- Full up-to-date documentation for the asset is available online :
  - https://github.com/BenHUET/hexamap-documentation/wiki
  - The offline documentation you are currently reading will set you up with the basic usage/understanding of the asset. For advanced usage (extend the assets, model…) please consult the online version.
- Contact for support : HUET Benjamin (benjamin.huet.1992@gmail.com) or GitHub issues

Hexamap is providing **fast, extensible, procedural** world generation for **hexagonal-tile-based** projects.

Thank you for buying Hexamap :)

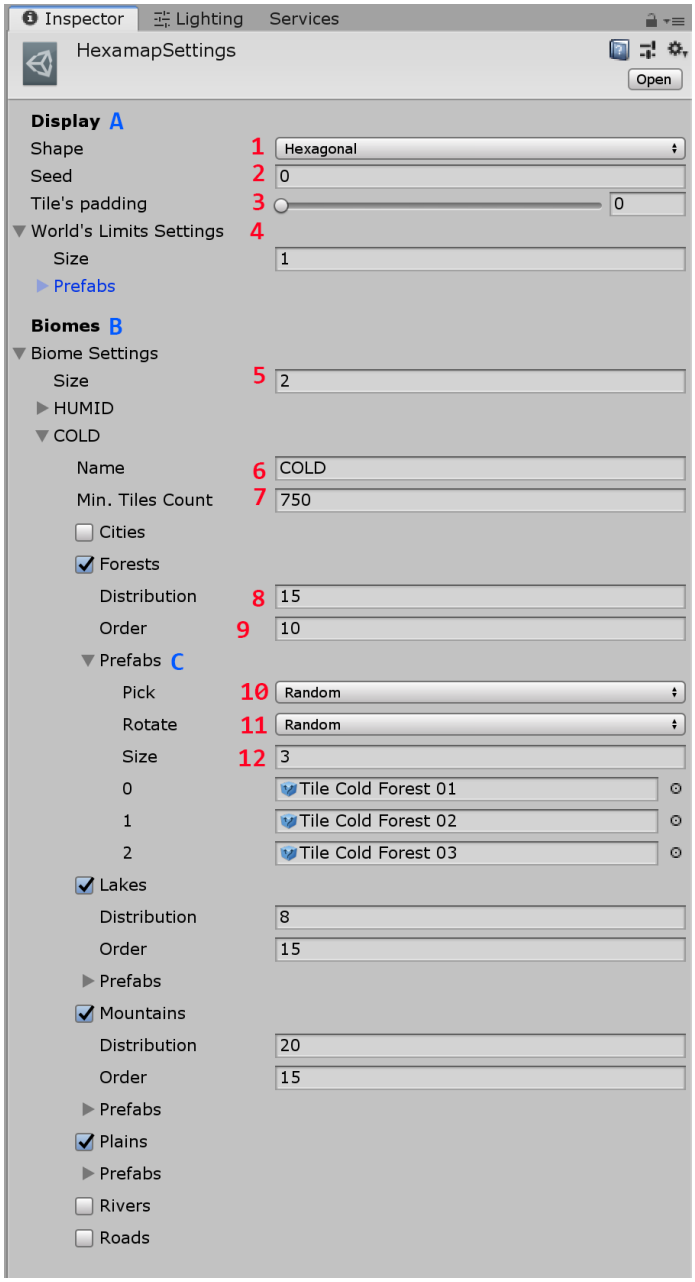
Documentation

Demo

- Grab the package from the Unity Asset Store
- Open the **Assets/Hexamap/_DEMO/_DEMO_SCENE_** scene
- Play
- Open the Hexamap settings object of the demo scene (Assets/Hexamap/_DEMO/HexamapSettings) to customize
    - You probably don't want to make changes to anything related to prefabs here for now if you are just discovering the asset
    - But you can play with biome's sizes, distributions of landforms, map's seed, tiles' padding...

The demo uses homemade 3D models... I did my best. Also, please note the demo scene will add noise to the Y position of the tiles of the map, this is handled by the DemoController and not by the core of the asset per se.

