第 1.5 节电影制作概述

从表面上看，视觉效果管道看起来很简单。把它想象成一条生产线。原材料，例如来自真人录像的印版，进入并通过一系列可重复的过程，将它们转化为成品。这条生产线的最终输出是 2D 图像序列的集合，与你开始时的图像序列没有太大差别 - 至少，如果你考虑一个特技双镜头绑在背带上并落在床垫上的镜头与 a 相当类似拍摄的，让我们说汤姆克鲁斯，从飞机上自由跳跃并坠入雪崩。

然而，现实却截然不同。视觉效果管道很复杂。之所以如此归结为技术和哲学考虑的混合。

在技术方面，事实证明原材料实际上比它们最初出现时更加多样化。事实上，这些板块只是雪崩的开始。让我们剖析我们假设的汤姆克鲁斯射击，看看为什么会这样。

这部虚构电影中的动作（如我们的虚构剧本中描述的粗略故事板和详细的预可视化）如下：汤姆克鲁斯站在一架移动飞机的货舱中，其门是敞开的，他的头发和衣服汹涌澎湃。相机将他画成中等特写镜头，但当他开始向前跑时，我们向后拉开，露出快速后退的飞机和他的暴跌的身体，在蔚蓝的天空背景下构筑。我们和他一起摔了一会儿，向下倾斜，在他上方飘过，露出冰雪覆盖的山脉，正在接近眩目。在最后一刻，相机转过身来，抓住汤姆，当他撞到地面并开始滚动，陷入粉状白色爆炸。也许这应该是两次拍摄，而不是一次，但我们想象中的内部导演，在第一人称射击游戏和主题公园游乐设施的饮食上，喜欢不可能的相机移动和连续动作。

我们的客户实际上为我们提供了几个元素我们有一个汤姆克鲁斯的镜头，站在一架飞机的货舱，他的头发和衣服被几个大风扇吹了，在拍摄开始时，它们不在框架内，但是当我们拉动它们时会看到它们回来，揭示飞机本身是一个声音阶段中间的固定片。已经设置了一个粗糙的绿色屏幕，以便我们可以通过以后“键入”框架的绿色部分来将 Tom 与背景分开 - 但它更像是绿色的建议，而不是实际的可按键背景。我们也得到了上述特技双重元素，以及汤姆在地面上滚动的第三个元素。带有汤姆的盘子在 35 毫米胶片上以变形格式拍摄，但特技镜头是由第二个单元拍摄的，使用不同的相机：数字具有不同的宽高比。这可能看起来像是一个不必要的细节，但它会对我们如何进行工作产生影响 - 就像拍摄的许多其他方面一样，不受我们 VFX 公司的控制。

我们还收集了其他材料，以帮助我们组装镜头。在场景中，我们协商拍摄镀铬球和 HDRI（高动态范围图像）以帮助我们捕捉环境照明。我们还对飞机内部进行了激光雷达扫描，以及一系列参考摄影。没有真正的飞机，所以我们采购了这种类型飞机的图像：波音 C-17 Globemaster III。我们在中性照明条件下拍摄了数百张照片，这些照片将用于模型和纹理参考。我们还有汤姆克鲁斯的高分辨率数字扫描和他的各种姿势的摄影参考，并非所有这些都是讨人喜欢的，没有人会看到任何未签署保密协议的人。还存在具有相机镜头信息，帧计数等的数据表。并参考了雪崩的摄影作品。这不是一个详尽的清单。

我们还没有开始研究这个镜头，但我们已经收集了大量不同格式的数据。

根据视觉效果标准，镜头并不太复杂：可能具有中等复杂性。我们将从汤姆克鲁斯飞机板开始，我们将重新投射到飞机内部的 3D 模型上。然后我们将转换到汤姆的 CG 数字双重落在天空中，并保持这个数字双倍一直到镜头结束，匹配姿势和重新投影汤姆克鲁斯的脸从最后的第二个元素。 （当然，当我们开始工作时，我们不知道这一点，所以我们将花一点时间尝试使用特技双板和面部更换。）天空将是一个哑光绘画：数字从照片和手绘元素组装的背景。山脉将是遮罩绘画和几何体投影的组合，前景树的 3D 几何体。雪崩将通过数字模拟的组合来创建，模拟流体，粒子和刚体的行为，以及来自图书馆的素材。

我们的每个真人摄像机都需要被跟踪，而汤姆克鲁斯和特技双的两个实例需要旋转，以便

数字角色的移动可以与现场演员的移动相匹配。这项工作是在中亚的一个不同的工厂完成的，在不同的时区运作。在本地，布局部门将对这些数据进行迭代，并构建单个摄像机轨道和粗略动画轨道，使此材料符合逻辑世界空间位置，并确保其符合序列的连续性。（这个镜头不是孤立的。它是一系列镜头的一部分，必须在剪辑中一起流动。）粗略的动画将被提供给动画部门，动画部门将对它进行精细处理，直到它看起来完美。

rotoscoping，布局和动画部门正在使用的数字模型由资产团队整合在一起，资产团队实际上分布在多个地理位置之间。资产是为了在电影中的几个不同的序列中使用而构建的，因此已经进行了许多变化。（虽然汤姆克鲁斯穿着他的燕尾服跳出飞机，这件燕尾服被撕掉了，而不是他在电影早期跳出酒店窗户时的原始状态。）此外，资产部门创建了一个号码不同的 LOD（细节水平）模型，包括专门用于布料模拟的高分辨率模型，以及用于物理模拟的低分辨率模型。所有这些变体之间需要保持一致的任何变化。我们还必须“修饰”数字汤姆克鲁斯的头发，这不仅要与真实汤姆克鲁斯的外观相匹配，还要与之相匹配。

值得庆幸的是，这个序列的所有工作都没有与其他视觉效果设施共享。但是，我们被要求将 Tom Cruise 模型，装备和纹理导出到另一家公司，该公司正在电影的其他地方使用它们。该公司运行与我们不同的管道，因此需要协商交换格式。（我们正在同时进行的另一场演出并不是那么幸运，要求我们通过谈判转移一座 1000 英尺高的石塔，以换取十五个劫掠的恶魔，一连串三十杆。）

时间尺度很短，因此镜头的许多不同方面都是并行开始的。照明部门在动画开始之前就开始设置，利用集中式外观开发工作，角色和飞机只完成了 70%。负责这些数字模拟的 FX 部门也很早就开始了，试图在等待更新动画时阻止某些事情。

我们已经同意我们的客户，我们将定期向他们展示他们正在进行的工作，他们告诉我们，他们不会签署任何动画，直到他们看到它至少基本的布料和头发模拟，点亮运动模糊和阴影。这意味着沿着管道推送不完整的迭代，跟踪不同部门使用的不同版本的资产，并确保只有经过批准的资产才能在下游进一步使用。它还会创建一些非常大且迂回的反馈回路。

此外，每次迭代都会创建更多数据。使用 FX 元素，数据集可能非常庞大，占用数 TB 的存储空间。虽然较小，但纹理，动画数据和哑光绘画也占据了相当大的空间，板块本身也是如此。有时，我们需要恢复到旧版本，或者同时使用多个备选方案。这使得归档数据成为一个复杂的过程，因为设施的服务器上没有足够的空间，但是你不能总是知道何时不再需要特定的文件。

生成数据也会导致复杂性。在这个镜头中，每次新的雪崩模拟都需要半天时间才能运行，这仅适用于几个子模拟中的一个，即“模拟层”。（这个镜头并不是特别复杂。雪崩真正开始的序列中的下一个镜头需要 72 个小时才能完成 - 也就是说，当它没有中途崩溃时。）所有这些模拟都在设施的服务器场，以及所有渲染工作 - 将 3D 数据转换为 2D 图像 - 以及许多其他更平凡的任务。调度所有这些计算是很困难的。

当谈到渲染本身时，我们的目标是每个渲染通道每帧不超过一个或两个小时。我们目前的镜头有五到六个 FX 层，加上飞机内部和外部以及环境的渲染，这意味着每个框架总共需要大约十个小时来计算。大约八秒钟，镜头本身由 197 帧构成，因此每次完整的迭代需要 2,000 小时的渲染时间。（还有一个或两个“故障框架”，每个需要 15 到 20 个小时，需要在农场外手动推送。）

迭代也会产生其他问题，因为我们正在执行的一些任务本质上是“破坏性的”。让我们来看看破坏性和非破坏性过程之间的区别。

"正如我们之前讨论的那样，汤姆克鲁斯的数字模型有几种变体。汤姆在他的燕尾服中有干净的版本，它在早期的序列中被使用，而汤姆则跳出了飞机。汤姆有五天的胡须生长，衣服上还有瘀伤，划痕和裂缝。但是，他基本上是一样的。有意义的是，干净变化的纹理首先被绘制，并且更改应用于这些之上。破坏性的工作方式是拍摄原始图像并在其上面直接绘制胡须，消除原始

像素。非破坏性的工作方式是将胡须涂在单独的层中。在第一种情况下，如果干净的艺术品发生变化，即使胡须本身没有变化，您也必须完全重做其他版本。在第二种情况下，您只需重新涂抹胡须层 - 最多，您需要稍微触摸它。显然，这种非破坏性的工作方式更有效，特别是因为组合这两层可以很容易地自动化。″

"在我们的拍摄中，汤姆跳出飞机拍摄期间生成的飞机的“跟踪”动画必须与特技双动画混合，手动充足的动画。如果更新了跟踪的动画，则手动动画也必须如此。但是，在我们的管道中，这不会自动发生，并且混合动画的过程具有破坏性。″

"这就是哲学发挥作用的地方。也许这个过程可以重新设计，使其具有非破坏性？也许所有的 VFX 流程都应该这样设计？显然应该是这种情况，但实际的考虑因素阻碍了我们。其中最直接的是我们无法完全控制管道的工作方式。在大多数情况下，我们正在使用第三方工具，这些工具由软件供应商设计，用于执行特定功能，并提供不同级别的可定制性。修改这些工具以便以新的方式工作可能很困难。此外，执行这项工作的艺术家可能不习惯在他们职业生涯的大部分时间内使用的工具中采用非标准工作流程。是否“改变它”或“与它一起工作”的问题是 VFX 中一个重要的，非常重要的问题。″

"更糟糕的是，有许多这样的第三方软件工具 - 甚至简短的列表可能包括 Maya，Mari，Photoshop，Katana，RenderMan，Houdini 和 Nuke--每个部门都使用不同的组合。管道必须围绕这套完全不同的工具进行编织，其中许多工具的设计并不能很好地相互配合。这提出了另一个哲学问题：我们应该使用哪种文件格式在它们之间传输数据？什么时候应该写出本机数据，什么时候应该切换到与应用程序无关的东西？这些决策中存在许多固有的权衡，这些也会产生深远的影响。″

"与此同时，为了提高效率，FX 部门已经投入了自己的工具，用于在雪崩中包含的不同巨石之间进行切换。由于此工具管理文件的方式不允许将它们轻松转移到照明部门，因此这会产生一些自己的雪崩。我们通过帮助 FX 利用资产管理系统来纠正这种情况，这使得有关艺术家的事情变得更加复杂，但总体上会使事情变得平滑。″

"在管道的另一端，合成部门负责将所有真实动作和数字元素混合成无缝的最终图像，已经意识到它需要利用当前不存在的特定相机信息。写出来。这需要管道团队对资产管理系统进行更改并部署更新版本的摄像机导出/导入代码 - 但仅限于此特定节目。″

"这教给我们三件事。首先，VFX 管道需要灵活，因为它们将经历不断变化。其次，部门之间进行此类变更的能力差别很大。第三，人们经常乐于做出改变，虽然对他们有好处，但对下游有重大影响。″

"为了完成，我们的汤姆克鲁斯射击将通过十几个部门。总的来说，我们将在节目中制作 300 个这样的镜头，其中五六十个并行。我们还在制作另外三部电影，每部电影都有相似的尺寸和复杂程度。其中一些镜头将在生产中生产数月，其他镜头仅生产几天，但所有镜头都必须达到一致的外观和质量水平。汤姆在货舱中躲过子弹的镜头（我们加上枪口闪光和火花）需要与他最终摆脱手铐（天空更换）相匹配，而且他们都需要匹配雪崩，即使每个人的工作可能会间隔很多个月，由不同的艺术家使用不同的管道部分。所有这些都需要跟踪和安排。″

"在查看此特定镜头时，如果我们包含数字双倍的资产构建，我们可以轻松完成 200 个人日的工作。这需要在不到两个月的时间内完成，因为这个镜头是我们的客户在超级碗期间播放的预告片中的五个之一。电影本身还没有超过一年的电影院，但有一些像这样的早期交付。″

"我们对这部电影的工作总共约为 15,000 人日，我们在 9 个月内有一个最终交付截止日期，让工作室有时间“维度化”电影，以便能够以立体形式显示。假设一个月有 21 个工作日，这意味着团队平均人数为 80 人。但是，在整个过程中，八十人的工作极不可能。一般来说，参加演出所需的团队规模更像是钟形曲线，在制作开始时缓慢增加，然后迅速上升和下降。这意味着我们可能在生产期间达到约 250 人的峰值 - 所有人都需要在某个地方坐下，第三方软件的许可证

以及所需的其他基础设施都可以完成工作。″

"在开发过程中，这些镜头将分别送到我们的客户 10 或 20 次，并在内部进行数百次审查。我们的客户也将收到其他六个从事电影工作的工作室的更新，因此他们对自己想要提供数据的格式有自己的规格，这在生产过程中经常会改变几次。我们的管道不仅需要足够灵活，以应对工作室内发生的事情，还需要处理外部发生的事情。″

"因此，视觉效果管道并不完全是生产线。它更像是世界上最长的 Scalextric 轨道。你沿着它跑，切换车道和做循环循环。每隔一段时间你就会意识到你需要改变音轨的布局，但你很少被允许停车。你可以在其中一辆汽车中想象汤姆克鲁斯。但那是一部不同的电影。″

第 1.6 节游戏制作概述

"在 Fictitious Studios Inc，我们将近一年的时间开发我们新的 AAA 动作/冒险游戏 Final Violent Outburst XII：Unvengeance（或者只是团队所指的“Outburst”）。原型设计和规划阶段已经完成，并且生产正在全面展开。″

"Outburst 是一款非常标准的第三人称角色动作游戏。玩家穿过走廊，向人们射击很多，并观看切割场景，其中有疯狂的面部伤疤。有时他们被要求解决简单的谜题，虽然与这一类型的大多数头衔一样，但很少有人会对平均受过训练的仓鼠的智力征税。游戏非常线性，这使我们的生活更轻松，因为我们可以单独生成每个级别，而不必担心如果玩家决定徘徊会发生什么。″

"我们刚刚开始在 Level 5 上工作，过去是 Level 6，直到 4 级从设计中被切断，因为没有足够的时间来构建它。 （在此之前，它是 3 级，但是当设计师开始计划游戏的流程时，他们认为连续的套件太多并且移动它。因为我们都知道下周它可能会是 Level 8，每个人都称之为“摩天大楼”并继续他们的生活。）″

"毫不奇怪，它被称为摩天大楼，因为它设置在摩天大楼中。之所以这样，是因为这个故事需要在一个通用研究实验室中设置一个阶段 - 抱歉，这是一个遗传研究实验最初，我们谈到了将它变成一个海底复杂的坦克中的各种实验，但事实证明，由多边形制作死亡的团队（我们之前的游戏，在发布之前被取消）已经模拟了一座摩天大楼，并且从那时起看起来很紧张，管理层决定重用我们的能力。因此，一个带有朝东窗户和室外游泳池的顶层遗传实验室诞生了。″

"首先，我们从项目档案中检索旧模型的数据。我们在办公室走来走去说：“有没有人从 DBP 那里得到摩天大楼的副本？”大声地直到那个从事这个项目的人得到充分的骚扰才能把它从硬盘上捞出来。″

"当然，我们发现的并不是一个美丽的景象。我们没有一些文件的所有纹理（后来，我们会意识到这是因为创建它们的艺术家在游戏封装后离开公司，并且在任何人检查之前是否擦拭并重新分配了他的机器他忘了复制到网络上的任何东西都是 - 而且一切都是错误的大小，因为沿着这条线的某个地方，我们从使用米变成了厘米作为我们的基本单位。这不是什么小问题都无法解决，但这一切都加起来，而且在我们知道之前，一半的艺术团队花了两周的大部分时间来调整资源，重新设计和再出口资产。″

"所有这一切都在进行中，设计师们正在为这个级别构建粗略的“白盒”布局，编程团队正在为新的敌人（可怕的 Cyber -Apes-“半猿，一半）试验 AI（人工智能）机器人，所有陈词滥调“）。剪辑场景的故事板（以便利贴的数字照片的形式，在白板上四处移动的棍子）也开始出现在项目的网络驱动器上，因为写作和艺术团队进入了超速状态。″

"在此之后，事情已经平静了一段时间：团队形成了一堆小工作单元，每个工作单元都在查看不同的问题。一些艺术家开始为我们需要的额外风景和道具创作概念艺术和粗糙模型，一些艺

术家 - 设计师 - 程序员团体形成原型游戏和角色 AI，一般来说，每个人都对自己的小世界感到高兴。 （几乎在音频团队的情况下，这是在隔音办公室工作。）唯一的中断是当营销猛进时让我们知道明年在战场战斗系列中的发布会有十五个独特的武器，因为我们只有十四岁，我们最好在某个地方找到另一个，否则就有可能失去子弹点战争。″

"稍后一些仓促的会议，我们有一个看起来可行的计划：玩家可以从 Cyber -Apes 的头部偷走并向他们扔去的爆炸式大礼帽，造成巨大的伤害。这是营销的方框，同时将艺术要求保持在最低限度（一个额外的模型，没有动画），几乎不需要新的编程（就代码而言，帽子大多只是一个具有非常大的爆炸半径的手榴弹）并且不会使整个游戏失去平衡（因为你只能在这个级别上获得帽子，对已经完成的工作没有任何连锁反应）。″

"有了这个决定，我们欢快地前进，制作第一道艺术，测试代码和关卡设计原型。所以我们到了整合。″

"从理论上讲，整合是伟大游戏拱门的基石 - 所有艰苦工作汇集成一个光荣，功能齐全的整体。在实践中，它确实完全符合您对任何大型重型岩石的期望，当它放置在由半构造游戏制成的摇摆跨度之上时。因此，整合的诀窍在它发生时并没有在底层。″

"当我们开始整合时，很明显我们忽略了一些事情。从玩家导航的角度来看，设计师们构建的白盒级布局效果很好，但它们与艺术团队创建的模型并不相似。他们都没有考虑到这样一个事实：在某些时候（没有人确定何时，但似乎生产要求“让怪物更可怕”是罪魁祸首），Cyber -Apes 的规模增加了一倍，这意味着他们实际上并没有穿过关卡中的任何门道。最重要的是，剪辑场景的故事板已被重新绘制，现在包括与舞台中途的邪恶科学家相遇（在便利贴上的草图之外不存在的控制室），以及它事实证明，爆炸帽确实需要新的动画效果，因为当玩家使用标准的“投掷手榴弹”动画投掷一个时，他看起来真的非常愚蠢。″

"解决这一切需要几个疯狂的工作时间，但说实话，但比我们在时间表中允许的时间更长，给剩下的阶段施加更多压力。尽管如此，通过整合首过资产，我们终于可以看到隧道末端的灯光，并了解哪些区域需要最多的工作才能达到所需的质量。″

"通过手中的集成构建，进行了生产审查，产生了相当长的一系列要改变的事项：从使走廊缩短（不那么无聊“从 A 到 B”为玩家）到制作 Cyber -Apes 的一切在攻击之前大喊（让玩家有机会对来自他们视野之外的攻击做出反应）。虽然现在必须进行如此多的改动是令人讨厌的，但是在我们急于将所有内容集成之后，当资产接近完成时，更好地做到这一点要好得多。″

"进行这些变更与改进资产以使其达到类似可交付质量的过程同时发生。构建现在处于一个可以通过关卡进行播放的状态 - 虽然有很多房间都是由立方体构建而成，剪切场景由一个框架组成，上面写着“I AM CUT-SCENE 6”一个使用 MS Paint 的程序员，以及那些声音听起来很可疑的敌人，就像其中一个音频设计师用一种非常不感兴趣的声音说出“咆哮”这个词。″

"这意味着当资产被更改或添加时，我们可以在实际发生的环境中对它们进行审查，这使得微调更容易（尽管也比较慢，因为编译游戏就绪数据的过程，运行游戏然后导航到关卡中的正确点是非常耗时的，即使有调试功能和作弊）。每周审查会议开始发挥作用，并检查正在进行的整体进展，这提供了有用的反馈和强烈的激励，不是通过在会议即将开始之前添加错误代码来打破游戏的人。″

"随着时间的推移，越来越多的人提供反馈：生产，营销，测试部门，以及我们发现在前门外游荡的随机人员（也称为“焦点小组”）。有些被处理成变更请求或错误报告，这些变更请求或错误报告存储在我们的错误跟踪系统的相应数据库中;有些是非正式地处理 - 或者因为解决它需要不同团队之间的讨论，或者因为虽然 VP 的女儿喜欢恐龙非常好，但我们正在悄悄地忽略他在游戏中放入一些内容的指示。″

"不幸的是，我们不能忽视我们从执行制作人那里得到的（同样令人沮丧，但稍微不那么轻浮）的反馈，他们已经玩过我们的竞争对手刚刚发布的“战争之犬”，并决定我们应该瞄准更加

坚韧不拔的视觉风格。为了解决这个问题，我们使用屏幕截图或带有过滤器和手绘效果的游戏镜头制作了许多模型，并将它们呈现给他。这使我们能够超越他模糊的初步反馈，并回归他想要的风格的关键特征。由于此时的重建环境或角色会将项目推迟到很晚（如“超越时间表”）到很晚（如“死”），我们将这些讨论引向我们认为可以在合理的时间范围内实施的功能：当玩家发射或受到伤害时更多相机抖动，运动模糊滤镜从明亮物体产生光迹，并重新点亮水平以增加对比度。″

"幸运的是，我们能够实现每个人都满意的外观，因此水平逐渐完成。（当然，即使是现在，它也不是一帆风顺的，因为我们发现如果玩家在房间之间跑得太快，流媒体系统无法跟上，导致他们出现在一个没有特色的地方，我们必须加长走廊。当墙壁装载时，墙壁会慢慢地存在。）″

"另一个打嗝发生在其中一个出版商的其他动作标题被推迟时，危险地接近我们的发布日期并且意味着另外两个月从计划中消失，因为我们自己的发布被提出以避免冲突。幸运的是，我们可以通过删除我们设计但尚未构建的其中一个级别并将其添加到 DLC 来解决该问题：在游戏发布后将遵循的可下载扩展包。″

"随着整个游戏到达 alpha-beta-final 过程的开始，这些关卡逐个内容锁定。这促使人们急于提供任何缺失的资产。从理论上讲，在内容锁定之后，除了修复 bug 之外什么都不能添加，但是一如既往，我们最终会抛出一些占位符资产，以防发布者（或者更可能是测试部门使用内容锁定来完成）它自己的剩余密集测试阶段的清单抱怨后来出现的新事物。完成后，我们可以专注于最后一轮抛光和修复错误。″

"传统的是，在这个阶段，一些巨大的，事后看来，愚蠢明显的东西会破坏事物。在我们的例子中，它是屋顶游泳池。因为，在剩下的水平，玩家可以游泳，他们可以在游泳池游泳。如果他们在地面上，他们甚至可以在游泳时开枪。″

"然而，Cyber -Apes 无法游泳。这部分是因为他们沉重的机器人组件会导致它们下沉，但主要是因为没有人想到创建他们需要的 AI 或动画才能这样做。此外，他们没有任何远程武器。″

"鉴于这些事实，当测试团队的错误报告显示为：“观察到的行为：如果玩家进入房间，警告敌人，然后跑回游泳池，那么这不应该让人感到意外。然后跳进去，所有的敌人都会跟随并在游泳池的边缘排队，当玩家在闲暇时将他们射死时，他们看起来毫无意外。预期的行为：敌人不应该看起来无所畏惧，而是跳进游泳池并撕裂玩家四肢肢体。“″

"然而，它仍然是一个问题（或者，因为不喜欢“问题”这个词的制作人员倾向于说“问题”）。一方面，这可以被视为玩家的合法策略。另一方面，能够在不击中一击的情况下清除整个敌人的水平并不是一场伟大比赛的标志。你的游戏在互联网上传播的视频通常是好事，但不是当他们嘲笑你可怕的遗传机器人无法应对豪华水景时。″

"所以我们看看我们的选择。显而易见的答案是让 Cyber -Apes 有游泳的能力。然而，这将意味着一套全新的动画和人工智能代码（他们必须能够离开，进入游泳池 - 否则，我们将面临更加尴尬的风险“有多少网络 - 阿佩斯，你可以进入游泳池吗？“视频”。在开发的这个阶段，这将是非常昂贵的，所以它已经出局了。我们可以移除游泳池，但这意味着艺术的变化和一个奇怪的无意义的露台区域出现在它的位置。或者我们可以给 Cyber -Apes 一个远程攻击，但这将涉及更多的资产工作并重新平衡其余的水平。″

"最后，我们选择了一个廉价的黑客作为最简单和最安全的解决方案：程序员在 AI 代码中添加一个特殊情况，这样当玩家靠近游泳池时，所有的敌人都会停止追逐他们并返回他们的原始职位。这使得游泳池对于玩家来说是一个稍微不协调的“安全区”，但至少它不能被用来杀死敌人。″

"在随后的几周内，数据库中的错误数量逐渐减少，随着艺术家转移到其他项目或去度假，团队中留下的人数也随之而来。最终，只剩下少数人 - 主要是程序员修复硬件制造商的第一方

质量保证报告的最终错误，生产商试图按摩一系列 DLC 计划以匹配各种第一周的销售预测。然后，有一天，当几乎所有人都忘记了项目仍然存在时，从生产者那里收到一封电子邮件：″

第 1.7 节记住：每个产品独特的

第 2.1 节你将从中学到什么章

"在上一章中，我们简要介绍了一个重要问题：“什么是管道？”在本章中，我们将尝试给出更详细的答案，概述游戏或电影项目将要经历的主要阶段，并讨论每个人如何对管道和管道团队提出不同的要求。再次，我们将看看电影和游戏之间的相似点和不同点，试图突出每个游戏的独特要求。″

"在电影中，大部分挑战在于组织动画或视觉效果工作室本身以及在整个制作过程中提供内容的各种供应商生成的大量数据。在游戏中，挑战通常是管理游戏引擎。我们将研究为什么引擎是许多创造性决策和限制的基础，以及游戏引擎的数据需求如何导致与电影大不相同的管道。″

"我们还将探讨生产如何随着时间的推移而演变。“没有计划计划失败”的格言不能比电影或游戏更真实。视觉效果项目需要数百名艺术家 - 甚至数千名艺术家 - 需要数百名艺术家和开发者的设施和游戏之间的分割，单靠运气无法成功运行。应急计划避免了否则将不可避免地破坏项目甚至使公司陷入破产的问题。″

"但在描述生产阶段之前，让我们先看看生产经济学，并研究这些财务压力对管道的要求。″

第 2.2 节电影制作的经济性

"电影有时是非常有利可图的，但更常见的是风险很高的业务。对于一个主要的工作室来说，制作电影是一个昂贵的过程：在营销和发行之前通常需要花费 50 到 2.5 亿美元，这可以为总投资增加 1 亿美元甚至更多。″

"有些电影肯定是在主要工作室或通过主要工作室制作的。但是近年来，大型电影公司倾向于制作更多的大片或“帐篷”电影，并将市场的低端市场留给了他们的专业部门，以及通过低成本组织以低成本制作电影的众多独立电影公司。开销。″

"由于这种轰动一时的心态，成功电影的利润可能非常大。然而，2.5 亿美元电影的损失也很大。平均而言，约有 65-75％的电影会为投资者赔钱。大约 15-20％将收支平衡;只有 5-10％才有利可图。″

"为了有效地控制成本，电影公司齐心协力在提供优惠税收条件的州或国家制作电影，其中许多电影积极为电影制作提供税收优惠，以此作为刺激经济的一种方式。″

"直接结果是，许多视觉效果设施已在相同的区域设置中设置操作。除税收优惠外，这些通常还提供较低的人工费率。在许多这样的场所中，这种组合使得一个行业能够在以前不存在的地方成长。″

"该业务的风险性质也促使电影公司和制片人向提供电影服务的供应商施加压力，以降低其费率。大多数工作室的口头禅是以最低的价格获得最好的质量。″

"这适用于视觉效果设施，如果不是更多，则适用于其他供应商。业界对数字技术的使用意味着可以在世界任何地方进行工作，只要有必要的设备和合格的工人。

"这个结果是外包和离岸外包。视觉效果工作目前正在全球范围内进行，重点是提供税收优惠的地区。如今，西方的视觉效果设施不仅与当地竞争对手竞争，还与亚洲的公司竞争。″

"此外,制作电影不仅是一项昂贵的业务,而且是一项耗时的业务。开发电影资产的过程 - 从故事和人物的基础开始，通过脚本编写，预制作和预可视化，预算编制，拍摄，后期制作和视觉效果，以及分发 - 可能需要 36 个月或者更多。″

"然而，所有这几个月工作的商业价值仅取决于两个关键日：电影的开幕周末。来自主要电影公司的主要电影的发行模式在很大程度上取决于在这个开放周末最大化营销投资回报的需求，特别是在电影公司对电影在一段时间内的表现能力没有信心的情况下。行业语言，“有腿”。″

"无论是由于国家政治还是国际政治，还是仅仅是天气，这些重大的计划外事件都会对开幕周末造成严重破坏。虽然有些电影可以在这种情况下幸存下来，但很多电影都没有，而且大部分都没有完全弥补票房的损失。″

"开幕周末不能简单地放回去：发布日期是提前设定的，主要是因为需要在某些周末获得竞争优势，例如假期和学校假期。制作是为了满足那些铁定的目标，从理论上有足够的时间来完成电影开始 - 但不会在发布日期之前太远，因此工作室可以更好地管理其现金流。由于日期设定在目前为止，没有“明天我们将完成工作”。没有达到交付日期的组织通常不会在未来的电影上获得第二次机会。″

"VFX 通常是完成电影工作的最后一个主要部门，因此高效执行的压力是巨大的。对于那些进行尺寸化工作的设施 - 将电影从 2D 转换为立体 3D，这一过程经常与视觉效果的创建重叠并延伸，甚至更短，以及按时完成工作的压力甚至更大。″

"为了可靠地按时完成工作，同时保持速度尽可能具有竞争力，VFX 设施必须采取多种措施。这些包括但不限于：″

开发一个强大的管道，根据为尽可能高效地完成工作所需的许多任务进行组织

确保将整个工作流程传达给所有需要了解的人员，从内部人员到工作室的人员，这样就不会有意外的惊喜

与将要接收工作要素的各种其他供应商进行通信，以便能够相应地进行规划

使用 DAM（数字资产管理）系统，管理和项目制作人员可以随时查看正在制作的各种镜头的状态

让艺术家了解完成每个镜头所需时间的估计

在需要时将镜头的状态传达给工作室

以最佳方式维护所有设备

为正在生产的镜头类型制定基准（这使得工厂能够估计生产镜头所需的时间和材料 - 以及由此产生的成本，以便在将来的出价中使用）

根据出价跟踪完成所选镜头的成本，以优化出价过程，使设施能够竞争性地竞标以备将来的工作

第 2.3 节游戏制作的经济学

"自从他们主要由十几岁的男孩演奏以来，电子游戏已经发展了很多。根据娱乐软件协会的说法，今天的平均游戏玩家已有 30 年历史，并且已经玩了 12 年的游戏。现在，与 17 岁或 17 岁以下的男孩相比，现在有更多的成年女性玩游戏。这代表了一个多元化和规模庞大的市场。透明度市场研究预测，2012 年将从约 700 亿美元的行业增长到 2015 年的 1199 亿美元，其中亚太地区增长最快。″

"然而，视频游戏不仅是一个利润丰厚的市场，而且是一个不稳定的市场。 1972 年，Atari 发布了第一款主流视频游戏 Pong，整个行业诞生于此。在撰写本文时，Atari 正处于破产状态。前十大美国游戏发行商 THQ 已经完成了自己的破产程序，并且已经完全不存在，而 Square Enix 的总裁和 Electronic Arts 的首席执行官由于财务业绩不佳而退出了他们的职位。″

"直到最近，游戏行业的大部分收入来自高端个人电脑和家用游戏机系统（如索尼 PlayStation 3 和微软 Xbox 360）的所谓“AAA”游戏的销售。这些产品的开发并不罕见。游戏耗资 3000 万美元甚至更多。最昂贵的游戏预算可以与好莱坞大片 1 亿美元以上的预算相媲美，可能需要 3 年或更长时间才能开发出来。还必须考虑营销成本,这种成本经常与发展成本相匹配，或者在某些情况下会超过发展成本。″

"随着每一代游戏机的出现，这些成本会持续上升。预计下一代游戏机的开发成本将增加一倍，这可能是一个保守的估计：虚幻引擎的创造者 Epic Games 的 Tim Sweeney 被引用说，如果没有合适的工具，成本可能会增加五倍。支持发展。″

"由于少数游戏占收入的大部分，因此很明显 AAA 游戏的风险概况非常高。对于少数精英而言，利润可能是巨大的：根据其出版商 Activision Blizzard 的说法，使命召唤特许经营已经创造了超过 10 亿美元的收入。但这种成功是例外，而不是规则。″

"作为回应，许多游戏开发商和发行商都试图通过利用现有品牌来降低风险。这种“冲洗和重复”模式导致 AAA 市场以续集为主导，游戏设计缺乏创新。这些因素可能至少部分地导致游戏发行商的许多“旧守卫”的财务问题。″

"相比之下，市场的低端正在蓬勃发展。 Apple iPad 和三星 Galaxy 等基于互联网的平板电脑和智能手机的迅速普及，再加上 Apple 的 App Store，Google Play，Amazon.com 和 Valve Software 的 Steam 平台等新的移动和在线分销渠道，已经有效地消除了障碍“货架空间”，为小型工作室创造了许多机会。创新的企业公司已经避开了游戏机及其夸大的开发成本，并期待移动，社交和浏览器游戏以寻找新的受众。这些市场现在占整体销售额的比重越来越大。″

"在这个新的环境中，像愤怒的小鸟的创造者 Rovio 娱乐公司这样的公司已经能够以相当于 AAA 控制台游戏成本的一小部分产品获得高额回报，截至 2012 年 12 月，拥有超过 17 亿的下载量和 2.63 亿活跃用户.Minecraft 的制造商 Mojang 最初是一个团队，并已发展成为一家全面的出版商（该公司的网站列出了 23 名员工），2012 年的总收入超过 2.5 亿美元。″

"在这样一个多元化的市场中，很难概括出经济因素对发展渠道的要求。像电影一样，AAA 游戏要求开发人员管理大型数据集，跟踪复杂的生产工作，并严格控制成本。相比之下，独立游戏需要轻量级，灵活的管道，允许开发人员快速创新;而像免费在线游戏这样的新兴行业创造了他们自己的新需求，例如需要跟踪用户行为以微调从中获得收入的游戏内付款。″

"虽然可能无法预测视频游戏接下来会发生什么，但它们的潜力却是惊人的。移动设备的迅速普及正在创造新类型的游戏，新的商业模式和新的营销技术。现在几代人已经在游戏中长大，有一件事是肯定的：视频游戏就在这里 - 无论你是谁，在某些时候你都可能会玩一个。″

第 2.4 节生产阶段

"讨论制作过程的任务很复杂，因为视觉效果电影中使用的术语（CG 增强的实时动作），CG 动画特征（仅限 CG）和游戏（仅限 CG）沿着不同的路径发展，偶尔导致定义不同过程的相同术语。不熟悉这种细微差别的客户倾向于在“后期制作”一词下覆盖所有内容，进一步加剧了这种混乱。因此，在与客户沟通时，最好确认他们真正指的是哪个阶段。″

"在电影中，主要阶段是：″,
″前期制作″,
″生产″,
″后期制作″,
″在游戏中，主要阶段是：″,
″前期制作″,
″生产″,
″Finalling″

"我们将简要介绍每个阶段的内容。后期制作或演出是两个行业分歧最大的地方，电影通过输出线性系列的视觉效果镜头努力符合预定义的故事情节，而游戏提供动态和互动体验，引擎产生视觉效果作为回应对玩家的行为。″

第 2.5 节其他语言障碍

"在不同行业工作的艺术家不仅使用不同的技术术语。尽管他们需要在整个制作过程中有效协作，但艺术家，开发人员和管理人员可能会以完全不同的方式使一个关注图像美学的艺术家可能会描述创作它的过程与一个与模拟光的数学有关的程序员完全不同。作为管道开发人员，您需要了解两种沟通方式。″

"为了决定在与艺术家交谈时使用什么语言风格 – 或者在构建管道工具时要实现什么类型的界面 – 考虑我们将嬉戏地称之为“蓬松谱”。″

"蓬松的光谱是一端具有冷酷的技术逻辑，另一端是温暖，蓬松的创造力。计算机生活在光谱的冷酷，合乎逻辑的末端，而人类则占据了更温暖，更加蓬松的一端。在逻辑结束时，一切都是真或假，对或错，黑色或白色。在蓬松的一端，我们可以用情感或美学来描述事物。″

"当人们进行交流时，他们会有一定程度的“蓬松”，适合手头的任务。创建了令人惊叹的分形图像的程序员可能会描述以某种方式创建它的代码，以及他们在另一种方式中做出的艺术决策。我们使用的语言的蓬松程度取决于我们所谈论的内容，以及我们正在与谁交谈。″

"当然，蓬松的规模只是有点乐趣。但是，希望这也是思考沟通的难忘方式。你与同伴交谈的方式变得如此习惯，当你被要求与其他团体沟通时，往往很容易忘记他们可能不会分享你对术语的假设。″

"一个制作团队由许多不同的个人组成，从概念艺术家到程序员，每个人都以适合他们所做工作的方式使用语言。下次构建界面时，编译手册或编写电子邮件，说明新工具的工作原理，考虑其目标受众。管道应该弥合以非常不同的方式进行沟通的工作人员之间的鸿沟。″

第 2.6 节前期制作：概述

"预生产，也称为“pre-pro”或“pre-prod”，是项目的规划阶段，是未来一切的基础。预生产是确定工作的全部范围，并确定可能的故障点。执行不力的预生产倾向于为随后的所有事情定下基调，使接下来的几个月或几年的工作变成一场噩梦。″

"艺术部门在预制作中的目标是确定最适合电影或游戏的视觉风格。这可能包括创建概念图稿，构建测试资产或在拍摄中拍摄素材并对其应用不同的视觉处理。它还可能包括预可视化：创建动画或静止图像的过程，有助于定义电影的拍摄方式，以及它将需要的其他视觉元素。 （有关此主题的更长时间的讨论，请参阅 Interlude“Previs and Related Data”。）″

"这些探索性工作大部分是在纸上或使用数字绘画软件，以故事板和概念草图的形式完成的。其余大部分都可以通过现成的 3D 软件完成。但是，通常，某些任务需要管道团队的支持。例如，一组艺术家正在阻止拍摄 - 计划将拍摄的相机角度，以及将素材一起编辑的方式 - 可能会请求一个工具来帮助他们生成可以进行的预览渲染更快地交给编辑或合成师。″

"临时工具给管道团队带来了一个有趣的困境。如果这些工具过于笨重或功能太强，它们会阻碍艺术家们提供帮助 - 但如果它们的制造标准与传统生产工具相同，那么它们就会垄断少数程序员可用的时间。也很难以足够快的速度进行任何必要的改变以跟上预生产的繁忙步伐。″

"出于这个原因，许多工作室发现 TA（技术艺术家，游戏中常用的术语）或 TD（技术导演，电影工作中更常用的术语）- 具有生产工作所需的艺术敏感性但也有能力做自己的脚本 - 在预生产期间是非常宝贵的资源。″

"在技术方面，预生产也是风险最小化的时间：测试和验证来自创意团队的想法的时间，以确保它们可以在不破坏工作室或将其他人链接到他们的办公桌的情况下实现，工作墓地轮班。″

"在许多方面，预生产是管道团队项目中最紧张的部分。新的想法不断涌现，验证它们通常需要一个快速而肮脏的迷你管道，可以将它们转化为具体的视觉效果。作为概念草图或口头描述看起来很棒的想法往往在技术上是不可能的或艺术上无聊的。承诺最先进的新工作方式的工具可能会证明与您的其他管道不兼容。避免在不可行的想法上浪费不可替代的时间和金钱的唯一方法是尽早开始验证 - 这意味着原型管道必须在运行中进行组装。″

"原型设计通常类似于其自身的微型生产，包括用于创建测试内容的陪审团管道。如果游戏应该涉及一种新的流体模拟方法作为游戏元素，那么这个想法只能通过原型流体模拟引擎和管道来验证，以便为测试创建数据。″

"这是一个快速发展的过程。需求是模糊的，计划不断变化，新工具必须每天拼凑在一起。管道工作人员需要快速即兴地解决新问题，这通常意味着尝试常识和良好的编码实践会不屑一顾的事情。随着原型制作的结束，管道工作人员将开始将这种摇摇晃晃的代码重建为更严格的标准。由于原型代码是众所周知的不可靠，大多数公司的目标是完全重新实现他们的原型工具，而不是简单地构建它们 - 但在实践中，“临时”解决方案通常由于缺乏时间或资源而变得永久。″

"与此同时，必须制定“真实”管道的基础 - 一旦项目从预生产中移出，将由数十或数百名艺术家使用。当建模师和动画师到达时，他们将需要工具，文档和培训。如果他们完成工作所需的资源不是等待他们的，那么你还没有完成自己的工作。″

"还有一件事必须强调预生产：总是计划变革。然而，经过深思熟虑的管道就在这个阶段，随着技术的发展，导演会改变主意，或者设计师想出更好的游戏玩法，他们将会改变生产。构建管道，以便您可以容纳这些更改，而不会使您已经完成的工作无效，或者要求不必要地重新编写内容。在这个行业中，有一条简单的规则可以遵循：“唯一不变的是变革。”″

第 2.7 节胶片管道中的预生产

"在视觉效果中，术语“预生产”适用于相机开始滚动之前涉及的所有准备工作。故事板，预可视化和收集照明信息都属于这个标题，以及与预算，招聘演员，设置建筑和服装设计等非图形相关的任务。″

"对于艺术家来说，预生产期间发生的事情取决于项目的性质。常见任务包括视觉研究，角色和环境的概念设计，以及定义整个电影中使用的调色板。″

"接下来确定电影的整体感觉，包括编辑的速度和关键镜头的构成：例如，相机将放置在与角色相关的位置以及如何移动以捕捉动作。这可以在布局（CG 动画中使用的术语）或预可视化（实时动作中使用的术语）期间处理。″

"在某些情况下，这项工作的结果可能是一个完整的 3D 动画，使用低分辨率模型作为最终资产的占位符，与粗略的对话和声音效果一起编辑。在其他情况下，它可能是一个动画：一系列静止图像，如故事板框架，与最终镜头的正确时间一起切割。″

"预生产也是您定义生产期间需要创建的资产的时间点。对于完整的 CG 镜头，这将包括场景中可见的所有内容，包括相机，相机动画和数字环境。对于真人拍摄，它将包括一个拍摄板或绿色屏幕元素，可能需要进一步的油漆准备工作（油漆和准备：例如，在镜头中绘制出可见的电线或相机装备），rotoscoping 或匹配-移动。可能还需要构建集合的“数字扩展”。此外，某些类型的资产 - 例如,数字字符,其他前景模型和物理模拟 - 对于这两种类型的工作都是通用的。″

"对于开发人员而言，预生产可以确定管道中的任何主要差距，并修复可能导致生产停滞的任何问题。也许电影在人物工作上会很沉重，你知道你的索具或动画管道很薄弱。或者您可能需要实施一项新技术：例如，接收 Alembic 数据交换格式的文件，以便与另一个工作室合作拍摄电影。现在是时候进行尽可能多的准备工作，以确保您的管道稍后将处理负载。″

第 2.8 节奥运会筹备工作

"在游戏开发中，第一项任务通常是定义游戏的“概念”。这在正式生产开始之前进行，并在整个阶段继续进行。游戏的概念和类型将决定哪个引擎和管道选择最适合任务。″

"例如，并非所有游戏都有故事，因为它们将在电影中被理解 - 许多游戏通常只具有“进展结构”。如果故事对游戏很重要，那么就需要创建一个脚本，故事板和一系列规则来管理角色的能力和库存。所有这些都需要得到管道的支持。″

"可能还需要评估新技术。虽然大多数网络和手机游戏的制作速度更快，但 AAA 游戏经历三到四年的开发周期并不罕见，早期阶段涉及一系列强大的技术评估。例如，现在可以实时尝试由学术界开发的新图形技术，从而有助于塑造游戏的外观。或者也许其他游戏已经找到了从旧硬件中挤出额外性能的方法，要求您以新的方式创作内容。″

"同时，游戏设计团队将忙于新的游戏元素。这些元素需要进行原型设计，以便对其进行正确评估。如果没有具体的实现，通常无法确定新功能是否会使游戏失去平衡，或者甚至根本无法实现。″

"游戏预生产通常会产生两种不同的输出，称为“垂直切片”和“水平切片”。这些中的每一个都旨在为生产工作的不同方面奠定基础。″

"水平切片由原型环境中的一组广泛的游戏功能组成，称为“白盒级别”。白盒级别可能包括无纹理资产，缺少动画和非常粗糙的声音：它们的功能只是为了了解游戏将如何流动，或游戏

需要多长时间才能完成。结果是从开始到结束可播放的级别，使用占位符图片或占位符游戏元素（如谜题）进行阻止。由于游戏玩法驱动艺术，在预生产期间可以创建的水平切片越多，艺术内容的再加工在生产后期就越少。″

"相比之下，垂直切片由游戏的一小部分组成，通常是单个级别或级别的一部分，尽可能接近可交付的质量。其目标是为开发过程本身提供信心，作为管道的压力测试和艺术资产质量的基准。″

"垂直切片是生产的门户，安抚生产者并放松他们在支票簿上的把握。生产结束前的里程碑是该项目的主要决定/禁止决策点之一，并且有一个乏味的演示游戏很可能会被装罐而不是前进，或者至少被送回进入另一轮前期制作。″

"预生产结束时的一个共同目标是制作可播放的演示，可以公开发布以提升游戏的形象。这只能通过水平切片（最终质量游戏）与垂直切片（最终质量资产）的结合来实现。″

"理论上，在所有组件管道都到位之前，并非所有这些数据都可以编写。在实践中，有些不如其他人重要。一个典型的例子是所谓的“本地化数据”：创建适合特定语言的游戏版本所需的信息，包括屏幕上文本的翻译，新的录音，甚至语言特定的模型或纹理。然而，尽管本地化往往在开发后期完成，但在预生产期间对其进行规划至关重要，以便以后可以收集所有必要的资产。″

第 2.9 节生产：概述

"在生产阶段，工作重点从规划转向建筑。在 CG 动画和游戏中，屏幕上显示的所有内容都是计算机生成的，这意味着创建所有数字资产：模型，纹理，动画师操纵的控制装备以及动画数据本身。（相比之下，在视觉效果的世界中，术语“生产”通常仅指实时拍摄和现场数据的收集：数字资产本身是在后期制作过程中创建的。我们稍后将探讨这种区别。 ）″

"生产是团队规模扩大的地方。虽然前期制作通常需要数十位艺术家，但制作可能需要数百位。生产通常也会产生大量数据：通常比相对轻量级的预生产过程高出几个数量级。预生产是广泛的;生产是关于细节。″

"因此，虽然生产建立和存储资产的早期阶段对于游戏和 CG 动画来说都是相似的，但后期阶段差异很大。这种差异发生在个别资产组合成更大群体的时刻：在动画的情况下，当它进入镜头制作时;在游戏的情况下，当关卡创建开始时。″

"在这里，硬件限制开始发挥作用。即使像 PlayStation 3 或 Xbox 360 这样的现代游戏机在内存和系统资源方面与 PC 相比也非常有限。关卡设计涉及的众多考虑因素之一是如何将游戏所需的所有数据塞进可用的存储空间。超出预算，游戏根本不起作用。相比之下，来自特征动画的场景在大小上基本上是无限的：给定足够的计算机 - 并且足够的时间 - 可以渲染它。″

"良好的管道有两个关键特征，有助于管理完整生产的混乱：透明度和区域化。″

"管道中的透明度类似于文字处理器中的查找和替换功能：它使管道团队能够自动查找有问题的内容，并以最少的艺术家时间花费更新或修复它。在包含数十万资产的制作中，在不需要手动重写单个资产的情况下进行大规模更改至关重要。″

"例如，对生产中途发生的引擎（在游戏中）或渲染器（在电影工作中）的更改可能会改变资产的外观。这迫使团队调整他们已创建的资产，使其与新内容保持一致。手动执行此操作是一项生产灾难 - 整个艺术人员必须放弃当前任务并开始打开并重新导出数千个文件。另一方面，透明管道将包括用于自动查找和更新内容的工具或脚本。″

"成功的另一个关键是划分。团队的目标是建立一个自我维持的单元，每个单元都能很好地完成一件事，而不是创造一个巨大的，单一的工具和流程，这些工具和流程彼此密不可分。假设 - 经常发生 - 管道的变化要求动画需要以不同的格式存储。即使具有良好的透明度，重新出口

所有现有的动画也是一项严肃的任务，需要相当长的时间。但是，如果管道已经很好地划分，那么所花费的时间至少可以最小化。想象一下，同样的变化需要每个资产 – 无论是否动画 – 都要重新出口。这将使生产陷入停滞。″

第 2.10 节胶片生产线

"在特色动画中，制作有多个阶段。一旦简报和故事最终确定，高分辨率资产的制作就开始了。这包括模型，纹理，环境，装备和动画。一旦创建了这些资源，下一步通常是将这些资产组合成镜头和构建序列，准备渲染和合成。我们将在下一章中更详细地介绍这些单独的过程。″

"在生产阶段，管道开发人员最常见的三个任务是修复错误，添加新的工作流功能和优化现有功能。新功能可能是原始开发工作期间遗漏的细节，您最终有时间处理的有用的补充，或者是在实际生产环境中使用工具的方式所带来的细微变化。″

"优化也很重要，特别是如果工具比预期慢或者需要在比预期更多的镜头上使用。如果特定的计算过程非常慢，一种常见的解决方案是在渲染场而不是艺术家的工作站上运行它。这样可以释放工作站以获得更多有用的任务，并且意味着可以将作业分布在多台计算机上，从而缩短计算时间。″

"一旦确定了正在生产中正常执行的任务，通过软件实现自动化是有意义的。例如，动画师通常会导出几何缓存 – 数据文件，显示动画每帧中模型上每个点的位置 – 供照明部门使用。导出通常是手动完成的，并将动画工作站连接起来直到完成。自动执行任务以便只需单击一次 – 甚至可能在渲染场上执行整个过程 – 释放动画师以进行更多创造性工作。″

"如果事实证明您在预生产期间做出的假设是错误的，那么自动化通常也是必要的。例如，如果脚本只调用两个字符，那么假设动画师有时间手动导出缓存是合理的。如果在生产过程中将其更改为 200，那么自动化流程要好得多。″

"请注意，优化和自动化有助于尽可能高效地通过管道推送资产：它们不会改变其结构。一旦生产阶段开始，重要的是避免做出可能导致工作停顿的重大变化。虽然生产通常是设施最繁忙的时间，但它也是您最不可能与关键利益相关者讨论管道变化的时间。此时的重点是拍摄电影。″

第 2.11 节奥运会生产线

"游戏的制作阶段主要是制作内容：艺术资产，音频，游戏元素，关卡设计等。 （同样，我们将在后面的章节中更详细地介绍这些过程。）游戏团队将随着内容创建者转移到项目中而扩展，以满足所需的大量资产以及创建过场动画等自包含任务 – 预渲染的场景，有时也作为游戏的预告片发布 – 可能是外包的。″

"从理论上讲，系统编程和管道开发应该在生产前的预生产，内容创建和组装过程中完成，并在结束时修复剩余的错误。在实践中，这很少发生。如果增加到完全生产的过程显示任何弱点，管道可能会发生相当大的变化。游戏的运行时间也可能仍处于开发阶段。″

"但是，一旦投入生产，重要的是只关注最重要的变化：通常，修复产生最多投诉的工具。在构建新工具或修改现有工具之前，您应该评估这样做的相对成本：在优化次优管道时，实际上可能更好地使用暴力并在任务中投入更多艺术家而不是保持生产。″

第 2.12 节后期制作或收尾：概述

"直到后期制作（通常简称为“后期”）或结束，电影和游戏管道非常相似。在这个阶段，他们发生了根本性的分歧。在一部动画片中，这部电影将近乎完整：最终的影像将被渲染出来，并在将它们移交给客户之前进入工作的最后阶段。（视觉效果也是如此，我们将在稍后讨论。）但在游戏中，一些最技术性的工作尚未到来，因为错误得到修复，艺术或游戏玩法问题在决定期间得到解决。″

"后期制作和发布会的共同点是它们代表了改进产品的最后机会。当观众观看您的作品时，如果您有更多时间，没有人会关心您可以做得更好。这个阶段是你最后的机会，可以进行那些可以拯救失败的电影，或将一个好的游戏变成一个伟大的游戏。″

"即使一个项目最终完成，疲惫也没有休息。这是对节目进行尸检，评估哪些进展顺利，哪些进展不顺利,以及将这些课程应用于下一个项目的时间。现在也是进行“蓝天”工作的时候了，比如评估新的图形技术;并且在生产过程中使更改变得过于根本，例如切换到新的数据库系统。这是管道开发人员应该最繁忙的时候 – 不是后期制作中的项目，而是关于即将进入建筑物的项目。″

第 2.13 节胶片管道的后期制作

"严格地说，后期制作仅指在创建最终图像后执行的任务。根据这个定义，唯一真正的后期制作任务是编辑，尺寸化（DM），数字化素材以创建数字中间片（DI），色彩校正或“分级”，母带制作和胶片输出：打印数字图像拍摄电影以分发给电影院。这是该特征动画中常用的术语。″

"然而，在真人电影制作中，这个术语的用法通常不同。大多数视觉效果专业人士使用术语“制作”仅指实时拍摄和集合中记录的任何数据的集合,而“后期制作”则指代之后的任何内容 – 包括大部分视觉效果工作。″

"根据这个定义，大多数艺术团队的工作都是在后期制作过程中完成的。在动画片制作过程中发生的同样的资产创建过程 – 建模，纹理绘画，装配，动画等等也会在真人电影中发生，但在后期制作阶段，作为视觉效果。同样，我们将在下一章中更详细地介绍这些单独的过程。″

"但无论您是将资产创建视为生产还是后期制作的一部分，帖子都是您必须优化所创建图像的最后机会。许多设施从未使用“最终”一词来描述一个镜头，直到它被移交给客户：他们认为，如果时间允许，他们可以继续调整它直到交付截止日期，并且，相反，它应被视为“CBB”（可能更好）。″

"这些调整是否真正可以实现三个限制：预算，时间和连续性。连续性非常重要，因为镜头不是孤立地批准，而是在整个电影的背景下，因此对一次拍摄所做的改变可能需要在其他镜头中进行进一步的改变。″

"出于这个原因，通常更有效地移动尽可能多的镜头来完成，而不是专注于抛光单个镜头。最好让整个序列“完成 75％”，并在上下文中查看所有镜头，而不是单次拍摄“99％完成”，但是可能会花费时间将其余部分提升到相同的质量标准。″

"您是否有时间进行更改可能取决于反馈循环中涉及的视觉效果过程的步数。更改镜头的光照比更改其布局更容易，因为前者只需要再次渲染和合成，而后者可能还需要您更改光照和效果模拟。″

"因此，电影往往逐个部门退出生产部门：例如，布局管道通常会比照明和合成更早地安顿下来。在大型制作中，工作在多个设施之间分配，每个设施都有自己的专长，设施本身可能会发

生同样的事情：例如，专门从事流体模拟的公司将比专门从事尺寸化的公司更早完成。″

第 2.14 节奥运会决赛

"验证游戏的过程反映了软件开发的过程，其中几乎完成的产品被转交给 QA（质量保证）部门以进行“防弹”。除了开发人员自己的内部质量检查外，控制台标题还必须通过控制台制造商运行的正式 QA 流程，以防止低质量游戏严重影响硬件。但与防弹不同，finalling 受到艺术和技术问题的驱动：以及代码的质量，QA 团队必须考虑与其一起发布的数据的质量。″

"在此过程中，游戏应尽早打包并在最终媒体上进行测试。掌握 - 在 DVD 或 Blu-ray 上布置资产的过程 - 可能会引发一些之前并不明显的问题：尤其是文件可能只是太大而无法适应。可能还需要为每个要分发游戏的区域刻录单独的光盘，每个区域包含一组不同的本地化数据。这些应该单独测试，因为只有在某些条件下才能显示出错误。″

"尽管对文件大小的限制可能不是绝对的，但同样适用于要以数字方式分发的游戏。糟糕的数字发行可能会破坏产品的盈利能力：如果用户下载游戏的时间超过几个小时，许多人根本就不会打扰。为了解决这个问题，一些公司使用实时流媒体解决方案，使用户能够在整个游戏下载之前开始游戏。这种分发模型应该从第一天开始构建到管道中，因为一旦游戏完成就实现流式传输系统非常容易出错，并且可能需要对资产进行大量的重新设计。″

"在决赛开始时，游戏将完成，但有很多错误。其中一些将反映代码问题，导致崩溃或帧速率不佳。其他人将反映艺术资产的问题：例如，物理网格中允许角色穿过世界的洞，或者内部翻转的单个多边形，导致图形故障。必须确定并修复所有这些问题。″

"这通常是通过一系列正式步骤完成的，通常称为“里程碑”。虽然确切的定义因开发人员而异，但通常会认识到以下里程碑：″

Alpha：Alpha 意味着游戏正式完成，理论上可以一直播放。 大部分作品都是占位符，性能通常远低于预测的帧率，并且会有很多错误，但游戏仍然以骨架形式存在。 Alpha 通常是内部里程碑，在工作室和游戏发行商之间进行协商。

Content complete：内容完整通常意味着占位符已被最松散的术语替换为可交付 - 尽管不是最终资产的抛光或无错版本。 对内存和运行时性能的现实限制现在得到了尊重，为了提高效率，早期资产经常被大幅削减。 故事弧和游戏的基本流程现在被视为锁定; 图形调整仍在进行，但（理论上至少）不会添加新的过场动画，角色或游戏区域。 一旦项目达到其内容完整的里程碑，一些生产人员开始转向其他项目是很常见的。 一些艺术家仍在进行错误修复或抛光，但重点是完善现有资产，而不是创造新资产。

Beta：通过测试版，所有游戏内容都是完整的，游戏处于可交付状态，无论是在资源使用方面还是在资产质量方面。 还有一些问题需要解决，但游戏现在基本上是最终版本的错误版本，而不是它的建议。 Beta 也是该项目的游戏测试和公众示范达到顶峰的时候。 专业测试人员，无论是内部还是由出版商提供，都将持续不断地进行游戏，探讨缺陷和不一致之处。 有些游戏还有“开放测试版”程序，允许一组精选的客户玩游戏：部分是为了产生积极的口碑，部分是为了获得可以帮助调整设计的最后一分钟反馈，以及 部分是为了发现专业测试人员漏掉的错误。

"对于内容团队而言，测试阶段是优化，抛光和错误修复的混合体。优化资产以使其在运行时更有效率可能会很痛苦，因为将资产装入较少量的内存通常会导致视觉质量下降。因此，管道应该提供分析和跟踪工具，以帮助确定“低悬的果实”：可以完全削减的资产，因为它们未被充分利用，或者那些内存占用与其真正重要性不成比例的资产。″

"虽然通过测试版没有占位符资产，但通常可以给游戏带来一些图形润色。现在可以调整匆忙构建的资产，或者没有适当的上下文，以匹配游戏的整体感觉。由于到目前为止应该很好地理解引擎的局限性，因此通常可以以相同的运行时间成本创建外观更好的资产。″

"当然，在测试期间处理错误仍然是最重要的。由于反馈的泛滥，这是良好沟通最重要的一点，以指导（缓慢缩小的）生产团队解决正确的问题。管道的错误跟踪，任务分配和错误检查功能在测试期间进行了测试。″

Final：决赛阶段往往是情绪上的反击。 随着游戏走向发布商的最终认证 - 从而进入制造或在线发布 - 通常会锁定内容：不会在游戏中添加新资产，而只允许修改报告的错误。 这部分是因为资产可以签署（特别重要的是有执照的头衔，可能需要经过几轮批准），部分原因是任何变化，无论多么无害，都会产生非常真实的产生爆震的风险 - 臭虫。 （模型使用的额外内存字节可以说是完美运行且不断崩溃的游戏之间的区别。）

"随着时间的推移，允许越来越少的开发者接触游戏。在游戏获得最终认证之前，大多数艺术家已经转移到其他项目 - 或者花费很长时间与他们的家人在一起 - 并且团队被简化为有权在最后时刻制作的高级人员的骨架变化。游戏已经“淘金”并且现在已准备好进行复制和分发的消息经常到达一个几乎被遗弃的办公室。″

"当然，在传统意义上，整个游戏类型都不是“最终的”。持久的在线游戏，从“魔兽世界”到适度的教育游戏等泰坦尼克号制作，都在不断更新和扩展。随着持久性世界项目接近主要版本 - 可能是引擎更改，特殊的游戏内事件或新游戏区域的开放 - 它与传统的盒装产品一样经历了相同的质量保证流程。″

"凭借其非常长的寿命，以及不断更新的细微更新，管道本身就像在线游戏。一些开发人员对于今天的失误可以在下周的更新中得到纠正的知识感到欣慰，但更聪明的公司意识到遗留问题可能会在很长一段时间内成为负担，并进行相应的测试。管道开发人员在较小规模上面临同样的问题，他们可以从他们的例子中学习。″

插曲：previs 及相关数据

"Previs 在九十年代早期通过视觉效果进入电影制作世界。我们的想法是利用相同的高端计算机图形硬件和软件，用于在后期制作中创造突破性的视觉效果，但在预生产的早期阶段利用这种力量作为设计和规划工具。观看低分辨率模型和简单动画，电影制作人可以参与迭代设计过程，测试想法，看看哪些有效，哪些无效，早在视觉效果的艰苦工作开始之前。″

"Previs 遵循今天完全相同的过程，但计算机硬件和软件的进步使得预演艺术家能够将他们的技能应用于电影制作过程的更大部分。最初只有一两个预制艺术家从事几个镜头的工作现在是一个高度专业化的团队，由十几个或更多的艺术家组成，他们工作了几个月，创造了令人难以置信的复杂动画，复制了电影的动作，外观，情绪和基调。在一个典型的 VFX 驱动的帐篷上，预览团队可以在一个快速，迭代的环境中可视化一小时或更长时间的电影，这使得电影制作人可以轻松地进行更改并尝试新的想法。″

"无论是一部轰动一时的工作室画面还是更为适度预算的电影，previs 遵循相同的基本步骤：″

创建准确但低分辨率的必要资产 3D 模型，如角色，生物，集合，道具，车辆等。
动画动画
添加相机以覆盖操作
使用硬件或软件渲染输出摄像机
编辑 coverage 以创建动画
如果需要，添加声音效果
复习，修改和重复！

"这项工作经常在电影制作办公室完成。靠近创意决策者是必不可少的，因此像 Proof 这样的预览公司将嵌入一个艺术家团队与电影制作人一起工作。如果制作旅行，那么预览团队会与他们一起旅行。这可能意味着要去一个不同的城市，但它也可能要求艺术家前往不同的州或国家。因此，预览团队保持较小的足迹非常重要，这样他们才能保持敏捷并轻松移动。″

"典型设置包括每个艺术家的工作站，共享服务器，备份驱动器和网络硬件。工作站装载了艺术家可能需要的所有软件以及有用的3D模型和动画库。大多数故事片预览是使用Maya创建的，但也可以使用其他建模和动画包。工作站还将配备 Photoshop，Illustrator 和视频编辑软件。较大的预览团队将经常保留一些额外的工作站，可以作为渲染农场或编辑工作站。整个设置可以在几个小时内取下并重新设置。″

"Previs 在电影制作中占有独特的地位。它存在于各个部门之间的界限上。为了有效，预览团队需要准确地表现和反映制作在拍摄时将遇到的具体情况。例如，艺术家使用的预览相机必须符合实际胶片相机的规格，包括光圈大小，宽高比和镜头。设置需要匹配艺术部门设计的那些，并且需要在现实世界中存在的位置。″

"为实现这一目标，预览团队需要前往各个电影制作部门，并收集尽可能多的信息。并非每个产品都包含此列表中的所有内容，并且某些数据仅在主要摄影开始后才可用，因此信息收集是一个持续的过程，在数据可用时合并。以下是 previs 团队收集的内容及其来源。″

制作（导演和制片人）
脚本页面（PDF;约 10 MB）
物理和/或数字形式的故事板（JPG，PDF;约 250 MB）
临时乐谱或音效的音频文件

艺术系
概念艺术（JPG;约 100 MB）
参考图像（JPG;约 100 MB）
位置照片（JPG;约 100 MB）
声场的尺寸（DWG，PDF;约 10 MB）
物理和/或数字形式的 2D 设计（DWG，PDF;约 50 MB）
环境，道具，车辆等的 3D 模型（各种 3D 格式，包括 SKP，OBJ，3DM，MA;约 500 MB）

相机部门
用于拍摄的相机规格（例如帧速率，宽高比，分辨率）
相机光圈（胶片）或传感器尺寸（数字）和完整的镜头列表
任何特定摄像设备的详细信息，如起重机，臂架，电缆凸轮，运动控制装置等。

视觉效果
激光雷达数据（OBJ 或 MA;约 1 GB）
来自 VFX 供应商的人物，生物，环境，道具，车辆等的 3D 模型（MA，OBJ;约 10 GB）
来自 VFX 供应商的哑光画（PSD，TIF;约 500 MB）

特技
排练镜头（MOV;约 100 MB）

社论

粗略编辑（MOV;约 50 GB）
主要摄影板（MOV;约 100 GB）

"此外，在初始数据转储期间，存在关于特定困难或生产预期问题的对话。这些问题可以告知 previs 团队首先接近的镜头或序列。有了所有可用信息，团队就开始了建模，动画，渲染和编辑的过程。虽然根据镜头的复杂程度而变化很大，但是四个艺术家团队每周制作 1-2 分钟的动画并不少见。在整个制作过程中，可以转换成数千个单独的镜头和数百个需要 TB 级存储的编辑序列。″

在单个项目的过程中，预览团队的基本输出是：
场景文件（MA;约 500 GB）
电影（MOV;约 250 GB）
图表（PDF;约 100 MB）
KeyFrames（JPG，PDF;约 100 MB）
3D 模型（MA;约 10 GB）

场景文件：脚本或故事板中描述的操作分为几秒到几分钟。 每个部分都是动画的并保存为 Maya 场景文件。 在该场景文件中，将包含所描述的动作的本质的所有角色，生物，道具，车辆和环境。 场景文件还将包含许多摄像头以覆盖动作。 在某些情况下，每个场景文件可能只有一个或两个摄像机，而在其他情况下可能有几十个。

电影：Maya 场景文件中的每个摄像机都将作为 Quicktime 影片输出。这些电影也称为镜头，其持续时间从几帧到一分钟或更长。 一起编辑镜头以创建持续几秒到二十多分钟的序列。 声音效果和临时分数通常会添加到编辑中，以帮助传达情绪，语气和情感。 电影制片人审查序列，预览团队进行修改，直到工作获得批准并分发给相关部门负责人。 副本可能会发送给工作室主管或负责最终工作的 VFX 供应商。

图表：在某些情况下，镜头表示为图表，在 2D 程序（如 Photoshop 或 Illustrator）中创建。 图表基本上是描述摄像机位置及其行进路径的蓝图。 图表还可以记录演员或车辆的距离，速度，位置以及任何其他相关的生产数据。 拍摄图很有用，因为即使在智能手机和 iPad 时代，分发，查看和注释一张纸也比使用 Quicktime 文件容易得多。

关键帧：编辑后的序列经常缩减为一系列静止帧，因为 PDF 文件更易于分发，共享和打印。关键帧经常在现场看到，有时贴在大板上，用作当天拍摄时间表的指南。

3D 模型：随着项目的结束，预览团队将打包单个资产以及完整的场景文件，以便交付给各个 VFX 供应商。 在某些情况下，视觉效果将在项目开始之前授予供应商，并且预览团队可以与供应商合作以确保预览和供应商管道之间的场景比例和方向一致。 这种情况很少发生，所以 previs 团队竭尽全力确保场景文件干净整洁，以便任何继承工作的人都可以更轻松地使用它。

"在电影制作环境中，DAM（数字资产管理）的方式很少，无论是监督数据共享过程的人还是跟踪版本和变化的软件。随着数字工具变得越来越强大，随着艺术家变得越来越复杂，数据共享将有更多的机会。 3D 模型已经在 previs 和艺术部门之间轻松和频繁地移动。视觉效果供应商偶尔会将资产转移到 previs，作为测试设计和确保 previs 符合其管道规格的方法。随着这种跨部门合作与协作的不断发展，数字资产管理者和 DAM 软件将成为电影制作过程中不可或缺的一部分。″

电影资产创造

第 3.1 节您将从本章中学到什么

"在上一章中，我们研究了不同的生产阶段。在本章中，我们将重点介绍创建资产的过程：构成电影或游戏的数字元素。虽然两个行业的资产创建方式相似，但也存在重大差异。出于这个原因，我们只会看这里的电影作品。在下一章中，我们将探讨游戏管道的不同之处。″

"电影的资产开发分为三个生产阶段。概念艺术，maquettes（人物或生物的物理雕塑，用作概念工具，或用于 3D 扫描），故事板和预览都是在预生产期间创建的。接下来会发生什么取决于电影是动画特征还是具有视觉效果的实时动作。″

"在视觉效果影片中，在制作期间创建的唯一资产是在集合中捕获的资产。这些可能包括 LIDAR 环境扫描，照明数据和参考摄影。在特征动画上，生成是创建大量数字资产（如模型，纹理，装备，FX 和动画数据）的地方。视觉效果项目需要一组类似的资产，但正如我们在前一章中所讨论的那样 - 这项工作通常在后期制作期间进行。此外，VFX 项目需要具有动画功能的资产类型，例如摄像机跟踪数据和准备好的背景板。″

"在本章中，我们将简要介绍如何创建这些类型的资产。而不是为每个过程提供详细的指南，目的是从管道设计的角度提供高级概述，并介绍您在本书后面可能遇到的技术概念。″

"我们还将列出用于这些任务的一些关键工具。虽然许多大型设施依赖于他们自己的专有软件，但其他大型设施使用商业工具，通常通过自定义插件和脚本进行扩充。本章并非目前可用产品的完整列表：仅为最常用于特征动画和 VFX 的简要指南。″

