Interaktive Computergrafik

Michael Gabler

20. Juli 2019

Inhaltsverzeichnis

1 Grundlagen

Computergrafik beschreibt das Erstellen von 2D-Bildern aufgrund von 3D-Daten.
Anwendungsgebiete

- Human-Computer-Interaction
- CAD & (wissenschaftliche) Visualisierung
- Filme
- Computer Spiele

3D-Repräsentation Wie können Objekte als 3D-Modell abgebildet werden?

- Implizite Parameter (z.B. als Funktion)
- Oberfläche annähernd beschrieben durch Dreiecke oder Polygone (manuell, Laser Scanner, Fotos von allen Seiten)
- Volume Solids (z.B. durch Sensoren wie MRT oder CT)

Animation z.B. über Referenzpunkte, die mit echter Welt gemappt werden
Rendering Abbilden von 3D-Daten auf 2D-Repräsentation z.B. durch Raytracing oder Rasterization
Immersion Maß in wie weit eine virtuelle Darbietung äußere, reale Wahrnehmungen ausgrenzt und diese durch virtuelle ersetzt.
Präsenz/Presence In wie weit fühlt sich ein Subjekt in einer Umgebung angekommen/eingebungen auch wenn es sich in einer anderen befindet.
Digitalisierung analoger Signale

Digitization

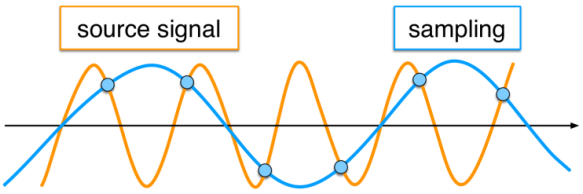


Figure: Sampling an analog wave.

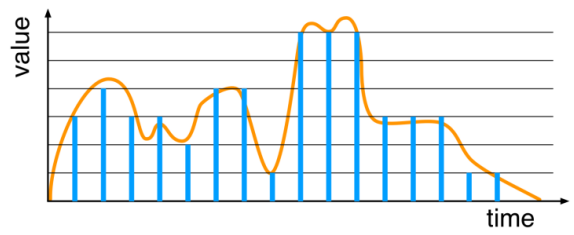


Figure: Quantization of values.

Conversion process of information into a digital (i.e., computer-readable) format, in which the information is organized into bits.

1. Discretization

- Reading (sampling) of an analog signal at regular intervals (frequency).
- Each reading (sample) may be considered to have infinite precision at this stage.

2. Quantization

- Approximating/rounding samples to a fixed set of numbers (such as integers).

Rastergrafik Grafik wird als Pixel beschrieben, die jeweils eine Farbe haben → Skalierung schwierig. Beispiel: JPG, PNG, GIF, TIFF, PBM

Vektorgrafik Inhalt der Grafik wird durch geometrische Formen beschrieben. Kann gerastert und beliebig skaliert werden. Beispiel: SVG, PS (Postscript), CGM, IGES, DWF/DXF

1.1 Menschliche Wahrnehmung von Licht

zwischen 380nm (violet/blau) und 780nm (rot)
Zapfen/Cones Farbliche Wahrnehmung (ca. 6 Millionen) je für einen Farbkanal zuständig (64 % rot, 32 % grün, 4 % blau)
Stäbchen/Rods Helligkeitswahrnehmung (ca. 120 Millionen)
Farbsysteme Repräsentation durch unterschiedliche Modelle, wie:

- biologisch orientiert: CIE XYZ
- Hardware-orientiert: RGB, CMY, CMYK (mit Schwarz, um Tinte zu sparen)
- Anwender-orientiert: HSV, HSB

Steven's Power Law physikalische Intensität (Helligkeit) ist nicht proportional zur wahrnehmbaren Helligkeit.

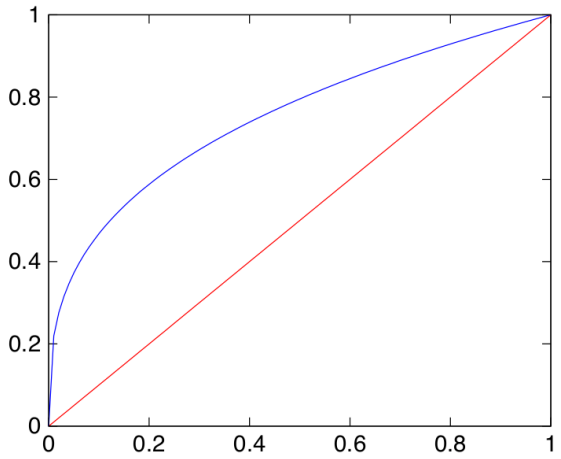


Figure: Example plot of $y = x$ (red) vs $y = x^{\frac{1}{3}}$ (blue)

Perceived Intensity by Eye

$$0.3 \leq a \leq 0.5$$

e.g.:

$$a = \frac{1}{3} \Rightarrow \psi(I) = kI^{\frac{1}{3}}$$

Observation

- Sensitivity is intensity-dependent
- **High** in dark areas
- **Lower** in bright areas

Question

- What consequences does this have on the digital representation of values?

