Kamil Kasprzak
Jakub Skalski

AI4Games project - 2021/2022

Procedural terrain generation

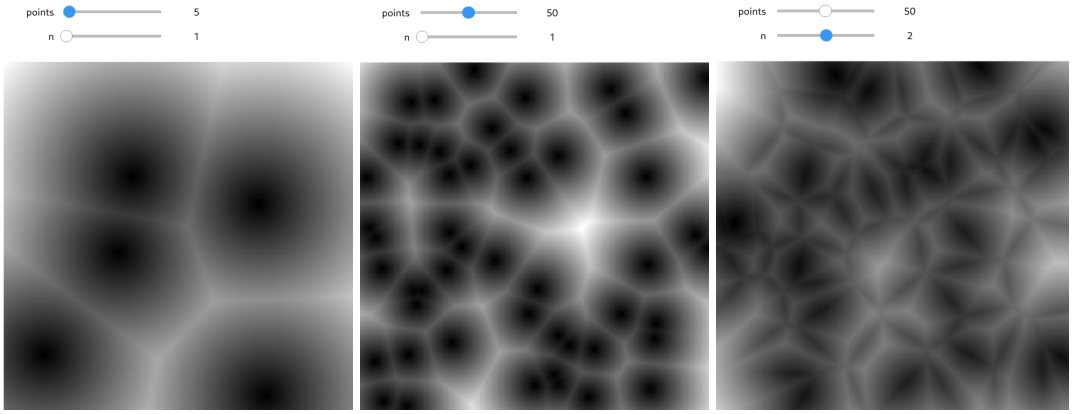
Description

The main idea was to try out different approaches and mix them together to create an interesting and variant terrain. We ended up relying mostly on noises (Simplex, Perlin, Worley and a few others) and all of the noises used with the exception of Simplex noise were implemented manually. Project was realised mostly in python with the excessive use of jupyter notebooks providing useful and interactive insight into the creation process.

Noises

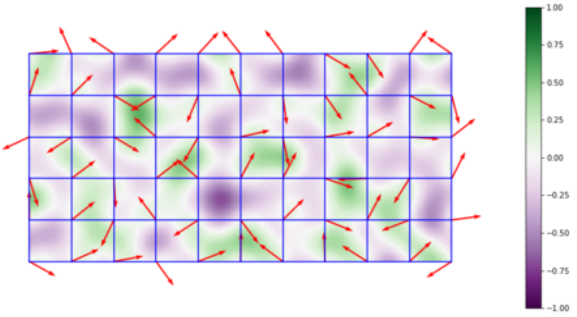
Worley

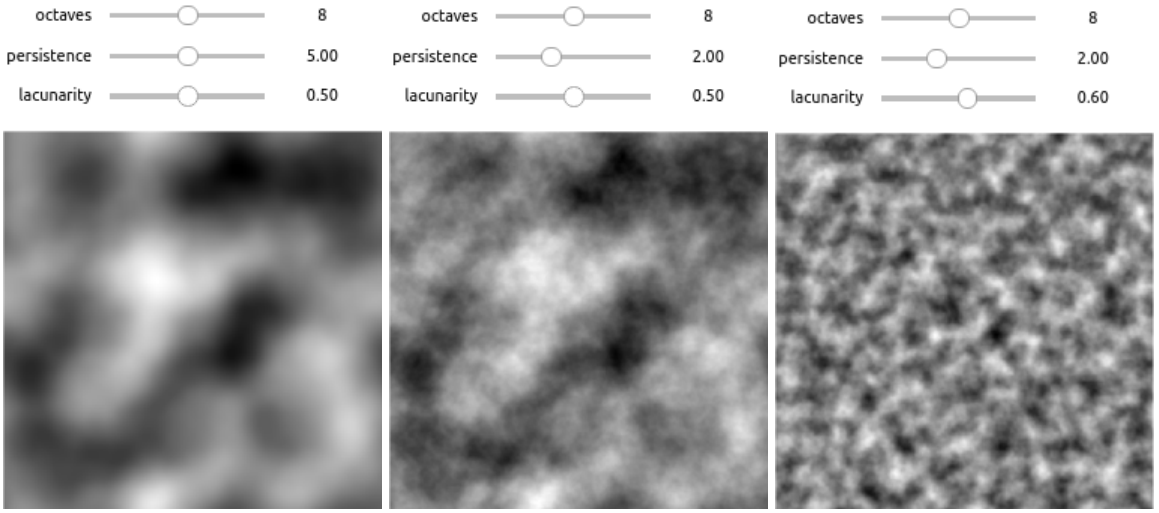
This noise is pretty simple. We begin by randomly selecting *k* points. Now we calculate the euclidean distance from each pixel to the n-th closest point. We used it as a layer for the intensity of the heightmap.



