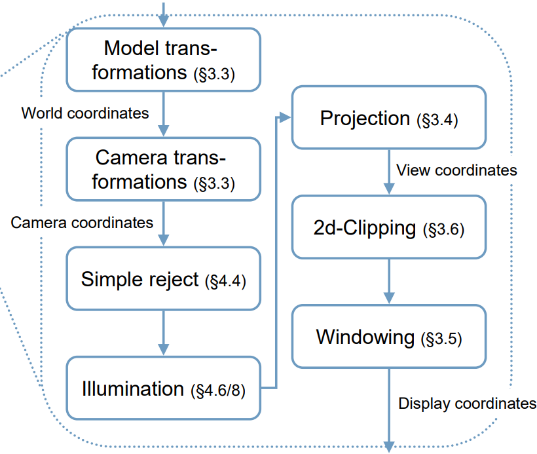
Rendering: Mapping den Geometrische Modellen, Objekten oder der Szene auf ein Bild auf dem Display.

Rendering Pipeline: Konkrete Implementation des Rendering in Soft oder Hardware.

De-Facto Standard Pipeline: Application – Geometry – Rasterization – Fragments – Display

Application

* Receiving and sending of commands and data
* Command buffering/interpretation

Geometry

* Model transformations
* Camera transformations
* Simple reject (Back-Clipping)
* Illumination
* Projection
* 2d-Clipping
* Windowing

Rasterization

* Primitive assembly (Triangles, lines etc)
* Triangle setup (Tiefen/-Farbwerte, Textur Koordinaten)
* Rasterization
* Anti-Aliasing
* Z-Buffer

Fragments

* Transparency
* Texturing

Display

* Frame buffer
* Gamma-correction
* Analog/digital-converter

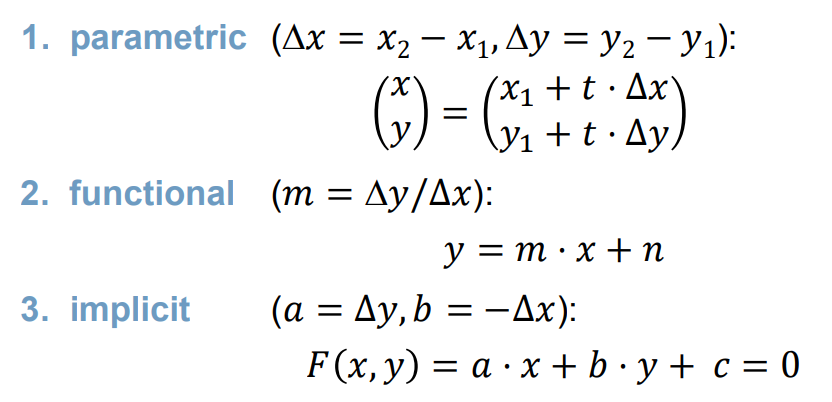
**Kapitel 2 – Rasterization**

Dekomposition von geometrischen 3D Objekten in ein 2D Image mit im Raster angeordneten Pixeln.

Problem: Linie die abgebildet werden soll überlappt die Pixel mal mehr, mal weniger.

Anforderungen:

* Linie muss konstante Dicke, Helligkeit und keine Lücken haben
* Endpunkte muss exakt sein
* Berechnung muss schnell sein, Algorithmus muss sich in Hardware implementieren lassen.

1. Stütz + Richtungsvektor + Parameter t
2. Steigung m \* X-Wert + Y-Offset n
3. Nach Y umstellen, oder Punkt einsetzen

zum Prüfen

**Naive algorithm**:

Probleme:

Floating point values 𝑦 and 𝑚,

divisions and multiplications,

rounding,

vertical lines,

appearance of lines depends on the slope 𝑚.

Use functional representation: 𝑦 = 𝑚 \* 𝑥 + 𝑛

1) Iterate from 𝑥1 to 𝑥2 in pixel-steps

2) Compute 𝑦-values

3) Round

4) Draw

**Bresenham Algorithmus:**

Aufteilung in Oktanten, nur im 1. Zeichnen. Bei Linie in anderen Oktanten die Gerade durch Transformation in den 1. Verlegen.

