MODULE: SPLINES

Overview

This module allows you to use existing spline solutions to perform actions such as terrain conforming, make ramp, paint road & path, spawn/remove vegetation, etc. Whatever you can imagine with a spline.

Currently the spline tools being supported are:

- Unity's Splines package (com.unity.splines), free, requires Unity 2022.1+, installed via the Package Manager.
- Polaris Spline Tool: included in Polaris 2021. This tool itself only works for Polaris terrain, but with this module it also works with Unity terrain or any terrain system.
- Curvy Spline 8 from ToolBuddy.
- Dreamteck Spline from Dreamteck.
- Bezier Solution from Yasirkula. In this document we will refer to it as "Yasirkula Spline" to be more specific than the common name "bezier spline".
- Your own spline tool by implementing the ISplineEvaluator interface, or deriving from the SplineEvaluatorBase class. Please contact via Discord if you need help on this.

Note: This module doesn't provide a spline placement tool, you will need at least one of the tools above.

Version: 2023.1.0

Requirements: Vista 2023.1.0+

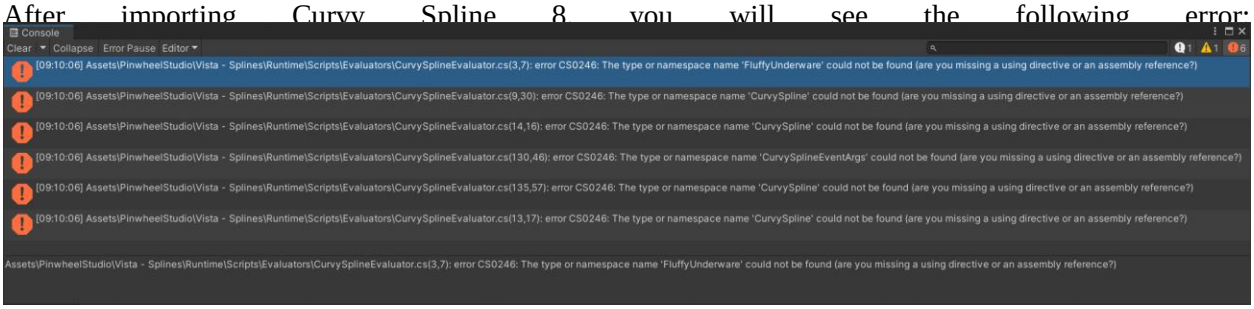
Installing a spline placement tool and creating a spline

As usual, you install a spline placement tool using the Package Manager. Each spline tool has a different way to add a spline point, adjusting tangents, etc. Please refer to the specific tool documentation for detail.

Setting up for Curvy Spline 8

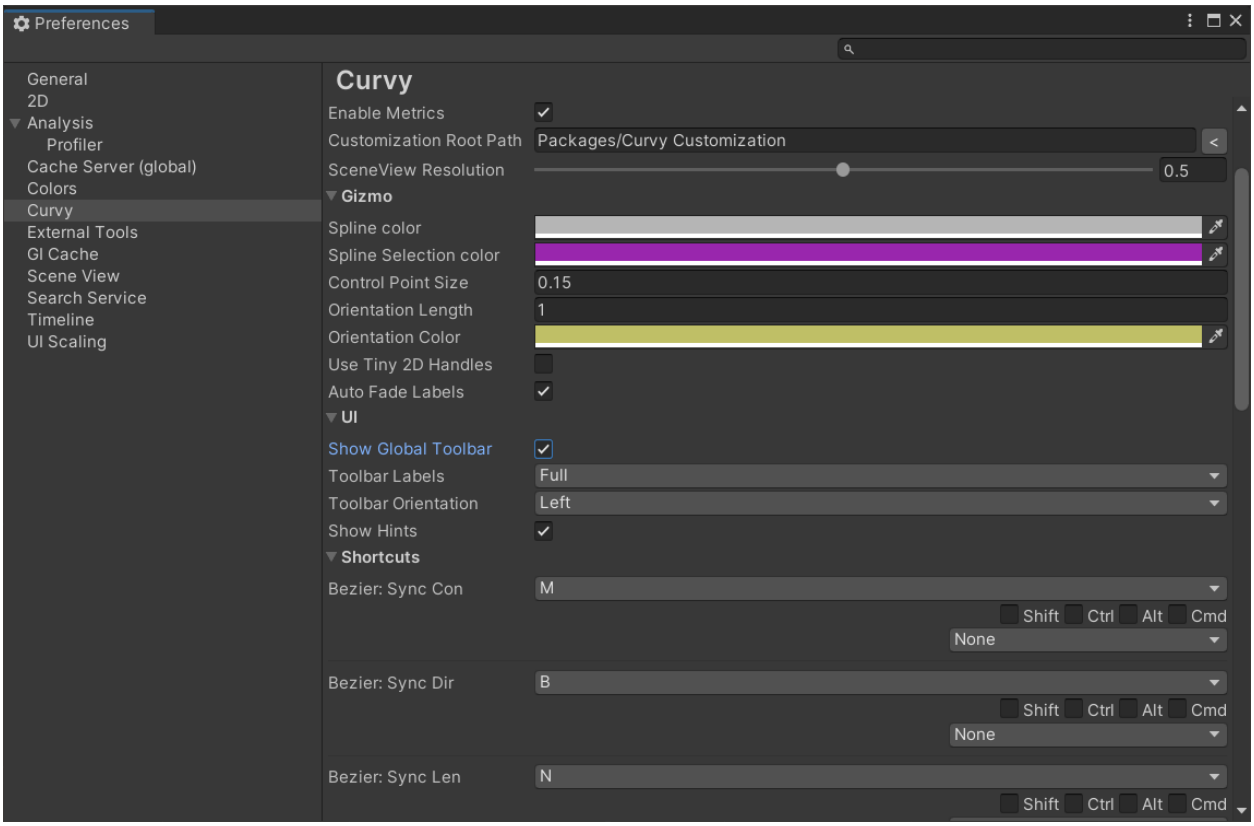
Please read this section if you're using the Curvy Spline 8 asset.

After importing Curvy Spline 8 you will see the following error:

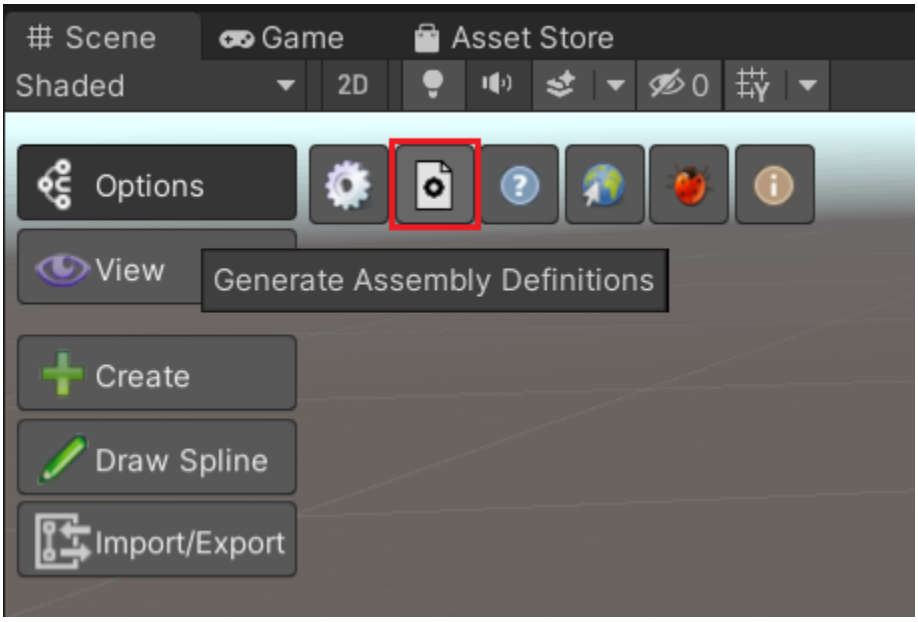


To resolve them, please do the following steps:

1. Go to **Edit>Preferences…>Curvy** and enable **Show Global Toolbar**



2. In the Scene View, select **Option>Generate Assembly Definitions**

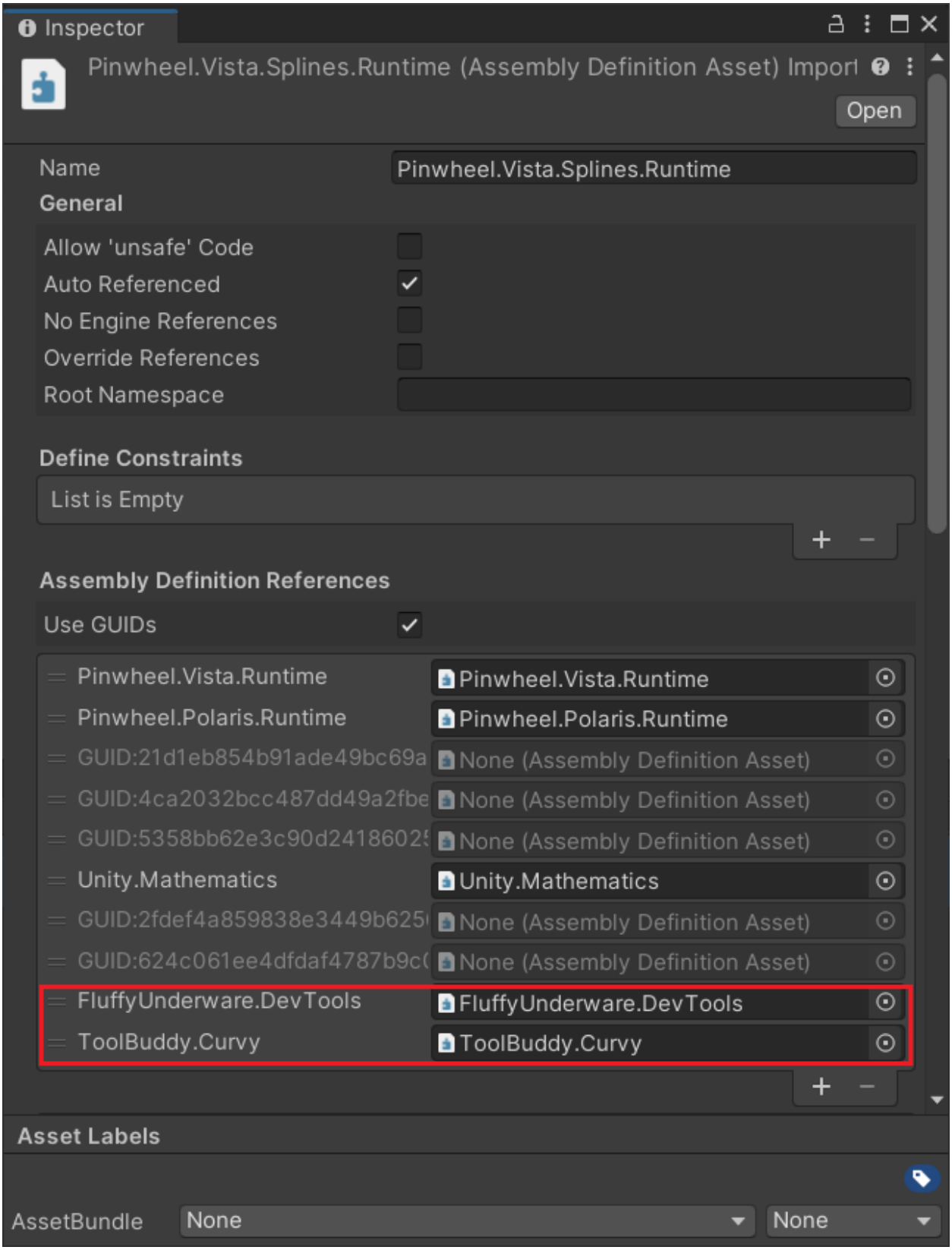


3. Let it compile.
4. Select the assembly definition at **Assets/PinwheelStudio/Vista - Splines/Runtime/Scripts/Pinwheel.Vista.Splines.Runtime.asmdef**

   Under the Assembly Definition References, add 2 assemblies:

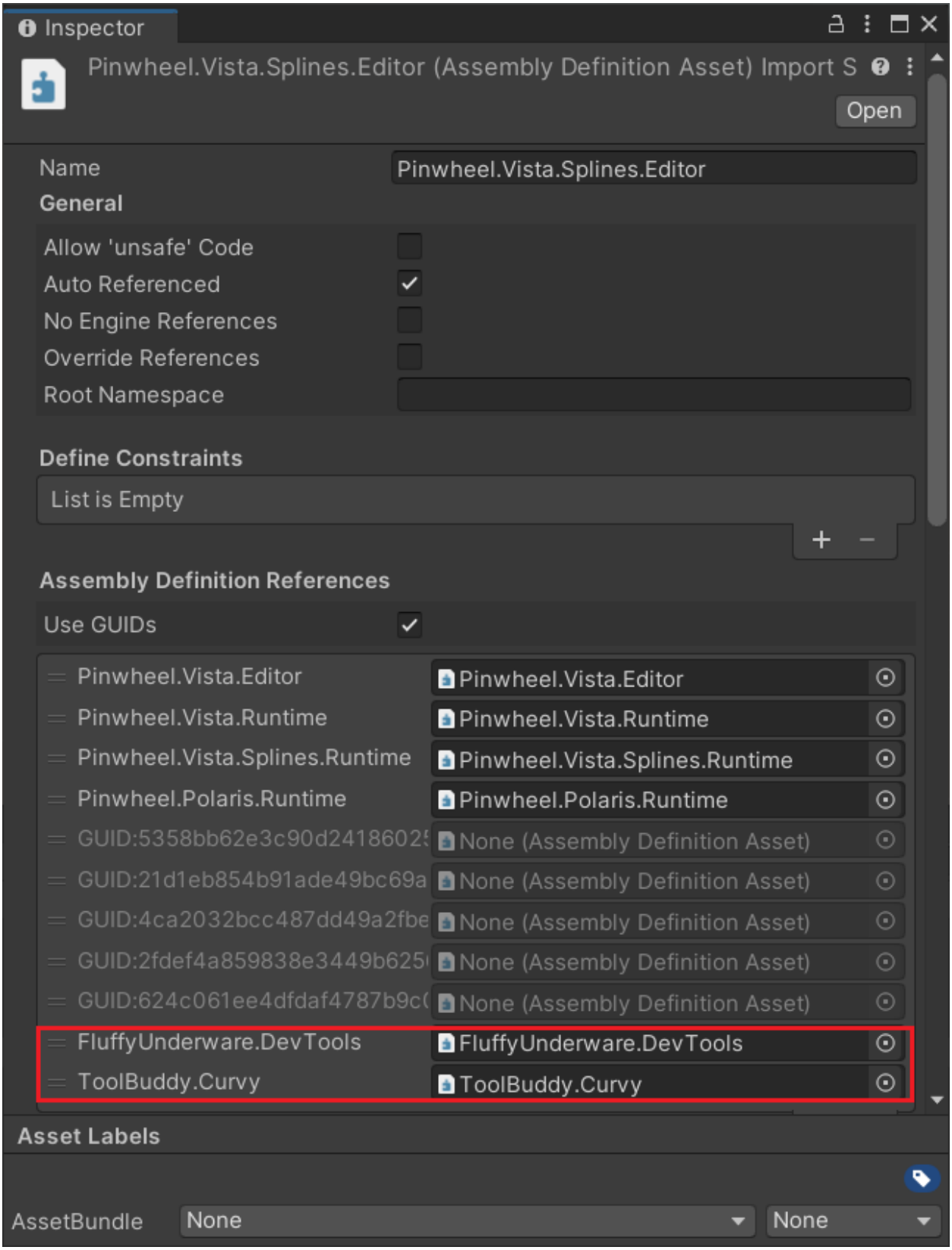
   - FluffyUnderware.DevTools
   - ToolBuddy.Curvy

Click Apply.


Inspector
Pinwheel.Vista.Splines.Runtime (Assembly Definition Asset) Import
Open
Name
Pinwheel.Vista.Splines.Runtime
General
Allow 'unsafe' Code
Auto Referenced
No Engine References
Override References
Root Namespace
Define Constraints
List is Empty
Assembly Definition References
Use GUIDs
Pinwheel.Vista.Runtime
Pinwheel.Vista.Runtime
Pinwheel.Polaris.Runtime
Pinwheel.Polaris.Runtime
GUID:21d1eb854b91ade49bc69a
None (Assembly Definition Asset)
GUID:4ca2032bcc487dd49a2fbe
None (Assembly Definition Asset)
GUID:5358bb62e3c90d2418602
None (Assembly Definition Asset)
Unity.Mathematics
Unity.Mathematics
GUID:2fdef4a859838e3449b625
None (Assembly Definition Asset)
GUID:624c061ee4dfdaf4787b9c
None (Assembly Definition Asset)
FluffyUnderware.DevTools
FluffyUnderware.DevTools
ToolBuddy.Curvy
ToolBuddy.Curvy
Asset Labels
AssetBundle
None
None

5. Select the assembly definition at **Assets/PinwheelStudio/Vista - Splines/Editor/Scripts/Pinwheel.Vista.Splines.Editor.asmdef**
   Under the Assembly Definition References, add 2 assemblies:
   - FluffyUnderware.DevTools
   - ToolBuddy.Curvy

Click Apply


Inspector
Pinwheel.Vista.Splines.Editor (Assembly Definition Asset) Import S
Open
Name
Pinwheel.Vista.Splines.Editor
General
Allow 'unsafe' Code
Auto Referenced
No Engine References
Override References
Root Namespace
Define Constraints
List is Empty
Assembly Definition References
Use GUIDs
Pinwheel.Vista.Editor
Pinwheel.Vista.Runtime
Pinwheel.Vista.Splines.Runtime
Pinwheel.Polaris.Runtime
None (Assembly Definition Asset)
FluffyUnderware.DevTools
ToolBuddy.Curvy
Asset Labels
AssetBundle
None

