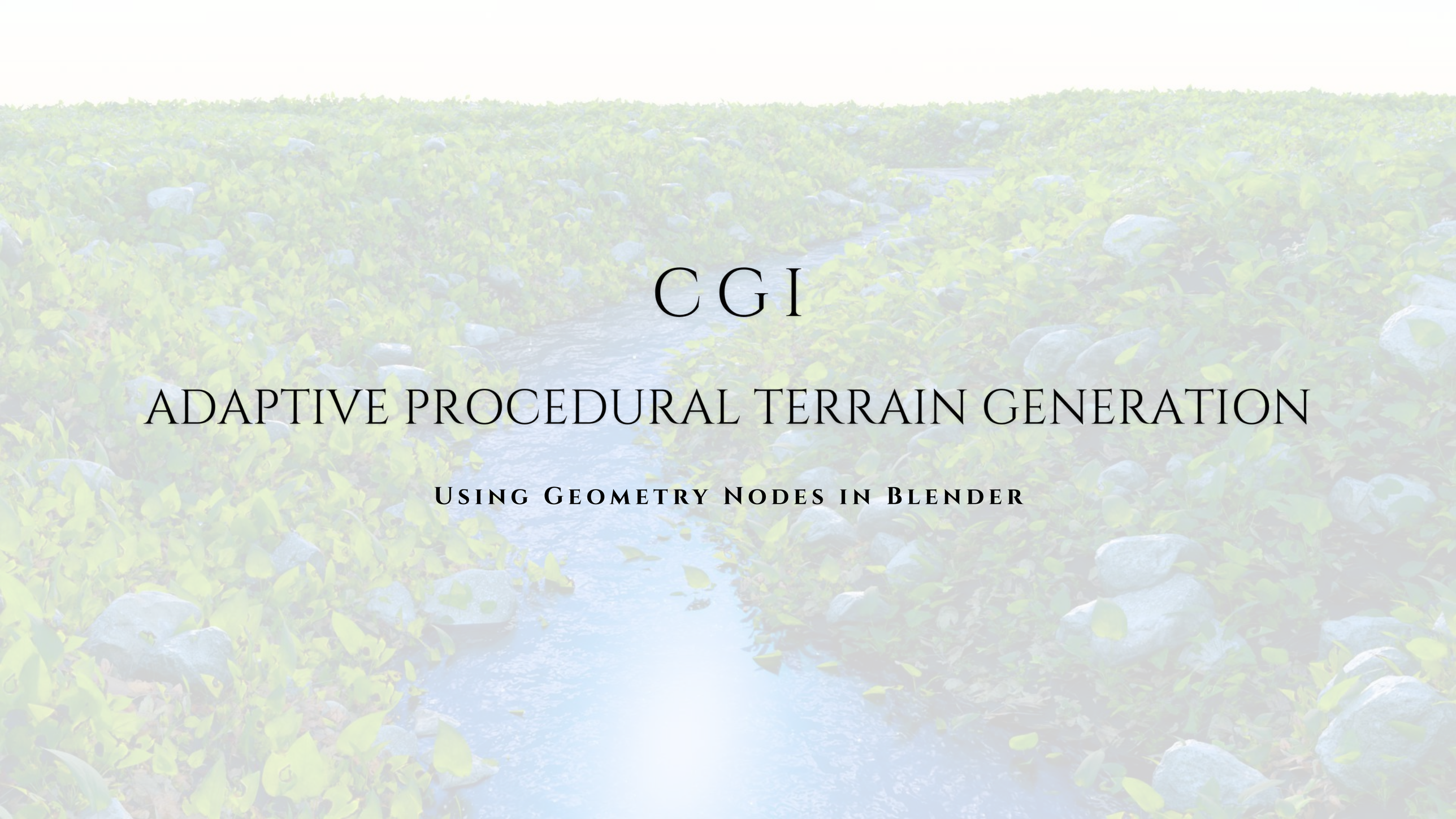


# CGI

# ADAPTIVE PROCEDURAL TERRAIN GENERATION

## Using Geometry Nodes in Blender

# WHAT ARE GEOMETRY NODES

Geometry Nodes in Blender are a node-based visual programming system that lets you create, modify, and control 3D geometry procedurally — without manually modeling.

