Graphics Core Technologies - "The Underpinnings of Everything GUI"#

A rich graphics environment offers a variety of functionality to suit one's target devices. Anywhere from basic instrinsics, to widgets and windows, to hardware abstraction is possible with the following technologies. In all cases, hardware acceleration is used when possible, to provide the fastest pipeline to the GPU.

Advanced Graphics (GF)

The QNX Graphics Framework (GF) is a framework for creating application user interfaces without the overhead of a full windowing system. Its architecture allows it to render directly to hardware, making it faster and more responsive in an embedded environment where resources are limited. GF achieves this by providing direct access to graphics drivers (there's no message-passing or context-switching while rendering), and by using hardware acceleration where possible. This makes it perfect for embedded environments, where it can be used as the basic graphics layer for anything from a simple fullscreen UI to a complex windowing system for multiple application GUIs. It can also act as a porting layer for existing GUIs.

For embedded 3D requirements, the GF supports the OpenGL ES 1.0 specification, a well-defined subset of OpenGL designed for embedded applications. The Neutrino implementation supports the Common profile and EGL platform interface layers.





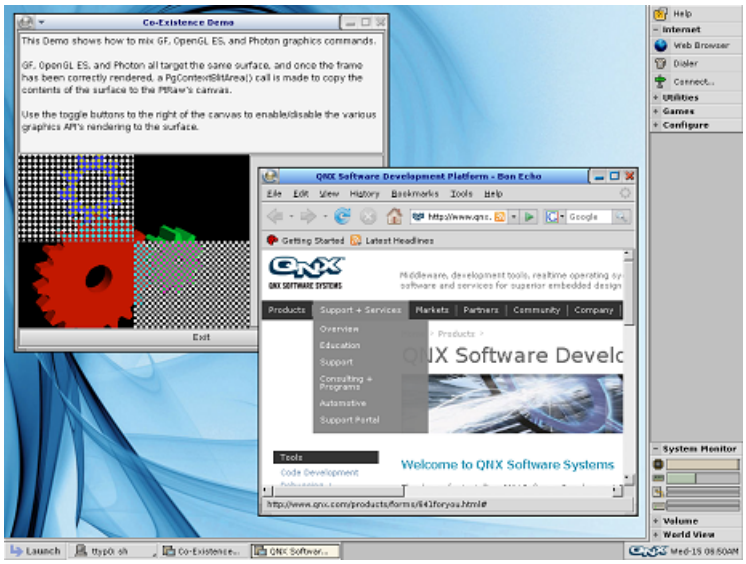
Advanced Graphics offers minimal software fallback for instrinsics. This puts more of an onnus on the hardware to provide the required capabilities, but also removes some of the " bureaucracy" associated with managing software fallbacks.

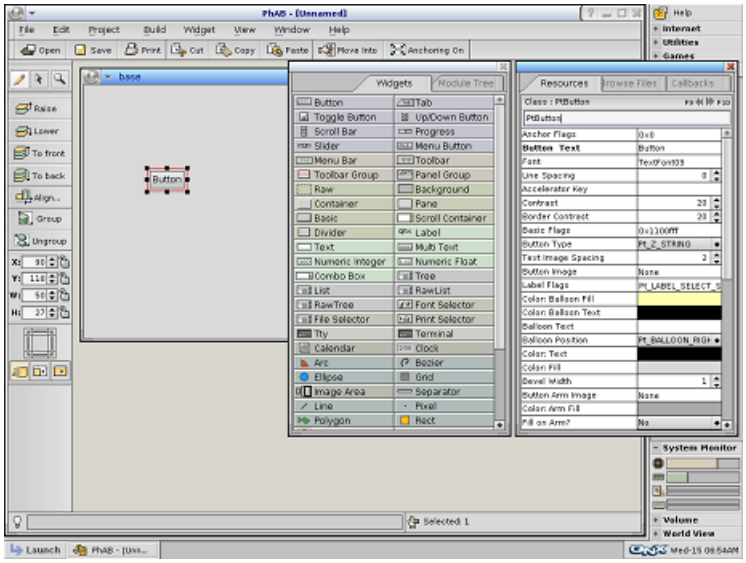
Text output can be generated using Bitstream's Font Fusion 2.4 rendering subsystem directly, or by hooking into the font server provided by Photon.

Advanced Graphics FAQ#

Photon

Photon is a microGUI is a small, powerful, and flexible widget and windowing environment. As of the QNX SDP 64x release, Photon is now underpinned with Advanced Graphics, where the majority of rendering primitives are shunted down to the Advanced Graphics interface. With this integration, applications can now be a **hybrid of Photon and Advanced Graphics API calls**, including OpenGL ES. A minimal windowing environment, with 2D and 3D support, can be **easily accomplished in less than a 4 megabyte IFS** (image file system).





Photon provides software fallbacks for operations not supported by the hardware, allowing designers to be less "tied" to the capabilites of their hardware. Great care has been given to minimize any "bureaucracy" introduced by these software fallbacks.

The architecture of Photon uses central servers for events, graphics, and font processing, in order to minimize system footprint. Photon uses Bitstream's Font Fusion 2.4 rendering subsystem within the font server to generate highly legible, quality text in both monochrome and antialiased output.

Photon FAQ#

Composition Window Manager (CWM) and OpenKODE

CWM provides a hardware abstraction layer to composition hardware (display controller, 2D and 3D cores) so that the same applications can run on various hardware configurations.

Application developers only need OpenKODE and EGL to create windows and make them visible. They don’t have to tie their application to a specific layer and screen. They don’t even have to position or size their application. They can use OpenGL ES, Advanced Graphics, or software renderers to generate the window contents. eglSwapBuffers is used to update the contents of a window.

A subset of the Khronos OpenKODE 1.0 specification is provided. At a minimum, the subset will include the event functions, the error functions, and the windowing functions. This OpenKODE implementation will also support the kdMain entry point. Together, these will allow application developers to create windowed applications without relying on vendor-specific APIs, increasing the portability of their code.

Understanding Composition#

CWM/OpenKODE FAQ#

FlashLite FAQ#

SGX FAQ#