Gamma Korrektur korrigiert physikalische Intensität, um kontinuierlichen wahrnehmbaren Intensitätszuwachs zu bekommen.

Before

After

Pros

- Finer intensity resolution in dark areas.
- Reduced perceived discontinuities.

Cons

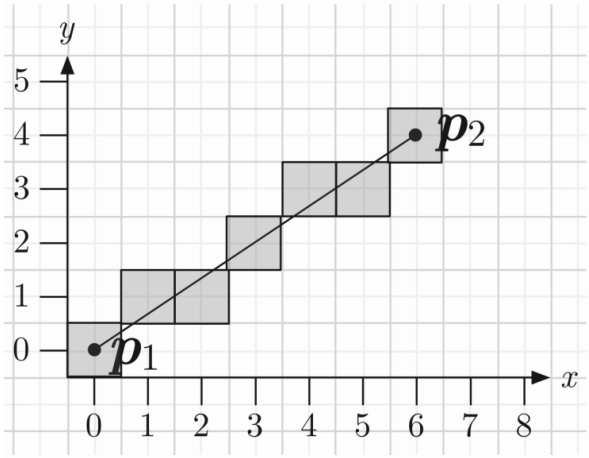
- Image appears overall to bright.
- Specifically relevant in dark areas.

$$n = \lfloor I^{\frac{1}{\gamma}} 2^N \rfloor$$

mit $I \in [0,1]$, $n \in [0, 2^N]$: Abbildung der physikalischen Intensität auf wahrnehmungskorrigierte mit N Bit Genauigkeit.

1.2 Rasterisierung von Vektorgrafiken

Digital Differential Analyzer (DDA) Rastern von Linien zwischen zwei beliebigen Punkten p_1 und p_2. Linie kann als Funktion $y = mx + b$ repräsentiert werden.



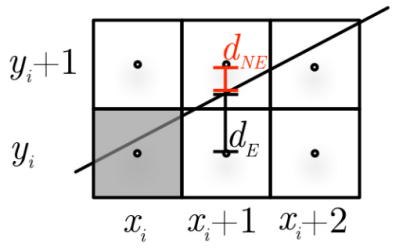
Follows

```
p1x = round(p1x); p1y =
  round(p1y);
p2x = round(p2x); p2y =
  round(p2y);
double m = (p2y-p1y)/(p2x-p1x);

pixelSet(p1x, p1y);
pixelSet(p2x, p2y);

for (int x = p1x+1; x < p2x; x++)
  {
    pixelSet(x, round(m * x + b));
}
```

Bresenham's Algorithmus Verfeinerung von DDA



Assume

- Pixel (x_i, y_i) is set.

Follows choice for next pixel as

- $(x_i + 1, y_i)$ or $(x_i + 1, y_i + 1)$.

Observation

$$y_{i+1} = \begin{cases} y_i & \text{if } d_E < d_{NE} \\ y_i + 1 & \text{otherwise.} \end{cases}$$

Given coords (x', y') follows

1. $d_E = y' - y_i$
2. $d_{NE} = y_i + 1 - y'$
3. $\Rightarrow \epsilon = d_E - d_{NE} = 2y' - 2y_i - 1$

Sign of decision variable ϵ determines pixel:

$$y_{i+1} = \begin{cases} y_i & \text{if } \epsilon \leq 0 \\ y_i + 1 & \text{otherwise.} \end{cases}$$

Aliasing Da Pixel entweder an oder aus sind (haben Farbe oder nicht), bildet sich eine Treppe beim Rastern von Linien. Kann beim Samplen auftreten
→ **Nyquist-Shannon Sampling Theorem** Sample Frequenz >= Doppelte Signal Frequenz
⇒ **Antialiasing** hat keine harten Übergänge sondern bildet "Farbverlaufäm Kantenrand (Pixel sind an, aus oder abgeschwächt farbig) durch Supersampling (mehr Punkte als Raster berechnen) und Durchschnittsbildung
Supersampling feinere Auflösung als Zielbild wählen und regelmäßig samplen (beste Ergebnisse), zufällige Punkte wählen im gesamten Bild, zufällige Punkte in definierten Räumen

1.3 Vektoren

Skalarprodukt/Dot-Produkt $u \cdot v = (u_0, u_1, u_2)^T \cdot (v_0, v_1, v_2)^T = u_0v_0 + u_1v_1 + u_2v_2$
Vektorlänge $||v|| = \sqrt{v \cdot v}$
Kreuxprodukt $w = u \times v = \begin{bmatrix} u_1v_2 - u_2v_1 \\ u_2v_0 - u_0v_2 \\ u_0v_1 - u_1v_0 \end{bmatrix}$
w ist orthogonal zu u und v. Sie bilden ein rechthand-(Koordinaten)-System

1.4 Homogene Koordinaten

Repräsentieren von Vektoren als homogen, um Transformationen durchführen zu können. (wie in Robotik 1). Vierte Komponente ist 0 für Vektoren, 1 für Punkte.

