Interaktive Computergrafik

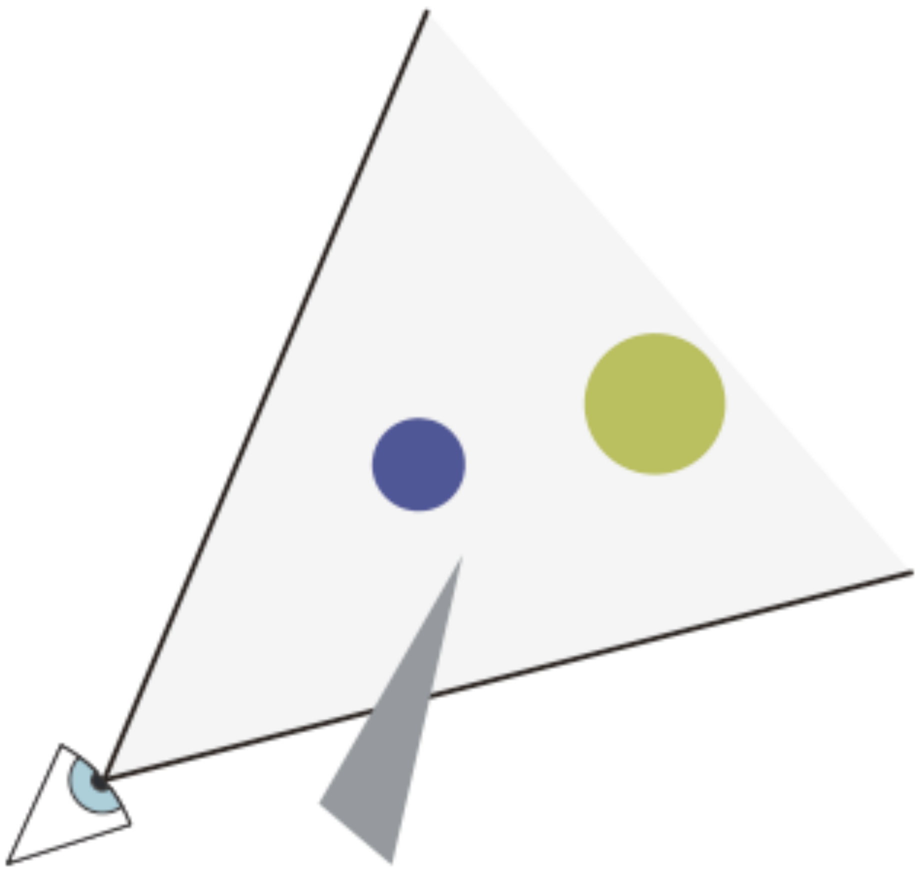
Prof. Dr. Frank Steinicke
Human-Computer Interaction
Fachbereich Informatik
Universität Hamburg

Interaktive Computergrafik

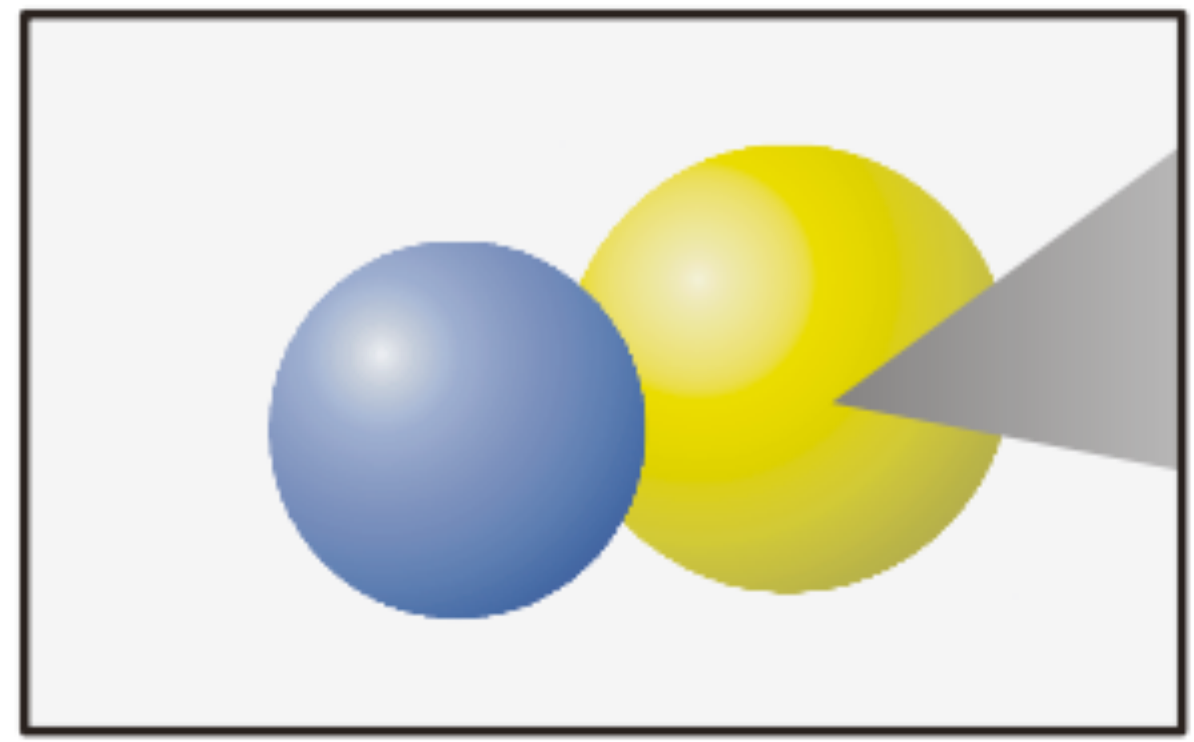
Kapitel Polygonale Modellierung

Prof. Dr. Frank Steinicke
Human-Computer Interaction, Universität Hamburg

Rendering

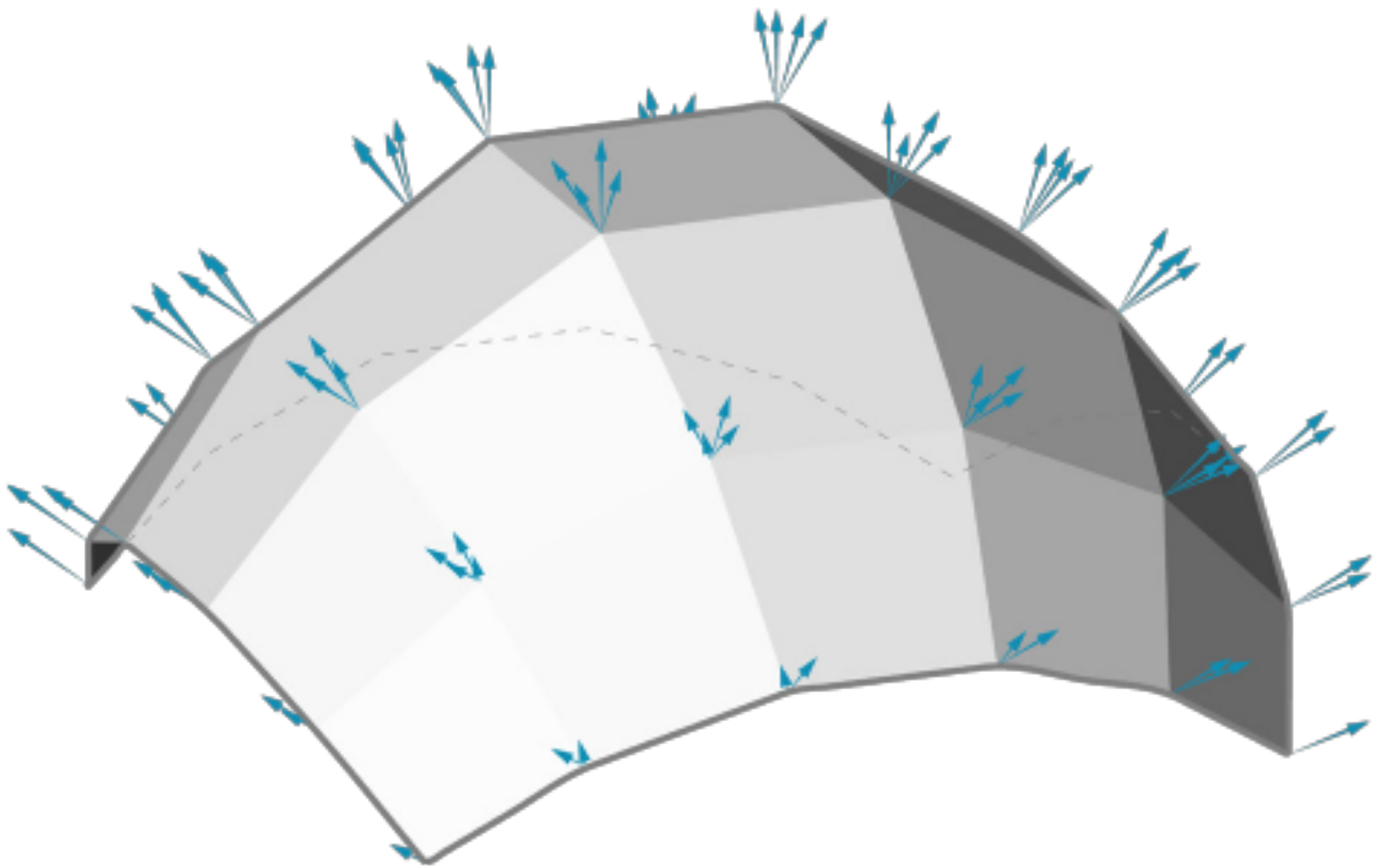


2D/3D-Modell

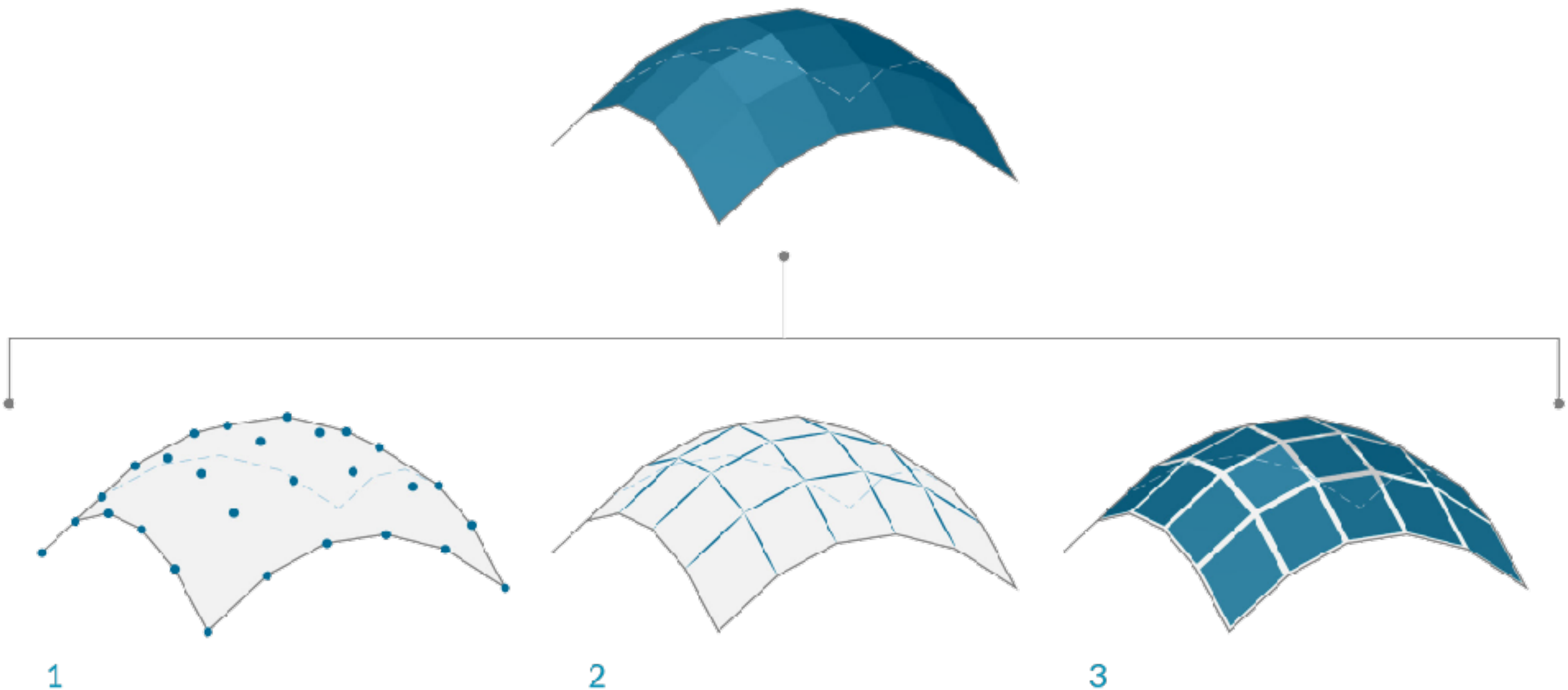


2D-Bild

Polygonnetz
Beispiel



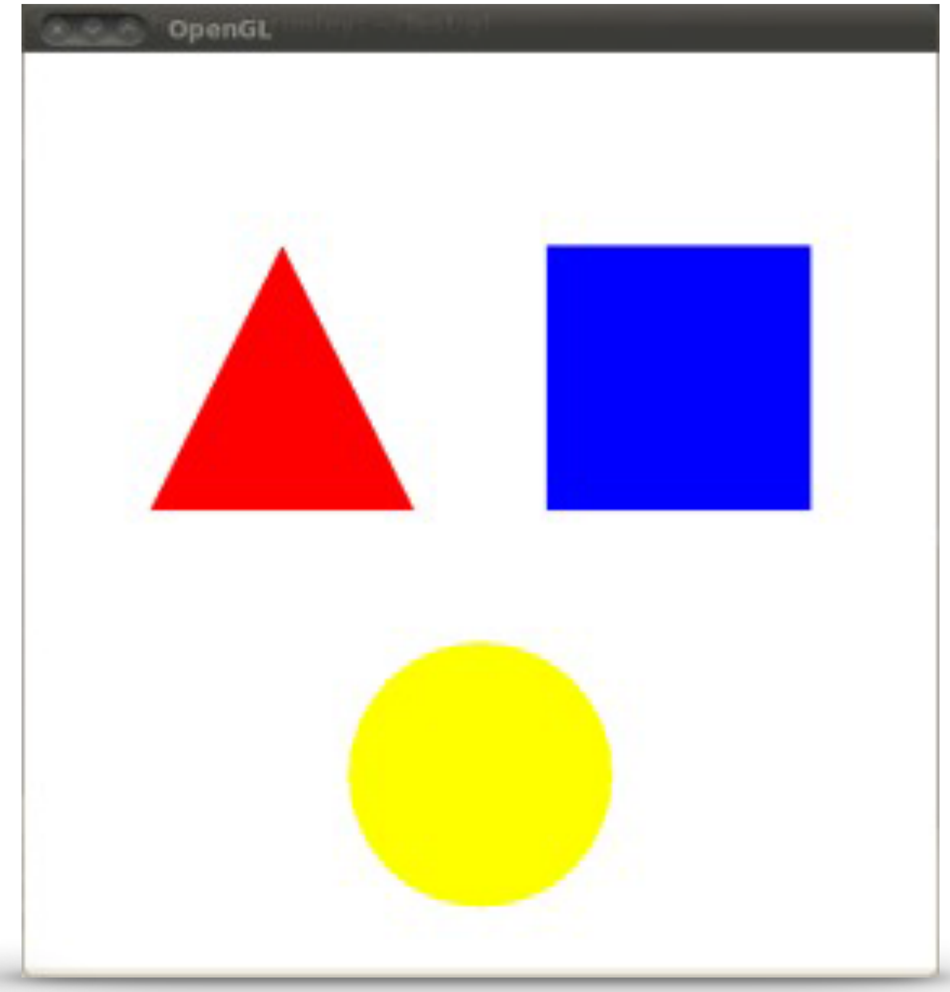
Polygonnetze
Beispiel



Übungen

WebGL, JavaScript

```html
<!DOCTYPE html>
<html>

    <head>

        <title> My Cool Net Art</title>

    </head>

    <body>

        <script src="http://cdnjs.cloudflare.com/ajax/libs/three.js/r58/three.min.js"></script>
        <script>

            renderer = new THREE.WebGLRenderer();
            renderer.setSize( window.innerWidth, window.innerHeight );
            document.body.appendChild( renderer.domElement );

            camera = new THREE.PerspectiveCamera( 50, window.innerWidth/window.innerHeight, 1, 10000 );
            camera.position.z = 500;

            scene = new THREE.Scene();

            geometry = new THREE.CubeGeometry(200, 200, 200);
            material = new THREE.MeshNormalMaterial({shading: THREE.FlatShading});
            mesh = new THREE.Mesh(geometry, material);
            scene.add(mesh);

            renderer.render( scene, camera );

        </script>

    </body>
</html>
```



