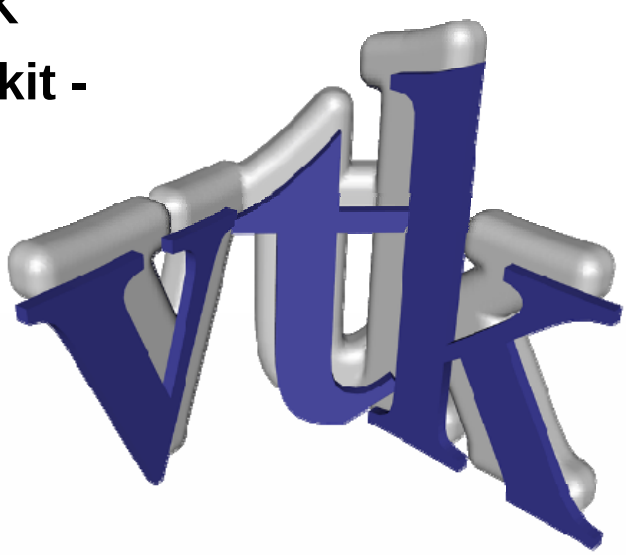
Visualisierung in VTK
- The Visualization Toolkit -

Steffen Oeltze



Otto-von-Guericke-Universität Magdeburg,
FIN/ISG

Inhalt:

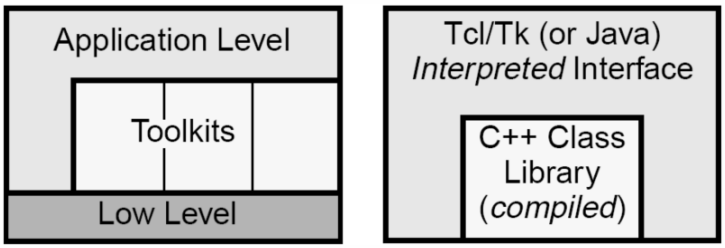
2D- und 3D-Visualisierung zur Exploration medizinischer Schichtdaten	(B. Preim, 15 Min.)
Oberflächenvisualisierung - Marching Cubes und seine Verbesserungen - Glättung von Oberflächenvisualisierungen	(B. Preim, 30 Min.)
Direkte Volumenvisualisierung - Raycasting und texturbasierte Ansätze - Projektionsmethoden	(B. Preim, 45 Min.)
Visualisierung in VTK und MeVisLab	(**S. Oeltze**, C. Tietjen, 30 Min.)
Modellbasierte Gefäßvisualisierung	(S. Oeltze, 20 Min.)
Illustrative Visualisierung	(C. Tietjen, 20 Min.)

Gliederung

- Was ist VTK?
- Was kann VTK?
- Die VTK – Visualisierungspipeline
- Ein VTK – Beispiel
- Zusammenfassung

Was ist VTK?

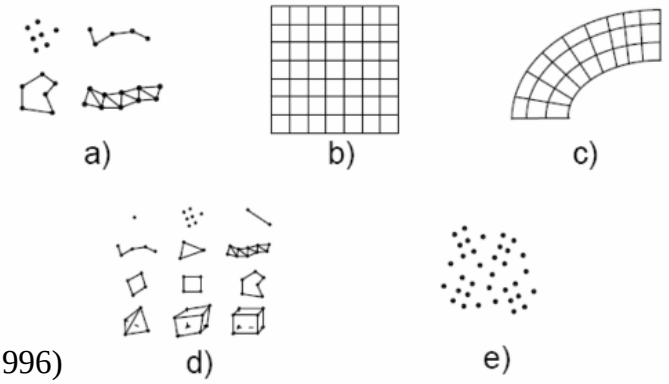
- Frei erhältliches *Open Source* Softwaresystem (*Kitware Inc.*)
- **Visualisierung,** 3D Computergraphik, Bildverarbeitung
- Implementiert für fast jede UNIX-Plattform, PC's (Windows 95/98/NT/2000/XP) und ab Mac OSX Jaguar aufwärts
- Objektorientiert (C++ Kern)
- Interpretersprachen Tcl/Tk, Python und Java für schnelle Applikationsentwicklung



(© Schroeder, 1996)

Was kann VTK?

