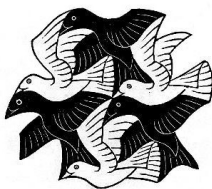


ZUSAMMENFASSUNG

GRAFISCHE DATENVERARBEITUNG

SOMMERSEMESTER 2025



Niklas Conrad
Fakultät Informatik
Bachelor of Science Informatik
Juni 2025



NIKLAS CONRAD

Grafische Datenverarbeitung

FAKULTÄT INFORMATIK:

<https://www.hs-schmalkalden.de/hochschule/fakultaeten/fakultaet-informatik>

© Juni 2025

INHALTSVERZEICHNIS

1	Grundkonzepte der Computergrafik	1
2	Modelle	3
2.1	Theoretische Grundlagen	3
2.1.1	Modellbildung in der Computergrafik	3
2.1.2	Rekonstruktion	3
2.1.3	Dreiecksmodelle - Meshes	4
2.1.4	Einschub: Mannigfaltigkeit	5
2.1.5	Topologie und Geometrie	6
2.1.6	Attribute und Merkmale (Features)	7
2.1.7	Dimensionalität	7
2.1.8	Einschub: Interpolation	8
2.1.9	Splines und NURBS (Interpolationstechniken)	9
2.2	Flächenmodelle	9
2.2.1	Polygonale Flächen (Facettierung)	9
2.2.2	Algorithmische Flächen (Parametrische Flächen)	10
2.2.3	Variationsbasierte Flächen (Minimierungsprobleme)	10
2.2.4	Ruled Surfaces (Regelflächen)	11
2.2.5	Freiformflächen (Freiformmodellierung)	11
2.3	Meshes	11
2.4	Volumenmodelle	13
3	Transformation	15
3.1	Einführung	15
3.2	Einschub: Mathematische Abbildungen	16
3.3	Lineare Transformationen	17
3.4	Transformation im \mathbb{R}^2	19
3.4.1	2D-Translation	19
3.4.2	Uniforme Skalierung	19
3.4.3	Nicht-uniforme Skalierung	21
3.4.4	2D-Rotation	22
3.4.5	Kritische Rückschau	24
3.4.6	Einführung homogener Koordinaten	25
3.4.7	Transformationen in homogenen Koordinaten	27
3.4.8	Verkettung von Transformationen	30
3.4.9	Isometrien - Abstands- und Winkeltreue Transformationen	31
3.4.10	Typisierung von Transformationen	32
3.4.11	3.5.5 Transformation entlang einer Achse	33
3.5	Verkettung affiner Transformationen im \mathbb{R}^2	35
3.5.1	Motivation und Bedeutung	35
3.5.2	Allgemeine Form affiner Transformationen	35
3.5.3	Zusammensetzen und Nicht-Kommutativität	36
3.5.4	Transformation um einen lokalen Punkt (Pivot)	37

3.6	Transformation im \mathbb{R}^3	39
3.7	Kamera- und Projektionstransformationen	40
3.7.1	Koordinatensysteme	40
3.7.2	Kamera-Transformation (View Matrix)	41
3.7.3	Projektion (Projection Matrix)	42
4	Sichtbarkeit – Visibility	45
4.1	Transformationspipeline und Projektionsraum	45
4.2	Tiefenberechnung und Normalisierung	47
4.3	Clip Space und NDC-Raum	48
4.4	Zusammenfassung der Schritte	49
5	Prüfungsfragen	50
	Abbildungsverzeichnis	54

1

GRUNDKONZEPTE DER COMPUTERGRAFIK

SEPARATIONSPRINZIP Das Separationsprinzip trennt klar zwischen Modellierung, Szene und Bildsynthese. Zunächst werden Objekte unabhängig von der Szene erstellt und definiert. Danach wird die Szene komponiert, indem man diese Objekte anordnet und ihre Beziehungen festlegt. Schließlich erfolgt die Bildsynthese, bei der die eigentliche Darstellung (Rendering) entsteht. Szene

SZENE Eine Szene beschreibt eine konkrete Zusammenstellung von Objekten, Lichtquellen, Kameraposition und weiteren grafischen Elementen. Sie bildet die Grundlage für die Erstellung eines computergenerierten Bildes. Betrachter

BETRACHTER Der Betrachter (auch „virtuelle Kamera“) bestimmt den Blickwinkel und Sichtbereich auf eine Szene. Durch seine Position, Ausrichtung, Brennweite und Perspektive definiert der Betrachter, wie die Szene wahrgenommen und später gerendert wird. Bildsynthese

BILDSYNTHESE Bildsynthese (auch Rendering genannt) bezeichnet den Prozess, aus einer definierten Szene ein sichtbares Bild zu erzeugen. Es gibt verschiedene Techniken der Bildsynthese:

- Rasterisierung
- Raytracing (Strahlenverfolgung)
- Radiosity

Je nach gewählter Technik werden Realismusgrad und Berechnungsaufwand unterschiedlich beeinflusst.

LICHT UND OBERFLÄCHEN Das Zusammenspiel von Licht und Oberflächen bestimmt, wie Objekte visuell erscheinen. Lichtquellen (z. B. Punktlichter, gerichtetes Licht) interagieren mit Materialien der Oberflächen durch:

- Reflexion (Spiegelung, diffus, spekulär)
- Absorption (Licht wird geschluckt)
- Transmission (Durchdringung, Transparenz)

Dadurch entstehen Farbe, Schatten und Textur auf den dargestellten Objekten.

LICHT ALS GLOBALES PHÄNOMEN berücksichtigt, dass Licht nicht nur direkt auf Oberflächen trifft, sondern indirekt weiter reflektiert und gestreut wird. Diese globalen Effekte umfassen:

- Indirekte Beleuchtung
- Weiche Schatten
- Farbliche Interaktion zwischen Objekten (Farbabstrahlung)

Die realitätsnahe Simulation solcher globalen Lichtphänomene erfordert komplexere Methoden wie Global Illumination, Radiosity oder Photon Mapping.

2 | MODELLE

2.1 THEORETISCHE GRUNDLAGEN

2.1.1 Modellbildung in der Computergrafik

In der Computergrafik werden reale, oft komplexe Objekte und Umgebungen in einfachere mathematische und symbolische Modelle übersetzt. Dies dient sowohl zur effizienten und präzisen Abbildung komplexer Geometrien, als auch der Reduzierung von Datenmengen durch Abstraktion.

BEISPIEL

Ein Kreis in der Mathematik wird allgemein durch die implizite Gleichung $X^2 + Y^2 = R^2$ beschrieben. Dabei beschreibt R den Radius des Kreises. Diese Form nennt man implizit, weil nicht explizit definiert wird, wie x und y einzeln bestimmt werden, sondern nur ihr Verhältnis. Für die Computergrafik (also zur praktischen Nutzung in Programmen) wird oft eine parametrische Darstellung verwendet. Diese Form beschreibt explizit die Positionen auf dem Kreis durch einen Winkelparameter α :

$$x(\alpha) = R \cdot \sin(\alpha), \quad y(\alpha) = R \cdot \cos(\alpha)$$

Hier ist α ein Winkel, welcher typischerweise Werte zwischen 0 und 2π annimmt. Die Wahl der Schrittweite von α beeinflusst die Genauigkeit der Modellierung. Kleinere Schrittweiten ergeben präzisere Kreismodelle mit mehr Vertices, aber erhöhen den Rechen- und Speicheraufwand.

2.1.2 Rekonstruktion

Unter **Rekonstruktion** versteht man in der Computergrafik die Wiederherstellung einer visuellen Darstellung aus symbolisch gespeicherten Modellen oder Rasterbildern. Rasterbilder sind pixelbasierte Bilder, welche das Gehirn visuell zu einer zusammenhängenden Struktur rekonstruiert. Rekonstruktion in digitaler Bildverarbeitung erfolgt mithilfe sogenannter Deskriptoren, welche charakteristische Merkmale eines Objekts beschreiben (z.B. Ecken, Kanten).

- Ein Algorithmus findet zunächst wichtige Merkmale wie Kanten oder Eckpunkte in einem Pixelbild.
- Diese Deskriptoren dienen dann als Grundlage, um das Objekt, etwa einen Kreis, wiederherzustellen oder zu rekonstruieren.
- Die Genauigkeit dieser Rekonstruktion hängt stark von den verwendeten Merkmalen (Deskriptoren), der Auflösung der Daten und den eingesetzten Algorithmen ab.

2.1.3 Dreiecksmodelle - Meshes

Meshes sind das wichtigste Modell in der digitalen 3D-Datenverarbeitung. Sie setzen sich aus mehreren grundlegenden Elementen zusammen:

VERTEX Punkt im dreidimensionalen Raum (Plural: Vertices)

EDGE Verbindung zwischen zwei Vertices – kann lineare oder interpolierte Verbindung sein

FACE Flächenstücke zwischen Kanten – häufig Dreiecke, manchmal Vierecke

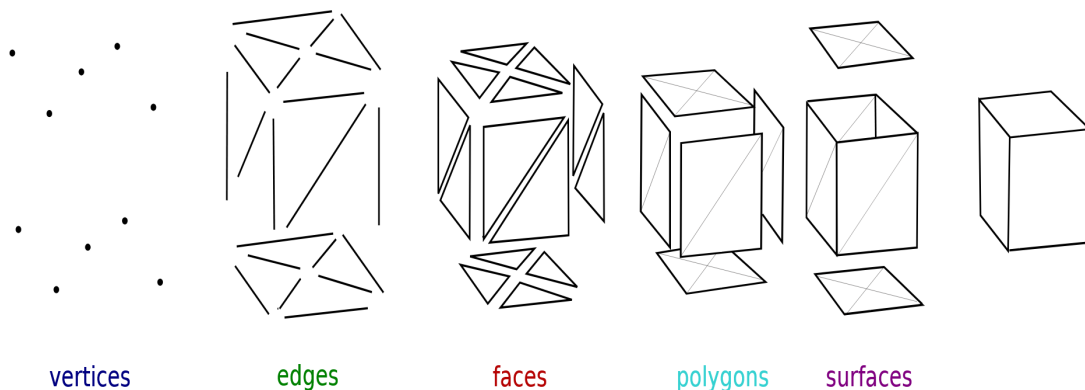
POLYGON Koplanare Flächen (in derselben Ebene liegend), die eine größere Fläche bilden.

SURFACE Nutzerdefinierte Zusammenfassung von mehreren Polygonen.

SOLID Geschlossenes, "wasserdichtes" Modell (keine offenen Kanten). Alle Nachbarschaftsbeziehungen sind definiert (manifold).

WARUM DREIECKE?

Ein wesentlicher Grund dafür liegt in ihrer geometrischen Eigenschaft der **Koplanarität**: Drei Punkte im dreidimensionalen Raum spannen immer genau eine eindeutige Ebene auf. Das bedeutet, dass jedes Dreieck per Definition immer in einer Ebene liegt – unabhängig von der Form oder Krümmung der gesamten Oberfläche. Dies vereinfacht viele Berechnungen erheblich, etwa bei der Bestimmung von Flächennormalen, bei Beleuchtungsmodellen (z.B. Phong-Shading), beim Z-Sorting oder bei der Kollisionserkennung



2.1.4 Einschub: Mannigfaltigkeit

MATHEMATISCHE GRUNDLAGEN

Eine **Mannigfaltigkeit** ist ein topologisches Objekt, das lokal so aussieht, als wäre es ein Teil eines reellen euklidischen Raums \mathbb{R}^n . Das bedeutet, dass kleine Abschnitte (Nachbarschaften jedes Punktes) einer Mannigfaltigkeit in einem Koordinatensystem dargestellt werden können, auch wenn die gesamte Struktur global komplexer ist.

FORMALE DEFINITION

Eine n -dimensionale Mannigfaltigkeit ist ein topologischer Raum M , für den gilt:

- Jeder Punkt $p \in M$ hat eine offene Umgebung, die homöomorph (strukturidentisch) zu einer offenen Teilmenge des \mathbb{R}^n ist.
- Diese Eigenschaft nennt man lokal euklidisch.

BEISPIELE

1D-MANNIGFALTIGKEIT Eine Linie oder Kreislinie, denn jeder Punkt darauf hat eine Umgebung, die einer offenen Strecke im \mathbb{R} entspricht.

2D-MANNIGFALTIGKEIT Eine Kugeloberfläche (Sphäre), da jede Umgebung auf der Oberfläche lokal aussieht wie ein Stück ebene Fläche.

NICHT-MANNIGFALTIGKEIT Eine Figur, in der mehrere Flächen an einer einzigen Kante zusammentreffen, sodass diese nicht mehr lokal einer Ebene oder einem Raum entspricht, ist keine Mannigfaltigkeit.

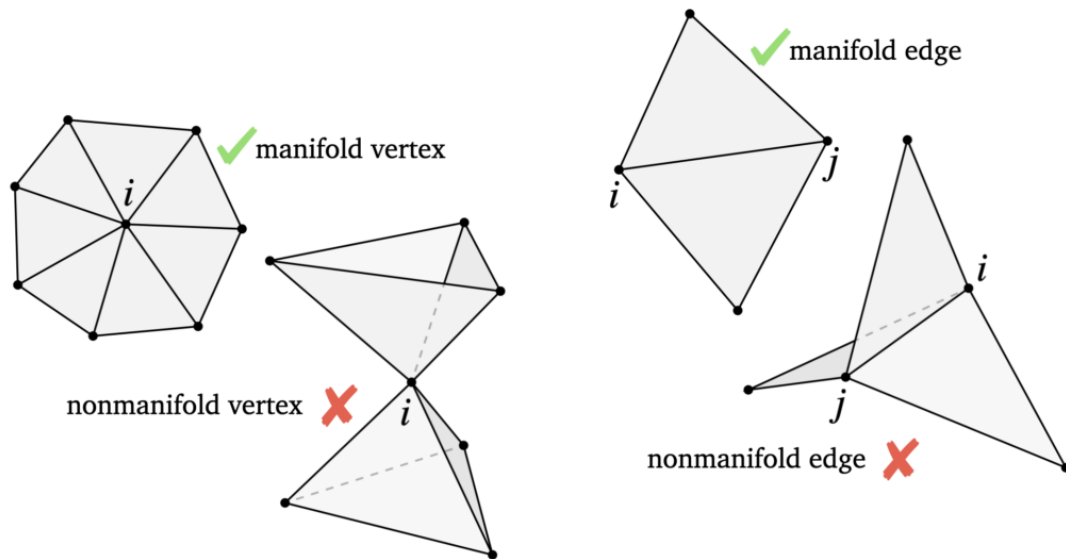
MANNIGFALTIGKEIT IN DER COMPUTERGRAFIK

In der Computergrafik und Modellierung bedeutet ein manifold mesh (mannigfaltiges Netz):

- Jeder Punkt, jede Kante und jede Fläche hat klar definierte Nachbarschaftsbeziehungen.
- Eine Kante darf maximal zwei Flächen verbinden.
- Lokale Struktur ist stets eindeutig definiert, ohne Brüche, Löcher oder überflüssige Überschneidungen.

BEISPIELE IN DER COMPUTERGRAFIK

- Würfel, Kugel, Torus (Donut): Diese Objekte haben jeweils klar definierte Oberflächen ohne offene Kanten. Jeder Punkt auf ihnen ist eindeutig lokal euklidisch
→ **manifold Mesh**
- Zwei Würfel, die sich nur an einer einzelnen Kante oder einem einzelnen Vertex berühren (Kanten oder Vertices sind mehrfach belegt). → **non-manifold Mesh**
- Eine Fläche, von der mehr als zwei andere Flächen abgehen. → **non-manifold Mesh**



WARUM MANIFOLDS?