Similar to how python scripting works, geo nodes is essentially visual scripting, that is very useful for geometric application and real time editing.
You connect nodes that perform operations like generating shapes, modifying meshes, scattering objects, or applying effects.
It's non-destructive — changes are driven by data, not manual edits.

**Features**

**Procedural Control:** Adaptable systems where changing one input (egheight, density) affects the entire setup, but can always be reverted/modified later on.
**Efficiency:** Automate repetitive tasks like placing rocks, grass, or trees.
**Scalability:** Ideal for environments, terrains, and VFX that need flexibility.

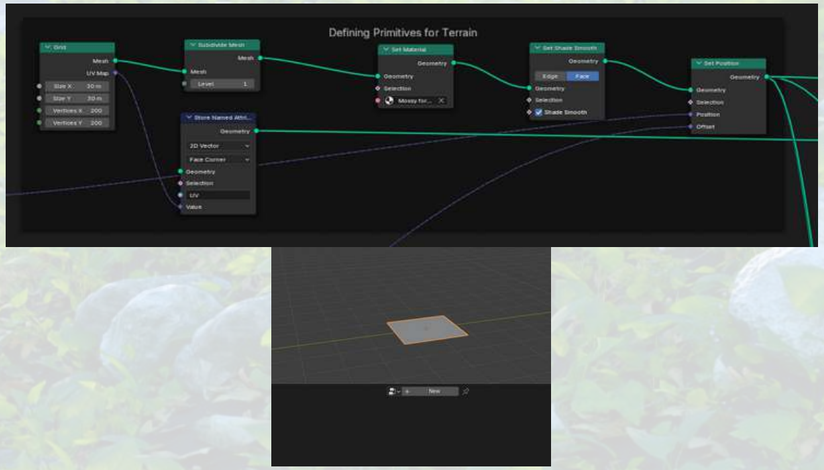
**Use Cases:**

- Procedural terrain and erosion systems
- Scattering vegetation or debris
- Parametric modeling (buildings, props)
- Dynamic effects (growth, disintegration, trails)

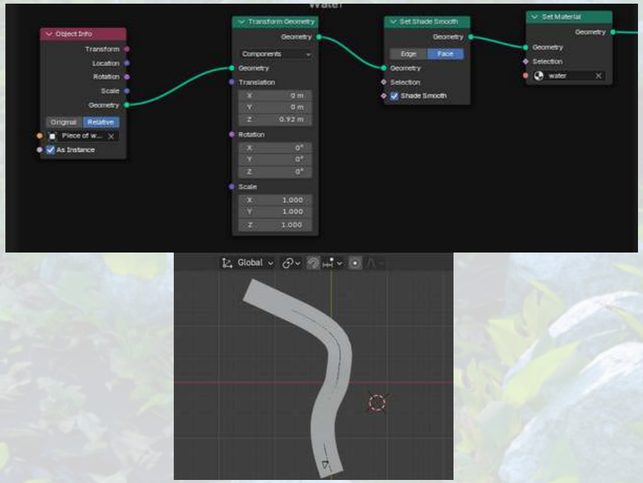
# CREATING THE TERRAIN & WATER

**Base Terrain**
Starting with a square Plane (4 vertices).
I first scale and subdivide the mesh and set a material to it.

The starting mesh does not really matter, because I am anyway deleting it and replacing it with a plane. I could have even used a sphere or cube for the terrain.

