Installing

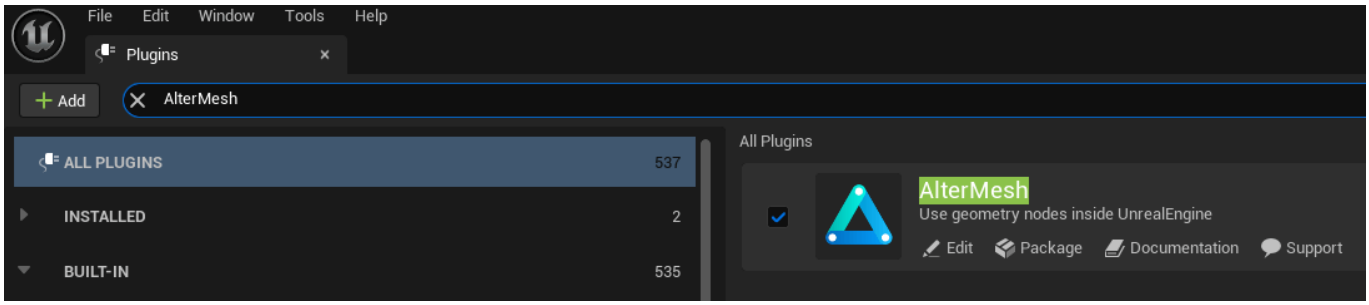
November 3, 2023

Download AlterMesh

If you haven't done so already, download AlterMesh from the Unreal Marketplace.

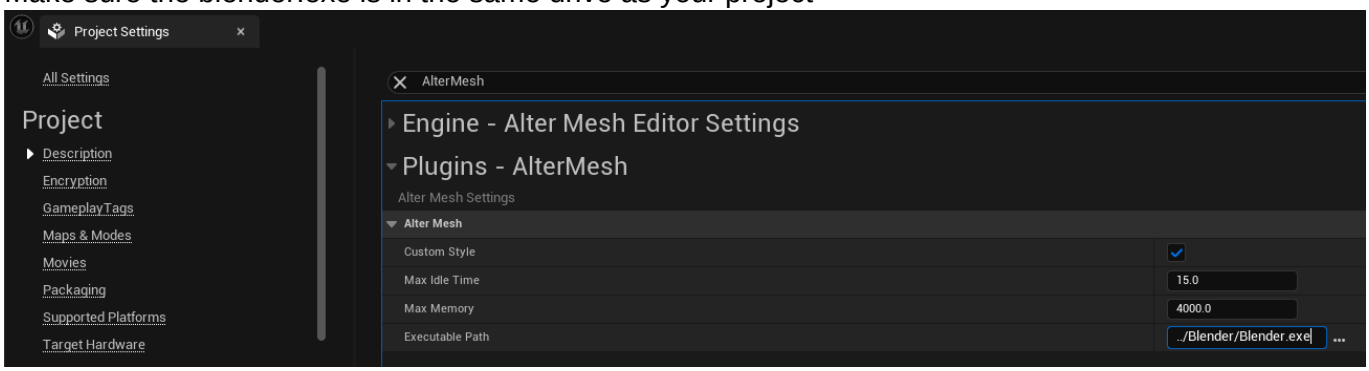
Activation

After downloading, locate the plugin in the plugin tabs within Unreal Engine. Activate it to enable its functionalities.



Configure Blender Executable

Open the project settings and navigate to the Plugins section. Locate AlterMesh and select the appropriate Blender executable file to start the integration.
Make sure the blender.exe is in the same drive as your project



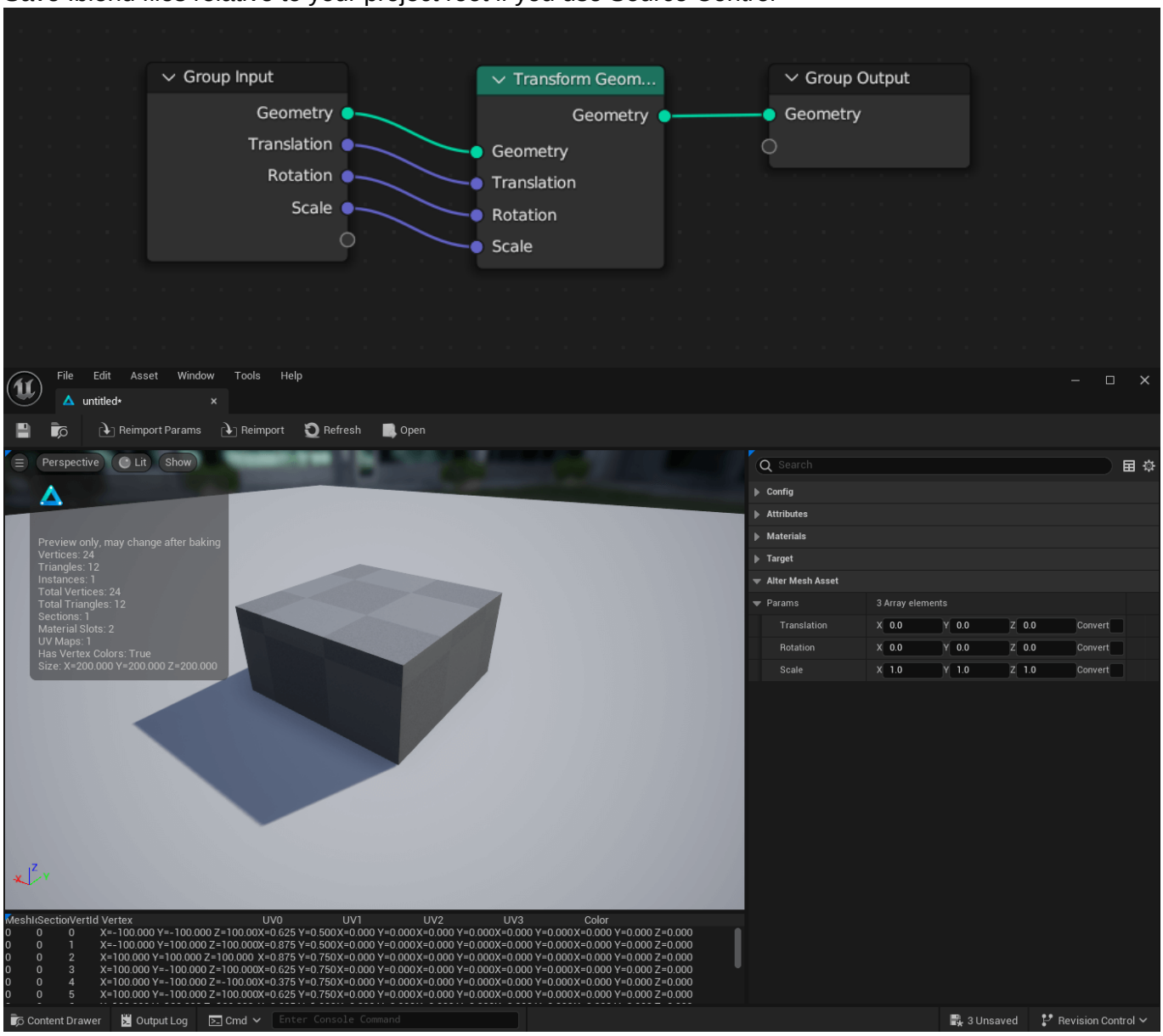
Getting Started

With AlterMesh successfully installed and configured, you can now start using its features. Explore bundled examples by clicking the content button on the toolbar or read below how to create your first tool

