The GLP OpenGL Extension

OpenGL Rendering Without A Window System

Paul Ramsey

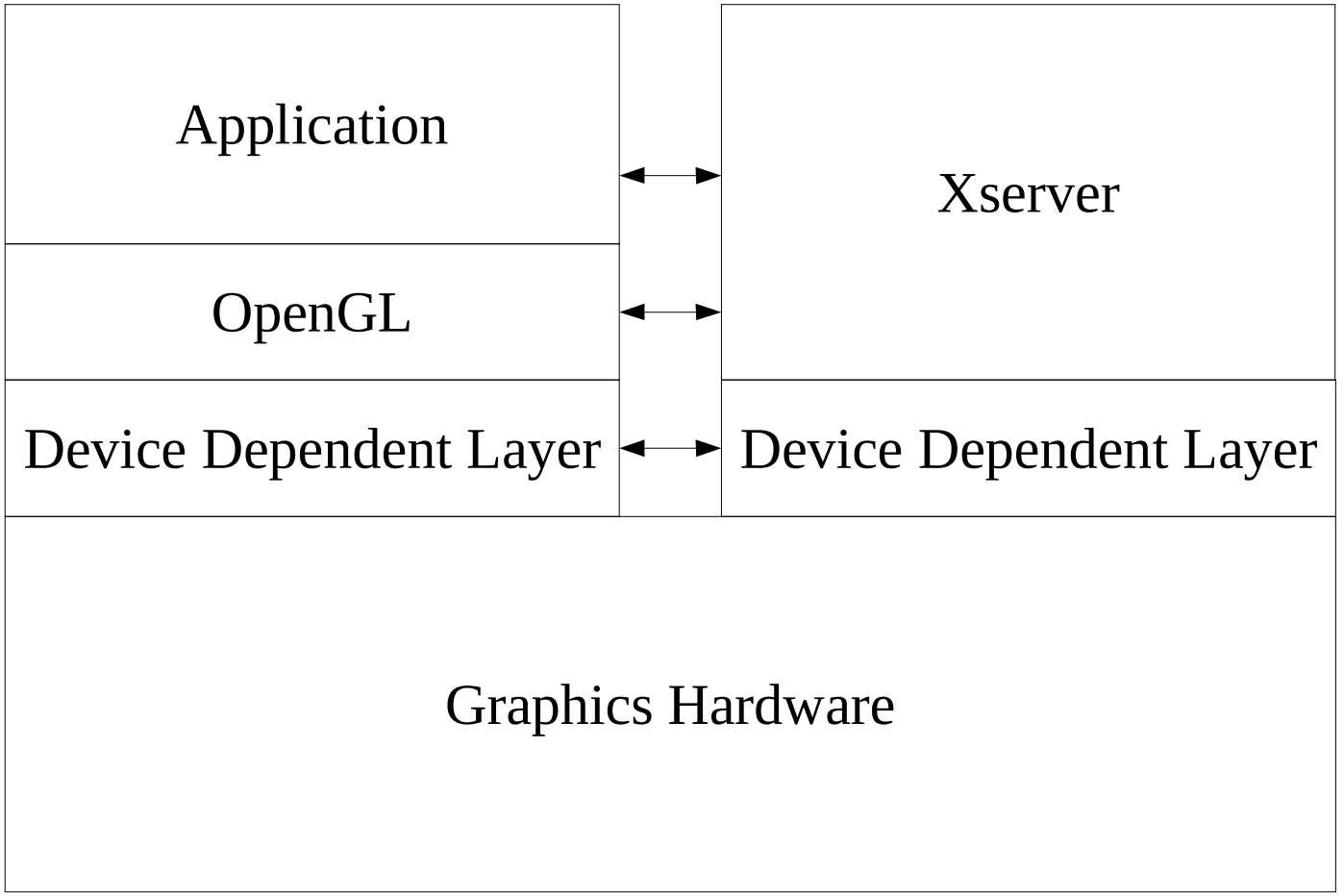
Sun Microsystems, Inc

Sun microsystems
We make the net work.

Graphics Hardware Isn't Just For Visual Programs Any more

- Highly programable pipelines are useful for more general vector computing (GP^2)
- Graphics cluster may only have one display
- Remote rendering renders on a server and displays on a client over the network
- Why talk to a window system?

Window System Adds Another Process To the Solution



Window System Adds Unnecessary Complexity

- Complex Failure Modes
  - Application process starts without the Xserver
  - Xserver process dies
  - Communication failures
  - This matters much more with unattended software
- Complex Permission Model
  - standard file permissions for direct hardware access
  - Xserver permissions for communication with Xserver
  - More of a problem for many simultaneous users than for traditional single user login model

Why Do We Need a New Library?

