Unannounced Project: Preliminary Large World R&D

Primary Objective:

- Large outdoor worlds (as large as possible) that hit 60fps on PS4 Base.

Secondary Objectives:

- Industry leading Post FX & Lighting.
- FX-heavy magic particle systems.
- Support for cutscenes/cinematics.

Overall Concerns/Unknowns

1. Interplay of goals are all performance bound. Achieving balance and performance targets for each system.
    a. Example: If we decide we want 64 players in a large open world all casting magic FX, can we also have things like dynamic time of day and constant screen space effects all at the same time while still hitting 60FPS on PS4?
        i. Which “team” or systems would compromise? Once figured out, make sure teams know basic profiling to keep an eye on their own budgets. Establish “lanes” and budgets for each aspect quickly to maintain balance.
2. Design requirements and art direction will dictate how systems are used. We should prepare for a spectrum of possibilities.
    a. Example A: If we decide we wanted to hide culling with close fog and shorter view distances, but Art departments decide they want long vistas even if the former hits performance targets, have Hierarchical LOD/Merge Actor workflows ready.
    b. Example B: If post-processing and screen-space effects are too strong for PS4, how can we compensate? What compromises are art teams willing to make? If we wanted SSAO, but it’s cost to GPU was too high, can we drop it and do it at the material level or keep it baked into static lighting only.
3. Level of effort for expand existing UE4 systems/Tripwire-expanded systems to facilitate Artist workflows.
    a. Example: If the volume-driven procedural systems for Foliage and H-LOD Clusters is too time consuming for rapid iteration, can we add a color ID painter to Landscape to define regions and have the engine understand the color IDs the same as volumes? Or a breakoff Orthographic view to paint IDs for use in any volume-based system.
4. If we go to PC, decide what visual aspects will be dropped per settings bucket quickly so art teams can tune for each.
5. What unforeseen technical hurdles will our decided workflow come up? Who will be responsible for addressing them?
    a. Example: If a system such as world composition/grid streaming work well on local tests, what information do servers need to track? Will server architecture create any issues, or will it be okay for local clients to stream/process level/zone changes with no issue. Conversely, if the server could handle some of it, would it allow the PS4 to handle the game better?
6. How open are we to marketplace solutions if the cost is < man hours required to create the same thing? Can we use any of it for prototyping?

Large Outdoor Worlds

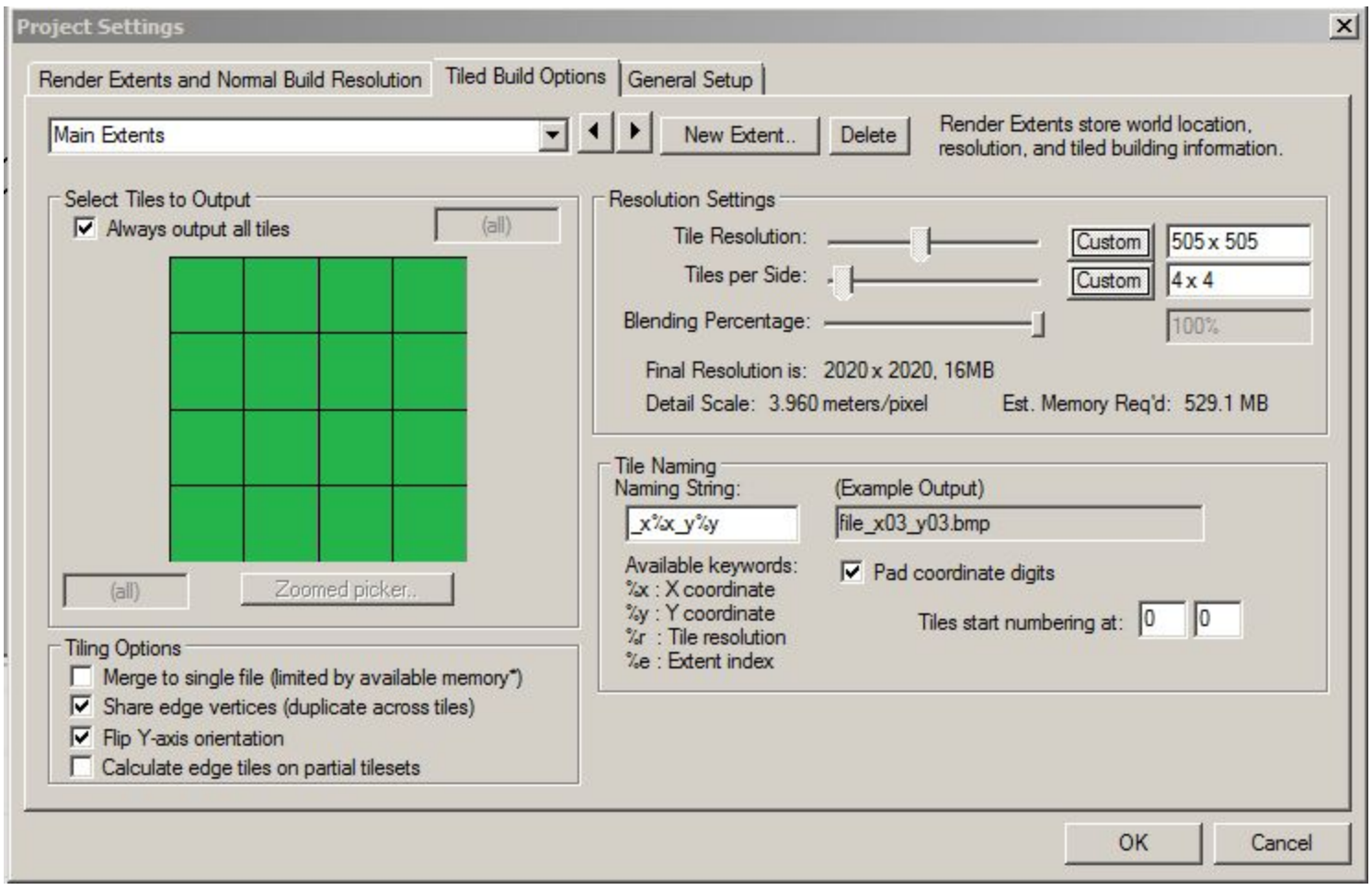
Primary Question: Which workflow do art/design teams want? See a few possibilities below:

1. **Procedural/Art Driven:** World Machine -> UE4 -> Overall ProcFoliage/Grass Pass + Landscape Material -> Design Carves Gameplay Areas out of world -> Create Gameplay Areas (Art) -> Performance Pass -> etc.
2. **Design Driven (Level Design Creates World Around Gameplay):** Design Rough in UE4 -> Heightmap to World Machine -> Back to UE4 via World Composition -> Overall ProcFoliage/Grass Pass + Landscape Material -> Subtract Village/Gameplay Areas -> Create Gameplay Areas (Art) -> Performance Pass -> etc.

Potential Workflow #1: Procedural/Art Driven: WM -> UE4

If we’re going for the largest worlds possible, we will need to leverage tiling heightmaps using World Machine to UE4 via World Composition. This will give us gigantic 8km x 8km worlds.

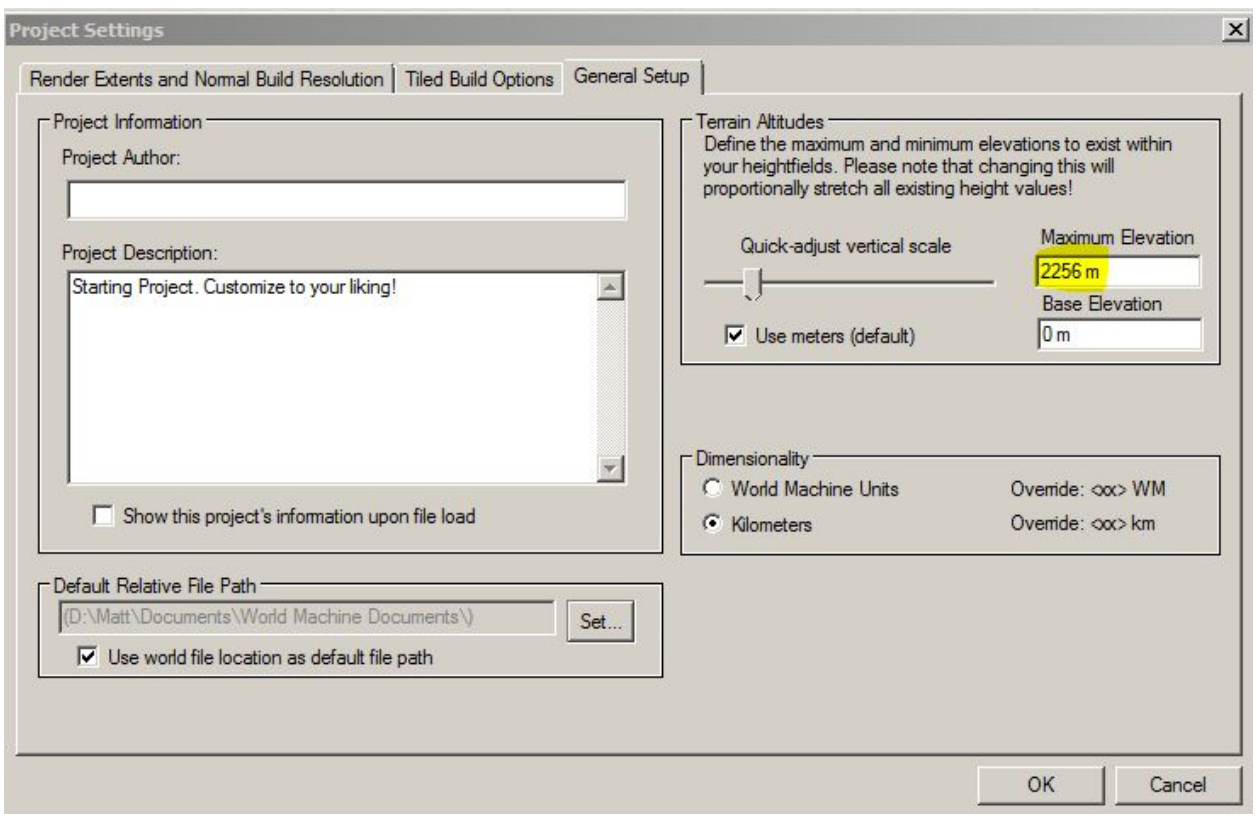
Step 1: World Machine Heightmap/Weight Map/Splat Map Generation