Agenda

- Vektoren und Vektorräume
- Polygonnetze
- Boundary Representation
- Alternative Modellrepräsentationen

Interaktive Computergrafik

Kapitel Polygonale Modellierung

Vektoren und Vektorräume

Koordinatensystem

- Grundlage ist **2D/3D-Vektorraum** repräsentiert als **kartesisches Koordinatensystem** mit **orthogonalen Richtungsachsen**, d.h. Achsen schneiden sich im 90°-Winkel

Vektoren

- **Vektoren** werden zur Darstellung von geometrischen Objekten (Koordinaten) verwendet

- **Vektoren** im $\mathbb{R}^n$ sind geordnete Tupel von n reellen Zahlen:

$$v = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} \in \mathbb{R}^n; v_1, ..., v_n \in \mathbb{R}$$

Vektoren

- **Transponierte Repräsentation** eines Vektor ist gegeben durch

$$\begin{pmatrix} x \\ y \end{pmatrix}^T = (x, y)$$

- Vektoren können addiert/subtrahiert oder mit bzw. durch Skalare multipliziert bzw. dividiert werden
- Resultat ist Vektor desselben Vektorraums

Vektoren

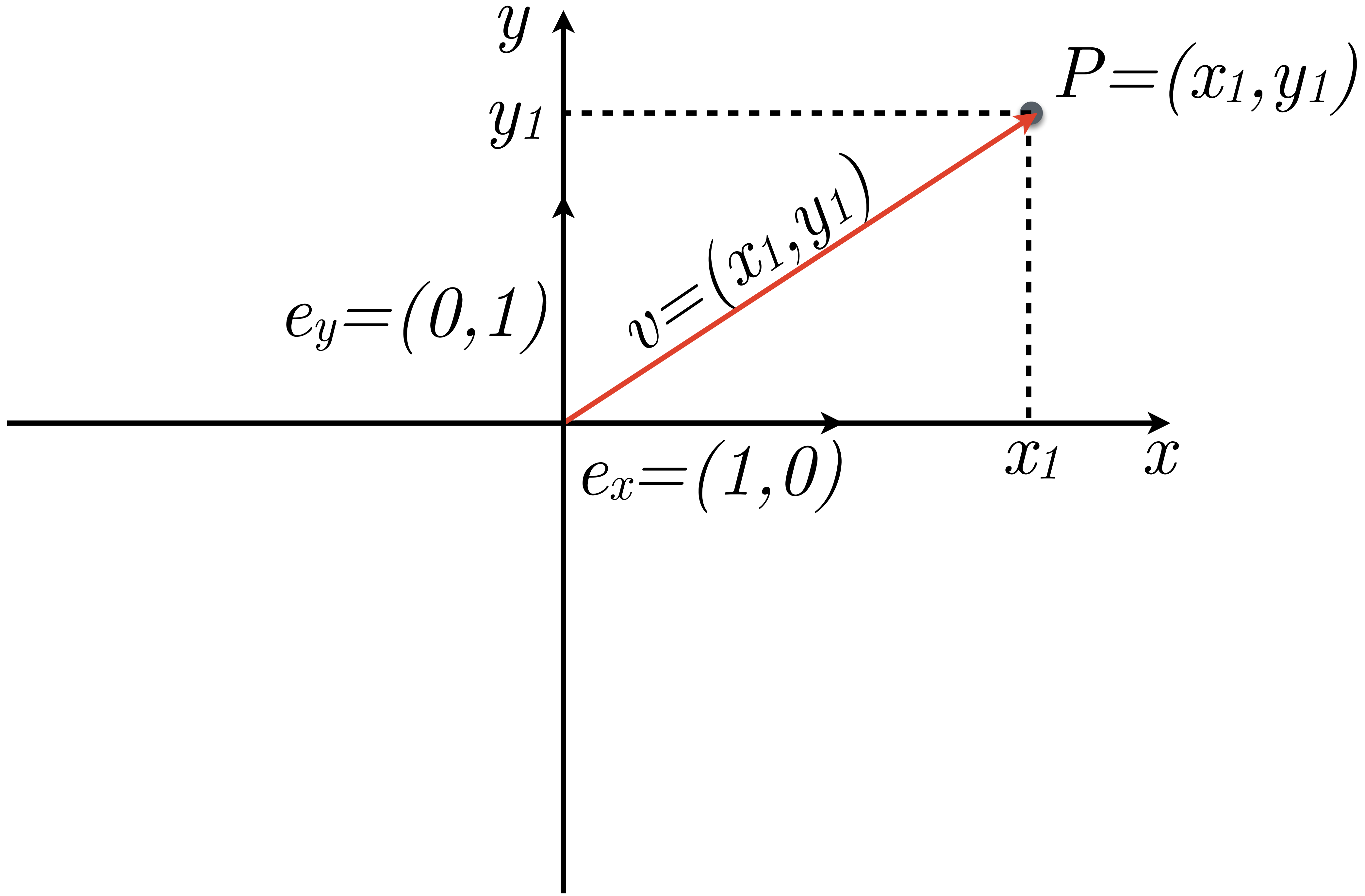
- typischerweise betrachten wir zwei-, drei- und vierdimensionale Vektoren, d.h.

$$\begin{pmatrix} x \\ y \end{pmatrix}, \begin{pmatrix} x \\ y \\ z \end{pmatrix}, \begin{pmatrix} x \\ y \\ z \\ w \end{pmatrix}$$

- Vektoren können zur Repräsentation von geometrischen Objekten durch **Koordinaten** oder **Punkte** (*engl.* ***Vertices*** *(sing.* ***Vertex****))* verwendet werden

2D-Vektoren

Beispiel



Vektoraddition

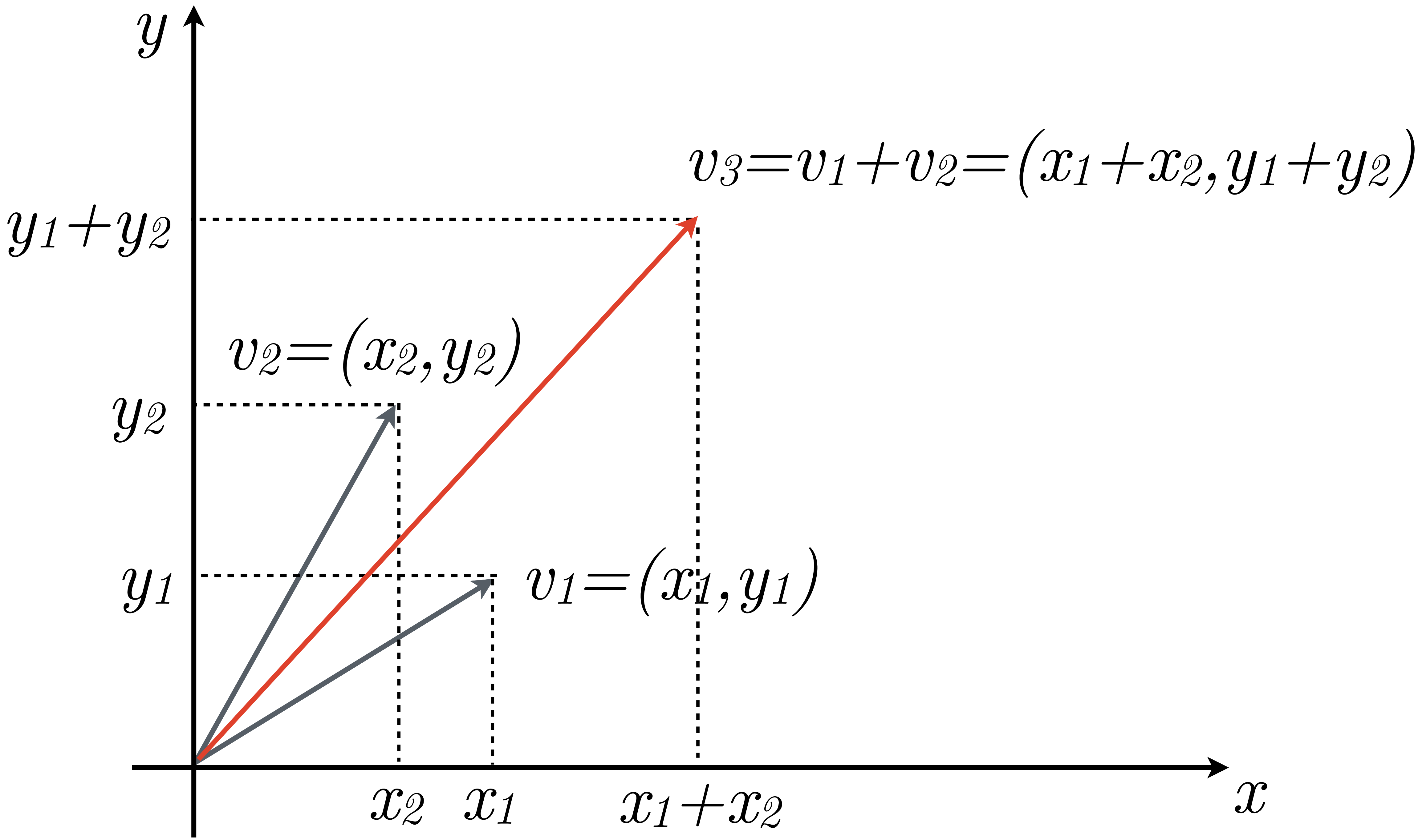
- Für n-dimensionale Vektoren v, w gilt:

$$v + w = \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} + \begin{pmatrix} w_1 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} v_1 + w_1 \\ \vdots \\ v_n + w_n \end{pmatrix}$$

- **Vektoraddition** erfolgt komponentenweise

Vektoraddition

Beispiel



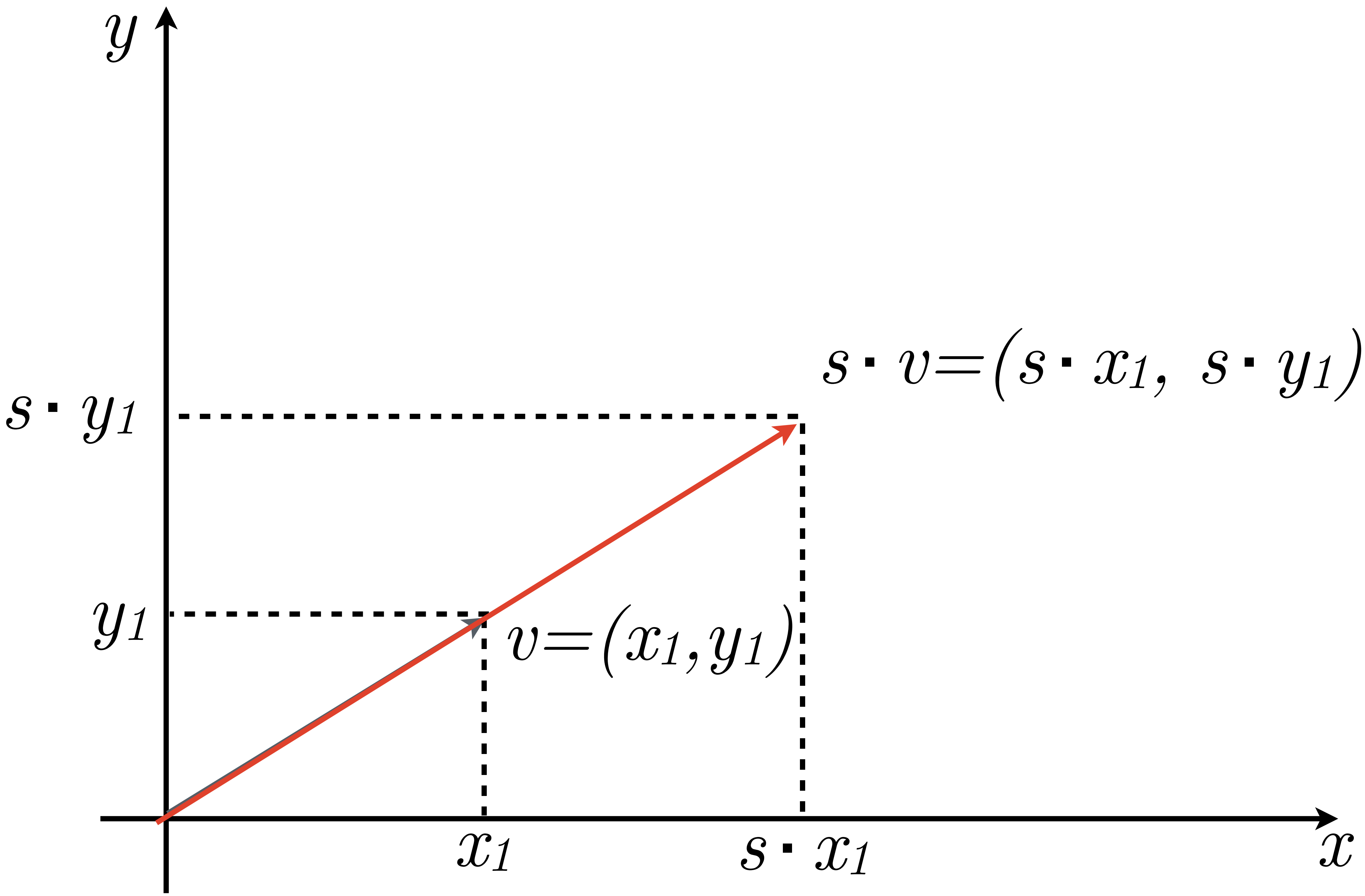
Skalierung von Vektoren

- Jeder Vektor v kann mit reellwertiger Zahl s skaliert werden

$$s \cdot v = s \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} s \cdot v_1 \\ \vdots \\ s \cdot v_n \end{pmatrix}$$

