SKALLA

TECHNOLOGY

FOR THE NEXT GENERATION OF OPEN WORLD GAMES

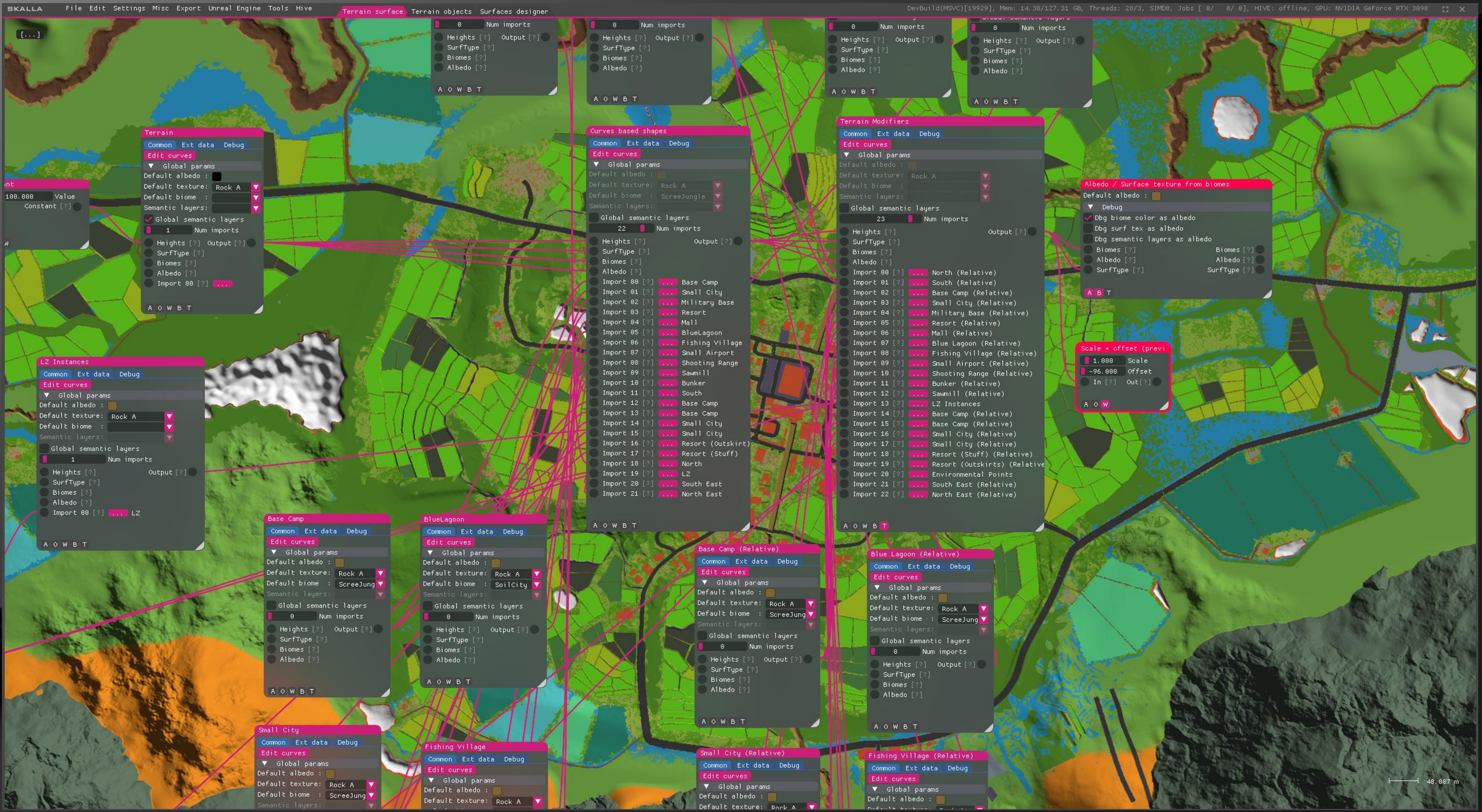
All graphics presented here is courtesy of MADFINGER GAMES:

GRAY ZONE: WARFARE

What is SKALLA ?

- SKALLA is a software for the creation of massive virtual worlds. Utilizing the vector based modeling primitives, augmented by the real-time procedural generation.
- It adopts the powerful node-graph based non destructive workflow pipeline.
- It allows multiple users to work simultaneously on a single world in real-time.
- It has tight integration with Unreal Engine 5, where edits of the world are immediately reflected in the engine.

Typical view of the SKALLA viewport


SKALLA
File Edit Settings Misc Export Unreal Engine Tools Hive
Terrain surface Terrain objects Surfaces designer
[...]
48.007 m

DevBuild(MSVC)[19929], Mem: 14.38/127.31 GB, Threads: 28/3, SIMD8, Jobs [0/ 0/ 0], HIVE: offline, GPU: NVIDIA GeForce RTX 3090

0 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
A O W B T

0 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
A O W B T

0 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
A O W B T

0 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
A O W B T

nt
100.000 Value
Constant [?]
W

Terrain
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome :
Semantic layers:
☑ Global semantic layers
1 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
Import 00 [?] ...
A O W B T

