The Grid System

A good base for a lot of tools

Reference for creation of UMG and Editor Widgets:

UMG: https://docs.unrealengine.com/5.0/en-US/widget-blueprints-in-umg-for-unreal-engine/

Editor: https://docs.unrealengine.com/5.0/en-US/editor-utility-widgets-in-unreal-engine/

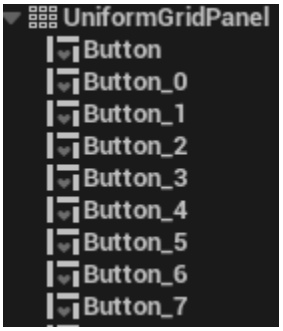
Base:

Custom UMG button widget

Editor widget blueprint

Steps:

Put a number of the custom button class as children of your uniform grid panel

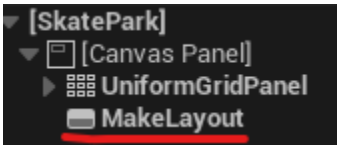


The number of buttons will have to be able to be cleanly squared, in this example there are 9 buttons

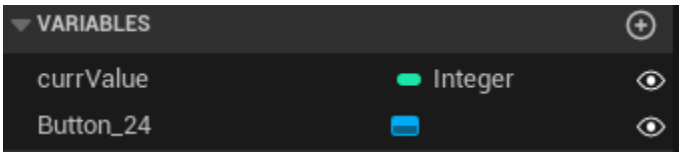
3 rows

3 columns

I also have a button, which is not a child of the uniform grid panel that is used to initialize the grid.

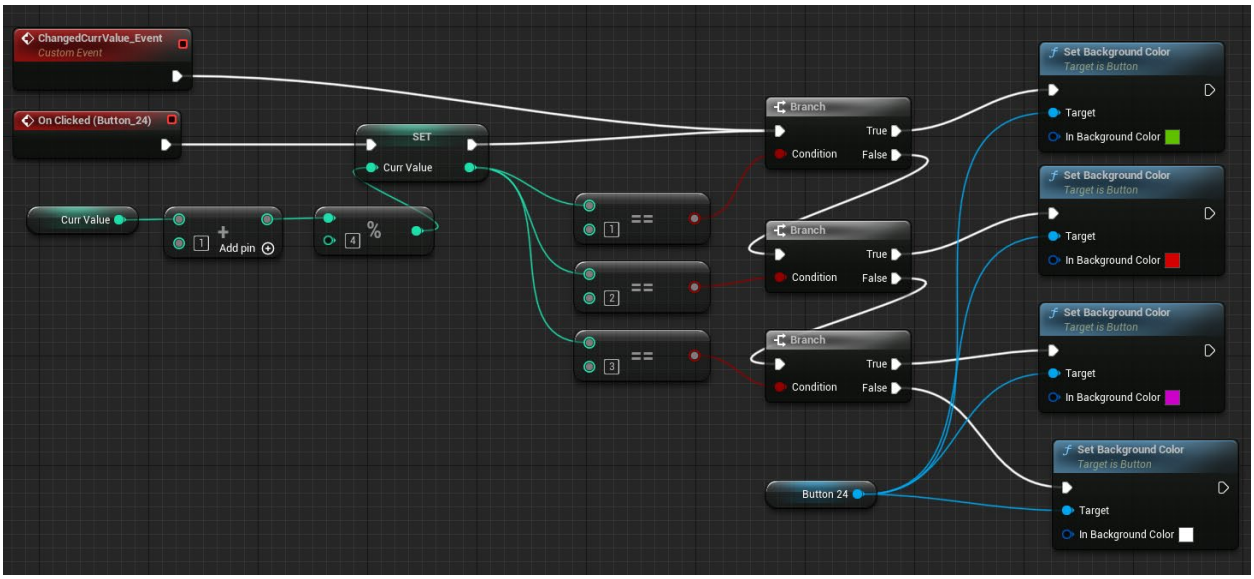


Custom Button Variables:



CurrValue must be public

Custom Button Graph:



Whenever a button is clicked the variable CurrValue is incremented. In the case that I am using this grid system there are four different settings I want the buttons to be at, but you could have as many settings as you want. Just set the modulus operator as the number of options you want to have.

