GUI Widgets for OpenGL

- If app is not full-screen or a simple GLUT program you'll likely need to use an OpenGL drawing widget within a traditional 2D UI
- In the past, Xt/Motif was the most common GUI on 3D workstations. Today on Linux, there's Qt (used by KDE) and GTK+ (used by GNOME), and others.

General Steps for all OpenGL Widgets

- Choose framebuffer/window config
- Instantiate a drawing area widget
- Setup callbacks for initialization, resizing, redrawing, etc.
- Write code for input event processing

OpenGL with Qt

- QGLFormat - like a GLX Visual, describes the framebuffer
- QGLContext - GL context which may be bound to QGLWidgets
- QGLWidget - the drawing area

Simple usage:

- Create a new class derived from the QGLWidget class
- Implement the initializeGL(), resizeGL(), paintGL() methods
- Instantiate the new class within your UI

Advanced usage:

- Use QGLFormat class to specify frame buffer attributes
- Create one or more QGLContexts
- Create one or more QGLWidgets
- Explicity manage binding of contexts to widgets yourself

OpenGL with GTK+

- GtkGLArea - C bindings
- GtkGLArea-- - C++ bindings

Example GtkGLArea usage

1. Setup attribute list:

This is very much like the GLX interface:

```
const int attribs[] = {GDK_GL_RGBA,
                GDK_GL_RED_SIZE, 1,
                GDK_GL_GREEN_SIZE, 1,
                GDK_GL_BLUE_SIZE, 1,
                GDK_GL_DOUBLEBUFFER,
                GDK_GL_DEPTH_SIZE, 1,
                GDK_GL_NONE };
```

Example GtkGLArea usage

2. Check if GL supported:

```
if (!gdk_gl_query()) printf("GL not
  supported\n");
```

If you don't do this you may get an X protocol error later.

We want to fail gracefully.

Example GtkGLArea usage

3. Create the GL widget

```
GtkWidget *glwidget =
  GTK_WIDGET(gtk_gl_area_new(attribs));
```

Example GtkGLArea usage

4. Setup widget's signal handlers (callbacks)

- Redraw
- Resize
- Etc (mouse event handling)

Example GtkGLArea usage

5. Example redraw function:

```
gint redraw(GtkWidget *w, GdkEventExpose *event)
  {
     if (gtk_gl_area_make_current(GTK_GL_AREA(w))) {
   glBegin(GL_TRIANGLES);
   ...
   glEnd();
   gtk_gl_area_swapbuffers(GTK_GL_AREA(w));
     }
     return TRUE;
  }
```

Summary

- **Don't be afraid to try and use these widgets/libraries- they're generally simple and useful.**
- **Don't reinvent the wheel, build on other people's work. There's probably already too many UI and OpenGL toolkits.**
- **Make the effort to create a usable, polished interface- it makes a good impression for Linux.**

OpenGL Solutions for Linux

- Mesa
- Utah-GLX
- XFree86/DRI
- NVIDIA
- Xi Graphics
- Metrolink
- SGI and the S.I.

OpenGL vs. Mesa

Official OpenGL

- Purchase a license to use the trademark
- Passes (most of) the conformance tests

Mesa

- Passes (most of) the conformance tests

Mesa

- An open/free implementation of the OpenGL API
- Available for over five years
- Well established as the "OpenGL solution" for systems with no official OpenGL support otherwise
- Modular and very portable

Mesa (cont)

- Good conformance
- Good performance
- Originally only a software rendering library
- Now being used for hardware acceleration
- www.mesa3d.org

Mesa Software Rendering

X Window System (Linux)

GGI (Linux)	SVGAlib (Linux)
BeOS	MGL (SciTech)
OpenStep	MacOS
MS Windows	

OS-independent off-screen rendering

Mesa Software Rendering for X on Linux

- The original Linux OpenGL solution
- Entirely client-side; built on Xlib
- Allows rendering in almost all display modes (monochrome to truecolor)
- Remote display to any X server (doesn't need GLX)
- Full featured, many OpenGL extensions
- Slow rasterization

Mesa Hardware Acceleration

- There have been three incarnations of hardware-accelerated Mesa:
  - "Stand alone" Mesa + Glide for 3dfx hardware (Linux)
  - Utah-GLX (XFree86 3.3.6)
  - DRI (XFree86 4.0 and later)

Mesa + Glide

- Original hardware acceleration for Linux
- Mesa uses Glide for fast rasterization
- Designed for single-context, full-screen apps (mostly games and demos)
- Simplistic hack allows for rendering into an X window (pixel copy)
- Used by many Linux games

Utah-GLX

- Based upon an open-source implementation of the GLX protocol
  - Original by Steven Parker of the U of Utah
- Uses XFree86 3.3.6 and Mesa 3.2
- Fairly broad hardware support
- Goal was/is to merge into DRI/XFree86 4.0
- Open-source
- utah-glx.sourceforge.net

Utah-GLX Hardware Support

- Matrox G200 and G400
- ATI Rage Pro
- Intel i810
- NVIDIA Riva, TNT, GeForce
- SiS 6326
- S3 ViRGE

Utah-GLX

- Pros:
  - Don't need to recompile X server
  - Simple setup (glx.so X server extension, libGL.so library)
  - Simple driver development environment
  - Useful performance level and feature set