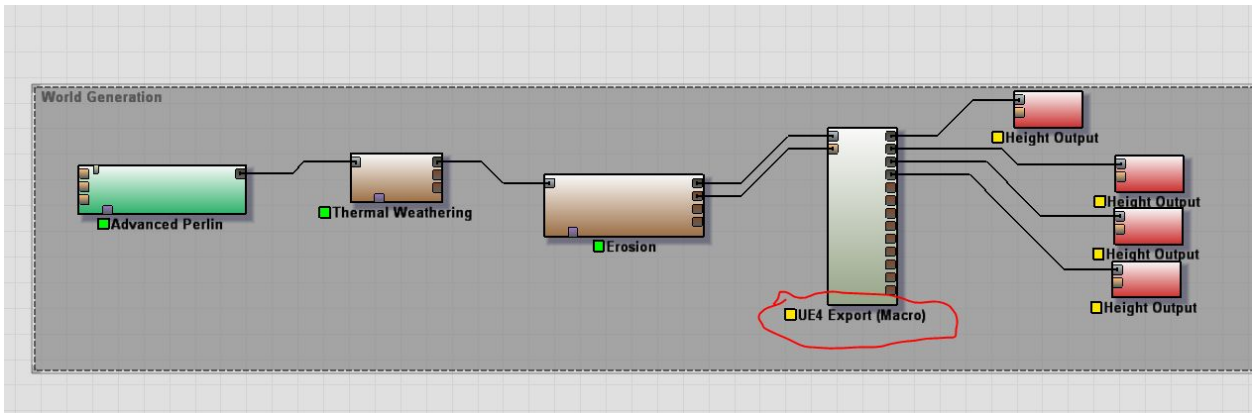
Starting Project Settings in WM for World Composition for UE4. **2017x2017** is the standard for 8km x 8km worlds.

Important Note: For this document, 2017x2017 build times had such slow (hours long) iteration times that I went with 505x505 for proof of concept.

Make note of your Max Height (used for calculations later):



Create node network to desired look. Here's a quick example node network to give a good variety of areas:

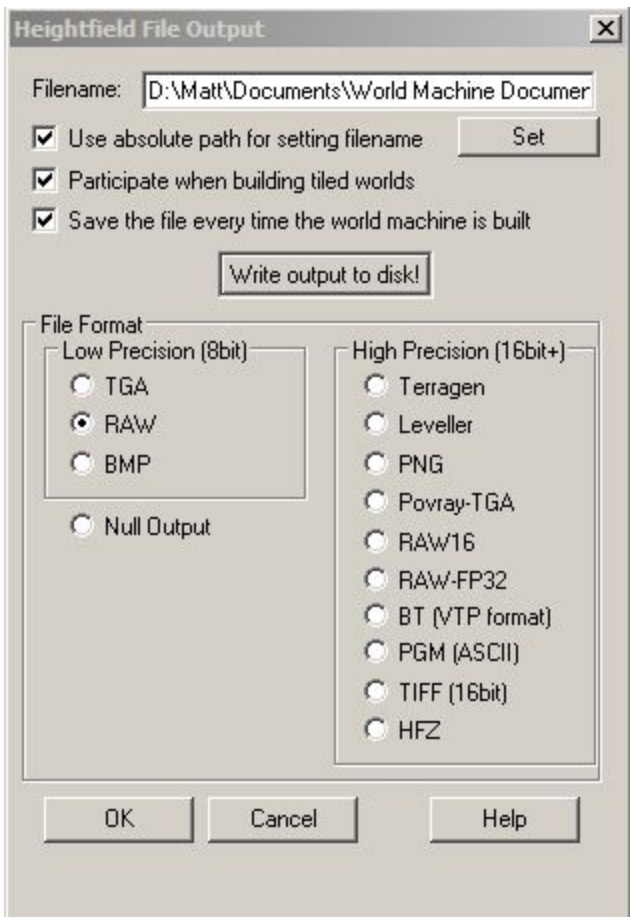


After node network is in a good spot, do a ***Tiled Build*** to export your Height Outputs for your Heightmap and Masks, and output your Bitmaps (Macro Normal & Splat Map [just in case we use it instead]).

WARNING: Tiled builds take A LONG TIME at 2017v2017. A dedicated rig for World Machine Tiled Builds should be considered for this workflow.

The UE4 Export macro contains the Epic-suggested Slope options found in the World Machine -> UE4 Documentation.

This will create a grid of your maps creating files for each texture layer and heightmap layer to coordinate files ex. LandscapeMap_x0_y0, x0_y1, etc. until it has a file for each quadrant of a 4x4 grid. Select “Participate in Tiled Builds” for every Height/Bitmap Output Node you want grid heightmaps for.

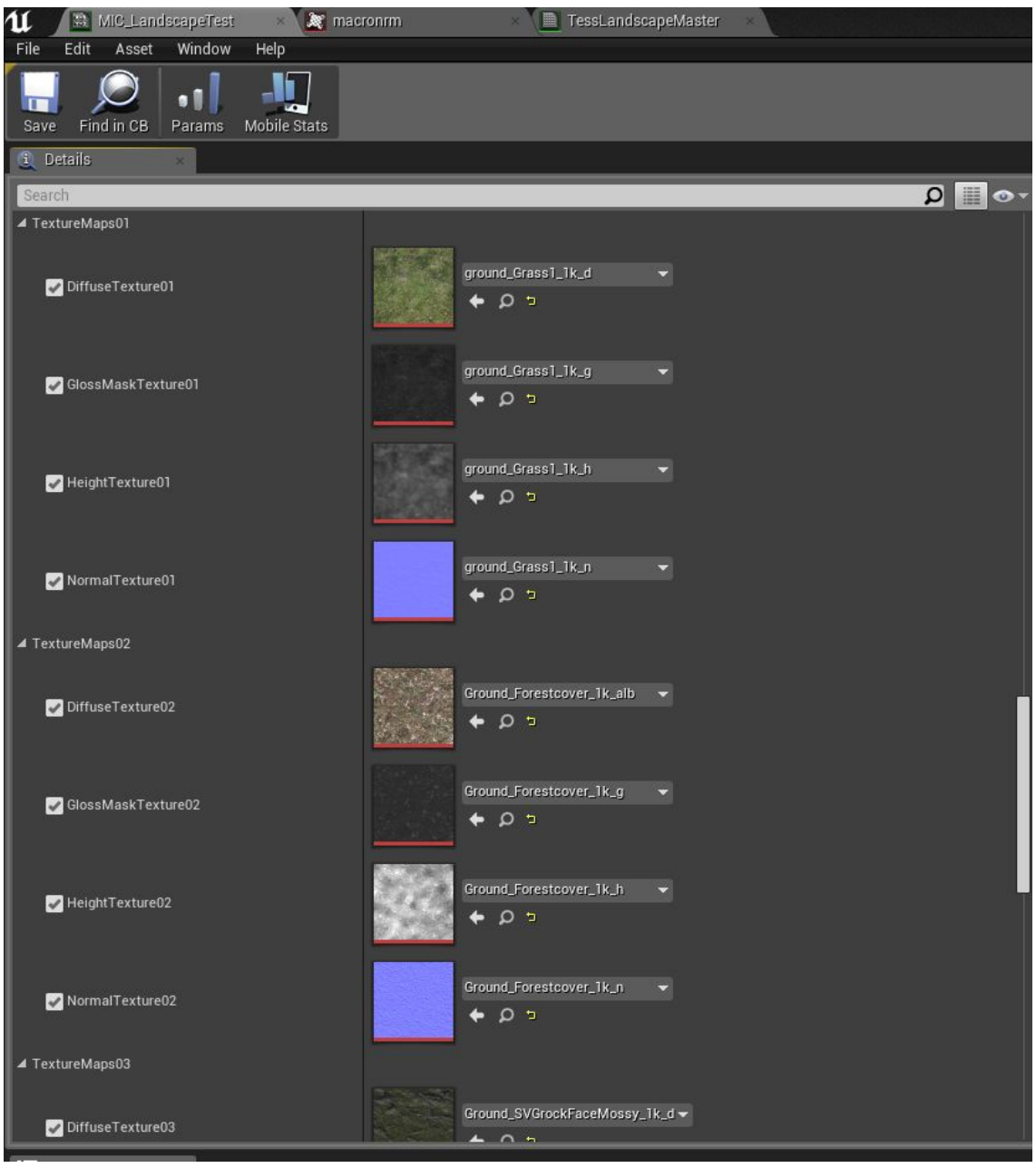


Step 2: Material/MIC Creation

I used the Advanced Tesselated Landscape Material from the Marketplace (It’s ~$10). **Note:** UE4 Landscape does not allow beyond 4 material layers (or 16 textures) to exist on a single landscape component by default (may be able to change that restriction in code). Quality switches would also be added to this material and changes to do camera-based tiling (far away textures tile larger).

