Computer-Graphik
Einführung in OpenGL

G. Zachmann
Clausthal University, Germany
zach@in.tu-clausthal.de

Hintergrund und Geschichte

- ... auf der Suche nach einer einheitlichen Software-Schnittstelle (API: Application Programming Interface) zur Programmierung von Graphiksystemen
- Standardisierungsbemühungen
  - GKS, PHIGS, ...
- „Proprietäre Systeme“
  - HP: Starbase, SGI: GL (Graphics Library)
- Gewinner: SGI mit GL in Verbindung mit sehr guter Hardware
- OpenGL (1992, Mark Segal & Kurt Akeley)
- Konkurrenz nur noch durch Microsoft (Direct3D)

OpenGL

- OpenGL ist ein Software-Interface für Graphik-Hardware mit ca. 250 verschiedenen Kommandos
  - Hardware-unabhängig
- Warum „Open“?
  - offen für Lizenznehmer
  - verwaltet vom Architecture Review Board (ARB)
    - NVIDIA, ATI, IBM, Intel, SGI, ...
  - von jedem Lizenznehmer erweiterbar (Extension)
- Nicht dabei:
  - Handhabung von Fenstern/Windows
  - Benutzereingabe

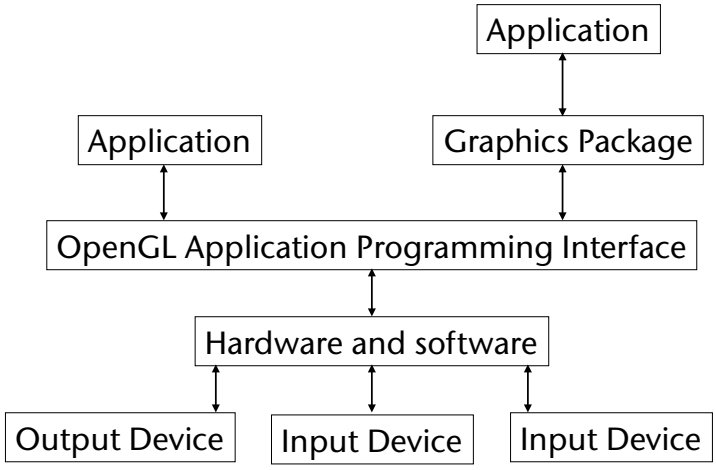
Warum OpenGL

- Standard für Rendering von 3D Graphiken
  - Implementierung als C/C++ Bibliothek von diversen Herstellern: nVIDIA, ATI, SiliconGraphics, Microsoft, [*Mesa*]
  - Enthalten in jedem Windows-, MacOS-, Linux- und Unix-System
- Plattform-unabhängig
- Hardware-unterstützt
- Schnell & einfach
- Unabhängig vom Window-Manager
- Voraussichtlich der Standard im Handheld-Markt
- Viele weitere offene Standards der Khronos-Group

Einführung

- OpenGL Core: Basisprimitive (Punkte, Linien, Polygone...)
- Darauf aufbauend gibt es diverse Tools:
  - OpenGL Utility Library (GLU): standardmäßig dabei für Oberflächen (Quadrics, NURBS...)
- OpenGL ist eine „State-Machine“
  - Man versetzt die „Maschine“ in einen Zustand, der so lange besteht, bis er wieder verändert wird
  - Beispiel: ab jetzt alles rot, ab jetzt dieses Material, ab jetzt diese Transformation
  - Effizienter, als Daten jedes Mal neu zu übergeben

Programm-Übersicht

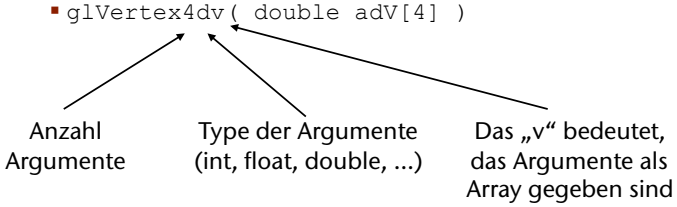


OpenGL Grundstruktur

- Low-Level-API
  - Hardware-nah aber Hardware-unabhängig
- 2 Arten von Funktionen
  - Zustand ändern
  - Primitive darstellen
- Ein reines *immediate mode* System (zumindest früher):
  - Sehr einfache Befehle
  - Direktes Durchreichen an die HW
  - Dreiecks-basiert, keine interne Repräsentation der Szene
- Klarer Namensraum
  - Befehle fangen mit gl... an
  - Konstanten mit GL_...

Einführung in OpenGL

- Die Bedeutung der Suffixe:
  - `glVertex2f( float fV1, float fV2 )`
  - `glVertex3i( int iV1, int iV2, int iV3 )`
  - `glVertex4dv( double adV[4] )`



