**Smoothing**

* Each tile will run a list
* Top tile of the list will always be ground or water. Whether the ground is grass/ Dead grass / snow etc
* When smoothing we always enter smoothing as the 2nd and subsequent tile. See drawing
* So each list is a list of GroundLayer. The groundtype references pre made ground type. They have to be premade for memory consumption
* Each ground type, could be layer 0 and this orientation. So we would have have a bunch of different ground types for different orienations
* So each ground type will need a smooth type. However we need to go a positive smooth type, not a negative one like the river. This does make it easier to make smooth types. Also because we stack smooth types we only need 8 tiles of smoothing for each tile type. (Why couldn’t we stack with water? I don’t know. Oh well)

**Ground Sprite**

* Create a separate sprite for ground GroundType. It sucks we need a separate layer from water smoothing and for roads. Groundtype is whats referenced in the list in each tile. So you would have groundtype Grass, int bitmask index. Groundtype GrassSmooth, int bitmaskIndex.
* So we still use bitmask indexes to reference them. However only the water will use multipliers. Instead it is going to be reversed however. Looking at the neighbour to the right. If it’s a different groundtype, we translate to that neighbour. That neighbour then adds the bitmask index of the tile to the left and adds that smoothing tile to itself.
* Improve smoothing with support for different ground textures. So on all edge borders add another type.

WAIT WE COUNT FROM HIGHEST TO LOWEST WHEN ADDING BITMASKING. THEN WE CAN JUST ADD IT TO THE END OF THE LIST AND OTHER REASON SE PICUTER

**Multithreading**

* Run input and most (Vehicle etc) game logic in the main thread
* Run fire spread in a separate thread. When vehicles interact with the fire, they will set on the fire how many liters \* gametime is being dumped on the fire. The fire update thread will read this number and work out a spread.
* At the start of the frame, call the separate thread update method
* Do not complete update until the fire spread thread is done
* Make a thread pool. This pool can be used for loading (creating Perlin noise) and then also for calculating path finding. We will use a queue or stack and we will push jobs onto the queue