



Mod Content Creation Guide



This guide will walk you through all the steps to create and import your own custom decorative building for Farthest Frontier.

1. Initial Setup

After purchasing Farthest Frontier, you can find an additional app in your Steam library called "Farthest Frontier Mod Tools." If not, you can also search for and install that app specifically on Steam.

When you install and launch Farthest Frontier Mod Tools, a new project for Unity will be installed on your local hard drive called "FFMod Project." You should be able to find it here:
steamapps\common\Farthest Frontier Mod Tools\FFMod Project

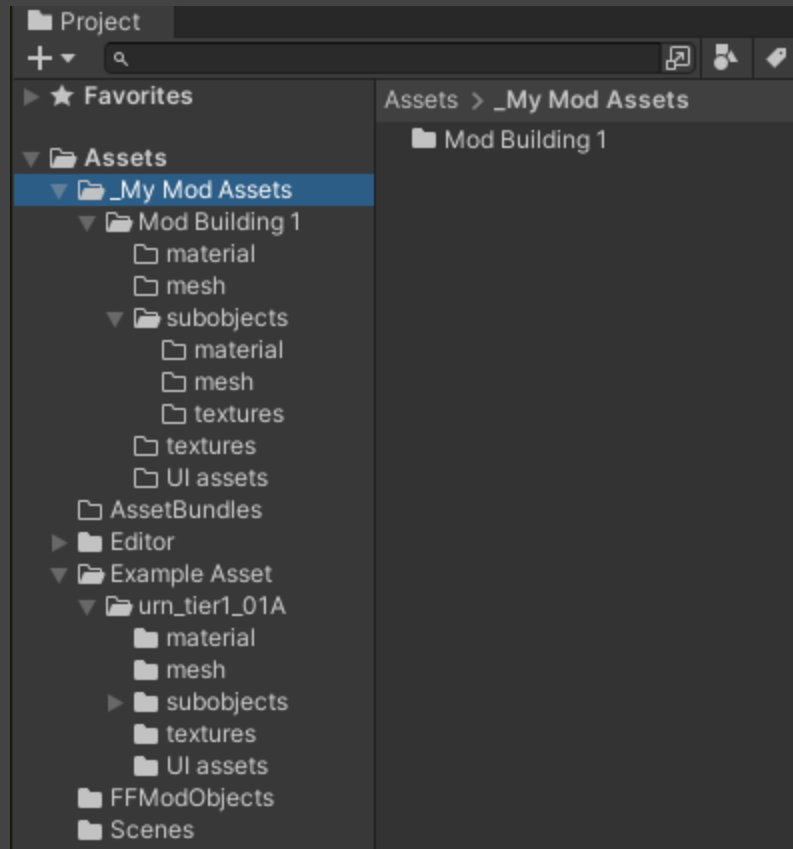
Before doing anything else, copy that FFMod Project folder somewhere else on your hard drive, in order to avoid interference in the future with the Steam client overwriting your local changes when the game is patched.

With the FFMod Project folder moved safely to another location, you are ready to begin!

Open the FFMod Project from the Unity Hub. If you don't have Unity installed, you can find information and links on that topic here:

<https://docs.unity3d.com/Manual/GettingStartedInstallingUnity.html>

Once the FFMod Project is open in the Unity Editor, open up the Project Browser to see how we have the mod files set up. There are two folders to take a look at: "_My Mod Assets" and "Example Asset." If you expand those, you'll see that within both of them are subfolders organized like this:



Each building you plan to create for a mod should have its own folder (we created a starter one called "Mod Building 1"). Within that, you'll see:

A Prefab: This is the main prefab for your building where you will connect everything else. A prefab in Unity is where all the individual assets of a building connect together: material, mesh, textures, gameplay parameters, and everything else.

Material: This will contain the material your building uses; this is almost always only a single material. Materials are where you plug in all the texture maps you've made, and then apply the material to your mesh.

Mesh: This contains the model reference for your building. Frontier meshes are saved in .FBX format.

Subobjects: Folders within here will contain models, textures and materials for any custom subobjects you create to accompany your building. Subobjects are a custom entity that we use in Farthest Frontier for small accessories that spawn alongside your building in game: typically things like plants, shrubs, or grass.

Textures: Texture maps for your building.

UI Assets: Images for UI elements associated with your building, such as icons, portraits, or anything else.

We'll get into all these files and their interactions in Unity in greater detail later.

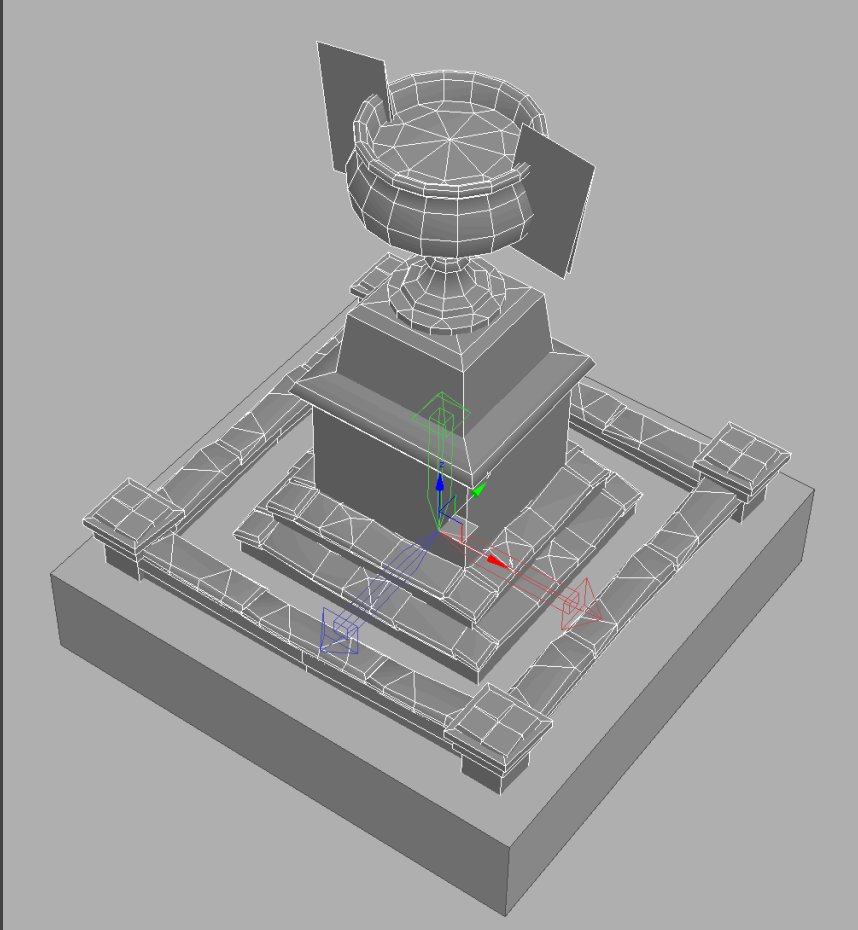
You'll see that the same file structure exists within the Example Asset folder. We included the decorative Urn building, already in Farthest Frontier, as an example case. You can see all the files associated with that building, broken down into the folders described above. A more detailed explanation of the role and interaction between those files is later on in this guide.

You can rename "_My Mod Assets" to whatever the title of your mod is going to be.

You can rename "Mod Building 1" to whatever the title of your first building in your mod will be. If you decide to include more than one building with your mod, each building will need its own file structure set up the same as "Mod Building 1" is by default.

But, the first step to making your custom mod building is to create the art assets themselves!

2. Modeling Your Building



Although sometimes the process can begin with making the texture maps for your building, in general, modeling is the first step. You can use any modeling app that you prefer to create your building model, as long as it can export an FBX file to send into the game.

File handling:

Because Unity will auto-import everything within a Project, we recommend saving your modeling working file somewhere else on your hard drive, to avoid unnecessary overhead when opening/saving things in Unity, or having Unity tediously re-import the entire modeling file every time you save it.

Scale:

The easiest way to get a sense of the scale of building models in Farthest Frontier is to import the example FBX of the decorative urn building. Once that's loaded into whatever modeling software you've chosen to use, you can calibrate the rest of your scene's scale around it. For reference, the decorative urn building is 1x1 grid squares in game, which equals 5x5 meters.

High-Poly Baking vs Low-Poly Modeling:

In Farthest Frontier, most of our buildings (typically those with repeating architectural elements) are modeled all in low-poly, and UV mapped onto flat atlas textures for easy repetition. However, some buildings (those that are more organic or character-centric, like our statues) are modeled in high-poly first in a sculpting app, and then baked down into a low-poly game ready model with textures.

Either workflow will work, it only comes down to whichever you prefer for your building.

Vertex Count:

Farthest Frontier buildings have a wide range of vertex counts that vary mainly with the building's scale. Small 1x1 buildings can be as low as a few hundred vertices, whereas very large, ornate end-game buildings can be over 10,000. You'll want to model enough detail to hold up when the player zooms in close with their camera, but try not to go overboard with modeled detail that doesn't add anything visual.

Vertex Colors:

Our shader that we use for building materials on Farthest Frontier includes the ability to add subtle wavy animation on a per-vertex level. This is how we accomplish things like laundry waving in the wind, or the leaves of plants gently rustling back and forth.

You can add this animation by painting more red into the vertex colors of the vertices that you want to animate. Black = no animation, fully static. The more red there is in the vertex color, the more that vertex will animate in game. Green and blue are both disregarded, don't worry about those. Typically only a small amount of red (up to roughly 20%) is enough to get the wavy animation you're looking for.

Important: many modeling apps default the vertex colors to white, which will mean crazy animation across all the vertices of your model! To avoid this, set the vertex colors of your entire model to black before exporting (and then, if you desire, add in red to those vertices you might want to animate).

Axes:

Many modeling apps are Z-up, meaning the Z axis points upward. However, Unity is a Y-up app. If your modeling app is not Y-up, you'll need to rotate the local axes of your model before exporting your FBX so that the Y axis points upward. Typically this is done by rotating the pivot of the model 90 degrees. This is easiest to do as the final step before hitting the export button.

LODs:

Currently, Farthest Frontier does not use any LOD (level of detail) technology on our buildings, so you don't have to worry about making those. If that changes in the future, we'll update this documentation to explain how it works!

Exporting:

Make sure your model's pivot is zeroed out at ground level, with the axes rotated to be Y-up, and then export it as an FBX with smoothing groups included. You can export it directly into the Mesh folder of your mod building within the FMod Project, and Unity will automatically import it for use in game.

3. Texturing Your Building

Although we're listing texturing as the next step in this guide, in truth it usually happens alongside the modeling process to some degree! Often it comes down to personal preference how much you enjoy modeling your building first, versus coming up with the textural details and modeling the building around those. Or maybe you're creating something like a statue, where you're baking all the detail down from a high-poly sculpt, and 75% of the texturing work is already completed just from that baking process.

Whatever the case, in the end you'll need to produce a series of texture maps that you'll plug into the building's material in Unity. Each texture map has a different purpose and different things to keep in mind when creating it.

