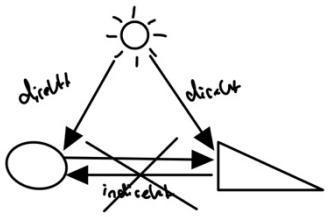
6 Kapitel 6: Beleuchtung und Schattierung

6.1 Beleuchtung vs. Schattierung

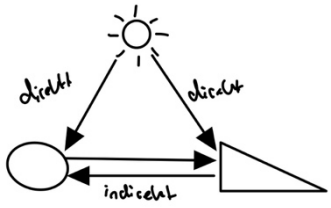
Definition 6.1 (Lokales Beleuchtungsmodell)

- Man berechnet die Intensität eines Punktes/Pixels in Abhängigkeit von direktem Lichteinfall einer oder mehrerer Lichtquellen.
- Indirektes Licht wird nicht reflektiert, es gibt nur direkte Beleuchtung.



Definition 6.2 (Globales Beleuchtungsmodell)

- Man berechnet die Intensität eines Punktes/Pixels in Abhängigkeit von direktem Lichteinfall einer oder mehrerer Lichtquellen
- Licht, das über die Reflexion an anderen Objekten in der Szene eintrifft wird beachtet.



Schattierungsmodell

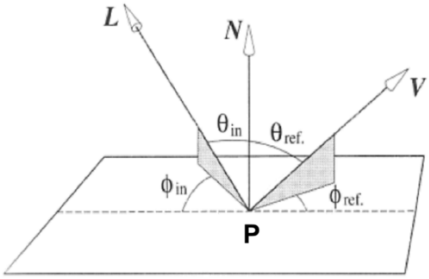
Ein Schattierungsmodell bestimmt, wann und wo ein Beleuchtungsmodell angewendet wird.

a. Auswertung eines Beleuchtungsmodells **für die Vertices**, Farben der Zwischenwerte für die (feiner aufgelösten) Pixel werden per Interpolation bestimmt
b. Auswertung eines Beleuchtungsmodells **für jedes Pixel** (aufwändiger, da es wesentlich mehr Pixel als sichtbare Vertices gibt)

Lokale Beleuchtungsmodelle - Reflexionen

Es gibt 2 Einfallswinkel: 1. zur Normale und 2. die Verdrehung. Zu komplex, Verdrehung wird vernachlässigt, sodass nur eine Ebene Reflexion stattfindet mit Einfallswinkel zur Normalen

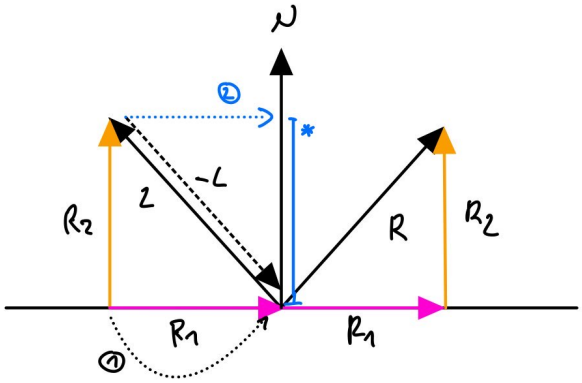
P Punkt auf Objektoberfläche,
N Flächennormalenvektor in P, normiert,
L Vektor von P zu einer Punktlichtquelle, normiert,
V Vektor von P zum Augpunkt (View), normiert,



ϕ_{in}, θ_{in} sphärische Koordinaten von L,
ϕ_{ref}, θ_{ref} sphärische Koordinaten von V.