Utah-GLX

- Cons:
  - Very limited direct rendering support
  - Limited to XFree86 3.3.x and Mesa 3.2
  - No official release at this time

Direct Rendering Infrastructure (DRI)

- An architecture for direct 3D graphics hardware support with XFree86 4.0 and Linux
- Clients talk (almost) directly to the hardware, no GLX protocol encoding, transmission or decoding (bypass the X server)
- dri.sourceforge.net

DRI

- Design goals:
  - High performance - maximize potential of hardware
  - Flexibility - for a variety of hardware designs
  - Window multiplexing - multiple 3D windows
  - Portability - to other OSes and architectures
  - Secure - prevent malicious misuse
  - Robustness - don't crash or deadlock the system
  - Open-source - obvious benefits

Components of the DRI

- Kernel module - DMA and AGP memory, etc
- 2D XFree86 driver - traditional 2D X
- 3D DRI driver - 3D hardware support
- libGL.so encodes GLX or loads DRI driver
- DRI extension - communication and resource allocation for 3D
- GLX extension - server-side GLX protocol handling, remote rendering

DRI Fallacies

- Drivers have to use Mesa: FALSE!
- Drivers have to be open-source: FALSE!
- DRI development is closed: FALSE!

DRI Hardware Support in XFree86 4.0 (and later)

- 3dfx Voodoo3 and Voodoo5
- Matrox G200, G400
- ATI Rage 128
- Intel i810
- 3Dlabs Oxygen
- Sun Creator/Creator3D

DRI / 3dfx Voodoo3 and Voodoo5

- **Voodoo3 - 16-bit RGB, 16-bit Z buffer.**
  - multitexture, paletted texture
  - pretty good performance
- **Voodoo5 - 32-bit RGBA, 8-bit stencil, 24-bit Z buffer**
  - hardware stencil operations
  - 2Kx2K textures, texture compression, combiner env
  - T-buffer (support underway), very high fillrate

DRI / Matrox G200, G400

- **G200**
  - single texture unit
- **G400**
  - multitexture, multi-screen
  - much better performance
- **Both**
  - 16-bit RGB, 16-bit Z buffer
  - software stencil and accumulation

DRI / ATI Rage 128

- **16 and 24-bit RGB**
- **16, 24, 32-bit Z**
- **multitexture**
- **software stencil and accumulation**
- **Radeon support coming**

DRI / Intel i810

- **inexpensive graphics integrated into motherboard chipset**
- **16-bit RGB and 16-bit Z**
- **software stencil and accumulation**
- **single texture unit**

DRI / 3Dlabs Oxygen

- An early, experimental driver
- Hardware transform and lighting
- 32-bit RGB, 24-bit Z buffer
- single texture unit
- poor fill rate

DRI Documentation

dri.sourceforge.net

- **DRI Compile Guide**
  - How to download, compile, setup XFree86 4.0 with the DRI
- **DRI User Guide**
  - How to configure, troubleshoot, and use the DRI

DRI Configuration

- **Userful Environment Variables from the DRI User Guide:**

```
LIBGL_DEBUG
LIBGL_DRIVERS_PATH
LIBGL_ALWAYS_INDIRECT
LIBGL_MULTIHEAD
MESA_DEBUG
```

Getting the DRI

- **Pre-packaged in distribution**
  - Debian, SuSe, Red Hat, 3dfx, XFree86.Org
- **Official Releases**
  - Included in XFree86 releases (4.0.1)
- **Latest Stable Code**
  - Download from DRI CVS tree on SourceForge
- **Bleeding Edge**
  - Download from DRI CVS dev branch

Mesa is part of the XFree86/DRI package

- **DRI includes Mesa**
- **No need to install Mesa separately**
- **If no hardware accel available, DRI will render using Mesa in software**

libGL from Mesa vs from XFree86/DRI

- **Traditional Mesa libGL**
  - Pseudo-GLX implementation
  - Works with any X server
  - Limited hardware support
- **XFree86DRI libGL**
  - Real GLX interface
  - GLX protocol encoder
  - Dynamic Loader for 3D hardware drivers

Upgrading to the DRI

- **Remove any existing Mesa installation**
- **Install XFree86 4.0.1**
  - Software Mesa will still be used if you have no hardware
- **Still need standalone Mesa and Glide for Voodoo 1 and 2**
- **Use SGI Sample Implementation's GLU**

DRI Open-Source Development

- **The usual open-source benefits, especially:**
  - Broad coverage testing
  - Quick bug identification and repair (in many cases)
  - Anyone can develop new drivers or new OS support
  - Close user/developer community
  - Shared driver codebase: bug fixes, optimizations, new features are of benefit to all

DRI Future Work

- **Support new graphics chips**
- **Improve performance of existing drivers**
- **Implement new OpenGL extensions**
- **Port to non-x86 and other operating systems**
- **Inclusion of XFree86 4.0.x with DRI into mainstream Linux distros**

DRI Summary

- **DRI facilitates 3D hardware on Linux**
  - Good hardware support and will just get better
  - Open source improves quality and acceptance
- **Hardware vendors decided to bet on Linux**
  - IHVs and ISVs funded the development
  - Let IHVs know that you use their hardware on Linux
  - Let IHVs know you'll buy new products if they support Linux