an asset for Unity 3d
by Oliver Wuensch
email: support@wuenschonline.de

the
Nodecanvas Actions

What is Rangemapper?

Do you need to **convert input data** like the value of a slider to move an object in 3 d space?

Do you need to **calculate the rotation** of the hands of a clock?
Rangemapper makes it easy to **convert floats, vector2 or vector3** from one space of numbers into another.

RangemapperCustom components can be used to create presets of Rangemappers stored in your scene so you you only need to pass the input to the correct RangeMapperCustom and you can reuse them whenever you need them in your scene.

Also you can **modify the output data by curves** which adds a convenient way to implement slow-in and slow out for animation or other animation purposes.

RangeMapperManager provides an easy way to **call Rangemappers by** a friendly **name**, if you have many RangeMappers in your scene.

RangeMapper comes with Nodes for Hutong NodeCanvas and Paradoxnotion Nodecanvas/ Flowcanvas visual programming environments.

NodeCanvas Actions installation

This document assumes that you are familiar with the NodeCanvas usage and interface and that you own NodeCanvas **and have it already installed in your project.**

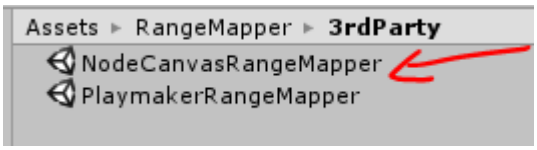
NodeCanvas is an asset created by Paradoxnotion bringing visual programming with FSMs and Behavior Trees to Unity 3d. Paradoxnotion has also a fantastic Flow Programming asset calles FlowCanvas available, that compliments NodeCanvas perfectly.

(For **Flowcanvas** all you have to do is add the **namespace Wuensch** to Flowcanvas and it will **automatically create all RangeMapper functionality** in FlowCanvas Nodes for you, tested and works great).

If you have NodeCanvas installed:

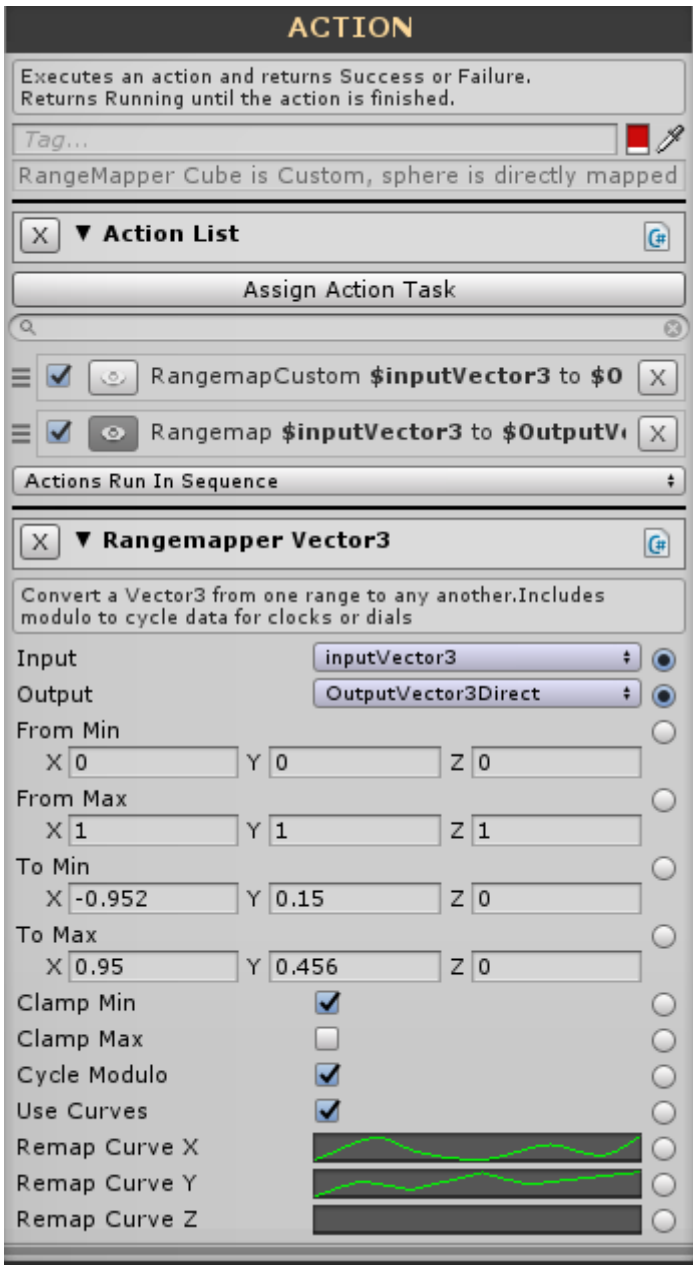
To use the NodeCanvas RangeMapper Nodes simply install the provided **NodeCanvasRangeMapper.unitypackage** found in Assets->3rdParty

