

Mit VTK von Daten zu Bildern

# Objektiv dargestellt



**Jürgen Schuck**

Bilder sagen bekanntermaßen mehr als Worte – aber wie aus Worten Bilder machen? Zum Beispiel mit der frei verfügbaren Klassenbibliothek VTK, für die es mittlerweile ein passendes GUI gibt.

Das Visualization Toolkit (VTK) ist ein frei verfügbares Softwaresystem für die 3D-Datenvisualisierung. Es entstand 1993 als Begleitsoftware eines Buches über einen objektorientierten Ansatz für 3D-Grafik [1] und hat sich seitdem zu einer umfassenden und universell einsetzbaren Software weiterentwickelt, die mittlerweile auch kommerzielle Unternehmen im Zusammenspiel mit ihren eigenen Produkten einsetzen.

Aus den Autoren des Buches, die seinerzeit bei General Electrics in Forschung und Entwicklung tätig waren, ist das Unternehmen Kitware, Inc. ([www.kitware.com](http://www.kitware.com)) hervorgegangen, das schlüsselfertige Produkte und Individuallösungen für Visualisierungsaufgaben anbietet. Kitware koordiniert außerdem die Weiterentwicklung und Wartung von VTK und bietet zusätzlich professionelle Beratungs-, Unterstützungs- sowie Ausbildungsdienstleistungen an.

VTK vereint zwei wichtige Trends der Computerindustrie Anfang der

90er-Jahre: Die zunehmende Entwicklung und Verbreitung objektorientierter Systeme und komplexe, grafische Benutzerschnittstellen, hierbei insbesondere die Verwendung von 3D-Grafik. Des Weiteren soll die Arbeit mit einem grafischen Visualisierungssystem so einfach wie möglich sein; sowohl für Anwender, die primär die Gegenstände ihrer eigentlichen Arbeiten im Blick haben, als auch für Implementierer, die grafische Subsysteme in ihre komplexen Anwendungen integrieren und über einfache, konsistente Schnittstellen ansprechen wollen.

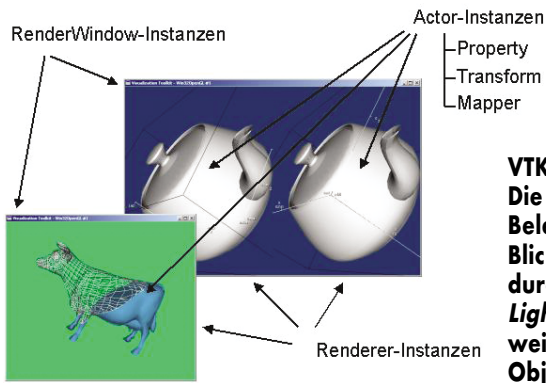
## Plattformspezifische Installation

Diese Leitgedanken führten zu einem Baukasten (Toolkit) in Form einer C++-Klassenbibliothek, die optional von einer Interpreterschicht für Tcl/Tk, Python und Java umgeben sein kann. Die VTK-Architektur ist einfach. Ihr Kern-

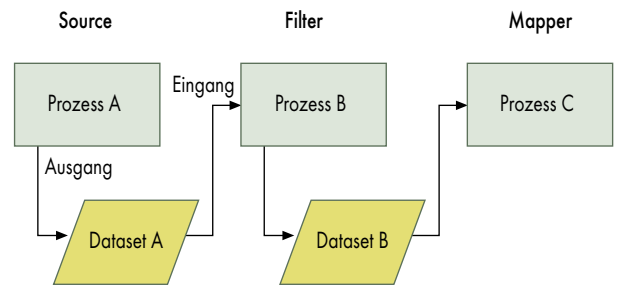
stück bildet die Klassenbibliothek, die unter anderem Schnittstellen zu den APIs der unterstützten Fenstersysteme und Grafikbibliotheken zur Verfügung stellt. Die umgebende Interpreterschnittstelle transportiert die Funktionen der

### X-TRACT

- VTK ist eine frei verfügbare C++-Klassenbibliothek für die 3D-Visualisierung mit Tcl/Tk- Python- und Java-Anbindung.
- Für die komfortable Arbeit mit dem Visualization Toolkit stellt der Hersteller Kitware mit Paraview eine ebenfalls kostenlose grafische Benutzerschnittstelle zur Verfügung.
- Diverse kommerzielle Unternehmen setzen VTK zusammen mit ihren eigenen Produkten ein, beispielsweise für die Analyse und Visualisierung großer Datenmengen.



