Computergraphik

Vorlesung im Wintersemester 2012/13
Kapitel 7: Rasterisierung und OpenGL

Prof. Dr.-Ing. Carsten Dachsbacher
Lehrstuhl für Computergrafik
Karlsruher Institut für Technologie

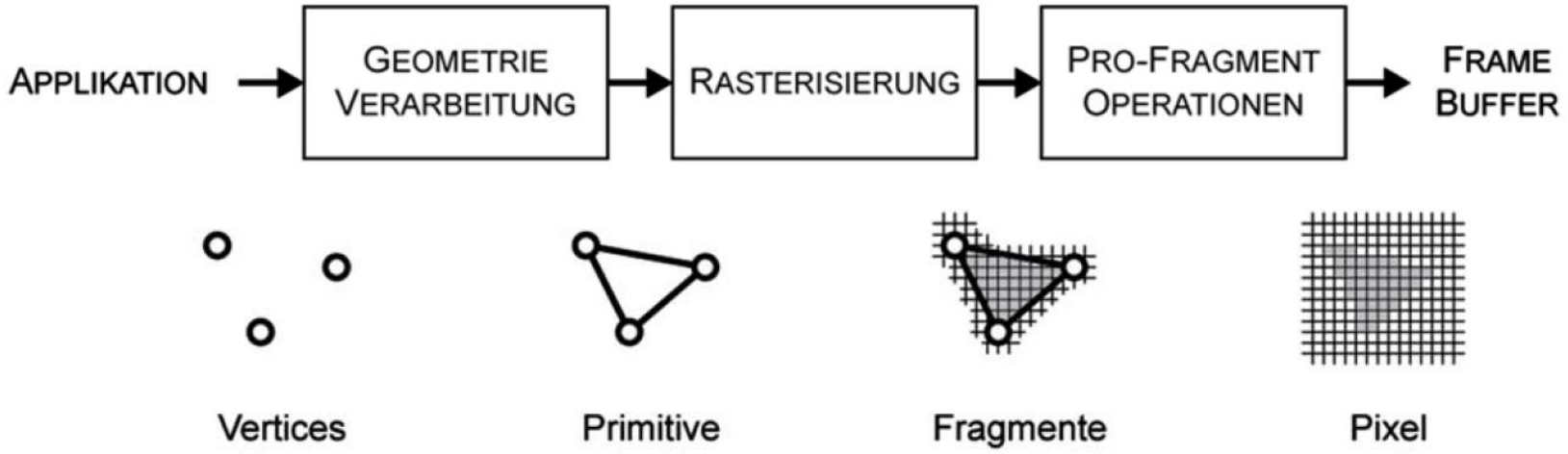
Inhalt

- Rasterisierungspipeline
- Projektionen
- Perspektivisch korrekte Interpolation
- Tiefenpuffer (Z-Buffer)

- OpenGL

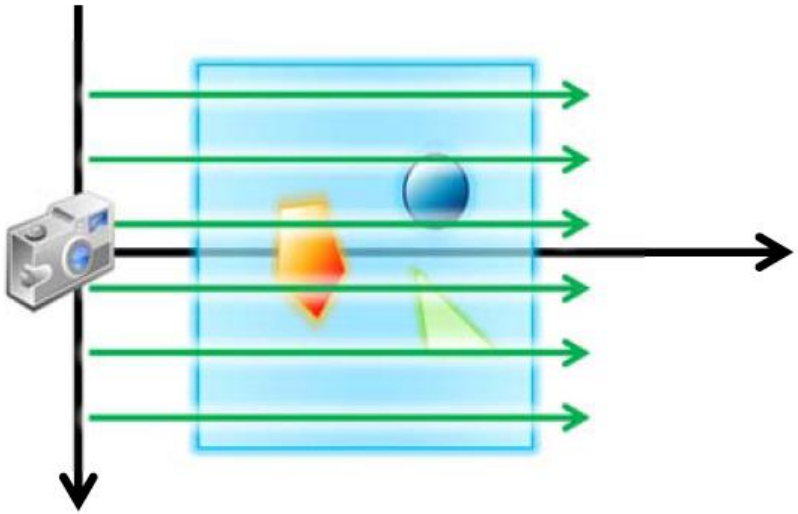
Rasterisierungspipeline

- Abbilden der Geometrie (Dreiecke) auf 2D Bildschirmkoordinaten
- Rasterisierung (siehe 1. Übungsblatt)
- Interpolation von Vertex-Attributen
- Tiefentest und andere Fragmentoperationen