Annahme:

- L und R liegen in einer Ebene und sind normiert → Einfallswinkel = Ausfallswinkel
- Die Größen in der Formel sind Vektoren
- L = Lichtvektor, R = Reflektionsvektor, N = Punktnormale



$R = R_1 + R_2$ (3)
$\Leftrightarrow (R_2 - L) + R_2$
$\Leftrightarrow 2 \cdot R_2 - L$
$\Leftrightarrow 2 \cdot (L \cdot N) \cdot N - L$
(N, L normiert)

(3) $R_1 = R_2 - L$

(2) $R_2 = (L \cdot N) \cdot N$
Skalarprodukt
* = Faktor

6.2 lokale Beleuchtung: Phong-Beleuchtungsmodell

Lokale Beleuchtungsmodelle - Phong Beleuchtungsmodell

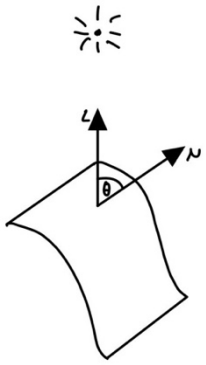
Das Modell simuliert stark vereinfacht folgende physikalische Reflexionsphänomene

- Diffuses Licht, spiegelndes Licht und ambientes Licht

Vereinfachte Annahmen:

- Reflexionen werden nur lokal (also unter direktem Lichteinfall) betrachtet
- Lichtquellen sind punktförmig
- die Geometrie der Oberflächen, außer den Oberflächennormalen, wird ignoriert

Perfekte diffuse Reflexion

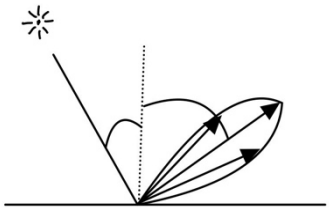


- $I_d =$ Intensität des reflektierten Lichts (skalarer Wert)
- $I_i =$ Intensität des einfallenden Lichts
- L = Lichtvektor, N = Punktnormale $\theta = \ Winkel\ zwischen\ N\ und\ L$
- Formel: $I_d = I_i * \cos\theta = \ I_i * (L * N)$

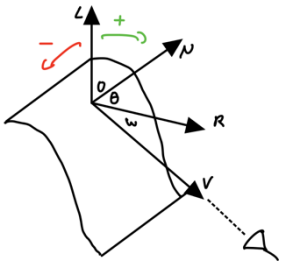
Welche mathematische Eigenschaft müssen die Vektoren in der Formel besitzen?

- die Vektoren müssen normiert sein

Unvollkommene spiegelnde Reflexion



- Der Lichtstrahl wird bei der Reflexion „aufgespalten“, es entsteht eine **Reflexionskonus** um die ausgezeichnete Reflexionsrichtung. Das ist **abhängig** von der Betrachtungsrichtung.
- Die Menge der reflektierten Helligkeit hängt gleichzeitig vom Einfallswinkel des Lichtes und vom Betrachtungswinkel ab.
- **Oberfläche**: glänzende aber doch leicht raue Oberfläche

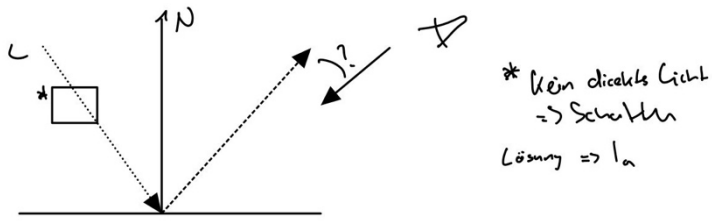


- $I_s = I_i * \cos^S \omega = \ I_i * (R * V)^S$
- $I_s =$ Die zu berechnende Intensität des reflektierten Lichts (skalarer Wer)
- $I_i =$ Intensität des einfallenden Lichts
- $\omega = \ Winkel\ zwischen\ N\ und\ L$
- R = Reflektionsrichtung, V = Betrachtungsrichtung, L = Lichtvektor, N = Punktnormale
- S = Perfektionsgrad, S -> inf (in Richtung R)
- die Vektoren müssen normiert sein

