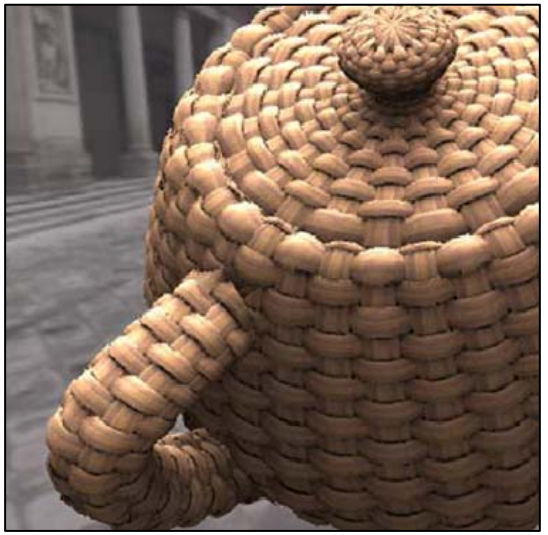
GPU Displacement Mapping
-
Generalized Displacement Mapping

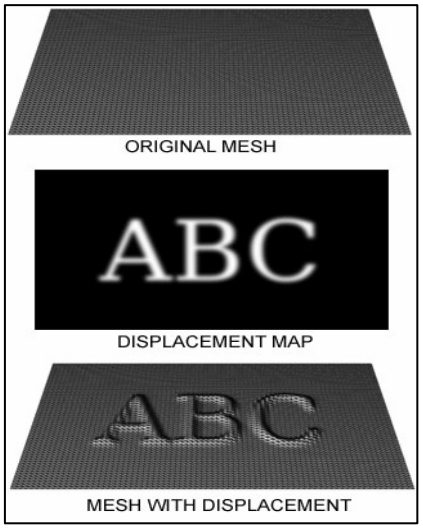
HS Vortrag
Computergrafik

Enrico Leonhardt

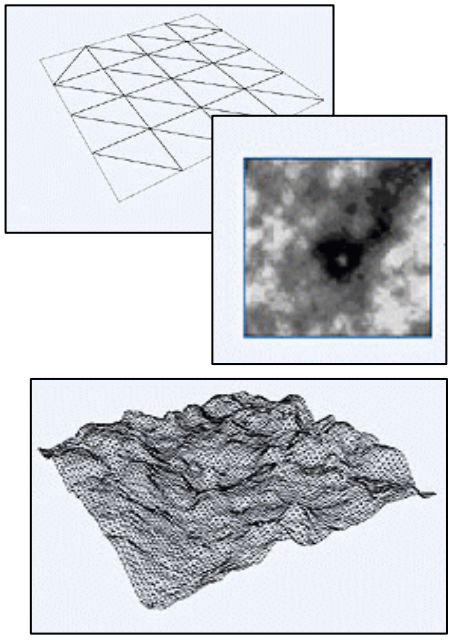
- Displacement Mapping
- GPU Rendering
- Displacement Mapping
  - Relief Mapping
  - VDM
  - Generalized Displacement Mapping
- Generalized Displacement Mapping
  - Modellierung
  - Rendering
  - Optimierung
  - Beispiele
- Zusammenfassung



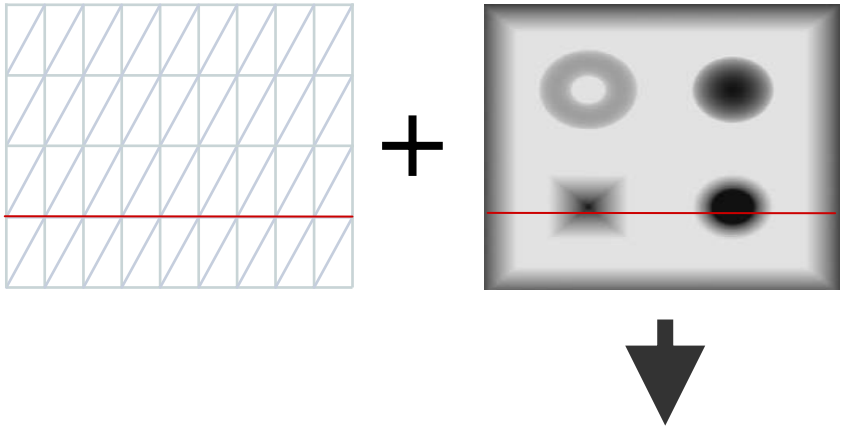
- Displacement Mapping
  - Pre-Sampled Displacement Mapping
    - Detailiertes Polygon mit Displacement Map belegen
    - Höhenposition der Vertices wird modifiziert
      → Oberflächenstruktur
        ➢ Adaptive Tesselation ist hierbei nicht möglich



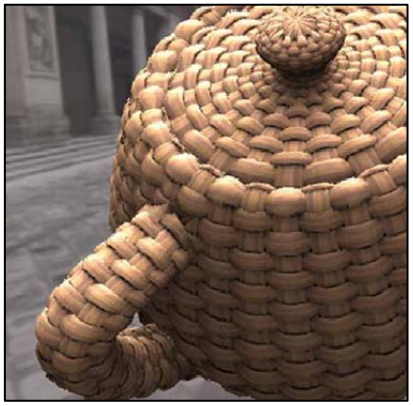
  - Sampled Displacement Mapping
    - einfaches Polygon mit Displacement Map belegen
    - Vertexanzahl erhöht sich → Oberflächenstruktur
      ➢ Adaptive Tesselation möglich
      ➢ für detaillierte Objekte oder Landschaften

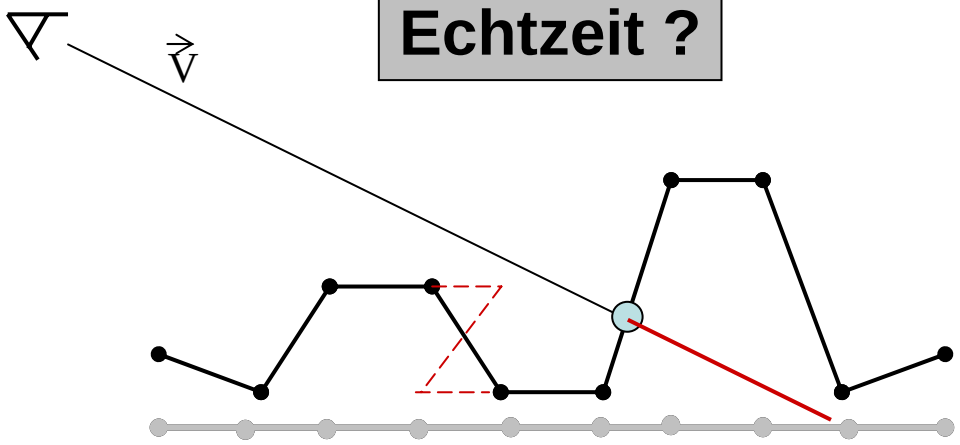


- Displacement Mapping
  - Zutaten
    - Mesh
    - Höhen - Map



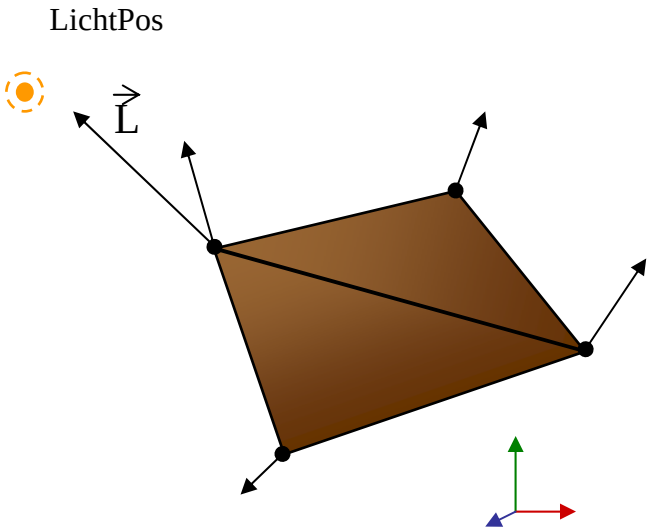
  - Resultat
    - detaillierte Oberfläche