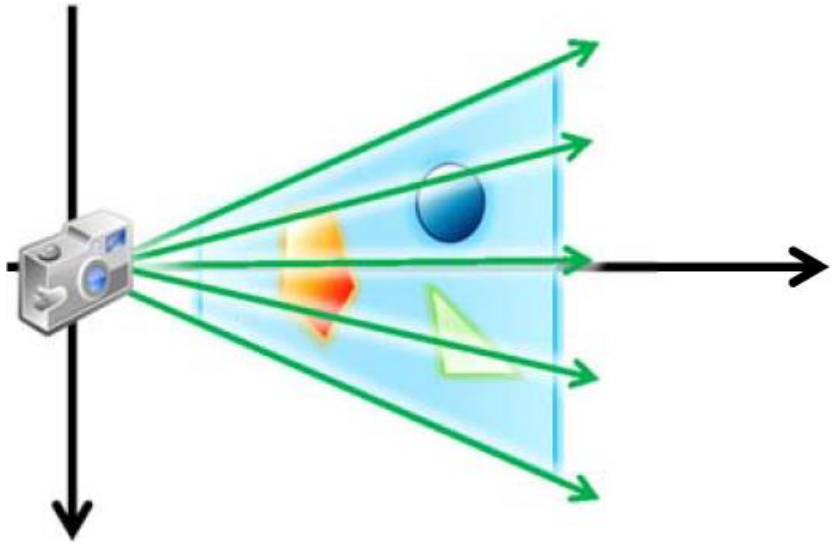


Projektionen

- Abbilden der Geometrie (Dreiecke) auf 2D Bildschirmkoordinaten
- Mehrere Möglichkeiten
  - Orthographisch
  - Perspektivisch



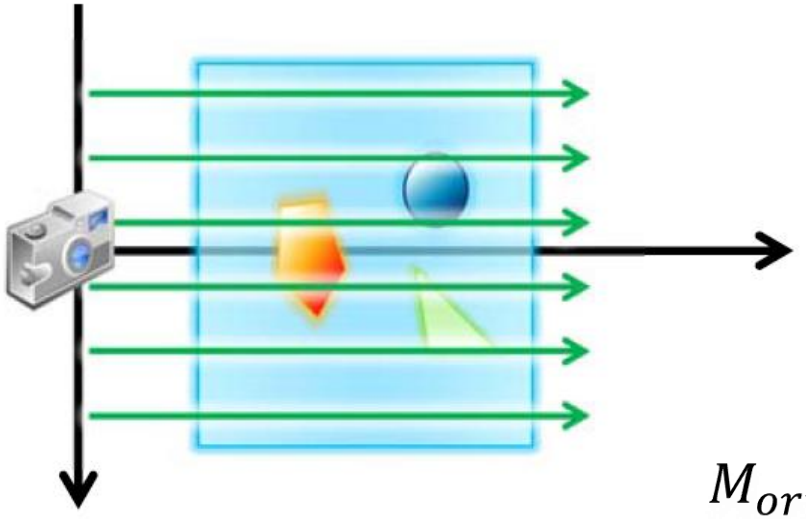
Orthographische Kamera:
Sichtvolumen ist ein Quader



