Tab 1

TEARDOWN

A Guide to Terrain and Vox-Model Creation

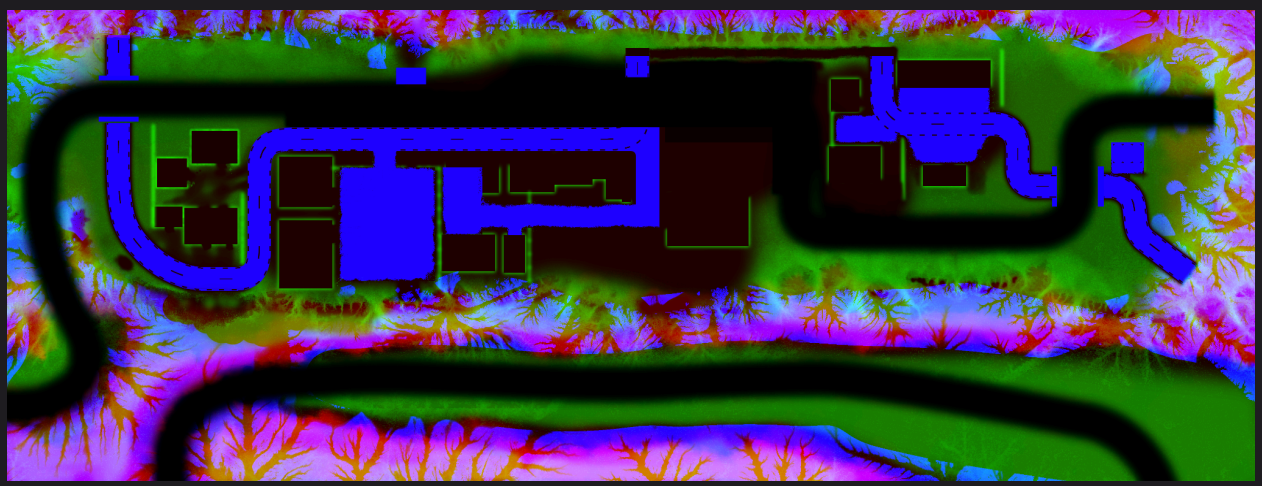
When creating a new level in **Teardown**, it might be overwhelming with all the options for creating terrain. In this document, there is information on how to navigate the different tools used for terrain creation.

Using Ground.lua

The oldest and perhaps most well-known tool for terrain generation is the **ground.lua** Voxscript. This script is added to a scene by inserting a **Voxscript node** into the **Scene Explorer**, typically located on the left side of the screen. The **ground.lua** script reads heightmap data from a PNG file and generates terrain directly within the editor.

In **Voxtool**, there is also a tab labeled **"ground.lua"**, which will be explained later in this section.

The **ground.lua** script reads three **RGB channels**, each determining a different material for the terrain. Many Teardown levels utilize this voxscript, and below is an example of a heightmap for the level Frustrum.



- The **Red channel** represents the **heightmap** and determines the elevation of the terrain. The brighter the red is, the higher the elevation.
- The **Green channel** is for **grass** and controls where grass will appear.

- The **Blue channel** is for **roads and road markings**.

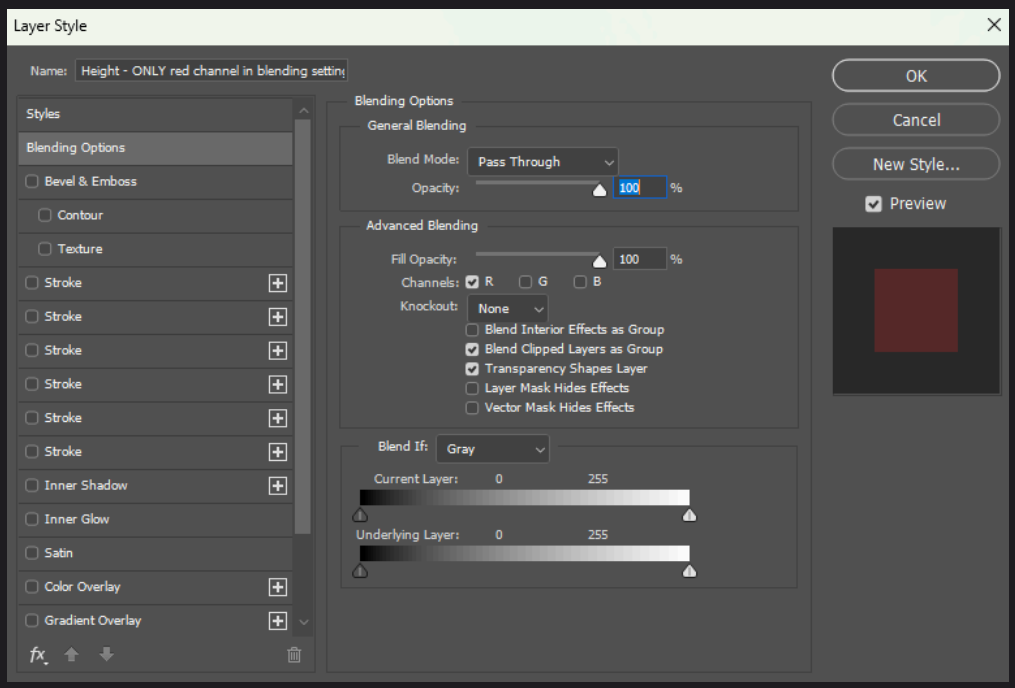
Before opening **Voxtool**, it is important to understand how to create a heightmap. Any image editing software similar to Photoshop or GIMP can be used for this purpose. This guide will demonstrate the process using Photoshop. A base template has also been provided for convenience. Download it here. The template includes example layers within each folder. These are provided purely for reference and to illustrate how the setup works, you don't need to use them in your final version.

Make sure the image mode is set to **16-bit mode** to ensure the smoothest terrain, as 8-bit can result in some jagged slopes. In Photoshop, this is easy, just go to **Image > Mode**, and select **16 Bits/Channel** from the list of options.

Decide what type of scene to create. For this tutorial, a beach scene will be used to illustrate the different ways to work with **Voxtool**. The focus of this document will remain on the beach scene throughout.

To organize heightmap layers properly, create **three folders** in Photoshop. Set the **blending settings** for each folder so that they only affect **one** of the three **RGB channels, if you have the photoshop template downloaded this will already be set up.**

To set a folder/layer to only allow one channel, simply double click it in the layers tab and the menu below will open:

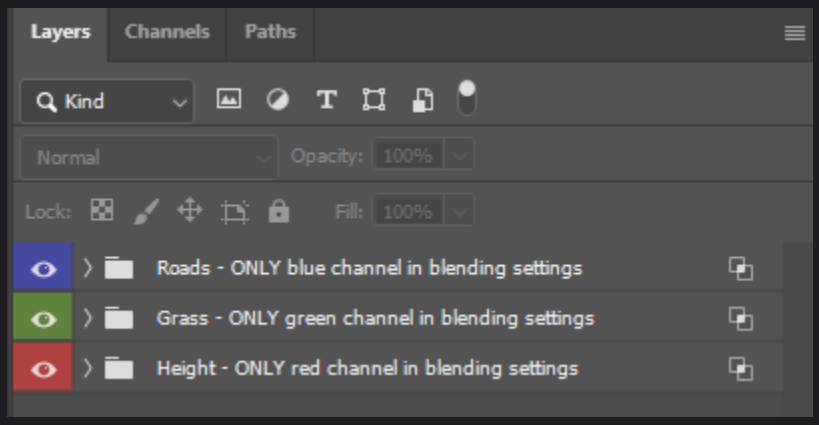


Here is an example of what it looks like when only the red channel is active (this is the height folder).

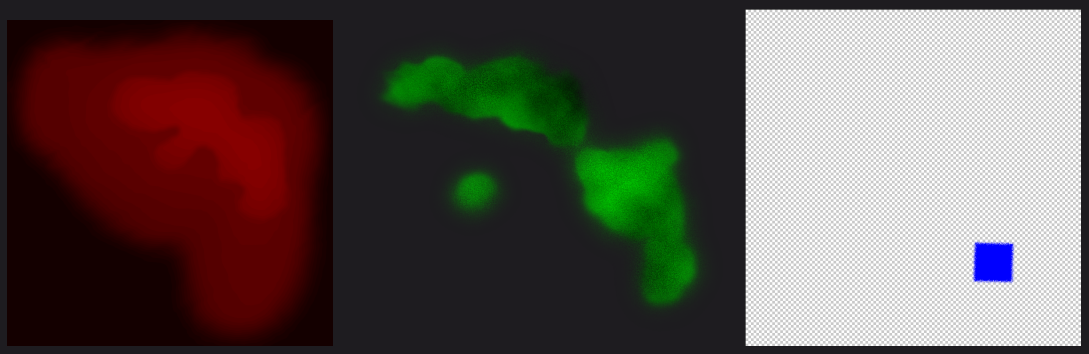
You will need:

- **One folder for Red** (heightmap)
- **One for Green** (grass)
- **One for Blue** (roads and markings)

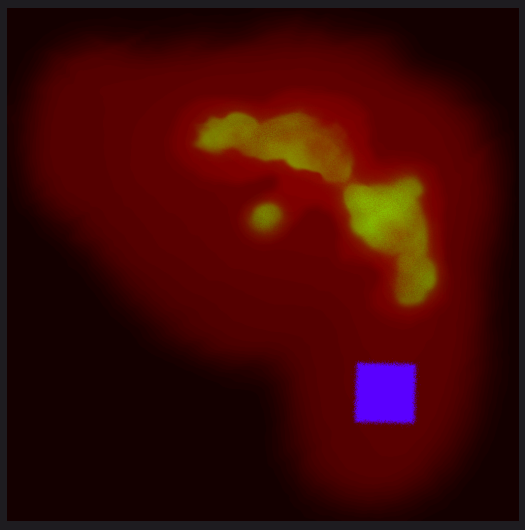
This setup will help keep the project structured and ensure accurate terrain generation. Assuming the setup has been downloaded, the following layout should be visible in the lower right corner of the screen:



Each group's blending is configured to isolate a specific color channel. As mentioned in the beginning, **ground.lua** uses different color channels for different materials. So grouping them like this makes it easy to paint out one material at the time. Each color channel has a range of 256 values (0–255). In the red channel (used for the heightmap), a value of 255 represents the highest possible terrain, while 0 represents no terrain at all. For the green channel, the green value represents the height and thickness of the grass. The blue channel functions a bit differently, it only uses three distinct values, each corresponding to a specific type of road material.



From left to right: heightmap (red channel), grassmap (green channel), road (blue channel). Combined they will create the complete heightmap:



Once all the layers have been painted to the desired look, the result can be exported as a PNG and tested in the level. To use the heightmap in a level, follow these steps:

1. Add a **Voxscript** to the level.
2. Select **ground.lua** as the Voxscript.
3. Insert the heightmap-PNG into ‘file’ under **properties**.
4. Set the **scale to 256** if one brightness value should equal one voxel in height in the game.

   For example, if there is a flat area with the RGB value (red = 20, green = 0, blue = 0) and the intention is for red = 20 to represent **20 voxels** in height, the scale should be **256**. If the goal is for that same red = 20 value to represent **10 voxels** in height, then the scale should be set to **128**.

Great! Now there should be a piece of terrain that likely has some **brown dirt, green grass, and even a bit of road**. The result could look something like this:

Earlier, it was mentioned that the focus of this tutorial is a tropical beach scene. However, when using the **ground.lua** script from the editor, terrain can only be generated using the preset materials:

- **Brown dirt** for ground
- **Green grass**
- **Grey concrete** for roads

This means that creating sand or adjusting the grass color is not possible when using **ground.lua** within the editor.

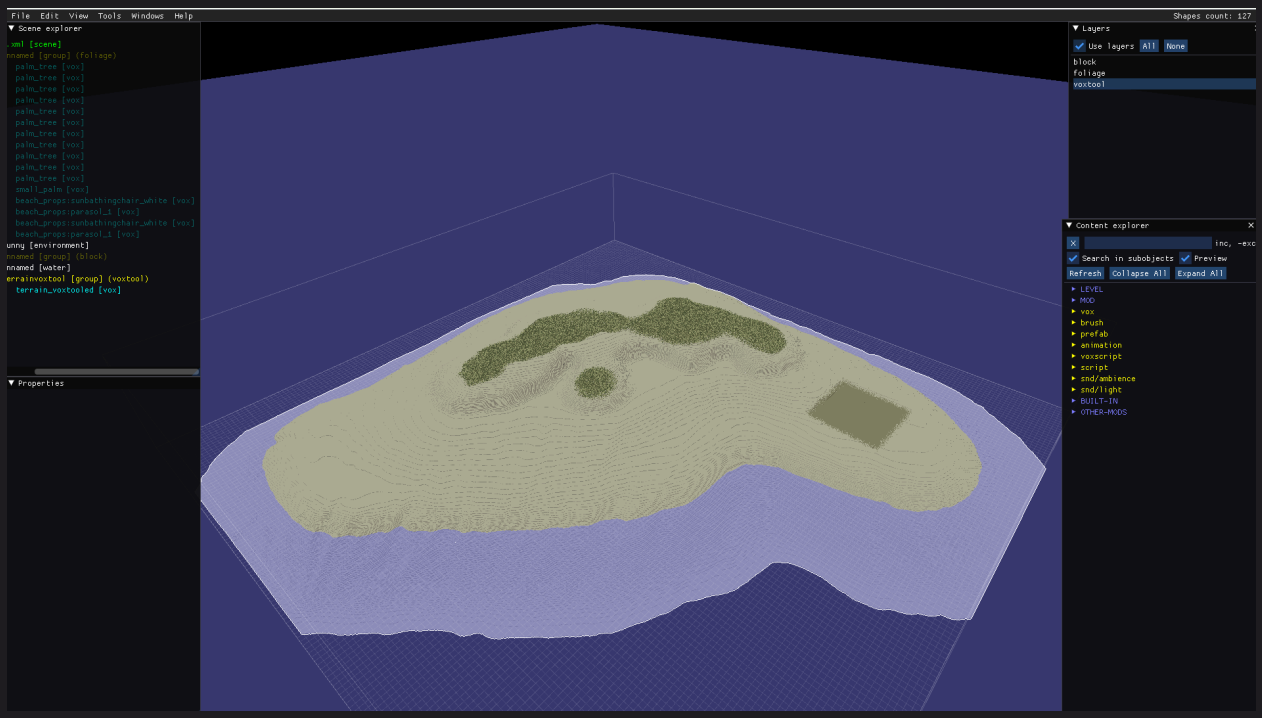
This is where the **Ground.lua tab** in [Voxtool] comes in.