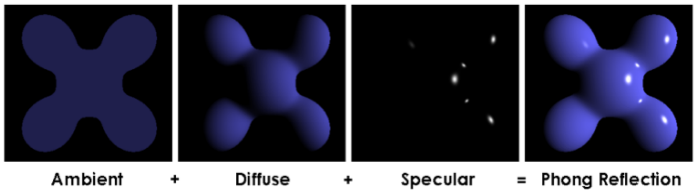
Gesamtmodell - Linearkombination des reflektierten Lichts

$$I = \ k_d I_d + \ k_s I_s + k_a I_a = \ I_i(k_d(L * N) + k_s(R * V)^n) + k_a I_a \qquad (k_d + k_s + k_a = 1)$$

$$- I_a = \ Ambientes\ Licht, konstante\ Grundhelligkeit$$



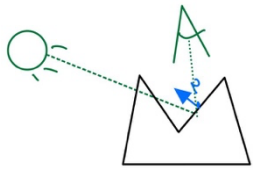
* kein direktes Licht
=> Schatten
Lösung => I_a



Problematik Schatten und Lichtabgewandte Oberflächenpunkte:

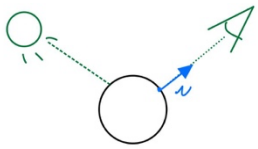
Schatten:

- Punkt liegt im Schatten, aber wird trotzdem beleuchtet, da das Modell nur auf die Oberflächenpunkte achtet und den Weg des einfallenden Lichts nicht berücksichtigt



Lichtabgewandte Objekt-Teile:

- Punkt ist dem Licht abgewandt und die Intensität des einfallenden Lichts ist 0. Das würde bedeuten Punkt ist schwarz, deswegen gibt es das ambiente Licht, welches unabhängig vom einfallenden Licht ist.



Farbe und Materialien

- Der reflektierte Lichtanteil wird durch die Eigenschaften der Lichtquelle und des Materials bestimmt.

$$ambient = k_a \mathbf{I}_a = \mathbf{a}_{light} * \mathbf{a}_{mat} = \begin{bmatrix} R_{\mathbf{a}_{light}} \\ G_{\mathbf{a}_{light}} \\ B_{\mathbf{a}_{light}} \\ A_{\mathbf{a}_{light}} \end{bmatrix} * \begin{bmatrix} R_{\mathbf{a}_{mat}} \\ G_{\mathbf{a}_{mat}} \\ B_{\mathbf{a}_{mat}} \\ A_{\mathbf{a}_{mat}} \end{bmatrix} = \begin{bmatrix} R_{\mathbf{a}_{light}} \cdot R_{\mathbf{a}_{mat}} \\ G_{\mathbf{a}_{light}} \cdot G_{\mathbf{a}_{mat}} \\ B_{\mathbf{a}_{light}} \cdot B_{\mathbf{a}_{mat}} \\ A_{\mathbf{a}_{light}} \cdot A_{\mathbf{a}_{mat}} \end{bmatrix}$$

Nur die Oberflächen, die dem Licht zugewandt sind, können überhaupt direkt beleuchtet werden.

- Prüfe mit Skalarprodukt > 0, ob ein Oberflächenpunkt der Lichtquelle zugewandt ist, d.h. der Einfallswinkel des Lichtes im Intervall [−90◦,+90◦] liegt.

$$diffus = k_d \cdot (\mathbf{L} \cdot \mathbf{N}) \cdot \mathbf{I}_i = \max(\mathbf{L} \cdot \mathbf{N}, 0) \cdot \mathbf{d}_{light} * \mathbf{d}_{mat} = \max(\mathbf{L} \cdot \mathbf{N}, 0) \cdot \begin{bmatrix} R_{\mathbf{d}_{light}} \cdot R_{\mathbf{d}_{mat}} \\ G_{\mathbf{d}_{light}} \cdot G_{\mathbf{d}_{mat}} \\ B_{\mathbf{d}_{light}} \cdot B_{\mathbf{d}_{mat}} \\ A_{\mathbf{d}_{light}} \cdot A_{\mathbf{d}_{mat}} \end{bmatrix}$$

