摘要

该项目目的是结合虚幻游戏引擎实现在虚拟现实环境下的三维模型交互。该项目的实现方法是在基于虚幻引擎高效的渲染能力下，将模型读入引擎中，由虚幻引擎特有的蓝图可视化脚本获取模型数据，通过 C++编程，将虚幻引擎中的 FBX 模型数据转换为方便交互的半边数据结构。根据虚幻引擎中的用户在虚拟现实环境下的手柄的拉伸，对半边数据结构中模型的点，线，面以及一些数据结构中特有的信息进行实时地更改。最终转换为虚幻引擎可接受的数据结构，再依靠存储的模型拓扑信息生成新的模型，完成在虚拟现实环境下的三维模型交互。

关键词： 虚拟现实，虚幻引擎，半边数据结构

Abstract

The purpose of this project is to combine the 3D model interaction in the virtual reality environment with the Unreal Game Engine. The implementation method of this project is to read the model into the engine based on the efficient rendering ability of Unreal Engine. The blueprint data visualization script unique to the Unreal Engine acquires the model data. Through the C++ programming, the FBX model data in the Unreal Engine is converted into half-edge data structure for convenience of interaction. According to the stretching of the user's handle in the virtual reality environment in the Unreal Engine, the point-specific data in the data structure of the half-side data structure is changed in real time. It is finally converted into an Unreal Engine acceptable data structure, and then relies on the stored model topology information to generate a new model to complete the interaction of the 3D model in the virtual reality environment.

Key words: Virtual reality, Unreal Engine, Half-Edge data structure

目录

第一章 绪论

1.1 课题研究背景

1.1.1 中国 VR 市场概况

VR 在中国的市场规模和需求快速增长

尽管中国虚拟现实市场的规模还处于萌芽状态，但是中国市场规模增长非常快速，中国虚拟现实市场规模在 2016 年将达到 35 亿元,在 2018 年，VR 市场规模翻了三番，[1]超过百亿元。在未来五年中，VR 市场规模的年增长率将超过 80%。按照当前的增长速度，到 2021 年中国 VR 市场规模将到达 790.2 亿元，全面超过欧美发达国家。

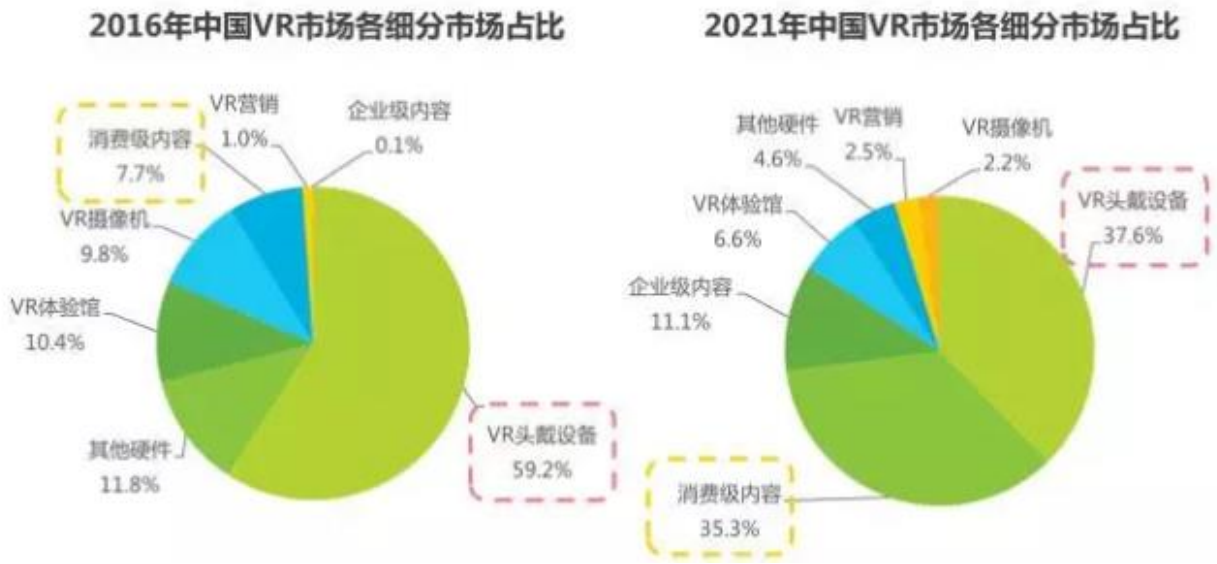
图 1 2016-2021 年中国 VR 市场规模

1.1.2 VR 内容市场规模有望超越 VR 头戴设备

在 2016 年，VR 头戴设备在中国虚拟现实市场居于主要地位[2]，以 20.5 亿元的规模占据整体份额的 59.2%。VR 内容市场（包括教育，娱乐，企业应用，营销等行业）尚处于萌芽发展状态,但其因为需求巨大,规模在未来势必会增长迅猛,市场规模将保持每年 163%的增长速度，在 2021 年超过 380 亿元。届时 VR 内容市场规模将取代头戴设备成为中国虚拟现实市场霸主。而 2015 年初始崭露头角的 VR 设备体验馆也将随着 VR 内容市场的不断

丰富和扩大，在未来几年持续增长，在 2021 年将达到 52 亿元的市场规模，排在 VR 头盔和 VR 内容市场之后。

图 2 中国 VR 市场各细分市场占比



1.1.3 VR 内容市场收益分析及预测

大众消费部分：以 VR 为内容的各行业项目将开始逐步盈利，VR 游戏行业将成为最大盈利行业。

消费部分的 VR 内容包括影视、直播、游戏、其他四大类型。[3]VR 消费市场在 2019 年将实现巨大转折，在这一阶段，包括游戏行业在内的主要的内容制作商会逐渐转亏为盈。2021 年，VR 消费级内容市场的规模有望超过 270 亿元，其中 VR 游戏的规模占比接近 35%，其市场规模预计为 96 亿元。VR 影视内容的市场规模也将超过 87 亿元，略低于 VR 游戏规模，排在第二。

图 3 2016-2021 年中国 VR 消费级市场规模

企业级：市场规模年增长率超 300%，VR 教育和培训需求持续增大，推动行业发展

尽管现在 VR 内容市场中的企业级内容的比例还低于 1%，但其未来市场增长势头良好，年增长率为 355%，由于基数小，远远超过 VR 消极级中增长率最高的游戏行业。。尤其是教育和培训企业对 VR 内容的需求甚大，将导致 VR 企业级内容市场实现飞速增长，并且近年来我国对 VR 教育和培训企业的信息化也提供了许多政策层面的帮助，政策对 VR 教育的重视势必会进一步促进企业级 VR 内容市场规模扩大。

图 4 2016-2021 年企业级内容市场规模及增长率

1.2 VR 建模现状和本项目意义

Oculus Medium

现如今市场上主要的 VR 建模是 Oculus Medium。[5]

Oculus Medium 是一款专为 Oculus 虚拟现实设备设计的交互应用，用户可以在虚拟环境之中雕刻模型以及用交互设备建模，以及绘制和创建有形对象。Oculus 公司最近新增了“移动工具”，用户可以重新布局，拉动，拾取模型的任何一部分。[6]

Medium 的设计灵感来源于让设计家和艺术家们从已有的 2D 平面中脱离出来，利用数字工具，借助计算机图形和色彩，并且可以在一种全新的视角下交互和创作。

在 VR 环境下创作三维模型的最大好处是：我们可以免费获得 2D 平面中没有的立体感，比如大量的纵向和音频信息及三维空间立体渲染和设备跟踪。用户也没有必要学习如何调整在 VR 中，而在显示器中，这个步骤可能需要一些练习。视角在 VR 中，摄像头就是我们的头，我们和三维模型处于同一个空间，关系自然，大小和谐。这些元素会让我们和模型产生自然的连接。在 VR 环境中用手柄雕刻，操纵放置虚拟粘土，就像是在现实空间中一样，只不过在 VR 中没有重力和其他一些物理规则。[7]

Oculus Medium 虽然好处多多，但是从模型本身分析，用户在 VR 空间生成的模型是基于多个网格来实现的，即整个模型分属于不同的网格。这种做法在建模时会大大减轻建模的复杂性，但是另一个弊端就是建好的模型无法进行打印，而本项目对模型进行交互是基于一个网格物体上进行操作，虽然在模型更改上会有些限制，但是可以对整个模型进行 3D 打印。这对于将人们在 VR 场景的创作转换成实体模型很有帮助，VR 建模不再是一个只能存在于计算机上的技术，而是可以真正和我们的现实世界相连接。[8]

第二章 虚幻引擎

2.1 引擎概述

[9]虚幻引擎 4 经过 1988 年虚幻引擎第一带开发以来，为使用实时开发的业内人员开发的完整开发工具。从企业应用和游戏体验到高品质的个人电脑、主机、移动端、VR 及 AR 游戏，虚幻引擎 4 能支撑项目从雏形发展到项目发行这一系列工作流程，在和寒霜引擎以及现在在手游端流行的等同类产品中优势明显。

2.1.1 实时逼真渲染

虚幻引擎基于物理的渲染技术，支持最先进的 DirectX11 渲染语言，高级动态阴影选项高级动态阴影选项、屏幕空间反射以及光照通道，物理模型通道等强大功能可以实现电影特效级别的视觉效果，让人感觉更加真实。

2.1.2 引擎公布 C++源代码

与其他游戏开发引擎不同，虚幻引擎向开发者开放了引擎的所有源代码，这意味着每一个开发者都能按照自己的想法修改引擎，更有助于开发者们的创意性设计和引擎自身的

优化。并且 C++语言和 C#，Java 等语言相比较，虽然逻辑上比较复杂，但是它的优点是有更高的执行效率，能够让引擎更流畅地运行，实现更为强大的功能。

2.1.3 蓝图：可视化编程

虚幻引擎的蓝图节点脚本对设计师帮助巨大，即使没有编程经验的人也可以快速制作出一个项目雏形。蓝图可以用来构建游戏世界中模型，规定模型的逻辑运算，创建交互，可以这么理解：蓝图就是经过简化版本的 C++，它同样有变量，函数，结构体等定义，但是这些基础的内容以及逻辑之间的连接，我们不需要像编程一样用代码来完成，而是可以通过拖拽一些控件，来组成我们的游戏框架。

2.1.4 稳健的多人框架

从 1988 年以来，在多个系统平台上已经有各类游戏采用了虚幻引擎多人框架来搭建，其中许多游戏都是业内顶尖水平。[9]近来虚幻引擎推出的“安装可用”型客户端/服务器端架构可拓展性极高，得到开发者们一直好评，在框架内开发者可以自行设计多人游戏组件，并发布到开发者社区，实现“即插即用”。

2.1.5 电影级后期处理效果

开发者可以借助虚幻引擎的后期处理轻松地调整场景的外观和感觉。用户只需调用一些虚幻引擎中已经完成的场景特效，即可体验电影级别的视觉盛宴，包括环境立方体贴图、环境遮挡、光溢出、颜色分级、景深、人眼适应、镜头光晕、光束、随机采样抗锯齿和色调映射等众多实用功能。