Size:

Buildings in Farthest Frontier typically use texture maps at 1024x512 size. Very ornate, large buildings might use a full 1024x1024 map. Small buildings with very little texture needs, such as many of the decorative buildings, can share space on a combined atlas texture - as you can see in the example maps below that the decorative urn building uses, there are areas of the texture map used by other buildings beyond just the urn. Combining buildings on a shared atlas like this helps keep the texture use footprint a little smaller, but also makes it easier to keep materials looking consistent from one building to the next.

Diffuse Map:

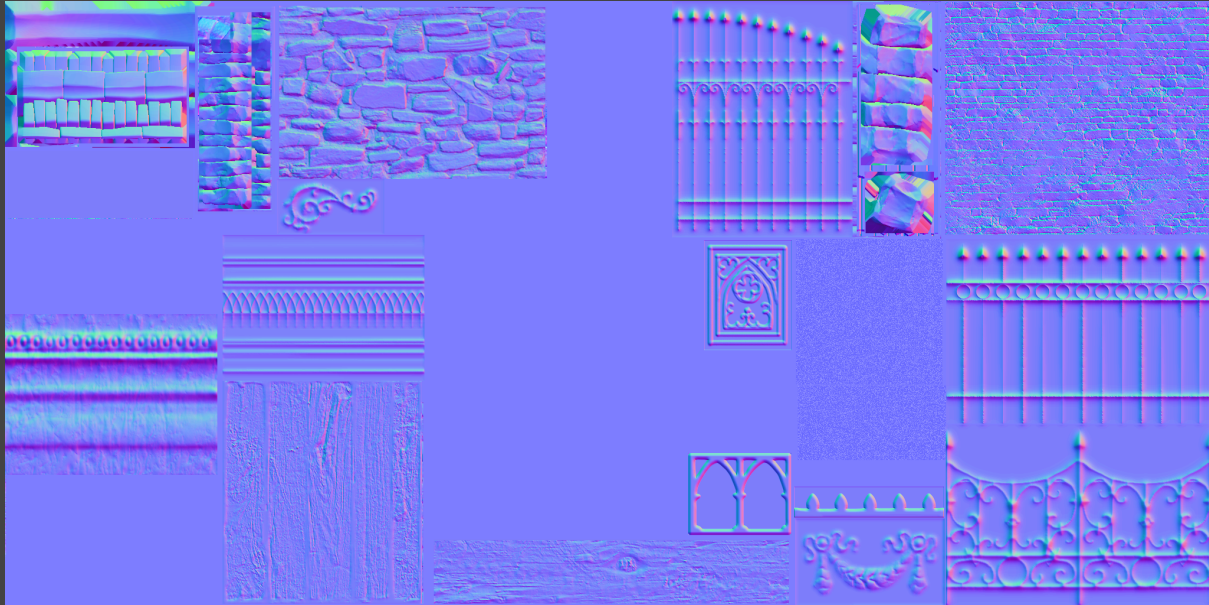


In Farthest Frontier, the majority of the visual information about a building's surface is contained in the diffuse map. This map shows the base colors, textures, and materials of the building. Because we use a relatively simple lighting setup on Farthest Frontier, we do include a good amount of supplementary lighting information within the diffuse map - soft shadows underneath overhangs, shadows on the underside of carved details, gentle highlights on the outer sides of rounded areas. If you're baking your maps from a high poly sculpt (or high poly building elements like bricks or shingles), you can get some of this lighting information from an AO map, and then multiply that into your diffuse map at whatever opacity looks good to you.

The in-game lighting environment in Farthest Frontier can be very bright, especially in summer, so we recommend keeping the overall values of your diffuse map on the lower side. Anything above 75% value will probably look extremely bright and glowy in-game - which maybe is something you want sometimes! But usually not.

If you want to do anything with transparency, that information is saved in the alpha channel of the diffuse map. Keep in mind that transparency in Farthest Frontier buildings is either fully opaque or fully transparent, no shades of gray between.

Normal Map:



How you generate the information in your normal map is up to you, and will probably depend on how you modeled the building originally. If you did a lot of high-poly sculpting of various elements of your building, or built architecture elements out of high-poly components like stones or bricks, you'll be able to bake that high-poly information down into a normal map. However, if you modeled everything in low-poly and jumped straight into texturing, you may find you need to run your diffuse map through something like Crazybump to generate normal map information. This process can actually yield good results if you put a little extra thought into it - you may want to make a quick grayscale heightmap, where dark = valleys and white = peaks, to send to Crazybump for more accurate results.

Note about the green channel: Farthest Frontier uses normal maps in OpenGL format, which are +Y (where the green channel has white on the "top" surfaces, and black on the "bottom").

Specular Map:



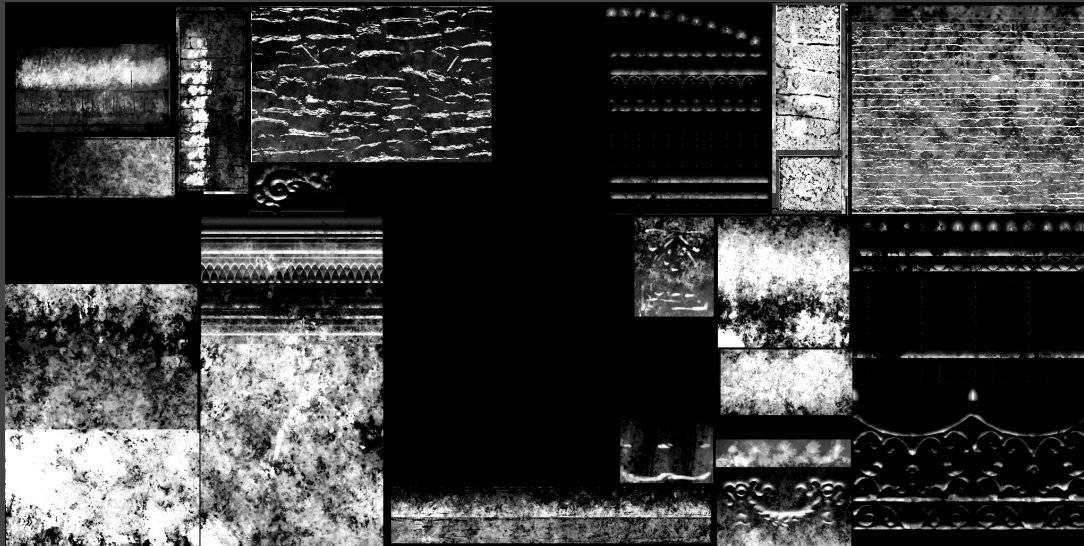
Our shader for Farthest Frontier buildings uses a specular map (with an accompanying gloss map in the alpha channel) to simulate highlights and reflections. The specular map shows what color and intensity the highlights will be on a given surface. We recommend keeping the values of the specular map very dim, around 20% or less, to prevent surfaces from exploding with unwanted brightness when the sunlight hits them in game. The exceptions to this might be in the case of very shiny materials like polished metal or glass.

Gloss Map:



The gloss map is a grayscale map contained in the alpha channel of the specular map that supports the specular lighting information by showing how glossy, shiny, and reflective various surfaces are. Black = fully rough, no glossiness at all; white = extreme mirror-like reflective surface. You can get interesting results with this - try experimenting with different values across both the specular and gloss maps to observe how they interact with each other.

Snow Map:



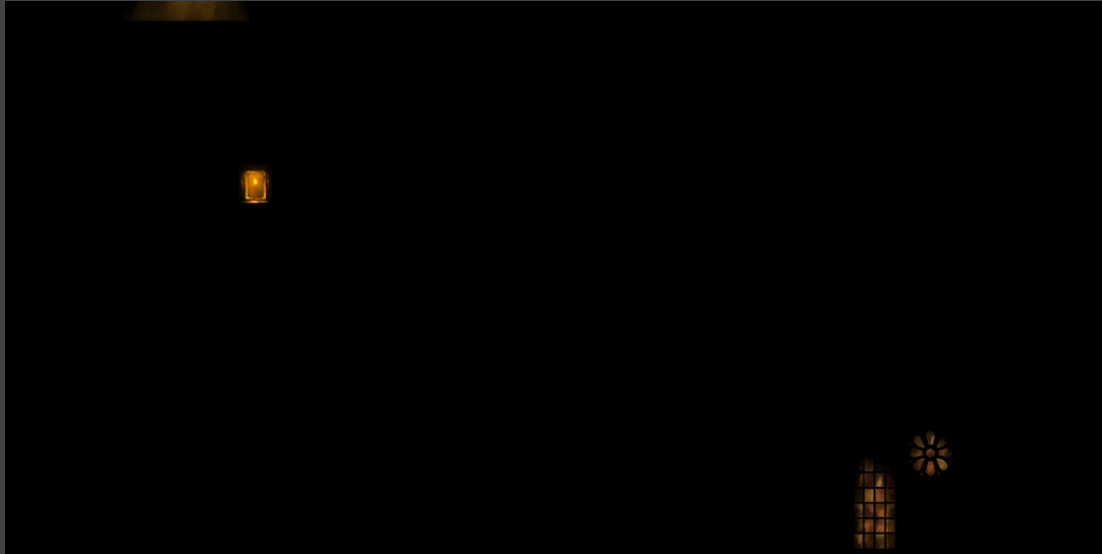
The snow map is a grayscale image that is stored in the alpha channel of the file - the RGB channels of the snow map file will be disregarded by the engine, so all that matters is the grayscale map in the alpha channel.

This map shows where snow is applied on the surface of the building in winter. The actual snow texture itself is separate altogether, and you'll want to use the "generic snow" texture we included with the mod materials for consistency with other buildings.

A great starting point for your snow maps is to open up your normal map, and look at the green channel. You can copy the grayscale of the green channel into your snow map and then crank up the contrast on it to the extremes, and this will naturally place white on the more up-facing areas of your building, with all your fine details of the normal map included. You can paint over it from there to get the snow map looking more natural and organic.

When you save your snow map, put the completed grayscale map in the alpha channel, and don't worry about anything that's in the RGB channels of the file. Only the alpha channel will be read by Unity, because we'll set the texture to "single channel" in the editor later on.

Glow Map:



Glow maps show what areas show light up in winter when the lights come on, and what colors those areas should be. Decorative buildings typically don't have any glowing areas, so they don't have any associated glow maps - the example image above is from another building. However, you may decide you want to have lanterns or lampposts in your decorative building, and so you will want to create a glow map where you show what color and intensity those areas should glow in winter.

4. Decal Textures

But wait, there's more! Just getting your building's model and texture maps done is only the beginning. The next phase of getting your building ready to go into the game is to create the decal textures for it. The decal textures are applied to the ground beneath and around your building in game. This is where you can add details like cobblestones around the entryways, or brick patios, or just grass sprouting around the bases of your fences and packed dirt yards.

