ALL IN 1 SPRITE SHADER

By Seaside Studios
For technical support and requests write to:
seasidegamestudios@gmail.com
Twitter of the creator: https://twitter.com/GerardBelenguer

Index

Overview

First of all thanks for downloading this asset! The intention of this asset is to provide you with an all in one solution to include cool popular sprite and UI images effects to your project in the easiest and fastest way possible.

What makes this asset unique is that you choose which effects you desire and the material will blend and stack all your desired effects appropriately. So the same material allows you to create a huge variety of effects without altering the setup of your sprites.

Link to the youtube playlist that explains how to use this asset:
https://www.youtube.com/playlist?list=PLKS0HUbkxp-lL2P67N9W0_fJ_55lynpPP

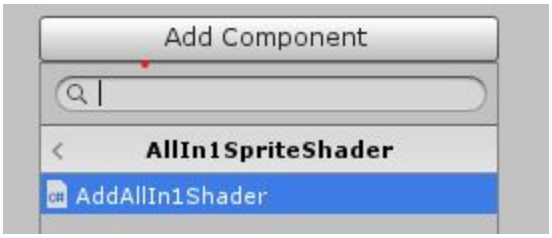
Feel free to contact me over at this email if you have any issue, request, question or want to show off your work: seasidegamestudios@gmail.com

Please make sure to drop a review on the Asset Store page if you like it. I helps a ton:
https://assetstore.unity.com/packages/vfx/shaders/all-in-1-sprite-shader-156513

How to use

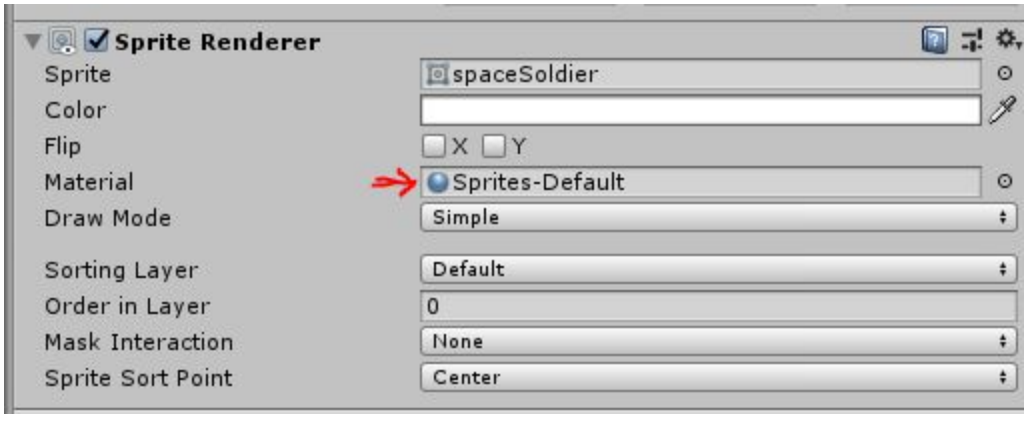
Here you have a link to a video that explains how to add the tool to your sprites in case you prefer a visual explanation:
https://youtu.be/47Q9uPL_C0U

The asset includes a component that will do all the setup for you. The component is called “AllIn1Shader”:



When you add it, the component will swap the current material for a new instance of the AllInOneSpriteShader material. The component also has some features that are overviewed in the next point.

But you can also do it the classic way: right clicking the AllInOneSpriteShader and then going Create->Material. You can then name this material and drag it into the Sprite Renderer Material slot in the Inspector of the desired Sprite:



The same process can be followed for UI images.

Component Features

Once added the component will look like this:

The buttons do the following:

- **Deactivate All Effects**: It will deactivate all effects but won't touch any properties. So if you activate an effect again you will obtain your previous visual results.
- **New Clean Material**: It will create a new instance of the AllInOneSpriteShader material and assign it to the Sprite.
- **Create New Material With Same Properties**: It will create a new instance of the AllInOneSpriteShader material with the same properties of the previous one. This is useful when you want to create a material similar to another one.
- **Save Material to Folder**: Creates a Material asset with the name of the current GameObject and saves it in the following path:"Assets/AllIn1SpriteShader/Materials". This can be used to assign the same Material to many different Sprites.
- **Sprite Atlas Auto Setup**: Use this in case your sprite is contained inside an atlas. This will add the SetAtlasUvs component for you and will make sure that the effects get properly drawn on your sprite (See Sprite Atlases section for more details).

- **Remove Sprite Atlas Configuration**: Removes SetAtlasUvs component and the rest of the sprite atlas configuration.
- **Remove Component and Material**: Removes the component from the GameObject and sets the Sprite Material back to the Sprite/Default one.

