VC

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES
fbi
FACHBEREICH INFORMATIK

Visual Computing

Grafische Objekte und deren Programmierung

Hochschule Darmstadt

Prof. Dr. Elke Hergenröther
Björn Frömmer
Prof. Dr. Benjamin Meyer

Wichtige Hinweise

Besuchen Sie die Vorlesung!

Warum?

- Das Skript und die Folien sind nur eine Zusammenfassung
- Tafel- und Hörsaalübungen sind wichtige Hilfen für die Klausur
- Fragen, auch die der anderen Studierenden, helfen Ihnen beim Verständnis
- interessante und motivierende Inhalte :-)

Bereiten Sie den Stoff vor bzw. nach!

Warum?

- Die Klausur ist keine „Auswendiglernklausur“
- Sie müssen den Stoff wirklich beherrschen, um ihn dann in der Klausur richtig anzuwenden.

Inhalte der Vorlesung VC

1. Computergrafik (OpenGL und C++)

- Grafische Programmierung (OpenGL v3.3+, ein Framework wird bereitgestellt)
  - Shader (GLSL)
- Koordinatensysteme
- Visualisierungstechniken
  - Farben, Beleuchtung, Texturen
- Geometrische Transformationen

2. Bildverarbeitung (OpenCV in Python)

- Bildbearbeitung (Bild heller machen, etc.)
- Bildverarbeitung (Bsp. Information aus einem Bild extrahieren)
- Farbmodelle

Um was geht es hier eigentlich?

… (fast) immer um Pixel

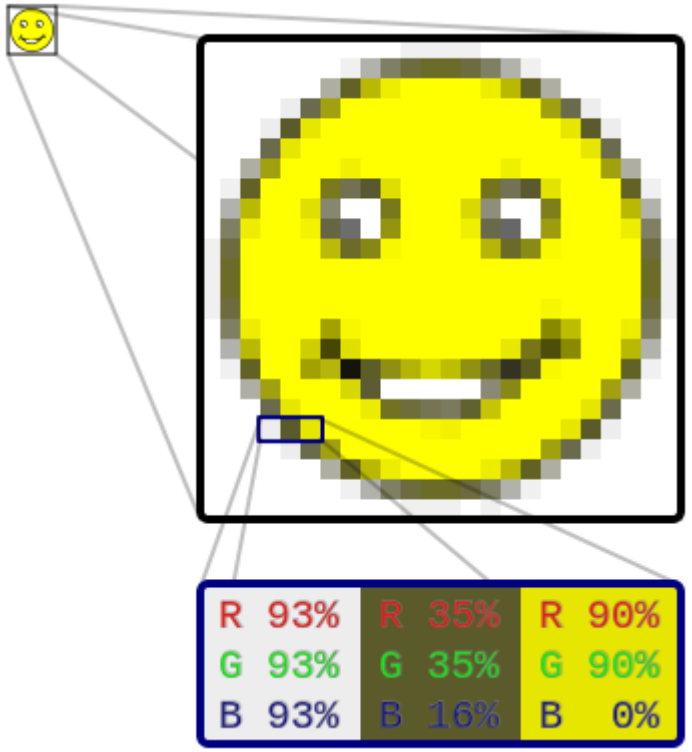
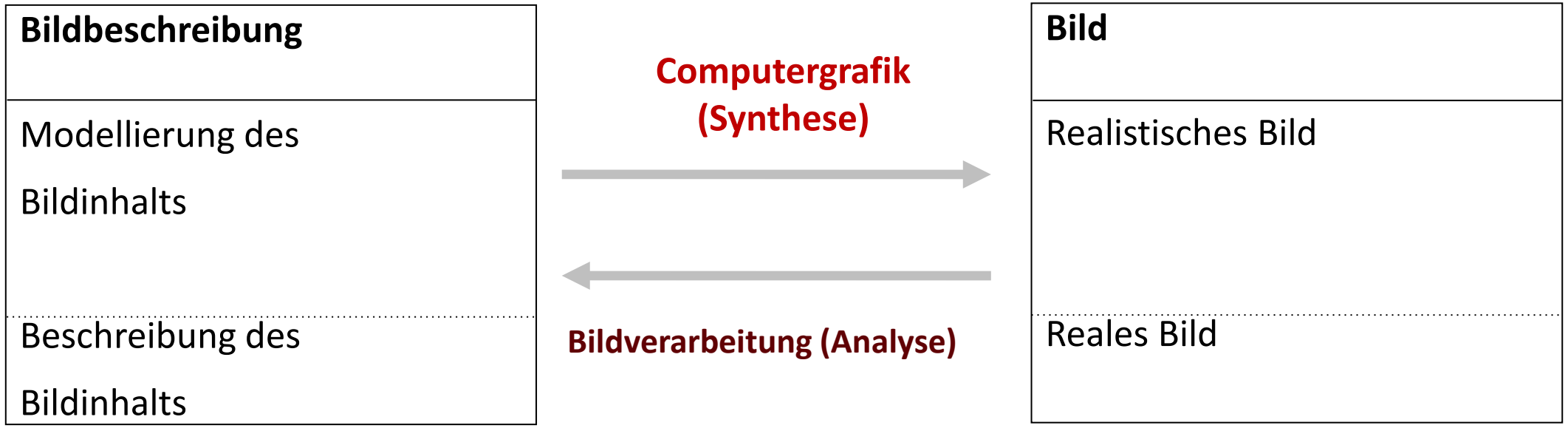
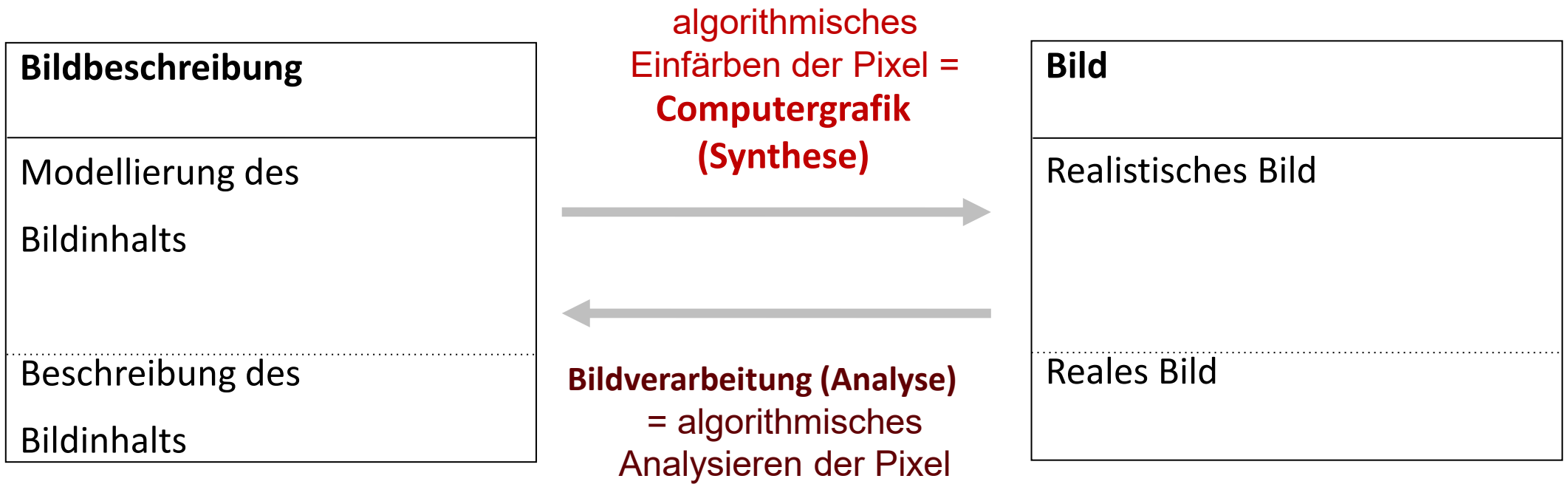


Bild aus: https://developer.tizen.org/forums/native-application-development/what-rasterisation-opengl-graphic-pipeline

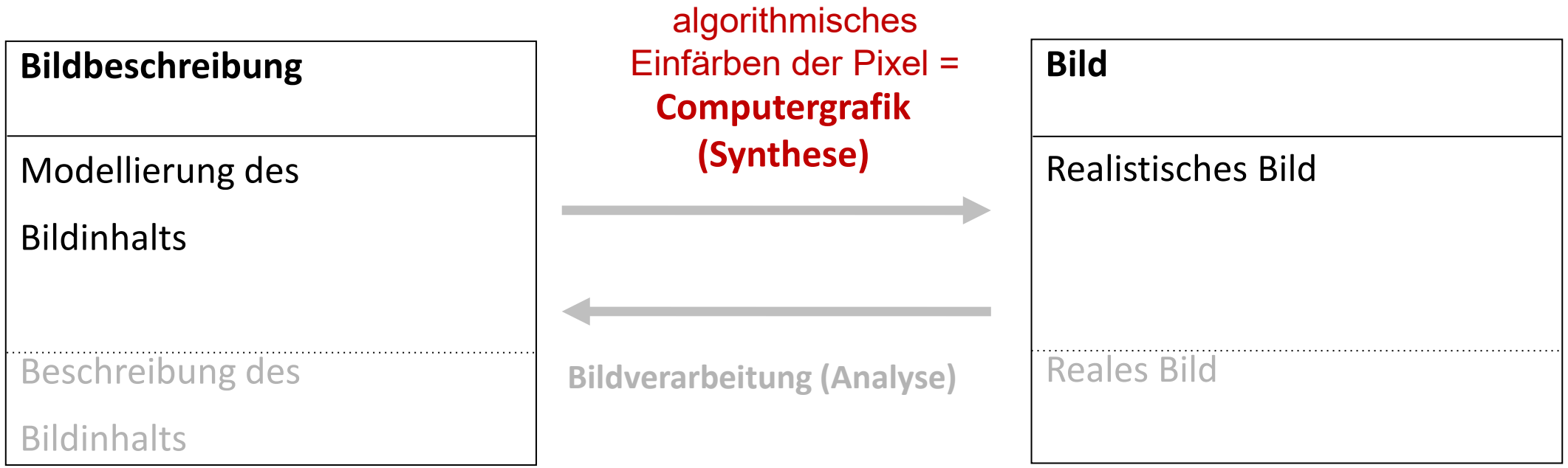
Aufteilung zwischen Computergrafik und Bildverarbeitung



Aufteilung zwischen Computergrafik und Bildverarbeitung



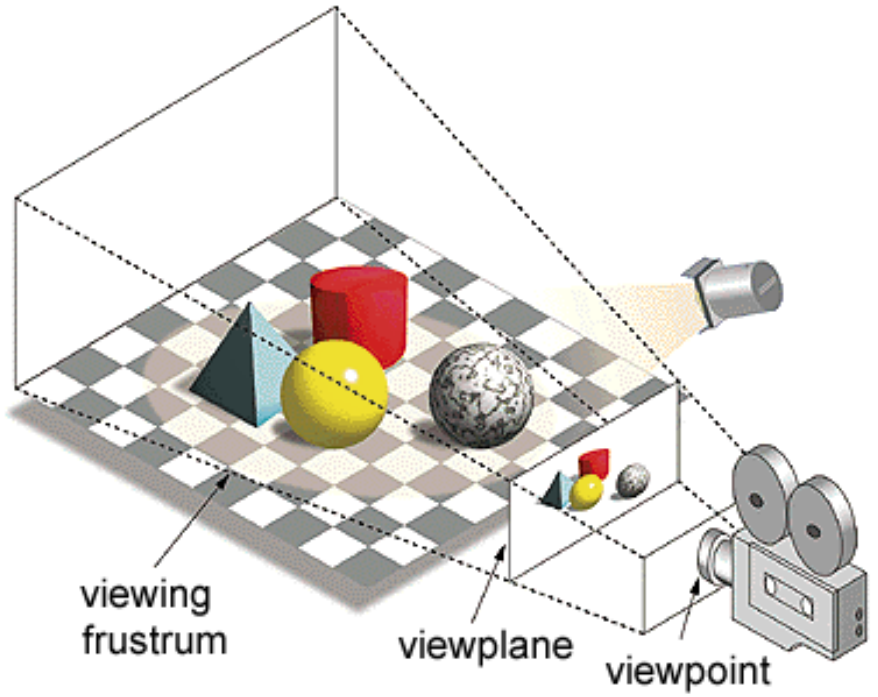
Aufteilung zwischen Computergrafik und Bildverarbeitung



Computergrafik

Modellierung des Bildinhaltes

- 3-dimensionale Geometrien
- Lichtquellen
- Texturen
- Virtuelle Kamera
- Inkl. viewpoint für Perspektive
- Sichtbarer Bereich (viewing frustum)
- Position der Bildebene (viewplane)



Ziel: Pixel der viewplane anhand der aufgenommenen Szene einfärben

Computergrafik

Die Formen der Objekte in der Szene werden durch dreidimensionale Dreiecksgitter angenähert.

