An OpenGL-accelerated desktop with Xgl and Compiz

# BEYOND EYE CANDY

A member of Suse's X11 team delivers an insider's look at Xgl.

BY MATTHIAS HOPF

Mac fans were ecstatic when Apple introduced the Quartz Extreme [1] graphics interface, which accelerated desktop effects using 3D hardware. Microsoft's Windows Vista with its Aero technology looks to close this gap with the Mac. In the world of Linux, Xgl [2] now provides a comparable and even more advanced technology that supports similar effects.

Xgl is an X Server by David Revemann that uses OpenGL to implement graphics output. When a program tells Xgl to draw a line, Xgl passes the vertices to the OpenGL subsystem, which then sends the matching commands to the graphics hardware. Despite this emphasis on OpenGL, Xgl also retains the protocol that existing applications use to talk to the X Server, removing the need to rewrite application programs.

## Background

Following an extended period of standstill in X server development, programmers have integrated new features in recent years to provide a basis for the new Xgl technology. Modern toolkits such as Qt and GTK already use many of these new features without users actually realizing it. Two major protocol extensions, Render and Composite, play an important role for Xgl and the composite manager Compiz.

The Render extension adds new basic primitives for displaying images and polygons, along with a new glyph system for enhanced font displays. This particularly reflects the fact that the legacy graphics commands, called core requests, no longer meet the demands placed on modern toolkits such as Qt and GTK. All primitives can now be linked to data in the framebuffer using Porter-Duff operators [3], thus supporting the rendering of semitransparent surfaces (alpha blending) and fonts with anti-aliasing (pixel coverage). Many modern applications make extensive use of antialiased fonts in particular.

Up to now, the window system used in X has supported overlapping windows, but it has not provided the ability to draw to invisible window areas and display this window content. To achieve this, all windows first have to be drawn in an invisible area of the framebuffer, before all windows are joined (composited) in the visible frame buffer. This is exactly what the X server's Composite extension does.

An external process handles the task of combining all the windows to provide an overall view in a similar way to the well-known window manager. It can use the Render extension to draw multiple superimposed, semi-transparent windows. As compositing and window management must work hand in hand, we can expect to see more compositing window managers in the future with the ability to merge both processes.

Another important X server component that desperately needs reworking is the hardware acceleration architecture, which is responsible for efficient hardware representation of graphic commands. The previous XAA architecture is built around core requests, and is therefore difficult to extend. The architecture outlived its usefulness and needs replacing. The most promising alternatives are EXA and OpenGL.

EXA is straightforward and easy to implement, but OpenGL has the advantage of being a widespread programming interface supported by working drivers. There is no need for the X server to control the hardware. In the future, there will only be one graphics hardware interface, rather that two separate interfaces for XAA/EXA and OpenGL.

## Looking Deeper

In constrast to popular claims, Xgl does not accelerate the execution of OpenGL programs. On the contrary, only indirect rendering is possible for technical reasons at this time of writing. In other words, OpenGL commands are handed to Xgl via the GLX protocol before being passed on to the graphics hardware. Indirect rendering is much slower than direct rendering for programs that need to generate large numbers of polygons (games) or textures (video).

