Accessing PowerVR 2DC Features Under Windows® CE

Jason Powell
GDC - March 18, 1999
www.powervr.com
Talk Overview

- PowerVR Chip Features
- Windows CE for PowerVR-2DC
- Enabling PowerVR Features
PowerVR-2DC

- Graphics chip for the console.
- Chip Features
  - Renders triangles, triangle strips, and quadrilaterals.
  - Culling of back faces and tiny polygons.
  - ARGB Gouraud shading, flat shading.
  - Specular highlighting.
  - Perspective-correct mip-mapped texturing.
  - Texture clamping, wrapping, and mirroring.
  - Bilinear, trilinear, and anisotropic filtering.
  - Full-scene anti-aliasing.
  - Vertex fog as well as all table fog modes.
PowerVR-2DC

Features Continued...

- Per-pixel translucency sorting.
- 32-bit on-chip z-buffering.
- Hardware clipping to a viewport.
- 640x480x24 maximum resolution.
- RGBA 5650, 5551 and 4444 texture formats.
- YUV 422, 420 texture formats.
- 8bpp and 4bpp palletized texture formats.
- Bump mapping.
- VQ texture compression.
- Scene capture architecture.
Windows CE for PowerVR-2DC

- Windows API compatible
- Windows 32-based
- DirectX™ 5.0
  - D3D IM, DDraw, DSound, DInput, DPlay, Dshow
  - Developer Studio™ 97 IDE
- VC++ 5.0

- Priorities on performance and portability
Punch-Through

- Really a color key mode.
- A new render pass in chip to increase color key performance.
- First opaque, punch through, then translucent pass.
- Punch through updates z only when texel is opaque.
- Performance should fall between opaque and translucent performance.
Punch-Through

To create a punch through surface:

Option 1
- Create 1555 surface.
- Set ALPHABLENDENABLE to TRUE.

Option 2
- Create 8 or 4 bit palettized surface.
- Set COLORKEYENABLE to TRUE.

Will not work when...
- BLENDMODE = MODULATEALPHA, DECALALPHA
Texture Compression

- Vector Quantization (VQ)
  - 7:1 compression ratio.
  - “Lossiness” depends on the size and noise of the texture compressed.

256x256 Texture is divided into 2x2 pixel blocks

Full RGBA of all 4 pixels are written to codebook

Resulting “texture” is a 128x128 array of indices into it’s codebook

Compression achieved when 2x2 blocks refer to same entry in codebook.
Texture Compression

- Larger textures compress better, but are “lossier”.
- Diminishing returns at 64x64.
- Compression ratios...

<table>
<thead>
<tr>
<th>Texture Size</th>
<th>Original Texture Size</th>
<th>Compressed Texture Size</th>
<th>Codebook Size (bytes)</th>
<th>Compression Ratio (1:n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>32x32</td>
<td>2048</td>
<td>256</td>
<td>2048</td>
<td>0.89</td>
</tr>
<tr>
<td>64x64</td>
<td>8192</td>
<td>1024</td>
<td>2048</td>
<td>2.67</td>
</tr>
<tr>
<td>128x128</td>
<td>32768</td>
<td>4096</td>
<td>2048</td>
<td>5.33</td>
</tr>
<tr>
<td>256x256</td>
<td>131072</td>
<td>16384</td>
<td>2048</td>
<td>7.11</td>
</tr>
<tr>
<td>512x512</td>
<td>524288</td>
<td>65536</td>
<td>2048</td>
<td>7.76</td>
</tr>
<tr>
<td>1024x1024</td>
<td>2097152</td>
<td>262144</td>
<td>2048</td>
<td>7.94</td>
</tr>
</tbody>
</table>
Texture Compression

- Texture compression done at author time.
  - PVRCONV.EXE converts .bmp’s to VQ format.

- To create a VQ surface...
  - Use PVRCONV to create your artwork.
  - CreateSurface...
    - `ddpfPixelFormat.dwFlags |= DDPF_COMPRESSED`.
    - `Width & height = dimensions of uncompressed texture`.
  - Copy VQ bytes directly to texture.
Bump Mapping

- Requires a two pass render.
- Bump map contained in second texture.
- Each texel (aka buxel) is a vector which is normal to the "bump".
Bump Mapping

- Additional light states...
  - D3DLIGHTSTATE_BUMPINTENSITY
    - The strength of the highlight.
    - Type float.
  - D3DLIGHTSTATE_BUMPDIRECTION
    - The light’s direction within the scene.
    - Type D3DVECTOR.
  - D3DLIGHTSTATE_BUMPAMBIENT
    - The light color.
    - Type D3DVECTOR.
Bump Mapping

To create a bump map surface...

- Use FOURCC code for CreateSurface.
  - #define FOURCC_PNBM MAKEFOURCC('P', 'N', 'B', 'M')
  - ddsd.ddpfPixelFormat.dwFourCC = FOURCC_PNBM;
  - Create surface in system memory.

- Fill surface with bump map pixels.
  - Easiest way is to use HeightToBump_Wrap function in WinCE SDK.
    - Converts a height map into K1K2K3Q pixel (aka buxel) format.

- Create video memory surface and load from system surface.
Bump Mapping

- Rendering the bump map surface...
  - Pass 1: Bump texture pass
    - Set Z compare mode to LESS_EQUAL.
    - Set BUMPINTENSITY, BUMPDIRECTION, and BUMPAMBIENT light states.
    - Set texture blend modes...
      - SRCBLEND = ONE
      - DESTBLEND = ZERO
      - TEXTUREMAPBLEND = DECALALPHA
      - ALPHABLENDENABLE = FALSE
    - Render all bump map surfaces.
Bump Mapping

Rendering the bump map surface...
Pass 2: Base texture pass

Set texture blend modes...
- SRCBLEND = DESTCOLOR
- DESTBLEND = ZERO
- TEXTUREMAPBLEND = MODULATEALPHA
- ALPHABLENDENABLE = TRUE

Render all base textures.
Optimized Textures

- Non-linear video memory format.
- Also referred to as “twiddled”.
- Slight performance hit on texture load.
- Bilinear filtering becomes free.
Optimized Textures

To create an optimized surface...

Option 1:
- Create system memory surface (non-optimized).
- Load image data (ie. bitmap) onto system memory surface.
- Create video memory texture surface with 
  ddsd.ddsCaps.dwCaps |= DDSCAPS_ALLOCONLOAD
- Load system memory surface onto video memory surface.
- Driver automatically optimizes ("twiddles") texture on load.

Option 2:
- Create video memory texture surface with  
  ddsd.ddsCaps.dwCaps |= DDSCAPS_ALLOCONLOAD
- Write to texture memory directly in optimized ("twiddled") format.
Writing to Texture Memory

- Texture memory is linear except optimized/twiddled format.
- Twiddled format is...

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>2</th>
<th>8</th>
<th>10</th>
<th>32</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>9</td>
<td>11</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>6</td>
<td>12</td>
<td>14</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>13</td>
<td>15</td>
<td>47</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>18</td>
<td></td>
<td>48</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
<td></td>
<td>31</td>
<td>63</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Bits are ordered in a set of reverse “N”s.
- Bits 0-15 represent a texel, 16-31 the next texel, etc.
- Supports all texture formats.
That’s All Folks

- Questions

- For additional information…
  - Visit Sega, Microsoft, NEC on show floor at booth #808.
  - Sign on with Sega as a Dreamcast Developer!
    - Call the New Developer Hotline at 415-701-7070.