Author: Lê Vương Gia Huân.

Animated bar chart prototype documentation

This document provides instructions and write-up for the animated bar chart prototype.

Table of contents

1. Quick instructions:

1.1. Requirements:

1.1.1. The data table must be in this format:

	Data
08:00	10
09:00	28
10:00	13
11:00	42

Or (csv):

```
,data
08:00,10
09:00,28
10:00,13
11:00,42
```

The reason is because of the way Unreal handles data tables, data of the same type should be placed in a column. In case you have them in a row, you can swap row and column using something like this: https://www.liveflow.io/product-guides/switch-rows-and-columns.

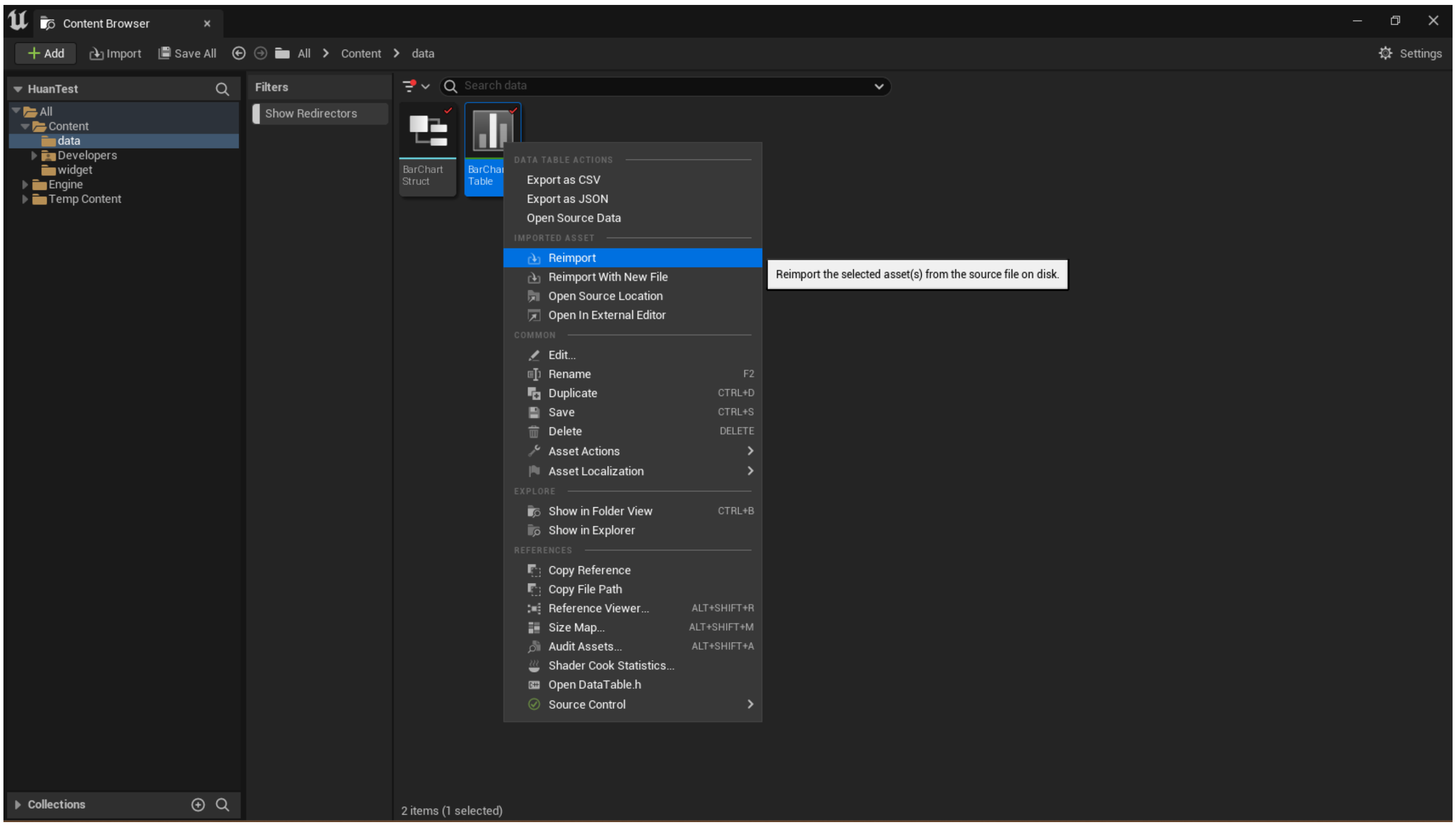
1.1.2. Unreal version 5.1+ and understanding of Blueprint and the widget system.