Your First tool

1. Open Blender
2. Add a geometry nodes to the default cube
3. Add a transform node
4. Connect the inputs to the Group Input
5. Save and Close
6. Drag the saved file to the UnrealEngine content browser
7. Drag the asset into the viewport

Save .blend files relative to your project root if you use Source Control



Asset Editor

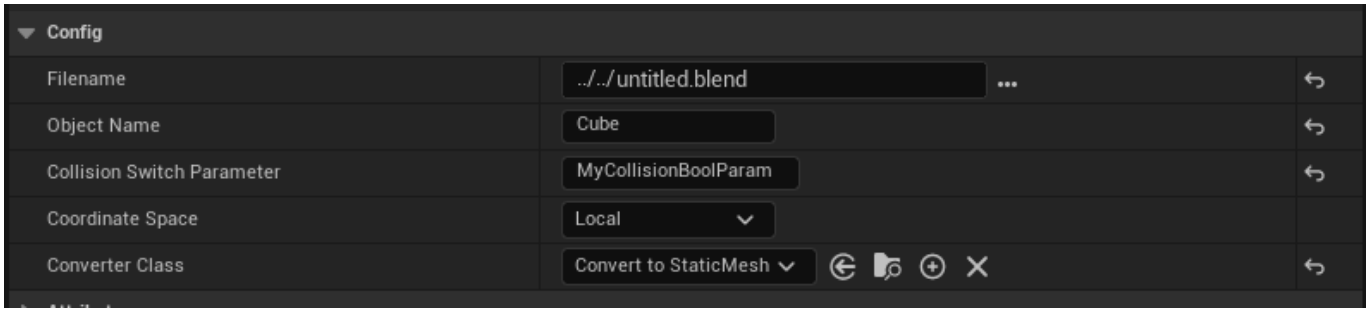
November 3, 2023

File Configuration

Filename and Object name will be filled automatically upon importing a .blend file, and can be left as is.

The collision parameter lets you choose a parameter to toggle during the collision creation, allowing to import custom collision created by your GeometryNodes.

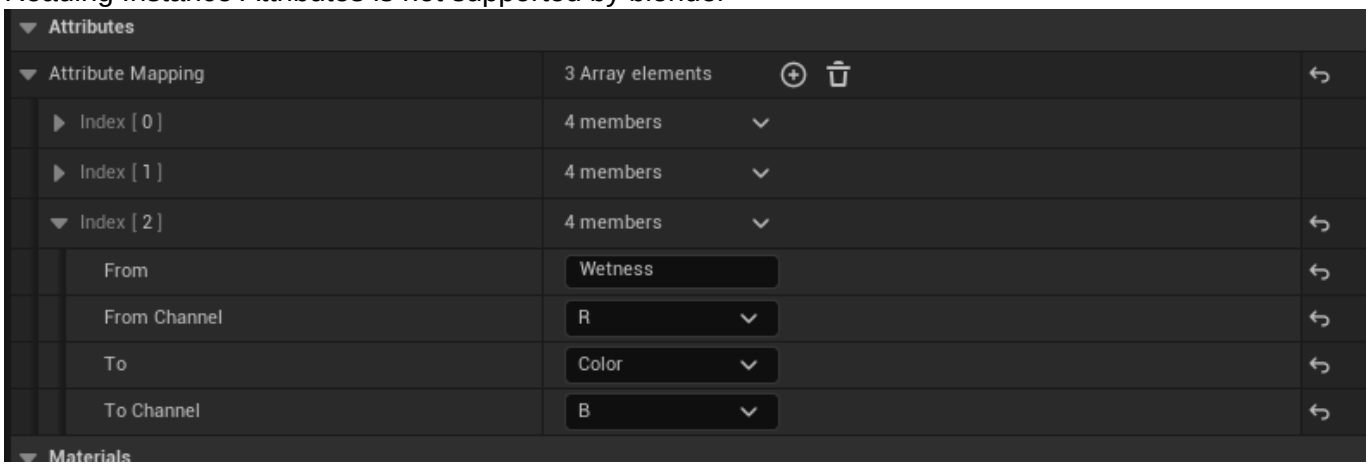
By default, assets will be converted to Static Meshes, you can change this behavior in the Converter Class setting.



Attributes

Here you can map attributes from Blender's output to Unreal attributes. By default "UVMap" output will be mapped to Mesh UVs and "Colors" output will be mapped to Mesh Vertex Colors.

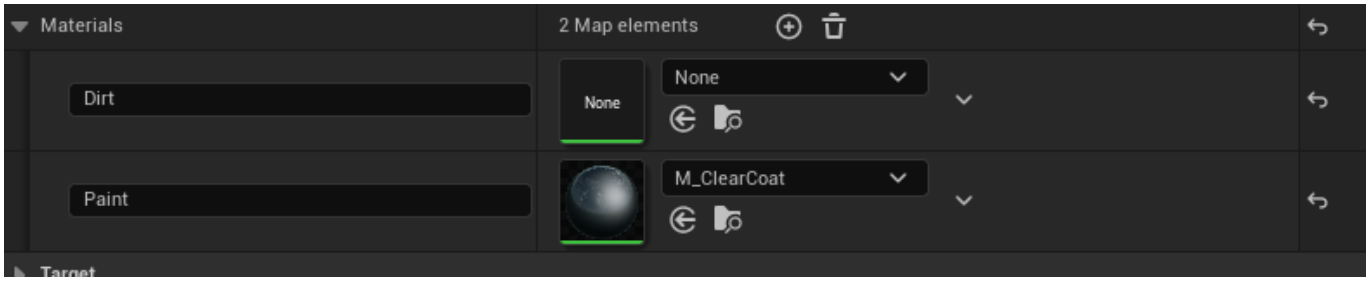
However you can map any float, vector or color from the Output Group of your Geometry Nodes into a mesh attribute.
Reading Instance Attributes is not supported by blender



Materials

During the import process, AlterMesh will look in your blend files and list all of the materials available in your .blend, this will allow you to map used or unused materials with an Unreal equivalent.

Unfortunately AlterMesh will not import your materials automatically, this would require a translation of Blender-Unreal materials which is not simple. Please remake the material in Unreal or assign a different one.