Curves based shapes
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome : ScreeJungle
Semantic layers:
☐ Global semantic layers
22 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
Import 00 [?] ... Base Camp
Import 01 [?] ... Small City
Import 02 [?] ... Military Base
Import 03 [?] ... Resort
Import 04 [?] ... Mall
Import 05 [?] ... BlueLagoon
Import 06 [?] ... Fishing Village
Import 07 [?] ... Small Airport
Import 08 [?] ... Shooting Range
Import 09 [?] ... Sawmill
Import 10 [?] ... Bunker
Import 11 [?] ... South
Import 12 [?] ... Base Camp
Import 13 [?] ... Base Camp
Import 14 [?] ... Small City
Import 15 [?] ... Small City
Import 16 [?] ... Resort (Outskirt)
Import 17 [?] ... Resort (Stuff)
Import 18 [?] ... North
Import 19 [?] ... LZ
Import 20 [?] ... South East
Import 21 [?] ... North East
A O W B T

Terrain Modifiers
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome :
Semantic layers:
☐ Global semantic layers
23 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
Import 00 [?] ... North (Relative)
Import 01 [?] ... South (Relative)
Import 02 [?] ... Base Camp (Relative)
Import 03 [?] ... Small City (Relative)
Import 04 [?] ... Military Base (Relative)
Import 05 [?] ... Resort (Relative)
Import 06 [?] ... Mall (Relative)
Import 07 [?] ... Blue Lagoon (Relative)
Import 08 [?] ... Fishing Village (Relative)
Import 09 [?] ... Small Airport (Relative)
Import 10 [?] ... Shooting Range (Relative)
Import 11 [?] ... Bunker (Relative)
Import 12 [?] ... Sawmill (Relative)
Import 13 [?] ... LZ Instances
Import 14 [?] ... Base Camp (Relative)
Import 15 [?] ... Base Camp (Relative)
Import 16 [?] ... Small City (Relative)
Import 17 [?] ... Small City (Relative)
Import 18 [?] ... Resort (Stuff) (Relative)
Import 19 [?] ... Resort (Outskirts) (Relative
Import 20 [?] ... Environmental Points
Import 21 [?] ... South East (Relative)
Import 22 [?] ... North East (Relative)
A O W B T

Albedo / Surface texture from biomes
Default albedo :
▼ Debug
☑ Dbg biome color as albedo
☐ Dbg surf tex as albedo
☐ Dbg semantic layers as albedo
Biomes [?] Biomes [?]
Albedo [?] Albedo [?]
SurfType [?] SurfType [?]
A B T

Scale + offset (previ
1.000 Scale
-96.000 Offset
In [?] Out [?]
A O W

LZ Instances
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome :
Semantic layers:
☐ Global semantic layers
1 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
Import 00 [?] ... LZ
A O W B T

Base Camp
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome : ScreeJung
Semantic layers:
☐ Global semantic layers
0 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
A O W B T

BlueLagoon
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome : SoilCity
Semantic layers:
☐ Global semantic layers
0 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
A O W B T

Base Camp (Relative)
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome : ScreeJung
Semantic layers:
☐ Global semantic layers
0 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
A O W B T

Blue Lagoon (Relative)
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome : ScreeJung
Semantic layers:
☐ Global semantic layers
0 Num imports
Heights [?] Output [?]
SurfType [?]
Biomes [?]
Albedo [?]
A O W B T

