

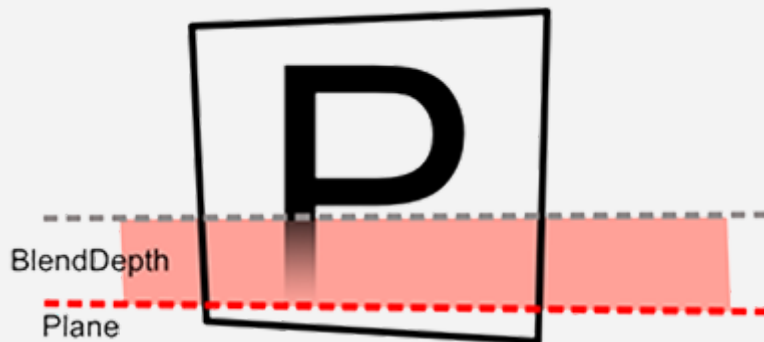
FAST SOFTPARTICLE

VisualWorks



The Fast Softparticle Package might make a soft edge on the particle where the hard-edge occurs. (aka Soft-particles)

KEY IDEA



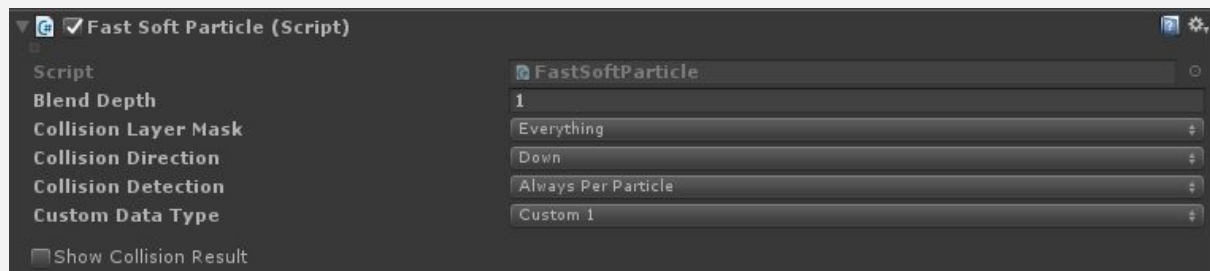
Detect one of the bottom planes of the particle.
Fade alpha based on distance from the plane.

KEY FEATURES

- NO G-buffer is used.
 - Mobile friendly
 - Forward rendering-pipeline friendly
- It uses vertex-streams for fade information per particle.
- Collision detection for fading is computed in a scene.
 - Per Particle-playing
 - Per a particle-system
 - Per Particle
- The changed shader is small and too cheap.

HOW TO USE

1. Import the package
2. Select a particle system that has a hard edge on the ground. It is not necessary to apply to all particle systems.
3. Attach a FastSoftParticle Component.



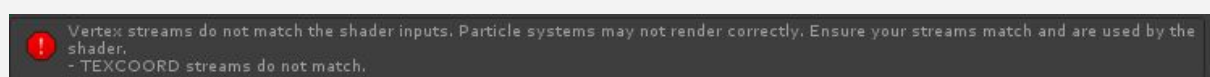
4. Change a shader to a built-in shader in the package. If it is the built-in shader, you can switch by pressing the "Switch Shader to Fast SoftParticle Shader" button.

This package is provided by modifying the shader below. If the shader is your own, please refer to the next section.

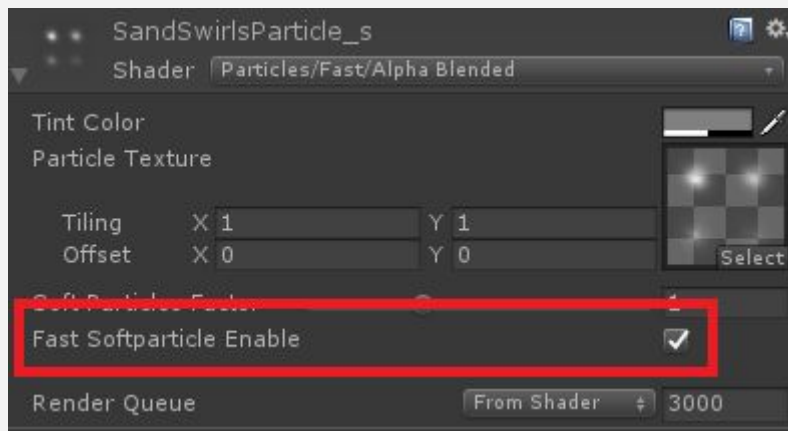
Particles/Additive Legacy Shaders/Particles/Additive	Particles/Fast/Additive
Particles/~Additive-Multiply Legacy Shaders/Particles/~Additive-Multiply	Particles/Fast/~Additive-Multiply
Particles/Additive (Soft)	Particles/Fast/Additive (Soft)
Particles/Alpha Blended	Particles/Fast/Alpha Blended
Particles/Anim Alpha Blended	Particles/Fast/Anim Alpha Blended *
Particles/Blend	Particles/Fast/Blend
Particles/Multiply	Particles/Fast/Multiply
Particles/Multiply (Double)	Particles/Fast/Multiply (Double)
Particles/Alpha Blended Premultiply	Particles/Fast/Alpha Blended Premultiply