Perlin

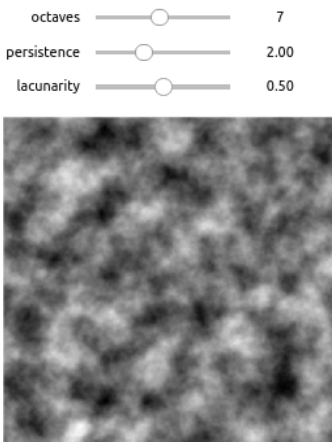
We start off by evenly spacing out unit-length gradient vectors. Then for each pixel we calculate the dot product of an offset vector from that pixel to each of its four closest gradient vectors. Finally we interpolate thus achieved values.





Lacunarity of more than one means that each octave will increase its level of detail (frequency).

Persistence determines how much each octave contributes to the resulting noise. For instance, each octave will contribute equally should the persistence be set to one.

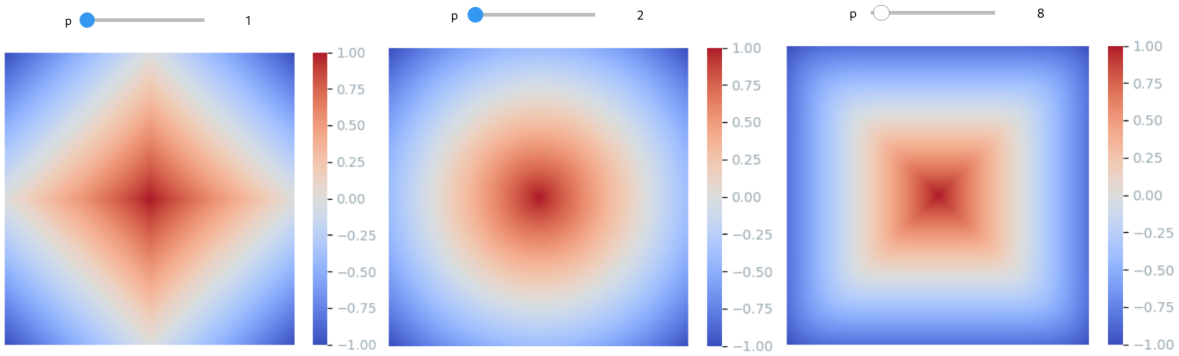


Generally speaking, Perlin noise is very volatile. That is to say that a very slight change in any of its parameters usually changes the outcome almost completely. As we will later see, finding the right parameters is the key to achieving convincing terrain texture.

Generic heat function

Another simple “noise” generator. It will serve the purpose of adding and subtracting mass in a gradient-like manner from the given shape. It uses [Minkowski distance metric](#) but is also heavily scaled so that the much greater values accumulate in the centre unless specified otherwise.

