



# Understanding the Linux Graphics Stack training

On-site training, 2 days

Latest update: May 17, 2024

Title	Understanding the Linux Graphics Stack training
Training objectives	<ul style="list-style-type: none"><li>• Be able to understand the basics of graphics display: image and color representation, pixel drawing, pixel operations.</li><li>• Be able to understand graphics hardware: display pipeline components, display and rendering hardware.</li><li>• Have a solid understanding of the Linux kernel graphics stack components and role: TTY, framebuffer and DRM subsystems.</li><li>• Have a solid understanding of the Linux user-space graphics stack components and role: DRM from user-space, X.org, Wayland, OpenGL.</li></ul>
Duration	Two days - 16 hours (8 hours per day)
Pedagogics	<ul style="list-style-type: none"><li>• Lectures delivered by the trainer: 75% of the duration</li><li>• Practical demonstrations done by the trainer: 25% of the duration</li><li>• Electronic copies of presentations, lab instructions and data files. They are freely available at <a href="https://bootlin.com/doc/training/graphics">https://bootlin.com/doc/training/graphics</a>.</li></ul>
Trainer	One of the engineers listed on: <a href="https://bootlin.com/training/trainers/">https://bootlin.com/training/trainers/</a>
Language	Oral lectures: English, French. Materials: English.
Audience	People developing multimedia devices using the Linux kernel



<b>Prerequisites</b>	<ul style="list-style-type: none"><li>• <b>Solid experience with the C programming language:</b> participants must be familiar with the usage of complex data types and structures, pointers, function pointers, and the C pre-processor.</li><li>• <b>Experience with low-level development in Linux and hardware interfaces:</b> participants should have a minimal understanding of memory management, interaction with common hardware interfaces (registers, interrupts), the interaction between Linux user-space applications and the Linux kernel (system calls). Following Bootlin's <i>Linux kernel driver development</i> course at <a href="https://bootlin.com/training/kernel/">bootlin.com/training/kernel/</a> allows to fulfill this pre-requisite.</li><li>• <b>Minimal English language level: B1</b>, according to the <i>Common European Framework of References for Languages</i>, for our sessions in English. See <a href="https://bootlin.com/pub/training/cefr-grid.pdf">bootlin.com/pub/training/cefr-grid.pdf</a> for self-evaluation.</li></ul>
<b>Required equipment</b>	<p><b>For on-site sessions at our customer location, the customer must provide:</b></p> <ul style="list-style-type: none"><li>• Video projector</li><li>• Large monitor</li><li>• Drawing board</li></ul>
<b>Certificate</b>	Only the participants who have attended all training sessions, and who have scored over 50% of correct answers at the final evaluation will receive a training certificate from Bootlin.
<b>Disabilities</b>	Participants with disabilities who have special needs are invited to contact us at <a href="mailto:training@bootlin.com">training@bootlin.com</a> to discuss adaptations to the training course.



## Day 1 - Morning

### Lecture - Image and Color Representation

- Light, pixels and pictures
- Sampling, frequency domain, aliasing
- Color quantization and representation
- Colorspaces and channels, alpha
- YUV and chroma sub-sampling
- Pixel data planes, scan order
- Pixel formats, FourCC codes, modifiers

*Introducing the basic notions used for representing color images in graphics.*

### Lecture - Pixel Drawing

- Accessing and iterating over pixel data
- Concepts about rasterization
- Rectangle drawing
- Linear gradient drawing
- Disk drawing
- Circular gradient drawing
- Line drawing
- Line and shape aliasing, sub-pixel drawing
- Circles and polar coordinates
- Parametric curves

*Presenting how to access pixel data in memory and draw basic shapes.*

### Lecture - Pixel Operations

- Region copy
- Alpha blending
- Color-keying
- Scaling and interpolation
- Linear filtering and convolution
- Blur filters
- Dithering

*Providing basic notions about filtering, with very common examples of how it's used.*

### Demo - Drawing and operations

- Examples of various shapes and region drawing
- Examples of basic pixel operations

*Illustrating the concepts presented along the way.*



## Day 1 - Afternoon

### Lecture - Pipeline Components Overview and Generalities

- Types of graphics hardware implementations
- Graphics memory and buffers
- Graphics pipelines
- Display, render and video hardware overview

*Presenting the hardware involved in graphics pipelines.*

### Lecture - Display hardware

- Visual display technologies: CRT, plasma, LCD, OLED, EPD
- Display timings, modes and EDID
- Display interfaces: VGA, DVI, HDMI, DP, LVDS, DSI, DP
- Bridges and transcoders

