LUDWIG-
MAXIMILIANS-
UNIVERSITÄT
MÜNCHEN

LFE Medieninformatik • Prof. Dr. Ing. Axel Hoppe

Computergrafik I

Datenstrukturen für die 3D-Computergrafik

Rückschau

- Ausgabegeräte für die Computergrafik
- Erinnerung: alle relevanten Ausgabegeräte arbeiten rasterorientiert

Fahrplan für heute

- Vom Modell zum Bild: Vorüberlegungen für Geometrierepräsentation
- Rendering-Pipeline
- Transformationen in 2D
- Transformationen in 3D

LFE Medieninformatik • Prof. Dr. Ing. Axel Hoppe

Vorüberlegungen

Vom Modell zum Bild

3D-Bildberechnung

- Gegeben:
  - möglichst genaues 3D-Modell in geeigneten Koordinaten,
  - Beschreibung für Oberflächen und Lichtverhältnisse sowie
  - eine Beschreibung der Sicht in die Szene
- Gesucht:
  - gerastertes (Pixel-)Bild, das das gegebene Modell realitätsnah darstellt.

3D-Szene

- für eine Szenenbeschreibung werden benötigt:
  1. Objektrepräsentation
  2. Beschreibungsmöglichkeit für Oberflächenanmutungen
  3. Modell für Interaktion der Oberflächen mit Lichtquellen
  4. Modell für ein Sichtbeschreibung in die 3D-Szene
  5. Möglichkeit, Objekte anzuordnen

Objekt-Repräsentation

1. Mächtigkeit (sinnvolle Menge von Objekten modellierbar)
2. Eindeutigkeit (Was wird repräsentiert?)
3. Genauigkeit (Approximation des Objektes)
4. Effizienz (Darstellung, Speicherplatz)
5. Abgeschlossenheit (Transformationen, Boolesche Mengenoperationen)
6. weitere Forderungen:
   - unmöglich, eine ungültige Repräsentation herzustellen
   - einfach, eine gültige Repräsentation herzustellen

Objekt-Repräsentation

- Volumenmodelle
  - Primitive Instancing
  - Constructive Solid Geometry
  - Sweep-Körper
  - Raumunterteilung
- Oberflächenmodelle
  - Polygone
  - Freiformflächen
  - Quadriken

Objekt-Repräsentation – Volumina

Kennzeichnung	Anwendungsgebiete
• Unterscheidung zwischen innen/außen und Grenzflächen eines Modells häufig notwendig • Modellierung der inneren Struktur eines Objektes • Modellierung von Objekten als Volumina	• CAD/CAM • physikalische Simulation • medizinische Visualisierung, etwa Operationsplanung

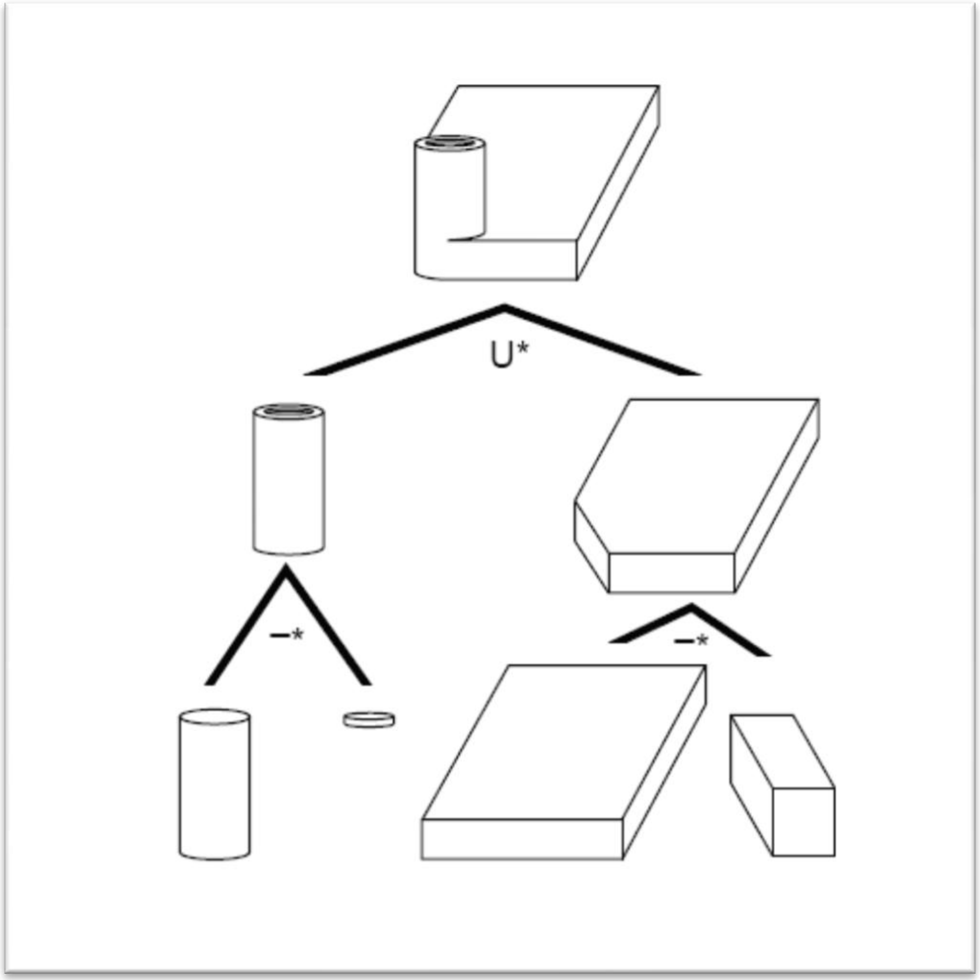
Primitive Instancing

- Modelliersystem definiert Menge von primitiven 3D-Volumina
  - anwendungsbezogen
  - Parametrisierung dieser Primitive
  - Komposition über Transformationen
- Eigenschaften
  - Zusammenbau zu „Baugruppen“
  - keine Kombination der Objektvolumina zu Gesamtvolumina

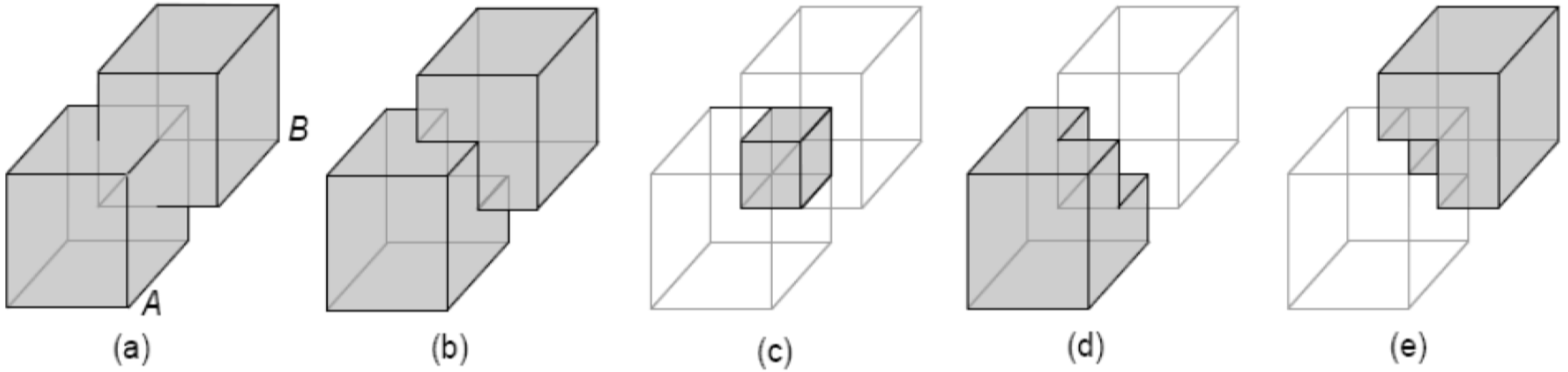
Constructive Solid Geometry (CSG)

- Herstellung komplexer Volumina über boolesche Operationen
- Primitive: geometrische Grundobjekte (Pyramide, Quader, Kugel, etc.)
- Ein komplexes Objekt wird über einen Baum beschrieben:
  - Knoten beschreiben die Verknüpfung der Nachfolger
  - Teilbäume repräsentieren Teilobjekte
  - pro Knoten gibt es zwei Nachfolger
- Volumina der Teilkörper werden verbunden zu Gesamtvolumen

Constructive Solid Geometry (CSG)



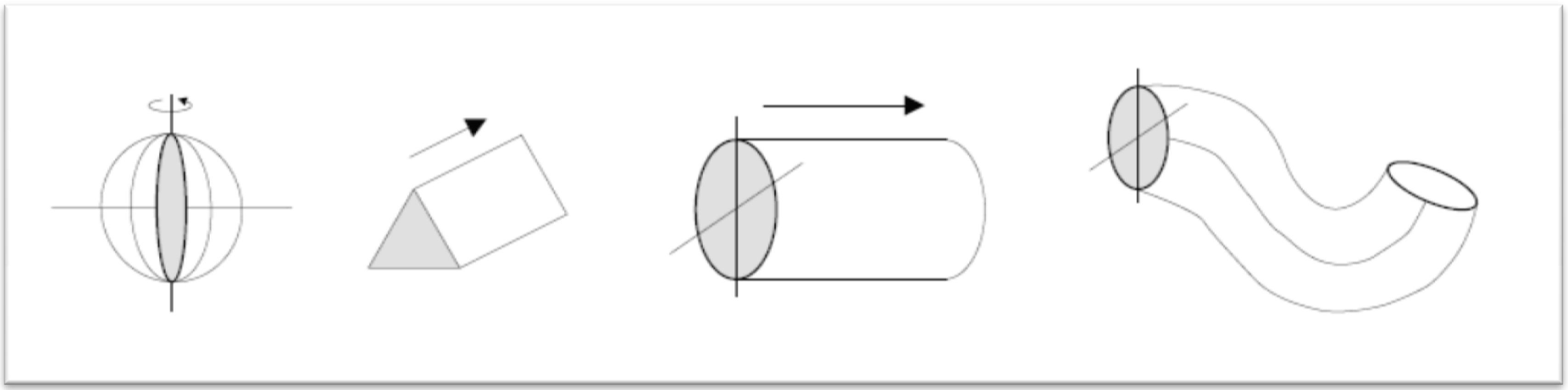
Boolsche Operationen



(b): $A \cup B$, (c): $A \cap B$, (d): $A - B$, (e): $B - A$

Sweep-Körper

- Objekt wird durch eine Fläche (Grundfläche) und eine Transformationsvorschrift beschrieben (=generalisierter Zylinder).
- problematisch:
    - Selbstüberschneidungen der Kurve (Was ist dann das Volumen?)
    - Kurve liegt in der Flächenebene (Leeres Volumen?)