In my case:

0: empty

1: Ramp

2: Loop

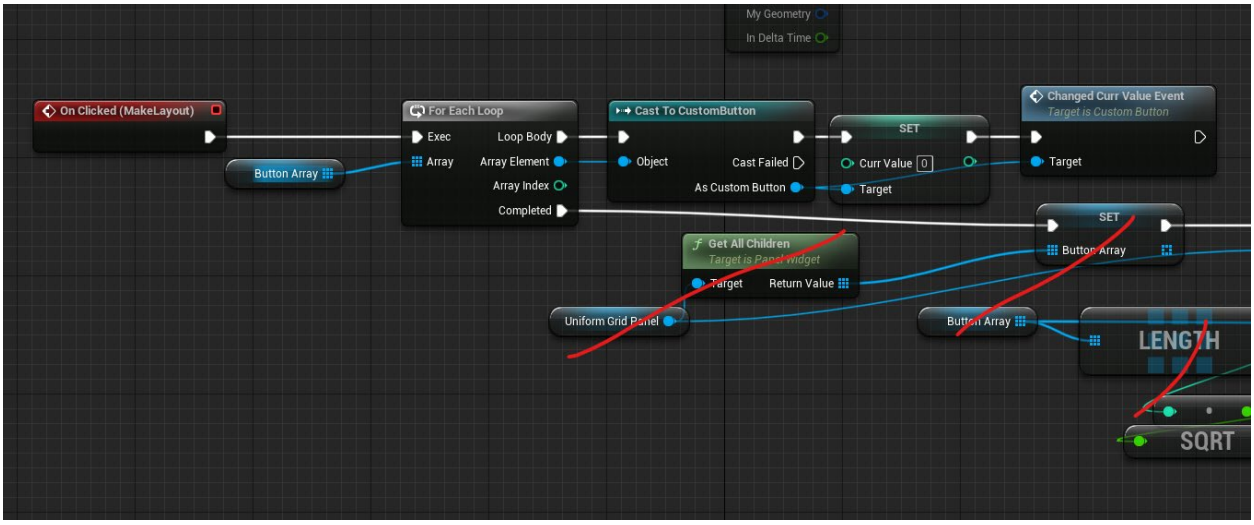
3: Bank

I created an event called ChangedCurrValue, this event exists so that when the user wants to set all the values of the buttons, they update color accordingly.

Grid System Variables:

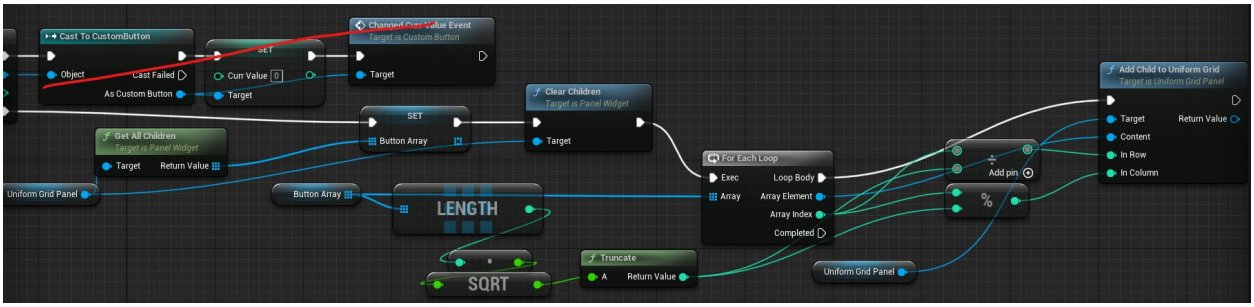
These are the two variables that matter. I also have 169 CustomButton objects initialized, but this was only because I could not figure out how to create widgets inside of an Editor Widget blueprint. In an editor widget blueprint you need the World Context Object, while in a UMG widget you do not. My workaround was just pressing Ctrl+V a whole bunch of times to make all the buttons. If the designer wanted to change the size of the grid they would have to go into the designer section and update the number of buttons to an amount that cleanly squares, but other than that it is a very scalable system in my opinion.