- In Voxtool, **input the same heightmap PNG used before**.
- It is also possible to modify the colors of the predefined materials in **ground.lua**.

- However, the physical properties of these materials cannot be changed—for example, roads will always use the **masonry** material and are therefore indestructible with a sledgehammer. This should be kept in mind when making modifications.

By tweaking these values, I can adjust the terrain's appearance to be more in line with the beach scene I had imagined.

The final output will be a **.vox file**, simply add it to the level, and **voilà!** There should now be terrain with the new chosen colors.



Converting 3D model props to voxel models with Voxtool

This section covers how to use **Voxtool** to convert 3D models into vox-files that can be used to decorate the level.

For this example, free assets from **PlantFactory** were used to convert a 3D-model palm tree into a vox-file.

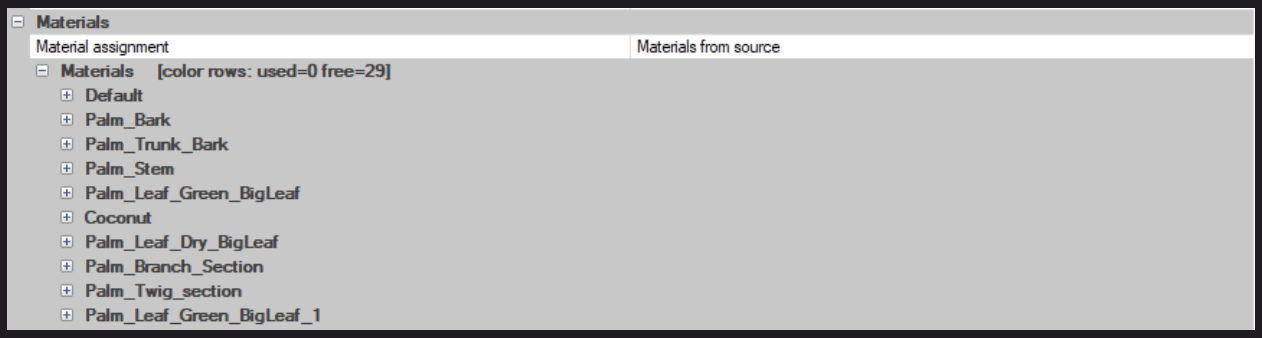
Any 3D model can be used for this process, or a custom model can be created. In this example, a palm tree was selected to fit the beach level theme. If the model is not in **OBJ** or **FBX** format, it must first be converted so Voxtool can use it. This can be done using a variety of programs. For this example we'll be using the free software **Blender**. You can download it here. However, any other 3D modeling software capable of exporting to OBJ or FBX can also be used.

Preparing models for use in Voxtool using Blender

First, the model needs to be imported into **Blender**. This is done by selecting **File > Import**, then choosing the file format that matches the model. Locate the model in the file explorer and click **Import**.

Above is the 3D model selected for the process. The next step is to export it as an **OBJ** or **FBX** file. To export the model, navigate to **File > Export > Wavefront (.obj)** or **FBX (.fbx)**. Voxtool supports both file formats. If uncertain which one to choose, **FBX** is recommended.

Once the model has been exported, it is ready to be imported into **Voxtool**. After importing, the **Materials** section will expand, displaying all available materials similar to this:

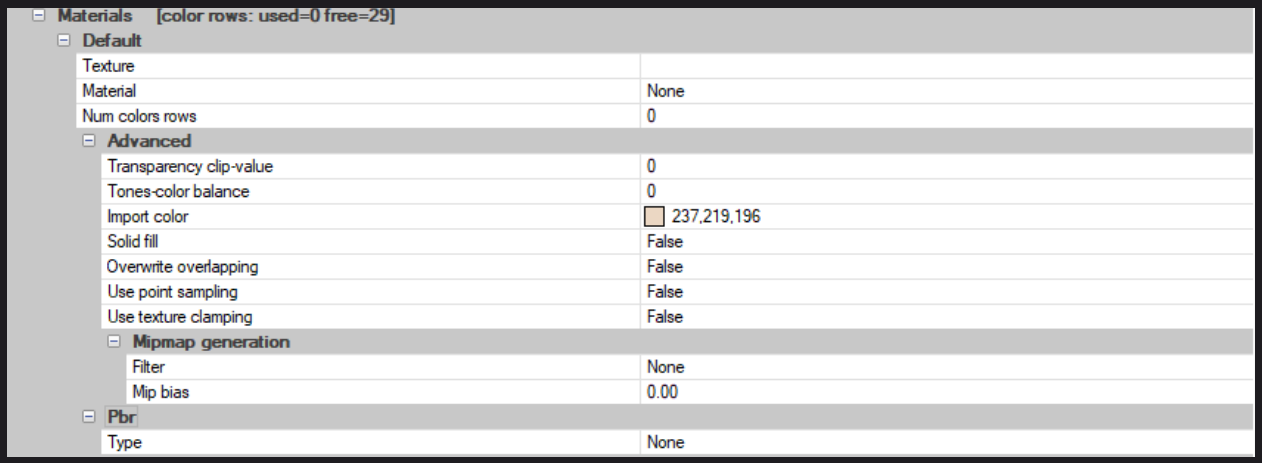


Setting Up Materials

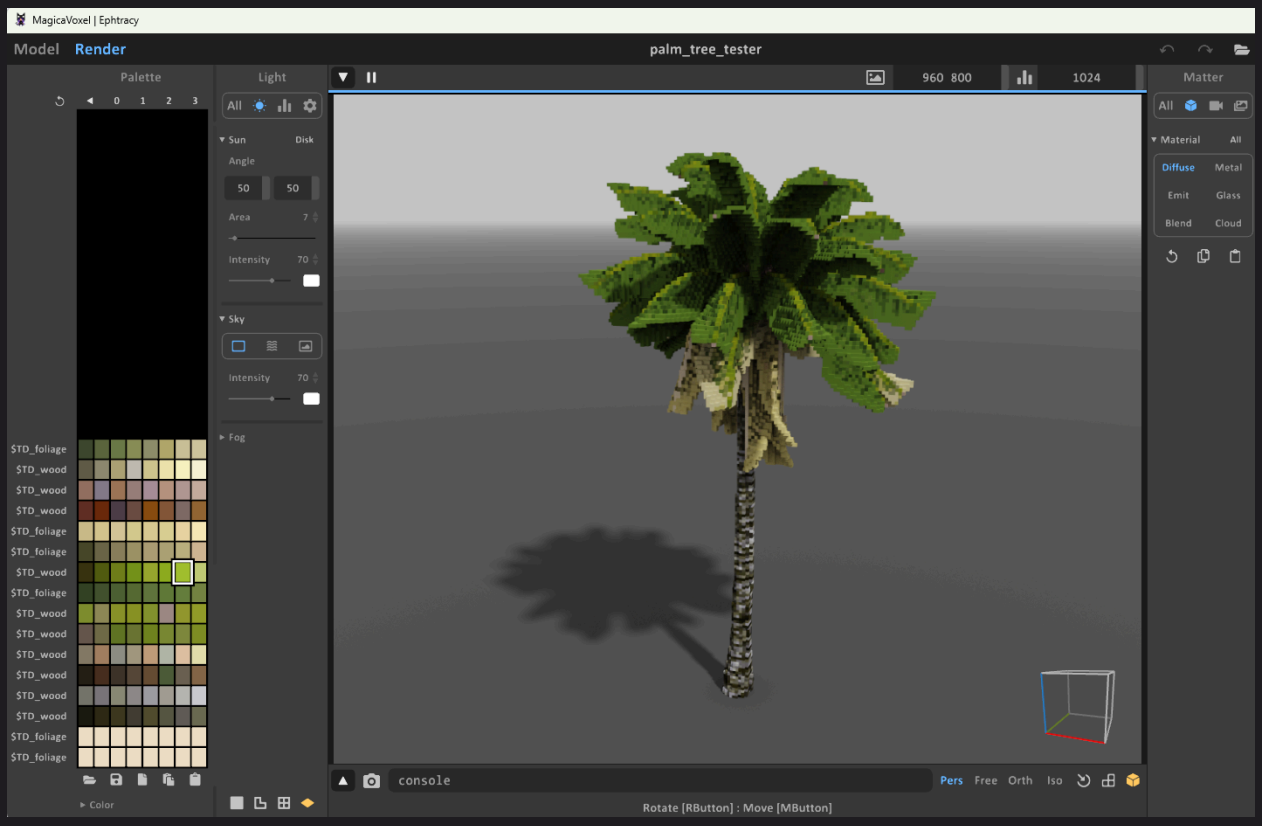
Here, the materials assigned to the model are displayed. By opening the material settings, several options become available.

It's important to:

- Choose materials that match the mesh's purpose. For example, **Palm_Bark** should likely be assigned the **Wood** material.
- Select how many color rows to allocate for each material. Each color row corresponds to a row in MagicaVoxel's palette. For example, choosing 3 color rows means the selected material will use 3 consecutive rows of colors from the palette. The more rows that's reserved to a material, the more colors.
- For more in depth information about each option in Voxtool, refer to the [Voxtool documentation].



NOTE: This is what the material section looks like with no input. If a .vox file is generated with both the material and color rows left empty like this, the resulting MagicaVoxel file will be empty.

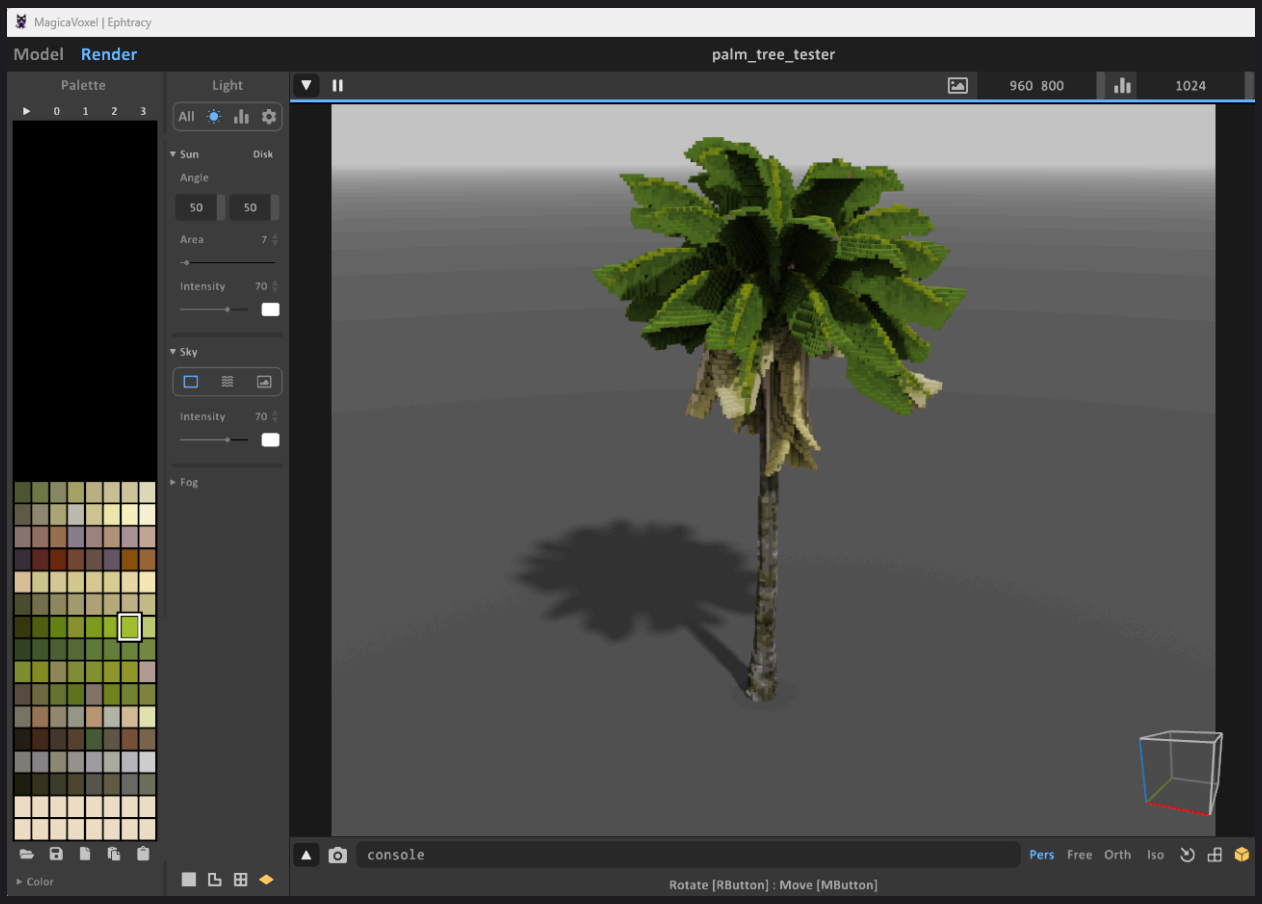


Here's the result with just materials and color rows assigned. On the left, the color rows in MagicaVoxel are divided based on the materials chosen and how many rows were specified (for example, the trunk should be assigned as Wood with two color rows, etc.).

