我们那个项目主要是采用地图分块的方式去加载地图，是先写好地图切块的工具，然后让美术那边去对地图进行切块和地图资源商的分配，美术那边做好之后再发过来写逻辑。

我们分块之后如果地图出现缝隙的话，可能是因为贴图边缘没有处理好或者相邻的地图块之间没有对齐，如果是贴图的原因，我们会去检查UV映射，如果是地图块之间没有对齐，就在地图块的边缘增加一些重叠的区域。

我们地图块的管理方案是采用九宫格分层进行管理，一共有两个层，一个是显示层，一个是缓存层，显示层显示的是脚下的地图块和脚下地图块周围的八个地图块，除了这九个地图块，其他的地图块就在缓存层里，当我们控制人物行走的时候根据人物的位置去计算出需要显示的地图块，要显示的地图块从缓存层里去调用放进显示层，显示层里不需要显示的地图块放进缓存层里，当时我们有一个方案就是我们会把不需要显示的地图块资源直接卸载掉，后来没有选择这个方案，因为地图块占用的内存不是很大，可以接受，并且也不用频繁的去读取资源，所以就没有用卸载的方案。

地图上的资源是采用四叉树去进行管理，就是把整个地图当成一个根节点，在这个节点上去划分成四个节点，如果划分的节点中有资源的话就继续划分下去，没有的话就不用划分节点，然后如果有模型太大的资源的话，就按照中心点去划分，中心点在哪个节点就把这个资源划分到那个节点去。资源划分好之后每个节点会有一个包围盒，玩家碰到包围盒之后会显示对应节点的资源，但是这样的话会有一些bug，比如说玩家碰到节点的包围盒之后会在玩家面前突然显示出来节点上的资源或者是有的资源会横跨好几个节点，所以我们之后的方案就是在资源身上去挂载包围盒，玩家身上也有包围盒，当玩家的包围盒碰到资源的包围盒之后，通过四叉树找到碰到的资源进行显示。

最后是地图优化这块，我们可以使用LOD技术去进行优化，改变一下模型的精度，较少定点数和三角形的数量，减少CPU的负担；我们还可以再玩家进行远处观望的时候把远处的资源或者地图块改成一个2D的贴图，等玩家走近了再改回来，在玩家走近的过程中，我们可以使用MipMap技术去把这个2D贴图的精度进行变化。这是地图优化的方案。