Check video tutorials:

https://www.youtube.com/watch?v=cTWlXsiXcRU

https://www.youtube.com/watch?v=86_EmsOjewk

`Adjustments API` can be brought into scope with this **using** directive:

- (c#) `using VacuumShaders.TextureAdjustments;`
- (java) `import VacuumShaders.TextureAdjustments;`

Classes:

Color correction:

- Adjust_ColorSpace
- Adjust_Levels
- Adjust_HueSaturationLigthness
- Adjust_BrightnessContrast
- Adjust_Grayscale
- Adjust_Threshold
- Adjust_ColorOverlay
- Adjust_ColorReplace
- Adjust_GradientRamp
- Adjust_LUT

Channel

- Adjust_ChannelInvert
- Adjust_ChannelSwap
- Adjust_ChannelImport

Blur effects

- Adjust_BlurGaussian
- Adjust_BlurDirectional
- Adjust_BlurGrainy (may not work on mobile devices)

Other

- Adjust_Pixelate (may not work on mobile devices)
- Adjust_EdgePadding
- Adjust _Noise
- Adjust _Sharpen
- Adjust_TextureBombing (may not work on mobile devices)

Generate

- Adjust_GenerateGradient
- Adjust_GenerateCheckerboard
- Adjust_GenerateGrid
- Adjust_GenerateShape
- Adjust_GenerateGradientNoise (may not work on mobile devices)
- Adjust_GeneratePixelatedNoise (may not work on mobile devices)
- Adjust_GenerateSimpleNoise (may not work on mobile devices)

Transform

- Adjust_Flip
- Adjust_TilingOffset
- Adjust_Rotator
- Adjust_Twirl
- Adjust_Rotate
- Adjust_Resize
- Adjust_Crop

Adjustment is applied only if it is enabled - `public bool isEnabled`

For Appling adjustment to Texture2D:
`public void Apply(Texture _srcTexture, ref Texture2D _dstTexture)`
Current adjustment will be applied to `_srcTexture` and results will be saved inside `_dstTexture`.

For Appling adjustment to RenderTexture:
`public void Apply(Texture _srcTexture, ref RenderTexture _dstTexture, bool _dstTextureIsTemporary)`
`_dstTextureIsTemporary` – if `_dstTexture` is created with `RenderTexture.GetTemporary()`.

For applying multiple adjustments to Texture2D use static method:

```
TextureAdjustments.RenderAll(Texture _srcTexture,
                             ref Texture2D _dstTexture,
                             params AdjustmentBase[] _adjustments)
```

For applying multiple adjustments to RenderTexture use static method:

```
TextureAdjustments.RenderAll(Texture _srcTexture,
                             ref RenderTexture _dstTexture, bool _dstTextureIsTemporary,
                             params AdjustmentBase[] _adjustments)
```

Adjustments order is taking into account – except:

- `Adjust_Resize` class - It is always applied as the first adjustment. If there are multiple `Adjust_Resize` classes, only first one is used.
- `Adjust_Rotate` class – It is always applied as the last adjustment. If there are multiple `Adjust_Rotate` class, only first one is used.

Adjustments are calculated using their own shaders (check ***Texture Adjustments/Shaders*** folder).
Do not forget to include them into ***Always Include Shaders*** array in ***Edit/Project Settings/Graphics*** window if adjustment will be used in the final build.

Note!
Even adjustments calculation are GPU accelerated there shouldn't be used in every update function – they are very fast but aren't unlimited.
Each adjustment adds several draw calls and may decrease scene's overall performance.

Encode Texture2D objects in run-time

Add `VacuumShaders.TextureExtensions` directive to the script:

- (c#) `using VacuumShaders.TextureExtensions;`
- (java) `import VacuumShaders.TextureExtensions;`

Now Texture2D class built-in methods EncodeToJPG, EncodeToPNG and EncodeToTGA will have additional ***boolean*** parameter determining to use EncodePro or not. If variable is ***false*** Unity default method will be used to encode texture, otherwise EncodePro.

Resize Pro

`public void ResizePro(int width, int height);`
Resizes the texture - Changes size of texture to width by height, with mip maps and original texture format.
Texture must be readable and in uncompressed format.

`public void ResizePro(int width, int height, out Texture2D dstTexture);`
Resizes the texture - Changes size of texture to width by height, with mip maps and saves result in `dstTexture.`
Original texture is not modified.
Texture can be in any format and not necessary to be readable.

The `ResizePro` extension method can be brought into scope with this **using** directive:

- (c#) `using VacuumShaders.TextureExtensions;`
- (java) `import VacuumShaders.TextureExtensions;`

The `ResizePro` extension method is added to the UnityEngine Texture2D class.

Split

`public Texture2D[] Split(int xCount, int yCount);`
Splits texture horizontally in `xCount` and vertically in `yCount` and returns result as Texture2D array.
Source texture is not modified.
Source texture must be readable and in uncompressed format.

`public void Split(int xCount, int yCount, out Texture2D[] dstTexture);`
Splits texture horizontally in `xCount` and vertically in `yCount` and returns result in `dstTexture.`
Source texture is not modified.
Texture can be in any format and not necessary to be readable.

The `Split` extension method can be brought into scope with this **using** directive:

- (c#) `using VacuumShaders.TextureExtensions;`
- (java) `import VacuumShaders.TextureExtensions;`

The `Split` extension method is added to the UnityEngine Texture2D class.