Perspektivische Kamera:
Sichtvolumen ist ein Pyramidenstumpf

Projektionen

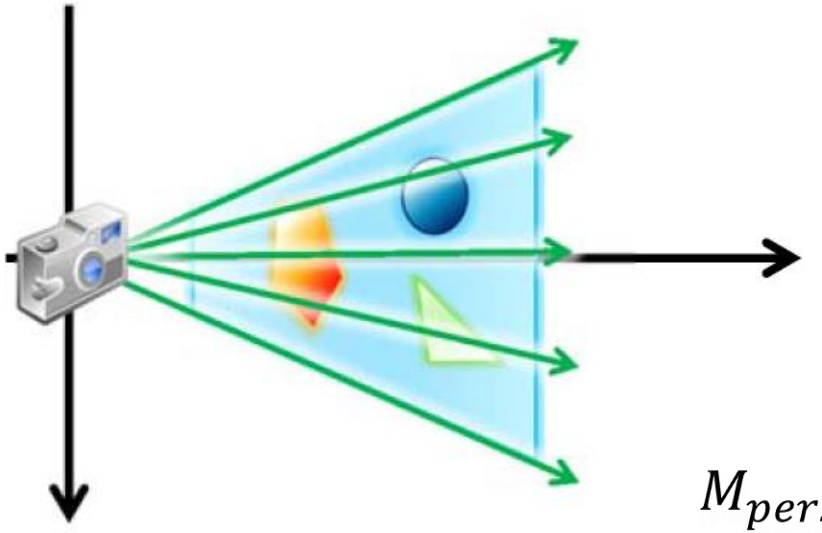
- Abbilden der Geometrie (Dreiecke) auf 2D Bildschirmkoordinaten
- **Zwischenschritt:** Abbildung des Sichtvolumens auf den Einheitswürfel
- Orthographisch: Einfach!
- Sichtvolumen: $[r, l] \times [t, b] \times [n, f] \rightarrow [-1,1]^3$



$$M_{orth} = \begin{pmatrix} \frac{2}{r-l} & 0 & 0 & -\frac{r+l}{r-l} \\ 0 & \frac{2}{t-b} & 0 & -\frac{t+b}{t-b} \\ 0 & 0 & \frac{2}{n-f} & -\frac{n+f}{n-f} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Projektionen

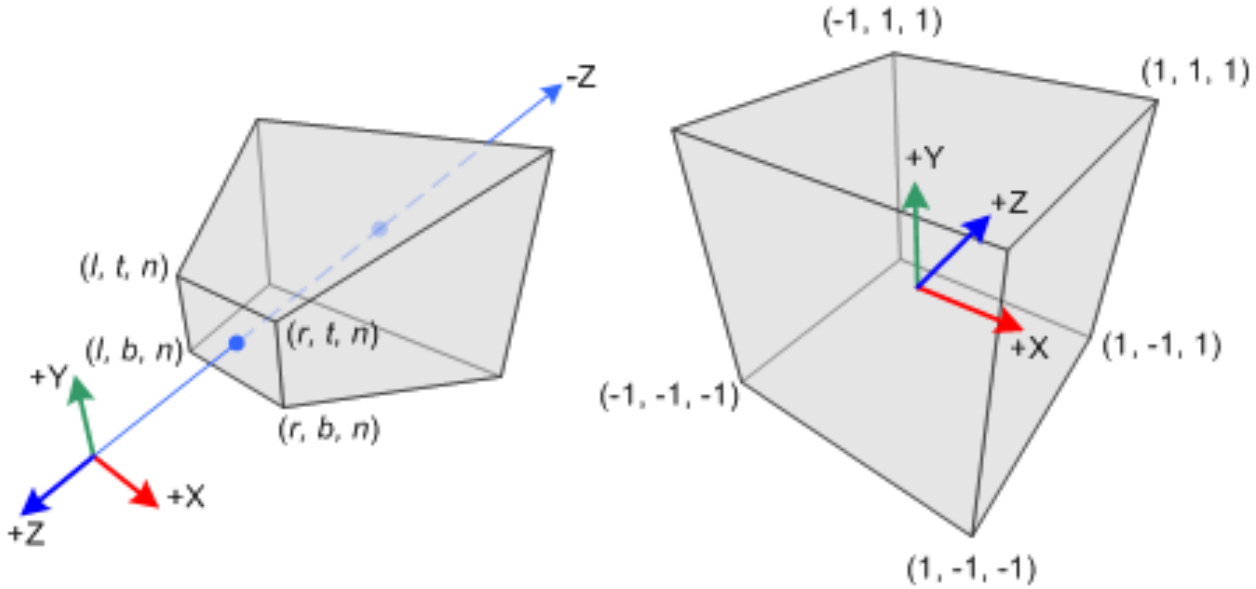
- Abbilden der Geometrie (Dreiecke) auf 2D Bildschirmkoordinaten
- **Zwischenschritt:** Abbildung des Sichtvolumens auf den Einheitswürfel
- Perspektivisch: Projektionszentrum im Ursprung
- Sichtvolumen definiert durch: $\mathrm{r}, l, b, t, n, f$