Textures Setup

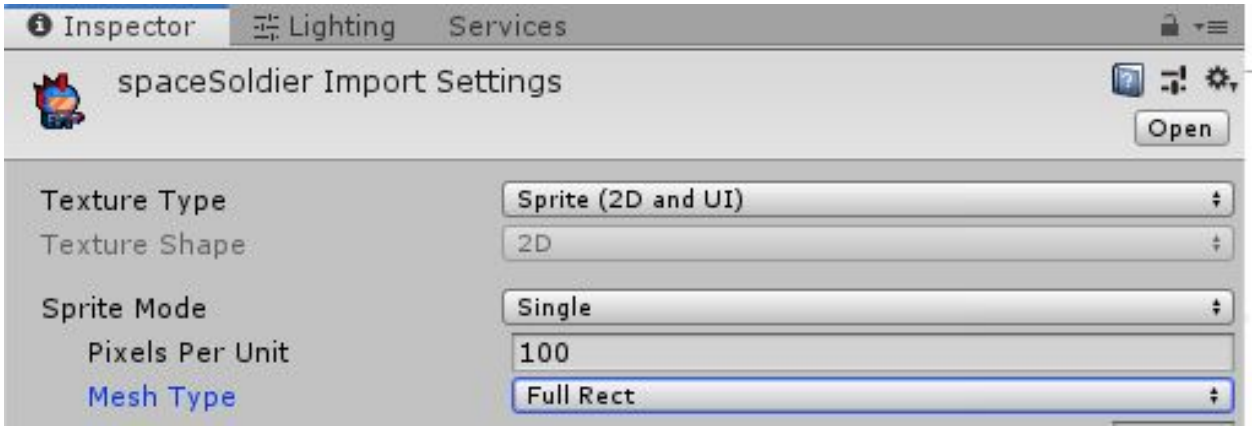
In order to get the effects looking as they should, it's important to know is how to import and setup the textures we'll be using for our sprites and UI images.

Here you have a link to a video that explains how to setup your textures in case you prefer a visual explanation:
https://youtu.be/K4I8OCZAMio

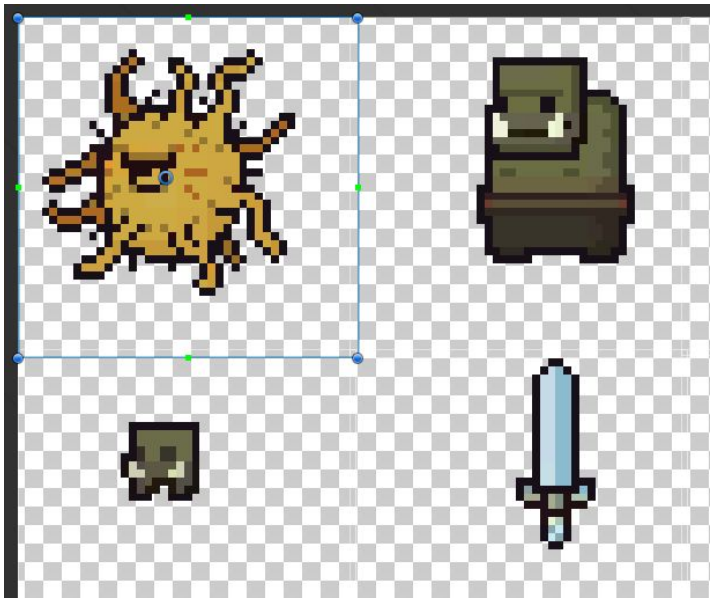
1. The most important part is having room within the sprite shape to show the effects. In the top of the Scene window you can choose how the scene view is rendered. If we change from Shaded to ShadedWireframe we'll be able to see the sprite rect size:

In the Shaded Wireframe view we can see black lines, that's the shape of the mesh where your sprite is being rendered. To get all effects to display properly we'll need more space.

The easiest way of doing so is changing the Mesh Type to Full Rect in the import settings of the sprite:



2. In a similar fashion if we are using a spritesheet you must import the sprites in Multiple Sprite Mode (as you always do) and then make sure that when you Slice your sprites they have some spacing:



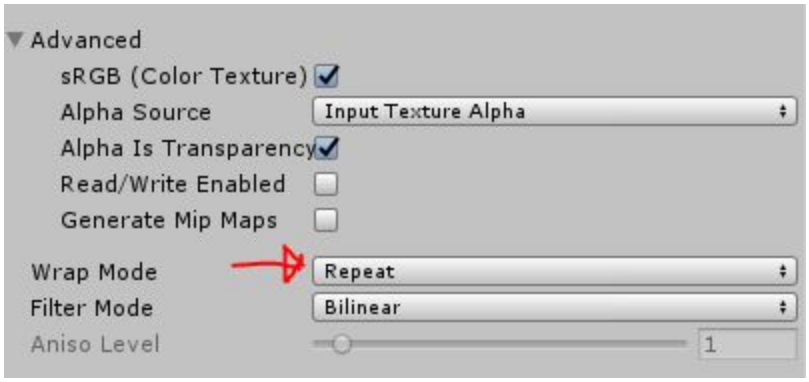
Notice how sprites have a generous spacing between them.
Be aware that some effects won't display correctly unless you setup your atlas sprites properly, see next section to know how (Sprite Atlases).

