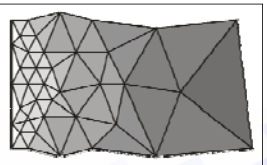
Visualisierung: 2D Skalarfelder

Andreas Kolb und Martin Lambers

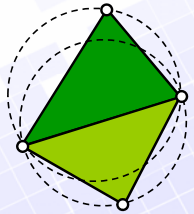
computergraphik und multimedia systeme
universität siegen

Review: Letzte Stunde

Triangulierungen:



- *Reguläre Triangulierungen*
- *Delaunay-Triangulierung*
  - Definition über Voronoi-Diagramm
  - globales und lokales Umkreis-Kriterium
- *Algorithmen zur Triangulierung:*
  - Flipping-Algorithmus
  - Inkrementeller Algorithmus
  - Divide & Conquer Algorithmus



Review: Letzte Stunde

- *Flipping Algorithmus*
  - Komplexität $o(n^2)$
  - sehr einfacher erfordert initiale Triangulierung
- *Inkrementeller Algorithmus*
  - Komplexität $o(n^2)$

Verwenden bei kleinen Netzen (optimal für $n < 1000$)

- *Divide & Conquer-Algorithmus*
  - Komplexität $o(n \log n)$
  - relativ kompliziert
- Nicht besprochen
  - *(Plane-Sweep Algorithmus)*
  - Komplexität $o(n \log n)$
  - sehr kompliziert

Verwenden bei großen Netzen (optimal für $n > 100000$)

Triangulierungen

Delaunay-Triangulierung nur in 2D

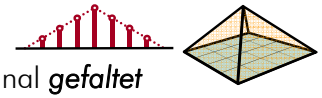
- Die *Triangulierung von Oberflächen in 3D* ist bisher ungelöst!

Mögliche Abhilfe: Hilfsgeometrie mit ebener Parametrisierung (z.B. Ebene, Zylinder, Kugel)

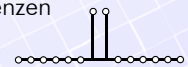
- Projiziere alle Punkte auf Hilfsbeometrie
- Bestimme Triangulierung dort
- Projiziere zurück

Review: Letzte Stunde

Lineare Filter:



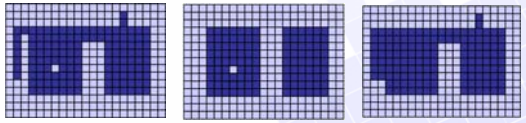
- *Filterkern* wird mit Signal *gefaltet*
- Lineare Filter entfernen bestimmte *Frequenzen* aus dem Signal
- Glättungsfilter:
  - Box-, Dreieck-, Gauss-Filter
  - *Tiefpassfilter:* hohe Frequenzen werden entfernt
- Kantenfilter:
  - *Hochpass-Filter:* niedrige Frequenzen werden abgeschwächt



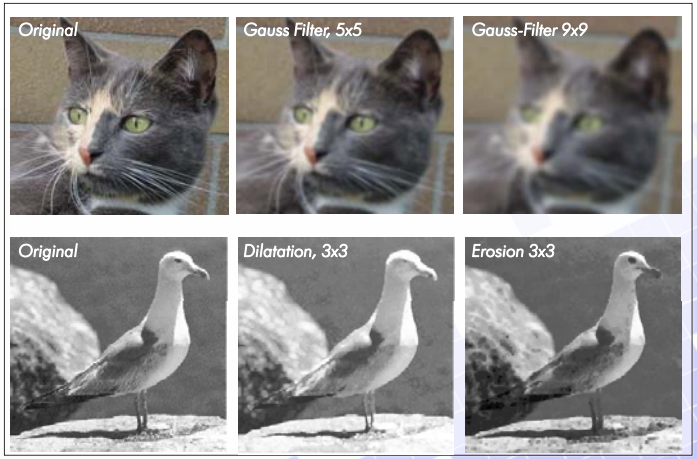
Review: Letzte Stunde

Nicht-Lineare Filter:

- Keine *Faltung,* sondern beliebige nichtlineare Verknüpfung
- Rangordnungsoperationen:
  - Median-Filter:
    Rauschunterdrückung ohne Kantenglättung
  - Morphologische Operationen
    Dilatation, Erosion, Opening, Closing

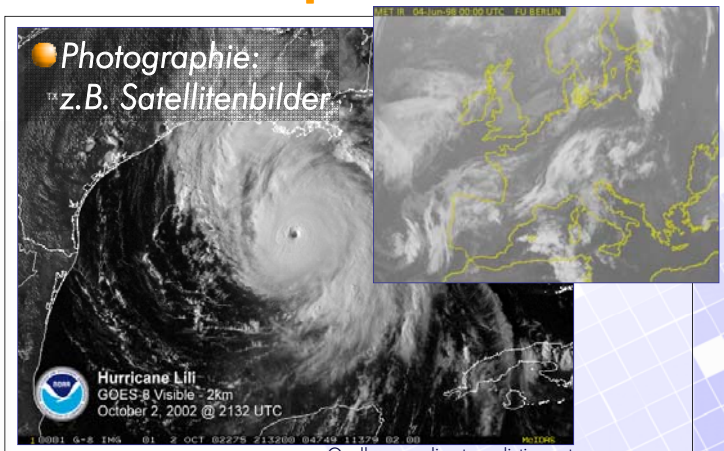


Bildbearbeitung



2D Datenbeispiele

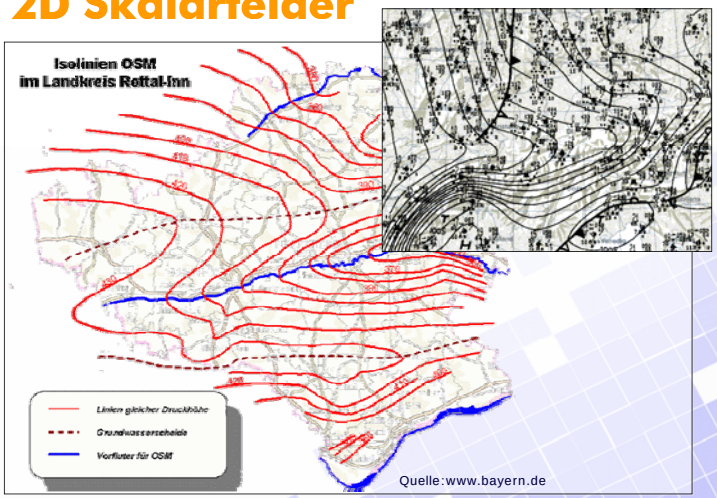
- *Photographie:*
  z.B. Satellitenbilder



