# 模型规范

本规范面向对象为技术人员。

定位方向是使技术人员快速理解劲源业务对应的技术实现方式。

## 1.命名

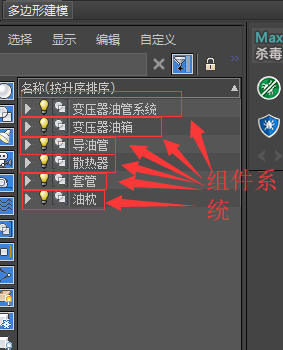
设备类型包含了主变，刀闸，端子箱等可操作交互设备。

Revit：必须以标准的命名规则进行规范（根据调账如果调账，如果找不到需要找变电站人员咨询），不允许出现没有任何意义的名字，比如cube001,cube002等，每个零件名字后面不允许出现后缀001,02,等，如果这些编号如果是软件算法自动生成的需要手动删除，或者手动排序改写后缀为1,2,3,4,5...。需以设备——组件（组件层可以嵌套多个）——零件三个层级为基准，另外需要执行轴心居中，缩放初始化，单位重置，整体坐标成组归零。

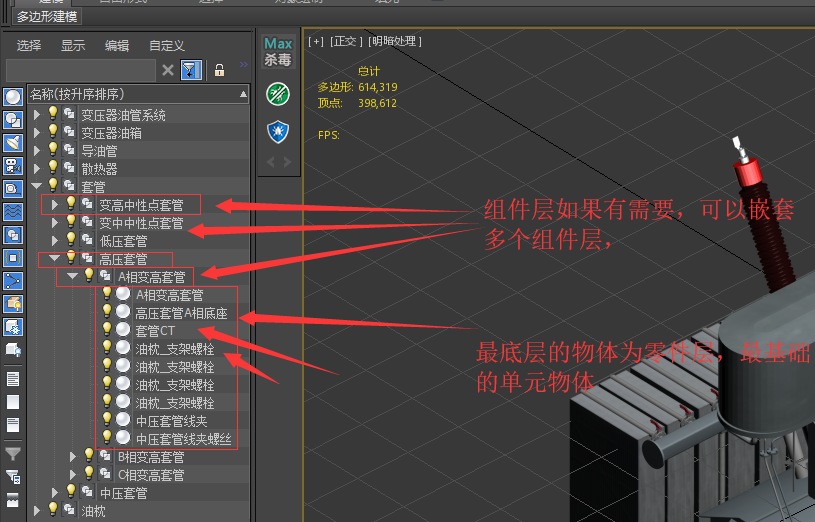
1. **设置节点（子父级）**

比如变压器:设备层(变压器)

