this._aniType === Constant.PLAYER_ANI_TYPE.ATTACK && this.isAniPlaying === true;
14.  }
```

3.6.4 在主场景中调用

目前,我们已经可以播放指定的动画,获取玩家模型状态,并同时触发事件,这些都会在玩家进行相应操作时发生。我们将会在第 7 讲中实现玩家操控主角模型的脚本 player.ts,因此在这里提前创建该脚本,在其中调用 playerModel.ts 播放骨骼动画的函数,使模型处于初始的 IDLE 状态。

首先,删除 playerModel.ts 的 start 函数内播放骨骼动画的脚本。接着,在 script/fight 文件夹下创建 player.ts 脚本,并挂载到 player01 节点上。

为了获取到控制玩家模型的脚本,在 player.ts 中声明一个 playerModel 类的变量,并创建 init()函数完成跨脚本调用,至此 player.ts 的脚本内容如下:

```
1.   import { _decorator, Component, Node } from 'cc';
2.   import { Constant } from '../framework/constant';
3.   import { playerModel } from './playerModel';
4.   const { ccclass, property } = _decorator;
5.
6.   @ccclass('player')
7.   export class player extends Component {
8.     @property(playerModel)
9.     public scriptPlayerModel: playerModel = null!;          //玩家动画组件播放脚本
10.
11.    start() {
12.
13.    }
14.
15.    update(deltaTime: number) {
16.
17.    }
```

```
18.
19.    public init() {
20.      this.scriptPlayerModel.playAni(Constant.PLAYER_ANI_TYPE.IDLE, true);
21.    }
22.  }
```

最后，在 gameManager 导入玩家模型时执行 player.ts 对玩家模型的初始化，即调用刚刚完成的 init()。依然是先声明，再获取，再调用：

（1）gameManager.ts 增加如下声明：public static scriptPlayer：player = null!；

（2）接着将_createPlayer 函数修改为如下所示内容：

```
1.   private _createPlayer () {
2.     ResourceUtil.loadModelRes("player/player01").then((pf: any) =>{
3.       GameManager.ndPlayer = PoolManager.instance.getNode(pf, this.node) as Node;
4.
5.        let scriptGameCamera = GameManager.mainCamera?.node.getComponent(GameCamera) as
     GameCamera;
6.
7.       scriptGameCamera.ndFollowTarget = GameManager.ndPlayer;
8.
9.       let scriptPlayer = GameManager.ndPlayer?.getComponent(player) as Player;
10.      GameManager.scriptPlayer = scriptPlayer;
11.      scriptPlayer.init();
12.    })
13.  }
```

此时进行演示，玩家模型的动画应该与 3.6.3 节完成时并无区别，但我们改变了其调用方式，为后面与玩家移动的结合做好了准备。

3.7　小结


观看视频

3D 建模是 VR 的基础能力。为了建立一个逼真的数字化虚拟世界，需要运用 3D 建模技术对虚拟环境、角色进行建模。本章简要介绍了其基础方法和基本理论，同时也介绍了较为前沿的自动化建模方法，以供读者参考。

在本章的实践部分，完成了战斗场景中玩家模型的加载，并实现了主角模型 playerModel.ts 脚本的功能，包括播放玩家模型骨骼动画、实现动画帧事件触发的回调、根据正在播放的玩家动画判断玩家状态等。

习题

1. 调研现在主流的用于实现包括场景建模、角色建模、动画建模等任务的商业软件，分析现有工具的优缺点，思考在实际 VR 应用中的价值与不足。

2. 从场景建模、角色建模、动画建模等任务中选择一个，调研其前沿科研成果，完成一份调研报告，描述前沿技术带来的 VR 应用的体验提升和技术革新。