DynaRes

Dynamic resolution for Unity3D

Documentation / **Bit2Good** // Feb. 17th 2016

Table of contents

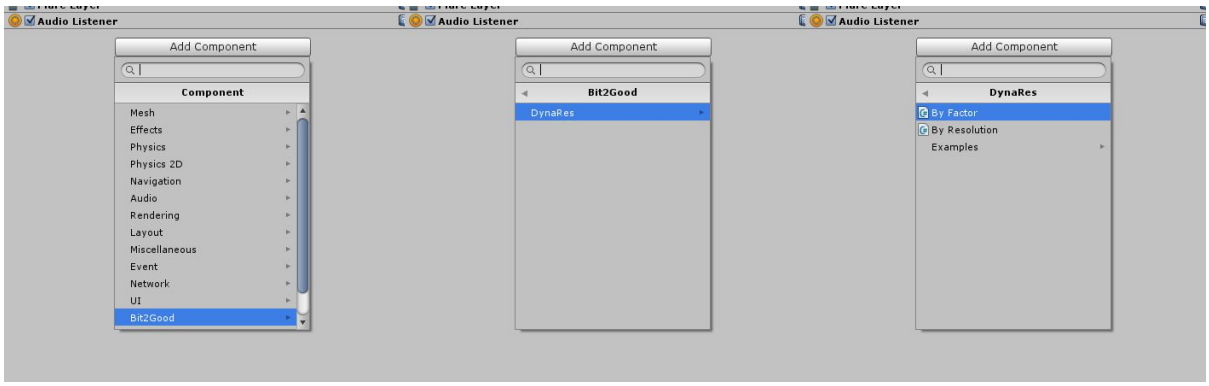
What is DynaRes?

At its core, DynaRes is a class derived from MonoBehaviour that interfaces with some of Unity's render callbacks to scale the renderbuffer before stuff gets rendered on it and scales it back to screen size before it's displayed.

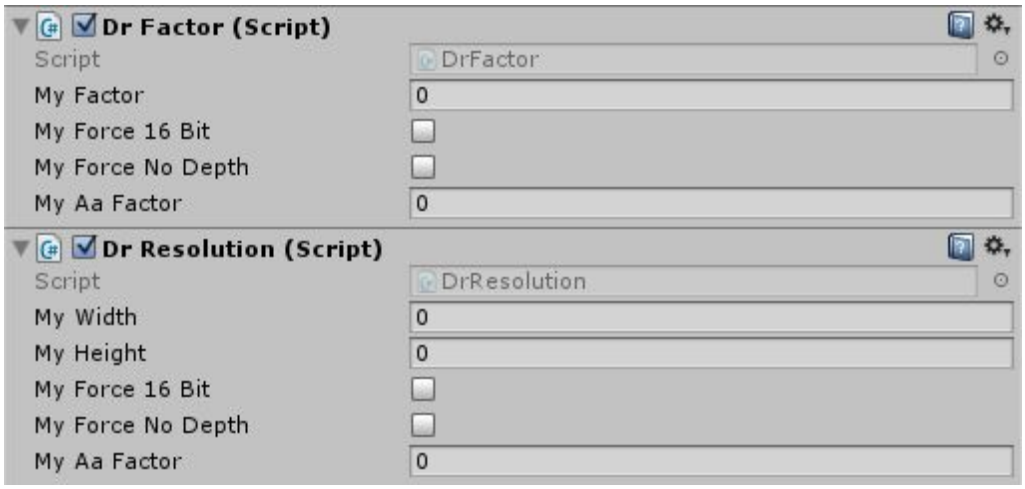
What this gives you is simply a way to sub- and supersample rendering on the fly at any time which can be put to use for different effects. The DynaRes package contains the base class, some easy to use scripts, some examples and this documentation.

Quick use

The easiest way to use DynaRes are the premade scripts that can be applied to any camera.



Those scripts have easy to use inspector interfaces which should give you a very quick and easy setup to use without getting into the DynaRes class at all.