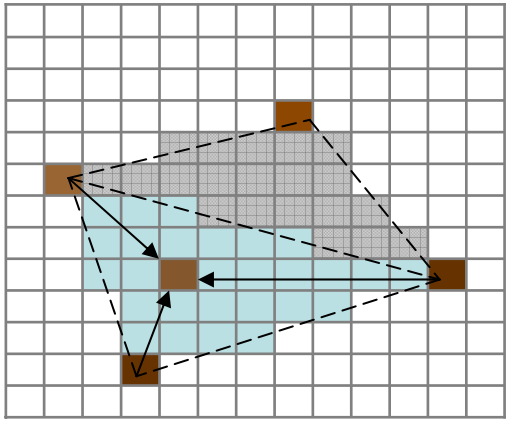


  - Einschränkung
    - Entlang der Oberflächennormale
    - Keine konkaven Strukturen möglich

- GPU Rendering
  - Vertex Programm
    - Position der Vertices auf Viewplane projizieren
    - Berechnung der Beleuchtung
    - Berechnung der Vertex-Farbe
      - ➢ Ausgabe: Vertex-Farbe
  - Rasterung
    - Interpolation der
      - » Fragment -Farbe aus Vertex-Farben
  - Fragment Programm
    - ➢ Ausgabe: Fragment -Farbwert



Szene in Welt - Koordinaten

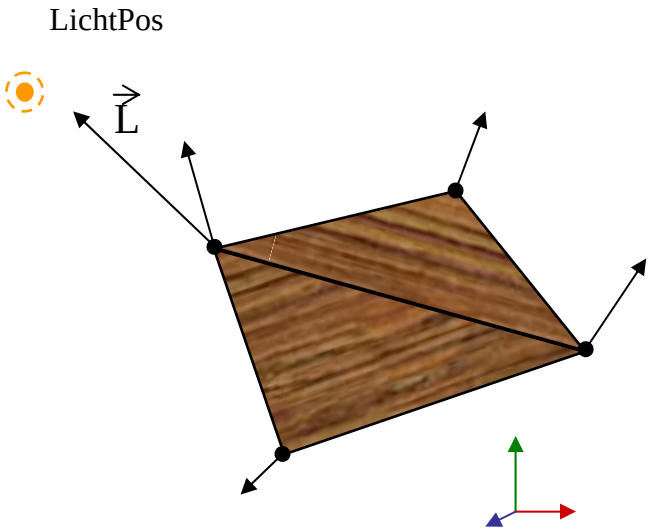


Szene in ModelView - Koordinaten

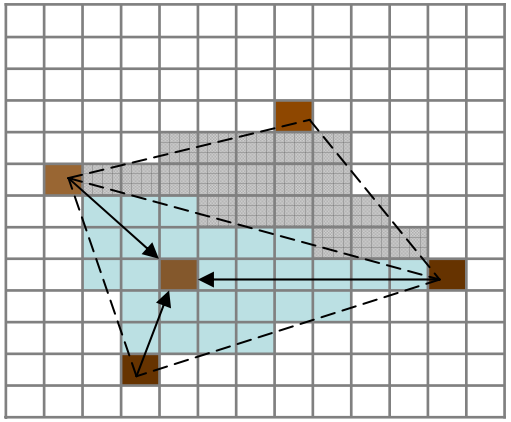
- GPU Rendering
  - Vertex Programm
    - Position der Vertices auf Viewplane projizieren
    - Berechnung der Beleuchtung

      ➢ Ausgabe: Beleuchtung
      UV-Koordinate
  - Rasterung
    - Interpolation der
      - » Beleuchtung
      - » UV-Koordinate
  - Fragment Programm
    - Farbwert aus Bilddatei
    - Berechnung der Fragment -Farbe

      ➢ Ausgabe: Fragment - Farbwert



Szene in Welt - Koordinaten

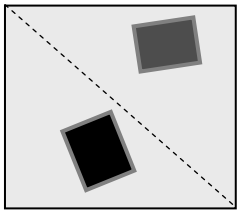


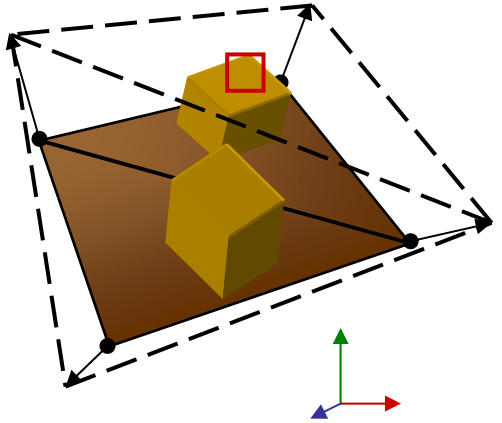
Szene in ModelView - Koordinaten

GPU Rendering – Displacement Mapping

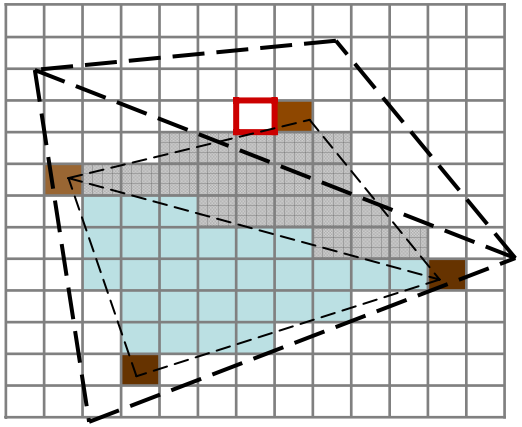
- Motivation:
  - Detaillierte Oberfläche mit Low Poly Meshes
  - Details kosten nur Rechenzeit, wenn wirklich sichtbar

- Methode:
  - Bounding-Mesh rendern
  - Per Pixel Raycasting im Fragment Programm





Szene in Welt - Koordinaten

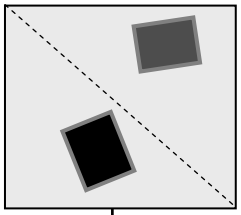


Szene in ModelView - Koordinaten

GPU Rendering – Displacement Mapping

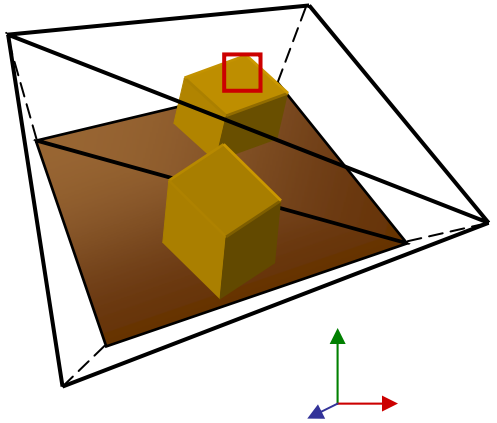
- Vertex Programm
  - Position der Vertices auf Viewplane projizieren
  - Berechnung der Blickrichtung

    ➢ Ausgabe: Blickrichtung
    UV-Koordinate

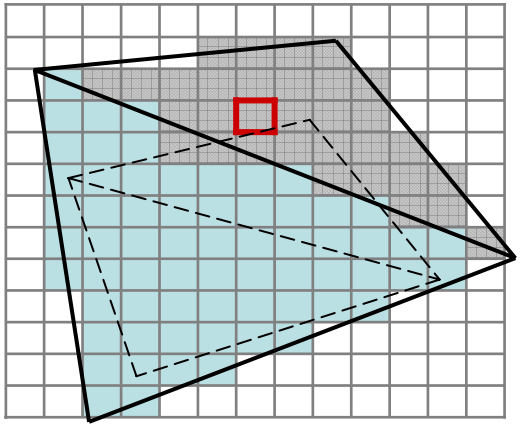


- Rasterung
  - Interpolation der
    - » Blickrichtung
    - » UV-Koordinate
- Fragment Programm
  - Raycasting im Texturraum
  - Berechnung der Fragmentfarbe