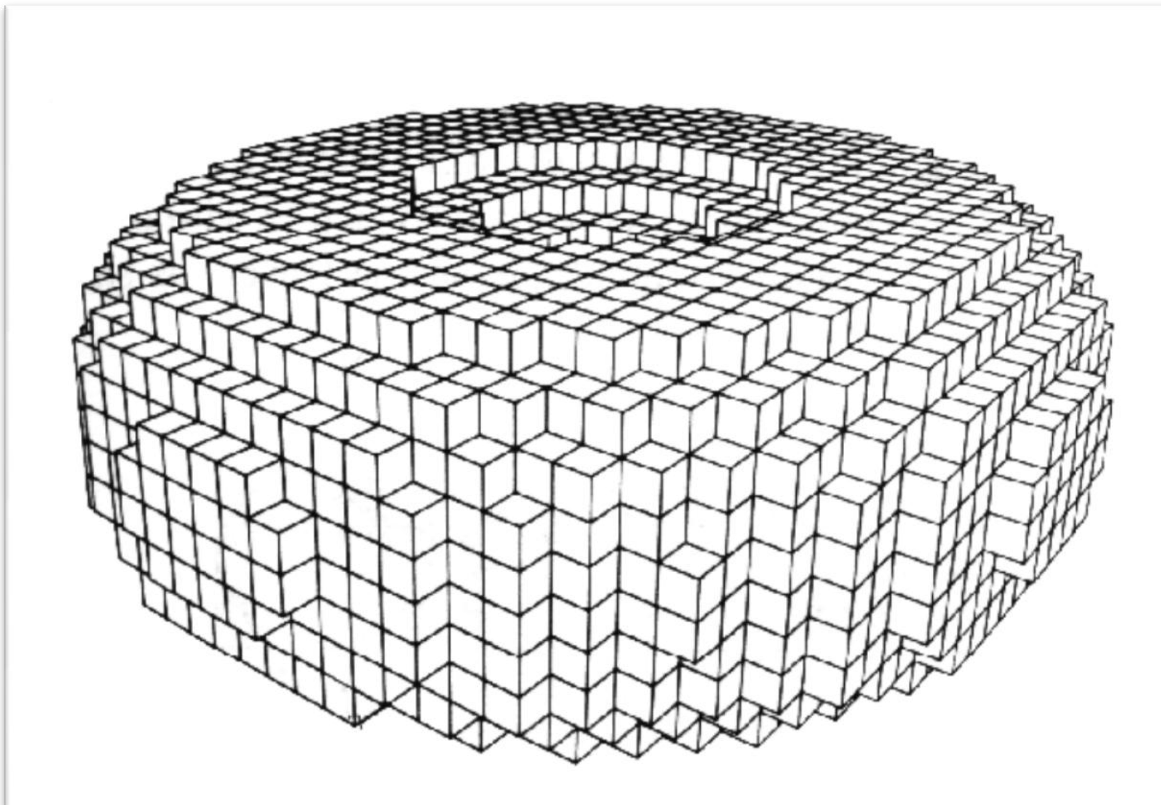
Raumunterteilung

- Unterteilung des Gesamtvolumens in Teilvolumina nach verschiedenen Strategien:
  - Zelldekomposition – Unterteilung in eine Menge unterschiedlicher Primitive
  - räumliche Aufzählung – Unterteilung in gleiche Voxel
  - Octrees – Unterteilung durch achsenparallele Ebenen
  - BSP-Trees – Unterteilung durch arbiträre Ebenen

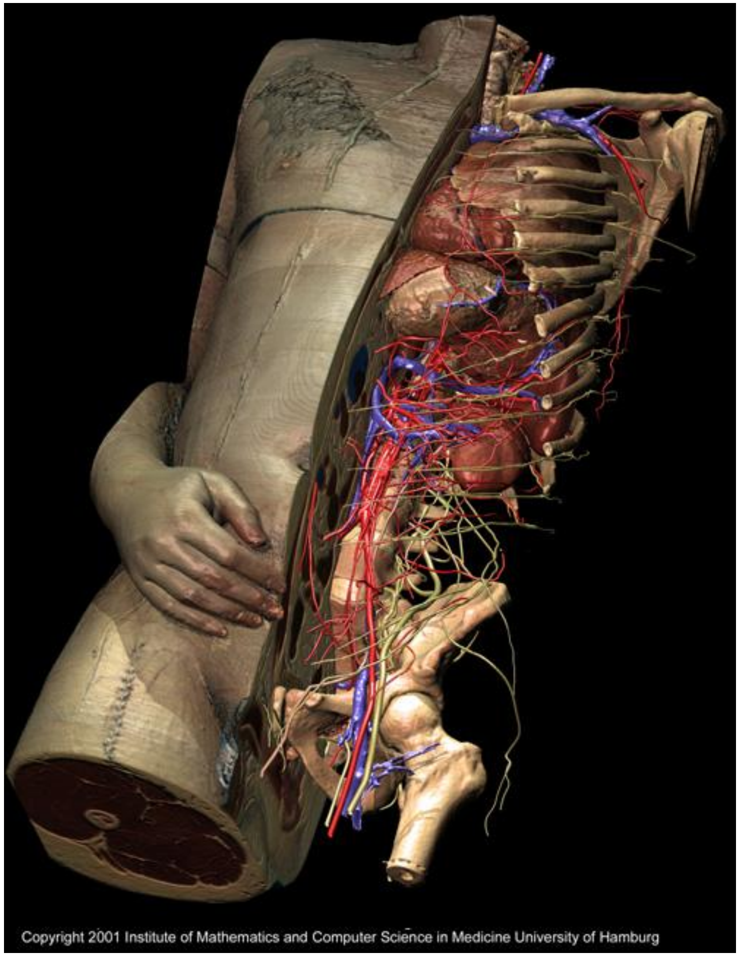
Voxel

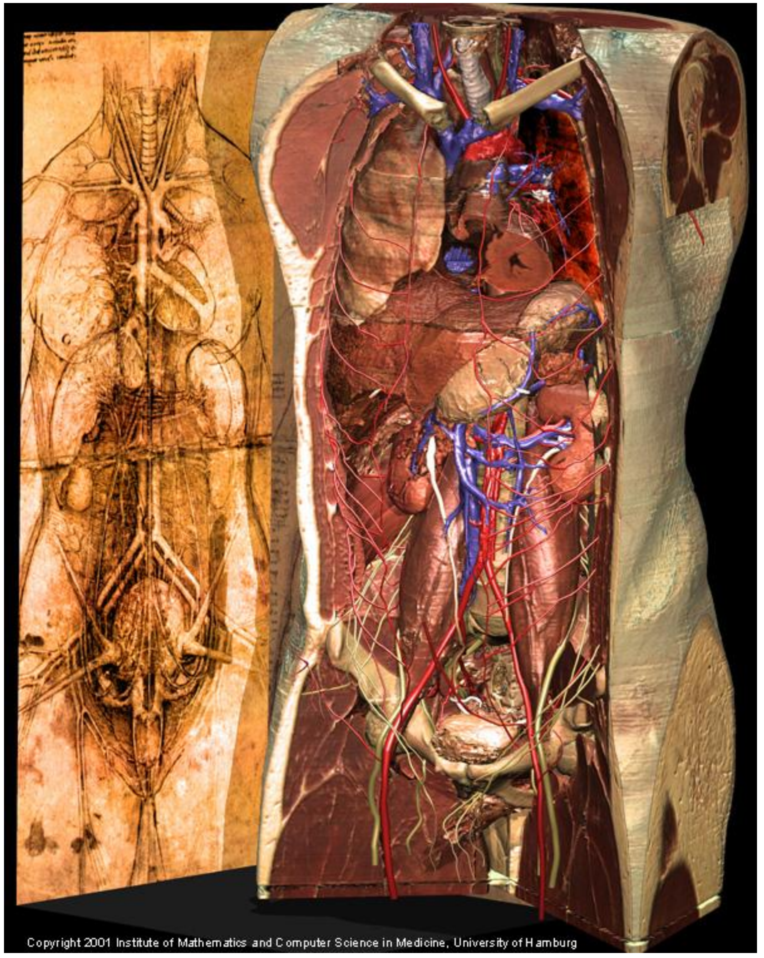
- Zelldekomposition in identische Zellen, die in einem regelmäßigen Gitter angeordnet sind
- *voxel* = *volume element*, dreidimensionale Pixel
- An- oder Abwesenheit des entsprechenden Voxels
- Möglichkeit der Speicherung weiterer Parameter
- Approximation führt zu Aliasing-Artefakten
- hoher Speicherplatzbedarf (n^3)

