Mitglied der Helmholtz-Gemeinschaft

JÜLICH FORSCHUNGSZENTRUM

Integration von 3D-Visualisierungstechniken in 2D-Grafiksystemen

24. August 2012 | Florian Rhiem

Inhaltsverzeichnis

Motivation

- Die Realität ist räumlich
- Der Monitor ist 2-dimensional
- Wie können 3D-Inhalte in 2D-Grafiken integriert werden?
- Wie können die . . .
  - Ansätze
  - Schnittstellen
  - Ausgabemedien

  in einer Anwendung vereint werden?

Grundlagen

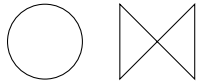
2D-Grafiksysteme – GKS

Das ***G**raphische **K**ern**s**ystem*

- ISO-Standard
- Hohe Qualität
- Grafische Primitiven
- Plattform-unabhängig

- Linienzüge

- Symbole

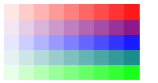


- Flächen

- Text

Beispieltext

- Farbmatrizen



Grundlagen
2D-Grafiksysteme – GR

Eine auf dem GKS aufbauende **Gr**afikbibliothek

- Erzeugt komplexe Objekte mit Primitiven des GKS
  - Koordinatensysteme
  - Spline-Funktionen
  - Fehlerbalken
  - Rastergrafiken
  - LaTeX-Formeln
  - ...
- Vereinfacht die Benutzung
- Die Vorteile des GKS bleiben erhalten

Grundlagen
3D-Visualisierungstechniken – OpenGL

Die ***Open Graphics Library***

- 3D-Grafik API
- Spezifikation
- Plattformunabhängig
- Benutzbar aus C, C++, Python, Fortran, Delphi, Java, C#,...
  - Weitgehend sprachunabhängige Schnittstelle
- Weit verbreitet
- Aktuell: OpenGL 4.3

Grundlagen
3D-Visualisierungstechniken – WebGL

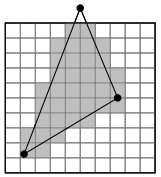
Web Graphics Library

- OpenGL im Web
- 3D-Grafik in HTML5-Dokumenten
- JavaScript-Integration

Grundlagen
3D-Visualisierungstechniken – Rasterisierer

Umwandlung von Vektorgrafik zu Rastergrafik

1. Durch Eckpunkte beschriebene Vektorgrafik
2. Pixel unter der Fläche feststellen
3. Pixelfarbe bestimmen

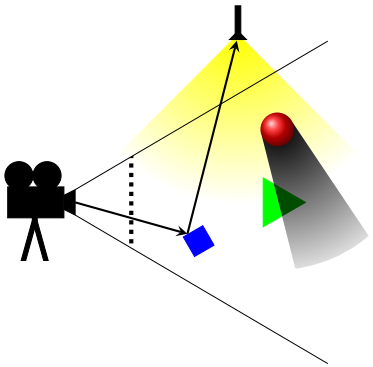


Grundlagen
3D-Visualisierungstechniken – Raytracing

„Strahlenverfolgung“, Pixel für Pixel

1. Szene mit Objekten und Lichtquelle im Raum
2. Strahl von der Kamera durch einen Pixel bestimmen
3. Ersten Schnittpunkt mit der Szene finden
4. Weitere Strahlen verfolgen und dabei
   - Reflektion
   - Lichtbrechung
   - Absorption

   beachten.



Grundlagen
3D-Visualisierungstechniken – POV-Ray

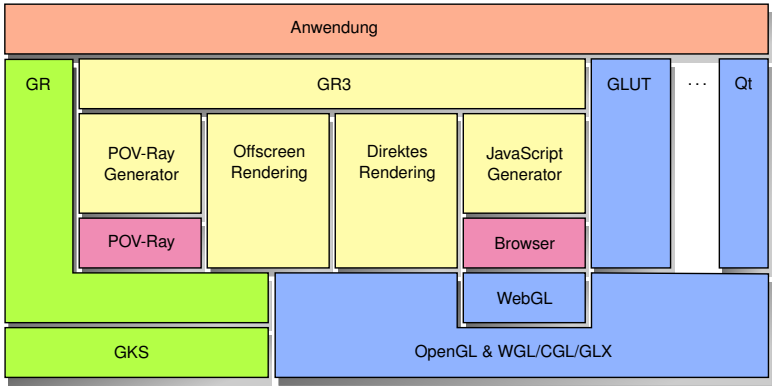
***P**ersistence **o**f **V**ision **Ray**tracer*

- Raytracer
- Sehr hohe Qualität
- Zeitaufwändiges Rendering
- Szenenbeschreibungssprache

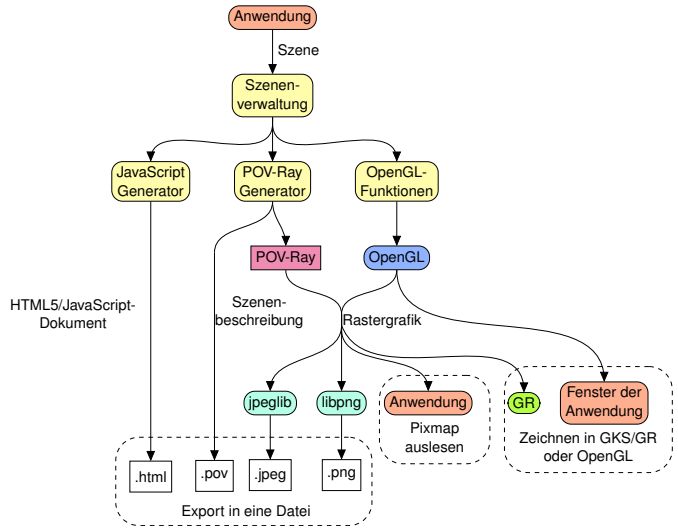
Grundlagen
3D-Visualisierungstechniken

- Gleiche Inhalte
- Verschiedene Ausgabemedien
  - Rastergrafiken (JPEG, PNG oder „rohe“ Pixmap)
  - HTML5-Dokument
  - Anzeigefenster
    - GKS
    - GLUT
    - Qt
    - wxWidgets
    - ...