Grid System Graph pt. 1:



This section of the graph is where the values of the buttons are reset. I have a for each loop, which iterates through the ButtonArray and sets CurrValue to 0. The loop also calls the ChangedCurrValue event, which updates the color of the button.

Grid System Graph pt. 2:



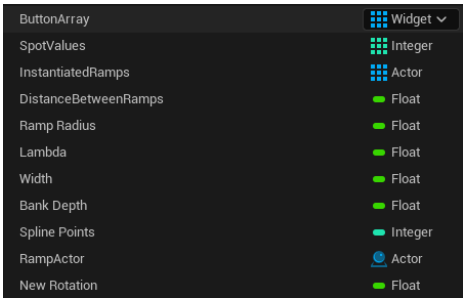
In this section of the graph, I am taking all the children from the uniform grid panel and setting the ButtonArray equal to them, I am then clearing the children from the uniform grid panel. This is because I spawned in all the buttons at the start so there were 169 buttons all sitting in the top left of the panel. I

then loop through the ButtonArray. Since you cannot make multidimensional arrays in blueprints I used division to find the row and modulus to find the column that the index is currently at. The variable that is affecting the index is made by finding the square root of the number of buttons in the ButtonArray. The final step of an iteration through the for loop is adding the button back into the uniform grid panel.

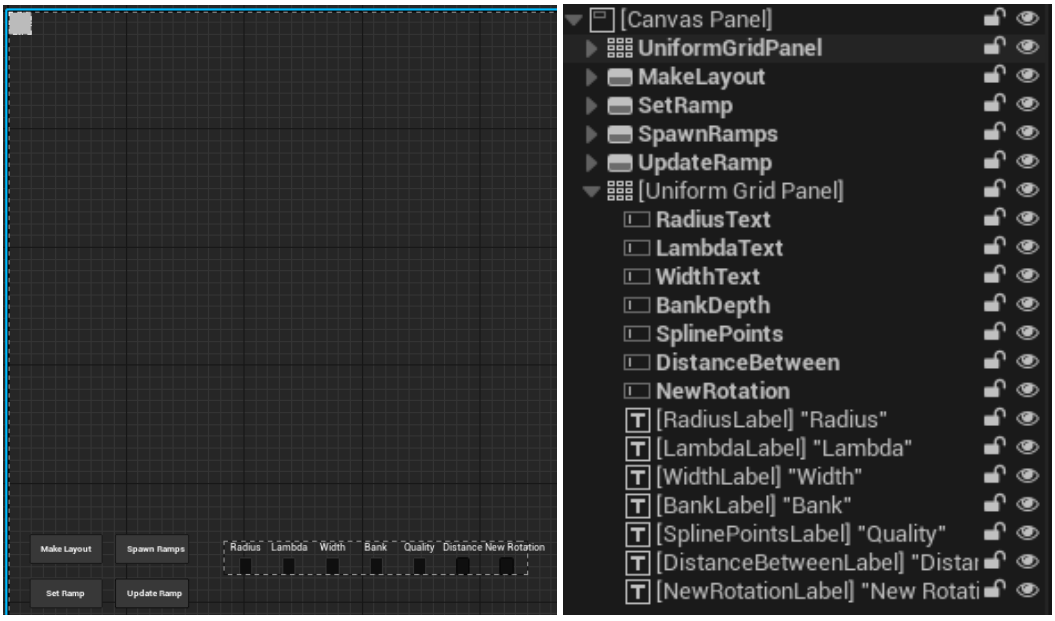
The Skate Park Tool

What we have all been waiting for

Variables:



I am going to start with the layout of the designer widget, since that was not covered at all in the above section.

