DTT
APPS WEB GAMES

SINCE 2010

Processing Effects

Documentation | 28-10-2022

Table of Contents

1. Get started quickly 3
    Shader Dependencies 4

2. Introduction 5

3. Set-Up 7
    Required steps 7
    Shader 8

4. Editor 10

5. API 11
    Posterize Handler 11
    Pixelate Handler 11
    Grayscale Handler 12
    Resize Handler 12

6. Known Limitations 13

7. Support and feedback 14

1. Get started quickly

Get started quickly to get a quick preview of the Processing Effects packages' capability.

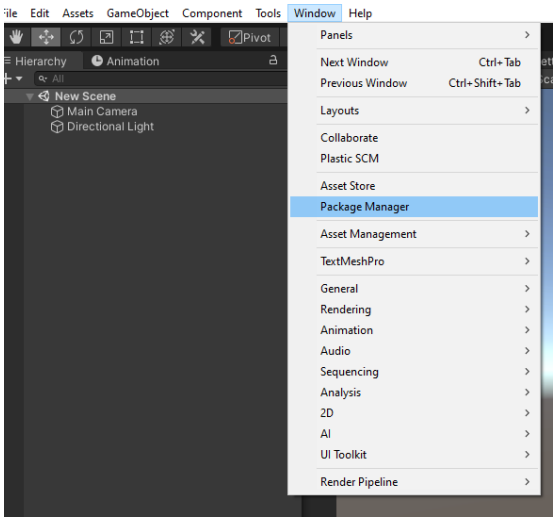
1. Before converting a texture make sure read/write is enabled and the format is set to RGB.
2. Call the Desired modification on the corresponding Handler:
    a. PosterizeHandler
    b. PixelateHandler
    c. GrayscaleHandler
    d. ResizeHandler

```
Texture2D posterizeTexture = PosterizeHandler.Posterize(_texture, _powerBase, _exponent);
Texture2D pixelateTexture = PixelateHandler.Pixelate(_texture, _pixelCount);
Texture2D grayscaleTexture = GrayscaleHandler.AddGrayscale(_texture, _grayscale);
Texture2D resizeTexture = ResizeHandler.Resize(_texture, _newResolution);
```

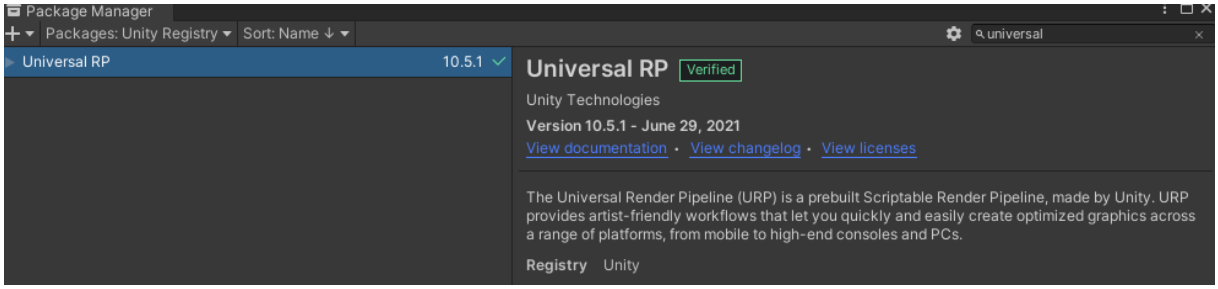
3. Go to **Tools > DTT > Processing Effects > Texture Effects Window** for the editor window which previews the effects the asset offers.
4. This package also contains shaders for posterization and pixelation. An example of the shaders can be found inside of the packages' demo folder (Packages > Processing Effects > Demo > Scenes).
5. For the shaders to work, it is required to install and set up the following dependencies: (**The process behind installing the dependencies is described on the next page.)**
    a. Either the Universal Render Pipeline (**URP**) or High Definition Render Pipeline (**HDRP**), depending on the desired render pipeline.