Refining the Model with Filters

The first result looked **a bit noisy**, so let's fix that with **VoxTool** by using **noise filters**.

Under the **Mipmap Generation** tab, there's an option to apply a filter. Play around with different filters to see which one works best for your model. The **Lanczos5** filter was used here, which nicely smoothed out the texture.

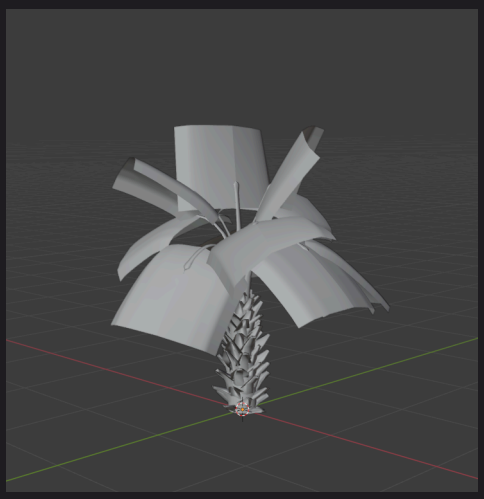


Vox-model with **Lanczos5** noise filtered applied.

Finalizing the Model

With the **filter applied**, the model is now ready to be added to the **beach level**!

Much better! Let's add another tree variation to explore a different example. I'd like to include this small palm tree (pictured below). When checking the model in Blender, notice how it needs **transparency** to properly render the leaves. Without it, the voxel model will end up looking more like the image on the right.



Mexican Fan Palm Leaf	
Texture	
Material	**Foliage**
Num colors rows	**2**
Advanced	
Transparency clip-value	**100**
Tones-color balance	0
Import color	204,204,204
Solid fill	False
Overwrite overlapping	False
Use point sampling	False
Use texture clamping	False
Mipmap generation	
Filter	**Lanczos5**
Mip bias	0.00

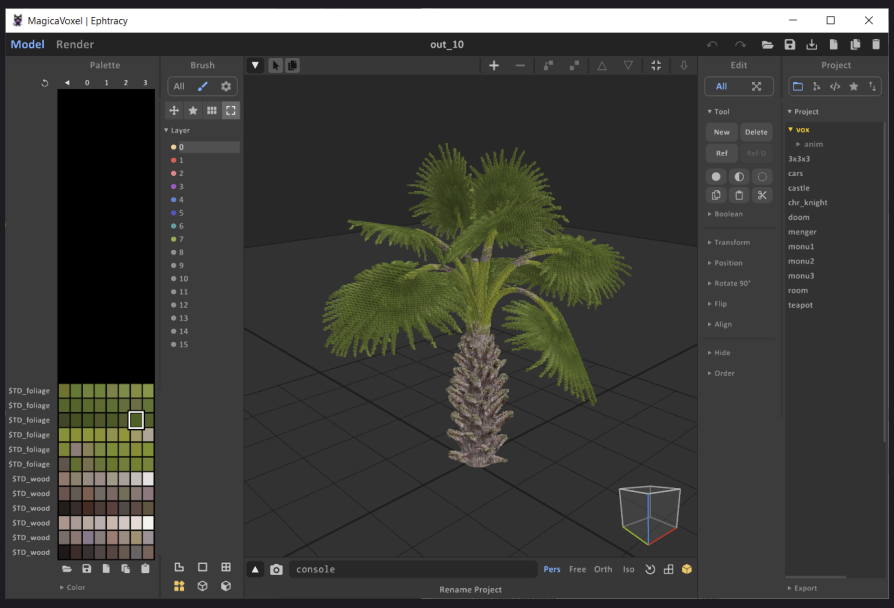
Under **Advanced** there is an option for **Transparency clip-value**.

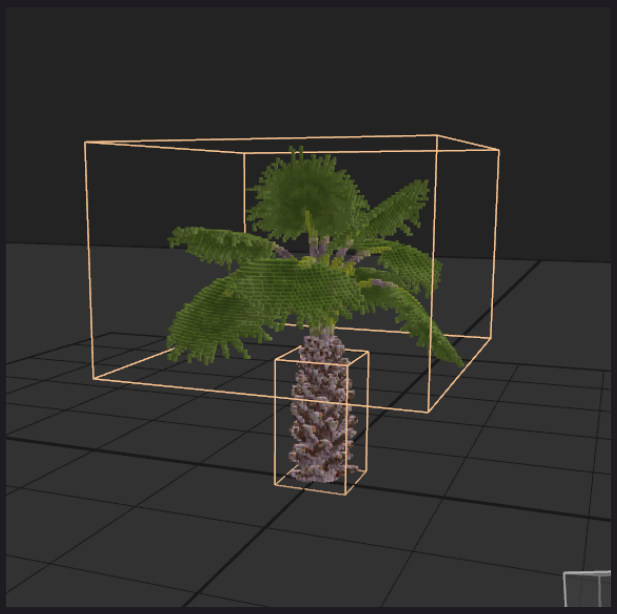
The **Transparency Clip-Value** is like a cut-off point for deciding what parts of a texture should be **see-through** (transparent) and what should be **solid** (opaque).

- It uses a number between **0 and 255**.
- If set to **100**, any part of the texture with transparency below 100 will disappear (become fully transparent).
- Anything **above 100** will stay visible (opaque). As an example, this palm tree used a transparency clip value of 100.

When generating vox models it's usually a good idea to set the scale to a higher value to spot any errors that are harder to see if it's smaller. Below is an image of the result of the model with scale set to 10. With this scale, it's clear to see that the model looks clean and that the settings used in Voxtool are good for this particular model, or if there were errors it would be easier to spot if the model was smaller.

This model has a significant amount of volume, which also means a lot of empty space that can cause lag when the model is broken. To optimize it, it is best to break it down into multiple shapes. Below, the model is scaled down again to its intended scale with the optimization target count set to 2. In voxtool under **Optimization,** there is an option called **target count** where you can set how many shapes you would like the model to have. Setting it to 2, will divide it into 2 shapes.





Here is the optimized model with target count set to 2 in Magicavoxel, and also the finished product in the editor!

At this point, assuming the steps and pipeline outlined in this document have been followed, there should be a solid foundation for creating terrain and for converting complex 3D models into vox-files for use in levels.

From this point forward in the document the options for creating terrain are going to be more complex and less straightforward.

Using Heightmap

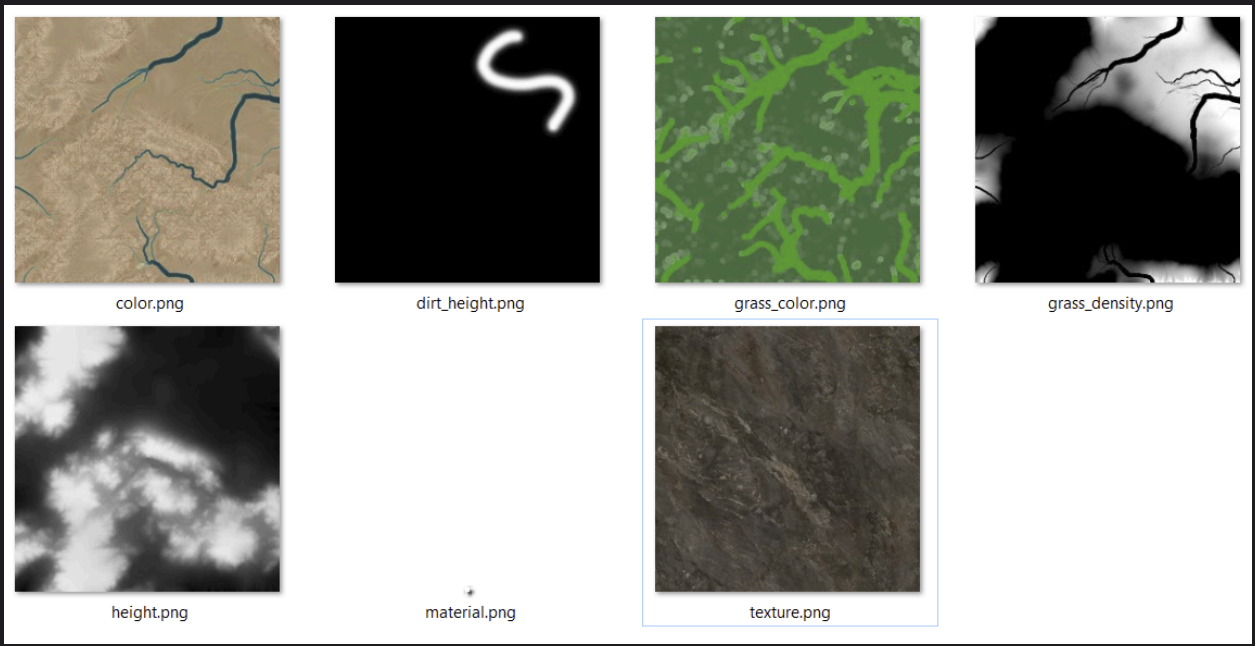
The **Heightmap** tab allows generation of 3D voxel terrain from different PNG files. To create a complete terrain, the following maps/images are required:

- **Heightmap** – Defines terrain elevation
- **Colormap** – Controls terrain colors
- **Materialmap** – Assigns materials using indexed colors
- **Grassmap (optional)** – Adds grass details

This workflow is very similar to **ground.lua**, except it is no longer tied to using RGB channels. This provides greater flexibility and freedom.

Projective Materials

Unlike **ground.lua**, which has material limitations, heightmaps use a **material map** that assigns materials through indexed values. This mode is available only in **8-bit mode**.



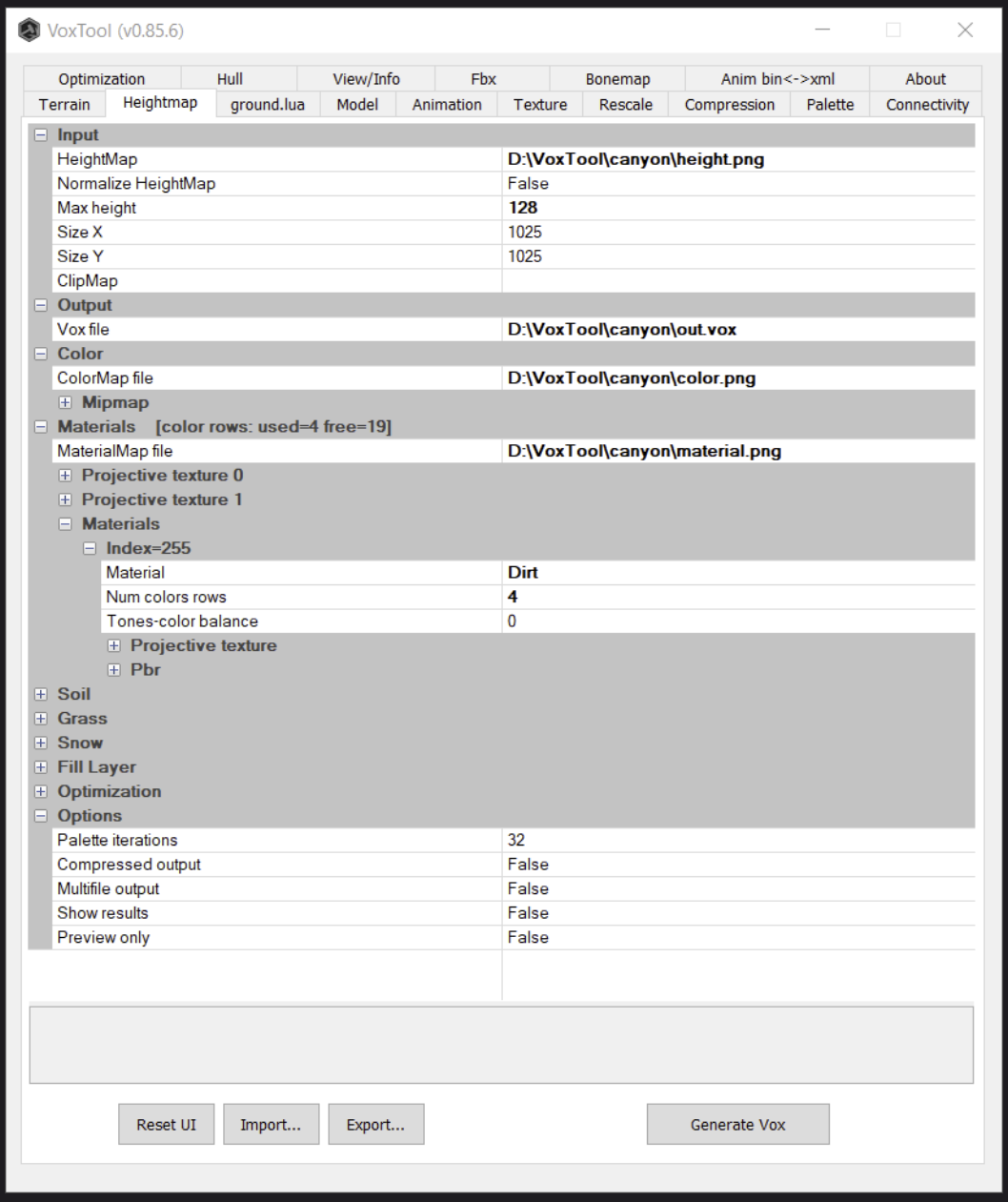
Required Textures

Here are the primary textures used in voxel terrain generation:

- **color.png** – Standard **8-bit** image defining terrain colors.
- **dirt_height.png** – Adds extra height layers to the base terrain.
- **grass_color.png** – Defines the **grass color.**
- **grass_density.png** – Controls **how dense the grass** will be.
- **height.png** – **16-bit grayscale** image ranging from **0 (min height)** to **1 (max height)**
- **material.png** – **an 8-bit color** image where each unique color corresponds to a material slot in VoxTool. In the example above, the image is entirely white because this test case uses only a single material.
- **texture.png** – Used for **triplanar mapping.**

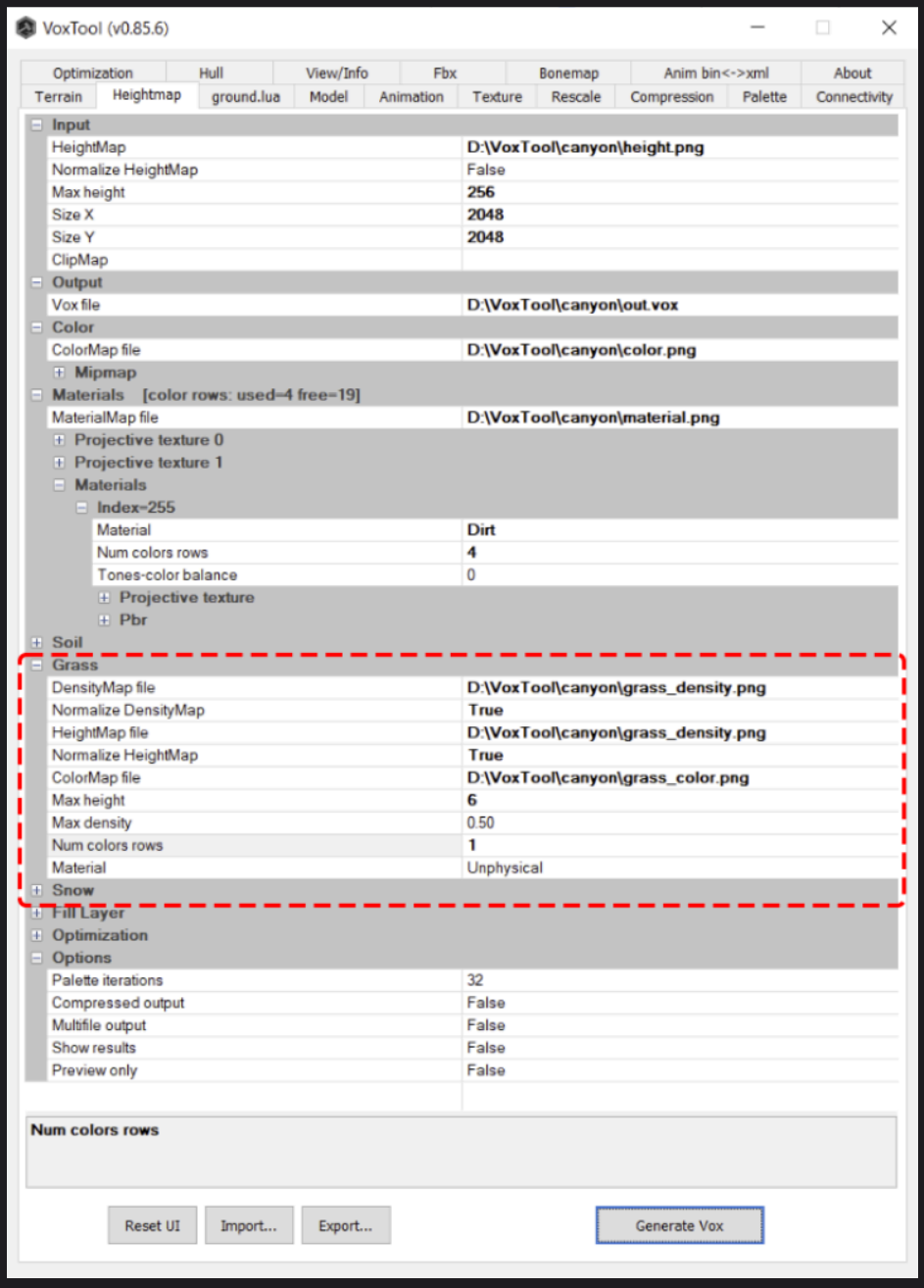
Basic Settings and Materials

In the simplest setup, **one material** is used with the index **255**. The **material.png** file is a plain **white image**, meaning **only index 255** is assigned. More materials can be added later.



The MaterialMap file here is, (as before mentioned) only one index (white). Which in this case will be Index=255. Assign its material (in the game's context) and how many color rows.

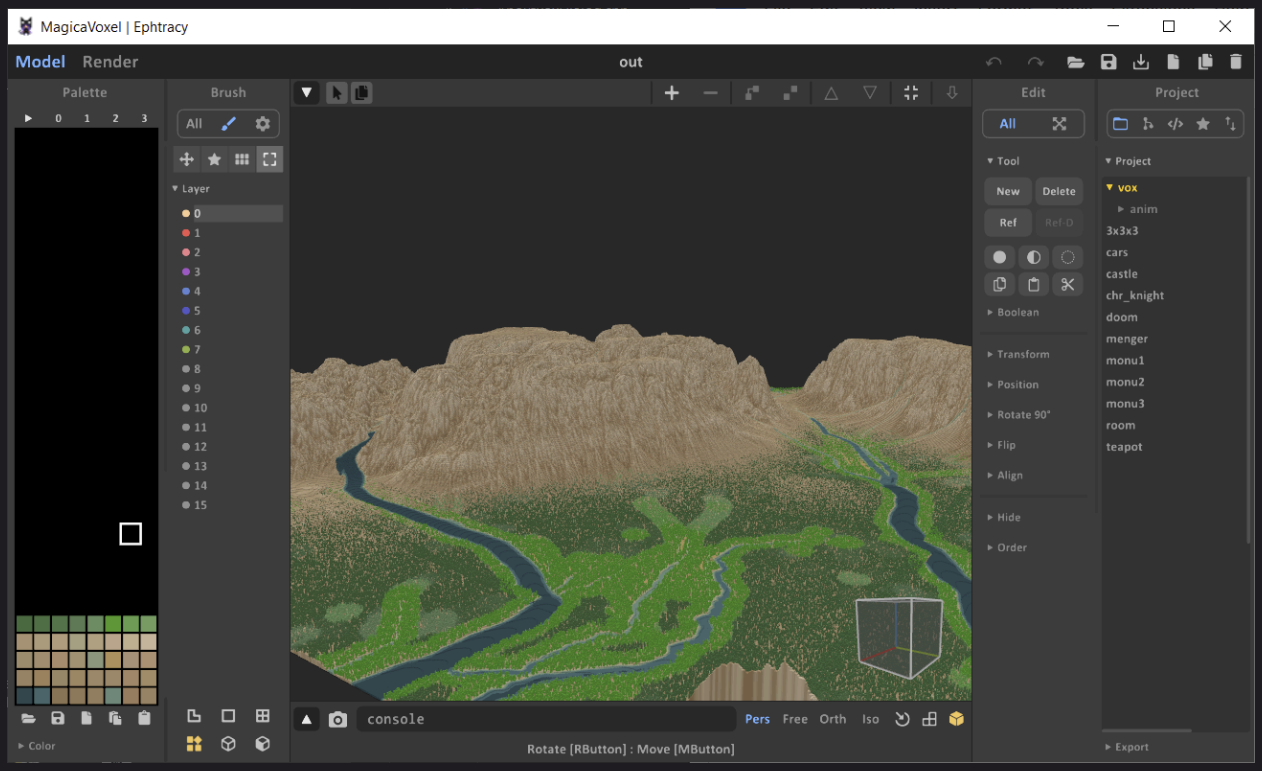
Grass Density and Height Control



- The **density map** controls how thick the grass is:
  - **White areas** → Maximum grass density
  - **Black areas** → No grass

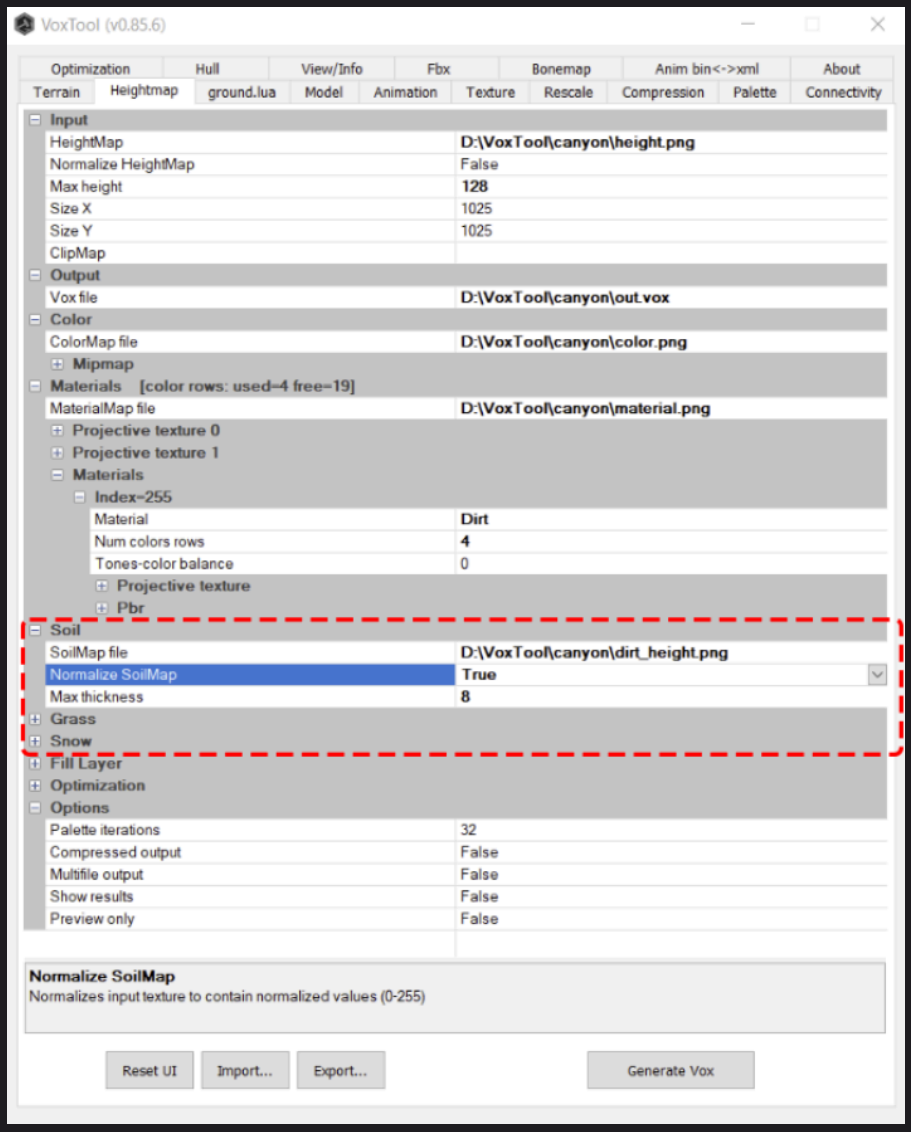
- The **Max Density** setting adjusts the overall density, multiplying it with the values from the density map.
- **Normalize** applies an automatic contrast adjustment, ensuring the darkest values become **black** and the brightest become **white**.
- The **heightmap, max height, and normalize height** settings work similarly but control the **grass height**.
- The **ColorMap** determines **grass color**. If not specified, the terrain **colormap** is used instead.

Both **grass density and height** are controlled by the **grass_density** image.