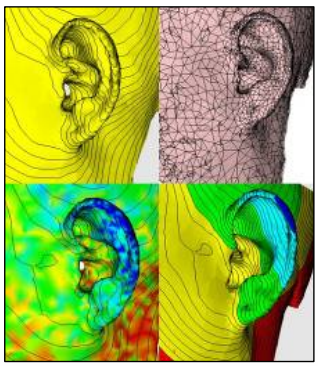
- Datentypen:
  - Polygonale Daten (a): Punkte, Linien, Polygone, Dreiecksnetze
  - Bild- und Volumendaten (b)
  - *Structured Grids (c)*
  - *Unstructured Grids (d)*
  - *Unstructured Points (e)*
  - *Rectilinear Grids*



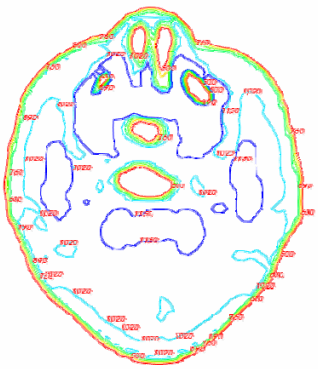
(© Schroeder, 1996)

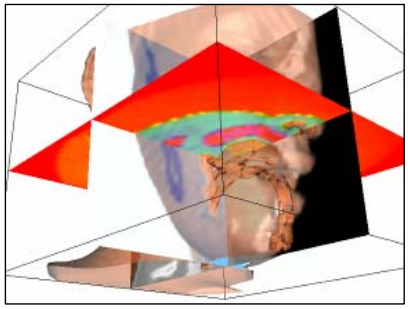
Was kann VTK?

- Visualisierungstechniken:
  - Skalare
    - Iso – Konturierung (2D/3D)
    - Abbildung auf Farbe



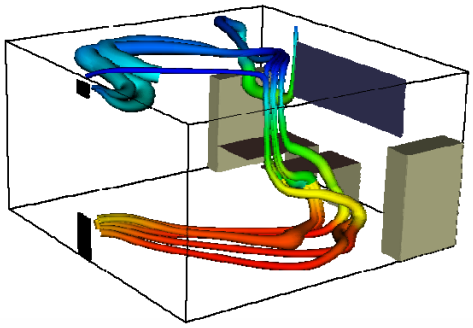
© GeoCap AS

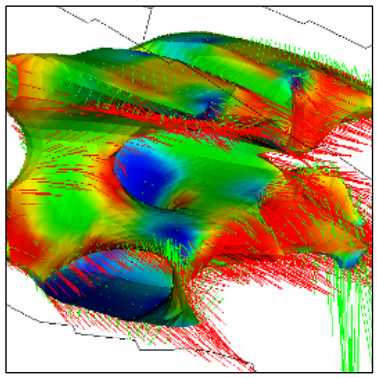


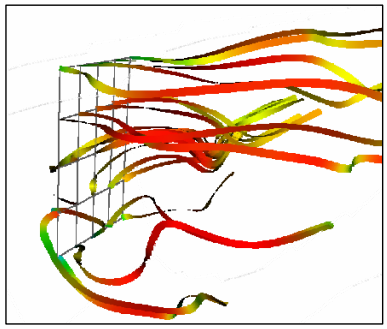


Was kann VTK?

- Visualisierungstechniken:
  - Skalare
    - Iso – Konturierung (2D/3D)
    - Abbildung auf Farbe
  - Vektoren
    - *Hedgehogs*
    - *Streamlines, -tubes, -ribbons*

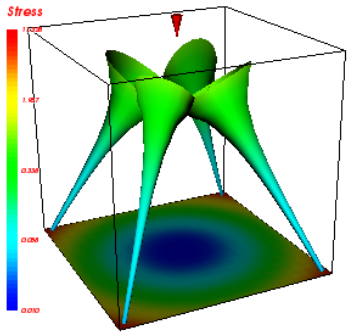


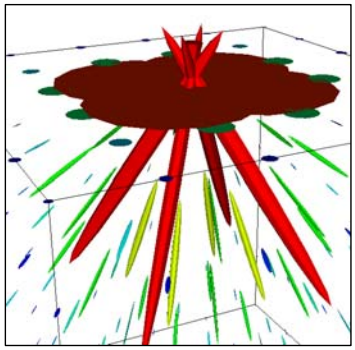




Was kann VTK?