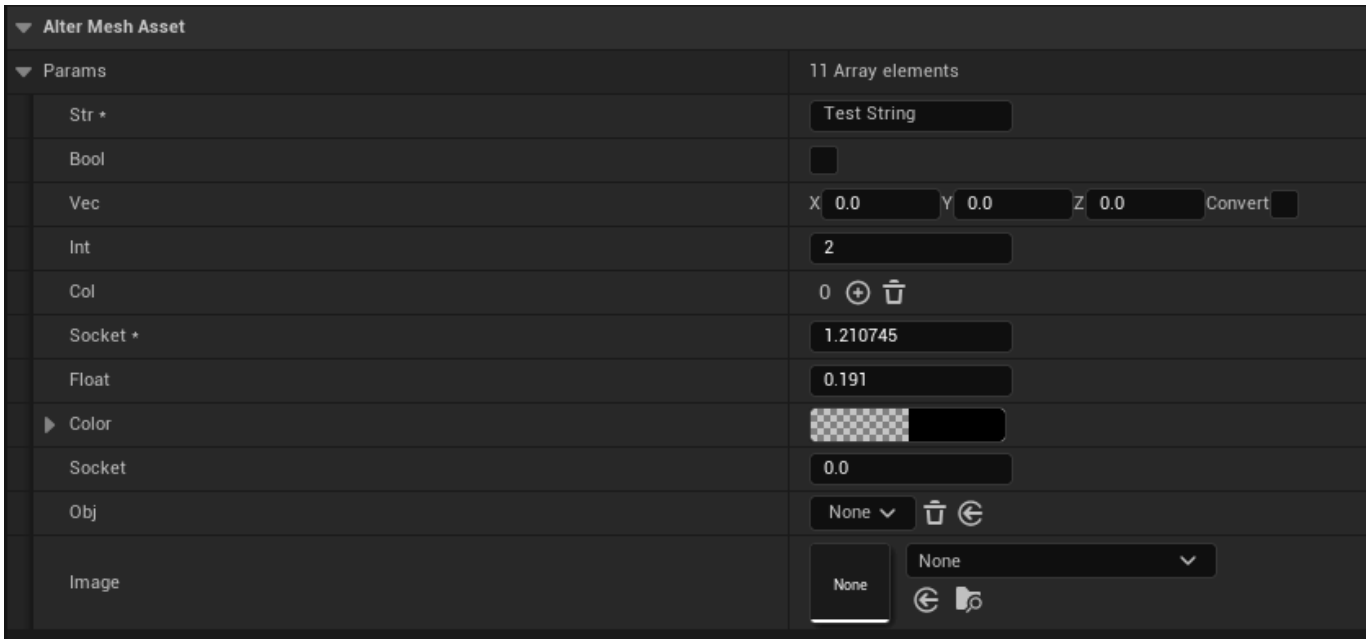


Parameters

Only parameters that are exposed to the **Group Input** are available, if you add more parameters to the group input, you can click on the Reimport Parameters button on the toolbar to refresh the parameter list.

Adding more parameters will not automatically update actors placed on the map, this prevents from you accidentally changing models that were final, you can manually refresh a placed actor by changing a parameter or clicking the refresh button on the details panel.

Node panels, parameter tooltips, Min/Max values are supported

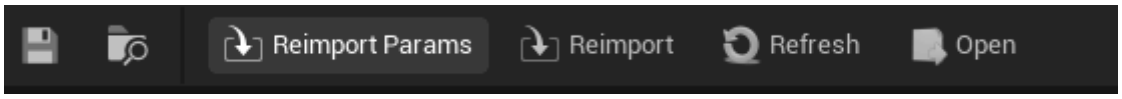


Exposing Parameters

November 3, 2023

Input Group

Only parameters that are connected to the Group Input are available. If you add more parameters to the group input, you can click on the Reimport Parameters button on the toolbar of the asset editor to refresh the parameter list.



Adding more parameters will not automatically update actors placed on the map, this prevents from you accidentally changing models that were final, you can manually refresh a placed actor by changing a paremeter or clicking the refresh button on the details panel.

Param settings

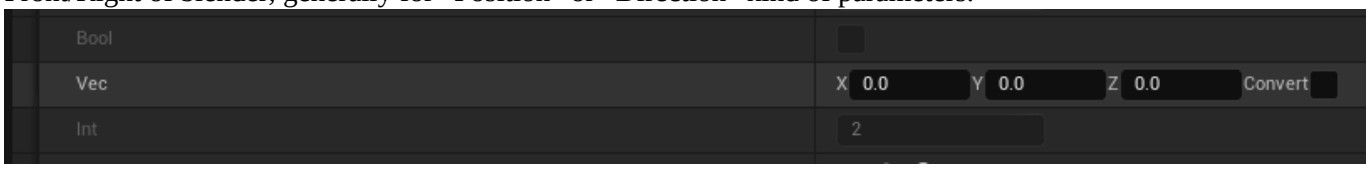
Node panels, parameter tooltips, Min/Max values are supported.

You must assign unique names to Node panels, otherwise they will be combined into a single panel.

Vector Param

Vector Params have a special toggle called Convert Space, this toggle will mirror and swap the X and Y axis, converting from Left Handed coordinate system to Right Handed coordinate system.

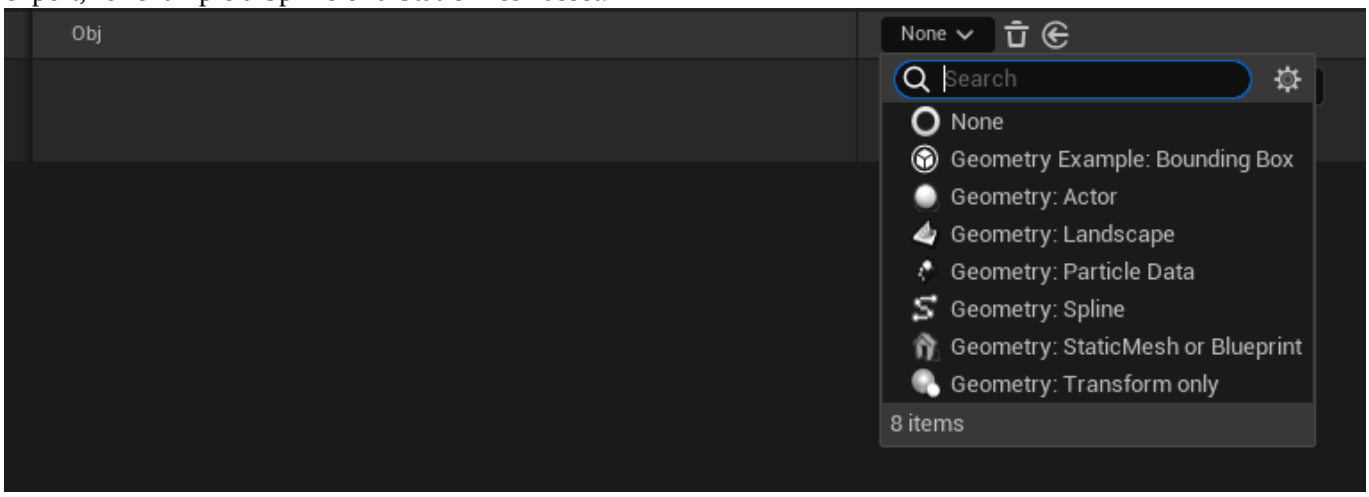
This is disabled by default, and can be turned on when you need that the Front/Right of unreal is the same as the Front/Right of blender, generally for “Position” or “Direction” kind of parameters.