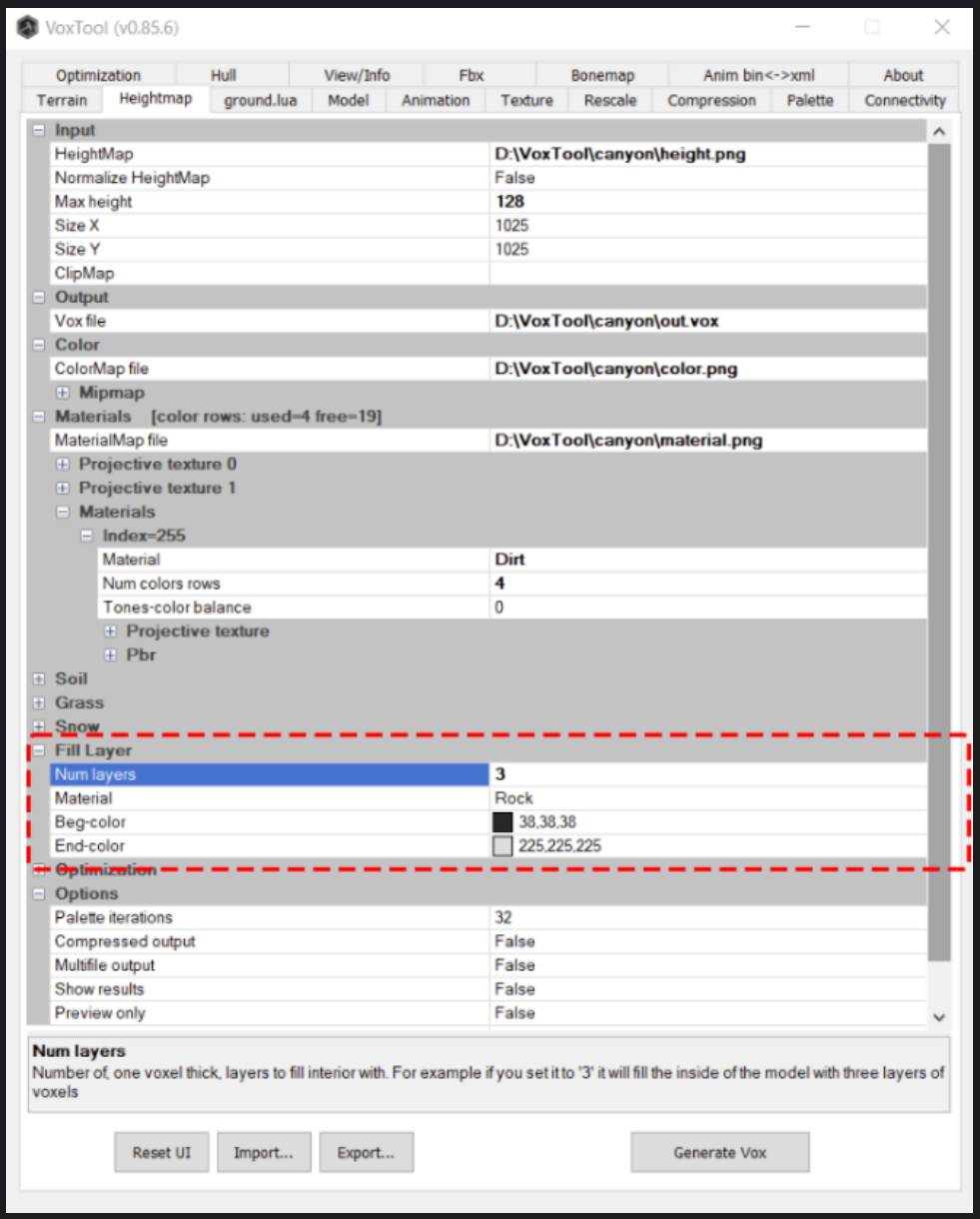


Grass density and height is both controlled by the same grass_density image

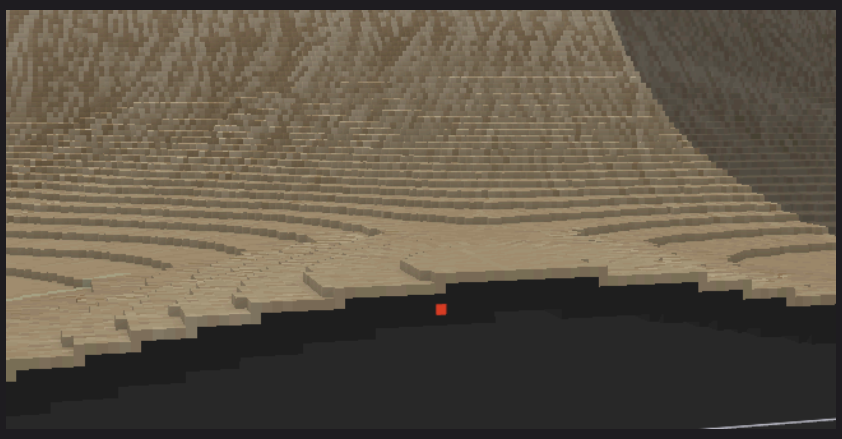
Additional Layers and Fill Layers



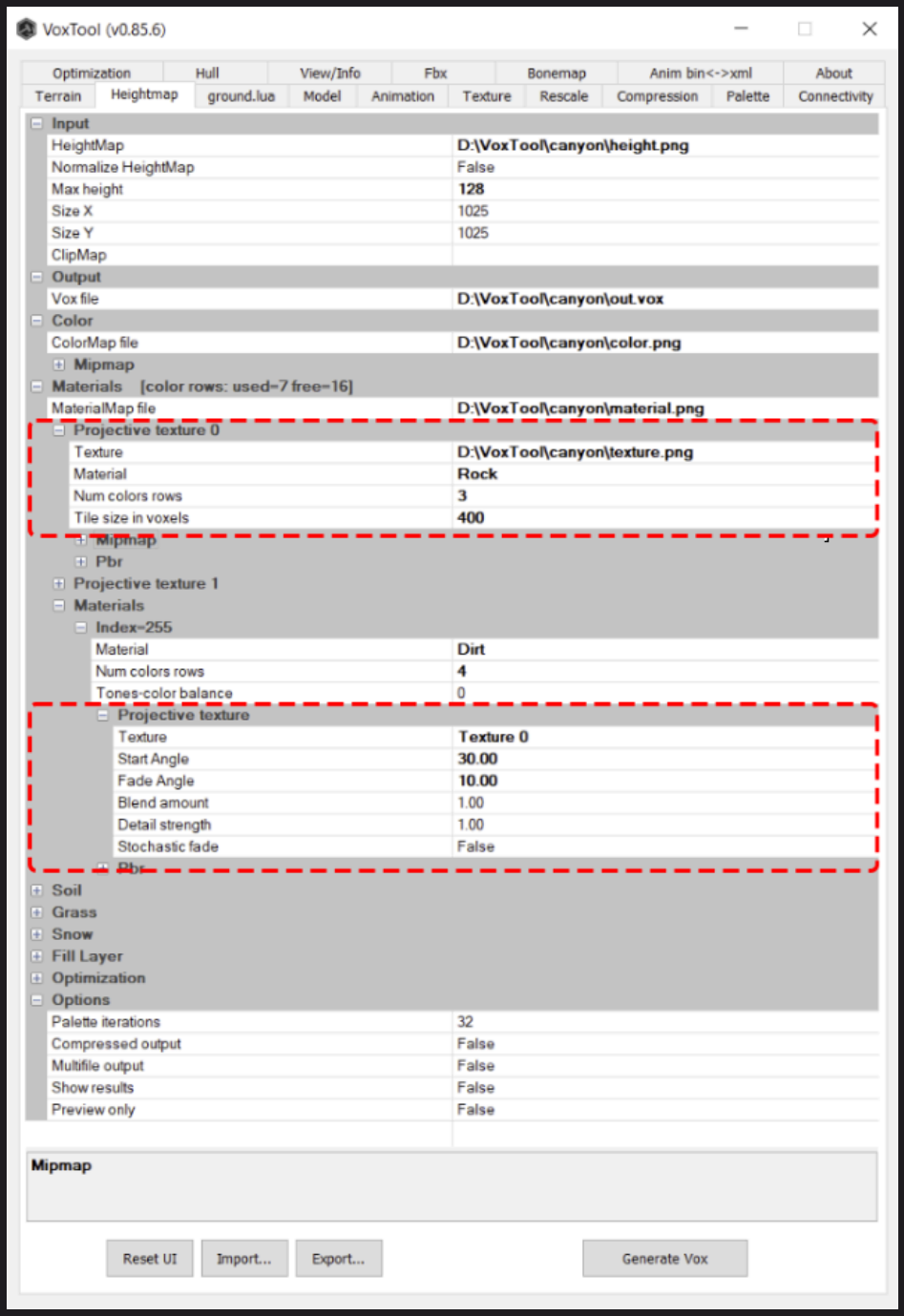
- **Soil:** Adds additional dirt layers on top of the terrain. This can be used for features such as piles of dirt or snow piles. It is destructible and acts like dirt.a



- **Fill Layers**: Adds layers **beneath the surface**, often used for **indestructible rock** to prevent terrain from getting unwanted holes.



Projective Texturing for Steep Slopes



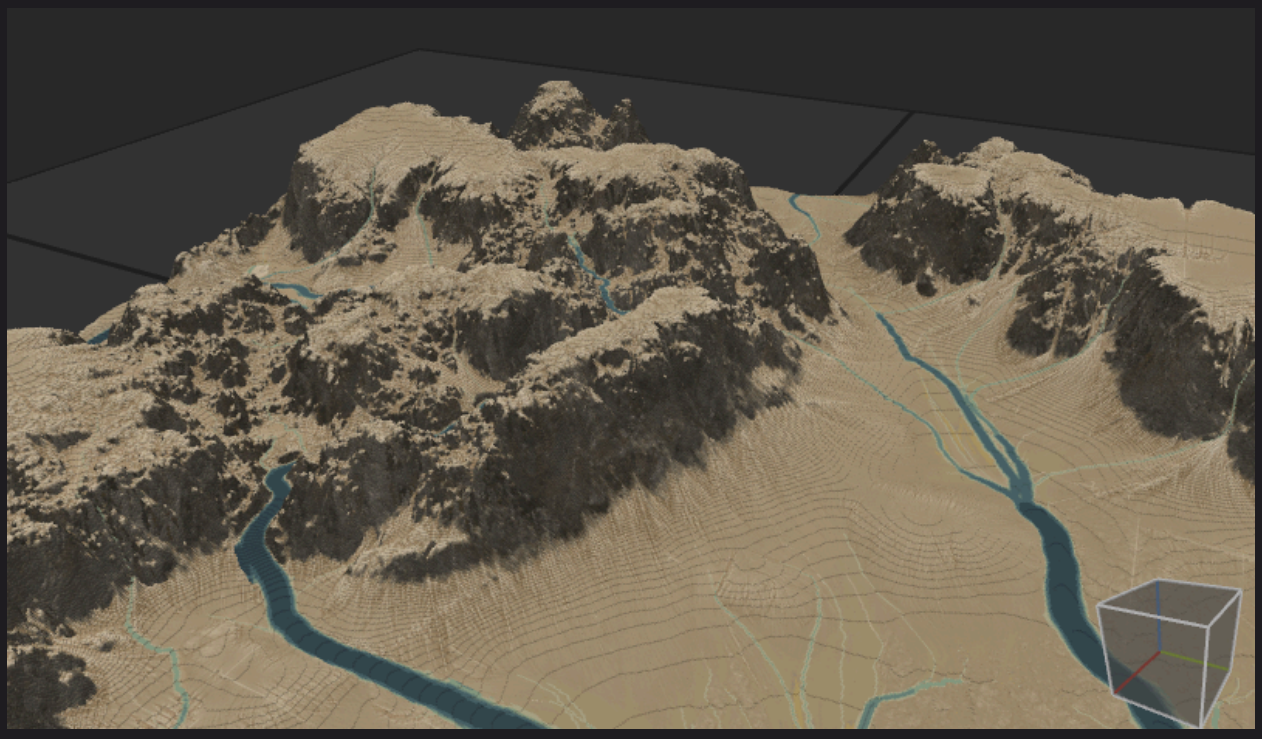
Projective texturing uses **triplanar projection** to apply textures more naturally on **steep slopes**. When using heightmaps, textures are often applied **top-down**, which can cause **stretched textures** on cliffs and steep terrain.

With **projective textures**, detailed textures can be added to these areas by adjusting parameters such as **angle**, **falloff**, and **blend strength**.

How It Works

- **Start Angle (e.g., 30°)** – The angle at which the texture starts applying.
- **Fade Angle (e.g., 40° with Fade = 10)** – Controls how smoothly the texture blends in.
    - At **30°**, blending **begins**.
    - At **40°**, the texture is **fully applied**.
- **Blend Amount** – Adjusts how much the texture blends with the background.
  This is useful when the underlying terrain should remain partially visible.
- **Detail Strength** – Controls how sensitive the texture is to **small bumps** in the terrain.
    - **High detail strength** – Detects even small bumps (e.g., 4 voxels high) and applies the texture.
    - **Lowering detail strength** – Prevents textures from appearing on minor variations.

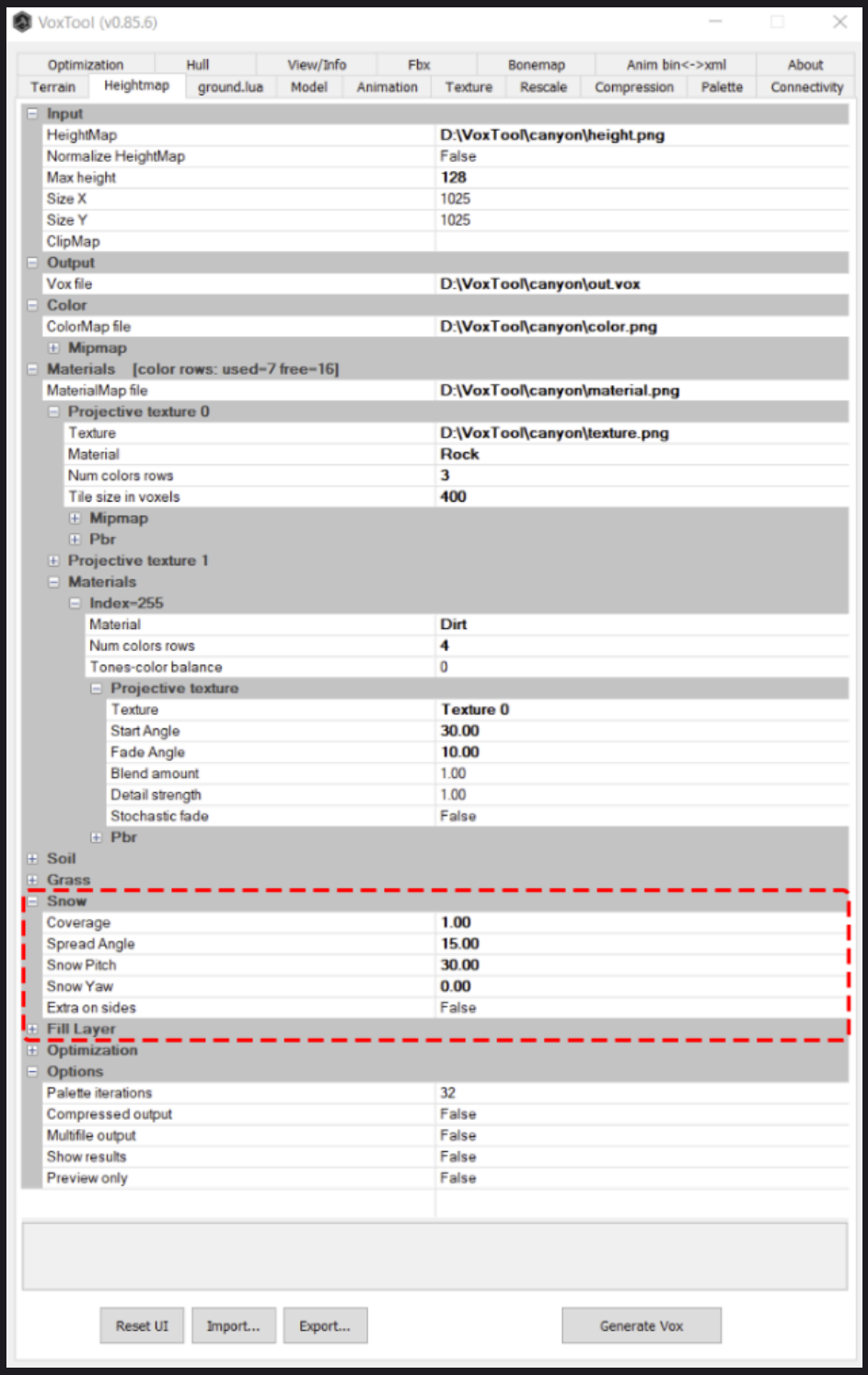
This technique improves **steep cliffs and slopes**, preventing stretched textures and adding finer details where needed.