2.1.6 灵活的材质编辑器

虚幻引擎 4 中的材质编辑器采用基于基础物理的像素着色技术，与蓝图相结合，可以让开发者更加直观地感受到颜色的生成，便于我们的修改和调试，其中包括颜色，混合，光照，粗糙等一系列材质属性都可以由开发者自己定义，开发者可以拥有对模型和角色的外观和风格绝对把控力。通过和蓝图配合形成的直观工作流程来创建多种生动形象的材质贴图，可以由设计师独立完成的材质贴图和数字化的材质参数能制作出任何我们梦寐以求

的感觉。

2.1.7 包罗万象的动画套件

开发者能够通过虚幻的网格体以及动画编辑工具实现游戏角色的完全 DIY。动画工具中实用的功能有状态机、混合空间、逆向运动学和由动作驱动的物理特性。[1]开发者可以在一种特殊蓝图——动画蓝图中对模型动画进行实时更改，通过即时预览动作，配套外在动作捕捉套件，就能创造出形象逼真的人物动画。

2.1.8 完整的 VR 模式编辑器

在虚幻引擎的帮助下，我们可以通过 VR 头盔和手柄，来完成在 VR 模式下的游戏编辑。不同的是用手柄取代了鼠标移动，用扳机取代了鼠标点击。在 VR 编辑器中还有配套的操作面板，让我们的编辑过程更加轻松，大大增加了开发者身临其境的体验和开发过程中的乐趣，虚幻引擎是目前引擎中最稳定、功能最完整、最实用的 VR 开发工具。

2.1.9 专为 VR 而生

Epic 公司一直与全球顶尖硬件公司，如英伟达公司等以及一些游戏软件开发商保持交流和合作。虚幻引擎能够为 VR 创作者带来最佳的使用和视觉体验。[9]通过与最流行的各大平台实现本地集成，以及前向渲染、多采样抗锯齿以及实例化双目绘制，以及单视场远景渲染等优化手段，虚幻引擎可以导出极佳质量的展示结果。此外，Epic 公司也帮助促进了 OpenXR 领导的 VR 规章化和标准化进程。

2.1.10 植被与地形

我们可以使用虚幻引擎的 terrain system 创建广阔的，丰富且独特的世界环境。这主要是因为虚幻引擎地貌系统强劲的 LOD 系统和对内存的高效分配，开发者可以轻松创建出更广阔的地图。同时使用 Landscape Grass 功能用各式各样的地貌覆盖我们创建的地图，还能使用植被工具对游戏地貌进行多种类的装饰，这对于要制作庞大的游戏世界的开发者来说确实是一个很大的福音。

2.1.11 虚幻音频系统

我们项目的音频效果可以借助系统实现突破性的升级，包括实时合成、动态 DSP 效果以及物理音频传播模型。此外，Value 公司已经与 Epic 公司建立合作，我们可以在 UE4 中使用 Steam Audio。[10]Steam Audio 有助于游戏中的音频立体化，并且使传输更加有效，解决了跨平台的问题，能够使 VR 中的音频更加真实。

2.1.12 内容浏览器

开发者可以借助内容浏览器对虚幻引擎中产生的大量游戏数据和外部导入的资源进行管理、查找、添加关键字、过滤及修改。虚幻引擎支持资源直接拖入内容浏览器并从内容浏览器将其直接放入屏幕以创建属于我们的世界。资源集可以帮助开发者完成集体开发和交流。

2.1.13 商城生态圈

虚幻商城拥有众多由开发者们分享的或者是由官方发布的高质量的资源及插件，这些引擎可以省去我们制作模型的时间，从而专心的投入到游戏的制作过程中，其中具有动画等高级功能的模型可快速使我们的模型角色成型，加快整个模型的制作过程，开发者们还可以通过商城获得全新的地形、人物、动画、材质、模型、音频及电影中的一些特效、音轨、用蓝图完成的插件、帮助开发者操作更便利的辅助插件以及完整的初学者资源。

2.1.14 无限可扩展性

虚幻引擎项目可以通过模块化插件系统来集成任何功能。并且由于虚幻引擎资源可以自由访问，开发者们可以借此非常轻松地创建自己的工具包。此外，国内外的众多开发者以及虚幻官方商店也提供了许多开发者们已经完成的插件和工具包让我们学习下载。

正是由于虚幻引擎的无限拓展性，才能够在它上完成许多本不属于“游戏引擎”范畴的功能。

2.2 虚幻蓝图

蓝图（BluePrint）是虚幻引擎研发的一款编程脚本系统，它的主要理念是，在引擎蓝图面板中，可以通过拖拽一些图形化的函数和变量来搭建完整的游戏执行流程框架。就像其他的游戏脚本，蓝图的用法也是通过定义在游戏世界中的模型对象或者类来为游戏对象增加脚本从而控制游戏行为。[10]在我们开发过程中，即使对于一个经验老到的 C++程序员，也不可避免会用到由蓝图构造的对象，在引擎中，这类对象也会被直接成为”BluePirnt”。

蓝图系统非常灵活，因为正如上面所提到的，它为编程人员提供了代码中所需要的所有概念，如类，变量，函数，结构体，流程判断（循环，序列），并且在虚幻引擎中 C++与蓝图也被很好的结合起来，这一点我会在虚幻引擎 C++与蓝图中详细介绍。这样与 C++的兼容功能，可以使开发人员更加高效地使用蓝图填补一些无需使用编程就可以完成的游戏框架搭建或者是其他游戏行为。

蓝图的形式是通过在不同的节点（代表变量，函数）来完成游戏流程的控制。

蓝图通过各种样式的节点来代表不同的概念，同时在编辑器的属性设置面板轻松更改蓝图节点的属性（如更改变量类型，增加函数输入，输出）

2.2.1 关卡蓝图

对于游戏中的每个关卡，都有一个关卡蓝图、它的作用是控制当前关卡中的对象，控制游戏流程，也可以用来设置过场动画，并可以管理一些像开场引导，地形管理以及其他关卡相关的功能。关卡蓝图就像是 C++中的类一样，它一样可以调用别的“类”的变量和功能，具体表现为它还可以与关卡中的其他蓝图（主要是蓝图类）进行交互，譬如监视场景中蓝图类的行为变化（蓝图类的变量改变）从而触发蓝图类中的功能事件。

2.2.2 蓝图类

蓝图类非常适合处理游戏中的交互式模型，比如游戏场景中常见的门，灯光，可操作的物体以及可更改的景观，如门的蓝图类中可以包含空间重叠时间，门开关动画，开关音频，更改材质，与玩家交互等功能。

但是蓝图类中包含的事件既可以有蓝图类本身出发，也可以放在关卡蓝图中由关卡中其他流程事件来触发。设置好蓝图类后，我们只需把它放入游戏场景，就像编程中的“实例

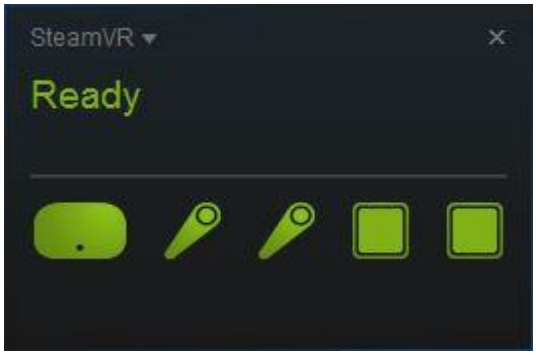
化”，我们就可以自由使用蓝图类中的功能了，并且在蓝图类中的更改会影响每一个场景中的实例对象。

2.3 虚幻引擎与 SteamVR

在虚幻引擎 4.8 版本之后，虚幻引擎已经支持 SteamVR，可以实现真正的即插即用，并且充分利用了 VR，房间，输入，灯光以及 SteamVR 的最新的激光跟踪解决方案。SteamVR 通过蓝图可视化脚本和本机代码完全继承到虚幻引擎中，因此可以在不需要依赖程序员支持的情况下构建项目。[11]

安装好 SteamVR 后，我们将在电脑屏幕右下方看到下图

图 5 SteamVR 运行状态



这表明 SteamVR 已经正常启用，并且手柄和头盔都被成功检测。

要在虚幻引擎中使用 SteamVR，还需要启用 SteamVR 插件，该功能可以在虚幻引擎的 Plugin 中找到。做完上述步骤，我们就可以在游戏播放按钮下拉菜单中启用 VRperview 了，这意味着我们可以在 VR 模式下在游戏场景中移动。

第二步就是通过手柄在引擎中交互了。首先我们要设置一个新的 VR 蓝图角色。这将代替我们游戏中的默认角色，之后再蓝图中设置一个摄像机，在摄像机的右边属性栏可以将之连接到我们的头盔，接下来我们在人物组建中添加 Motioncontroller（即我们的手柄），然后在每个手柄下添加一个模型，这样手柄的移动转向都可以在游戏中展示出来了。

完成上述步骤，VR 人物就基本设置好了，如果要启用 VR 手柄交互功能，我们只需在 VR 人物的蓝图事件面板找到运动控制器（这是虚幻引擎中 SteamVR 手柄的名字）对应的输入就可以了。

在本次项目中主要使用了手柄的 Trigger 输入。

2.4 蓝图与 C++相互调用

UE4 为开发者们提供两种工具包，能够大大提高程序员的开发效率。新的游戏类，世界场景中的蓝图类中的组件可以为 C++类，一些涉及到逻辑运算，数学处理的比较庞大的模块可以用 C++语言来写，并且能够直接在虚幻引擎中编译我们写好的 C++代码。而蓝图可视化脚本如概述中介绍的：可以在功能模块之间进行连线以及变量和属性设置在编辑器中记性创建。同时蓝图类可以作为 C ++的子类来使用，这种情况下，开发者可以借助 C++先完成基础游戏类的创建，然后用可视化程度较高的蓝图来处理关卡设计和一些简单的逻辑处理，以及一些直接与游戏场景相关联的功能。

C++和蓝图是相互配合的，不论创建项目时选择的是 C++项目还是蓝图项目，都可以同时使用蓝图和 C++分别处理不同的模块。

虚幻蓝图和 C++相互调用一共有以下三种方法：

2.4.1 蓝图调用 C++变量函数或函数

首先在 C++类的 Public 作用域中声明函数或变量，然后在声明之前添加虚幻引擎特有的属性声明。一般的，变量属性声明使用 UPROPERTY，而函数使用 UFUNCTION。在括号中可以添加枚举值表明我们想设置的属性。本项目主要使用了：BluePrintcallable（蓝图可调用），Category（蓝图分类目录），EditAnywhere(可任意编辑)。再次编译 C++代码后即可在调用该 C++的蓝图类中找到在 C++中声明的变量或函数。

2.4.2 C++调用蓝图函数

首先创建一个 C++游戏类，在该类的声明中添加 UCLASS 属性：BluePrintable（可被蓝图化），然后就可以在编辑器的内容浏览器中找到该 C++类，基于该 C++类创建一个蓝图类。

