Photo Editor Docs

Made by Jordan Roth,

Contact me at anchorappsdev1@gmail.com

Quick Intro:

I'm going to make these docs as easy to read as possible, because I like things simple. If you feel as though anything within these docs or the plugin is too complicated, or you think needs improvement, please don't hesitate to tell me, this plugin is ours so let's make it great!

Table of Contents:

- **Getting Started** **2**
- **Setting Up a Custom Image** **5**
- **Demo Scene** **10**
- **Functions** **13**
  - **All Functions Explained** **14**

GETTING STARTED

Step 1 – Setup Image Effect:

Write this variable to access scripts.

```
ImageEffect nameOfYourVariable;
```

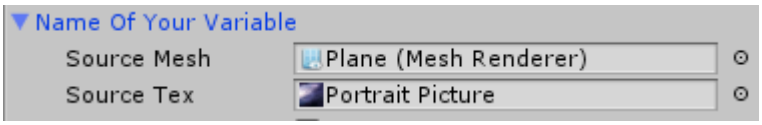
The name "nameOfYourVariable" is the name in which you want to give ImageEffect. You can add more ImageEffect variables with any name you want.

Step 2 – Setup SourceMesh and SourceTex:

Either assign the variables in the script as such,

```
void Start()
{
    nameOfYourVariable.sourceMesh = nameOfMesh;
    nameOfYourVariable.sourceTex = nameOfTexture;
}
```

or drag and drop



Source Mesh is a MeshRenderer
Source Tex is a Texture2D

Step 2.1 – Setup Cutout Image

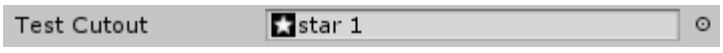
Create a Texture2D variable to hold your cutout image

```
Texture2D testCutout;
```

Either assign the vaibale in the script as such,

```
testCutout = nameOfCutout;
```

or drag and drop



To create your own cutout image check out the "Setting Up a Custom Image" section.

Step 3 – Use Function to Create New Images

To use a function, use this setup:

```
nameOfYourVariable.ToCircle();
```

void CondenceTexture(int constrain)
Compresses image to a new height and width

void ToCircle()
Cuts out a circle in the middle of your picture

void ToCircle(int constrain)
Compresses image, then cuts out circle

void ToSquare()
Cuts down image to match width and height

void ToSquare(int constrain)
Compresses image, then cuts out square

void ToEllipse()
Cuts out an ellipse into your picture, keeps the same width and height

void ToEllipse(int constrain)
Compresses image, then cuts out ellipse

void CustomCutout(Texture2D cutout)
Cuts out an image of your custom cutout template

void CustomCutout(Texture2D cutout , int constrain)
Compresses image, then cuts out custom cutout

Texture2D GrabCondenceTexture(int constrain)
Returns the new compressed image

Texture2D GrabToCircle()
Returns the new cutout circle

Texture2D GrabToCircle(int constrain)
Returns compressed circle

Texture2D GrabToSquare()
Returns the new cutout square

Texture2D GrabToSquare(int constrain)
Returns compressed square

Texture2D GrabToEllipse()
Returns the new cutout ellipse

Texture2D GrabToEllipse(int constrain)
Returns compressed ellipse

Texture2D GrabCustomCutout(Texture2D cutout)
Returns the new custom cutout

Texture2D GrabCustomCutout(Texture2D cutout, int constrain)
Returns compressed custom cutout

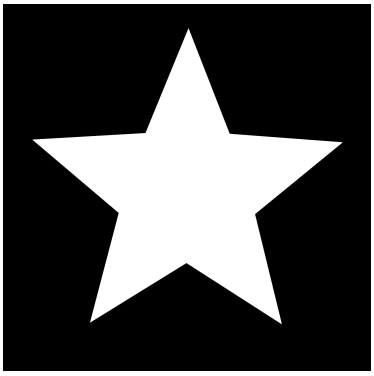
void set(Texture2D tex)
Sets SourceMesh texture as tex

void saveFile(string fileName, Texture2D tex)
Saves a new picture (tex) in your Assets folder named fileName

