# Terrain generator :

## Prototype :

1. UML Class Diagram for terrain manager
2. Data structure selection (for voxel storage) and explanation
3. Simple voxel terrain generator (1 thread, 1 chunk)
   1. Surface nets
   2. Dual Contouring
   3. Transvoxel (that can deal with different neighboring LODS)
4. Voxel terrain generator optimization (Multithreading)
5. Storing and Loading of generation data (function, noise, or even compressed raw voxel weights)
6. Chunk management (player-based)

### Chunk management

Each chunk is generated independently of others, and because there will rarely be a need to generate multiple of them at the same time, parallelism should be done locally, in each chunk generation. The generation consists of 2 steps : the computation of the vertex position, and their triangulation. For the first step, compute shaders can be used to speed up the process. For the second step, it is to be decided wether jobs or compute shaders with (W)R compute buffers can/should be used. It is a more complex step because the triangulation is based on a common shared resource.