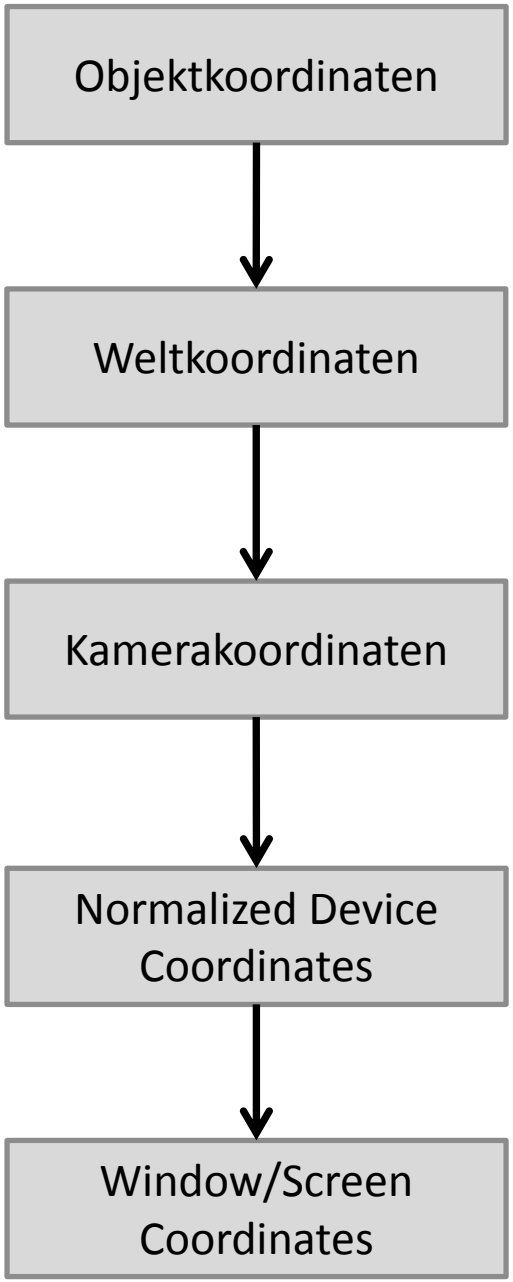
$$M_{pers} = \begin{pmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & \frac{n+f}{n-f} & -\frac{2nf}{f-n} \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

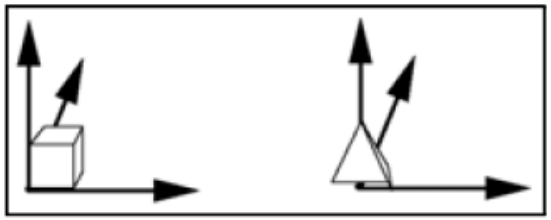
Projektionen

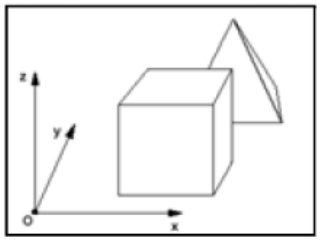
- Multiplikation mit M_{pers}/M_{orth} und anschließender Dehomogenisierung beschreibt die Abbildung „Sichtvolumen → Kanonisches Volumen“
- Die eigentliche Projektion ist dann das Weglassen der z-Komponente
- Letzter Schritt: Viewport Transformation
  - $[-1,1]^2 \rightarrow [0, screenWidth] \times [0, screenHeight]$