SETTING UP A CUSTOM IMAGE

Step 1 – Make a Black and White Image

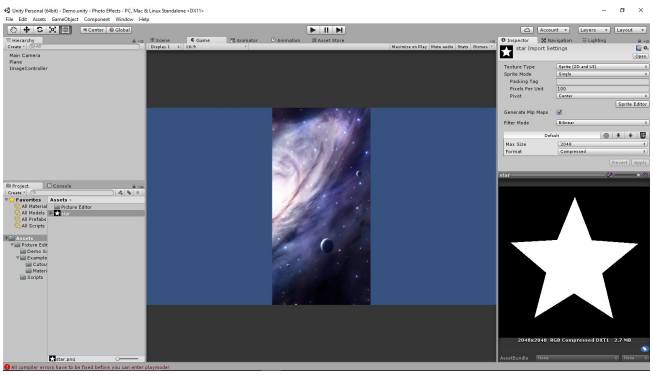
Creat an image that is only black and white. White reresents all the pixels that will stay the same, and black represents the pixels that will turn clear.



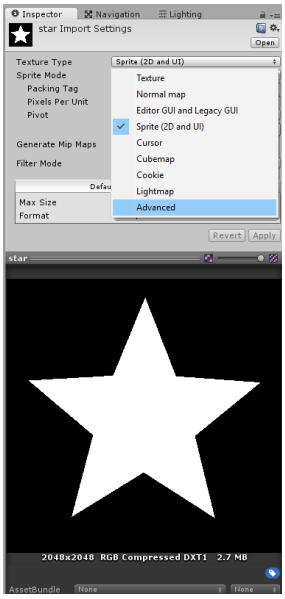
Black can actually be any color, only white pixels are looked at in code, I just like the way it looks as a template

Step 2 – Import and Change Settings

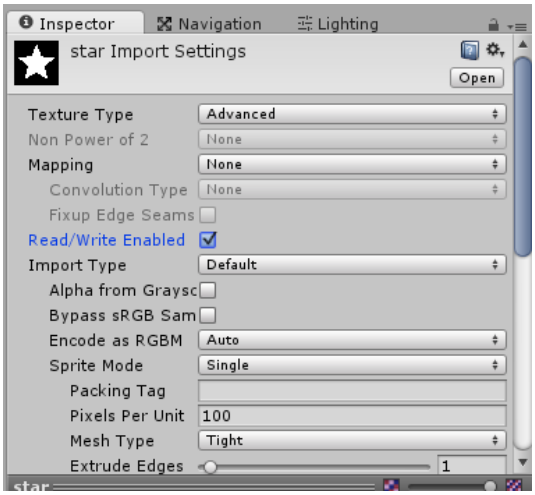
First, import the image into your project



