Blink: 3D Display Multiplexing for Virtualized Applications

Jacob Gorm Hansen, University of Copenhagen

January 20, 2006
Introduction
  Motivation

History: The Tahoma Project
  Sprites and Tiles
  Lessons Learned

Blink
  GL in, GL out
  Communication Protocol
  JIT Compiler for OpenGL
  Stored Procedures
  Safety Checks

Evaluation
  Compiler
  Graphics Throughput

Demo

Jacob Gorm Hansen, University of Copenhagen
Virtual Machines are not only for data centers:

- VMWare now ships a free Firefox VM
- The Stanford Collective “Virtual Appliance” Application Bundles
- The CMU Internet Suspend Resume project
- Some distro installers need graphics
- We should be able to do better than VNC
Project Background

- Blink started as part of the “Tahoma” Browser OS
- First developed at the University of Washington, early 2005
- For Tiled 2D graphics
- Idea: Treat 4kB pages as 32x32 tiles, track updates with MMU
- Backend draws tile-grids (Sprites) to the screen with OpenGL
struct {
    int tile_MFN; // tile frame #
} tile;

struct {
    int x, y;       // sprite posn.
    int width, height // # tiles
    tile tile_array[ ];
} sprite;

struct {
    int num_sprites;
    sprite sprite_array[ ];
} virt_screen;
Blink: 3D Display Multiplexing for Virtualized Applications
Lessons Learned

- Modern GPU’s are extremely fast
- Tiles make update tracking easy, thus reduces bus traffic
- But most software expects a linear framebuffer
- And porting QT embedded to tiles was quite painful
A Natural Thought

So...

▶ If we are going to \textit{output} OpenGL anyway...
▶ Why not just take OpenGL as input as well?
▶ Read serialized from Client
  ▶ Interpret it
  ▶ Check that it's safe
  ▶ Execute it
▶ (Then we can always do the tiling in the client if we want)
Serialize OpenGL in Client

Turn:

```
glBegin(n);
```

into:

```
op->code=GL_Begin;
op->args[0]=n;
```
switch(op->code) {
    case GL_Begin:
        glBegin(op->args.integers[0]);
        break;
    ...
}

(turns in to a rather large switch() statement)
Versioned Shared Objects

VSO array

(1,17,sp)  (2,5,tex)

Machine Page Frames
Jacob Gorm Hansen, University of Copenhagen

Blink: 3D Display Multiplexing for Virtualized Applications
A Couple of Improvements

- Turn the Interpreter into a JIT Compiler
- Add virtual registers, arithmetic, conditionals
- Call client code as *Stored Procedure* callbacks
- Results: Native speed asynchronous execution
List of Callbacks

<table>
<thead>
<tr>
<th>Callback Name</th>
<th>Executed</th>
</tr>
</thead>
<tbody>
<tr>
<td>init()</td>
<td>At first display</td>
</tr>
<tr>
<td>update()</td>
<td>On client VM request</td>
</tr>
<tr>
<td>reshape()</td>
<td>On window move or resize</td>
</tr>
<tr>
<td>redraw()</td>
<td>For each display</td>
</tr>
<tr>
<td>input()</td>
<td>On user input</td>
</tr>
</tbody>
</table>
Static Verification

Safety:

► We need to check certain properties during compilation...
► Not all callbacks are allowed to draw to screen
► Client should not exceed its *scissor* rectangle, or disable scissor

Performance:

► Advance knowledge of client actions can also be used for optimizing display
► E.g. if client does not enable Z-buffer, there is no need to clear it first
► If client does not use transparency, we do not have to draw windows behind it
Evaluation

Test Machine:

- 2GHz single-threaded Intel Pentium4 CPU
- 1024MB SDRAM
- 2+ years old
- ATI Radeon 9600SE 4xAGP graphics card with 128MB DDR RAM ($70).
JIT Performance

<table>
<thead>
<tr>
<th>Type of input</th>
<th>#Instr.</th>
<th>Compile</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGL-mix</td>
<td>8,034</td>
<td>102 (41) cpi</td>
<td>41 cpi</td>
</tr>
<tr>
<td>Arith-mix</td>
<td>8,192</td>
<td>99 (55) cpi</td>
<td>50 cpi</td>
</tr>
</tbody>
</table>
## Quality of JIT’ed Code

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Execute</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenGL-mix Native</td>
<td>552 cpi</td>
</tr>
<tr>
<td>OpenGL-mix Blink</td>
<td>554 cpi</td>
</tr>
<tr>
<td>OpenGL-mix Blink + JIT</td>
<td>656 cpi</td>
</tr>
</tbody>
</table>
MPlayer VM’s

Jacob Gorm Hansen, University of Copenhagen
Gears VM’s

![Graph showing the screen redraw rate for different number of GLGears VMs. The graph has x-axis labeled 'Number of GLGears VMs' ranging from 5 to 35, and y-axis labeled 'Update rate (s)' ranging from 0.015 to 0.055. The graph compares Screen redraw rate, GearsBSP, and GearsSwitch.]
Blink: 3D Display Multiplexing for Virtualized Applications

Jacob Gorm Hansen, University of Copenhagen