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

MANUAL

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.

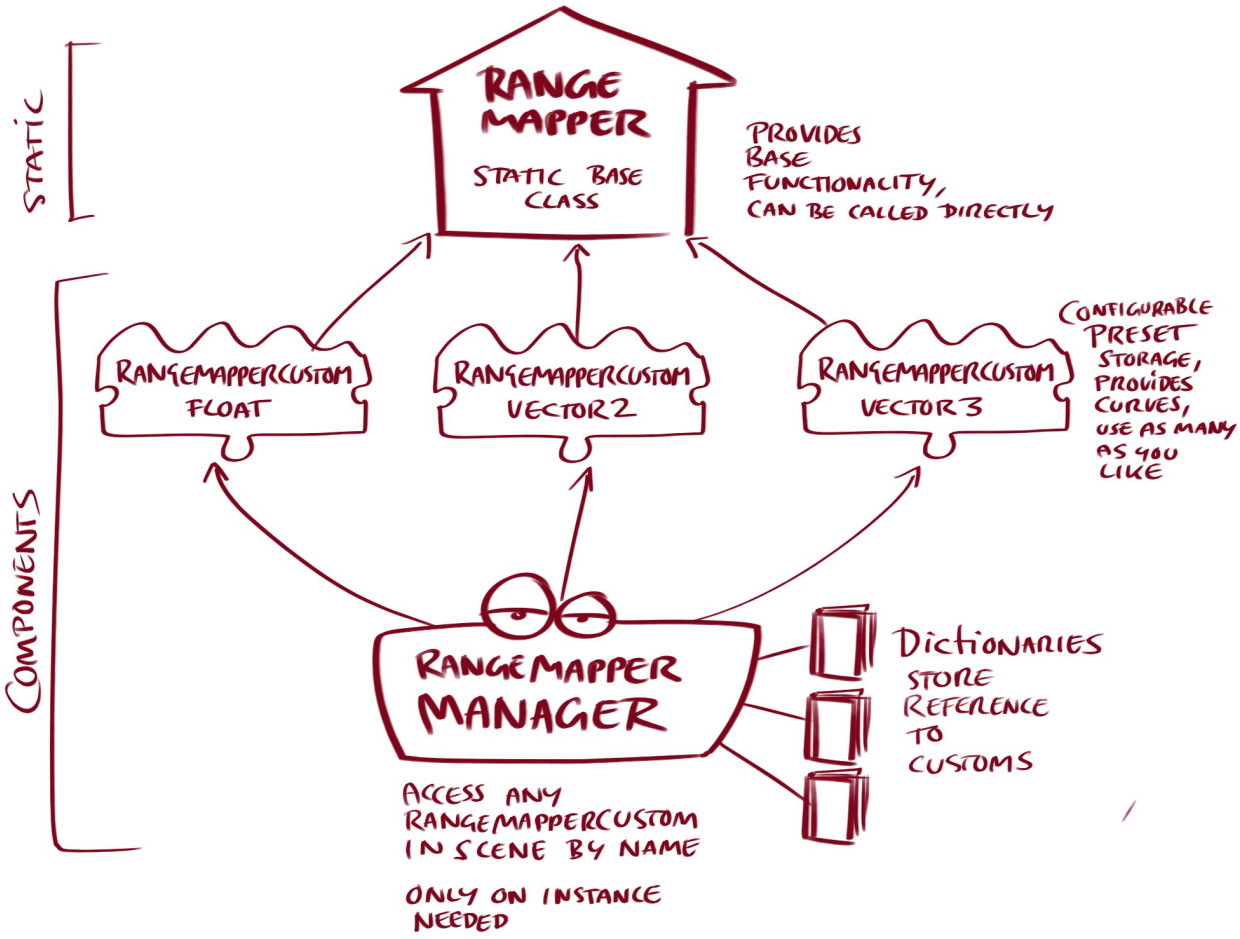
RangemapperCustom components can be used to create presets of Rangemappers stored in your scene so you you only need to pass the input to the correct RangeMapperCustom and you can reuse them whenever you need them in your scene.