* DataType = Custom2 (Texcoord2)

If this is completed, the below warning disappears.



Then turn on the Fast Softparticle Enable



5. Set options

- **Blend Depth : Blending range (world distance)**
- **Collision Layer Mask**
- **Collision Direction**
- **Collision Detection Type**
 - **Only Once** : Detect a common plane only once at the start (Fast, Recommended for flat geometry)
 - **Always** : Detect a common plane every frame
 - **Always Per Particle** : Detect planes for each particles every frame
- **Show Collision Result in editor**

MODIFIED BUILT-IN PARTICLE SHADER SAMPLE

```
// Unity built-in shader source. Copyright (c) 2016 Unity Technologies. MIT license (see
license.txt)

Shader "Particles/Fast/Additive" {
Properties {
    _TintColor ("Tint Color", Color) = (0.5,0.5,0.5,0.5)
    _MainTex ("Particle Texture", 2D) = "white" {}
    _InvFade ("Soft Particles Factor", Range(0.01,3.0)) = 1.0
    [Toggle(FASTSOFTPARTICLE_ON)] FASTSOFTPARTICLE_ON("Fast Softparticle Enable", Float) = 0
}

Category {
    Tags { "Queue"="Transparent" "IgnoreProjector"="True" "RenderType"="Transparent"
    "PreviewType"="Plane" }
    Blend SrcAlpha One
    ColorMask RGB
    Cull Off Lighting Off ZWrite Off

    SubShader {
        Pass {

            CGPROGRAM
            #pragma vertex vert
            #pragma fragment frag
            #pragma target 2.0
            // #pragma multi_compile_particles
            #pragma multi_compile __ SOFTPARTICLES_ON FASTSOFTPARTICLE_ON
            #pragma multi_compile_fog

            #include "UnityCG.cginc"
            #ifndef UNITY_DECLARE_DEPTH_TEXTURE
```

```

#define UNITY_DECLARE_DEPTH_TEXTURE(A) sampler2D_float A
#endif

sampler2D _MainTex;
fixed4 _TintColor;

struct appdata_t {
    float4 vertex : POSITION;
    fixed4 color : COLOR;
    float2 texcoord : TEXCOORD0;
#ifdef FASTSOFTPARTICLE_ON
    float4 plane : TEXCOORD1;
#endif
    UNITY_VERTEX_INPUT_INSTANCE_ID
};

struct v2f {
    float4 vertex : SV_POSITION;
    fixed4 color : COLOR;
    float2 texcoord : TEXCOORD0;
    UNITY_FOG_COORDS(1)
#ifdef SOFTPARTICLES_ON
    float4 projPos : TEXCOORD2;
#endif
#ifdef FASTSOFTPARTICLE_ON
    float depth : TEXCOORD2;
#endif
    UNITY_VERTEX_OUTPUT_STEREO
};

float4 _MainTex_ST;

v2f vert (appdata_t v)
{
    v2f o;
    UNITY_SETUP_INSTANCE_ID(v);
    UNITY_INITIALIZE_VERTEX_OUTPUT_STEREO(o);
    o.vertex = UnityObjectToClipPos(v.vertex);
#ifdef SOFTPARTICLES_ON
    o.projPos = ComputeScreenPos (o.vertex);
    COMPUTE_EYEDEPTH(o.projPos.z);
#endif
#ifdef FASTSOFTPARTICLE_ON
    float3 worldPos = mul(unity_ObjectToWorld, v.vertex).xyz;
    o.depth = dot(v.plane, float4(worldPos, 1));
#endif
    o.color = v.color;
    o.texcoord = TRANSFORM_TEX(v.texcoord, _MainTex);
    UNITY_TRANSFER_FOG(o,o.vertex);
    return o;
}

UNITY_DECLARE_DEPTH_TEXTURE(_CameraDepthTexture);
float _InvFade;

fixed4 frag (v2f i) : SV_Target
{
    #ifdef SOFTPARTICLES_ON
        float sceneZ = LinearEyeDepth (SAMPLE_DEPTH_TEXTURE_PROJ(_CameraDepthTexture,
UNITY_PROJ_COORD(i.projPos)));
        float partZ = i.projPos.z;
        float fade = saturate (_InvFade * (sceneZ-partZ));
        i.color.a *= fade;
    #endif
#ifdef FASTSOFTPARTICLE_ON
    i.color.a *= saturate(i.depth);
#endif
    fixed4 col = 2.0f * i.color * _TintColor * tex2D(_MainTex, i.texcoord);
    UNITY_APPLY_FOG_COLOR(i.fogCoord, col, fixed4(0,0,0,0)); // fog towards black due
to our blend mode
    return col;
}

ENDCG