- A possible cross platform library
  - By not depending on a window system GLP can be implemented on any system which can provide 3D graphics
- Less hacking
  - GLX without an Xserver would need things like “magic” display and screen values

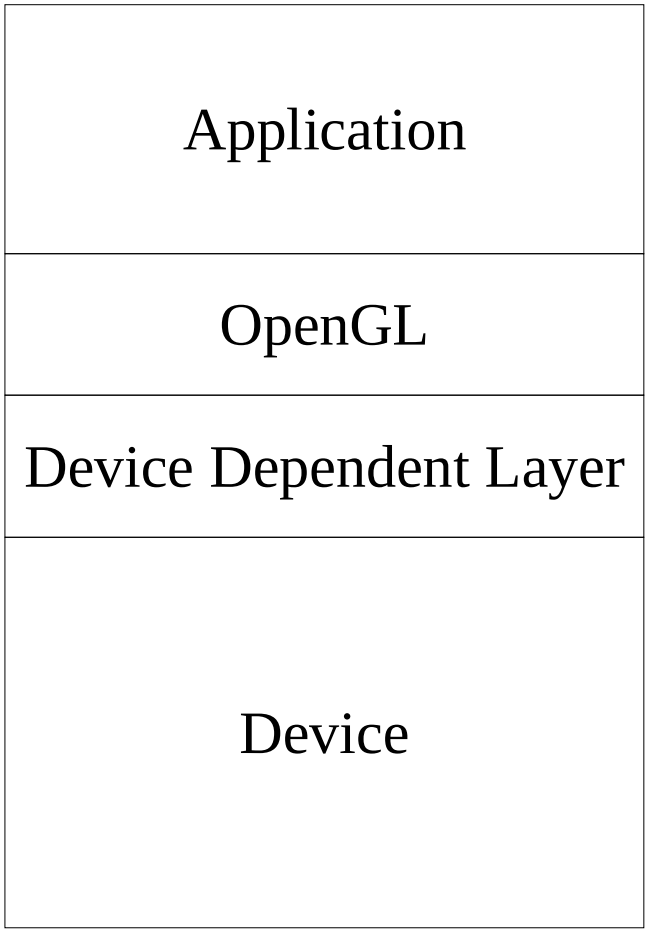
Why Do We Need a New Library? (cont)

- Cleaner starting point
  - If server and cluster specific capabilities are added, it won't be necessary to consider the implications on windowing functions

The GLP Library

- A new library analogous with wgl and glX
- OpenGL rendering with no window system
- pbuffer is the only main drawable (hence GLP)
- Extra (optional?) extensions designed specifically for server rendering

GLP Uses One Process Per Application



GLP Conserves Resources

- No Process Switching
  - Everything is within the same process
- Saved System Memory
  - X-server is replaced with much smaller device and kernel resource allocation software
- Saved Graphics Memory
  - No need to allocate a display buffer

GLP Reduces Complexity

- Fewer Failure Modes
  - No Xserver to start
  - No Xserver to die
  - No interprocess communication
- Simple Permission Model
  - Uses standard device access permission model

The Library Design

- Started with the GLX library
- Removed all window and visual functions
- Removed obsolete functions (why not?)
- Discarded Display parameters
- Replaced screen # with Device handle
- Made esthetic function name changes
- Added functions for directly opening device

Removed these functions

- glXQueryExtension (no server to extend)
- glXIsDirect (always direct)
- glXWaitX, glXWaitGL (no X to synchronize with)
- glXSelectEvent, glXGetSelectedEvent (no events)
- glXChooseVisual, glXCreateContext,glXGetConfig (obsolete)
- glXGetVisualFromFBConfig (there are no visuals)
- glXCreateWindow, glXDestroyWindow (there are no windows)
- glXCreatePixmap, glXDestroyPixmap (there are no pixmap)
- GlXSwapBuffers (buffers aren't visible)
- GlXCreateGLXPixmap, glXDestroyGLXPixmap (obsolete)

Removed glXUseFont

- Can be easily implemented at the user level if an X-server is available somewhere to get the fonts
- Or use other font services like Standard Type Services Framework or platform specific font services

Just changed the name of these

- glPGetCurrentBuffer (was glXGetCurrentDrawable)
- glPGetCurrentContext
- glPGetCurrentDevice (was glPGetCurrentDisplay)
- glPGetCurrentReadBuffer(was glXGetCurrentReadDrawable)