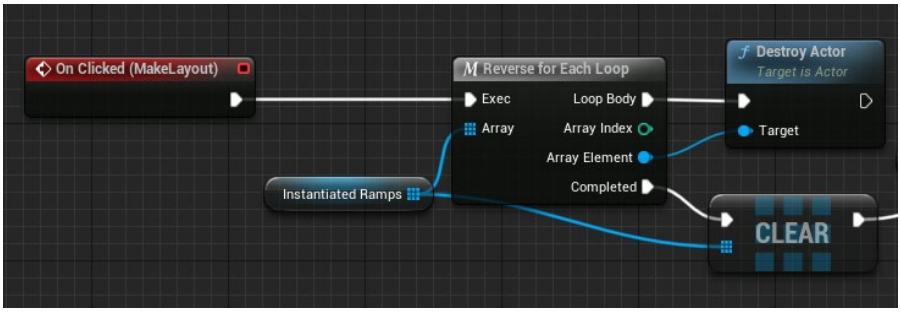


In the top left there is a uniform graph panel, in the bottom left there are four buttons, and on the right there are seven text boxes which are bound to variables in the graph.

The four buttons work as follows

Make Layout:

Initializes the grid. If there is already information on the graph, it will remove that information and destroy any ramps that have been spawned.



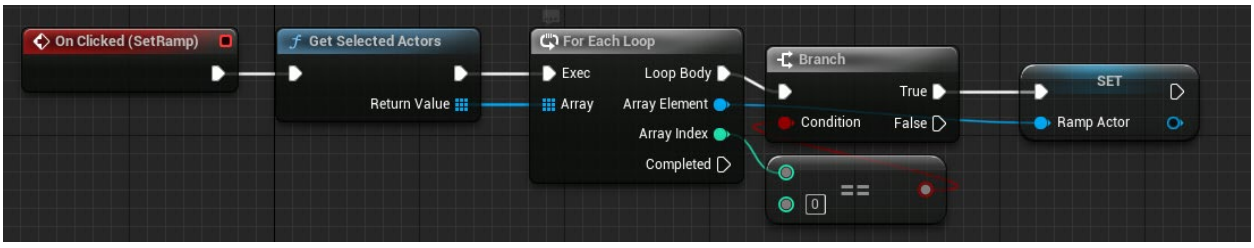
Update to Make Layout, destroys the instantiated ramps when clicked

Spawn Ramps:

Takes the information from the grid and spawns maps in the world using the information from the text boxes. The code for this will be covered in the graph section, since it is the most complicated.

