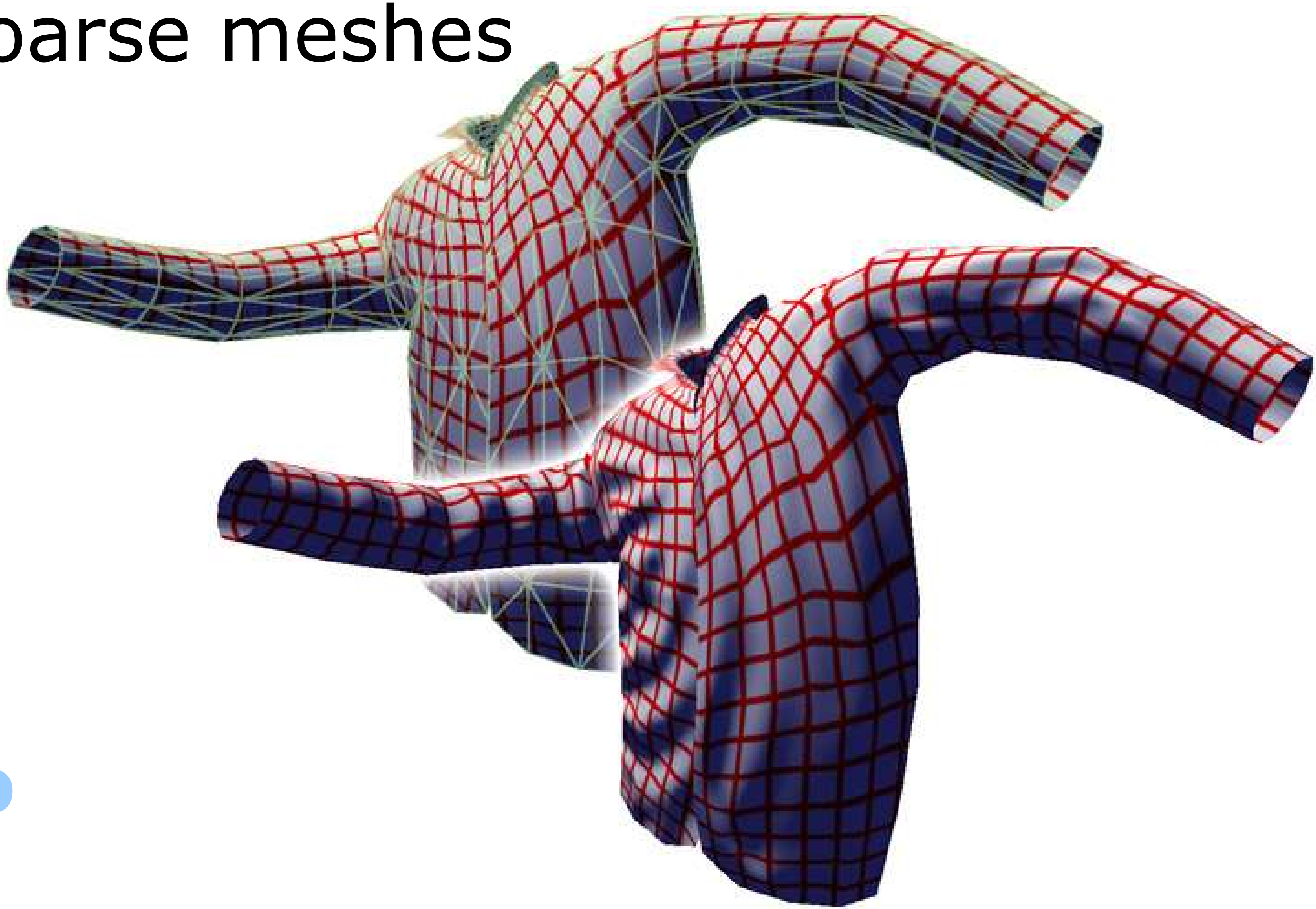
Dynamic Wrinkle Patterns on Animated Meshes

& some other current topics

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

Objective

- Plausible wrinkles and folds
- On coarse meshes
- Fast



Demo

Outline

- Approaches to Cloth Simulation
- Overview of the Method
- Preparing the Mesh
- Deforming the Mesh
- Determining the Fold Vector Field
- Generating the Height Field
- Rendering
- Results
- Outlook

