Shader kreativ

Scharfe Texturen, Bewegungsunschärfe und Stofffalten

Jörn Loviscach
Hochschule Bremen
jlovisca@informatik.hs-bremen.de
www.L7H.cn

Einführung

Standardanwendungen
für aktuelle Grafikhardware:

- Verformungen per Vertexshader
- Texturierung (inkl. Bump etc.)
  im Pixelshader

Demo

Einführung

Standardanwendungen
für aktuelle Grafikhardware:

- Verformungen per Vertexshader
- Texturierung (inkl. Bump etc.)
  im Pixelshader

Demo

Aber ist das alles?

Überblick

- Einführung
- Randscharfe Texturen
- Bewegungsunschärfe
- Stofffalten
- Ausblick

Randscharfe Texturen

Randscharfe Texturen

Problem: Hart berandete Grafiken wie Straßenzeichen, Etiketten usw. werden mit Standardtexturen zu unscharf oder zu voluminös

hohe Auflösung | niedrige Auflösung | DXT1 | neues Verfahren

gleicher Speicherbedarf

Randscharfe Texturen: ähnliche Ansätze

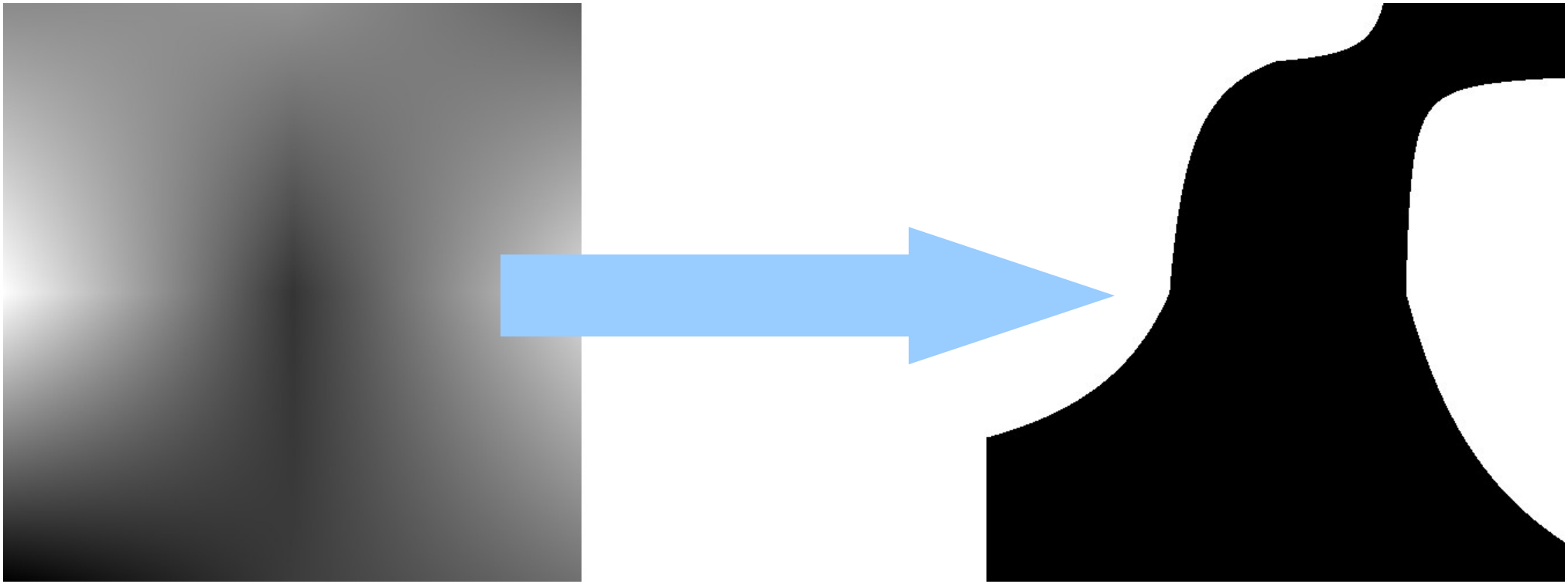
Ramanarayanan, Bala, Walter: Feature-based textures, 2004