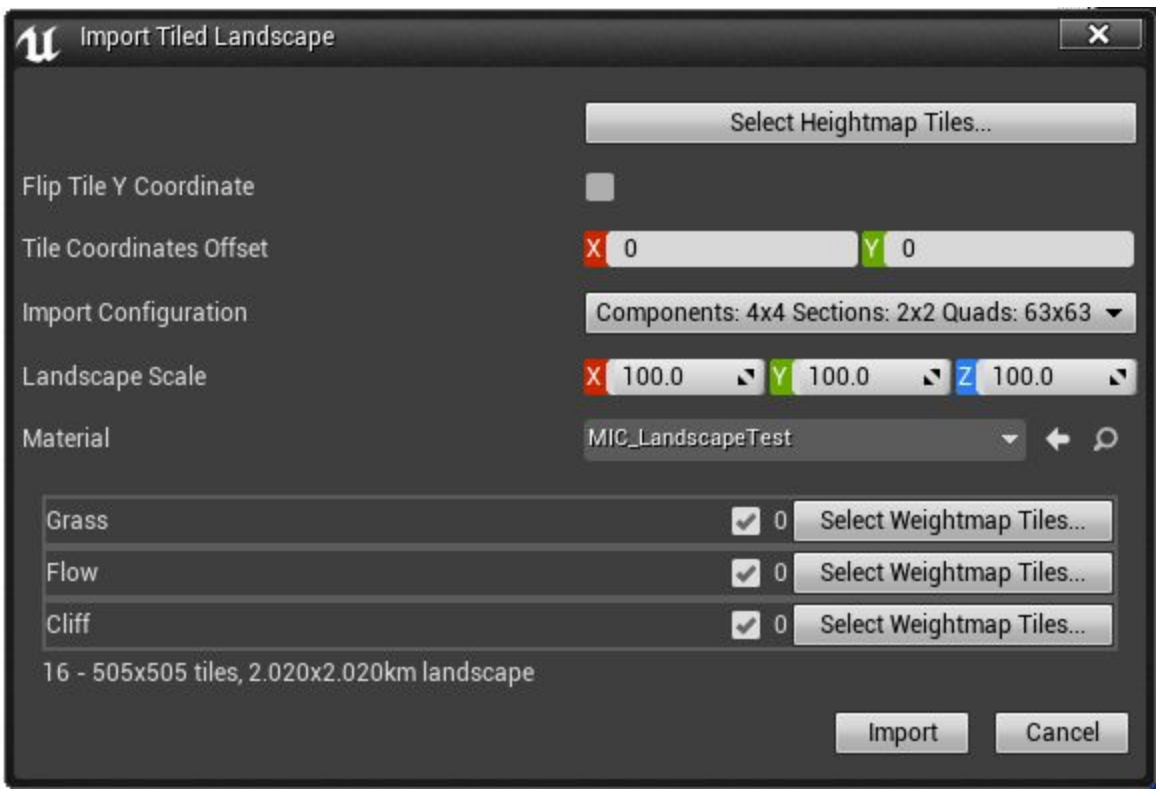
Each landscape layer is made up of Diffuse, Roughness, Height (Displacement) and Normal.

Import your Landscape texture and assign them to the Master Landscape Material MIC



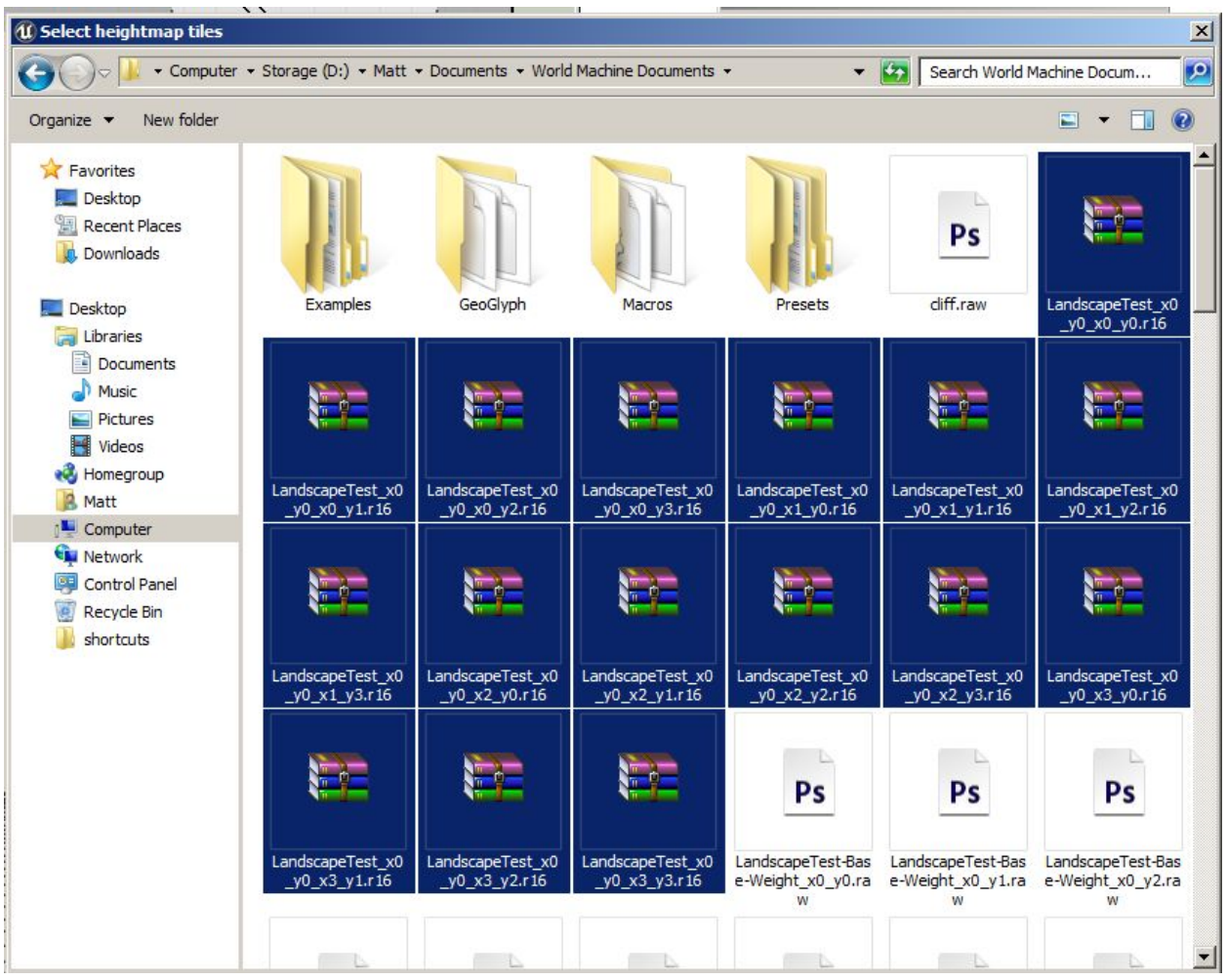
Step 3: Import Tiled Landscape

From the ***Levels*** tab, select the Import Tiled Landscape option. This will bring up the following prompt:

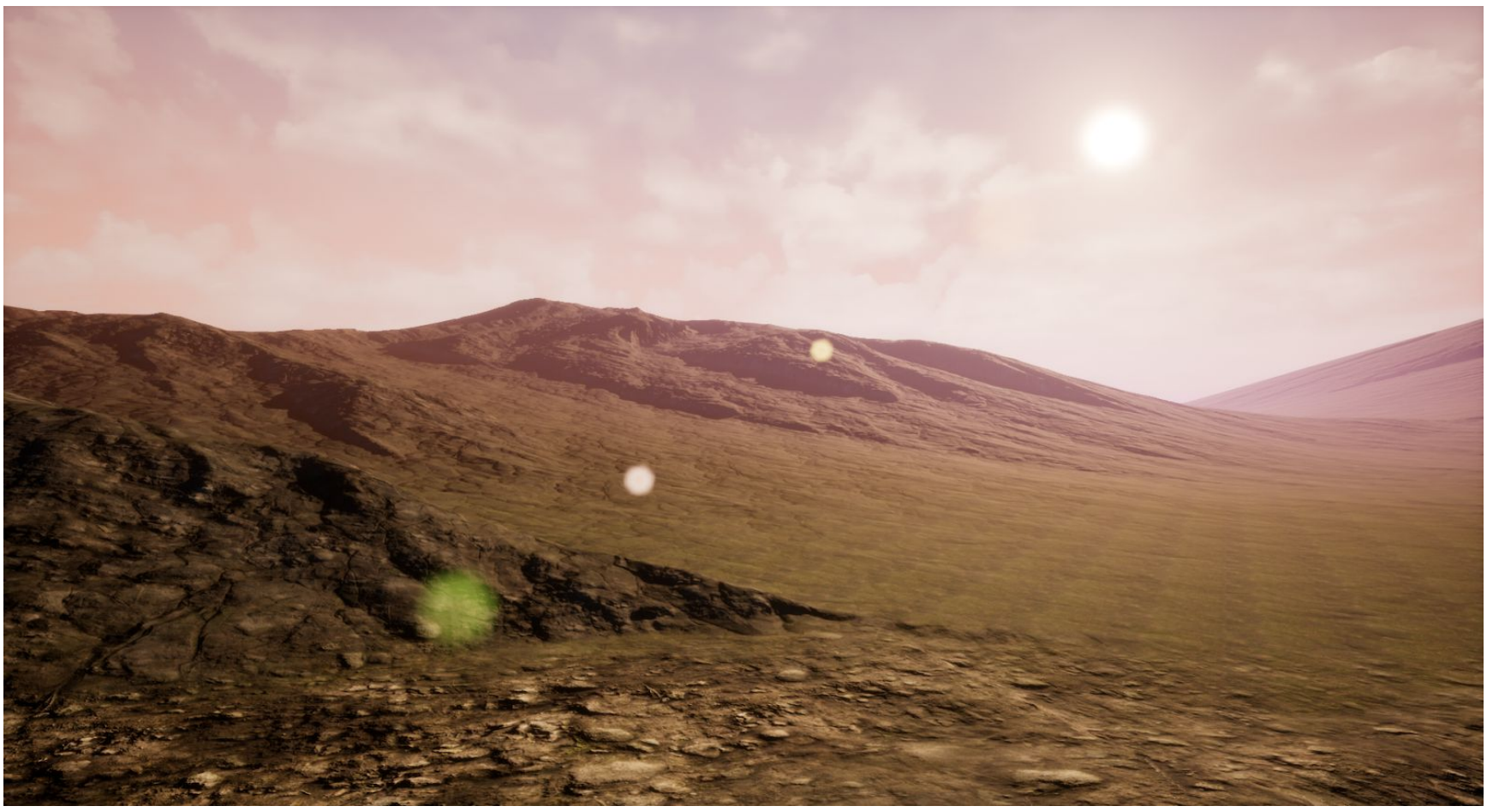


Click ***Select Heightmap Tiles…*** and select the 16 files below that match the heightmap name you set in World Machine (the x0_y0 and so on is added automatically by the Tiled Build in World Machine). Make sure you UNCHECK ***Flip Tile Y Coordinate*** since it was already turned on in World Machine.

Assign the MIC you made in Step 2. Now you've selected your 4 mask layers to export heightmaps for, select those 16 files per layer as well.



Step 4: Determine Pass 2 Iterations



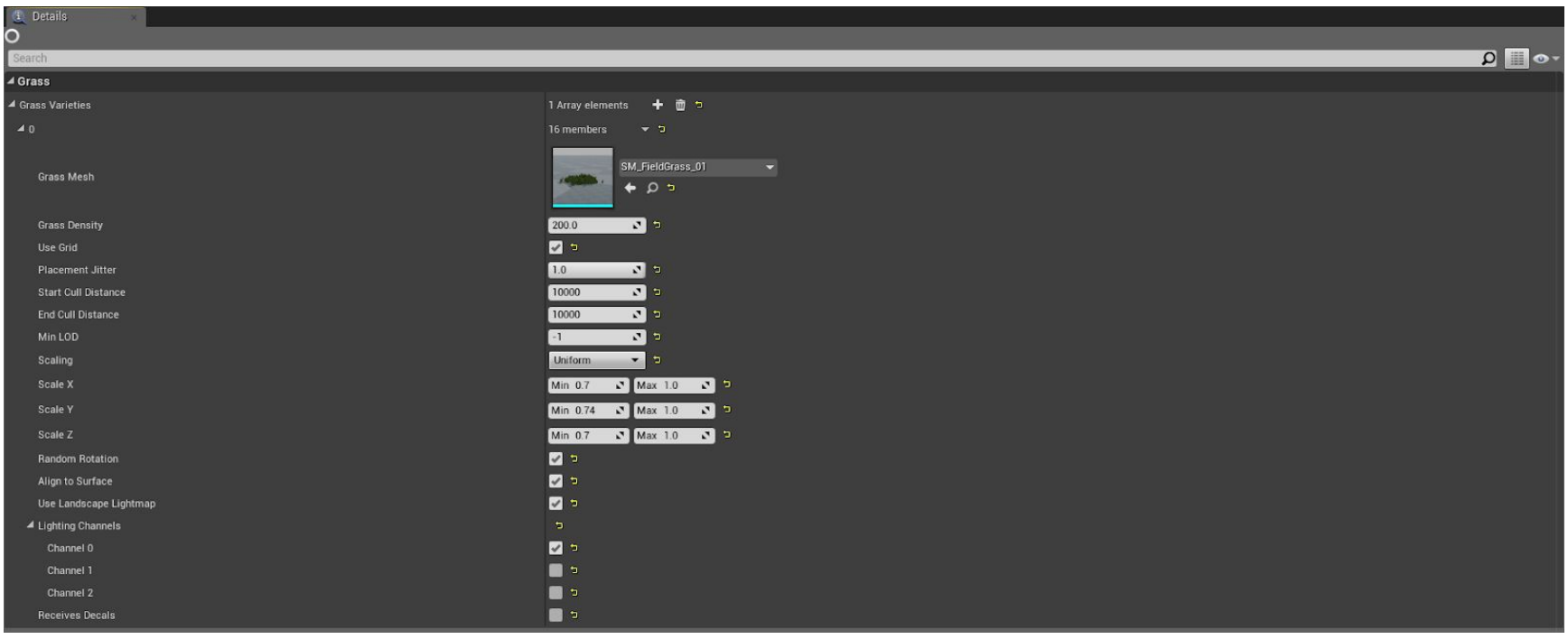
Bringing in the Landscape to a solid lighting scheme/post-processing volume and environment has a decent first pass result. There are a few problems that would need addressed and artist feedback:

Issues to consider for Pass 2

1. Camera-based tiling. Far textures need to be larger, closer needs to be tiled to player scale - no obvious/bad transitions.
    a. **Note: In actual implementation, this would already be solved in the Master material. Will be solved in Step 6.**
2. Artists may have better feedback on getting better masks out of World Machine.
    a. **Potential Workflow Drawback: Because this workflow is from a tiled world that is larger, it seems to possibly generate more generic masks. Adjusting weight maps is a slow iteration time and requires a new tiled build each time. Whereas exporting Splat/Weight maps for a single landscape is fast and easy to experiment with.**
    b. **The World Machine -> World Composition Import is a slow process. Worth it for 8k? (Remember my sample is a 2k by 2k world).**
3. Do the texture selections for the layers play nicely with the masks or hurt the masks?
4. What will foliage help hide or mask?
5. What gameplay areas do LDs want, and what ground paint layers are they missing?

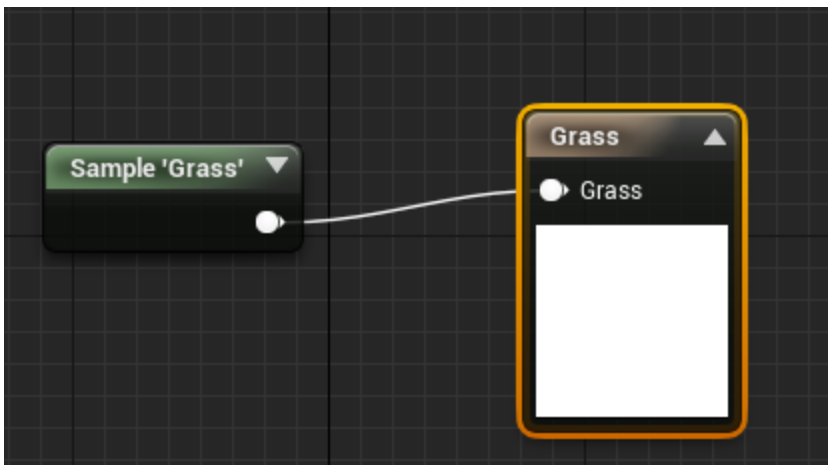
Step 5: Base Grass/Proc Foliage Pass

In this step we'll configure the grass system and overall proc foliage volumes to start to fill out the world.
In the content browser start by right clicking in a folder and go to Miscellaneous -> Landscape Grass Type. Choose a grass mesh of your liking.



Note: You can add more members to the Grass Varieties array if you had weeds/cattails/etc. To add variation. Since this is very dense, I'm going to change the Min LOD setting to 1 to use the first LOD layer. Best practice would be to make your base as low as possible because of the density.

