

**3dwm -
Plattform zur Entwicklung von 3D-GUIs**

Stefan Ziegler

15.08.2003

Inhaltsverzeichnis

1	Konzept	3
2	Aufbau	11
3	Vergleich mit Quartz	16
4	Programmierung	20
5	Quellen	27

1 Konzept

Zur Grafikausgabe hat jedes Betriebssystem ein Subsystem.
Dessen Fähigkeiten stiegen mit der Zeit.

1.1 Erste Generation

- nur Adressierung einzelner Pixel
- Farbe je Pixel zu setzen
- Terminals: Zeile, Spalte, Farbe, Buchstabe
- Beispiele: DOS, Unix-Textterminals

Pixel

- Linie zeichnen : alle Pixel einzeln
- Kreis vergrößern: alte Pixel löschen und alle neu

1.2 Zweite Generation

- nur ein Aufruf für einfache Geometrien
- Anfangs-, Endpunkte und Radien
- Rand- und Füllfarben, Füllmuster
- Bitmaps von zusammengehörigen Pixeln
- Beispiele: Windows, X Windows System, ältere Mac OS



- Linie zeichnen: nur Anfang und Ende
- Kreis vergrößern: alten löschen und für neuen nur Radius

1.3 Dritte Generation

- geometrische Objekte als Einheit gespeichert
- jede Geometrie mit: Ort, Ausmaßen, Farben/Texturen
- 2D: Rechtecke und (Bézier-)Kurven
- 3D: Würfel, Kugel und Zylinder
- Transparenz, Größe (auch in cm) und Drehwinkel änderbar

- Beispiele: 3dwm, Fresco, Quartz



- Linie zeichnen: nur Anfang und Ende
- Kreis vergrößern: nur Skalierungsfaktor angleichen

1.4 3dwm

- *3-dimensionaler Workspace-Manager*
- Darstellungssystem der dritten Generation
- könnte mit Framebuffer X Windows ersetzen
- OpenGL-Ausgabe für VR-Umgebung / Monitore
- Eingaben per Tastatur, Maus, Joystick, Headtracker-Sensor

- API-Unterstützung für 3D-GUIs, selbst ohne
- VNC: Programme zweiter Generation nutzbar
- Ortstransparenz: Ausgabe auch auf anderen Rechnern
- plattformunabhängig (soweit OpenGL, CORBA existent)
- CORBA: API-Funktionen sprachunabhängig nutzbar
→ Klienten in vielen Programmiersprachen

1.5 CORBA

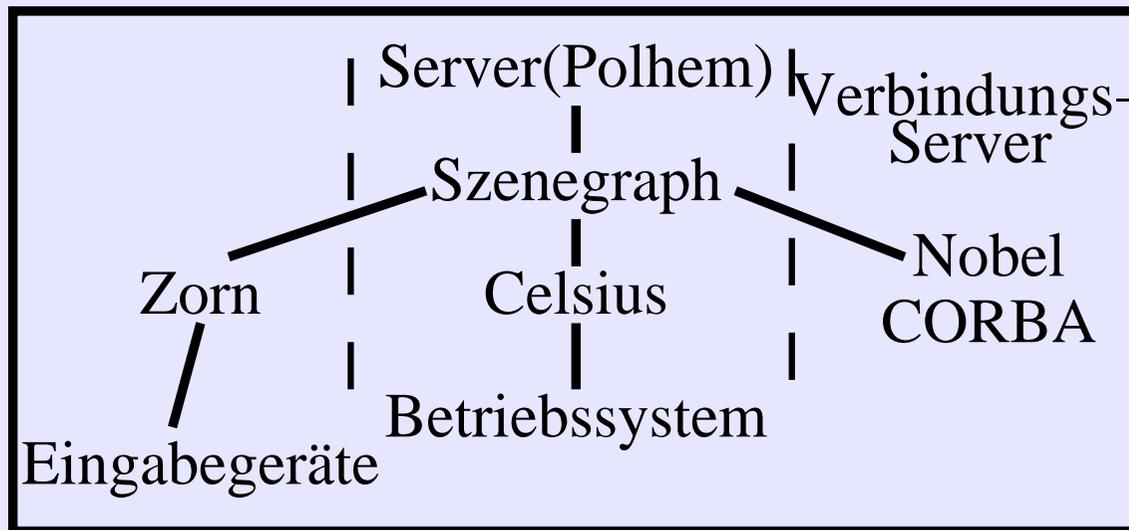
- Protokoll über TCP, Port 2809
- Aufruf von Funktionen auf entfernten Rechnern
- Datenstrukturen verteilt nutzbar
- bei 3dwm zur Zeit nur omniORB
- später auch ORBit (von GNOME benutzt) denkbar

