Workshop: Einführung in die 3D-Computergrafik

Julia Tolksdorf
Thies Pfeiffer
Christian Fröhlich
Nikita Mattar

Organisatorisches

- **Tagesablauf**:
  - Vormittags: Theoretische Grundlagen
  - Nachmittags: Bearbeitung der Übungsaufgaben
  - 15.30 Uhr: Abschließendes Treffen

Organisatorisches

- **Übungsaufgaben**:
  - Pflicht- und Bonusaufgaben
  - Bearbeitung erfolgt alleine
  - Pflichtaufgaben müssen (bis zum ersten Seminartermin) bearbeitet werden
  - Rückmeldung an euren Betreuer vor dem ersten Treffen!
  - Bonusaufgaben dienen der Vertiefung der erlangten Kenntnisse

Inhalt

- Intro
- Mathematische Grundlagen
- Rendering Pipeline
- Transformationen
- Lighting / Shading
- OpenGL & GLUT
- Szenengraphen

Was ist 3D-Computergrafik?

- Repräsentation und Modellierung von 3D-Objekten
- Erstellen von 3D-Szenen mit Beleuchtung
- Rendern von 3D-Szenen
- 3D Computergrafik vs. Virtual Reality

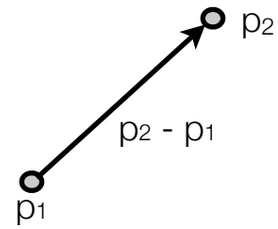
3D CG	VR
einfache visuelle Darstellung	Multimodale Präsentation (visuell, akustisch, haptisch
keine Zeitanforderungen	Echtzeit!!
statische Szenen oder definierte Animationen	Echtzeit Interaktion und Simulation
2D Interaktion (Maus, Tastatur)	3D Interaktion mit speziellen Eingabegeräten

Mathematische Grundlagen

- Skalar:
  - reelle Zahl
- Vektor:
  - Richtung und Länge
- Punkte:
  - Position im Raum
- Matrix:
  - m x n Array von Zahlen

Mathematische Grundlagen

- Vektoren:
  - haben Länge und Richtung aber keine Position!
  - Vektoren als Differenz von 2 Punkten: $x = p_2 - p_1$



- Länge: $\|\mathbf{x}\| = \sqrt{\mathbf{x}^T\mathbf{x}} = \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2}$
- 3D Vektor: (x_x, x_y, x_z)

Mathematische Grundlagen

- Vektor Operationen:
  - skalar · vektor = vektor

$$\alpha \cdot v = (\alpha \cdot v_x,\ \alpha \cdot v_y,\ \alpha \cdot v_z)$$

  - vektor · vektor = skalar

$$\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x} \cdot \mathbf{y} = \mathbf{x}^T \mathbf{y} = \sum_{i=1}^{n} x_i\, y_i$$

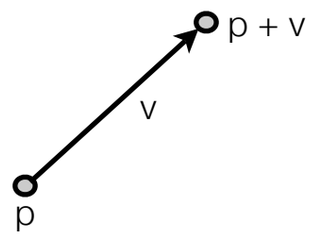
    - auch Skalarprodukt, inneres Produkt, Punktprodukt

- vektor x vektor = vektor
  - auch Kreuzprodukt

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_2 b_3 - a_3 b_2 \\ a_3 b_1 - a_1 b_3 \\ a_1 b_2 - a_2 b_1 \end{bmatrix}$$

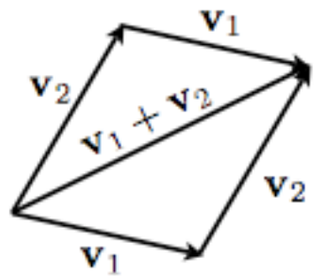
Mathematische Grundlagen

- Punkte
  - Positionen im n-dimensionalen Raum
  - 3D-Punkt: $p = (p_x, p_y, p_z)$
  - Punkt = Punkt + Vektor

