1. **总体技术路线**

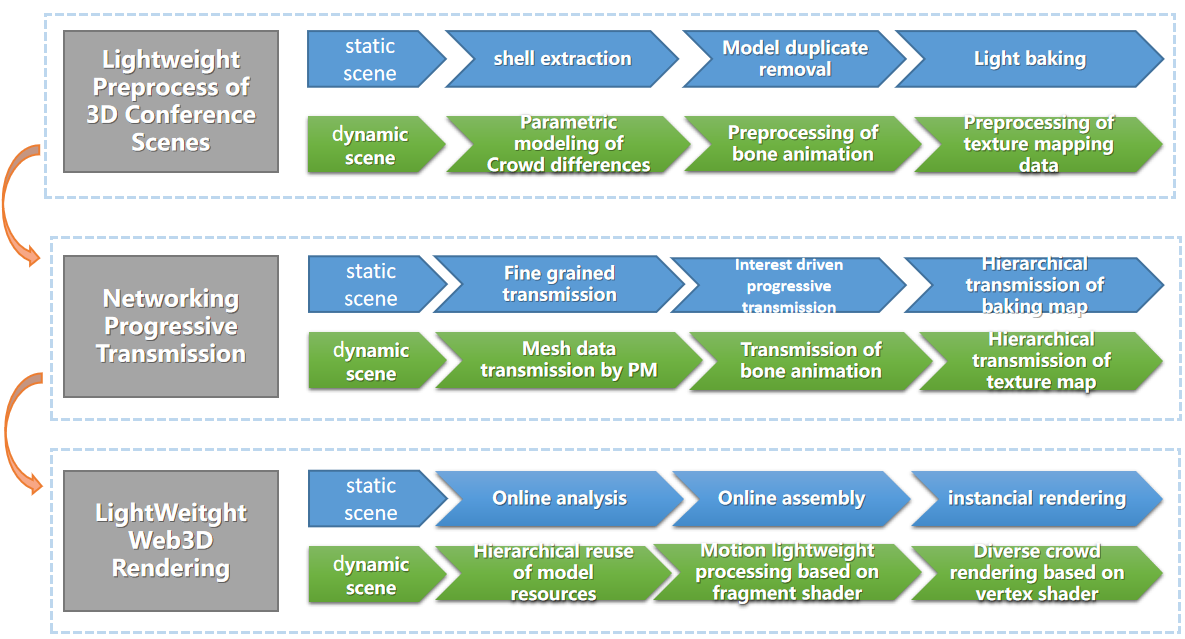


图 1 总体技术路线

本文使用的轻量化技术主要通过服务器预处理、网络传输、浏览器渲染三个阶段的处理来完成。

服务器阶段要实现大规模会议情境的轻量化预处理。对于静态场景，这一阶段首先要提取场景的外壳，然后进行去重处理，去除场景中的重复单元，之后再对场景进行光照烘焙。

对于动态情景，首先要对大规模人群进行参数化处理，然后提取出骨骼动画，并对纹理贴图进行预处理。

网络传输阶段要实现细粒度化渐进式传输调度。对于静态场景，将场景分解为更小的单元进行细粒度化传输，通过兴趣度分析各个部分的重要性，从而确定这些单元的传输优先次序。用于使用了烘焙贴图，所以有些部件的纹理贴图比较大，这时对贴图的分级传输处理就非常重要。对于动态情景，考虑到人物模型的精细度比较高，所以采用了PM的方式来传输网格数据，并且也使用分级传输的方式处理纹理贴图。

浏览器阶段要实现轻量级大规模会议场景在线渲染。对于静态场景，首先要分析收到的数据，然后将各个部分的资源组装到一起，最后利用实例化渲染技术生成场景中的重复单元。

对于动态情景，首先要对人群的模型资源进行分级复用，再使用fragment shader实现人群的动态化，然后再使用vertex shader实现人群的多样性。

1. **关键技术之一：大规模会议情境的轻量化预处理**

轻量化处理的核心是资源重用，我们使用该技术必须首先明白哪些资源是可重用的，并且尽可以提高资源的重用度。另外为了效果要保留丰富的多样性，参数化调整每个对象，另外可以通过对不同资源进行搭配组合来提高多样性。