Size:

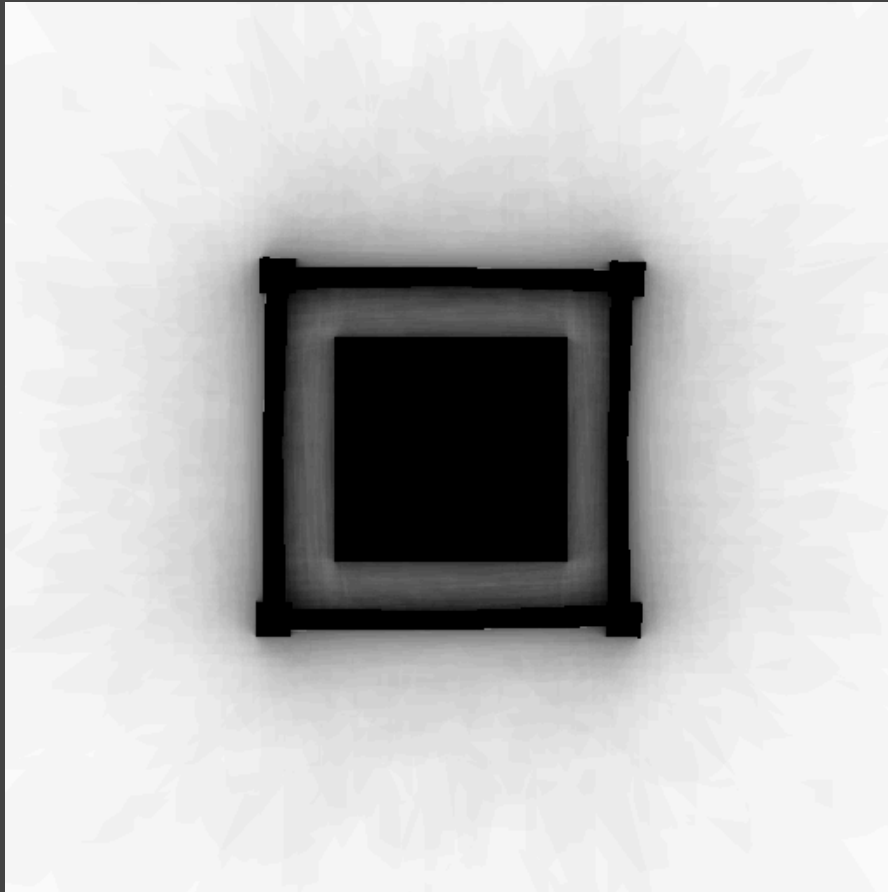
A good rule of thumb to follow for decal texture sizes is that 1x1 and 2x2 buildings (measured in in-game grid squares) can use 512x512 size decal textures. Anything 5x5 or larger will need a full 2048x2048 decal texture. And everything in between, which is the majority of buildings in Farthest Frontier, can use 1024x1024 size.

Our example file is 2048x2048; if your building is smaller than 5x5, you can crop it down to whatever is an appropriate texture size.

Important note: in Farthest Frontier, decals are always projected on the terrain at a size that is one grid square greater than the actual size of the building. So, a 2x3 building like the Fletcher will come with a decal that is projected at 3x4 size, centered on the building. This is to allow us to have decals that softly fall off 0.5 squares beyond the boundaries of the actual building, and avoids having ugly gaps between the decals of one building to the next.

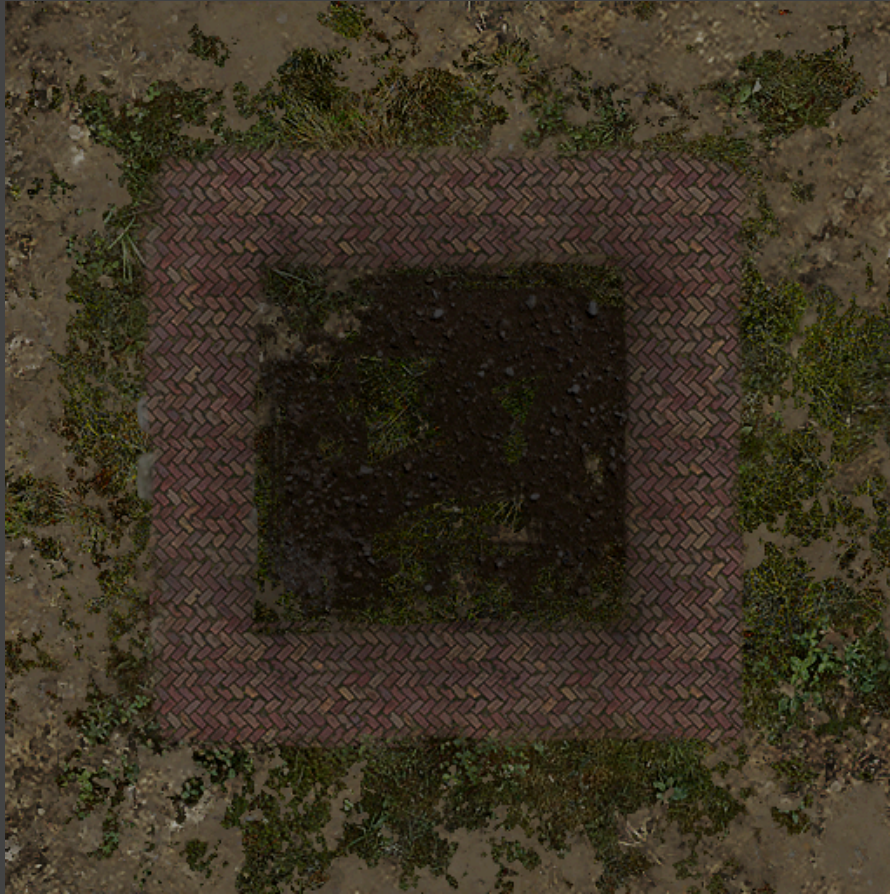
Baking an AO "footprint":

When diving into the decal textures in Photoshop (or whatever app you're using for 2D painting), you may find yourself wishing you knew exactly where the boundaries, or footprint, of your building was on the 2D image. There's a way to get that information, as well as some helpful shadows you can bake into your decal's diffuse map! Try baking an ambient occlusion map of the building mesh sitting on top of a ground plane that is sized to the same size as what the decal will be in game. The resulting map will look something like this:



You can bring that into your 2D painting app and multiply it on top of everything at whatever opacity you decide looks good. It will bring in some subtle shadows to help ground your building, and also help you see exactly where its boundaries are.

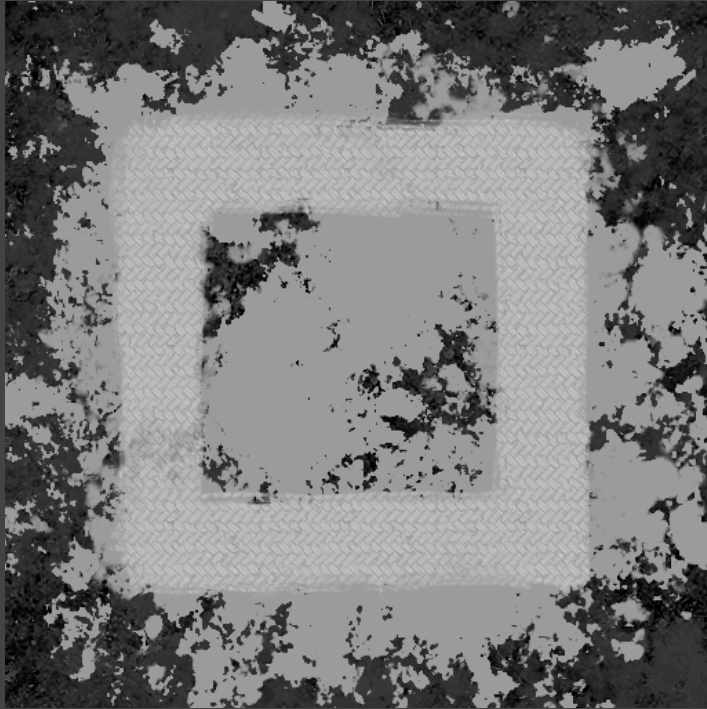
Standardized Decal Textures:



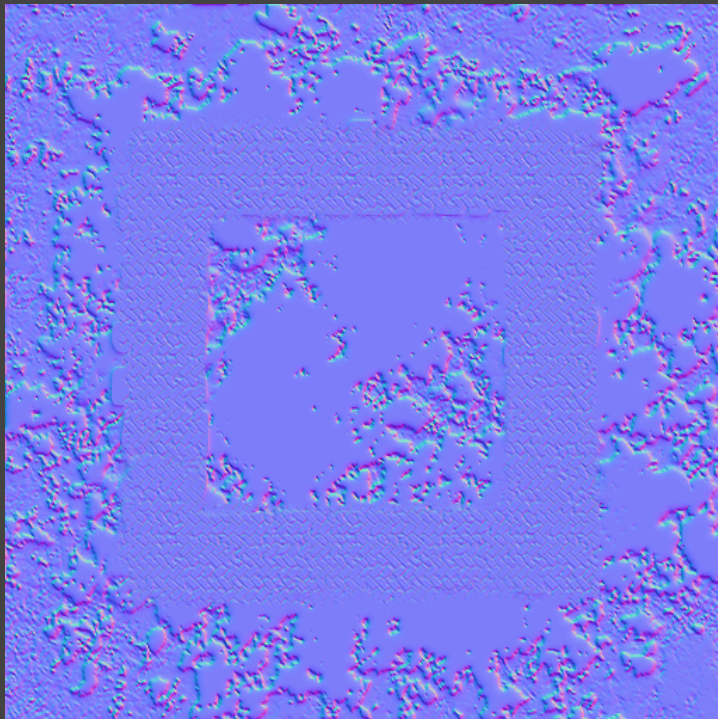
In Farthest Frontier, nearly every building's decal is built out of the same handful of stock textures. That means that all the dirt, all the grass, all the paving stones, all the brick etc, are all consistent from one decal to the next. Unless you specifically want your decal texture to look different, you'll probably want to use the same stock textures as all the other buildings in the game. Included in the materials we provided to modders, you can find a Photoshop file with layers including all the basic textures, titled **"FFMod_Decal_Texture_Layered_2048x2048.psd"**. All you need to do is paint in the masks for each layer where you want those to be; e.g. paint in areas where you want grass to show up, or paint rocky shapes on the rocks layer where you want to see rocks on the ground.

Your decal's diffuse map will also store the transparency in the alpha channel. You can use the transparency mask in our example file, or paint your own.