Realisierung
GR3 in der Grafikinfrastruktur



Realisierung



Realisierung

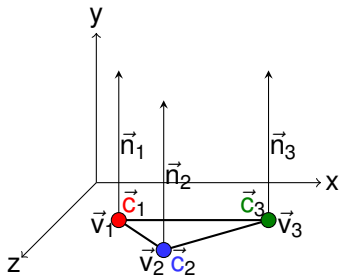
POV-Ray Generator

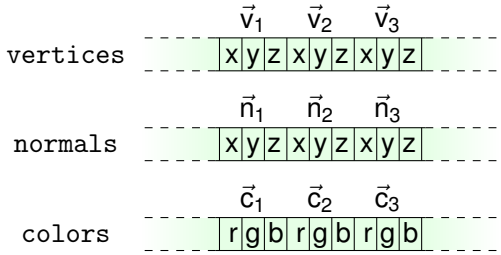
- Szenenbeschreibung
- Eigenschaften der Szene
  - `camera`
  - `light_source`
  - `background`
- Körper in der Szene
  - `mesh`
  - `sphere`
  - `cylinder`
  - `cone`

Realisierung

OpenGL

- Kein Konzept einer Szene
- Dreiecksgitter
- Grafik-Pipeline
  - Transformationen
  - Rasterisierung
  - Farbberechnungen

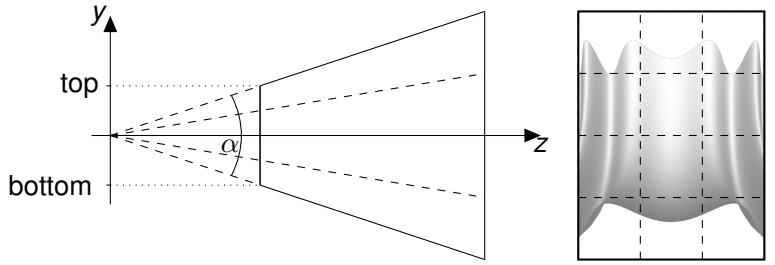




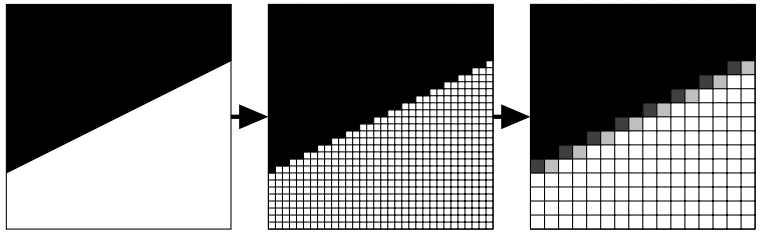
Realisierung

OpenGL

- Auf Monitore ausgelegt
- Maximale Auflösung begrenzt
- Zerlegung in Einzelbilder
- Zusammensetzen im Hauptspeicher



Realisierung
OpenGL



Vektorgrafik | Rastergrafik in 4-facher Auflösung | Rastergrafik mit Kantenglättung

Realisierung
OpenGL

- OpenGL Kontext
- Bestehender OpenGL Zeichenbereich wird verwendet
- `glX/CGL/wglGetCurrentContext()`
- Alternativ: Offscreen Rendering Kontext
- Framebuffer Objekt

Realisierung
WebGL

- WebGL Kontext
- `<canvas>`
- `canvas.getContext("experimental-webgl")`

C Code (OpenGL)

```
GLuint vbo;
glGenBuffers(1, &vbo);
glBindBuffer(GL_ARRAY_BUFFER, vbo);
```

JavaScript Code (WebGL)

```
var vbo = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, vbo);
```


Realisierung
GKS/GR

- Rastergrafiken als Farbmatrix (*Cell Array*)
- `gr_drawimage`
- `gr3_drawimage`

Ergebnisse
Moldyn

- Molekülvisualisierung
- Darstellung mit:
  - ~~X11~~
  - ~~GDI~~
  - OpenGL

Ergebnisse
Moldyn

Bisher:

- Interaktive Darstellung mit OpenGL
- Grafikexport mit GLor und jpeglib oder mit POV-Ray

$\Rightarrow$ Redundanter Darstellungscode

Jetzt:

- Szenenbeschreibung bei Änderung des Inhalts
- Zeichnen und Export mit GR3-Funktionen

$\Rightarrow$ Erweiterungen einfacher

Zusammenfassung und Ausblick

- Eine Schnittstelle für mehrere Ausgabeformen
- Python-Anbindung
- 2D-Elemente fehlen, z. B. Text
- Isoflächen

Vielen Dank für Ihre Aufmerksamkeit! Fragen?