Approaches to Cloth Simulation

Full Physical Dynamics

- ↗ Stunning results if done right
- ↘ High-dimensional stiff PDEs
- ↘ Non-robust collision detection

Kinematics

= Shape depends only on deformation

- ↗ Fast and plausible if done right
- ↘ Large folds difficult
- ↘ No temporal variation

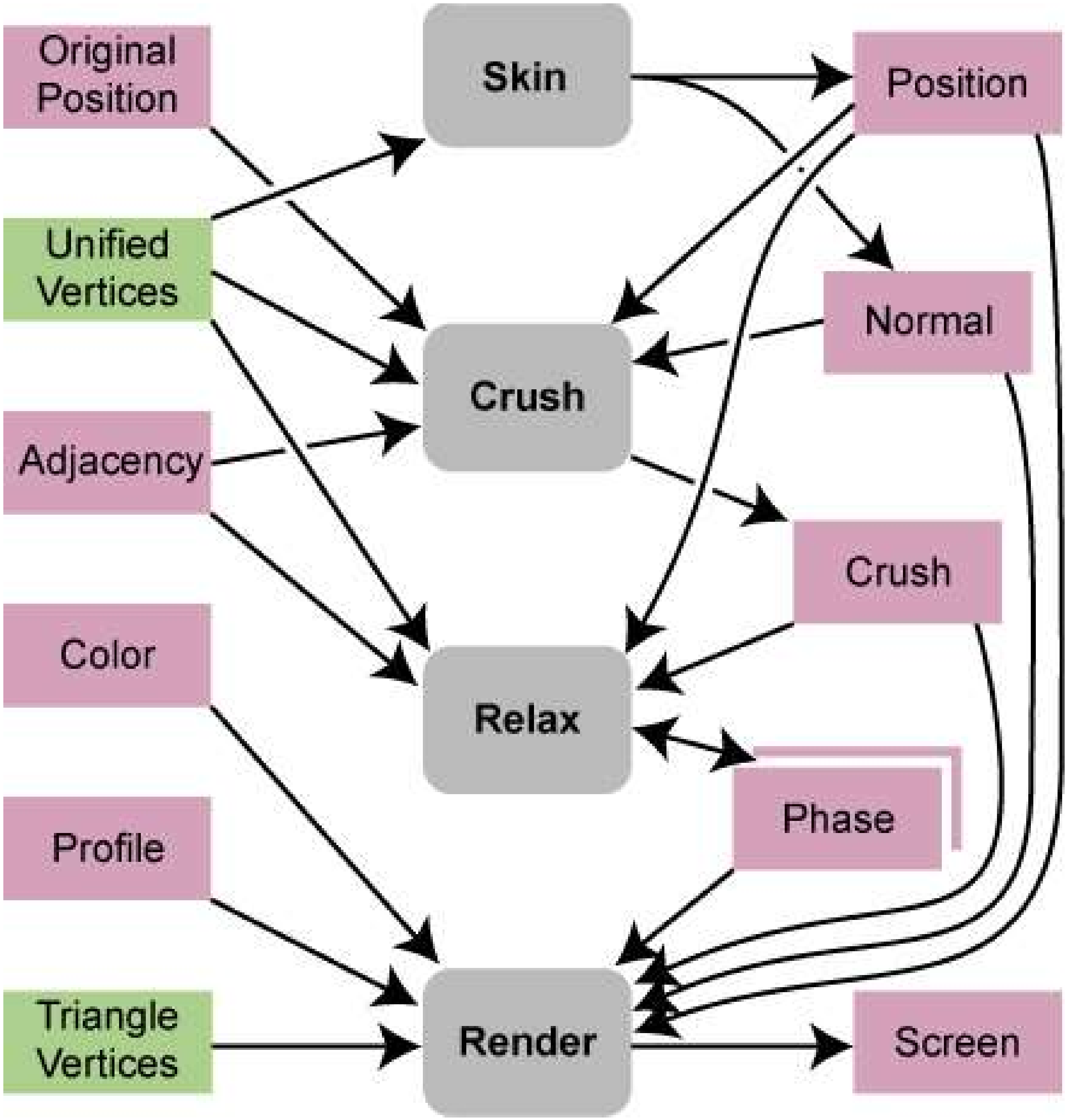
Overview of the Method

- Input coarsely tessellated surface
- Deform using standard methods, e.g., matrix palette skinning
- Determine per vertex: strength and direction of local contraction
- Compute oscillating height field
- Render through pixel shader: lighting, texture deformation