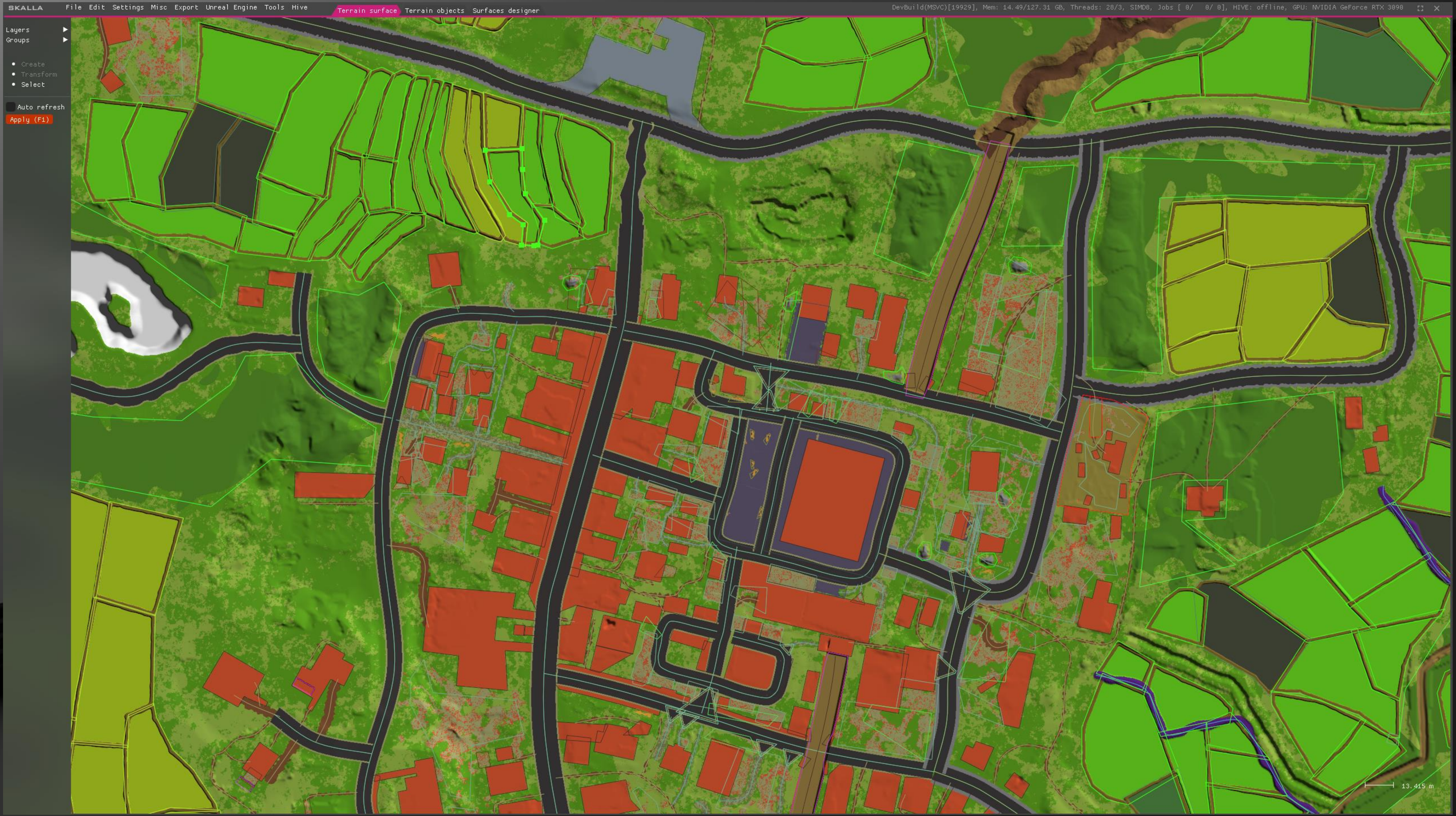
Small City
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A
Default biome : ScreeJung
Semantic layers:

Fishing Village
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A

Small City (Relative)
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :
Default texture: Rock A

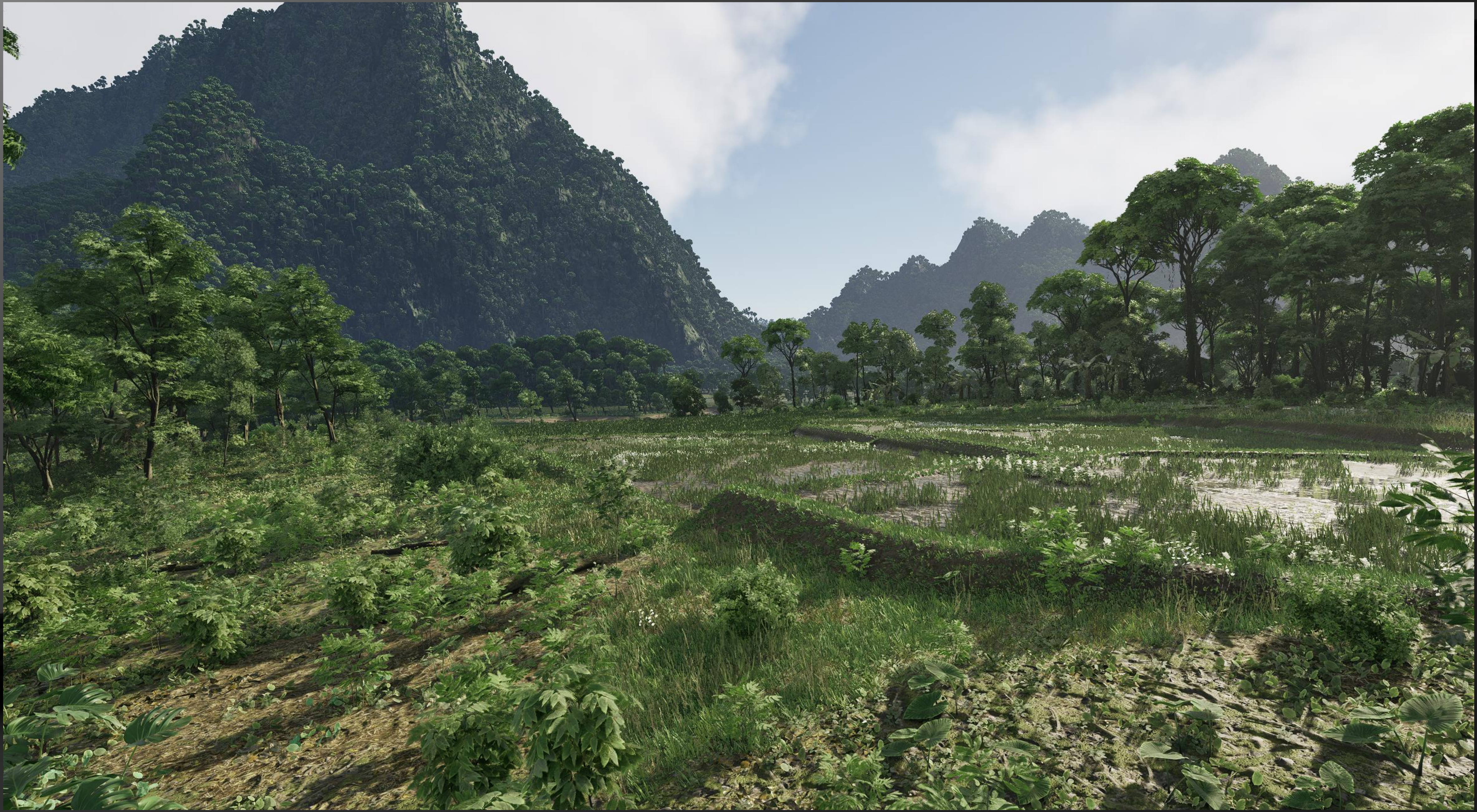
Fishing Village (Relative)
Common Ext data Debug
Edit curves
▼ Global params
Default albedo :