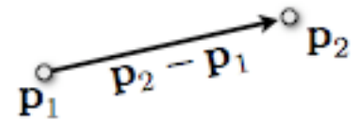


Mathematische Grundlagen

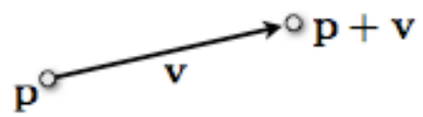
- mögliche Operationen:
  - vektor + vektor =



  - punkt - punkt =



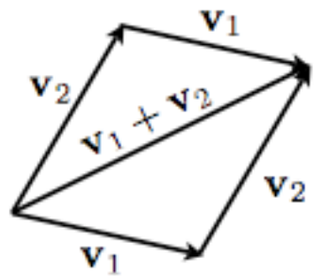
  - punkt + vektor =



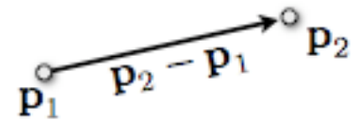
  - punkt + punkt =

Mathematische Grundlagen

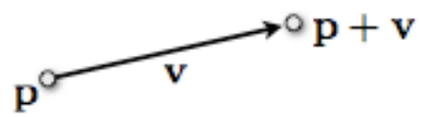
- mögliche Operationen:
  - vektor + vektor = vektor



  - punkt - punkt = vektor



  - punkt + vektor = punkt



  - punkt + punkt = ???

Mathematische Grundlagen

- Matrizen

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \end{bmatrix}$$

  - Matrizenaddition:

$$\begin{pmatrix} 1 & 3 & 2 \\ 1 & 2 & 2 \end{pmatrix} + \begin{pmatrix} 0 & 0 & 5 \\ 2 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1+0 & 3+0 & 2+5 \\ 1+2 & 2+1 & 2+1 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 7 \\ 3 & 3 & 3 \end{pmatrix}$$

  - Skalarmultiplikation:

$$2 \cdot \begin{pmatrix} 1 & 3 & 2 \\ 1 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 2 \cdot 1 & 2 \cdot 3 & 2 \cdot 2 \\ 2 \cdot 1 & 2 \cdot 2 & 2 \cdot 2 \end{pmatrix} = \begin{pmatrix} 2 & 6 & 4 \\ 2 & 4 & 4 \end{pmatrix}$$

  - Matrizenmultiplikation:

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \cdot \begin{pmatrix} 6 & -1 \\ 3 & 2 \\ 0 & -3 \end{pmatrix} = \begin{pmatrix} 1 \cdot 6 + 2 \cdot 3 + 3 \cdot 0 & 1 \cdot (-1) + 2 \cdot 2 + 3 \cdot (-3) \\ 4 \cdot 6 + 5 \cdot 3 + 6 \cdot 0 & 4 \cdot (-1) + 5 \cdot 2 + 6 \cdot (-3) \end{pmatrix} = \begin{pmatrix} 12 & -6 \\ 39 & -12 \end{pmatrix}$$

Mathematische Grundlagen

- Matrizen
  - Identitäts- oder Einheitsmatrixmatrix:

$$E = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

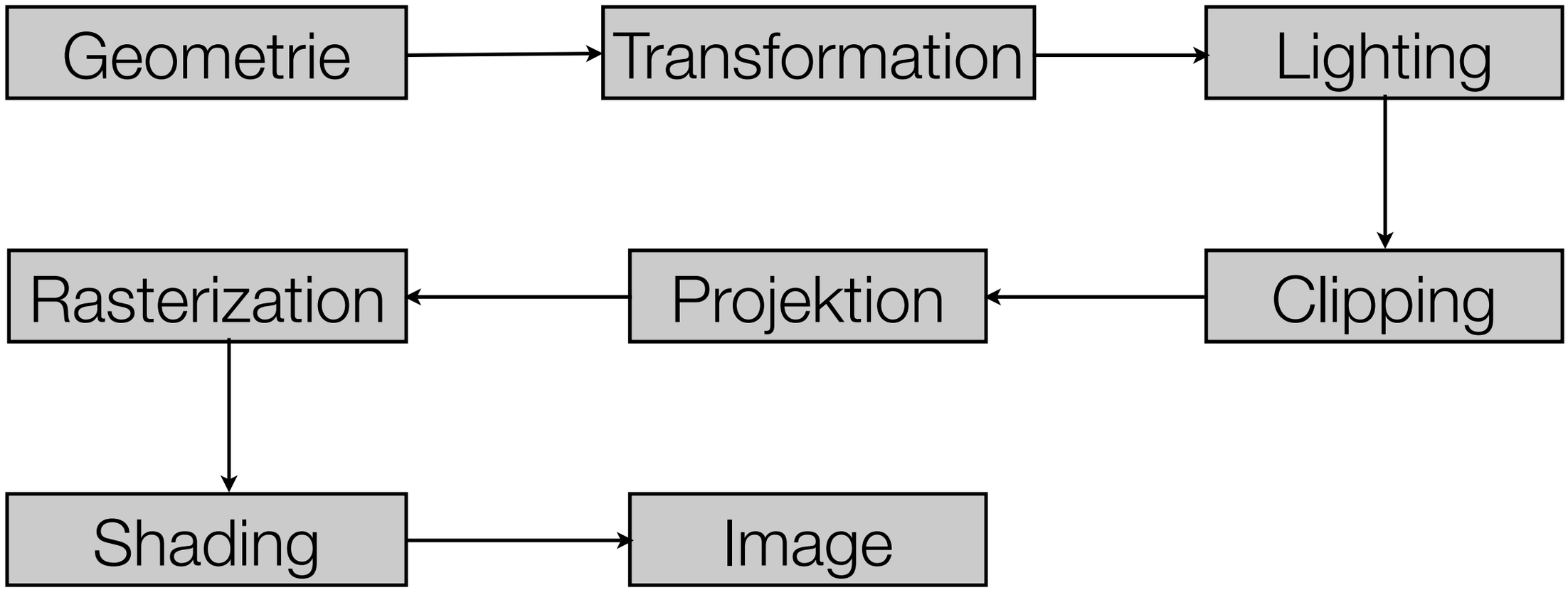
  - transponierte Matrix:

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix} \qquad A^T = \begin{pmatrix} a_{11} & \cdots & a_{m1} \\ \vdots & \ddots & \vdots \\ a_{1n} & \cdots & a_{mn} \end{pmatrix}.$$

  - inverse Matrix:

$$A \cdot A^{-1} = A^{-1} \cdot A = E$$

Rendering Pipline



Transformationen

- Was ist eine Transformation?
  - Rotation
  - Skalierung
  - Schärung
  - Translation
- Warum sind Tranformationen wichtig?
  - Ein Objekt in der Welt bewegen
  - Beziehungen von Objekten spezifizieren
  - ...

Transformationen

- Problem:
  - Wie können wir Punkte und Vektoren unterscheiden?
  - Wie lassen sich alle Transformationen als Matrixoperationen darstellen?

- Lösung: Homogene Koordinaten:
  - 3D Transformationen sind als 4x4 Matrizen darstellbar
    - Kartesische Koordinaten werden in Homogene Koordinaten umgewandelt:

$$(x, y, z)^T \rightarrow (x, y, z, 1)^T$$

Homogene Koordinaten

- 3D-Darstellungen ab jetzt: (x, y, z, w)
  - Punkt: (x,y,z,1)
  - Vektor: (x,y,z,0)

3D Transformationen

- Transformationen lassen sich als 4x4 Matrizen darstellen:
  - Objekt im Ursprung ohne Transformation wird durch Einheitsmatrix repräsentiert

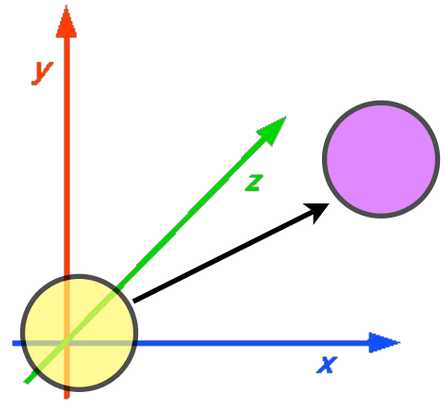
$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - alle weiteren Transformationen werden mit der Ausgangsmatrix multipliziert