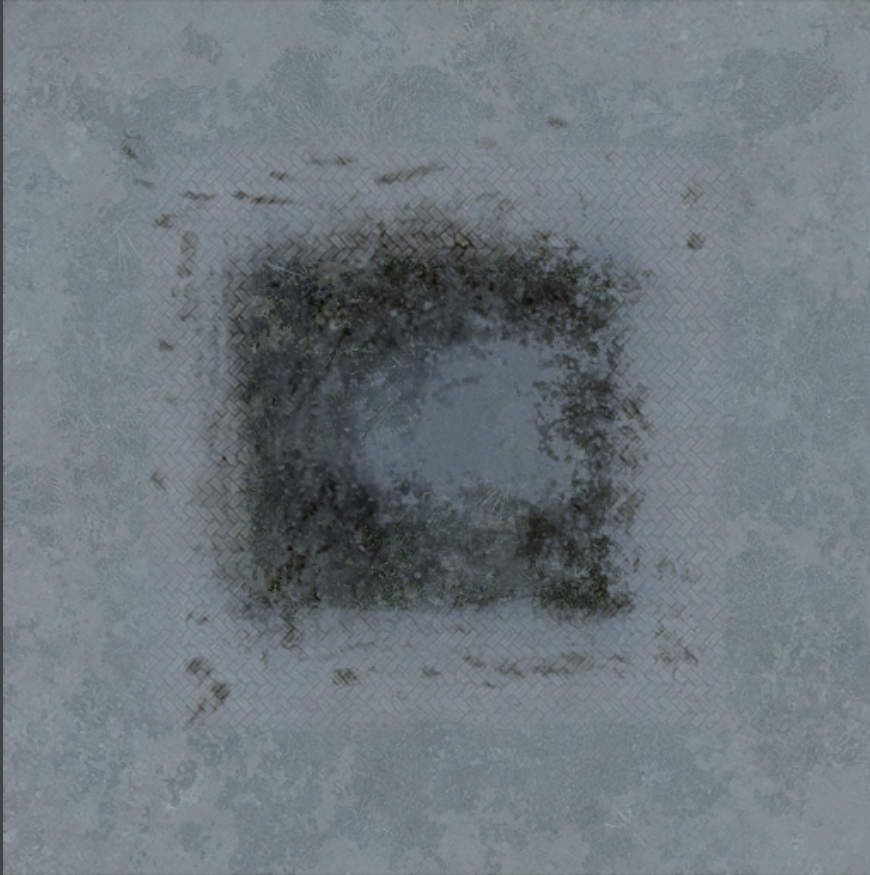
Generating Normal Map:



Once you have your masks painted the way you like, you can copy-paste those masks onto the corresponding layers in the "Heightmap" section of the Photoshop file we provided. You should get something akin to the image above: a grayscale image showing a height map of all your terrain decals in your decal. Then, run that heightmap through Crazybump or something similar, and you'll have a normal map for your decal ready to go!

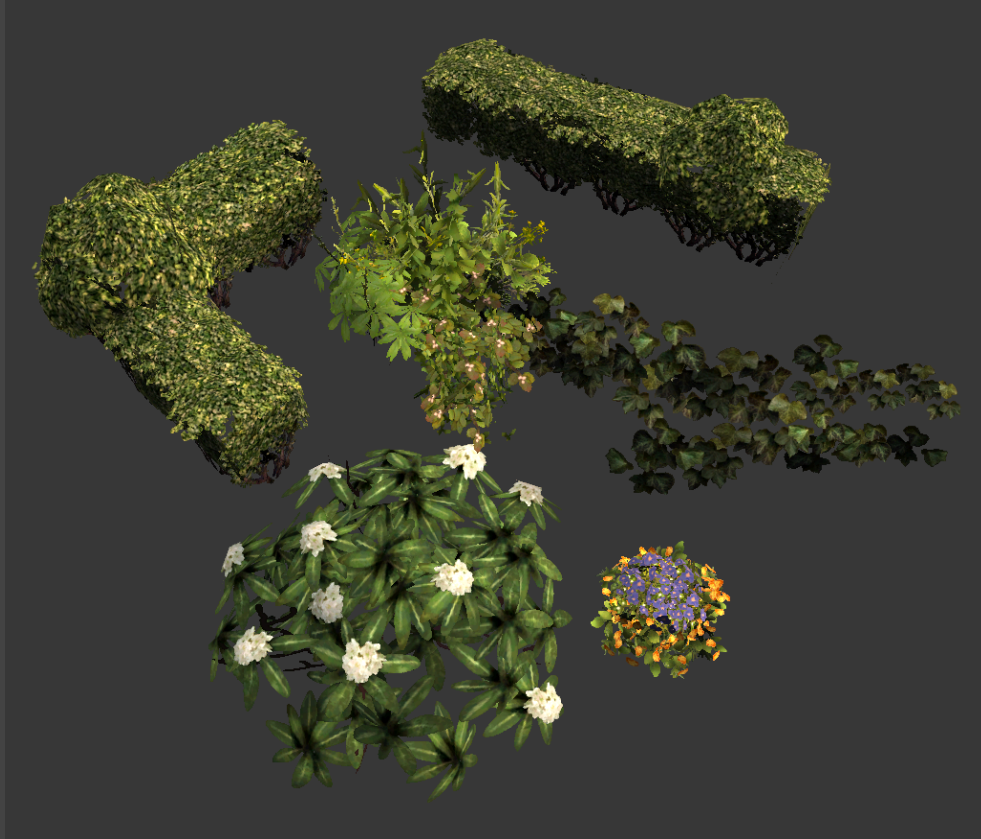


Winter Version:



Decals need a winter version, to show how they look covered in snow in winter. You'll see the "Winter" group of layers within the "Diffuse" section of the layered Photoshop file. Just turn on visibility for those winter layers, and then paint out in the mask where you want snow to disappear - you can paint footprints, cart tracks, or cleared ground this way, and the rest of the decal will be snowy.

5. Subobjects



Next it's time to think about what subobjects, if any, you'll place around your building in-game. We use the subobject system mainly for plants, shrubs, weeds, and sometimes trees, because subobjects can grow over time if desired, and can automatically snap to the ground when the building is constructed.

You can choose to model and texture your own subobjects, or you can choose some from the pack of subobjects we included in the mod materials. You can find their prefabs in the folder called "Subobject Library" along with all their models, textures, and materials.

If you decide you want to choose from the collection of included subobjects, you can skip the rest of this section!

If you model your own subobjects, you can treat them like you would a building, and export an FBX. Don't forget about the vertex colors! When it comes to texturing, if you want to take advantage of our seasonal subobject shader, you'll want to create 4 diffuse maps - one for spring, summer, autumn, and winter. They can each have their own alpha channels to show transparency changing from one season to the next.

6. Building Icon



In game, your new building will show up in the Build menu under the Decorations section. You'll need to make a graphic for the button within the build menu. For consistency, you can use the same background as all the existing buttons that we included with the mod materials. On top of that, you can create whatever graphic you want to represent your new building. Save out a **TGA file at 256x256**, and then another version that's slightly brightened. The normal version will be the "Up" state of the button, and the brighter version will be the "Down" state, for when the button is clicked.

7. Building Portrait



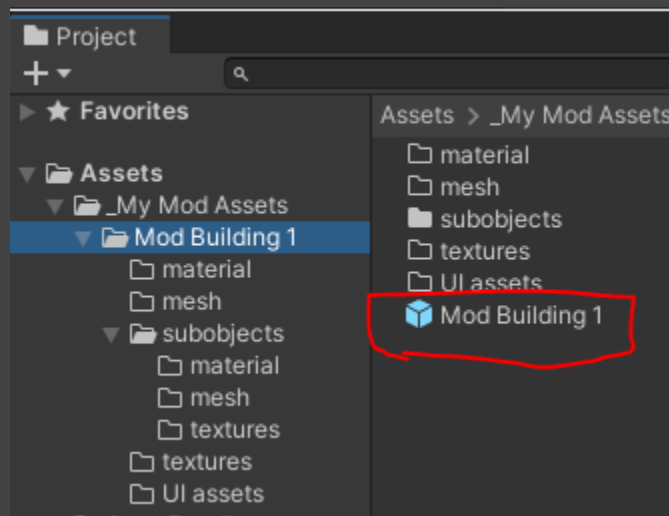
The last art asset you'll need for your building is a splashy "portrait" that shows up in the info window when a player clicks on your building. You can go about making this however you want to, but in the end you'll need a **PNG image with transparency at 512x350 size**.

8. Making Your Prefab in Unity

Time to fire up Unity and start getting your art assets connected together to get ready for use in game!

The first step is creating your building's prefab. The prefab is the home base for all your assets; everything gets plugged into the prefab in one way or another.

Create a new, blank prefab in Unity within your mod asset's folder, or use the blank one that we already created within the Mod Building 1 folder here::



Your prefab only needs to have 3 simple components on it:

- Mesh Filter, where you plug in the model of your building
- Mesh Renderer, where you'll connect your building's material under the Materials tab (more on that in the next section of this guide)
- and FF Mod Guid (Script). This is a component we use on all our buildings, and yours will need one too. **Important:** All Guids must be unique from one building to the next! Duplicate Guids will cause errors in-game, so if you duplicate your building's prefab to make a new one, make sure to regenerate a fresh Guid for it.

Once your mesh is plugged into the prefab (don't worry about the material yet), you'll be able to see it in Unity's Scene view.

You'll also need to include nav mesh blocking volumes to designate the collision boundaries of your building - in other words, where villagers can't walk through. You can do this with the volume called **NavMeshModifierVolume** that you can see on the example Mod Building 1 prefab. You can rotate, move, and scale this volume as needed to fit the collision boundaries of your building. If the shape of your building requires more than one volume to make accurate collision boundaries for, you can duplicate this volume as much as you need - though, try not to use volumes unnecessarily, to minimize performance costs. 3-4 should be more than enough for most buildings.

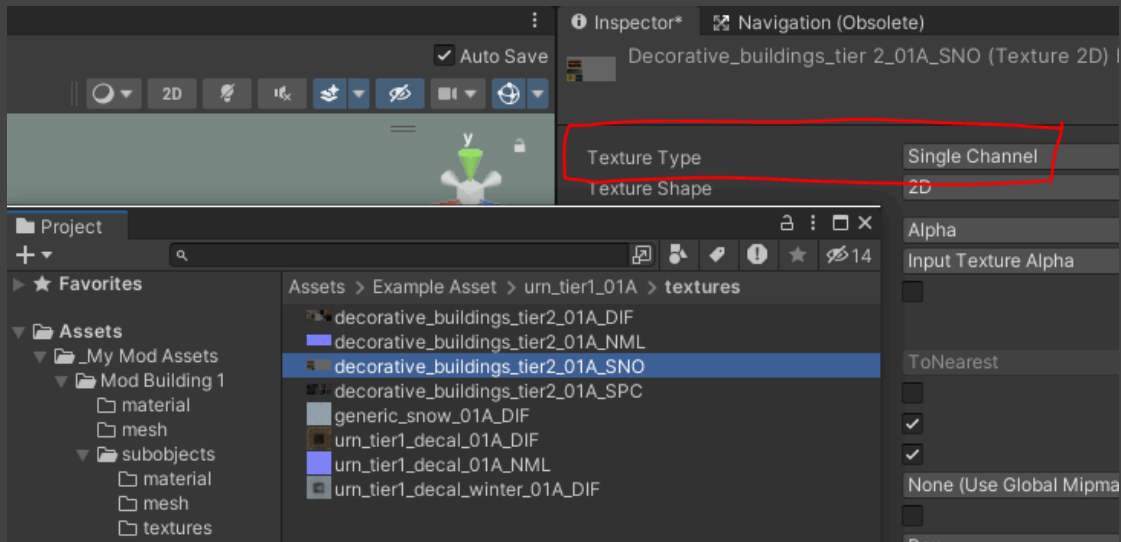
Note: use the "Scale" parameters on the FF Mod Nav Mesh Modifier Volume component to change the scale of the volume object instead of scaling it using Unity's scale tool in editor.

