Introduction to GPUs and to the Linux Graphics Stack

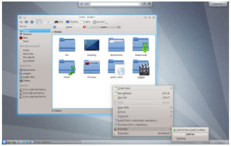
Martin Peres

Nouveau developer
Ph.D. student at LaBRI

November 26, 2012

Outline

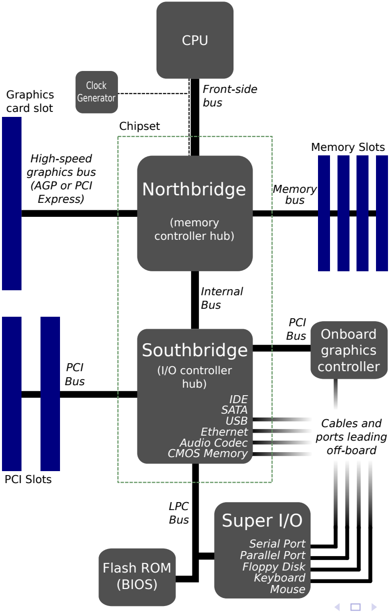
1. I - Hardware : Anatomy of a GPU
   - General overview
   - Driving screens
   - Host < − > GPU communication
2. II - Host : The Linux graphics stack
   - General overview
   - DRM and libdrm
   - Mesa
   - X11
   - Wayland
   - X11 vs Wayland
3. Attributions
   - Attributions

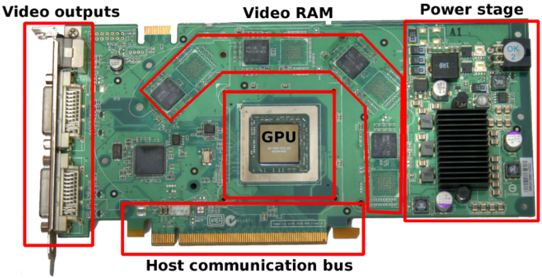


General overview of a modern GPU's functions

- Display content on a screen
- Accelerate 2D operations
- Accelerate 3D operations
- Decode videos
- Accelerate scientific calculations

General overview



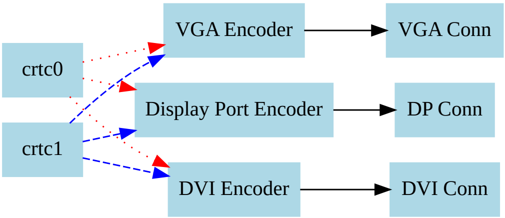


Source: http://www.flickr.com/photos/stefan_ledwina/557505323

Hardware architecture

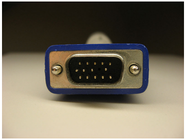
- GPU: Where all the calculations are made
- VRAM: Stores the textures or general purpose data
- Video Outputs: Connects to the screen(s)
- Power stage: Lower the voltage, regulate current
- Host communication bus: Communication with the CPU

Outline



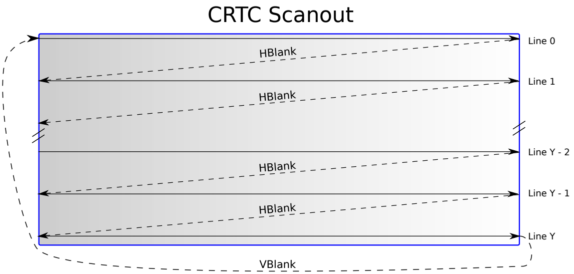
Driving screens : the big picture

- Framebuffer: The image to be displayed on the screen(VRAM)
- CRTC: Streams the framebuffer following the screen's timings
- Encoder: Convert the CRTC's output to the right PHY signal
- Connector: The actual connector where the screen is plugged



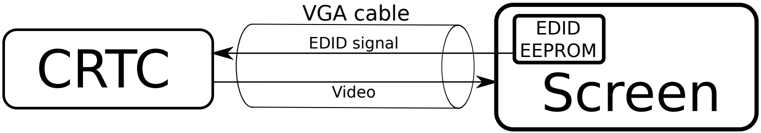

Screen connectors

- VGA: Video, introduced in 1987 by IBM
- DVI: Video, introduced in 1999 by DDWG
- DP: Video & Audio, introduced in 2006 by VESA
- HDMI: Video & Audio, introduced in 1999 by HDMI Founders