Point-point difference

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ 1 \end{pmatrix} - \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ 1 \end{pmatrix} = \begin{pmatrix} r_1 - s_1 \\ r_2 - s_2 \\ r_3 - s_3 \\ 0 \end{pmatrix}$$

Vector-vector sum

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ 0 \end{pmatrix} + \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ 0 \end{pmatrix} = \begin{pmatrix} r_1 + s_1 \\ r_2 + s_2 \\ r_3 + s_3 \\ 0 \end{pmatrix}$$

Point-vector sum

$$\begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ 1 \end{pmatrix} + \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ 0 \end{pmatrix} = \begin{pmatrix} r_1 + s_1 \\ r_2 + s_2 \\ r_3 + s_3 \\ 1 \end{pmatrix}$$

Scalar-vector product

$$r_1 \begin{pmatrix} s_1 \\ s_2 \\ s_3 \\ 0 \end{pmatrix} = \begin{pmatrix} r_1 s_1 \\ r_1 s_2 \\ r_1 s_3 \\ 0 \end{pmatrix}$$

Fundamental rotation matrices $R_n(\phi)$ describes the rotation around the n axes with the angle ϕ.

$$R_1(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{1}$$

$$R_2(\phi) = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \tag{2}$$

$$R_3(\phi) = \begin{bmatrix} \cos\phi & -\sin\phi & 0 \\ \sin\phi & \cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Homogenous transformation Describe coordinates as homogenous coordinates to describe rotations and translations
vector q as homogenous coordinates is $[q_1 q_2 q_3 1]^T$
Homogenous transformation matrix

$$T = \begin{bmatrix} R & p \\ \eta^T & 1 \end{bmatrix} \tag{4}$$

with
$R \in \mathbb{R}^{3\times3}$ is a rotation matrix
$p \in \mathbb{R}^{3\times1}$ is a translation vector
$\eta^T \in \mathbb{R}^{1\times3}$ is a perspective vector, here zero vector

$$Rot(\phi, k) = \begin{bmatrix} & R_k(\phi) & & \begin{matrix} 0 \\ 0 \\ 0 \end{matrix} \\ 0 & 0 & 0 & 1 \end{bmatrix}, Tran(p) = \begin{bmatrix} 1 & 0 & 0 & p_1 \\ 0 & 1 & 0 & p_2 \\ 0 & 0 & 1 & p_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5}$$

Reflection Use identity matrix with -1 for the reflection axis. Reflection at a point is reflection at the intersection of three orthogonal planes (-1 for every axis).

Transformation for abitraty points Translate everything so that the transformation point is located at the origin. Do the Transformation. Translate everything back.

Inverse homogenous transformation If the transformation matrix T maps coordinates from coordinate frame A to B, the inverse T^{-1} maps coodinates from B to A.