Geometry Params

Geometry params are a special kind of socket, they allow for multiple stuff to be selected in blender, such as geometry, curves, points, etc.

When you expose a pin of this type, in Unreal you will be able to select which kind of geometry you want to export, for example a Spline or a Static Mesh asset.

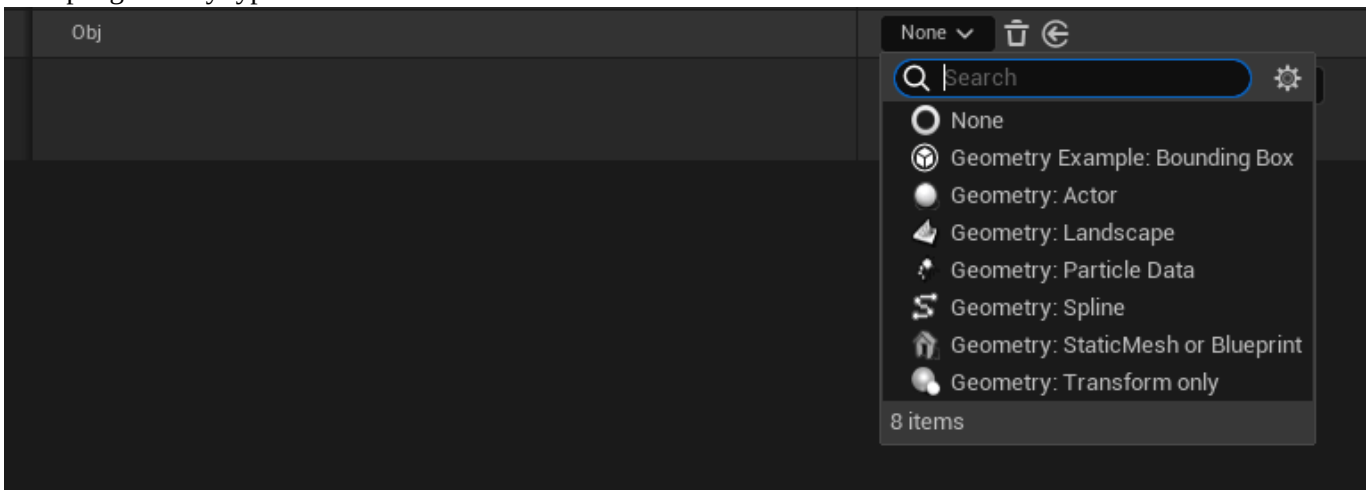


Geometry Parameters

November 3, 2023

Overview

When you expose an Object Info (orange pin) to the group input, you will receive the option to choose between multiple geometry types within unreal.

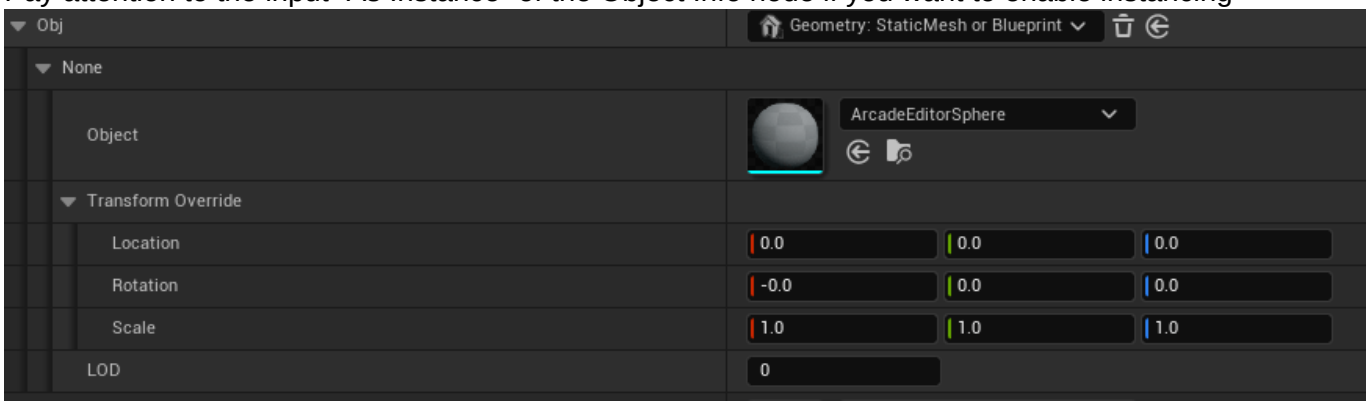


Static Mesh asset or Blueprint

This geometry type will allow you to select an asset, all the meshes inside this asset will be exported to blender. If you keep the object as instances (not realized) within your node group, they will be replaced by instances of the

selected asset upon map save.

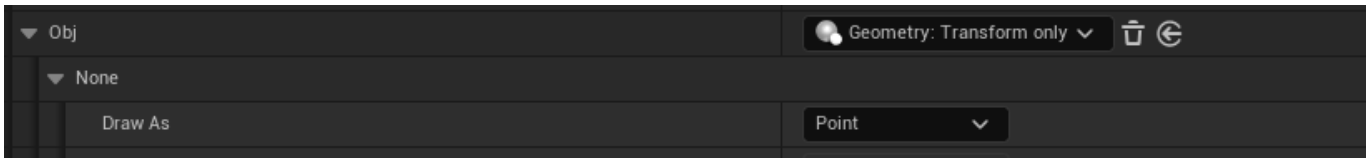
Transform override: allows you to make modifications to the transform of the exported mesh, such as rotating or scaling.
LOD: Which lod of the static mesh to export
Pay attention to the input “As Instance” of the Object Info node if you want to enable instancing



Transform Only

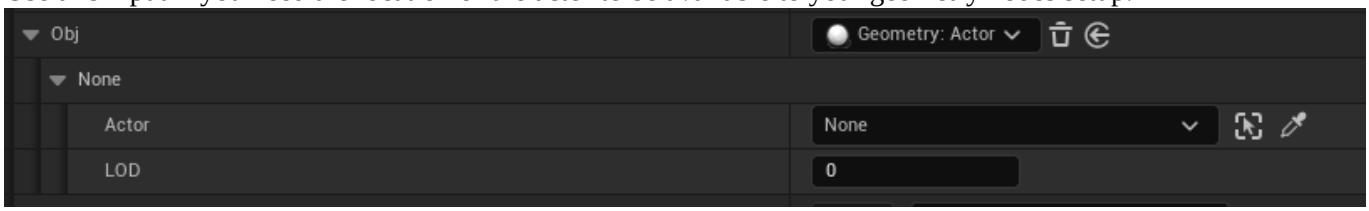
Equivalent to the “Empty” object in blender, it will only export transform information to your node setup, generally used as a pivot or as a target.