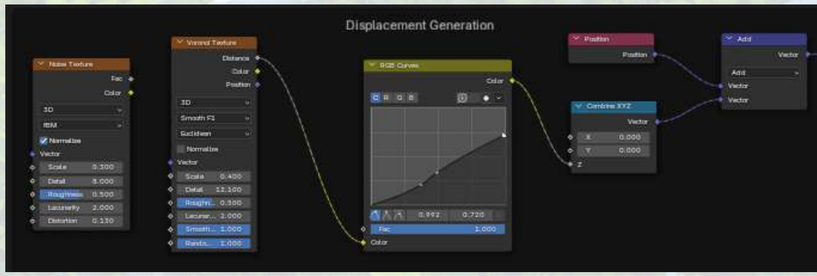


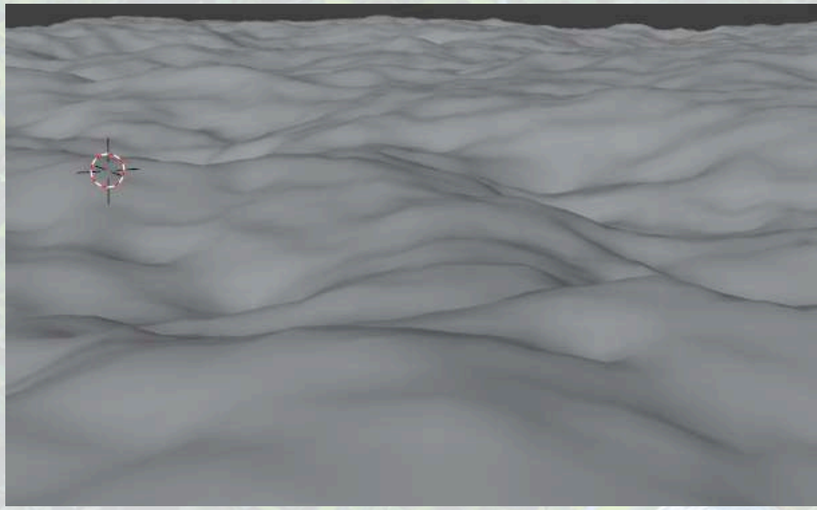

**Water**
I've taken a NURBS Path, and wrapped an array of planes to it, such that the planes follow the curve and set a material to it. To interact with the terrain, I have converted the curve to mesh.



# DISPLACEMENT GENERATION

I've done this using a noise generation node/function. (Perlin/Voronoi etc.). This is a 2D noise pattern that I have used to generate 3D height maps using the Combine XYZ node, which allows us to separate the 3 axes and choose which one we want to affect. To generate displacement, we need to use the Z value of the plane.
We control the displacement using a curve. This is a nonlinear adjustment to the height values from the noise texture. The noise is a gradient of black and white. I've used it such that the white regions mean elevation and black means depressions - but this can be inverted at any time by multiplying by -1.
On the curve,
Lower distances are lifted slightly.
Midrange is more steeply boosted.
Higher values are closer to linear.
What I've done next is to take the original position of that point and then vector add it to the new value from the curve. This pushes points on the terrain up and down and generates displacement depending on the noise texture.