3. You'll see that some effects have a Glow property. In order for these properties to have the desired effect you need to add Post Processing and Bloom to the Main Camera of your scene.

   If you don't know how this is done you can follow this video:
   https://youtu.be/Iy6EGFLvLBQ

4. Most of the time it will be a good idea to set the Wrap Mode of the Import Settings of the textures you use to Repeat. This will assure a

proper result when using effects that use scrolling textures



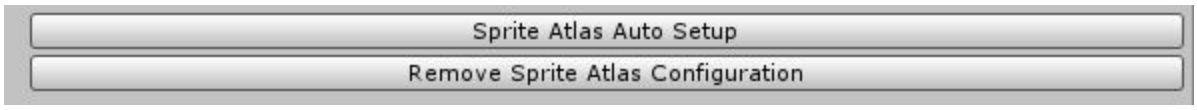
5. The same exact process applies to UI images

Sprite Atlases

You can also find a video about it here:
https://www.youtube.com/watch?v=mD03qkDTwCI

By default if you use this tool with a sprite that's inside of a sprite atlas you'll see that the visual results that you obtain are not the ones that you expect. This is because the Sprite Renderer knows that it's a partial image (since it's inside an atlas) but the shader doesn't, which causes unexpected results.

To avoid this there are 2 buttons in the component editor to automatically add a component that will manage all of this for us:



With the Setup button the SetAtlasUvs component will be automatically added:

This will manage everything for us so we don't have to worry about manually doing anything. And in case we want to remove this setup we can press the second button and everything will go back to normal.

If we are using a sprite that won't be changing we can leave everything as is after pressing the Setup button but if we are using an animation that swaps the sprite (a flipbook) we'll need to check the Update Every Frame checkbox in the SetAtlasUvs component:

UI Images work the same but have a slight exception: for this to work every sprite inside the same Atlas must have a different material assigned. This can be achieved in several ways:

1. Adding the AllIn1SpriteShader component (this will create a fresh new material)
2. Duplicating an existing image gameObject and then pressing the New Clean Material or New Material with same properties button
3. Duplicating a saved material (Ctrl+D when selected) and assigning this new copy to the desired Image component

Finally keep in mind that there are some effects that won't work properly with this setup and that will only work with sprites that are outside an atlas, these effects are:
Rect Size, Zoom, Twist, Polar Coords, Rotate, Fish Eye and Pinch.

How to animate effects

The custom material inspector properties can be animated through the Animation window as any other Unity component.

If you don't know how this is done you can follow this video:
https://www.youtube.com/watch?v=28kfPcAIwuY&list=PLKS0HUbkxp-lL2P67N9W0_fJ_55lynpPP&index=5&t=0s

Scripting

If you prefer avoiding animations or want to change properties through code you also have the possibility.

To do so you'll need to use the following Unity functions:

- Material.SetFloat: https://docs.unity3d.com/ScriptReference/Material.SetFloat.html
- Material.SetColor: https://docs.unity3d.com/ScriptReference/Material.SetColor.html
- Material.SetTexture: https://docs.unity3d.com/ScriptReference/Material.SetTexture.html

You can find all property names on AllIn1SpriteShader/Resources/AllIn1SpriteShader.shader. All properties are located from line 6 to 193 and can also be found at the Effects and Properties Breakdown section.

Here an example code snippet:

```
Material mat = GetComponent<Renderer>().material;
mat.SetFloat("_Alpha", 1f);
mat.SetColor("_Color", new Color(0.5f, 1f, 0f, 1f));
mat.SetTexture("_MainTex", texture);
```

*Note that there is an important distinction to be made between a "material" and a "sharedMaterial" of a Renderer. You shall use "material" if you only want to change a property of that instance of the material. And "sharedMaterial" if you want to change the property of all the instances of that material

How to Enable/Disable Effects at Runtime

There are 2 ways of achieving this:

1. All effects have a property value combination that makes them look deactivated (usually by reducing the amount to 0, but it may vary depending on the effect). So the most clean way of deactivating and activating effects is by enabling all the effects you'll use and then dynamically changing the property values either by animating the properties or by modifying the values by script as seen in the previous section.
2. This other way is less efficient, messier and may not even work on some platforms. So be warned, use this with caution and test it on the target platform before committing to this solution.
   It consists on enabling and disabling the shader compilation flags at runtime, so Unity will compile and replace the shader at runtime. To do so you first need to have a reference to the material and then use the Enable/Disable Keyword method like so:

```
Material mat = GetComponent<Renderer>().material;
…
mat.EnableKeyword("GRADIENT_ON");
mat.DisableKeyword("GRADIENT_ON");
```

