# 虚幻引擎三位建模实现流程

## 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）

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。

## 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。

## 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。

//是否要说明过渡顶点结构

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。

## 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。

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.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。

## 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。

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。

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。

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。