RETRONATOR

Documentation v1.4

Content

About this Document

This is the official documentation for the Unity Asset "Retronator", a script that enables you to pixelize and colorize your Game in an easy and highly efficient way!

- ➢ If you did not purchase this Asset on the Unity Asset Store or directly from the Author, this is an illegal copy and you are legally not allowed to use it. Inform the author immediately about where you received this copy! Contact information is available at http://www.garvin-gurbat.de

About Retronator

Retro Style Creation

Retronator is an easy to use high performance filter for rendering the camera view in a custom classic resolution retro style. Make your game feel like in good old times! Back to the roots!

Ease of Use

Retronator is designed to be as easy as possible. Integration is done by simple drag and drop and all settings can be made in the inspector. There are previews for the transitions and access at runtime is made very easy since all methods are static and available in any script.

High Performance

Retronator is designed with rendering performance in mind. You will definitely not recognize any performance loss – actually your framerate will be even increased in big or complex scenes!

Feature Richness

There are many options for any use case and fades (transitions) between pixelicious and native resolutions are included.

Integrating Retronator into your Project

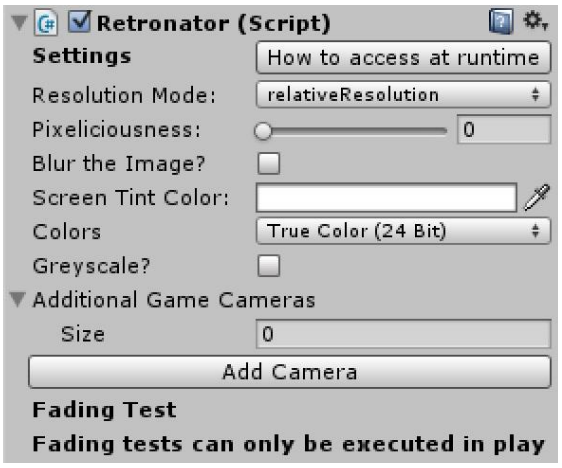
To use Retronator you simply drag and drop the Script "**Retronator.cs**" to your **camera**.

- ✓ That's it! You're ready to use Retronator.

The Inspector Window

Settings

You can find the Settings in the **Retronator component** of your **camera**.



Retronator (Script)
Settings — How to access at runtime
Resolution Mode: relativeResolution
Pixeliciousness: 0
Blur the Image? ☐
Screen Tint Color:
Colors: True Color (24 Bit)
Greyscale? ☐
Additional Game Cameras
Size: 0
Add Camera
Fading Test
Fading tests can only be executed in play

Here you can set the Retronator up to your needs.

➢ You will only see the effect in your **game view**! It is no longer necessary to enter play mode since Retronator 1.3!.

How to access at runtime

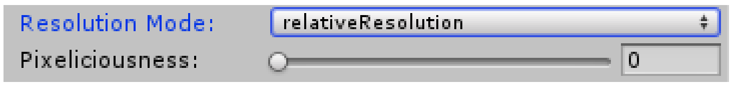
Clicking this button will pop up a dialogue windows giving you some information about how to access the settings and fades from within your scripts. For more detailed information see "Accessing from Code".

Resolution Mode

You have five options described in the following.

Relative Resolution Mode

The option "relativeResolution" allows you to simply adjust the pixeliciousness of your camera view using a slider or the input field next to it.



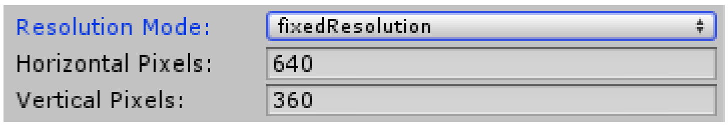
Resolution Mode: relativeResolution
Pixeliciousness: 0

✓ Any float value from 0f (inclusive) to 100f (inclusive) is valid.
0 means that the image will not be pixelized. The native resolution will be rendered.
100 means maximum pixelicousness and you will only see a few pixels on the screen.