Unity Settings

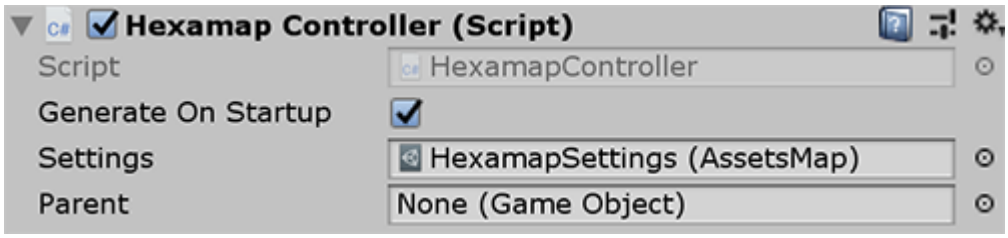


A : Display section
B : Biomes settings
C : Prefabs picking section
1 : The shape of the map. This field is populated by reflection, it is looking for classes with the attribute MapShapeName
2 : Seed (0 = random). Used by the RNG
3 : The space between a tile and its neighbours (1 = the size of a tile)
4 : Settings to customize the border of the map
5 : Number of biomes for this map
6 : Name of the biome
7 : The biome's expected size. It will always **at least** be this size, it can be more depending of the shape and RNG
8 : Distribution/Quantity/Missing (depends of the type of the landform) is the amount of this landform expected in the biome. Distribution is a percentage (0-100) and Quantity is the precise amount. Some type of landforms do not have this field because they will fill the whole space left (plains) or take as much as they want (roads)
9 : Order of generation of the landform, the less the earlier it will be created
10 : Picking method of the prefab. This field if populated by reflection, looking for classes with the attribute "PickingMethodName". It defines how the prefab for this landform will be chose
11 : Rotate method of the prefab. Can be 0, 60, 120, 180, 240, 300 or random. It defines the rotation of the prefab on its local Y axis
12 : Size of the list of prefabs below available for this landform

Controllers

HexamapController

This script is the entry point of using the asset in Unity.



- Generate On Startup : should the map be generated when this script is started ? (this will call Generate() followed by Draw() in the Start method)
- Settings : the Hexamap settings object to use when generating maps with this controller
- Parent : the GameObject to use as a parent of the generated tiles. If none is specified, the controller will create a GameObject called "Hexamap" at the root of the scene
- public void Generate() : generate the data of the map
- public void Draw() : select prefabs and position them in the scene based on the generated map
- public void Destroy() : destroy GameObjects and data associated to the map

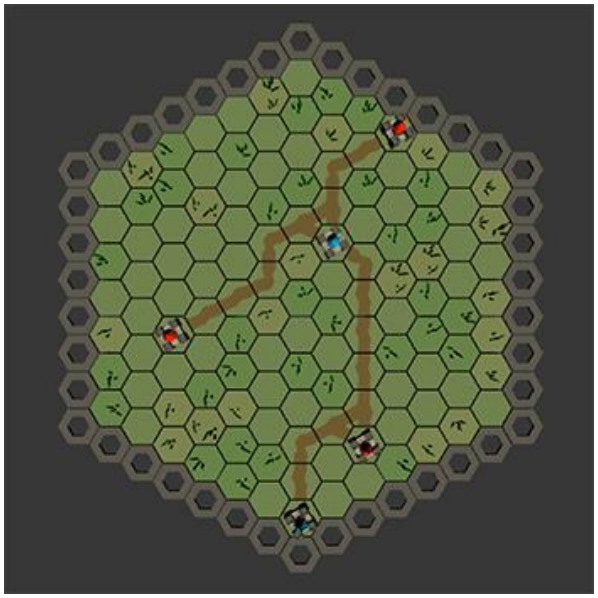
TileController

This controller will be attached to every tiles generated by the HexamapController. This the bridge between the GameObject of the tile and the library's model. You can grab the model by accessing the Tile Model property of this script.

Templates

Some templates are provided with the asset to quickly start working with it. This is also possible to create your own templates.