Voxel – Beispiel

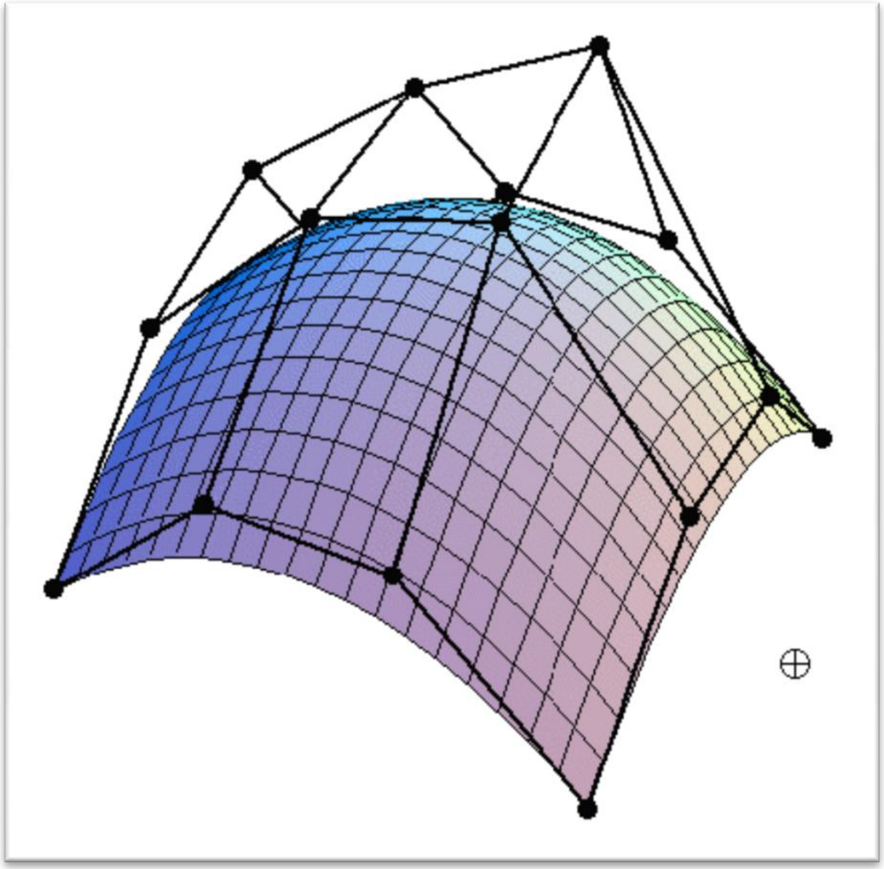


Voxel – Anwendungen





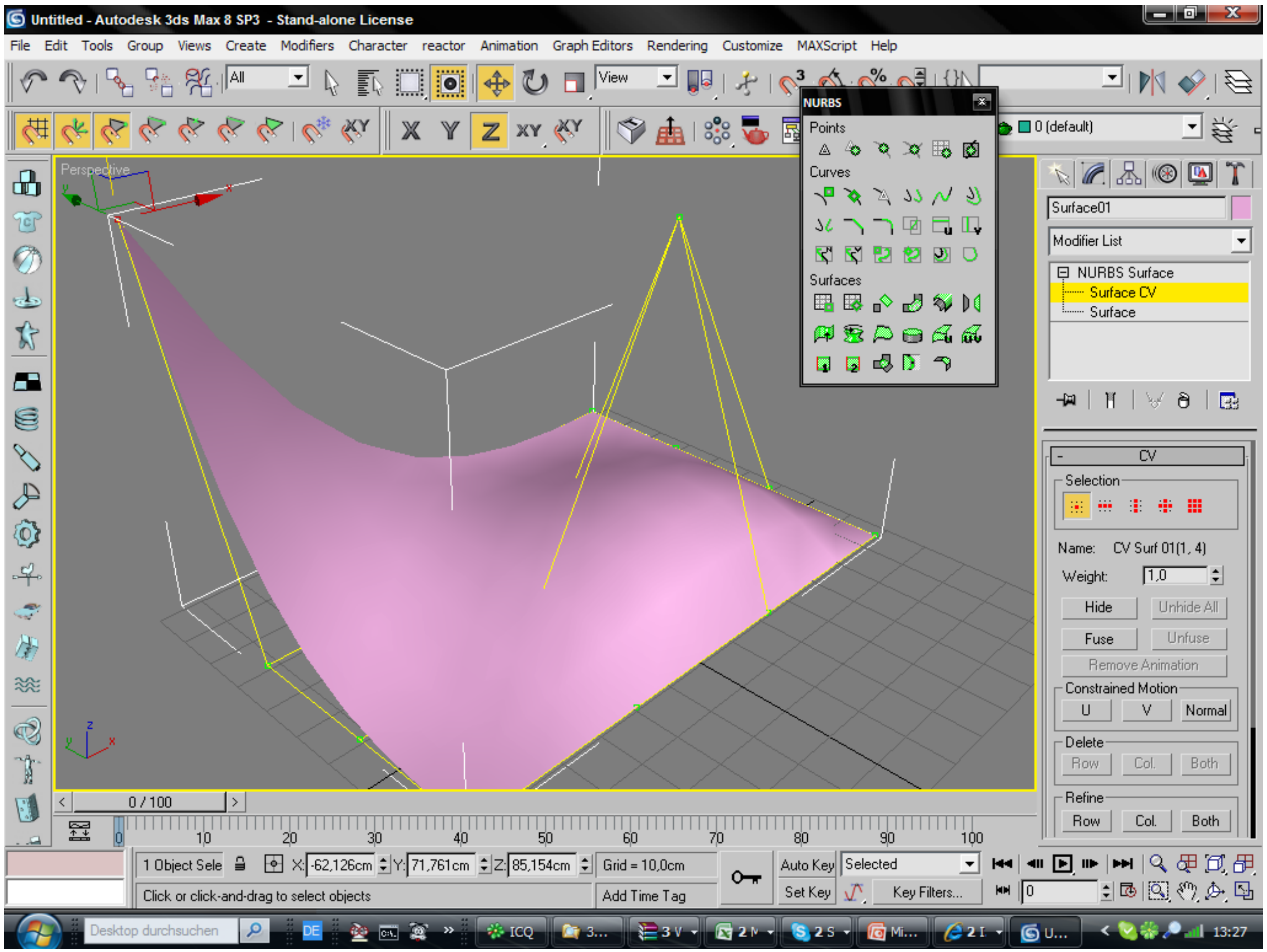
Freiformflächen



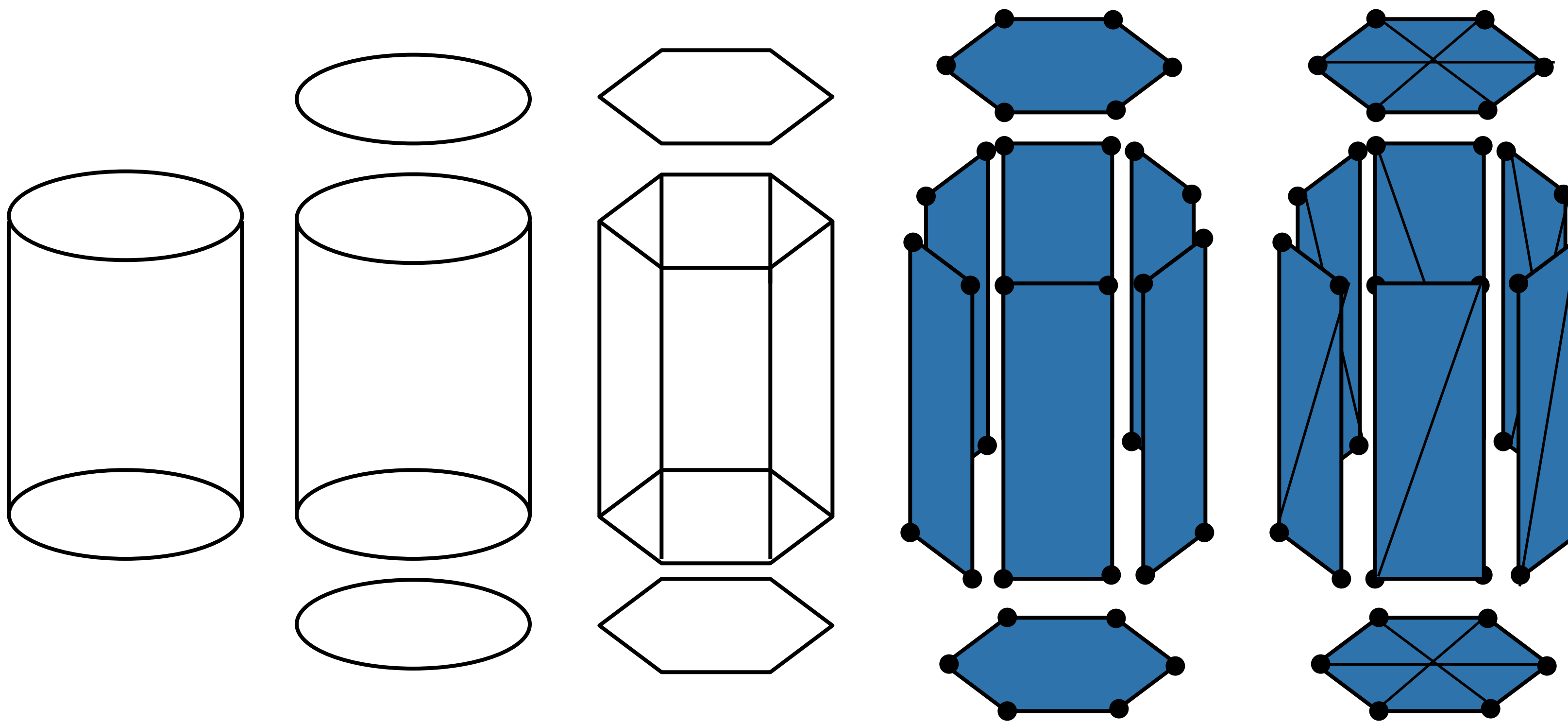
Freiformflächen – Eigenschaften

- sind invariant für projektive Transformationen
- gemeinsame mathematische Darstellung für sowohl analytische Standardformen (z. B. Kegelschnitte) als auch Freiformflächen
- reduzierter den Speicheraufwand für geometrische Objekte
- können durch numerisch stabile und präzise Algorithmen verhältnismäßig schnell ausgewertet werden
- sind Verallgemeinerungen von nicht-rationalen B-Splines und nicht-rationalen und rationalen Bézier-Kurven und -Flächen
- ermöglichen organische Anmutungen von Oberflächen

Nurbs aus Sicht des Anwenders

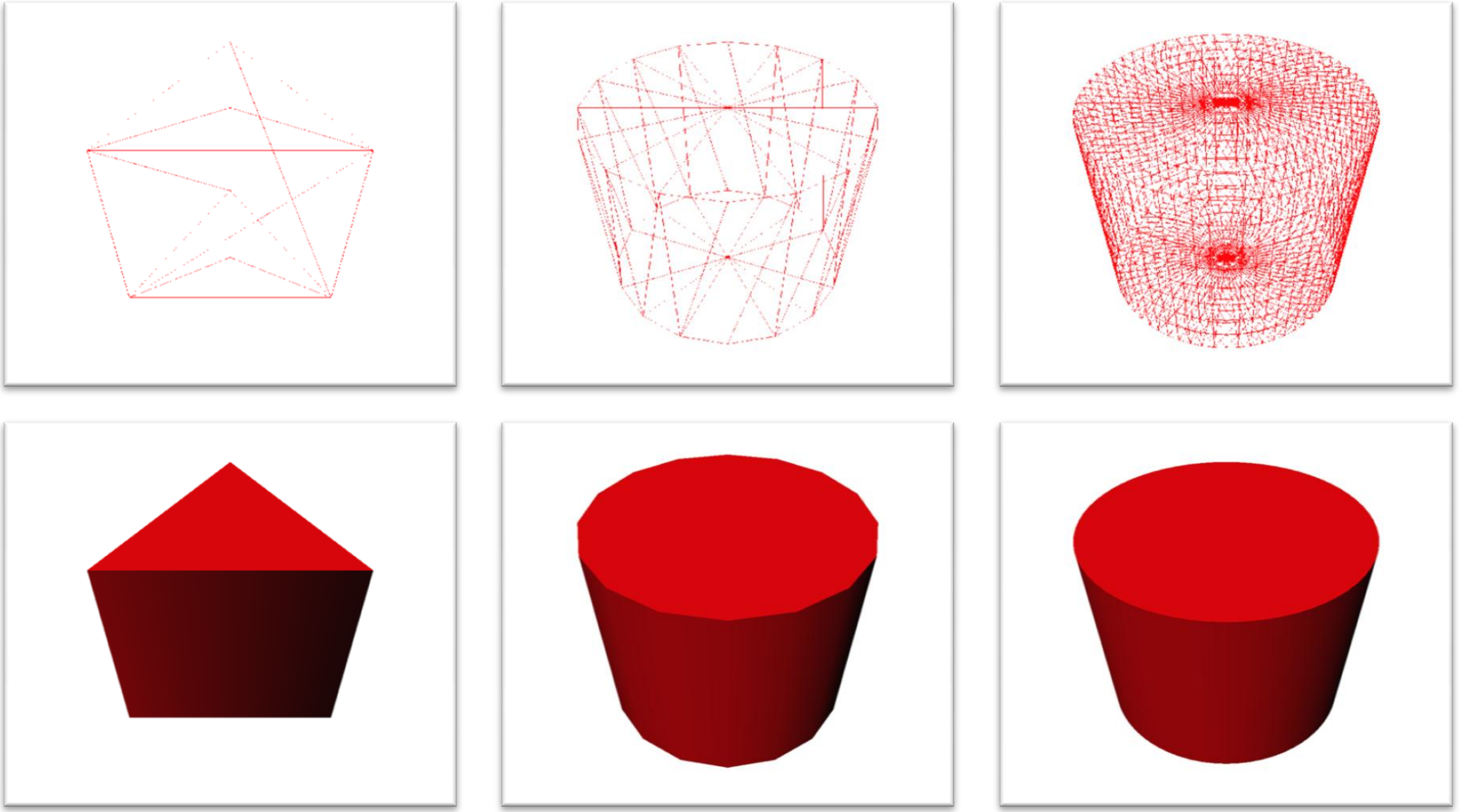


Polygone



ideales Objekt | Oberflächen | polygonale Oberfläche | Eckpunkte, Kanten für Polygone | Facetten (*Faces*)

Objekt-Repräsentation, Illustration



vom Modell zum Bild

Gestalterische Überlegungen

Ziel

- möglichst realistische, schöne Bilder
- gleiche Komplexität wie die komplexen Interaktionen zwischen Licht und Oberflächen in der Realität
- „Fotorealismus“
- Was ist der Ursprung der folgenden Bilder? Begründen Sie Ihre Aussage!

Real oder synthetisch?

Real oder virtuell?

Real oder virtuell?

Real oder virtuell?

Real oder virtuell?

Objekt-Repräsentation

- Vorüberlegung: Ausreichend ist Betrachtung der Oberflächen
- daher Konzentration auf polygonale Oberflächenmodelle
- Analytische und implizite Darstellungen lassen sich zudem in explizite Darstellungen umwandeln (Polygonalisieren).