"许多工厂使用具有广泛功能的应用程序作为其主要 3D 包,将其用于资产创建中涉及的几个任务。这种“全能”软件包包括由 Autodesk-Maya 开发的软件包，3 ds Max 和 Softimage，它们占据了这个市场的大部分份额 - 以及来自 Houdini，LightWave，EIAS 和 Cinema 4D 等小型开发商的软件包。然而，设施运行“异构”管道越来越普遍，这些管道汇集了许多针对个别任务量身定制的专业工具。″

在开始之前，列出我们将在此处查看的任务可能会有所帮助：

在集合上进行：

数据采集（激光雷达，照明数据，参考摄影）在中进行

设施：

处理实时素材（制版准备，旋转扫描，相机跟踪）

造型

纹理和着色器创建

布局

索具

动画

FX

灯光

合成

"这是实际生产管道的简化版本：有关更详细的版本，请参阅第 1 章中的流程图。虽然任务按大致时间顺序列出，但重要的是要注意许多重叠;而且不是一个线性过程，资产开发是周期性的，由一系列迭代循环组成。其中最重要的一个循环是“外观开发”或“外观开发”，其中角色，生物和道具的视觉风格 - 包括它们的几何形状，纹理，头发和毛发，有时还包括它们的角色的

运动范围钻机能够生成 - 在将资产传递到镜头布局之前进行细化。″

"现在让我们更详细地看一下每个任务。我们将从生产过程中最早发生的那个开始：收集数据集。″

第 3.2 节激光雷达和现场测量数据

"诸如“全站仪”（用于测量角度和距离的测量仪器）和 GPS（全球定位系统）之类的测量工具已经在电影工业中使用了几十年。然而，随着 20 世纪 90 年代末 LIDAR 扫描仪的推出，地理空间数据现在可以更有效地用于视觉效果镜头的制作。″

"激光雷达扫描涉及使用激光源测量从固定位置到周围刚性表面（例如建筑物）上的点的距离。这些数据存储为“点云” - 三维空间中的一系列断开点 - 可以转换为该组的三维模型，并可用于为视觉效果镜头创建精确的数字背景，设置扩展或者提高相机跟踪的准确性。″

"但是，LIDAR 数据不是集合上捕获的唯一数据。照明部门可能需要镀铬或灰球数据（在集合上拍摄的反射和哑光球的图像，用作试图以数字方式重建相同照明条件的艺术家的参考）和 HDRI（高动态范围图像）。数字摄影还可以用于捕获稍后将用于创建纹理贴图的图像 - 或者仅用作附加参考数据。″

"执行 LIDAR 扫描时，捕获任何相关元数据非常重要。在适当的数据输入表上，使用笔和纸或数字手持设备在设备上收集大部分辅助数据,例如“拍摄”扫描所代表的数据。在每天结束时，必须对这些数据进行排序并输入到现场数据库中。″

"其他原始参考材料必须以类似的方式进行分类和转移。用于收集参考材料的摄像机必须正确校准，以便时间戳匹配，理想情况下使用同步时间码。这可确保数据与其他参考源正确同步，例如用于从其他角度记录集合的“见证摄像机”。由于相机还可以记录拍摄照片的位置，因此可以及时定位参考材料和 3D 空间，从而可以执行以下查询：“显示三天前在 100 米范围内拍摄的所有图像点。”″

"包括已知尺寸的参考对象通常是有用的，以给出被扫描的集合的比例的概念。但是，LIDAR 扫描仪是测量级仪器，因此在设置过程中有时会自动处理：Leica Geosystems 硬件具有非常强大的校准程序，因此只要处理软件设置为已知的测量单位，如英尺或米，规模是正确的。在处理软件和供应商的艺术软件设置为不同单元的情况下，在最终数字数据中具有参考对象作为最终“感测检查”也是重要的。了解艺术软件的坐标系也很有用：例如，Maya 喜欢在 Y-up 坐标系中工作（其中软件中的 Y 轴代表垂直高度），因此数据被转换为 Y-坐标空间。″

"您还应按照视觉效果工作中可能的重要性对要扫描的区域进行优先排序。如果时间有限，您可能需要考虑采用“选择性分辨率方法”，以高分辨率捕获重要区域;其他人分辨率较低。但是，请尽可能以高分辨率捕捉整个场景。以后很容易降低数据的分辨率，但反过来是不可能的！

"扫描完成后，下一步是制作扫描的备份副本。始终保留原始数据的副本，直接将其发送到存档而不进行任何处理。但是，LIDAR 技术人员应该意识到他们的扫描仪能够记录数十亿个点，从而产生数十亿字节的数据。因此，以多种分辨率将数据传送到视觉效果设施是有帮助的。这可以通过“抽取”算法来实现，该算法减少构成从点云数据生成的 3D 表面的三角形的数量，同时仍然保留其整体形式。最大的挑战是正确对齐各个扫描。大多数激光雷达制造商为这类工作提供某种软件，尽管目前还没有完全自动化的解决方案。您的最终目标是以一种形式提供数据，使视觉效果艺术家能够尽可能快速有效地完成工作。考虑提供三个版本：10％抽取，50％抽取，以及原始高分辨率扫描数据供参考。″

"由于人类记忆迅速消失，组织和管理所有这些数据是数据采集过程中非常重要的一部分。考虑到典型调查所产生的信息量，如果没有关于如何将数据输入到现场数据库的严格指导，则几

乎不可能找到相关信息。最重要的是定义文件命名约定，以避免在项目进展时出现任何混淆。当涉及到集合本身的扫描时，这通常是相对简单的，但是当涉及到道具或演员时也可能成为挑战，也可以在集合上进行扫描。″

"一旦交付给 VFX 设施，所有这些数据都必须使用，以便它以便于员工导航的形式出现在设施的资产管理系统中。这涉及将数据从现场数据库（按日期排序）重新映射到设施的数据库，该数据库通常由镜头或资产类型构成。我们将在后面的章节中更详细地探讨数据结构的问题。″

"有关 on-set 捕获的更详细讨论，请参阅 Interlude“LIDAR 资产捕获集合”。″

第 3.3 节匹配移动、旋转镜检查和板材制备

"匹配移动，rotoscoping 和制版准备发生在管道的早期，准备板材供 3D 艺术家和合成师使用。虽然在资产创造过程中很早就开始实施，但它们可以对成品拍摄的成功产生很大影响。″

"匹配移动是在 3D 艺术软件中创建数码相机的过程，该数码相机匹配真实相机的移动（平移和旋转）。然后，工作室创建的数字资产可以通过此数码相机呈现，确保它们无缝匹配实时素材。这个过程是使用图像处理技术自动完成的，通常需要手动调整自动“解决”;但非常复杂的镜头可能需要逐帧手动匹配相机位置。通常用于匹配移动的工具包括内置于设施的 3D 或合成软件中的工具，以及 3DEqualizer，boujou 和 SynthEyes 等专业应用程序。″

″Rotoscoping（或“roto-animation”）要求艺术家在真人版块中的关键元素（如角色，道具或风景）周围绘制或跟踪遮罩（平面遮罩）。这些元素通常被划分为表示距离相机不同距离的物体的层，从而可以轻松地将渲染的 3D 元素合成到板中。例如，要在背景场景和前景角色之间插入渲染图层，在角色周围旋转角色，将其与背景隔离。这允许角色在渲染层上合成。常用于 rotoscoping 的工具包括内置于 Nuke 和 Fusion 等合成软件中的工具,以及 Silhouette 和 mocha 等专业应用程序。″

"板制备（或简称“制备”）是制备用于合成的活动板的过程，并且包括任务的混合。最常见的是钻机和导线去除（在镜头中绘制出可见的结构，特别是特技和摄像机装置的部分），去除噪音，灰尘和胶片颗粒，并消除镜头变形（通过移除镜头来“平整”板相机镜头引入的圆形失真，最明显的是在图像的角落）。这些任务使得合成器更容易使用。然而，在最终复合材料中必须重新引入这些元素中的一些（特别是噪声，胶片颗粒和镜片失真），以避免 CG 图像的不自然的完美外观,确保具有视觉效果的镜头与没有的镜头相匹配。设施通常使用相同的工具进行板材制备，就像它们用于合成工作一样。我们稍后会讨论合成。″

"建模是创建用于动画和视觉效果工作的数字 3D 几何的过程，称为“模型”或“网格”。通常，建模者只负责重新创建对象的整体三维形式：其表面属性（包括其局部颜色和小的不规则性，如凸起和划痕）由纹理贴图和着色器表示 - 我们将在稍后讨论本章。″

"除了准确地重新创建对象的形式之外，建模者还负责确保其拓扑是正确的。这包括确保构造网格的多边形遵循几何的底层“流动”，并且具有合适的尺寸和形状。建模实践在不同设施之间略有不同，但一般来说，建模人员会尝试确保多边形的“密度”在网格表面上大致均匀，并且模型主要或完全由四边形多边形构建，或者“四边形”。包含不均匀大小的多边形或大量具有四个以上边的多边形的网格可能会导致表面纹理扭曲，或者在动画时可能会奇怪地变形。″

"可能还需要确保模型保持低于最大多边形数，以便更快地进行操作和渲染。这个问题在游戏中比电影作品更重要，因此我们将在下一章详细研究它。″

"可以通过扫描真实对象或通过从参考照片重建几何图形来半自动地创建模型：称为“摄影测量”的过程。同样，我们将在下一章中更详细地讨论这个问题。″

"然而，动画和视觉效果工作中使用的大多数模型都是手工创建的，或者至少是手工修改的。用于此工作的工具根据资产是“硬表面”模型（例如建筑物或车辆）还是有机模型（例如角色或

生物）而变化。为了创建硬表面模型，设施倾向于使用其主要 3D 艺术包或专业 CAD（计算机辅助设计）软件中内置的工具。有机模型倾向于使用专业应用程序创建，提供更像传统雕刻的工作流程，包括 ZBrush 和 Mudbox。″

第 3.5 节阴影和纹理

"为了重现对象的外观，3D 软件依赖于着色器：确定如何渲染该对象以生成最终图像的计算指令集。着色器模仿材料对光的反应方式，创建表面颜色和表面反射率等属性的数学表示。它们的范围从非常简单到令人生畏的复杂，并且可以准确地再现光的行为或创造完全由艺术和想象驱动的效果。″

"由于仅仅通过编写代码描述材料属性在对象表面上变化的方式将非常耗时，3D 软件使艺术家能够创建“纹理贴图”。这些是表示模型表面上每个点的材料属性值的 2D 图像，并且可以通过操纵数字照片，通过手工绘制，作为着色器中的计算计算结果或组合来创建。三个。″

"为艺术资产创建的地图集因管道而异。但是，常见的地图类型包括漫反射（模型的表面颜色），凹凸（用于模仿精细表面细节，如凸起和皱纹）和镜面反射（用于控制对象表面上高光的形状）。其他类型的地图由 3D 软件本身生成。这些包括法线贴图（一种特殊形式的凹凸贴图，通常用于为模型的低分辨率版本提供更高分辨率版本的外观）和环境遮挡贴图（用于表示周围几何块的范围）从撞击表面上的每个点反射光线）。″

"确定如何在 3D 模型的表面上显示 2D 纹理图的过程被称为“UV 映射”。这可能涉及简单地将纹理图“投影”到模型上，就好像它是一个几何对象，如平面，圆柱或球体;或者 - 对于复杂的有机物体更常见 - “展开”表面以将其分解成可以施加纹理的平坦“壳”。壳的这种布置被称为“UV 布局”。广泛用于现代制作工作的 3D 油漆包 Mari 进一步将布局细分为方形瓷砖：称为“UDIM”的系统。″

"UV 映射依赖于 3D 模型的形式和拓扑，而纹理又依赖于 UV 布局。这对于管道团队来说可能是一个令人头疼的问题，因为对模型的更改可能会使部分或全部 UV 映射无效，从而导致遵循这些 UV 布局的任何纹理无效。一些团队试图通过强制在模型经过仔细审查之前不能开始纹理来避免这个问题，尽管这通常使得难以迭代资产的外观。其他人创造的工具可以将纹理从一个版本的模型转移到另一个版本的不同的紫外线 - 这个过程很少产生完美的结果，但至少可以减轻重做工作的痛苦。还有一些人转向不需要 UV 分配的纹理映射系统，例如“Ptex”系统，其中为网格的每个面分配单独的纹理。然而，目前在诸如 UV 映射的广泛商业 3D 工具中不支持这一点。″

"通常用于纹理工作的工具包括 Photoshop，它与 UV 绘图工具结合使用;和 3D 绘画包，如 Mari 和 BodyPaint，使艺术家可以直接在模型表面上绘画，减少了处理 UV 贴图的需要。″

"虽然纹理贴图通常由非技术艺术家创建，但着色器通常由技术艺术家或程序员编写。这些是将高端数学技能与艺术能力相结合的专家精英阶层：他们需要能够编写复杂的模拟，但也需要在硬件限制要求时进行艺术启发的妥协。″

"对于管道开发人员，重要的是要注意实时着色器（用于在艺术软件本身的屏幕上显示资产）在实现方面与离线着色器（用于渲染最终图像）完全不同，即使底层图形算法是相同的。实时着色器受运行软件的硬件（特别是 GPU 或图形处理单元）的功能限制;离线着色器可以是任意复杂的。我们将在下一章中更详细地探讨这种区别。″

"在许多方面，着色器和艺术资产之间的关系是任何生产中最关键的依赖之一。对着色器代码的更改可以轻松更改渲染资源的外观，而对几何体或纹理贴图的更改也可能会更改着色器的输出。″

第 3.6 节 镜头布置

"构成镜头的各个资产必须组合成一个 3D 场景，准备渲染。这被称为“镜头布局”或“镜头组装”。在此过程中，布局艺术家将确保数字角色，道具和环境在虚拟空间中正确定位;并且数码相机的位置可以在渲染场景时正确显示动作。在视觉效果工作中，3D 场景可以与板组合或者与在组上捕获的参考图像组合以确保数字元素与现场镜头匹配。″

"布局使设施能够可视化和调整数字集，规划摄像机位置和移动，并消除这些摄像机永远不会看到的几何体，以便使场景更快地进行操作和渲染。它还确保了序列之间的连续性：例如，在早期镜头中从城堡墙上跳下的角色不会在城垛上看到，或者他骑马的马不会改变颜色射击″

"由于布局在开始拍摄所有最终资产之前开始，艺术家依赖于低分辨率占位符模型，这些模型将在稍后被高分辨率几何体替换，并且基本的“阻挡”动画代替最终版本。此工作流程创建必须由管道团队管理的每个资产的两个版本之间的依赖关系。″

"此外，布局确定几乎所有 3D 资源的文件名。能够从布局到合成一直谈论相同的文件避免了很多混乱。″

第 3.7 节 索具

""装备”是动画师工具箱的核心 - 至少是角色动画师的工具包。虽然可以通过在设施的艺术软件中直接操作简单对象（例如灯光和照相机）来制作动画，但通过单独移动网格上的每个点来动画更复杂的模型将非常耗时。相反，艺术家创造了一个更简单的底层结构，或“骨架”，以及一组操纵它的控件。当骨架移动时，它所绑定的模型部分随之移动，从而更容易将角色置于所需的姿势中。骨架及其控件一起构成了角色装备。″

"钻机不仅适用于有机模型：它们还可用于控制复杂的硬表面模型，如车辆或机械。但是，生物和角色的装备通常是设施创建的最复杂的。″

"视觉效果工作中使用的典型角色装备有三个组成部分。首先，有骨架本身。它由刚性元素组成，通过类比于真实解剖结构，通常称为“骨骼”。装配艺术家的工作是确定这些骨骼是如何连接在一起的。这里经常遇到的两个技术术语是正向运动学（FK）和反向运动学（IK）。在 FK 动画中，通过单独旋转每个骨骼来构成一系列骨骼，例如控制手臂的骨骼（在手臂的情况下，这可能是上臂，前臂和手）;在 IK 动画中，动画师操纵链的末端，其他骨骼自动跟随（在手臂的情况下,这意味着当动画师移动角色的手时,前臂和上臂随之被拖动）。装配工具还设置了“约束”：一种控制形式，可以直接操纵骨骼，无论它们与其他骨骼的关系如何。例如，旋转手臂通常会使手依次旋转，但是可以“约束”手，使其保持其原始方向，而不管手臂的其他部分是什么。约束使动画师能够在对特定作业有意义的参照系中操作 - 例如，即使在人体移动时保持角色的头部看着目标。″

"其次，有“木偶装备”。虽然通过移动骨骼来构建角色比移动模型表面上的每个点更快，但这仍然是一个繁琐且耗时的过程。出于这个原因，设施通常围绕角色创建一组外部控件，动画师可以更容易地操作。这些外部控制引导骨骼的运动，进而引导网状物表面的运动。这是一个复杂的计算链，因此设计木偶装置时的挑战是使其足够复杂，为动画师提供他们需要的所有控制，但又足够简单，以便他们可以继续实时工作。″

"第三，钻机的某些部分决定了骨架如何影响网格表面。装配工控制网格表面上的哪些点受到单个骨骼的影响，以及在何种程度上：称为“蒙皮”或设置“皮肤重量”的过程。然而，为了模拟更复杂的二次运动，例如肌肉膨胀和相互滑动的方式，钻机可能包含单独的肌肉设置，或者可能触发“混合形状” – 网格表面的复杂变形，通常通过编辑创建数字雕刻包内的模型。″

"除了为动画师提供控制之外，装备还可以生成其他部门使用的数据，例如 FX 或照明。因此，创建钻井平台的任务通常由专门的专家来处理。优秀的索具艺术家对解剖学结合编程和脚本技能有着深刻的理解：虽然可以在设施的主要 3D 软件中执行装配，但通常需要创建自定义工具。″

"索具对管道团队来说是一个特殊的挑战，因为它是一个依赖管理问题变得特别严重的领域。角色装备通常用于各种不同的动画中。如果在项目过程中简单地将骨架从现有文件复制到新文件中，则很可能不同的镜头将不同步。为了确保骨架保持一致，许多工作室使用“引用”-3D 软件中的常用功能，在两个文件之间创建实时链接，以便对一个文件的更改自动传播到第二个文件。但是，这意味着对主骨架文件的更改可能会在整个生产过程中产生影响，通常会产生无法预料的后果。″

"像骷髅一样，钻机控制设置可以通过复制或引用来分发 – 具有类似的缺陷。此外，通常通过使用自定义脚本扩充设施的主要 3D 软件中提供的工具来创建装备控制。就像编写自定义代码所创建的任何内容一样，这意味着它们通常是错误的，并且需要在生产过程中进行更新。管道团队以一种允许对钻井平台进行迭代改进而不会危及生产计划的方式来管理这一过程非常重要。″

"通过提供良好的元数据和依赖关系跟踪服务，管道还可以使装配工和动画师的生活更轻松。例如，建议依靠引用来分发装备的团队使用数据库跟踪引用流，以便装配工可以有效地检查他们所做的任何更改的结果 – 如有必要，重新处理已经失效的任何下游文件。如果动画师即将使用过时的装备处理文件，也可以通知它。″

第 3.8 节 动画

一旦模型被装配好，它就可以用于动画了。有许多不同的方式可以生成动画数据。我们将在下面讨论其中的一些，探索与生产管道相关的每个关键特性。

手动 K 动画

手动键控动画是由动画师手动创建的动画。这是通过设置“关键帧”（定义角色运动的姿势）来完成的，3D 软件可以在其间自动插值以产生运动的错觉

好处：

小型制作的设置成本往往较低

适用于具有奇怪或不真实比例的模型

适合不遵循物理定律的程式化运动和运动

允许使用传统动画中的技术，例如“壁球和拉伸”（保持角色总体积的变形但不保留其相对比例，用于卡通动画以夸大运动的视觉影响）

缺点：

高质量的结果需要一位才华横溢的动画师

与其他解决方案相比，高度逼真的结果往往需要更长时间才能创建

所需的工作量根据结果的长度和复杂程度而变化

手动键控动画通常在设施的主要 3D 艺术软件中创建，有时由自定义工具辅助。然而，对于需要大量动画数据的项目，特别是如果它们代表逼真的人体运动，设施通常会转向涉及记录现场演员（通常是人类，但有时是动物）的动作的技术，并将数据处理成可以被导入到 3D 软件中。这些包括全身动作捕捉，面部动作捕捉和表演捕捉。

动捕

术语“动作捕捉”倾向于用于指代记录演员全身动作的过程。使用了许多不同的方法。一些最常见的是光学动作捕捉（其中一组摄像机用于记录演员的动作，通常通过跟踪附着在其皮肤或衣服上的标记）和惯性或机械捕捉（其中传感器附着在演员的身体记录上）他们身体部位的相对运动）。

好处：

使用经验不足的动画师可以获得相对高质量的结果

可以在给定时间内生成更多动画数据

创建复杂或冗长的表演往往成本更低，允许演员执行许多不同的拍摄

更容易创建逼真的物理交互，例如一个角色将另一个角色推向地面

缺点：

对于较小的制作，初始设置成本可能过高

使用外部捕获工作室的项目的周转时间可能更长

如果需要新的动作捕捉拍摄，迭代可能会成本或时间过高

移动仅限于现场演员可以实际执行的移动

无法捕捉风格化的动作，或不遵循物理定律的动作

可能难以将捕获的数据“重新定位”到具有与演员不同的身体比例的模型

捕获的数据仍需要手动清理

必须稍后手动添加传统的动画技术，如壁球和拉伸

面部运动捕捉

面部动作捕捉特指记录演员面部动作的过程。这通常使用光学捕获技术来完成。与全身捕捉一样，这些可能需要将标记粘合到演员的皮肤上，或者可能依赖于“无标记”技术，例如使用图像处理来重建演员面部的表面几何形状。与手动键控动画相比，面部捕捉与全身动作捕捉具有许多相同的优点和缺点。然而，除了移动之外，一些面部捕捉技术能够记录其他有用的数据，例如演员脸部的详细几何形状或来自其皮肤的纹理信息。

性能捕获

出于讨论的目的，我们将使用术语“性能捕捉”来指代同时记录演员的面部和身体表现的动作捕捉解决方案，以及相机的位置和方向，以及其他参数，例如镜头校准。性能捕获的优点和缺点反映了全身和面部动作捕捉的组合优点和缺点，但是因为它同时捕获面部和身体，因此产生的性能趋于更加统一。

也可以自动创建面部动画而不记录现场演员的动作。这在特征动画和游戏工作中比在视觉效果中更常见，并且通常用于使数字角色能够与现场演员的录音同步。

自动唇音同步

有许多商业软件工具将录制的语音处理成其组成的“音素”：用于构成口语单词的声音单元。一旦艺术家雕刻了一系列变体模型，这些模型将角色的嘴部显示在与每个模型相对应的位置，软件每次在音频流中检测到新的音素时都会触发适当的面部形状，从而产生角色正在说话的错觉。

好处：

可能是大量唇形同步的最便宜和最有效的解决方案

适用于无法及早锁定脚本的制作

（仅限游戏）在运行时动态生成语音可减少动画文件所需的存储空间
（仅限游戏）音频文件可以更容易地重复使用，因为通过将大量较短的语音文件拼接在一起可以生成更长的对话框
缺点：
与面部动作捕捉和手动动画相比，质量低
（仅限游戏）需要在游戏引擎中进行更多设置

设施有时会区分“角色动画” - 角色本身的表现，通常是通过上述技术之一创造的 - 以及“技术动画”或“技术动画”，其中包括头发和衣服的运动。由于这些通常是通过物理模拟半自动生成的，因此一些设施将工作称为“字符 FX”。可以使用设施的主要 3D 包中内置的工具，Shave 和 Haircut（用于头发）或 SyFlex（用于布料）等专业软件或通过自定义代码创建模拟。模拟数据通常需要手工细化，以修复角色身体的交叉点，以适应角色的表现。

模拟还用于控制人群中人物的移动,或者通过使用 AI 系统将动画行为分配给各个“演员”，或者通过粒子模拟来驱动他们的动作。这通常是一个单独部门的责任。同样，这可以通过商业软件来实现 - 这些工具中最着名的是 Massive，它是基于 AI 的或专有的代码。我们将在下一节中更详细地研究模拟。

镜头动画
角色动画分为几个阶段。使用 previs 或布局数据，动画师首先“阻止”动作，设置角色所需的关键姿势行动。在这个阶段，重点是整体的时间安排，而不是细节。下一步是改进拦网传球。在这一点上,这个过程不再是线性的,因为包括 fx 和 lighting 在内的许多部门正在同时进行拍摄。分发向多个部门开枪有助于每个人在上下文中看到自己的工作，使缺陷变得明显早一点，有助于加快生产进度。许多设施都是按顺序处理动画的。而不是努力使最后一枪，他们努力让一个完整的序列通过管道。它是动画师通常会阻止一个镜头，获得批准，然后转到下一个镜头，而不是改进拦网传球。这使导演能够一次查看整个序列，编辑根据需要退出或添加快照。一旦关键帧动画完成，可以使用默认的肌肉设置来添加第二个移动到角色，然后是处理几何图形的技术动画过程相交，并添加布料和头发模拟。动画师将继续优化结果，直到镜头被批准或标记为如果时间允许的话，cbb（可能会更好）可以在以后进行调整。设施通常会指定在阻塞之后动画可以发生的最大迭代次数，称为“动画预算”。它也是标准的，以同意客户的各个阶段的完善动画将通过一些公司使用的术语“封锁”，“初级”和“次要的”,而其他人则使用“主要的”和“最终的”。定义什么也很重要向客户端呈现动画的视觉质量级别：例如，使用“播放预览”（直接从 3D 软件中录制的动画，通常没有最终纹理和照明）；一级标准照明；或准备最终批准，包括最终头发 75 个 www.do-vfx.com 网站以及布料模拟。在电影作品中，动画过程的输出通常由一个网格缓存--一个数据文件组成显示模型曲面上每个点的位置，而不是在其中动画仍然可以直接编辑。因为钻机通常由模拟部件组成，如肌肉设置时，必须先计算它们对网格曲面的影响，然后才能将数据传递到照明和渲染。

第 3.9 节效果和模拟

效果（通常缩写为“FX”）是一个通用术语，用于描述通过数学过程计算的镜头中的动态元素，而不是手动动画。常见的例子包括火，烟，水，头发，布料，破碎和破坏效果 - 以及幻想电影中用于表示魔法的效果。一些设施使用术语“CFX”（角色效果）来区分与角色相关的 FX，如头发和布料模拟，以及其他类型的 FX。

传统动画需要艺术家输入或动作捕捉数据来驱动角色装备或英雄道具，而 FX 采用数学程序来生成模拟（通常简称为“模拟”）。与科学研究或工程中使用的物理精确模拟不同，sims 通常会进行显着近似以降低计算复杂性，例如使用简化代理几何，LOD（细节层次）技术，复杂多体相互作用的单向耦合，或实例化预烘烤模拟。准确模拟浇注到玻璃杯中的水可能需要数周才能计算出来，而 FX 艺术家可以在几分钟内达到相同的近似效果。

sims 的另一个特点是它们通常基于运动物理方程的重新构造，以提供某种程度的艺术控制。相比之下，精确的物理模拟是单片“交钥匙”解算器，一旦模拟开始就不允许进行艺术干预。

外汇艺术家最喜欢的技术是“程序主义”。一个例子是用于模拟火灾，烟雾或灰尘的粒子模拟。控制模拟力，约束，单个粒子寿命等的输入由平衡物理精度和艺术控制的程序或规则决定。

通常，程序主义比物理上精确的模拟提供更多的可定向性，并且计算密集度更低。然而，这两种方法都有其优点和缺点，这导致许多外汇艺术家采用混合方法。一个例子可以是折叠建筑物的刚体动力学模拟，其中碎片准确地响应重力和其他力，但是它们对碰撞的响应方式由非物理精确的程序定义，以确保整体效果具有导演想要的时间。

FX 工具为管道开发商带来了重大挑战。有些人会产生大量数据;其他人生成的数据需要具有挑战性的转换才能被其他下游工具使用。前者的示例包括流体或液体 SIM，其倾向于产生具有显着存储器和存储足迹的大数据集（密度，速度，温度等）。后者的示例包括通常用于流体和刚体模拟的隐式等面，其需要在呈现之前转换为诸如多边形模型的显式表示。此外，FX 工作的逐个镜头特性通常会导致具有独特要求的一次性工具，在微管道中实施。

为了更好地说明所有这些问题，让我们关注外汇工作的一个方面：体积数据。但是，请记住，此讨论的大部分内容适用于现代外汇管道中遇到的许多其他类型的数据。

体积数据对于电影制作中的许多“面包和黄油”效果至关重要，包括云，灰尘，水，火和气态流体。通常，体积数据以空间均匀的网格编码，其值表示整数坐标位置处的索引空间中的离散样本。索引空间中的这些离散样本点通常被称为“体素”。

这些数据对于管道开发人员来说具有挑战性，因为它占地面积大，并且因为它经常需要转换为其他格式才能被其他工具使用。让我们分别讨论这两个问题，并使用 OpenVDB 数据结构来说明具体的解决方案。

传统的 FX 工具既产生又消耗由索引空间中的密集“盒子”体素表示的体积数据。这是一个简单的数据表示，几乎不需要了解底层数据结构，并且由于简单性在大多数管道中都是一种优点，因此它已经流行多年。

然而，随着以更高分辨率产生体积效应的需求增长，这种简单的方法越来越多地引起 FX 管道的问题，主要是由于这些密集网格的巨大足迹。解决方案是采用更紧凑的稀疏体积数据结构，该结构使用先进技术仅在需要它们的地方分配体素：例如，液体的速度非零。

一个例子是 OpenVDB 卷数据结构，它为内存紧凑卷设置了新的行业标准。概念上，OpenVDB 使用新颖的分层数据结构建模无限大的 3D 索引空间，该结构与现代 CPU，文件系统和关系数据库共享多个特征。 OpenVDB 可以减少数量级的内存消耗，并用于许多工作室管道和商业软件包，如 Houdini，RenderMan 和 Arnold。

困扰许多外汇管道的另一个问题是，数据通常必须在管道中的工具之间传递之前进行转换。同样，体积数据就是一个很好的例子。转换可能是必要的，因为不同的工具使用不同的卷表示（例如，密集与稀疏数据结构）或更常见，因为许多工具根本无法操作（或“消耗”）体积数据。

为了更好地理解原因，重要的是要记住几何（即实体表面）实际上可以显式和隐式地表示。显式曲面表示是 FX 流水线中最常见的，通常采用多边形或参数曲面的形式。然而，隐式表面表示 - 其中表面被定义为 3D 体积的等表面 - 越来越受欢迎，因为它们允许复杂的表面变形，紧凑并且易于与许多模拟工具集成。因此，要将生成级别集的流体模拟器链接到仅支持三角形的渲染器，需要进行转换或“多边形化”步骤。根据所涉及的效果的性质，这种转换可能是微不足道

的，具有挑战性的，甚至是不可能的，这可能对哪些工具可以在 FX 管道中组合产生根本影响。

第 3.10 节 照明

照明通常是拍摄在一起的地方。除了需要简单地照亮动作之外，照明还定义了镜头的情绪，设置了一天中的时间和天气状况，确定了调色板，并有助于引导观众围绕场景。

如果要将镜头的 CG 元素与实时动作集成，则数字照明需要使用诸如 HDRI 图像的现场参考来尽可能地匹配板的 CG 元素。如果镜头完全是 CG，那么目标是匹配预览和概念艺术。通常，纯 CG 的卡通风格鼓励在现实世界中很少发现的令人发指的光照条件。

在生产中，为每个关键位置创建单独灯光或“照明装备”的集合。通常对关键镜头进行处理，一旦获得批准，在进行镜头特定调整之前，将照明设置复制到其他镜头。

照明工作传统上是在设施的主要 3D 艺术包内进行的。然而，越来越多的专业工具使艺术家能够更快地迭代照明(或“重新点亮”一个镜头)。其中最着名的是Katana,最初由Sony Pictures Imageworks 开发，现在作为商业产品提供。

第 3.11 节 渲染

渲染是在 3D 场景中获取数据并将其转换为从数码相机的 POV（视点）看到的一系列 2D 图像的过程。在电影工作中，这些计算是离线完成的，通常需要花费一个小时到几天来处理单个帧。

在电影中，通常以一系列图层或“通道”渲染场景，每个图层仅显示场景中资产的子集（例如，仅显示背景，或仅显示前景字符）;照明数据的子集（例如，反射，阴影或环境遮挡）;或者在合成镜头时有用的其他信息，例如，Z 深度通过（表示场景中每个点距相机的距离的灰度图像）或其他 AOV（任意输出变量）。在通道中渲染场景可以更轻松地将 CG 元素与实景镜头组合在一起。通过组合合成包中的传递而不是完全重新渲染它来迭代镜头的外观也更快。

虽然一些大型设施使用自己的内部软件，特别是对于渲染 FX 等特定任务，商业渲染器也被广泛使用。多年来，电影市场一直由 RenderMan 和基于其“雷耶斯”架构的其他工具主导，旨在在现实主义和渲染速度之间取得平衡。然而，更多地使用“射线追踪”技术的渲染器 - 传统上比雷耶斯算法慢，但能够产生更逼真的输出 - 正变得越来越流行。这些包括 Arnold，V-Ray 和 mental ray。

第 3.12 节 合成

合成是将构成镜头的所有渲染和实时动作元素混合成一系列完成图像的过程。除了渲染元素和准备好的印版外，合成商还依靠相机跟踪数据将两者无缝地整合在一起; LUT 数据（查找表，用于表示将显示图像的介质的颜色空间），以确保合成的镜头具有正确的色彩平衡;和镜头失真，噪音和纹理信息，以匹配原始素材的外观。

一般来说，在合成期间比在 3D 中更容易调整镜头的外观。例如，假设您希望整个图像的颜色略微变红。在合成包中执行此操作要容易得多，在合成包中，它可能只涉及移动单个滑块，而

不是调整 3D 软件中一百个单独灯光的色调。同样，合成器工具包的一个关键部分是一个包含“实用元素”片段的图书馆 - 可以将火焰，烟雾，灰尘和水溅等物体的动作片段整合到图像中，而不是要求 FX 部门进行额外的模拟。

但是，要谨慎，因为“他们可以在 comp 中修复它”，因此在管道中的早期问题尚未得到解决。就像“他们可以在帖子中修复它”这样的短语有时被用作在拍摄中放置设备的邋 work 工作的借口，这需要稍后绘画 - 例如，合成器的作用是对图像进行艺术调整，不解决别人的问题。管道应该最小化需要在 comp 中修复的问题的数量，而不是添加到它们。

合成所需的渲染过程的数量和性质取决于镜头的主题和所需的视觉风格。优化传递集是管道团队的一项关键任务：您不希望仅生成传递以发现它实际上不是必需的，或者仅以高分辨率渲染角色才发现它只是可见的在一枪的背景。

虽然 After Effects 等 2D 合成软件包广泛用于广播工作，但视觉效果行业主要受到 Nuke 和 Fusion 等 3D 解决方案的支配。这些可以创建场景的内部三维表示，从而可以对最终图像进行更复杂的调整。

这就是我们对为电影工作创造资产的过程的探索。在下一章中，我们将了解资产创建过程如何因游戏而异。

3A 激光雷达：现场资产捕获

VFX，游戏或玩具的编目集是我们在 Gentle Giant 的计算机图形工作室的很大一部分。自从九十年代中期开始为星球大战，哈利波特，指环王和星际迷航等大型项目完成这项工作，我们必须开发和利用数字资产管理技术，用于收集期间收集的大量参考数据。电影的制作。我不能过分强调这对娱乐业有多重要。乍一看，你可能会认为我们已经扫描了千禧猎鹰仅仅是一个复杂的视觉特效镜头，但这些东西并不是短暂的图像 - 它们最终成为玩具，游戏，并且有足够的动力，是流行文化的珍贵部分。您在电影集上收集的信息可能会在几十年后被评估和使用，这也是像 Lucasfilm 这样的公司将 Gentle Giant 视为用于存储这些数据的记忆 Alpha 的原因。因此，我们非常重视 DAM。对我们来说，DAM 是在我们第一次得知新项目的时候开始的，任何任务的第一步都是组建一个客队。

还记得 20 世纪 70 年代的超高频电视节目（由于某些原因，演出空间：1999 年），不可能的任务吗？在每次任务之前，我们的英雄将获得一个自毁的档案，概述目标，所涉及的主要参与者，任务的位置，期望和目标。继续设置非常类似，最后没有爆炸。（“不可能的使命基金会”中的那些人将这些档案放在一起从未得到我的充分感谢和赞赏。）制定一个全面的计划是一个关键因素，它将收集数据，同时与汤姆克鲁斯交往一个愉快的经历，而不是等待毫无准备的恐慌。

确定关键参与者，并确保为您的团队准备一份关于相关人员的关系和显着成就的全面概述。您可能与 VFX 协调员或制作人有直接关系，但由于您可能会从 VFX 主管获得现场指导，因此有助于了解他或她所参与的项目以及他们所做的任何值得注意的流程开发首创。一个健康的 Cinefex 图书馆，美国电影摄影师，游戏开发者和 3D 杂志是一个有用的参考，以及针对他们参与的电影或游戏的网络搜索。

该制作的 VFX 协调员可能已经在 Supe 的最后一部电影中担任数据争夺者或助理，并且可能无法理解你所提供的所有内容，所以永远不要假设他们确切知道下游使用你的数据的 VFX 供应商会想要什么或需要。例如，在激光雷达的情况下，我们可能会被委托扫描原始汽车的碰撞现场，因此我们建议在实际效果和特技工作“改变几何形状”之后扫描车辆是明智的。机动车。这是因为当一位导演在 VFX 供应商的工厂审查正在进行的镜头时，人们永远不知道下游几个月需要什

么，并且可能决定“加”一枪。捕获并可能更容易丢弃最终未被利用的数据而不是限制场景的艺术可能性因为该组被击中并且特技车辆被丢弃。

可能需要扫描电影的演员以便在渲染场景中使用。诊断您的收件人供应商的管道，以针对其典型工作流程定制资产，这将因公司而异。一个供应商可能正在通过“运动研究范围”寻找演员的壁球和伸展肌肉运动极端，而另一个可能需要特定的自然表达，例如“愤怒”或“恐惧”。一旦演员到达并参与其中，他们可能对他们的演奏方式有更具体的了解，或者可能会在拍摄过程中扮演一个可能涉及从现场转换为数字双重的场景。虽然告诉奥斯卡获奖演员如何看起来很开心可能会引发相反的表达，但是可以有条理地放置源图像来引用 Paul Ekman，George Brant Bridgman，Gary Faigen 等专家的作品。更好的是，提前参与 VFX Supe 并询问他们对源参考资料的意见。他们可能会回应“Ekman，Bridgman，Faigen 等”。但至少在他们看来你正在倾听。当然，这些人都在忙着拍电影，所以不要纠缠他们，也不要暗示获得这些信息是做这项工作的先决条件。未来的工作将倾向于流向能够完成工作且最容易合作的供应商，因此计划成为该供应商。

我们甚至委托 Neville Page 3D 打印出一个演员冻结在示例表达式中。这些项目，或者像 Andrew Cawrse 的解剖工具之类的东西，有助于提供参考，并为您的视觉特效团队提供符号保证，让您欣赏正在创作的艺术和魔法。

完成设置 LIDAR 扫描的可用时间会影响可捕获的信息量。如果在船员午休时间内对一组进行控制，则必须设置扫描仪分辨率以最大化覆盖范围并仍然捕获在机组返回之前所需的内容。（它们总是比预期的更早回来。）理想情况下，在工作人员为一天包裹之后，一套将被移交给扫描和纹理摄影，这对你来说意味着一个全能的。然而，由于繁忙的生产计划对 VFX 需要完成的工作进行了有限的考虑，通常有一个工作人员站在旁边，看着你的肩膀，意图打击设置并开始构建一个新设置的时刻你宣布你已经完成了。让这些人知情并参与数据收集过程可能意味着他们提供帮助而不是阻碍您的进步。

完成设置 LIDAR 扫描的可用时间会影响可捕获的信息量。如果在船员午休时间内对一组进行控制，则必须设置扫描仪分辨率以最大化覆盖范围并仍然捕获在机组返回之前所需的内容。（它们总是比预期的更早回来。）理想情况下，在工作人员为一天包裹之后，一套将被移交给扫描和纹理摄影，这对你来说意味着一个全能的。然而，由于繁忙的生产计划对 VFX 需要完成的工作进行了有限的考虑，通常有一个工作人员站在旁边，看着你的肩膀，意图打击设置并开始构建一个新设置的时刻你宣布你已经完成了。让这些人知情并参与数据收集过程可能意味着他们提供帮助而不是阻碍您的进步。

由于一个视点会从各个角度排除视觉，因此必须将摄像机和扫描仪移动到多个位置，以收集整个设备的覆盖范围以及设备所需的所有较大项目。 （较小的道具将在稍后介绍。）熟练的 3D 建模将允许您判断对象以确定需要多少覆盖。例如，圆柱形或对称的硬表面固定件可能更容易通过较少的扫描数据再现，因为通过镜像或旋转重建它所需的适当横截面更容易创建。资产管理的一部分涉及避免创造超出需要的工作量，但这是一个很好的平衡，因为如果你没有获得足够的覆盖率，那么几周之后就不可能回去捕获。事实上，因为这种认识甚至可能发生在另一家 VFX 供应商的下游，而你没有在那里捍卫你的工作，它可能只是简单地归结为你的无能。这可能会对您的声誉产生负面影响，即使您因电影冲突的现场条件而无法进行彻底的工作。 （也许可以开发一种元数据方案，其中包括诸如“当我们从金属剪叉式升降机上方 80 英尺处扫描时从北方接近的雷暴”时的注释。）

始终优化您的时间。借用或带上您自己的收音机，这样您就可以通过聆听任何好莱坞制作的生命线的对讲机通信，了解可用的部分或不可用的部分。始终尝试最大化机会并最大限度地减少可能影响数据资产质量的中断。如果你收到很快就会知道一套的消息，那么即使在你被 VFX 协调员告知要移动到那里之前，也要靠近它，而不是从半英里外的午餐帐篷里赶来。

由于缠绕扫描仪所需的时间，使用系留笔记本电脑虽然理想，但可能不可能，因为必须优

先考虑在分配的时间内捕获最大量的信息。在资产管理方面，相机或扫描仪现在包含重要的设置细节;因为扫描仪通常提供模糊的文件名，一旦捕获了一组，必须立即优先考虑扫描的安全备份以及正确识别数据。

相机卡或扫描系统具有有限的内部存储限制，并且每次收集通过通常导致几千兆字节的数据，因此扫描仪可能需要“擦除”，通常是尽快的，以便可以开始扫描另一组。第一步是将该数据下载到便携式现场存储。但是，认为现在“备份”可能是一个悲剧性的错误，因为如果现在擦除扫描仪的存储，则再次处于只有一个副本且备份尚不存在的情况。数据通常被复制到两个独立的现场驱动器，理想情况下使用固态技术，因为旋转硬盘更容易出现故障。第三个副本保留在用于卸载数据的计算机上，直到它被安全地返回办公室进行处理。如果这套装置已经被破坏，或者因为其他工作人员正在使用它而根本无法再进入，那么这将是你抓住它的唯一机会 - 并且一个肯定的方式是不被邀请回到下一部电影将是打破向 VFX 主管告知您丢失了数据。

因为第一次哈利波特在 9/11 恐怖袭击事件发生后不久就开始投入生产，有关高管要求知道我们如何防止他们的数据在发生另一次攻击时丢失，因为它正被飞回工作室。虽然我们最初的反应是我们已经死了并且不会真正关心，但我们能够如实地向他们保证，副本将在两个单独的航班上从伦敦运到洛杉矶。这可能是一个极端的例子，尽管人们应该努力建立一个可靠归档数据的声誉，这些数据可能在几个月后的后期制作中再次被访问。

纹理和纪录片摄影与扫描是一个单独的步骤，因为虽然一些扫描仪提供了捕获颜色信息的方法，但它对于真实的 VFX 渲染目的来说从未像现在这样有效。纹理摄影也发生在分配用于扫描的同一时隙中，这增加了协调工作的难度，因为扫描中的任何人或设备而不是该组的一部分变成了在后处理中必须被移除的无关数据。 。这可能听起来微不足道，但请记住，扫描不是即时的，因此机组人员不仅可以在移动时破坏大面积的扫描数据，同一个人在走路时也可能出现在数据的几个不同位置关于3D扫描进展。有时提醒生产的VFX协调员有用的是从模型中消除这种“噪音”会增加下游处理工作的天数。

确定摄影位置的好方法来自使用 3D 软件将纹理映射到模型上的经验。通过了解 VFX 艺术家几个月后将需要什么，您将能够很好地判断设定的需求。 HDR 包围是优选的，以允许在处理下游图像时具有最大的灵活性。如果需要使用与正在进行的阴影条件相矛盾的虚拟光照渲染对象，则 HDRi 将允许纹理艺术家处理阴影，从而显示重新渲染具有不同光照角度的对象所需的基础颜色。

纹理摄影旨在复制拍摄过程中出现的设置片段，因此在摄影期间请求拍摄期间使用的相同灯光是良好的做法。这可能是昂贵的，因为如果在主单元包裹一天之后进行扫描，则必须要求工作人员加班。 Nodal 全景拍摄距离使用激光雷达扫描仪的位置非常接近，因此通常会出现“交易场所”的情况，因为扫描仪在同一个地方使用 360 全景拍摄，反之亦然。此外，还采用“直接”拍摄，最大限度地减少失真并最大化物体到图像的捕捉。有时这必须从高处发生，以便俯视一套，如果你不能说服 VFX 主管你有多年操作装配设备的经验，可能会为电梯操作员增加额外的费用。用你自己的安全带来抵达这一点并保持安全是令人放心的;即使在一个声场内，电影也可以是五层高，还有一块你不想降落的硬质水泥地板。在这方面，永远不要假设电影中的任何东西是真实的或安全的。虽然生产将尽其所能为您提供安全的工作环境，但您仍然需要意识到您所倾倒的建筑物壁架实际上可能是聚苯乙烯泡沫塑料上的薄石膏贴面，或者是您倾斜的钢柱可能真的是纸板的 sonotube。我记得在一部电影中，当我注意到从天花板延伸到地板的绳索略微摇晃时，我的“蜘蛛侠”感觉刺痛。我好奇地盯着，但觉得没有理由担心。几秒钟后，一些绳子像蛇一样跳起来，然后掉到了地上。令人难以置信的砰砰声和空气爆炸之后突然意识到，七层长的绳子可能重达一百磅。最好忘记你的任何期望，并且在你采取的每一个行动之前，考虑“它是否安全？”

因此，您已经结束了工作日，需要管理三组重复的 3D 数据，以及“每视点括号内拍摄的 36 万像素 11 张照片”，在离开套装之前还必须经过三重备份过程。由于人类记忆可以通过大量数

据快速失败（并且在 Keg 强制性工作人员聚会之后），我们使用 Apple Aperture 快速组织和标记图像以及 3D 扫描数据到一个关键字索引文件夹和项目子集。令人信服的 Aperture 可以摄取和存档 3D 数据，只需将扩展名“.mov”添加到文件中，无论真正的内部格式如何。忽略 Aperture 抗议它无法预览文件，3D 扫描软件的额外屏幕截图整齐地覆盖了这个“现场存档”的需求。稍后需要数据时，只需导出文件并删除“.mov”添加内容，以便准备好在本垒处处理。这也允许集合上的任何注释与数据直接关联，并且可以执行额外的 Aperture 功能，例如基于云的备份，假设位置提供连接。另外，在任何电影机上都有十几个无线热点，可靠性各不相同。虽然可以使用名为“Costume Gang”的热点检查笔记本电脑上的电子邮件，但尝试使用此类连接传输大型数据集是不太可能的，也可能是粗鲁的。最好向 VFX 协调员询问登录工作室官方连接所需的正确协议和密码，或使用 VFX 预告片或主要制作办公室中可能提供的硬连线连接。冒险未来的读者会嘲笑这个概念：基于蜂窝的热点从未提供足以传输千兆字节数据的速度或数据限制。理想情况下，您提供的有关互联网访问，电话号码，电子邮件地址等所需的所有信息将在您到达之前以爆炸性档案的形式提供。

4 游戏资产创建

第 4.1 节您将从本章中学到什么

在上一章中，我们研究了为电影创建资产所涉及的步骤。其中一些是视觉效果所特有的：显然，游戏没有直接的等同于诸如平板准备或 rotoscoping 等任务。其他如建模，纹理工作，动画，FX 和渲染与游戏开发共享共性。但是，也有重要的差异。

在本章中，我们将简要介绍一些更重要的差异，并探讨它们对管道开发人员的影响。同样，我们将按照大致时间顺序执行资产创建所涉及的步骤。 （有关每个步骤的定义及其含义，请参阅上一章。）首先，我们将研究为电影和游戏创建资产之间最根本的区别之一：数据导出的格式。

第 4.2 节数据导入和导出

与电影作品不同，在游戏中，资产不能总是以用于创建它们的软件通常会保存它们的格式使用。准备使用它是一个两阶段的过程，其中数据被“导出”出艺术包并“导入”到游戏中。导出会删除对处理资产但对游戏不感兴趣的艺术家有用的数据：例如，模型的构建历史记录或用于保持文件可管理性的可见性组。导出过程通常会生成一个称为“中间”或“桥接”文件的单独文件。导入将导出的数据转换为围绕游戏引擎需求设计的优化格式。

一些游戏开发人员将这两个操作组合成一个单独的软件，这避免了管理大量中间文件的需要。但是，大多数人更喜欢两步法。保持导出与导入分离使工程团队能够在需要时重新格式化资产（例如，当引擎更改需要更改文件格式时），而无需艺术家手动重新导出它们。虽然导出单个文件相当轻松，但是导出数千或数万个资产是一个巨大的时间间隔，并削弱了分配给该任务的艺术家的士气。两步过程意味着资产只需从创建它们的软件手动导出一次。然后可以编写工具来批量转换生成的中间文件，从而大大提高了进程的速度。

拆分进出口还可以重新格式化资产，以便在多个不同平台上使用，而无需艺术人员的过度努力。这些更改可以像使文件名区分大小写一样简单，也可以像更改在内存中填充值的方式或反转位顺序（在内存字节中写入位的方向）一样复杂。要求艺术人员为几个输出平台中的每一个选择正确的选项组合，并为成千上万的资产中的每一个选择正确的问题。工程师以艺术家看不到的

方式处理转换过程要容易得多。

资产转换管道的两个关键组成部分是出口商和进口商本身。我们将依次查看每一个：

导出

出口商通常 - 而且应该相当简单，因为它的工作主要是过滤掉不需要的数据。出口商通常是插件，可在 Maya，Softimage 或 3 ds Max 等 DCC（数字内容创建）工具中使用。它们可以使用主机包的 C ++ API 编写，也可以通过脚本语言实现。随着计算机变得越来越快，后一种选择变得越来越流行：虽然 Python 和嵌入式脚本语言（如 MEL 或 MAXScript）处理数据的速度比 C ++ 慢得多，但随着项目的发展，它们也更容易维护和更新。一个好的出口商几乎是看不见的，不应该给艺术家带来许多设置和选择。在许多情况下，它甚至不应该向用户提供文件对话框 - 直接从工作文件的名称或数据库中导出导出文件的名称效率要高得多，因此它们之间的依赖关系 Maya 或 3 ds Max 场景文件和最终的游戏内资产是明确的。

导入

进口商通常是比出口商更复杂的软件，因为它与游戏引擎同步发展，并且经常受到游戏及其目标平台的严格要求。鉴于出口商通常在艺术家的 DCC 软件内部运行，进口商应尽可能负责“繁重”工作。为场景生成碰撞几何体或照明数据可能需要数小时，因此最好将此工作卸载到构建场，而不是将艺术家的工作站占用一段时间。

虽然出口商通常由技术艺术人员设计，但进口商通常是硬核工程师的省。这种分工的一个不幸的副作用是进口过程充满了误解。许多艺术家都对每次处理资产时产生的神秘警告感到困惑。如果该工具提供了太多信息，他们倾向于将其调出（对于来自进口商的反馈而言，普通艺术家俚语是“呕吐”）。另一方面，如果进口商在艺术家做错事情时没有提供指导 - 例如，如果它静静地停止处理具有超过一定数量多边形的几何图形，而不是提供明确警告，例如“此资产太多多边形！“ - 确实会产生混淆。因此，技术艺术团队必须充当管理进口商的工程师和使用出口商的艺术家之间的桥梁。技术艺术家需要努力游说，以确保进口商生成的警告写得清楚，并包含可操作的信息。对比这个经典的神秘错误消息：

两者都代表了艺术文件中完全相同的问题 - 但其中只有一个可能会立即修复。理想情况下，错误消息应包含 wiki 或 http：//文档链接，以更详细地解释问题。还应在日志文件或数据库中跟踪它们：这有助于工程师识别容易出错且应重新设计的过程部分。

第 4.3 节详细程度

对于在特写镜头中观察时能够经受严格审查的模型，需要详细说明。角色模型可能需要精细的细节，例如单独建模的睫毛或牙齿，而环境资产可能需要单独建模的螺母和螺栓。这使得模型“沉重”：文件大小很大，创建和操作也很慢。

在电影工作中，这种模型的多边形数量可以延伸到数千万。在游戏中，传统上模型的分辨率较低，通过巧妙的纹理和阴影效果添加细节。但随着游戏硬件功能的增加，多边形数量正在攀升。

虽然这两部电影和游戏都试图通过使用 LOD（细节层次）技术来减轻重数据的一些缺点，但这些技术在游戏中更为重要。简单来说，当物体进一步远离相机时，LOD 系统将低分辨率版本（或一系列版本）的高分辨率模型换出。这样可以提高性能并节省内存，但它也会使资产创建过程变得复杂，因为 LOD 模型需要与原始模型绑定，以便对一个模型进行的任何更改都可以传播到其他模型。这是依赖管理的一个典型示例：一个可靠的生产管道必须为艺术家提供他们需要的工具，以确定何时应将更改传播到 LOD 模型。

LOD 不仅用于网格：当从远处查看该角色时，角色装备可能包含较少的骨骼，或者角色可能使用较少的低保真动画。也可能需要修改纹理和着色器以创建不同级别的细节。 Mipmap（包

含逐渐减小尺寸的相同纹理贴图的多个版本的单个文件）提供了一种创建纹理细节级别的方法，并有助于最小化某些纹理在缩放时看到的视觉伪像。

由于 LOD 创建是一项吃力不讨好的任务，因此它通常是自动化的候选者。自动生成 LOD 的技术已经存在了一段时间，有些技术非常复杂。但是，没有一种通用技术能够同样处理所有类型的内容。

例如，逐行网格缩小通过一次删除一个顶点来降低网格的密度，使用算法选择产生视觉上最明显变化的顶点。该技术在具有复杂有机形式的密集模型上非常有效，特别是高分辨率的特征模型。但是，它很少在机械或建筑模型上做得很好。另一方面，环境模型通常可以简化为“广告牌”或“卡片” - 简单的平面承载它们所代表的对象的 2D 图像。这是在森林中显示远处树木的常用技术，但不适用于角色或动画对象。大型现代化生产通常会有几个不同的并行工具链，用于创建不同类型的 LOD 内容，需要自动化和手工劳动的混合。

尽管自动化 LOD 系统通常以高分辨率资产开始并逐渐降低其复杂性，但相关技术却反过来做同样的事情。 曲面细分利用现代 GPU 的功能来细分网格，并且当伪弯曲多边形表面处于轮廓时，特别适合于平滑难看的角度。 这类似于从较低的 LOD 网格生成非常高的 LOD 网格，但仅限于重要的位置。

对于像骷髅剧院这样的小型独立视频游戏公司来说，最大的挑战之一就是创造高质量的数字艺术。三维艺术通常是非常劳动密集型的生产，并可能需要一套独特和多样的技能。较大的开发人员依赖于适应不同团队的预算、昂贵的工具套件和长的开发周期。独立工作室通常没有这些资源，因此必须要么简化他们的项目目标，要么找到一个创造性的解决方案，一个昂贵的艺术生产线的问题。

一种称为“摄影测量”的技术可以找到一种解决方案，该技术使用一组对物理物体拍摄的二维摄影图像来生成物体的三维表示：

通常是有纹理的三维网格。摄影测量工具使艺术家能够绕过数字网格和纹理的创作过程，回到古典技巧上来，在现实主义是自然副产品而不是有意识的努力的介质中工作。

摄影测量并不是一个新概念，在电影制作中已经使用了相当长的一段时间。

然而，最近，这些工具变得更容易为公众所使用。作为一家小公司，骷髅剧院选择利用这些强大的工具，生产几乎所有的艺术资产使用在我们的三维世界。

图 4.3 独立游戏开发商骷髅剧院使用基于摄影测量的管道来更快地创建资产。上图：准备拍摄的实物道具。中间：同样的道具重建从这些照片作为纹理三维几何。下图：游戏中使用的资产。

使用摄影测量进行艺术资产创作的主要优势是，它可以产生令人信服的结果，而无需所有的数字创作大惊小怪。但是，它确实需要一个独特的生产管道，可以处理过程的物理和数字元素。我们在骷髅剧院创建的管道在概念上可以分为三个阶段：艺术资产捕获、数字资产后处理和数字资产映射。在每个阶段，我们的目标始终是保持原始物理对象的质量，同时将有限员工的管理费用降到最低。

艺术品资产捕捉

进入我们艺术生产线的入口是照相馆。摄影测量“三维捕捉”工作室至少应包括一个中性照明空间，中心安装有目标物体，并有足够的空间让相机从四面八方拍摄，因为摄影测量只能重建照片中所示物体的那些部分。我们有建立了一个可调的自定义装备，以容纳相机，以便它可以围绕一个中心表，滚动轮上。这加快了摄影的速度，否则可能是一个繁琐的过程，并给我们一致的结果。

摄影棚必须解决两个对立的问题。3d 捕捉过程得益于尽可能多的视觉提示，以帮助软件确

定每张照片的空间位置。在这里，多色背景和强烈的灯光对比都有助于创造更好的结果。然而，这些照片也用于产生“反照率纹理”（指定表面颜色和反射率的纹理贴图），为此，应避免强烈的灯光对比和环境干扰，烘焙到反照率纹理中的照明信息可能与游戏引擎动态生成的信息冲突。

在我们的摄影棚中，我们选择保持灯光尽可能漫反射，目标对象周围的覆盖范围均匀。我们用白色的背景拍摄，但是加入了很多高频噪声。明亮的彩色标记和涂鸦横跨背景表面和周围的目标对象的基础都有助于创建一个参考框架之间的照片，同时保持照明中性。

数字资产后处理

在拍摄了一组照片后，它们被交给摄影测量工具，该工具为我们提供了一个有纹理的网格。结果看起来不错，但还没有准备好投入生产。就我们的目的而言，我们发现生成的网格对于视频游戏资源来说太重了，并且缺少在远处有效渲染对象所需的 lod。纹理也可能很零碎，利用不好紫外线空间，不适合 mipmapping。不幸的是，手动解决这些问题可能相当费力。为了尽量减少管理费用，我们选择自动化这部分管道。lod 是使用一种称为“二次边折叠”的网格简化技术自动生成的。通过构造网格的高分辨率插值版本并将颜色数据从纹理传输到网格来优化纹理。然后展开网格以创建新的 UV 映射，该映射用于从顶点数据重建颜色纹理。此外，高分辨率网格可用于生成高质量运行时动态照明的法线和高度贴图纹理。在理想条件下，这个过程是完全自动化的。然而，生产环境远不理想。网格 lod 可能无法生成，而 uv 自动展开有时会生成不合格的结果。为了处理这个问题，我们的管道使我们能够在每个阶段停止流程，并在需要更好的结果时切换到手动实现。

数字资产测绘

处理完艺术资源后，它们将与父对象关联并导入到引擎中。现在，它们已经可以在生产环境中使用了。然而，作为一家小公司，我们仍然面临着规模问题。如果我们不得不花时间手动创建游戏世界中的每个对象，我们有限的艺术团队将无法建立一个引人注目的环境。为了解决这个问题，我们将我们的游戏资产进行组件化。创建复杂对象（如建筑物）时，我们不会将整个对象创建为单个资源。相反，我们创建它的一部分。

在我们的世界编辑器环境中，门、窗、扩展、屋顶、基础和基础件都可以独立建造和组装。这有很多优点。它允许对象重用和高度的变化，这是任何小团队的关键。它还允许艺术家在创建每个组件时以最适合他们的比例工作。最后，由于现代摄影测量工具的局限性，它允许组装比数字化复杂得多的物体。由于我们的艺术管道依赖于组件化，因此我们的引擎是专门为利用这些组件化对象并简化组装过程而编写的。

每个对象都指定了一个特定的“拓扑”，用于控制子对象如何附着和遍历其曲面。拓扑可以看作是几何基本形状和高度映射的组合。艺术家选择最匹配对象网格的形状，并生成高度贴图来表示该形状与实际网格之间的差异。这使我们的艺术家能够用很少的努力构建复杂的静态对象。

对于独立开发人员来说，这是一个令人兴奋的时刻，因为越来越强大的工具在预算内可供公司使用。摄影测量就是这样一种工具。当在注重质量保护和自动化的生产流水线上使用时，任何规模的公司都可以构建高质量的 3D 游戏艺术，艺术家将时间花在工作台上而不是键盘上。

第 4.4 节优化资产

内存和处理能力是宝贵的，有限的资源，所以使用它们至关重要。游戏艺术家们经常把资产当作国家财政辩论的一部分来谈论，担心资产有多“贵”，担心资产是否触及“预算”，这并非偶然。由于游戏必须纳入固定的预算，所以优化对于开发者来说是一个长期的问题。数字可能会随着硬件的发展而增加，但艺术抱负和技术现实之间的冲突永远无法解决。即使在运行良好、计划周密的项目中，团队直到项目进行到相当晚的时候才真正了解其运行时环境的限制。通常，最初的估计结果过于乐观，这意味着几何体、动画和纹理的预算必须向下修正。很少有这样的情况，编程天才会释放出意外的内存，并使团队能够向他们的资产添加更多的细节。在这两种情况下，必须在资产的生命周期后期对其进行修改，以反映运行时环境的实际情况。假设一个“完美”的预算可以在生产开始前制定出来，就如同假设一个“完美”的开发计划是可能的一样危险。两种相反的力量影响资产的运行时成本。随着硬件变得更加强大，随着开发人员利用增加的容量，资产往往会变得更加内存密集。在20世纪90年代，一个典型的游戏角色模型可能有500个三角形、256kb的纹理内存和几百帧动画。今天，一个类似的角色可能有20000个三角形，8mb的纹理内存，以及数千帧动画。

同时，压缩技术也在不断改进，并进行了新的优化可以降低以前昂贵的数据结构的成本。例如，当存储高分辨率模型的表面信息的普通贴图纹理贴图，使应用它们的低分辨率资源在渲染时看起来更为详细时，它们首次出现在游戏中，则无法压缩。然而，现在有多种普通地图的压缩技术，它们在视觉质量和内存使用之间提供了不同的权衡。任何项目开始时的一项关键研究任务是尝试将新的压缩和优化技术融入到管道中，特别是那些“仅起作用”且不需要艺术家手动输入的技术。然而，问题最终不是技术问题。优化是关于艺术选择的，所以对于相关的艺术家来说，了解他们的资产在游戏中的使用方式是很重要的。担心远处山体的细节是没有意义的，但对于一个将在特写镜头中进行关键对白的角色来说，保留面部细节是至关重要的。某些资源可能不需要详细的模型，但需要详细的纹理。最终，这些都是美学上的决定，而不是技术上的决定：

计算机无法判断不断重复使用一些廉价资产是否会令人厌烦或者如果模型的块状轮廓对于游戏的视觉风格来说太粗糙。因此，没有任何技术解决方案能够完全消除优化过程中的人力投入。然而，管道团队可以通过提供良好的、易于访问的数据来极大地帮助这个过程。对于必须管理游戏运行时预算的艺术家来说，能够实时查看总资产成本和资产使用模式是至关重要的。通过与同类中的其他资源进行比较，可以很容易地发现需要优化的资源：如果一个游戏中的大多数树的重量都在1000个三角形左右，但其中一个树使用10000个三角形，这就很好地表明它需要工作。良好的使用信息还可以更容易地将很少使用的资产替换为更常见的资产：如果10000个三角形树在游戏中只出现一次，那么将其全部删除可能比优化它更容易。数据挖掘可以提供对游戏某些部分表现不佳的原因的洞察，或者提出更好的折衷方案，例如在纹理内存受到限制的区域交换需要更低分辨率纹理的更详细几何体。

第 4.5 节创建运行时动画

动画的使用方式决定了它进入游戏的最佳管道。为在 fmv（全动态视频：即预渲染的剪切场景）中使用而创建的动画将遵循类似于用于电影的管道，并可能外包给专门从事电影工作的公司。在前一章中，我们研究了为电影创建角色装配和动画的工作流程，其中的很多信息在这里也很相关。通过游戏引擎创建的动画-那些使用引擎设置蒙皮网格的动画并实时渲染它-将遵循类似的管道直到导出动画数据的点。通常，这些动画需要额外的信息来触发粒子效果、音频、人工智能或游戏事件。当序列播放时，玩家保持一定程度的控制并不少见。在一个极端，这个序列可能只是在玩家完全控制的情况下进行场景穿衣；在另一个极端，剪切场景可能控制摄像机和它前面的一

切。

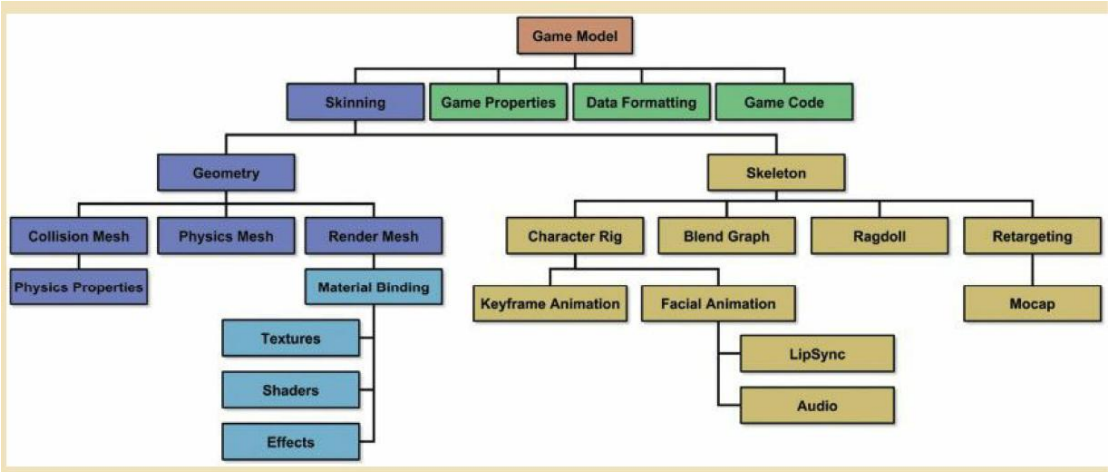


图 4.4 游戏管道中艺术品资产的依赖链。请注意，动画管道的结构与 vfx 工作管道的结构有多大的不同，有额外的动画数据源，如碎布玩偶物理。在这个图表中，深蓝色是几何体，浅蓝色是纹理和材质，绿色是数据、脚本和代码，金色是动画。

然而，游戏中的动画要复杂得多。虽然一些原始数据是以相同的方式创建的，但玩家角色和 npc（非玩家角色）的移动很少是简单的关键帧串：相反，游戏在运行时将各种较小的动画混合在一起。当角色在不同状态之间移动时，这可能是从一个动画序列到另一个动画序列的相当简单的过渡：例如，从站立到跑步再到行走。或者，它可能是一个非常复杂的动画相互叠加的混合体，可能通过程序修改来增强，例如使用反向运动学来防止角色的脚滑动，或者切换到物理模拟的“碎布玩偶”动画来响应像被拍摄的角色这样的事件，或者从楼梯上摔下来。这些元素在游戏中的组合方式可以通过使用专有代码或“中间件”来控制，即可以将现成的软件工具（如奶奶、语素或圣人）集成到游戏中。由于用于创建最终游戏动画的一系列输入通常显示为图形结构，因此运行时动画集通常称为“混合图”、“混合树”或“混合图”。由于混合在运行时由玩家输入或游戏引擎生成的数据控制，这些系统也被称为“参数化”动画。混合图形跨越了传统艺术作品和编程之间的界限，知道如何操作它们的艺术家通常是高技能的专家。

混合图中的每个节点表示单个动画或混合在一组动画之间。混合节点使用参数输入。一个简单的例子是由一个变量控制的两个动画之间的直接交叉淡入，当另一个淡入时淡出一个动画。一个更复杂的例子可以同时组合多个动画，在角色的上半身上播放一组动作，在腿上播放另一组动作，同时使用反向运动学和更改播放速度以防止脚打滑。实现不同动画之间的自然融合需要仔细的规划。重要的是，动画具有相似的动力学特性例如，如果不创建跳跃或滑冰运动的外观，则不能在行走和跑步之间混合。同样重要的是动画有相似的频率，否则结果将包含恼人的口吃。用于创建运行时动画的不同数据集之间的复杂关系网使得很难从动画中推断出在 maya 或 motionbuilder 中显示的游戏结果。由于这个原因，许多团队投资于专门的工具，让动画师可以实时预览他们的工作成果。这可以在专用编辑器中进行，也可以在游戏引擎中使用特殊模式。为动画师提供查看和测试其作品的能力是非常重要的。它们迭代的速度越快，结果就越好。在游戏中查看资源时，动画师可能会发现其动画存在运行时问题。在这种情况下，能够将所涉及的资产追踪回源代码是很有用的。这需要将调试系统写入运行时环境（这些系统通常在游戏编译发布时被排除），并且管道需要跟踪从艺术软件导出的文件与游戏中使用的资产之间的关系。如前所述，由于资产通常是在游戏引擎之外构建的，因此通常会将准备过程及其生成的任何错误记录在日志文件中。同样，很重要的一点是，这个报告过程必须设计得很好。有用的信息通常隐藏在“记录垃圾邮件”中——这些报告反映了细微的变化，或者根本不清楚，需要大量的技巧和努力才能提

取出来。

第 4.6 节-游戏中的面部动画

游戏中的面部动画通常比游戏中的要复杂得多电影，由于运行时性能的限制和游戏只包含更多内容的事实：虽然大多数电影运行两个小时或更少，AAA 游戏通常需要玩家从 20 到 50 个小时来完成。为这种数量的材料制作手键面部动画，甚至仅仅是手键唇部同步动画都是非常昂贵的。游戏通常严重依赖程序技术来驱动面部动画，特别是将嘴部位置与录制的对话或脚本文本相匹配。专门用于此任务的中间件解决方案，如 facefx 和 annosoft 开发的中间件解决方案，在整个行业中得到了广泛的应用，并为大多数游戏中的大部分对话提供了足够的质量水平。面部捕捉技术成本更高，但能产生更好的效果，通常用于特写镜头。两种主要的装配技术用于控制游戏中的面部动画：变形目标和骨骼。变形目标是描述单个面部表情或嘴部形状的网格，可以通过手动重新筛选角色网格，也可以通过捕获实时演员的面部表情来创建。通过在运行时在目标之间混合，可以创建面部动画的错觉。变形目标可以创建非常好的面部表情近似，包括皱纹和面部肌肉隆起等细微的细节，但是储存起来很昂贵，所以使用骨骼来变形皮肤的网状网的情况并不少见，就像骨骼可以用来控制全身动画一样。另外值得注意的是，在游戏中，与电影作品不同，从动画中计算几何缓存几乎是闻所未闻的：存储需求实在太大，在运行时更改结果太困难。