组件层：6个

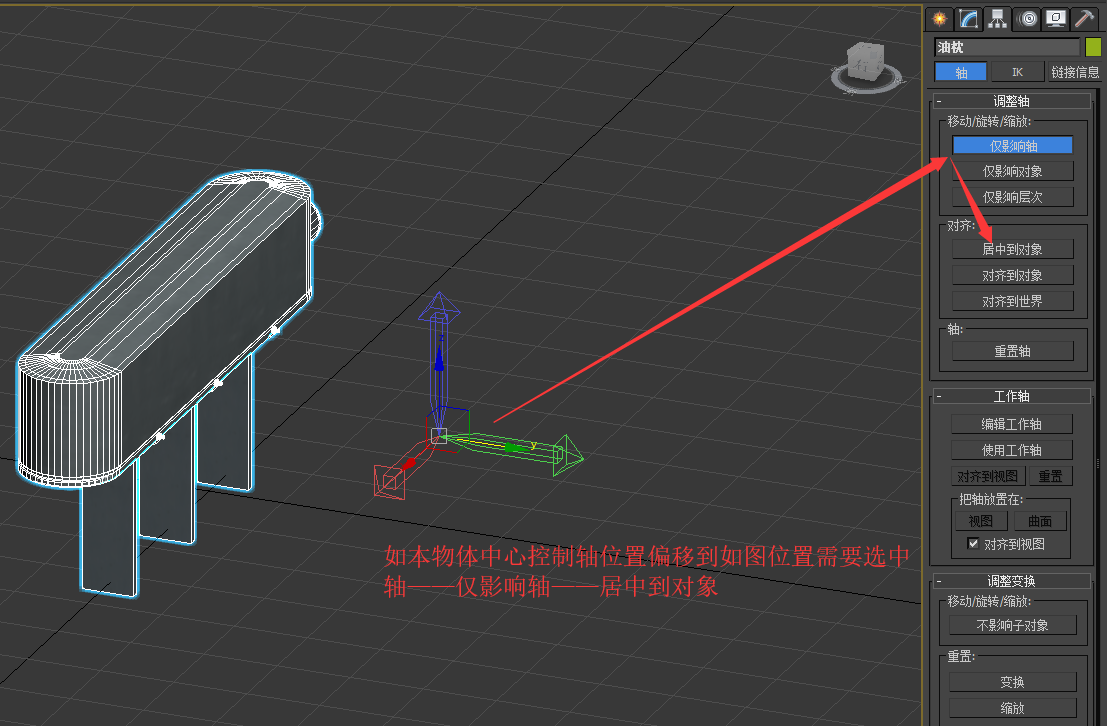


零件层：n个



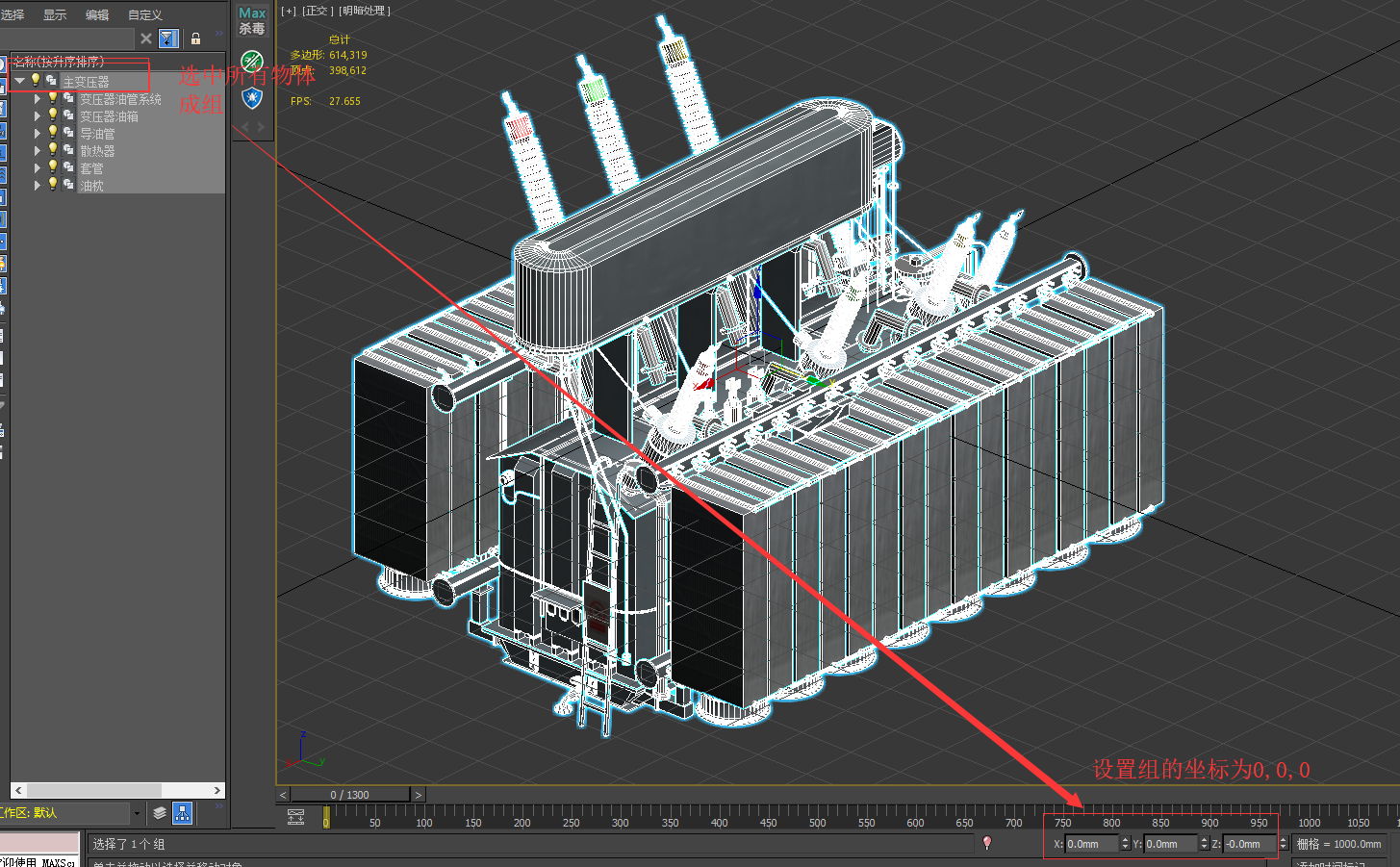
**3.轴心**

轴心居中



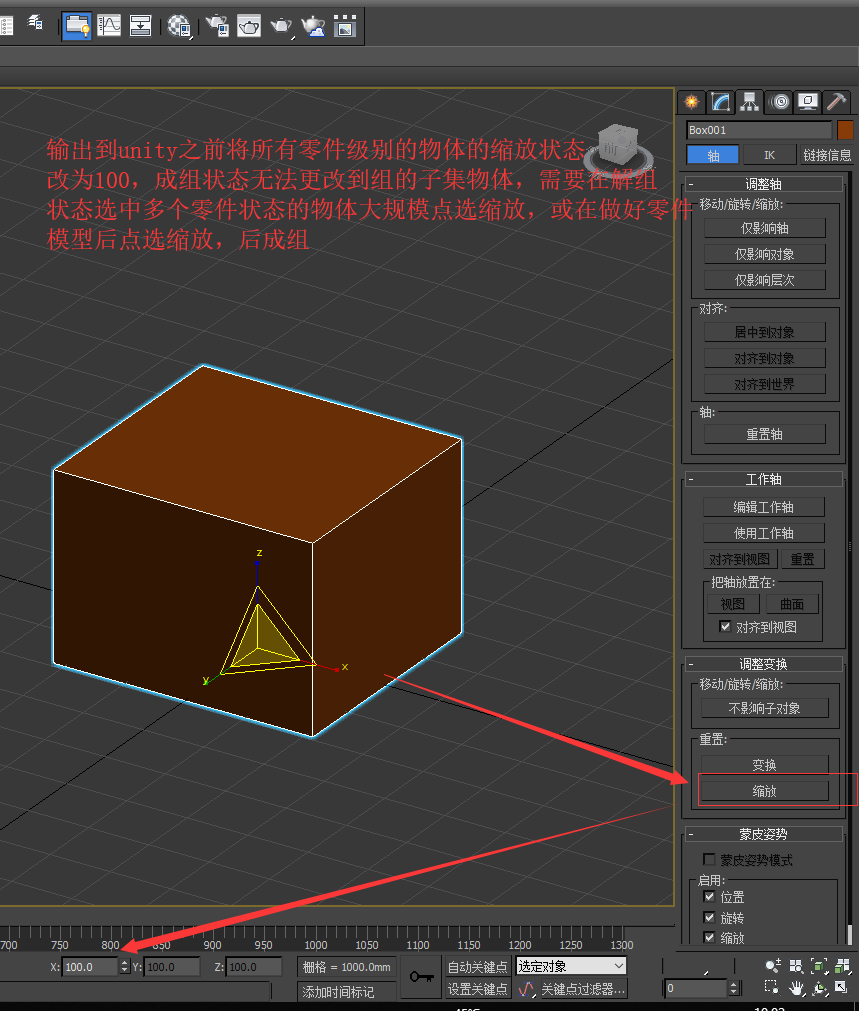
**4.坐标**

坐标归零



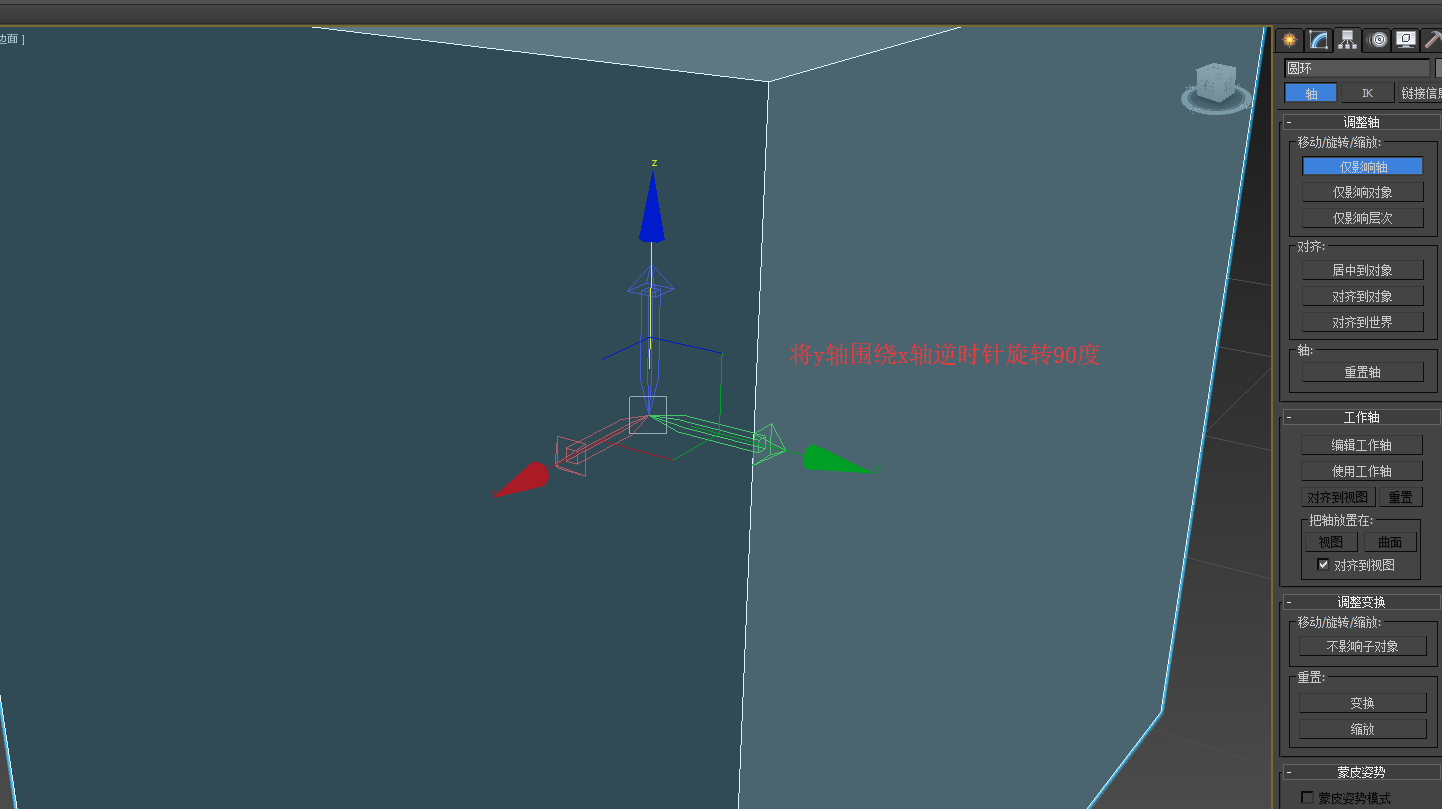
**5.缩放**

缩放归零(如果被缩放的物体有动画，则需要在做动画前将物体的缩放系数初始化为100,100,100)



1. **旋转Rotation**