Vorteile polygonaler Modelle

- geometrischer Modelle
- Relativ einfache Algorithmen, daher Implementierung in Hardware möglich
- Wichtig: Polygone müssen in einer Ebene liegen (z. B. um zu bestimmen, was im Inneren liegt) – das ist bei Dreiecken immer der Fall
- Daher: (interne) Triangulierung

Zusammenfassung Polygone

Vorteile	Nachteile
• Einfache Darstellung der Objekte • Einfache und einheitliche Handhabung bei Berechnungen • Weit verbreitet – „kleinster gemeinsamer Nenner“ bei 3D-Modellen	• Polygonale Modelle approximieren die Oberfläche eines „runden“ Objektes. • Je genauer diese Approximation sein soll, um so mehr Polygone werden benötigt – speicheraufwändig

Notation von Polygonen

- Polygone sind gekennzeichnet durch
  - Eckpunkte (engl. *vertex*, Plural *vertices*)
  - Kanten (engl. *edges*)
  - Merken: Polygone werden zusammengesetzt aus Facetten (engl. *faces*)

Notation von Polygonen

- Angabe der Koordinaten der Eckpunkte als Koordinatentripel
- Verbindung der Eckpunkte zu einem Polygon wird implizit hergestellt:
  - Verbinde den i-ten mit dem $(i + 1)$-ten Punkt
  - Verbinde den letzten mit dem ersten Punkt zum Schließen des Polygons

Notation von Polygonen

- Polygone werden nach 2D abgebildet
- Für weitere Betrachtungen reicht also zunächst:
  - Betrachtung 2D-Primitiven
  - Erörtern der Aufgaben der Computergrafik im 2D-Fall

LFE Medieninformatik • Prof. Dr. Ing. Axel Hoppe

3D-Bildberechnung

Rendering-Pipeline

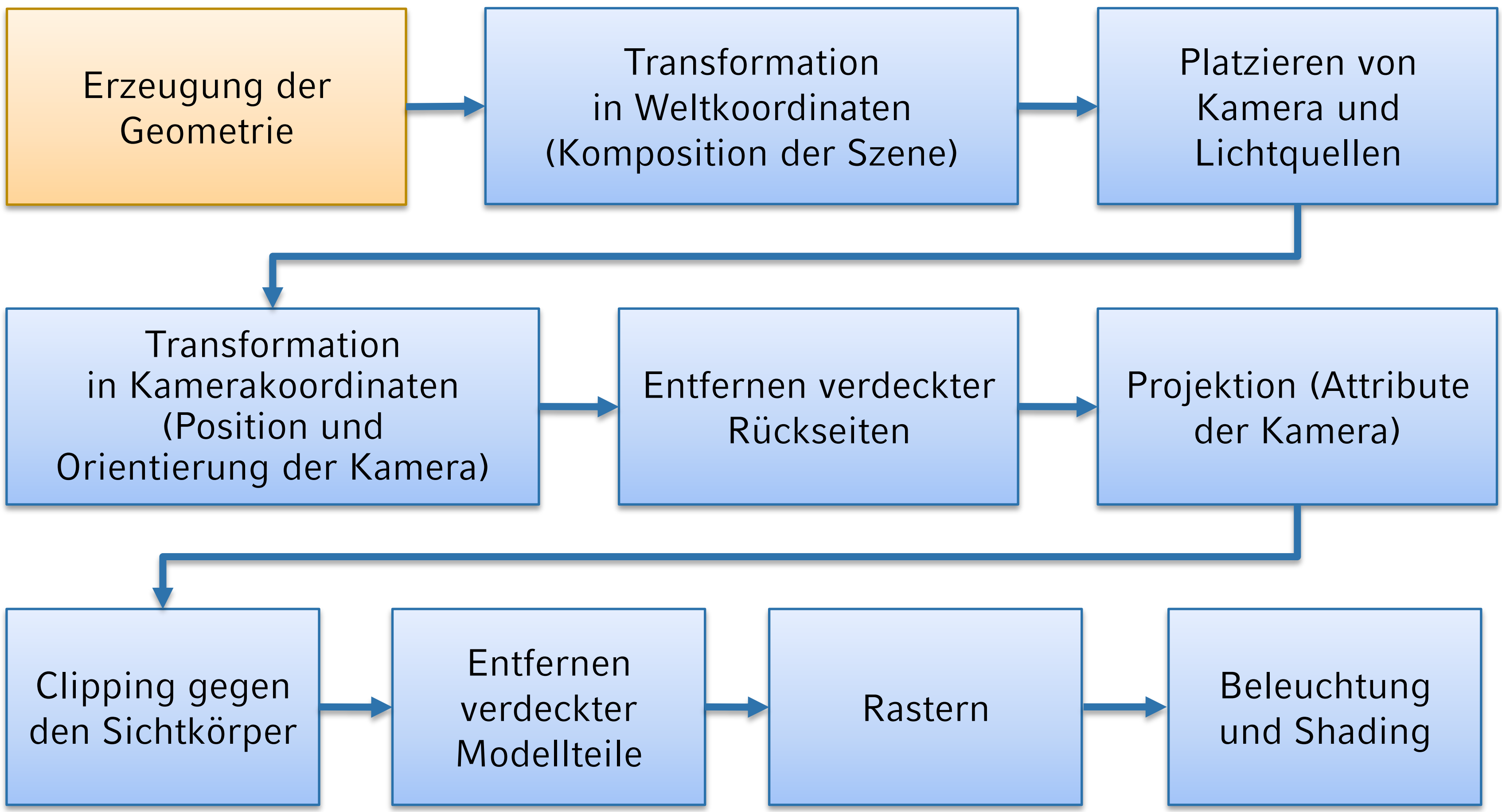
3D-Bildberechnung

- Ziel: Integration von einzelnen Techniken in ein System zur Bilderzeugung aus einem 3D-Modell
- Große Anteile:
  - Viewing,
  - Shading,
  - Rasterisierung
- Rendering von engl. *to render* = Bilderzeugung

3D-Bildberechnung

- Gegeben:
  - 3D-Modell in geeigneten Koordinaten,
  - Beschreibung für Oberflächen und Lichtverhältnisse sowie
  - eine Beschreibung der Sicht in die Szene
- Gesucht:
  - gerastertes (Pixel-)Bild, das das gegebene Modell realitätsnah darstellt.

Rendering-Pipeline



LFE Medieninformatik • Prof. Dr. Ing. Axel Hoppe

Spezifikationen in 2D und 3D

Koordinatensysteme

Vektoren und Vektorräume

- Vektor – Definition: Ein Vektor ist ein Element eines Vektorraumes.
- Vektorraum – Definition: Eine nichtleere Menge V heißt Vektorraum über einen fest gegebenen Körper K, falls gilt
  - auf V gibt es eine Verknüpfung +, so dass V mit dieser Verknüpfung eine abelsche Gruppe ist, d. h. für alle x, y, z aus V gilt:
    - $(x + y) + z = x + (y + z)$
    - Es gibt einen Nullvektor o mit: $x + o = x = o + x$.
    - Zu jedem x aus V existiert ein $-x$ aus V mit $x + (-x) = o$.
    - $x + y = y + x$
  - es gibt eine Skalarmultiplikation, d. h. eine Abbildung *K* ´ *V* ® *V*, so dass für alle x, y aus V und alle r, s aus K gilt:
    - $1v = v$.
    - $r(v + w) = rv + rw$.
    - $(r + s)v = rv + sv$.
    - $(rs)v = r(sv)$.

Vektoren und Vektorräume

- Beispiele für Vektorräume:
  - Menge aller Polynome mit Koeffizienten aus K
  - Menge aller $n \times n$-Matrizen mit reellen Elementen
  - Menge aller geordneten Paare (x, y) (x, y reelle Zahlen)
  - Menge aller geordneten Tripel (x, y, z) (x, y, z reelle Zahlen)
  - Menge aller geordneten n-Tupel $(x_0, x_1, \dots x_n)$ (x_i reelle Zahlen)
- besonders die letzten drei Beispiele für Computergraphik interessant $\rightarrow$ reellwertige Koordinaten in 2D, 3D und nD

Vektorraum $\Re^2$

- V ist die Menge aller geordneter Paare (x, y) mit $x, y \in \Re$
- K ist der Körper der reellen Zahlen
- Nullelement in V: (0, 0)
- besondere Elemente in V: (1, 0) und (0, 1)
  - Basis des Vektorraumes $\Re^2$
  - = maximale Menge linear unabhängiger Elemente aus V
- Verknüpfung „+“ entspricht komponentenweiser Addition $(x, y) + (a, b) = (x + a, y + b)$
- Skalarmultiplikation entspricht komponentenweiser Multiplikation mit dem Skalar $\alpha(x, y) = (\alpha x, \alpha y)$

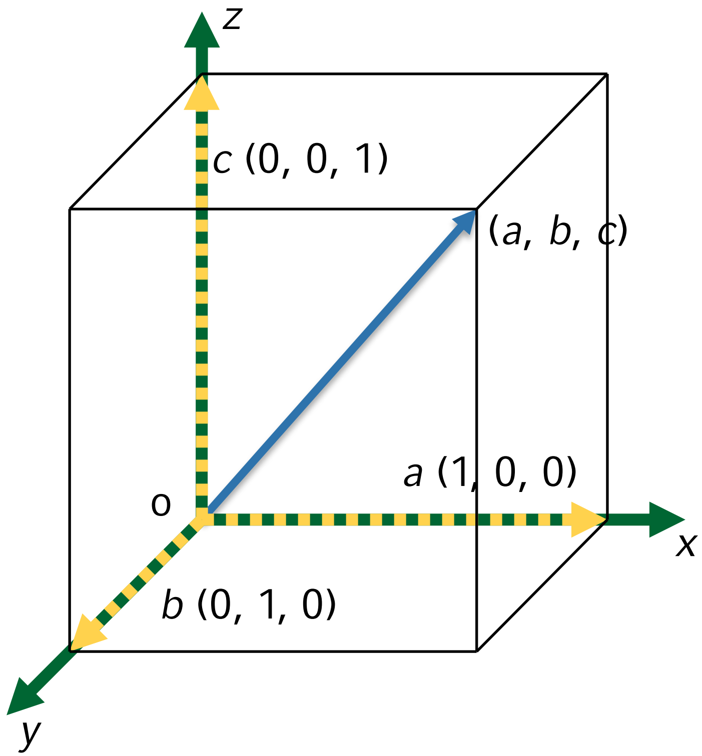
Vektorraum $\Re^3$

- V ist die Menge aller geordneter Tripel (x, y, z) mit $x, y, z \in \Re$
- K ist der Körper der reellen Zahlen
- Nullelement in V: (0, 0, 0)
- besondere Elemente in V: (1, 0, 0), (0, 1, 0) und (0, 0, 1)
  - Basis des Vektorraumes $\Re^3$
  - = maximale Menge linear unabhängiger Elemente aus V
- Verknüpfung „+" entspricht komponentenweiser Addition $(x, y, z) + (a, b, c) = (x + a, y + b, z + c)$
- Skalarmultiplikation entspricht komponentenweiser Multiplikation mit dem Skalar $\alpha(x, y, z) = (\alpha x, \alpha y, \alpha z)$

Punkte und „Vektoren“

- Geometrische Interpretation der beiden letzten Beispiele führt zu Punkten und Vektoren, wie sie aus der Geometrie bekannt sind.
- Jeder Vektor (a, b, c) kann eindeutig in eine Linearkombination der Elemente der Basis des Vektorraumes zerlegt werden:
  - $(a, b, c) = a\,(1, 0, 0) + b\,(0, 1, 0) + c\,(0, 0, 1)$
  - Die Skalare a, b und c sind die kartesischen Koordinaten des Vektors im System der Einheitsvektoren des Koordinatensystems.
  - Die kartesischen Koordinaten eines Vektors sind die Projektionen dieses Vektors auf die einzelnen Koordinatenachsen.