选择动画创建管道

在开始一个新的游戏项目时，要做的主要选择之一是使用什么方法来创建动画资产。从性能和运动捕捉解决方案到关键帧动画，您可以使用许多不同的选项。所有人都有自己的优点和缺点，需要被理解并与你的具体项目相关。一个关键的问题是：“角色之间的互动是静态的还是动态的？“静态交互，例如在预先渲染的电影中，是不受玩家或游戏中任何其他实体影响的交互。让我们考虑两个角色：吉姆和苏。如果吉姆和苏之间的对话在展示时总是以相同的相对位置展示，那么他们之间的对话将是静态的，并且在结束之前不会受到影响。在本例中，性能捕获可以很好地工作：除了清理数据之外，在录制的动画可以在游戏中使用之前，只需要很少的工作。动态交互是一种可以受玩家或游戏中其他实体影响的交互。例如，如果吉姆在谈话中对球员做出反应，如果球员在他周围移动，吉姆就会转向球员，如果球员靠得太近，吉姆和苏之间的对话就会是动态的。在这种情况下，性能捕捉不太可能是正确的解决方案，因为 Jim 可能需要单独的面部和身体动画。将运动和面部捕捉和/或手动设置关键帧的动画结合起来会更合适。

选择动画解决方案时要考虑的另一件事是您正在创建的游戏类型。如果你正在制作一个像《黑色洛杉矶》这样的游戏，玩家是一个侦探，必须审问嫌疑犯，高质量的面部动作捕捉是必须的，因为玩家需要看到角色表情的每一个细微差别。另一方面，如果你正在创建一个实时策略游戏，其中的角色从来没有显示超过一英寸高高的屏幕，面部捕捉将浪费宝贵的资源：相反，一个简单的手动循环“谈话”动画应该是完全足够的。

需要注意的是，单一方法可能不是在项目中创建所有资产的正确选择。了解不同类别的资产在游戏中的使用方式将有助于您为项目的每个方面选择正确的管道。

第 4.7 节 效果和效果

与电影作品不同，“效果”和“特效”这两个词通常有不同的含义游戏中的意思，特别是 rpg。如果角色被火球击中，由伤害造成的角色健康降低通常被称为“效果”，而造成屏幕爆炸的粒子则是“效果”。fx 是动画艺术资产，代表爆炸、烟雾或魔法效果。虽然它们可能是手工创建的，但通常使用与视觉效果中使用的类似的模拟技术，按程序生成它们。由于 fx 与游戏引擎紧密集成，为 fx 艺术家创建工具包的问题与为运行时动画人员创建工具的问题类似。如果艺术家必须不断地重新加载游戏，并使角色投掷手榴弹之前，他们可以查看爆炸效果，他们将有一个艰难的时间迭代效果的外观有效。不幸的是，由于 fx 团队通常规模很小，因此它通常是管道支持的最后一个团队。虽然使用 fork particle 和 bishamon 等中间件可以避免从头开始开发 fx 编辑工具的需要，但即使是现成的解决方案也需要管道团队的额外工作，才能在游戏中准确地查看结果。

第 4.8 节系统和水平设计

当电影资源被安排成镜头时，游戏资源通常被安排成级别。这个过程和镜头布局有几个重要的区别。然而，在我们研究这些之前，我们需要区分系统和级别设计。系统设计师是技术设计师，他们与程序员密切合作，开发游戏机制，例如玩家是否跳跃、如何跳跃、角色与对象交互的方式，以及角色清单中设备的功能。水平设计师（LDS）决定艺术资产的位置。系统设计器负责确定玩家在获取健康包时获得的健康状况;级别设计器负责确定该健康包是隐藏的还是在普通视图中。系统设计者负责决定玩家可以跳多高；级别设计者负责把窗台放好让他们跳上去。

关卡设计者的首要任务是将艺术资源放置到场景中。这些范围从背景对象，如遥远的地形或“天空盒”-旨在通过显示天空或无法到达的区域使级别看起来比它们更大的资源，到供玩家跳跃的平台，供他们战斗的敌人角色，供他们拾取或摧毁的对象，照亮场景的灯光，以及决定游戏性的触发器和其他逻辑对象。

但是，放置这些资源只是级别设计器工作的一半。它们的作用还在于确定资产的交互方式。这种“等级逻辑”可以简单到在拉杠杆时把门打开，也可以复杂到一场多阶段的战斗，在这场战斗中，玩家必须完成一系列的任务，才能使对手进入位置，然后将吊灯落在他们的头上。（请注意，“级别逻辑”与“游戏逻辑”是不同的，后者是系统设计者的职责范围。关卡设计师放置了导致吊灯掉落的触发器；系统设计师确定吊灯落在敌人头上时对敌人造成多大伤害。）

创建开放世界游戏的大公司可能会进一步细分这些任务，创建更多的专业角色。通常，“世界建设者”负责创建世界的布局，而级别设计者则负责创建其中的交互。然而，不同的公司在分工上略有不同。

系统设计师使用的工具在不同的管道之间有很大的不同。在一些公司，系统设计者直接在 visual studio 等工具中编写脚本；在另一些公司，他们使用 excel 等电子表格软件生成数据，这些数据稍后将导入数据库解决方案。用于关卡设计的工具链接到游戏引擎本身。没有所谓的“通用”级别设计工具：如果开发人员设计了自己的引擎，则会编写自己的工具；如果使用商业引擎，则级别设计人员将使用该引擎附带的级别编辑器。

所有级别的编辑器都有一些共同的功能：它们允许设计者创建、装饰和填充游戏世界。因为工作太多了涉及到放置在其他地方创建的资源，级别编辑器通常必须提供某种形式的资源浏览器（理想情况下，一种具有基于元数据的标记和搜索功能的浏览器）。编辑器还必须允许设计者放置与传统艺术资源无关的游戏对象：例如，告诉游戏玩家进入关卡的标记。有关该主题的更详细讨论，请参阅“级别编辑器的类型”。

第 4.9 节渲染和着色器管理

游戏引擎中的实时渲染器与视觉效果工作中使用的渲染器有很大不同。尽管一个视觉特效镜头的一个帧可能需要几个小时甚至几天的渲染时间，但实时图形受每秒渲染 30 到 60 次整个游戏世界的需要限制。为了使这成为可能，实时图形被渲染在专门的图形处理芯片（GPU）上，加速了用于创建图像的数学运算。

gpu 计算的关键是并行化。GPU 包含许多个人处理器，并排放置在芯片上，用于处理信息同时。这提供了大量的计算能力，但仅适用于可以并行执行而不是按顺序执行的任务。例如，许多脱机渲染都支持光线跟踪，这是一种模拟光线在场景中反弹方式的技术。光线跟踪可以产生精确的反射和折射，但它并不能很好地并行化：它本质上是一系列线性的数学运算，每一个运算都依赖于之前一个运算的结果。由于这个原因，gpu 传统上不适合光线跟踪，因此这种技术在游戏中仍然很少使用。

类似地，虽然脱机渲染引擎中使用的着色器可以是任意的复杂的是，游戏引擎中的那些被实时处理的需求所限制。在游戏中，有两种截然不同的方法来管理艺术资产与阴影之间的关系。一些工作室试图直接在他们的三维软件的视图。3ds max、maya 和 softimage 等工具使技术人员能够使用 hlsl 或 cg 等语言编写自定义着色器。以这种方式重新创建运行时着色器允许快速迭代艺术资源，因为不需要导出数据。

这种方法在第六代游戏结束时非常普遍游戏机（playstation 2、gamecube 和最初的 xbox），但在过去的 5 年中已经失宠。首先，由于游戏引擎的一些特性，如使用预先计算的光照贴图（用于模拟曲面照明的二维图像贴图，无需运行完整的照明计算）和运行时镶嵌，游戏中的阴影和视区版本精确相同的情况极为罕见，或者无法在艺术软件中复制。当在游戏中看到资源时，仅仅依赖于视区明暗器通常会导致不愉快的惊喜;但是如果需要将每个资源导出到游戏中以准确地查看它，则视区明暗器不会加快迭代过程。其次，维护同一着色器的并行副本对于管道来说是一个严重的复杂问题。除了它带来的数据依赖性问题外，如果未正确维护视区着色器，它们可能会使 art 应用程序本身崩溃，从而阻止 art 工作人员在诊断和修复问题之前执行任何其他工作。

由于这些原因，大多数工作室目前都不尝试直接在其艺术工具中再现其着色器的行为。相反，他们专注于缩短导出管道，以便艺术家可以在游戏中或在共享同一渲染引擎的专用编辑器中看到他们的资产。这种系统的主要挑战是确保资源绑定到正确的 ingame 着色器。最常见的方法是将游戏中的材质与在 3d 应用程序中按名称创建的代理材质进行匹配。在这个系统下，艺术家将他们的 3d 软件的一个内置材料分配给一个资源，并给它一个独特的名称。在游戏编辑器中，将为导出资源中存在的每个命名材质创建材质球。然后可以在游戏的渲染器中编辑这些内容。这鼓励艺术家在游戏环境中迭代，并使他们能够实时查看其更改的结果。

与混合图一样，着色器跨越了艺术和编程之间的边界。许多现代商业游戏引擎，特别是虚幻引擎和 Unity，使艺术家无需学习编程就可以通过链接一组预先构建的组件来创建自己的着色器。其他工具提供了一个类似于程序员文本编辑器的编辑环境，具有自动完成和语法突出显示功能，以提高工作效率。

允许艺术家直接创建阴影提供了一个引人注目的优势，即将项目的视觉方向牢牢掌握在艺术部门手中。然而，它也会导致性能问题：没有工程训练，缺乏准确预测其着色器网络实际成本所必需的工具，艺术家经常被指责通过创建过于复杂、效率低下的着色器来降低游戏的帧速率。

一个常见的折衷方案是让艺术家提供“第一关”或“原型”明暗器，用于建立资源的外观，但在工作结束之前，将由明暗器专家编写的更有效的版本替换。当然，通常情况下，这些计划中的替换会偏离计划，并且所运送的着色器会很混乱且效率低下。为了避免这个问题，许多团队提供了一个单一的一体式着色器，涵盖了所有允许的渲染技术。这些“ubershaders”通常使美工人员能够逐个激活特殊功能，通过勾选“shader editor”中的复选框添加折射或程序噪声等特殊效果。ubershaders 提供了一个更可预测的性能封套（例如，很容易设计编辑器 ui 以防止美工人员同时打开所有昂贵的功能），并有助于最小化项目所需的着色器数量。这一点很重要，因为切换材质球是 GPU 上计算成本最高的操作之一，而动态照明等效果可能会导致材质球计数膨胀。不过，ubershader 确实倾向于使游戏的外观同质化。

在过去的几年里，许多游戏都采用了一种新的渲染技术这使着色器管理的工作更加复杂。2007 年之前，大多数着色器系统允许每个着色器直接写入最终渲染图像：一种称为“正向”或“直接”渲染的方法。许多现代系统试图通过将渲染过程分解为一系列离散的过程来提高性能和真实感，在这些过程中，图像的不同方面被分别渲染并在屏幕上组合，这与在过程中渲染图像并在视觉效果工作中在合成器中重新组合的方式非常相似。这种被称为“延迟”渲染的方法可以更有效地处理复杂的灯光效果。但是，它增加了着色器作者的赌注。在延迟渲染器中，明暗器需要以严格定义的顺序写出一系列“缓冲区”或图像层，而且很少有生产美工人员具有处理此问题的技术背景。在许多情况下，具有延迟渲染系统的团队必须与 UBeSoeLead 方法简单地管理它们的图形引擎的复杂性。

构建游戏引擎

正如我们在本章中所指出的，游戏引擎执行许多任务。其中包括我们在这里所涉及的内容，包括场景组装、生成运行时动画、物理模拟、AL 和渲染，以及一些我们没有涉及的内容，包括音频、内存管理、处理器线程、网络、流和本地化支持。因此，从头开始编写自己的引擎通常是拥有大型编程团队的大型工作室的专利。较小的开发人员通常选择授权商用引擎（游戏行业网站 development 最近发布了一个主要替代方案的概述：http://bit.ly/yosx9r），或者通过连接一组现成的中间件来构建一个。目前可用的大多数主要工具的列表可以在网站 gamemiddleware.org（www.gamemiddleware.org/middleware）上找到。

管道的基本功能

第 5.1 节您将从本章中学到什么

在前面的章节中，我们研究了在生产过程中创建数据的方式。在本章中，我们将了解如何管理数据。我们将首先探讨管道的一些通用设计标准，以及影响它们的问题 – 例如对技术标准，数据交换格式和“微管道”的需求 – 然后再看看数据管理的四个关键方面：目录结构，文件 – 约定约定，元数据和版本控制。最后，我们将研究如何利用您正在跟踪的生产数据。

第 5.2 节管道的作用

生产流程的目标是以高效且经济的方式生产产品。要实现这一目标，必须做两件事。首先，它必须管理从一个进程到另一个进程的数据流。其次，它必须指导工作流程从任务到任务。

这两个子目标是相互关联的：要管理工作流程，必须先了解数据流。想象一下建房子。在这个类比中，“产品”是一个完工的建筑。 “数据”是由砖，木材，石膏，电线等组成的材料 – “工作”就是：施工人员完成建筑所必须完成的任务。

这个任务列表包括安装布线和抹灰墙壁。由于电线进入墙内，电工必须在泥水匠完成工作之前安装它们。如果你不理解这种关系，施工进度将是错误的：要么在电工完成之前，泥水匠必须无所事事，否则电工将不得不摧毁泥水匠的工作来安装电线。电影和游戏制作也是如此。如果您不了解必须在动画开始之前完成模型和装备，动画师将坐在那里没有任何动画。

值得探索创造艺术与创造实体产品之间的类比。我们使用术语“管道”将我们的流程包裹在工业能力的氛围中。昼夜嗡嗡作响的巨型炼油厂的形象，不断将艺术灵感的原材料转化为精美的像素，这是一个非常令人放心的形象。它表明了强大的力量和令人安慰的可预测性：一方面是创意，另一方面是产品。在这两者之间，白色涂层的技术人员监控整个过程，当他们看到表明事物顺利流动的表盘时，他们点头。

不幸的是，在现实世界中，生产管道更像是苏斯博士的书籍，而不是工厂的那些，充满了环回，短路和面孔。通常情况下，电影或游戏管道与装置一起保持，甚至连帽子里的猫都不敢依赖。那么为什么呢？为了回答这个问题，我们来看看为什么管道发生变化。

第 5.3 节管道变更原因

在某种程度上，管道因技术而发生变化。新技术意味着新类型的数据，创建它的方法通常必须在生产过程中加入到现有的管道中。一个用于流体模拟的新产品出现了，现在有必要弄清楚它如何与现有的着色器库一起工作。或者开发了你一直依赖于刚体动力学的插件的公司倒闭了，让你决定是否坚持使用不再更新的工具，或编写自己的版本。

更重要的是，管道改变是因为它们被人们使用，而不是机器。管道实际上是复杂的人类活动网络的技术表达，包括从最高级的创意总监到最卑微的生产助理的每个人，这些人的品味与任何技术限制同样重要。围绕满足您所有技术目标的软件构建管道，但是您的艺术家无法忍受这种管道是徒劳的。管道开发最终是服务业务，而不是工程项目。

正如十九世纪的军事战略家陆军元帅赫尔穆特·冯·莫尔特克所说，没有战斗计划，无论多么出色，都能与敌人保持联系。在生产中，这可能被重新描述为“没有管道与客户保持联系”。该工作室对剧本的最新改变需要一种新的，意想不到的效果。或者游戏的多人版本证明非常重要，你必须建立五十个玩家角色，你已预算五个。时间表改变。资金不断下降。制作电影或游戏并不像制造汽车：在“完成”之前，你可以做更多的工作。

第 5.4 节定义目标

为了应对变化，你需要仔细挑选自己的战斗。您的资源有限，您将没有足够的时间同时为生产的每个方面提供服务。您需要对项目有一个坚实的战略性理解，这样您才能在规划期间和生产过程中出现不可避免的危机时优先考虑您的决策。让我们来看看你需要问自己的关键问题。

项目的核心价值是什么？

每个项目都有一些定义功能，您需要清楚地表达这些功能，以便能够很好地支持它。一部关于海盗的电影需要一套强大的工具来创建流体模拟;在梦幻城市中的一套需要创建建筑物的工具。具有严格脚本线性游戏玩法的游戏可能沿着同样线性的生产路径前进;一个拥有强大多人游戏的人需要在关卡设计中快速转变，因为当 beta 测试人员发现破坏游戏的攻击时。

这个项目有多大？

项目的规模和复杂程度各不相同，所需的管道也相应变化。在具有长时间表的大型项目上投资定制技术通常更为明智：可能有时间为一个游戏开发一个新的关卡编辑器，需要两百名艺术家三年才能制作，但是六个月内由两个人完成的独立游戏可能不得不使用现成的软件。但是，也有例外：小型团队也必须长期思考，定制软件可以成为精品工作室的独特卖点。相反，艺术家需要接受如何使用定制软件的培训，因为大型工作室可能不想破坏关键项目以使每个人都加快速度。

我们可以重用我们开发的工具吗？

您创建的每一段代码都应该考虑到模块化的概念，以便下一个项目的管道不必从头开始构建。然而，即使对于大型设施，创建能够承受持续技术进步压力的软件也会非常困难。其他形式的技术也是如此，包括运行软件包所需的机器以及在它们之间传输数据的文件格式。有时，“建筑扔掉”可能是务实的选择。

还有其他技术限制吗？

项目不是在真空中出现的。在大多数情况下，在项目开始之前，将为您选择生产环境的许多重要方面。出于商业原因，您的设施可能已经承诺使用特定的软件或特定的外包供应商，即使您的特定项目可以通过其他方式提供更好的服务。在游戏中，您的选择可能受到游戏引擎或目标平台选择的限制。最后，但同样重要的是，您必须在限制您选择工具和供应商的预算范围内工作。John Pearl 首席艺术家，Crytek USA

管理遗留系统

多年来，我一直负责监督在许多不同的工作室创建许多不同的管道。最好的管道是那些你在使用时不必考虑的管道：它们只是起作用。虽然没有神奇的方法来创建其中一个，但我学到的一件事是记住遗留系统。

遗留工具可能是一个巨大的瓶颈，尤其是在游戏行业，软件平均每六个月就要更换一次。有时候，只要更新到最新版本就可以解决这些问题，而这通常是最便宜的方法。但如果你使用的是最新版本的软件，而它仍然没有完成你需要的工作，那就上网看看吧。周围有很多初创开发人员，他们一直在创造有趣的新方法来解决常见的开发瓶颈。

如果两个选项都不可用，并且您发现需要坚持使用遗留软件，请在您的管道计划中说明。不要气馁：还有其他方法可以改进管道。

最难克服的遗产问题是“遗产思维”。每当你创建一个新的管道，你遇到阻力幸运，通常不是一个侵略性的类型，但仍然是一个挑战。常见的反对意见包括“到目前为止对我们有效”和“我对现状很满意，所以我不想改变。”这两个观点都是正确的，但它们也扼杀了项目的潜力。很容易处理“到目前为止对我们很有用”的问题。如果你知道，在你提议的管道中，处理一项资产需要 2 分钟的时间，而在传统管道中，则需要 10 分钟，每个资产每人 8 分钟的浪费是一个令人信服的原因。这显然是一个黑白案件，但类似的例子经常出现。没有人关心一个团队的工作效率可能会与严格的数字相矛盾。如果这些变化的好处难以量化，或者你无法与之前的管道进行比较，那么最好看看你希望替换的管道，然后问：“它真的对我们有用吗？“。

在我工作过的一家公司，我们所有的 3d 资产都使用了一个 3ds max 着色器。这已用于前三个项目，并随着新的技术需求的确定而增加。但是，由于没有删除任何内容，在 6 年多的时间里，该着色器已累积了 100 多个输入和参数，有些完全多余。这些名字很难浏览，而且大多数名字都很隐晦，曾经对那些不再在公司工作的人意味着什么。更糟糕的是，一次失误就可能破坏游戏中的资产。“有用”的借口，为什么要改？“超过了常识。更难对付的反对意见是“我对现在的情况很满意”，而“迄今为止工作过”的阵营通常是由那些没有真正使用管道的人组成的，他们有从过程的外部来看，“舒适”营是由每天使用它的人组成的。有时候，他们被过去的管道变革烧死了；有时候，这只是归结为一个事实，大多数人不喜欢变革。虽然你通常需要让“到目前为止工作”的人相信你的决定，但归根结底，你是在为“舒适”的人做准备。了解他们的需求以及他们与当前工具集的联系在这里至关重要。最不喜欢改变的人是通常，最不满意他们现有工具的人是：他们只是精心设计了自己的解决方案，结果发展出一种斯德哥尔摩综合症。在这种情况下，很难用语言来改变主意：通常最好表现出来。一步一步地让犹豫不决的开发人员通过你提议的管道。听他们说：那些日复一日地使用管道的人，会有一种你单靠观察无法获得的洞察力。然后向他们展示你正在努力在你的修订计划中解决他们的问题。让这些人感觉到他们正在你身边精心设计管道，这有助于他们感觉到对变革的投入，而不是仅仅被强加给他们。即便如此，你可能仍有批评者。在那一点上，你只需要继续你的计划。改变是游戏开发者工作的一部分：那些不接受改变的人最终会落在后面。这些年来，我见过这样的情况，人们拒绝成长，他们的技能也在萎缩，直到最终让自己过时。最好的希望是，一旦新的管道到位，他们将欣赏其效率，并与团队其他成员拥抱它。传统技术不必成为障碍。事实上，提前处理会让你的渠道更强大。做那些你需要证明你的案例的研究可能会给你带来其他你本不想做的改变或补充。在这个过程的最后，你会感到自信，你正在做最好的管道你可以。

艺术家们喜欢什么？

众所周知，艺术家对软件很挑剔，他们倾向于依附于他们最熟悉的工具。你可以通过切换到一个新的艺术包来节省金钱，但你必须忍受很多关于它不熟悉的用户界面的抱怨。管道开发人员(他们经常知道并喜欢许多不同工具的各个方面)很难同情艺术家坚持他们喜欢的软件的倾向。但是，生活中的事实需要包含在您的计划中。如果不是彻头彻尾的叛变，试图强迫你选择的工具在不情愿的劳动力上可能导致数月的生产力降低。选择的明确理由，加上良好的培训，可以避免过渡，但这个过程仍需要机智和自由裁量权。您需要像预算一样仔细考虑最终用户的政治资本。请记住，软件的变化会对招聘产生连锁反应。新包装的确可能比旧包装快三倍，但如果没有一位自由职业者知道如何使用它，您可能会浪费更多时间而不是保存。

艺术家需要多少反馈？

艺术家的经验将决定您的管道设计的重要方面。经验丰富的员工可能需要从您的工具中获得非常详细的技术反馈 - 反馈更多的初级艺术家会发现令人生畏和困惑。专家可能会受到保护措施的挫败甚至侮辱，以防止没有经验的用户危及工作流程。同样，为那些只在项目上工作几个月的外包商设计的工具必须比那些可能需要多年才能掌握其微妙之处的长期员工更简单，更强大。

世界上有多少艺术家，他们在哪里？

小型团队可以依赖电子邮件或即时消息，但大型团队 - 尤其是那些分布在全球的团队 - 将需要一个更加精细的协作平台。国际合作通常可以用英语进行，但您仍需要了解其他本地化问题：假设您的外包工作室的所有艺术家都将拥有最新版本的 Windows 或在其工作站上始终保持互联网连接是不明智的， 例如。

还有其他的文化因素吗？

组织文化的其他方面难以量化，但可能对管道开发产生深远影响。一些团队热情关注他们工具链的外观;其他人都是严格的功利主义者，他们鄙视花哨的偶像，浪费时间和金钱。不幸的是，没有一种通用的方法来揭示这些偏好 - 软件开发的文献充满了对用户故事，执行摘要，功能规范，UI 线框，问答环节，测试用例，相关优点的讨论。等等 - 但承认它们的意愿是成功管道开发的关键。值得注意的是，有多少杰出的技术艺术家准备花费数天时间缩短文件阅读时间，却没有投入相同的精力来充分利用其他人。

我们必须面对哪些制约因素？

通常，确定可以完成哪些工作的最重要因素是可用的时间。在第一次构建管道时，请关注“必备”：一套完整的工具，自动完成在合理的时间范围内无法手动完成的任务 - 最重要的是稳定性。您创建的每个工具都应以一致，可预测的方式运行。在设计稍后需要构建的低级功能时，这一点尤为重要。只有当所有的基础知识到位时，你才能开始关注“有趣的人”。

让合适的人参与进来

在规划和建设管道之前，与合适的人协商是很重要的。其中包括设施内多个不同部门的代表，以及不同级别的资历：

•管理层：定义他们对公司业务发展的预期，以及他们认为的管道战略目标。

•监管者和制作者：从后勤角度监督项目的人员，以及能够确定他们希望管道日常运行方式

的人员。

•创意人员：每个创意部门的关键人员，他们可以解释与渠道合作的艺术家的需求。

•技术人员：就任何与基础设施有关的事项提供咨询，包括新的技术进步和在发展工作方面可行的事项。

在决策过程中保持参与者之间的适当平衡是至关重要的。这包括平衡技术目标和实现艺术家友好的工作流程，平衡“厨房里厨师太多”和让关键声音无人听闻的风险。这里需要注意的是，没有两个艺术家或程序员是一样的。在规划管道时，几年的额外经验可以起到很大的作用。如果你找到了一个有着合适经验的合适人选，一份工作可以在很短的时间内完成。对于技术人员来说，这一点尤为重要。人们通常认为，建模师、动画师、打火机和其他艺术人员都有不同的技能，而程序员往往被视为完全错误地认为可以互换。所涉及的人员（特别是主要管道建筑师）需要考虑公司的长期利益，与年项目的短期需求相平衡。生产。他们应该是长期的全职工作人员，而不是按项目雇用。人员的连续性消除了每个管道开发周期开始时耗时的学习阶段，并将管道结构中的随机“搅动”最小化。所有相关人员，尤其是主管，都需要了解管道的目标，并了解与之合作的最佳实践。监管者应该随时督促他们监督的人员在管道内工作。

第 5.5 节定义标准

接下来，我们将介绍设计管道时出现的一些技术问题。第一个是需要定义技术标准。这些标准有以下几种形式：

该设施将使用哪种软件来执行哪项任务

在这些应用程序之间共享数据的文件格式

工作流标准，描述创建资产的步骤以及与管道其余部分共享的方式

将作为元数据存储的这些资产的属性

组织标准，例如安排任务或进行资产审查的方式

这些标准对您的设施而言是独一无二的，因此务必确保工作人员充分理解这些标准并明确记录。当您开始一个新项目时，良好的文档将减少新员工加快速度所需的时间。每个标准应该是独立的，独立于管道的其他部分，并且是不可知的：其文档应该说明所需的结果，而不规定或约束所选择的解决方案。

一个具体的例子可能是将新模型添加到资产管理系统的方式。该标准将以抽象术语定义构成“有效”模型的内容（例如，几何是否可以包含 n-gons 或仅包含四边形，文件的最大大小以及文件的命名方式）。然后，对于开发人员来说，实现一个系统可以自动导出软件中的有效模型，就像个别艺术家手动输出一样;由此产生的模型应该是相同的，因为它们符合相同的标准。

第 5.6 节文件交换格式和脚本语言

一种特别重要的标准类型是数据在工具之间移动的格式。图形制作通常涉及使用许多不同的第三方软件包，管道开发人员的工作是确保它们彼此协同工作。例如，准备好在游戏中使用的模型可能涉及从 ZBrush 获取高分辨率的雕刻，并进入 xNormal 以创建在 Maya 中使用的法线贴图;创建着色器可能涉及从 Photoshop 中获取特殊打包的纹理贴图到 FX Composer 中。

不幸的是，这些不同的包不共享本机文件格式。即使它被认为是他们存在的理由，DXF，FBX 和 Collada 之类的交换文件格式因其无法可靠地移动数据而臭名昭着：虽然可以将数据从一个包

传递到另一个包，但这个过程很少是平滑或可预测的。

Alembic 是一个由 Lucasfilm 和 Sony Pictures Imageworks 领导的开源开发项目，旨在解决这个问题 - 至少就共享动画的几何数据而言。 Alembic 格式被许多第三方软件供应商采用为一种不断发展的标准，并且最终可能会被普遍接受。 Hannes Ricklefs MPC 管道全球主管

另一个问题是软件包不共享通用的脚本语言。今天的大多数主要应用程序都包含某种脚本，但是使用的语言种类繁多 - 从 Python 等通用语言到仅存在于一个应用程序中的专有语言。缺乏通用标准使得设施难以支持用多种语言编写的工具，通常需要大量的冗余工作。

在整个行业中，特别是在视觉效果方面，Python 被广泛用于管道集成。还有特定的开源开发，例如 Cortex - 一组 C ++库和为 VFX 工具开发工作量身定制的 Python 模块 - 旨在提供可嵌入任何应用程序的通用框架。

第 5.7 节微型管道

生产过程中出现的另一个技术问题是需要在较小规模上进行开发。虽然本书的主要关注点是开发大型，端到端，跨部门的管道，但重要的是要认识到微管道的影响。在生产中，宏观管道提供了规则;微管道认识到规则被打破的事实。

即使在宏观层面，管道可能适合设施的需求，但不可避免地会出现不适合的情况。在视觉效果中，这些通常是单独的镜头，其创造性要求打破了模具，需要一次性解决方案，与所讨论的镜头一起生存和死亡。虽然希望限制这些例外，但如果只是为了开发人员的理智，商业压力和艺术自由要求管道应该能够在必要时支持它们。幸运的是，这些微型管道的影响只能由少数从事拍摄工作的艺术家感受到，有时只有一个人。

或者，可能会出现部门内工作流程，这些工作流程在生产结束时不会被丢弃。这些通常最初是为解决新需求而设计的临时解决方案。这些变通办法形成了一条微管道，以后可以与整个管道统一起来。

在这种情况下，微管道可能需要在统一之前进行“消毒”。在某些情况下，艺术家只需在现有参数中重新配置其工作流程 - 但更常见的是，他们将开发自定义脚本或插件。为了避免需要更改周围管道的连锁效应，可能需要重写这些工具，以使其输入和输出与宏管道当前所需的输入和输出相匹配。

第 5.8 节数据管理策略：概述

生产过程中出现的主要技术问题是需要管理数据。大型现代游戏或视觉效果制作会产生难以想象的大量数据：数十万个文件，占用数 TB 的存储空间。幸运的是，如果您专注于用户关心的内容，这个庞大的文件云会凝聚成更易于管理的内容。

当一个艺术家的任务是修复一个看似抖动的运行周期或一个看起来不受欢迎的环境时，对话总是从一个有形的问题开始：“食人魔的动画是几帧太长”或“我们需要更多的树来筛选出来寺庙水平的露台“。管道开发人员的任务是确保艺术家能够将这些有形问题转化为无形的数据世界，找到食人魔的运行周期的动画文件，或者为寺庙设计关卡。

在本章的其余部分，我们将探讨管理数据的四个关键策略：

设置直观的目录结构

采用良好的文件命名约定

使用元数据使文件可搜索

使用版本控制或版本控制来确保文件保持最新

我们将在这里简要介绍每个主题，然后在后面的章节中更详细地回到其中一些主题。

第 5.9 节目录结构

生产中的每个文件都为不同的人扮演不同的角色，但是计算机操作系统坚持认为一切都只适合一个地方而且只有一个地方。将文件隔离到文件夹层次结构当然可以避免因将文件作为单个列表查看而导致的信息过载，但使文件夹有用的功能 - 将信息过滤到可管理大小的能力 - 实际上是一种旨在隐藏的技术而不是定位数据。文件夹无法控制您可以看到的内容：它们控制着您无法控制的内容。

一个现代化的管道将补充这一工具，提供更灵活的数据视图 - 我们将在本章后面的内容中查看其中的一些内容 - 但是有一个组织良好的目录结构仍然很重要。无论您拥有多么精美的数据库或资产浏览器，在某些时候您都会发现自己手动点击查找特定文件的文件夹。因此，找出一套用于归档数据并坚持下去的规则。更好的是，清楚地记录它，确保您的团队理解其基本原理，并使用可以更容易地将文件保存在正确位置的工具进行备份。但无论你做什么，都不要把事情搞得一团糟。

良好层次结构的根源是功能性的，基于谁将要查看数据。如果您可以合理地确定不同的人群可以查看不同的文件集，那么这是将它们分成自己的文件夹的好机会。例如，管道工具的源代码应该在层次结构中具有自己的分支，与艺术文件或生产数据分开。游戏项目通常希望将游戏代码与艺术资产分开，而电影可能希望分割数据，以便承包商只能看到他们正在处理的项目部分。国际项目可能希望将核心组件与一个区域的核心组件分开。

在艺术资产本身中，一些进一步的细分是显而易见的。例如，在电影中，视觉效果数据通常以项目序列拍摄的方式组织，因此在目录结构中有三个相应的级别是有意义的。

在艺术资产本身中，一些进一步的细分是显而易见的。例如，在电影中，视觉效果数据通常以项目序列拍摄的方式组织，因此在目录结构中有三个相应的级别是有意义的。

这些论点很常见，也是不可避免的。人们不像电脑那样思考;他们使用重叠的模糊类别，而不是任何/或选择的严格等级。更糟糕的是，层次结构不仅仅是帮助他们查找数据的一种方式，而是一种标记草皮的方法。在有草皮的地方，有草皮战争。

用户希望数据隔离的方式通常因部门而异。例如，动作捕捉部门可能希望将所有捕获和设置数据保持在基于拍摄日期的层次结构中。另一方面，将要使用该数据的动画师更喜欢围绕他们自己的工作流程进行组织：按角色或按场景。建模者可能希望以更容易共享纹理和着色器的方式对资产进行分组，而环境艺术家可能更喜欢按功能组织它们，从而更容易浏览武器，道具或其他类型的资产。

有这么多相互矛盾的观点，设计层次结构总是一个有争议的过程。在可能的情况下，尝试在需要知道的基础上这样做：有没有办法分割文件夹，以便您只向团队的某些成员展示其中一些？但最终，你会遇到无法调和的冲突。如果您无法代理妥协，则需要选择一个选项并使用它运行。记录决策背后的基本原理，以便用户可以有效地使用系统，即使他们不同意;并计划加强组织结构的工具：例如，通过自动在正确的位置归档资产。

构建层次结构时，为命名文件夹制定明确的规则也很重要，尤其是在使用混合操作系统时。Windows 不区分大小写，因此在 Windows 计算机上，路径 c：/ path / to / a / file 和 C：/ Path / To / A / File 之间没有区别。但是，在标准的 Unix，Linux 或 Mac OS X 计算机上，这两条路径将指向磁盘上完全不同的位置。为了避免问题进一步发生，您需要选择适用于您的系统组合的约定并在您的工具中强制执行。我们将在下一节中更详细地介绍命名约定。

第 5.10 节文件命名约定

除了决定目录结构外，设置命名文件和文件夹的约定也很重要。描述性文件系统至关重要。如果所有其他方法都失败了，如果直观地命名文件，艺术家将能够通过标准的“打开”和“保存”对话框找到所需的资源。

令人惊讶的是，对于这样一个看似平凡的话题，命名约定是管道设计中最激烈争论的问题之一。我们将在后面的章节中更详细地讨论这一争议。但就目前而言，让我们来看看一些关键问题。

命名约定的主要关注点是为资产提供唯一的全局标识符。例如，VFX 工作中常用的镜头命名约定包含多个标识符。在诸如“XX_TR_001”的文件名中，“XX”表示项目，“TR”表示场景，“001”表示镜头。

仅基于数字或代码字母的命名约定将无法导航，因此使用人类大脑可以直观理解的标识符非常重要。例如，包含构建字符的内容的所有目录可能使用前缀“char”：“charBob”，“charFoo”等。同样的逻辑可以扩展到“vhcl”（即车辆），“prop”等等。即使使用基于标准文件系统的标准搜索和列表操作，使用此类标签也可以轻松找到包含字符的所有文件夹。

某些工具还使用文件名来存储有关这些文件状态的额外信息 - 例如，在视觉效果工作中，您可以将“_render_”添加到文件名以表示场景已准备好进行离线渲染。其他设施使用后缀来指示文件是“WIP”（正在进行中），“最终”还是“已批准”。通常包括最后一个人在文件上工作的名称或首字母，或版本号：我们将在本章后面讨论的问题。

遵循严格的命名约定，可以将自动化内置到管道中。

如果文件名以管道可以解密的系统方式形成，则放置在特定文件夹中的资源可以自动集成到镜头或游戏级别。这使艺术家无需运行其他工具即可将内容添加到作品中。

当然，结构化命名约定的所有好处只有在文件名正确形成的情况下才能实现，因此依赖此系统的工作室通常会在软件应用程序中创建工具以确保合规性。这些工具应该提供一种方便的方法来填写命名约定所需的信息。假设您要确保存在拍摄编号：“保存”对话框应该有一个字段，艺术家必须在完成保存之前键入数字，而不是依赖它们手动添加信息。类似地，当导出到游戏时，可以向游戏动画制作者呈现已识别动作的列表：例如，从已经预先填充有动作的列表中选择“步行”或“跳跃”，游戏知道如何正确地格式化并且保存到适当的文件路径。

当然，结构化命名约定的所有好处只有在文件名正确形成的情况下才能实现，因此依赖此系统的工作室通常会在软件应用程序中创建工具以确保合规性。这些工具应该提供一种方便的方法来填写命名约定所需的信息。假设您要确保存在拍摄编号：“保存”对话框应该有一个字段，艺术家必须在完成保存之前键入数字，而不是依赖它们手动添加信息。类似地，当导出到游戏时，可以向游戏动画制作者呈现已识别动作的列表：例如，从已经预先填充有动作的列表中选择“步行”或“跳跃”，游戏知道如何正确地格式化并且保存到适当的文件路径。

将命名约定编码到工具中也会使它们更难转移到新项目。这可以通过以下方式构建工具来缓解：非程序员可以更改自动化的工作方式：例如，通过在设置面板中公开所有假定变量（如文件夹路径和命名前缀）。然而，即使是最好的工具设计师也无法预见每一种可能性，因此自动化将不可避免地为您的管道增加一定程度的不可预测性。 Tim Green 高级程序员，超大型游戏

目录扫描和条件文件

高级程序员，超级大容量游戏当一个工具需要加载数据时，分布在许多文件中，有两种常见的选择。给定一个根文件，工具可以从该文件中读取文件引用，将这些引用解析为绝对路径，然后加载这些文件。或者，可以给它一个文件夹路径，并对该文件夹中的所有文件进行目录扫描。后者更易于编码和维护。在前一个系统中，如果要添加或者删除一个文件，您需要首先编辑根文件；而目录扫描只需要将文件保存到适当的文件夹或从中删除。

但我相信目录扫描是邪恶的。它诱惑着你使用方便，然后，当你最不期望的时候，转身咬你。让我解释一下。

通常，文件既存在于用户的本地驱动器上，也存在于修订控制系统（如 PurrCE）中的主副本中。用户在源代码管理期间同步这两者，更新本地文件夹的内容。如果您创建了一个新文件，它将在该文件夹中创建、测试，然后签入源代码管理，这样当您的团队中的其他人同步时，他们将收到一个副本。只要一切保持同步，就可以正常工作。但问题是：流氓文件经常进入本地文件夹。这些文件不存在于源代码管理中，只存在于您的机器上。它们很容易在文件夹中数以千计的其他类似名称的文件中丢失。在游戏中，结果是拥有流氓文件的艺术家将看到游戏在他们的机器上的行为与团队其他人不同。这可能会导致管道工作人员浪费时间来解决问题是——我个人花了几百个小时在其他人的机器上搜寻流氓文件，或者更糟的是，艺术家签入他们的内容只会为团队的其他人破坏游戏，因为流氓文件需要使其正常运行。条件文件是管道根据文件的存在而改变其行为的文件，存在和不存在都产生一个有效的结果。它们与目录扫描有同样的问题，出于同样的原因应该避免。

除了确保正确命名文件外，系统可能还需要确保以正确的格式保存文件。这比听起来更困难。你多久尝试打开一个带有.jpg 扩展名的文件才意识到它实际上是一个 TIFF 图像？因此，可能需要依赖诸如“幻数”之类的系统：将文件格式定义为以唯一标识符开头的实践。例如，以 JPEG 格式保存的文件始终以十六进制字符“ff d8 ff e0”开头。

管道开发人员可能会再次编写工具来确保艺术家以正确的格式保存资产。例如，如果以 Maya 的本机 MA 格式保存，则无法在 Nuke 中打开在 Maya 中创建的摄像机。为确保在合成期间可以使用管道中早期创建的数据，可能需要在 Maya 中编写一个工具，提示艺术家将相机保存为 FBX 或 Alembic 交换格式。

第 5.11 节元数据

我们已经讨论了层次结构如何不是真正用于查找信息。相反，他们擅长隐藏它。这并不完全是一件坏事：如果你的项目包含数十万个文件，你真的不希望在大多数时间里看到它们中的大多数，如果它们被呈现在你们身上，你将永远无法找到任何东西。单个清单。

但是，当您知道自己想要什么并且想要快速检索它时，分层文件夹设置可以快速阻挡您。我们已经指出，不同的部门 - 甚至不同的人格类型 - 将构建一个非常不同的项目心理图。一个人高效，合理的布局是另一个人的混乱噩梦。

幸运的是，文件夹不是查找资产的唯一方法。我们大多数人都拥有数千甚至数万首歌曲的音乐库。但只有最硬的硬核设计了一个文件夹树来组织它们。相反，我们让 iTunes 或 Windows Media Player 处理文件存储位置的问题，使用其他类型的信息来浏览它们。

此信息类型，艺术家姓名，日期等称为“元数据”。它看起来很平凡，因为我们每天都看到它，但事实上，它是一个复杂的数据库系统。它在最好的意义上是复杂的：我们认为它是理所

当然的。

元数据是查看文件的一种更强大的方式，而不仅仅是在树视图中上下跳跃。与文件夹不同，它是包容性的，不是独占的。通过元数据进行组织，将填写信息的闷热，隐藏的业务转变为更有效的过程，以便在当时满足您需求的任何路线找到您所需的内容。如果这些需求发生变化，它不会强迫你在磁盘上实际改变它们。

由于基于搜索的导航比导航目录层次结构要快得多，它正在取代大多数计算领域的层次结构：只要看看像 Google 这样的现代搜索引擎取代雅虎推广的老式索引，或稳定 Spotlight 在 Mac OS X 中的增长。由于您的用户肯定熟悉搜索网络或搜索硬盘的过程，因此您的工具以相同的方式工作是有意义的。

在这样的系统中，文件被元数据“标记”的方式非常重要。遗憾的是，标签不是标准的 OS 功能，因此需要由各个应用程序来实现它们。标签可以以多种方式存储：在数据库中，在文件系统元数据中，在磁盘上的专用元数据文件中，或甚至直接在其他文件类型中：例如，Microsoft Office，Photoshop 和 MP3 文件都包含固有元数据像作者姓名和版权的东西。

但标记的真正挑战不是技术，而是语义：计算机在你说出之前不会知道给定资产是什么。幸运的是，您可以通过利用旧式文件层次结构免费完成大量标记 - 许多文件夹名称本质上都是标记，因此您可以将它们自动应用到它们包含的文件中。例如，“characters”文件夹中的所有文件都可以自动标记为字符。这种方法过于慷慨 - 在这个例子中，characters 文件夹中的任何道具都会被错误地自动标记 - 但通常无关紧要。搜索像“字符”这样的大类别中的所有内容并不常见，因此您通常可以依靠标记将结果过滤到可管理的大小。

同样，可能有必要编写工具 - 独立应用程序或在艺术软件中运行的脚本 - 以确保艺术家生成良好的元数据。这些工具通常提供向导式界面，向用户询问有关他们想要创建的内容的一些基本问题，然后代表他们为文件命名和标记。

像 Maya 和 Nuke 这样的重量级应用程序支持的脚本语言可以很容易地实现这一点，但是资产浏览器 - 关于哪个，更多 - 仍然可以帮助使用拖放或本机 OS 文件启动的不太复杂的应用程序的工作流程。将文件路径复制到剪贴板的热键是另一种在即使是最古老的对话框中获得现代搜索技术优势的好方法。

第 5.12 节构建资产浏览器

可以添加到任何管道的最有价值的工具之一是资产浏览器：专用于查找和管理项目中文件的应用程序。与 Windows 资源管理器或 OS X Finder 一样，资产浏览器是列出文件的工具，但与标准 OS 查找器不同，它针对图形制作进行了优化。一个好的资产浏览器有四个关键特性：

搜索

资产浏览器最重要的功能是通过关键字，元数据或标签搜索的组合来查找文件。接口应该针对速度和即时性进行优化，而不是数据库般的精度 - 用户希望快速找到他们的文件，而不是编写 SQL 查询。逐步搜索功能可在用户输入时改进搜索结果，例如 Google 网页界面中的关键字自动完成功能，或在 iTunes 中快速搜索，提供快速反馈，让用户可以前往他们的目的地。

依赖关系

资产浏览器应该了解文件如何相互关联，使用户能够找到所有相关资产，即使它们存储在不同的虚拟位置。例如，它可以允许用户查看与模型相关联的所有纹理，或者与特定合成相关联的所有视频剪辑。

浏览

资产浏览器的“浏览”部分对于不太确定他们需要什么的艺术家最有用 - 例如，设置梳妆台寻找用工作室库中的道具填充环境，或选择视频剪辑的合成器。通过标记或关键字搜索，此类用户可以查看缩略图，文本描述或元数据，以帮助确定要导入其工作的资产。

标记

搜索工作流的本质是资产被“标记”：即，它们以对用户可能重要的所有方式编目。标签告诉项目，“这个资产是一种载体，但它也是可驱动的，军事的和受损的”，或者说，“这种资产是一种道具，但它也是动画，可破坏的，以装饰艺术为主题的。”没有标签的基于搜索的系统实际上只是一个精心设计的系统，用于强制用户记住他们想要的每个文件的名称，这几乎不是一个进步。资产浏览器需要让用户轻松标记资产，并确保整个团队从标记中受益。

第 5.13 节版本和版本控制

到目前为止，我们已经了解了管道开发人员如何确保文件正确命名，放置在正确的目录中并进行搜索。但是，数据管理还有其他重要方面，包括跟踪对文件所做的更改，以及确保只向用户显示正确的版本。这是版本控制和版本控制的用武之地。

几乎每个应用程序都允许用户将他们的工作保存到文件中。但有些还提供了增量保存选项，这意味着程序会自动在文件末尾添加版本号，并在每次用户保存文件时将其递增。这称为“版本控制”，是最简单的版本管理形式。即使您的软件不支持增量保存，也可以执行此操作，只需在表示版本号的文件名中添加一个字符串并手动更新即可。

另一种方法是使用专用的“版本控制”软件（有时也称为“源控制”或“修订控制”软件）。在版本控制系统中，在艺术家可以访问资产之前，必须“检出”它。这将在用户的计算机上创建主文件的本地副本。完成编辑本地副本后，必须在更新主副本之前将其“签入”系统。

有许多商业产品可以做到这一点，包括 Perforce，Subversion 或 Mercurial 等通用版本控制软件，以及专门针对图形工作的资产管理工具，如 Shotgun 和 TACTIC。使用此类工具或具有自己的专有资产管理系统的设施通常会使用资产管理器的签入和签出对话框覆盖其艺术软件中的打开和保存对话框。

这两种方法都有其优点和缺点。版本控制的关键优势在于它的简单性。您不需要特殊工具来实现版本控制系统，也无需培训艺术家正确使用它。

但是版本控制系统，特别是手动系统，容易出现人为错误。如果文件名只对文件进行了少量更改，艺术家可能不会费心增加文件名，或者他们可能只是忘记这样做。即使艺术软件内置了

增量保存选项，目录结构中也可以使用所有增量保存，从而可以无意中移动或删除文件的历史记录。两个艺术家可能同时开始处理同一个文件，一个保存覆盖另一个，而版本控制系统可以防止这个 - 或者至少通知第二个艺术家该文件已经被检出。更新历史记录 - 谁对哪个文件进行了哪些更改以及何时进行了集中存储，从而难以在出现问题时进行故障排除。 Manne Ohrstrom 高级软件工程师，Shotgun 软件 Ryan Mayeda 产品制作人，Shotgun 软件 Rob Blau 管道工程主管，Shotgun 软件

相比之下，版本控制系统使用起来更复杂，并且在软件许可或员工培训方面可能带来财务成本，但提供更多功能。例如，可以编写挂钩到版本控制软件的工具，以便在文件签入或退出系统之前验证文件，从而更容易防止常见问题，如非流形几何（无法展开的几何形状）单个平板，导致某些建模操作失败）或从里面翻出的多边形。

即使没有自定义工具，也会在为每个文件存储的历史记录中自动捕获大量相关信息。该文件的每个版本都将与特定日期和用户相关联，并将带有该用户的一组注释，说明更改的内容。

这些记录（通常称为“更改列表”或“修订版”）可以按日期，用户或文件名进行搜索，这样可以很容易地找到问题的根源。比方说，例如，每日评论显示角色的最新版本没有头脑。快速检查文件历史记录显示其中一位艺术家已编辑角色的骨架以添加面部动画的骨骼。从更改列表中打开字符文件会显示艺术家在此更新期间意外断开了头部。由于变更清单记录了罪魁祸首的身份，因此将角色送回头部重新附着手术是一件简单的事情。 Tim Green 高级程序员，超大型游戏

当然，困扰大型现代制作的问题往往更加微妙和难以调试。大多数情况下，我们在屏幕上看到的资产不是单个文件的产品。在电影制作中，可能存在许多给定角色的不同版本，以不同方式操纵以适应各个场景或镜头的需要。此外，每个变体可以包括与其他角色共享的大量元素：纹理，道具，头发或布料模拟 ，重用动画或照明装备等。同样，在游戏中，每个角色实际上都是模型，着色器，动画甚至游戏代码或 AI 脚本的复杂网络。

这些文件之间的关系通常被称为“依赖关系”，并为管道带来了许多严峻的挑战。因为依赖文件并不总是在服务器上共存，所以它们可能使设计逻辑的，分段良好的文件层次结构变得更加困难。艺术家在打开文件时会因为缺少依赖性而面对错误消息而感到恼火 - 或者更糟糕的是，因为依赖文件已过时而感到困惑。艺术家也难以预测其变化的后果，因为轻微调整可能会在其他地方产生重大影响：例如，调整椅子的比例以使其在一个场景中看起来更好可能会破坏另一个场景中的动画，其中 a 人物居然坐在里面。

出于这些原因，严重的管道需要为用户提供良好的取证工具，以便理解和跟踪文件之间的依赖关系。不幸的是，这些工具通常不会成为版本控制软件的标准，很大程度上是因为存在一系列令人眼花缭乱的可能关系，所以它们必须由管道团队编码。

第 5.14 节良好的版本控制策略

版本控制有两个面。对于个别艺术家来说，它提供了一个安全网：无论你弄乱特定文件多么糟糕，你总能回到最后一个版本。对于团队来说，这是一种保持项目资源最新的方式 - 您只需要同步到服务器，以确保您拥有其他人工作的最新版本。

但如果误用，版本控制可以很快打开你。如果团队成员没有遵守纪律，确保他们办理登机手续的资产有效，那么每天早上都会成为一个令人紧张的过山车，因为你等着找出这次破坏了什么。一旦共享资产定期变得丑陋，更多人将开始避免使用版本控制系统。

这使员工保持工作 - 但现在他们在与生产流程脱节的小型私人宇宙中工作。他们根据已经发生变化的资产制定审美决策，并根据不再存在的特征做出技术决策。当他们提交自己的工作时，这会导致更多的破损，整个生产逐渐陷入不信任和相互指责的恶性循环。艺术家不相信管道，工

具团队厌倦了追捕由过时文件引起的错误，没有人做任何事情。

由于这是一个社会问题，解决方案也是社会问题。每个团队都需要创造一种文化，让人们认真对待他们对队友的责任，并采取一切必要措施，确保他们不会因为仓促或粗心而破坏生产。Tim Green 高级程序员，超大型游戏。

对于艺术家来说，这些责任很简单。确保您拥有其他人的最新版本，并使用最新版本的工具。在检入之前，目视验证您的工作是否良好，并确保将所有依赖项添加到系统中。

对于技术艺术家和程序员来说，负责任更复杂。良好的纪律仍然是关键，但拥有一个可靠的构建过程以确保团队其他成员使用的工具可靠且始终可用也很重要。当你为数百名艺术家提供支持时，捕获 bug 的成本很快就会迅速增加 - 对于 3 ds Max 和 Maya 插件这样的工具来说更是如此，这些工具可能会在原始问题修复后很长时间内将错误数据留在艺术文件中。错误是不可避免的，但是一个积极的测试程序 - 与真实的现场测试人员一起完成，无论是质量保证人员还是来自艺术人员的志愿者 - 在缓解痛苦方面走了很长的路。正式的构建和测试程序确实增加了处理功能请求和次要修复所需的时间，但工具质量和可靠性方面的好处 - 因此，工具团队和内容团队之间的信任 - 足以使起来。

你进入一个项目越深入 - 在游戏中，你越接近一个演示或里程碑 - 打破构建的后果就越激烈。在这种情况下，程序员经常使用“代码伙伴”在办理登机手续之前审查他们的工作。代码伙伴并不总是从字面上检查彼此的工作：通常，通过你所做的改变来讨论你的伙伴的过程足以让你意识到你做错了什么。艺术家可以做同样的事情。在早期阶段，抓住某人并审查你所做的每一点改变都不一定是实际的，但是当你获得成功时，同行评审变得至关重要。 Matt Hoesterey 微软设计主管。

监管您的管道

在某些项目中，仅仅依靠艺术家负责任的行为可能还不够：可能有必要强制其良好行为。当我从事 MMO（大型多人在线游戏）工作时，其树（文件夹和资产的层次结构）需要保持清洁长达十年之久，我们的一位技术艺术家将担任我们的“管道警察”。基本上，这是他们的全职工作，需要他们浏览每个变更列表，以确保正确命名所有文件和文件夹。即使在紧要关头，他们可以在签入之前批准每个文件，他们将在以后检查更改列表，如果他们发现问题，则无论工作量有多大，都需要您在重新提交文件之前签出文件并解决问题。是的。如果遗漏了变更列表问题，则甚至在几个月后，您还必须在最终发现变更时进行更改。
显然，如此繁重的举动并不适合所有项目-在某些情况下可能适得其反-但在 MMO 上，或多或少要求维持服务的长期健康。

虽然专业礼貌是良好版本控制的核心，但管道本身也可以帮助人们成为好公民。版本控制规则较差的团队的一个常见问题是覆盖：艺术家 A 长时间处理文件的本地副本，但不通过检查来声明它的“所有权”。在此期间，艺术家 B 检出文件，进行一些小改动，并将其重新检入。当艺术家 A 完成时，他检出文件，保存他的本地副本并将其重新检入以破坏艺术家 B 的工作。使用 Max 或 Maya 中的脚本可以最小化此问题，每次打开时都会自动获取文件的最新版本，并在出现版本不匹配时向您发出警告。类似地，系统可以警告想要处理当前正被其他人使用的文件的用户。

另一个常见问题是依赖关系：艺术家经常忘记检查使新资产工作所需的所有文件。在这里，版本控制软件可以在每次提交新文件时运行检查，以查看它是否需要系统中尚未存在的任何文件。

最后，工具可以设计为在整个网络中自动更新 - 尽管这给开发团队带来了沉重的负担：确

保所有工具始终正常工作。

第 5.15 节资产审查和批准

当艺术家更新文件时，他们不会随意进行更改。相反，他们正在响应客户或其主管的变更请求（通常称为“注释”）。另一个关键的数据管理任务是确保有效地传达这些注释，并在逐步完成时跟踪每个文件的状态。我们将首先看一下这个过程，因为它适用于电影作品，然后适用于游戏。

在电影中，资产通过循环工作流程得到完善：进行更改，查看更改，然后定义所需的下一组更改。评估每天进行（如果项目允许，有时每周进行一次），这样任何任务都不会在错误的方向上耗尽太长时间。

评论可以通过多种方式进行，从部门负责人或领导看着他们坐在工作站的艺术家的肩膀，到在专用剧院或回放套件中举行的正式“日报”会议，有时在客户。这些通常更有条理，一次审查整个序列或整个部门的工作。

第一种类型的审查通常在主机应用程序内完成。在剧院或播放套件中，更常见的是审查艺术家一直在研究的资产所产生的一系列高分辨率图像，特别是对于色彩关键的工作。

在这两种情况下，都会有人记录所请求的更改。其中一些可能与名义上正在审查其工作的部门无关。例如，在动画评论中，客户可以对照明进行评论。此反馈也必须记录并转发给照明部门。因此，注释可以与资产管理系统中的单个文件或文件组相关，这样艺术家只能收到与他们实际工作的任务相关的反馈。

确保笔记使用标准术语也很重要。例如，在谈论角色时，通常使用术语“左”和“右”来指代角色的左和右，而不是屏幕的左侧和右侧。

如果反馈来自客户端，那么在对消费进行消毒时，必须“分阶段”处理这些消息并不罕见。有时，反馈不是免费的 – 因此您可能不希望将其未经编辑地传递给每周七天，每天工作十六小时的艺术家。必须将此暂存过程内置到注释工作流程和工具中。

这些工具因项目和工厂而异。在某些情况下，笔记可能是基于文本的;在其他人中，他们可以使用虚拟笔工具在各个框架上以数字方式书写或绘制。但是，在这两种情况下，所有信息 – 已审核的内容，审核的内容，提供的反馈以及是否已进行更改 – 都应在某种中央电子表格或数据库中进行跟踪。

当它达到某个状态时，工作将被批准。这个州因部门而异。通常，工作在获得批准之前不一定是最终质量：它可能只是给予下一个部门足够好，以便那里的艺术家可以开始研究它。例如，可以在阻挡阶段批准动画，以便照明部门可以在动画师继续完善其工作的同时开始点亮序列。

正如我们在前面的章节中所提到的，在电影中，许多设施从未将资产描述为“最终”。他们认为，如果时间允许，他们仍然可以进一步调整它们直到交付点，相反，资产应该始终被视为CBB（可能更好）。

在游戏中，构成“最终”的问题更为严重。现代游戏涉及令人眼花缭乱的资产数量，几乎没有人会直接知道它们是什么，它们是什么样的，以及它们是否已准备好被公众看到。从事资产工作的艺术家和负责跟踪制作的管理人员可能会以完全不同的方式使用语言，从而加剧了这个问题。

例如，经理可能认为某些事情已经“完成”，因为它已按计划进行了检查，而艺术人员可能会认为随着项目的进展可以进一步完善。这给资产管理系统的开发人员带来了问题，因为计算机很难准确地记录这种模糊性。

也就是说，使用一套严格的工具直接处理如此庞大的数据密集型任务通常比尝试通过电子

邮件和 Excel 电子表格管理它更好。如果生产具有良好的标记系统，则资产状态实际上只是一种稍微特殊的元数据形式。但是，更多的演进工具可以使管理人员更有效地传达他们的愿望：例如，资产浏览器可以允许艺术线索或艺术总监在查看资产时将评论（同样，一种形式的元数据）附加到资产。如果需要更多反馈，下一位处理该文件的艺术家将会提供这些评论并知道联系谁。

将状态作为资产元数据的一部分处理 - 而不是作为生产列表中的特殊类别 - 还可以更轻松地利用依赖性分析工具来掌控生产。查看资产以查看其中有多少依赖项被视为“最终”是评估相对风险的简单方法。如果在生产的后期时间紧迫，最好削减几乎没有最终依赖性的资产，而不是那些已经完善道具和动画的资产。

然而，这种信息存在真正的危险，导致错误的信心。数据库可以记录文件“Dr_ Atomic.mb”已被标记为“最终”的事实 - 但是它不太可能知道艺术总监一直都讨厌那个特定的超级恶棍而且他能够做到。任何计算机都无法与消息灵通的人类对项目状态的理解相媲美。自动跟踪可以帮助生产者和销售人员，而不是替代他们。

在游戏中，还有另一种必须记录的状态信息：错误。资产可以是“最终的”，因为它的外观和功能已经被固定，但它可能仍然存在技术问题：例如，模型的一部分纹理映射不良。在理想的世界中，使您能够在浏览时查看资产状态的相同工具还可以让您查看与其相关的任何错误。

第 5.16 节跟踪生产数据

在上一节中，我们讨论了生产数据如何帮助生产者和监督者及时了解生产进度。总结本章，我们将更详细地探讨数据跟踪的概念。

有两种生产数据值得追踪。第一种是帮助回答重要生产决策的那种。艺术家在手动记录保存方面很糟糕 - 这就是为什么 Perforce 仓库中的一半签到注释是“更新模型”或“修复错误”的原因 - 如果收集这些数据需要手动操作，您需要证明这一努力是正确的给负责的工作人员。一般而言，您应事先知道信息将回答的问题。这些通常是这样的：

谁正在拍摄这张照片，以及他们需要多长时间才能完成任务？

目前正在审核什么，以及自上次以来发生了哪些变化？

是否已解决上次审核中的注释？

这个部门处理的任务有多接近完成？

您还应该知道这些答案的显示位置。如果这些信息对艺术家有用，请将其显示在他们花费大部分时间的应用程序中。如果这些信息对于花时间四处奔波的工作人员有用，请确保它可以打印出来，或者在平板电脑上提供。通常，当您捕获的数据多于可以轻松显示在监视器上的数据时，制作人员会在中央位置专门设置墙壁以固定信息，例如拍摄卡片，以便每个人都可以轻松查看项目的状态。

尽可能简化收集信息。如果您希望艺术家提供有关资产修订期间更改内容的注释，请不要让他们切换应用程序以输入信息。此外，确保所有需要查看这些笔记的人都能看到这些笔记。换句话说：让艺术家轻松做正确的事情，并在他们不做的时候让它变得明显。

第二种值得跟踪的数据是您可以自动收集的数据。如果您拥有托管数据仓库的基础架构，则应尽可能从管道中捕获尽可能多的数据。渲染所需的时间或给定脚本运行的次数可能看起来无关紧要，但随着时间的推移，可以挖掘信息来回答您从未预料到的问题。记录运行脚本的位置以及软件的版本，升级到核心应用程序后，您将能够找到您错过的任何计算机。记录项目上的渲染小时总数，您将能够决定是否需要为下一个渲染场升级渲染场。

通常，您会发现您已经在捕获相关信息：您只需要将其转换为可以利用的格式。例如，您可以通过 Perforce 搜索来查找生产中每个文件的大小。通过将此与元数据交叉引用，您可以运行查询以提取更多有用的信息。

在跟踪外包工作室正在做什么时，这可能很有用。例如，他们是否会向您收取费用以创建详细的 3D 资产，而这些资产只会在镜头背景中看到 – 也许在 2D 纹理就足够的情况下？通过保持按三角形计数和物理量排序的模型的运行选项卡，您将能够快速发现差异。同样，关键是要使这些信息易于收集。如果您必须要求艺术家手动输入数据，则无法完成。但是，如果在工具中包含特定于供应商的修补程序，用于标记他们使用供应商 ID 创建的资产，则会自动为您捕获信息。

第 6.1 节您将从本章中学到什么

生产管道只能与构建它的系统基础架构（IT）一样强大。在本章中，我们将探讨这些硬件和软件如何影响管道的开发和管理。我们将首先了解电影和游戏工作所需的各种基础设施，然后检查与任何管道相关的关键问题，包括管理操作系统和实用程序软件的需求，以及管道安全性的需求。

第 6.2 节电影的 IT：硬件类型

电影制作对数据有着极大的兴趣。凭借其高度详细的模型，复杂的动画装备，模拟缓存和高分辨率渲染，单个电影可以生成许多 TB 的数据，而运行多个项目的工具可以很容易地发现自己在 PB 级范围内。在这种极端环境中，坚固的硬件基础设施至关重要。

这个硬件究竟是什么取决于公司的组织方式。公司可以由单一设施组成，或者 – 因为在海外收购前竞争者或开设卫星办事处变得越来越普遍 – 通过专用网络或互联网连接的多个设施。

任何这些方案至少需要以下类型的硬件：

工作站（台式机和笔记本电脑）

网络（电缆，交换机和 Wi-Fi 连接）

核心服务器（用于电子邮件，用户帐户，DNS，数据库等）

存储集群（用于数据存储和备份）

渲染农场（用于处理计算密集型任务）

电话和通信硬件

备用电源（通常不足以为整个设施供电，特别是渲染服务器场，但足以使核心服务保持活动状态或正常关闭电源）

在任何这些情况下，以下类型的硬件都是可选的，但是接近通用：

额外的监视器（艺术家）

图形平板电脑（艺术家）

耳机（用于动画师进行唇形同步工作）

剧照相机，摄像机和麦克风（用于录制参考资料：这些通常可以在几位艺术家之间共享）

具有多个设施的方案需要以下其他类型的基础架构：

专用网络连接或专用路由设置（以确保站点之间有足够的带宽）

每个设施之间的某种形式的加速数据传输

视频会议设备（理想情况下，配有远程屏幕共享系统）

其中两个主要成本中心是存储集群和渲染农场，特别是涉及“资本支出”（资本支出）时。 我们将更详细地研究每一个。

第 6.3 节电影的 IT：存储集群

存储群集是保存所有生产数据的文件服务器群集。 为了确保有足够的数据可用，大多数工作室将其存储分解为多个层：

第 1 层：可供生产使用的存储

第 2 层或“近线”存储：无法从生产机器访问但可用作“停车场”的磁盘，以便更快地从第 1 层存储中获取数据

基于磁带的存储：用于生产数据的长期归档，最有可能是最终输出到客户端

虽然经常被忽视，但设施的工作站和本地渲染机也是存储空间的来源。本地计算机上可用的总空间通常大于托管存储中可用的空间,这意味着在规划存储要求时应考虑本地缓存策略和代理生成。

第 1 层和第 2 层存储包括来自 EMC Isilon 或 NetApp 等供应商的 NAS（网络连接存储）设备。在第 1 层存储中使用的那些更昂贵但更快，针对读取性能进行了优化;在第 2 层中使用的那些更便宜但更慢。

在规划托管存储时，重要的是要考虑许多问题：

渲染服务器场将如何与存储进行交互

如何跟踪生成的数据量以及上次访问数据的时间

如何向艺术家呈现存储：作为多个磁盘或单个安装点

哪些数据对生产至关重要

需要归档哪些数据以及需要多长时间

如何监控存储

最大的问题之一是渲染农场与存储交互的方式。在 VFX 生产中工作的每个人都会遇到一种情况，即单个渲染器已经使整个设施瘫痪。在大多数情况下，这是由于存储在单个磁盘上的纹理文件导致数千台计算机试图通过网络访问。为了避免这种情况，渲染服务器场可以连接到 SSD 缓存系统，例如由 Avere 生成的系统，或者为文件 I / O（输入/输出）优化的其他专用系统。但是，这样的系统价格很高。每位 CTO 都会听到诸如“如何存储如此昂贵？”这样的评论？我只花 100 美元买了一台 2TB 外置硬盘！当等效的 2TB 高性能系统成本高达数千美元时。

集群的采购通常是递增的，每年或每次主要生产都会增加额外的产能。在较大的设施中，这是由专门的数据操作部门处理，与 IT，研发和生产联系。在较小的设施中，它由 IT 处理。

设施的备份和归档功能也必须与存储群集一起增长。这里,存储器由 LTO(Linear Tape-Open）半英寸磁带盒组成。磁带采用条形码编码，以将它们与特定项目相关联，然后存档在“库”中。这些通常是由 HP 和 IBM 等供应商生产的自动化系统，包括一个机器人，可在需要时自动将磁带加载到磁带机中。

备份过程由诸如 CommVault 等商业供应商或 Bacula 等开源替代品生成的软件控制。备份按照工作室可以管理的频率逐步增加：通常至少每天，但不少于每周。

一旦项目完成，存档通常是一次性操作 - 或者对于大型项目，一旦完成一次或序列完成 - 并且可以涉及将数据的副本发送给客户端。通常会保留两份磁带，一份在现场，一份在场外。这两个过程都有很高的“运营成本”（“运营支出”或“运营费用”的缩写）。此时的“快照”，即只读的数据副本，也将用于辅助数据管理。

经常被忽视的另一个步骤是恢复过程：确保您彻底了解该过程，并在需要使用之前对其进行测试。由于数据恢复管理不善导致的停机时间不是一种选择，因此未经测试的备份系统根本不

是备份系统。

一旦到位，存储群集将需要持续监控，以确保有足够的可用磁盘空间可用于生产。常见的数据管理策略包括在一段时间后将文件从快速存储移动到慢速存储，或者仅在线保留一定数量的先前版本的资产。一些工作室甚至设置了自动数据剔除程序，要么删除在给定时间段内未访问过的文件，要么根据文件路径（例如，Playblasts 或临时渲染数据很容易从其他文件，无需长时间在线存储）。在资产管理系统的范围之外，艺术家和制作人员有责任管理他们自己的本地数据足迹，删除他们不需要的硬盘驱动器文件。

需要考虑的另一个问题是 3D 数据往往具有相当短的半衰期。主要商业工具的开发人员每 12 到 18 个月发布一次他们软件的新版本，专业工具和插件来去匆匆。绝对不能确定使用最新版本的软件可以可靠地打开四年或五年的文件。

这对于需要维持正在进行的 IP 的核心资产的工作室来说是一个严重的问题，例如可能产生续集的电影。即使不再需要旧资产进行制作，它们也是制作最新电影的艺术家的重要参考资料。

一些工作室以稳定的文件格式（例如 OBJ 或 FBX）存档关键资产，这些资产与特定版本的第三方软件的关系较少。已知其他人将存储单元中的当前 OS，软件和内容存储在工作计算机中。当然，这是解决高科技问题的一种令人尴尬的低技术解决方案，但实用主义是参与生产管道的任何人的关键优势。虚拟机映像是实现相同结果的另一种方法，无需保留可能不可靠的硬件。

管道团队应记录存储设施的历史指标。在规划下一次生产需要多少存储空间时，了解每个用户，部门，镜头或项目生成了多少数据是有用的信息。在强制方法中，专用进程将每天搜索文件系统，查询每个文件以获取上次访问它的信息，其所有者，其大小和位置等信息。这可以与来自资产管理系统的数据（例如，拍摄文件所属的数据）相结合，并通过公司网页上的“仪表板”界面提供,高级员工可以通过该界面执行自定义分析。许多第三方软件供应商专门从事此类任务：例如，Zenoss 提供用于监控存储使用情况的系统。

"有几个很好的理由这样做：″

数据有成本

计算每千兆字节的存储数据，包括购买，供电和维护服务器的成本，以及支付数据管理员来管理流程，使工厂能够更准确地预算未来的工作。

数据占用空间

计算每个镜头，每个资产或每个艺术家所需的平均存储量使设施能够预测未来项目所需的存储量。

数据使用模式随时间变化

随着工作流程的发展，“典型”项目生成的数据量也会发生变化。有时，因果关系可能并不明显：例如，新工具可能会导致磁盘使用意外突然出现。将这些变化追溯到其来源使管道工作人员能够最大限度地减少问题并更准确地预测未来的数据使

