Eurographics 2003
Tutorial T7
2. September 2003

Programming Graphics Hardware

Thomas Ertl

Institute of Visualization and Interactive Systems
University of Stuttgart

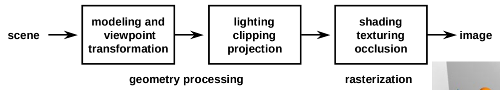
Overview of the Tutorial – Morning

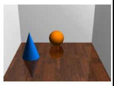
09.30 – 10.30	**Introduction to the Tutorial**	**Thomas Ertl**
10.30 – 11.00	**Low-Level Vertex Shader Programming**	**Martin Kraus**
11.00 – 11.30	***Coffee Break***	
11.30 – 12.00	**Low-Level Pixel Shader Programming**	**Martin Kraus**
12.00 – 12.45	**High-Level Shading Languages**	**Daniel Weiskopf**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

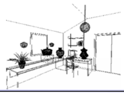
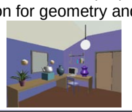
Interactive Computer Graphics





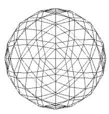
scene: polygonal objects (triangle mesh)
image: raster image of pixels (true color)

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

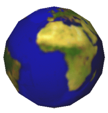




Texturing

- 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







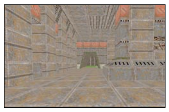
Multi-Textures

Light maps in Quake2

Precomputed Illumination

× (modulate)

Surface Structure



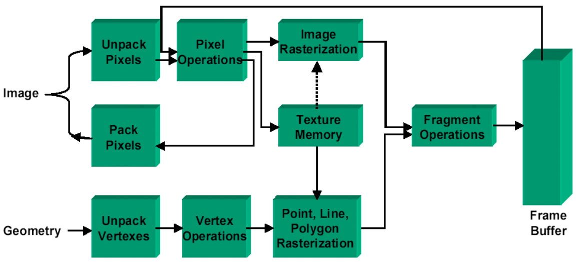
Light Map Texture

=

Decal Texture

Combine 2 textures onto scene geometry

OpenGL Pipeline (by Kurt Akeley)

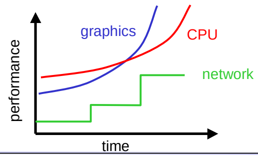


Graphics Hardware Characteristics

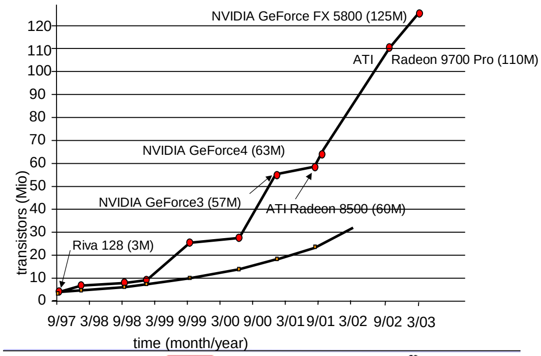
- **Performance characteristics**
  - Geometry: shaded triangles per second >> 10 Mio
  - Rasterization: fill rate in pixels per second >> 100 Mio
- **Computational requirements: geometry subsystem**
  - ca. 100 FLOPs per vertex (about 30 for T&L each)
  - 10 Mio. triangles/s T&L performance need 3 GigaFLOPs 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)



Transistor Functions



High-end Cards – Characteristics

Brand:	ATI Radeon 9800 P	Nvidia GeforceFX 5900 U
• Transistors	107 Mio	130 Mio
• Technology	0.15 micron	0.13 micron
• Clock rate	380 MHz	450 MHz
• Mem bandwidth	22 GB/s	27 GB/s
• Fill rate (peak)	3 GigaPixel/s	1.8/3.6 GigaPixel/s
• Pixel Pipelines	8	4/8
• Textures per Unit	8	16
• FSAA	6x 18 Gsample/s	4x 27 Gsample/s
• Bits per channel	10	10
• Tri transform (peak)	380 Mio	315 Mio
• Tris (3Dmark)	19 Mio	28 Mio
• Vertex shaders	4	4+

www.tomshardware.de

20 Years of Graphics Hardware

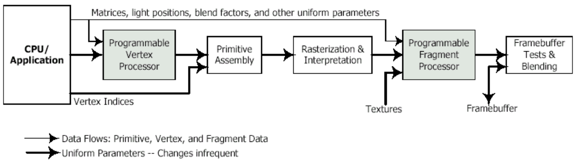
- **1980s**: Simple rasterization (bitBLT, windows, lines, polygons, text fonts)
- **1990-95:** Geometry engines only for high-end workstations (e.g. SGI O2 vs. Indigo2)
- **1995:** New rasterization functionality (realism with textures) z.B: SGI Infinite Reality
- **1998:** Geometry processing (T&L) 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