    ➢ Ausgabe: Fragment - Farbwert

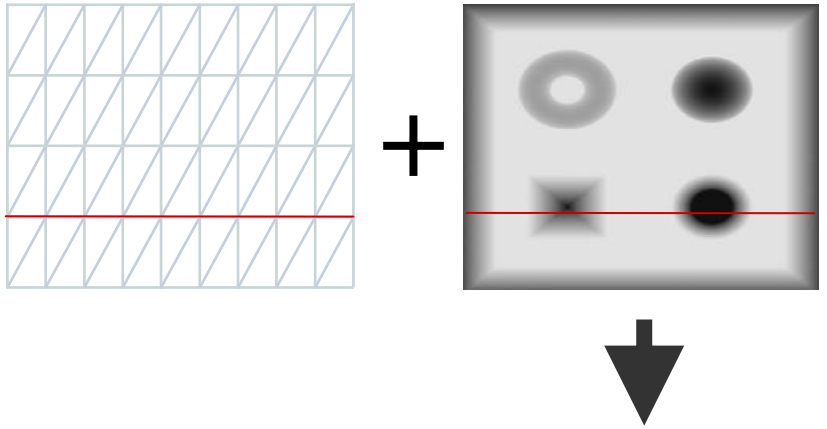


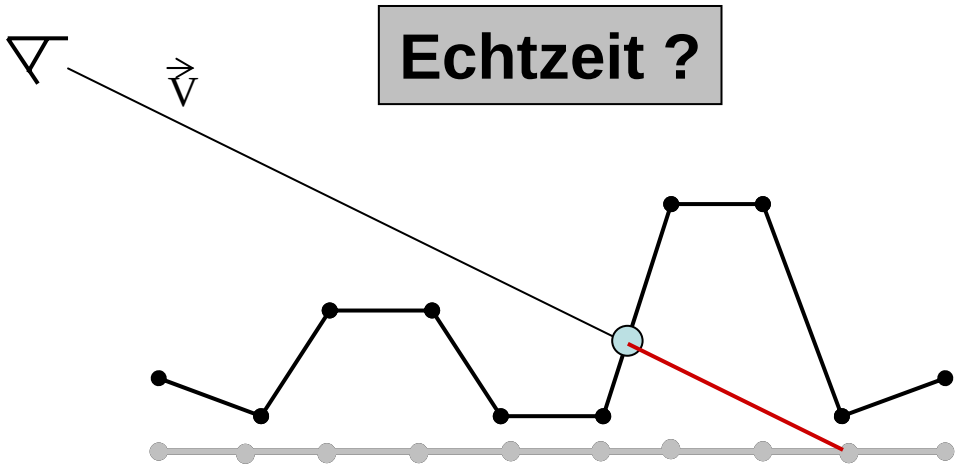
Szene in Welt - Koordinaten



Szene in ModelView - Koordinaten

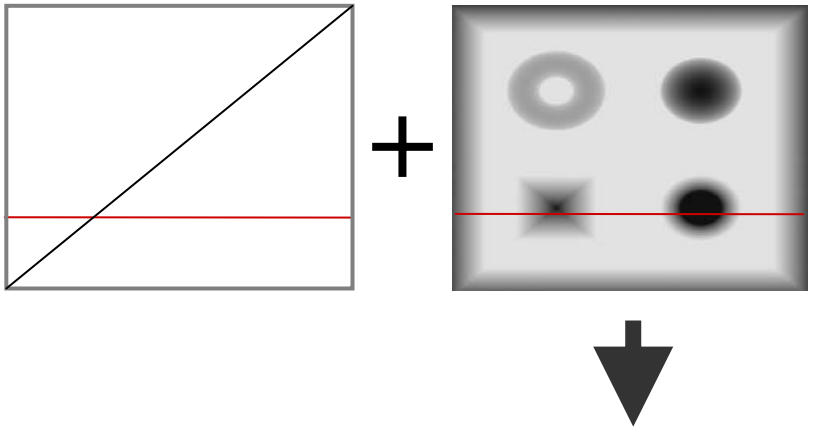
- Displacement Mapping
  - Zutaten
    - Mesh
    - Höhen - Map
  - Resultat
    - detaillierte Oberfläche



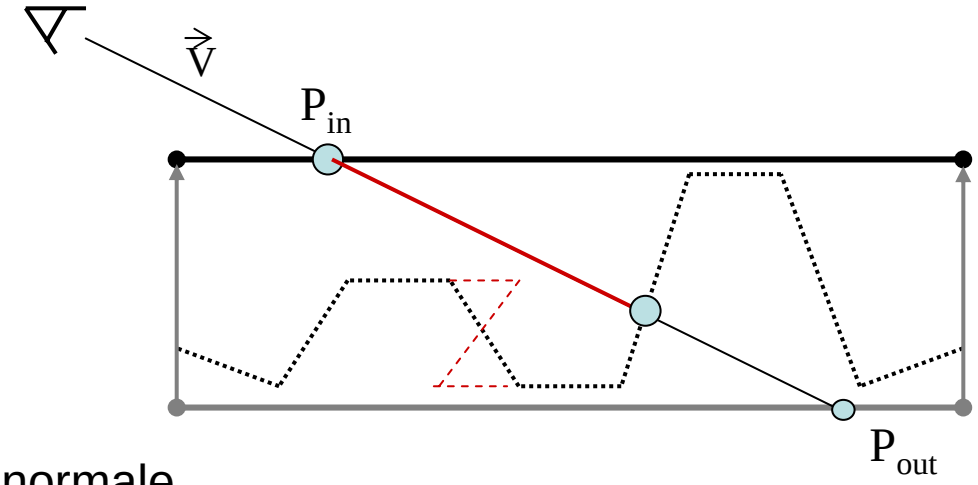


  - Einschränkung
    - Entlang der Oberflächennormale
    - Keine konkaven Strukturen möglich

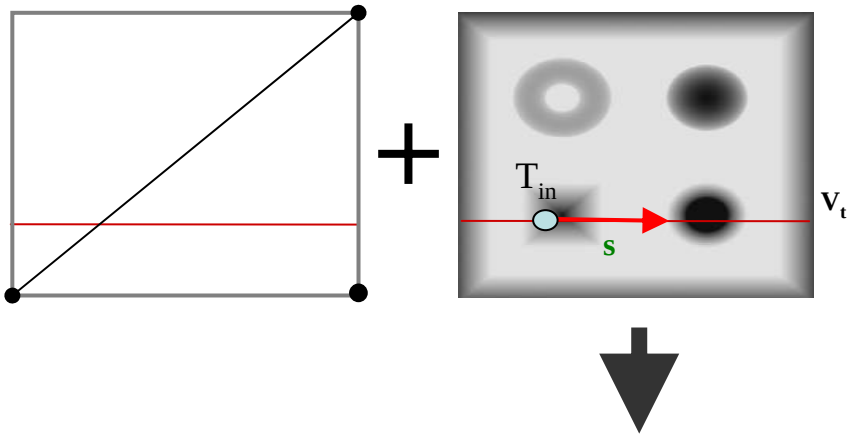
- Displacement Mapping (GPU)
  - Zutaten
    - Mesh
    - Höhen - Map



  - Resultat
    - detaillierte Oberfläche
  - Einschränkung
    - Entlang der Oberflächennormale
    - Keine konkaven Strukturen möglich



- Displacement Mapping (GPU)
  - Zutaten
    - Mesh
    - Höhen - Map
    - Normal - Map

