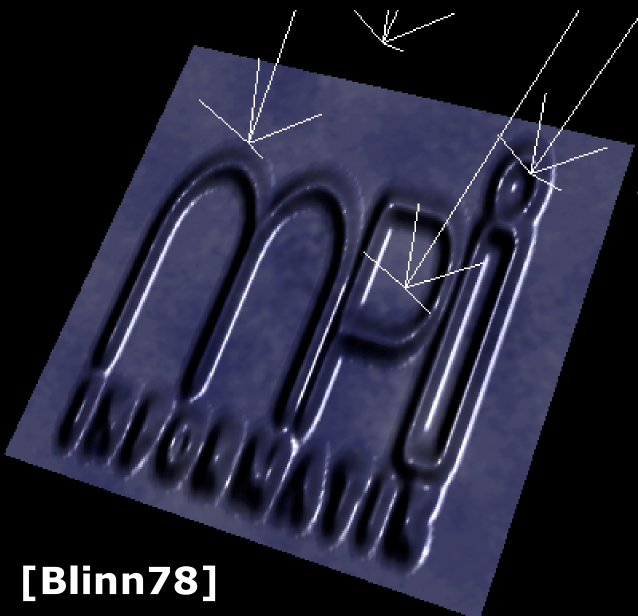
Towards Interactive Bump Mapping with Anisotropic Shift-Variant BRDFs

Jan Kautz
Hans-Peter Seidel

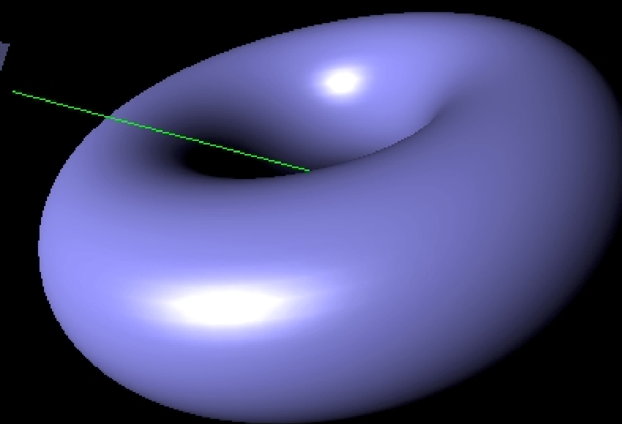
MPI INFORMATIK

Motivation

Incompatible techniques:



[Blinn78]



[Heidrich99]

[Kautz99]

Bump Mapping (Phong model)

Rendering with arbitrary BRDFs

⇨ **Combine bump mapping & shift-variant BRDFs**

Motivation

mpi INFORMATIK

Combine bump mapping & shift-variant BRDFs

Per-pixel:

- Evaluation of BRDF
- BRDF parameters
- Normal and tangent

Useful for e.g.:

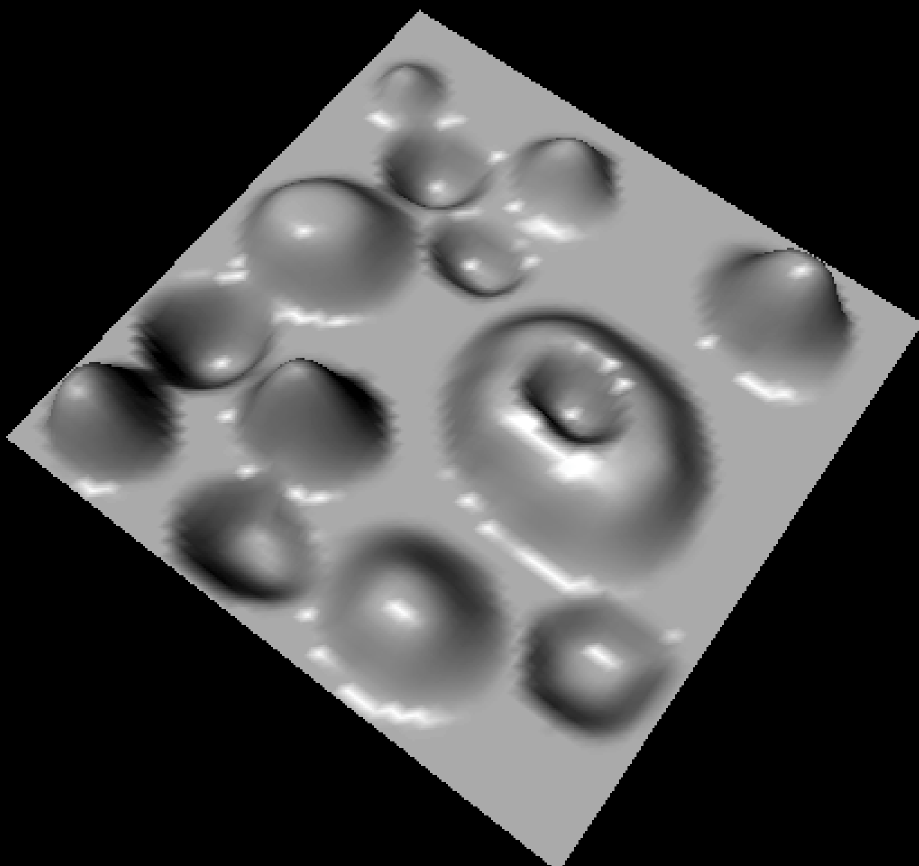
- Human skin
- Corroded metal
- …

Overview

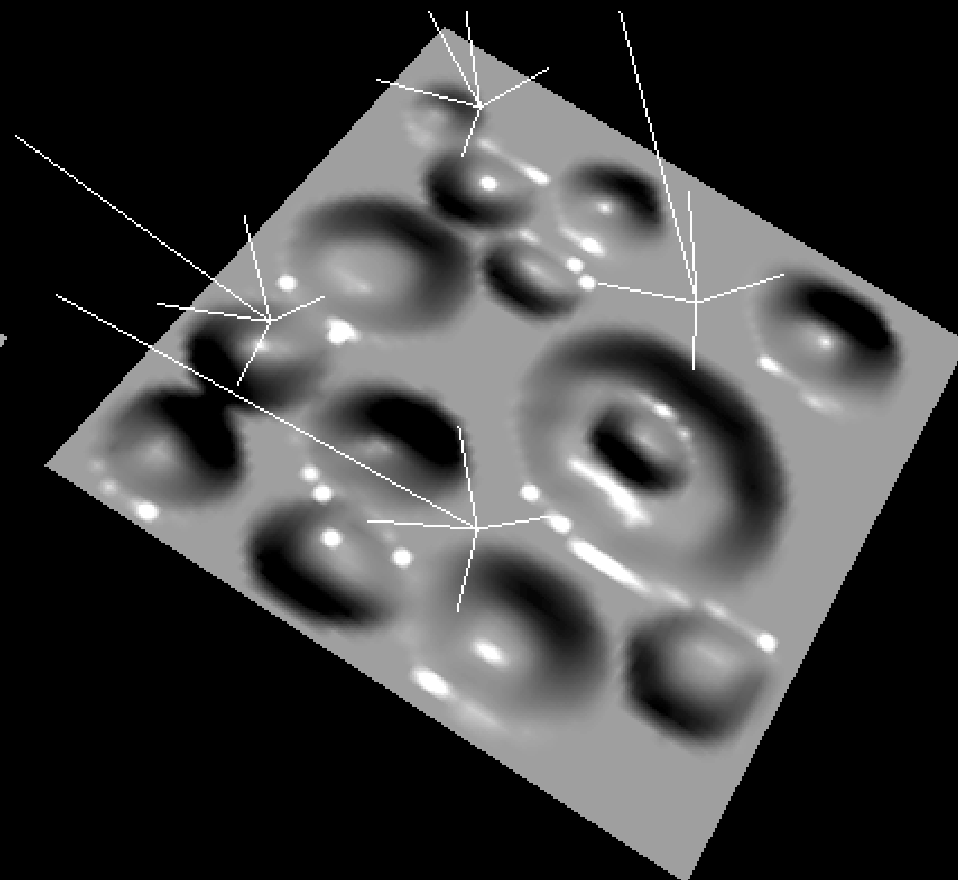
mpi INFORMATIK

- Introduction
  - Bump mapping
  - Reflectance models
  - Hardware capabilities
- Mapping reflectance models to hardware
- Examples
- Results
- Issues
- Conclusion