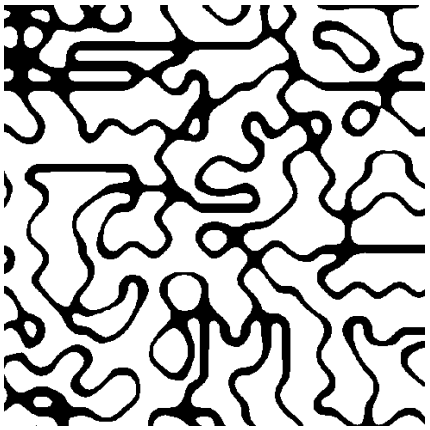
$$D(X,Y) = \left(\sum_{i=1}^{n} |x_i - y_i|^p \right)^{\frac{1}{p}}$$



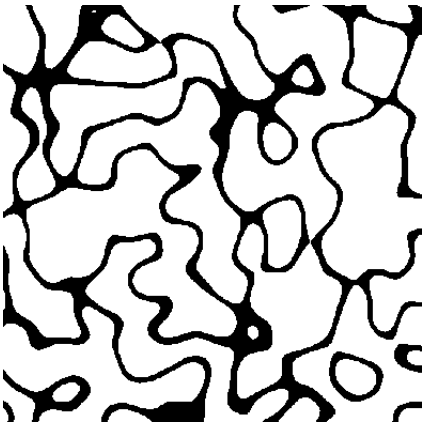
Simplex noise

Simplex noise is very similar to the Perlin noise. We did not implement it ourselves and instead opted to use a [noise library](#). The advantage of Simplex over Perlin is that the latter has the tendency to form horizontal and vertical lines which look very unnatural and thus we opted to use Simplex instead.

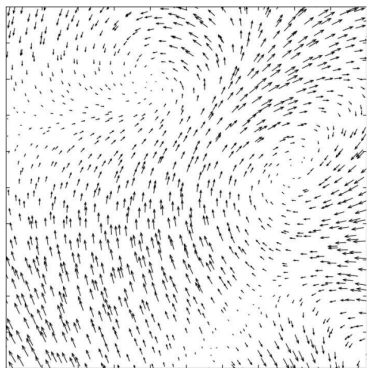
Perlin



Simplex



Vector fields

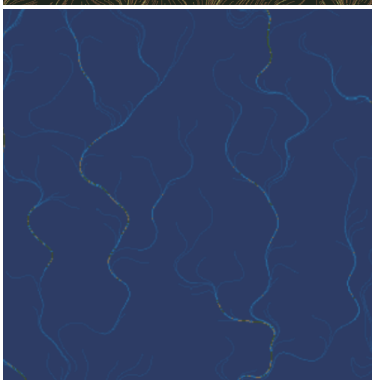


Vector fields can be created by assigning unit-length vectors to each value. Given a matrix of scalar values s_i we calculate vector $v_i = (cos(s_i \cdot \pi), sin(s_i \cdot \pi))$ for each i.

We proceed by randomly placing particles which move in the directions pointed to by said vectors. All that remains to be done is to wait for the particles to travel a meaningful distance and draw their trajectories.

Fur

This method is very potent when it comes to generating hairy looking textures especially when applied to a noise matrix as was done here.



Rivers

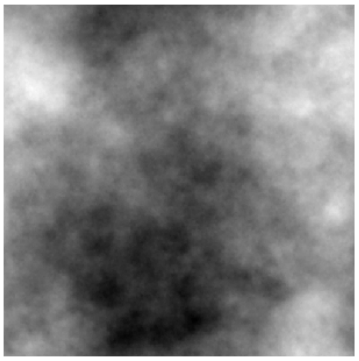
This way we would also generate shapes very much like those of rivers.

In the end it turned out that generating rivers in this manner looks very unnatural when combined with actual height maps and so we scratched that idea. Yet we do believe that it is possible with the right input matrix.