Shader Dependencies

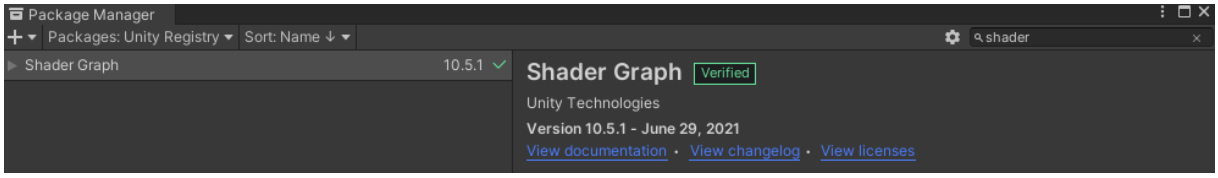
1. Open the **Unity Package Manager**: Window > Package Manager.



2. In the dropdown next to the **+**, select **Unity Registry** and search for **Universal RP**.

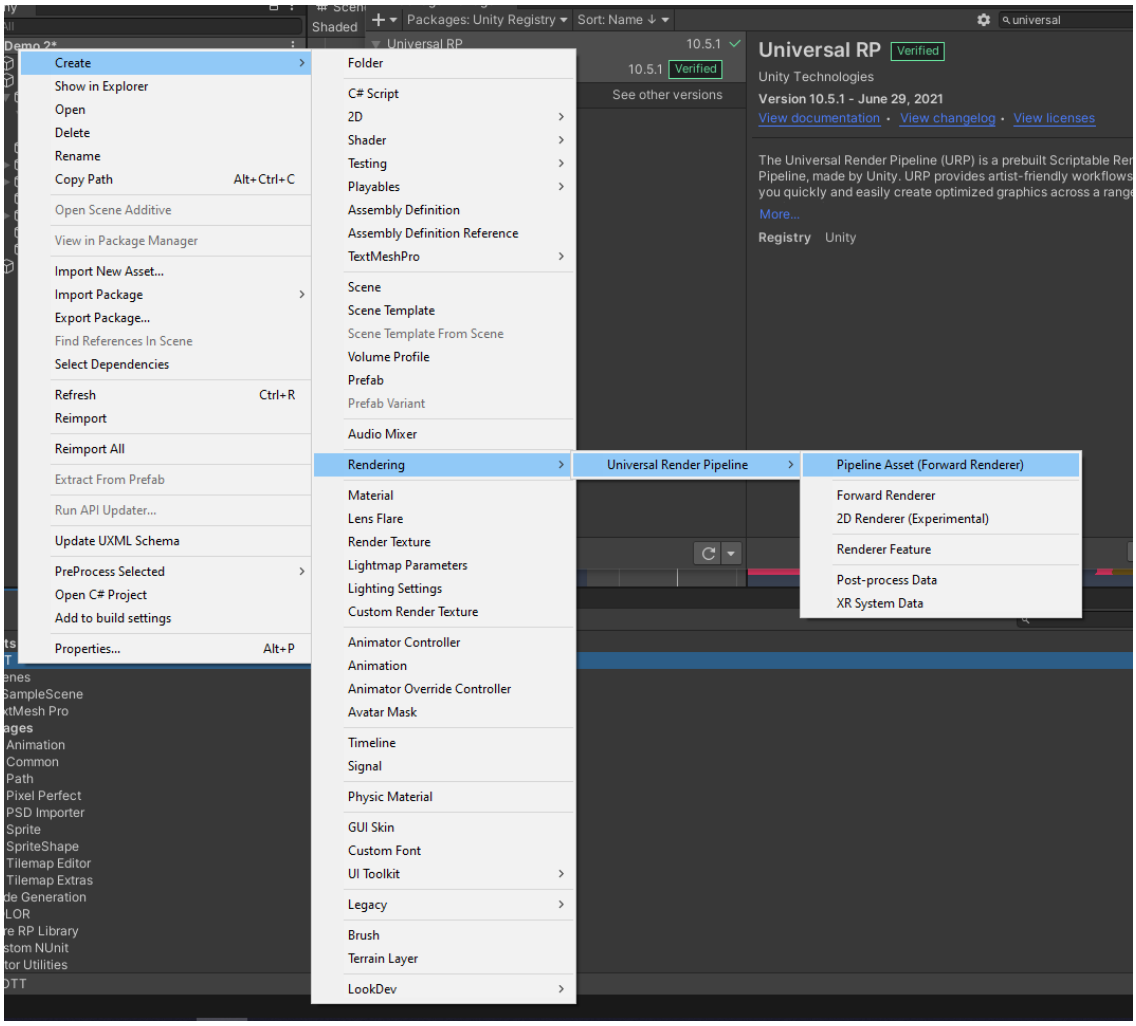


For the shader graph, search **Shader Graph**.