4.1**静态会场场景的轻量化预处理**

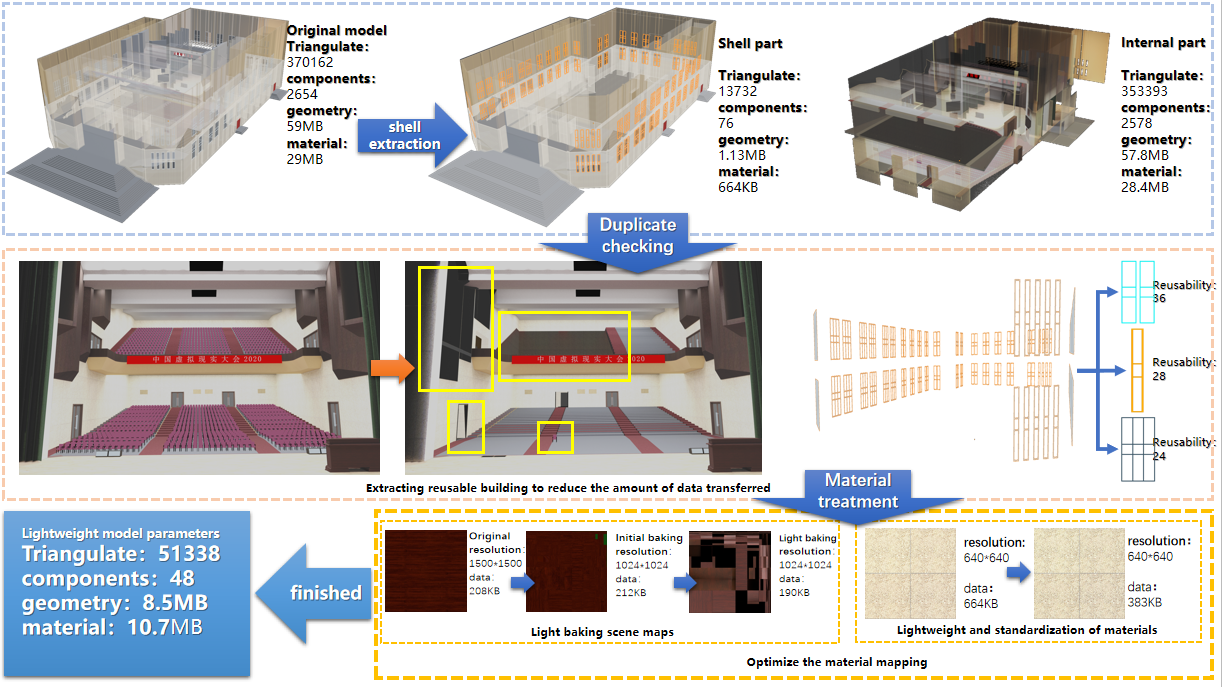


图 2 **静态会场场景的轻量化预处理**

在本项目案例中，由于场景数据量较大且模型结构复杂，会对模型传输与渲染带来很大的阻力，为例能够突破这些阻碍，因此就需要对模型场景进行进行轻量化处理：

1．对于模型材质的轻量化与标准化：由于在本案例中无透明材质的需求，因此需要将贴图中所带有的透明通道进行去除，以达到模型轻量化与标准化的目的，大大降低材质文件的数据量。

2．对于室内场景材质的光照烘焙处理：为加强模型的观赏性和轻量型，因此需要对模型进行光照烘焙。光照烘焙可以加强模型的观赏性，且由于模型材质并非会占据整块贴图，因此大大提高了贴图的利用率。