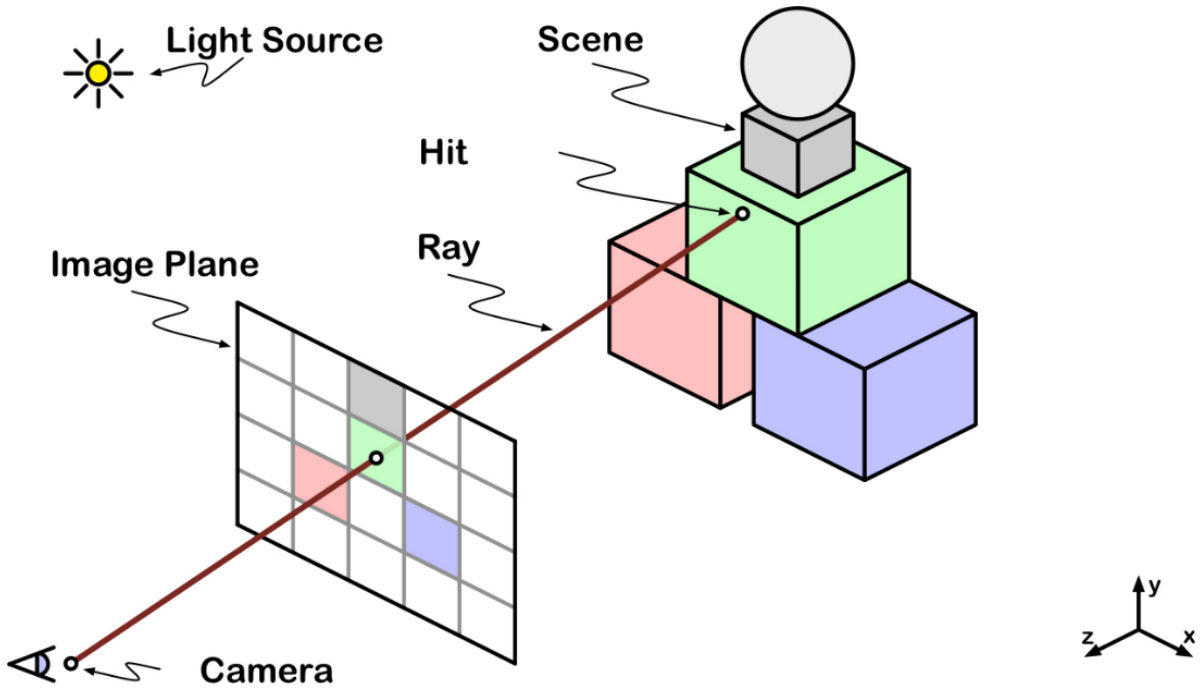
$$T^{-1} = \begin{bmatrix} & R^T & & -R^T p \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

with $\eta = 0$ and $\sigma = 1$

2 Raytracing

Rendern von 3D-Objekten als 2D-Repräsentation. Orientiert an Physik, dadurch sehr realistische Darstellungen möglich. Strahlen werden von Kamera in die Szene gesendet. Schneiden diese ein 3D-Objekt, wird der Strahl von dort zu einer Lichtquelle verfolgt und der aussendende Pixel entsprechend gefärbt.

```
raytrace(scene, camera, image):
   # For all pixels in image
   for (x, y) in image:
      # 1. Generate ray through pixel
      ray = camera.generateRay(x, y)
      # 2. Find closest intersection with scene
      hit = scene.intersect(ray)
      # 3. Calculate light intensity
      color = shade(hit, scene)
      # 4. Set pixel color
      image.set(x, y, color)
```



Photon Lichtstrahl mit bestimmter Energie bzw. Farbe

$$E = h \cdot f$$

mit E: Energie, h: Plancksche Konstante, f: Frequenz

$$1 Lumen = 4 \cdot 10^{15} Photonen/sec$$

Absorption Photon verschwindet, wird von Gegenstand geschluckt
Reflektion Photon prallt an Oberfläche ab
Refraktion Photon geht durch eine Oberfläche hindurch (z.B. Glas)
Ray/Strahl Dargestellt als Startpunkt mit Richtungsvektor: $x(t) = x_0 + t\vec{d}$ **Licht-Material Interaktion** Mögliche Modelle zur realistischen Farbermittlung:

- Quantum Theorie (Emission, Absorption)
- Spezielle Relativität (aberration, blueshift, redshift, time dilatation)
- Wellenoptik (diffraction, dispersion, Interferenz)
- Geometische Optik (Reflektion, Refraktion)

2.1 Shading

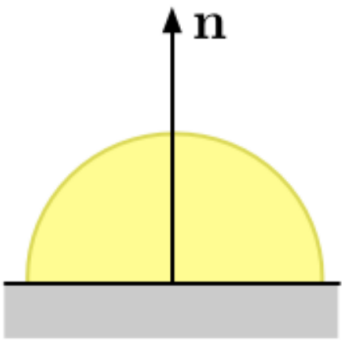
Verwende Objektfarbe bei Rayintersection. Naiver Ansatz, da physikalische Gesetze wie Reflektion oder Schatten und Objektstruktur ignoriert werden. Gleichmäßig gefärbte Objekte.

2.2 Phong Relektionsmodell

Modell zur Farbberechnung. Basiert auf geometischer Optik. Setzt sich zusammen aus der Farbe (Ambient) + Oberflächenbeschaffenheit und Schatten (Diffuse) + Reflektion (Specular). Verwendet folgende Vereinfachungen:

- Oberfläche: Isotropic, betrachtet nicht die Wellenlänge und Polarisierung
- Raum: nimmt Vacuum an, keine atmospherischen Effekte
- Licht: Lichtquellen als einzelne Punkte

Ambient



Note

- No explicit light sources
- Indirect, constant illumination for the entire scene