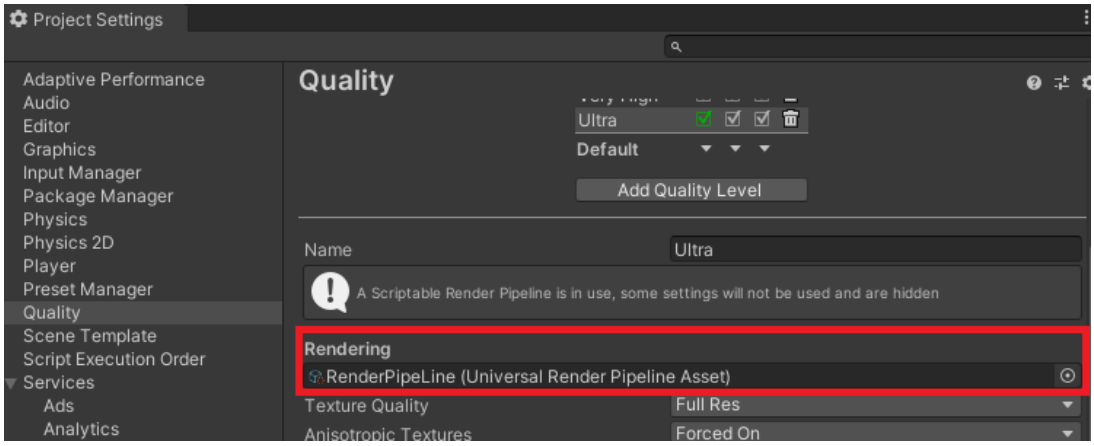


3. Click the install button in the bottom right corner to install the package.

4. After installing the Universal Render Pipeline. Create the option by going to (Create > Rendering > Universal Render Pipeline > Pipeline Asset).

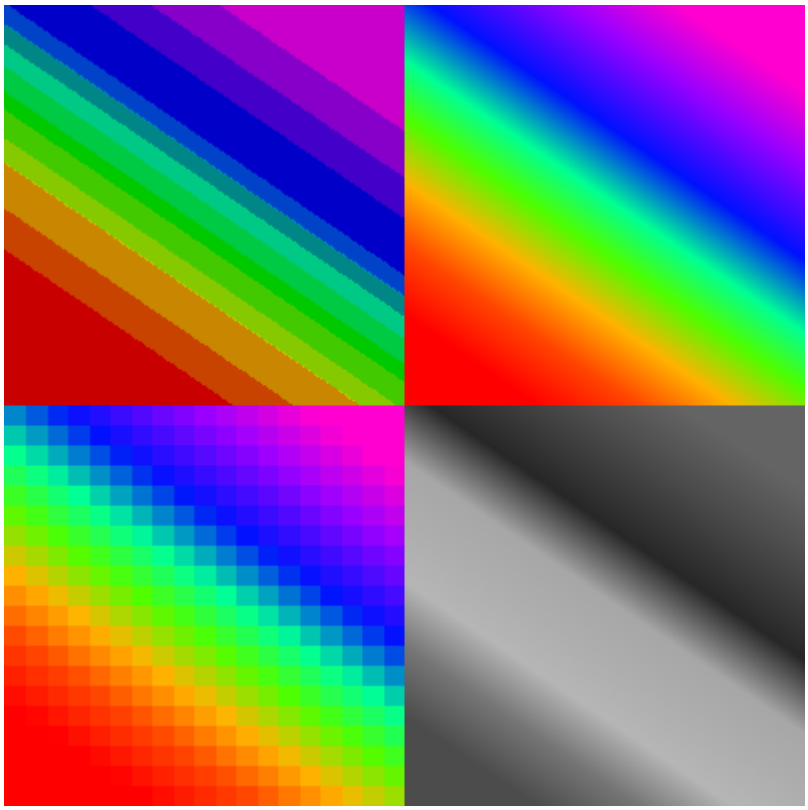


5. Set the Unity renderer to the newly created renderer in the render settings. (Project settings > Quality > Rendering).