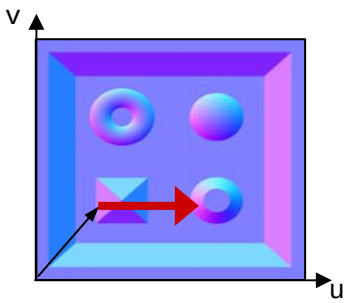


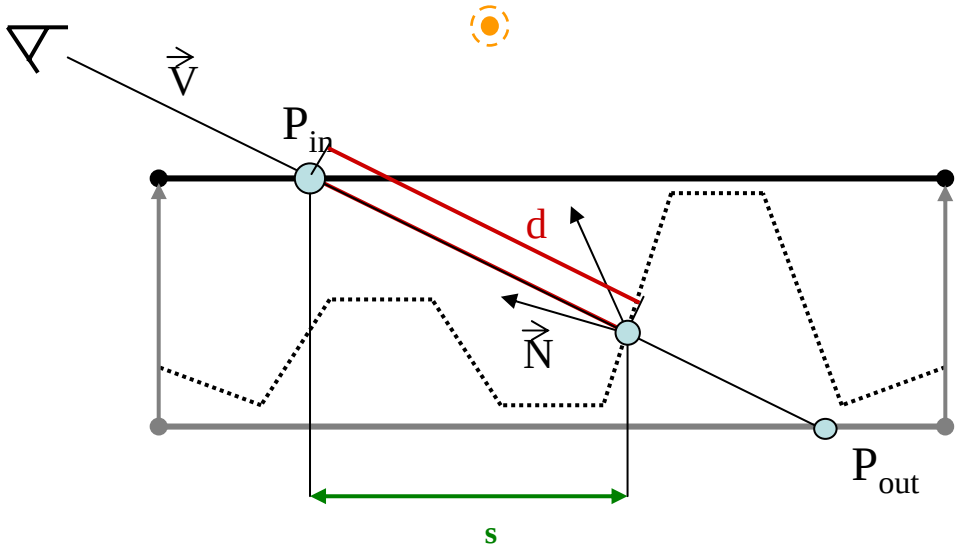
→Neue Position

$P = P_{in} + \mathbf{d}*V$

→Neue Textur-Koordinate

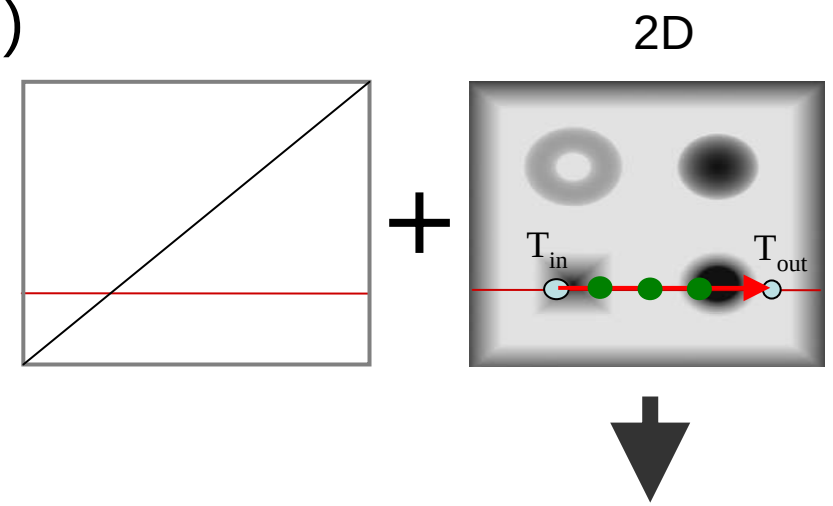
$T = T_{in} + \mathbf{s}*V_t$





- Displacement Mapping (GPU)
  - Relief Mapping
    - Mesh + Höhen Map (2D)
    - Raycasting:

```
for(s=0.0; s<=1.0; s += 0.2)
      depth = tex2d(relief_Map,
                    Tin + s*(Tout-Tin));
      if (depth < s)
          break;
```

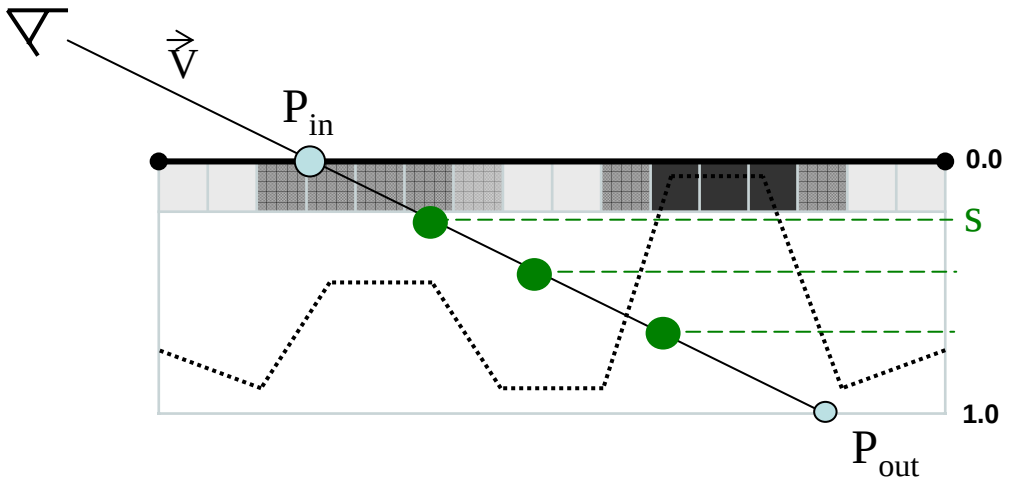


→Neue Position

$$P = P_{in} + s*(P_{out}-P_{in})$$

→Neue Textur-Koordinate

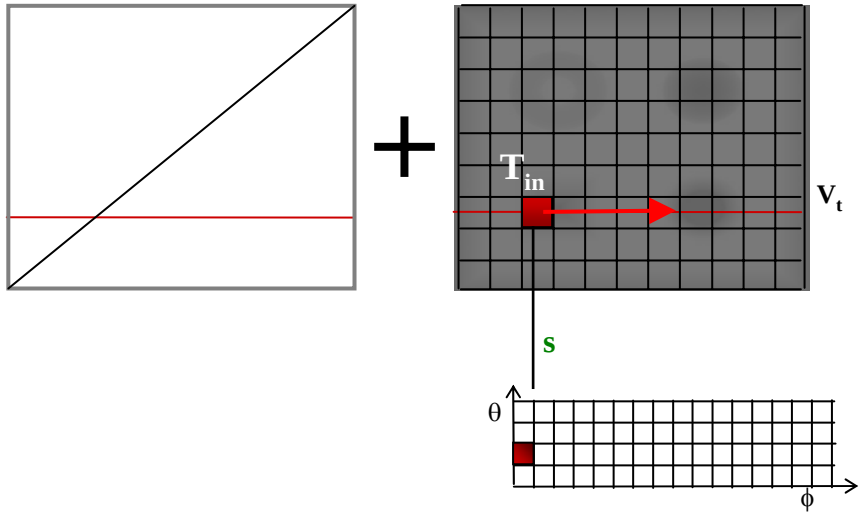
$$T = T_{in} + s*(T_{out}-T_{in})$$



Displacement Mapping (GPU)

- View Dependant Displacement
  - Mesh + VDM (4D)
  - Raycasting:

```
for(s=0.0; s<=1.0; s += 0.2)
        s = tex4d(vdm_Map,
                  Tin, V);
    if (depth < s)
        break;
```

