FARCRY® 5

Procedural World Generation

Etienne Carrier

Technical Artist

etienne.carrier@ubisoft.com

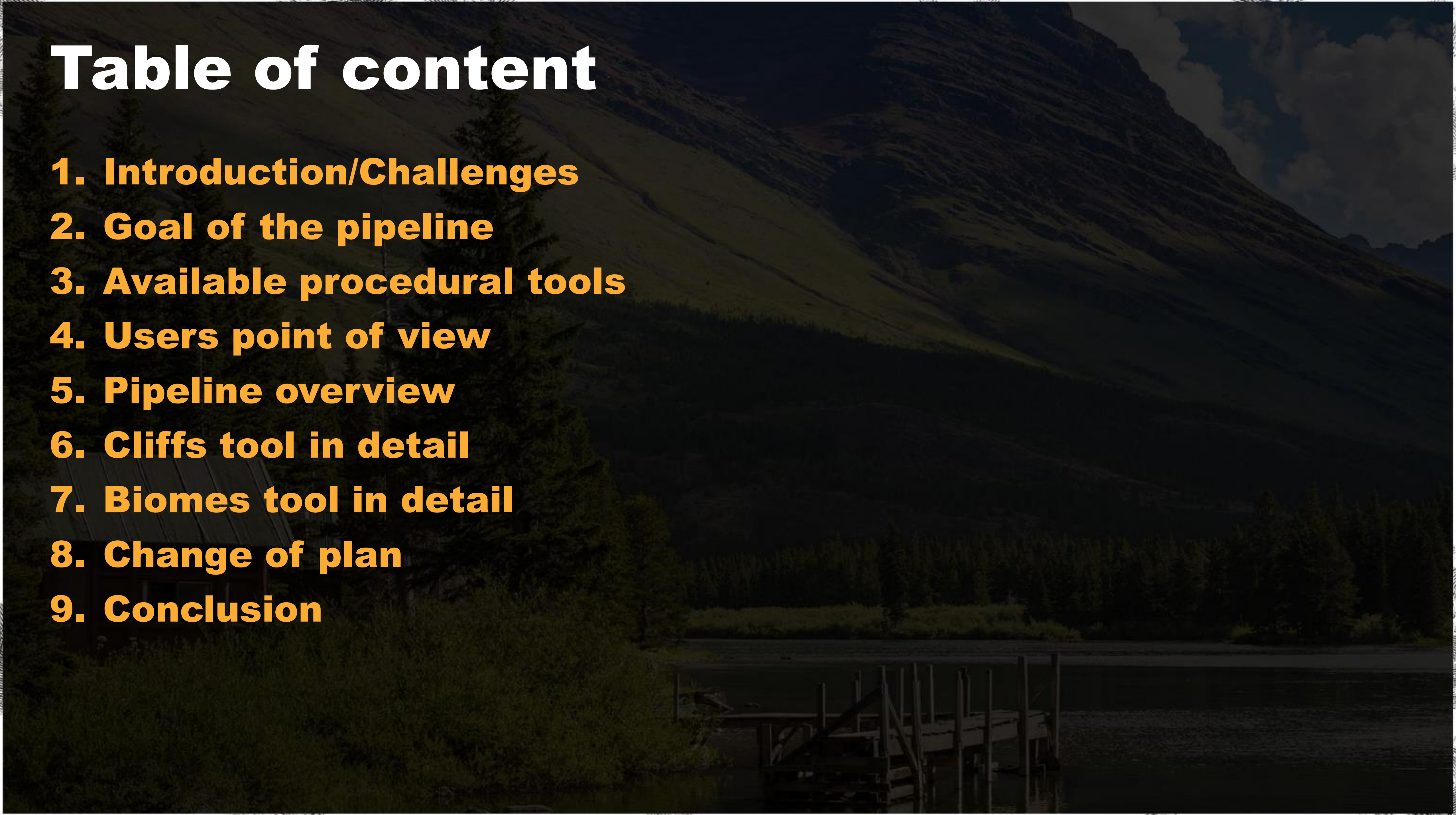


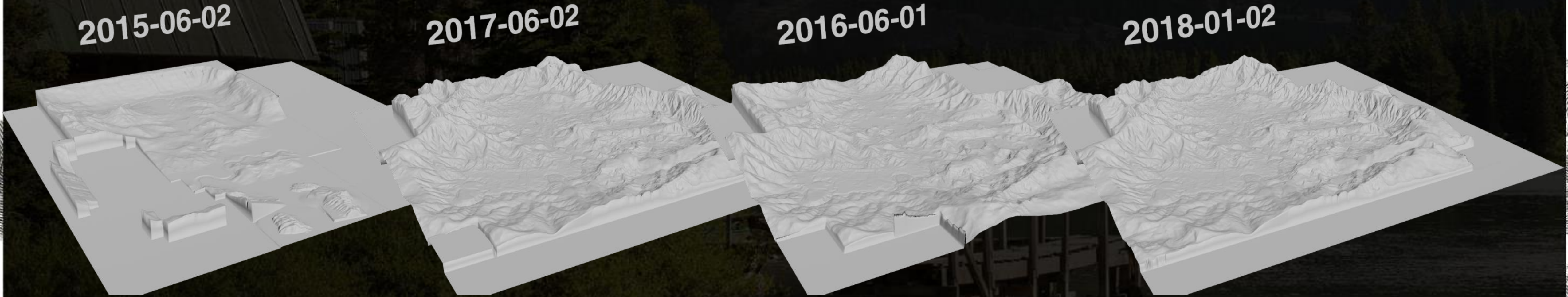
Table of content

1- INTRODUCTION

FarCry5 Terrain

evolution over 2.5 years

How do we manage our content with a terrain that is changing constantly?

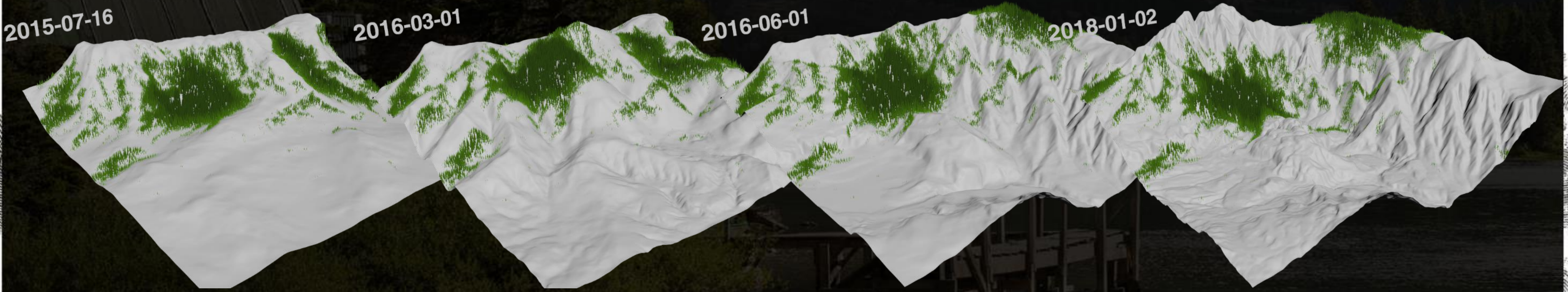


Fixed Forest Position

Over 2.5 years

Same evolution on smaller portion of the terrain with a forest applied manually. The forest distribution patterns is initially coherent with the terraforming. However, it becomes really incoherent over time as the terrain changes

How do we solve that?



2- GOAL OF THE PIPELINE

Fill up the world

1st objective

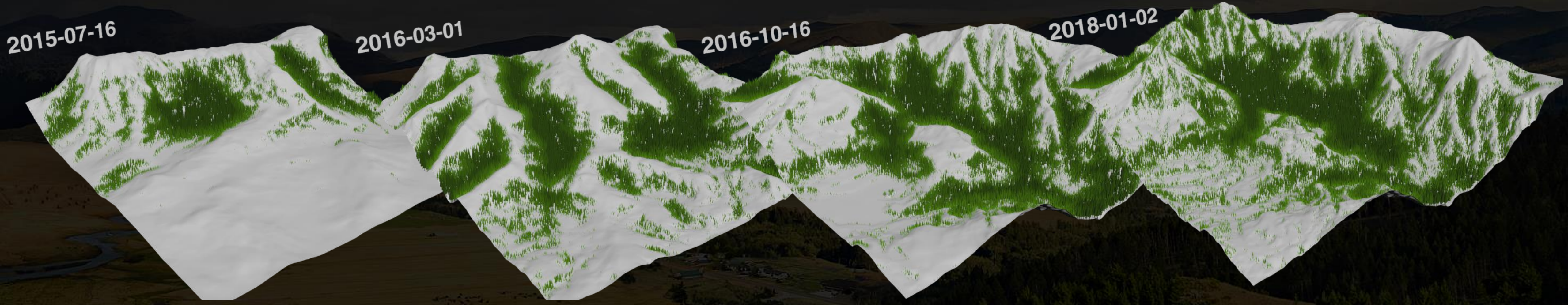
Macro management tool to fill up the world with natural looking content.

Consistent with terrain topology

2nd objective

The content needs to be consistent with terrain topology.

Here we have the same example as before, but this time with a forest distribution that is adapting to the terrain shape. And as we can see the result remains coherent with any of the terraforming changes.



Automated

3rd objective

The whole pipeline needs to be automated. We used Houdini and Houdini Engine and nightly generation on build machines to fully refresh the world every night.

Several build machines will process different part of the world one by one, until the full world is generated. This process ensures us that the users will have updated world data every morning.

Houdini™
3D ANIMATION TOOLS

Houdini
ENGINE™

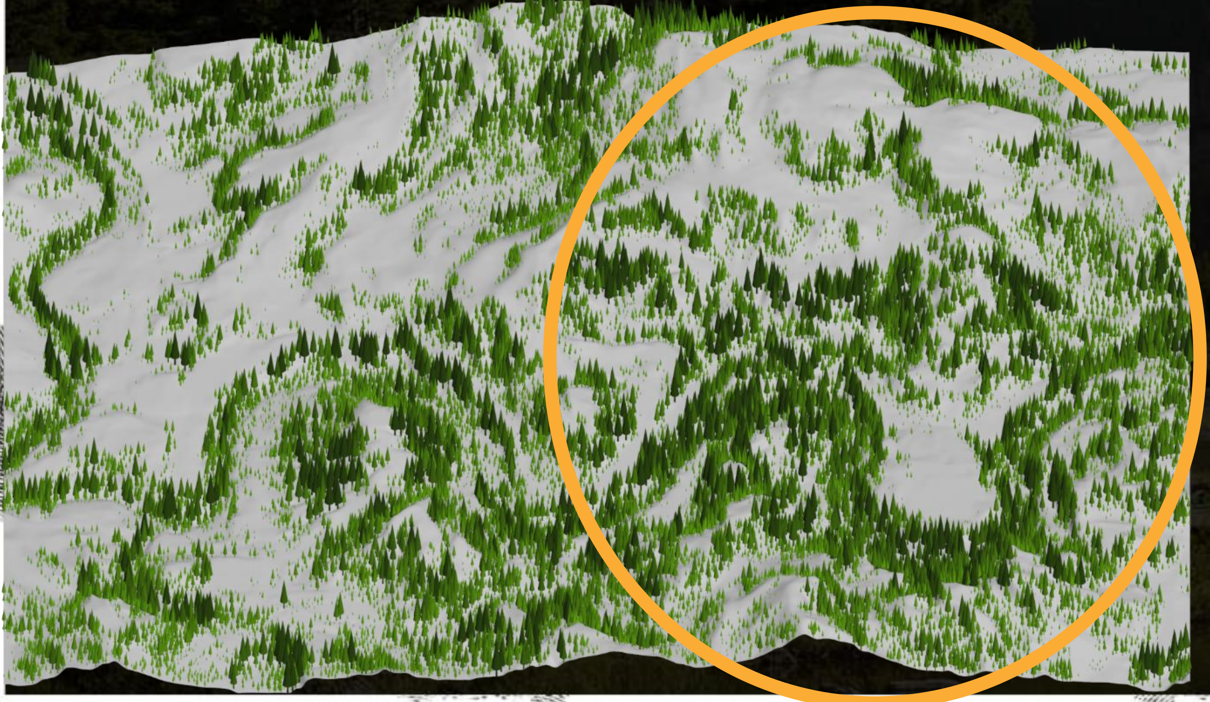
Deterministic

4th objective

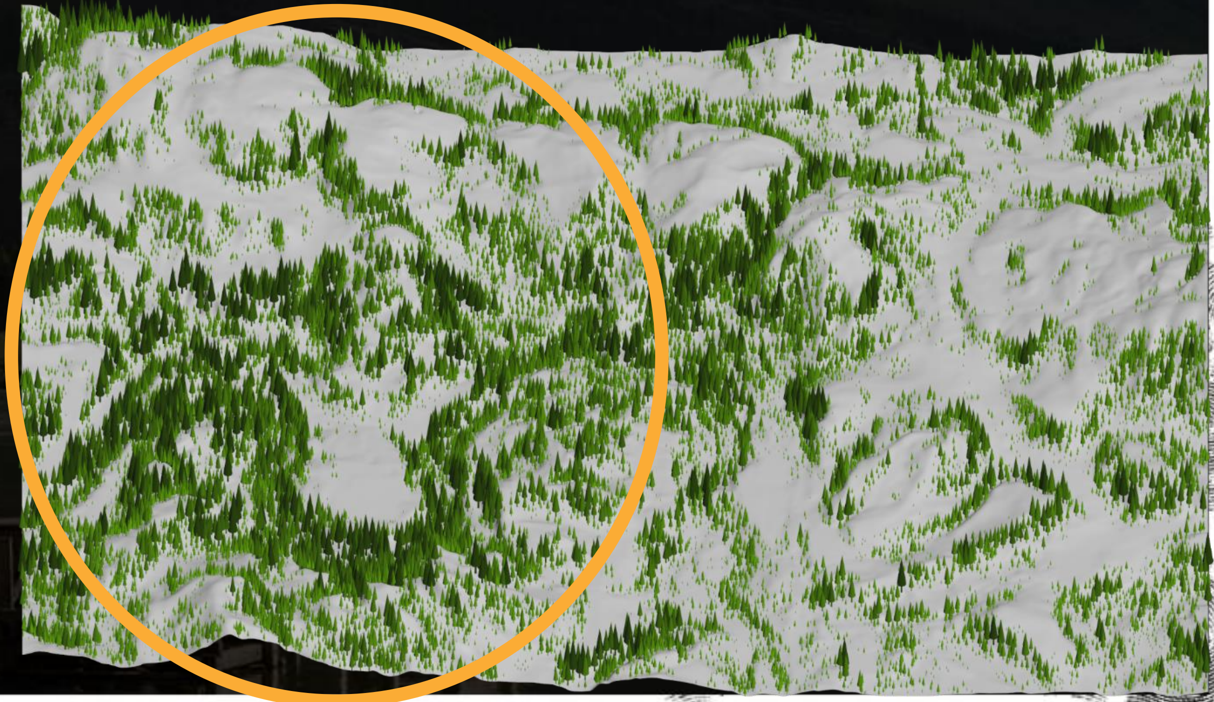
All the data generated also needs to be deterministic. Which means that, the generation needs to yield the same result given the same inputs.

Even if we bake on different build machines, same part of the terrain will always give the same results. This is important for the navmesh generation among other things because we have a nightly build that bakes the world map per map so the junctions between maps needs to be seamless.

Build Machine A



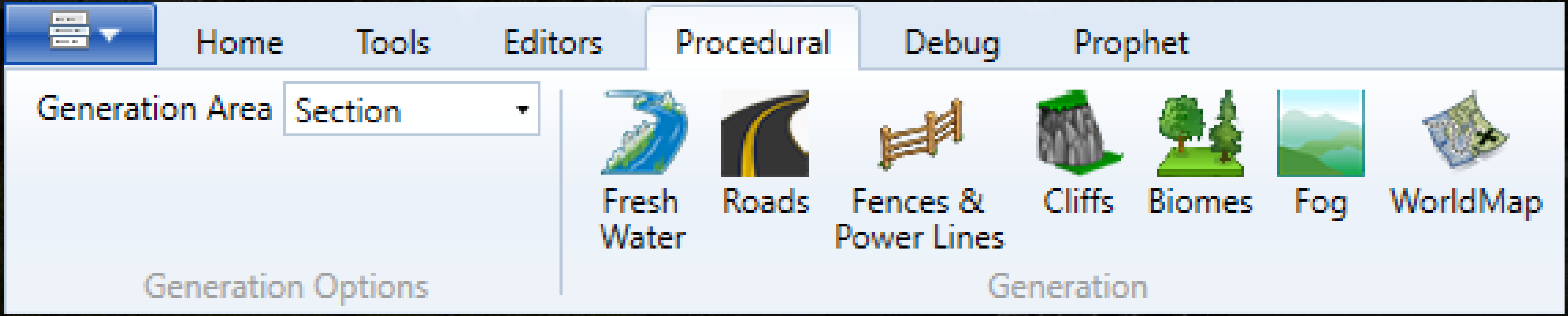
Build Machine B



User friendly

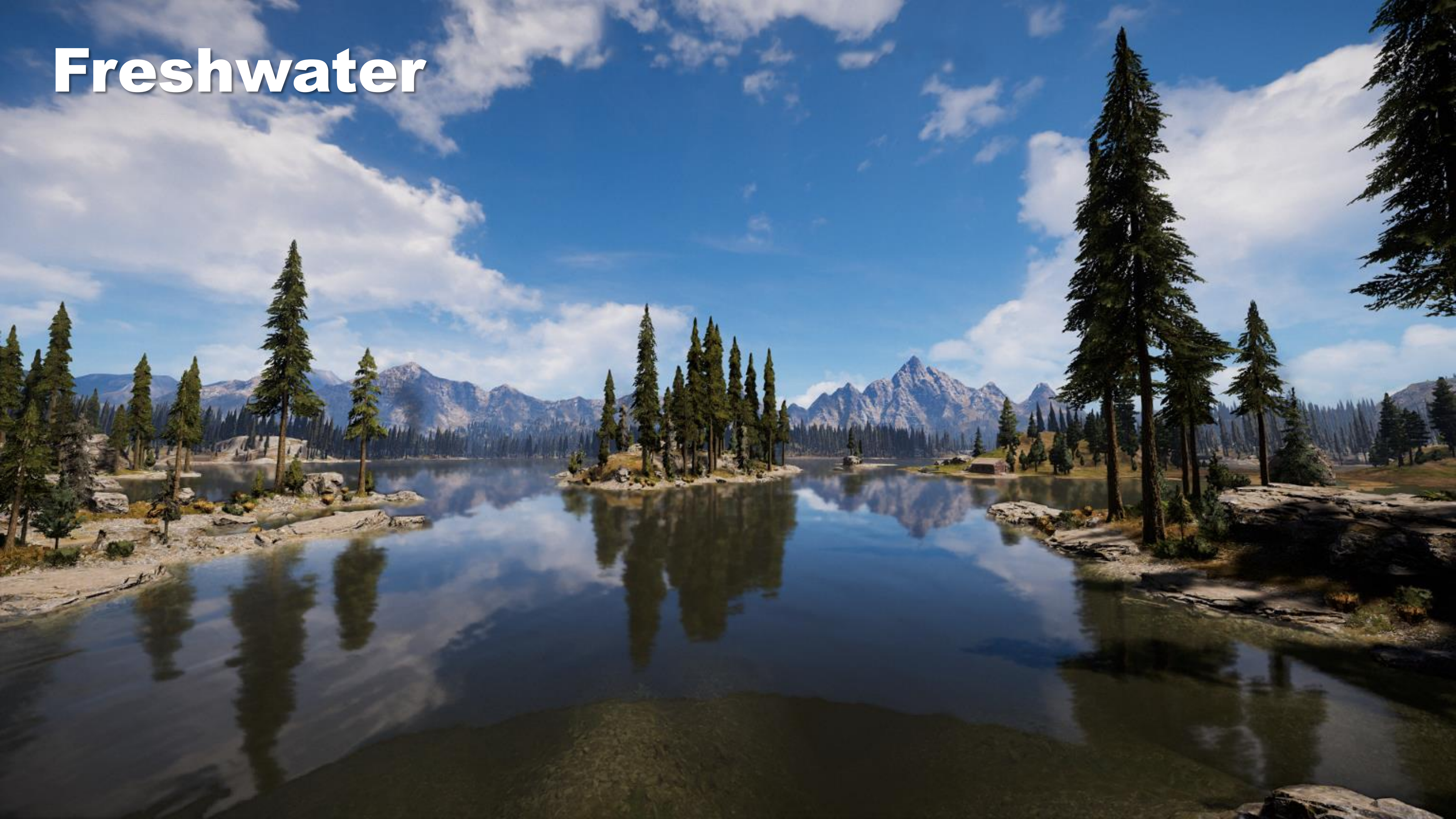
5th objective

- Shelf tools in the editor
- Easy to manage by the users
- Users can bake as they work
- Ability to override procedural results

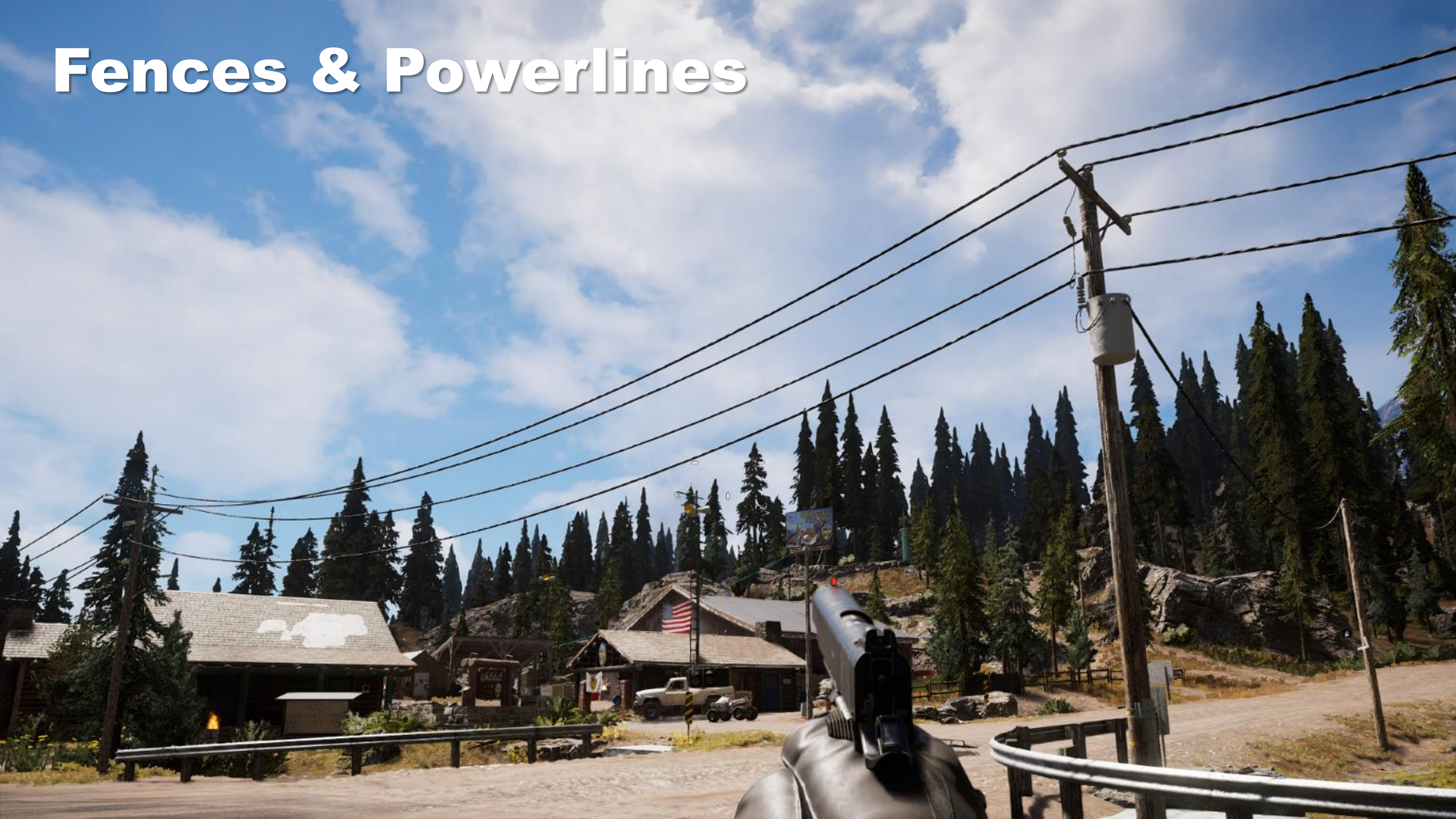


3- THE TOOLS

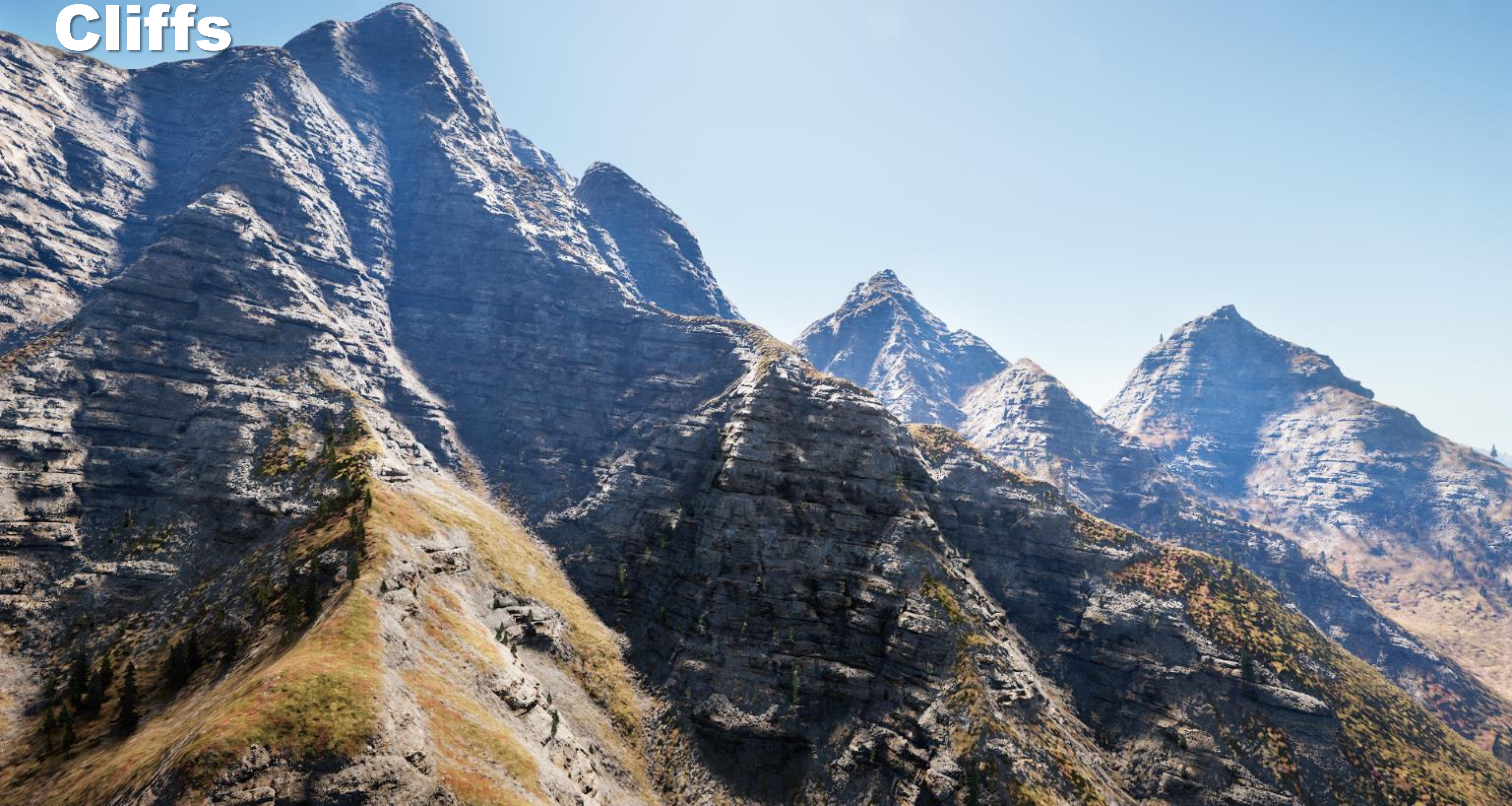
Freshwater



Fences & Powerlines

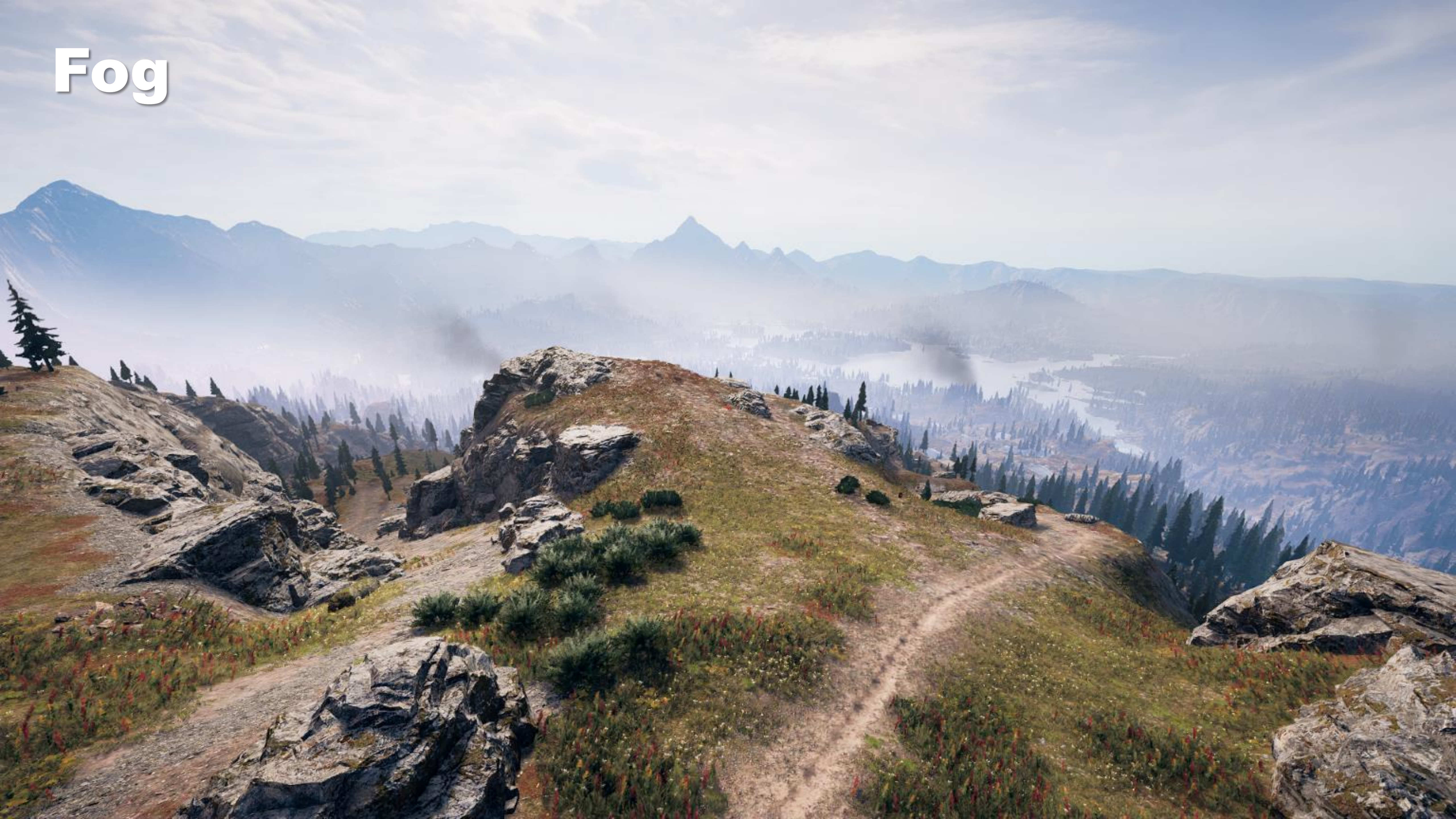


Cliffs

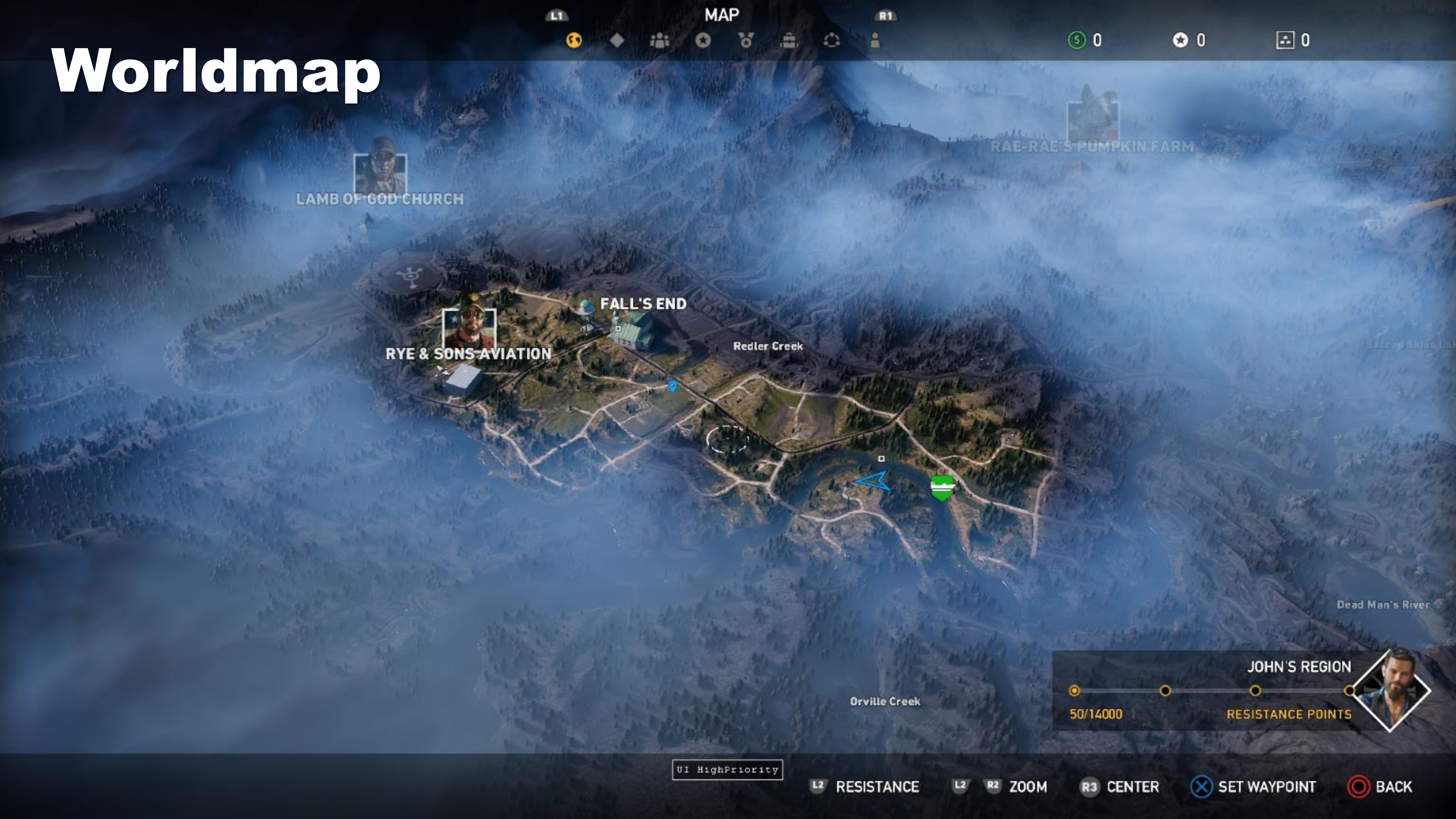


Biomes

Fog



Worldmap

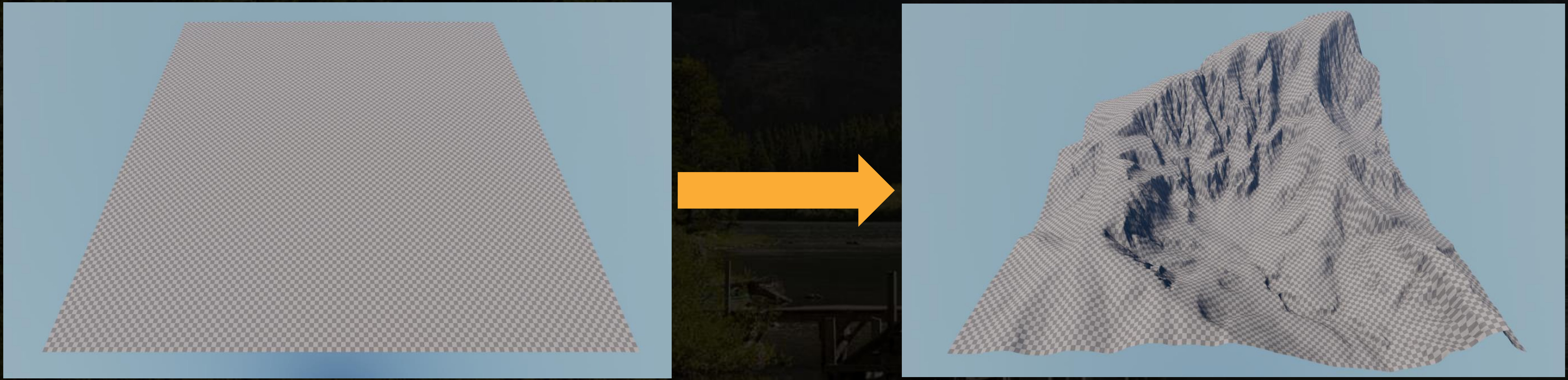

MAP
L1
R1
0
0
0
LAMB OF GOD CHURCH
RAE-RAE'S PUMPKIN FARM
FALL'S END
RYE & SONS AVIATION
Redler Creek
Orville Creek
Dead Man's River
JOHN'S REGION
50/14000
RESISTANCE POINTS
UI HighPriority
RESISTANCE
ZOOM
CENTER
SET WAYPOINT
BACK

4- USERS POINT OF VIEW

Terrain

Terraforming pass

Users will terraform the terrain using Dunia editor tools.

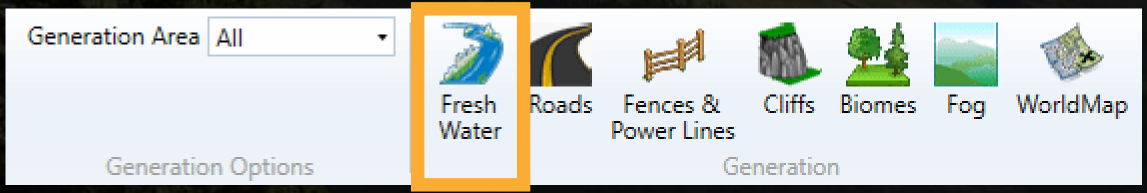


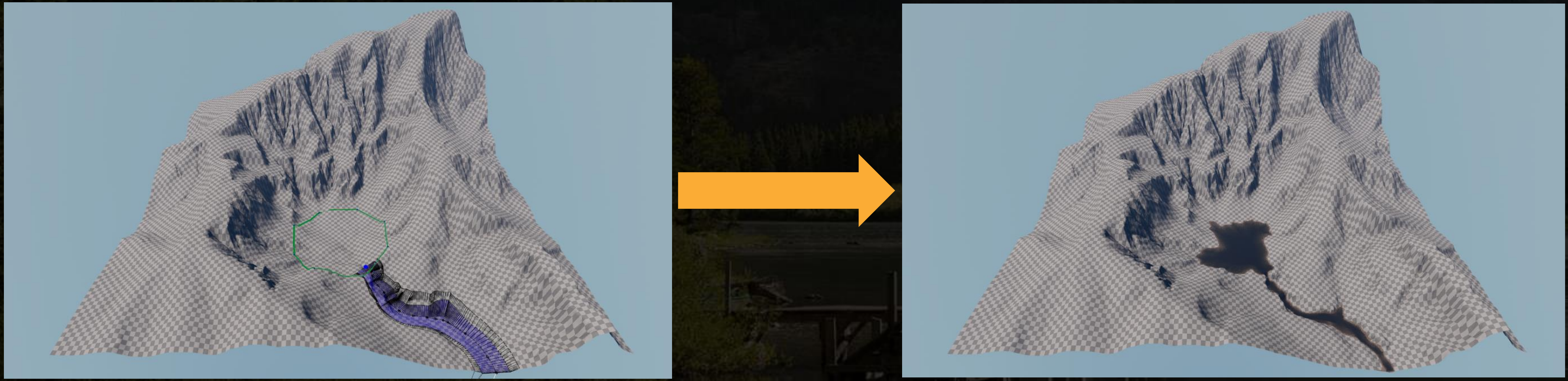
Freshwater

Setup lake, rivers, streams and waterfalls

The artist can lay down a freshwater network. That is achieved by using curves and spline that will be used as inputs for the freshwater tool. For rivers we have controls directly on the spline for the width.

When happy with the inputs the user will run procedural generation from the Freshwater shelf button. The tool will produce water surface and terrain texturing

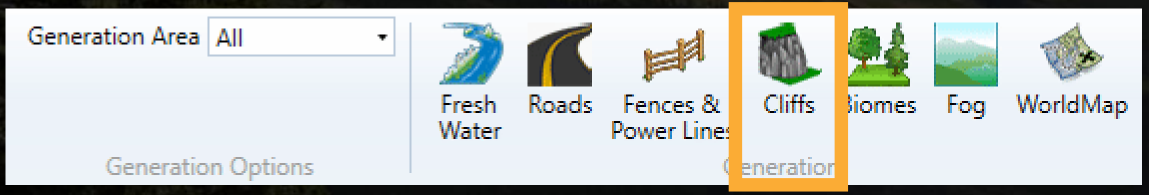


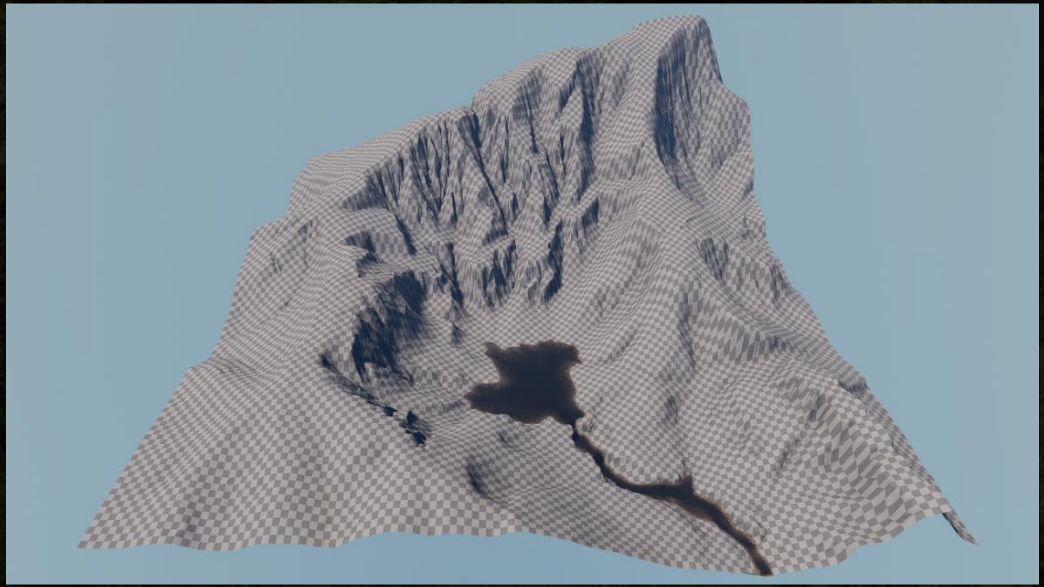


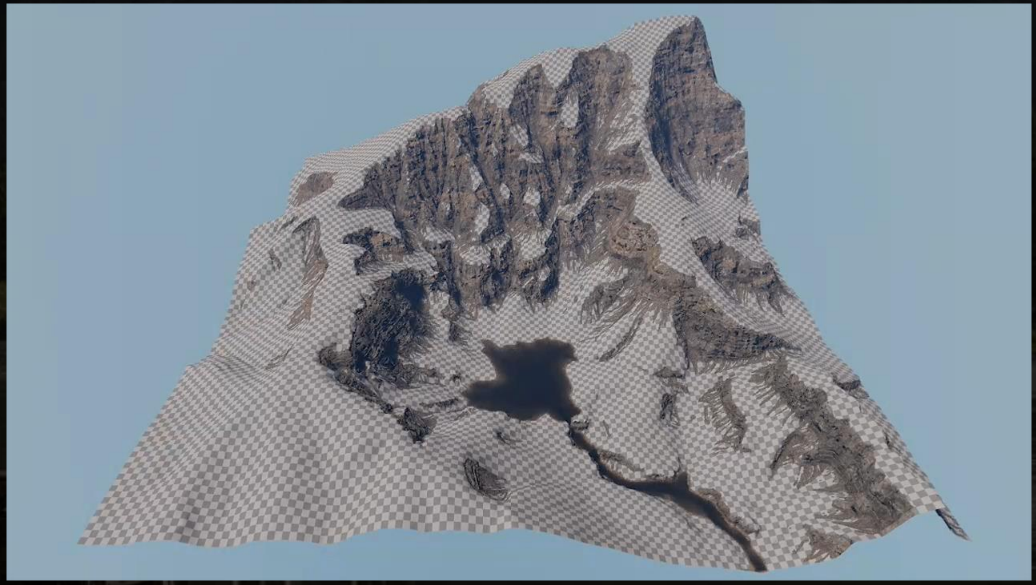
Cliffs

Generating cliff meshes and terrain texture

The cliffs tool is simply based on the terrain slope so the user just have to run the generation on the desired terrain area.





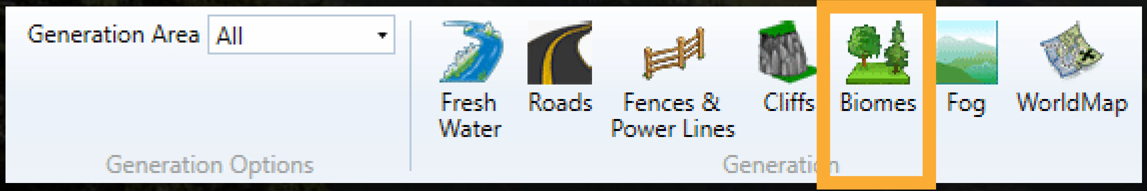


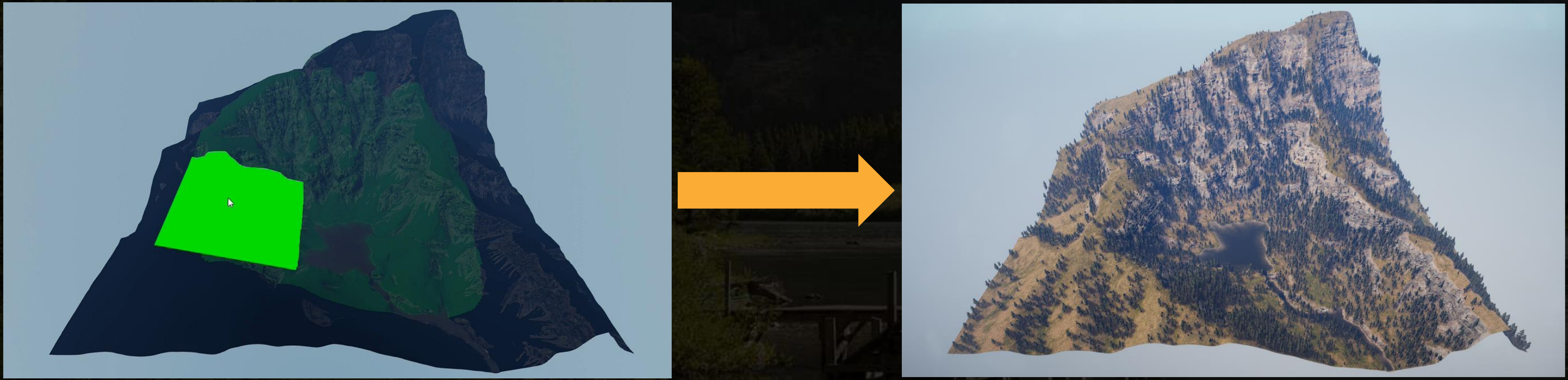
Biomes

Painting biomes

To apply vegetation in the world the user can paint the desired biomes with the biome painter tool.

The main biomes naturally distribute grass and forest sub-biomes.



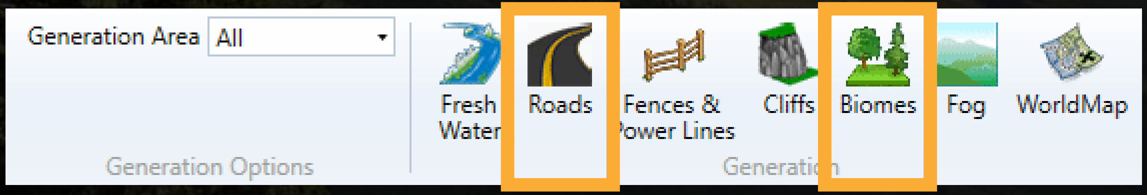


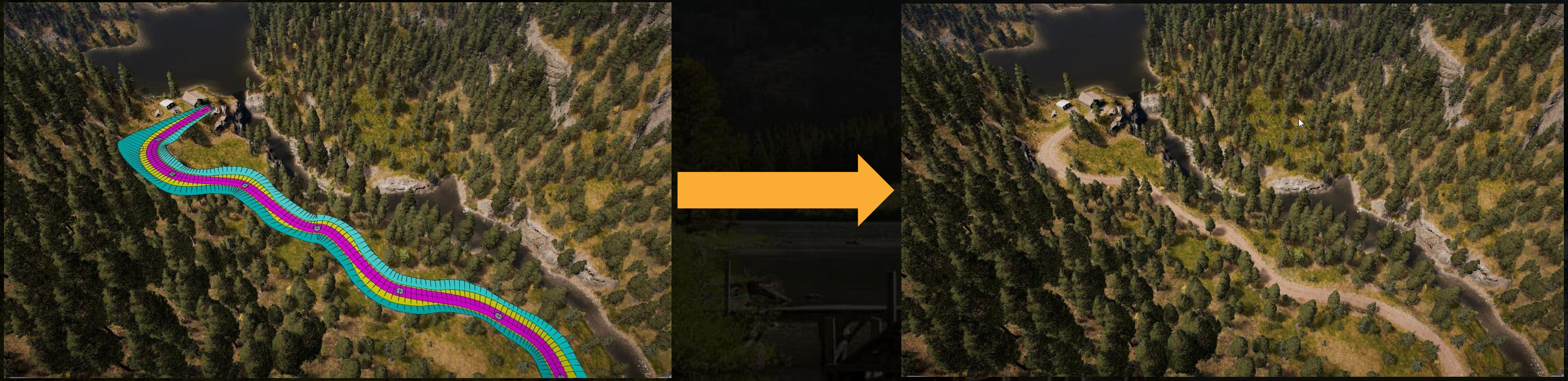
Points of Interest

Adding a road

So far, our result is quite natural. To bring men's influence on the environment and override mother nature the user can start by adding roads. This is done by using the road spline.

After baking the road tool, the user can also refresh the biome tool to clear out the vegetation in the way of the road.

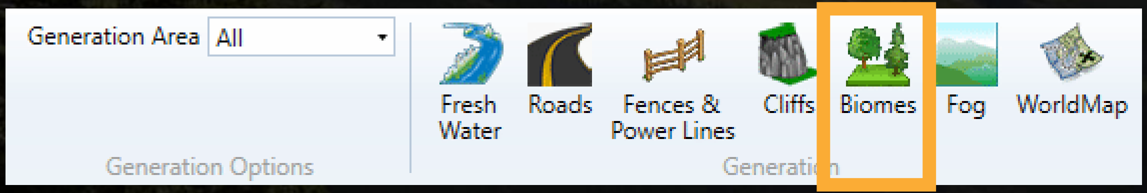


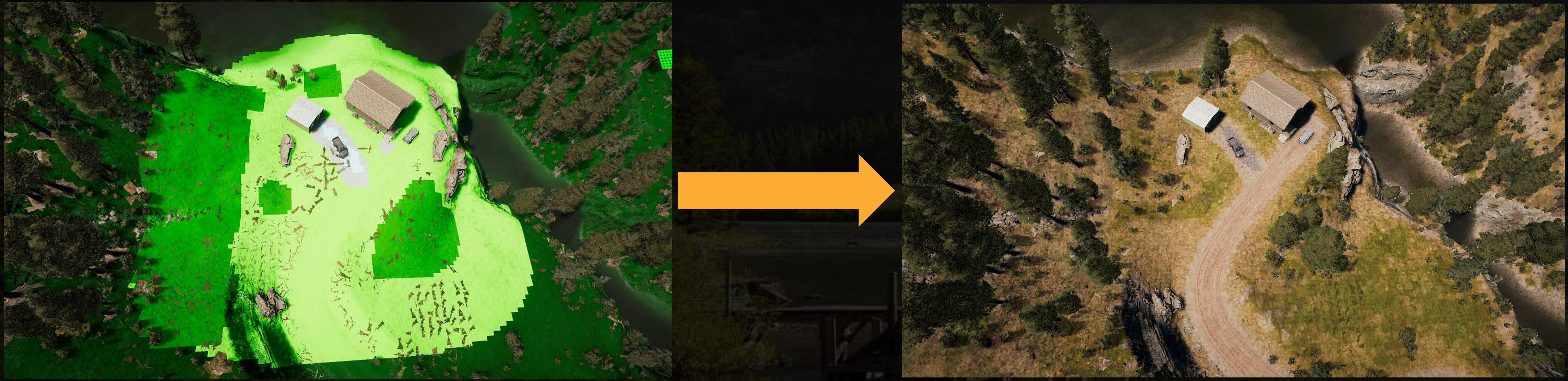


Points of Interest

Setup a location

In order to customize a location, the user can override the main biomes by painting sub-biomes like grass and adding clumps of forest here and there. This will clear the way to hand place buildings and such.

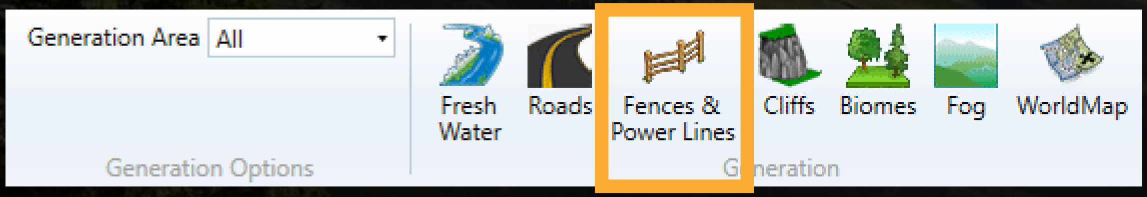


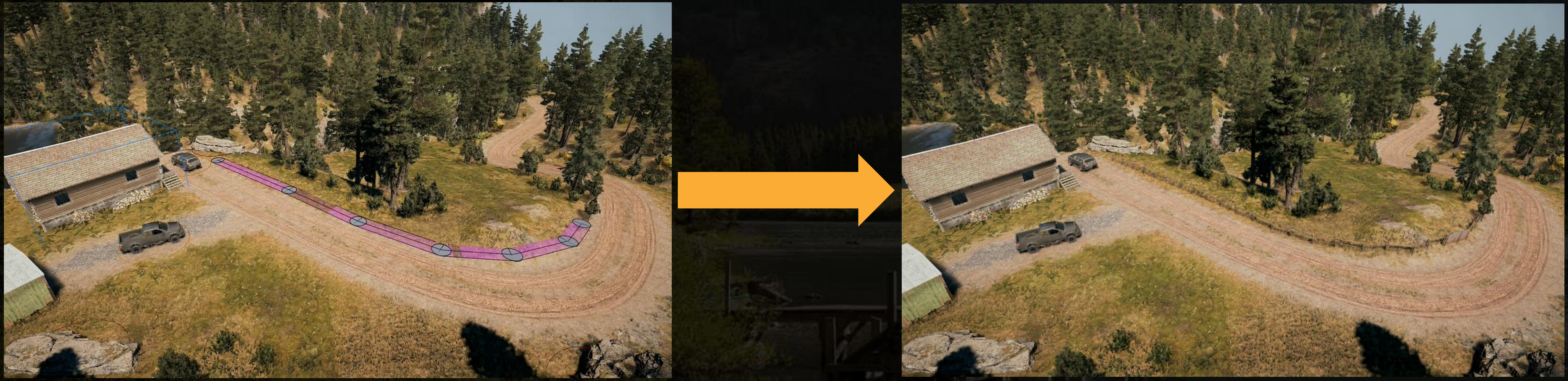


Fences

Adding a fence

The input for fence is once again a spline. The user will simply set the fence type he wants to generate on the spline parameters and run the generation.

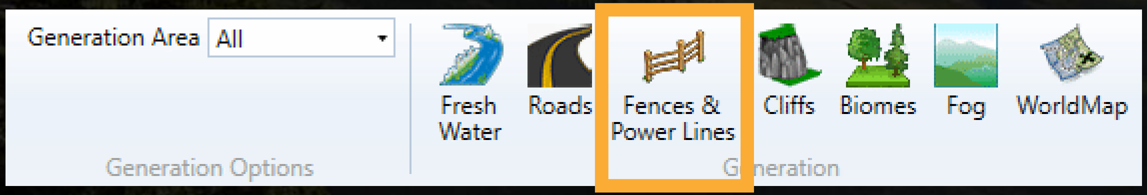


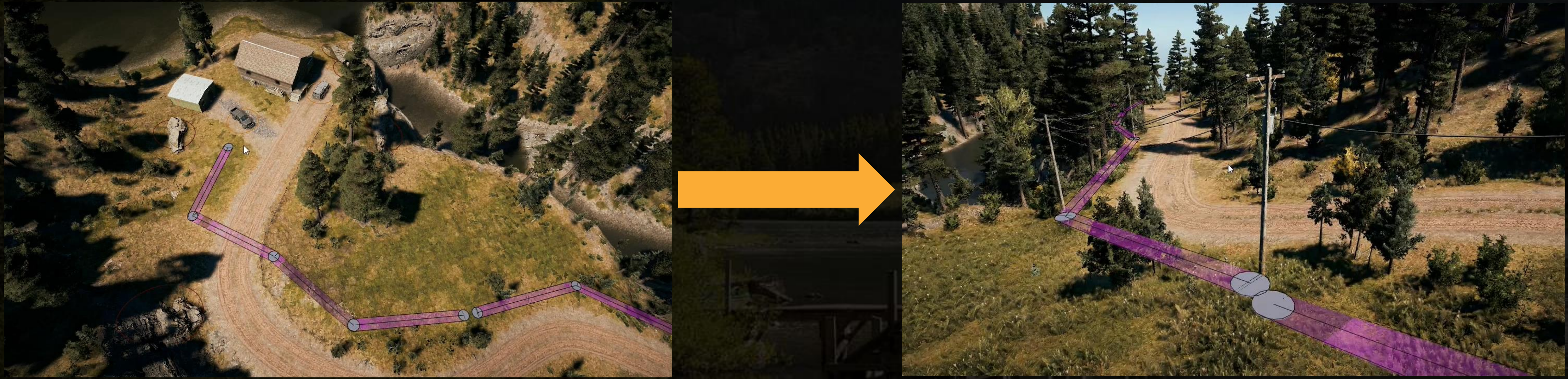


Power lines

Adding power lines

Electric poles will spawn on each control points of the spline. In this example we add 3 different types of power lines that will automatically connect with one another (Standard lines, Single line and 2 House connectors).
Transformer boxes will be automatically added where required.
If the user does not snap the splines correctly the system will do it within a certain distance.





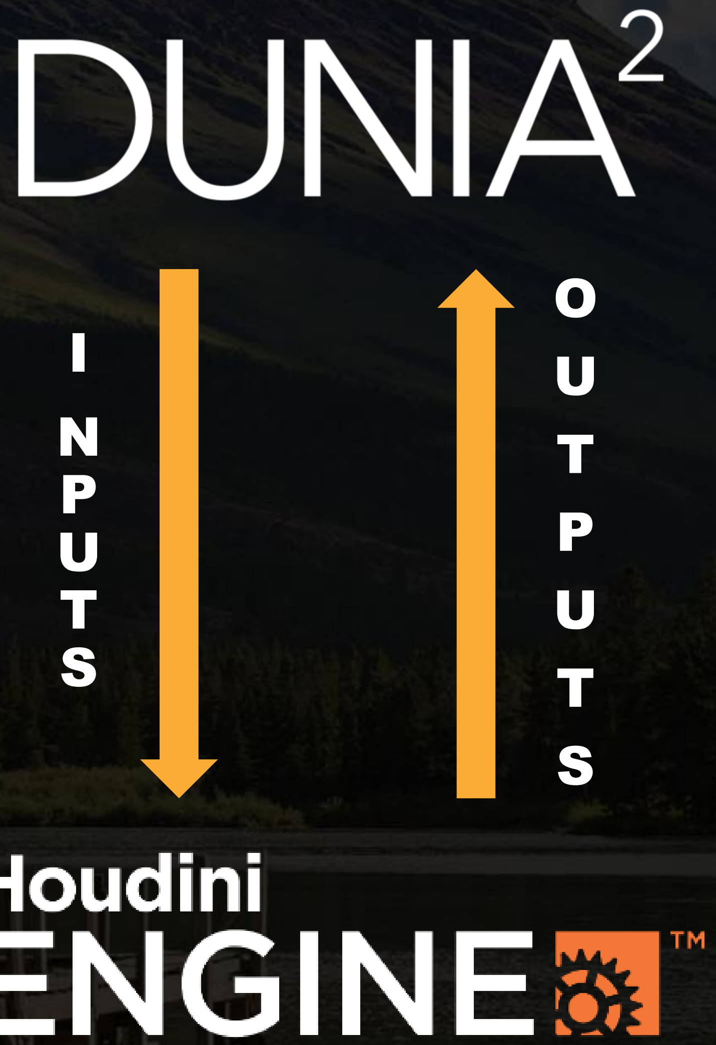
5- THE PIPELINE

How does it work?

Houdini Engine implementation

We have an Houdini Engine implementation with our in house engine and editor Dunia 2.

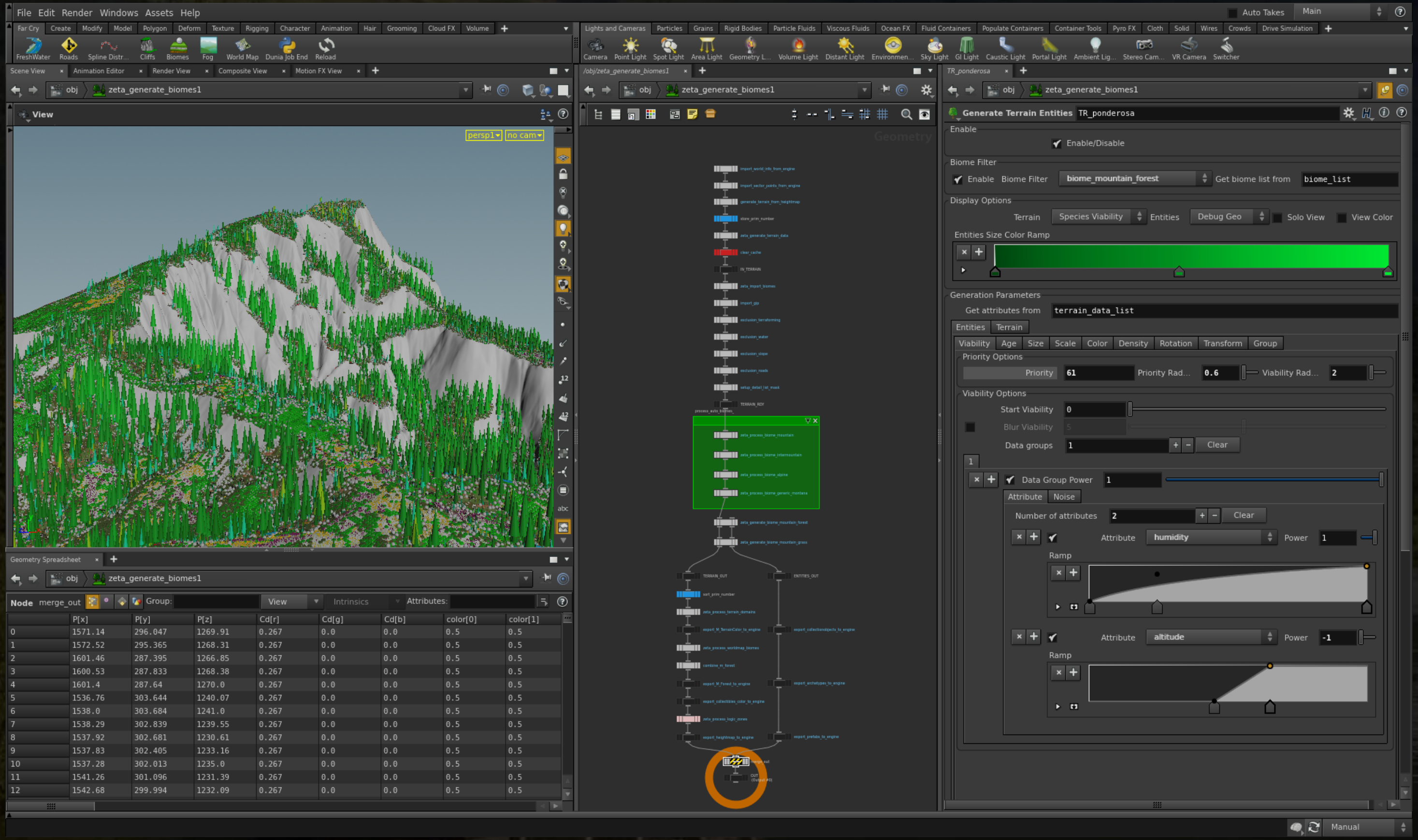
What is opening up doors and possibilities with this pipeline really is the available inputs and outputs. The data exchange between the two!



What is Houdini ?

Houdini is a 3D software with node base workflow, that also allows you to develop tools that we call Houdini Digital Assets (or HDA in short).

HDA can be run in Houdini Engine API when implemented as a plug-in with a specific application, like a game editor.