(Keyword names of every effect can be found at the Effects and Properties Breakdown section)

Experimental Sprite Lighting

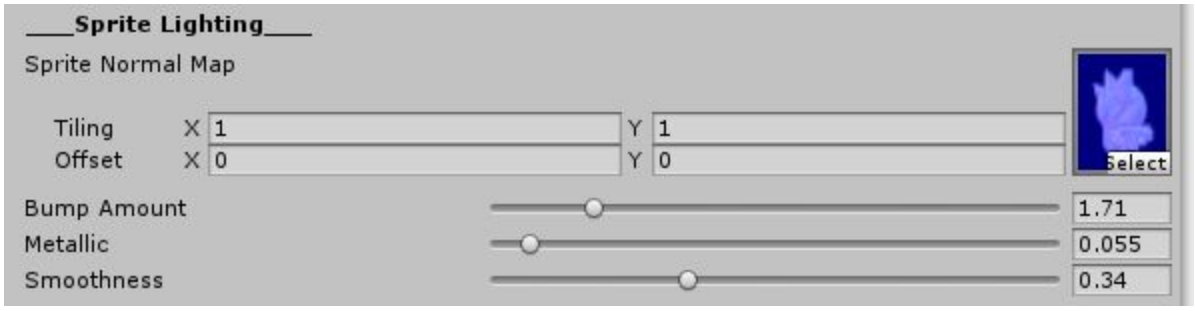
Sprite Lighting was an addition of the current version (1.4) as a user request and it's still in an experimental state. Read this whole section or watch the provided Youtube link in order to learn how it works and its limitations but in case you find any problems or have any feedback, please contact me: seasidegamestudios@gmail.com.

You can find a video tutorial of this feature here:
https://youtu.be/UdowIOCDq4I

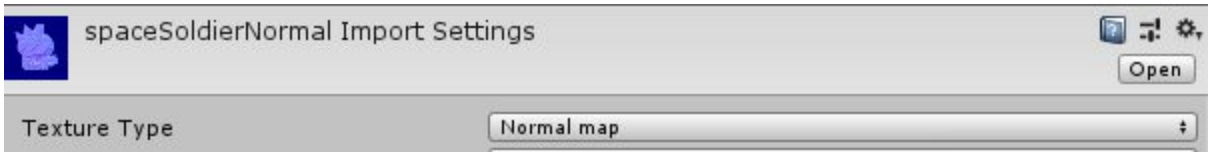
In order to make the light work a lot of boilerplate code had to be added, if I added this to the default shader the performance would decrease even when the lighting is deactivated. For this reason I decided to create a new shader just for lighting. So you won't be able to use the default tool workflow if you want to use this feature. That being said this is how you would set it up:

1. Got to: AllIn1SpriteShader/Materials/SpriteLighting.material
2. Duplicate the SpriteLighting Material by selecting it and pressing Ctrl+D
3. Properly name the copy
4. Assign this new Material to the desired Sprite
5. Modify its properties as wished

When you do so you'll see some different properties on the Material Inspector:



These properties are explained on the Effects and Properties Breakdown section, but for now is relevant to mention that Sprite Lighting supports normal maps. You can create a normal map from a sprite for free on this website: https://cpetry.github.io/NormalMap-Online/ **(invert the Red channel on the generated texture to get the correct map for Unity)**
And remember to set the normal maps textures you import as “Normal Map” in the import settings:



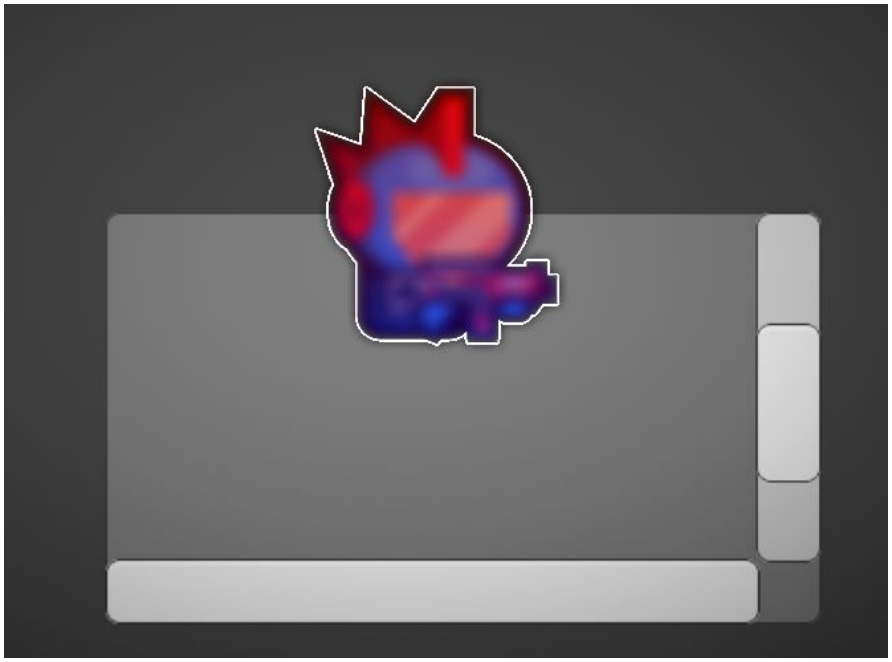
Sprite Lighting limitations:

- It supports only up to 4 of them, if there are more lights some visual artifacts may happen
- Point Lights may have unexpected behaviours, Directional Lights work fine
- This shader is way less performant than the default AllIn1SpriteShader one. And it gets more expensive for the hardware the more lights you have

UI Masking

You can find a video tutorial of this feature here:
https://youtu.be/iRL2sVUhw_0

If you try to use the asset material on a masked UI element such as the ones contained on a Scroll view you'll see that the object won't get masked.



In this image we can see how we have a UI image inside of a Scroll View and it's not getting masked (it should only be visible inside the grey area). This happens because elements that need to be masked have a unique Stencil configuration, this configuration is not included on the default material since it would bring problems to the regular use cases.

For this reason a Material with the proper Stencil configuration for this cases is included. This Material can be found at: AllIn1SpriteShader/Materials/UIStencil.material.

In case you need to use this material I recommend this workflow (the same that for Sprite Lighting):

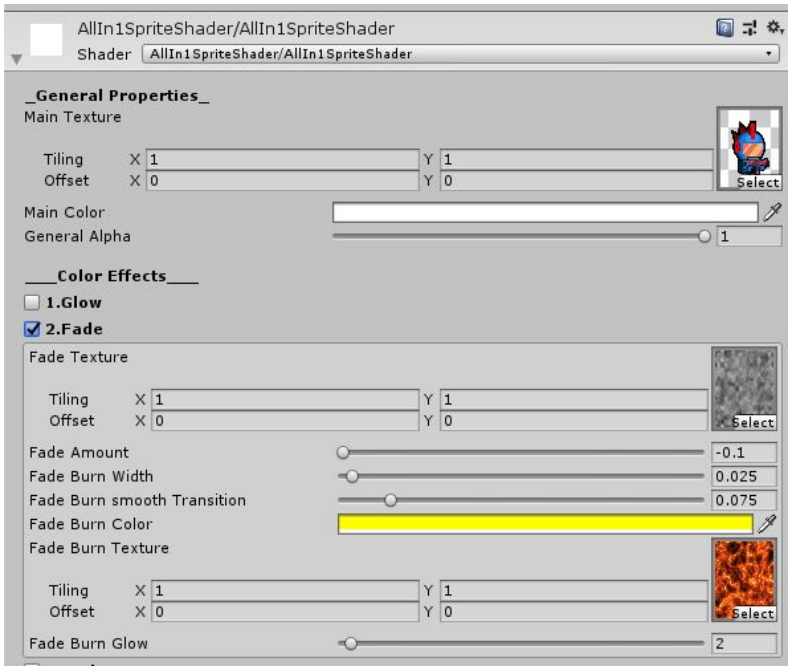
1. Duplicate the UIStencil Material by selecting it and pressing Ctrl+D
2. Properly name the copy
3. Assign this new Material to the desired UI element

If we follow these steps the previous setup should look like this:

Now it’s been properly masked (note that you’ll need to re-apply the Properties of the default material).

Effects and Properties Breakdown

The AllIn1SpriteShader has a custom Material Inspector that allows you to activate and deactivate effects. When an effect is activated it displays it's properties so that they can be modified:



This is the custom Material Inspector. In this image we can see that the Fade effect is activated and all its properties.

Keep in mind that there's an example of every effect on the Demo scene.

Down below all the shader properties are explained. In between [] (ex:[GLOW_ON]) you can find the shader keyword name of each effect (see How to Enable/Disable Effects at Runtime section to see how to use it). In between () (ex: _MainTex) you can find the shader property names in case you want to modify then in a script (see Scripting section).