Draw As: how you want to visualize the transform, such as a cube or a single point.



Actor

This will export the selected actor to blender, including its mesh and transform information.
Use this input if you need the location of the actor to be available to your geometry nodes setup.



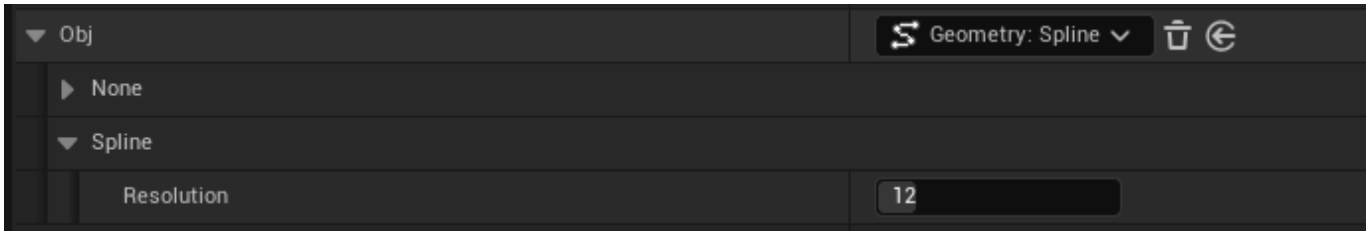
Landscape

This will export the landscape geometry to blender. Use this input if you need to place assets on top of the landscape.

Spline

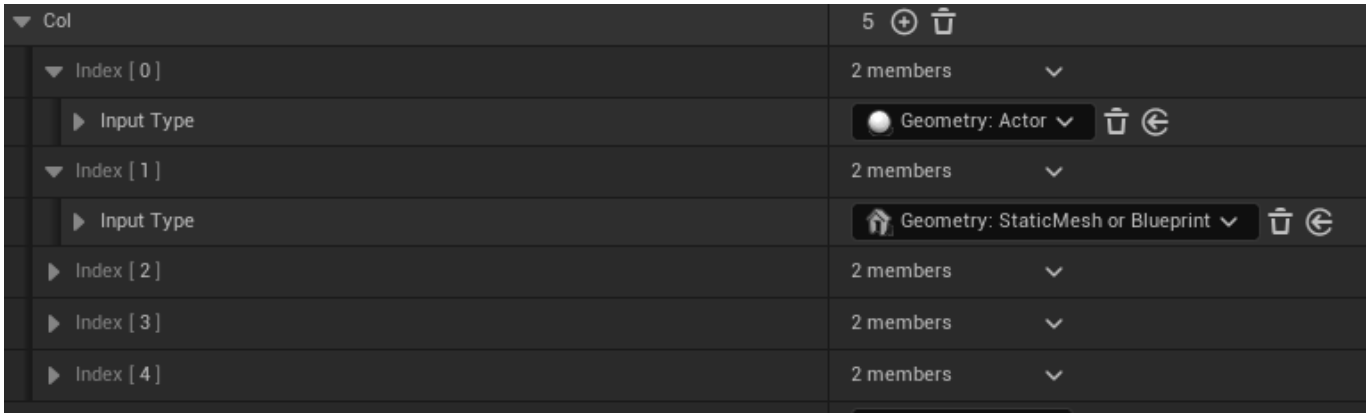
Creates a spline curve to be edited in Unreal. Exposed params with a curve in its modifier defaults will be defaulted to this geometry type.

Resolution: How many points per segment the curve will have in Blender.



Collection

When a collection pin is exposed, it will become an array of geometry types, so you can mix and match Assets, Actors, etc.



Expanding Geometry Types

Your project may create new geometry types (C++) if you need extra functionality.

This is an advanced feature

```
                    // .h

// Example Geometry type that exports the bounding box of the selected
 actor
// Must be paired with a file in ThirdParty/Extensions folder
UCLASS(meta=(DisplayName="Geometry Example: Bounding Box"))
class ALTERMESHEXAMPLESEDITOR_API UAlterMeshExamplesGeometryBoundingBo
x : public UAlterMeshGeometryBase
{
    GENERATED_BODY()
public:

    UPROPERTY(EditAnywhere, BlueprintReadWrite)
    AActor* Actor;

    virtual bool ShouldExport() override { return !!Actor; }
    virtual void Export(FAlterMeshExport& Exporter) override;
};


// .cpp
```

```
void UAlterMeshExamplesGeometryBoundingBox::Export(FAlterMeshExport& E
xporter)
{
    Super::Export(Exporter);

    FVector Origin, Extent;
    Actor->GetActorBounds(false, Origin, Extent);
    FBox Bounds = FBox::BuildAABB(FVector::ZeroVector, Extent);

    // Create a cube from bounds
    TArray Vertices;
    TArray Indices;
    UAlterMeshExamplesLibrary::MakeCube(Bounds, Vertices, Indices);

    // Convert scene
    for (FVector& Vertex : Vertices)
    {
        Vertex = Exporter.ToBlenderMatrix.TransformPosition(Vertex);
    }

    Exporter.WriteArray(Vertices);
    Exporter.WriteArray(Indices);
}
```

The exporting code must be paired with a python file with the same name in the AlterMesh/Source/ThirdParty/ folder that will be used to import your geometry in Blender:

```
                    import bpy
from library import Reader
import numpy as np

def import_obj(object_name):
    locations = Reader(np.float32).as_list(3)
    indices = Reader(np.int32).as_array(3)

    #create a mesh
    mesh = bpy.data.meshes.new("NewMesh")
    obj = bpy.data.objects.new(mesh.name, mesh)

    mesh.from_pydata(locations, [], indices)

    return obj

def used_for_object(obj):
    False
```

```
def get_defaults(obj):
    pass
```