Tumblin & Choudhury: Bixels, 2004

Sen: Silhouette maps, 2004

Randscharfe Texturen: Grundidee

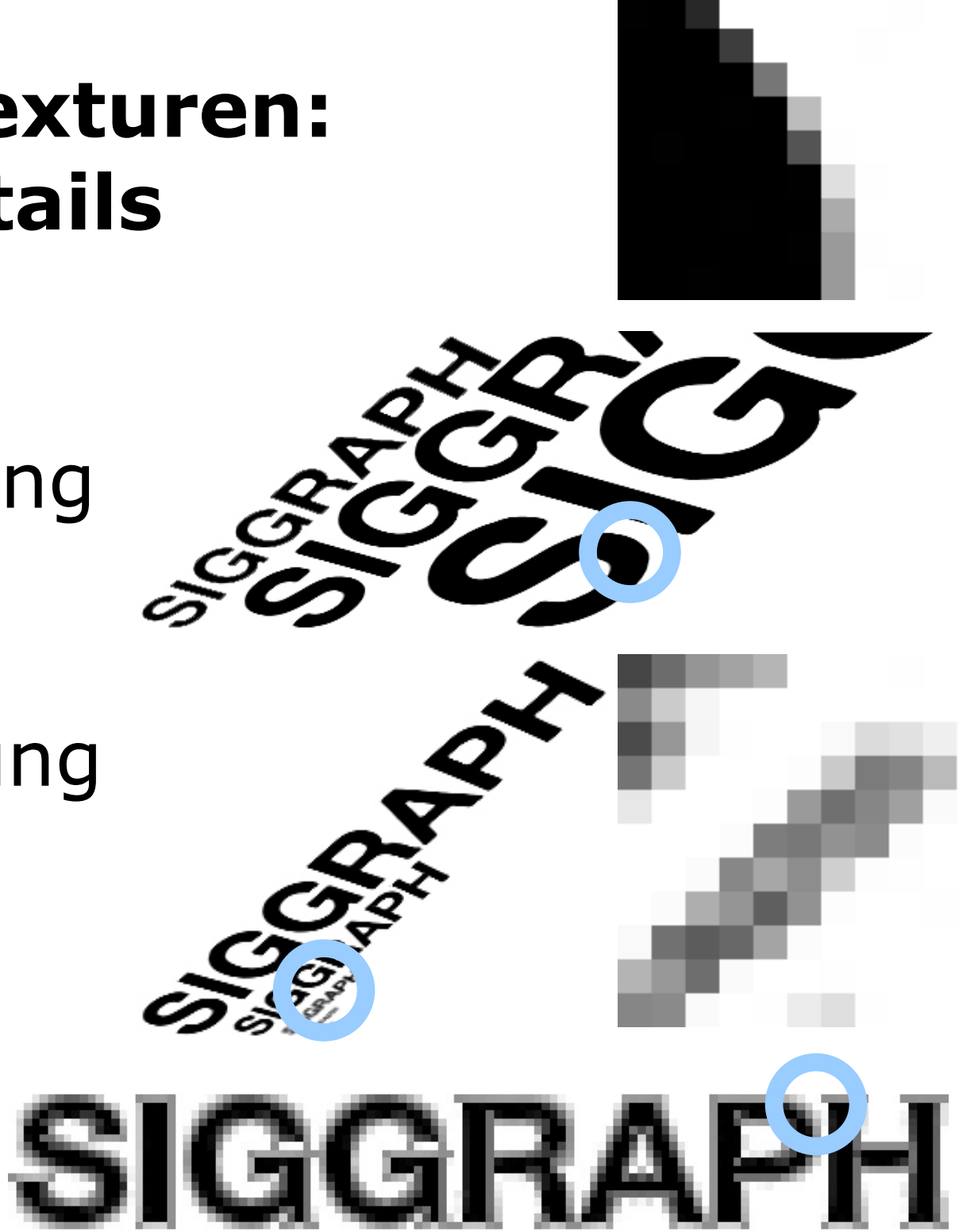
Bilde Schwellwert
von bilinearer Interpolation



Demo

Randscharfe Texturen: die nötigen Details

- Antialiasing bei Vergrößerung
- MIP-Mapping bei Verkleinerung
- Textur für Schwellwert optimieren

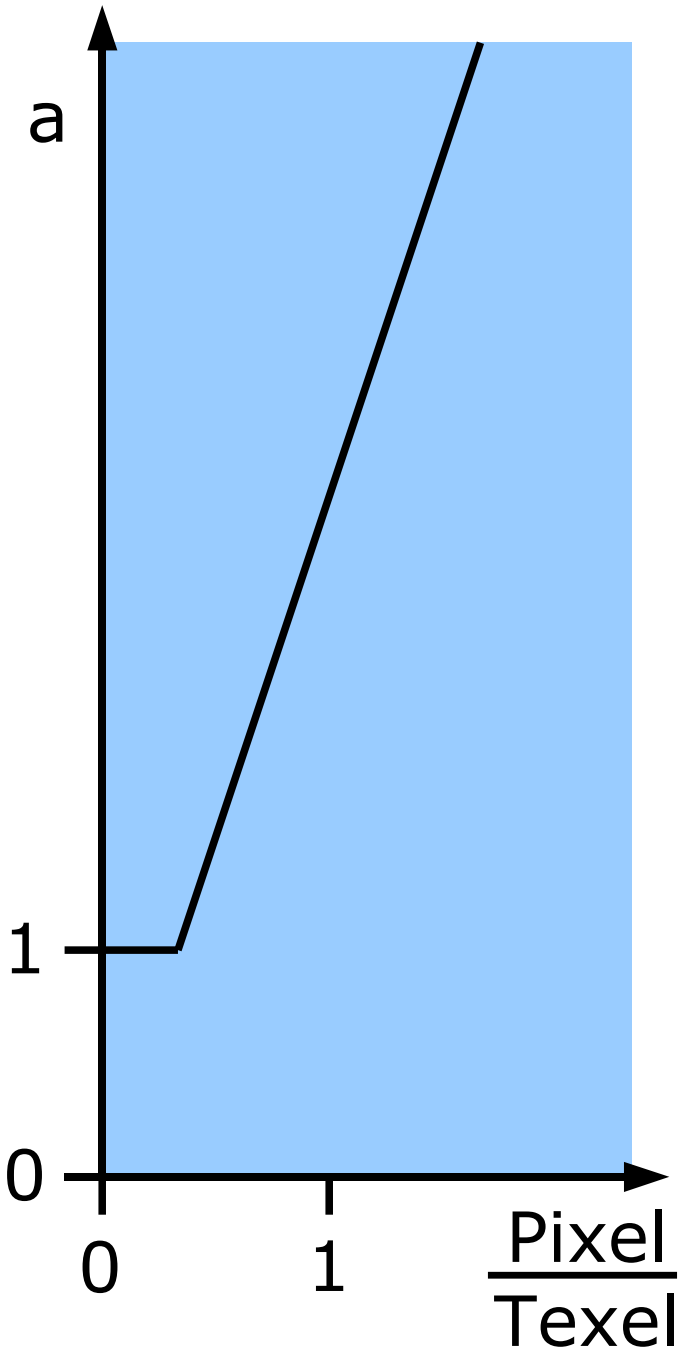


Randscharfe Texturen: Vergrößerung

Weicher Übergang
von rechnerisch
1/3 Bildschirmpixeln

Einfach und billig:
clamp(0.5+a*(tex-0.5))

ddx, ddy für Verhältnis
von Pixeln zu Texeln



Randscharfe Texturen: Verkleinerung

MIP-Map wie bekannt, aber feinster Level optimiert

Feinster Level darf keine Werte in der Nähe von 127 enthalten

Randscharfe Texturen: Optimierung (1)