2. Introduction

The **Processing Effects** asset can apply a certain effect onto an image. These effects can be adjusted depending on the given settings. Below are a few examples of what this asset is capable of.

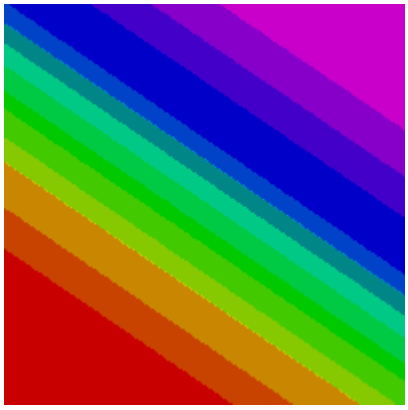


Original

The image used in this preview.

Posterization reduces the number of color tones in an image.

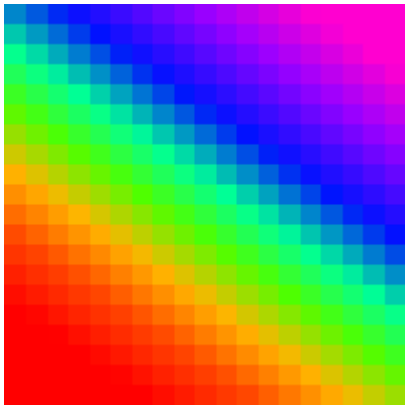
Every pixel gets checked, it then reduces the total amount of color tones. Giving a result as below.



Pixelation reduces the number of pixels in an image.

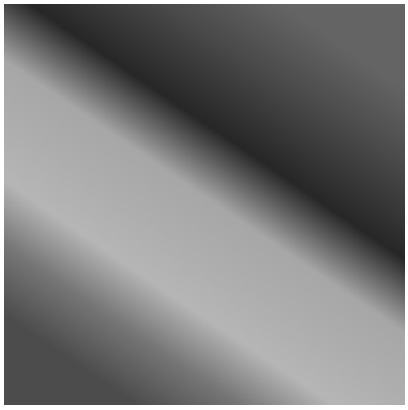
For each pixel in the image, the nearest color of the pixel will be used for that pixel.

Following this process for each pixel will give the following result.



Grayscaling turns the image into a black-and-white variant.

It takes the grayscale of the colors and turns them into gray based on it.



3. Set-Up

Down below is a basic procedure described on how to make use of the asset's functionality to a working result.

Required steps

The steps described below are important to get a working result of what the asset has to offer by modifying images and setting up the main foundation of the asset.

1. Add the asset to the Unity project.
2. Call one of the image effects' handler functions.

```
Texture2D posterizeTexture = PosterizeHandler.Posterize(_texture, _powerBase, _exponent);
Texture2D pixelateTexture = PixelateHandler.Pixelate(_texture, _pixelCount);
Texture2D grayscaleTexture = GrayscaleHandler.AddGrayscale(_texture, _grayscale);
Texture2D resizeTexture = ResizeHandler.Resize(_texture, _newResolution);
```

PosterizeHandler

Reduces the total range of color tones used.

PixelateHandler

Pixelates a texture. Lowers the resolution of the image while keeping the same dimensions.

GrayscaleHandler

Grants the options:

- Make a texture black/white.
- Make the texture fade into black/white.
- Exclude certain pixels of a texture based on the grayscale of the pixels.

ResizeHandler

Resizes the texture to fit the given resolution.

3. The Handler function will return the given image modified by the Handler.

Shader

The shader is made in **Unity Shader Graph**. Thus the shaders will only work in URP or HDRP. These render pipelines can be installed with the **Unity Package Manager**.

There are four shaders:

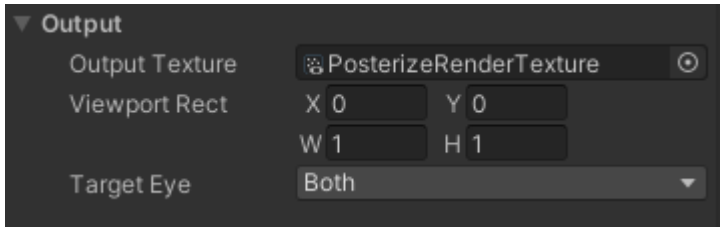
- **Posterize object:** This shader posterizes the object it is attached to.
- **Posterize camera:** This shader posterises the camera view.
- **Pixelate object:** This shader pixelates the object it is attached to.
- **Pixelate camera:** This shader pixelates the camera view.