Ambient reflection term

$$L_r = k_a L^a$$

With

L_r: Reflected (ambient) energy

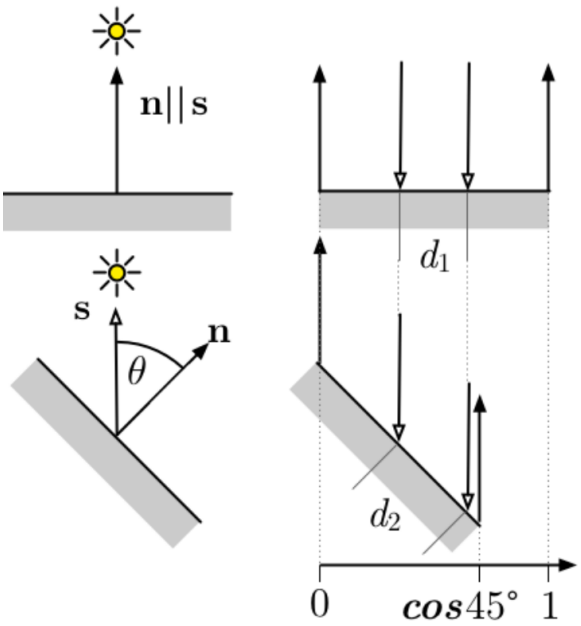
k_a: Material's ambient reflectivity coefficient

L^a: (fictitious) ambient light energy

Rational

- Model for 'background lighting'
- Simulates a global contribution

Diffuse Betrachtet reflektiertes Licht von Oberflächen (unterschiedlich je nach Oberfläche matt/glänzend)



Theorem

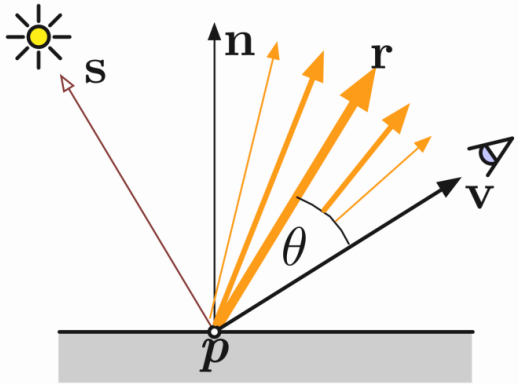
The radiant intensity observed from a Lambertian surface is directly proportional to the cosine between the the direction of the incident light and the surface normal. (J. H. Lambert, 1760).

Calculate Lighting Term

$$I_d = I_s k_d \max\left(\frac{s \cdot n}{|s|\,|n|}, 0\right)$$

$$L_d(p) = k_d \sum_j L_j \max(0, n \cdot s_j)$$

mit L_d: Reflektiertes Licht (Diffuse-Anteil)
k_d: Material-Koeffizient für Diffuse Reflektion
L_j: Licht (Farbe) der Lichtquelle j
n: Normale vom Punkt p zur Oberfläche
s_j: Vektor von p zur Lichtquelle j
Specular Betrachtet starke Reflektionen für glänzende Oberflächen (weiße/helle Punkte)



- Reflection of glossy surfaces.
- Emitted radiance is proportional to angle θ between r and v.
- Proportionality is cos(θ) or r · v.

Specular reflection term

$$L_s(p, v) = \int_w k_s L_j\, n \cdot w_j\, dw_j$$

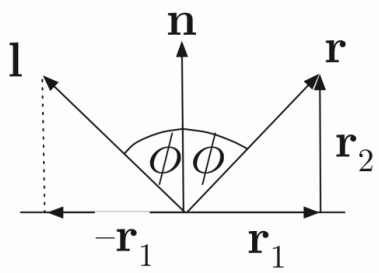
$$\approx k_s \sum_j L_j \max(0, r_j \cdot v)^{k_e}$$

$L_j = L_i(p, w_i)$: Energy of isotropic point light sources j with constant radiance
$L_s(p, v)$: Reflected (specular) energy.
k_s: Material's specular reflectivity coefficient.
k_e: Material's specular shininess.

Note

- k_e models sharpness of highlights.

Reflektionsberechnung Berechnung des Reflektionsvektors



Note

Calculation of reflectance vector often necessary for lighting models etc.

Example

- Phong model's specular term.

Question

How to calculate the reflectance vector given light direction and normal?