Next, change your Texture Type to Advanced



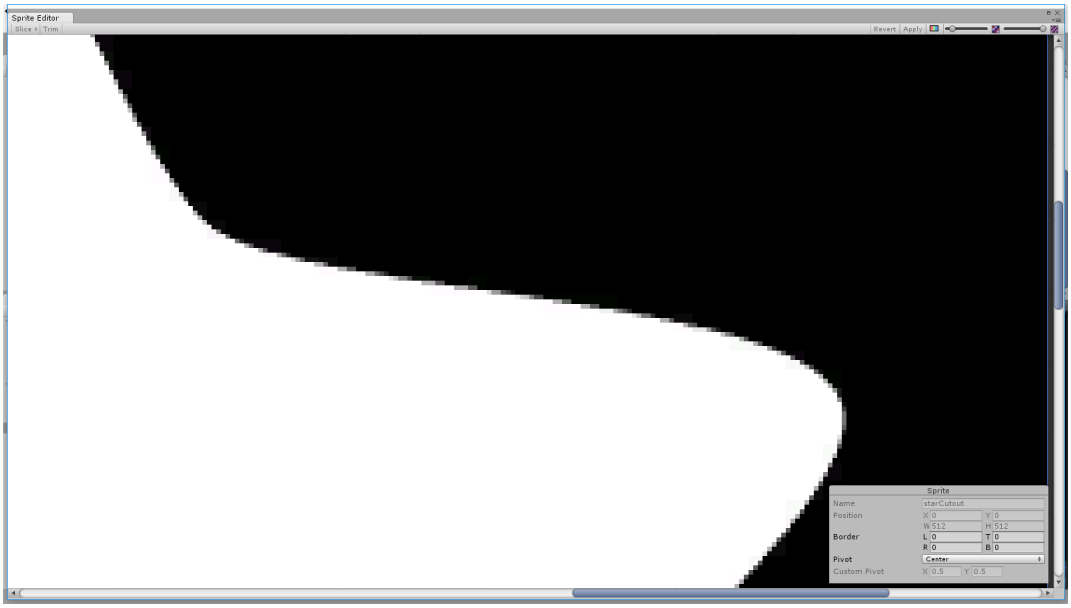
Third, check the Read/Write Enabled box



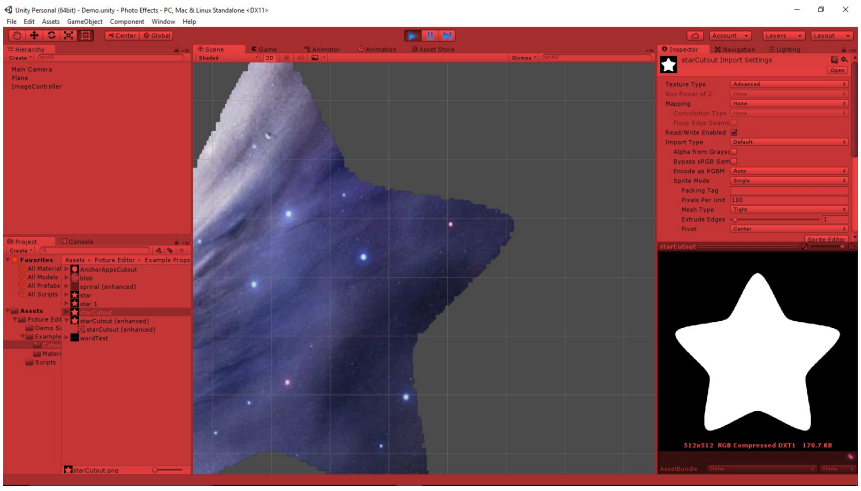
Finally, apply the changes

Step 3 – Take Advantage of the System

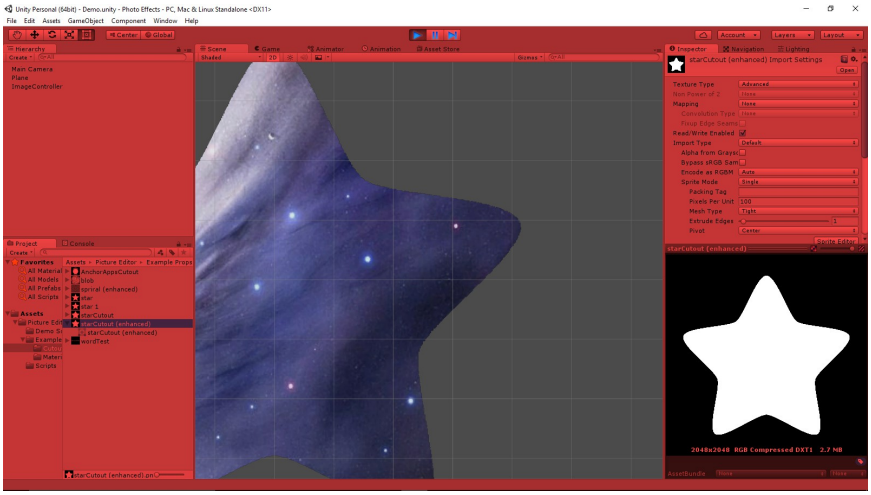
The program is set up to only register white pixels as pixels that will keep the original picture, so when you have an image like this:



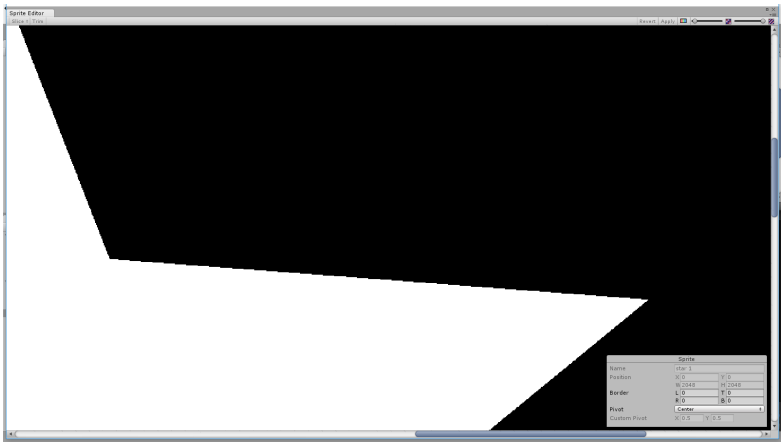
the grey pixels can affect your final image and create something ugly like this:

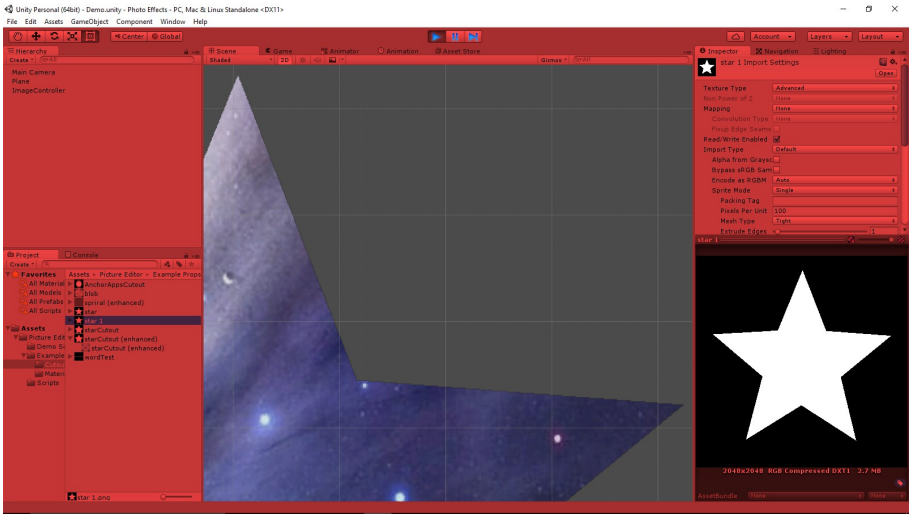


but to fix this issue you can increase the size of your template to a much larger texture (like 2048x2048) and don't worry, this won't affect the speed at which your image is cut out! If you recreate this cutout to 2048x2048 the results are a lot prettier:

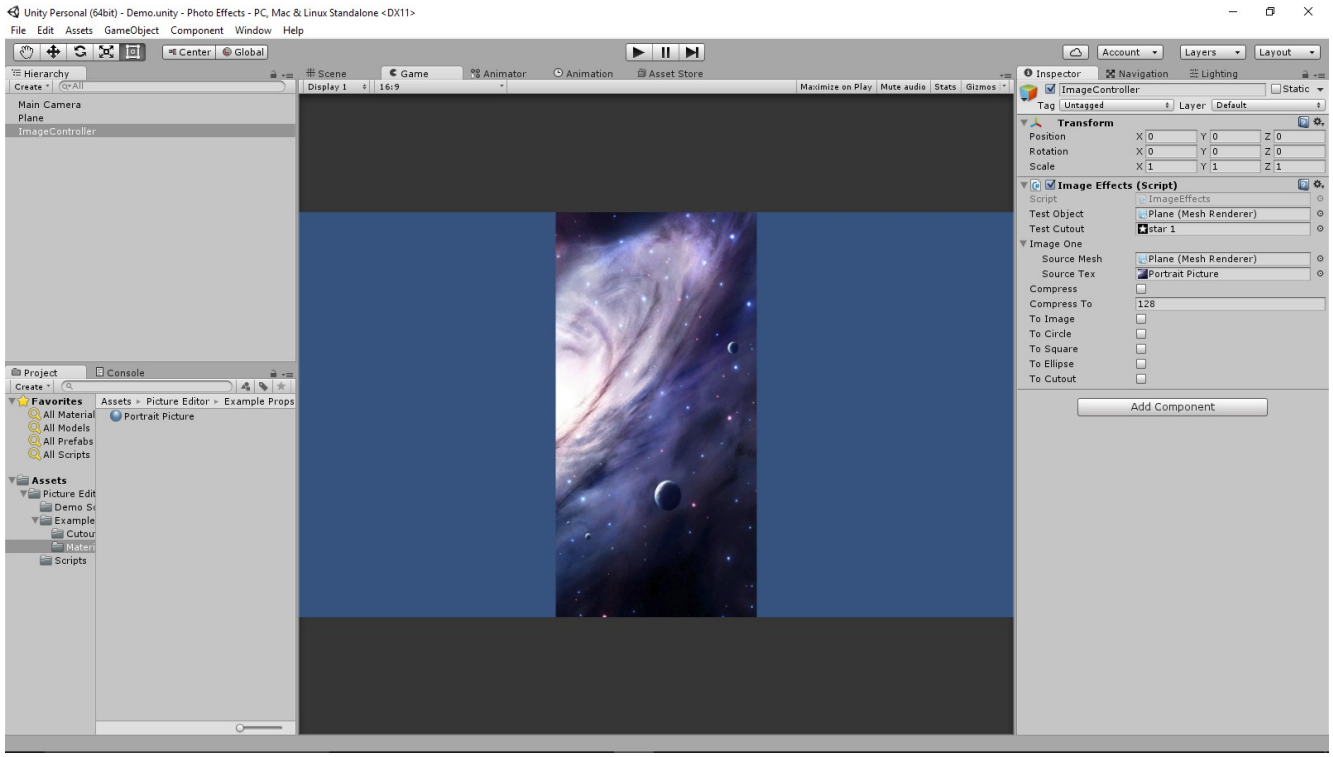


If all your pixels are a solid white where you want it to be, you'll get the nice crisp image you want:




File Edit Assets GameObject Component Window Help
Hierarchy
Main Camera
Plane
ImageController
Scene
Game
Animator
Animation
Asset Store
Inspector
Navigation
Lighting
star 1 Import Settings
Texture Type
Mapping
Read/Write Enabled
Import Type
Sprite Mode
Pixels Per Unit
Mesh Type
Extrude Edges
Project
Console
Favorites
Assets
star 1.png
2048x2048 RGB Compressed DXT1 2.7 MB

