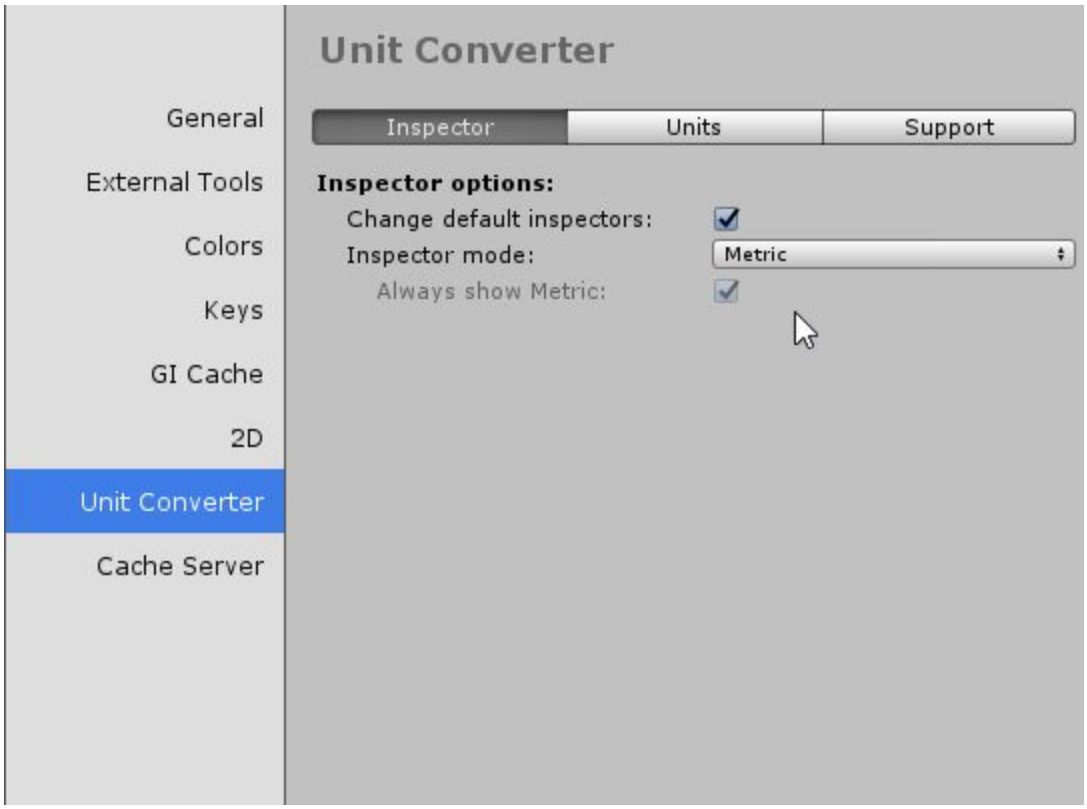
Unit Converter

Current Version 1.0
Last edit: 6/18/2018

The Unit Converter is a Unity editor extension that allows users to work with the Imperial system right into the editor itself. It will make small changes to the unity components like the Transform so you can use feet instead of meters.

Settings and options

Inside the unity editor, select the top menu ‘Edit’
Then select ‘Preferences’ this will open the unity preferences window, near the bottom on the menu you will see the option ‘Unit Converter’



Inspector options

Change default inspectors:

This will change the unity default inspectors so they will work with the imperial and Metric systems,
depending on what **inspector mode** you select.

Inspector mode:

If set to Metric, all the inspectors in the unity editor will show metric values, if set to the Imperial system the inspectors will show the imperial versions of it, feet instead of meters.
The asset will automatically convert your entered value to whatever Mode you selected.

Always show metric:

This option will will only unlock if you select the imperial system.
And will show both the feet and meter values on the inspectors.

Units

Here you can enable and disable other conversion options.
When a object is disabled, its also not compiled, resulting is reduced compiling times, choose only what you really need.