Quelle: www.climateprediction.net

2D Skalarfelder

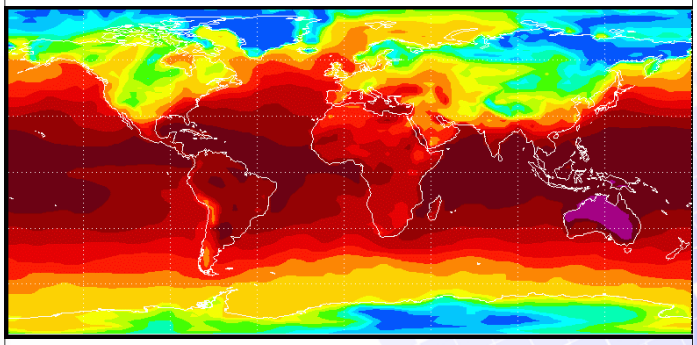
9



Beispiele

10

- **Meteorologie:** Temperaturverteilung

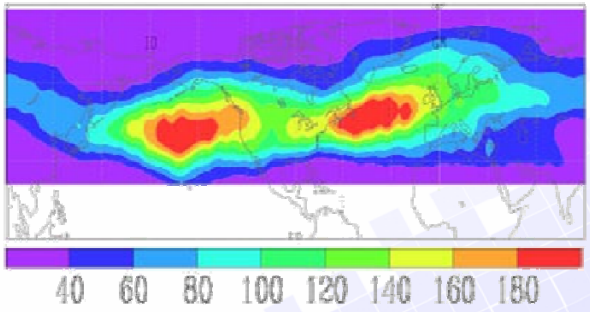


Quelle: www.climateprediction.net

Beispiele

11

- *Meteorologie:*
  Sturmbahnen (kinetische Energie der Luft)



Quelle: www.climateprediction.net

Überblick

12

2D Skalarfelder:

- selten analytisch gegeben: $s = f(x, y)$
- üblicherweise als diskrete Daten an Messpunkten $(x_i, y_i) : (x_i, y_i) \mapsto f_i \qquad i \in [0, \ldots N]$

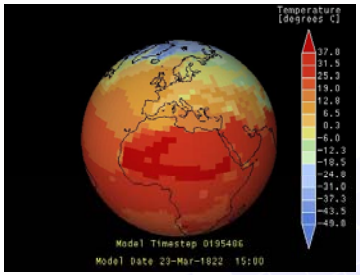
Visualisierungmöglichkeiten:

- *Color Plots (Farbcodierung)*
- *Surface Plots (Oberflächendarstellung)*
- *Isolinien*

2D Skalarfelder

13

- *„2D“* heißt nicht zwangsläufig *„kartesische* Koordinaten“!
- 2D Skalarfelder müssen nicht ***eben*** sein!



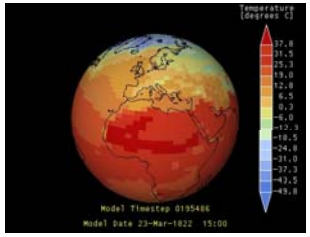
Quelle: www.climateprediction.net

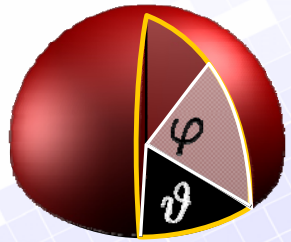
Beispiel: Temperaturverteilung über dem Globus

2D Skalarfelder

14

- ***Beispiel:*** Skalarfelder auf der Kugeloberfläche lassen sich in Polarkoordinaten (Kugelkoordinaten) parametrisieren: $s = f(\varphi, \vartheta)$



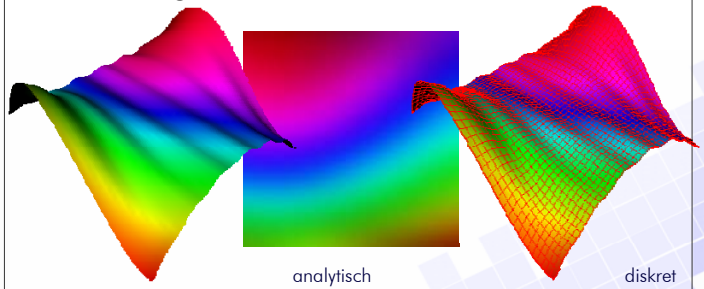


- z.B. Längen und Breitengrad

2D Skalarfelder

15

auf beliebigen Oberflächen in 3D



analytisch diskret

$$\begin{pmatrix} x(u,v) \\ y(u,v) \\ z(u,v) \end{pmatrix} \leftarrow (u,v) \rightarrow f(u,v) \qquad \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} \leftarrow (u_i, v_i) \rightarrow f_i$$

Farbkodierung

16

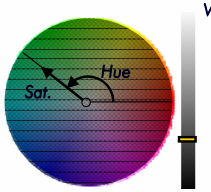
- Spezifiziere eine Farbtabelle

d.h eine Abbildung: Skalarwert s ➞ Farbwert

Beispiel: Gegeben 2D Druckverteilung

$$\rho_{min} \leq \rho \leq \rho_{max}$$

Lineare Abbildung im HSV-Farbmodell auf ***Hue***



Hue (Farbton) = Winkel

Saturation = Abstand vom Mittelpunkt

Value = Helligkeit (Brightness)