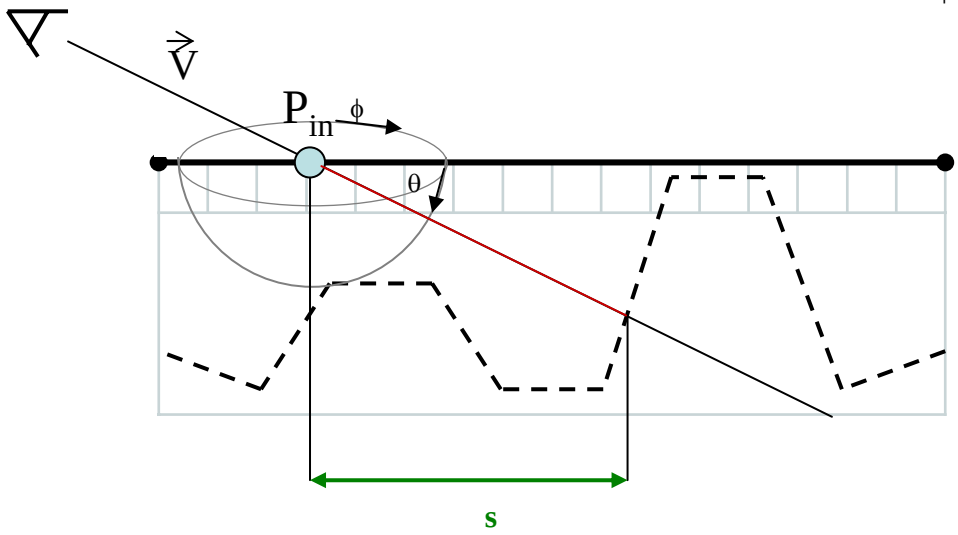


→Neue Position

$P = P_{in} + d*V$

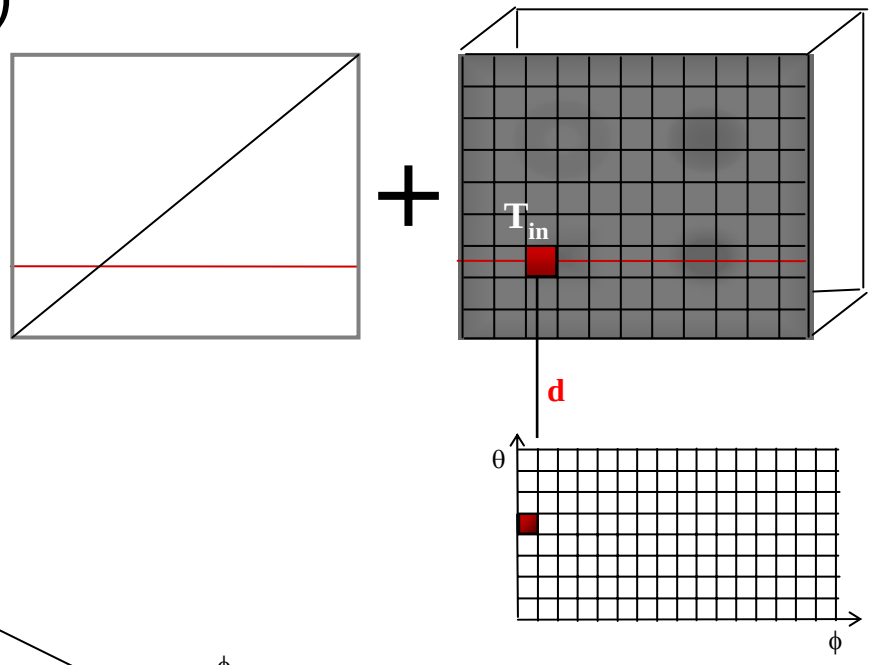
→Neue Textur-Koordinate

$T = T_{in} + s*V_t$



- Displacement Mapping (GPU)
  - Generalized Displacement Map
    - Mesh + GDM (5D)
    - Raycasting:

```
for(s=0.0; s<=1.0; s += 0.2)
        d = tex5d(gdm_Map,
                  Tin, V);
    if (depth < s)
        break;
```

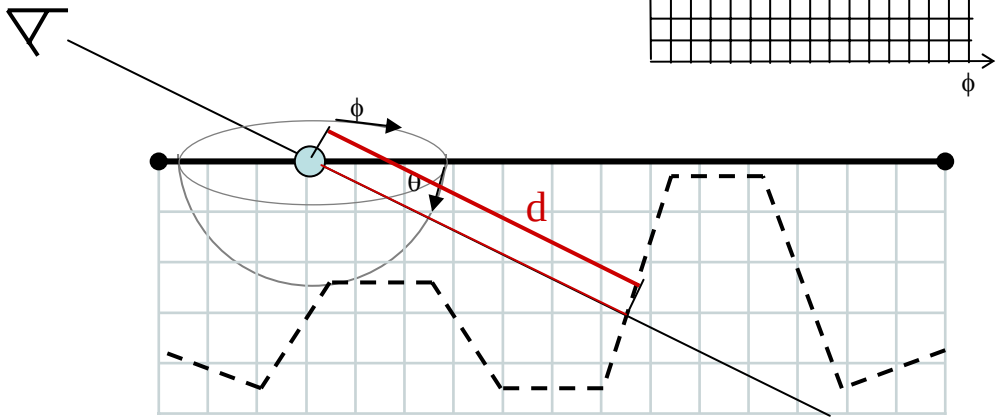


→Neue Position

$$P = P_{in} + d*V$$

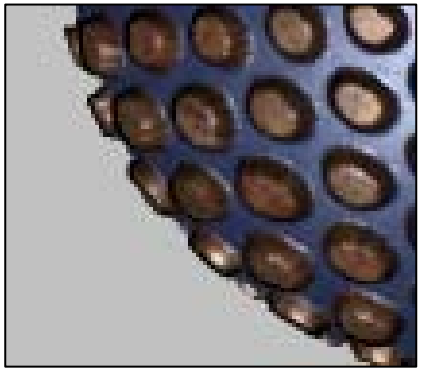
→Neue Textur-Koordinate

$$T = T_{in} + d*V_t$$

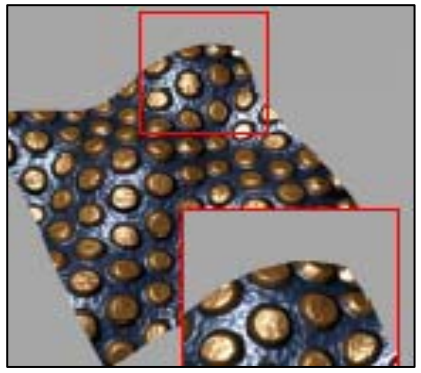


- Displacement Mapping – Spezialfälle
  - Silhouette

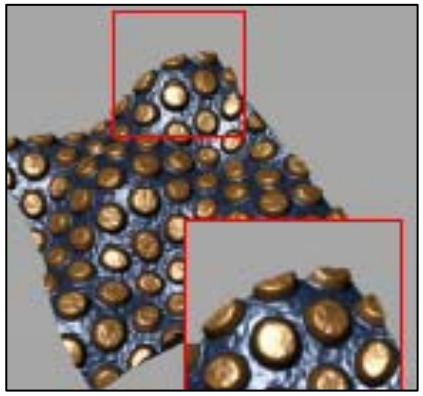
Relief Mapping

VDM 

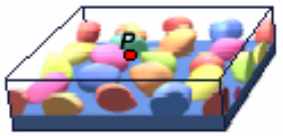
  - Silhouette bei offenem Mesh

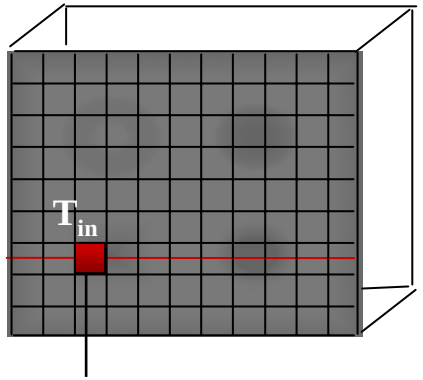


VDM

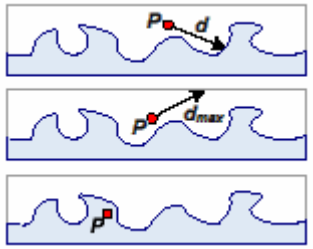
GDM 

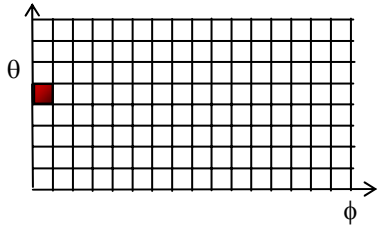
- Generalized Displacement Map
  - Modellierung
    - Gleichmäßiges 3D Gatter
    - Für jeden Punkt (x,y,z) → eine 2D Map
    - Entfernung zur Oberfläche messen