Demo

Overview of the Method

All done in four rendering passes (= pairs of vertex and pixel shaders)



Preparing the Mesh

- Collect adjacency data in a pseudo-texture

- Unify vertices along texture seams
  Demo

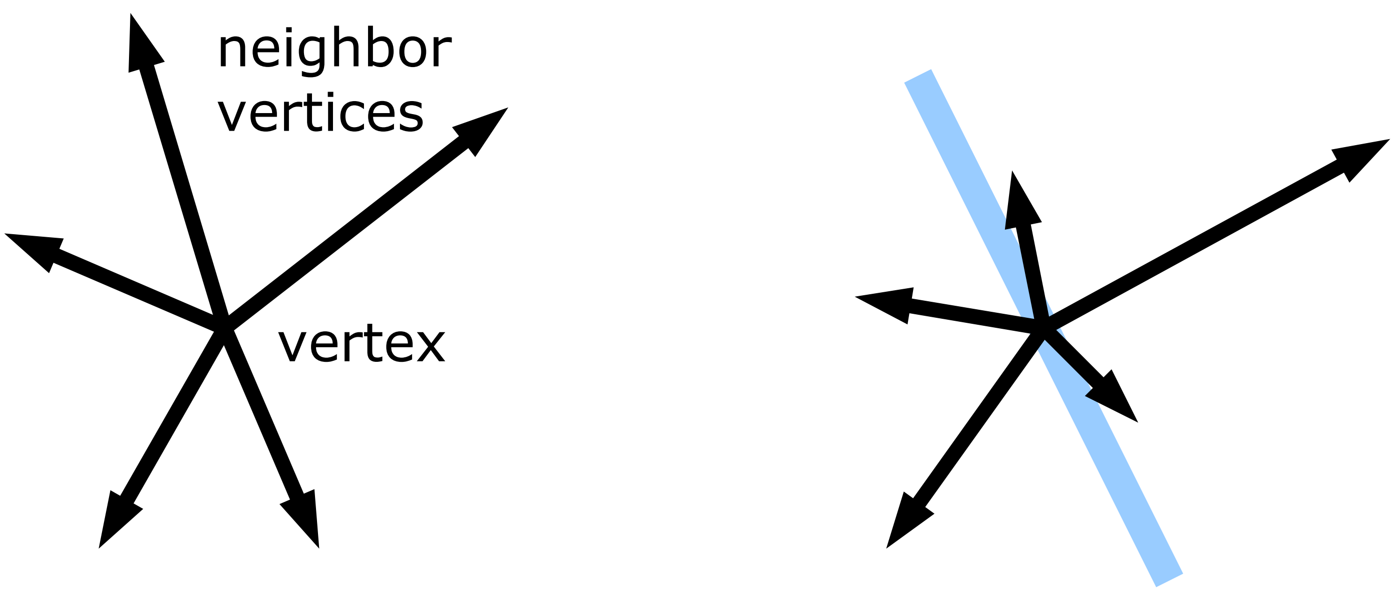
Deforming the Mesh

- Standard matrix palette skinning
- .x file from standard 3D software: mesh, skeleton, bone weights, bone animation
- Matrix palette prepared by CPU
- Vertex Shader evaluates weighted sum
- Positions and normals stored in pseudo-textures for later use

Determining the Fold Vector Field

For every vertex:

- Linear approx. M of local deformation
- Find direction and amount of strongest contraction, eigenanalysis of $M^T M$



before deformation   after deformation

Determining the Fold Vector Field

How to produce folds for the rest pose?

Bias the computation of the linear approximation M:
$M \rightarrow M(E - \mathbf{q} \otimes \mathbf{q})$,
where $\mathbf{q}$ gives direction and amount.