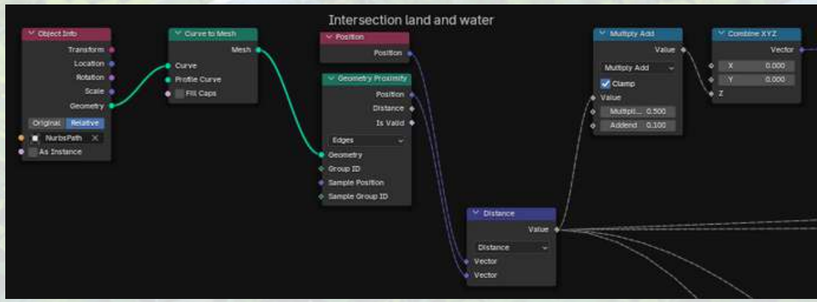


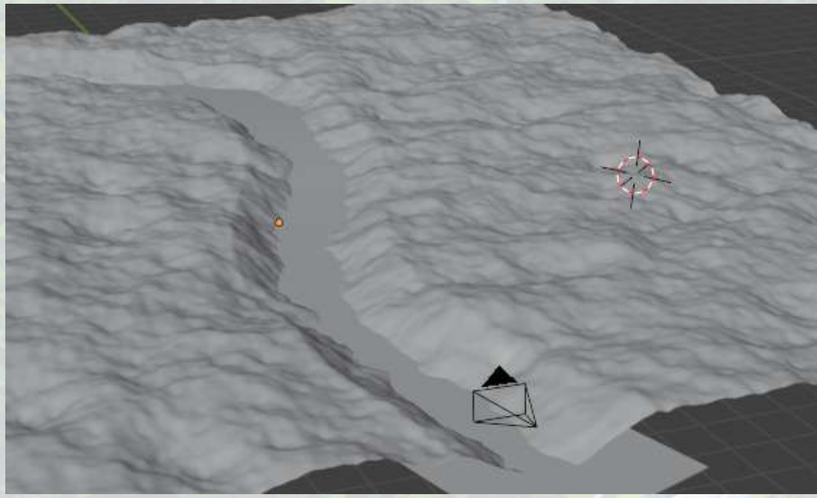



# LAND & WATER INTERSECTION - GEOMETRY PROXIMITY

Geometry Proximity and distance node calculate points on the terrain from the river (NURBS Path). This is connected to a XYZ node, which allows us to affect each of the axes separately. For this, we want the Z value to be affected, So that points on terrain that are close to the river, their Z values (displacement values) are lowered, below river level.

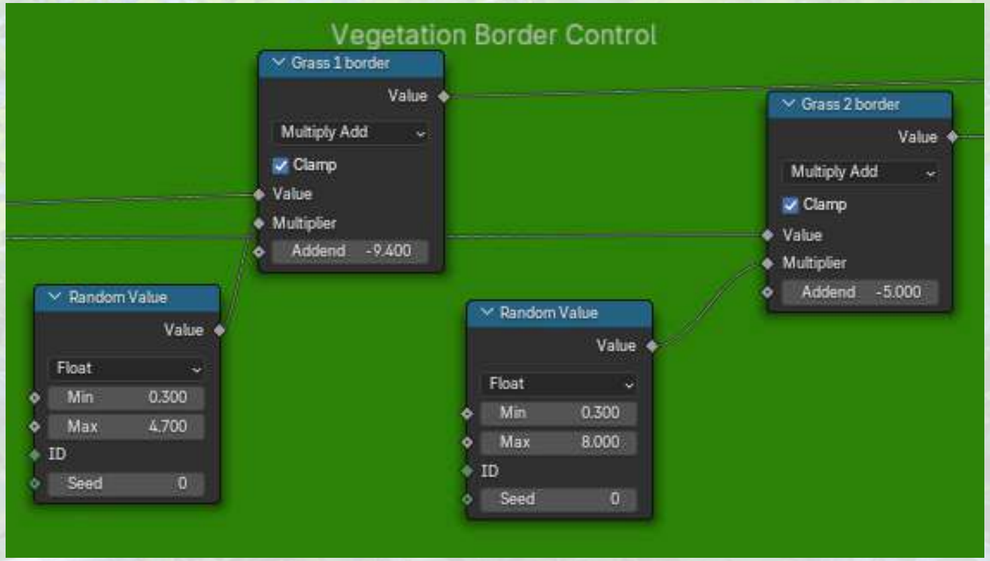
How much to lower the Z value is done by the multiply add node - which decides from where to start lowering the Z values. This basically controls how wide the river should be and the falloff of the land.
If Distance = 0.0 → Output = 0.04 (closer to river edge - these points will be have the lowest Z value)
If Distance = 1.0 → Output = 0.77 (0.73 * 1 + 0.04) (farther away from river - no change in Z value.
But Clamping is on, so any values outside the 0-1 range will be clipped.
This gives a falloff from the river outward, between 0.04 and 0.77 ( for this setting of the Multiply node.)