请记住：在视觉效果工作中，没有“太多存储”这样的东西。这是一个单个流体模拟可以生成数 TB 数据的领域。 Fran Zandonella 国际高级管道 TD，节奏和色调识别具有大量 I / O

要求和隔离的部门识别和管理磁盘使用的高峰时间将不同类型的数据分配给适当类型的存储开发处理常用文件的策略遵循最佳创建脚本时的实践使用监视和调试工具

第 6.4 节电影信息技术：渲染农场

渲染农场”或“渲染墙”用于需要大量计算资源的任务。在完成这些工作站的时候，不是将艺术家的工作站绑起来，而是单独处理它们更有效。尽管它的名称，渲染农场不仅处理渲染作业：它是一个高性能计算集群，所有离线处理都发送到该集群，包括模拟（用于 FX），缓存（用于动画），代理（用于合成），以及与服务相关的任务，例如创建 QuickTimes 以交付给客户。因此，通常最好根据部门或任务类型等标准划分服务器场。这允许服务器场调度程序公平地为所有用户分配资源，并在最符合其要求的计算机上执行每个作业。

构成渲染农场的机器通常是“无头”的 - 也就是说，缺少监视器和输入设备 - 并且历史上是基于 CPU 的。然而，如果没有某种板载显卡，很难买到任何东西，近年来 GPU 上可以处理的任务数量增加，这意味着许多现代农场也包含了大量的机器。图形处理器。对于特殊任务 - 例如，可能需要其唯一任务是处理动作捕获数据专用机器的服务器，并且将需要特定配置。如果没有使用艺术家的工作站，IT 部门也可能将它们作为渲染农场中的“奴隶”，使用原本闲置的处理能力。

与存储集群一样，渲染服务器场的采购通常是递增的，每年或每个主要生产都会增加额外的容量。在较大的设施中，这是由专门的渲染操作部门处理，与 IT，研发和生产联系。在较小的设施中，它由 IT 处理。

在中型和大型工作室中，服务器场可能会运行数百或数千个 CPU。这意味着电源和冷却也是主要考虑因素。为了说明原因，让我们看一下运行 80 服务器场所需的内容：按现代标准规模适中。假设两台服务器可以装入 1U（一个“机架单元”：一个标准的测量单位，相当于标准服务器机架中的 1.75 英寸高度），每台服务器包含两个 8 核 Intel Xeon CPU 和 RAM ，本地系统驱动器等。这意味着 80 个服务器将完全适合 40 U 机架（标准机架为 42 U 至 44 U），并将包含 1,280 个 CPU 核心（80 个服务器 x 每个服务器 2 个 CPU x 每个 CPU 8 个核心）。

服务器已经相当沉重，但增加了电源管理，以太网电缆，机架导轨，螺母和螺栓，每个 900 mm x 600 mm 机架可能重 600 kg，可能更多：可能超过办公室推荐的重量限制。

两个 Xeon 处理器，RAM，硬盘驱动器和服务器的其他组件的总功耗可以轻松为 250 W 至 500 W，因此整个 40 U 机架将消耗 20 kW 的功率。在空调和空气处理方面增加 50％，总功率消耗为 30 kW。

这通常意味着限制渲染农场性能的主要因素不是通常假设的处理器功率，而是提供电力和冷却以保持农场运行的需要。为了保持运行性能，必须将农场保持在较窄的温　　度和湿度范围内。出于这个原因，数据中心的设计本身正在成为一种艺术形式，涉及建筑，设计，结构，机械和电气工程的各个方面。

渲染服务器场通常由单个软件管理 - “调度程序”或“渲染管理器” - 负责分发任务。生产中常用的包括截止日期，拖拉机和阿森纳等开源工具。

调度程序管理的任务可以像渲染单个高分辨率图像一样简单，也可以像在不同软件包中执行的一系列操作一样复杂。调度程序收集每个艺术家提交的任务，并根据算法将它们分配给特定服务器。该调度算法可以使用几种不同的方法，从简单的“先到先得”任务分配，到基于优先级或权重的调度，再到基于先前渲染时间，当前文件管理器或网络性能的纯度量驱动方法，以及生产计划。为此，它还必须跟踪哪些服务器当前空闲，哪些服务器正忙或离线。然后，调度程序将任务的进度报告给艺术家，如果任务遇到问题，则向他们发送错误消息。 Hannes Ricklefs MPC 管道全球主管。

渲染场由渲染操作部门管理，以确保其有效运行。这通常意味着监控农场工作的进度和分类错误：鼓励艺术家自己解决普通问题，并将更大的问题引向技术支持。技术人员（绰号“渲染争霸者”）也将管理工作的优先级，以确保关键任务按时完成。接近截止日期的镜头或项目将优先于其他镜头或项目，但每个艺术家和部门都有其公平的资源份额也很重要。与存储集群的数据争夺一样，渲染争论被视为入门级职位，也是更高级技术角色的途径。

与存储群集一样，保留渲染场的历史度量标准非常重要，原因如下：

渲染有成本

计算每小时一美元的处理数量，包括购买，供电，冷却和维护服务器的成本，以及支付渲染调换器来管理流程，使设施能够更准确地预算未来的工作。

渲染农场是一种有限的资源

计算每个镜头，每个资产或每个艺术家的平均渲染时间使设施能够预测未来项目所需的农场容量。

加工用途随时间变化的模式

随着工作流程的发展，“典型”项目使用渲染农场的方式也会发生变化。将这些变化追溯到其来源使管道工作人员能够最大限度地减少问题并更准确地预测未来的数据使。

第 6.5 节电影的 IT：管理基础结构

管理设施的基础架构有两个主要方面：配置基础架构和监视该基础架构的状态。

配置基础架构时，能够快速，轻松地部署新的计算资源非常重要。这意味着最好为艺术家的工作站定义标准设置，并坚持下去。有多种工具可以帮助系统管理员自动化构建和配置新机器的过程：最着名的配置管理（CM）工具包括 CFEngine，Chef 和 Puppet。拥有现成的磁盘映像也很重要，可以通过单次安装创建完全指定的工作站。最后，要随时备有备件和驱动程序：在大型工作室中不断运行硬件故障，您需要防止艺术家被坏硬盘驱动器，贫血电源或死机 GPU 所困。

Fran Zandonella 国际高级管道 TD，Rhythm&Hues

需要可重复的系统设置

设施是拥有五台机器还是五千台机器，具有可靠的基准系统映像（可用来构建新系统），因此可以在项目开始时更轻松地推出机器，或更换有缺陷的机器。可能需要几个单独的磁盘映像（例如，对于台式机和渲染服务器，或针对不同部门的单独设置），但为简单起见，请尝试将磁盘映像的数量降至最低。在整个机构的不同计算机上可能还需要安装同一软件的多个版本。例如，Side Effects 软件有时会在一天之内发布 Houdini 的两个新版本，通常每周都会发布多个版本。也许一个部门需要特定的错误修复，而其他部门则不需要。能够对应用程序进行版本控制有助于管理这些需求。考虑本地化不经常更改的应用程序，如果本地副本存在问题，则提供对网络版本的备用。在部署之前，需要根据每个部门的需求以及当前和计划中的渲染机样本对基准映像进行测试。自动化测试可降低成本并确保部署之间的一致性。重要的是要有一个过程来检查各个机器是否符合基准，特别是在一个设施有员工在不同办公室工作的地方。如果世界各地的艺术家收到不正确的更新，则他们的作品可能会比原定进度落后一天。

在为设施构建基础架构时，需要考虑的一个重要事项是在每台计算机上安装软件的方式。最常用的两种方法是将软件部署到每台计算机的本地磁盘，或者运行安装了每个应用程序的共享软件服务器。

共享软件服务器意味着只需要在各台计算机上安装基本操作系统组件：其他软件将通过服务器提供。这也意味着一旦安装了新的软件，它就会立即供每台机器使用。

然而，这样的系统需要适当的基础设施：同时为用户的软件请求提供相当大的机器，以及稳固的网络基础设施 - 特别是在网络负载较重的关键时期。它还会产生单点故障：如果软件服务器出现故障，整个系统将失败。解决这个问题的方法是在系统中构建“冗余”（多个服务器能够满足相同的需求，因此如果一个失败，另一个可以接管）。

在每台机器上安装所有软件可以消除对网络基础架构的这些需求。但是，安装和配置软件可能非常耗时。有时在生产的热量中，需要安装快速修复程序以在一夜之间完成渲染。如果渲染服务器场包含数千台计算机，则可能无法以足够快的速度完成安装以按时完成作业。

一些工作室使用混合方法，其中不快速变化的软件被部署到每个单独的机器，而经常变化的软件通过共享网络服务器可用。

管理设施基础设施的第二个主要方面是监控该基础设施的状态。 这包括：

网络带宽利用率

网络流量

单个机器上的 CPU 负载

各台机器的内存使用情况

特定于应用程序的统计信息，例如内存占用量，使用的 CPU 资源或数据库性能如何随所服务的查询数量而变化

一旦达到某个临界阈值，监控解决方案应该能够发出警报：例如，如果磁盘容量达到 75%，或者数据库服务器上的负载超过某个阈值。警报会在潜在问题影响任何用户之前通知系统管理员。监控解决方案也应该远程工作：系统运行缓慢并不总是很清楚，因此能够在不必登录每台机器的情况下检查机器至关重要。

“慢”的许多面孔

在某个时候，所有在管道或系统管理中工作的人都会接到用户打来的电话，说他们的计算机“运行缓慢”。由于存在如此之多的可能原因，因此，要找出问题的根源，就必须进行认真且耐心的调查。

可能的本地原因包括：

•由于运行多个模拟，机器由于负载重而运行缓慢。

•机器正在交换内存，因为诸如渲染之类的单个进程正在消耗所有可用内存。或原因可能在于用户的机器之外：

•整夜将计算机添加到渲染场，并且渲染尚未完成。

•用户正在处理文件，该文件存储在服务器上，负载很大，无法足够快地检索到该文件。

•用户的应用程序正在从锁定了表且函数调用将其锁定的数据库中请求一条信息。

要问的第一个问题是，缓慢性是否仅影响用户或周围的其他人。在第一种情况下，原因很可能是用户的机器。

然后，重要的是要找出运行速度是否仅影响单个应用程序或每个应用程序。在第二种情况下，原因很可能是中心原因。在这种情况下，重要的是快速了解整个设施系统的状态。有多种工具可以执行此操作，包括 Cacti，Nagios，OpenNMS 和 Zabbix

第 6.6 节游戏的 IT：构建农场

游戏的数据需求很少像电影那样庞大，因此它们通常需要不那么令人印象深刻的基础设施。游戏工作室通常会拥有您在任何其他办公环境中所期望的资源：电子邮件，非关键数据的共享驱动器，Wiki 或其他集中文档等等。

但是，找到某种集中式“构建框架”或“服务器场”的情况并不少见，这种集中式“构建框架”或“服务器场”用于执行开发人员个人计算机上执行速度太慢的所有任务。此类任务取决于游戏引擎，但通常包括渲染光照贴图或生成光探测器。该服务器场还经常用于单元测试（测试游戏引擎源代码的各个单元），连续构建（将这些单独的单元合并到代码的单个“主线”），以及烟雾和浸泡测试（测试关键特性是否有效）当构建首次运行时，它在持续负载下不会失败）。没有两个游戏团队或设施将执行完全相同的构建农场任务。

构建服务器场可以由高端机器（有或没有 GPU，取决于他们期望的任务）或现在太慢而无法提供给开发人员的旧工作站组成，但可用于增加“农场的节点数。

管理服务器场的软件与其包含的硬件一样独特。有许多商业解决方案可供使用 - 本章前面列出了一些 - 但是工作室开发自己的管理软件并不罕见。即使是简单的耕作任务的方法也可以在短时间内实现，特别是如果服务器场中的每台机器都配置为特定任务。 Tim Green 高级程序员，超大型游戏。

小任务很快就会积累起来

在管理游戏项目时，不要只关注大任务。一次又一次地重复，小任务很快就会累积起来。《哈利波特与火焰杯》的游戏已经开发出来了使用 renderware studio：一个级别的创作工具，允许用户放置对象并创建以事件为中心的游戏逻辑。在 renderware 中，一个级别由成千上万个对象组成，每个对象都有一组属性和到其他对象的链接。每个对象都保存到一个 xml 文件中，每一级都会产生成千上万个非常小的 xml 文件。

当我在为冠军而努力的时候，球队的一个普遍持久是水平加载很慢，通常需要 10 分钟我刚加入这个团队，所以我同步了 level XML 文件，但发现在我的机器上，所有东西都在 30 秒内加载完成在排除了其他原因后，我重新替换了，这纯粹是由于磁盘访问时间。我注意到，在速度较慢的机器上，硬盘驱动器碎片严重，所以我们对这些驱动器进行了碎片整理并重新测试，希望能立即提高速度，但结果是一样的。怎么回事？在这一点上，我需要解释一下 NTFS，PCs 上使用的文件系统。NTFS 将每个文件的元数据存储在名为主文件表（MFT）结构的中元型态数据类似于文件名创建[33]状语从句：日期，甚至在文件中存储部分数据。此数据的固定块大小为 1KB。这意味着文件的数据通常在硬盘上的某个地方，可能分裂成碎片，元数据在 MFT 中，但是如果文件的大小小于 1 kb，则可能在 MFT 中完整存在。碎片 整理会在计算机硬盘上移动文件碎片，并将它们合并到我们数以万计的 XML 文件中，大多数大小都不到 1kb，因此它们只存在于 MFT 中，没有受到影响。连续块以减少加载文件时的查找时间，但它不会移动 mft 中的任何内容。影响当加载一个等级时，RenderWare Studio 将按顺序加载这些 XML 文件，要求文件系统在 MFT 中查找所有的 XML 文件如果只是一次性将文件写入文件系统，那么它们的元数据将按顺序写入 mft ，这意味着

需要进一步的块查找和加载。解决方案是重命名包含文件的文件夹，将其复制回原始名称，然后删除旧文件夹这意味着元数据文件现在在 mft 中很接近，增加了 mft 的缓存块包含所有所需数据的可能性。一个简单的工具之后，一百多人有更好的水平加载时间，节省了约 30 分钟每人每天，或约 50 小时，总的一天。

第 6.7 节游戏信息技术：版本控制

虽然游戏项目可能不需要像视觉效果项目那样多的存储，但 IT 基础架构的其他方面对其成功至关重要。游戏开发者必须应对游戏设计和游戏引擎的不断变化，有时甚至是不可预测的变化。出于这个原因，他们需要能够非常精确地跟踪项目过程中游戏内容的演变：例如，他们可能需要将资产回滚到早期版本以分析特别微妙的错误，或者维护不同游戏控制台的并行版本。因此，它们尤其依赖于版本控制软件。我们在前一章中研究了版本控制软件的工作原理。在这里，我们将更详细地探索特定于游戏的方面 - 特别是那些影响工作室基础设施的方面。

随着文件的更改，这些更改的数据和元数据将存储在文件历史记录中 - 但该文件的旧版本仍然可用。为这些数据维护一套安全且“冗余”的备份是迄今为止 IT 部门最重要的工作。如果更改导致错误，则用户能够恢复到更早，不间断的版本至关重要。

版本控制系统中包含的历史信息对于诊断和修复更微妙的问题也至关重要。由于许多人都在处理每项资产，因此很容易因通信失误或简单的人为错误而导致问题，而这些问题可能难以解决。

版本控制软件的另一个重要优点是它能够创建项目的修改版本，同时保留原始代码和数据的完整性。 “分支”使系统能够维护相同文件的多个并发版本（包括其历史记录和元数据）。分支对于使团队的不同部分能够安全地并行工作非常有用。例如，一小部分动画师和设计师可能想要为游戏玩家角色制作一种新的动作动画原型。这将是一个缓慢的，迭代的过程，产生大量的临时错误。如果团队被分配了自己的分支，它可以在不影响工作室其他部分的情况下工作。一旦新功能可靠地运行，分支就可以重新集成到代码的主线中，以便整个团队可以共享它。

由于版本控制对游戏团队至关重要，因此选择正确的版本控制解决方案是创建游戏管道的最重要步骤之一。市场上有几种版本控制系统，包括商业版和开源版，但并非所有版本控制系统都同样适用于游戏。对于程序员来说，版本控制的当前趋势是针对分布式系统，例如 Git，Subversion 或 Mercurial。不幸的是，这些系统针对基于文本的代码文件进行了优化，并且它们倾向于扼杀组成现代游戏的大量图形文件。配置分布式系统以使用图形文件逐渐成为可能，但这仍然是专家级任务，应谨慎对待。出于这个原因，像 Perforce 和 Alienbrain 这样的集中式版本控制系统在游戏制作中仍然比较常见。

选择系统时的一个重要考虑因素是工具链的规模和复杂性。如果您在自动化方面投入了大量资金，则需要一个为脚本工具和工具程序员提供良好 API 访问权限的系统，例如 Perforce 或 Mercurial。另一方面，如果您坚持使用现成的软件，您可以选择像 Alienbrain 这样的系统，为 Maya 和 Photoshop 等艺术工具提供更加精美的 UI 和现成的插件。

选择版本控制系统的另一个关键因素是分发。如果您的项目涉及多个地点之间的协作，或者与场外承包商和外包商合作，您将需要决定是否允许这些团队访问您的版本控制系统。有些团队绝对拒绝以安全名义对其数据进行异地访问;其他人坚持这样做，以便他们可以使用版本控制功能来跟踪合作伙伴的活动。如果您计划允许外部访问数据存储，则需要通过基于 Web 的客户端或通过 VPN（虚拟专用网络）连接找到允许远程访问的系统。特别是，Perforce 使您能够创建代理服务器，该服务器充当来自项目主版本的数据的本地缓存。如果您希望为卫星位置的用户提供对常用文件的更快访问权限，则这些功能非常有用。

确保您有一个明确的策略来控制对数据的访问也很重要。大多数版本控制系统都包含某种

形式的身份验证，以确保只有授权人员才能查看数据。管理员还可以创建用户配置文件，以确定版本控件的哪些部分对哪些用户可见：例如，允许程序员访问已完成的游戏资产但不是原始 Maya 或 Photoshop 文件，或艺术家访问最新版本的游戏引擎但不是所有的源代码。如果您计划让外包商直接访问您的版本控制系统，则访问控制绝对至关重要。

最后，您需要确保您的版本控制系统具有足够的硬件支持。这应该包括足够的磁盘空间（这些大二进制文件的历史记录加快了），至少是一个称职的处理器。定期备份和冗余架构（如 RAID 驱动器）至关重要：如果您的版本控制系统出现故障，您的项目就会陷入困境。如今，一些团队为其版本控制服务器选择基于云的托管，以利用大数据中心提供的高正常运行时间和全天候维护 - 尽管许多其他团队在考虑允许他们的宝贵项目时感到不寒而栗离开大楼的数据。网络本身也必须非常强大，以应对“早晨同步”：获取游戏文件的最新版本的日常仪式，这是艺术家或开发者的第一杯咖啡的一部分 - 和实际上，通常是在同一时间发生的。

第 6.8 节 管理操作系统

在电影和游戏中，通常在整个设施中使用多个操作系统（OS）以适应不同的任务或软件包。最常见的操作系统是 Windows，Mac OS X 和 Linux。尽管许多 VFX 工具将自己定义为“Linux 机构”，但它们通常必须在某种程度上支持所有三种操作系统，因为总有一些应用程序 - 例如，Photoshop 和 After Effects - 在主操作系统上不起作用。

分割单个机器以便在其上运行许多 OS 实例变得越来越普遍：称为“虚拟化”的过程。每个实例或“虚拟机”都获得主机物理资源的一部分，但与其他实例完全隔离，因此对于大多数用途，它们可以被视为单独的机器。 （例如，重新启动虚拟服务器不会影响在同一硬件上运行的任何其他服务器。）虚拟化允许您最大限度地利用资源，通过“热备份”启用“透明故障转移”（即，在用户使用时备份数据）访问它），并水平缩放。此方法适用于资源较少的任务，例如运行 Web 或 FTP 服务器或许可证管理。更密集的任务（例如运行文件服务器）可能仍需要专用硬件和物理连接。

为了简化管理操作系统的过程，除非严格必要，否则通常会使每台计算机都相同，并根据其“英雄应用程序”选择一个操作系统：例如，如果一台机器主要用于运行 Nuke，它会适合安装 Linux;对于 Photoshop，有必要安装 Mac OS X 或 Windows。或者，可能需要选择与机器 GPU 兼容的操作系统，或与图形平板电脑等关键外围设备兼容的操作系统。

当您确定每台机器的操作系统时，只需更改它（通常是升级到新版本）！

第 6.9 节管理实用软件

许多实用工具对于保持设施运行至关重要。许多程序都使用系统软件（例如 Java Runtime Environment 和 Microsoft 的.NET Framework 和 Visual C ++可再发行程序包），因此保持最新状态并供所有用户使用非常重要，尤其是在您部署的情况下依赖于这些库的生成工具。

图形和音频驱动程序是潜在问题的另一个来源，特别是对于 3D 艺术家。确保您的用户拥有这些关键基础架构的最新批准版本非常重要。如果您有一个大型工作室，那么投资远程审核软件配置的工具是明智的 - 手动检查一百个工作站上的图形控制面板不是一个晚上度过的好方法！

第 6.10 节生产安全

最后，我们将研究生产安全性。这是 IT 基础架构的一个方面，很容易被低估。生产合同需要严格的机密性，因此 IT 必须有适当的设置，以确保不会发生安全漏洞。这包括身份验证（是访问他或她声称的数据的人？）和授权（该用户实际上是否有权访问这些文件？）

一般而言，除非在供应商和客户之间签订合同，否则任何生产都是保密的，并且在任何禁运日期结束之前，该过程的细节仍然是禁止的。鉴于资金岌岌可危，这并不奇怪：大型项目的成本和潜在收入达到数亿甚至数十亿美元，因此有必要确保在发布前尽可能保证内容的安全。

在视觉效果工作中尤其如此。 VFX 行业受到非常强大的安全审核，可以测试生产的各个方面：员工和访客如何访问建筑物，员工如何登录机器，是否可以安装外部存储设备或刻录 CD 和 DVD，网络是如何分区，以及将数据上传到互联网是多么容易。但是，到目前为止，VFX 行业最大的担忧是云中的安全性：主要的电影工作室合同要求供应商确保数据在内部托管，而无需从外部访问数据。在建造管道时必须考虑这些因素，因为以后改装安全措施可能会非常昂贵。这些审计每年发生几次，这一点并不罕见，无论是由客户还是 MPAA：美国电影协会发起的。

三十多年来，MPAA 代表其成员公司，美国六大电影制片厂进行了网站安全调查：沃尔特迪斯尼影城电影公司，派拉蒙影业公司，索尼影视娱乐公司，二十世纪福克斯电影公司公司，Universal City Studios LLC 和华纳兄弟娱乐公司。

这些 MPAA 调查涉及广泛的主题，包括事件响应等管理系统;物理安全性，例如如何保护设施的访问点，以及物理存储介质和文件传输的安全性;和数字安全，包括身份验证和授权协议，以及文件管理系统的各个方面。

此类调查确定了应该实施的生产和控制风险，以将这些风险降低到适当的水平。国际标准化组织（ISO）将风险定义为“事件概率及其后果的组合”。在这种情况下，前者将包括从设施的网络中窃取内容的可能性;如果发生这种情况，后者将包括对设施和客户的业务后果（例如，违反合同和/或该发布窗口的收入损失）。

在与客户协商后，设施负责确定该客户的哪些资产需要更高级别的安全性。这是通过四个步骤完成的：

识别和分类资产

监督和评估有效性

确定最低安全控制集

实施控制

通常根据资产的分类，对组织的价值以及泄露或被盗的风险来选择安全控制。为了减轻已识别的风险，设施应实施与每种特定风险相称的控制措施。应根据当前的威胁环境定期评估此类措施的有效性。

根据 MPAA 内容安全模型组织最佳实践，该模型提供评估设施保护客户内容的能力的框架。它包含 49 个安全主题，可以在 MPAA 的内容安全最佳实践文档中找到，该文档可从其战斗电影盗窃网站 www.fightfilmtheft.org 的最佳实践部分获得。

插曲：通过定期维护和灾难规划降低风险

"工作室通常期望他们的项目能够在项目交付之前顺利运行而不会中断。遗憾的是，现实世界有干扰的习惯：新的硬件，软件更新，硬件故障，服务器崩溃或飓风和地震等自然灾害很少被生产计划考虑在内，但如果计划是没有到位来解决意外。规划和定期维护是控制和最小化对计划的影

响的关键。″

第 6 节插曲 1：计划停工

设施需要停机维修，但项目无论如何都要全天候运行。无论生产压力如何，都需要尽可能保护停机维护窗口。计划停机的三种类型是：

定期维护窗口通常用于持续 4-12 小时的较大任务，并且在非工作时间执行。

增量维护窗口适用于可以轻松分解为持续 10 分钟到 1 小时的子任务的任务，并且通常在一周内执行，最好在非高峰时段执行。

照明维护窗口用于快速重启服务器，持续时间不超过 10 分钟，并在工作日期间随时执行。

计划的停机时间类型取决于设施的需求和要完成的工作。如果工厂在多个时区设有办事处，这将需要办事处之间的良好沟通，以确保停机不会对他们产生负面影响。不安排计划停机时间会产生后果：无论如何都会发生故障，导致无法预测的停机时间和频率和持续时间的增加，并且会在最糟糕的时间发生。

第 6 节插曲 2：一般准则

规划有助于在分配的时间内完成平稳的停机时间。这些是适用于所有类型停机的一般指南：

如果存在以前未公开的生产问题，请尽早将生产纳入计划。 所有工作都应在少于 Production 同意的时间内完成。

创建包含以下内容的沟通计划：

有关停机时间需要联系的人员列表。 该列表可以集成到停机通知系统中。

如果系统关闭包含列表，则列表的硬拷贝很方便。 将此列表保留在电话（办公室或移动电话）旁边是明智之举。

需要了解影响它们的中断的其他办公室和供应商的列表。 如果有人将文件发送到停机维护并希望它可用的系统，该怎么办？ 谁是您的 ISP 或服务器供应商的联系人？

如果通知人员的通常方式因维护或紧急情况而停机，则通知人们停机时间。

选择小而有凝聚力的维护区域，以尽量减少测试和如果有问题，请排除故障。

创建一个清单，列出计划完成的任务，并确定哪些任务是关键的，哪些是好的。

创建测试计划或自动测试套件，以验证系统的状态是否已准备好供 Production 使用。

计划如何在发生意外情况时回滚。 将计划停机时间的大部分计划中的意外事故计划不及。

练习在停机期间完成的工作，以发现假设并减少意外。

一个好的策略是在侧面进行工作，然后在停机期间将其交换进去。

工作完成后，向相关方发送通知。 本说明应包含中断的详细信息：更改的内容，中断的持续时间，中断的原因（如果不是很明显），将来要查找的问题以及出现问题时联系的人员。

保持停机时间成功和失败的统计数据和“经验教训”将有助于规划未来的停机时间。 应跟踪此信息，因为这对于计划每年的停机时间，预算和改进停机流程非常重要。

保持停机时间成功和失败的统计数据和“经验教训”将有助于规划未来的停机时间。 应跟踪此信息，因为这对于计划每年的停机时间，预算和改进停机流程非常重要。

以焦点，测试和回滚时间计划停机时间可以成功维护。

第 6 节插曲 3：定期维护窗口

安排定期的时间段来执行维护是一种很好的做法。某些维护任务需要更大的窗口（四个小时或更长时间）才能完成工作，特别是如果它很复杂或影响很多系统或需要与多个供应商或系统进行大量协调。此外，如果大多数人在办公室，可以安排多个所需的任务，并执行可能影响设施的工作。在一些工作室，这项工作是在星期日进行的，当时大多数制作和工作人员都暂停了。完成的工作类型的示例可能是安装新的家庭服务器，网络维护，升级数据库或其他关键软件。与 Production 一起安排定期时间会考虑他们的需求，使调度更容易，并减少意外停机时间。

第 6 节插曲 4：增量暂停时间

当难以获得更大的时间时，请在设施最能负担的时间内计划少量的定期停机时间。增量可以持续 10 分钟到 1.5 小时。除上述指南外，以下提示还可以简化：

选择一个固定的时间在一天的某个时间进行维护，这将影响最少的人数。 确保此时间不会对其他办公室或任何期望交付的人产生不利影响。

选择单个维护区域，以便在出现问题时最大限度地减少测试和故障排除时间。 维护时间可能短至 10 分钟。

准备好执行回滚计划，以防出现意外情况。 将计划停机时间的一半到三分之一计划在意外之中。

练习在停机期间完成的工作，以便快速完成工作。

当停机时间很短时，在停电期间进行工作然后将其交换进去会增加成功的可能性。

短暂，专注，增量的停机时间以及测试和回滚的时间导致成功维护。

第 6 节第 5 段：翻滚

翻车停机时间是闪电停机时间，不到十分钟，专注于一项维护任务。这种类型的停机时间通常在清晨，午餐或其他人们不急于出版或出口他们的工作时进行。常见任务包括“滚动”数据库（上载新模式，发布非常小的补丁），添加更多磁盘空间，重新分配硬件或更换冗余或备份硬件，或分发紧急补丁。关键是通过使维护窗口非常短，维护类型极低（通过边线，测试和练习）以及易于撤消的工作来尽可能少地破坏设施。理想情况下，艺术家不会忽视停机时间。在白天进行此类维护的好处是，IT 人员随时可以监控系统状态，并在发现问题时快速进行调整。即使滚动时间短而小，仍应发送有关方面的通知，并应对事件进行简要审查并记录下来，以便以后分析。

非计划/紧急停车时间

将发生计划外或紧急停机。即使人们不知道何时会发生紧急情况，仍应进行规划。管理此风险的关键技术是：

制定沟通策略

制定一个在紧急情况下该怎么做的计划

知道谁可以提供帮助。 哪些供应商可用？ 你需要打电话给谁？

有一个定义的升级政策。 何时以及如何将其踢到楼上？

练习，练习，练习！ 这就像消防演习一样，目的是了解计划的工作原理以及在紧急情况发生之前需要改进的地方。

有替代路线/流程到位，以便设施尽可能继续运行。

例如：

驻留在另一个城市的备份，以防设施丢失建筑物

备用工作人员，以防有人生病或退出或休假/无法访问

备用网络路由

备用磁盘驱动器

备用计算机（可能会从上次升级中遗留下来）

从紧急情况中恢复后，请务必执行以下操作：

告知感兴趣的各方紧急情况已经结束，所以人们知道他们可以重新开始工作。如果你知道，紧急情况是什么以及是什么造成的，他们可能遇到什么挥之不去的问题，他们应该如何以及向谁报告问题。

举行追溯或验尸，以便如果可能的话，将来可以避免这种类型的问题（通过定期维护，改进硬件或进行物理改进，例如捆绑地震安全设备）下次可以改进响应时间和程序，并且可以在下一个预算周期中衡量和计算成本。在可搜索的位置（维基，Alfresco，数据库等）保存有关“经验教训”的信息，并在下一年的计划周期中对其进行审核。总之，无论是维护还是紧急情况，计划都可以在成本，进度和压力方面保持流程的可控性。

用于工作室环境的软件

第 7.1 节您重置本章中学到什么

在前面的章节中，我们研究了工作室完成电影或游戏项目所需的软件类型。在本章中，我们将了解该软件的来源。

传统上，有两种方法可以获得新工具：购买它们，或者在内部构建它们。然而，随着嵌入式脚本语言在艺术软件和 Python 等通用脚本语言中的兴起，设备越来越采用第三种方式：编写较小的工具来扩展或修改现有应用程序的功能。

我们将在本章中查看所有三个选项。在讨论了每种策略最合适的情况后，我们将列出工作室在购买软件之前应该考虑的问题;研究脚本如何为没有正式编程背景的艺术家开放工具开发世界;并探讨运营内部研发部门所涉及的一些关键问题。但首先，让我们简要介绍一下计算机图形软件开发的历史。

第 7.2 节我们和他们的：管道软件开发的方法

电影和游戏的图形制作一直与软件开发密切相关。在计算机图形学的早期，想要探索新媒体的工作室必须发明用于创建新类型图像的工具。早期计算机图形中最着名的名字，如 Jim Blinn，因其在 Blinn-Phong 着色模型上的工作而闻名; Adobe 的创始人 John Warnock; Per Perlin，Perlin 噪音的发明者;而后来 Pixar 总裁 Ed Catmull 都是计算机科学博士。甚至最早的 CG 工作室的名称，如数学应用集团公司（负责 1982 年 Tron 开创性数字效果的四家公司之一）和太平洋数据图像（现在是梦工厂动画 SKG 的一部分），清楚地证明了第一个一代图形工作室本

质上是软件公司。

那些第一代公司必须为自己创造几乎所有东西：不仅渲染算法和管道工具，还有硬件;皮克斯曾试图通过销售自己的定制工作站皮克斯图像计算机赚钱。当然，现在有一个庞大而发达的市场，可以在商用硬件上运行现成的图形软件。不再需要为每部电影或游戏发明新技术。即便如此，大多数制作仍然需要大量的内部软件开发，从简单的脚本编写到创建完整的应用程序。

最常见的策略是将商业软件用于核心生产任务，并创建轻量级工具和脚本库，将商业软件包组合在一起形成一个连贯的管道。然而，许多工作室仍然从头开发完整的应用程序。其中一些内部工具变得如此复杂，以至于它们本身就演变为商业产品：RenderMan 开始作为皮克斯的内部渲染工具，而 3D 雕刻程序 Mudbox 和人群模拟引擎 Massive 都是在 Weta 创建的在“指环王”三部曲的工作中数字化。虽然大多数内部应用程序没有达到这种复杂程度，但这些程序说明了复杂的专有软件是多么复杂。

因此，任何成功的生产都需要一种明确的方法来创建和管理其软件基础架构。这不仅仅是从可靠的供应商那里选择好的工具：了解管道的长期目标和生产团队的文化也很重要。

选择生产软件时，文化往往是最棘手的问题。很容易想象一个由负责专家做出技术决策的世界，而生产线上的人们很乐意接受该计划。然而，这不是我们实际工作的世界。对于领导者来说，应用 X 或产品 Y 是新项目的方式，但对于投入多年开发应用专用技能的艺术家而言，这似乎是显而易见的。可能会少得多。最好的情况是，转换可能需要数月的再培训和降低生产力。在新的软件包中，熟悉的热键可能不起作用，这意味着需要重新学习高度进化的肌肉记忆，并且习惯性工作流可能以不同的方式工作。对于经验丰富，高效的艺术家来说，这可能是非常令人沮丧的，特别是如果生产期限迫在眉睫。在最糟糕的情况下，不受欢迎的工具选择可能导致被动抵抗 - 甚至大规模叛逃 - 艺术家宁愿辞职而不是以他们认为不自然的方式工作。挑选或更换软件不仅仅是一种技术选择。

工作室的管道如何随着时间的推移而变化也决定了它的软件选择。具有高营业额的工作室 - 例如，快速扩展以完成大型项目的设施，然后缩减到下一个合同 - 通常需要更贴近现有应用程序的熟悉界面和工作流程。他们的选择可能是由当地的承包商提供的。在这种情况下，大多数定制工作需要在幕后完成，而艺术家直接接触的任何新工具必须直观且易于教授。另一方面，具有较长战略视野的工作室，例如那些从事知识产权工作的工作室可能会产生多年的续集，在引入自定义工具方面有更多的余地。如果您期望从长远来看能够收回您的培训投资，那么创建具有重要学习曲线的工具才有意义。

经过多年的生产，工作室建立和购买的软件数量很快就开始增加。随着新版本的商业工具的发布以及为特定项目编写的一次性软件，使用的应用程序数量可以迅速从数十个增加到数百个。当您考虑管道术语中的软件时，尤其如此。虽然对于大多数艺术家来说，“软件”是一个大型的，易于识别的，自包含的应用程序，如 Maya 或 Shotgun，但对于研发部门来说，它还包括从单行脚本到低级图形库的所有内容。

在本章的其余部分，我们将讨论工作室在决定如何管理笨重的软件基础架构时面临的一些关键问题，从最明显的一个开始：它应该购买新工具还是自己开发？

第 7.3 节何时建造，何时购买，何时修补

有许多企业需要技术，法律或财务方面的专有工具。大型制造和服务公司经常聘请外部顾问来创建定制应用程序，而不是依靠现成的软件。相反，较小或较不专业的企业通常完全依靠商业上可用的工具来完成文字处理，图像编辑或网络发布等任务。

电影和游戏工作室介于这两个极端之间。很少有工作室想完全依赖专有工具：不仅前期成本很高，而且培训和支持的持续成本令人生畏。同时，大多数图形制作都有商业产品无法满足的

要求。无论是对尖端渲染技术的需求，与生产管理系统的集成，还是对自定义数据格式的需求，管道中的一些重要部分通常无法使用可用的非现成技术进行处理。 -shelf 解决方案。

当然，大多数工作室宁愿专注于开发知识产权而不是开发软件。从头开始创建软件是一项重大投资。即使整个生产团队确信需要一个自定义工具，肯定会有一些会计人员想要知道什么，确切地说，需要所有时间，金钱和风险。一个好的经验法则就是不要在内部构建任何东西，除非你必须这样做。 “这里没有发明综合症”，需要重新发明轮子，因为现有的解决方案并不像你自己那样完全按照你自己的方式设计，是各地软件开发人员的副手，并且是一个受到强烈抵制的人。

因此，每个管道团队都需要努力跟上工具市场的步伐。每个 SIGGRAPH 或 GDC 都会推出一批新的生产工具，找到合适的生产工具非常值得一次会议通行证。对于扩展商业软件包（如 Maya，Houdini 或 Photoshop）插件的混乱二级市场尤为重要。插件是扩展管道功能的经济高效的方式，无需面对不熟悉的新用户界面的艺术家。其中一些非常复杂，一些最终成为行业标准。但是，插件供应商通常是具有不确定未来和有限支持能力的小公司，因此在将工具构建到管道中时仔细审查它们非常重要。

当很明显存在无法通过现有商业软件填补的需求时，团队需要对可能的解决方案进行成本效益分析。例如，工作室可能会发现它在处理大型模拟时遇到问题。艺术家在等待模拟结果时闲置显然是一个问题 - 但实际上失去了多少时间？这个问题是针对少数个人孤立的，还是影响到每个人？每天浪费一个艺术家一小时的时间是一种耻辱;每个人每天浪费十分钟的时间浪费了 2％的艺术预算。以这种方式量化问题应该可以帮助您找到最具成本效益的解决方案。

第一步通常是寻找“作弊”：一种随时可用的低技术解决方案。在上面的示例中，这可能是为艺术家提供了额外的 CPU。也许仿真可以由现有服务器场处理，它们可以在无人值守的情况下运行，允许艺术家处理其他项目。对于许多生产问题而言，蛮力可能是经济上有效的 - 如果不是优雅的解决方案。

但有时候，仅凭蛮力是不够的。通常，下一步是尝试使用大多数主要图形应用程序提供的脚本功能来解决问题。脚本工具的运行速度往往较慢，而且比定制软件的灵活性低，但它们的开发和测试速度更快。在上面的示例中，团队可能会尝试通过创建联网作业管理系统来解决长模拟时间，以使用 Python 或 MAXScript 在远程计算机上排队和运行模拟。

但是，有些情况下从头开始构建工具是唯一的答案。例如，实现一种全新的渲染或模拟技术可能只能使用自定义代码实现。精心编写的，高度优化的工具也可能是满足性能要求的唯一方法：在上面的例子中，真正的问题可能不是艺术家闲置，而是导演需要更快地看到模拟结果。正确的寻找一个镜头。

当必须构建新工具时，重要的是要确定这需要多长时间以及花费多少。开发时间表必须包括足够的测试时间，而预算必须包括维护成本和未来开发。管道软件永远不会“完成”。

有可能通过商业化来弥补创建工具的一些成本。我们查看了本章前面已成功商业化的内部软件示例：在该列表中，我们可以添加 Nuke，Mari，Katana 和虚幻引擎等等。然而，还有很多商品在商业上发布但失败了，因为他们的母公司无法有效地关注销售软件的重要但不那么光鲜的方面，如错误修复，文档，客户服务，支持和长期 - 术语产品开发。

对于游戏开发者而言，选择是建立还是购买特别重要，因为游戏通常是围绕许可引擎构建的：要么全部，要么处理关键任务。许可引擎可以使团队在预生产的第一天开始工作，而不是仅需要数月或数年的工作来启动项目。使用许可引擎也可以更容易地找到有经验的人工：例如，找到已经使用过虚幻引擎编辑器的艺术家相当容易，而源和 Unity 引擎支持大多数业余爱好者社区，他们通常和专业艺术家一样熟练 - 谁渴望在这个行业找工作。

同时，使用许可引擎会让团队使用可能无法反映其项目需求的管道 - 遗憾的是，引擎附带的管道工具通常在引擎开发人员的优先级列表中较低。可以理解的是，团队不愿意修改一个可以

从项目的第一天开始有效使用的即用型工具。但是，由于游戏与许可引擎的模板不同，因此可能需要更新随附的工具，或者即兴创作变通方法。

在某些情况下，尝试调整现有引擎可能会比从头创建一个更大的项目。但是，编写自己的引擎是一项艰巨的任务，而且经常被低估：团队经常无法计划创建和维护所需的数十种支持技术所涉及的工作量，从自定义数据格式到级别编辑器。

许多开发人员试图通过利用现有软件来最小化所需的自定义工具的数量：例如，小型团队直接在 3 ds Max 或 Maya 内构建游戏关卡是很常见的。虽然这节省了前期成本，但它还涉及到以后可能不明显的风险：此类软件包专为建模和动画而设计，而非水平设计，并且其用户界面未针对从玩家角度查看空间或注释交互式水平。中途之家将使用旨在减少开发自定义应用程序痛苦的新技术。例如，Fabric Engine 的 Creation 平台提供了一系列现成组件，可用于构建高性能 3D 工具，而一些团队甚至授权 Unity 游戏引擎，以便围绕 Unity 的可扩展编辑器构建自定义工具。

第 7.4 节购买软件：考虑要点

在决定购买软件而不是在内部开发之前，管道团队应该提出几个关键问题。我们将在这里介绍一些最重要的内容。

这个软件能满足你的需要吗？

很明显，您应该检查软件包是否可以完成您期望的所有操作。你不应该检查它是否可以做任何你不做的事情。公司经常陷入购买超出需求的软件的陷阱，而不考虑艺术家掌握这些额外功能的时间。

你对软件供应商有多大的信心？

评估供应商本身及其生成的工具非常重要。工作室和供应商之间存在着有趣的关系，因为它们是相互依赖的，但却有相反的需求。虽然供应商更喜欢每个人都定期升级其软件的世界，但工作室宁愿不必每年再次购买相同的软件 – 但仍希望快速响应错误报告和支持电话。重要的是要确信您的供应商将来仍然会提供这种支持。

这里有几点需要考虑。首先，您在与大公司或小企业打交道吗？作为一种粗略的概括，大企业往往更稳定，更不容易完全消失。另一方面，如果供应商拥有数千个其他客户，您应该考虑将其优先级排在何处。第二，目前企业是否存在财务问题？近期股价的下跌可能暗示未来的问题并需要进行额外的购买前调查。第三，谁是供应商的现有客户？这些客户是知名的还是有信誉的？联系过去的客户可以让您深入了解您可以期待的支持类型。

用户社区有多活跃？

即使您不能依赖供应商进行培训或支持，您也可以依赖其用户社区。强大的社区可以帮助您掌握工具，为您提供改进工作流程的技巧和窍门，或帮助您解决技术问题。在进行购买之前，请始终考虑软件的其他用户在论坛，邮件列表和社交媒体上的活跃程度。

软件的维护费用是多少？

这是一种常见的误解，即在初始购买完成后，软件购买成本就会停止。实际上，成本正在持续。除了购买软件包需要多少成本之外，还要考虑集成到管道中的成本，添加功能或修复错误需要多少成本，以及支持合同或未来升级的价格。

不要吝啬于支持合同！

早在 1991 年，我就很幸运地进入了美国仅有的三所教授计算机动画的大学之一。它投资了 20 万美元在两个 SGI 工作站和两个席位的别名的 PowerAnimator，前身为玛雅。它们各自被藏在一个特殊的房间里，只有精英学生才能希望去那里不幸的是，学院已经放弃了在支持合同上花费几千美元，这不仅仅是技术支持，还包括软件升级。更不幸的是，第 3 版是 PowerAnimator 有史以来发布的最疯狂的版本之一：以至于 Alias 几乎立即发布了一个补丁。然而，由于学院在补丁发布前购买了该软件，并且没有支付支持合同的费用，因此它没有资格获得升级。在接下来的四五年里，学生们被迫使用软件个性比我们创作的动画还要强，而且由于学校为了避免进一步的扩展而延长了几年的升级，最终面临着是继续教授古老的软件，还是重新购买的决定。结果是，（几年内）这间房，容纳了 20 万美元的投资，使学校如此先进，已被改造成一个储藏室。

您需要多久升级一次？

为了鼓励现有用户花更多的钱，软件开发人员倾向于每 12 到 18 个月发布一次升级。决定您需要购买哪些产品不仅仅取决于您是否需要它们所包含的新功能。管道中工具之间的复杂交互意味着您还应该考虑应用程序的“依赖关系”：它需要在其中运行的其他工具，以及依赖于它的那些工具。例如，如果您的公司购买的插件仅适用于其父软件的特定版本，则会影响您升级父应用程序的能力。您是否已充分致力于插件，您将暂停升级，直到您的插件供应商也发布新版本？

这种依赖关系可以同时影响许多应用程序：例如，如果用于包 X 的新 SDK 需要新版本的 Visual Studio或GCC编译器,这可能会影响大量表面上不相关的工具。“DLL Hell”，由Windows 库之间的冲突引起的问题，是许多工作室现在更多地依赖于像 Python 这样的脚本语言的原因之一：用这种语言编写的工具通常不太容易受到版本问题的影响。

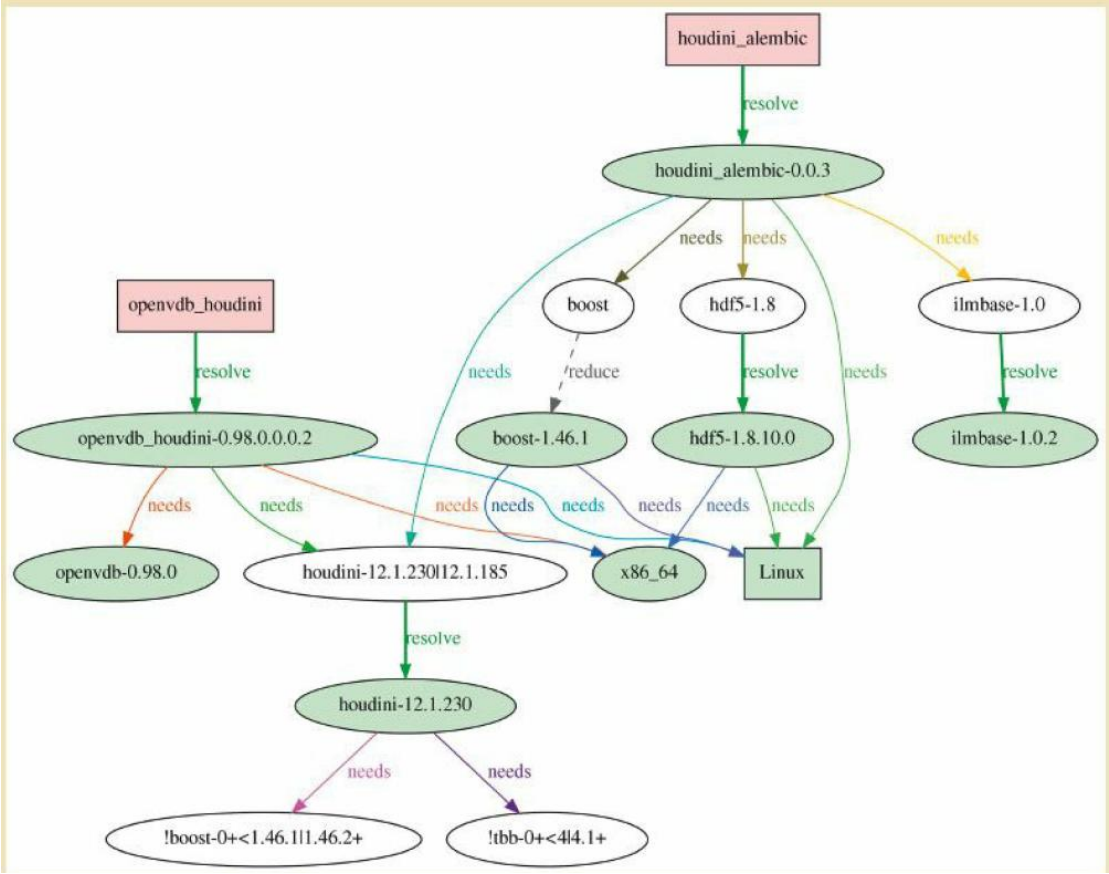
即使您决定需要特定更新，也可能无法立即将其推出。在制作过程中尝试升级通常被认为是蛮干的，因为这可能会导致生产力的短期打击，因为艺术家会接受变化，并引入错误或工作流问题。出于所有这些原因，寻找以三四年过时的技术运行软件处于最前沿的声誉的工作室并不罕见。

rez：一个用于管理 vfx 软件的开源工具

管理视觉效果软件有其独特的一系列问题。每个通常都有很多不同的版本同时在一个设施中使用：在不同的作业，序列，甚至个别镜头上。维护所有这些配置是一项艰巨的任务，因为每个突破可能依赖于其他几个附加，所以在一个地方进行更新可能会在其他地方导致冲突。视觉效果甚至可以说将旧的“依赖地狱”的概念提升到一个新的水平。一个低效的依赖关系管理系统将在不应该出现出现的地方产生降低。除非，如果没有健壮的依赖管理，您可能无法同时支持不同版本的依赖软件，这限制了您在不分区代码的情况下支持新应用程序的能力。而且，由于一个系统不能正确地描述依赖关系，或者依赖于大量容易出错的手工工作来

完成这些工作，因此常常会导致运行时错误，因此开发人员将花费大量时间来处理构建和发布

他们本来可以花在开发新工具上的问题本身，在描述依赖关系时过于严格的系统会增加维护规模，因为发布新版本的核心库会议可以您重新释放很多依赖的开销。从而使健壮高效的软件管理之路充满了这样的漏洞。一种有效的技术是将问题分成两部分：管理依赖状语从句：环境管理依赖关系管理的英文为了回答这个问题，“如果我需要这些软件包的版本，我还需要哪些版本的依赖软件包？“解决这些依赖关系必须避免版本冲突，因为在同一环境中使用同一软件的两个版本是有问题的（如果不是完全不可能的话）环境管理就是要回答这样一个问题：“我如何确定这些软件包我想在这个特定的工作，序列，或镜头？这两个问题常常结合在一起这导致了脆弱的开发环境，其中开发人员维护的更改可以打破现有的生产环境。它并管理作业配置的责任推给开发人员，而实际上他们应该只关心自己项目的直接依赖关系。当开发人员可以自由地编写独立于这些关注点的软件，并且当指定项目依赖关系很容易时，他们可以集中精力将软件正确地划分为组件，并保持这种方式。



rez 是一个用于构建、管理和部署软件的开放源码项目，是为数不多的针对视觉效果工作的依赖性管理解决方案之一。它使开发人员只关心管理自己项目的直接依赖项，并且具有依赖解析确保环境中不发生版本冲突的算法这既用于为软件包配置生成环境，也用于配置生产环境，以确保不会出现运行时版本冲突。rez 软件包是自包含的，在一个配置文件中描述它们所依赖的其他软件，以及它们如何影响它们所使用的环境。版本要求是灵活的，可以按要求严格地或严格地描述。让我们看看这样一个配置文件的示例：

```
name: my_maya_utility
```

```
version: 1.0.2
requires:
- maya-2012
- acme_util_lib-4.5 + <5
- foo-5.0.7
commands:
- export PYTHONPATH = $PYTHONPATH:!ROOT!/python
    - export MAYA_PLUGIN_PATH = $MAYA_PLUGIN_PATH:!ROOT!/lib
```

这里，Maya 插件需要 Maya 2012 的任何版本（2012.0、2012.1 等）；acme_util_lib 的任何版本等于或大于 4.5，但小于 5；以及 foo 软件包的特定补丁号当解析为环境时，python 和 maya 插件路径变量将更新，以便软件对相关应用程序可见。

当软件被中断时你会怎么做？

在考虑是否购买软件时要考虑的最后一个问题是，一旦供应商不再支持它，您将会做什么。即使是一个运行良好多年的工具，如果出现根深蒂固的错误，或者您的工作流程发生变化，将来也可能会出现问题。如果该工具深入集成到您的管道中，那么缺乏支持可能会对您的公司产生巨大影响。

为什么软件支持很重要

缺乏供应商支持对我的职业生涯造成的最大影响是，当时我正在使用一家知名工作室生产的引擎开发一款小型独立游戏。当时，它是唯一可用的 XNA 引擎，在过去对工作室的产品有过很好的体验，我们决定围绕它来构建我们的游戏开发一年多后，游戏在 Xbox 上运行超过 30 分钟后，我们就开始遇到巨大的性能问题因为我是我们三人组中唯一一个可以编程的成员，所以我开始优化代码以查找内存泄漏当我这样做的时候，我发现几乎在引擎的每个部分都有巨大的内存问题，但是在那一点上，制造引擎的公司宣布不再支持它。由于其他人对引擎进行了投资，用户社区联合起来。月后来，它成功地找到并修复了大部分的内存问题，但是在发动机的核心系统中发现了一个更大的内存问题。社区向小贩求助，但不幸的是，小贩不久就破产了虽然我们从未解决过这个问题，但我们请来了一位专家程序员和社区领袖，他帮助我们解决了不可修复的引擎问题，这样我们就可以免费发布我们的游戏了然而，由于我们的发动机选择所引起的问题实际上使我们的开发时间。从我们这样的经验来看，一个不受支持的产品如何对一个公司产生巨大的影响是显而易见的。我们的产品不仅受到影响，而且在运输之后，我们不得不更换引擎，这意味着我们不得不放弃现有的管道和我们所拥有的大部分框架。建造。许多定制工具的丢失，以及我们花在学习新工具和重新构建管道上的时间，都可以归因于一个糟糕的软件选择。

第 7.5 节使用开源软件

生产软件可能非常昂贵，每个用户通常运行数千美元。近年来，使用免费的开源软件作为传统商业产品的替代品已经变得很普遍。

最着名的例子是操作系统：拥有大型渲染农场的大型工作室使用免费的 Linux 操作系统而不是微软和苹果的商业替代品来节省大量资金。但是，许多其他管道工具都是开源的。 Python，

Lua 和 JavaScript 等流行的脚本语言是免费的，并且通常支持许多免费的高质量编程工具，如文本编辑器。 MySQL 和 MongoDB 等数据库引擎以及 Apache 等 Web 服务器也是开源项目。

还有一些开源艺术工具，包括 GIMP 图像编辑器; Blender，一个完整的 3D 建模，动画和渲染套件，提供与 3 ds Max 和 Maya 类似的功能;甚至可以处理用于 RenderMan 的文件的渲染器。

开源软件的低成本是一个明显的吸引力，但开源工具的另一个重要好处是它们的可扩展性：如果现有的软件包不能满足您的需求，可能有可能扩展或更新它，因为源代码是免费提供的。即使您不打算更改软件，访问源代码也可以跟踪错误或理解为什么事情没有按预期工作。大型开源项目通常也具有很高的质量声誉。由于有大量的贡献者社区和许多人关注他们的问题，开源项目的核心功能通常与竞争付费产品一样好或更好。

但是，使用开源软件存在缺陷。并非每个开源项目都有与 Apache 或 MySQL 相同数量的人，而较小的开源计划往往成为社区利益丧失或其贡献者之间争吵的受害者。更新可能是不稳定的，并且可能反映项目开发人员的内部政治而不是用户的需求。在考虑开源解决方案时，不仅要审查软件本身，还要审查开发它的社区，这一点尤其重要。这就是为什么许多公司成功销售“免费”软件的原因之一：它们提供了一定程度的可预测性，当涉及到松散的爱好者群体无法或不会提供的升级时。

但对于管道软件而言，主要的关键点往往是美学或情感而非技术。开源项目在历史上擅长于重量级计算任务，如编译代码或提供编程工具，因为大型，热情的用户社区非常善于发现错误。然而，用户界面设计通常是一个弱点，因为用户交互的微妙之处并不总是适用于基于社区的渐进式方法。艺术家往往对他们使用的工具的美学和人体工程学非常挑剔。这些偏好不容忽视，因为它们对生产力有很大影响，这意味着从长远来看，选择商业软件包可能更便宜，即使每个座位花费数千美元。

使用开源软件也可能具有重要的法律意义。有许多不同的开源软件许可证，有些可能会对工作室产生重大影响。 GNU 通用公共许可证（GPL）和其他“copyleft”许可证规定任何包含 Copylefted 组件的软件本身必须在同一许可下发布，这意味着它的创建者不能出售或分发它，除非他们愿意提供源代码码。许多公司（特别是游戏开发商，可能希望将他们的一些工具分发给粉丝或引擎许可证持有者）积极禁止使用 copyleft 工具，以避免与世界分享他们的秘密。因此，法律审查是评估工作室使用的开源软件的标准步骤。

第 7.6 节脚本和修补

重要的是要记住，“购买与构建”二分法仅包含生产中使用的部分软件。具有精美图形用户界面，3D 视图和惊人计算能力的大型应用程序引人注目，但在许多工作室中，大部分内部开发都发生在脚本编写领域，创建了各种较小的生产力工具。

消除容易出错的文书工作，减少执行常见任务所需的鼠标点击次数，并确保重要文件总是很容易找到，这可能不是最迷人的任务，但在拥有数百名艺术家的大工作室中，这种投资的回报可能是巨大的。如果您已经拥有一百名艺术家，那么您每天可以节省 5 分钟，每人免费获得相当于另一名工作人员的费用。

同样重要的是要记住，生产管道的任何部分都不能独立存在。在工作室中不与其他工具一起工作的工具并不比一端焊接完全的管道更有用。连接性是任何管道的“杀手级应用”，连接的关键是明确定义的，开放的 API 和数据交换格式，以及定制交互的脚本。

大多数专业艺术应用程序包括嵌入式脚本语言：Python 或 Lua 等共享语言，或者 3 ds Max 中的 MAXScript（MXS），Maya 中的 MEL 或 ZBrush 中的 ZScript 等应用程序特定的语言。图形应用程序的脚本功能从简单的宏开始，然后演变成功能更全面的脚本工具，能够自动执行复杂但重复的任务，例如更改大量场景文件的渲染设置。然而，脚本编写已经从简单地加速无聊的咕噜咕

噜工作变成现代制作中的一个关键（有些甚至可能说是关键）角色。脚本语言是将图形管道连接在一起的粘合剂。

脚本是非常通用的：它们可以处理各种任务，从简单的文书工作，如重命名文件一直到创建复杂的工具，如可从内部艺术软件访问的资产数据库。此外，它们通常比完全传统的应用程序更容易创建和分发，更少依赖于特定的共享库和处理器体系结构，并且通常是跨平台的：基于脚本的工具可用于 Mac OS X，Windows 和 Linux 机器很少或没有额外的工作。

脚本可以在主机应用程序中运行，用于特定于艺术的任务，也可以单独运行，并且这两组工具可以共享代码以避免重复工作。

然而，脚本的最大优点是可以编写脚本的速度。虽然高级脚本语言提供与 C 或 C ++等传统语言相同的复杂编程结构，但脚本也可以在现场即兴创作，以处理每个生产中出现的许多轻微紧急情况。创建一个 C ++应用程序，将项目中每个文件的扩展名从“.tiff”更改为“.tif”，正在构建一个火箭发射器，用于拍打苍蝇。在脚本中处理相同的任务就像在 Lua 解释器或 Python shell 中键入四行或五行一样简单。

通用脚本语言和传统编程语言之间的主要实际区别在于性能。对于计算密集型应用程序（如模拟或渲染），脚本语言可能要慢几个数量级。出于这个原因，数学繁重的任务往往属于使用 C 或 C ++的工具程序员。然而，对于许多普通的生产力应用程序，性能可能不是主要关注点：虽然渲染需要 12 小时和 1 分钟需要 10 分钟之间的差异非常重要，但需要 4 毫秒的文件操作和需要 4 毫秒的文件操作之间的区别 400 对于小批量任务可能不重要。

直到最近，DCC 应用程序中的脚本由 MEL 或 MAXScript 等特定于应用程序的语言主导。但是，在撰写本文时，有一种强大的趋势，即 Python 等通用语言可以在更广泛的环境中运行。

Python 是现代管道中最普遍的脚本语言。它嵌入在 Maya，MotionBuilder，Softimage，Nuke，Houdini，LightWave 和 Blender 中（并且可以在其他应用程序中使用，例如 3 ds Max，通过插件）。它也是一种“严肃的”编程语言;它通常用于教授计算机科学课程，它完全支持复杂的面向对象编程技术。然而，Python 最大的优势在于其庞大的内置扩展模块库。 Python 爱好者喜欢说语言“附带电池”，这意味着可以使用默认安装中包含的工具解决许多一般编程问题，这些工具涵盖从读取 XML 文件到设置 Web 服务器的所有内容。这为包含 Python 的任何应用程序增加了很多功能和灵活性。

然而，Python 并不完美。即使按脚本语言的标准，它也相当慢，并且它需要大量的内存。在现代机器上，特别是那些具有 64 位架构的机器上，这些缺点不一定是致命的，但在某些环境中它们可能很重要。另一个重要的缺点是缺少标准的 GUI 库：虽然有许多工具包可用，但没有一个工具包具有通用标准的状态，这意味着分发需要 GUI 的 Python 工具需要一个工具来分发第三方模块和 DLL 他们。然而，PySide 很快就扮演了这个角色，并且已经包含在 Nuke 和 Maya 中。

由于脚本在生产环境中非常有用，因此对脚本语言的支持是选择工具的主要标准之一。无法编写的无法编写的工具总是比那些具有良好脚本支持的工具更难集成到管道中，特别是那些支持通用语言的工具。如果管道中的所有工具都使用相同的语言，则可以更轻松地共享代码并节省开发工作量。

第 7.7 节内部软件开发：研发部门的作用

有时候，使用像 C ++这样的语言，用传统方式开发工具是无可替代的。因此，可以说每个管道团队最终都是一家小型软件开发公司。如果是这种情况，那么很好地理解软件开发在生产中所起的作用，即使您希望将内部工作保持在最低限度。

每个内部开发团队都需要提供四种主要服务：

开发全新的项目

加强现有项目

维护项目以保持最新的操作系统和相关库更改

错误修复和支持

这四种服务之间的分配取决于很多因素，但对于除了最大的工作室以外的所有工作，列表中较低的任务将占用大多数开发人员的时间。

开发团队的规模通常与工作室的规模有关。在视觉效果或动画设施中，艺术家与“技术”人员（包括 IT 和研发部门）之间的比例为 20：1 并非不合理。这里需要在技术和预算问题之间进行权衡：较大的研发团队会增加工厂可以创建的工具的范围，但在招标新工作时可能会使其在财务上失去竞争力。

相比之下，游戏具有更重要的工程组件，因此比率通常要小得多。从 1：1 到 6：1 的任何东西都是“正常的”，尽管如果你计算参与创建游戏中没有直接使用的材料的艺术人员，如预渲染的场景和营销材料，这个数字可能会更高。游戏公司很少有实际的研发部门，如果他们这样做，通常用于蓝天研究：开发是作为团队的一部分完成的，或者由中央“工具组”完成。

第 7.8 节内部软件开发：招募谁

雇用开发人员时，请记住程序员不是一样的。像艺术家一样，他们专注于许多不同的学科。例如，专门从事游戏低级运行时任务的程序员将拥有与专门为电影编写模拟工具的人员截然不同的技能。虽然可以找到能够在公司内部担任多种不同编程角色的员工，但您不太可能找到能够完成所有工作的人。

但程序员并不是团队中唯一能够为内部软件做出贡献的成员：能够编写代码的设计师和艺术家可以成为最好的工具开发人员。在许多游戏公司中，技术艺术家（TA）已经接管了传统上由工具程序员占据的部分角色。首先，技术艺术家或设计师通常使用他们创建的工具。让构建工具的人每天使用它将确保它保持最新并与项目不断变化的需求保持联系。技术艺术家也擅长交互设计：他们倾向于对直观，视觉上令人愉悦的界面有良好的关注。

然而，寻找具有艺术背景的工具开发人员的主要原因之一是图形工具是如此专业化，使得没有艺术经验的程序员难以有效地解决问题。对于来自该学科以外的人来说，解决问题可能比为自己编写解决方案更难。聪明的团队使非程序员能够自助。

在电影中，技术总监（TD）也扮演着类似的角色。他们往往是艺术家，他们在生产中开始了他们的职业生涯，但他们逐渐承担了更多的技术职责，要么追求对技术的热情，要么因为他们因工具不足而感到沮丧，并感到被迫将事情掌握在自己手中。他们专注于内容和编码之间的界限模糊的任务：例如，创建角色装备，构建着色器和设置复杂的模拟。它们通常使用脚本语言而不是开发大型独立应用程序，尽管许多应用程序在其职业生涯后期都会进行全时编程。因为他们每天都在处理制作工具和内容的现实，所以他们可以帮助技术水平较低的艺术家和核心工程师有效地协同工作。此外，TDs 可以很好地创建艺术家真正想要的工具。对艺术家需求和工作环境的深入了解使 TD 能够从工作流程中挖掘出粗糙的边缘，并为常见操作提供快速捷径。一个节省五到六次鼠标点击的按钮不值得创建一个完整的 C ++插件，但它仍然可以大大提高艺术家的工作效率和士气。

然而，从创作艺术到创作工具的转变需要时间。艺术家编写的工具需要像程序员编写的那样可靠且易于维护，这意味着技术顾问和运输署需要在工作中学习传统上培训程序员在学校学到的许多基本原则。

最后，请记住，艺术家不需要能够编码，甚至不需要脚本，以创建简单的工具。在游戏中，Excel 是可用于构建其他工具的工具的一个很好的示例。例如，设计师可以创建 Excel 电子表格，自动执行创建渐进曲线等任务（图形定义参数，如怪物韧性或武器强度，随着玩家在游戏中的进

展而变化，从而使后期阶段比以前的阶段更具挑战性）在将数据复制到数据库之前。

第 7.9 节内部软件开发：开发政策

内部研发部门通常会尝试遵循一些正式的软件开发方法，以确保它们有效运行。通常，这是“敏捷”编程的变体，其中开发以简短的周期迭代地进行。（我们将在后面的章节中讨论“敏捷”和“瀑布式”方法。）这种正式开发过程的要素可能包括：

定义软件规范的方法（例如，用户故事：描述用户需要软件执行操作的简短句子）

一个计划和预算，用于定义软件何时需要完成以及可用于构建软件的资源

用户界面标准（甚至可能是专用的 UI 设计者）

用于构建软件的构建/发布/部署系统

代码审查流程

测试环境（可能与构建过程相关联为“持续集成”：我们将在本章后面讨论的概念）

质量控制和/或质量保证流程（质量控制涉及检查软件的缺陷;质量保证涉及创建系统以防止首先出现缺陷）

文档过程

用于跟踪错误报告和功能请求的系统

我们将在本章的其余部分中更详细地介绍其中的一些元素。虽然像这样的高度正式化的过程通常是理想的而不是现实的，特别是在游戏开发中，在任何开发项目开始时都应该就一些关键点达成一致：

哪个版本控制系统用于源代码

开发团队将使用哪些语言

如何构建软件的标准

有关如何安装/部署软件的标准，以及版本控制模式

如何配置软件的标准：这应该定义特定于应用程序的配置，以及要运行的软件版本

第 7.10 节内部软件开发：测试新工具

在某些时候，必须决定是否正在构建内部工具“以完成工作”，或者构建为持久。这两种方法都是有效的，但有明显的缺点 – 并且对工具的测试方式也有同样重要的影响。

前一种方法几乎肯定会生成无法扩展的代码，难以维护，包含更多错误，并且几乎没有文档。但是，它将很快生产，并且需要更少的正式测试。后者需要完全不同的开发方法，以及更正式的测试方法。这可能会显着提高代码库的质量，但也可能显着增加所需的开发工作量。

重要的是要确保开发人员和“客户”对软件的可能质量和寿命有相似的期望。艺术家通常期望内部工具看起来像第三方工具一样精美，并且在部署之前将进行广泛测试。现实是，很多时候，艺术人员充当工具的测试团队：他们只是不知道它。对于仅用于单个项目的工具，进行更广泛的初步测试将是浪费金钱。

但是，有些情况下，最初只打算从门上获取特定镜头的软件被认为值得重用。应尽快将这些工具转移到更正式的开发方法上：越晚完成，重构软件就越困难。

单元测试是一种强调在逐个模块的基础上自动测试代码的实践（单独测试每个“单元”而不是整个程序）。特别是，使用单元测试的“回归测试”有助于在代码更改时保持一致的行为。例如，假设某个特定模块被设计为正确解析文件名。该模块的测试将输入已知文件名列表，并检查输出是否符合预期。然后，可以使用此方法验证随着模块随时间演变而行为不会发生变化。

为了支持回归测试，测试通常集成到一个框架中，允许它们作为批处理操作运行。为了提供帮助，可以将各种开源工具集成到版本控制系统中，以便对所有提交的代码运行这些测试，并提供每个项目状态的报告。这通常与“持续集成”的实践结合使用，其中开发人员经常将他们的更改合并到主版本中，以便比在合并之间允许长时间段更快地捕获问题。

单元测试和回归测试不保证软件没有错误，但至少它们确保它是外部稳定的。通过保证项目中的每个代码模块都能提供具有可预测输入和输出的稳定公共面，它们可以培养用户对内部工具的信心。对于使用脚本语言的程序员来说，这是一个特别重要的实践：在 C 或 C ++中，编译器会阻止您意外地将函数的返回值从字符串更改为数字。在 Python 或 JavaScript 中，除了单元测试之外，没有办法阻止它。

通过使测试过程成为工程工作的基础，测试驱动的开发使这些原则更进一步。在 TDD 中，在任何开发项目中编写的第一个代码是单元测试;那么实际的程序代码就是以满足测试标准的方式编写的。这个过程不仅可以确保对所有代码进行全面测试，而且可以帮助减少不必要的工作量（至少在理论上 - 实际上只实现满足测试条件所需的功能）。

测试非常适合捕捉行为的变化或不同软件彼此交谈的方式。但是，只有人类测试人员可以告诉您何时一个软件在技术上有效，但实际上并没有做正确的工作！尽管可能无法为内部软件提供与商业产品相同的质量保证水平，但让测试人员在部署之前检查工具是否正在完成其设计工作仍然很重要。这是 TA 和 TD 擅长的角色之一，因为他们经常发现工具在实践中的工作方式存在缺陷，纯粹的程序员可能会错过。