在该蓝图中创建一个函数（在 Function 下拉栏中），接着在该父类 CPP 文件中使用函数 CallfunctionByNameWithArguments()，我们通过字符串将蓝图中刚刚创建的函数传递给 C++，并且以空格形式隔开，在后面紧跟参数，即可完成在 C++中对指定蓝图函数的调用。

2.4.3 C++中调用蓝图事件

在虚幻蓝图类中，有一个节点叫做自定义事件，它的作用和函数类似，都是执行一系列自定义行为。但是事件和函数不同的地方在于没有输入参数，也没有输出值。自定义事件更类似于一个火车头，作为某个功能模块的驱动事件。

首先在我们 C++基类中做一个广播,声明变量类型，本项目使用了：DECLARE_DYNAMIC_MUTICAST_DELEGATE,用定义的变量类型定义一个广播变量,即可在 C++中调用该广播变量的成员函数（本项目中调用了广播函数），这个行为指定了该事件在何时被执行，接着可以在蓝图类的 Event Dispatchers 分类中找到该事件，我们可以在蓝图中填充该事件的具体内容，这样就实现了蓝图和 C++的相互调用。

一般来说，蓝图效率比 C++要低一半以上。所以在应用到比较复杂的模块处理时，程序员往往使用 C++，本项目中蓝图只提供了获得数据和展示数据的功能，占总工程量的 20%。

2.5 虚幻引擎的渲染

虚幻引擎 4 拥有全新的、DirectX 11 管线的实时渲染系统，包括延迟着色，全局光照，半透明光照，后处理以及使用矢量场的 GPU 粒子模拟。

虚幻引擎 4 中所采用的光照为延迟光照，这点与前几代的引擎有所不同，虚幻引擎 3 采用的是前置光照。材质将他们的属性写到后台缓存中，然后在游戏运行执行光照时再读取材质的属性，根据一定数学运算对像素进行光照处理。

虚幻引擎一共有三条光照路径：可移动光源（适用于完全动态），固定光源（适用于部分静态），静态光源（适用于完全静态），这几个不同的工具在光照品质，耗费性能和游戏中的可变化性和实时性有多样的选择，开发者可根据游戏所需特性来选择最适合的光照。

可移动光源：可生成随游戏进程变化的光源，并且能随时改变包括光源的位置，旋转角度，自发光颜色，亮度，强度衰减参数，光照范围在内的所有属性，可移动光源的属性基本都可以在游戏中变化，并且光照贴图中不会存储可移动光源产生的光照效果，也不会产生间接光照结果（由被光照的模型反射到别的物体上产生的光照效果）。可移动光源使用全场景动态的方式投射阴影，性能开销巨大。性能消耗的程度主要取决于接收可移动光

源光照的模型的数量，以及这些模型的复杂程度（具体表现为模型接收光照的面的数量）。也就是说一个同等大小和衰减半径的光源可能因为所照射的物体模型复杂度的不同而产生完全不同的

固定光源：光源位置在游戏过程中保持静止，不过剩余属性如颜色和亮暗均可以被修改。固定光源与可移动光源最大的区别在于静态光源的任何属性都不能被更改。并且，值得一提的是，在运行时对固定光源的亮暗进行改变仅仅会改变直接光照产生的结果。UE4的间接光照（由反射产生）是根据 lightmass 产生的，是保持不变的。

在以上三种光源中，固定光源质量较高，性能消耗居中，可变性也居中。

静态光源是指在运行时不会随游戏进行或者用户操作而产生改变或移动的光源。与固定光源不同，他们的所有光照都会在光照贴图中完成（因为没有间接光照），在光照效果计算完成之后，便不会再对游戏场景中的其他事物造成影响。因此，一些在游戏进程中改变或者移动的物体，它们因为静态光源所产生的效果不会发生变化，这样静态光源作用的范围就非常有限了。

在三种不同的光源可移动性属性中，静态光源具有中等品质，最差的可变性，最低的设备性能损耗。

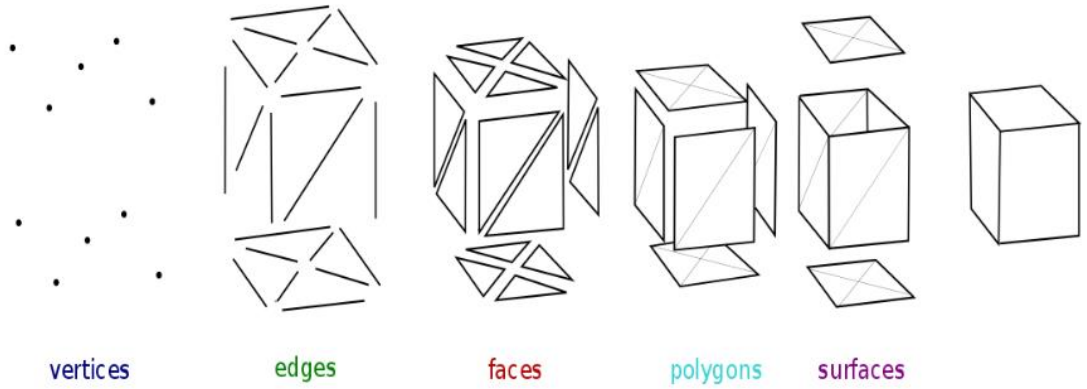
因为静态光源产生的光照效果仅使用光照贴图，故得到的阴影效果在游戏运行前已经被确定。这也意味着它们不能在游戏运行时产生除了已经完成的阴影之外的效果。正如我们在示例中所看到的。但是我们可以通拿过提高他们的光照贴图分辨率，来获得更佳的阴影效果。静态光源主要出现在低消耗的移动平台中。

本项目采用的光源为固定光源，因为我们实时改变的三位模型会产生新的顶点和平面，这意味着我们要对新产生的平面和顶点计算光照，但是又不需要改变光源的位置，固定光源无疑是最好的选择。

第三章 Polygon Mesh

多边形网格是顶点，边和面的集合，用于在 3D 计算机图形和实体建模中定义多面体对象的形状。面通常由三角形（三角形网格），四边形或其他简单的凸多边形组成，因为这简化了渲染，但也可能由多个凹多边形或带孔的多边形组成。[12]

图 6 PloygonMesh 示意图



使用多边形网格创建的对象必须存储不同类型的元素。这些包括顶点，边，面，多边形和曲面。 在许多应用中，只存储顶点，边和任意面或多边形。 渲染器可能只支持三角形面，所以多边形必须由多个三角形构成，如上所示。 然而，许多渲染器或许支持四边形和高边多边形，或者能够将多边形动态地转换为三角形，从而无需以三角形形式存储网格。 此外，在某些应用程序中，如头部建模，最好能够创建 3 边和 4 边多边形。

多边形网格可以用各种方式表示，使用不同的方法来存储顶点，边和面数据。这些包括:

面顶点网格

一个简单的顶点列表，以及一组指向它使用的顶点的多边形。

翼边

其中每个边指向两个顶点，两个面以及与它们接触的四个（顺时针和逆时针）边。翼边网格允许恒定的时间遍历曲面，但是具有更高的存储需求。

半边网格

类似于有边网格，除了只使用边缘遍历信息的一半，本次项目所采用的即半边网格，下一节会详细介绍。

四边形网格

其存储边缘，半边缘和顶点，而不涉及多边形。多边形隐含在表示中，并且可以通过遍历结构来找到。内存要求与半边网格类似。

角桌

将顶点存储在预定义的表格中，以便遍历表格隐式定义多边形。这实质上是硬件图形渲染中使用的三角形风扇。该表示更紧凑，更有效地检索多边形，但更改多边形的操作很慢。此外，角落表格并不完全代表网格。需要多个转角表（三角形风扇）来表示大多数网

格。

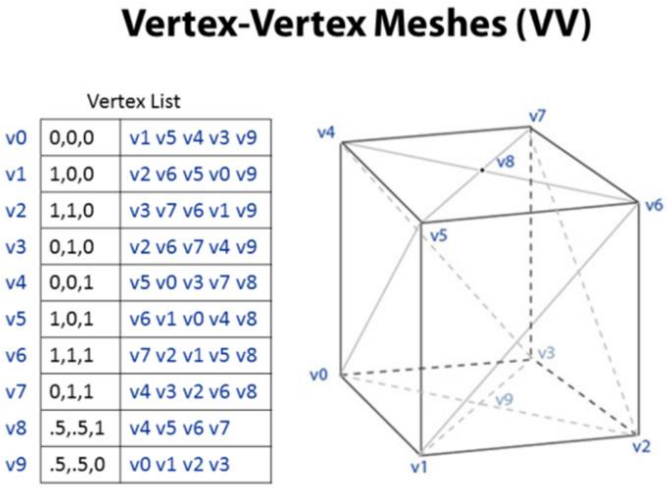
3.1 顶点网格

“VV”网格仅表示指向其他顶点的顶点。表示中隐含着边缘和面部信息。但是，表示的简单性不允许在网格上执行许多有效的操作。

上面的每一个表述都有特别的优点和缺点，但本项目主要涉及到网格的形变，顶点的增加，所以使用了半边结构。

数据结构的选择取决于应用程序，所需的性能，数据的大小以及要执行的操作。例如，处理三角形比一般多边形更容易处理，特别是在计算几何中。对于某些操作，有必要快速访问拓扑信息，如边缘或邻近面;这需要更复杂的结构，例如边缘表示。对于硬件渲染而言，需要紧凑而简单的结构;因此角落表（三角扇）通常被整合到低级渲染 API 中，例如 DirectX 和 OpenGL。

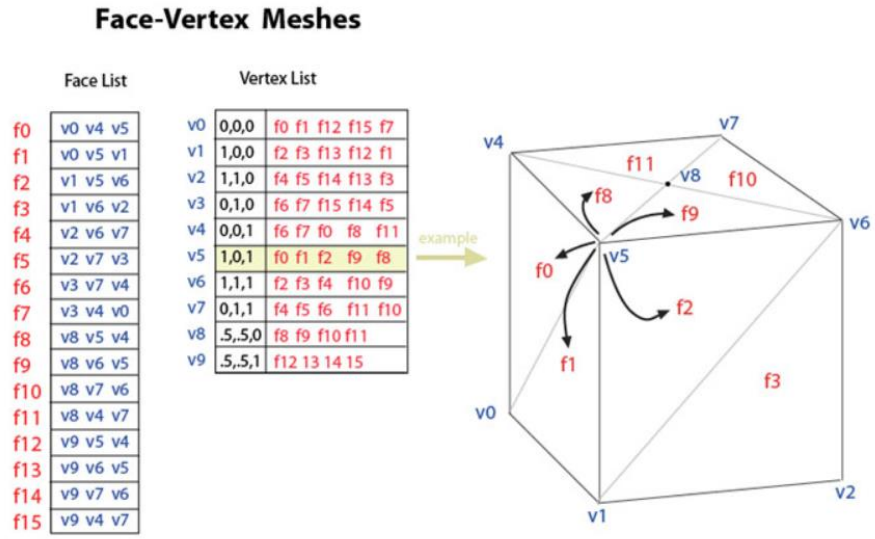
图 7 顶点网格示意图



顶点 - 顶点网格将对象表示为连接到其他顶点的一组顶点。 这是最简单的表示法，但由于面部和边缘信息是隐含的，因此没有广泛使用。 因此，有必要遍历数据以生成用于渲染的面的列表。 另外，边缘和面部的操作不容易完成。