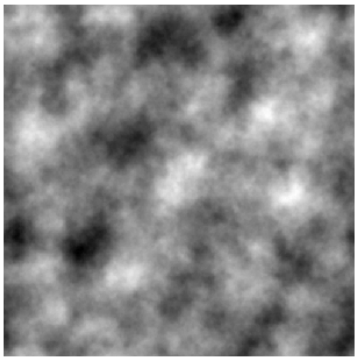
Terrain generation algorithm

1. Define terrain shape

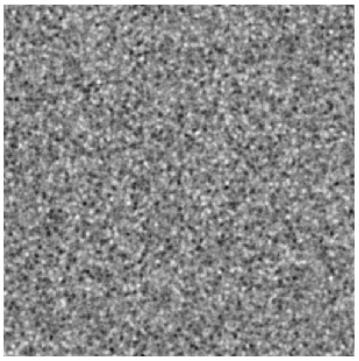
Large

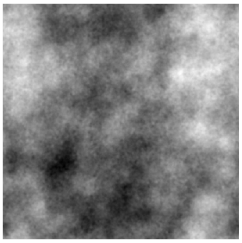


Medium



Small granules



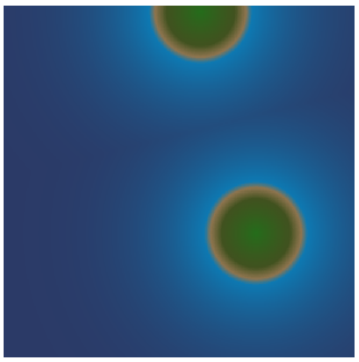


Shape matrix

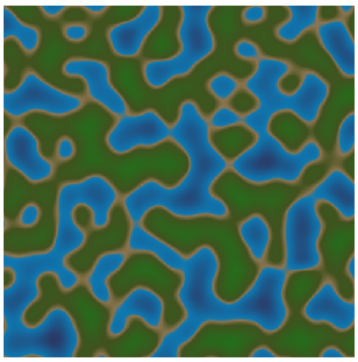
We can combine noises with different sizes of granules to obtain a heightmap that can imitate mountains.

2. Deploy clusters

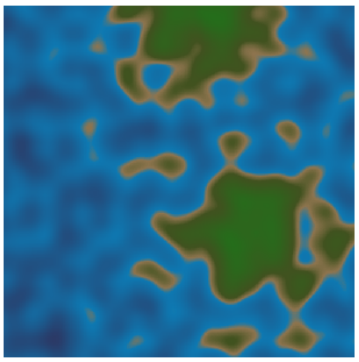
Worley clusters



Simplex noise



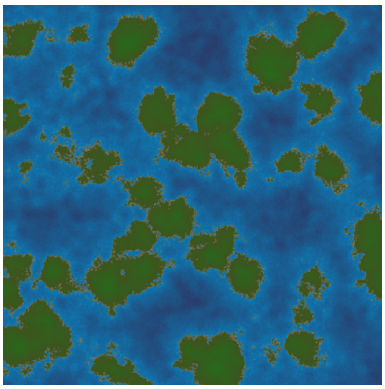
Islands



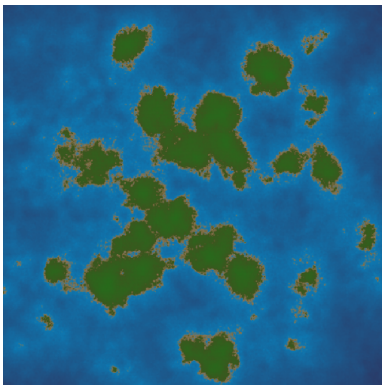
Mix worley and simplex noise so that we can exert control over the number of terrain clusters present but also retain the simplex terrain shape.

3. Erase borderline clusters

Without border



With border

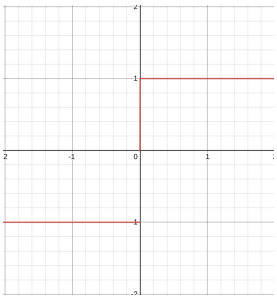


Apply heat function to focus the terrain formation around the centre of the map or use it to subtract mass from the map borders.

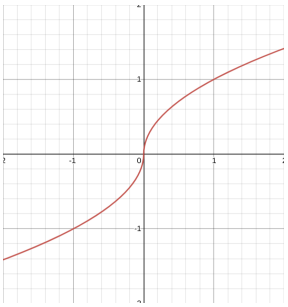
4. Beaches

We can control the sharpness of the terrain with a simple formula $sgn(x) \cdot abs(x)^b$.