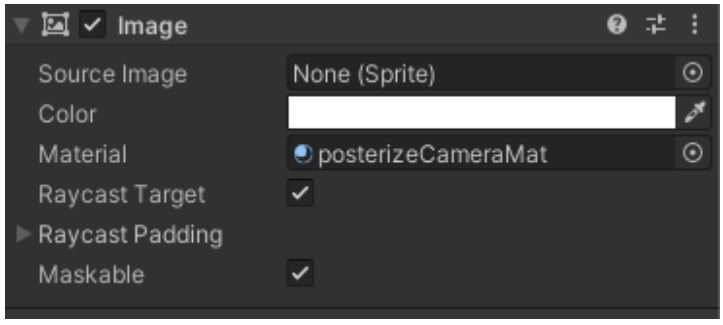
To set up an object with the shader effect applied to it:

1. Create a **material**. Right click in the Assets window > Create > Material.
2. Select the **material** that was just created.
3. Click on the shader button at the top of the **material**. In the search bar, type: posterize or pixelate.
4. Select the PosterizeObject or PixelateObject material.
5. Attach the **material** to the object that needs to be posterized or pixelated. This can be done by dragging the **material** onto the **gameobject**.

To set up a camera with the shader effect applied to it:

1. Create a material. Right click in the Assets window > Create > Material.
2. Select the material that was just created.
3. Click on the Shader button at the top of the material. In the search bar, type: posterize or pixelate.
4. Create a Render texture. Right click in the Assets window: Create > Render texture.
5. Drag the render texture into the Texture field of the material.
6. Make sure to have **2** cameras in the scene.
7. Select the camera that is used for rendering.
8. Place the created render texture in: Output > Output texture.
9. Now create an **Image** component or add the **Image** component to an existing UI object.
   To create a new **Image** component right click inside of the hierarchy view: Create > UI > Image.
10. Drag the created material in: Image > Material.





You will now have:

- A render texture.
- A material with the shader attached.
- A canvas with an object that has the **Image** component attached to it.
- Two cameras, one with the Render Texture as output and a normal one.

4. Editor

DTT Processing Effects has a preview editor window. Here is the possibility to see what the image will look like after posterizing it.

Go to the Editor window in the menu bar. **Tools > DTT > Processing Effects > Processing Effects Window**.

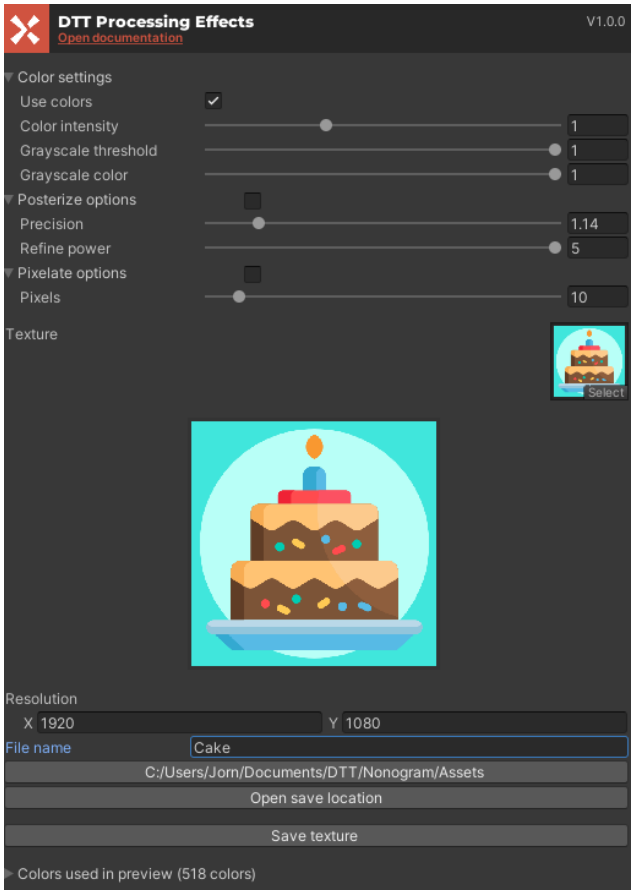
The options that can be adjusted:.