3DMAX中使用的是右手坐标系，但是Unity则使用的左手坐标系，所以导出Fbx的时候需要改一下3D模型的轴向，将max模型 的轴心y轴围绕x轴逆时针旋转90度，如果不处理这一步会导致模型在unity的rotation.x始终处于-90度



**7.单位**

系统单位设置：unity和3dmax的单位为1:100，需要改写3dmax的单位

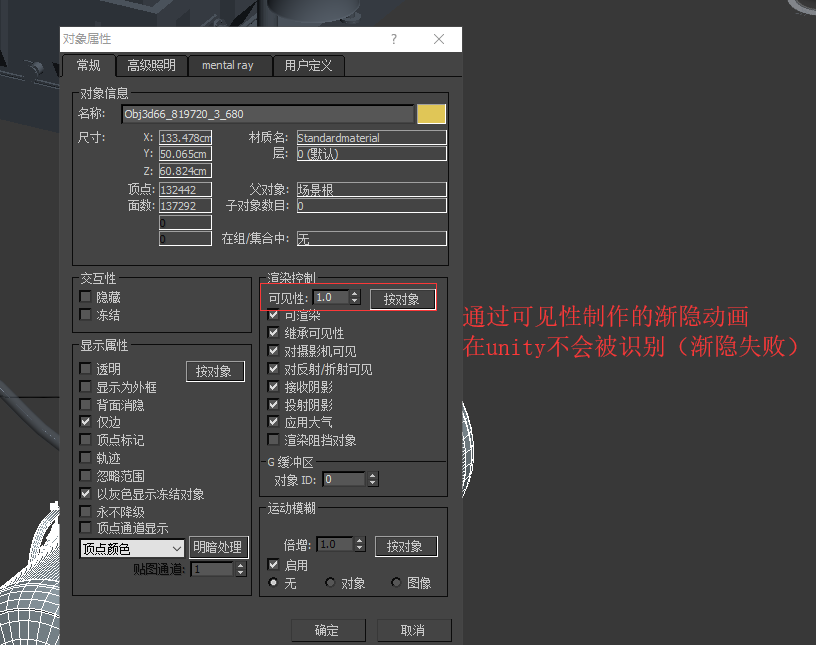


**8.动画**

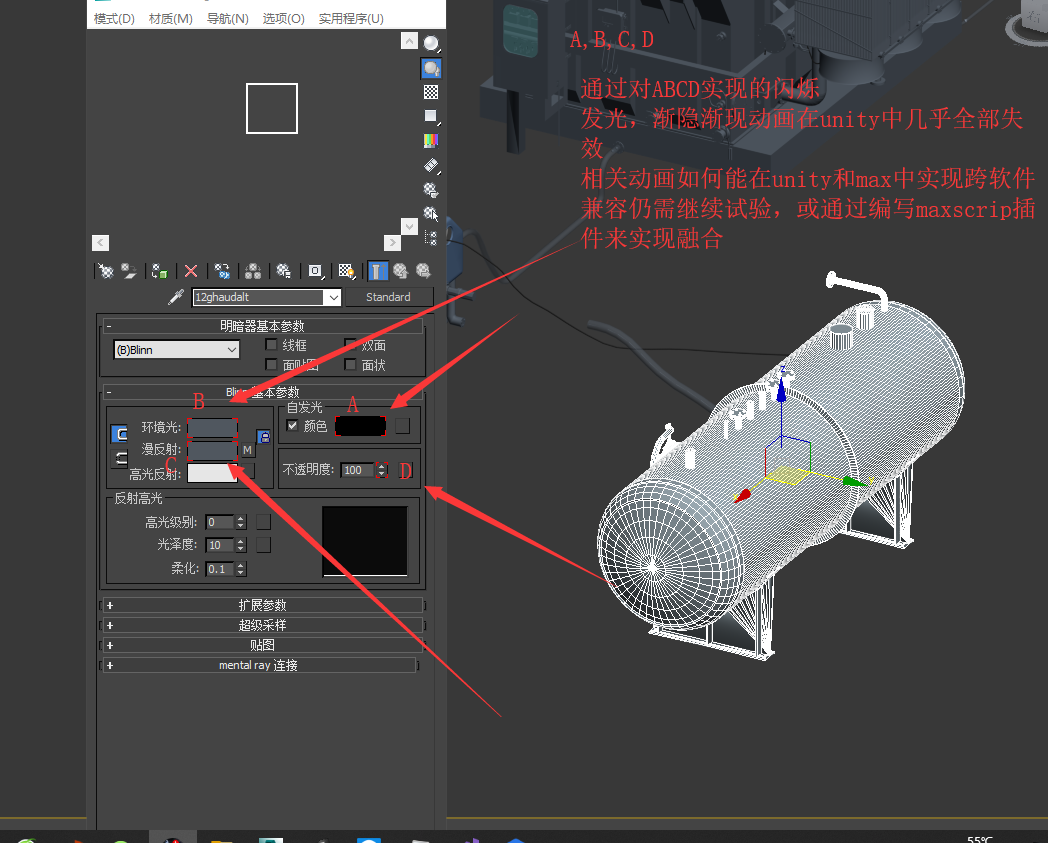
1.Unity默认不具备操纵模型点层级移动，涉及到改写顶点层级的动画需靠辅助骨骼来实现，否则导入unity后动画会消失，