Derivation

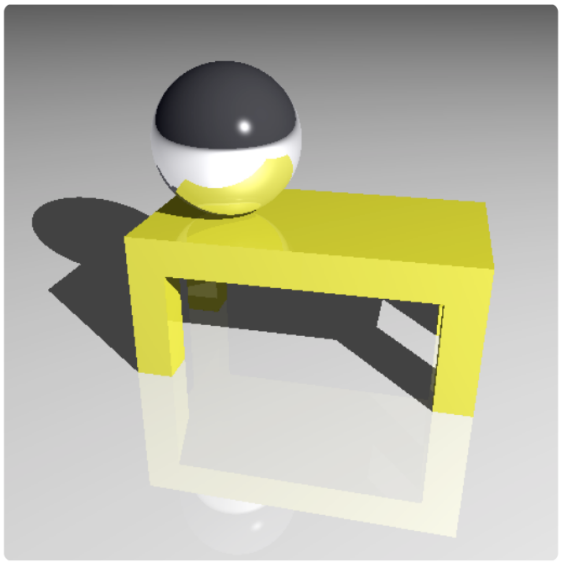
1. $r = r_1 + r_2$
2. $r_1 = -l + r_2$
3. $\Rightarrow r = 2r_2 - l$
4. $r_2 = (n \cdot l)n$

Solution Reflection Vector

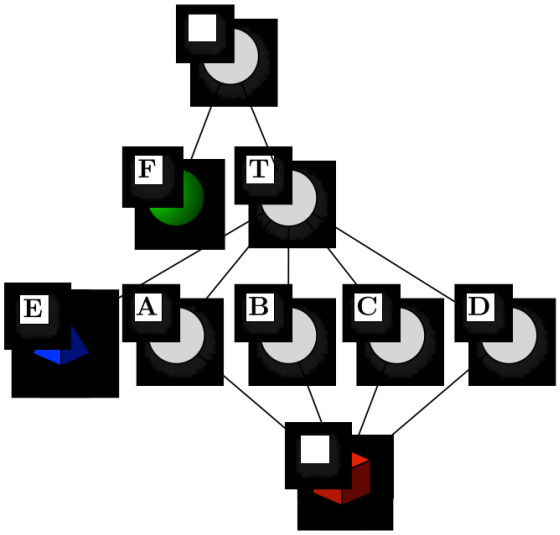
5. $\Rightarrow r = 2(n \cdot l)n - l$

3 Scene Graph

Ein Szenengraph repräsentiert alle Objekte einer Szene als Graph. Dabei werden Positionen immer relativ zum Elternelement angegeben. Wird eine globale Position relativ zum Weltkoordinatensystem benötigt, müssen alle Transformationen entlang des Graphs vom Ursprung zum Objekt angewendet werden.



Ball on top of the table



Scene graph of the still

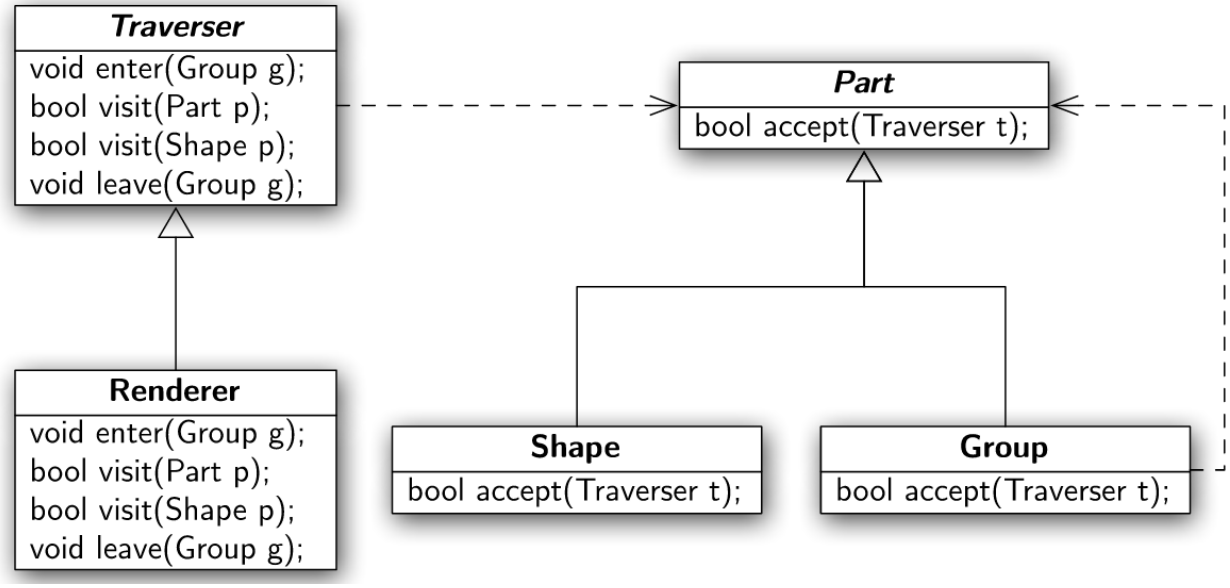
3.1 Koordinatensysteme

Man unterscheidet mehrere Koordinatensysteme
Weltkoordinatensystem W Globales Referenzsystem für alle Objekte

Objektsystem O Relatives System für jedes Objekt. Ursprung im Objektursprung. Verwendet z.B. um Punkte eines Objekts relativ zum Ursprung anzugeben.
Kamerasystem C System relativ zur Kameraposition

3.2 Rendering

Zum Rendern des Scene-Graphs wird das Visitor-Pattern verwendet.