Farbkodierung

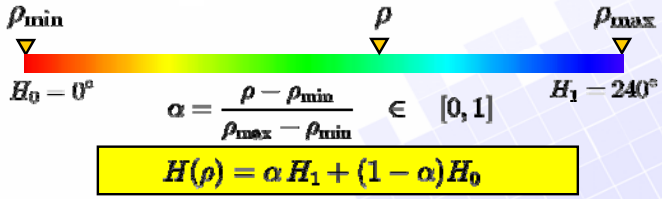
- Spezifiziere eine Farbtabelle

  d.h eine Abbildung: Skalarwert s ⟶ Farbwert

  Beispiel: Gegeben 2D Druckverteilung

  $\rho_{min} \leq \rho \leq \rho_{max}$

  Lineare Abbildung im HSV-Farbmodell auf *Hue*



Colorplots

Uniforme, rectilineare 2D Gitter

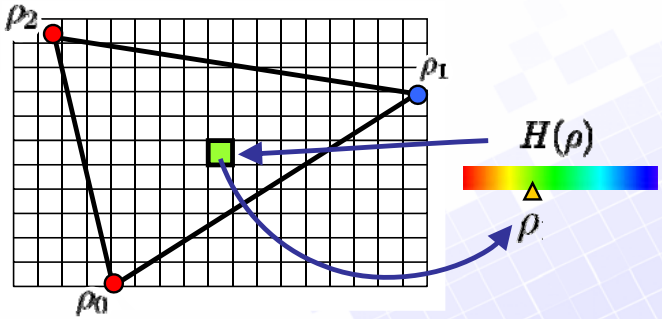
- *trivial bei feinen Gittern:*
  - interpretiere jeden Abtastpunkt als einen Pixel
  - Die Farbe des Pixels bestimmt der Skalarwert anhand der Farbtabelle
- *im allgemeinen Fall:* Rasterisierung/Interpolation

Curvilineare und Unstrukturierte 2D Gitter:

- Rasterisierungstechniken

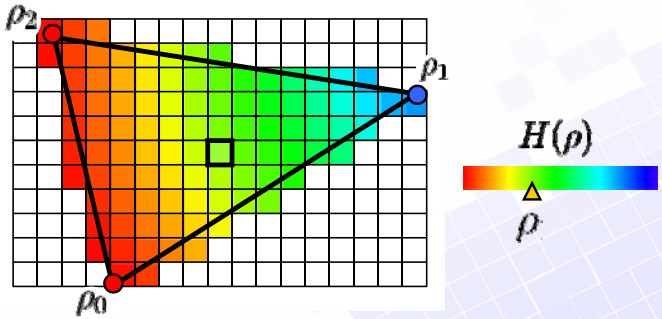
Rasterisierung

- Zerlege das Dreieck in einzelne Pixel
- Interpoliere für jeden Pixel den Skalarwert
- Bestimme die Farbe des Pixels anhand des Skalarwerts



Rasterisierung

- Zerlege das Dreieck in einzelne Pixel
- Interpoliere für jeden Pixel den Skalarwert
- Bestimme die Farbe des Pixels anhand des Skalarwerts



Einfache Implementierung

21

- *Nutzung von Graphik-Hardware:*
  Hardware übernimmt die Rasterisierung der Dreiecke

Beispiel OpenGL:

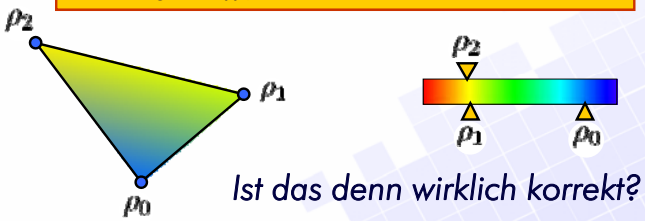
```
00    glBegin(GL_TRIANGLES);
01       glColor3d(R[i], G[i], B[i]);
02       glVertex2d(x[i], y[i]);
03       glColor3d(R[j], G[j], B[j]);
04       glVertex2d(x[j], y[j]);
05       glColor3d(R[k], G[k], B[k]);
06       glVertex2d(x[k], y[k]);
07    glEnd();
```

Ergebnis: OpenGL zeichnet ein Dreieck

Einfache Implementierung

22

```
00    glBegin(GL_TRIANGLES);
01       glColor3d(R[i], G[i], B[i]);
02       glVertex2d(x[i], y[i]);
03       glColor3d(R[j], G[j], B[j]);
04       glVertex2d(x[j], y[j]);
05       glColor3d(R[k], G[k], B[k]);
06       glVertex2d(x[k], y[k]);
07    glEnd();
```



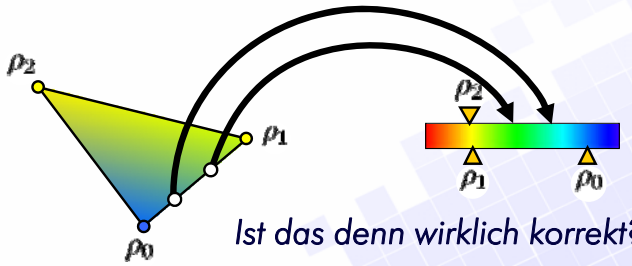
Ist das denn wirklich korrekt?

Einfache Implementierung

23

- Betrachte z.B. Punkte entlang der Kanten:

Farbinterpolation ist nicht korrekt!
Es muss der Skalarwert interpoliert werden!



Ist das denn wirklich korrekt?