If you create other buildings for your mod beyond the first, copy/paste those NavMeshModifierVolume(s) from the first building over to any others.

9. Making Your Material in Unity

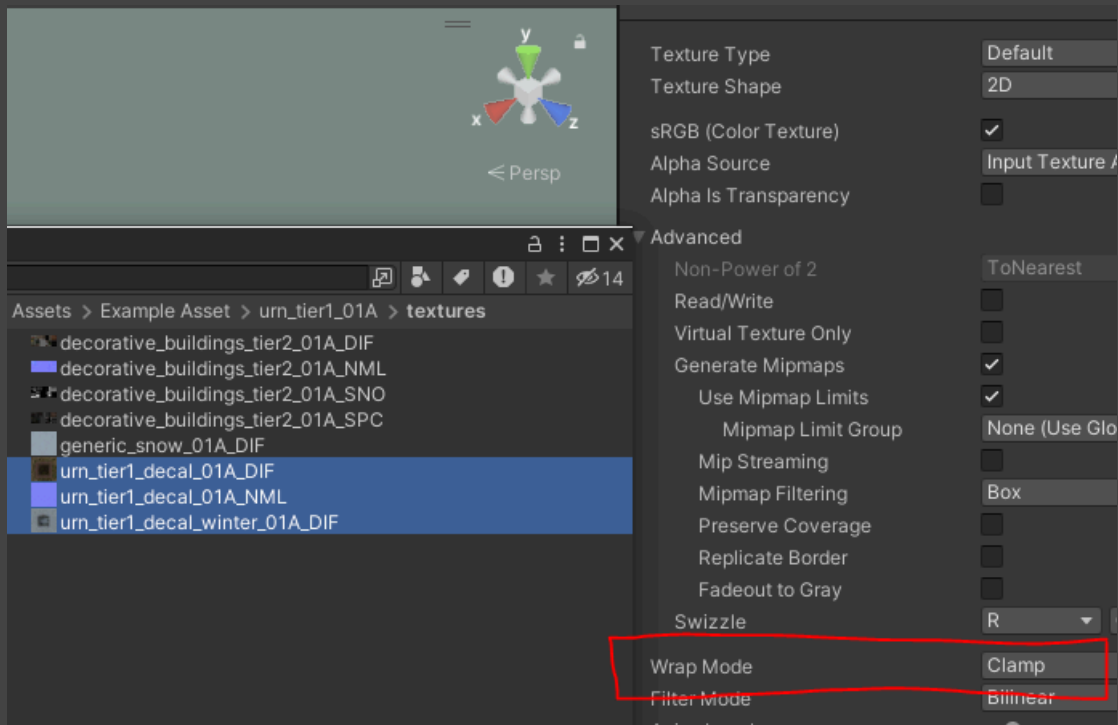
Now that the prefab's set up, it's time to create the material in Unity to plug all your textures into. But first, there's a couple things to do on the texture files themselves in Unity to get them to work correctly.

- Your Snow map needs its Texture Type to be set to Single Channel. Change that here:

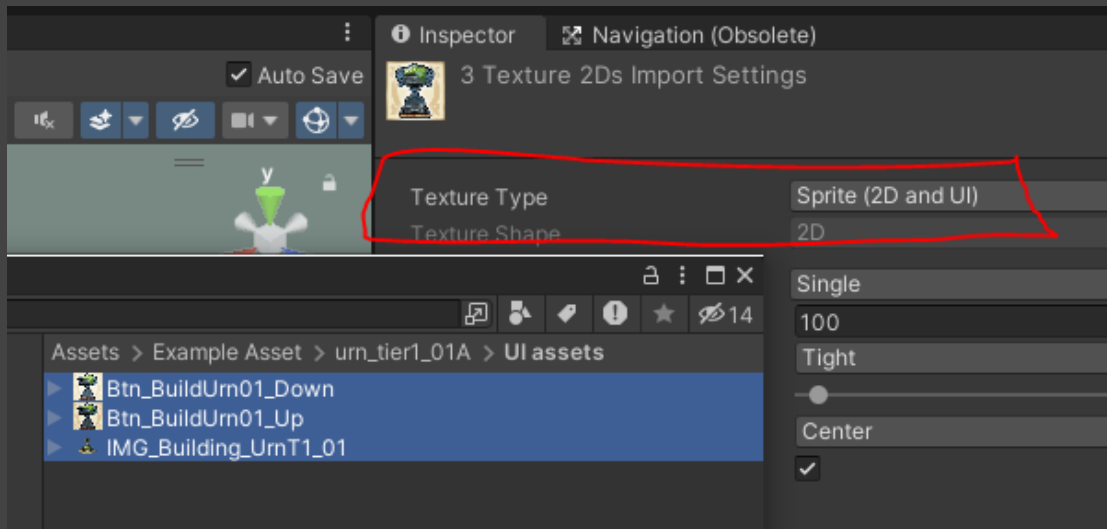


- Likewise, your Normal map needs its Texture Type to be set to Normal Map.

- Your decal textures need their Wrap Mode to be set to Clamp instead of Repeating. This is to make sure they don't tile across the landscape, and instead stop at the borders of the decal correctly. Change that here:



- And lastly, your UI assets (2 button images and 1 splash portrait) all need their Texture Type to be set to Sprite. Like this:



After setting your textures up properly, it's time to create a new material for your building.

In the Materials folder inside Mod Building 1, you can see a material we've created and set up for you already.

The shader type selected in the dropdown menu at the top is StandardWithSnow (Specular Setup). This is the shader we use for all Farthest Frontier buildings. If you create other materials for other buildings, make sure they always use this shader.

Albedo: This is your diffuse map.

Specular: Your spec map, which includes the gloss map (smoothness information) in the alpha channel.

Normal: Your normal map. Unity will prompt you to change the texture type to a Normal map when you connect it; say okay and confirm the change.

Snow Alpha Texture: Your snow map goes here.

Emission: Check this box if you are using a glow map to show things lighting up in winter on your building; then plug your glow map into the Color field that appears.

10. Adding Subobjects in Unity

Now it's time to move on to any subobjects you want to include in your building.

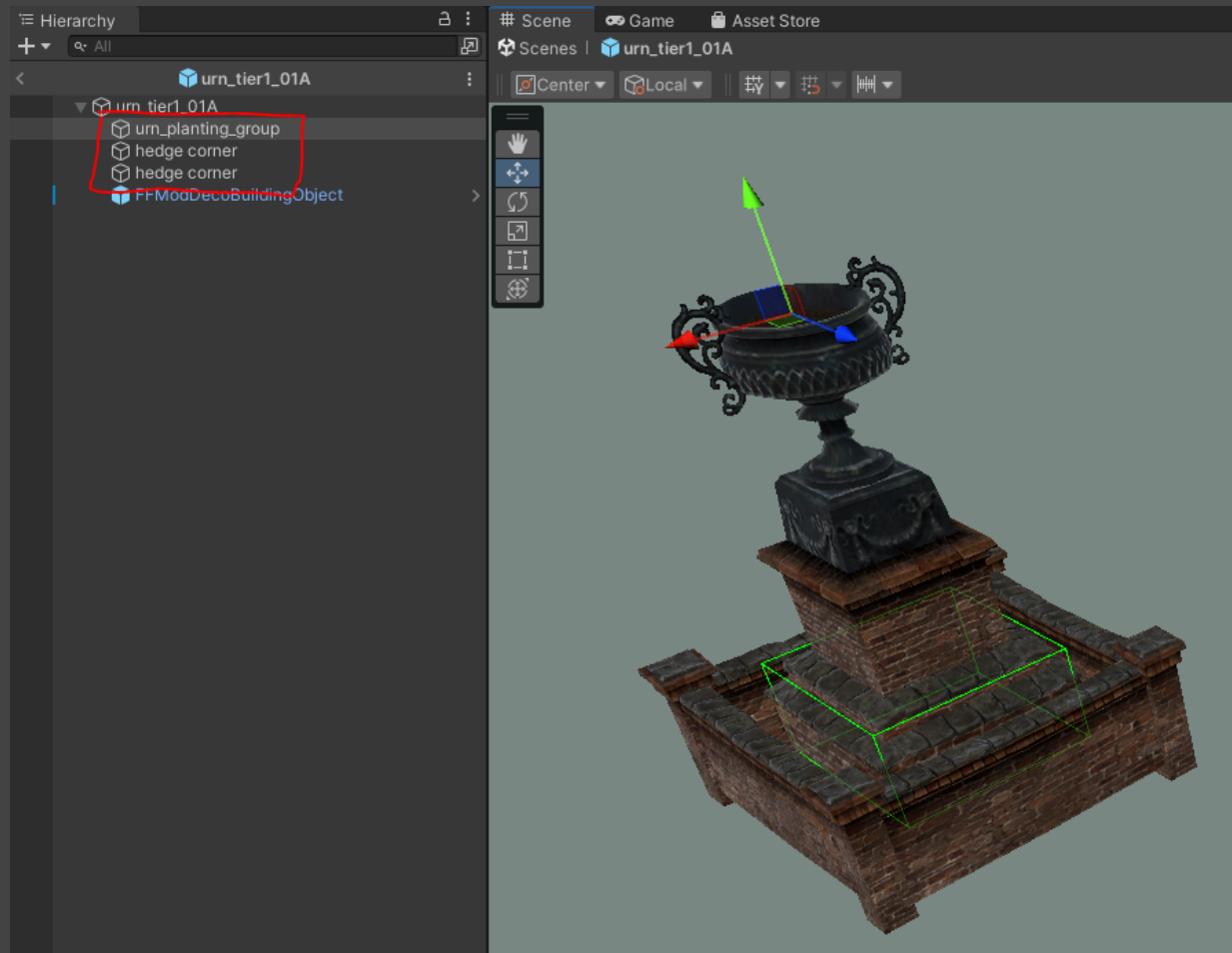
If you built your own custom subobjects, each of those will need a prefab of their own. Follow the same steps as above to create new prefabs. Subobject prefabs only need Mesh Filter and Mesh Renderer components.

Your subobjects will also need a material, and you can find one that we set up in Mod Building 1 > Subobjects > Materials. This material is set up to use our seasonal shader, so you can plug all of your seasonal textures into the four slots (spring, summer, fall, winter, descending from the top). If for whatever reason you don't want the seasonal shader on your subobject, you can duplicate the material that the building itself uses, rename it, and plug in your subobject textures there instead.

If you're using subobjects from our collection that we provided, find their prefabs in the project browser in Unity.

Subobject prefabs are connected to the main building prefab through a special component (which we'll get to in the next section); they're not attached directly as child prefabs. Instead, we use empty nodes as children of the main prefab that only contain transform information: position and rotation (and scale, which is disregarded). Then, those nodes plug into the subobject component, and in that component we designate which prefab will spawn at the location of those nodes.