Koordinaten



- Interpretation als Vektor:
  - Ein Vektor selbst hat keine Position.
  - Ausgehend von einem festen Punkt (z. B. o) definiert ein Vektor einen Punkt.
  - Vektor (*a*, *b*, *c*) kann als Punkt im Raum dargestellt werden, der dem Endpunkt eines Vektors (*a*, *b*, *c*) ausgehend vom Koordinatenursprung (0, 0, 0) entspricht.

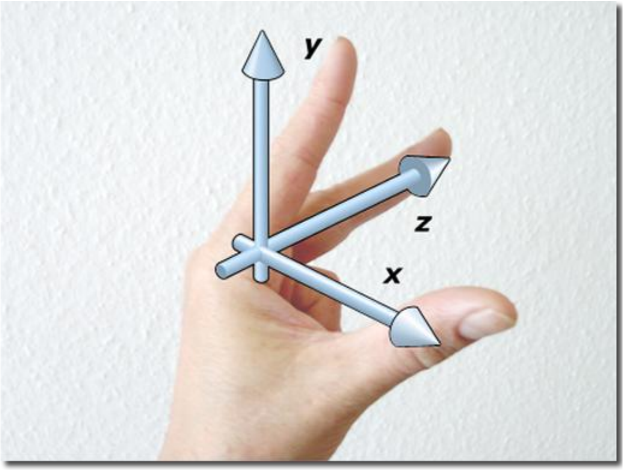
Koordinatensystem

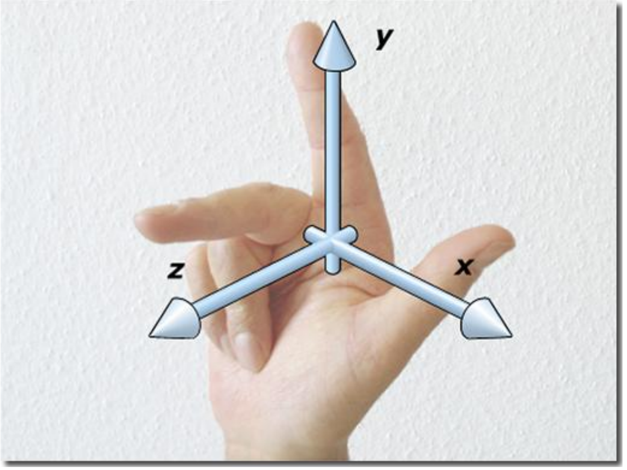
- Eine Menge (o, e_1, e_2, ..., e_n) bestehend aus einem Punkt o $\in$ A^n und der Basis (e_1, e_2, ..., e_n) von A^n heißt Koordinatensystem
- Punkt o heißt Koordinatenursprung
- Für jeden Punkt $p \in$ A^n ist v = (op) Ortsvektor von p
- Komponenten von v heißen Koordinaten bezüglich (e_1, e_2, ..., e_n) d. h. p besitzt die Koordinaten (x_1, x_2, …, x_n):

$$\left(\overrightarrow{op}\right) = v = x_1 e_1 + x_2 e_2 + \ldots + x_n e_n$$

Koordinatensysteme

- X – Richtung des Daumens
- Y – Zeigefinger
- Z – Mittelfinger
- Die beiden Koordinatensysteme sind spiegelbildlich
  und nicht durch Drehung ineinander zu überführen.





Spezifikationen in 2D und 3D

Transformationen

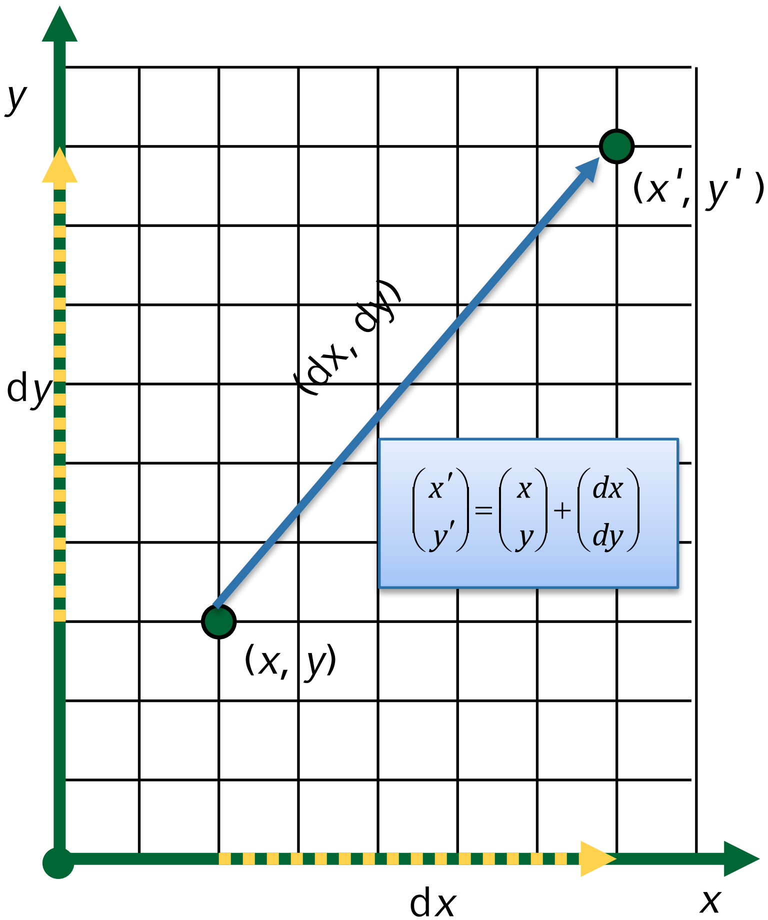
Koordinatensysteme

- Wie werden Bewegungen beschrieben?
- Wie wird die Position von Objekten nach Bewegungen berechnet?
- zunächst Betrachtung von Transformationen in 2D

Koordinatensysteme

- Bewegungen = Transformationen
  - Veränderung der Position von Punkten
  - Verschiebung = Translation
  - Größenveränderungen = Skalierung
  - Drehung = Rotation
  - Weitere affine Transformationen:
    - Spiegelung
    - Scherung

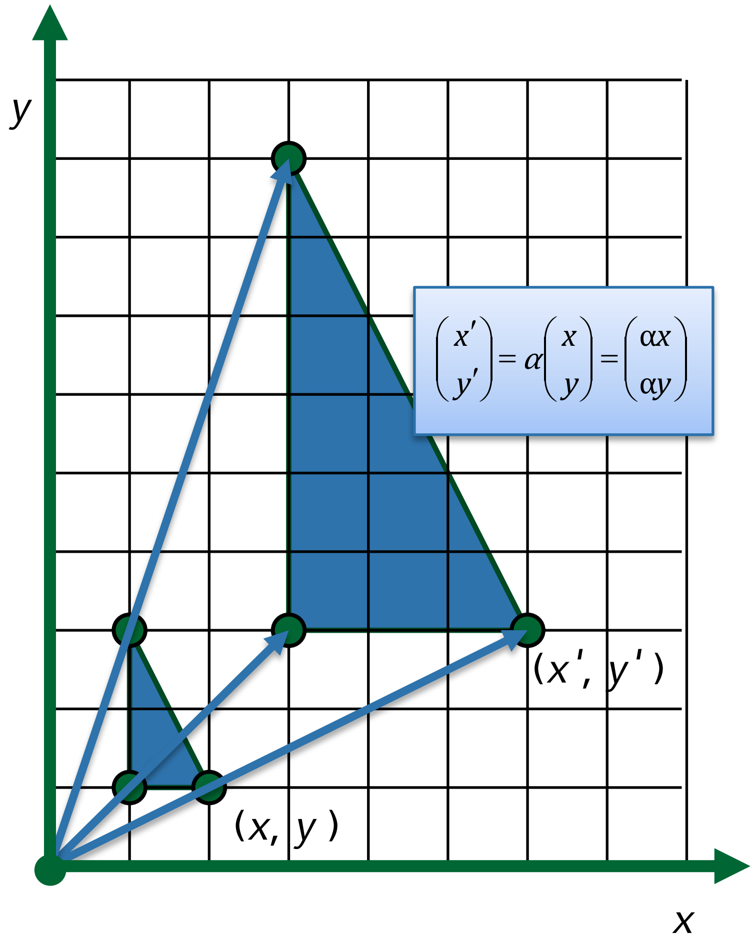
Translation



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} dx \\ dy \end{pmatrix}$$

- Punkt (x, y) wird auf gerade Linie nach (x', y') verschoben.
- Addition des Verschiebungsvektors
- Beschreibung der Translation durch einen Vektor (dx, dy), der die Verschiebungsweite in x- und y-Richtung angibt

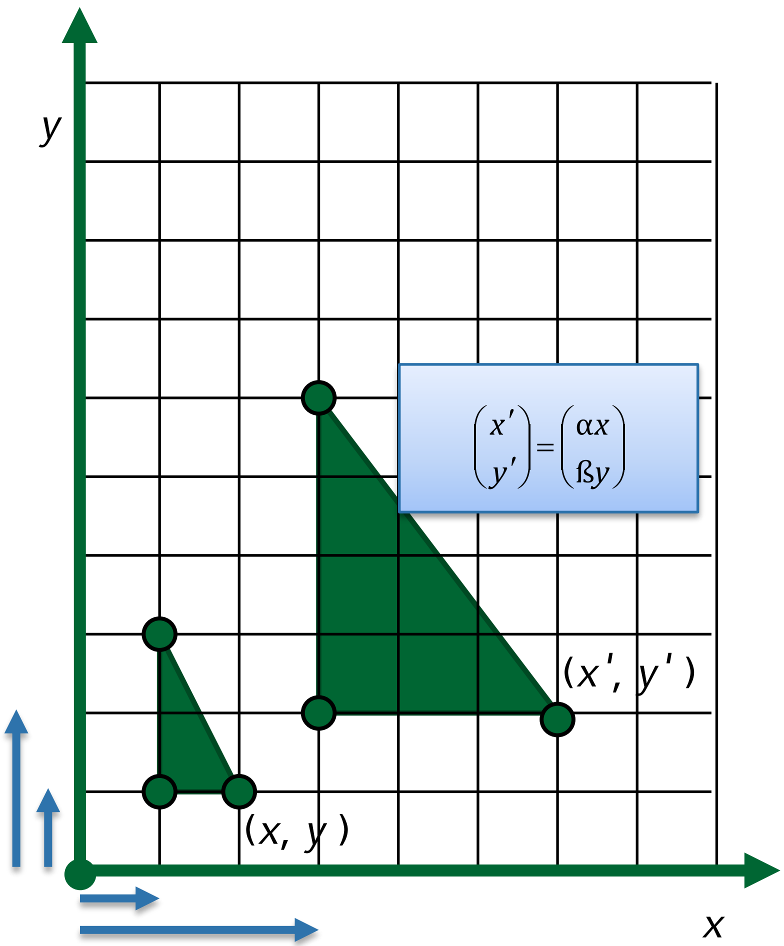
Uniforme Skalierung



$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \alpha \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \alpha x \\ \alpha y \end{pmatrix}$$