然而，VV 网格受益于小的存储空间和有效的形状变形。 上图显示了一个由 VV 网格表示的四边框。 每个顶点索引它的相邻顶点。 请注意，“方块 - 圆柱体”顶部和底部中心的最后两个顶点 8 和 9 具有四个连接的顶点而不是五个顶点。 一般系统必须能够处理连接到任何给定顶点的任意数量的顶点。[13]

图 8 面顶点网格示意图



3.2 面顶点网格

面顶点网格将一个对象表示为一组面和一组顶点。这是最广泛使用的网格表示，是现代图形硬件通常接受的输入格式。

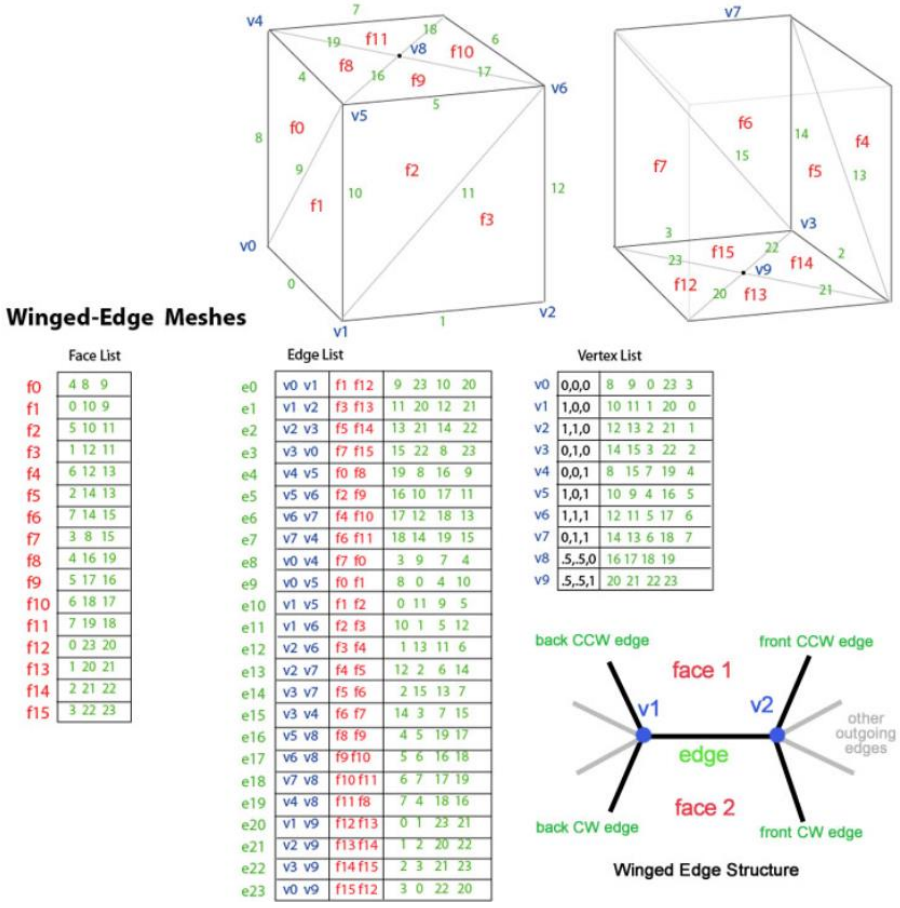
面对顶点网格改进 VV 网格进行建模，因为它允许显式查找面的顶点以及围绕顶点的面。上图显示了作为 FV 网格的“箱形”示例。突出显示顶点 v5 以显示围绕它的面。注意，在这个例子中，每个面必须有 3 个顶点。但是，这并不意味着每个顶点都具有相同数量的周围面。

对于渲染，通常将面部列表作为顶点的一组索引传送给 GPU，并将顶点作为位置/颜色/普通结构发送（图中仅给出了位置）。这样做的好处是可以通过重新发送顶点数据而不更新面连通性来动态更新形态变化（不涉及顶点的增加或者减少）。

建模需要轻松遍历所有结构。使用面顶点网格很容易找到围绕面的顶点。此外，顶点列表包含连接到每个顶点的面的列表。与 VV 网格不同，面和顶点都是明确的，因此定位相邻面和顶点的时间是恒定的。但是，边缘是隐含的，因此仍然需要搜索来查找给定面部周围的所有面部。其他动态操作（如分割或合并面）对于面顶点网格也很困难。

3.3 翼边网格

图 9 翼边网格示意图



由 BaulgART 1975 引入，翼边网格明确地表示网格的顶点、面和边。这种表示被广泛地应用于建模程序中，以在动态改变网格几何结构方面提供最大的灵活性，因为分割和合并操作可以很快地完成。它们的主要缺点是大的存储需求和由于维护许多索引而增加的复杂性。在图 GEMS GEMS II 中可以找到关于翼缘网格的实现问题的很好的讨论。

翼边网格解决了从边到边遍历的问题，并提供了围绕边缘的有序的一组人脸。对于任何给定的边缘，出射边缘的数目可以是任意的。为了简化这一点，翼缘网格只提供四个，在每个末端最近的顺时针和逆时针边缘。其他边可以增量遍历。因此，每个边缘的信息类似于蝴蝶，因此“翼缘”网格。上面的图显示了“盒子圆柱”作为翼缘网格。边缘的总数据由 2 个顶点（端点）、2 个面（每个边）和 4 个边（翼边）组成。[14]

绘制图形硬件的翼边网格需要生成面部索引列表。这通常只在几何变化时进行。翼缘网格理想地适合于动态几何，如细分曲面和交互式造型，因为网格的变化可以局部发生。可以有效地完成穿越网格，如碰撞检测所需的遍历。

第四章 HalfEdge DataStructure

4.1 数据结构简介

表示多边形网格的一种常见方法是共享顶点列表和存储顶点的指针的列表。这种表示对于许多目的都是方便和有效的，但是在某些领域中证明是无效的。

例如，网格简化通常需要将边缘折叠成单个顶点。该操作需要删除边缘边缘的面部并更新在边缘的端点共享顶点的面部。这种类型的多边形“手术”要求我们发现网格的组成部分之间的相邻关系，例如面部和顶点。虽然我们当然可以在上面提到的简单网格表示上实现这些操作，但它们会花费开发者过多的时间；许多结果将需要我们搜索整个面部或顶点列表，或者甚至可能两者都需要。[16]

多边形网格上的其他类型的邻接查询包括：

哪些面用到了这个顶点

哪些边用到了这个顶点

哪些面和这个边邻接

哪些边组成了这个面

哪些面和这个面邻接

为了有效地实现这些类型的邻接查询，已经开发了更复杂的边界表示（B-ReP），其明确地对网格的顶点、边和面进行建模，并在其内部存储附加的邻接信息。

这些类型的表示中最常见的一种是翼缘数据结构，其中边缘用指针指向它们接触的两个顶点，两个面邻接它们，以及指向从端点发出的四个边缘的指针。这种结构允许我们在恒定的时间内确定哪一个面或顶点与边缘相交，但是其他类型的查询可能需要更小号时间的处理。

半边数据结构是一个稍微复杂的 B-ReP，它允许在上面的所有查询（以及其他）在恒定的时间（*）中执行。此外，即使我们在面、顶点和边中包含邻接信息，它们的大小仍然是固定的（不使用动态数组）以及数据具有合理的紧凑。

这些性质使得半边缘数据结构对于许多应用来说是一个极好的选择，然而它只能代表歧管表面，在某些情况下，它被证明是无效的。歧管在数学上被定义为表面，每个点被一个具有圆盘拓扑的小区域包围。对于多边形网格，这意味着每个边都由两个完全的面接界，不允许 T 结点、内部多边形和网格中的断点。

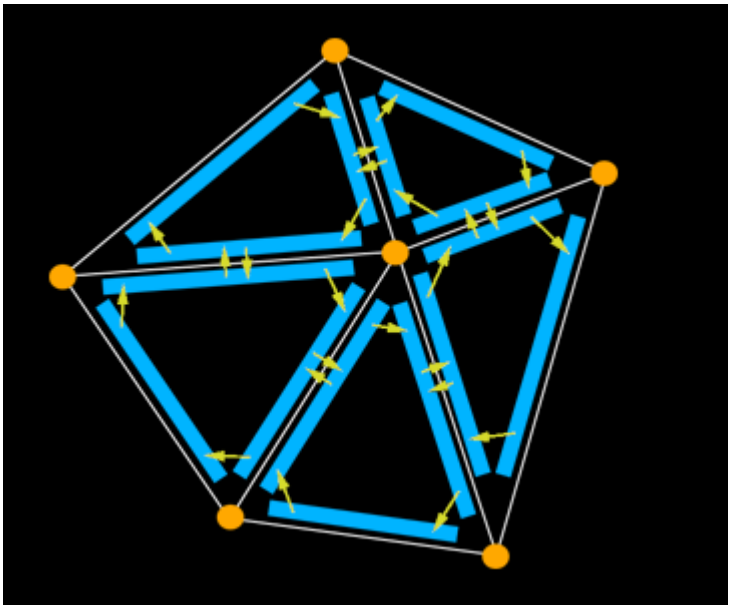
更准确地说，每条信息收集的时间是固定的。例如，当查询与顶点相邻的所有边缘时，该操作将在与顶点相邻的边缘的数量上是线性的，但查询每个边缘的时间是恒定的。

4.2 数据结构表示

因为不是存储网格的边缘，所以我们存储半边，结构也被称为半边结构。顾名思义，半边是边的一半，并通过沿其边缘分裂边缘来构造。我们把两个半边称为一对边。半边是有向的，一对的两个边有相反的方向。

下面的图显示了三角形网格的半边表示的一小部分。黄色点是网格的顶点，而浅蓝色的条是半边。图中的箭头表示指针，尽管为了防止图表变得凌乱，有些图已被忽略。

图 10 半边网格示意图



正如上图所示，边框边的半边在其周界形成一个圆形链表。这个列表既可以顺时针或逆时针方向围绕脸部，只要使用相同的规则。循环中的每一个边沿都存储指向其边界的指针（图中未示出）、顶点在其端点（也未示出）和指向其对的指针。它可能在 C 语言中看起来像这样：

```
struct HE_edge
{
    HE_vert* vert;     // vertex at the end of the half-edge
    HE_edge* pair;     // oppositely oriented adjacent half-edge
    HE_face* face;     // face the half-edge borders
```

```
    HE_edge* next;     // next half-edge around the face
};
```