Generate Terrain Entities TR_ponderosa

Enable
- ✓ Enable/Disable

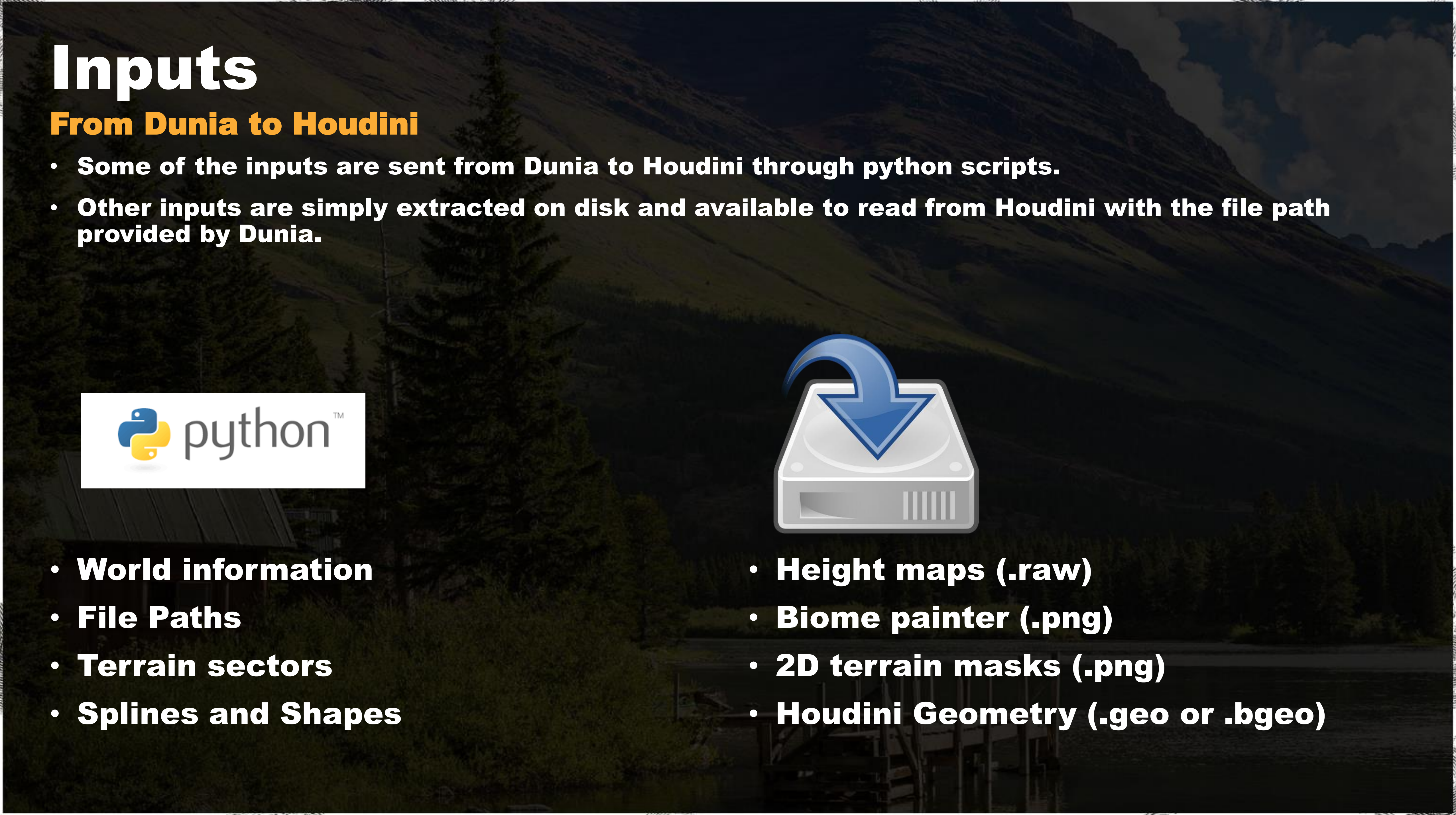
Biome Filter
- ✓ Enable Biome Filter: biome_mountain_forest — Get biome list from: biome_list

Display Options
- Terrain: Species Viability — Entities: Debug Geo — Solo View — View Color

Generation Parameters
- Get attributes from: terrain_data_list
- Priority: 61 — Priority Rad...: 0.6 — Viability Rad...: 2
- Start Viability: 0
- Blur Viability: 5
- Data groups: 1
- ✓ Data Group Power: 1
- Number of attributes: 2
- ✓ Attribute: humidity — Power: 1
- ✓ Attribute: altitude — Power: -1

Geometry Spreadsheet — obj > zeta_generate_biomes1 — Node: merge_out

	P[x]	P[y]	P[z]	Cd[r]	Cd[g]	Cd[b]	color[0]	color[1]
0	1571.14	296.047	1269.91	0.267	0.0	0.0	0.5	0.5
1	1572.52	295.365	1268.31	0.267	0.0	0.0	0.5	0.5
2	1601.46	287.395	1266.85	0.267	0.0	0.0	0.5	0.5
3	1600.53	287.833	1268.38	0.267	0.0	0.0	0.5	0.5
4	1601.4	287.64	1270.0	0.267	0.0	0.0	0.5	0.5
5	1536.76	303.644	1240.07	0.267	0.0	0.0	0.5	0.5
6	1538.0	303.684	1241.0	0.267	0.0	0.0	0.5	0.5
7	1538.29	302.839	1239.55	0.267	0.0	0.0	0.5	0.5
8	1537.92	302.681	1230.61	0.267	0.0	0.0	0.5	0.5
9	1537.83	302.405	1233.16	0.267	0.0	0.0	0.5	0.5
10	1537.28	302.013	1235.0	0.267	0.0	0.0	0.5	0.5
11	1541.26	301.096	1231.39	0.267	0.0	0.0	0.5	0.5
12	1542.68	299.994	1232.09	0.267	0.0	0.0	0.5	0.5



Inputs

From Dunia to Houdini

- Some of the inputs are sent from Dunia to Houdini through python scripts.
- Other inputs are simply extracted on disk and available to read from Houdini with the file path provided by Dunia.

- World information
- File Paths
- Terrain sectors
- Splines and Shapes

- Height maps (.raw)
- Biome painter (.png)
- 2D terrain masks (.png)
- Houdini Geometry (.geo or .bgeo)

Inputs

From Dunia to Houdini

Some of the inputs are sent from Dunia to Houdini through python scripts.

- **World info** (world name and the size of it, because we didn't have only one world, we also had gyms for testing purposes)
- **File paths** (because we cant expect all pcs to have their stuff installed at the same paths)
- **Terrain sectors** (which area of the terrain we want to generate, we will see more about this in a moment)
- **Splines and Shapes** (along with metadata as geometry attributes. For instance the fences spline would have the fence type as a primitive attribute)

Other inputs are simply extracted on disk and available to read from Houdini with the file path provided by Dunia.

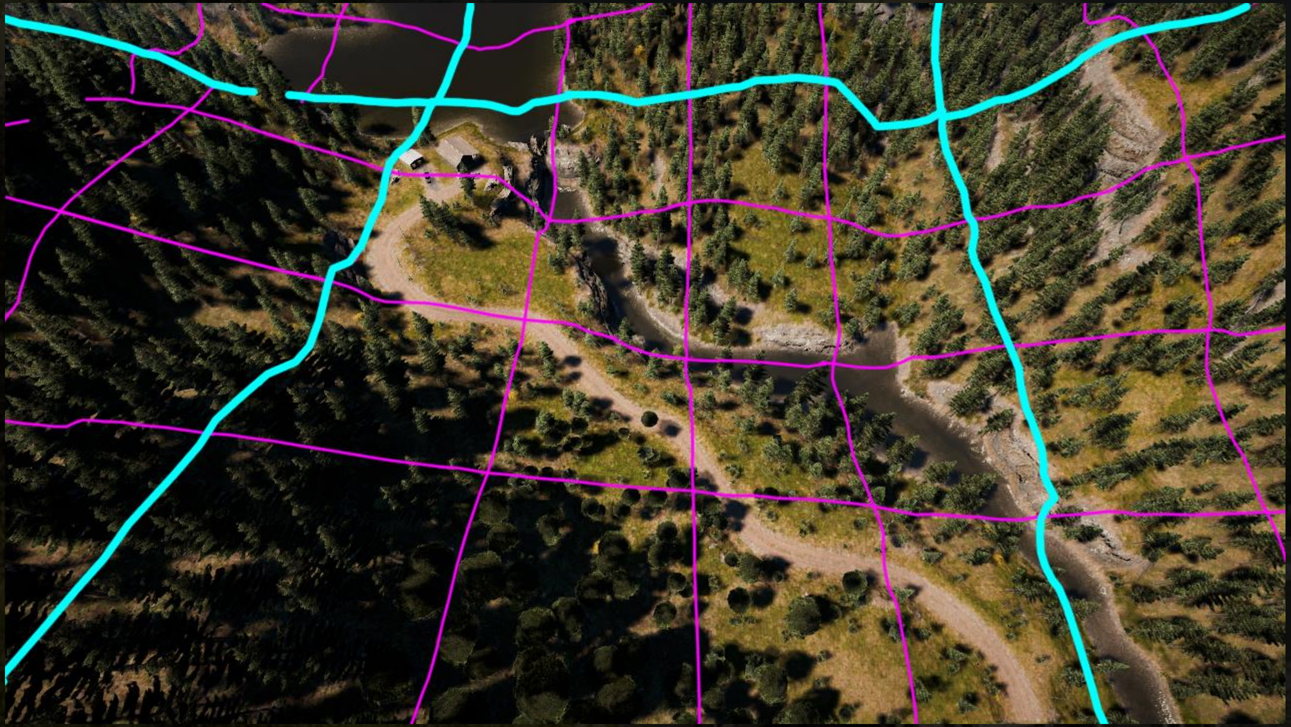
- **Height maps** (.raw)
- **Biome painter** (.png)
- **2D terrain masks** (.png)
- **Houdini Geometry** (.geo or .bgeo that might have been generated by specific procedural tools)

Inputs

Terrain is the main input

Procedural generation is linked to specific area of the terrain. Our smallest granularity is a sector of 64x64 meters. Which means it is the smallest area a user can bake.

Sectors (64m x 64m) - Sections (256m x 256m)

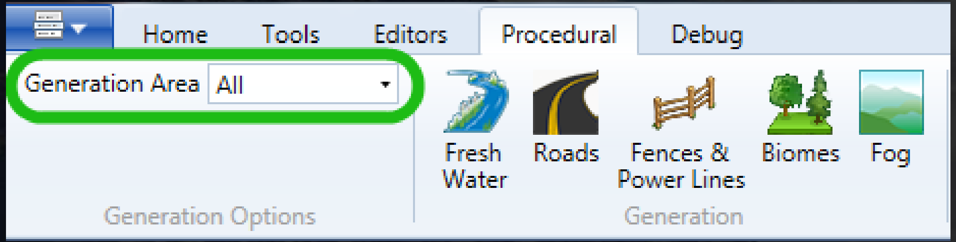


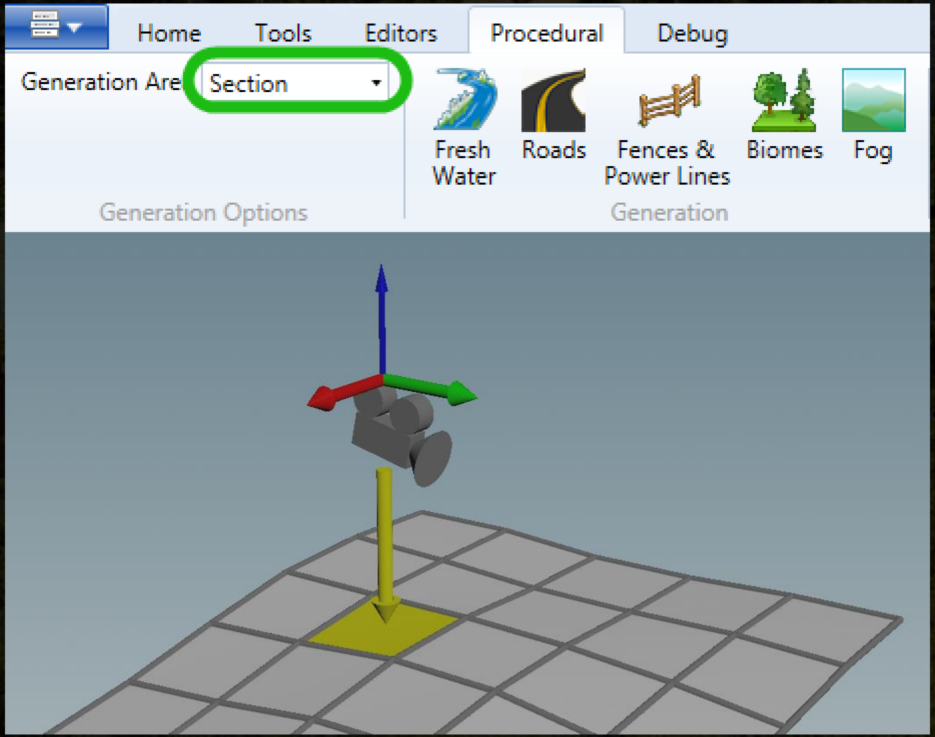
This is how the terrain is subdivided in the editor.

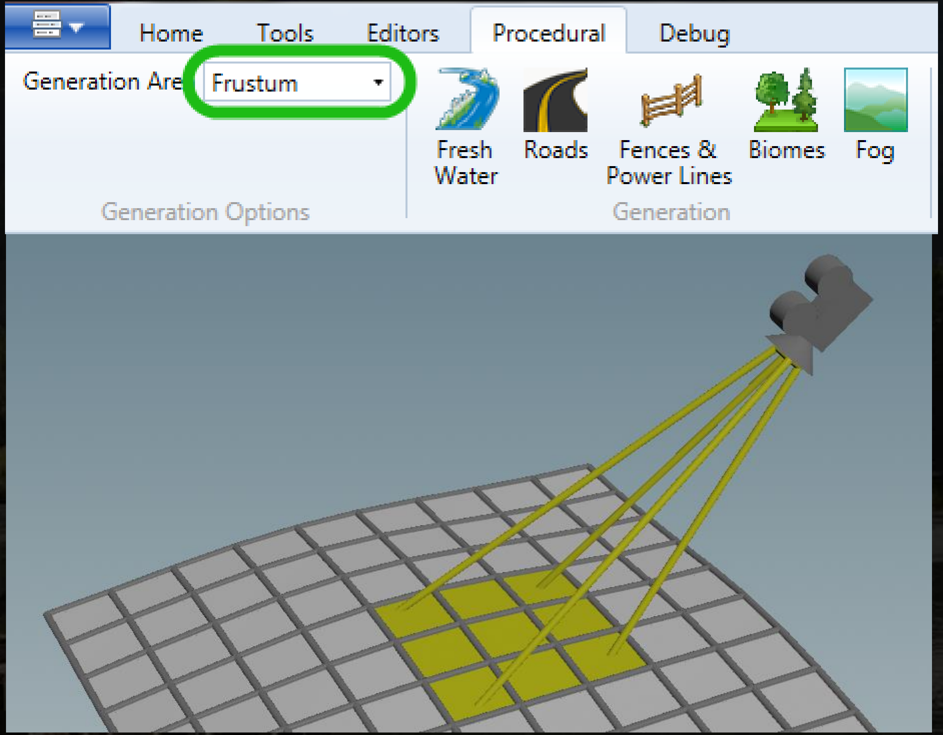
Baking Procedural

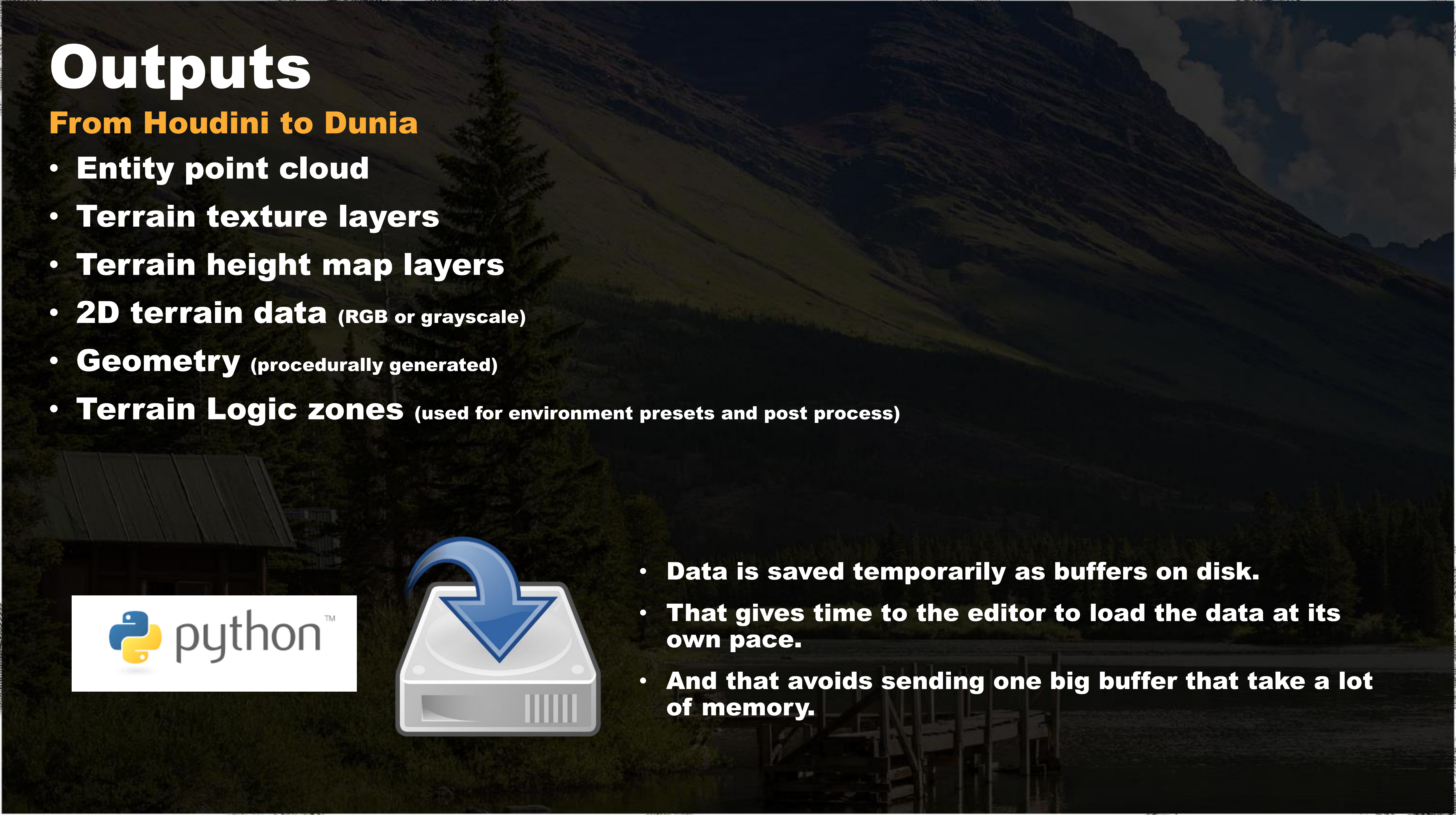
Generation Area

- **All** (all the terrain loaded in the editor)
- **Map (1024m x 1024m)** (located directly under the camera in editor)
- **Section (256m x 256m)** (located directly under the camera in editor)
- **Sector (64m x 64m)** (located directly under the camera in editor)
- **Frustum** (all sectors visible by the camera)


Generation Area: All


Generation Area: Section


Generation Area: Frustum



Outputs

From Houdini to Dunia

- Entity point cloud
- Terrain texture layers
- Terrain height map layers
- 2D terrain data (RGB or grayscale)
- Geometry (procedurally generated)
- Terrain Logic zones (used for environment presets and post process)

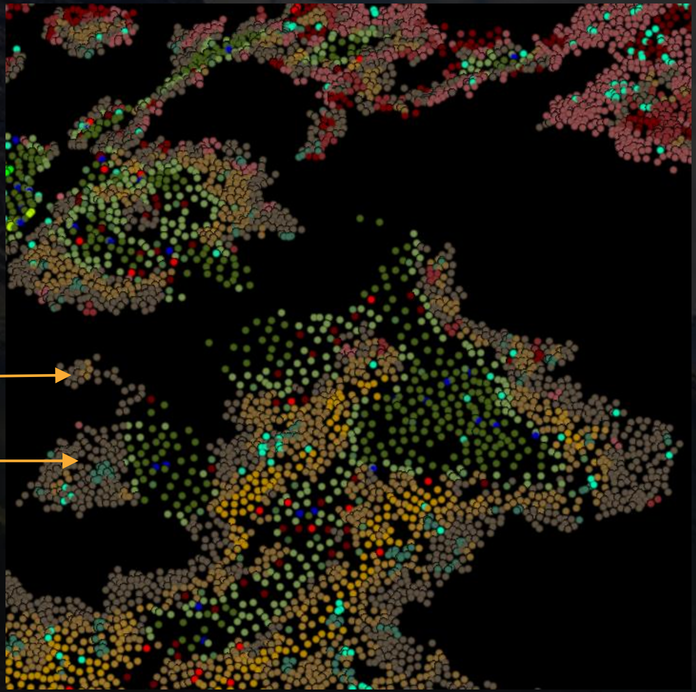
python™

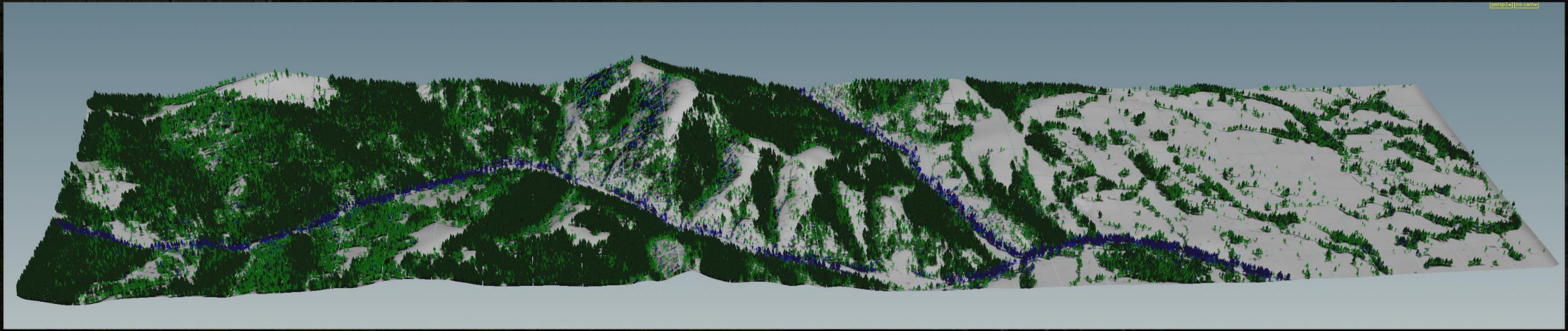
- Data is saved temporarily as buffers on disk.
- That gives time to the editor to load the data at its own pace.
- And that avoids sending one big buffer that take a lot of memory.

Entity Point Cloud

Could be any type of object that has a position in editor

- Vegetation assets
- Rocks
- Collectibles
- Decals
- VFX
- Prefabs
- etc

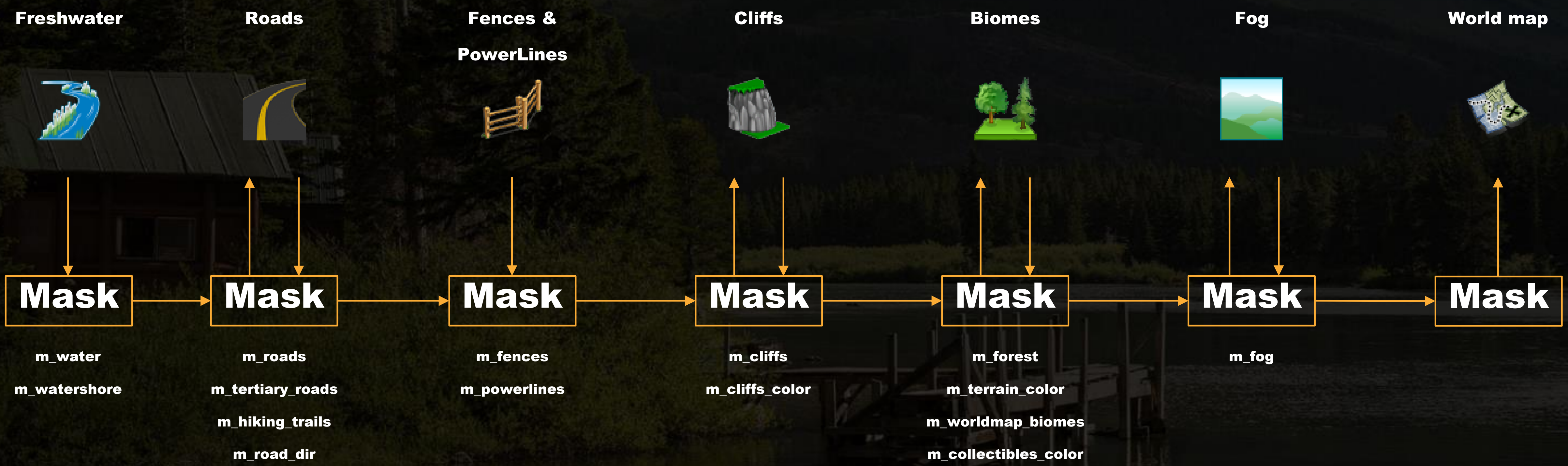




Tools interconnectivity

Relations between each tools

- Each tool will output necessary masks to affect the next ones.
- Cooking order is important if one tool requires input from a previous one.



6- THE CLIFF TOOL

Table of content

Previous tech

= none

On Far Cry 4 and Primal, besides the hand placed rocks, the cliffs were only bare terrain.

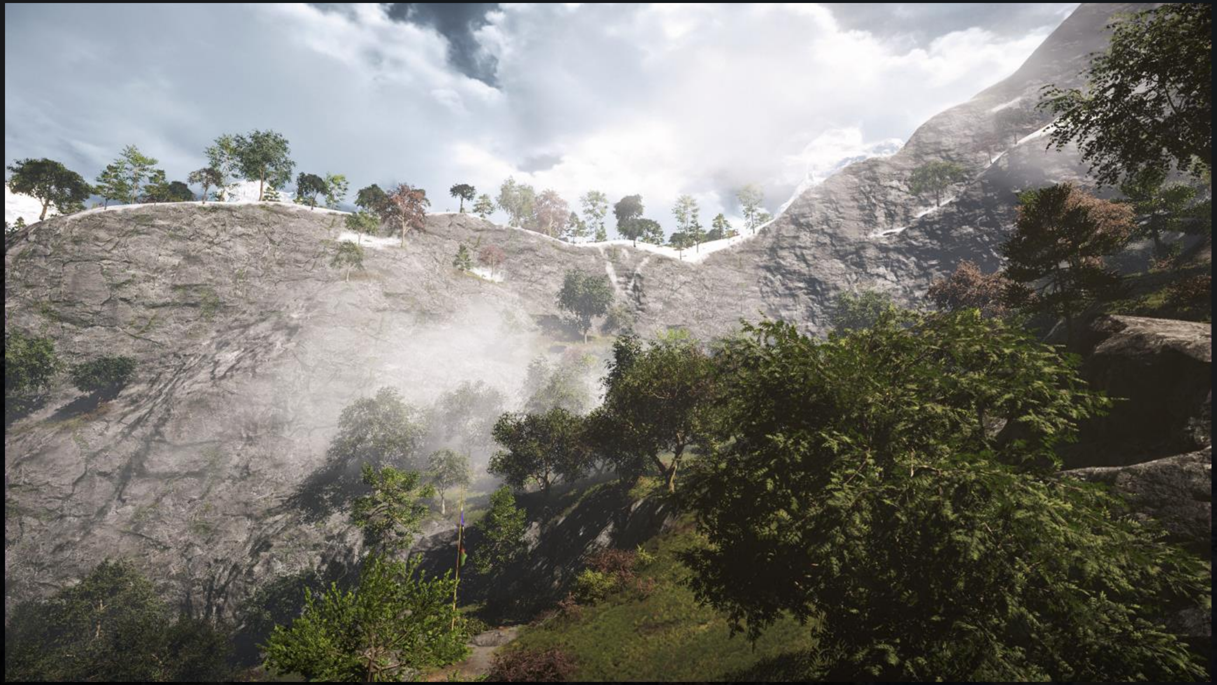
Back then these worlds were design to avoid having large cliffs surfaces.

But now on Far Cry 5, we suddenly had a world with a crazy amount of cliff surfaces. They were much bigger than before and visible from much further.

We either needed to hand place thousands of rocks to cover bare terrain cliffs or find another solution…

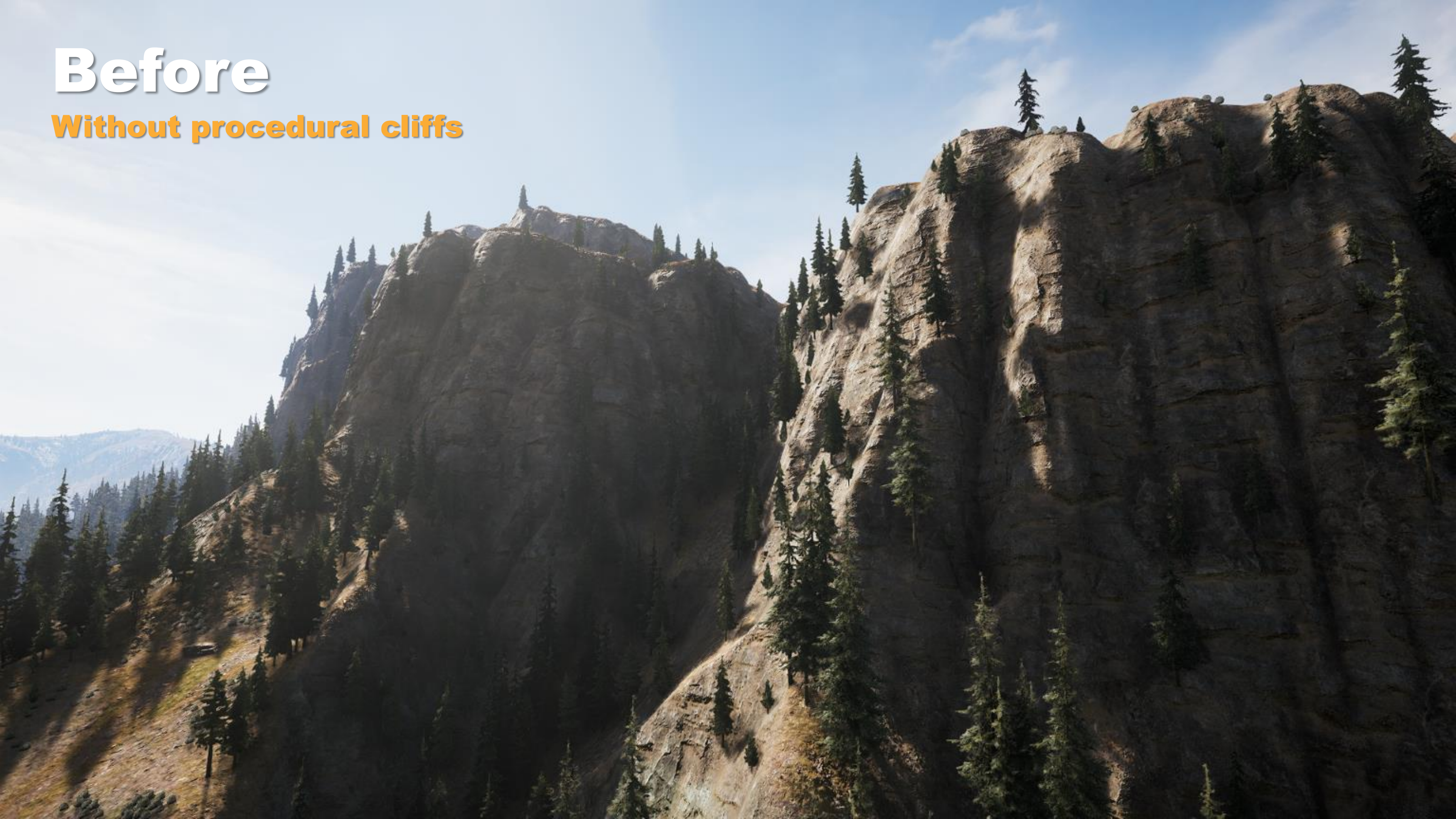
That's the reason why we decided to develop a tech to improve the situation a little bit.

Far Cry 4 cliffs example



Before

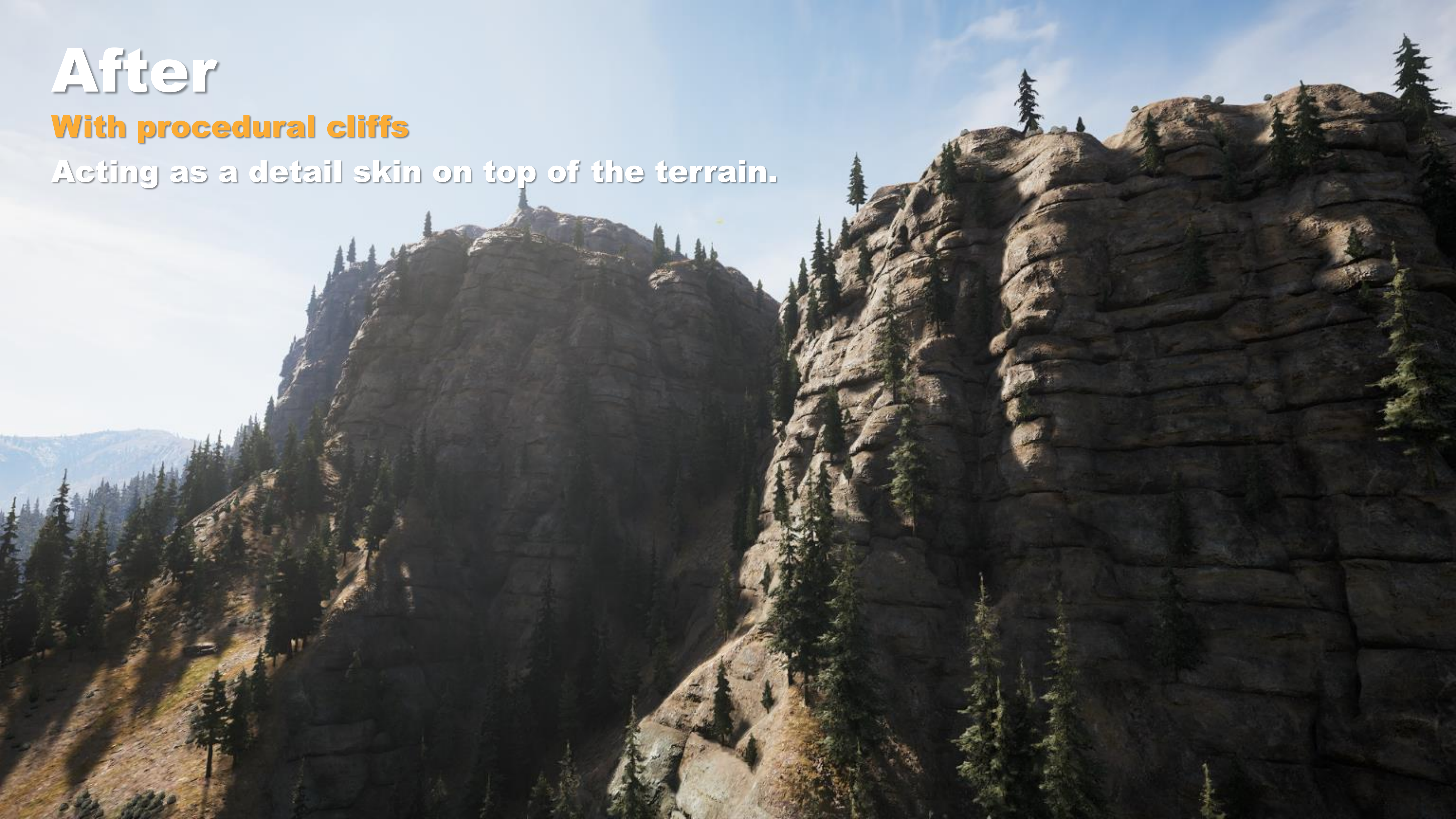
Without procedural cliffs



After

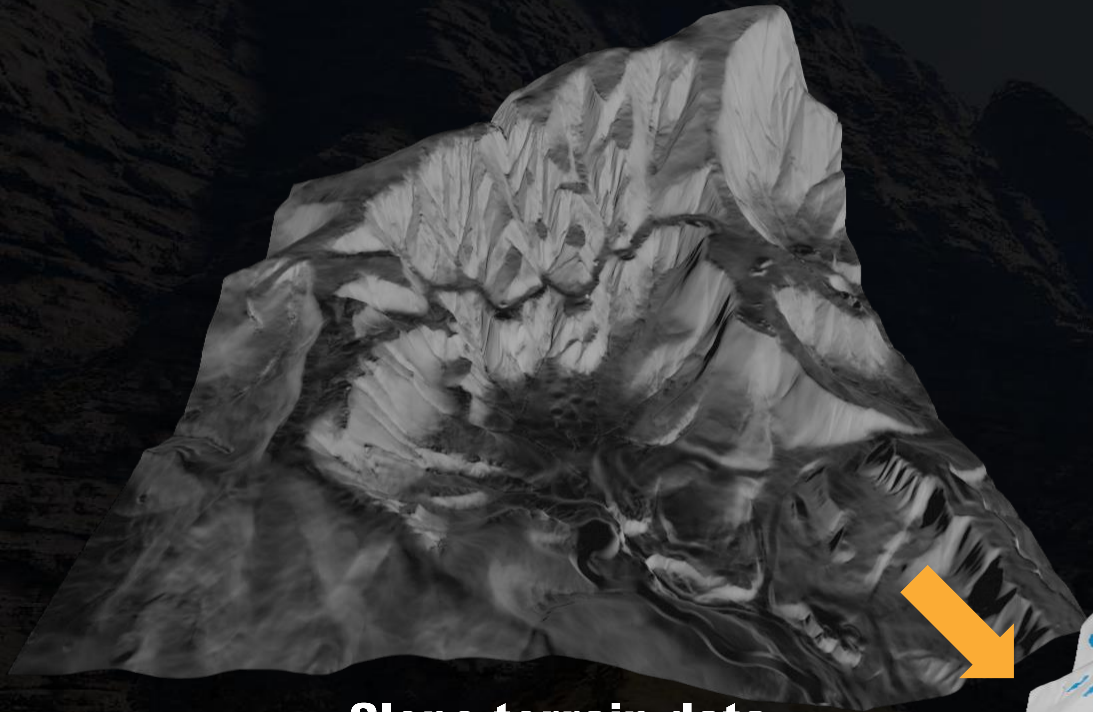
With procedural cliffs

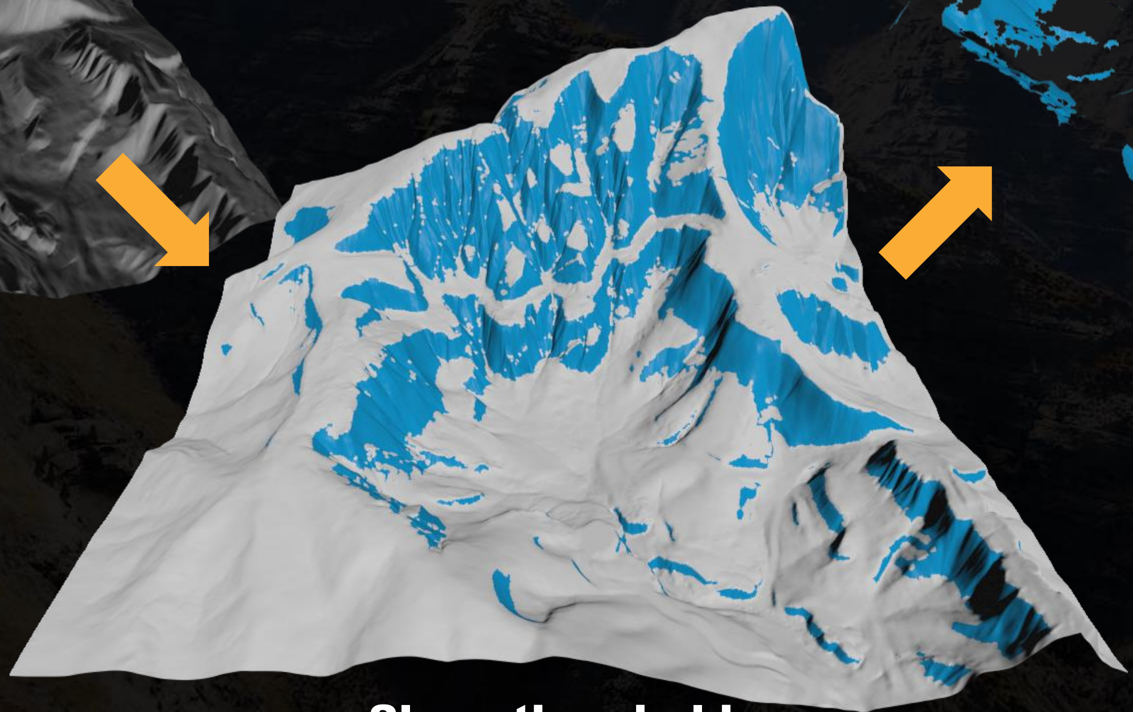
Acting as a detail skin on top of the terrain.

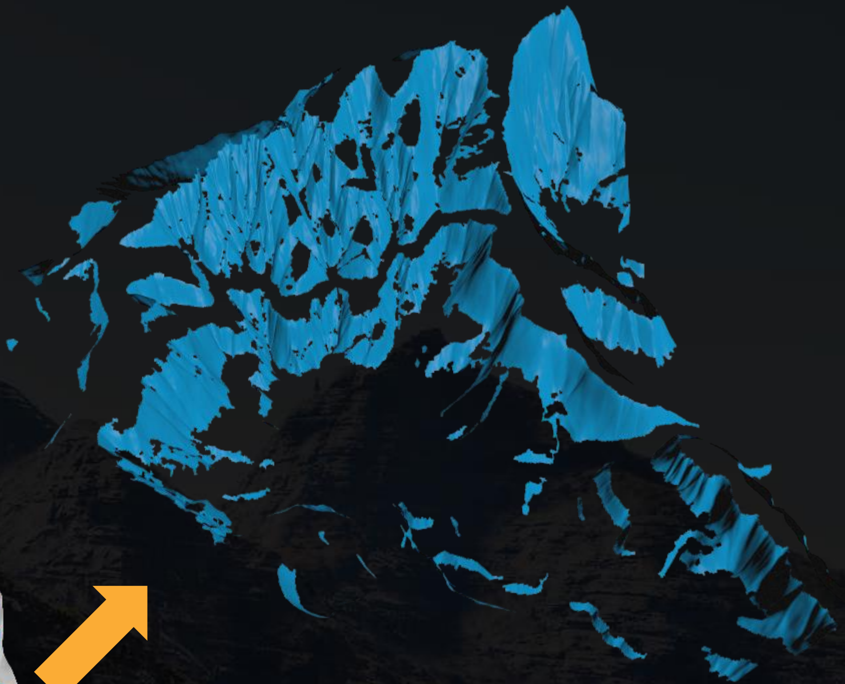


Start up point

Terrain slope


Slope terrain data

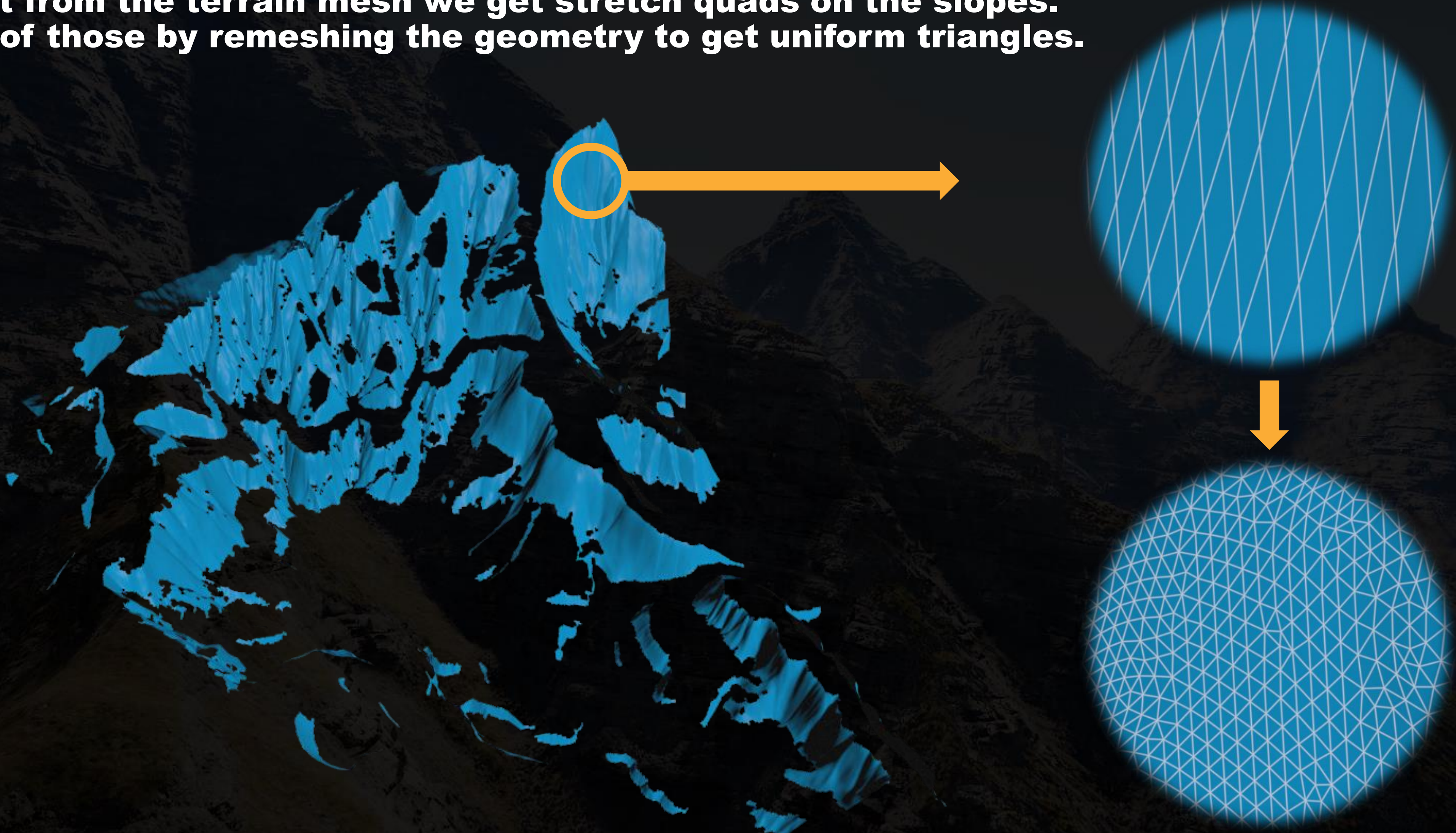

Slope threshold


Cliffs input

Preparing geometry

Remeshing

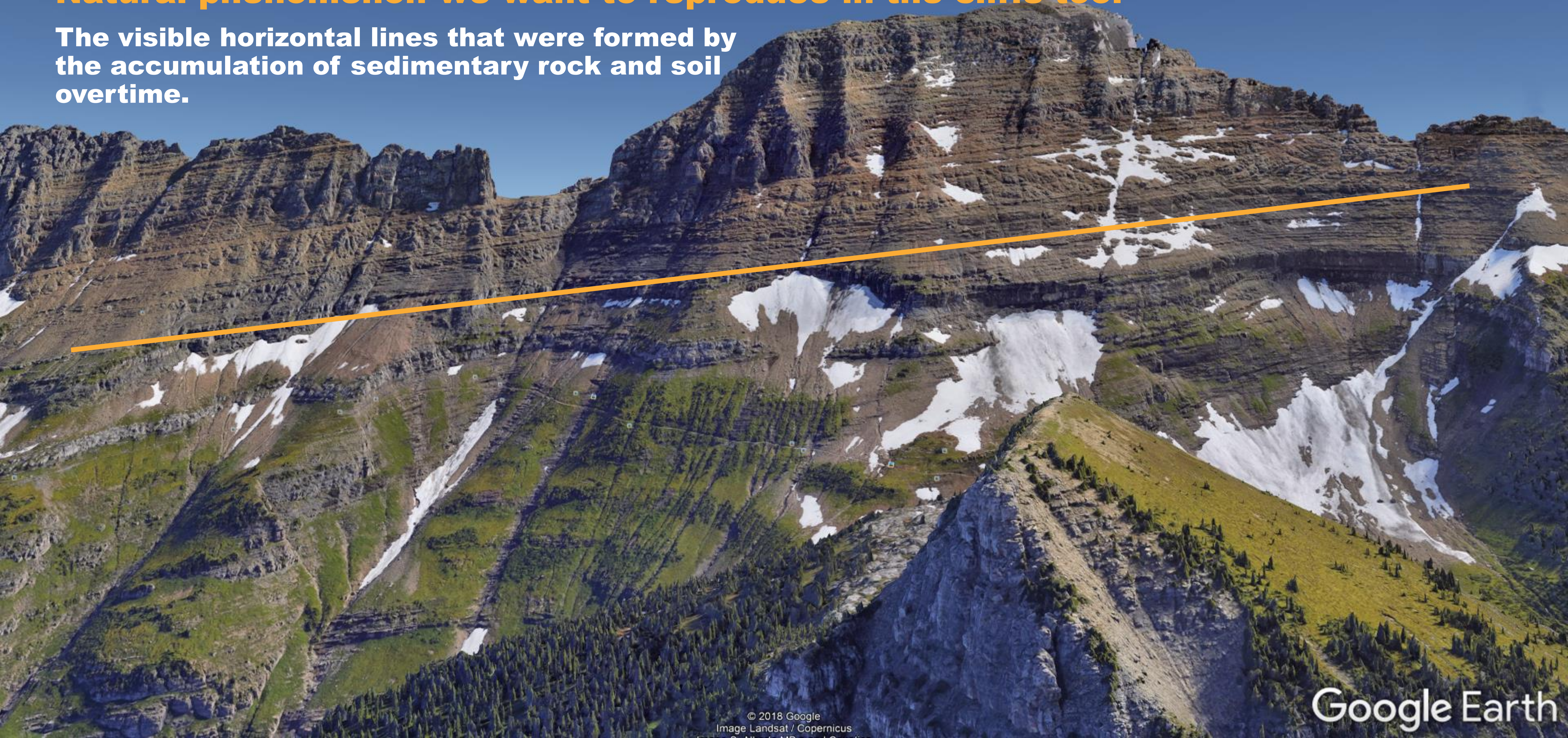
Since we start from the terrain mesh we get stretch quads on the slopes.
So we get rid of those by remeshing the geometry to get uniform triangles.



Geological Stratification

Natural phenomenon we want to reproduce in the cliffs tool

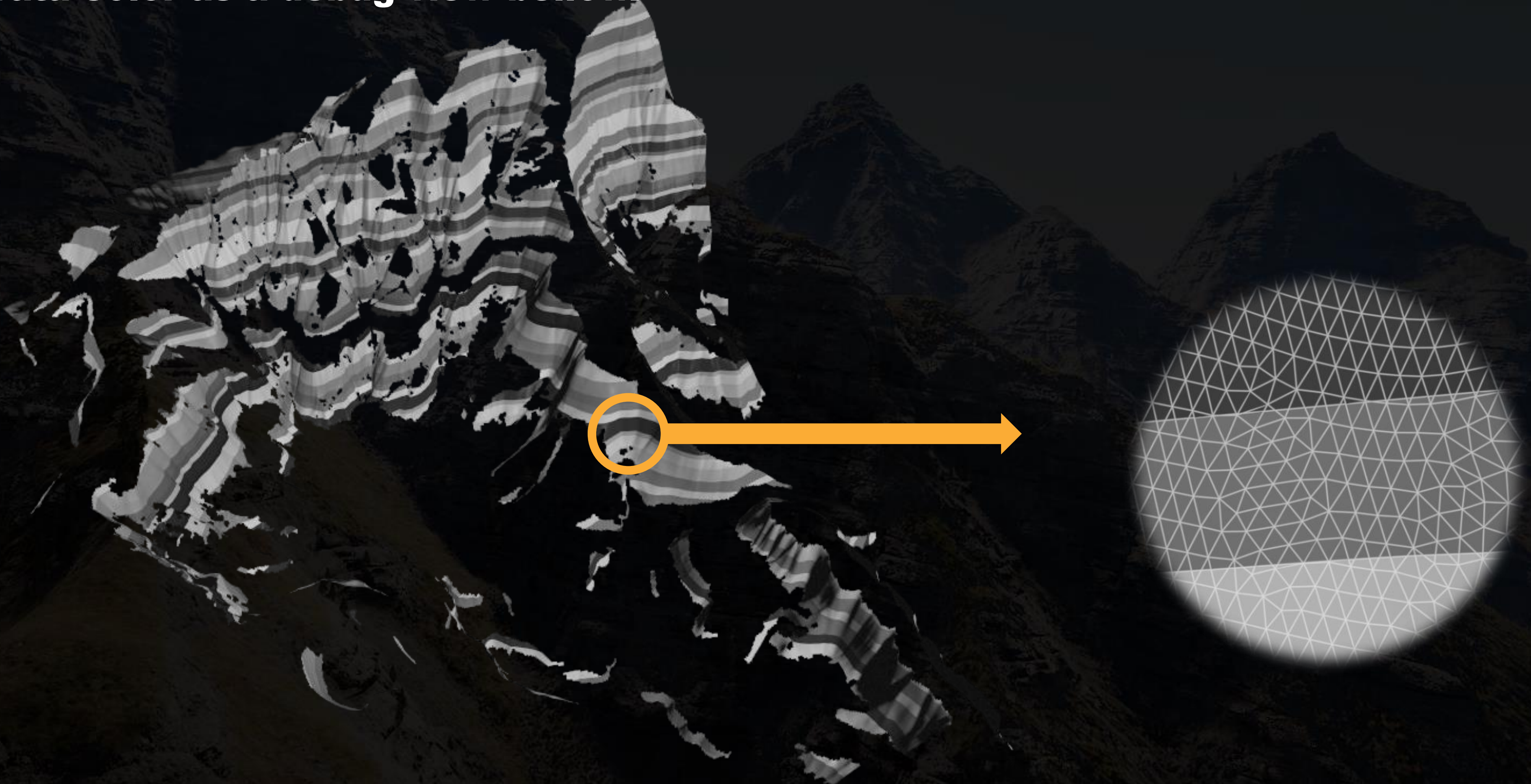
The visible horizontal lines that were formed by the accumulation of sedimentary rock and soil overtime.



Stratification

Creating geologic strata

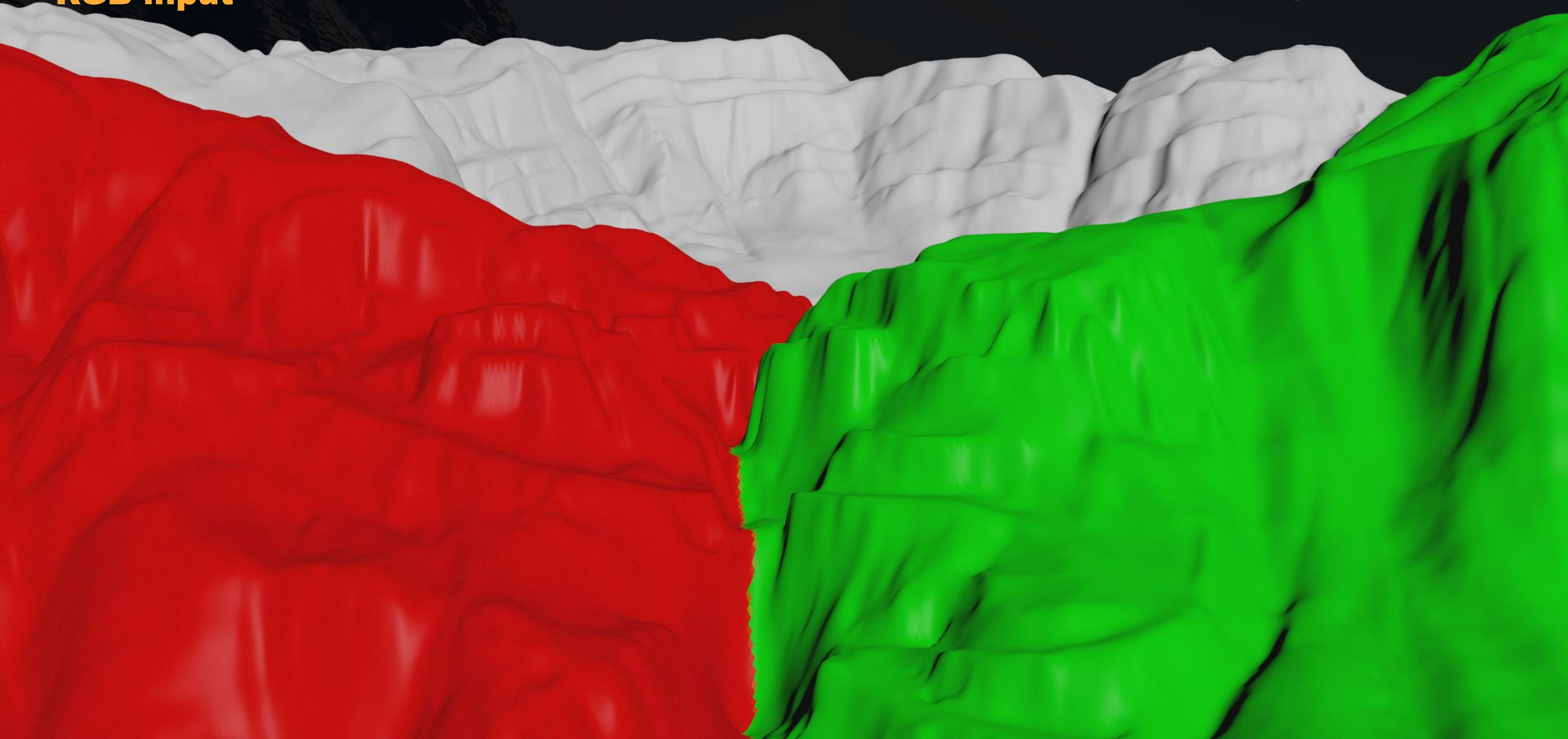
- To create this effect, we have a tool that slices the input geometry into strata chunks.
- Each strata has a random thickness and it assign a strata ID to each slice, which we can use to randomize strata color as a debug view bellow.



Strata angle

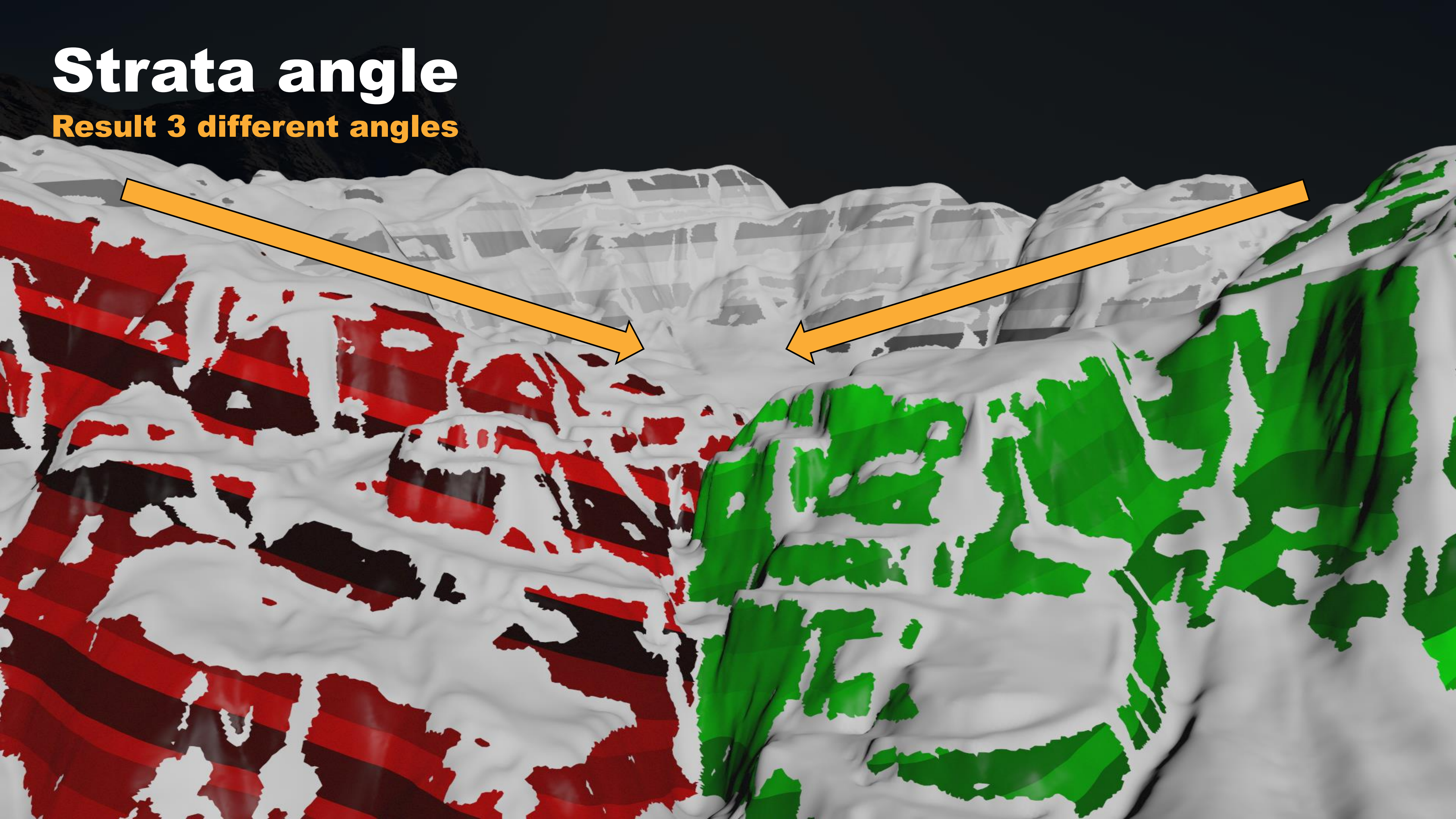
RGB input

- In the editor, we have a few presets to control the strata angles.
- Users will control them with this RGB input on the terrain.
- Houdini then use this to drive the stratification angles parameters.



Strata angle

Result 3 different angles

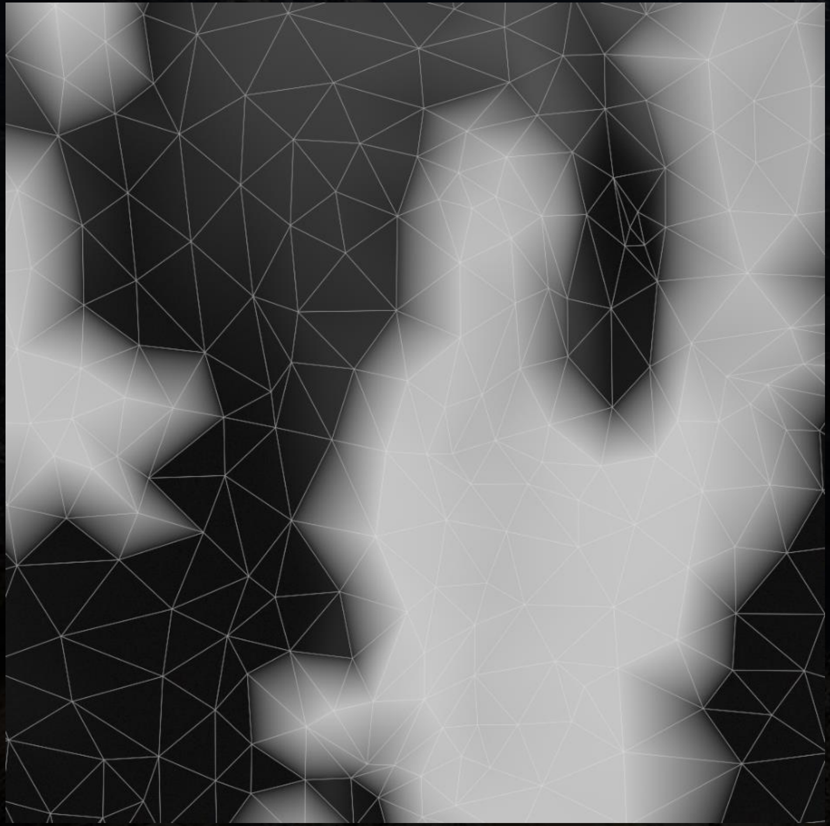


Split noise

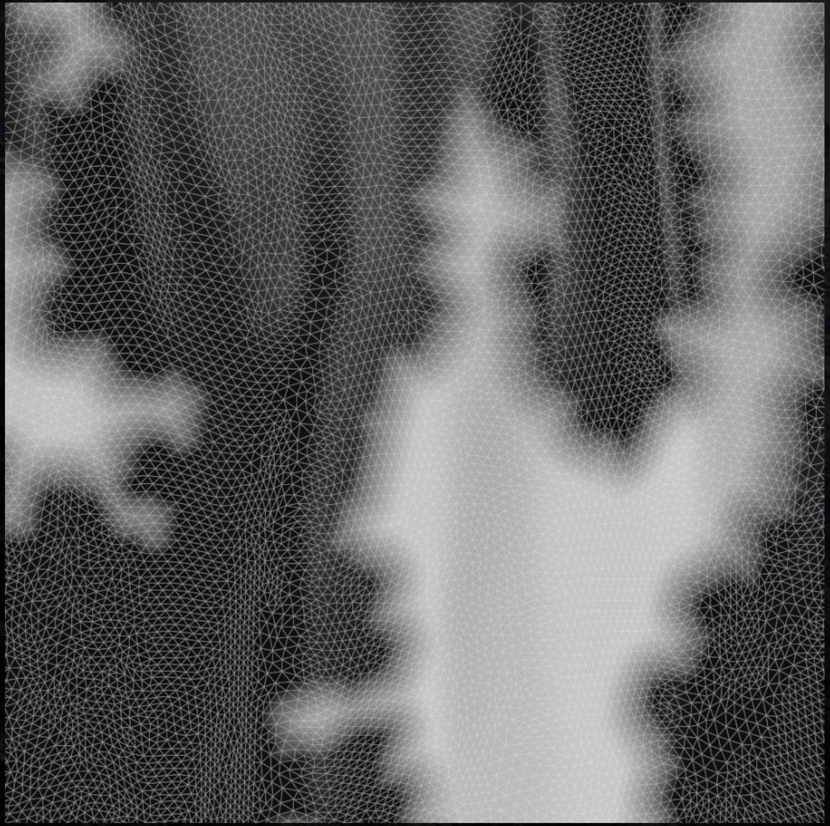
To break strata lines

The strata lines are too perfect and unnatural, so to break them up and bring chaos we generate a noise.

The noise is generated on a lower resolution mesh and transferred to the hires mesh to get larger and blockier patterns.