2.3dmax动画导入unity后fbx的层级和任何gameobject的名称一律不能改写、拖动，否则动画会识别不到模型。

1. 如果动画过长，需要十分谨慎处理模型优化，过长的帧动画导入unity会对硬件cpu,内存等产生极重负担，需要将带动画的模型优化强度提高，尽力减少模型顶点数，减少子父节点。极端处可以用贴图代替模型。
2. 帧动画交接程序前建议使用网格合并将无关动画的模型附加成一个gameobject,尽量压缩模型容量
3. 在max中将动画分割成几段，导出fbx时，fbx的容量大小几乎一样，再导入unity切割编辑时对硬件的负担似乎并没有降低。相关实验需要继续探索。

通过对象属性的渲染控制中的可见性操作来实现的动画渐隐在unity中会无法识别（这段渐隐会失效，但对Transform的position的改变仍然是起作用的），用这种方式时请注意此路不通。

1. 3dMax标准材质球动画在unity中无法被识别，两个软件采用的是不同的材质系统相互只能通过max中的stand材质球的漫反射颜色和漫反射项后的位图插入来实现微弱的互联。因此涉及到颜色闪烁，渐隐等非位移和顶点变换的动画建议在unity中制作。



**9.贴图**

问题一：图纸。

涉及到贴图部分由于劲源业务的特殊性（图纸在BIM模型组开工的时候都是断断续续的拿到，而且经常残缺不全，图纸与现场安装的设备并不完全吻合，缺个零件和对不上某某个阀门，这里少个洞那里少个某某啥的。。。），前期上级会安排按照图纸来，根据现有的（注意是残缺的，这种图纸经常不准）图纸做，给个进度安抚客户。后面准确的图纸来了模型师无奈照着图纸再来一遍马拉松。。

问题二：还是图纸。

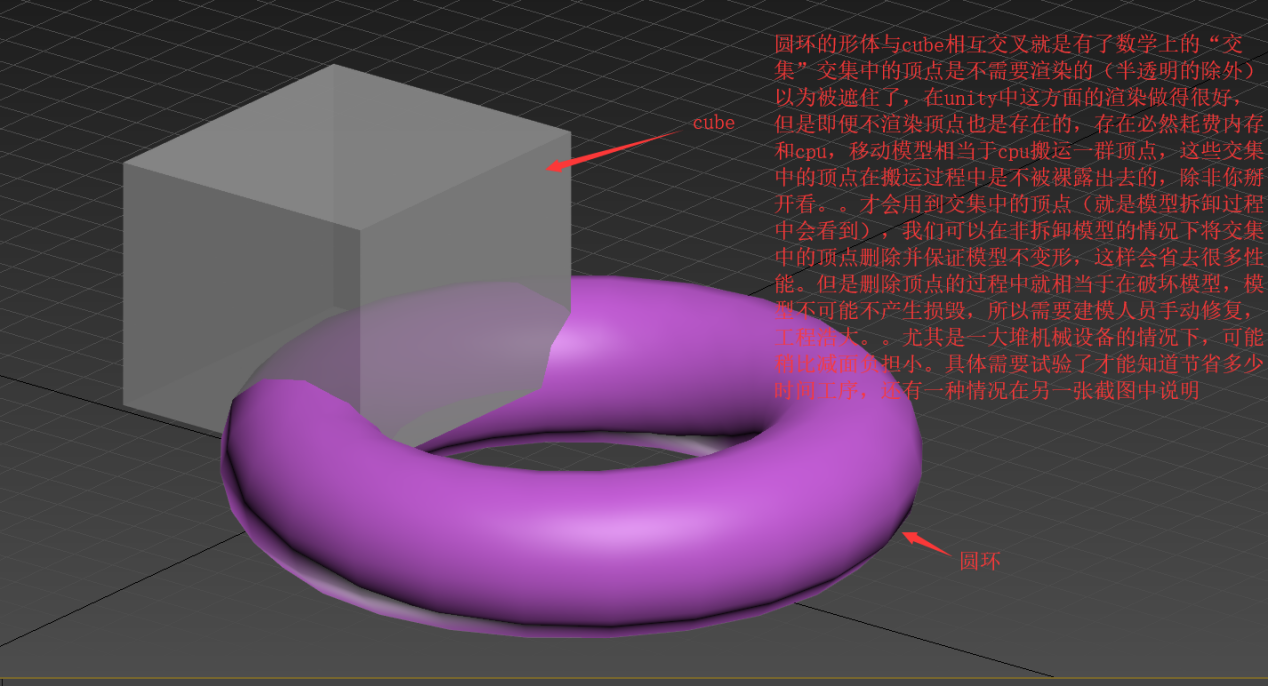
后面客户一看对不上现场的造型，完了就要模型师需要对比现场照片更改（这个过程时间跨度会很长，都是后面陆续发现的），因此模型会有初版，中期版，末期版，最终版，最最终版。。。。