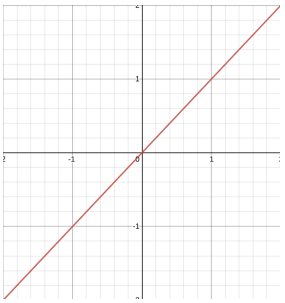
b = 0.0



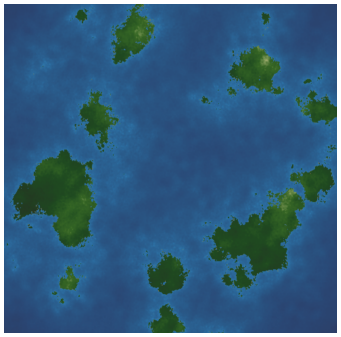
b = 0.5



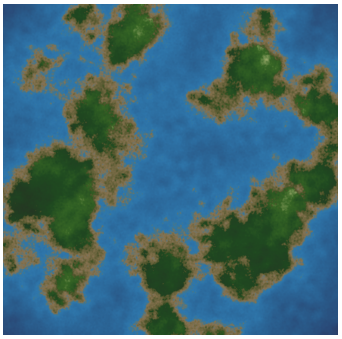
b = 1.0



b = 0.0

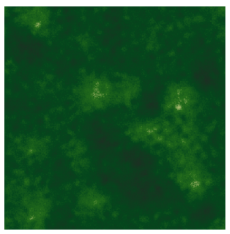


b = 0.8



5. Moisture levels

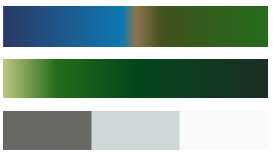
Apply some variation of noise to generate moisture levels and combine those with the terrain height.



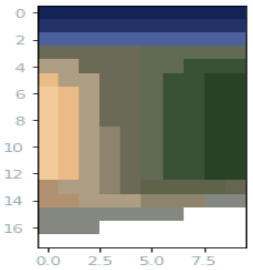
Moisture matrix

Moisture is just another take on Simplex noise with parameters adjusted accordingly.

We tested two different approaches:

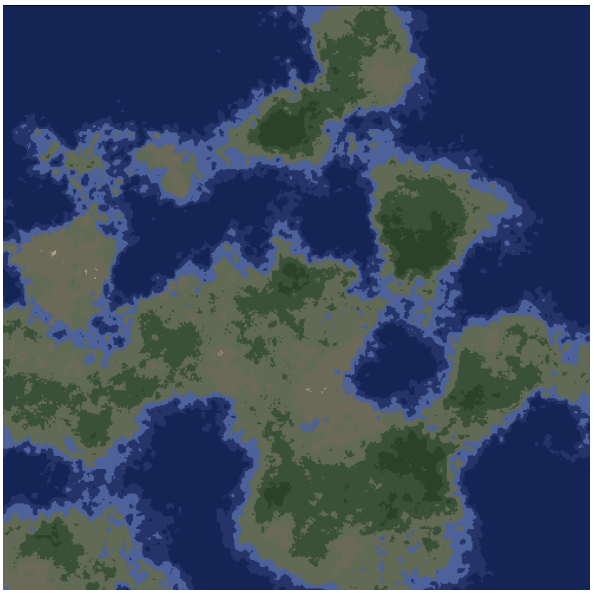


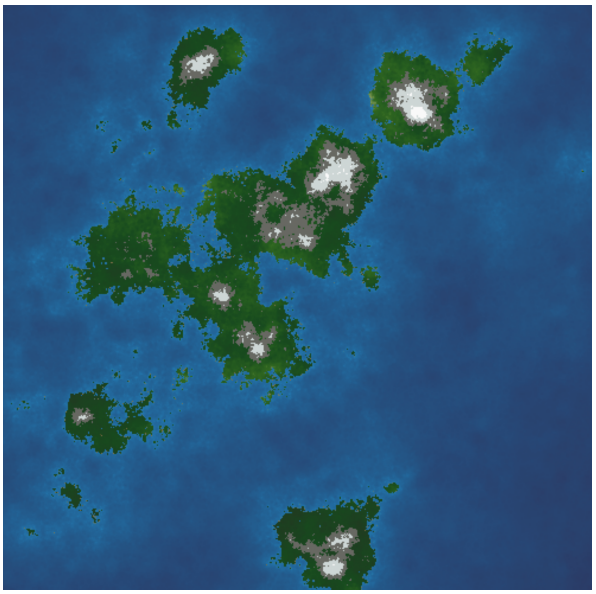
1. Dividing biomes into ocean, land and mountains by terrain height and applying moisture to obtain different values of biome intensity.