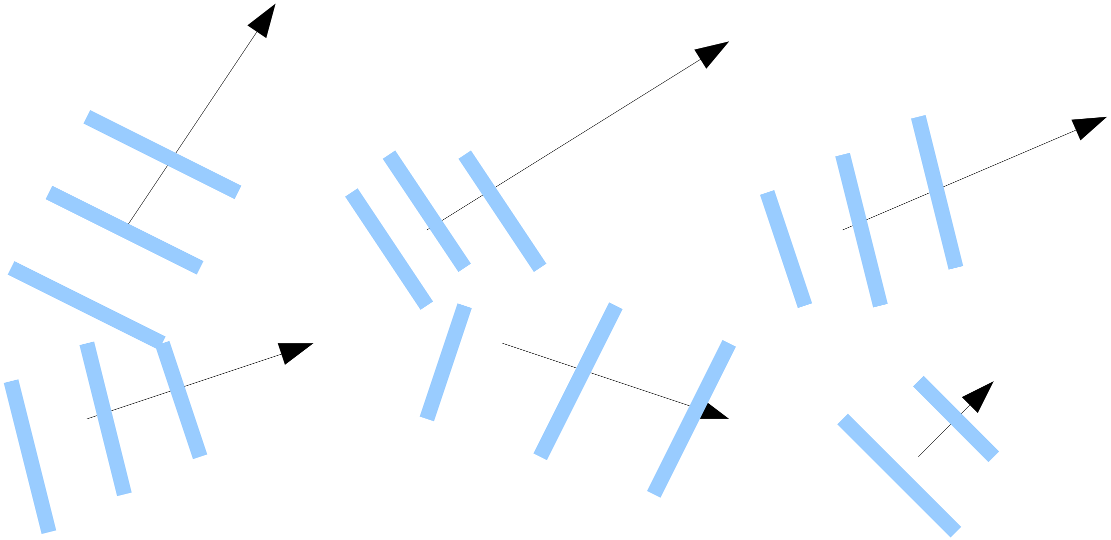
User interface: 3D painting.

Demo

Determining the Fold Vector Field

Result:

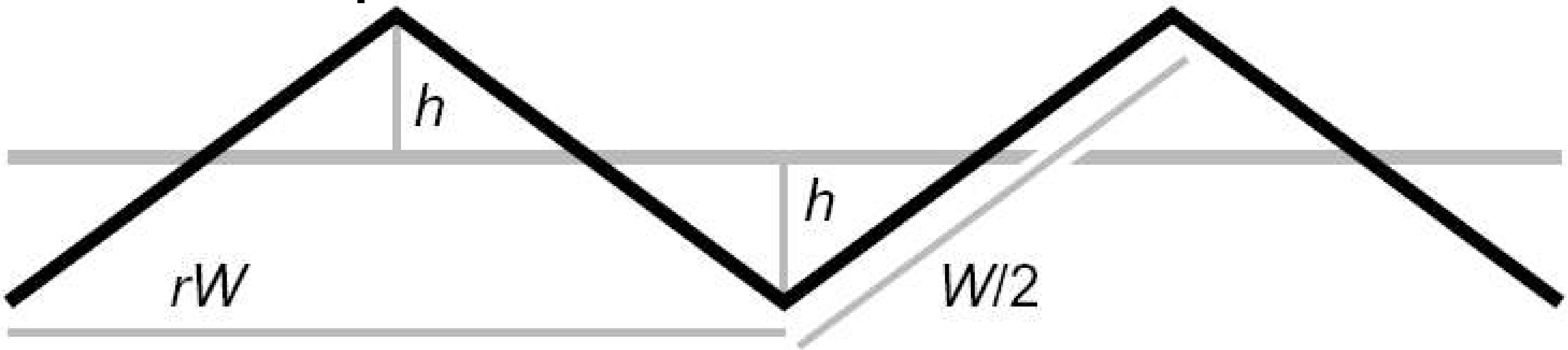
A tangent vector at every vertex,
direction = strongest contraction,
length = amount of folding



Local plane waves!

Generating the Height Field

Height h of fold
depends on width
of uncompressed fold



Real-time control on W

Demo

Generating the Height Field

Form a plane wave around every vertex:

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

Evaluate h with linear interpolation in post-deform space:

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

Generating the Height Field

Problem: The phases of the local plane waves are not yet determined.