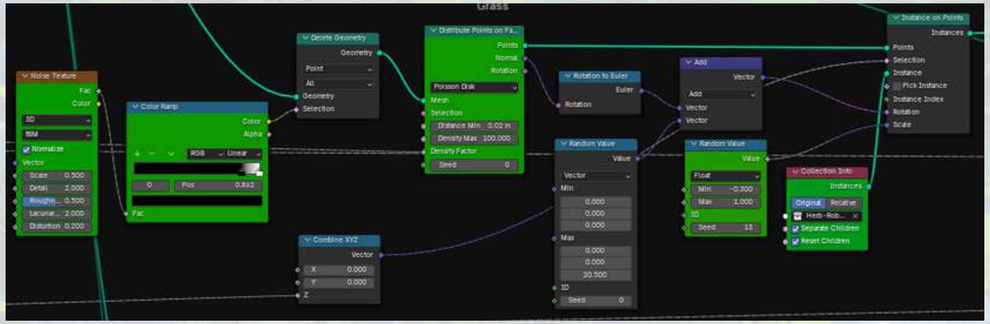


# SCATTER SYSTEMS

I have used 2 scattering systems.
One is a global scattering, that scatters the chosen model over the entire landscape - but is gradually removed closer to the river.
The points(grass models) that are scattered are controlled by the geometry proximity setup above.
Scattering using geo nodes is done by marking points all over the geometry - this is what the density value of the node does(how many points do you want over that geometry). And each point is instanced to a model/object or a collection of objects that we can choose. So now I am rending the points as plant meshes.
Output = (Distance Value × Multiplier) + Addend.
[Distance value is dist of point from river.]
This fades out the density of Grass 1 and Grass 2 near the river.
The way they are scattered is through a Perlin noise and a Voronoi noise texture. I've done this to make the distribution a more random. I've applied random rotation according to the normal of the surface.
Scale is also randomized between a set range.



# SCATTER SYSTEMS

The second is a system that scatters rocks and leaves close to the water. The scattering method is the same, but the border control is inverted. That's why I have multiplied the the distance value by -1. This inverts the scattering and now, experimenting with the multiplier and add nodes, I can get more rocks and leaves close to/in the river.

Random Value Node

Outputs a random float per instance between 0.3 and 4.0.

Multiply Node

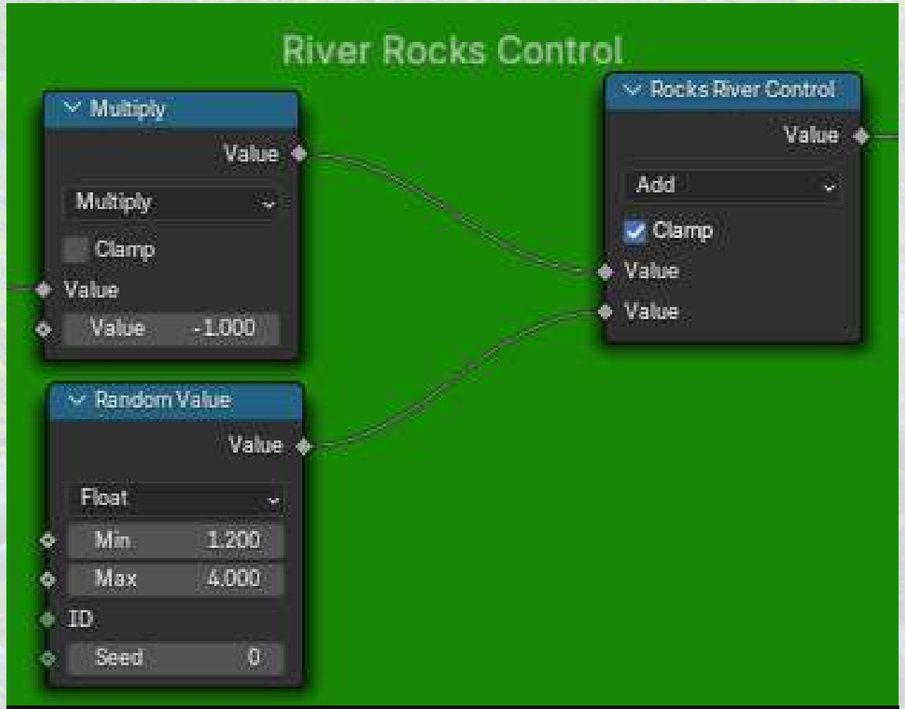
Multiplies R by -1.0.

