

Block Modification Guide (Basic)

Overview

This tutorial provides a quick start guide to modify an existing block, in *Blender 3D program*, and the initial set-up inside VRAGE3 Editor.

This is a basic guide. For more detailed information, check the **Advanced Modding Guide (link it)**

Prerequisites

Ensure you have the necessary files to continue. Prerequisite files from the SE2 modding SDK needed for this tutorial can be found here:

- steamapps\common\Space Engineers 2 - Mod SDK\ExampleMod1\Modded Ion Thruster

- steamapps\common\Space Engineers 2 - Mod SDK\Blender Library Template\SE2Decal&Material.zip

Material Library Set-Up in Blender

To use it as a library, the template .blend file needs to be added as “asset library”, doing these steps:

1. Unzip the **SE2Decal&Material.zip** to a preferred location
2. In your Blender Startup file go to menu **Edit -> Preferences -> File Paths**
3. In the “**Asset Library**” section, press the folder icon to set the path to where you stored the **SE2Decal&Material**. Press “**Save Preferences**”.
4. Extend a new window in the UI, in any place you want, and go to the top left icon and choose **AssetBrowser**
5. On the top left drop down, select **SE2Decal&Material**. You now have an organized **library with Decals, Trims and Materials** to be placed on any model.
6. Go to menu **File -> Defaults -> Save Startup file**

If you try to Drag&Drop one of the materials, but the material looks **pink (no texture)** in Blender’s Viewport Material preview, follow these steps to fix it:

1. Go to **File -> External Data -> Find Missing Files**
2. Set the path to the projects Material DDs Textures (Vanilla/Content/Assets/**BlockMaterials**) and press **Find Missing Files**
3. Wait a few seconds and the **materials will be updated.**
4. Go to **Preferences** and press **"Save Preferences"**.
5. Go to menu **File -> Defaults -> Save Startup file**

Basics

Mesh Types

There are three types of meshes, in any given block in SE2:

1. Non-Fractured mesh - intact object that doesn't have construction stages and cannot fracture
2. Fractured mesh - same as the non-fractured, but this time it will fracture - you can break the model by parts.
3. Deformed mesh - Exactly copy of Fractured mesh but the meshes are deformed

LODs

LOD - Level Of Detail: refers to the complexity of a 3D model representation, which can be decreased as the model moves away from the viewer

As for creating them it's the normal process of decimating a mesh using your preferred techniques, keeping in mind of what distance the mesh is going to be seen.

If we make changes in the LOD0, we also need to make the same changes in the rest of the LODS - including baking the *Last LOD* texture if the changes are visible at LastLod distance - as to not have visual mismatch when the Engine switches from one LOD to another.

The amount of triangles per LOD is based by the blocks size and calculated in the LOD Calculator:

Block Size (m)	Has fractures	Max LOD0 triangles	LOD0 triangles	Number of LODs	Max triangles per LOD	Max materials	Distance (m)		
X	0.5	<input type="checkbox"/>	8444	9276	4	0	8444	8	0.00
Y	0.5					1	1422	8	5.00
Z	0.25					2	236	6	10.00
Surf.	1					3	32	1	20.00
						4	-	-	-

Collisions

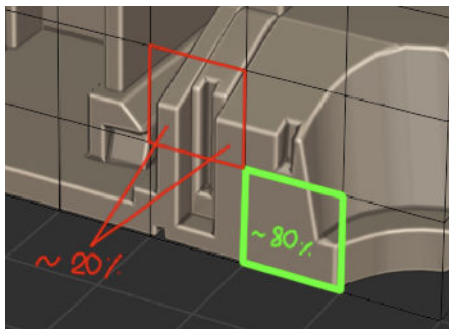
Besides providing the meshes with all of their versions and respective LODs, we also have to provide collision geometry, for physics to behave correctly and be able to collide the model with the environment and characters. For the physical simulation, we generally use a simplified geometry representation of the actual model.

In our example, for collisions we are not going to change anything. We will provide the collision files (.bin and .gtlf) and will use them on our modified model.

MountPoints

MP - Mount Points are specific locations on the surface where blocks can be arranged/attached. These are essential to connect anything to our block.

In order for the Editor to generate MPs properly, the mesh needs to cover **>50% of the 25cm grid cell**.



With this information, we can at least predict if a surface of our block will be detected as mount points or not. So make sure that you are aware of this to avoid any unexpected mountpoints areas where you don't want them, or lack of mountpoints where you need them.

Also check if the Area is actually **touching the Bounding Box**. Even if a surface is slightly pushed down, the Generator will create MPS there. Placing anything there will look like it's floating.