Vector based primitives for part of world



DevBuild(MSVC)[19929], Mem: 14.49/127.31 GB, Threads: 28/3, SIMD8, Jobs [0/   0/ 0], HIVE: offline, GPU: NVIDIA GeForce RTX 3090

Typical UE view



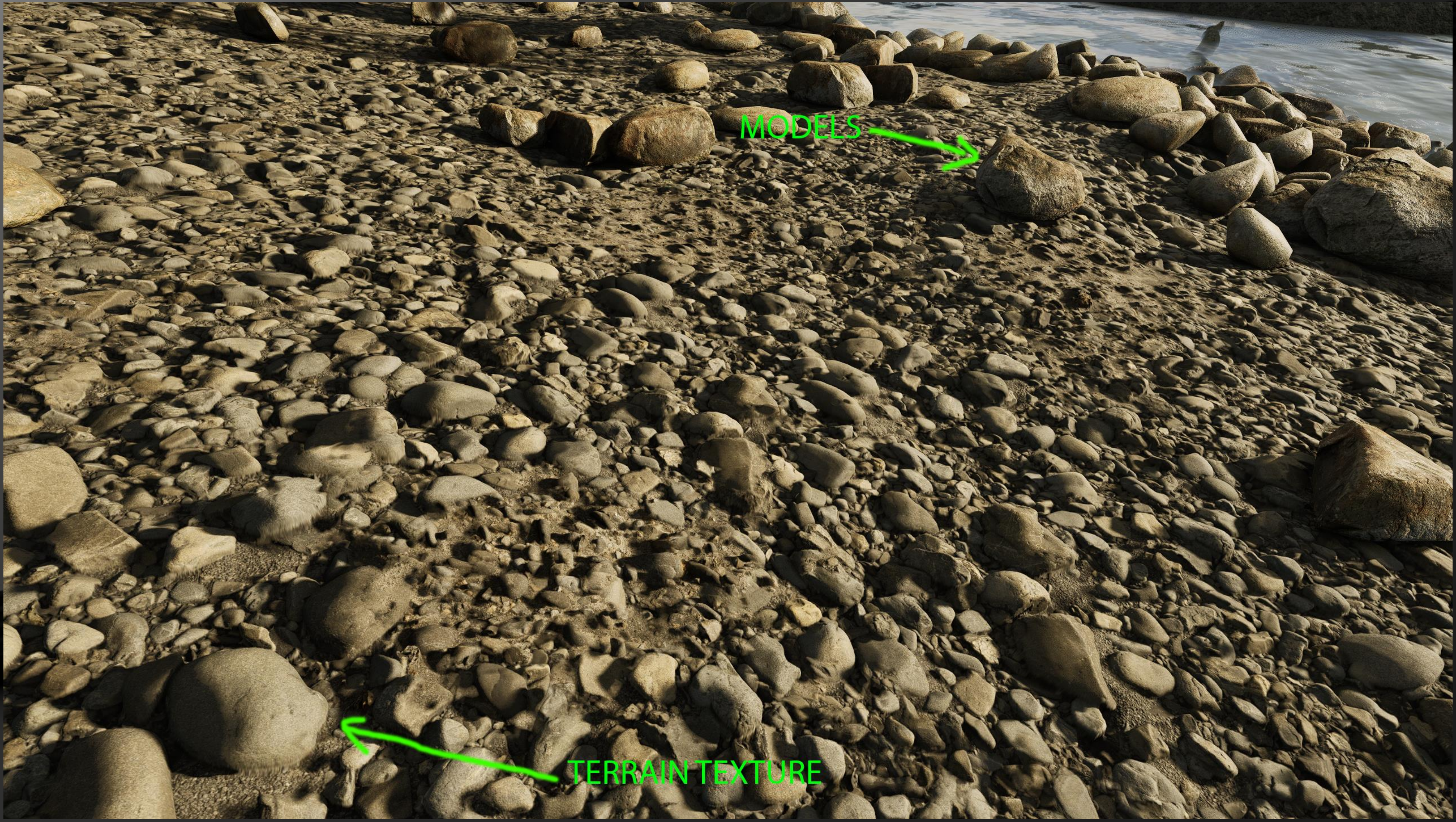
Terrain representation

- The terrain is a heightfield based and it utilizes the Virtual Heightfield Mesh (VHM) for the rendering.
- All the terrain data (elevation, splat mask, water height, water flow direction, …) are represented as a runtime virtual textures allowing for extreme resolutions while keeping fixed memory budget.
- In the current project we use elevation data resolution of 524288^2 for 32x32 km terrain giving us 6.1 cm details.
- The terrain water is also VHM, but it is rendered only in the tiles containing the water.