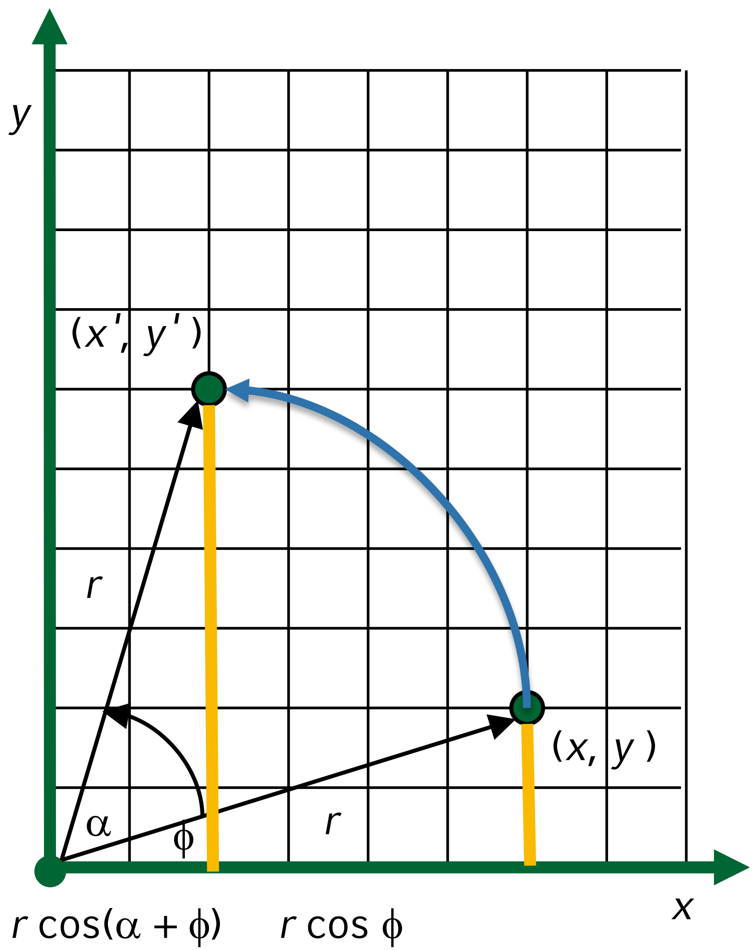
- Zentrum der Skalierung ist o, Skalierung erfolgt in alle Richtungen uniform mit dem skalaren Faktor α
- Multiplikation mit dem Skalierungsfaktor
- Ortsvektor zu (x, y) wird auf das α-fache verlängert, um (x', y') zu erhalten

Nicht-uniforme Skalierung



- Zentrum der Skalierung ist o, Skalierung erfolgt in *x*-Richtung mit dem Faktor α, in *y*-Richtung mit β (Skalierungsvektor $(\alpha, \beta)^T$)
- Multiplikation mit entsprechenden Skalierungsfaktoren
- Ortsvektor zu (*x*, *y*) wird auf das α-fache in *x*-Richtung und das β-fache in *y*-Richtung verlängert.

Rotation



- Rotationszentrum ist o.
- Positive Werte von α ergeben eine Drehung entgegen dem Uhrzeigersinn.
- Punkt (x, y) wird um den Winkel α um o gedreht, so dass sich der Punkt (x', y') ergibt.

Rotation: Herleitung der Berechnungs-Vorschrift

I. $x = r \cos \phi$

II. $y = r \sin \phi$

III. $x' = r \cos (\alpha + \phi) = r \cos \phi \cos \alpha - r \sin \phi \sin \alpha$

IV. $y' = r \sin (\alpha + \phi) = r \cos \phi \sin \alpha - r \sin \phi \cos \alpha$

- anschließend (I) in (IV) einsetzen:

$$x' = x \cos \alpha - y \sin \alpha$$

$$y' = x \sin \alpha + y \cos \alpha$$

Rotation: Herleitung der Berechnungs-Vorschrift

- Rotationen um negative Winkel erfolgen mit dem Uhrzeigersinn; ausnutzen:
- $\cos(-\alpha) = \cos(\alpha)$ und $\sin(-\alpha) = -\sin(\alpha)$
- die Berechnungs-Vorschrift

  $$x' = x \cos \alpha - y \sin \alpha$$

  $$y' = x \sin \alpha + y \cos \alpha$$

- kann als Matrix-Vektormultiplikation ausgedrückt werden:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$$

Kritische Rückschau

- Bisherige Betrachtung:
  - Translation: Addition des Verschiebungsvektors
  - Skalierung: Multiplikation des Skalierungsfaktors
  - Rotation: Matrixmultiplikation
- **Fazit:**
  - Hierbei erfolgt keine einheitliche Behandlung
  - Es ergeben sich Schwierigkeiten bei zusammengesetzten Transformationen
- **Gesucht:** Einheitliche Repräsentation von Transformationen

Homogene Koordinaten

- Ein Koordinatensystem wird in ein homogenes Koordinatensystem überführt, indem eine zusätzliche Dimension eingeführt wird: $n \rightarrow n + 1$ Dimensionen.
- Ein Punkt (x, y, z) wird in homogenen Koordinaten durch das Tripel $(x \cdot w, y \cdot w, w)$ repräsentiert, mit $w \neq 0$.
- Normalisierte Darstellung: $w = 1 \Rightarrow (x, y, 1)$
- Jeder Punkt hat unendlich viele äquivalente Repräsentationen in homogenen Koordinaten.

Vorteile

- Repräsentation aller Punkte in homogenen Koordinaten ermöglicht einheitliche Behandlung der Transformationen

Vorteile

- Fragen:
  - Was steht für das Fragezeichen?
  - Welcher Art ist die Operation ist „*“?
- Antwort:
  - Transformationen werden als Matrizen repräsentiert
  - Verknüpfung durch Multiplikation

$$\begin{pmatrix} wx' \\ wy' \\ w \end{pmatrix} = ?* \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Homogene Koordinaten

- **Translation**
  - Vorher: Addition eines Vektors
  - Jetzt: Translationsmatrix

$$\begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- **Skalierung**
  - Vorher: Multiplikation mit Skalierungsfaktoren
  - Jetzt: Skalierungsmatrix

$$\begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} sx & 0 & 0 \\ 0 & sy & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

homogene Koordinaten

- **Rotation**
  - Vorher: komplexe Gleichung oder Matrixmultiplikation
  - Jetzt: Rotationsmatrix

$$\begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \Rightarrow$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Homogene Koordinaten

- Allgemeine 2D-Transformations-Matrix

$$T = \begin{pmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{pmatrix}$$

Skalierung

Rotation

Translation

Inverse Transformationen

- Frage: Wie macht man Transformationen wieder rückgängig, also was sind die entsprechenden inversen Transformationen?
- für die elementaren Transformationen relativ einfach:
  - Translation: Verschiebung um den negativen Verschiebungsvektor $T^{-1}(dx, dy) = T(-dx, -dy)$
  - Skalierung: Skalierung mit dem reziproken Skalierungsfaktor
    - $S^{-1}(\alpha) = S(1/\alpha)$
  - Rotation: Rotation um den negativen Rotationswinkel. Da aber Rotationsmatrizen speziell orthogonal sind, gilt hier $R^{-1} = R^T$ und keine Neubesetzung der Matrix ist notwendig.

Zusammengesetzte Transformationen

- Nacheinander-Ausführung zweier Translationen
- Translation ist additiv, d. h. Ergebnis ist eine Verschiebung um die Summe der beiden Vektoren

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & dx_2 \\ 0 & 1 & dy_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & dx_1 \\ 0 & 1 & dy_1 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & dx_1 + dx_2 \\ 0 & 1 & dy_1 + dx_2 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Zusammengesetzte Transformationen

- Nacheinander-Ausführung zweier Skalierungen
- Skalierung ist multiplikativ, d. h. Ergebnis ist eine Skalierung um das Produkt der beiden Faktoren

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} sx_2 & 0 & 0 \\ 0 & sy_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} sx_1 & 0 & 0 \\ 0 & sy_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

$$\Downarrow$$

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} sx_1 \cdot sx_2 & 0 & 0 \\ 0 & sy_1 \cdot sy_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Zusammengesetzte Transformationen

- Nacheinander-Ausführung zweier Rotationen
- Rotation ist additiv

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\alpha_2 & -\sin\alpha_2 & 0 \\ \sin\alpha_2 & \cos\alpha_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\alpha_1 & -\sin\alpha_1 & 0 \\ \sin\alpha_1 & \cos\alpha_1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

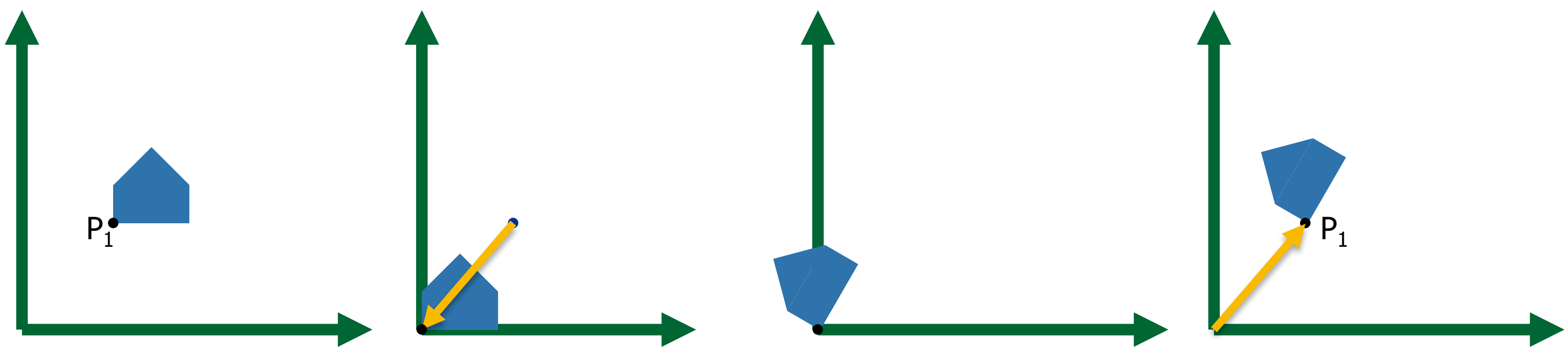
Zusammengesetzte Transformationen

- Allgemein: Arbeit mit Transformationsmatrizen
  - vereinheitlicht alle Transformationen
  - einfache Möglichkeit der Kombination von Transformationen
- hier genutzte Schreibweise
  - Transformationen werden in der Reihenfolge T_1, T_2, ..., T_n ausgeführt $\Rightarrow P' = T_n \cdot \ldots \cdot T_2 \cdot T_1 \cdot P$

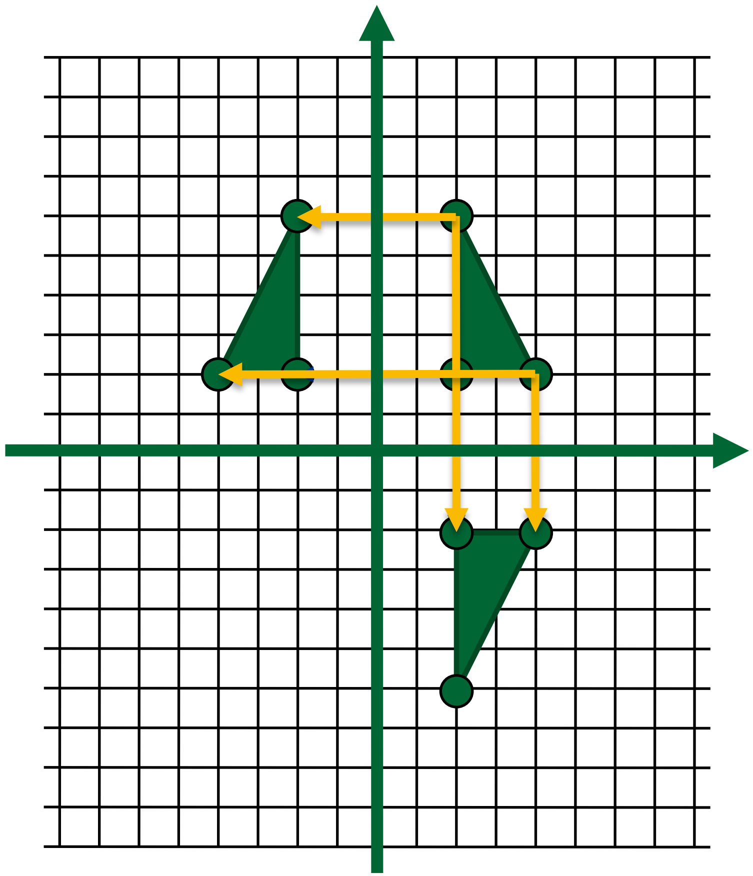
Zusammensetzen beliebiger Transformationen