Solution: Relax the phases to diminish local misfit.

This is no longer pure kinematics!

Demo

Rendering

Want to render coarse polygons and fake folds simply with coloring

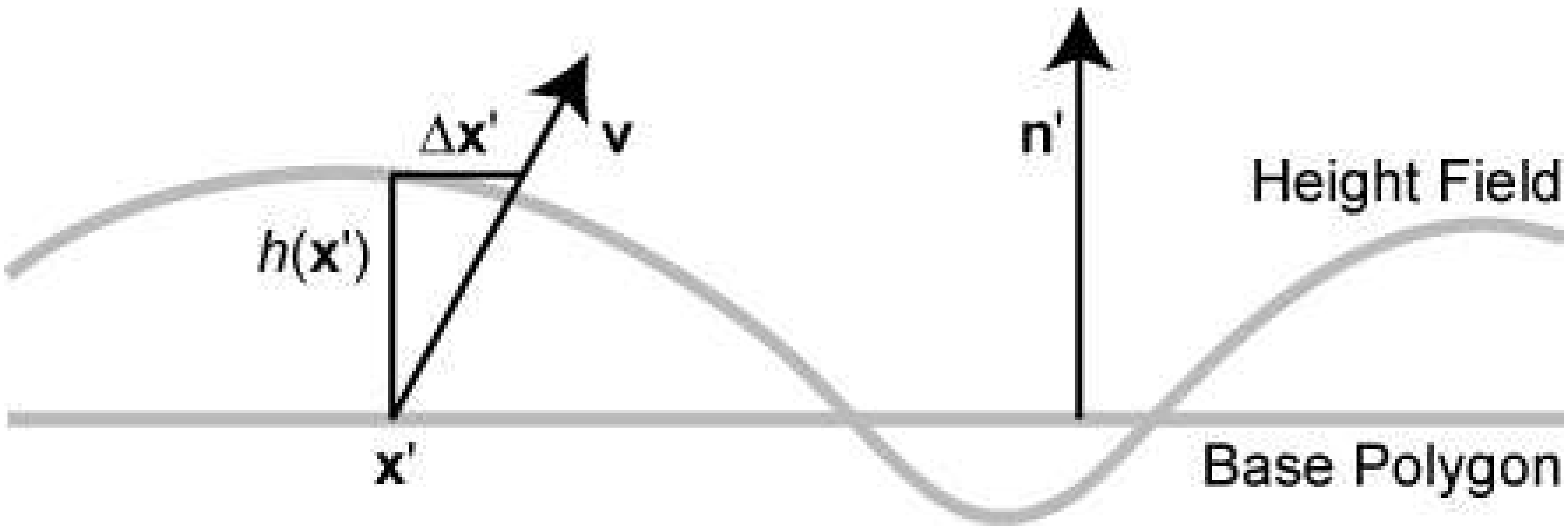
Two issues to address:

- Texture. Deform the texture as though there was curvature.
- Lighting. Adjust the normal vector per pixel.