Materials

SE2 doesn't use individual (baked) textures for most of the block's models, except its last LOD. The majority of models use materials from our Material Library and are assigned automatically in the Editor, if the material of the mesh from our modeling SW has the same name.

In Editor, these materials and decals are located in **Assets/ BlockMaterials**. They can also be previewed inside Blender, after the **SE2Decal&Material** set-up.

The rules for Material limits are given by the LODCalculator, depending on the Block's size and respective LOD:

Max triangles per LOD	Max materials	Distance (m)	
0	8444	8	0.00
1	1422	8	5.00
2	236	6	10.00
3	32	1	20.00
4	-	-	-

Block Modification

Fetching the original model

To start, we are going to fetch the Small Ion Thruster block from the project folder - Vanilla/Assets/blocks.

We are specifically looking for the FBX files only for the NonFractured and Fractured LODs, so that we can import them to the 3D software, make small modifications, and import again.

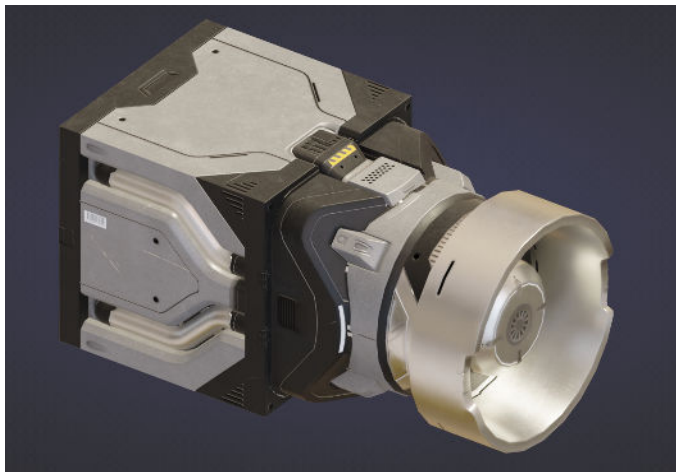
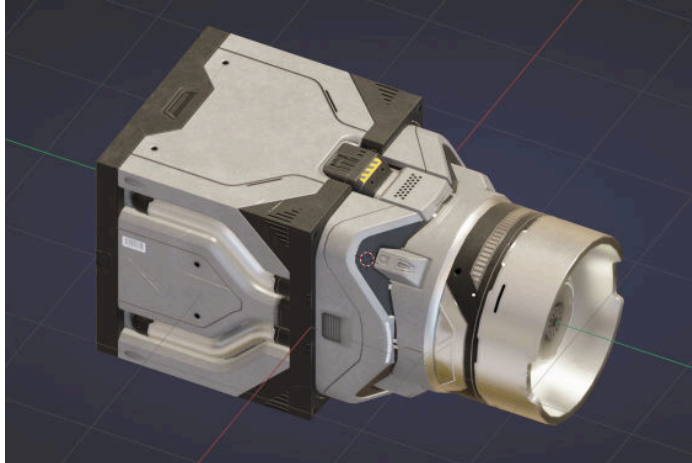
If you want you can skip the **Block Modification** process and used the already modified files, provided files mentioned in the [Prerequisites](#)

Changing the Mesh & Materials

In our example, we are just going to change one of the materials and resize part of the nosel.

1. To be organized inside Blender, create different Scene Collections to import your Nonfractured, Fractured meshes.
2. Import the LODs for each collection created.
3. In Non-Fractured LOD0 Mesh, modify one of the materials to a different material by dragging the material from the Material Library. You can also select a mesh part and, in edit mode, assign a different material from the Material Properties of the mesh.
4. Modify the mesh to your liking, respecting the Tris Limit given by the LOD Calculator and not getting outside of the Bounding box of the object.
5. Do the same modifications for the rest of the Non-Fractured Mesh, except the Last LOD.

Original / Modified Ion Thruster



Fractured Model

To fracture our models, we use **planes with deformation**, and make **boolean cuts** to the model. We do this process to all LODs and the cuts need to match and respect the tris limit for each fractured LOD - given by the LODCalculator:

Fractures
5

Since the fractures are already done in the Ion Thruster Fractured files, we only need to **respect and do the same changes** that we did on all NonFractured LODs:



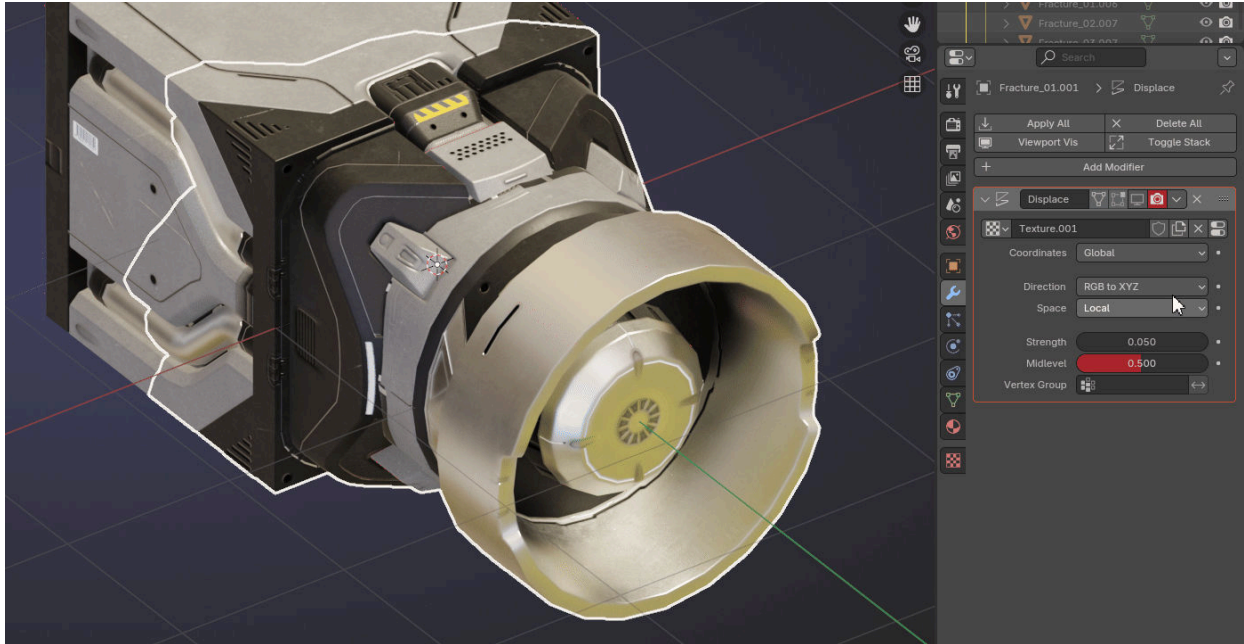
Deformed Model

Since we also changed the form of the nozzle, this would make it harder to modify in the already deformed IonThruster mesh.

So we will guide you through the process of deforming the mesh:

1. To keep things organized, we create a duplicate of Fractured LOD0 Collection and rename it to Deformed.
2. Now to deformed the already fractured pieces, we need to apply a **Displace Modifier with the following specifications:**
 - We apply the Noise Displacement based on **global** coordinates, **RGB to XYZ** direction and **Local** Space. Try to keep the Strength between **0.05 - 0.1**
 - In the Mesh **Texture Properties** add a **Clouds** Texture: Type **Soft, Grayscale** Color, **size 0.25** and **Depth 4**.

With these settings we have a subtle and controlled deformation:



3. When you have the proper deformation that you want, you can select all meshes (selecting the and **copy the modifier to the selected.**
4. For the LODs, duplicate the rest of the fractured LODs and follow the same process - add the modifier and copy to the rest of the fractures.

Creating the Icon

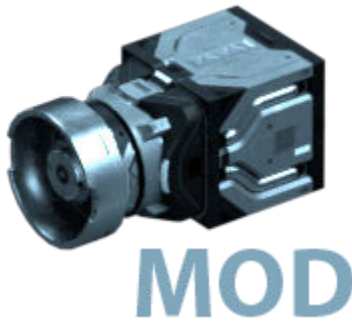
Icons are an essential part of the blocks implementation, which allows us to visually identify our block from a list or UI shelf, inside the game.

We're using Marmoset toolbag 4 to render the icons but you can render it in blender as well (or any other program of your choice)

Mandatory specifications:

- Size: 200x200px
- Format: PNG
- Transparent PNG
- Non-colorable
- Try to render at a similar camera angle of the existing icons

Final icon Texture:



Just to be easily distinguishable from the original one, we added the Text *MOD*

LastLOD Baked Texture

We're now using **Marmoset Toolbag 4** (paid) to bake the lastLOD textures but you can use **Blender** to bake the same maps.

Useful tutorial: [▶ How to Bake textures in Blender under 3 minutes Beginner Tutorial](#)

Rules for the Bake:

- Your **high-poly** mesh needs to be the **penultimate LOD** (in our case the LOD3) and the **low-poly** is the lastLOD mesh (in our case the LOD4). This way we will have a correct transition between them.
- Give the right name to the textures. Needed maps and names:

<yourModelName>_LastLod_**albedo**

<yourModelName>_LastLod_**ao**

<yourModelName>_LastLod_**metal**

<yourModelName>_LastLod_**normal**

<yourModelName>_LastLod_**roughness**

<yourModelName>_LastLod_**colormask**

EXTRA; <yourModelName>_LastLod_**alphamask** - if your model has glass

parts or planes that need alpha masking.