Introduction – Bump Mapping



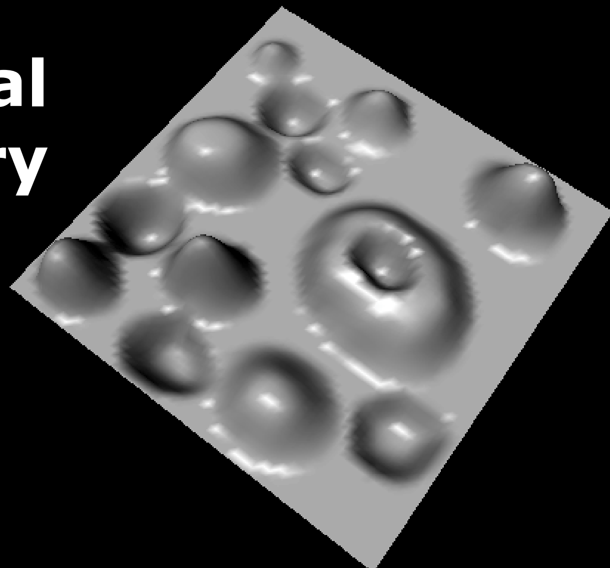
Real Geometry

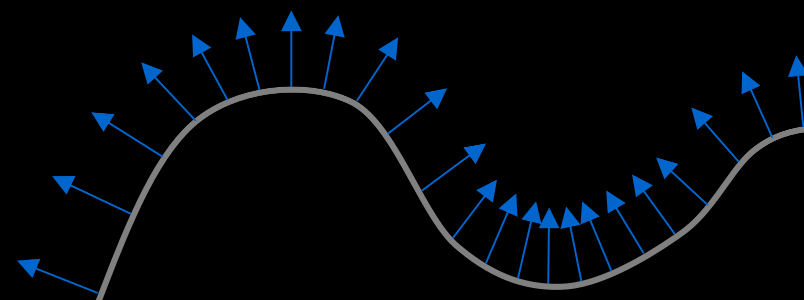


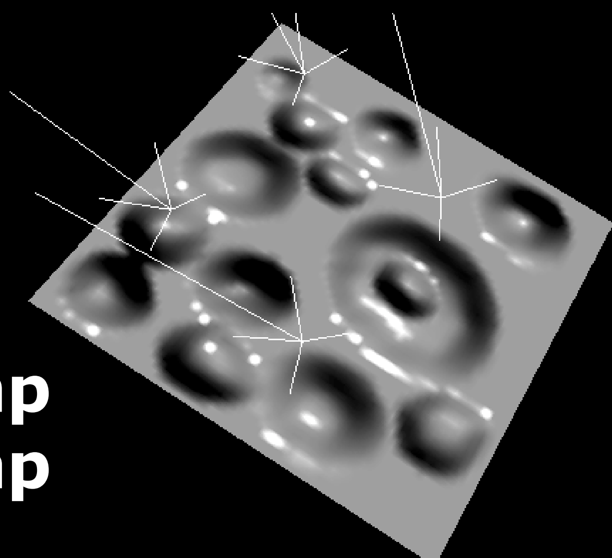
Bump Map

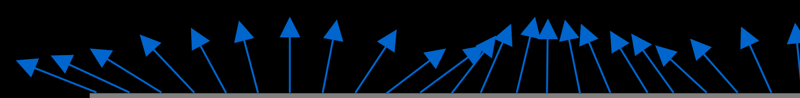
Introduction - Bump Mapping

Real Geometry









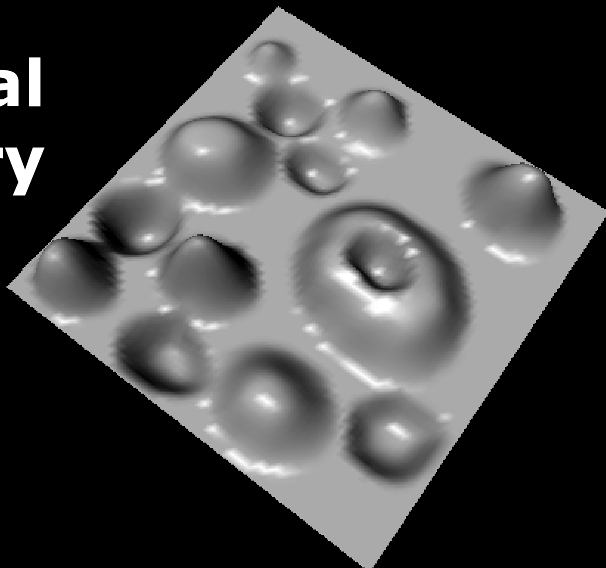
Bump Map

Illumination:

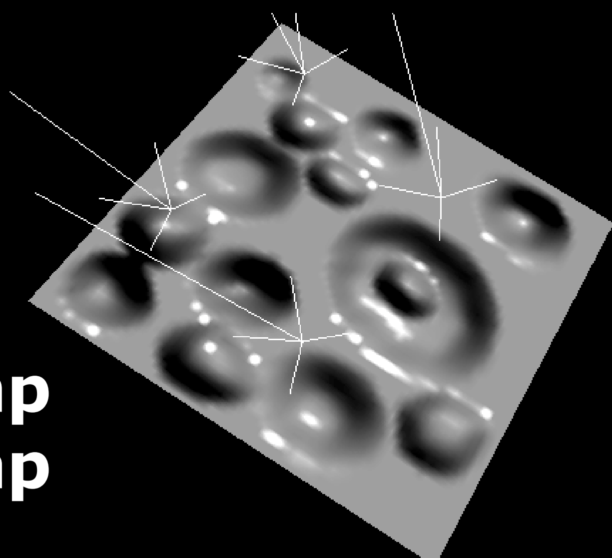
$$L_0 = k_s(\hat{h}\cdot\hat{n})^N + k_d(\hat{n}\cdot\hat{l})$$

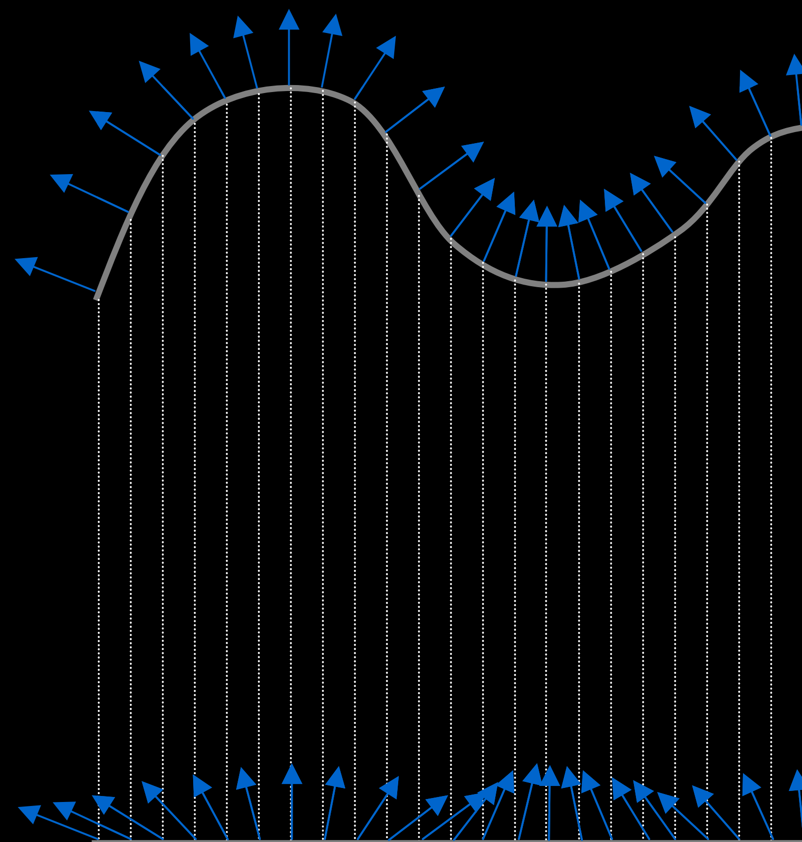
Introduction - Bump Mapping

Real Geometry



Bump Map



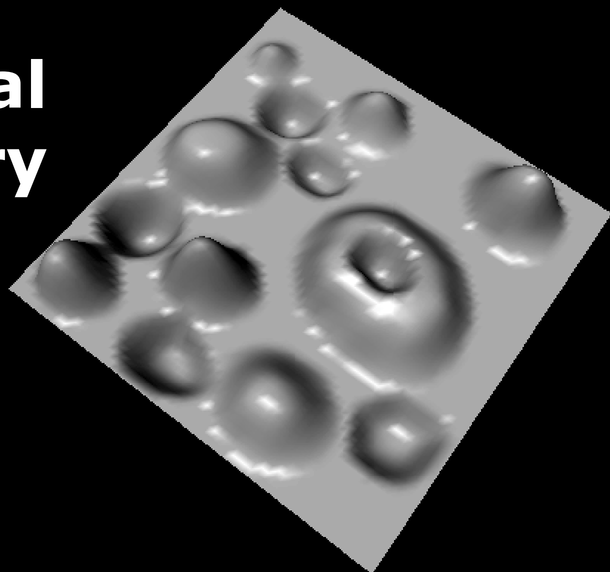


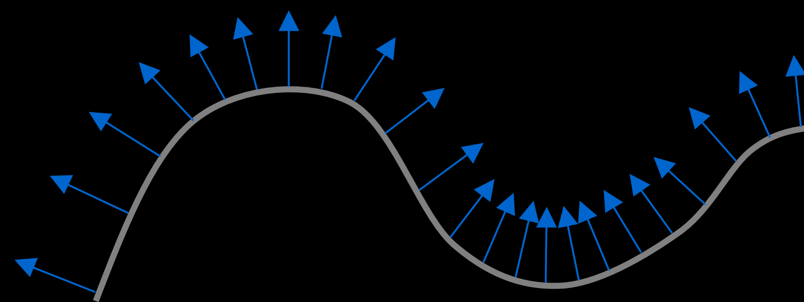
Illumination:

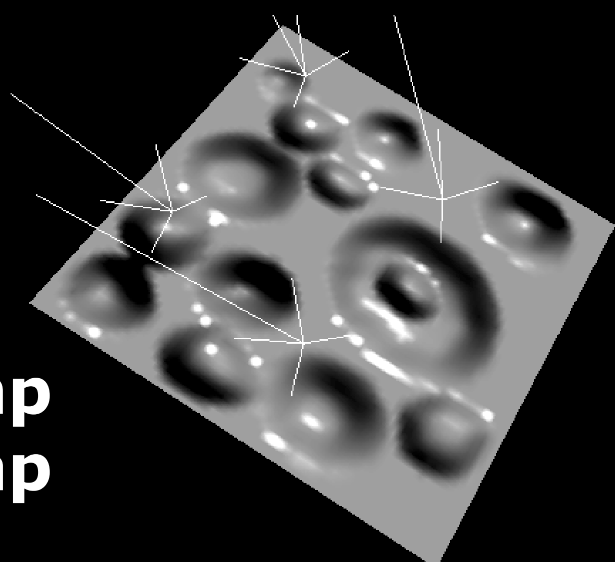
$$L_0 = k_s(\hat{h}\cdot\hat{n})^N + k_d(\hat{n}\cdot\hat{l})$$