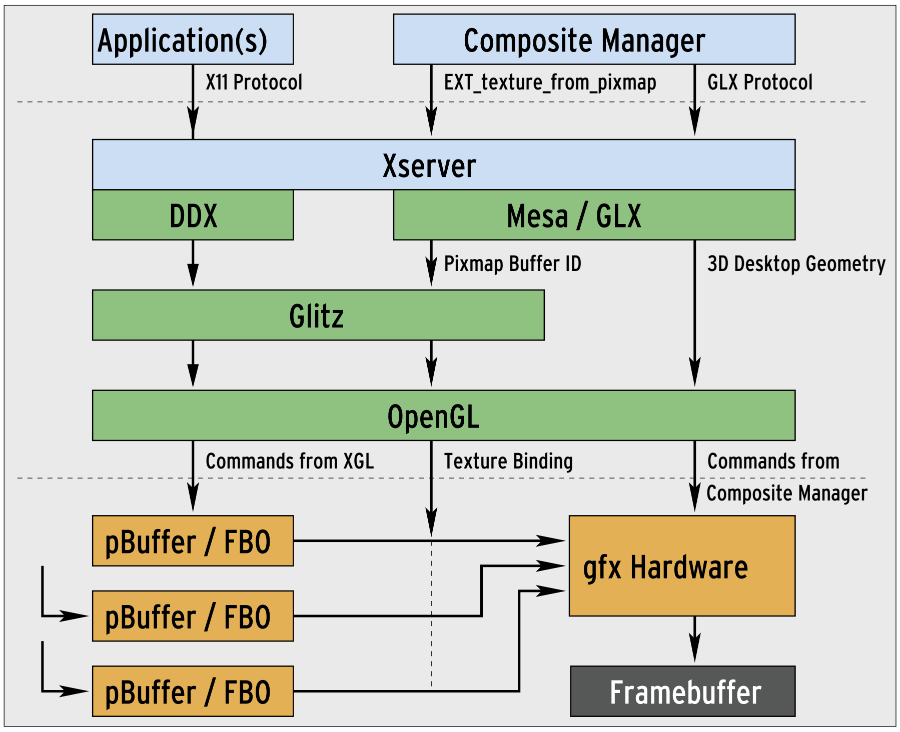


**Figure 1: Pixel pipeline of a composite managers under Xgl.**

At the current time, Xgl cannot natively access the hardware; instead it relies on a system that initializes the framebuffer and provides an OpenGL interface. Right now, that is the popular Xorg X Server; in other words, Xgl opens a window that covers the whole screen on the Xorg server. After this has happened, X applications can connect to Xgl, while the standard X server only has to deal with the Xgl client throughout the whole session.

The X11 commands the server has to handle can be fairly complex; this is why an abstraction layer is used to encapsulate the OpenGL statements. The layer, in the form of the Glitz library, is basically the OpenGL-accelerated back-end for the Cairo library, a system for graphics operations that works independently of the resolution.

When a composite manager enters the array, more complexity is added to the graphics pipeline. As Figure 1 shows, the X Server first redirects all window output to non-visible areas of the framebuffer. A memory area of this kind is created by a pBuffer or Frame Buffer Object (FBO). All X11 commands issued by an application are redirected to this memory space and rendered by OpenGL. This process occurs separately for each program. Then the composite manager draws the window content as textures on OpenGL objects. The objects are typically rectangles, but they can be more complex, three dimensional objects for transitions.

Therefore, Xgl is not responsible itself for the breathtaking effects we have heard so much of, however, it does allow programmers to create a composite manager that can use OpenGL commands to display windows. This option is not open to the normal X Server, as OpenGL is not linked to the underlying window system; that is, it cannot access window content drawn using X11 commands.

As Xgl uses OpenGL internally, it can make window content accessible to an external composite manager using the GLX_EXT_texture_from_pixmap extension. This extension is not provided by the OpenGL driver, but by Xgl. X.org has included this extension since the implementation of AIGLX, but it is still missing support for a number of features.

As previously mentioned, the composite manager uses indirect rendering to draw the desktop; that is, all OpenGL commands are sent to Xgl using the GLX protocol before being passed to the graphics hardware. This is the only way for another process to use textures in the Xgl address scope. This issue also affects all OpenGL applications, as they have to draw in an invisible area of the framebuffer, which has to be in the X server address scope as well. Refer to [4].

## Compiz

Humans are accustomed to understanding three-dimensional scenarios. It thus makes sense to project the GUI onto a three dimensional desktop, assuming the interactions with non-two-dimensional program representations are kept to a minimum. Genuine 3D interaction still