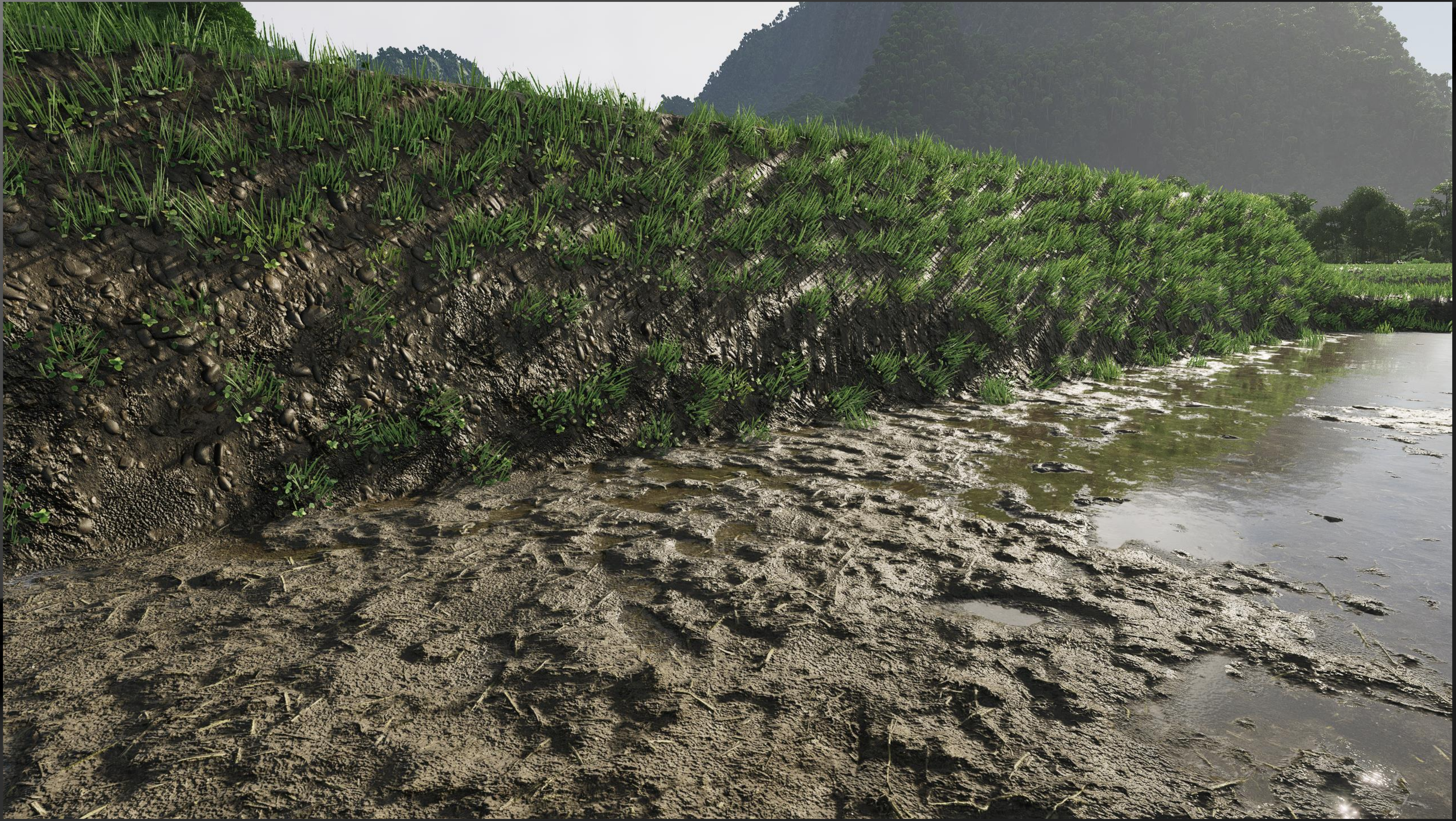
Terrain texturing

- SKALLA implements state of the art terrain texturing system.
- The terrain can use up to 127 textures (grass, sand, rock, soil, …). Number of textures used does not affect the performance. They are kept in the texture array, so they must fit into the GPU memory.
- The terrain material features.
  - Per pixel displacement mapping with soft shadows.
  - Automatic tri-planar mapping – textures on cliffs are not stretched.
  - Support for procedural wetness.
  - Height based blending between different texture types.
  - Stochastic texturing to hide texture tiling.