2. Combining moisture and height to obtain points from 2D arrays of biomes. The x-axis represents moisture (10 levels) and y-axis represents height (18 levels).

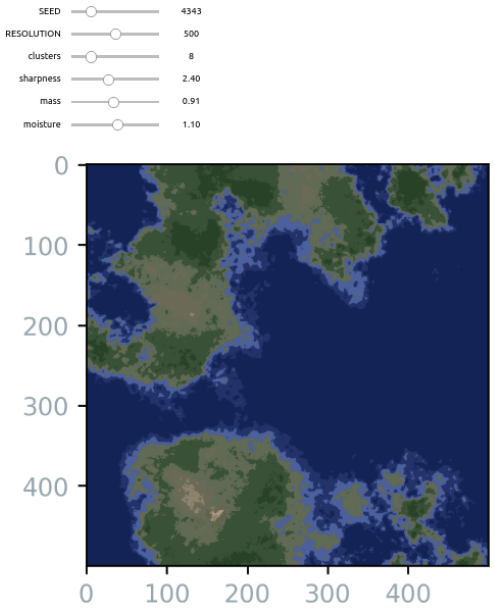
Final effects

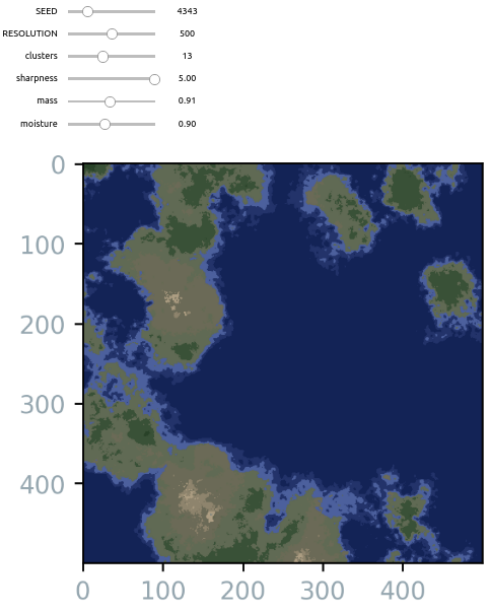


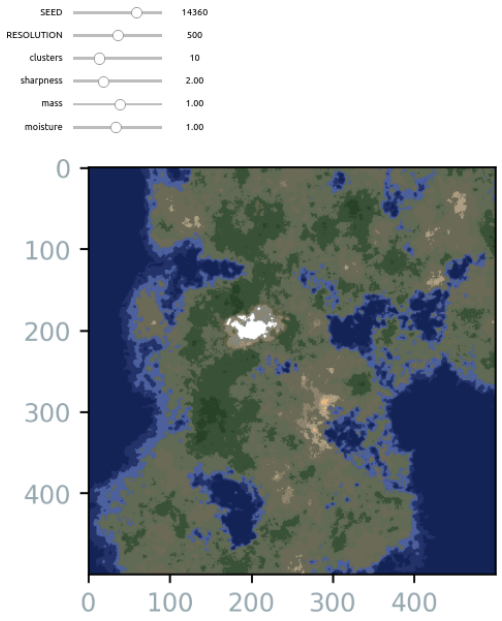


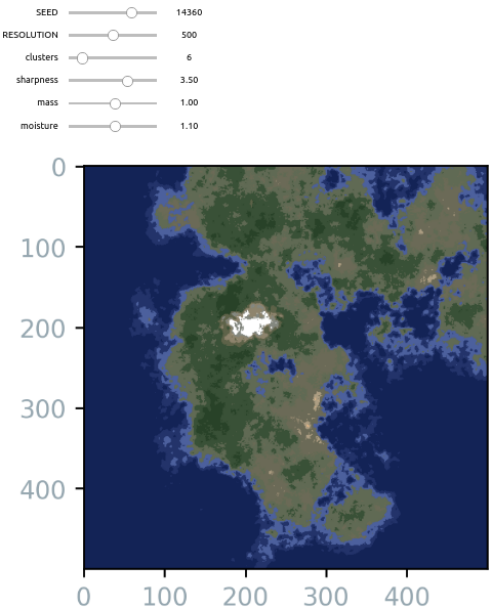
PCG Gallery

In this section we present some examples of the maps we happened to come by when playing around with