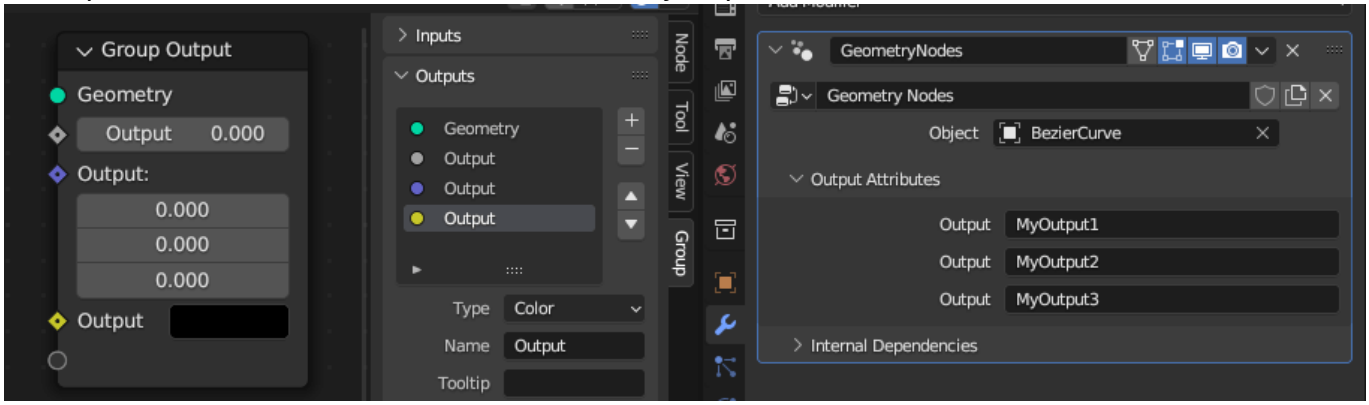
Attribute Mapping

November 3, 2023

Supported Output Types

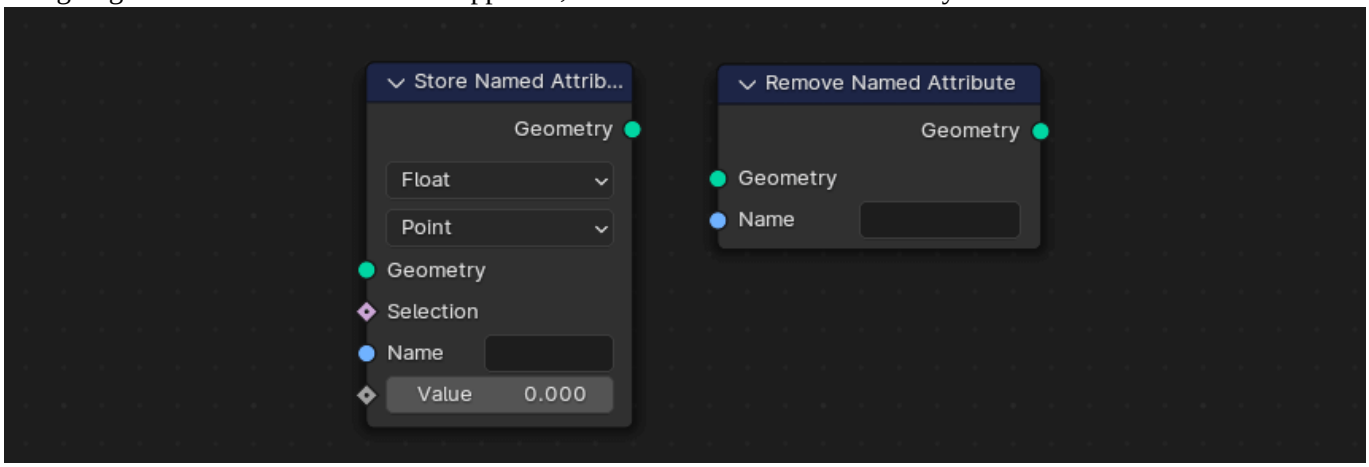
When exposing pins to the group output, you can send them over to unreal. Only a few types are supported: Floats, Colors, and Vectors.

By default AlterMesh will import Colors and UVMap0
The name of the socket does not matter, the attribute name does.
In the picture below, the float attribute is named “MyOutput1”



Attribute by name

Assigning to named attributes is also supported, the nodes below works correctly.



Mapping

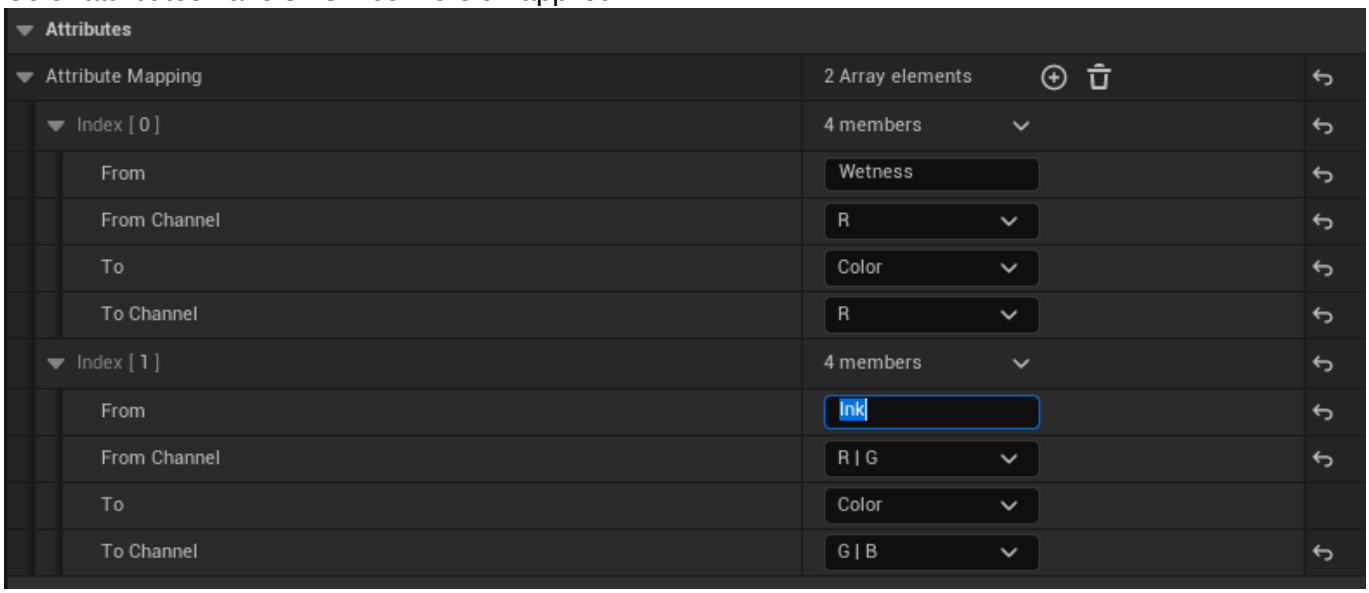
Once your attributes are being exported from your Geometry Nodes setup, you can import them by setting up the attributes panel in the asset editor

From Attribute: The attribute name inside blender. (Eg. Wetness)
From Channel: Which channel you want to map. (Eg. R)

To Attribute: Where to save the attribute value.