**VTKs Grafikmodell: Die Einstellungen für Beleuchtung und Blickwinkel erfolgen durch Instanzen von Lights- beziehungsweise Camera-Objekten (Abb. 1).**



**Das Darstellungsmodell verbindet einzelne Module miteinander, die Operationen an Datasets ausführen (Abb. 2).**

Klassenbibliothek in die Sprachräume von Tcl/Tk, Python und Java.

Kitware stellt über ihren Website Distributionen für Unix, Microsofts Windows und Mac OS X zur Verfügung. Aktuell ist die Release 4.2, der Stand der Entwicklung entspricht 4.5. Für die Windows-Plattform installiert ein Wizard ein ausführbares VTK inklusive Unterstützung für Tcl/Tk, Java und Python. Unix- und Mac-OS-X-Anwender müssen den Sourcecode übersetzen, was selbst auf einem PC mit zeitgemäßen Leistungsdaten einige Stunden in Anspruch nehmen kann.

Die Konfiguration der Umgebung, das heißt die Erzeugung der Makefiles, stellt *cmake* vor der Übersetzung plattformunabhängig ein, wobei die Distribution auch Microsofts Visual C++ und Cygwin unterstützt. Erfahrungsgemäß können sich während der Übersetzung Probleme ergeben, zu deren Lösung sich die Mailing-Liste [vtkusers@vtk.org](mailto:vtkusers@vtk.org) als wertvolle Hilfe erweist. Kitware-Mitarbeiter geben dort kurzfristig Unterstützung. Als Testplattform für den vorliegenden Artikel diente ein handelsüblicher PC unter Cygwin V1.5 und W2K Professional. Bei der probeweisen Übersetzung der Quellen traten eine Reihe von Fehlern auf, die das VTK-Entwickler-Team umgehend zu beseitigen half.

Als einführende Lektüre für die Arbeit mit dem Toolkit empfiehlt Kitware neben dem eher theoretischen Werk von Schroeder et al. [1] vor allem das Benutzerhandbuch [3], das anhand von Beispielen auf rund 300 Seiten nahezu alle Aspekte des Umgangs mit VTK behandelt. Für Anwender, denen die Materie der 3D-Computergrafik bekannt ist, empfiehlt sich vor dem Kauf der Bücher ein Blick auf das Kurztutorial der Handbuchautoren [4] sowie in die umfassende Sammlung lauffähiger Programme, die in der Distribution enthalten sind und deren Komple-

xität von einfach bis anspruchsvoll reicht. Darüber hinaus steht als Referenz und Hilfe für den Umgang mit den rund 700 Bibliotheksklassen ein Hypertextdokument in HTML und als Hilfedatei für Microsofts Windows zur Verfügung.

## Visualisierung nach dem Baukastenprinzip

VTKs objektorientierter Ansatz gewährleistet die einfache Eingliederung des Toolkits in den Design- und Implementierungsprozess anderer objektorientierter Systeme bei durchgängiger Anwendung etablierter Methoden [2]. Die Interpreterschicht garantiert kurze Entwicklungszyklen und leicht verständlichen Code. Dabei trägt die C++-Klassenbibliothek den Erfordernissen zeitkritischer Anwendungen Rechnung und gewährleistet durch ihre strikte Trennung von der Interpreterschicht eine einfache Einbettung von VTK.