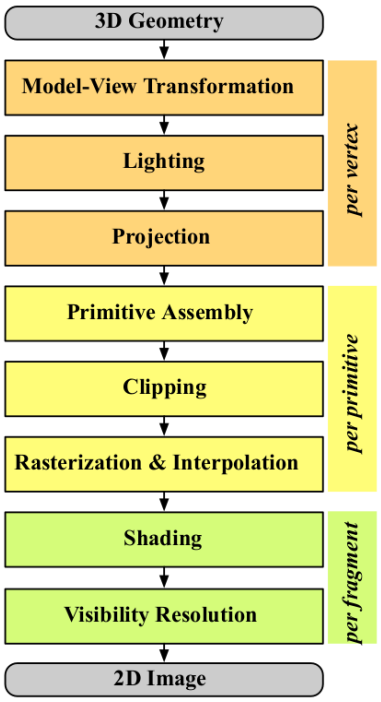


4 Rasterisierung

Abbildung von 3D-Objekten auf 2D-Projektionsfläche → Forward Rendering
Unterschied zu Raytracing Szene wird auf Betrachter projektiert. Bei Raytracing wird vom Betrachter aus die Szene mit Rays abgetastet, dadurch flexibler. Rasterisierung ist an geometische Formen und deren 2D-Abbildung gebunden, dadurch aber schneller.

4.1 Pipeline

Ablauf der Rasterisierung lässt sich gut durch Pipeline darstellen, deren Schritte durchlaufen werden:

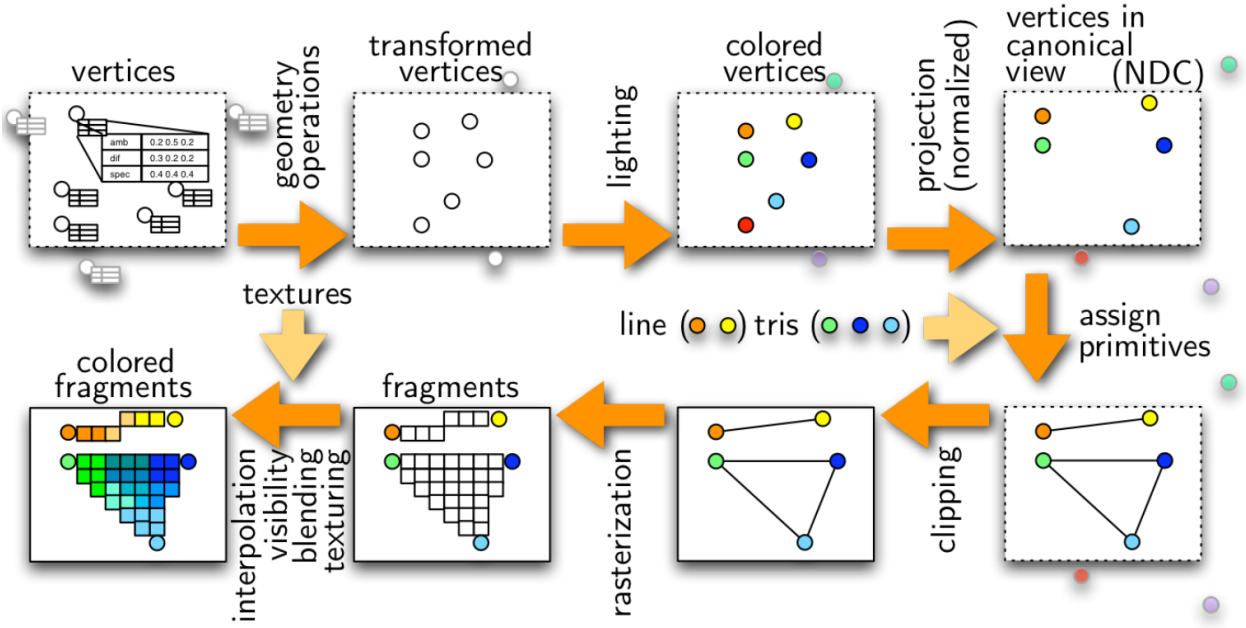


Input

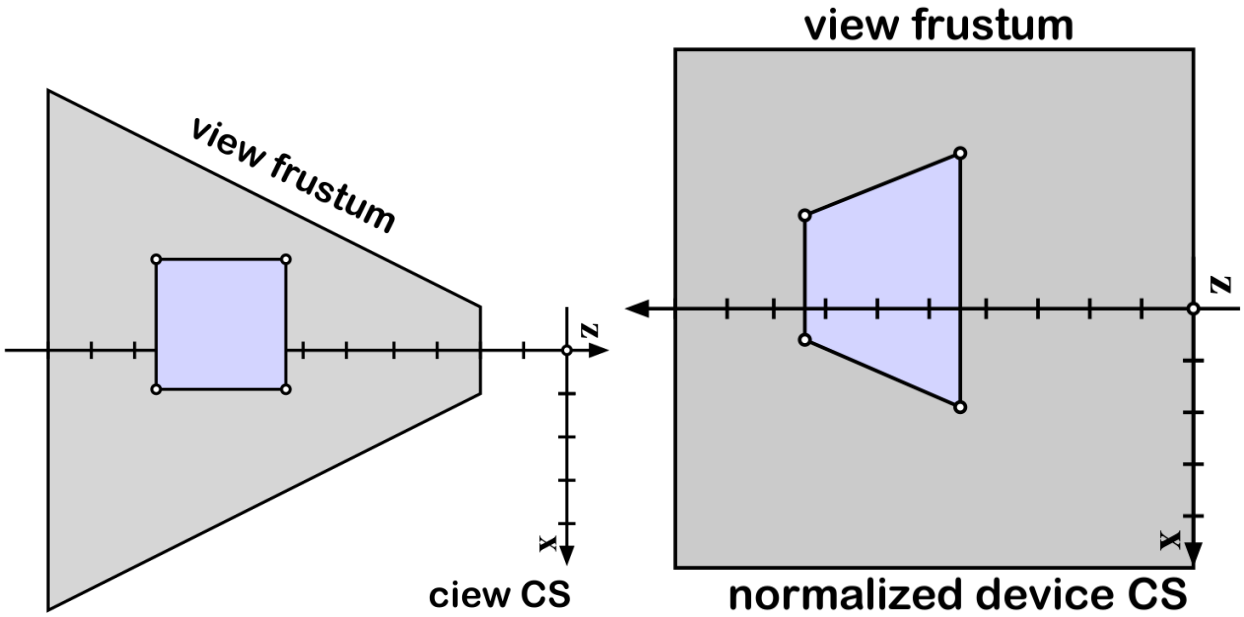
1. **3D Geometry:** points (1 vertex), lines (2 vertices), triangles (3 vertices), polygons (>3 vertices)
2. **Vertex Attributes:** position, color, normals, texture coordinates, ...
3. **State:** Material properties, light source parameters, texture images, ...
4. **Algorithms:** vertex shader, tessellation shader, geometry shader, fragment shader

Output

- Pixel color values in frame buffer.



Model-View Transformation Transformiert die 3D-Objekte von den Objektsystemen in das Kamerasystem (analog Scene Graph)
Lightning Bestimmt die Farben der 3D-Objekte (z.B. mit Phong)
Projection Bestimmt den Bereich, der durch die Projektionsfläche abgebildet werden kann. Dabei wird die Projektionsmatrix auf die Geometriedaten angewendet.



Primitive Assembly Gruppiert Vertices zu Primitives wie Punkt, Linie oder Dreieck. Benötigt dafür die Information, welche Vertices zusammen gehören.
Clipping Schneidet vorhandene Primitives zu anhand des vorher bestimmten sichtbaren Projektionsbereichs. Dabei können Primitives komplett verschwinden oder verändert werden, wenn Teile davon außerhalb der Projektionsfläche liegen.
Rasterization & Interpolation Generiere Attribute (wie Farbe) für Werte (Pixel), die innerhalb von Primitives liegen (z.B. das Innere vom Dreieck). Dies geschieht durch Interpolation der Eckpunkte der Primitives. Dabei entsteht für jedes Primitive ein Stream an Fragments (Position und Farbe), der diese Informationen speichert.
Fragmente gehören immer zu einem Pixel. Es kann allerdings mehrere Fragmente pro Pixel geben, wenn z.B. Multisampling durchgeführt wird. Allerdings gibt es mindestens ein Fragment pro Pixel.
Shading Mappt die einzelnen Fragments auf die Farben der Pixel. Es entsteht ein Stream mit $0..n$ Farbwerten.
Visibility Resolution Prüft die Tiefenwerte der Fragmente und füllt den z-Buffer. Dieser gibt an, wie weit ein Pixel/Fragment von der Projektionsfläche nach hinten entfernt liegt.

4.2 Zusammenfassung

Da die einzelnen Schritte der Pipeline klare Schnittstellen haben, können diese als Black-Boxes angesehen werden und individuell implementiert werden.
Vertex Shader Die Verarbeitung von Vertices wird Vertex Shader genannt.
Fragment Shader Die Verarbeitung von Framgents wird Fragment Shader genannt.