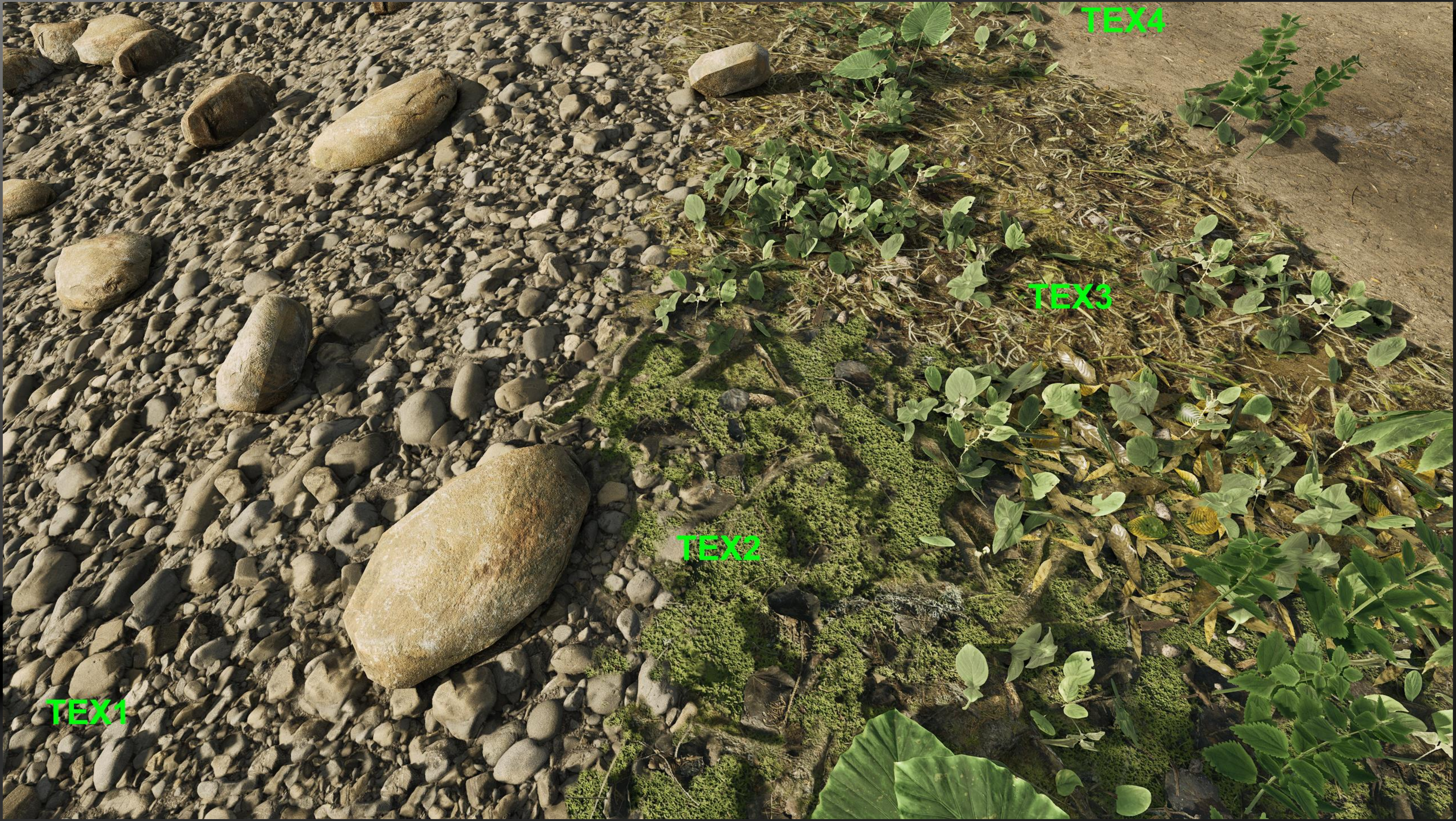
Per pixel displacement mapping in action



Wetness & tri-planar in action



Height based blending in action



Terrain objects

- The terrain is populated by objects based on biomes, where biome defines the rules for spawning of specific models.
- Importing objects placed in UE to SKALLA and than using our system to render them is also supported.
- On Unreal Engine side, we have implemented custom solution for rendering massive amounts of objects.
- Also, we support fast collision queries with procedurally placed objects through custom API, which can be used by game code.

Terrain objects

- Terrain objects are organized into layers based on object size, which puts limits on their density and max view distance :
  - Small (grass, flowers, small rocks, …), max vis dist ~200 m
  - Medium (bushes, large rocks, …), max vis dist ~800 m
  - Large (most trees, big rocks, …), max vis dist ~3 km
  - Far (this is automatically created layer for rendering trees into “infinity”)

Objects. Lots of objects.

