Overview of the Tutorial – Morning

09.30 – 10.30  Introduction to the Tutorial  Thomas Ertl
10.30 – 11.00  Low-Level Vertex Shader Programming  Martin Kraus
11.00 – 11.30  Coffee Break
11.30 – 12.00  Low-Level Pixel Shader Programming  Martin Kraus
12.00 – 12.45  High-Level Shading Languages  Daniel Weiskopf
12.45 – 14.30  Lunch Break

Overview of the Tutorial – Afternoon

14.30 – 15.15  Advanced Shading Techniques  Joachim Diepstraten
15.15 – 16.00  Non-Photorealistic Rendering  Mike Eißele
16.00 – 16.30  Coffee Break
16.30 – 17.15  Hardware-Based Volume Ray Casting  Manfred Weiler
17.15 – 17.45  Flow Visualization  Daniel Weiskopf

Interactive Computer Graphics

- scene: polygonal objects (triangle mesh)
- image: raster image of pixels (true color)
- Interactive graphics:
  - fast processing of the pipeline (>10 frames/s)
  - in spite of high scene complexity (millions of triangles)
  - realistic illumination effects and material properties
  - use of hardware acceleration for geometry and rasterization

Texturing

- Pasting of images onto geometry
- Assigning texture coordinates of the image to vertices of the geometry
- For each pixel: bilinear interpolation from surrounding texels
- Hardware acceleration provides texture mapping without delay

Multi-Textures

Light maps in Quake2

- Precomputed Illumination
- Surface Structure
- Light Map Texture
- Decal Texture

Combine 2 textures onto scene geometry
## Graphics Hardware Characteristics

- **Performance characteristics**
  - Geometry: shaded triangles per second >> 10 Mio
  - Rasterization: fill rate in pixels per second >> 100 Mio

- **Computational requirements: geometry subsystem**
  - ca. 100 FLOPs per vertex (about 30 for T&L each)
  - 10 Mio. triangles/s T&L performance need 3 GigaFLOPs however only 500.000 triangles in the scene at 20 Hz!

- **Computational requirements: raster subsystem**
  - >10 operations per pixel (without special texturing!)
  - 100 MegaPixel/s fill rate need 1000 MIPS performance
  - at 20Hz and 10 pixel/triangle: 500.000 tris per frame
  - for a 1Kx1K frame buffer 5-fold overdraw of each pixel

## Graphics Hardware Trend

- Faster development than Moore’s law
  - Double transistor functions every 6-12 months
  - Driven by Game industry
- Improvement of performance and functionality
  - Textures, Multi-textures, texture shaders
  - Pixel operations (transparency, blending, pixel shaders)
  - Geometry and lighting modifications (vertex shaders)

## High-end Cards – Characteristics

<table>
<thead>
<tr>
<th>Brand</th>
<th>Transistors</th>
<th>Technology</th>
<th>Clock rate</th>
<th>Mem bandwidth</th>
<th>Fill rate (peak)</th>
<th>Pixel Pipelines</th>
<th>Textures per Unit</th>
<th>Bits per channel</th>
<th>Tri transform (peak)</th>
<th>Tris (3Dmark)</th>
<th>Vertex shaders</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATI Radeon 9800</td>
<td>107 Mio</td>
<td>0.15 micron</td>
<td>380 MHz</td>
<td>22 GB/s</td>
<td>3 GigaPixel/s</td>
<td>8</td>
<td>8</td>
<td>10</td>
<td>380 Mio</td>
<td>19 Mio</td>
<td>4</td>
</tr>
<tr>
<td>NVIDIA GeForceFX 5900</td>
<td>130 Mio</td>
<td>0.13 micron</td>
<td>27 GB/s</td>
<td>1.8/3.6 GigaPixel/s</td>
<td>4/8</td>
<td>16</td>
<td>16</td>
<td>315 Mio</td>
<td>315 Mio</td>
<td>28 Mio</td>
<td>4+</td>
</tr>
</tbody>
</table>

## 20 Years of Graphics Hardware

- **1980s**: Simple rasterization (bitBLT, windows, lines, polygons, text fonts)
- **1990-95**: Geometry engines only for high-end workstations (e.g. SGI O2 vs. Indigo2)
- **1995**: New rasterization functionality (realism with textures) z.B: SGI Infinite Reality
- **1998**: Geometry processing (T&L) for PC graphics cards
- **2000**: PC graphics reaches high-end performance numbers, 3D becomes PC standard
- **2001**: PC graphics offers additional functionality (multi-texturing, vertex and pixel shaders)
- **2003**: Shading Languages: NVIDIA Cg, OpenGL 2.0, DX9 GPUs > 100 Mio. transistors, 8 Pipes and 16 texture units
From Configuration to Programming

• Configurability:
  Select hardware processing options by state changes
  – T&L: various texture generation modes
  – Rasterization: imaging subset
  – Fragment processing: various blending modes

• Programmability:
  Download small assembly programs to change hardware behavior
  – T&L: vertex shaders
  – Rasterization: texture shaders
  – Fragment processing: pixel and fragment shaders

Programmable Processors (from NVIDIA Cg Manual)

• 2 or more programmable processors per GPU
• Fixed pipeline (with configuration) remains where no flexibility is necessary (or possible)

OpenGL 2.0 Pipeline (from 3Dlabs presentation)

Vertex Shaders

• Programmable transformation & lighting
  – Register architecture with up to 128 instructions
  – Replaces standard transformation pipeline and Phong lighting
  – Special perspective projections (lens effects)
  – Advanced lighting models
  – Automatic generation of texture coordinates
  – Procedural geometry, morphing, skinning, ...