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 Playmaker and Paradoxnotion Nodecanvas visual programming environments.

The architecture:



At the core there is a **static base class RangeMapper** that provides the basic functionality.
Since the functions are static you can **call** them in your scripts **without the need to instanciate RangeMapper** in your scene.

For more convenience **RangeMapperCustom components** can be attached to any GameObject in your scene to store a RangeMapper input configuration.
You can them comfortably reference these in your script and only need to pass the input to get back your mapped data.
You can give each RangeMapperCustom an **individual name**.

The **RangeMapperManager** component uses these names to provide an easy way to access any RangeMapperCustom in your scene by name (if the name is being used).

For a detailed description of all classes and public fuctions provided please take a look at the RangeMapperAPI documentation provided as PDF and HTML documentation.

The components explained:

The comfortable way to use RangeMapper is by using RangeMapperCustom components, but you can use it directly if you want. Skip to the next page if you want to dive into RangeMapperCustom directly and do not care about the direct access.

RangeMapper static base class:

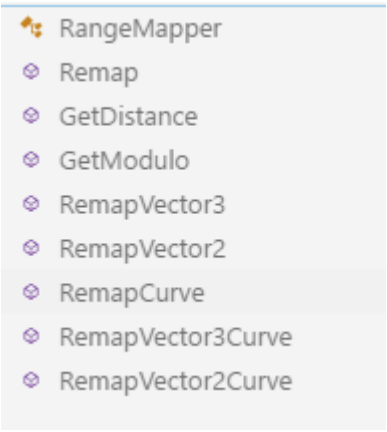
At the core of RangeMapper is the **static base class RangeMapper** provides all basic functionality and since the functions are static it can be used in your project without the need to instance it.
Simply add
using Wuensch;
to the top of your script to access RangeMappers namespace and then you can access any of these functions. For example with a call like (replace variables with your data, variables use is explained for the RangeMapperCustom component on the following pages):

```
float myResult = RangeMapper.Remap (input, fromMin, fromMax, toMin, toMax, clampMin, clampMax, cycleModulo);

to map an input value with an Animationcurve supply the variables and call:

float myResult = RangeMapper.RemapCurve (input, fromMin, fromMax, toMin, toMax, clampMin, clampMax, cycleModulo, remapCurve);
```

for **Vector2 and Vector3** similar functions exist, here is an **overview over the static functions** for details check out the comments in your IDE or check the C# scripts commentaries or the RangeMapperAPI documentation.



Remap, RemapVector2 and RemapVector3 Remap number spaces. RemapCurve (and Vector2 and Vector3 variants) additionally take an Animationcurve for remapping.

GetDistance returns the absolute distance between 2 numbers, GetModulo returns the Modulo value.

RangeMapperCustom components:

The RangeMapperCustom components are used to create custom RangeMappers in your scene so that you do not have to pass the many variables in your script any time you want to rangemap something.

There are 3 components , **RangeMapperCustom** is for floats,
RangeMapperCustomVector2 and **RangeMapperCustomVector3** are for Vectors.

They function similarly.

I will only explain the use of RangeMapperCustom here in detail.

Step1:

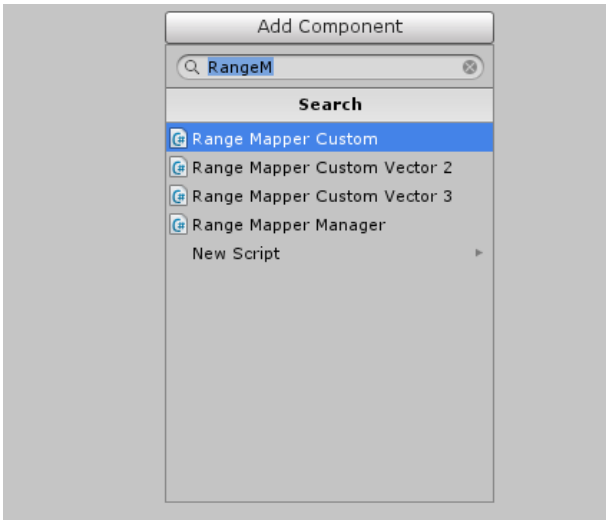
Select an empty GameObject in your scene.