第 7.11 节内部软件开发：制定发布政策

每个软件版本都有可能将错误引入生产中。一旦新版本的工具经过测试和批准，重要的是以有计划，可预测的方式推出该工具，以最大限度地减少对计划的干扰。

在可能的情况下，以小增量进行更改。几乎每天都会发生微小的变化，但艺术团队需要提前警告工作流程或数据布局的任何重大变化，并在必要时提供文档和培训。

让艺术人员参与发布计划尤为重要。新工具的有效性取决于生产团队使用它的意愿，并且一个拙劣的版本可能会使一个新工具成为一个声誉不好的新工具，这个工具在问题得到修复后会持续很长时间。在一个重要的里程碑之前或者在一次大危机中，要求艺术人员吸收新工具（可能是一大堆小瑕疵）是不公平的。当然，推出不反映用户反馈的工具不太可能产生很大的热情。出于所有这些原因，工具团队进行内部营销和市场调查非常重要：找出艺术团队最需要的东西 - 以便将每个新版本顺利地整合到生产中。

第 7.12 节内部软件开发：编制文档

如果没有专门的软件开发公司的资源，许多工作室开发的工具设计精良但记录极差。未记录的部落知识是任何管道的祸根。承担严格内部项目的团队需要了解，创建准确，可读的文档和培训材料与编写新功能或修复错误一样，都是开发的一部分。如果建筑物中没有人知道如何使用它，那么你的软件很糟糕 - 无论它有多少基准测试。

要创建基本文档，开发人员可以使用开源文档生成器 Doxygen 等工具。 Doxygen 可以从源文件中的注释中提取信息，从而帮助半自动构建文档。它最初设计用于 C ++代码，但支持一系列其他编程语言，包括 Python。

对于希望将文档提升到更高水平的团队来说，合同技术编写者是一个很好的资源。没有更好的办法迫使开发人员澄清他们的想法，而不是让他们向一个聪明的，通常知识渊博的局外人解

释他们：向技术作家解释事物经常揭示设计或实施以前没有标记的工具的长期问题，因为他们只是“每个人都知道”的事情。一个优秀的技术作家能够提供有文化和准确的文档这一事实通常是一个额外的好处：他们的真正价值在于迫使开发团队去检查其工作。正如爱因斯坦曾经说过的那样：“除非你能向祖母解释，否则你真的不明白。”

任何编写软件的人都应该理解“技术债务”的概念 - 在该项目被认为是“完整”之前必须在开发项目上完成的幕后工作的积压。每当一个软件在没有文档和测试的情况下投入生产时，该软件的技术债务就会增加。超过某一点，债务将变得不可持续：在不严重影响生产的情况下，不能对软件进行任何改变，因为无法准确预测该变化的影响。很容易低估生产软件需要多长时间，或者需要多少工作来维护您添加的每个新功能。

第 7.13 节内部软件开发：报告错误

安全内部开发的最后一个重要组成部分是强大的维护。如果没有在出现问题时立即通知工具团队的系统，则不应向用户发布任何工具。错误报告可能与转储到用户硬盘驱动器的日志文件一样简单，也可能像每次崩溃后上传到数据库的完整内存转储一样复杂。

良好错误报告的关键是应该自动报告错误，并尽可能详细地记录上下文信息。显然，用户反馈仍然很重要，但由于使用这些工具的艺术家不是经过培训的 QA 测试人员，他们很少提供必要的信息来帮助程序员找出问题所在。

此外，艺术家有自己的工作要做：报告错误需要时间，如果最后期限迫在眉睫，他们更有可能解决问题而不是冒险危及他们自己的日程安排以产生良好的调试信息。如果一个工具获得了不好的声誉，那么艺术家在没有将文字传回给程序员的情况下完全停止使用它的情况并不少见。许多工具程序员惊讶地发现，他们的“用户”实际上并没有使用特定的工具数周甚至数月，因为发布不稳定或从未报告过的错误。除了记录故障和错误之外，许多工作室还记录了实际使用工具的频率，因为这在规划未来如何分配资源时非常有用。

良好错误报告的最重要好处是它可以产生良好的意愿。如果错误在发生时以电子邮件或 SMS 消息的形式报告，工具程序员，TD 或 TA 可以立即响应。对于迅速应对紧急情况的声誉，没有任何事情可以提高工具团队的地位：如果几分钟之后每次事故发生后都会出现有用的 TA 或程序员，艺术人员就会接受与管道团队的共生关系，并积极合作，使工具更好。

另一方面，如果投诉和错误报告消失在官僚机构中，那么生产人员很容易将工具团队视为冷漠和漠不关心。在这种环境中，艺术部门会开发迷信和秘密的解决方法，并且有效地使用工具而不是使用工具。服务是管道团队的真正要求 - 没有什么能说出“良好的服务”，如对问题的及时，明智的回应。

报告错误的系统包括 Bugzilla，MantisBT 和 Request Tracker 等开源工具，以及 JIRA 和 YouTrack 等商用系统。此类系统还可用于跟踪对新功能的请求。有些人可能会争辩说每种方法应该有不同的机制，但为了简单起见，一起处理错误报告和功能请求通常是有益的。

深入研究数据管理

第 8.1 节您将从本章中学到什么

在前面的章节中，我们研究了构建管道的硬件和软件。在本章中，我们将介绍如何设计管道。特别是，我们将探讨四个关键主题：

目录结构
文件命名约定
版本控制
元数据和数据库

其中一些概念在第 5 章中介绍。在这里，我们将更详细地探讨它们。但首先，让我们简要了解数据管理工作流程通常如何发展。

第 8.2 节数据管理工作流程的演变

由于资产通常由磁盘上的文件组成，因此文件系统构成了 1. 2. 3. 4. 5. 6. 7.构建管道和资产管理系统的基础。因此，最简单的数据管理方法是就合理的目录结构和文件命名约定达成一致，并手动维护。

虽然这种方法适用于小型工作室，但它不适合大型工作室。通常打破系统的是由于用户不遵守已经建立的惯例而导致的人为错误，因此数据的完整性受到影响。随着设施努力提高数据可访问性，安全性和质量水平，数据库和元数据成为显而易见的解决方案。元数据和/或数据库不会替换文件系统：它只是提供了一种更方便的文件搜索方式。

因此，管道的开发往往经历以下阶段：

1 获取适当数量的工作站。
2 获取集中存储。
3 定义和实现目录结构。
4 根据员工使用这些目录结构的方式，定义其他有用的资产分类方法。
5 构建数据库以存储此元数据，从而创建资产管理系统的基础。
6 定义最有效的分组资产和定义依赖关系的方法。
7 链接资产和生产计划任务。

我们将在本章的过程中讨论其中的一些过程。 首先，让我们看一下确定目录结构的一些问题。

第 8.3 节目录结构：平面与深结构

在开始新项目时，从一开始就规划目录结构非常重要。这里有几个基本的决定。第一个是您要在目录名称中编码的数据。第二个是如何安排目录结构。

目录结构可以是“深度”（每个目录中有多个级别的子目录，但每个目录中只有很少的单个文件）或“平面”（子目录的级别很少，但每个目录中有更多的单个文件）。这种选择会对人类用户和自动化系统访问文件的速度产生重大影响。最终，你想要一个不那么深的结构，你很容易迷失在其中，但不是那么平坦，你被淹没在你无法排序或过滤的文件中。这里没有确定正确解决方案的科学方法：相反，建立一个有效的目录系统更像是一种艺术形式。

在定义目录结构之前，您必须决定直接使用文件系统还是使用抽象层，使用户可以选择如何组织数据视图。诸如 FUSE（类 UNIX 操作系统的内核模块）之类的抽象层允许用户在抽象目录结构之上指定任意文件系统视图，这可能是原始形式的人类无法理解的，但需要大量的开发工作，并且通常不是首选的解决方案。

目录结构和版本控制系统

当涉及到设计目录结构时，有一些基本的决定将作为将来所有选择的框架。其中之一就是如何处理工作的迭代。在文件系统中有两种表示版本的主要方法：作为版本控制的文件或签入/签出系统正如我们在第 5 章中所讨论的那样，版本化的文件在于它们的简单性。您不需要特殊工具来为文件指定符合版本控制策略的名称，也不需要选择文件的适当版本但是，这种设置的结构远不如签入/签出系统。因为您联机的所有文件都在目录结构中可用，并且可以以任意方式访问，所以它可能是可能会无意中移动或删除文件的历史记录，并跟踪“依赖关系”--一个文件依赖另一个文件的方式更为困难。而且，由于版本控制每次更新文件时都会更改文件名，因此对该文件的所有引用也必须更新：通常是一个冗长的过程，如果不正确执行，会产生重要后果签入/签出设置更复杂,需要更多基础设施才能实现。技术水平较低的艺术家可能会迷失在这样一个系统中，因为他们正在开发的资产的哪个版本的细节可能会被隐藏起来但是,为长时间运行的呈现之类的事情提供真正的数据隔离更容易。依赖项跟踪也更容易，因为通常在给定的时间只有给定文件的一个版本可用。当一个更新的文件被签入时，这样的系统可以自动更新对它的所有引用，从而节省大量的手动工作。在设计目录结构时要做的另一个基本决定是如何将文件更新流到下游任务。这里的选择是“推系统”和“拉系统”。在 push 系统中，更新是自动获取的；在 pull 系统中，需要有人将更新拉进来明确地。这不是一个全有或全无的选择：管道的某些部分可以拉动，而其他部分可以推动。任务在人工或计算方面的开销越大，对于基于拉取的工作流，候选任务就越好，因为这使得实现真正的数据隔离变得更容易。但是，在很长的一段时间内，例如动画的制作时间表,花在跟踪某个资产的哪个版本在可能变得重要的地方使用的时间基于推送的工作流将减少正在使用的资源的不同版本的数量，并使确保所有活动任务都使用所有文件的最新版本更加容易。即使在使用基本上基于推送的工作流时，也有可能需要锁定某些任务（例如，当快照接近最终版本时，决定不允许对其组件资产进行任何进一步的更新），因此确保管道支持这两种操作模式是一种很好的做法。文件版本控制和签入/签出系统都与推式或拉式工作流兼容。为了实现推送工作流，版本控制系统通常使用“符号链接”（符号链接：包含对另一个文件或目录的引用的文件，如 windows 快捷方式）等技巧来表示文件的最新版本，而签入/签出系统则使用共享等策略沙盒环境的工作区或定期更新。有了这两个基本的决定，就没有对错的答案了。说到目录结构，“尽可能简单，但不简单”这句话非常贴切。

第 8.4 节电影目录结构

从很大程度上讲，电影资产创作过程分为两部分。首先是建造生产所需的所有角色，生物，道具，建筑物，植被，环境，车辆等，通常在生产中称为“资产构建”。第二个是将这些资产构建组合成场景和镜头。

可以说，组织资产最常见的模型是在目录结构中对项目，场景和镜头进行编码。因此，初始结构可能类似于：

/projects /<project >/<scene >/<shot >

这里要注意的第一件事是资产构建没有合适的位置，因为它们通常不是基于场景或镜头 - 并且在存储在共享库中的资产的情况下，甚至可能不属于单个项目。解决方法是创建一个名为“assets”的场景，它将存储所有资产构建，然后将资产构建文件归档为单独镜头。例如：

/projects /<project >/assets /<asset >

现在，目录结构在任何从事镜头或资产构建的人身上都是相同的 - 这几乎都是制作中的所有艺术家。下一个决定是目录结构应该更深入。虽然你可以简单地将所有内容存储在镜头文件夹

中，但是这个文件夹很快就会变得混乱，其中包含名称相似的混淆文件。另一种方法是创建更多目录级别

一个明显的选择是根据纪律划分下一级目录：

/projects /<project >/<scene >/<shot >/2d

/projects /<project >/<scene >/<shot >/3d

另一种选择是按部门划分：

/projects /<project >/<scene >/<shot >/anim

/projects /<project >/<scene >/<shot >/comp

/projects /<project >/<scene >/<shot >/model

在此之下，您可以为每种类型的数据添加目录：

/projects /<project >/<scene >/<shot >/anim /maya /cache

/projects /<project >/<scene >/<shot > /anim /maya /curves

为清楚起见，您可能还包含一个指示资产版本号的更高级别的目录：

/projects /<project >/<scene >/<shot >/anim /maya /cache /v1

/projects /<project >/<scene >/<shot >/anim /maya /cache /v2

/projects /<project >/<scene >/<shot >/anim /maya /cache /v3

资产管理系统使用的目录而不是手动访问的目录遵循类似的结构，通常使用紧靠镜头级别的目录来使其功能显式：

/projects /<project >/<scene >/<shot >/assetManagement /

<assetType >/<assetName >/<version >/

或者使用一个具体的例子：

/projects /bob /SB /SB_0001 /assetManagement /AnimatedCharacter /foo /v10

目录本身的命名约定因设施而异。 VFX 项目往往有多个名称，包括实际的电影标题，代码名称和此代码名称的缩写。其中任何一个都可以用于项目目录。场景和镜头由字母数字字符串表示，通常在项目和场景之间具有额外的级别 - 不在目录结构本身中反映 - 以表示序列。例如，场景可能被称为“XX_01”，其中“XX”表示序列，“_ 01”表示场景落在序列中的位置。镜头往往会继承场景的名称，加上一个额外的字母数字后缀：例如，“XX_01_01a”。

设施必须做出的另一个基本决定是，它用于项目的文件命名约定是否应与客户端的文件命名约定相匹配，或者它是否应反映其自己的内部工作流程。使用这种内部或“虚拟”命名约定意味着必须在传递给客户端进行审查或交付之前重命名文件，但意味着相同的结构可用于多个项目，并且镜头的名称可以是如果客户在生产中途改变自己的惯例，则更容易改变。

第 8.5 节游戏目录结构

游戏的目录结构与几个重要方面的电影不同。首先，在开发游戏时，您不只是在开发内容：您还在开发代码库。这反映在高级目录结构中：

/projects /<project >/code

/projects /<project >/tools

/projects /<project >/assets

与电影作品一样，每个资产都包含游戏所需的重要数据块：例如，骨架，动画，纹理或脚本。然而，与电影作品不同，资产不是简单地根据创作它们的软件来划分的。

正如我们在前面的章节中所讨论的，Maya 等艺术工具不会以适合游戏引擎直接使用的格式保存内容。它们的本机文件格式可能包含引擎不需要的信息，因此读取速度慢;或者它可能缺乏引擎确实需要的信息。游戏引擎通常以二进制格式读取内容，该二进制格式与目标平台的“字节

顺序”（二进制数据被编码用于存储的顺序）和“结构对齐”（数据结构与存储器地址边界的对齐）相匹配。

这意味着通常有离线准备过程将艺术软件创建的资产（“工作文件”）转换为特定于平台的版本（“游戏资产”）。在某些情况下，这是通过从工作文件中导出所有数据或选择数据以创建游戏资产的插件来完成的。这意味着为两种类型的内容创建单独的目录：通常发生在文件路径根目录附近的分区：

/projects /<project >/assets /workFiles

/projects /<project >/assets /gameAssets

此级别以下的大多数目录都是从 work-Files 镜像到 gameAssets。这意味着导出过程可以通过将“workFiles”替换为“gameAssets”来自动生成正确的导出路径。

或者，一些管道将工作文件转换为引擎端的游戏资产。在这些情况下，艺术家只需将其文件保存到 gameAssets 目录中。引擎在下次运行时检测文件，并将转换后的适合游戏的版本保存到缓存或数据管理系统中。游戏在构建过程中使用此转换后的文件。

这样一个系统的好处是它避免了重复 workFiles 目录树的需要，这意味着艺术家不必花时间手动转换文件。缺点是如果要正确转换，艺术家必须保持文件“干净”。例如，如果导入过程从 Maya 场景文件中引入所有网格，则在动画在游戏中正常工作之前，处理角色攻击动画的动画制作者可能必须删除对剑的任何引用。通常可以构建文件转换器以自动处理引用，但问题不能完全消除：艺术家添加的额外材料，地平面或对象都可能导致问题。此外，在发生错误时没有即时反馈 - 即使转换工具可以检测到错误，它也无法在运行之前报告，这可能是在保存违规文件之后的一段时间。

在 gameAssets 和 workFiles 目录的级别之下，资产必须进一步细分。虽然电影被分解为场景和镜头，但游戏通常分为级别，章节或轨道。例如：

projects /<project >/assets /workFiles /levels /<level >

projects /<project >/assets /gameAssets /levels /<level >

然后将每个级别进一步细分为一些逻辑高级别分离。例如：

…/levels /<level >/levelInfo

…/levels /<level >/scripts

…/levels /<level >/sequences

游戏也经常将大型级别分解为更小的子级别，这些子级别单独“流式传输”（加载到内存中）：

…/levels /<level >/section

游戏可以进一步按目的对文件进行分类，以便可以在适当的时间加载和卸载正确的资产。常见的子部分包括碰撞，效果，游戏玩法，几何，灯光，道具，序列和声音（环境，FX 和对话）。例如：

…/levels /<level >/section /collision

…/levels /<level >/section /sound /dialog

并非所有资产都在一个级别内：许多资产在多个级别共享。因此，与 levels 目录并行地，通常创建反映共享内容类型的其他目录。常用目录标题包括 AI，游戏流，systemAssets，animatedProps，角色，交互，过场动画，控制器，调试，效果，实体，图标，前端，模型，存储游戏，模式，评分，脚本，着色器库，奖杯和 vfx。

大多数这些目录将在 gameAssets 和 workFiles 中进行镜像：

projects /<project >/assets /gameAssets /AI

…

projects /<project >/assets /gameAssets /vfx

projects /<project >/assets /workFiles /AI
…
projects /<project >/assets /workFiles /vfx

但是，如果使用直接以游戏就绪格式导出的自定义工具创作特定类型的内容，则无需转换过程。相应的目录将仅存在于 gameAssets 中，不会在 workFiles 中进行镜像。

为您的文件夹选择正确的文件夹结构

您选择的目录结构将对您的项目产生重要的后勤后果。
为了说明这一点，让我们看一个虚构的例子。假设您正在开发一个游戏项目，其中环境资产的结构如下：
•游戏包含二十种不同的环境
•每个环境由三种不同类型的对象组成：结构元素，交互对象和装饰
•每种对象类型都有三种尺寸：小，中和大
•每个对象由三种文件类型组成：模型，纹理和动画
每种环境和对象类型都需要自己的目录。让我们研究一下可以完成此操作的两种方法。首先，您可以将资产划分为多个环境；然后将环境分为对象类型；物体分为大小；最后是文件类型。这将导致如图 8.1 所示的目录结构，总共有 800 个单独的目录。另一种选择是先将资产分为对象类型，然后再分为大小，最后再分为环境。最低级别的目录还是由文件类型组成。生成的目录树如图 8.2 所示，由 732 个单独的目录组成。上面的示例代表了实际资产所需的很小一部分资产
游戏，但是需要创建的文件夹数量已经有 8.5%的差异，而且更重要的是，在项目的整个生命周期中，需要浏览数百次。这很容易
看看这可能代表什么时间和精力上的差异。

第 8.6 节目录结构：为便于导航而设计

除了目录结构生成的目录总数之外，考虑导航的容易程度也很重要。为此，请考虑其预期用户将如何从一个对象转到另一个对象。

真正强调最小化导航时间重要性的游戏的一个例子是关卡设计师的工作流程。关卡设计师需要访问大量不同类型的艺术对象 - 平铺集，结构，装饰，FX，角色等等 - 以便组装关卡，因此关键是他们可以尽快完成。

假设您的关卡设计师需要访问两种特定类型的资产才能装饰房间：从天花板悬挂的链条，以及悬挂在这些链条上的灯笼。您可以设计一个文件夹结构，如图 8.3 所示。

如果艺术家的工作流程是：
导航到链文件夹。
1 放置链条。
2 放置链条。
3 放置链条。
4 导航到灯笼文件夹。
5 把灯笼放在链子上。
6 把灯笼放在链子上。

7 把灯笼放在链子上。
另一方面，如果艺术家的工作流程如下：那将是非常麻烦的：
1 导航到链文件夹。
2 放置链条。
3 导航到灯笼文件夹。
4 把灯笼放在链子上。
5 导航到链文件夹。
6 放置链（等等）。

在后一种情况下，您可能希望与艺术家合作解释他们如何改进他们的工作流程 - 或者只是改变目录结构以最大限度地减少浪费的导航时间。同样的原则适用于电影作品。

第 8.7 节目录结构：规划共享资产使用

设计目录结构时的另一个考虑因素是如何处理不符合任何预定类别的资产。这通常发生在共享纹理上。

例如，在游戏中，“地板”对象可以与“斜坡”对象或甚至“墙”对象共享相同的纹理。对共享纹理的对象进行分组非常重要,这样设计团队就可以快速了解将它们添加到关卡中的性能成本。一种历史上非常常见的方法是将对象分组为“环境集”。
例如，您可以按如下方式对环境资产进行分组：
洞穴：使用洞穴纹理的资产
林：使用森林纹理的资产
我的：使用矿山纹理的资产
Generic：对象意味着在多个级别中使用它们自己的较小纹理

使用这种方法，关卡设计师可以通过检查它们加载的环境集来跟踪它们使用的纹理数量以及占用的内存量。您还可以通过限制设计人员使用来自给定数量的环境集的对象来帮助确保级别保持在其纹理预算内：例如，洞穴和矿山集，或洞穴和森林集，但不是全部三个同时。

虽然电影没有相同类型的内存限制，但您的团队仍然可以计划最小化内存中保留的纹理数量，并尽可能重用纹理。这些纹理可以从库中拉出，在同一个项目中的另一个镜头，甚至是另一个项目。

第 8.8 节目录结构：从最一般到最不一般的构建

作为一般规则，项目的目录结构应该从大多数到最不通用构建。如果其他搜索方法失败，忽略此规则将很难知道从哪里开始查找特定资产。

让我们想象一下，你正在制作一个游戏，其中艺术家需要创建三个螺旋楼梯，两个普通楼梯和一个非常特殊的楼梯，称为“Zenobia 楼梯”。图 8.4 显示了根据 Most to Least Generic 规则构建的目录结构。

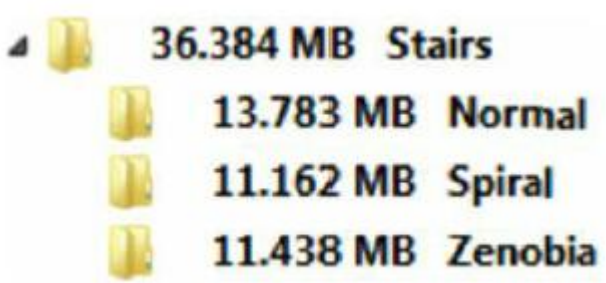


图 8.4 根据从大到小的一般规则组织的目录顶层目录是最通用的资源类型（楼梯）；下层目录是特定类型的楼梯。

图 8.5 显示了破坏规则的目录结构。想象一下，如果设计师需要将楼梯添加到某个楼层，但又不知道哪种类型的楼梯可用，这会让人感到困惑。这就像试图通过门牌号码而不是街道名称来查找地址。

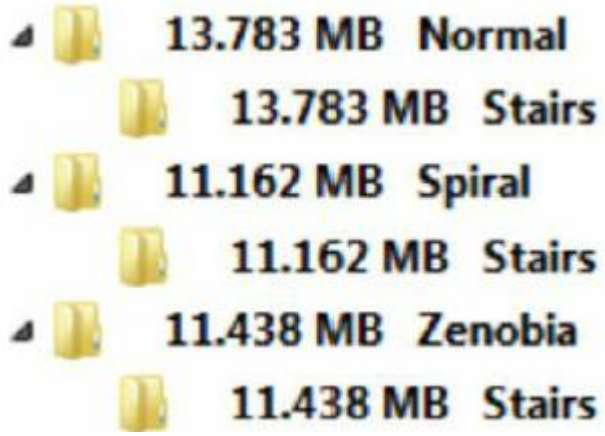


图 8.5 一个目录结构，它打破了大多数到最少通用的规则。除了创建比所需更多的目录之外，这种结构对新用户来说更难导航。

值得注意的是，“最通用”对于不同的项目可能意味着不同的事情。为了理解原因，我们先来看一些具体的例子，首先是电影，然后是游戏。

想象一下僵尸电影，其中艺术家正在为角色的皮肤和头发创造 FX。如果结果令人信服，每个僵尸必须有自己独特的效果。这可以通过两种截然不同的方式实现。

在第一个中，“泛型”被定义为字符类型。在这种情况下，每个角色都会收到自定义效果，因此一个角色的僵尸 FX 看起来不像另一个角色。这将产生如图 8.6 所示的目录结构。

当只有少数僵尸时，这很有效。但是，随着字符数量的增加，为每个字符创建单独的 FX 很快就会变得非常耗时。如果有很多僵尸，则需要采用不同的方法。

在这种情况下，“generic”被定义为 FX 类型。对于分配给它的每个字符，特定的头发或皮肤效果将是相同的。但是，通过为每个角色赋予其自己独特的 FX 排列，仍然可以使每个角色看起来不同。这将产生如图 8.7 所示的目录结构。

需要注意的一点是，图 8.7 中所示的目录树仅包含每个 FX 的三种变体。如果您管理大量变体，最好使用资产管理系统来封装信息，而不是文件系统。

接下来，让我们看一下来自游戏世界的例子。想象一下，你正在研究一种有六种角色类型的回合制 RPG。角色具有神奇的能力：治疗，伤害，对其他党员进行打击等等。每个人的 FX 都

需要在视觉上与众不同，但由于游戏是回合制的，因此玩家在使用时无需对能力作出反应。

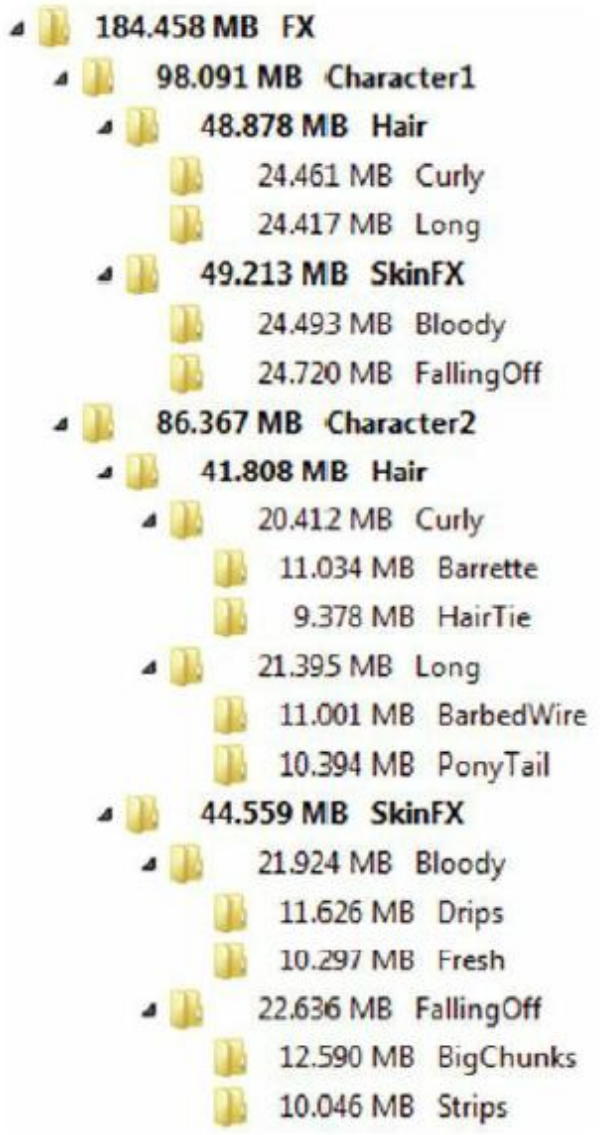


Figure 8.6 A directory tree for a zombie movie in which each character has its own unique FX. In this case, individual FX are defined as being less generic than the characters to which they are assigned. This works well if there are only a few zombies.

图 8.6 僵尸电影的目录树，每个角色都有自己独特的特效。在这种情况下，单个 fx 被定义为比分配给它们的字符更通用。如果只有几个僵尸的话，这个效果很好。

在考虑这些目标时，FX 艺术家可能会决定每个角色都会为他们的每个角色获得独特的效果：例如，一个角色施放的治疗法术看起来不像是另一个角色施放的法术。在这种情况下，“generic”被定义为字符类型。这可能会产生如图 8.8 所示的目录结构。

现在让我们看一下动作 RPG 的等效目录结构。同样，有六种角色类型，但在这里，玩家需

要能够在使用时对能力作出反应：当 FX 完成游戏时，他们应该能够决定另一个角色是否已经施放治疗或伤害拼写。这意味着 FX 不仅必须在视觉上与众不同，而且必须一致且可识别。

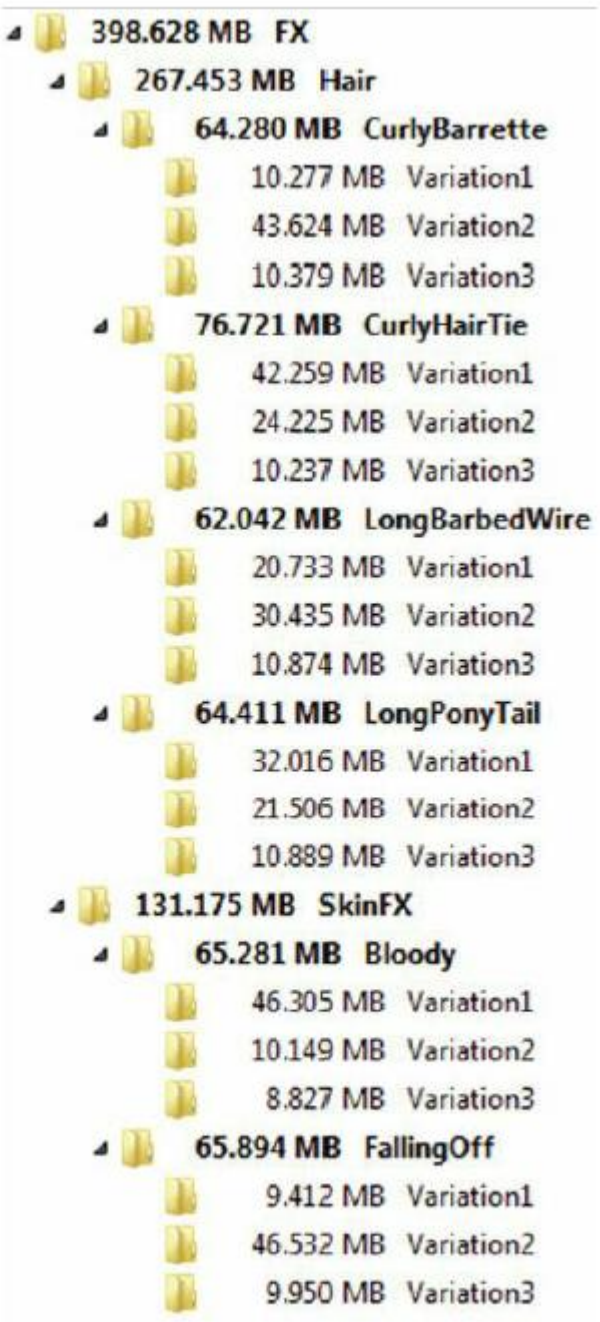


Figure 8.7 A directory tree for a zombie movie in which FX are identical for each character to which they are assigned. Here, characters are made to look unique by assigning unique permutations of FX.

图 8.7 僵尸电影的目录树，其中 FX 对于每个角色都是相同的在这里，通过指定 fx 的唯一排列，使字符看起来是唯一的。

当有很多僵尸的时候，这个效果很好。

在这种情况下，艺术家可能会决定角色将使用共享 FX 来表示各种能力类型。虽然治疗和伤害的效果仍然不同，但是一个角色施放的治疗法术看起来就像是另一个角色施放的治疗法术。这里，“通用”被定义为能力类型。这可能会产生如图 8.9 所示的目录结构。

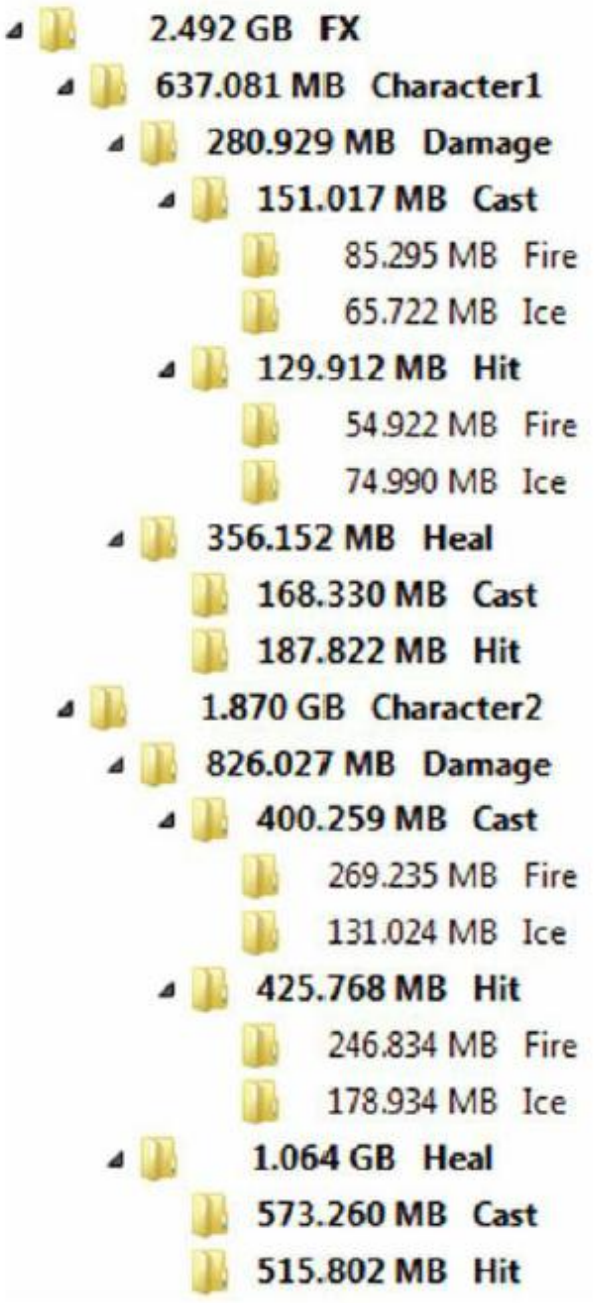


Figure 8.8 A directory tree for a game in which each character has its own unique FX. Here, character type is defined as being more generic than FX type. This might be appropriate for a turn-based RPG.

图 8.8 一个游戏的目录树，每个角色都有自己独特的 fx。这里，字符类型被定义为比 fx 类型更通用的类型。这可能适用于基于转弯的 RPG。

如您所见，术语“通用”表示每个项目都有所不同。了解电影或游戏的设计将有助于您创建文件夹结构，使团队成员能够快速轻松地查找资产。

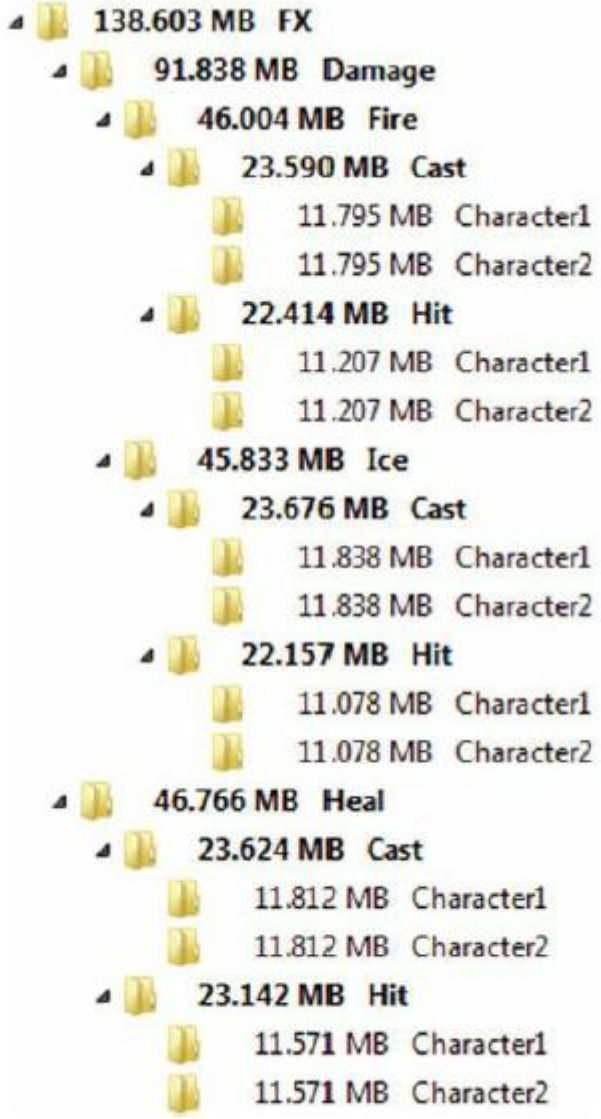


Figure 8.9 A directory tree for a game in which the FX for a given ability are identical for each character. Here, ability type is defined as being more generic than character type. This might be appropriate for an action RPG.

图 8.9 一个游戏的目录树，在这个目录树中，给定能力的 FX 对于每个角色都是相同的这里，能力类型被定义为比字符类型更通用的类型。这可能适用于动作 RPG。

第 8.9 节目录结构：合并资产模板

创建可以开发艺术资产的“模板” - 例如，通用人体模型，标准烟雾 FX 或库存运动捕获数据库 - 可以节省大量的生产时间。虽然这些模板是构建单个资产的起点，但它们不一定用于成品。因此，它们必须与成品艺术资产分开。有几种组织目录树的方法，每种方法都有自己的优点和缺点。

第一种是为模板创建一个目录树，以反映已完成的艺术资产的目录树。这导致如图 8.10 所示的结构。

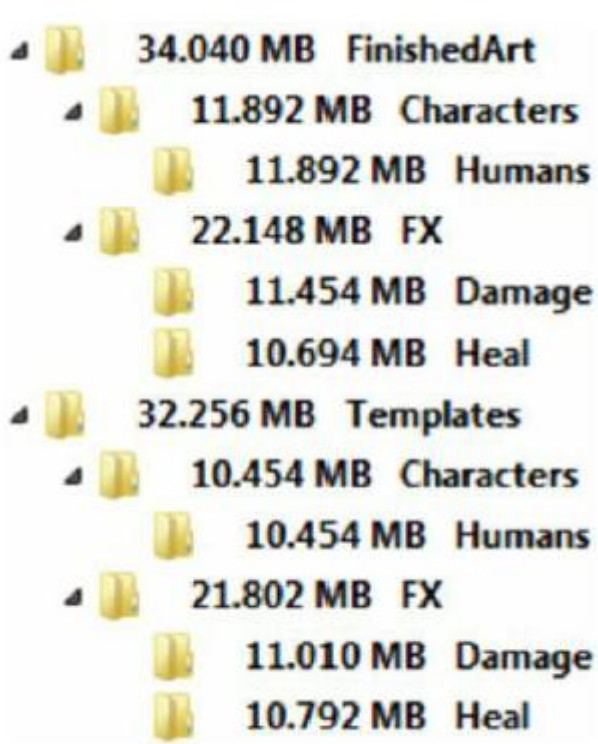


Figure 8.10 A setup in which the directory tree for the templates mirrors that of the finished art separates the two sets of assets clearly, but increases navigation time.

图 8.10 一种设置，其中模板的目录树与已完成技术的目录树相镜像，将两组资产清晰地分离，但增加了导航时间。

这种设置的优点是可以清楚地分离两组资产。在一部电影中，这有利于保持项目区域的大小，有助于确保模板资产在工作完成后不会与其他文件一起移动到离线存储;在游戏中，它可以缩短构建时间。但是，在项目过程中，在模板和完成的艺术目录之间来回导航所花费的时间可能很长;并且还必须对完成的艺术资产的目录树进行任何更改，以便对模板进行更改。

第二个选项是创建模板特有的目录结构，如图 8.11 所示。

此设置的优点与以前相同：两种类型的资产保持独立，减少了项目区域或游戏构建的大小。同样，在导航目录中花费的时间可能会降低首先使用模板的好处，但现在对完成的艺术资产的目录树的更改不需要自动导致模板的更改。

第三种选择是将模板和成品艺术资产集成到一个目录树中，如图 8.12 所示。

此设置的优势在于减少了导航时间并且文件更易于管理，但增加了模板与最终艺术资产混淆的风险。在电影中，它们可能会被改为“一次性”，或者只是在文件存档时被遗忘;在没有工程支持的游戏中，它们可能成为构建的一部分，增加安装大小和构建时间。

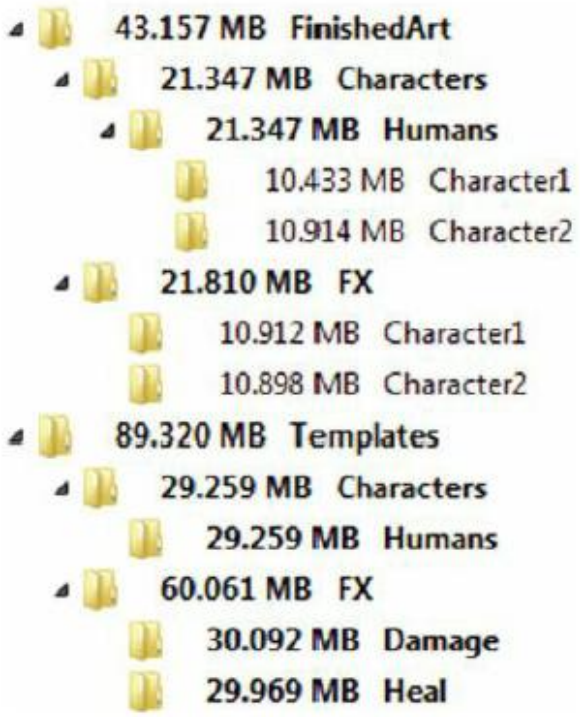


Figure 8.11 A setup with different directory trees for templates and finished art assets. Again, the two types of asset are kept separate, but now changes made to the directory tree for finished art do not have to be made to that for the templates.

图 8.11 模板和成品资源的不同目录树的设置。同样，这两种类型的资源是分开的，但是现在对已完成艺术的目录树所做的更改不必对模板的目录树所做的更改。

Figure 8.12 A setup with templates and finished art assets integrated into a single directory tree: quick to navigate, but running the risk of unwanted template assets being archived with the project files or incorporated into the game build.

图 8.12 将模板和成品艺术资产集成到一个目录树中的设置：快速导航，但存在不需要的模板资产与项目文件一起存档或合并到游戏构建中的风险。

理想情况下，艺术家应该在项目开始时创建所需的所有模板。然而，在实践中，模板是通过生产过程创建的 – 随着时间的推移，模　　板越来越不通用。这些“非真正模板的模板”导致目录树膨胀，使艺术家更难找到实际有用的模板。在创建新模板之前，请考虑实际可以使用多

少资产。如果有疑问，不要创建它！

第 8.10 节文件命名约定：通用语法

接下来，我们将讨论文件命名约定。正如我们已经建立的那样，强大的命名约定对于维护高效的管道至关重要。它们不仅使艺术家更容易找到资产，而且使编码人员能够编写自动定位或修改这些资产的工具。我们在前一章中讨论了这种自动化系统的优缺点。在这里，我们将查看命名约定所基于的语法。

分隔名称元素

建立强大的命名约定的一个重要步骤是确定如何分离构成资产名称的元素。使用文件名来反映钢剑特别大的事实可能很诱人，但是调用资产“swordsteellarge.extension”有点难以解析。

在文件名中使用空格通常是不受欢迎的，原因很简单，虽然空间在 90％的应用程序中都能正常工作，但剩下的 10％（最常见的是脚本）会造成如此大的破坏，以至于这个“便利”不值得麻烦创建。相反，有两种常用方法可以分隔文件名中的元素。一个是使用下划线：

sword_steel_large.extension

下划线很容易解析，但它们会增加资产名称的长度，这可能会导致将文件名限制为特定字符数的软件出现问题。另一种方法是使用大写字母和小写字母的混合，这个系统有时被称为“驼峰案”：

swordSteelLarge.extension

Camel case 不像下划线那样容易解析，但提供了更短的文件名。

大写

另一件需要考虑的事情是文件名是如何大写的。重复文件和文件夹的最常见原因之一是大写字母的使用不一致。当绑定到区分大小写的脚本系统时，这些问题尤其成问题。

避免这种不一致的最简单方法是不使用大写字母。如果无法做到这一点，请建立一个管理其用途的可靠规则集。这些可能包括总是资本化资产的第一个字母（“剑”而不是“剑”），或如上所示的骆驼套管。

复数

不一致的多元化导致了与不一致的大写相同的问题。同样，避免此类问题的最简单方法是使用复数或永远不使用它们。

缩写

缩写通常用于项目中以缩短文件名的元素。这减少了开发人员输入每个名称所需的时间。如果没有缩写，环境中使用的树模型的文件名可能是：

environment_tree_maple_red_large.extension

使用缩写，相同的资产可以命名如下：

env_tree_maple_red_lrg.extension

这可以节省十个字符 - 大约占整个文件名的 30%。

在设计缩写约定时，首先要确定的是文件名称的哪一部分将被缩写。第二个是它可能包含的每个单词将如何缩写：例如，“环境”是缩写为“env”还是“envi”？

这里有一些通用规则：

缩写资产类型的名称通常是好的，例如“环境”，“特殊效果”或“武器”。

通常不应缩写“枫”等特定资产名称。

诸如“红色”或“大”的描述符只有在完全清楚描述符所指的内容时才应缩写。

如果描述符是缩写的，例如当“large”变为“lrg”时，该缩写应该在项目的任何地方使用！

如果在多个项目中使用特定缩写，请尝试确保它们在每个项目上具有相同的含义。 如果资产稍后共享，这有助于避免意外 - 这可能不会在原始项目结束后很久。

一旦建立了将在整个项目中使用的约定，请记录它们！团队中的每个人都必须使用相同的缩写，否则他们的大部分用处将会丢失。然后，内容创建者可以阅读文档以学习，引用或添加已建立的约定结构。

第 8.11 节文件命名约定：在文件名中镜像文件夹结构

在电影中，一种常见且有用的做法是以资产名称镜像文件夹结构。要做到这一点，只需包含“旅行”到达资产所需的每个文件夹名称的名称或缩写。

这会导致文件名如下：

<shot >_< assetType >_< assetName >_< version > .extension

For example:

SB_0001_animationCurves_fooRun_v30.ma

在电影工作中，许多类型的数据阴影图，动画的几何缓存，模拟等都被写出来作为图像序列。该帧信息通常存储在文件名中。大多数设施将它放在文件扩展名的前面：

<shot >_< assetType >_< assetName >_< version > .####.extension

通常的做法是使用四位数来表示帧数。每秒二十四帧，这足以覆盖不到七分钟的镜头，远远超过电影中的平均镜头持续时间。然而，在管道的后期阶段（例如在数字中间体的创建期间），其中胶片作为整体被处理，而不是作为单独的镜头，该惯例变得不合适，并且使用六位填充。

在游戏中使用类似的约定。例如，考虑位于以下　　目录中的文件：

/GameRoot /Assets /Models /Equipment /Weapon /Sword /Rare /

这可以命名为：

eq_wp_sword_rare_demonshard_a.mdl

or:

eqWpSwordRareDemonshardA.mdl

请注意，“GameRoot”，“Assets”和“Model”不包含在资产名称中。由于可以安全地假设每个资产都存储在 GameRoot / Assets 目录中，包括文件名中的目录名称不会提供任何其他信息。包含单词“Model”也是不必要的，因为.mdl 后缀已经将资产定义为模型。保持名称信息，但不要浪费空间与不必要的混乱。

在资产名称中镜像目录结构有几个主要好处。首先，资产的位置和类型可以仅从其文件名中确定，从而为艺术家节省了大量的搜索时间。考虑一个名为的文件：

fx_wpn_sword_glowy_a.extension

这必须是剑的 FX 元素，并驻留在以下目录中的某个位置：

/FX /Weapon /Sword /

镜像文件名中的目录结构对于可以创建使用该信息来加速工作流的工具的程序员也是有益的。例如，工具可以仅根据名称自动保存或导出特定文件夹中的资产，从而缩短导航时间。

最后，镜像资产名称中的目录结构还有助于确保没有两个资产具有相同的文件名。这对于拥有多个艺术家的大型项目来说非常重要，例如，两个独立的艺术家更有可能在两个单独的文件夹中创建名为“stairs_evil”的资产。拥有两个或多个具有相同名称的资产会给所有相关人员带来困惑，使识别和修复错误变得更加困难，并且实际上可以通过“扁平化路径”架构破坏项目，因为他们可能需要每个资产都具有唯一的名称为了正确加载文件。

第 8.12 节版本控制：排他性和非排他性文件存取

我们简要介绍了版本控制系统以及它们在前几章中的工作原理。接下来，我们将更详细地讨论版本控制的一些关键方面。

第一个是文件的独占访问和非独占访问的问题。由于许多人并行处理资产，因此当两个人想要同时修改同一文件时可能会发生冲突。发生这种情况时，有两个选项：授予一个用户修改文件的独占权限，或允许两个用户同时修改文件，然后将其更改合并在一起。

授予文件的独占使用是有限制的，并且意味着一个用户拥有另一个用户。但是，如果文件是二进制格式或文本格式没有考虑合并的结构，它通常是唯一的选择。

如果版本控制系统允许两个用户签出相同的资产，则第一个将其重新签入的用户照常执行。第二个应该收到一个警告，表明资产已经签入，还有三个选项：覆盖第一个用户的工作，放弃他们自己的更改，或者将两组更改合并在一起。由于大多数版本控制系统不理解文件内容的含义，因此合并趋向于逐行工作，并且主要由用户确保结果有效。

第 8.13 节版本控制：分别处理代码和艺术资产

另一个考虑因素是代码和艺术资产是否应该单独处理。在电影中，程序员倾向于使用标准版本控制系统来跟踪代码变化，而艺术家则使用专门用于跟踪资产的自定义系统。由于涉及大量数据，非专业版本控制系统往往不能很好地处理艺术资产。

但是，在游戏中，当涉及到管理变更冲突时，内容与代码没有什么不同。对内容的更改可能看起来不对，播放错误，导致“断言”（游戏运行时显示错误消息，警告您数据中存在错误），或者最糟糕的是，导致运行时崩溃。因此，管道开发人员应该以与程序员相同的方式对待艺术家：他们应该被提供一个沙盒环境来测试内容，并且只有在他们确定不会对其他团队成员产生负面影响时才提交更改，以同样的方式程序员只提交经过测试的代码。

第 8.14 节版本控制：处理特别项目

在处理项目时，可能需要解决特殊要求：例如，为管理人员或营销部门创建电影预告片或演示。这意味着将决定用于创建这些请求的资产的存储位置，以及如何跟踪它们。

处理这些特殊版本的一种方法是通过获取项目的“快照”来“分支”副本：在特定时间点的所有资产的副本。这具有以下优点：快照存在于单独的位置，以便可以在项目本身上继续工作，而不会在特殊版本中引入任何意外更改或错误。

当主项目中的更改需要合并到特殊版本中时，会出现分支的缺点，反之亦然。这些合并可能非常耗时，并且需要谨慎处理，因为可能会引入错误和/或数据丢失。

在游戏中，分支还有一个缺点：错误修复通常需要进行两次。可以通过源代码控制来缓解此问题，但如果在发现错误之前仅在一个分支中更改了文件，则合并会变得复杂。因此，在主要分支通过质量保证并推迟发布之前，让团队在低风险区域工作通常是明智之举。

让我们看看分支如何更详细地工作，首先是电影，然后是游戏。在电影中，分支有两个主要原因。一个是创造一些特别的东西：预告片，DVD 演员，商业广告，营销材料等。第二种是用于生成资产的分支代码：例如，将文件从一种格式转换为另一种格式，或者处理它们以用于其他媒体，例如游戏或印刷营销。

分支镜头时，通常只是从原始文件复制目录结构并将特殊字符添加到名称上。例如，请拍摄以下照片：

SB_0001_animationCurves_fooRun_v30.ma

通过在文件名上标记“A”，分支文件变为：

SBA_0001_animationCurves_fooRun_v30.ma .

用于处理资产的分支代码以与游戏中类似的方式完成：新代码在源代码存储库中用项目名称或名称标记，然后签出到项目中。

在游戏中，分支的一个常见用途是创建一个演示。理想情况下，演示应该使用游戏构建，可能通过提高公共消费水平。然而，这并不总是可行的，特别是在项目的早期阶段，或者在进行彻底的改变时：如果设计师后来决定改变角色的跳跃高度，那么花时间进行修饰的水平可能变得无法实现。

将演示分成单独的分支的优点是它确保完整游戏中引入的错误不会影响它。缺点是演示所需的任何更改都必须集成到两个版本中，从而增加完成任务所需的时间。

分支是正在进行的项目的一个重要问题，特别是那些在发布后很久就更新的项目，例如 MMO（大型多人在线游戏）。对于 MMO，通常会在发布更新之前将产品按月分支到 QA 构建。这种做法允许团队在测试隔离更新时安全地继续处理新内容，从而保护更新免受新内容可能无意中创建的任何错误的影响。

第 8.15 节元数据：嵌入数据与提取数据

正如我们在前一章中所讨论的那样，管道生成了大量元数据：存储在文件中的“额外”数据以及构成资产本身的信息。这可以来自许多来源：用户输入（用户通过在一个软件的 UI 中填写表单生成的信息），文件系统（文件大小，访问时间等），文件本身（像 EXR 和 DPX 这样的文件类型包括嵌入在文件头中的信息，例如创建文件时的信息，以及管道中的其他进程（例如，在渲染时生成的信息，例如最大内存使用，CPU 负载和 I / O）。

这里有两个要点需要考虑。首先，您需要确定哪些信息对您有用。我们在前一章中看过这个问题。其次，您需要确定它们是否以当前格式对您有用，或者是否需要将它们记录在其他位置。

这里有两个要点需要考虑。首先，您需要确定哪些信息对您有用。我们在前一章中看过这个问题。其次，您需要确定它们是否以当前格式对您有用，或者是否需要将它们记录在其他位置。

定义存储元数据的最佳位置取决于您需要支持的工作流程。通常，如果多个应用程序需要元数据，或者进程需要快速访问某条信息，则最好从资产文件中提取元数据并单独存储。

例如，假设 Maya 场景文件中的对象数是另一个进程的有用数据。要提取此数据，另一个进程必须读取 Maya 文件（不是一个简单的任务）并快速读取（对于大型场景不太可能）。更好的解决方案是在创建资产时提取此数据并单独存储。

在资产外部存储元数据有两种主要方法：作为文件系统上的单独文件（通常是 XML，YAML 或 JSON 等格式的纯文本文件），或者存储在数据库中。重要的是要注意资产本身通常不存储在数据库中：只有元数据。数据库只是在现有目录结构和命名约定之上形成一个额外的层，并包含指向文件路径形状的实际资产的“指针”。

这两种方法应视为互补：您采用哪种方法取决于相关用例。我们将在下一节中介绍每种方法的优缺点。

第 8.16 节元数据：平面文件与数据库

存储元数据的最简单方法是在文件系统中作为平面文件。但是，这种方法通常会导致许多小文件（每个资产的每个版本一个，每个只有几兆字节或更少），或者很少有非常大的文件（每个资产一个，或每个项目一个，每个都运行到几千兆字节）。两种结果都没有顺利扩展。通常，文件系统在处理大量小文件方面很差，元数据的分散使得查询运行缓慢。相反，包含许多资产信息的文件存在较高的损坏风险，并且在并发编辑期间存在较高的争用风险。

数据库解决了通过文件管理元数据的问题。数据库旨在存储大量数据，并提供快速查询此数据的机制。

这意味着它们比平面文件系统更有效地处理查询，特别是包含许多资产的查询。

数据库的另一个优点是它们可用于为不可能违反的数据创建验证规则。例如，您可以将“资产类型”等字段的值限制为一组已知的单词，如“字符”和“效果”。如果操作正确，这有助于保持资产管理系统中数据的完整性。

设置数据库还使具有多个办公室的设施能够更轻松地在它们之间共享或复制数据：基于文件的系统通常无法很好地扩展到多站点设置。近年来，在没有停机的情况下，为“热”备份和更新做出了很多努力：这是一项重要的要求，因为数据库通常在不断使用。

最后，数据库使公司更容易组建或减少工作人员。用户数量的这些变化可以快速改变访问元数据的方式。数据库可以通过诸如“集群”（在一组联网机器上运行数据库软件），“主从复制”（维护数据库的规范“主”副本和复制“从属”等技术来处理这一问题。 “可以承担部分负载的其他计算机上的副本”和“分片”（将数据库中的记录拆分为多个独立的分区，然后可以单独查询）。

这一切听起来都很棒。那你为什么不想使用数据库呢？首先，因为数据库比磁盘上的文本文件复杂得多，并且这种复杂性带来了巨大的成本。对于不需要聚合的简单数据或二进制数据，文件比数据库更快，更容易使用。

第二个原因是数据库与文件系统本身一起引入了另一个故障点。如果数据库出现故障或损坏，您可能会失去对部分或全部资产及其元数据的访问权限，这可能会导致生产停止事件。通过良好的基础架构支持，数据库可以变得非常可靠，但是这种基础设施并不便宜，维护和监控将是一项持续的成本。如果您的设施是全球性的，您可能还需要花费较长的网络传输时间来考虑数据

库访问。

第三，有效地使用数据库就像其他任何技能一样，需要专业知识来构建数据库查询，以便快速有效地返回结果。如果通才会编写资产管理系统的重要部分，则需要确保至少有一位数据库专家密切关注事物。

您还需要提供一个接口，以适当的方式将信息输入或输出数据库 - 通常是用户的 UI 和开发人员的 API。这些接口必须由管道团队提供，但请记住，虽然一些商业工具提供了数据的通用视图，但它们缺乏非技术艺术家和管理层所需的用户友好功能。

同样重要的是，数据库中的数据是准确和最新的，并且由于数据库不是无所不知，这通常是管道团队的责任。重要的是在开始时定义数据库是什么，不希望知道什么，以及设施中的其他工具可以依赖于它存储的数据的强烈程度。例如，在游戏中，通常使用数据库来记录每个单独资产的内存占用量。从 DCC 工具导出资产，签入版本控制时，甚至通过显式手动更新过程时，可能会更新此值。在任何这些情况下，如果数据库与版本控制系统不同步，则用户可能正在处理与数据库中记录的内存占用量不同的资产版本。在许多情况下，数据库是否滞后于版本控制系统几个小时无关紧要，反之亦然。在其他情况下，有必要设计更精细的更新机制。

最后，在某些情况下，单个元数据文件可以提供积极的好处。在视觉效果中，让渲染场上运行的进程访问数据库通常是不受欢迎的，因为同时从这么多机器处理请求会使数据库停滞不前。提供外部元数据可以避免此问题。

元数据文件还可以为具有许多不构成单个文件的依赖项的复杂资产提供代理：例如，由几何，照明数据，脚本和其他信息的集合组成的游戏级别。在一个可以通过版本控制的文件中捕获所有这些输入，可以更加轻松地准确跟踪项目的进度。

第 8.17 节数据库：关系数据库和非关系数据库

我们将通过研究如何为生产工作选择数据库结构来完成本章。在我们这样做之前，让我们来看看一些基本的技术概念。这只是对一个更大的主题的一个非常简短的概述，所以如果你打算使用数据库，你应该自己阅读这个主题。

有两种基本类型的数据库。使用诸如 SQL 之类的语言创建的“关系数据库”使用由严格规则组织的高度结构化的信息集。其他数据库（如 MongoDB 或 CouchDB（有时组合为“NoSQL 数据库”））使用结构较少的存储方法以获得更大的灵活性。我们将在后面详细说明这些术语的含义。但首先，让我们看一下关系数据库如何存储信息。

在这样的数据库中，数据被分组为“表格”，类似于 Excel 等软件中的工作表。与 Excel 工作表一样，表格分为“列”和“行”。每列（或“字段”）表示要捕获的特定数据。该列有两个主要属性，可以存储在列中的名称和数据类型（例如，整数，浮点数或字母数字字符串）。每行（或“记录”）表示要记录的数据集合。每行在每列中放置一个值，符合为该列指定的数据类型。与在电子表格中一样，可以在表中插入行或从表中删除行，并更新行中的各个值。

例如，我们可能有一个表格，其中包含标题为“描述”的列（每个任务的描述：一个字母数字字符串），“开始”（任务的开始日期：日期），“艺术家”（指定的艺术家）任务：另一个字符串）和“艺术家 ID”（唯一标识符：整数）。该表中的每一行都代表一个任务，并为我们定义的四个字段中的每一个提供数据。您可以在图 8.13 中看到它的工作原理。

数据库的“模式”定义了如何将数据划分为表，包含每个表的列以及数据结构的其他方面。

从理论上讲，架构可以定义一个包含大量列的单个表，每个列对应于您要记录的每种数据类型。但是，这将是一个非常低效的结构，导致大量重复的信息。这可以通过“规范化”数据库来减少：以最小化冗余的方式组织数据库。这通常涉及将较大的表分成较小的表，并定义它们之间的关系。

Figure 8.13 A table from a database, storing information on the tasks that make up a project. It consists of four columns, each one storing different types of data about the tasks in question, and two rows, each one representing an individual task.

图 8.13 数据库中的一个表，存储关于组成项目的任务的信息它由四列组成，每列存储有关任务的不同类型的数据，还有两行，每行代表一个单独的任务。

例如，假设我们有另一个表，如图 8.14 所示，存储有关艺术资产的信息。列定义了作业，场景和镜头，它显示的是什么类型的资产，资产的名称以及它是哪个版本。

Asset					
Job	Scene	Shot	Asset Type	Asset Name	Version
Job1	Scene1	Shot1	Model	Joy	1
Job1	Scene1	Shot1	Model	Joy	2

Figure 8.14 Another table from the database, storing key information about each asset used in production, including its type and version number, and in which job, scene, and shot it appears.

图 8.14 数据库中的另一个表，其中存储了生产中使用的每个资产的关键信息，包括其类型和版本号，以及在哪个作业、场景和快照中显示的信息。

重命名单个作业，场景或镜头的操作将相当昂贵，因为这意味着扫描整个表以识别需要更新的行。但是，如果作业，场景和镜头存储在它们自己的表中并且使用唯一 ID 引用此数据，则此类更新只需要更改相应表的相应行，因为 ID 是自动生成的数字，并且永远不会改变，即使其他细节也是如此。

正是这种将数据分成多个表并定义它们之间关系的行为为我们提供了“关系数据库”这一术语。在这种简单的情况下，可以简单地使用唯一 ID 号作为“键”来定义关系。被引用的表中的列（在本例中为 Job，Scene 或 Shot 表）包含唯一标识符，称为“主键”。引用它的其他表中的相应列称为“外键”。主键和外键值匹配的记录被称为“相关”。

为了了解这是如何工作的，让我们说我们希望我们的 Asset 表存储有关负责创建资产的艺术家的信息。此信息可以通过多种方式表示：名字，姓氏，电子邮件地址，桌面号码等。

但是，我们的任务表目前还在其 Artist 列中存储了一些此类信息。我们不想复制此信息，因为这会使数据库难以维护。相反，我们只将相关艺术家的艺术家 ID 值存储为外键。在图 8.15 中，Artist ID 列也已添加到资产表中，Asset 和 Task 表现在引用我们（新创建的）Artist 表，其中“Artist ID”作为主键并存储单个规范艺术家信息的副本。

SQL（结构化查询语言）是一种通常用于管理关系数据库系统中的数据的编程语言。虽然严格来说，SQL 在许多细节上偏离关系模型，但使用它的数据库系统（包括 Oracle，DB2，MySQL，Postgres 和 SQLite）通常被描述为“关系型”。

关系模型还有许多替代方案，包括“文档”，“对象”和“图形”模型。使用此类模型的数据库通常称为“NoSQL”数据库。

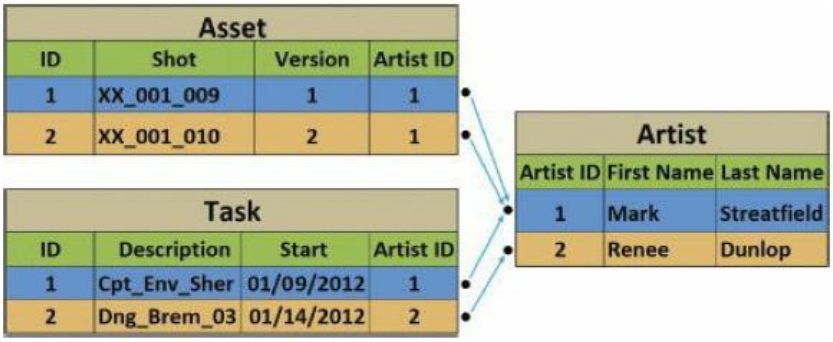


Figure 8.15 A relational database structure. The Artist ID column in the Asset and Task tables links them to a separate Artist table storing information about the artist responsible. This is more efficient than duplicating the information in both the Asset and Task tables.