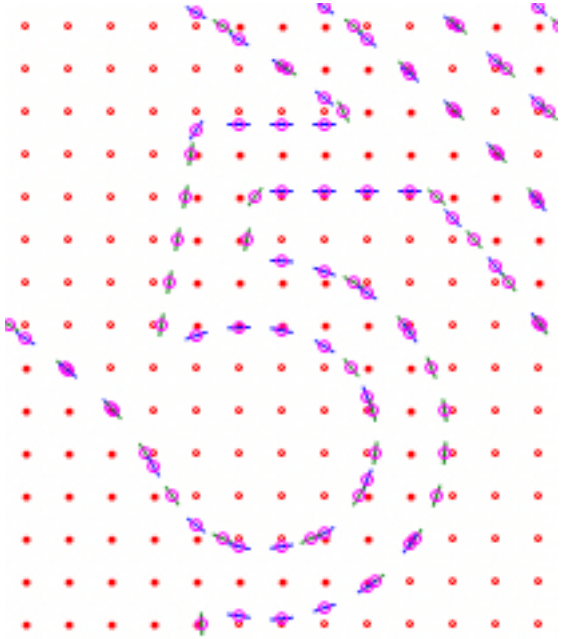
MIP-Map Level 0 offline vorbereiten

Eingabe: hochaufgelöste Bitmap

Ausgabe: Texturdatei in .dds-Format

Randscharfe Texturen: Optimierung (2)

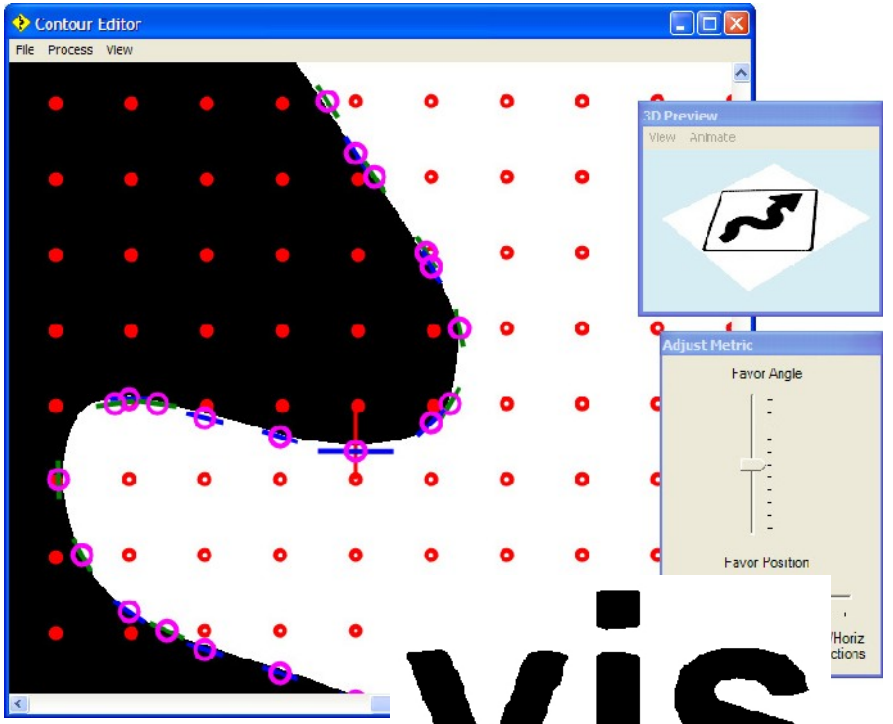
Schnittpunkte und
-richtungen auf Texelkanten
bestimmen



Geünscht: optimale
Übereinstimmung
mit diesen Daten

Keine exakte Lösung,
da (leicht) überbestimmt

Randscharfe Texturen: Optimierung (3)



Software:

- Schnittpunkte/ -richtungen editieren
- Optimierung online im Hintergrund
- Metrik für Optimierung

vis
vis
vis

Demo

Randscharfe Texturen: Ergebnisse

800 MPixel/s auf GF 6800 GT,
14 Pixel-Shader-Befehle
inkl. einer Texturabfrage

Grenzen der Qualität:

Randscharfe Texturen: weitere Infos

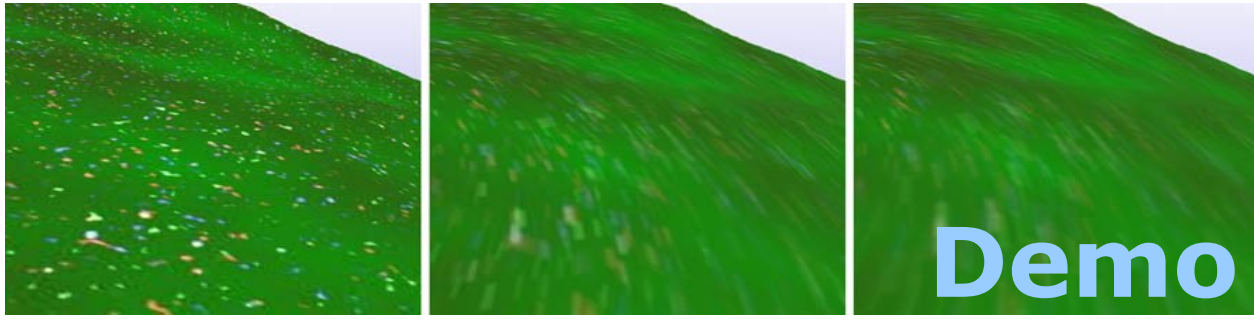
- SIGGRAPH 2005 Sketch
- Game Programming Gems 6

Bewegungsunschärfe

Bewegungsunschärfe

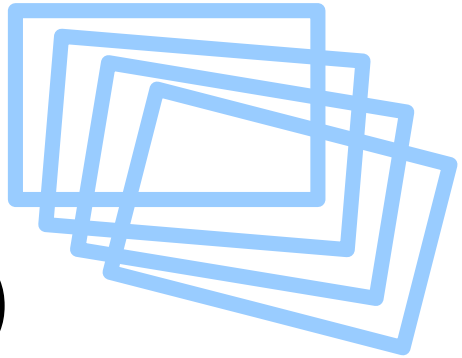
Oft genügt Bewegungs-unschärfe für fast ebene Objekte, ggf. mit Aus-stanzungen per Alpha:

- Terrain
- Schilder
- Räder
- Propeller
- Klingen
- ...

