Limitations

EEVEE's goal is to be an interactive render engine. Some features may not be there yet or may be impossible to implement into EEVEE's architecture without compromising performance.

Here is a rather exhaustive list of all the limitations you can expect while working with EEVEE.

Attributes and Properties

- Only 14 attributes from Geometry Nodes are supported in a material
- Only 8 custom object properties are supported in a material

Cameras

- Only perspective and orthographic projections are currently supported.

Lights

- Lights can only have one color and do not support light node trees.
- Unlike in Cycles, the Size of spot lights does not change the softness of the cone.
- The area light Beam spread option is not supported.

Light Probes

- EEVEE supports up to 128 active light probe spheres.
- EEVEE supports up to 16 active light probe planes inside the view frustum.
- Active light probe volumes must fit inside the Light Probes Volume Memory Pool.

Indirect Lighting

- Light probe capture does not support specular reflections. Specular energy is treated as diffuse.

Shadows

- *Shadow Map Raytracing* can produce light leaking because of overlapping shadow casters. This can be mitigated by using lower step count, enablin jitter, or reducing the light shape size.
- Thin objects (e.g. walls without thickness) might have light leaking on the shadowed side. This can be mitigated by making the object have some thickness or lowering Resolution Limit.

Volumetrics

- Only single scattering is supported.
- Volumetrics are rendered only for the camera "rays". They don't appear in reflections/refractions and probes.
- Volumetric shadowing only work in volumetrics. They won't cast shadows onto solid objects in the scene.
- Volumetric shadowing only work for volumes inside the view frustum.

Depth of Field

- Blended materials cannot be correctly handled by the post-processing blur, but will be correctly handled by the sample-based method. For this, you need to disable the post-process depth of field by setting the *Max Size* to 0.

Screen Space Effects

Ray-triangle intersection is not currently supported. Instead of this, EEVEE uses the depth buffer as an approximated scene representation. This reduces the complexity of scene scale effects and enables a higher performance. However, only what is in inside the view can be considered when computing the effects. Also, since it only uses one layer of depth, only the front-most pixel distance is known.

These limitations creates a few problems:

- The screen space effects disappear when reaching the screen border. This can be partially fixed by using the *overscan* feature.
- Screen space effects lack deep information (or the thickness of objects). This is why most effects have a thickness parameter to control how to consider potential intersected pixels.
- Objects behind other objects (occluded) are not considered by these effects.
- Blended surfaces are not considered by these effects. They are not part of the depth prepass and do not appear in the depth buffer.
- Objects that a part of Holdout Collections will not be rendered with screen space effects.

Raytracing

- Blended materials and materials using raytrace refractions will not appear in dithered materials reflections.
- Blended materials are not compatible with raytracing.
- Only one refraction event is correctly modeled. An approximation of the second refraction event can be achieved using the Thickness workflow.
- Only dithered materials *not* using Raytrace Refractions can be refracted.

Shader Nodes

- All BSDF's are using approximations to achieve realtime performance so there will always be small differences between Cycles and EEVEE.
- Some utility nodes are not yet compatible with EEVEE.
- Certain combinations of BSDF's will result in more noise than others. This is the case when mixing Diffuse BSDF and Refraction BSDF.
- Displacement of flat shaded surfaces will split the mesh into triangles. See Displacement for a workaround.

See also

For a full list of unsupported nodes see Nodes Support.

Memory Management

In EEVEE, GPU Memory management is done by the GPU driver. In theory, only the needed textures and meshes (now referred as "the resources") fo single draw call (i.e. one object) needs to fit into the GPU memory.

So if the scene is really heavy, the driver will swap things in and out to make sure all objects are rendered correctly.

In practice, using too much GPU memory can make the GPU driver crash, freeze, or kill the application. So be careful of what you ask.

There is no standard way of estimating if the resources will fit into the GPU memory and/or if the GPU will render them successfully.

CPU Rendering

Being a rasterization engine, EEVEE only uses the power of the GPU to render. There is no plan to support CPU (software) rendering as it would be ve inefficient. CPU power is still needed to handle high scene complexity as the geometry must be prepared by the CPU before rendering each frame.

Multiple GPU Support

There is currently no support for multiple GPU systems.

Headless Rendering

Headless rendering is not supported on headless Windows systems.