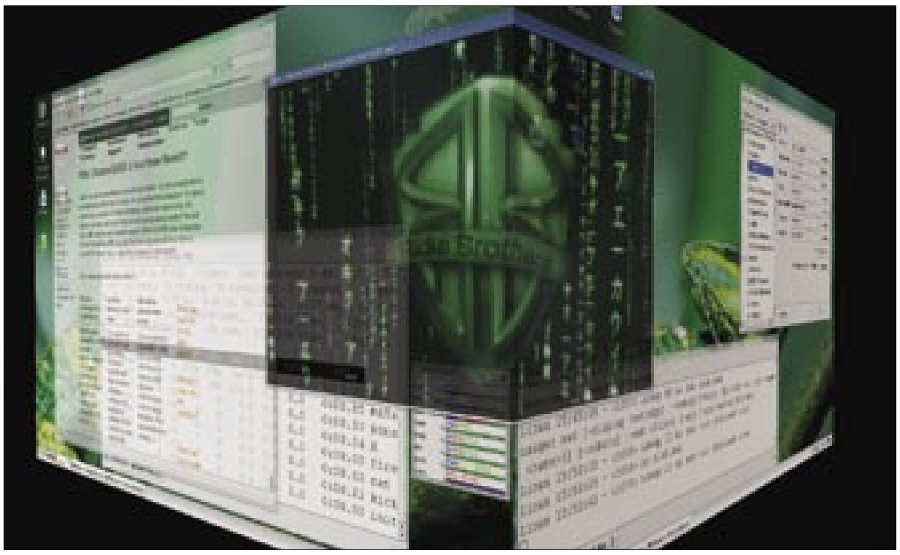
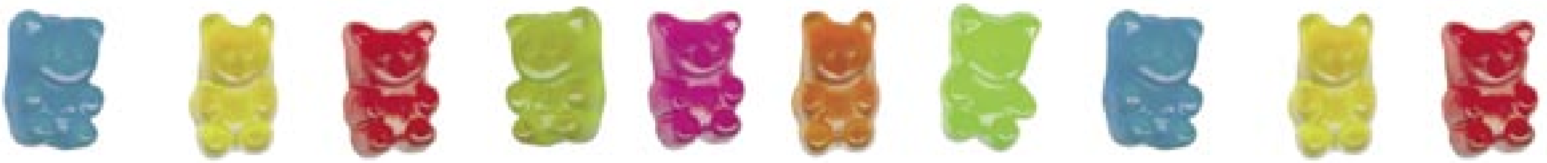


**Figure 2: Compiz toggling to another virtual desktop.**



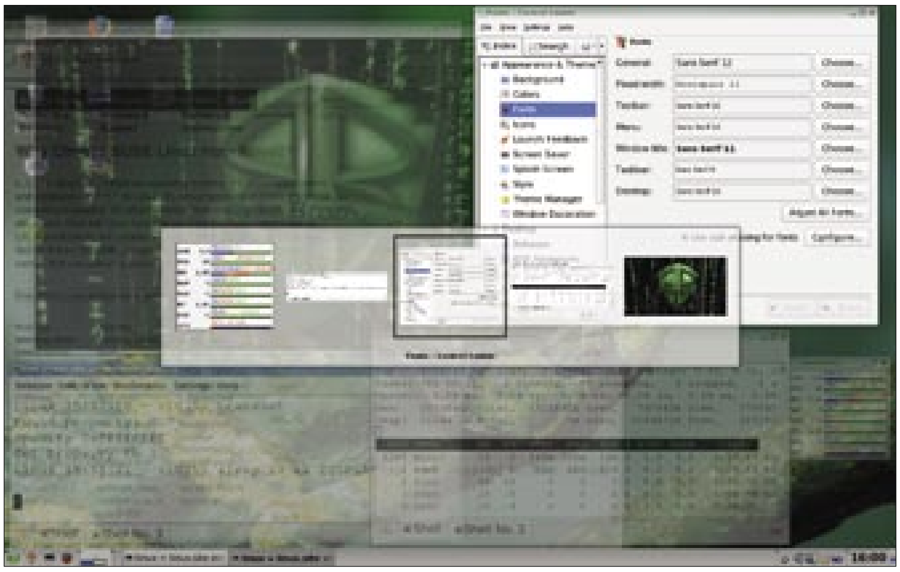


Figure 3: The Switcher plugin with live thumbnails of applications.

poses a number of technical issues and is typically unintuitive.