- The resolution of the baked maps should be **128x128 pixels** - no need for something larger because this mesh will be seen pretty far, and a low resolution texture has low impact on performance.
- To bake a **Metal** texture in Blender, go to Painted Material **Colorable** and Painted Material **Darker** materials , in Shader Editor, and set their Emissive to **Full White** with **1.0 of Intensity**. Then bake the map using the **Emit** option.
- The **colormask** is a mask that has only 0 and 1 color values. We use this to define which part has a “colorable” material in the engine and which does not. To generate the exact same thing in blender, to the same steps for the baked Metal but keep the PaintedMetal **Darker** Emissive **Intensity to 0**.
- The rest of the maps are created with the **normal process** of blender baking.
- Save the outputs as **.png** images.

In our example we got:

- ModdedIonThruster_LastLod_**albedo**
- ModdedIonThruster_LastLod_**ao**
- ModdedIonThruster_LastLod_**metal**
- ModdedIonThruster_LastLod_**normal**
- ModdedIonThruster_LastLod_**roughness**
- ModdedIonThruster_LastLod_**colormask**

Editor

By this stage you should have all the parts you need:

- All LODs of Non-Fractured Meshes
- All LODs of Fractured Mesh
- All LODs of Deformed Mesh

- Collision of Non-Fractured Mesh (contains only 1 collision)
- Collisions of Fractured Mesh - (contains all collisions for each fracture)

- Collisions file of Deformed Mesh (exact copy of the Fractured one)
- Icon image
- LastLod Textures

Folder Structure

- Materials - Created in Assets / BlockMaterials
- Blocks - Created in Assets/ Blocks

Creating the LastLOD Material

1. Check in Blender if your LASTLOD Materials names match the name of the LastLOD baked texture (Material name = Texture Name)
(Nonfractured LOD3 mesh - LastLOD - will be the same for Fractured and Deformed versions)
2. Create a folder for all your Model files in **Assets/ Blocks**
3. Create a folder inside, called **LODMaterial**
4. Drag & Drop you *Last Lod* Textures
5. Select all imported textures, then **Right-Click -> Create Asset -> Material -> PBR**

Material

6. Double click on the generated material file to check if it is working, in the render view.

You can also change the preview type, from small astronaut to Box, in RenderView by going to the **Cube Icon** and changing in the drop down list.

Import NonFractured Model

1. Create "**Models**" Folder in the already created Assets/ Blocks/ ModdedIonThruster folder and the sub-folders (**NonFractured, Fractured and Deformed**)

2. From blender, we are going to export each version and respective LODs. Importing *fbx* between different softwares **may create troubles** for scaling.

To uniform things between Blender and Vrage let's use this settings:

- Go to Blender's **Scene Properties** and set Unit system to **Metric** and Scale to **1.0**
- In the **export window**, select **"Fbx all"** (by default is "all local")

3. Export each LOD. LOD0 version of the file has no suffix to it, so is just the name of the model:

ModdedIonThruster.fbx (LOD0)
ModdedIonThruster_LOD1.fbx
ModdedIonThruster_LOD2.fbx
ModdedIonThruster_LOD3.fbx

4. Drag & drop the FBX files + the provided nonfractured collisions into the Models/ Nonfractured folder
5. Select all files (except collision.bin) , **Right-Click -> Create Asset -> Model**
6. Click on the .model file and zoom in/out to see if your model is showing all LODs correctly

Import Fractured and Deformed

Repeat the process:

1. Export the FBxs and drag&drop the Files (Fractured and Deformed + Respective Collisions) to their respective folder
2. Select all files (except collision.bin) , **Right-Click -> Create Asset -> Model**
3. Double-click on the generated .model files and check if there is any issue.
4. If the LastLod Material was not automatically fetched, you can always go to the inspector view of the .model and search for the material manually, by clicking on the "3 dot icon".