- General Properties

- Main Texture (_MainTex): Main Texture, supports Tiling and Offset
- Main Color (_Color): The Tint of the the Main Texture
- General Alpha (_Alpha): Transparency of the end result

- Lighting Properties
  - Sprite Normal Map (_NormalMap): Dictates how the light will affect the sprite (https://en.wikipedia.org/wiki/Normal_mapping)
  - Normal Strength (_BumpAmount): Will make the Normal Map more or less pronounced
  - Metallic (_Metallic): How metallic the sprite is
  - Smoothness (_Smoothness): How smooth the sprite is. This property and the previous one dictate the color and shine of the sprite when it's illuminated
  -
- Color Effects

1. Glow (Needs Post Processing Bloom to work as intended) [GLOW_ON]
   a. Glow Color (_GlowColor): Color of the Glow
   b. Glow Intensity (_Glow): Indicates how much the sprite will glow
   c. Glow Texture used? (_GlowTexUsed): When checked shows the following property
   d. Glow Texture (_GlowTex): Acts as a mask. The glow will only be applied where the alpha of this texture is greater than 0
2. Fade [FADE_ON]
   a. Fade Texture (_FadeTex): Maps how the fade will be made. The fade will be made from black to white
   b. Fade Amount (_FadeAmount): How much fade to apply. 0 is no fading and 1 is completely faded
   c. Fade Burn Width (_FadeBurnWidth): Size of the burned edge. Can be set to 0 to have no burned edge
   d. Fade Burn Smooth Transition (_FadeBurnTransition): How sharp the burned edge is

e. Fade Burn Color (_FadeBurnColor): Tint of the burned edge
f. Fade Burn Texture (_FadeBurnTex): Texture of the burned edge
g. Fade Burn Glow (_FadeBurnGlow): How much the burned edge glows (needs Bloom in the scene)

3. Outline (when the width is large it doesn't look good) [OUTBASE_ON]
    a. Outline Base Color (_OutlineColor): Tint of outline color
    b. Outline Base Alpha (_OutlineAlpha): Transparency of the outline
    c. Outline Base Glow (_OutlineGlow): How much the outline glows (needs Bloom in the scene)
    d. Outline Base High Resolution (_Outline8Directions): When toggled the outline has double the resolution and looks smoother on corners. It's more expensive computation wise
    e. Outline Base is Pixel Perfect (_OutlineIsPixel): When toggled the outline width increases pixel by pixel (ideal for pixel art games)
    f. Outline Width (_OutlineWidth/_OutlinePixelWidth): How thick the outline is
    g. Outline uses texture (_OutlineTex): Toggles if the outline has a texture overlay or not (is affected by Outline Base Color)
    h. Outline Texture Scroll Speed (_OutlineTexXSpeed / _OutlineTexYSpeed): Scroll speed of the outline texture in the X axis and Y axis
    i. Outline Texture is Greyscale (_OutlineTexGrey): When toggled the outline texture will be greyscaled
    j. Outline uses distortion (_OutlineDistortToggle): When toggled distortion will be applied to the outline and the outline distortion properties will be shown
    k. Outline Distortion Texture (_OutlineDistortTex): Noise texture that determines how the distortion is done
    l. Outline Distortion Amount (_OutlineDistortAmount): How much is the outline distorted following the distortion texture

m. Outline Distortion Scroll Speed (_OutlineDistortTexXSpeed / _OutlineDistortTexYSpeed): Scroll speed of the distortion texture in the X axis and Y axis

4. Gradient [GRADIENT_ON]
    a. Gradient Blend (_GradBlend): How much of the gradient we show. 0 means the gradient will be fully transparent and 1 means that it will be fully visible
    b. Gradient Colors (_GradTopLeftCol /_GradTopRightCol / _GradBotLeftCol / _GradBotRightCol): The color of each corner of the gradient. Colors will be automatically blended together
5. Color Swap (needs a color swap texture to work) [COLORSWAP_ON]
    a. Color Swap Texture (_ColorSwapTex): This texture must contain pure red, blue and green sections. This sections will then be recolored with whatever color we choose in the following properties
    b. Color Swap Red Channel (_ColorSwapRed): New color of the red parts of the Color Swap texture
    c. Color Swap Red Luminosity (_ColorSwapRedLuminosity): How bright the Red Channel Color will be
    d. The green and blue properties work like the red channel properties
6. Hue Shift [HSV_ON]
    a. Hue Shift (_HsvShift): How much the colors will be shifted
    b. Hue Shift Saturation (_HsvSaturation): Saturation of the hue shift result
    c. Hue Shift Bright (_HsvBright): Brightness of the hue shift result
7. Change 1 Color [CHANGECOLOR_ON]
    a. Tolerance (_ColorChangeTolerance): How similar to the Color to Change we need to be in order to change to the new color
    b. Color to Change (_ColorChangeTarget): This is the color that the shader will look for. The shader will change the pixels that are similar to this color by the New Color