In the Landscape Master Material we need to add a ***Landscape Grass Output*** and a ***Landscape Layer Sample***. Give the Landscape Layer Sample the name of the layer you want Grass, in this case *Grass*. Plug it into the Landscape Grass Output and hook up the Landscape Grass Type you just made.



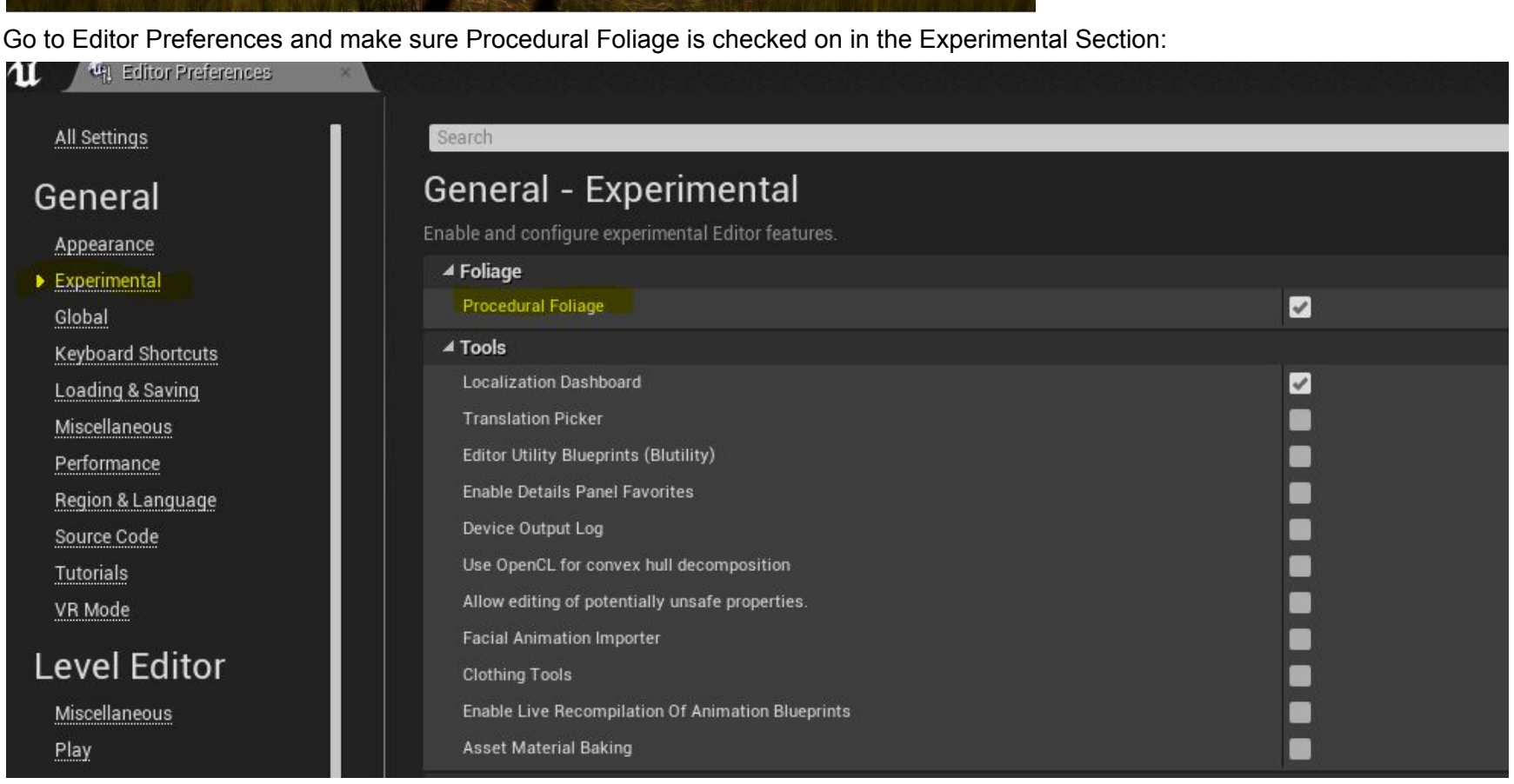
The grass isn't color matched with the landscape under it. I'm going to go into Unlit to color correct, then do a quick preview Light Bake to verify. There are a few options for this:

- Edit the diffuse texture (fast/efficient)
- Change MIC parameters to tune in material (easy, but if it can be done in texture it should be done in texture)
- Material solution - sample SceneColor from landscape (more expensive, but more accurate and flexible if we used tons of foliage types).

Drawbacks to Landscape Grass:

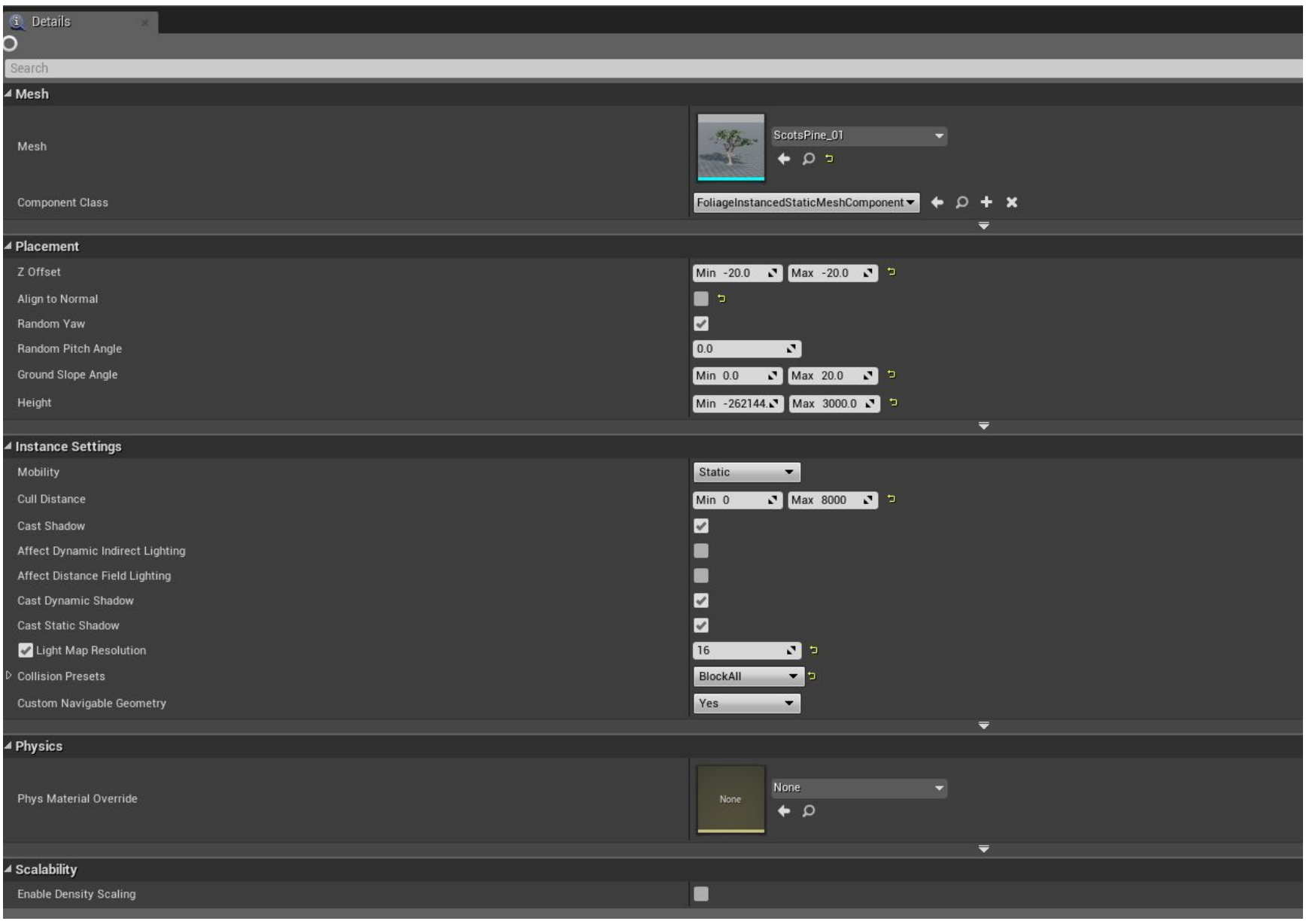
- Culling seems based on landscape component which lets you see the "grid blocks" at distance. We need shorter cull distance for these. Maybe occluders can solve partially to break up that seam, but I'm inclined to try grass within ProcFoliage volumes instead for more control over slope/height restrictions. Just paint layer restrictions doesn't seem sufficient.