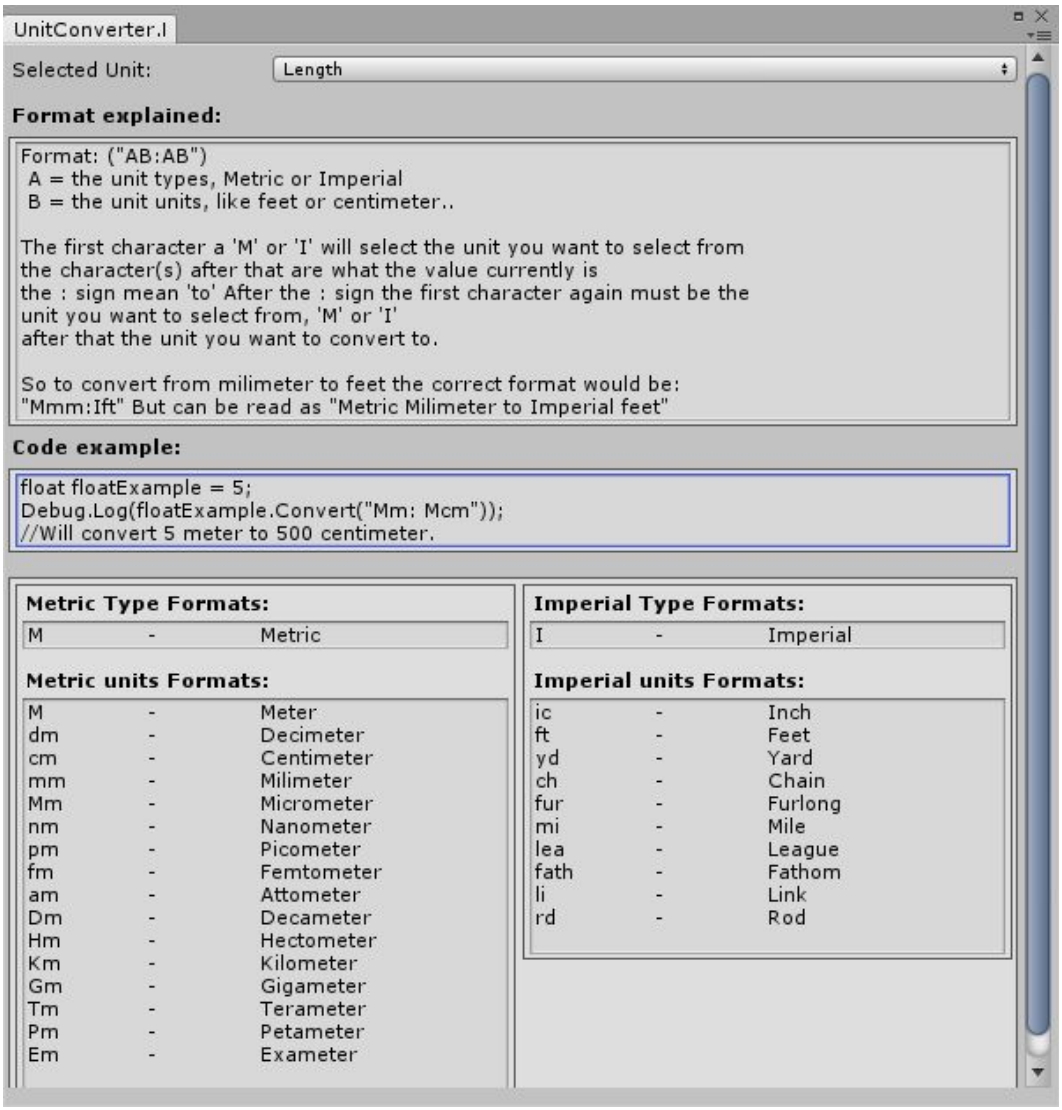
Support

Currently the support page gives you a small example about how to use it from code, there is also a button to the website, this manul and a button to open in short version of the manual inside the editor.