半边数据结构中的顶点存储它们的 x、y 和 z 位置以及指向顶点的一个半边的指针，该顶点使用顶点作为起始点。在任何给定的顶点，我们可以选择一个以上的一半边，但是我们只需要一个。在 C 中，顶点结构看起来是这样的：

```
struct HE_vert
{
    float x;
    float y;
    float z;
    HE_edge* edge;   // one of the half-edges emantating from the vertex
};
```

对于半边数据结构的一个简单的版本，一个面只需要存储一个指针到一个半边沿。在一个更实际的实现中，我们可能也会在面结构中存储纹理、法线等信息。面中的半边指针类似于顶点结构中的指针，尽管每个面都有多个半边，但我们只需要存储其中一个，不必考虑哪一个。（因为半边之间有相邻关系，我们很容易由一个半边找到其余所有的半边）下面是 C 中的面部结构：

```
struct HE_face
{
    HE_edge* edge;   // one of the half-edges bordering the face
};
```

4.3 邻接查询

大多数邻接查询的答案直接存储在边缘、顶点和面的数据结构中。例如，边框或顶点边框可以很容易地找到这样的边沿：

```
HE_vert* vert1 = edge->vert;
HE_vert* vert2 = edge->pair->vert;
HE_face* face1 = edge->face;
HE_face* face2 = edge->pair->face;
```

一个稍微复杂的例子是在与面相邻的半边上迭代。因为面周围的半边形成了一个循环链表，并且面结构存储了指向这些半边中的一个的指针，所以我们这样做：

```
HE_edge* edge = face->edge;
    do {
        // do something with edge
        edge = edge->next;
    } while (edge != face->edge);
```

类似地，我们可能对迭代在与某个顶点相邻的边或面感兴趣。回头看一下图表，可以看到除了在面的边界周围的循环链表之外，指针还围绕顶点形成循环。迭代过程对于发现相邻的边或顶点到顶点是相同的；这里是在 C 语言中的实现：

```
HE_edge* edge = vert->edge;
    do {
        // do something with edge, edge->pair or edge->face
        edge = edge->pair->next;
    } while (edge != vert->edge);
```

注意，在这些迭代示例中不包含空指针的检查。这是因为对曲面的限制是多方面的，为了满足这一要求，所有的指针都必须是有效的。

通过这些例子可以快速找到其他的邻接关系。

在本项目中使用了经过拓展的 HalfEdge Data Structure，功能更加完整，具体将在下一节做出介绍。

4.4 本项目使用的 HalfEdge 库

顶点结构：增加了 HalfVertices 概念，主要用于存储平面和对应的法线。（具体原因在算法实现流程中介绍）

边结构：传统的 HalfEdge Data Structure 中是没有 Edge 结构，本项目引入该结构是为了管理一对半边，当增加平面或者删除平面时，对应平面的半边消失，但是由于边结构是由两个顶点来确定的。我们依然可以对这两个顶点生成新的半边，否则对模型更改后悔导致原有的数据丢失。[17]

面结构：增加了面法线，便于模型生成时光照计算。

半边结构：除了增加了对应以上结构的指针外，还增加了源顶点和目标顶点指针，这两个指针可以帮助从半边结构到顶点的双向寻找。

HalfEdgeIterator：本项目采用了多个迭代器，包括对整个模型的边，半边，顶点，平面进行迭代，包含了上一节中提到的传统存储网格的方式，我们可以在整个模型中寻找符合条件的属性，再通过半边结构对其周围进行处理。以及面-顶点，面-半边，点-半边迭代器，是对传统半边结构迭代的一个优化，通常用户需要对半边结构足够了解，通过顶点，半边，面的转换才可完成对周围属性的搜索，迭代器将这些操作封装起来，我们只需要使用迭代器即可完成一系列复杂的结构寻找。

Edgeclass：本项目使用的边预处理结构，用于存储即将被处理的边的信息，包含了长度属性。

Vertexclass：本项目使用的顶点预处理，用于 FBX 与 HalfEdge Data Structure 结构转换。

第五章 虚幻引擎三位建模实现流程

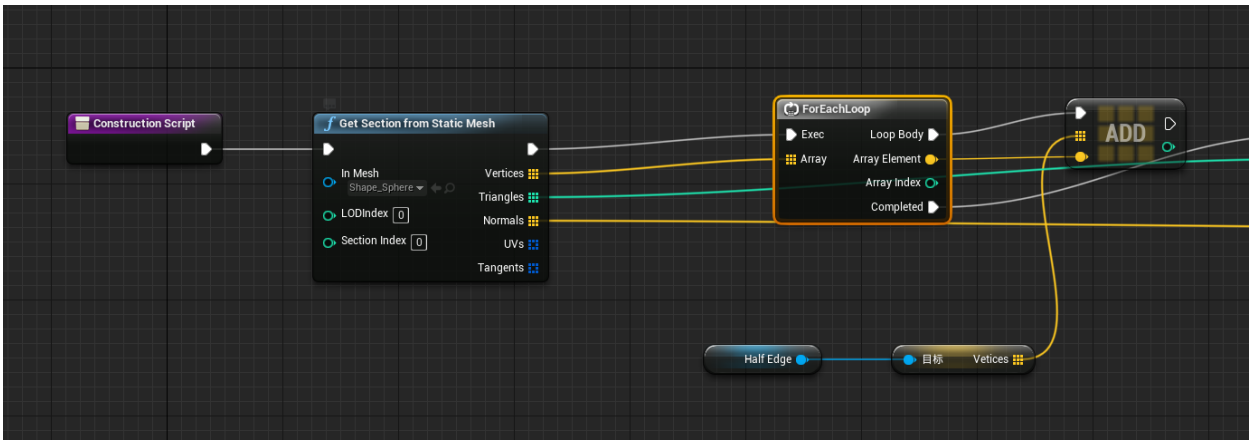
图 11 三维模型交互流程图

5.1 导入模型数据（蓝图完成）

通过蓝图节点 GetSectionfromStaticMesh 从资源模型中获取模型数据，我主要获取了模型的顶点坐标，顶点拓扑关系（这决定了顶点是按照什么顺序来组成模型的表面），顶点法向量（这可以决定我们模型每个面的光照强度）。[18]

按照在虚幻引擎蓝图与 C++交互一节中提到的方法，我在一个蓝图类中添加一个 C++组件，（在我的项目中我命名为 Halfedge），并且在 C++类中设置共有变量，在变量声明之前添加虚幻引擎特有的声明：Uproperty（BlueprintReadWrite），这将使得我的变量可以在蓝图类中可见并且可编辑。通过这样设置，就可以完成蓝图类和 C++类中的数据交换。

图 12 模型数据导入



5.2 模型数据格式转换

根据之前提到的 HalfEdge 结构的优点，我需要把从蓝图中得到的模型数据（由数组来存储）转换为 Halfedge 数据结构的形式。

首先要实现的就是数据类型的转换，Halfedge 中实现存储主要是通过标准空间下的容器以及 OpenGL 图形库的 glm 数学库，而虚幻引擎有自定义容器。我把每个顶点的坐标单独拿出来然后存入一个 glm::vec3 类型的变量。同样地，可以实现顶点拓扑结构，顶点法线结构的数据类型转换。

5.3 数据结构转换

由于虚幻中模型资源只能通过 FBX 格式导入。FBX 模型格式的一个特点是每个物理位置有多个顶点，其中的每个顶点对应着不同的面，以及这个面的法向量。这意味着一个四面体有 12 个顶点，对应着 12 个法向量。但是在半边结构中我们在一个物理位置上只需要一个顶点，面的索引和法线将被作为 HalfEdge 结构的属性存储。

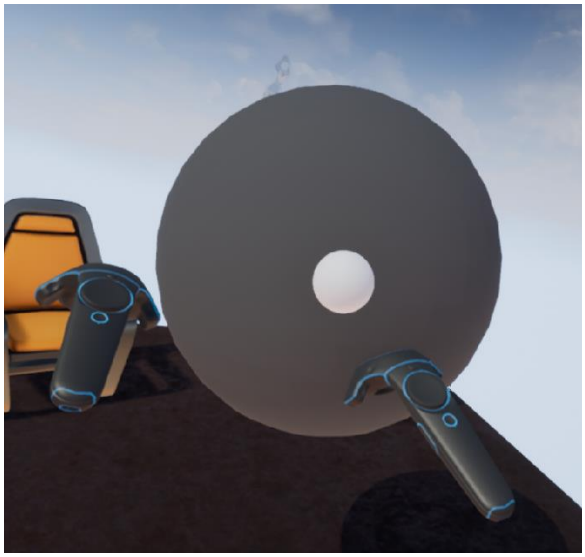
我的顶点转换流程是：依次检查从蓝图中获得的顶点，通过判断位置信息，只保留该位置上出现的第一个顶点，同时更改顶点拓扑中的顶点索引：在检查到该位置上的第一个顶点时，存入我创建的过渡顶点结构，在检查到第二个及以后的重复顶点时，对顶点索引数组不做处理，但将顶点拓扑数组中所有用到该重复顶点的位置替换为第一个顶点的下标，并且保存该面的索引和重复顶点的法线，作为过渡顶点的属性存储起来。

通过上述操作，我就得到了一个过渡顶点数据结构，接下来，我们再将数据从过渡顶

点转换为 HalfEdge Structure。这个过程相对比上一步要简单，只需按照 HalfEdge 库提供的接口来添加顶点，然后每三个顶点一组，按照修改后的顶点拓扑数组来添加 HalfEdge 中的面，接着添加顶点属性中的法线以及法线所在的面索引。

在本项目完成过程中，由 FBX 到 HalfEdge 转换与 HalfEdge 到 FBX 转换是首先完成的，这样可以检测是否可以将 HalfEdge 结构用来存储虚幻引擎中顶点。但为了逻辑顺序明确，从 HalfEdge 结构转换到 FBX 中将放在最后一小节说明。

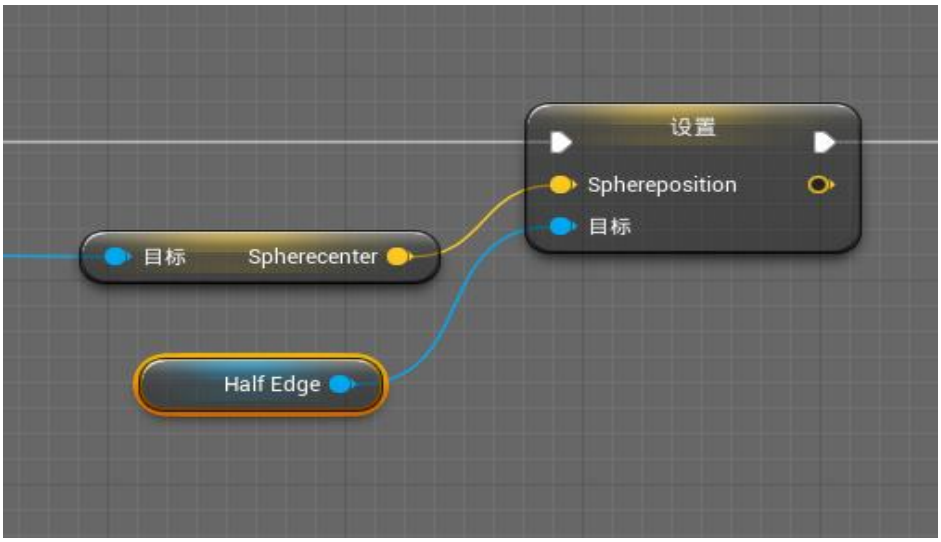
图 13 半边数据结构模型显示



5.4 存储顶点并移动

我们想要完成的三维模型交互是通过一个小球来实现的，当小球与模型表面顶点足够近时（这个距离是由我来定义的），我们把圆球范围内的顶点索引都存储起来方便以后操作。接下来是处理顶点法线的问题，在第二节我们提到了 FBX 模型格式的顶点对应每个不同的面有不同的法线，虽然经过处理，但这些法线依旧被当成属性存储在顶点结构中。但是我们在移动顶点时必须指定顶点的移动方向，我们选择移动方向，这意味着我们需要把顶点的多个法线合并成一个。我这里只是将顶点的每个法线按照向量相加，然后归一化处理。[19]