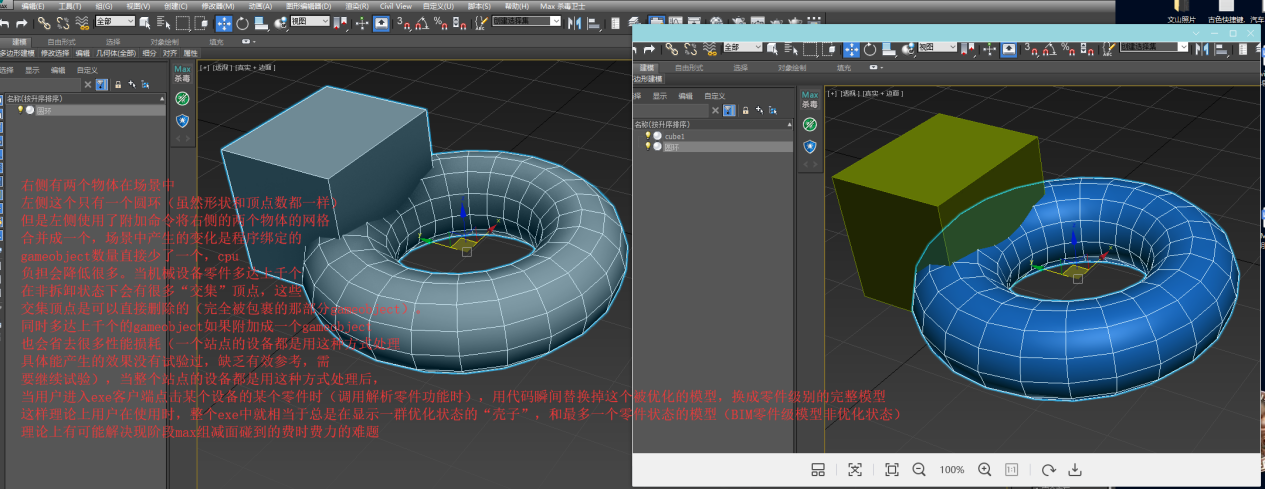
贴图是模型定型后才会制作，展uv，刷sp，可以理解为盖完房子往上面刷漆涂料。所以如果模型没有定型不建议动工制作贴图，尽量安排在后面，避免拆了重做，拆了重做，拆了再重做。。。。。

**10.模型轻量化**

Fbx轻量化一直是业界的难题，目前市面上没有发现（暂时）智能插件可以将庞大的无序乱线重新建立规整的边面，劲源的模型数量庞大（BIM人多，做的快），用一些市面上的减面插件，面减下去了，造型削没了。目前使用的是最稳的方式3dmax手工改。这里由于图纸的问题建议3dmax人员与BIM人员了解清楚哪些模型后面大概率会改，哪些小概率会改，避免再来n次马拉松。

目前通过探索找到了一条网格合并的路线（实验尚不完全），可以通过网格合并（程序称呼网格合并，在max就是附加多个模型为一个GameObject），具体的做法就是在做一层设备的外壳放在unity的sence中给人看，这群外壳的模型约等于附加的模型，这里需要详细解析下“约等于”这个现象，还是上贴图解释吧





通过上面两幅图可以看到（不方便的话就拉大看）附加仅仅是通过节省节点减少场景中的gameobject来实现了优化，但是附加并没有解决“交集”中的可删减的点所以，用要想达到真正的外壳级别的模型还需要删除“交集”中的点，从诺斯顿公司扫描的模型看他们几乎达到真正的外壳级别，除了命名和节点（父子关系）和轴心，几乎不用二次处理。所以理论上在不考虑成本的情况下可以使用扫描的外壳模型结合BIM制作零件级的高精模型，然后通过程序替换来实现整个工程的优化。（具体的实施还需要测试实验）

下面是建模基础性规范。

模型轻量化之网格合并（俗称附加）

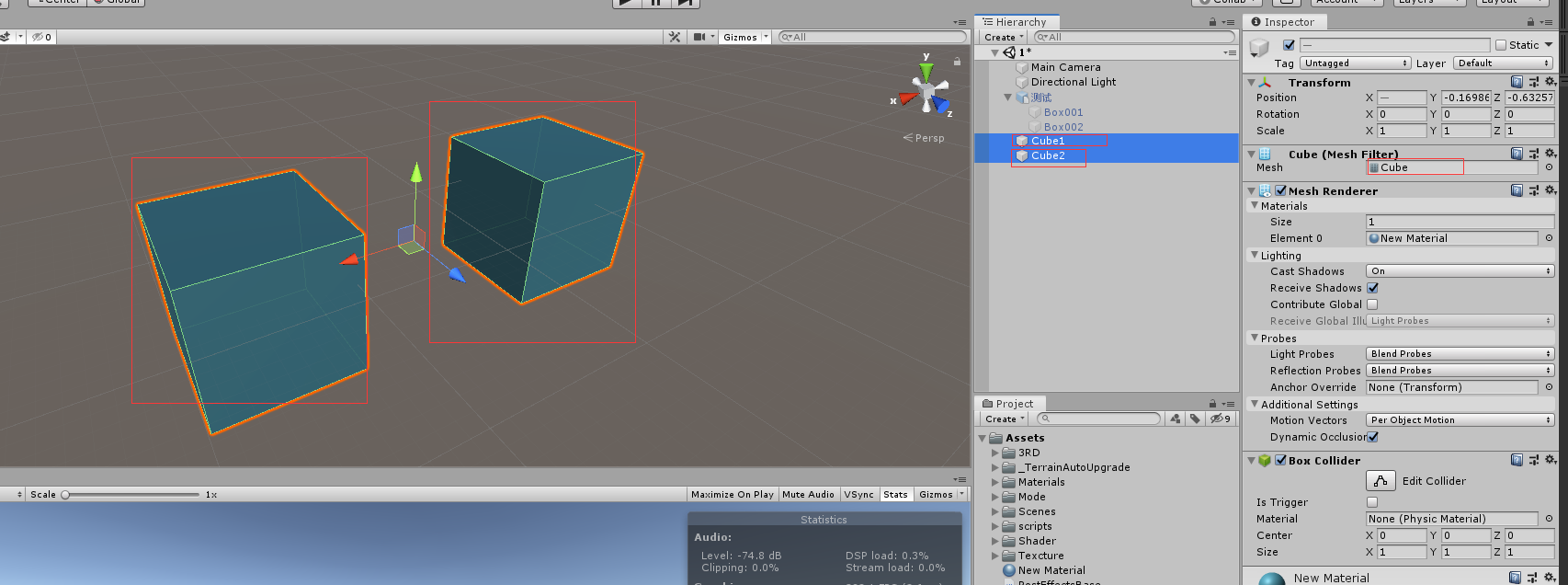
这种操作方式在上面的论点中有提到，最近笔者试验了不少次，得出的结果虽不尽完善但是总体结果并不乐观