Verbesserte Implementierung

24

- *Verwendung von 1D Texturen:*
  Speichere die Farbtabelle in einer 1D Textur:

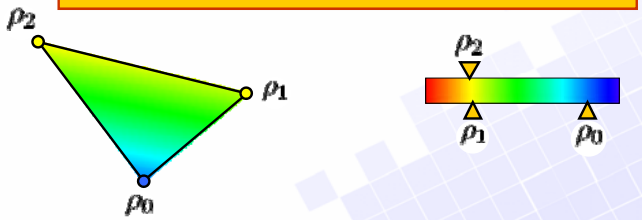
Beispiel OpenGL:

```
00    glBegin(GL_TRIANGLES);
01       glTexCoord1d(Rho[i]);
02       glVertex2d(x[i], y[i]);
03       glTexCoord1d(Rho[j]);
04       glVertex2d(x[j], y[j]);
05       glTexCoord1d(Rho[k]);
06       glVertex2d(x[k], y[k]);
07    glEnd();
```

Verbesserte Implementierung

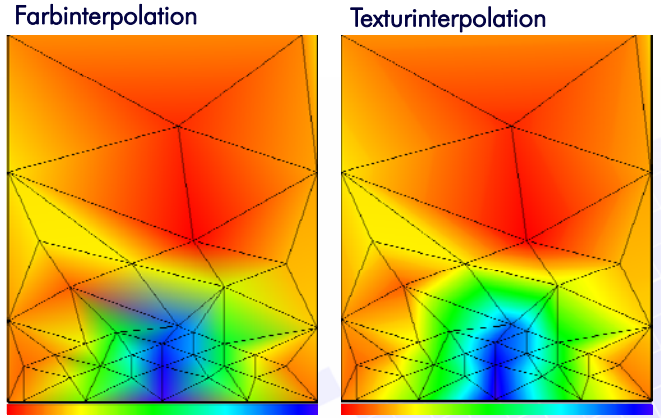
25

```
00    glBegin(GL_TRIANGLES);
01       glTexCoord1d(Rho[i]);
02       glVertex2d(x[i], y[i]);
03       glTexCoord1d(Rho[j]);
04       glVertex2d(x[j], y[j]);
05       glTexCoord1d(Rho[k]);
06       glVertex2d(x[k], y[k]);
07    glEnd();
```



Vergleich

26



Fazit

27

Farbkodierung von Skalarfeldern:

- Einfache Implementierung:
  - Bestimme Farbwerte für die Eckpunkte der Zellen (mittels Farbtabelle)
  - Interpoliere die Farbwerte für jeden Pixel im Inneren der Zelle

Per-Vertex Operations

„Gouraud-Shading"

- Aufwändigere Implementierung:
  - Interpoliere den Skalarwert für jeden Pixel im Inneren einer Zelle
  - Bestimme Farbwerte für jeden interpolierten Skalarwert (Farbtabelle)

Per-Pixel Operations

„Phong-Shading"

Zusammenfassung

28

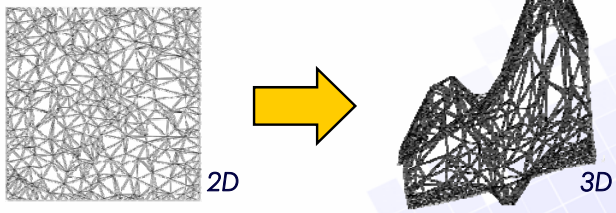
Color Plots / Farbcodierung

- Spezifiziere eine Farbtabelle, die den Skalarwert auf einen Farbwert abbildet *(= Transferfunktion)*
- Unstrukturierte und Curvilineare Gitter müssen rasterisiert werden
- *Rasterisierung* mit Grafik-Hardware
  - Einfach aber inkorrekt: Farbinterpolation *(Gouraud)*
  - Korrekte Interpolation des Skalarwerts mit 1D Texturen

Surface Plots

29

- *Oberflächendarstellung:*
  Interpretiere das Skalarfeld als Höhenfeld und zeichne eine 3D Oberfläche $(x_i, y_i) \mapsto (x_i, y_i, z_i)$

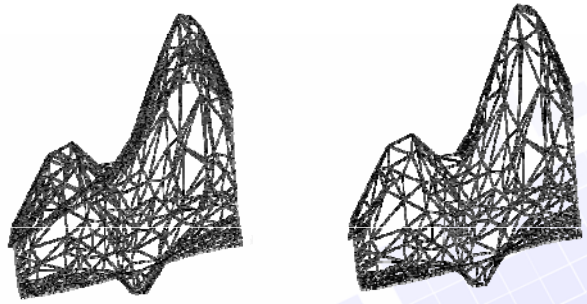


$$\alpha = \frac{\rho - \rho_{min}}{\rho_{max} - \rho_{min}} \qquad z_i = \alpha\, z_{max} + (1-\alpha) z_{min}$$

Surface Plots

30

- Liniendarstellungen



„wireframe“ „hidden line removal“

Hidden Line Removal

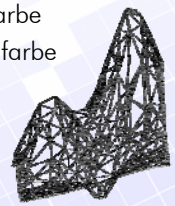
31

HLR Algorithmen für Vektorgrafik:

- z.B. Robert's Algorithm, Appel's Algorithm
  (siehe *Foley, et al: Computer Graphics, Principles and Practise)*

HLR für Rastergrafik:

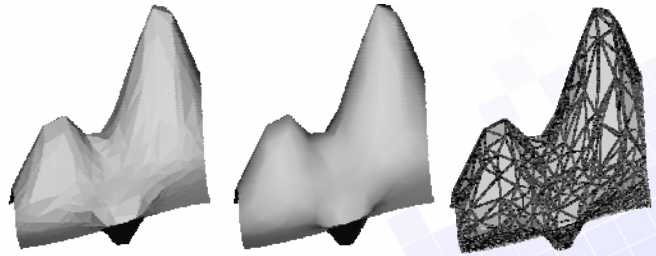
- OpenGL-Tricks:
  - Zeichne die Linien in Vordergrundfarbe
  - Zeichne die Flächen in Hintergrundfarbe
  - Der Z-Buffer besorgt den Rest.
    (siehe auch `glPolygonOffset`)