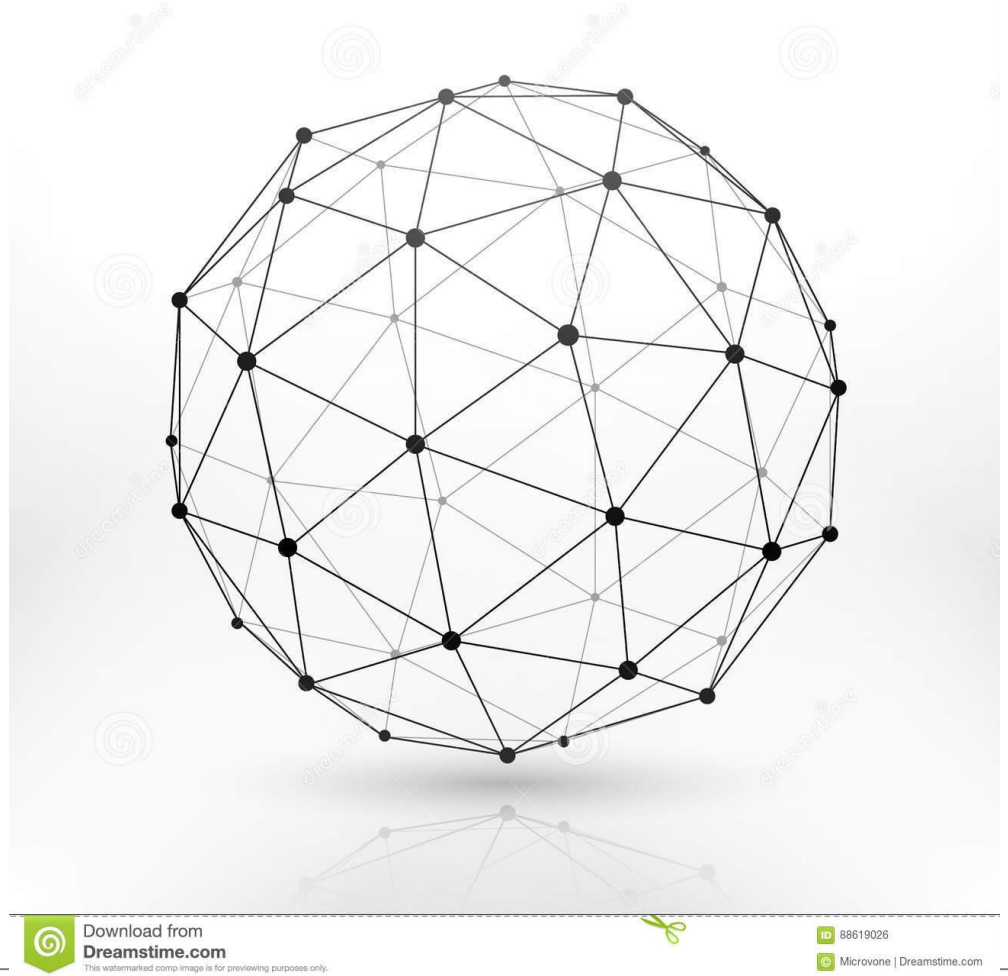


Bild aus: https://de.dreamstime.com/stock-abbildung-wireframe-kugelbereich-zusammenhang-netztechnologieverbindungs-vektorkonzept-image88619026#_

Computergrafik

Um vom Dreiecksgitter zur gerenderten Darstellung zu kommen gibt es im wesentlichen zwei Möglichkeiten:

- Rastergrafik
- Raytracing

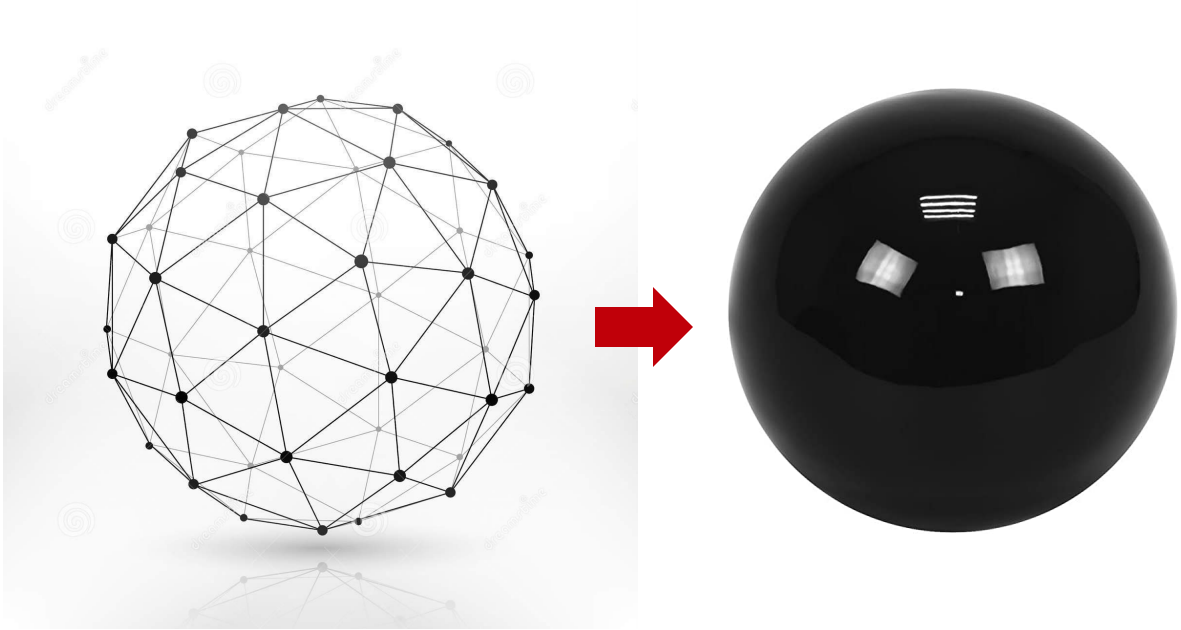


Bild aus: https://de.dreamstime.com/stock-abbildung-wireframe-kugelbereich-zusammenhang-netztechnologieverbindungs-vektorkonzept-image88619026#_

Bild aus: https://www.amazon.de/Keramik-Moderne-Tischdeko-gl%C3%A4nzend-Blackball/dp/B01HETFSD8

Computergrafik

Das Generieren einer **Rastergrafik** ist eine Möglichkeit, um vom Dreiecksgitter zur gerenderten Darstellung zu kommen.

1.) Eckpunkte des Dreiecks (vertices) auf Pixel der Bildebene (viewplane) abbilden.

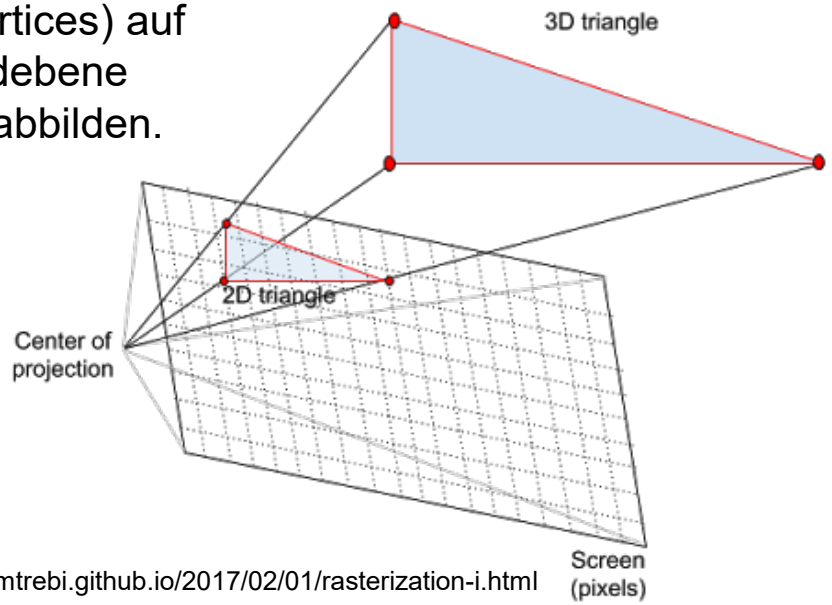


Bild aus: https://mtrebi.github.io/2017/02/01/rasterization-i.html

2.) Anhand der Pixel der Eckpunkte des Dreiecks (vertices) die restlichen Dreieckspixel in der Bildebene (viewplane) bestimmen und einfärben.

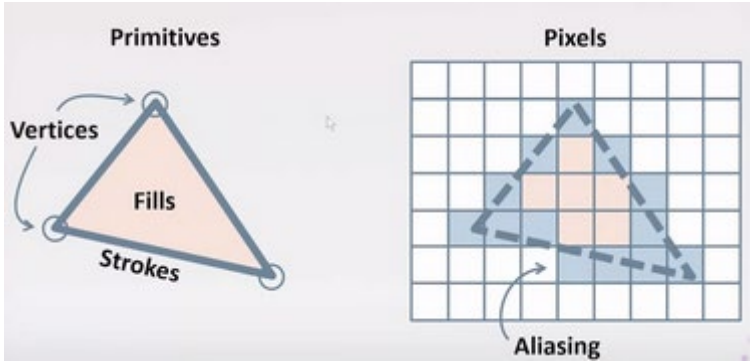


Bild aus: https://developer.tizen.org/forums/native-application-development/what-rasterisation-opengl-graphic-pipeline

Computergrafik

Das Generieren eines Bildes mit **Raytracing** ist eine andere Möglichkeit, um vom Dreiecksgitter zur gerenderten Darstellung zu kommen.

Statt von den Eckpunkten aus die Darstellung der Geometrien zu erzeugen, versucht man die Szene mit „Sehstrahlen“, die von der virtuellen Kamera ausgehen, zu rendern.

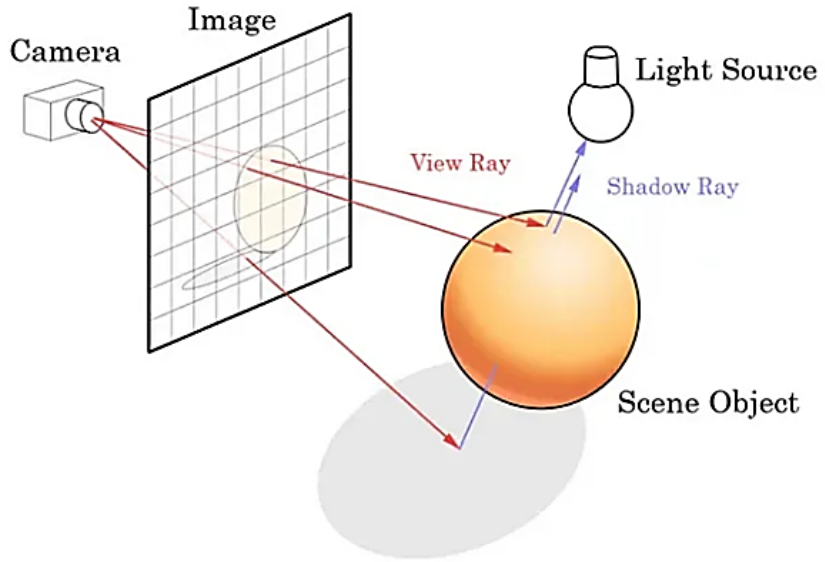


Bild aus: https://medium.com/@junyingw/future-of-gaming-rasterization-vs-ray-tracing-vs-path-tracing-32b334510f1f

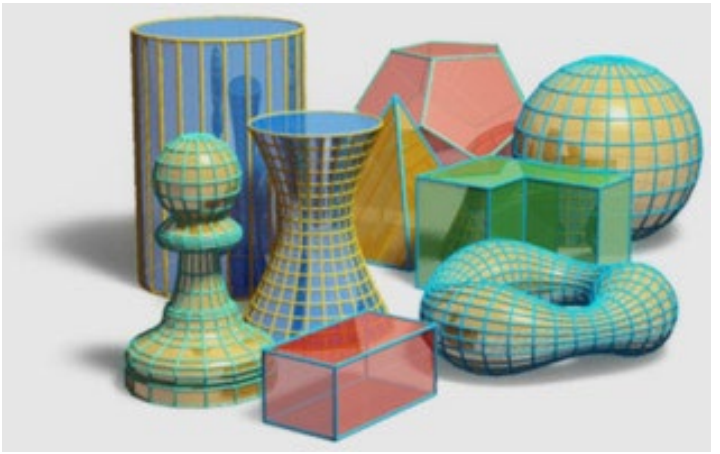
Beispiel für Computergrafik: Ray Tracing Variante: Path Tracing