图 8.15 关系数据库结构。“资源”和“任务”表中的“艺术家 ID”列将它们链接到一个单独的艺术家表，该表存储有关负责的艺术家的信息这比复制资产表和任务表中的信息更有效。

文档数据库对关系数据库使用类似的术语：这里，表是“集合”，行是“文档”，列是“字段”。但是，它们具有更开放和灵活的结构：与关系数据库不同，并非每个文档都需要包含相同的字段集。这有时被称为“动态模式”。

SQL 和 NoSQL 数据库都有其优缺点。 SQL 数据库提供了强大的工具，可确保存储在其中的数据可靠且一致：例如，可以使艺术家的每个记录都具有其电子邮件地址的字段，或者使纹理的每个记录都具有文件格式字段。数据结构紧凑的事实也使得执行复杂查询变得容易，例如：“查找在 Shot 145 中使用的超过 6 MB 的所有资产，并且具有三个以上的纹理。”但是，它们的设置和维护相对复杂，而且不灵活：添加新类型的信息需要将数据库重建为新模式。

相比之下，NoSQL 数据库更加灵活。它们也是可扩展的：它们可以“并行”到大量服务器上，以提高响应速度和存储容量。但是，它们存储的数据的完整性无法保证，因此使用数据库的代码必须始终能够处理丢失或格式错误的信息。

第 8.18 节数据库：选择数据库结构

您选择哪种类型的数据库取决于您需要的任务。对于资产管理，文档数据库可能更适合，因为其结构较少的方法意味着可以通过包含不同字段的文档集合来描述单组资产，而不会对关系模型产生一些性能损失。

如果您确实使用了关系结构，那么一个好的指南是为您需要跟踪的每个事物设置一个表：例如，场景，镜头，任务，用户或资产。然后，每个表都应包含指定记录该“事物”所需的每条信息的列。在数据库模式中模仿数据的本质使开发人员（可能还有您的用户）更直观地构建查询以检索所需的信息。

但是，构建数据库有点像选择深层或平面目录结构，特别是在涉及如何记录资产的问题时。使用太多单独的表可能会使数据库难以维护。相反，可以只使用一个表来存储每个资产的信息。但是，这意味着要么只存储适用于每个资产的列，要么创建一个存储您要使用的每个列的宽表，但是其中每个列都是稀疏填充的（也就是说，并非每行都包含每列的值）。

实际上，您选择的结构将是最佳实践（规范化过程：最小化表之间的冗余和依赖性）以及实际用途（非规范化：故意添加冗余数据以提高读取性能）之间的平衡。大多数数据库系统都提供了更容易实现这种平衡的功能。

插曲：元数据

第 8 节插曲 1：什么是元数据？

无论何时创建数字文件，保存在系统中，搜索，转换，编辑，使用或交付，元数据都是关键部分。元数据是任何生产流程或交付流程的基本组成部分，无论是游戏，电影还是视觉效果。它用于描述，控制，构造或理解它所归属的内容。元数据本质上是“关于数据的数据”。没有它，我们的产品将以各种可能的方式受到影响，包括延长生产周期，降低安全性，甚至减少创意表达。

虽然它可能并不明显，但我们每天都使用元数据而不是真正考虑它。要了解元数据是什么，它与内容的本质有何不同，以及它的内在价值，请考虑一个简单的比喻，用一罐汤。在杂货店购物时，想要的不是金属罐或标签，而是汤本身。汤（内容）是我们真正想要的，但标签（关于汤的信息）用于帮助确定关于汤的相关细节，例如成分，重量，营养价值和有效期。标签上的这些信息位相当于您内容的元数据。如果要移除标签并且购物者只能浏览一排希望使用正确类型的银罐，结果将是大规模混淆，最终导致很少出售罐头。质量元数据可为流程带来信心。

在考虑数字资产（如电影剪辑，背景板，音频文件或图形）的元数据时，创建和维护质量元数据至关重要。就像保留汤罐标签对于维持质量标准至关重要一样，维护内容文件的元数据也是如此，以优化其使用和控制。

一些元数据由资产本身携带，并且可以由系统或应用程序读取。此类元数据的示例可以是资产的文件大小或创建日期。归因于资产的其他元数据（取决于文件类型和用于管理它的系统）只能链接到资产并存储在数据库中。视频可能有一个名为 John Smith 的导演，但视频文件不带有此自定义值。必须通过使用元数据管理系统（例如 DAM（数字资产管理））或将内容文件链接到其相关的可自定义元数据字段和值的应用程序来通过资产引用该值。

有人可能会争辩说，导演的名字可以用在数字资产的命名惯例中来提供这些信息。但是，这是一种短视的解决方法，用户在处理共享文件夹中的文件时经常采用这种解决方法，而没有使用元数据管理媒体的任

摄影和大多数创意文档（如图形，布局和打印材料）都可以使用多个嵌入式元数据模式模型。这些类型的文件中的元数据使用标准模式，例如 EXIF（可交换图像文件），IPTC（国际新闻电信委员会）或 XMP（可扩展元数据平台）。这些元数据值要么硬编码到文件中，要么允许用户创建，编辑或删除所携带的值。只要在支持该架构的系统或应用程序中查看或编辑文件，就会维护并使用元数据。

视频文件通常更依赖于专有文件类型，应用程序和包装器（例如 MXF，代表材料交换格式）来维护一些有限的嵌入式元数据。确定一个人所需的资产类型以及如何将元数据链接或嵌入到给定类型中是开始设计元数据模式的好地方。

通常向应用程序内的用户提供元数据模式或模型结构，以支持其管理的资产类型或工作流。其他系统允许完整的独立和可定制的模式。其他人使用基于应用程序和可定制模式的混合。通常，元数据模式分为五类：

描述的

技术

权

档案

结构

描述性元数据是用于描述资产的预期信息。在 405 高速公路上骑蓝色摩托车的金发女子的视频剪辑需要通过搜索找到它的良好描述：“一位金发女子骑着她的蓝色摩托车在 Mulholland Drive 附近的 405 上。”此类型的其他元数据包括关键字（摩托车，女人，金发，蓝色，高速公路，加利福尼亚，405，山脉，穆赫兰道）。其他描述性元数据将包括诸如标题，主题，项目 ID 等字段。这些字段通常用于搜索和检索。

技术元数据提供有关通常不可编辑的资产的信息。这些是关于资产的嵌入式方面，例如比特率，文件扩展名/ mime 类型，分辨率，宽高比等。这些在确定邮政或分发中特定用途的正确格式时非常有用。

权利元数据处理资产的控制和使用。资产的所有者，到期日期和合同 ID 等元数据是限制资产的重要字段。

归档元数据用于资产的长期保存和维护。资产将具有无限期的生命周期，在其创建时可能与其利益相关者高度相关且重要。随着时间的推移，这种相关性可能会减弱，但其重要性可能会根据情况而消退。可能需要以特定格式保存资产以将其保存在存档介质上。多年来，它可能需要以一定的间隔“重新保存”到另一种格式。存档元数据通知资产托管人如何在整个生命周期内维护资产。

结构元数据用于指示资产在其他更复杂的对象（如书籍，布局或效果）中的位置。一个例子是放在另一层后面的图形。在这种情况下，元数据的存在是为了使一个资产相对于另一个资产。

从战略上讲，为了减少自动化和断开信息孤岛的数量，鼓励组织开发所有系统和用户都可以订阅的整体元数据模型。然后，在各种应用程序和平台中使用该整体结构的子集以用于一致的词汇表和信息数据源。在内部或外部时，需要集成元数据，这种通用模式允许在系统之间进行更简单的数据映射。一个常见的错误是每个部门都要创建自己的元数据模式。随着时间的推移，这些孤岛发展成为部门“机构”，这些机构难以集成并且成本高昂。

组织通常会转向 PRISM，都柏林核心等元数据标准，或者用于标准图像许可，Plus Coalition。这些标准非常适合某些工作流程和行业，但不提供，也不打算提供完整的组织元数据模型。组织可以发现这些有助于集成并符合行业标准，但不应将它们视为整个模型。同样，应用程序供应商尝试创建开放标准或框架，如 MXF，Quicktime，AAF（高级创作格式）等，以帮助集成元数据和内容文件。

即使接受相同内容文件和嵌入式元数据的系统也需要更高级的集成。由于没有可以满足每个业务操作或生产管道的标准模式，元数据通常需要导入映射过程。当一个采用各种元数据字段和值的系统必须将这些值移植或迁移到具有相似或完全不同的元数据字段的另一个系统时，需要进行映射。这可能发生在希望共享元数据的部门或单独公司之间。

例如，假设公司 A 将生产 ID 称为“作业”，而公司 B 将其称为“项目”，但每个都需要共享信息。工作和项目对每个工作意味着同样的事情。如果公司 A 向公司 B 发送元数据记录，则接收公司将无法将值导入其系统。接收公司的系统希望找到“项目”，但收到“工作”。

为了解决这个问题，支持诸如 XML（可扩展标记语言）或 API（应用程序编程接口）之类的 Web 服务的类似甚至不同的系统可以连接和共享它们的元数据。 XML 是一种开放标准文档，用于传输和存储元数据，客户列表或任何其他信息等数据。 XML 可由机器和人类读取，并提供强大的元数据共享方式。希望使用 XML 作为元数据共享过程的用户必须遵守接收系统中格式化，命名，区分大小写和导入配置规则的严格规则。

在维护多个平台或数据存储库的大型组织中，部署了元数据管理系统。元数据管理系统允许用户捕获和管理标准业务术语和结构，包括所有断开连接的存储库中的词汇表，首选术语和对象关系，同时保持通用平台。然后，用户可以搜索和检索来自所有元数据存储库的结果，以获得最大可见性，而无需通过 XML 进当在完全不同的业务单位（如新闻档案，照片库和营销材料）中搜索媒体资产时，这尤其有用。

鼓励组织避免随便命名的元数据字段，以减少集成项目的问题。这是通过在可能的情况下从面向业务的策略中标准化元数据约定来实现的。一旦业务需求明确，这些策略就会产生技术要求，而不是技术上决定业务需求。通过彻底了解管理和控制生产过程中资产的业务需求，可以获得质量元数据（技术，权利等）。通常，生产过程的利益相关者是元数据字段及其与资产的关系的最佳来源。业务经理，主题专家和数字工作流专家可以共同合作，从主要工作流程，数字资产类型和资产生命周期中开发模式。

资产质量受其在整个生命周期中培养和维护的程度的影响很大。部门或组织在资产完成后开始输入元数据或由用户输入元数据而对资产的价值或相关性没有真正了解的情况并不少见。这将产生极其有限和低价值的元数据。为了提高元数据的质量，尽可能在其生命周期的早期开始进入是非常重要的。例如，相机制造商提供地理位置坐标以及稍后在其他生产阶段中使用的其他信息。随着开发资产元素的特定授权用户或最终资产的各种版本，可以输入少量元数据。在生产和发布过程中，元数据经过改进，清理和更新，从而实现高度可靠性。元数据的低可靠性将降低对系统的信任。

元数据所有权也是维持元数据质量的关键。维护其元数据的组织已实施元数据所有权团队或联合跨职能职责。如果没有这些，部门和组织将对元数据和整个过程缺乏信心。这些团队还可以作为一个治理实体，在业务需求或资产类型发展时维护和改进元数据。

组织和部门内的业务经理应与其技术经理密切合作，以创建将采用，依赖和扩展元数据框架的平台。如果没有这些元数据的基本原则，资产就会丢失，不受控制，并对已经受到压力的生产周期造成严重破坏。

资产管理

第 9.1 节 您将从本章中学到什么

我们在前几章中谈到了资产管理的重要性。在本章中，我们将探讨资产管理系统本身的本质。我们将概述此类系统的组件，检查成功的资产管理系统应满足的目标，然后讨论与系统设计相关的关键技术主题。有些版本控制和元数据已经在前面的章节中讨论过了，这里只简单介绍一下。其他的，如依赖性跟踪，是新的，将详细讨论。

第 9.2 节 什么是资产管理？

资产管理是任何生产的支柱 – 但不是一个易于定义的术语。它包含许多任务，包括资产类型的分类，跟踪数字和物理资产，版本控制以及自动化和控制数据流的过程。

我们在前面的章节中研究了其中的一些任务，特别是版本控制。值得注意的是，资产管理和版本控制并不相同：相反，资产管理通常在版本控制之上实现，提供元数据提供的搜索和分析功能的强大组合以及在“垂直”内浏览的选项“由版本控制系统提供的项目片段。

任何资产管理系统都有两个关键组件：

数据模型描述了资产管理系统管理的数据是如何构建的。这必须通过某种形式的 API（应用程序编程接口）提供，开发人员可以通过它创建工具并定义工具使用的自定义数据类型。数据模型是定义资产类型并实现版本控制和依赖关系跟踪的地方。

存储库是用于存储数据的基础结构：硬件和软件，如数据库系统。它必须使用户能够访问数据模型中包含的元数据和资产本身。存储库应该能够处理生产的高峰和低谷，从少数用户和少

量资产的项目扩展到数百个用户和数百万资产，同时保持一致的响应时间。

正如这个细分所暗示的那样，资产管理系统不仅仅是纯软件。虽然商业资产管理工具确实存在，但它们并非一刀切的解决方案。目前，还没有单一的现成产品可以处理上面列出的所有任务。

第 9.3 节资产管理的目标

在我们研究如何设计数据模型之前，让我们回顾一下设计必须达到的目标。要取得成功，资产管理系统必须符合许多标准。我们将在这里看一些最重要的内容。

可扩展性

正如我们所看到的，电影和游戏会产生大量数据。对于视觉效果项目，个别学科可以生成数 TB 的数据，而整个节目可以达到数 PB。资产管理系统必须能够处理总数据量和峰值吞吐量时生成的用户请求数。

此外，电影和游戏需要大量的人力投入。一个 6 到 9 个月的 VFX 项目可以代表 10 到 2 万人日的劳动力。由于设施通常需要工作人员或船员来满足生产中的高峰和低谷，因此资产管理系统必须能够应对访问它的用户数量的变化。由于大多数大型设施在每个项目和每个部门的基础上工作，因此将这项任务略微简化：艺术家被分配到一个单独的项目，并孤立地进行工作;并使用一套一致的工具在一个部门内工作。但是，仍然有必要设计资产管理系统，使其能够适应长期的长期增长。

对可扩展性的需求也扩展到系统将管理的资产类型范围。有必要考虑是否将通过系统内置的配置机制添加新资产类型，或者编写代码。如果资产管理系统是用面向对象的编程语言编写的 - 它倾向于扩展描述系统内所有资产的核心功能的“基础”资产 - 这通常会导致每种资产类型一个类。在代码中定义每种资产类型还意味着可以更轻松地引入新功能，而不会影响系统中的其他资产。

除了能够容纳新的资产类型之外，系统还应具有筛选出不符合现有类别的资产的能力，或者将其标记为特殊处理。

并行访问

生产计划不断缩短。在过去，主要电影和游戏通常分配两年或三年的时间表。今天，一些游戏特许经营权每年更新，而视觉效果项目可能需要在几个月内完成。因此，让每个部门尽快工作至关重要 - 为此，艺术家需要能够并行访问资产。

当我们讨论资产创建过程时，我们探讨了其中的一些问题（例如，需要将粗略的代理模型传递给照明部门，以便在资产完成之前开始拍摄工作）。为了防止艺术家并行工作以覆盖或使彼此无效的变更，设施通常使用版本控制系统。我们再次在第 5 章介绍了版本控制的基本概念，因此我们不再在这里讨论它们。然而，版本控制系统面临的另一个问题是，在两位艺术家参与其中之后，难以合并资产的单独版本。标题为“管理不可合并文件”的标注更详细地探讨了这个问题。

管理不可合并的文件

许多项目（尤其是游戏项目）遇到了一个问题，即单个文件包含需要由多个团队成员更新的信息。当两个团队成员并行地对文件进行更改时，将其重新签入版本控制系统时有两种选择：合并两组更改，或丢弃其中的一组更改并迫使艺术家从重新开始。另一个已更新的资产版本。

合并文件是一个非常容易出错的过程。但是使文件不可合并可能会更糟。对于无法合并的共享文件，我最糟糕的经历是使用特别笨拙的管道来构建角色能力。由于该系统是专为非技术人员设计的，因此该信息存储在 Excel 电子表格中，因此我们将其称为 CharacterType_Ability.xls，每个字符类别或怪物类型对应一个。

每当设计师想要创建一项技能时，他们都会在电子表格中填写各种属性，例如施法时间和伤害。那么动画将分配正确的动画技巧，和 FX 艺术家将分配正确的 FX 铸造，由效果被击中，等等。每个 CharacterType_Ability.xls 文档最多可以包含一百项技能。

问题在于，设计师经常会在同一 CharacterType_Ability 中从事多种技能的工作。 xls 文档，但无法将其检入完成的版本控制系统中，因为这将破坏其他所有人的构建。动画师和 FX

试图在游戏中测试其资产的艺术家将被迫等待该文档。更让人痛苦的是，一个团队可能正在迭代旧技能，而另一个团队正在建立新文件-都需要访问相同的文件。

解决不可合并文件问题的最简单解决方案不是首先创建它们。当大多数计划工作可以在项目开始时完成时，这会更容易。

但是，即使那样，也始终无法避免共享文件。也许您正在使用将它们引入管道的游戏引擎或中间件。也许您正在使用始终生成它们的旧式管道。在这种情况下，您需要决定是尝试删除共享文件还是将它们一起使用是更好的选择。

做出此决定时，请权衡相对成本。将生成多少个共享文件？每个人浪费多少工时？随着公司的成长，这种变化会改变吗？相反，每个人要花费多少工时？

如果删除共享文件的成本超过了使用它们的成本，则您至少可以尝试将其影响降至最低。一种方法是将每个共享文件拆分为较小的，可单独编辑的子文件的层次结构：例如，将包含游戏中每个角色的动画列表的共享文件拆分为与每个游戏角色相对应的单独文件。这减少了两个艺术家需要同时处理同一文件的机会。但是，这很快就会变得很麻烦：在上面的示例中，每次添加新角色时都需要创建相应的动画列表。

共享的编辑工具是另一个探索情况的途径。假设几个团队成员需要调整同一文件中的值：例如，环境光颜色和不同游戏区域的强度。一个使他们可以同时编辑这些值的工具，每个工具都在文件的单个中央版本上工作，可以避免共享文件通常会产生的问题。但是，此类共享编辑工具可能会破坏沙盒环境的隔离性，从而使在其中进行的所有测试无效。共享的编辑工具可以为解决大问题提供强大的解决方案，但始终要考虑它们对测试的影响。

第三种选择是将数据切换为可以使用现有工具自动或至少半自动合并数据的格式。例如，如果可以在适当格式的文本文件中设置角色的能力，则可以使用文本合并版本控制软件具有合并来自多个用户的更改的功能。

访问控制

后勤和安全方面的考虑要求并非每个艺术家都应该能够访问制作中的每个资产。在最低级别，此类权限由文件系统处理，但资产管理系统可能还需要了解用户的访问权限 – 例如，提供

批准过程或发布机制，艺术家只能看到新资产标记为准备好接收时。一个好的资产管理系统还将使用户能够锁定签出的文件，以防止其他用户开始处理相同的，不可合并的文件。

知道已删除或已归档的文件

资产管理系统通常也知道每个文件在网络中的位置之外的状态：特别是它是否已被完全删除。如果设施使用离线存储，那么良好的资产管理系统也应该能够告诉您资产何时被移动到磁带，磁带编号是什么以及当前可以找到该磁带的位置。

自动化性

任何需要艺术家反复执行相同任务的操作都是通过资产管理系统实现自动化的主要选择。在 VFX 工作中，一个典型的例子是动画。每当动画师发布新动画曲线时，都应生成几何缓存。不是让动画师手动创建每个缓存，而是在计算完成之前占用他们的工作站，每当签入一组新的动画曲线时，资产管理系统会自动将作业发送到渲染场。任何缓慢的，CPU 密集型，易于分发的任务都可以以这种方式自动化。

游戏管道还广泛利用自动化进行测试 - 例如，报告崩溃，帧速率和内存使用情况，同时运行自动浸泡和烟雾测试 - 以及组装生产过程中所需的众多不同构建。这些构建对应于每个目标平台的每个可能的代码配置（调试，发布，提交等），以及每个 SKU 或“Stock-Keeping Unit” - 可供销售的游戏的不同配置，包括收集器的版本和特定地区的版本。旧的艺术家工作站通常被回收为“autobuilders”：致力于此任务的机器。

透明度

许多艺术家认为资产管理会在他们的工作流程中引入障碍，或迫使他们以某种方式工作，因此最好在可能的情况下将系统的复杂性隐藏起来。任何资产管理系统的最终目标是使其用户完全透明，以便人们只谈论保存和打开文件，就好像他们使用应用程序中的标准“保存并打开”对话框一样。

要成功与主机应用程序集成，资产管理系统必须为艺术家，导入程序和导出程序提供用户界面，以便以适当的位置和格式加载和保存文件。这引出了我们在前面章节中提到的一个问题：以应用程序自己的本机格式保存文件，还是将其转换为与应用程序无关的格式（如 Alembic）更好？转换可能导致数据丢失 - 例如，模型的构建历史 - 但确保资产可以由使用不同软件的其他部门的艺术家加载。另一个选项，虽然是一个更复杂且存储量更大的选项，但要做到这两点：以应用程序的本机格式发布资产，但要创建模型的内部数据表示，以便在发布时与其他应用程序一起使用。在游戏开发中，由于数据以仅由特定游戏引擎使用的格式存储，或者甚至是在项目过程中经常演变的特定游戏格式，因此这项任务变得更加困难。

与宿主应用程序的集成应该是上下文相关的：也就是说，它应该使用与该主机惯用的构造。（例如，在视觉效果工作中，资产管理集成到 Maya 和 Houdini 中的方式应该反映其各自的工作流程和 UI 约定。）它也应该是可编写脚本的，特别是在更复杂的工作流程中，因此并不总是需要艺术家手动控制过程的每个部分。

阿凡达上的资产管理

在《阿凡达》上的工作中，我们学到了很多有关跟踪镜头各个组成部分的知识。开始时，我们只是在跟踪基本拍摄，然后将它们演变为选择进行编辑的镜头，但最后，我们可以跟踪每棵树，云朵和角色的位置。我们还开始跟踪每个移动资产的动画数据，以便我们可以用新的定时重建场景并将其渲染出来以可视化镜头的变化。

无需人工打开 3D 软件包，就可以完成所有这些设置和执行。进行此类分钟跟踪的最关键，最强大的原因之一是场景构建和实时可视化的形式。众所周知，阿凡达的“虚拟相机”依赖于实时渲染的场景。 Lightstorm 的团队将构建代表潘多拉魔幻世界的身临其境的环境，并用电影的字符。然后，我们使用运动捕捉来记录“摄影机”对象在现实世界中的运动，并将该数据映射到 3D 包中的视口中，然后保存：一个虚拟摄影机，通过它可以查看数字环境。对我们来说，艰巨的任务是没有为实时渲染和内容创建的组合构建 DCC 程序包。传统上，美术师必须在一个程序包中创建数据，然后将其移入游戏引擎以使其处于最佳状态。参与阿凡达（Avatar）的艺术家必须构建实时运行的场景，但仍存在于可编辑环境中。这是进行这种强迫性资产管理的主要动机之一。通过构建高分辨率模型和一系列较低分辨率的变体，我们可以自动选择最合适的版本来重建镜头。我们的实时场景都是用于最终渲染中的英雄模型的低分辨率代理。结果是拍摄舞台变成了导演詹姆斯·卡梅隆（James Cameron）的巨大沙盒。他能够在拍摄时移动环境，改变动画时间，以及改变照明和效果。最重要的是，所有这些决定都反映在高分辨率文件中。如果他在舞台上放了一棵树，当我们六个月后看到最终的渲染图时，那棵树（或分辨率更高的版本）将处于完全相同的位置。这使导演，制作设计师和其他主要创意人员可以反复考虑想法

很快做出明智的决定，他们知道该决定会通过生产。只需查看著名的“ Hometree Destruction”序列即可了解其代表的功能。在正常情况下，导演和各种 VFX，CG 和动画主管的工作要花几个月的时间来计划和执行这样的序列。在 Pandora 上，我们能够获得每个角色，爆炸，特别是坠落的时间。

甚至在将镜头发送到 VFX 房屋之前，正确的大树。 Avatar 带来了许多用于数字电影制作的新方法和工具，但是制作中真正鲜为人知的英雄之一是资产管理系统，该系统使我们能够按照自己的方式工作。

第 9.4 节资产管理在电影和游戏之间的区别

在总体上回顾了资产管理系统的目标之后，让我们来看看这些目标在电影和游戏之间的相对重要性。电影和资产管理游戏的资产管理是两个截然不同的问题。首先，所管理数据的性质和数量差异很大，支持它所需的基础设施的性质也是如此。其次，版本控制和依赖性跟踪的相对重要性各不相同。我们将依次研究这些问题。

正如我们在第 6 章中讨论的那样，电影制作通常会产生比游戏更多的数据。对于视觉效果制作尤其如此，即使在非常高的分辨率下，作品也必须看起来非常逼真，因此必须创建非常详细的资源。电影还会生成大量的大图像序列：背景板和渲染的 CG 元素都可以进行合成。此外，VFX 工作中常见的紧凑周转时间意味着许多用户请求必须同时由资产管理系统处理。

除了能够处理大量数据和用户请求数量之外，资产管理系统还必须能够与设施的基础设施连接。这包括各种硬件工作站，核心服务器，渲染农场，存储集群等等和软件。后者的范围从低级多线程 C ++库处理数字，用于物理模拟或图像处理，通过高频消息传递中间件，到高级脚本语言，如 Python。

为电影项目构建元数据

资产管理的最重要功能之一是从许多不同的角度监视系统中的文件。 通常，这是通过使用表示以下内容的元数据属性来完成的

生产需要跟踪的文件的各个方面。 这些属性可以直接与文件本身相关（“这是什么类型的文件？”和“在磁盘上的什么位置存储？”），也可以与组织考虑因素（“文件使用的是什么镜头？”和“谁在工作？ 资产管理系统的一项关键任务是声明和存储此类元数据，并使其可搜索。以其最纯粹的形式，元数据可以描述为键值对。 键定义元数据的结构，而值定义单个资产的属性。

以下是一些可能与电影作品中的资产一起存储的元数据密钥的示例：

•工作，场景，镜头

•资产类型

• 名称

•版本

•创作者

•时间戳

•已批准

•已删除

• 位置

将元数据与资产相关联的两种方法是批准和标记。正如我们在前几章中所讨论的，资产批准的过程因设施而异，尽管常见的阶段包括冻结，主要，次要，CBB 和最终阶段。相反，标签是用户可以分配给资产以跟踪它们的任意字符串。举例来说，假设您的客户决定按照动作中的重要点，按“节奏”排列一个顺序。下一个客户端可能不想使用相同的方法。标记提供了一种机制，艺术家可以通过该机制识别属于某些节拍的资产，使您能够回答诸如“我们必须为节拍 1 拍摄哪些照片？”之类的问题，而不必引入节拍的概念作为每种资产的专用属性。

但要注意：容易过度使用标记。只允许艺术家添加您真正需要的那些。您还应该注意不要在代码中直接引用标签本身，因为这样很容易导致由长 if / else 语句组成的难以管理的代码库。资产管理系统不应基于特定资产是否带有“ beat1”标签的决策。

相比之下，游戏通常会在更长的时间内生成更少的数据，并且需要不太复杂的硬件基础架构。然而，他们的软件环境（如果有的话）甚至更复杂，因为内容不仅在大型商业应用程序中创建，而且在编写高度游戏特定数据的小型定制工具中创建：级别编辑器，动画混合树创作工具，状态机编辑器，游戏脚本 IDE 等。修改每个工具以使用中央资产管理系统可能是一项巨大的工程任务。

版本控制的相对重要性在电影和游戏之间也有所不同。虽然电影行业更容易维护仅通过命名约定或位置版本的大量文件，并允许个人“选择”加入当前版本，但正式版本控制软件是任何游戏团队最重要的基础架构之一。

游戏最终是一个复杂的软件，需要数百万行代码和数千个图形资产才能协同工作。通常，代码和内容以锁步方式更改：代码更改需要新资产或更新资产，反之亦然。如果代码和资源不同步，游戏将无法正常播放。在一个大标题上，数百名程序员和艺术家将在制作过程中处理这些资源。确保源代码和数据得到严格管理是避免混乱的唯一方法。

电影和游戏之间的另一个区别是依赖管理问题的相对规模。游戏是围绕资产和代码之间复杂的关系网络构建的。对网络任何部分的更改可能会产生不可预测的后果：例如，更改动画长度的艺术家可能会无意中改变角色在战斗中的效果。不幸的是，没有现成的工具可以理解这种复杂

的依赖关系蜘蛛网。然而，重要的是要了解资产如何相互关联，以便设计管道以最小化这种不可预见的后果。

但是，当我们说一个资产“依赖”另一个资产时，我们的意思是什么？在本章的其余部分，我们将研究这个问题及其对管道开发人员的影响。

第 9.5 节 依赖性跟踪：什么是资产依赖性？

资产之间依赖关系的典型示例是 3D 模型与其纹理之间的关系。大多数依赖关系都是直接的单向连接 - 在大多数情况下，依赖关系被明确记录在艺术文件中。例如，模型需要一组特定的纹理，并且当这些纹理不可用时尝试打开或渲染它将为艺术家生成错误消息。

由于缺少依赖性是摩擦的来源，因此管道通常会为艺术家提供监督这些关系的工具。这可以像游戏引擎一样简单，提供一组预定义的后退纹理，代替丢失的资产。提供后备装备足以吸引注意力，他们应该提醒团队成员缺少依赖关系而不会实际崩溃游戏并使开发停止。其他团队更喜欢更精细的方法，分析艺术文件的依赖关系，以确保依赖关系与核心资产一起保存：“您尝试签入的模型使用两个当前不在源代码控制中的纹理。您想要吗？添加它们？“

依赖关系管理的真正问题在于，随着项目的增长，资产之间的关系变得更加复杂。想象一下，游戏动画师注意到他们正在处理的场景中的一个角色的帽子出了问题。问题很明显，但找到原因可能需要通过几层嵌套引用向后工作，无论是在艺术包中还是在游戏本身中，直到找到有问题的数据。

正如我们已经注意到的，这个问题在游戏中尤其严重，因为任何资产都可能依赖于任何其他资产。即使您忽略了许多不太有用的依赖项，文件或其他数据存储之间关系的复杂性也是巨大的。

第 9.6 节依赖性跟踪：上游和下游依赖性

依赖关系跟踪是资产管理系统跟踪资产之间关系的过程：用于生成有问题资产的其他文件（其“上游”或“传入”依赖关系）以及其他人依次使用它（它的“下游”或“外向”依赖关系）。

跟踪可以是单向的，也可以是双向的。单向跟踪下游依赖关系使资产管理系统能够在将新版本的文件签入系统时向用户通知他们需要更新的下游资产，或者自动为其进行更新。

双向跟踪更复杂，但允许用户检查资产的使用频率和位置。如果一个产品按计划运行，或者游戏内存不足，并且内容必须被删除，这是很有价值的：删除一个只出现一次的资产要比删除一个出现在每个级别的资产要好得多。

选择双向跟踪并不意味着必须记录每个资产关系的两端。资产的使用方式存在根本的不对称性。大多数资产的存在是为了放置更大的东西：场景内的动画，内部的道具等等。大型“容器”资产，如游戏关卡或电影镜头，一直在增加和失去资产。在容器中跟踪这些更改非常简单，因为这仅意味着更新单个资产列表，但在放置或删除所有单个资产时更新它们要复杂得多，并且需要更多工作球队。

如果您的依赖关系信息是外部化的 - 也就是说，如果它被记录在数据库或单独的文件中，而不是存储在资产本身内 - 上游跟踪的大部分好处都可以通过数据分析来实现。数据库非常适合通过遍历每个容器中的资产列表并将它们链接在一起来编译完整的依赖项列表来计算上游依赖项。虽然这个过程可能不是实时的，但它通常足够快，可以使生产顺利运行，并节省大量的工程工作量。

第 9.7 节依赖性跟踪：手动与自动化系统

在我们研究如何设计自动依赖跟踪系统之前，让我们考虑另一种选择：手动跟踪。这带来了明显的缺点，即它依赖于人类的输入。无法信任人们正确更新依赖关系元数据，尤其是在截止日期的压力下 - 确切地说是准确数据最关键的点。

但是，让我们暂时不要注销人类！开发自动化解决方案既昂贵又耗时;人类便宜且适应性强。给人类一个电子表格，很少或根本没有开发工作，你有一个依赖跟踪系统 - 不一定是一致的或彻底的，但如果专注于一小部分依赖关系，它可以很好地工作。

例如，在游戏工作中，该电子表格可以列出构建角色的混合树所需的动画，它们的位置和状态。通常情况下，这将包含几百个动画 - 游戏中使用的动画的一小部分，以及所有资产中更为细小的子集 - 但完全适合特定任务，并且足够小以便手动更新。

团队规模在这里至关重要。如果没有依赖跟踪系统，那么员工流动率低且建立了命名惯例的小型甚至中型团队将会幸福地生存下来。增加团队规模或员工流动率，自动化依赖关系跟踪对于保持生产力至关重要。

第 9.8 节依赖性跟踪：存储依赖性数据

接下来，让我们考虑一些影响依赖关系跟踪系统设计的问题。第一个是存储依赖关系数据的形式。

跟踪依赖关系可能需要存储层次结构，树，甚至连接图。在传统上用于资产管理系统的关系数据库中，这很难有效，因为这些数据库基于平面数据，而不是图形结构。文档或对象数据库可以更好地完成这项工作，但现在有更新的数据库，如 Neo4j 和 AllegroGraph，它们基于图论，并以更自然的格式存储节点和边。与 SQL 中的等效模型相比，它们通常更快，更具可伸缩性，功能更丰富。

第 9.9 节依赖性跟踪：可视化依赖性

一个好的管道应该包括可视化构成生产的依赖关系网络的工具。没有它们，很难真正了解在任何给定时间实际使用了多少资源。特别是游戏团队总是担心他们可以使用的内存量。一个好的依赖跟踪系统可以找到可以优化内存使用的方式：例如，支持库的子集，它共享大量纹理，占用较少的内存作为一个单元，或者一个字符，有一个昂贵的动画集，应该削减。

在预算运行时成本时，可视化依赖关系也很重要。在一个关卡中放置一个额外的模型可能看起来像添加额外颜色的廉价方式 - 但如果该模型带有巨大的纹理或大量的动画，则不会。资产的实际运行时成本包括其在场景中不存在的其他运行时资产的所有传入依赖项的成本。因此，为了有效地控制成本，能够可视化资产之间的关系并评估其对游戏的影响至关重要。

资源成本在电影制作中不是一个问题，但对于电影团队而言，能够可视化资产依赖性仍然很重要：例如，查看场景中有多少道具已完成以及仍有多少道具工作上。艺术家能够看到他们的变化对制作产生什么影响也很有用。如果您可以让您的艺术家清楚地了解他们的工作如何在下游使用，您将最大限度地减少令人不快的意外。

可视化依赖性本质上是图论的问题。大型生产的依赖图可能非常复杂，但基本构建块只是资产和这些资产之间的链接。相对简单的脚本可以通过分析艺术工具，内部软件和游戏引擎创建的文件来收集此信息，并将其组合到节点图中。

真正的挑战是演示：以可用的格式在团队面前获取信息。将依赖关系信息呈现为树或节点视图似乎很自然，类似于艺术工具中的大纲或图形视图。这适用于项目的小部分（例如，显示单个资产的输入），但对于大量数据会快速分解。

另一方面，本质上围绕页面之间链接构建的网站是表示资产之间链接的自然方式。分析图表后，创建与您的资产对应的 HTML 或 Wiki 页面，使艺术家可以使用熟悉的用户界面轻松导航复杂的数据集。例如，模型的资产页面将包括其纹理的上游链接以及使用模型本身的场景或游戏关卡的下游链接。包含用于跟踪大图片项目的摘要页面也是一个好主意：资产总数，资源使用情况，按审核状态分组的资产等。

第 9.10 节依赖性跟踪：解决隐式依赖性

要考虑的另一个问题是隐式依赖。当通过查看资产本身无法发现两个文件之间的关系时，会出现最棘手的问题。在游戏中，一个臭名昭着的例子是角色骨架。在许多游戏中，技术限制要求角色的骨架在其出现的任何地方都是相同的。这意味着如果艺术家意外地将骨骼重命名，删除或添加到现有角色，整个管道可能会停止 - 尽管受变化影响的数百个文件本身没有变化，并且可能仍然看起来像任何在 3 ds Max 或 Maya 中打开它们的人都一样。

隐含的依赖是麻烦。他们可以在不发出警告的情它们也很难跟踪，因为 - 不像我们原始的模型示例及其纹理 - 它们不包含简单的引用文件。找到并解决隐含依赖性问题的原因可能涉及长期拖网，通过最近的变化寻找可疑的东西：几乎没有有效利用任何人的时间。

解决隐式依赖问题的唯一真正解决方案就是不要拥有它们。当两个文件之间存在依赖关系时，它应该以某种方式显式化。例如，主骨架文件和依赖于它的动画之间的关系可以通过使用艺术工具内的文件引用功能来表达。这会自动将更改从主文件推送到动画。在这种情况下使用引用将一个讨厌的，有潜在危险的隐式依赖转换为一个简单的显式依赖 - 虽然根据管道，可能还需要重新处理或重新导出所有相关文件。

或者，可以通过将隐式依赖关系记录在数据库中或通过创建文件（通常称为“边车文件”）来明确隐含依赖关系，该文件的作用是跟踪关系。在上面的示例中，可能有一个 XML 文件列出了依赖主骨架文件的所有动画。更改主服务器后，可以向用户发出有关需要更新的大量文件的警告。相反，当打开单个动画时，动画文件可以查阅跟踪文件以确保它与主控制器同步。

第 9.11 节依赖性跟踪：缓存查询

每次需要评估依赖关系时，并不总是需要查询数据库。某些查询（例如检索镜头列表）可以多次进行，而不会更改结果。在这种情况下，每次查询数据库都是浪费。更好的模型是在第一次请求信息时查询数据库一次，然后将结果存储在中间缓存中。然后，未来的查询可以优先于数据库使用缓存。一旦缓存的信息变得陈旧，就可以再次从数据库中获取结果并将其放回缓存中。

无论如何，数据库都采用这种形式的缓存，但它也可以作为外部服务提供，例如 Memcache 或 Redis。缓存通常是内存中的键值对存储。每个项目都有一个“生存时间”值，当达到此值时，它将从缓存中删除。如果缓存已满，则根据其剩余生存时间或根据“最近最少使用”模型删除项目。

此外，如果查询处理缓慢但产生可预测的结果，您可能决定对其进行预计算。例如，如果您知道您将始终需要一个镜头的资产列表，则无论何时创建新资产，您都可以将其添加到存储在其他位置的列表中。这涉及额外的维护 - 如果出现问题，您应该始终能够重新计算数据库中的值 - 但这是一种常见的优化技术。同样，结果可以存储在缓存中，也可以作为单独的表添加到

数据库中。

第 9.12 节相关性追踪：资产分组

资产管理系统还应使用户能够将资产组合成组。这些组本身应该是版本可控的。例如，角色由多个资产组成：模型，骨架，肌肉装备，着色器，纹理等。不是单独处理这些资产，而是可以定义分组资产来存储信息。告诉艺术家他们应该加载 Character_1 的第 56 版，而不是向他们展示每个单独资产的版本列表，并期望他们分别加载它们。

集团内的资产也应受到某种批准程序的约束。在电影工作中，这些资产之间的大部分依赖关系都会回归到几何体。纹理艺术家使用模型的 UV 来定义要绘制的区域，但绑定团队也可以使用它来绘制权重贴图。反过来，这些装备将传递给动画团队，然后导出几何缓存以供照明部门使用。

假设照明部门要求更新模型的几何体，这反过来需要更新其 UV。在没有批准的系统中，该模型将立即供所有下游学科使用。如果光照是在装配和动画之前拾取新模型，那么任何缓存的动画都不会起作用。通过提供允许一个部门在其他部门完成所需工作之前获取资产的批准标签，可以避免此问题。

这总结了我们对依赖性跟踪的探索 - 以及一般管道设计的技术方面。在下一章中，我们将研究流程的人性方面：管道如何用于管理人们的工作方式。

MPC 的资产管理系统概述

MPC 的资产管理系统的核心是一种机制，该机制可以唯一地标识单个资产，例如模型，装备，纹理，缓存，图像，配置文件，颜色决策列表，查找表和 QuickTime 电影。这些资产按类型标识并具有唯一的名称。通过增量版本号跟踪更改。作为定义每个资产位置的一种方式，将它们与围绕作业，场景和镜头的上下文相关联。我们可以使用简单的点符号来引用资产，格式如下：job.scene.shot.type.name.version。在许多情况下，需要将单个资产组合在一起以形成更复杂的结构，就像原子链接在一起形成分子一样。一个示例可能是角色及其动画的分组。我们将这些资产集合称为“包”，并将其视为版本控制资产本身。软件包提供了多种类型资产的结构化分组，其中一些资产本身就是软件包。在给定的示例中，字符与动画还是定义了装备，模型，纹理和其他类似资产之间关系的软件包。我们定义了不同类型的程序包，就像定义了不同类型的资产一样。除了定义关系外，程序包定义还建立了一些规则，用于验证数据的一致性并自动生成其他资产。例如，当发布新版本的模型时使用更新的拓扑，系统可以检测到现有的动画缓存不再适用于该模型，并可以重新生成它们。因此，打包系统描述了资产的分层网络，它们之间的依赖关系，以及在资产的上游或下游依赖关系发生变化时管理资产再生的规则。检索这些关系和依存关系的能力为员工提供了一种机制，可以识别管道中相关资产的状态。为了进一步简化生产流程，该系统还支持批准。如果对资产的更改进行了标记，则仅将资产的更改传播到下游软件包。

适当地。例如，灯光艺术家只能使用从批准的角色包和批准的缓存构建的动画角色包。这很重要，因为它使我们能够在将资产传播为快照之前验证其是否正常运行。可以将此系统视为一系列锁和钥匙。包装上有锁，资产只有在拥有正确钥匙（即批准）的情况下才能解锁它们并进入内部。

涉及：莱卡的数字资产管理

Laika 的故事片完全由内容开发，从脚本到最终打印（或 DCP）。角色和布景设计为在我们的商店手工制作，序列和镜头在我们的舞台上逐帧进行数字化捕捉，然后交付给我们的 VFX 团队，以获得更多元素和最终合成。我们通过故事，角色和布景设计，从剧本初稿到舞台和 VFX，以数字方式跟踪资产。我们的系统基于文件系统，依赖于定制发布工具网络，以确保名称和路径保持一致，从而允许每个部门的设计和构建过程完全不同。一旦进入文件系统，每个人都使用 Shotgun 来浏览，访问和链接资产，版本和注释。创建的几乎所有内容都可以链接回三种主要的资产类型：角色，设置和镜头。

我们在开发资产管理系统时面临的一些挑战包括：将物理设计元素（如雕塑，编织服装和手绘设计）导入数字文件系统，从原始设计中跟踪成千上万的零件，并通过它们构建的商店进行跟踪。到拍摄阶段，通过 3D 打印，质量控制，作为动画套件传送到舞台，跟踪面部形状以获得认可的表演，并将所有设计送入传统的 VFX 管道，用于构建 CG 角色并设置扩展和开发 FX 任何不能在舞台上拍摄的东西。

以下是管道的高级概述，我们正在跟踪的数据类型以及我们如何将资产链接在一起。所有发布，解析和报告均由我们的生产技术团队开发的定制软件完成。

早期故事发展

打击板，概念艺术和参考是从 Photoshop 或独立发布应用程序发布的，该应用程序根据类型命名和定位文件服务器上的文件，并链接到称为样式的外观开发的官方字符，环境或类别指南。节拍板是序列的前身。所有这些艺术作品都可以在 Shotgun 中查看，其中状态字段被更新以指示每个类别的批准版本。我们在 Shotgun 中添加了用于检索的工具，以及继续作为正在进行的工作发布的设计，以便艺术家不需要浏览文件系统本身。通过 Shotgun 播放列表也可以为导演和制作设计师提供评论，这些播放列表非常适合管理笔记和更新版本状态。批准的设计版本通过 Shotgun 和印刷报告与其他部门和制造团队共享。

脚本

一旦脚本被绿灯照亮，我们就会解析它并建立最终将被登上并发展成镜头的序列和场景。解析应用程序可以进行初始分解，将集合和说话角色链接到场景资产。所有这些都是在 Shotgun 中创建的，并且可以在脚本版本化时进行更新和编辑。场景，场景和角色资产成为创建初步拍摄计划的第一个单位。他们还为所有继续从外观开发和编辑发布的设计作品和故事板创建定义的类别。初步拍摄时间表是将集合，木偶和脸部替换的资产构建时间表与动画时间表相关联的内容。

要编辑的故事

故事板发布到社论。每个电路板都使用唯一的编号和艺术家名称进行跟踪，以便于在音高工具中进行修改，重复使用和组装。一旦编辑成序列，剪辑可以在 Shotgun 中发布，将登上的场景链接到基于原始脚本的场景及其资源。剪辑也可以在 Shotgun 中查看，并更新资产的细分。编辑发布场景的运行时间用于计算用于在舞台上调度动画的拍摄任务的估计持续时间。当场景最终

被分解为定义的镜头时，资产列表被进一步定义，允许我们为集合，木偶和面部替换预测每周资产交付要求。故事越早巩固，我们预计的时间表就越好，但它永远不会同时出现，所以我们的系统允许拍摄和不太具体的父母场景在长期安排中共存。

布景制作

从批准的概念设计艺术开始，艺术总监将环境细分为集合，并根据他们在已发布的故事板中看到的内容来设置组件。每个组件都是为构建而设计的，包括插图，建筑草图和参考照片。设计已发布并链接到设定组件。在 Shotgun 中，这些组件具有与集合的链接，这些集合与镜头及其拍摄日期相关联，从而通知设定的制作团队需要构建什么以及何时构建。每一套都有一个 Shotgun 中的数字 Build Bible，它是所有批准的设计的集合，用于构成一组的所有组件和道具。圣经的版本作为报告提供给专门从事不同领域的商店，从建筑物，建筑细节，图形，绿色，道具到软商品。圣经中跟踪的数据包括项目的计数，油漆的 pantone 颜色，画笔描边细节，父母和子女资产以及所需的批准类型。如果该项目是设置扩展镜头的一部分，则可以将相同的详细信息传递给 VFX。

木偶制作

木偶是手工制作的，几乎没有使用数字资产跟踪工具。艺术家利用镜头分解和日程安排工具按周获得准确的木偶计数，这样他们就知道每个角色需要多少，以及何时，因为第一个木偶的交付时间可能是几个月。他们还发布了每个完成的设计和制作阶段的照片，以便有最终电枢，模具，油漆，头发和服装设计的数字记录。

通过 Shotgun，这些高质量的设计资产可供商店负责人使用，他们在 VFX 中构建复制木偶作为数字双打和背景角色的参考。 Puppet 部门还使用电枢数据库来管理构成每个角色的自定义加工零件和组件的库存。在设计零件时，Autodesk 的 Inventor 文件将被发布并链接到角色，因此当木偶需要修复或重复构建时，可以快速找到每个关节和肢体的骨架部件列表。

面部更换

设计和建造木偶面部的部门称为 RP，以快速原型 3D 打印机命名，用于批量生产构成人物头部和面部的部件。这些头部非常复杂 - 典型的英雄木偶头部可以有多达八十个不同的部分。

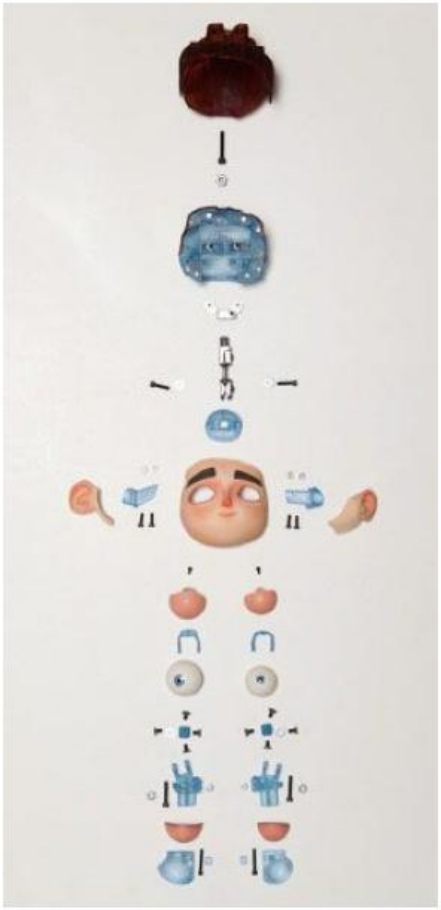


Figure 9.2 Interlude Tracking assets for Laika's hero puppet heads is particularly important. Custom parts are digitally designed and printed on 3D printers for the core assembly, which must fit hundreds of facial shapes. Photo courtesy of LAIKA, Inc.

图 9.2 插曲追踪莱卡英雄木偶头像的资产尤为重要。 定制零件经过数字化设计，并在 3D 打印机上打印以用于核心组件，该组件必须适合数百种面部形状。 照片由 LAIKA，Inc.提供

这些部件中的大多数构成了附着在木偶颈部的核心，并且具有铰接的眼睛和盖子以及对准磁铁，其中眼睛和嘴巴的形状可以卡入到位;这些眼睛和嘴巴的形状被动画师逐帧取代，以创造性能。设计过程全部采用数字化设计，基于经过批准的粘土雕塑，扫描并带入 Maya。从那里，建模人员设计适合面部的核心，眼睛钻机和登记系统。面部被操纵，动画师为每个镜头中的每个角色创建面部表演。通过将模型和装备链接到角色以及面部表演到镜头的发布工具，在整个过程中跟踪数字资产。可接受形状的范围被打印并布置在物理库中基于字符的套件中。基于射击的套件，称为 XSheets，在 Shotgun 中进行跟踪以获得所有批准的演出。面部图书管理员可以查找镜头并找到 XSheet，告诉他们要将哪些形状组装到舞台上。我们还跟踪使用数据，以便我们可以找到打印融合的眼睛和嘴巴形状的机会，并避免 VFX 清理完成的动画。我们希望有更多的能力来跟踪物理资产，即图书馆中数以千计的面部形状。我们正在开发一个数据库，以跟踪单个形状，并在资产上打印出唯一的数字，并可以帮助图书馆员找到兼容的打印批次。颜色质量随时间而变化，因此在不同时间打印的形状与逐帧使用不兼容。

舞台/相机/动画

作为舞台工作流程的一部分进行跟踪的主要资产是镜头。帧从舞台上发布作为灯光测试，

角色块，排练，英雄动画，干净的盘子，绿色屏幕元素和各种相机通行证。它们与发布时的镜头资产相关联，如果获得批准，则转交给 VFX。 Shotgun 拥有所有这些版本，批准状态和编辑说明。此外，对于摄像机团队，我们通过设备登记系统跟踪镜头，运动控制开关，色彩精确显示和捕捉系统的位置，该系统显示舞台地板上的位置。在生产高峰期可能有多达 50 个活动单元，因此如果没有这些信息，找到一个特殊的镜头将是一个挑战。动画师和照明技术人员还在舞台上使用 Shotgun 和 RV（来自 Tweak）观看剪辑，当前故事板，周围镜头以及可能为镜头发布的任何预览或布局。视觉特效团队还收集和发布调查信息，包括大多数单位的 HDR，参考相机帧和摄影测量源照片。

VFX

我们的 VFX 团队使用常见的第三方工具，如 Maya，Nuke，Houdini，Katana 和 Renderman 来开发 CG 资产，并创建所有阶段传递的最终合成镜头。他们发布了工作文件和每个步骤的版本，我们有工具将当前和批准的元素版本提取到下游任务中。所有版本都在 Shotgun 中并链接到资产或镜头以及任务和艺术家。

评论通过收集所有每日发布的自动播放列表进行管理。笔记也在 Shotgun 中跟踪，并与镜头和艺术家相关联。其中大部分被认为是标准的 VFX 工作流程。对于资产开发，在木偶和集合制造部门中进行的所有详尽设计工作都用作需要创建的数字字符或集合元素的基础。 VFX 资产可以在 Shotgun 中链接到父资产，从而允许它们继承设计和参考。通过我们的计数表工具，可以更轻松地在舞台上使用多个通道。该工具解析 Film Scribe 中的剪切列表文件并将其上传到 Shotgun。编辑可以与导演一起组装板块，设置进出框架，重新定时等，然后通过 Shotgun 将这些信息传递给 Nuke。我们的合成器可以加载一张计数表，从舞台拍摄中获取所有必要的通行证并将它们排成一行开始工作。

生产管理

第 10.1 节 您本章中学到什么

管道不仅仅是关于工具：它们也是关于使用这些工具的人。在本章中，我们将介绍管道开发人员可以用来管理人员和数据的方法。

我们将首先查看管理生产的方法，然后继续讨论团队可以采用的更具体的策略，以确保他们的管道能够最大限度地提高效率 - 最终，项目按计划完成，并达到预算。

一旦我们探讨了理论问题，我们将研究项目管理中涉及的一些技术，包括跟踪资产完成情况所需的工具，管理资产评估和跟踪记录，以及安排生产任务。

第 10.2 节生产管理策略：敏捷与瀑布式开发

我们将首先研究可以管理生产的方法。其中两个最常见的是“敏捷”和“瀑布”模型。这些术语起源于软件开发领域，但现在已广泛应用于项目管理。

在敏捷开发中，自组织的跨学科团队每个人都“拥有”最终产品的一部分，随着时间的推移进行合作。开发以短周期迭代进行，开发和测试同时进行，反馈指导未来发展。随着每个新功能的上线，它被集成到最终产品中，逐渐扩展：敏捷开发的一个关键原则是，如果有必要，您可

以削减尚未实现但仍具有工作产品的功能。

当最终产品应采用的形式尚不清楚时，敏捷开发是很好的。它的主要缺点是该产品的开发过程很难预测。迭代工作流程可以促进创造力，但它可能与支持者或出版商设定的严格里程碑相冲突。

相比之下，瀑布式开发是一种更加严格的线性方法，其中产品按顺序阶段构建，通常定义为需求分析，设计，实现，测试，集成和维护。每个阶段必须在团队进入下一阶段之前完成。

当最终产品已经详细规划，或者开发需要在特定时间范围内产生可预测的结果时，瀑布开发非常有效。但是，由于它是一个高度结构化的过程，纯瀑布式开发不容易适应产品规范中的各种变化，这在图形制作中很常见。

在实践中，设施很少使用任何一种方法。即使是一个严重依赖瀑布式开发的工作室，在探索新技术时也可能会采用更灵活的方法。每种方法的适用性也因市场部门而异。电影，他们的刚性内裤和紧迫的期限倾向于瀑布发展;游戏通常更敏捷。 Tim Green 高级程序员，超大型游戏

实施敏捷方法论

我们的敏捷开发过程如下。我们首先将已知的高层目标放在“backlog”上：一个上面钉着卡片的板。每个目标有一张牌。卡片从左到右沿板的 X 轴排列，使得在开发之前必须实现的目标出现在依赖于它们的那些目标之前。在可能的情况下，卡片也沿 Y 轴的规则分组。左边的卡片显示了详细的目标，但当你向右移动时，目标会逐渐模糊。随着开发的进行，董事会不断更新，随着任务的完成，团队会删除左侧的卡片，随着这些目标的明确，会在右侧的卡片上添加细节。可以将点数分配给卡片，以表示目标大小和目标风险。一般来说，一张卡在董事会右边越远，为了应付它必须面对的未知因素越多，风险就越大但是，也有例外：为游戏编写手册是一个低风险的目标，例如：它恰好位于右侧，因为游戏必须在完成之前锁定。我们从猜测一个转换因子开始,将点转换为实现每个目标所需的时间,使用一个任意的单位，我们可以在以后随着更多数据的输入进行调整，而不必进行诸如“你只花了两周时间就完成了本应是一周的工作”之类的尴尬对话。我们跟踪转换因子的误差并加以修正，使其趋于更精确的值。这为预测未来的进展提供了一个越来越精确的模型。

一旦工作开始，每个目标都被分解成任务，每个任务都是分配给一个人的一项工作。这些任务写在便签上，放在目标卡旁边的另一块板上，完成后删除。这个更详细的董事会拥有 sprint 的工作价值，其中“sprint”是一个短的、预定义的时间段：通常是一到两周。将这一过程结合在一起的技术至关重要，而董事会、董事会、董事会和董事会只是实现这一目标的唯一途径。他们的优势在于，人们不必打开软件就可以看到目标，而且当有人做出改变时，这一点很明显，因为你看到他们为了实现目标而走向董事会。此外，由于“界面”（用记号笔在纸上书写）是如此自然，因此更容易集中在卡片的实际内容上。缺点是，除了在废纸篓中搜索旧的 post-it 注释外，无法跟踪项目的开发历史。此外，空间也是一个限制因素：大型项目需要大型板，而且很难容纳大型数据集，例如项目的所有动画任务。有一些数字解决方案可以做同样的事情，在某些情况下，这些可能是唯一的选择：特别是在团队成员并不都是在同一个办公室工作。

第 10.3 节生产管理策略：效率最大化

接下来，我们将研究更具体的策略，以最大限度地提高管道效率。但在我们这样做之前，让我们回顾一下制作并不总是按照时间表运行的原因。

客户计划更改

制作并不总是顺利进行的最明显原因是外部环境发生变化。由于 VFX 在生产过程中占据的位置，这在视觉效果工作中比在游戏或动画特征上更重要。实时拍摄经常超支 - 因为舞台不可用，因为演员被绑在其他工作上，或者因为恶劣的天气影响户外拍摄，例如导致板块被送到供应商的后期。由于电影的发布日期是固定的，由营销和发行的需求决定，视觉效果工作的可用时间可能会急剧缩小。

目标改变

制作游戏或电影并不像搭建一座桥梁。最终产品是无形的，这使得客户 - 生产者或主管 - 更容易改变他们对需求的看法。很可能只给出反馈，例如“继续向我展示事物：我会在看到它时知道我想要什么”，或者经过数周的迭代，只有导演决定他们更喜欢版本一。

工具更换

管道使用的软件和硬件是错误的，内部开发的工具通常在工作中进行测试。事情有一种破坏的习惯，往往是在最不合时宜的时候。等待商业工具被修补，或者内部软件被更新，重新编译和测试，导致不可避免的延迟，特别是如果艺术家没有其他任务可以进入。由于服务器或网络问题导致的停机时间也会降低管道效率。

人不是机器

即使您的工具一直在工作，操作它们的工作人员也不会。人们出去吃饭和抽烟;他们生病或找工作;偶尔，他们甚至想在晚上回家。而且，他们犯了错误。在管理生产时，始终需要考虑由简单的人为错误引起的延迟。

设施几乎无法控制前两个因素。很难预测工作室何时会提供印版，或者客户在审查会议中要求的变更。但他们可以控制自己的基础设施，在较小程度上也可以控制他们的员工。我们先看看基础设施。

在一周的过程中，在视觉效果设施中使用渲染农场 - 在较小程度上，在游戏工作室中使用建造农场 - 将经历高峰和低谷。工作日晚上，特别是星期五，将提交大量工作;周日晚上和清晨会更安静。

这两种极端都不可取：在农场上积压工作会浪费艺术家的时间，但让农场闲置的机器会浪费电力。这同样适用于存储和网络带宽，这两者都是具有限制的资源。在高峰时段，存储容量很快就会消耗殆尽，这可能会导致艺术家在将空间分配给更重要的镜头时闲置。

某些软件进程也会将工作站捆绑在一起，直到它们完成，从而阻止员工转向其他工作。对于开发人员来说，编译软件通常是罪魁祸首。对于 FX 艺术家来说，它正在运行模拟。对于打火机，它是渲染。对于动画师来说，这是 Playblasting。这里有几种可能的策略。一种是使用蛮力，将耗时的工作推入农场，或者为艺术家提供两台机器，以便他们可以在忙碌时来回切换。另一个是优化软件，以便更快地完成这些任务。另一种方法是简单地为员工提供他们可以在其他软件中工作的任务，直到他们的主要工具重新上线为止：例如，生成文档或培训。

但是，从管道开发人员的角度来看，最困难的任务是让员工高效工作。与机器不同，人们不会同化所有输入数据，处理它，然后第一次输出数学上正确的结果。相反，他们使用效率低得多的工作流程：给定可用的输入，他们产生输出，审查它，然后重复包含他们所学知识的任务。

我们可以通过识别他们的弱点并发挥他们的优势来改进我们如何在管道中使用人类。人类需要迭代才能产生好的内容，因此无论人类如何与管道交互，都要构建迭代功能并使每次迭代都变短。如果你必须在某个地方节省开发工作，那么通过快速迭代工作流程来获得有些过时的技术通常比使用迭代速度较慢的尖端技术更好。

第 10.4 节生产管理战略：按时、按预算完成

当然，最大限度地提高生产效率只是达到目的的手段。生产管理的最终目标是确保项目按计划完成并达到预算。接下来，我们将研究规划这些时间表的策略，并确保您坚持这些时间表。

建立缓冲区

在前面的章节中，我们讨论了从您完成的每个项目中捕获数据的需求，以便更准确地预测未来的制作需要多长时间。但是你也应该花费额外的时间来保护你免受不可预见的不测事件的影响。制定加班是唯一错误的时间表的制作人正在设定几乎肯定会导致长时间工作和疲惫，士气低落的工作人员的条件。

在额外时间建设也有助于提高成品的质量。在时间压力下，人们变得不那么有创意，更倾向于偷工减料。例如，已经工作了 12 个小时的程序员不太关心他们的代码是否可重用。鉴于一个充满激情的团队，调度不足通常会产生不成比例的更好的结果，因为人们会花费额外的时间来使产品“更好”，而不仅仅是“完成工作”。

然而，在视觉效果中，建立一个有意义的缓冲区几乎是不可能的 – 这既是因为影响制作的大量外部因素，也是因为一些导演在截止日期前要求他们完成拍摄。相反，它足以知道有多少镜头到期，以及是否有可能在剩下的时间内完成它们。而不是安排缓冲区，最好确保每个镜头的预算成本涵盖艺术家的时间和工作室的开销。在定价时，很容易省去支持成本。

从小处开始，然后全体行动

不是在整个生产过程中保持团队规模不变，为 70％的项目运营一个小而紧凑的工作人员可能更具成本效益，从而为这个核心团队提供解决问题的机会。一旦管道被测试，编辑被锁定，并且关键镜头或故事弧被批准，您可以参加项目的最后 30％。这同样适用于硬件资源：最好在最终推送中使用渲染节点或存储空间，而不是让它们在大多数生产中闲置。

当然，那些额外的工作人员必须来自某个地方。在大型设施中，可以从其他项目中获取它们。在这种情况下，每周安排会议对于合理化不同项目团队的竞争需求至关重要。否则，他们将需要从外部招聘。这意味着要关注当地的人才市场 – 例如，在毕业季节，初级员工更容易招聘；当附近的设施工作人员停留在制作上并且在足够的交付时间内建立自由职业者时，尤其是在招聘可能必须在国际上招聘的高级职员时。

但是，请记住，在项目后期添加人员会增加现有团队成员的负担，因为他们必须简要介绍并监督新员工。由于不熟悉设施的工具和流程，新员工不可避免地会引入更多错误，导致效率下降。在生产后期添加额外的硬件也可能带来灾难性后果。例如，增加渲染容量所产生的额外流量

可能会降低网络速度或限制存储空间。

错开你的紧缩期

如果您一次只处理多个项目，则必须确保最后期限错开。对一个项目的延迟将使人员和硬件资源的时间超过预期，从而导致其他产品可能会产生影响。

计划什么该放弃什么不该放弃

如果项目超限并且没有更多可用资源，则解决方案可能比您最初计划的要少。在电影工作中，这可能意味着放弃整个镜头。这通常是最后的手段，因为它可以在情节中留下漏洞 - 或者至少迫使一个序列被重新编辑 - 并且必须由客户驱动，通常是为了响应预算而不是时间压力。一个不太激烈的选择是缩小效果：例如，通过减少屏幕上的字符数或爆炸的大小。最后，你可以回拨质量：接受你只能得到一个“90％那里”可能节省相当多的时间 - 并且观众可能实际上没有注意到差异。

在游戏中，相当于删除功能，甚至删除整个级别。从项目开始就计划这一点非常重要：在以故事为中心的游戏中，在制作后期降低一个级别可能会对叙事产生巨大影响。从第一天起将某些级别指定为“奖励级别”会更好，并且只有在完成更多关键任务后才能生成它们。

如果一切都失败了，合同就生效了

如果即使切割内容无法保存项目，您也可能被迫签订合同。在视觉效果中，这被描述为“911项目”：设施 A 意识到生产迟到，无法按时完成所有工作，因此它将“911 调用”外包给工厂 B。不言而喻，这是最后的手段：除了增加成本之外，对 911 项目的影响质量经常受到影响。

到目前为止，这种做法在游戏开发中很少见：虽然外包很常见，但游戏工作的性质意味着它必须提前进一步规划。出于这个原因，许多游戏外包合同也有内置的支持协议，以确保在初始合同结束后出现无法预料的需求时，承包商将可用，例如需要额外拾取线的错误或脚本更改。Manne Ohrstrom 高级软件工程师，Shotgun 软件 Ryan Mayeda 产品制作人，Shotgun 软件 Rob Blau 管道工程主管，Shotgun 软件参考图像相关工作说明

管理生产

管理生产就像管理游轮，只是该游轮通常有一名以上的船长，并且在航行后几乎可以保证目的地会改变。不过，船东希望您能准时到达并燃烧掉最后一滴燃油。强大的生产管理团队是您实现这一壮举的最佳机会。在宏观上，生产管理（PM）的目标是按时按预算完成项目。从微观上看，这意味着：

- 确保人们知道该做什么，什么时候该工作
- 确保人们知道他们正在做的工作的背景
- 跟踪完成和批准的时间

让我们更详细地了解这些要点。首先，您需要让人们知道该做什么。

传达谁在处理时间表。您计划在哪个工作单元（任务，工单等），以及您计划的任何方法（瀑

布或敏捷），一支优秀的生产团队可确保每个人还知道他们需要多长时间才能完成当前任务以及下一步将要做什么。为此，您需要了解：

- 如何将项目分解为可调度的单元（招标过程）
- 如何发布时间表，以便人们知道要做什么
- 如何更新时间表
- 从长远来看如何松散地安排时间表，在短期内如何以更细粒度的方式安排

一旦每个人都知道他们应该在何时进行工作，他们就需要有关如何设置任务的更详细的信息。不应告诉任何人“为汽车建模”，并且应期望其返回项目所需的确切模型。作品中的所有内容都需要与艺术家交流，以使他们正确地完成工作。该上下文可以分为以下几类：参考图像从互联网上下载或在内部创建的，应为每位艺术家提供一组批准的参考图像，以及解释每个图像为何如此重要的注释。 （例如，应该清楚是否提供了特定的图像，以便艺术家可以参考 其调色板或图像本身的内容。）相关工作在制作的大部分时间，艺术家将创建与先前创建的类型相似的内容。他们可以从已有的模型示例中看到更多的历史记录已获得编辑团队的动画批准，情况越好。笔记在生产跟踪的世界中，重要的笔记是关键。艺术家应该有权访问与其工作有关的所有反馈和变更请求。工作完成后，知道是否已经完成是生产管理工作的一部分批准或是否需要进一步迭代。跟踪批准意味着跟踪工作的不同迭代，并将每个迭代与一个状态相关联。需要收集批准和注释通常需要某种形式的正式审查过程。评论是生产中最有用的会议之一，但也是最昂贵的会议之一。为了充分利用这些信息，无论涉及的审查是由主管发言，还是在专门的筛选区域进行更正式的会议，都应由生产管理人员参与。审查过程越结构化，就越容易捕获其生成的大量信息。如果可能，应提前准备内容，以便每个人都知道正在审核中，以方便以后关联批准和注释。生产管理所需的工具在很大程度上取决于工作人员的规模和项目的时间。由数百位艺术家在一段时期内创作的功能动画相较于三个人在短短几周内创建的商业广告，十年的需求更为深刻。但通常来说，在评估生产管理系统时，请注意要注册的跟踪数量。很难（即使不是不可能）自动化大量工作流，并且生产经理和协调员往往很忙，他们将停止输入数据，除非这样做显然对生产有利。对哪些信息具有选择性，使其成为生产管理系统，并确保易于使用。数据到达那里对于成功的生产跟踪至关重要。

第 10.5 节生产管理技术：概述

管道是生产过程的技术实施例。它的关键部分对应于工作室内的部门，每个部门本身都由一组艺术家组成。因此，很多工作室都将用于管理这些艺术家作品的工具纳入管道中并不奇怪：由于管道是围绕工作室的工作方式设计的，所以至少在理论上应该很容易添加能够使工作室管理能够查看项目的进展情况，将任务分配给个人或团队，以及监控日程安排。

当然，在实践中，很少有可能仅使用软件来提供完美的信息。经理想要的答案通常是主观的：计算机可以提供“已完成”场景或“已完成”资产的列表，但只有人类可以决定哪些实际上足够好而不需要重新加工。同样，虽然计算机可以跟踪给定艺术家打开或更改的文件，但它对该艺术家作品的质量一无所知。此外，对于一个项目而言，良好信息的错觉可能比对大多数作品笼罩的“战争迷雾”的坦率承认更糟糕。由于所有这些原因，综合生产跟踪是管道团队中一个有争议的主题。

尽管如此，值得考虑管道可以帮助项目经理的许多不同方式。以下是有时集成到管道中的生产管理功能的快速概述。

10.6 生产管理技术：跟踪资产

大多数项目涉及如此多的文件，没有人能够希望跟踪它们。因此，大多数资产管理系统为您提供有关资产状态的信息：哪些只是原型，哪些是完成的，或者哪些是需要纠正的问题。理想情况下，系统将生成图表和报告，以便管理人员了解项目的运作方式。

管理状态数据非常简单：就存储和所需的可视化工具而言，它与项目中的许多其他类型的元数据没有什么不同。棘手的部分是首先生成良好的数据，因为很少有一种简单的方法来描述项目中所有资产的状态。

例如，静态环境模型理想地以线性方式发展：从概念到原型，到详细模型，再到具有纹理的详细模型。在这一点上，它将“完成”，因为它已准备好融入更大的环境。在实践中，创意反馈打破了这种线性流动。也许在概念艺术中看起来很棒的建筑作为 3D 模型失败了;或者可能必须改变概念本身以匹配环境的新设计。状态跟踪系统通常需要不仅维护资产的当前状态 - 例如，其文件类型或它在磁盘上占用的空间 - 而且还要维护其状态历史记录。重要的是不仅要知道这个资产处于概念阶段，而且由于来自客户或项目负责人的反馈，Supervisor X 将其发送回概念团队。

值得注意的是，不同类型的资产将经历不同的发展阶段。从阻塞传递到完成的工作的动画的开发与上述模型的开发非常不同。在设计资产跟踪系统时，务必确保用户对每个资产生命周期阶段的性质和顺序有所了解。许多商业系统，如 Shotgun 和 TACTIC，使用户能够创建自己的工作流程步骤。

另一个需要考虑的问题是如何更新资产的状态以及由谁更新。这将反映公司文化。一个小而紧密的工作室可能允许个别艺术家自己设定资产的状态，而一个团队较少的大型工作室可能需要创建一个更正式的流程，其中资产必须转发给主管或团队负责人评论。

10.7 节生产管理技术：管理说明

我们已经研究过资产的状态如何与其发展历史密切相关，以及资产如何因为技术或创意反馈而经常被“降级”为较不完善的状态。因此，许多资产跟踪系统会记录此开发历史记录和此反馈，以便艺术家了解每个资产的位置以及未来应该在哪里。

记录的第一件事是谁在开发过程中处理过资产。当需要调试文件时，联系最后一位处理该文件的艺术家并询问发生了什么而不是尝试从第一原则进行诊断时效率要高得多。看起来像一个令人费解的艺术选择可能就像一块应该被删除的参考几何一样简单：这对于原始艺术家来说是显而易见的，但可能不是以前从未见过该文件的人。大多数版本控制系统都是免费提供此类信息，但重要的是要确保技术人员和需要它的技术人员能够轻松访问这些信息。

记录的第二件事是对资产给出的反馈的性质。处理此类“注释”的最常见系统是通过内部电子邮件，在简单的情况下通常就足够了。但是，在更大的团队中，可能需要实施更正式的流程。确保存储在这样的系统中的数据是完整的是很困难的：如果反馈的重要方面是在电子邮件中而不是在数据库中，那么艺术家可能会有错误或误导性的信息，这可能比根本没有。出于这个原因，一些资产跟踪系统试图通过中央系统路由所有与资产相关的通信。虽然这解决了部分信息的理论问题，但它只适用于整个团队的大量支持。如果大部分生产人员更喜欢电子邮件或其他不太正式的通信方式，那么系统就变得难以信任。

10.8 节生产管理技术：评审工作

在生成票据之前，必须由客户或资深艺术家或主管审查资产。接下来，我们将看看进行评论的方式，以及可能用于这样做的技术：首先是电影，然后是游戏。

日报，前一天的工作评论，是电影制作的生命线。它们决定了生产节奏如心跳。有两种主要类型的评论：在艺术家的桌子上进行的评论，通常由主管进行;以及在放映室进行的那些。后者通常更正式，涉及更多人，通常包括客户自己。但是，基本要求是相同的：两种类型的审查都依赖于合适的人和正确的资产，平稳地回放这些资产的手段以及记录笔记的方法。

对于桌面审查，需要不太专业的软件。资产管理系统应该已经使艺术家能够找到他们所处理的资产，而他们的工作站应该足够强大以操纵模型或 Playblast 动画。但是，可能需要提供工具以便更容易查看镜头：例如，通过“洋葱皮”动画（前一帧和后一帧中的重影），或者从一个版本的镜头擦除到另一个版本。

标准工作站很少足够强大，无法快速显示全分辨率色彩校正图像，尤其是立体声。这就是为什么最终批准总是在放映室进行的原因。这种放映室评论更复杂，涉及更多人，需要更多准备。

首先，每个人都应该知道审核何时进行，谁应该在那里（通常只有部门的高级职员正在审核工作）。其次，为防止混淆，正在审查的每个项目都应在管道中具有唯一标识符。第三，应该有一个标准的系统，用于排队工作项目以供审查，包括（但不限于）将其转换为正确的格式并将其转换为审查系统的正确颜色空间，并将其分级为正确的磁盘使其以全分辨率平滑播放。最后，每个在场的人都应该有一个简单的方法来给予和记笔记。为审查中的每个项目打印注释表并将其交给在场的每个人，以便他们记下他们的贡献，这种情况并不少见。这些表格应包括上下文信息，例如同一镜头的先前迭代的注释，缩略图图像，编辑注释，VFX 注释和连续性注释。

放映室所需的软件取决于您正在使用的分辨率，位深度和帧速率，以及您是否使用立体声。当被要求快速从一个英雄帧跳到另一个英雄帧时，依赖缓存的系统通常会开始出现断言，但是可以在没有预先缓存的情况下播放 5 K 10 位立体声帧的系统需要大量功率和非常快的 I / O.