Entscheidungsvariable d gibt an, ob Mittelpunkt zwischen nächsten Pixelkandidaten (NE/E) über oder unter der Linie liegt. ( d >= 0 => Mittelpunkt unter Linie, d < 0 => Mittelpunkt über Linie)

Initialisierung: Umformulierte Gleichung, die prüft, ob Steigung > oder < 0.5 ist. \* 2 um Division zu vermeiden.

**Bresenham Algorithmus für Kreise:**

Diesmal im 2. Oktanten, andere Punkte extrapolieren.

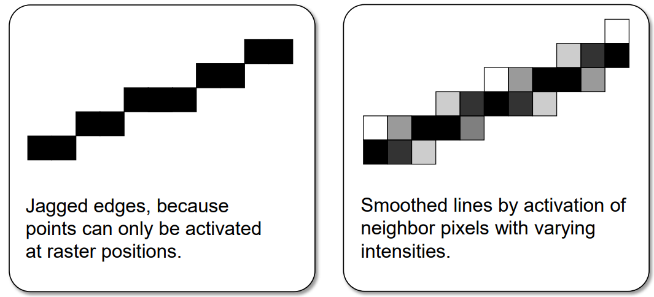
Ähnliches Vorgehen, mit Kreisgleichung prüfen, ob Punkt innerhalb oder Außerhalb vom Kreis liegt.

Implizite Gleichung: 𝐹(𝑥, 𝑦) = (𝑥 – 𝑥𝑀)2 + (𝑦 – 𝑦𝑀)2 − 𝑟2 = 0

Initialisierung von d: d = 5 - 4 \* r;

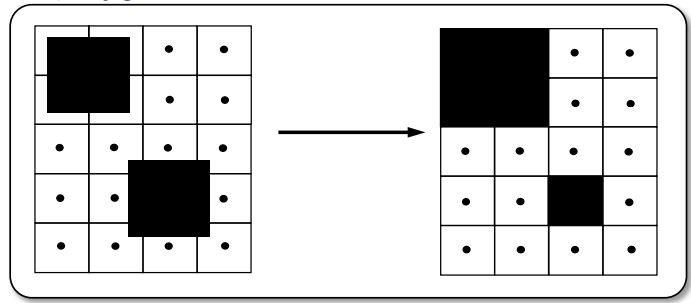
**Aliasing**

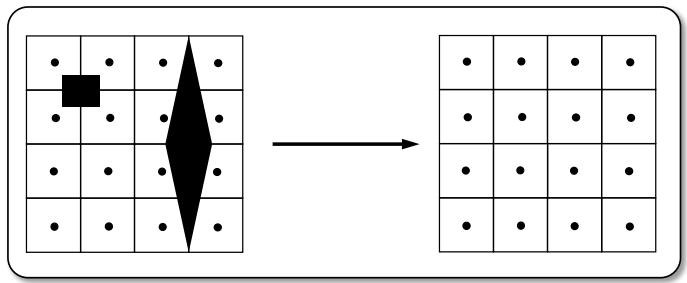
Tritt auf wenn die Abtastfrequenz im Vergleich zum abgetasteten Signal zu klein ist. (Abtastfrequenz = Pixel)

Visual Effects: Aliasing Effekte, ausgelöst durch undersampling. Objekte werden mit niedrigerer, falscher Frequenz dargestellt.

Visual Artifacts (Spatial Aliasing):

* Jagged edges/jaggies

Falsche Größe, Verschwindende Objekte:



Temporal Aliasing: Rad eines Autos scheint während der Fahrt stillzustehen

Anti-Aliasing Techniken:

Edge Smothing: Anti-Aliasing Methoden, die durch Rasterisierung hervorgerufene Artefakte bereinigen

Super-Sampling: Pixel wird in Sub-Pixel unterteilt. Die tatsächliche Pixelfarbe richtet sich nach dem Durchschnitt der Sub-Pixel.