different parameters. From the two different approaches we decided to choose only one as the ‘flagship’ of this presentation.

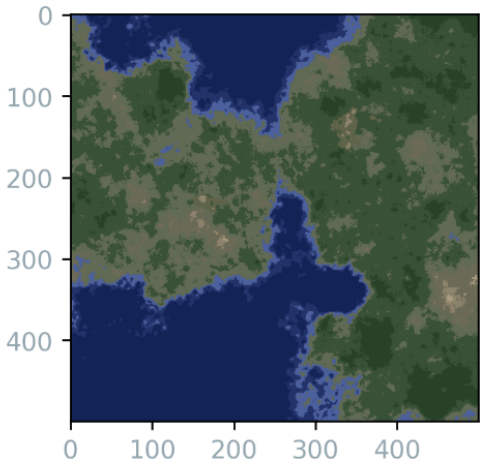




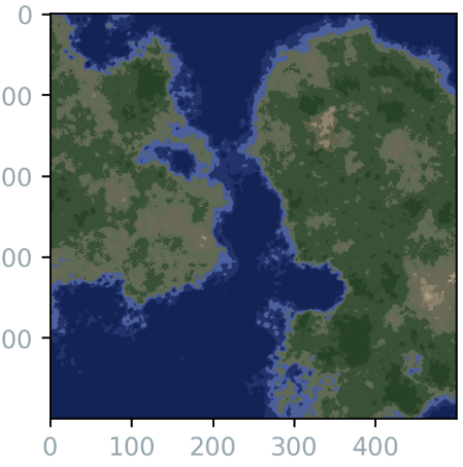




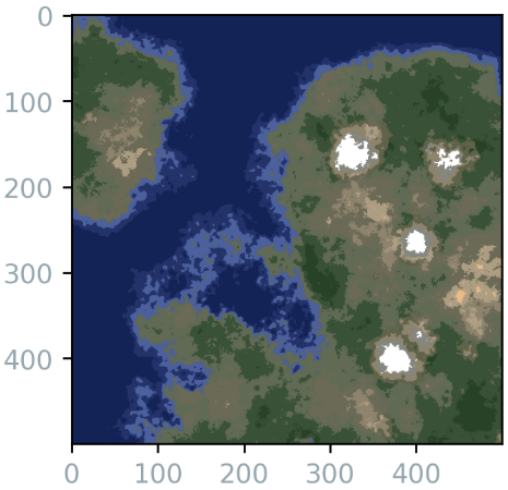
SEED 13446
RESOLUTION 500
clusters 10
sharpness 2.00
mass 0.91
moisture 1.10



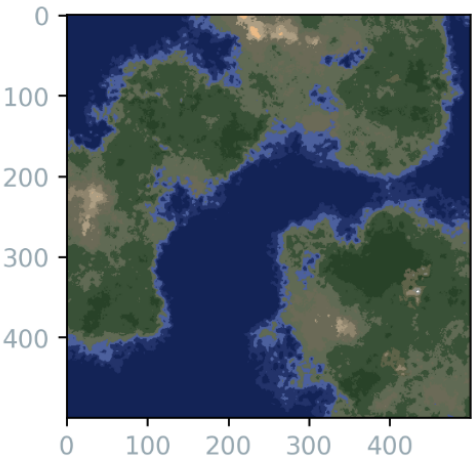
SEED 13446
RESOLUTION 500
clusters 7
sharpness 1.50
mass 0.90
moisture 1.10