Driving screens : the CRT Controller

- Streams the framebuffer following the screen's timings
- After each line, the CRTC must wait for the CRT to go back to the beginning of the next line (Horizontal Blank)
- After each frame, the CRTC must wait for the CRT to go back to the first line (Vertical Blank)
- Timings are met by programming the CRTC clock using PLLs



Configuring the CRTC : Extended display identification data

- Stored in each connector of the screen (small EEPROM)
- Is usually accessed via a dedicated I2C line in the connector
- Holds the modes supported by the screen connector
- Processed by the host driver and exposed with the tool xrandr (see `xrandr --verbose`)

Example: Some display standards

- 1981 : Monochrome Display Adapter (MDA)
  - text-only
  - monochrome
  - 720 * 350 px or 80*25 characters (50Hz)
- 1981 : Color Graphics Adapter (CGA)
  - text & graphics
  - 4 bits (16 colours)
  - 320 * 200 px (60 Hz)
- 1987 : Video Graphics Array (VGA)
  - text & graphics
  - 4 bits (16 colours) or 8 bits (256 colours)
  - 320*200px or 640*480px ($<=$ 70 Hz)

Outline

Modern host communication busses

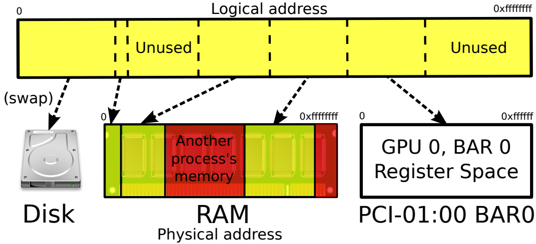
- 1993 : Peripheral Component Interconnect (PCI)
  - 32 bit & 33.33 MHz
  - Maximum transfer rate: 133 MB/s
- 1996 : Accelerated Graphics Port (AGP)
  - 32 bit & 66.66 MHz
  - Maximum transfer rate: 266 to 2133 MB/s (1x to 8x)
- 2004 : PCI Express (PCIe)
  - 1 lane: 0.25 – > 2 GB/s (PCIe v1.x – > 4.0)
  - up to 32 lanes (up to 64 GB/s)
  - Improve device-to-device communication (no arbitration)

Features

- Several generic configuration address spaces (BAR)
- Interruption RQuest (IRQ)

Programming the GPU : Register access via MMIO

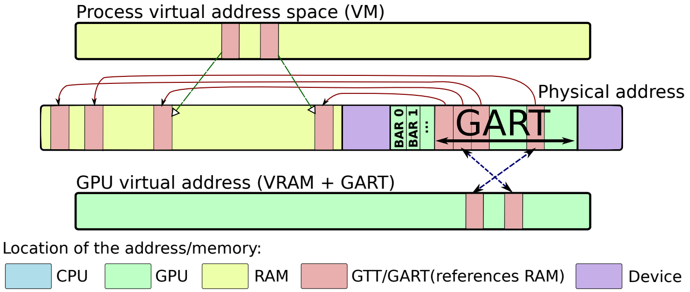
- A GPU's configuration is mostly stored in registers;
- A register is usually identified by an address in a BAR;
- We can then access them like memory;
- This is called Memory-Mapped Input/Output (MMIO).



Example of a CPU process's virtual memory space

GTT/GART

Providing the GPU with easy access to the Host RAM



GART as a CPU-GPU buffer-sharing mechanism

A program can export buffers to the GPU:

- Without actually copying data (faster!);
- Allow the GPU to read textures & data from the program;

Outline

The GPU needs the host for:

- Setting the screen mode/resolution (mode setting);
- Configuring the engines and communication busses;
- Handling power management;
  - Thermal management (fan, react to overheating/power);
  - Change the GPU's frequencies/voltage to save power;
- Processing data:
  - Allocate processing contexts (GPU VM + context ID);
  - Upload textures or scientific data;
  - Send commands to be executed in a context.