3D Transformationen

- Translation
  - Verschiebung eines Objekts in eine Bestimmte Richtung

$$T(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



  - Objekt ist um t_x, t_y, t_z im Raum verschoben

3D Transformationen

- Rotation (um eine Achse)
  - X-Achse:

$$R_x(\delta) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\delta) & -\sin(\delta) & 0 \\ 0 & \sin(\delta) & \cos(\delta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

  - Y-Achse:

$$R_y(\delta) = \begin{pmatrix} \cos(\delta) & 0 & \sin(\delta) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\delta) & 0 & \cos(\delta) & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

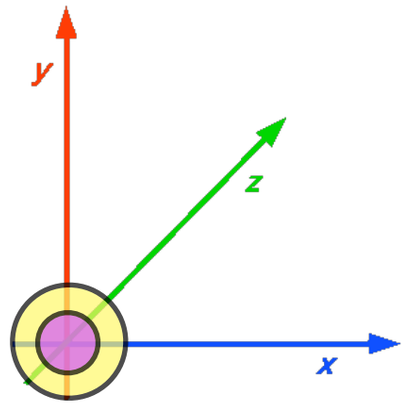
  - Z-Achse:

$$R_z(\delta) = \begin{pmatrix} \cos(\delta) & -\sin(\delta) & 0 & 0 \\ \sin(\delta) & \cos(\delta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

3D Transformationen

- Skalierung
  - Änderung der Größe eines Objekts

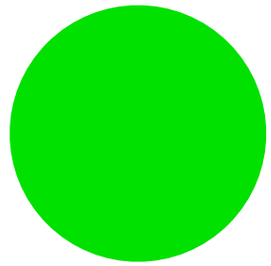
$$S(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$



  - Objekt ist um s_x, s_y, s_z verkleinert/vergrößert

Lighting

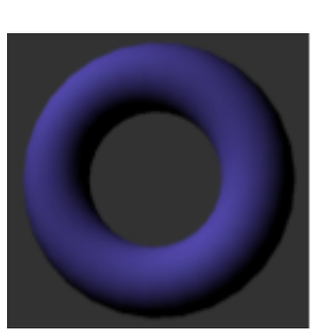
- Tiefenhinweise können nur mit Licht gewonnen werden.
- Beispiel: Grüne Kugel
  - besteht aus vielen Dreiecken
  - jedes Dreieck wird eingefärbt
  - Ergebnis: flacher Kreis



- in der Realität hätte jeder Punkt auf der Kugel einen anderen Grünton
  - Lichtquelle
  - Materialeigenschaften
  - Standort des Betrachters

Lighting

- Ambient:
  - globale Effekte (Sonne,...)
  - gleiche Intensität überall
- Diffuse:
  - Beleuchtung durch eine Lichtquelle
  - Richtung
  - Unabhängig vom Betrachter
- Spekular:
  - direkte Lichteinstrahlung
  - Reflektion

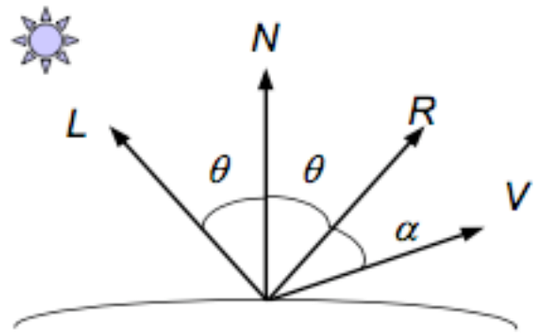


Lighting

- Berechnung mit dem Phong Beleuchtungsmodell:

$$I_\lambda =$$

  - I_a = Intesität des ambienten Lichts
  - $k_{a,d,s}$ = Empirisch bestimmter Reflexionsfaktor (Materialkonstante)
  - O = Farbe des Objekts (rgb) (d = diffuse, s = spekular)
  - I_p = Intesität der einfallenden Lichtquelle
  - L = Vektor in die Richtung der Lichtquelle (Einfallsvektor)
  - N = Normale des Objekts
  - R = Ausfallsvektor
  - V = Vektor zu Kamera