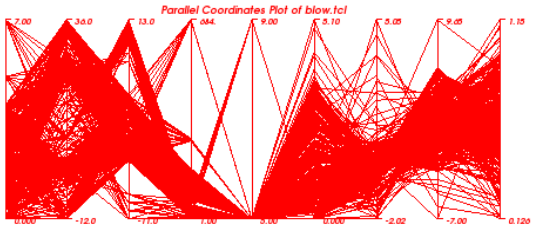
- Visualisierungstechniken:
  - Skalare
    - Iso – Konturierung (2D/3D)
    - Abbildung auf Farbe
  - Vektoren
    - *Hedgehogs*
    - *Streamlines, -tubes, -ribbons*
  - Tensoren
    - *Hyperstreamlines*
    - *Tensor Glyphen*

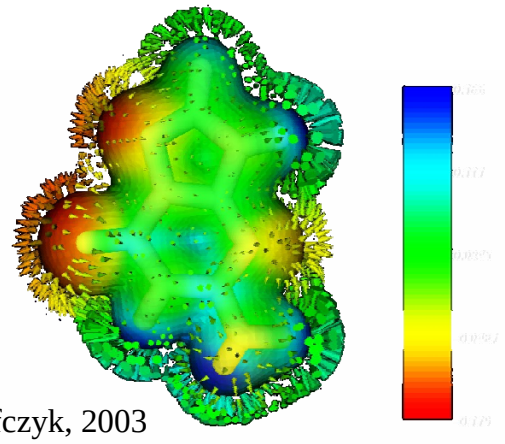




Was kann VTK?

- Visualisierungstechniken:
  - Skalare
    - Iso – Konturierung (2D/3D)
    - Abbildung auf Farbe
  - Vektoren
    - *Hedgehogs*
    - *Streamlines, -tubes, -ribbons*
  - Tensoren
    - *Hyperstreamlines*
    - *Tensor Glyphen*
  - Informationsvisualisierung
    - Parallele Koordinaten
    - Glyphen
- Modellierung:
  - Polygonreduktion
  - *Mesh Smoothing*

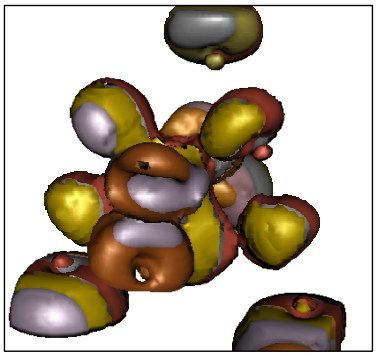




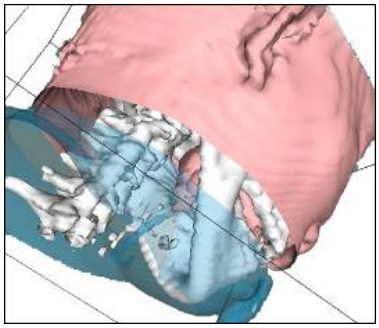
© Szefczyk, 2003

Was kann VTK?

- 3D – Computergraphik:
  - Oberflächenrendering
    - Isoflächen 3D-Rekonstruktion



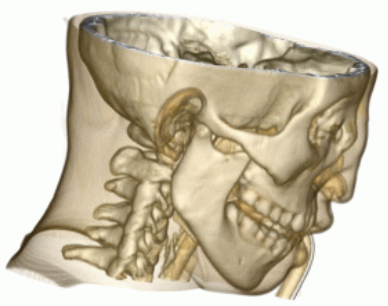
© Lorensen, 2001: „Visible Woman“



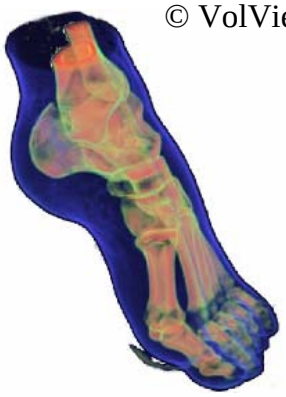
© GeoCap AS

Was kann VTK?

- 3D – Computergraphik:
  - Oberflächenrendering
    - Isoflächen 3D-Rekonstruktion
  - Volumenrendering
    - *Ray casting*
    - *Texture mapping* (2d)
    - *Volume Pro* Unterstützung
    - Mix aus Oberflächen- und Volumenrendering
  - Lichter und Kameras
  - Materialeigenschaften
  - Texturen
  - *Level of Detail* Unterstützung