- Rotation eines Punktes um einen *beliebigen* Punkt P_1 in der Ebene, Ausführung in drei Schritten:

  1. Translation, so dass P_1 im Ursprung liegt
  2. Rotation um den Ursprung
  3. Rück-Translation des Ursprungs nach P_1



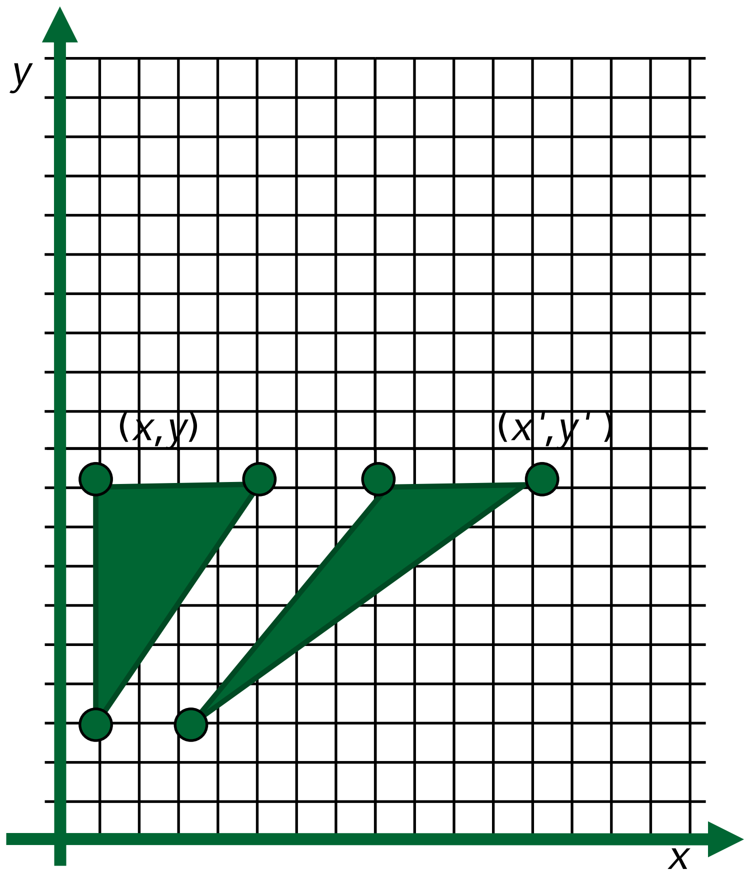
Weitere Transformationen: Spiegelung



- zwei Standard-Möglichkeiten:
  - Spiegelung an der x-Achse
  - Spiegelung an der y-Achse
- Spiegelung wird implementiert als Skalierung mit dem Skalierungs-Faktor –1

$$T = \begin{pmatrix} (-1) & 0 & 0 \\ 0 & (-1) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Scherung



- Versatz parallel zur *x*-Achse, proportional zur *y*-Position (bzw. umgekehrt)
- zwei Standardmöglichkeiten:
  - Scherung entlang der *x*-Richtung
  - Scherung entlang der *y*-Richtung

$$T = \begin{pmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad T = \begin{pmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Spezifikationen in 3D

Transformationen in 3D

Dreidimensionale Transformationen

- Vorgehensweise gleich zu 2D
- Arbeit mit homogenen Koordinaten
- homogene Koordinaten sind jetzt vierdimensional
- Transformationsmatrizen demzufolge 4×4-Matrizen
- Anwendung wie in 2D

Transformationsmatrizen: Translation

- Addition eines Translationsvektors bzw.
- Multiplikation mit einer Translationsmatrix

$$T = \begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Skalierung

- Multiplikation mit Skalierungsfaktoren bzw. Multiplikation mit einer Skalierungsmatrix
- uniforme Skalierung, wenn *sx* = *sy* = *sz*, sonst
- nichtuniforme Skalierung

$$S = \begin{pmatrix} sx & 0 & 0 & 0 \\ 0 & sy & 0 & 0 \\ 0 & 0 & sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Rotation

- nicht mehr ganz trivial, da jetzt Rotationen um die verschiedenen Koordinatenachsen betrachtet werden müssen
- drei verschiedene Rotationsmatrizen
- Achse, um die gedreht wird, bleibt „Einheitsvektor“ in der Matrix

$$R_x = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_y = \begin{pmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$R_z = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Konvertierung

- Überführung des rechtshändigen in das linkshändige Koordinatensystem
- Konvertierung:

$$T_{r \to l} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

homogene Koordinaten

- Zusammensetzen von Transformationen
- auch über Multiplikation der Matrizen
- generelle Transformationsmatrix in 3D

$$T = \begin{pmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{pmatrix}$$

Skalierung

Rotation

Translation

Zusammenfassung Koordinaten

- Einheitliche Behandlung der Transformationen durch Übergang zu homogenen Koordinaten und zur Darstellung der Transformationen durch Matrizen
- Zusammengesetzte Transformationen durch Hintereinanderausführen von elementaren Transformationen, entspricht Multiplikation der Matrizen
- Transformation der Objekte oder des Koordinaten-Systems

Unterschiedliche Koordinaten

- Bisher sind Objekte in **lokalen Koordinaten** gegeben, d. h. jedes Objekt hat sein eigenes Koordinatensystem
- Berechnungen, die mehrere Objekte einbeziehen, sind schwierig
- Transformation in ein gemeinsames Koordinatensystem notwendig
- **Weltkoordinaten**
- Platzieren der Objekte zu einem globalen Koordinatensystem

Zusammenfassung Transformationen

- Geometrische Transformationen sind Abbildungen vom $\Re n$ in den $\Re n$
  - von besonderem Interesse $\Re 2 \rightarrow \Re 2$ und $\Re 3 \rightarrow \Re 3$
- für Computergraphik relevant:
  - Translation
  - Skalierung
  - Rotation
  - (Scherung, Spiegelung)
- einheitliche Behandlung der Transformationen durch Übergang zu homogenen Koordinaten und zur Darstellung der Transformationen durch Matrizen

LFE Medieninformatik • Prof. Dr. Ing. Axel Hoppe

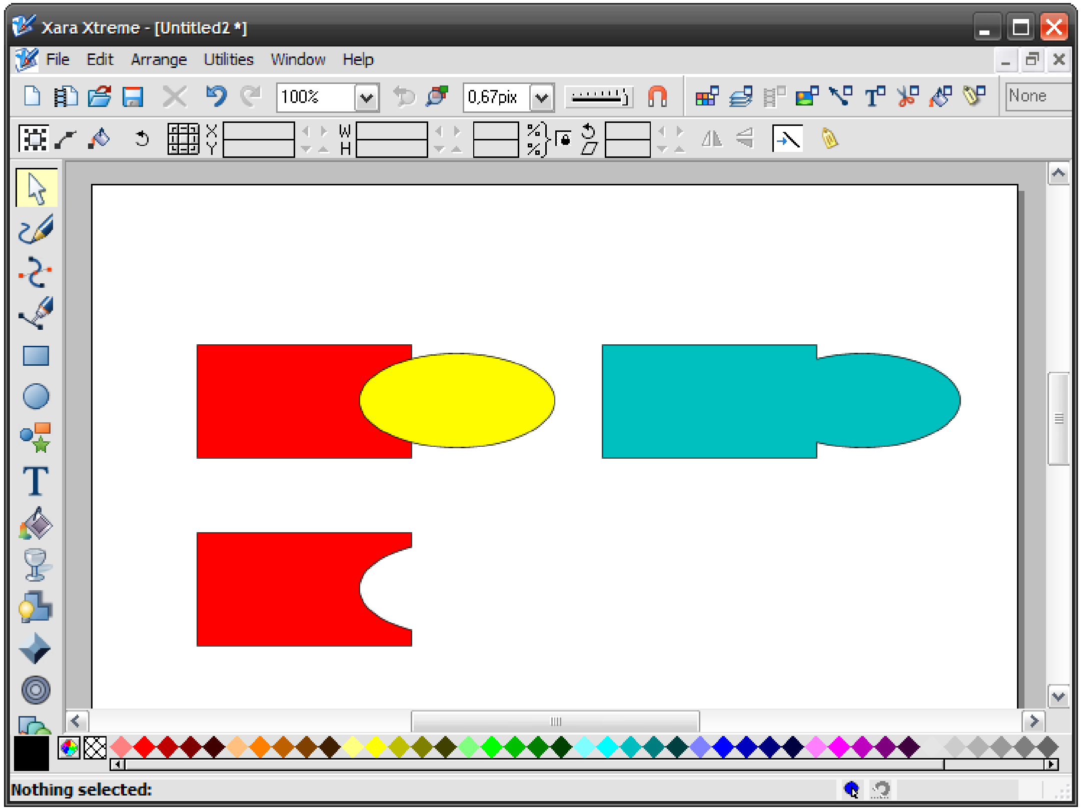
Spezifikationen in 2D und 3D

Transformationen für den Anwender

Arbeiten mit 2D-Primitiven

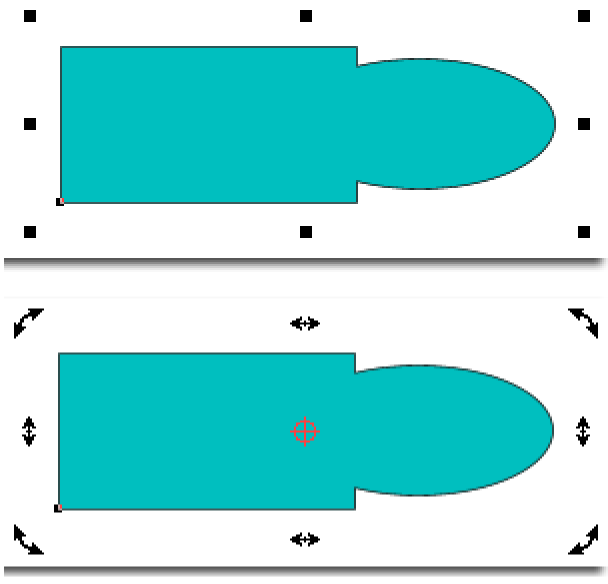
2D-Vektorgrafik: Szene wird u. a. erstellt, indem 2D-Primitive über boolesche Operationen zu neuen Formen erstellt werden

Formen können anschließend verlustfrei transformiert werden, um die 2D-Szene zu komponieren (= gemäß der Vision alle Objekte zueinander anzuordnen)



Arbeiten mit 2D-Primitiven

- Werkzeug: Xara
- Für das interaktive Ausführen von Transformationen existieren unterschiedliche Manipulatoren
- Translation per Drag & Drop
- Skalieren 1 × Klicken
- Rotieren, Scheren 2 × Klicken

