Mobile Graphics Lab, KNU

• Mobile Graphics Lab
  - School of Computer Science and Engineering
  - Kyungpook National University

• Full Software Engines and Emulators
  - OpenGL ES 1.1 and 2.0
  - OpenGL SC
  - EGL

• Consulting and Educational Services
  - Khronos standards
  - most graphics stuffs
Safety Critical – OpenGL SC 1.0

- a variation of OpenGL library

- targeted at safety-certified avionics and automotive displays
  - minimum driver size and complexity

- to be used as an avionics software, airborne system certifications are required:
  - DO-178B
  - DO-248B
  - DO-254
  - DO-278
OpenGL SC 1.0

- originally, OpenGL ES Security Critical Profile
  - started as a variation of OpenGL ES

- spin out as OpenGL SC 1.0
Relationships

- OpenGL 1.3 $\rightarrow$ OpenGL ES 1.0
  $\rightarrow$ OpenGL SC 1.0
- OpenGL 1.5 $\rightarrow$ OpenGL ES 1.1
- OpenGL 2.0 + extra features $\rightarrow$ OpenGL ES 2.0
- OpenGL 3.x $\rightarrow$ ???
- OpenGL 4.x $\rightarrow$ ???

http://mobilegraphics.co.kr/
OpenGL SC features

• based on OpenGL 1.3
  - also from OpenGL ES 1.0

• extinguished features (w.r.t. OpenGL ES)
  - glBegin / glEnd paradigm
  - display list support: a kind of script

• recently,
  - OpenGL SC 1.0 conformance test suites
  - OpenGL SC 1.1 ?
  - OpenGL SC 2.0 ?
OpenGL SC features

- OpenGL SC core functions: totally 101 functions
  - 32 functions are **completely new features** w.r.t. OpenGL ES
    - `glBegin / glEnd` paradigm
    - `glVertex` functions
    - display list support: a kind of script

- **core addition**
  - OES_single_precision

- **required extension**
  - EXT_paletted_texture

- **optional extension**
  - EXT_shared_texture_palette

© Copyright Khronos Group, 2010 - Page 7
OpenGL SC at a glance

• overview for the specification documents
Geometric Primitives

- All geometric primitives are specified by vertices
- no quadruples, no polygons

GL_POINTS
GL_LINES
GL_TRIANGLE_STRIP
GL_LINE_STRIP
GL_TRIANGLES
GL_LINE_LOOP
GL_TRIANGLE_FAN
GL_POLYGON
GL_QUADS
GL_QUAD_STRIP
Begin/End Paradigm

• state machine approach
  - not supported by OpenGL ES
  - deprecated in new OpenGL specifications

• glBegin( GL_TRIANGLES );
  - glColor4f( 1.0F, 0.5F, 0.5F, 1.0F );
  - glVertex2f( 0.0F, 0.0F );
  - glVertex2f( 1.0F, 0.0F );
  - glVertex2f( 0.0F, 1.0F );

• glEnd( );
Light and Material Features

• no spotlight support

• no two-side lighting

• no multisampling support

• no fog support
Output Primitives

- **line stipple support**
  - not supported by OpenGL ES

- **polygon stipple support**
  - not supported by OpenGL ES
Texturing

- **texture pipeline**
  - at least 2 texture units

![Diagram of texture pipeline]

- Texture unit $#i - 1$ sends color $C_p$ (previous) to texture unit $#i$.
- Texture unit $#i$ sends color $C_v$ (computed) to texture unit $#i + 1$.
- Texture unit $#i + 1$ sends color $C_c$ (environment) back to texture unit $#i$.
- Rasterizer sends color $C_f$ (fragment) to texture unit $#i$.
- Active texture $#i$ sends color $C_s$ (source) to texture unit $#i$.

$C_f$: fragment color
$C_p$: previous color
$C_v$: computed color
$C_s$: source color
$C_c$: environment color
Getting to the Framebuffer

Fragment → Scissor Test → Alpha Test → Stencil Test

Depth Test → Blending → Dithering → Logical Operations → Framebuffer
Pixel Rectangles / Bitmaps

- **DrawPixels**: RGBA image only
- **Bitmap**: B/W image, drawn with current raster color
  - current raster color = current color when the last `glRasterPos()` called.

### OpenGL 1.3

<table>
<thead>
<tr>
<th>Function</th>
<th>Safety-Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>PixelStorei</code>&lt;br&gt; - <code>pname = PACK_ALIGNMENT, UNPACK_ALIGNMENT</code>&lt;br&gt; - <code>pname = all other values</code></td>
<td>✓ -</td>
</tr>
<tr>
<td><code>DrawPixels</code>&lt;br&gt; - <code>format = RGBA type = UNSIGNED_BYTE</code>&lt;br&gt; - <code>all other combinations</code></td>
<td>✓ -</td>
</tr>
<tr>
<td><code>Bitmap</code>&lt;br&gt; - <code>sizei width, sizei height, float xorig, float yorig, float xmove, float ymove, const ubyte *bitmap</code></td>
<td>✓</td>
</tr>
</tbody>
</table>
Pixels

- ReadPixels: RGBA, unsigned byte only
- CopyPixels: specified region → current raster position

<table>
<thead>
<tr>
<th>OpenGL 1.3</th>
<th>Safety-Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>ReadBuffer</code> (enum mode)</td>
<td></td>
</tr>
<tr>
<td><code>ReadPixels</code> (int x, int y, sizei width, sizei height, enum format, enum type, void *pixels)</td>
<td>✓‡</td>
</tr>
<tr>
<td><code>CopyPixels</code> (int x, int y, sizei width, sizei height, enum type)</td>
<td></td>
</tr>
<tr>
<td>type = COLOR</td>
<td></td>
</tr>
<tr>
<td>type = DEPTH, STENCIL</td>
<td>✓</td>
</tr>
</tbody>
</table>
## Display Lists

- no release of allocated lists

<table>
<thead>
<tr>
<th>OpenGL 1.3</th>
<th>Safety-Critical</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>NewList(uint list, enum mode)</code></td>
<td>✅</td>
</tr>
<tr>
<td>mode = COMPILE</td>
<td></td>
</tr>
<tr>
<td>mode = COMPILE_AND_EXECUTE</td>
<td>−</td>
</tr>
<tr>
<td><code>EndList(void)</code></td>
<td>✅</td>
</tr>
<tr>
<td><code>CallList(uint list)</code></td>
<td>−</td>
</tr>
<tr>
<td><code>CallLists(sizei n, enum type, const void *lists)</code></td>
<td>✅</td>
</tr>
<tr>
<td>type = UNSIGNED_BYTE, UNSIGNED_INT</td>
<td></td>
</tr>
<tr>
<td>type = BYTE, SHORT, UNSIGNED_SHORT, INT</td>
<td>−</td>
</tr>
<tr>
<td><code>ListBase(uint base)</code></td>
<td>✅</td>
</tr>
<tr>
<td><code>GenLists(sizei range)</code></td>
<td>✅</td>
</tr>
<tr>
<td><code>IsList(uint list)</code></td>
<td>−</td>
</tr>
<tr>
<td><code>DeleteLists(uint list, sizei range)</code></td>
<td>−</td>
</tr>
</tbody>
</table>
Core Addition

• OES_single_precision extension
  - DepthRangef(clampf n, clampf f)
  - Frustumf(float l, float r, float b, float t, float n, float f)
  - Orthof(float l, float r, float b, float t, float n, float f)
  - ClearDepthf(clampf depth)

• same to OpenGL ES 1.0 and 1.1
Required Extension

• **EXT_paletted_texture**
  - need COLOR_INDEX8_EXT extension support for texturing functions
  - need some extensions into the TexImage2D function

  - ColorTableEXT
  - ColorSubTableEXT
  - GetColorTableEXT
  - GetColorTableParameterivEXT
Required: EXT_paletted_texture

- paletted texture
Optional Extension

- **EXT_shared_texture_palette**
  - several textures can share a single texture palette

- need some extensions to Enable, Disable, isEnabled, GetBooleanv, GetFloatv, GetIntegerv functions

- extra extensions to:
  - ColorTableEXT
  - ColorSubTableEXT
  - GetColorTableEXT
  - GetColorTableParameterivEXT
Technical Issues

• new implementations are needed for:
  - required extension: EXT_paletted_texture
  - optional extension: EXT_shared_texture_palette

• since year 2004, there is no graphics card supporting the above extensions
  - old nVIDIA chips such as GeForce 5 series can support it.
  - no ATI chips support it.

• Conclusively, new implementations are required
Implementation Strategy

• **Step 1: emulator implementation**
  - OpenGL SC on (desktop) OpenGL
  - done

• **Step 2: software implementation**
  - implementing

• **Step 3: silicon implementation**
  - FPGA version
# Emulation Results

<table>
<thead>
<tr>
<th></th>
<th>OpenGL</th>
<th>SC on GL</th>
<th>ratio</th>
<th>delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>gears with GLX</td>
<td>4742.7</td>
<td>4676.9</td>
<td>98.61%</td>
<td>1.39%</td>
</tr>
<tr>
<td>gears with EGL</td>
<td>4776.2</td>
<td>4692.6</td>
<td>98.25%</td>
<td>1.75%</td>
</tr>
<tr>
<td>gears with glut</td>
<td>1325.5</td>
<td>1301.8</td>
<td>98.21%</td>
<td>1.79%</td>
</tr>
<tr>
<td>clock with glut</td>
<td>1178.6</td>
<td>1159.0</td>
<td>98.34%</td>
<td>1.66%</td>
</tr>
<tr>
<td>spin with glut</td>
<td>1261.3</td>
<td>1239.0</td>
<td>98.23%</td>
<td>1.77%</td>
</tr>
<tr>
<td>angeles with glut</td>
<td>339.4</td>
<td>332.6</td>
<td>97.99%</td>
<td>2.01%</td>
</tr>
<tr>
<td>average</td>
<td></td>
<td></td>
<td>98.27%</td>
<td>1.73%</td>
</tr>
</tbody>
</table>

http://mobilegraphics.co.kr/
Certifications for Airborne Systems

- DO-178B
- DO-248B
- DO-254
- DO-278
DO-178B

- DO-178B, Software Considerations in Airborne Systems and Equipment Certification
  - from RTCA, Inc. (Radio Technical Commission for Aeronautics)
  - FAA (Federal Aviation Administration) certified
  - guidance to determine if the software will perform safely and reliably, in an airborne environment

- also known as ED-12B
  - from EUROCAE (European Organization for Civil Aviation Equipment)
DO-178B: process

- from the **Software Engineering** point of view,
- requires **lots of documents**

at Planning Stage,
- Plan for software aspects of certification (PSAC)
- Software development plan (SDP)
- Software verification plan (SVP)
- Software configuration management plan (SCMP)
- Software quality assurance plan (SQAP)
- System requirements
- Software requirements standards
- Software design standards (SDS)
- Software code standards (SCS)
DO-178B : process

- **Development stage,**
  - Software requirements data (SRD)
  - Software design description (SDD)
  - Source code
  - Executable object code

- **Verification stage,**
  - Software verification cases and procedures (SVCP)
  - Software verification results (SVR):
    - Review of all requirements, design and code
    - Testing of executable object code
    - Code coverage analysis
DO-178B: process

• Configuration management
  - Software configuration index (SCI)
  - Software life cycle environment configuration index (SECI)

• Quality Assurance
  - Software quality assurance records (SQAR)
  - Software conformity review (SCR)
  - Software accomplishment summary (SAS)
DO-248B

- clarification of the guidance material in DO-178B.
- 12 errata to DO-178B
- 76 frequently asked questions (FAQ)
- 15 discussion papers
  - Verification Tool Selection Considerations
  - The Relationship of DO-178B/ED-12B to the Code of Federal Regulations (CFRs) and Joint Aviation Requirements (JARs)
  - Application of Potential Alternative Methods Compliance for Previously Developed Software (PDS)
  - ...
DO-254

- RTCA/DO-254, design assurance guidance for airborne electronic hardware
  - providing guidance for the development of airborne electronic hardware
- Complex electronic hardware includes devices like Field Programmable Gate Arrays (FPGAs), Programmable Logic Devices (PLDs), and Application Specific Integrated Circuits (ASICs).
DO-278

- DO-278, Guidelines for Communications, Navigation, Surveillance, and Air Traffic Management (CNS/ATM) Systems Software Integrity Assurance