Vorteil:
- Spiegelungen
- Indirekte Beleuchtung
- …

Nachteil:
- leichte Unschärfe, ist im Verfahren begründet
- Hoher Anspruch an die Hardware

https://youtu.be/aoW4HRi8VKg

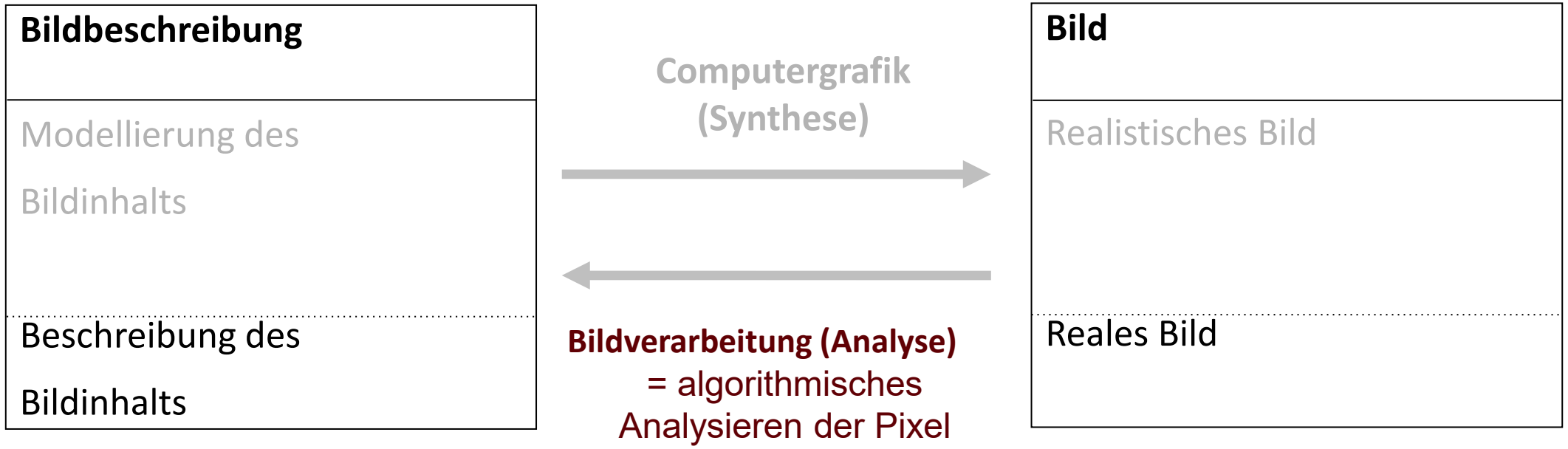
Computergrafik



- Algorithmen zur Erzeugung von Geometrien
- Algorithmen zur „Interaktion“ mit Geometrien
- Algorithmen zur Visualisierung von Geometrien (inkl. Beleuchtungsberechnung, Texturen, …)
- Shader Technologie

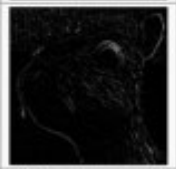
Bilder entnommen aus: http://www.pytha.de/produkt/modeler_1.de.php

Aufteilung zwischen Computergrafik und Bildverarbeitung



Bildverarbeitung

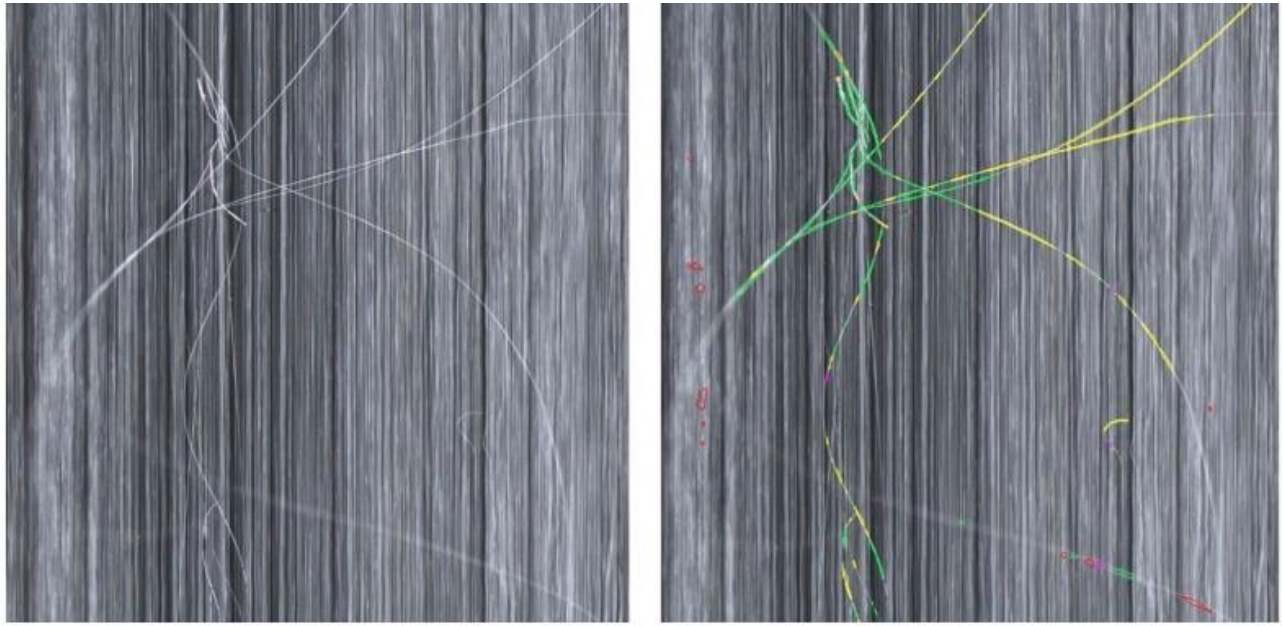
Einfachste Möglichkeit ist, Bilder durch „Faltungen“ (engl. Convolutions) zu analysieren:

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

Bilder aus: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Bildverarbeitung

Mit Faltungsmatrizen kann man Bilder analysieren:
Beispiel links Kratzer erkennen, Beispiel rechts Bakterienstämme zählen



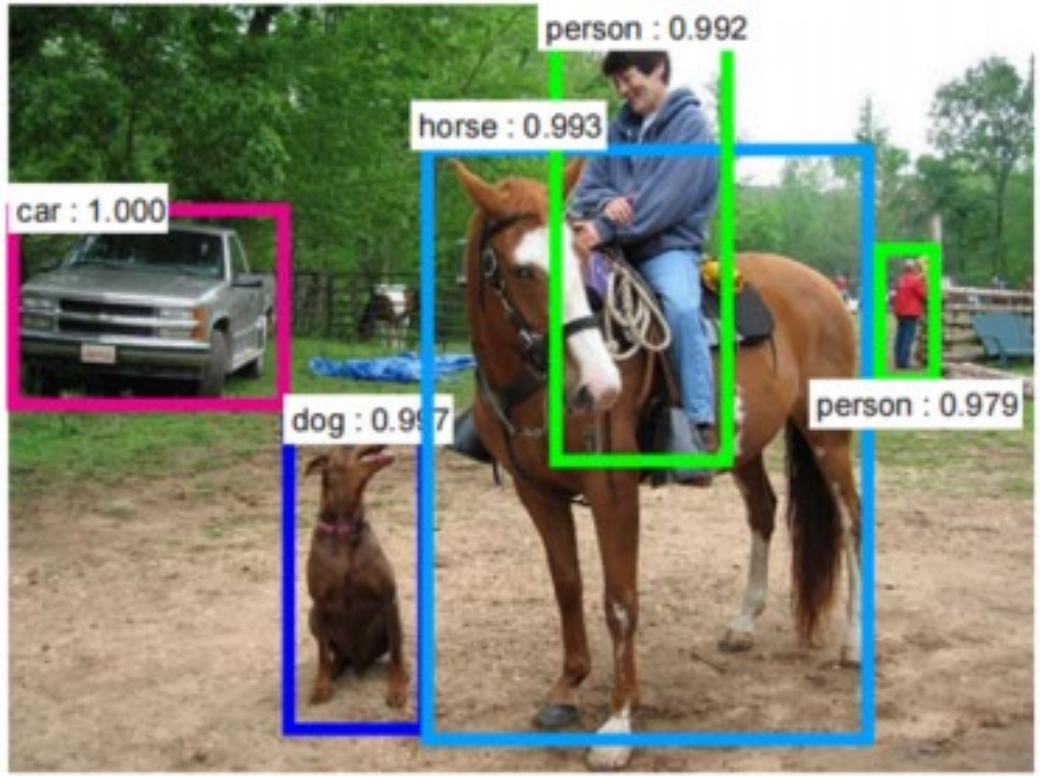
© Fraunhofer IGCV
Aufnahme eines Faserbündels auf Carbonfaserteppich
Links: Originalaufnahme
Rechts: Segmentierter Defekt

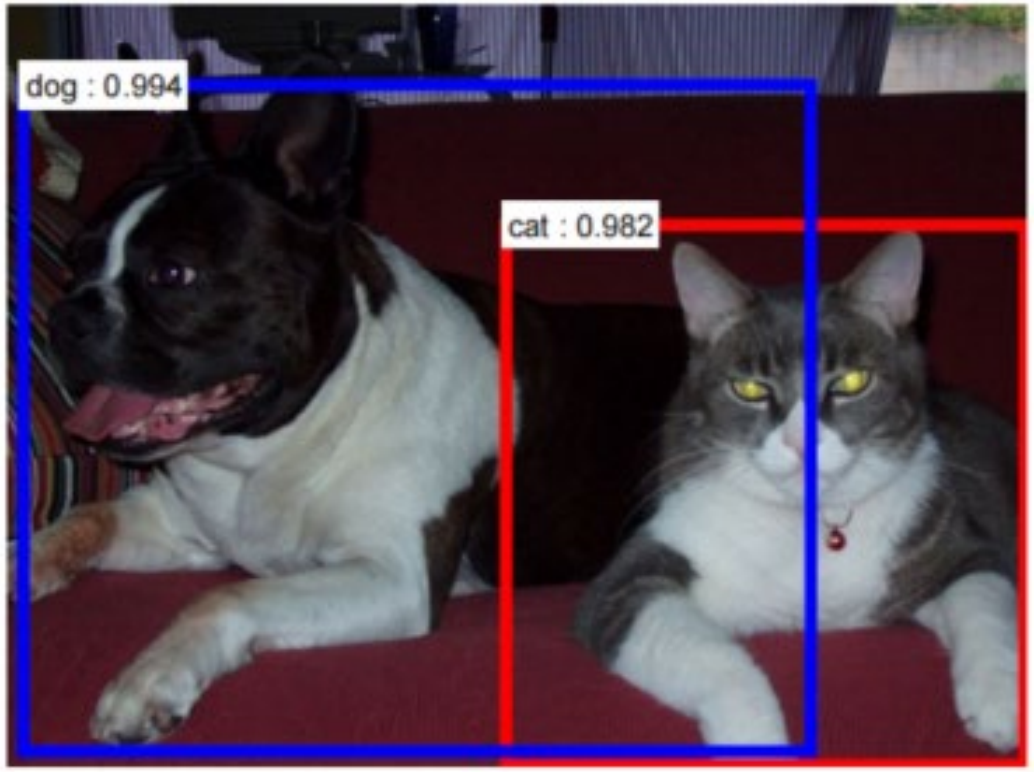
Bild aus: Ilchev Trendafil – R&D – Beissbarth GmbH | LinkedIn

Bild aus: Bilderkennung - GedonSoft GmbH

Bildverarbeitung

Mit einem Netzwerk aus Faltungsmatrizen, den Convolutional Neural Networks, kann man komplexe Objekte im Bild oder Video erkennen.