Import Icon

Just Drag & drop the icon image to the main block's folder

Skybox Set-Up

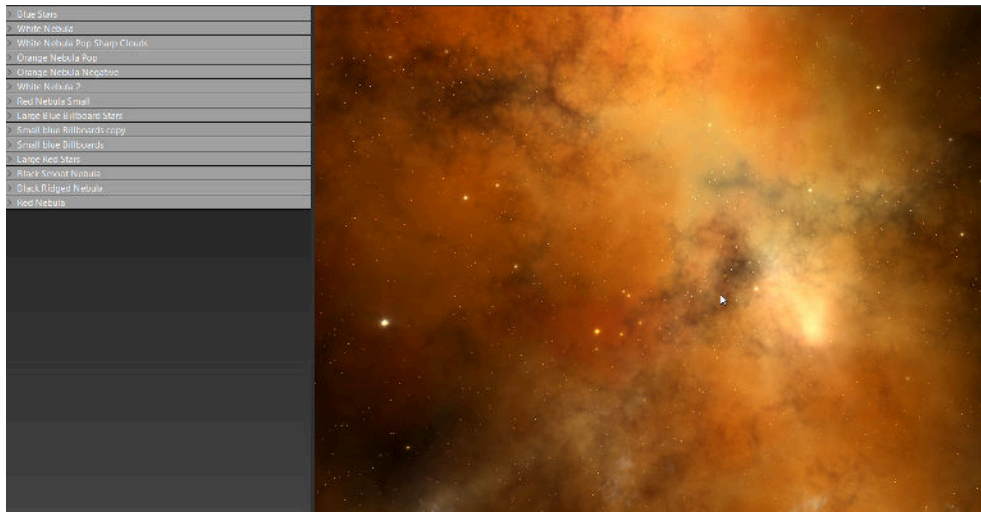
In the following example, we will guide you on how to set up a new skybox in Editor, and how to make it active in-game.

1. Create/Download any skybox for this purpose.

For this example, we made a new skybox from scratch using the awesome SpaceScape App (free).

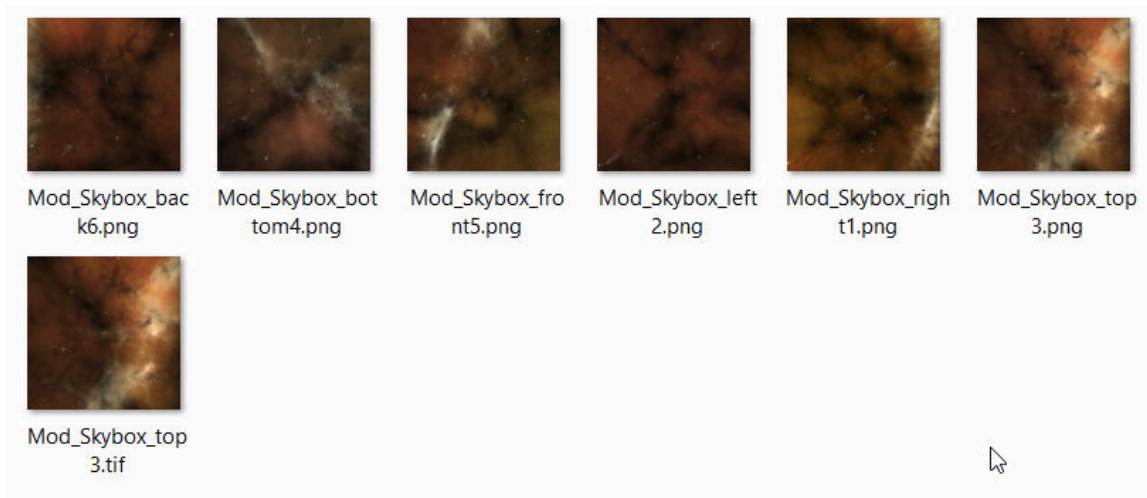
You can create any space theme skybox, using layers and noise generators, and it will export the files with the right orientations.

<http://alexcpeterson.com/spacescape/>









2. Now we export all of the images needed from SpaceScape. Since the app doesn't export in .tiff format - needed for skybox - we will open them in photoshop

individually, do a simple tweak to contrast/light to our preferred taste, and export in the right format. (you can use Krita or other software)



3. Export .tiff with color ICC Profile activated and give the proper name to the files. Since editor generates the definitions automatically by name given to the files, it's best to follow this structure name:

-  back_pz.tif
-  down_ny.tif
-  front_nz.tif
-  left_px.tif
-  right_nx.tif
-  up_py.tif