Drag a RangeMapperCustom component from the Project Manager

(Assets->RangeMapper->script->RangeMapperCustom)

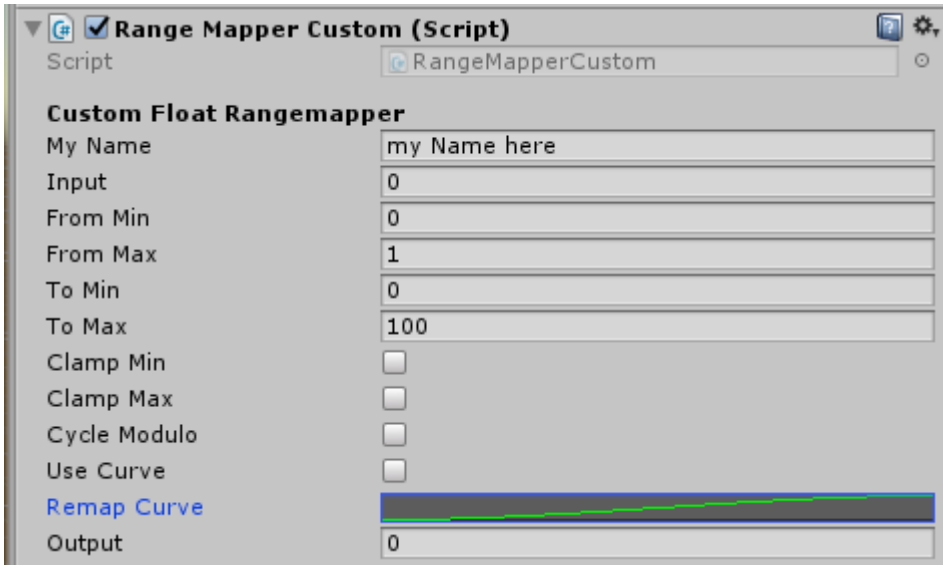
in Unity to the GameObject ,

or use the **add Component** button in inspector.



Step2:

You will see this component interface in Inspector:



Give the RangeMapperCustom a unique name (no other RangeMapperCustom should have the same name) by entering a string into the **myName** variable.

Next configure your custom RangeMapper.

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 into infinity.

ClampMin and clampMax:

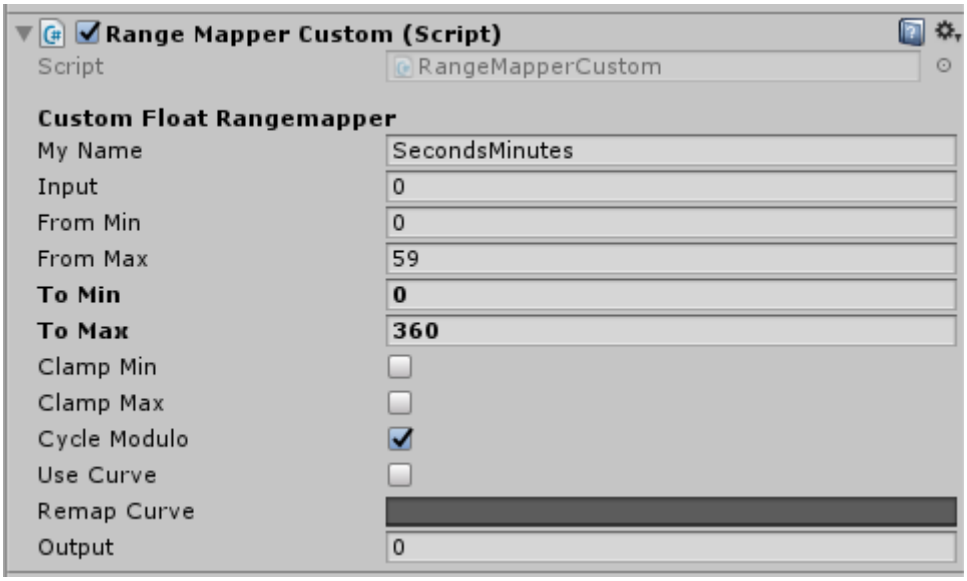
To limit the input so that the min or max are never exceeded simply activate the **clampMin** and/or **clampMax** boolean switches. If you clamp both and your fromMin/fromMax are 0f/1f an input of 2.4 will be clamped at the maximum of 1.0 and an input of -200.5f will be treated as if it were 0f.

