Chroma Key Kit

ChromaKeyKit delivers a background removal feature with a wide range of chroma key color options. The Asset contains *FragFilter*(*FF*) components that allow you to apply additional fragment shaders to the target texture in your material. You can use the chroma, blur, and mask tools to create a sequence of shaders, each one applied to the result of the previous one. This allows you to achieve the best results possible.

FF CONTROLLER

FFController – sequentially applies *FFComponents* shaders to the original texture. The shaders will be applied in the order in which they are on the *GameObject*. You can assign the source texture through bridge components - inheritors of the *IFFBridge* interface. The controller shows the current bridge in the inspector.

FF COMPONENTS

FFComponent – sets values to it's shader properties. *FFcontroller* will be added automatically when any *FFcomponent* is added to *GameObject*.

FF ChromaKey Alpha

KeyColor – color that will be transparent on the result
DChroma – chroma differense in Color between Key and Source
DChromaT – chroma tolerance
DLuma – luma differense in Color between Key and Source
DLumaT – luma tolerance

FF ChromaKey Bg

KeyColor – color that will be transparent on the result
BgColor – color that will be placed instead of KeyColor
BgTex – texture that will be placed instead of KeyColor
DChroma – chroma differense in Color between Key and Source
DChromaT – chroma tolerance
Chroma – result chroma of color: closer to Source(0) -> closer to Bg(1)
Luma – result luma of color: closer to Source(0) -> closer to Bg(1)
Saturation – result saturation of color: 0(0) -> closer to result chroma(1)
Alpha – result alpha of BgColor

FF Blur

BlurMatrix – spread matrix
BlurOffset – spread by XY (X = Y) used when filtering the texture

FF Mask Source

AlphaPow – pow of alpha value
AlphaEdge – alpha gradient edge

FF Filter HSBC

BaseColor – color multiplier
TintColor – color tint
Hue – color hue (0 -> 360);
Saturation – color saturation
Brightness – color brightness
Contrast – color contrast

TARGET RENDER

TargetRender - is a component for modifying and assigning textures to target render objects, such as *Renderer* materials, *RawImage*, *RenderTexture*.

RenderMode – set the target object for the rendering
SourceMode: Manual – set the source texture manually
SourceMode: From Target – the texture that is currently in the target object will be used as the source texture

FF BRIDGES

To connect *FFController* to texture source, use FFBridge (inheritors of the *IFFBridge* interface). You can create your own bridges, using the methods to get the texture from your source, and the FFController's methods:

SetSourceTexture(Texture t) – use when the texture instance reference changes;
RenderIn() – use for one render iteration into own texture and return it;
RenderOut(RenderTexture rt) – use for one render iteration into rt;

FF TargetRender

FFTargetRender - a general class for modifying textures using the *FFController*, without extra render update events.

FF VideoPlayerRender / FF BridgeVideoPlayer

These 2 components have similar functionality. Since *VideoPlayer* already has its own render implementation (via the *RenderMode* property), you only need to use *FFBridgeVideoPlayer* to connect it with the *FFController*. But you can also use *TargetRender*'s successor implementation – *FFVideoPlayerRender*. In this case, set the *RenderMode* property of *VideoPlayer* to *APIOnly*.

FF WebCamRender

WebCamRender is a rendering implementation for *WebCamTexture*.
FFWebCamRender - connection with *FFController*.

FF AVPLiveCameraRender / FF BridgeAVPLiveCamera

As with *VideoPlayer*, these components have similar functionality. See demo scenes with usage examples. To get the bridge files unpack / import:
Assets/Nexweron/ChromaKeyKit/ChromaKeyAVProLiveCamera/Package/ChromaKeyAVProLiveCamera.unitypackage