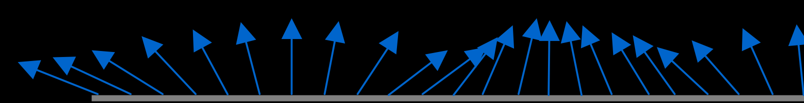
Introduction - Bump Mapping

Real Geometry









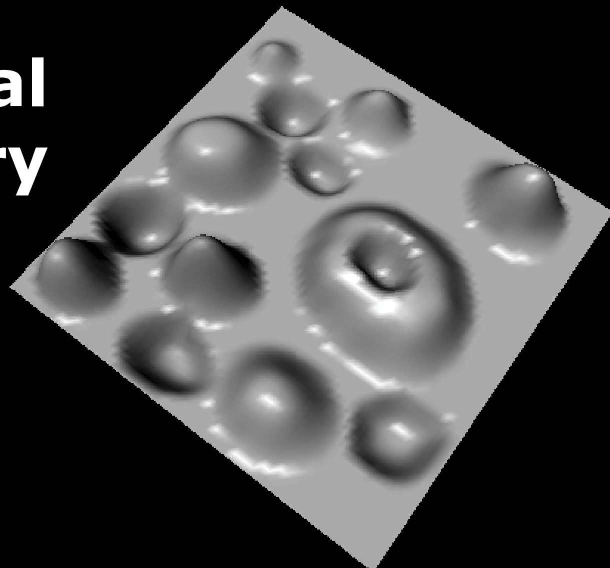
Bump Map

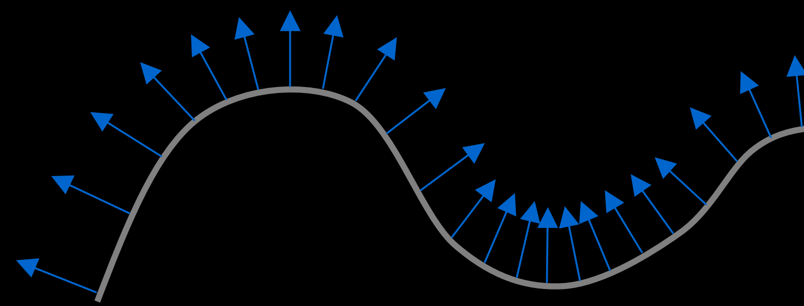
Illumination:

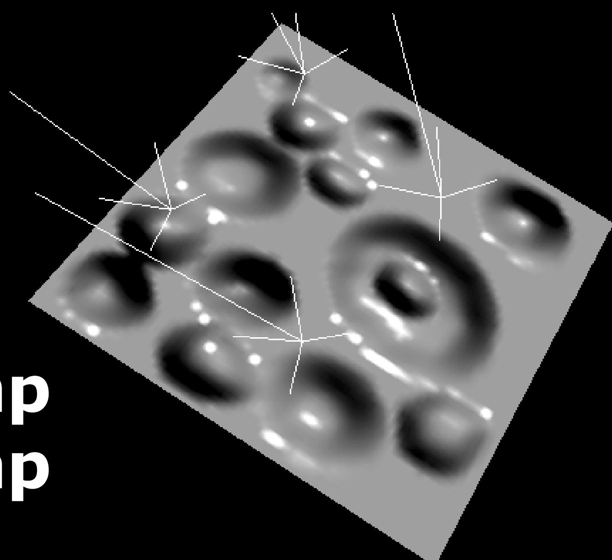
$$L_0 = k_s(\hat{h}\cdot\hat{n})^N + k_d(\hat{n}\cdot\hat{l})$$

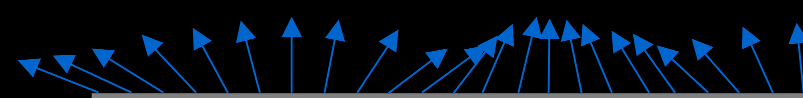
Introduction - Bump Mapping

Real Geometry









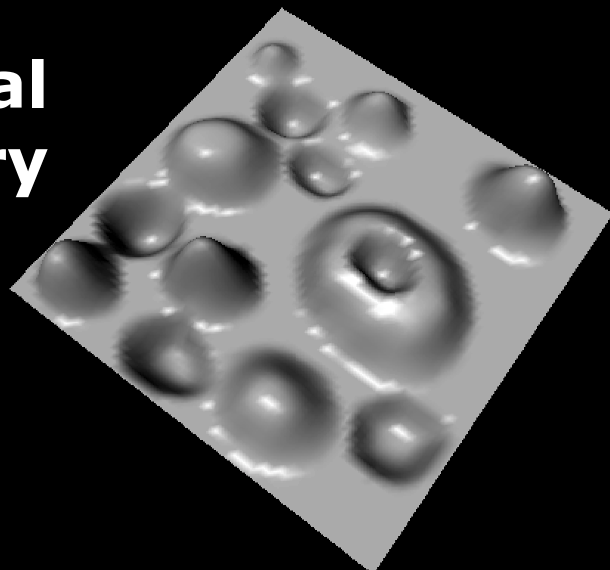
Bump Map

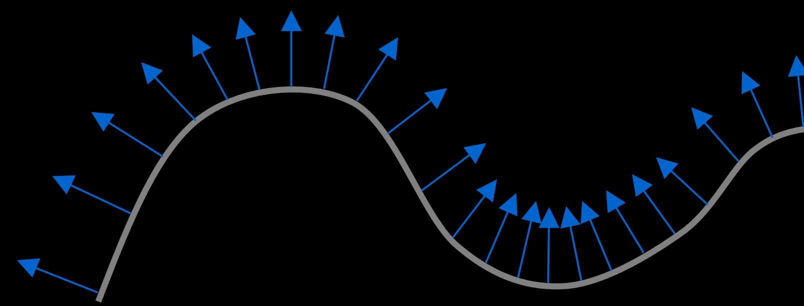
Illumination:

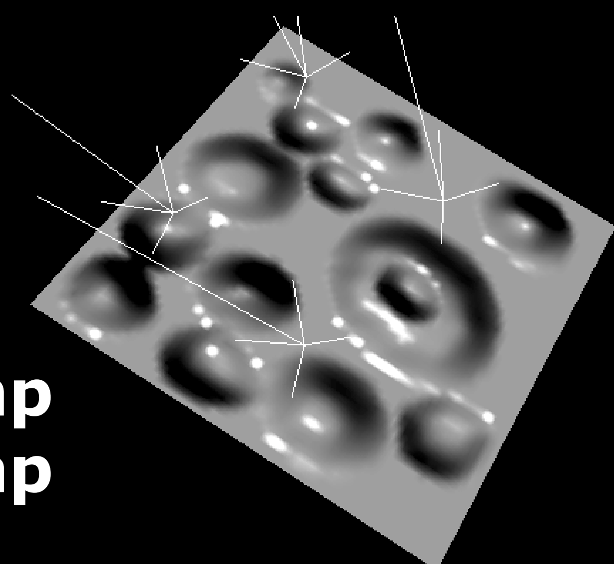
$$L_0 = k_s(\hat{h} \cdot \hat{n})^N + k_d(\hat{n} \cdot \hat{l})$$

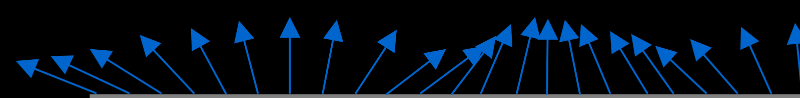
Introduction - Bump Mapping

Real Geometry









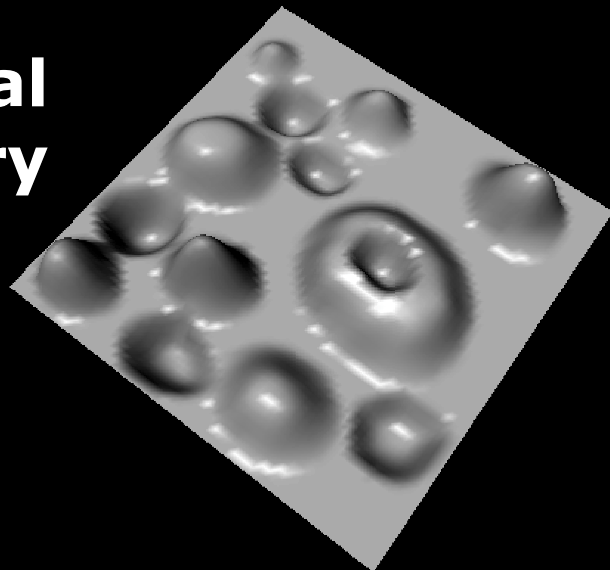
Bump Map

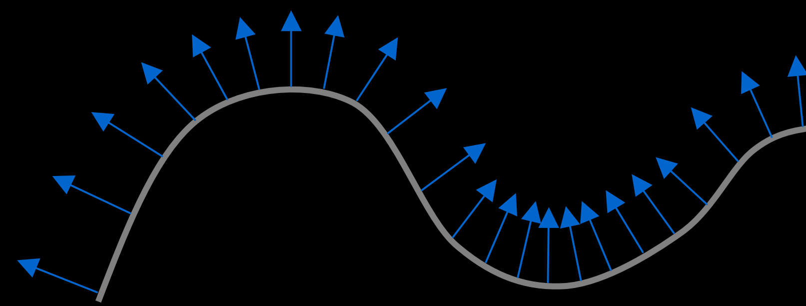
Illumination:

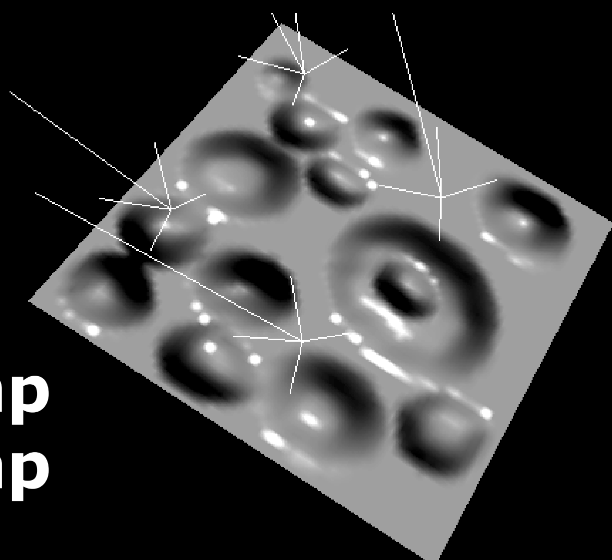
$$L_0 = k_s(\hat{h}\cdot\hat{n})^N + k_d(\hat{n}\cdot\hat{l})$$

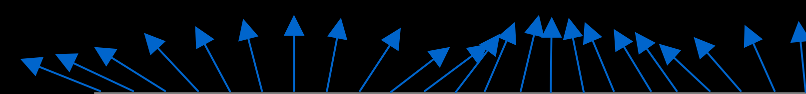
Introduction - Bump Mapping

Real Geometry









Bump Map

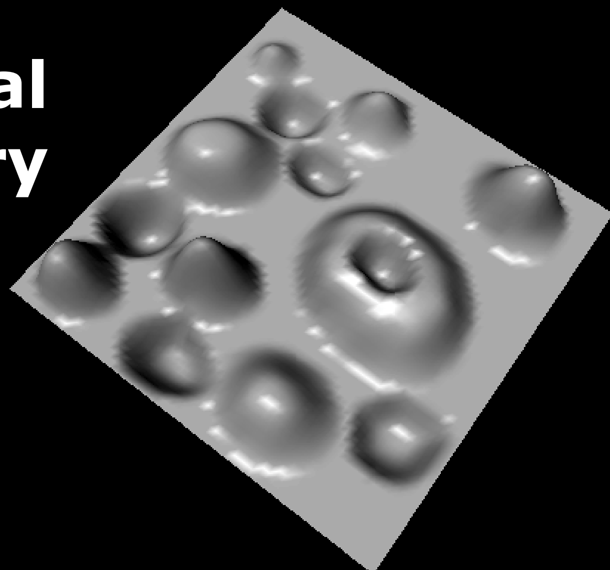
Illumination:

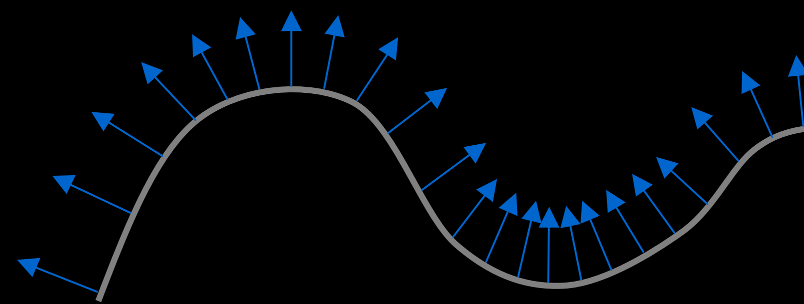
$$L_0 = k_s (\hat{h} \cdot \hat{n})^N + k_d (\hat{n} \cdot \hat{l})$$

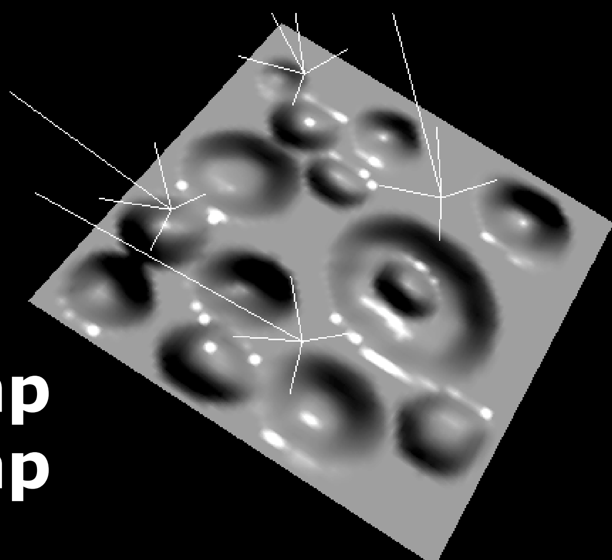
Introduction - Bump Mapping

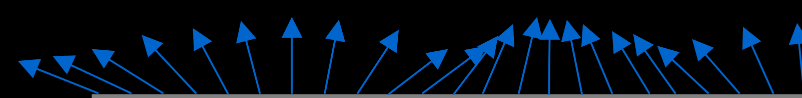
mpi INFORMATIK

Real Geometry







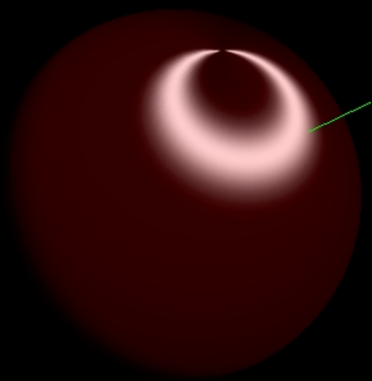


Bump Map

Illumination:

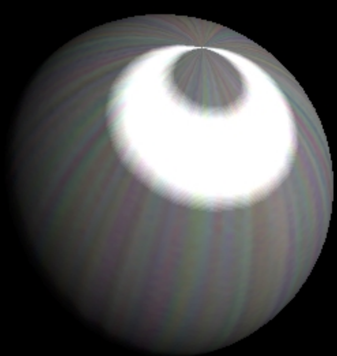
$$L_0 = k_s (\hat{h} \cdot \hat{n})^N + k_d (\hat{n} \cdot \hat{l})$$

Introduction – Reflectance Models



Banks:

$$L_0 = k_s(\hat{n}\cdot\hat{l})\left(\sqrt{1-(\hat{v}\cdot\hat{t})^2}\sqrt{1-(\hat{l}\cdot\hat{t})^2}-(\hat{v}\cdot\hat{t})(\hat{l}\cdot\hat{t})\right)$$



Anisotropic Blinn-Phong:

$$L_0 = k_s\sqrt{1-\left(\hat{h}\cdot\frac{\hat{t}}{\alpha_x}\right)^2-\left(\hat{h}\cdot\frac{\hat{b}}{\alpha_y}\right)^2}^{\,N}$$

Ward:

$$L_0 = \frac{k_s(\hat{n}\cdot\hat{l})}{\sqrt{(\hat{l}\cdot\hat{n})(\hat{v}\cdot\hat{n})}}\frac{1}{4\pi\alpha_x\alpha_y}\mathrm{e}\left(\frac{-2}{1+\hat{h}\cdot\hat{n}}\left(\left(\frac{\hat{h}\cdot\hat{t}}{\alpha_x}\right)^2+\left(\frac{\hat{h}\cdot\hat{b}}{\alpha_y}\right)^2\right)\right)$$

Hardware Capabilities

INFORMATIK

Modern graphics hardware has per-pixel:

- Addition and subtraction
- Multiplication
- Dot-product
- Extended range [-1:1]

⇨ **Not enough for complex reflectance models!**

Dependent Texturing

Colors of 1st texture map serve as texture coordinates of 2nd texture map:

Mathematically: $f : (u, v) \rightarrow (R, G, B)$

⇨ **Allows arbitrary functions**

Mapping Reflectance Models to Hardware

MPI INFORMATIK

Idea:

- Decompose reflectance model into: supported and unsupported operations
- Use per-pixel operations for supported ops
- Use dependent texturing for unsupported operations/functions
- Put BRDF parameters into texture maps

Two phases:

- Precalculation (for unsupported operations)
- Rendering

Precalculation

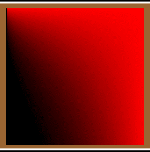
Example: Anisotropic Blinn-Phong

$$L_0 = k_s \sqrt{1 - \left(\hat{h} \cdot \frac{\hat{t}}{\alpha_x}\right)^2 - \left(\hat{h} \cdot \frac{\hat{b}}{\alpha_y}\right)^2}^{\,N}$$

decompose

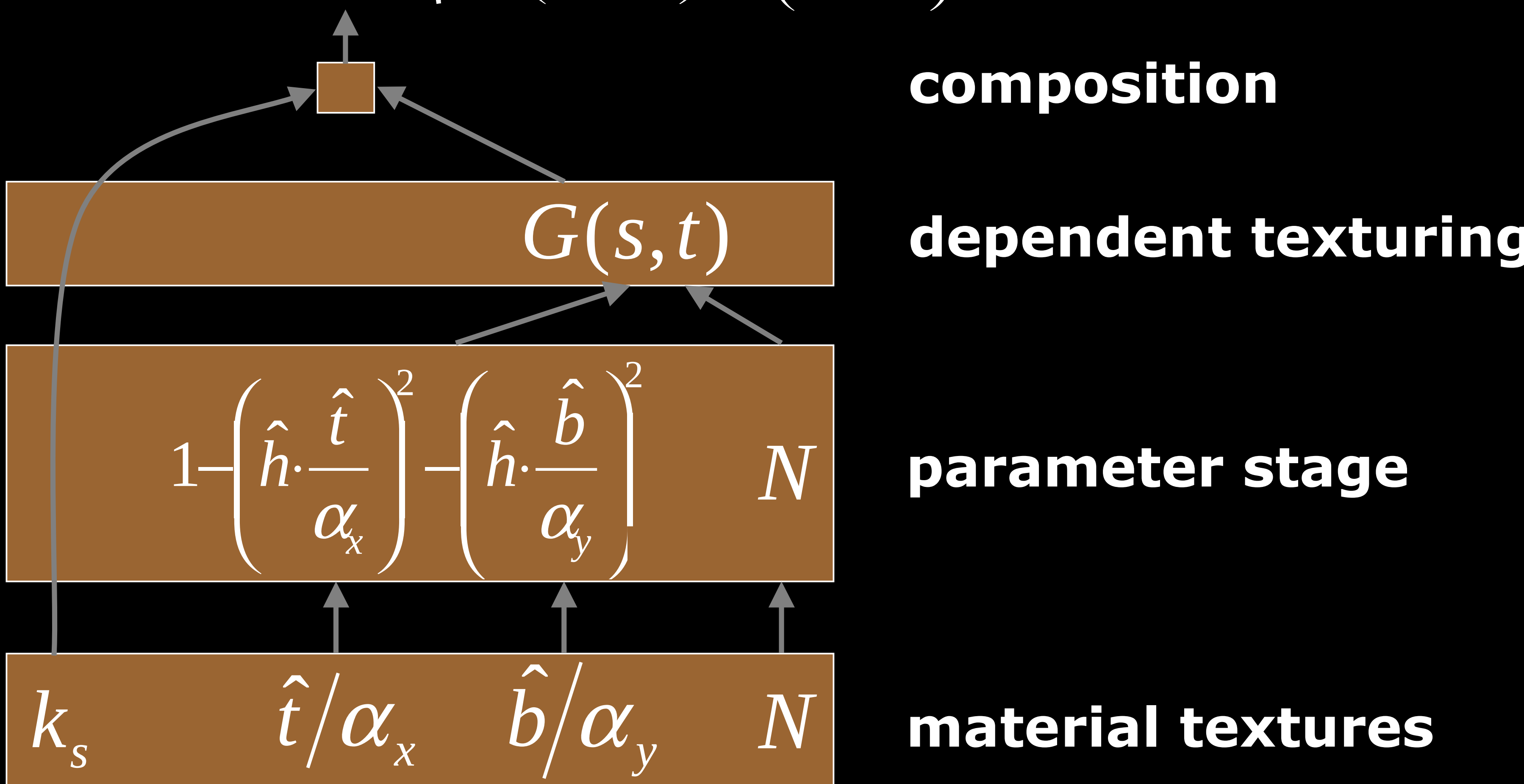
$$G(s,t) = \sqrt{s}^{\,t}$$

put into textures

$G(s,t):$ 

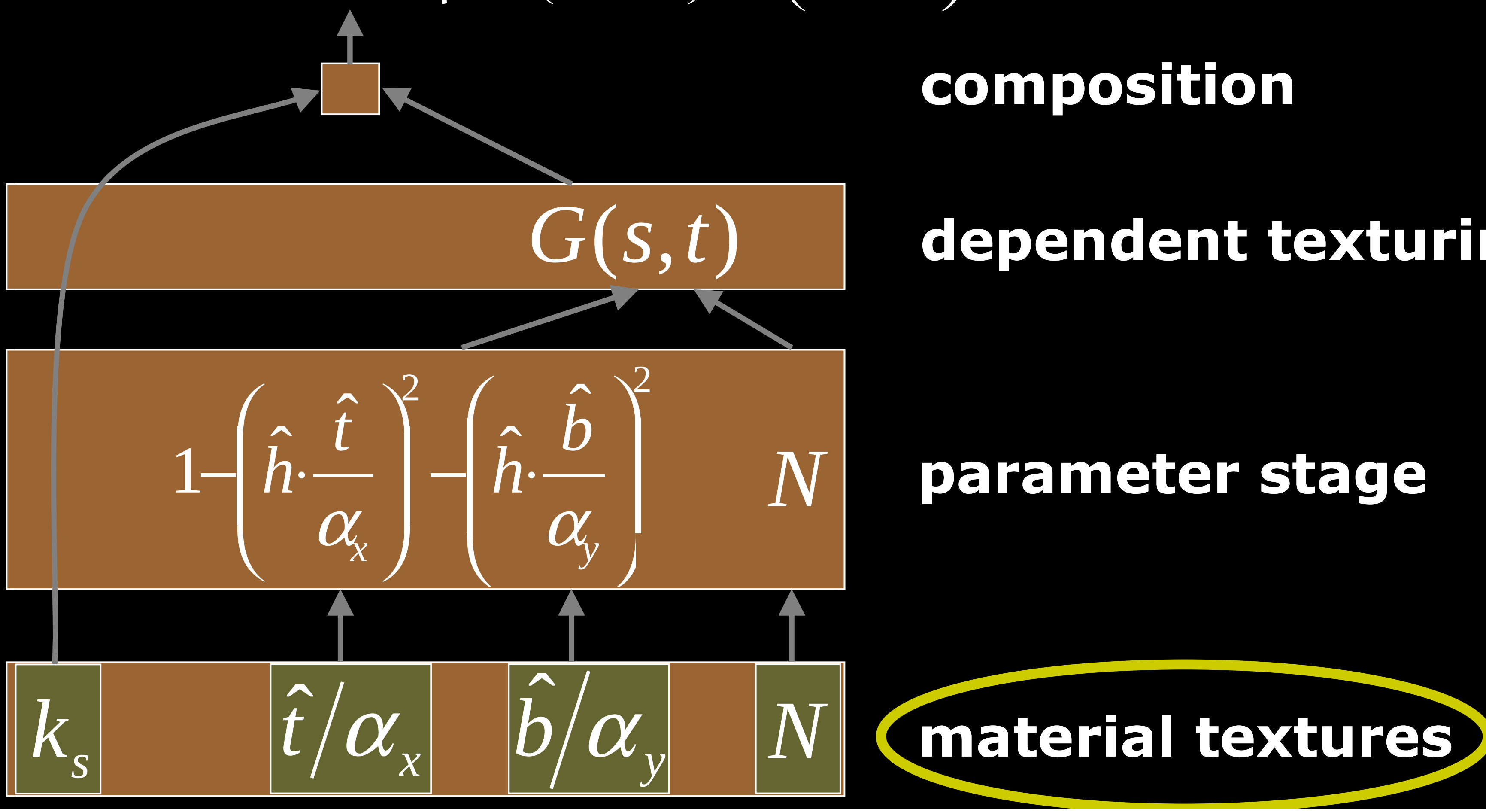
Rendering

$$L_0 = k_s \sqrt{1-\left(\hat{h}\cdot\frac{\hat{t}}{\alpha_x}\right)^2-\left(\hat{h}\cdot\frac{\hat{b}}{\alpha_y}\right)^2}^{\,N}$$