➢ Values smaller than 0 or greater than 100 are not allowed!

Fixed Resolution Mode

The "fixedResolution" mode allows you to enter a width and height. Your camera view will be rendered in exactly these dimensions.

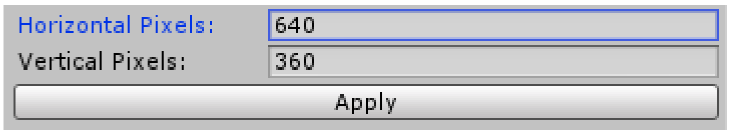


Resolution Mode: fixedResolution
Horizontal Pixels: 640
Vertical Pixels: 360

✓ All values divisible by two (even numbers) are valid! The values should not be greater than the games native resolution.

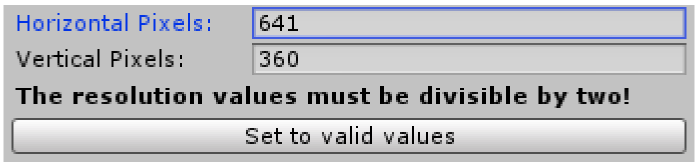
- ➢ Uneven numbers are not allowed!

As you enter values, these are checked for validity. If the values are ok, you can accept them using the “Apply” button.


Horizontal Pixels: 640
Vertical Pixels: 360

- ➢ The values must be confirmed by clicking the “Apply” button. Otherwise they will be ignored!

If you entered invalid values, you will be informed by a message and have to correct them. You can do this either manually by entering new values or automatically by clicking the “Set to valid values” button. Don’t forget to apply your changes afterwards.


Horizontal Pixels: 641
Vertical Pixels: 360
The resolution values must be divisible by two!

Fixed Width Mode
The “fixedWidth” mode allows you to enter a width for rendering the camera view. The vertical resolution (height) will be calculated automatically depending on the aspect ratio.

- ➢ The value must be valid as described in “Fixed Resolution Mode” and confirmed as shown before.

Fixed Height Mode
The “fixedHeight” mode allows you to enter a height for rendering the camera view. The horizontal resolution (width) will be calculated automatically depending on the aspect ratio.

- ➢ The value must be valid as described in “Fixed Resolution Mode” and confirmed as shown before.

Fixed Max Mode
The “fixedMax” mode allows you to enter a resolution for the larger dimension. The filter will check the current aspect ratio and set the higher dimensions resolution to the entered value. In common cases this means, that in landscape resolutions the width will be set to the given value and in portrait resolutions the height will be set to it. The other dimension (height in landscape and width in portrait) will be calculated depending on the aspect ratio.

- ➢ The value must be valid as described in “Fixed Resolution Mode” and have to be validated and confirmed as shown before.

Blur the Image
This setting allows you to blur the rendered image. This gives the effect some kind of a fuzzy look.

Check the checkbox to enable blurring. Uncheck it to get square edge pixels.

Screen Tint Color
This option allows you to pick a color to dye the rendered image.

- ➢ Do not change the alpha channel of the tint color since this will lead to probably unwanted effects.

Colors

This option allows to adjust the number of colors that are used to render he scene. This can help you to enhance the retro feeling of your game. Available options are:

True Color (24 Bit)

The image will be rendered with 16.77 million colors.

32k Colors (15 Bit)

The color palette will be limited to 32,000 colors.

4k Colors (12 Bit)

Colors will be adjusted to 4,000 allowed colors.

512 Colors (9 Bit)

The image will be recalculated to 512 colors.

64 Colors (6 Bit)

Only 64 colors will be used to render the image.

8 Colors (3 Bit)

The scene will be displayed very minimalistic using only 8 colors.

➢ Colors will control the number of grey shades, if Greyscale is active.

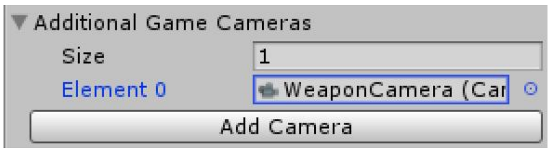
Greyscale?