by double clicking it in the Project manager of Unity3d.



After installation when you open a BehaviorTree or FSM NodeCanvas and create an action node you will find these new actions in the **NodeCanvas Action** manager in the category “Rangemapper”:

Create a new FSM Owner or Behavior Tree Owner on a GameObject and add a new Action Node, then with the Node selected call up the Action Browser in NodeCanvas to find the Rangemapper Actions in the category RangeMapper to assign them to your action list.



The nodes **Rangemapper** (for floats input) , **RangemapperVector2** and **RangemapperVector3** (for Vector input) are used to directly rangemap inputdata into a different number space.

This means that you setup all releavant variables in the nodes.

The Nodes RangemapperCustomFloat, RangemapperCustomVector2, RangemapperCustomVector3

are meant to be used to access RangeMapperCustom components set up somewhere in your scene.

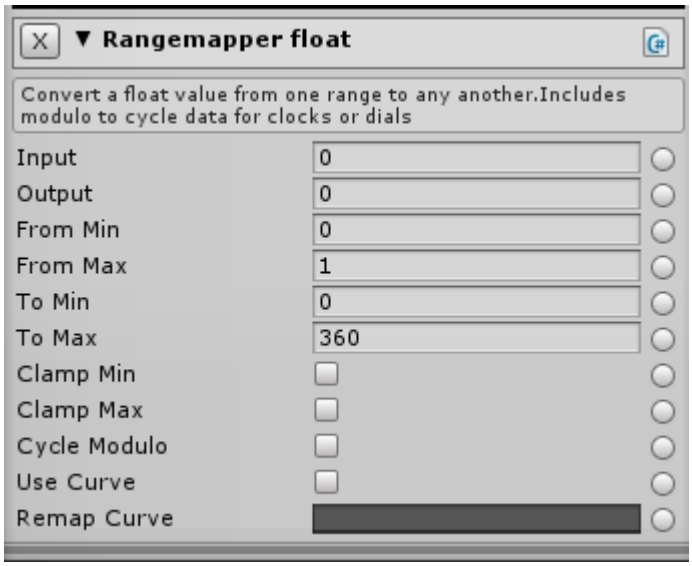
RangeMapperCustom components are **like presets that you configure** in your scene and can then reuse any time you want without configuring all variables again and again.

Read everything about RangeMapperCustom components in the RangeMapper_Manual.pdf that comes with this asset.

Also Check out the NodeCanvas version of the **demo files** to see how the Rangemapper actions can be used to drive animation.

I assume that you know how to use actions in NodeCanvas.

If you assign a Rangemapper action to an action node in NodeCanvas it will look like this:



The variables to use are the same as in the RangeMapperCustom components. You can read a detailed explanation in the RangeMapper_Manual.pdf .

Assign a variable for the output data in the Output field of the action like you normally do in any NodeCanvas action and configure your input data and you are ready to go .

Rangemapper action variables explained:

The parameters are:

FromMin and **fromMax:**

These are float numbers for the minimum and maximum of your **input number space**, **toMin** and **toMax** are the minimum and maximum of the corresponding **output space**.

Your minimum and maximum values **can be in the negative number ranges** if you need that. RangeMapper will take care that everything maps correctly.

For example if you set fromMin and toMin to a range of to 0f (0 float value) and 1f and your toMin/toMax to 0f and 360 you get a RangeMapper that will output 180 if you input 0.5.

This can be used to map input to a rotation for example (Check out the DemoClock scene in the demos folder for this).

If you do not clamp or use cycleModulo or curves the RangeMapper **will not limit the return value to the minimum and maximum**, you use these min max ranges basically to match the input and output ranges and the number spaces continue below and above.