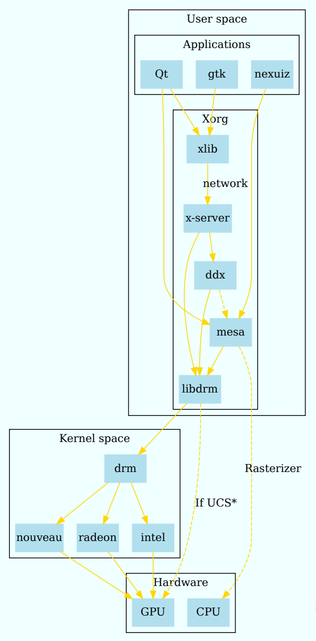
Overview of the components of a graphics stack

- A GPU with its screen;
- One or several input devices (mouse, keyboard);
- A windowing system (such as the X-Server and Wayland);
- Accelerated-rendering protocols (such as OpenGL);
- Graphical applications (such as Firefox or a 3D game).

Components of the Linux Graphics stack

- Direct Rendering Manager (DRM) : exports GPU primitives;
- X-Server/Wayland : provide a windowing system;
- Mesa : provides advanced acceleration APIs;

General overview



Outline

Direct Rendering Manager

- Inits and configures the GPU;
- Performs Kernel Mode Setting (KMS);
- Exports privileged GPU primitives:
  - Create context + VM allocation;
  - Command submission;
  - VRAM memory management: GEM & TTM;
  - Buffer-sharing: GEM & DMA-Buf;
- Implementation is driver-dependent.

libDRM

- Wraps the DRM interface into a usable API;
- Is meant to be only used by Mesa & the DDX;

Outline

Mesa

- Provides advanced acceleration APIs:
  - 3D acceleration: OpenGL / Direct3D
  - Video acceleration: XVMC, VAAPI, VDPAU
- Mostly device-dependent (requires many drivers);
- Divided between mesa classics and gallium 3D;

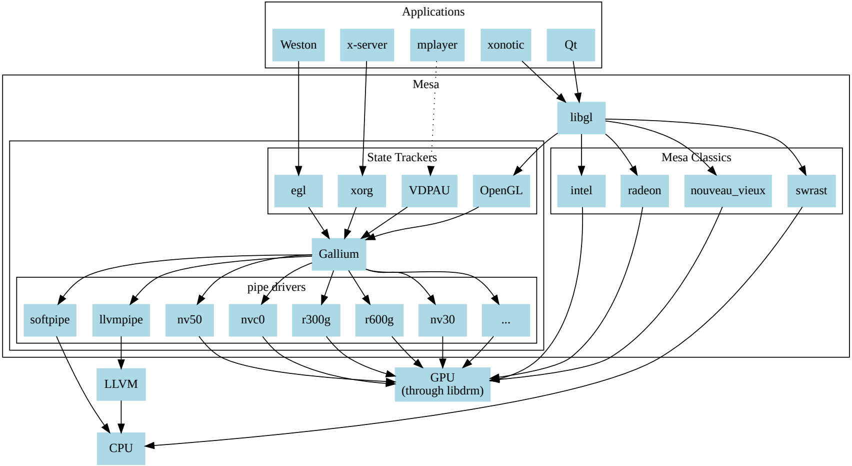
Mesa classics

- Old code-base, mostly used by drivers for old cards;
- No code sharing between drivers, provide only OpenGL;

Gallium 3D

- Built for code-sharing between drivers (State Trackers);
- Pipe drivers follow the instructions from the Gallium interface;
- Pipe drivers are the device-dependent part of Gallium3D;

Mesa


Applications
Weston
x-server
mplayer
xonotic
Qt
Mesa
libgl
State Trackers
egl
xorg
VDPAU
OpenGL
Mesa Classics
intel
radeon
nouveau_vieux
swrast
Gallium
pipe drivers
softpipe
llvmpipe
nv50
nvc0
r300g
r600g
nv30
...
LLVM
GPU
(through libdrm)
CPU

Outline

X11 and the X-Server

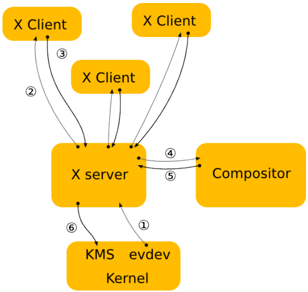
- X11 is a remote rendering API that is 25 years old;
- Exports drawing primitives like filled circles, lines;
- Is extensible via extensions: eg. DRI2, composite, AIGLX.