To Channel: Which channel to save the attribute to.

In the example below, three channels of the Mesh Vertex Color will be filled with values
Float channel R, vector RGB, and color RGBA is equivalent to XYZW
Color attributes have sRGB conversion applied



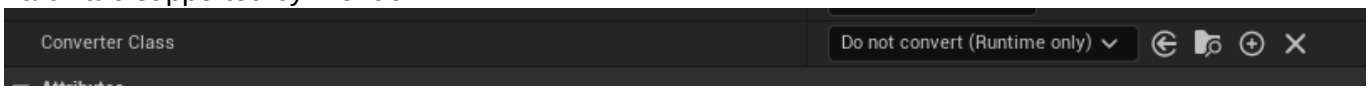
Runtime Cinematics

November 3, 2023

Runtime

AlterMesh now supports importing your object while in play mode, if you do not want the object to become a Static Mesh during map save, open the asset editor and select the “Do not convert (Runtime only)” converter class. This will prevent the object from ever getting saved down to a mesh.

You can force the object to be updated in runtime using the Refresh blueprint node, or using the sequencer.
Blender calculates the Node Tree on the background. This means that your project will only work on hardware supported by Blender.



If you just want animations, try the VAT converter instead

When should I use runtime?

- Cinematics
- Virtual Presentations
- Music visualization
- Experimental projects
- Visual Effects

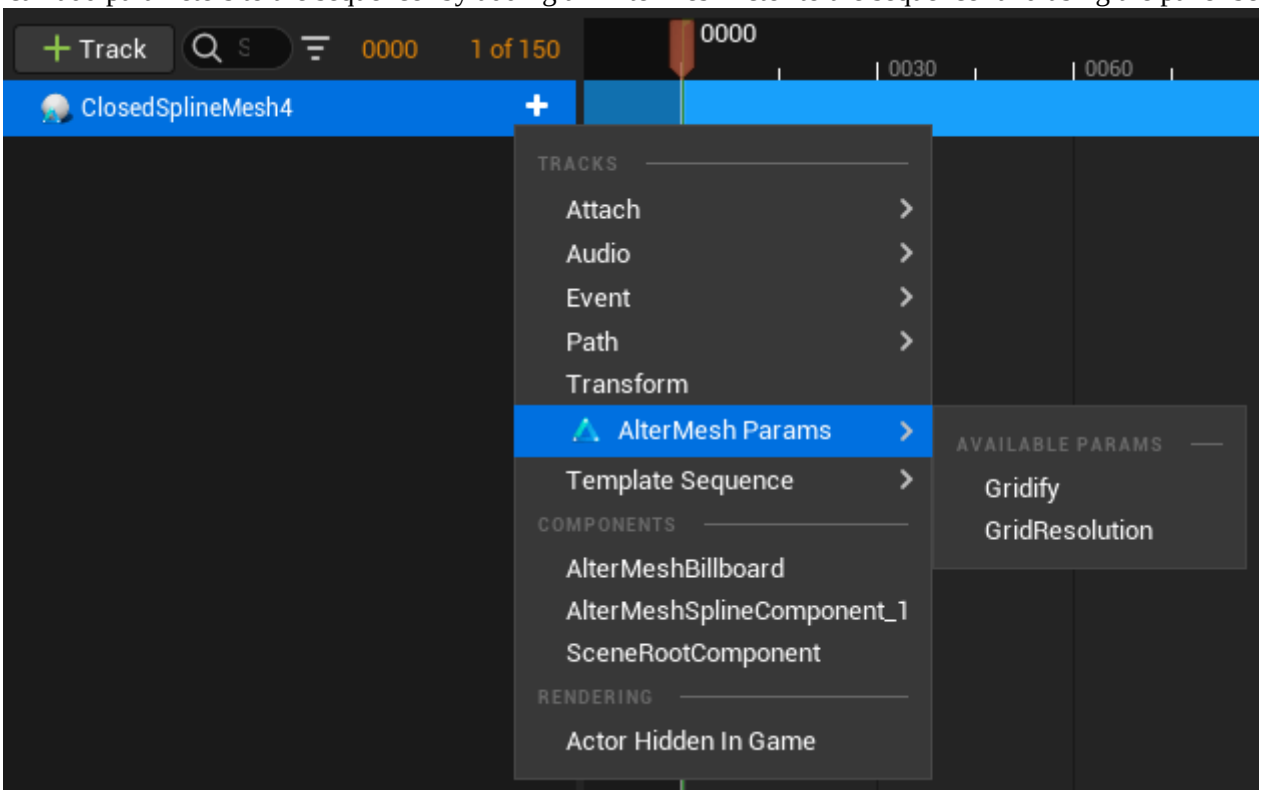
When should I NOT use runtime?

- Games that will be ported to consoles
- When VAT/ALEMBIC are better alternatives
- When performance is a concern