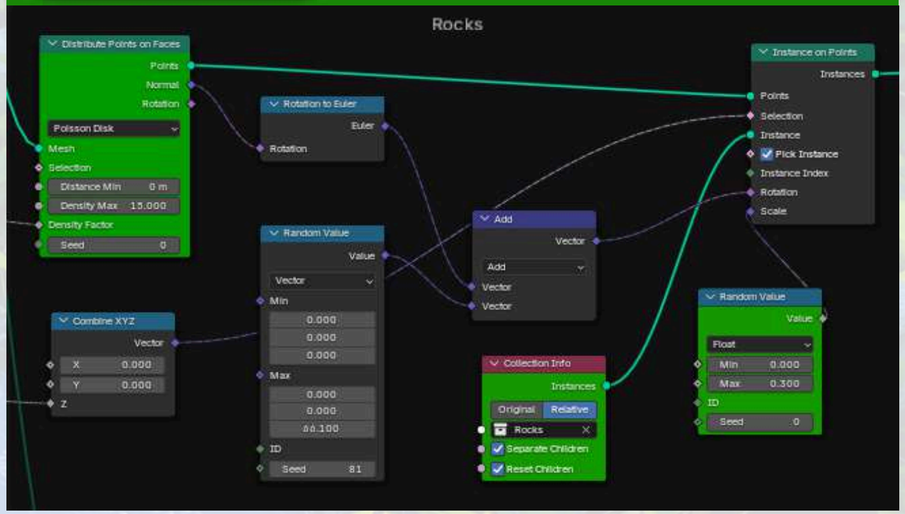
So now:

M=R × -1.0

This just inverts the random value, flipping the range from (0.3 → 4.0) to (-4.0 → -0.3).

The add node randomly subtracts a value between the range from some other scalar input, but clamps it to prevent the value from going below 0.

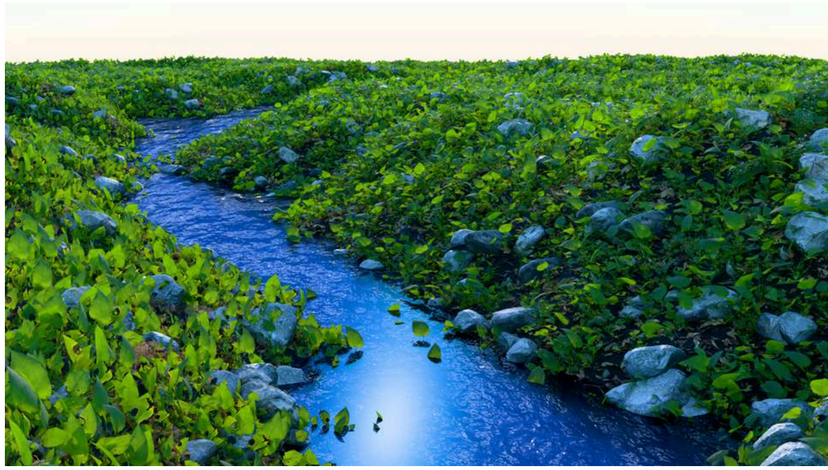


Why use this entire Geometry Node system, instead of using older methods.

