# Sunlight generation

## Lightning in Xbuilders 3

* We need to wait for all neighbors to be loaded before sunlight generation (skipping neighbors that won’t be generated on the top and bottom of the world )
  + We could also wait for all neighbors before loading chunk due to the creation being the first step.
* We need to implement the sunlight algorithm just like in Xb2.
* We need to test the mesh generation with light to see if we need to multithread the mesh generation algorithm first
  + We don’t need to do that.

# Light rendering

Let’s determine how mine-test handles light rendering

* <https://wiki.minetest.net/Light>
* <https://minecraft.fandom.com/wiki/Light>

## How does Minecraft render light?

* Minecraft renders light by setting the color of vertices to match the light values of the chunk, so yes. Minecraft bakes lighting into the chunk mesh
* Entity lighting is determined by the light of a voxel at the entity's feet.

## What is the best way to render light for performance?

<https://www.reddit.com/r/VoxelGameDev/s/Y0fE0D7og3>

* The clear winner is a texture lightmap. However, implementing a texture lightmap that is performant and memory efficient is a **major challenge** compared to just baking it into a mesh.
  + In order to make sure that the texture map is efficient, each chunk gets its own 3d texture map. That’s because if we were using a world map, we would have to send the entire thing over when the map shifted.
    - So large entities still can't have a lightning gradient with this configuration, unless there was a way for meshes to load 9 textures or more at the same time. Sounds like a big pain, in of itself.
  + The texture map must be memory efficient, there are lots of challenges to making that happen.
  + The texture maps must be multithreaded.

## My Reddit posts:

* <https://www.reddit.com/r/VoxelGameDev/s/VzGsqnbgFR>
* <https://www.reddit.com/r/gamedev/s/OQaC7HqIok>

# Sunlight generation for infinite world height

A screenshot of a graph

Description automatically generated

Sunlight generation must happen before the mesh gets rendered if you are baking light into the mesh.

* + All sun values should be 0 on initialization
  + Neighboring chunks are just chunks that are sharing a face

**Generate the sunlight immediately, taking nodes from our future chunk (if it exists).**

* + If the top neighbor is loaded, propagate sun from the top chunk’s bottom layer
  + If top neighbor is not loaded
    - if the top of the chunk is above the ground, propagate sun from our top layer
    - If the top of the chunk is below the ground, skip it
  + If nodes go outside of our chunk
    - If the chunk exists, put them in a cache in the chunk
    - Otherwise place them in the future chunk
  + Regenerate the below chunk if it already had sunlight

**If this chunk already has generated sunlight**

* + If the above neighbor was generated with opaque objects or had opaque objects placed in it
    - Propagate downwards all sun values from the bottom layer of that neighbor that are less than 15 in value,
    - Repropagate light from the edges of the darkened area of this chunk
    - The next chunk will continue the shadow downwards

**Let's wait for all neighbors to have loaded terrian before doing sunlight**

* + Its simpler to do it this way
  + It will save on performance because we don’t have to recalculate for every chunk

**Let's wait for all neighbors to have loaded terrain before creating mesh**

* + We have had several artifacts from meshes that generated prematurely
  + We already would have to regenerate the mesh when all neghbors have terrain
  + It saves preformance and simplifies things