桌面评论的另一个不同之处在于演示显示器（投影仪）与控制显示器（显示器）分开。为了避免分散试图查看内容的人的注意力，用于控制回放的软件界面通常只在监视器上可见。虽然所有相关文件都应该提前排队，但评论是一个动态环境，这意味着人们可能会意外地要求查看旧版本的镜头进行比较，或者当前版本的不同镜头。一个好的评论系统可以很容易地找到这些内容，并在当前电影继续播放时排队。如果你可以在两者之间擦除，或者甚至用自定义的 alpha 贴图（例如棋盘）覆盖它们，以便从底部图像显示到顶部的方块（用于捕捉立体声错误），则可以获得奖励积分。此外，在编辑的上下文中应该很容易看到每个镜头。这意味着能够快速排队前一次和后一次拍摄，并能够打开和关闭“编辑手柄”（剪辑的开头和结尾处的额外帧）。

自从听取评论员的要求并采取行动（“我能看到上周我们看到的那个人物 X 从左边开始拍摄的东西吗？这就是他应该如何移动到这里！”）可能需要你全神贯注，一个人的唯一工作就是运行审查站，这通常是一个好主意。此外，虽然每个人都应该能够做笔记，但应该有一个主要的记录员，负责整理所有反馈并在审核后分享。这个记录者应该能够破译人们使用的经常隐秘的速记，并且能够明智地决定记录的用途。

应将注释记录在生产跟踪系统中，并记录它们的制造日期以及制作它们的人的姓名。一旦解决了任何变更请求，就应该可以将其标记为已解决。如果时间和技术允许，记录日报会话并提供与每个音符对应的视频/音频以及文本也是一个好主意。到目前为止，视频是艺术家在要求将特定建筑物向前拉时确切知道导演指向激光指示器的位置的最简单方法。

在游戏中，它都是关于迭代的。与电影不同，游戏工作室倾向于不使用日常评论过程，主要是因为艺术资产和显示它们的系统 - 即游戏引擎 - 一起发展。在开发周期的长期阶段，该显

示系统不可用或不完整。即使是这样，也很难有意义地评估资产：例如，如果驱动它们的游戏系统仍在开发中，可能很难对动画做出判断。

这是交互式编辑工具日益普及的一个原因，如 Unity，CryENGINE 和虚幻引擎等商业引擎提供的交互式编辑工具。这些工具允许艺术家通过游戏中使用的相同渲染技术查看他们的作品，使他们能够迭代内容，而无需冗长的构建过程，或者耗费游戏时间。使用交互式编辑器的团队面临的主要挑战是确保实现新功能，而不会在编辑器视图和游戏结果之间引入差异。出于这个原因，许多编辑器都包含模拟硬件平台之间差异的仿真模式。

尽管存在这些复杂情况，但每个工作室都有一些过程可用于审查资产，关卡和游戏玩法。许多工作室安排艺术家定期举行会议，与电影中使用的不同。然而，通常，真正的审查过程是“全部呼叫”或全公司范围的游戏测试。游戏测试让团队有机会了解游戏如何整合在一起，提供对项目整体状态的更好理解，而不是对个别资产进行艰苦的评估。

然而，游戏测试仍然是一个不完美的评论媒介，因为很少有游戏在发展的最后阶段稳定且功能齐全。如果没有经理的反馈，或者能够在上下文中看到自己的工作，那么在几周内处理资产是令人不安的。

减少艺术家盲目工作的时间的需要是许多游戏开发者接受敏捷开发模型的原因之一。在敏捷开发中，游戏代码和内容在简短的集中突发中一起演变：例如，图形程序员，游戏程序员和几个密切合作的艺术家可能会开发出一种新的视觉效果。每个合作者都会检查其他人的工作，结果直到下一次计划的测试时公司才会看到结果。

敏捷方法倾向于支持短期限和跨学科团队。这导致许多开发人员围绕小组（“pods”或“罢工团队”）而不是大型部门重新组织他们办公室的布局。这些天我们非常依赖电子邮件和其他形式的电子通讯，但没有什么比在椅子上旋转并与坐在你后面的同事交谈更好的了。小型团队和开放式座椅的组合在项目的早期阶段特别受欢迎，当时有许多未知因素需要探索。随着项目的成熟，或者管道已经很成熟 - 例如，在制作续集时 - 小组可能会逐渐淡化为更大的部门组。

将游戏分成可以并行生成的较大部分也是很常见的：通常是关卡。同样，这意味着将员工分成多学科团队，每个团队内的任务管理都是在流动的基础上进行处理。在这种模型下，只在项目层面进行高级管理和调度工作。

10.9 节生产管理技术：计划任务

由于记录每个文件的状态和开发历史的资产跟踪系统已经包含如此多的数据，因此不难看出为什么有些团队试图将这种方法扩展到项目管理信息。例如，跟踪资产 Y 是否准备好照明的同一系统也可用于告诉艺术家 X 设置灯光并在下周一准备好它们。沿着这条路线前进的系统通常最终会复制传统项目管理软件的关键元素，例如微软的项目：它们会跟踪分配给谁做什么和何时做的事情;生成报告，显示经理项目的哪些部分滞后，以及资源闲置的地方;并且可以查看和安排依赖项。

这类系统的成败在很大程度上取决于工作文化。全公司致力于精确流程跟踪的工作室可以从集成管理中获得很多价值。首先，集成系统可以自动化许多常见的通信过程。例如，艺术家可以通过检查已完成的文件，通知他们的主管资产已准备好进行审查;而主管可以通过在系统中输入注释，将资产重新投入艺术工作人员进行进一步的工作。以这种方式生成的新工作甚至可以自动记录，以适应工会或供应商会计要求。

其次，自动跟踪系统可以在规划未来项目时生成重要信息。如果有关于以前产品实际成本的可靠实证数据，那么制定合理的人员配置计划要容易得多。艺术家对于解决工作需要多长时间的估计是众所周知的乐观，并且忽略了许多创造性的曲折，技术问题以及延迟现实世界任务的随

机事故。当与从版本控制系统自动收集数据的能力相结合时，良好的任务跟踪工具可以为需要为更多可实现的数字集预算未来项目的人提供支持。

但是，端到端跟踪软件通常与瀑布式方法相关联，其中强调规划和事先分配资源。就像一般的瀑布式开发一样，自动化系统会产生详细知识的幻觉 - 如果系统与团队真正的工作方式不匹配，这是一种危险的错觉。如果艺术家觉得系统试图对它们进行微观管理 - 或者更糟糕的是，它正在监视它们 - 它们将很快停止与它合作。跟踪信息的名义受益人也可能不想仅仅因为它与资产数据库集成而致力于新系统。很难说服已经熟悉特定日程安排工具，报告程序或电子表格的资深管理人员转向集成系统。

游戏中常见的端到端跟踪变体是使用专为跟踪错误而设计的系统。错误跟踪软件通常提供将工作分配给各个团队成员的选项，以及最小的完成跟踪功能。例如，一个错误可能来自测试人员，他注意到特定角色动画中的缺陷。然后可以将此错误分配给首席动画师，该动画师将审查问题并确定它是否反映了游戏代码或艺术本身的问题。在前一种情况下，他们会将其提交给首席程序员;否则，它将被传递给动画师进行修复。修复错误后，会向原始测试人员发送通知以进行验证。但是，错误跟踪系统通常不适合时间管理：它们的主要目标只是确保问题得到修复而不是被遗忘，而不是在未来的最后期限内安排。

10.10 节生产管理：最后的想法

鉴于现代化生产项目的规模和复杂性，管道团队必须为管理人员提供尽可能多的支持，这些管理人员将需要处理数以万计的个人任务和参与项目的数百名艺术家。在大型项目中尤其如此，许多员工只是作为承包商或外包商经过：当你的一半艺术家离开大陆时，很难依靠“走动管理”。

同时，您创建或部署的管理软件为生产增加价值至关重要，而不是增加一层不受欢迎的官僚机构。正如我们在本书中多次强调的那样，团队的创造力是一个至关重要的资源，你不想把它浪费在世俗的文职任务上。

也许管道团队在构建报告系统时可以做的最重要的一件事就是从将要向系统添加信息的艺术家和将使用它来监视项目的管理者那里获得支持。一个没有人真正使用的炫目软件是一个远远不如维护良好的 Moleskine 笔记本的有效工具。在这方面，流程管理工具就像管道中的其他工具一样：它们与团队接受的程度直接相关。

插曲：音色

颜色和声音是任何电影或游戏的组成部分，但该过程的许多方面仍然是视觉特效制作的一个谜。以下是各种专家在其视角领域提供的信息集合，从颜色开始，然后转向电影中的声音，然后是游戏。

第 10 节插曲 1：工作流中的颜色管理

工作流设计中有时被忽视的因素是需要在显示内容的任何地方标准化显示和使用 RGB 图像文件。今天任何生产的关键部分都是通过图像进行通信，在网络化，虚拟化的生产环境中，创建，批准，审查和观看新作品的所有人都在看到和谈论同一件事情。

有时候，这很简单。如果所有内容都是针对固定设备（例如 iPhone）制作的，那么只需在 sRGB 色彩空间中工作并将所有内容保存在该设备上即可查看。但是，要为各种不同的显示设备制作内容，必须建立生产色彩管理策略。

考虑如何从创建时刻开始查看内容，直到将内容发送给消费者。将显示哪些类型的显示器？它有多少种不同的可交付成果？什么是最好的质量版本？内容可以从单个主元素传递，还是必须创建多个版本？使用什么软件或硬件来捕获和操纵颜色？在日光条件下，内容是否被近距离观看，还是在黑暗的剧院中播放？所有这些都将导致选择颜色空间和需要创建的输出质量。

游戏也有特殊的考虑因素。鉴于消费者监视器的广泛功能，您如何管理黑暗场景的可见性？是否有足够的对比度可以看到阴影细节或黑色的灰色文字？你能利用更多彩色 LED 显示器吗？现在 UHDTV 的未来色彩空间怎么样？如果您的目标是真实，那么如何渲染最准确的物体，角色，布景和光照条件？

所有这些问题都导致了对色彩科学问题的考虑，因为如果不对这项技术有所了解，就很难创造出艺术。

图像工作流设计

无论您是否意识到图像，都会在色彩空间中创建图像。您可以查看它们并在颜色空间中对它们进行操作以获得所需的结果。无论是摄影还是计算机图形，色彩空间决定了可以显示的色彩范围，并设置了在监视器上再现的亮度的色调比例。对于许多基于计算机的应用程序，常见的颜色空间是 sRGB。对于视频应用，ITU Rec BT.709 很常见，UHDTV（Rec BT.2020）将很快出现。对于电影制作，各种各样的色彩空间发挥作用：相机 RGB，自定义显示器设置，DCI-P3 投影，激光投影仪原色等。

工作流的设计需要考虑正在使用的图像文件以及文件为工作流的每个步骤表示的颜色空间。这从源图像的颜色特征开始。如果使用数码相机拍摄，了解相机的光谱色彩灵敏度，或使用相机文件的校准色彩空间可以帮助确定源参考色，以便以后进行转换和操作。在各种工作流程中通常转换为“工作色彩空间”以在设施内或在特定软件包内使用。不正确的转换可能会无意中限制源图像中可用的色彩空间的范围和质量。例如，当生产工作流程围绕图像在特定监视器上的外观设计时，就会发生这种情况。专门针对该监视器的色彩空间操纵图像可能限制色域的动态范围和宽度，使得内容在具有更大能力的其他媒体上不太有用。这种工作方法被称为“输出参考”颜色，并且如果诸如电影，数字投影仪和 UHDTV 的各种显示媒体是内容的目标媒体，则可以是限制性的。

工作流设计还需要考虑“信号”（线上图像）从一个设备（或软件包）移动到另一个设备（或软件包）的方式。在硬件情况下，这需要一些计算机输出和显示器输入的知识。是 HDMI，HD-SDI 还是 Display Port？有许多时候需要转换信号，这些也必须进行检查和管理。出现这种情况的常见情况是需要应用 LookUp 表（LUT）以将其原始捕获形式的图像外观转换为用于显示的颜色呈现版本。

设置和校准

显示设备通常需要校准以保持一致性，特别是，监视器的亮度应设置为已知且一致的亮度。对于大多数视频应用，这是 100 cd / m。如果您的工作流程设计依赖于 WYSIWYG“所见即所得”，则所有显示都需要彼此保持一致。测试图表或颜色条对于提供监视器按预期执行的可视检查非常有用。

交货

可以在各种各样的显示器上查看内容，并且完整的交付列表对于确定如何在所有媒体上制作和提供色彩正确图像是至关重要的。颜色是预先设计用于渲染，还是之后使用颜色分级进行调整，校准显示和适当的元数据以识别每个文件的工作空间对于保持准确的颜色至关重要。需要在工作流设计中规划和管理自动转换和手动“修剪通道”以管理色域差异。当输出介质是胶片时，需要特别考虑，因为它是一个减色的色彩空间，与普通观察显示器上看到的完全不同。建议在整个胶片预定工作流程中使用胶片预览 LUT，以在这种情况下获得最佳效果。工作流设计的目的是确保正确管理所有工作步骤，并且可以维护最佳图像质量并向生产中的所有用户显示。

色彩管理系统

虽然大多数图像以某种形式的 RGB 色彩空间显示，但输出显示的设置种类繁多，这可能会对图像的外观产生很大的影响。要理解的最重要的一点是颜色空间编码：要生成的颜色与表示颜色的像素中的代码值之间的关系。对于图像中的每个代码值，存在应该基于颜色原色再现的独特颜色。颜色编码还定义“传递函数”，线性 RGB 空间中的颜色与显示设备的输出伽马（或其他曲线）之间的关系。可以使用几种不同的 RGB 编码：线性光，视频伽马和日志。这些中的每一个都有优点和缺点，并且它们都是常用的。线性光与现实世界中光的行为类似，可用于合成，CGI 渲染和成像传感器的转换。 Log 对于保留整数系统中的各种值很有用。视频伽玛非常适合在显示器上显示 RGB，因为它可以合理地匹配人类视觉系统的性能。有许多不同的 RGB 编码，它们可以通过 LUT 和颜色转换矩阵的组合来回转换。

在大多数桌面系统中，已经有一个明确定义并可在大多数现代操作系统上使用的色彩管理系统：国际色彩联盟（ICC）配置文件系统。该系统允许通过分配 ICC 配置文件来定义图像文件中的颜色，并且还允许在已经用其自己的输出配置文件表征的输出设备上自动转换和查看图像。

电影业缺乏类似的生产色彩空间能力，涵盖高动态范围和宽色域。美国电影艺术与科学学院一直在为这些独特的要求开发色彩管理系统，现在称为 ACES 系统（学院色彩编码系统）。

ACES 系统

ACES 是一种图像和色彩管理架构，主要用于电影和电视的制作，母带制作和长期存档。ACES 提供了一组数字图像编码规范，转换和推荐实践，可以为各种输出设备创建和处理高保真图像。

工作空间

ACES 是一个工作的 RGB 颜色空间，包含整个可见颜色集，提供超过 30 个动态范围的停止，并使用基于 OpenEXR 文件格式的线性光半浮点编码。 ACES 中的线性光意味着，尽可能地捕获原始场景颜色并将其保存在 RGB 文件中 - 称为场景引用颜色。已知色彩空间中的其他图像可以容易地转换为 ACES，从而允许混合来自包括 CGI，数码相机和胶卷的大量不同来源的元素。

输入图像

对于可能是 ACES 源的每种介质，必须具有输入设备变换（IDT），其提供从捕获的图像到 ACES 的颜色校准转换。对于胶片扫描仪，有一种称为 ADX（学院密度编码）的光密度色彩空间，它也可以很容易地转换成 ACES。为视频输出或计算机显示器创建的图像也可以转换为 ACES，但此路径会失去系统的高动态范围和宽色域功能，因为您只转换最初在显示器上看到的范围和颜色。

查看转换

就像历史电影系统的负片和印刷品一样，ACES 色彩空间旨在保持最宽的纬度和精度，但看起来不正确，就像负片看起来错误一样。观看 ACES 文件的一个重要部分是观看变换，它为暗环绕的戏剧观看环境增加了对比度和饱和度。这称为参考渲染变换（RRT），它可以在任何地方提供一致的可再现图像。所有 ACES 图像都是通过 RRT 查看的。最终可交付成果可以从此输出到其他设备和查看条件。

输出图像

校准输出设备的特性在输出设备变换（ODT）中描述，其将大范围的 RRT 图像压缩到显示器可以再现的可见部分。在实践中，RRT 和 ODT 被组合并变成用于特定类型的输出设备的 LUT。可以在 ODT 中创建自定义色域映射策略，也可以进行电影录制的转换。 ACES 值可以转换为 ADX 胶片密度以进行胶片记录。如果项目已在另一个输出介质中完成（“输出参考”），则可以通过使用从 RRT / ODT 到创建特定颜色的 ACES 值的逆变换将其转换为 ACES 文件。

档案文件

ACES 系统具有单个 RGB 色彩空间，单个视觉变换和一组明确定义的变换，允许在知名状态下保存内容，以帮助归档用户，并保留所有原始内容捕获动态范围和颜色。

ACES 支持灵活的图像流水线开发，适用于许多不同的流程，包括胶片和数字采集，CGI 和 GPU 渲染，数字中间，视觉效果制作，重新制作和现场色彩管理。

色彩的未来

很长一段时间以来，视频和电影都依赖于一致明确的色彩标准，这些标准至少可以使用五十年。新技术，数码相机，OLED 电视，HEVC“BluRay”，激光投影仪，3D 电视，UHDTV（4 K 和 8 K）正在推动更大的真实感，包括分辨率，高帧率，色彩和动态范围。

新内容的创作者需要仔细思考今天制作的数字材料的寿命，因为明天看起来更明亮，更丰富多彩。

第 10 节插曲 2：电影声音文件生命中的一天，大约 2013 年

无论是动画还是实时动作，动态影像的声音都将以录制某些对话框开始。该对话框可能只是一个助理摄像机人员宣布场景并拍摄一个场景，其中包括一个人走动或只是一些房间音调，但如果记录的第一件事不包括对话，那么很快就会记录一些对话。对于动画功能，声音通常记录在录音室中。对于实时动作，它可能在声场或现场。无论是室内还是室外，对话框都将以数字方式捕获。无论是在工作室的工作站还是便携式录音设备上，声音都会被数字化并存储到计算机文件中，从创建它到完成制作的那一刻起，它就有很长的路程。

现行标准

目前的电影录音标准是 Wave 或 Broadcast Wave 文件(分别为 WAV 或 BWF),采样率为 48 Khz。16 位录音是可以接受的，但 24 位是标准。 16 位文件（通常是一个错误）被快速转换为 24 位。用于电影的 WAV 文件应包含未压缩的线性 PCM 音频数据。48k / 24 位音频每曲目每分钟约需 8.24 MB。

WAV / BWF 文件在很大程度上是可互换的，主要区别在于为“专业”用途添加的元数据的附加组块（“bext”组块是主要的一个 bext 是“广播扩展”的首字母缩写）。这些文件是最初由 Intel 和 Microsoft 创建的 RIFF 文件格式的变体。在历史上，这些文件有一个致命的缺陷：它们的实际文件大小限制为 4 千兆字节（在许多情况下为 2 千兆字节）。我们将在下面看到为什么这很重要。 WAV 文件是 mxf 和 AAF 文件组的可接受成员。

WAV 文件通常是“多渠道”组的成员。实际上，这可能是一个环绕音频主机，有左，中，右等通道。这些可以是左声道名为“myfilm_ v1_6chmaster.L.wav”的单声道文件，另外还有五个文件添加.C ,.各种环绕声道的 R，.Ls，.Rs，.Lfe。问题是文件可能彼此分离，或者如果其中一个文件在磁盘或服务器上重命名，则可能很难修复。 Polyphonic WAV 文件可以包含许多通道。通道可以是环绕声轨道，以制作音频母带或来自记录器的许多轨道捕获胶片组上的声音。在动画或 ADR 录制中，通常会录制多个频道。轨道可以是一个靠近的麦克风，或者是一个非常接近演员的远距离和一个小的领夹式话筒，以模仿机组的机身麦克风。不是命名音频曲目.L，.C，.R 等，而是简单地将文件命名为.1，.2，.3 等。文件大小限制可以在这一点上开始。将所有通道捆绑在一起非常方便，但对于 8 小时音频主机，两小时功能的数学运算如下：

8.24 MB per minute X 8 channels X 120 Minutes = 7,910.4 MB or approximately 8 gigs .

这对于标准的复音 WAV 文件来说太大了。

虽然存在将文件“链接”在一起的标准，但文件彼此分离或重命名的实际问题仍然存在。有一个名为 RF64 的新标准文件（基本上是带有新标题数据的 WAVE 文件，表示“我是 64 位”），它充分利用了 64 位操作系统的能力。文件大小的实际限制是 16 艾字节或 16,000,000,000 千兆字节，希望在可预见的未来足够大。有关各种格式的更多信息可从 SMPTE，EBU，AES，AMWA 等在互联网上获得。

对话录音

回到我们的对话录音。我们从场景或序列中记录一行或一系列行。怎么办？录音将由工作站或专用位置录音机捕获。位置记录器 - 由 SoundDevices，Zaxcom，Aaton，Nagra 等公司制造，

可以直接生成 WAVE 文件或在需要时导出。这些文件可能会或可能不会被命名为非常有用的东西。虽然这有待改进，但可能需要重命名文件。这些记录器还可以将元数据嵌入到关于场景，拍摄，频道 ID，频道内容等的文件中，这些文件可以在以后查看和使用。这可能是至关重要的，因为文件可能在设备 Sound0001_001.wav 上命名，它没有完全从屏幕跳跃并尖叫“我跟踪场景 1 中的一个，拍摄一部电影中的 1 个并且我包含一个 Boom 麦克风的录音“。文件通常被重命名为“CL2_0001_001_1.wav”，即“Cloudy 2，Scene 1，Take 1，Track 1”。嵌入式元数据听起来很精彩，并且有许多实现显示出承诺，但它在实践中可能非常不稳定，因此文件名 - 好的 - 是可以依赖的一件事。在工作站上进行的录制更容易命名，并且可以在会话结束时重命名，或者在此之后立即重新命名以解决可能出现的任何问题。动画项目的文件名可能看起来像“CP_0600_004_ AF_02.wav”，这是“Cloudy Principal，Sequence 600，Shot 4，Anna Ferris（演员正在说话，以防万一发生变化，或者如果它是临时线，而制作等待为了有机会记录实际的演员，取＃2“。

文件名可以对作品产生无法估量的影响。如果做得好 - 并且可能更重要 - 一致地完成，人们可以遇到一个文件并知道它包含什么。如果做得不好，问题就会很快开始，似乎永远不会结束。许多人对文件命名采用了理论，你应该能够将所有文件放在计算机上的一个文件夹或目录中，按名称对它们进行排序，它们应该按内容和顺序或卷轴进行分组。

录音下一步去哪儿了？在实景制作中，到实验室，在通往切割室的途中，它将与图像“同步”。存储集合中的文件，然后传递给声音部门以用于后期制作。动画声音文件直接进入剪辑室。如果正在录制多首曲目，Polyphonic 文件是首选，因为它们被大多数 NLE（非线性编辑系统）（如 Adobe，Avid，FCP 等）作为单个项目处理。这是一个需要进一步检查的关键时刻。

在功能制作中，声音和图片（曝光的负片或来自相机的文件/磁带）通常被发送到实验室进行某种“处理”。无论过程是电子过程还是化学过程，我们的目标都是为未来的漫长旅程做好准备。声音和图像相互连接并相互关联，希望它们彼此同步运行。以每秒 24 帧拍摄的图像和以 48 赫兹记录的声音很好地结合在一起。 “采样率”可以达到每帧 2000 个样本。使用时间码排列并且在镜头的头部或尾部拍板的“棍棒”的共同音频视觉参考点进行验证，建立同步关系。结果可以传输到文件（或磁带或两者），以便在日报中查看，也可以发送到剪辑室开始编辑过程。

通常，发送到图片编辑的文件的分辨率低于主媒体。图片文件可以是适用于 NLE 的中间编解码器。声音可以是由制作混音器准备的“混音”或在电视电影套件或实验室中产生的混音。可能存在由单个混合轨道表示的许多声道。混合可以是“分离”轨道，其中涉及多个字符的场景被分配给各个轨道。例如，轨道 1 可以包含悬臂式麦克风，轨道 2 可以包含无线或车身话筒，以实现更大的隔离。混合或分离混合轨道可以是在集合上记录的许多轨道的代理。八个声音轨道并不罕见。例如，如果镜头是音乐会，那么很容易就会有二十四个或更多的音轨。如果图像编辑器需要比混合轨道提供的声音更多的控制，则可以在 NLE 中加载和使用附加轨道。

NLE 保留源的数据库以允许从集合中追溯到“主”元素。整个编辑过程都使用此数据库。视觉效果部门不断使用主视觉元素来创建制作所需的图像。同样，声音部门将在创建最终音轨时使用原始主声音元素。由人类创建和维护的数据库的质量极大地影响了流程的简易性。同样，此时削减的角落永远不会停止为所涉及的每个人创造问题。

关于“24 fps pix 和 48 khz 声音很好地排列”的早期声明是正确的。如果编辑部想以 23.9 FPS 工作怎么办？声音仍然以 48 千赫兹的速度运行（与过去的 NTSC 视频一样，不是 47952 赫兹）那么呢？有时，制作混音器将以 48,048 赫兹的速度捕捉声音，这样声音可以“减速”以与较慢的运动图像同步运行。需要在生产开始之前制定这种安排，希望如此。 “每帧样本”的这种差异是当事情“失去同步”时最常见的罪魁祸首。用于以 23.9 fps 图片播放的 24 fps 图片的 48 k 声音会漂移或“走路”不同步。片段越长，它们就越分开。当一个片段只有几秒钟且没有适当的领导者和同步弹出片段时，这很容易被遗漏。

很多时候，图片和轨道的各个部分往返各种设施，进出各种缺乏真正同步参考的平台。过去，需要通过电视电影投影或传输胶片和磁声片段。它们在物理上具有相同的长度，并且在视觉上被标记以允许它们同步。同步“pop”（通常为 1 khz 音色的一帧）在电影的同一时刻或 SMPTE 倒计时领导者的“2”中播放，确保观众事物处于同步状态。同样地，在片段的结尾处，另一个“弹出”将在图片的最后一帧之后播放三英尺（两秒） - 再次在屏幕上伴随着打孔或“2”。这允许编辑和技术人员验证同步。随着我们进入数字时代，“2 流行音乐”失宠了。可能是出于无知或节省空间 - 我不知道为什么 - 但同步通常会被“假设”。关于“假设”的古老格言适用。此外，墨菲定律似乎在电影制作中得到严格执行，所以要小心。需要通信，清晰的标记，标准化，适当维护的设备以及测试和验证，以保持同步。希望事情好转，很少有效。

随着制作的进行，编辑开始从原材料创建场景或序列。在动画制作中，这将涉及根据声音的录制创建故事，带有声音效果的静止图像和根据需要添加的音乐。随着“剪切”的进行，编辑器可能需要将场景或序列发送到声音或视觉效果以便完成工作。也许镜头会出现在动画制作者的动画制作过程中。声音部门可以创建将用作动画参考的声音。这些通常由图片编辑器添加到剪辑中并发送给动画师或 VFX 艺术家。更常见的是，动画师和视觉特效艺术家将图像发送给图片编辑器，图片编辑器又与声音部门进行通信。在任何一种情况下，对同步和技术卓越的永恒警惕是其自身的回报。

我们在集合或录音室的原始录音继续在 NLE 中使用。每次播放序列或发送其他作品时都会包含它。随着生产的进展，可能需要混合。无论是临时混音还是准备最终混音，都会继续引用原始主唱片。图片编辑器会将图片的剪切“翻转”到声音部门。翻转通常包括具有嵌入式混音的视频文件，来自 NLE 的时间线编辑的 AAF 或 OMF 翻译，以及可能追溯到原始音频文件的 EDL。原始声音文件通常不包含在 NLE 中，但由它引用。如前所述，NLE 可能只包含代理音频和视频文件，而 EDL 允许各个部门使用原始主元素，无论它们是音频还是视觉项目。

声音部门将准备编辑的曲目，以便用于制作项目的最终版本。利用 EDL，他们可以追溯到音频母带并使用所有可用的通道来准备混音，就像实验室或视觉效果部门将在需要时追溯到原始相机材料一样。声音部门经常使用图片部门翻过的一些文件。这些文件可能具有某些独特的特征，这些特征要么不容易复制，要么效率低下。在这样的时候，经常使用“听起来不错 - 好”的格言。

在准备最终混音时，音响部门将创建通常随后“预混合”的音轨。此时使用原始录音 - 可能是最后一次。预混合或“预配音”的过程允许混音团队从声音编辑中获取编辑过的曲目并准备用于最终混音。在预混合期间，混音器将调整音量，均衡和动态范围，以及可能选择平移位置 - 声音将播放的扬声器。这些预混合用于代替原始主文件。对话，ADR，组 ADR，声音效果，背景效果，Foley 等都会有预混音。预混过程的主要目的是在最终组合中使事情更易于管理。花费三十或四十条 Foley 的时间可以花费时间，所有这些都需要各种治疗，然后将它们变成需要稍微调整的东西。

在最终混合期间，进一步调节预混物以制成“茎”。茎包含预混合物组。最低限度，会有对话，音乐和效果。这些词干将被格式化以匹配预期的最宽版本格式。例如，在 7.1 胶片中使用的杆将具有用于左，中，右，左环绕，右环绕，左后环绕，右后环绕和 Lfe（低频效果）或“悬臂”的通道。当所有的茎一起玩时，它们构成了成品混合物或“Printmaster”的基础。Printmaster 这个词实际上更具历史性，因为它指的是将在“Print”版本中使用的音轨。它现在有点简单，因为一个功能通常会在世界各地以多种格式同时发布。许多人已经开始将 Printmasters 称为“大师”。您可以将茎视为声音的“层”，它们共同形成最终混音，就像需要合成形成最终视觉效果的所有视觉元素层一样。

当为后续版本的胶片制作母版时，“宽”的茎很有用。例如，删除英语电影的对话框形成了电影的“外国”版本的基础。其他示例包括使用 7.1 主干制作 7.1 家庭影院母版或 5.1 混音或

杜比立体声兼容母版或单声道 DME 混音等。
Premixes，Stems 和 Printmasters 的 Polyphonic WAVE 文件通常被命名为：
CL2_01_v12_0411_DX_Premix
CL2_01_v12_0411_DX 7.1 Stem
CL2_01_v12_0411_7.1_Printmaster

这些是可用于卷轴 1，版本 12 和/或 4 月 11 日版本或 Cloudy 2 的所有文件。版本号和/或日期允许文件的用户期望它与相同版本的图片同步运行。这些文件可以存档，将来的用户将知道声音文件的内容。这些文件名是“生产中”使用的典型文件名。工作室或经销商通常会有关于声音元素最终传送的规范，这些从演播室到工作室的情况很少相同。在整个制作过程中使用一致且清晰的文件名的重要性不能过分强调。

第 10 节插曲三：实拍与动画的音频差异

实时动作与动画中的音频之间存在显着差异，从捕获声音到应用声音的方式。从实时动作开始，第一个问题之一是除了在集合上捕获的音频之外还添加了多少音频。传统上，只有大约 20％的声音录制在电影的最后一部分。今天，大型预算电影可能仍然适用。其余的作为 ADR，声音效果，Foley 和背景添加。声音设计旨在成为每个场景声音的自然表现。效果可能会被修饰，但目的并不是要让电影观察者摆脱体验。

另一方面，技术使电影制作人能够以较低的预算制作（可销售）电影。无论是真人秀节目还是低预算恐怖电影，更多的电影都在跳过对话框替换和声音设计，只使用制作声音。

那么这些不同的音频文件，流和轨道是如何管理的呢？在电影世界中，声音和图像是独立的实体和独立的过程，直到展览的那一刻。对于胶片上的释放，声音从其磁性形式转移到光学形式。现在有一个关于声音的真实“图片”;预数字立体声轨道看起来像是连续示波器波形的图像，数字声道看起来像编码的黑白点。光学声像与图像一起曝光，成为复合打印。此时，声音与图片结合。投影仪读取图像并将其转换回电脉冲，以便在声音系统中处理。对于数字电影的发布，最终的声音和最终的图片汇集在一起。此时，声音与图片结合。

我们在 NT Audio 做的大部分工作都处于后期制作过程的最后阶段。工作室必须保留其资产的价值。最终的音轨转移到 35 毫米磁性薄膜或 LTO 磁带或备用硬盘驱动器，最终图像转移到黑白 35 毫米胶片作为分色（称为 YCM）或 LTO 磁带或备用硬盘驱动器。为什么要用胶片作为保存介质？有利有弊。电影价格昂贵，未来很难获得，但它可以持续一百年，并且仍然很容易设置，以便在未来的任何平台上播放。驱动器很便宜但可能仍然无法长期运行或兼容。与 LTO 一样，迁移是每五到十年强制执行的。

显然，完全动画电影使用数字媒体。根据我的经验，完全动画电影的音乐曲目在整个过程中更加稳定。它经常取代背景。当混合音频和对话时，在图片编辑仍在进行时最终确定声音通常是低效的。在某些时候，无论是按计划施加还是创作过程，镜头，场景，卷轴，电影，都必须完成并“锁定”。声音工作人员致力于将声音与最终的画面进行匹配，并将这些声音混合在一起。对图片的任何进一步更改现在将影响许多同步的声音元素并危及时间表。声音工作实际上是为了支持和/或增强视觉体验。传统上，所有图片工作需要在合理的工作向前发展之前确定。

在为外语配音时，您必须记住动画的嘴部动作是在导轨上完成的，然后使用的实际对话应该类似于指南。对于外语配音，同步尽可能放置，通常在嘴的末端与嘴部运动对齐。

第 10 节插曲 4：游戏音频管道

音频占用了大量空间。玩家将遇到的游戏的每个方面都有音频因素，因此需要管理很多文件。并且文件可以是相当大的流音频资产，例如音乐，氛围，语音和过场动画，即使在有损压缩的情况下缩小也具有相对大的文件大小。不仅如此，电子游戏的互动性意味着音频需要改变并适应不断变化的环境（例如控制音乐强度和进展的游戏状态，或影响环境的昼夜循环），这会增加音频的数量需要创造令人满意的体验。即使是简单的声音效果，例如脚步，虽然相对较小，但需要多种变化来对抗重复，以及与角色可以行走的每个表面相关的独特样本集。这一切都增加了音频拥有游戏中任何组件最大的足迹之一 - 视频是这个王冠唯一的另一个真正的竞争者。

虽然游戏项目的规模和范围将影响其音频管道的复杂性，但高级工作流程仍然基本相同：

源音频资产创建。 首先，未压缩的音频文件由声音设计者或作曲家创建。

音频资产设置。 然后将这些文件配置为在游戏中使用。

游戏内音频资产构建过程。 然后创建压缩的游戏内资产。

音频资产的实施。 最后，资产与游戏中的事件相关联。

让我们更详细地探讨这些步骤，并考虑典型游戏音频管道中遇到的一些常见问题。

源音频资源创造

虽然拆分管道并以线性方式描述它使得更容易掌握，但实际上每个步骤都是深层交织并影响另一个。例如，如果不理解它们最终将如何实现，就无法创建这些源音频资产 - 这种“鸡蛋和鸡蛋”场景意味着，对于项目的大部分内容，这两个方面将一起快乐地向他们的方向发展最终的游戏体验，工作流程和管道。

同样，人们很容易认为，对于构建工程师，IT 经理或技术总监而言，管道的这个初始方面并不存在任何问题。但是整个流水线和流程不仅影响通过它的结果内容，而且在此过程中每个步骤仍然考虑到所有各方：

需要将这些源资产签入游戏的源代码控制存储库。它们是构建过程所需的依赖项 - 如果没有这些文件，则无需构建。

对于内部音频人员，即创建您有责任支持的资产的人员，您需要一个备份解决方案来完成创建这些源资产的所有工作。 虽然对于艺术家来说，检查他们的完整来源（例如 Photoshop 或 3 ds Max 项目）并不是闻所未闻，但对于音频人员的工作并不是一个好主意 - 个别音频项目（例如单个音乐）可能是几千兆字节 在规模上，如果团队中的每个人都在同步这些数据，那就是浪费大量的浪费时间和网络带宽，更不用说那些在异地工作的穷人非常沮丧了！ 因此，一个单独的存储库只是为音频人员备份他们的工作是明智的,如果音频团队中有多个人并且他们的职责重叠，那就非常重要。 只要密切注意你的服务器空间 - 音频一旦达到它的步伐就会有快速增长的习惯。

音频资源设置

此过程涉及音频团队将各个音频资产分组为复杂的音频事件并设置其参数 - 音频事件是游戏将触发的内容，以便发出一些音频，但音频如何表现（以及它是否决定均匀完全播放）由这里如何配置事件定义。它可能是一组声音，从中随机选择一个声音并以随机的音高变化播放（如前面提到的脚步示例），或者它可能是一个复杂的动态事件，根据两个不同的游戏状态而变化进

入它（例如，当它在空中摆动并被玩家操纵时，擒抱钩绳的旋转速度和长度）。

这项工作最常见于交互式音频“设计工具”，它可能是一种专有技术，也可能是中间件解决方案 - 游戏音频技术通常由运行时音频引擎和设计工具（或软件套件）组成。用于设置和配置音频及其行为。这里有几件事需要考虑：

此音频项目元数据需要签入游戏的存储库 - 构建过程也需要它，并转换为等效的游戏内二进制资产。

即使只有两名音频人员试图每天访问它，设计工具的项目文件也很容易成为工作流瓶颈。虽然项目文件可能是可合并的（如果它以基于文本的格式存储，如 XML），这很容易出错并且不便于用户使用。 如果您的设计工具没有针对多个用户的解决方案，那么使用多个项目文件会分成合理的类别（音乐，氛围，声音，生物等，或者如果适合您的游戏，甚至只是好的老式'级别'可以大大改善这种状况。 有很多方法可以为这只猫提供皮肤，但理想情况下，你想要允许粒度，以便许多人可以同时处理音频，同时还提供某种高级控制，允许整个类别的声音和 音乐很容易调整（游戏离线和实时）。

游戏内音频资源重建过程

这是事情变得有趣的地方！虽然较小的游戏项目可以直接引用和加载其所有音频文件 - 甚至可以在交互式音频设计工具中使用并选择在游戏代码中手动设置所有内容 - 对于大多数游戏项目，音频资产将构建为压缩的“ soundbanks“（打包的压缩声音文件的集合）。这可以从整个游戏的一个声音库（由所有声音效果组成的存储器驻留库，可能直接访问流文件）到每个音频事件的一个音库（即数百个音库）以及中间的任何地方，取决于关于如何分割音频以供游戏访问。基本上，对于手头的游戏来说最好的一切都很好，但是最小化人为错误的平滑工作流程应该是首要任务。

从历史上看，音库是完整加载的，因此音库的内容与游戏加载方案的要求之间存在直接关系。手动管理是困难和有问题的 - 如果你忘了将声音放入音库，或者没有足够的内存来加载音库，则不会加载声音以供游戏播放。随着游戏的增长和大型流媒体世界成为可能，在自动构建过程中进行了投资，以确定给定位置所需的声音并将其打包以优化加载。如今，自动化银行构建，动态内存管理和良好的诊断工具为音频设计师创造了一个强大的游乐场，可以通过声音和音乐将虚拟世界变为现实。

但在中小型项目中，音频团队手动配置和管理音库仍然是必要的。这听起来并不像听起来那么毛茸茸 - 现代音频引擎能够通过将声音拉出音库来加载声音，这意味着你可以避免使用大量无法管理的声音。然而，将所有内容都安装到内存中仍然是一种需要考虑，规划和技术解决方案的约束。

与任何构建过程一样，周转时间是最关键的因素之一 - 音频内容创建者和实现者在游戏中听到他们的工作所需的时间越长，他们能够越慢地迭代它并且它们的生产效率越低。一个构建过程，要求他们等待游戏的完整部署（如果音频打包和构建时）是一个破碎的，而不是时代的背后。具有讽刺意味的是，这更适用于具有复杂构建过程的大型项目。理想情况下，内容创建者应该能够在本地构建他们的更改，以便他们可以在提交工作之前预览和测试它们 - 小项目的一个优点是通过手动构建音库，他们能够做到这一点。通常，需要重新启动游戏才能听到音频内容的变化，但实时更新是可能的（虽然技术上很棘手且有点笨拙）并且变得越来越普遍。

信任音频团队总是完美地提交最新的音库并不是一个万无一失的解决方案，因此游戏部署构建过程的混合方法将基于最新提交的音频项目文件构建最新的银行可能是有用的。

音频资产的实现

游戏中有多种音频实现方法：
直接在游戏代码中调用声音事件
将对声音事件的调用插入脚本
为复杂的音频实现创建定制的解决方案（例如，设置交互式音乐转换，虽然中间件在这里越来越有用）
为常见任务创建数据驱动的音频系统（例如轻松设置和标记物理音频）
使用关卡编辑器将声音事件和触发器直接实现到游戏环境或关卡逻辑中
使用编辑工具将声音事件实现到其他系统中，例如动画

任何能够在不占用编码器时间的情况下实现音频实现的方法对所有各方和项目质量都有明显的好处。但也许值得您关注最后两种方法，这些方法基于音频功能的概念被添加到已有的技术（例如关卡编辑器或动画工具）。当音频由另一个系统驱动时，这是有意义的，并且是将两者结合在一起的好方法。访问瓶颈是这种解决方案的固有内容 – 如果将音频事件触发器添加到资产（无论是级别，地图区域还是动画），那么音频实施者只需要访问该资产即可变化。过程在解决这类工作流冲突方面发挥着重要作用，但将音频实现分离到一个单独的可编辑层并不是闻所未闻 – 这可以起作用但它不是一个神奇的子弹，因为它意味着对资产的更改不会更新音频层并在无知音频的情况下完成，这实际上最终会产生比解决的问题更多的问题。这里的正确方法实际上取决于项目，团队和工具。

一旦音频体验的核心已经实施，可以在生产结束时发生的最激进的变化之一是本地化资产的交付和实施。这可能会影响包含语音的每个资产，这恰好恰好是游戏中最大的文件之一，并且会将其占用空间乘以支持的语言数量 – 这些日子对于控制台发布来说十六种语言很常见，并且数量不断增加随着全球新市场的出现而不断增长。用于分析和比较本地化资产与主要语言主资产的工具非常宝贵，有助于保持跨语言的一致性，并有助于查找错误或错误命名的资产或长度错误的恶意文件等问题。一个自动化向游戏中添加本地化资产的过程的管道（例如，将文件放在相应的文件夹中，然后关闭它构建银行，设置文件替换和跟踪更改）应该如何，并使游戏部署到本地化 QA 在项目非常繁忙的时候相对快速且无痛。

与游戏开发的其他方面一样，音频需要技术，工艺和团队合作的神奇融合，如果我们要为玩家创造引人入胜的互动娱乐体验，所有这些都需要掌握。

第 10 节插曲 5：游戏音频：2D，3D，Mono 和立体声

单声道源的 3d 与 2d 还原

了解如何整合声音将让您深入了解如何掌握特定的声音组。此外，了解声音是否会在飞行中衰减可能会从根本上改变您的设计方式。将这些声音视为 2D 或 3D 可以帮助您准备资产，并在对它们进行评估后对其进行整合。

2D 音频是任何没有动态衰减或定位的单声道声音。它们通常通过中央声道进行路由，但在基本的立体声设置中，文件通过模拟中心效果的左右声道同等播放。例如：画外音和 UI 声音。这些声音在声谱中是相对较高的优先级，应该在混音之上和你面前听到。

3D 音频是任何动态衰减或定位的声音。 3D 声音位于左右光谱中，通常附加到游戏内对象

中。
几个例子：

静态物体，如瀑布或风车，您希望它们在接近它们时变得更响，在您移开时更安静，并且动态定位 – 这可以真正增加环境的再现游戏中的动画，如战斗攻击或来自敌人的表情（通常 3D 定位和衰减可以通过使用声音定位敌人的方向和距离来给予玩家战略优势）
动态生成的环境声音，例如鸟类，可以真正创建“非循环”的有机环境（此实现通常与其他参数配对，如动态音高变换，随机声音文件调用和播放频率）。

关于立体声音频源的注意事项：它们属于 2D 类别，因为它们没有定位或衰减。然而，立体声文件将被视为 3D，因为它们将包含将在中心的左侧和右侧听到的内容，通常具有更多动态。

单声道与立体声

为了确定波形应该是立体声还是单声道，您首先需要知道如何使用和/或集成声音。不同的音频引擎具有不同的功能集，应该告知允许的内容。在某些情况下，将不会有包含音频引擎，代码库中可能只有一些功能允许，例如，流音频或在事件或动画上触发声音的能力。在考虑性能的情况下进行立体声/单声道区分也很重要，其中单声道文件的大小是具有相同长度的立体声文件的一半。作为一般规则，如果源文件是单声道，则永远不应该成为立体声。以下是准备音频资产的一些一般准则。
单声道示例：
配音
用户界面声音和警报
3D 声音（在立体声或 360 度声场中动态播放和衰减的声音）
立体声示例：
音乐
环境
为视频准备的音频

第 10 节插曲 6：游戏环境中的音频准确性

游戏开发周期中不变的一件事就是改变。无论“最终”如何出现艺术或设计资产清单，总会有细化，调整和调整。通常情况下，这些变化会在最后一刻发生，并且肯定会影响音频设计的美感。为这种情况准备音频设计的一种方法是组织和实现具有灵活性的资产，作为音频管道的一部分。

实现这一目标的最简单方法是利用 FMOD 等中间件。通过多轨编辑，可以轻松地使用多个层创建声音组合，而不是提交到一个事件一个声音的实现。作为一个例子，如果你正在为爆炸的汽车撞击设计声音，那么声音可能包括金属嘎吱嘎吱，玻璃破碎，轮胎尖锐，汽车撞击地面以及车辆爆炸的瞬间。作为声音设计师的思考，这些声音可能会被创作成一个声音文件。然而，如果项目结束接近并且该车辆的艺术或设计影响发生变化并且汽车不再爆炸，则声音编辑器将不得不返回原始资产以重新设计声音。相反，如果声音的每一层都可以作为自己的文件使用，并且声音事件是通过中间件的功能构建的，那么很容易移除爆炸层并让其余的声音继续支持新的影响视觉效果。在我的最新项目中几乎每个游戏系统都出现了这些情况，从战斗到物理影响到 Foley。

这种方法需要来自游戏引擎的更多处理能力，因为它必须将几个资产相加以创建单个事件。如果开销可用，您可以使用这些碎片事件来运送项目。通过这样做，您为声音设计提供了更多的

音频多样性，因为每个层都可以为资产选择，音高和音量设置随机值。没有两个事件听起来会一样。但是，如果需要优化传递，您可以在设计完成时调整每个层，并且可以使用 FMOD 的内部记录工具轻松地将这些资产汇总在一起，以输出复杂事件的单个文件（注意：使用的项目） Wwise 或其他中间件可以轻松地将项目的输出重新录制到任何音频编辑器中以实现相同的目标）。这样的美妙之处在于，相加的声音文件将利用您在原始事件中创作的音高和音量变化，并且在重新实现到引擎中以替换分层事件时将准备好以适当的音量播放。刚刚使用这种方法完成了一个项目，阻止了许多声音设计的返工。这一点尤其重要，因为在一个部门，那个时间根本就不可用。

这种方式需要在灵活性和效率之间进行协商。项目的音频管道将有助于确定这种平衡。确定哪些声音组合在一起以及它们在项目结构中的位置将使音频设计人员能够在需要更改或优化时快速响应。继续上面的车祸示例。所有这些层都可以单独实施，但有些可以轻松分组以最大限度地提高效率。在初始撞击时发生的金属嘎吱嘎吱声和玻璃破碎可能是被归为一种声音的候选者。然而，如果使用 FMOD 或 Wwise，轮胎尖锐可能会继续作为单独的层保留，并且很可能在不同的项目或工作单元中加载。请记住，您的资产在中间件结构中分布的越多，如果需要优化传递，则将这些事件加在一起就越困难。

在游戏项目中，往往会有一组全局声音在整个标题中共享和重用。用户界面声音效果通常在此列表中，但声音类别如火灾，爆炸甚至物理影响也是如此。通过音高和音量变化，可以转换单个声音资源以支持各种视觉效果。使用的声音越多，为简化分层事件而应该求和的可能性越小。使用常用声音对事件求和可能会更有效，但是您会失去很大的灵活性，特别是如果该声音样本在游戏中存在多次。如果这些常见声音全局加载到您的标题中，则此功能尤其有效。

这种方法的关键是要意识到没有“正确”的方法来实现和管理分层管道。这是为您的项目找到合适的平衡点。

把它们绑在一起

设计，设计，设计的口号将有助于确保您创建的管道在扩展和收缩时满足项目的需求。如果你通过猜测来设计你的管道，你很可能会失败，所以不要简单地根据你把事情放在适当位置时发生的随机要求来构建你的管道。管道是一项投资，就像商业中的许多事情一样，您经常需要在长期和短期之间平衡您的必需品清单。

第 11.1 节 您本章中学到什么

在本章中，我们将概述本书的内容，必须考虑哪些要求以及要避免的陷阱。与往常一样，我们将包括电影和游戏之间的异同，试图突出每个人的独特要求。我们将以一个简短的教育资源列表结束本章。

第 11.2 节分析业务需求

最好在项目早期识别和记录不可移动的约束，即使是一个简单的事情，比如知道你是否必须支持 Linux 和 Windows，也会影响你的设计。花点时间考虑一下您的管道需要如何工作，让关键的利益相关者参与进来，讨论要求，实际将其写在纸面上。你需要多少时间来建造它？每个项目都有最后期限，而且常见的陷阱是在“管道”上花费太多时间而没有足够的时间来构建项目。如果你错过最后期限并且公司破产，拥有世界上最好的管道对你没有任何好处。

考虑一下您的设施的独特之处：是否存在使您的公司脱颖而出的特定文化或意识形态？您想在管道中推广什么方法？您的管道任务，资产或故事是否受到驱动？彻底检查管道的需求将影响数据的结构和流经管道，并作为某些决策的理由。

管道需要支持的生产的核心价值是什么？您需要能够确定功能的优先级并进行投资以支持您的生产中的独特挑战和需求 - 为专注于巨型宇宙飞船的生产创建复杂，创新的动作捕捉管道或创建自定义没有意义用于节目的流体渲染器，其中所有水射击在远处关闭，或者构建复杂的流体模拟器，当要发送的细节量需要数月计算时，该模拟器变得无用。相反，当他们支持项目的雄心时，你应该考虑雄心勃勃的计划：管道的最终工作是让艺术家找到项目的创造性核心，从而识别和应对最艰难的挑战。

构成公司文化的技能是推动管道风格的重要因素。你已经拥有了什么人，你还需要什么人？如果你没有你需要的人，可能需要几个月的时间才能找到合适的人，并且在他们加速之前需要更多的月份。

作为一般规则，项目管道需要维护的时间越长，构建它的时间就越长。贵公司是否计划在一年内过渡到全新技术，或者您是否正在制作需要维护十年的大型多人在线游戏？您可以在未来的项目中使用哪些管道元素？您的管道的某些元素可能是项目特定的，而其他元素可能会多年来一次又一次地使用。您是否为独特平台创建了一次性产品？如果管道的某个元素是项目特定的，一般规则是它应该节省您在该项目上的时间。花一周时间构建一个自动导入系统来保存您的艺术家不必手动导入数千个资产可能是值得的，但花一个星期构建工具来自动化一天繁琐的项目特定任务并不是明智的利用您的时间。但是，如果您的管道中的一个元素将用于未来的项目，那么可能需要花费更多时间来构建管道，然后它将在单个项目的过程中节省成本。

在电影中，多个项目经常与各个团队一起开展和逐渐减少，重要的是要了解工作室的长期愿景/目标是什么。是否需要支持多种项目类型：动画功能，视觉特效制作，TVC，数字安装等？预计的全球射击次数是多少？是否有任何潜在的收购或兼并？工作量是否会与其他公司共享，还是会完全隔离？工作室往往属于某些专业;工作室 A 可能非常适合角色和人群，工作室 B 非常适合 FX，工作室 C 可以处理大量音乐。这些示例中的每一个都有不同的要求，软件和硬件，以支持不同的工作流程，因此请务必了解应该强调的位置。

了解工作如何完成的总体方法同样重要，了解管道中的哪些阶段将以何种格式呈现给客户是至关重要的;你需要对完全合成的镜头进行动画审批，还是要将动画显示为简单的播放器？

您需要不断地向自己背诵：管道的目的是帮助艺术家和制作人员完成项目的视觉目标。因此，构建高效的管道通常是一种“外部”方法。首先，您必须了解您的客户是谁，然后您决定开始时需要什么。

第 11.3 节从工作流到映射组织的流程决策

了解在管道的任何阶段需要实现的目标不仅可以明确整体工作的内容，还可以帮助确定优先级，例如首先构建哪些组件，因为它们是必不可少的，以及哪些组件要求很好。要正确处理此问题，请首先考虑以下约束：

人力和机器资源的计划和资源计划是什么？

如何构建，源代码控制，配置和部署自定义软件？

用什么软件语言编写自定义软件？

什么系统将用于跟踪自定义软件（Track，RT，Bugzilla，Mantis）的错误以及与“内部”客户（其他部门）签订的支持合同将会到位？

什么是工作流程，您希望数据如何流经公司？ 例如，合成器是否能够发布新的相机版本，还是应该总是返回相机部门？ 照明开始的先决条件是什么？

了解数据流将有助于设计有效的工作流程。特别注意阻滞剂;你不想阻止人们工作，因为他们无法访问他们需要的数据。

你的评论是什么？在管道的每个阶段，您需要定义什么是可审阅的内容，并决定如何查看正在生成的资产，提供建设性的反馈并将其集成回系统以进行下一步;你不想制作一个封闭的盒子，你应该能够查看在任何阶段制作的每一件资产。

什么个人或团体将对管道负责？负责任的人员或团队应始终了解管道的工作原理以及管道的能力。他们应批准并对管道的任何变更负责，以确保它们与整体设计保持一致。

评估渲染农场软件，渲染器等中的选项。不要只是推动一个听起来最性感或带有漂亮花哨标志的选项。评估不同的产品，确定哪个适合您要解决的问题，哪些可以负担得起等等。使其成为客观的决策，您可以做出最好，最明智的决策，但最重要的是，做出决定。

在评估交互式工具时，请考虑它们在大型团队环境中的工作方式。考虑是否需要共享和合并工具编辑的文件，以及将工具插入管道的简易性。数据如何从上游源流入此工具，数据将如何从工具流向下游目标？即使该工具为该作业提供了最佳 UI，如果它所创建的数据无法在下游访问和使用，也没有用处。在考虑交互式工具（如销售团队在会议室环境中演示的级别编辑器，动画混合树创作或粒子系统编辑器）时，这一点尤为重要。

唯一一致的是变化，这是一种格言，从电影导演重建整个序列或大型游戏项目，取代目标平台到已经不再可用的既定管道不可或缺的商业工具。改变管道是不可避免的，当它发生时，重要的是要仔细评估并向所有相关方宣布这些变化。这可确保了解即将发生的变更，团队知道何时实施变更，以及这些变更将如何影响工作流程。但是，几乎不可能在一开始就计划所有方案。你不能总是预测会发生什么样的变化，但是你可以预测无论发生什么变化都会发生变化，并且为不可避免的变异做准备可以获得更好的成功率。在管道中建立这样的设施可能成本很高，但是当需要进行更改并且管道太硬而无法容纳时，它会更加昂贵。

第 11.4 节技术和基础设施决策

同样重要的是要了解构建管道所需的基础知识，以及决定购买什么和构建什么。您需要考虑基础，这些核心技术将构成您管道的基础。至少，这将包括资产管理系统，生产跟踪，命名约定/目录结构和文件交换格式。确定这些是什么，并确保其他一切围绕着它们。

在定义标准时，必须在解决以下问题时执行此操作：

将使用哪些操作系统？什么软件将管理渲染农场以及将在其上运行哪些进程（Qube，Alfred，Tractor，诸如 SunGridEngine 等开源选项）？

资产管理系统有哪些要求？ 你会使用 Shotgun / Tank，Tactic 或 Build a Custom 解决方案吗？ 您是要在 Excel 等程序中跟踪小时数和烧毁率，还是计划整合成熟的产品跟踪解决方案？ 资产如何组织？ 你有没有想过访问时间？ 工作文件如何与构建文件分开？ 有你减少不必要的裁员？

什么网络基础设施正在建立 - 基于星形，子网？ 什么硬件可用于交换机等？

您需要哪些中间件解决方案？ 您是否已完成尽职调查以了解使用特定中间件所涉及的所有潜在缺陷和风险？ 你知道整合需要多长时间吗？

你的团队有多大？ 有多少人需要访问特定文件？ 您是否设计了管道以避免共享和不可合并的文件？

如何对用户帐户进行身份验证以及如何实施和管理帐户权限？

您将如何在内部和与客户一起审查材料？

应该为哪些存储供应商，数量和存档选项提供服务？ 是否要求归档所有生产数据以及所需的时间段？

你的英雄应用是什么？ 对于任何 CG 工作：Maya 或 Houdini？ 渲染：PRMan 还是 Arnold？ 用于管理渲染农场：拖拉机还是拉什？ 什么软件将用于每个学科？ 您需要购买/运行多个？

像 Maya，SoftImage，3DStudioMax 和 Houdini 这样的 3D 软件包具有经常重叠的功能，如果工作室应该购买一个，两个或者是否应该使用全部问题，那就提出了一个问题。

什么硬件供应商将用于 renderfarm 和工作站？您将使用什么渲染器以及如何使用它？虽然管道应该能够支持多个渲染器，但是集成特定渲染器需要花费大量时间。每当与渲染器交互时，您应该尝试通过一个抽象层，但更重要的是花时间确保您能够利用渲染器中的高级功能。没有打火机会放弃一个干净的工作流程来围绕渐进式重新渲染，以便可以支持他们的项目不使用的渲染器。

另一个主要问题是如何在您的管道中构建可扩展性 - 至少您必须承认其存在并确定明显需要扩展的区域，将很容易扩展，并且需要进行大规模的投资。如果你把它作为一个额外的额外螺栓固定，你将永远无法获得管道所能提供的全部好处。

在做出这些决定时，请确保避免因重新发明轮子而导致的陷阱;通过一些研究，你可能会发现其他人之前已经做过，而且他们做得比你好。您可以使用什么技术开源（Alembic，OpenEXR 等），您需要购买什么？如果您需要进行更改，是否可以回馈这些项目？并考虑什么打破了规则。规则的例外情况是可以的，但前提是有充分的理由。如果您做出的决定不符合管道的其他部分，请了解您做出决定的原因，确定后果，并记录决策的信息和原因。

但等等，还有更多！一个好的管道需要提供调试工具，帮助您查找和诊断可能遇到的平台特定问题。您需要为所有工具提供通用基础结构，以管理调试和诊断信息。这可以从非常简单的，例如日志文件的标准化位置，到更复杂的系统，其中错误被记录到数据库，并且在出现问题时将问题通知发送到管道团队。如果您的工作室已经使用问题跟踪系统，最简单的方法是利用它，但是您决定继续，鼓励所有管道开发人员使用相同的标准和工具来处理错误报告。调试管道问题已经足够困难了，如果管道的两个部分没有以同样的方式报告问题，那就更难了。

您需要一种策略来向用户分发工具，确保用户保持最新，并且可以轻松地将错误修复分发给团队，同时平衡即时性和安全性，确定一种方法，确保管道工具的演变不会扰乱生产。
您的工具开发将向用户展示多少？
他们会在准备好后立即看到您创建的所有内容，还是通过正式的质量检查流程发送工具？
您是否会在完成工具或按预先计划的时间表发布工具？

一旦你计算出格式，你还需要考虑分发工具的物理方法。一个简单的系统，例如将最新版本的工具保存在共享网络驱动器上很容易实现，但需要用户方面的纪律。使用版本控制系统可以利用您已有的版本管理工具，只要您的用户已经习惯使用它们并使自己保持最新状态。但是，版本控制系统不会自动运行任何更新过程，例如文件格式转换或旧数据的重构，它只是确保用户在其磁盘上具有相同的位。

另一种选择是基于 Web 的工具包，如 Microsoft 的 ClickOnce 或 Java Web Start，它允许工具“回家”到 Web 服务器并根据需要自动更新。但是，它们还需要服务器和维护它所需的 IT 资源。在浏览器中运行的仅限 Web 的工具总是按照定义更新，并且可以与世界各地的团队共享而无需额外的努力，但请记住，基于 Web 的工具本身无法与用户交互'本地文件系统，最适用于处理数据库或向远程服务器提交文件等内容。

第 11.5 节电影和游戏的独特考虑

虽然电影和游戏管道的许多管道要求重叠，但每个管道要求都有不同的要求。例如，有多个工作室（电影中的常见情况和游戏中不断增长的工作室），在设施之间传递数据时，您需要考虑带宽要求，如果您需要专用网络链接，如何共享工作，以及意味着什么你需要提供的沟通。

在电影或游戏过场动画中，你必须考虑渲染的方法，它是基于物理的，光线追踪还是扫描线。例如，决定使用像 Arnold 这样的光线跟踪器可能会减少对存储的需求，但对渲染能力的需求将会增加。另一方面，使用皮克斯的 Renderman 增加了对更多存储的要求，但可能需要更少的渲染时间。

在游戏中，要始终牢记您正在制作什么类型的游戏以及您正在构建的平台。用于 PC 的构建具有独特的考虑因素，为多个控制台构建也是如此，并且每个平台都需要特定的测试步骤。您正在构建的游戏平台越多，您的管道就越复杂，每个平台都需要不同的导出流程。不同的平台可以具有不同的性能，并且通常需要为每个平台创建不同的资产。例如，移植到 Xbox 上的 PC 上的游戏可能需要为角色和环境创建的较低分辨率纹理。根据平台，良好的管道可以通过允许资产引用多个纹理来简化此过程。您需要设置管道以适应这些变化。

必须在生产之前构建和测试为各种游戏系统提供数据的每个管道，并且游戏需要许多不同类型的数据。艺术数据源虽然是字节数据的最大数量，但在很多类型中只有少数类型，例如物理模拟世界，AI 数据或游戏逻辑。这些数据源中的许多是相互依赖的，并且艺术数据和游戏玩法数据如此紧密耦合以至于艺术家创作者（也将在游戏团队或编码人员的指导下）创作另一个，例如物理网格，类似于可见网格，但不太详细。对于从电影业转向游戏业的建模师和动画师来说，这通常是一种文化冲击。

您的管道设置是否支持分支？在游戏开发中，通常需要分支。如果您的产品需要分支，则源控制解决方案支持该要求非常重要。

如果您在线分发项目，则必须确保拥有所有必要的在线基础架构来分发您的产品。您可能需要优化下载时间，如果这样做，则必须确定管道如何支持这些未来的优化。

第 11.6 节 建造和验证管道

在完全生产的第一天之前完成管道的竞赛经常因原型制作期间的意外发现或方向变化而变得复杂。计划的功能被削减，新的功能被添加，而生产的第一天不可避免地接近。唉，远远没有闻所未闻的生产开始，而管道的关键区域仍然在绘图板上。与负责生产计划的人员进行认真协调可以最大限度地减少对项目的不便，但是使用半成品工具集开始全面生产仍然是一个重大风险。

将风险降至最低的一种方法是设计适合进化的管道。就像网站或在线服务一样，管道实际上是一种持续的服务，而不是一套简单的软件工具 - 它永远不会“完成”。相反，管道和构成它的各个工具应该使用模块化部件构建，可以随着项目的成熟而扩展和细化。基线功能需要在第一天准备好，以便艺术人员可以开始工作;这意味着，诸如如何通过管道移动数据或如何跟踪资产等关键决策是尽早做出的。然而，对用户界面的改进和艺术家的生产力增强工具可以逐渐上线 - 这不仅减少了过于无法满足生产期限的竞赛危险，还意味着面向用户的决策可以通过来自更大，更有信息的用户群。

幸运的是，对于管道开发人员而言，敏捷编程运动提供了一套优秀的实践，可以在这种面向服务的流动环境中工作。敏捷的理念强调快速交付，持续改进，以及对大规模集中规划的紧急问题的快速响应。敏捷鼓励编码人员与他们的“客户”密切合作，明确定义需求，然后迭代地满足这些需求。敏捷团队通常在“sprint”中工作 - 短时间，紧密集中的工作（通常短至一周或两周），每个工作只提供一个功能或系统。然后根据当前的优先级计划下一个冲刺，帮助团队专注于实际上当前对项目重要的问题，而不是在 18 个月前的计划会议中看似重要的问题。由于管道工作完全是为了处理紧急的生产混乱，因此很容易理解为什么敏捷方法强调适应变化和根据情况需要改变方向的意愿，这已成为管道团队看待其工作的方式的主导。

敏捷方法的一个重要补充是大量使用自动化软件测试，通常称为“测试驱动开发”或 TDD。在 TDD 中，编码人员创建简单的测试程序，以确保他们的代码行为不会以意想不到的方式发生变

化。例如，假设一个团队使用一个软件为其管道生成可预测的结构化文件名。对该位代码的测试将提供一组已知参数，并保证产生预期结果。这为不断发展的管道的持续流失增加了非常宝贵的安全性和稳定性。

第 11.7 节开发方法

构建管道的主要目标是不断确保工作从头到尾安全有效地传递给所有艺术家，但是如何实现这一点是首要考虑因素。

定义用户界面指南非常重要，包括 UI 元素的样式指南，命令行工具的行为方式，编码标准和错误处理。为了鼓励直观的工作体验，避免不一致的 UI 和应用程序行为，而是寻求重用 UI 库来帮助保持外观，感觉和用户体验的一致性。构建模块化结构并提供可重用自定义软件的 API 是实现此目的的一种方法。存储管理（例如，创建，删除，移动和重命名目录和文件）就是一个例子。另一个是数据库连接，图像序列（重命名和重新编号），访问资产管理和生产计划。一定要包含详细的文档，因为经常处于压力之下的人会倾向于编写自己的代码，而不是花时间去理解别人的代码。

确保 API 不与特定软件供应商或开源项目绑定，因为无法保证将来可以使用，支持和维护该软件。这同样适用于硬件;供应商通常会发布用于打开特定于自己硬件的功能的自定义 API。如果这些功能成为整个系统的重要组成部分，那么以后转移到其他供应商可能会非常耗时。

在构建任何管道时，重要的是要牢记安全性方面。身份验证（确认此人是他们声称的人，通常通过用户名和密码实现的步骤）和授权（用户是否具有执行此操作的权限）都是必不可少的。VFX 行业需要进行非常强大的安全审计，控制人们进入大楼的方式，可以访问的网站，员工登录机器的方式，允许将数据上传到互联网的限制以及安装外部存储设备的能力，刻录 DVD 或 CD，或检查网络是否已分区。从 MPAA 和客户工作室开始，这些审计全年多次发生并不罕见。强烈建议在构建管道时确认安全要求，因为必须在以后对其进行改造可能会非常昂贵。

一个问题是如何建立问责制。人们有责任维护管道，这一点很重要。通常，确保管道在产品过程中不会降级的最佳方法是分配负责监控和维护管道的唯一所有者。这样，如果有人违反命名约定，创建重复的文件夹结构，或发明自己的编码标准，将设置负责实体以提醒违规者他们的错误并确保问题得到解决。

没有记录，就没有记录发生了什么和出了什么问题。记录应该是一个基本服务，并且应该以某种形式存在于管道中的每个工具中。日志并不完美，如果设计不当，它们确实存在问题。没有任何日志几乎与无法从大量数据中轻松提取信息的日志一样糟糕。

记录数据时，请考虑有人需要搜索以提取有关故障情况的信息，这是在压力情况下经常需要的任务。提供一组一致的可搜索关键字和句子结构，与正则表达式配合使用可以节省很多分钟，如果不是几小时的时间来确定问题的原因。在某些情况下，更好的方法是完全抛弃线性日志并将结构化日志记录信息存储在数据库中，从而为用户提供了向下钻取所需信息的机制。花些时间考虑可能出现的问题并制定一项规定，以减少您在其他人办公桌上解决问题的时间。使诊断工具更好，用户将解决自己的问题，让您有时间编写更多工具。

如果每个人都能明确定义流程，你可能会说管道几乎没有必要。如果所有内容都已布局，记录，理解，并且人们不会出现命名错误，理论上应该可以在不需要任何提供版本控制和依赖性跟踪的软件的情况下运行生产。但严重的是，谁相信会发生什么？！

第 11.8 节继续教育

有多种选择/资源可以了解更多关于整个 VFX 或特别是管道的信息，从在线博客到用户组，从邮件列表到教育课程。虽然仍然太少，但是有越来越多的额外教育资源可供使用。列出它们是不可能的，但这里有一些可以帮助您入门：

学分和非学分课程

以实体大学课程和在线学费形式的进一步教育为学习 VFX 和游戏制作艺术提供了更正式和更有条理的方法，通常与许多行业合作伙伴合作。

在线资源

从新版本到新闻到论坛，有一些看似无穷无尽的在线信息可用于涵盖行业内容。 有些是每天更新，有些则归档了很长时间仍然有价值的文件。 许多 VFX 专业人士已经推出了自己的网站，涵盖了从创建软件应用程序插件到操作指南等各个方面。 与在互联网上找到的所有信息一样，您必须注意信息的可靠性，但互联网可以成为社区网站的重要来源。 需要第二本书才能列出所有可用的资源，这本书永远不会是最新的或包罗万象，但下面是一些让你开始的更受欢迎的网站。

fxguide: www.fxguide.com/ . A very popular news site that is updated daily.

Slashdot: http://slashdot.org/ . Another popular news site “powered by your submissions.”

Gizmodo: http://gizmodo.com/ . A news site that covers a broad range of topics.

Game Studies: http://gamestudies.org/1102 . Focusing on games, a collection of thought-provoking articles discussing “ideas and theories.”

Center for Computer Game Research: http://game.itu.dk/index.php/About . A collection of articles and papers from some of the best sources, including IEEE and AMC SIGGRAPH.

Digital-Tutors: www.digitaltutors.com/11/index.php . A membership site offering a wide range of courses from basic modeling to scripting.

fxphd: www.fxphd.com/fxphd/courseInfo.php . This site offers a range of courses with a special section just for pipelines (see checkbox menu near the top of the page).

CGTalk Development and Hardware forums: http://forums.cgsociety.org/forumdisplay.php?f=108 . A subsection of CGTalk focusing on development and hardware discussions, news, and opinions.

Studio Sys Admins: www.studiosysadmins.com/ . An active forum of developers, engineers, system administrators, hardware/software vendors, and CTOs etc. who discuss problems related to the more technical side of building a studio.

Scriptspot: www.scriptspot.com/

Tech-Artists.Org : http://tech-artists.org/

The Toolsmiths: http://thetoolsmiths.org/ . The blog of the IGDA special interest group on tools. It has been a forum for discussing pipeline programming since 2009.

会议和组织

全年举办了多次会议，重点是数字媒体制作，尽管管道相关内容的水平每年都有所不同。这些会议中的大多数都在网上公布他们的会议记录，可以参考，有时可以收取费用。这可能包括谈话视频，技术论文或其他网站的链接。如果您不能亲自参加会议，使用这些程序可以获得大量通常符合研究/大学标准的信息。

FMX: http://www.fmx.de/ . One of the primary European conferences covering animation, effects, games, and transmedia.

AMC SIGGRAPH: www.siggraph.org/ . Two conferences a year, one in USA/Canada and one in Asia. The biggest conference within the VFX industry covering a wide variety of topics.

NAB: www.nabshow.com/ . Held in the USA looking at content creation, delivery, and management.

VIEW Conference: www.viewconference.it/ . An Italian-based conference on computer graphics, interactive techniques, VFX, and computer games. While it focuses mainly on the graphic side of production it does offer the occasional course on topics related to the pipeline.

GDC Conference: www.gdconf.com/ . The Game Developers Conference, held every spring in the US, is the global gathering of the games industry. In addition to a full program of case studies and lecture courses on all aspects of game development, the show hosts several events of special interest to pipeline developers. The Tech Artists Bootcamp is an all-day series of lectures for technical artists where many key aspects of the pipeline, including both general tools programming practice and special problems in graphics are covered. The Technical Art, Technical Animation, and Tools Programming roundtables are open forums for specialists to discuss technical problems and share experience.

E3 Conference: www.e3expo.com/ . For the latest in games and electronic entertainment held in the USA.

IBC Conference: www.ibc.org/ . European conference similar in nature to NAB.

Createasphere Digital Asset Management conference: http://createasphere.com/ . Annual conference in New York City.

DAM NY: http://henrystewartconferences.com/dam/damny2012/ . Conference covering the “art and practice of managing digital media”.

Eurographics: www.eg.org/ . A European professional computer graphics association.

IEEE Computer Society: www.computer.org/portal/web/guest/home .