Skalierung von Vektoren

Beispiel



Gerade

- **Gerade** kann in **parametrischer Form** durch einen **Stützvektor** v (auf Linie) und einem **Richtungsvektor** w mit Parameter $t \in \mathbb{R}$ definiert werden

$$g(t) = v + t \cdot w$$

$$g : \mathbb{R} \to \mathbb{R}^n$$

Linie

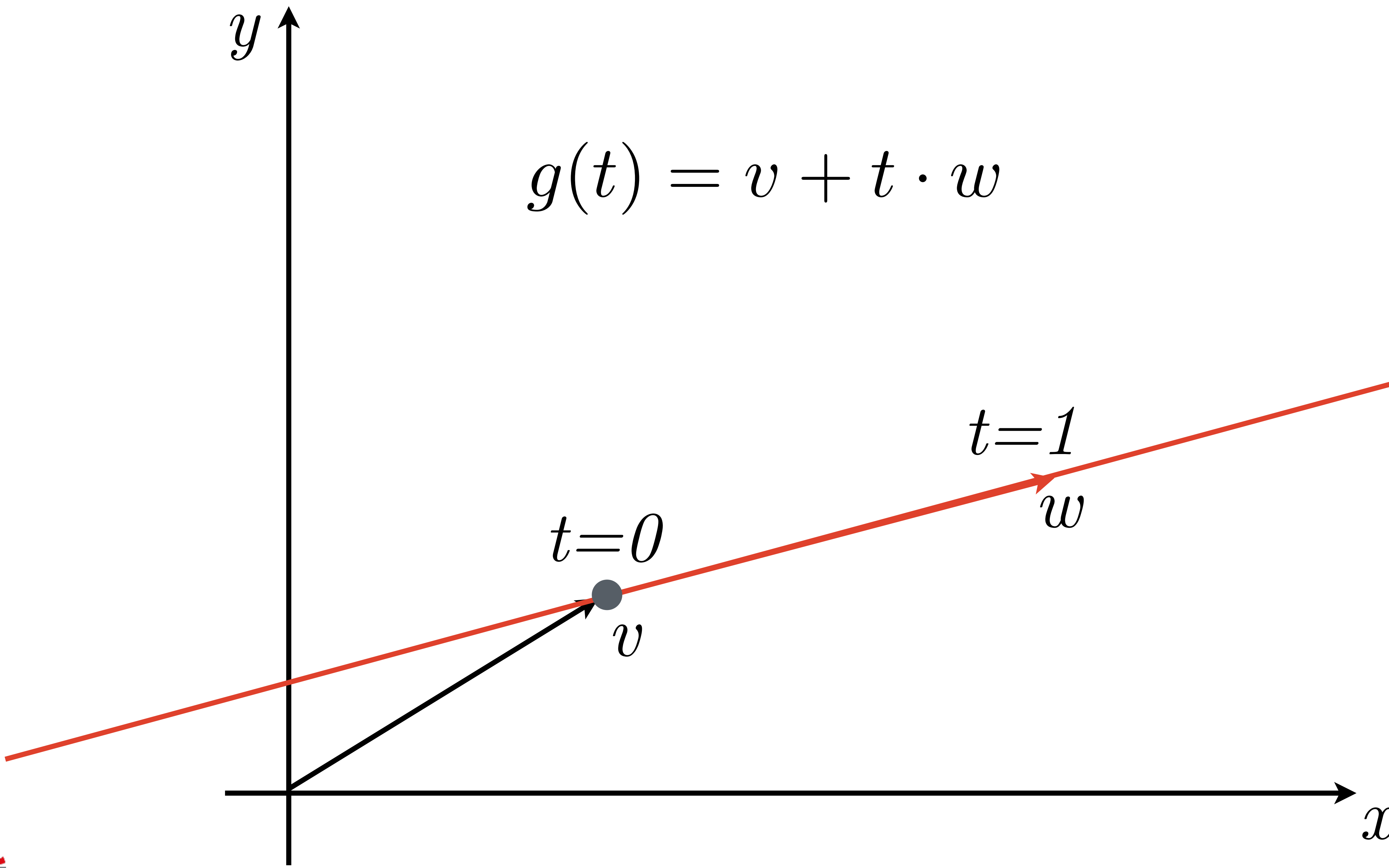
- **Linie** kann in **parametrischer Form** durch einen **Startvektor** v (zum Startpunkt) und einem **Endpunkt** $v+w$ mit Parameter definiert werden $t \in [0,1]$

$$g(t) = v + t \cdot w$$

$$g : \mathbb{R} \to \mathbb{R}^n$$

Linie / Gerade

Beispiel

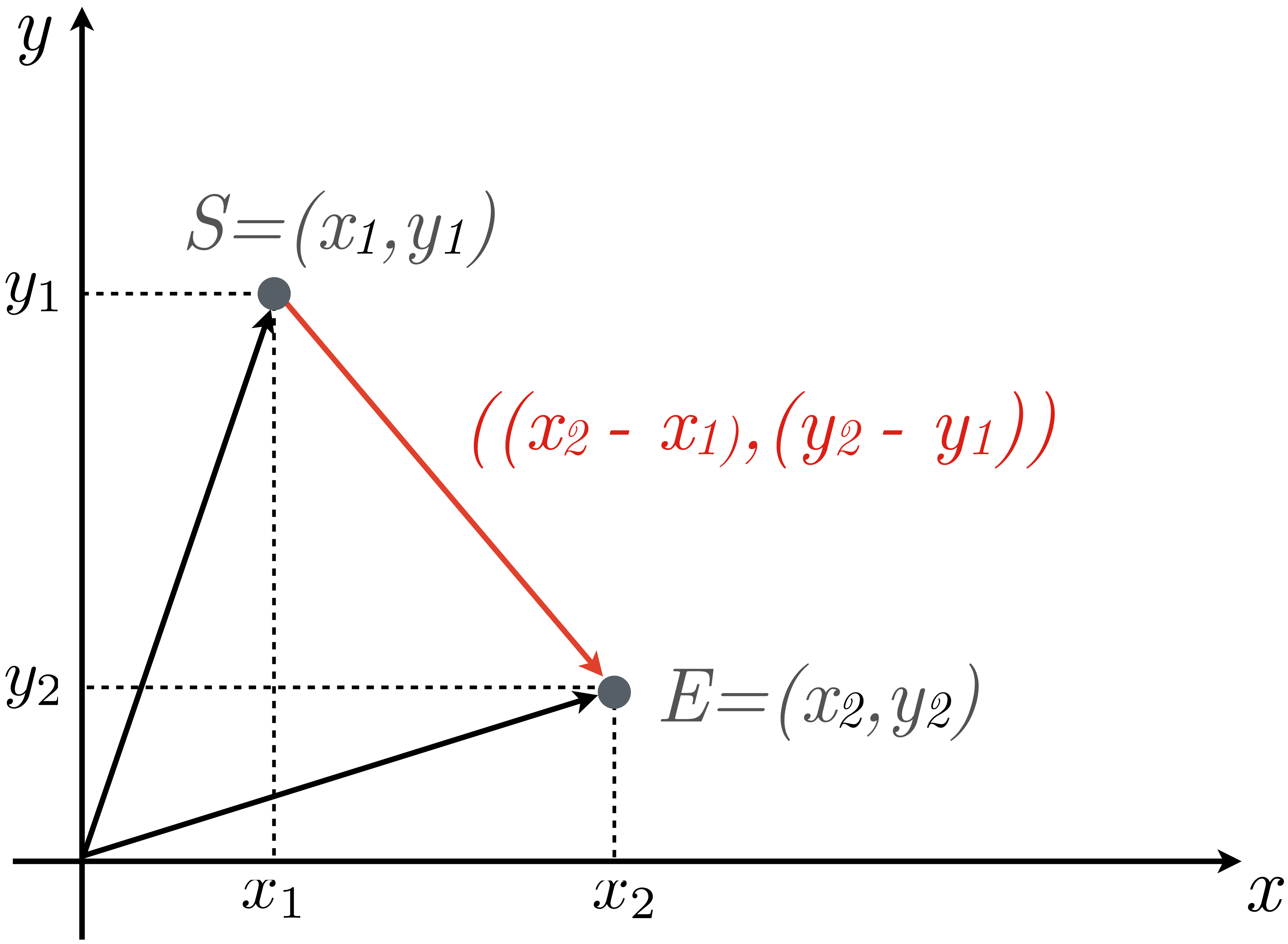


Punkte und Linie

- **Punkt** bzw. **Vertex** wird durch Koordinaten $V=(x_1,y_1)$ beschrieben
- **Linie** wird durch **Startpunkt** $S=(x_1,y_1)$ und **Endpunkt** $E=(x_2,y_2)$ beschrieben, d.h. $L=\{S,E\}=\{(x_1,y_1),(x_2,y_2)\}$
- **Vektor** vom **Start**- zum **Endpunkt** $v=E-S=((x_2 - x_1),(y_2 - y_1))$

Linie

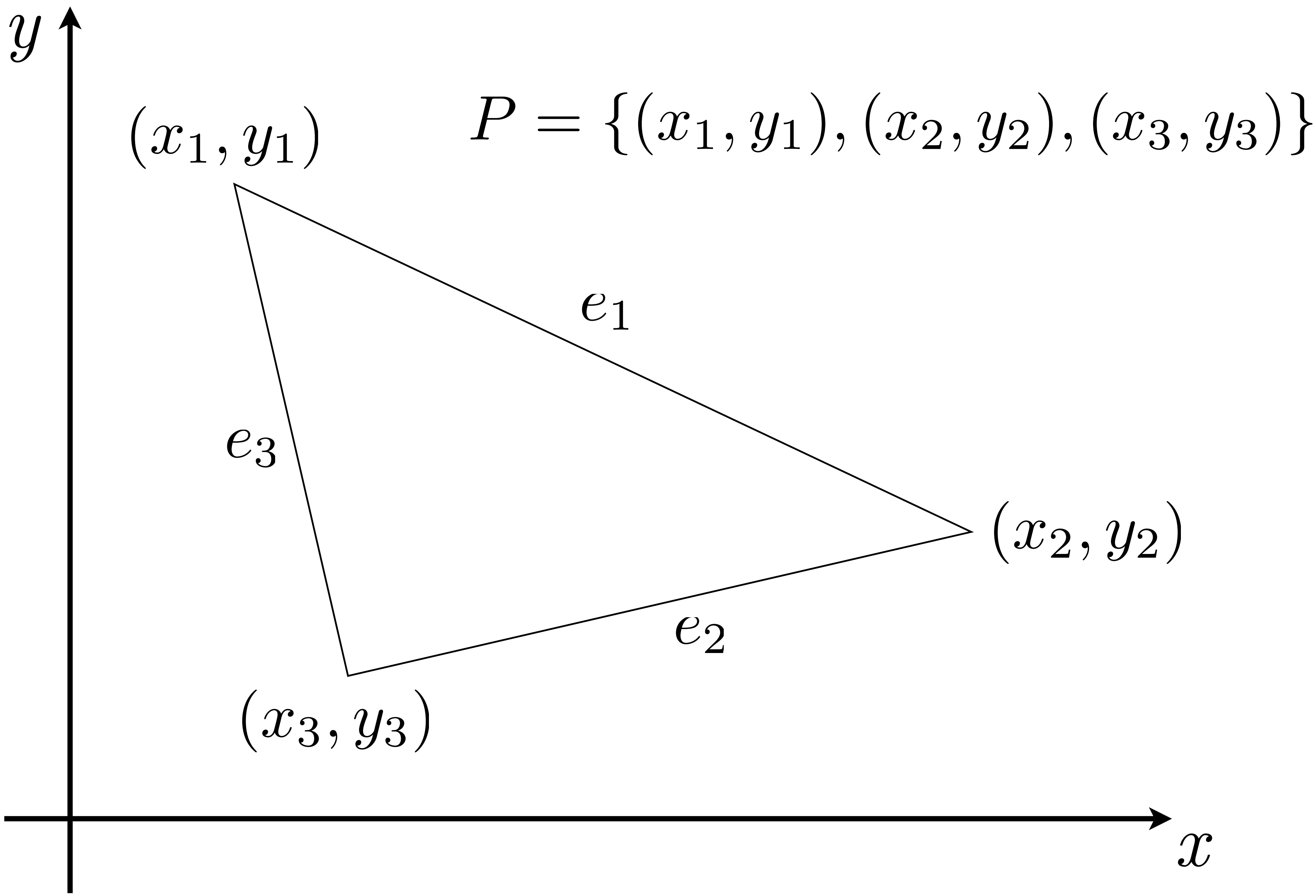
Beispiel



Polygon

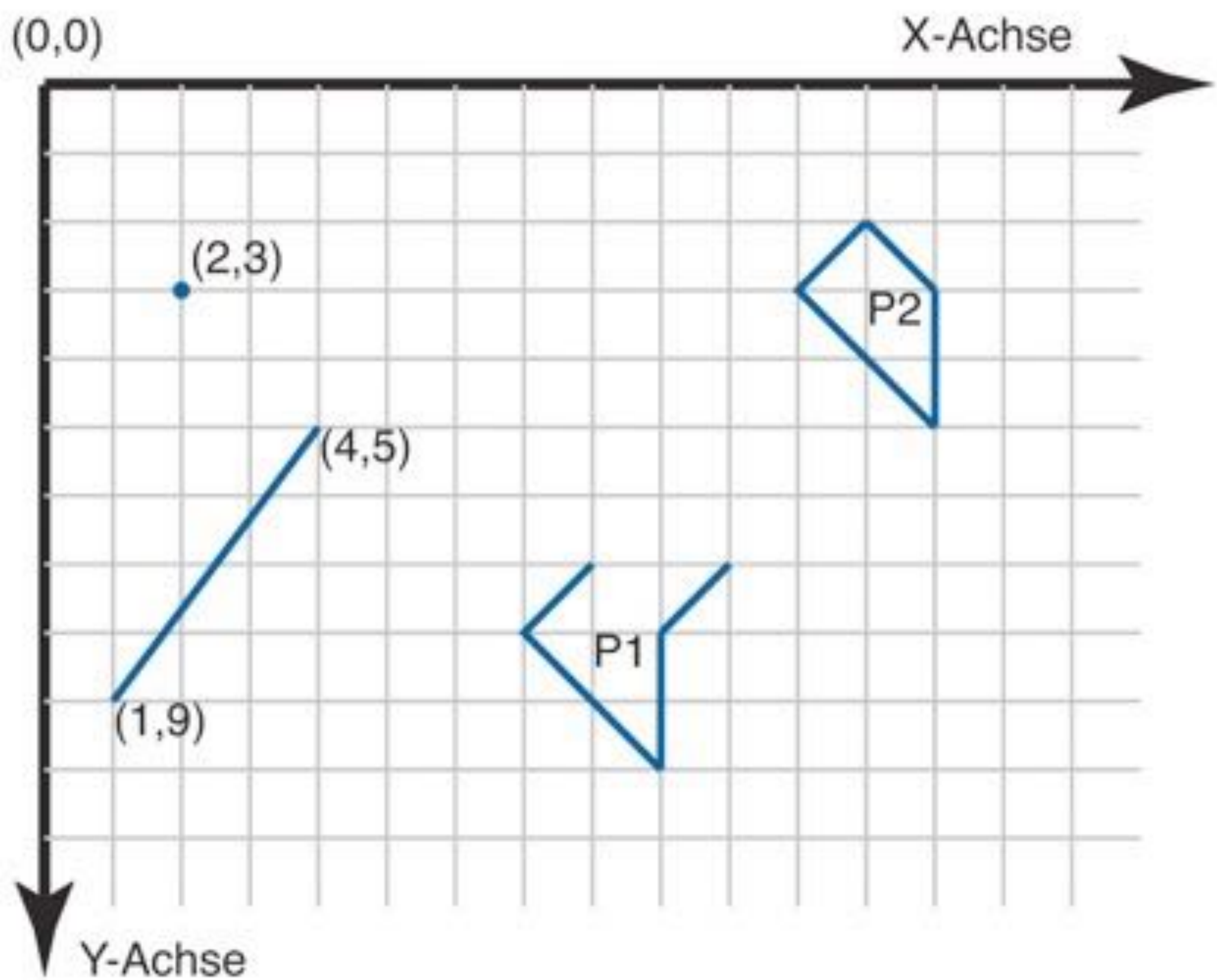
- **Polygon** besteht aus mehreren verbundenen Punkten, d.h. $P=\{(x_1,y_1),\ldots,(x_n,y_n)\}$
- **Linien** $e_i=\{(x_i,y_i),(x_{i+1},y_{i+1})\}$ mit $i=1,\ldots n\text{-}1$ heißen **Kanten** (*engl.* ***Edges***) des Polygons