Format list

The format list is a small manual on how to use the asset from code. This format list can be found from the unity preferences window and also at the top ‘UnitConverter’ > ‘Format list’

This window will show you the short version on how to use the asset from code, at the top you can select the unit you want information about.



How to use the Unit Converter in c#

To use the Unit Converter from code might be a bit hard to understand at first, but once you get used to it is easy.

To fully understand how it works we will first have to explain it step by step

You can convert most value types, currently the supported value types are:

- Vector2
- Vector3
- Vector4
- Double
- Int
- Decimal
- UInt
- Long
- ULong
- Short
- UShort
- Float

To convert a value to another value you first have to know what the value currently is, for this example we will create a float value called jumpDistance.

```
float jumpDistance = 5f;
```

JumpDistance is currently in meters, but we want to convert it to feet, to do so we make a new float called jumpDistanceInFeet.

```
float jumpDistanceInFeet;
```

Now to convert the jumpDistance from meter to feet we need to convert it.
To convert a value we need to know what the value currently is, and what it needs to become, we know its value is in meters and we want it in feet.

We will use Convert() for this, The convert function can convert most units but you do have to tell it its format.

This is the format to convert Metric Meter to Imperial Feet:

```
jumpDistanceInFeet = jumpDistance.Convert(“Mm:Ift”);
```

(“Mm:Ift”)

The first character ‘M’ means Metric, We want to convert from the Metric system.

The second character ‘m’ means meter, as we want to convert from a meter.

The you have the ‘:’ sign this means ‘to’ as in you want ‘to’ convert it to.

After the ‘:’ sign we want to tell the function to what we want to convert to, so the first character after that is ‘I’ meaning Imperial, and the second character(s) ‘ft’ meaning feet, means we want to convert to a (‘I’) Imperial, (‘ft’) Feet.

So (“Mm:Ift”) can be read as (“**M**etric **m**eter To **I**mperial **f**ee**t**”)

So to come back to the jumpDistance example:

```
Float jumpDistance = 5f;
Float jumpDistanceInFeet;

jumpDistanceInFeet = jumpDistance.Convert(“Mm:Ift”);

//jumpDistanceInFeet will now be 16.4042 feet.
```

But we can also convert it back again, like this:

```
jumpDistance = jumpDistanceInFeet .Convert(“Ift:Mm”);
```

(“Ift:Mm”)

Again this can be read as (“**I**mperial **f**ee**t** To **M**etric **m**eter”)

But you can also convert from Metric to Metric and Imperial to Imperial, as every combination is possible!
Currently version 1.0 has 676 combinations that you can use.

The full format list

(case sensitive)

Length Metric

m	Meter
dm	Decimeter
cm	Centimeter
mm	Millimeter
Mm	Micrometer
nm	Nanometer
pm	Picometer
fm	Femtometer
am	Attometer
Dm	Decameter
Hm	Hectometer
Km	Kilometer
Gm	Gigameter
Tm	Terameter
Pm	Petameter
Em	Exameter

Length Imperial

ic	Inch
ft	Feet
yd	Yard
ch	Chain
fur	Furlong
mi	Mile
lea	League
fath	Fathom
li	Link
rd	Rod

Hint:

You can also bypass the formating way of going stuff by

```
Using RealisticPhysics.UnitConverter.Metric;
Using RealisticPhysics.UnitConverter.Imperial;
```

Allowing you to call the functions directly like:

```
Float jumpDistance.MeterToCentimeter();
```

However you can not convert from meter to imperial and from imperial to meter like this.

Also don’t forget!

If you found any bugs or conversions that are not correct
Please contact us so we can fix it right away!

If you are missing a conversion, contact us and we will try to add it with the next update.

And please take your time to leave a review and just a few stars,
They really help us a lot!

You can contact us at:

http://realisticphysics.com/contact