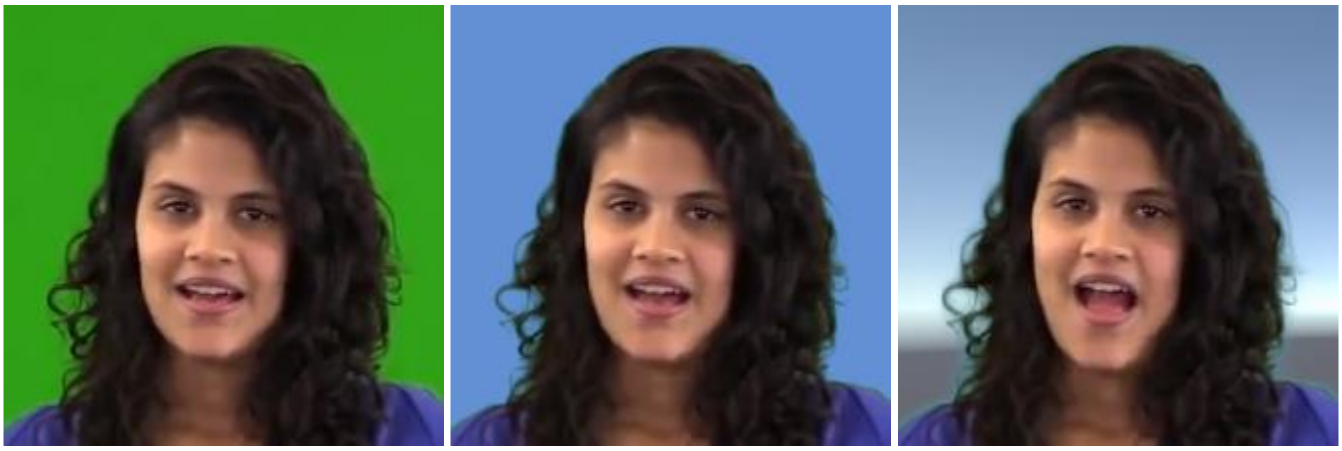
USAGE

- remove the previous version if it has already been imported
- import the asset package into your project
- if you are using the Universal Render Pipeline (URP), you need to import:
  Assets/Nexweron/ChromaKeyKit/ChromaKeyShaders/URP/Package/ChromaKeyShadersURP.unitypackage
  Assets/Nexweron/ChromaKeyKit/ChromaKeyFragFilter/URP/Package/ChromaKeyFragFilterURP.unitypackage

SHADER ONLY

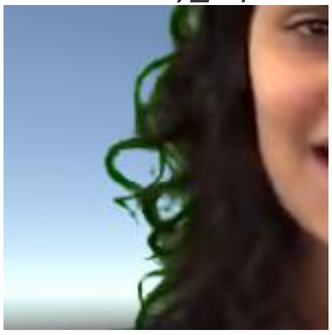
The fastest way is to use one of ChromaKey_Alpha shaders to material. The Asset contains several shaders with different implementations of surface settings. And also *ShaderGraph* versions for customization in *URP.*

More options and border color settings can be configured using ChromaKey_Bg. As opposed to ChromaKey_Alpha, the bg components change the color of the key to the color of the background. This is good for images with transparent objects, small details, etc:

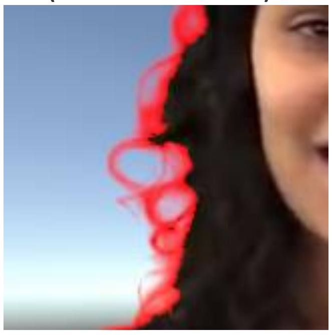


If after removing *KeyColor* the pixel's transparency < 1, it is treated as border and its chroma can be changed in the settings to be more consistent with the background

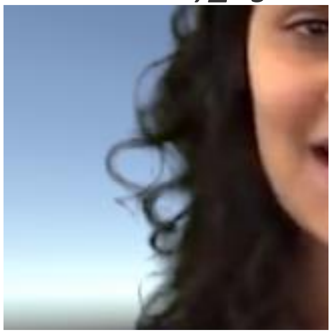
ChromaKey_Alpha:



(border color)



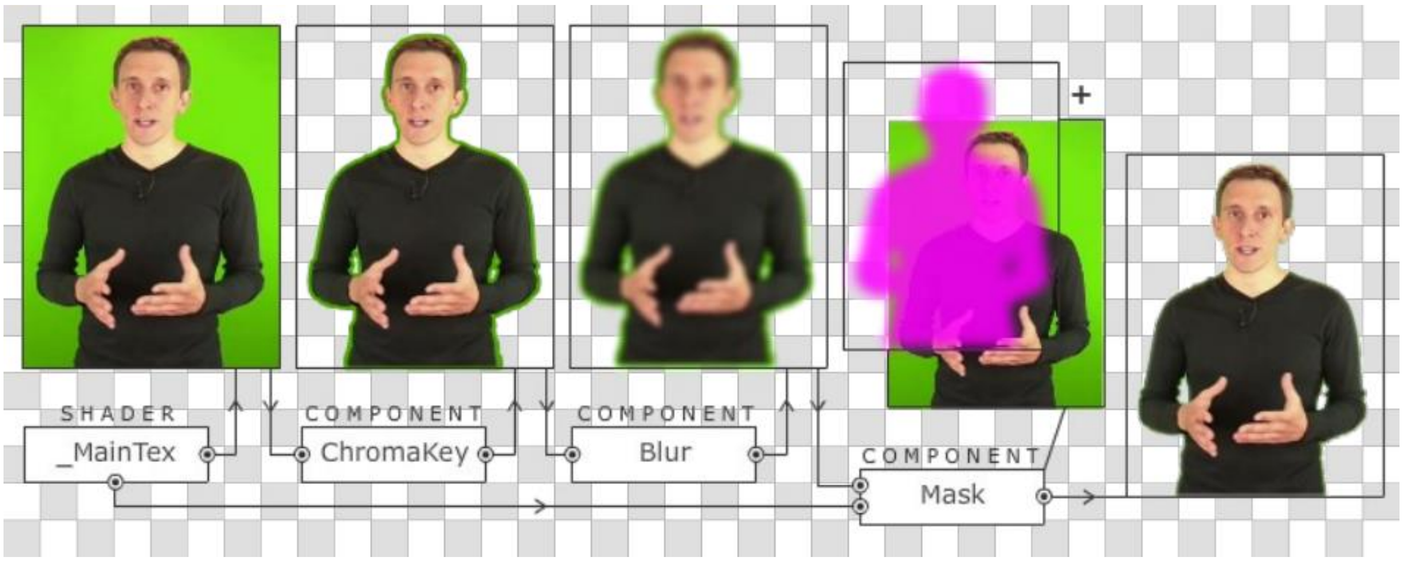
ChromaKey_Bg:

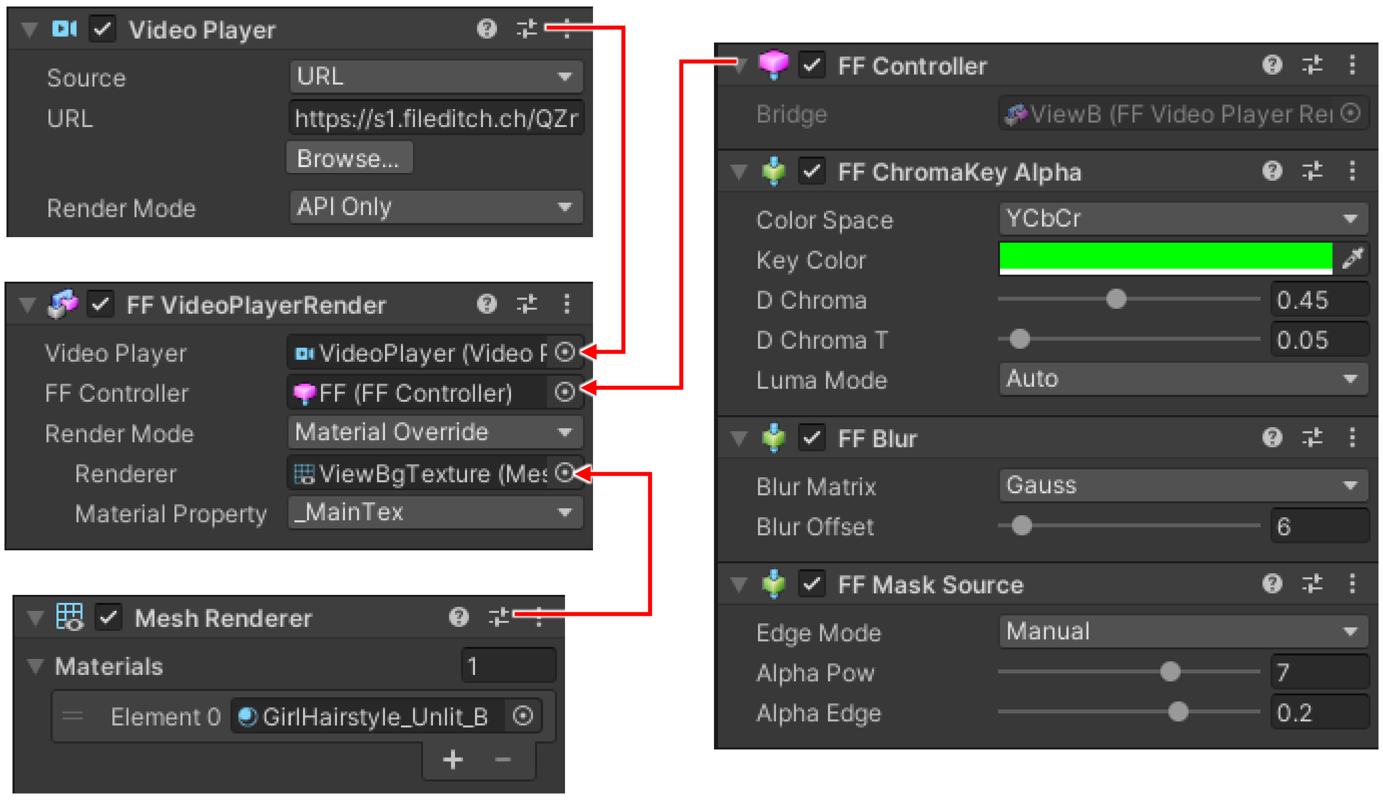


BEST QUALITY

The best result can be achieved by using a sequence of FF components:

ChromaKey → Blur → MaskSource





Video Player

Source	URL
URL	https://s1.fileditch.ch/QZr
	Browse...
Render Mode	API Only

FF VideoPlayerRender

Video Player	VideoPlayer (Video F
FF Controller	FF (FF Controller)
Render Mode	Material Override
Renderer	ViewBgTexture (Mes
Material Property	_MainTex

Mesh Renderer

Materials	1
Element 0	GirlHairstyle_Unlit_B

FF Controller

Bridge	ViewB (FF Video Player Rer

FF ChromaKey Alpha

Color Space	YCbCr
Key Color	🟩
D Chroma	0.45
D Chroma T	0.05
Luma Mode	Auto

FF Blur

Blur Matrix	Gauss
Blur Offset	6

FF Mask Source

Edge Mode	Manual
Alpha Pow	7
Alpha Edge	0.2

NOTE

If you use SRP - don't convert built-in shaders unnecessarily, use shaders for URP instead from *ChromaKeyShadersURP.unitypackage*

SUPPORT

If you have any comments, questions, or issues, please email me at **nexweron@gmail.com**