Shading

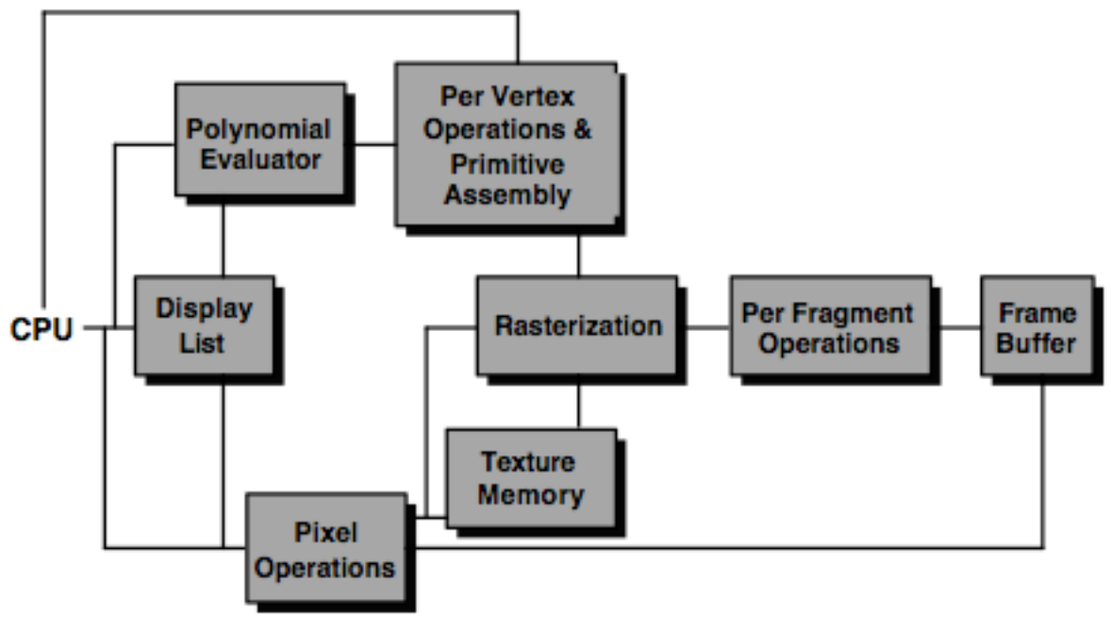
- Schattierungsverfahren (wie wird ein Objekt eingefärbt?)
  - Flat Shading
    - pro face wird eine Farbe berechnet
    - harte Übergänge

  - Gouraud Shading (Smooth Shading)
    - pro Vertex wird eine Farbe berechnet
    - Interpolation zwischen den Farben

  - Phong Shading
    - Berechnung pro Pixel
    - Lineare Interpolation der Normalen

OpenGL

- OpenGL = Open Graphics Library
- Graphics rendering API
- Vorteil:
  - Unabhängig bezüglich Betriebssystem, Hardware, Window system

