Skip to content

GPU Rendering

GPU rendering makes it possible to use your graphics card for rendering, instead of the CPU. This can speed up rendering because modern GPUs are designed to do quite a lot of number crunching. On the other hand, they also have some limitations in rendering complex scenes, due to more limited memory, and issues with interactivity when using the same graphics card for display and rendering.

To enable GPU rendering, go into the Preferences ‣ System ‣ Cycles Render Devices, and select either *CUDA*, *OptiX*, *HIP*, *oneAPI*, or *Metal*. Next, you must configure each scene to use GPU rendering in Properties ‣ Render ‣ Device.

Rendering Technologies

Blender supports different technologies to render on the GPU depending on the particular GPU manufacturer and operating system.

CUDA – NVIDIA

CUDA is supported on Windows and Linux and requires a NVIDIA graphics cards with compute capability 3.0 and higher. To make sure your GPU is supported, see the list of NVIDIA graphics cards with the compute capabilities and supported graphics cards.

OptiX – NVIDIA

OptiX is supported on Windows and Linux and requires a NVIDIA graphics cards with compute capability 5.0 and higher and a driver version of at lea 495.89. To make sure your GPU is supported, see the list of NVIDIA graphics cards.

OptiX takes advantage of hardware ray-tracing acceleration in RTX graphics cards, for improved performance.

GPU acceleration for OpenImageDenoise is available for compute capability 7.0 and higher, which includes all NVIDIA RTX cards.

HIP – AMD

HIP is supported on Windows and Linux and requires a AMD graphics card with the RDNA1 architecture or newer. Both discrete GPUs and APUs ar supported.

Supported GPUs include:

- Radeon RX 5000 Series
- Radeon RX 6000 Series
- Radeon RX 7000 Series
- Radeon RX 9000 series
- Radeon Pro W6000 Series
- Radeon Pro W7000 Series

Minimum driver versions:

- Windows: Radeon Software 24.6.1 or Radeon PRO Software 24.Q2
- Linux: Radeon Software 23.40 or ROCm 6.0

Please refer to AMD's website for more information about AMD graphics cards and their architectures.

Hardware ray-tracing support is available with the most recent drivers. This can be enabled in the preferences, and is supported on Radeon RX 6000 ar newer.

GPU accelerated denoising is available on discrete Radeon RX 6000 and Radeon RX 7000 GPUs.

Shadow caustics are not supported on Radeon RX 5000 or Radeon RX 9000 GPUs.

oneAPI – Intel

oneAPI is a computation library that is supported on Windows and Linux and requires a Intel® Arc™ graphics card with the Xe HPG architecture. Hardware acceleration for ray-tracing and denoising is supported.

Supported GPUs include:

- Intel® Arc™ A-Series
- Intel® Arc™ B-Series

Minimum driver versions:

- Windows: Intel Graphics Driver XX.X.101.5518
- Linux: `intel-level-zero-gpu` package 1.3.27642, typically available through the `intel-compute-runtime` package XX.XX.27642.38

Please refer to Intel's website for more information about Intel graphics cards and their architectures.

GPU accelerated denoising is available on all supported GPUs.

Metal – Apple (macOS)

Metal is supported on Apple computers with Apple Silicon. macOS 13.0 or newer is required to support all features.

GPU accelerated denoising is available on Apple Silicon.

Limitations

- Path Guiding is not supported on any GPU.
- Open Shading Language is only supported for OptiX, with some limitations listed in the documentation.

Frequently Asked Questions

Why is Blender unresponsive during rendering?

On older GPU generations, graphics cards can only either render or draw the user interface. This can make Blender unresponsive while it is rendering. Heavy scenes can also make Blender unresponsive on newer GPUs, when using a lot of memory or executing expensive shaders, however this is genera less of a problem.

The only complete solution for this is to use a dedicated GPU for rendering, and another for display.

Why does a scene that renders on the CPU not render on the GPU?

There may be multiple causes, but the most common one is that there is not enough memory on your graphics card. Typically, the GPU can only use the amount of memory that is on the GPU (see Would multiple GPUs increase available memory? for more information). This is usually much smaller than th amount of system memory the CPU can access. With CUDA, OptiX, HIP and Metal devices, if the GPU memory is full Blender will automatically try to use system memory. This has a performance impact, but will usually still result in a faster render than using CPU rendering.

Can multiple GPUs be used for rendering?

Yes, go to Preferences ‣ System ‣ Compute Device Panel, and configure it as you desire.

Would multiple GPUs increase available memory?

Typically, no, each GPU can only access its own memory.

The exception is NVIDIA GPUs connected with NVLink, where multiple GPUs can share memory at a small performance cost. This can be enabled wi Distributed Memory Across Devices in the preferences.

What renders faster?

This varies depending on the hardware used. Different technologies also have different compute times depending on the scene tested. For the most up to date information on the performance of different devices, browse the Blender Open Data resource.

Error Messages

In case of problems, be sure to install the official graphics drivers from the GPU manufacturers website, or through the package manager on Linux. The graphics drivers provided by the computer manufacturer can sometimes be outdated or incomplete.

Error: Out of memory

This usually means there is not enough memory to store the scene for use by the GPU.

> Note
>
> One way to reduce memory usage is by using smaller resolution textures. For example, 8k, 4k, 2k, and 1k image textures take up respectively 256MB 64MB, 16MB and 4MB of memory.

The NVIDIA OpenGL driver lost connection with the display driver

If a GPU is used for both display and rendering, Windows has a limit on the time the GPU can do render computations. If you have a particularly heavy scene, Cycles can take up too much GPU time. Reducing Tile Size in the Performance panel may alleviate the issue, but the only real solution is to use separate graphics cards for display and rendering.

Another solution can be to increase the time-out, although this will make the user interface less responsive when rendering heavy scenes. Learn More Here.

Unsupported GNU version

On Linux, depending on your GCC version you might get this error. See the NVIDIA CUDA Installation Guide for Linux for a list of supported GCC versions. There are two possible solutions to this error:

Use an alternate compiler

If you have an older GCC installed that is compatible with the installed CUDA toolkit version, then you can use it instead of the default compiler. This is done by setting the `CYCLES_CUDA_EXTRA_CFLAGS` environment variable when starting Blender.

Launch Blender from the command line as follows:

```
CYCLES_CUDA_EXTRA_CFLAGS="-ccbin gcc-x.x" blender
```

(Substitute the name or path of the compatible GCC compiler).

Remove compatibility checks

If the above is unsuccessful, delete the following line in `/usr/local/cuda/include/host_config.h`:

```
#error -- unsupported GNU version! gcc x.x and up are not supported!
```

This will allow Cycles to successfully compile the CUDA rendering kernel the first time it attempts to use your GPU for rendering. Once the kerne is built successfully, you can launch Blender as you normally would and the CUDA kernel will still be used for rendering.

CUDA Error: Kernel compilation failed

This error may happen if you have a new NVIDIA graphics card that is not yet supported by the Blender version and CUDA toolkit you have installed. this case Blender may try to dynamically build a kernel for your graphics card and fail.

In this case you can:

1. Check if the latest Blender version (official or experimental builds) supports your graphics card.
2. If you build Blender yourself, try to download and install a newer CUDA developer toolkit.

Normally users do not need to install the CUDA toolkit as Blender comes with precompiled kernels.

Last updated on 2025-05-10