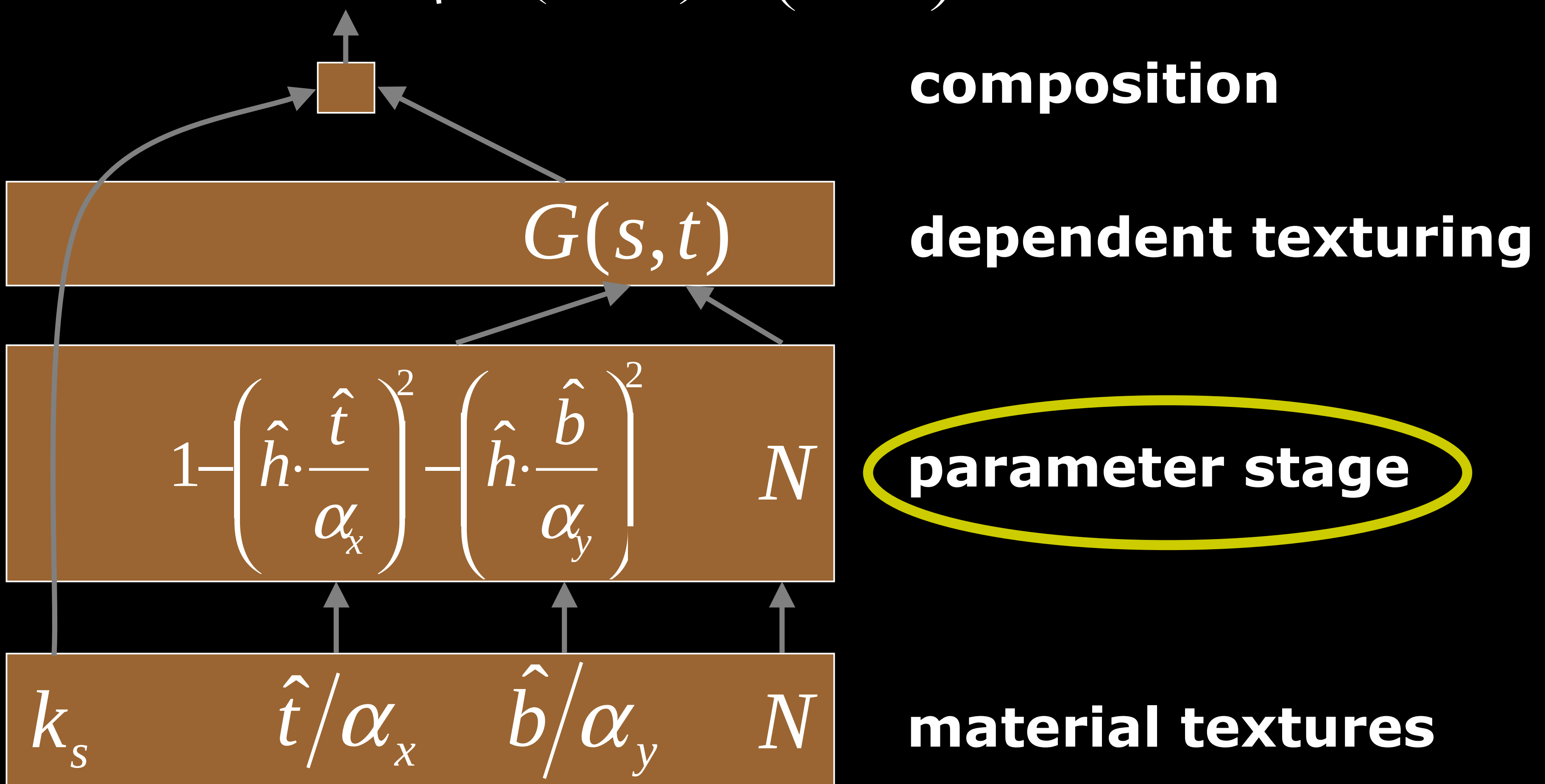
Rendering

$$L_0 = k_s \sqrt[N]{1 - \left(\hat{h}\cdot\frac{\hat{t}}{\alpha_x}\right)^2 - \left(\hat{h}\cdot\frac{\hat{b}}{\alpha_y}\right)^2}$$



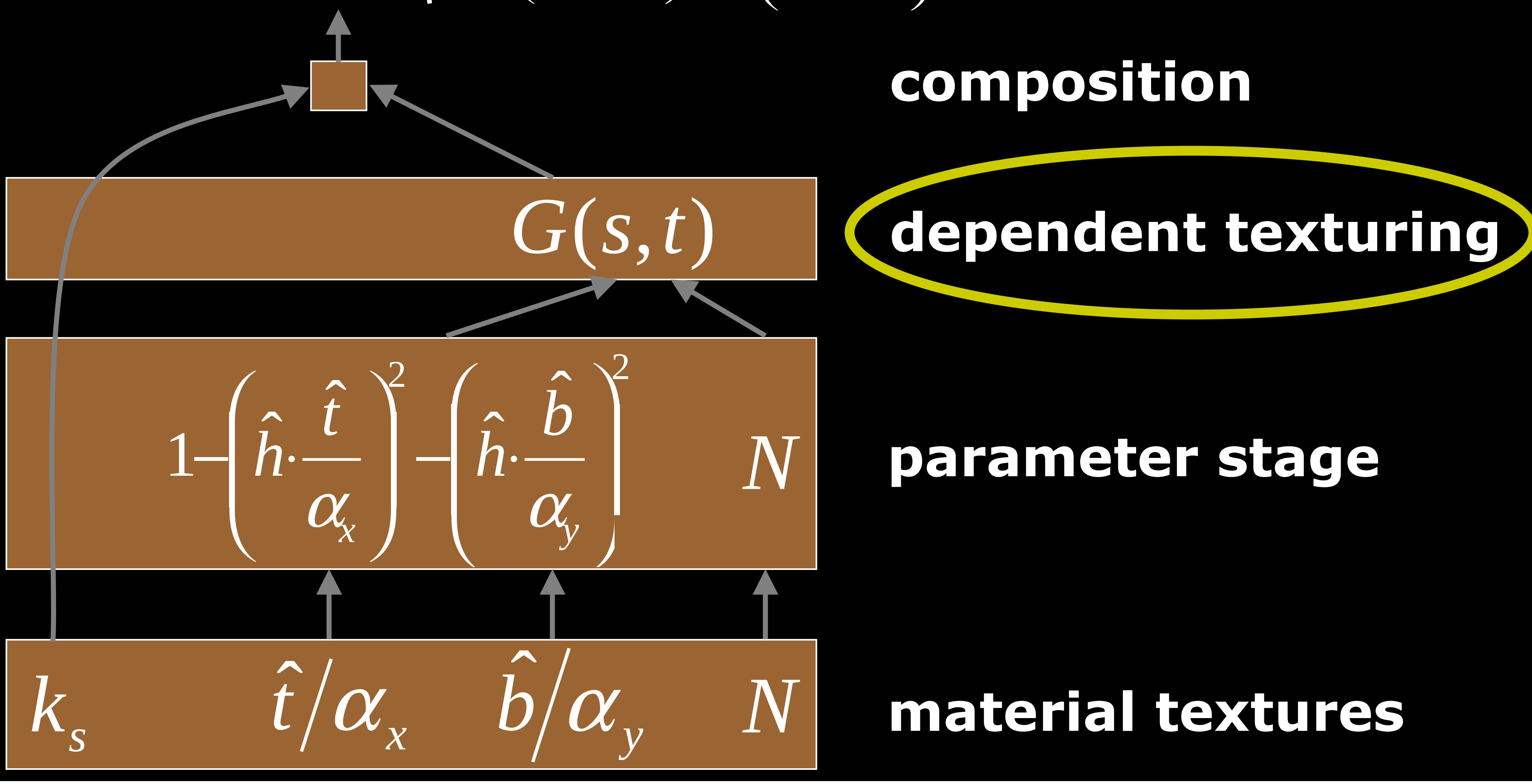
Rendering

$$L_0 = k_s \sqrt[N]{1 - \left(\hat{h} \cdot \frac{\hat{t}}{\alpha_x}\right)^2 - \left(\hat{h} \cdot \frac{\hat{b}}{\alpha_y}\right)^2}$$



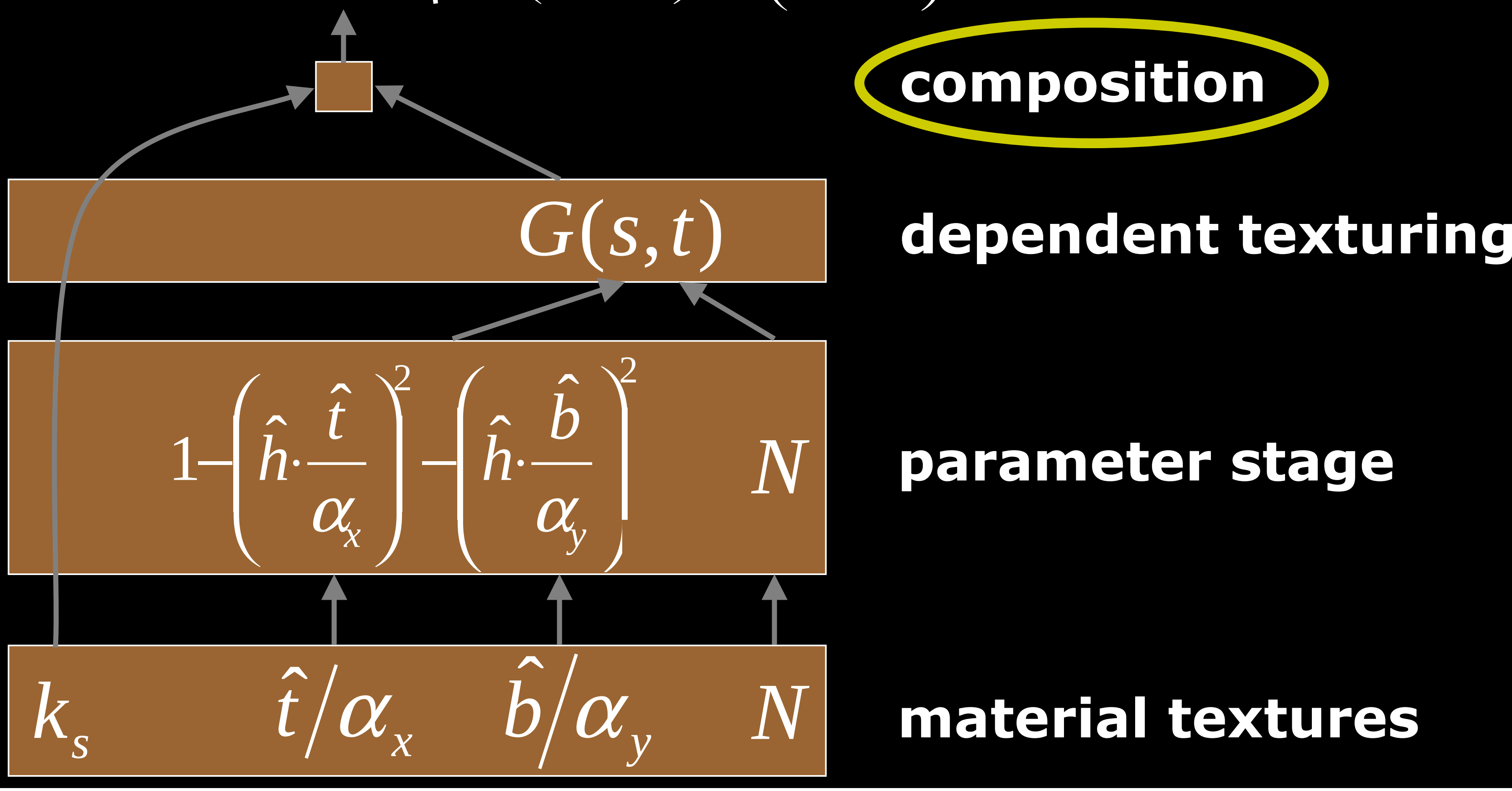
Rendering

$$L_0 = k_s \sqrt{1-\left(\hat{h}\cdot\frac{\hat{t}}{\alpha_x}\right)^2-\left(\hat{h}\cdot\frac{\hat{b}}{\alpha_y}\right)^2}^{\,N}$$