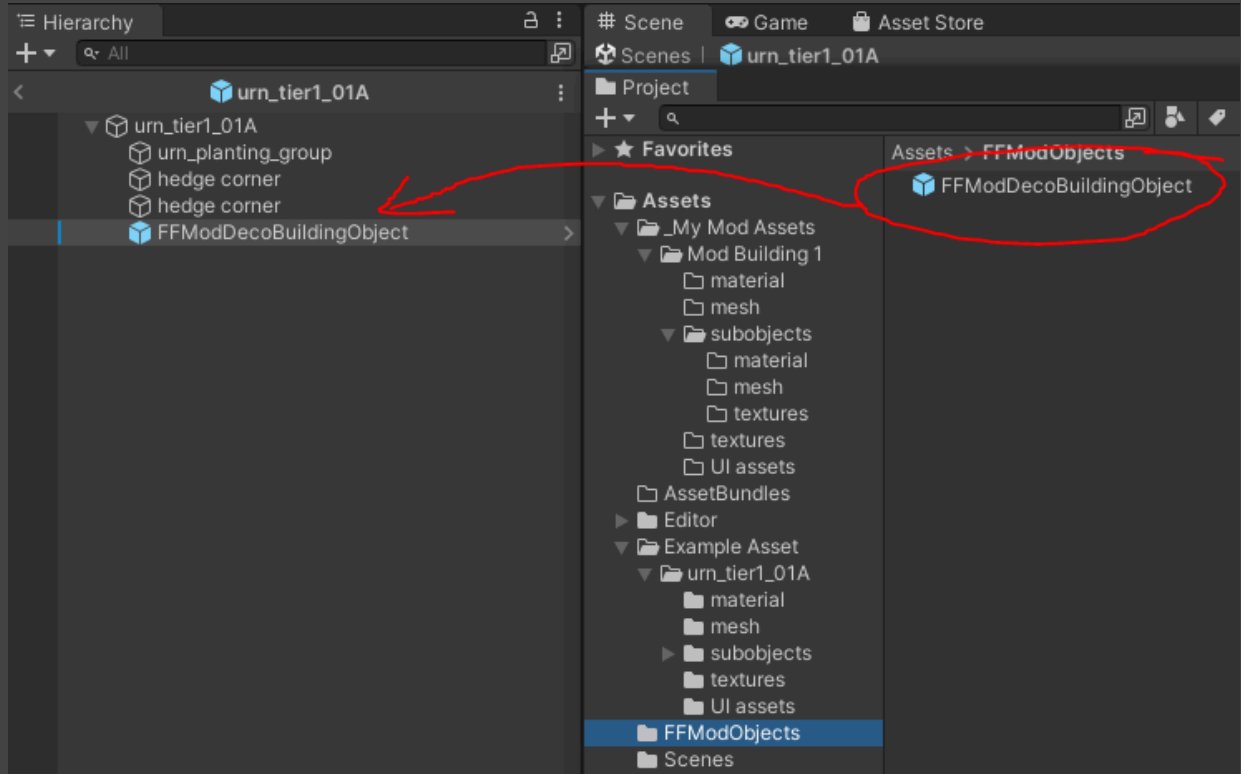
To place those subobjects, create blank nodes as children of your building prefab, name them whatever subobject you want to spawn there, and position and rotate them where you want your subobjects to spawn in, like this:



If you find placing the empty nodes "blind" to be too difficult, one thing you can do is drag the actual subobject prefabs into the scene, position them how you want them, then copy their transforms and paste those transforms onto your empty nodes. Once you're done, delete the subobject prefabs.

11. Adding FModDecoBuildingObject

The last step before exporting your new mod asset is to add the FModDecoBuildingObject to your building prefab as a child. You can drag an instance of it from the FModObjects folder that we provided, and make the new instance of it a child of your building prefab, like this:



Then, select your new instance of the FModDecoBuildingObject that now sits within your building prefab, and you can begin editing all of the parameters in the Inspector. Let's get into what each component does:

FF Mod Deco Building (Script):

This is the largest component with many subsections.

Building Identifier: This is a unique identifier for this building. Name it NameOfMyMod_BuildingName to ensure its identifier tag will be unique. Example:
MyUrnMod_DecoUrn01

Building Prerequisites

BP_Crate Identifiers: You can choose from a dropdown menu which buildings you want to be prerequisites for your mod building. If you want multiple requirements, you can add them to the list with the + button.

Placement Settings

Grid Size: The size of your building in in-game grid squares. Remember that 1 grid square in game is equal to 5 meters.

Construction Settings

Max Allowed: Here you can designate the maximum number allowed of your building in a single game. If you want it to be unlimited, set this to zero. By default, buildings in Farthest Frontier do not have limits, with a few exceptions like the Trading Post.

Building Materials: A list of the materials a player will need to spend to construct your building. You can add or subtract from the list and choose which materials you want from the drop-down menus.

All of the following fields can safely be left at their default values without any issues:

Default Builders: The number of builders that will construct your building by default.

Max Builders: The maximum number of builders that will work on constructing your building.

Work Required to Construct: The amount of work, in work units, required to construct your building once all materials have been delivered to the construction site.

Gold Required to Relocate: Here you can specify how much gold a player needs to spend to relocate your building, if desired.

Work Required to Deconstruct: The amount of work, in work units, required to unbuild your building.

Clear Details Border Width: This value specifies how wide of a margin around your building the game should clear terrain details, such as grass, weeds, branches etc.

String Tags

Building Name String Tag: This is where you can write the name of your building as it will appear to the player. Note that this needs to be translated into all the languages that Farthest Frontier is available in. If you do not plan to localize your building's name, each language still needs a placeholder name (e.g. the name of the building in English).

Building Description String Tag: This is your building's description text that will appear to the player when they click on the building in game. Same as above, you will need to translate this text as well, or enter placeholder entries for other languages.

UI Assets

Build Button Up: Here's where you plug in the 256x256 image that you made earlier to be your building's button art in the in-game Build menu.

Build Button Down: Same as above, except this should be the brighter "down" state version of the button art.

Building Portrait: Your 512x360 art for the building's in-game portrait when a player clicks on it.

Misc

Highlight Similar Buildings on Select: If you want the player to see all instances of your buildings highlighted in game when they select a single one, check this box.

Weight to Catch Fire: If you want your building to be especially flammable, increase this slider.

FF Mod Painted Terrain Area (Script)

This component is for displaying your buildings decal texture on the terrain below it.

Painted Terrain Settings

Width and Height: These are the dimensions of your building's decal texture. These values are in meters, not grid squares. Also, remember that decals should be scaled to 1 grid square greater than the actual footprint of the building - so if your building is 2x3 grid squares, the decal should be scaled to 3x4, which measured in meters would be 15x20.

Texture Data

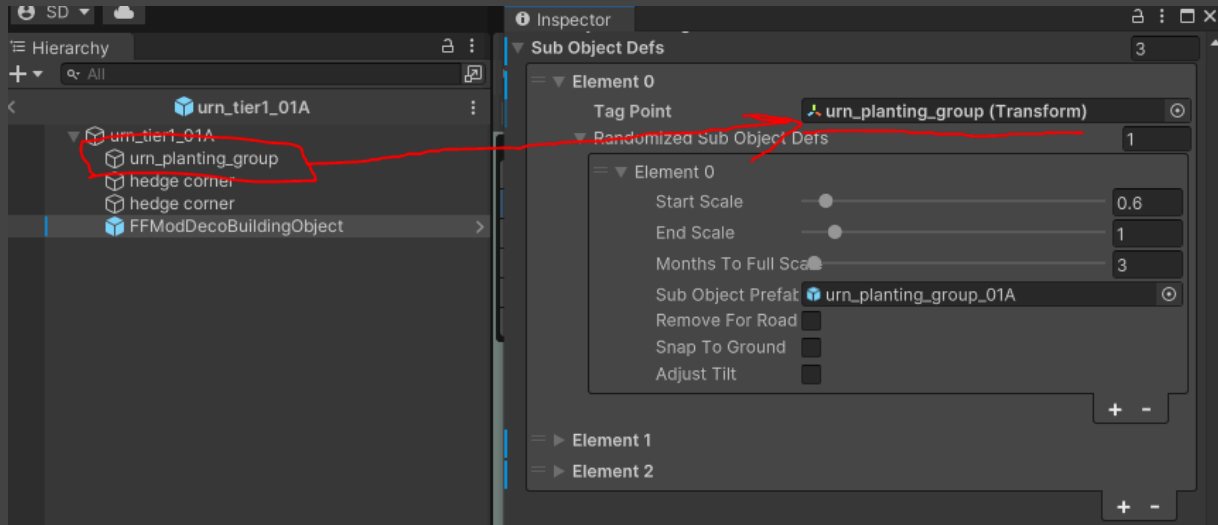
Plug in your decal textures to these fields. "Winter Normal Map" is currently unused by any of our buildings in Farthest Frontier; you don't need to worry about that one.

FF Mod Sub Objects (Script)

This component is for connecting any subobjects that you want to include around your building.

Sub Object Defs

Each element of this section is for a single subobject node. Drag the node from your prefab into the "Tag Point" field on the element, like this:



Then, designate which subobject prefab you want to appear at that node's location with the "Sub Object Prefab" field. Typically this is what sort of plant you want to grow there.

Start Scale, End Scale, and Months to Full Scale all determine the growth rate of the subobject prefab.

Remove for Road refers to whether this subobject should be cleared by a road being built alongside your building (typically unchecked).

Snap to Ground refers to whether the subobject should always spawn at ground level of the terrain regardless of hills (typically checked).

Adjust Tilt will make the subobject tilt to conform to the angle of the ground if this box is checked.

Note that, if desired, you can add additional Elements within Randomized Sub Object Defs. This allows you to have the game choose randomly from multiple subobjects (typically different variants of plants) to spawn at a single node. This allows you to avoid having every single instance of your building look exactly the same, with the same plants growing around it.

FF Mod Damageable Component (Script)

This component covers the combat stats of your building. You can safely leave these default values untouched if you desire.

Life: How much damage the building can withstand in combat before it's destroyed.

Base Armor: You can leave this as default for decorative buildings.

Base Search Range: Same as above, deco buildings should leave this unchanged.

Can be Targeted: If you want your building to be untargetable in combat (an example would be Plazas), uncheck this box.

Combat Target Type: This is where you designate how much you want raiders to prioritize attacking your building.

FF Mod FOW Revealer (Script)

This component specifies how much fog of war you want your building to reveal around it. X and Y are radius values; the building will reveal everything within a range of X, and partially reveal everything within a range of Y. You can leave the default values untouched if you desire.

FF Mod Box Collider (Script)

The box collider is a simple cube that designates the boundaries of the building for selection and projectiles. Typically this cube generally matches the visual boundaries of your building. Use the Size values within this component to change the dimensions of the cube to fit your building within it.