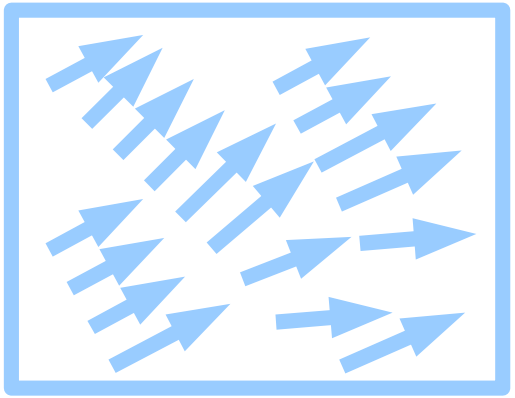


Bewegungsunschärfe: klassisch

- Zeitliches Supersampling (viele Render-Durchgänge)



- Verwischen gemäß Bewegungsvektorfeld (Post-Effect)

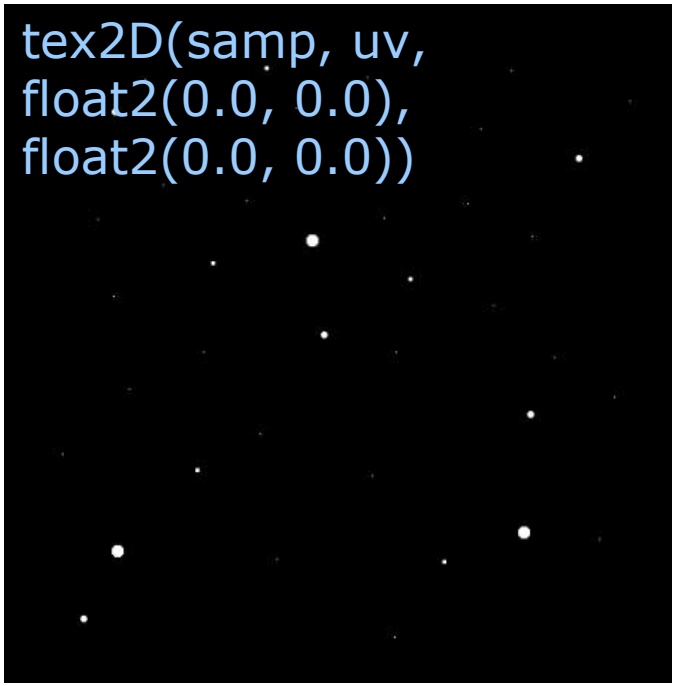


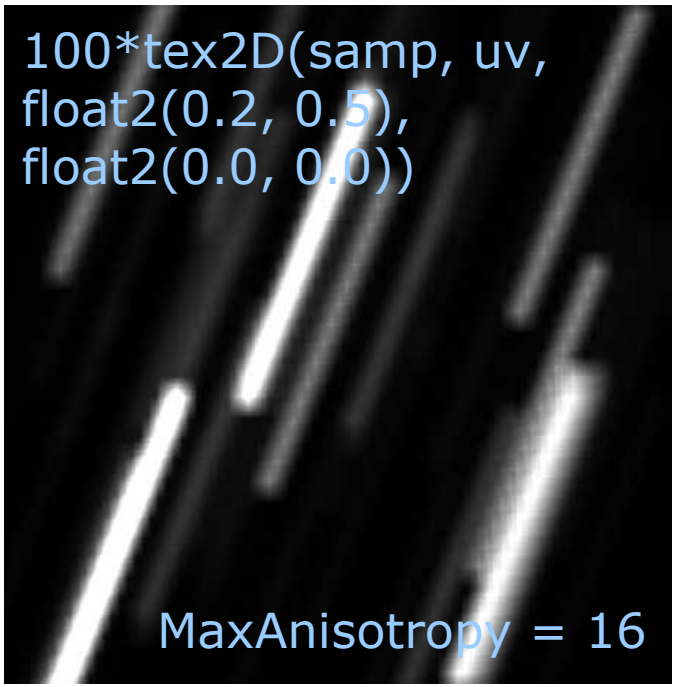
Bewegungsunschärfe: Grundidee

Texturen stark anisotrop filtern

tex2D-Befehl in HLSL kann mit gewünschten Filterprofil gesteuert werden

Demo

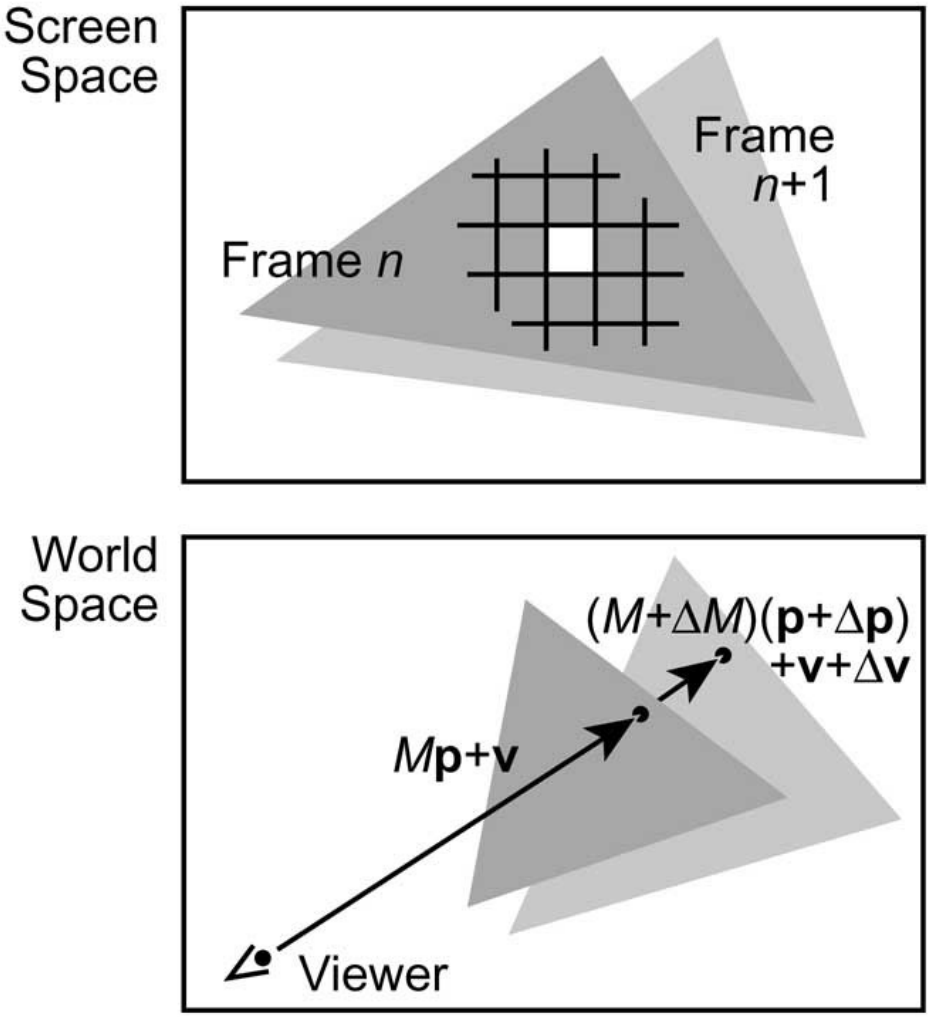




Bewegungsunschärfe: pro-Pixel-Bewegung

Zwei Komponenten:

- euklidisch
- Perspektive



$$\mathbf{s} := \Delta M \mathbf{p} + \Delta \mathbf{v}$$

$$\mathbf{r} := M \mathbf{p} + \mathbf{v}$$

$$\begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} = -UM^{-1}\mathbf{s} + \frac{(M^{-1\mathrm{T}}\mathbf{n}) \cdot \mathbf{s}}{(M^{-1\mathrm{T}}\mathbf{n}) \cdot \mathbf{r}} UM^{-1}\mathbf{r}$$

Bewegungsunschärfe: Raum und Zeit (1)

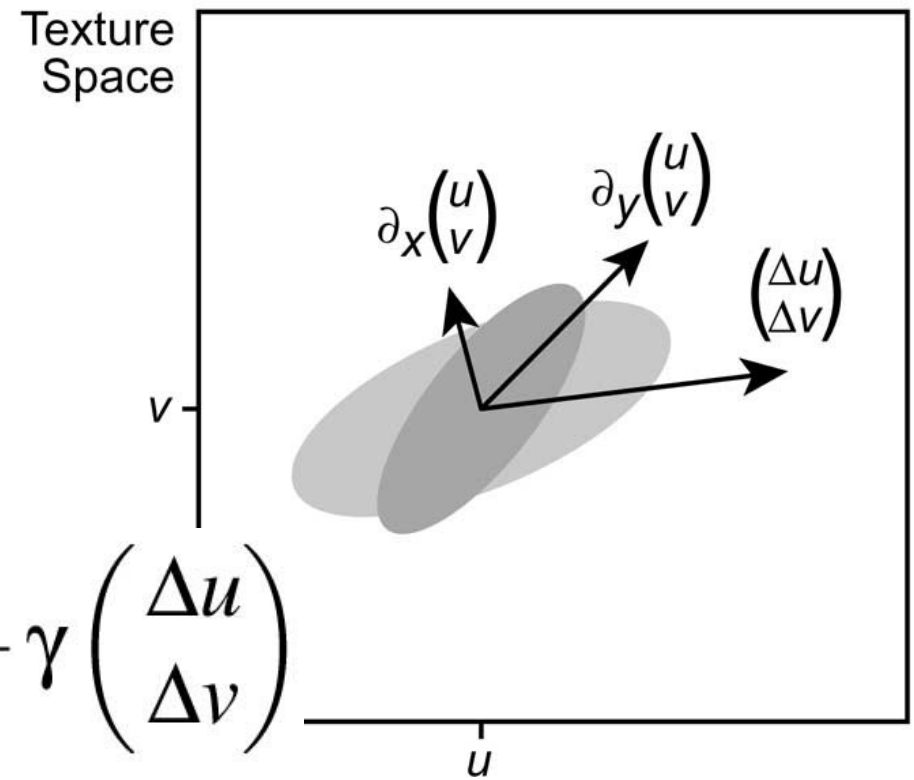
Standard:
automatische räumliche Anisotropie

Vereinigung mit zeitlicher Anisotropie?

Bilde gemeinsames Filterprofil!

tex2D(smp, uv, float2(e, f), float2(g, h))