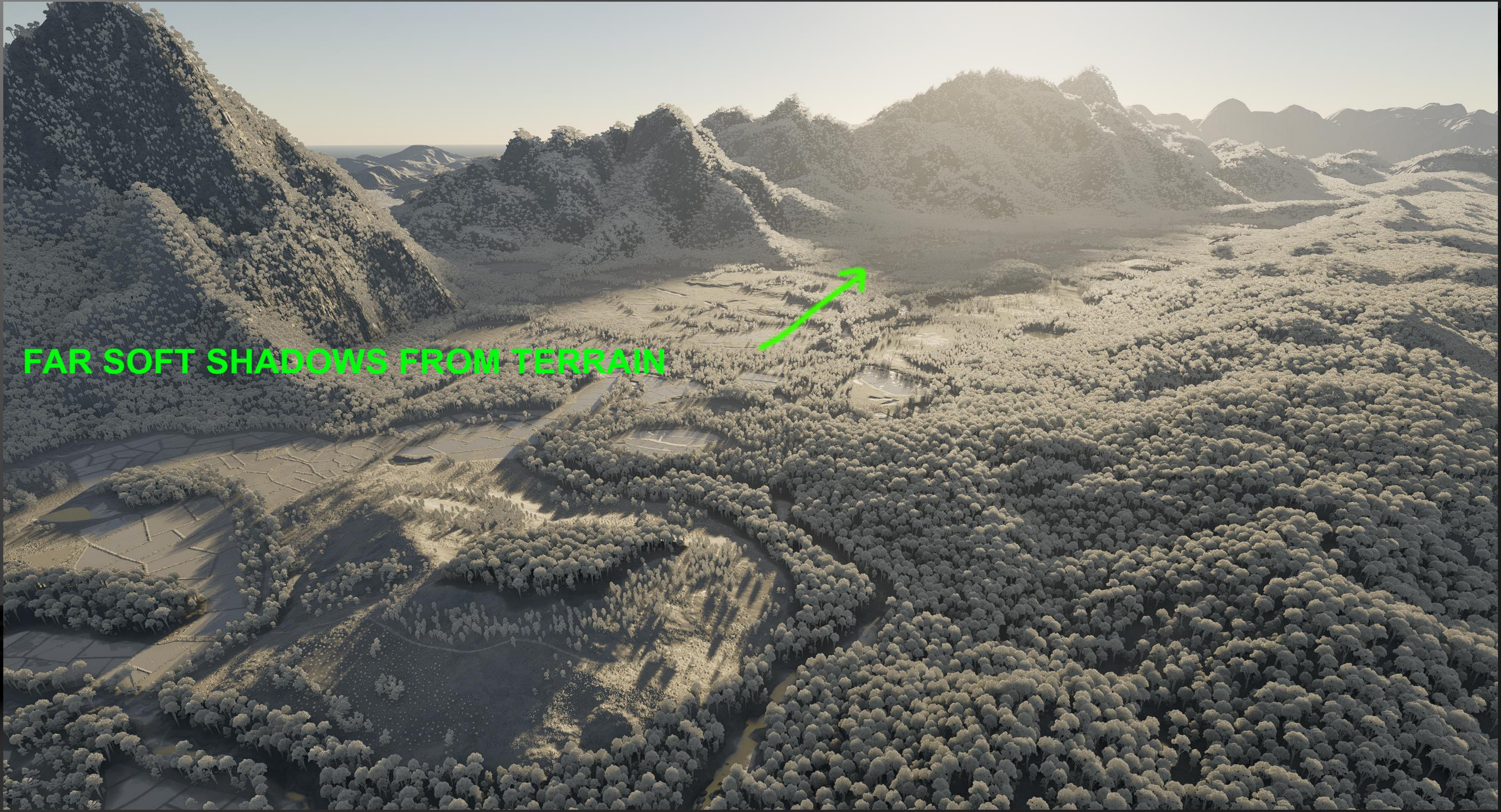
Vegetation rendering & wind

- For rendering realistic vegetation, we have created preprocessor, which takes input model and automatically generates skeleton and auxiliary data (baked into textures) for wind simulation.
- Further, we automatically generate simplified shadow casting proxy models and precompute medium-freq ambient occlusion.
- Billboard based LODs are also automatically generated.
- Finally, we also pre-generate 32bit low resolution texture which is used to generate correct self-shadowing for billboard trees over entire view range.
- We support global simulated wind as well as local wind sources like helicopters, explosions etc.