3．对于模型构建的查重处理：由于场景建筑模型中存大量重复构件，包括门窗、室内摆设、礼堂座椅等。因此通过查重处理，系统会将重复的模型构件仅保留一份作为样本进行传输，并在完成传输后，在网页前端进行重用构建，以达到还原场景的效果。

通过上述处理，将原本有370162个三角面片构成，数据量约为80MB的模型文件。轻量化为仅有51338三角面片，数据量仅有约19MB的可传输模型数据包，轻量化非常效果明显。以场景中的座椅为例，原始场景中共有2252把座椅数据量约28.5MB。通过上述处理，系统仅保留了一把座椅作为重用样本，因此传输数据仅有10KB

4.2**大规模参会人群的轻量化预处理**

1.模型资源的分级复用。大规模人群渲染需要存储的信息可以分为三级。第一级，所有对象都共用的数据，如模型的网格信息（网格点位置，UV等信息）；第二级，选择性使用的数据（如纹理贴图），这类数据每个对象只需要其中的一部分数据（如模型贴图有多套可供选择，但每个对象只使用其中的一套）；第二级，描述对象的参数信息（如，贴图类型，动画播放速度，高矮胖瘦，色调），这些信息每个对象都可以不同，所以每个对象的这些参数信息都要单独存放。