Bewegungsunschärfe: Raum und Zeit (2)



Modell:
α, β, γ gaußsche Zufallsgrößen

$$\begin{pmatrix} u \\ v \end{pmatrix} + \alpha \partial_x \begin{pmatrix} u \\ v \end{pmatrix} + \beta \partial_y \begin{pmatrix} u \\ v \end{pmatrix} + \gamma \begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix}$$

Für tex2D in *e*, *f*, *g*, *h* umwandeln:

$$\begin{pmatrix} u \\ v \end{pmatrix} + \alpha \begin{pmatrix} e \\ f \end{pmatrix} + \beta \begin{pmatrix} g \\ h \end{pmatrix}$$

Bewegungsunschärfe: Implementierungen (1)

Pixel-Shader-lastige Variante:

- ok für große Polygone
- 30 Befehle im Pixelshader

$$\begin{pmatrix} \Delta u \\ \Delta v \end{pmatrix} = -UM^{-1}\mathbf{s} + \frac{(M^{-1\mathrm{T}}\mathbf{n}) \cdot \mathbf{s}}{(M^{-1\mathrm{T}}\mathbf{n}) \cdot \mathbf{r}} UM^{-1}\mathbf{r}$$

Bewegungsunschärfe: Implementierungen (2)

Vertex-Shader-lastige Variante:

- kleine Polygone nötig
- minimaler Pixelshader
- Wie das Filterprofil zwischen Vertizes interpolieren?
- Keine ddx/ddy-Befehle für Vertex

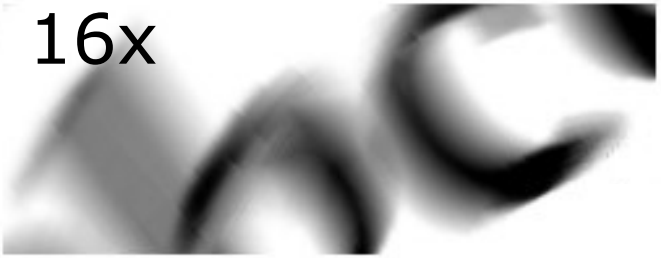
Bewegungsunschärfe: Resultate (1)

Form der Artefakte

Anisotrope Filt.