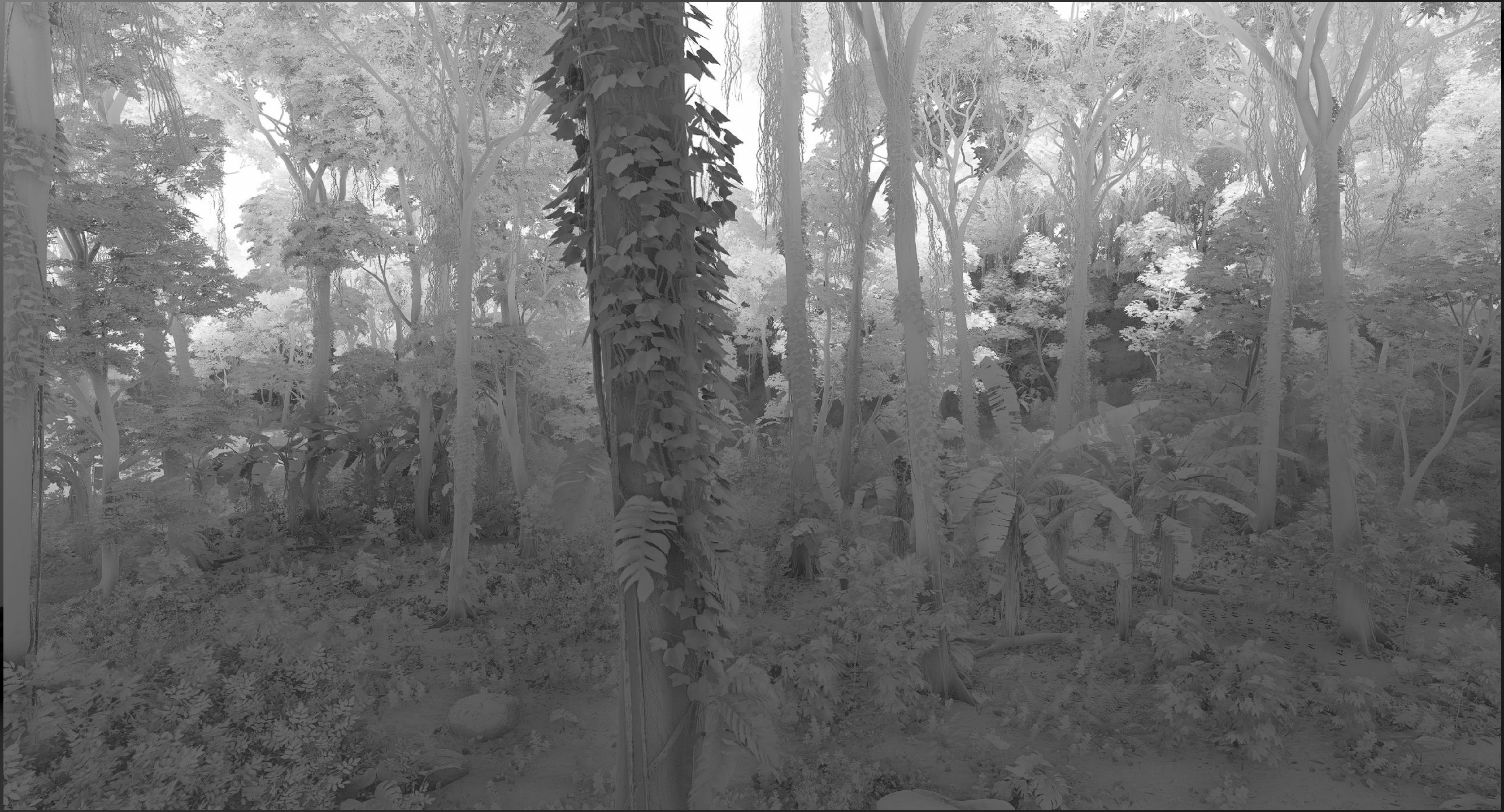
Rendering improvements and optimizations

- We have implemented several improvements and optimizations for rendering inside Unreal Engine 5:
- Terrain heightfield soft shadows over entire viewing range (shadows cast by terrain onto everything else).
- Custom shadows for foliage – using non-linear warping to get hires dynamic shadows even with single cascade (much better performance) + ‘FAR’ low resolution static shadows covering ~2 km view range
- World Ambient Occlusion – custom solution for providing consistent AO over entire world.
- Z-Prepass optimizations for foliage rendering.

Terrain soft shadows



World Ambient Occlusion



Game code support

- We support all the features that are necessary to actually build game with our tech:
- From terrain and objects, we automatically generate collisions and navmesh.
- Also, we support fast queries for collision detection to be used by AI as well as querying terrain attributes (elevation, biome, water height, water velocity, …) at specific world position.
- It is possible to spawn any UE actors from SKALLA, however in much smaller quantities due to Unreal Engine overhead. It can be used for population of biomes by sounds, particle FX etc.
- Finally, it is possible to affect wind by local ‘sources’ using supplied API.

World partition, streaming etc.

- Good news: there is no world partition and streaming necessary for SKALLA world. World description data are very small and everything else is generated from it at runtime.
- Real-time generation of data can be seen as substitute for (inferior) concept of precomputation (of large data) and streaming
- Because there is no precomputation, iterating on world is very fast even for extremely large and detailed worlds. Once you taste it, there is no going back :-)
- Also, for SKALLA data, there is zero loading time. It only depends on performance of your machine. The 32x32 km detailed world of Gray Zone: Warfare is rebuild completely from scratch in less than 3 seconds on 7950x CPU.

Terrain modeling

- We have researched & experimented with many techniques for terrain synthesis (https://youtu.be/_n7iwPLbSaM).
- However what finally works best for us is ‘bashing’ of vector based shapes and curves which in turn produce ‘distance fields’ which are procedurally enhanced and composited into heightfield.
- Shapes and curves control terrain elevation as well as biomes, texture synthesis, water etc.
- Because of vector based source data, output is resolution independent and it is also crucial for non-destructive workflow. Any aspect of terrain can be changed at any time.

Multi-user collaboration

- HIVE is simple, yet powerful multi-user collaboration system that is integrated in SKALLA.
- It uses concept of per-node ownership (in nodes graph), where each node is owned by exactly 1 user. Node can represent part of the world spatially (certain region on map) or semantically (certain component of world – ie. rivers). Or it can be combination of both.
- It uses local server to store data and serve it to users.
- Because data representing world are quite small (few MB compressed), passing data between server and clients is very quick.

Limitations

- SKALLA placed objects does not support Unreal Virtual Shadow Maps. We don't insert terrain objects into UE 'GPU scene' because of massive performance overhead. Also, we found out, that using VSM with high density foliage is currently no-go for performance, due to high resolution shadow maps combined with high overdraw of alpha keyed foliage.

- Also, because of our objects not being inserted into GPU scene, they are not visible to Lumen software / hardware ray tracing and only use information available in screen space. We compensated for this by introducing custom 'WorldAO' subsystem, which combined with Lumen screen space based tracing gives great results.