图 14 获取手柄位置



这样就完成了三维模型交互的第一步：顶点移动，但只是沿着它的法线方向移动一小段距离，生成的新模型不具有可拓展性（顶点，面的数量都没有发生变化）。

5.5 根据移动顶点生成新的顶点和平面

顶点移动会造成顶点附近的边变长，如果没有新增顶点的话，就会导致模型形状不能发生变化。所以我们需要在被改变顶点周围的边上生成新的顶点，这在 HalfEdge Structure 中需要新的半边，边，顶点来生成新的平面。当一条边大于某个阈值时，我们在这条边的中心点生成一个新的顶点，并且由这个顶点和原来的顶点生成新的平面，生成平面的时候，由于半边结构的限制，我们必须按照一定的拓扑顺序来生成，默认情况下我们取逆时针，这样生成的平面所包含的半边才能够和原来的半边进行匹配。

显然，当我们拉伸顶点的时候，不会只有一条边发生变化，甚至不会只有一个顶点发生变化，这就涉及到半边结构中数据的有效性和时效性。我们必须保证每次处理的顶点和边都是有效的，因为在上一段中我们删除了一些边和平面，这会对我们模型的结构造成影响。

在上一节中我存储了在圆球半径内的顶点，在这时派上了用场。由于半边结构的便利性，我可以拿到顶点周围的边，并且在保持没有重复的边的情况下把每条边存储起来（因为两个邻接顶点直接会有重复的边，而对于每条边我们只需要处理一次）

这时我创建了一个新的 Edge 结构来存储顶点周围的边，因为半边结构中的边是不包

含

长度属性的，我通过边两端顶点的距离得到边的长度，并且和边之前的属性一起存入我的新 Edge 结构中，然后将容器中的边按照长度属性来排序，这样就得到了我所有需要改变的边，并且是有序的。之所以要排序，是因为在模型改变过程中，我们优先给较长的边增加新顶点和面。这样不会造成顶点过于拥挤，面过小而造成形状上的畸形。

这样，我们就可以实现在被拉伸的边上生成新的边和平面，保证了我们在拉伸一个模型之后，形状会随之发生变化，保证了模型的可拓展性。

最后还需要注意的是清除上面用到的顶点结构和边结构，这样可以确保每次按下手柄扳机的时候都可以重新规划顶点以便做出处理。

5.6将修改后数据显示

在第一节和第二节中我们把数据从蓝图的模型资源中转换到了 C++HalfEdge 结构中。现在我们需要把数据逆向转换，从而显示在游戏世界中。

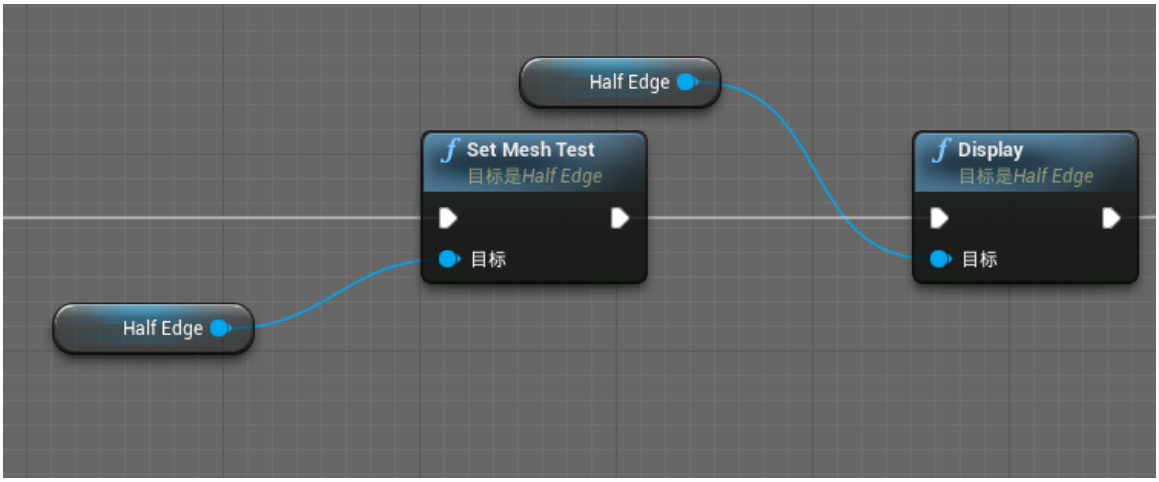
首先我们需要在 C++一些函数前做出一些声明，就像我们在第一节中对顶点数组变量做的那样。[20]

UFUNCTION(BlueprintCallable, Category = "VRinteracting")

如上所示，UFUNCTION 表明我们是对函数属性做出规定，第一个参数是规定了该函数可以在对应的蓝图类中被调用，第二个参数表明在蓝图中该函数将出现在的分类(这里我自定义了一个分类 VRinteracing，代表 VR 交互类功能)

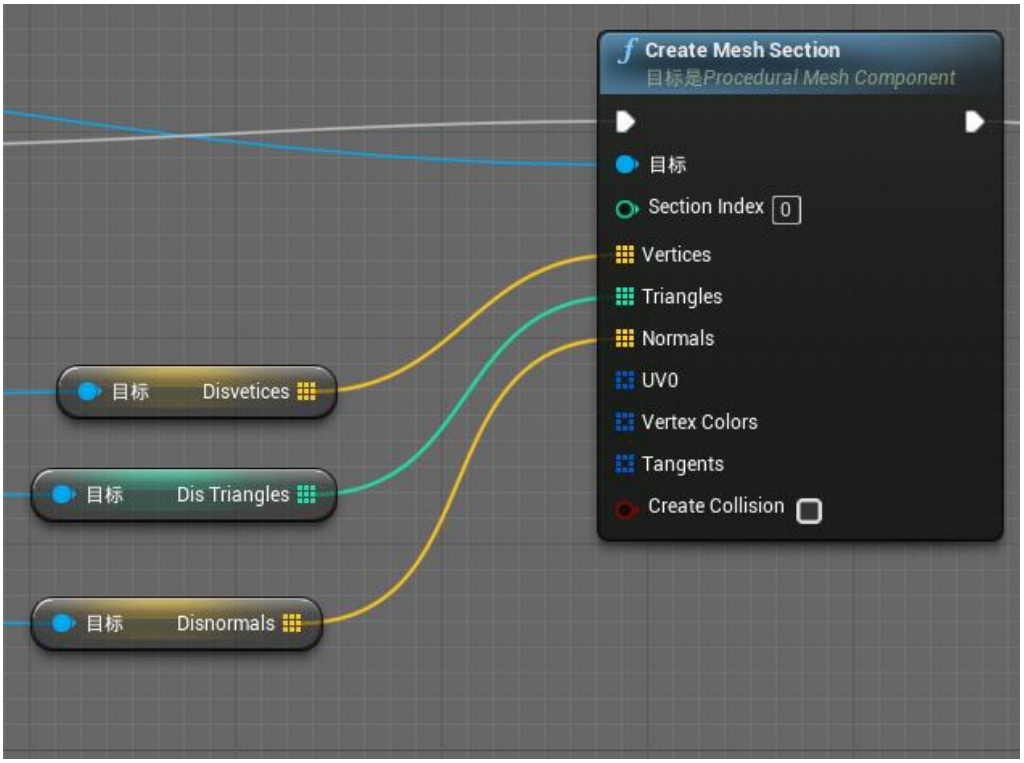
修改后的数据要恢复之前的每个顶点对应每个面的法线，简单来说即将四面体的四个顶点恢复为之前的十二个顶点。如果要按照顶点数组来拆分这显然是行不通的，因为每个顶点对应之前的哪一个面虽然被记录下来了，但之前的顶点数组信息已经被修改了，在增加过新的顶点和平面后，之前的数据已经相当于无效。所以我们需要从顶点拓扑数组入手。

图 15 蓝图调用调用 C++函数



每个三角形面是由三个顶点组成的，并且每个顶点存储着对应该平面的顶点。我们只需将每个平面都输出，将顶点和法线从原来的顶点结构中拆分成独立的数据（但是他们的下标还是一一对应的关系），再输出到用来展示的数组中。

图 16 蓝图生成模型



得到可以在蓝图中调用生成模型的顶点数组，顶点拓扑数组，法线数组后，我们需要用到一个蓝图的新功能：ProceduralMesh。它的意思是可编程的 Mesh，我们通过给该节点提供模型所需要的数据信息（我们在上一段得到的数组），它就会根据这些数据生成一个在世界中的模型物体。

至于我们每次的更新操作，对顶点拉伸造成形变必须在每帧都实时更新到世界场景

中，我在每帧中将用于展示的数据数组清空，然后导入更新后的数据。再用之来生成新的ProceduralMesh。

至此，我实现了在虚幻引擎中 VR 环境下基于 HalfEdge 结构的三维模型交互。

图 17 球体单次拉伸(一次顶点变形)