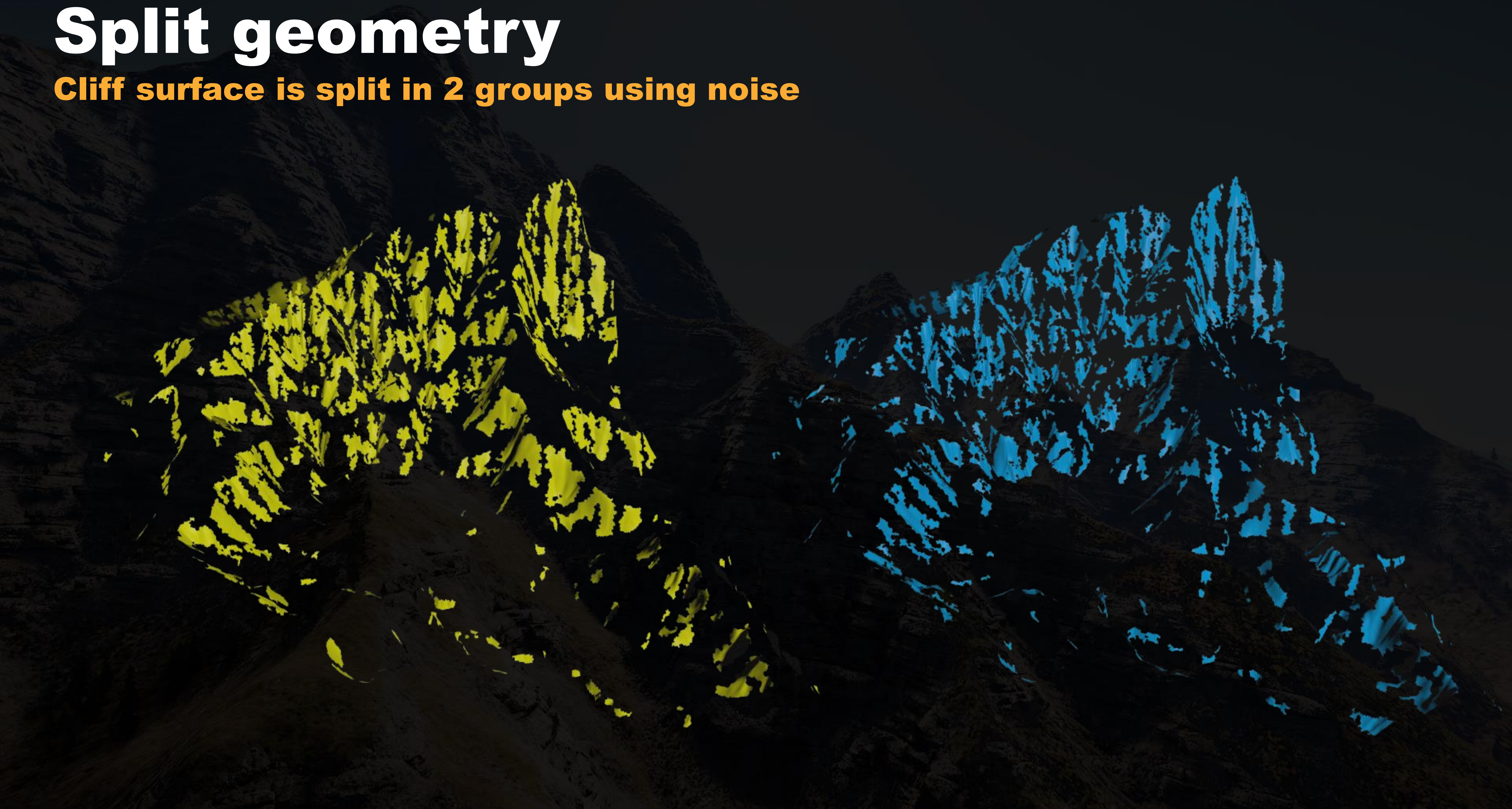


→



Split geometry

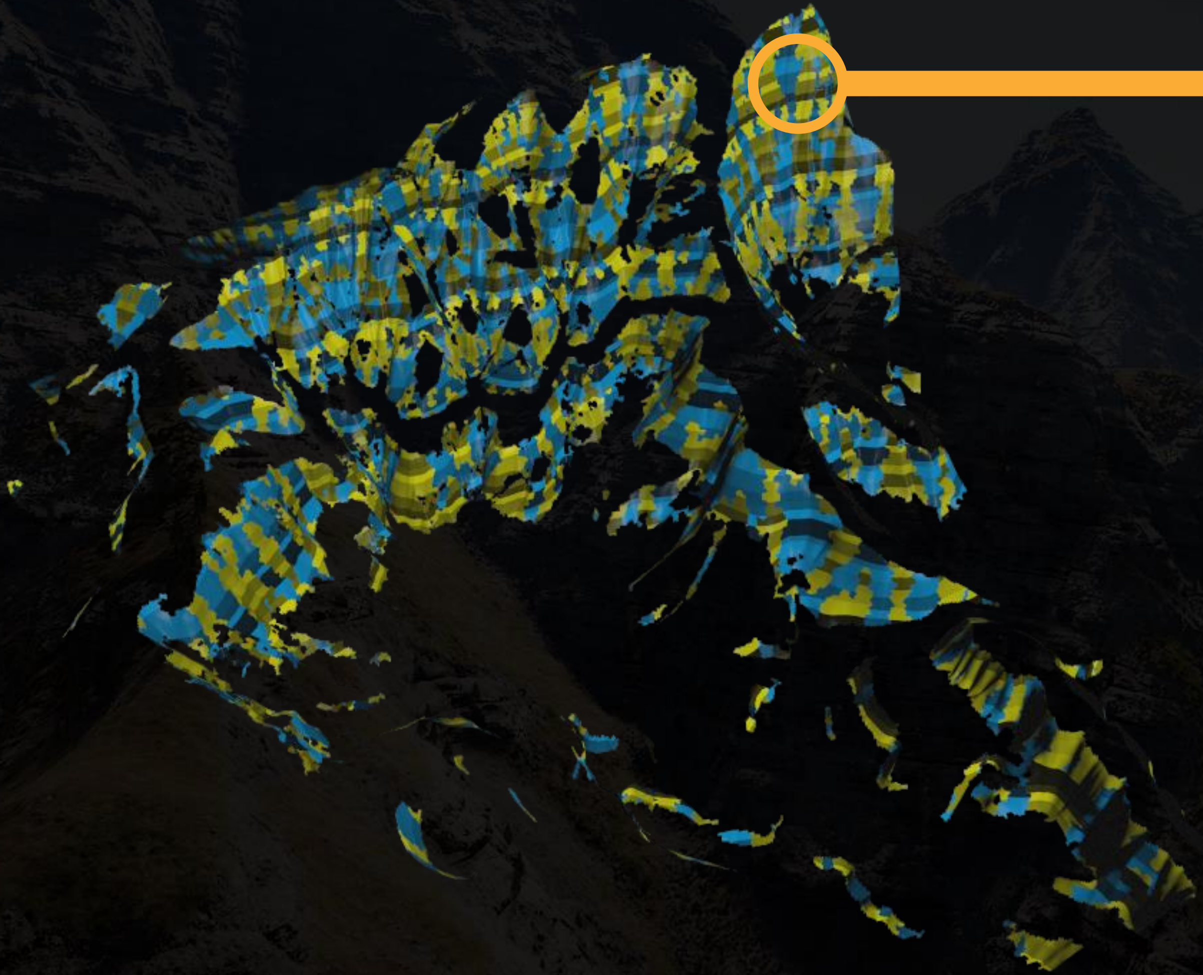
Cliff surface is split in 2 groups using noise



Stratification

Creating geologic strata

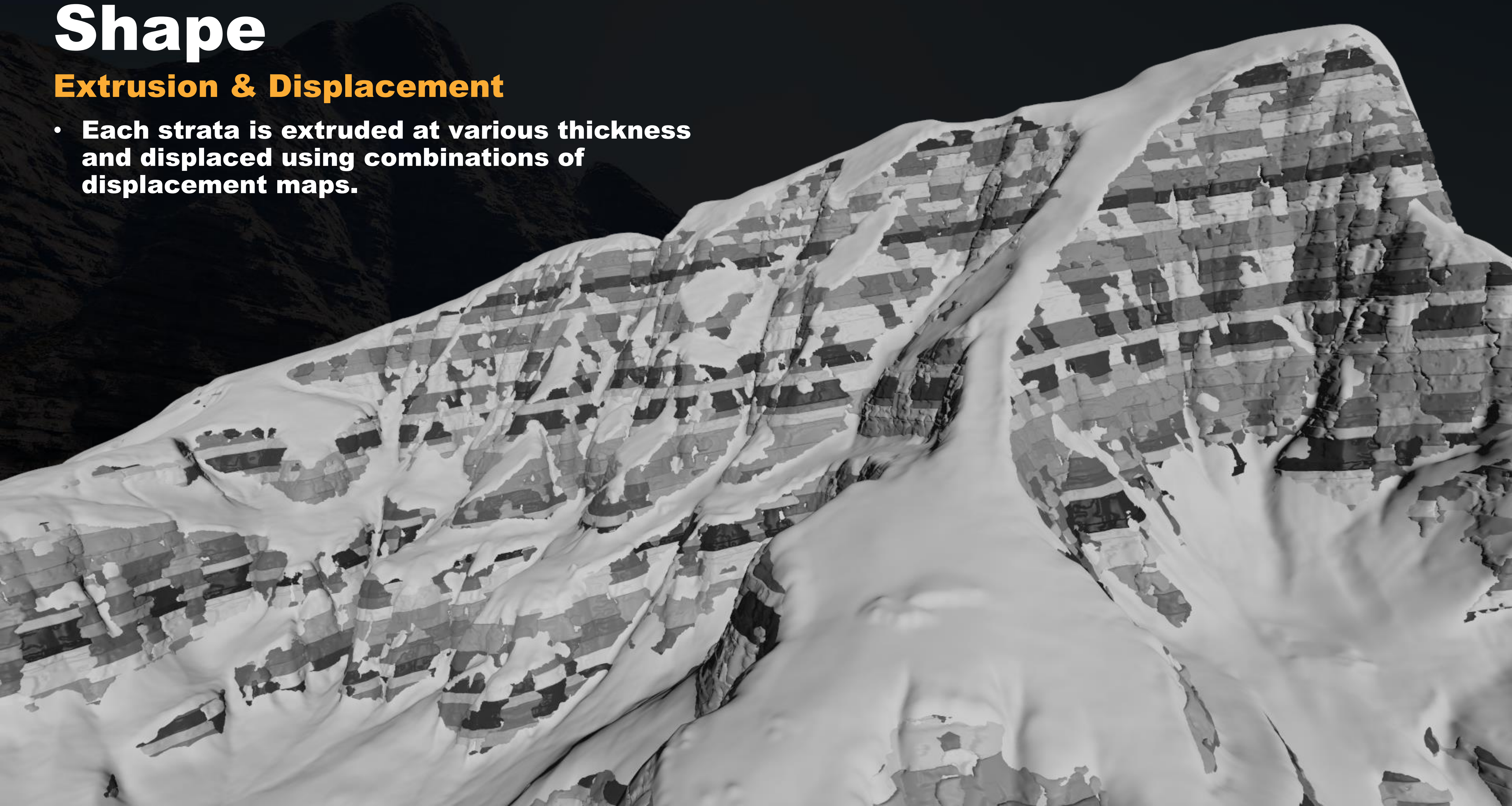
The stratification tool is run on both groups with different seed value to break up our strata lines.



Shape

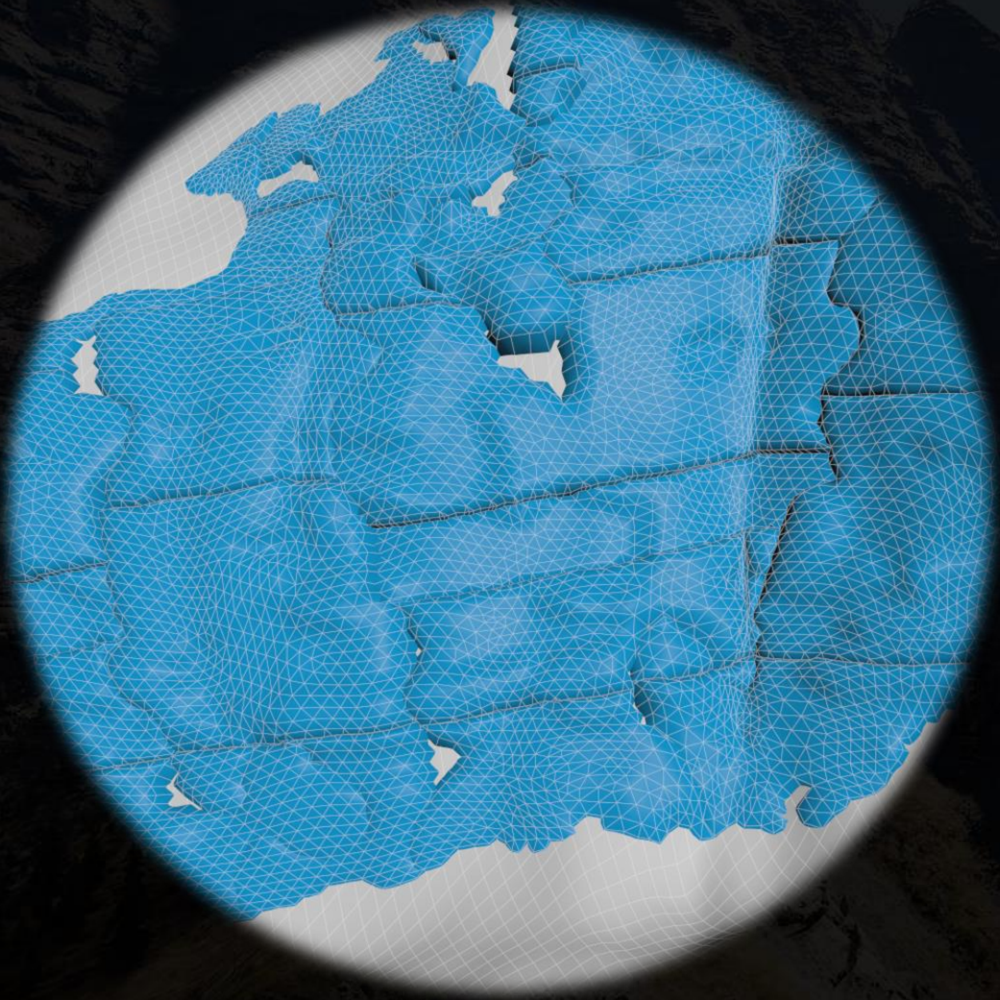
Extrusion & Displacement

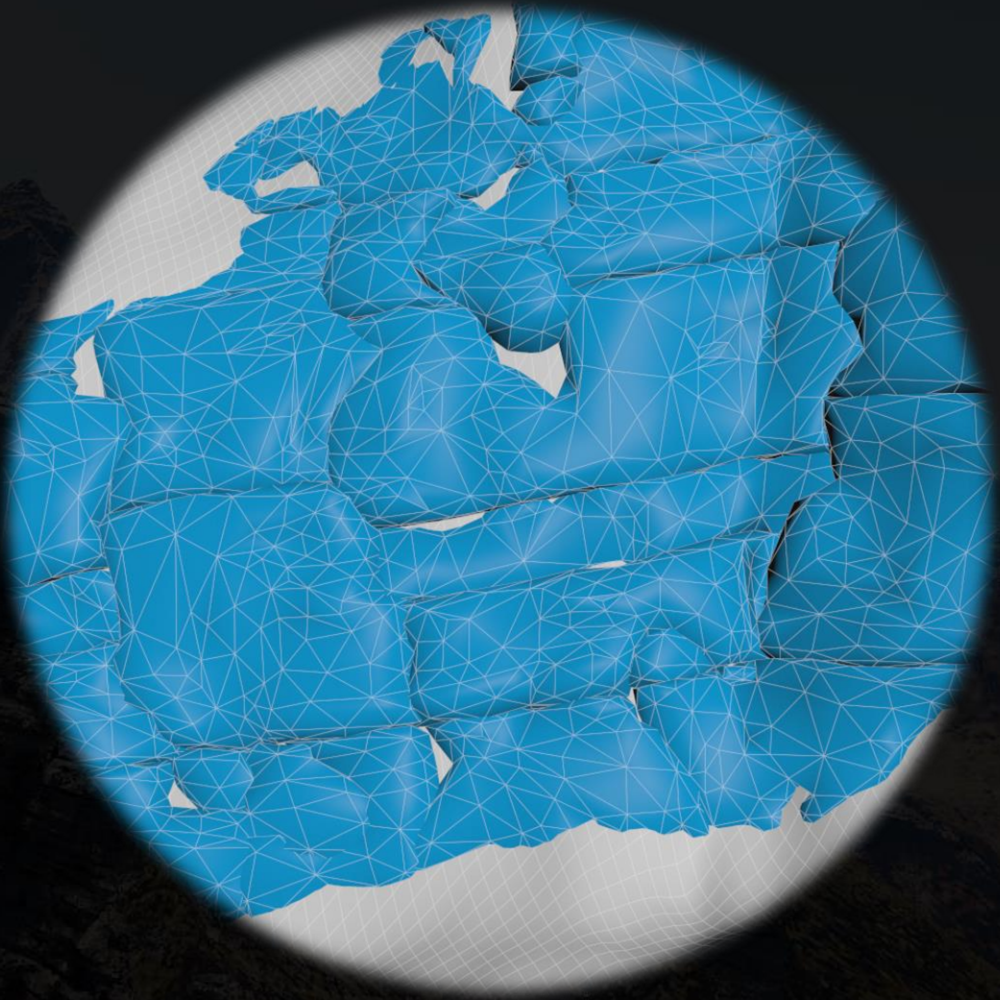
- Each strata is extruded at various thickness and displaced using combinations of displacement maps.



Optimizing geometry

Reducing triangle count

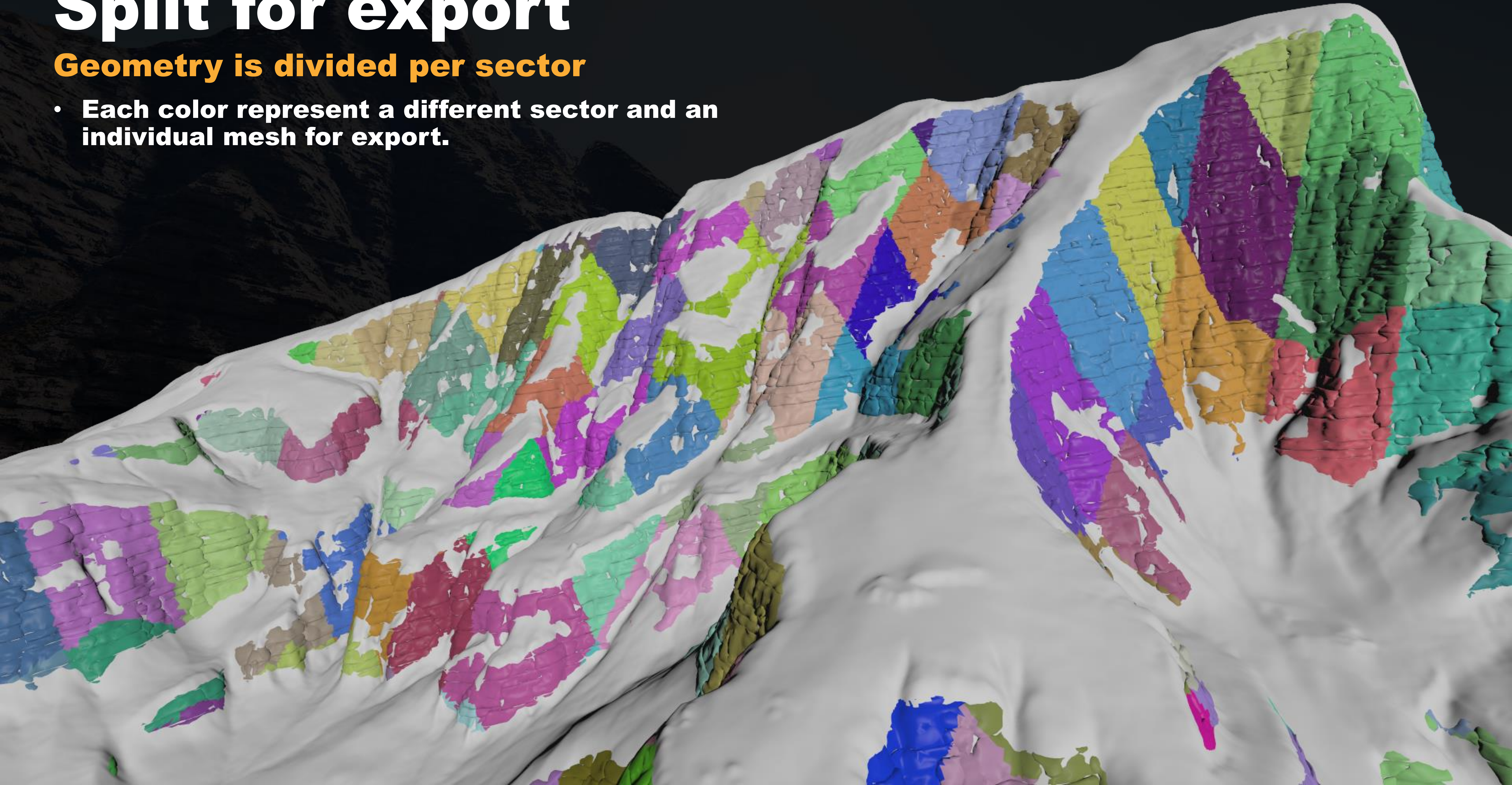




Split for export

Geometry is divided per sector

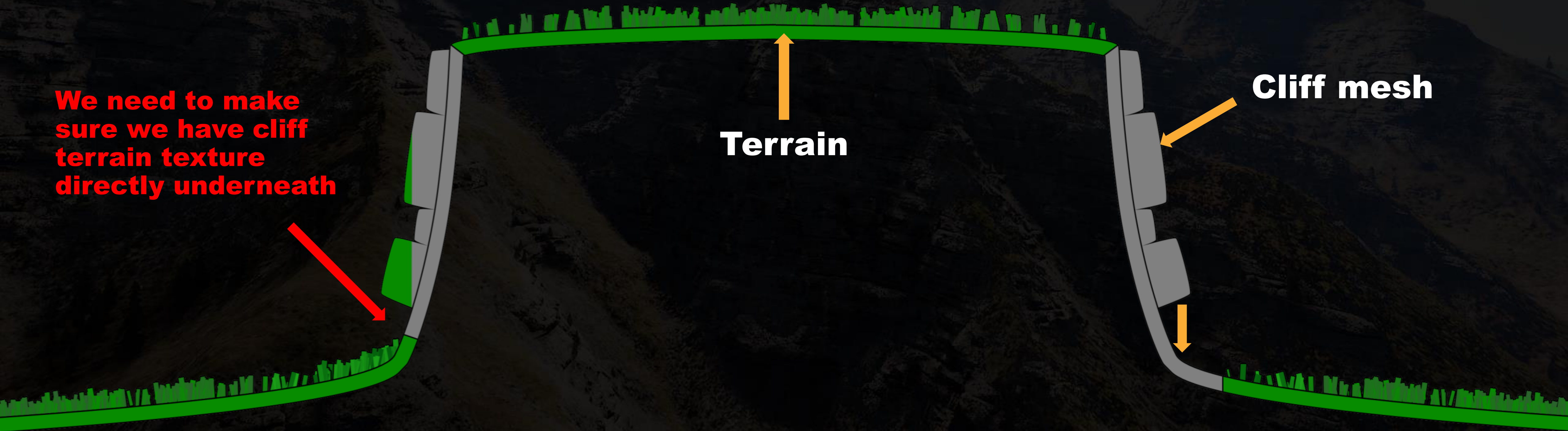
- **Each color represent a different sector and an individual mesh for export.**



Shading

Cliffs shader

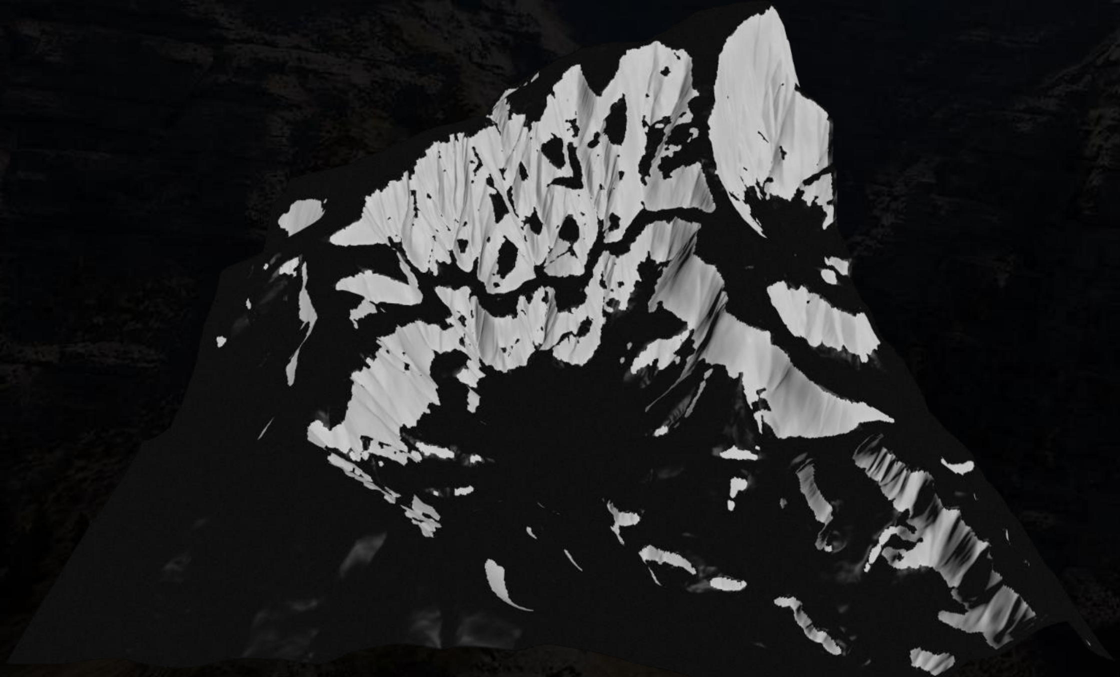
- No UVs required
- X Y Z texture projections
- X&Y have a projection angle setting per texture (elevation & rotation)
- Picks up the terrain texture ID and color


We need to make sure we have cliff terrain texture directly underneath
Terrain
Cliff mesh

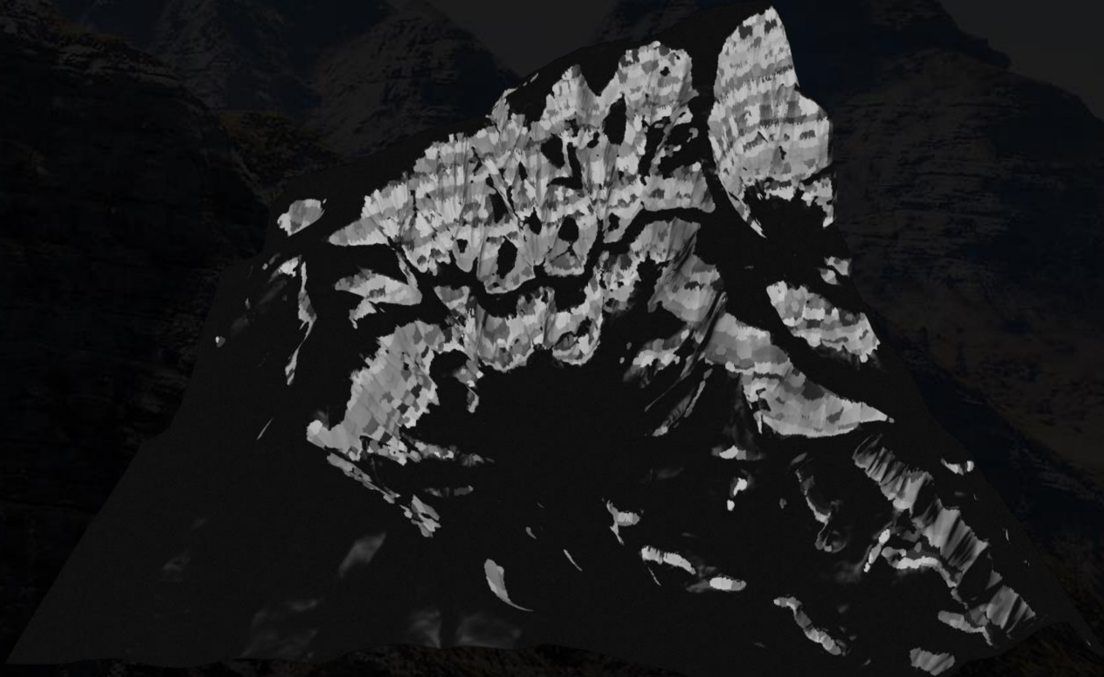
Cliffs terrain data

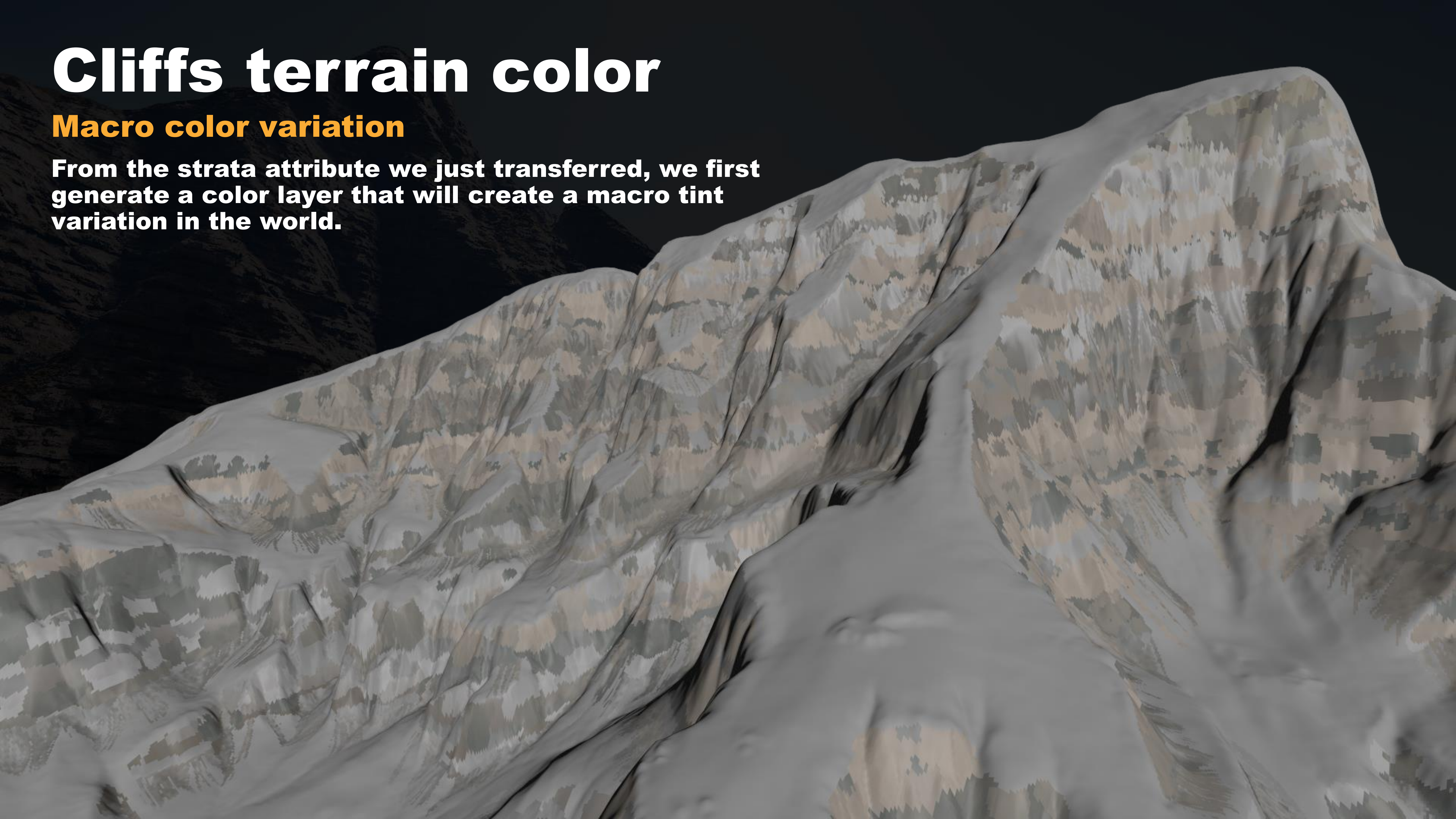
Cliffs mesh attributes transferred back to the terrain

- Displaced Cliffs mask



- Strata attribute





Cliffs terrain color

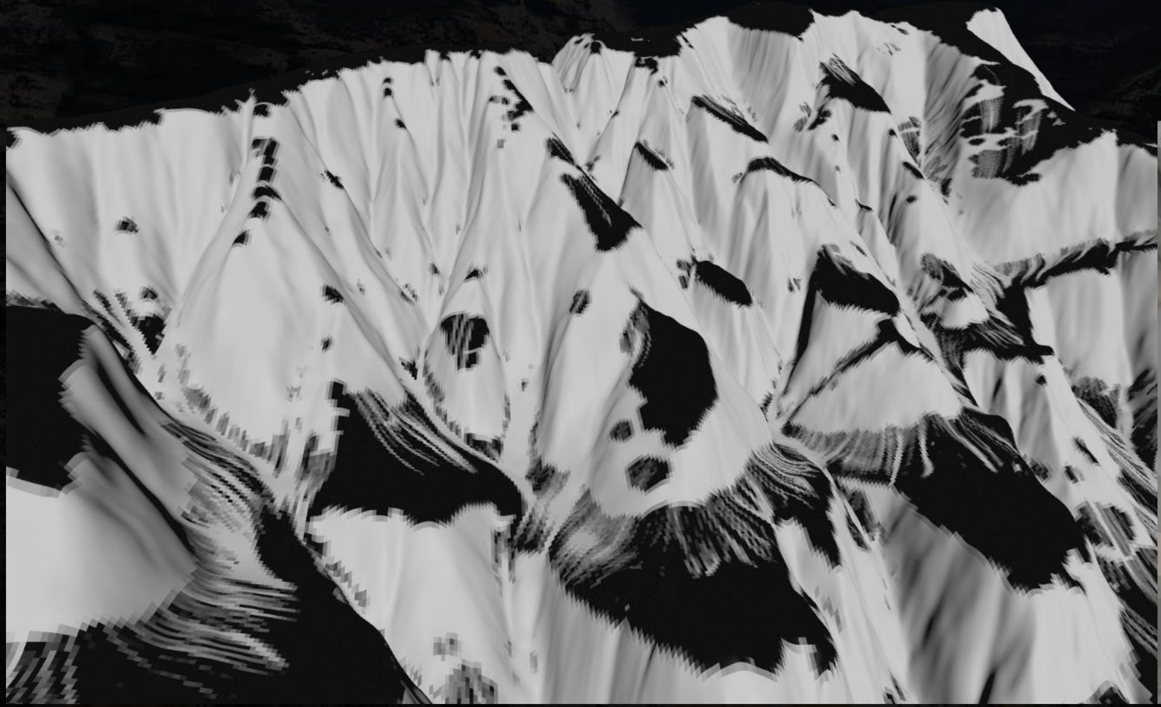
Macro color variation

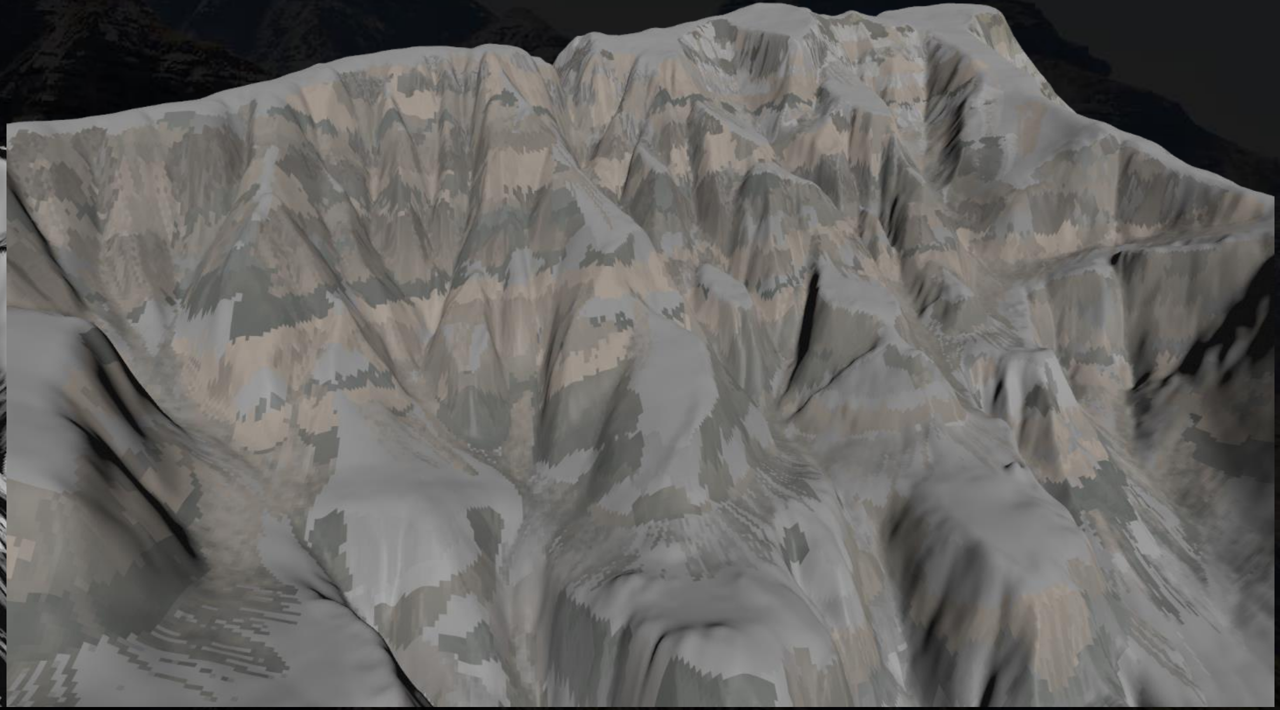
From the strata attribute we just transferred, we first generate a color layer that will create a macro tint variation in the world.

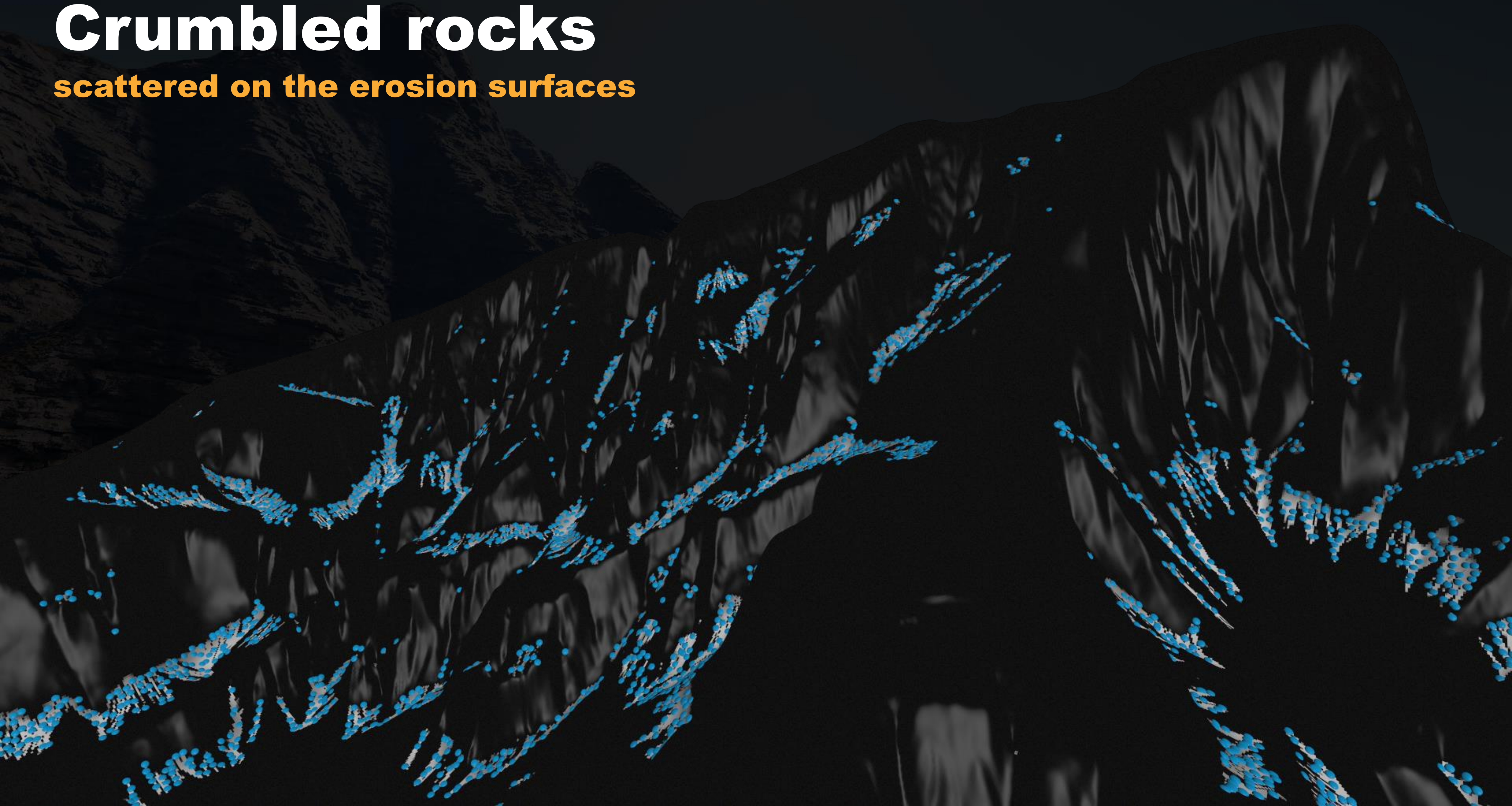
Cliffs Erosion

Flow simulation

From the cliff mask we transferred to the terrain, we further extend the cliffs by running a flow simulation. Points scattered on the cliff surface will flow down the slope to create an erosion effect. The origin strata color is retained on the erosion areas.







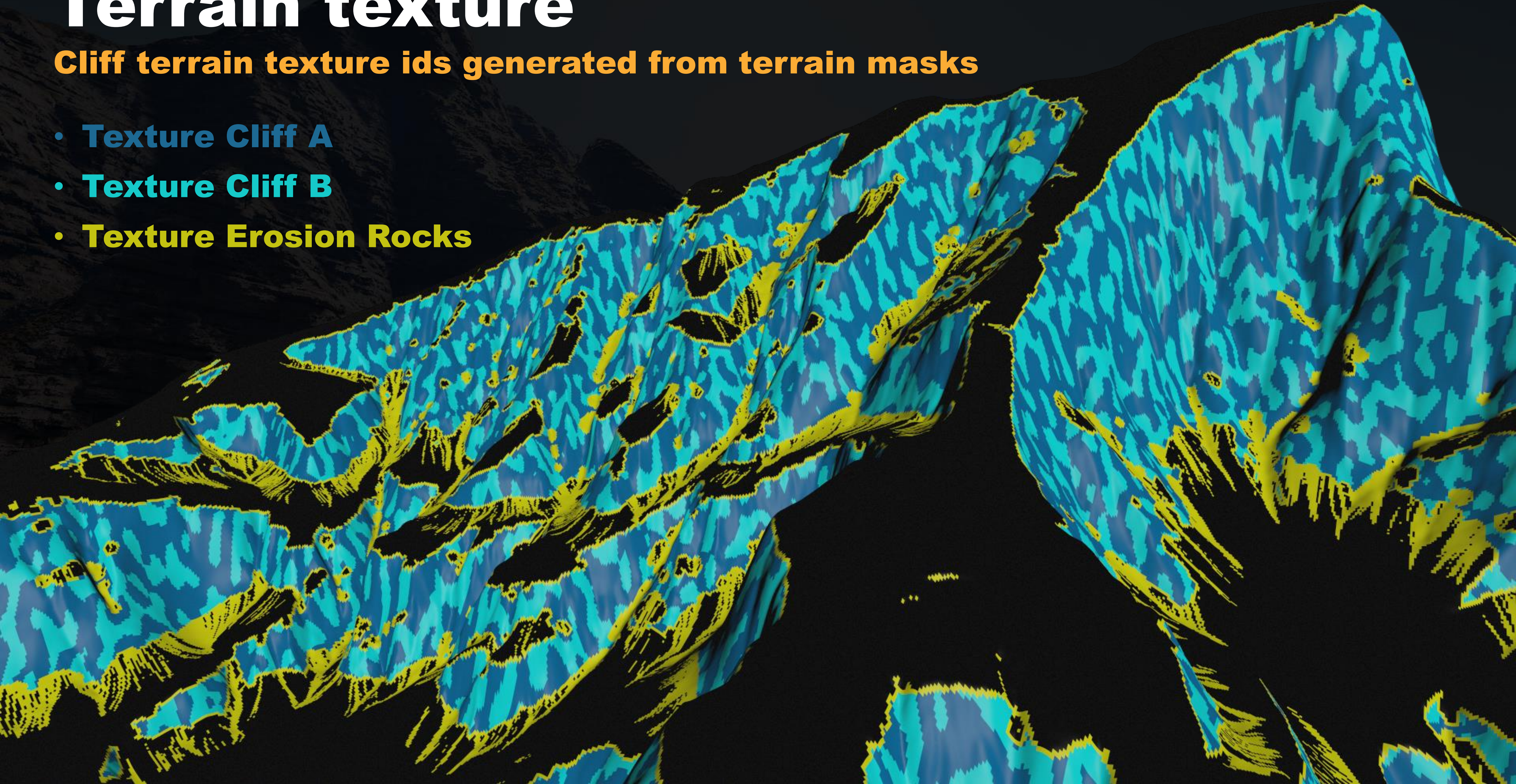
Crumbled rocks

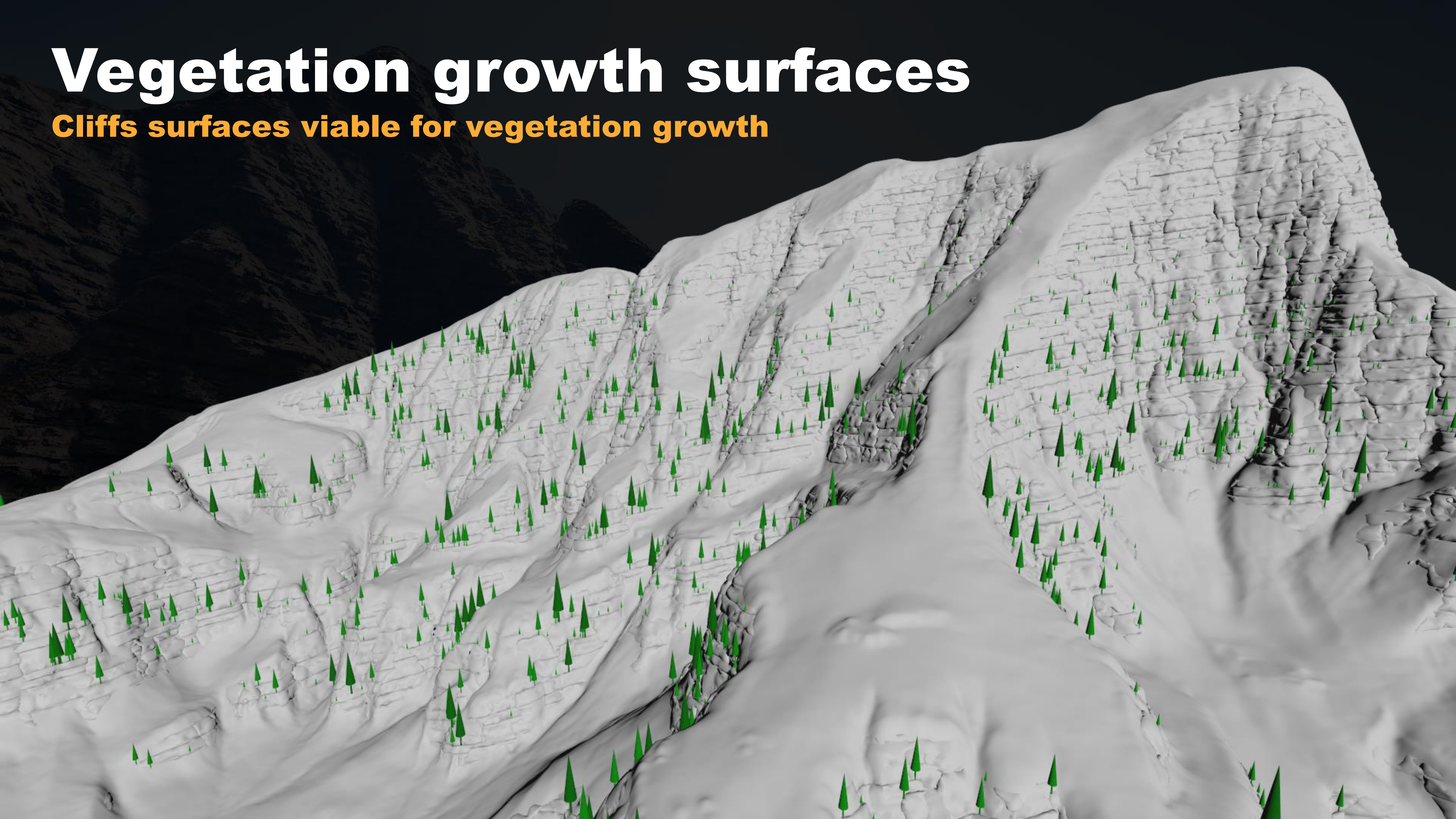
scattered on the erosion surfaces

Terrain texture

Cliff terrain texture ids generated from terrain masks

- Texture Cliff A
- Texture Cliff B
- Texture Erosion Rocks



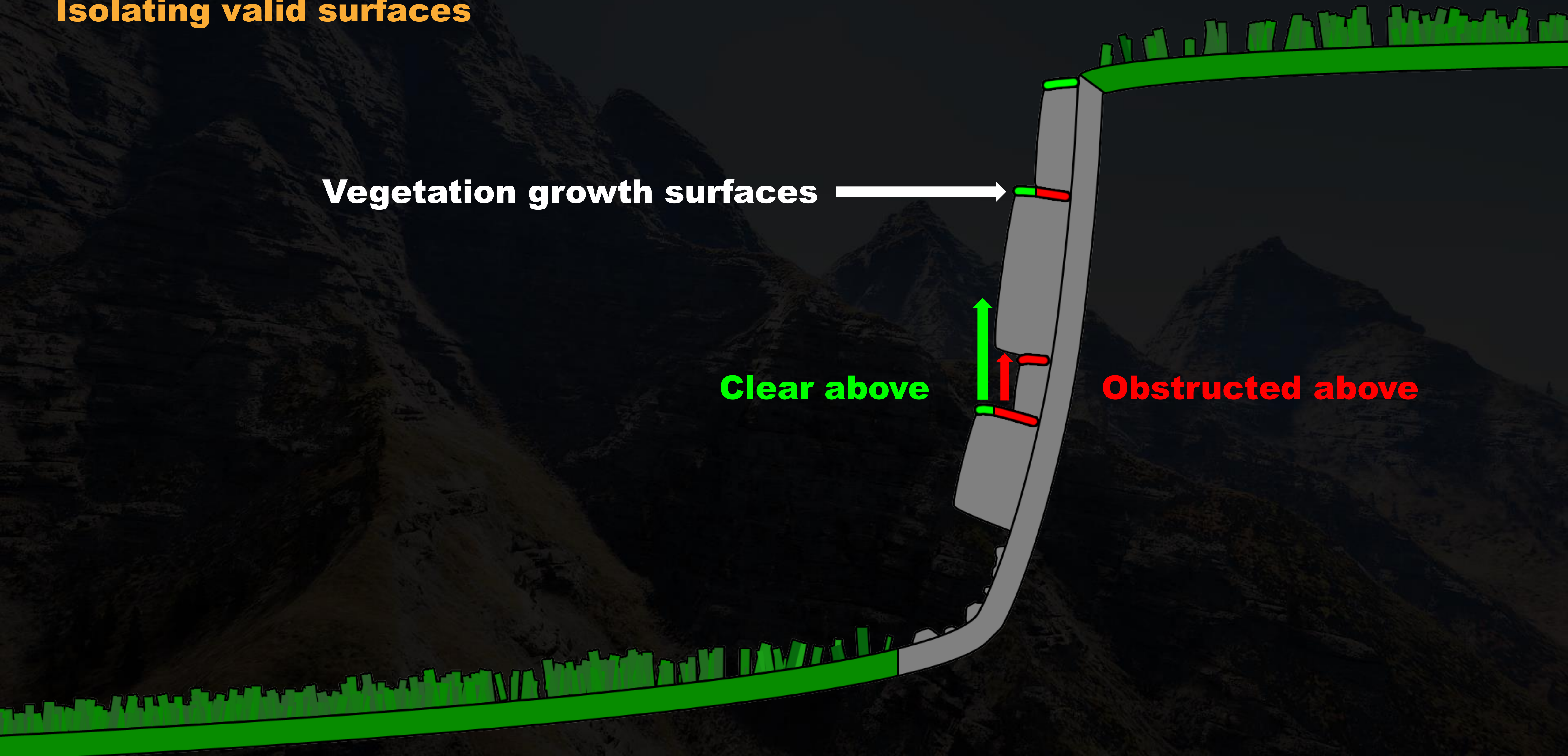


Vegetation growth surfaces

Cliffs surfaces viable for vegetation growth

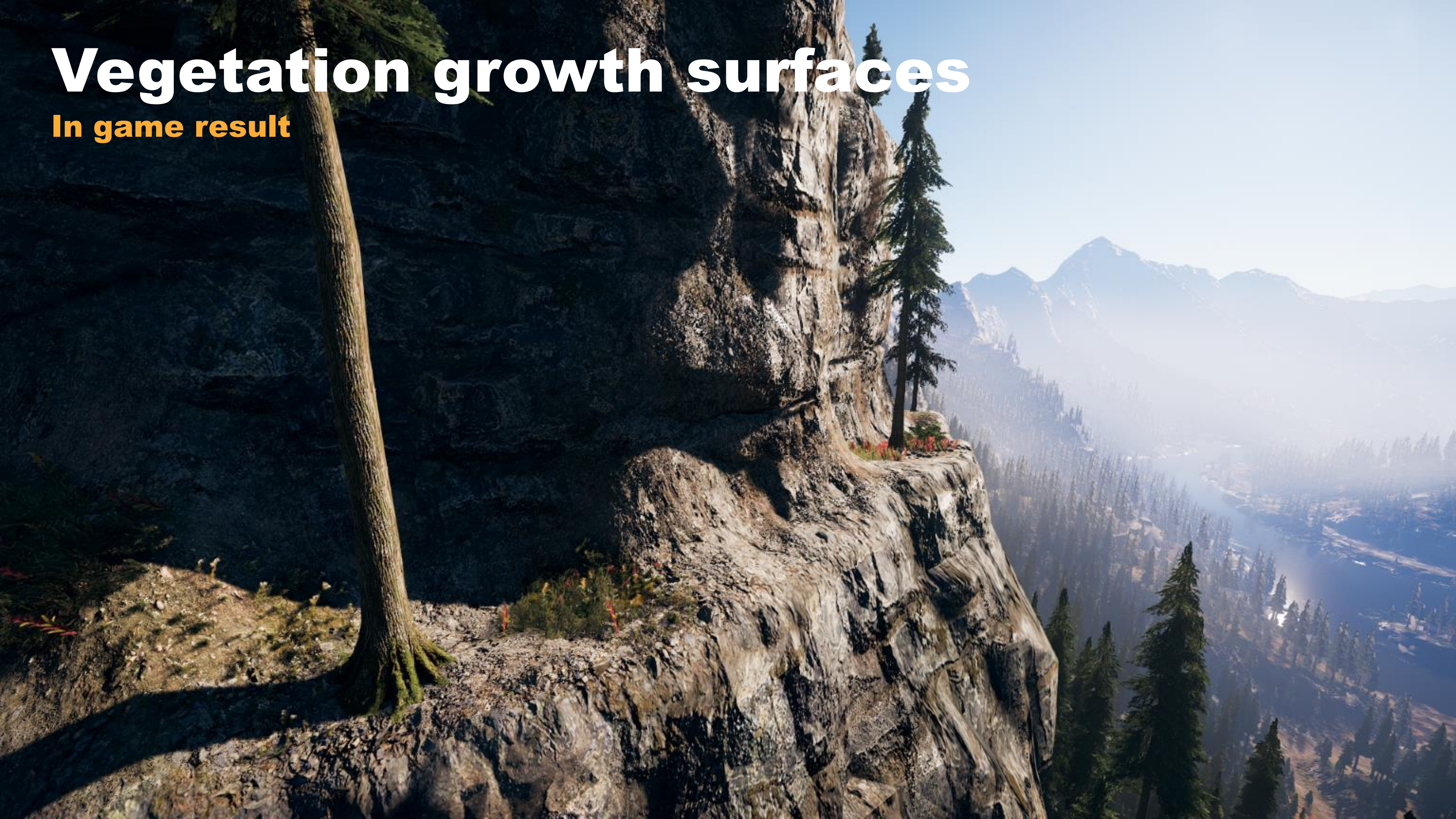
Vegetation growth surfaces

Isolating valid surfaces



Vegetation growth surfaces

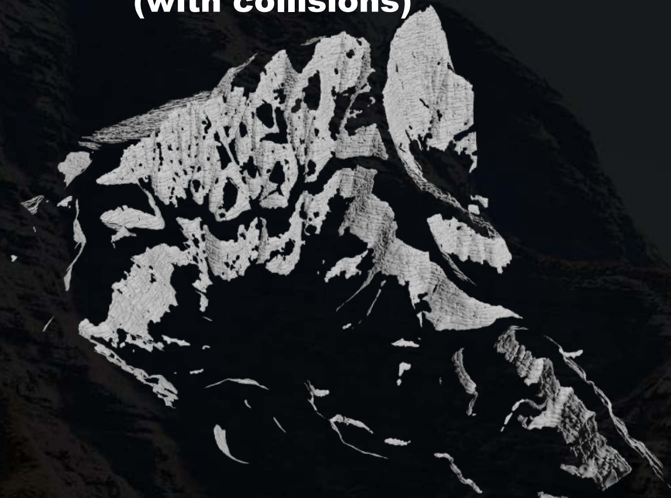
In game result



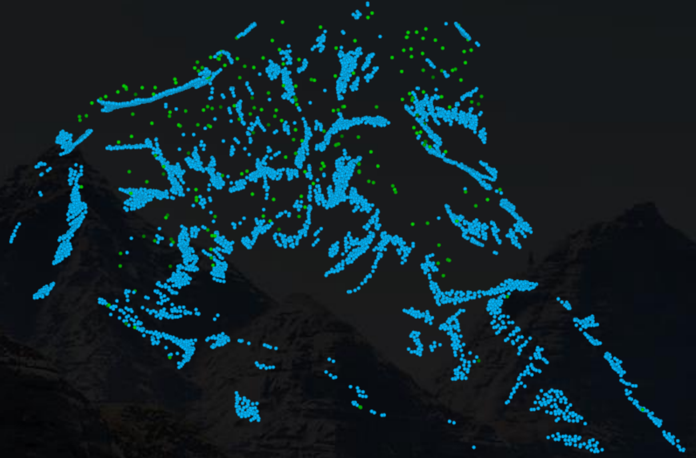
Export

Data exported to editor

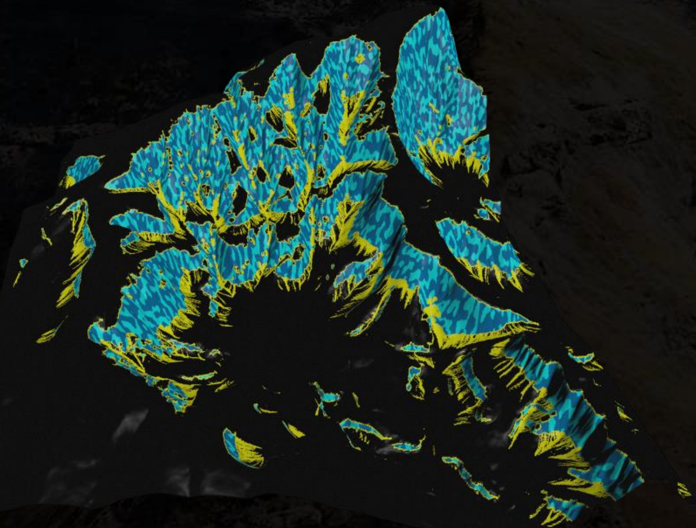
Cliffs geometry
(with collisions)



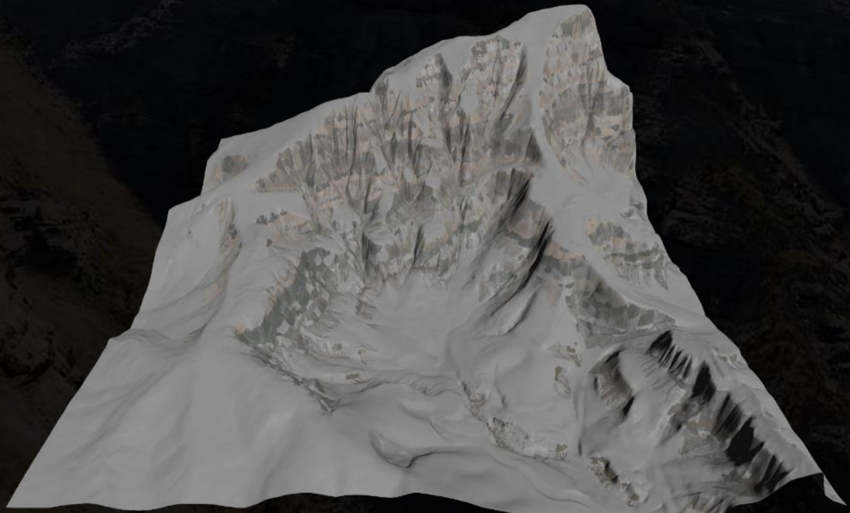
Entities point cloud
(rocks & vegetation)



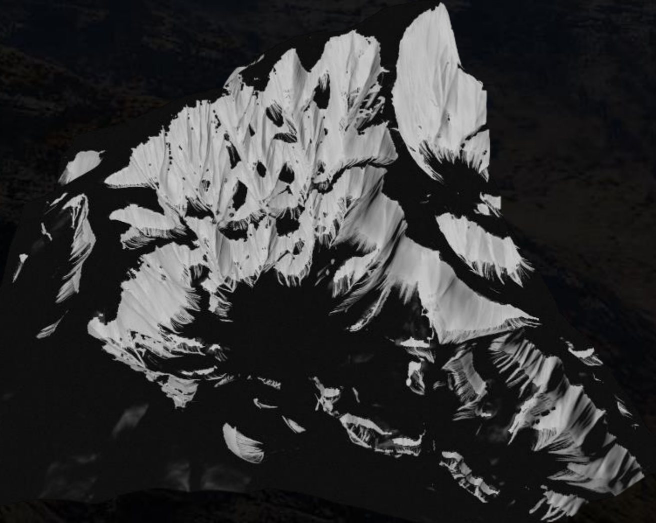
Terrain Texture IDs



2D Cliffs color



2D Cliffs mask



7- THE BIOME TOOL

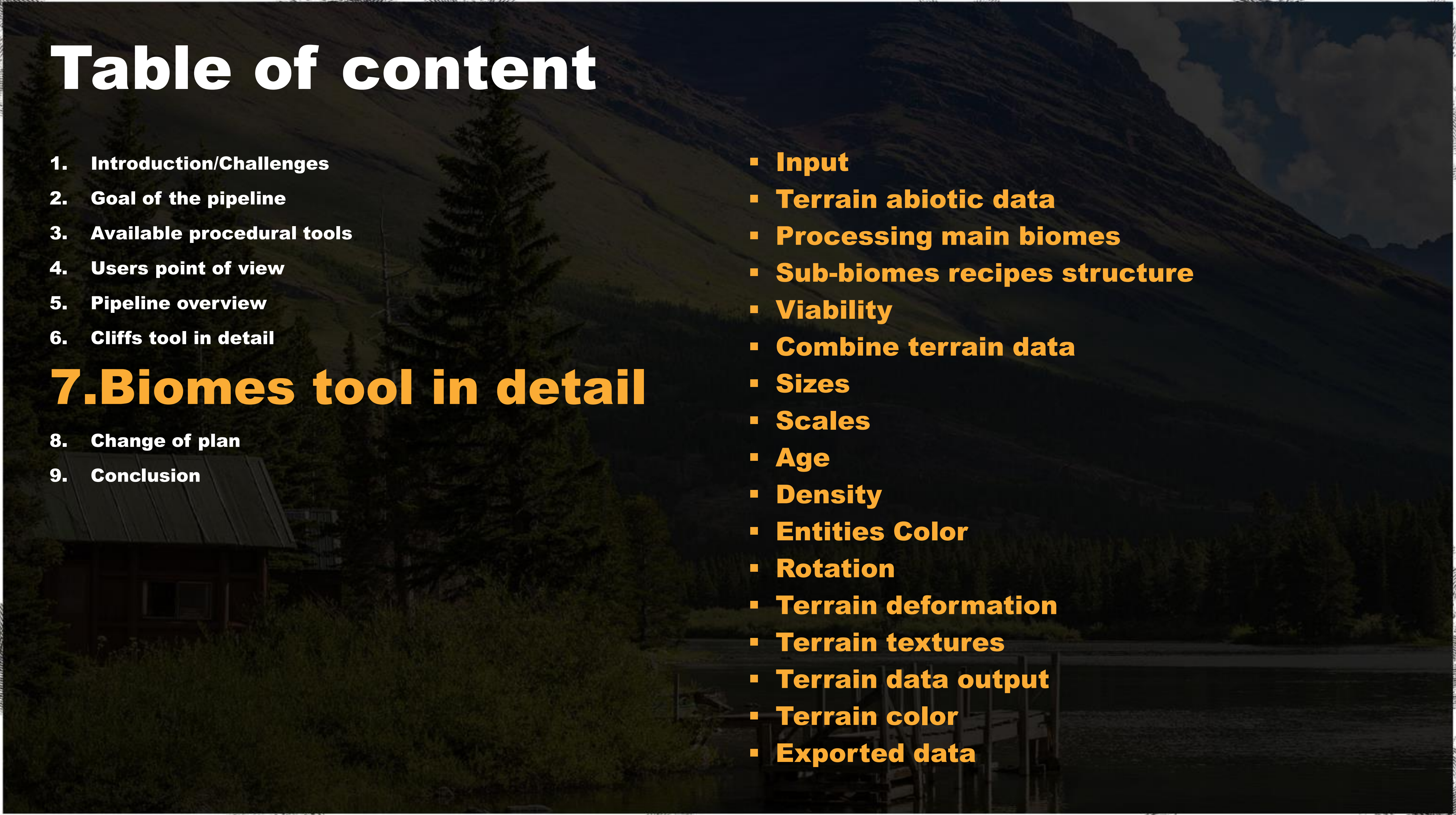


Table of content

Generate Terrain

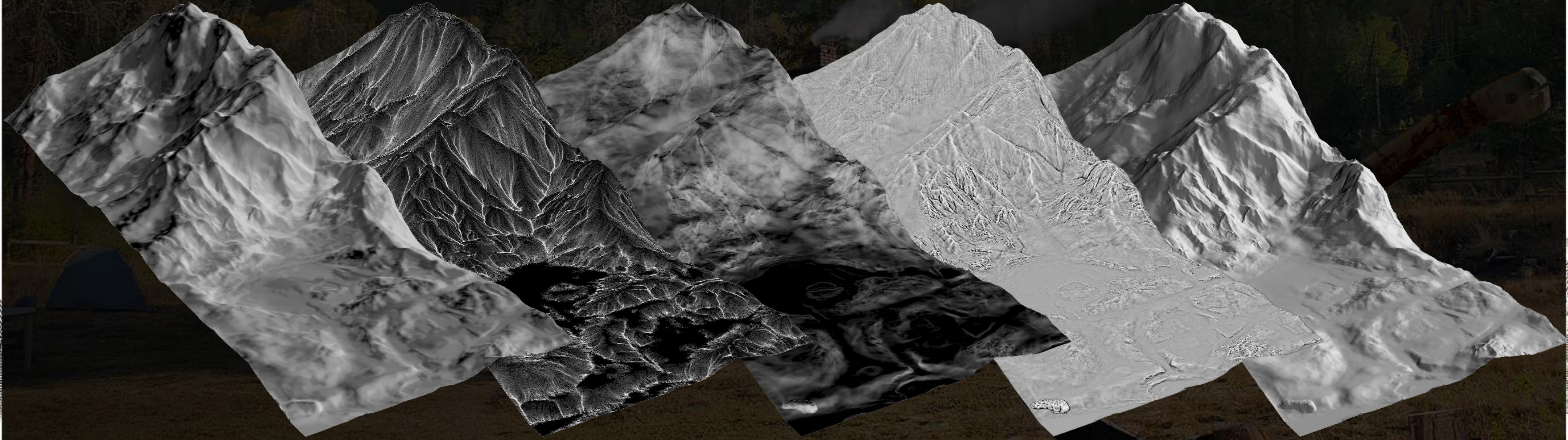
From Heightmap

Generate Terrain Abiotic Data

Physical features of the land that are generated from terrain topology.

- Those terrain attributes will become the pillars to most biomes recipes.