Polygon
Beispiel



Geometrische Primitive

Beispiele



Länge / Norm

- **Länge** oder **Norm** $\|v\|$ eines Vektors wird berechnet durch

$$\|v\| := \sqrt{v_1^2 + ... + v_n^2}$$

- Vektoren mit Länge $\|v\| = 1$ werden **Einheitsvektoren** oder normalisierte Vektoren genannt

Vektornormalisierung

- jeder Vektor $v \neq 0$ kann **normalisiert** werden, durch Skalierung mit seiner Länge $\|v\|$

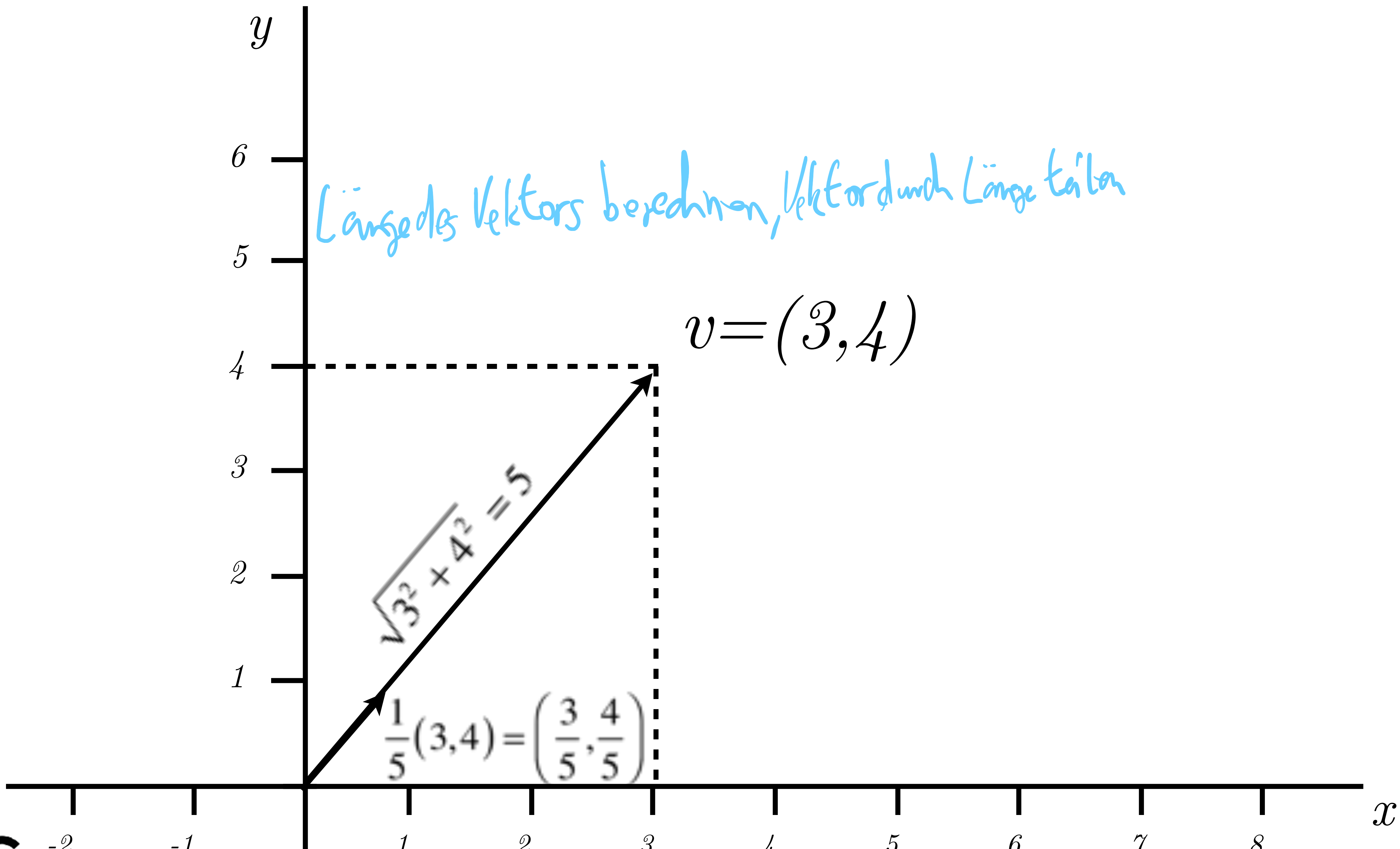
$$\frac{1}{\|v\|} \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix}$$

Diskussion

Normieren Sie folgenden Vektor!

Vektornormalisierung

Länge des Vektors berechnen, Vektor durch Länge teilen



Skalarprodukt

- **Skalarprodukt** (*engl.: **dot product***) ist reellwertige Funktion
- Gegeben: Vektoren u und v im $\mathbb{R}^n$:

$$u \cdot v = u_1 \cdot v_1 + u_2 \cdot v_2 + \ldots + u_n \cdot v_n$$

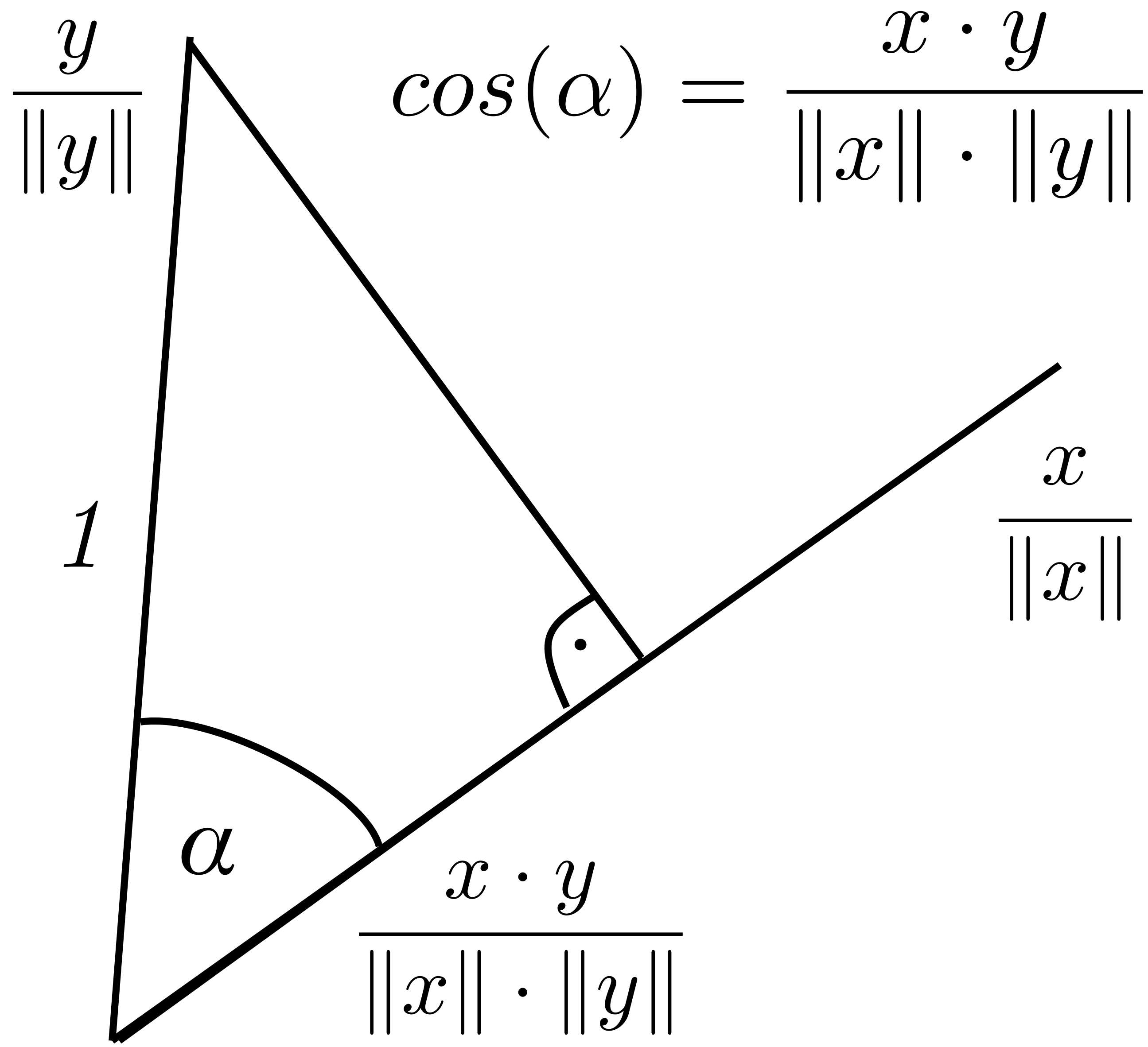
Skalarprodukt
Geometrische Interpretation

- Winkel α zwischen zwei Vektoren x und y

$$cos(\alpha) = \frac{x \cdot y}{||x|| \cdot ||y||}$$

- Winkel α zwischen zwei normalisierten Vektoren

$$cos(\alpha) = x \cdot y$$



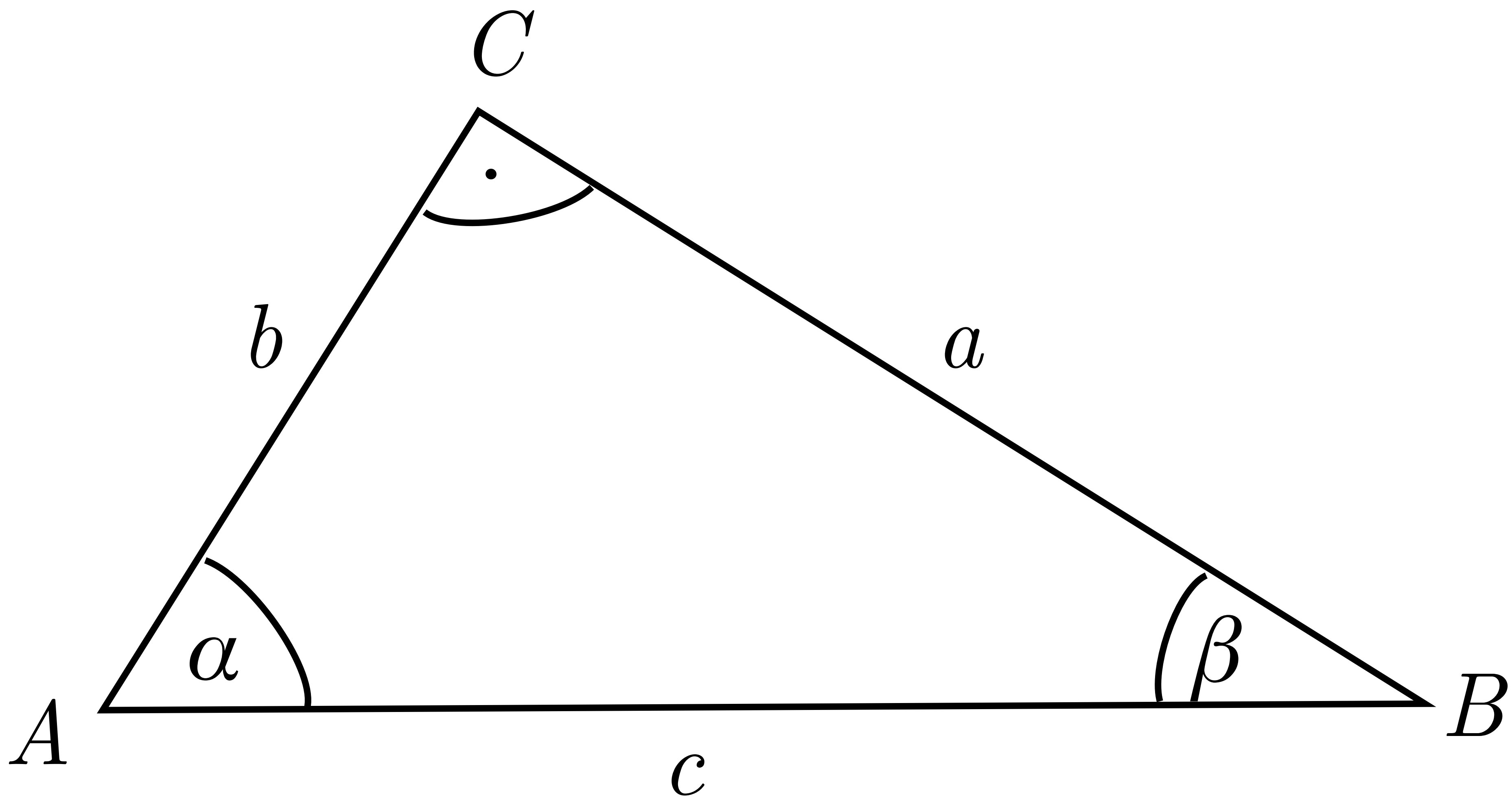
$cos(\alpha) = \frac{x \cdot y}{\|x\| \cdot \|y\|}$
$\frac{y}{\|y\|}$
1
$\frac{x}{\|x\|}$
α
$\frac{x \cdot y}{\|x\| \cdot \|y\|}$

Trigonometrie

$$\sin(\alpha) = \frac{a}{c}$$

$$\cos(\alpha) = \frac{b}{c}$$

$$\tan(\alpha) = \frac{a}{b}$$



Matrix

- **Matrix** (Plural: **Matrizen**) ist tabellarische Anordnung von Elementen in Zeilen und Spalten

$$\begin{array}{cc}
a_{ij} & \xrightarrow{\text{n Spalten, j läuft}} \\
\Big\downarrow \text{m Zeilen, i läuft} & \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} \end{pmatrix}
\end{array}$$

Matrix

Interaktive Computergrafik

- **Matrix** beschreibt Transformationen, hauptsächlich durch quadratische Matrizen, insbesondere 2×2, 3×3, 4×4

a_{ij}

n Spalten, j läuft

m Zeilen, i läuft

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & \cdots & \cdots & a_{mn} \end{pmatrix}$$

Matrizen

- **Einheitsmatrix** *E* (aka **Identitätsmatrix**)
  - alle $a_{ij} = 0$ (für $i \neq j$) und alle $a_{ii} = 1$ (auf Hauptdiagonale)

$$\begin{pmatrix} p_x \\ p_y \\ p_z \\ p_w \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} p_x \\ p_y \\ p_z \\ p_w \end{pmatrix}$$

Matrizen

- **inverse Matrix** M^{-1} für quadratische (n×n)-Matrix M
  - $M^{-1} \cdot M = M \cdot M^{-1} = E$
- **transponierte Matrix** M^T für $(m \times n)$-Matrix M
  - Originalmatrix gespiegelt an Hauptdiagonalen, d.h. $(n \times m)$-Matrix

Matrix
Operationen

- Multiplikation mit Skalar

$$s \cdot \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} = \begin{pmatrix} s \cdot a_{11} & s \cdot a_{12} & s \cdot a_{13} & s \cdot a_{14} \\ s \cdot a_{21} & s \cdot a_{22} & s \cdot a_{23} & s \cdot a_{24} \\ s \cdot a_{31} & s \cdot a_{32} & s \cdot a_{33} & s \cdot a_{34} \\ s \cdot a_{41} & s \cdot a_{42} & s \cdot a_{43} & s \cdot a_{44} \end{pmatrix}$$

- Addition

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix} + \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & b_{34} \\ b_{41} & b_{42} & b_{43} & b_{44} \end{pmatrix} = \begin{pmatrix} a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} & a_{14}+b_{14} \\ a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} & a_{24}+b_{24} \\ a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} & a_{34}+b_{34} \\ a_{41}+b_{41} & a_{42}+b_{42} & a_{43}+b_{43} & a_{44}+b_{44} \end{pmatrix}$$

Matrix-Multiplikation

- Sei $A=(a_{ik})$ eine $(m\times n)$-Matrix und $B=(b_{kj})$ eine $(n\times p)$-Matrix
- Ergebnis der **Matrixmultiplikation** $A\cdot B$ ist $(m\times p)$-Matrix $C=(c_{ij})$ mit
$$c_{ij} = \sum_{k=1}^{n} a_{ik} \cdot b_{kj}$$
- Achtung! Matrixmultiplikation ist nicht kommutativ!