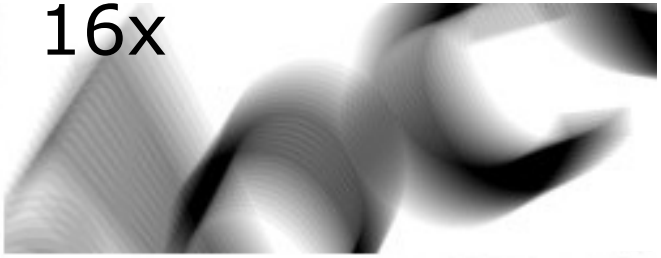
8x

16x



Mehrf. Rendern

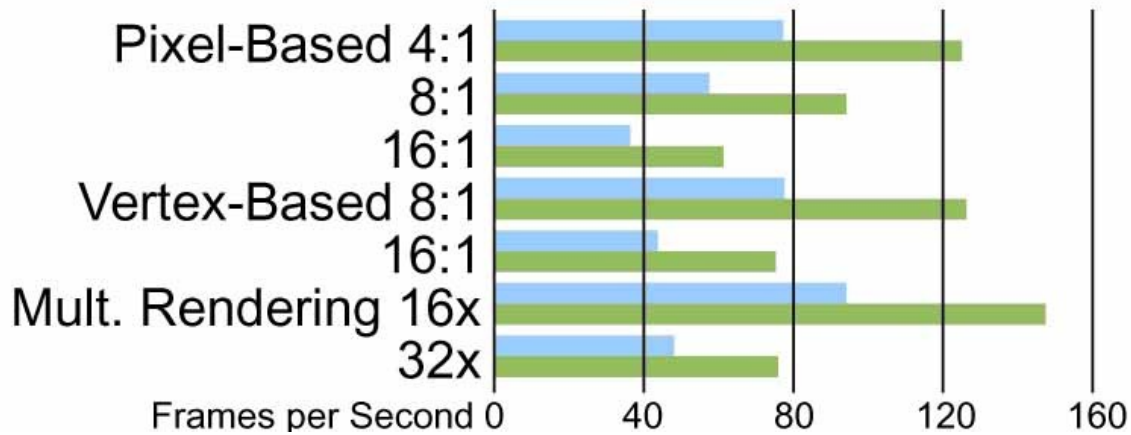
16x



32x

Bewegungsunschärfe: Resultate (2)

Geschwindigkeit vergleichbar mit einer Minimalvariante des mehrf. Renderns



Flug über Terrain
20.000 Dreiecke

Rotierende Kugel
528 Dreiecke

Zukünftig weiter für anisotrope Filterung optimierte HW?

Bewegungsunschärfe: weitere Infos

Eurographics Symposium on Rendering 2005

Stofffalten

Stofffalten

Problem:
Physikalische Stoffsimulationen sind

- rechenaufwendig
- nicht immer robust
- schwer zu steuern

Kinematische Lösung:
falte Oberfläche längen-
oder flächentreu

Stofffalten

Demo

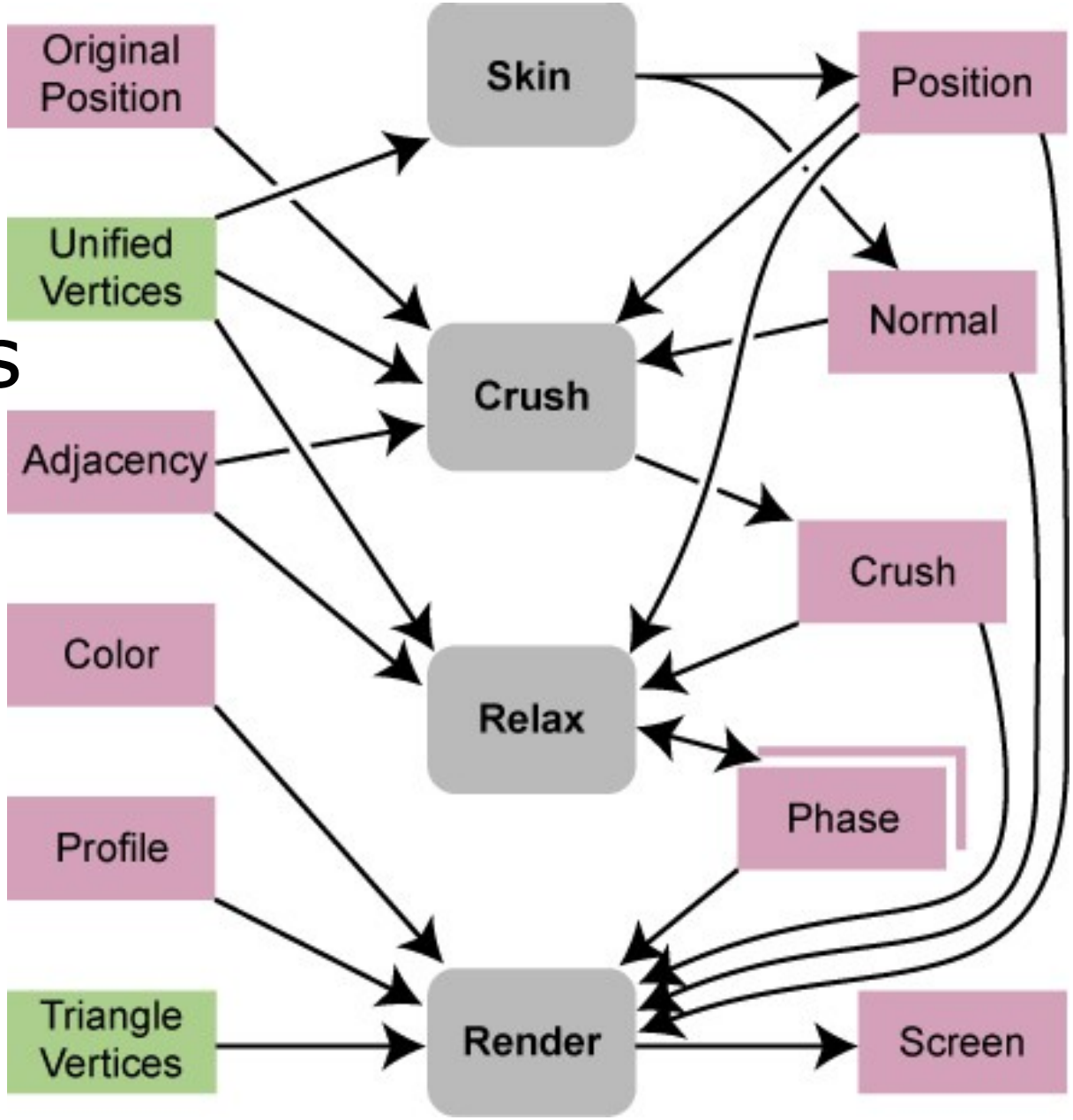
- sehr schnell (300...2000 fps)
- Wellenlänge änderbar
- Profil änderbar
- Falten für Ruhepose malbar