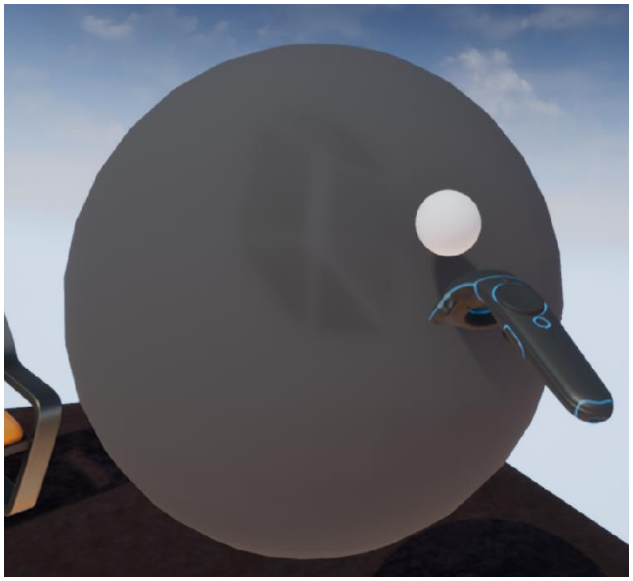


图 18 手柄多次拉伸（多次顶点变形）

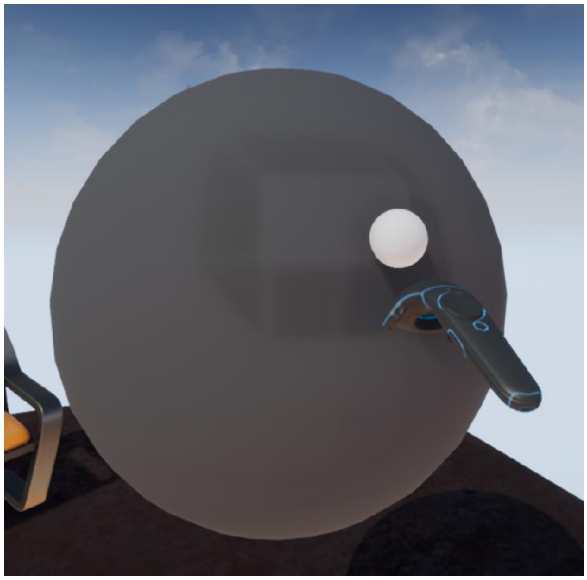


图 19 兔子模型（交互前）

图 20 兔子模型交互后（对耳朵进行拉伸）

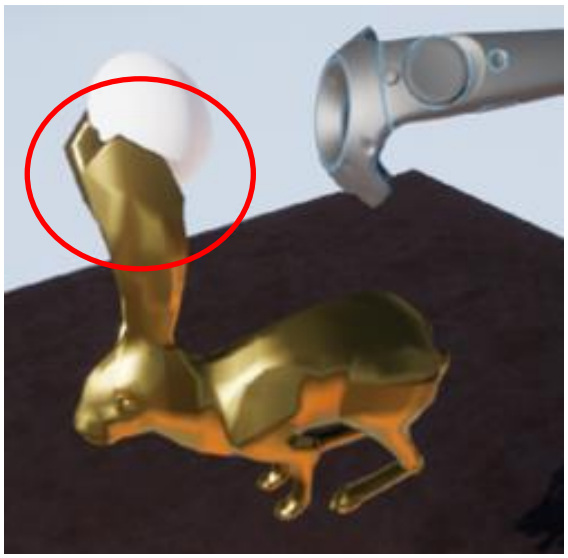


图 21 兔子模型（交互前）

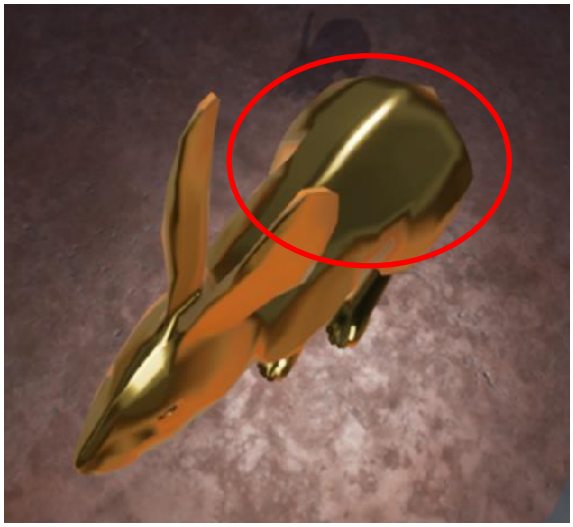
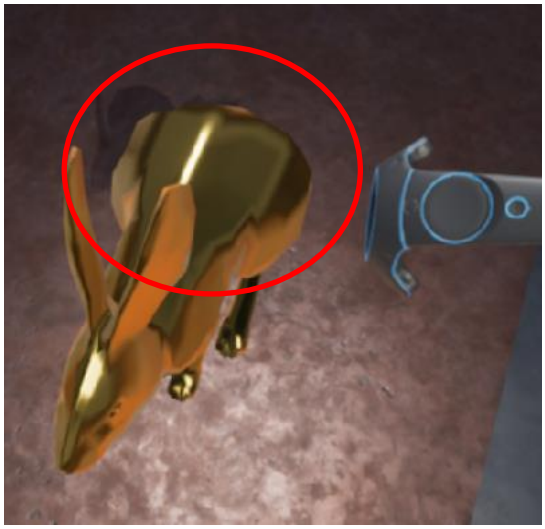


图 22 兔子模型（对腿部进行拉伸）



致谢

时光飞逝，转眼间四年紧张而又充实的大学生生活即将画上句号。在这四年的学习期间，我得到了很多老师、同学和朋友的关怀和帮忙。在学位论文即将完成之际，我要向所有期间给予我支持、帮忙和鼓励的人表示我最诚挚的谢意。

首先，我要感谢我的指导老师朱晓强老师对我的教导。从论文的选题、构思、撰写到最终的定稿，朱老师都给了我悉心的指导和热情的帮忙，使我的毕业论文能够顺利的完成。朱老师对工作的认真负责、对学术的钻研精神和严谨的学风，都是值得我终生学习的。从开始接到论文题目到系统的实现，再到论文文章的完成，每走一步对我来说都是新的尝试与挑战。最后，感谢我的家人在此期间给予我的包容、关爱和鼓励，以及所有陪我一路走来的同学和朋友，正是由于他们的支持和照顾，我才能安心学习，并顺利完成我的学业。

毕业在即，接下来的我会踏进另一片知识的海洋，探寻科学的真谛，希望这篇论文不会是学术思考的终点。

参考文献

[1] Bian F., Jiang M.Q.,Sang Y.Y. (2007). The development of virtual reality and its application, Computer Simulation, 24 (6), 1-4.

[2] Lu S.F., Jin X.G. (2014). Marbling overview of computer simulation technology, Journal of Computer Aided Design and Computer Graphics, 26 (2), 179-190. Yang B., Luo Z.Q., Xiong

[3] Cui Q.W., Sun W.L., Wan X.J., Wang C.X. (2012). Research on oil field visual simulation system based on virtual reality technology, Mechanical Design and Manufacture, 50 (9), 249-251.

[4] Jiao P.F., Zhang M.C., Li B.F. (2014). Development of 3D anatomical atlas based on Virtools virtual reality technology, Computer Engineering and Science, 36 (8), 1555-1559.

[5] J.L., Zhu Y.M. (2010). Research on interactive product virtual exhibition technology based on Virtools, Modern Manufacturing Engineering, 33 (5), 36-39.

[6] Kang W. (2010). Research on virtual scene technology based on resource space model, Micro Computer Information, 26 (6), 206-208.

[7] Zhu Z.X., Chen L., Li S.S. (2013). Key technologies of virtual assembly of combine harvester chassis based on virtual reality, Journal of Agricultural Machinery, 44 (s2), 262-267.

[8] Zhang K.X., Li P., Ma Q. (2012). Complementarity between VRML and 3DS MAX modeling, Computer System Application, 21 (2), 85-88.

[9] 虚幻引擎功能简介，https://www.unrealengine.com/zh-CN/features，2017.7.17

[10]虚幻引擎蓝图简介，http://api.unrealengine.com/CHN/Engine/Blueprints/GettingStarted/index.html，2017.7.17

[11]虚幻引擎 SteamVR 开发，http://api.unrealengine.com/latest/CHN/Platforms/SteamVR/index.html，2017.7.17

[12]PolygonMesh,https://en.wikipedia.org/wiki/Polygon_mesh,2018.3.17

[13]樊银亭,滕东兴,汪恭正,杨海燕,王宏安,戴国忠.虚拟家居定制系统中的自适应用户界面实现机制[J].计算机辅助设计与图形学学报,2011,23(04):705-712.

[14] L.X., Xie C.Y. (2014). 3D modeling and VRML network mining technology research and application based on virtual reality, Mining and Metallurgical Engineering, 34 (5), 19-22.

Zhang

[15]Zhu M., Yang R. (2012). Research on the software and hardware development of 3D visual simulation system, Aircraft Design, 33 (1), 39-42.

[16]Zou H., Yang H., Cui X.W., An X.M. (2014). Virtual simulation scene and interactive roaming system, Oil and Gas Field Surface Engineering, 37 (1), 22-23.

[17]刘剑锋. 虚拟环境下几种三维交互技术的研究[D].[硕士学位论文].浙江大学,2010,15-20

[18]蔺薛菲.虚拟现实三维场景建模与人机交互应用技术研究[J].艺术与设计(理论),2017,2(04):100-102.

[19]万华明,杨丽,邹湘军,罗陆锋,顾邦军,刘彪.基于 VR 技术的几何建模与优化[J].苏州科技学院学报(自然科学版),2009,26(04):20-24.

[20]吴桐桐,周国辉.基于虚拟现实的三维建模技术的研究[J].智能计算机与应用,2016,6(02):113-115.

附录 1 三维建模技术在虚拟现实环境中的应用

虚拟现实中的三维建模技术分析

VR 建模技术的特点

虚拟现实系统强调沉浸感，生动性以及即时互动。 实时和现实是主要评估计算机图形计算的标准。VR 建模的一个显着特点是采取生动性，实时性和互动性作为整个 VR 建模过程中的主要原则，这也是 VR 模型与传统的 CAD 和卡通建模方法的不同之处。首先，VR 建模涉及更多内容：除了对象的主要建模之外，它还涉及自由度，图层细节，这些内容贯穿整个系统。其次，三维建模主要以更容易的方式处理模型（例如：纹理，形状，颜色），而不是要提升虚拟对象的几何模型的复杂性来呈现一个更真实的对象。评估 VR 建模技术的技术参数包括便捷性，通用性，准确性，显示速度，即时显示和运行效率等。

VR 建模的主要内容

虚拟环境的建模是构建 VR 系统的基础。 虚拟建模环境主要包括 3D 视觉建模和 3D 音频建模。 三维可视化建模涵盖方面如几何建模，物理建模，运动建模，行为建模等。 其中，几何造型起主要作用，可以进一步分为图层建模和所有者建模。 物理建模需要物理和计算机图形的结合。它是 VR 系统中的高级建模。运动建模旨在解决位置变化，敲击，捕捉，夸张和缩小，以及表面重整等。动作建模指定形状的特征，对象的感受以及他们的行为和回应方式。

实现 VR 建模的模式和关键技术

基于构建 VR 场景的不同方式，主要有三种建模方式来实现 VR 场景：MBR，IBR 和基于图形和图像的组合建模模式。三者中，GBMR 基于计算机图形学。它从场景抽象的所有过程漫游整个场景，形成一个虚拟的形状，形成一个虚拟模型。 IBMR 实现的效果是一个全视角的 3D 场景，通过基于许多分开的图像或流利的视频来组织全场景图像。在基于图形和图像的组合虚拟建模中，需要通过 IBMR 构建虚拟场景，然后使用 GBMR 在虚拟场景中形成交互对象。许多专家对 VR 建模中使用的关键技术进行了研究和实践。有三种