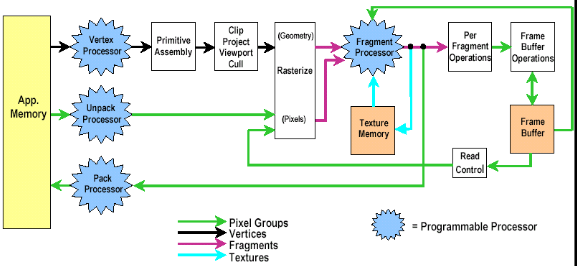
- **Configurability**:
  Select hardware processing options by state changes
  - T&L: various texture generation modes
  - Rasterization: imaging subset
  - Fragment processing: various blending modes
- **Programmability**:
  Download small assembly programs to change hardware behavior
  - T&L: vertex 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)



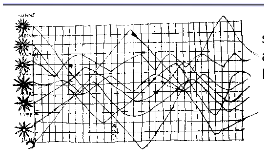
OpenGL 2.0 Pipeline *(from 3Dlabs presentation)*



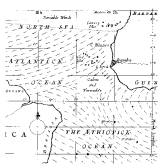
Vertex Shaders

- **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, ...

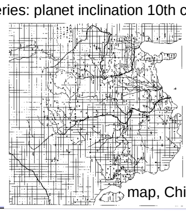
Scientific Visualization – Historic Examples



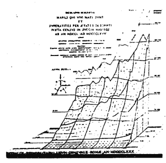
time series: planet inclination 10th century



stream lines, arrow plots: Halley 1686



map, China 11th century



height fields: census data, Perozzo 1879

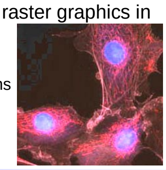
Modern Scientific Visualization

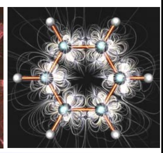
- Traditional plotting techniques are not appropriate for visualizing the huge datasets resulting from
  - computer simulations (e.g. CFD, physics, chemistry, ...)
  - sensoric measurements (e.g. medical, seismic, satellite)

  „The purpose of computing is insight not numbers“

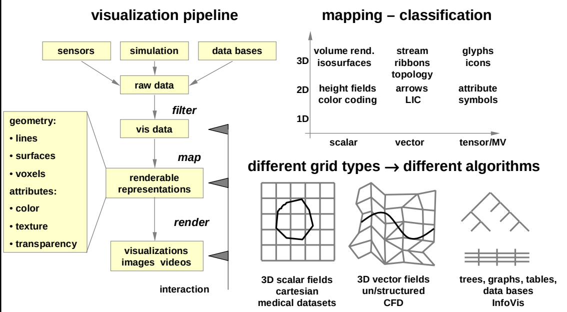
- Map abstract data onto graphical representations
- Try to use colorful 3D raster graphics in
  - expressive still images
  - recorded animations
  - interactive visualizations

  „To see the unseen“



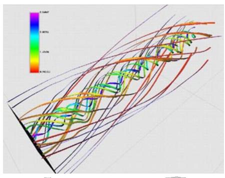


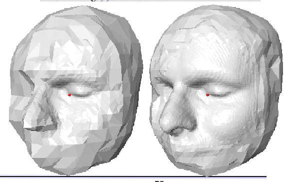
Visualization – Pipeline and Classification



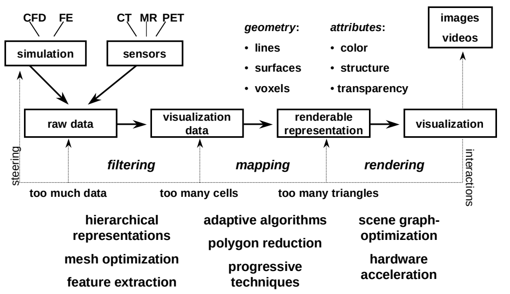
Visualization – Examples

- Height fields
- Stream ribbons
- Isosurfaces





Interactive Visualization of Huge Datasets



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

Images → Renderer ⇒ Mapper ⇒ Filter → Data

Graphics HW and VIS Pipeline Stages

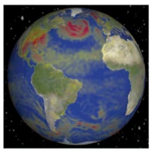
- **Renderer**
  - Texture based techniques (3D textures, LIC, ...)
  - Large textured terrain height fields
- **Mapper**
  - Classification & transfer functions in volume rendering
  - Integrate ray segments (in unstructured volumes)
  - Integrate particle traces (in flow fields)
  - Assign color and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

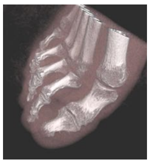
Prog. Graphics HW and VIS Applications

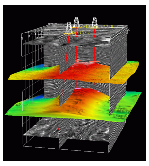
- End users of VIS still use classical Unix workstations (no programmable graphics HW)
- VIS applications (pre- & post processing, toolkits, MVEs) are cross-platform, use minimum funct.
- Texturing and transparency are „advanced“
- Exception: volume rendering
  - Doctors can afford PCs, no Unix workstations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