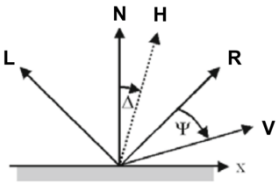
$$specular = k_s \cdot (\mathbf{R} \cdot \mathbf{V})^S \cdot \mathbf{I}_i = (\max(\mathbf{R} \cdot \mathbf{V}, 0))^S \cdot \mathbf{s}_{light} * \mathbf{s}_{mat} = (\max(\mathbf{R} \cdot \mathbf{V}, 0))^S \cdot \begin{bmatrix} R_{\mathbf{s}_{light}} \cdot R_{\mathbf{s}_{mat}} \\ G_{\mathbf{s}_{light}} \cdot G_{\mathbf{s}_{mat}} \\ B_{\mathbf{s}_{light}} \cdot B_{\mathbf{s}_{mat}} \\ A_{\mathbf{s}_{light}} \cdot A_{\mathbf{s}_{mat}} \end{bmatrix}$$

Problem: Die Berechnung des reflektierten Strahls (beim spiegeln) muss für jeden Oberflächenpunkt neu durchgeführt werden ⇒ **Performance-Verlust.**

Approximation mit Half-Way-Vektor

- Das Problem bei dem spiegelnden Anteil ist das wir für jeden Punkt den reflektierten Strahl neu ausrechen müssen →Performance Verlust.
  Mit dem Half-Way-Vektor ist dies billiger zu berechnen und man erkennt kaum einen Unterschied.
- Der Half-Way Vektor liegt zwischen L und V.
- Benutze statt R·V den Term H·N mit $H = (L + V)/||L + V||$

$$specular = (\max(\mathbf{H} \cdot \mathbf{N}, 0))^S \cdot \mathbf{s}_{light} * \mathbf{s}_{mat} = (\max(\mathbf{H} \cdot \mathbf{N}, 0))^S \cdot \begin{bmatrix} R_{\mathbf{s}_{light}} \cdot R_{\mathbf{s}_{mat}} \\ G_{\mathbf{s}_{light}} \cdot G_{\mathbf{s}_{mat}} \\ B_{\mathbf{s}_{light}} \cdot B_{\mathbf{s}_{mat}} \\ A_{\mathbf{s}_{light}} \cdot A_{\mathbf{s}_{mat}} \end{bmatrix}$$



Welcher Fehler kann bei der Benutzung des Half-Way Vektors passieren und wie wird er vermieden?

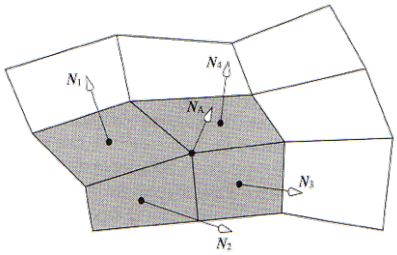
- Das das Skalarprodukt L * V zwischen der Lichtquelle L und der Betrachtungsrichtung V positiv ist, obwohl das Licht bereits die Rückseite der Oberfläche bestrahlt.
- Es kann ein Fehler für größere Winkel entstehen (H*N > 0), dies kann man umgehen, indem man zuerst den diffusen Anteil berechnet und ihn davon abhängig macht.
- Falls dieser 0 ist muss der spiegelnde auch 0 sein.

6.3 Schattierungsmodelle: Flat, Gouraud, Phong Shading

Beleuchtung beschreibt, wie man die Intensität bzw. Farbe berechnet und das Schattierungsverfahren beschreibt, wo und wie häufig es berechnet wird. Entweder pro Polygon flat-shading, pro Eckpunkt gourard-shading oder für jeden Pixel phong-shading

Flat Shading

Die Beleuchtung wird pro Polygon genau einmal in einem ausgewählten Oberflächenpunkt (face_normal) ausgewertet und alle Punkte des Polygons übernehmen die Farbe.