4. In editor, create a new folder (Mod) in Environment / Skybox and import your Textures

Ice Asteroid Voxel

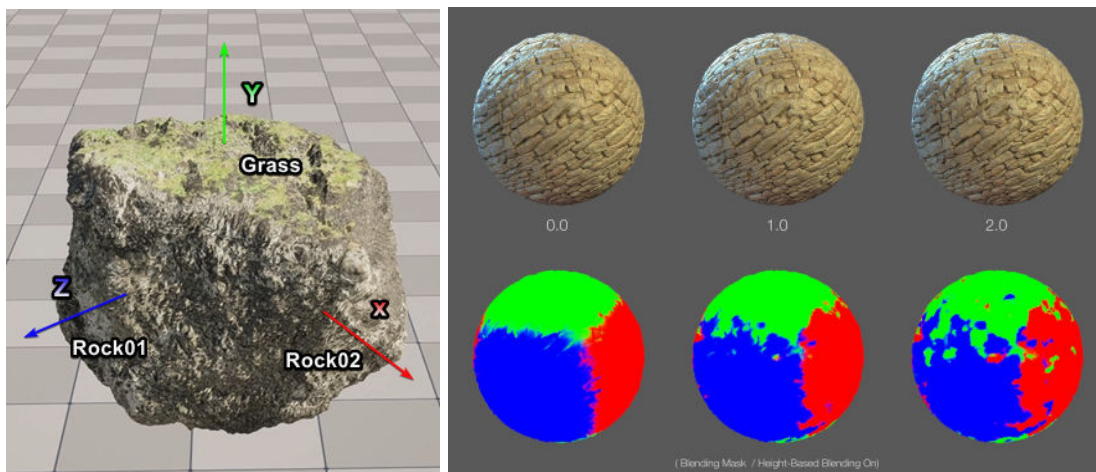
In the following example, we will guide you on how to set up a new voxel material in Editor, and how to make it active in-game in an asteroid.

Brief Intro

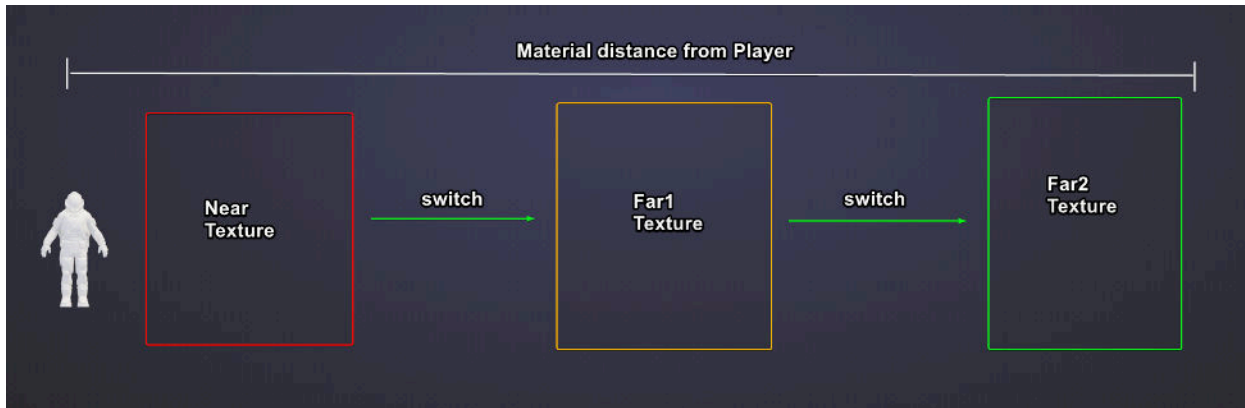
In the engine, we're importing a single separated texture for each color and grayscale (base color, roughness, metallic, height and so on). This means, we don't have to pack textures like Occlusion/Roughness/Metal in RGB channels like it happens in other engines. This is because the engine will automatically create composite textures based on the created material.

But we are not going to create a normal PBR Material. Instead we are going to make a Voxel Material.

Voxels are made of triplanar material projections. This means that we can map a texture three times with planar maps along the X, Y, and Z axes, and then blend between these three samples based on the angle of the face, using the one that fits best with the least stretching.



Besides that, our Voxel Materials have another 3 types of textures, based on the distance that is going to be seen by the player: **Near, Far1 and Far2**



Depending on the distance of the character to this voxel, it will transition from one texture set to another.

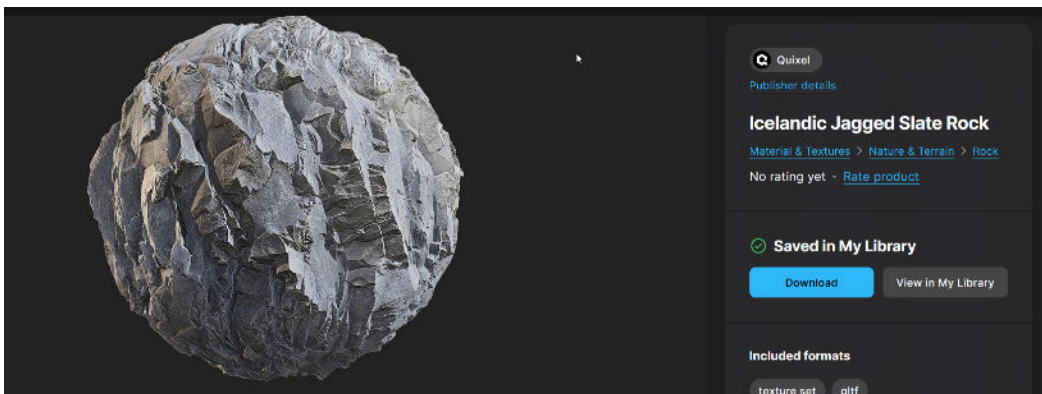
We could use different textures in the top(Y), front(Z) and side(X) but to keep things simple, we are just going to use the same texture on all axes, in this example, only varying by distance.