CycleModulo:

For a clock for example: if you input a counter that counts the passed seconds (like you get in Unity from the Time.time function) you would want the RangeMapper for the hand of your clock's 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 further 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.



In the case of the demo example (actually the screenshot is wrong, sorry) the input range is 0 to 60, which means a difference of 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 correct 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.

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):

a public variable mirroring the output value returned by the RangeMapCustom function. Check this at Runtime in Unity editor to see what is being returned at the moment.

Accessing RangeMapperCustom in your script:

Each RangeMapperCustom component has a **RangemapCustom() function** that returns the custom RangeMappers result.

To use it, first enable access to the Wuensch namespace by adding

using Wuensch;

at the top part of your script.

In your create a reference to the RangeMapperCustom component (easiest way is to declare a public variable of type RangeMapperCustom or RangeMapperCustomVector2 or RangeMapperCustomVector2.

Then use the reference to call the function RangemapCustom() and pass the input like this:

float myResult=myRangeMapperReference.RangeMapCustom(inputValue);

This is an shortened example from the demo using a Vector3 custom RangeMapper:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using Wuensch;

  public class DemoCubeWobbler : MonoBehaviour {
    public RangeMapperCustomVector3 myRangeMapperVector3;
    public Slider mySlider;
    public GameObject myCube;
    private Vector3 tempVector = new Vector3 (0f, 0f,0f);
    /// use slider value to control cubes X and Y position via RangeMapperCustom curves

    void Update () {

      //construct a Vector3 to pass to the RangeMapper
      tempVector.x = mySlider.value;
      tempVector.y = mySlider.value;
      tempVector.z=0f;//Z position should not change, that is why the curve for z is
flat value 0 also
```

```
      //apply resulting vector to position of cube
      myCube.transform.position = myRangeMapperVector3.RangeMapCustom (tempVector);
    }
  }
```

The RangeMapperManager:

If you have several RangeMappeCustom components used in your scene and want an easy way to access them all through one central manager then RangeMapperManager would like to assist you.

Simply attach the component (only one needed in the scene) to any GameObject and it will automatically search the scene at Awake and collect all RangeMapperCustom in the scene to make them accessible by name.

The manager has the functions

```
public float RangemapCustom (float valueToMap, string rangeMapperCustomName)

public Vector3 RangemapCustomVector3 (Vector3 valueToMap, string rangeMapperCustomName)

public Vector2 RangemapCustomVector2 (Vector2 valueToMap, string rangeMapperCustomName)
```

to access the RangeMappers by name.

Again you have to build a reference to the manager in your script.

Since only one RangeMapperManager should be in the scene, the easiest way is to search it in Awake() and assign it to a public variable:

```
using Wuensch;
//...in your class:
  public RangeMapperManager myRangeMapperManager;

    // Use this for initialization
    void Awake () {
      myRangeMapperManager = Object.FindObjectOfType<RangeMapperManager> ();