Go to Editor Preferences and make sure Procedural Foliage is checked on in the Experimental Section:

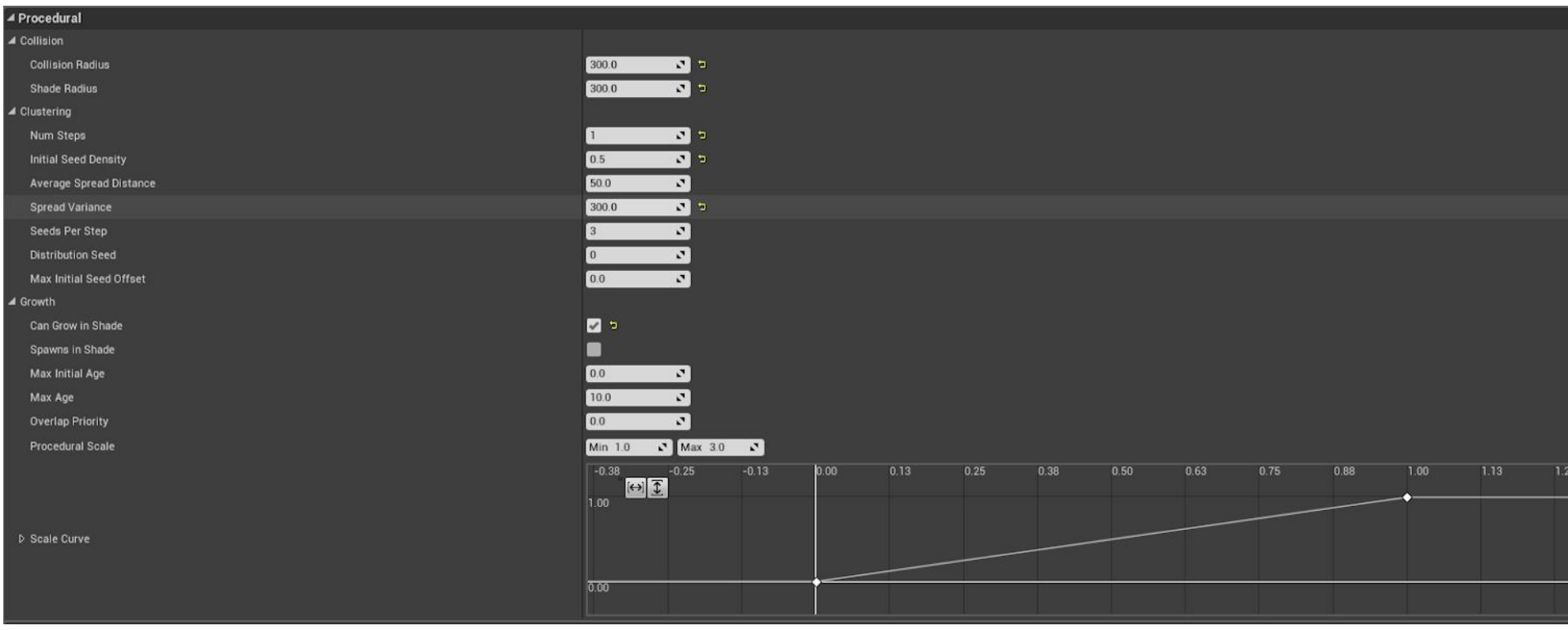


Create two Foliage Types by right-clicking in Content Browser - Miscellaneous -> Foliage Types. Choose two trees from your library to be your base trees. This volume will serve as a generic basis for coverage throughout the whole level, and we'll subtract out areas

in another step. We will tweak parameters to get good spacing and settings, I started with fairly aggressive slope and height restrictions:



I also added a Landscape layer to protect against trees far up on the cliffs.



The Foliage Type asset has a variety of awesome procedural generation options. Tweak these to your desire, but these settings gave me good spread without being too dense.

Now create a Procedural Foliage Spawner and add two Array entries under Foliage Types and add the two trees from the previous steps.

Guideline on Tree Authoring:
These trees should be more optimized and light weight than hero/hand-placed trees. LODs should be aggressive and try to minimize overdraw as much as possible between switches (sprites clipped as close to the atlas texture as possible).

Place a ProceduralFoliage Volume scaled to the size of the landscape. **Additional R&D Question:** Does volume size affect clustering? If so, split this volume into quadrants to match World Composition.

Hook up the Procedural Foliage Spawner to the volume. Click Re-Simulate in the volume's settings to see initial results. Tweak the Procedural section of the two tree Foliage types to your liking. These were my results after a preview light bake.

Step 6: 2nd Pass - Solve any Issues from Pass 1 after seeing world populated with things.

This step requires collaboration with Art Director/Environment/Design teams to call out any issues and revise. For the sake of Proof-of-concept we'll move forward, it looks pretty good.

In this phase we should also set up and configure streaming settings now that we have an idea of how the map is coming together.

Step 7: LD's subtract out gameplay areas with ProceduralFoliage Blocking volumes and Landscape painting.

During this stage, LD's should paint paths/gameplay related Landscape painting, add lighting/torches/lamps for navigation/gameplay and any high-level visual changes to the map.

Step 8: Rest of LD/World Building pipeline

Props passes, small/medium cover, etc. would be done here. Any carved out gameplay areas that were blocked out previously, would get art passes and so on.

Step 9: Polish/Bug Fixes/Performance Passes/Etc.

Workflow #1 Overall Assessment

Pros

- Potentially up to 8km x 8km worlds.
- World Composition's Streaming "Zones"
- With more R&D could potentially be almost the same as the more design/art flexible Workflow #2.

Cons

- Very slow tiled build times to generate height/weight maps for 8km.
- Pipeline flow to update masks/experiment on the artist's side is clunky and inefficient (If an artist wanted to test a sharper mask or a different output [say Ridgelines instead of Flow] they'd have to essentially repeat Steps 1 and 3 and wait hours to see if it worked.
- Optimization concerns with the size.
- Possibly inflexible for LDs/Gameplay b/c of World Composition/Tiling Build Iteration Times.

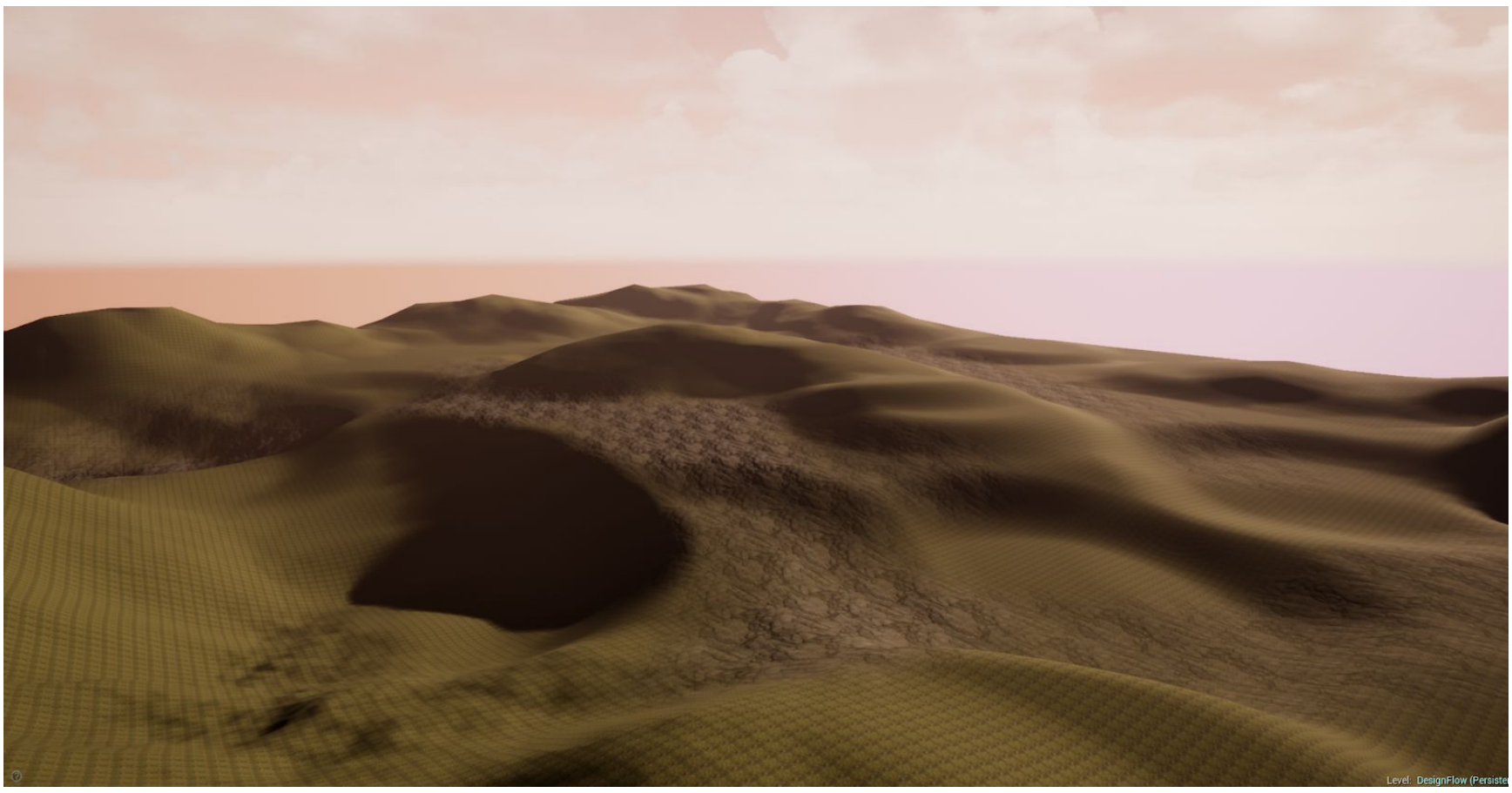
Conclusion:

If we want 8km x 8km gigantic worlds, this would be the way to do it. The streaming capabilities alleviate some performance concerns, but the workflow on the artist side is fairly slow and inflexible compared to other workflows. Commitment to a workflow of this nature would require more R&D to solve for some of the concerns. I don't think this would be a great way to go "out of the box" but we may be able to figure out systems to make it more feasible.