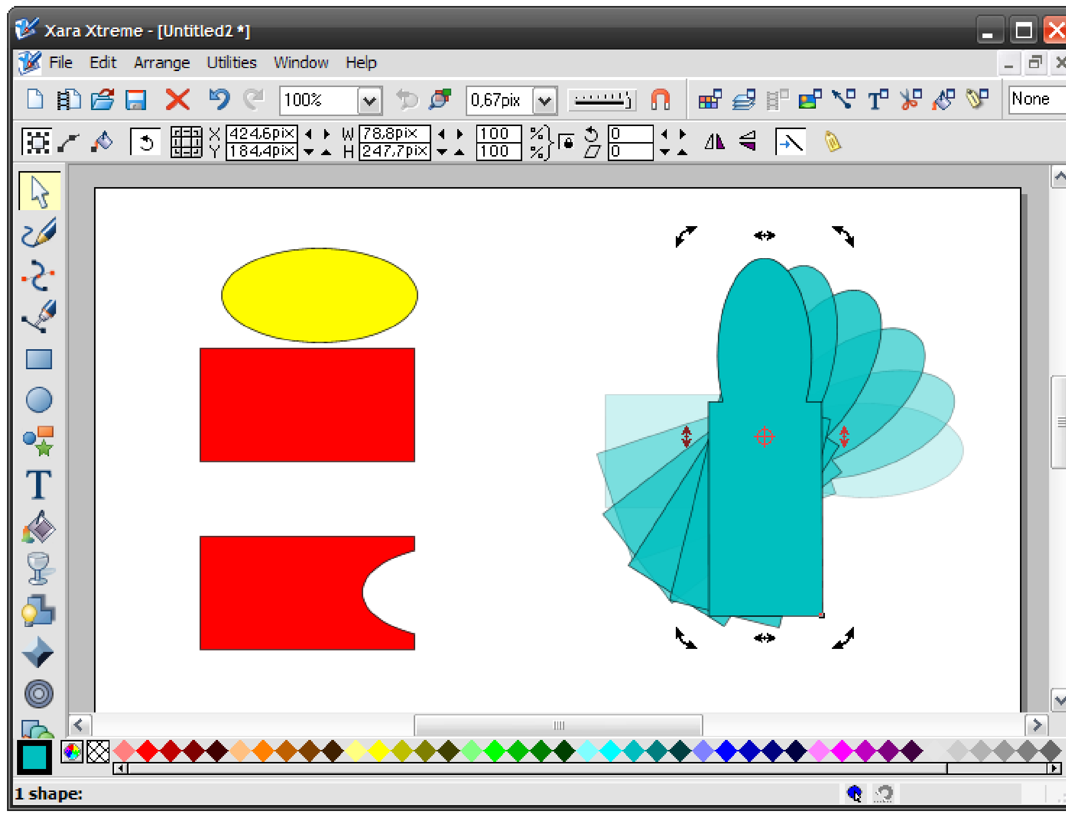


Arbeiten mit 2D-Primitiven

Rotation bezieht sich auf Bezugspunkt

dieser ist Ursprung eines lokalen Koordinatensystems pro Objekt

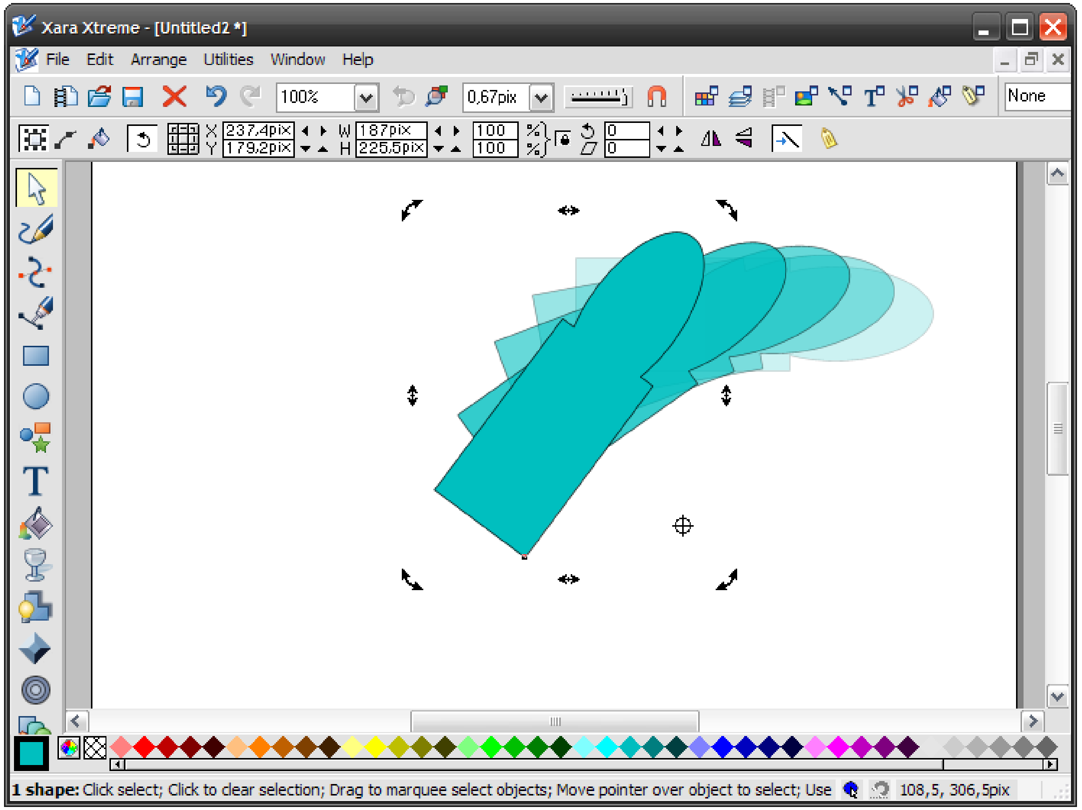
standardmäßig im Zentrum des Objekts



Arbeiten mit 2D-Primitiven

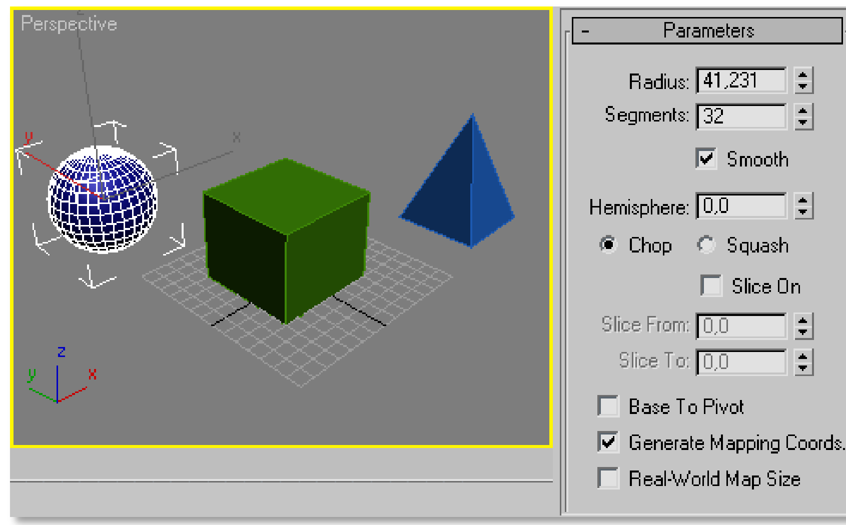
Bezugspunkt kann interaktiv geändert werden

Das Ergebnis der Rotation ist entsprechend



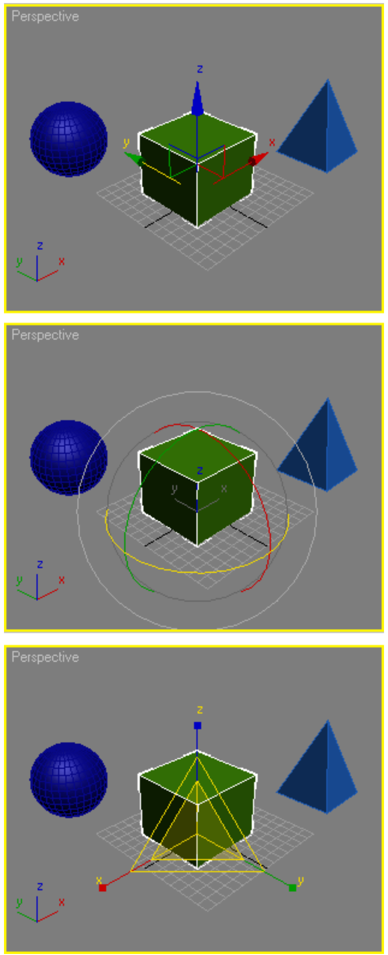
Arbeiten mit 3D-Primitiven

- Werkzeug: 3D-Studio MAX
- 3D-Primitive werden auf Parameter abgebildet
- Objektorientierung: Jedes Primitiv „kennt" seine Parameter



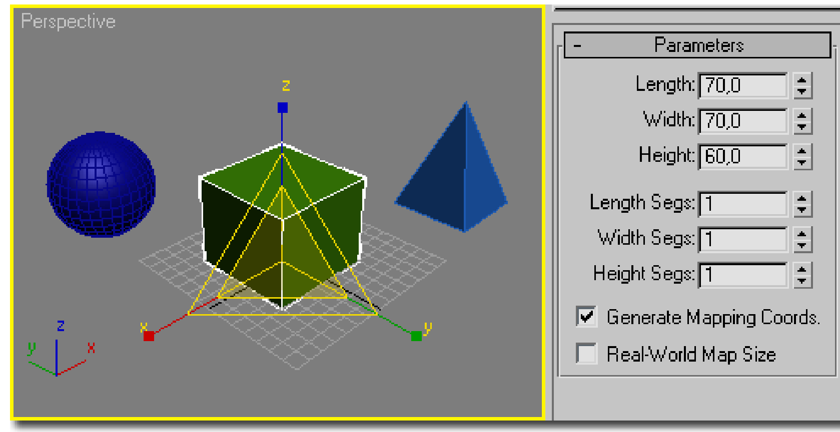
Arbeiten mit 3D-Primitiven

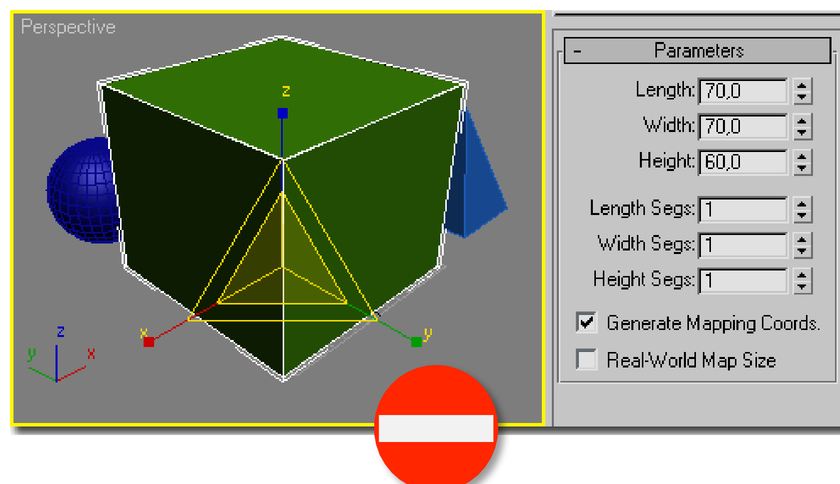
- Transformationen sind interaktiv möglich über so genannte „Gizmos“
- das lokale Koordinatensystem wird repräsentiert durch den „Pivot-Point“



Problem Skalierung

- Zur Erinnerung: 3D-Primitive werden über Parameter eingestellt
- nach Skalieren:
- weiß das Objekt nichts davon, weil sein lokales Koordinatensystem mitskaliert wurde
- Mit Blick auf (mögliche) Programmierung: Niemals skalieren!





Quellen

- Foley, van Dam, Feiner, Hughes. *Computer Graphics, Principles and Practice*. Zweite Auflage, Addison Wesley. ISBN 0-201-84840-6.
- Bernhard Preim. *Computergraphik 1*. Universität Magdeburg, Vorlesungsskript, Juli 2005.