Demo

Rendering: Texture Deformation

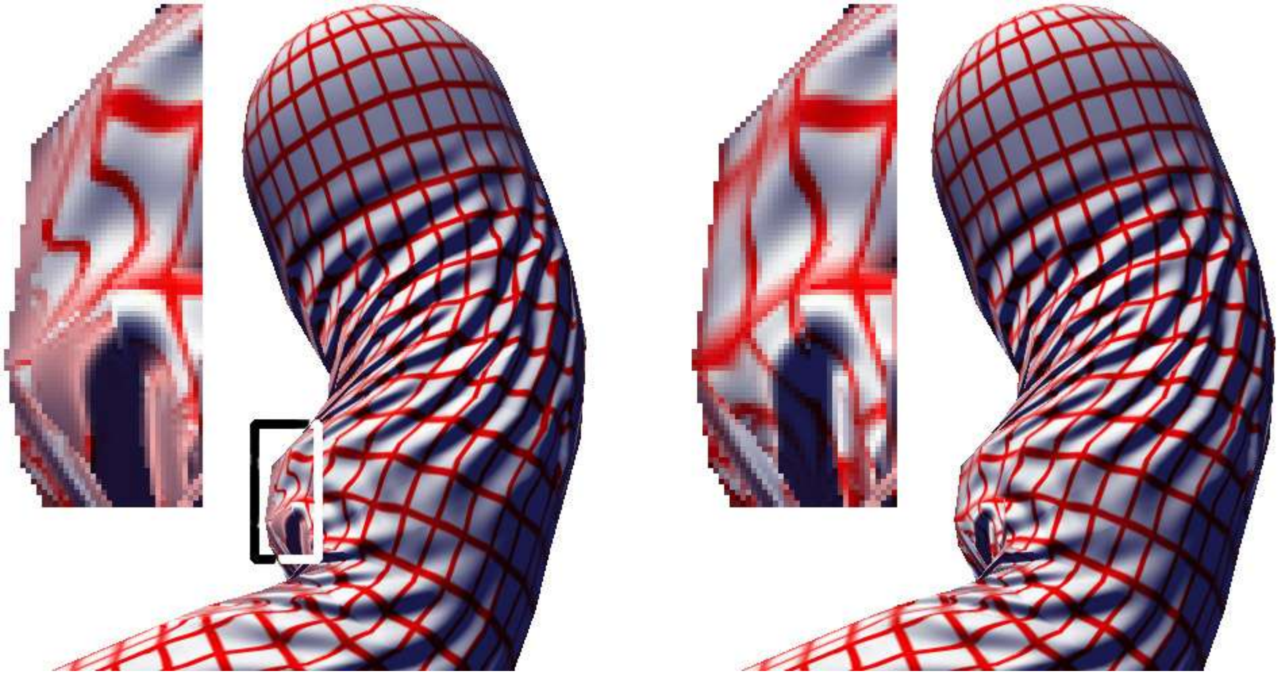
Principle similar to Parallax Mapping



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

Rendering: Texture Deformation

Problem: bad artifacts
