



Where is my mind?

Memory management in the X server

Adam Jackson
ajax@nwnk.net



Overview

- X has several rendering models
 - Classic
 - Direct rendering
 - Render extension
 - Composite
- We get Classic kinda right
- We totally suck at the other three
 - Even individually!



X has Problems

- X has all the usual problems
 - Fragmentation
 - Excess copies
 - Fairness
- And some unusual ones
 - Multiple object types
 - Different usage pattern
 - Bidirectional data flow



Classic model

- Very basic 2D primitives
- Effectively solid fills and blits
 - Sometimes with rops or planemasks
 - Because everything else is ugly
- Xv is a special blit
- Everything done server side



Direct rendering

- Speed hack for OpenGL
- Client sends rendering commands straight to the card
- Server arbitrates access
- Some IPC to maintain consistency



Render

- Porter-Duff image compositing model
- Really pretty alpha blending
- Can be considered a generalized blit
- Can also be considered a GL subset
 - Almost
- Handled on the server side
 - But mostly in software, boo



Composite

- Server no longer sets rendering policy
- External application draws everything
- Have to get bits into the compositing manager somehow
- Has to integrate with all the older rendering models



XAA memory model

- Tall framebuffer
- Offscreen memory must match format
- Naïve allocation policy
- Severe fragmentation problems
- Not the prettiest code ever written
- Does pretty okay at handling Classic



XAA versus DRI

- Varies by driver (!)
- DRI introduces more buffers
 - Back and depth
 - DMA
- Split offscreen memory
 - Once, statically, at server startup
- Not a great tradeoff



XAA versus Render

- Has minimal Render acceleration
- Makes AA fonts slightly less painful
- Only from host to card



XAA versus Composite

- Window contents are redirected to offscreen pixmaps
- And then probably blended using Render
- Which XAA doesn't accelerate
- So you lose



EXA

- Acceleration architecture with no satisfactory acronym
- Based on KAA
- Linear memory management
- Real Render acceleration
- Still split allocation for DRI
- Migration heuristics



Composite integration

- EXA is great, if your compositing manager uses Render
- EXA is potentially great if your compositing manager uses GLX
- Xv can be texture or overlay
- Need ability to redirect DRI surfaces
 - And the ability to not redirect them



The Plan

- Memory tracking in the DRM
- Access all card objects through handles
- DRM-assisted kickout
- DRI protocol extensions
 - redirection
 - damage post
- GLX_EXT_texture_from_drawable



Wild Ideas

- Shared regions among DRI clients
- Journal for distributed synchronization
- Better memory allocation algorithms
- Intelligent use of GART space
- Everything in fragment shaders



Questions