LandformTemplateConnection

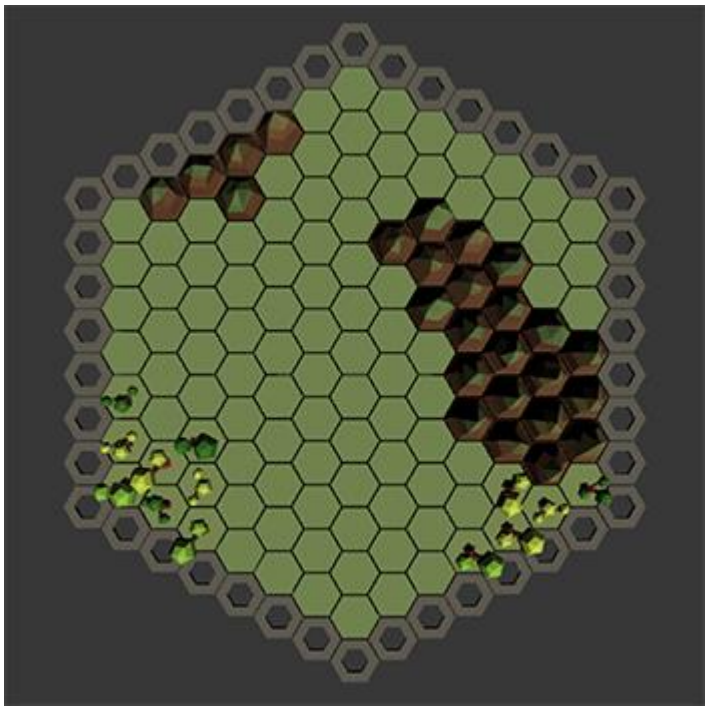


- Examples included : roads, rivers
- This template tries to connect landforms of the same type in a biome together by using the A* pathfinding algorithm
- This template can quickly blow generation times, see the "travelling salesman problem" on google
- This template should be used with the dedicated picking method "PickingMethodConnections" in order to give good visual results

LandformTemplateNoise

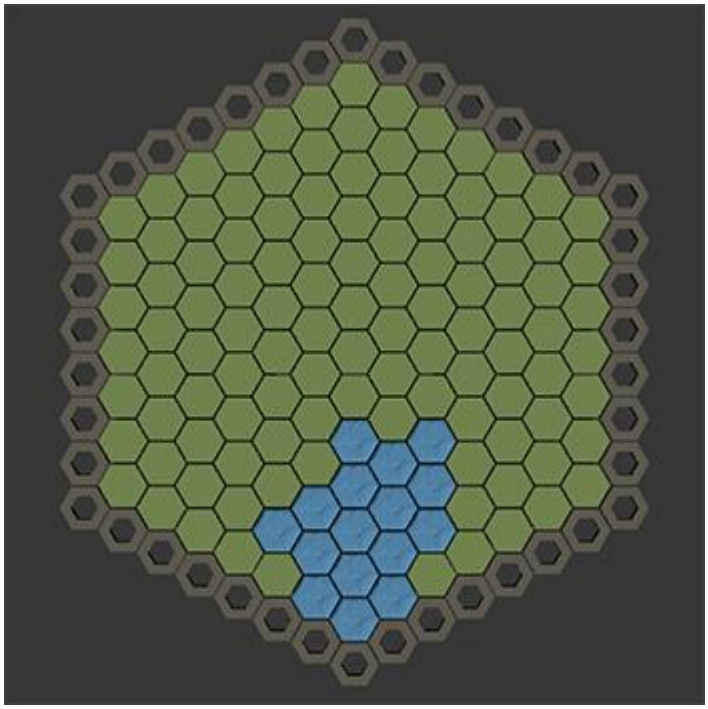
- This one is a template of templates
- Based on https://github.com/Auburns/FastNoise_CSharp
- Provide simple integration of a lot of different noises

LandformTemplateNoisePerlin



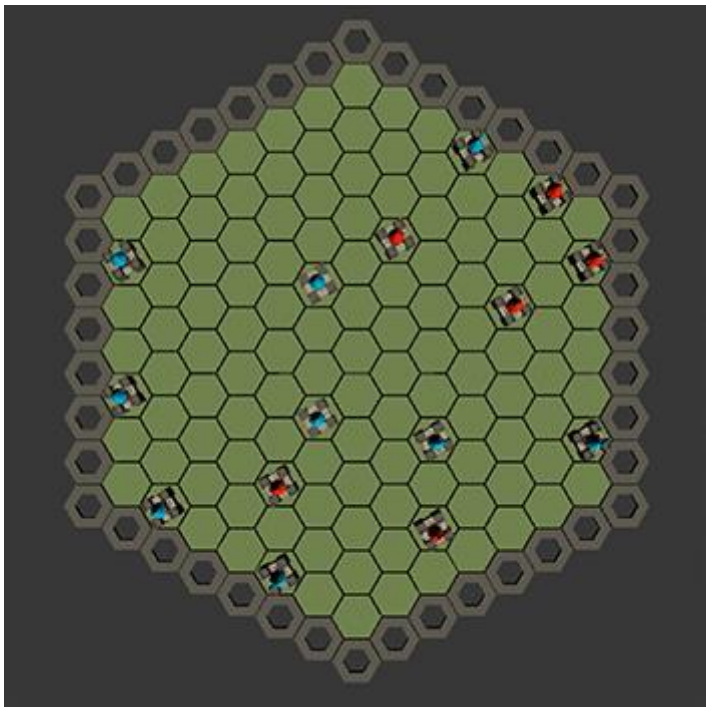
- Examples included : mountains, forests
- Based on LandformTemplateNoise
- Use Perlin noise to draw landforms