The X-Server

- Implements the X11 protocol and provides extensions;
- Needs a window manager to display windows (like compiz);
- Holds 2D acceleration drivers (DDX): nouveau, radeon, intel;
- Logs in /var/log/Xorg.0.log (check them for errors).

The X Resize, Rotate and Reflect Extension (XRandR)

- Common X API to configure screens and multi head;
- Implemented by the open and proprietary drivers;



Reaction to an input event

- 1: The kernel driver evdev sends an event to the X-Server;
- 2: The X-Server forwards it to the window with the focus;
- 3: The client updates its window and tell the X-Server;
- 4 & 5: The X-Server lets the compositor update its view;
- 6: The X-Server sends the new buffer to the GPU.

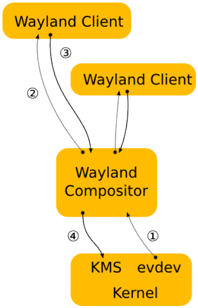
Outline

Wayland

- Protocol started in 2008 by Kristian Høgsberg;
- Aims to address some of X11 shortcomings;
- Wayland manages:
  - Input events: Send input events to the right application;
  - Copy/Paste & Drag'n'Drop;
  - Window buffer sharing (the image representing the window);

Wayland Compositor

- Implements the server side of the Wayland protocol;
- Talks to Wayland clients and to the driver for compositing;
- The reference implementation is called Weston.



Reaction to an input event

- 1: The kernel driver evdev sends an input event to “Weston”;
- 2: “Weston” forwards the event to the right Wayland client;
- 3: The client updates its window and send it to “Weston”;
- 4: Weston updates its view and send it to the GPU.

Outline

X11 vs Wayland

- Rendering protocol vs compositing API:
  - X11 provides old primitives to get 2D acceleration (such as plain circle, rectangle, ...);
  - Wayland let applications render their buffers how they want;
- Complex & heavy-weight vs minimal & efficient:
  - X11 is full of old and useless functions that are hard to implement;
  - Wayland is minimal and only cares about efficient buffer sharing;
- Cannot realistically be made secure vs secureable protocol.

X11 : Security

- X doesn't care about security and cannot be fixed:
    - Confidentiality: X applications can spy other applications;
    - Integrity: X applications can modify other apps' buffers;
    - Availability: X applications can grab input and be fullscreen.
- An X app can get hold of your credentials or bank accounts!
- An X app can make you believe you are using SSL in Firefox!

Wayland : Security

- Wayland is secure if using a secure buffer-sharing mechanism;
- See https://lwn.net/Articles/517375/.

Outline

1. I - Hardware : Anatomy of a GPU
   - General overview
   - Driving screens
   - Host < – > GPU communication
2. II - Host : The Linux graphics stack
   - General overview
   - DRM and libdrm
   - Mesa
   - X11
   - Wayland
   - X11 vs Wayland
3. Attributions
   - Attributions

Attributions : Anatomy of a GPU

- Moxfyre: https://en.wikipedia.org/wiki/File:Motherboard_diagram.svg
- Boffy b: https://en.wikipedia.org/wiki/File:IBM_PC_5150.jpg
- Katsuki: https://fr.wikipedia.org/wiki/Fichier:VGA_plug.jpg
- Evan-Amos: https://fr.wikipedia.org/wiki/Fichier:Dvi-cable.jpg
- Evan-Amos: https://en.wikipedia.org/wiki/File:HDMI-Connector.jpg
- Andreas -horn- Hornig: https://en.wikipedia.org/wiki/File:Refresh_scan.jpg
- Own work: https://en.wikipedia.org/wiki/File:Virtual_memory.svg

Attributions : Host : The Linux graphics stack

- X.org community: X.org schematic
- Kristian Høgsberg: `http://wayland.freedesktop.org/`