Matrizen-Multiplikation

Beispiel

$$\begin{pmatrix} \boxed{a_{11} \quad a_{12} \quad a_{13}} \\ a_{21} \quad a_{22} \quad a_{23} \\ a_{31} \quad a_{32} \quad a_{33} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} \boxed{c_{11}} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

$$a_{11} \cdot b_{11} + a_{12} \cdot b_{21} + a_{13} \cdot b_{31} = c_{11}$$

Matrizen-Multiplikation

Beispiel

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ \boxed{c_{21}} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

$$a_{21} \cdot b_{11} + a_{22} \cdot b_{21} + a_{23} \cdot b_{31} = c_{21}$$

Matrizen-Multiplikation

Beispiel

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ \boxed{a_{31} \quad a_{32} \quad a_{33}} \end{pmatrix} \cdot \begin{pmatrix} \boxed{\begin{matrix} b_{11} \\ b_{21} \\ b_{31} \end{matrix}} & \begin{matrix} b_{12} \\ b_{22} \\ b_{32} \end{matrix} & \begin{matrix} b_{13} \\ b_{23} \\ b_{33} \end{matrix} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ \boxed{c_{31}} & c_{32} & c_{33} \end{pmatrix}$$

$$a_{31} \cdot b_{11} + a_{32} \cdot b_{21} + a_{33} \cdot b_{31} = c_{31}$$

Matrizen-Multiplikation

Beispiel

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} c_{11} & \boxed{c_{12}} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

$$a_{11} \cdot b_{12} + a_{12} \cdot b_{22} + a_{13} \cdot b_{32} = c_{12}$$

Matrizen-Multiplikation
Beispiel

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{pmatrix} = \begin{pmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{pmatrix}$$

Diskussion

Berechnen Sie das Produkt aus den beiden Matrizen!

$$\begin{matrix} 1 & 16 & 11 \\ 5 & 16 & 7 \\ 2 & -4 & 8 \end{matrix}$$

$$\begin{pmatrix} 2 & 3 & 2 \\ 0 & 4 & 1 \\ 0 & -1 & 3 \end{pmatrix} \cdot \begin{pmatrix} -2 & 2 & 1 \\ 1 & 4 & 1 \\ 1 & 0 & 3 \end{pmatrix}$$

Matrizen-Multiplikation
Matrix mit Vektor

- **Matrix-Vektor-Multiplikation** ist wichtiger Spezialfall
- Motivation: Matrix beschreibt Transformation, Vektor einen Eckpunkt

$$\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ \vdots \\ v_n \end{pmatrix} = \begin{pmatrix} a_{11} \cdot v_1 + \ldots + a_{1n} \cdot v_n \\ \vdots \\ a_{n1} \cdot v_1 + \ldots + a_{nn} \cdot v_n \end{pmatrix}$$

Diskussion

Berechnen Sie das Produkt aus Matrix und Vektor!

$$\underset{3\times 3}{\begin{pmatrix} -2 & 2 & 1 \\ 1 & 4 & 1 \\ 1 & 0 & 3 \end{pmatrix}} \cdot \underset{3\times 1}{\begin{pmatrix} 1 \\ 1 \\ 3 \end{pmatrix}} = \underset{3\times 1}{\begin{pmatrix} 3 \\ 8 \\ 10 \end{pmatrix}}$$

Interaktive Computergrafik

Kapitel Polygonale Modellierung

Boundary Representation

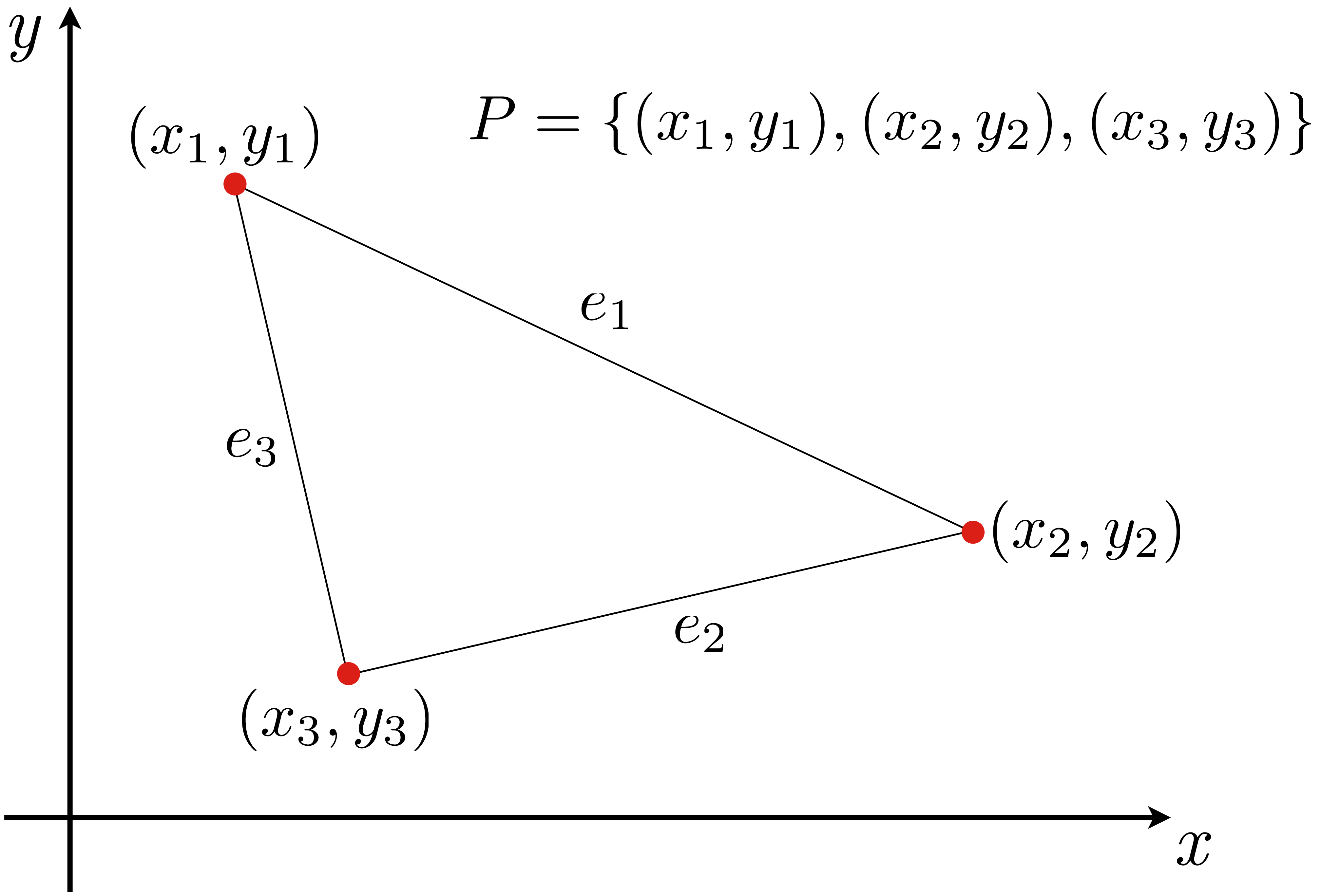
B-Rep

- **Begrenzungsflächenmodell** (kurz **B-Rep**) (engl. ***Boundary Representation***) ist Darstellungsform eines Flächen- oder Volumenmodells, in der Objekte durch begrenzenden Oberflächen beschrieben werden
- Beispiel: Polygonale Repräsentation

Polygon

- **Polygon** (**Vieleck**) ist zusammenhängende Fläche, die aus verbunden Linienzug (mit mindestens drei verschiedenen Punkten in Ebene) besteht
- Beispiele: Dreiecke, Vierecke, Sechsecke …

Polygon
Beispiel



Polygon
Darstellung

- **3D-Polygon** ist *Tupel*

$$P := \{V_1, V_2, ..., V_n\}, V_i \in \mathbb{R}^3, 1 \leq i \leq n$$

aus n **Eckpunkten** (Knoten/Vertices) und n Kanten (engl. ***Edges***):

$$E_i := \overline{V_i V_{i+1}}, i = 1, ..., n-1$$

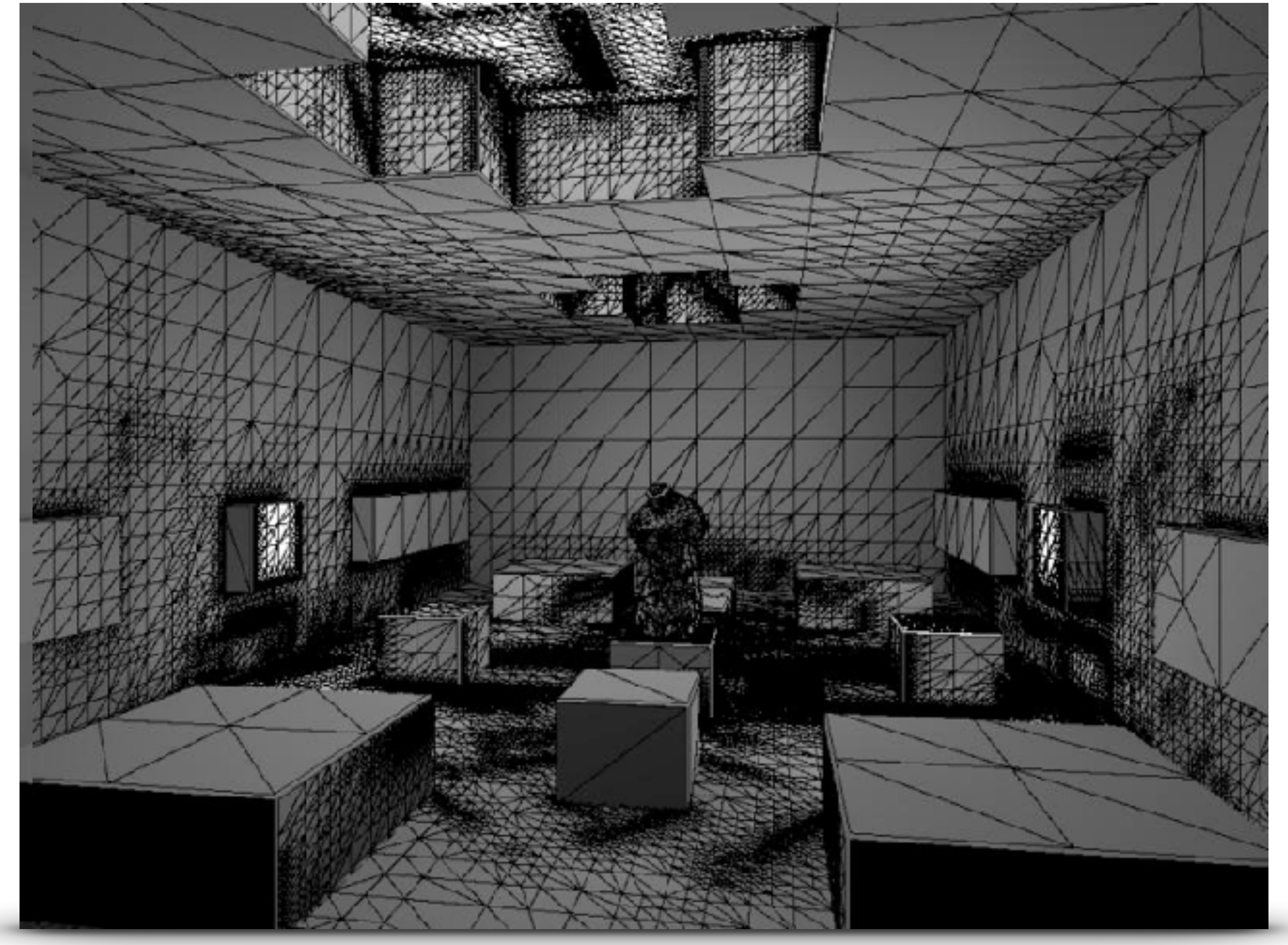
und

$$E_n := \overline{V_n V_1}$$

Polygonnetz

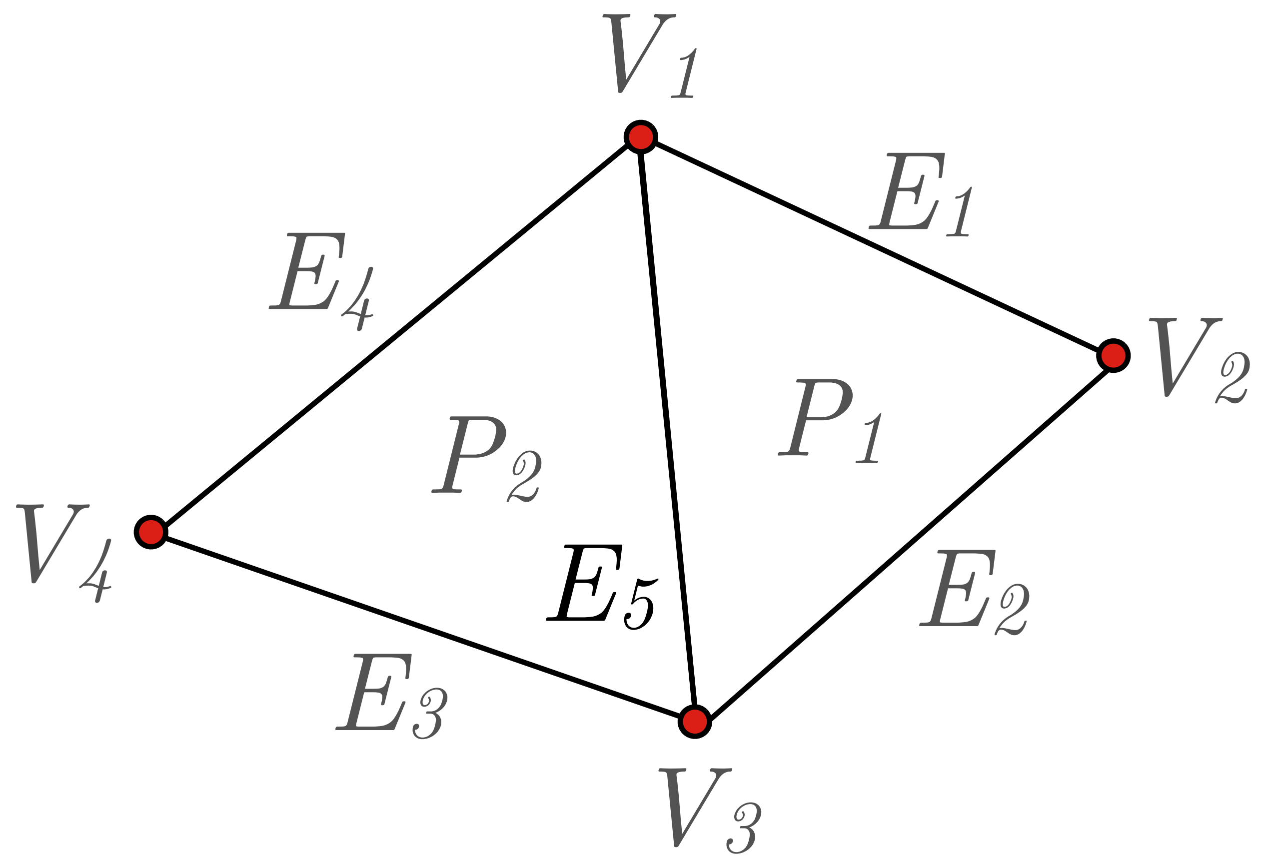
- **Polygonnetz** ist durch Menge von untereinander mit Kanten verbundenen Punkten gegeben
- Beispiele: am häufigsten werden Dreiecks- und Vierecksnetze verwendet