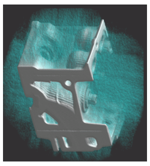
Volume Visualization

- 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



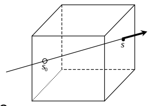




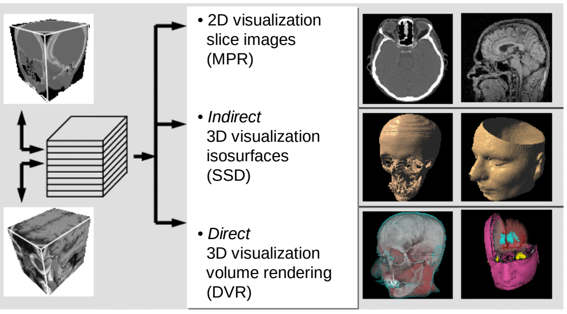


Volume Visualization

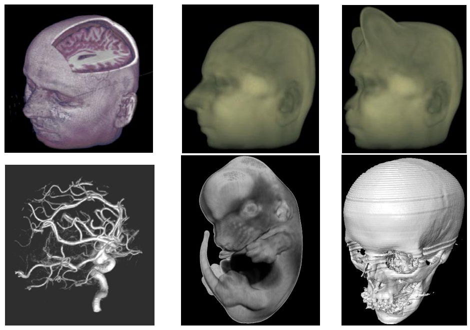
- Focus on 3D scalar fields (e.g. medical data)
  some concepts extend to non-cartesian grids, vector fields,...
- Isosurfaces
  - reconstruction of polygonal surfaces with Marching Cubes
  - fast rendering with OpenGL standard hardware
  - non-interactive for huge datasets (millions of triangles)
- 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)



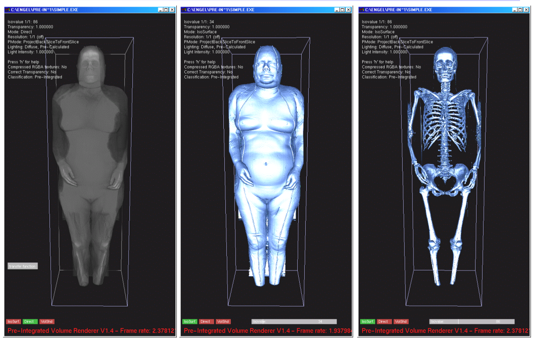
Volume Visualization of Medical Datasets



Volume Rendering of Medical Datasets

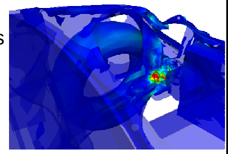


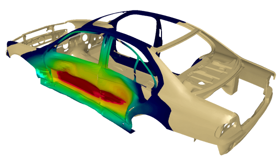
Different Transfer Functions

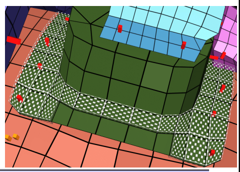


Textures in CAE Visualization

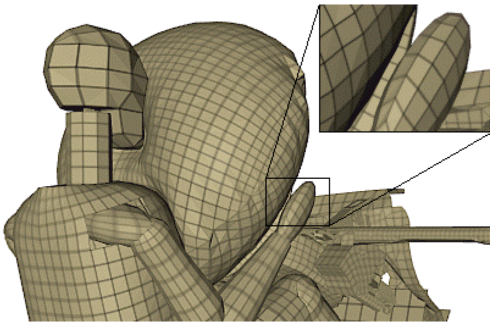
- Color coding of scalar entities with 1D texture lookups
- Intrusion depth of crash-worthiness simulations
- Transparency for detecting numerical instabilities
- Assembly of finite element models



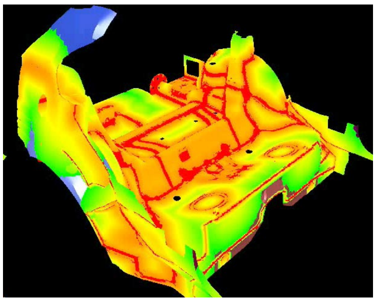




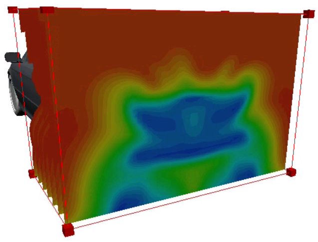
Wireframe Rendering by Textures



Detection of Flanges – Transparent Texture



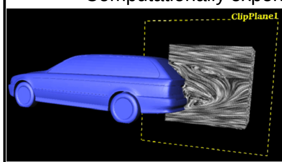
Stack of Semi-transparent Slice Planes

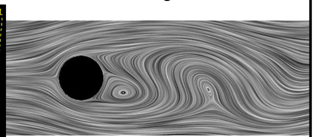


- Transpareny reduces occlusion of irrelevant data

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!