- Occlusion
- Flow
- Slope
- Curvature
- Illumination

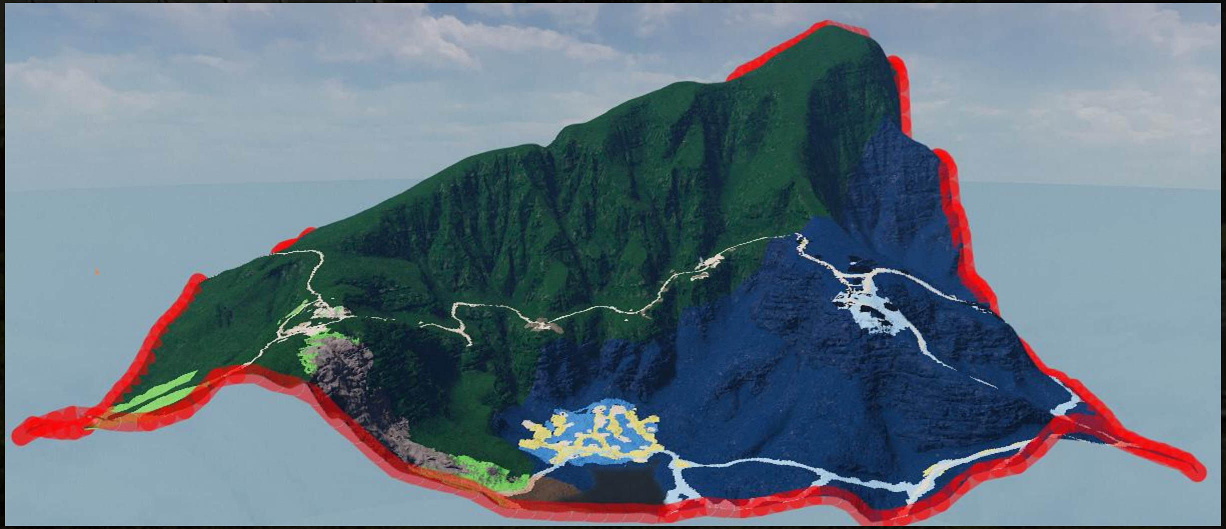


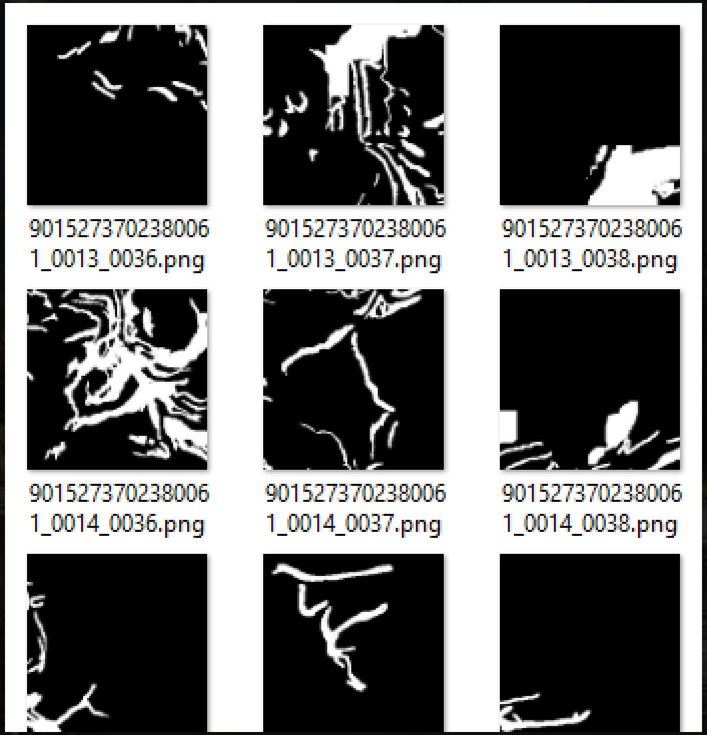
Altitude - Latitude - Longitude - Wind

Importing 2D data

Loading as terrain attributes from PNGs on disk

- Biome Painter data
- Procedurally generated data
  - Freshwater masks
  - Roads masks
  - Fences mask
  - Power lines mask
  - Cliffs mask



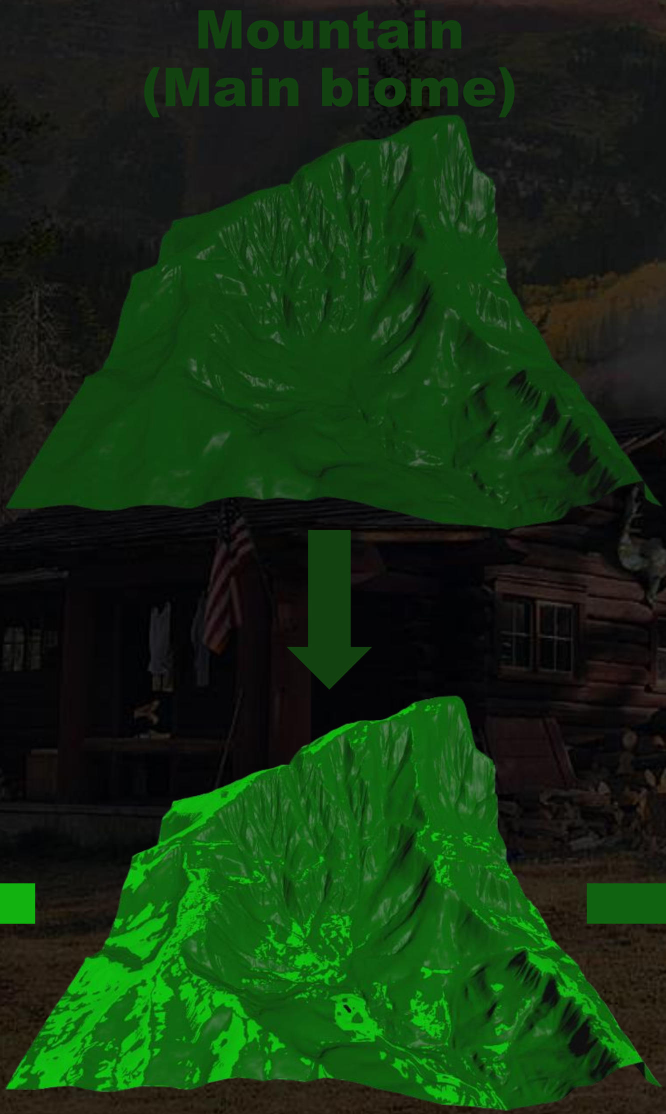


9015273702380061_0013_0036.png
9015273702380061_0013_0037.png
9015273702380061_0013_0038.png
9015273702380061_0014_0036.png
9015273702380061_0014_0037.png
9015273702380061_0014_0038.png

Process Main Biomes

Biome and Sub-Biome

- Main biomes are covering the most part of the world, about 75 to 85 %.
- The main biomes will automatically process where the sub-biomes should be based on the abiotic terrain data.



Mountain Grass

(Sub biome)

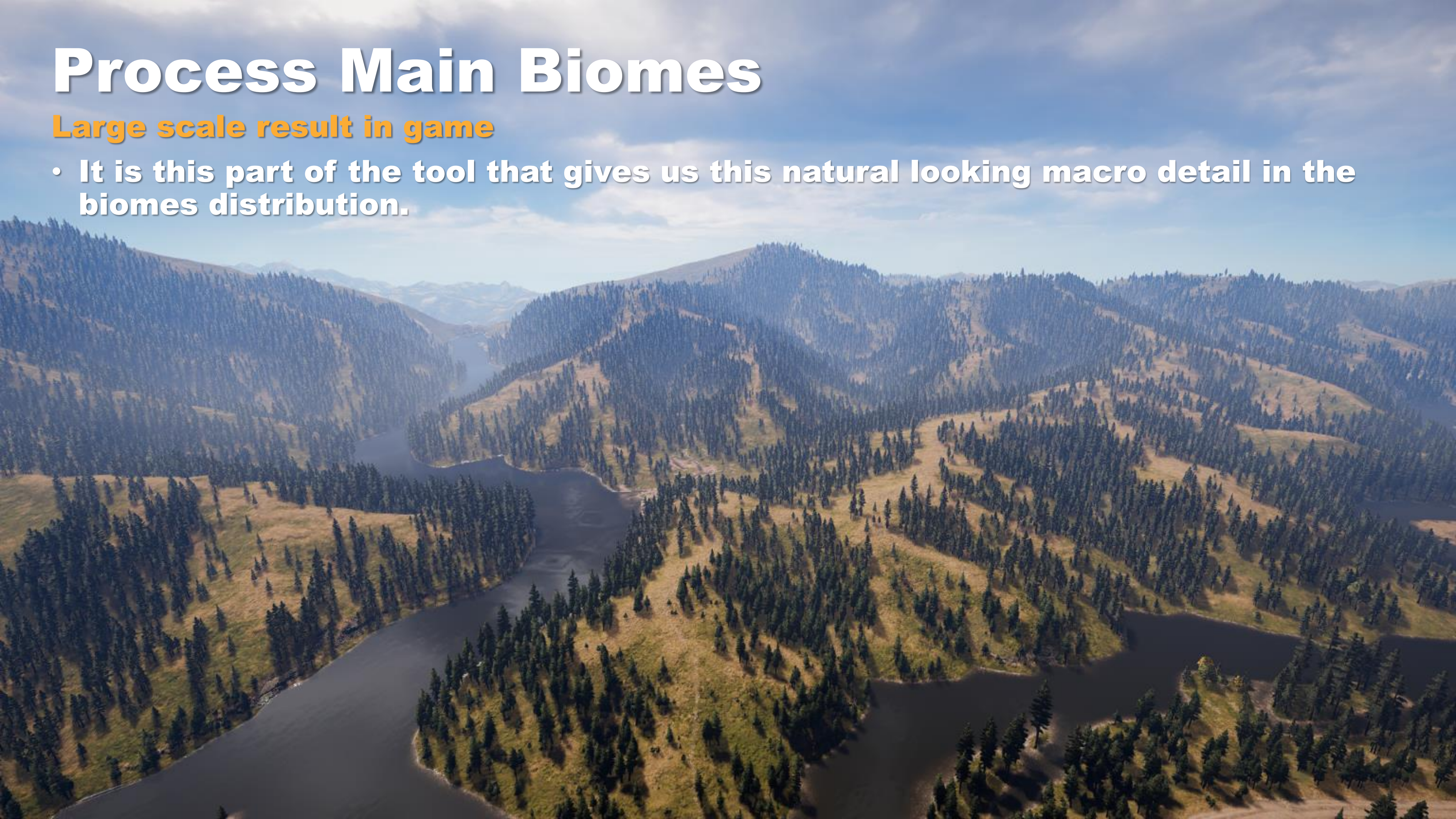
Mountain Forest

(Sub biome)

Process Main Biomes

Large scale result in game

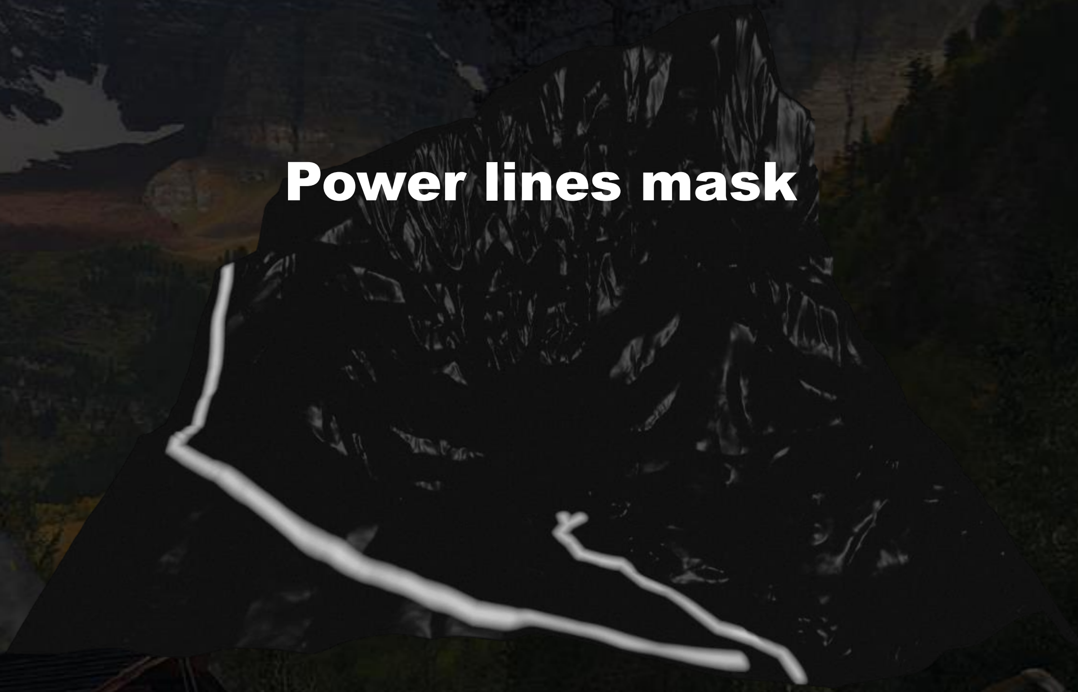
- It is this part of the tool that gives us this natural looking macro detail in the biomes distribution.

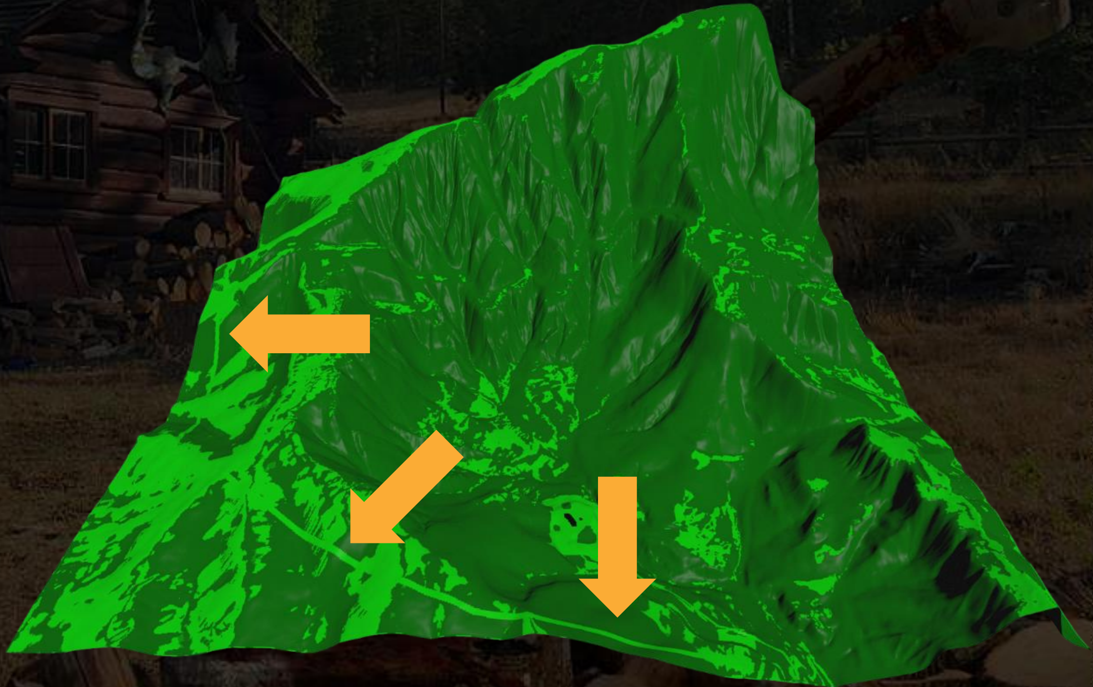


Process Main Biomes

Power lines clearings

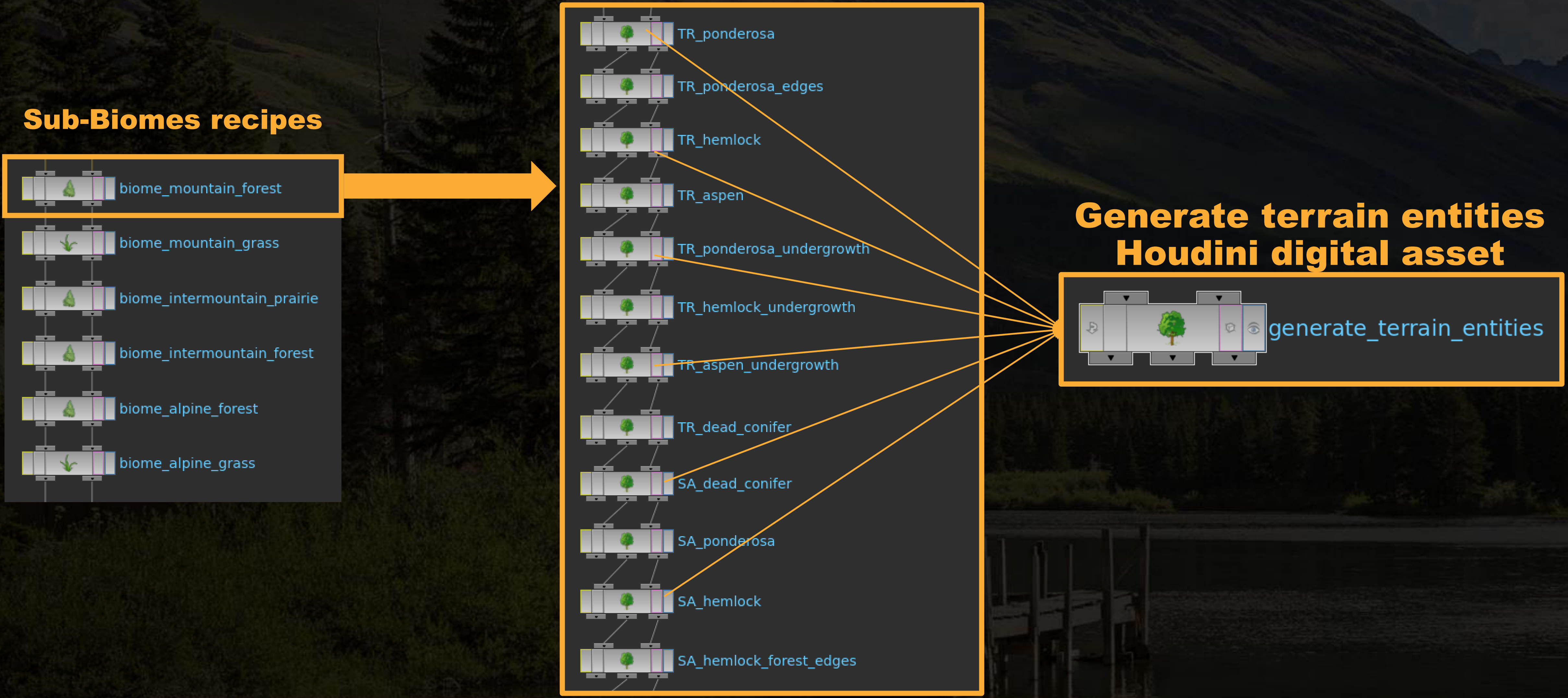
- Main biomes also process other fancy things like replacing forest with grassland where the user placed power lines.





Sub-Biomes Recipes

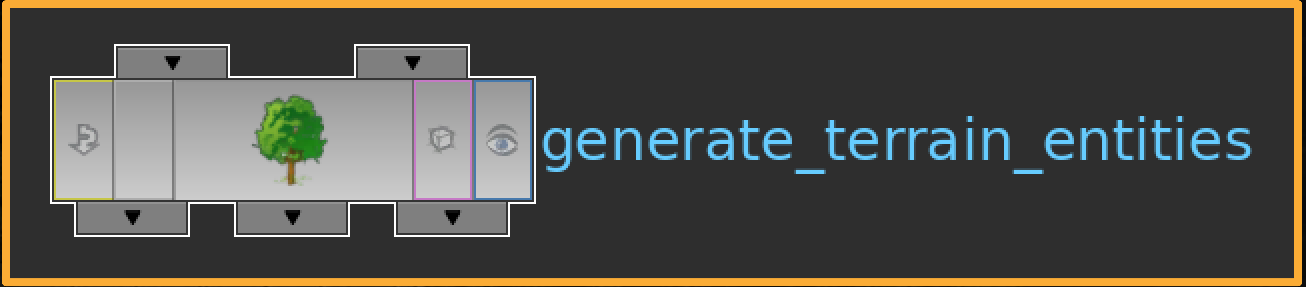
Structure

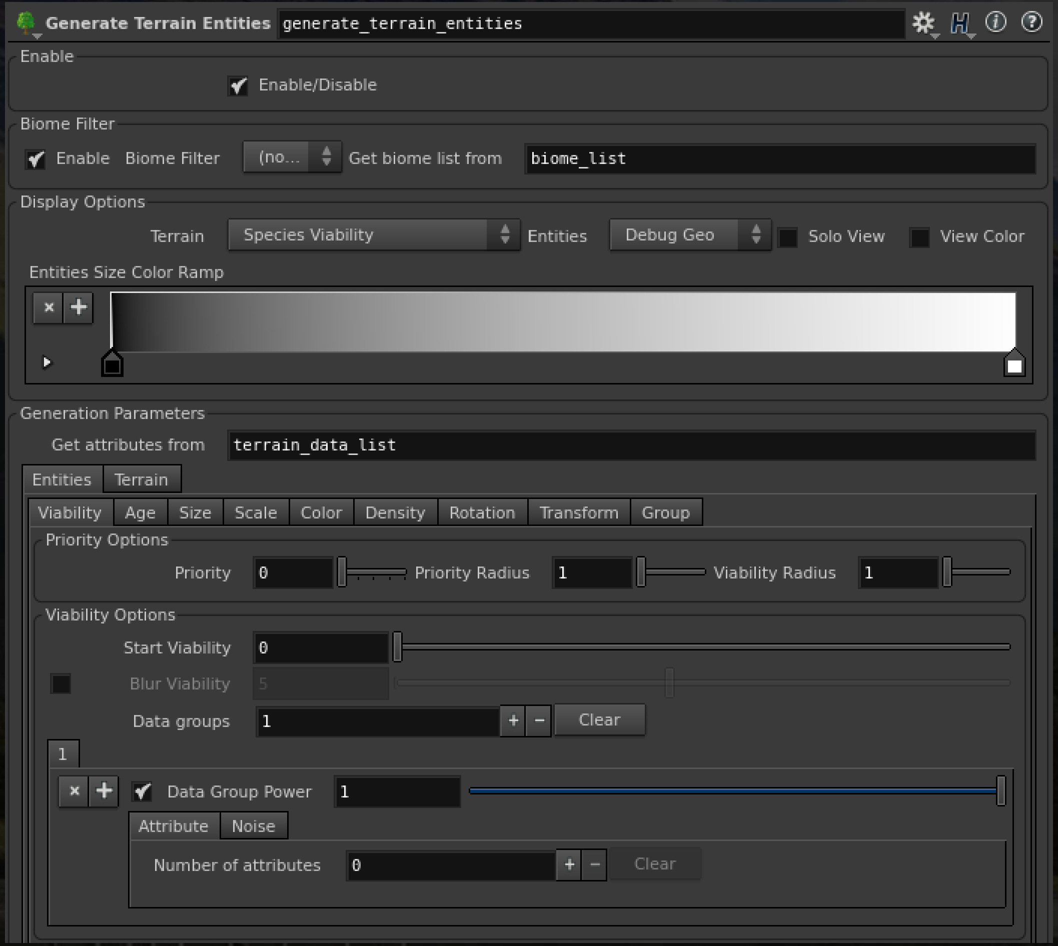


Generate terrain entities

Houdini digital asset

- **Scatter entities on the terrain (point cloud)**
- **Modify & create terrain attributes**
- **Defines a viability for each species**





Generate Terrain Entities generate_terrain_entities

Enable

✓ Enable/Disable

Biome Filter

✓ Enable Biome Filter (no... Get biome list from biome_list

Display Options

Terrain Species Viability Entities Debug Geo Solo View View Color

Entities Size Color Ramp

Generation Parameters

Get attributes from terrain_data_list

Entities | Terrain

Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

Priority Options

Priority 0 Priority Radius 1 Viability Radius 1

Viability Options

Start Viability 0

Blur Viability 5

Data groups 1 + − Clear

1

✓ Data Group Power 1

Attribute | Noise

Number of attributes 0 + − Clear

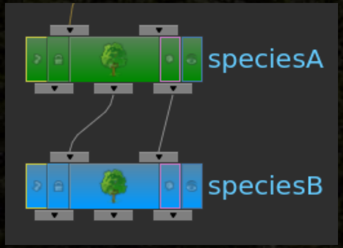
Used to define the spawning behavior of :

- **Trees**
- **Bushes**
- **Grass**
- **Rocks**
- **Etc**

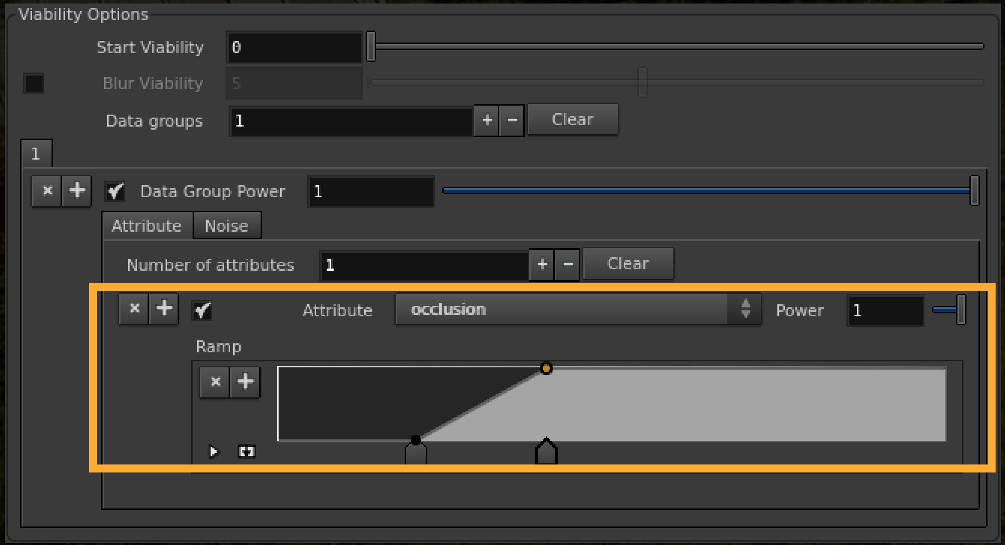
Viability

Each species is fighting for his ground to grow and thrive

- **Viability is defined by setting up favored terrain attributes for each species.**
- **Species that accumulate the most viability will win over species.**



Species A viability parameters



Viability Options

Start Viability 0

Blur Viability 5

Data groups 1 + − Clear

1

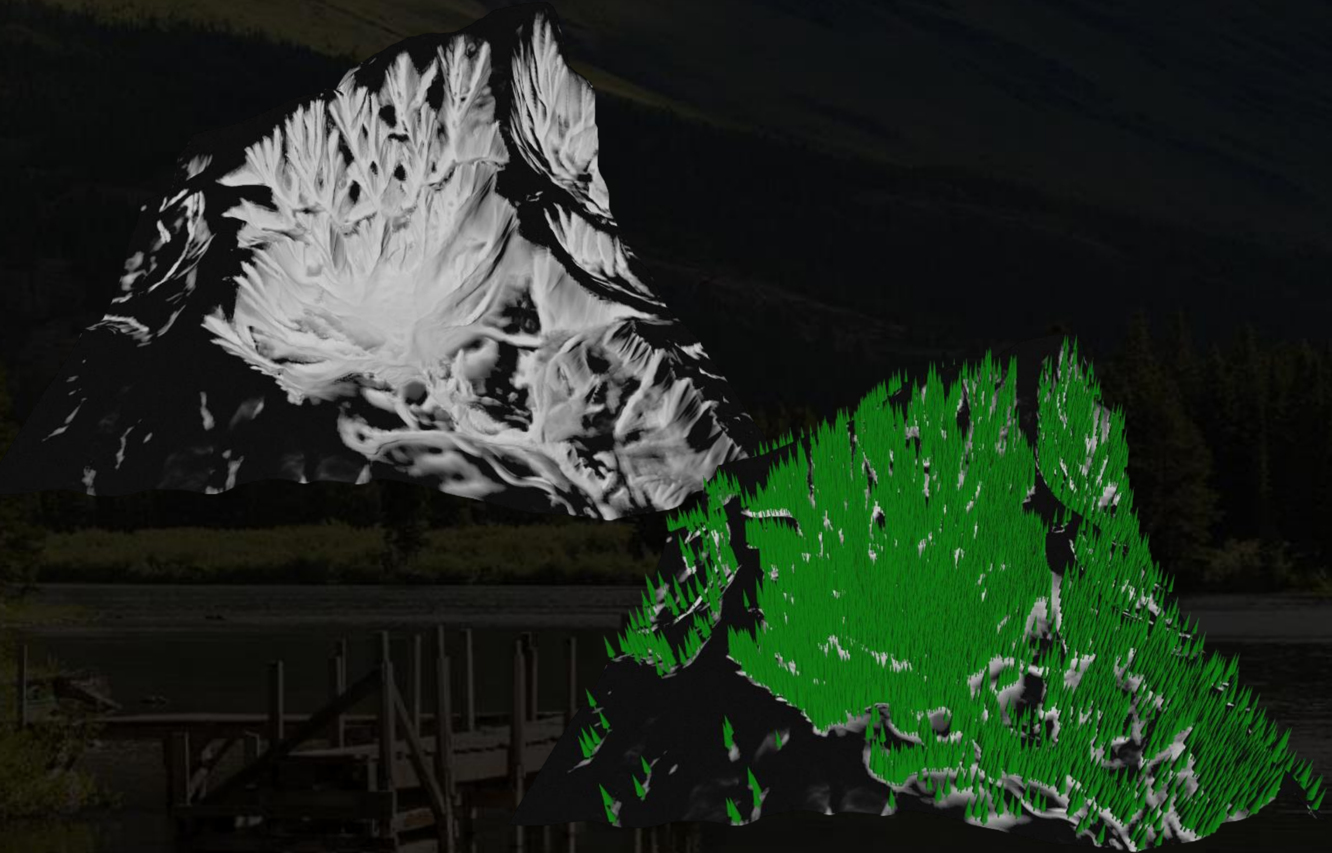
Data Group Power 1

Attribute | Noise

Number of attributes 1 + − Clear

Attribute occlusion Power 1

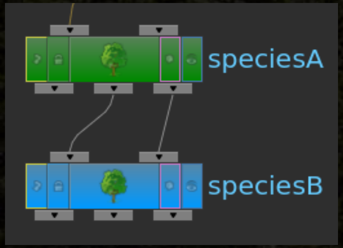
Ramp



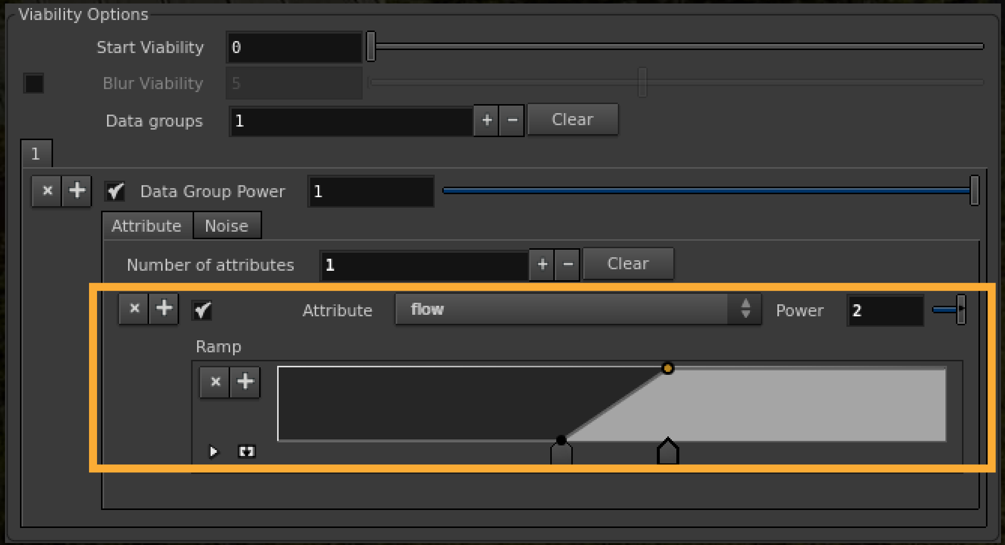
Viability

Each species is fighting for his ground to grow and thrive

- **Viability is defined by setting up favored terrain attributes for each species.**
- **Species that accumulate the most viability will win over species.**



Species B viability parameters



Viability Options

Start Viability 0

Blur Viability 5

Data groups 1 Clear

1

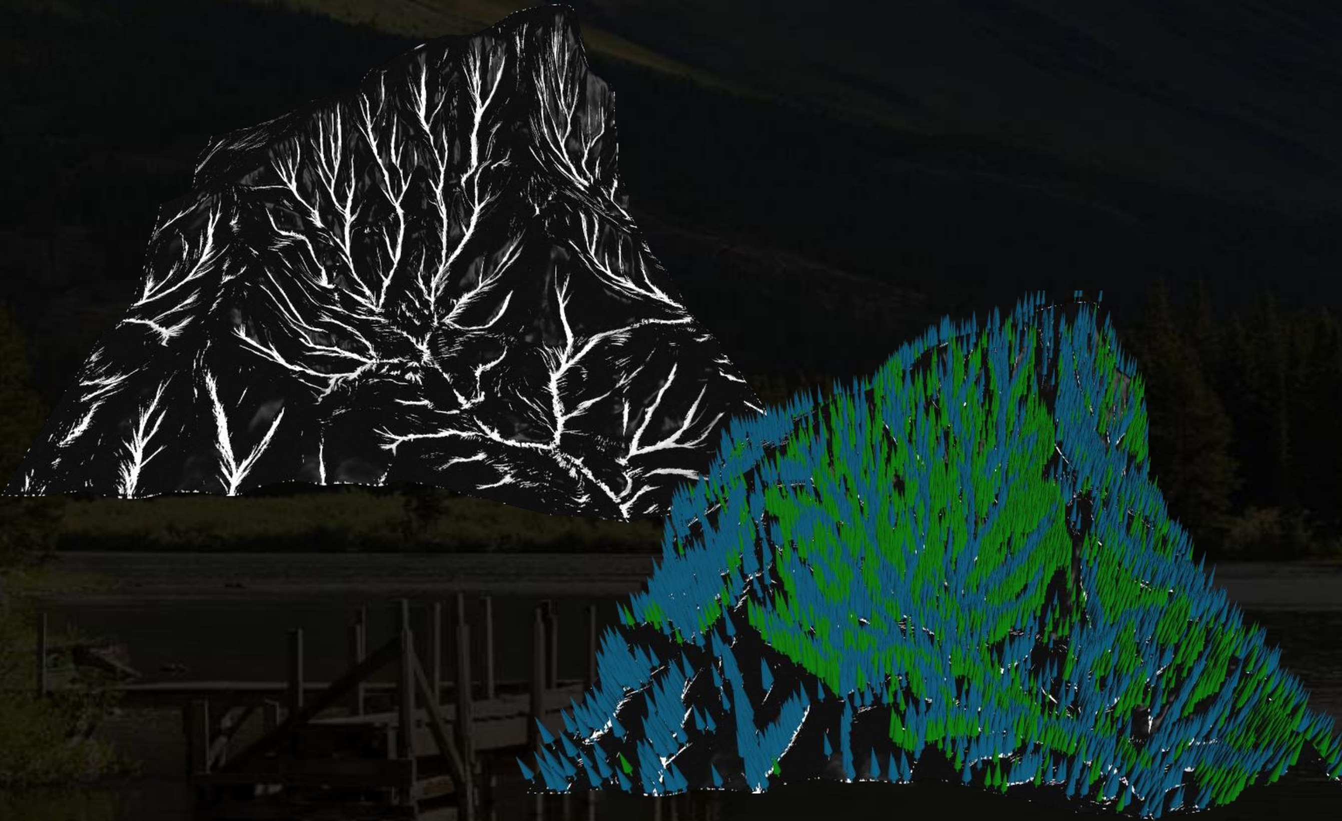
Data Group Power 1

Attribute Noise

Number of attributes 1 Clear

Attribute flow Power 2

Ramp



Viability Radius

Selecting winning species

- In this example one blue tree is discarded (in red) as it is within viability radius of the green tree with higher viability.

Species A

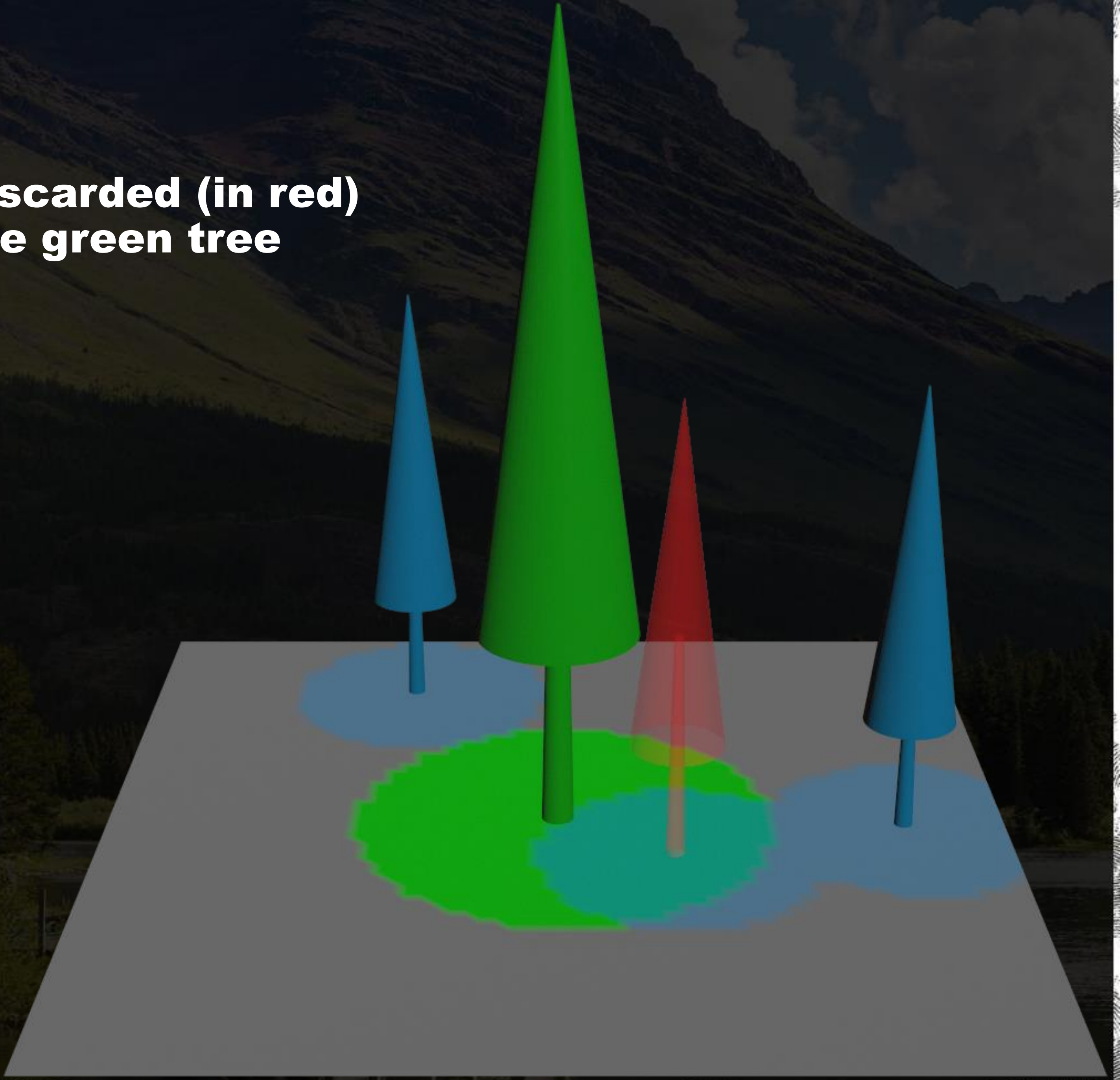
Viability = 2

Viability Radius = 15

Species B

Viability = 1

Viability Radius = 10



Priority Radius

Selecting winning species

- The filtering will process the priority first
- If priority is equal the viability will be use instead

Species A

Priority = 10

Priority Radius = 2

Species B

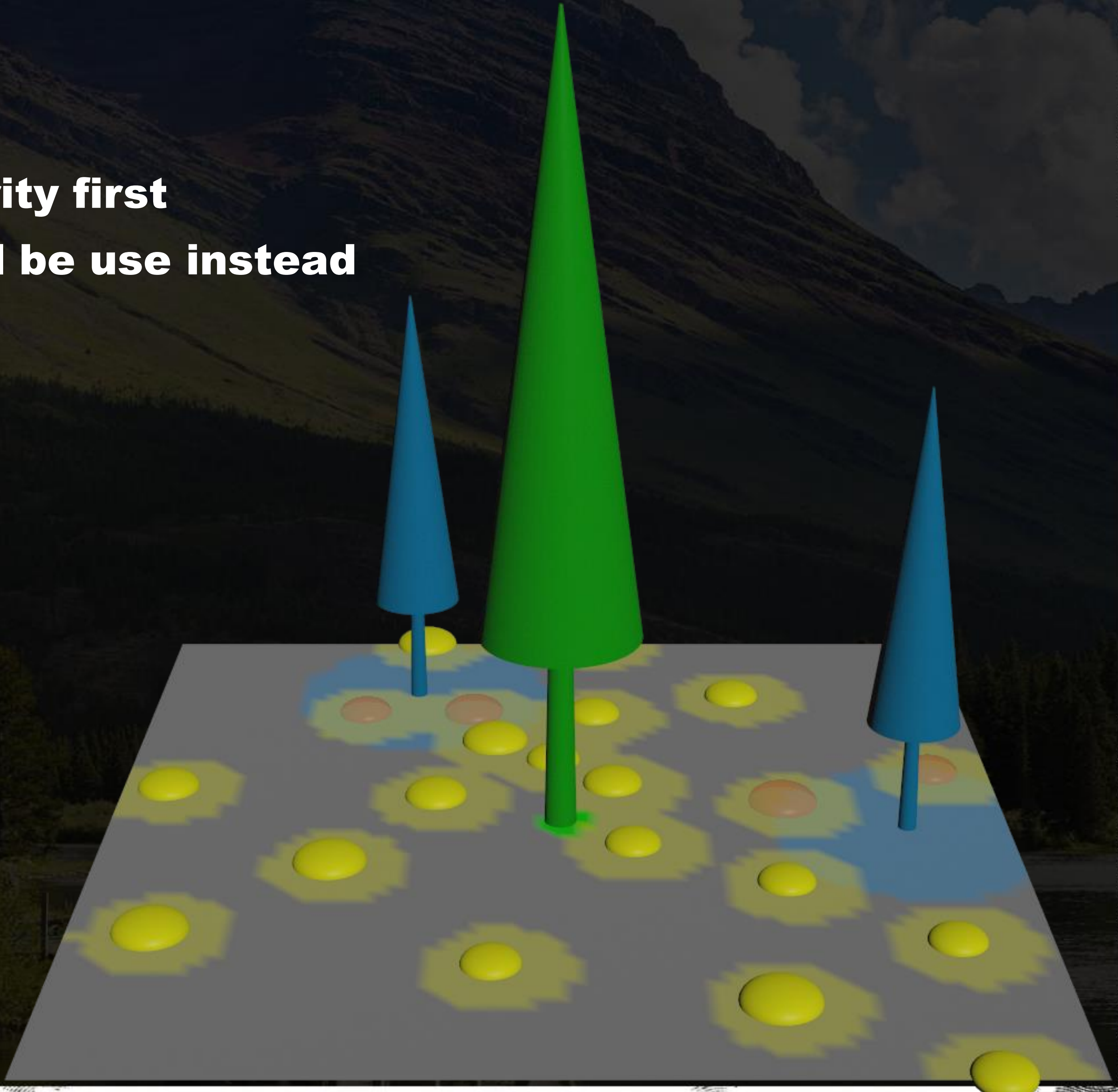
Priority = 10

Priority Radius = 10

Species C

Priority = 0

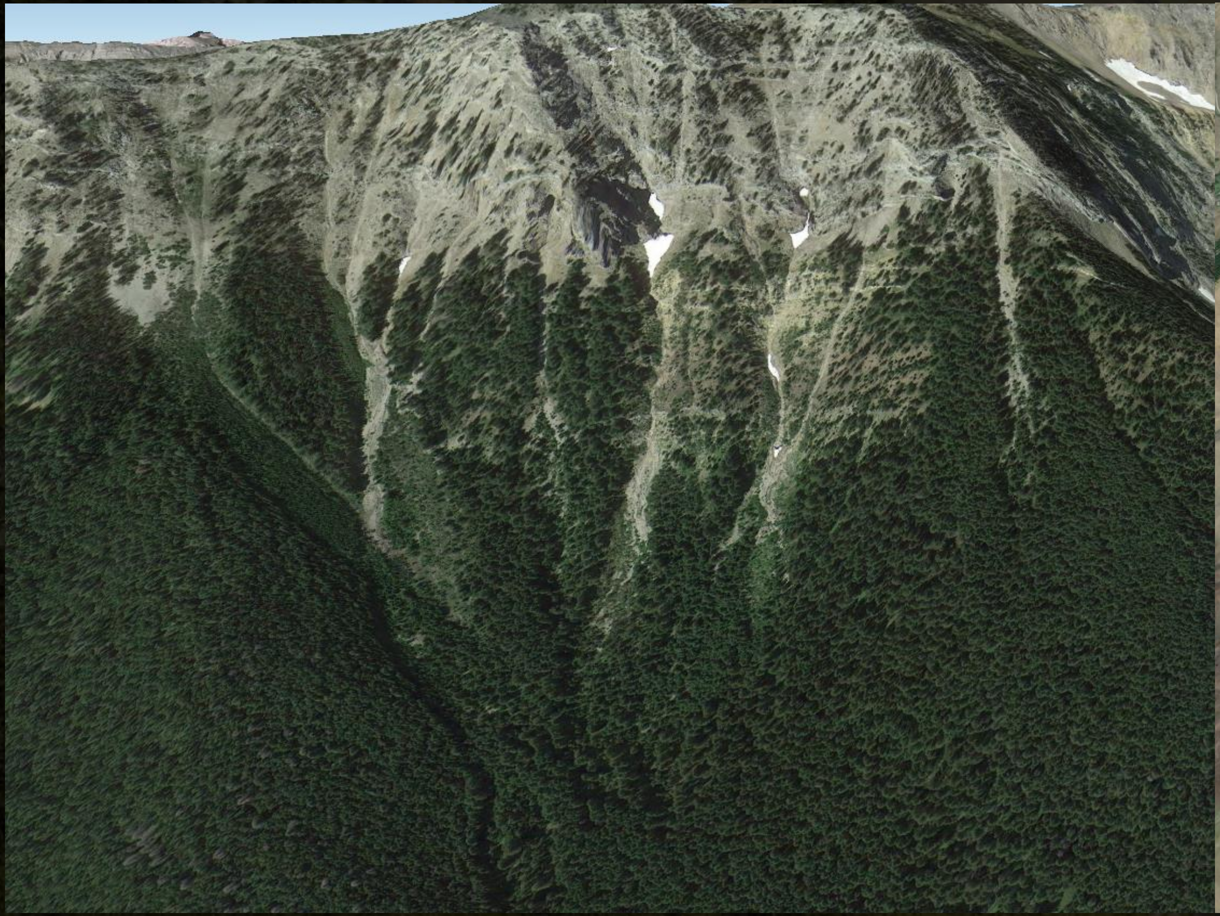
Priority Radius = 0.5

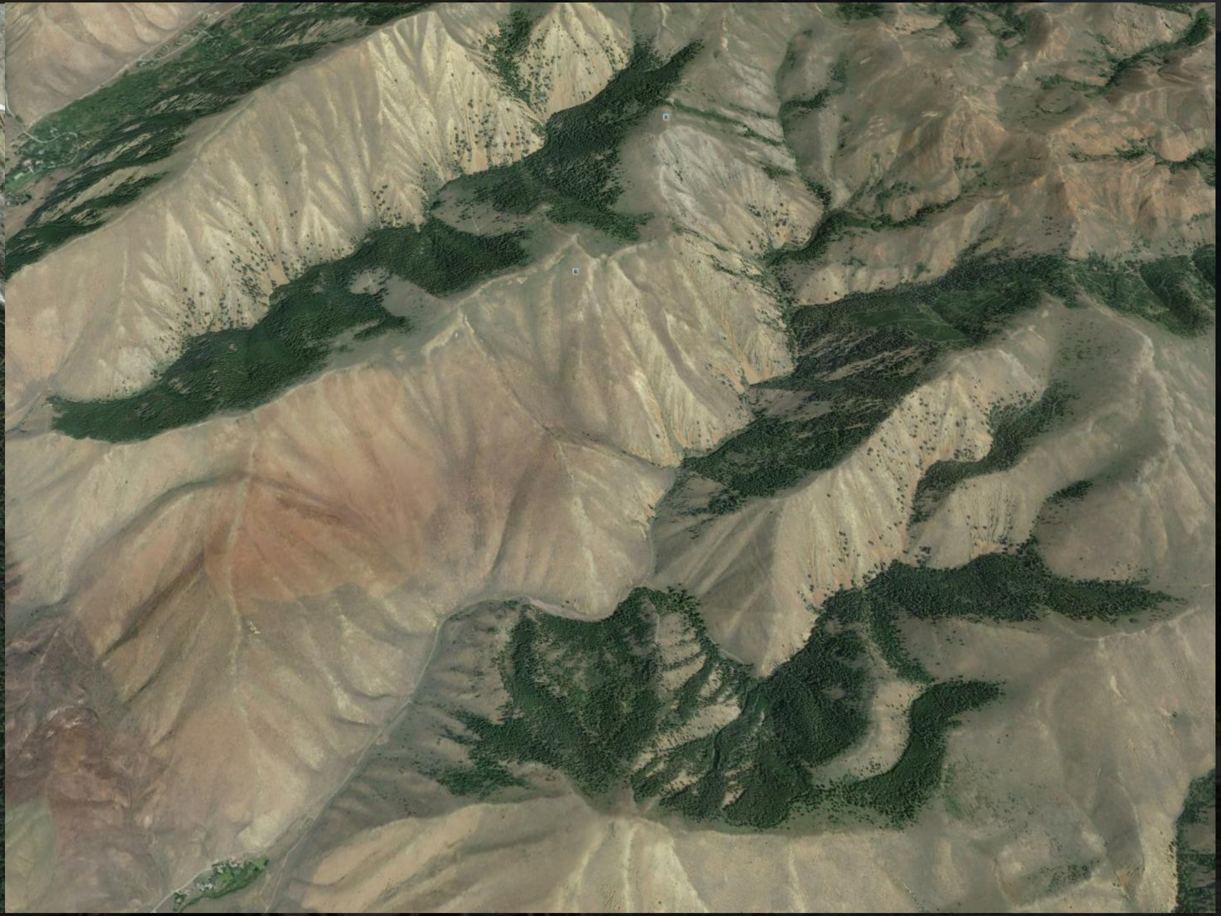


Natural phenomena

Vegetation patterns

- No vegetation growing on flow lines in steep slopes.
- Almost no vegetation growing on the south face of those mountains.

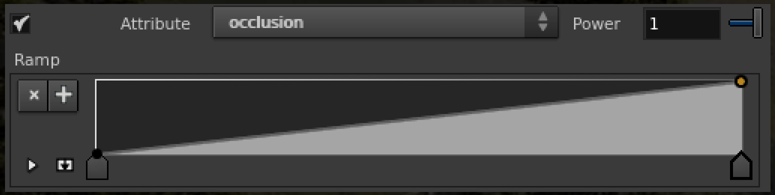




To mimic similar phenomena we need to be able to combine different abiotic terrain data together...

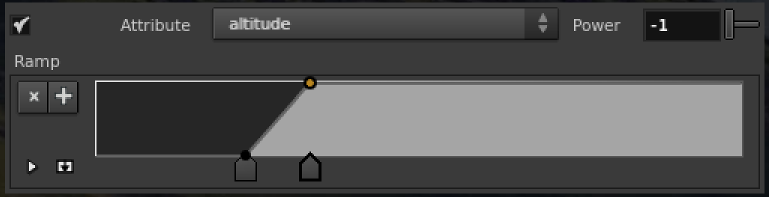
Combine terrain data

Mixing terrain abiotic data

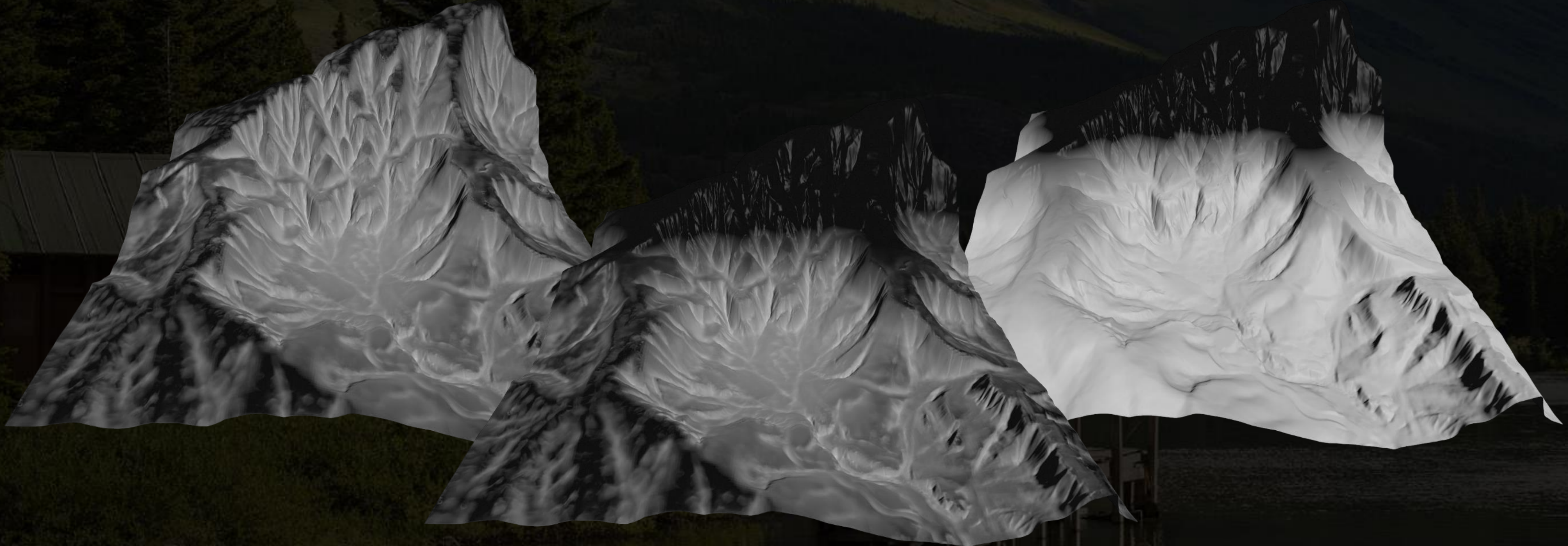

Attribute occlusion Power 1

Ramp

X

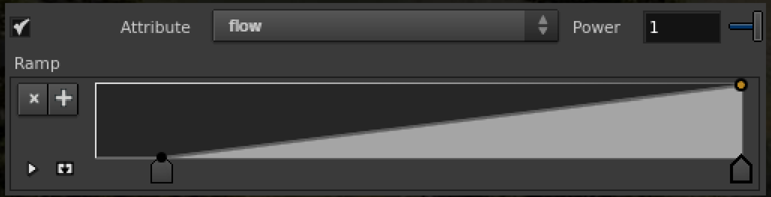

Attribute altitude Power -1

Ramp



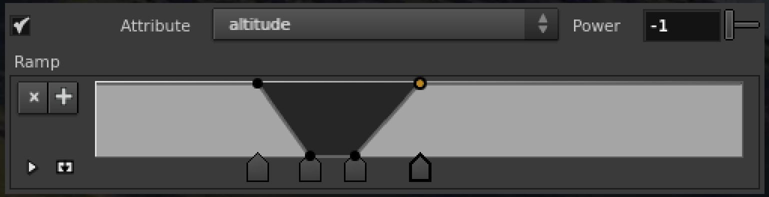
Combine terrain data

Mixing terrain abiotic data

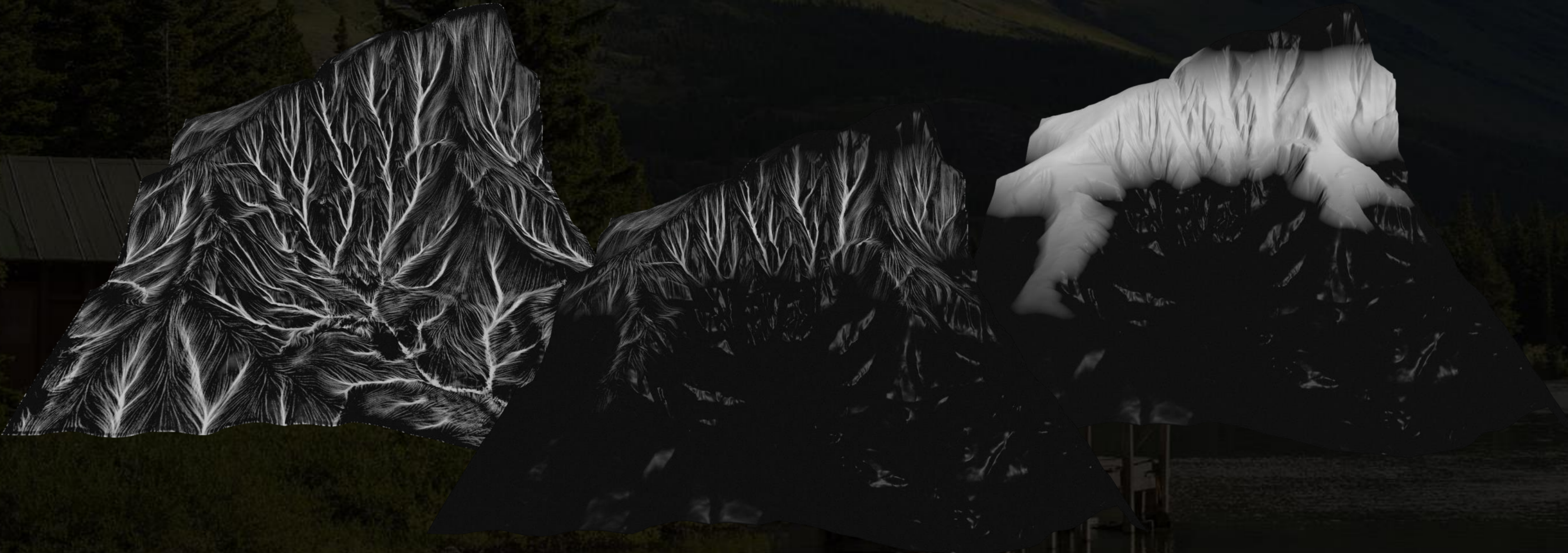


Attribute: flow
Power: 1
Ramp

x



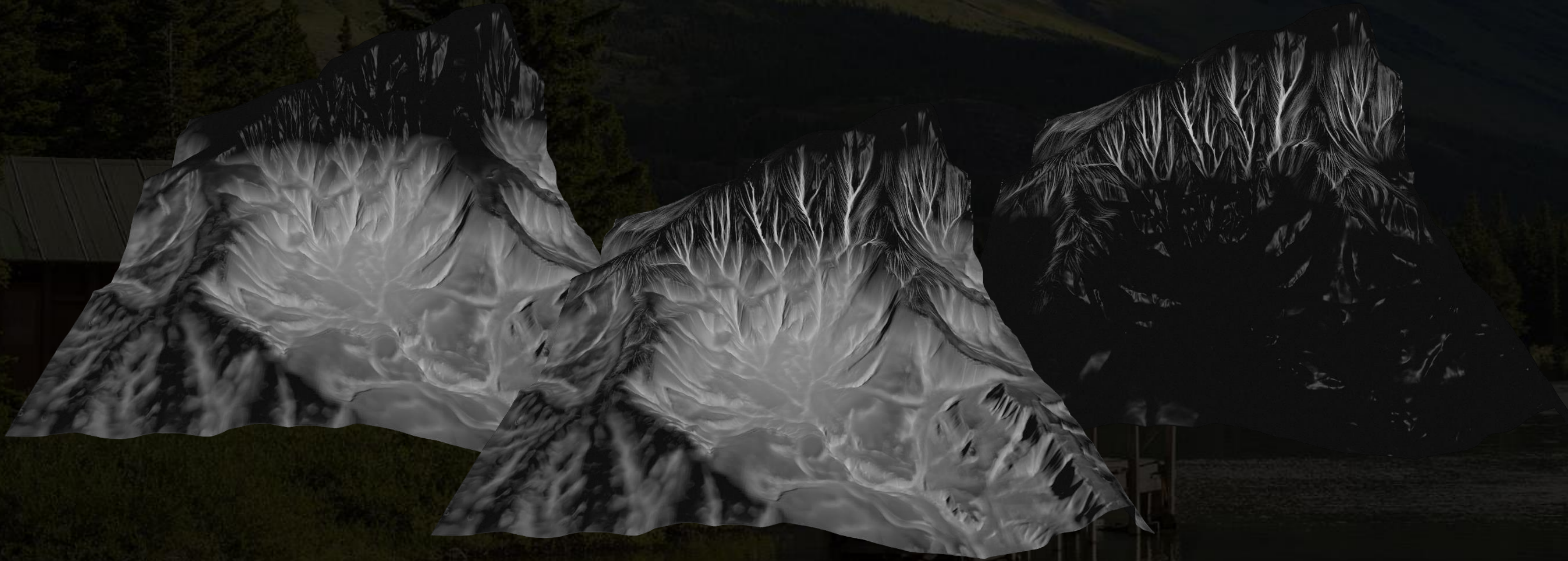
Attribute: altitude
Power: -1
Ramp



Combine terrain data

Mixing terrain abiotic data

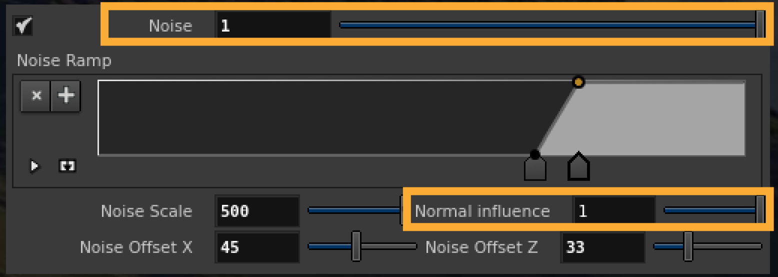
- Adding the result of the 2 previous sets together.

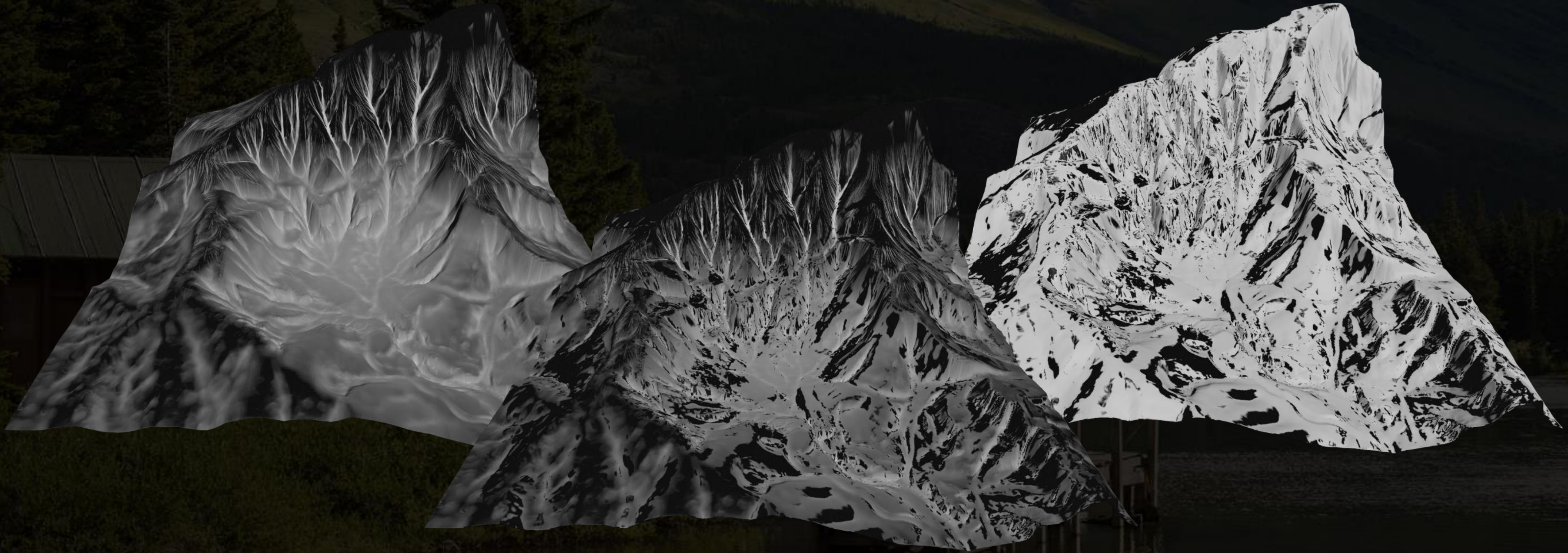


Combine terrain data

Noises

- **To create extra chaos we can combine noises with the terrain data. This is a standard perlin noise with control options for scale and offset. And it can also be warped by the terrain normal to create more interesting patterns.**

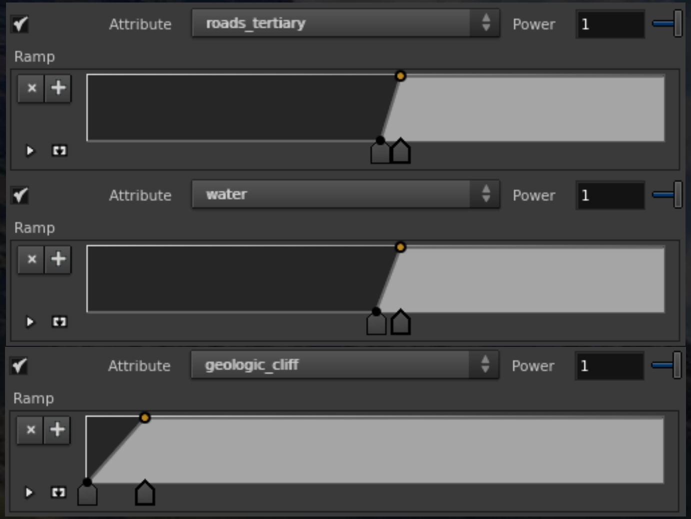

Noise 1
Noise Ramp
Noise Scale 500
Normal influence 1
Noise Offset X 45
Noise Offset Z 33



Combine terrain data

Exclusion masks

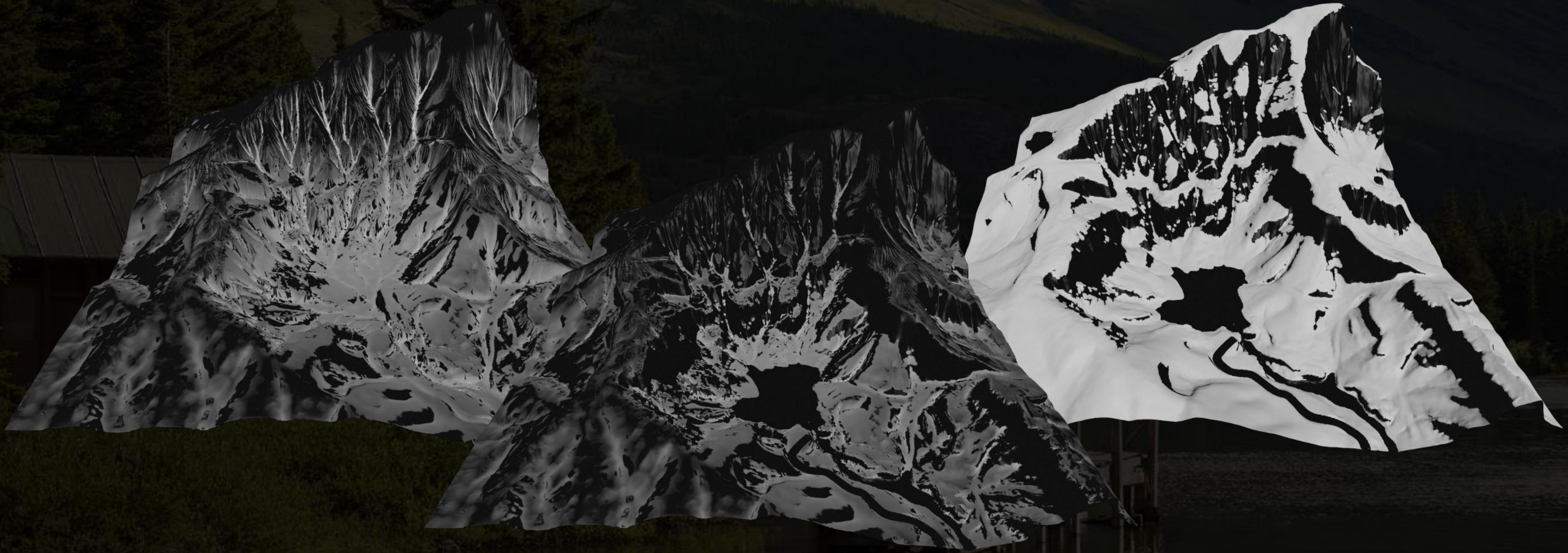
- Then we would also bring in various exclusion masks generated by previous tools (such as freshwater, roads or cliffs)



Attribute roads_tertiary Power 1
Ramp

Attribute water Power 1
Ramp

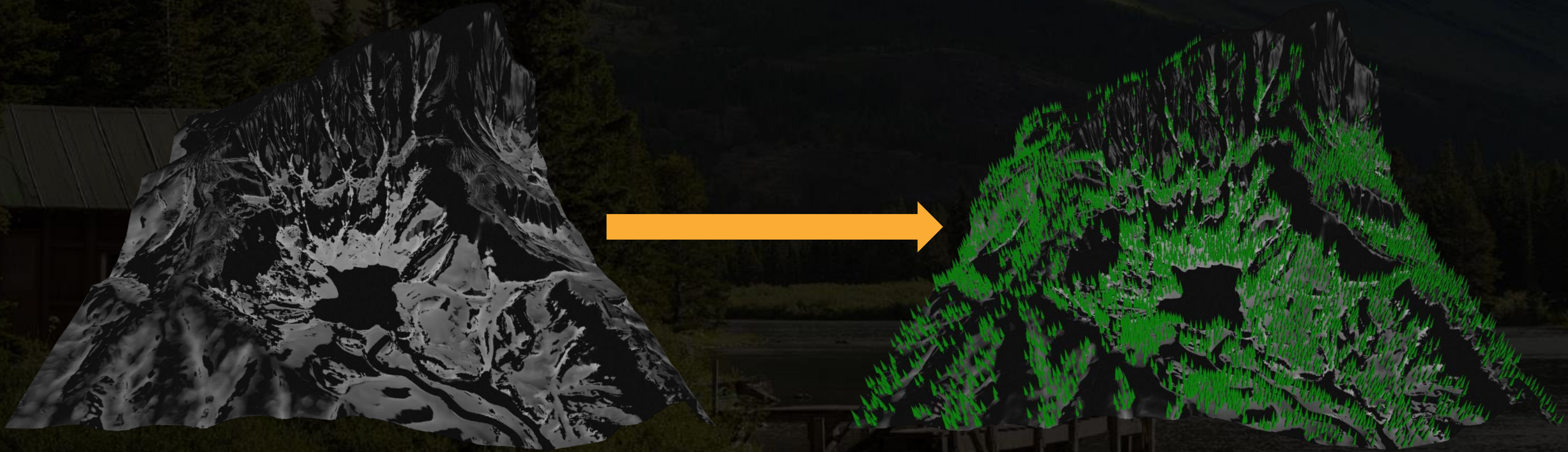
Attribute geologic_cliff Power 1
Ramp



Combine terrain data

Result used as viability for a species

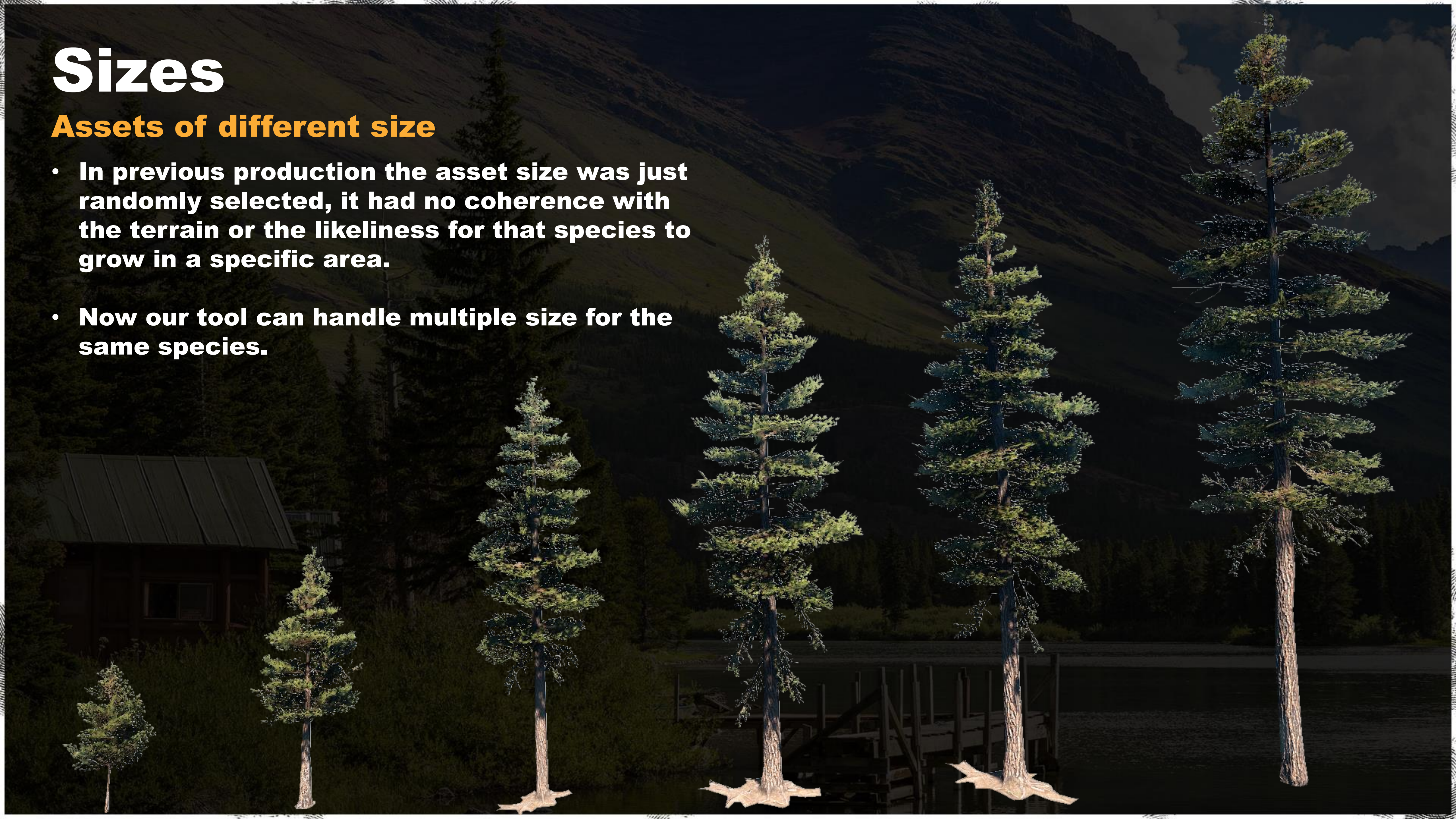
- And for this example, that's what we would use as viability for this species.
- As we see, combining terrain data is at the core of the biomes generation workflow. It is by mixing various terrain attributes we can create very specific patterns for species distribution and accumulate a fluctuating viability that will help blending various species together organically.