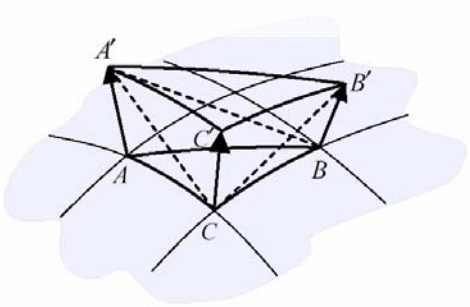


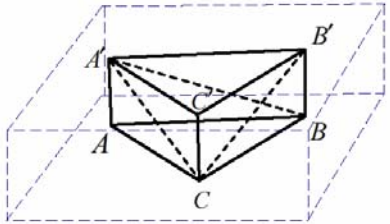
$$d_{GDM}(x,y,z,\theta,\phi) = \begin{cases} d \\ d_{max} \\ 0 \end{cases}$$

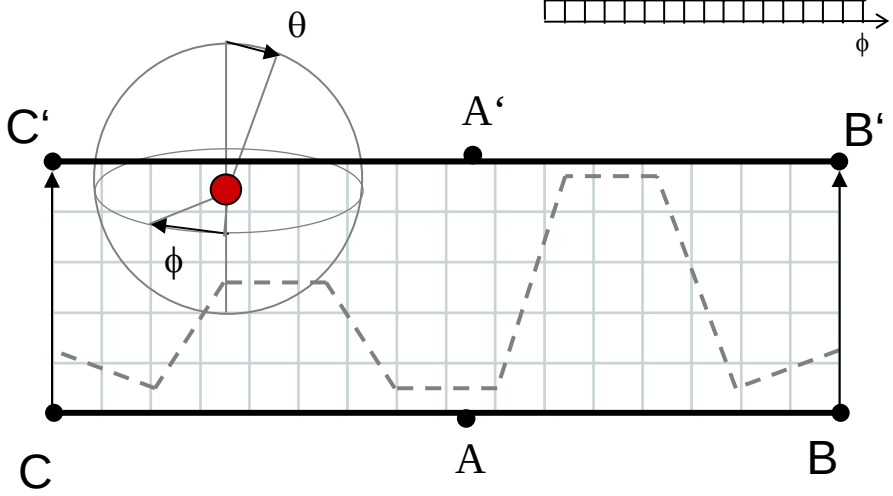




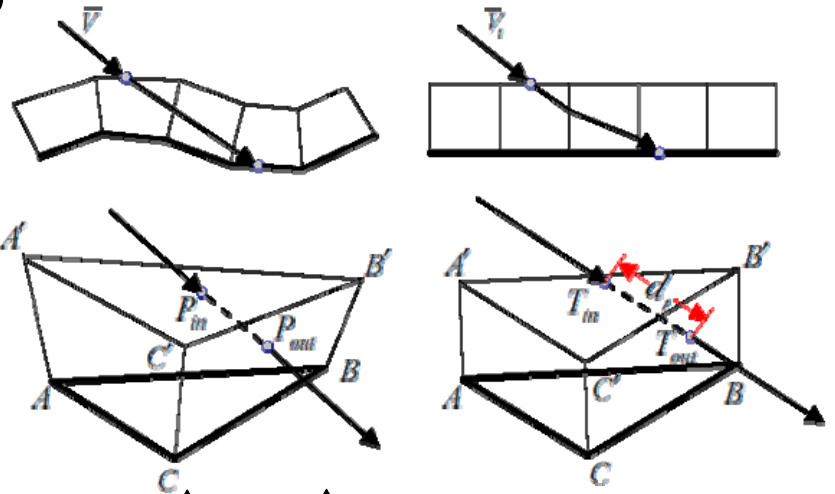
- Mesh modifizieren (V-Anzahl x 2)

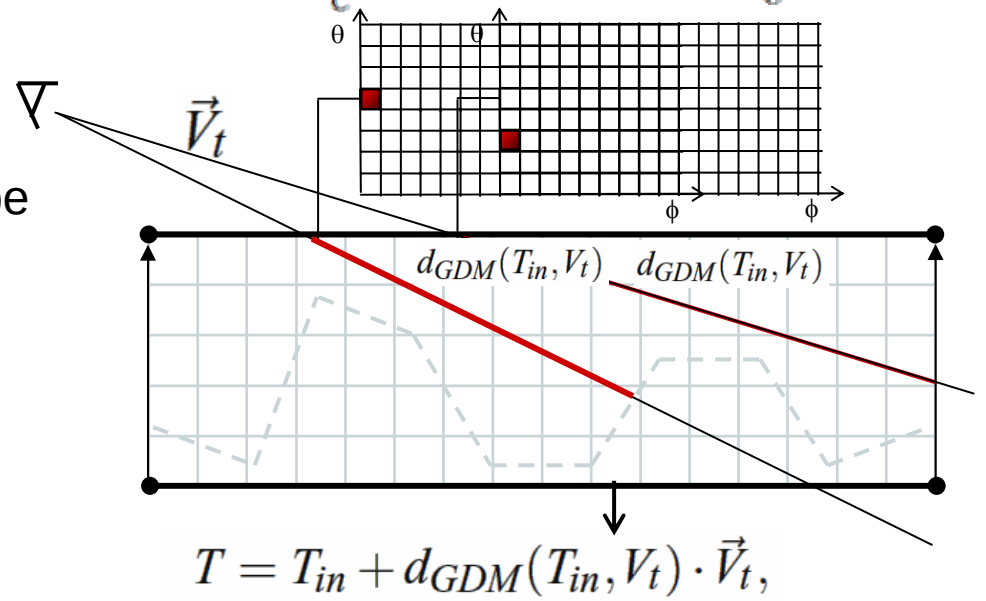






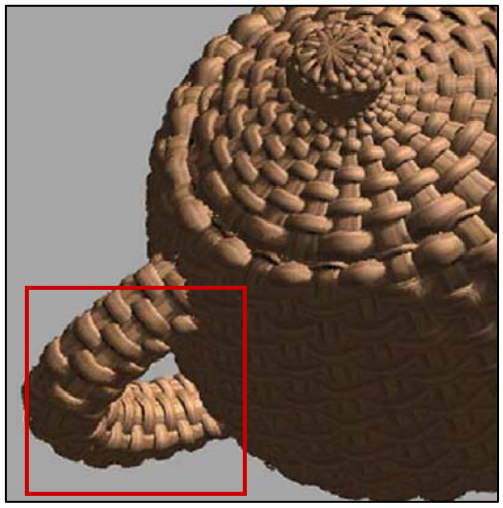
- Generalized Displacement Map
  - Rendering
    - Strahlenverfolgung im Objektraum
    - JEDES Dreieck rendern
    - Übertragen der Winkel in Texturraum
    - Vergleichen von $d_t = |T_{out} - T_{in}|$ mit $d_{GDM}(T_{in}, V_t)$
    - Wenn $d_{GDM}(T_{in}, V_t) < d_t$:
      - ➢ Berechnung von T
      - ➢ Berechnung der Farbe

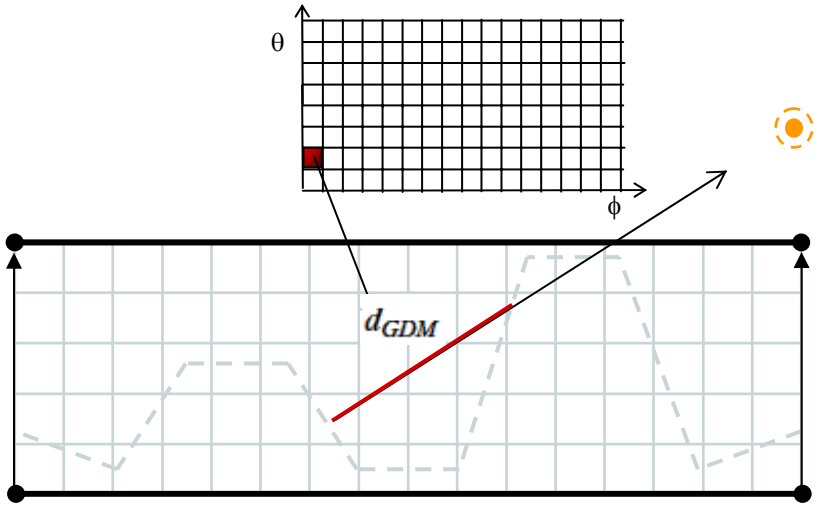




$$T = T_{in} + d_{GDM}(T_{in}, V_t) \cdot \vec{V}_t,$$

- Generalized Displacement Map
  - Lokale Beleuchtung
    - Wenn $\mathbf{d_{GDM}(x, y, z, \phi, \theta) < d_{max}}$ →Schatten
    - Kein Berücksichtigung von anderer Geometrie