Normally, to make a terrain like this, first you would have to make the landscape and add all the displacement. And to add the waterbody, you would have to line up the water manually to get the correct height and width. Then add different scatter systems for different vegetation types. And if you would need to make a change to the river for example, you would have to edit the mesh of the landscape again and vice versa.

This geo node approach is procedural & non-destructive. -You can adjust rivers, paths, vegetation zones, etc., in real-time without redoing the terrain.
In a few clicks you have a full terrain with you - one can keep adding scatter systems, more waterbodies. This is very helpful for creating biomes(forest, mountains, arctic).
All the user has to do is have their models/obj files downloaded so that they can drop it into the tree.
You can make custom drivers and keep adding to the node tree and testing new variations in real time.
This geo node setup can be used across multiple projects.
But, this system has its limitations and older systems are better for different purposes



CAMERA POSTION IS THE SAME FOR ALL 3 RENDERS - ONLY MANUAL CHANGE IS SKY POSITION.
VEGETATION, ROCKS, LEAVES, LANDSCAPE- ALL MODIFY ACCORDING TO RIVER SHAPE IN REAL TIME.

# EROSION

Simulates basic thermal erosion on a mesh by smoothing out steep slopes across connected vertices.

Variables:

- Talus Angle: The slope threshold; erosion happens if a vertex is too high compared to its neighbors.
- Erosion Strength: Controls how much elevation is transferred per step.
- Iterations: Number of times erosion is applied — more = smoother.

How Erosion works:

- Select a mesh object in Blender and run the script.
- Switches to Edit Mode and accesses the mesh geometry via BMesh.
- For each iteration:
  - Loops through every vertex.
  - Compares height (z) with connected vertices.
  - If too steep (greater than talus_angle), transfers some height (erosion).
- Applies the new height values after each loop.
- Switches back to Object Mode.

Effect of running this on the landscape:

- Smooths out sharp peaks.
- Mimics natural erosion over time.

NOTE: There is a problem with this erosion script - it is not procedural. Only once I apply the geometry node system to the landscape will the erosion run properly. But this means the landscape can't be procedurally edited anymore.

There is a way to use geometry nodes to simuate hydraulic erosion, but I couldn't figure it out in time