Examples

If you want to harness the full feature set of DynaRes, you will need to familiarize yourself with the DynaRes class and the ways you can interface with it. There are 2 reasonable ways to use the DynaRes class:

1. *Call DynaRes functions from an instance*
   For this you'll first need to add the DynaRez base class to your camera; you can then call DynaRes functions from your script by reference to the base class instance. The `FpsWatchdog` and `PixelationAnimation` example scripts will help you to understand how to do this.
2. *Create a new class that inherits from DynaRez*
   This is as easy as creating a new script and swap `MonoBehaviour` with `DynaRes`. The `SoftwareFov` example script will help you to understand this concept.

All example scripts are commented and should be easy enough to understand :-)

API

```
camera      GetCam          ()
```

`GetCam` will return a reference to the camera that DynaRes is working on.

```
void        SetAA           (int aA)
```

`SetAA` takes an integer value and will set the anti aliasing value specified by http://docs.unity3d.com/ScriptReference/RenderTexture-antiAliasing.html - any value other than 1, 2, 4 or 8 will be changed to the next smaller value. DynaRes will default to a value of 1 which means no anti aliasing.

```
void        SetDBuffer      (int bSize)
```

`SetDBuffer` takes an integer value and will set the depth buffer size in bits. As specified by http://docs.unity3d.com/ScriptReference/RenderTexture-depth.html only 0, 16, 24 are supported and only the 24 bit depth buffer offers stencil support. DynaRes will default to a 24 bit depth buffer for maximum compatibility with other scripts and plugins.

```
void        SetFactor       (float sFactor)
```

`SetFactor` is one of two ways to manipulate the internal resolution and takes a float value that represents the scale factor. A value of 1.0f means 100% resolution which is the default value and means the internal resolution will match the actual screen size. A value of 0.5 will mean 50% etc. It is worth noticing that `sFactor` will be clamped between 0.01 and 4.0 meaning the internal resolution will always be between 1% and 400% of the actual screen size.

```
void        SetRes          (int sWidth, int sHeight)
```

`SetRes` takes two integer values and will set the internal resolution to meet width and height as specified by the `sWidth` and `sHeight` values. It is worth noticing that this method always keeps the screen ratio, even if you `sWidth` by `sHeight` doesn't match the screen ratio.

```
void        SetRes          (int sWidth, sHeight, bool forceRes)
```

This variant of `SetRes` takes an additional boolean variable `forceRes`; if `forceRes` is false then `SetRes` will act just like declared without `forceRes`. If set to true however, the exact specified resolution will be enforced. This can result in distorted views but may be useful for interesting effects.

```
void       SetTFormat      (RenderTextureFormat tFormat)
```

`SetTFormat` can be used to specify the texture format used by DynaRes' render texture as specified by http://docs.unity3d.com/ScriptReference/RenderTextureFormat.html - when possible, DynaRes will default to `RenderTextureFormat.DefaultHDR` or `RenderTextureFormat.Default` if HDR is not available.

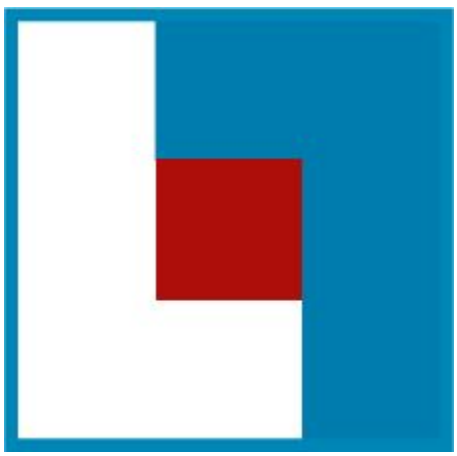
Contact

DynaRes was built as an inhouse solution because we wanted to know if dynamic resolution would be possible with Unity3D. Therefore it contains mainly methods we at **Bit2Good** want and need. It's also explained in the same way we would expect a similar plugin to be explained.

So if you would like to see some additional functionality, have any questions or just want to drop us a message, feel free to contact us via sales@bit2good.com. You can also find us on Twitter `@Bit2Good` and also on Facebook.

Thank you for supporting us! :-)

Sebastian Erben
Technical and creative director / **Bit2Good** // Feb 2016
`@TheBossti`



Bit2Good