- Mannigfaltigkeit garantiert konsistente Berechnungen, klare Modellierung und Simulation.
- Ermöglicht zuverlässige physikalische Simulationen und Kollisionserkennung.
- Vereinfacht Berechnung von Normalen und Beleuchtung (Shading).

2.1.5 Topologie und Geometrie

GEOMETRIE beschreibt konkrete Größen, Positionen, Maße und Formen von Objekten im Raum (z.B. Koordinaten, Winkel, Längen). Geometrie ist essentiell für die präzise Abbildung von Objekten und Messungen (z.B. CAD-Systeme, Simulationen).

TOPOLOGIE beschreibt strukturelle Beziehungen unabhängig von der konkreten Form oder Größe. Sie beschäftigt sich damit, wie Elemente miteinander verbunden oder angeordnet sind (z.B. Nachbarschaftsbeziehungen). Topologie ist besonders wichtig für Modellierungsprozesse und Optimierungen (z.B. Nachbarschaftsbestimmung, Raytracing, Netzanalysen).

Ein Objekt kann dieselbe Topologie, aber unterschiedliche Geometrien haben und umgekehrt.

BEISPIELE

- Gleiche Geometrie – Unterschiedliche Topologie: Zwei identisch aussehende 3D-Modelle (z.B. Würfel), jedoch mit unterschiedlicher interner Struktur (z.B. ein Würfel hohl, einer massiv).
- Gleiche Topologie – Unterschiedliche Geometrie: Ein Würfel und ein verzerrter Würfel (Quader) haben dieselbe Anzahl an Vertices, Edges und Faces und gleiche Verbindungstopologie, unterscheiden sich aber in geometrischen Maßen (Längen, Winkel).

TOPOLOGISCHE VOLLSTÄNDIGKEIT beschreibt, wie gut oder vollständig die Beziehungen zwischen Elementen in einem Modell definiert sind.

- Je mehr Informationen über Topologie (räumliche Beziehungen) vorhanden sind, desto leichter lassen sich Geometrien manipulieren oder rekonstruieren.
- Vollständige Topologie erlaubt z.B. einfache Änderungen am Modell (z.B. Ersetzen von Teilen), ohne dass die Struktur neu berechnet werden muss.

TOPOLOGISCHE BESTIMMTHEIT

Je nach vorhandenem topologischem Informationsgehalt entstehen unterschiedliche Arten von Modellen:

1. Punktwolken (Point Clouds) – Enthalten nur Vertices, keine Beziehungen oder Kanten.
2. Drahtmodelle (Wireframe Models) – Enthalten Vertices und Edges, aber keine Flächeninformationen.
3. Flächenmodelle (Surface Models) – Enthalten Faces zusätzlich zu Vertices und Edges, jedoch keine volumetrischen Informationen.
4. Volumenmodelle (Solid Models) – Geschlossene Modelle mit definierten Volumina. Alle Nachbarschaftsbeziehungen sind klar definiert.

Diese Abstufungen erhöhen sich mit der Komplexität der topologischen Beschreibung und der Möglichkeiten zur Manipulation und Nutzung der Modelle.

2.1.6 Attribute und Merkmale (Features)

Neben geometrischen und topologischen Informationen können Modelle zusätzliche physische oder funktionale Eigenschaften enthalten:

ATTRIBUTE Eigenschaften wie Masse, Material, Farbe, Temperatur, etc.

MERKMALE (FEATURES) Charakteristische Besonderheiten, die z.B. für Simulationen relevant sind (Reibungswerte, Reflexionseigenschaften).

BEISPIEL

Ein Fahrzeugmodell könnte Attribute wie Gewicht, Materialstärke oder Farbe enthalten.

2.1.7 Dimensionalität

Die Dimensionalität eines Modells beschreibt, in welchem Raum es vollständig definiert ist:

- Modelle sind meist vollständig bestimmt im \mathbb{R}^n , also einem n-dimensionalen reellen Vektorraum.
- Computergrafik nutzt oft Projektionen vom höherdimensionalen Raum \mathbb{R}^4 (3D plus Zeit oder homogene Koordinaten) auf \mathbb{R}^3 (rein räumliche Darstellung).

BEISPIEL

3D-Objekt in einer 3D-Szene wird für die Darstellung auf einem 2D-Bildschirm projiziert.

2.1.8 Einschub: Interpolation

Interpolation bezeichnet das mathematische Verfahren, um zwischen bekannten Werten (Punkten) neue, unbekannte Werte zu berechnen und somit eine stetige Kurve oder Fläche zu erzeugen.

ERKLÄRUNG

- Ziel ist eine glatte, kontinuierliche Verbindung zwischen diskreten Punkten.
- Unterschiedliche Arten der Interpolation existieren:
 - Lineare Interpolation: einfachste Form, verbindet Punkte direkt durch Geraden.
 - Polynomial-Interpolation: Kurven höheren Grades.
 - Spline-Interpolation: nutzt Teilpolynome (Splines), um glatte Kurven zu generieren.

VERWENDUNG

- Modellierung glatter Kurven und Oberflächen.
- Animationen in der Computergrafik.
- Rekonstruktion und Approximation in der Bildverarbeitung und 3D-Modellierung.

BEISPIEL - LINEARE INTERPOLATION

Zwei Punkte $P_0(x_0, y_0)$ und $P_1(x_1, y_1)$. Die lineare Interpolation zwischen ihnen lautet:

$$P(t) = P_0 \cdot (1 - t) + P_1 \cdot t, \quad t \in [0, 1]$$

2.1.9 Splines und NURBS (Interpolationstechniken)

SPLINES sind Kurven, definiert durch Kontrollpunkte.

- Interpolierende Splines: Kurve verläuft exakt durch Kontrollpunkte.
- Approximierende Splines (z.B. B-Splines): Kurve nähert sich den Kontrollpunkten an.

NURBS (Non-Uniform Rational B-Splines):

- Spezialform von B-Splines im homogenen Koordinatensystem.
- Ermöglichen präzise Darstellung komplexer Kurven und Flächen (z.B. exakte Kreise).
- Häufig genutzt im CAD-Bereich.

BEISPIEL

Exakte Modellierung einer Automobil-Karosserie oder Flugzeugoberfläche.

2.2 FLÄCHENMODELLE

Flächenmodelle repräsentieren dreidimensionale Objekte durch ihre äußeren Oberflächen. Sie sind essenziell in der Computergrafik, da sie die Form, das Aussehen und die Oberflächenbeschaffenheit von Objekten definieren. Diese Modelle unterscheiden sich primär nach der Methode, mit der ihre Oberflächen erzeugt werden.

2.2.1 Polygonale Flächen (Facettierung)

Polygonale Flächenmodelle bestehen aus einer Vielzahl einfacher polygonaler Elemente, meist Dreiecken (Triangulation) oder Vierecken (Quadrangulation).

EIGENSCHAFTEN

- Einfache mathematische Struktur
- Schnelle Berechnung und Transformation (Rendering, Animation)
- Breite Unterstützung durch Grafikhardware

TYPISCHE ANWENDUNG

- Echtzeitgrafik (Videospiele, Virtual Reality)
- Animationen und Visualisierungen

2.2.2 Algorithmische Flächen (Parametrische Flächen)

Algorithmische Flächenmodelle sind explizit mathematisch definierte Oberflächen, beschrieben durch parametrische Gleichungen.

BEISPIEL

ELLIPSOID $x(u, v) = a \cdot \sin(u) \cos(v)$, $y(u, v) = b \cdot \sin(u) \sin(v)$, $z(u, v) = c \cdot \cos(u)$

KLEINSCHER FLASCHE komplexe Fläche, die nur algorithmisch definiert werden kann:

$x(u, v), y(u, v), z(u, v)$ mit komplexen Funktionen und periodischen Parametern u, v

EIGENSCHAFTEN

- Exakte mathematische Beschreibung
- Hohe Genauigkeit
- Ideal für analytische Anwendungen (CAD, Physikalische Simulationen)

2.2.3 Variationsbasierte Flächen (Minimierungsprobleme)

Diese Flächen entstehen durch mathematische Optimierungsverfahren. Ziel ist meist eine Fläche minimaler Energie oder geringster Abweichung von definierten Kriterien.

BEISPIEL – SIGNED DISTANCE FUNCTION (SDF)

- SDF definiert Oberflächen implizit durch Distanzwerte zu einer Fläche. Punkte auf der Oberfläche haben den Wert 0.
- Beliebt in Echtzeit-Rendering (Shader) und 3D-Rekonstruktion.
- Formel SDF (Kugel): $SDF_{\text{Kugel}}(p) = \|p - c\| - r$
Wobei p ein Punkt im Raum, c Mittelpunkt der Kugel und r Radius ist.

EIGENSCHAFTEN

- Implizite Definition erlaubt flexible Operationen (Vereinigung, Schnitt, Differenz).
- Weit verbreitet in computergenerierten Grafiken und modernen Schriftarten (3D Fonts).

ANWENDUNGEN

- Realistische Rendering-Techniken (Ray Marching).
- Computergestützte 3D-Rekonstruktion.

2.2.4 Ruled Surfaces (Regelflächen)

Regelflächen entstehen durch die Bewegung einer Linie (Gerade) im Raum entlang einer festgelegten Bahn.

BEISPIEL – GAUDIS TECHNIK (SAGRADA FAMILIA)

- Antoni Gaudi verwendete für die Modellierung architektonischer Formen hängende Modelle (Kettenmodelle). Durch Umkehrung der Gravitation (hängende Kette -> stehende Bögen) entstanden Regelflächen.

EIGENSCHAFTEN

- Einfache Herstellung und mathematische Klarheit
- Architektur, Design und Ingenieurwesen nutzen dies zur Herstellung komplexer und dennoch stabiler Strukturen.

2.2.5 Freiformflächen (Freiformmodellierung)

siehe 2.1.9 Splines und NURBS

2.3 MESHES

In der Computergrafik stellen Meshes die wichtigste Form dar, um komplexe dreidimensionale Formen zu modellieren und darzustellen. Besonders verbreitet sind dabei Dreiecksmeshes, also Netze, die sich aus einer Vielzahl kleiner Dreiecke zusammensetzen. Jedes Dreieck ist definiert durch drei Punkte (Vertices), die durch Kanten (Edges) verbunden sind und eine Fläche (Face) aufspannen.

Der Grund für die Dominanz von Dreiecken liegt in ihrer mathematischen Stabilität: Drei Punkte liegen immer exakt in einer Ebene – anders als bei Vierecken oder anderen Polygonen, die leicht verzogen sein können. Daher können Dreiecke zuverlässig zur Annäherung beliebiger Formen verwendet werden. Komplexe Oberflächen, wie die eines Würfels oder einer Kugel, können durch Triangulation – also das Aufteilen von Flächen in Dreiecke – nachgebildet werden.

Meshes ermöglichen dabei sowohl exakte als auch approximative Modellierungen. Indem viele kleine Dreiecke eingesetzt werden, kann eine Oberfläche nahezu glatt erscheinen, obwohl sie in Wirklichkeit aus vielen flachen Einzelteilen besteht. Die verwendeten Normalenvektoren (senkrechte Vektoren zu den Flächen) sind dabei zentral für Lichtreflexionen und realistische Darstellungen. Allerdings sind die Normalen in einem Mesh nicht kontinuierlich wie bei einer echten glatten Fläche, sondern stückweise konstant innerhalb eines Dreiecks.

STRUKTUR VON MESHES

Ein Mesh wird typischerweise in zwei Tabellen organisiert:

- Eine Vertex-Tabelle, die die Positionen aller Punkte enthält.
- Eine Face-Tabelle, die beschreibt, welche drei Vertices jeweils ein Dreieck bilden.

Diese Trennung zwischen Geometrie (Vertex-Positionen) und Topologie (Verbindungen der Vertices) erlaubt flexible Operationen wie das Verschieben einzelner Punkte ohne Veränderung der Verbindungsstruktur.

Die Orientierung der Dreiecke – festgelegt durch die Laufrichtung der Vertices (Winding Order) – bestimmt dabei, was als Vorder- oder Rückseite einer Fläche interpretiert wird. Das Kreuzprodukt zweier Kantenvektoren liefert die Flächennormale und spielt eine entscheidende Rolle bei Lichtberechnungen.

TYPEN UND OPTIMIERUNGEN VON MESHES

Um die Effizienz zu erhöhen, existieren verschiedene Varianten der Speicherung:

TRIANGLE SOUP Jedes Dreieck ist unabhängig gespeichert, was Speicherplatz verschwendet und Topologieinformationen verliert. (Abb. 9)

INDEXED FACESETS Vertices werden nur einmal gespeichert, und Dreiecke referenzieren diese über Indizes – spart viel Speicher. (Abb. 10)

TRIANGLE STRIPS UND TRIANGLE FANS Reihen von Dreiecken teilen sich Kanten, was die Datenmenge weiter reduziert. (Abb. 11, Abb. 12)

Subdivision ist eine Technik, bei der ein Mesh durch Teilung von Dreiecken verfeinert wird, um glattere Oberflächen zu erzeugen. Im Gegensatz dazu steht die Simplifikation, bei der komplexe Meshes reduziert werden, um eine effizientere Darstellung bei kleiner Auflösung zu ermöglichen.

ERWEITERTE ANWENDUNGEN UND GRENZEN

Neben klassischen Oberflächenmodellen gibt es auch 1D-Meshes (Polylines) in der Ebene, die aus Punkten und Linien bestehen. Diese Konzepte führen direkt zu grundlegenden Begriffen wie Manifolds – Strukturen, bei denen jede lokale Umgebung „flach“ wirkt und die zentrale Voraussetzung für viele stabile Mesh-Verfahren sind.

ZUSAMMENGEFASST BIETEN MESHES eine hohe Flexibilität in der Modellierung und Darstellung von 3D-Objekten, effiziente Speicher- und Rechenoptimierungen durch den Einsatz von Indizierung und Kompression sowie mathematische Sicherheit aufgrund der Verwendung von stabilen Dreiecksstrukturen. Allerdings sind Meshes nicht für alle Objekte geeignet: Strukturen mit feinem Detail auf jeder Skala (z.B. Haare oder gebrochene Oberflächen wie Marmor) lassen sich nur schwer sinnvoll als Mesh darstellen.

2.4 VOLUMENMODELLE

Während viele klassische 3D-Darstellungen auf Oberflächenmodellen basieren, gibt es Situationen, in denen eine Volumenbeschreibung nötig oder vorteilhaft ist. Volumenmodelle (auch als Solid Models bezeichnet) repräsentieren nicht nur die äußere Hülle eines Objekts, sondern das gesamte Volumen — inklusive seines Inneren.

DIESE DARSTELLUNGEN ERMÖGLICHEN:

- Realistische Simulationen von physikalischen Eigenschaften (z.B. Wärmeleitung, Druckverteilung, Materialverformungen)
- Darstellung transparenter oder transluzenter Objekte (wie Flüssigkeiten oder Gewebe)
- Genauere Berechnungen für Effekte wie Lichtbrechung oder Streuung.