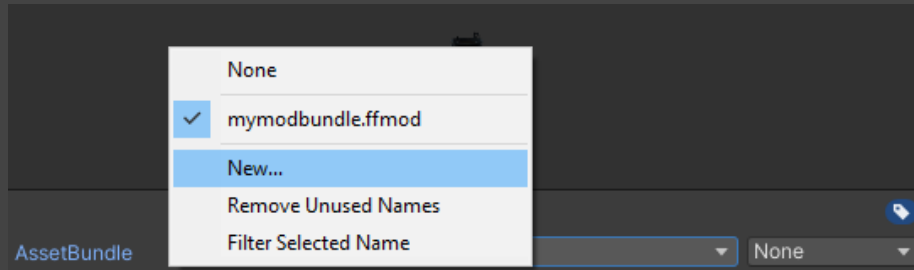
12. Exporting Your New Mod

Finally, you're ready to bundle up your assets and export them as a mod to get them playable in Farthest Frontier. Before you take those final steps, here's a quick checklist to make sure everything's ready to go:

1. Visually inspect your building prefab in Unity's scene viewer - does everything look okay? Check for unwanted vertex animation, smoothing groups, and flipped normals on your model. Check that your textures are displaying correctly.
2. Check your subobjects. Remember that you connect the subobjects with empty nodes, parented to your building's prefab, which are placed where you want the subobject prefabs to spawn in game, and then connect those within the Subobject component on FModDecoBuildingObject. You don't want any actual subobject prefabs sitting inside your building prefab, only empty nodes that are connected in the component. Remember you can refer back to the Urn building as an example of how this setup works!
3. Check your NavMeshModifierVolume. You should see the purple wireframe of the volume, which is attached as a child of the building's main prefab. To minimize bugs, make sure the Scale of the actual volume is 1.0, and that you've scaled it using the FF Mod Nav Mesh Modifier Volume component's values instead.
4. Check everything in FModDecoBuildingObject. Make especially sure that your decal textures are plugged in correctly and scaled correctly - remember that the scale of the decal is 1 grid square larger, on both X and Y axes, than the footprint of the building. Also check that your subobject prefabs and nodes are plugged into the Subobjects component.
5. Check your texture assets and their setup. Check that your snow map is set to Texture Type: Single Channel, and your decal textures have their Wrap Mode set to Clamp.
6. Check the FF Mod Guid component on your building prefab. Make sure this Guid value is unique; if you have multiple buildings in your mod, they all need to have unique Guid values.

Now it's time to bundle up everything for export.

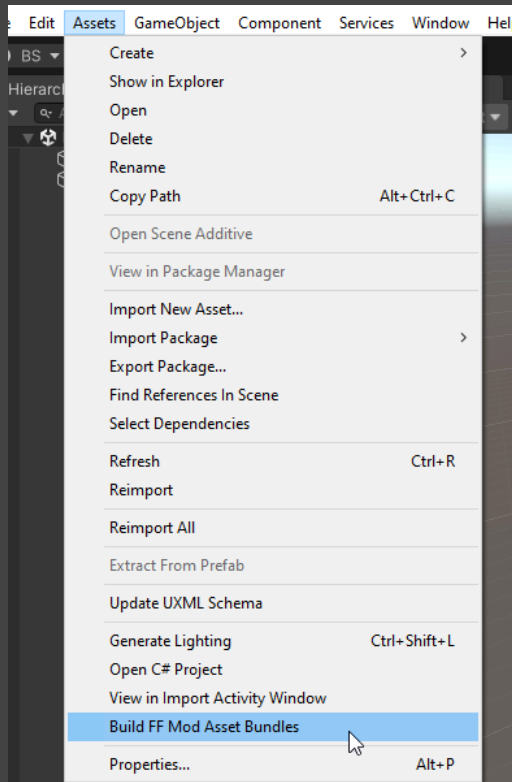
Select your building's prefab, and in the bottom of the Inspector pane, you'll see a dropdown menu labeled AssetBundle. From this dropdown menu, you can select "New..." and create a new asset bundle titled whatever you want to call your mod:



Make sure to include ".ffmod" at the end of your asset bundle's name, matching the format of the example one we made called "mymodbundle.ffmod". It will make finding it much easier later on and keep things consistent.

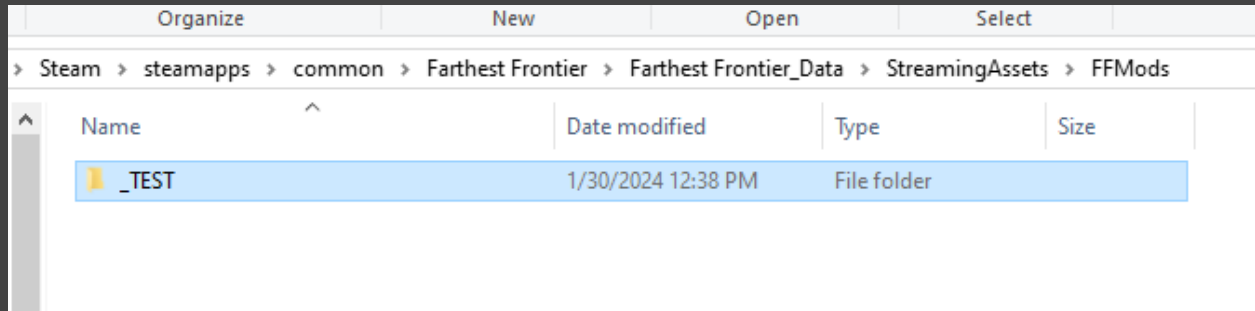
Then, select any other buildings beyond the first that you want to include in your mod, and mark their prefabs to be included in the same asset bundle.

Once all your building prefabs that you want to include in your mod are marked, go to the Assets menu at the top, and select Build FF Mod Asset Bundles:



This will bundle up all the building prefabs you had marked, including all of their associated assets, subobjects, and information, and create a .ffmod file within Assets/AssetBundles. It should have the same title as you tagged your buildings with for export. You'll notice that there is a Manifest file also, which is a list of everything included in your mod. The actual mod file is the FFMod file type.

You can test your mod now! In windows explorer, drop that .ffmod file into steamapps\common\Farthest Frontier\Farthest Frontier_Data\StreamingAssets\FFMods_TEST:

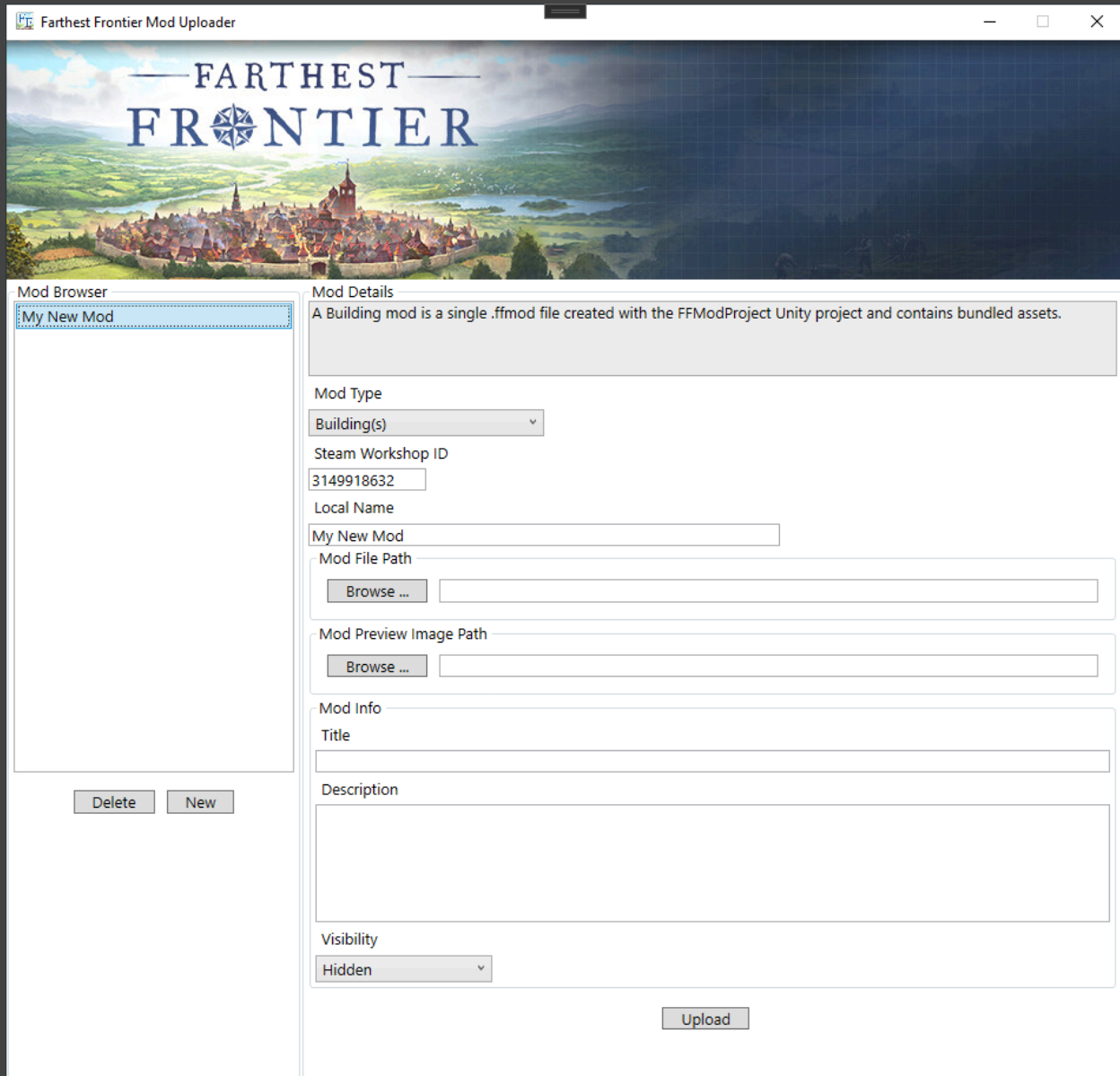


You will need to create a new folder called "_TEST" within FFMods. **Important: If you put your .ffmod file anywhere except a folder called "_TEST", it will be automatically removed by Steam when you launch Farthest Frontier.**

When you launch Farthest Frontier, any new buildings contained within that .ffmod file will now show up in game and be buildable. Test everything out and make sure there aren't any bugs!