首先：这种附加方式在3dmax中做就必须有一个强大的硬件支持，在附加命令调动时刻是单线程的运算，cpu并不会满载，但是内存情况很严峻因为合并的时刻会把大量的顶点面片放到内存中储存运算，当顶点超过一个峰值时内存会被撑爆

直接导致电脑黑屏，对应应用程序死掉。从劲源的业务来看这种情况比较今后会比较常见，需要面对的是大规模、巨量顶点群和贴图，这种情况放在笔记本上模型工序的后期将完全无法应用，甚至放在台式机上也无法正常展开编辑

**Mesh共用详细解说**

相同的模型共用Mesh能节省内存和提高工程的流畅度。如图：

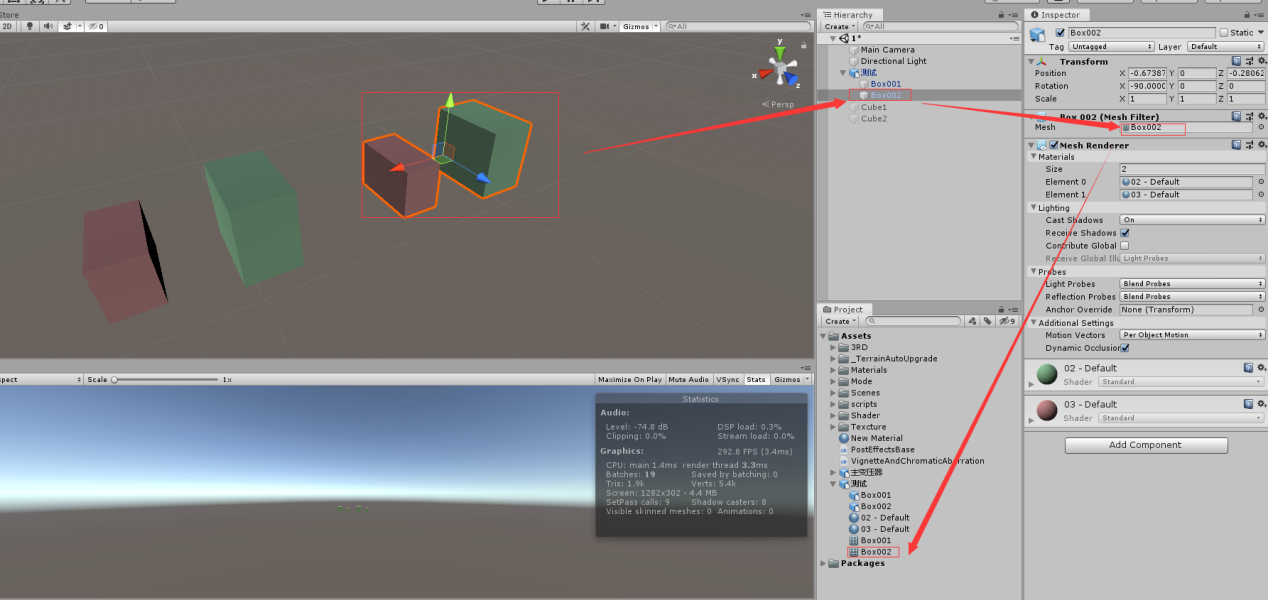


cube1和cube2是共用一个Mesh:Cube这样子满足正确渲染两个模型的需求同时节省了资源。

而如果我们直接将3dmax二次处理的模型直接导入unity，现象如图：



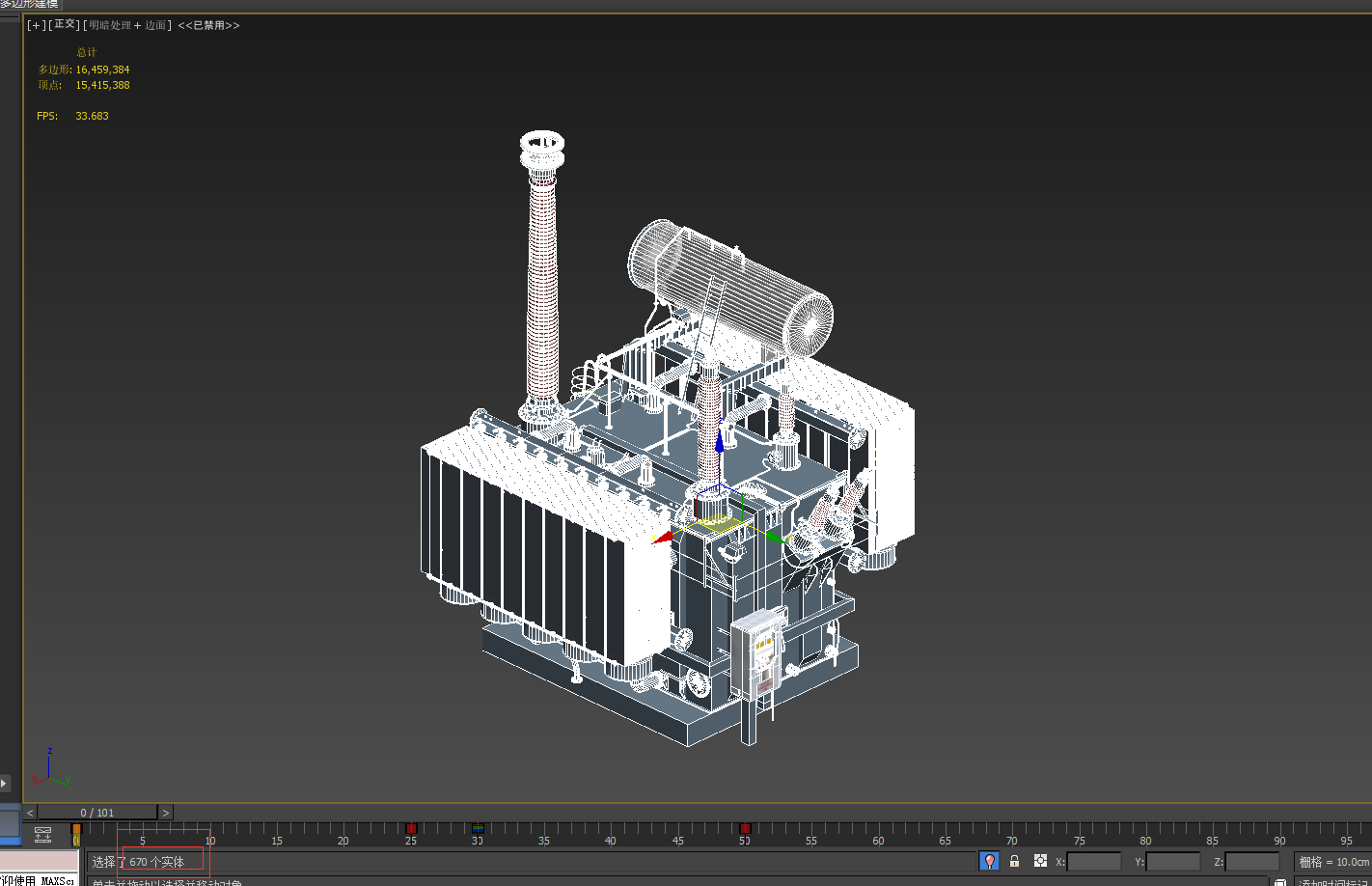
![UR}@J]41769G2F)](1)K]L8](data:image/png;base64,)