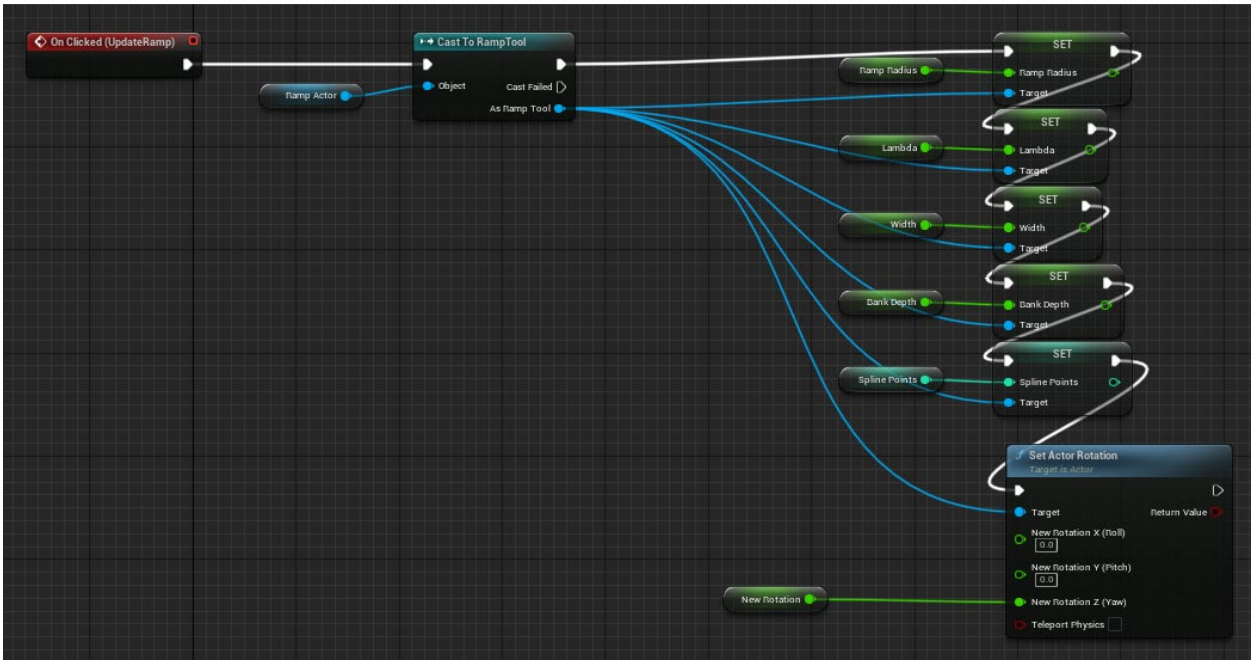
Set Ramp:

In order to spawn the ramps, you have to have a reference in game already. You need to select the object in game then click this button to set the value.



Update Ramp:

This will update the ramp you have set with the different values that are in the text boxes.



The text boxes are as follows:

Lambda: This determines the angle between each point placed on the spline

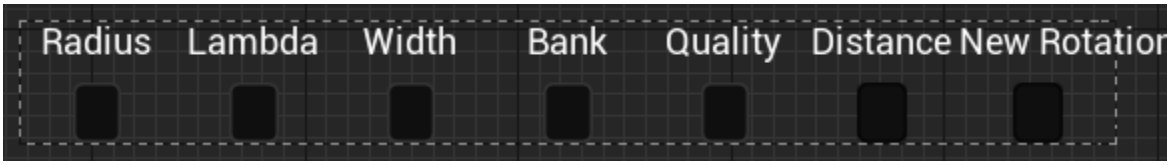
Width: This determines the width of the ramp

Bank: This determines the bank of a banked ramp only

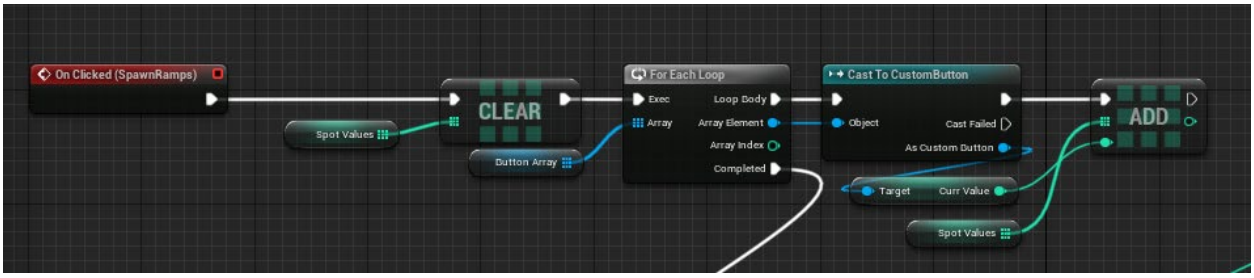
Quality: This determines the amount of spline points