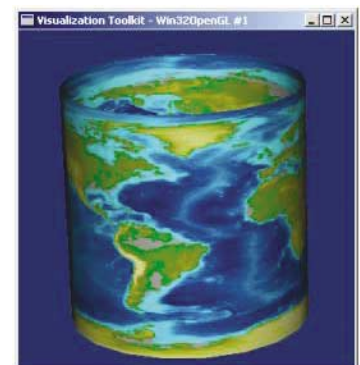
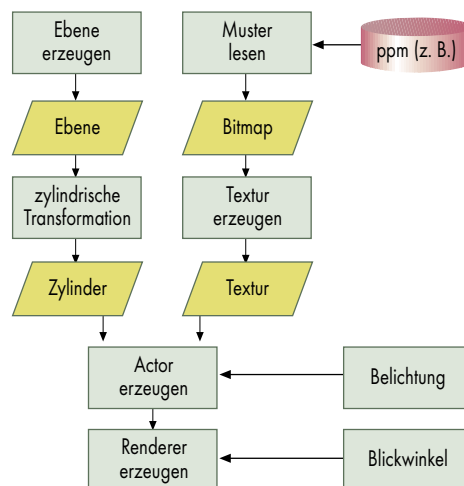
Das Objektmodell unterscheidet zwischen Grafik- (Graphics Model) und Darstellungsmodell (Visualization Model). Ersteres nimmt eine Abstrahierung der 3D-Grafik vor, indem es grundlegende Objekte definiert und

deren Eigenschaften sowie ihr Zusammenwirken beschreibt. Die Namen der definierten Objekte – *RenderWindow*, *RenderWindowInteractor*, *Renderer*, *Light*, *Camera*, *Actor*, *Property*, *Mapper*, *Transform* legen die jeweilige Bedeutung nahe.

*RenderWindow*, von dem es pro Anwendung mehrere geben kann, verwaltet ein Fenster des zugrunde liegenden Fenstersystems – beispielsweise X11 oder Microsofts Windows. *RenderWindowInteractor* ist ein Werkzeug zur Interaktion mit der durch *RenderWindow* wiedergegebenen Szene. Es wirkt auf ein einzelnes *RenderWindow*-Objekt und ermöglicht beispielsweise die Manipulation der Kameraposition oder das Ein- und Ausschalten der Stereodarstellung.

Ein oder mehrere *Renderer*-Objekte zeichnen in ein *RenderWindow*. *Renderer* koordinieren *Lights*-, *Camera*- und *Actor*-Objekte. Letztere repräsentieren die Gegenstände einer Szene und werden durch *Property*-, *Trans-*

**Die Actor-Instanz zeigt, dass Prozessobjekte mehrere Eingänge haben können, was für die Ausgänge ebenfalls gilt. Das Prozessobjekt Actor verfügt über eine prozedurale Source-Objekt-Klasse („Ebene erzeugen“) und eine externe („Muster lesen“) (Abb. 3).**



form- und Mapper-Objekte definiert, deren Komposition und Eigenschaften sie für die Wiedergabe in einem *Renderer* zusammenfassen.

## Daten durchwandern ein Netz von Modulen

Das Darstellungsmodell basiert auf einem Datenflussschema, das einzelne Module miteinander verbindet. Dabei erfolgt eine Unterscheidung in Prozess- und Datenobjekte (Datasets). Bei Ersteren handelt es sich um einzelne Module, die Operationen an Datasets ausführen, die das Netz durchlaufen, indem VTK sie vom Ausgang eines Moduls an einen Eingang des nächsten Moduls weitergibt.

Prozessobjekte unterteilt VTK in die drei Klassen *Sources*, *Filters* und *Mappers* (Abbildung 2). *Source*-Objekte bilden die Datenquellen des Darstellungsnetzes. Ein oder mehrere *Source*-Objekte erzeugen ein oder mehrere Datasets, die das Netz speisen. *Filter* nehmen die von den *Source*-Objekten erzeugten Datasets auf, um sie zu verarbeiten und anschließend an andere *Filter* weiterzuleiten. Diese übergeben sie schließlich am Ende einer Verarbeitungskette an *Mapper*, die das Netz terminieren. Abbildung 3 illustriert ein einfaches Darstellungsnetz.