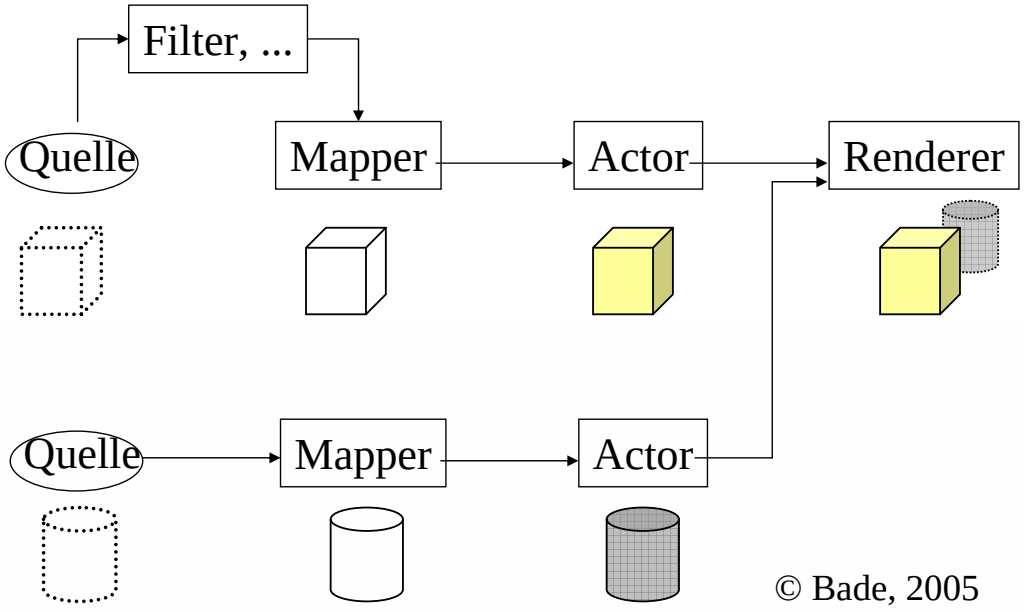
© VolView



Was kann VTK?

- Zusätzliche Funktionen:
  - Parallele Algorithmen, *Multithreading*
  - Stereounterstützung (Rot/Blau, Crystal Eyes, *Vertical Interlaced*)
  - *Motion / Focal blur*
  - Einfache Integration mit Motif, Qt, Tcl/tk, Python/Tk, X11, Windows, …
  - 3D *Widgets,* Interaktionsmöglichkeiten, Ereignisverwaltung
  - Schnittstellen für zahlreiche Datenformate, z.B.:
    - Inventor Writer, IV Exporters
    - 3D Studio Importer
    - SLC (Volume) Reader
    - VRML Exporter
    - Wavefront .OBJ Exporter, .OBJ Reader

Die VTK - Visualisierungspipeline



Ein VTK - Beispiel

- Isoflächen 3D-Rekonstruktion (Marching Cubes)

```
...
vtkVolume16Reader v16
   v16 SetDataDimensions 64 64
   [v16 GetOutput] SetOrigin 0.0 0.0 0.0
   v16 SetDataByteOrderToLittleEndian
   v16 SetFilePrefix "$VTK_DATA_ROOT/Data/headsq/quarter"
   v16 SetImageRange 1 93
   v16 SetDataSpacing 3.2 3.2 1.5

vtkMarchingCubes iso
   iso SetInputConnection [v16 GetOutputPort]
   iso SetValue 0 1150
   iso ComputeGradientsOn
   iso ComputeScalarsOff
```

Quelle

Filter

Ein VTK - Beispiel

- Isoflächen - *Fortsetzung*

```
vtkDataSetMapper isoMapper
   isoMapper SetInputConnection [iso GetOutputPort]
   isoMapper ScalarVisibilityOn
   isoMapper SetScalarRange 0 1200
   isoMapper ImmediateModeRenderingOn

vtkActor isoActor
   isoActor SetMapper isoMapper

vtkRenderer ren1
   ren1 AddActor isoActor
vtkRenderWindow renWin
   renWin AddRenderer ren1
vtkRenderWindowInteractor iren
   iren SetRenderWindow renWin
...
```

Mapper

Actor

Renderer

Zusammenfassung

+

- Frei erhältlich und *open source*
- Schnelle, einfache Entwicklung von Applikationen
- Objektorientiert – leicht zu erweitern
- Plattformunabhängig
- Umfangreiche Auswahl an aktuellen Algorithmen
- Unterstützung von Interpretersprachen Tcl, Python, Java
- Große Nutzergemeinde ermöglicht umfassenden Austausch
- Kommerzielle Unterstützung und Beratung durch *Kitware Inc.*

-

- Keine extrem schnelle Graphikengine aufgrund von Portabilität und C++ *dynamic binding*
- Riesige, schwer erfassbare Klassenhierarchie