*Presenting the inner workings of display hardware.*

### Lecture - Rendering Hardware Specifics

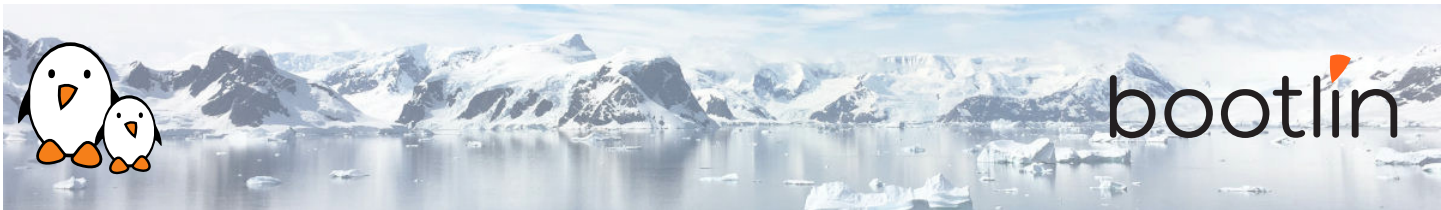
- Digital Signal Processors (DSPs)
- Dedicated hardware accelerators
- Graphics Processing Unit (GPUs)

*Describing the architecture of processing and rendering hardware.*

### Lecture - System Integration, Memory and Performance

- Graphics integration and memory
- Shared graphics memory access
- Graphics memory constraints and performance
- Offloading graphics to hardware
- Graphics performance tips

*Topics related to graphics integration, memory management and performance aspects.*



## Day 2 - Morning

### Lecture - Display Stack Overview

- System-agnostic overview: kernel, userspace display and rendering
- Linux kernel overview
- Linux-compatible low-level userspace overview
- X Window and Wayland overview
- High-level graphics libraries and desktop environments overview

*Presenting what software components are required for modern computer graphics and how they are divided between kernel and userspace.*

### Lecture - TTY Kernel Aspects, Framebuffer Device Kernel Aspects

- Linux TTY subsystem introduction
- Virtual terminals and graphics
- Virtual terminals switching and graphics
- Fbdev overview
- Fbdev basic operations
- Fbdev limitations

*How TTYs interact with graphics in Linux along with a short presentation of fbdev and why it's deprecated.*

### Lecture - DRM Kernel Aspects

- DRM devices
- DRM driver identification and capabilities
- DRM master, magic and authentication
- DRM memory management
- DRM KMS dumb buffer API
- DRM FourCCs and modifiers
- DRM KMS resources probing
- DRM KMS modes
- DRM KMS framebuffer management
- DRM KMS legacy configuration and page flipping
- DRM event notification
- DRM KMS object properties
- DRM KMS atomic
- DRM render
- DRM Prime zero-copy memory sharing (dma-buf)
- DRM sync object fencing
- DRM debug and documentation

*An exhaustive presentation of the DRM interface.*

### Demo - Kernel Aspects

- Linux TTY and virtual terminals
- DRM KMS mode-setting
- DRM KMS driver walkthrough
- DRM render driver walkthrough

*Illustrating how kernel aspects work.*



## Day 2 - Afternoon

### Lecture - X Window Userspace Aspects

- X11 protocol and architecture
- X11 protocol extensions
- Xorg architecture and acceleration
- Xorg drivers overview
- X11 and OpenGL acceleration: GLX and DRI2
- Xorg usage, integration and configuration
- Major issues with X11
- Xorg debug and documentation

*Presenting all things related to X11 and Xorg.*

### Lecture - Wayland Userspace Aspects

- Wayland overview and paradigm
- Wayland protocol and architecture
- Wayland core protocol details
- Wayland extra protocols
- Wayland asynchronous interface
- Wayland OpenGL integration
- Wayland status and adoption
- Wayland debug and documentation

*An in-depth presentation of Wayland.*

### Lecture - Mesa 3D Userspace Aspects

- Standardized 3D rendering APIs: OpenGL, OpenGL ES, EGL and Vulkan
- Mesa 3D overview
- Mesa 3D implementation highlights
- Mesa 3D internals: Gallium 3D
- Mesa 3D internals: intermediate representations
- Mesa 3D Generic Buffer Management (GBM)
- Mesa 3D hardware support status
- Mesa 3D versus proprietary implementations
- Mesa 3D hardware support: debug and documentation

*Presenting 3D APIs and the Mesa 3D implementation.*

### Demo - Userspace Aspects

- Xorg code walkthrough
- Wayland compositor core walkthrough
- Wayland client examples
- Mesa code walk-through
- OpenGL and EGL examples

*Illustrating userspace aspects, client and server implementations.*