Potential Workflow #2: Design Driven - UE4 -> WM -> UE4

Step 1: LDs Create/Blockout Landscape in UE4 to design specifications.

With the right settings, straight landscape can go to 8k x 8k but they do not benefit from UE4's World Composition Streaming, however components that are completely obstructed would cull/occlude. For proof-of-concept I'll stick with a 2k x 2k landscape for World Machine build efficiency.



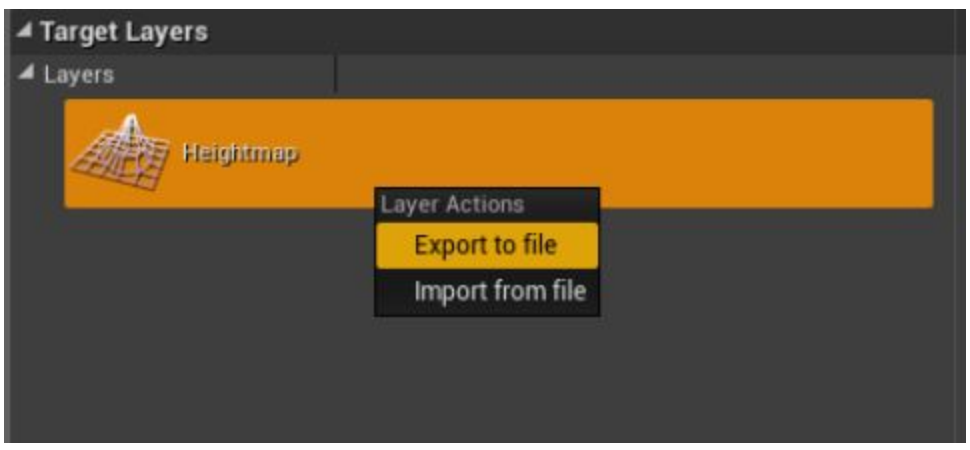
I'm using an existing Landscape MIC just to quickly color out basic paths and where gameplay will go.

Gameplay areas would be blocked out in BSP in this phase as well.

Step 2: Material/MIC Creation

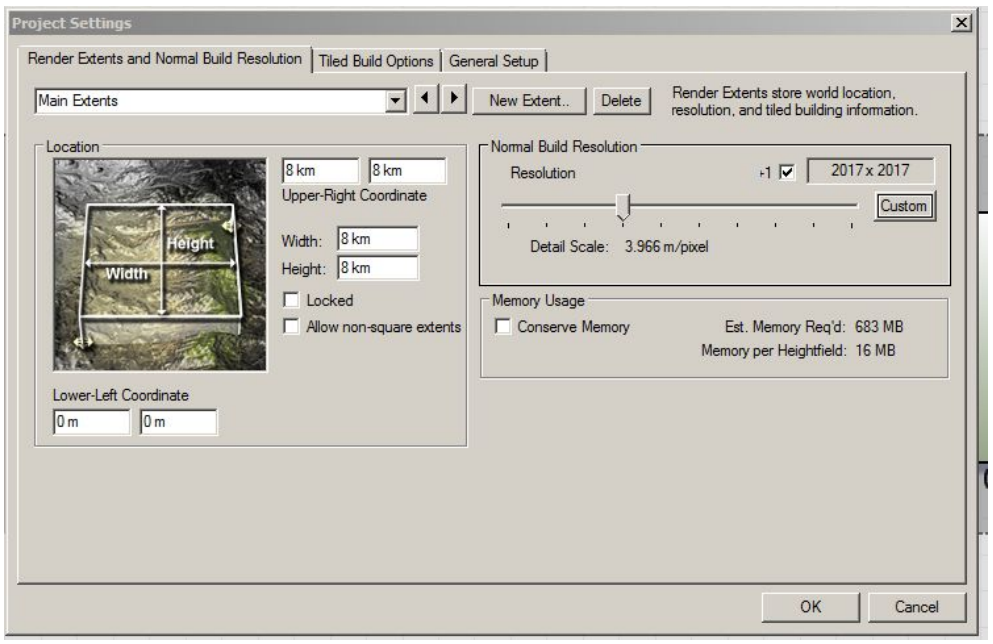
[See Step 2 of Workflow #1 - Same Steps]

Step 3: Export Landscape Heightmap to World Machine to Generate Detail + Masks

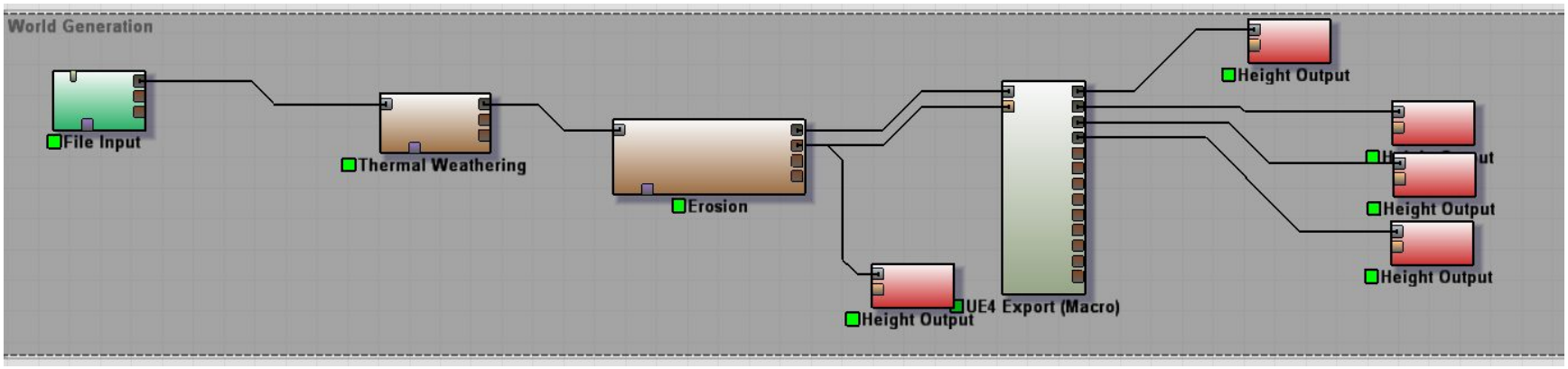


Launch World Machine and Add a File Input and load in the Heightmap you exported from UE4.

Make sure your project settings match the resolution of your landscape heightmap:



Add a node network of your liking. I'm going to use the UE4 Layers macro again for exporting weight maps for our paint layers.