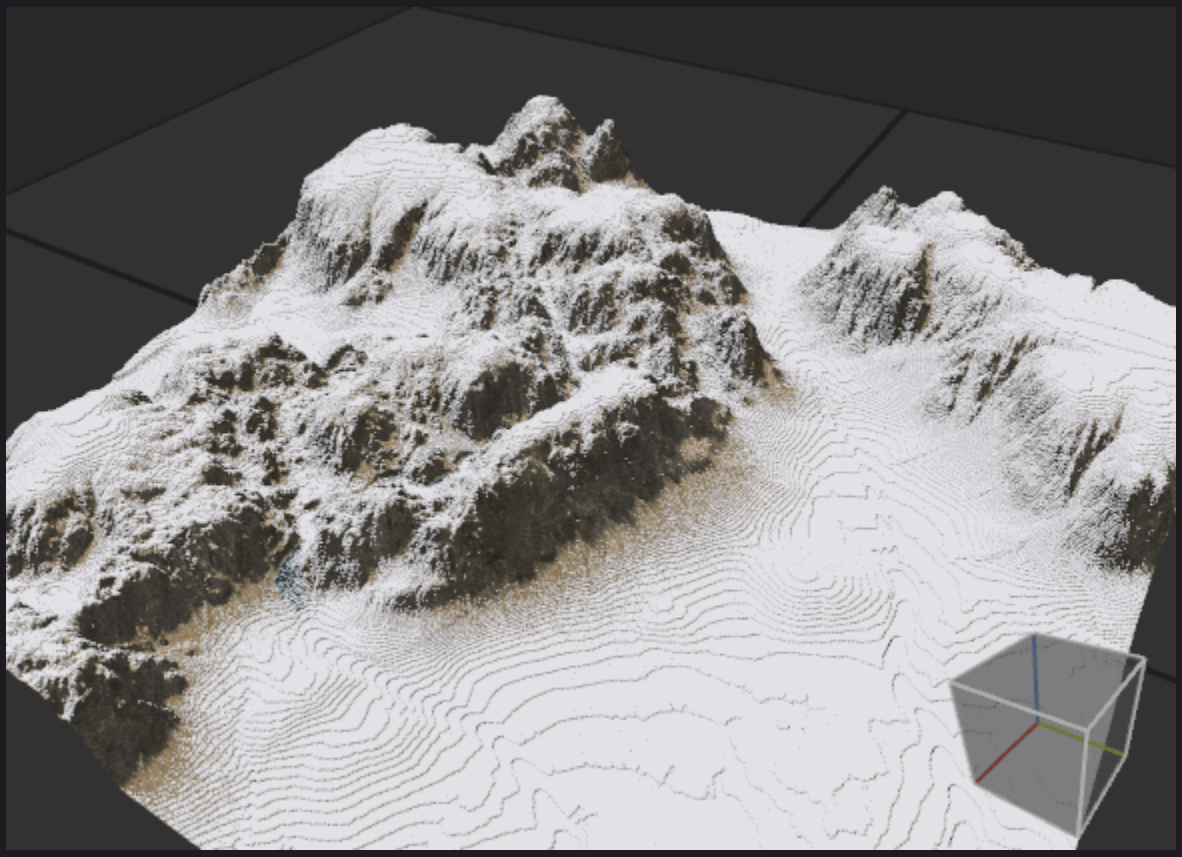


Note how the texture has been applied to the slopes of the terrain.

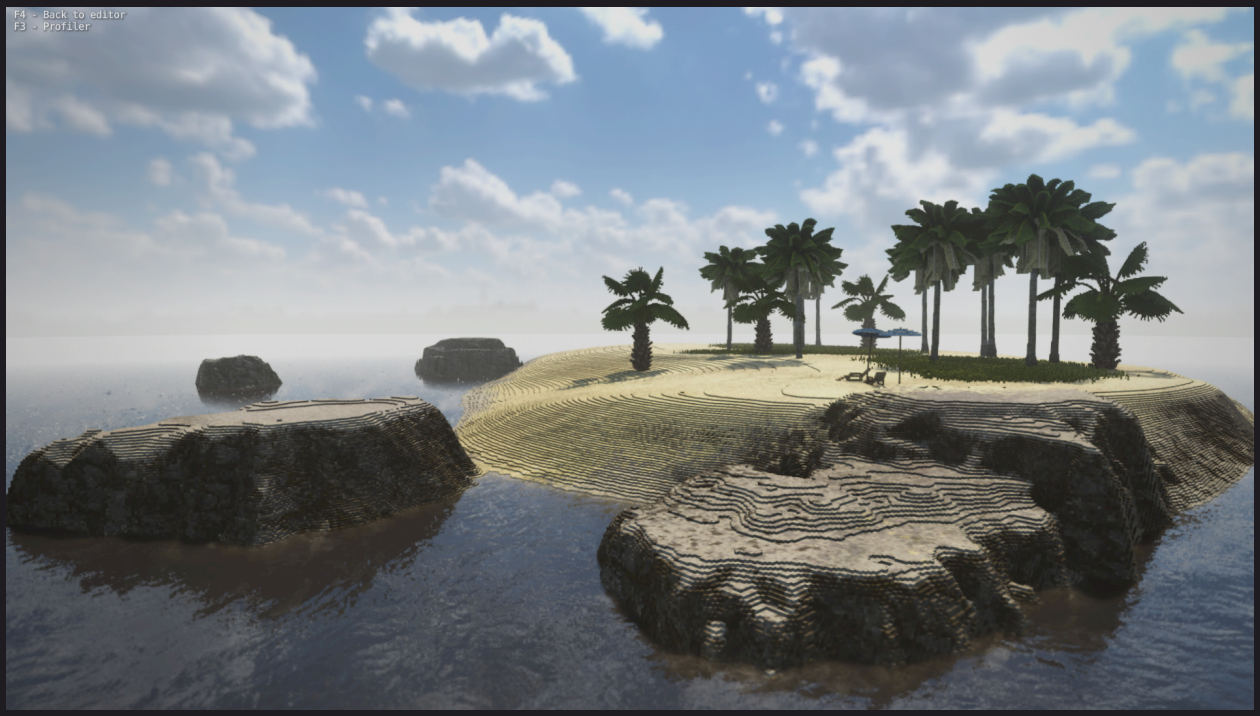
Snow coverage and final example

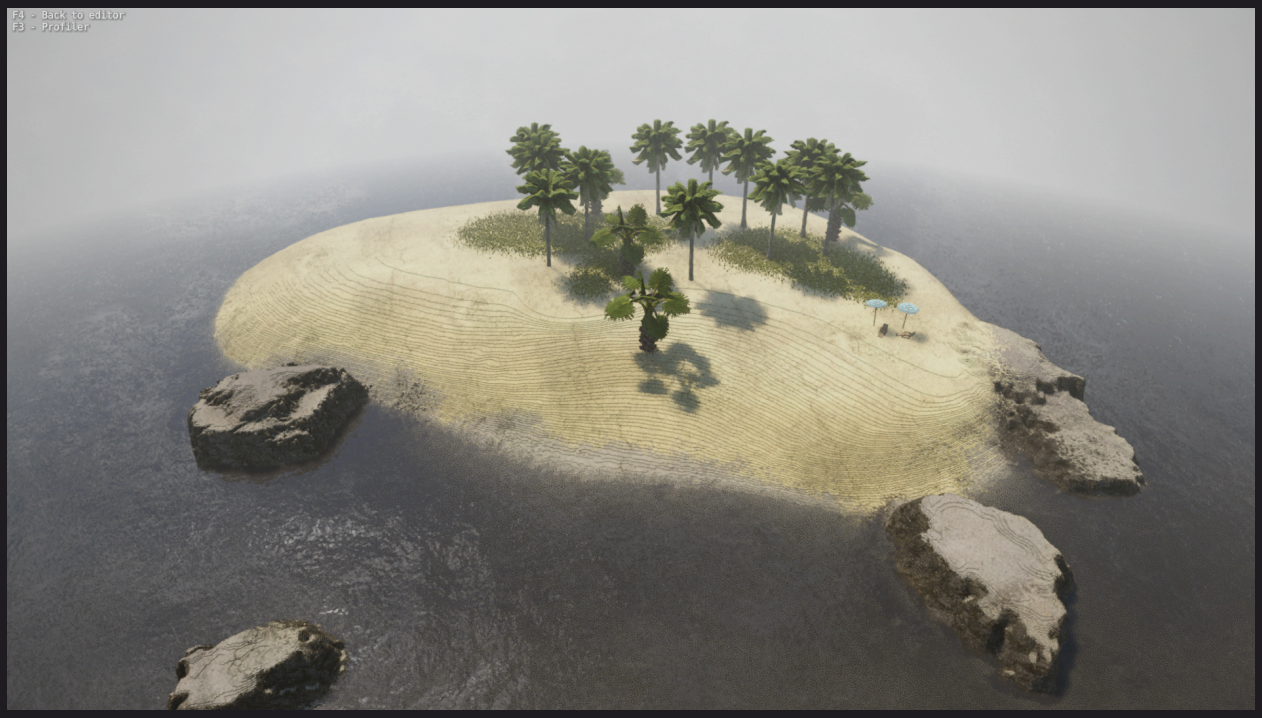
The snow tab allows for generating snow on the terrain. It's quite straightforward, but for more in depth explanations refer to the documentation.

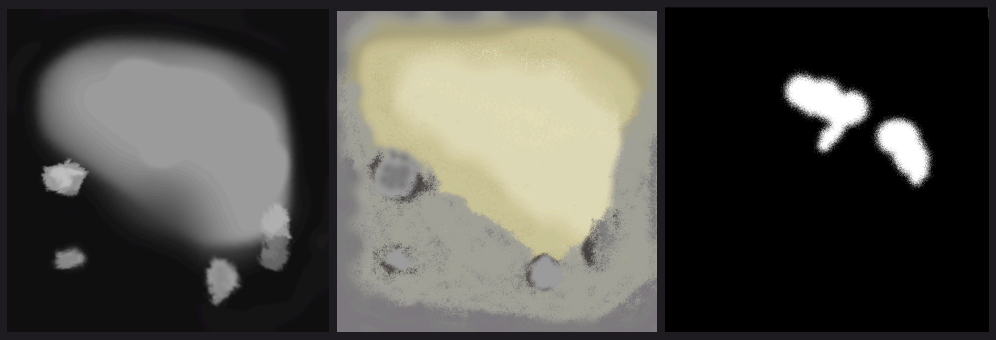


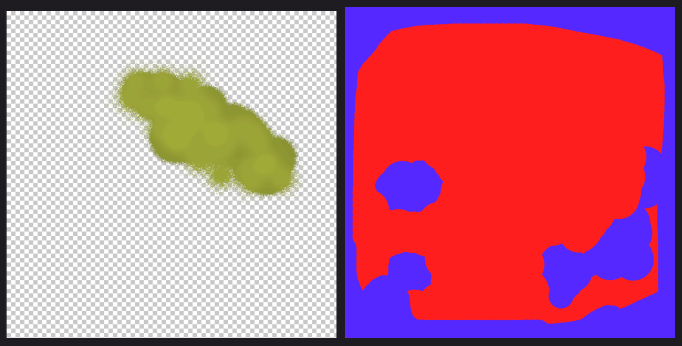


With this new strategy, it becomes possible to create entirely new terrain for the beach scene featuring a wider range of colors and materials, because the colors are no longer directly tied to specific materials. This approach was used to create the rocky texture on the sides of the cliffs shown in the image below.









Here are the terrain maps used in the beach example. The final image is the **Material Map**. In **Photoshop**, **Indexed Color** mode is used, where blue and red appear as separate indexes in **Voxtool**. This allows different materials to be assigned to each color—**red** represents dirt, and

blue represents rock. A **projective texture** is also applied to the blue areas so the rocks have texture on their sides.

Since Indexed mode is only available in 8-bit, it's best to create a separate project for the material map. The other images should remain in 16-bit mode to ensure the highest possible quality.