Polygonnetz
Beispiel



Polygonnetz

- **Polygonnetz** $P_M = \{V, E, P\}$ (engl. ***Polygon Mesh***) besteht aus Menge von Vertices V, Kanten E und Polygonen P



Polygonnetz
Datenstruktur

- Verschiedene Repräsentationen für verschiedene Anforderungen
- Unterschiede bei Repräsentationen:
  - Speicherplatzbedarf
  - Effizienz einzelner Operationen
- Speicherplatzbedarf steigt, je expliziter Kanten-Knoten-Polygon-Verbindungen modelliert werden

Polygonnetz
Explizite Darstellung

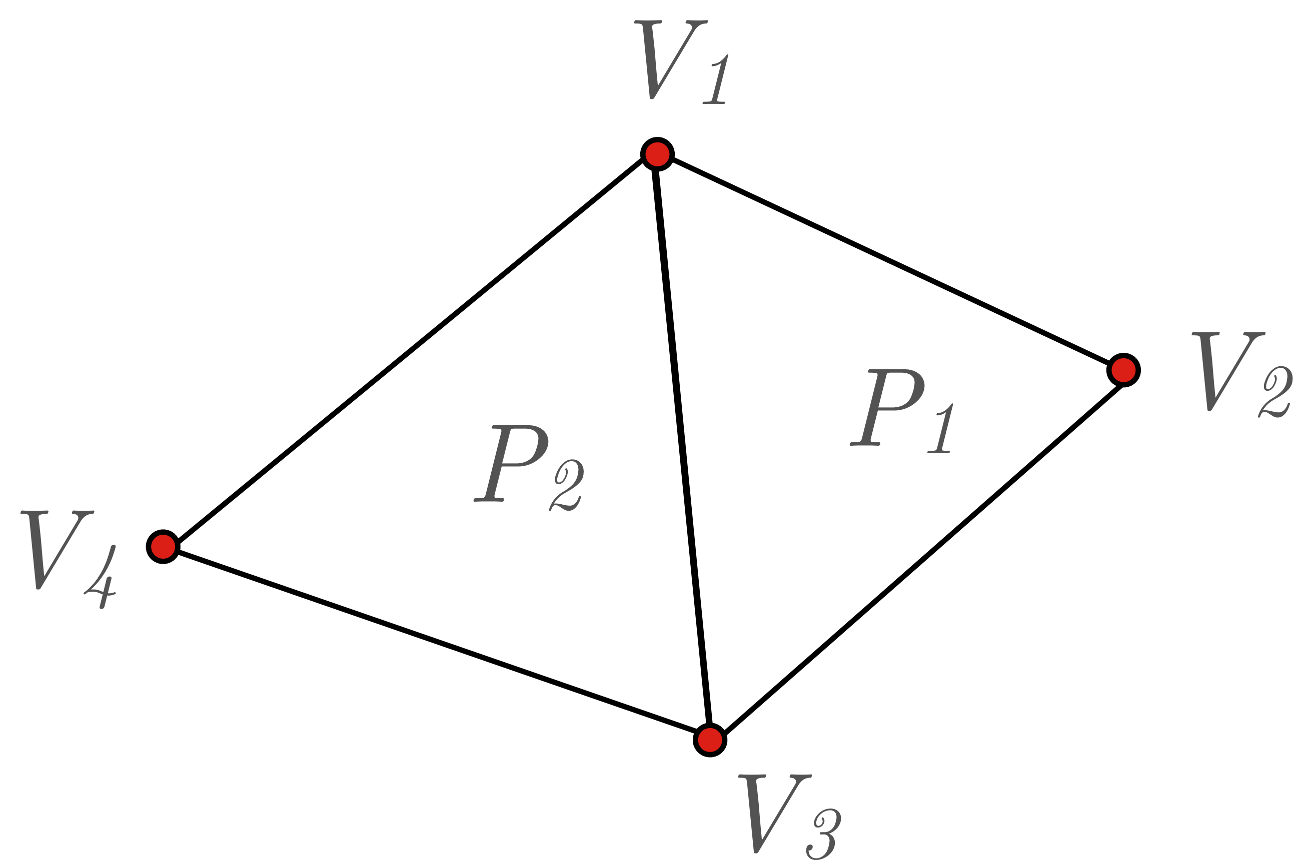
- Darstellung eines Polygons durch Liste der Knotenkoordinaten

$$P = \{V_i\} = \{(x_1, y_1, z_1), ..., (x_n, y_n, z_n)\}$$

  - Vertices sind geordnet
  - Kanten existieren zwischen aufeinanderfolgenden Vertices und zwischen letztem und ersten Vertex
  - bekannt als **Raw Data Format (RDF)**

Polygonnetz

Beispiel: Explizite Darstellung



$$PM = \{V_1, V_2, V_3, V_1, V_3, V_4\}$$

Explizite Darstellung

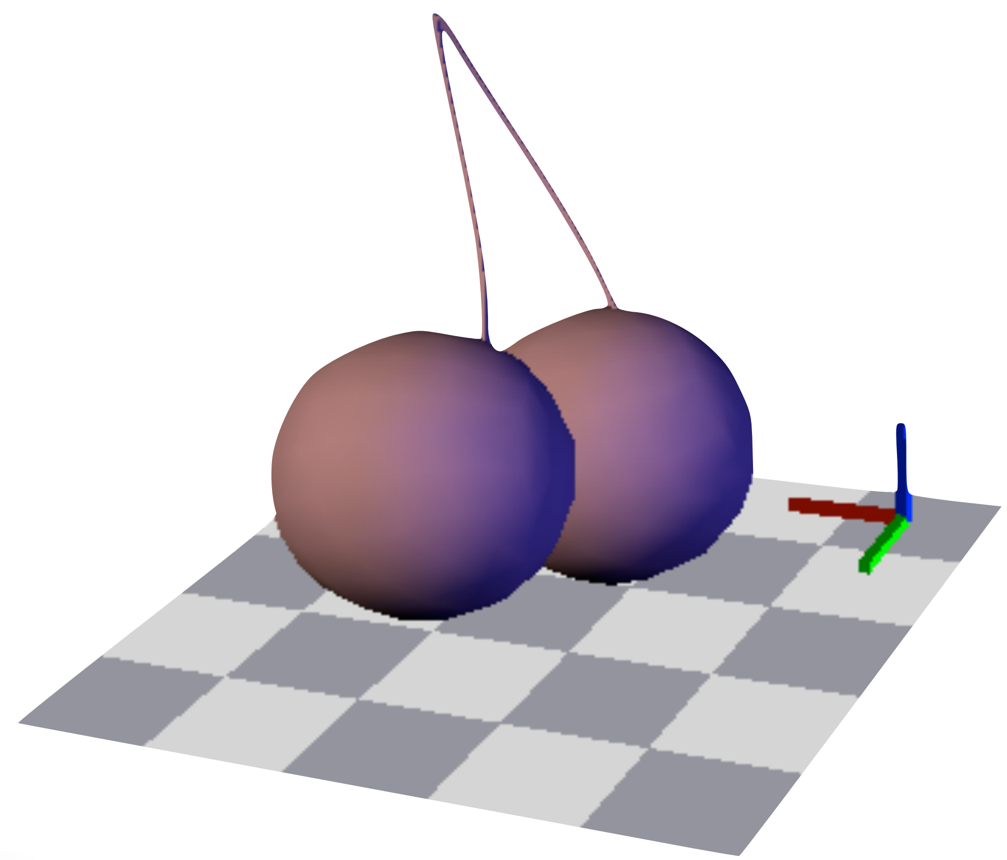
Eigenschaften

- Vorteile:
  - einfache sequentielle Speicherung
  - einfaches Format zum Datenaustausch
- Nachteile:
  - redundante Speicherung von Vertices und Kanten
  - keine Darstellung gemeinsam genutzter Vertices und Kanten

Explizite Darstellung

Beispiel

```
var positionBuffer = [
 -0.866196 1.03911 -4.48913
 -0.228063 1.35006 -4.45752
 -0.086119 1.03911 -3.83457
  -0.85765 2.73376 -2.85353
 -0.228063 1.35006 -4.45752
  -0.99826 3.04191 -3.47817
 -0.866196 1.03911 -4.48913
             .....
```



3D-Objekt mit

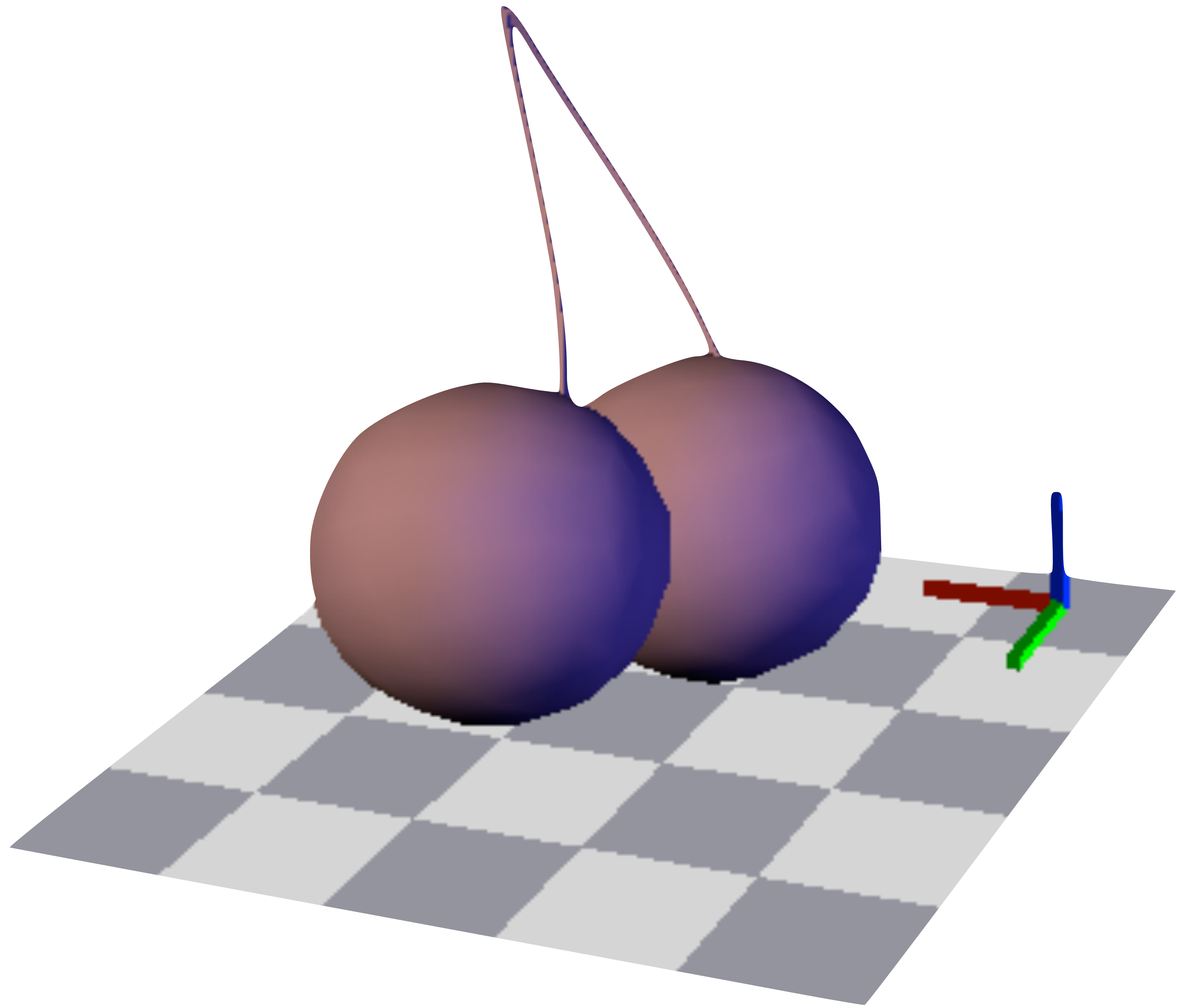
420 Vertices und **824 Polygonen** (Dreiecke)

$3 \cdot 3 \cdot 4 \cdot 824 = 29.664$ byte

Gruppenarbeit

3 Vertex pro Polygon, 3 Koordinaten $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ pro Vertex, 4 Byte pro Koordinate

Wieviel Speicher benötigt die explizite Darstellung bei 4 Byte Koordinaten?



3D-Objekt mit

420 Vertices und **824 Polygonen** (Dreiecke)

Explizite Darstellung

Beispiel: Speicherbedarf

- Objekt mit 824 Polygonen (Dreiecke):

  824 Polygone mit jeweils 3 Vertices, d.h.

  - $824 \cdot 3$ Vertices und
  - $824 \cdot 3 \cdot 3$ Koordinaten

  bei 4 Byte pro Koordinate:

  - $824 \cdot 3 \cdot 3 \cdot 4 = 29.664$ Bytes

Polygonnetz
Indizierte Vertexliste

- Konstruktion einer **Vertexliste** (engl. ***Vertex List***) für Polygonnetz:

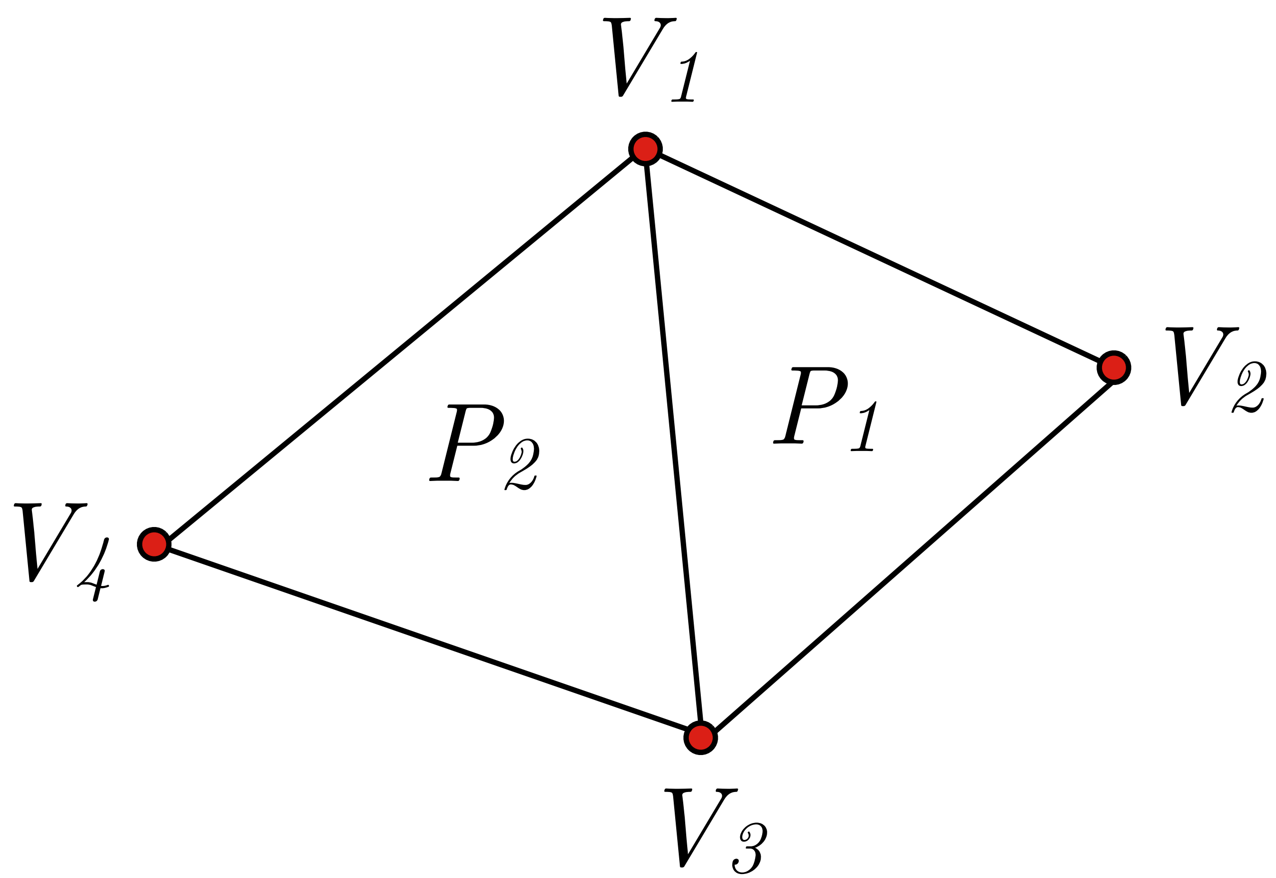
$$V = \{V_1, ...V_n\} = \{(x_1, y_1, z_1), ..., (x_n, y_n, z_n)\}$$