    c. New Color (_ColorChangeNewCol): The result color of the pixels that are similar to Color to change
8. Color Ramp [COLORRAMP_ON]
    a. Color Ramp Texture (_ColorRampTex): This texture will be the new color palette of the sprite
    b. Color Ramp Luminosity (_ColorRampLuminosity): Extra of luminosity used to shift the color palette to where you want
    c. Color Ramp Affects Outline (_ColorRampOutline): When checked the color ramp will also affect the outline
9. Hit Effect [HITEFFECT_ON]
    a. Hit Effect Color (_HitEffectColor): The tint of the effect
    b. Hit Effect Glow (_HitEffectGlow): Glow of the effect. Needs Bloom in the scene
    c. Hit Effect Blend (_HitEffectBlend): How much of the effect is shown. When set to 0 it's not shown, when set to 1 it shows fully. This is meant to be animated to get cool looking results
10. Negative [NEGATIVE_ON]
    a. Negative Amount (_NegativeAmount): How much of the negative effect we want to show
11. Pixelate [PIXELATE_ON]
    a. Pixelate size (_PixelateSize): The lower the number the more pixelated the sprite gets. This effect looks bad with combined with distortions
12. Greyscale [GREYSCALE_ON]
    a. Greyscale Luminosity (_GreyscaleLuminosity): Make sprite whiter
    b. Greyscale Affects Outline (_GreyscaleOutline): When checked the greyscale will also affect the outline
    c. Greyscale Tint Color (_GreyscaleTintColor): Tint of the greyscale
13. Posterize [POSTERIZE_ON]
    a. Posterize Number of Colors (_PosterizeNumColors): The higher the number the more different colors the sprite will display

b. Posterize Amount (_PosterizeGamma): The higher the number the more different the posterize colors will be
c. Posterize Affects Outline (_PosterizeOutline): When checked the posterize will also affect the outline

14. Blur (won't affect the outline) [BLUR_ON]
    a. Blur Intensity (_BlurIntensity): How much the sprite is blurred
    b. Blur is low res (_BlurHD): When active an alternative and less expensive (computation wise) blur version is used
15. Motion Blur [MOTIONBLUR_ON]
    a. Motion Blur Angle (_MotionBlurAngle): The direction of the motion blur
    b. Motion Blur Distance (_MotionBlurDist): The amount of motion blur
16. Ghost [GHOST_ON]
    a. Ghost Color Boost (_GhostColorBoost): How white the ghost effect gets
    b. Ghost Transparency (_GhostTransparency): How transparent the effect gets
17. Inner Outline (places outlines over the Main Texture) [INNEROUTLINE_ON]
    a. Inner Outline Color (_InnerOutlineColor): Color of the Inner Outline
    b. Inner Outline Thickness (_InnerOutlineThickness): How thick the Inner Outline is
    c. Inner Outline Alpha (_InnerOutlineAlpha): How transparent the Inner Outline is
    d. Inner Outline Glow (_InnerOutlineGlow): How much the Inner Outline glows (needs Bloom in the scene)
18. Hologram [HOLOGRAM_ON]
    a. Hologram Stripes Amount (_HologramStripesAmount): How many hologram stripes are displayed
    b. Hologram Stripes Fill (_HologramStripesFill): How tall the stripes are

c. Hologram Stripes Alpha (_HologramFlickerAlpha): Transparency of the stripes
d. Hologram Stripes Speed (_HologramStripesSpeed): How fast the stripes scroll
e. Hologram Stripes Luminosity (_HologramStripesLuminosity): How much the stripes glow

19. Chromatic Aberration [CHROMABERR_ON]
    a. ChromaticAberr Amount (_ChromAberrAmount): How visible the effect is
    b. ChromaticAberr Alpha (_ChromAberrAlpha): How transparent the effect is
20. Glitch [GLITCH_ON]
    a. Glitch Amount (_GlitchAmount): The higher the number, the more intense the effect gets
21. Flicker [FLICKER_ON]
    a. Flicker Percent (_FlickerPercent): The percentage of time the sprite is invisible (from 0 to 1)
    b. Flicker Frequency (_FlickerFreq): How often does the flicker happen
    c. Flicker Alpha (_FlickerAlpha): How transparent the sprite is when it flickers
22. Shadow [SHADOW_ON]
    a. Shadow X Axis (_ShadowX): Shadow position offset on the X axis
    b. Shadow Y Axis (_ShadowY): Shadow position offset on the Y axis
    c. Shadow Alpha (_ShadowAlpha): Transparency of the shadow
    d. Shadow Color (_ShadowColor): Tint of the shadow
23. Alpha Cutoff [ALPHACUTOFF_ON]
    a. Alpha cutoff value (_AlphaCutoffValue): Pixels that are more transparent than this value are not drawn. This is useful to make more cartoon looking effects and to discard unwanted transparencies from certain effects