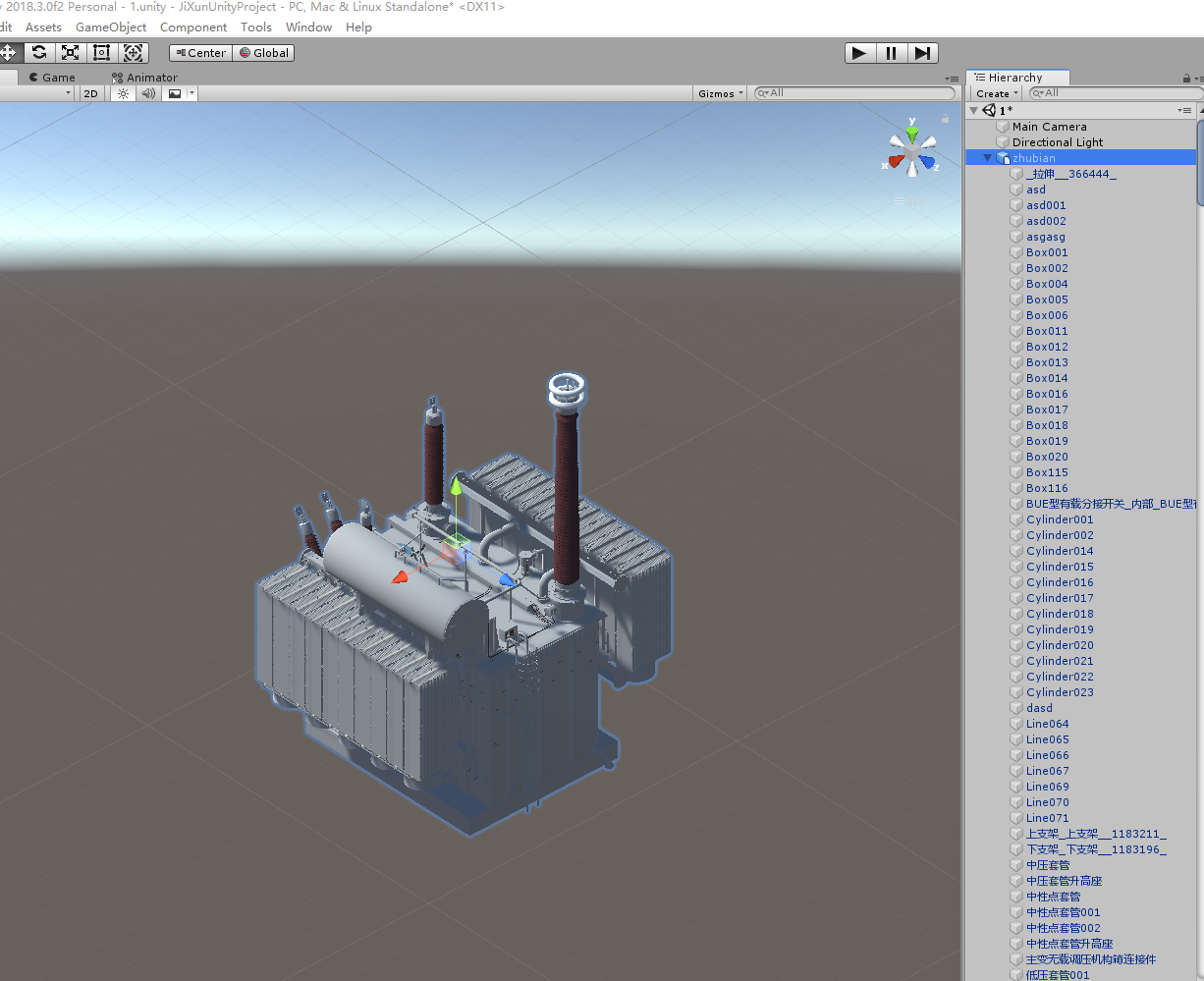
我们通常在Revit模型中就会把整个场地的所有设备模型制作完毕，然后经过3dmax二次处理后导入unity，但是在这个过程中重复的设备模型将失去实例效果在引用Mesh时Box001和Box002将会自动能生成各自不同的Mesh,而在变电设备场景中往往存在大量的模型是copy状态的，形体外观都是一致的，此时如果经过美术人员的Mesh共用处理将会进一步提高资源流畅度

**工序优化措施**

将模型子物体炸开到零件层子物体状态遍历各个导出fbx，可以有效减低模型之间的耦合度，此情景适合绑定众多信息的设备，如：主变压器。



当我们选中打开组或解开组或炸开组状态的变压器导出选中对象时会输出一个fbx，然后此fbx里面坠上了670个子物体如图：



此时如果我们需要对变压器的670个子物体中的任何一个进行更改那么将需要在3dmax中处理完替换整个主变压器fbx，模型之间的耦合度非常严重，尤其是劲源业务中主变压器模型的完善是经过很漫长的周期才会完善到一定程度，后期经常会不定期更新更改的情况，最好将670个子物体单独导出成670个fbx,这样如果其中任何一个零件更改了那么只需要精确替换对应的涉及模型。此时若手动操作执行则需要导出670次，极容易遗漏，不具备操作可行性，需要编写插件用程序自动导出处理，编写代码如下，直接运行即可：

--批量导出成组模型的fbx

rollout rl\_Rename "批量导出建筑fbx" width:250 height:110

(

label lbl\_Export "导出路径：" pos:[8,10] width:250 height:20

edittext edt\_Path "" pos:[8,32] width:200 height:20 across:2

button btn\_Liulan "浏览" pos:[215,32] width:30 height:20

button btn\_Group\_Export "成组导出" pos:[8,70] width:110 height:20

button btn\_Obj\_Export "普通导出" pos:[130,70] width:110 height:20

on btn\_Liulan pressed do

(

edt\_Path.text = getSavePath caption:"Get Directories" initialDir:(maxfilepath)