DEMO SCENE



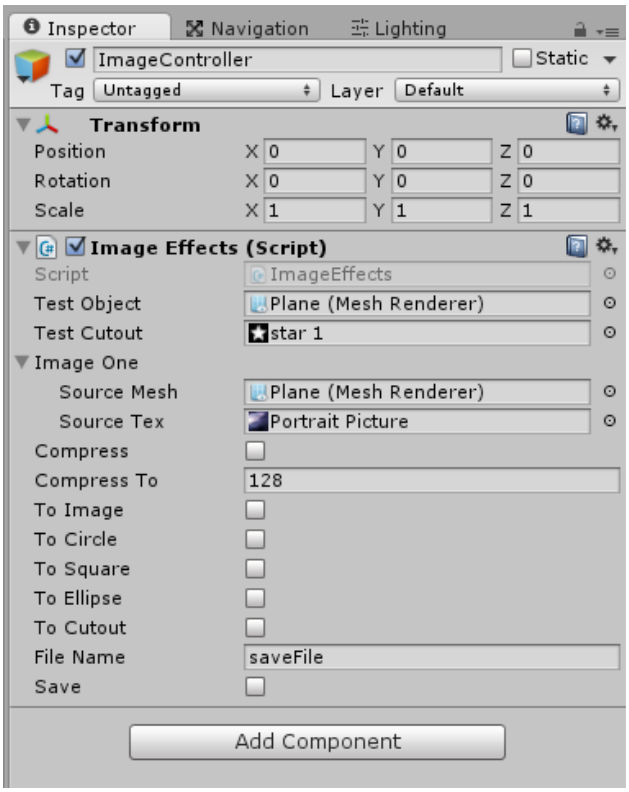
Let's look at the **hierarchy** and what each element does:

Main Camera: As you may know, this is what we use to look at our scene

Plane: This is a 3D element used to hold our texture. In the middle of the picture you will see an image of space, this is the image we are using, and is the texture on the **Plane**

ImageController: This is a script I designed for easy testing in our Demo Scene. When clicked, many booleans and other miscellaneous items appear, we will look deeper into this

ImageController:



ALL BOOLS MUST BE CLICKED DURING RUNTIME TO SEE THE EFFECTS

Test Object: The Plane's Mesh Renderer is assigned to this variable. It is used in our script to resize the image in the scene according to its height to width ratio (if we do not resize the plane, no matter what the height or width of the image may be in pixels, the image in the scene will have a static height and width).
Test Cutout: This is our cutout template used when cutting our image.
Image One: The variable used when accessing **Photo Editor**.
Source Mesh: The Mesh Renderer we want to assign the changes to.
Source Tex: The Picture we want to edit.
Compress: When clicked in runtime, the **Source Tex** compresses to **Compress To**.
Compress To: An integer used to state either the max height or width (whichever is larger) in pixels.
To Image: Changes the Image to the original image (**Source Tex**).

To Circle: Changes **Source Tex** to a cutout of a circle (makes the new image width equal to the new image height).
To Square: Changes **Source Tex** to a cutout of a square (makes the new image width equal to the new image height).
To Ellipse: Changes **Source Tex** to a cutout of an ellipse. The width and height stay the same.
To Cutout: Changes **Source Tex** to a cutout of **Test Cutout**.
File Name: The name of the file you want to create.
Save: Creates an image of what is currently on the **Source Mesh**. The new image is saved in the main Assets Folder.

FUNCTIONS

The functions where stated earlier in the **GETTING STARTED** section under Step 3, but I am going to use this section to hopefully better organize the functions and help give a quicker reference to you. If you find a better way to organize this section, please don't hesitate to contact me!

How to Access Functions and Variables

Use this quick and easy integration to access the functions and variables with ease!

First assign ImageEffect

```
ImageEffect nameOfYourVariable;
```

Next, access a function with any of these examples:

```
nameOfYourVariable.ToCircle();
nameOfYourVariable.ToSquare(256);
nameOfYourVariable.CustomCutout(nameOfYourCutout, 128);
nameOfYourVariable.saveFile("Condenced Image", nameOfYourVariable.GrabCondenceTexture (512));
```

Basic Set Functions:

Used to quickly crop the image into a shape without a decrease in pixel size

```
void ToCircle()
void ToSquare()
void ToEllipse()
void CustomCutout(Texture2D cutout)
```

