Report virtual world generation

Utilitary

I can export several scalar field from my terrain: the Slopes, occlusion, drain, derivate (on x and y) and laplacian.

Here is the results:

Erosion

I implemented a rain erosion using the equation $\nabla h = u + S^{\frac{m}{2}} A^m + \Delta h$ (with h the height field, u the uplift constant, S the slop map, A the drain map and m a constant (I used $m = 1$)). After what I can add colors to help visualize the terrain, puting water bellow the height 0 and at every point where the drain is greater than a certain threshold (to have rivers). I then add grass where the slope is small enought and rock every where else.

There is some artefact at the sides of the terrain. After simulating the erosion, the terrain seems to be melt down. This is due to a snowball effect with the slope and drain map. At the border, the simulated water have limited choice to spread down, this create a very narrow but powerful stream that will create a deep river. This river will increase the local slope, that will further increase the drain and the power of the water stream and will further inrease the erosion at the border of the terrain. I tried to fix that by not moving the height of point at the terrain border, but it didn't work.

Civilisation

Roads between cities

Given a set of position (the cities locations), I am able to create a network linking ccities.

I start by computing the shortest path between each pair of cities (using Dijkstra algorithm on a weigthed network (from rosetta code) (weigths are computing depending on the local feature of the terrain (slope, height, water...))).

After what, I remove useless path. A path between A and B is useless if, for a fixes real α, there exists another point C such that $|AB|^\alpha > |AC|^\alpha + |CB|^\alpha$.

Streets inside cities

To generate cities, I used a Eden growth algorithm to create crossroads centers. After what, I simply link them together in the same way as I linked cities. I then walk along each street, adding a house if possible (If the house does not intersect a street or another

house, the intersection computation is done using a Bounding Volume Hierarchy (BVH) to speed up the process).

results

Nature

I can generate forest of different tree types.
In order to do that, I precompute several forest tiles using a dart throwing algorithm. Of course, I make sure that I can put every tile next to each other. Once the set of tile generated, I simply pave the terrain with forest tile and create a tree at each tree location of each tiles.

To create a tree, I compute a probability of creation for every tree types (depending on the local feature of the terrain) and create it acordingly (More details in anexes).

Anexes

BVH

I use a tree to compute the intersection between an object and a big set of objects. This tree is build using the following reccursive rules:

- If there is less than n objects (I use $n = 100$), I simply store all objects and compute the bounding box of the set of object.
- If there is more than n objects, I compute the bounding box of the set of objects and the longest axis of it. I then divide the set of objects in two part along this axis and create two children from it.

Once the tree created, to compute the intersection with an object, I reccursively check the intersection with the bounding boxe of a node and if needed I continue to check the intersection with it's children.

I can also trivialy add an object in the tree dynamically (I just compute in wich leaf the object must be and I put it here. I change the leaf in an internal node if there is more than n object).

Probability of tree

In theoriee, what I want to sample T define by:
For $i \in [\![1, n]\!]$ (n being the number of tree types), let T_i be the random variable following a Bernoulli distribution of probability (telling if I create or not this tree type) p_i. Then, we can define random variables $I = \{i \in [\![1, n]\!] | T_i = 1\} \subseteq [\![1, n]\!]$ and T such that $\forall i \in I, \mathbb{P}(T_I = i) = \frac{p_i}{\sum_{j \in I} p_j}$.

To do that, I draw a random variable r for each tree type of probability p_i. If $r < p_i$, I include i in my set I.
At the end, if I is empty, I create no tree.
Otherwise, I compute the differents probabilities $\hat{p_i} = \frac{p_i}{\sum_{j \in I} p_j}$ and draw one single random number r. Then, for each i in I, if $r < \hat{p_i}$, I stop and create the tree type ascociated to the probability $\hat{p_i}$. Else, I substract $\hat{p_i}$ from r and continue.