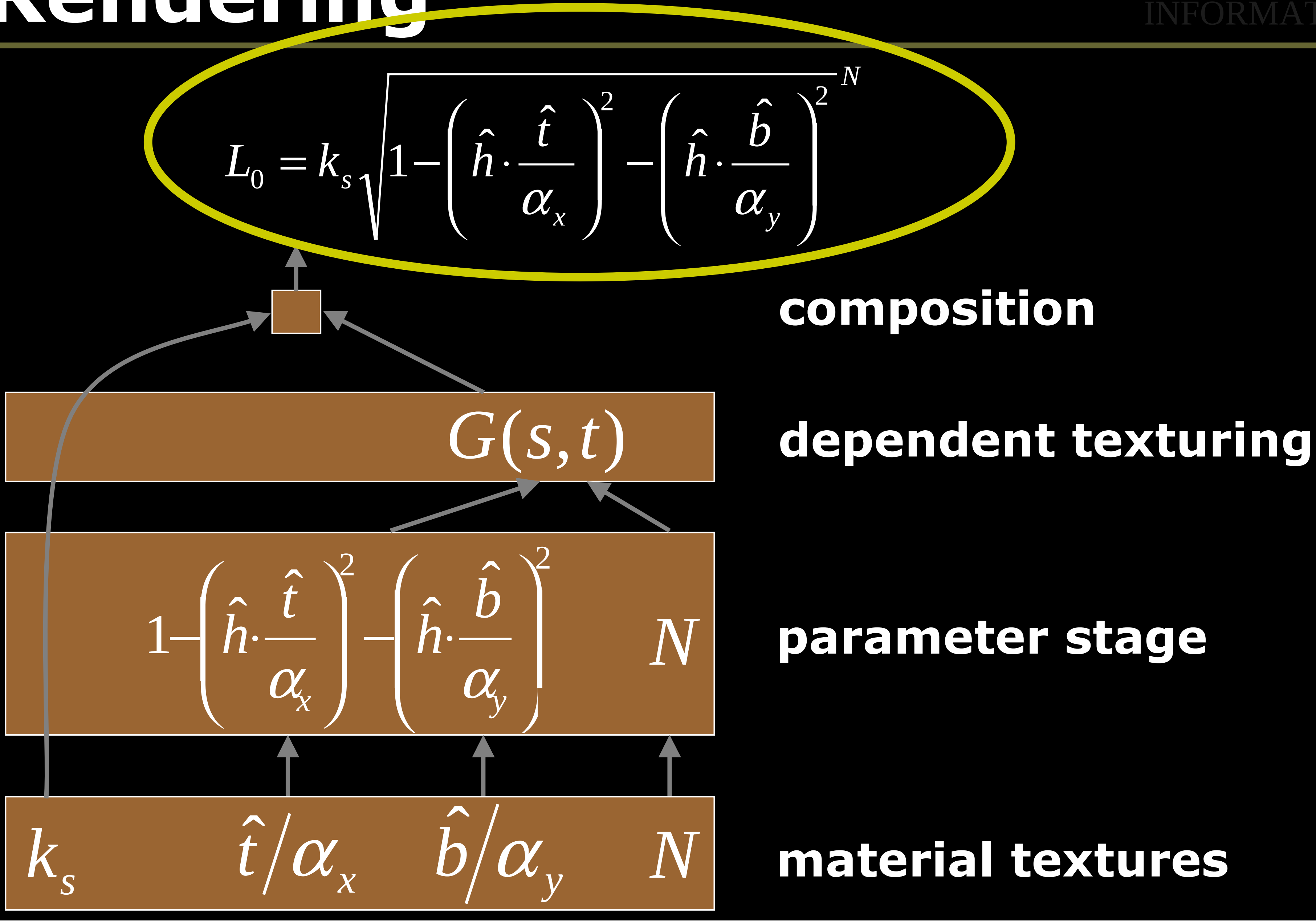


Rendering

$$L_0 = k_s \sqrt[N]{1 - \left(\hat{h} \cdot \frac{\hat{t}}{\alpha_x}\right)^2 - \left(\hat{h} \cdot \frac{\hat{b}}{\alpha_y}\right)^2}$$



Rendering

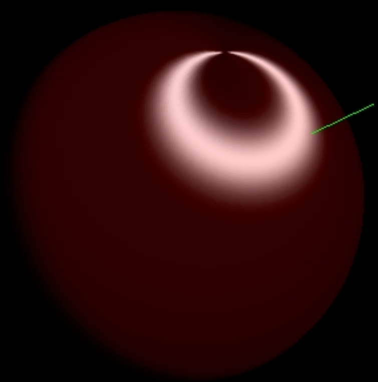


$L_0 = k_s \sqrt{1-\left(\hat{h}\cdot\frac{\hat{t}}{\alpha_x}\right)^2-\left(\hat{h}\cdot\frac{\hat{b}}{\alpha_y}\right)^2}^{\,N}$
composition
$G(s,t)$
dependent texturing
$1-\left(\hat{h}\cdot\frac{\hat{t}}{\alpha_x}\right)^2-\left(\hat{h}\cdot\frac{\hat{b}}{\alpha_y}\right)^2$
N
parameter stage
k_s
$\hat{t}/\alpha_x$
$\hat{b}/\alpha_y$
N
material textures

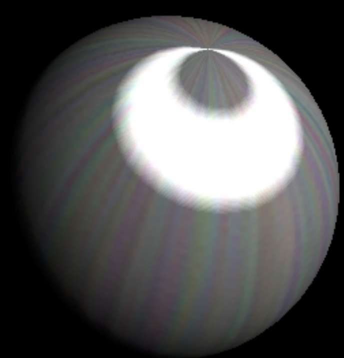
Other Models

Works with most analytical models

We have tried:



Banks



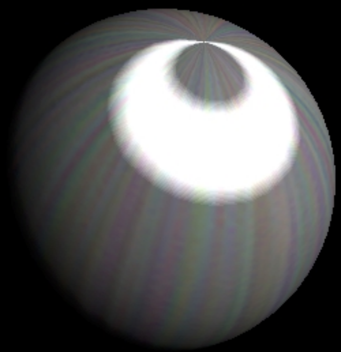
Anisotropic Blinn-Phong

Ward

Results

Rendering are done with:

- Modified anisotropic Blinn-Phong model:

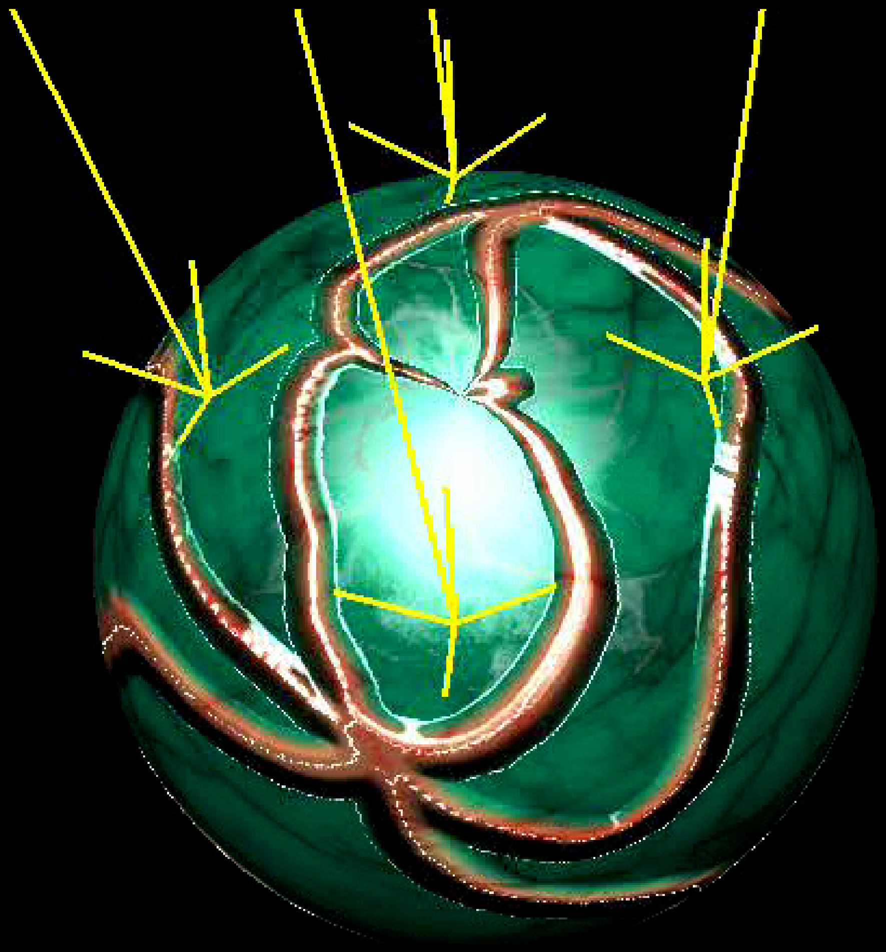


$$L_0 = k_s \left(1 - \left(\hat{h} \cdot \frac{\hat{t}}{\alpha_x} \right)^2 - \left(\hat{h} \cdot \frac{\hat{b}}{\alpha_y} \right)^2 \right)$$