ClampMin and clampMax:

To limit the input so that the min or max are never exceeded simply activate the **clampMin** and/or **clampMax** boolean switches. If you clamp both and your fromMin/ fromMax are 0f/1f an input of 2.4 will be clamped at the maximum of 1.0 and an input of -200.5f will be treated as if it were 0f.

CycleModulo:

For a clock for example if you input a counter that counts the passed seconds like you get in Unity from Time.time you would want the RangeMapper for the hand of your clocks rotation degrees to return 360 or 0 every 60 seconds, since this means that a minute has passed and the clock's hand has completed a turn.

Since the input counter like that of Time.time in Unity simply adds up furter and further the RangeMapper has to take care of that.

By activating the **cycleModulo** bool switch the rangeMapper **calculates how often the fromMin-fromMax number distance fits into the input and maps the resulting rest** (this is called a modulo) to the output number Range.

if the input range is 0 to 60, so whenever the input exceeds a multiple of 60 it will become 0 again.

This is mapped to a range of 0 to 360 (degrees) to get the rotation for the clock.

Edit:

To be correct with the clock I should have entered a fromMin/ fromMax of 0 to 59 and 1 to 360 as output for a correct result. I made a mistake when setting up the demo, sorry.Edit:
To be correct with the clock I should have entered a fromMin/ fromMax of 0 to 59 and 1 to 360 as output for a correct result. I made a mistake when setting up the demo, sorry.

Remap Curve:

if you activate the **useCurve** bool switch and configure the **remapCurve** (simply click on the curve rectangle) this curve is used to manipulate the input values between fromMin and fromMax (by default a Unity AnimationCurve clamps input, no matter if the ClampMin/clampMax is being used).

You can use this to achieve nice curved output or slow-in/slow-out for animation purposes. For Vectors you can manipulate the separate axis with independent curves, resulting in a lot of flexibility.

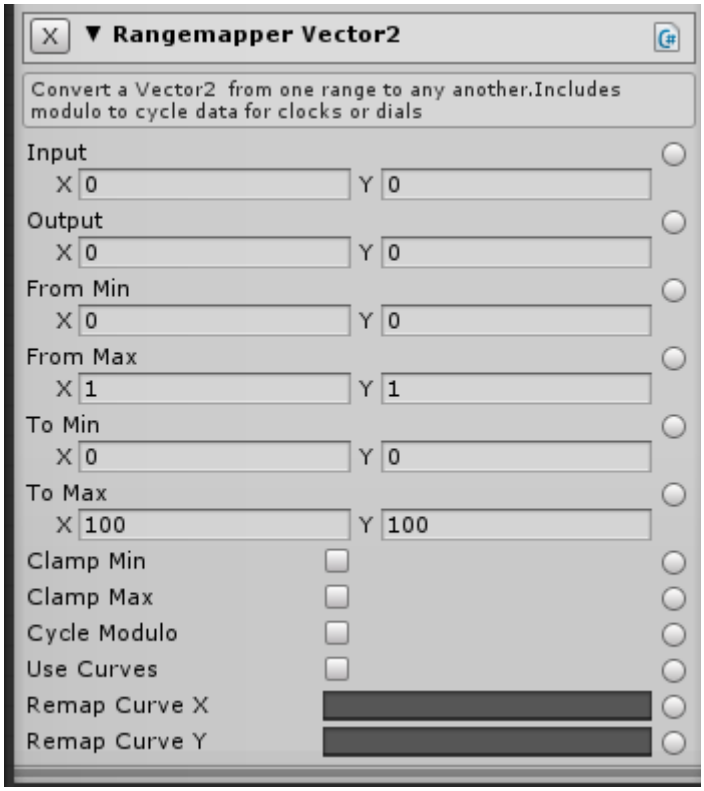
One important thing to know when working with vectors and curves:

If you use only the curve for one vector axis- and do not configure the other curves - only the value for the vector that has a valid curve is being used and clamped. **Empty curves without any points are ignored** and the input value is passed into RangeMapper unaltered.

If you do not want this behavior you either have to configure the other curves (with the linear preset for example) or clamp the input value with the clampMin/ clampMax switches.

Output (read only):

Assign a variable of the appropriate type (float, vector3, vector2) here to use your output data.



The actions for Vector2 and Vector3 use work similarly, only with Vector data as input and output..