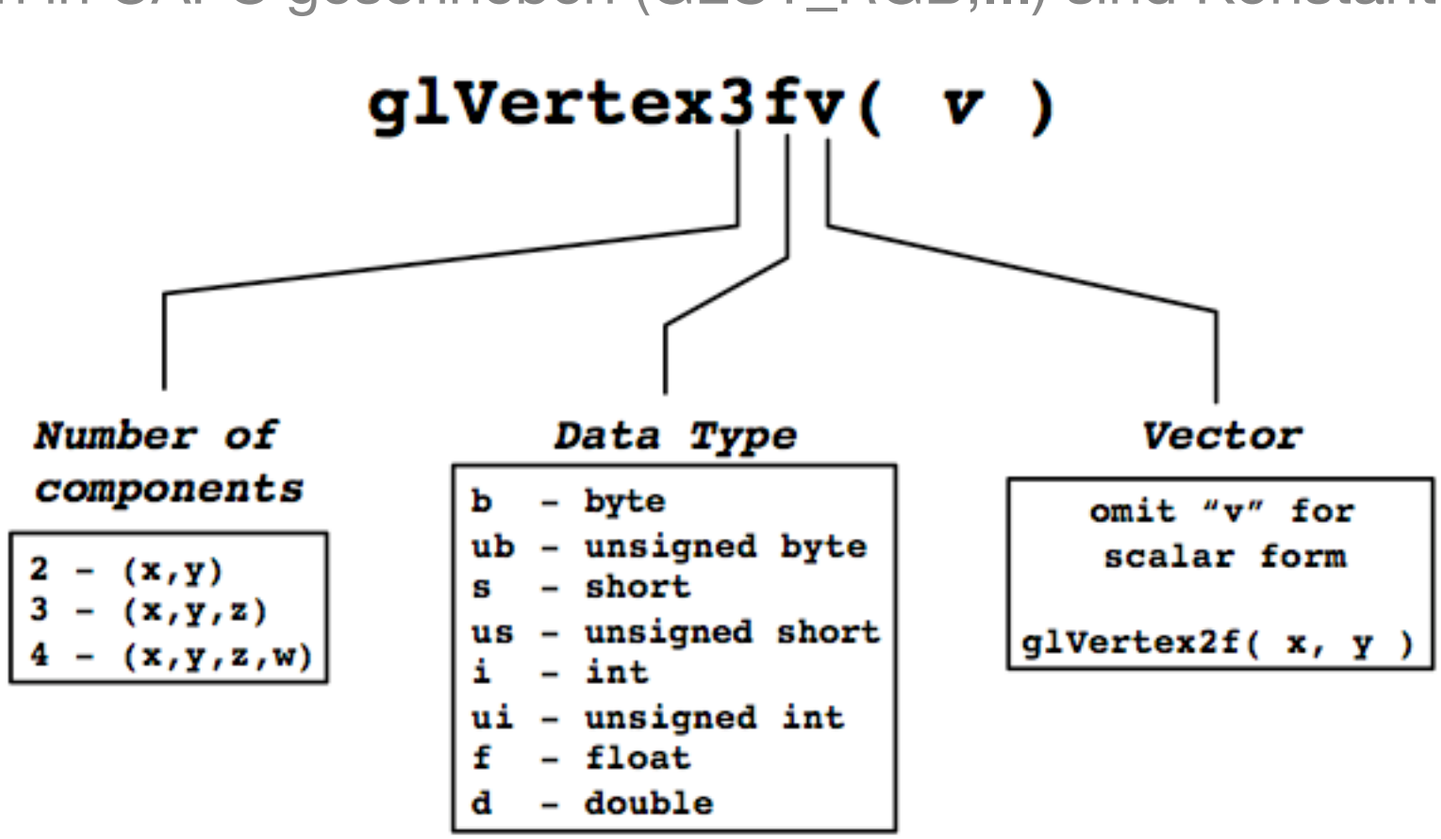


OpenGL

- Verwandte APIs
  - AGL, GLX, WGL
    - Verbindung von OpenGL und Window system
  - GLU (OpenGL Utility Library)
    - Teil von OpenGL
- GLUT (OpenGL Utility Toolkit)
  - Portable windowing API
  - Tastaturein- und -ausgaben
  - Würfel, Zylinder, Kugel, Teapot,...