By activating this checkbox you can render the scene in greyscale (black and white).

➢ The amount of available grey shades is controlled by the „Colors” option.

Additional Game Cameras

If you are using additional game cameras (cameras, which display the scene but not user interface elements), add them to this list. The main camera which holds the script should not be added to this list.



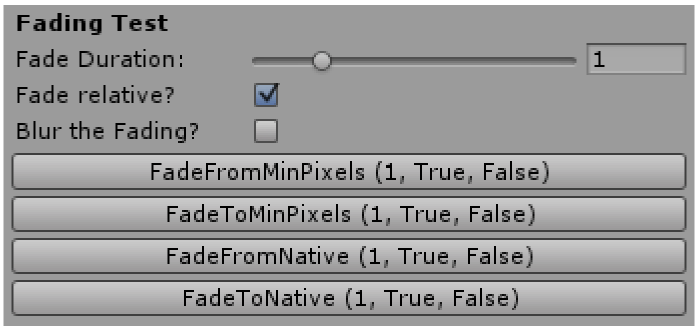
Additional Game Cameras
Size 1
Element 0 WeaponCamera (Car
Add Camera

Clicking the button "Add Camera" adds another slot to the list. If you do not see the list, click on the little triangle left to the text "Additional Game Cameras". You can add a camera by if you place the game object with the camera from the scene into one slot via drag and drop or by clicking the little circle next to the slot and selecting the camera.

You can remove a camera from the list by clicking on the entry with the right mouse button and choosing "Delete Array Element".

Fading Test

The fading test area in the inspector can be used to test some fading effects that are supported by Retronator.



Fading Test

Fade Duration: 1

Fade relative? ☑

Blur the Fading? ☐

FadeFromMinPixels (1, True, False)

FadeToMinPixels (1, True, False)

FadeFromNative (1, True, False)

FadeToNative (1, True, False)

➢ The fading tests are only present in **play mode**!

Fade Duration

Adjust this value to set the time in seconds that a fade will need to be completed.

Fade relative

If this checkbox is checked, the fades will take the settings into account. See “Fade Buttons” for detailed information.

Blur the Fading

If this checkbox is enabled, blurring will be applied during the fade effect, even if “Blur the Image” is not checked. After the fade the blur status will be set according to “Blur the Image”.

Fade Buttons

There are four buttons that allow you to test the fades.

The resolution, the fade starts and ends at, depends on “Fade relative”. The following table visualized the start and end resolutions depending on the button and “Fade relative.

Button	Fade relative	Start resolution	End resolution
FadeFromMinPixels	☐	■	░
FadeFromMinPixels	☑	■	▦
FadeToMinPixels	☐	░	■
FadeToMinPixels	☑	▦	■
FadeFromNative	☐	░	■
FadeFromNative	☑	░	▦
FadeToNative	☐	■	░
FadeToNative	☑	▦	░

░ = native resolution

▦ = pixelized resolution according to the settings

■ = maximum pixelation (only few, big pixels)

After testing a fade, you can click the “ResetFade()” button to reset the pixelation to the settings of the Retronator.

Accessing from Code

Accessing the setting and methods of Retronator from your own scripts is very easy and gives you the ability to adjust the effect to your needs and to perform fades (transitions) at any time.

- Access from scripts is only available, if the "Retronator.cs" script has been assigned to the camera before!

Using the class

You have to call all functions using the class name as prefix. For example calling the method "SetBlur()" would be done using "Retronator.SetBlur()". The methods can be called from any script in the scene!

Methods for Settings

SetPixeliciousness

```
void SetPixeliciousness(float rate, bool setMode = true)
```

Set the "Pixeliciousness" and optionally set "Resolution Mode" to "relativeResolution".

GetPixeliciousness

```
float GetPixeliciousness()
```

Get current "Pixeliciousness".

