Innovative Software Architecture
for
Real-Time Image Generation

Rémi Arnaud
Director, Developer Relations

Michael T. Jones
Chief Technical Officer

Intrinsic Graphics, Inc.
www.intrinsic.com
Electronic Visual System in 1964

Astronauts Pete Conrad (L) and Al Bean (R) in the Lunar Module simulator

Lunar Module simulator Exterior View
Evans & Sutherland CT Series

*Separated system into three pieces*

- The database
  - *Modeler*
  - *Compiler*
- The visual subsystem
  - *Image Generator*
  - *Host interface via ICD*
- The display subsystem
General Purpose Graphic Workstation

Low-level graphic libraries:

- IRIS GL
- OpenGL
- Direct 3D
- Glide

OpenGL State Machine
Workstation Graphic Toolkits

**High-level graphic libraries:**

- OpenInventor
- IRIS Performer
- VTree
- VisKit
Scene Graph Flaws

Poor environment for code reuse

- Surreptitious extension model (callbacks and subclasses)
  - Can extensions learn about each other?
  - Can extensions negotiate with each other?

- What level for abstraction, virtualization, and extension

Clash of imperative and constraint styles

- Mirrors, shadows, refraction, paging

Struggle to be center of computation
Modern Image Generator Run-Time

Thomson Training and Simulation Space Magic/Space Basic
Simulation Run-Time Problems

Not a satisfactory solution

- Weak extension model
- Almost impossible to reuse code
- State duplication
- Multiple platforms and multiple operating systems
- No real help for display/database integration
Redefine Application Development

**Analyze and repackage:**

- Hardware-specific graphics APIs
- Popular hierarchical scene-graph APIs
- Fielded visual simulation run-time applications

**To enable and provide:**

- Portable platform for application execution
- Extensive set of reusable components
- Standard application framework
Evolution of Solution Structure

Past Graphics Environment
- Application Level
  - Simulation Run-Time Application
  - Intent Level
    - Declarative Level
      - Procedural Level
        - Three-Level Developer Task
          - IRIS GL
            - StarBase
              - XGL
          - Standard and Custom Features
            - IRIS Performer, VisKit, OpenGVS, VTree, 596 more
          - Standard Retained-Mode APIs
            - OpenGL
              - Direct 3D
                - Glide
              - Bare Metal
            - Standard Immediate-Mode APIs

Present Graphics Environment
- Application Level
  - Simulation Run-Time Application
  - Intent Level
    - Declarative Level
      - Procedural Level
        - Two-Level Developer Task
          - IRIS GL
            - StarBase
              - XGL
          - Standard and Custom Features
            - IRIS Performer, VisKit, OpenGVS, VTree, 596 more
          - Standard Retained-Mode APIs
            - OpenGL
              - Direct 3D
                - Glide
              - Bare Metal
            - Standard Immediate-Mode APIs

Future Graphics Environment
- Application Level
  - Application Graph
  - Intent Level
    - Declarative Level
      - Procedural Level
        - One-Level Developer Task
          - Standard Application
            - OpenGL
              - Direct 3D
                - Glide
              - Bare Metal
            - Standard Immediate-Mode APIs
Application Platform

- Execution strategy and schedule generation
- Multi-threaded/multi-processed component execution
- Execution profiling and performance instrumentation
- Data, synchronization, and resource management
Application Components

Hierarchy of generality

- Standard features
  - I/O, media, math
- Market-oriented features
  - VASI, FLOLS, smoke
  - FLIR, universal texture
- Application-specific features
  - F-18 HUD, threat logic
Component Definition Tools

(block
  ; Identification
  (identifiers 'name)
  (name 'iFormula)
  ; Interface
  (input (name 'x)(type 'iFloat))
  (input (name 'y)(type 'iFloat))
  (output (name 'f)(type 'iFloat))
  ; Implementation -- internal blocks and connections
  (block (name 'm1) (objectRef (block (name 'iMultiplier))))
  (block (name 'm2) (objectRef (block (name 'iMultiplier))))
  (block (name 'a) (objectRef (block (name 'iAdder))))
  (block (name 'two) (objectRef (block (name 2))))
  (connection (from 'x)(to 'm1 'left))
  (connection (from 'y)(to 'm1 'right))
  (connection (from 'x)(to 'm2 'left))
  (connection (from 'two 'out)(to 'm2 'right))
  (connection (from 'm1 'product)(to 'a 'left))
  (connection (from 'm2 'product)(to 'a 'right))
  (connection (from 'a 'sum)(to 'f))
)
Application Framework

**Standard application structures**

- Driver, train, armored vehicle, low-level flight, etc.

**Standard application sub-structures**

- Display configuration and management
- Behavior, morphing, dynamics, intersection
- Database paging, decoding, and formatting
- Graphics (culling, occlusion, optimization, drawing)
- Audio and video media (stream, buffer, process)
Adaptive Software

Alternate component implementations

- Variations in resource usage and performance
- Vendor-tuned hardware-centric implementations

Algorithmic negotiation and selection

- Compiler for “Integration Phase” of simulation projects

Algorithmic MP & scheduling of execution

- Self-adapting software enables retroactive upgrades
- Application structure reshaped by database and hardware
**Intrinsic Vision**

*Publishable graphics applications*

- Portability with excellence

*Publishable graphics content*

- Adaptive appearance and behavior

*Champion innovative hardware*

- Retroactive enhancement

*An intrinsic Graphical Application Platform*
Enable *publishable applications*

- Portable at/near peak efficiency
- Tuned *once* by hardware vendors
  - CPU, Graphics, Audio, Video
- Strategy compiled at run-time
- New concept in software reuse
GAP: Database Interoperability

Enable publishable content

- Compiled high-level definitions
  - Multi-resolution geometry
  - Arbitrary resolution textures
  - Programmable shading
- Portable negotiating behavior
  - Content-driven resource negotiation
  - Behavior code bundled with geometry
GAP: Hardware Adaptivity

**Champion innovative hardware**

- Replaceable HW-specific modules
  - *Feature-level device drivers*
- Modules shape application strategy
- Retroactive enhancement
  - *New features for old binaries*
  - *Choose the hardware at the last minute*
  - *Developer predefines future reshaping*
- Support business model of hardware innovators