Process

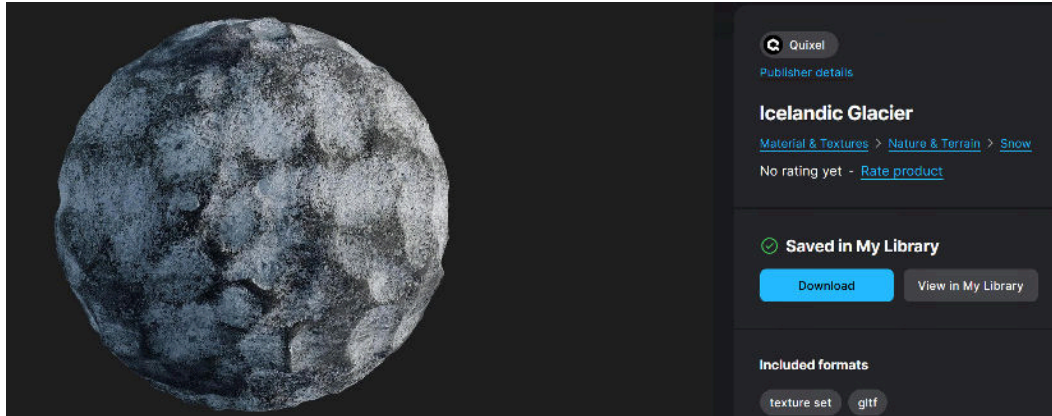
1. Search for 3 different materials to make the different versions by distance (Near, Far1 and Far2).

In our case we used 3 different materials from megascans to make our Ice Asteroid.

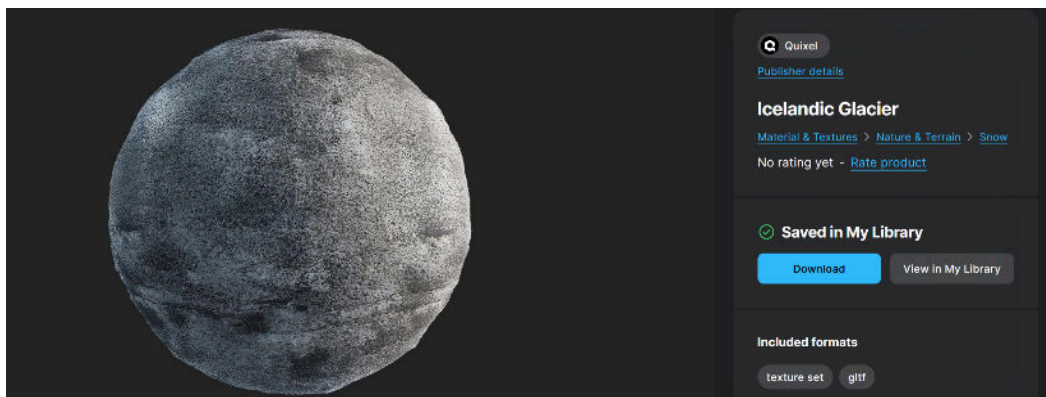
Near



Far1

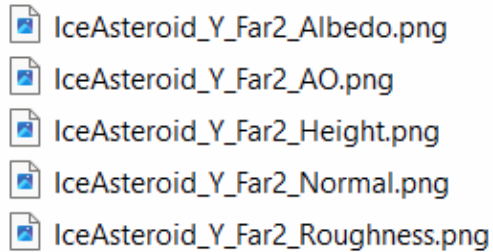
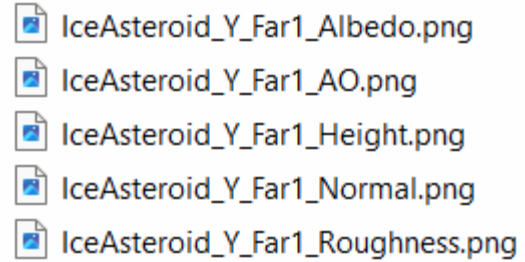
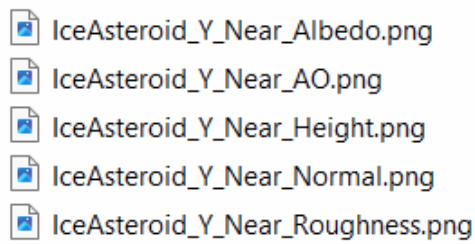