TYPEN UND METHODEN DER VOLUMENMODELLIERUNG

1. BREP (Boundary Representation)

- Klassische Methode auch für Oberflächenmodelle: Kanten und Flächen definieren das Objekt.
- Erweiterbar auf Volumen, wenn Topologie vollständig angegeben wird (z.B. manifold solids).
- Typisch bei CAD-Anwendungen.

2. Constructive Solid Geometry (CSG)

- Komplexe Volumen werden durch Kombination einfacher geometrischer Primitiven erstellt.
- Verwendet boolesche Operationen:
 - Union (\cup) – Vereinigung
 - Difference ($-$) Subtraktion
 - Intersection (\cap) Schnittmenge

3. Voxel-Modelle

- Voxel = Volume Element (analog zu Pixel in 2D).
- Raum wird in ein regelmäßiges 3D-Gitter unterteilt; jeder Voxel speichert Materialeigenschaften oder Dichte.
- Anwendung in: Medizinischer Bildgebung (CT, MRT), Fluid- und Strömungssimulation, Spieleentwicklung (z.B. Minecraft: bewusst grobe Auflösung für Stil und Performance)
- Vorteile: Direkte Simulation physikalischer Prozesse, einfache Speicherstruktur (3D-Array), leicht komprimierbar bei großen homogenen Regionen

- Nachteile: Hoher Speicherbedarf bei hoher Auflösung, begrenzte Detaildarstellung

4. Octree

- Hierarchische Strukturierung von Voxelräumen:
 - Jeder Raumabschnitt wird rekursiv in 8 Unterabschnitte (Oktanten) aufgeteilt.
- Vorteil: Effiziente Speicherung großer leerer Bereiche.
- Grundlage für Methoden wie Marching Cubes für Rekonstruktion glatter Oberflächen aus diskreten Daten

5. Cell Decomposition

- Zerlegung eines Raums in Zellen identischer Topologie, aber unterschiedlicher Geometrie.
- Häufig in Finite-Elemente-Methoden (FEM) genutzt, etwa für die Simulation von Kräften und Temperaturfeldern.

3 | TRANSFORMATION

3.1 EINFÜHRUNG

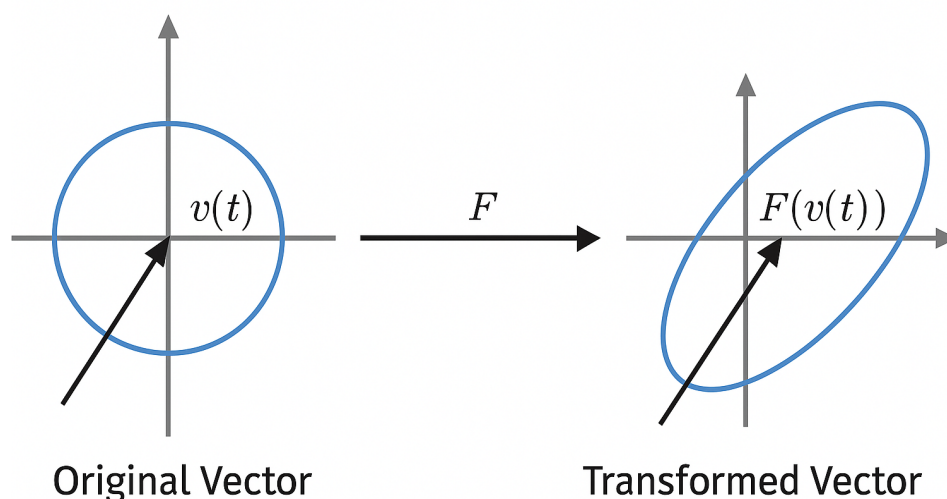
In der Computergrafik ist die Fähigkeit, Objekte im Raum zu bewegen, zu skalieren oder zu drehen, eine Grundvoraussetzung. Diese Vorgänge werden als Transformationen bezeichnet. Transformationen sind essenziell, um Modelle in eine Szene einzufügen, sie richtig auszurichten oder Animationen zu erzeugen.

Man stelle sich vor, wir schieben ein Schachbrett auf einem Tisch (Translation), drehen es leicht, um eine bessere Ausrichtung zu erreichen (Rotation) und vergrößern oder verkleinern es (Skalierung), um es an den Platz anzupassen.

In der Computergrafik vollziehen wir genau diese Operationen mathematisch auf den Koordinaten der Objektpunkte.

BEISPIELE - IN DER COMPUTERGRAFIK

- Das Platzieren eines 3D-Charakters in einer virtuellen Welt
- Das Animieren einer Kamera entlang eines Pfades
- Das Skalieren von Gebäudemodellen auf eine einheitliche Größe
- Das Rotieren von Objekten für eine realistische Darstellung aus unterschiedlichen Perspektiven



Transformationen erlauben es, Geometrie und Topologie von der eigentlichen Manipulation zu trennen. Man verändert nicht das Objekt selbst, sondern nur seine Darstellung im Raum.

3.2 EINSCHUB: MATHEMATISCHE ABBILDUNGEN

Eine Abbildung oder Funktion ist eine Zuordnungsvorschrift, die jedem Element eines Ausgangsraums (Definitionsbereich) genau ein Element eines Zielraums (Wertebereich) zuordnet:

$$f: X \rightarrow Y$$

In unserem Kontext:

- Abbildungen verknüpfen Punkte oder Vektoren mit anderen Punkten oder Vektoren.
- Abbildungen können die Struktur (z.B. Geradheit, Längenverhältnisse) erhalten oder verändern.

Ein spezieller Typ sind lineare Abbildungen, die die Vektorstruktur erhalten und sich gut durch Matrizen darstellen lassen.

BEISPIEL – LINEARE ABBILDUNG VOM \mathbb{R}^2 IN DEN \mathbb{R}^3

Um den Übergang von allgemeinen linearen Abbildungen zu Transformationen in der Computergrafik vorzubereiten, betrachten wir eine lineare Transformation, die einen Vektor aus dem \mathbb{R}^2 in den \mathbb{R}^3 überführt.

Wir definieren eine lineare Abbildung durch eine 3×2 -Matrix A :

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix}, \quad \vec{v} = \begin{pmatrix} x \\ y \end{pmatrix}$$

Die Abbildung $A\vec{v}$ ergibt dann:

$$A \cdot \vec{v} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \\ y \\ x + y \end{pmatrix} \in \mathbb{R}^3$$

Interpretation

- Die x - und y -Komponenten bleiben erhalten.
- Die neue z -Koordinate ergibt sich aus $x + y$.
- Eine Fläche im \mathbb{R}^2 (z. B. ein Quadrat) wird in eine geneigte Fläche im Raum abgebildet.

Bedeutung in der Computergrafik

- Solche Transformationen erlauben es, 2D-Objekte kontrolliert in den 3D-Raum zu überführen.
- Dies ist beispielsweise nützlich für perspektivische Effekte, dynamische Geometrie oder die Positionierung von UI-Elementen im Raum.
- Das Beispiel demonstriert, dass jede lineare Transformation durch einfache *Matrixmultiplikation* beschrieben werden kann.

3.3 LINEARE TRANSFORMATIONEN

Eine Transformation F eines Vektorraums ist **linear**, wenn sie die Struktur des Raums erhält, d. h. insbesondere die Vektoraddition und Skalarmultiplikation. Formal gilt:

$$F(\alpha \vec{u} + \beta \vec{v}) = \alpha F(\vec{u}) + \beta F(\vec{v}) \quad \text{für alle } \alpha, \beta \in \mathbb{R}, \vec{u}, \vec{v} \in V$$

Diese Eigenschaft nennt man das **Superpositionsprinzip**. Es erlaubt, jede lineare Transformation durch eine Matrixdarstellung zu beschreiben.

Beispiel: Skalierung im \mathbb{R}^2

Wir definieren die lineare Abbildung F durch eine Diagonalmatrix:

$$F(\vec{v}) = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} ax \\ by \end{pmatrix}$$

- Wenn $a = b = 2$, ergibt sich eine **uniforme Skalierung** um den Faktor 2.
- Wenn $a = 2, b = 1$, ergibt sich eine **nicht-uniforme Skalierung**, die nur in x -Richtung dehnt.

Diese Transformation ist linear, da sie das Superpositionsprinzip erfüllt:

$$F(\alpha \vec{u} + \beta \vec{v}) = \alpha F(\vec{u}) + \beta F(\vec{v})$$

Nicht-lineares Beispiel: Translation

Eine Translation $F(\vec{v}) = \vec{v} + \vec{u}$ verschiebt alle Punkte um einen festen Vektor \vec{u} . Diese Transformation ist *nicht linear*, da:

$$F(\alpha \vec{v}) = \alpha \vec{v} + \vec{u} \neq \alpha(\vec{v} + \vec{u}) = \alpha F(\vec{v})$$

Sie verletzt somit das Superpositionsprinzip und kann nicht durch eine einfache 2×2 -Matrix dargestellt werden.

Lineare Abbildungen als Matrizenoperation

Jede lineare Abbildung $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ kann durch eine Matrix $A \in \mathbb{R}^{m \times n}$ beschrieben werden:

$$F(\vec{v}) = A \cdot \vec{v}$$

Beispiel einer linearen Abbildung $F : \mathbb{R}^2 \rightarrow \mathbb{R}^2$ mit Rotation um den Winkel θ :

$$A = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}, \quad F(\vec{v}) = A \cdot \vec{v}$$

Diese Matrix dreht jeden Vektor gegen den Uhrzeigersinn um den Winkel θ .

Lineare Transformationen im Kontext parametrischer Flächen

Gegeben sei ein parametrisierter Kreis im \mathbb{R}^2 :

$$\vec{v}(t) = \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix}$$

Eine lineare Transformation kann hier z. B. eine Dehnung des Kreises zu einer Ellipse darstellen:

$$F(\vec{v}) = \begin{pmatrix} a & 0 \\ 0 & b \end{pmatrix} \begin{pmatrix} \cos(t) \\ \sin(t) \end{pmatrix} = \begin{pmatrix} a \cos(t) \\ b \sin(t) \end{pmatrix}$$

Für $a = 1, b = 2$ entsteht eine Ellipse mit doppelter Ausdehnung in y-Richtung.

Geometrische Interpretation

- **Lineare Transformationen** bewahren den Ursprung und führen zu Streckung, Spiegelung, Scherung oder Rotation.
- **Nicht-lineare Transformationen** verschieben den Ursprung oder verzerren das Koordinatensystem, z. B. durch Translation.

Lineare Transformationen sind die Grundlage für alle affine Transformationen in der Computergrafik. Durch sie lassen sich geometrische Objekte effizient manipulieren und in verschiedene Koordinatensysteme überführen.

3.4 TRANSFORMATION IM \mathbb{R}^2

3.4.1 2D-Translation

BESCHREIBUNG

Eine Translation verschiebt ein Objekt im zweidimensionalen Raum \mathbb{R}^2 um einen festen Vektor \mathbf{t} . Dabei bleiben Form, Größe und Orientierung des Objekts unverändert.

GEGEBEN

Ein Punkt $\mathbf{p} = \begin{pmatrix} x \\ y \end{pmatrix} \in \mathbb{R}^2$ und ein Translationsvektor $\mathbf{t} = \begin{pmatrix} t_x \\ t_y \end{pmatrix} \in \mathbb{R}^2$.

NEUE POSITION

Durch die Translation ergibt sich der verschobene Punkt $\tilde{\mathbf{p}}$ als:

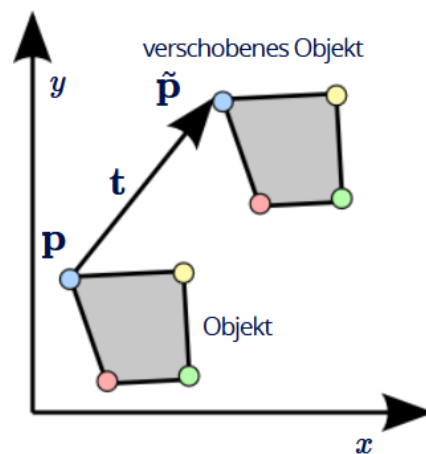
$$\tilde{\mathbf{p}} = \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \begin{pmatrix} x + t_x \\ y + t_y \end{pmatrix} = \mathbf{p} + \mathbf{t}$$

BEISPIEL

Ein Punkt $\mathbf{p} = \begin{pmatrix} 2 \\ 3 \end{pmatrix}$ wird um den Vektor $\mathbf{t} = \begin{pmatrix} 4 \\ -1 \end{pmatrix}$ verschoben.

$$\tilde{\mathbf{p}} = \begin{pmatrix} 2 \\ 3 \end{pmatrix} + \begin{pmatrix} 4 \\ -1 \end{pmatrix} = \begin{pmatrix} 6 \\ 2 \end{pmatrix}$$

Der Punkt wurde somit 4 Einheiten nach rechts und 1 Einheit nach unten verschoben.



3.4.2 Uniforme Skalierung

Die **uniforme Skalierung** ist eine spezielle lineare Transformation, bei der ein Objekt gleichmäßig in alle Richtungen (typischerweise x- und y-Richtung) vergrößert oder verkleinert wird. Mathematisch geschieht dies durch Multiplikation mit einem konstanten Skalar $s > 0$.

MATHEMATISCHE BESCHREIBUNG

Gegeben sei ein Punkt (Ortsvektor) $\vec{p} = \begin{pmatrix} x \\ y \end{pmatrix}$. Die uniforme Skalierung mit dem Skalierungsfaktor s ergibt den Bildpunkt \vec{p}' als:

$$\vec{p}' = s \cdot \vec{p} = s \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} s \cdot x \\ s \cdot y \end{pmatrix}$$

Alternativ formuliert als Matrixtransformation:

$$\vec{p}' = \begin{pmatrix} s & 0 \\ 0 & s \end{pmatrix} \cdot \vec{p}$$

Diese Matrix streckt jeden Vektor im \mathbb{R}^2 um den Faktor s vom Ursprung aus.

BEISPIEL - VERGRÖßERUNG EINER FIGUR

Ein Quadrat mit der Seitenlänge 1 im Koordinatenursprung wird durch eine uniforme Skalierung mit $s = 2$ zu einem Quadrat der Seitenlänge 2. Dabei bleiben die Winkel und die relative Anordnung der Punkte erhalten – die Figur bleibt ähnlich, nur größer.

BEISPIEL - VERGRÖßERUNG EINES PUNKTES IM RAUM

Ein Punkt $\vec{p} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ wird bei $s = 3$ auf

$$\vec{p}' = \begin{pmatrix} 6 \\ 3 \end{pmatrix}$$

abgebildet. Der Punkt entfernt sich also vom Ursprung, wobei seine Richtung erhalten bleibt.

BEISPIEL - ANWENDUNG AUF EIN 2D-DREIECK

Ein Dreieck mit den Eckpunkten $(0,0)$, $(1,0)$ und $(0,1)$ wird bei $s = 2$ auf die Punkte $(0,0)$, $(2,0)$ und $(0,2)$ transformiert. Das resultierende Dreieck ist doppelt so groß, liegt aber weiterhin an der gleichen Position relativ zum Ursprung.

Eigenschaften

- Die uniforme Skalierung ist eine **lineare Transformation**.
- Das Objekt wird in **allen Richtungen gleichmäßig** skaliert.
- **Winkel und Formverhältnisse** bleiben erhalten – Figuren bleiben ähnlich.
- Der **Ursprung** bleibt unverändert.