a. **Color settings:**
   i. **Use colors:** Uses colors in the preview when true. Uses black & white when false.
   ii. **Color intensity:** Makes the preview brighter/darker.
   iii. **Grayscale threshold:** The threshold to ignore a color.
   iv. **Grayscale color:** Grays out the image.

b. **Posterize options:**
   i. **Precision:** The main value to adjust the posterize effect.
   ii. **Refine power:** Refines the posterize effect.

c. **Pixelate options:**
   i. **Pixels:** The amount of pixelation used.

d. **Resolution:** The resolution the saved image will have.



e. **File name:** The name to save the preview as.
   i. **Button below:** The location to save the preview.
   ii. **Save texture:** Save the texture with the file name at the select path.

Select an image to use.

5. API

Posterize Handler

Method name	Return Type	Parameters	Description
Posterize	Color	Color : color, float : powerBase, float : exponent	Posterizes the given color.
Posterize	Color[]	Color[] : colors, float : powerBase, float : exponent	Posterizes the given color array.
Posterize	Textured2D	Textured2D : texture, float : powerBase, float : exponent	Posterizes the given texture.
Posterize	Textured2D	Textured2D : texture, Vector2Int : newSize float : powerBase, float : exponent	Posterizes the given texture and resizes it.

Pixelate Handler

Method name	Return Type	Parameters	Description
Pixelate	Color[]	Color[] : colors, int : pixelCount	Pixelates the given color array.
Pixelate	Textured2D	Textured2D : texture, int : pixelCount	Pixelates the given texture.
Pixelate	Textured2D	Textured2D : texture, Vector2Int : newSize int : pixelCount	Pixelates the given texture and resizes it.

Grayscale Handler

Method name	Return Type	Parameters	Description
ToBlackWhite	Color	Color : color, float : grayScale	Converts the given color into black and white.
ToBlackWhite	Color[]	Color[] : colors, float : grayScale	Converts the given color array into black and white.
ToBlackWhite	Textured2D	Textured2D : texture, float : grayScale	Converts the given color into black and white.
AddGrayscale	Color	Color : color, float : grayScale	Grays out the color to make it look grayish.
AddGrayscale	Color[]	Color[] : colors, float : grayScale	Grays out the color array to make it look grayish.
AddGrayscale	Textured2D	Textured2D : texture, float : grayScale	Grays out the texture to make it look grayish.
ExcludeGrayscale	Color	Color : color, float : grayScale	Return the given color or white based on the grayscale.
ExcludeGrayscale	Color[]	Color[] : colors, float : grayScale	Exludes the colors based on the grayscale.
ExcludeGrayscale	Textured2D	Textured2D : texture, float : grayScale	Exludes the pixels based on the grayscale.

Resize Handler

Method name	Return Type	Parameters	Description
Resize	Texture2D	Texture2D : texture, Vector2int : newSize	Scales the image up/down to fit the newSize.

6. Known Limitations

- The larger the image is, the more computing power it costs to process the modifications.
- Before an image can be modified. It needs read/write enabled and the format needs to be set to RGB.

7. Support and feedback

If you have any questions regarding the use of this asset, we are happy to help you out. Always feel free to contact us at:

unity-support@d-tt.nl

(We typically respond within 1-2 business days)

We are actively developing this asset, with many future updates and extensions already planned. We are eager to include feedback from our users in future updates, be they 'quality of life' improvements, new features, bug fixes or anything else that can help you improve your experience with this asset. You can reach us at the email above.

Reviews and ratings are very much appreciated as they help us raise awareness and to improve our assets.

DTT stands for Doing Things Together

DTT is an app, web and game development agency based in the centre of Amsterdam. Established in 2010, DTT has over a decade of experience in mobile, game, and web based technology.

Our game department primarily works in Unity where we put significant emphasis on the development of internal packages, allowing us to efficiently reuse code between projects. To support the Unity community, we are publishing a selection of our internal packages on the Asset Store, including this one.

More information about DTT (including our clients, projects and vacancies) can be found here:

https://www.d-tt.nl/en/