Bilder aus: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Convolutional Neuronal Network (CNN bzw. ConvNet)

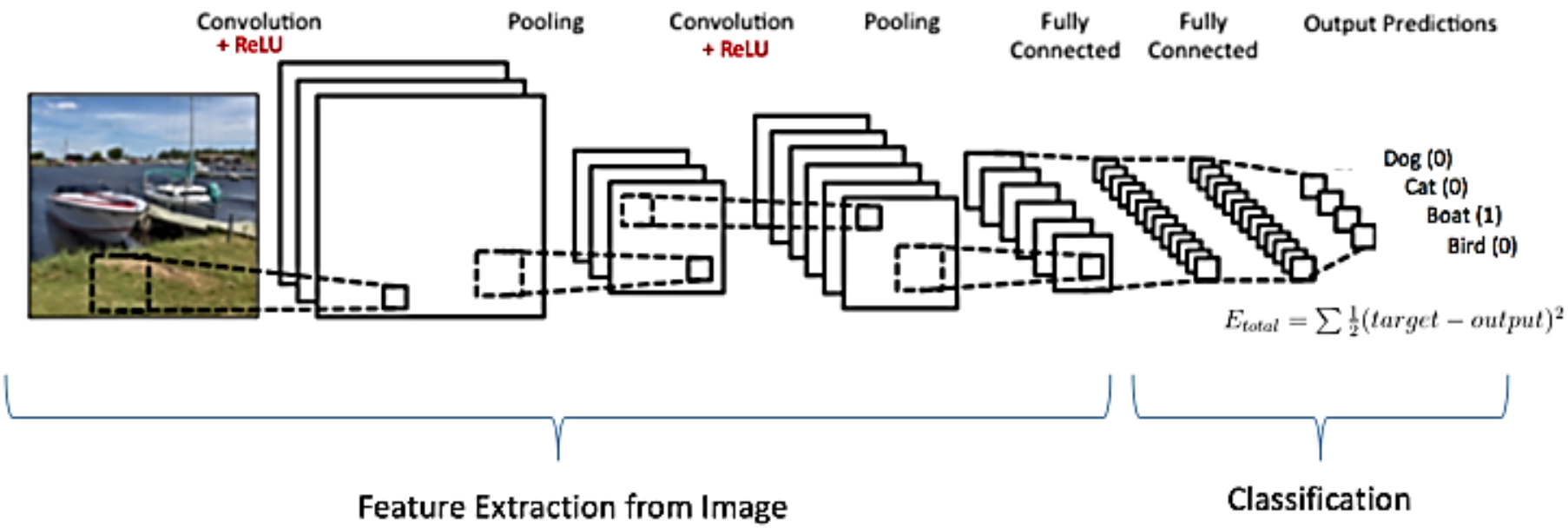


Bild aus: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

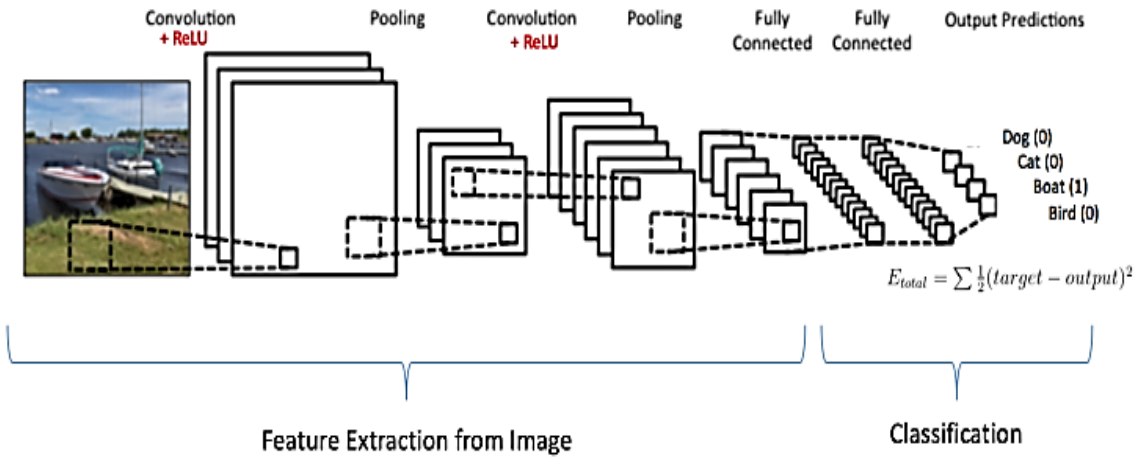
Bildverarbeitung

Das funktioniert auch für 3D-Objekte ☺

[3D-Net: Monocular 3D object recognition for traffic monitoring]

Visualisierung der Funktionsweise eines CNN

Durch die Faltungsschichten werden die Eingabebilder schrittweise in eine abstrakte Beschreibung überführt. Ähnliche Bilder haben ähnliche abstrakte Beschreibungen.



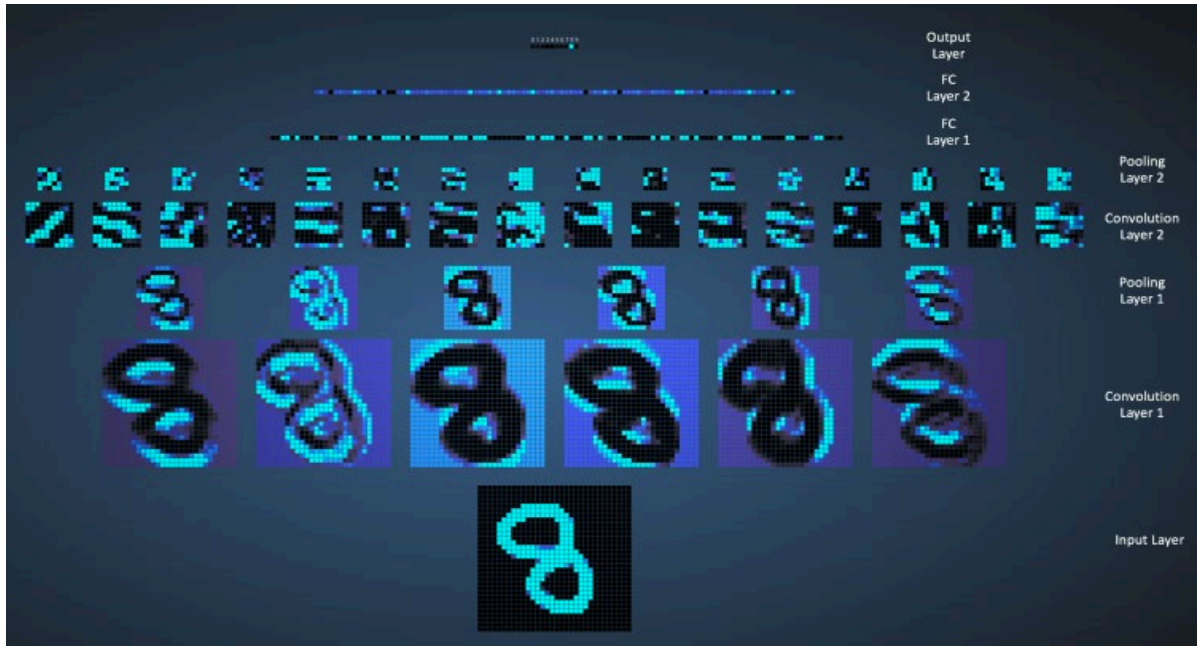


Bild aus: https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

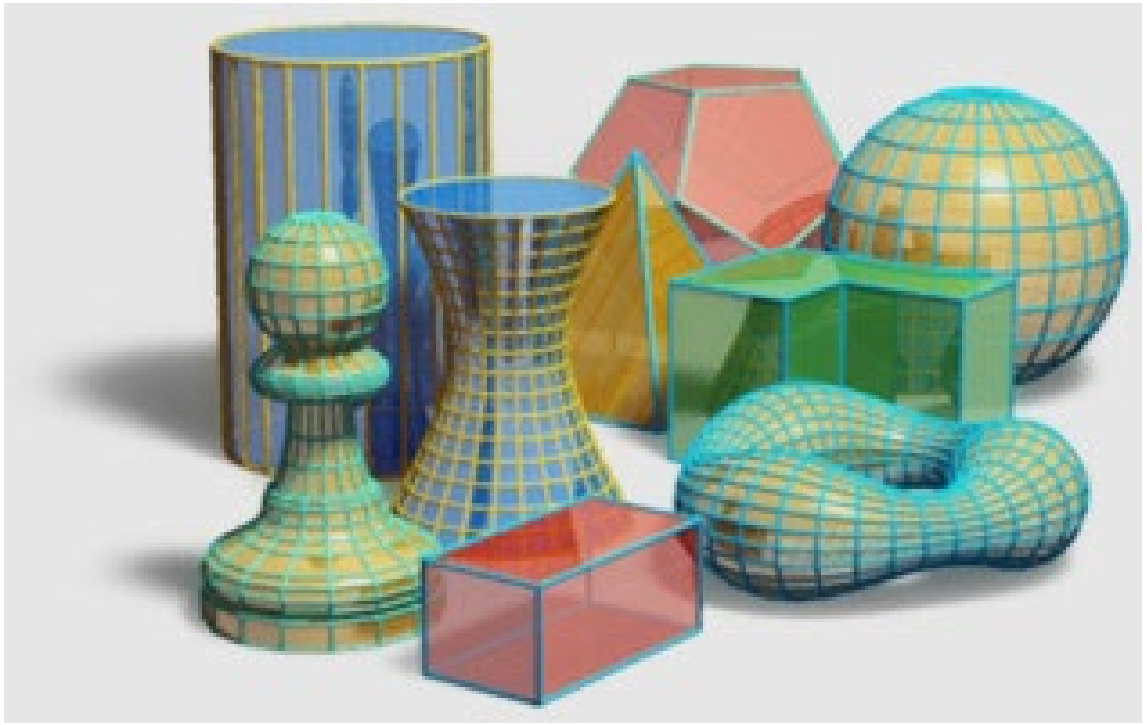
Bild aus: http://cs231n.github.io/convolutional-networks/

selbst ausprobieren: https://adamharley.com/nn_vis/cnn/3d.html

KAPITEL 2

Grafische Objekte

Erzeugung grafischer Objekte



Alle grafischen Objekte bestehen letztendlich aus Punkten, die mit Kanten zu Dreiecken verbunden werden.

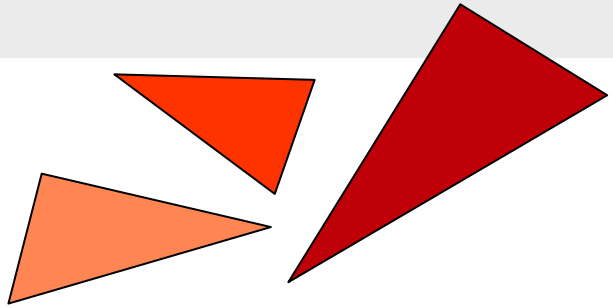
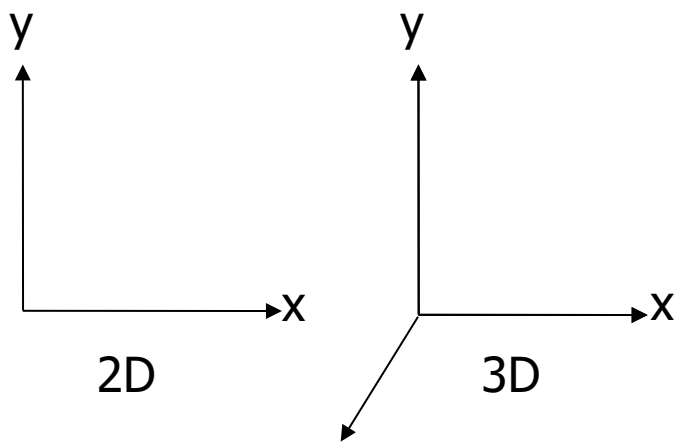
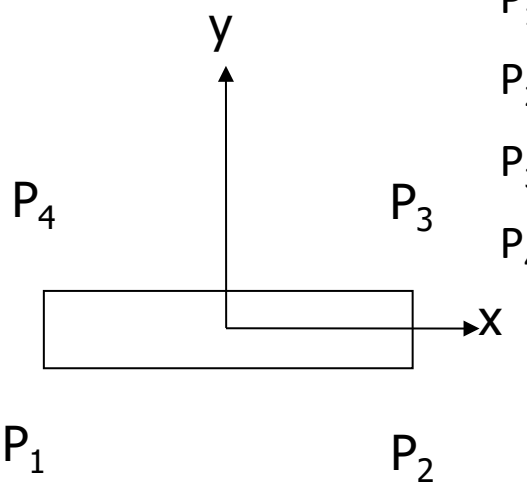