Gewichtete und Ungewichtete overlap area: Korrekten overlap von Line mit Pixeln berechnen => Pixel mit Grauwert\*

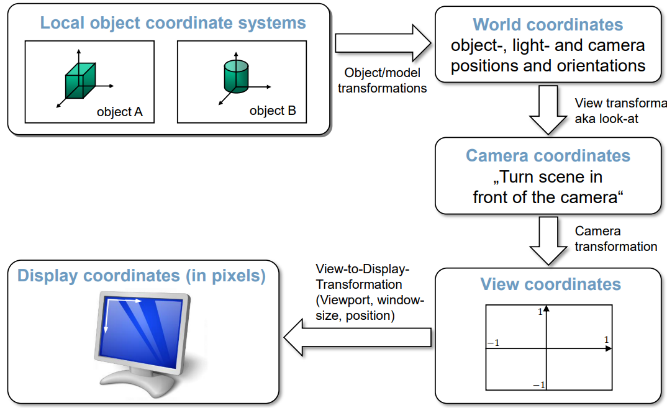
Stochastische Methode: Anti Aliasing nur an Zufälligen Punkten. Effizienter, aber kann bei Animationen flackern

FSAA (Full scene anti-aliasing): Super-sampling for every pixels of the scene, very expensive.

TXAA (Temporal anti-aliasing): Filtering of multiple samples inside and outside the pixel with pixels of previous frames.

DLSS (Deep learning super sampling): A NN is trained (an stored on the driver) to render the anti-aliased high resolution image.

**Kapitel 3 - Transformations and Projections**



A window defines the visible sub-area of the image plane

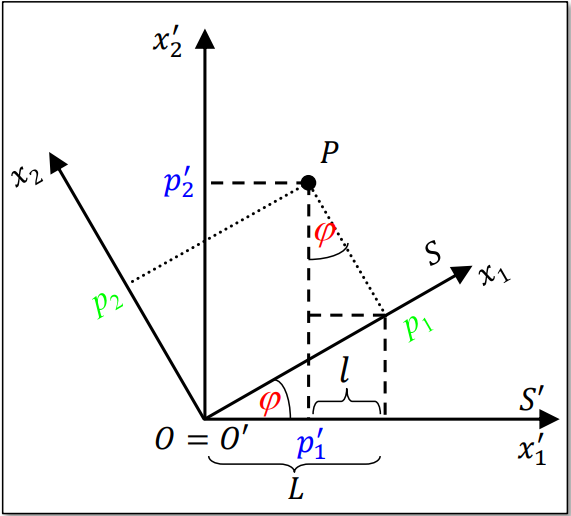
A viewport defines a sub-area of the display, where the content

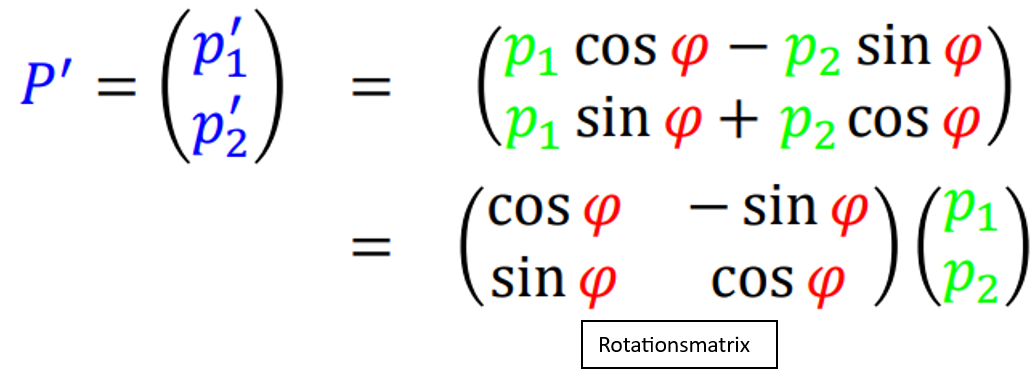
of the window is displayed.

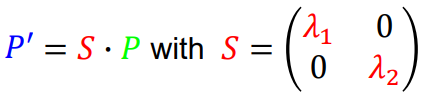
Window-viewport-transformations (windowing) are

2d-2dtransformations.

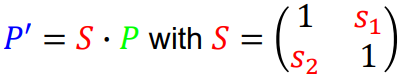
Wir nehmen an, dass ein Koordinatensystem Kartesisch (orthonormal + right handed) ist.