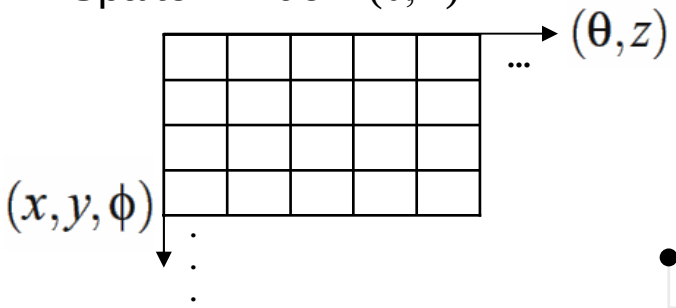




- Generalized Displacement Map
  - Optimierung
    - 1Farbkanal (8Bit):

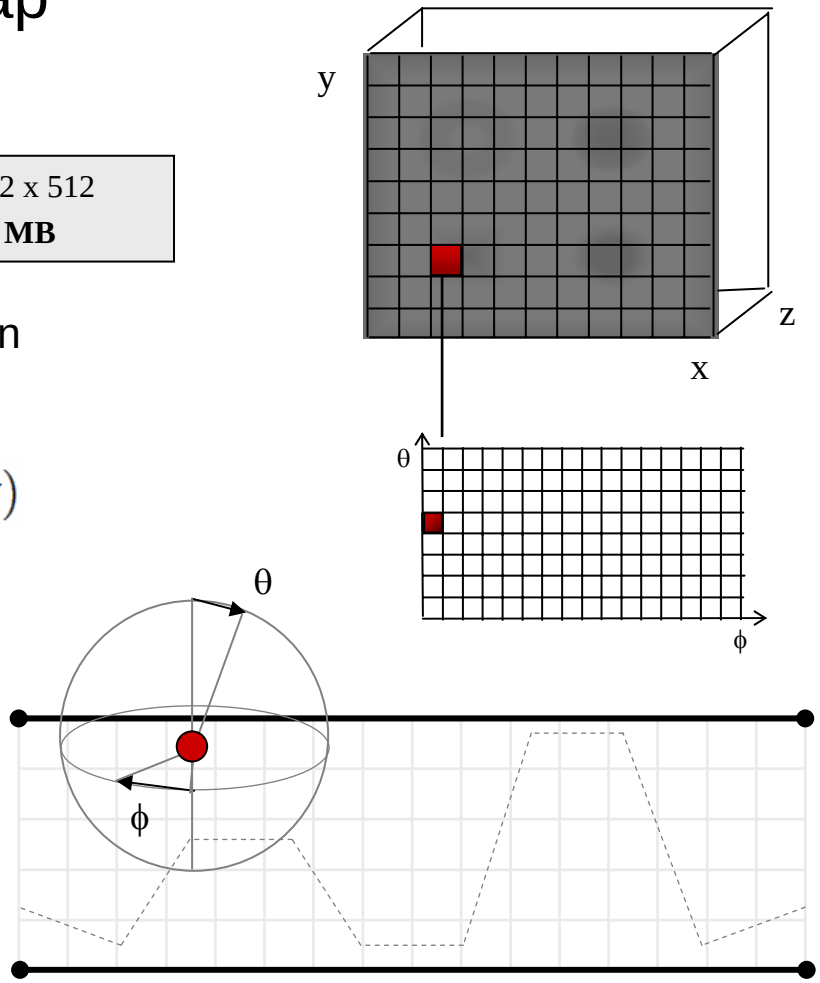
      (64 x 64 x 32) x (16 x 32) = 131.072 x 512
      = 67.108.864 Byte = 65.536K = **64 MB**

    - 5D Map in 2D Matrix umstrukturieren
      - Zeilen- Index: (x, y, ϕ)
      - Spalten- Index: (θ, z)

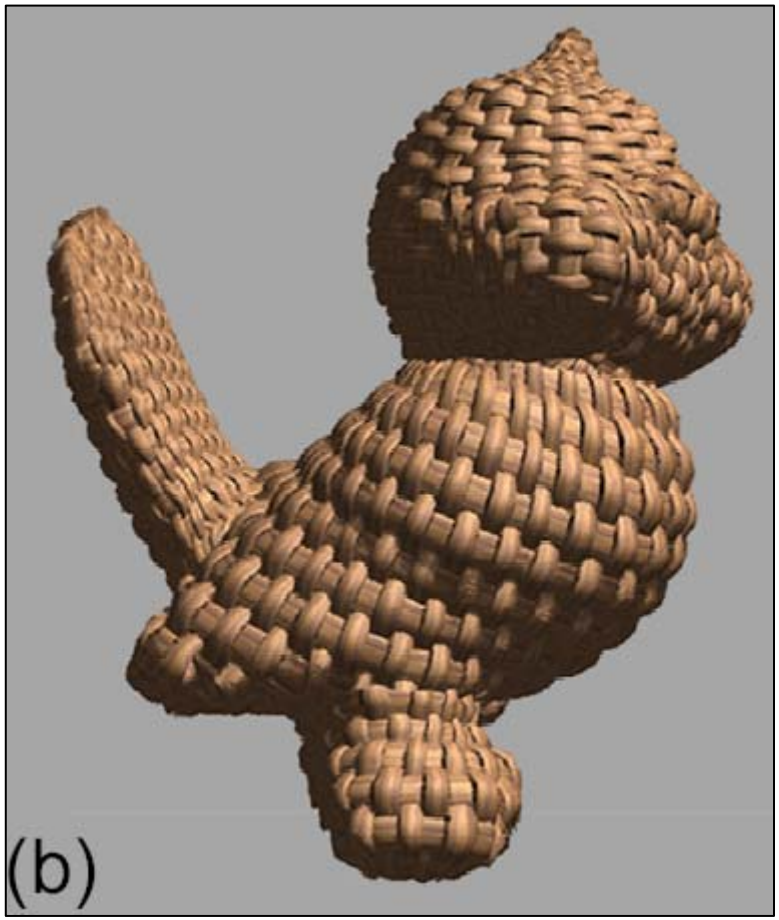


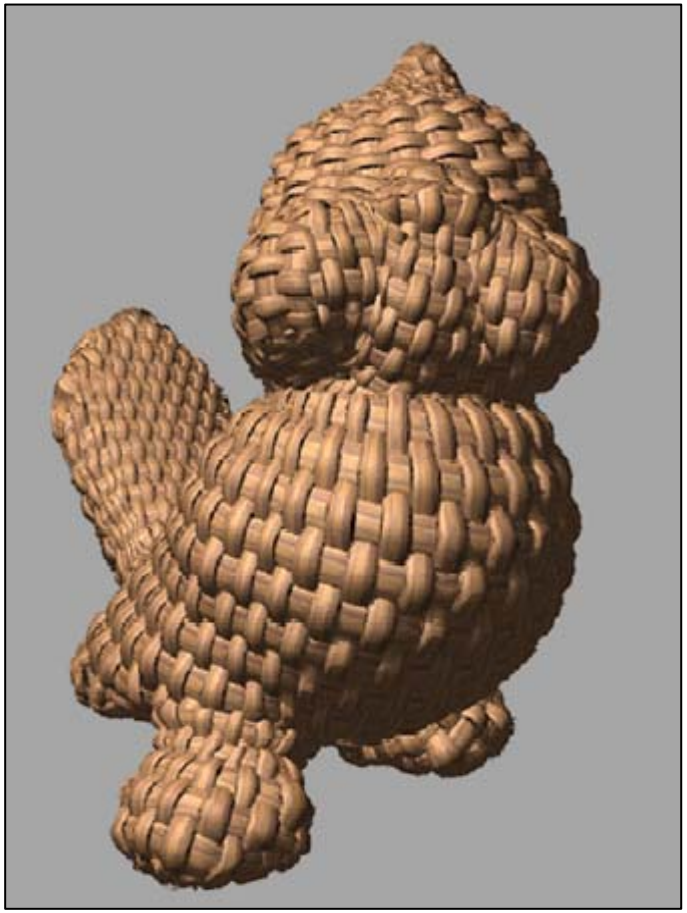
    - „Singular Value Decomposition“

→ $d_{GDM} = \sum_i W_i(x,y,\phi) E_i(\theta,z)$



- Generalized Displacement Map
  - Beispiele (auf P IV 2.8GHz + ATI 9800XT 256MB)