)

on btn\_Group\_Export pressed do

(

if edt\_Path.text == "" then messagebox"请先选择保存路径"

else(

selectionarray = (selection as array)

if selection.count == 0 then messagebox"请先选择需要导出的物体"

else(

for i in 1 to selectionarray.count do

(

if classof selectionarray[i] == Dummy then

(

select selectionarray[i]

exportfile (edt\_Path.text + "\\" + selectionarray[i].name +".FBX") #noprompt selectedonly:true

)

)

)

messagebox("导出完毕，路径为 " + edt\_Path.text)

)

)

on btn\_Obj\_Export pressed do

(

if edt\_Path.text == "" then messagebox"请先选择保存路径"

else(

selectionarray = (selection as array)

if selection.count == 0 then messagebox"请先选择需要导出的物体"

else(

for i in 1 to selectionarray.count do

(

if classof selectionarray[i] != Dummy then

(

select selectionarray[i]

exportfile (edt\_Path.text + "\\" + selectionarray[i].name +".FBX") #noprompt selectedonly:true

)

)

)

messagebox("导出完毕，路径为 " + edt\_Path.text)

)

)

)

createdialog rl\_Rename

上端断码大意是执行选中的对象遍历导出，效率极高。

**导出格式**

经过探索建议尝试使用revit导出时不再以fbx为输出格式，更改为sat格式导入到3dmax，在近期尝试的过程中发现这种格式相对fbx会大幅降低点面，同事模型形体不失真，sat导入3dmax时转弯的顶点需要设置焊接否则会炸开顶点，相关规则细节还需要继续完善。

### 建筑类型

包含了常见的房子，大楼，大门，走廊，地面，围墙

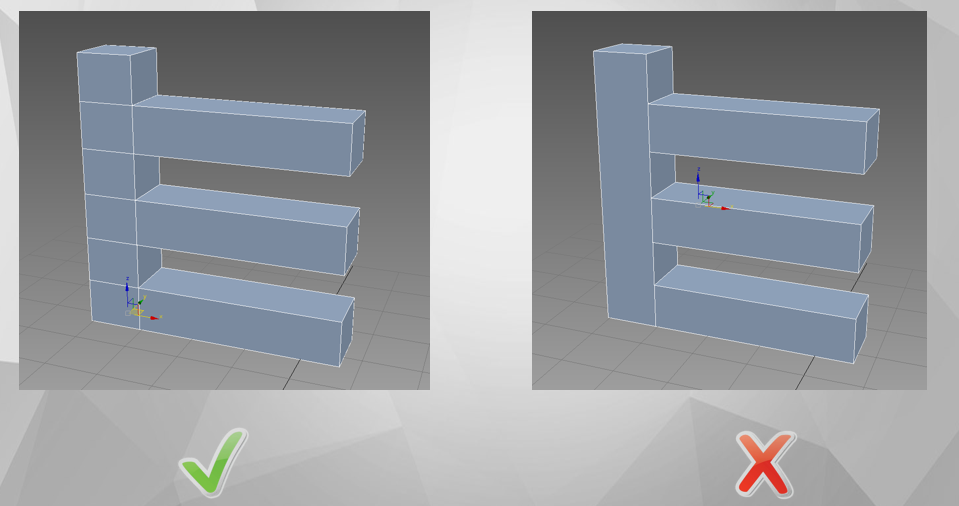
### 辅助类型

## 制作要求

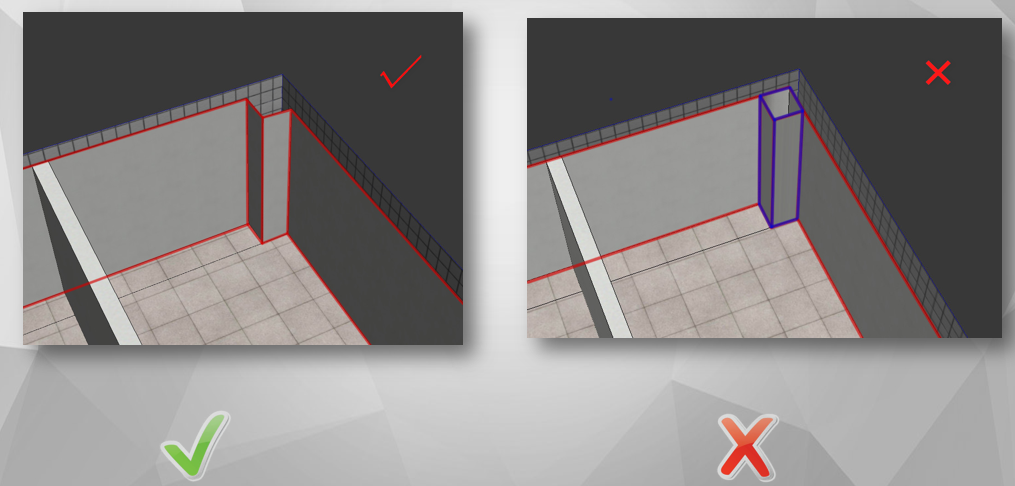
### 通用

1. 如遇到CAD图纸与现状照片有偏差，一切按现状照片为主，需要按照现实中实际尺寸建模
2. 制作完成后根据地形对位之后将 模型 的“x y z” 轴坐标归“0 0 0”
3. 建模前在三维软件中设置统一单位，如没有特殊要求，统一为厘米（cm）；导进unity里面，大小比例必须是“1 1 1”
4. 模型整体法线统一，不能有反面，叠面，烂面，一旦出现这些状况，必须重新建模
5. 对于表面光滑的模型，为模型添加平滑组导出
6. 模型完成后，需要以模型外框为基准勾画挤出一个不用带细节的box，大小、高度需和原模型一致，以用来地形对位和烘焙阴影
7. 要保留模型的修改器，再次修改时，防止出错
8. 习惯性按阶段保存模型文件

### 建模

1.在建模布线上，要有一个整体布线的思路，尽量避免穿插面，叠面，遮挡面

2.内构承重柱在内墙加线简化处理，不用另外挤出承重柱，减少叠面的情况



### 贴图

* 相同一张贴图纹理要保持尺寸一致，不能随模型大小缩放uv。相同贴图不同模型可以给统一给通道1“uvw map”修改器，方便统一贴图尺寸。
* 不一样的模型但有相同材质的，要公用同一张贴图，材质球命名要一致，uv纹理密度要相同。

### 烘焙

检查UV是否有错位，模型是否有重面。