Runtime will not work Consoles or Mobile

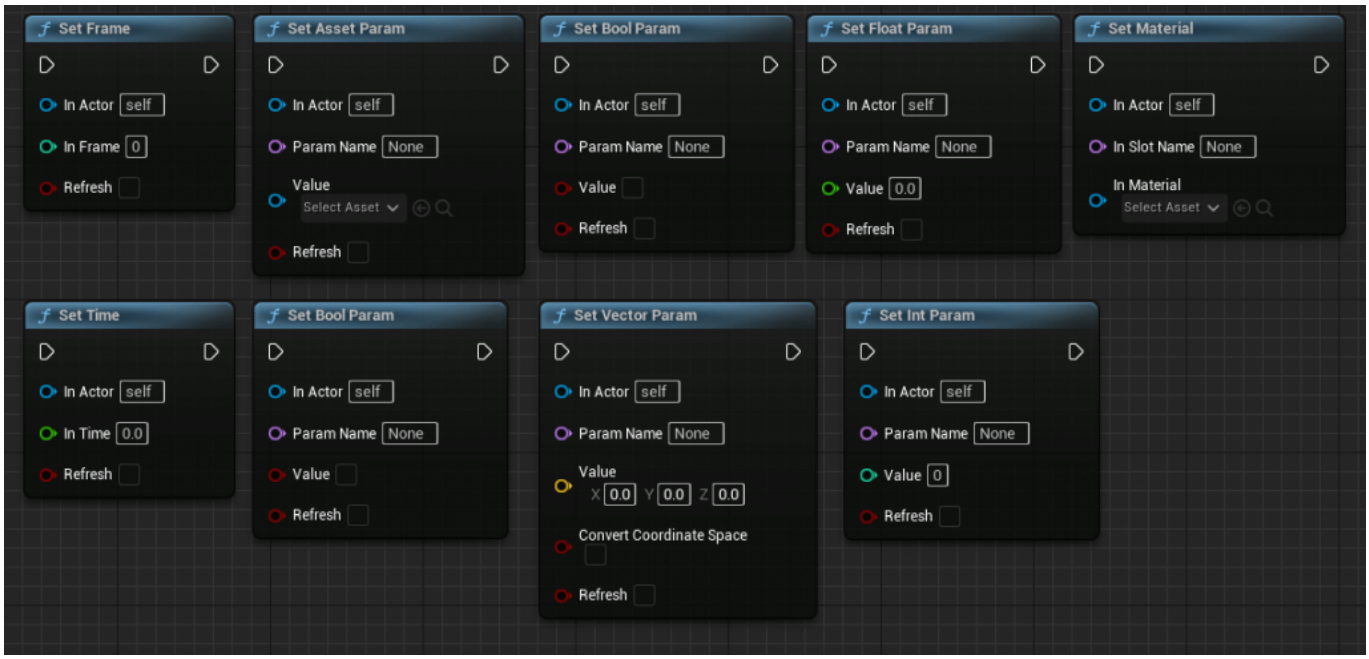
Keyframing

You can add parameters to the sequencer by adding an AlterMeshActor to the sequencer and using the panel below



Blueprints

You can set parameters programmatically from external inputs, such as Music events, Keyboard controls, etc.



Performance

The simulation is done by Blender and can only be as fast as the Geometry Node Tree.

Use the timings in Blender to profile how long each frame will take. AlterMesh will add from 10% to 50% on top of the node Tree, heavily depending on the number of triangles that have to be exported.

Troubleshooting

November 6, 2023

Nothing shows up

Here are some reasons the plugin may not be working for you, from most likely to least likely:

- Make sure you set up your executable path in Project settings -> AlterMesh -> Executable path
- Make sure the .blend files, Unreal, and the Blender executable are in the same folder, altermesh uses relative paths for source control purposes, and it may not find your files if its not in the same drive
- Check if the .blend file you are trying to import and the executable path are the same blender version.
- Check the output log for any Unreal errors
- Check the output log after using the command *altermesh.debugprocessoutput 1*
- Try us*ing altermesh.debuginteractive 1* to enable launching a visible (but not interactable) blender instance and see if it is indeed opening

Object not being converted

In AlterMesh 2.0, there is currently a problem when your project have *One File Per Actor* enabled,
where your actor will not be converted a Mesh upon map save.
As a workaround, click the Convert button on the details panel of actors that have to be saved.
This is a problem that affects AM 2.0. A patch will be made as soon as possible

Curves cannot be edited

Remember that only parameters connected to the group input can be edited in Unreal, if you use a curve in Blender, make sure its using an Object Info node connected to the group input.

In Unreal, use the dropdown on your object info parameter and choose the type “Spline”

Objects are not instancing

For the instancing features to work, either instances of Blender objects inside Unreal, or instances of Unreal Assets, the objects need to be instanced inside Blender, you can verify using the spreadsheet feature in Blender, instanced objects will show in the “Instanced” tab

Make sure your Object Info nodes have the *As Instance* option turned on, and that you do not Realize instances unless needed.

Converters

November 6, 2023

Converter Types

The converter types will decide what is the output of your mesh, one geometry node can have one output type, you can edit the configuration of the output in the details panel of the actor.

If you want to save the configuration of a specific converter type, you can create a new blueprint inheriting from a certain type. For example, you may always want to use the merge materials option on the Static Mesh Converter, for this, you can subclass the Static Mesh Converter to create your own preset.

This is a problem that affects AM 2.0. A patch will be made as soon as possible

If you have enabled *One File Per Actor* the mesh won't be converted on map save
This is a problem that affects AM 2.0. A patch will be made as soon as possible

Convert to Static Mesh

This is the default converter, upon saving the map, your mesh will become a Static Mesh, this converter also takes care of instancing

Convert to Foliage

This converter will first convert your mesh to a Static Mesh, and then place the meshes on Unreal's foliage system, you can use the foliage editor to move, add or remove instances. You can revert the edits to the foliage instances by refreshing the actor.

Convert to Vertex Animation Textures (VAT)

This converter will create a texture containing the animation information of your geometry nodes, you can change the initial and final frame of the animation in the details panel of the actor.

Do not convert

This option will not allow the output mesh to be saved, useful if you don't need a Static Mesh to be created, and will instead rely on having the object updating at runtime, eg. for cinematics

Creating new converters

You can create new converters (C++) if you need different output types, in this example, we will create a converter that saves the bouding box of the resulting GN output
This is an advanced feature

```
                    UCLASS(BlueprintType, Blueprintable, meta=(Display
Name="Converter Example:
Convert to Bounding box"))
class UAlterMeshExamplesConverterBoundingBox : public
UAlterMeshConverterBase
{
GENERATED_BODY()
public:
UAlterMeshExamplesConverterBoundingBox();
// Properties added here will be prompted for user input
UPROPERTY(EditAnywhere)
```

```
UMaterialInterface* Material;
virtual void Convert(AAlterMeshActor* InActor) override;
};




                    void UAlterMeshExamplesConverterBoundingBox::Conve
rt(AAlterMeshActor*
InActor)
{
// If you inherited from UAlterMeshConverterStaticMesh call this to cr
eate
the StaticMesh assets
// Super::Convert(InActor);
// And this would contain your newly created meshes, or the replacemen
t
meshes/blueprints
// ConverterSteps[i].AssetToUse
FBox Bounds;
// Find the bounds of all meshes and all instances
ForEachSection(InActor, [&](FAlterMeshSection& Section)
{
FBox MeshBounds = FBox(Section.Vertices);
for (FMatrix InstanceMatrix : Section.Instances)
{
Bounds += MeshBounds.TransformBy(InstanceMatrix);
}
});
FActorSpawnParameters SpawnInfo;
SpawnInfo.OverrideLevel = InActor->GetLevel();
SpawnInfo.SpawnCollisionHandlingOverride =
ESpawnActorCollisionHandlingMethod::AlwaysSpawn;
SpawnInfo.bNoFail = true;
// Proc mesh to display bounds
AActor* NewActor = InActor->GetWorld()->SpawnActor(SpawnInfo);
UProceduralMeshComponent* ProcMesh =
NewObject(NewActor);
NewActor->SetRootComponent(ProcMesh);
NewActor->AddInstanceComponent(ProcMesh);
ProcMesh->RegisterComponent();
NewActor->SetActorTransform(InActor->GetActorTransform());
if (Material)
{
ProcMesh->SetMaterial(0, Material);
}
// Create a cube from bounds
TArray Vertices;
TArray Indices;
UAlterMeshExamplesLibrary::MakeCubeFromBox(Bounds, Vertices, Indices);
TArray Normals;
```

```
TArray UVs;
TArray Colors;
TArray Tangents;
ProcMesh->CreateMeshSection(0, Vertices, Indices, Normals, UVs, Colors
,
Tangents, false);
}
```