Removed “Display” from these functions

- glPCreateBuffer (was glXCreatePbuffer)
- glPCopyContext
- GlPDestroyBuffer (was glXDestroyPbuffer)
- glPDestroyContext
- glPGetFBConfigAttrib
- glPMakeContextCurrent
- glPQueryContext
- glPQueryBuffer (was glXQueryDrawable)
- GlPQueryVersion

Changed glPCreateNewContext

```
GLPContext glPCreateNewContext(GLPFBConfig config,
                               int render_type,
                               GLPContext shareList)
```

- No Display
- No direct flag (always direct)

Added These Functions

- char** glPGetDeviceNames()
  Get the list of available devices
- GLPDevice glPOpenDevice(char devicename[])
  Open a device by name
  - Library probably won't help choose a device to open
  - Leave that for application or higher level API
  - Maybe let NULL mean “just open any device for me”?
- int glPGetDeviceAttrib(GLPDevice device, int attrib, int* value)
  Get information about a device
  - Information which might help with device selection
- Bool glPCloseDevice(GLPDevice device)
  Close the device

Multiuser Access to Graphics Hardware

- Access limited only by device permissions
- Cross user data security will be implementation dependant
- Level of sharing will be implementation dependant

Hardware Initialization

- No X-server to do initialization
- Move it to kernel?
- Move it to boot up initialization?
- Move it to first process to access device?

Resource Management

- No X-server to do resource management
- Need to keep it in the kernel

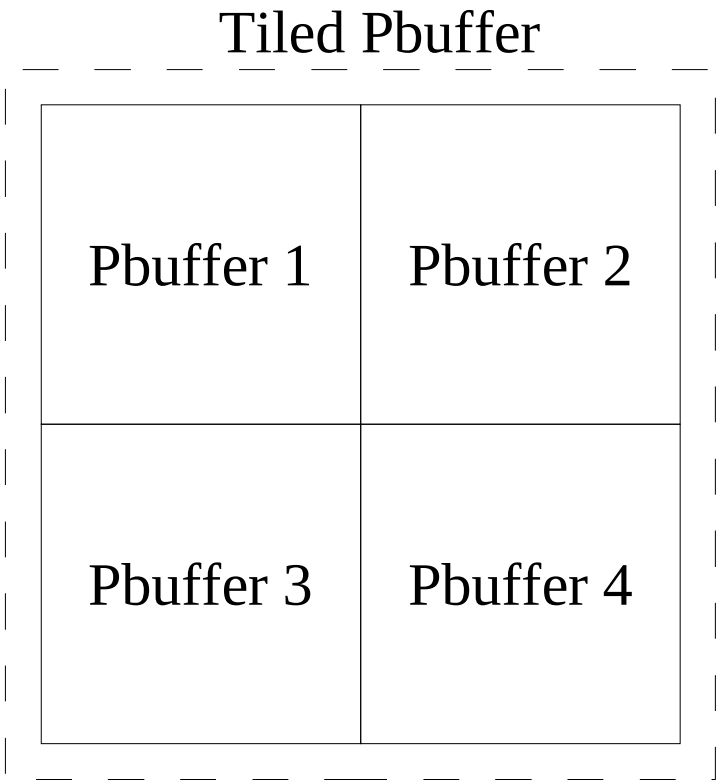
Additional Extensions

- Tiled Pbuffers
- Graphics Checkpointing/Load Balancing

Tiled Pbuffers

- Combine multiple pbuffers into a tiled pbuffer
- Supports larger pbuffers than would normally be possible
- Allows distribution of pbuffer rendering across multiple graphics devices for faster rendering

Tiled Pbuffers (cont)



Graphics Checkpoint/Restore

- Save graphics state and release graphics resources
- Restore graphics state and continue where the application left off
- Allows graphics resources to be freed up when batch or idle graphics applications are suspended
- Checkpointing of the rest of the application is outside of the scope of this feature

Moving Pbuffers

- Conveniently move a pbuffer to another device without destroying it and creating a new one
- Allows redistribution of rendering for load balancing
- Temporarily move individual pbuffers off of hardware into stasis

Conclusion

GLP provides a convenient way to allow applications without windows to run without dependance on a window system, simplifying the runtime environment, reducing system administration complexity and increasing sharability of the graphics hardware

The GLP OpenGL Extension

OpenGL Rendering Without A Window System

Paul Ramsey

paul.ramsey@sun.com

Sun microsystems

We make the net work.