Distance: This determines the distance between each ramp

New Rotation: This value will only be applied on Update Ramp

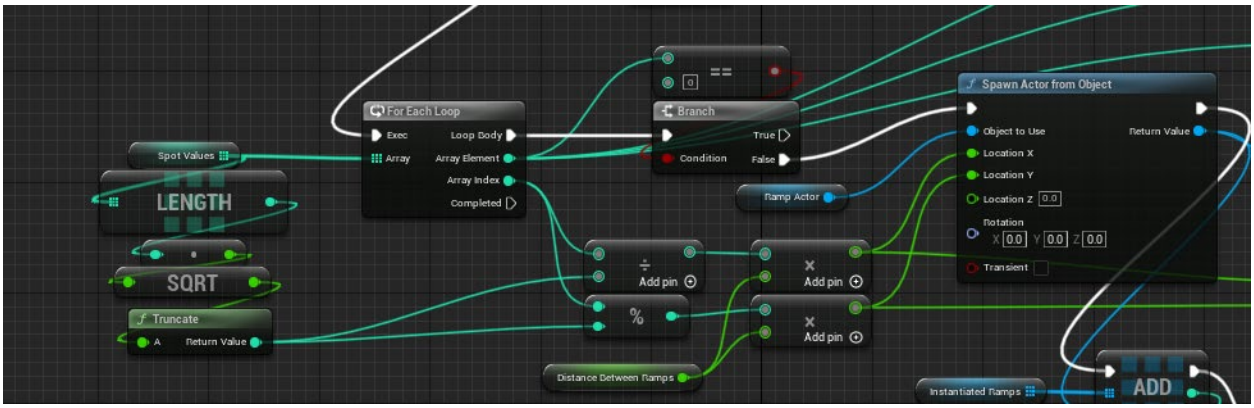


Graph pt. 1



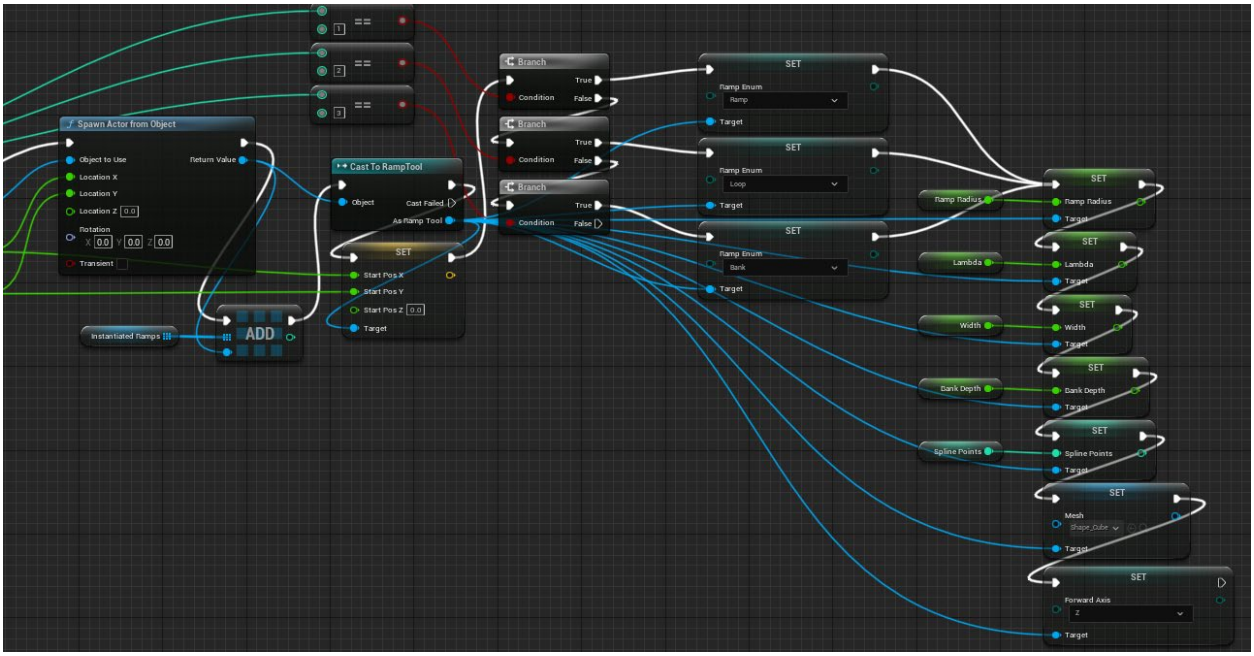
This portion of the graph takes the CurrValues of each button in the button array and adds them to an integer array.