    }
```

then you can simply call any of your RangeMappers by name like this:

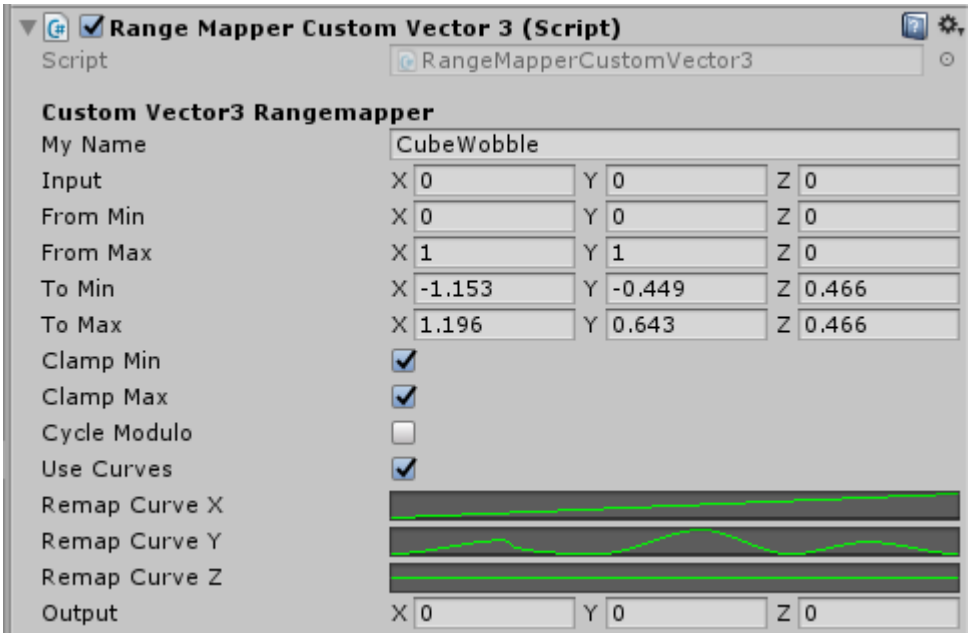
```
float result = myRangeMapperManager.RangemapCustom (inputFloat, "myName1");
```

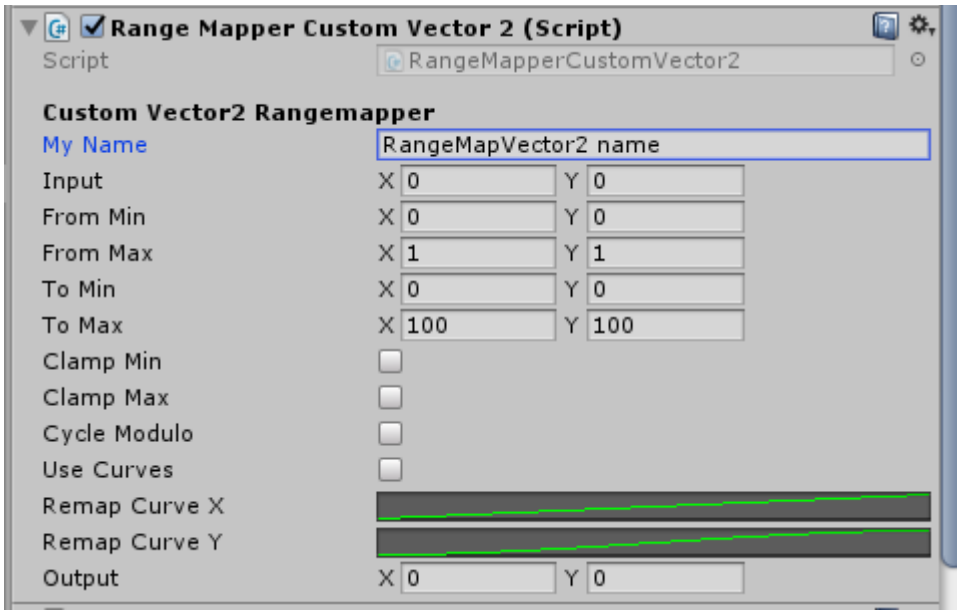
or for a Vector3 RangeMapper:

```
Vector3 result = myRangeMapperManager.RangemapCustomVector3 (inputVector3, "myName2");
```

Make sure the names for the RangeMappers are unique at least for each type of dictionary (float, vector2, vector3), as RangeMapperManager uses the first found. If the name does not exist, an Error is returned to Console and a value of 0f (or a Vector with 0f values) is returned.

The custom Vector RangeMappers are similar to the float one, only with Vector input and more curves.





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

support@wuenschonline.de

Have fun remapping,
Oliver Wuensch