1.2. How to update the bar chart:

1.2.1. It is confirmed in an email that the “connection between chart and data table” is “editor only”.

1.2.2. Reimport the data table:

1.2.2.1. If the .csv file’s path isn’t changed, right click the Content/data/BarChartTable and click reimport.

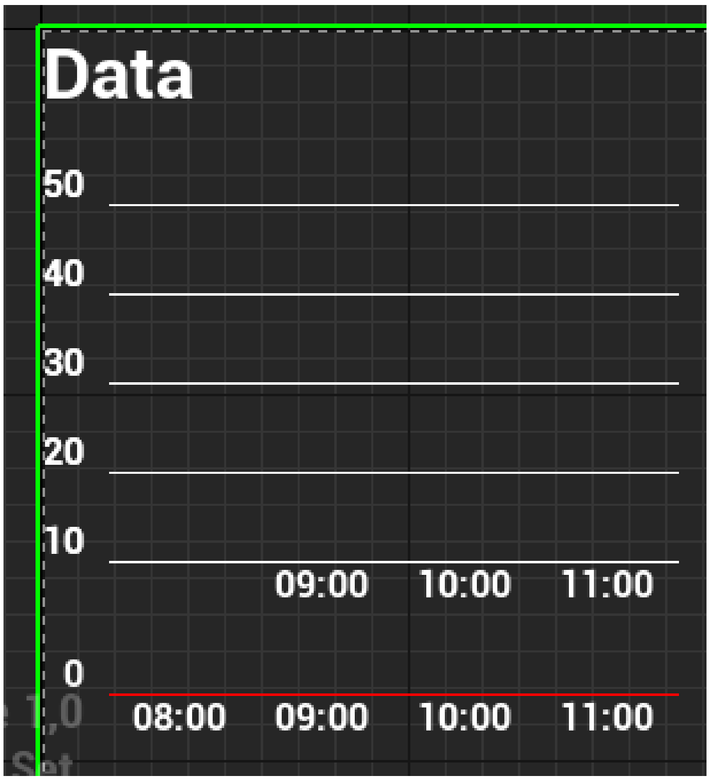


Reimport the selected asset(s) from the source file on disk.

1.2.2.2. If the file path changed (new .csv with different name or location for instance), choose reimport with new file instead.

1.2.3. Recompile the bar chart Content/widget/BarChartWidget.

1.2.4. Sometimes, the bar chart will have unwanted behavior due to the nature of PreConstruct, like the screenshot below. Simply recompile it or restart the editor to fix the issue.