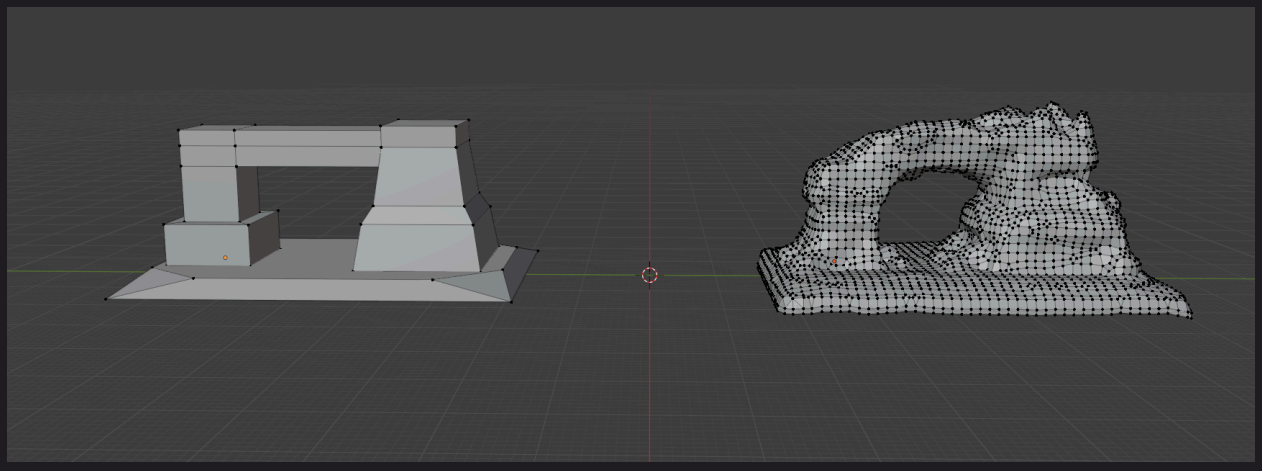
3D terrain pipeline

The last method to create terrain is with 3D modeling. The benefit of using **3D modeling** to create voxel terrain is that it allows for **overhangs, caves, and more complex structures** that standard heightmaps cannot achieve. Prior knowledge of **Blender** or another 3D modeling software is recommended, but the overall workflow is similar to other terrain generation methods.

Like other terrain generation tools, this method also requires several PNG images. In Blender, a mesh must first be created. While 3D modeling experience is beneficial, it is not strictly necessary and depends on the complexity of the terrain being created.

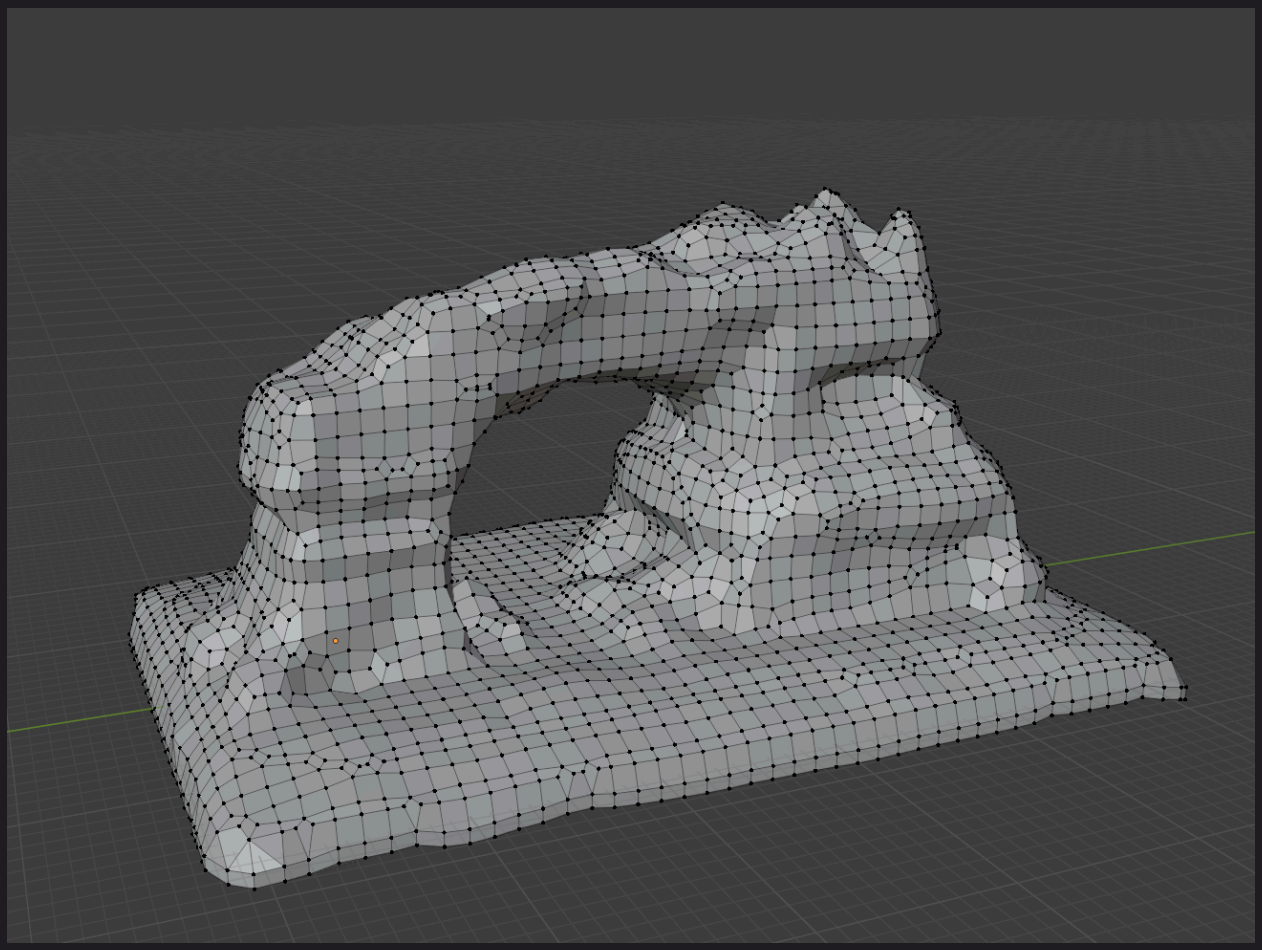
The mesh **does not need to be optimized**, but it **must be watertight—any holes in the mesh will prevent Voxtool from generating the terrain properly**.

To demonstrate how this method can be used, the example will focus on creating an island in the beach scene that includes an overhang, something that is not possible with heightmaps.



This documentation won't cover the **basics of Blender**, but above are two key stages of what the **sculpting process could look like.**

First, **edit the mesh** using the **Edit mode**, refining the base shape. Once the basic form is set, move on to **Sculpt Mode** for the finer details.



The model after sculpting.

For this process, Blender’s **Dyntopo** (Dynamic Topology) feature can be a useful option, as it automatically subdivides the mesh during sculpting. This helps create smoother, more detailed terrain without manually increasing the polygon count in advance.

How clean the topology is doesn’t matter too much, since the mesh will be converted into a voxel model anyway. However, it's still a good idea to keep an eye on excessive distortion just in case something goes wrong, a cleaner mesh will be much easier to troubleshoot and fix.

UV Mapping & Material Setup

Once the mesh shape is ready, it’s time to create a UV map.

In order to paint on the mesh, there needs to be a UV map. A UV map is like a 2D version of a 3D object's surface, so you can paint or apply images (called textures) more easily. It’s like “gift-wrapping” around the object, giving it color and texture.

Create a UV Map

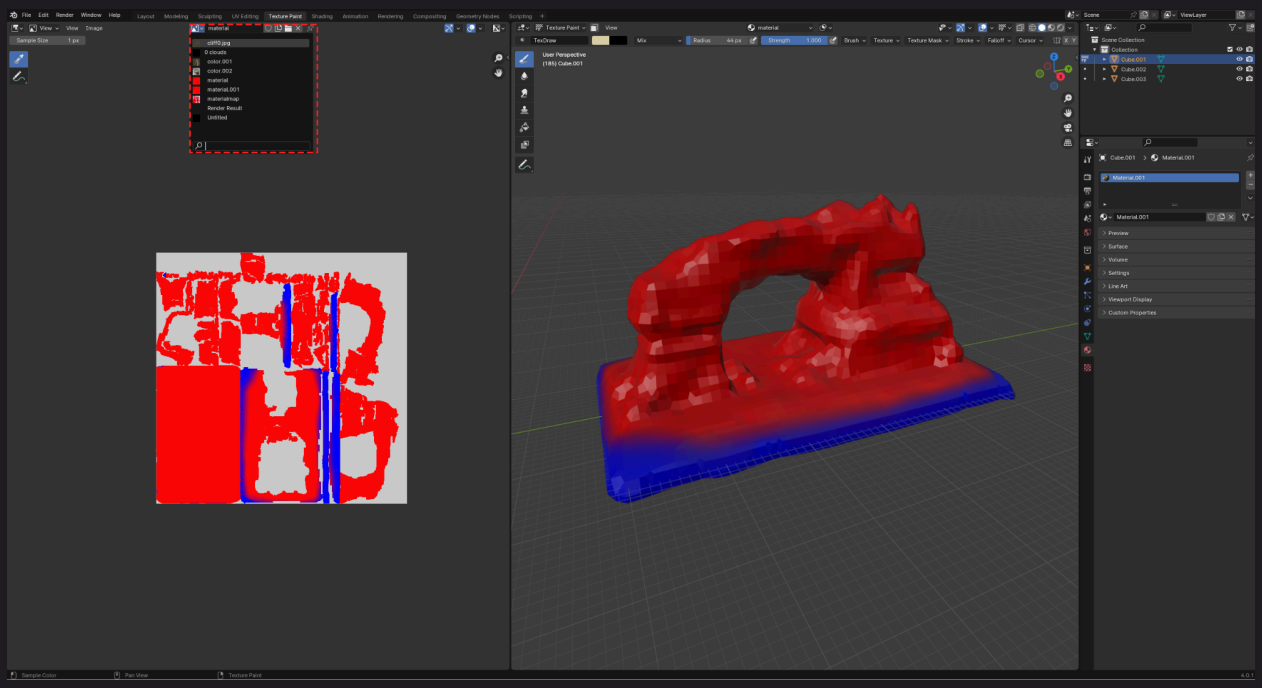
- **In edit mode, press A** to highlight all of the mesh.
- Right click once everything is highlighted, to use **Smart UV Project** in Blender to generate a UV map automatically.

Assign a Material

- A material is **required** to paint on the mesh. In the **properties menu,** locate the material tab and add a new material.
- **In edit mode**, first make sure **all faces are selected**, then assign the material to the mesh.

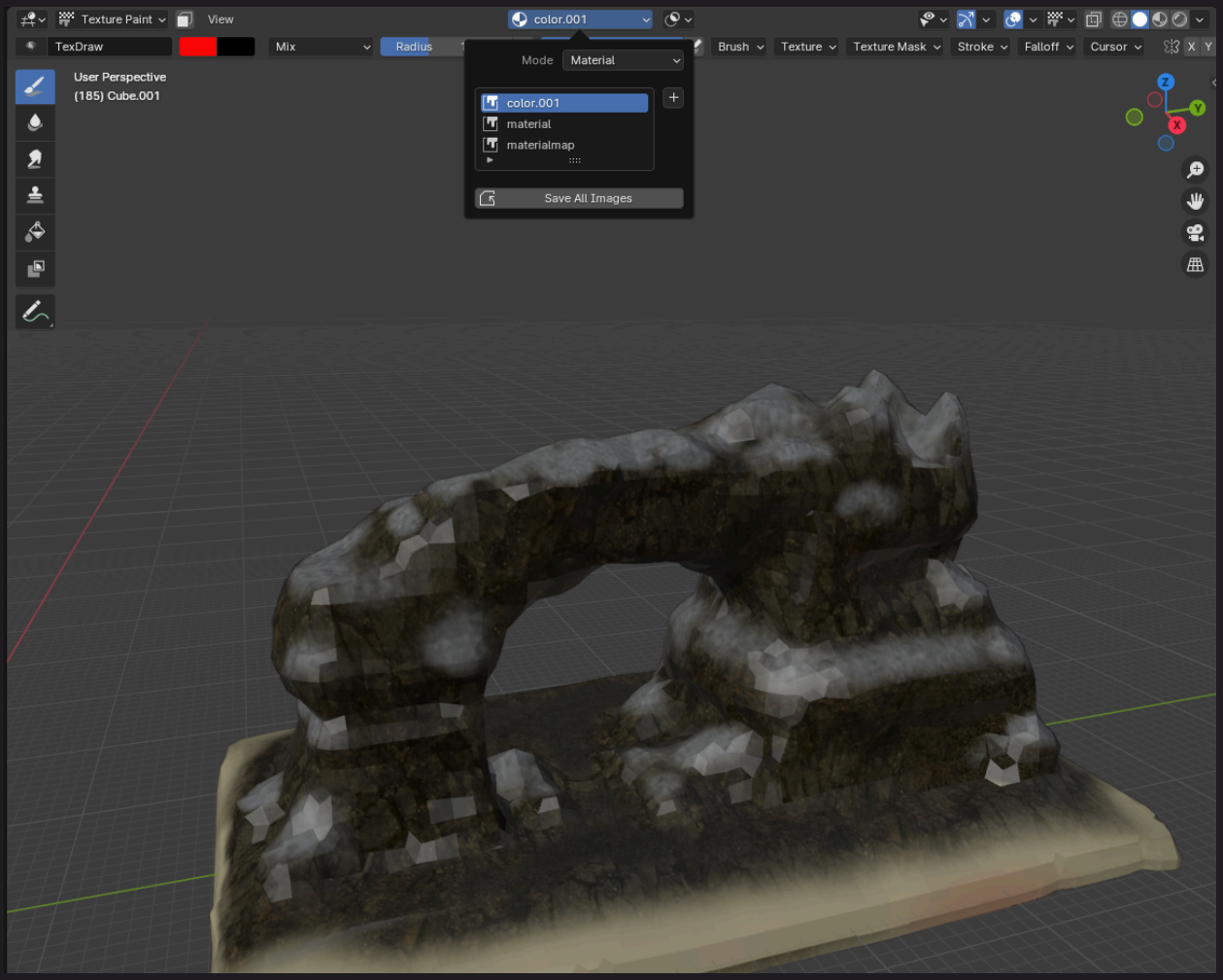
Texture Painting

- Switch over to the Texture Paint workspace tab.
- To achieve the various maps that Voxtool needs to generate the terrain, there needs to be different images(textures) that can be exported from Blender. Each of these images will represent one of the different “maps” (colormap, materialmap etc) we will input into Voxtool, and use to create our voxel terrain. Add a new image to the material in the image editor, which is highlighted in the image below.