Sizes

Assets of different size

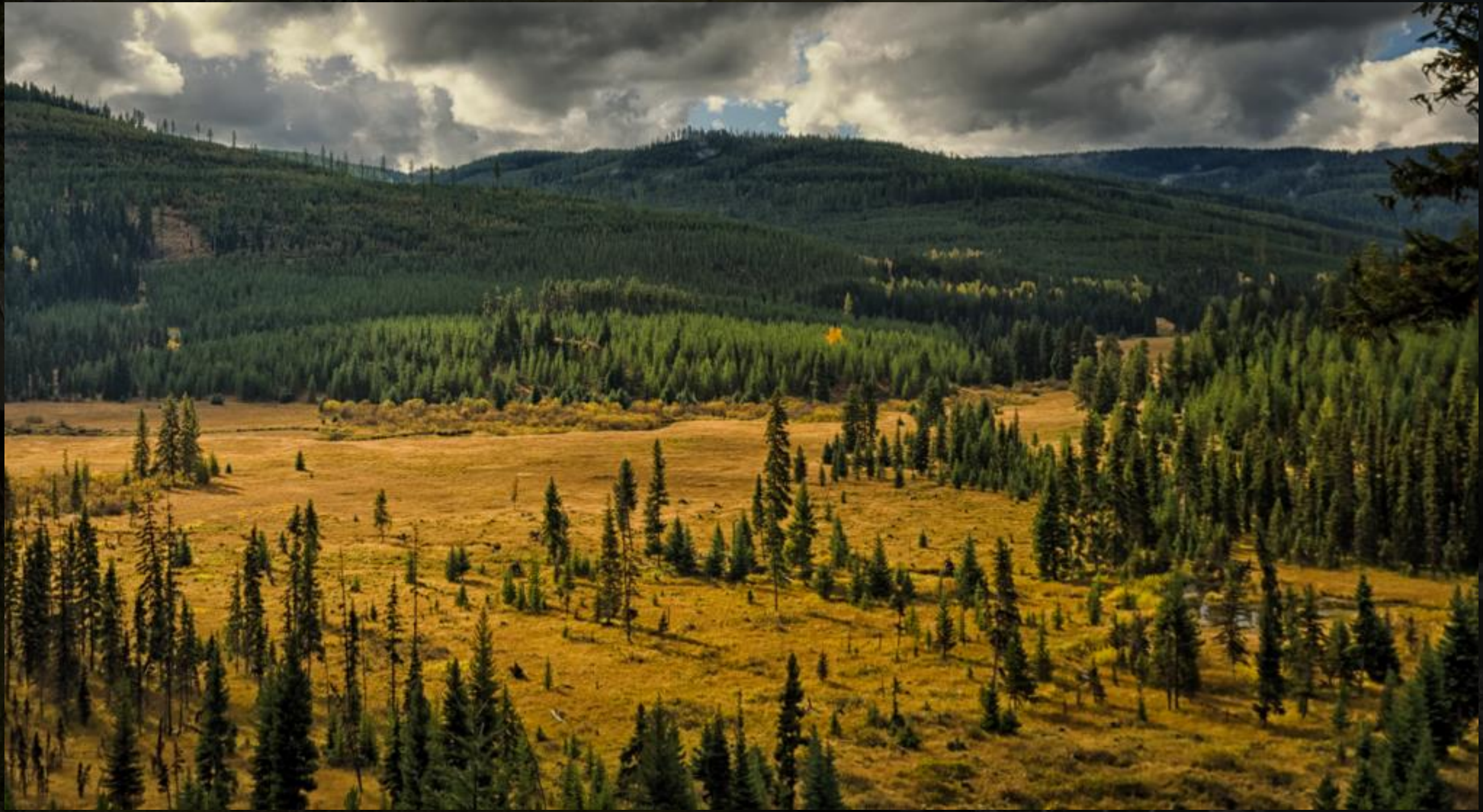
- In previous production the asset size was just randomly selected, it had no coherence with the terrain or the likeliness for that species to grow in a specific area.
- Now our tool can handle multiple size for the same species.



Size of trees

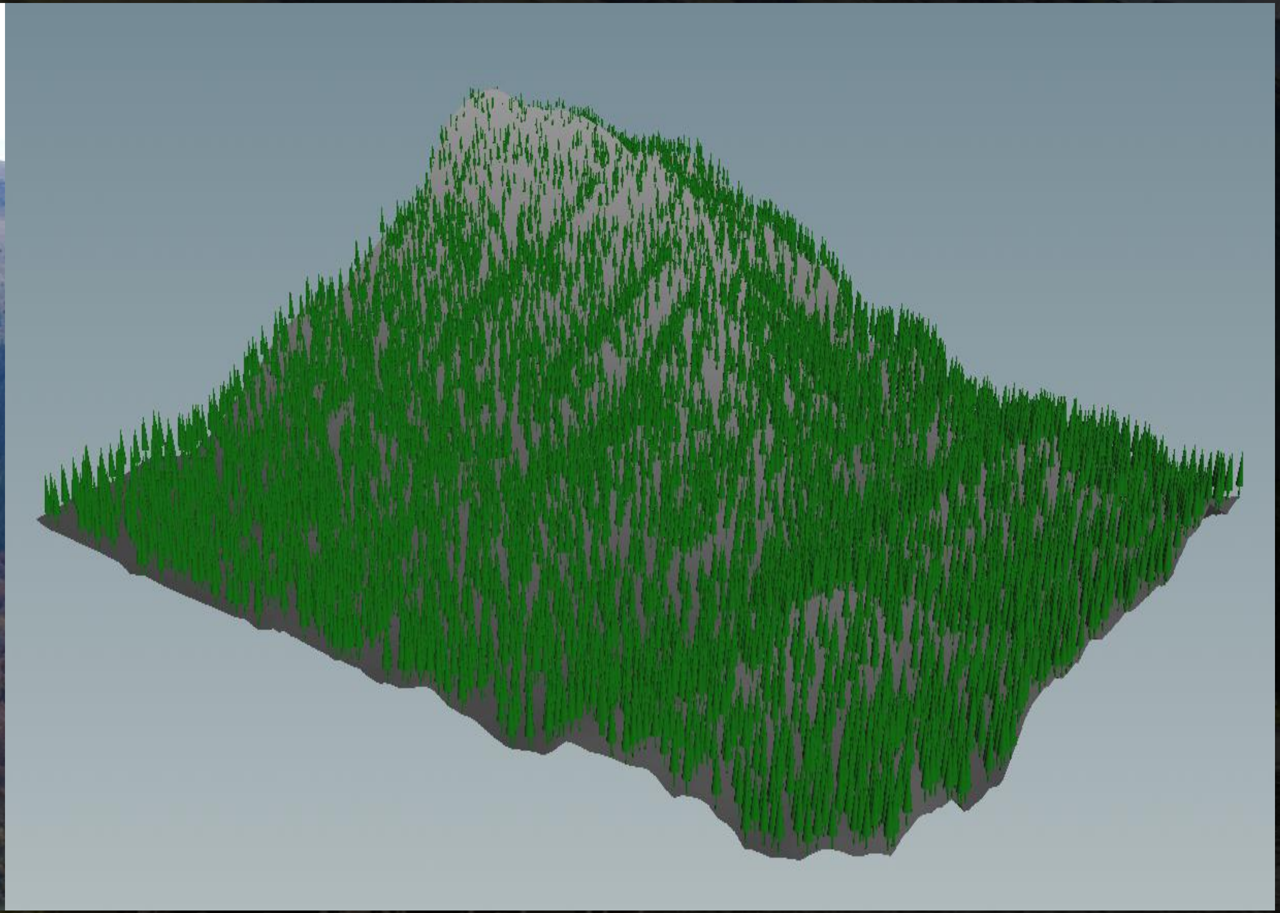
Observing nature

- Various things that can affect the tree size
- Small/young trees will tend to spawn at the edges of a forest
- Tall/old trees will be more present at the core of a large forest patch.



Sizes

Altitude influence on size

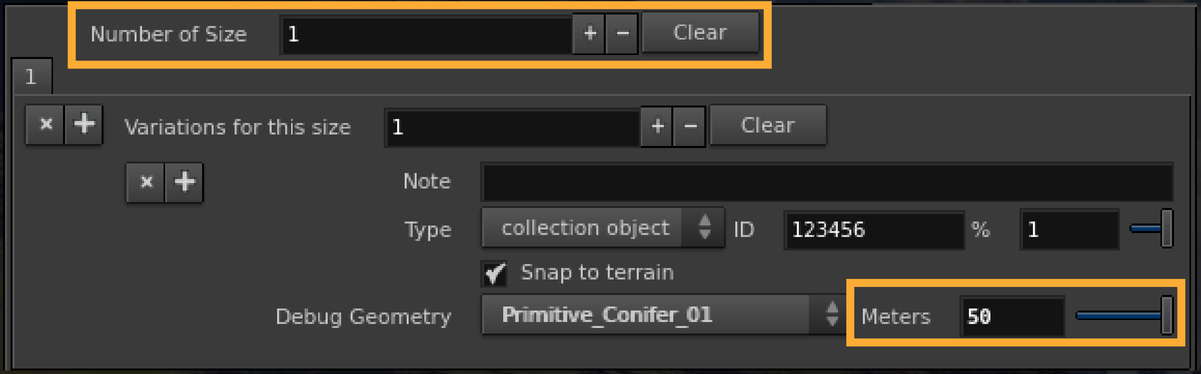


Sizes

Assets of different size

How do we manage our tree size?

We link the asset size selection to our viability!



Number of Size 1 + − Clear

1

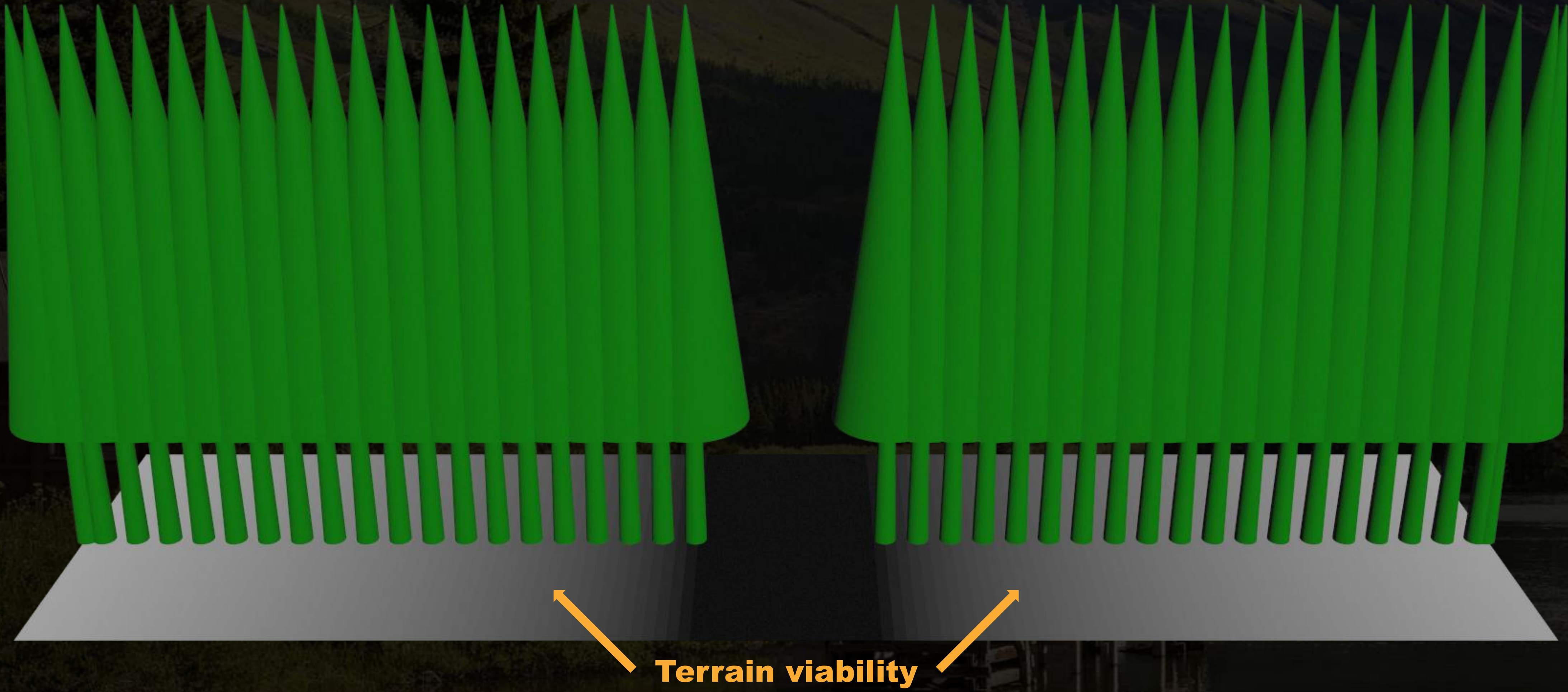
× + Variations for this size 1 + − Clear

× + Note

Type collection object ID 123456 % 1

✓ Snap to terrain

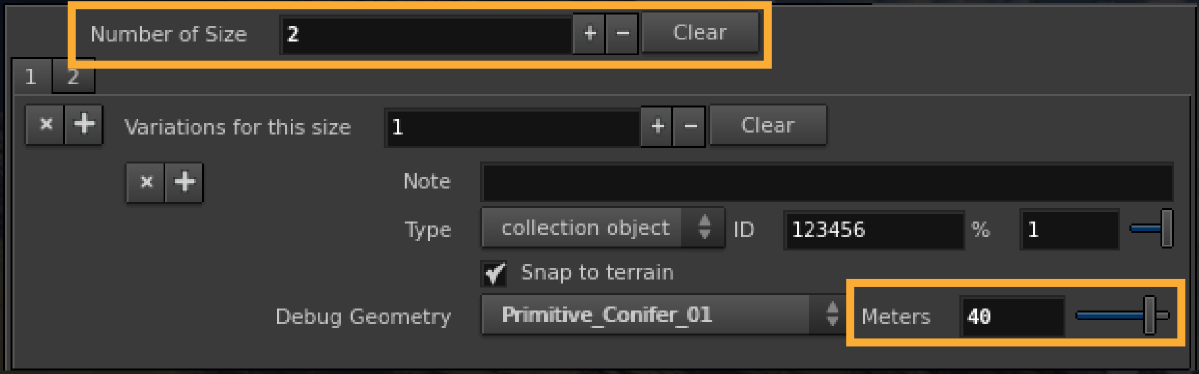
Debug Geometry Primitive_Conifer_01 Meters 50



Sizes

Assets of different size

- Adding a 2nd tree size of 40m.



Number of Size 2

Variations for this size 1

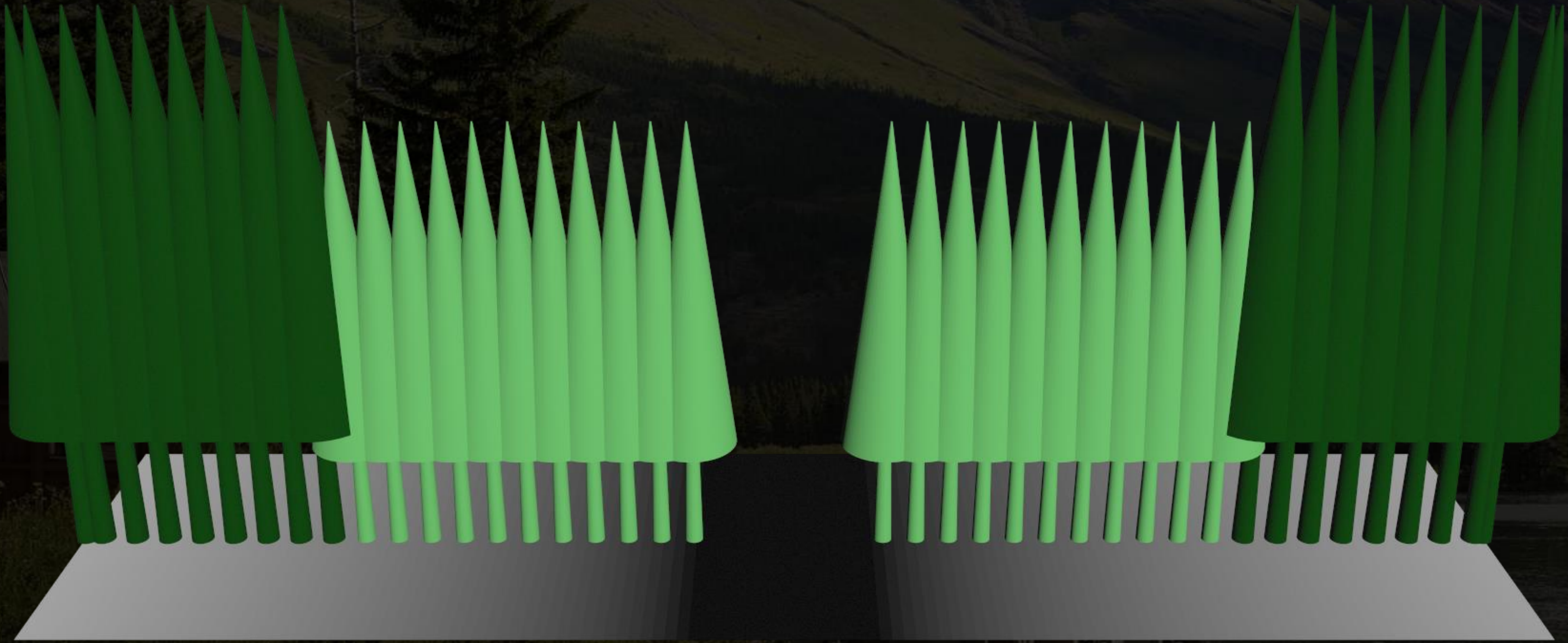
Note

Type collection object ID 123456 % 1

Snap to terrain ✓

Debug Geometry Primitive_Conifer_01

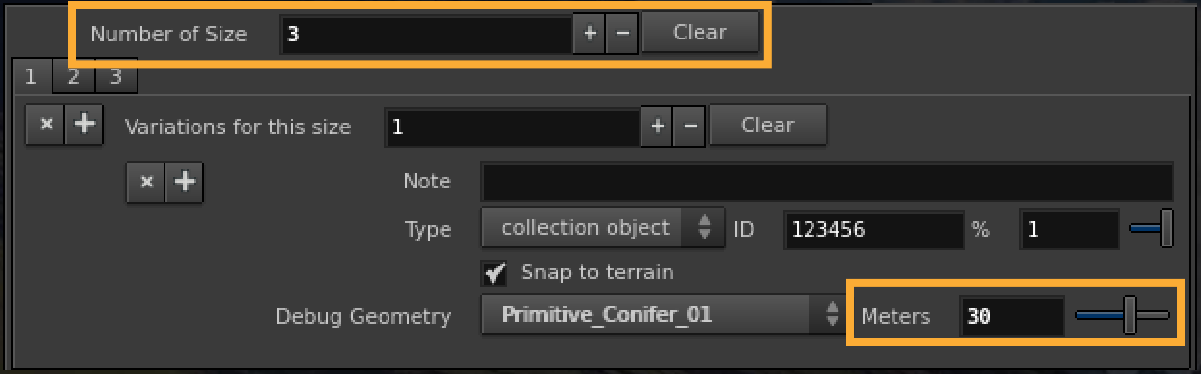
Meters 40



Sizes

Assets of different size

- Adding a 3rd tree size of 30m.



Number of Size: 3

1 2 3

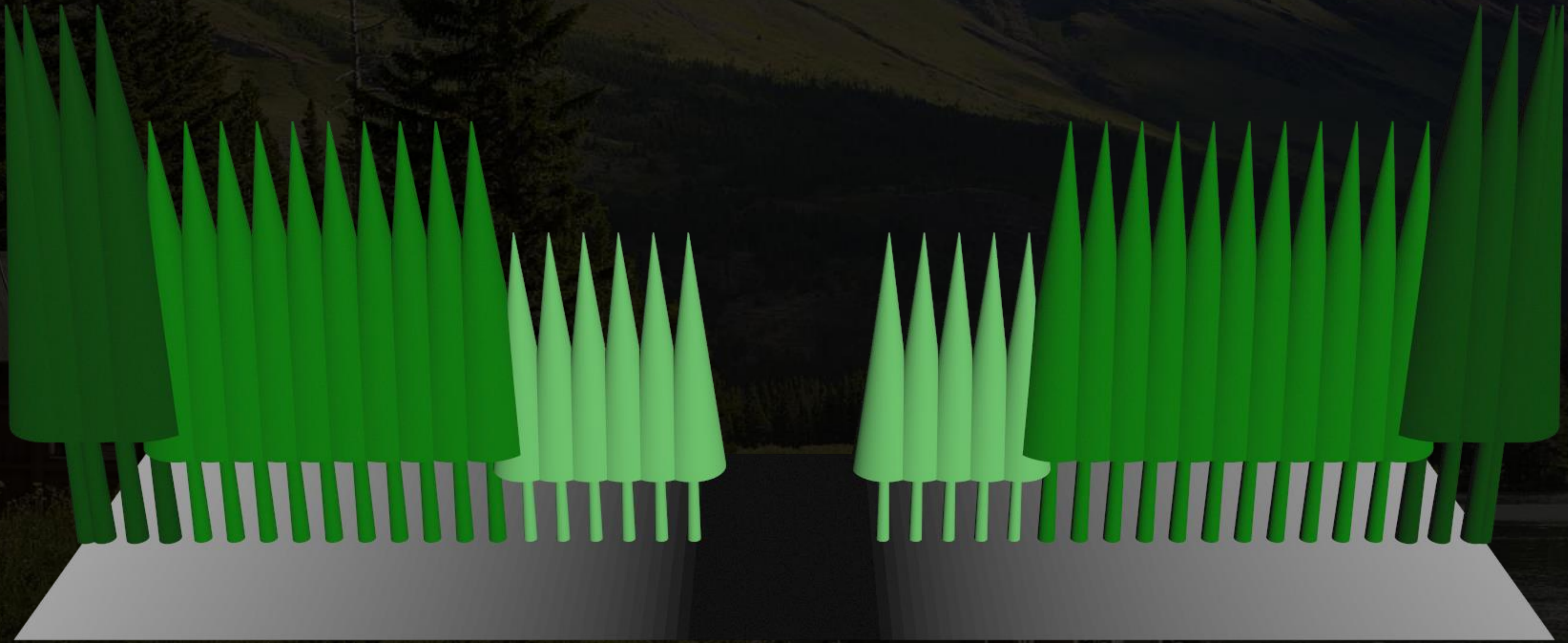
Variations for this size: 1

Note:

Type: collection object ID: 123456 %: 1

✓ Snap to terrain

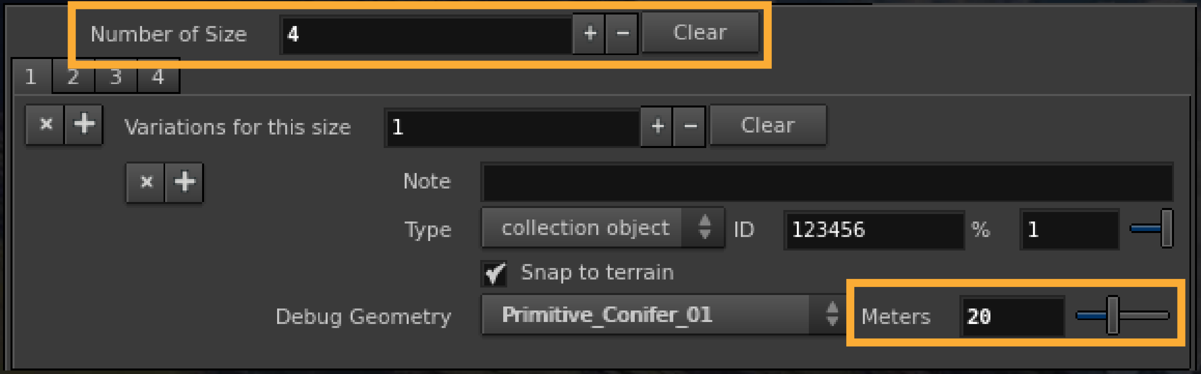
Debug Geometry: Primitive_Conifer_01 Meters: 30



Sizes

Assets of different size

- Adding a 4th tree size of 20m.



Number of Size 4

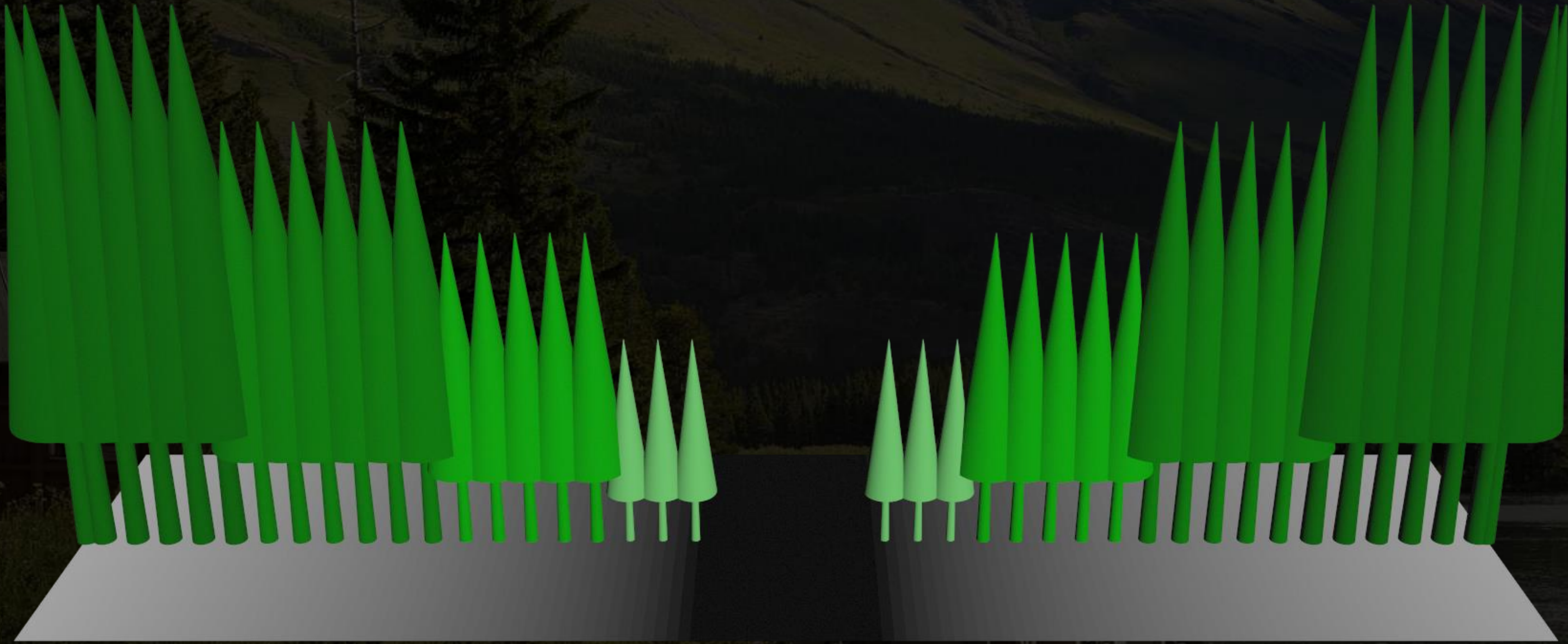
Variations for this size 1

Note

Type collection object ID 123456 % 1

✓ Snap to terrain

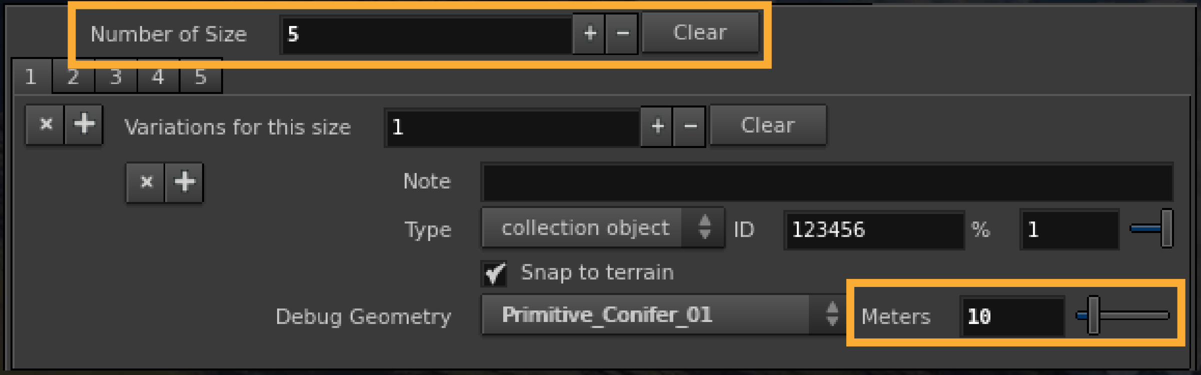
Debug Geometry Primitive_Conifer_01 Meters 20



Sizes

Assets of different size

- Adding a 5th tree size of 10m.
- Each size is positioned to the proper viability range on the terrain.



Number of Size 5 + − Clear

1 2 3 4 5

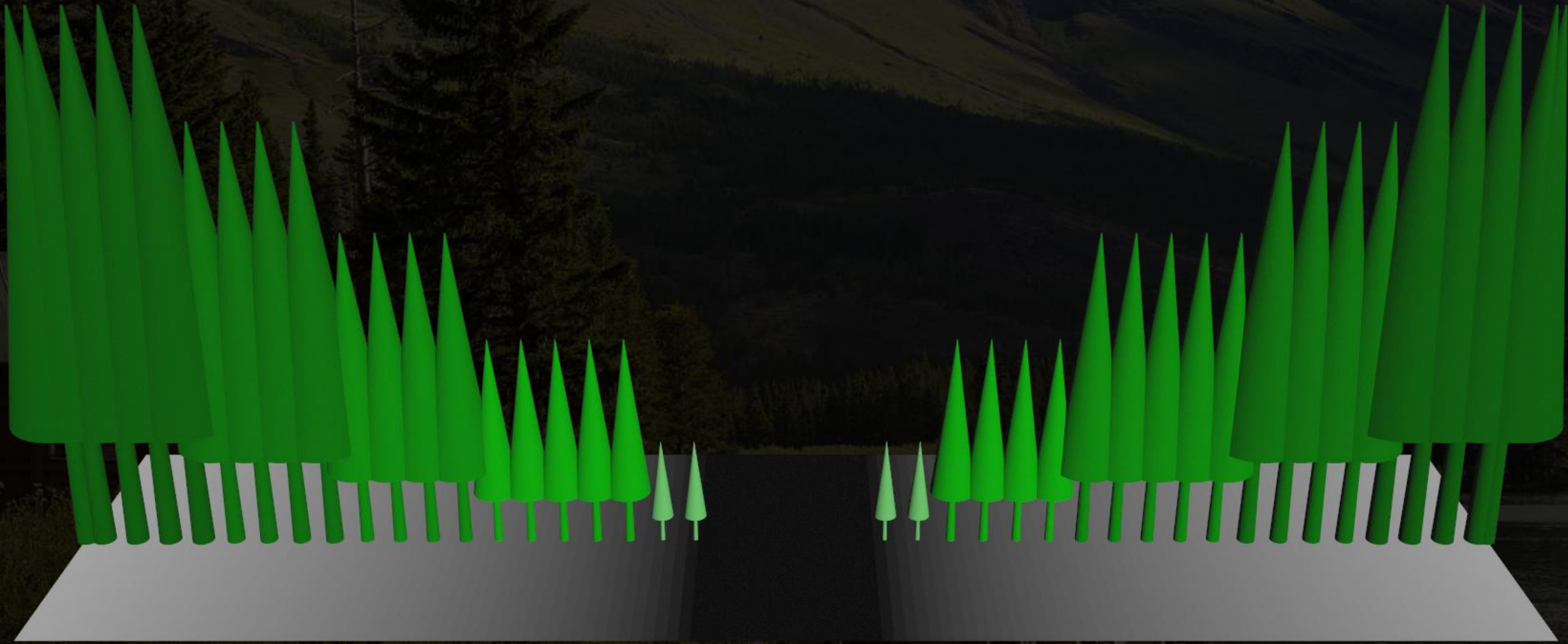
× + Variations for this size 1 + − Clear

× + Note

Type collection object ID 123456 % 1

✓ Snap to terrain

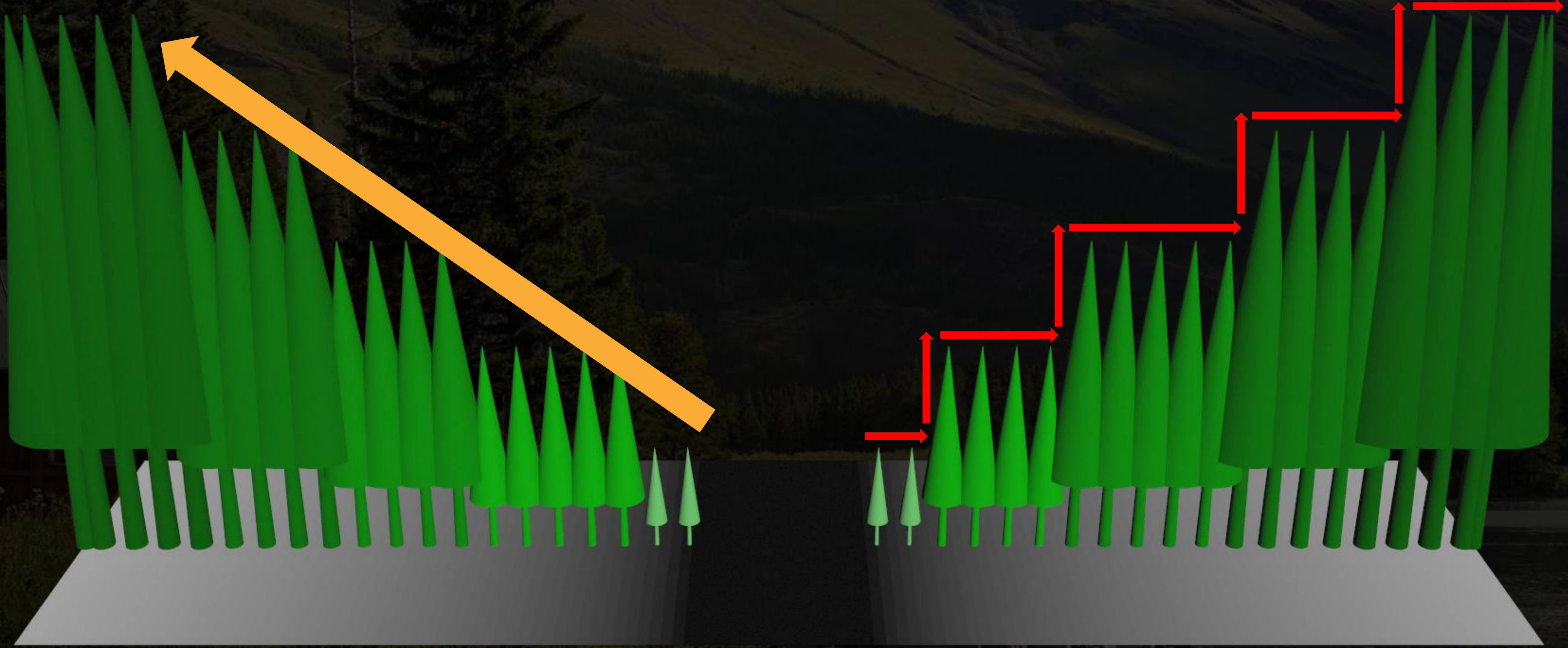
Debug Geometry Primitive_Conifer_01 Meters 10



Sizes

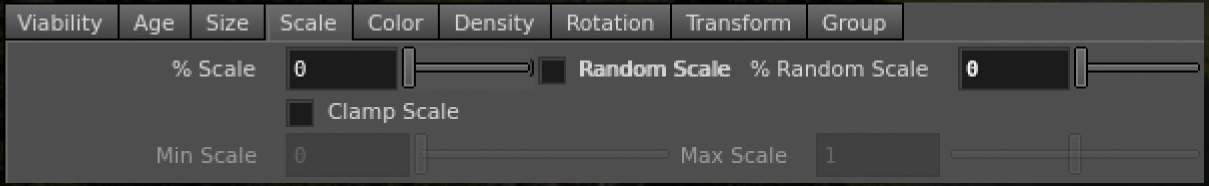
Assets of different size

- This will give us a nice tapering effect at the border of our forest.
- However we have one more problem, the staircase effect.



Scale

Scale percentage allowed to bridge gap between sizes

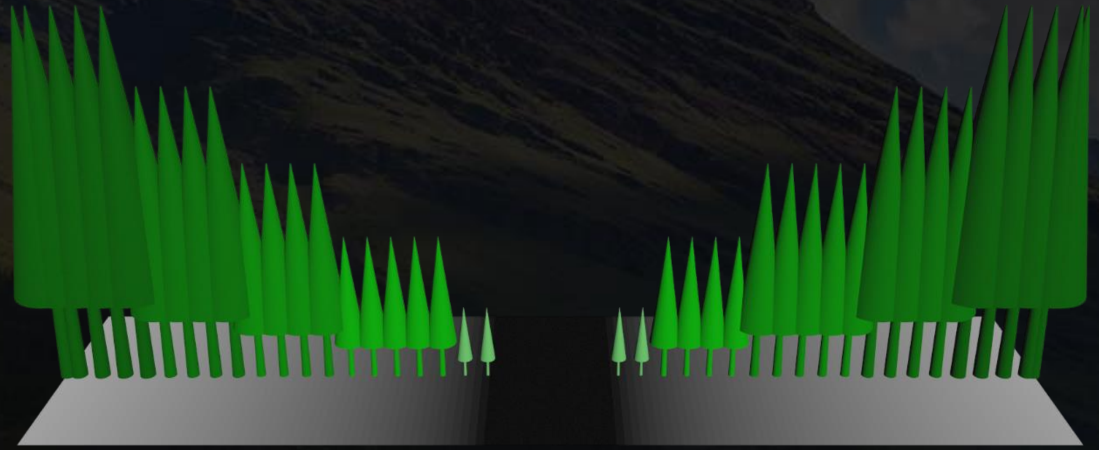


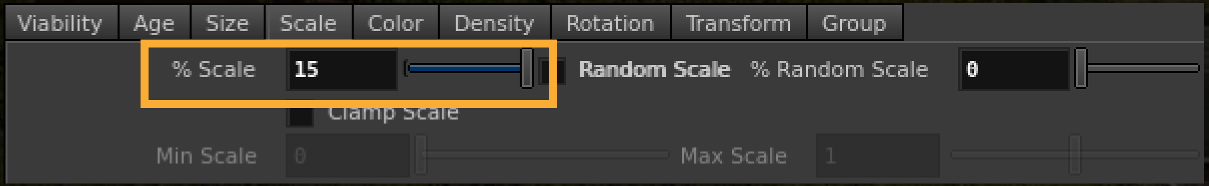
Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

% Scale 0 Random Scale % Random Scale 0

Clamp Scale

Min Scale 0 Max Scale 1



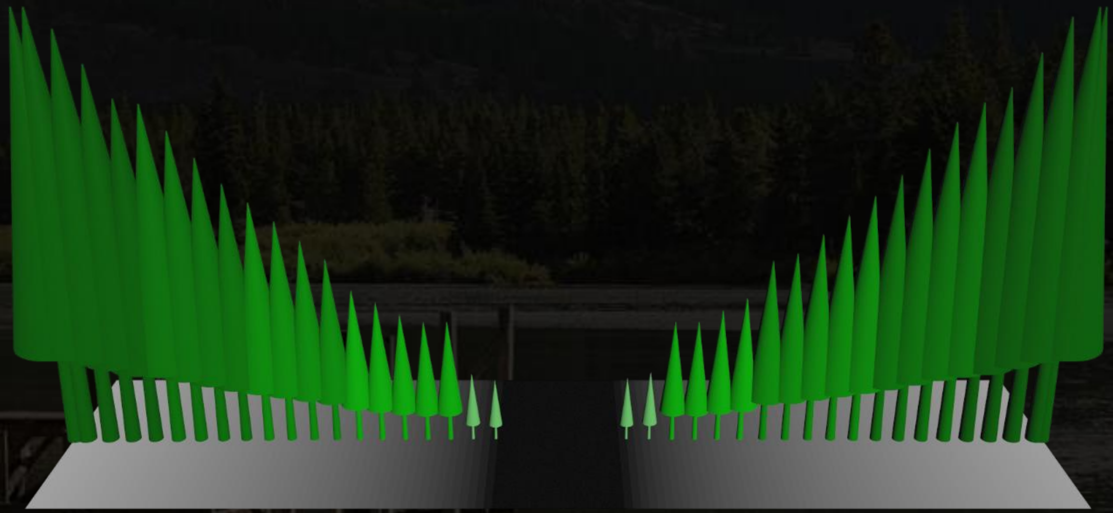


Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

% Scale 15 Random Scale % Random Scale 0

Clamp Scale

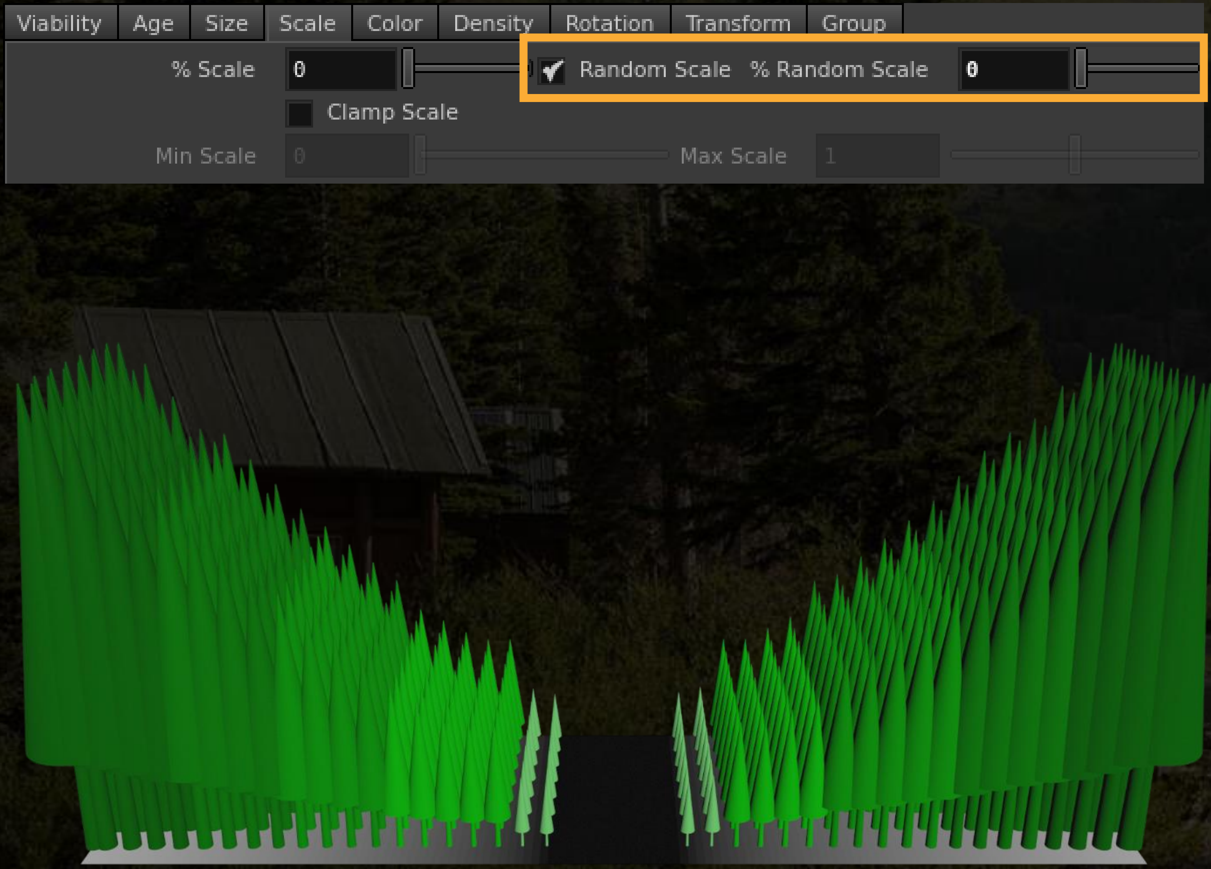
Min Scale 0 Max Scale 1



Scale

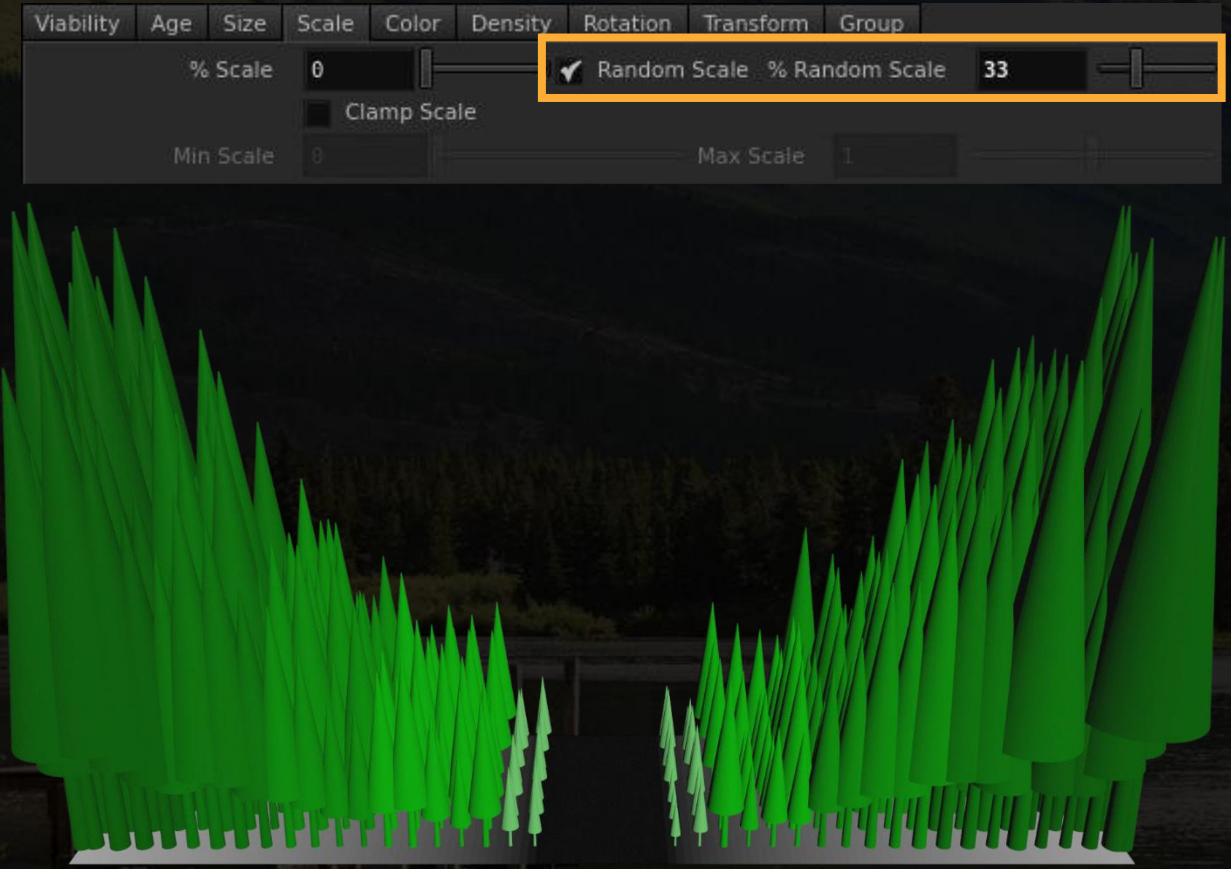
Random Scale percentage allowed

- **We can also play with a random scale if we don't want to mess the viability too much to create more chaos.**



Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

% Scale: 0
✓ Random Scale — % Random Scale: 0
Clamp Scale
Min Scale: 0 — Max Scale: 1



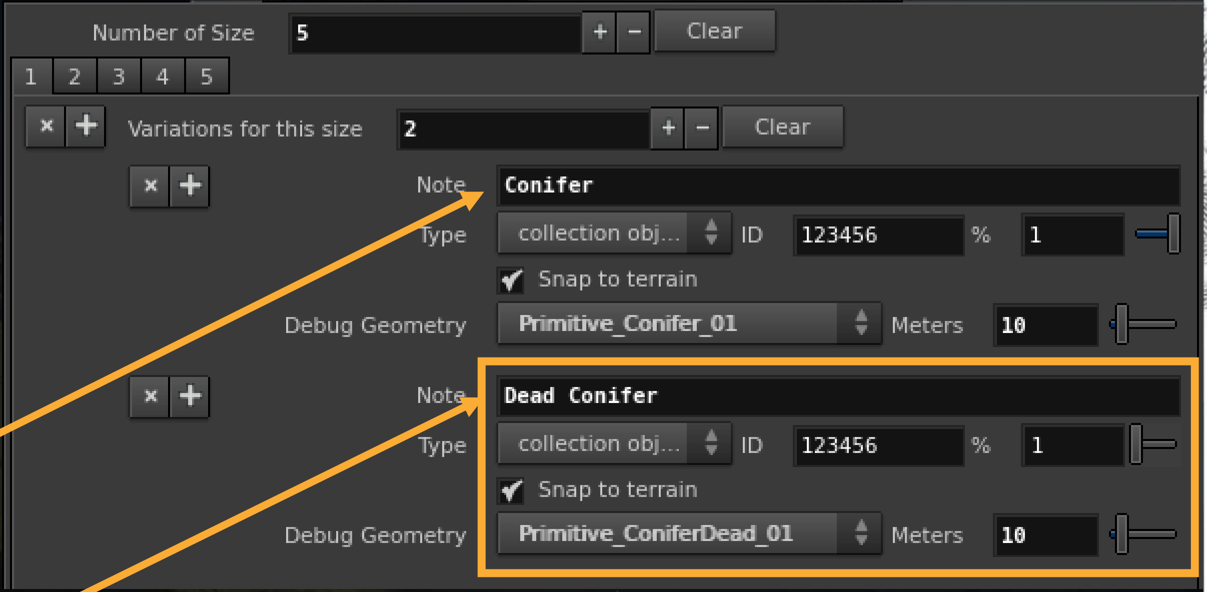
Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

% Scale: 0
✓ Random Scale — % Random Scale: 33
Clamp Scale
Min Scale: 0 — Max Scale: 1

Sizes Variation

Several asset of the same size

- **Probability control on each variation**



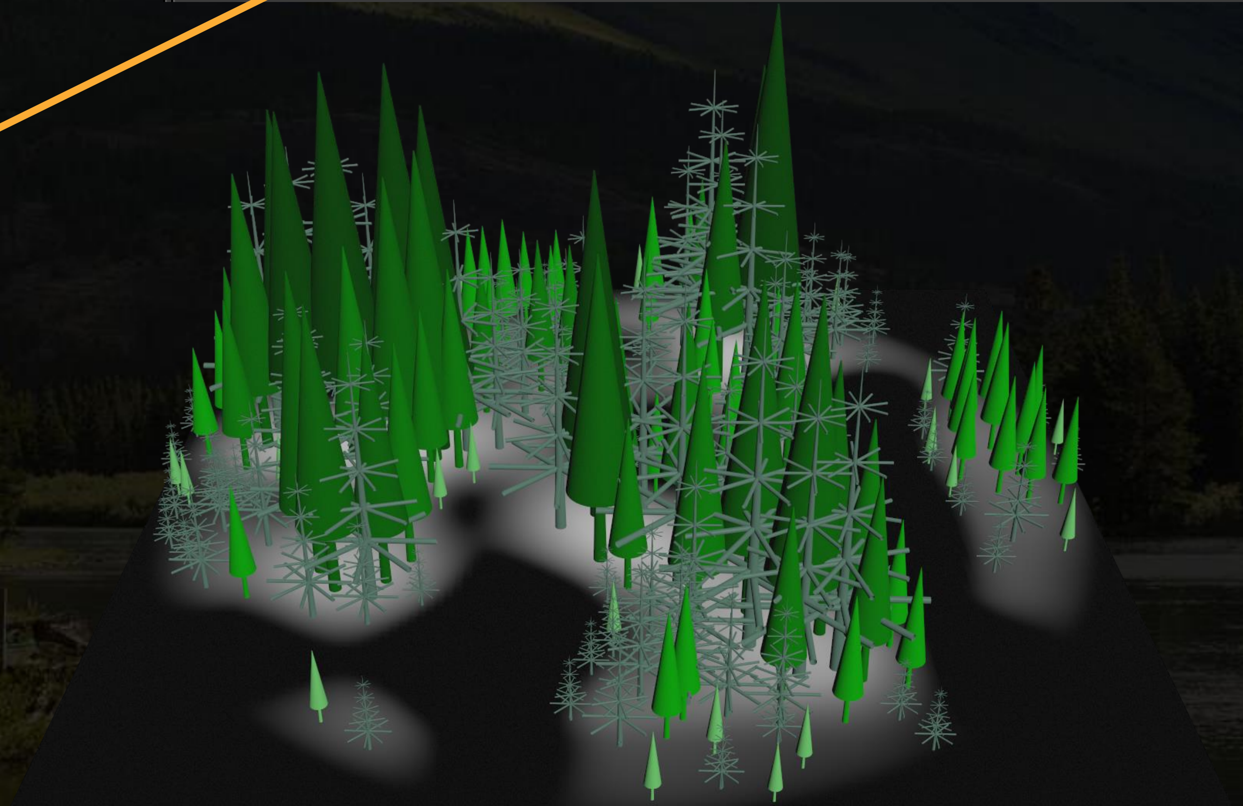
Number of Size 5 Clear

1 2 3 4 5

Variations for this size 2 Clear

Note: Conifer
Type: collection obj... ID 123456 % 1
✓ Snap to terrain
Debug Geometry: Primitive_Conifer_01 Meters 10

Note: Dead Conifer
Type: collection obj... ID 123456 % 1
✓ Snap to terrain
Debug Geometry: Primitive_ConiferDead_01 Meters 10



Forest Canopy

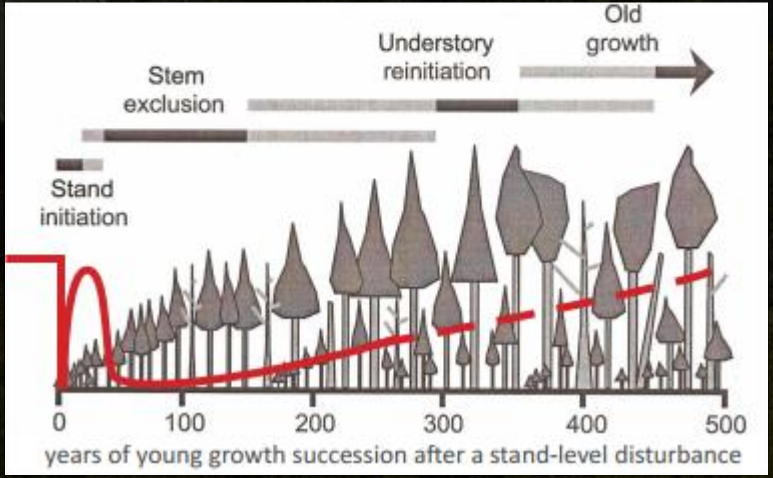
Ecological succession

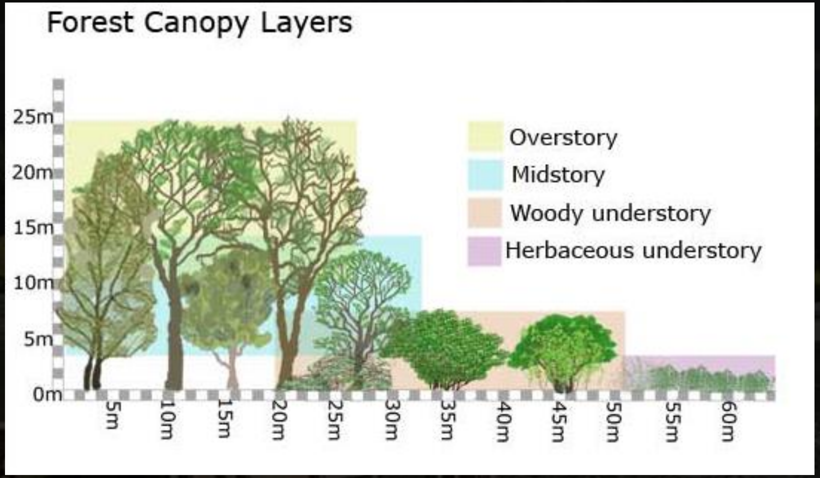
As we saw earlier, young tree will tend to be growing at the edge of a forest and old ones will be deeper inside. But there is also young regrowth possible inside a forest.

How can we achieve that?

We saw how the viability can affect the tree size selection.

However..





SOURCE: *http://www.riverpartners.org/resources/riparian-ecology/veg-wildlife-habitat/vegetation-structure/index.html*

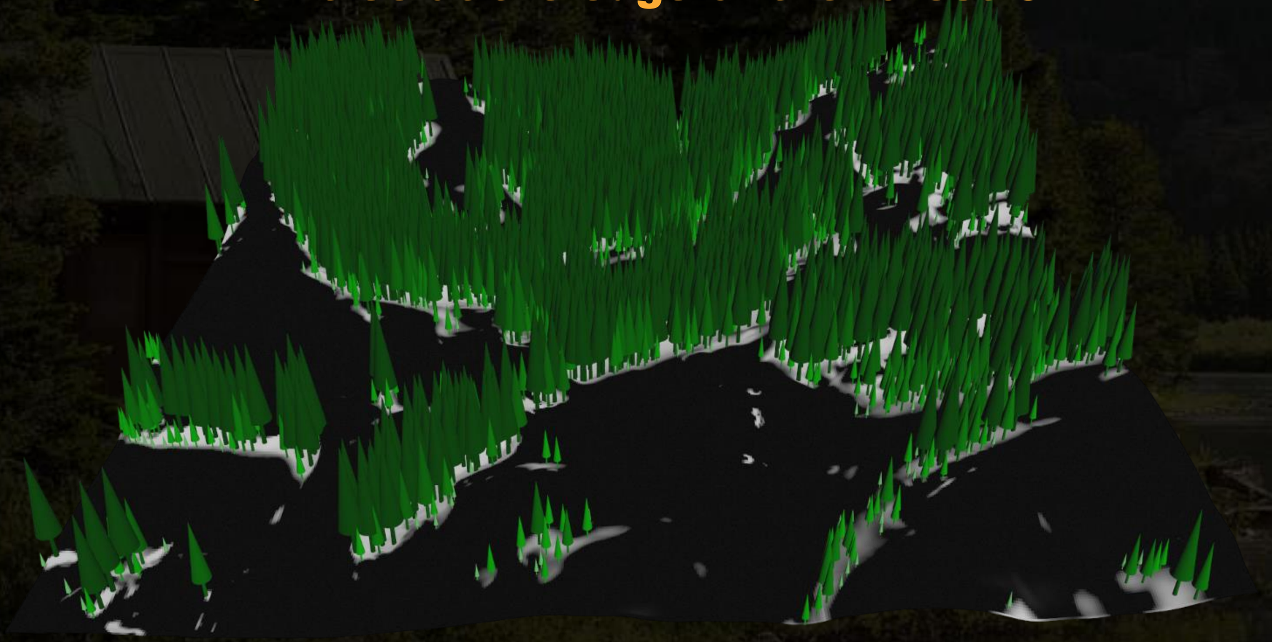
Age

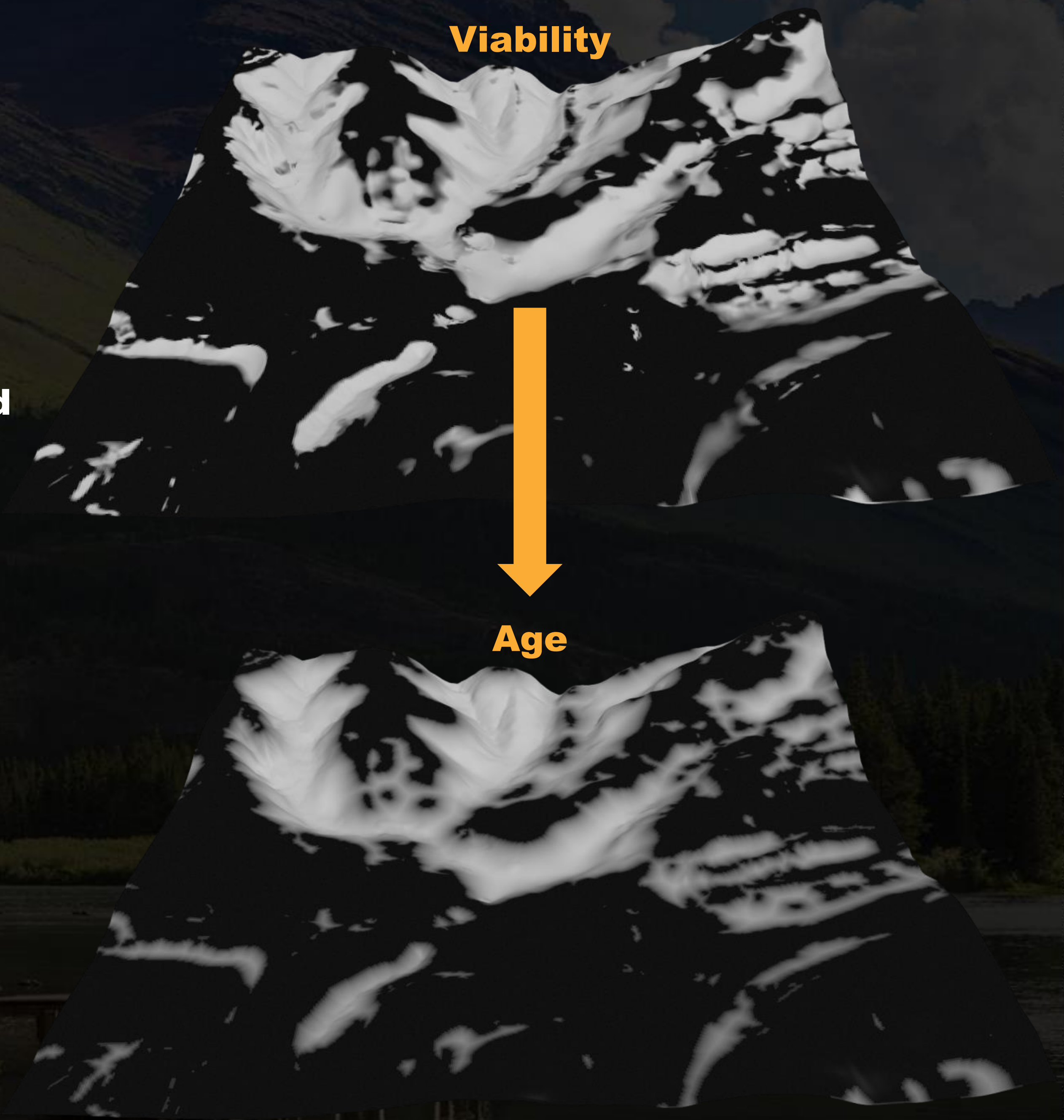
Signed distance field from viability

Depending on the terrain data used, we don't always end up with smooth gradient on the viability. Which can result of having our tallest trees at the edge of the forest for instance.

For this reason we added the "age" parameter which is basically a signed distance field generated from the viability.