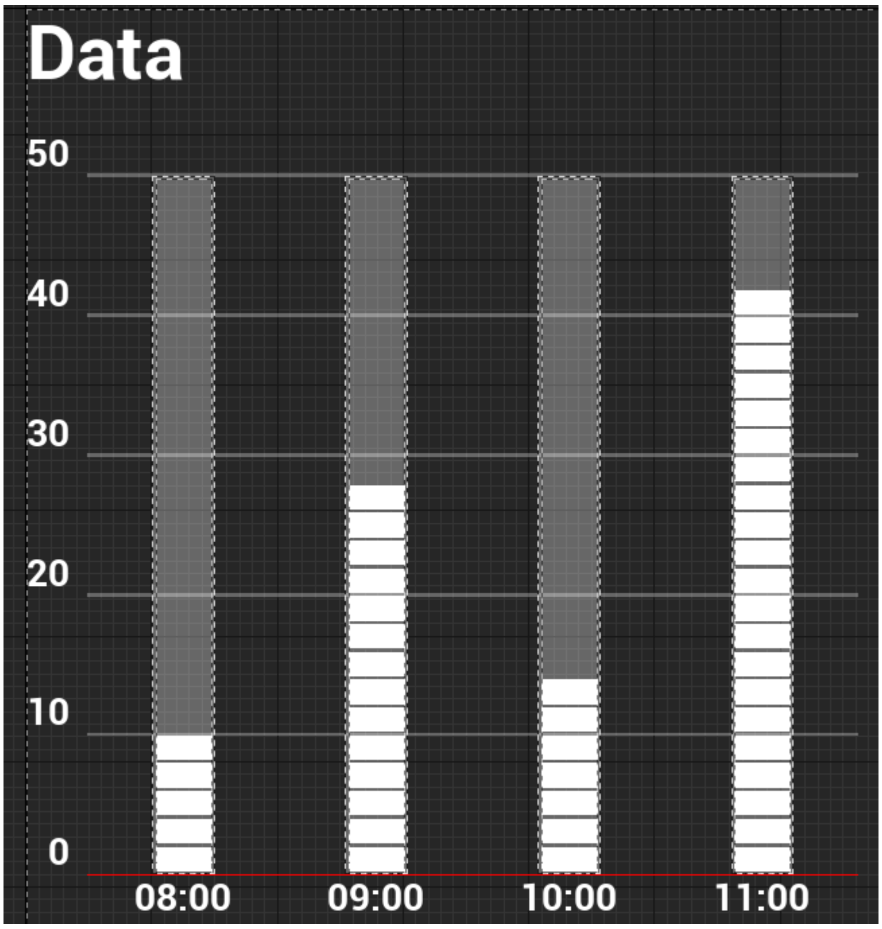


2. References:

2.1. The flow of the game starts from the level TestLevel, it spawns the TestWidget in the level blueprint. BarChartWidget is placed in TestWidget, it will generate the bar chart.

2.2. The bar chart has 2 main parts:

2.2.1. The bar chart widget (Content/widget/BarChartWidget) generates the content of the bar chart from the given data table. Here are the terminologies used:



2.2.1.1. "Data" is the title.

2.2.1.2. The numbers in the left columns are marker numbers.

2.2.1.3. The lines which strike through the chart are marker lines.

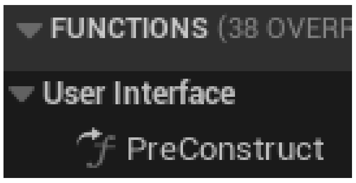
2.2.1.4. The 4 stacked columns are the bars.

2.2.1.5. The texts in the row below are value names.

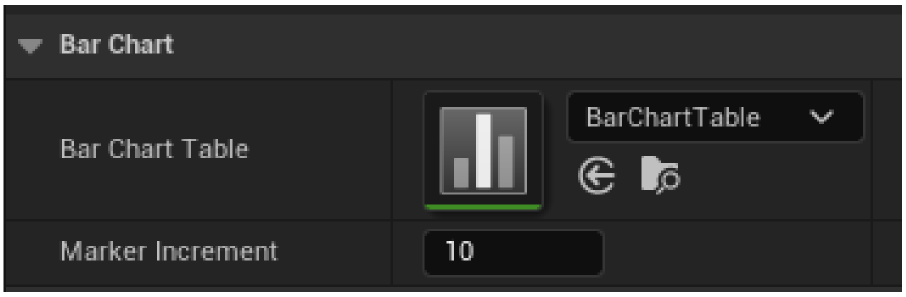
2.2.2. The bar widget (Content/widget/BarWidget) will be filled with stacks. A stack is a visual representation of a fraction of the given value. The bar widget also handles the animation of the stacks.

2.3. BarChartWidget's PreConstruct behavior:

2.3.1. PreConstruct is implemented as an overridden function, not event. You can find it in the Functions/UserInterface category in the graph editor.



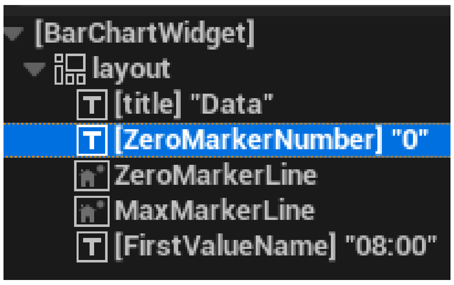
2.3.2. In the widget default settings, you can select the data table used for this bar chart and the increment of the marker numbers. The data table must use the bar chart struct (Content/data/BarChartStruct).


Bar Chart

Bar Chart Table: BarChartTable

Marker Increment: 10

2.3.3. Changing the ZeroMarkerNumber:



[BarChartWidget]
layout
[title] "Data"
[ZeroMarkerNumber] "0"
ZeroMarkerLine
MaxMarkerLine
[FirstValueName] "08:00"

2.3.3.1. Its top padding and right padding will be copied to the other marker numbers.
2.3.3.2. Its font will be copied to all other text blocks except the title.