Vorteile:

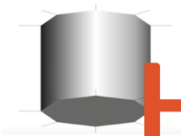
- Einfaches und effizientes Verfahren
- Interpolation findet nicht statt.
- Wird meist als Voransicht verwendet

Nachteile:

- Die meisten Kanten bleiben sichtbar
- Runde Objekte brauchen extrem viele Polygone

Gouraud-Shading

- Die Beleuchtung erfolgt an den Polygoneckpunkten und wird mit der vertice_normal ausgerechnet und dann interpoliert.

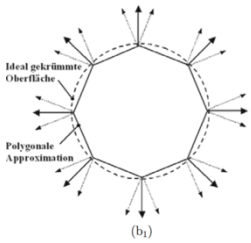


Vorteile:

- innere Kanten in Polygonnetzen werden geglättet
- Intensivitätsverlauf ist stetig

Nachteil:

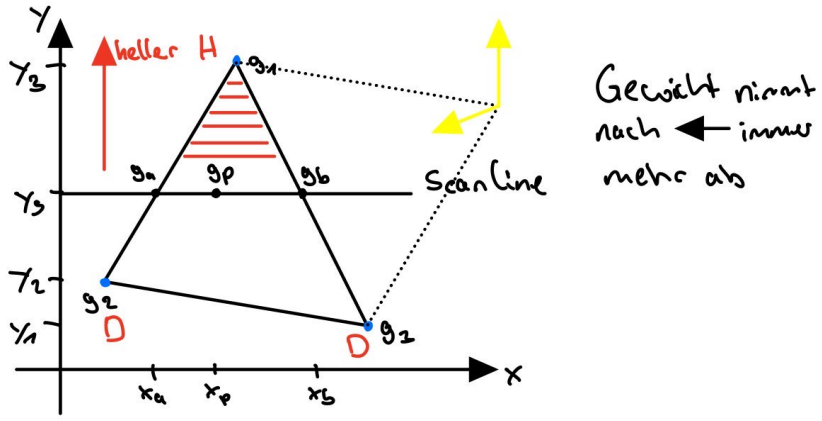
- Wenn das Licht nah an dem Polygon ist, dann flackert es
  => Highlights springt, deswegen nie bei bewegter Kamera
- Kleine Highlights innerhalb eines Polygons werden nicht berücksichtig, da Ecken diese nicht abbekommen
- Silhouette bleibt eckig – Geometrie wird nicht verändert



- Beim Highlight. Zum einem wegen der Interpolation werden die Highlights nach den Polygonen verschieden stark intensiv und nicht stetig. Zum anderen bei Bewegung, weil die Highlights dann springen und plötzlich verschwinden oder sichtbar werden
- An der Silhouette, weil die Geometrie nicht verändert wird

Bi-Lineare Interpolation

- Es wird ein Helligkeitsverlauf im Inneren des Polygons (Scan Line) erzeugt und auf dem Rand/Kanten wird auch einer ermittelt.
- Die Intensitätswerte werden entlang der Polygonkanten im Bildraum linear interpoliert, und danach zwischen den Kanten entlang der so genannten Scan-Lines



Gewicht nimmt nach ← immer mehr ab

glatte vs. scharfe Kanten

- Schattierungen wie Gouraud oder Phong wollen Kanten glätten. Wenn man bewusst scharfe Kanten haben möchte, dann muss man doppelte Kantenfestlegen, um verschiedene Normalen für einen Punkt zu haben.

Probleme bei je einem Netz für jede Teilfläche:

- Man verliert Konnektivitätsinformationen →Deformationen nicht möglich
- Doppelte Punkte mit derselben geometrischen Position
- Netzauflösung muss in beiden Meshes gleich sein!