Geometrische Primitive in OpenGL

- Alle geometrischen Primitive werden durch ihre Eckpunkte beschrieben

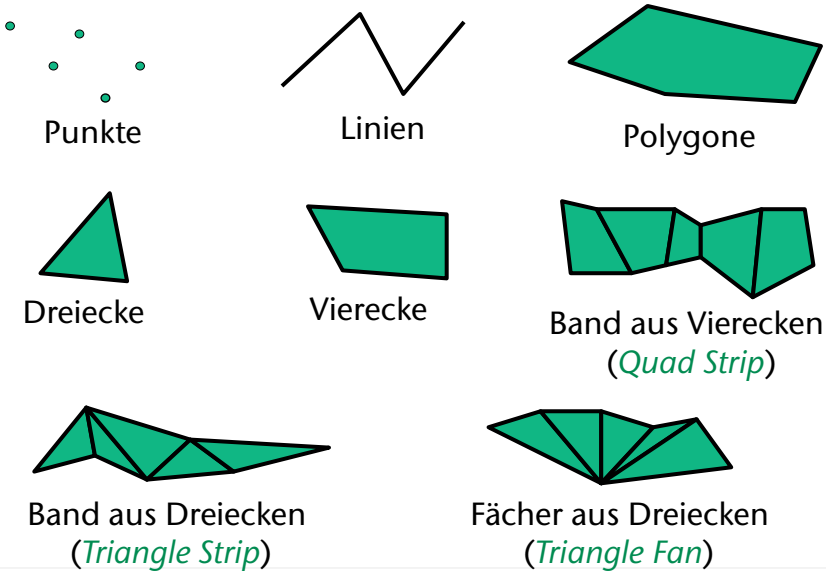
```
glVertex2s( 2, 3 );
glVertex3d( 0.0, 0.0, 3.1415926535898 );
glVertex4f( 2.3, 1.0, -2.2, 2.0 );

Gldouble ad_vect[3] = {5.0, 9.0, 1992.0};
glVertex3dv( ad_vect );
```


Prinzipielle Vorgehensweise des Zeichnens

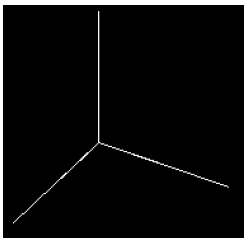
- Einfügen der Eckpunkte zwischen **`glBegin()`** ... **`glEnd()`**
  - Kann beliebigen C Code enthalten
  - Beinhaltet Befehle wie **`glVertex3f`**, **`glColor3f`** (= Attribute der Vertices)
  - Keine sonstigen OpenGL-Befehle
  - Attribute müssen gesetzt sein, bevor die Koordinaten eines Vertex abgeschickt werden!
- Client-Server Modell:
  - Client (= App.) erzeugt Eckpunkte, Server (= OpenGL + Hardware) zeichnet; auch wenn beides auf dem selben Rechner läuft
  - Dazwischen ein Buffer
  - **`glFlush()`** & **`glFinish()`** melden das Ende eines Frames (=Bildes)

Primitive in OpenGL



Beispiel

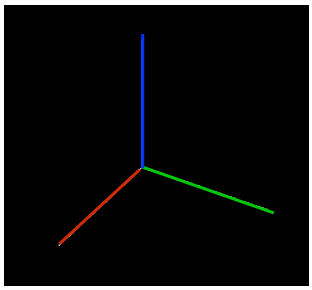
```
glBegin( GL_LINES );
   glVertex3f( 0.0, 0.0, 0.0 );
   glVertex3f( 1.0, 0.0, 0.0 );
   glVertex3f( 0.0, 0.0, 0.0 );
   glVertex3f( 0.0, 1.0, 0.0 );
   glVertex3f( 0.0, 0.0, 0.0 );
   glVertex3f( 0.0, 0.0, 1.0 );
glEnd( );
```



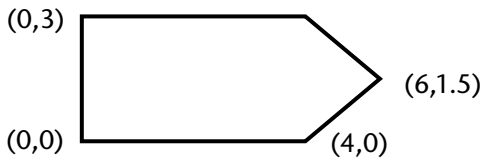
```
glBegin( GL_LINES );
   glColor3f (1.0, 0.0, 0.0);
   glVertex3f( 0.0, 0.0, 0.0 );
   glVertex3f( 1.0, 0.0, 0.0 );
   glVertex3f( 0.0, 0.0, 0.0 );

   glColor3f (0.0, 1.0, 0.0);
   glVertex3f( 0.0, 1.0, 0.0 );
   glVertex3f( 0.0, 0.0, 0.0 );

   glColor3f (0.0, 0.0, 1.0);
   glVertex3f( 0.0, 0.0, 1.0 );
glEnd( );
```




```
glBegin(GL_POLYGON);
    glVertex2f( 4.0, 0.0 );
    glVertex2f( 6.0, 1.5 );
    glVertex2f( 4.0, 3.0 );
    glVertex2f( 0.0, 3.0 );
    glVertex2f( 0.0, 0.0 );
glEnd() ;
```



- Abschließen der Graphikdarstellung:
  - `void glFlush(void);`

    Diese Funktion sorgt dafür, daß alle OpenGL-Befehle aus dem Command Buffer an die Hardware geschickt werden (ist für die Signalisierung des Endes eines Frames gedacht)

  - `void glFinish(void);`

    Wie glFlush(), wartet aber, bis alle Aufrufe im Framebuffer angekommen sind
- Bei der Verwendung von Qt braucht / sollte man diese Funktionen i.A. nicht selbst aufrufen!

Immediate oder Retained Mode

- Immediate Mode ("direkter Modus"):
  - Primitive werden sofort, wenn sie festgelegt sind, an das Display geschickt (Standard)
  - Graphiksystem hält keine Primitive im Speicher
- Retained Mode ("Zurückhaltender Modus"):
  - Primitive kommen in eine sog. Display-Liste
  - Display-Liste kann auf dem Server (Graphikkarte) gehalten werden
  - Kann noch mal gezeichnet werden mit unterschiedlichen Eigenschaften
  - Performanz wird so erhöht