Tall tree at the edge of the forest ☹

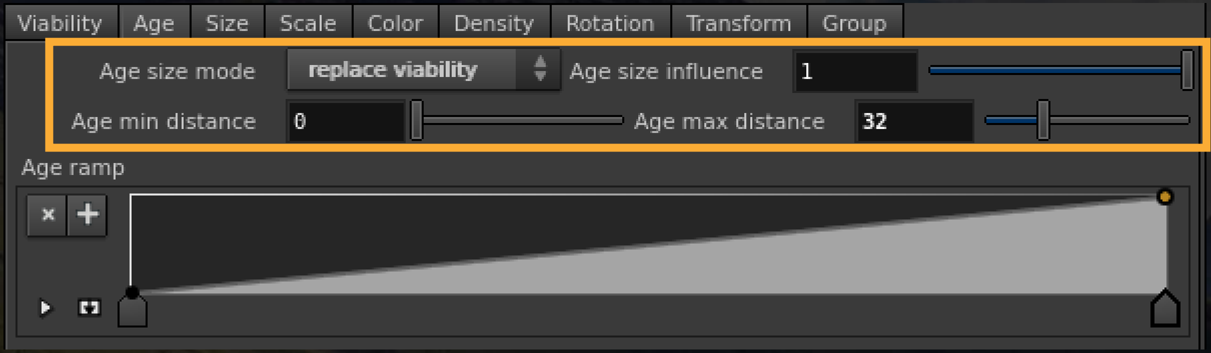




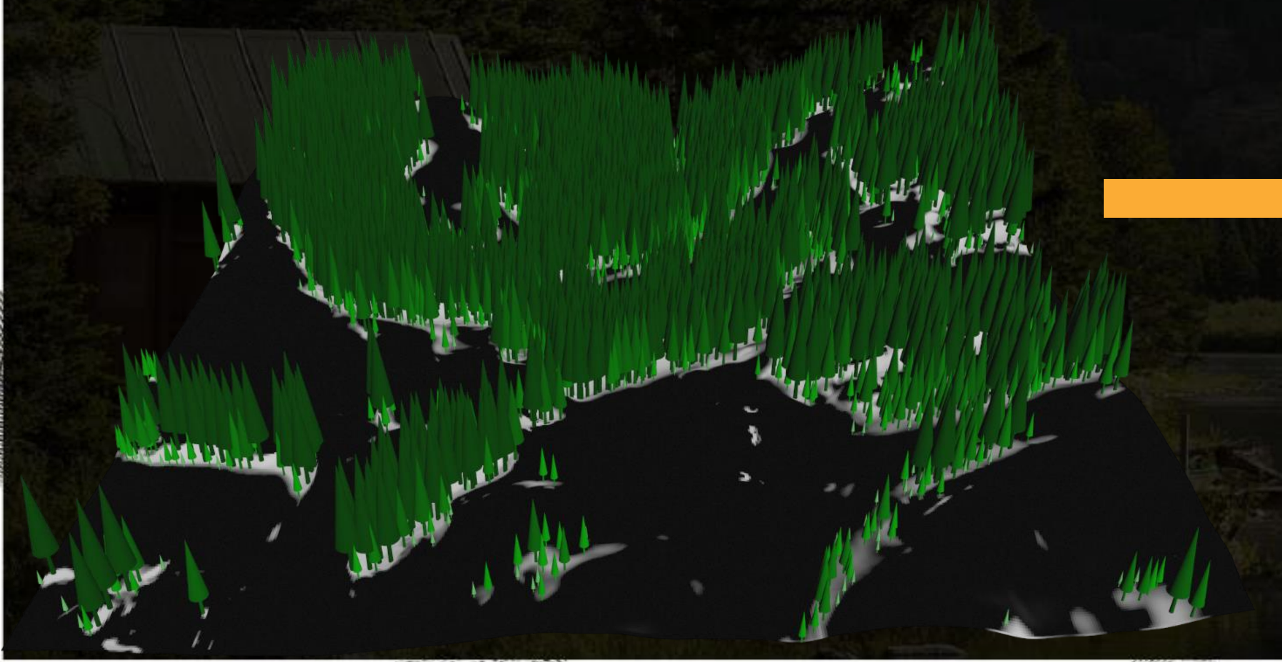
Age

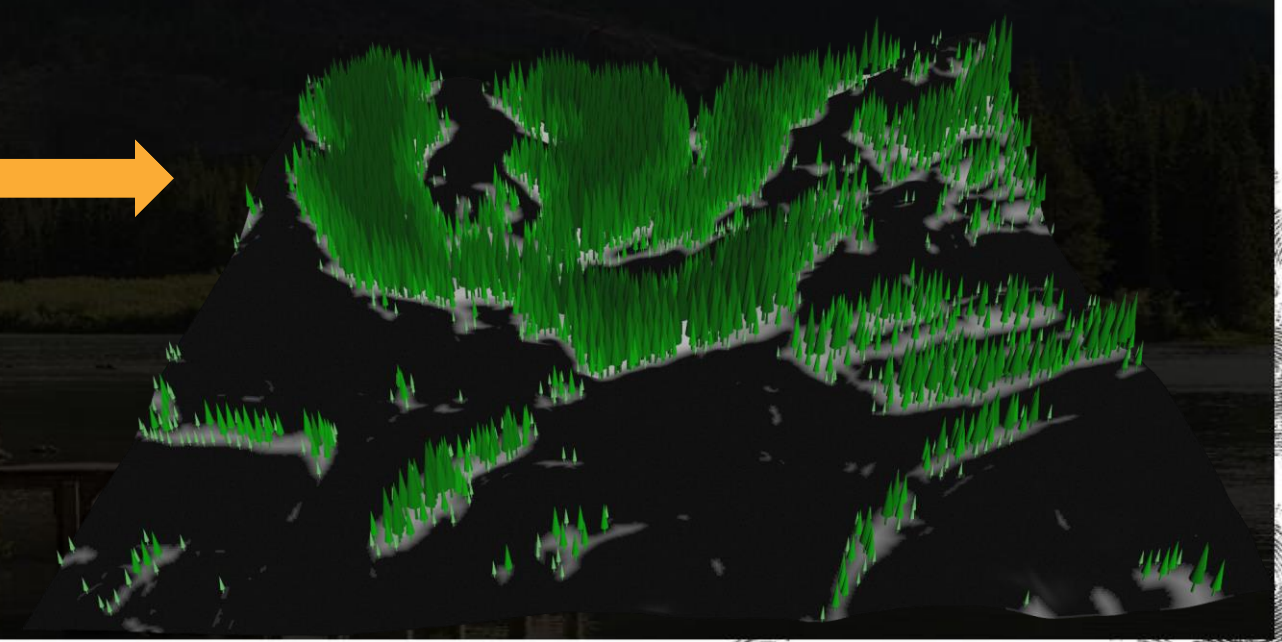
Age replacing viability for size selection

- We can control the amount of influence that the age has on the size selection
- Adjusting the age maximum distance gives us control on how deep the border of our forests will be.



Age size mode: replace viability
Age size influence: 1
Age min distance: 0
Age max distance: 32

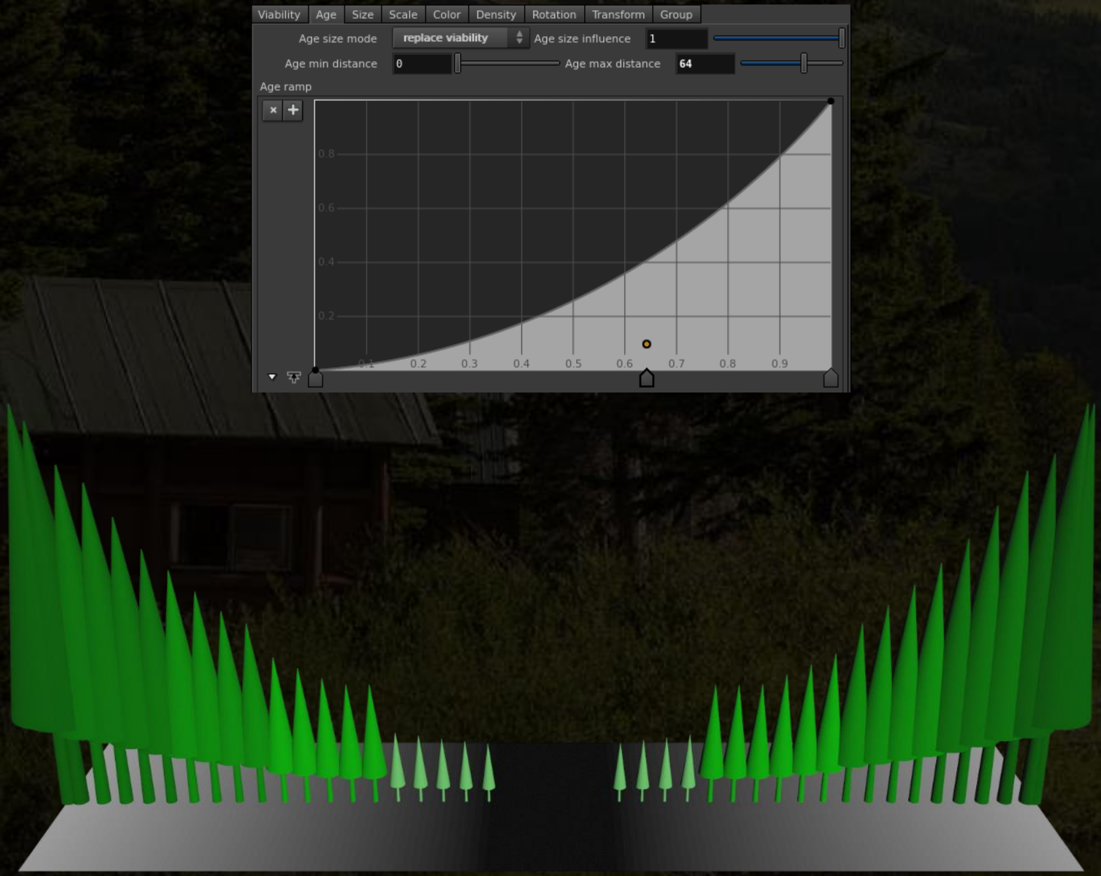




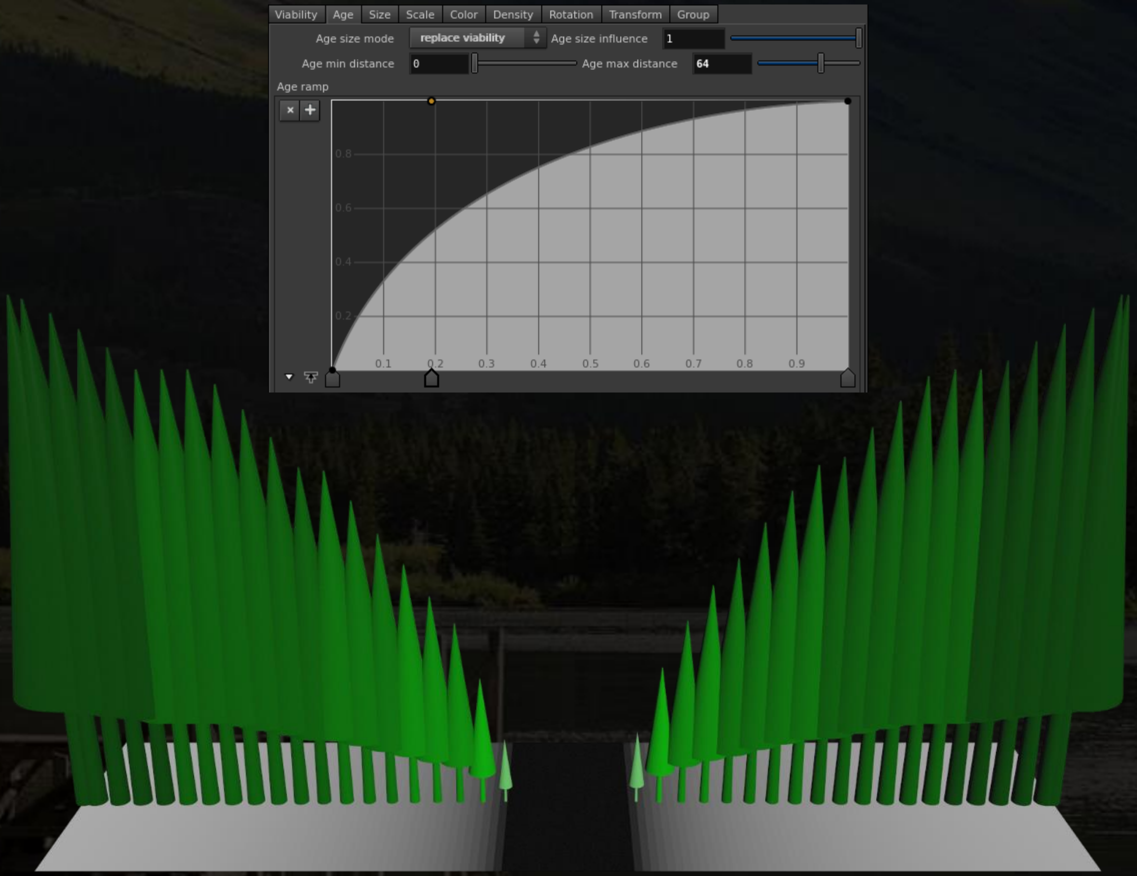
Age

Ramp for profile shape

- As a bonus side effect, by using a ramp on the age we can profile the shape of our forests.



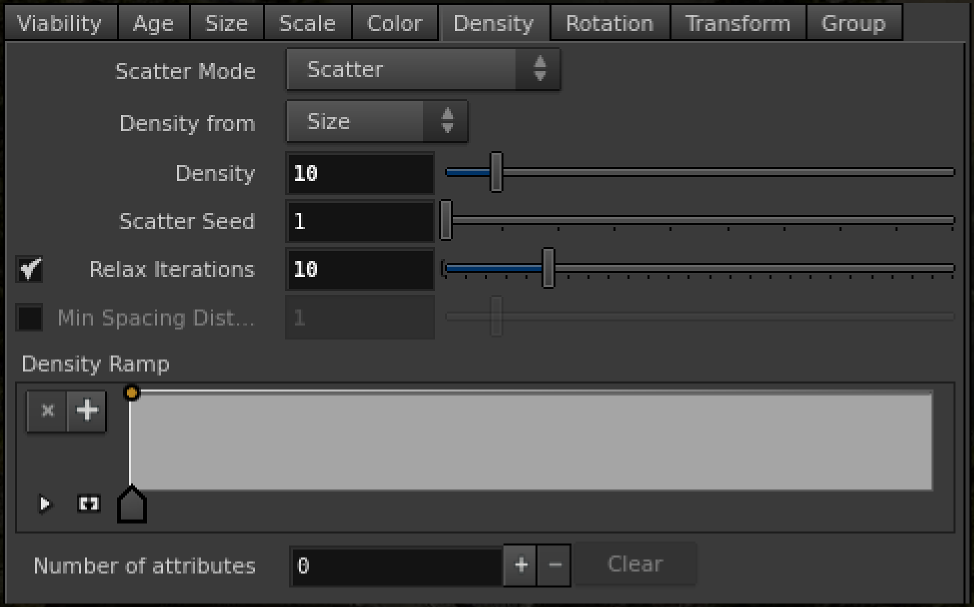
Age size mode: replace viability
Age size influence: 1
Age min distance: 0
Age max distance: 64



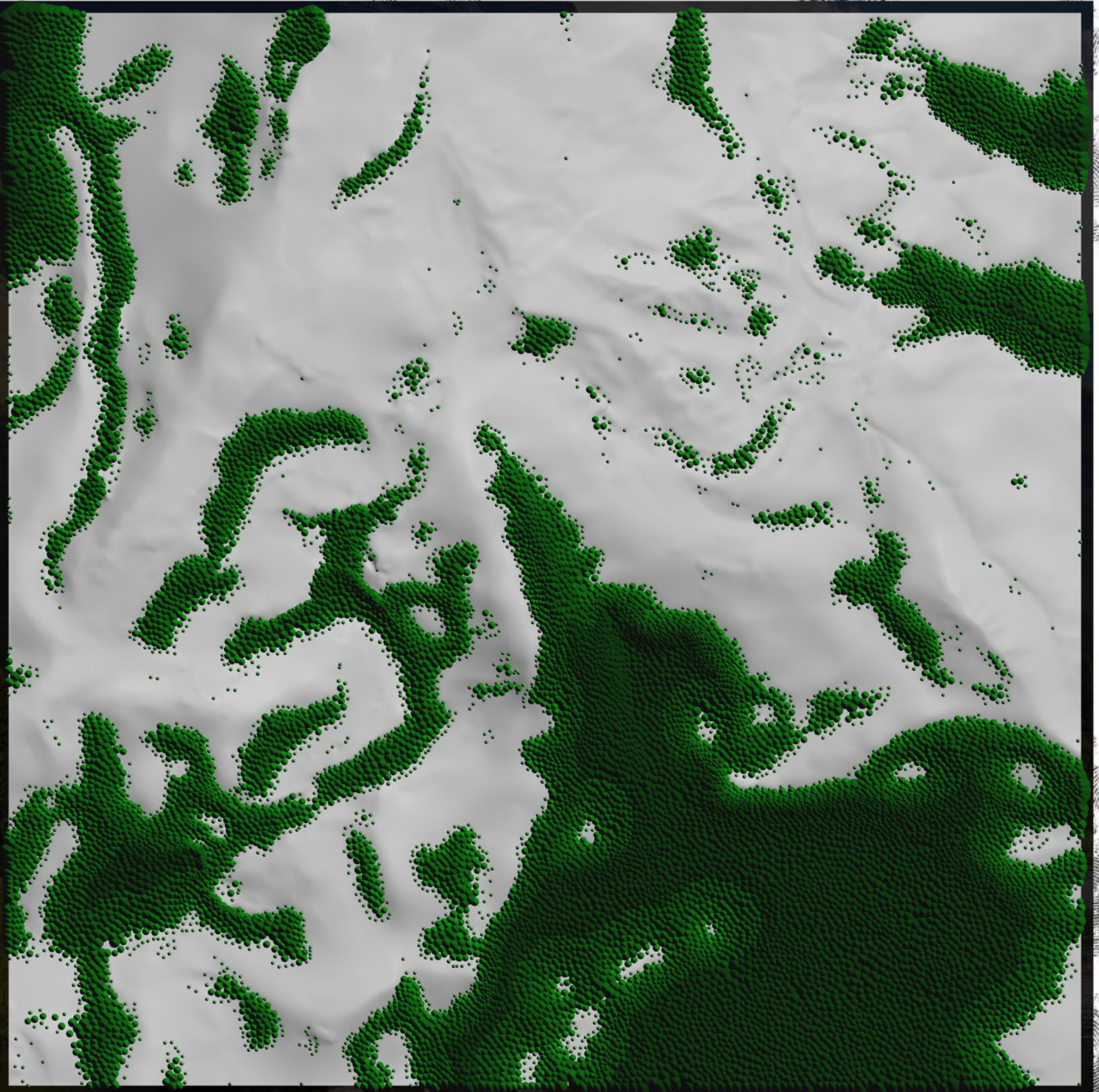
Age size mode: replace viability
Age size influence: 1
Age min distance: 0
Age max distance: 64

Density

Constant

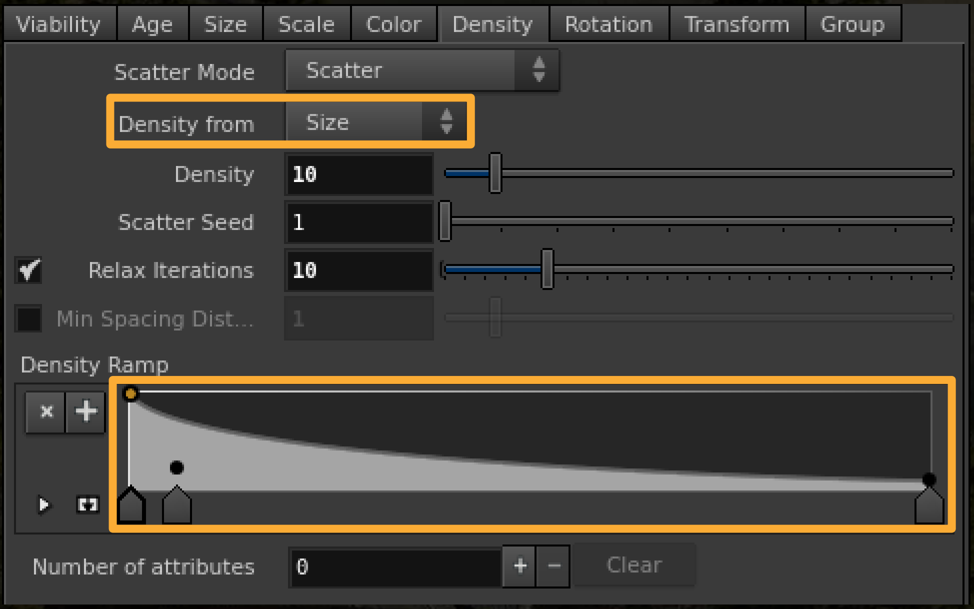


Scatter Mode: Scatter
Density from: Size
Density: 10
Scatter Seed: 1
Relax Iterations: 10 [✓]
Min Spacing Dist...: 1
Density Ramp
Number of attributes: 0

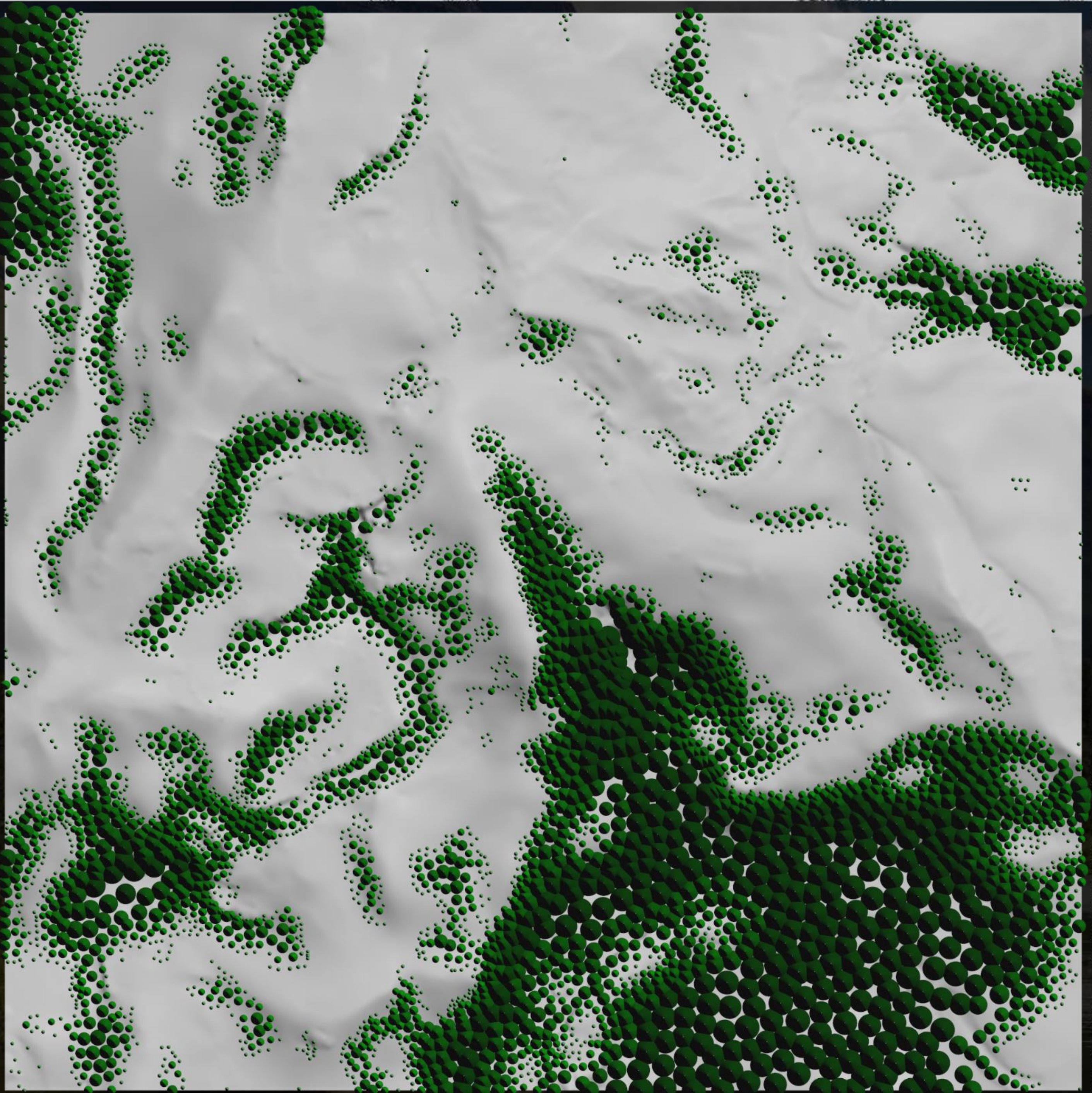


Density

Ramp from size



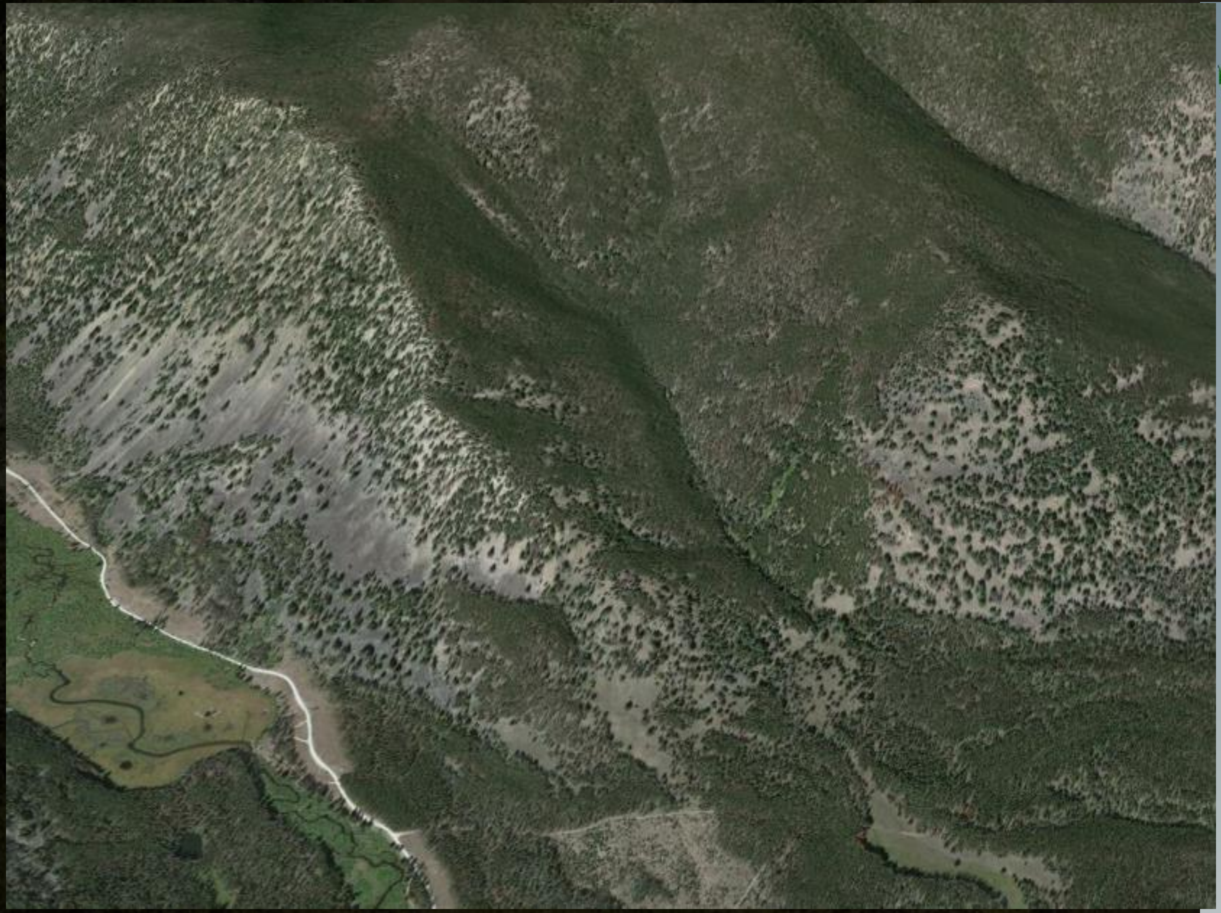
Scatter Mode: Scatter
Density from: Size
Density: 10
Scatter Seed: 1
✓ Relax Iterations: 10
Min Spacing Dist...: 1
Density Ramp
Number of attributes: 0

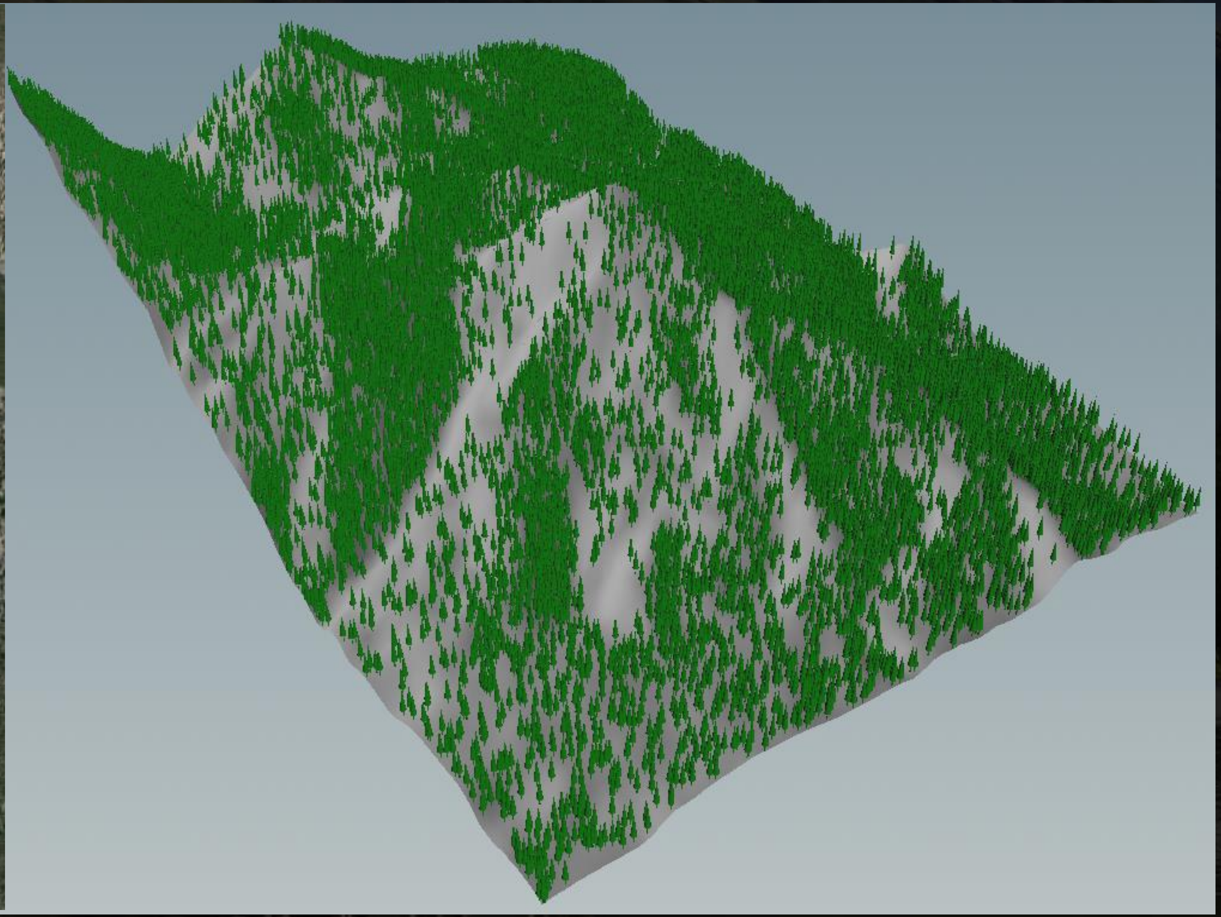


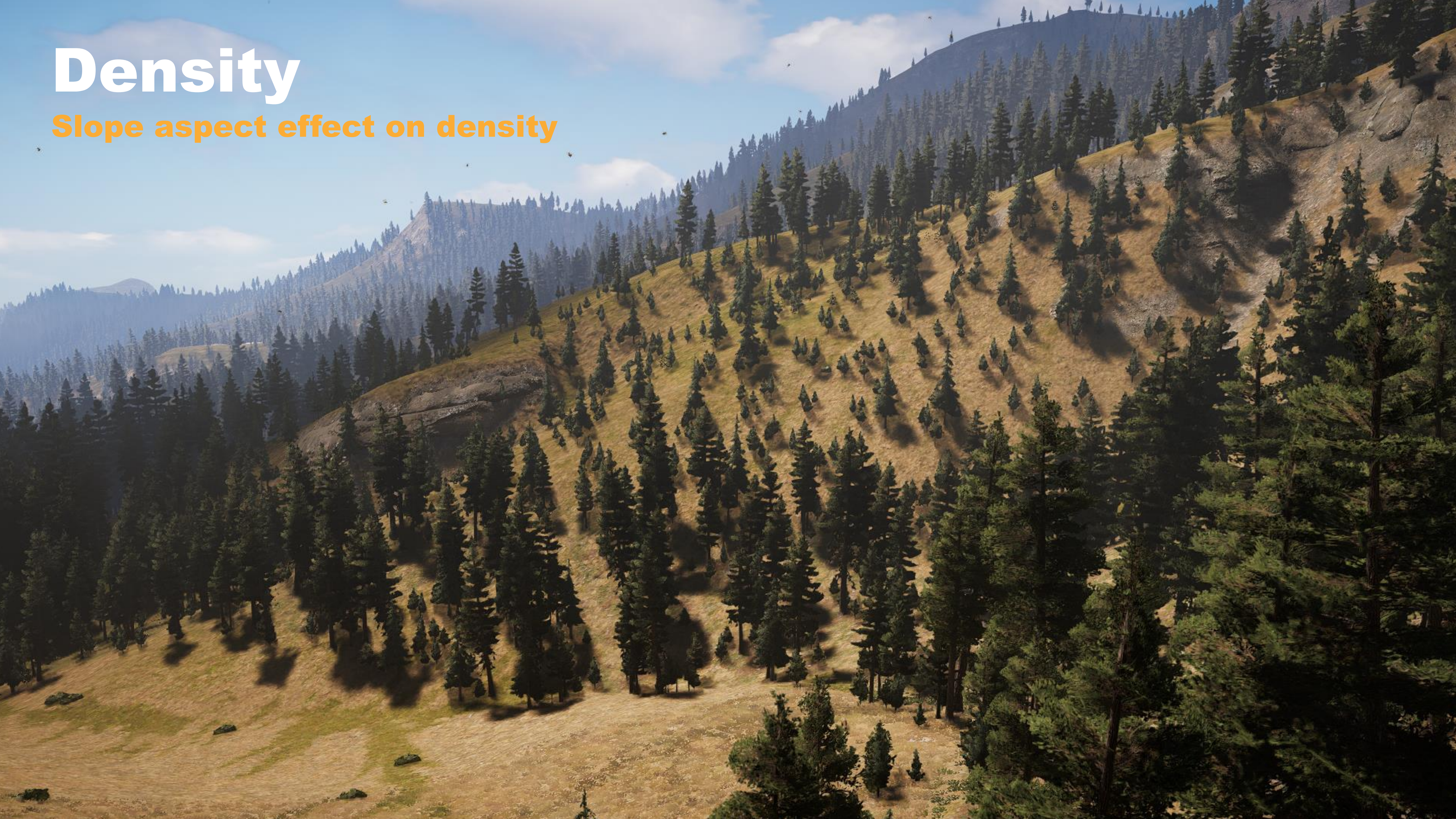
Density

Slope aspect effect on density

- We can also use terrain data such as illumination (slope aspect) to affect density directly.

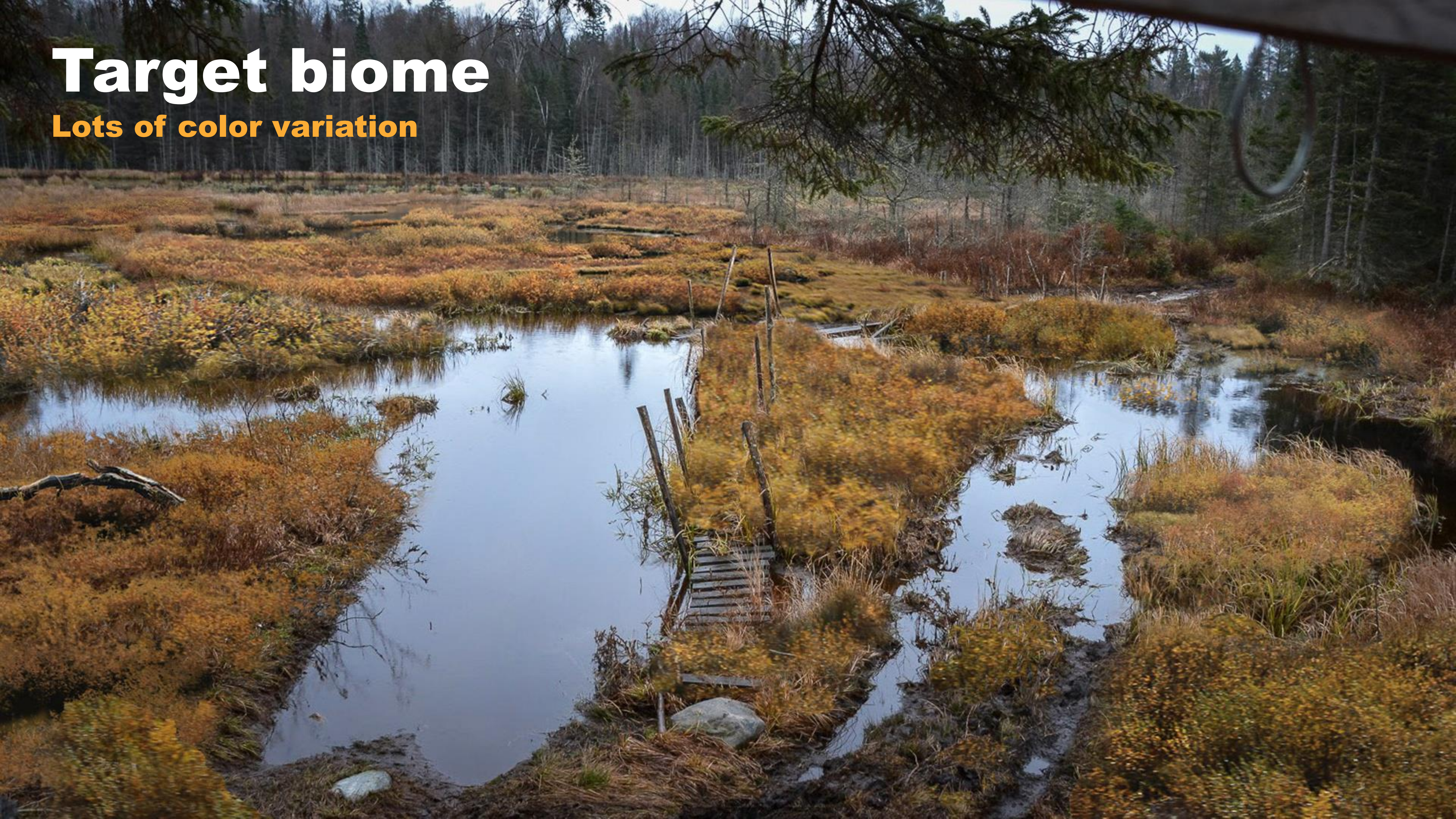






Density

Slope aspect effect on density



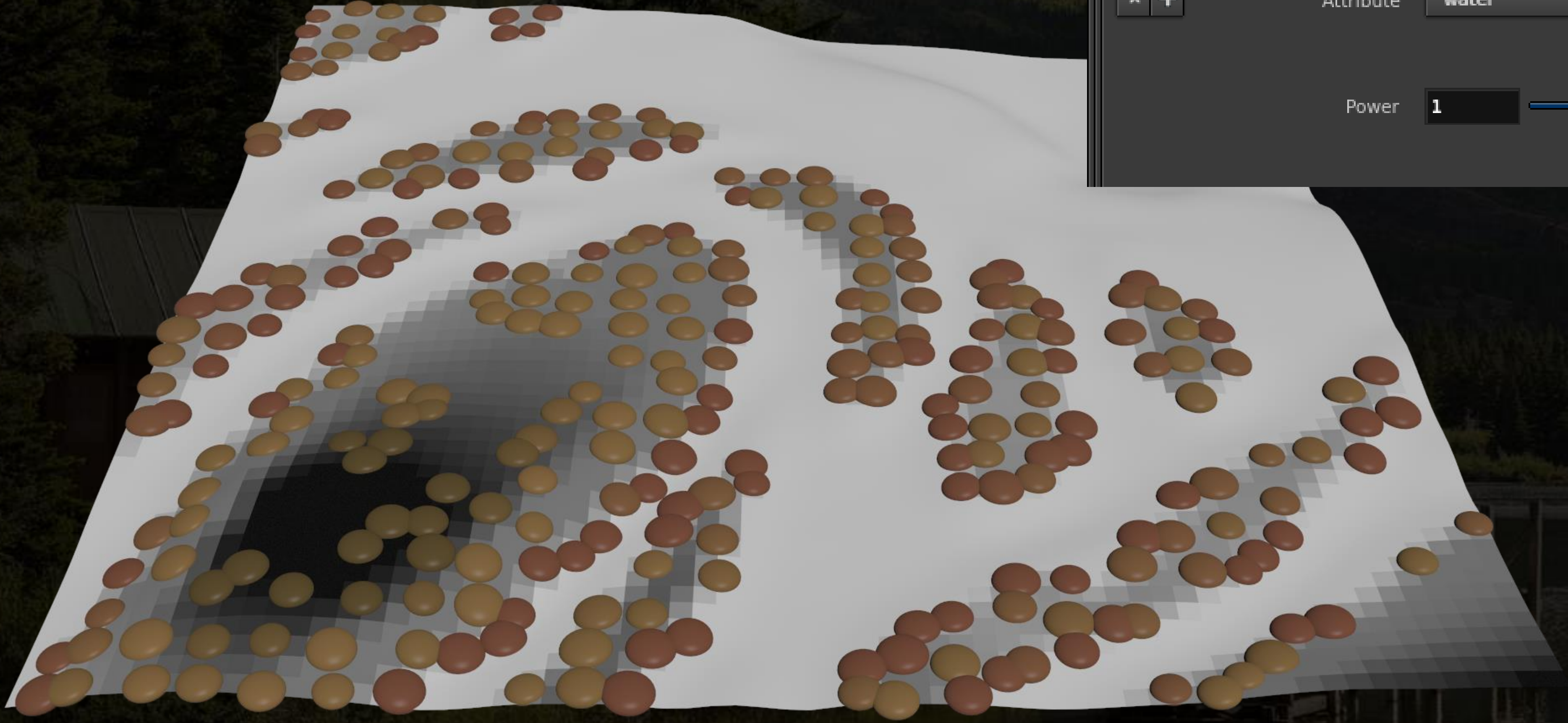
Target biome

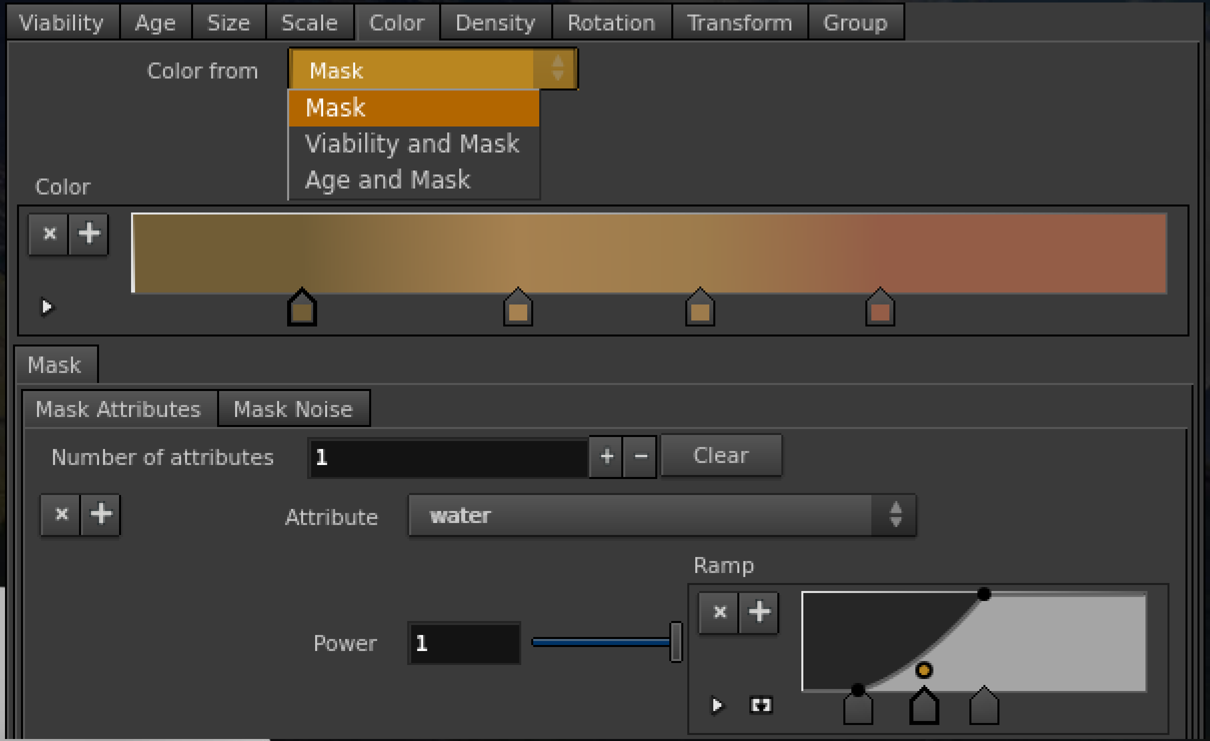
Lots of color variation

Color

Per instance color variation

- Using water signed distance field mask to drive color variation of this grass species




Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

Color from: Mask

Mask
Viability and Mask
Age and Mask

Color

Mask

Mask Attributes | Mask Noise

Number of attributes: 1

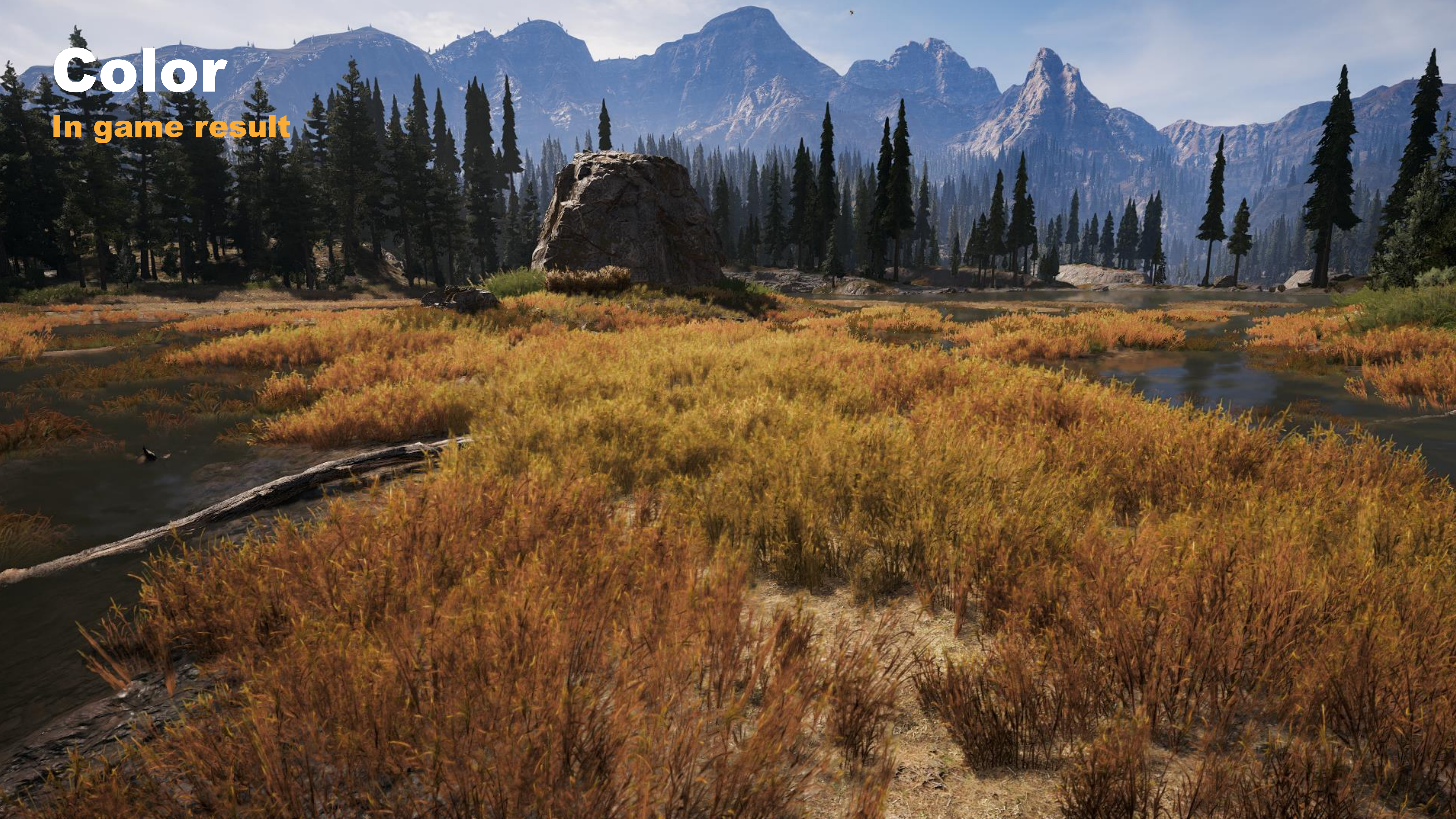
Attribute: water

Ramp

Power: 1

Color

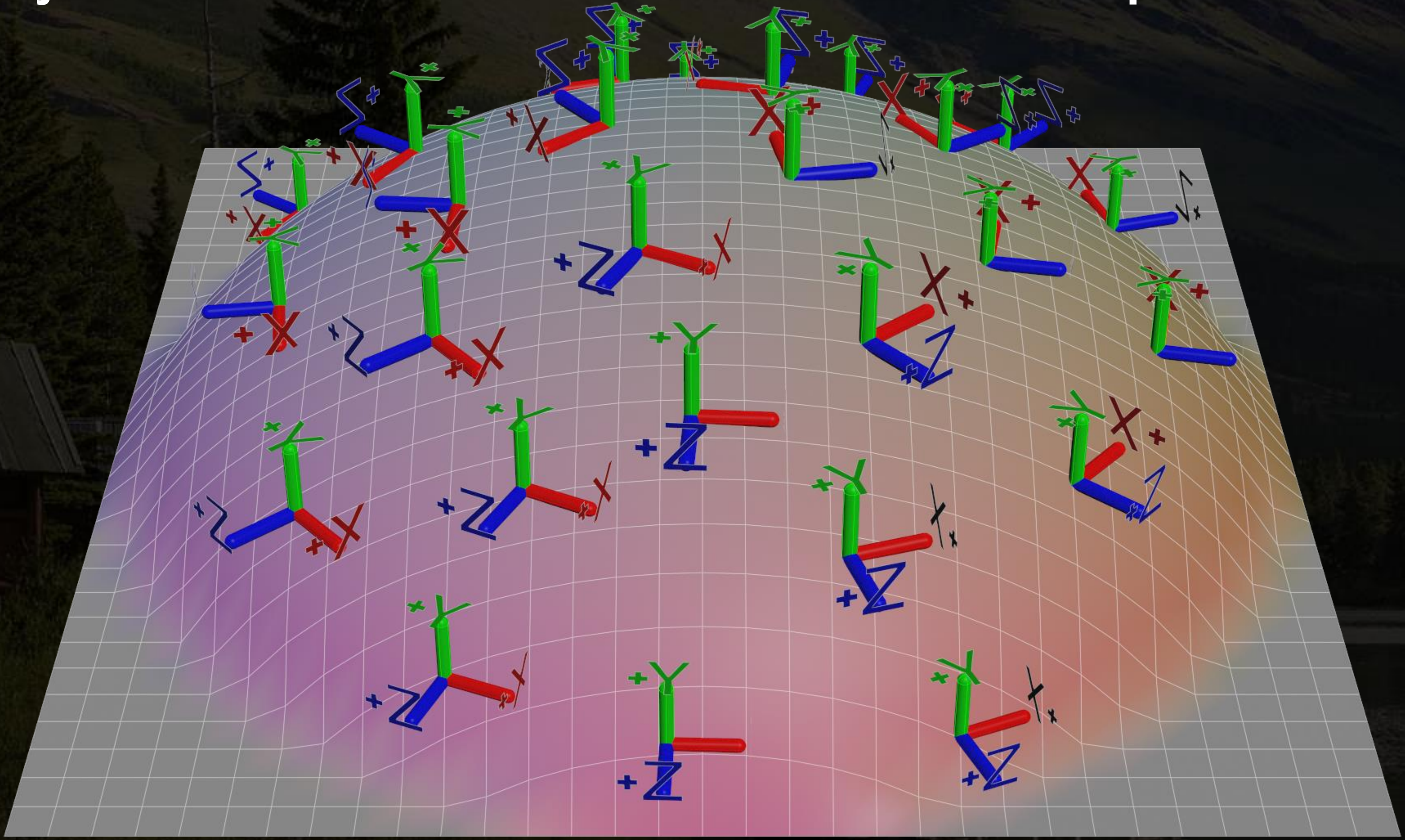
In game result



Rotation

Orient on terrain slope

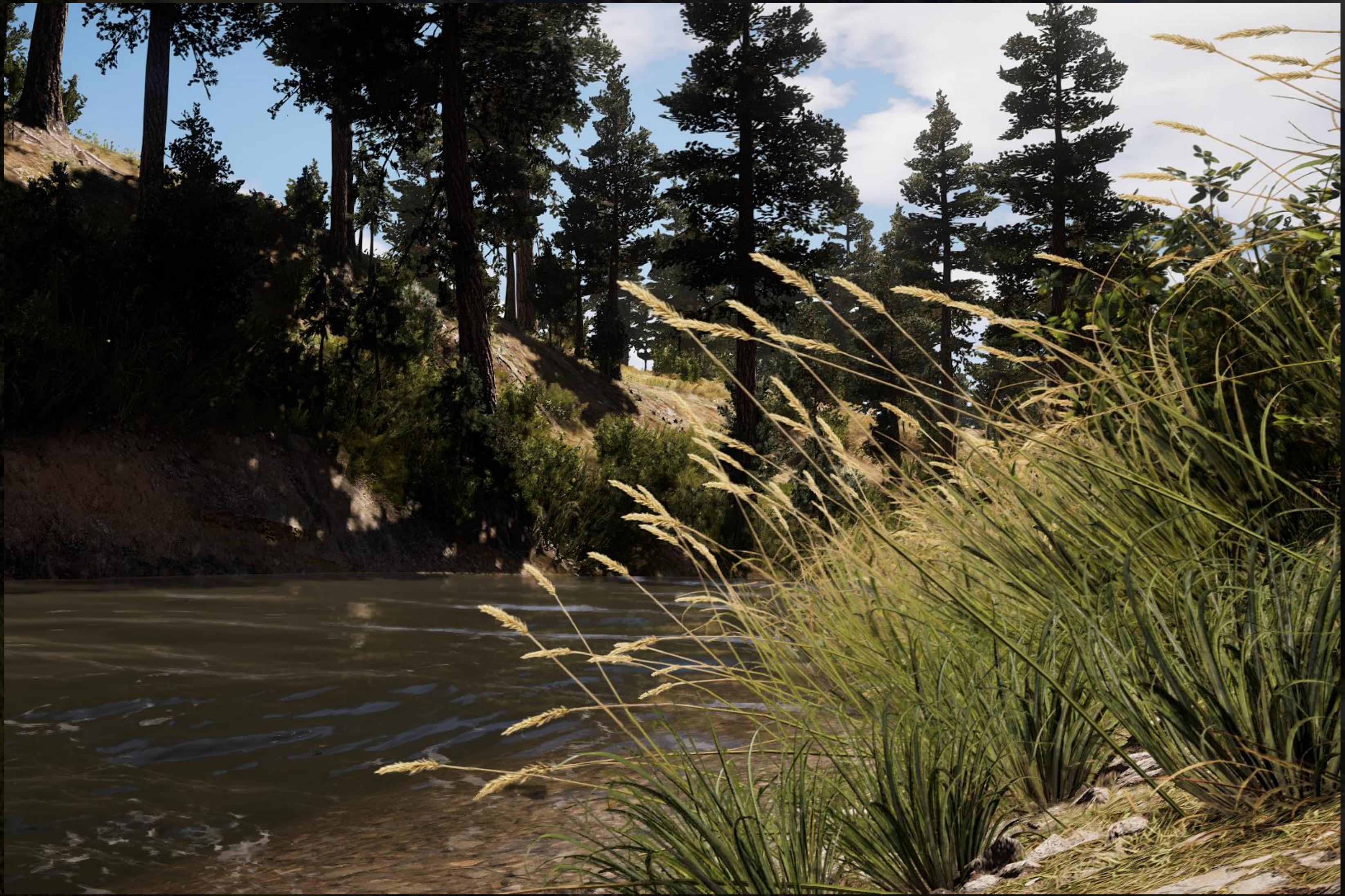
- The scattered entities also need to have their own specific rotation.
- By default they have their forward axis oriented towards the terrain slope.



Rotation

Grass leaning towards water example

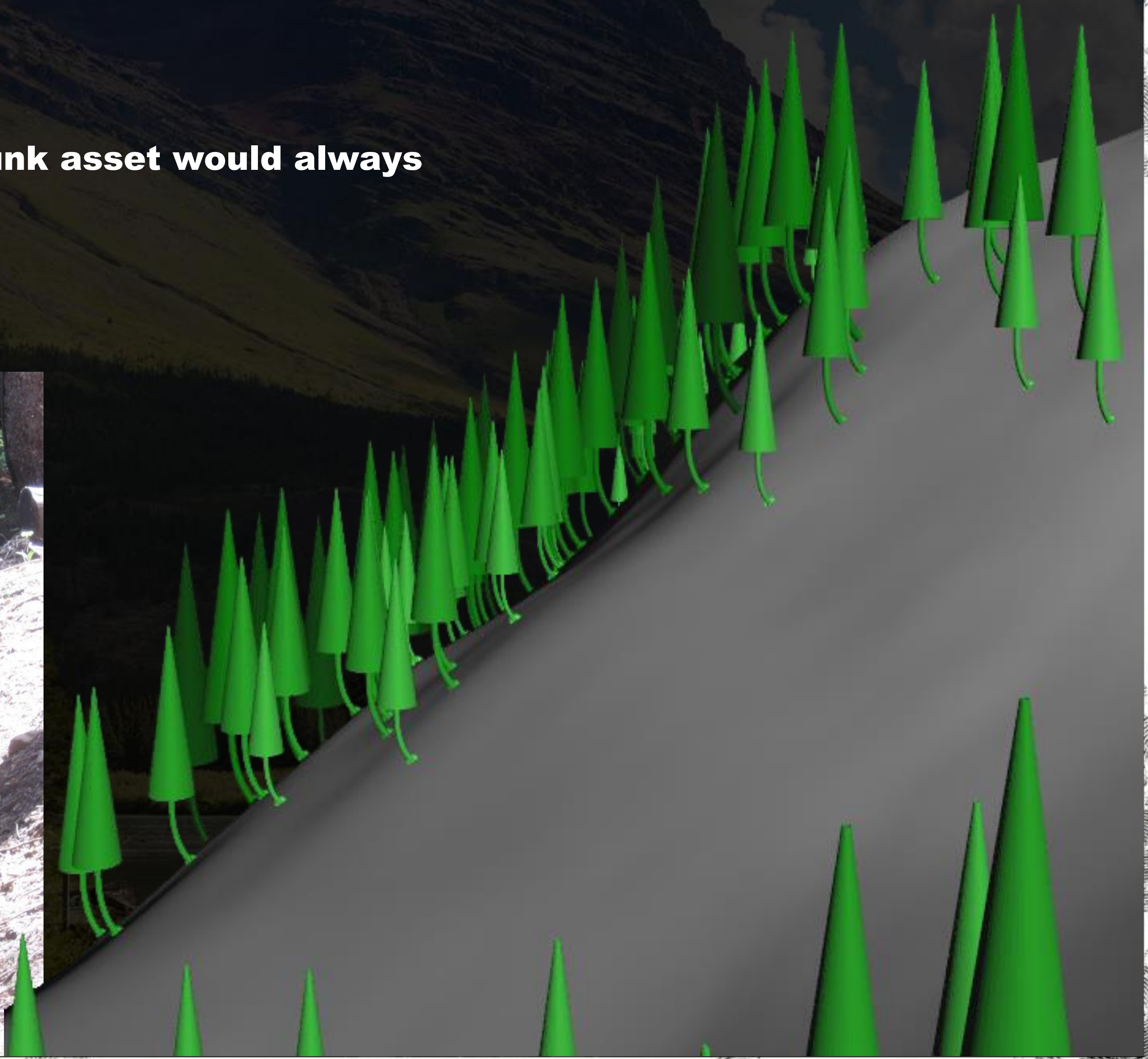
Because assets forward axis are oriented towards terrain slope it allows us to do things like this. This grass asset will always be leaning towards the water.



Rotation

Bended tree trunks example

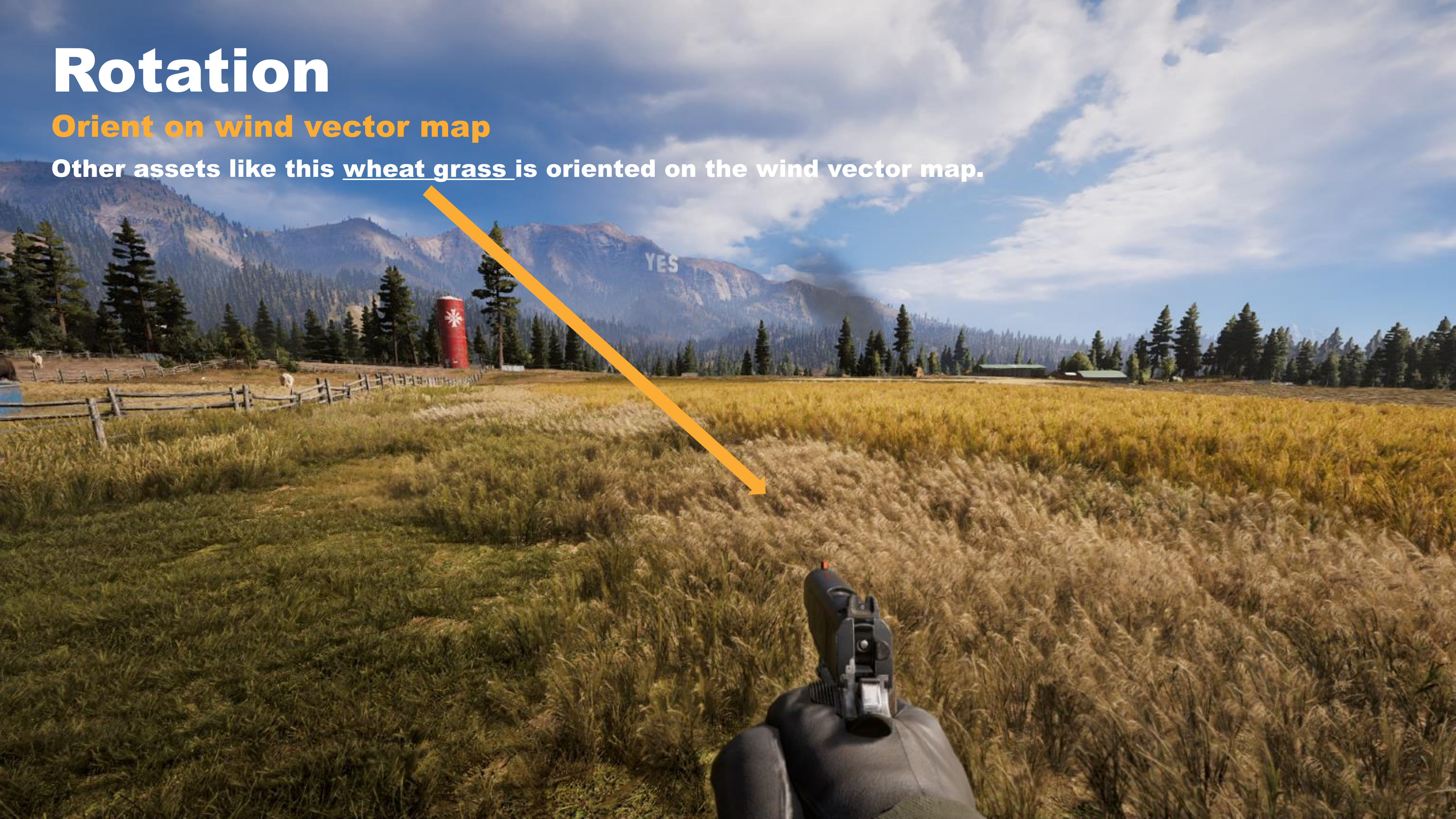
Another example... This pre-bended tree trunk asset would always be oriented properly towards terrain slope



Rotation

Orient on wind vector map

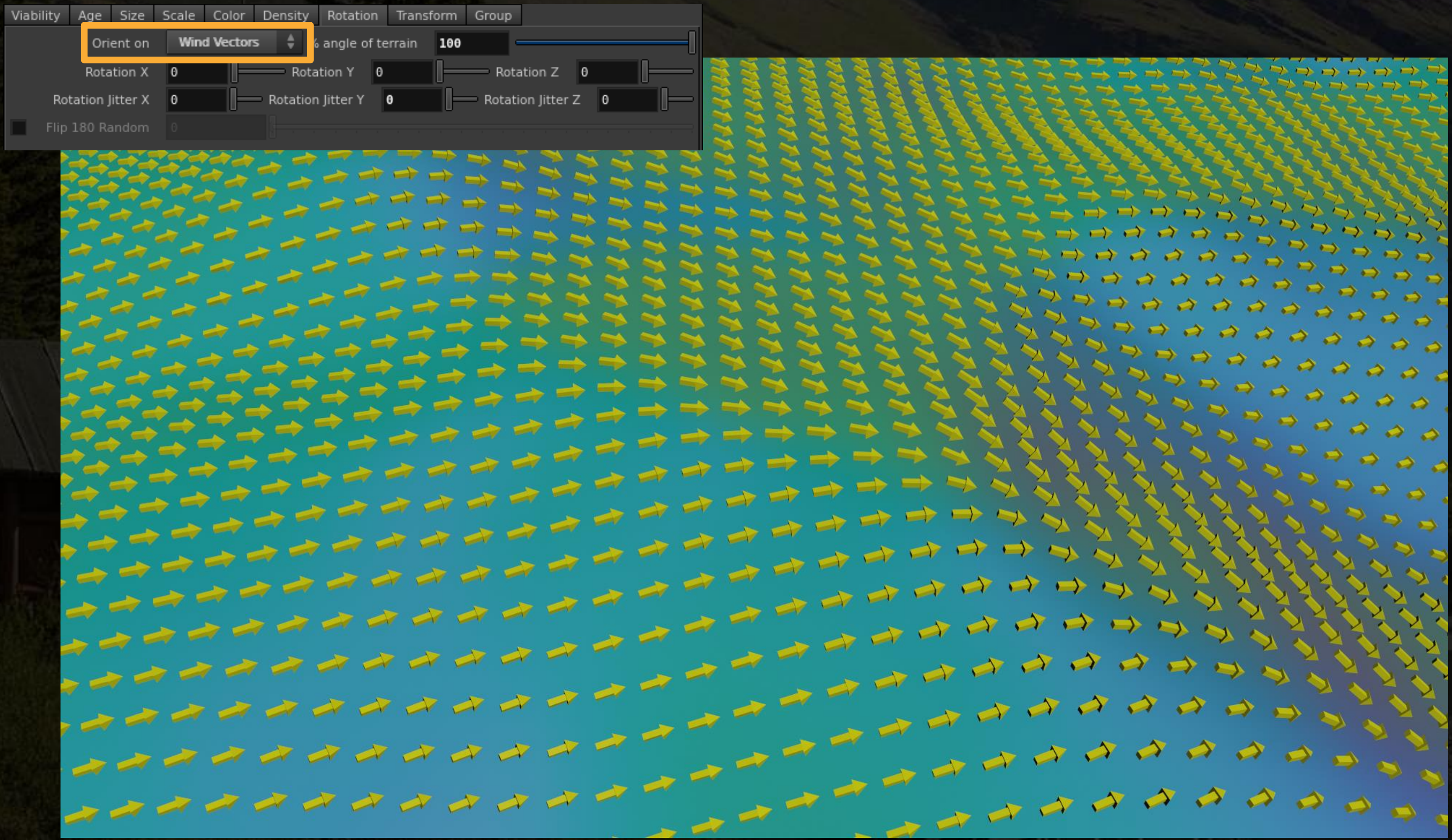
Other assets like this wheat grass is oriented on the wind vector map.



Rotation

Orient on wind vector map

The wind map is based on an overall wind direction but it is fluctuating slightly based on the terrain shapes as well. When blowing against a hill, it will tend to flow around it.