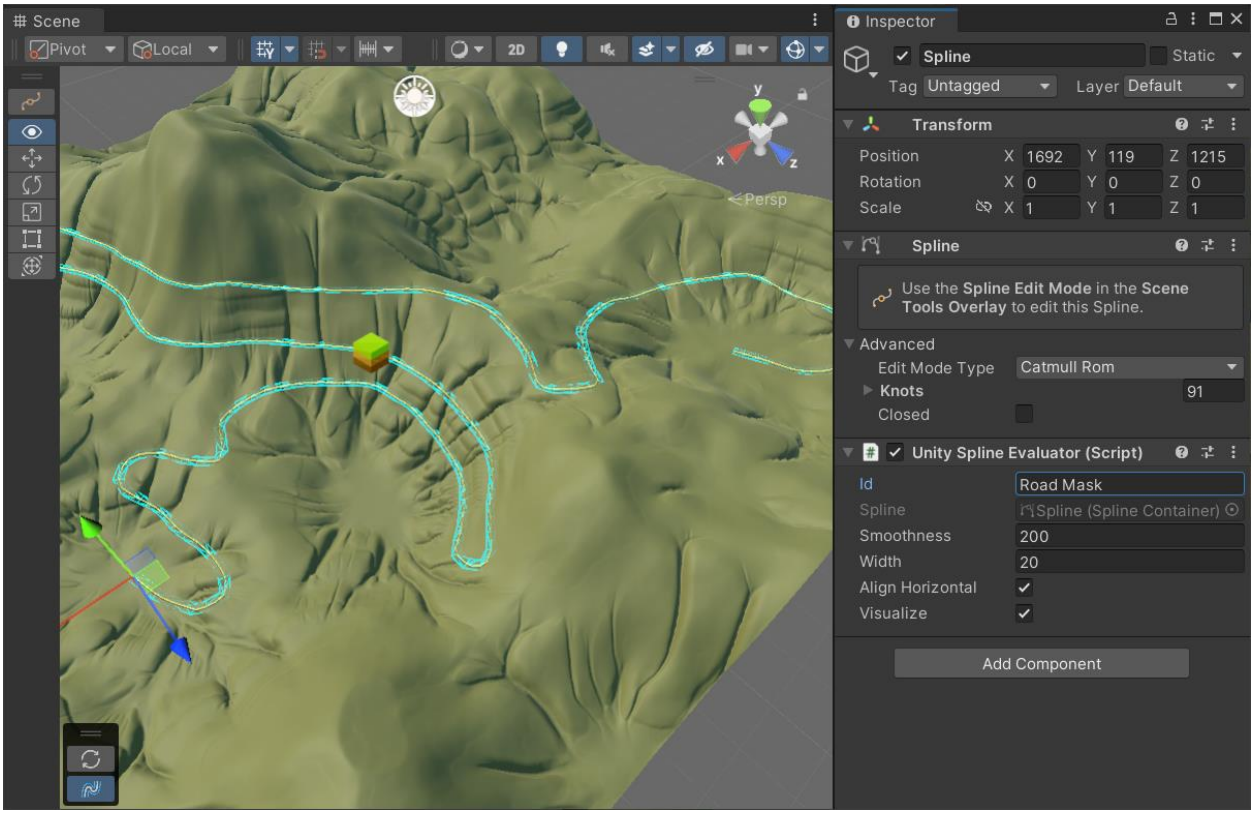
After this step, there should be no script error in the console.

Adding proper Spline Evaluators

For Vista to recognize and read data from the spline object, which comes from many third party tools out there, you need to add proper Spline Evaluator components to the object, depending on which spline tool you are using:

- Unity Splines: add a Unity Spline Evaluator component.
- Polaris Spline Tool: add a Polaris Spline Evaluator component.
- Curvy Spline 8: add a Curvy Spline Evaluator component.
- Dreamteck Spline: add a Dreamteck Spline Evaluator component.
- Yasirkula Spline: add a Yasirkula Spline Evaluator component.

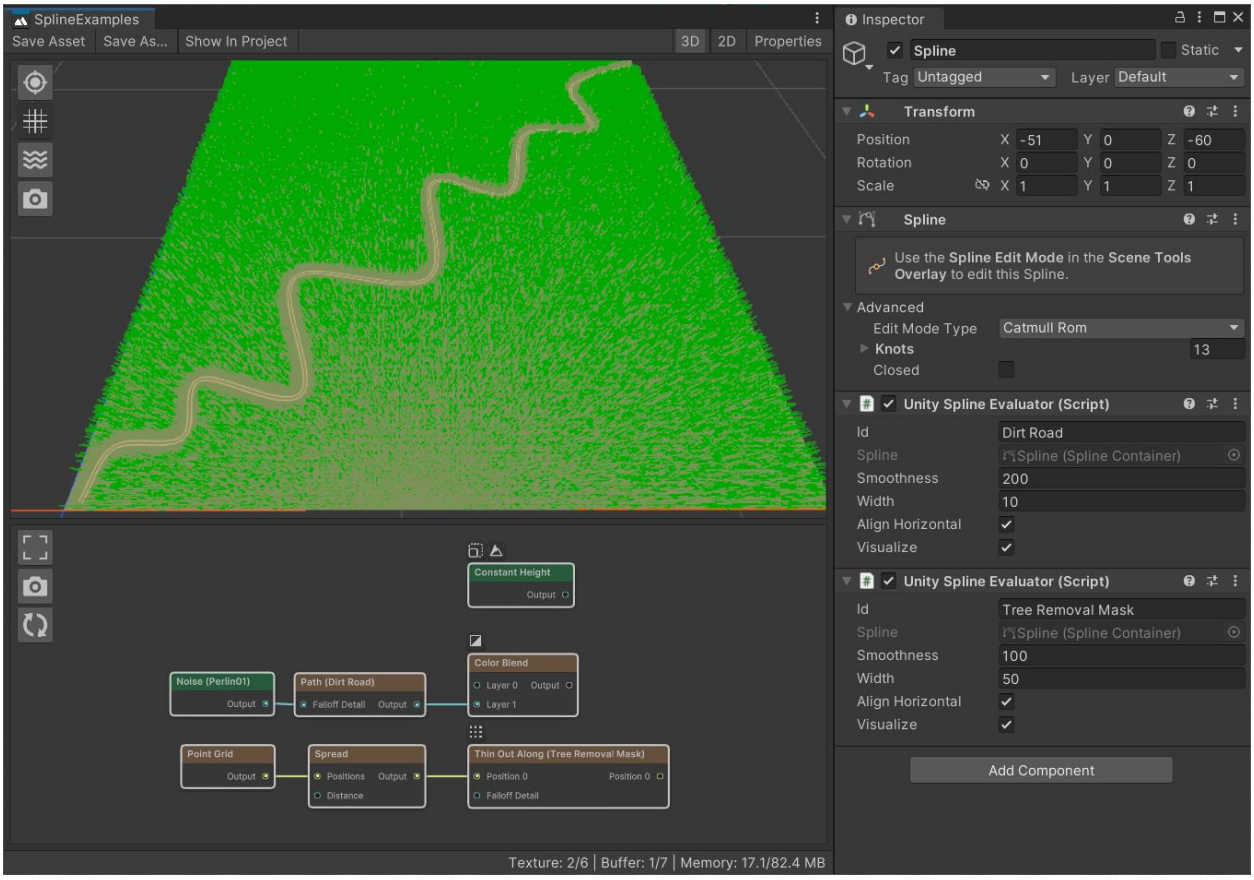
From now on, we will use Unity Splines for example.



A Spline Evaluator will has the following properties:

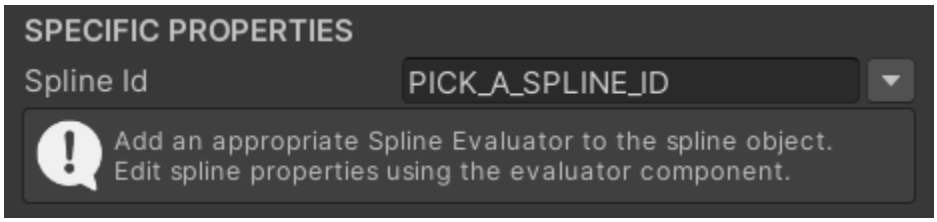
- Id: An id (should be unique) assigned to the evaluator. This id will then be used to refer to the spline inside the graph editor. It's best to use an id that describes how the spline will be used, such as "Road Mask" or "Vegetation Removal Mask", etc. **If id is empty, the spline will be ignored from all operations, including reference from the graph editor, draw gizmos, etc.**
- Spline: The component that holds the spline and provides spline evaluation methods. This will be assigned automatically in most cases.
- Smoothness: Decide the detail level of the spline, higher value will generate more vertices internally. For most spline tools, this value is "per-spline object", but for Polaris spline it is "per-segment". Because of this, the Polaris spline will use a much smaller value.
- Width: Width of the spline, aka: how far the spline will expand from any point along the normal vector at that point.
- Align Horizontal: This will try to "rotate" the spline mesh so it aligns with the horizontal plane, useful when making a ramp where you don't want a very steep angle when the spline takes a sharp turn. Note that this only affects the internal spline mesh, it doesn't modify the rotation value of spline points.
- Visualize: A global value indicates that it should render the spline mesh in the Scene View or not. On Unity 2021.2+, there is also a button in the Scene View Overlay for this.

You can add multiple Spline Evaluator components to the same spline object to perform different actions using the same spline. Below is an example of painting a dirt road and removing tree instances along its way with a greater distance.



Refer to the spline in the graph editor

All spline-related nodes will have a Spline Id field, you can manually type in or use the selector by clicking on the arrow button next to the text field.