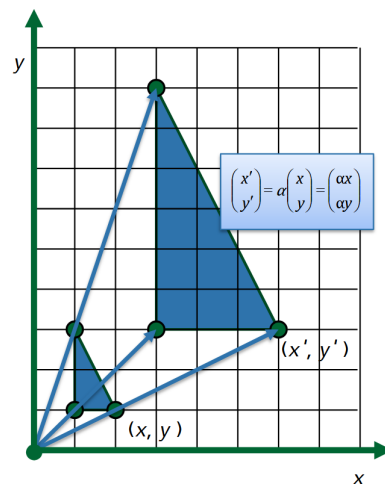


Abbildung 1: Uniforme Skalierung

3.4.3 Nicht-uniforme Skalierung

Die **nicht-uniforme Skalierung** beschreibt eine Transformation, bei der die Skalierung in den einzelnen Raumrichtungen unterschiedlich ist. So kann ein Objekt z.B. nur in x -Richtung gestreckt oder in y -Richtung gestaucht werden. Diese Art der Skalierung verändert die Proportionen eines Objekts.

MATHEMATISCHE BESCHREIBUNG

Gegeben sei ein Ortsvektor $\vec{p} = \begin{pmatrix} x \\ y \end{pmatrix}$ und zwei Skalierungsfaktoren s_x und s_y für die jeweiligen Achsen. Dann ergibt sich der Bildvektor durch:

$$\vec{p} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} s_x \cdot x \\ s_y \cdot y \end{pmatrix}$$

BEISPIEL – STRECKUNG NUR IN x -RICHTUNG

Ein Punkt $\vec{p} = \begin{pmatrix} 2 \\ 1 \end{pmatrix}$ wird mit $s_x = 3$, $s_y = 1$ transformiert:

$$\vec{p} = \begin{pmatrix} 6 \\ 1 \end{pmatrix}$$

Das Objekt wird in x -Richtung gedehnt, in y -Richtung bleibt es unverändert.

BEISPIEL – DEHNUNG UND STAUCHUNG

Ein Dreieck mit Punkten $(0,0)$, $(1,0)$ und $(0,1)$ wird mit $s_x = 2$ und $s_y = 0.5$ skaliert. Die neuen Punkte sind $(0,0)$, $(2,0)$ und $(0,0.5)$. Die Form wird verzerrt, da x und y unterschiedlich verändert werden.

Eigenschaften

- Auch die nicht-uniforme Skalierung ist eine **lineare Transformation**.
- Der Ursprung bleibt erhalten, da keine Verschiebung erfolgt.
- **Verhältnisse** zwischen den Achsen ändern sich – es entstehen gestreckte oder gestauchte Objekte.
- **Winkel** werden im Allgemeinen nicht bewahrt – insbesondere rechtwinklige Formen werden verzerrt.

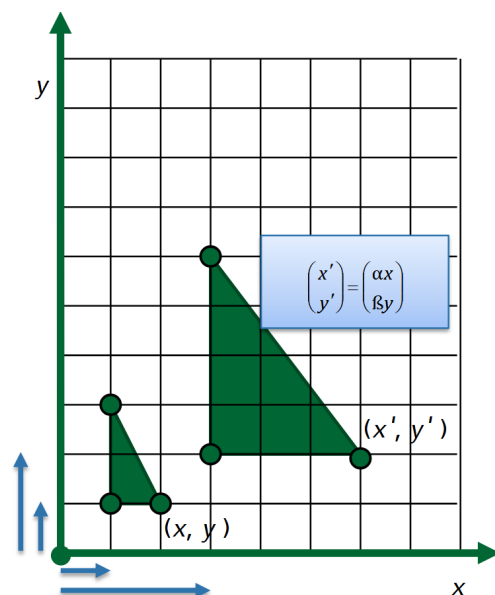


Abbildung 2: Nicht-uniforme Skalierung

3.4.4 2D-Rotation

Die **Rotation** ist eine lineare Transformation, bei der ein Punkt oder ein Objekt in der Ebene um einen festen Winkel α um den Koordinatenursprung gedreht wird. Diese Transformation ist winkel- und längentreu, d.h. sie verändert weder Abstände zwischen Punkten noch die Orientierung von Objekten – lediglich deren Richtung.

MATHEMATISCHE BESCHREIBUNG

Ein Punkt $\vec{v} = \begin{pmatrix} x \\ y \end{pmatrix}$ wird bei einer Rotation um den Winkel α gemäß folgender Matrix transformiert:

$$R(\alpha) \cdot \vec{v} = \begin{pmatrix} \cos(\alpha) & -\sin(\alpha) \\ \sin(\alpha) & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} x \cdot \cos(\alpha) - y \cdot \sin(\alpha) \\ x \cdot \sin(\alpha) + y \cdot \cos(\alpha) \end{pmatrix}$$

Der Winkel α wird in der Regel gegen den Uhrzeigersinn gemessen. Negative Werte führen zu einer Drehung im Uhrzeigersinn.

GEOMETRISCHE BEDEUTUNG

Die Basisvektoren \vec{e}_x und \vec{e}_y des kartesischen Koordinatensystems werden bei einer Rotation ebenfalls transformiert. Die neuen Richtungen lassen sich als Spalten der Rotationsmatrix interpretieren:

$$\tilde{\vec{b}}_x = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}, \quad \tilde{\vec{b}}_y = \begin{pmatrix} -\sin(\alpha) \\ \cos(\alpha) \end{pmatrix}$$

Somit entspricht eine Rotation auch einer Basisänderung, wobei die gedrehten Achsen den neuen Raum aufspannen.

EIGENSCHAFTEN

- Die Rotation ist eine **lineare Transformation**.
- Der **Ursprung bleibt invariant** – er bewegt sich nicht.
- **Längen und Winkel bleiben erhalten** (isometrisch).
- Die Rotationsmatrix ist **orthogonal**, d.h.:

$$R^{-1}(\alpha) = R^T(\alpha) = R(-\alpha)$$

- Die Determinante der Rotationsmatrix ist:

$$\det(R(\alpha)) = \cos^2(\alpha) + \sin^2(\alpha) = 1$$

→ Die Fläche bleibt unter der Transformation erhalten.

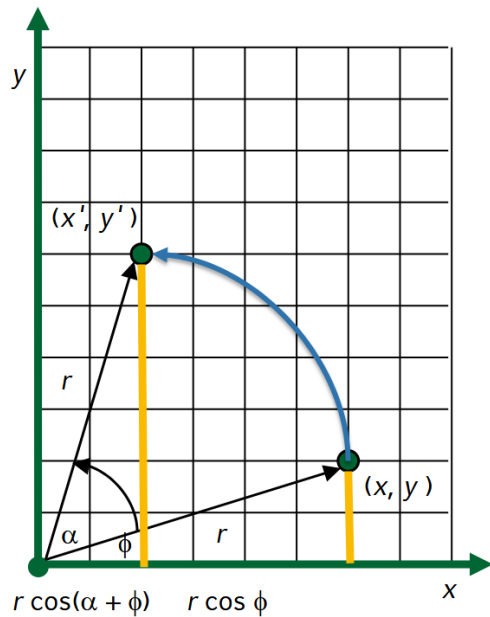


Abbildung 3: Rotation Punkt

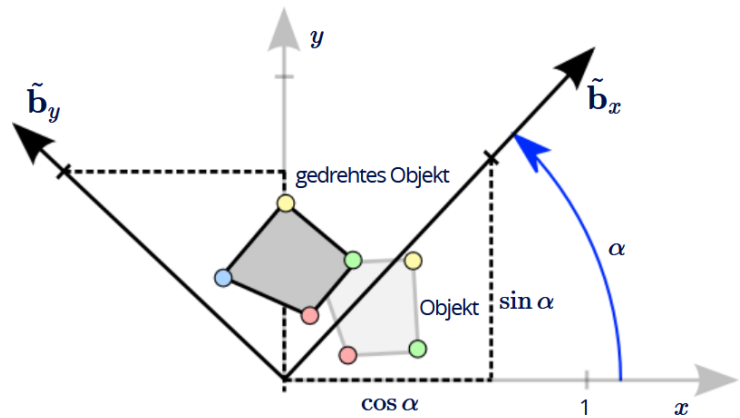


Abbildung 4: Rotation Fläche

Herleitung der Rotationsformel im \mathbb{R}^2

Um die Rotationsmatrix herzuleiten, betrachten wir einen Punkt P in der Ebene, gegeben durch kartesische Koordinaten (x, y) sowie seine Darstellung in Polarkoordinaten:

$$x = r \cos(\varphi), \quad y = r \sin(\varphi)$$

DREHUNG UM DEN URSPRUNG

Wird der Punkt P um einen Winkel α gegen den Uhrzeigersinn um den Ursprung gedreht, so ergibt sich der neue Winkel $\varphi + \alpha$. Die neuen Koordinaten (x', y') des gedrehten Punktes lauten:

$$x' = r \cdot \cos(\varphi + \alpha)$$

$$y' = r \cdot \sin(\varphi + \alpha)$$

VERWENDUNG TRIGONOMETRISCHER ADDITIONSTHEOREME

$$x' = r \cdot \cos(\varphi) \cdot \cos(\alpha) - r \cdot \sin(\varphi) \cdot \sin(\alpha)$$

$$y' = r \cdot \cos(\varphi) \cdot \sin(\alpha) + r \cdot \sin(\varphi) \cdot \cos(\alpha)$$

Nun setzen wir die ursprünglichen Ausdrücke für x und y aus den Polarkoordinaten ein:

$$x' = x \cdot \cos(\alpha) - y \cdot \sin(\alpha)$$

$$y' = x \cdot \sin(\alpha) + y \cdot \cos(\alpha)$$

MATRIXDARSTELLUNG DER ROTATION

Diese Gleichungen lassen sich elegant in Matrixform schreiben:

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

Diese Matrix nennt man **Rotationsmatrix** $R(\alpha)$ und sie beschreibt eine Drehung um den Ursprung im \mathbb{R}^2 um den Winkel α gegen den Uhrzeigersinn.

EIGENSCHAFTEN DER ROTATIONSMATRIX

- Orthogonal: $R(\alpha)^{-1} = R(\alpha)^T = R(-\alpha)$
- Determinante: $\det(R(\alpha)) = 1$
- Flächen- und längentreu (isometrisch)

3.4.5 Kritische Rückschau

In den vorangegangenen Abschnitten wurden grundlegende Transformationen behandelt:

- **Translation:** Verschiebung durch Addition eines Vektors
- **Skalierung:** Multiplikation mit einem Skalierungsfaktor (diagonale Matrix)
- **Rotation:** Drehung mittels Multiplikation mit einer Rotationsmatrix

PROBLEM: UNEINHEITLICHE DARSTELLUNG

Diese Transformationen lassen sich derzeit auf unterschiedliche Weise beschreiben:

- Translation: $\vec{p}' = \vec{p} + \vec{t}$ (Addition eines Vektors)
- Skalierung, Rotation: $\vec{p}' = M \cdot \vec{p}$ (Matrixmultiplikation)

Diese Differenz wird insbesondere dann zum Problem, wenn mehrere Transformationen kombiniert werden sollen. Eine lineare Matrix lässt sich nicht direkt mit einer Translation additiv verknüpfen. Es existiert keine einheitliche Rechenregel wie:

$$T \cdot (M \cdot \vec{p}) \neq \text{eine Matrix} \cdot \vec{p}$$

ZUSAMMENFASSUNG

- **Keine einheitliche mathematische Struktur:** Nicht alle Transformationen sind linear.
- **Zusammensetzen von Transformationen wird kompliziert:** Rotation gefolgt von Translation benötigt unterschiedliche Rechenregeln.
- **In der Praxis hinderlich:** Besonders in Computergrafik, Robotik oder CAD-Systemen, wo Transformationen ständig verkettet werden.

Lineare Transformationen wie Rotation und Skalierung lassen sich durch 2×2 -Matrizen darstellen. Punkte werden dabei als Vektoren im \mathbb{R}^2 interpretiert und durch Matrix-Vektor-Multiplikation transformiert. Translation hingegen erforderte bislang eine separate Vektoraddition und ließ sich nicht durch eine reine Matrixoperation ausdrücken.

ZIEL

Gesucht ist ein Modell, das:

- alle Transformationen einheitlich durch Matrizen darstellt,
- das Zusammensetzen (Verkettung) von Transformationen über einfache Matrixmultiplikation erlaubt,
- den Ursprung korrekt mitberücksichtigt,
- Translationen korrekt integriert.

LÖSUNG: HOMOGENE KOORDINATEN

Die Einführung **homogener Koordinaten** erweitert den euklidischen Raum \mathbb{R}^n um eine zusätzliche Dimension. Dadurch lassen sich nun auch Translationen — und damit sämtliche Affintransformationen — **einheitlich als lineare Matrixoperation** darstellen.

Damit wird es möglich, komplexe Transformationen (z. B. Rotation + Skalierung + Translation) elegant und effizient über **Matrixmultiplikation** zu kombinieren. Diese Vereinheitlichung ist die Grundlage für moderne Grafiksysteme, Transformationen in der Robotik oder rechnergestützte Geometrieverarbeitung.

3.4.6 Einführung homogener Koordinaten

Um auch Translationen in eine einheitliche mathematische Darstellung mittels Matrizenoperationen zu integrieren, erweitern wir das Koordinatensystem um eine zusätzliche Dimension. Ein Punkt $(x, y) \in \mathbb{R}^2$ wird durch homogene Koordinaten als Tripel

$$\mathbf{p}_{\text{hom}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \quad \text{im } \mathbb{R}^3 \text{ dargestellt.}$$

GRUNDIDEE

Durch die zusätzliche dritte Komponente w kann ein Punkt (x, y) in der Ebene durch

$$\begin{pmatrix} x \cdot w \\ y \cdot w \\ w \end{pmatrix} \text{ beschrieben werden.}$$

Wählt man $w = 1$, so ergibt sich die sogenannte **normalisierte Darstellung**.

Wichtig: Jeder Punkt in kartesischen Koordinaten hat unendlich viele äquivalente Darstellungen in homogenen Koordinaten:

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} \lambda x \\ \lambda y \\ \lambda \end{pmatrix} \quad \text{für } \lambda \neq 0$$

MOTIVATION: VEREINHEITLICHUNG DURCH MATRIZENMULTIPLIKATION

Im bisherigen Modell mussten Translationen durch separate Vektoraddition behandelt werden:

$$\tilde{\mathbf{p}} = \mathbf{M} \cdot \mathbf{p} + \mathbf{t}$$

Dies ist aus Sicht der Mathematik keine lineare Abbildung. In homogenen Koordinaten kann jedoch jede affine Transformation — einschließlich Translation — durch eine **einzige Matrix** realisiert werden:

$$\tilde{\mathbf{p}}_{\text{hom}} = \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} m_{11} & m_{12} & t_x \\ m_{21} & m_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}}_{\mathbf{T}} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Diese Darstellung erlaubt:

- die **einheitliche Beschreibung** aller Transformationen (Skalierung, Rotation, Translation, Scherung, Spiegelung),
- die **Verkettung beliebiger Transformationen** durch einfache Matrixmultiplikation,
- eine direkte Implementierung in Grafikpipelines, Robotik, CAD-Systemen u.v.m.