Unterschiede zu verwandten Arbeiten:

- vollständig auf GPU,
  aber keine vorgefertigen Falten
- Steuermöglichkeiten

Stofffalten: Prinzip

1. Führe Animation aus
2. Bestimme ebene Wellen
3. Gleiche die Phasen an
4. Rendere



Stofffalten:
Bestimmung der Stauchung

Finde Matrixnäherung der Verformung um jeden Vertex.

Bestimme Stärke *r* und Richtung **k** der stärksten Verkürzung.
Amplitude der Falten:

$$h := \frac{W}{4}\sqrt{\max(1 - r^2, 0)}$$

Stofffalten:
Ebene Wellen

Bilde um jeden Vertex eine entsprechende ebene Welle:

$$h(\mathbf{x}) = h_A \cos\left(\frac{2\pi}{W}\mathbf{k}_A \cdot (\mathbf{x} - \mathbf{x}_A) + \phi_A\right)$$

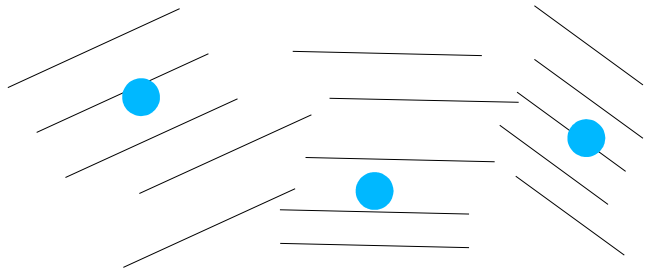
Werte das mit linearer Interpolation im Post-Deformations-Raum aus:

$$\begin{aligned} h(\mathbf{x}') \;=\; & \alpha h_A \cos\left(\frac{2\pi}{W}\mathbf{k}'_A \cdot (\mathbf{x}' - \mathbf{x}'_A) + \phi_A\right) \\ & + \beta h_B \cos\left(\frac{2\pi}{W}\mathbf{k}'_B \cdot (\mathbf{x}' - \mathbf{x}'_B) + \phi_B\right) \\ & + \gamma h_C \cos\left(\frac{2\pi}{W}\mathbf{k}'_C \cdot (\mathbf{x}' - \mathbf{x}'_C) + \phi_C\right) \end{aligned}$$

Stofffalten: Phasenangleichung

$$h(\mathbf{x}) = h_A \cos\left(\frac{2\pi}{W}\mathbf{k}_A \cdot (\mathbf{x} - \mathbf{x}_A) + \phi_A\right)$$

Phase ist noch unbestimmt,



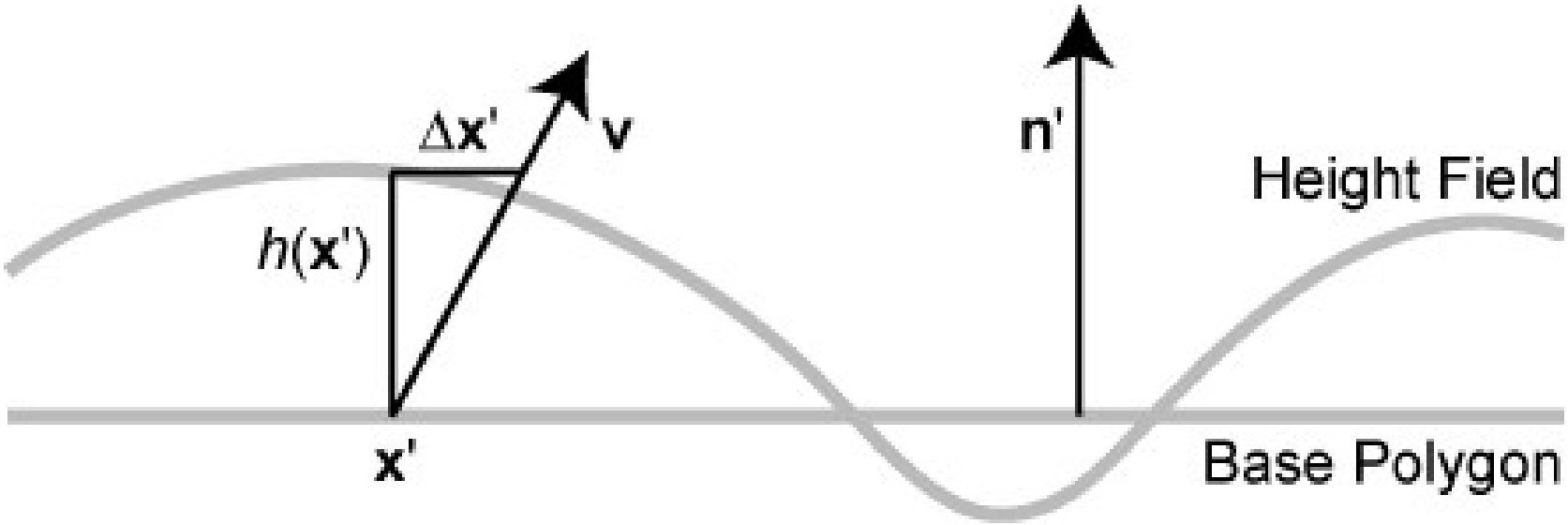
Angleichung nötig!