Projecting two-dimensional pixel data onto three-dimensional objects is a standard application for OpenGL. At the same time, you get effects such as semi-transparency more or less for free, as they are part of OpenGL's standard bag of tricks. So far, the only reason not to release a composite manager of this kind has been that Xgl was needed to open up the OpenGL interface to composite managers.

Compiz is a composite manager specifically designed for the Xgl environment. In a Compiz session, you might not notice that OpenGL is used for output. Shadows and slight semi-transparent window decorations are the only hints you get. But when you toggle to another virtual desktop, the three-dimensional nature of the desktop becomes very obvious (Figure 2).

Compiz uses a highly-flexible, but not completely stable, plugin architecture for all its effects. An external program renders the window decorations and hands them over to Compiz. This makes it easy to integrate themes or different sets of widgets. At present, there is a window dressing program for Gnome that also works on KDE.

Compiz plugins are available for basic window manager functionality, (*decoration*, *move*, *place*, *resize*), for extended functions, (*cube*, *scale*, *switcher*), and for effects (*fade*, *minimize*, *rotate*, *wobbly*, *zoom*). As plugins with comparable functions are exchangable, users can easily modify Compiz to suit their own taste. Check out [6] for a more detailed list of plugins, along with function descriptions and instructions on how to use them.

OpenGL-based programs are often criticized for being eye candy and nothing more. However, this technology really does support useful developments in the field of accessibility aids for users with sensory impairments (using a zoom plugin, for example), and it supports the development of selection aids such as Exposé (scale plugin), which have proved very useful in the Mac world. Thumbnails for applications when users are selecting a productivity tool (Figure 3) are also very useful, especially if they are capable of showing the current live output from the program.

Although Compiz is at the start of its development lifecycle, it is already quite usable. Features common to today's window mangers may be missing, and some exception handlers for special program types are not fully implemented, such as miniature windows for panels. Some work remains before complete KDE integration is achieved.

## The Future of X

Is Xgl the future of the X Window System? Opinions were divided at the last X.org Developers Conference [7] in Santa Clara. Whereas some bemoaned the state of open source drivers, and the incompatibility of proprietary drivers and GPL, others consider OpenGL to be a suitable graphics interface and would like to see the complete driver code ousted from the X server. An X server working on the basis of OpenGL could more easily support a future protocol that dropped core graphics primitive support to improve client-server communications.

One of Xgl's major weaknesses is its lack of native hardware support. For the time being, it is forced to rely on the Xorg server as an intermediary. An experimental branch dubbed Xegl is capable of talking directly to the graphics hardware, although it is restricted to R100 and R200 based Radeon cards, and the current server status is unknown.

Everyone involved seems to agree that the future belongs to OpenGL-based compositing window managers. And Compiz in particular could be *the* window manager of the future thanks to its flexible plugin architecture.

As of this writing, Xgl and Compiz are both fairly stable, but they are at an early stage of their development. Packages are available for openSuse [5] to give users an opportunity to test the new technology without going to much trouble. The current snaphots of the imminent Ubuntu release, Dapper Drake, allow users to install Xgl and Compiz. The system is so stable with some drivers that it is suitable for production use. However, as Xgl does not pass on all OpenGL extensions to its clients, many OpenGL applications and games still trigger runtime errors. ■

## INFO

[1] Apple's Quartz Extreme: *http://www.apple.com/macosx/features/quartzextreme*

[2] Xgl Wiki at freedesktop.org: *http://www.freedesktop.org/wiki/Software/Xgl*

[3] T. Porter & T. Duff, Compositing Digital Images, Computer Graphics Volume 18, Number 3, July 1984, p. 253-259

[4] Matthieu Herrb and Matthias Hopf, New Evolutions in the X Window System, OpenBSDCon '05: *http://www.openbsd.org/papers/eurobsd2005/herrb-hopf.pdf*

[5] Xgl Wiki: *http://www.opensuse.org/Xgl*

[6] Compiz Wiki: *http://www.opensuse.org/compiz*

[7] X.org Developer's Conference: *http://wiki.x.org/wiki/XDevConf*