- Darstellung eines Polygons durch eine Liste von Vertex-Indizes:

$$PM = \{P_i\}, P_i = \{i_1, ..., i_m\}, 1 \leq m \leq n$$

Indizierte Vertexliste

Beispiel



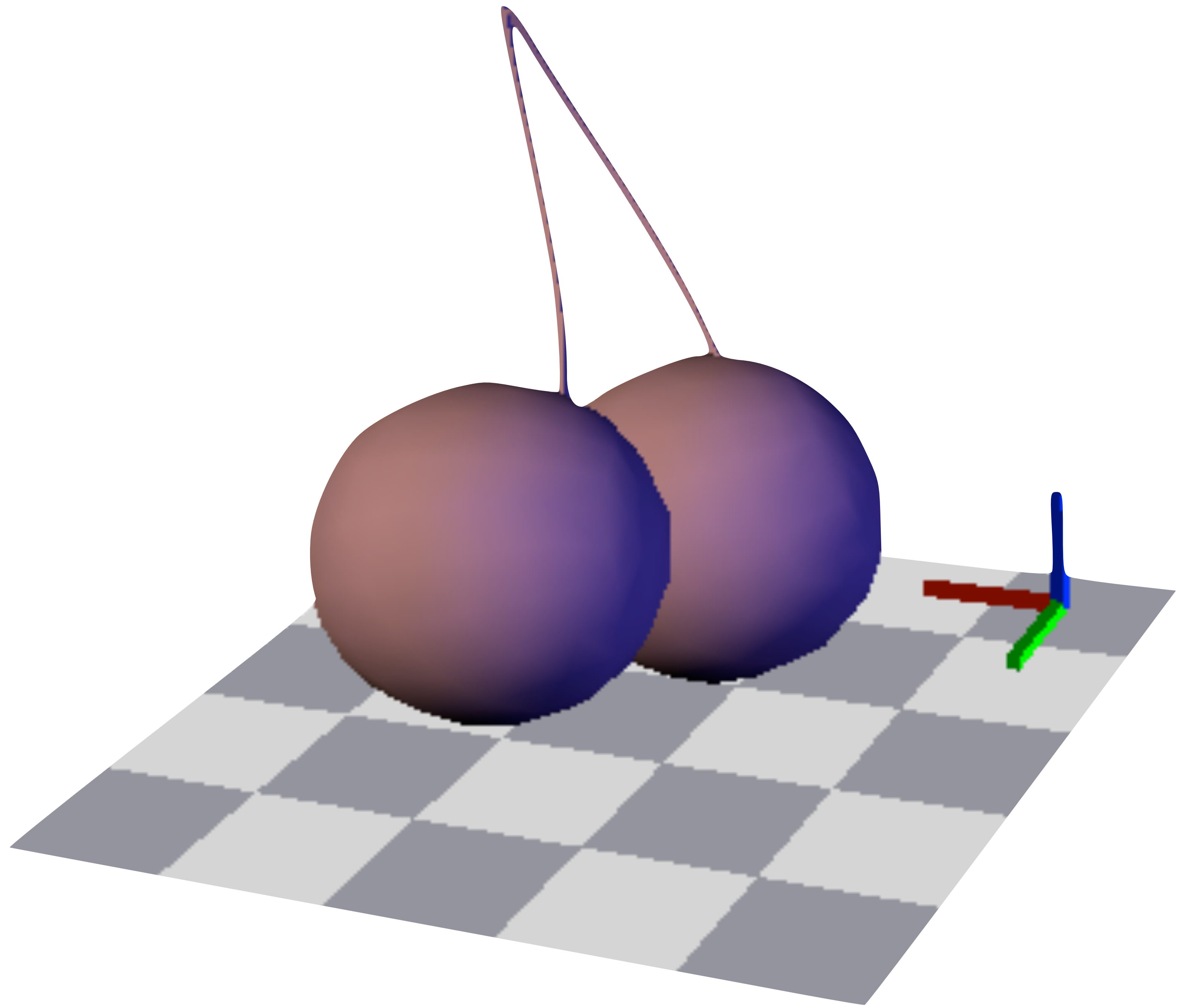
$$V = \{V_1, V_2, V_3, V_4\}$$

$$P_1 = \{1, 2, 3\}$$

$$P_2 = \{1, 3, 4\}$$

Gruppenarbeit

Wieviel Speicher benötigt die indizierte Vertexliste?



3D-Objekt mit
420 Vertices und **824 Polygonen** (Dreiecke)

Indizierte Vertexliste

Beispiel: Speicherbedarf

- Objekt mit 420 Vertices und 824 Polygonen (Dreiecke)
  - Explizite Speicherung von 420 Vertices: $420 \cdot 3 \cdot 4$ Bytes = 5.040 Bytes
  - 824 Polygone mit jeweils 3 Knoten: $824 \cdot 3 \cdot 2$ Bytes = 4.944 Bytes
  - ‣ insgesamt 9.984 Bytes

Vertices · 12 + Polygone · 6

Indizierte Vertexliste

Eigenschaften

- Vorteile:
  - geringerer Speicherplatzbedarf (wg. gemeinsam genutzter Vertexliste)
  - leichte Änderung der Koordinaten einzelner Vertices
  - Unterstützung durch Grafiksysteme, z.B. Three.JS, WebGL …

Indizierte Vertexliste

Eigenschaften

- Nachteile:
  - gemeinsame Kanten können nur schwer gefunden werden
  - gemeinsame Polygonkanten werden mehrfach gezeichnet

Polygonnetze
Kantenliste

- Konstruktion einer **Vertexliste** und **Kantenliste** (engl. ***Edge List***)

$$V = \{V_1, ..., V_n\}, E = \{E_1, ..., E_n\}$$

- Darstellung eines Polygons durch Liste von Kantenindizes

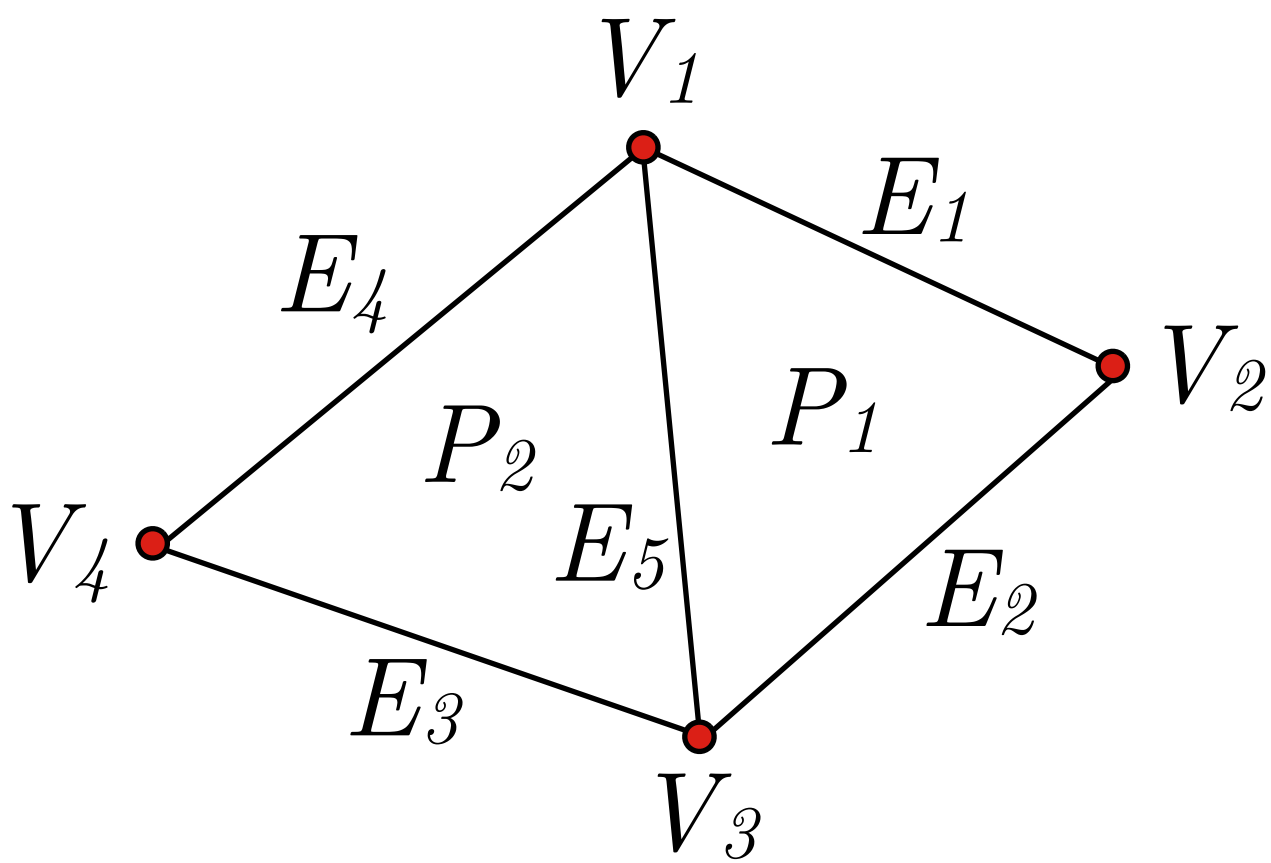
- Eckpunkte der Kante + Polygone zur Kante

$$E_i = \{\boxed{i_V, j_V}, \boxed{P_k, ...}\}$$

$$PM = \{P_k\}, P_k = \{u_E, ..., v_E\}$$

Kantenliste

Beispiel



$$V = \{V_1, V_2, V_3, V_4\}$$
$$E_1 = \{1_V, 2_V, P_1\}$$
$$E_2 = \{2_V, 3_V, P_1\}$$
$$E_3 = \{3_V, 4_V, P_2\}$$
$$E_4 = \{4_V, 1_V, P_2\}$$
$$E_5 = \{1_V, 3_V, P_1, P_2\}$$
$$P_1 = \{1_E, 2_E, 5_E\}$$
$$P_2 = \{5_E, 3_E, 4_E\}$$

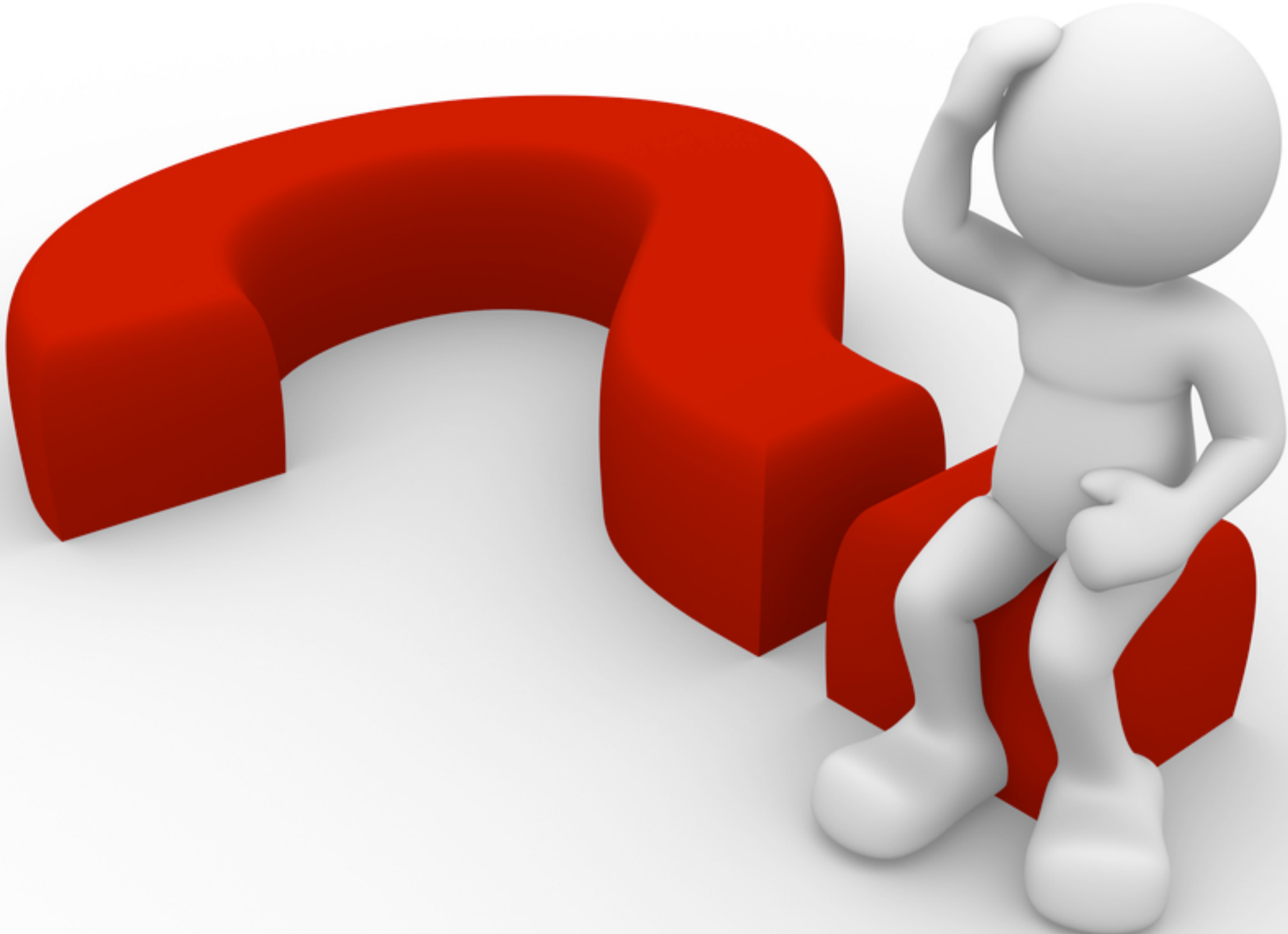
Kantenliste
Eigenschaften

- Vorteile:
  - leichte Änderbarkeit der Vertex-Koordinaten
  - einmaliges Zeichnen der Polygonnetzkanten
  - einfaches Auffinden gemeinsamer Vertices und Kanten

Kantenliste

Eigenschaften

- Nachteile:
  - größerer Speicherplatzbedarf wg. der Kantenliste
  - keine Unterstützung für Low-Level-Grafiksystem (z.B. WebGL, OpenGL ...)



