UI Particle System

by
Marcin Olszewski
(MODev)

March 24, 2018

1 About UI Particle System

UI Particle System is advanced solution for 3D particles used in Unity GUI. Probably everyone who tried to add particles to GUI had problems with making the same type of particles going around buttons or culling inside scroll views, but with this plugin you will not need to worry about changing sorting orders, adding multiple canvases, etc. With it you can just add next particles to GUI in few quick steps.

2 Where can be used

UI Particle System can be used with Unity 2017.2+ (it may also work on previous versions but this wasn't tested). Shaders and depth buffer are prepared to be working on PC as well as on mobile devices.

3 How to use it

- First you should add **UIParticleCanvas** component to your GUI canvas (which will contain particles, gui buttons, etc.). GUI Canvas needs to have render mode set to *Screen Space - Camera* or *World Space* and set *Render Camera* parameter.

- Now in UIParticleCanvas component set Mask layer parameter wich will be used to generate depth buffer (best would be to create new layer for this case). If you have multiple rendered canvases in one

place, you will need to create more layers.

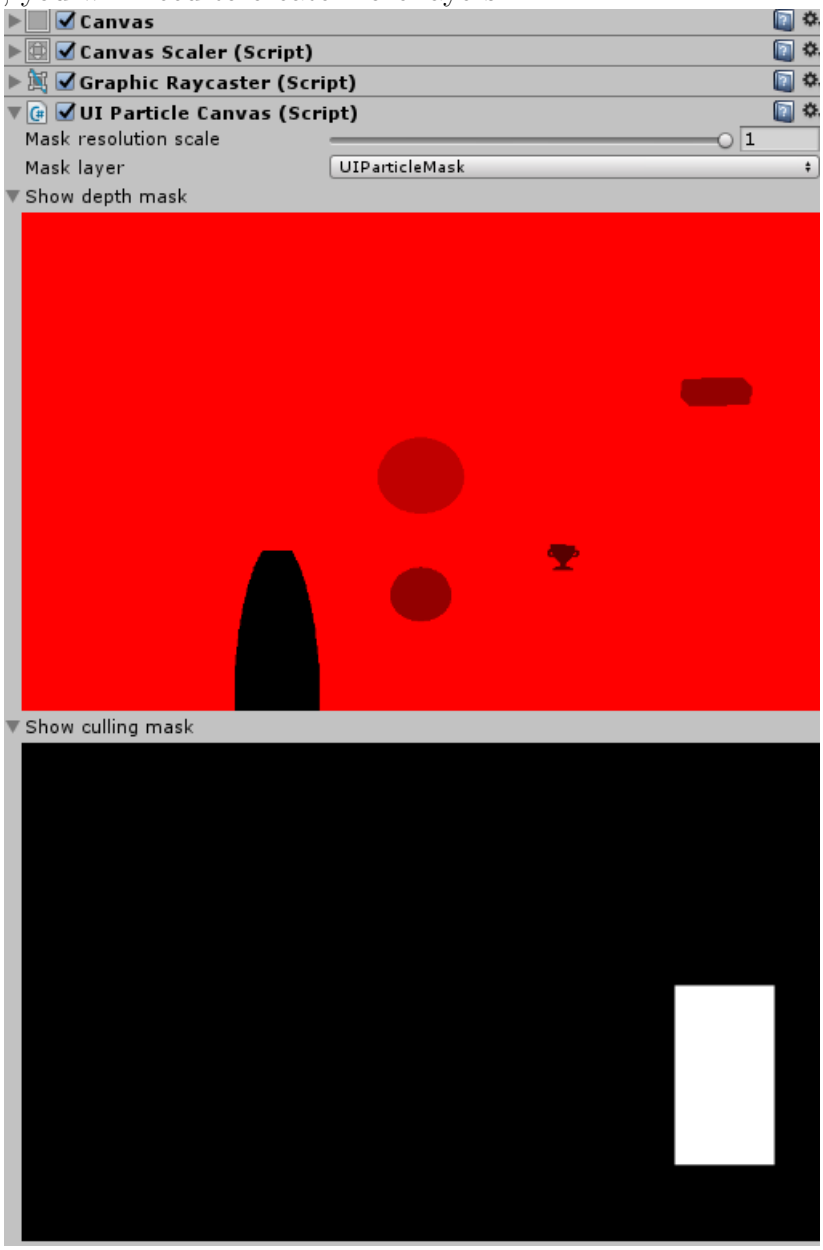

Figure 1: UIParticleCanvas

- You can set up / view **UIParticleCanvas** parameters like this:
  - **Mask resolution scale** - it defines resolution of mask. 1 is 100% of screen size, 0.5 is 50%. Usually 1 is fine in most cases but if you fill it costs too much memory you can reduce it till masking will look badly

- **Mask layer** - it defines on which layer depth/mask buffer will be rendered
- **Show depth mask** - allows check how your depth mask looks like
- **Show culling mask** - allows check how your culling mask looks like. When you set culling mask layer values, best is to set them from 255 to 1 because they can be easy visible on this view

- Next step is to add to every button/image/rect **UIParticleDepthObject** which will make this object visible for our particles in depth (or culling mask) buffer. Like on screens: 2 or 3.
  Please remember that Z position is now important and objects must be in front of Canvas (so if Canvas Z position is 0, than depth of objects should be lower than 0). Recomended is also not to make too big differences in Z position between depth objects because of limited depth buffer size.