LandformTemplateNoiseCubic



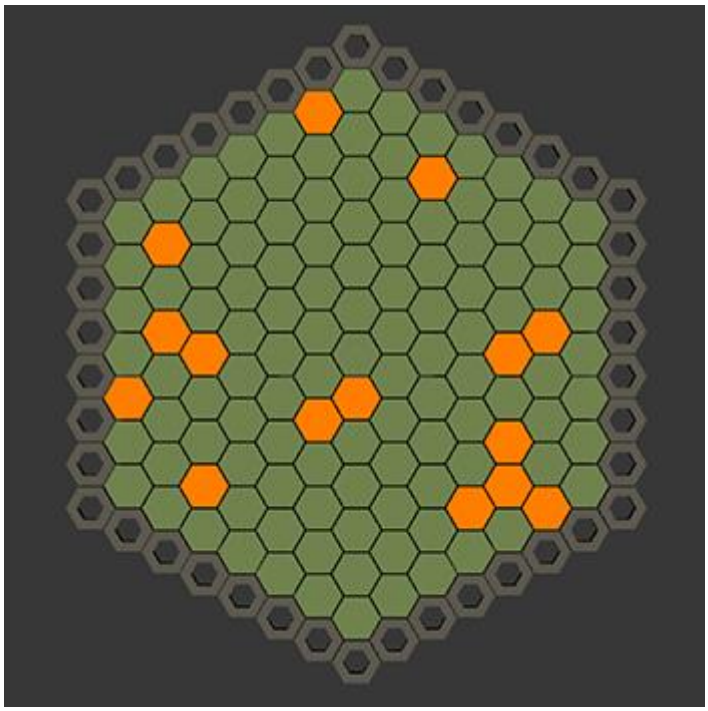
- Example included : lakes
- Based on LandformTemplateNoise
- Use cubic noise to draw landforms