2 Aufbau

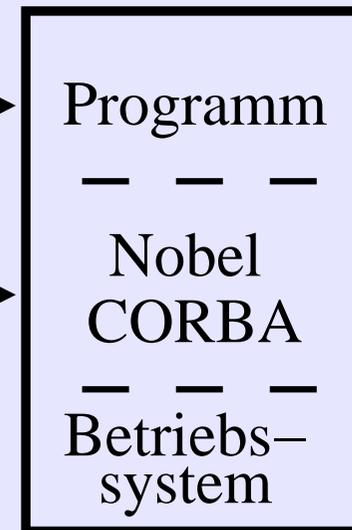
Wie bisher: Server zur Darstellung und Geräteverwaltung und Klienten-Anwendungen, die die Geometrien vorgeben.

2.1 Server und Klient

Server



Klient



2.2 Szenegraph

- gerichtet und azyklisch → Baum
- alle 2D/3D-Objekte in Hierarchie
- kleinste Teile: Geometrien (face) und Farben/Texturen (appearance)
- zusammengeführt und Transformation darüber (Größe, Ort)
- insgesamt ein Knoten im Baum
- andere Knoten: Controller für Verhalten nach Events
- oberster Knoten Wurzel, Söhne alle Anwendungen daran

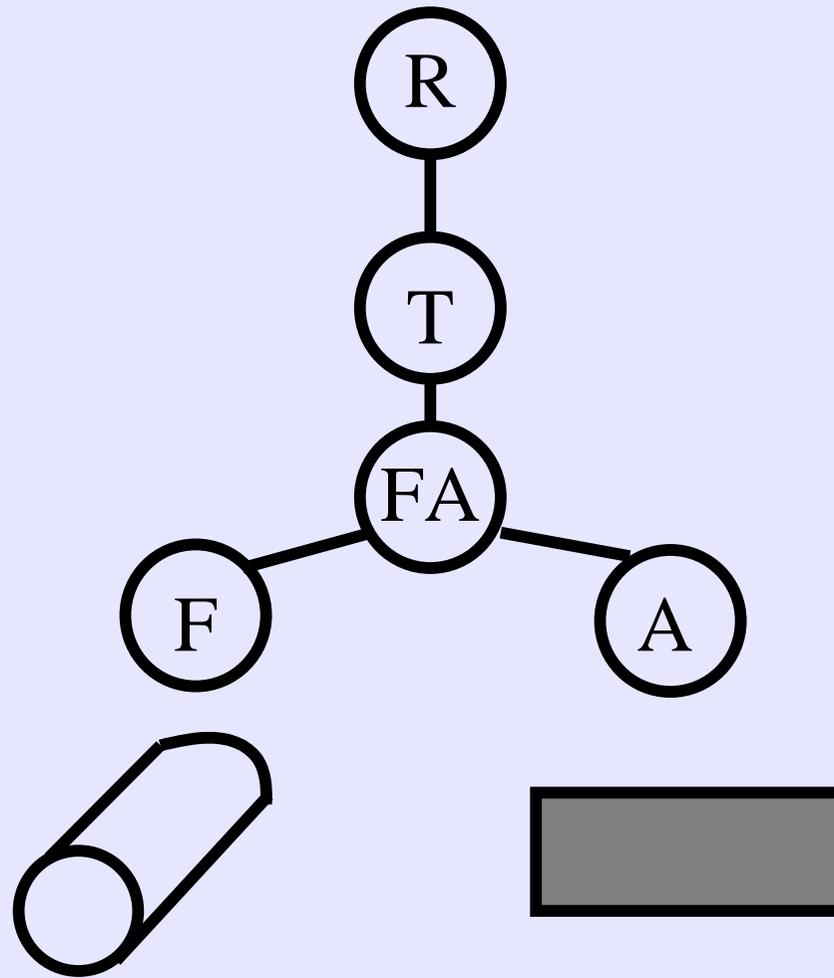


Abbildung 1: Szenegraph

2.3 Module

- **Celsius:** Abstraktion vom Betriebssystem (Mutex, Threads, u.a.)
- **Garbo:** VNC-Schnittstelle zu bisherigen Window-Systemen, als Textur auf beliebiger Geometrie
- **Nobel:** API für Klienten als CORBA-Beschreibung,
- **Polhem:** Display-Server, Hardware, Plugin-Erweiterungen
- **Solid:** Umsetzung Szenegraph (Vertexe, Normalen, Texturen)
- **support:** Bilder laden (PNG), Hilfsklassen für Klienten
- **Zorn:** Parsen der Einstellungs-Dateien für Eingabegeräte

3 Vergleich mit Quartz

Quartz und 3dwm haben gemeinsame Grundlagen
aber auch viele Unterschiede.

3.1 Quartz

- Grafiksystem der 3. Generation
- seit MacOS X im Apple-Betriebssystem
- Darstellungsebene der Oberfläche Aqua
- Implementierung in XFree ab 4.0.2



Abbildung 2: Systemaufbau von MacOS (vereinfacht)

3.2 Gemeinsamkeiten

- unabhängig von der Auflösung
- OpenGL genutzt
- Objekte drehen / skalieren / verschieben