in particular near the horizon

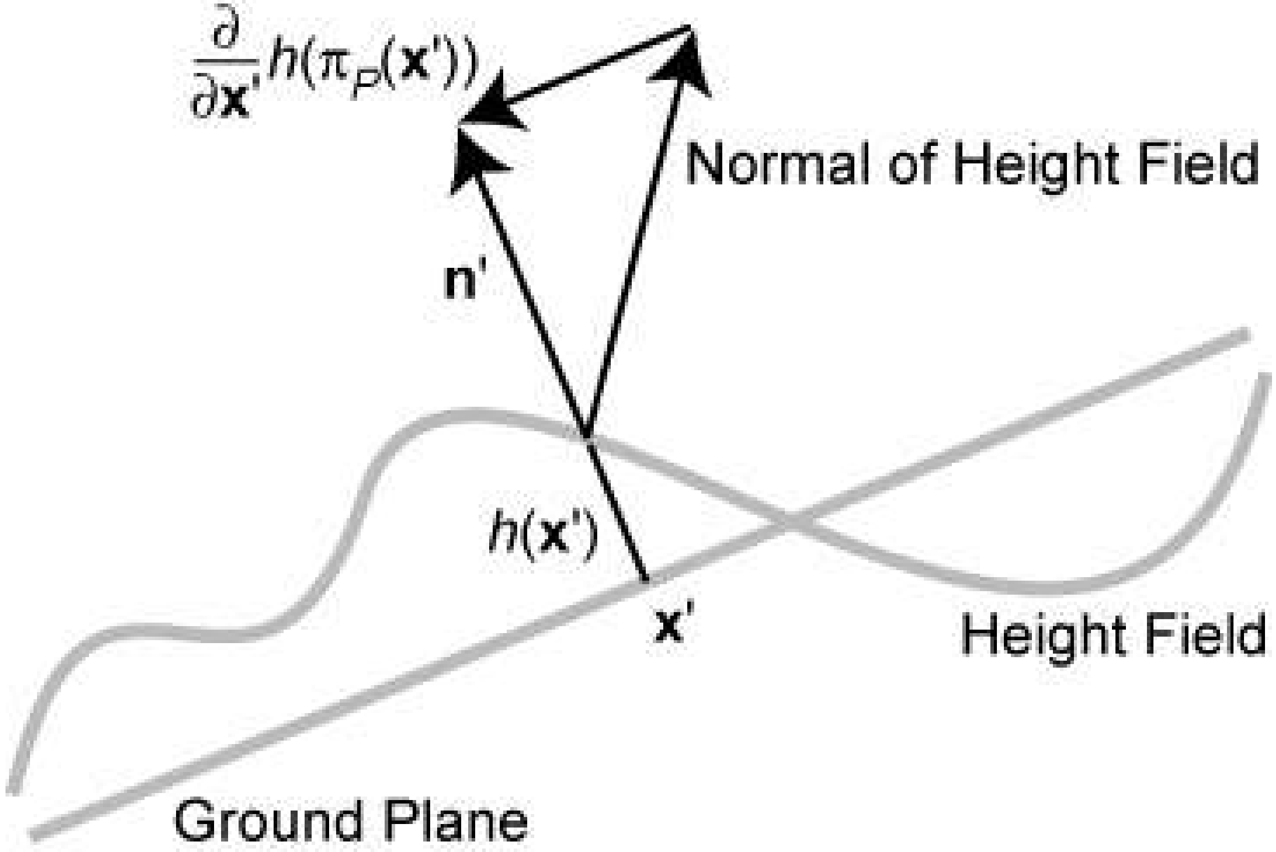


Solution (partially):
Limit the deformation

Rendering: Illumination

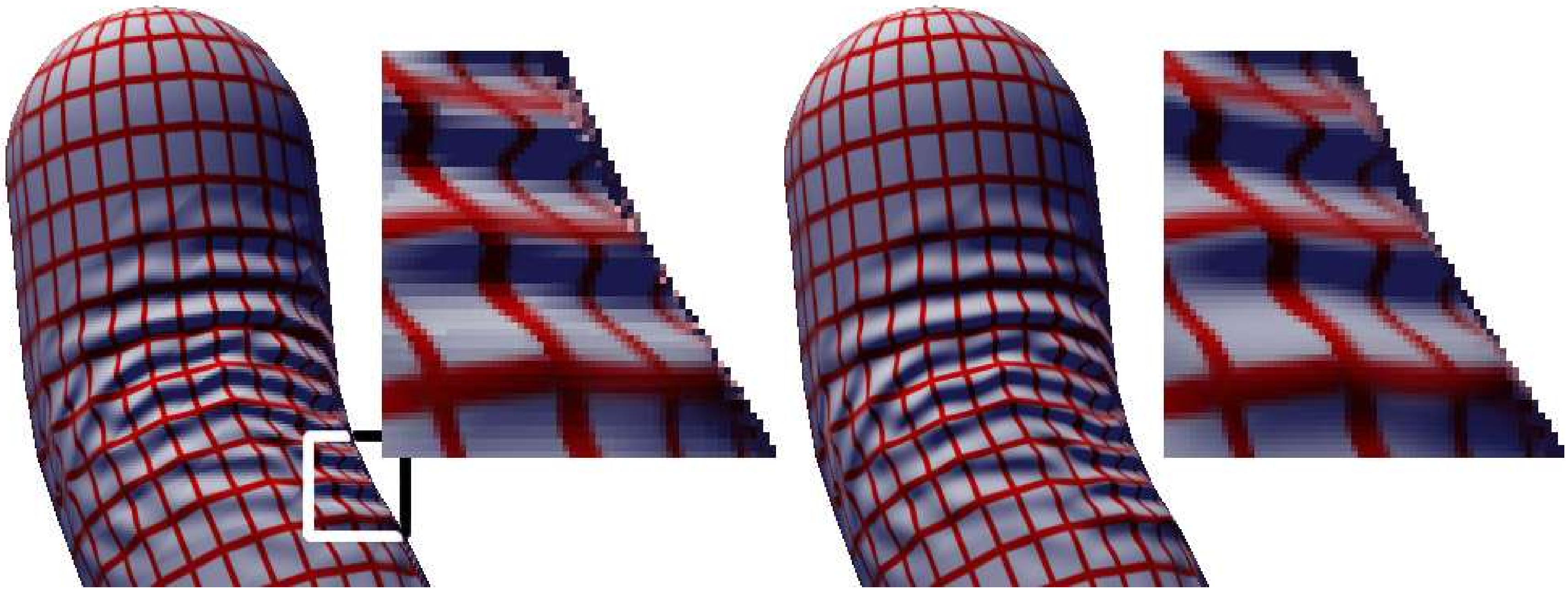
Bump map with dynamic (!) height field

Need the gradient of *h*.