Graph pt. 2



This section of the graph is spawning in the actors, it first breaks the array up into a multidimensional array as seen above. It then multiplies the current row and column by the distance between and instantiates an actor from the ramp object (which had been set with the aforementioned button). If the current value in the array is 0, then nothing happens in that iteration.

Graph pt. 3



Every time an actor is spawned it is added to the Instantiated Ramps list, then all the different values of the ramp are set. Please see the ramp tool documentation, if you would like to understand the inner workings of the class.

The tool in practice:

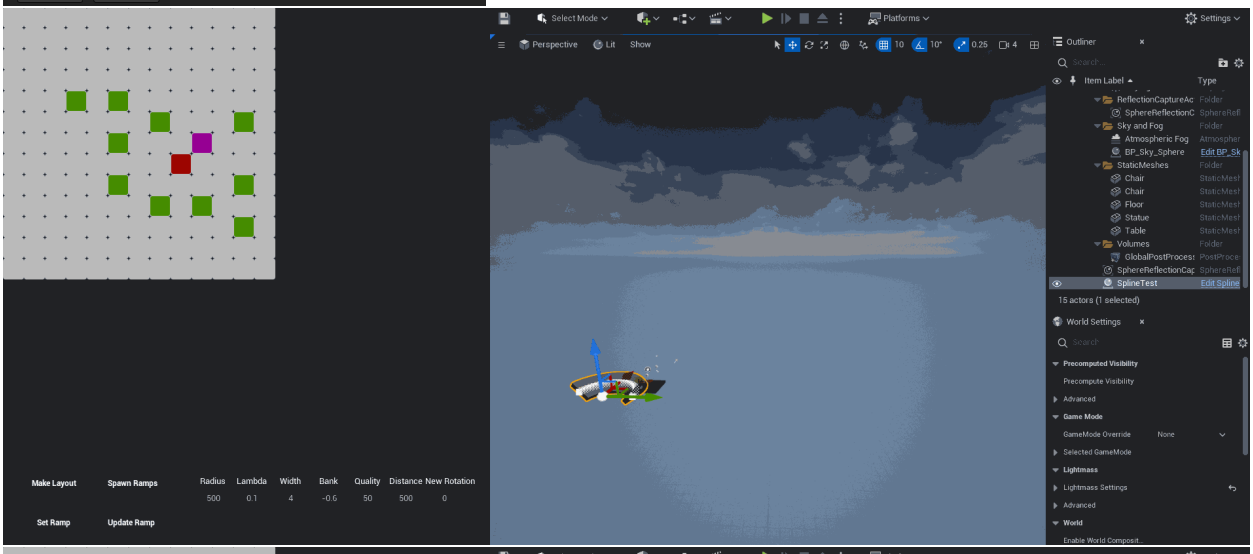
After you have spawned in all the ramps, you will notice that they are all the same as the ramp you selected. You have to select all the ramps and move them slightly in order to update them. One of the possible reasons for this I found was because the constructor is only called when the actor is first created. I did some tests to try and take advantage of this, but I was stuck with the problem of having to move all the ramps.