Surface Plots

32

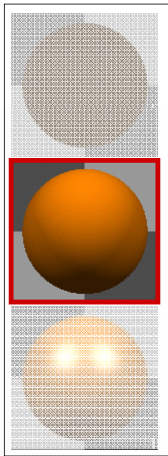
- Flächendarstellungen mit *Shading*
  (Lokale Beleuchtung)



flat shading *smooth shading Gouraud-Shading* *Kombination mit wireframe*

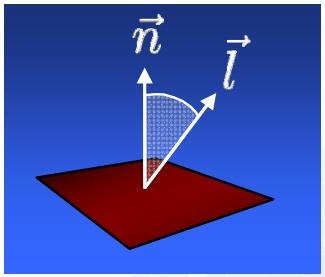
Diffuse Reflexion

33



Konstanter Term

Abhängig vom Einfallswinkel des Lichts

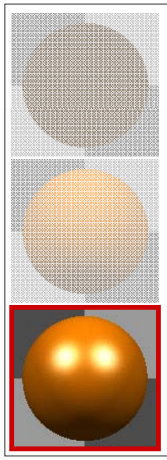


Abhängig vom Lichtrichtung und Blickrichtung

- gleichmäßige Reflexion in alle Richtungen
- matte Oberflächen
- „Lambert'sche" Reflexion

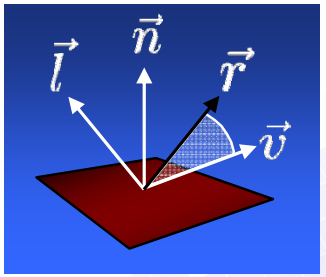
Spiegelnde Reflexion

34



Konstanter Term

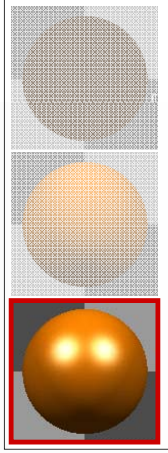
Abhängig vom Einfallswinkel des Lichts



Abhängig vom Lichtrichtung und Blickrichtung

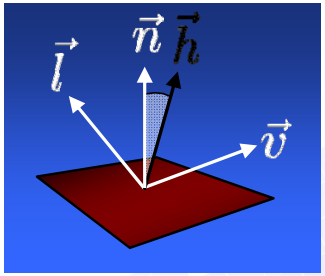
Spiegelnde Reflexion

35



Konstanter Term

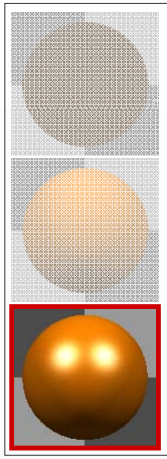
Abhängig vom Einfallswinkel des Lichts



Abhängig vom Lichtrichtung und Blickrichtung

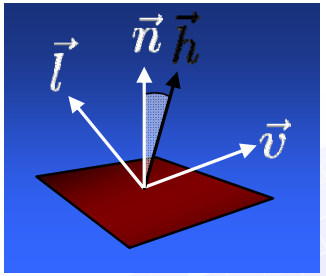
Spiegelnde Reflexion

36



Konstanter Term

Abhängig vom Einfallswinkel des Lichts



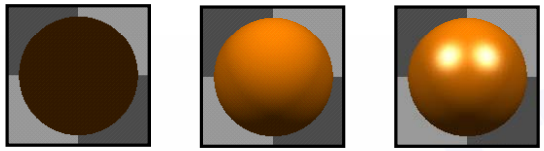
Abhängig vom Lichtrichtung und Blickrichtung

- glänzende Flächen
- Spiegelnde Reflexion
- „Specular Highlights"

Lokale Beleuchtung

37

Blinn-Phong Beleuchtungsmodell:



ambient | diffuse (Lambert) | spekular (spiegelnd)

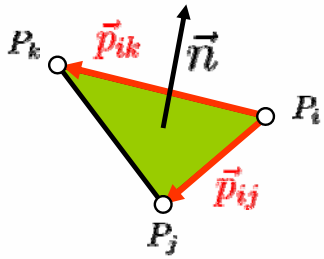
$$I = I_a + I_d(\vec{n} \circ \vec{l}) + I_s(\vec{n} \circ \vec{h})^r$$

- Wie bestimme ich die Oberflächennormale ?

Oberflächennormale

38

- Die Normale für ein einzelnes Dreieck:



$$\vec{p}_{ij} = P_j - P_i$$
$$\vec{p}_{ik} = P_k - P_i$$

Das normalisierte Kreuzprodukt ergibt den Normalenvektor

$$\vec{n} = \frac{\vec{p}_{ij} \times \vec{p}_{ik}}{\|\vec{p}_{ij} \times \vec{p}_{ik}\|}$$

Surface Plots

39

flat shading:
Bestimme einen Normalenvektor für jedes Dreieck

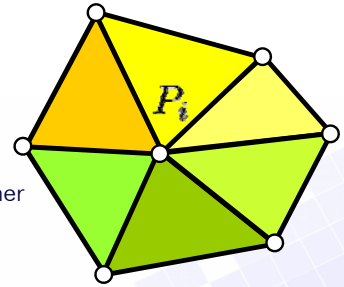
OpenGL Beispiel:

```
00    glBegin(GL_TRIANGLES);
01       glNormal3d(nx, ny, nz);
02       glVertex3d(x[i], y[i], z[i]);
03       glVertex3d(x[j], y[j], z[j]);
04       glVertex3d(x[k], y[k], z[k]);
05    glEnd();
```


Surface Plots

40

smooth shading:
Bestimme einen Normalenvektor für jeden Vertex

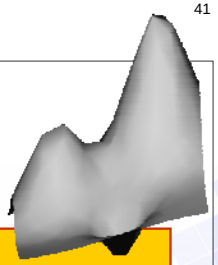


- Betrachte den Dreiecksfächer um den Punkt P_i
- Bestimme die Normalen-vektoren für alle Dreiecke
- Der Normalenvektor der Punktes P_i ergibt sich als Mittelwert der Normalen der angrenzenden Dreiecke

Surface Plots

smooth shading:

Bestimme einen Normalenvektor für jeden Vertex



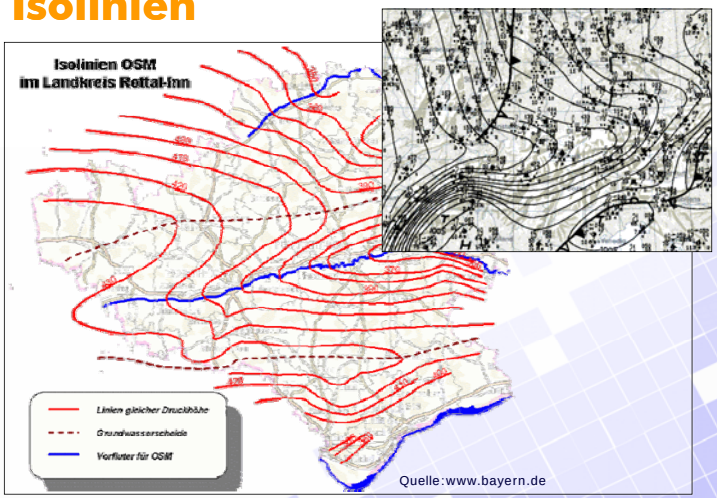
```
00    glShadeModel(GL_SMOOTH);
01
02    glBegin(GL_TRIANGLES);
03       glNormal3d(nx[i], ny[i], nz[i]);
04       glVertex3d(x[i], y[i], z[i]);
03       glNormal3d(nx[j], ny[j], nz[j]);
05       glVertex3d(x[j], y[j], z[j]);
03       glNormal3d(nx[k], ny[k], nz[k]);
06       glVertex3d(x[k], y[k], z[k]);
07    glEnd();
```


Zusammenfassung

Surface Plots / Oberflächendarstellung

- Interpretiere den Skalarwert als Höhenwert und zeichne eine 3D Fläche
- Liniendarstellung:
  wireframe und hidden lines
- Flächendarstellung mit Beleuchtung:
  - benötigt Normalenvektoren
  - Normalenvektor pro Dreieck *(flat shading)*
  - Normalenvektor pro Vertex *(smooth, Gouraud shading)*

Isolinien



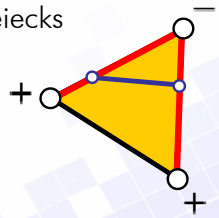
Definition

- Für ein gegebenes Skalarfeld $f : \mathbb{R}^2 \mapsto \mathbb{R}$ und einen Skalarwert $c \in \mathbb{R}$ entspricht die Menge
$$\{(x, y) \mid f(x, y) = c\}$$
einer Isolinie.
- Falls f differenzierbar ist und $\mathbf{grad}(f) \neq 0$, dann sind die Isolinien *Kurven*.
- Falls $\mathbf{grad}(f) = 0$, können die Isolinien auch *Flächen* und *Punkte* sein.

Isolinien

Sequentieller Ansatz (cell-order approach):

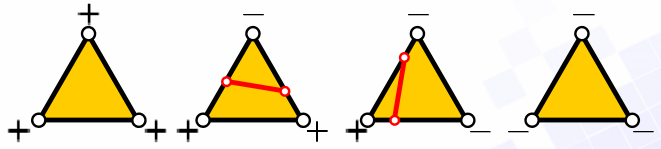
- Betrachte jede Zelle separat
- Markiere alle Vertices des Dreiecks
  - mit + wenn $f_i > c$
  - mit − wenn $f_i \leq c$
- Untersuche alle Kanten mit Vorzeichenwechsel
- Interpoliere Schnittpunkte entlang der Kanten
- Verbinde die Schnittpunkte durch Liniensegmente



Isolinien für Dreiecksgitter

Für dreieckige Zellen:

- entweder keine oder zwei Kanten mit Vorzeichenwechsel

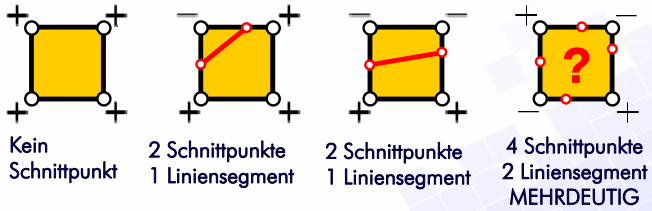


(symmetrische Fälle wurden weggelassen)

Isolinien für Rechtecksgitter

Für rechteckige Zellen:

- entweder keine, zwei oder vier Kanten mit Vorzeichenwechsel

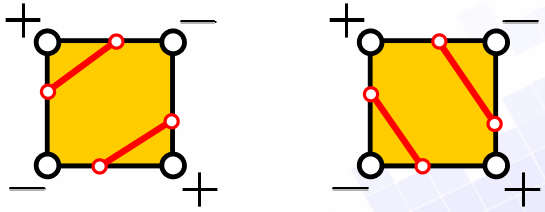


(symmetrische und invertierte Fälle wurden weggelassen)

Isolinien für Rechtecksgitter

Mehrdeutigkeit:

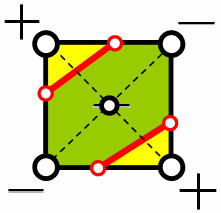
- vier Kanten mit Vorzeichenwechsel
  2 mögliche Konfigurationen:



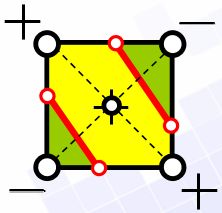
Verwende einen *„Decider"* um zu entscheiden welche Konfiguration vorliegt.