Fehlermaß für lokalen Versatz, Gradientenabstieg parallel zur Animation (nicht kinematisch!)

Demo

Stofffalten: Texturverformung

Prinzip wie Parallax Mapping

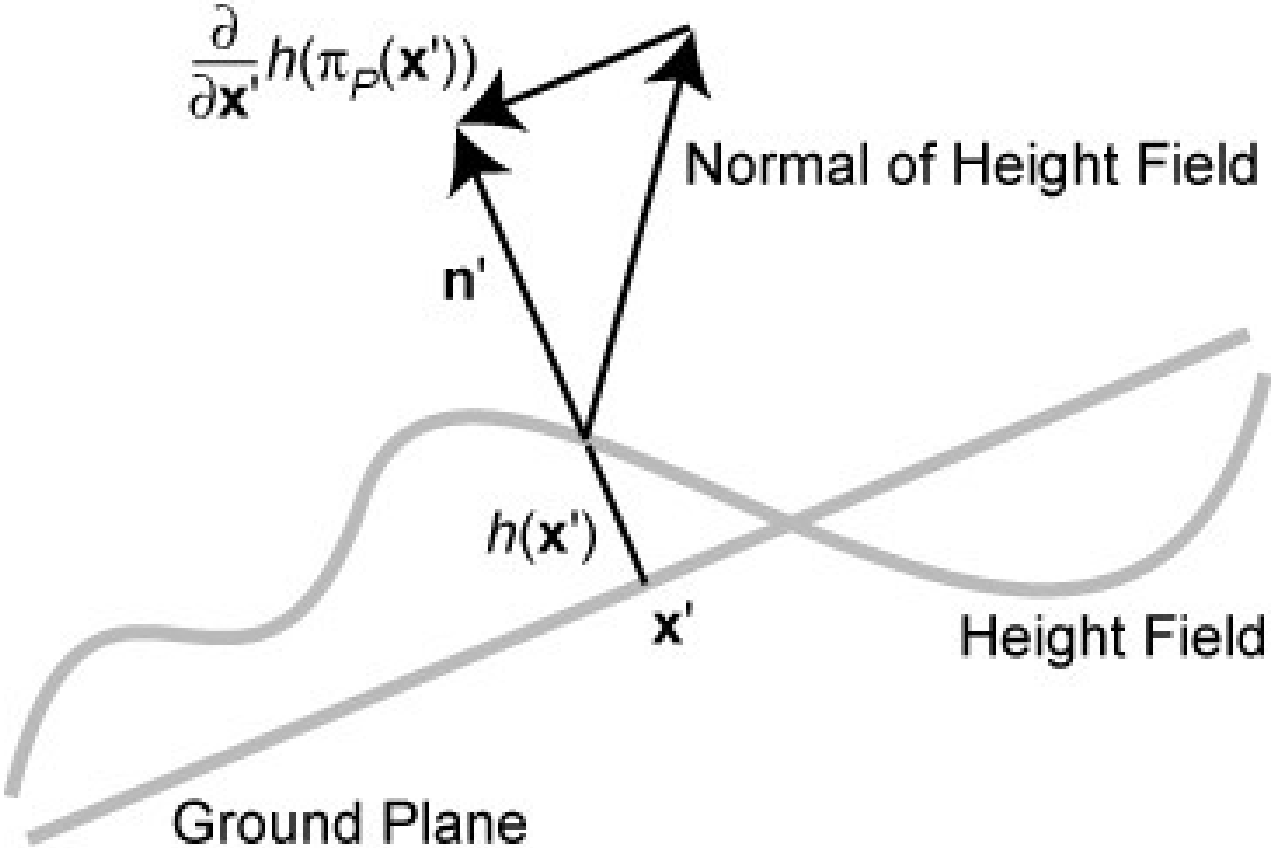


$$\Delta\mathbf{x}' = \left(\frac{\mathbf{v}}{\mathbf{v}\cdot\mathbf{n}'} - \mathbf{n}'\right)h(\mathbf{x}')$$

Stofffalten: Beleuchtung

Bump Map mit dynamischem Höhenfeld

Gradient von *h* benötigt!



Stofffalten: Details

Profil: 1D-Textur mit einer Funktion und deren Ableitung

$$h(\mathbf{x}) = h_A \cos\left(\frac{2\pi}{W}\mathbf{k}_A \cdot (\mathbf{x} - \mathbf{x}_A) + \phi_A\right)$$

Falten für Ruhepose:

Störe die Matrix M, die die lokale Verformung approximiert:

Statt M nehme M(E–**q**⊗**q**)
mit Vektor **q** für Richtung und Stärke.

Stofffalten: Geschwindigkeit

Name	# Vertices	# Pixels (average)	fps
Shirt	455	≈ 330.000	328
Zeppelin	508	≈ 260.000	540
Curtain	92	≈ 505.000	1961

Stage	# Shader instructions		Contribution	
	Vertex	Pixel	A	B
Skin	$13M+20$	2	0.06 ms	0.25 ms
Crush	8	$28N+102$	0.08 ms	1.83 ms
Relax	7	$33N+23$	0.11 ms	1.75 ms
Render	67	47	3.11 ms	5.01 ms
Total time incl. non-shader part			3.45 ms	8.90 ms

Ausblick

Ausblick

- Immer mehr Shader-Einheiten
- Gleichberechtigte Shader-Einheiten, die nach Bedarf als Vertex- oder Pixel-Shader auftreten
- Freierer Zugriff auf Buffer
- Stream Output aus Vertex-Shader
- Geometry-Shader

Vielen Dank!

Fragen?