Spline related nodes

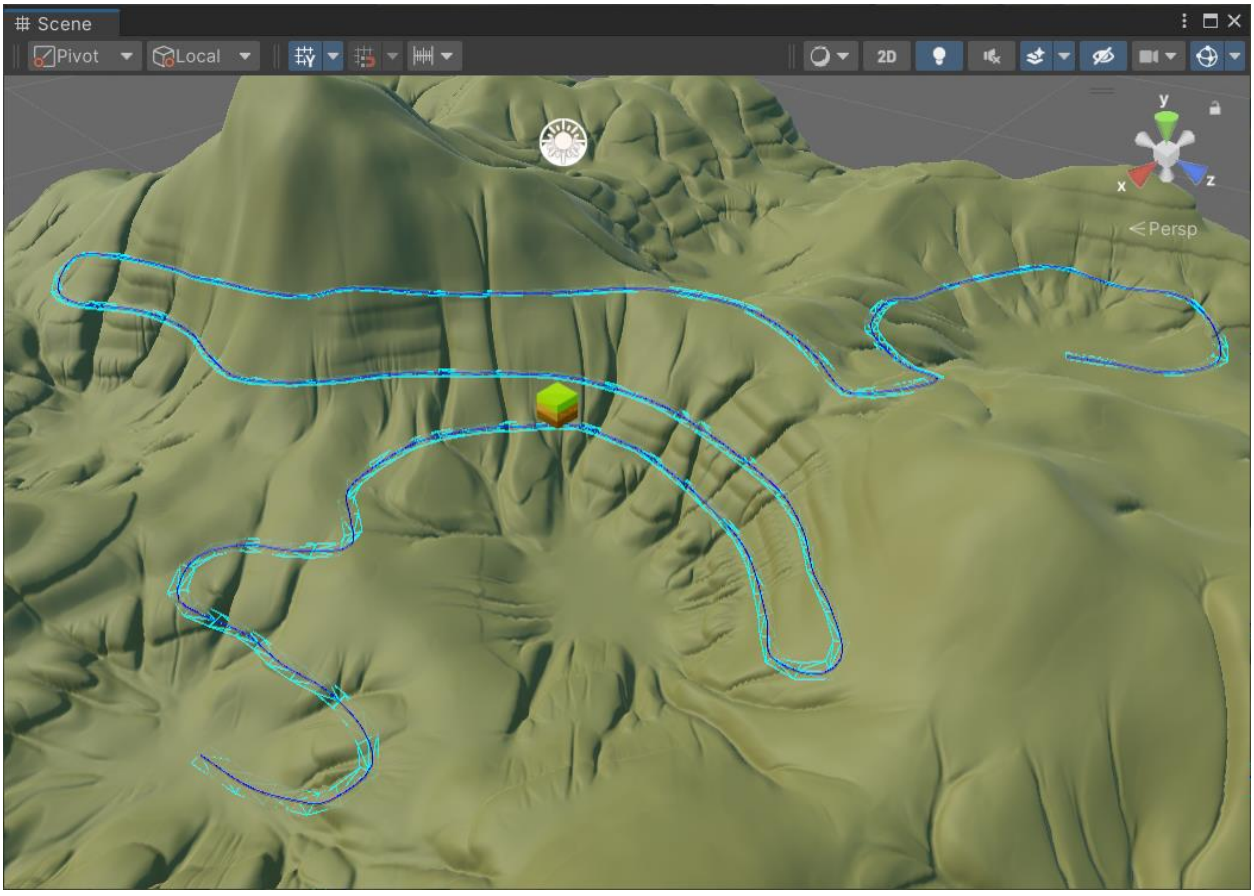
Below is the list of nodes packed in this module:

- Spline Extract: Extract the spline information such as mask, region, height map, anchors and points along the path to do something. You will only use this node in special cases as there are other nodes for common jobs.
- Ramp: Conform the terrain height to the spline mesh, useful for mountain pass, matching the terrain height to the road mesh, etc. Note that this node doesn't generate any road mesh.
- Path: Paint a path along the spline as a mask. The mask can be later used with a Weight Blend or Color Blend node for texturing.
- Thin Out Along: Remove vegetation instances along the spline, or the opposite, remove instances that are not on the spline.

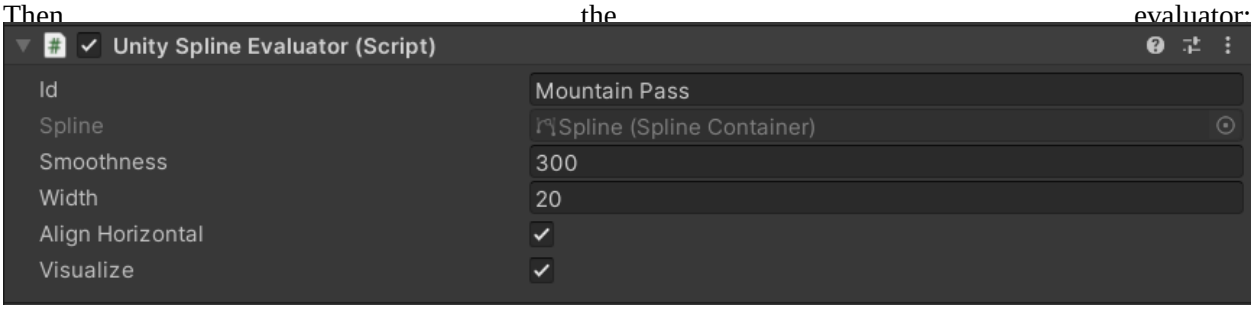
Examples

Mountain pass

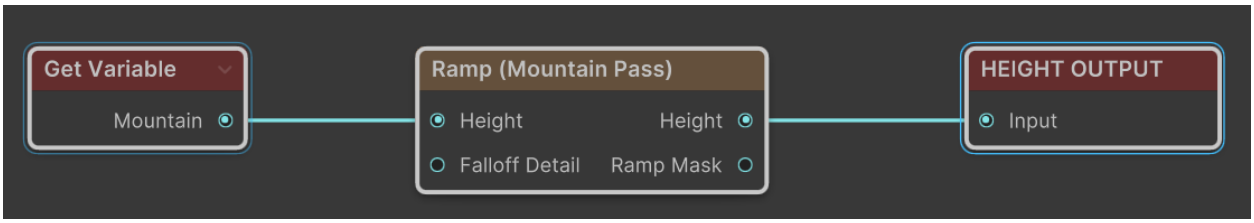
Assuming that you already have a good mountain height map. Add a spline that go around with a reasonable elevation change like this:



Then the evaluator:

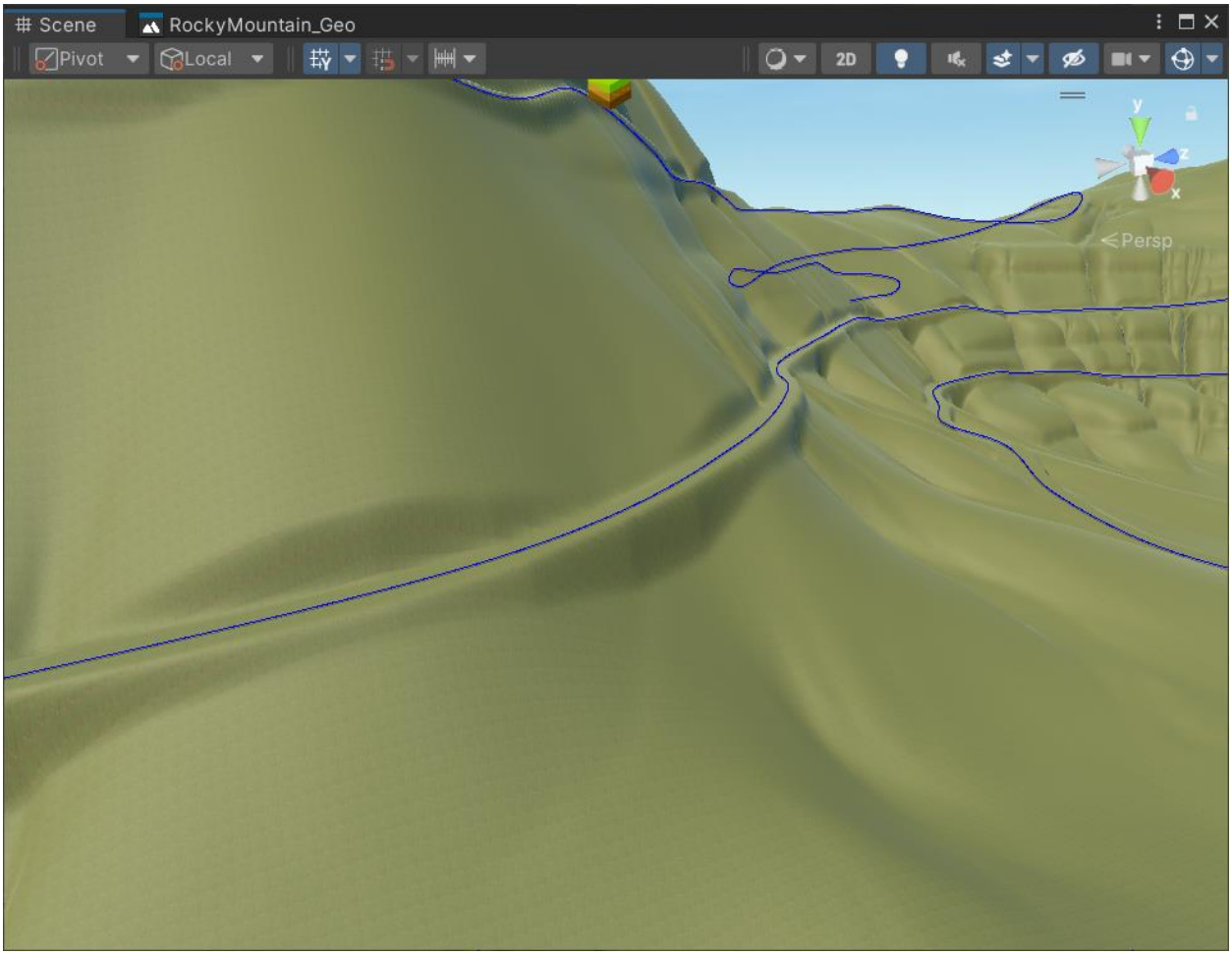


In the graph:



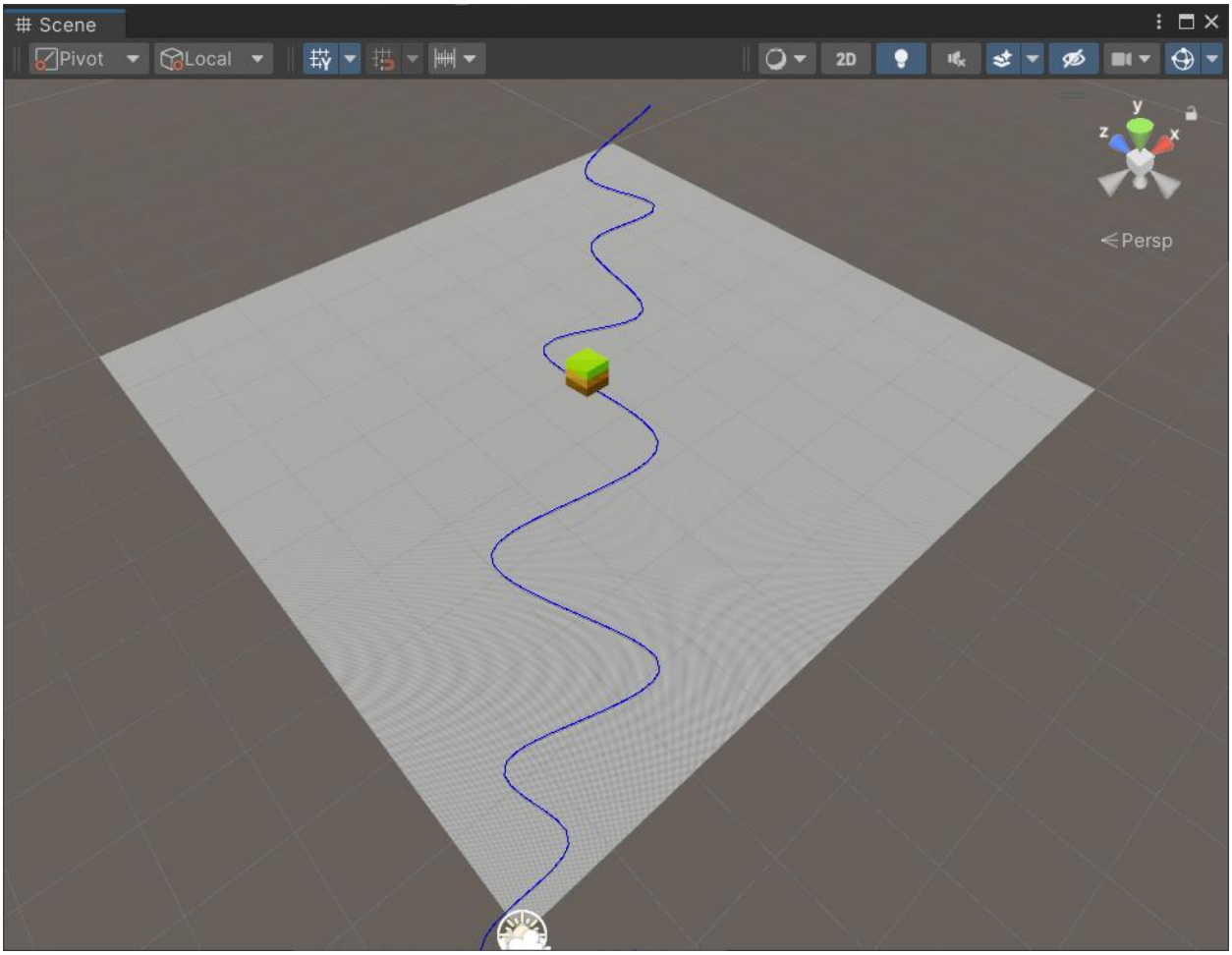
The Mountain variable is something you generate before this step, then register with the Set Variable node.

Then the result:

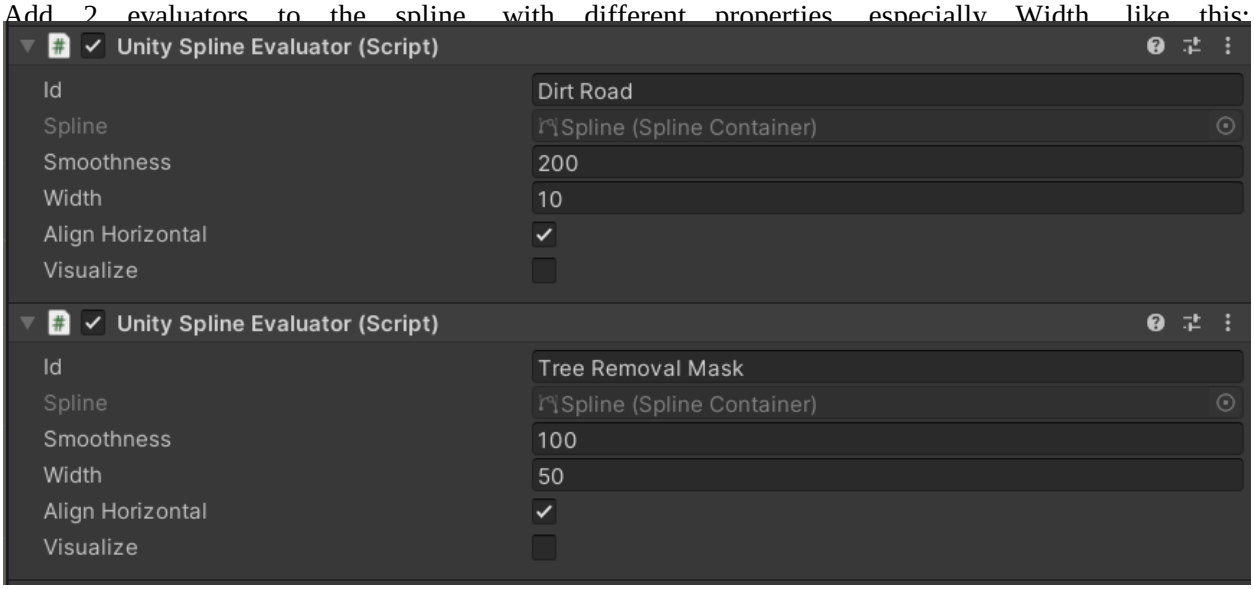

Scene
RockyMountain_Geo
Pivot
Local
2D
Persp

Paint a dirt road and remove trees along its path

Make a simple terrain and a spline like this:



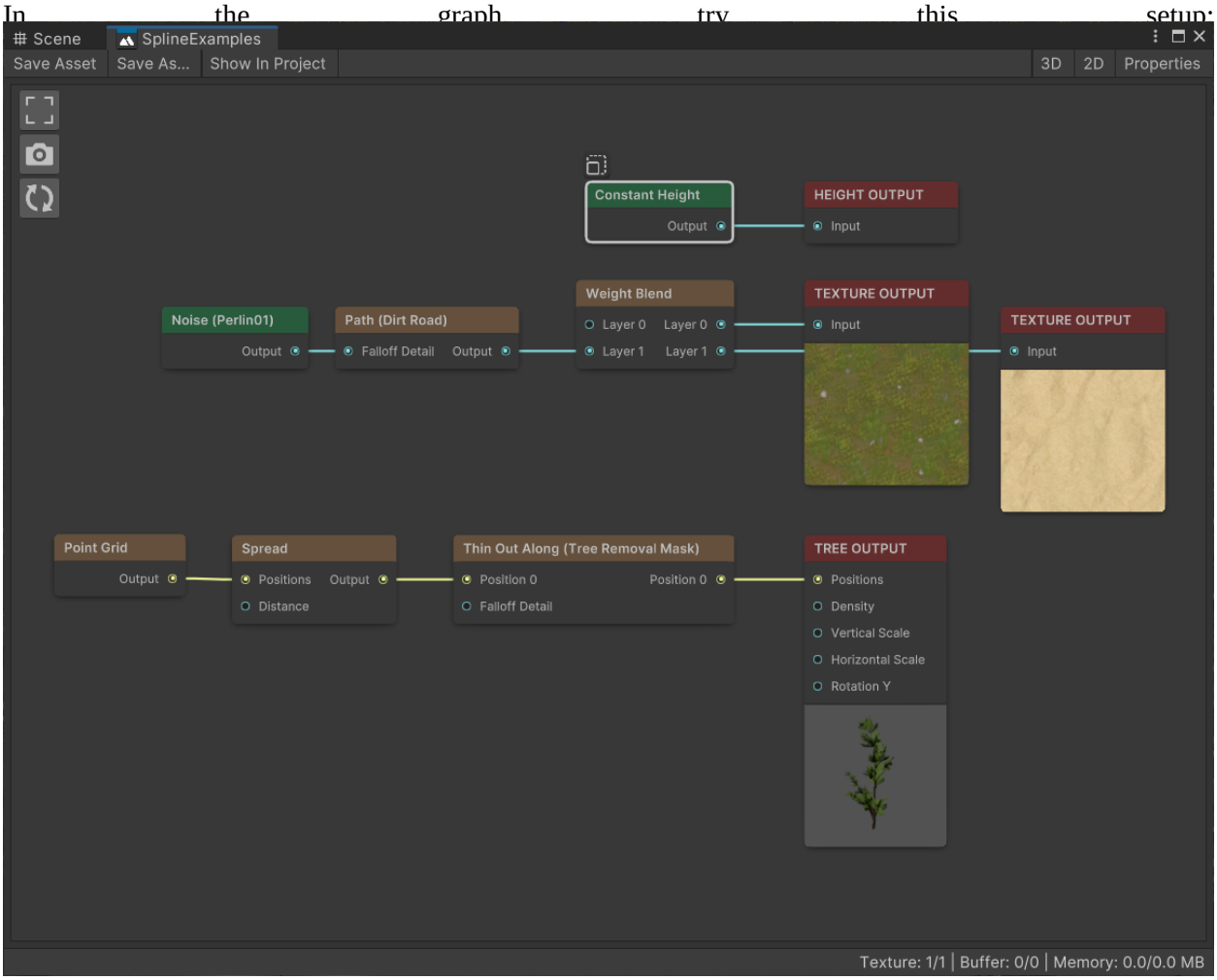
Add 2 evaluators to the spline with different properties especially Width like this:



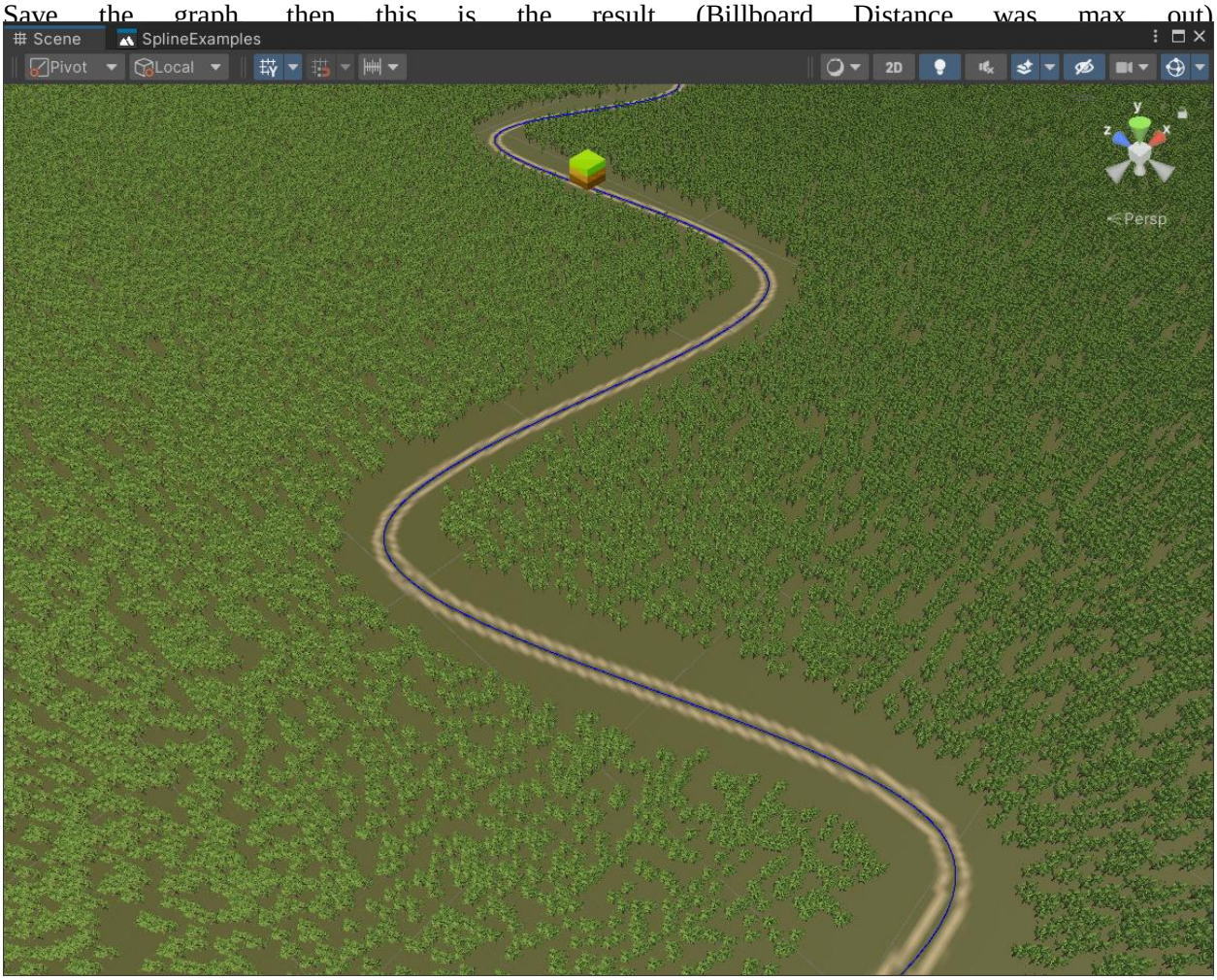
The first evaluator will be used for painting the path, while the second is for removing tree instances.

Setup a biome just to cover the terrain, and a graph for it.

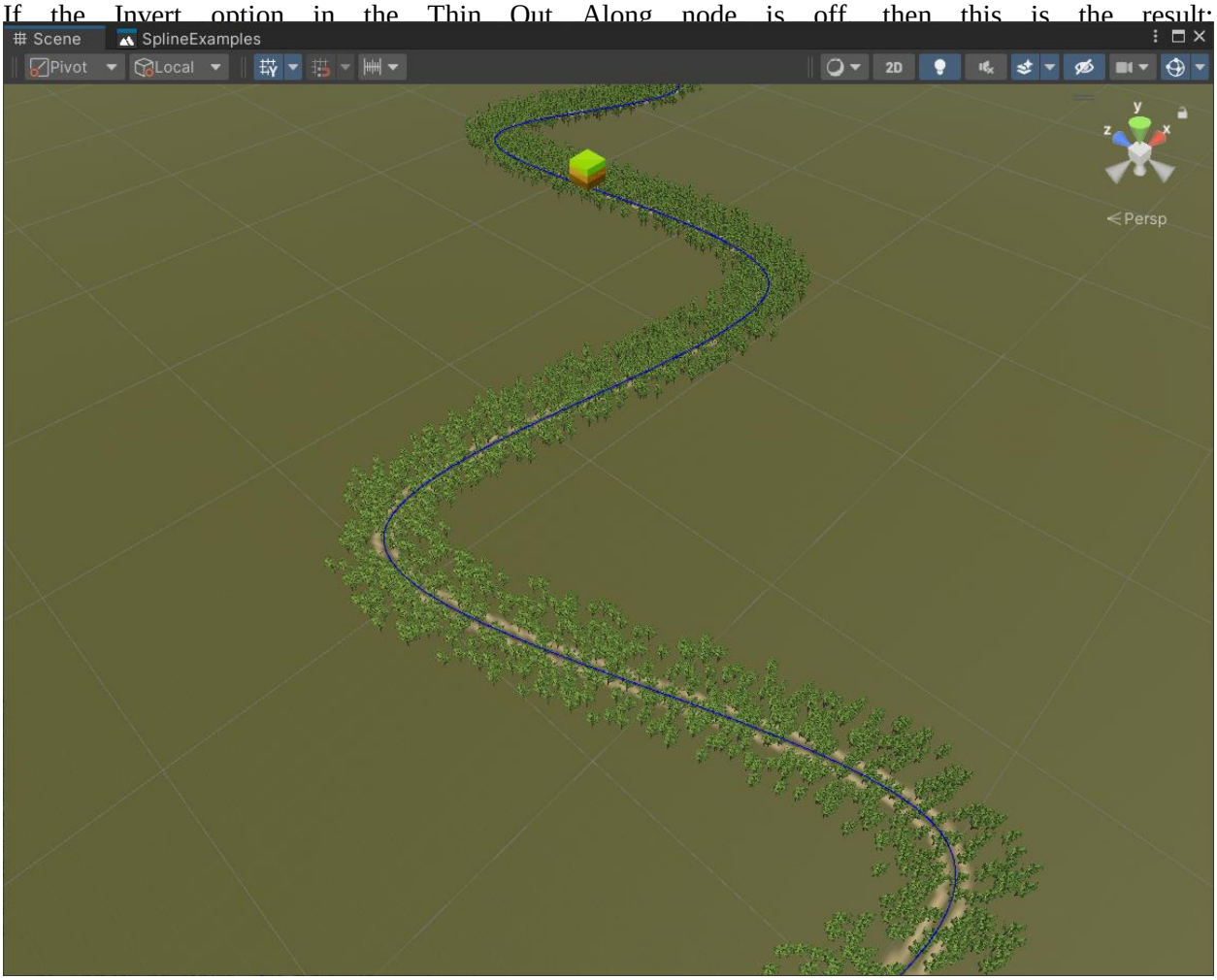
In the graph try this setup:



Save the graph then this is the result (Billboard Distance was max out)