- Website: www.vtk.org
  - Download von Quellcode und ausführbaren Dateien (V5.0.0)
  - Download von Beispieldaten
  - Dokumentation im HTML-Format
  - Mailing Liste
  - Wiki
  - FAQ

- Website: www.kitware.com
  - VTK Textbook
  - VTK User‘s Guide
  - Kommerzielle Produkte, z.B. *ParaView*
  - *ITK* (Insight Segmentation and Registration Toolkit)

MeVisLab
Medical Image Processing and Visualization

Visualisierung in MeVisLab

Christian Tietjen

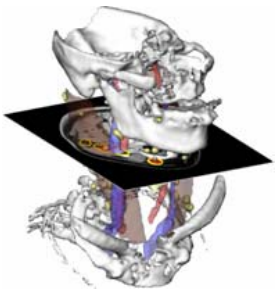
Otto-von-Guericke-Universität Magdeburg,
FIN/ISG

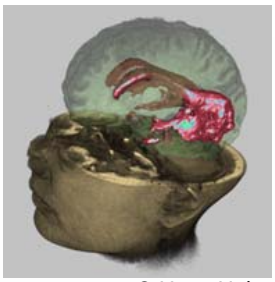
Gliederung

- Was ist MeVisLab?
- Eingebundene Software-Bibliotheken
- Rapid Prototyping
- Verfügbare Programmierebenen
- Vorführung

Was ist MeVisLab?

- Entwickelt von MeVis (Bremen)
- Entwicklungsumgebung für medizinische
  - Bildverarbeitung
  - Visualisierung
- Rapid Prototyping Umgebung (visuelle Programmierung)
- Freie Version Verfügbar unter http://www.mevislab.de/





© Horst Hahn

© Florian Link

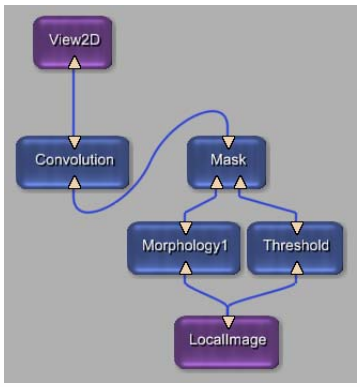
Was ist MeVisLab?

Eingebundene Software-Bibliotheken

- Bibliotheken zur Visualisierung
  - Open Inventor (Objektorientierter Wrapper für OpenGL)
  - VTK
  - Eigenentwicklungen von MeVis
- Bibliotheken zur Bildverarbeitung
  - ITK
  - ML (Eigenentwicklung von MeVis)
- Einbindung weiterer/eigener Bibliotheken
  - Xerces, OpenGL, CLAPACK…
  - OpenNPAR (http://www.opennpar.org/)

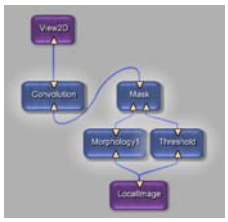
Rapid Prototyping

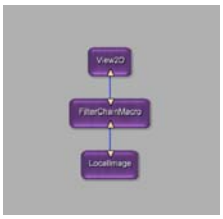
- Gemeinsamer Nenner eigentlich nur C++
- Graphische Benutzeroberfläche
- Kapselung der C++-Klassen in Module
  - Funktionalität einzelner Klassen lässt sich optisch miteinander verknüpfen
- Zusammenfassung von Modulen in Netzwerken
- Graphische Kapselung wird nur für ML und Open Inventor Module unterstützt

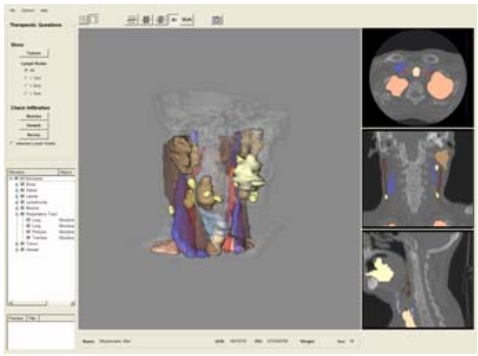


Verfügbare Programmierebenen

- Verwendung bereits verfügbarer Module
- Zusammenfassung in Makromodulen
- Dynamische Steuerung der Makromodule über Scripting
  - Python
  - JavaScript
- Gestaltung des Nutzer-Interface
  → Erstellung von Applikationen möglich
- Entwicklung eigener Module in C++







Vorführung