OpenGL

- Konventionen
  - Funktionen starten mit gl (glColor( ), ...)
  - glu Funktionen sind Utility Funktionen (gluLookAt( ), gluPerspective( ), ...)
  - Variablen in CAPS geschrieben (GLUT_RGB,...) sind Konstanten



OpenGL

- Basic OpenGL Template

```
/* simple program template for
    OpenGL progs */

#include <GL/glut.h>

void myDisplay()
{
    /* clear the window */
    glClear(GL_COLOR_BUFFER_BIT);
    /* draw something */
    glBegin(GL_LINES);
          glVertex2f(-0.5, -0.5);
          glVertex2f(0.5, 0.5);
    glEnd();
    glFlush();
}
```

```
int main (int argc,
            char** argv)
{
    glutInit(&argc, argv);
    glutCreateWindow("basic
    template 1");
    glutDisplayFunc(myDisplay);
    glutMainLoop();
}
```

OpenGL

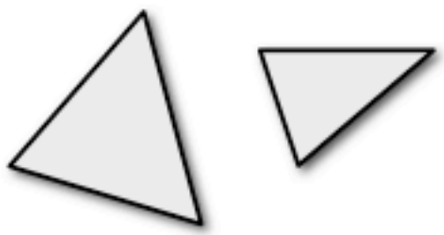
- OpenGL Primitive

GL_POINTS

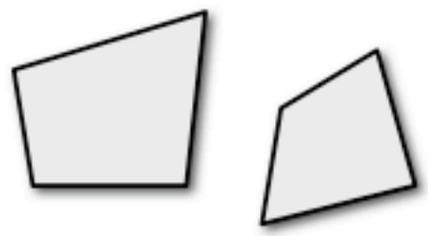
GL_LINES

GL_LINE_STRIP

GL_LINE_LOOP

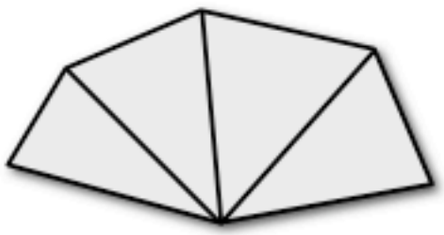


GL_TRIANGLES

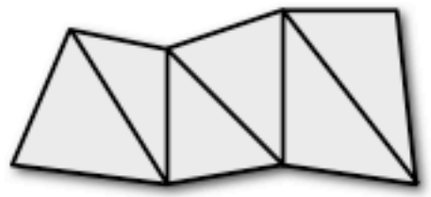


GL_QUADS

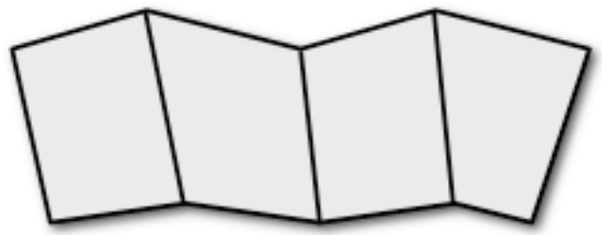
GL_POLYGON



GL_TRIANGLE_FAN



GL_TRIANGLE_STRIP



GL_QUAD_STRIP

OpenGL

- GLUT Callbacks

```
glutDisplayFunc( display );
glutIdleFunc( idle );
glutResizeFunc( resize );
glutKeyboardFunc( keyboard );
glutSpecialFunction( special )
glutMouseFunc( mouse );
glutMotionFunc( mouse_motion );
glutPassiveMotionFunc( mouse_pmotion );
glutEntryFunc( on_focus_change );
```

OpenGL

- In der Display passiert alles wichtige!!

```
void display( void )
{
  glClear( GL_COLOR_BUFFER_BIT );
  glBegin( GL_LINES );
    glVertex2f( 50.0, 50.0 );
    glVertex2f( 100.0, 100.0 );
    glVertex2f( 70.0, 10.0 );
    glVertex2f( 100.5, 70.1 );
  glEnd();
  glFlush();
}
```

Herzlichen Dank!

- Mehr Informationen findet ihr bei:
  - Mario Botsch: graphics.uni-bielefeld.de
  - Marc Latoschik: AG WBS (ehemalige Mitarbeiter)
  - Uni Osnabrück: http://www-lehre.inf.uos.de/~cg/

- In 15 Minuten treffen auf M4-122