Condencing Functions

Used to possibly apply a cutout while limiting the height and width of a picture (Can greatly enhance RAM)

```
void CondenceTexture(int constrain)
void ToCircle(int constrain)
void ToSquare(int constrain)
void ToEllipse(int constrain)
void CustomCutout(Texture2D cutout , int constrain)
```

Grabbing Textures

Grabs the new cutout texture assigned to your variable without a decrease in pixel size

```
Texture2D GrabToCircle()
Texture2D GrabToSquare()
Texture2D GrabToEllipse()
Texture2D GrabCustomCutout(Texture2D cutout)
```

Grabbing Condenced Textures

Grabs the new cutout image while limiting the height and width of the picture (Can greatly enhance RAM).

Texture2D GrabCondenceTexture(int constrain)
Texture2D GrabToCircle(int constrain)
Texture2D GrabToSquare(int constrain)
Texture2D GrabToEllipse(int constrain)
Texture2D GrabCustomCutout(Texture2D cutout, int constrain)

Extra Functions

A couple extra functions to make your experience better

void set(Texture2D tex)
void saveFile(string fileName, Texture2D tex)

All Functions Explained

A small dictionary to help you figure out everything about the functions and variables.

MeshRenderer sourceMesh;
The MeshRenderer you want to assign the image to.

Texture2D sourceTex;
The image used on the sourceMesh.

void CondenceTexture(int constrain)
Compresses image to a new height and width. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

void ToCircle()
Cuts out a circle in the middle of your picture.

void ToCircle(int constrain)
Compresses image, then cuts out circle. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

void ToSquare()
Cuts down image to match width and height.

void ToSquare(int constrain)
Compresses image, then cuts out square. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

void ToEllipse()
Cuts out an ellipse into your picture, keeps the same width and height.

void ToEllipse(int constrain)
Compresses image, then cuts out ellipse. The constrain variable limits the height and width in pixels. For

example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

void CustomCutout(Texture2D cutout)
Cuts out an image of your custom cutout template. The cutout variable is the cutout template picture you want your `ImageEffect` sourceTex to be cut out of.

void CustomCutout(Texture2D cutout , int constrain)
Compresses image, then cuts out custom cutout. The cutout variable is the cutout template picture you want your `ImageEffect` sourceTex to be cut out of. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

Texture2D GrabCondenceTexture(int constrain)
Returns the new compressed image. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

Texture2D GrabToCircle()
Cuts out a circle in the middle of your picture and returns new image.

Texture2D GrabToCircle(int constrain)
Cuts out a circle in the middle of your picture and returns new compressed image. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

Texture2D GrabToSquare()
Cuts down image to match width and height and returns new image.

Texture2D GrabToSquare(int constrain)
Cuts down image to match width and height and returns new compressed image. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

Texture2D GrabToEllipse()
Cuts out an ellipse into your picture, keeps the same width and height and returns new image.

Texture2D GrabToEllipse(int constrain)
Cuts out an ellipse into your picture, keeps the same width and height and returns new compressed image. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

Texture2D GrabCustomCutout(Texture2D cutout)
Cuts out an image of your custom cutout template and returns new image.The cutout variable is the cutout template picture you want your `ImageEffect` sourceTex to be cut out of.

Texture2D GrabCustomCutout(Texture2D cutout, int constrain)
Cuts out an image of your custom cutout template and returns new image.The cutout variable is the cutout template picture you want your `ImageEffect` sourceTex to be cut out of. The constrain variable limits the height and width in pixels. For example, if the original image is 2048x1024, with a constrain value of 128, your new resolution will be 128x64.

void set(Texture2D tex)
Sets SourceMesh texture as tex. The tex variable is an image of what you want to be seen on the SourceMesh

texture.

void saveFile(string fileName, Texture2D tex)
Saves a new picture (tex) in your Assets folder named fileName. The fileName varible is what is used as the new name of your saved file. The tex variable is the image you want to save to your Assets folder.