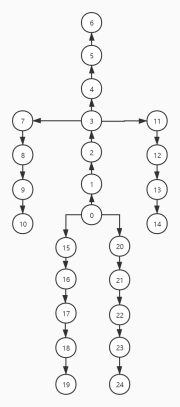


图 3 人物模型骨骼结构的分析（未完成）

2.增大临近人物对象间的差异性

（1）需求

为了使得人群渲染的多样性效果更加明显，我们希望使用相同头部贴图的模型距离越远越好。

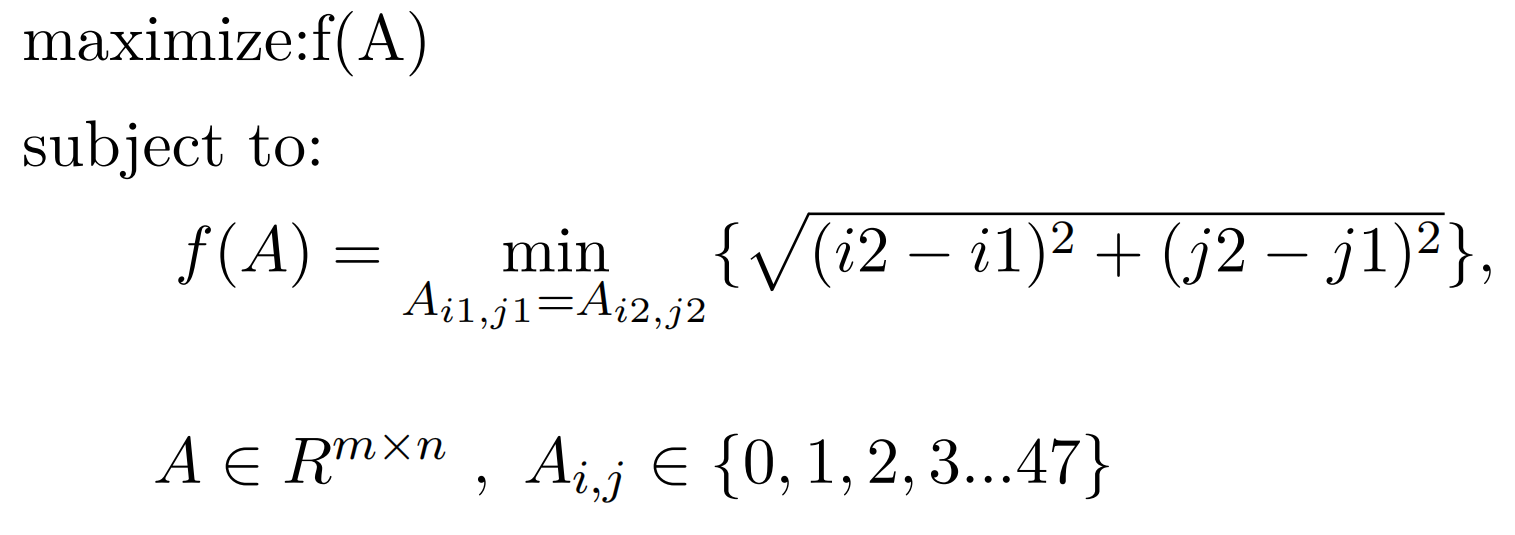
（2）问题分析

可以将这一需求抽象为以下数学问题：

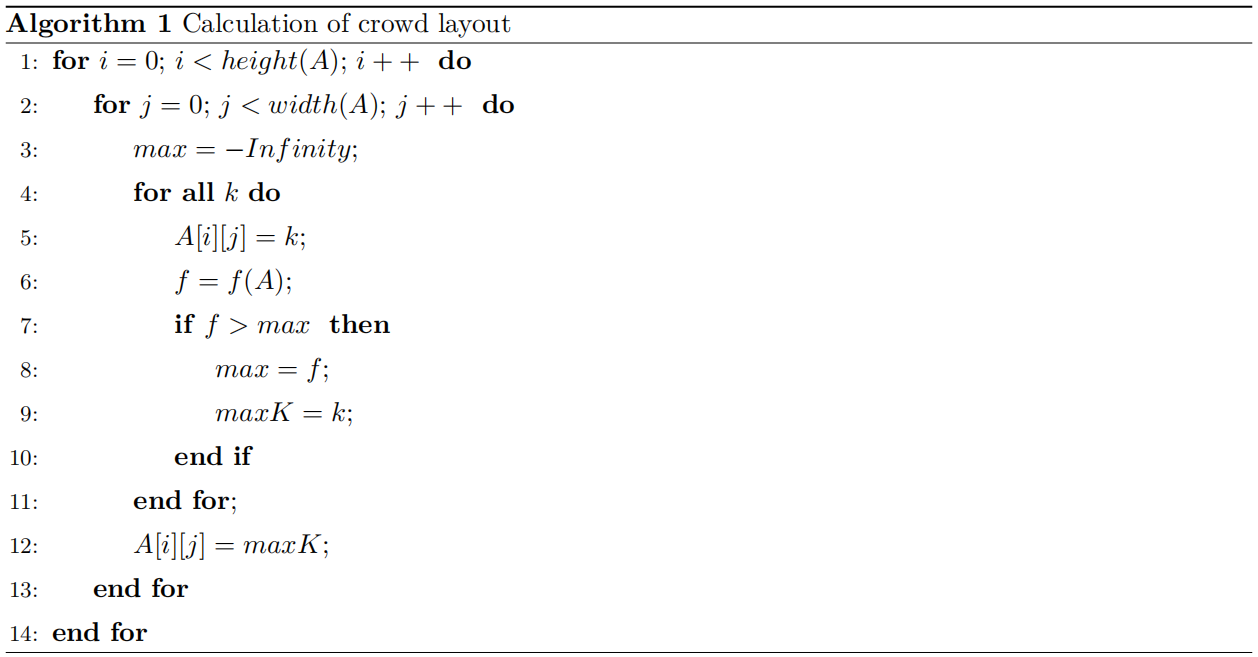
人和椅子位置相同，椅子的摆放为规则的矩阵，可以用一个长宽固定的矩阵A记录信息。头部贴图共用48种，可以用0-47这48给整数表示矩阵A中每个元素。

我们定义一个叫**分散度**概念，用来描述相似对象的分散程度。分散度的值为相同类型相同的元素间距离的最小值。我们要求一个最佳的人群设置方式A，使得分散度最大。

可以将这个问题表示为下面的公式形式，其中矩阵A为人群设置方式，f表示分散度，i和j表示矩阵的行号和列号：



（3）解决问题



（4）优化效果

我们进行了1000次不经过处理的测试，分散度f的值始终为1，既存在相邻的两个人物对象贴图相同。经过我们的处理后分散度约为5.831，所以在我们的场景中任意使用相同贴图的人物对象之间的最小距离是5.831，他们之间相隔4人以上。