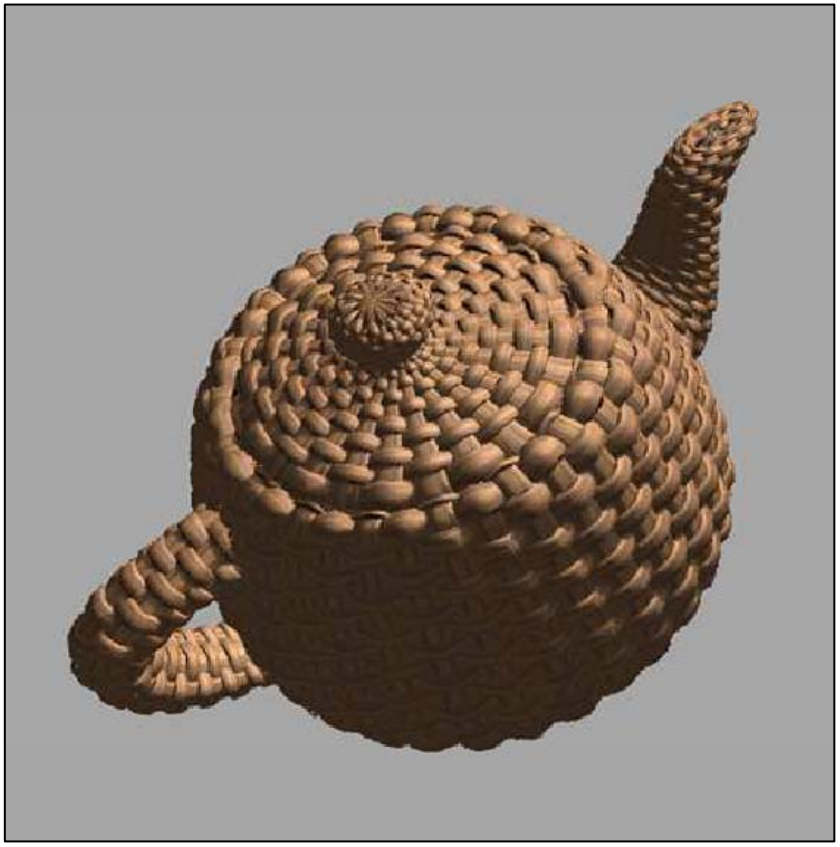


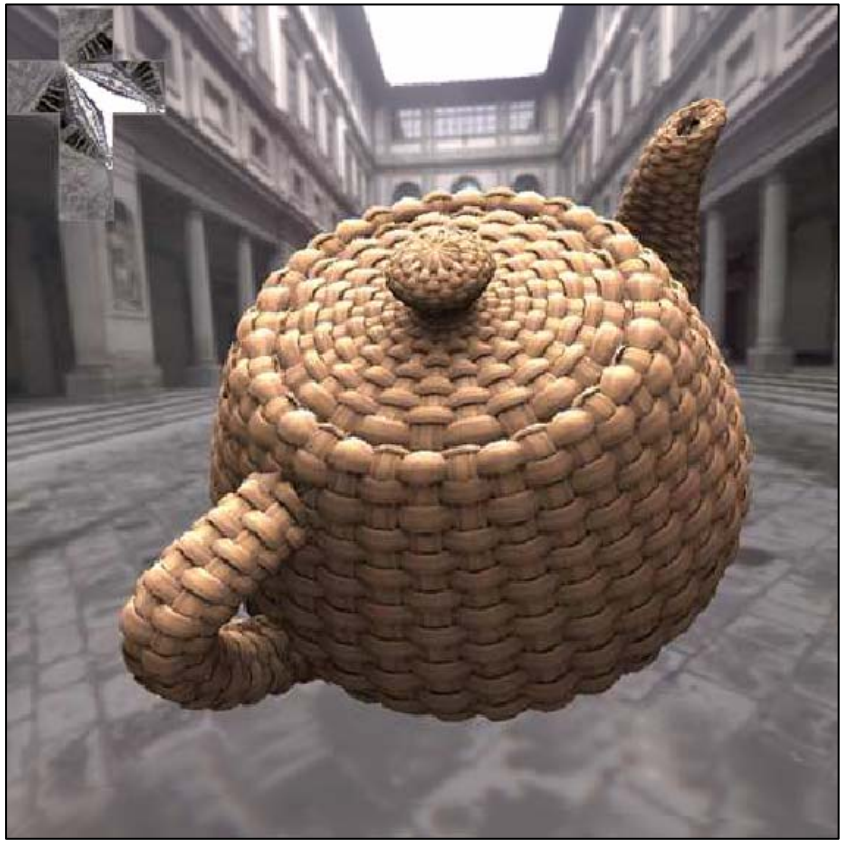
Bird	11603	Weave	64× 64×32	2MB	N/A	46.4

- Generalized Displacement Map
  - Beispiele (auf P IV 2.8GHz + ATI 9800XT 256MB)

Dinosaur	2142	Squama	64× 64×32	2MB	N/A	80.9

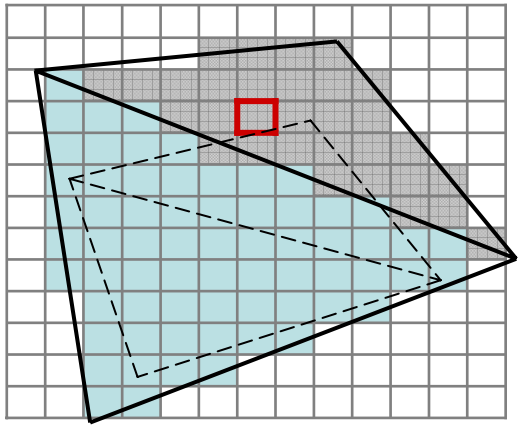
- Generalized Displacement Map
  - Beispiele (auf P IV 2.8GHz + ATI 9800XT 256MB)





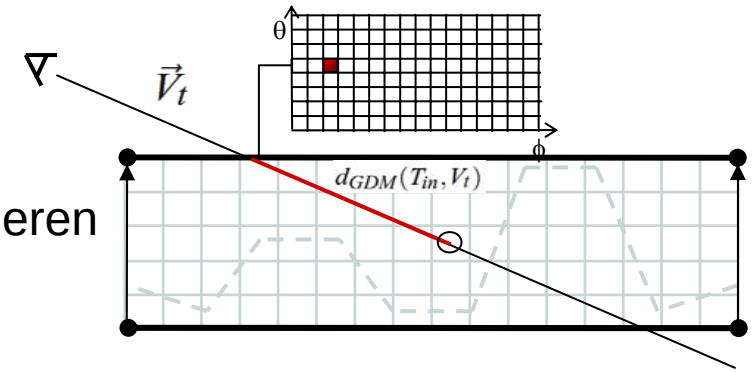
Teapot	4276	Weave	64× 64×32	2MB	8MB	56.0/16.3*

Zusammenfassung



- GPU Rendering
  - Vertex – Programm
    - Berechnung aller Vertexdaten
  - Fragment – Programm
    - Berechnung der Pixelfarbe mittels interpolierten Vertexdaten

- GDM
  - Modellierung
    - 5D Texturdaten generieren
    - Mesh → Bounding – Mesh generieren
  - Rendering
    - Per Pixel Raycasting
    - Neue Oberflächenposition mit GDM ermitteln
    - Beleuchtung → Farbe berechnen



Fragen?