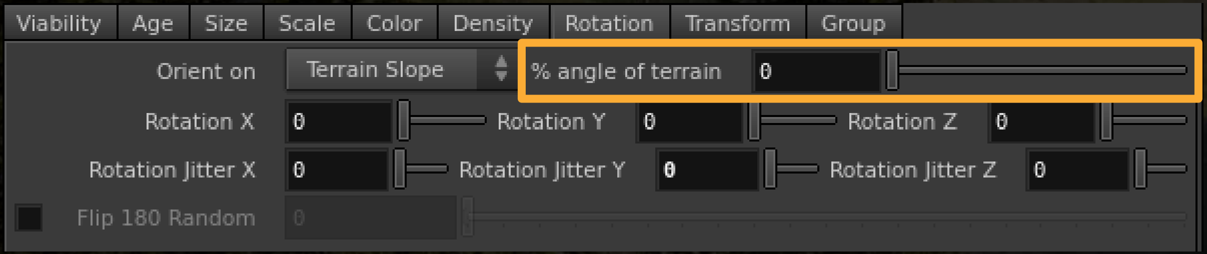


Orient on: Wind Vectors
% angle of terrain: 100
Rotation X: 0
Rotation Y: 0
Rotation Z: 0
Rotation Jitter X: 0
Rotation Jitter Y: 0
Rotation Jitter Z: 0
Flip 180 Random: 0

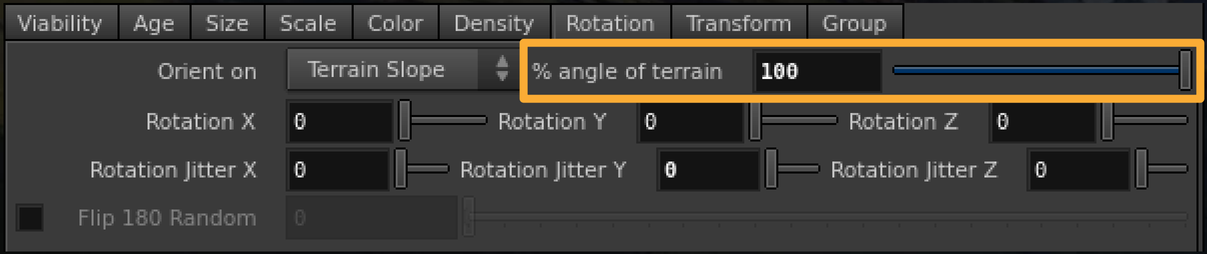
Rotation

% angle of terrain

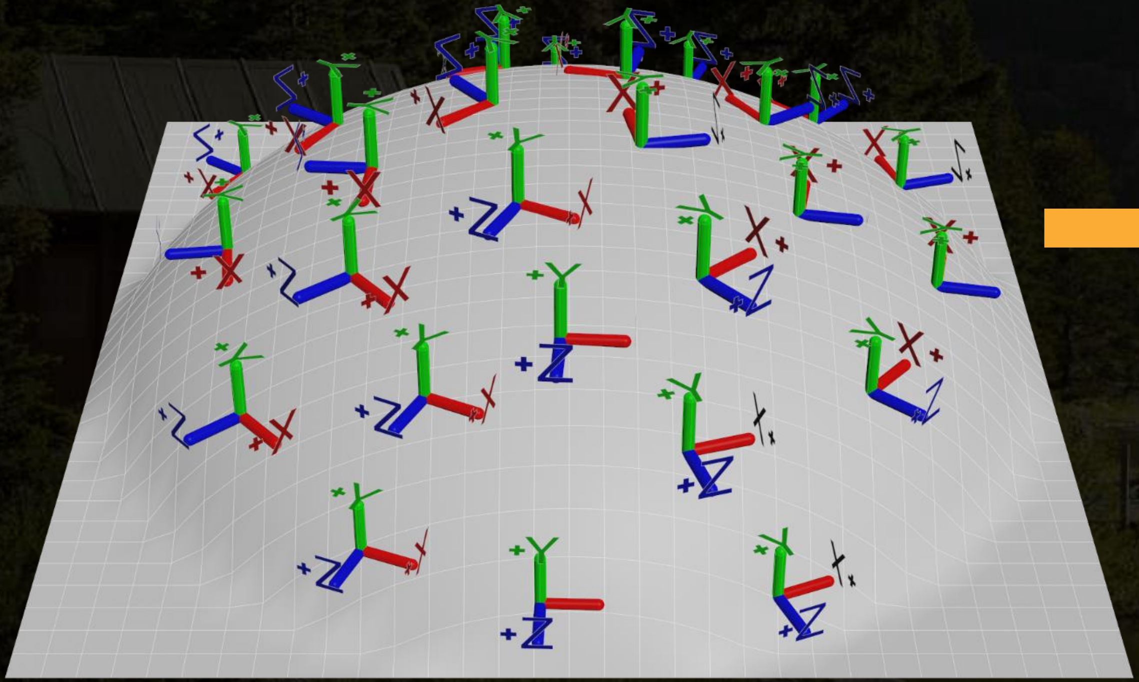
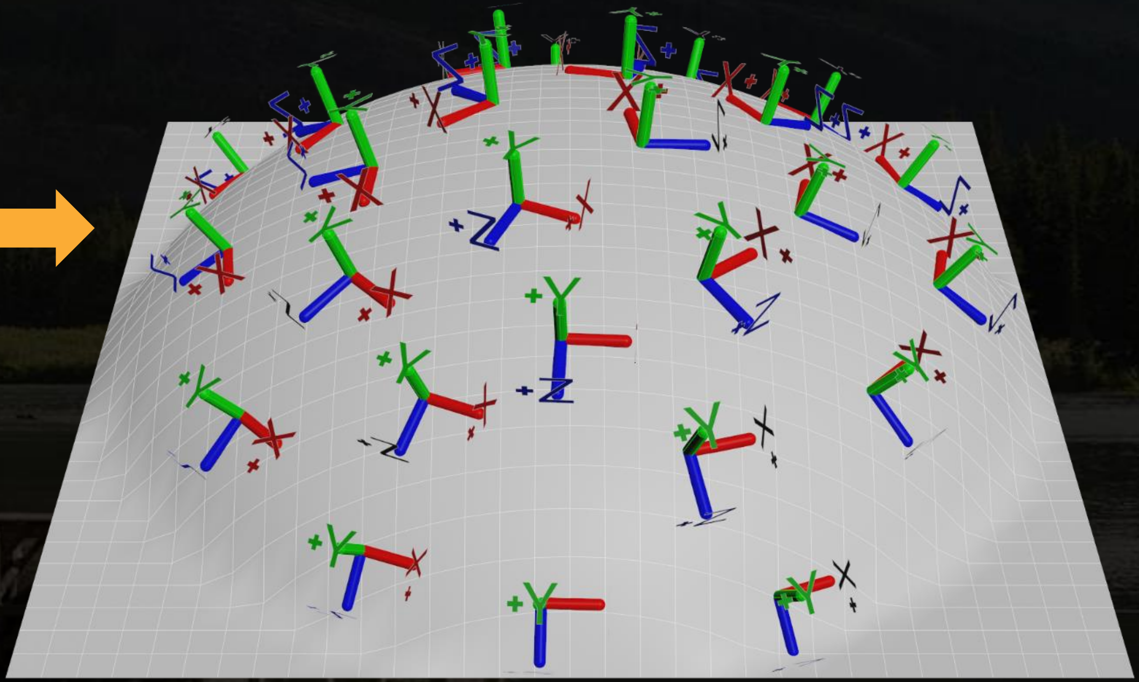
With all orientation options, the asset can be horizontal or aligned to the terrain slope controllable by a percentage



Orient on: Terrain Slope
% angle of terrain: 0
Rotation X: 0
Rotation Y: 0
Rotation Z: 0
Rotation Jitter X: 0
Rotation Jitter Y: 0
Rotation Jitter Z: 0
Flip 180 Random: 0

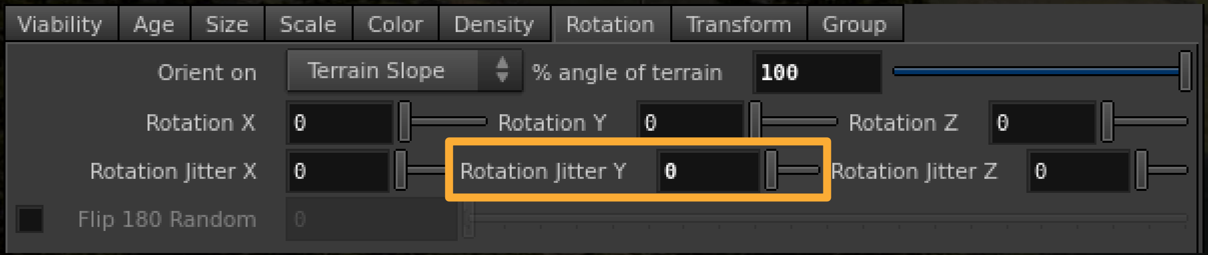


Orient on: Terrain Slope
% angle of terrain: 100
Rotation X: 0
Rotation Y: 0
Rotation Z: 0
Rotation Jitter X: 0
Rotation Jitter Y: 0
Rotation Jitter Z: 0
Flip 180 Random: 0

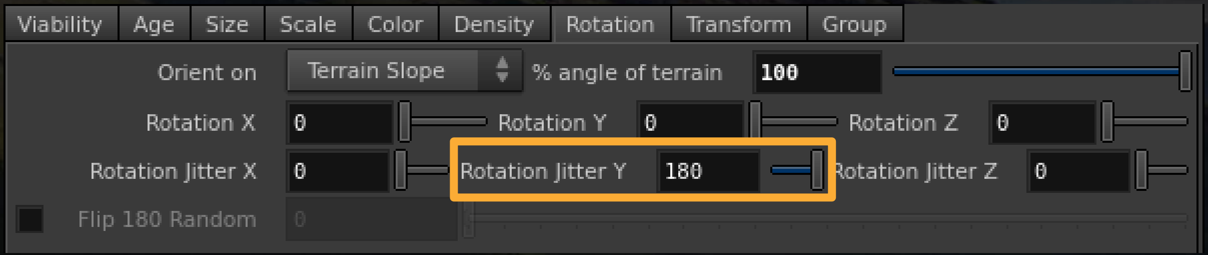



Rotation

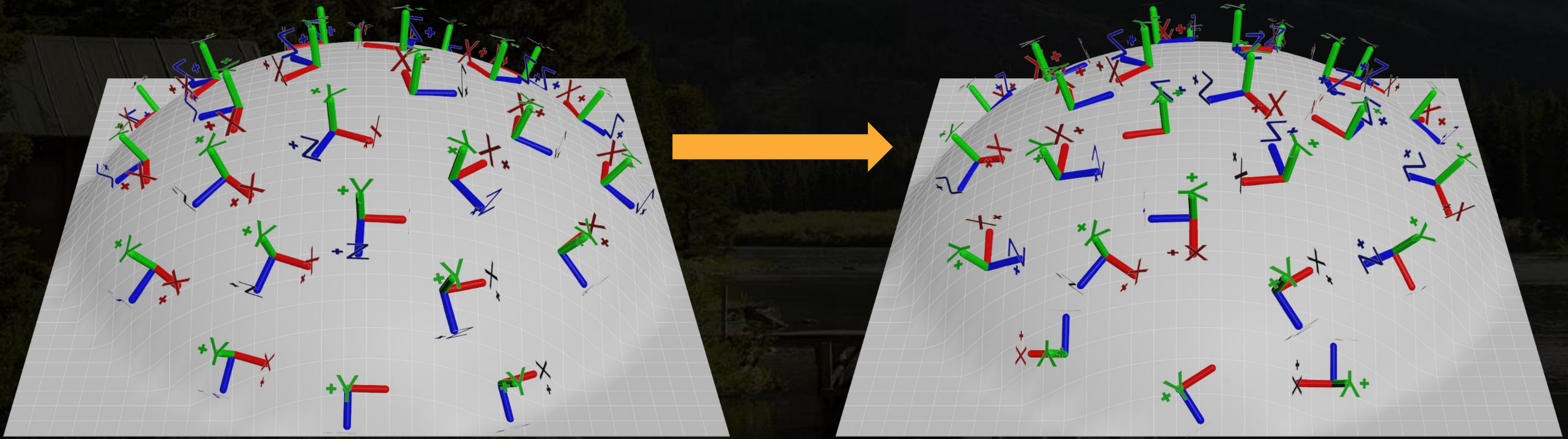
Jitters


Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

Orient on: Terrain Slope % angle of terrain: 100
Rotation X: 0 Rotation Y: 0 Rotation Z: 0
Rotation Jitter X: 0 Rotation Jitter Y: 0 Rotation Jitter Z: 0
Flip 180 Random: 0


Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

Orient on: Terrain Slope % angle of terrain: 100
Rotation X: 0 Rotation Y: 0 Rotation Z: 0
Rotation Jitter X: 0 Rotation Jitter Y: 180 Rotation Jitter Z: 0
Flip 180 Random: 0



Terrain

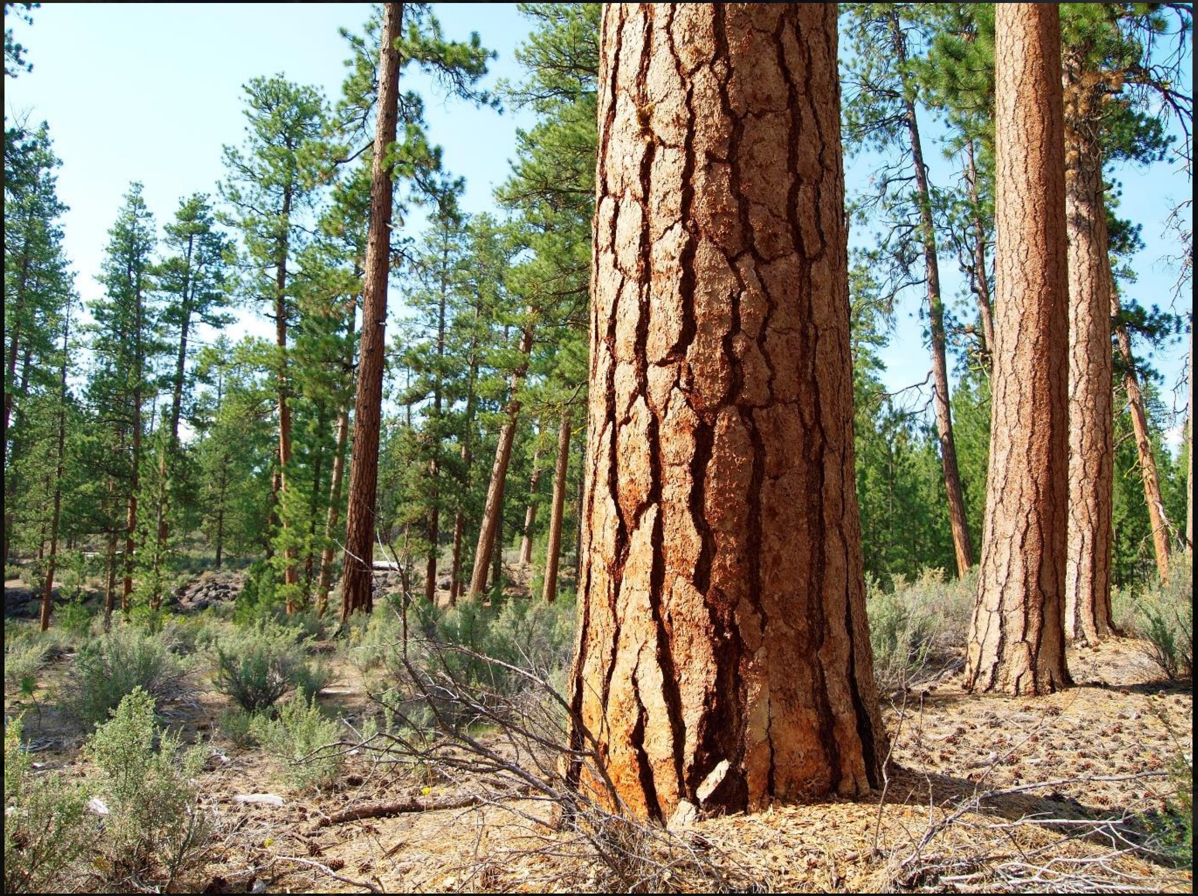
Observing nature

The biomes are not be limited to asset placement only. The positions of assets can also have an effet on:

- **Some terrain properties**
- **The surrounding assets as well**

In this picture, we can observe 4 different things that the presence of those trees influence visually in their surroundings:

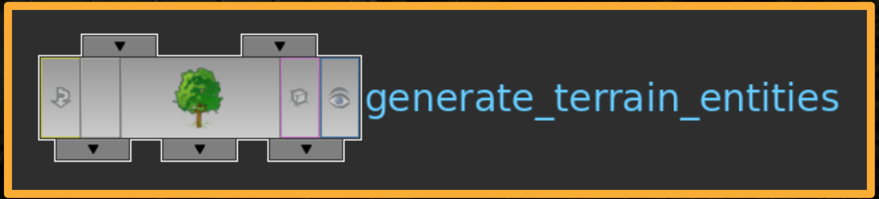
1. **The terrain is covered with pine needles (so terrain texture is affected)**
2. **We have a slight terrain elevation arround the trunks (so the terrain heightmap is affected too)**
3. **We have pinecones and dead branches on the ground**
4. **And the shadow provided by the trees might encourage or prevent the growth of other species.**

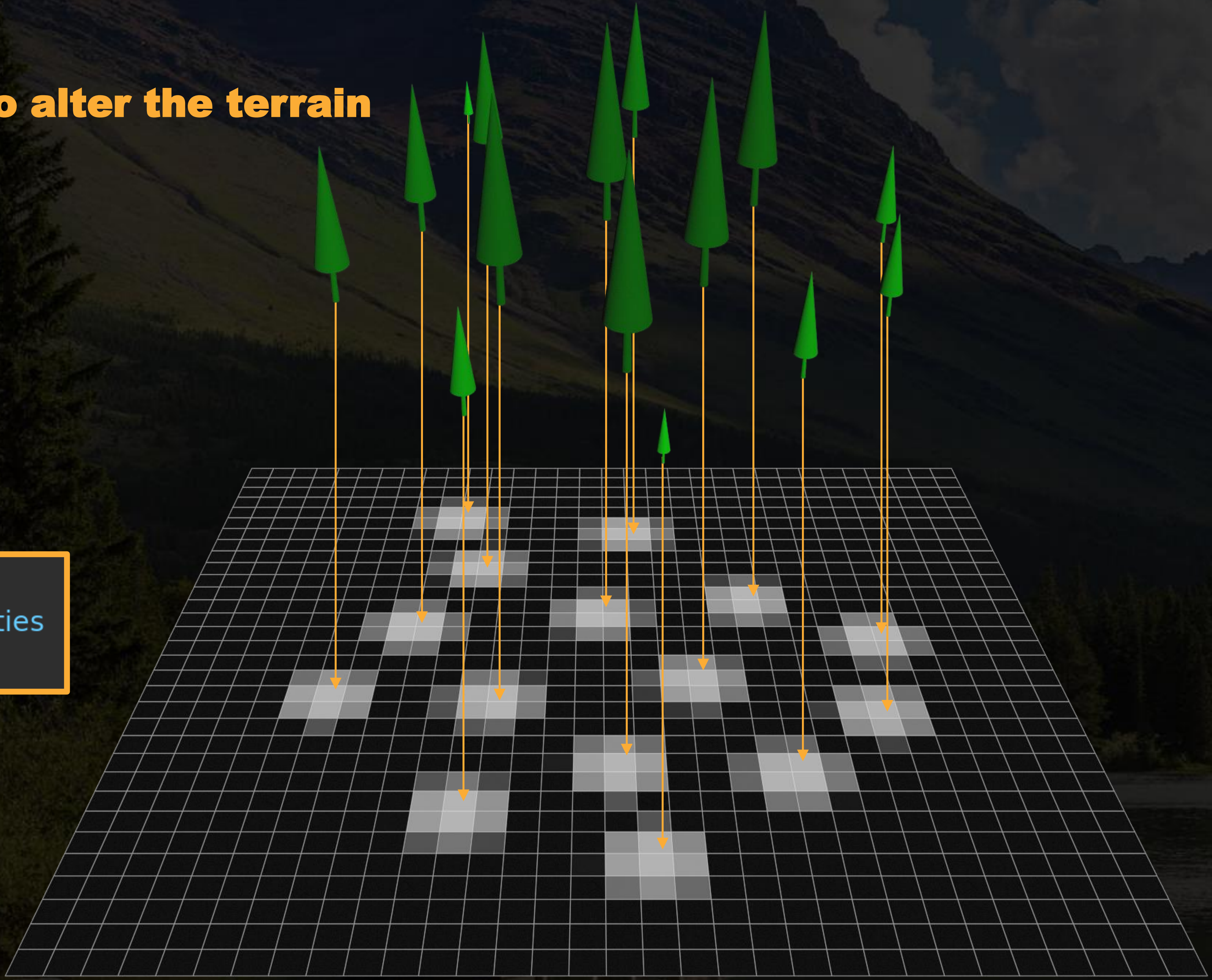


Terrain

Generated entities can also alter the terrain

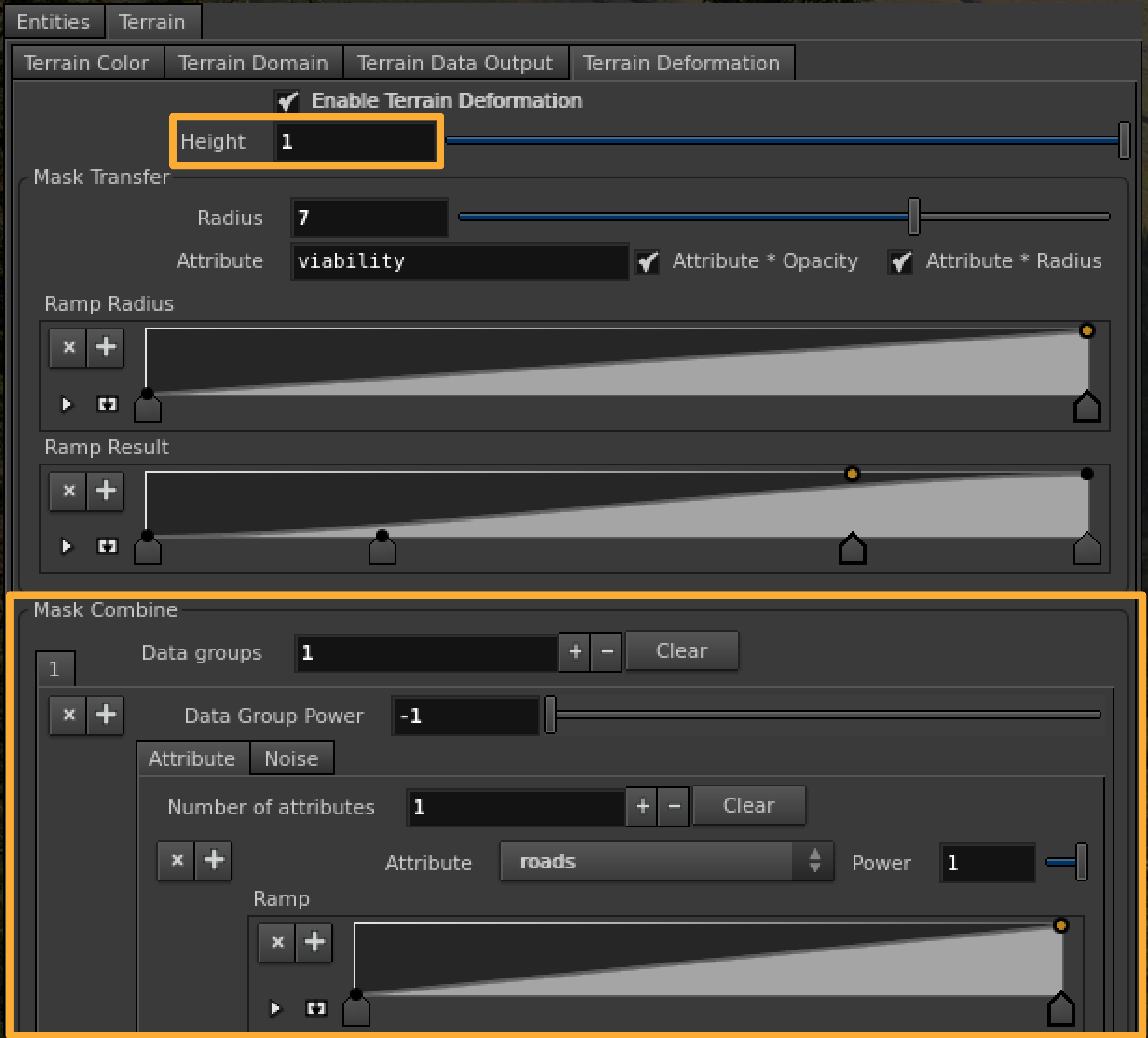
- **Terrain Deformation**
- **Terrain Textures**
- **Terrain Data Output**
- **Terrain Color**





Terrain Deformation

Heightmap layer



Entities | Terrain

Terrain Color | Terrain Domain | Terrain Data Output | Terrain Deformation

✓ Enable Terrain Deformation

Height: 1

Mask Transfer

Radius: 7

Attribute: viability ✓ Attribute * Opacity ✓ Attribute * Radius

Ramp Radius

Ramp Result

Mask Combine

Data groups: 1 + − Clear

1

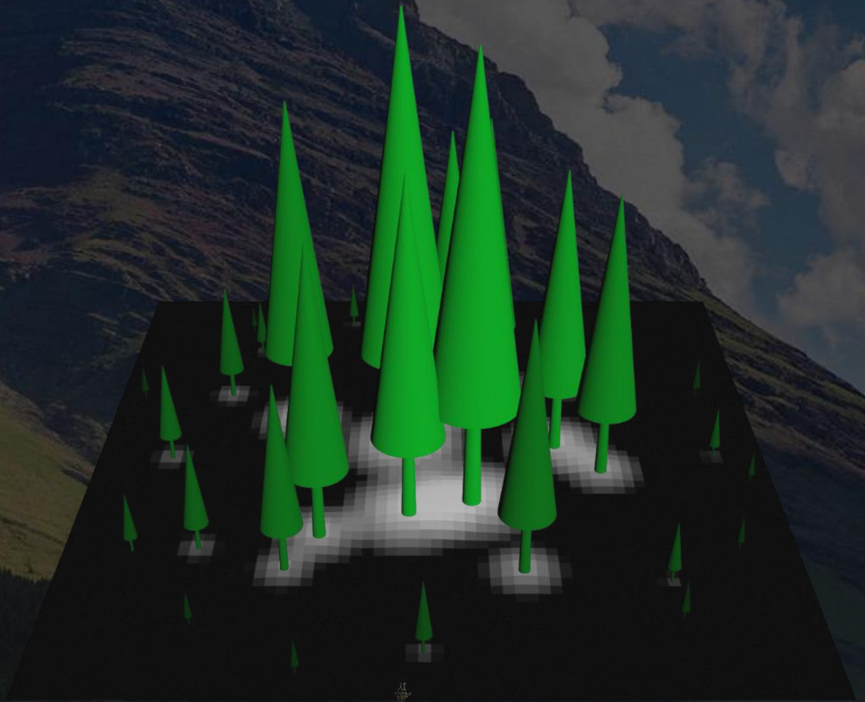
Data Group Power: -1

Attribute | Noise

Number of attributes: 1 + − Clear

Attribute: roads Power: 1

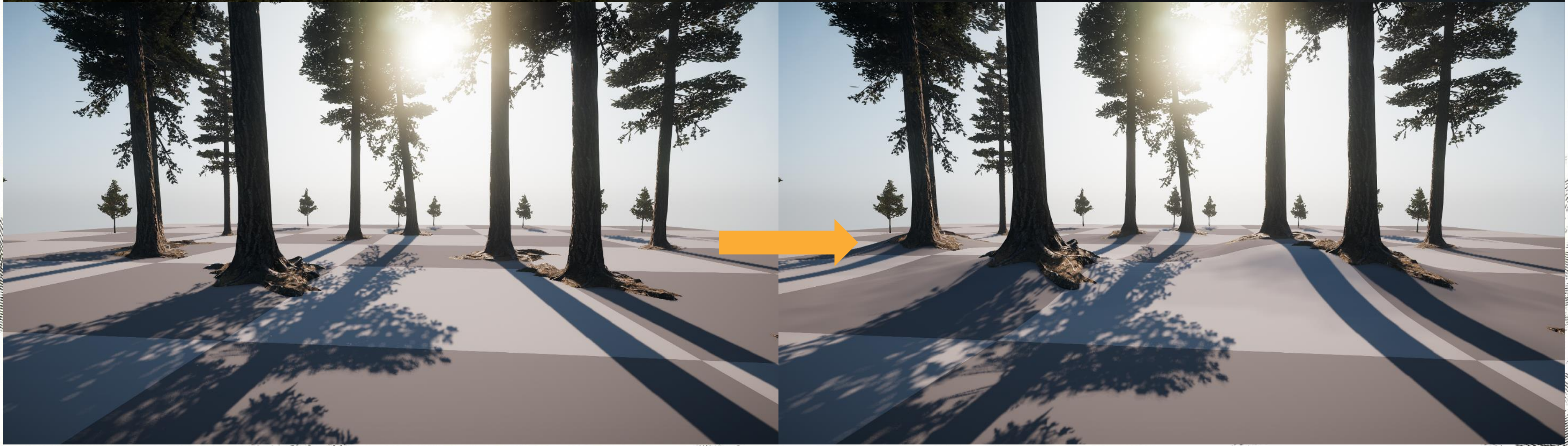
Ramp

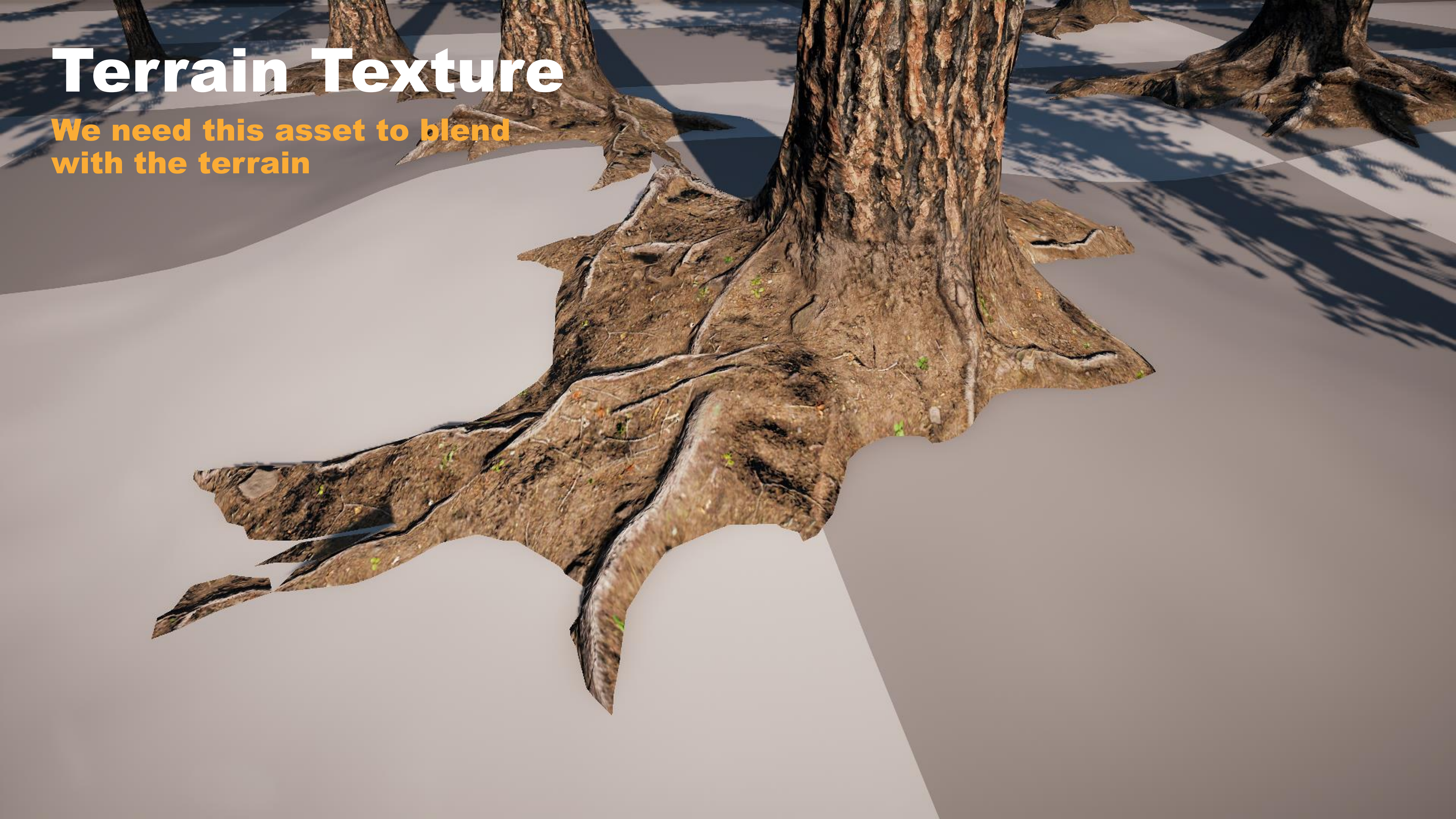


Terrain Deformation

Result in editor

In nature, tree roots are lifting the ground a little bit and are also holding the soil together which limits the erosion. This create these elevation shapes around tree trunks.



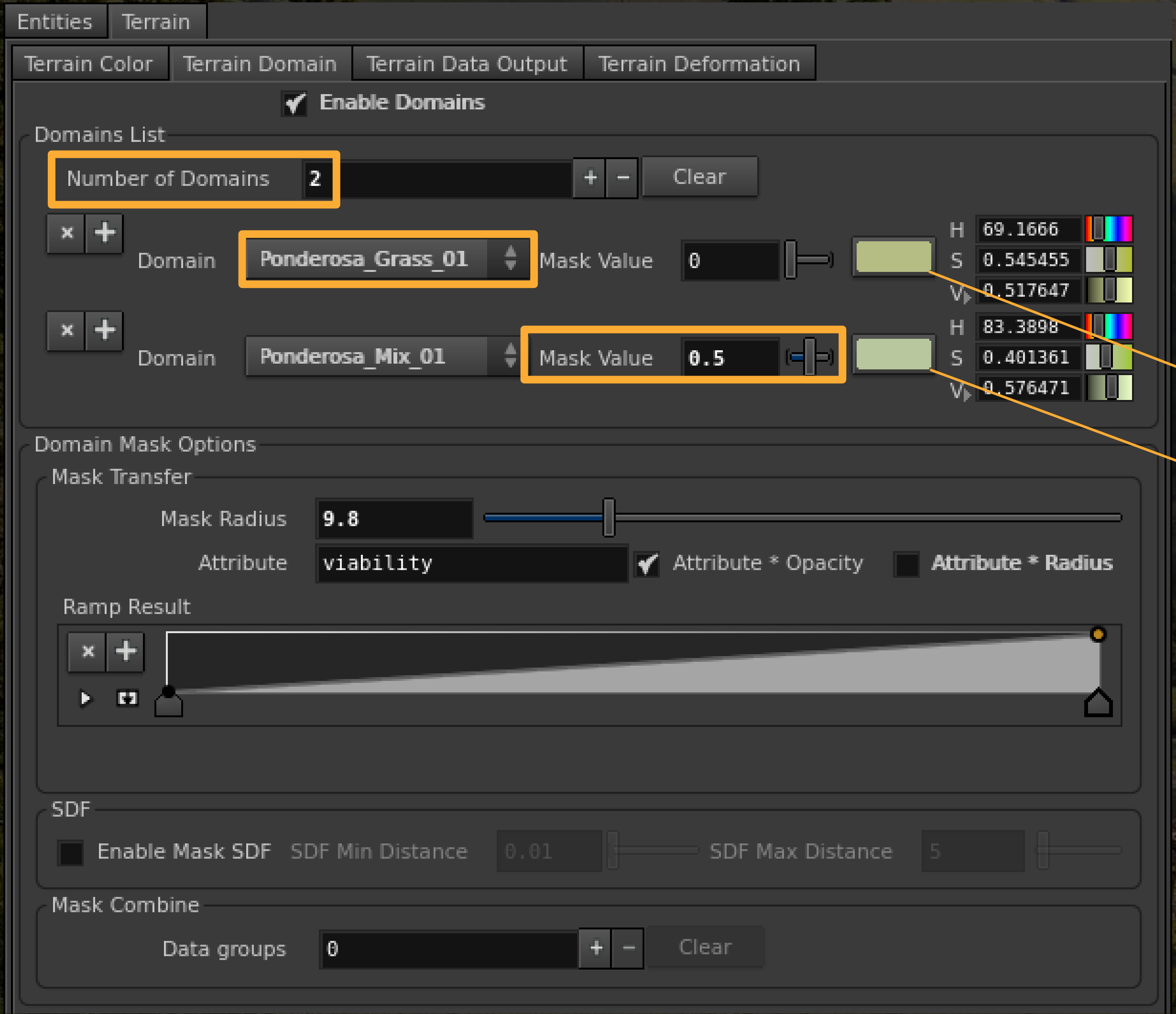


Terrain Texture

We need this asset to blend with the terrain

Terrain Texture

Generating terrain textures IDs from mask



Entities | Terrain

Terrain Color | Terrain Domain | Terrain Data Output | Terrain Deformation

✓ Enable Domains

Domains List

Number of Domains 2 | + | − | Clear

Domain Ponderosa_Grass_01 | Mask Value 0 | H 69.1666 | S 0.545455 | V 0.517647

Domain Ponderosa_Mix_01 | Mask Value 0.5 | H 83.3898 | S 0.401361 | V 0.576471

Domain Mask Options

Mask Transfer

Mask Radius 9.8

Attribute viability | ✓ Attribute * Opacity | Attribute * Radius

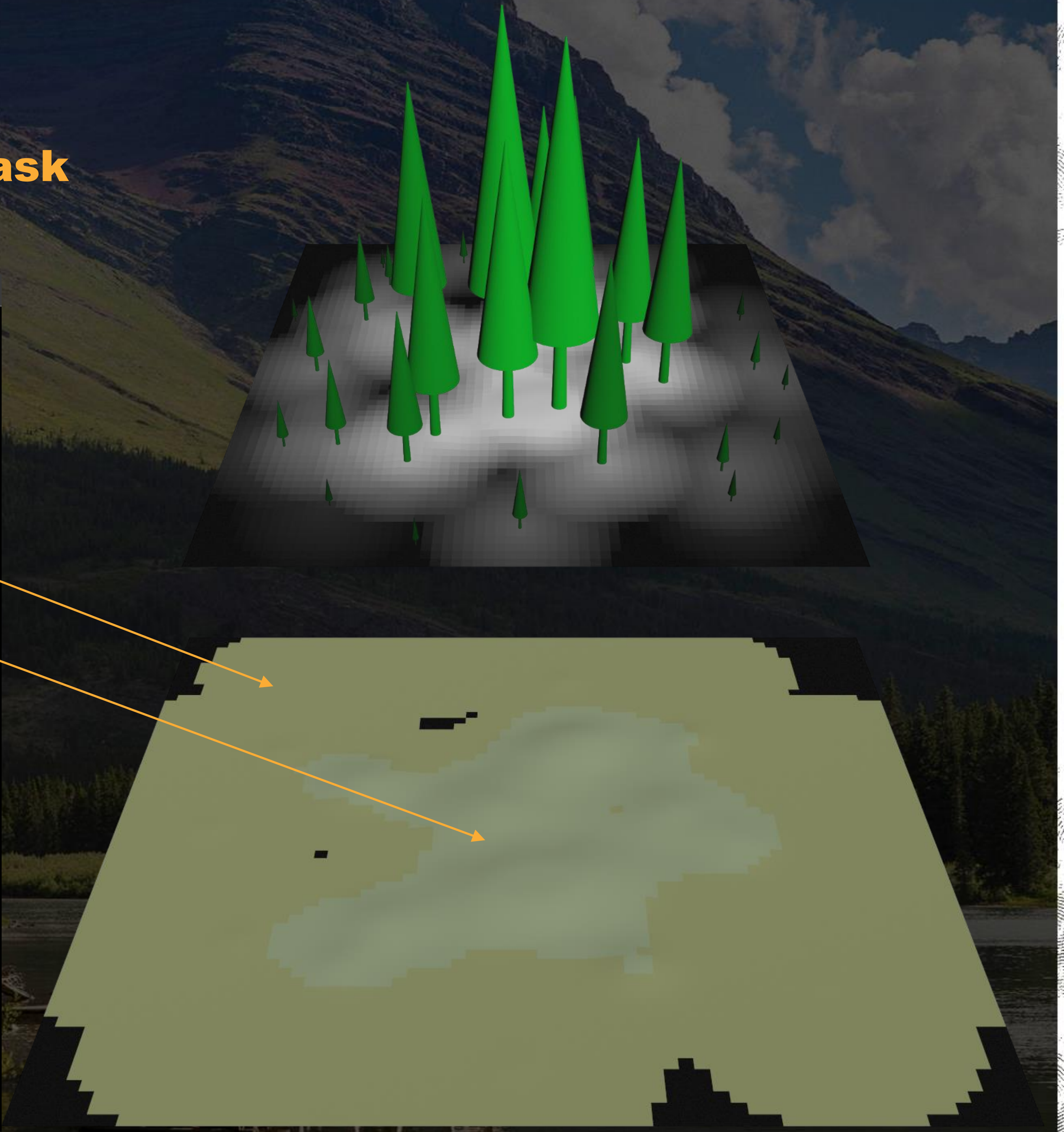
Ramp Result

SDF

Enable Mask SDF | SDF Min Distance 0.01 | SDF Max Distance 5

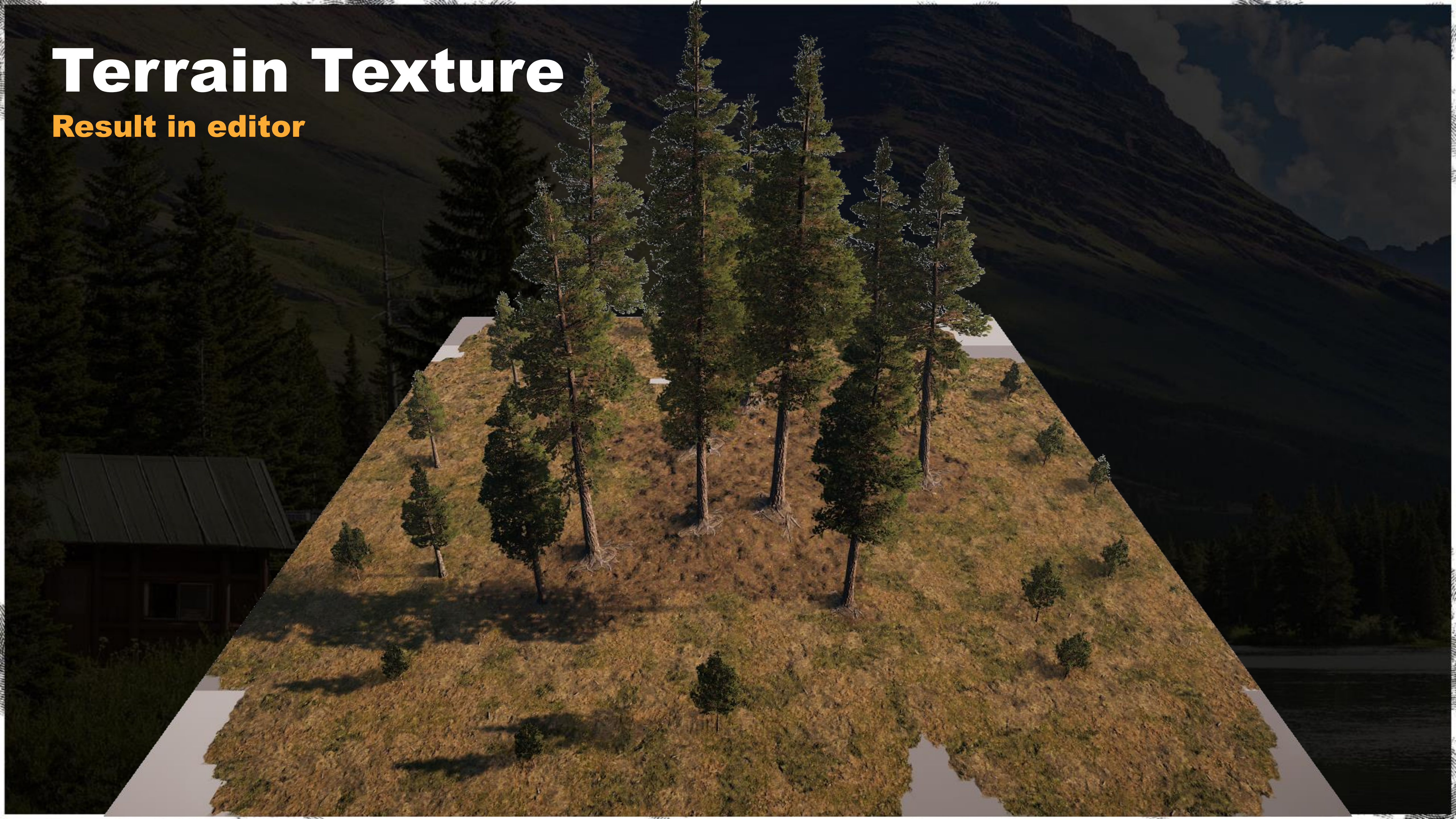
Mask Combine

Data groups 0 | + | − | Clear



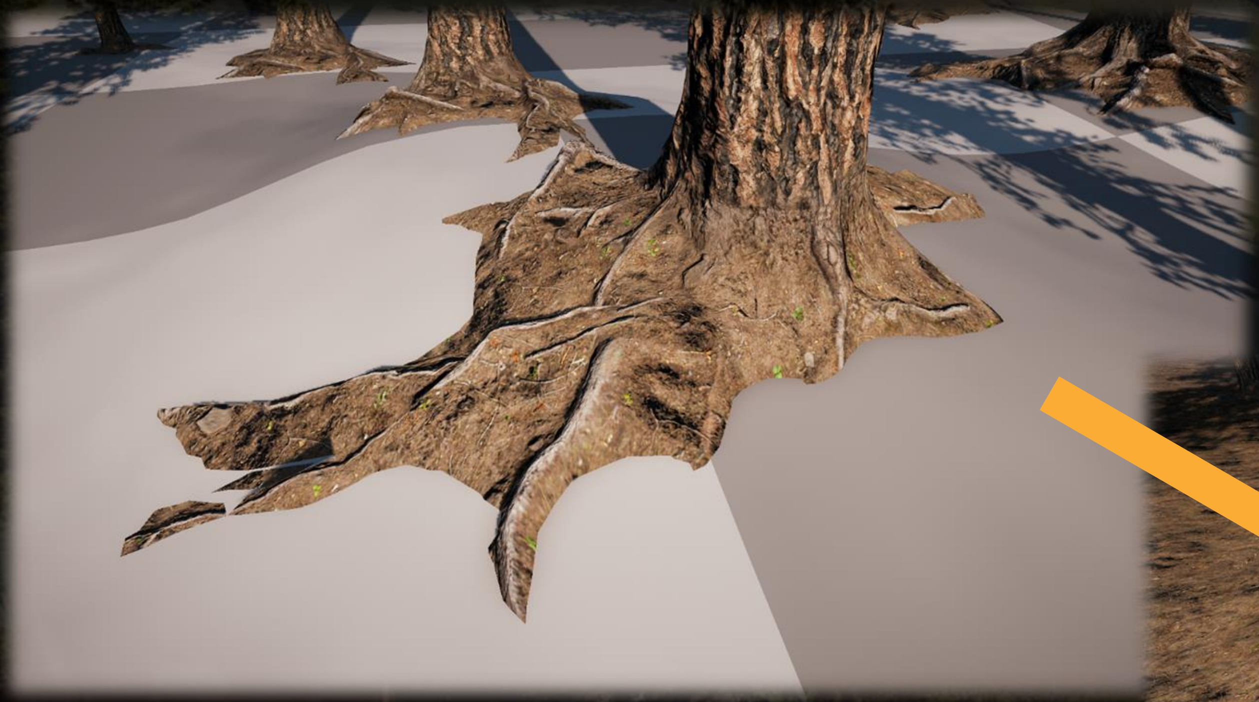
Terrain Texture

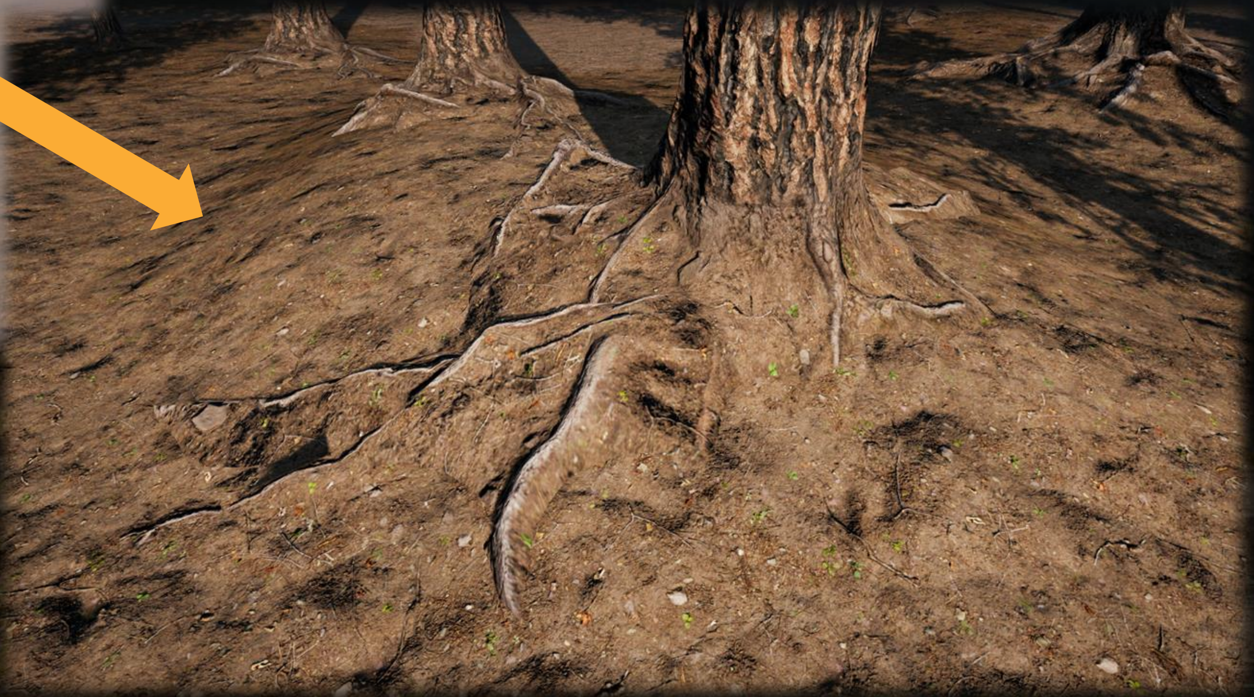
Result in editor



Terrain Texture

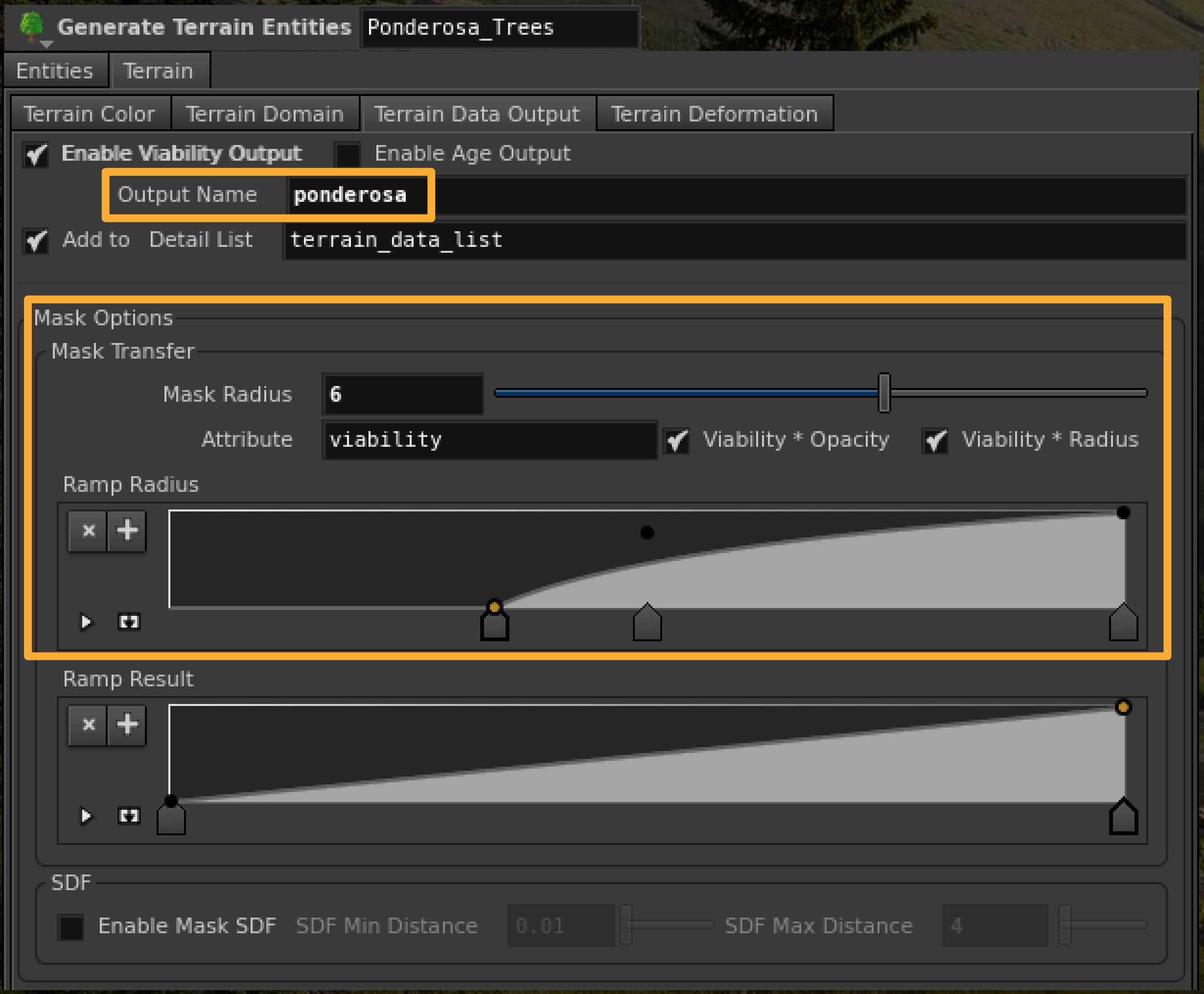
Result in editor





Terrain Data Output

Generating a new terrain data mask from entities



Generate Terrain Entities Ponderosa_Trees

Entities | Terrain

Terrain Color | Terrain Domain | Terrain Data Output | Terrain Deformation

✓ Enable Viability Output ☐ Enable Age Output

Output Name: ponderosa

✓ Add to Detail List: terrain_data_list

Mask Options

Mask Transfer

Mask Radius: 6

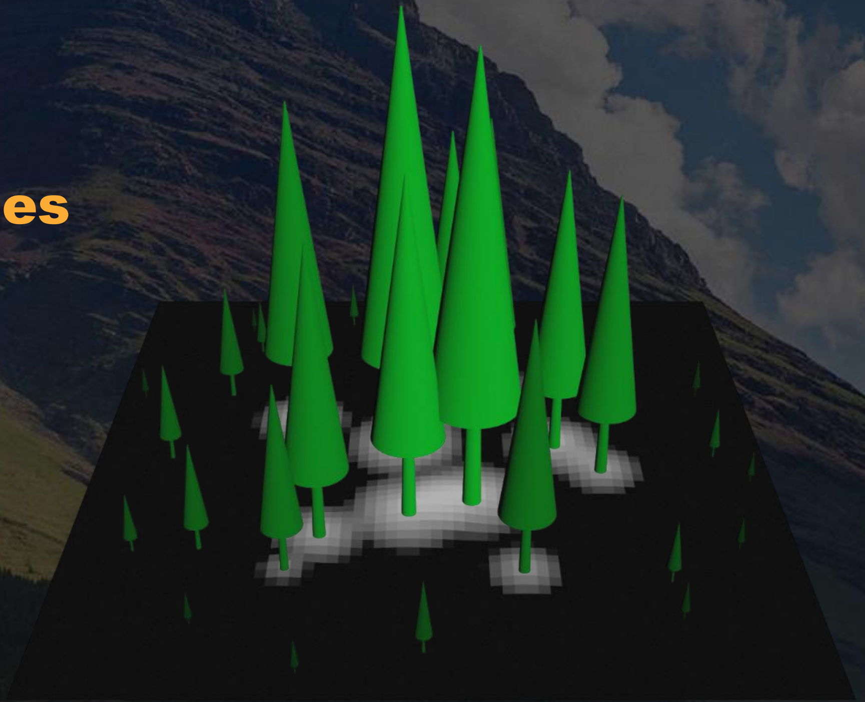
Attribute: viability ✓ Viability * Opacity ✓ Viability * Radius

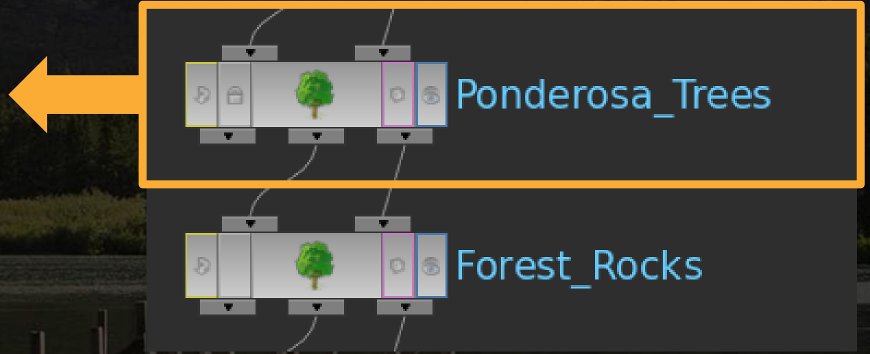
Ramp Radius

Ramp Result

SDF

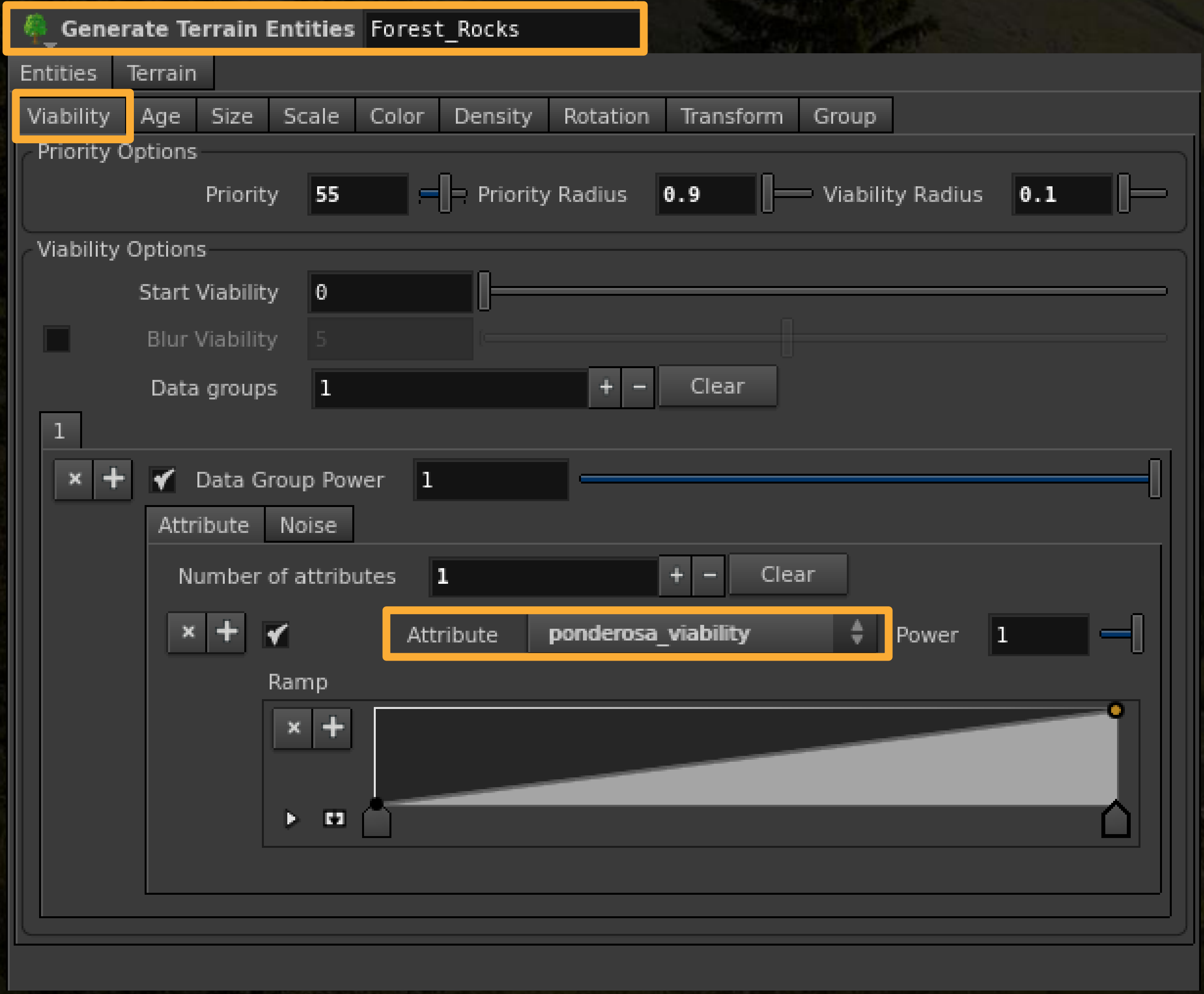
☐ Enable Mask SDF SDF Min Distance 0.01 SDF Max Distance 4





Terrain Data Output

Use new terrain data as viability on a following species



Generate Terrain Entities: Forest_Rocks

Entities | Terrain

Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

Priority Options

Priority 55 — Priority Radius 0.9 — Viability Radius 0.1

Viability Options

Start Viability 0

Blur Viability 5

Data groups 1 + − Clear

1

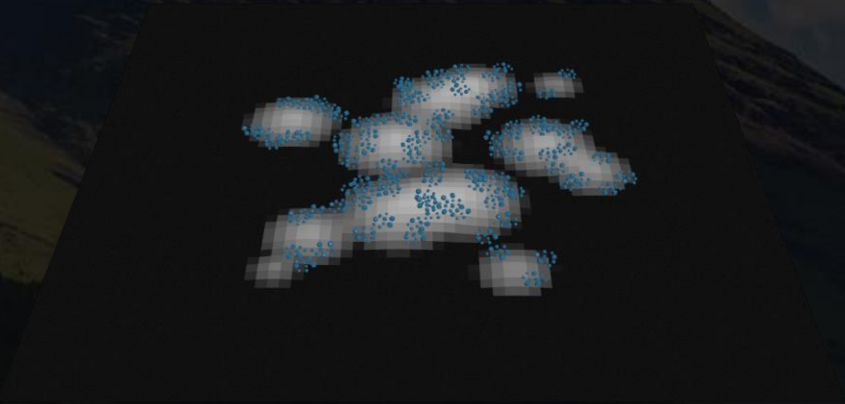
Data Group Power 1

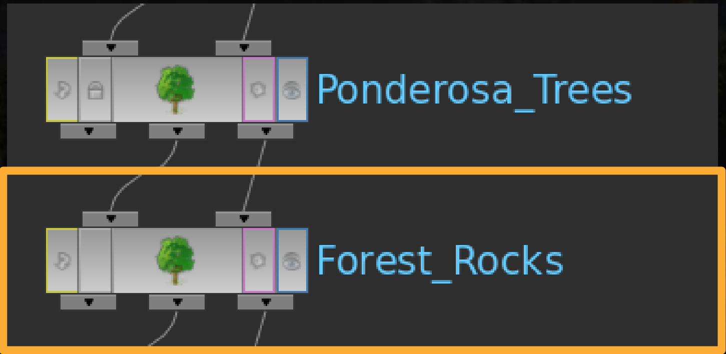
Attribute | Noise

Number of attributes 1 + − Clear

Attribute: ponderosa_viability — Power 1

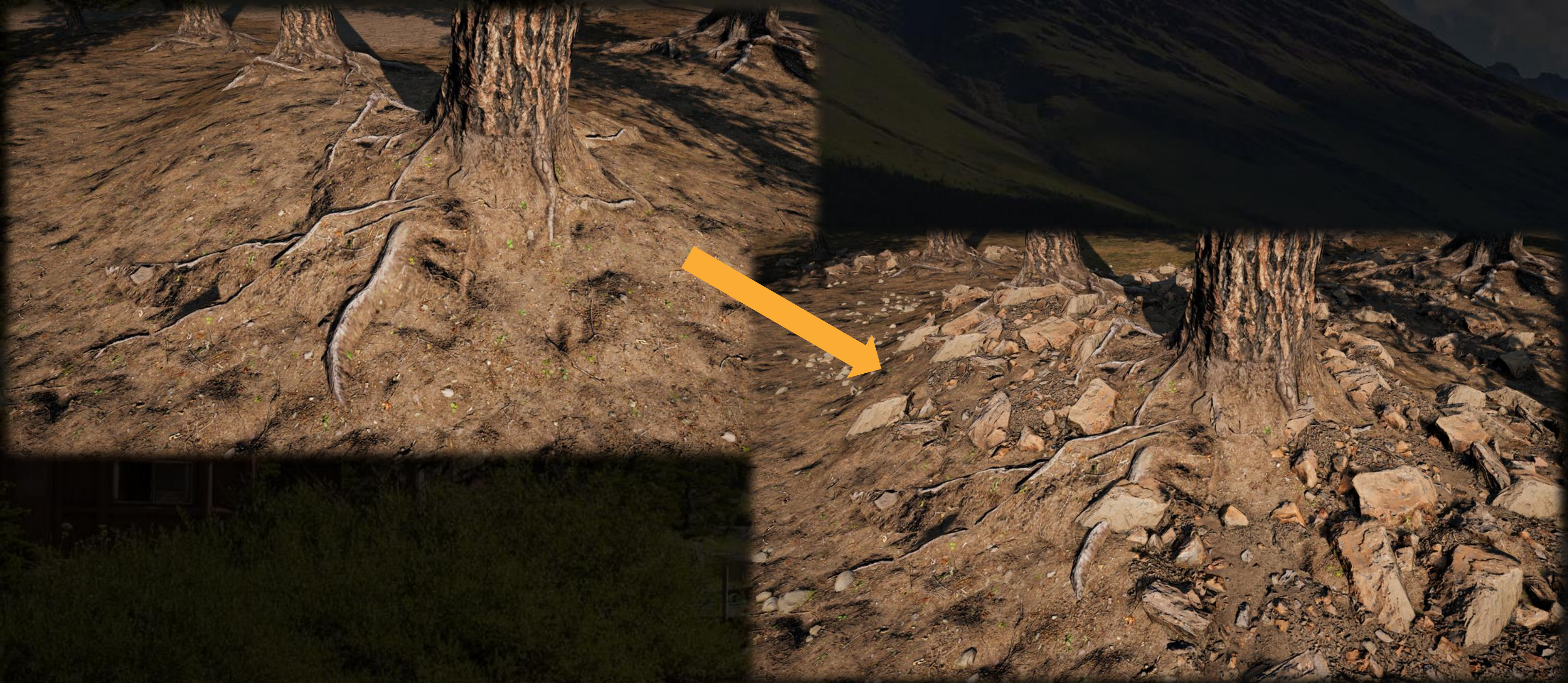
Ramp





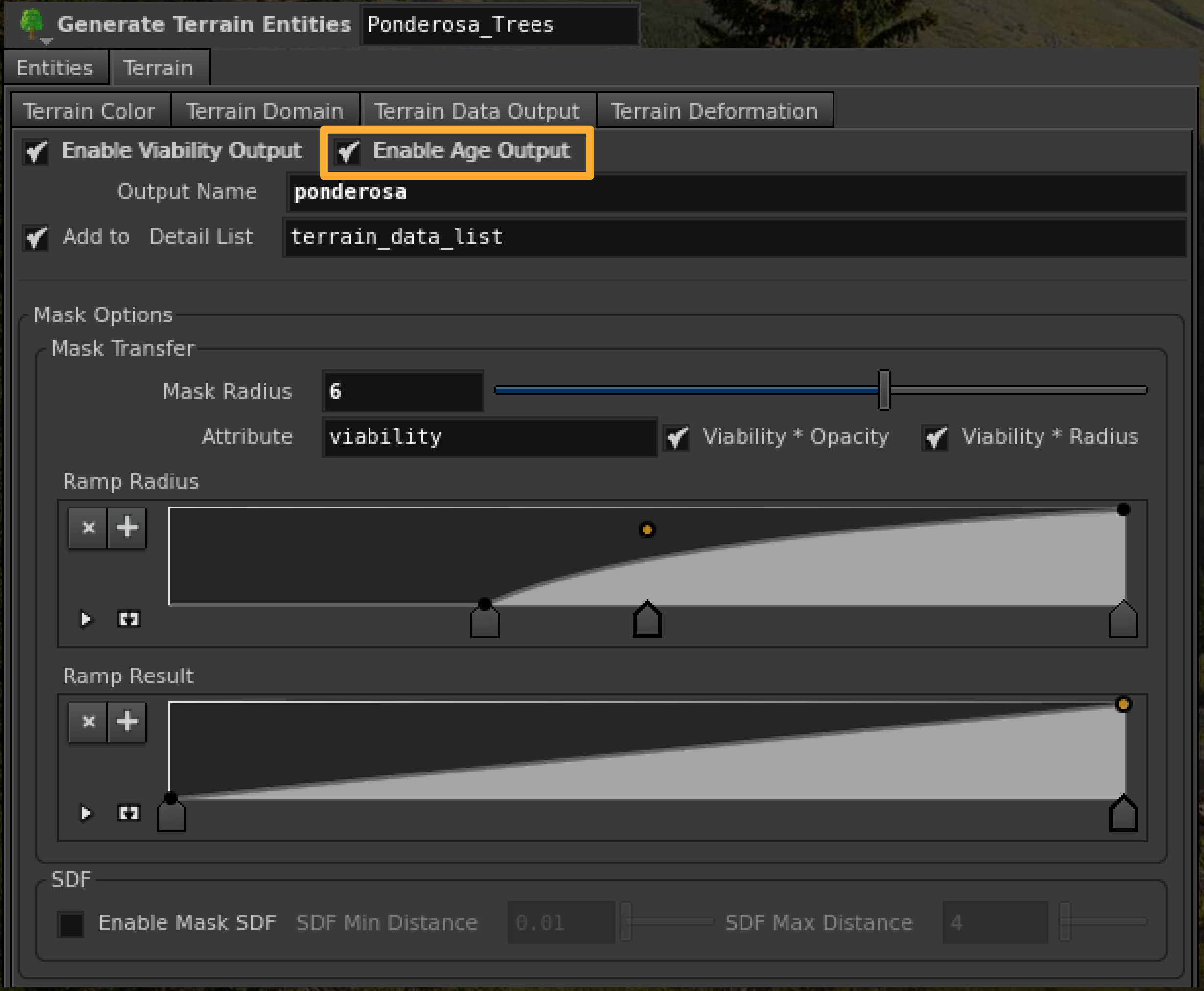
Terrain Data Output

Result in editor



Terrain Data Output

Species age output



Generate Terrain Entities Ponderosa_Trees

Entities | Terrain

Terrain Color | Terrain Domain | Terrain Data Output | Terrain Deformation

✓ Enable Viability Output ✓ Enable Age Output

Output Name ponderosa

✓ Add to Detail List terrain_data_list

Mask Options

Mask Transfer

Mask Radius 6

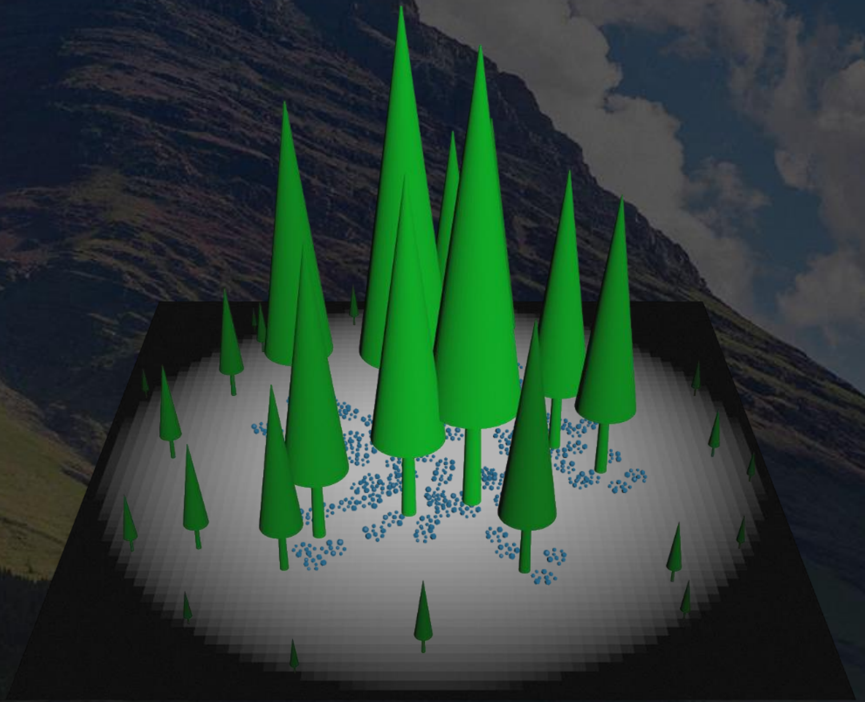
Attribute viability ✓ Viability * Opacity ✓ Viability * Radius