2.3.4. ZeroMarkerLine's vertical alignment will be copied to other line(s).

2.3.5. Changing the MaxMarkerLine:

2.3.5.1. Its brush will be copied to other lines except the ZeroMarkerLine.
2.3.5.2. Its top padding will be copied to the max number marker.
2.3.5.3. The bars will be offset (nudged as a grid slot) by half of its size.

2.3.6. FirstValueName's padding will be copied to other value names.

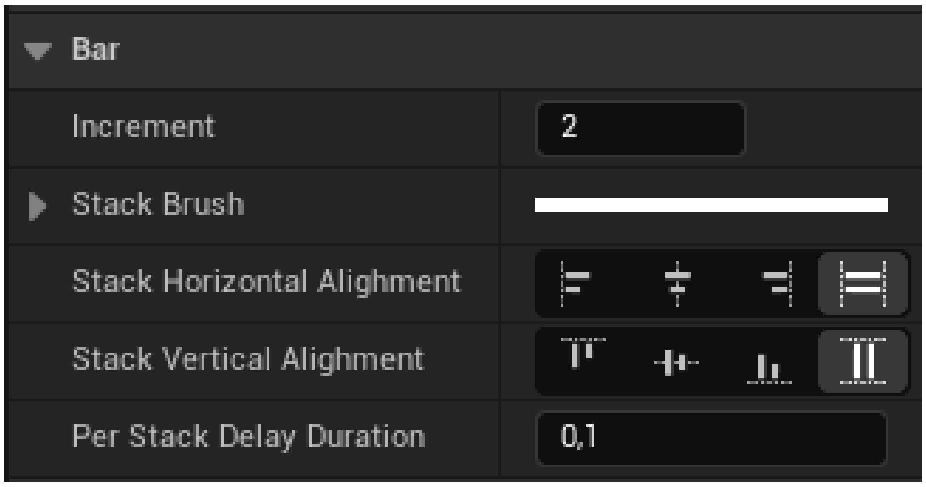
2.3.7. The flow as commented in the graph editor:

2.3.7.1. Generates marker(s).
2.3.7.2. Offsets placed widget(s).
2.3.7.3. Generate value name(s).
2.3.7.4. Spawn bar(s).

2.4. BarWidget's PreConstruct behavior:

2.4.1. PreConstruct is implemented as an overridden function, not event. You can find it in the Functions/UserInterface category in the graph editor.

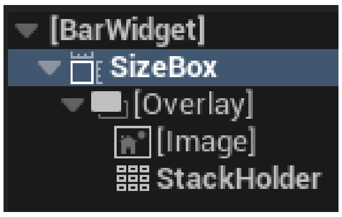
2.4.2. You can change how the value is separated into stacks and stack's style in the widget default settings.



Bar

Increment	2
Stack Brush	
Stack Horizontal Alighment	
Stack Vertical Alighment	
Per Stack Delay Duration	0,1

2.4.3. You can change the size of the bar using the SizeBox.



[BarWidget]
SizeBox
[Overlay]
[Image]
StackHolder

2.4.4. PreConstruct will first separate the bar into part(s) then spawn stack(s) with predefined style.

2.5. BarWidget's EventConstruct behavior:

2.5.1. In the widget default settings, changing the PerStackDelayDuration affects the speed of the animation. Higher is slower.

2.5.2. EventConstruct will first make all stacks hidden then slowly make them visible by sequence.

3. A write-up about this bar chart:

- 3.1. There are 3 factors being considered in the making of the animated bar chart prototype;
  - 3.1.1. The given requirement to connect the data table to the bar chart in the editor. This is why the chart is highly procedural and most of the logic is in PreConstruct.
  - 3.1.2. The given design concept graphic. Although it is stated that "you don't have to make it exactly the same as the design", the design is interesting to follow.
  - 3.1.3. Further design development consideration. This prototype may require more work by an UI artist, so it's good to provide the artist the ability to easily insert new changes to the bar chart. For instance, as described in 2.3.3, changes to many texts can be automated.
- 3.2. I use byte as the data type because the given value is fairly small, I assume the value would stay under 256. It wouldn't be difficult to change to type int since Blueprint will notify us what needs to be changed after we change a variable type. And the process is easier in C++ if we use type aliases in the first place.