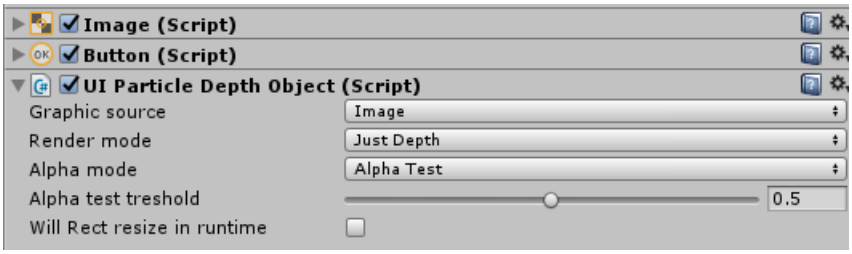

Figure 2: Example of setup to render Image into depth buffer

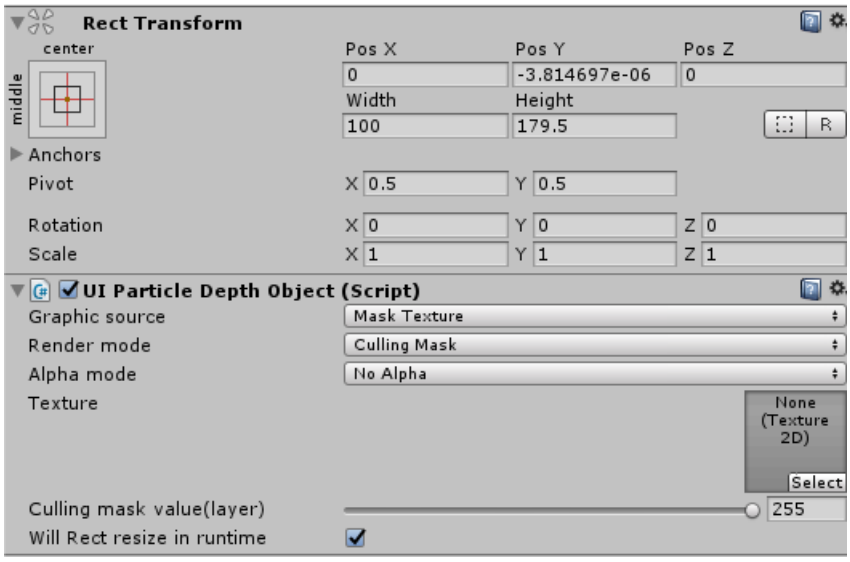

Figure 3: Example of setup to render just filled rect into culling mask buffer

- You can set up **UIParticleDepthObject** parameters like this:
  - **Graphic source** - allows select what source will be used to render into buffer. You can select:
    - Image (gui component). Mask in this case works as you would select Image type "Simple"
    - RawImage (gui component)
    - Texture

    Of course if you choose component, you will need to have it attached to game object too
  - **Render mode** - here you can chose if you want to render just to depth buffer or make this object as culling mask for particles
  - **Alpha mode** - defines how alpha in texture will be treated: as AlphaTested (so just cut under certain treshold), as Dithering (so dependly on alpha value some pixels will be visible some not, it can be good solution for semi transparent masks) and as NoAlpha (it ignores alpha in texture but in case of tight Sprites from Image component it can have mesh dependent shape)
  - **Alpha test treshold** - available in Alpha Test mode only. Defines under what value pixels will be cut in buffer

- **Alpha dithering steps** - available in Alpha Dithering mode only. Defines how many dithering density steps should be done. Higher value not always means better view
- **Culling mask value (layer)** - available in Culling Mask mode only. Allows select what layer value will it have
- **Will Rect resize in runtime** - select this if you are going to resize RectTransform in runtime. It's not enabled by defult because it woudl require in every frame to check if vertices of rect has been changed

- Final step is changing shader in particle's material to **UIParticle** (MODev ⇒ UIParticle ⇒ Particle).