SetFixedResolution

```
void SetFixedResolution(int width, int height, bool setMode = true)
```

Set the "Horizontal Pixels" and "Vertical Pixels" values of „fixedResolution" and optionally set "Resolution Mode" to „fixedResolution".

GetFixedResolution

```
Vector2 GetFixedResolution()
```

Get current „Horizontal Pixels" and „Vertical Pixels" of „fixedResolution".

SetFixedWidth

```
void SetFixedWidth(int width, bool setMode = true)
```

Set the "Horizontal Pixels" of "fixedWidth" and optionally set "Resolution Mode" to „fixedWidth".

GetFixedWidth

```
int GetFixedWidth()
```

Get current „Horizontal Pixels" of „fixedWidth".

SetFixedHeight

```
void SetFixedHeight(int height, bool setMode = true)
```

Set the "Vertical Pixels" of „fixedHeight" and optionally set "Resolution Mode" to "fixedHeight".

GetFixedHeight

```
int GetFixedHeight()
```

Get current „Vertical Pixels" of „fixedHeight".

SetFixedMax

```
void SetFixedMax(int maximum, bool setMode = true)
```

Set the "Maximum Pixels" of „fixedMax" and optionally set "Resolution Mode" to "fixedMax".

GetFixedMax

```
int GetFixedMax()
```

Get current „Maximum Pixels" of „fixedMax".

SetMode

`void SetMode(ResolutionMode mode)`

Set the „Resolution Mode” to “relativeResolution”, “fixedResolution”, “fixedWidth”, “fixedHeight” or “fixedMax”. You have to use `Retronator.ResolutionMode.relativeResolution`, `Retronator.ResolutionMode.fixedResolution`, `Retronator.ResolutionMode.fixedWidth`, `Retronator.ResolutionMode.fixedHeight` or `Retronator.ResolutionMode.fixedMax` for the “mode” parameter.

GetMode

`ResolutionMode GetMode()`

Get current „Resolution Mode”.

SetTintColor

`void SetTintColor(Color color)`

Set the „Tint Color“, to dye the camera view.

GetTintColor

`Color GetTintColor()`

Get current „Tint Color”.

SetBlur

`void SetBlur(bool blur)`

Activates or deactivates „Blur the image”.

GetBlur

`bool GetBlur()`

Get current state of „Blur the image”.

SetGreyscale

`void SetGreyscale(bool active)`

Activates or deactivates greyscale rendering.

GetGreyscale

`bool GetGreyscale()`

Get current state of „Greyscale“.

SetColorDepth

`bool SetColorDepth(int depth)`

Set bit depth of the used color palette.
Valid values are: 24, 15, 12, 9, 6 and 3 (see Colors).

GetColorDepth

`int GetColorDepth()`

Get the current bit depth of the used color palette (see Colors).

AddGameCamera

`bool AddGameCamera(Camera camera)`

Adds a camera to the “Additional Game Cameras” list (see Additional Game Cameras).
Returns *true* on success. If *false* the camera could not be added to the list or was already present in the list.

RemoveGameCamera

```
bool RemoveGameCamera(Camera camera)
```

Removes the given camera from the list of “Additional Game Cameras” (siehe Additional Game Cameras).
Returns *true* on success. If *false* the camera could not be removed from the list or was not present in the list.

Methods for Fading

- ➢ These methods are used to fire a fading animation (transition). All parameters are optional. The exact way the parameters work is described in “

Colors

This option allows to adjust the number of colors that are used to render he scene. This can help you to enhance the retro feeling of your game. Available options are:

True Color (24 Bit)
The image will be rendered with 16.77 million colors.

32k Colors (15 Bit)
The color palette will be limited to 32,000 colors.

4k Colors (12 Bit)
Colors will be adjusted to 4,000 allowed colors.