VTK unterscheidet prozedurale Datenquellen und so genannte *Reader*, Objekte, die bestimmte Dateiformate lesen und in Datasets, die VTK-interne Datenrepräsentation, transformieren. *Reader* gibt es für Dateiformate mit polygonalen Daten, beispielsweise Binary Marching Cube, Wavefront (.obj), Stanford University Polygonal Data (.ply) und Stereo Lithography (.stl). Weitere existieren für Images und Volumendaten wie PC Bitmap (.bmp), JPEG, Portable Network Graphics (.png), TIFF sowie aufbereitete Daten von Computertomografen und Magnetic Resonance Images. Außerdem definiert das Toolkit eigene Dateiformate, für die es ebenfalls *Reader*-Objekte zur Verfügung stellt.

Eine Beschreibung dieser VTK-eigenen Formate enthält ein Auszug des VTK User's Guide [3]. Leider eignet sich dieser Auszug nicht als Leseprobe, da die Beschreibungen der Formate nicht sehr verständlich sind – anders als der Rest des Buches. Diesen Mangel kompensieren jedoch zahlreiche Beispieldateien, die für sämtliche Dateiformate in der Distribution enthalten sind. Für die meisten *Reader* existiert ein Pendant in Form eines *Writer*, der die wiedergegebene Szene in einer Datei speichern kann. Prozedurale Datenquellen sind Objekte, die Datasets für geometrische Körper wie

Linie, Kubus und Konus erzeugen und die es praktisch für jeden Zelltyp gibt.

Die Pfade, auf denen die Datasets das Netz durchlaufen, heißen Pipelines. Datasets sind eine Art Container zur VTK-internen Gruppierung der Daten, die vier Typen unterscheidet: polygonal, strukturierte Punkte sowie strukturierte und unstrukturierte Netze. Dabei hat sich hinsichtlich der Möglichkeiten zur Manipulation einzelner Elemente in unstrukturierten Netzen im Vergleich zu den Vorgängerversionen nichts geändert. Falls dies – beispielsweise zur Änderung der Farbe, erforderlich ist – muss der Anwender für das betreffende Element ein eigenes Grafikobjekt vorsehen, was je nach Datenquelle mehr oder weniger aufwendig sein kann. Für den medizinischen Bereich gibt es mittlerweile Abhilfe in Form des Insight Toolkit for Segmentation and Registration (ITK). Diese freie Software verarbeitet die Daten eines CT im Sinne einer differenzierbaren Darstellung vor.

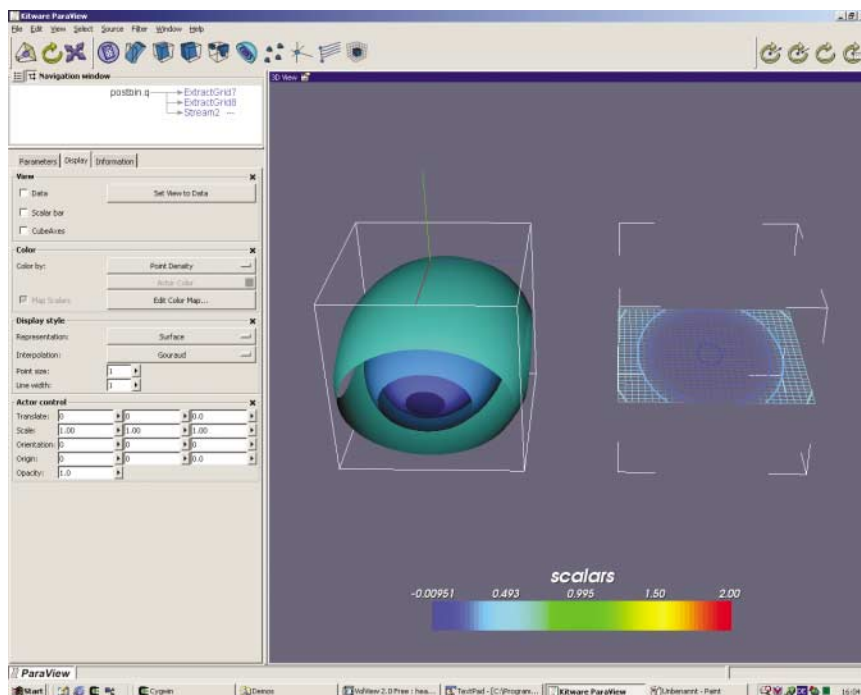
Die Zelltypen haben die VTK-Entwickler um nicht-lineare Typen erweitert, bei denen die Verbindungen zwischen den Punkten nicht durch Geraden, sondern Kurven zweiter Ordnung gegeben sind. Eine Unterstützung für NURB(S) ist nicht vorhanden; das heißt, man muss weiterhin mit externen Mitteln eine Zerlegung in Polygone vornehmen.