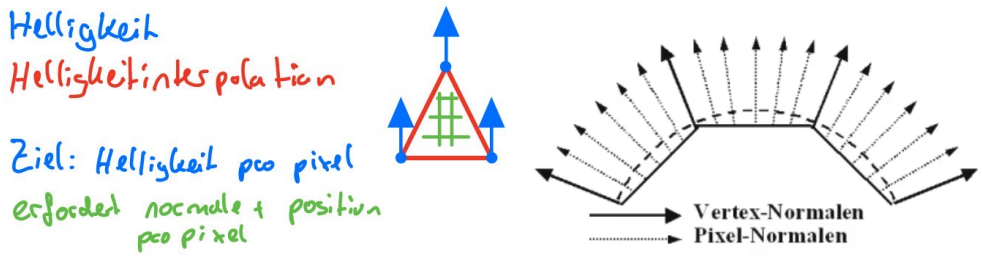
Lösung:

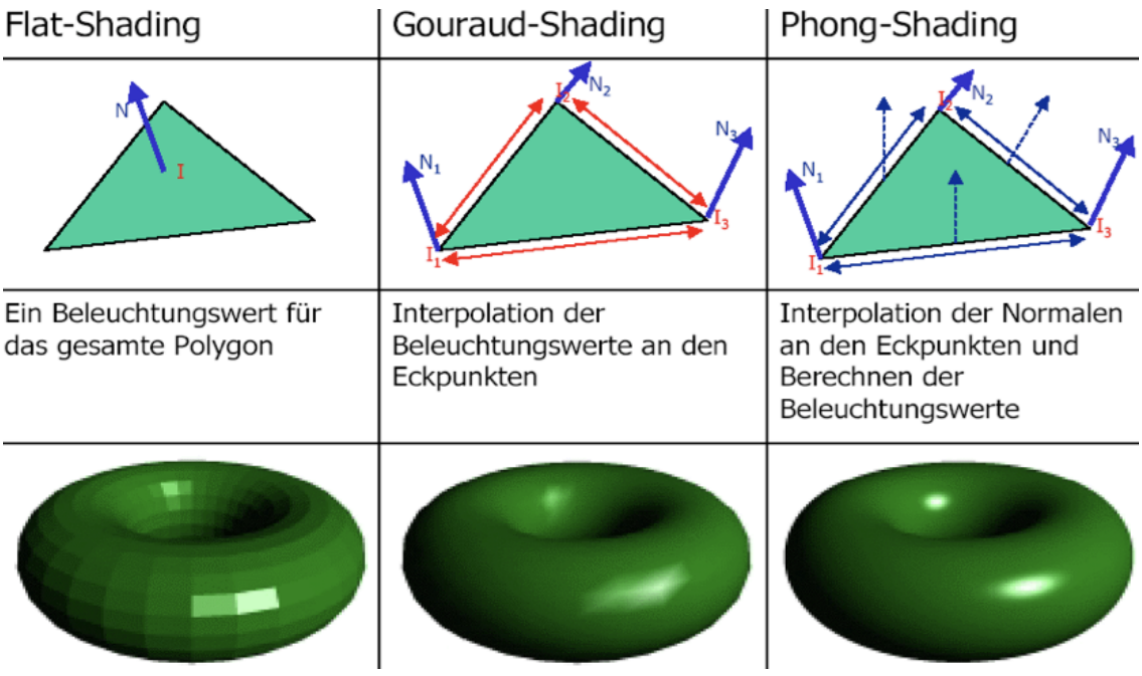
- Verwende ein Arbeitsmesh, i.d.R. mit optimiertem Zugriff auf die Nachbarschaftsinformation
    - darin Attribut pro Kante für scharf oder geglättet

Phong-Shading

- Die Auswertung des Beleuchtungsmodells erfolgt für jeden Pixelmittelpunkt der Polygonoberfläche.
- Die notwendigen Oberflächennormalen in den Polygonpunkten werden mittels Interpolation aus den Eckpunktnormalen ermittelt.
    - innere Kanten in Polygonnetzen werden geglättet
    - Highlight sind immer vorhanden und haben die erwartete Form.
    - **NT**: Da die Geometrie nicht verändert wird, bleibt auch hier die Silhouette „eckig“

