**Chunks**

- Divide the crack length ways into short (~500m?) chunks
- Each chunk can be represented by VB in mem
- Means distant chunks can be updated less/rendered in less detail
- Different modes for rendering chunk:
    - When a crack fills a chunk, all that needs to happen is the crack expands. So the VB contains attributes for two positions for each vertex, and a uniform parameter is used to lerp between the points and produce a smooth animation without reuploading VBs
        - These chunks should be offset from each other so they dont all swap at same time
    - When the crack has not reached the chunk, can render static (no need to reupload)
    - When the crack is opening in the chunk, will potentially have to reupload crack structure on the fly (or recalc with OpenCL) this is more intense but only needs to be done for one chunk
- Since we know what the chunk should look like next, can compute and upload next VB before needed.
- Chunks could also be used to query crack data from data store

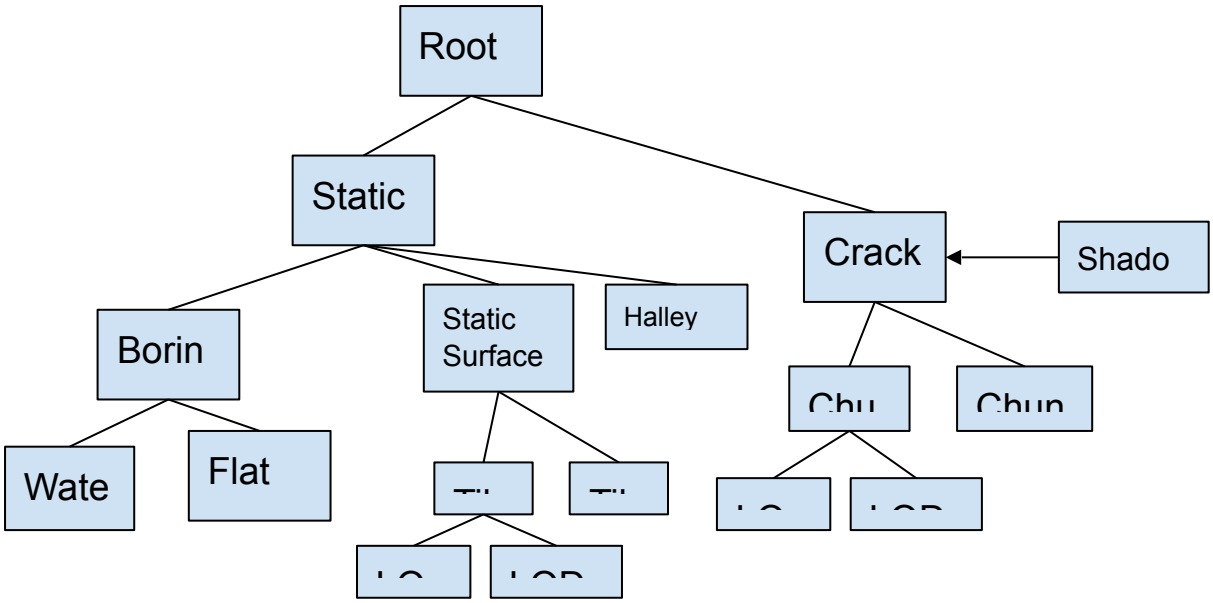
**Alternative; direct**

- Generate each scene as one VB for each frame
- Pretty expensive but easier to implement

**Ice surface**

- Bump mapping using textures produced from perlin noise
- Each vertex has attributes that help pick texel from bump map; calculated from distance along crack within chunk and distance down crack
    - This will make the texturing seamless
- Will have to find a way to give that glowy appearance
- Shadow mapping fairly trivial

**Scene graph**




**Tiling**

- Whole scene represented by tiles over large square area
- A tile is either water, static ice, boring ice, part ice, unmapped (crack)
    - Water - No scenery needed, just a flat water texture
    - Boring ice - Flat ice texture
    - Static ice - Ice for which we have surface maps, models (height map matrix) must be downloaded
    - Part ice - tricky one. When ice boundaries against water, need to map the ice surface, but also indicate where in height map the ice becomes water. Also use for crack, indicate where static scenery stops and crack scenery begins (this could be unnecessary if that boundary is always at a tile boundary)
    - Unmapped (crack?) - Where other scenery exists (e.g. crack)
- Models attached to tiles and added as additional scenery
- Tile downloading:
    - Tile height matrix can be simplified (easiest technique is just to remove nodes, e.g. simplify 3x3 to 2x2 reduces triangles by a factor of 4
    - Initially low detail tiles are downloaded. Improving the quality therefore only requires downloading the remainder of data for that crack
- OpenSceneGraph can handle choosing the correct tile LOD model for the distance, but we'll need to handle unloading/loading models for various distances
- Seems fairly acceptable to have low quality models in the distance especially in the case of ice.
- Unloaded "Static Ice" tiles can be represented in the same way "Boring Ice" tiles are