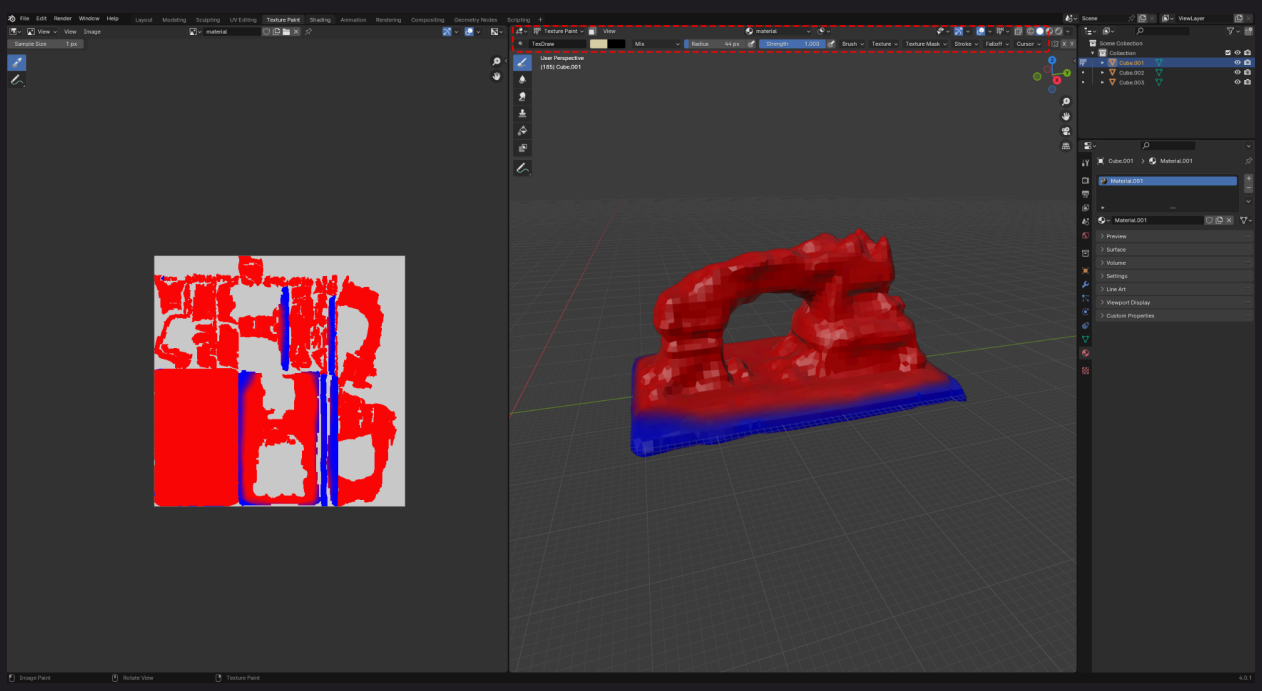
- Each image represents one type of map. In Voxtool there is support for three - material, color, grass density and grass color.

- A good tip is to regularly save your textures manually while working in Blender, as image edits are not autosaved, even when you save the blend file.



Here are the different images at the **top of the 3D viewport**. Switch the material here to change which image is being painted.

The different images that have been created are available at the top of the 3D viewport. Switching to a different image here will change that to be the active painting surface.



In the top menu that's highlighted in red, the color currently assigned to the brush is displayed, allowing direct painting onto the mesh. At the moment, the selected image is the **Material Map**, which appears entirely red because this model will use only one material type (rock).

Create & Export Texture Maps

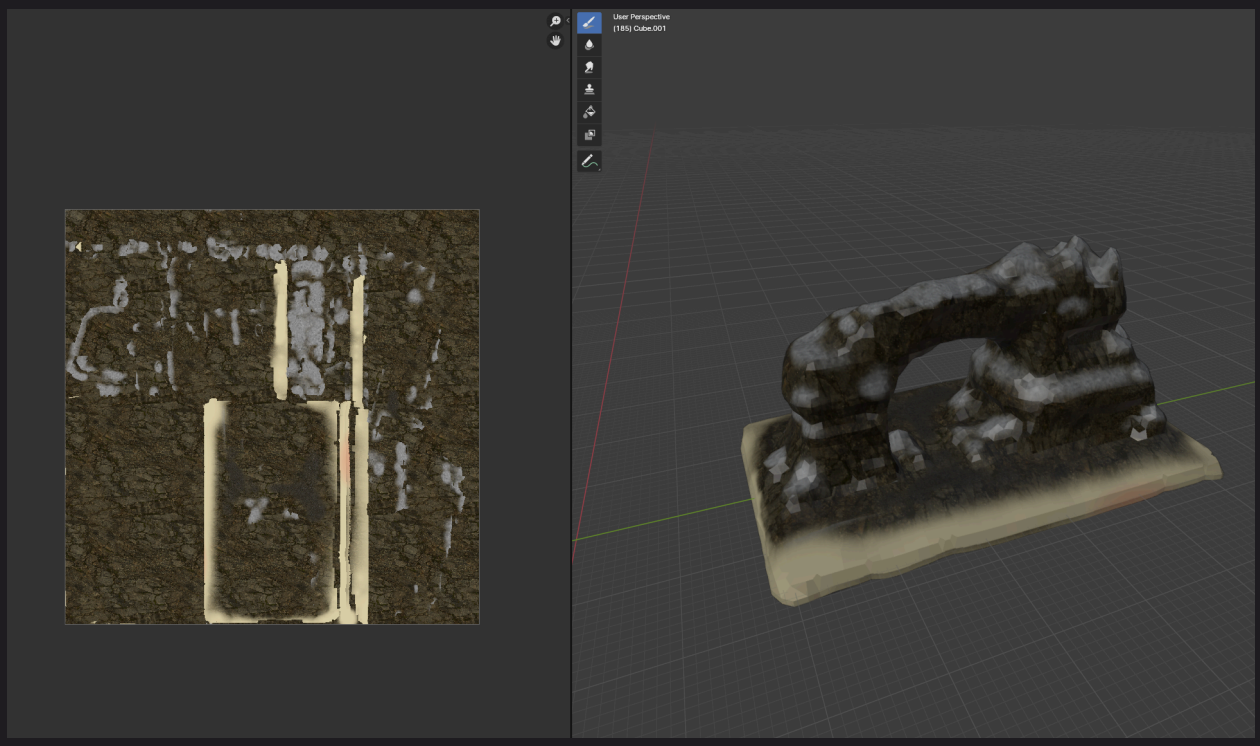
- **Paint on each individual image** as desired.
- Remember that the **Material Map** only reads the RGB values, just like **ground.lua**. Make sure to use pure reds, greens, and blues to ensure that Voxtool correctly assigns the materials.

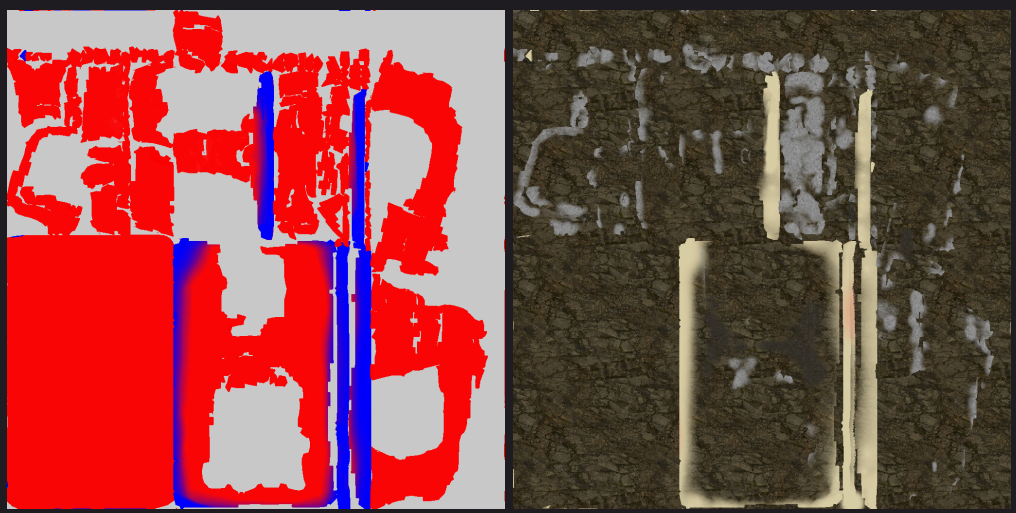
  For example:

  - Pure red = RGB(255, 0, 0)
  - Pure green = RGB(0, 255, 0)
  - Pure blue = RGB(0, 0, 255)

Using shades or blended colors (like RGB(128, 25, 0)) to paint rock material won't work properly, as it needs RGB(255, 0, 0). So stick to the exact values when painting materials.

- Once finished, **export the UV maps individually as PNG images** and import them into **Voxtool** for further processing.



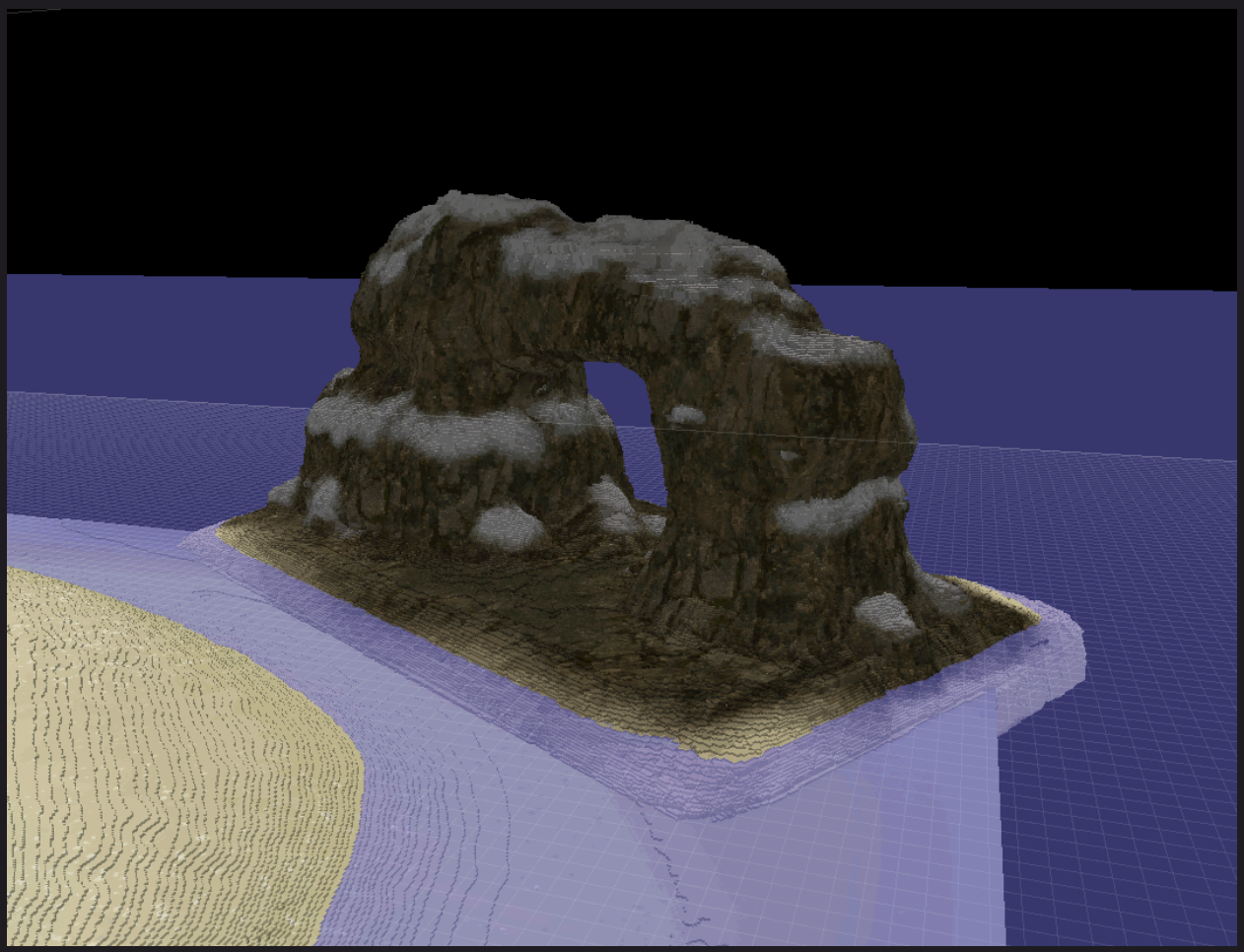


Here are the textures exported from the example model above, ready to be put into Voxtool!

Red-Channel	
Material	Rock
Num color rows	3
Tones-color balance	0
Pbr	
Green-Channel	
Material	None
Num color rows	0
Tones-color balance	0
Pbr	
Blue-Channel	
Material	**Dirt**
Num color rows	**2**
Tones-color balance	0
Pbr	

Note the material settings in **Voxtool**. All areas painted red will be assigned the **rock** material, and all areas painted blue will be assigned the **dirt** material. **Green** is left empty, as there is no green present in the texture.

Result in Editor



Here is the terrain in the editor.

That’s it! Experiment with different shapes, colors and materials to bring your levels to life!