Ein besonderer Aspekt des Dataset ist das Zellenkonzept: Datasets bestehen aus so genannten Zellen, VTKs elementaren Darstellungsprimitiven. Zellen repräsentieren die Topologie der Punkte, aus denen ein Dataset zusammengesetzt ist. Ein einfaches Beispiel bildet der Zelltyp Dreieck, der aus drei Punkten besteht und festlegt, dass diese Punkte die Form eines Dreiecks bilden. Andere Beispiele sind Polygon und Tetraeder.

Neben diesen linearen Zelltypen gibt es fünf weitere, bei denen die Verbindungen zwischen den Punkten Kurven zweiter Ordnung sind. Diese nicht-linearen Typen – *Edge*, *Triangle*, *Quad*, *Tetra* und *Hexahedron* – entsprechen in der Abbildung prinzipiell ihren linearen Pendants.

## Mehr Komfort durch passendes GUI

Richtig komfortabel erweist sich die Arbeit mit VTK durch die Verwendung von Paraview, einer in Tcl/Tk imple-



**Paraview während des Abspielens einer Demo: Links oben sieht man unter der Toolbar das Netz der aktiven Prozessobjekte (Pipeline). In dem darunter angeordneten Steurbereich kann der Anwender die Parameter des ausgewählten Objekts verändern (Abb. 4).**

## DATEN UND PREISE

### VTK 4.2: 3D-Visualisierungs-Toolkit

**Paraview:** Tcl/Tk-GUI für VTK

**Systemvoraussetzungen:** Unix, Microsofts Windows oder Mac OS X; Tcl/Tk, Java und Python

**Hersteller/Bezugsquelle:** Kitware, Inc., [www.kitware.com](http://www.kitware.com)

**Preis:** kostenlos

mentierten frei verfügbaren grafischen Benutzerschnittstelle, die ebenfalls auf Kitwares Website zu finden ist. Kitware hat Paraview in Zusammenarbeit mit dem Advanced Computing Laboratory ([www.acl.lanl.gov](http://www.acl.lanl.gov)) des Los Alamos National Laboratory ([www.lanl.gov](http://www.lanl.gov)) zur parallelen Verarbeitung und Visualisierung großer Datenmengen entwickelt; es eignet sich jedoch auch als Stand-alone-System für die Entwicklung von VTK-Anwendungen, die man als Tcl/Tk-Skripte exportieren kann, um sie als eigenständige Anwendungen zu betreiben.

Paraview erlaubt der Anwenderin einen komfortablen Zugriff auf eine Vielzahl der VTK-Klassen, die sie mit Hilfe dieses Werkzeugs zu vollständigen Applikationen zusammensetzen kann. Ändert sie dabei Klassenparameter, ist deren Auswirkung unmittelbar zu beobachten (Abbildung 4). Durch die hierarchische Sicht auf die Pipeline kann sie navigieren und beispielsweise einzelne Module temporär deaktivieren. Paraview ist skriptfähig (Tcl) und eignet sich somit für Demonstrationen und als Ablaufumgebung für eigene Anwendungen. Da es in Tk geschrieben ist, lässt es sich leicht um eigene Funktionen erweitern.

Mit Activiz/COM und Volview bietet Kitware kommerzielle Produkte an, die es erlauben, die VTK-Funktionen als eingebettete Objekte (COM) in Microsoft-Office-Dokumenten zu nutzen (Activiz/COM) beziehungsweise eine schlüsselfertige Lösung (Volview) für die Darstellung wissenschaftlicher und

medizinischer Daten bieten, ohne dass Forscher oder Ärzte über spezielle Kenntnisse des Visualisierungssystems verfügen müssen.