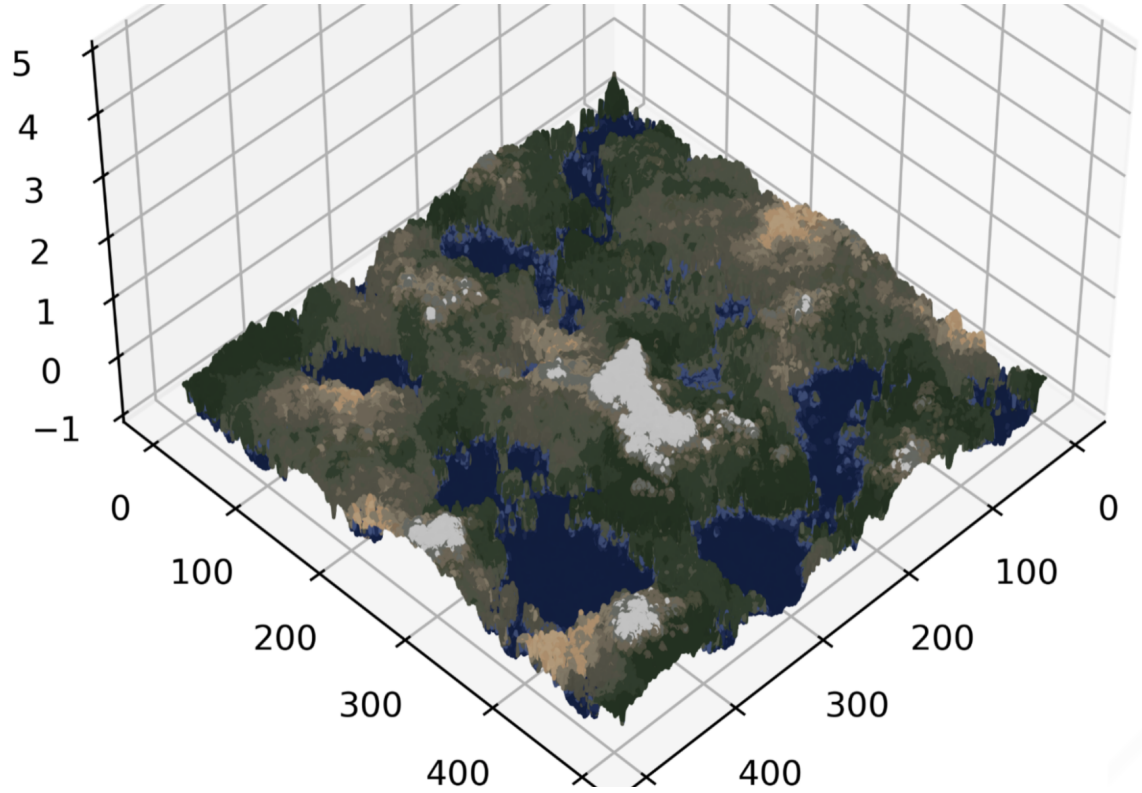
SEED 6500
RESOLUTION 500
clusters 16
sharpness 2.75
mass 1.00
moisture 1.05



SEED 6622
RESOLUTION 500
clusters 10
sharpness 2.80
mass 0.91
moisture 1.10



3D Projection



Sources

- https://medium.com/@travall/procedural-2d-island-generation-noise-functions-13976bddeaf9
- https://medium.com/@yvanscher/playing-with-perlin-noise-generating-realistic-archipelagos-b59f004d8401
- https://weber.itn.liu.se/~stegu/simplexnoise/simplexnoise.pdf
- https://nelsonslog.wordpress.com/2021/08/28/2d-noise-functions/
- https://rtouti.github.io/graphics/perlin-noise-algorithm