Isolinien für Rechtecksgitter

- **Midpoint Decider**
  Interpoliere den Skalarwert im Mittelpunkt des Rechtecks:



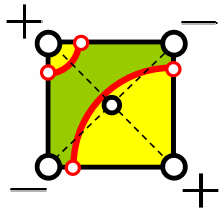
$f_M \leq c$



$f_M > c$

Isolinien für Rechtecksgitter

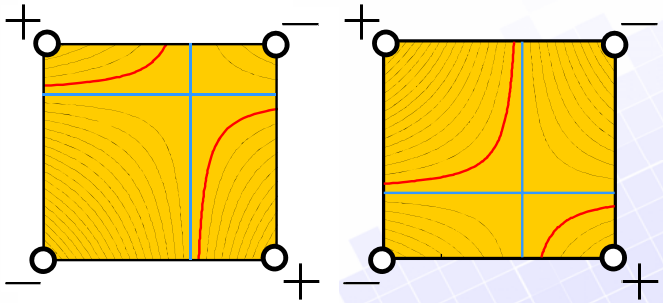
- **Midpoint Decider**
  Schwachpunkt des Midpoint Deciders:
  Isolinien sind keine Geraden, sondern Kurven



- Hier ist $f_M > c$, obwohl der Mittelbereich negativ ist

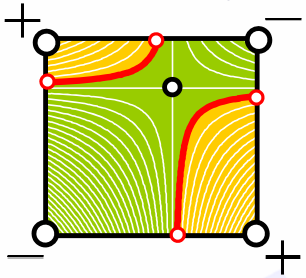
Isolinien

Bei bilinearer Interpolation innerhalb der Zelle sind die Isolinien Parabelbögen



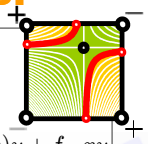
Isolinien für Rechtecksgitter

- **Asymptotic Decider**
  Bestimme den Skalarwert nicht am Mittelpunkt, sondern am *Schnittpunkt der Asymptoten*



- Der Schnittpunkt der Asymptoten liegt immer innerhalb des Mittelbereichs

Isolinien für Rechtecksgitter



- **Asymptotic Decider**

  Stelle die Interpolationsfunktion auf:

$$f(x,y) = f_{00}(1-x)(1-y) + f_{10}x(1-y) + f_{01}(1-x)y + f_{11}xy$$

  Gradient verschwindet am Asymptoten-Schnittpunkt

$$\frac{\partial f}{\partial x}(x,y) = (1-y)(f_{10}-f_{00}) + y(f_{11}-f_{01}) = 0$$

$$\frac{\partial f}{\partial y}(x,y) = (1-x)(f_{01}-f_{00}) + x(f_{11}-f_{10}) = 0$$

$$(x_0,y_0) = \left(\frac{f_{00}-f_{01}}{f_{00}+f_{11}-f_{10}+f_{01}}, \frac{f_{00}-f_{10}}{f_{00}+f_{11}-f_{10}+f_{01}}\right)$$

$$\text{und } f(x_0,y_0) = \frac{f_{00}f_{11}+f_{10}f_{01}}{f_{00}+f_{11}-f_{10}+f_{01}}$$

Isolinien

Nachteile des *sequentiellen Ansatzes:*

- Jeder Vertex der Isolinie und jede Kante des Gitters wird zweimal bearbeitet
- Das Ergebnis ist eine ungeordnete Folge von Liniensegmenten. Diese müssen erst sortiert werden um Polygonzüge zu erhalten.

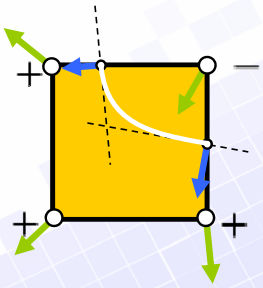
→ **Contour Tracing**

Contour Tracing

- Markiere alle Zellen mit Vorzeichenwechsel
- Solange es noch markierte Zellen gibt:
  - Finde die Isoline innerhalb dieser Zelle
  - Solange die Isolinie nicht geschlossen ist und der Rand des Datensatzes nicht erreicht ist:
    - Folge der Isoline in die Nachbarzelle
    - Entferne die Markierung (außer die Zelle enthält 4 Schnittpunkte)

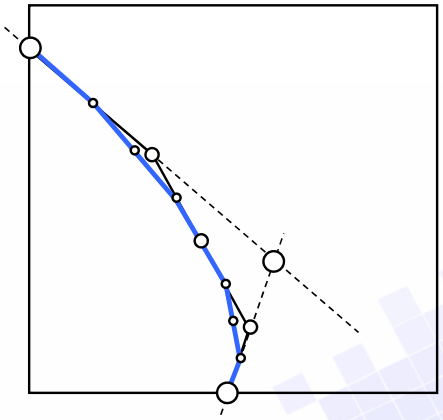
Isolinien geglättet

- Bestimme die Gradientenvektoren an den Vertices
- Interpoliere die Gradientenvektoren an den Schnittpunkten
- Bestimme Tangenten an den Schnittpunkten (senktrecht auf Grad.)
- Zeichne einen Bogen, der an den Schnittpunkten die Tangenten interpoliert



Isolinien (Subdivision)

57



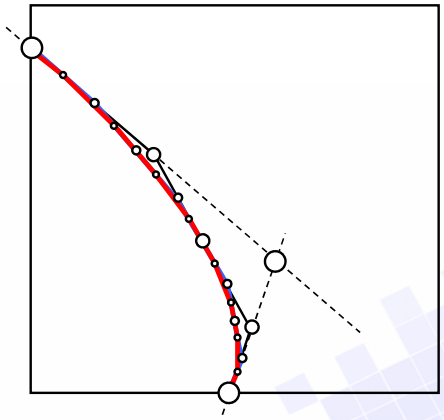
quadratischer Beziér-Kurve

Approximation durch wieder-holte Unterteilung bei $1/2^n$

siehe Vorlesung *Computergraphik 2*

Isolinien (Subdivision)

