Real-Time Graphics Architecture

Lecture 16: The Design of OpenGL

Kurt Akeley
Pat Hanrahan
http://graphics.stanford.edu/cs448-07-spring/

Early SGI history (dates are approximate)


Jin Clark writes LDS for ES5 picture system
Clark and Hanrahan develop Geometry Engine
Silicon Graphics (SGI) incorporated
IRIS 1000 terminal
IRIS 2400 workstation
IRIS GL licensed to IBM
SGD and Puck project
GTX and Personal Iris

CS448 Lecture 16
Kurt Akeley, Pat Hanrahan, Spring 2007
Historical context (circa 1990)

SGI is the dominant workstation 3-D graphics player
- High-end multiprocessor: PowerSeries GTX, VGX
- Desktop: Personal Iris, Indigo

Workstation 3-D APIs are proprietary, not interoperable
- IRIS GL (SGI)
- Starbase (HP)
- ...

IRIS GL is a mess
- Evolved (accreted) by SGI from 1982 on
- Poorly specified with inconsistent implementations
  - IBM licensing deal
  - GTX/Personal IRIS development

PC is becoming an interesting platform
- Intel 486 included on-board FPU and cache memory
- Microsoft exploring 3-D collaboration with SGI

PHIGS/PEX history* (dates are approximate)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>CORE published</td>
<td>CORE reorganized at ANS 1980</td>
<td>PHIGS becomes ANSI standard</td>
<td>PEX 3.0 spec completed</td>
</tr>
<tr>
<td>PHIGS/DHIC group organized by Dave van Dam</td>
<td>Digital and Sun propose X3D, leads to PEX effort</td>
<td>PEX interoperability demo**</td>
<td></td>
</tr>
</tbody>
</table>


** Programmers Hierarchical Interactive Graphics System

*** Convex, DEC, E&S, HP, IBM, Kubota, ShoGraphics, Stardent, Sun, Tek
PEX (unofficially the PHIGS Extension to X)

The PEX threat (circa 1990)

PEX considered sub-optimal by SGI
  - Retained scene graph* (immediate mode desired)
  - X protocol extension (API desired)
  - Differed from IRIS GL
  - Designed by committee
  - NIH (maybe, we had ignored it)

But PEX will soon be very real ...

* Actually, while PHIGS had no immediate mode, PEX had been extended by this time to include one
The (controversial) choice

Remain proprietary
- Evolve IRIS GL

Drive standards
- PEX
- Our own initiative
  - Internally called GL5
  - Became OpenGL

Talk outline

Goals
Design decisions
Successes and failures
Moving forward
Goals for OpenGL

Industry-wide acceptance (beat PEX!)
Consistent implementations
Innovative implementations
Innovative and differentiated applications
Long life
High quality
Non-goals

- Make graphics programming easy
  - OpenGL is a power tool
  - You might lose a finger (i.e., see a black screen)

- Integrate digital media and 3D graphics
  - This is a much larger task
  - See discussion of Khronos later in the talk

Goal: Industry-wide acceptance

- Avoid compromising performance
  - Allow explicit application trade-offs (modes)

- Get it right the first time
  - Make conservative changes from IRIS GL
  - Collect lots of input during design
  - Do implementation during design

- Create an open standard
  - Licensing program
  - Architecture Review Board (OpenGL ARB) to control future evolution of specification
Goal: Industry-wide acceptance (cont.)

Achieve compatibility with multiple ...
- Operating systems (Windows, IRIX, HP UX, AIX, ...)
  - Ultimately Linux, Mac OS, OS/2, Be, ...
- Window systems (Windows UI, X Window System, ...)
  - Frame buffer is not part of OpenGL state
- Programming languages (C, FORTRAN, Java, ...)
  - No pointers, structures, function overloading
  - No 2-D arrays at interface (row-major / column-major)

Target Microsoft acceptance
- Omit 2D-only functions, window interface, font support
- Allow full application compliance with Windows standards
- Issue: no driver interface specified

Goal: Industry-wide acceptance (cont.)

Marginalize PEX
- API, not protocol, is the right interface
- But client-server mattered too
  - Defined GLX protocol
  - Included server-side display list storage
  - Carefully specified client-side state

Match current hardware capabilities
- Get input from other IHVs

Meet current application needs
- Get input from ISVs
Goal: Consistent implementations

- Tightly written specification
- Conformance tests and required verification
- Complete implementations
  - No sub-setting of 1.n specification
  - Minimum resource requirements specified
- Required runtime error semantics (small performance cost)
  - Check and report
  - No other side effects
- Incentives to share extensions
  - Balance desires for consistency and for innovation
Result: portable applications

Goal: Innovative implementations

- Specification not too tight
  - Not pixel-exact
  - Smooth antialiasing loosely defined
- Abstraction sometimes tilted in implementor’s favor
  - Texture images and filters are paired
  - Evaluated attributes do not become “current”

Extensibility
- Key to IHV innovation
- Requires IHV control of entire driver
  - Obvious for hardware systems companies
  - Not at all obvious for Microsoft
Extensibility

Registry is maintained on www.opengl.org

- Over 340 extensions so far
- Names, token values, GLX protocol, ...

Careful extension documentation

- Suffix/prefix rules
  - Clearly identify all non-core commands and tokens
- Extension numbers
  - Must account for all lower-numbered extensions
  - OpenGL is more than the sum of its parts
- Extension specification template

Extension template

<table>
<thead>
<tr>
<th>Name</th>
<th>Dependencies (details)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name strings</td>
<td>Errors</td>
</tr>
<tr>
<td>Version</td>
<td>New state (table format)</td>
</tr>
<tr>
<td>Number</td>
<td>- Get Value</td>
</tr>
<tr>
<td>Dependencies (list)</td>
<td>- Get Command</td>
</tr>
<tr>
<td><em>Issues</em></td>
<td>- Type</td>
</tr>
<tr>
<td>Overview</td>
<td>- Initial Value</td>
</tr>
<tr>
<td>New procedures and functions</td>
<td>- Attribute Set</td>
</tr>
<tr>
<td>New tokens</td>
<td>New implementation-dependent state (table format)</td>
</tr>
<tr>
<td>Additions to chapter [2,3,4,5,6]</td>
<td><em>Usage examples</em></td>
</tr>
<tr>
<td>Additions to GLX specification</td>
<td></td>
</tr>
<tr>
<td>GLX protocol</td>
<td></td>
</tr>
</tbody>
</table>
Extension syntax rules

Abbreviations (required, specifically not allowed)

Compound words (required)

Naming rules

- General (e.g., “depth” rather than “z”)
- Procedures (e.g., verb-noun or adjective-noun)
- Defined constants (all caps, underbars)
- Parameters ([0],[1],[2], <target>, <params>)
- Extensions (prefixes, suffixes)

Parameter order and typing rules

Suffix codes

Extension categories

Proprietary

- Use corporate prefix/suffix (e.g. NV)

EXT

- Use EXT prefix/suffix
- Must be implemented by at least two licensees

ARB

- Use ARB prefix/suffix
- Specification controlled by the ARB

1.n

- No prefix/suffix
- Specification controlled by the ARB
Goal: Innovative applications

Design qualities that encourage innovative use
- Imperative (procedural) interface
  - In X-speak, “mechanism, not policy”
- Orthogonal mechanisms
- Data recirculation
  - Similarity of pixels and texels
  - Machine shop analogy
- Sufficiency of capability (e.g., stencil, blending)
- Intuitive usability (not ease of use)
- Invariant operation

Taken together, these design qualities allow
- “programmable” operation, in the multi-pass sense
- Usages that were not anticipated by the designers

But application developers will not innovate if implementations are not consistent!

Goal: Long life

Extensibility
- Enabled
- Directed (process in place)

Strong foundation
- Precise specification
  - And culture that values it
- Anticipation of important trends
  - Texture mapping
  - Integration of image processing and 3-D graphics
Goal: High quality

Aim for beauty, not just sufficiency or correctness
- SGI culture supportive of this

Avoid design-by-committee
- Build a horse, not a camel

Correct well known IRIS GL deficiencies
- No command prefixes
- Can't be used by libraries (incomplete save/restore)
- Error reporting via printf()!

Provide documentation
- Specification
- Man pages
- Programming guide

Design Decisions
Unit-square pixels and (matching) texels

- Exact tiling of window area
- Matches window system ownership model
  - No window compositing
  - WIDS (later: CIDs and DIDs)
- Correct frustum arithmetic
  - Consider single-pixel frustum
- Correct texture wrapping

Not centered at integral units
- Non-negative in window
- Obvious viewport arithmetic

Matching pixels and texels
- Obviously important (in retrospect!)

No direct render to texture

Pixel and texel coordinates are equivalent
But pixels and texels are still distinguished
- Allows implementation with separate memories
- Supports OpenGL state model
  - Textures are OpenGL state
  - Frame buffer contents are not
    - Allows multiple renderers to share frame buffer
- An unnecessary orthogonality?
No return values (except from queries)

Status return values are a common library feature
But they have horrible performance implications
- Expose pipeline latency
- Expose client-server latency

OpenGL error system carefully engineered to be
- Asynchronous
- Distributed

```c
void glBegin(GL_POLYGON);
GLstatus glBegin(GL_POLYGON);
```

Evaluated attributes do not become current

“Correct” semantics
- Force separate hardware for evaluation and vertex ops
- Defeat “unified” implementation

Compromise allows practical implementation
More design decisions

Object IDs (not handles)
- No return value 😊
- Scriptable

Sifdv interface
- For application convenience and efficiency

Discourage incremental matrix arithmetic
- No pre-multiply
- Stack provided for undo

Application-specified clipping done in eye coordinates
- Allows use of singular projection matrixes

Successes and Failures
Success with respect to goals

Industry-wide acceptance
- Great for Unix, Linux, Apple, embedded systems
- Good as can be expected with Microsoft ☺
- Poor for game consoles

Consistent implementations
- Good for core features
- Marginal for new features

Innovative implementations
- Industry and academic standard
- Interesting work-arounds for constraints

Success with respect to goals (cont.)

Innovative and differentiated applications
- Very successful

Long life
- Fifteen years and counting

High quality
- Generally respected
- But implementations on Windows suffer as Direct 3D becomes more important
Technical successes

Imperative interface

- “Every graphics systems aspires to have an imperative interface”

Generalized texture capability

- All primitives (with pixel texture extension)
- Various dimensions of texture
- Texel and pixel conformance
- Incremental in-place modification
- Explicit format, filter specification, MIP level, ...
- Texture matrix

Formalization of fragment

Mistakes

Business

- Taking our eye off Microsoft

Technical

- Display lists as encapsulation for “objects”
  - Forced to introduce texture objects in version 1.1
- Overly-abstract storage management
  - Proxy-priority middle ground is not tenable
- Few “fast path” clues in API
  - Too much faith in pure mechanism
  - E.g., DL objects, window position, 2-D transform, in-place frame buffer to texture memory transfer, ...
- Persistent parameter state between Begin/End
  - But would have broken IRIS GL programs badly
Mistakes (cont.)

Probably should have been omitted
- Texture borders! (but not texture-border color)
- Edge flags
- Smooth polygon antialiasing
- Color index features (lighting, antialiasing, ...)

Needlessly complex or wrong
- ColorMaterial()
- Normal transformation
- Texture wrap semantics
- Bit fields (cannot support extensions)
- Viewport should accept doubles, not ints
- Presumption of “view direction” (lighting, fog)

---

Did OpenGL help SGI?

“While the quality of tangible goods has increased dramatically over the last 30 years the quality of software artifacts has been steadily declining.”

— Alex Stepanov
In *Industrializing Software Development*,
Zhejiang University, 2004

Software quality and corporate success may be inversely correlated
Lessons

It is worthwhile to specify carefully

- Don’t complete the implementation before starting on the specification!

It is critical to guide system implementers and application developers to a common understanding of how to support and achieve high performance.

People make and evolve standards

- Invest in developing a culture
- Keep competitive issues out of a specification’s controlling body.

Moving Forward
Future of OpenGL (from 2001 talk)

On the rebound

- Low period in the late 90s

Provides alternative to DirectX/Microsoft API control

Apple commitment is a positive sign

Programmability is an exciting development

- Fits well in OpenGL structure
- OpenGL may truly be a “library” some day

Lots of 2.0 activity, ARB still functioning

Tenth birthday this summer (at SIGGRAPH)

Khronos - Open Media Standards

Open International Membership
Any company is welcome
Funded by membership dues - $6K / year

Open Standards
Publicly available on web-site
To be used by any company in any products

Creating Market Opportunities
Bringing together industry’s leading media technology experts
Investing hundreds of man years to create state-of-the-art media API standards

Making those standards available to the industry – ROYALTY-FREE
Over 100 companies creating media authoring and acceleration standards

Khronos Standards Ecosystem

Cross-platform graphics authoring/acceleration ecosystem
All open standards for native 3D graphics acceleration and authoring are now being developed in Khronos. Tremendous opportunity for collaboration and synergy

Cross platform desktop 3D

Dynamic Media Authoring Standards

Dynamic Media

“DirectX-like” set of native APIs
Includes mixed media acceleration and OS portability APIs

Composition Working Group
Hardware acceleration for window systems
Khronos Participation Model

- **Promoters**: Board decides strategy – approves working groups, controls budget, ratifies specifications. $20,000 annual membership dues
- **Contributors**: Any company can join Khronos to participate in any number of working groups to produce specifications. $6,000 annual membership dues
- **Openly and publicly distributed – free of charge, royalty free**: Free libraries, utilities, examples
- **Conformance Tests and Conformance Test Process**: Typically $10K per API fee
- **Ratified Specifications**: Any company can download specifications and SDKs and implement royalty-free products
- **Conforming products can use API trademark and logo**: Conforming products can use API trademark and logo

Khronos Conformance Process

- **Non-Members → Promoters (Contributors)**
  - Become Adopter Member ($2,500 annual fee)
- **Sign Adopters Agreement to access Test Source and Process (typically $10K per API)**
- **Port Test Source to product and generate test results**
- **Upload test results to Khronos private web-site for peer review by other Adopters**
- **Successful Review means products can use Khronos trademarks**

Already released:
- OpenVG
- OpenMAX
- OpenKODE

COLLADA, OpenKODE coming soon

Kurt Akeley, Pat Hanrahan, Spring 2007
SGI / Pixar collaboration

“It seems to me that we are trying to merge troff and Scribe into a single hodge-podge. I don’t think that we’ll get a good result if we continue.”

“It seems that a key to a high-performance interface is choosing the right boundary between what and how. … Even the Pixar language has this boundary in it. It still does not support a description like ‘lighted like a Sunday afternoon in September’.”

“I am in touch with Pat Hanrahan. He will send me an updated copy of a document outline today. I am to be working on what I discuss in this note. What next?”

— Kurt Akeley (to Forest Baskett)

Fahrenheit

SGI promoting and supporting PC OpenGL
Microsoft asserting control of their own platform
Fahrenheit is the negotiated settlement
Results:

- Broad reach of agreement failed
  - No new low-level standard, little at scene graph level
- OpenGL still exists on all Windows platforms
- SGI learned a lot about Microsoft’s business
- Microsoft learned a lot about graphics
  - Seamus’s presentation at Graphics Hardware 2001
Direct3D

For years Direct3D has patterned itself after OpenGL
- Immediate mode
- Features such as stencil buffer

There are surprising examples in Direct3D 10
- Geometry shaders (anticipated by evaluators)
- Unified shaders (anticipated by evaluators)
- Stream out (anticipated by feedback)

Even GPU programmability is not really new
- Recall earlier lecture
- But also ...

From the Wikipedia article on PHIGS:

The rise of OpenGL and the decline of PHIGS

OpenGL, unlike PHIGS, is an immediate-mode rendering system with no "state"; once an object is sent to a view to be rendered it essentially disappears. Changes to the model have to be re-sent into the system and re-rendered, dramatically increasing programmer workload. For simple projects, PHIGS was considerably easier to use and work with.

However, OpenGL's "low-level" API allowed the programmer to make dramatic improvements in rendering performance by first examining the data on the CPU-side before trying to send it over the bus to the graphics engine. For instance, the programmer could "cull" the objects by examining which objects were actually visible in the scene, and sending only those objects that would actually end up on the screen. This was kept private in PHIGS, making it much more difficult to tune performance.

Given the low performance systems of the era and the need for high-performance rendering, OpenGL was generally considered to be much more "powerful" for 3D programming. PHIGS fell into disuse. Version 6.0 of the PEX protocol was designed to support other 3D programming models as well, but did not regain popularity. PEX was mostly removed from XFree86 4.2.x (2002) and finally removed from the X Window System altogether in X11R6.7.0 (April 2004).
OpenGL specifies an architecture

OpenGL is an API-based architecture specification
- Low-level imperative interface (like an ISA)
- Tight specification (like an ISA)
- Orthogonal operation (like an ISA)

This is what mattered most
- Allowed hardware innovation
- Allowed application innovation
- Allowed programmability
- Deferred need for “shaders”, then allowed them

“In terms of stifling innovation, good ideas are far more dangerous than bad ones. They take hold, assume momentum, and therefore result in inertia. Consequently, they are hard to displace, even when they are well past their prime.”

— Bill Buxton
In Sketching User Experience, Morgan Kaufmann, 2007
To learn more

www.opengl.org
www.khronos.org

Real-Time Graphics Architecture

Lecture 16: The Design of OpenGL

Kurt Akeley
Pat Hanrahan
http://graphics.stanford.edu/cs448-07-spring/