The IGDA (International Game Developers Association): www.igda.org . The largest professional forum for game developers. Local IGDA chapters around the world host regular meetings, usually in association with local development studios or educational institutions.

用户组和邮件列表

几乎所有的开源软件项目都有某种形式的社区与他们相关联，通常以邮件列表或组的形式。如果您使用或甚至只是对某个特定的开源软件感兴趣，您应该考虑加入讨论并在社区中变得活跃。通过订阅邮件列表，您将接触到其他用户和软件的使用，这可能为您的工作方式创造新的机会，确保项目继续蓬勃发展，并能够更适当地提供反馈和想法未来的发展。特别是，对于那些专门为 VFX 社区维护的项目，您将深入了解工作室的一些内部工作方式，这些工作在日常讨论中变得明显。您还可以更进一步，成为项目的贡献者。

Global VFX Pipeline Google group: https://groups.google.com/forum/#!forum/global-vfx-pipelines . One of the more popular pipeline groups that is frequented by many of this book's own contributors. Python Programming for Autodesk Maya https://groups.google.com/forum/?fromgroups#!forum/python_inside_maya . The group tracks the development of Maya's Python integration and is of particular interest to pipeline developers.

偶尔这些团体会面，或者作为 SIGGRAPH 这样的大型会议的一部分，或者在一个主要的城市 VFX 中心进行更多的本地聚会。加入这些团体是一种很好的方式，可以接触到更大的社区，让自己与其他专家心灵联系，每天解决同样的问题。

介入：电影和游戏中的虚拟生产

第 11 节插曲 1：电影中的虚拟制作是什么？

虚拟制作是在数字世界中实时制作电影，视频游戏过场动画，电视节目，广告，音乐视频等的过程。虚拟制作在传统电影制作世界中起作用，带来新技术和工作流程，为电影制作人，商业导演和游戏开发者提供创作自由和效率。虚拟生产跨越了广泛的学科领域，这些学科经常深入融合到预可视化，性能动作捕捉，实时动作（与 CG 元素的集合集成），后期可视化等等。因此，虚拟生产在处理资产和相关任务时面临着一系列挑战。

值得注意的是虚拟生产“是”生产。我们正在制作电影，电视节目，视频游戏这个过程中的电影。这构成了资产，资产管理和任务的主题。

虚拟生产中的一个有趣挑战是，它是一个在设置和生产中的每个人的实时交互过程。在虚拟世界中拍摄（或者部分地，当涉及到实时动作集成时）通常是实时完成的，这允许创作故事讲述者，电影制作者和制作背后的整个技术团队对设置为的物理世界进行快速迭代更改。撰写镜头或定义角色的表现。这种实时方法允许在虚拟制作中使用传统的电影制作技术，术语和学科。团队中的每个人都必须对请求和资产（物理和数字）的变化做出快速反应，更重要的是，跟踪这些变化和资产修改，以便在虚拟生产停止期间和之后将其下游传递到其他部门。没有人等待这些变化，因此我们需要快速使用，适应性强且允许此类更改的工具和工作流程。这些变化可以包括物理演员的表演被转发到他们的数字角色，以及改变的表演，环境，集合，道具，装置，车辆，装饰，照明（数字），氛围等的数字和物理变化。

在整个生产过程中的任何时候，我们都会尝试以一种允许我们以“合法”和定义的方式迁移请求和数字以及下游不太常见的物理变化的方式对创意和技术请求作出反应，以便他们可以使用正确。考虑到快速的时间框架，这通常很困难，因为我们需要及时对导演作出反应。当发生这种情况时，重要的是记下超出法定范围的任何内容，然后在将其交给下游流程和部门之前将其合法化。这都是生产的一部分。

第 11 节插曲 2：命名约定

最早的考虑因素之一就是如何命名，我们所做的很多事情归结为正确命名资产，并使这些资产名称变得人性化和可读取。在命名资产方面投入了大量精力，特别是在虚拟生产期间捕获的阶段性需求。这是为了便于跟踪生产阶段中的所有元素，因为它们最终将包括期望的场景，更重要的是，序列内的镜头。舞台拍摄的命名很重要，因为它通过编辑，摄像机工作以及通过导演剪

辑和最终的 VFX 镜头制作进行跟踪。命名这个阶段是最重要的事情之一。

虚拟制作通常不是在镜头世界中开始，而是在场景世界中开始。一个场景包括几个镜头，可能经常包括两件事。一个可以是几个“设置”，定义为物理和/或数字集的唯一变化或配置。它还可以包括由于场景不适合物理捕获体积而导致场景内的性能改变,或者当场景由于性能原因需要被分解并且平铺在一起时。第二个要求可能是“通过”，其中不同的演员或相同的演员在同一场景中扮演不同的角色。当我们通过制作进行迁移时，我们正在为这个命名约定添加元素，例如主场景，相机镜头和版本。我们需要能够跟踪一个镜头回到一个序列，一个场景，一个舞台片段，然后拉出集合，道具，角色，演员，参考材料笔记等。很重要，因为它在多个地方被注意到;在平板，刻录，视频辅助，声音，动作捕捉，脚本监督，数字脚本监督，编辑，视频见证摄像机，艺术部门等。

第 11 节插曲 3：标准阶段

虚拟生产涵盖了广泛的工作流程，具体取决于生产，但为了简单起见，我们将其分解为前期制作，生产，后期制作和最终交付。在这些阶段中，资产管理和关联任务是一个显而易见的必要性，也是不容低估的。通常，我们必须适应生产可能选择使用的不同工作流程和界面。通常，资产数据库和任务管理是通过独特但高度集成的软件和界面实现的。

前期制作

预生产通常涉及接收和/或开发关键资产。这些资产通常来自第三方，对于现场世界而言可能是也可能不一致或合法。通常，需要创建数字环境和道具，其可以在场景世界和镜头世界中的虚拟生产中使用。构建这些元素的元素通常来自预览或故事板组，艺术部门以及主要摄影（激光雷达，扫描，摄影测量，照片参考等）的现场调查。如果我们真的很幸运，我们可以利用 previs 团队的数字资产。通常需要降低复杂性和多边形分辨率（以及纹理），以便它们可以实时运行。它们还需要以允许实时设置容易的资产变化的方式构建（即，移动椅子，重新定位树，移除墙壁，打开门等）。在我们这样做时，所有这些资产都会分配一个资产代码并进行标记。同时，我们在欺骗灯光和大气效果，因此我们可以给出场景的感觉并实时拍摄。 （值得注意的是，最近的作品在规划和指定这些数字资产方面取得了巨大成功，因此它们可以从预览团队中建立，使用和传递，并直接用于虚拟生产。节省成本和规模经济是有价值的当这件事发生时。）

数字动作捕捉木偶（字符）被构造为用于虚拟制作的规格。由于创造性过程，这些通常会在最后一刻发生变化，虚拟生产必须在整个生产过程中做出反应并接受这些变化。为角色分配资产代码和版本;这些信息很重要，因为还必须考虑基于演员角色的资产生成。角色通常基于扮演它们的演员的外观，动作和身体。虚拟制作涉及演员的扫描，面部测量，测量和参考摄影。根据这些信息，可以创建物理资产（头部，面部，手部，牙齿等的 3D 打印）以及数字版本。从这里开始，角色被概念化，并为虚拟制作创建动作捕捉木偶。

值得注意的是，演员要求驾驶角色脸部所需的任何面部捕捉通常需要在制作前对演员的脸部进行详细调查。这通常通过一系列摄像机拍摄演员的定向和技术性能来实现，其中他们通过许多面部表情（姿势）和将在下游用于产生角色动画的表演来拍摄。

除此之外，道具，场景装饰，灯光设置和大气资产都以类似的方式创建和管理，直至拍摄。这些类型的关键项目也分配给场景和序列，然后交给虚拟艺术部门，艺术部门，最终建筑和道具制造商建立数字资产的物理版本。这些需要关联（资产名称和版本）并在整个生产过程中进行跟踪。

在这个阶段，引入“全球规模”一词很重要。演员通常可以扮演比自己更大或更小的角色。在这些情况下，物理世界必须适当缩放并跟踪包含此性能的阶段。

其他“平板”资产板组件，摄像机轨道，故事板等也被收集并使其成为虚拟生产的合法，并通过管道进行跟踪。通常这些都是在镜头世界中，需要放置并关联到场景世界中。

准备工作还包括将资产名称分配给演员，特技演员，动物以及作为演出的一部分而出现的任何人/任何事物。这些将在生产过程中进行跟踪，因为大多数物理元素都具有数字对应物。元数据始终使用。一切都从物理阶段追溯到数字阶段，直到现在臭名昭着的阶段。

通常被视为资产的其他物品包括将在集合中使用的视频见证摄像机，以及演员将用于性能捕获的面部安装的摄像机和头部装备。所有这些都是管道的一部分，并且已分配资产名称并将其输入数据库，并且所有这些都与阶段关联相关联。

在预生产期间，制作团队（在虚拟制作团队的帮助下）必须分解可能存在于序列，场景，然后设置中的预览，故事板和编辑参考，以及可以在集合上捕获的传递。这是资产管理树层次结构中的顶级节点。

所有这些对于生产的成功至关重要，特别是因为我们必须快速反应，制定（和跟踪）变化，并将时间作为我们的第四个维度。灵活性是这里的关键词，支持在前期制作中生成上述内容的所有工具和脚本以及生产期间所需的更改（如下）必须是模块化的，使用它们的艺术家都能很好地理解，并且具有灵活性。这允许快速更改，并且如果需要，对于集合中的艺术家可以违反规则以响应导演的请求。虚拟制作中的艺术家必须了解每个工具和脚本的作用以及围绕它们的工作流程，以便他们能够以有组织的方式快速适应变化。

生产

在生产过程中设置，如果根据上述过程做好一切准备，虚拟生产团队应该能够快速响应导演，DP，编辑，视觉特效监督员和虚拟制作主管（以及演员）的要求和特技协调员）。在拍摄期间，我们不仅使用和创建数字和物理资产，我们还在生成新资产和大量资产。

鉴于预算和后勤约束，见证摄像机用于获得尽可能多的视频集合。这些操作的摄像机条纹与所有其他设备具有相同的阶段时间码，并且它们记录在存储卡上用于全高清媒体，以及将实时视频馈送传递到视频辅助以便在设置上进行回放。每台摄像机的视频高清素材每天下载，并提供给编辑和虚拟制作。

TSound 是现场录制的音频，它使用相同的时间码进行条带化，并且舞台名称与所有其他设备一样。动作捕捉团队正在记录所有演员和道具的身体动作以及面部安装的摄像机记录演员的面部表演 – 所有这些都是用相同的时间代码和舞台名称条纹。实时 3D 可视化渲染场景的虚拟版本，使用时间码进行条带化并提供给视频辅助以进行记录和回放。数字脚本管理员正在记录与数字和虚拟生产世界相对应的所有注释，并且脚本管理员在场并执行他们的任务。

如果虚拟制作团队已完成预生产工作，则所有这些媒体都将使用正确的时间码，元数据和命名约定生成，并输入资产和任务数据库，供下游所有部门使用。

在拍摄期间，注意到了（理想情况下）有利的表演，并为虚拟后期制作的下一阶段做好了准备。

在这个阶段，资产正在发生变化。在拍摄过程中，简单的位置变化（如移动椅子）很容易记录和记录。通常会在飞行中创建新的道具。如果时间允许，这些都必须快速生成，在管道中合法创建（命名约定，索具等）。通常他们不可能，所以他们快速操纵笔记使拍摄合法化。由于性能方面的考虑，集合可能会发生变化，并且使用相同的过必须考虑和跟踪物理变化和数字变化。

通常在场景中捕捉场景并给出所描述的术语。最重要的是，通过使用虚拟相机，可以拍摄场景中的覆盖范围，并且还可以捕捉拍摄世界中的特定时刻。这是舞台采用严格的命名约定和资

产标记非常重要的地方。

后期制作

编辑的任务是从制作中摄取所有平面媒体，包括所有见证摄像机镜头，虚拟制作团队的实时渲染，声音以及视频辅助播放。

编辑然后将一个表演大会一起切割。这是给定序列的最佳性能，它以多个资产的形式提供给虚拟生产，QuickTime 从舞台媒体渲染详细说明性能和某种形式的编辑决策列表（EDL）详细介绍阶段名称，舞台时间代码输入和输出，目标场景或镜头时间代码，以及所需的字符。

来自编辑的这种性能装配周转成为虚拟生产的任务订单，以产生所选性能的抛光实时版本。通常在拍摄开始时对该过程进行设置，并且一旦完成性能拍摄拍摄也执行该过程。

然后，虚拟制作将 EDL 与生产拍摄期间捕获的所有 3D 资产相关联，并创建所需组件的实时 3D 性能，然后可将其转换为镜头以支持导演剪辑。我们整理拍摄的所有 3D 和 2D 资产，生成符合要求的装配的新的最终资产。这个过程的大部分是通过解析可读 EDL 文件的脚本自动完成的，并且可以潜入相应的数据库。

来自阶段拍摄请求的动作捕捉数据被清理并且最终用于角色和道具的身体动画。这样可以确保角色在最终的 VFX 镜头制作中出现在相机中，就像在虚拟制作中的镜头制作过程中一样。该过程可以包括拼接来自不同演员的多个演出并且在组件内采用相同的角色。

拍摄来自舞台的面部捕捉视频数据，并创建投影的“歌舞伎”，可以将其显示（投影）到数字角色的脸上，供创意团队观看面部表演。

为装配创建了数字环境，它结合了集合上发生的所有变化，以及照明和氛围。如果制作是使用 CG 元素的实时动作，则跟踪板，使用 temp roto 和 comp 工作以将 CG 元素（通常是字符和设置扩展）集成到板中。

从所有这些工作中创建的最终是一个与编辑性能组件相匹配的主场景组件。这通常是包含所有资产的 Maya 或 MotionBuilder 文件，通常会被引用。这可以是场景世界，序列世界，甚至是拍摄世界。主场景最初由编辑命名，并从为场景生成的主舞台拍摄中获得。尽可能在所有数字资产上传递元数据。通常需要在角色的表现之上进行动画调整和添加（从演员的表现中得出）。这些在 Maya / MotionBuilder 中以层的形式实现并进行跟踪，以及返回到 VFX 镜头制作的周转记录。

创建主场景组件后，通过编辑同步检查和批准，然后虚拟制作可以进入镜头世界（除非由于制作的性质我们已经在镜头世界中）。

可以通过多种方式从主场景装配中创建镜头，但最终会为导演，DP 和编辑器创建相机和镜头合成。在此过程中，从主场景中创建镜头，并相应地调整数字资源以用于镜头构图。为了构图目的，移动数字道具，角色和布景，并且设置粗略的照明 - 所有这些都是尽可能实时完成的。

一旦镜头完成（通常在几次修改之后），渲染就会产生并交回编辑，以支持导演剪辑。生成 3D 主镜头文件，其中包含镜头长度的所有资源和动画。这通常直接转换为布局文件，该文件作为 VFX 周转的一部分提供，用于最终的 VFX 镜头制作。

一旦序列被批准用于最终周转，因此 VFX 工作可以开始，编辑将镜头序列转换为虚拟制作以及 VFX 镜头制作。虚拟制作现在的任务是根据编辑中的镜头周转率和用于创建镜头渲染的 3D 主镜头文件，从制作拍摄中创建最终资产。

编辑的营业额与绩效大会的类似。现在在镜头世界中，编辑提供了镜头的 QuickTime 和图像序列，以及计数表和描述阶段时间码空间中的性能编辑并且用于镜头帧空间的 EDL。每个演员的身体和面部表现的参考 QuickTimes（来自见证摄像机）也被翻过来。

虚拟生产与参考 QuickTimes 一起摄取 EDL（理想情况是通过工具和脚本）。新资产来自生

产中获得的阶段。如上所述生成在性能组装过程期间未排序的背景字符和道具。然后生成最终面部递送。这是从面部安装的相机生成的 3D 资产，并且基本上是面部相机所看到的演员的 3D 表面和眼睛轨迹。然后将该动画表面与在预制作期间捕获的任何面部调查资产一起使用，以在动画接管之前将演员的表演“解决”到角色上以微调表演。

然后修剪所有最终 3D 资产并在镜头空间中传送以匹配计数表。目标资产通常已从预生产变为现在。出于创造性原因，角色，环境和道具设计可能已更改，因此所有现有 3D 数据都将重新定位或修改，以便在这些新资产版本上使用。然后将这些 3D 资源与 3D 主镜头文件一起翻转，以开始 VFX 镜头制作。

作为虚拟生产一部分的大量数据和流程可能看起来复杂，复杂，令人恐惧，而且势不可挡，特别是如果此描述是对该主题的介绍。但是，经验丰富的虚拟制作团队能够引导电影制作人完成整个过程，这实际上是非常合乎逻辑的，因此他们可以体验到众多优势并避免复杂性。当组织和执行良好时，上述过程是经济且非常有效的。它允许电影制作人和故事讲述者在最终的 VFX 制作开始之前非常快速和迭代地找到剪辑。

如果您在虚拟制作中迷失方向并感到困惑，那么您可以为我们的团队做些什么。创建一个名为“praw”的任务类别。这是在过程中丢失的东西，需要弄清楚。关于大型制作的每周大型会议至关重要。见面讨论陷入困境的东西，让它从卡住中退出。为什么“呐喊”？它的“扭曲”向后拼写并反映了虚拟制作的复杂性 – 它总是涉及四个维度，通常是五分之一。

第 11 节插曲 4：什么是游戏中的虚拟制作？

虽然我们并非直接参与游戏开发，但我们通常通过虚拟制作电影序列以及通过游戏运动混合树和运动过渡的动画创建间接参与游戏开发。

第 11 节插曲 5：虚拟生产和资产创建/捕获

从动作混合树和运动过渡的演员生成运动以在游戏引擎内的运行时支持角色移动与用于故事片的人群模拟高度相关。

对于游戏的人群模拟和混合/过渡树，我们被赋予生成具有标记的元数据的特定动画的任务，以帮助描述从动作捕捉阶段的演员指导和捕捉的表演中的动作。

通常，大量的思考，计划和努力是由游戏/人群团队推动的，因为他们自然拥有游戏背后的创造性眼睛和思维以及 AI 系统和玩家控制系统的开发。 。这些系统确定所有游戏角色所需的各个动作单元。这些操作单元基本上是给定字符的单个动画，它是支持所需的混合树和过渡树所必需的。

对于故事片和动作混合/过渡树开发的人群模拟，通常有几个动作捕捉镜头，因为过程通常很复杂，需要迭代来收集和完善整个开发周期中项目的所有所需动作单元（动画） 。

前期制作

一旦定义了主要操作单元并定义了动画列表，虚拟生产团队通常会参与其中。这通常以表格格式（Excel 或数据库）翻转。通常这个列表很大，有数百个，通常是数千个动作，并列出每个动作单元：

名称和类别（战斗，运动等）

类型（地形，机载，车辆驾驶员等）
道具关联（这会影响道具）
通常是身体面罩/过滤器（需要所需运动的身体区域）

我们的第一个任务是看看我们是否可以将多个动作单元动画合并为单个演奏。这为性能和导演提供了连续性和一致性，并且还允许一些经济体在捕获所有这些动作所花费的时间方面。

然后，我们添加一个审查阶段，以确定一组需要定义的常见身体姿势，以允许动作单元之间的运动一致性。这由混合树本身和运动转换之间定义 - 所有站立的环境动作单元应以松弛的共同姿势开始和结束。这些常见姿势通常被命名并且在每个阶段内被关联，其涵盖多个动作单元和/或过渡。我们的任务和资产创建已经变成了层次结构和关联管理。对于整个过程来说，这是一个真正的主题。

接下来，我们将审查动作类别，类型和道具关联，并确保在舞台拍摄中合并多个动作单元仍然有效，并且在技术上和创造性上都有意义。然后，我们为每个合并的绩效考核分配一个有意义的阶段名称或 ID。这样可以简化管理设置，以便调出我们将捕获的采取和操作，因此所有部门都清晰，沟通顺畅。

通常当使用道具时，支柱上的常见手位置需要运动一致性。这些姿势在道具上贴上标签并进行物理标记。这些也与新形成的动画镜头列表相关联，该动画镜头列表是从动作单元列表导出的。

所需的常见姿势通常具有关键的脚部位置，尤其是在站立，就座和静态环境运动期间。这些通常数量较少，但通常会创建一个物理资产，以帮助确定每次捕获时的足部位置。这种物理资产通常是剪切或磁带标记，因此演员可以至少以正确的姿势开始运动（脚部位置确实可以帮助姿势的质量定位中心）和照片以供参考。我们还经常在捕获会话期间创建姿势的数字形式，并通过集合上的实时可视化将其用作参考。
在这个预生产任务结束时，我们有一些资产：
一个新形成的镜头列表，它将舞台拍摄和关联直接引用到动作单元
常见的姿势资产（定义，标签，描述，实物资产）
物理支柱资产
道具上的手位置姿势（道具上的定义，标签，描述和标记）

生产

现在我们已准备好进行拍摄。在这个阶段，值得注意的是，我们在集合中捕获的所有东西在技术意义上都不会是理想的，因为演员很难用相同的常见姿势或过渡姿势重复运动。我们通过将多个动作单元合并为单个舞台，在拍摄清单的准备过程中尽力而为，以确保我们在创作和技术上充分发挥演员的表现，知道我们将在事后制作一些动画艺术。每个行动单位在技术上都很有效。

准备是这些类型的芽的关键。预算决定了拍摄日数较少，并且拍摄了大量动作单元和移动过渡。

在任务和资产生成方面，拍摄非常简单。我们有我们的镜头列表，我们清楚地了解舞台拍摄中包含的内容以及获得所需性能的动作捕捉。这些拍摄日产生的资产包括：
2D 和 3D 动作捕捉数据，带有工作室时间码
视频见证相机素材，条纹与工作室时间码
根据拍摄列表拍摄笔记

通常在这些会话期间，通过新的动作单元或运动转换生成新的阶段，因为发现了实现期望动作的不同方式。通常，您只能做很多计划，而且需要处理好事情。手前的排练会有助于预生产！

在制作拍摄期间，首选拍摄被拍摄并圈出并在拍摄笔记中注明。这些用于后期制作以帮助

改进选择列表并为每个动作单元和运动转换生成期望的运动捕捉动画。

后期制作

在拍摄之后，然后执行与游戏/人群团队的精细选择过程。对于每个期望的个体动作单元和运动过渡（以及常见姿势），进行选择。这是从包含它的阶段拍摄到字面上的特定帧。这些帧范围选择通常由帧输入和输出定义，或者从输入和输出的时间代码自身定义。

完成这些选择后，虚拟制作团队将动作捕捉数据处理并最终确定为游戏装备上的动画。然后，虚拟制作团队使用与动作单元名称，类别，类型，道具关联和身体蒙版/过滤器相关的已定义元数据标签来标记动作单元或移动过渡。此过程通常使用脚本/工具自动执行，该脚本/工具从预生产期间生成的原始操作列表中提取。

然后将动作单元或动作转换命名为首选名称，并传送给游戏/人群团队进行实施，并传递给动画团队进行修改以获取演员的表现，并在技术上使其适用于所选的动作过渡，混合树，和常见的姿势（包括身体掩蔽/过滤）。

资产管理和任务分配的整个过程是一个相对简单的过程（基本上是分层关联和元数据标记），但由于动作列表的庞大规模，这一过程往往势不可挡。

第 11 节插曲 6：未来

你可以想象，上面的过程听起来很古老，绝对是一个缓慢的过程，需要大量的计划，多次射击以帮助迭代，以及灵巧的经验丰富的手开发这些混合树和运动过渡。通常需要大型压缩运动数据库（动作单元），并且可以消耗宝贵的内存和 CPU 周期以在需要时进行解压缩。

通过在游戏引擎中实时建模和模拟角色的动作，未来肯定更加明亮，从而减少对大量运动捕捉动画的依赖。直到几年前，这些建模和模拟技术与一组精心混合在一起并在其间转换的运动捕捉动画相比，在视觉上看起来并不是特别好。更新的技术和增加的 CPU 马力（在 GPU 和 CPU 上）现在允许在程序建模，动态和物理建模中使用一些非常令人兴奋的技术，以及运行时特征装备，允许实时 IK / FK 混合以适应环境和多种情况。许多研究和技术正在生成用于特定人（演员）运动的统计分析，可用于生成训练模型（统计学），然后可用于实时驱动物理和程序性角色动画。它们看起来很好，并且以一种让玩家沉浸其中并丰富游戏景观的方式进行互动。有趣的是，他们仍然经常需要动作捕捉来帮助训练这些行为和模拟，以及在需要时触发的关键事件的特定动画。这是一个不断变化的技术环境。

未来趋势和技术

第 12.1 节您本章中学到什么

在前面的章节中，我们研究了今天的生产管道是如何构建的。在本章中，我们将研究这些结构将来如何以及为何会发生变化。我们将探索新的数据交换格式和虚拟计算方法如何使公司更有效地进行协作;虚拟生产等新工作流程如何将预可视化和后期制作更紧密地联系在一起;以及高帧率电影和游戏作为服务的新媒体如何影响负责创建它们的工作室。

我们将首先看看主要影响视觉效果和动画的趋势，然后是那些影响游戏开发的趋势，尽管

这两个类别之间存在一些重叠。在本章中，您还将看到对基于云的工具和服务增长的参考：这本身就是一个重要的趋势，我们将在本书后面详细讨论。

第 12.2 节开放标准和开源工具

在过去几年中，开放标准在动画和视觉效果领域取得了很大进展。那些被广泛采用的产品对业界来说是一个福音，因为它们使得在应用程序之间甚至工作室之间交换数据变得更加容易。

首先被广泛使用的是 OpenEXR，一种用于存储高动态范围（HDR）数据的文件格式，用于记录真实世界的照明条件并在数字环境中重建它们。 OpenEXR 最初由 Industrial Light＆Magic 开发，于 2003 年开源，在 DCC 工具和管道中几乎普遍采用。这种格式继续由包括 ILM，Weta Digital，Pixar Animation Studios 和 Autodesk 在内的贡献者扩展，版本 2.0 将于 2013 年 4 月发布。

最近，最初由 Sony Pictures Imageworks 和 Lucasfilm 开发并于 2011 年作为开源发布的几何交换格式 Alembic 已经开始被用作 FBX 等专有格式的替代品。

随着生产范围的扩大，所需的交换标准数量也会增加。大多数视觉效果制作现在将多个效果供应商之间的工作分开。这通常意味着将镜头的前景元素分配给一个供应商，将背景元素分配给另一个供应商。在过去，这是相对简单的，但现在多个角色与环境交互的动作序列正变得司空见惯。将来，需要一种开放的元描述语言，以便设施更容易地从项目中摄取其他供应商创建的数据，并以可以通过代码而不是代码分析的格式将其交还给它们。完全通过自定义脚本。

工作室基础结构的某些方面现在也具有可行的开源实现。现在，许多工具都支持 OpenColorIO，它也是由 Sony Pictures Imageworks 积极赞助的，并且正在迅速成为色彩管理的标准。

现在甚至有很好的开源选项可用于管理网络上的软件部署，例如 Walt Disney Animation Studios 的 Munki 或 Rez，最初由 Dr. D Studios 的 Allan Johns 开发。

这只是当前可用的开放标准和开源工具的选择。在开始开发项目之前，绝对值得一看有什么可以在网上找到，因为修改现有的开源解决方案通常比编写自己的解决方案更容易。

第 12.3 节 WebGL 和相关技术

WebGL 是一种相当新的技术，用于在不使用插件的情况下在 Web 浏览器中呈现交互式 3D 和 2D 图形。它基于 OpenGL ES 2.0 图形编程 API，通过 HTML5 Canvas 元素公开，并支持 GLSL 着色器编程语言。 WebGL 使开发人员能够访问计算机的图形卡以加速图像处理和物理模拟，并与移动设备兼容，使其越来越受游戏和大型数据集可视化的欢迎。

WebGL 等技术表明桌面 DCC 应用程序很快就可以作为托管的基于云的服务在线提供，允许许多艺术家同时进行协作。这在视觉效果管道中开辟了许多新的可能性：工作室中的艺术家可以与那些人合作;具有一定艺术能力的客户可以提供超出书面笔记的反馈;删除评论可以变得互动，而不是主管只是查看图像序列或 QuickTime 电影;艺术家可以在世界任何地方工作，而不是局限于办公室。

虽然数据安全性也是一个潜在的问题，但这种管道带来的挑战主要与资产管理有关：将数据远程创建到 Maya 等应用程序中，以便它可以与管道的其余部分一起工作，并将其与 Shotgun 等生产跟踪系统连接起来。

因此，像 WebGL 这样的技术已经开始研究 3D 资产的版本控制。在某些管道中，资产是线性版本的，并且依赖于两个人同时编辑同一资产流。但是，随着工作流程变得更加协作，并发编辑

变得越来越普遍，非线性版本控制导致需要合并多个更改集。通常，这需要人工协助来管理冲突，因此现在正在进行研究以提供一个有效的框架，用于在对象级别而不是文件级别对资产进行版本控制和跟踪，以便解决问题。

第 12.4 节 GPU 计算

在 2010 年初，围绕 GPU 计算进行了大量宣传。使用 GPU 进行计算密集型任务被视为视觉效果的“圣杯”之一：这将在 CG 的许多领域提供急需的性能提升，尤其是渲染和模拟。然而，在撰写本文时，这种热情大部分已经消失。虽然大型设施确实利用了 GPU 计算 - 特别是对于专门的任务，例如渲染头发或 FX--开发逻辑所需的转变，仍然在不断发展的 GPU 和 API 的技术标准，如 OpenCL 和 Nvidia 的 CUDA，以及有限的调试资源让公司意识到 GPU 的开发存在巨大的开销。

相反，公司正在寻求更多的多核 CPU 和多线程 CPU 计算技术来改进他们的工作流程。许多工厂现在正在研究使用模板库（如英特尔的 TBB（线程构建模块）或 OpenMP）开发多线程算法，以便在 GPU 上的本机实现上提供高度并行和高性能的算法。

在比赛中，情况略有不同。实时渲染时，CPU 和 GPU 都很重要：大多数现代 AAA 游戏都无法在没有专用高端专用 GPU 的情况下运行。游戏开发者面临的关键问题是如何充分利用两者，而不是使用 CPU 还是 GPU。

第 12.5 节大数据

各种组织都会产生大量数据 - 而 VFX 组织也不例外。虽然它生成的数据可能与医疗保健，运输或国防产生的数据类型不同，但能够整理和挖掘数据使设施能够更有效地运行生产。只需解析生成的所有日志文件，就可能会获得有关管道实际使用方式以及常见问题所在的有用见解。但是，由于生成的数据量太大而无法供人处理，因此必须自动执行此过程。

近年来，为了以合理，集中的方式收集这些组织数据，挖掘商业智能数据并以人类可以理解的形式可视化结果，已经付出了很多努力。让我们看一下两个如何对 VFX 起作用的例子。

首先是渲染。调度渲染的常用方法是“先到先得”，根据项目或镜头的相对紧迫性等标准，在顶部放置某种形式的优先级。但也许我们可以更聪明。通过分析以前的渲染（对于相同的艺术家，同一个项目，相同的镜头，相同的资产或相同的软件），我们可能能够更好地决定如何排队和分配作业以充分利用计算资源可用。通过足够复杂的分析，我们可以做出如下决定：“此镜头的所有其他工作要么在第 10 帧之前失败，要么成功完成，所以我们只会运行前 10 帧。如果它们通过，我们就会' 11 开始同时运行其余部分，“或者”，这些镜头受 I / O 约束，并且不会受益于更强大的硬件，因此我们将在较不常用的旧硬件上运行它们。“

第二是安排。我们依靠艺术家（或更可能是监督者和制作者）来预测完成特定任务所需的时间。这可能会考虑到艺术家是处于初级，中级还是高级经验，但没有更具体的经验;并且可能只使用估计工作完成的复杂程度。通过分析任务数据，我们可以更详细地评估绩效，跟踪个别艺术家过去实际完成类似工作的时间。然后，可以使用此信息来构建更准确的计划。

第 12.6 节虚拟生产

“虚拟制作”一词对不同的人来说意味着不同：导致虚拟制作委员会成立的问题，美国电影

摄影师协会，艺术总监协会，视觉效果协会，预视化协会，和美国生产者协会，旨在定义这一新兴领域的标准。然而，虚拟制作的一个关键组成部分是在电影项目中采用实时图形技术。虚拟制作过程使导演能够查看与镜头的 CG 元素的占位符版本集成的集合上的演员的实时镜头。

虚拟制作现在正成为任何需要大规模视觉效果的电影的标准。在创建角色，生物和环境时，一切都是以数字方式完成的，从建筑服装到照明，动画和相机工作。虚拟制作的一个目标是从一开始就让导演参与到这个过程中。传统上，导演只能看到电影的一半：而不是数字环境，他们看到了绿色屏幕;他们不是使用数字角色，而是通过动作捕捉服装看到现场演员 - 或者更糟糕的是，用作替身的无生命道具。结果，经常出现错误，并且由于现场拍摄仍然是电影中最昂贵的部分之一 - 典型的估计范围从每分钟 1,000 美元到 2,000 美元 - 这些错误可能是代价高昂的。

虚拟生产的一个巨大好处是可以在生产中更早地回答有关 CG 工作的问题。而不是向 VFX 公司提供行动的书面描述，现在可以在导演甚至到达集合之前批准阻止动画。例如，在 Real Steel 上，所有战斗序列都在电影的预可视化阶段进行动画制作，这意味着实时拍摄本身可以在创纪录的时间内完成。

随着在预可视化期间创建更多数据，如何集成生产阶段的问题变得越来越重要。目前，除了将动画的 QuickTimes 视为参考材料之外，视觉效果设施通常不可能使用任何预览资产。因此，董事们经常抱怨他们看到 CG 从预可视化转向 VFX 时质量下降，因为效果供应商需要时间将资产提升到 previs 中使用的质量，然后才能改进在他们。非常需要更好地标准化材料在设施之间传递的格式 - 或者从项目一开始就让视觉效果公司参与其中。

效果公司面临的另一大挑战是如何捕捉所有相关材料。在虚拟制作过程中，导演可以批准从虚拟摄像机移动到实时拍摄期间数字环境的放置。无论拍摄是在具有高速内部连接的受控工作室环境中进行拍摄，还是在沙漠中的远程位置进行拍摄，都必须以随后可以摄入 VFX 设施管道的格式记录此数据。

一些主要的技术挑战涉及照明和捕捉面部表现。电影摄影师使用灯光来讲述故事，因此效果室需要能够通过使用场景中的物理实时源实时再现所有可用的技术。通过面部捕捉，面临的挑战是使所需的硬件更加便携，并减少设置所需的人数，使导演能够以更少障碍的方式录制演员。

通过对实时图形的依赖，虚拟制作将视觉效果和游戏世界更加紧密地结合在一起。游戏已经能够实时生成非常逼真的环境，FX 和角色表现，并且已经有了一些努力 - 例如，使用 Crytek 的 CryEngine - 使动画师能够在更传统的 CG 环境中工作，然后输出结果是游戏引擎通过虚拟相机可视化它们。

第 12.7 节高帧速率电影院

高帧率电影看起来可能成为视觉效果设施的下一个挑战之一。虽然媒体目前处于起步阶段，但像杰克逊和詹姆斯卡梅隆这样有影响力的导演似乎决心让高帧率成为立体电影之后的“下一个大事”。两者相互连接：更高的帧速率正成为立体电影的必要条件，以消除快速移动相机引起的频闪，并提高图像的清晰度。

高帧率电影看起来可能成功的原因是它不需要改变现有的传送机制。与需要电影院购买全新设备的立体电影不同，现代数字投影机已经可以处理更高的帧速率。此外，将以高帧率创建的电影转换回传统的 24 fps（每秒帧数）的问题似乎已经解决了。

视觉效果设施面临的挑战是必须生产的帧数。平均 VFX 拍摄时间约为 5 秒。在 24 fps 时，这需要 120 帧。在霍比特三部曲等电影中使用的 48 fps,它需要 240 ff。如果帧率上升到 120 fps，现在看来可能，那就需要 600.所有这些数字都是用单声道拍摄的电影：立体项目，他们将加倍。

同样重要的是要注意，这并不仅仅表示渲染时间的增加：许多其他部门受到影响，包括匹配移动，旋转，模拟，照明和合成。减少负荷的一个有趣的建议是以可变的帧速率进行拍摄，其

中拍摄不具有以较低速率拍摄的快速动作。

第 12.8 节虚拟机

虚拟机为购买大量专用硬件提供了灵活，经济的替代方案。在过去，如果一家工厂想要一台台式工作站，它会买一台真正的电脑。如果它想要一台服务器，它会购买另一台计算机并在其上安装 Web 服务软件。当该 Web 服务未被使用时，硬件将处于空闲状态。

虚拟机 - 执行程序就像物理机一样的软件系统 - 允许更灵活地使用该硬件。例如，公司可以在其现有硬件上设置虚拟机并在该虚拟机上安装旧操作系统，而不需要在其上安装旧版本 Windows 的单独计算机以运行旧版软件。然后，可以将虚拟机移动到其他硬件，而无需重建，重新安装或重新配置其软件。此外，与物理对应物不同，可以克隆虚拟机。在电影项目中设置渲染农场，以及在游戏开发中设置自动农场和浸泡/烟雾测试环境时，不难看出能够虚拟化机器是一个巨大的好处。

虚拟机的另一个优点是它们运行的　　硬件不需要存储在设施本身中。台式机，特别是用于视觉效果工作的台式机，需要顶级处理器，图形卡和内存。这些组件使用大量电力，并产生大量的热量和噪音。能够将硬件存储在更易于管理的环境中，具有冷却，不间断电源，故障转移等，不仅具有巨大的物流优势，而且使艺术家的工作环境更加愉快。

虚拟机还使艺术家可以在共享虚拟桌面上工作：例如，提供培训或审查工作。在许多工作室，工作人员分布在几个楼层，甚至几个独立的设施。虚拟系统使一个艺术家可以登录到另一个艺术家的工作站，而无需亲近他们 - 甚至在同一时区。

但是，仍有一些实际问题需要解决。游戏和视觉效果工作都需要处理器密集型应用程序，这些应用程序难以通过 VNC（虚拟网络计算）等旧方法共享。硬件供应商现在正在开发技术，通过这些技术,他们可以为共享同一虚拟桌面的用户提供真正的实时体验。示例包括 Nvidia 的 GRID VGX 平台或 Hewlett-Packard 的 HP Remote Graphics 软件。

为什么视觉效果需要云计算

在现在，这里，这里有一个云计算之前的轶事，可能有助于说明为什么它是必要的。在 Weta 上，总体视觉效果的另一个趋势是云计算的兴起我们将在本章末尾更详细地讨论这个问题。数字制作《指环王》三部曲期间，我是彼得·杰克逊的首席技术官那是 2003 年，还有两个月，国王就要返回发射场了。彼得·杰克逊召集了一个特别会议来回顾一些最新的场景他画了一些低分辨率的佩伦诺战场的草图，显示了罗汉的骑兵冲向山下，在米纳斯提利斯城前迎接兽人总共，大约有 10 万兽人和上万罗汉骑士。每个角色都是独立的

由称为 Massive 的软件生成并放置在战场上我的下巴撞到地板上了。仅渲染其中一个场景的单个帧估计需要几百个小时,我们的数据中心已经满负荷运行我们不可能拍到这些照片并按时完成电影我们在一个香烟盒的背面做了一些计算,然后去找三部曲的制作人巴里奥斯本,告诉他:“你有两个选择。一个是建立一个拥有一千个处理器的全新数据中心，以便及时完成所有工作“选择二是什么？“你可以告诉彼得·杰克逊他不能开枪！“显然，这不是明智之举。所以他开了支票，我们建了一个全新的数据中心。由于 Weta 是当时最大的客户，IBM 让新的刀片服务器放置私人飞机赶往新西兰每天都有直升飞机向现场投放装备数据中心在十月份建成,这是一项非凡的壮举，影片按时完成，但在不到两个月的使用时间里，额外的成本和努力是巨大的如果当时有云，w eta 可以远程租用额外的服务器容量，所有这些压力，成本和努力都可以避免。更不用

说白发了！

第 12.9 节游戏即服务

随着消费者的期望不断提高 - 与他们一起，开发新的 AAA 游戏的成本 - 传统独立游戏的日子可能即将结束。许多游戏现在都在考虑长寿，可下载的内容，每月更新的 MMO 以及不断更新的“免费增值”游戏都越来越受欢迎。随着构成游戏的内容开始从玩家的计算机迁移到开发者的服务器，构建“游戏即服务”的过程 - 部分或全部通过互联网提供的游戏 - 正在为管道开发人员引入新的任务。

首先，免费增值游戏需要自定义计费服务器，让用户购买可从中获取游戏收入的可选内容。开发人员可能还需要复杂的数据挖掘系统来跟踪玩家的进度并调整游戏玩法的各个方面，以确保他们保持参与，从而最大化利润。由于任何中断都可能成本高昂，因此这类游戏的管道还需要提供推出“修补程序”的机制，而无需关闭服务器。任何需要服务器脱机的修复都必须尽快完成。

此外，游戏即服务的管道需要在标题初始发布后持续使用多年。需要在管道的开发和管理方面投入更多的工作，以确保它不会随着时间的推移而崩溃，产生工作流问题并引入错误。

第 12.10 节作为服务的管道

如果一个游戏可以作为服务提供，为什么不应该用于创建它的管道呢？作为服务的管道（或“盒子中的工作室”）是一种概念，其中设施将以与任何其他软件产品大致相同的方式购买或租赁其整个管道。这对于为单个项目工作而创建的工作室具有特别的吸引力，然后解散;或者在交货时间很短的项目中，无法进行管道开发。

当面对这个想法时，许多开发人员的第一反应是，管道太复杂，无法作为外部服务提供。这可能是真的，但令人惊讶的是管道的许多个别部分已经外包。大多数大型开发商目前将动作捕捉，本地化和语音录制以及 QA 外包给专业公司;至少部分建模工作适用于人力较便宜的国家。

即使是以这种有限的方式租赁服务也会给管道开发商带来挑战。正如我们已经讨论过的那样，艺术家必须能够在与团队其他成员分享之前看到他们在游戏中创建的内容。外包内容提供商不适合这种模式，因为开发人员很容易信任他们，以便在提交之前为他们提供游戏引擎和测试他们在游戏中工作所需的所有工具。

外包内容提供商通常一次性提供数据。但是在游戏管道中投入大量数据并不是一个好主意。现代管道是高度迭代的，旨在最好地与恒定的数据流一起工作，使开发人员能够评估游戏中的每个小变化，并在下一个到达之前进行必要的修复。一次引入的数据越多，导致严重崩溃的可能性就越大。在与外包提供商合作时，通常需要专门的内部员工来处理他们生成的内容，一次将其引入系统。任务外包的下游是谎言 - 质量保证是一个明显的例子 - 需要分配给这项任务的内部员工越多。 Jon Peddie 总裁，Jon Peddie 研究员 Kathleen Maher 副总裁兼高级分析师 Jon Peddie 研究一切改变后标准的危机来到半导体改变世界

概述：转型中的产业

从某些原因到经济原因，另一些是技术原因，但它们的累积效应是行业历史上前所未有的变化率在影视领域，内容和发行的数字化已经到了一个转折点。从 2008 年到 2013 年，这个行业发生了剧本变，单靠数字很难沟通。内容仍在创建中，工具仍在购买中，但创建内容的人与十年

前创造的内容的人不同，当时使用的工具与今天使用的工具或将来使用的工具也不一样。最明显的变化是，知名公司已经倒闭，新公司也已经成立电影公司正在更换其内部的后期设施，并在全球范围内增加对合同房屋的使用。这一趋势伴随着数字化的进程而来，数字化几乎吞噬了生产线富士影业宣布将不再发行电影用于生产。豪华工艺房和彩色工艺房正在规划一个全数字化的未来据估计，2013 年，75%的电影制作都是通过数字方式完成的。数字化意味着电影不再用于电影制作，省去了昂贵的设备和更复杂的模拟工作流程甚至在数码相机开始使用的时候，在制作过程中使用胶卷的做法仍然存在，只要内容最终被分发到电影上。但电影作为发行媒介的终结比任何人预期的都要快在美国，这部电影已经开始初露端倪，即使电影将继续在许多其他国家使用，但电影制片厂相关发行商已表示，他们计划转向全数字制作和发行。

一切都变了

向数字化的转变已经改变了向电影和视频行业提供工具和服务的供应商的布局那些依赖由硬件和软件组成的“交钥匙系统”的公司受到了向软件转移的挑战。受害者包括知名品牌，如 AUTODESK，狂热，草谷，宽泰和汤姆森（现在被称为 Technicolor 的），所有这些公司都因转向数字业务而损失了很大一部分收入，而许多规模较小的公司已经完全消失。尽管人们工作的方式并不像技术那样总是变化迅速，但拍摄图像的过程正在变得越来越简化。令人讨厌的是，数码相机的实用性早在 2006 年和 2007 年就遭到质疑，但现在它们已被用作美国大多数电影制作中校准的摄像机和软件工具正在取代昂贵的系统，这些系统阻碍了专业内容创作进入小型设施。

邮政危机。

这种变化的涟漪正在扩散到创造和分布内容的整个过程中。效果和后期制作的利润率一直很低，因为公司不断尝试做以前从未尝试过的事情。当工作完成并重做做到完美的效果时，这些利润率就会下降，直到一些公司真的项目创意向数字化的转变逐渐增加降低了利润率，因为现在有更多的人能够承担这些项目。朝向创意 COW 的黛布拉考夫曼（Debra Kaufman）在上一篇优秀的文章中所写：设备已经运行多年了没有标准化的合同和切实可行的投标实践他们一直继续容易受到技术和工作室政策变化的影响。”尽管岗位需求并未下降，但就业岗位正在消失，尤其是在美国。现在，人们把责任归咎于愿意为更少的国家提供服务的国际供应商，那些提供美国公司无法比拟的补贴的地区，以及全世界公司愿意剥皮理想主义青年工人这很有可能是问题的根源，大型电影制片厂试图通过招标来压低价格，但也必须成为解决方案。制片公司将必须对作品的成本进行更现实的替代，并与他们的特效供应商建立更紧密的合作关系而不是，那些做这工作的人也必须变得效率更高许多反对者认为，这意味着需要使用标准工具，现成的互补和插件，以及更高效的内容和资产管理。

标准脱颖而出

随着越来越多的公司规模缩小项目，发现需要更好的协作和共享数据的方法：在一个靠创意和大揭露发家的行业，这是一个艰难的挑战。在过去，工具供应商无法使用专有格式来保护他们的系统：这些格式可能与其他类似工具交换数据变得困难。在某些工作流中，Autodesk 的 FBX 文件格式仍然很常用，但该公司难以保持其有用的性专业人士对工具的需求是一致的，这些工具使协作更加容易沃尔特迪斯尼动画工作室开发了 PTEX，一个更高效，更容易使用的纹理映射系统，并可以进行开源，而索尼图片公司的图像工作室也因此而实现了 Alembic 数据交换格式确实同样的事情。这两种技术的迅速采用表明我们将在未来看到更多这样的合作努力。现在工作流已经完全数字化，新的工具提供了创造内容的新方法会成为众人瞩目的目的焦点过去十年的一个很好的例子是数字雕刻工具的兴起，如 zbrush 和 mudbox。在未来，将有更多的现实捕捉工具的使用，包括摄影测量应用，扫描仪，和点云工具。另一个增长市场是用于协作工作的基于云的工具在云中处理大型文件的问题正被这样的硬件公司积极地解决例如 nvidia，amd 和 intel，它们正在加速必要的算法；以及融合 io 等存储公司，它们使用闪存来实现巨大的本地缓存和快速的服务器性能。

半导体改变世界

在过去的十年中，有两件事改变了内容开发行业：处理器达到了性能的物理极限，开发处理器人员利用了 parallel-的能力如图产品 12.4 所示，CPU 和 GPU 性能的提高同步过去，CPU 和 GPU 制造商都通过提高时钟速度来提高处理器的性能。然而，在某些点上，cpu 时钟速度不能再增加；虽然可以添加更多的单独处理器来提高性能，但 CPU 处理不会线性扩展。

由于图形处理和渲染的任务本质上是并行的，gpu 制造商认识到需要通过添加更多的处理器和开发一种使它们能够并行工作的体系结构来提高总体性能。因此，gpu 在性能上比 cpu 快了 270％这对这个行业产生了巨大的影响，特别是在游戏开发方面，GPU 被使用运行物理算法和处理场景的所有照明和阴影最后，新型硬件正在开发中在撰写本文时，想到力技术部门苛刻性钠专业公司（苛性钠专业公司）刚刚推出了一种专用的光线跟踪硬件加速器，可以改变光线跟踪图像的渲染时间。

插曲：VFX 云计算

从历史上看，拥有基础设施一直是 VFX 的主要（如果不是唯一的）模型。每个工作室都在专用（通常是现场）数据中心购买和维护自己的硬件。此私有云构成公司的中心，包含 renderfarm，存储和其他基本服务，如许可证管理，电子邮件，Intranet 托管等。

然而，随着云计算在 Web 应用程序和移动技术领域变得无处不在，它对电影制作也产生了影响。工作室开始采用云范例来管理他们的基础设施，而软件和硬件供应商正在提供利用这种商业模式的新产品。不幸的是，由于带宽容量，安全性和流水线集成等问题导致占用缓慢。

现在，工作室可以使用公共云将其中一些组件交给外部服务提供商，从而降低管理开销和运营成本。对于在全球设有办事处的工作室以及那些发现自己在项目之间不断扩大和缩小的工作室来说，这一点特别有益。实际上，混合方法通常与私有云和公共云解决方案的组合一起使用，具体取决于所讨论的服务。

虽然流行产品的商业模式可能更紧密地围绕网络需求进行结构化，但它们可以（并且仍然）与创意媒体制作非常相关（可以看出 Adobe Creative Cloud 等商业工具集的增长）。电影肯定有极端的数据和处理要求，但越来越多的专业云和服务提供商提供更适合它的调整模型。

云计算允许访问按需计算资源，通常以每分钟或每小时定价。他们可以将云计算视为一种实用工具，为其运行时所使用的内容付费，而在不运行时支付任何费用，而不必像 VFX 设施那样预先购买固定数量的内部机器来进行繁重的计算。从预算的角度来看，计算资源的支出从 Op-ex 支出转变为 Cap-ex，并且更容易且清晰地能够针对特定工作进行计费。

第 12 节插曲 1：云服务

云是一个有点模糊的术语，描述由提供商托管和管理的技术和解决方案，并通过互联网或专用网络作为服务提供。由于系统设计图通常将互联网描绘为云形，因此出现了“云”这个名称。云计算与诸如网格和效用计算之类的旧方法具有相似之处，其中资源被汇集并在许多用户或租户之间共享。然而，云计算在技术和商业模式的许多领域已经成熟，这导致了更广泛的采用和普及。云计算的三个关键特征定义为：

按需自助服务：消费者无需提供商的参与即可获得服务。

资源池：计算资源根据需求动态分配给消费者。

测量服务：底层系统计量每个消费者的计算资源消耗，允许他们只为他们消费的内容付费。

云计算的概念可以指代任意数量的不同设置。它可以像在 30 英里外的办公室中运行 10 个

相同服务器的人一样简单，在那里他们为特定类型的工作提供远程访问，或者更普遍接受的大规模亚马逊或谷歌数据中心，其中包含数十万种不同的机器类型。有信用卡的人可以使用的不同用途的数量。

云计算的主要类型（称为服务模型）是基础架构即服务（IaaS），平台即服务（PaaS）和软件即服务（SaaS）。

基础设施即服务

在用于云计算的该模型中，原始计算能力（CPU 和/或 GPU）和/或虚拟机可供消费者用于通用计算。这种“裸机”方法要求最终用户从操作系统安装和配置其环境的每个方面。这为工作室提供了最大的灵活性，因为它们可以随意运行和利用资源，但需要更多的管理。除了计算能力之外，数据存储也属于这一类。在 VFX 中，此模型可用于渲染，模拟，媒体转码以及数据或处理器密集型的其他通用操作。

平台即服务

作为服务模型的平台通过抽象，软件和开发工具（例如数据库，Web 服务器）的基本层提供计算能力，允许开发人员专注于构建应用程序，然后使用提供的工具打包和部署应用到平台。最终用户仅负责提供他们的应用程序，但是它必须与所提供的平台兼容（即，与作为服务模型的基础设施相比，诸如操作系统之类的基本组件的灵活性较小）。此模型通常仅适用于开发新应用程序，因为平台强加了某些约束，这使得难以移植旧应用程序。此模型通常用于提供基于云的渲染服务，其中服务随附已安装和配置的渲染器。

软件即服务

该模型通过云为最终用户公开完整的软件应用程序，也许是人们最熟悉的模型。最终用户无法控制底层基础架构或平台，只能与提供的软件（应用程序）进行交互。通常，这是通过 Web 浏览器完成的，特别适用于协作和生产力应用程序。

GPU 和专业云

一些渲染器可以利用 GPU 计算能力，并且在许多情况下可以显着提高性能。目前，公共云中 GPU 的可用性有限。 Amazon Web Services 提供数量有限的 GPGPU（通用图形处理单元）。但是，您可能会在接下来的几年中看到 GPGPU 更丰富。除了在配置时选择适当的实例类型（当然还有使用支持 GPU 的渲染器）时，除了选择适当的实例类型之外，没有任何特定的方法可以利用这些功能。请注意，在 Amazon Web Service 中，GPU 仅在某些区域可用。

专业云也越来越受欢迎。这些是专注于特定垂直市场需求的云。例如，对于数字媒体，他们将拥有 GPU 功能，可能是 iSCSI（互联网小型计算机系统接口）存储接口，能够轻松设置虚拟桌面以运行远程 Maya 等软件，以及更高速的 CPU（大多数公共云具有相当低的性能） -spec 商品 CPU）。

第 12 节插曲 2：使用云

云计算将在未来几年内重塑生产的一些重要方式。本节介绍了在将云用于电影制作时需要考虑的一些挑战和注意事项。这些挑战将更多或更少，取决于所涉项目的范围，即其复杂性和商业性质。

定价

在考虑云计算选项时，重要的是要充分了解预算约束以及现有基础架构。云计算的好处在于能够在需要时动态轻松地访问计算资源。云不需要投入昂贵的前期成本高昂的硬件，而是允许您在需要时仅为您使用的内容付费。

另一个问题，特别是对于持续工作的大型设施，是定价。利用按需付费模式对于中小型设施和自由职业者来说具有经济意义，特别是在维护硬件时，偶尔需要大量计算能力的不一致的工作流可以从云服务中受益。规模较小的工作室和初创企业可能会发现云服务的灵活性和低前期成本使其成为构建大型传统 IT 系统的极具吸引力的替代方案。较大的工作室，特别是如果他们已经在硬件和人员方面进行了大量投资，则需要非常仔细地考虑使用托管解决方案和软件服务补充或替换现有功能的成本和收益。然而，较大的设施定期提供工作经常发现购买和建立自己的基础设施，当涉及渲染和存储等重型资源时，仍然比他们每周 7 天每天每小时支付更具成本效益。规模。

可扩展性

云对视觉效果行业具有明显的吸引力，因为总是有极端商业时期和低时期。在项目的最后几周，生产需求完全失去设施内可用资源的情况并不少见。建造一个完全满足这些需求的工作室并不符合成本效益，因为这样的构建费用太高。在这样的关键时刻，更好的安排和计划以确保项目的交付更为重要。云提供了在需要时提供这种额外基础设施的潜力，而无需在现场物理安装，测试，配置和维护盒子的必要性（和延迟）。

云允许您经济地横向扩展，但可以限制您可以在多大程度上垂直扩展，因为您拥有更受限制的服务器选择，除非您运行私有云。理想情况下，您希望构建一个可以在两个方向上扩展的管道。水平扩展（也称为扩展）是添加更多服务器;或者更好的是，根据需求动态扩展，增长和缩小。垂直扩展（扩展）是通过更改 RAM，CPU，GPU 等来更改服务器的资源。在私有云中，您选择服务器;在商业云中，您只能使用服务提供的内容。

连接和文件传输速度

由于云需要将数据远距离发送到远程数据中心，因此某些工作室可能会担心连接问题。由于创建渲染帧需要许多千兆字节的源数据，因此必须在工作室和云之间有效地传输大量数据。因此，连接的延迟和带宽可能成为采用基于云的解决方案的限制因素。在某些情况下，数据传输通过开放互联网进行;对于私有云，这可以通过专用的专用网络来克服。使用云服务的一个明显的瓶颈是来自 CG 渲染和高分辨率实时板的大量数据集，以及如何从云位置获取这些数据集。没有设置带宽可以保证性能，并且试图预测何时发生这种情况并不是很有用。也许当我们可以实时将

HD.exr 帧上传到云端时，我们都需要提供 8 K 立体声工作！

一个重要的潜在缺陷是互联网速度和可靠性的广泛分歧。例如，到目前为止，中国的外包供应商很少能够将高速，不间断的访问视为理所当然 - 许多使用基于 Web 的自定义工具分发的游戏工作室难以与中国合作伙伴合作，这些合作伙伴不能指望“永远关于“访问全球网络，其工作人员通常不直接从他们的桌面连接互联网。

当涉及到移动大量数据时，互联网无法与内部网络的带宽相匹配。因此，大型文件的云处理具有很长的内置启动时间。由于艺术家必须通过多次迭代对图像进行微调，因此在可预见的未来，他们将继续喜欢快速的本地资源和云中较慢的机器。然而，当关键时刻到来时，任何有用的东西都会受到欢迎。特别是对于中小型工作室而言，快速扩展到数百个额外处理器的能力至关重要。

如果要将云用作渲染管道的组成部分，那么专门转移资产差异的解决方案可能值得投资。常见的实用程序应用程序 rsync 是仅在文件版本之间发送增量的解决方案的示例。然而，文件管理的软件开发的进步越来越多地允许直接访问用于呈现和存储的云资源而无需人工干预。

这种直接访问集成和存储不仅支持 3D 渲染，而且还可以在云中渲染 2D 合成，例如 The Foundry 的 Nuke，由于所需元素的大小（包括实时动作和 CG），以前被认为是云不友好的复合材料。

一种更实用的方法是从一开始就将内容创建中涉及的许多任务转移到云中，而不是试图解决带宽上传问题。如果在云平台上呈现任何 CG 元素，则它们可以保留在整个内容创建过程中，用于合成，编码和分发。使用此过程，可以在采集时立即上传实时动作板，因此在需要渲染或编码或颜色时，它们是可用的。 VFX 流程导致如此多的数据重复，通常最好从头开始上传，因此随着数据集变大，未来的上传时间最小化。

无论您如何处理数据传输，拥有备份网络路由，与 ISP 的良好关系以及备份 ISP 都可以降低灾难影响您的日程安排的风险。当在云中渲染或异地存储数据时，外部因素，例如通过光纤骨干网的建筑切割，ISP 供应商变电站的爆炸，或其他一些自然灾害（沿着数据路径某处的地震），可能会使系统看起来很慢。如果通过微波炉传输数据，或者由于记录的降雨量最大而导致基础设施被淹没，即使下雨也可能是一场灾难。尽管存在“可靠的”互联网，但任何这些灾难都可能发生，并且总是在最糟糕的时间 - 在紧急情况下和交付之前。

资产管理

使渲染管道在云中运行的主要挑战之一是准确检测特定场景所需的资产。如果无法以编程方式识别这些依赖关系，则场景将无法正确呈现，并且所消耗的任何计算时间都将被浪费。

许多渲染器（如 Maya，V-Ray 和 Blender）支持单个场景文件，这些文件封装了渲染该场景所需的所有内容。但是，对于更复杂的项目，通常涉及许多外部资产。Maya 中的 RenderMan Studio 将为单个场景生成数万个文件，包括描述场景和几何的 RIB 以及纹理，着色器和其他外部插件。目录结构（称为 RIB 树）是神圣的，在渲染 RenderMan 场景时需要在云中保留。一种选择是在每个渲染节点上的云中运行 Maya，并使用 Maya 启动渲染。这消除了导出 RIB 并降低复杂性的需要。

第 12 节插曲 3：合作

云服务提供易于分发的信息，管理费用低，安全性高，无需您将自己的内部网络暴露在互联网上。托管软件，如 Shotgun 或 Tactic，专门用于管理电影和游戏制作，使其更容易远程协作。

此外，具有这种按需功能和灵活性对分布式员工来说非常有意义。委托 CG 艺术家在三个月的项目中远程工作要容易得多，当你知道他们能够在云中的五十台机器上渲染并且比在十个专用上花费相同数量的金钱更快，更便宜地工作时渲染他们必须设置和管理的节点。

一旦决定利用云作为独立功能或补充现场渲染农场，就可以产生其他潜在的好处。其中之一是与其他工作室和数字艺术家的合作，云作为分布式劳动力的中央存储中心。由于数字资产已经存储在云端，因此可以轻松地与其他艺术家共享这些资产，向客户提供反馈，让其他人为您的项目做出贡献，并为其他项目做出贡献。将所有资产放在一个可供多个工作人员使用的位置，意味着无需购买昂贵的内部存储空间，只能由当地员工访问。相反，您拥有一个无限可扩展的解决方案，您的全球员工都可以访问该解决方基于云的数字市场可以促进资产的共享和货币化。这些概念今天有些不发达，但这种情况在未来几年内不可避免地会发生变化。

备份和冗余

很少有公司能够承担大型云提供商所享有的大规模数据安全投资。手动创建和存储本地服务器的基于磁带的备份是当前的标准，但随着面向档案的云服务（如 Amazon Glacier）的兴起，工作室可能更愿意通过 Web 远程处理备份到安全位置的多个数据中心，而不是让他们回到 IT 主管的背包里回家。较小的工作室将特别受益于访问安全备份。

安全

云的一个显而易见的考虑因素是它公开托管并通过互联网或专用网络访问。对于某些工作室来说，这可能是一个问题，尤其是商业项目。 VFX 在云端工作的主要犹豫之一就是担心他们的数据会被异地传输;许多电影制片厂在与这些设施的合同中严格禁止这一点。尽管许多大型云提供商提供了一套严格的安全指南，并采取措施确保未经授权的数据访问比在其设施中访问本地更加困难，但将“受到保护的数据”移到“异地”的担忧却吓跑了许多艺术家。和工作室使用云。

数据需要上传到云平台的存储，并且至少在场景渲染时驻留在云中。然后需要在现场下载渲染的输出。这意味着您正在传输数据并存储数据。这被称为“传输中的数据”和“静止的数据”。要保护您需要解决的数据：

使用传输加密可以轻松保护传输中的数据。 HTTPS 是加密通过互联网传输的数据的标准。 所有主要云提供商都支持使用 HTTPS 将数据传输到其存储中。

静态数据受云提供商保护，但未加密。

提供商已经进行了大量投资以保护他们的平台。有些人在其存储服务中提供双因素身份验证，这使得任何潜在的黑客都很难破坏您的数据。如果仍然不够，则可以在上传到云之前对数据进行加密。这增加了显着的复杂性，并且通常不保证大多数用例。

更复杂的是，MPAA 缺乏真正的认证，即云被其标准“授权”。云提供商最接近此类认证的是由第三方进行安全审核，该审核根据 MPAA 标准进行评估并授予“最佳实践”结果。该标准已由一些云提供商实现。现在，VFX 工厂的责任是通过说服电影制片厂将云服务足够安全地融入到制作中来完成交易。加利福尼亚州埃默里维尔的原子小说成为云计算成功的一个例子，派拉蒙影业公司批准为罗伯特·泽米克斯/丹泽尔华盛顿电影“飞行”渲染超过 400 个 VFX 镜头。在 MPAA 颁发实际安全认证之前，主要的电影工作室对自上而下的云服务和安全性充满信心，向工作室销售云服务将主要是针对 VFX 设施的每个项目的个别运动。

可靠性

在使用云计算环境时，必须考虑服务的可靠性。可靠性（通常称为正常运行时间）以服务可用的一年内的时间百分比来衡量。云服务提供商提供超过 99.9999%（称为 6 个九）的正常运行时间，使服务每年仅减少 31.5 秒。

管理许可证

许可是一个重要的考虑因素。许多为 VFX 行业生产软件的公司仍然建立在传统模式之上，预先销售昂贵的许可证，其中包括有关在云中使用其软件的限制性政策。云的关键价值是能够在几分钟内扩展数千个节点以处理突发需求。但是，在云中实施渲染管道时，购买额外年度许可证以在每个节点上运行的成本可能是一个限制因素，并且在他们的 EULA（最终用户许可协议）中，供应商可以明确禁止他们的许可证到除非获得明确许可，否则请在“云”中运行。

随着越来越多的用户将其工作流程迁移到云端，公司无疑会调整其策略以适应这些考虑因素。诸如 Adobe 的 Creative Suite 和 Shotgun Software 的基于云的项目和资产管理工具等供应商已经开始朝着这个方向迈出了一步。

市场上有一些解决方案，Pixar 或 Autodesk 等关键软件供应商在云中管理他们的许可证并计算最终用户的消耗量，最终用户又享受支付一笔费用的优势，在某些情况下可以按毫秒计量，监控消耗计算能力，存储和软件许可证。这意味着可以为渲染作业配置数千个核心，而不会产生昂贵的许可影响。

包围临界

云服务和整个“软件即服务”模式的缺点之一是它使您的公司成为服务提供商的一种特殊的服务。如果您拥有运行 3D 软件的 DVD 和硬件密钥，您可以坚持使用满足您需求的版本，并继续使用适合您喜欢的版本的工具和设备。如果您是服务的订户，则无需选择何时以及如何升级。云服务的灵活性和内部占用空间很小，但是当它是您自己的硬件并由您自己的员工运行时，您是唯一一个决定管道何时以及如何更改的人。在某些方面，它类似于在 DVD 上购买电影和订阅有线电视之间的区别：很高兴不必管理一个装满光盘的音箱，但你可以控制日程安排，而不必担心你最喜欢的节目被取消警告。

绝对管理

在云中配置渲染环境可能需要相当多的努力，特别是如果这是一个自动化功能。需要考虑的主题包括：

根据需要自动配置渲染节点

能够排队渲染作业

一种将作业分解为单个帧并将其分布在可用渲染节点上的机制

将特定帧的所需资产从云存储下载到每个渲染节点上的本地磁盘

将渲染的输出上传回云存储

保存并提供渲染器的任何日志文件和标准输出，以便在作业失败或输出不正确时轻松执行诊断

能够监视每个渲染节点上的可用内存
在不再需要实例时自动关闭实例

还有关于如何远程与系统通信的考虑因素。使用 Web / REST API 就是一个例子。当然，上述中的一些可以根据需要手动启动和配置，这可能更适合偶尔使用，而不是构建完全自动化的云渲染系统。

应用程序和管道集成

一旦云功能到位，您需要考虑如何与该系统连接，同时避免向管道引入摩擦。需要支持的两个主要用例是数字艺术家渲染测试场景和呈现所有最终资产。第一个用例要求能够轻松地从艺术家的环境中将渲染卸载到云端，例如使用 Python 集成的 Maya。工作流程包括：
确定场景是否具有任何外部文件依赖性
将场景和依赖项上载到云存储，即 Amazon Web Services 中的 S3 或 Windows Azure 中的 Blob 存储
向云渲染系统发送指令，指定任何其他渲染设置

记忆

了解需要多少内存进程并确保充分指定云中的虚拟机非常重要。 如果达到内存限制，性能将急剧下降 - 或者更糟，渲染将失败。 较高的内存实例通常花费更多，因此在创建场景时需要密切关注内存要求。

词汇表

二维（电影、游戏）是二维的缩写当用于图像创建时，通常指平面手绘风格当用于图像输出时，指的是将图像输出为二维媒体的最后步骤（合成和电影制作的 DI 阶段）在游戏中，也常指游戏的展示和游戏性仅限于一个平面的情况。

2.5D（电影、游戏）一种将绘画投影到简化几何体上以赋予深度感的方法以及环境的视差这种投影映射技术通常用于蒙版绘画和设置扩展,基本上采取一个单一的图像和投影到一个简单的三维几何。在游戏中，2.5D 指的是视觉元素或游戏在两个维度，第三个维度中的有限自由（例如，可以在起伏的景观，但不能跳跃或垂直移动离开其表面）。

3D（电影、游戏）是 3D 的缩写可指胶片的立体部分，或采用三维几何的计算机图形，或计算机图形处理作为全部有些是指 3D 表示立体内容，另一些是指 3D 表示三维成像。

3D 转换（胶片）参见尺寸化。

3D Studio 请参见 3DS Max。

来自 Autodesk 的 3DS Max（电影、游戏）三维建模和动画包最初的“3D 工作室”“最大”，

通常被称为“3D 工作室”或简称为“最大”。

911（电影）一个落后于计划的项目，被送到另一个工作室做一些紧急工作准时交货。

ACES（胶片）学院彩色编码规范由电影艺术与科学技术学院委员会定义的新颜色空间，提供更精确的颜色管道

动画（电影，游戏）一个粗略的，分块的镜头（可能渲染成灰度，只有关键点）姿势动画）用于建立动作“动画片”仍然更常用于指手绘故事板艺术与一些有限的动画一起编辑(普雷维斯几乎总是指的是 CG 创造的材料。）这也用于游戏中，通常指的是使用静止图像或非常简单的模型组合起来的游戏或剪切场景序列的版本，生动地给出了节奏和流动的概念在游戏中，这通常与 previs 不同动画永远只是视频序列，而“previs”实际上可能是运行在一个游戏引擎与粗糙的艺术品和测试游戏代码。

AOV（film）任意输出变量，最初在 Renderman 中用于将数据（曲面法线、光照过程等）输出为图像用于照明和阴影。

应用程序编程接口的缩写一个（或多个）库，作为一个软件的公开入口点，允许其他开发人员以确定的方式与该库交互通过调用 API 中的方法，开发人员可以执行一个特定的函数或进程，并获得一个结果，该结果可以合并到一个较大的软件或进程中。

应用程序编程接口见 API。

批准（电影、游戏）对资产或镜头完整状态的描述。

阿森纳（电影）开源渲染管理软件最初由 Blur Studios 创建艺术家特定内容的创造者。

Aspera（film）商业软件，允许公司使用私有的高速数据传输进行协作资产（电影、游戏）术语资产有两种相关的使用方式一般来说，资产是产品中可识别的内容，例如字符、环境或道具但是，用于向观众呈现可见内容的存储文件的整个集合也被称为资源：例如，角色被非正式地称为“资源”，但实际上可见角色反映了许多不同的资源：模型、动画、纹理等不同的文件可能会在不同的情况下呈现（一个低分辨率的模型用于远距离拍摄，一个更详细的模型用于特写镜头），但它们都包含了一般意义上的相同资源。

资产管理（电影、游戏）跟踪、管理和版本化镜头资产的过程这通常是一个带有数据库后端的系统，在文件系统上存储物理资产的元数据资产管理系统将跟踪简单的信息，如创建日期和用户，但也会跟踪更复杂的数据，如上/下游依赖关系。

Avid（电影、游戏）创建商业编辑包 Avid DS 和 Alienbrain 资产管理的公司 Avid 以专业电影编辑使用的媒体作曲家编辑软件而闻名。