View Text Edit Select Format Templates Erosion

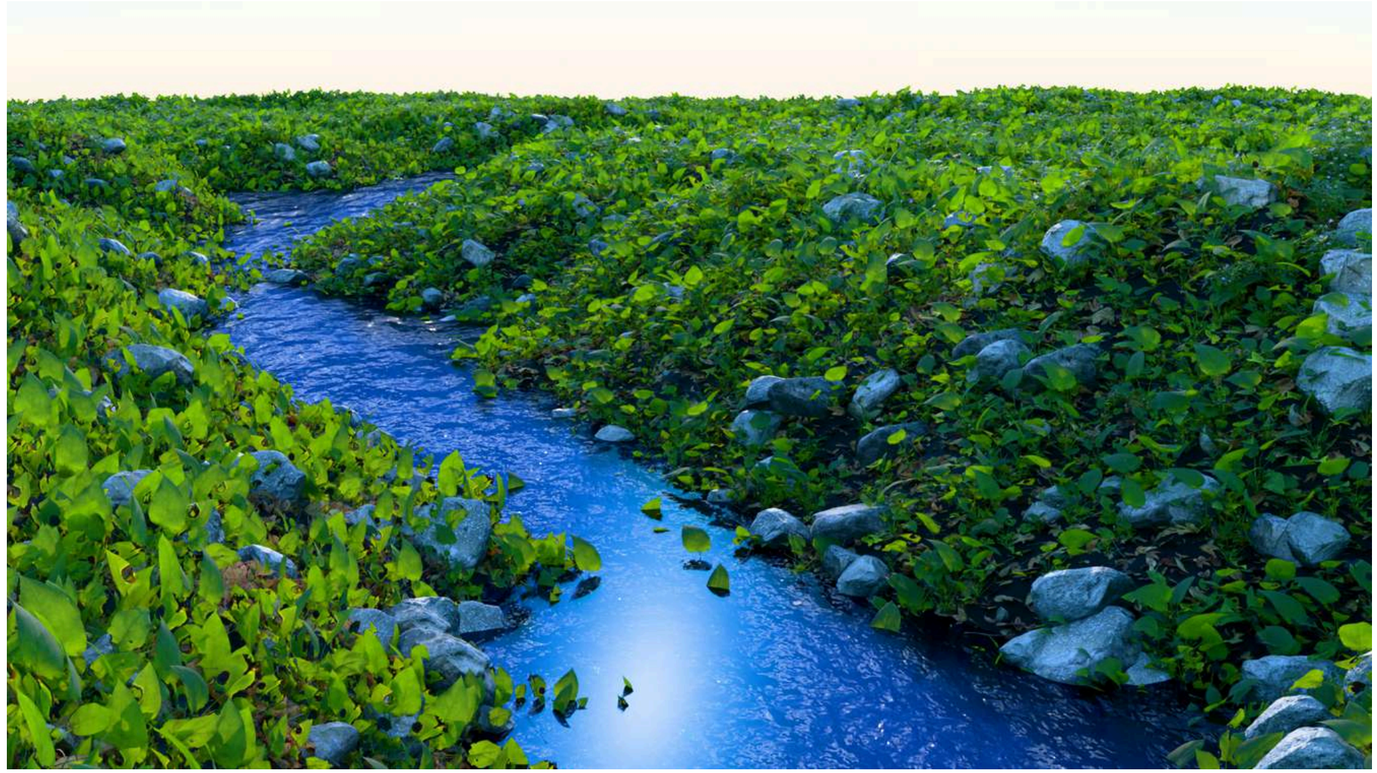
```python
import bpy
import bmesh
from mathutils import Vector

# Parameters
iterations = 10         # Number of erosion steps
talus_angle = 0.5       # Threshold slope (lower = more erosion)
erosion_strength = 0.1  # How much height is transferred per iteration

# Target object (select it first)
obj = bpy.context.object

if obj and obj.type == 'MESH':
    bpy.ops.object.mode_set(mode='EDIT')
    bm = bmesh.from_edit_mesh(obj.data)
    bm.verts.ensure_lookup_table()

    for _ in range(iterations):
        delta = [0.0] * len(bm.verts)

        # For each vertex, compare with connected verts
        for i, v in enumerate(bm.verts):
            for edge in v.link_edges:
                other = edge.other_vert(v)
                diff = v.co.z - other.co.z
                if diff > talus_angle:
                    move = (diff - talus_angle) * erosion_strength * 0.5
                    delta[i] -= move
                    delta[other.index] += move

        # Apply accumulated changes
        for i, v in enumerate(bm.verts):
            v.co.z += delta[i]

    bmesh.update_edit_mesh(obj.data)
    bpy.ops.object.mode_set(mode='OBJECT')
    print("Erosion simulated")
else:
    print("No object selected. Select any object and then run again.")
```



# TEST RENDERS








# REFERENCES


https://www.youtube.com/watch?v=-MvJ8NmV5Fg

https://www.youtube.com/watch?v=d0igAGQ3sZA

https://youtu.be/a-4oCHe-hDE?si=TbHfPecBJRrcyOX3

https://youtu.be/tyPbOb9P5rA?si=QIxe2Z73ojQBQpZ_

https://www.cs.cmu.edu/~112/notes/student-tp-guides/Terrain.pdf

https://www.youtube.com/watch?v=XpG3YqUkCTY


THANK YOU