类型引起人们在 VR 建模的兴趣。一种是提升实时效果的技术，包括详细图层技术，基于图像的图形绘制技术，模型简化技术，场景监视管理技术和实例化技术等。另一种技术着重于更真实的体验，包括纹理成形，照明和阴影生成技术以及快速绘画计算，旨在强调互动，主要是关于自由度。

构建三维 VR 场景

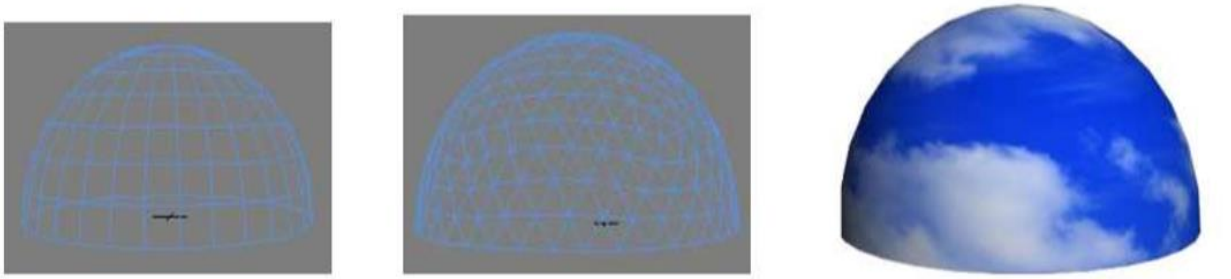
数据收集和预处理

建模收集的数据主要包含周边景观，纹理，实际模型和数据的信息以及交互信息。其中，自然场景，实际物体的外观，几何形状是从图片和视频中收集来的。收集数据时需要考虑以下几点：熟悉系统和系统的功能，建模实际对象的特征。数据应该在物体在真实场景中的位置采集，并且被命名和编号。照片也应该在几天中的同一时间段拍摄。阴天，雾天和下雨天都不适合采集图片。数据应定期备份。数据预处理意味着消除错误和无用的数据以保证数据的准确性。本文讨论了三种需要预处理的数据：自然风景，实际的物体和纹理。有关自然风景的数据可以使用 Photoshop 简单处理消除，颜色修改，拉伸和重整以达到基本标准。物体的形状，尺寸和位置都需要被修正。物体的纹理也会通过 Photoshop 的数据库进行预处理，转换成可以被 3DMax 读取的一个纹理文件。

三维场景的构建

首先构建天空和遥远的景象。遥远的景象对细节几乎没有什么要求，并且它的良好表达的效果是可以接受的。通过选择一个合适的三维建模软件，本研究使用了 3DMax，虚拟天空作为一个半球模式展示如图 23 由于遥远的景象对细节层次的要求不高，因此可以在第一层构造一个密封的“盒子”。然后，通过在“盒子”的不同面上制作相关的纹理，可以实现相关方向上的远景模拟。

图 23 球形天空

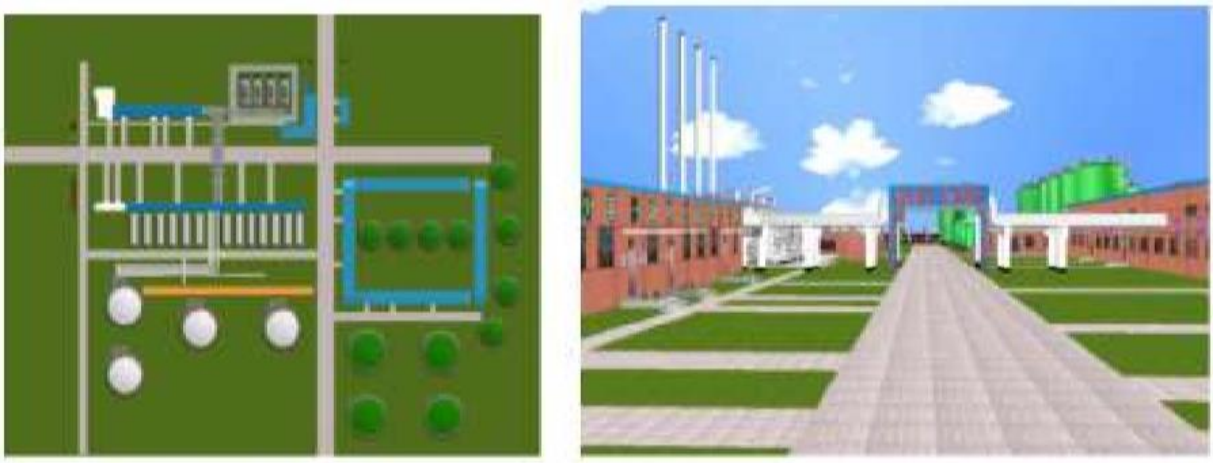


其次，构建场景中实际对象的模型。 场景中有移动和静态 3D 对象模型。构建独立静态对象的步骤包括：收集建模数据，修复模型的层结构，构建模型，消除无用的几何元素，将凹多边形变成三角形，计算阴影和设置纹理。为了构建运动物体，需要重建运动物体。可以通过在建模文档中增加运动部件的自由度关节，设置相关的局部坐标并固定它们，分析和解决多个运动部件之间的关系。 一些对象模型如图 24 所示。

图 24 对象模型

最后一步是构建环境景观（草地，树木和花灌木等）。 由于环境景观的位置分布广泛，数量多，形状不规则，因此需要通过处理来简化。 为此，应用简单的几何对象来显示它们的形状，并且使用纹理填充来表达他们的真实感。 草地的效果如图 25 所示。

图 25 草地



场景的组合调度管理

建模后的场景经过组合和分组形成一个完整的虚拟模型。 每个构建的对象模型都被分组，其文件被组合，并被添加到指定位置以形成整个模型。 图 26 通过使用外部参考技术显示了组合整个场景模型的效果。 场景的管理和调度是通过从一般场景漫游到复杂场景来实现的。 主要包括不同场景地形的分离和场景模型的动态管理。

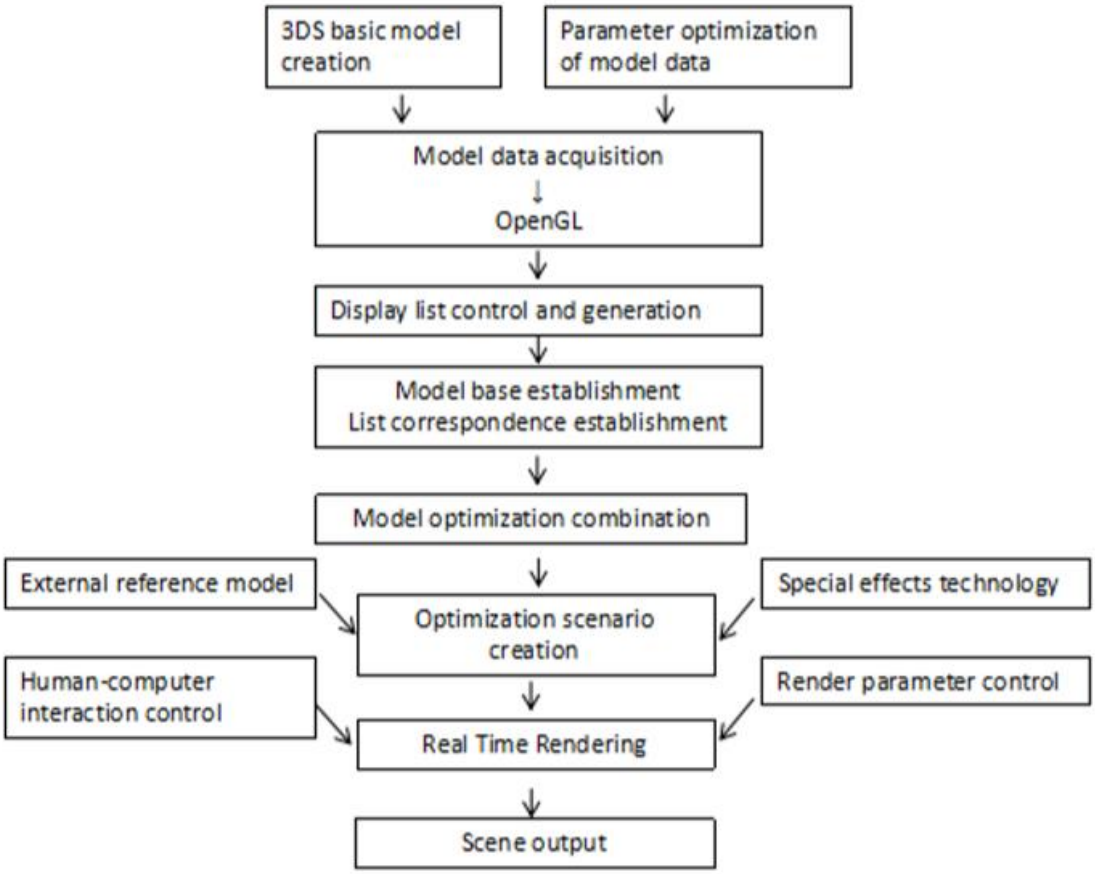
图 26 组合场景

三维建模技术在 VR 系统中的应用

系统分析与设计

为了详细地展示 VR 建模技术在虚拟现实系统中的应用，本文以联合站系统为例进行了虚拟对象的设计。联合站系统是油田系统的一个关键联合。它执行从不同转运站收集的油的一系列操作，然后将油输出到指定的接收器。一个统一的车站系统的大型场景特别要求 VR 系统和三维建模。它必须非常真实，即时和迭代。因此，本文利用 3dsmax 和 Virtools 分析和设计 VR 系统中应用的建模技术。根据开发目的和用户需求，系统设计成 13 个功能模块，如图 27 所示。

图 27 系统设计流程



系统实现

一旦联合站的三维虚拟场景形成，就可以用人机交互技术来编写一个用于即时控制系

统的模拟程序。 一般来说，系统的虚拟化主要体现在三个方面：三维物体移动的模拟，表现场景的艺术以及数据库技术。 Virtools 的 Array 函数作为数据库技术，Get Row 模块用于连接数据库和系统。 表达场景的艺术是增强生动和沉浸的感觉。 它涉及到光线的使用，视角的选择，场景中图层的构建，以及渲染的正确方法。 3D 物体移动的模拟旨在使整个系统看起来更真实，听起来更真实，移动更真实。 Virtools 也可以实现这一效果。

总结

随着 VR 技术在越来越多领域的应用，对大型复杂场景的建模需要越来越高的要求。由于三维建模技术是构建虚拟场景的基础，对三维建模技术应用的研究具有重要意义。本文采用 3dsmax 和 Virtools 构建虚拟现实系统，以油气统一处理站为例进行仿真操作。在数据采集与处理，场景构建，场景组合调度管理，框架设计，系统实现等方面进行了整体操作实践。与传统的建模技术相比，过程更加简单，虚拟场景更加生动，因此满足了对高真实性，沉浸感和交互性的需求。这种建模技术在 VR 系统中的应用可以扩展到许多其他领域，并作为有价值的参考。随着 VR 系统需求的不断提高，对建模技术以及系统的功能和应用将会有更多的创新和研究。