Graphics Cards

Most graphical user interfaces these days are hardware accelerated, so the appropriate graphics driver will be needed for optimal performance and a smooth desktop experience. Additionally, these drivers provide 3D acceleration and hardware video decoding/encoding capabilities.

The Linux graphics stack consists of several components, but the main component is the `mesa` package.

Manufacturer	OpenGL	Vulkan	Video acceleration
Intel	`mesa`	`vulkan-intel`	`intel-media-driver`, `libva-intel-driver`
AMD	`mesa`	`vulkan-radeon`	`libva-mesa-driver`
NVIDIA	`mesa`, `nvidia`	`nvidia-utils`	`libva-mesa-driver`, `nvidia-utils`

Intel

For Intel integrated graphics and Intel Arc, install the following packages:

```
pacman -S mesa vulkan-intel intel-media-driver libva-intel-driver
```

AMDGPU

For AMD integrated and dedicated graphics, install the following packages:

```
pacman -S mesa libva-mesa-driver vulkan-radeon
```

NVIDIA

In the case of NVIDIA, there's the option to either use the open source Nouveau drivers, or the Linux kernel modules provided by NVIDIA themselves.

If you have a relatively recent NVIDIA card, it is generally recommended to go with the official NVIDIA drivers. For older cards (GeForce 8xx or 9xx series or older) you should choose the Nouveau driver.

Nouveau open source driver

The Nouveau driver is included with `mesa`:

```
pacman -S mesa libva-mesa-driver
```

Additionally, NVIDIA cards after the "Tesla" line of GPUs (GeForce 8xxx, 9xxx) will need additional firmware files installed. Without these firmware files, the GPU will be stuck at the lowest performance level, because dynamic reclocking and power management of the graphics processor will not be available.

```
yay -S nouveau-fw
```

Proprietary driver

When using any NVIDIA graphics card after the "Maxwell" line of GPUs (GeForce GTX 9xx), the proprietary NVIDIA kernel module will provide the best performance for intensive graphic and video processing workloads.

NVIDIA provides two options for their GPU drivers: a closed and open kernel module.

For anything more recent than RTX 2xxx cards, recommends the `nvidia-open` kernel module, as they plan to support that one more long-term.

```
pacman -S nvidia-open nvidia-utils
```

For earlier GPUs (GTX 9xx, GTX 10xx) the closed `nvidia` driver remains available.

```
pacman -S nvidia nvidia-utils
```

Early KMS

In order to enable early KMS (Kernel mode switching) with the prorprietary NVIDIA driver, you will need to take additional steps.

The kernel modules of the proprietary driver need to be included explicitly in the `MODULES` array of your `/ets/mkinitcpio.conf` file (or a drop-in config file, e.g. `/etc/mkinitcpio.conf.d/modules.conf`):

```
MODULES=(nvidia nvidia_modeset nvidia_uvm nvidia_drm)
```

Additionally, remove the `kms` hook from the `HOOKS` array. This is to prevent the unintentional loading of the `nouveau` kernel module, which will conflict with the proprietary driver.

Enable Kernel Mode Setting

Since `nvidia-ultis` version 560.35.03-5, Kernel Mode Setting (KMS) is enabled by default with NVIDIA proprietary drivers. However, when using an older version or a very old card with proprietary drivers, KMS must be explicitly enabled through a kernel command line argument at boot time, otherwise Wayland compositors may not function properly.

```
nvidia_drm.modeset=1
```

> **NOTE:** Refer to Boot Loader for how to add the parameter to your boot configuration.

To verify that kernel mode setting is enabled (in the installed system) query the sysfs info with the following command:

```
cat /sys/module/nvidia_drm/parameters/modeset
```

`Y` means Kernel Mode Setting was enabled on boot.
`N` means Kernel Mode Setting was **not** enabled on boot.

Revision #9
Created 2022-02-11 19:37:01 UTC by Sebin
Updated 2025-09-20 11:55:06 UTC by Sebin