- Farbe u. Transparenz dynamisch änderbar

3.3 Unterschiede

- PDF als Grafikspeicher statt Szenegraph
- durch PDF: einfaches Drucken
- Antialiasing
- transparente Schatten
- alte MacOS-Anwendungen und neue auf gleicher Oberfläche
- verschiedene Farbräume (RGB, CMYK, Graustufen ...)

- ohne CORBA
- nur 2D-Geometrien

4 Programmierung

Installation und Beispielprogramm gezeigt.

Genaueres in Doku nachzulesen.

4.1 Voraussetzungen für 3dwm

- OpenGL 1.1 (z.B. Mesa)
- omniORB 3.0 (<http://www.uk.research.att.com/omniORB/>)
- MeshIO 0.2 (3D-Files) von 3dwm
- g++ ≥ 2.95 (besser g++ < 3)
- Automake 1.4, Autoconf 2.13, Libtool 1.3.5, bintools

Weitere Voraussetzungen:

- VNC-Server empfohlen (z.B. Xvnc)
- libpng
- SDL 1.2 (Multimedia-Lib)
- Expat (XML-Dateien)

4.2 Installation von omniORB

- omniORB 3.0 ohne configure → komplizierter
- omniORB 4.0: configure, make, make install
- Logverzeichnis erstellen, in Shell-Variable
- Konfiguration anhand Beispieldatei
- Lage der Konfig-Datei in Shell-Variable

4.3 Installation von 3dwm

- bei richtigen Libs: nur configure, make, make install
- Konfigurations-Datei ins Homeverzeichnis / Ort in Shell-Variable
- Lage der Konfig-Datei von omniORB in Shell-Variable
- libmeshio: configure, make, make install
- für omniORB 4.0:
 - in configure: `ORBLIBS = -lomniORB4 -lomnithread -lomniDynamic4`
 - in `3dwm-0.3.2/include/Nobel/CORBA.hh`:

```
#include <omniORB4/CORBA.h>
```
- weitere Änderungen bei `g++ >= 3`, neuerem OpenGL, neuerem Expat