RangeMapperCustom actions

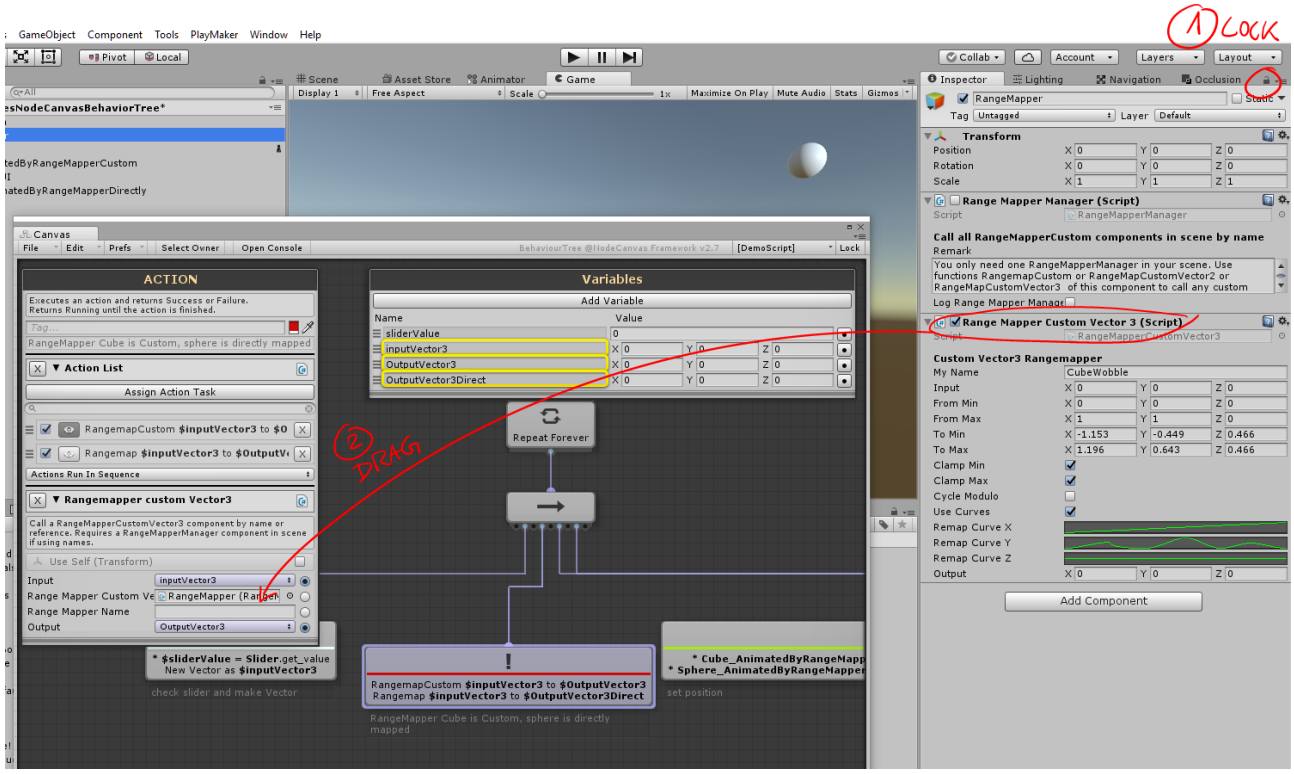
These actions send input and receive output using custom Rangemappers created with the **RangeMapperCustom components** in your scene.

Think of RangeMapperCustom components as **presets** of RangeMappers with individual names that you can conveniently call if needed.

Read all about **RangeMapperCustom** components in the RangeMapper_Manual.pdf that comes with this asset.

If you have set up RangeMapperCustom components in your scene use these actions to access them either by name or by direct link.

For inserting a direct link the easiest way is to **select the GameObject** your RangeMapperCustom is on and the **lock the Unity Inspector** with the tiny padlock symbol on the right top, then **select your NodeCanvas Owner** and the **Actionlist** the action is on and **drop the RangeMapperCustom** from inspector into the slot in the action.