Scientific Visualization – Historic Examples

Modern Scientific Visualization

• Traditional plotting techniques are not appropriate for visualizing the huge datasets resulting from
  • computer simulations (e.g. CFD, physics, chemistry, ...)
  • sensoric measurements (e.g. medical, seismic, satellite)

  "The purpose of computing is insight not numbers"

• Map abstract data onto graphical representations
• Try to use colorful 3D raster graphics in
  • expressive still images
  • recorded animations
  • interactive visualizations

  "To see the unseen"
**Visualization – Pipeline and Classification**

Visualization pipeline: sensors, data, renderable, visualization, images/videos.


Different grid types → different algorithms.

**Visualization – Examples**

- Height fields
- Stream ribbons
- Isosurfaces

**Interactive Visualization of Huge Datasets**

- CFD, FE, CT, MR, PET
- Simulation: sensors, raw data, visualization.
- Raw data → visualization data → renderable representation → visualization.

- Too much data → too many cells → too many triangles.
- Optimization of all steps of the visualization pipeline.

**Graphics HW and Interactive Visualization**

- First: Mapping generates polygonal geometry only, colored, lighted and shaded (e.g., isosurfaces, stream ribbons, glyphs).
- From 1995: Advanced rasterization functionality, textures and transparency (e.g., LIC, volume rendering).
- From 2000: Multi-textures and register combiners.
- From 2002: Texture shaders and vertex shaders.
- In the future: Shading languages for visualization.
- Trend: Graphics hardware on its way up through the visualization pipeline towards the data.

**Graphics HW and VIS Pipeline Stages**

- **Renderer**
  - Texture-based techniques (3D textures, LIC, ...)
  - Large textured terrain height fields.
- **Mapper**
  - Classification & transfer functions in volume rendering
  - Integrate ray segments (in unstructured volumes)
  - Integrate particle traces (in flow fields)
  - Assign color and transparency for NPR.
- **Filtering**
  - Data filtering in graphics memory (e.g., wavelet)
  - Compression/decompression (of textures).

**Prog. Graphics HW and VIS Applications**

- End users of VIS still use classical Unix workstations (no programmable graphics HW).
- VIS applications (pre & post processing, toolkits, MVEs) are cross-platform, use minimum func.
- Texturing and transparency are „advanced“.
- Exception: volume rendering
  - Doctors can afford PCs, no Unix workstations
  - Regular data structures profit most
  - Improvements are significant.
Volume Visualization

- Abstract 3-dimensional datasets
- X-ray absorption in material
- Humidity in the atmosphere
- Density distribution in the earth
- Data often given on uniform 3D grid
  - millions of cells (voxel)
- Problem: occlusion

Volume Visualization

- Focus on 3D scalar fields (e.g., medical data)
  - Some concepts extend to non-cartesian grids, vector fields...
- Isosurfaces
  - Reconstruction of polygonal surfaces with Marching Cubes
  - Fast rendering with OpenGL standard hardware
  - Non-interactive for huge datasets (millions of triangles)
- Direct volume rendering
  - For each pixel send a ray into the volume
  - Sample volume along ray by interpolation
  - Semi-transparent blending along rays
  - Transfer functions for color and opacity provide "segmentation" of structures
  - Interactivity even for many trilinear interpolations with hardware support (dedicated or 3D textures)

Volume Visualization of Medical Datasets

- 2D visualization slice images (MPR)
- Indirect 3D visualization isosurfaces (SSD)
- Direct 3D visualization volume rendering (DVR)

Volume Rendering of Medical Datasets

- Different transfer functions

Textures in CAE Visualization

- Color coding of scalar entities with 1D texture lookups
- Intrusion depth of crash-worthiness simulations
- Transparency for detecting numerical instabilities
- Assembly of finite element models
Wireframe Rendering by Textures

Detection of Flanges – Transparent Texture

Stack of Semi-transparent Slice Planes

Texture-based Flow Visualization

- LIC (Line Integral Convolution)
  - Transfer directional information of a vector field into a noise texture
  - High correlation in the direction of stream lines, no correlation orthogonal
  - Global visualization method
  - Computationally expensive, fast rendering

Programming Graphics Hardware

Let's jump into the details!