Weitere Hersteller kommerzieller Produkte, die VTK als Visualisierungssoftware nutzen, sind Mapinfo Corporation, Visbox, Inc., und Visual Numerics, Inc. Mapinfo ([www.mapinfo.com](http://www.mapinfo.com)) erstellt Lösungen für Branchen, deren Aufgabestellungen an die lokalen Gegebenheiten angepasst sein müssen, beispielsweise Logistikunternehmen. VTK verwendet der Hersteller für die 3D-Darstellung von Landkarten. Visbox ([www.visbox.com](http://www.visbox.com)) stellt Virtual-Reality-Systeme her und nutzt das Toolkit in seinen Anwendungen für die dreidimensionale Visualisierung. Visual Numerics ([www.vni.com](http://www.vni.com)) hat VTK in PV-Wave integriert, einer Programmfamilie zur Analyse und Darstellung großer Datenmengen, wie sie bei Trend- und Anomalieentwicklungen anfallen.

Im nicht-kommerziellen Bereich sind es vor allem Universitäten, die zur Ergebnisvisualisierung ihrer Computersimulationen auf VTK zurückgreifen. Hier sei stellvertretend das Los Alamos National Laboratory (LANL) aufgeführt, dem VTK seine Erweiterung um Mechanismen zur Parallelverarbeitung riesiger Datenmengen verdankt [5].

## Fazit


Seit *iX* 1997 VTK das erste Mal vorstellte [6], hat es sich zu einem universell einsetzbaren 3D-Datenvisuali-

sierungssystem weiterentwickelt. Das objektorientierte Konzept und seine Umsetzung als C++-Klassenbibliothek mit einer umgebenden Interpreterschicht ermöglichen einen schnellen Einstieg sowie kurze Entwicklungszeiten. Nach wie vor ist VTK kein Performancewunder; allerdings kompensiert Hochleistungshardware, die zu Discount-Preisen zu haben ist, dieses Manko teilweise. Verbindet man solche Rechner zu Clustern, kann VTK seine Möglichkeit zur Parallelverarbeitung ausspielen. (ka)

## JÜRGEN SCHUCK

ist Mitarbeiter der Materna GmbH und als Projektleiter im Bereich IT-Service-Management tätig.

## Literatur

- [1] Will Schroeder, Ken Martin, Bill Lorenzen; The Visualization Toolkit; An Object-Oriented Approach to 3D Graphics; 3rd Edition; Kitware, Inc., 2003
- [2] Will Schroeder, Ken Martin, Bill Lorenzen; The Design and Implementation Of An Object-Oriented Toolkit For 3D Graphics And Visualization; GE Corporate Research & Development; [public.kitware.com/VTK/pdf/dioot.pdf](http://public.kitware.com/VTK/pdf/dioot.pdf)
- [3] File Formats for VTK; Auszug aus dem VTK User's Guide; Kitware, Inc.; [public.kitware.com/VTK/pdf/file-formats.pdf](http://public.kitware.com/VTK/pdf/file-formats.pdf)
- [4] Lisa S. Avilla, William Hoffmann, William J. Schroeder; Visualizing with VTK: A Tutorial; Kitware, Inc; [www.computer.org/cga/cg2000/pdf/g5020.pdf](http://www.computer.org/cga/cg2000/pdf/g5020.pdf)
- [5] A Parallel Approach for Efficiently Visualizing Extremely Large, Time-Varying Datasets; Technical Report #LAUR-00-1620; Los Alamos National Laboratory; [www-unix.mcs.anl.gov/fl/publications/ahrens00.pdf](http://www-unix.mcs.anl.gov/fl/publications/ahrens00.pdf)
- [6] Carsten Zerbst: VTK: Freie Bibliothek zur Datenvisualisierung; Von Flüssen und Filtern; *iX* 12/97, S. 154 ff. 

## -WERTUNG

- ⊕ leichte Erweiterbarkeit
- ⊕ kurze Einarbeitungszeit
- ⊕ komfortables Arbeiten durch Paraview
- ⊖ hoher Ressourcenbedarf