58



quadratischer Beziér-Kurve

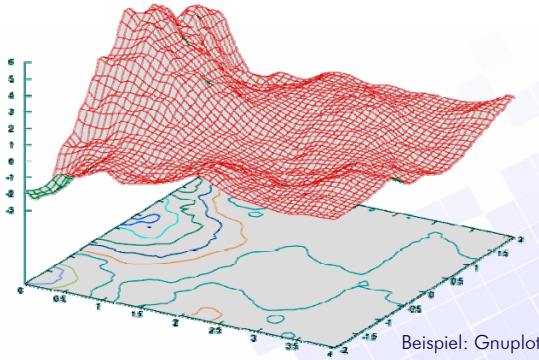
Approximation durch wieder-holte Unterteilung bei $1/2^n$

siehe Vorlesung *Computergraphik 2*

Kombination

59

- Isolinen + Surface Plot

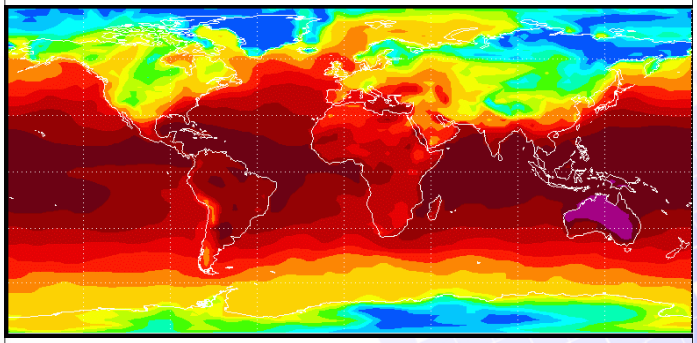


Beispiel: Gnuplot

Kombination

60

- Farbkodierung + Isolinien

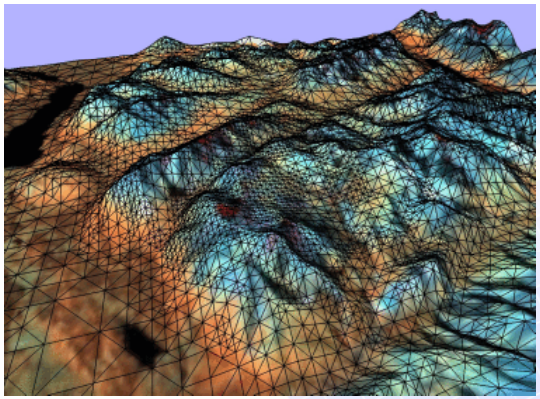


Quelle: www.climateprediction.net

Kombinationen

61

- Farbkodierung + Surface Plot

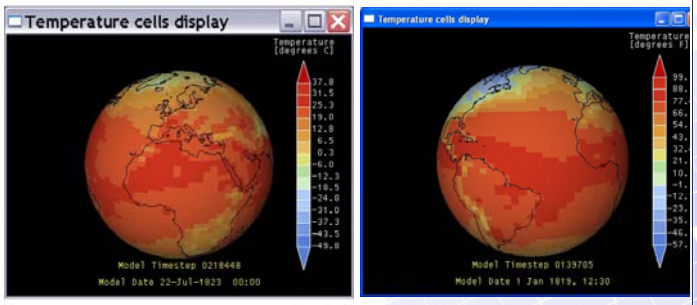


Zeitabhängige Skalarfelder

62

Animation

- Zeitliche Änderung der Temperaturverteilung



Quelle: www.climateprediction.net

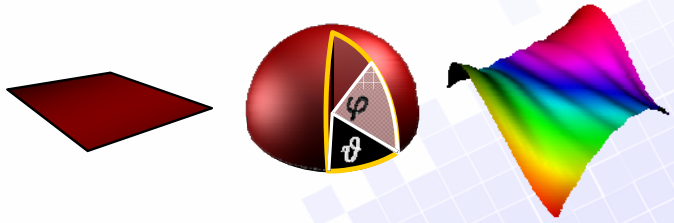
Zusammenfassung

63

2D Skalarfelder:

- Skalarwert gegeben über einem beliebigen 2D Definitionsbereich

  (z.B. Ebene, Kugeloberfläche, beliebige Oberfläche in 3D)

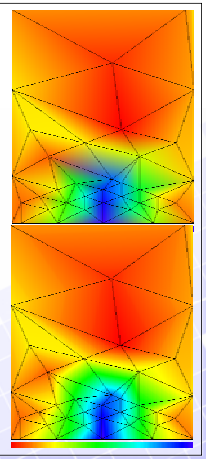


Zusammenfassung

64

Farbkodierung

- Spezifiziere Farbtabelle *(Transferfunktion)*
- *Rasterisierungstechniken:*
  - *Schnell und einfach:* Lookup an den Vertices + Farbinterpolation pro Pixel
  - *Genauer:* Skalarwert-Interpolation pro Pixel + Lookup in Bildschirmauflösung



Zusammenfassung

65

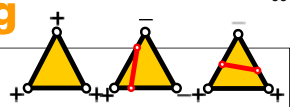
Oberflächendarstellung:

- Linienbasiert
  - Wireframe
  - Hidden Lines (erfordert Verdeckungsberechnung)
- Beleuchtete Oberflächen (Shaded Surface)
  - *Flat-Shading:* Normalenberechnung pro Fläche
  - *Gouraud-Shading:* Normalenberechnung pro Vertex

Zusammenfassung

66

Isolinien:



- Sequentieller Ansatz (cell-order appraoch)
  - Untersuche ob Isolinie die Zelle schneidet
  - Bestimme entsprechendes Liniensegment
- Für rektilineare Zellen: Mehrdeutigkeit erfordert „Decider“:
  - Midpoint Decider: einfach aber ungenau
  - Asypmtotic Decider: genauer

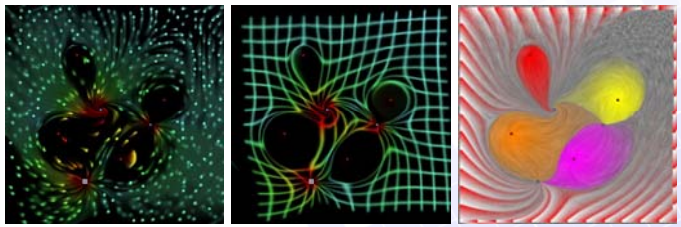
- Contour-Tracing: besser als sequentiell
- Glatte Isoflächen: Gradientenbestimmung

Nächste Stunde

67

2D Vektorfelder

- Differenzieren auf Vektorfeldern
- Integration von Partikelbahnen



Quelle: Jarke v. Wijk, Technische Universiteit Eindhoven, SIGGRAPH 2003