LandformTemplateSpreadPoints



- Example included : cities
- Ensures every picked tile are evenly distributed

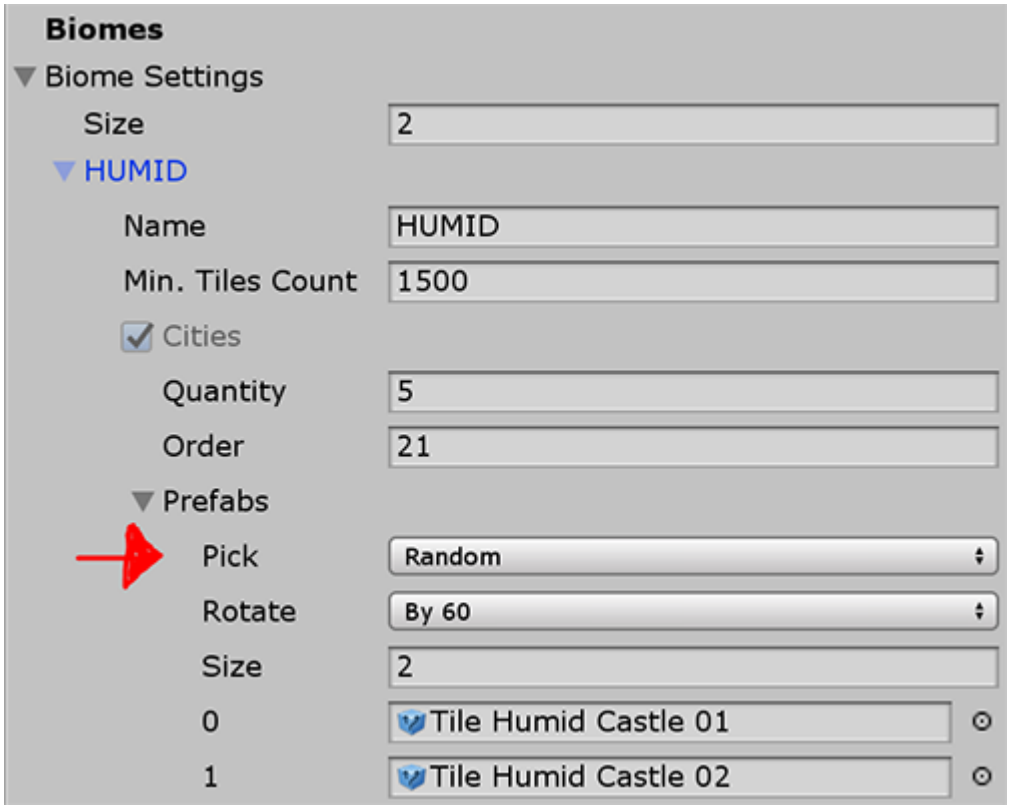
LandformTemplateRandomPoints



- Example included : none
- Totally random

Picking Methods

The picking method of a landform decide which and how its prefabs are positioned in the scene once generated.



3 picking methods are shipped with this asset.

First

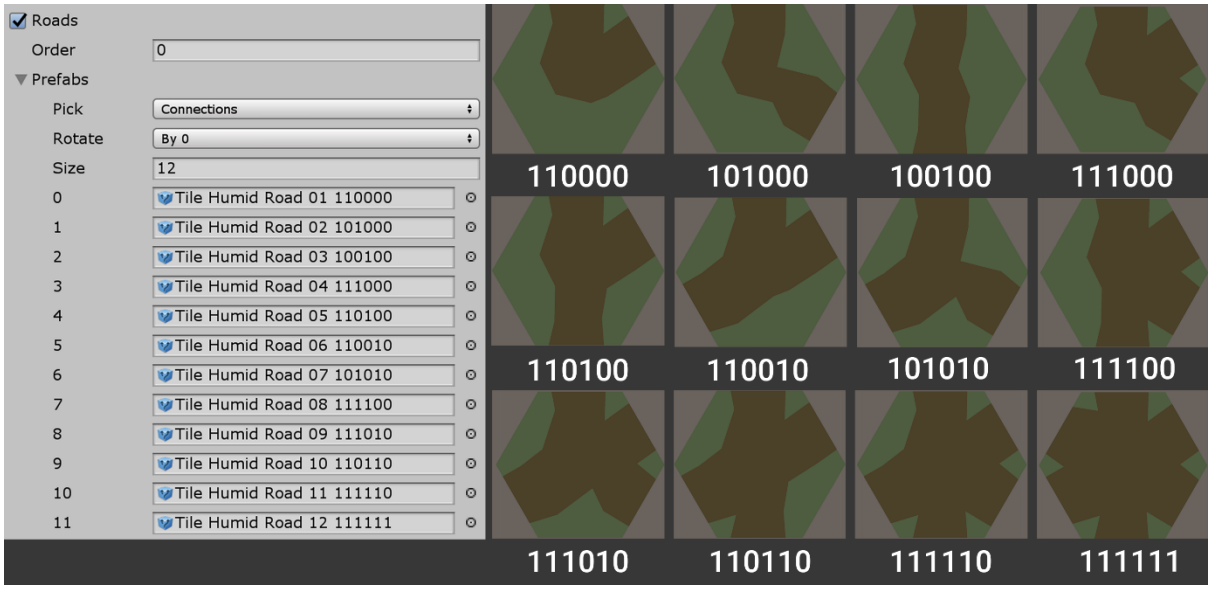
The first prefab of the list will always be picked.

Random

A random prefab from the list will be picked.

Connections

This picking method should only be used with landforms derived from LandformTemplateConnection (roads, rivers). When using this picking method, you should keep the Rotate field to By 0 since the picking method will handle the rotation.


Roads
Order
0
Prefabs
Pick
Connections
Rotate
By 0
Size
12
0 Tile Humid Road 01 110000
1 Tile Humid Road 02 101000
2 Tile Humid Road 03 100100
3 Tile Humid Road 04 111000
4 Tile Humid Road 05 110100
5 Tile Humid Road 06 110010
6 Tile Humid Road 07 101010
7 Tile Humid Road 08 111100
8 Tile Humid Road 09 111010
9 Tile Humid Road 10 110110
10 Tile Humid Road 11 111110
11 Tile Humid Road 12 111111
110000
101000
100100
111000
110100
110010
101010
111100
111010
110110
111110
111111