4.4 Beispielprogramm

- **main:** Optionen, Erstellen der Klasse, Starten
- **Konstruktor:** Mit Server und dessen Module verbinden
- **buildHello:** Geometrie, Szenegraph
- **run:** Endlosschleife mit Wartezeit, ruft redraw
- **redraw:** neue Farben setzen

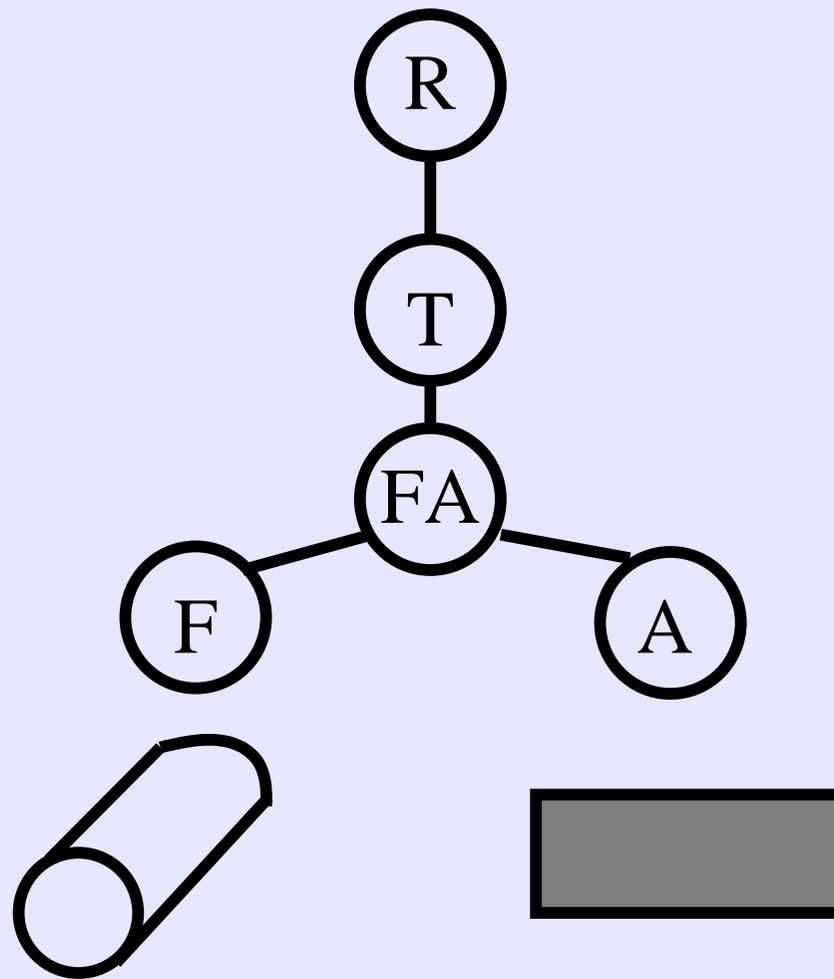


Abbildung 3: Szenegraph

5 Quellen

Von *www.3dwm.org*:

- Dokumentation 3dwm unter: <http://www.3dwm.org/docs/>
- http://www.3dwm.org/html/about_architecture.php
- http://www.3dwm.org/html/about_introduction.php
- tar-Pakete und Installations-Beschreibungen:
http://www.3dwm.org/html/development_download.php

Von *www.apple.com*:

Beschreibungen von Quartz unter:

- <http://developer.apple.com/quartz/overview.html>
- <http://www.apple.com/macosx/jaguar/quartzextreme.html>
- http://www.apple.com/macosx/pdfs/Quartz_TB.pdf
- <http://www.apple.com/macosx/technologies/graphics.html>

Von *www.arstechnica.com*:

- "Mac OS X Update: Quartz & Aqua" von John Siracusa,
Ausgabe 1/2000

<http://www.arstechnica.com/reviews/1q00/macos-x-gui/macos-x-gui-1.html>