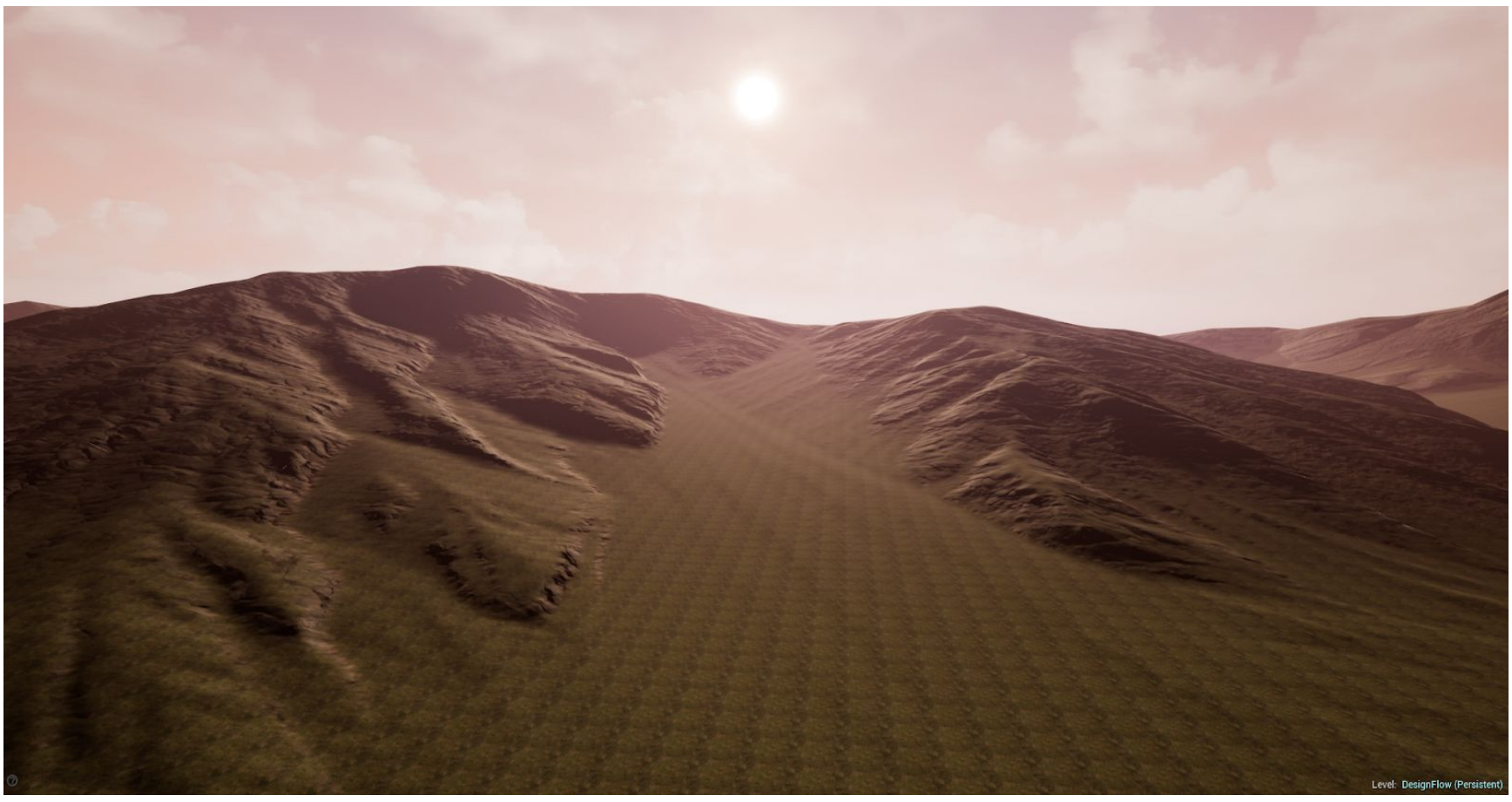


Important Note: UE4 seems to like .png format for paint layer weightmaps so make sure you output to that format. The landscape geometry heightmap should stay RAW16 (.r16)

This part of the workflow is quite flexible because you could use different macros to test different height outputs for paint layers easier. I didn't like the way the UE4 Export handled some of the slopes, so I exported the flow map directly to break up the cliff textures more.

Step 4: Re-import Height maps for Landscape + Paint Layers + First Art Pass Gameplay Areas

The order you import paint layer height maps matters. The last one you imported will always be "on top." So experiment with import ordering until you get the effect you want. I used the Lo & Hi slopes for Cliff and Grass, and the Flow Map for Flow (imagine that).



World Machine gave us some pretty good results. Grass tiling would be fixed by Camera distance-based tiling and covered by procedural grass in the fully integrated Landscape material.

Step 5: Base Grass/Proc Foliage Pass

[See Step 5 of Workflow #1 - Same Steps]

Results from this step:

Step 6: 2nd Pass Landscape - Resolve any issues from above steps.

[See Step 6 of Workflow #1 - Same Steps]

Step 7: Props Pass + Small/Medium Cover + etc.

Add small rocks, cover, props, etc. to the world. Configure with established performance parameters (draw distances, etc.)

Step 8: Rest of LD/World Building pipeline

Step 9: Polish/Bug Fixes/Performance Passes/Etc.

Workflow #2 Overall Assessment

Pros

- Limitless design and art flexibility.
- Fast iteration times on landscape texturing/masking/world machine steps.
- Easier to optimize at a smaller size (2k x 2k)

Cons

- Supports 8k x 8k BUT doesn't benefit from World Comp Streaming, only landscape component culling. (Might not matter).
-

Potential Workflow #3: Hybrid Workflow

We could take the benefits of both systems and attempt to combine them in some manner, rough idea below:

1. Import 4x4 grid of straight black heightmaps to give a base terrain.
2. Designers rough out this landscape the same as the 2nd workflow.
3. Generate masks and World Machine portions potentially an issue here since exporting maps from World Comp isn't a thing since it generates multiple landscapes. It would essentially be Step 2 of Workflow 2 times 4 on an 8k x8k level.

Conclusion on Workflow Options

In conclusion, I don't see too many cases where a 2k x 2k or 4k x 4k landscape couldn't offer the same gameplay options as an 8k x 8k world. Even with some hybrid options, I think UE4's World Composition system is a bit clunky, especially with the World Machine pipeline to get into it. I think Workflow #2 allows the most flexibility, creativity and exploration with its fast iteration times - leveraging World Machine to give it detail and quality, but giving design and art full control without long builds and other hurdles.

Addendum: Additional Suggestions & Considerations

Known Systems to Consider

General Systems

- World Machine's Tiled Builds/Procedural Detail Generation (Former for Workflow #1, Latter for Workflow #2)
- World Composition in UE4 - Hierarchy-Based Streaming Rules (Client-Side or Server-Side?)
- Advanced Tesselated Landscape Material (or similar Master Landscape Material, we can make a better one easy)
- Procedural Foliage & Grass Systems in UE4

Performance/Optimization Systems

- SimplyGon should be integrated to our UE4 engine build. Too easy and powerful, especially for large world games, even more so if we want vistas.
- Merge Actors for combining close area sets. Can be assembled modularly but combined to reduce draws to draw budgets later. The SimplyGon option within Merge Actors enables Material Baking that replaces the terrible base engine redundant material atlasing.
- Hierarchical LOD is an important performance system. Distance clustering to one mesh will be key for vistas and longer draw distances. Even for shorter, with such large worlds it's easy enough to use to take advantage of. H-LOD has the flexibility to procedurally generate clusters or allow volume-based or exclusion-based clusters.
- Base Engine culling/occlusion.

Additional Work to Consider

Systems/Tools

- Render farm for SwarmAgent Light Bakes. With giant landscapes and lots of foliage, we'll need faster iteration time on light bakes.
- Dynamic Time of Day/Dynamic Lighting Scenarios
    - See level of effort to create.
    - See if possible to hit performance target on gigantic maps.
    - Plenty of artist flexibility.
- Additional Streaming Level Support
    - Make sure World Composition's Streaming is sufficient. See if additional logic/rules are necessary.
    - Offer more control/soft transitions if necessary.
- Smart Run-Time Optimizations
    - Adaptive Cull Distances/Fog Start
        - Scale cull distance percentage during heavy action/battle to allow for higher FX budgets
        - Could also get further in clear periods of inaction.
        - Fog starting point could interpolate in/out based on circumstances (client-side) to hide cull changes.
    - Texture Pool control
        - If Texture Streaming system is too visually abrasive out-of-box, write additional logic to solve. Drop mip levels based on distance/screen pixel size w/ texture group priorities.
    - Dynamic Resolution Scaling
        - If PS4 encounters dense areas or battle, can scale down resolution within a clamped floor, temporarily until GPU/Draw can recover.
    - Distance-based Quality Switches
        - See if we can extend Quality Switches to have a toggle for distance based.
        - Being able to drop normals/detail normals at distance or other channels without Camera Position vs. World Position calculations each frame would potentially have big performance gains.
        - Material Quality Bias settings on Static Meshes?
- Automated performance stats system.
    - Would run nightly to show trends over time.
    - Should allow Tech Art to setup tagged cameras per world to cycle through and capture the following performance stats that are output to .csv:
        - GPU/Draw/Game Thread Times, Mesh Draw Calls, Particle Draw Calls, Texture Pool Usage
    - CSVs can be processed through batch scripts -> Google Drive -> Google Script for automatic reporting, or similar architecture through another language.
    - Allows all teams to see map content swings and keep an eye on map budgets.
- Commandlet to track FPS averages and minimums per match per player -> output to CSV.

Workflows/Content

- Monthly or semi-monthly performance playtests.
    - Playtests will gather memory data and FPS pulse checks on new maps and content with reporting following.
    - If we go to PC too, this will be especially important. Try to get spectrum of machines and hardware, especially our exact min spec, to catch unforeseen issues and test Min settings visuals in an accurate match environment.
- Performance meetings/Tech Art involvement in LD/Env Art/FX pre-production to call out performance concerns and possible solutions. Establish budget checkpoints.

- Exceptional master materials for landscapes, make sure it’s properly quality switched and art teams are fine with what drops.
- Master materials that interplay with landscapes w/ good ground blends (Dither TemporalAA for simple setups and possible tri-planar/normal correction for higher priority art).

Documentation

- Map draw call budgets and standards as quick as possible. Generally for PS4 spec keeping draws <800-850 in any given location is an easy starting point for PS4 Scalability.
- Guidelines for using Merge Actors/H-Lod -> Make sure LD/Env Art teams understand occlusion bounds Do’s and Don’ts.
- Performance targets and budgets for FX, Characters, Animation, etc.
- Tutorial Documents for any non-standard or altered world construction pipelines.

Possible Third Party Tools

GeoGlyph: It’s a third-party Macro Library/Wrapper for World Machine. It offers fast iteration macros to handle things like splat map distribution, more realistic macros for different biomes and generally gives higher quality results in faster time. It seems to not play nicely with tiled builds so it may only have use for Workflow #2 **Cost: $240/seat**

SimplyGon: Near-standard optimization tool for models. Too good at what it does to not have in the pipeline, especially for these types of games. Assuming we already have, but just in case.