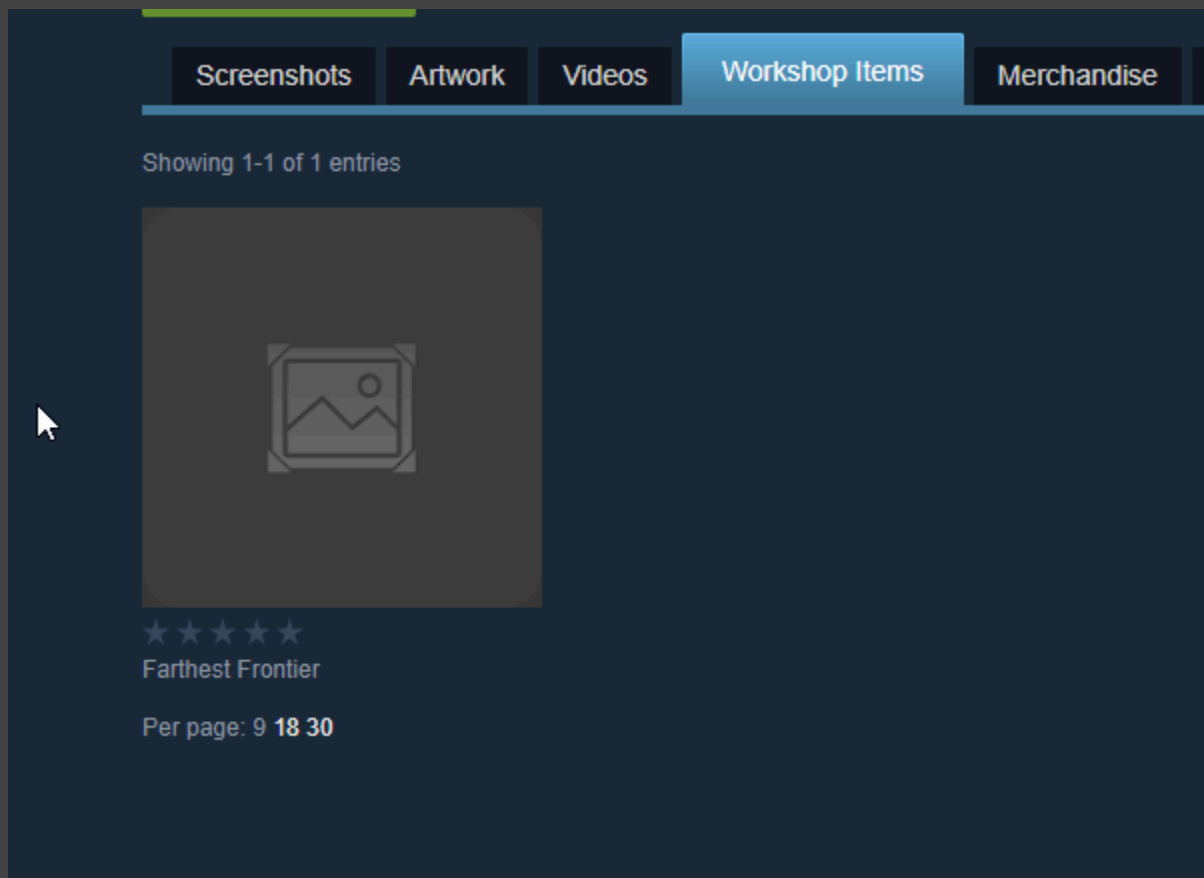
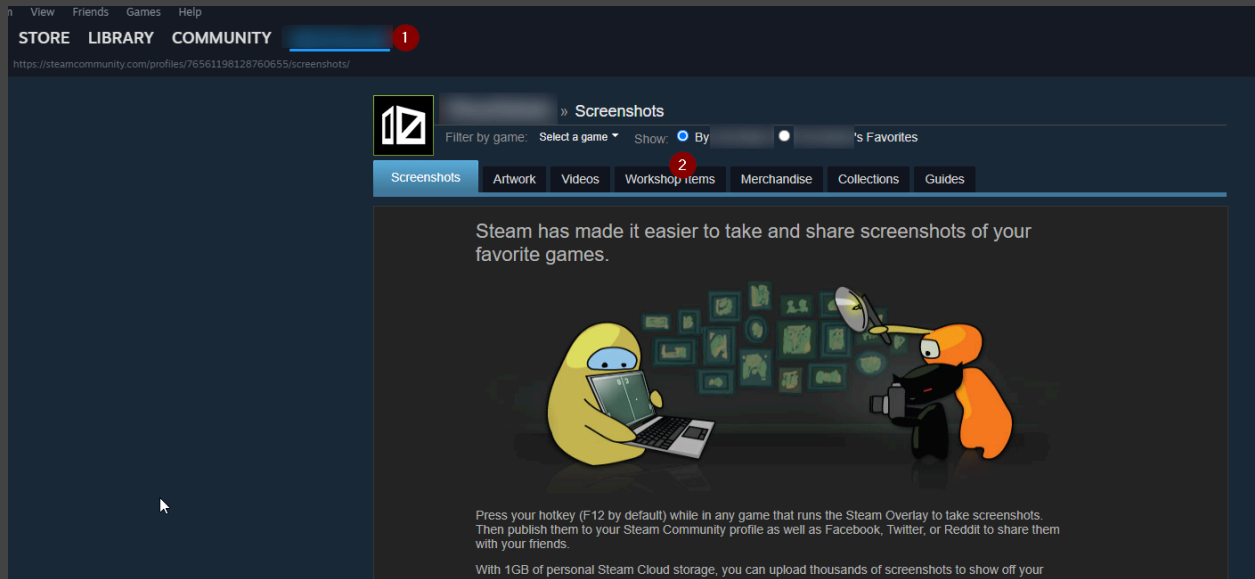
Uploading to Steam Workshop

Included in our mod development materials is a small standalone app called "Farthest Frontier Mod Uploader." At this point, you're ready to launch that app and use it to help upload your mod to Steam workshop.



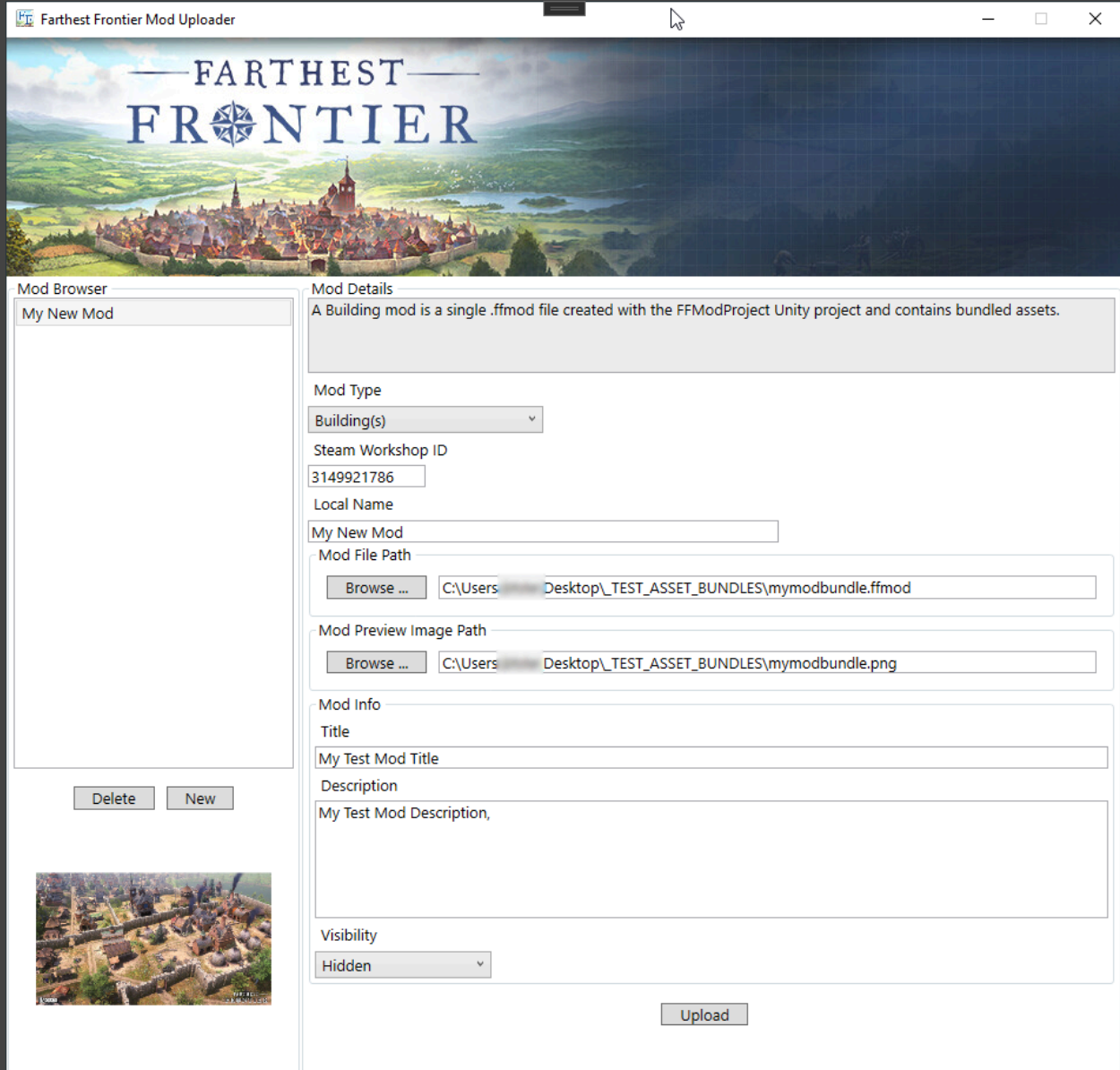
The screenshot shows the "Farthest Frontier Mod Uploader" application window. The title bar reads "Farthest Frontier Mod Uploader". The main header features the game's logo "FARTHEST FRONTIER" over a scenic landscape image. The interface is divided into two main sections: "Mod Browser" on the left and "Mod Details" on the right. The "Mod Browser" section contains a list with one item, "My New Mod", and buttons for "Delete" and "New". The "Mod Details" section includes a descriptive text box, a "Mod Type" dropdown menu set to "Building(s)", a "Steam Workshop ID" text box containing "3149918632", a "Local Name" text box containing "My New Mod", "Mod File Path" and "Mod Preview Image Path" sections each with a "Browse ..." button and an empty text field, a "Mod Info" section with "Title" and "Description" text boxes, and a "Visibility" dropdown menu set to "Hidden". An "Upload" button is located at the bottom right of the "Mod Details" section.

When you first launch the app, everything will be blank. Click the "New" button and the app will create a placeholder asset in the Steam workshop. You can see this placeholder workshop asset in your Steam client - in your user profile, under the Workshop Items tab, will be a new Farthest Frontier workshop asset:



Within the Steam client, you can change the title and description of your mod, but that's all. You'll need to use our Farthest Frontier Mod Uploader app to do the rest, such as specifying which assets to upload.

Back in our app, you can choose what type of mod you're uploading (Building, or Saved Game - if you're following this guide, you're probably uploading a Building mod!). You can also change the title and description here and it will update in Steam. Most importantly, you can point Mod File Path to the location of your .ffmod file on your local hard drive. You can point Mod Preview Image Path to a .png image that you'd like to use as a preview image for your mod. Lastly, you can toggle whether you want your mod to be visible to only you, or to the public, with the Visibility drop-down menu.



When you have everything plugged in correctly, you can Upload to Steam. Uploading your mod will update your Steam workshop asset with all the information that you just plugged in. From there, you can Subscribe to your workshop asset through Steam like you would a mod for any game, and Steam will automatically download and install it. At this point you can remove the test copy of your .ffmod file from the _TEST folder that you created.

Once everything looks good both on Steam and in game, set the visibility of your mod to public, and now your mod is ready to share with the world!

We hope this content creation guide was helpful, and we look forward to seeing your creations in Farthest Frontier!