- UV Effects

24. Hand Drawn [DOODLE_ON]
    a. Hand Drawn Amount (_HandDrawnAmount): How much of a distortion we apply to make it look hand drawn frame a frame
    b. Hand Drawn Speed (_HandDrawnSpeed): How often we distort the sprite
25. Grass Movement / Wind [WIND_ON]
    a. Grass Speed (_GrassSpeed): How fast it moves from side to side
    b. Grass Wind (_GrassWind): How much the sprite bends
    c. Grass is manually animated (_GrassManualToggle): When checked the sprite won't move on its own (useful if you want a sprite to interact with the player)
    d. Grass manual anim (_GrassManualAnim): If the previous property is checked this property dictates how the sprite bends. -1 means fully bent to the left and 1 fully bent to the right
26. Wave [WAVEUV_ON]
    a. Wave Amount (_WaveAmount): How many waves we make
    b. Wave speed (_WaveSpeed): How fast the wave scrolls across the sprite
    c. Wave Strength (_WaveStrength): How much the wave affects the sprite
    d. Wave X Axis (_WaveX): Position of the wave origin on the X axis (0 is left 1 is right)
    e. Wave Y Axis (_WaveY): Position of the wave origin on the Y axis (0 is bottom 1 is top)
27. Round Wave [ROUNDWAVEUV_ON]
    a. Round Wave Strength (_RoundWaveStrength): How much the wave affects the sprite
    b. Round Wave Speed (_RoundWaveSpeed): How fast the wave scrolls across the sprite

28. Rect Size [RECTSIZE_ON]
    a. Rect Size (_RectSize): Size of the mesh where the sprite is drawn. Set Scene to “Shaded Wireframe” shading mode to see the mesh shape
29. Offset [OFFSETUV_ON]
    a. Offset X axis (_OffsetUvX): Offset of the sprite on the X axis
    b. Offset Y axis (_OffsetUvY): Offset of the sprite on the Y axis
30. Clipping (useful for sliders) [CLIPPING_ON]
    a. Clipping Left (_ClipUvLeft): How much of the image we clip from left to right
    b. Clipping Right (_ClipUvRight): How much of the image we clip from right to left
    c. Clipping Up (_ClipUvUp): How much of the image we clip from up to down
    d. Clipping Down (_ClipUvDown): How much of the image we clip from down to up
31. Texture Scroll [TEXTURESCROLL_ON]
    a. Texture Scroll Speed X (_TextureScrollXSpeed): Scrolling speed on the X axis
    b. Texture Scroll Speed Y (_TextureScrollYSpeed): Scrolling speed on the Y axis
32. Zoom [ZOOMUV_ON]
    a. Zoom Amount (_ZoomUvAmount): How much the sprite is zoomed
33. Distortion [DISTORT_ON]
    a. Distortion Texture (_DistortTex): Noise texture that determines how the distortion is done
    b. Distortion Amount (_DistortAmount): How much the image is distorted following the texture pattern
    c. Distortion scroll speed (_DistortAmount): Scroll speed of the distortion texture in the X axis and Y axis
34. Twist [TWISTUV_ON]

a. Twist Amount (_TwistUvAmount): How much is the sprite twisted
b. Twist Pos X Axis (_TwistUvPosX): Position of the center of the twist on the X axis (0 is left and 1 is right)
c. Twist Pos Y Axis (_TwistUvPosY): Position of the center of the twist on the Y axis (0 is bottom and 1 is top)
d. Twist Radius (_TwistUvRadius): The radius of the twist effect

35. Rotate [ROTATEUV_ON]
    a. Rotate Angle (_RotateUvAmount): Indicates in radians the angle of rotation of the sprite texture
36. Polar Coordinates [POLARUV_ON]
    a. Transforms the uv coordinates into polar coordinates (this effect looks goods with tiling on the main texture + texture scrolling)
37. Fish Eye [FISHEYE_ON]
    a. Fish Eye Amount (_FishEyeUvAmount): How much fish eye distortion we want to apply
38. Pinch [PINCH_ON]
    a. Pinch Amount (_PinchUvAmount): How much pinch effect we want to apply
39. Shake [SHAKEUV_ON]
    a. Shake Speed (_ShakeUvSpeed): How fast it shakes
    b. Shake X Multiplier (_ShakeUvX): The higher the value the more it will move on the X axis while shaking
    c. Shake Y Multiplier (_ShakeUvY): The higher the value the more it will move on the Y axis while shaking

Considerations

The shader that the material uses compilations flags to enable and disable the code of the different effects. So only the effects that you enable will be taken into account by the GPU and therefore only those parts will be computed.
The shader code is also fast, uses as less memory as possible and has no conditionals.
That being said, having all these properties in the shader has a slight GPU overhead. But there's nothing to worry about unless your game displays a huge amount of different materials at the same time since Unity will batch instances of the same material into a single draw call.