Bild entnommen aus: http://www.pytha.de/produkt/modeler_1.de.php

Koordinatensysteme



Kartesische Koordinatensysteme



$P_1 = \{-2.4, -1.2\}$

$P_2 = \{ 2.4, -1.2\}$

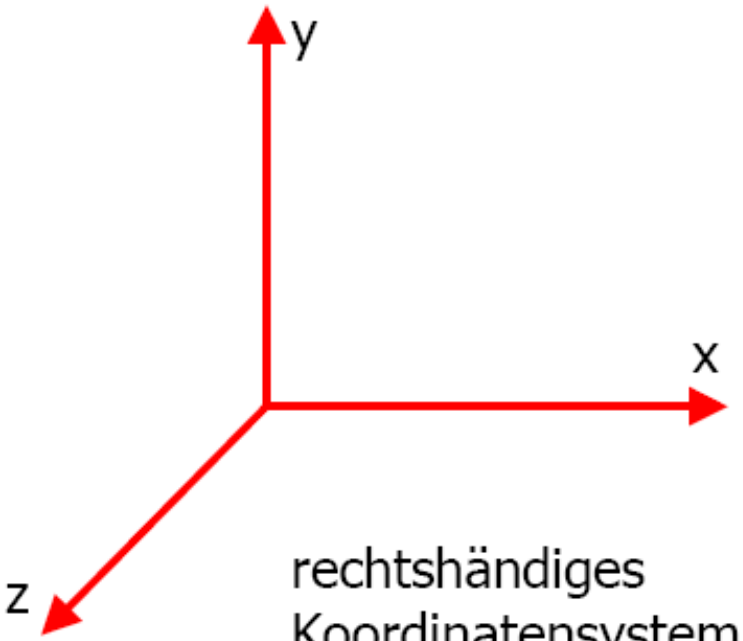
$P_3 = \{ 2.4, 1.2\}$

$P_4 = \{-2.4, 1.2\}$

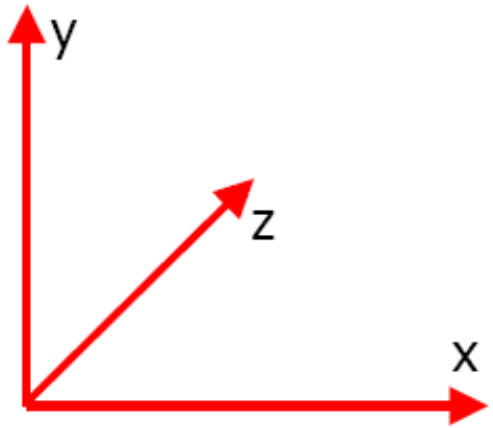
Definition eines Rechtecks in kartesischen Koordinaten

Jede Geometrie wird durch Punkte, Kanten, Flächen in einem bestimmten Koordinaten-system definiert

Kartesische Koordinatensysteme

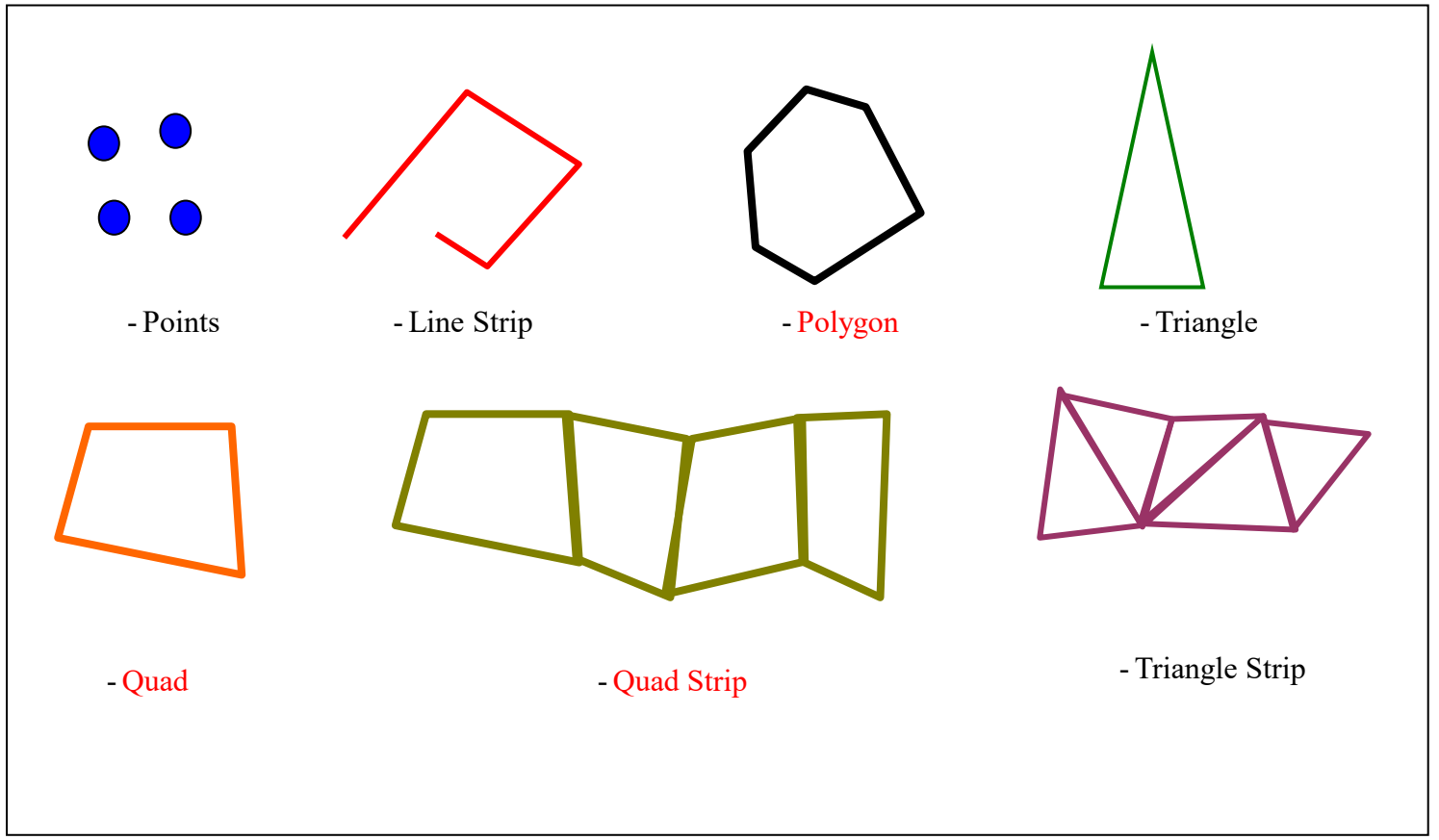


rechtshändiges
Koordinatensystem



linkshändiges
Koordinatensystem

Primitive und Objekte



OpenGL verbindet die Punkte zu bestimmten sog. Primitiven

Rot = deprecated in OpenGL 3.0, removed in OpenGL 3.1+

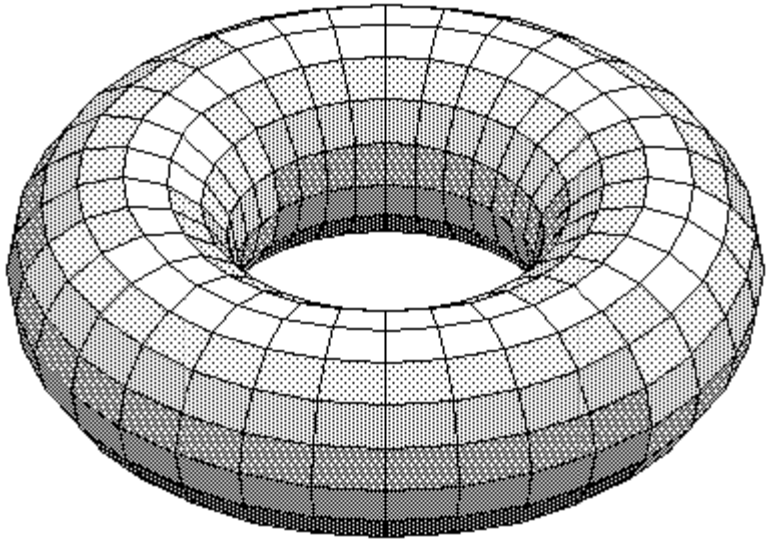
Außerdem:

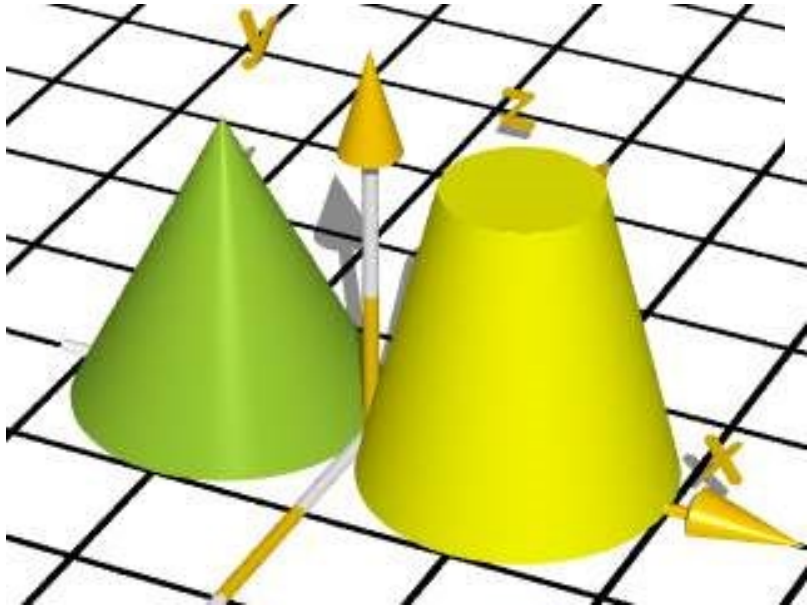
- Lines
- Line Loop
- Triangle Fan
- Patches (nur Tesselation)

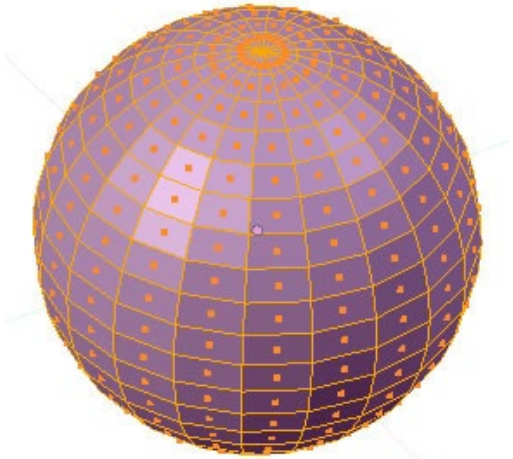
https://wikis.khronos.org/opengl/primitive

Primitive und Objekte

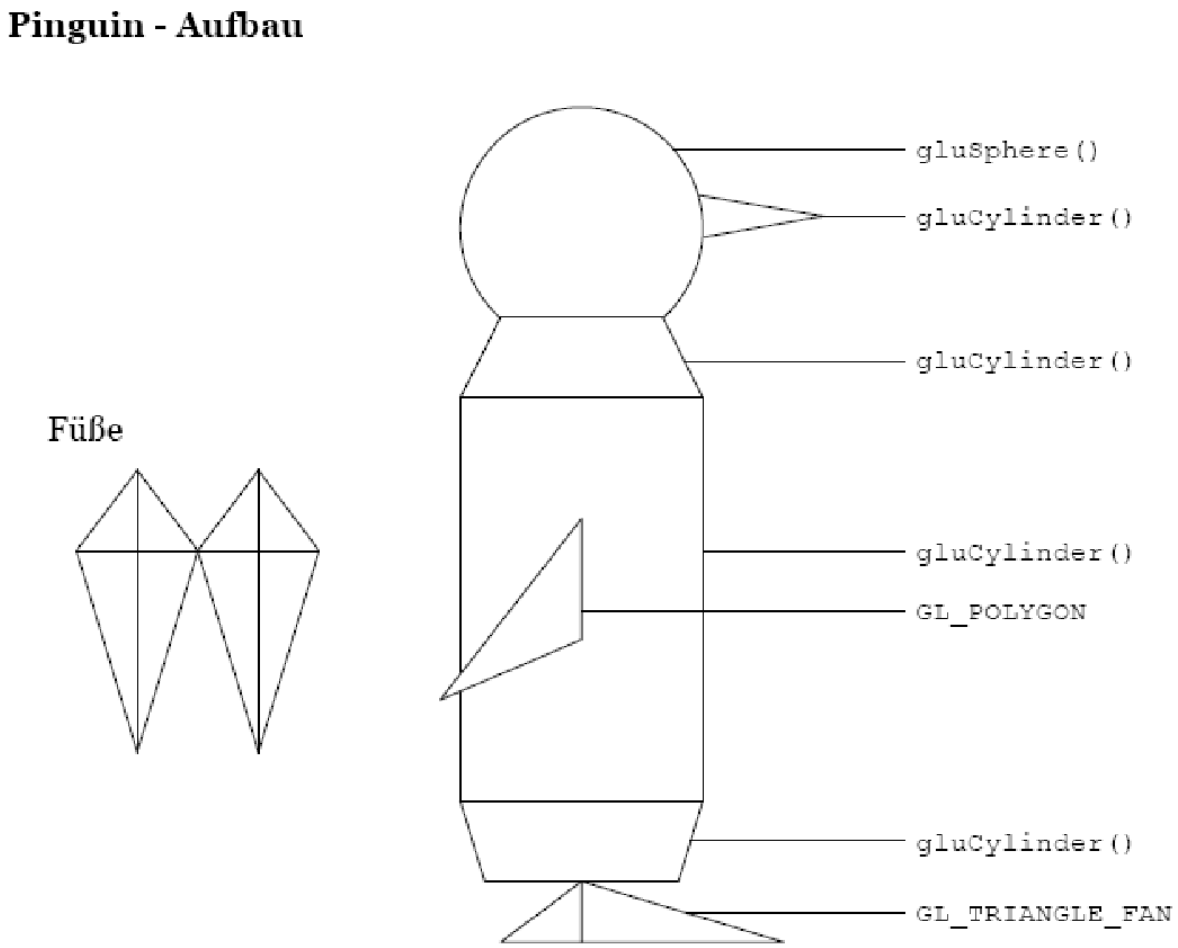
Hilfsbibliotheken stellen komplexere Geometrien zur Verfügung

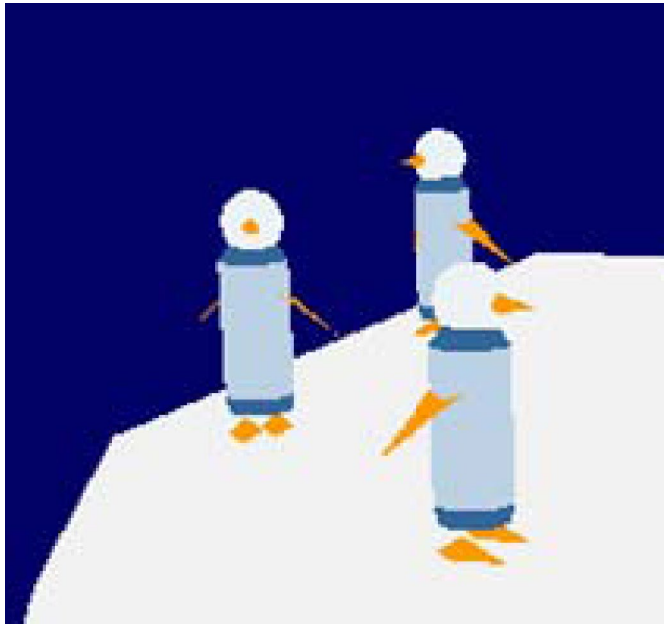






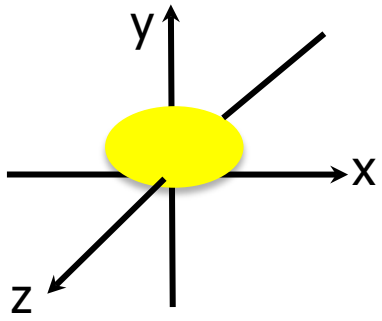
Erzeugen einer Geometrie am Beispiel eines Pinguins





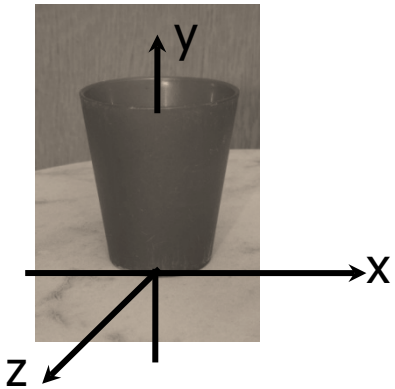
Erzeugen einer Geometrie am Beispiel eines Pinguins

Pinguinkopf im lokalen Koordinatensystem



Dreieck für die Flügel und die Füße im lokalen Koordinatensystem

Becher im lokalen Koordinatensystem



Pinguin im Welt-koordinatensystem

Erzeugen einer Geometrie am Beispiel eines Pinguins

Was fehlt noch zur vollständigen Modellierung des Pinguins?

- Neben den geometrischen Größen müssen noch die **grafischen Attribute** zugeordnet werden
  - Grafische Attribute beziehen sich auf das „**Aussehen**“ der Objekte:

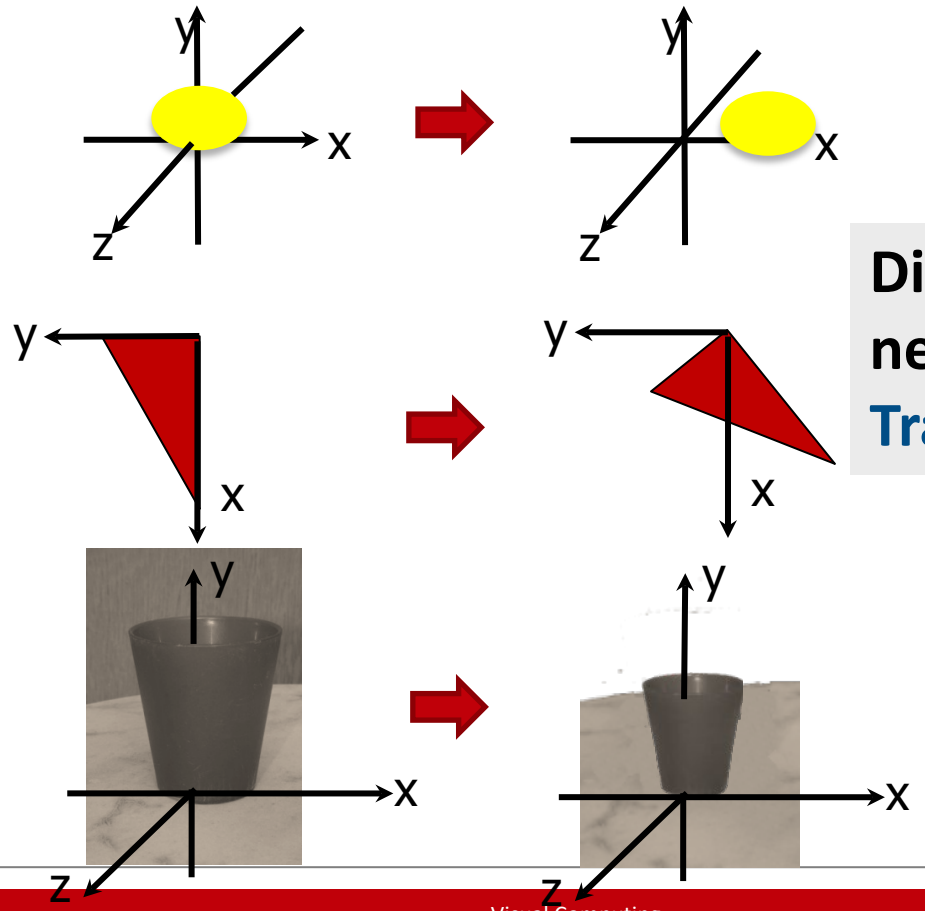
- Beispiel für graphische Attribute:
  - Farbe
  - Synthetische Textur auf die Fläche gemappt (=„geklebt“)
  - JPEG-Bild auf Fläche gemappt
  - …

Erzeugen einer Geometrie am Beispiel eines Pinguins

Wie kann die Primitive im lokalen Koordinatensystem „manipulieren“, um einen Pinguin zu konstruieren?

Man kann sie:

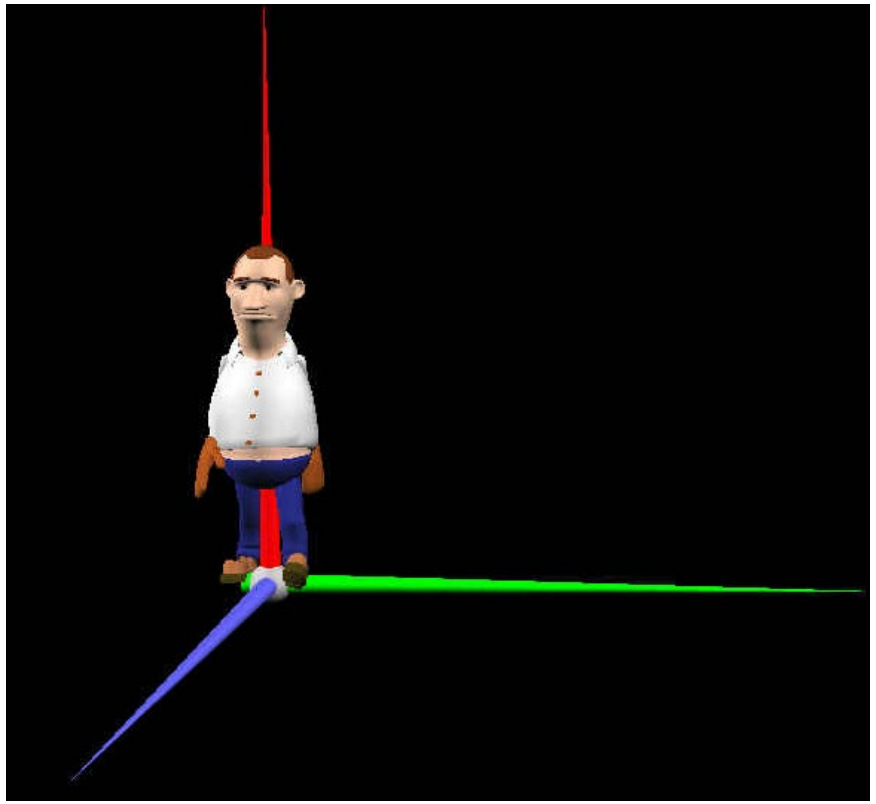
- verschieben (**translieren**)
- drehen (**rotieren**)
- vergrößern und verkleinern (**skalieren**)

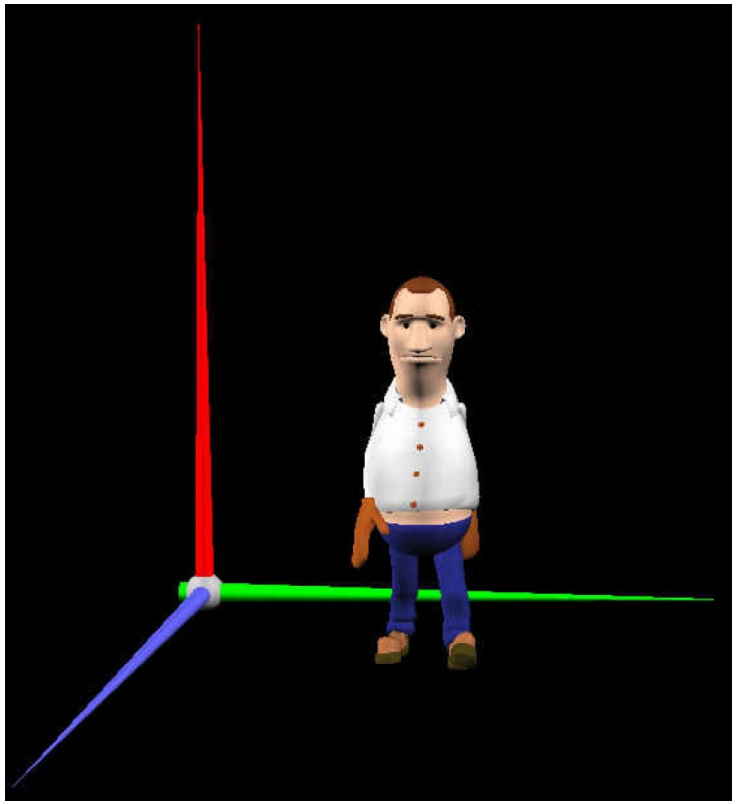


Diese Manipulationen nennt man Transformationen!

Transformationen: Translation

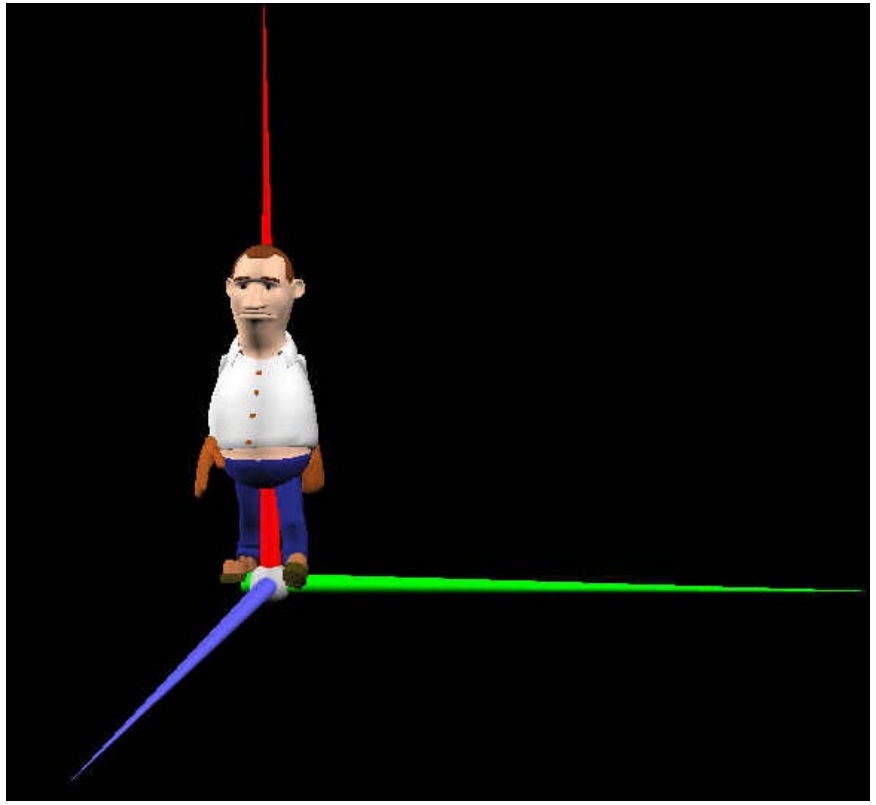
In welche Richtung wurde die Geometrie bei der Translation verschoben?



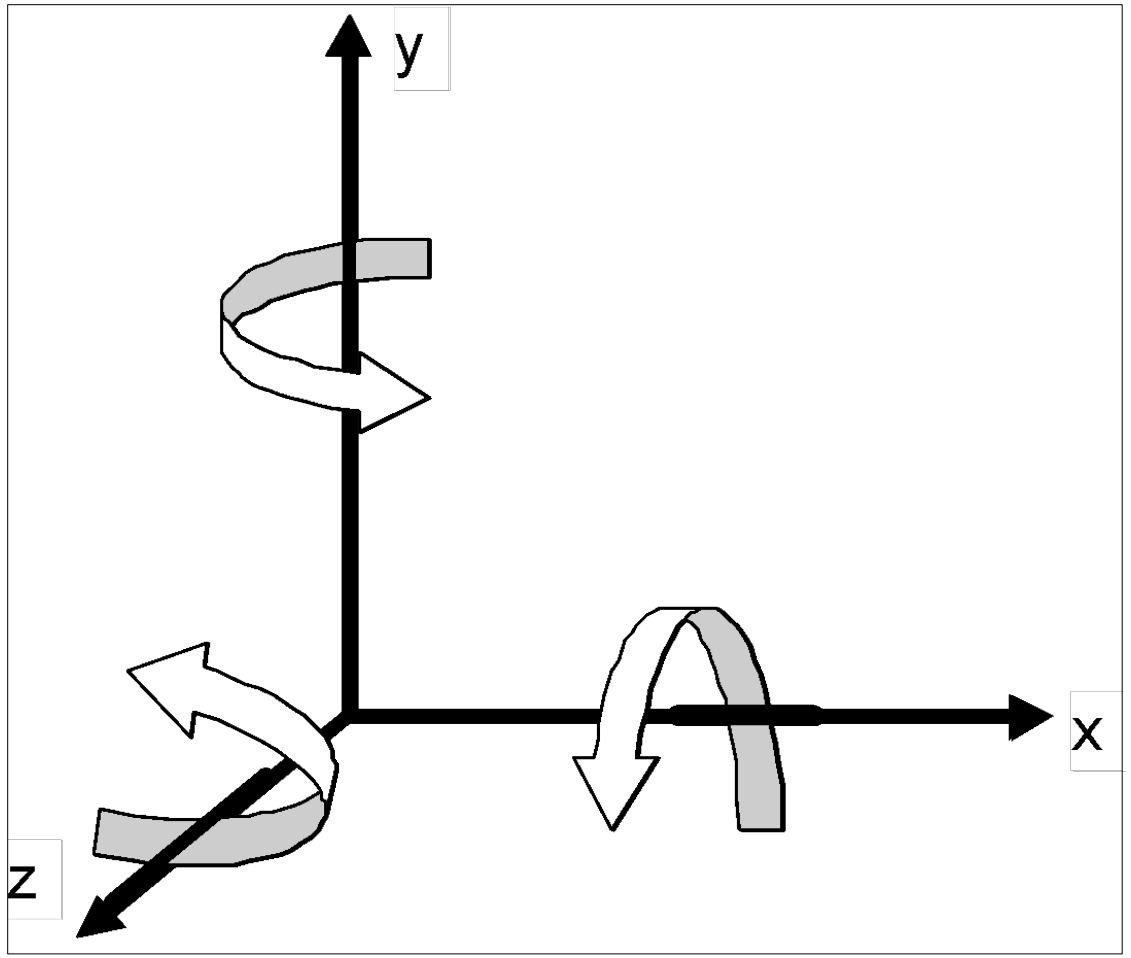


Transformationen: Rotation

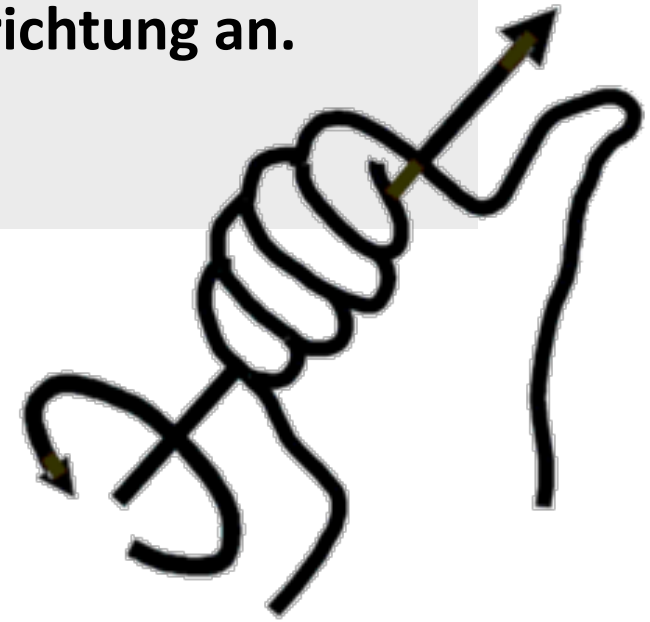
Um welche Achse wird die Geometrie bei der Rotation gedreht?



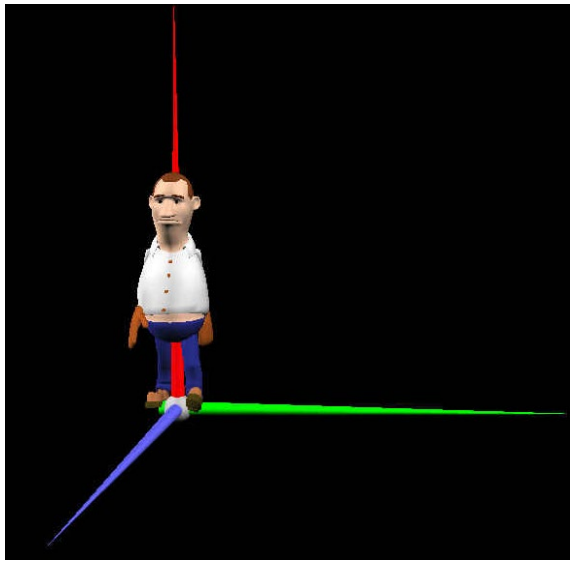
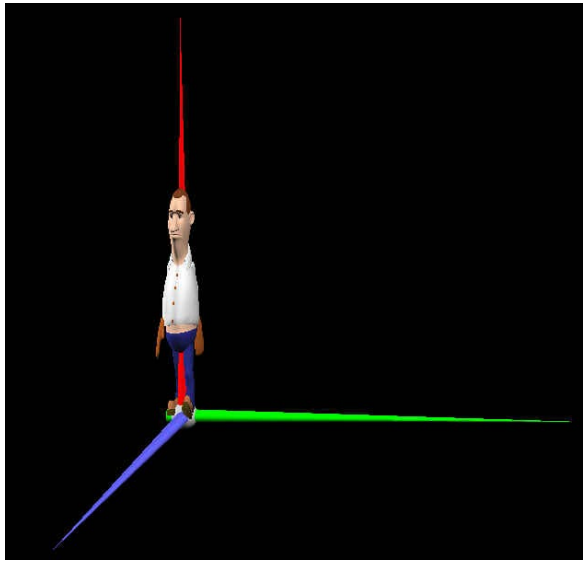
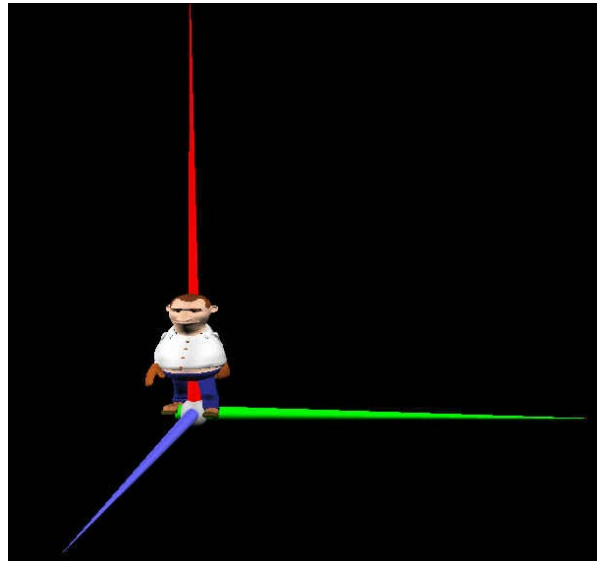
Demonstration des mathematisch positiven Drehsinns



Wenn der Daumen der rechten Hand die Koordinatenachse bildet, zeigen die angewinkelten Finger die Drehrichtung an.



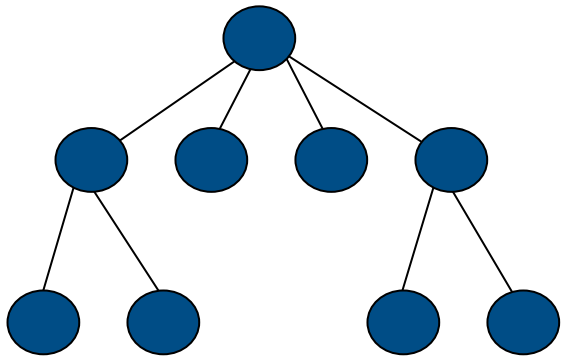
Transformationen: Skalierung

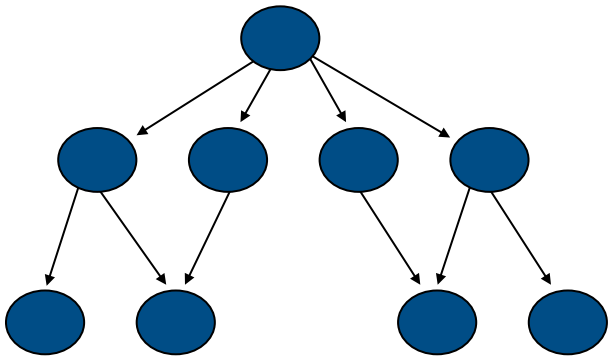
Wie verhält sich die Figur bei der Skalierung, wenn sie nicht am Nullpunkt definiert wurde?

Transformationen: der Szenengraph

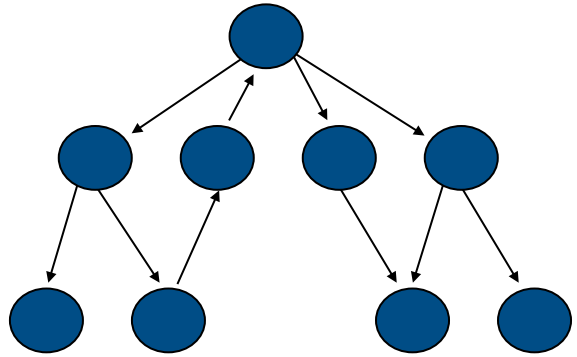
Unterschied Baum und Graph



Baum



gerichteter
zyklenfreier Graph



gerichteter
nicht-zyklenfreier Graph

Transformationen: der Szenengraph

Der Szenengraph besteht aus mindestens 3 Knotentypen:

Gruppen

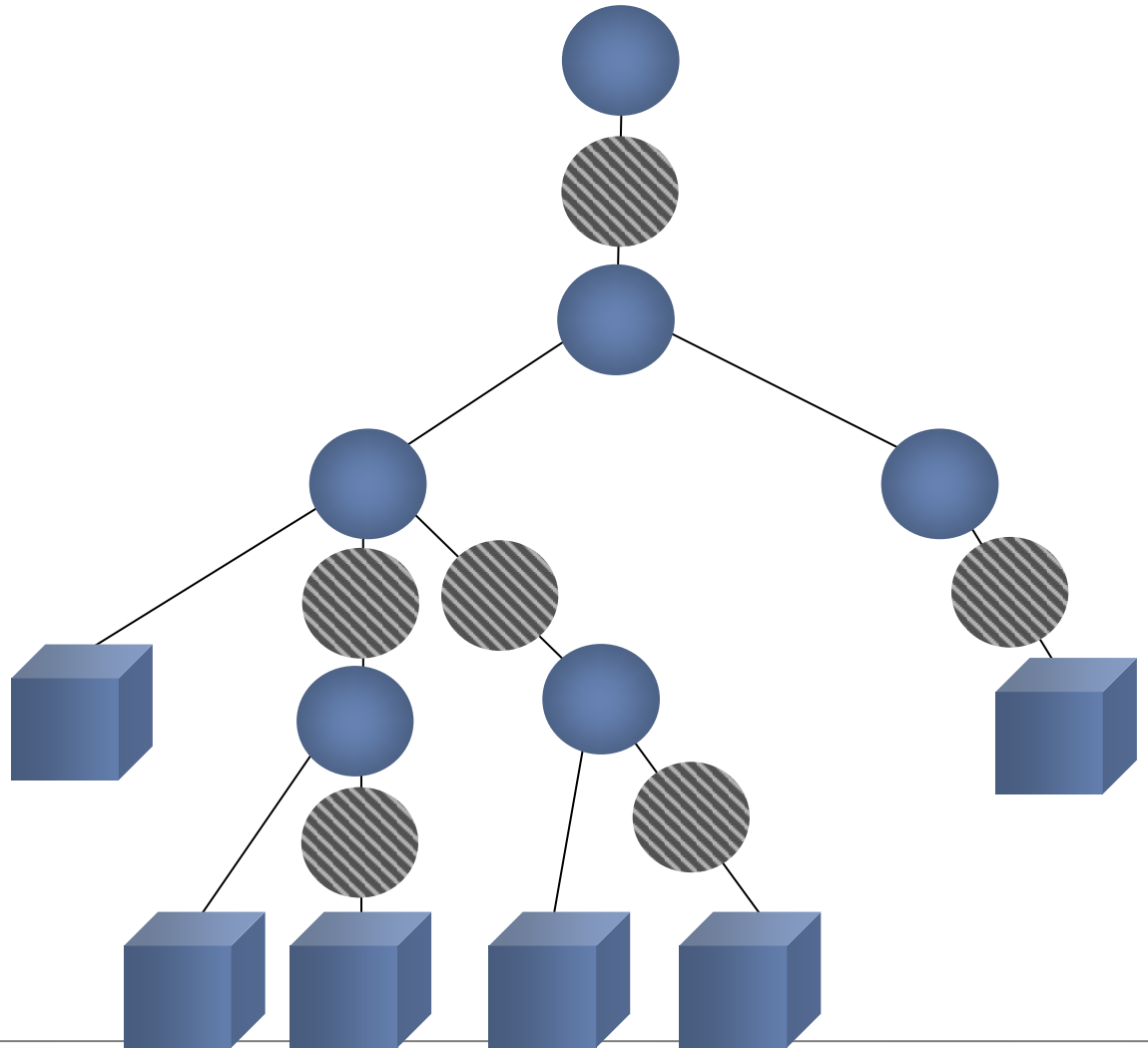
Geometrien (inkl. Materialeigenschaften)

Transformationen

Der Szenengraph dient zur Verwaltung einer komplexen Szene:

- Gruppierung von Geometrien zu Gruppen
- Gruppierung von Gruppen zu Gruppen
- Gruppierung von Gruppen zu einer Szene

Szenengraphstruktur



- gerichtet
- azyklisch
- heterogen
- Geometrie / darstellbare Primitive in den Blättern

Welchen Vorteil hat ein gerichteter azyklischer Szenengraph gegenüber Szenengraphen, die als Bäume realisiert wurden?

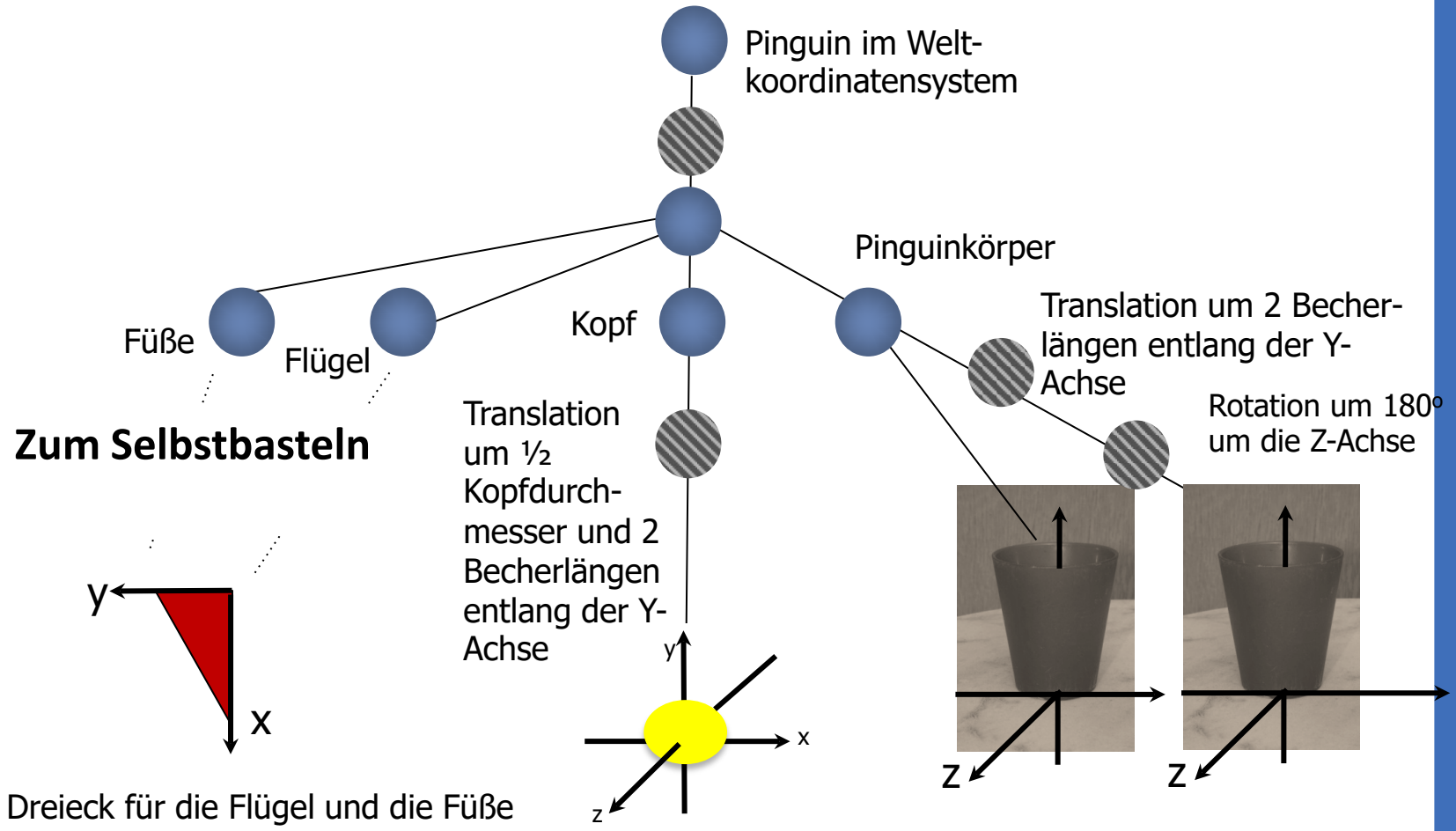
(Sehen wir dann beim Pinguin)

Erzeugung des Pinguins mit Hilfe eines Szenengraphen

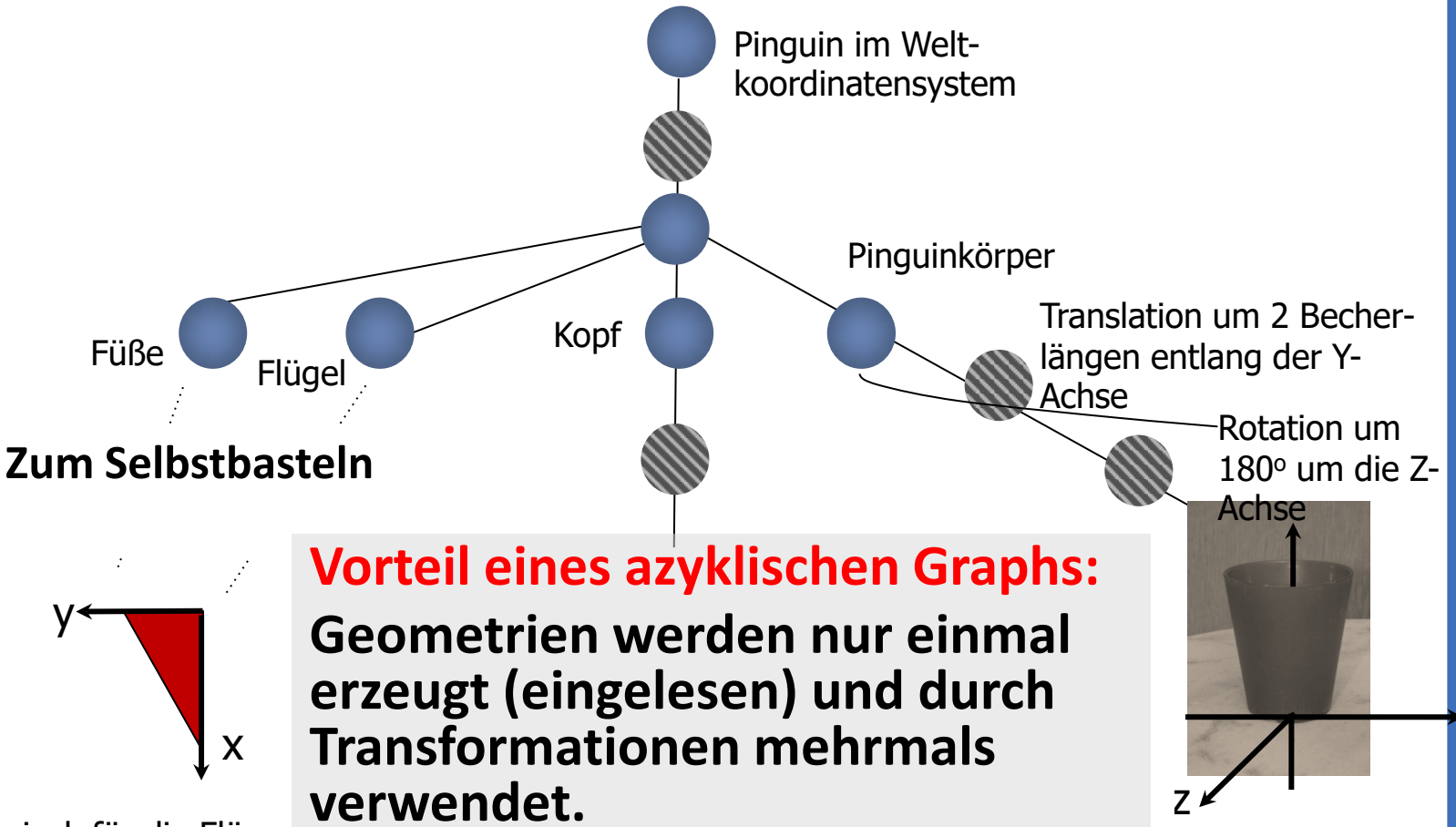
Folgende Primitive stehen zur Verfügung:

- Becher mit der Öffnung nach oben
- Ball (bereits mit Augen und Nase)
- Ein Dreieck

Erzeugung des Pinguins mit Hilfe eines Szenengraphen



Erzeugung des Pinguins mit Hilfe eines Szenengraphen



Vorteil eines azyklischen Graphs:

Geometrien werden nur einmal erzeugt (eingelesen) und durch Transformationen mehrmals verwendet.