UMRECHNUNG: HOMOGEN \rightarrow KARTESISCH

Ist ein Punkt $\tilde{\mathbf{p}} = \begin{pmatrix} u \\ v \\ w \end{pmatrix}$ gegeben, so ergibt sich der entsprechende Punkt im \mathbb{R}^2 durch:

$$\mathbf{p} = \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u/w \\ v/w \end{pmatrix} \quad \text{für } w \neq 0$$

Dabei gilt: Die Multiplikation des homogenen Vektors mit einem Skalar $\lambda \neq 0$ ändert das Ergebnis in kartesischen Koordinaten nicht.

ZUSAMMENFASSUNG

Die Verwendung homogener Koordinaten schafft die Grundlage für eine einheitliche Darstellung und Verarbeitung aller geometrischen Transformationen in der Ebene — sowohl konzeptionell als auch rechentechnisch.

3.4.7 Transformationen in homogenen Koordinaten

Durch die Einführung homogener Koordinaten können alle grundlegenden geometrischen Transformationen — einschließlich Translation — als 3×3 -Matrizen formuliert werden. Dies erlaubt die Darstellung jeder Transformation als Matrix-Vektor-Multiplikation.

TRANSLATION

Vorher: Verschiebung durch Vektoraddition $\vec{p}' = \vec{p} + \vec{t}$

Jetzt: Darstellung als Matrixmultiplikation

$$T_{\text{trans}} = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}, \quad \vec{p}_{\text{hom}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad \vec{p}' = T_{\text{trans}} \cdot \vec{p}_{\text{hom}} = \begin{pmatrix} x + t_x \\ y + t_y \\ 1 \end{pmatrix}$$

UNIFORME SKALIERUNG

Vorher: Skalierung durch Multiplikation mit Skalar

Jetzt: Skalierung entlang beider Achsen durch Diagonalmatrix

$$T_{\text{scale}} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \vec{p}_{\text{hom}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad \vec{p}' = T_{\text{scale}} \cdot \vec{p}_{\text{hom}} = \begin{pmatrix} s_x \cdot x \\ s_y \cdot y \\ 1 \end{pmatrix}$$

Dabei bewirkt:

- $s_x > 1$: Streckung in x-Richtung, $0 < s_x < 1$: Stauchung
- $s_y > 1$: Streckung in y-Richtung, $0 < s_y < 1$: Stauchung
- Bei $s_x = s_y$ spricht man von einer **uniformen Skalierung**

ROTATION

Vorher: Rotation über 2×2 -Matrix oder trigonometrische Gleichungen

Jetzt: als 3×3 -Rotationsmatrix mit Ursprung als Zentrum

$$T_{\text{rot}} = \begin{pmatrix} \cos \alpha & -\sin \alpha & 0 \\ \sin \alpha & \cos \alpha & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \vec{p}_{\text{hom}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, \quad \vec{p}' = T_{\text{rot}} \cdot \vec{p}_{\text{hom}} = \begin{pmatrix} x \cdot \cos \alpha - y \cdot \sin \alpha \\ x \cdot \sin \alpha + y \cdot \cos \alpha \\ 1 \end{pmatrix}$$

Dabei gilt:

- Positive Winkel α bewirken eine Drehung **gegen den Uhrzeigersinn**
- Der **Ursprung** bleibt bei der Rotation unverändert
- Die Form des Objekts bleibt erhalten (Längen und Winkel sind invariant)

SCHERUNG

Die Scherung ist eine affine Transformation, bei der eine Koordinatenachse in Richtung der anderen „verschoben“ wird. Dabei bleiben Linien parallel zu einer Achse erhalten, während sich die relative Lage in der anderen Richtung ändert.

X-Richtung:

$$T_{\text{shear-x}} = \begin{pmatrix} 1 & k & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad \text{mit } k \in \mathbb{R}$$

Y-Richtung:

$$T_{\text{shear-y}} = \begin{pmatrix} 1 & 0 & 0 \\ k & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Die Anwendung der Transformation auf einen Punkt $\vec{p}_{\text{hom}} = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$ ergibt:

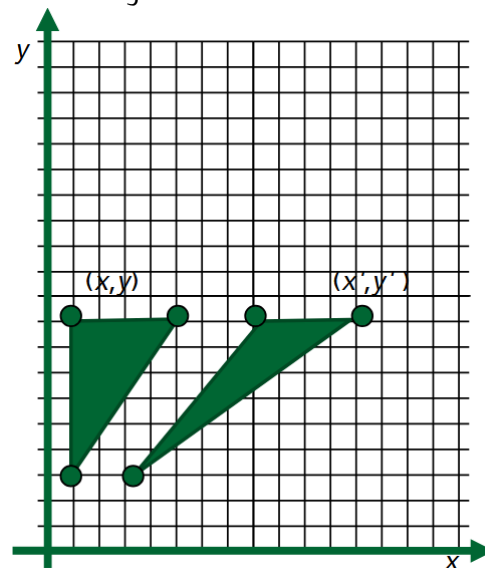


Abbildung 5: Scherung in x- bzw. y-Richtung

- Bei x-Scherung: $x' = x + k \cdot y$
- Bei y-Scherung: $y' = y + k \cdot x$

Eigenschaften:

- Linien bleiben gerade, aber Winkel und Längen werden verändert.
- Die Transformation ist **affin**, aber nicht isometrisch.
- In homogenen Koordinaten vollständig als Matrixmultiplikation darstellbar.

SPIEGELUNG (Reflexion)

Die Spiegelung ist eine Transformation, bei der Punkte an einer bestimmten Achse gespiegelt werden. Diese Transformation ist eine lineare Abbildung mit Determinante -1 und verändert die Orientierung der Objekte.

Spiegelung an der x-Achse:

$$T_{\text{ref-x}} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Spiegelung an der y-Achse:

$$T_{\text{ref-y}} = \begin{pmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

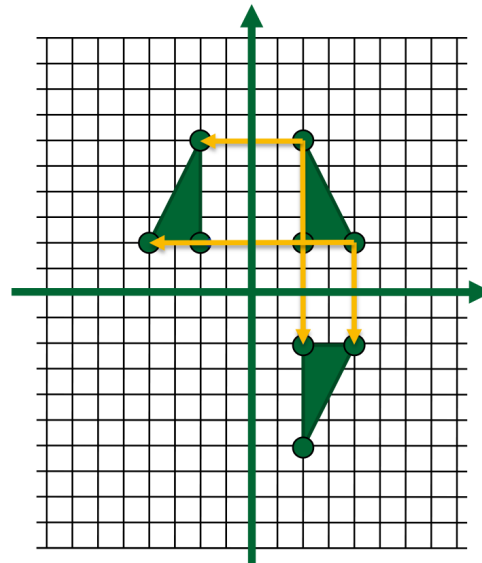


Abbildung 6: Spiegelung in x- bzw. y-Richtung

Die Anwendung erfolgt wie gewohnt durch Multiplikation mit dem homogenen Vektor:

$$\vec{p}' = T_{\text{ref}} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

Eigenschaften:

- Die Form bleibt erhalten, aber die Orientierung kehrt sich um.
- Spiegelungen sind **lineare Transformationen**, aber keine isometrischen Bewegungen im engeren Sinne.
- Spiegelungen entlang anderer Achsen (z. B. beliebiger Winkel) lassen sich durch Kombination aus Rotation, Achs-Spiegelung und Rück-Rotation erzeugen.

ZUSAMMENFASSUNG Allgemeine Transformationsmatrix

Jede Transformation im \mathbb{R}^2 kann im homogenen Raum durch eine Matrix beschrieben werden.

Die einzelnen Einträge kodieren:

- **Skalierung:** Diagonal (a, e)
- **Rotation / Scherung:** Off-Diagonal (d)
- **Translation:** Rechte Spalte (c, f)

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

Abbildung 7: Allgemeine Transformationsmatrix

3.4.8 Verkettung von Transformationen

Werden mehrere geometrische Transformationen nacheinander auf ein Objekt angewendet, so entspricht dies der **Verkettung** bzw. **Kombination** dieser Transformationen. Im homogenen Koordinatensystem erfolgt diese Verkettung durch **Matrixmultiplikation**:

$$\vec{p}' = T_n \cdot T_{n-1} \cdots T_1 \cdot \vec{p}$$

Wichtig: Die Reihenfolge der Multiplikation ist entscheidend! Die Transformation, die **zuerst** angewendet wird, steht **rechts** der Multiplikation.

BEISPIEL Translation gefolgt von Skalierung

$$T_{\text{ges}} = T_{\text{scale}} \cdot T_{\text{trans}} = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} s_x & 0 & s_x \cdot t_x \\ 0 & s_y & s_y \cdot t_y \\ 0 & 0 & 1 \end{pmatrix}$$

EIGENSCHAFTEN

- **Translation ist additiv:** Zwei nacheinander ausgeführte Translationen entsprechen einer Verschiebung um den Summenvektor.
- **Skalierung ist multiplikativ:** Zwei Skalierungen multiplizieren ihre Faktoren.
- **Rotation ist additiv:** Zwei aufeinanderfolgende Rotationen ergeben eine Rotation mit der Winkelsumme.

VERKETTUNG ALLGEMEINER TRANSFORMATIONEN

Eine allgemeine Transformationsmatrix in \mathbb{R}^2 (homogene Koordinaten) beliebig viele solcher Transformationen können durch Matrixmultiplikation verkettet werden. Dies erlaubt es, eine ganze Transformationssequenz in einer einzigen Matrix zusammenzufassen.

INVERSE TRANSFORMATIONEN

Jede Transformation, die durch eine reguläre 3×3 -Matrix T beschrieben wird, besitzt eine **Inverse** T^{-1} , sofern $\det(T) \neq 0$. Damit lässt sich eine Transformation rückgängig machen:

$$\vec{p} = T^{-1} \cdot \vec{p}'$$

HINWEIS: NOTWENDIGKEIT HOMOGENER KOORDINATEN Im euklidischen Raum lässt sich eine Translation nicht als reine Matrixoperation darstellen.

$$F(v) = M \cdot v + u$$

Verkettungen wie $G(F(v)) = M_G(M_F v + u_F) + u_G$ erfordern distributives Ausmultiplizieren. Damit ist keine einfache Darstellung durch eine Matrixmultiplikation möglich.

Nur durch die Erweiterung auf homogene Koordinaten wird die Verkettung linear:

$$T_{\text{ges}} = T_G \cdot T_F \quad \text{und} \quad \vec{p}' = T_{\text{ges}} \cdot \vec{p}_{\text{hom}}$$

3.4.9 Isometrien – Abstands- und Winkeltreue Transformationen

Eine **Isometrie** ist eine Transformation, die den Abstand zwischen allen Punkten erhält:

$$\|f(\vec{u}) - f(\vec{v})\| = \|\vec{u} - \vec{v}\| \quad \text{für alle } \vec{u}, \vec{v}$$

Im \mathbb{R}^2 gehören zu den Isometrien:

- **Translation** – verschiebt alle Punkte gleich weit; Orientierung bleibt erhalten
- **Rotation** – dreht um ein Zentrum; Abstände und Winkel bleiben erhalten
- **Spiegelung** – spiegelt an einer Achse; Abstände erhalten, Orientierung kehrt sich um
- **Gleitspiegelung** – Spiegelung + Translation entlang der Spiegelachse

Alle Isometrien bewahren:

- **Längen**
- **Winkel**
- **Form und Fläche**

Unterscheidung:

- *Orientierungserhaltend*: Translation, Rotation, Gleitspiegelung
- *Orientierungsumkehrend*: Spiegelung

ZUSAMMENFASSUNG

Eigenschaft	Isometrie	Ähnlichkeits- abbildung	Affine Abbildung	Projektiv Abbildung
Abstände erhalten	•	–	–	–
Winkel erhalten	•	•	–	–
Geraden bleiben Geraden	•	•	•	•
Parallele Linien bleiben parallel	•	•	•	–
Verhältnisse auf Linien bleiben gleich	•	•	•	–
Form wird bewahrt	•	•	–	–
Flächeninhalt bleibt erhalten	•	–	–	–

Tabelle 1: Vergleich grundlegender Transformationstypen

Die Isometrien bilden die oberste Klasse in der **Typisierung** homogener Transformationen.

3.4.10 Typisierung von Transformationen

AFFINE TRANSFORMATIONEN

Affine Transformationen sind eine Kombination aus:

- einer **linearen Transformation** (z. B. Rotation, Skalierung, Scherung)
- einer **Translation**

Allgemein lässt sich eine affine Abbildung schreiben als:

$$\vec{p}' = A \cdot \vec{p} + \vec{t} \quad \text{mit } A \in \mathbb{R}^{2 \times 2}, \vec{t} \in \mathbb{R}^2$$

EIGENSCHAFTEN AFFINER TRANSFORMATIONEN

- Geraden bleiben Geraden
- Parallele Linien bleiben parallel
- Verhältnisse entlang von Linien (z. B. Teilungsverhältnisse) bleiben erhalten
- Winkel und Längen werden im Allgemeinen nicht bewahrt

AFFINE TRANSFORMATIONEN IM HOMOGENEN RAUM

Im homogenen Koordinatensystem kann eine affine Transformation als eine 3×3 -Matrix geschrieben werden. Die Translation wird in die dritte Spalte eingebettet, die untere Zeile bleibt $(0 \ 0 \ 1)$:

$$T_{\text{aff}} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Oder allgemeiner:

$$T_{\text{aff}}(A, \vec{t}) = \begin{pmatrix} A & \vec{t} \\ \vec{0}^T & 1 \end{pmatrix} \quad \text{mit } A \in \mathbb{R}^{2 \times 2}, \vec{t} \in \mathbb{R}^2$$

GEOMETRISCHE BEDEUTUNG

Durch affine Transformationen bleiben erhalten:

- **Kollinearität** – Punkte auf einer Linie bleiben auf einer Linie
- **Parallelität** – parallele Linien bleiben parallel
- **Verhältnisse** von Abständen auf Linien

Sie verändern:

- Winkel zwischen Linien
- Längen von Strecken
- Formen (z. B. Rechtecke werden zu Parallelogrammen)

ZUSAMMENFASSUNG Typen homogener Transformationen mit Beispielen**1. TRANSLATION / ISOMETRIE** Translation um $\vec{t} = (3, 2)$:

$$T = \begin{pmatrix} 1 & 0 & 3 \\ 0 & 1 & 2 \\ 0 & 0 & 1 \end{pmatrix}, \quad \vec{p} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad T\vec{p} = \begin{pmatrix} 4 \\ 3 \\ 1 \end{pmatrix}$$

2. ÄHNLICHKEITSABBILDUNG Rotation um 90° + Skalierung $s = 2$:

$$T = \begin{pmatrix} 0 & -2 & 0 \\ 2 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \vec{p} = \begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}, \quad T\vec{p} = \begin{pmatrix} 0 \\ 2 \\ 1 \end{pmatrix}$$

3. AFFINE ABBILDUNG

$$T = \begin{pmatrix} 1 & 0.5 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \quad \vec{p} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix}, \quad T\vec{p} = \begin{pmatrix} 3.5 \\ 1 \\ 1 \end{pmatrix}$$

4. PROJEKTIVE ABBILDUNG

$$T = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.5 & 0.5 & 1 \end{pmatrix}, \quad \vec{p} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}, \quad T\vec{p} = \begin{pmatrix} 1 \\ 1 \\ 2 \end{pmatrix} \Rightarrow \vec{p}_{\mathbb{R}^2} = \begin{pmatrix} 1/2 \\ 1/2 \end{pmatrix}$$

3.4.11 3.5.5 Transformation entlang einer Achse

Manchmal sollen Transformationen – insbesondere **Skalierungen oder Spiegelungen** – nicht entlang der Standardachsen (x, y), sondern entlang einer **beliebigen Achse im Raum** erfolgen. Typische Beispiele sind:

- Skalierung entlang einer geneigten Achse
- Spiegelung an einer schräge verlaufenden Symmetrieachse
- Verzerrung entlang lokaler Objektachsen (z.B. Texturstreckung)

Da Transformationen in homogenen Koordinaten immer auf die globalen Koordinatenachsen bezogen sind, muss eine Achsenanpassung vorgenommen werden.