Für Rotation: Translation (Local System to (0,0) of Global System), Rotation (around origin), Back Translation (of Local System) Matrixmultiplikation: Zeile \* Spalte



**Matrix multiplication is not commutative.**

Skalierung eines Koordinatensystems:



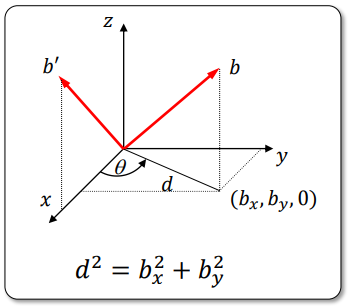
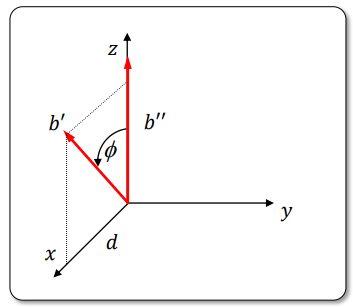
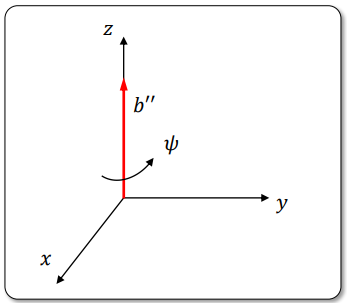
Scherung eines koordinatensystems:

Under an affine transformation parallel lines remain parallel.

Windowing: Translation of image into the (World)-origin, Scaling to size of viewport, Translation of image to viewport position

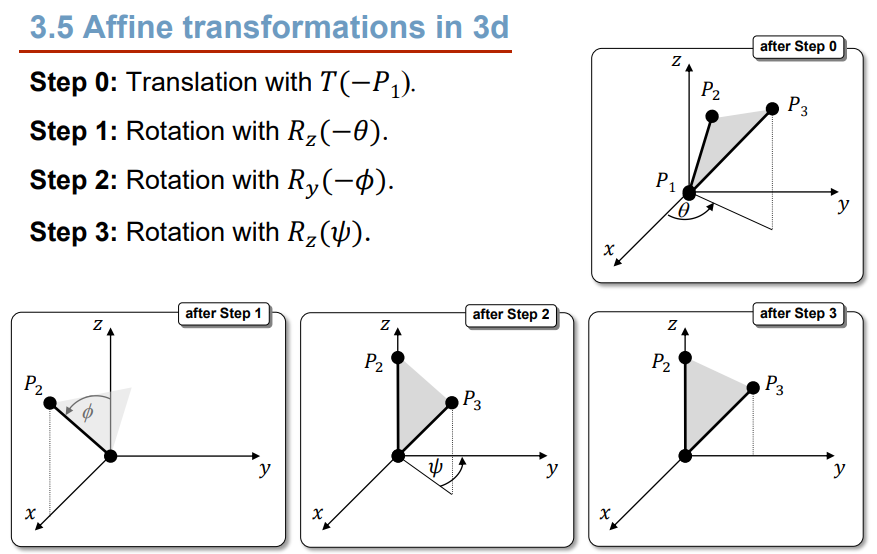
Transformation in die nächst höhere Dimension heben um auch für Translation Matrixmultiplikation nutzen zu können.

***Rotation GEGEN Uhrzeigersinn***

Rotation um beliebige Achse G: G um Z und Y rotieren, damit sie auf Z liegt. Eigentliche Rotation um Z ausführen. Punkt wieder um Y und Z zurück rotieren.