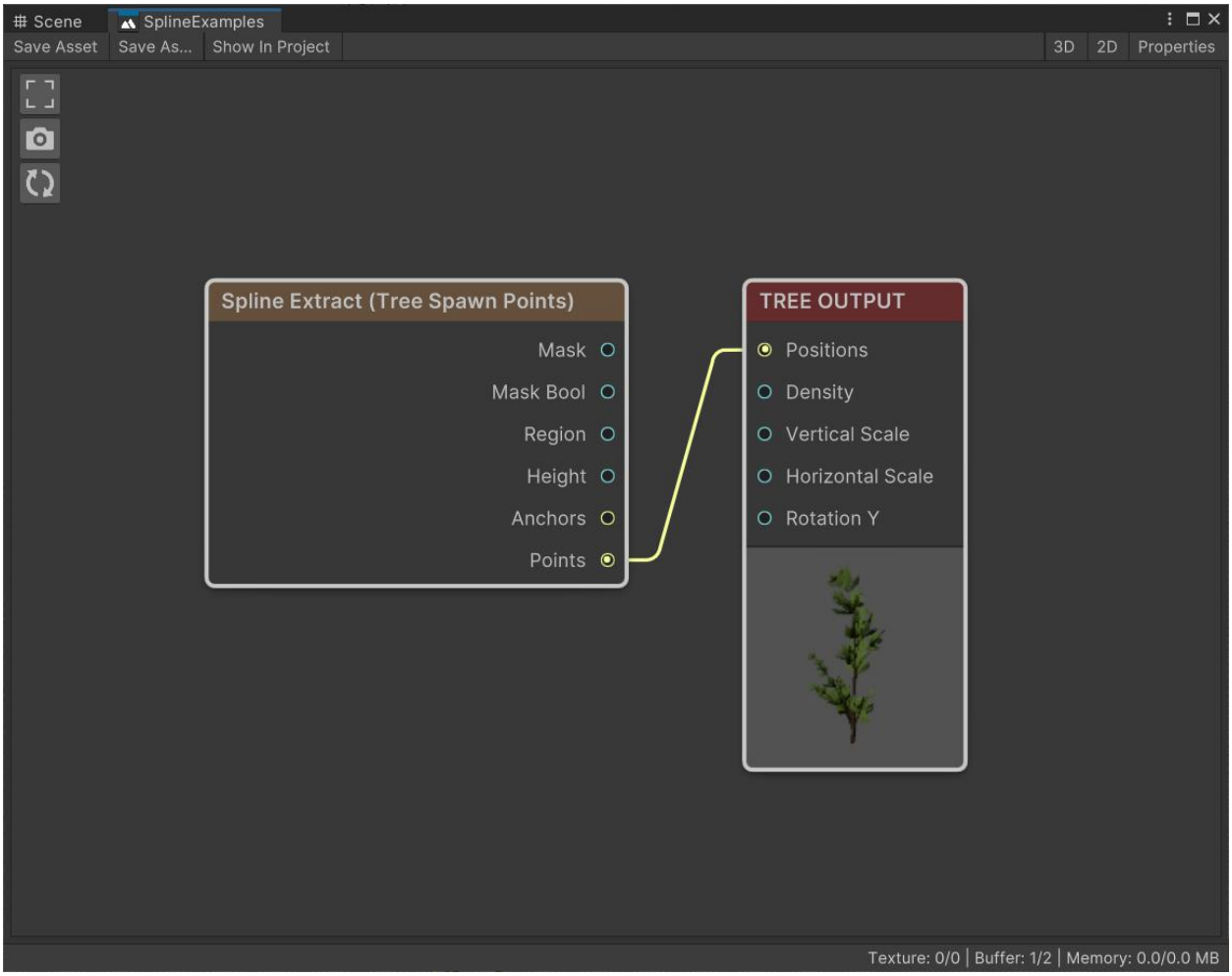


If the Invert option in the Thin Out Along node is off, then this is the result:

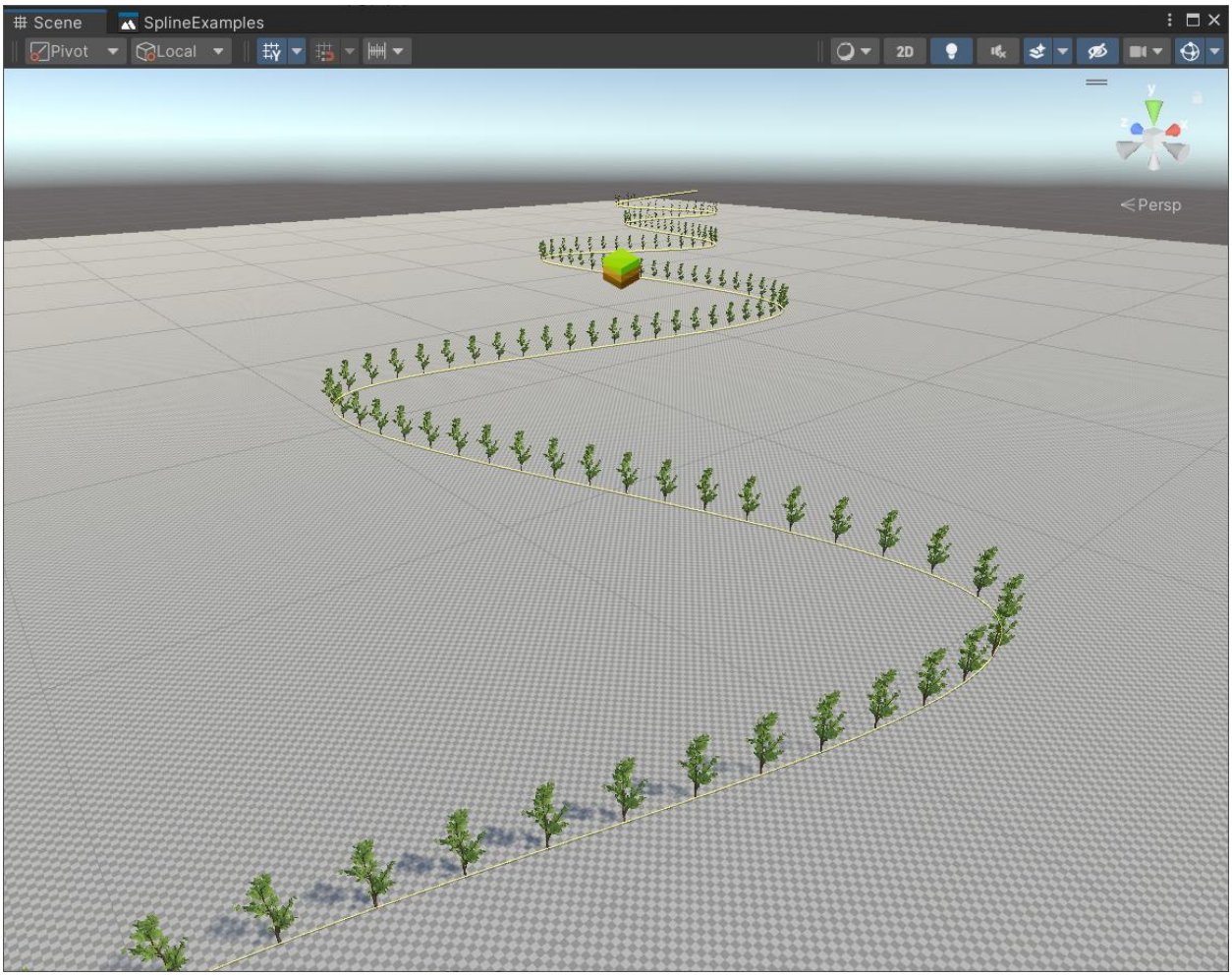


Spawn trees along the spline

Just a simple setup:

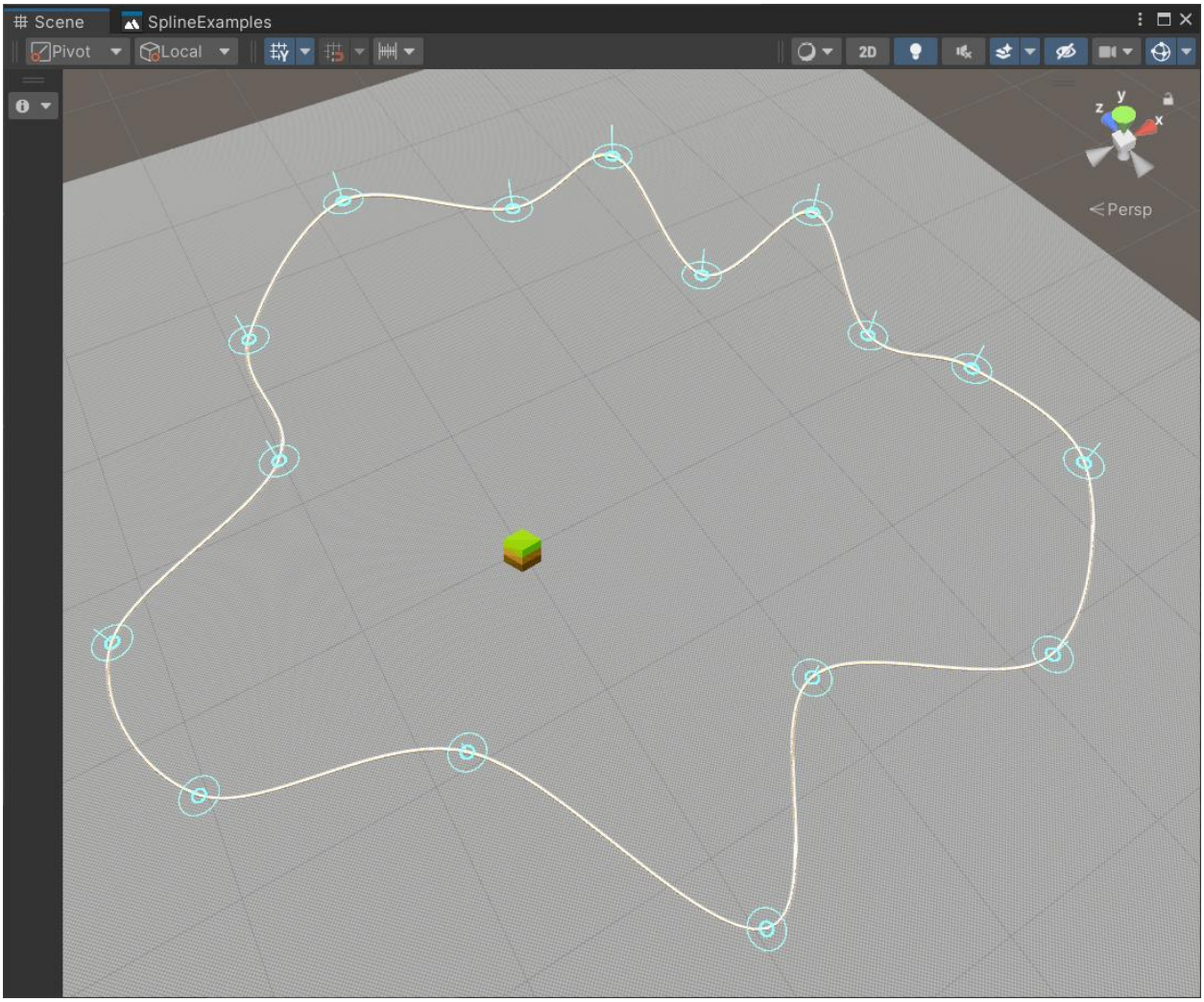


And the result:

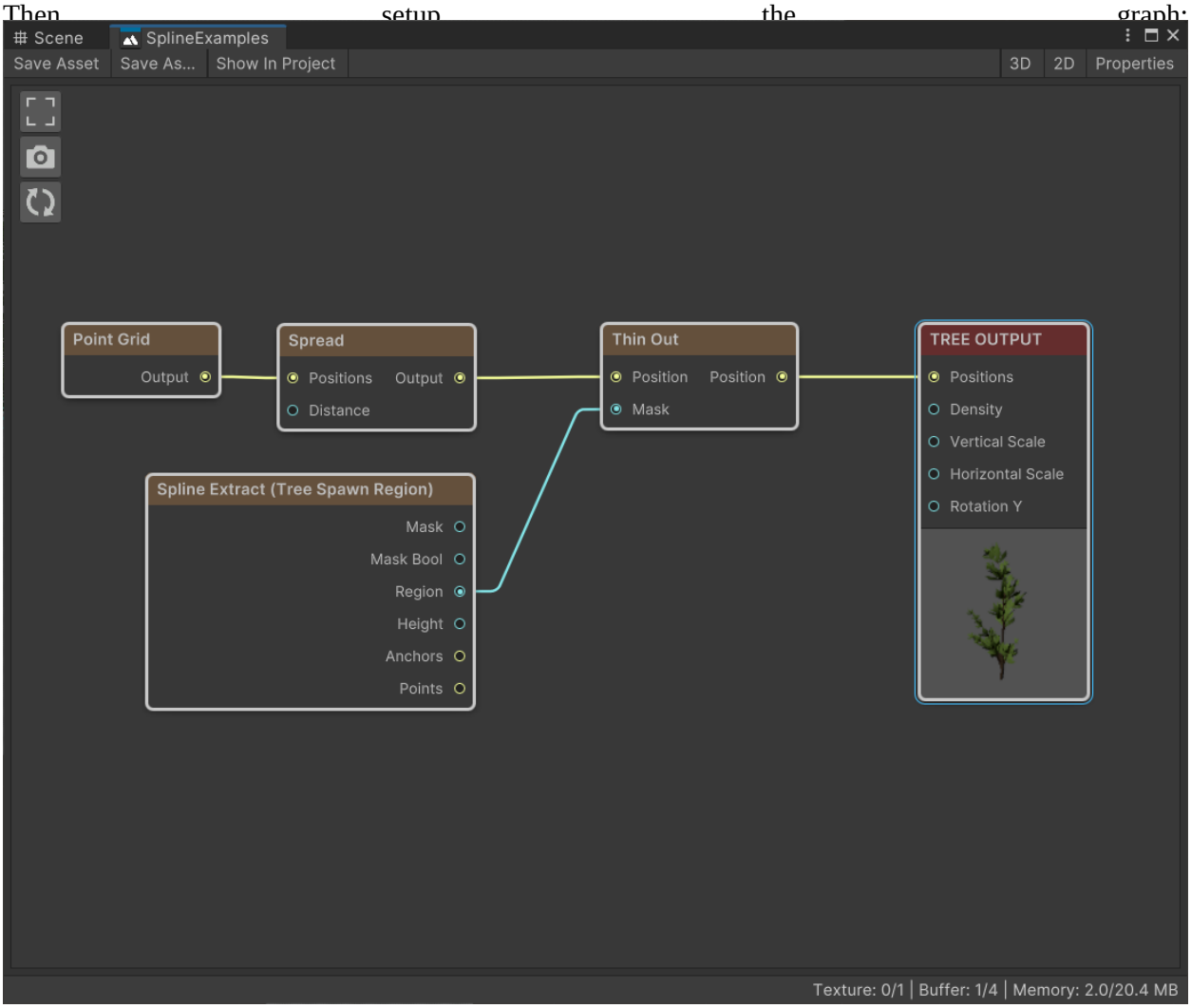

Scene
SplineExamples
Pivot
Local
2D
Persp

Spawn trees inside the spline loop

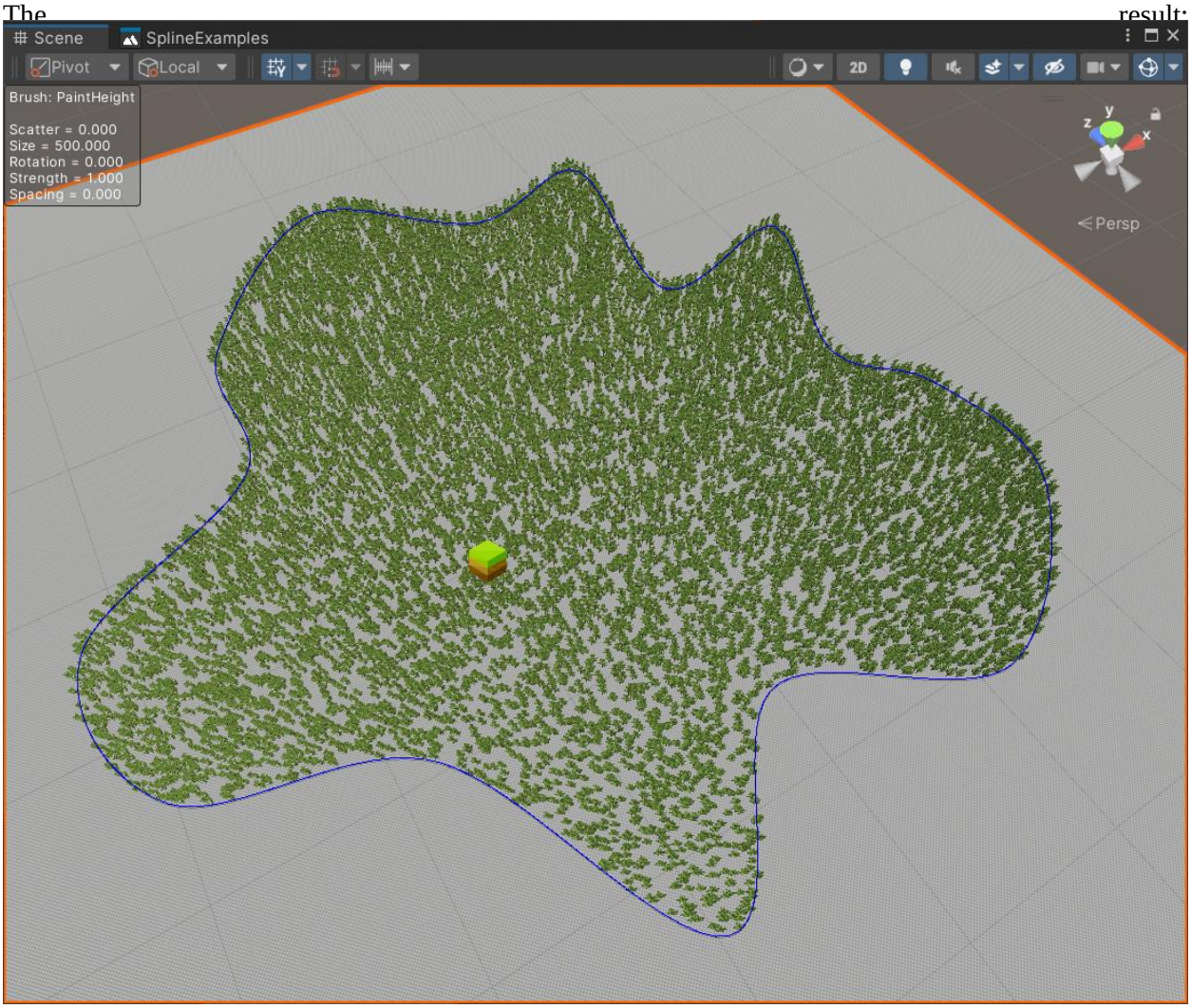
Create a “Closed” spline like this:



Then setup the graph:



The result:


Scene
SplineExamples
Pivot
Local
2D
Brush: PaintHeight
Scatter = 0.000
Size = 500.000
Rotation = 0.000
Strength = 1.000
Spacing = 0.000
y
z
x
Persp

When the mask is inverted with Math node: