Virtual Reality Technology and Programming

TNM053:
Lecture 5: Real Time Graphics

Real Time Graphics

- What is it?
- Software
- Hardware
- Performance Issues
- Optimization

What is Real-Time Graphics?

- Interactivity
  - Contents of the next frame depend on user’s actions
- Low Latency
  - Refresh Rate ~15-30Hz
  - Required calculation for next frame must be done in the inter-frame gap!

Software for Real-Time Graphics

- Low Level Software
  - OpenGL
  - Direct3D
- Toolkits and Systems for Scientific Visualization
  - AVS/Express
  - Visualization Toolkit (VTK)
- Scene-based Software
  - Inventor - SGI
  - VRML/Web3D
  - Java3D
  - Performer - SGI

Types of rendering

- Primitives
  - Pixels
  - Polygons
- Surface rendering
- Volume Rendering
- Image-(based) Rendering

Graphics pipeline

Application
Geometry
Coordinate transformations
Clipping
Rendering – illumination
Shading, hidden surface removal
Image
Application

- Geometry list
- Movement of objects, and aiming and movement of view camera
- Animated movement of object models
- Description of the contents of the 3D world
- Object Visibility Check (Occlusion Culling?)
- Select Level of Detail

Geometry operations

- Transforms (rotation, translation, scaling)
- Transform from Model Space to World Space
- Transform from World Space to View Space
- View Projection
- Trivial Accept/Reject Culling
- Back-Face Culling (can also be done later in Screen Space)
- Lighting
- Perspective Divide - Transform to Clip Space
- Clipping
- Transform to Screen Space

Triangle reduction

- Back-face Culling (or can be done in view space before lighting)
- Slope/Delta Calculations
- Scan-Line Conversion

Rendering / Rasterization

- Shading
- Texturing
- Fog
- Alpha Translucency Tests
- Depth Buffering
- Antialiasing (optional)
- Display

Summary

- Graphics pipeline:
  - Many, many functions
  - Very mathematics intensive
- Requires a great deal of power
- Many operations are formulaic
- Not application dependent

Solution: hardware graphics

- Application (CAD, games, etc.)
- API (Direct3D, OpenGL)
- Hardware Device Driver
- 3D Acceleration Hardware
Hardware for Real-Time Graphics

- PCs
  - SGI Visual PC
  - Intergraph
  - Graphics Cards: ATI, Matrox, NVidia...
- Workstations
  - SGI O2/Octane, Sun Ultra 10 with Elite3D
- High Performance systems
  - SGI Onyx2 InfiniteReality2, E&S
- Game Consoles
  - DreamCast, PS-2, MS X-Box

OpenGL...in 3 minutes

- Low Level
  - Accesses hardware directly
- State Machine
  - Matrix stacks and attributes
- Framebuffer
  - Multisample algorithms, invariance
- Not Programmable (but has 'extensions')
- At least partially) implemented in hardware
- Geometry and images (textures)
- Immediate Mode and Display Lists
  - Optimization, hierarchies
- Depth Buffer
- Illumination, Local Shading (Gouraud only)
- Rendering Only - Window System independent
- API not 'Protocol' – Well defined by code

OpenGL API examples (1)

- Immediate Operations:
  - `glClearColor(0.0, 0.0, 0.0, 0.0);`
  - `glMatrixMode(GL_PROJECTION);`
  - `glLoadIdentity();`
  - `glOrtho(0.0, 1.0, 0.0, 1.0, --1.0, 1.0);`

OpenGL API examples (2)

- State Management:
  - `glClearColor(0.0, 0.0, 0.0, 0.0);`
  - `glMatrixMode(GL_PROJECTION);`
  - `glLoadIdentity();`
  - `glOrtho(0.0, 1.0, 0.0, 1.0, --1.0, 1.0);`

OpenGL API examples (3)

- Display Lists:
  - `glNewList(TheTorus, GL_COMPILE);`
  - `glBegin(GL_QUAD_STRIP);`
  - `glVertex3f(...);`
  - `glEndList();`
  - `glCallList(theTorus);`

OpenGL Block Diagram
Accelerated Graphics

- What is accelerated?
- Lines and polygons
- Flat Surfaces
- Parametric lines and surfaces - Splines
- Illumination and/or Local Shading
- Depth (Z-) Buffering for occlusion
- Double Buffering
- Picking
- Textures
- Multi-Sampling or Anti-Aliasing

Realizing OpenGL on...

- SGI Onyx Infinite Reality 2
  - High End
  - Performance, performance, performance
  - Expandability
- SGI O2
  - Low end
  - Price/Performance

Onyx IR2 - Features

- 2.62 Million Pixels.pipe (10.48 Million/system)
- Multisampling and antialiasing
- CLIP mapping
- Projective textures
- Shadow Maps
- Convolution
- Dynamic Video Resolution (DVR)
- Genlock (generator locking device)
- Swap Synchronization for multiple pipes

Onyx IR2 - Architecture

- Graphics 'pipe' made up of a mix of:
  - Transform manager Board(s)
  - Raster Manager Board(s)
  - Display Generator Board(s)

Transform Manager Board
(2 or 4 TE's per board)

Raster Manager Board
(1,2 or 4 per pipe)
**Display Generator Board (Option for 8 channels)**

- Video Bus
- Video Buffer
- Video Processor
- Video Processor
- Video Processor
- Video Processor
- RAMDAC

**OpenGL on IR2**

- Transformation Engine
- Vertex Operations
- Pixel Operations
- Texture Memory

**So...**

- Entire OpenGL pipeline is Hardware!
- Each pipe can accept up to 300MB/s of OpenGL commands
- TM system can process 11M ?/s
- Pixel Generator can rasterize 12M ?/s
- Fragment processor can produce 800M textured pixels/s

**And there's more...**

- System is multi-pipe
- 'Reality Monster' configuration:
  - 16 R10K processors
  - 8 fully configured pipes
- Multi-pipe SDK provides 'simple' interface to drive multi-pipe system

**SGI O2 - System-Level Architecture**

- Imaging & Compression Engine
- R10K or R10K00 CPU
- Secondary Cache
- SGI Main Memory

**OpenGL on O2**

- Geometry
- Display List
- Pixel Operations
- Texture Memory
- Frame Buffer/Selection

---

5
Comparison: highend - low end

Performance Issues
- Bottlenecks
- Optimization
- Volume rendering issues
- Image-based methods

Potential Performance Bottlenecks
- Data Delivery
  - Host to rendering pipeline
- Geometric processing
  - Polygon generation
- Pixel Processing - Fill rate
- Texture processing

Optimizing (Graphics) applications
- Put the system and application in real-time mode for real-time performance
- Don't let the host be the bottleneck
- Use the fastest interface and routines when communicating with the graphics pipeline
- Design with multiprocessing in mind.
  - (and multi-pipe)

Optimizing (Graphics) applications - 2
- Isolate bottleneck stages
- Tune the slowest stage of the pipeline
- Order rendering to benefit the most expensive pipeline stage
- Fill the pipeline from back to front

Optimizing Graphics Applications
- Use knowledge about the application:
  - Partitioning the Scene
  - Bounding Box
  - Clipping
  - Culling
- Level of detail (LOD)
**LOD - Motivation**

- High complexity of 3D scene
  - High definition surfaces in CAD
  - Digital terrain models
- Reduce complexity:
  - when it makes least difference
  - Where it is least obvious

**LOD - Simplification**

- Reduce resolution with distance
- Reduce resolution with motion
- Approximation
  - On-the-fly simplification (-Progressive?)
- Multiresolution model
  - Coplanar facet merging
  - Retiling
  - Clustering

**LOD - Approximation**

- Problems
  - Preserving details
  - Preserving boundaries (e.g. cracks)
- Error evaluation
  - Quantitative constraints on approximation

**Texture maps and Multi-rendering**

- Powerful tools to approximate many features too expensive to implement.

**Direct Volume rendering**

- Ray Tracing
- Ray Casting (Hardware supported)
- Cheat: Use 3D textures (as in IR3)
- Currently ‘interactive’ framerate ~1Hz
- VolumePro DVR hardware

**Image-based rendering**

- Interactive 3D applications exhibit enormous inter-frame coherence
- Cost of Image Warp is independent of scene complexity
- Every Nth frame is conventionally rendered
- Transparent and moving objects are difficult
Summary

- Very expensive hardware is good enough for VR
  - But nowhere near as good as the eye
- Cheap (PC) hardware is getting better but still can't do the job
- There are a lot of optimization methods to apply to ensure good performance
- Volume rendering still a problem in VR applications
- If you can't optimize it, avoid it!