```

```

    }
}
}
}

```

MODIFIED STANDARD PARTICLE SHADERS SAMPLE

```

Shader "Custom/SurfaceShader_VC-S" {
    Properties{
        _Color("Color", Color) = (1,1,1,1)
        _MainTex("Albedo (RGB)", 2D) = "white" {}
        _Normal("Normap Map", 2D) = "bump" {}
        [Toggle(FASTSOFTPARTICLE_ON)] FASTSOFTPARTICLE_ON("Fast Softparticle Enable", Float) = 0
    }
    SubShader{
        Tags{ "Queue" = "Transparent" "RenderType" = "Transparent" }
        LOD 200
        Blend One OneMinusSrcAlpha

        CGPROGRAM
            // Physically based Standard lighting model, and enable shadows on all light types
            #pragma surface surf Standard fullforwardshadows vertex:vert alpha:fade
            #pragma multi_compile _ FASTSOFTPARTICLE_ON

            // Use shader model 3.0 target, to get nicer looking lighting
            #pragma target 3.0

            sampler2D _MainTex;
            sampler2D _Normal;

            struct Input {
                float2 uv_MainTex;
                float4 vertex : SV_POSITION;
                float4 color : COLOR;
                #ifdef FASTSOFTPARTICLE_ON
                float depth;
                #endif
            };

            void vert(inout appdata_full v, out Input o)
            {
                UNITY_INITIALIZE_OUTPUT(Input, o);
                o.color = v.color;
                #ifdef FASTSOFTPARTICLE_ON
                float3 worldPos = mul(unity_ObjectToWorld, v.vertex).xyz;
                o.depth = dot(v.texcoord1, float4(worldPos, 1));
                #endif
            }

            fixed4 _Color;

            void surf(Input IN, inout SurfaceOutputStandard o) {
                // Albedo comes from a texture tinted by color
                fixed4 c = tex2D(_MainTex, IN.uv_MainTex) * _Color;
                o.Albedo = c.rgb*IN.color;
                o.Normal = UnpackNormal(tex2D(_Normal, IN.uv_MainTex));
                o.Alpha = c.a*IN.color.a;
                #ifdef FASTSOFTPARTICLE_ON
                o.Alpha *= saturate(IN.depth);
                #endif
            }
            ENDCG
        }
        FallBack "Diffuse"
    }
}

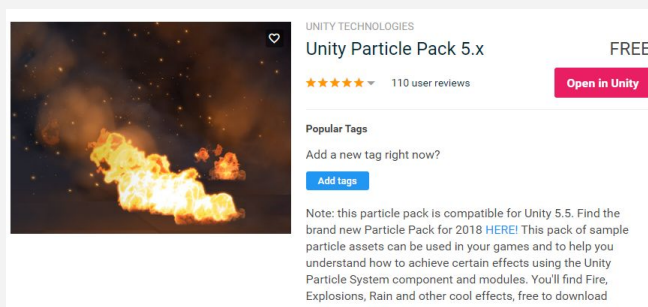
```

OVERVIEW VIDEO



https://www.youtube.com/watch?v=I9ooX_gafd8

RESOURCES



Unity Particle Pack 5.x by Unity Technologies

<https://assetstore.unity.com/packages/essentials/asset-packs/unity-particle-pack-5-x-73777>

Thanks for downloading “Fast Softparticle”.

Also, take a look at our other assets.

<https://assetstore.unity.com/publishers/40160>

If you have any questions, suggestions or feedback, please feel free to contact me at ksi@softnette.com