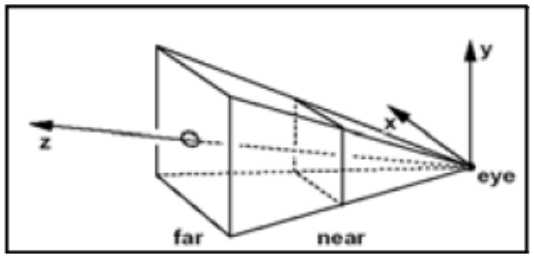


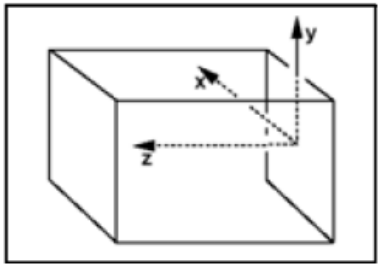
Koordinatensystem Pipeline











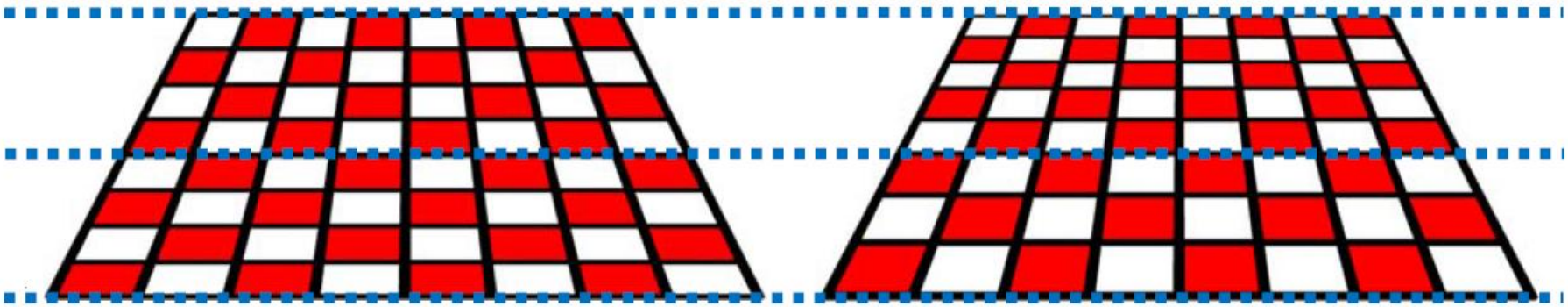
Perspektivisch-korrekte Interpolation

- Naive (affine) Interpolation im Bildraum führt zu falschen Ergebnissen bei perspektivischen Projektionen
- Richtige Tiefeninterpolation:

$$\frac{1}{z_t} = \left(s\frac{1}{z_1} + (1-s)\frac{1}{z_2} \right)$$

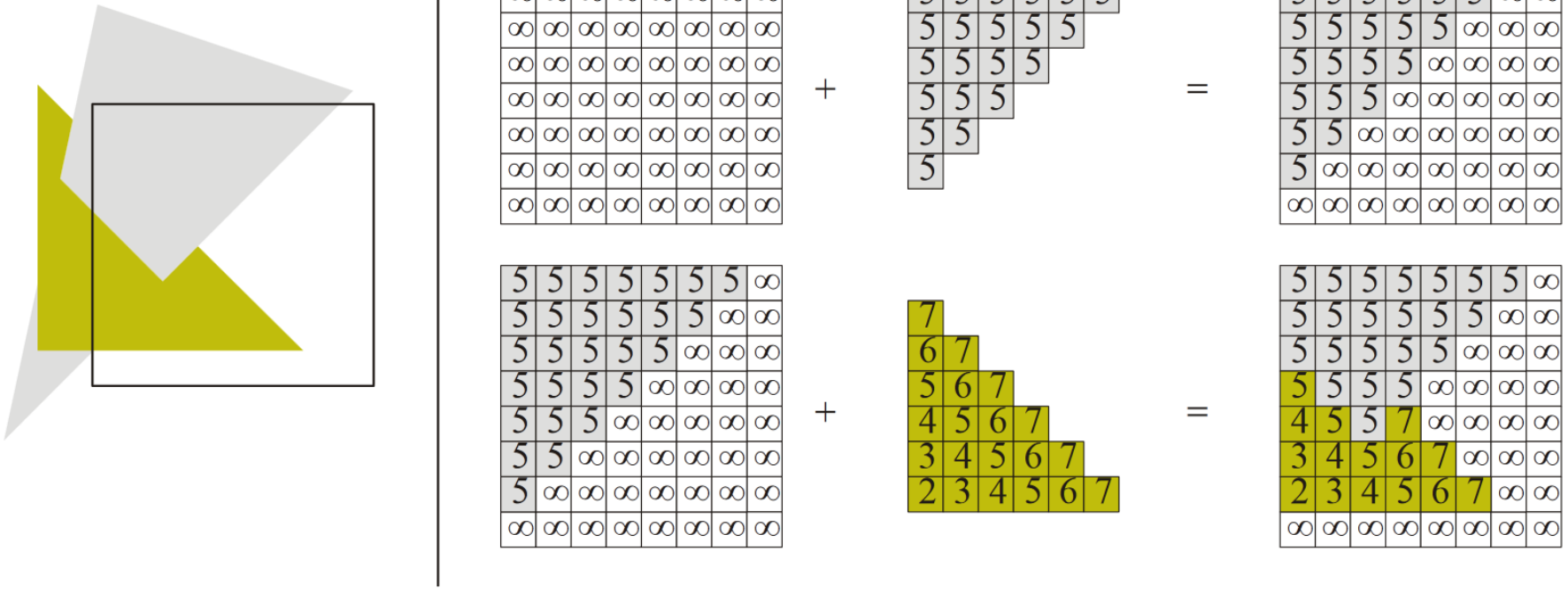
- Allgemein:

$$\frac{I_t}{z_t} = \left(s\frac{I_1}{z_1} + (1-s)\frac{I_2}{z_2} \right)$$



Z-Buffer

- Entscheide Sichtbarkeit durch Speichern der Distanz des nahesten Objekts pro Pixel



OpenGL

- Plattformunabhängige API zur Programmierung von Graphikhardware
- Statemachine
- Klassisches vs. modernes OpenGL

- Window- und Event-Handler: GLUT, GLFW, Qt, ...
  - Fenster erzeugen, Benutzereingaben behandeln, ...

- Extension Manager: GLEW
  - Hilft beim Umgang mit dem OpenGL-Extension Mechanismus

OpenGL

- Schematische Hauptschleife

```
display() {
        clearBuffers();
        setupCamera();
        drawScene();
        swapBuffers();
}


drawScene() {
        for each object {
                stateSetup();
                drawObject();
                stateClean();
        }
}
```

OpenGL – Immediate mode

- Matrix Stack
  - z.B. für die Model-View Transformtionen

```
glMatrixMode(GL_MODELVIEW);

glPushMatrix();
glTranslatef(40, 0, 30);
drawObject1();
glPopMatrix();

glPushMatrix();
glTranslatef(40, 0, -30);
drawObject2();
glPopMatrix();
```

OpenGL – Immediate mode

- Geometriedaten
  - Layout: GL_TRIANGLE, GL_TRIANGLE_STRIP, ...

```
drawObject() {
      glBegin(GL_TRIANGLES);
      glVertex3f(0, 0, 0);
      glVertex3f(1, 0, 0);
      glVertex3f(1, 1, 0);
      glVertex3f(0, 0, 0);
      glVertex3f(1, 1, 0);
      glVertex3f(0, 1, 0);
      glEnd();
}
```

GLOpenGL – Immediate mode

- Vertex-Attribute
  - Normale, Farbe, Texturkoordinate, ...

```
drawObject() {
      glBegin(GL_TRIANGLES);
      glColor3f (1, 0, 0);
      glNormal3f(0, 0, 1);
      glVertex3f(1, 0, 0);
      glColor3f (0, 1, 0);
      glNormal3f(0, 0, 1);
      glVertex3f(1, 1, 0);
      glColor3f (0, 0, 1);
      glNormal3f(0, 0, 1);
      glVertex3f(0, 0, 0);
      glEnd();
}
```