问题：

如果我们希望在游戏场景中得到如右下角所示的一座房子，需要经过哪些工序？每道工序包含哪些关键数据？



分析：

基本的工作流程如下：

1. 建模，获得模型的拓扑结构。
2. 根据模型不同部位的材质要求制作材质球。
3. UV展开，将模型的表面展平到二维的坐标空间，从而为纹理映射做好基础。
4. 绘制贴图，根据UV展开的情况，为模型绘制贴图。
5. 如果有角色动画，则需要对模型进行骨骼制作，绑定，蒙皮并制作控制器。
6. LOD制作，考虑到性能的话可能还需要制作几个低精度版本的模型。
7. 将模型通过fbx或其他格式导入Unity或其他游戏引擎。
8. 在Unity内进行打光，设置光照属性，根据引擎内材质球的具体要求进行进一步调整，如果需要烘焙光照贴图则需要进一步优化。

下面展开叙述：

1. 建模

如图所示的房子的模型结构本身十分简单，基本是标准的几何形，因而可以考虑通过基本几何体的布尔运算得到，例如长方体加三棱柱就得到一个房子大体结构，两个薄的长方体按一定角度摆放就编程屋顶，这样就获得了一个房子。复制这个房子之后垂直摆放两个房子就得到上图的交叉式房屋。最后简单使用几个小立方体得到房屋前的楼梯。

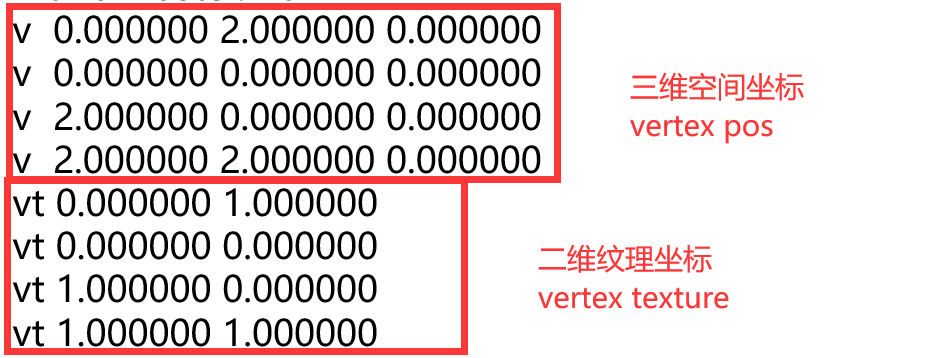
这个阶段最重要的是得到模型中每个顶点的空间坐标和拓扑结构。即顶点表和线表面表，用于交给GPU的vertex buffer和index buffer进行绘制。

1. 材质球

显然可以看到，房屋的不同部分，屋顶、墙面、镜子、靠近地面的砖块都需要不同的材质来进行表现，因此可以分别在dcc软件内指定不同的材质球，以区分物体的不同表现。当然这需要依赖于物体的具体UV展开，以获得正确的UV纹理坐标。

1. UV展开

模型的顶点是三维空间坐标和其他一系列数据的集合，这其中就要包含一个二维的UV空间坐标，如下图：



当然在DCC软件中不会以如此不直观的方式呈现给艺术家，而是使用可视化的表现方式，但是本质无二。

1. 贴图绘制

根据材质和UV坐标信息，我们能够知道在一张二维的图像上绘制的像素点对三维物体的表面属性有什么具体的影响。

因此在材质和UV信息确定的基础上，我们能够进行贴图的绘制，这个绘制可以将UV图像导出到PS中参照绘制（当然这是一种费力不讨好的方式），但我们最好是使用类似substance painter的专业贴图绘制工具，它们提供了直接在三维模型上绘制贴图信息然后导出二维贴图的功能。

1. 角色动画

在这个模型中，大概率是不需要角色动画的，因此暂且跳过这个制作工序。

1. LOD制作

LOD，即Level of detail。一般而言在上述步骤完成后，模型已经可以直接导入游戏引擎进行进一步调整了，但是如果考虑到游戏性能的话，可能还需要考虑制作LOD模型，即在每一层级制作一个精度减半的模型来满足需要。可以想见，制作一套LOD模型所耗费的存储量不会超过这个模型最高精度版本存储量本身，但是能够在运行时节省大量GPU绘制时间和CPU与GPU之间通信的负担，因此LOD是一个很经典的以空间换时间的trade off。

1. 使用FBX导入Unity

这部分较为简单，掠过

1. 设置引擎中的光照

引擎和DCC软件中，因为面向的渲染目标不同（实时渲染和离线渲染），会产生诸多差异。例如DCC软件中设置的材质不一定与引擎中提供的材质相兼容，引擎中使用的光照也不能直接由DCC软件中的光照设置得到（也许在将来PBR的管线高度普及的话，引擎和DCC软件共同兼容一类光照数据结构会成为可能）。

因而我们需要重新设置光照。

另外，在Unity中，很大概率我们需要烘焙光照贴图，或者是进行预计算的实时全局光烘焙来得到一个较为自然的光照结果。

这需要我们进行设置诸如物体的光照贴图UV，光照分辨率，采样率等等跟全局光照相关的配置。

另外如果有更高的需求，还可以调整引擎的色彩空间为Linear空间然后施加屏幕后处理效果。