Falls G nicht durch Ursprung geht

Translation auf Ursprung, zum Schluss

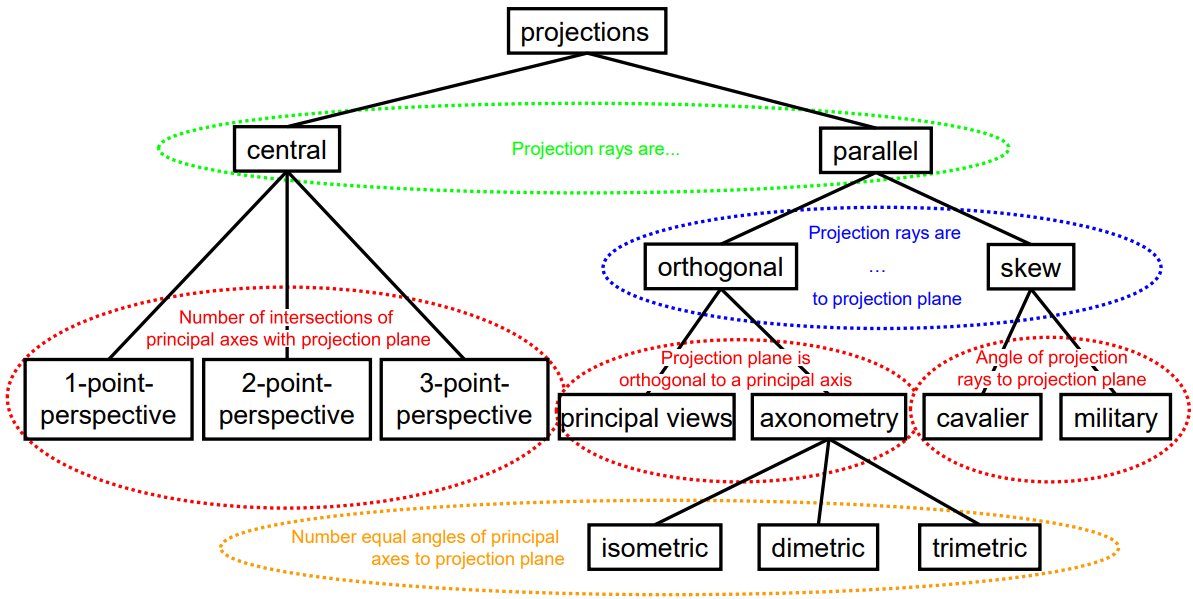
Backtranslation. G muss normalisiert sein.  
Wenn ein Objekt in m Globalen System um sich selbst  
gedreht werden soll (Orientation ändern), dann benötigt   
man dafür das Lokale System. Durch 3 (nicht kolineare)  
Punkte kann dieses Rekonstruiert werden.

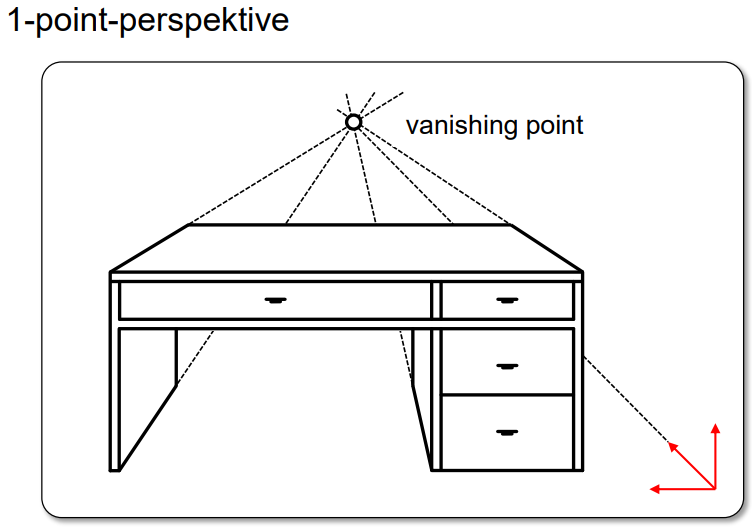
**Euler Winkel**: Stellen die Orientierung eines Objektes  
Im 3D Raum dar. Am intuitivsten.

**Gimbal Lock**: Tritt auf, wenn sich bei Rotation um Euler  
Winkel 2 Achsen Alignen, so dass man einen Freiheitsgrad verliert.

