TEXTURE UTILITY

Basic manual v1.0

Introduction

This nice package provides you with 25+ robust and convenient utilities to manipulate Texture2D assets.
It allows to change textures on pixel level right in runtime, to achieve a lot of complex and useful effects.
This drastically extends Unity's builtin functionality of changing textures, bringing such features as:

- Applying masks and Boolean operations
- Rotating, scale, mirroring, expanding, crop and auto-crop
- Changing contrast, brightness, HUE, saturation
- Colorizing, grayscale, negative and leaving color diapason
- Inverting transparency and making color transparent
- Stroke, etc

How to use

Usage is extremely easy and straightforward:

1. Prepare texture-assets you want to use. This can be done manually or using such functions as

    TextureUtility.PrepareAsset(_source: Texture2D); *- for existing texture*
    TextureUtility.CreateTexture (_width: int, _height: int, _color: Color); *- to create new texture*

    The most important - textures should be readable!

2. Call desired functions from *TextureUtility* namespace. For example:

```
TextureUtility.FlipHorizontally ( yourSourceTexture );
TextureUtility.Rotate  (yourSourceTexture, desiredRotationAngle );
TextureUtility.AutoCropTransparency ( yourSourceTexture );
```

    Please check more info about functions and parameters on the next pages.

Hints and tips

1. Some functions have to versions – for *Texture2D* snd *Color[]* sources. It's recommended to use *Color[]* version if you need to call function often (this will save memory and CPU, especially if you'll pass the same pixel-array through several functions).

2. In texture import settings it's better to have: *textureFormat = RGBA32, isReadable = true; mipmapEnabled = false; alphaIsTransparency = true;*

3. You can change Texture2D asset itself, even edit images within unity.
   To save new image as asset – use function TextureUtility.SaveAsAsset(_source: Texture2D, _path: String)

4. Be careful to not override source texture. It's recommended to create/use a copy right from the start.

5. For your comfort, some functions (ApplyBooleanOperation as instance) can change result texture width/height implicitly. Please take this into account.

6. Some functions using transparency for operations (like ApplyMask, ApplyBooleanOperation, AutoCropTransparency) uses binary values (1 or 0), so for proper result your transparency alpha should be exactly 0 (i.e. semitransparent pixels will be considered as non-transparent)

If you have any questions/issues – please don't hesitate to contact me via AllebiGames@gmail.com

FUNCTIONS & SYNTAXIS

Enums used for more convenient setup:

BooleanOperation : Union, Intersection, Subtraction

Union - *result Texture will contains all non-transparent pixels of both textures*
Intersection - *result Texture will contains only overlapped non-transparent pixels of textures*
Subtraction - *result Texture will contains only non-transparent pixels which aren't overlapped*

Alignment : TopLeft, TopRight, TopCenter, BottomLeft, BottomRight, BottomCenter
BlendMode : Normal, Additive, Subtraction, Multiply, Subdivide, MaskAlpha

Asset preparation functions:

Prepare the texture to be operable by TextureUtility (Modify the importer settings)
PrepareAsset (_source: Texture2D): Texture2D

Save texture as asset to path in Assets folder
SaveAsAsset (_source: Texture2D, _path: String)

Create new Texture2D/PixelArray filled by custom color
CreateTexture (_width: int, _height: int, _color: Color): Texture2D
CreatePixelArray (_width: int, _height: int, _color: Color): Color[]

Functions to manipulate colors etc:

Clear Texture2D/PixelArray by custom color
Clear (_source: Texture2D, _color: Color): Texture2D
Clear (_source: Color[], _color: Color): Color[]

Invert Texture2D/PixelArray transparency
InvertTransparency (_source: Texture2D): Texture2D
InvertTransparency (_source: Color[]): Color[]

Make custom color of Texture2D/PixelArray completely transparent (Alpha-key)
MakeColorTransparent (_source: Texture2D, _color: Color): Texture2D
MakeColorTransparent (_source: Color[], _color: Color): Color[]

Change Texture2D/PixelArray HSB (allows to adjust HUE, Saturation and Brightness)
ChangeHSB(_source: Texture2D, _hue: float, _saturation: float, _brightness: float): Texture2D
ChangeHSB(_source: Color[], _hue: float, _saturation: float, _brightness: float): Color[]

Change Texture2D/PixelArray contrast
Contrast (_source: Texture2D, _contrast: float): Texture2D
Contrast (_source: Color[], _contrast: float): Color[]

Leave in Texture2D/PixelArray within pixels with color only within custom diapason (all other pixel become grayscale)
ColorDiapason (_source: Texture2D, _colorStart: Color, _colorEnd: Color): Texture2D
ColorDiapason (_source: Color[], _colorStart: Color, _colorEnd: Color): Color[]

Colorize Texture2D/PixelArray by custom color and intensity
Colorize (_source: Texture2D, _color: Color, _intensity: float): Texture2D
Colorize (_source: Color[], _color: Color, _intensity: float): Color[]

Turn Texture2D/PixelArray to grayscale
Grayscale (_source: Texture2D): Texture2D
Grayscale (_source: Color[]): Color[]

Turn Texture2D/PixelArray to negative of itself
Negative (_source: Texture2D): Texture2D
Negative (_source: Color[]): Color[]

Functions to manipulate texture geometry etc:

Flip Texture2D horizontally
FlipHorizontally (_source: Texture2D): Texture2D

Flip Texture2D vertically
FlipVertically (_source: Texture2D): Texture2D

Change Texture2D canvas width/height by expanding in specified directions (function doesn't scale image itself)
Expand (_source: Texture2D, _newWidth: int, _newHeight: int, _sourceAlignment: Alignment): Texture2D

Crop Texture2D by extracting it piece in specified rectangle
Crop (_source: Texture2D, _rect: Rect): Texture

Automatically crop transparent pixels surrounding image
AutoCropTransparency (_source: Texture2D): Texture2D

Automatically crop pixels(custom color) surrounding image
AutoCropColor (_source: Texture2D, _color: Color): Texture2D

Apply transparency(Alpha-chanel) mask to Texture2D
ApplyMask (_source: Texture2D, _mask: Texture2D): Texture2D

Merge 2 Textures by apply Boolean operation(based on Alpha-channel) to them. Result-texture size will be expanded if needed.
ApplyBooleanOperation (_operationType: BooleanOperation, _base: Texture2D, _addition: Texture2D, _additionOffset: Vector2): Texture2D

Rotate Texture2D on custom angle
Rotate(_source: Texture2D, _angle: float): Texture2D

Scale Texture2D with new width/height
Scale (_source: Texture2D, _targetWidth: int, _targetHeight: int): Texture2D

Stroke(colorize edge pixels) Texture2D by color with specified thickness and blendMode
Stroke (_source: Texture2D, _thickness: int, _color: Color, _blendMode: BlendMode) : Texture2D

Accessory functions to calculate/convert colors:

Get average color in PixelsArray
GetAverageColor(_pixels: Color[]) : Color

Blend 2 colors using one of BlendModes
BlendColors (_color1: Color, _color2: Color, _blendMode: BlendMode) : Color

Convert Color to HSBA (Vector4)
ColorToHSBA (_color: Color) : Vector4

Convert HSBA(Vector4) to Color
HSBAtoColor (hsba: Vector4): Color