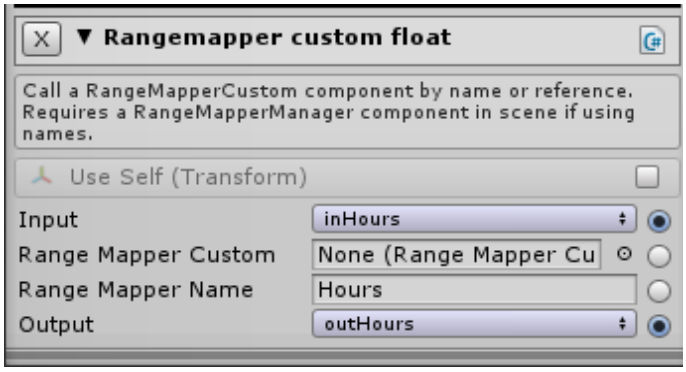


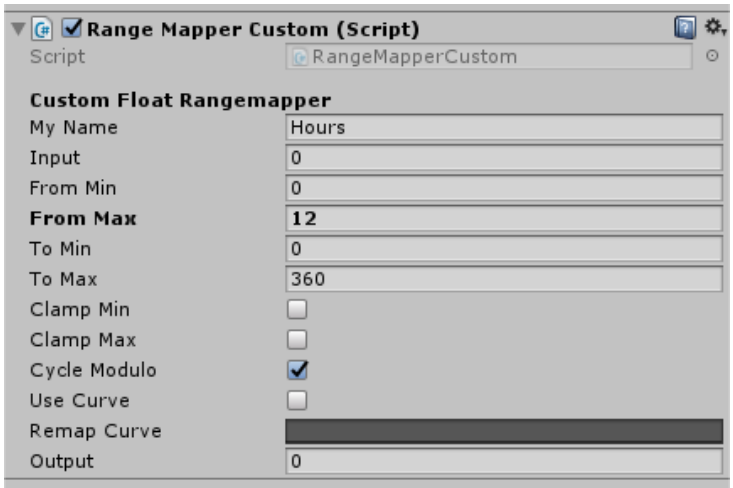
If a RangeMapperCustom has been assigned directly, the name in the RangeMapperName variable is NOT USED, but **the direct link is preferred**.

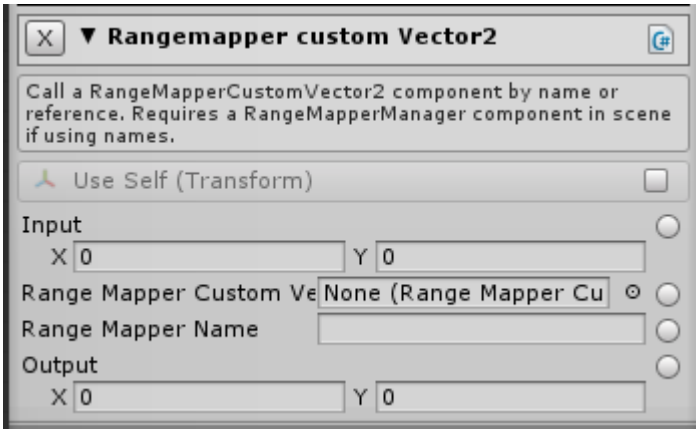
If no direct link is set up the name is used.

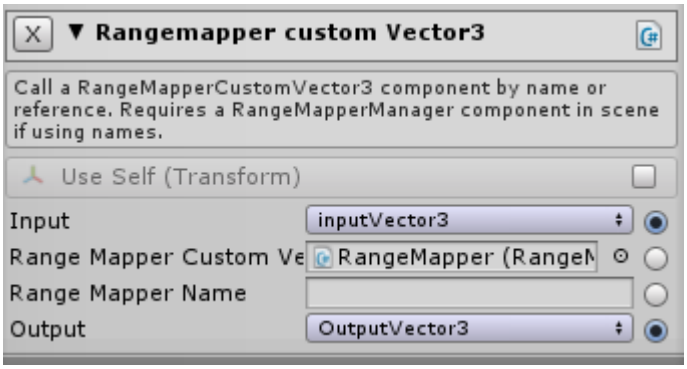
To call a RangeMapper by name you have to have the RangeMapperManager component installed on one GameObject in you scene (see Manual for details). Only one Manager is necessary. The manager needs no configuration.

Once the components are prepared simply call them with the action that fits the input type (float, vector2 or vector3). All you have to do is setup the input and output (and enter the name or direct link of the RangeMapperCustom component).









If you have questions, input or need support email me at:

support@wuenschonline.de

Have fun remapping,
Oliver Wuensch