Vorgehen: Lokale Achse auf globale ausrichten

Um eine Transformation entlang einer lokalen Achse auszuführen, wird folgende Matrixkombination verwendet:

$$M = T_\tau^{-1} \cdot R^{-1} \cdot S_y \cdot R \cdot \tau$$

Dabei gilt:

- τ : Translation des Achsenursprungs in den Koordinatenursprung
- R : Rotation, sodass die lokale Achse mit der y -Achse übereinstimmt
- S_y : Skalierung entlang der y -Achse
- R^{-1} : Rückrotation der Achse
- T^{-1} : Rücktranslation an die Originalposition

3.5 VERKETTUNG AFFINER TRANSFORMATIONEN IM \mathbb{R}^2

3.5.1 Motivation und Bedeutung

In komplexeren Anwendungen der Computergrafik, Geometrie und Robotik reicht es oft nicht aus, nur einzelne Transformationen wie eine Rotation oder eine Translation anzuwenden. Stattdessen müssen **mehrere Transformationen miteinander kombiniert** werden. Beispiele dafür sind:

- Ein Objekt wird zunächst lokal skaliert, dann gedreht und anschließend in eine Szene eingefügt.
- Eine Kamera bewegt sich durch den Raum und verändert gleichzeitig ihren Blickwinkel.
- Ein Gelenk eines Roboters dreht sich relativ zu seinem Elternteil.

Für solche Fälle ist es von zentraler Bedeutung, Transformationen zu **verketteten Transformationen** zusammenzufassen. Die homogenen Koordinaten ermöglichen es, diese Kombination durch **Matrixmultiplikation** darzustellen.

Die Vorteile liegen auf der Hand:

- Transformationen können effizient gespeichert und berechnet werden.
- Einzelne Operationen lassen sich vorab zusammenfassen (Preprocessing).
- Transformationen lassen sich als **einheitliche lineare Operation** formulieren.
- Lokale Koordinatensysteme und Hierarchien (z.B. in Szenengraphen) lassen sich sauber beschreiben.

Dieser Abschnitt zeigt daher, wie sich affine Transformationen elegant miteinander kombinieren lassen — inklusive der mathematischen Konsequenzen und praktischen Anwendungen.

3.5.2 Allgemeine Form affiner Transformationen

Affine Transformationen umfassen Kombinationen aus **Rotation, Skalierung, Scherung und Translation**. Im Gegensatz zu rein linearen Transformationen enthalten sie also zusätzlich eine Verschiebung im Raum.

Im homogenen Koordinatensystem lassen sich affine Transformationen durch eine 3×3 -Matrix darstellen:

$$T_{\text{aff}} = \begin{pmatrix} a_{11} & a_{12} & t_x \\ a_{21} & a_{22} & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

Dabei ist:

- $\begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$ eine lineare Transformation (Rotation, Skalierung, Scherung),
- $\vec{t} = \begin{pmatrix} t_x \\ t_y \end{pmatrix}$ der Translationsvektor,
- die letzte Zeile $(0 \ 0 \ 1)$ stellt sicher, dass wir im homogenen \mathbb{R}^2 bleiben.

BEISPIELE

- **Translation:**

$$T = \begin{pmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{pmatrix}$$

- **Rotation um den Ursprung (Winkel α):**

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

- **Nicht-uniforme Skalierung:**

$$T = \begin{pmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Da affine Transformationen über eine Matrixmultiplikation angewendet werden, ist ihre Verkettung besonders effizient und erlaubt die einfache Kombination mehrerer geometrischer Operationen in einer einzigen Matrix.

Im nächsten Abschnitt sehen wir, wie solche Matrizen zusammengesetzt werden – und welche Herausforderungen (z.B. Nicht-Kommutativität) dabei auftreten.

3.5.3 Zusammensetzen und Nicht-Kommutativität

Ein großer Vorteil homogener Koordinaten ist die Möglichkeit, mehrere Transformationen durch **Matrixmultiplikation** zu einer einzigen Gesamttransformation zusammenzufassen:

$$T_{\text{gesamt}} = T_n \cdot \dots \cdot T_2 \cdot T_1$$

Dies erlaubt es, z.B. Translation, Skalierung und Rotation effizient zu kombinieren. Die Reihenfolge der Multiplikation entspricht der **Reihenfolge der Anwendung von rechts nach links**:

$$\vec{p}' = T_{\text{gesamt}} \cdot \vec{p} = T_n \cdot \dots \cdot T_1 \cdot \vec{p}$$

Wichtig: Die **Matrixmultiplikation ist nicht kommutativ!** Das bedeutet:

$$T_1 \cdot T_2 \neq T_2 \cdot T_1$$

BEISPIEL

Eine Rotation gefolgt von einer Skalierung liefert ein anderes Ergebnis als die umgekehrte Reihenfolge.

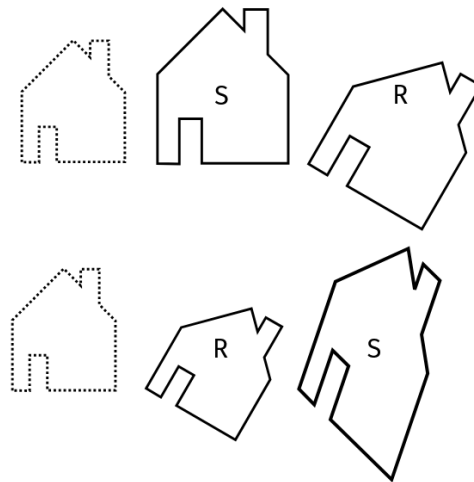


Abbildung: Reihenfolge von R (Rotation) und S (Skalierung) verändert das Ergebnis deutlich.

ZUSAMMENFASSUNG

Beim Aufbau komplexer Transformationen muss die Reihenfolge genau überlegt werden. In der Praxis bedeutet das:

- Transformationen werden oft in einem bestimmten lokalen Kontext (z.B. Objekthierarchie) geplant.
- Die „innere“ Transformation (z.B. Rotation) wird zuerst angewendet, die „äußere“ (z.B. globale Translation) zuletzt.
- Transformationen können vorab multipliziert und in einer einzigen Matrix gespeichert werden.

3.5.4 Transformation um einen lokalen Punkt (Pivot)

Viele Transformationen wie Rotationen oder Skalierungen sollen nicht um den Ursprung, sondern um einen bestimmten Punkt **im Objekt selbst** ausgeführt werden. Ein typisches Beispiel ist die Drehung einer Tür um ihr Scharnier oder die Bewegung eines Gelenks. Da affine Transformationen in homogenen Koordinaten immer relativ zum Ursprung definiert sind, müssen wir einen Trick anwenden:

- 1. Verschiebe den Pivotpunkt **in den Ursprung** (Translation T)
- 2. Führe die gewünschte Transformation (z.B. Rotation R) durch
- 3. Verschiebe das Objekt zurück (inverse Translation T^{-1})

Die Gesamttransformation ergibt sich dann zu:

$$M = T^{-1} \cdot R \cdot T$$

Diese Formel funktioniert analog auch für andere Transformationen (z.B. Skalierung S):

$$M = T^{-1} \cdot S \cdot T$$

Beispiel:

Ein Objekt soll um den Punkt $\vec{p}_0 = (2, 1)$ um 45° rotiert werden:

- Translation T verschiebt \vec{p}_0 in den Ursprung:

$$T = \begin{pmatrix} 1 & 0 & -2 \\ 0 & 1 & -1 \\ 0 & 0 & 1 \end{pmatrix}$$

- Rotation R :

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

- Rücktranslation T^{-1} :

$$T^{-1} = \begin{pmatrix} 1 & 0 & 2 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}$$

3.6 TRANSFORMATION IM \mathbb{R}^3

Die im \mathbb{R}^2 behandelten Transformationen lassen sich prinzipiell in den dreidimensionalen Raum erweitern. Im Unterschied zum 2D-Fall werden die Objekte nun nicht mehr nur in der Ebene, sondern im Raum verschoben, rotiert oder skaliert.

Neu: Im \mathbb{R}^3 muss zusätzlich die z-Komponente berücksichtigt werden. Daraus ergeben sich:

- 4×4 -Transformationsmatrizen (statt 3×3 in 2D)
- Drei unabhängige Rotationsachsen (x, y, z)
- Neue Anwendungen: 3D-Objekte, Kameras, Szenen, Beleuchtung

Homogene Koordinaten erweitern also den Raum \mathbb{R}^3 zu \mathbb{H}^3 , um Translationen als lineare Matrizenoperationen auszudrücken:

$$\vec{p}_{\text{hom}} = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad \text{und} \quad T \in \mathbb{R}^{4 \times 4}$$

Translation

Eine Translation verschiebt ein Objekt im Raum entlang x-, y- und z-Achse. Die zugehörige Matrix ist:

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

Diese Transformation entspricht der Addition eines Translationsvektors.

Skalierung

Skalierung im Raum kann unterschiedlich starke Streckung in jeder Achsenrichtung bewirken:

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

- **Uniforme Skalierung:** $s_x = s_y = s_z$
- **Nicht-uniforme Skalierung:** unabhängige Skalierungsfaktoren

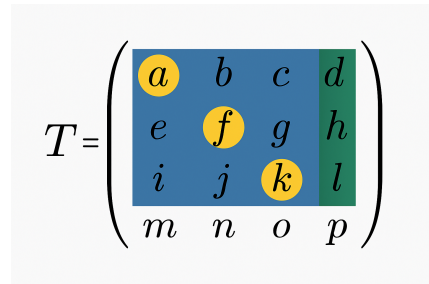
Rotationen im \mathbb{R}^3

Im dreidimensionalen Raum gibt es drei fundamentale Rotationen — um die x -, y - und z -Achse. Sie werden durch folgende 4×4 -Matrizen beschrieben:

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

Die einzelnen Einträge kodieren:

- **Skalierung:** Diagonal (a , f , k)
- **Rotation / Scherung:** Off-Diagonal (•)
- **Translation:** Rechte Spalte (d , f , h , l)



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

Abbildung 8: Allgemeine Transformationsmatrix

3.7 KAMERA- UND PROJEKTIONSTRANSFORMATIONEN

3.7.1 Koordinatensysteme

DEFINITION EINES PUNKTES IM RAUM

Ein Punkt im Raum \mathbb{R}^3 kann durch seine Koordinaten relativ zu einem definierten Koordinatensystem eindeutig bestimmt werden. Ein solches Koordinatensystem wird durch einen Ursprungspunkt und ein Set von orthogonalen Basisvektoren definiert. Jeder Punkt kann als Vektor vom Ursprung des Koordinatensystems aus betrachtet werden.

Beschreibung:

- Ein Punkt P in einem 3D-Raum kann durch einen Vektor $\vec{p} = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$ vom Ursprung des Koordinatensystems aus dargestellt werden. Die Komponenten x, y, z geben die Ausdehnung des Punktes entlang der jeweiligen Achsen an.
- Die Basisvektoren eines kartesischen Koordinatensystems sind typischerweise $\hat{i} = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix}$, $\hat{j} = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}$ und $\hat{k} = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$. Ein Punkt P kann dann als Linearkombination dieser Basisvektoren ausgedrückt werden: $P = x\hat{i} + y\hat{j} + z\hat{k}$.

HOMOGENE KOORDINATEN

In der Computergrafik werden Punkte und Vektoren oft in **Homogenen Koordinaten** dargestellt. Dies ermöglicht es, alle Transformationen (Translation, Rotation, Skalierung und perspektivische Projektion) durch Matrixmultiplikationen auszudrücken. Ein 3D-Punkt $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$

wird in homogenen Koordinaten zu $\begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$, während ein 3D-Vektor $\begin{pmatrix} x \\ y \\ z \end{pmatrix}$ zu $\begin{pmatrix} x \\ y \\ z \\ 0 \end{pmatrix}$ wird.

Beschreibung:

- Die zusätzliche vierte Komponente, oft als w -Koordinate bezeichnet, erlaubt es, Translationen (Verschiebungen) als Matrixmultiplikationen darzustellen, was mit reinen 3×3 -Matrizen nicht möglich wäre.
- Für Punkte ist $w = 1$, da sie einen Ort im Raum repräsentieren und von Translationen beeinflusst werden.
- Für Vektoren ist $w = 0$, da sie nur eine Richtung und Länge repräsentieren und unabhängig von Translationen sind.

3.7.2 Kamera-Transformation (View Matrix)**KAMERA-ANSICHT**

Die **Kamera-Transformation** oder **View Matrix** ist entscheidend, um die Szene aus der Perspektive des Betrachters (der "Kamera") darzustellen. Sie transformiert Objekte vom **World Space** in den **View Space**. Das Ziel ist es, die Kamera an den Ursprung des Koordinatensystems zu verschieben und sie so auszurichten, dass sie entlang einer Standardachse (oft die negative Z -Achse) blickt, während die Y -Achse nach oben zeigt.

Beschreibung:

- Eine Kamera kann im **World Space** durch ihre Position (Augpunkt \vec{e}), einen Blickpunkt (\vec{l}) und einen Up-Vektor (\vec{u}) definiert werden.
- Der **LookAt-Vektor** $\vec{g} = \vec{l} - \vec{e}$ definiert die Blickrichtung der Kamera.
- Der **Up-Vektor** \vec{u} definiert, welche Richtung oben für die Kamera ist.
- Die Kamera-Transformation wird konstruiert, indem ein neues Koordinatensystem basierend auf der Kameraorientierung definiert und dann die Invers-Transformation angewendet wird.

KONSTRUKTION DER VIEW MATRIX

Die **View Matrix** M_{view} wird aus den drei orthonormalen Achsen des Kamera-Koordinatensystems und der Kameraposition \vec{e} gebildet. Diese Achsen sind der normierte LookAt-Vektor (\vec{z}_{cam}), der Up-Vektor (\vec{y}_{cam}) und der Rechts-Vektor (\vec{x}_{cam}).

$$\vec{z}_{\text{cam}} = \frac{\vec{g}}{\|\vec{g}\|} \quad (\text{oft } -\frac{\vec{g}}{\|\vec{g}\|} \text{ für gängige OpenGL/DirectX Konventionen})$$

$$\vec{x}_{\text{cam}} = \frac{\vec{u} \times \vec{z}_{\text{cam}}}{\|\vec{u} \times \vec{z}_{\text{cam}}\|}$$

$$\vec{y}_{\text{cam}} = \vec{z}_{\text{cam}} \times \vec{x}_{\text{cam}}$$

Die View Matrix ist die Inverse der Matrix, die die Kamera vom lokalen Kamera-Koordinatensystem in das Weltkoordinatensystem transformieren würde. Sie besteht aus einem Rotationsanteil R und einem Translationsanteil T:

$$M_{\text{view}} = R_{\text{cam}}^{-1} \cdot T_{\text{cam}}^{-1} = \begin{pmatrix} \vec{x}_{\text{cam},x} & \vec{x}_{\text{cam},y} & \vec{x}_{\text{cam},z} & -\vec{e} \cdot \vec{x}_{\text{cam}} \\ \vec{y}_{\text{cam},x} & \vec{y}_{\text{cam},y} & \vec{y}_{\text{cam},z} & -\vec{e} \cdot \vec{y}_{\text{cam}} \\ \vec{z}_{\text{cam},x} & \vec{z}_{\text{cam},y} & \vec{z}_{\text{cam},z} & -\vec{e} \cdot \vec{z}_{\text{cam}} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Beschreibung:

- $\vec{x}_{\text{cam}}, \vec{y}_{\text{cam}}, \vec{z}_{\text{cam}}$: Die orthonormalen Basisvektoren des Kamera-Koordinatensystems. Diese bilden die Rotationskomponenten der View Matrix.
- \vec{e} : Die Position der Kamera im Weltkoordinatensystem.
- $-\vec{e} \cdot \vec{x}_{\text{cam}}, -\vec{e} \cdot \vec{y}_{\text{cam}}, -\vec{e} \cdot \vec{z}_{\text{cam}}$: Diese Terme sind die Komponenten des Translationsvektors, die die Verschiebung des Ursprungs der Kamera auf den Ursprung des Weltkoordinatensystems und die Ausrichtung der Achsen umfassen. Es ist die Transformation, die die Welt um "die Kamera herum bewegt.

3.7.3 Projektion (Projection Matrix)

PROJEKTION IM ALLGEMEINEN

Die **Projektion** ist der Prozess, 3D-Punkte auf eine 2D-Fläche abzubilden, um ein Bild zu erzeugen. Es gibt zwei Haupttypen von Projektionen:

- **Orthogonale Projektion:** Bewahrt parallele Linien und Längenverhältnisse. Wird oft für technische Zeichnungen oder topografische Karten verwendet, da keine Perspektive berücksichtigt wird.
- **Perspektivische Projektion:** Simuliert die menschliche Sehweise, bei der weiter entfernte Objekte kleiner erscheinen. Dies ist der Standard für realistische 3D-Grafik.

PERSPEKTIVISCHE SKALIERUNG - PERSPECTIVE DIVIDE

Wie bereits erwähnt, verwenden wir homogene Koordinaten, um 3D-Punkte im \mathbb{R}^4 ein-

zubetten. Ein homogener Punkt im Clip Space hat die Form $\begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix}$. Um von homogenen

Koordinaten zurück zu den kartesischen 3D-Koordinaten im **Normalized Device Coordinates (NDC)**-Raum zu gelangen, führen wir die **Perspective Divide** (perspektivische Division) durch.

$$\frac{1}{w_c} \cdot \begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} = \begin{pmatrix} x_c/w_c \\ y_c/w_c \\ z_c/w_c \\ w_c/w_c \end{pmatrix} = \begin{pmatrix} x_{ndc} \\ y_{ndc} \\ z_{ndc} \\ 1 \end{pmatrix}$$

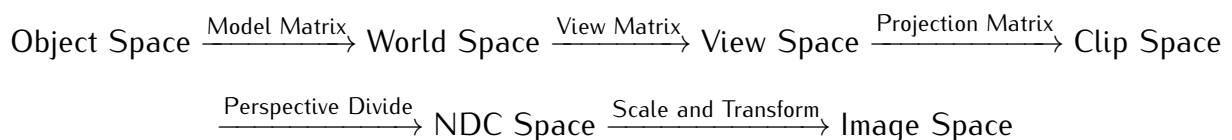
Beschreibung:

- x_c, y_c, z_c : Die Koordinaten des Punktes im homogenen Clip Space.
- w_c : Die vierte Komponente des homogenen Punktes, die bei der perspektivischen Projektion die Tiefeninformation trägt. Je weiter ein Punkt von der Kamera entfernt ist, desto größer ist sein w_c -Wert.
- $x_{ndc}, y_{ndc}, z_{ndc}$: Die resultierenden Koordinaten im **Normalized Device Coordinates (NDC)**-Raum. Durch die Division durch w_c werden die x - und y -Koordinaten perspektivisch skaliert (Objekte in der Ferne werden kleiner). Die z -Koordinate wird ebenfalls normalisiert und behält dabei ihre relationale Tiefeninformation.

Der NDC-Raum ist ein Würfel, dessen Koordinaten in allen Dimensionen (X, Y, Z) von -1 bis 1 reichen. Dieser standardisierte Raum ist die Grundlage für nachfolgende Schritte wie das Clipping und die Rasterisierung.

ZUSAMMENHANG IN DER TRANSFORMATIONSPIPELINE

Die **Transformationspipeline** vereint all diese Transformationen in einer sequentiellen Abfolge. Jeder Schritt bringt die Geometrie näher an die endgültige Bilddarstellung heran:



Beschreibung:

- **Model Matrix**: Transformiert Objekte vom lokalen *Object Space* in den *World Space*. Dies beinhaltet Skalierung, Rotation und Translation einzelner Objekte.
- **View Matrix**: Transformiert Objekte vom *World Space* in den *View Space* (Kameraraum). Sie positioniert und orientiert die Szene relativ zur Kamera.
- **Projection Matrix**: Transformiert Objekte vom *View Space* in den homogenen *Clip Space*. Dies ist der Schritt, der die Perspektive anwendet und die Tiefeninformation für die spätere Verwendung vorbereitet.
- **Perspective Divide**: Konvertiert die homogenen Koordinaten aus dem *Clip Space* in die kartesischen Koordinaten des *NDC Space*, indem durch die w_c -Komponente dividiert wird.

- **Scale and Transform:** Der letzte Schritt, der die NDC-Koordinaten in die tatsächlichen Pixelkoordinaten des Bildschirms im *Image Space* umwandelt, unter Berücksichtigung der Bildschirmauflösung und des Viewports.

Diese Kette von Transformationen stellt sicher, dass 3D-Objekte korrekt auf dem 2D-Bildschirm dargestellt werden, wobei die Perspektive und die relative Positionierung aller Elemente berücksichtigt werden.

4

SICHTBARKEIT – VISIBILITY

4.1 TRANSFORMATIONSPIPELINE UND PROJEKTIONSRAUM

TRANSFORMATIONSPIPELINE

Die **Transformationspipeline** ist ein sequentieller Prozess, der Objekte von einem Koordinatensystem in ein anderes überführt, um sie für die Darstellung vorzubereiten. Jeder Schritt transformiert die Geometrie näher zum Bild. Das Objekt durchläuft dabei folgende Koordinatensysteme:

Object Space → World Space → View Space → Clip Space → NDC → Image Space

Beschreibung:

- **Object Space:** Dies ist das lokale Koordinatensystem, in dem ein 3D-Objekt ursprünglich modelliert wird. Jedes Objekt hat seinen eigenen Ursprung und seine eigene Ausrichtung.
- **World Space:** Objekte werden aus ihrem lokalen Objektkoordinatensystem in ein gemeinsames Weltkoordinatensystem transformiert. Hier werden alle Objekte der Szene relativ zueinander positioniert.
- **View Space:** In diesem Schritt werden die Objekte aus der Perspektive der Kamera betrachtet. Die Transformation in den View Space richtet die Szene so aus, dass die Kamera am Ursprung positioniert ist und in eine definierte Richtung (z.B. entlang der negativen Z-Achse) blickt.
- **Clip Space:** Nach Anwendung der Projektionsmatrix entstehen Koordinaten im homogenen **Clip Space**. Dies ist ein 4D-Raum, in dem das Sichtvolumen (View Frustum) zu einem Würfel transformiert wird.
- **Normalized Device Coordinates (NDC):** Durch den **Perspective Divide** werden die homogenen Clip-Space-Koordinaten in die NDC transformiert. In diesem Raum ist das gesamte sichtbare Volumen auf einen Einheitswürfel normalisiert, typischerweise von $[-1, 1]$ in allen drei Achsen.
- **Image Space:** Der letzte Schritt ist die Transformation in den 2D-Bildraum, also die Pixelkoordinaten des Ausgabebildschirms.

PERSPEKTIVISCHE PROJEKTION UND PROBLEM DER TIEFE

Die **perspektivische Projektion** bildet 3D-Punkte auf eine 2D-Ebene ab, um den Eindruck von Tiefe zu erzeugen. Eine vereinfachte Form der Projektionsmatrix P lautet:

$$P = \begin{pmatrix} -f & 0 & 0 & 0 \\ 0 & -f & 0 & 0 \\ 0 & 0 & -f & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Beschreibung:

- f : Dies ist die Brennweite (focal length) der virtuellen Kamera. Sie bestimmt, wie stark die Perspektive ausgeprägt ist. Ein größerer Wert für f führt zu einer geringeren perspektivischen Verzerrung.

Wenn diese Projektionsmatrix auf einen homogenen Punkt $p = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}$ angewendet wird,

ergibt sich der transformierte Punkt p' :

$$p = \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \Rightarrow p' = P \cdot p = \begin{pmatrix} -fx \\ -fy \\ -fz \\ z \end{pmatrix}$$

Beschreibung:

- x, y, z : Die Koordinaten des Punktes im View Space.
- $-fx, -fy, -fz$: Die skalierten x, y, z Koordinaten.
- z : Die vierte Komponente des homogenen Punktes. Dieser Wert wird später für den **Perspective Divide** verwendet. Die Projektion beeinflusst die x - und y -Koordinaten nichtlinear, da sie später durch z geteilt werden.

Nach dem **Perspective Divide**, also der Division der ersten drei Komponenten durch die vierte Komponente z , erhält man die Koordinaten im 3D-Bildraum:

$$p_{ndc} = \left(-f \frac{x}{z}, -f \frac{y}{z}, -f \right)$$

Beschreibung:

- $-f \frac{x}{z}, -f \frac{y}{z}$: Dies sind die normalisierten x - und y -Koordinaten, die nun von der ursprünglichen z -Koordinate des Punktes abhängen und somit die perspektivische Verkürzung abbilden.
- $-f$: Die z -Koordinate ist nach dieser vereinfachten Projektion konstant $-f$. Dies stellt das Hauptproblem dar: Alle projizierten Punkte liegen auf einer Ebene bei $z = -f$, wodurch die ursprüngliche Tiefeninformation verloren geht. Folglich kann ohne weitere Maßnahmen keine Unterscheidung der Sichtbarkeit getroffen werden, und die Reihenfolge der Darstellung würde die Sichtbarkeit bestimmen, nicht die tatsächliche Tiefe.

4.2 TIEFENBERECHNUNG UND NORMALISIERUNG

INVERSE TIEFENFUNKTION

Um die Sichtbarkeit korrekt zu bestimmen, ist nicht der absolute numerische Wert der Tiefe (p_z) entscheidend, sondern die **Ordinalität der Tiefenwerte** – also welche Punkte näher oder weiter entfernt sind. Eine monotone Funktion erfüllt diesen Zweck. Eine häufig verwendete monotone Funktion für die Tiefenberechnung ist die inverse Tiefe:

$$\text{Tiefenwert} = \frac{1}{z}$$

Beschreibung:

- z : Die ursprüngliche z -Koordinate des Punktes im View Space.
- $\frac{1}{z}$: Dieser Wert ist monoton fallend mit zunehmendem z . Das bedeutet, ein größeres z (weiter entfernt) führt zu einem geringeren Tiefenwert, während ein kleineres z (näher) zu einem größeren Tiefenwert führt. Dies ermöglicht eine korrekte Sortierung von Tiefenwerten zur Bestimmung der Sichtbarkeit.

OPTIMIERTE PROJEKTIONSMATRIX

Um die inverse Tiefenfunktion in die Projektion zu integrieren und gleichzeitig eine Normalisierung des Tiefenbereichs zu ermöglichen, wird die Projektionsmatrix P modifiziert. Sie lautet dann:

$$P = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & a & b \\ 0 & 0 & -1 & 0 \end{pmatrix}$$

Beschreibung:

- f : Die Brennweite, wie zuvor.
- a, b : Dies sind Parameter, die so gewählt werden, dass die Tiefenwerte nach der Projektion und **Perspective Divide** in einen normalisierten Bereich, typischerweise $[-1, 1]$ für den NDC- Z -Wert, abgebildet werden. Diese Parameter erhalten die Tiefeninformation korrekt.

Die Parameter a und b werden aus den Definitionen der Near-Plane (z_n) und Far-Plane (z_f) abgeleitet:

$$a = \frac{z_f + z_n}{z_f - z_n}, \quad b = \frac{2z_n z_f}{z_n - z_f}$$

Beschreibung:

- z_n : Dies ist der Abstand der **Near-Plane** (Nah-Ebene) von der Kamera. Objekte, die näher als z_n sind, werden nicht gerendert (geclippt). Der Wert z_n wird auf -1 im NDC abgebildet.
- z_f : Dies ist der Abstand der **Far-Plane** (Fern-Ebene) von der Kamera. Objekte, die weiter als z_f sind, werden ebenfalls nicht gerendert (geclippt). Der Wert z_f wird auf 1 im NDC abgebildet.

- Diese Ebenen definieren das **View Frustum** (Sichtvolumen), das den Bereich der Szene festlegt, der sichtbar sein soll.

4.3 CLIP SPACE UND NDC-RAUM

PERSPECTIVE DIVIDE

Nachdem ein Punkt p mit der optimierten Projektionsmatrix P multipliziert wurde, liegt er

im homogenen **Clip Space** als $p'_v = \begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix}$ vor. Der nächste entscheidende Schritt ist der

Perspective Divide:

$$p'_v = \begin{pmatrix} x_c \\ y_c \\ z_c \\ w_c \end{pmatrix} \Rightarrow p_{ndc} = \left(\frac{x_c}{w_c}, \frac{y_c}{w_c}, \frac{z_c}{w_c} \right)$$

Beschreibung:

- x_c, y_c, z_c : Die kartesischen Koordinaten des Punktes im homogenen Clip Space.
- w_c : Die homogene Komponente des Punktes im Clip Space. Diese ist im Falle der Perspektivprojektion proportional zur ursprünglichen z -Koordinate des Punktes im View Space ($w_c = -p_z$).
- $\frac{x_c}{w_c}, \frac{y_c}{w_c}, \frac{z_c}{w_c}$: Dies sind die normalisierten Koordinaten im **Normalized Device Coordinates (NDC)**-Raum. Die Division durch w_c normiert die Koordinaten so, dass alle sichtbaren Punkte in den Intervall $[-1, 1]$ auf allen drei Achsen fallen. Dies ist ein standardisierter Raum für die weitere Verarbeitung (z.B. Clipping, Rasterisierung).

CLIPPING

Ein wichtiger Aspekt der Pipeline ist das **Clipping**, welches die Geometrie, die außerhalb des Sichtvolumens liegt, frühzeitig aus der Pipeline entfernt. Dies geschieht basierend auf den homogenen Clip-Space-Koordinaten und den Clipping-Bedingungen:

$$-w_c \leq x_c, y_c, z_c \leq w_c$$

Beschreibung:

- Die Bedingung besagt, dass die x_c, y_c und z_c Koordinaten eines Punktes innerhalb des Bereichs von $-w_c$ bis w_c liegen müssen.
- Nur Punkte, die diese Bedingungen erfüllen, werden behalten. Punkte außerhalb dieses Raums werden verworfen.
- Für Dreiecke und Linien bedeutet dies, dass nur der Teil, der in das **View Frustum** hineinragt, weiterverarbeitet wird. Dies optimiert die Rechenleistung, da nicht-sichtbare Geometrie frühzeitig eliminiert wird. Das Clipping findet somit vor der **Perspective Divide** statt, was für den Sichtbarkeitstest notwendig ist.

4.4 ZUSAMMENFASSUNG DER SCHRITTE

Die Sichtbarkeitsberechnung in einer Computergrafik-Pipeline folgt einer klaren Abfolge von Transformationen und Tests.

1. **Modell-, Welt- und Kamera-Transformation:** Szeneobjekte, ursprünglich in ihrem lokalen *Object Space*, werden durch die Modell-Matrix in den *World Space* transformiert und anschließend durch die Kamera-Matrix in den *View Space* überführt.
2. **Projektion in den Clip Space:** Die Punkte im *View Space* werden mithilfe der optimierten Projektionsmatrix P in den homogenen *Clip Space* projiziert. Dabei wird die Tiefeninformation so vorbereitet, dass sie erhalten bleibt.
3. **Clipping:** Geometrie, die außerhalb des definierten *View Frustums* liegt – also die Bedingungen $-w_c \leq x_c, y_c, z_c \leq w_c$ nicht erfüllt – wird zu diesem Zeitpunkt verworfen oder auf den sichtbaren Teil reduziert. Dies ist ein wichtiger Optimierungsschritt.
4. **Perspective Divide und Normalisierung:** Die homogenen Clip-Space-Koordinaten werden durch die vierte Komponente (w_c) dividiert, um die *Normalized Device Coordinates (NDC)* zu erhalten. In diesem Raum sind alle sichtbaren Punkte in einem standardisierten Würfel von $[-1, 1]$ in allen Achsen enthalten. Die Tiefeninformation, jetzt normalisiert, ist als z_{ndc} verfügbar.
5. **Bildraumtest und Sichtbarkeitsentscheidung:** Im NDC-Raum können dann mithilfe von Tiefenpuffern (wie dem Z-Buffer) die endgültigen Sichtbarkeitsentscheidungen getroffen werden. Die normalisierten Tiefenwerte entscheiden, welcher Pixel von welchem Objekt gefüllt wird, da näher liegende Objekte weiter entfernt liegende verdecken.

Wichtig: Die gesamte Sichtbarkeitsentscheidung basiert maßgeblich auf der korrekten Erhaltung und Normalisierung der Tiefe im NDC-Raum.

5 | PRÜFUNGSFRAGEN

GRUNDKONZEPTE DER COMPUTERGRAFIK

1. Warum ist das Separationsprinzip in der Computergrafik von zentraler Bedeutung?
2. Erläutern Sie den Begriff Szene und welche Elemente typischerweise darin enthalten sind.
3. Welche Faktoren bestimmen, wie eine Szene durch einen Betrachter (virtuelle Kamera) wahrgenommen wird?
4. Was versteht man unter Bildsynthese und welche Techniken kommen hier zum Einsatz? Nennen Sie mindestens zwei.
5. Beschreiben Sie die Interaktion zwischen Licht und Oberflächen in der Computergrafik. Welche Auswirkungen haben Reflexion, Absorption und Transmission?
6. Welche globalen Lichtphänomene gibt es und warum sind diese für realistische Darstellungen wichtig?

MODELLE – THEORETISCHE GRUNDLAGEN

2.1.1 Modellbildung in der Computergrafik

1. Warum ist es notwendig, reale Objekte in Computergrafik mathematisch zu modellieren?
2. Nennen und erläutern Sie einen Unterschied zwischen impliziten und parametrischen Modellbeschreibungen.
3. Welche Vor- und Nachteile ergeben sich durch eine kleinere Schrittweite bei der parametrischen Darstellung eines Kreises?

2.1.2 Rekonstruktion

1. Was versteht man unter Rekonstruktion in der Computergrafik?
2. Welche Rolle spielen Deskriptoren bei der Rekonstruktion digitaler Bilder?
3. Erklären Sie den Zusammenhang zwischen Auflösung, Deskriptoren und Genauigkeit der Rekonstruktion.

2.1.3 Dreiecksmodelle – Meshes

1. Welche grundlegenden Elemente bilden ein Mesh?
2. Warum sind Dreiecke besonders geeignet für Meshes in der Computergrafik?
3. Nennen und erläutern Sie zwei Vorteile der Verwendung von Dreiecken gegenüber anderen Polygonen.

2.1.4 Einschub: Mannigfaltigkeit

1. Definieren Sie den Begriff Mannigfaltigkeit in der Computergrafik.
2. Warum sind Mannigfaltigkeitsbedingungen für Meshes entscheidend?
3. Geben Sie ein Beispiel für ein mannigfaltiges und ein nicht-mannigfaltiges Mesh.

2.1.5 Topologie und Geometrie

1. Erklären Sie den Unterschied zwischen Topologie und Geometrie in der Computergrafik.
2. Geben Sie jeweils ein Beispiel für gleiche Geometrie mit unterschiedlicher Topologie und umgekehrt.
3. Welche Vorteile bietet eine vollständige topologische Beschreibung?

2.1.6 Attribute und Merkmale (Features)

1. Welche Bedeutung haben Attribute und Merkmale in einem Computergrafik-Modell?
2. Nennen Sie zwei Beispiele für Attribute, die typischerweise in 3D-Modellen verwendet werden.
3. Erläutern Sie den Unterschied zwischen Attributen und Features.

2.1.7 Dimensionalität

1. Was beschreibt die Dimensionalität eines Modells?
2. Warum nutzt die Computergrafik Projektionen vom \mathbb{R}^4 in den \mathbb{R}^3 ?
3. Geben Sie ein Beispiel, wie sich Dimensionalität auf die Visualisierung von Modellen auswirkt.

2.1.8 Einschub: Interpolation

1. Was versteht man unter Interpolation und warum ist sie wichtig in der Computergrafik?
2. Nennen und erläutern Sie kurz drei Arten von Interpolationsverfahren.
3. Zeigen Sie anhand einer Formel ein Beispiel linearer Interpolation zwischen zwei Punkten.

2.1.9 Splines und NURBS (Interpolationstechniken)

1. Erklären Sie den Unterschied zwischen interpolierenden und approximierenden Splines.
2. Was macht NURBS besonders geeignet für komplexe Modellierungen?
3. Nennen Sie ein konkretes Anwendungsbeispiel für NURBS aus der Praxis.

MODELLE – FLÄCHENMODELLE

2.2.1 Polygonale Flächen (Facettierung)

1. Was versteht man unter polygonalen Flächenmodellen und welche Polygone werden typischerweise verwendet?
2. Nennen Sie drei Eigenschaften polygonaler Flächenmodelle.
3. In welchen Anwendungsbereichen kommen polygonale Flächenmodelle bevorzugt zum Einsatz und warum?

2.2.2 Algorithmische Flächen (Parametrische Flächen)

1. Was zeichnet algorithmische (parametrische) Flächenmodelle aus und wie unterscheiden sie sich von polygonalen Modellen?
2. Nennen Sie zwei Beispiele für algorithmische Flächen und erläutern Sie eines detailliert.
3. Welche Vorteile bieten algorithmische Flächenmodelle insbesondere in CAD-Anwendungen?

2.2.3 Variationsbasierte Flächen (Minimierungsprobleme)

1. Erklären Sie das Prinzip variationsbasierter Flächenmodelle.
2. Was versteht man unter einer Signed Distance Function (SDF) und wofür wird diese verwendet?

3. Nennen Sie zwei Anwendungsbeispiele variationsbasierter Flächenmodelle und erläutern Sie eines davon kurz.

2.2.4 Ruled Surfaces (Regelflächen)

1. Was sind Ruled Surfaces (Regelflächen) und wie entstehen sie mathematisch?
2. Erläutern Sie, wie Antoni Gaudí Regelflächen bei der architektonischen Gestaltung verwendet hat.
3. Welche Vorteile bieten Regelflächen für Anwendungen in Architektur und Design?

2.2.5 Freiformflächen (Freiformmodellierung)

1. Was versteht man unter Freiformflächen in der Computergrafik?
2. Wie hängen Freiformflächen mit Splines und NURBS zusammen?
3. In welchen Bereichen kommen Freiformflächen besonders häufig zum Einsatz? Nennen Sie mindestens ein Anwendungsbeispiel.

MESHES

1. Was sind Meshes und warum sind sie essenziell in der Computergrafik?
2. Wie sind Meshes typischerweise strukturiert und organisiert?
3. Erläutern Sie drei verschiedene Optimierungstechniken zur Speicherung von Meshes.
4. Was versteht man unter Subdivision und Simplifikation in Bezug auf Meshes? Geben Sie jeweils ein Anwendungsbeispiel.
5. Welche Einschränkungen haben Meshes bei der Darstellung besonders detailreicher Objekte?

VOLUMENMODELLE

1. Was versteht man unter Volumenmodellen und wie unterscheiden sie sich von Flächenmodellen?
2. Erläutern Sie drei wesentliche Anwendungen von Volumenmodellen.
3. Nennen Sie und erläutern Sie kurz drei verschiedene Methoden zur Volumenmodellierung.
4. Welche Vor- und Nachteile bieten Voxel-Modelle gegenüber anderen Methoden?
5. Beschreiben Sie die Funktionsweise und Vorteile einer Octree-Struktur.

ABBILDUNGSVERZEICHNIS

Abbildung 1	Uniforme Skalierung	20
Abbildung 2	Nicht-uniforme Skalierung	21
Abbildung 3	Rotation Punkt	23
Abbildung 4	Rotation Fläche	23
Abbildung 5	Scherung in x- bzw. y-Richtung	28
Abbildung 6	Spiegelung in x- bzw. y-Richtung	29
Abbildung 7	Allgemeine Transformationsmatrix	29
Abbildung 8	Allgemeine Transformationsmatrix	40
Abbildung 9	Triangle-Soup	54
Abbildung 10	Indexed-Faceset	55
Abbildung 11	Triangle-Strip	55
Abbildung 12	Triangle-Fan	55

```
// Geometrie und Topologie
float[][][] = {
    {{ax,ay,az},{bx,by,bz},{cx,cy,cz}}, //T1
    {{bx,by,bz},{dx,dy,dz},{cx,cy,cz}}, //T2
    {{bx,by,bz},{gx,gy,gz},{dx,dy,dz}} // T3
};

// Jedes Dreieck hat seinen eigenen
// Speicherbereich, d.h. keine erkennbare
// Topologie zwischen den Dreiecken
```

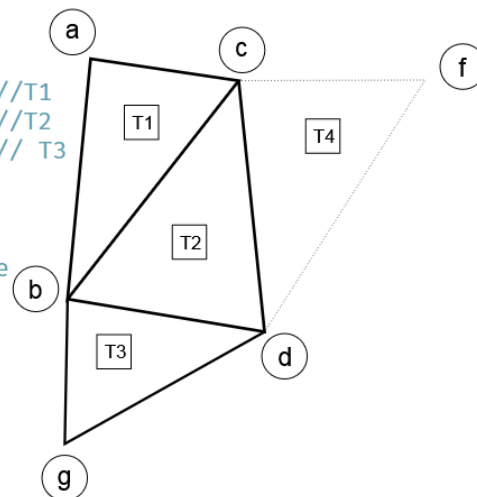


Abbildung 9: Triangle-Soup

```
// Geometrie
float geom[][] = {
    {ax,ay,az}, // Vertex a | 0
    {bx,by,bz}, // Vertex b | 1
    {cx,cy,cz}, // Vertex c | 2
    {dx,dy,dz}, // Vertex d | 3
    {fx,fy,fz}, // Vertex f | 4
    {gx,gy,gz}  // Vertex g | 5
};

// Topologie
uint32_t topo [][] = {
    { 0, 1, 2 }, // face T1
    { 1, 3, 2 }  // face T2
};
```

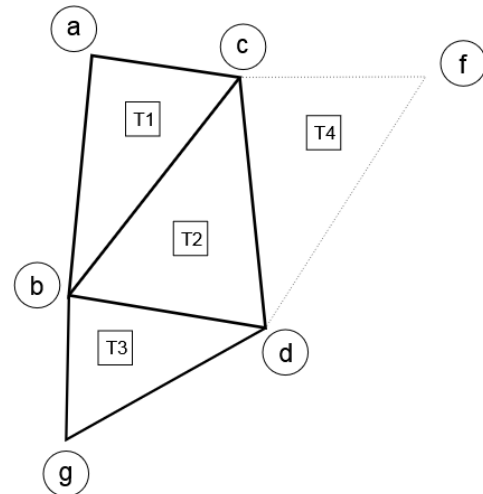


Abbildung 10: Indexed-Faceset

```
// Geometrie
float geom[][] = {
    {ax,ay,az}, // Vertex a | 0
    {bx,by,bz}, // Vertex b | 1
    {cx,cy,cz}, // Vertex c | 2
    {dx,dy,dz}, // Vertex d | 3
    {fx,fy,fz}, // Vertex f | 4
    {gx,gy,gz}  // Vertex g | 5
};

// Topologie
uint32_t topo_ts [] = { 0, 1, 2, 3, 4 };

// ergibt T1(abc), T2(bcd), T4(cdf)
// T2 wird umgedreht interpretiert
```

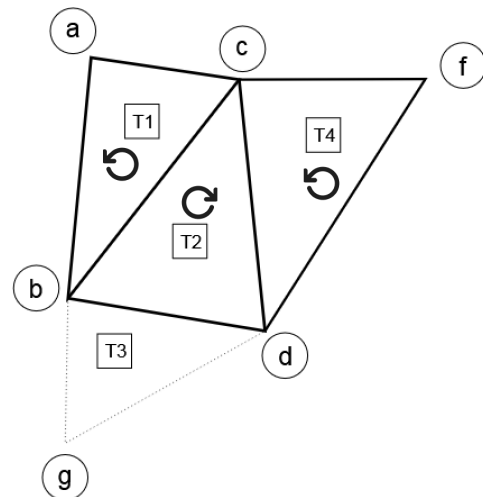


Abbildung 11: Triangle-Strip

```
// Geometrie
float geom[][] = {
    {ax,ay,az}, // Vertex a | 0
    {bx,by,bz}, // Vertex b | 1
    {cx,cy,cz}, // Vertex c | 2
    {dx,dy,dz}, // Vertex d | 3
    {fx,fy,fz}, // Vertex f | 4
    {gx,gy,gz}  // Vertex g | 5
};

// Topologie
uint32_t topo_tf [] = { 1, 5, 3, 2, 0 };
// T3(bgd), T2(bdc), T1(bca)
```

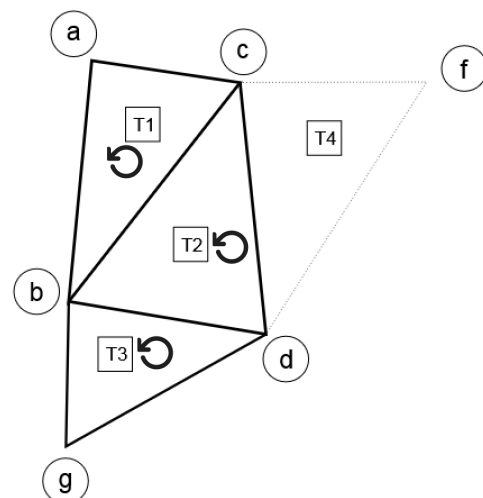


Abbildung 12: Triangle-Fan