512 Colors (9 Bit)
The image will be recalculated to 512 colors.

64 Colors (6 Bit)
Only 64 colors will be used to render the image.

8 Colors (3 Bit)
The scene will be displayed very minimalistic using only 8 colors.

- ➢ Colors will control the number of grey shades, if Greyscale is active.

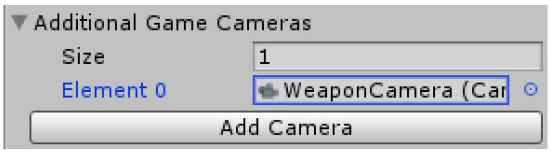
Greyscale?

By activating this checkbox you can render the scene in greyscale (black and white).

- ➢ The amount of available grey shades is controlled by the „Colors” option.

Additional Game Cameras

If you are using additional game cameras (cameras, which display the scene but not user interface elements), add them to this list. The main camera which holds the script should not be added to this list.



Clicking the button “Add Camera“ adds another slot to the list. If you do not see the list, click on the little triangle left to the text “Additional Game Cameras“. You can add a camera by if you place the game object with the camera from the scene into one slot via drag and drop or by clicking the little circle next to the slot and selecting the camera.

You can remove a camera from the list by clicking on the entry with the right mouse button and choosing “Delete Array Element”.

➢ Fading Test".

"durationInSeconds" is the "Fade Duration".
"toPixeliciousness" and "fromPixeliciousness" is the "Fade relative" flag.
"blur" is the "Blur the Fading" flag.

FadeFromMinPixels

```
void FadeFromMinPixels(float durationInSeconds = 1f, bool toPixeliciousness = true,
bool blur = false)
```

FadeFromNative

```
void FadeFromNative(float durationInSeconds = 1f, bool toPixeliciousness = true, bool
blur = false)
```

FadeToMinPixels

```
void FadeToMinPixels(float durationInSeconds = 1f, bool fromPixeliciousness = true,
bool blur = false)
```

FadeToNative

```
void FadeToNative(float durationInSeconds = 1f, bool fromPixeliciousness = true, bool
blur = false)
```

Compatibility with other Plugins

General

Using several plugins at the same time may cause several problems.
If you are facing problems using Retronator along with other plugin, I will do my best to assist you in solving them. You can find contact information on http://www.garvin-gurbat.de.

UFPS (Ultimate FPS)

UFPS is an awesome first person shooter plugin which uses multiple cameras. You can find it here on the Asset Store: https://www.assetstore.unity3d.com/en/#!/content/2943

Since the plugin uses multiple cameras, Retronator must be configured correctly to avoid problems. For instance it is possible, that the weapon is not displayed in game. To avoid this and other errors, you have to attach the Retronator script to the correct camera. This is the „FPSCamera". Additionally it is essential that the WeaponCamera is added to the list of „Additional Game Cameras". The names of the two cameras are according to those in the UFPS demo scene. You can find a detailed explanation of „Additional Game Cameras" here.

At this point I would like to thank the developers of UFPS for the great assistance in optimizing compatibility of our plugins!

Troubleshooting

If you have any problem using Retronator, you may find the solution right here.
For more information and direct contact information visit http://www.garvin-gurbat.de.

Q: Why does Retronator not pixelize the image?
A: There are multiple possible reasons for this:

- You did not add the script to your camera
- The Retronator settings are set to no or a very low pixilation effect
- The Retronator component is deactivated

Q: **Why is my User Interface not retronated?**

A: Retronator does not affect ingame UI if its canvas’ render mode is set to “Screen Space - Overlay”. If you want to retronate your UI, you have to set its canvas’ render mode to “Screen Space - Camera” or “World Space” and set the camera containing Retronator as the canvas’ render camera. Retronator is not designed to affect UI elements. So it may be a bit tricky to tweak the correct values for your UI to be affected by Retronator. But it is possible.

Q: **Why do some objects disappear when I start the game?**

A: You are probably using multiple cameras to render the scene. All cameras (except the one, that the script is attached to and UI cameras) must be added to the “Additional Game Cameras” list.