Interaktive Computergrafik

Kapitel Polygonale Modellierung

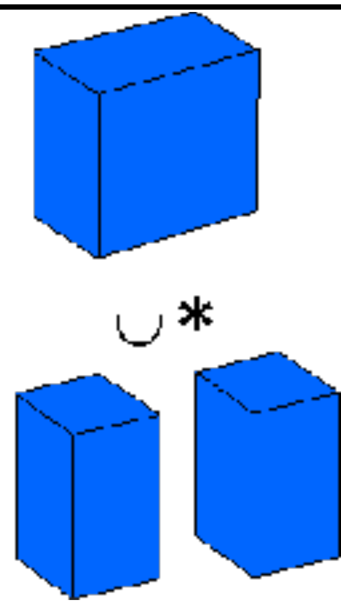
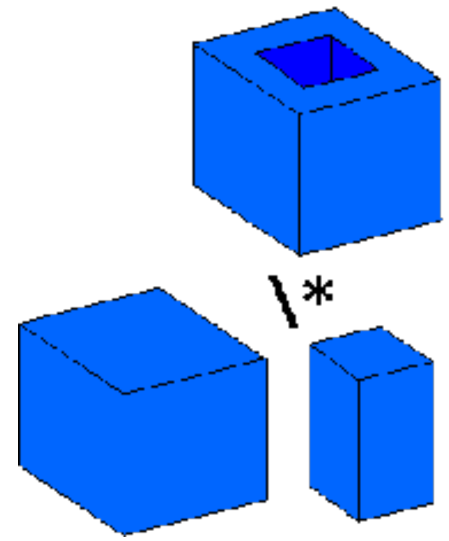
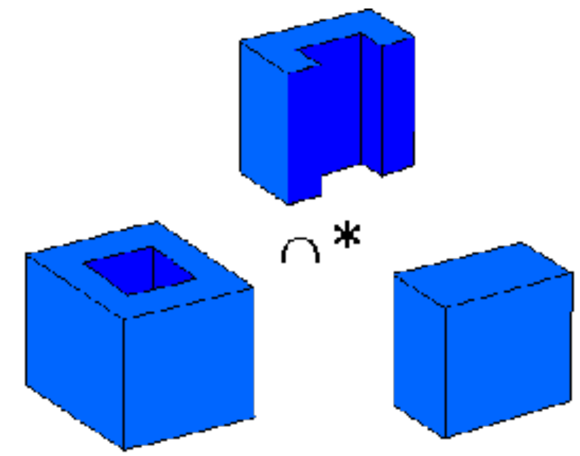
Alternative Modellrepräsentationen

CSG

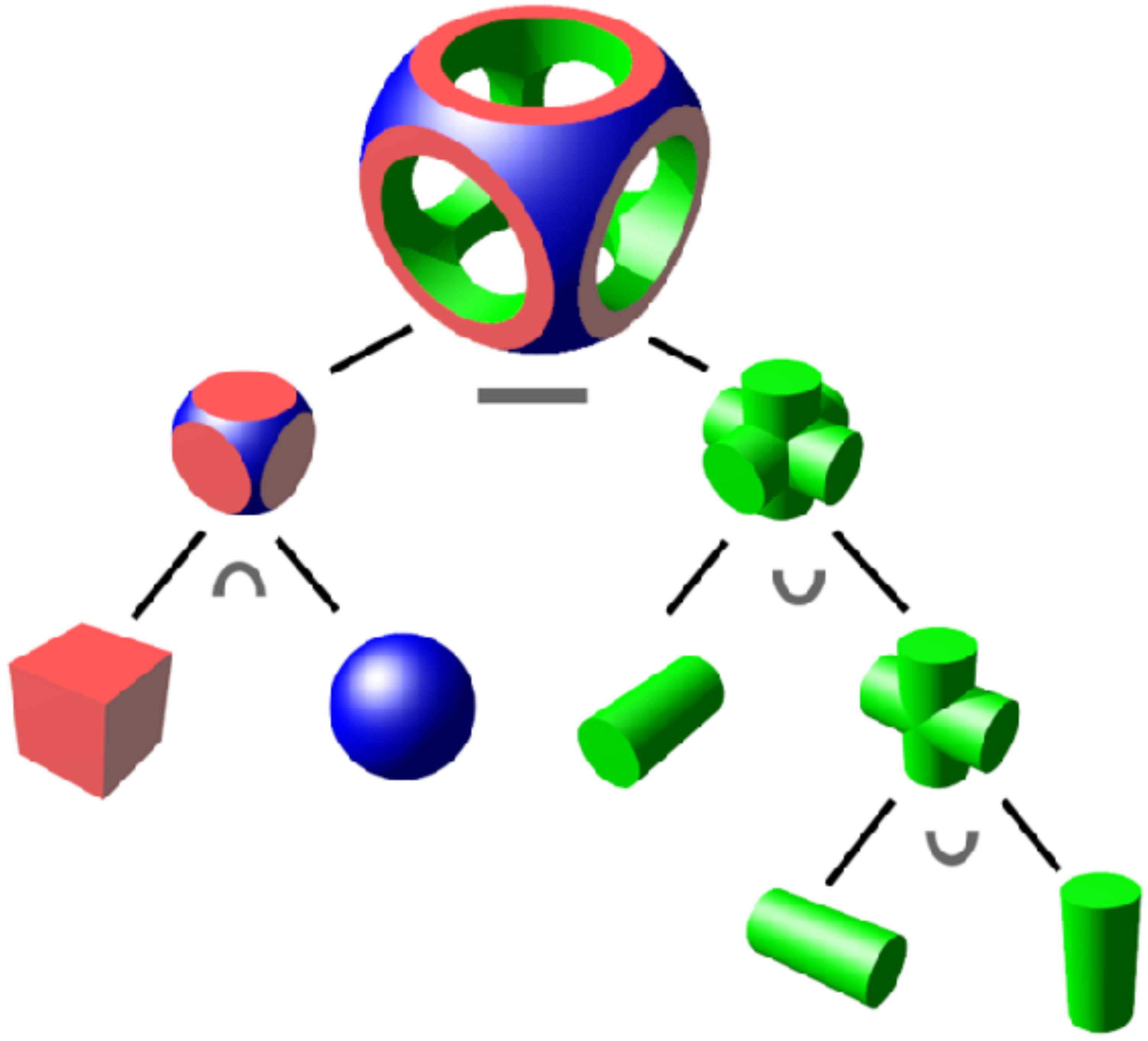
- **Constructive Solid Geometry (CSG)** ermöglicht Erzeugung komplexer Oberflächen und Körper durch Kombination von Objekten mittels **boolescher Operationen**
- verknüpfte Objekte wirken häufig wie komplexe Körper und werden auch **boolsche Körper** genannt

CSG

Operationen

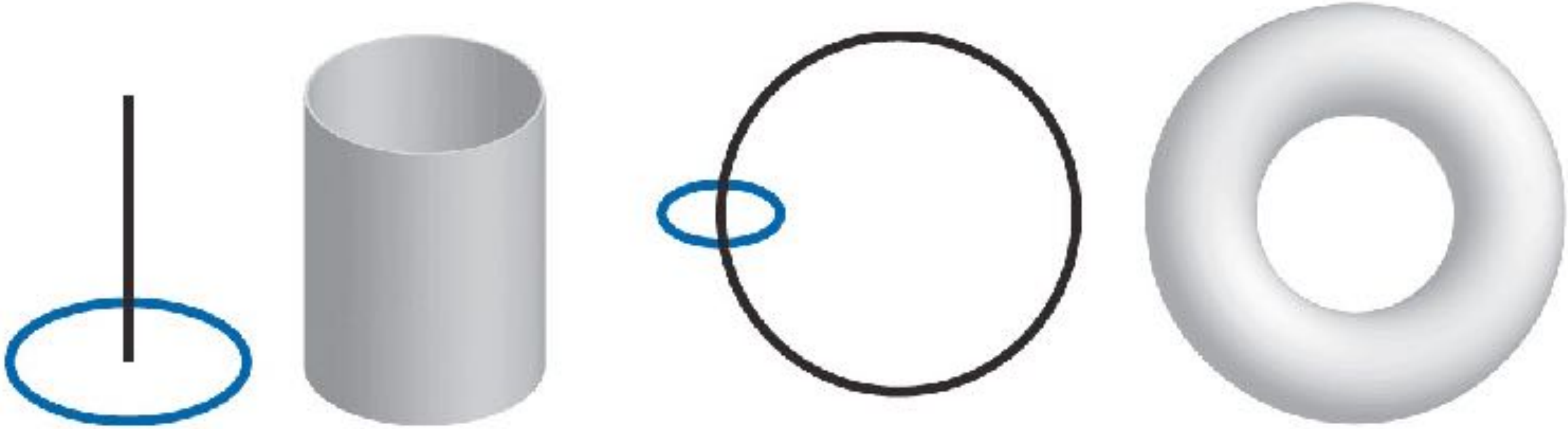
		
Vereinigung	**Differenz**	**Schnitt**
Zwei Objekte werden zu einem verschmolzen.	Teile des zweiten Objekts werden aus dem ersten herausgeschnitten.	Es wird der Teil extrahiert, den beide Objekte gemeinsam haben.

CSG
Beispiel



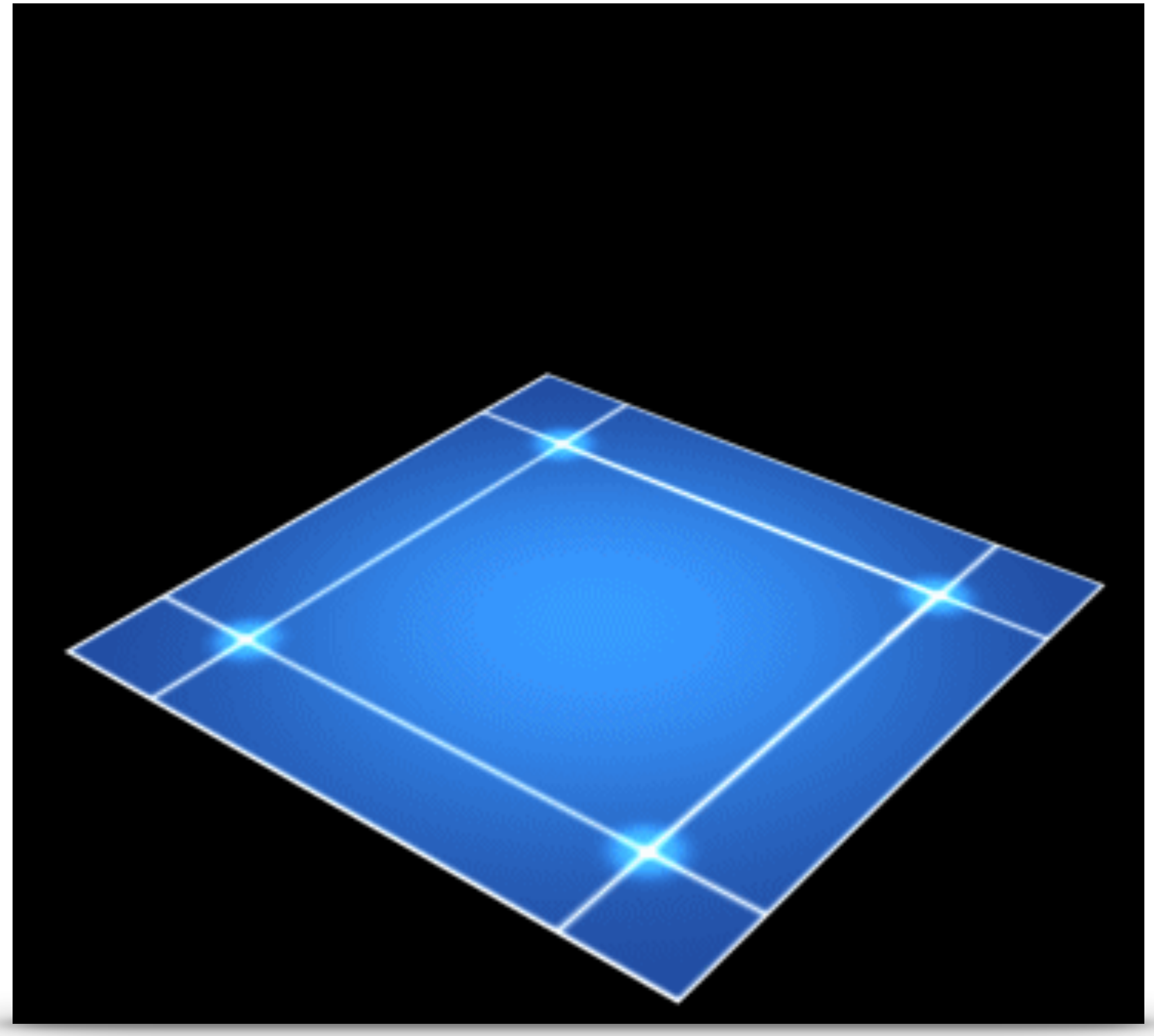
Extrusionskörper

- **Extrusionskörper** bezeichnet Dimensions-erhöhung eines Elementes durch Ver-schiebung (entlang eines Pfads) im Raum
- Beispiele



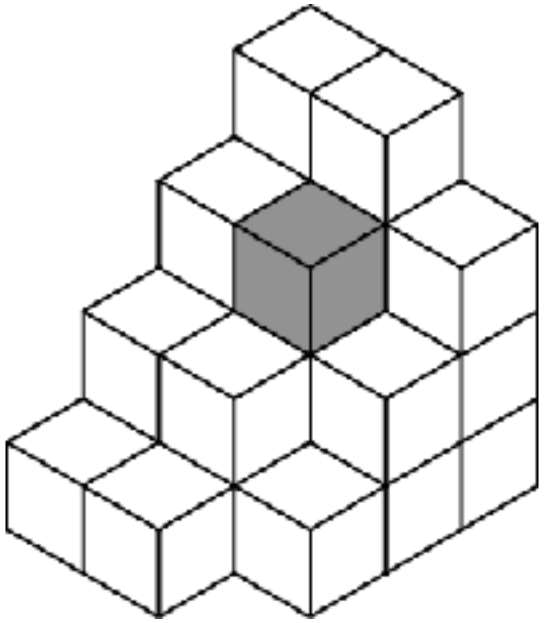
Extrusionskörper

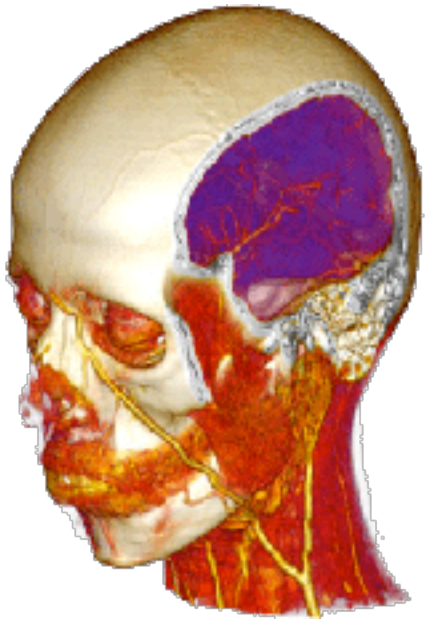
Beispiel



Voxel

- **Voxel** (Kunstwort aus engl. ***Volume*** und ***Pixel***) unterteilen Raum in dreidimensionale Raumeinheiten
- für jedes Voxel werden optische Eigenschaften beschrieben





Point-based Graphics

- Punktbasierte 3D-Grafik (engl. ***Point-based Graphics***) hat als grundlegendes Element nicht Polygon oder Voxel, sondern Punkt

Photo Tourism

Exploring photo collections in 3D

Noah Snavely   Steven M. Seitz   Richard Szeliski

University of Washington   *Microsoft Research*

SIGGRAPH 2006