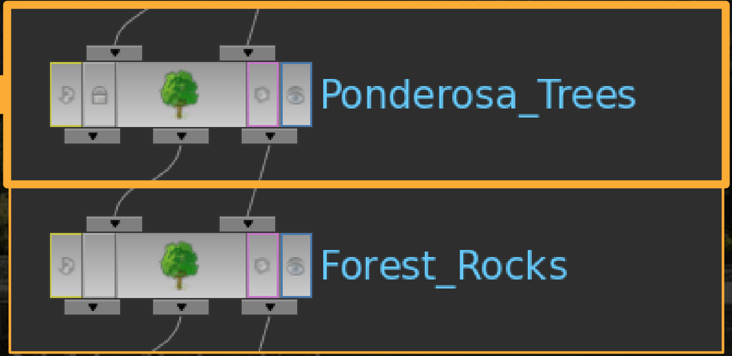
Ramp Radius

Ramp Result

SDF

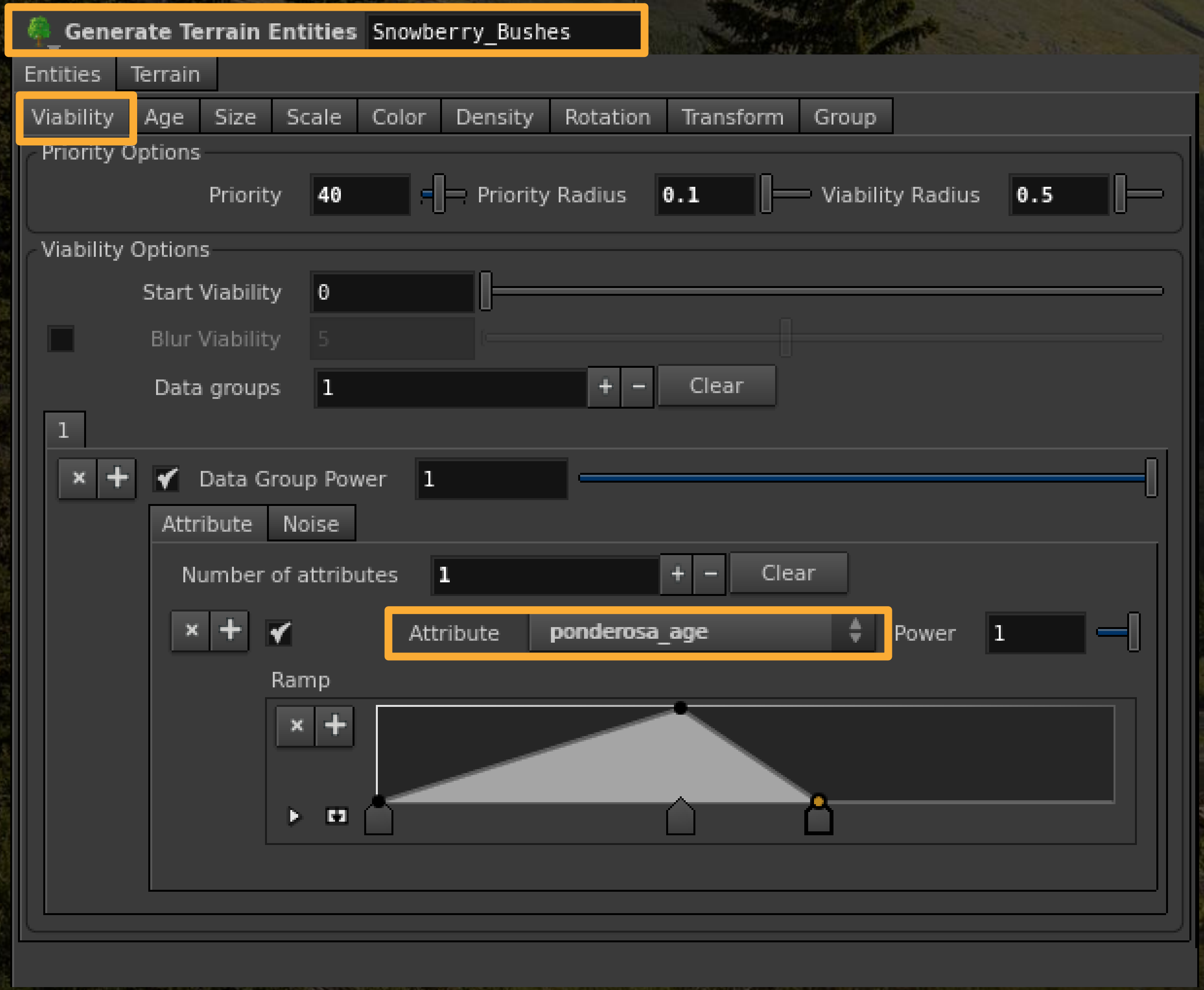
Enable Mask SDF SDF Min Distance 0.01 SDF Max Distance 4





Terrain Data Output

Species age output



Generate Terrain Entities Snowberry_Bushes

Entities | Terrain

Viability | Age | Size | Scale | Color | Density | Rotation | Transform | Group

Priority Options

Priority 40 Priority Radius 0.1 Viability Radius 0.5

Viability Options

Start Viability 0

Blur Viability 5

Data groups 1

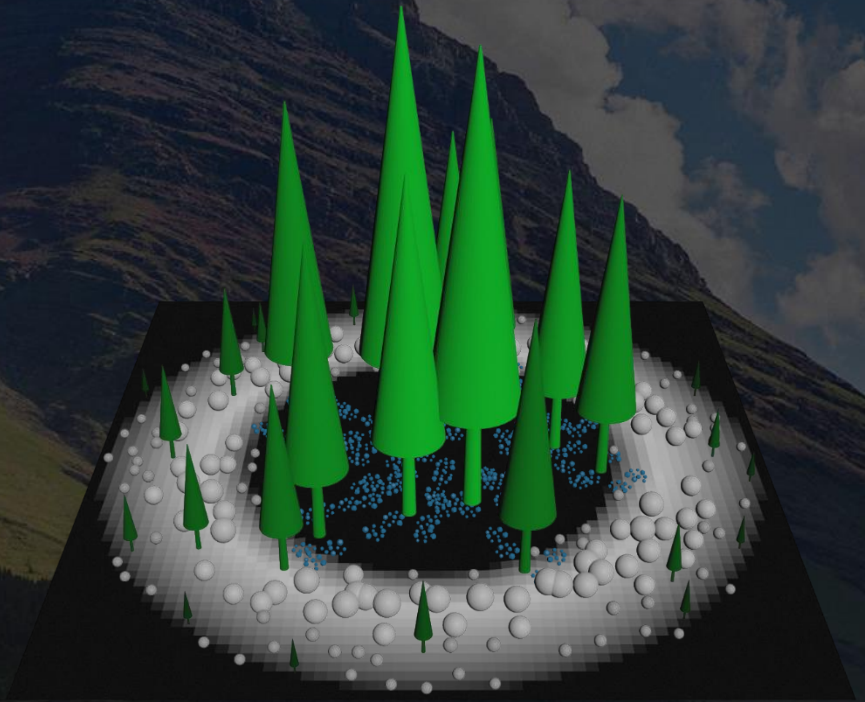
Data Group Power 1

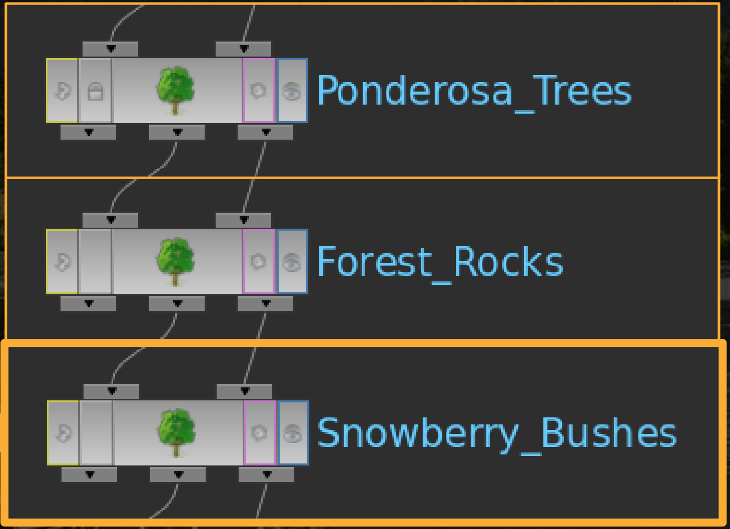
Attribute | Noise

Number of attributes 1

Attribute ponderosa_age Power 1

Ramp

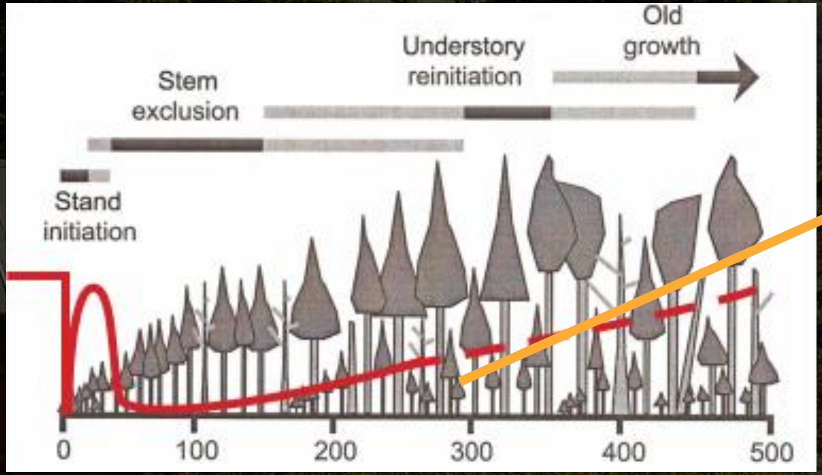


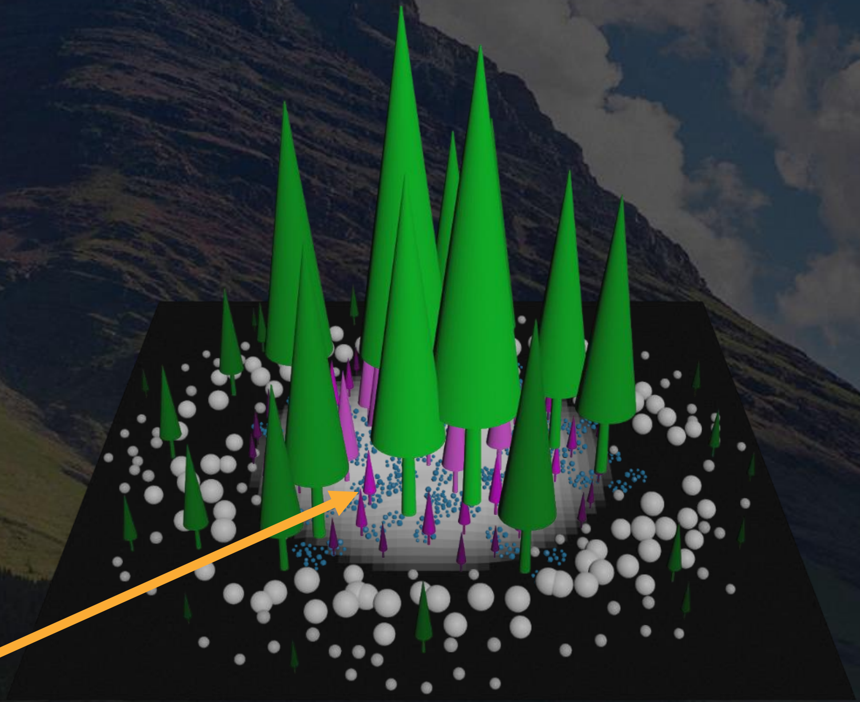


Terrain Data Output

Species age output

- **Using the age of our main tree species we can create the ecological succession young regrowth effect at the desired target distance from the border of the forest.**

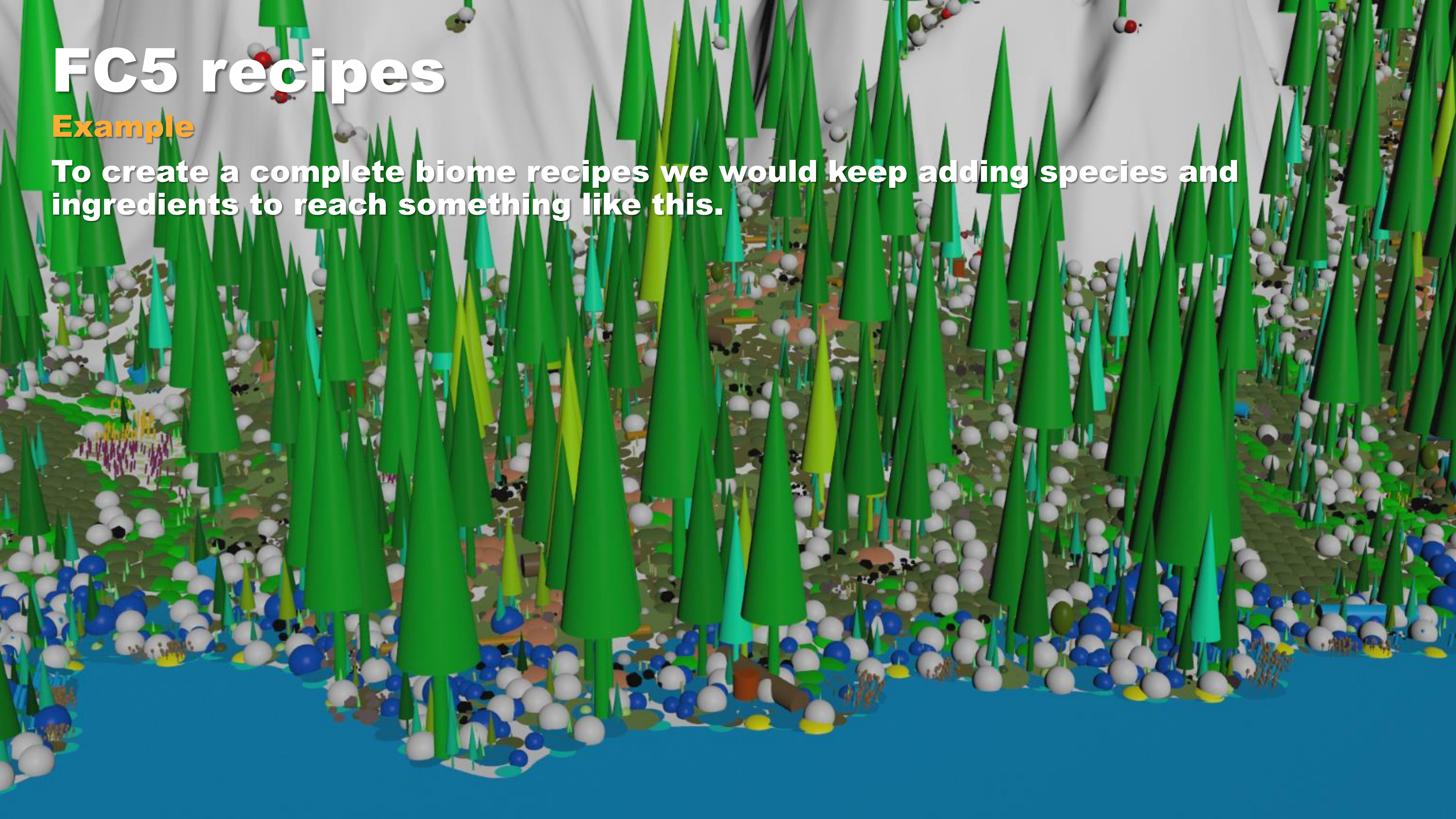




FC5 recipes

Example

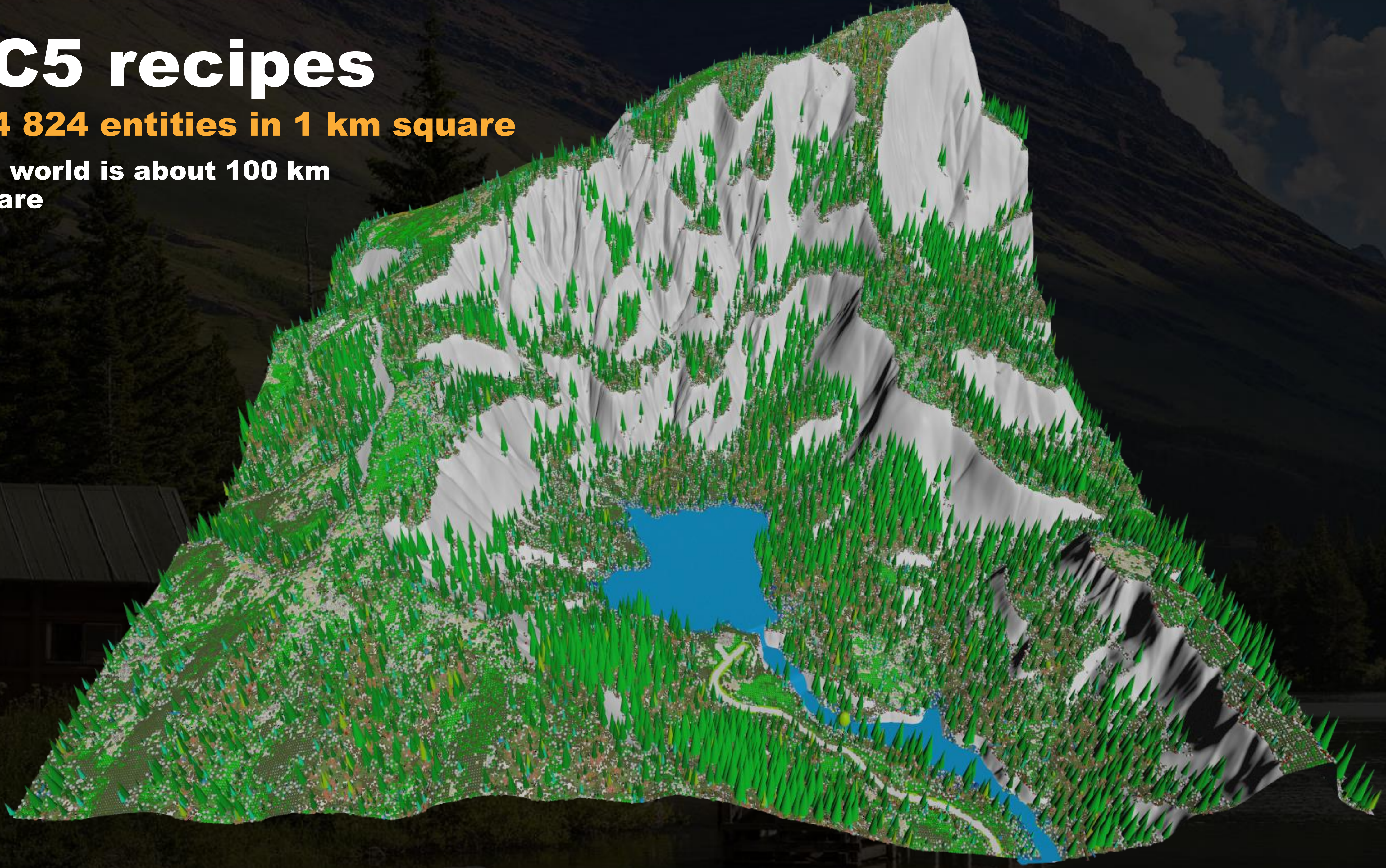
To create a complete biome recipes we would keep adding species and ingredients to reach something like this.



FC5 recipes

604 824 entities in 1 km square

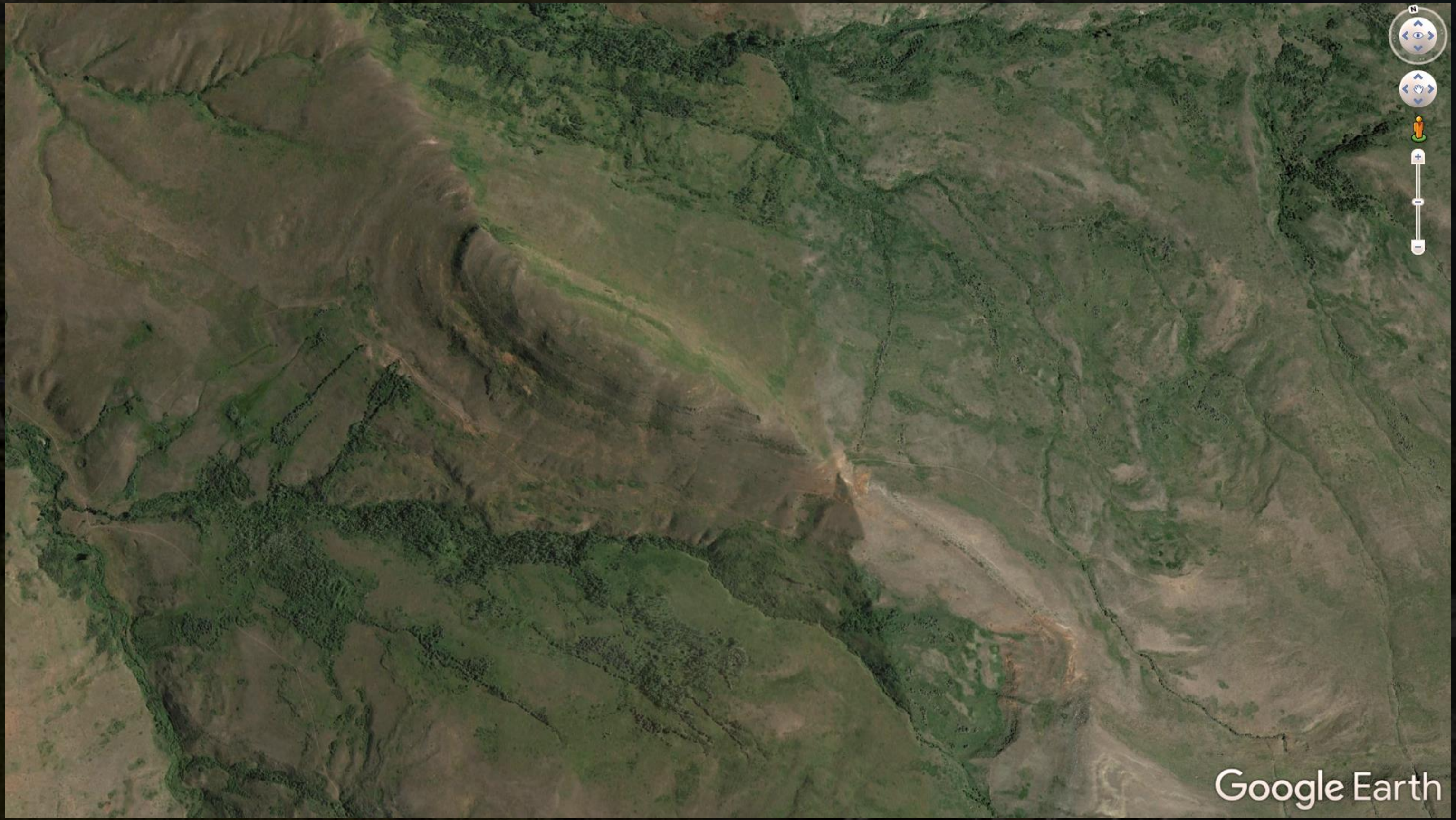
FC5 world is about 100 km square



Terrain Color

Observing nature

The humidity of the soil is observable from afar. It creates colorful patterns through the vegetation and the dry soil.

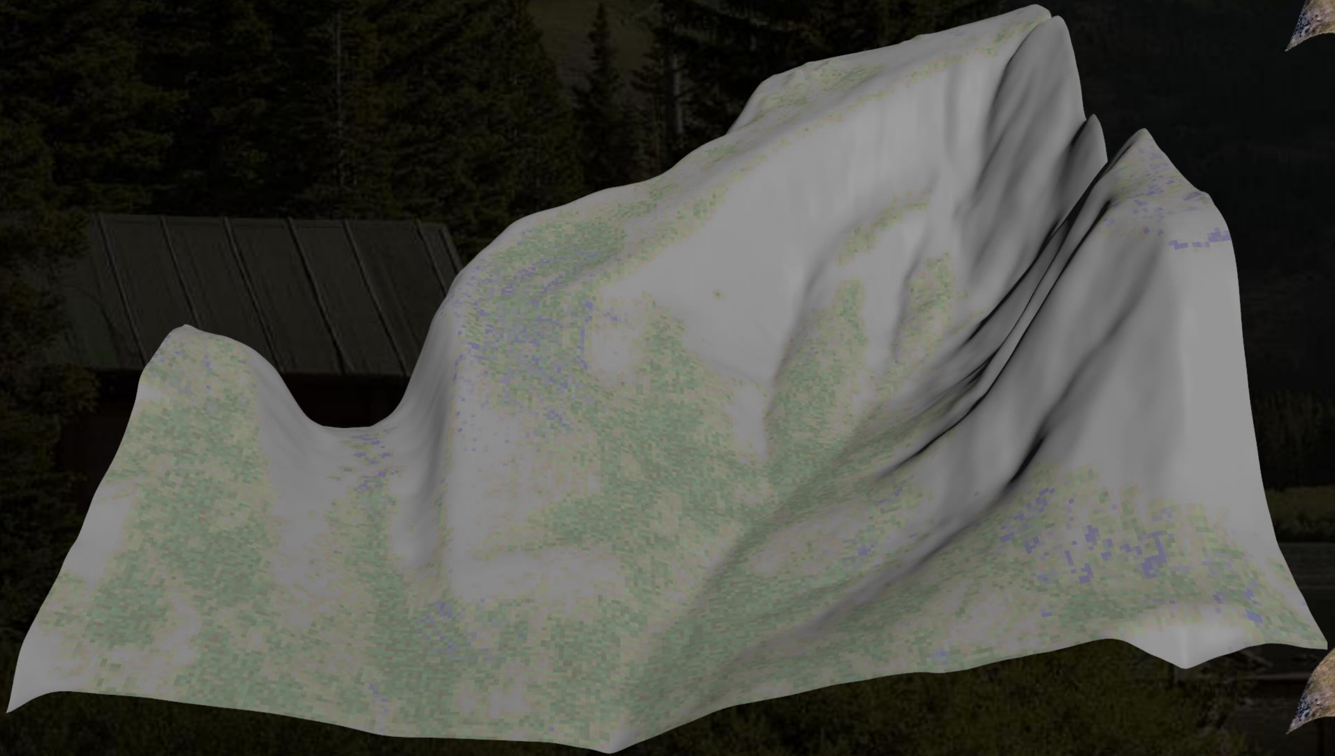


Terrain Color

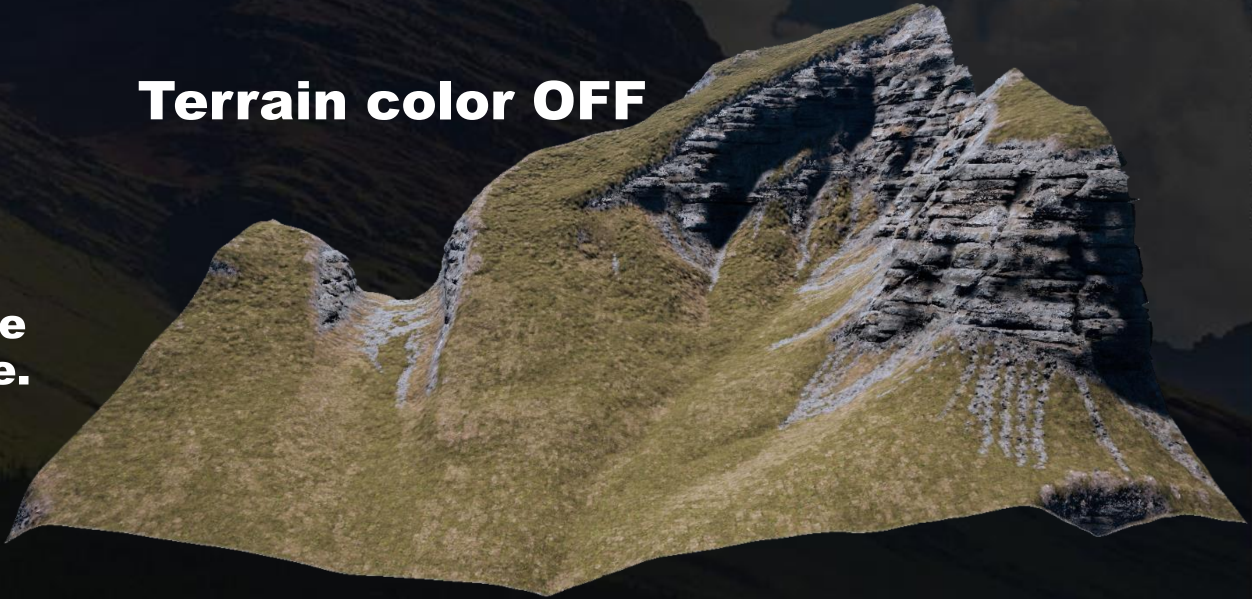
Terrain texture tint

We generate color variation on the terrain from our abiotic terrain data sets.

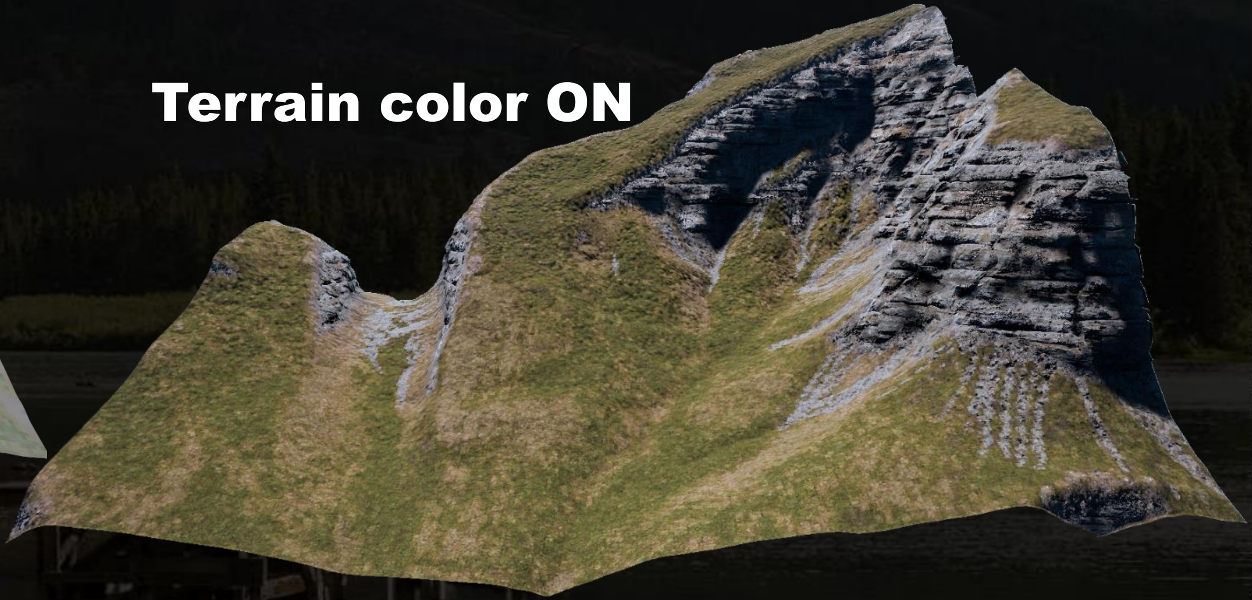
This gives us further variations and at the same time limit the amount of textures we need to use in game.



Terrain color OFF

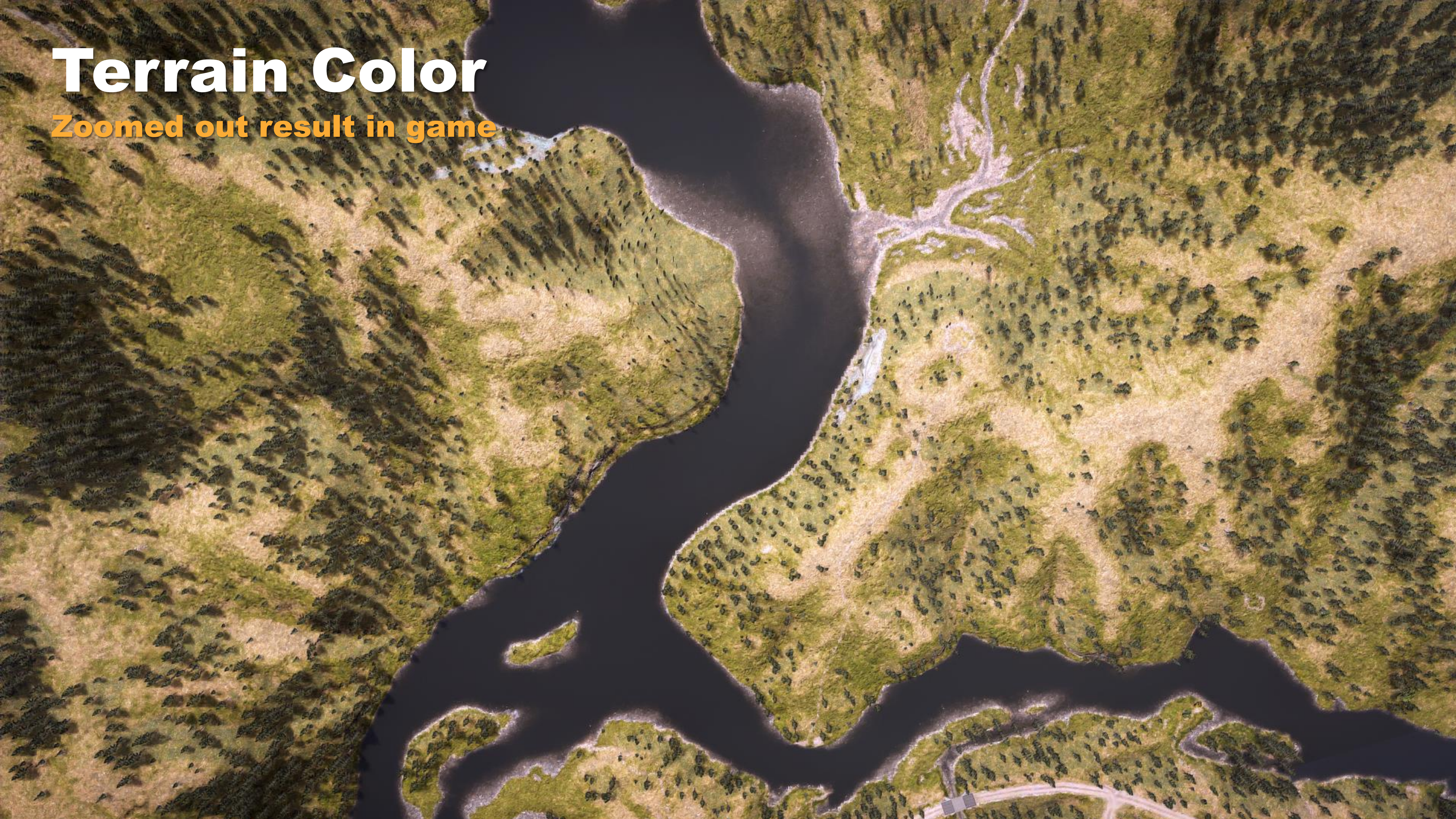


Terrain color ON



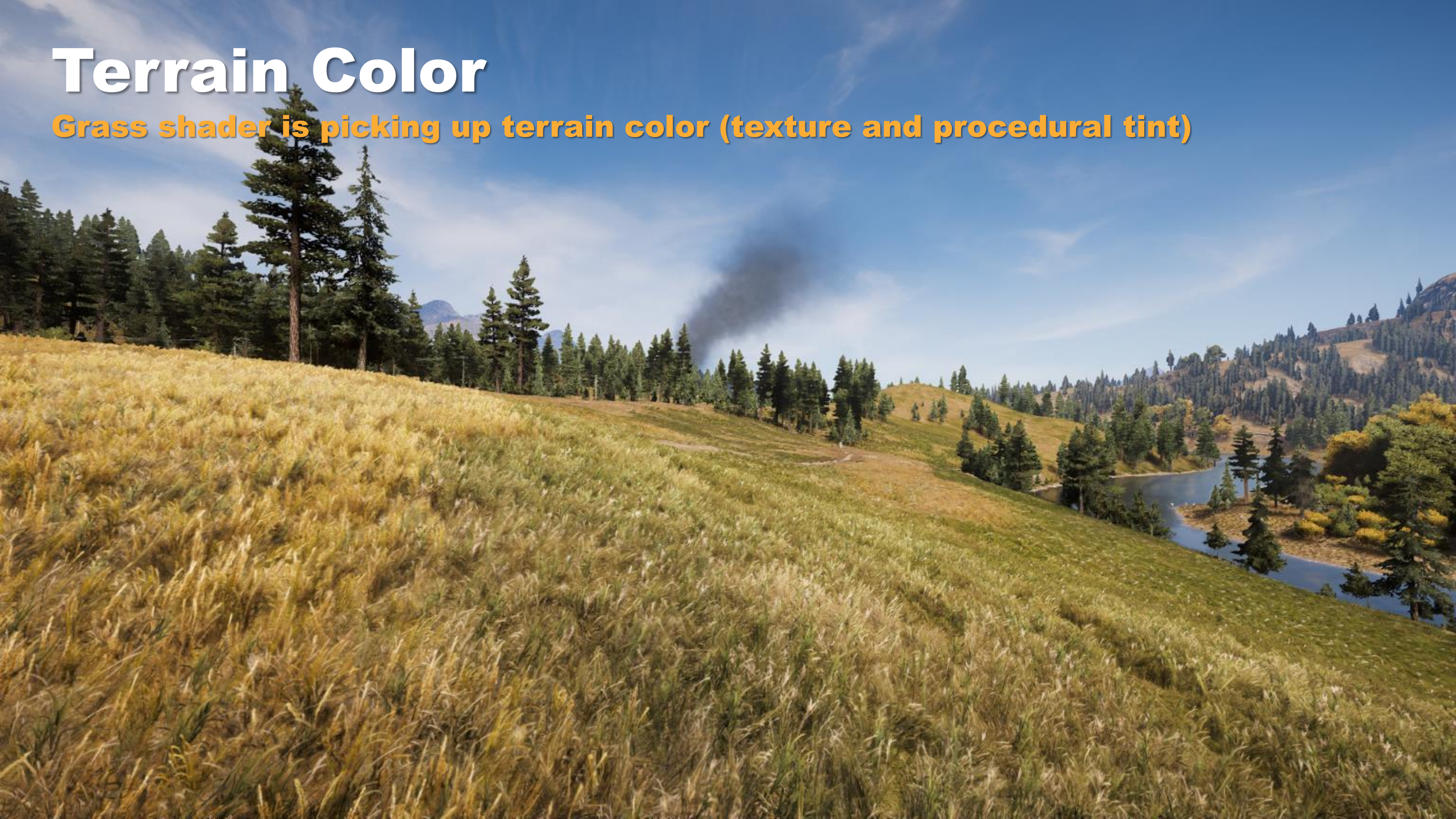
Terrain Color

Zoomed out result in game



Terrain Color

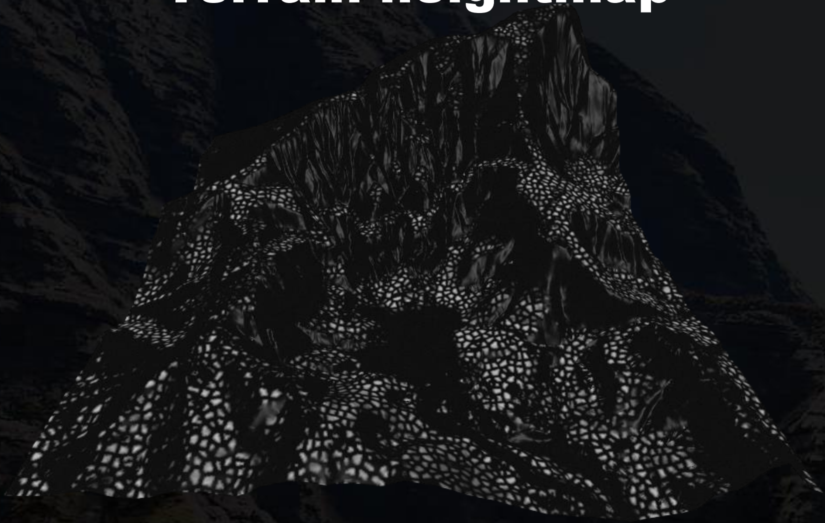
Grass shader is picking up terrain color (texture and procedural tint)



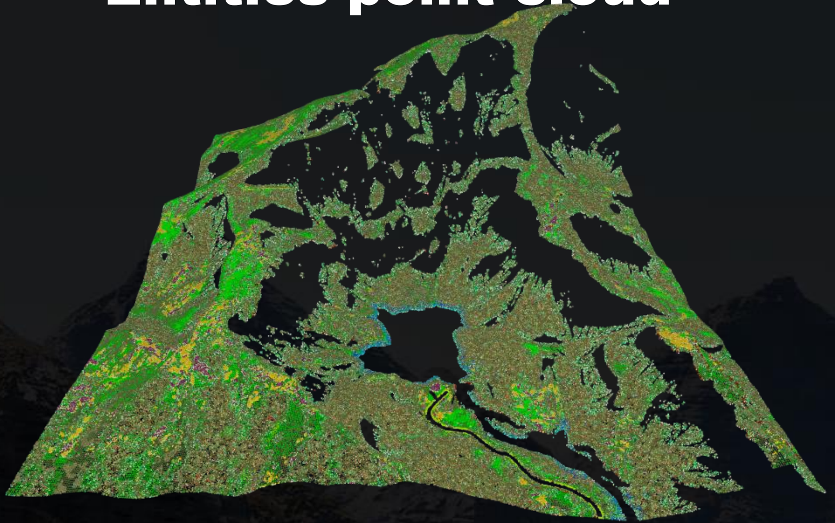
Export

Data exported to editor

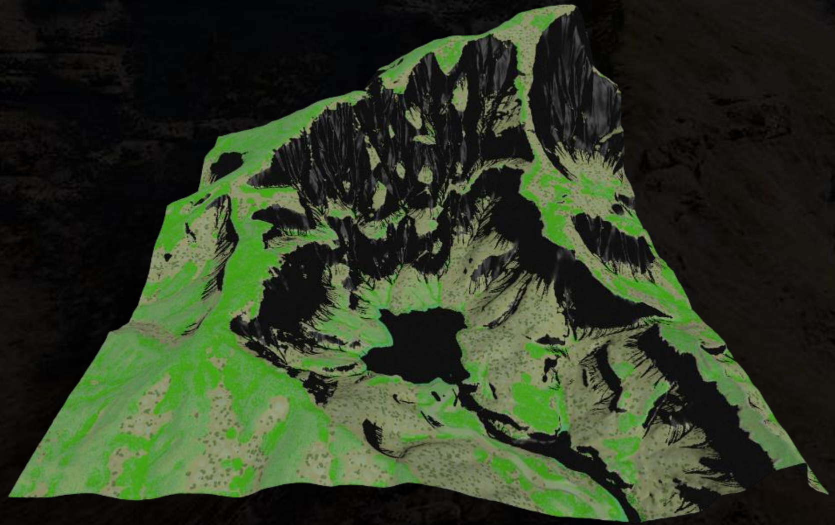
Terrain heightmap



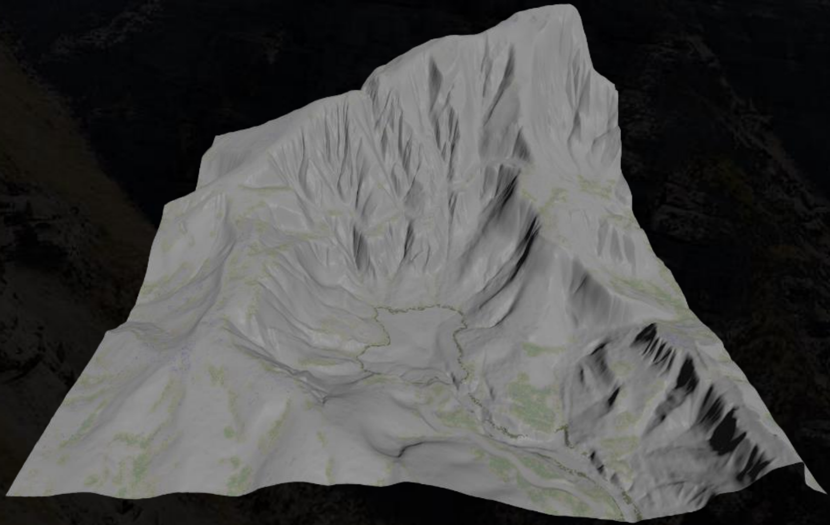
Entities point cloud



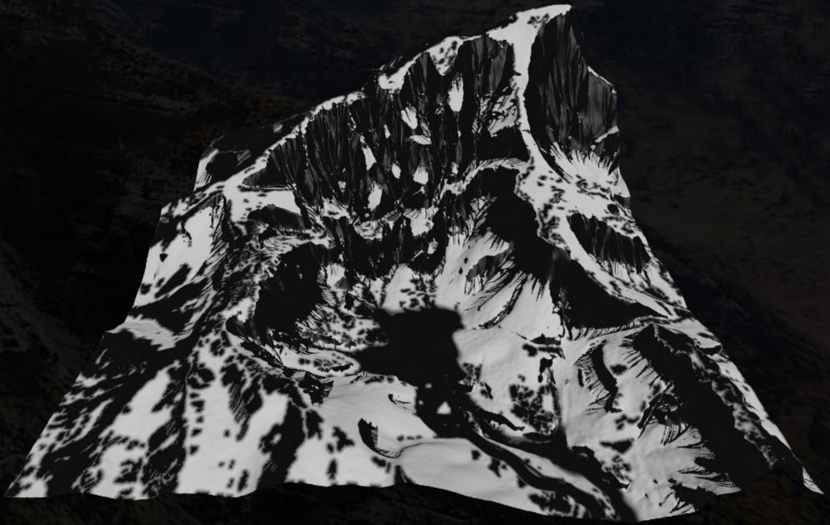
Terrain Texture IDs



Terrain color



Forest mask



8- CHANGE OF PLAN

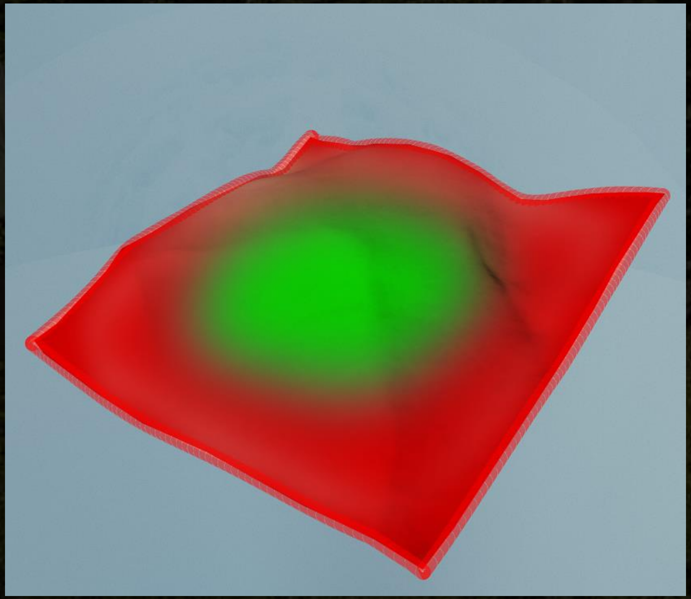
Biome Painter

Gradient vs Boolean

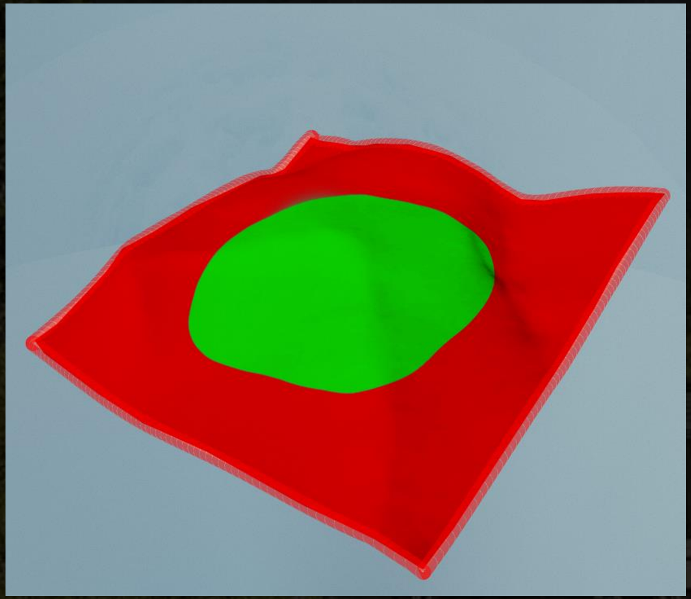
For the biomes painter, initially the plan was to support gradient painting so we could blend biomes together. It is supported in the Houdini tools (because the viability will kick in)

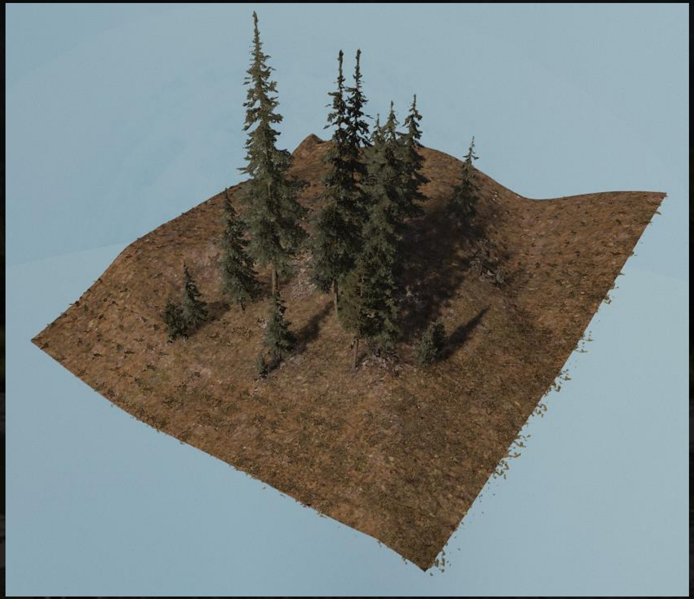
However from an editor and user point of view it was not such a good idea.

✗



✓

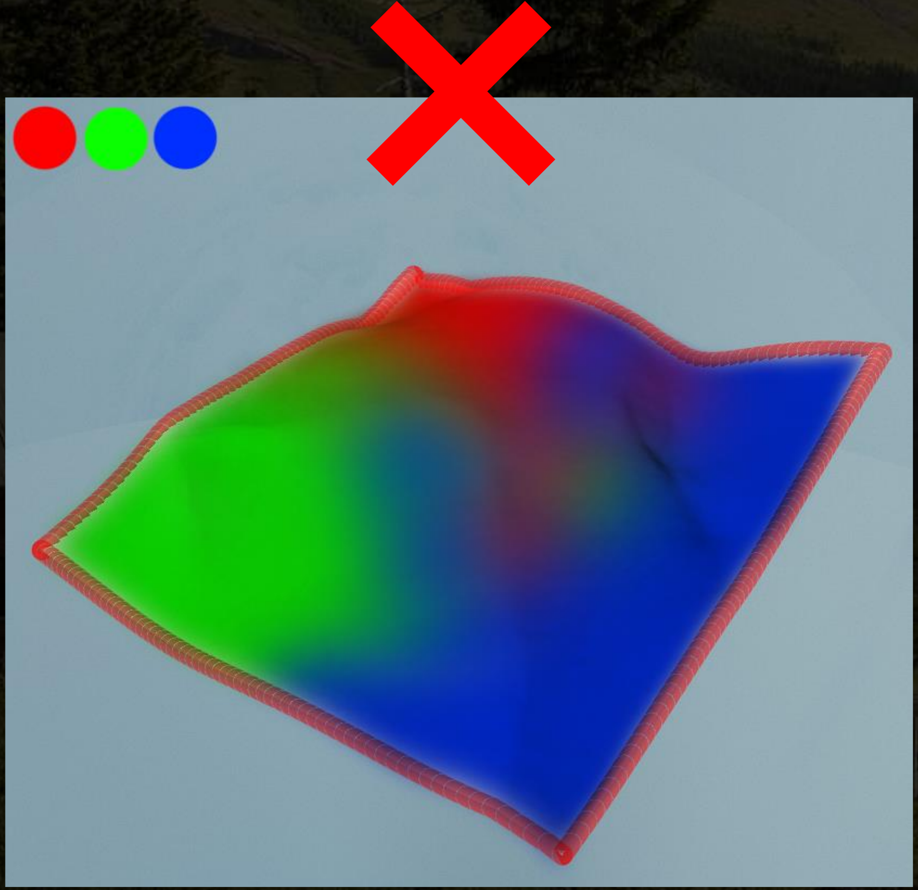


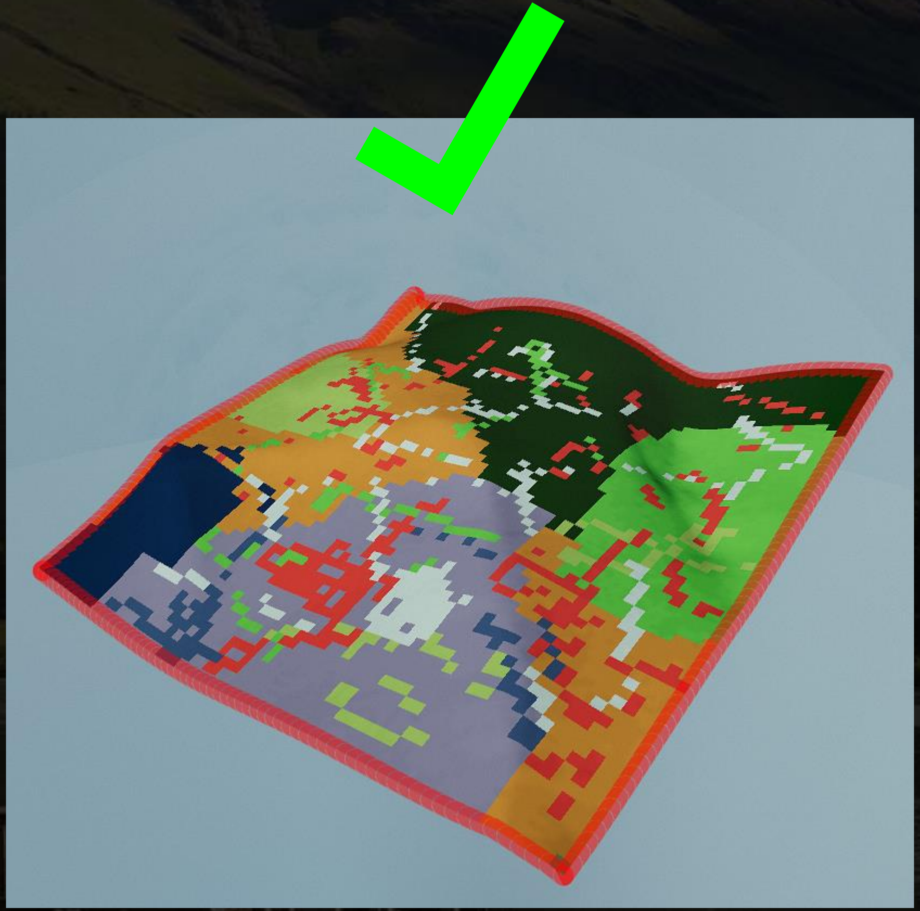


Biome Painter

Gradient vs Boolean

- Difficult to debug when several colors overlap
- Biome results were good without gradients





Terrain Data

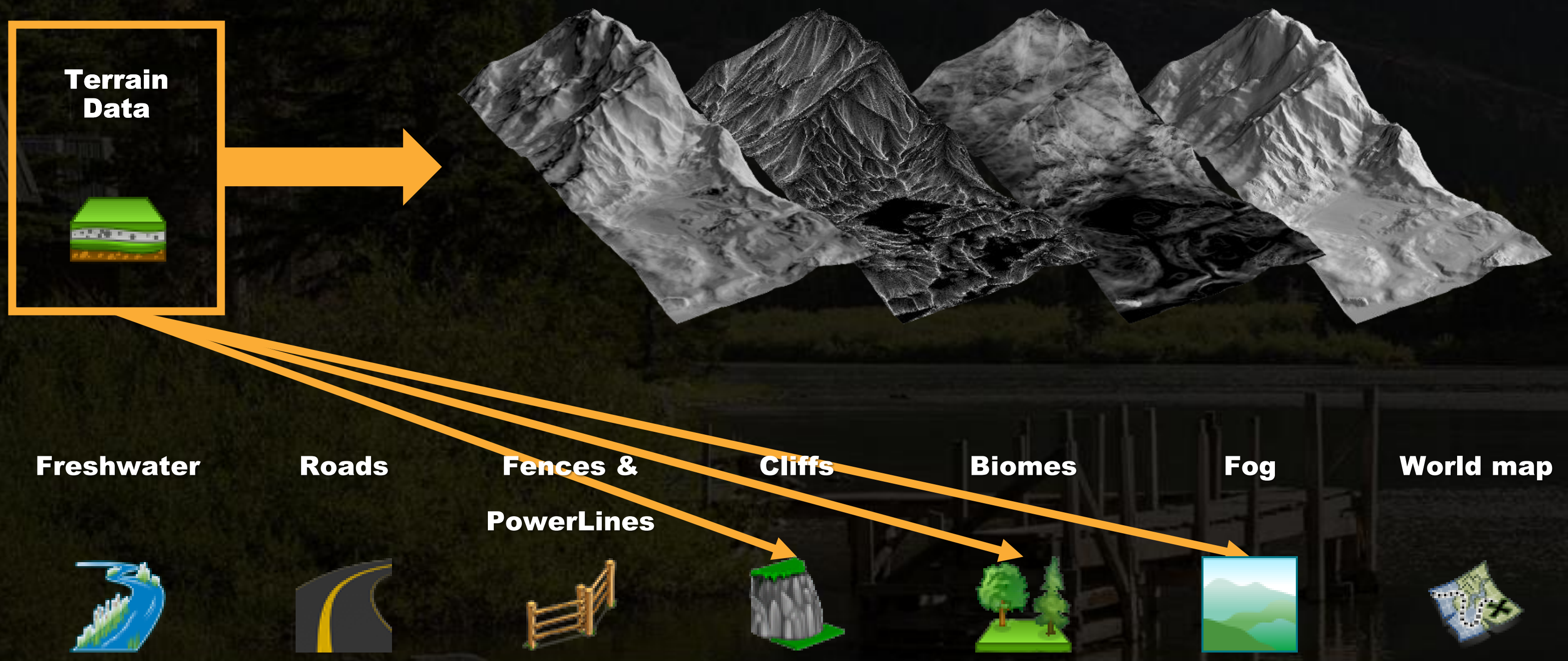
Pre-caching on disk was a bad idea

Another thing we change along the way is the terrain abiotic data generation. It is use in multiple tools. So at first we had a button to generate it separately and cache it on disk. If there was terraforming changes users had to rebake it before generating other operations like the biome tool. People got confuse about when it was required to rebake it.

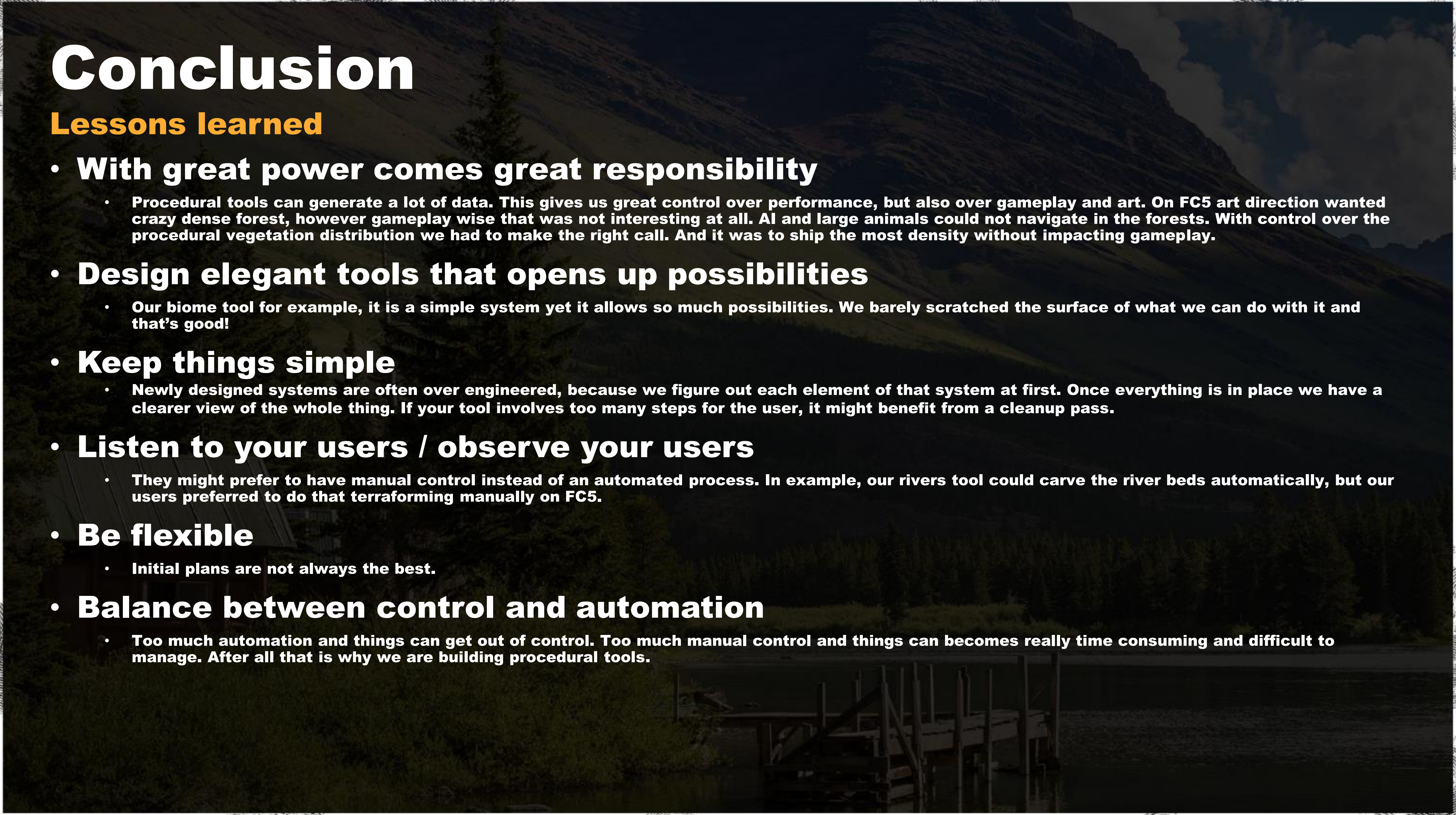
As a result, it was not always rebaked after terraforming changes which led to different result than the real final results given by the build machine (that was rebaking everything properly).

The reason why we separated this process was to save a bit of time when baking these other tools. In the end this was an extra step that brought complexity for the user and it was not worth it.

So we eliminated this extra step and simply incorporated the terrain data generation inside the tools that required that data. That way we are sure that it is always up to date.



9- CONCLUSION



Conclusion

Lessons learned

- **With great power comes great responsibility**
  - Procedural tools can generate a lot of data. This gives us great control over performance, but also over gameplay and art. On FC5 art direction wanted crazy dense forest, however gameplay wise that was not interesting at all. AI and large animals could not navigate in the forests. With control over the procedural vegetation distribution we had to make the right call. And it was to ship the most density without impacting gameplay.
- **Design elegant tools that opens up possibilities**
  - Our biome tool for example, it is a simple system yet it allows so much possibilities. We barely scratched the surface of what we can do with it and that's good!
- **Keep things simple**
  - Newly designed systems are often over engineered, because we figure out each element of that system at first. Once everything is in place we have a clearer view of the whole thing. If your tool involves too many steps for the user, it might benefit from a cleanup pass.
- **Listen to your users / observe your users**
  - They might prefer to have manual control instead of an automated process. In example, our rivers tool could carve the river beds automatically, but our users preferred to do that terraforming manually on FC5.
- **Be flexible**
  - Initial plans are not always the best.
- **Balance between control and automation**
  - Too much automation and things can get out of control. Too much manual control and things can becomes really time consuming and difficult to manage. After all that is why we are building procedural tools.

Development contribution

Thanks to:

- Christian Sirois
- David Kaufman
- Guillaume Gervais
- Israel Duchesne
- Jeremy Moore
- Jonathan Meunier

- Andrejs Verlis
- Gary Ng Thow Hing
- Julia Lynen
- Maurits Laanbroek
- Philippe Bernard
- Waldo Bronchart

Thanks =)