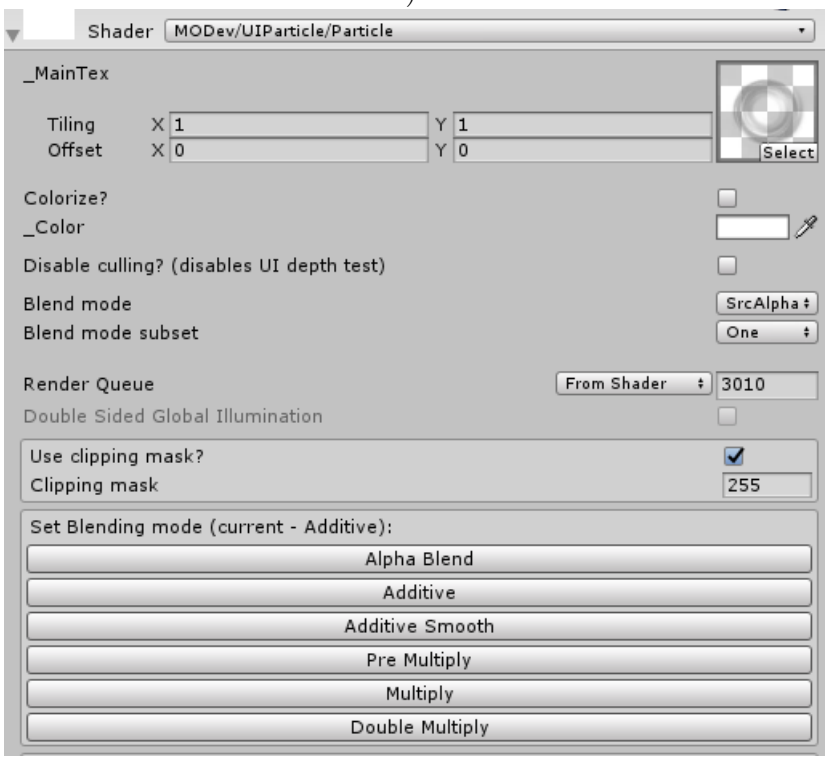

Figure 4: Example of particle material setup (with culling masking enabled)

- In **UIParticle** shader you can change parameters like:
  - **_MainTex** - select texture of particles
  - **Colorize?** - enables/disables coloring by material parameter. Vertex (Particle System) colorizing is always enabled

- **_Color** - when Colorize is enabled, multiplies value by texture color
- **Disable culling?** - enables/disables depth/mask culling. If you select this, particle will behave like without UI Particle System
- **Blend mode (subset)** - with this you can set your custom alpha blending mode. If you would like to set one of basic alpha blending modes you can use one of bottom buttons in "Set Blending mode" preview
- **UseClippingMask?** - enables/disables culling by clipping mask (Culling Mask mode in UIParticleDepthObject)
- **_ClippingMaskVal** - allows select culling mask layer. Please note that if you change this value from code, you need to divide layer mask value by 255f
- **Set Blending mode buttons** - allows select one of basic alpha blending modes
- **Enable Soft particle** - enables Soft Particles, which will better blend with colliding UI elements. Please note that this technique is a bit more expensive so use it only when it's needed
- **Soft particle factor** - defines how smooth soft particle blending is

Figure 5: Difference between missing and enabled soft particles

4 Semi-transparency of culling

UI Particle System works close to Z-test in 3D so it's binary (something passes test: it's visible, if not: it's culled). But theres solution which is frequently used in deffered shading: dithering alpha blending. You can achieve

this with UIParticleDepthObject and alpha mode set to Dithering + proper alpha dithering steps count set up.

5 Distortion effect

Since 1.06 version plugin allows to add distortion effect to GUI with *MODev* ⇒ *UIParticle* ⇒ *ParticleWithDistortion* shader instead of just Particle shader. Please note that it uses GrabPass so may be inefficient for mobile devices (rather low-end, than high-end). Distortion effect is controled by parameters in material and particle alpha. Recomended is to use alpha blending mode for transparency blending mode.

6 Make Particle objects follow UI RectTransform size

To make Particle object follow RectTransform's size you should add to GameObject *UIParticleRectTransSizeFollower* component. When object will be set properly on scene by you, use *Make rect snapshot* button to save Rect referenced setup. Please note that if you would like to make your object follow middle of RectTransform, middle of GameObject should also be on middle of it (at least in XY axis).

7 Contact

If you need any support or you would like to share with me what do you think about this tool feel free to write on support@modev.com.pl . Also if you need some additional parameters which can be used by many people write to me too - maybe I'll make this and put it into next UI Particle System update.

It would be great if you could rate my asset on Asset Store (if you like it). This will be good sign for me that tool is needed and should be still expanded.

8 Useful links

All of those links you can find in Asset Store but heres copy of them:
Example how to set up scene: `https://www.youtube.com/watch?v=8uuxMpxJqUw`