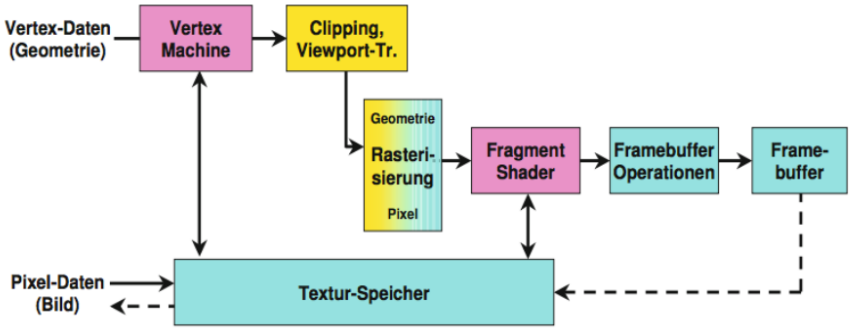
- Ziel: Helligkeit pro Pixel berechnen und erfordert Normale und Pos. von Pixel, werden mit Interpolation bestimmt





6.4 lokale Beleuchtung und Schattierung mit Shadern - Implementierungsaspekte

OpenGL-Pipeline



- Berechnungen in gelb: kontinuierlich, und in blau: pixelbasiert

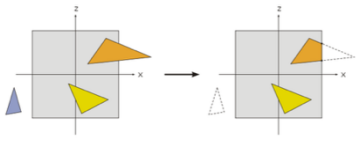
- Vertex-Machine: besteht aus mind. einem Vertex-Shader, ein Programm, dass durch alle Vertex einmal ausgeführt wird (parallel)
- Clipping, Viewport-Tr.: Objekte, die nicht gesehen werden, werden abgeschnitten und das Bild wird in die richtige Größe transformiert. Verdeckungsproblem wird gelöst
- Rasterisierung: projizierte Punkte (stetig) werden in Pixel (diskret) umgewandelt
- Fragment Shader: ähnlich wie Vertex-Shader, aber hier für alle Pixel
- Framebuffer Operationen: zum Beispiel Anti-Alising
- Framebuffer: enthält Pixel als Array

Vertex Machine:
- mindestens ein Vertex-Shader
- Parallel, performant pro Vertex berechnen
- mindestens: Berechnung der Welt-, Kamera- und Projektionstransformation

Vorteil: skaliert gut, d.h. schnellere GK sind um den Faktor der parallel vorhandenen Pipelines schneller
Nachteil: Geometrie-Konnektivität nicht verfügbar

Vertex Shader:
- Vertex Shader erhält Vertex / Normale in lokalen Koordinaten als Eingabe und alle benötigten Transformationsmatrizen
- Berechnet die Position des Vertex in NDC und ggf. dessen Helligkeit/Farbe
- gibt diese Ergebnisse weiter in der CG-Pipeline zur weiteren Verarbeitung.

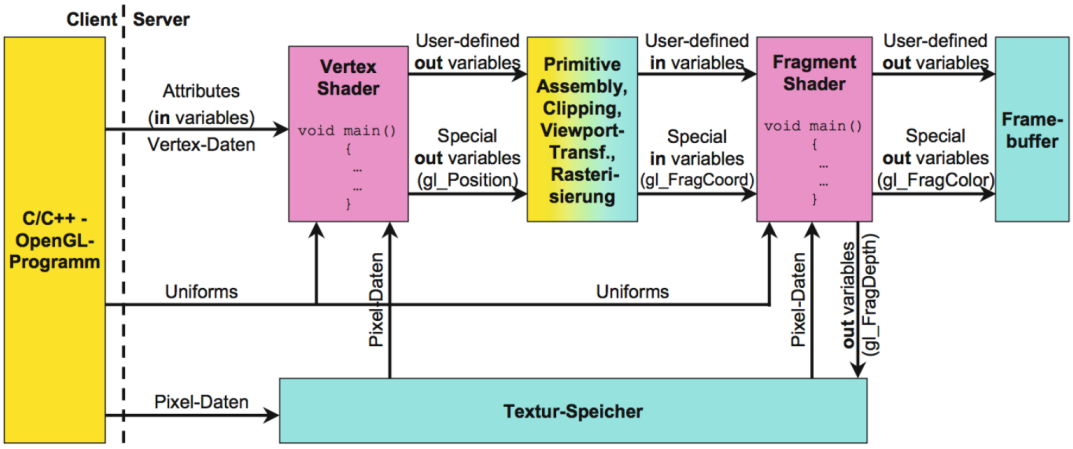
Clipping:
- Objekte, die nicht gesehen werden, werden dort abgeschnitten