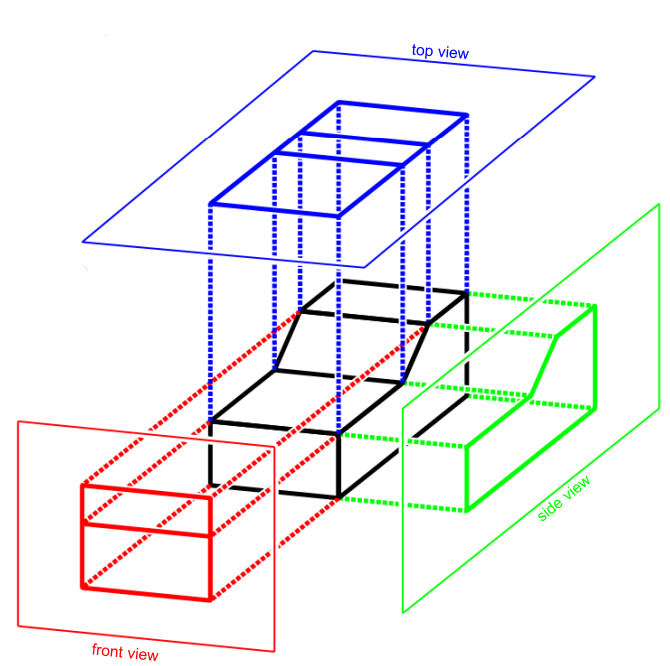
**Quaternions**: Alternative Darstellung einer Rotation.  
Theorie: Einheitsvektor stellt Rotationsachse G dar, Orthogonal zur Achse geht eine Ebene durch P. P wird (in der Ebene) um Winkel θ um die G rotiert.  
Normalisiertes Quaternion kann dargestellt werden als: 𝑞 = (cos(𝜃/2), sin(𝜃/2) 𝒏), wobei 𝒏 = G.  
Für Rotation Quaternion Produkt: 𝑞 ∘ (0, 𝒑) ∘ 𝑞′ 𝑞′ = q konjugiert.  
Attention: The quaternion multiplication is not commutative!

**Projection:** A projection is a mapping, that maps a space of dimension 𝑛 to a space of dimension < 𝑛.



****

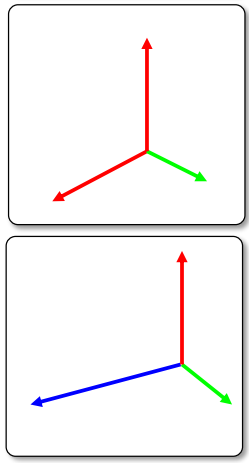
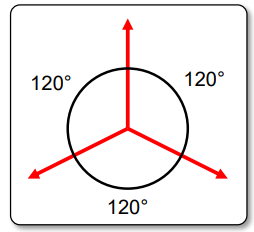
**Perspective projection / central projection**: „Normal“,   
wie in der Übungsaufgabe. Parallele Linien die nicht Parallel  
 zur Projektionsebene sind laufen in einem Fluchtpunkt   
zusammen. Klassifiziert nach Anzahl der Hauptachsen (x,y,z)   
die durch die Projektionsebene gehen   
(also nicht parallel dazu sind). Schreibtisch von   
Vorne = 1-Punkt-Perspektive, da zwei Hauptachsen  
 (Hoch/Runter + Links/Rechts).

**Parallel projections:** Spezialfall der Central Projection  
(Projektionspunkt = ∞). Weniger realistisch, dafür sind Messungen  
möglich.

**Orthographic projections:**

**Principal views:** Technische Zeichnung, wie 1-Punkt-Perspektive,  
aber ohne die nicht parallele Dimension. Winkel und Distanz  
bleiben erhalten.

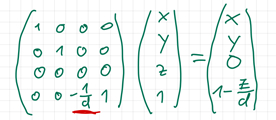
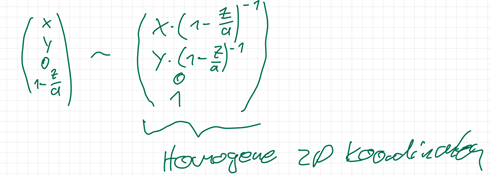
For an **axonometry** the projection plane is not orthogonal  
to any of the coordinate axes (of the world system).

* Parallel lines are mapped to parallel lines.
* Angels are not preserved.
* Distances can be measured along principal axes,   
  although using different scales for each axis.

Isometric, dimetric , trimetricaxonometry:

Winkel zwischen Allen (3), 2 bzw. keiner Achse und der   
Projektionsebene sind gleich.

Projektionsmatrix wird auf Punkt losgelassen



The projectors through the corners of the view window define the socalled viewing frustum.

Front and back clipping plane

Clipping:

Beide Endpunkte im Fenster: Plotten

Beide Endpunkte Oberhalb/Unterhalb/Links/Rechts vom fenster: nicht Plotten

Ansonnsten: Compute the intersections of the line with the window boundaries and determine the visible part of the line.