Rendering: Illumination

Built-in gradient command of HLSL produces blocks of 2x2 pixels.



Thus: Compute the gradient from the wave's equation.

Rendering:
Arbitrary Fold Profiles

- height: cosine
- gradient: sine

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

Replace each with a 1D texture lookup: arbitrary profile.

Has to be symmetric, though.
(Fold vector field only determined up to sign!)

Demo

Results

Name	# Vertices	# Pixels (average)	fps
Shirt	455	≈ 330.000	328
Zeppelin	508	≈ 260.000	540
Curtain	92	≈ 505.000	537

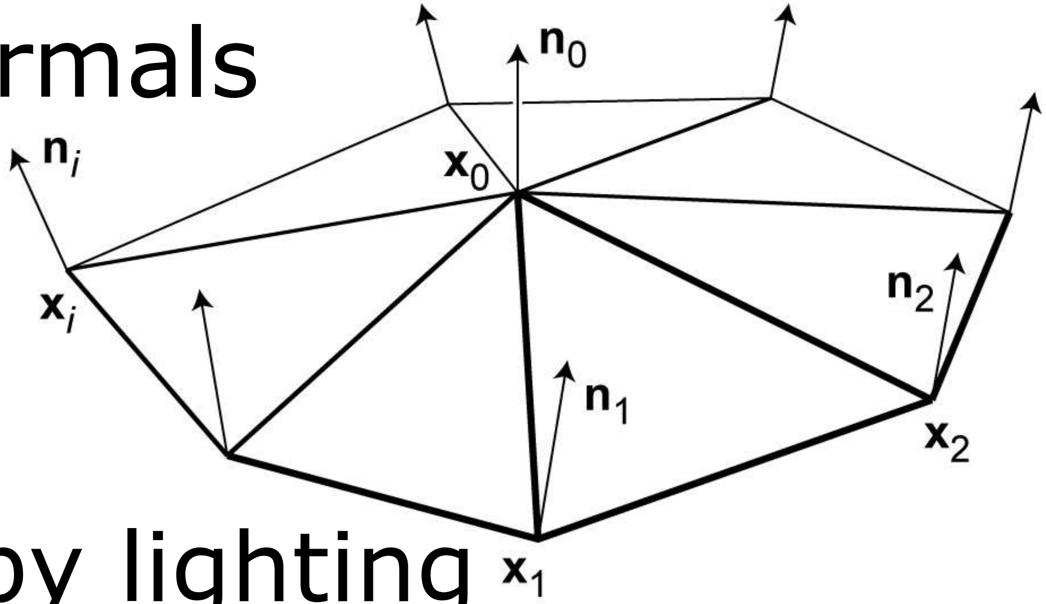
Stage	# Shader instructions		Contribution	
	Vertex	Pixel	A	B
Skin	$13M + 20$	2	0.06 ms	0.25 ms
Crush	8	$28N + 102$	0.08 ms	1.83 ms
Relax	7	$33N + 23$	0.11 ms	1.75 ms
Render	67	47	3.11 ms	5.01 ms
Total time incl. non-shader part			3.45 ms	8.90 ms

M = #Bones; N = #Neighbors

A: 1 Mpix, 100 verts; B: 55 kPix, 50 kVerts

Hatching instead of Wrinkles

- Matrix M: linear model of divergence of the adjacent normals



- Modulate pattern by lighting
- Soften the pattern depending on its on-screen wavelength

Demo

Outlook

- Introduce irregularities, e.g., through additional textures
- Combine with physical dynamics of coarse mesh
- Use geometry shaders (upcoming in DirectX 10) to access neighbor vertices

Some other Current Topics

- Lean rendering of hard-edged textures **Demo**
- Low-cost motion capture with accelerometers **Demo**
- Visual data mining for music files through procedural icons **Demo**

Thanks for your attention!

Questions?