1. **关键技术之二：细粒度化渐进式传输调度**

将场景中需要的数据细化拆分，相同部件只传输一次。

分级传输，提高初始加载速度。对静态场景的烘焙贴图和动态人群的纹理贴图都进行了分级处理，先传输分辨率低的贴图，再传输分辨率高的贴图。

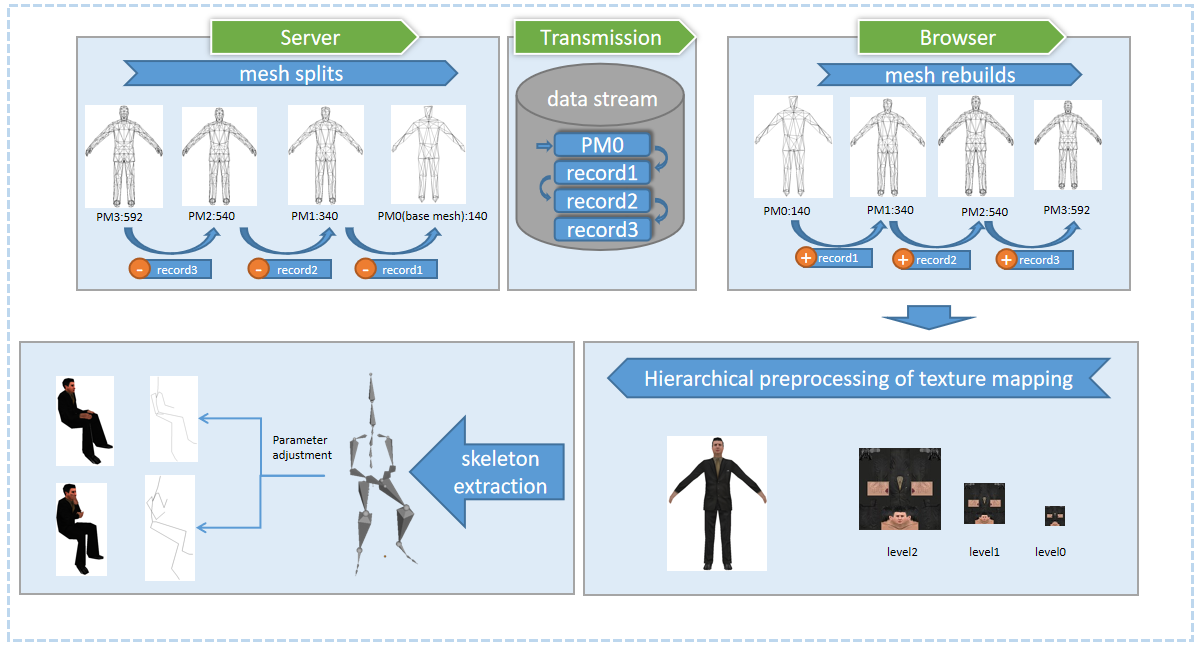


图 4 **细粒度化渐进式传输调度**

5.1轻量化内存管理

为了确保复用的人物对象资源不被重复存储，需要将场景中的人群统一管理，我们用JS设计了以一个对象来管理整个的人群。

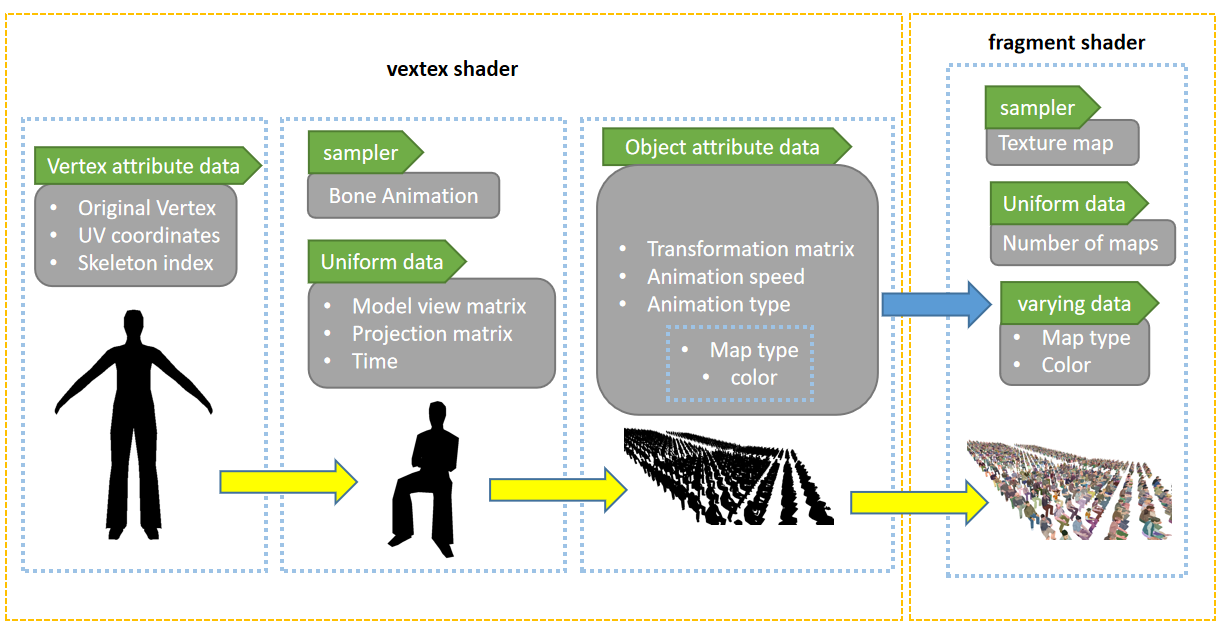
5.2基于兴趣度的细粒度化在线打包

将3D场景资源拆分为小的组件，进行细粒度化传输，通过兴趣度分析各个部分的重要性，求出每个单元的兴趣的，按照兴趣度由高到低的次序来依次传输这些单元对应的资源，从而确定这些单元的传输优先次序。

5.3 带宽自适应的渐进式传输调度

渐进式网格的增量数据被划分为多个数据包，客户端浏览器只有接收到上一个数据包后，才会请求服务器发送下一个包。当网络质量较高时，每个数据包的传输时间更短，客户端请求的频率更高，服务器发送数据包的频率也就更高；当网络质量较差时，每个数据包的传输时间更长，客户端请求的频率更低，服务器发送数据包的频率也就更低。

**6 轻量级大规模会议场景在线渲染**

图 5 大规模人群的轻量化**在线渲染**

首先将模型划分为多个区域，每个区域可以进行不同的操作，这样就可以使得搭配更加多样化。如果为每个区域匹配不同贴图，就可以通过不同贴图搭配各种效果。另外，可以通过对人物对象的高矮胖瘦进行设置，还可以对对象的色调进行编辑，在骨骼动画方面，可以为每个人物对象设置不同的动画播放速度。

为了提高渲染效率，所有的人物在渲染时作为一个整体，一次性将人群所需的全部数据输入着色器。这就需要我们管理好每个人的所有参数信息（位置，大小，贴图类型，等等），渲染时一次性将这些信息传入shader。

项目中的优化处理：

（1）设计了专门用于存储骨骼动画数据的浮点数格式

骨骼动画数据的输入格式是32的float浮点数，经过测试发现骨骼动画数据实际上并不需要这么高的精度，所以我们设计了一种16位的浮点数格式，这样就使得传入着色器的骨骼数据量减少了一半。

（2）利用人物的贴图的对称性

项目中人物的贴图左右对称，所以可以只传入着色器左半部分贴图，这样传入着色器的贴图数据量就减少了一半。