Viewport-Berechnungen:
- transformieren das Bild auf die richtige Größe
- danach erzeugt die Rasterung die benötigte Anzahl Pixel
- z-Verdeckung wird gelöst und Inhalt des **Framebuffer** gezeichnet.

Fragment-Shader:
- Pixelgenauigkeit für ein Beleuchtungsmodell
- wieder potentiell alle Berechnungen parallel ohne Nachbarschafts-Info.
- Ein so genannter Fragment-Shader setzt dies analog zum Vertex-Shader um.

Datentransfer - Client-Server-Prinzip



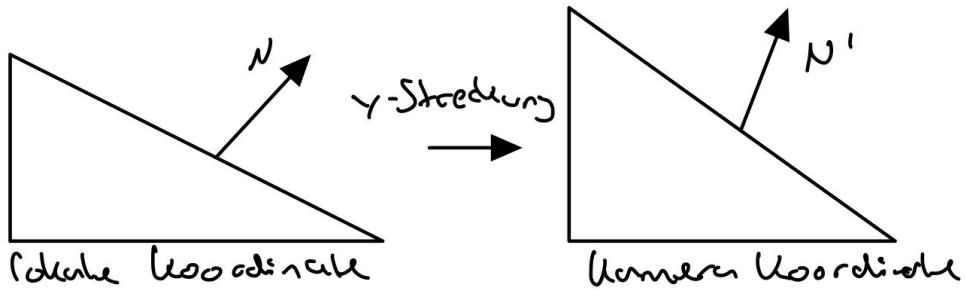
Gelb: Clientseitig Programm im Hauptspeicher und Code läuft auf CPU ab

Als Eingabe in einen Shader werden drei Daten-Arten unterschieden:

- **In-Variablen**, die pro Vertex oder Fragment variieren
- **Uniform-Variablen**, die pro Grafik-Primitiv variieren
- **Uniform sampler2D Variablen**, häufig Pixeldaten
- Zusätzlich gibt es **out-Variablen**, die können benutzerdefiniert sein oder in OpenGL fest eingebaut sein

Normal-Matrix

- Die wird benötigt da bei der Transformation die Normale mit Manipuliert wird.
- Da eine nicht gleichförmige Skalierung in x, y, und z die Normale manipuliert!



-

- Normale N und N' transformiert
- Tangente T und T' transformiert. T ist orthogonal zu N
- G ist die zu suchende Normal-Matrix
- M3 ist die model_view_matrix ohne homogene Koordinate

$$N' \cdot T' = (G \cdot N) \cdot (M_3 \cdot T) = 0$$

$$\Leftrightarrow (G \cdot N)^T \cdot (M_3 \cdot T) = N^T \cdot G^T \cdot M \cdot T = 0$$

$$G^T \cdot M = I \Leftrightarrow G = (M^{-1})^T = G$$

- Matrix multiplikation
- Skalar produkt

Da N und T nach der Transformation orthogonal sein müssen gilt: G = M

Erläutern Sie, warum die Beleuchtungsberechnung in Raster- Koordinaten mathematisch nicht korrekt ist

- Weil die Rasterkoordinaten nicht mehr stetig sind, sondern diskret. Deswegen sind die Teilverhältnissen falsch bei der Projektion von Pixeln in Raster