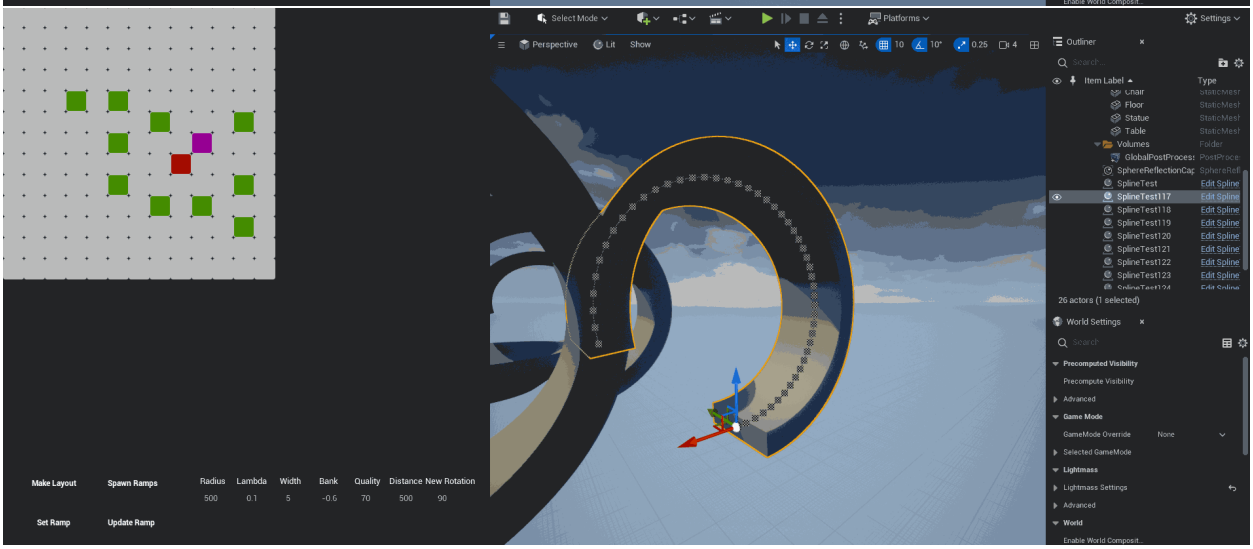
It is a bit of a hiccup in the working of this tool for a designer, but you have to select all the ramps (select the first one, hold shift, then select the last one), then move them all slightly. They will update to your preferred ramp enum.

You will also have to move a ramp that you have updated in order to see the changes.

Some gifs:


Make Layout
Spawn Ramps
Set Ramp
Update Ramp
Radius Lambda Width Bank Quality Distance New Rotation
500 0.1 4 -0.6 50 500 0


Make Layout
Spawn Ramps
Set Ramp
Update Ramp
Radius Lambda Width Bank Quality Distance New Rotation
500 0.1 4 -0.6 50 500 0
Select Mode
Platforms
Settings
Perspective
Lit
Show
Outliner
Item Label
Type
SplineTest
Edit Spline
15 actors (1 selected)
World Settings
Precomputed Visibility
Precompute Visibility
Advanced
Game Mode
GameMode Override
None
Selected GameMode
Lightmass
Lightmass Settings
World


Make Layout
Spawn Ramps
Set Ramp
Update Ramp
Radius Lambda Width Bank Quality Distance New Rotation
500 0.1 5 -0.6 70 500 90
Select Mode
Platforms
Settings
Perspective
Lit
Show
Outliner
Item Label
Type
SplineTest
SplineTest117
SplineTest118
SplineTest119
SplineTest120
SplineTest121
SplineTest122
SplineTest123
Edit Spline
26 actors (1 selected)
World Settings
Precomputed Visibility
Precompute Visibility
Advanced
Game Mode
GameMode Override
None
Selected GameMode
Lightmass
Lightmass Settings
World