Far2



2. After downloading, we went to the albedo files of each texture set, and tweaked the texture as to look more like ice (this can be done in any image editing program)
3. Export the files as .png and make sure that the remaining files have the same file format (Editor will not allow material creation with different formats mixed)

Also give the name XX_Y_Near_type - X being your material name. Do this for all the files (respecting the distance purpose):




Quick notes:

- Our textures are driven by Height map range from -20cm (black values) to +20cm (white values)
- Normal maps are OpenGL

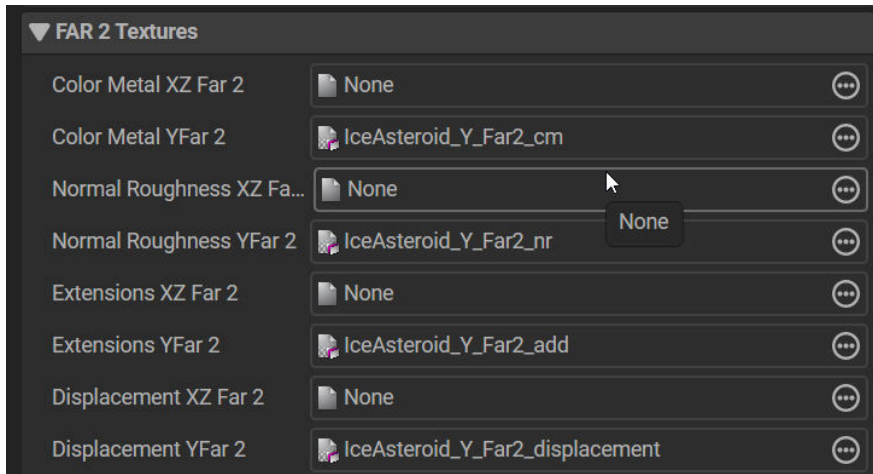
4. Open Editor and go to the Materials Folder : Vanilla/Assets/Environment/Materials
5. Create a new folder with your Material name: VoxelMod
6. Drag & drop your Textures to that folder
7. Select all, **Right-Click -> Create Asset -> Material -> Triplanar Single Material.**

Let Editor process the textures:

Processing assets: 4 in queue  Version 0.12.200.4112

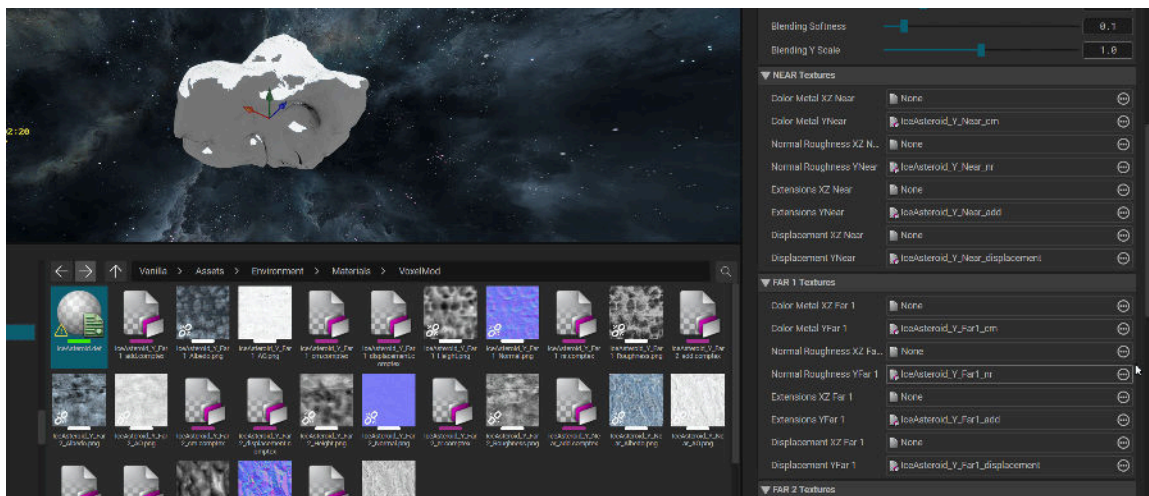
8. Going back to our Material's folder, we can now double click to preview the asteroid in render view.
But it doesn't look complete because it only has a Y projection texture to it (we

didn't provide more for X and Z projection):



9. Since we decided to use the same texture on all axes, we are going to drag&drop the same .comptex files above the corresponding empty slots.

You can click on the already assigned complex so that editor highlights the location:



10. Do this also for the Far1 and Far 2