- On a GeForce 256 using register combiners

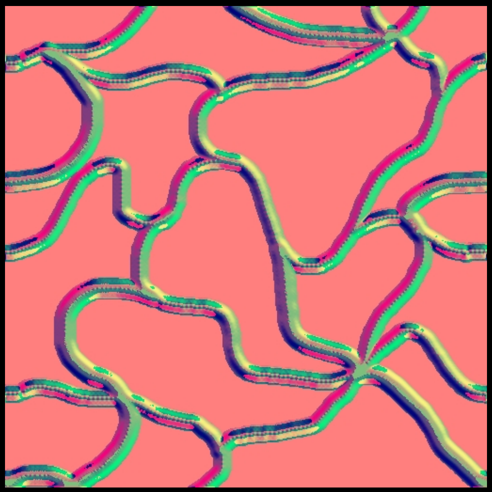
(Also works on SGIs using color matrix)

Results – Anisotropic Blinn-Phong

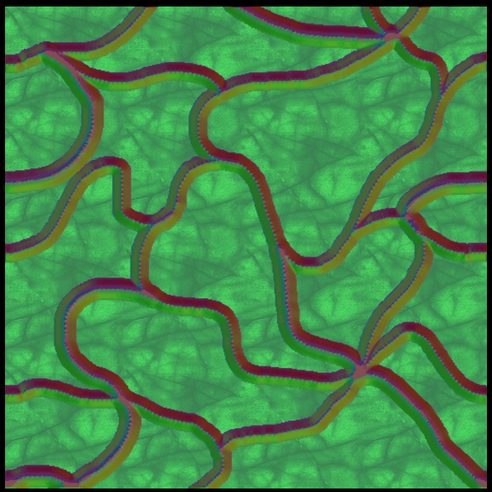


Material textures:

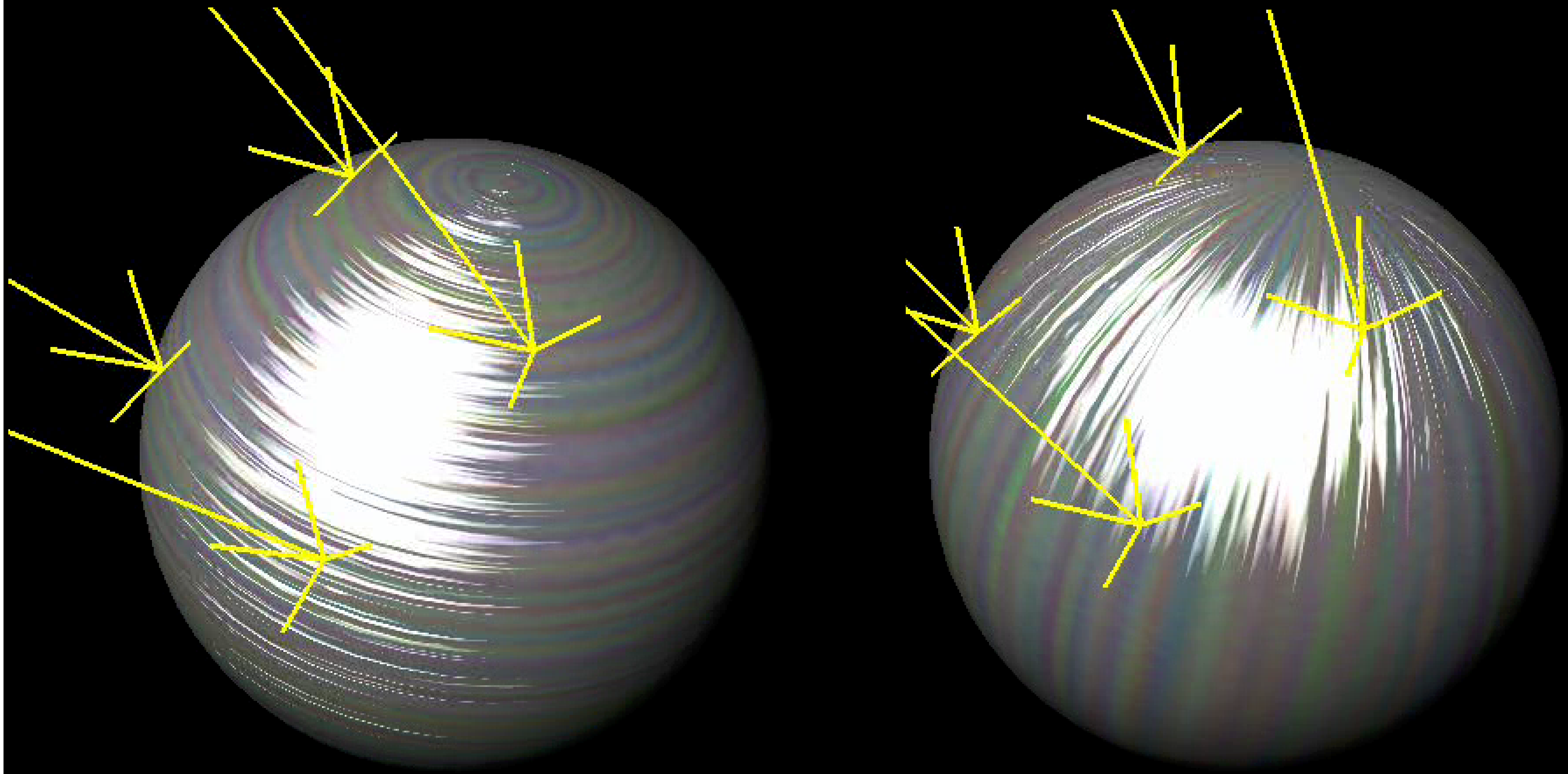
$\frac{\hat{t}}{\alpha_x}$:



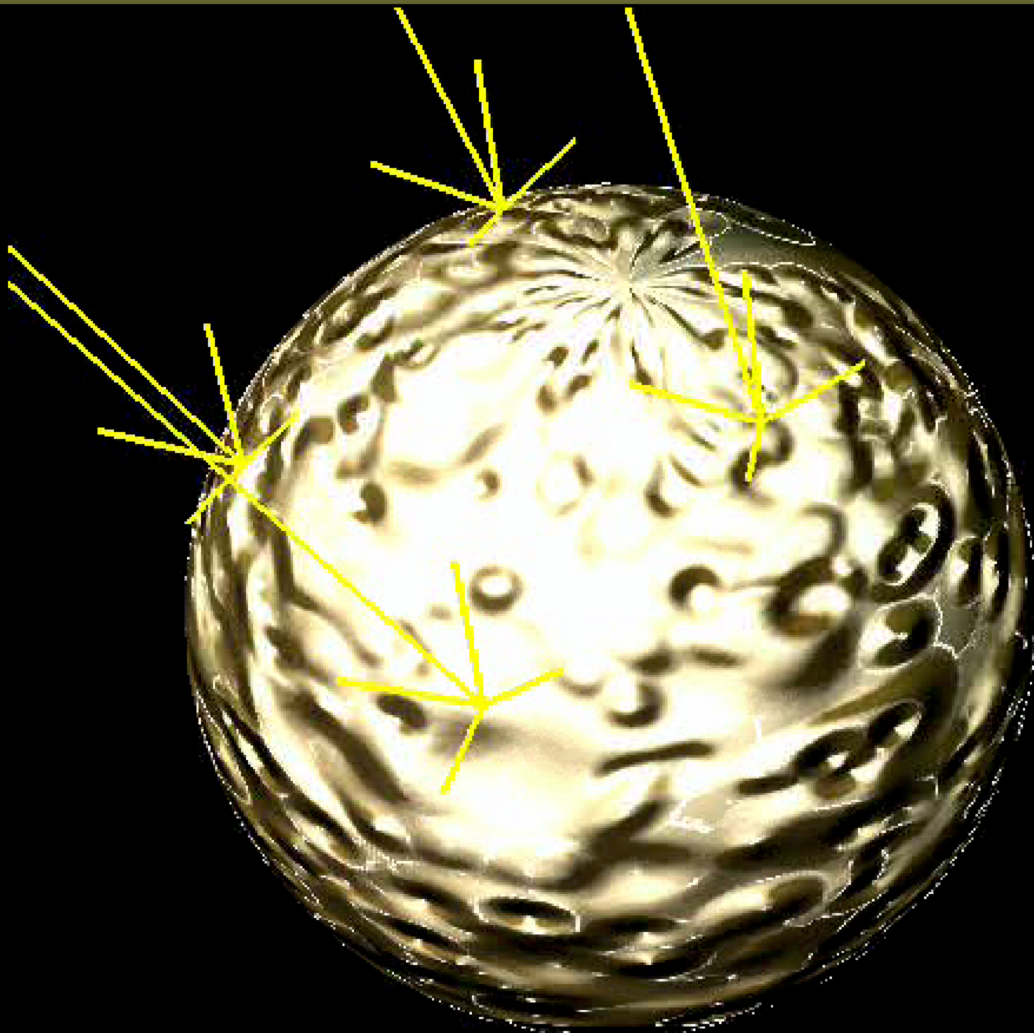
$\frac{\hat{b}}{\alpha_y}$:



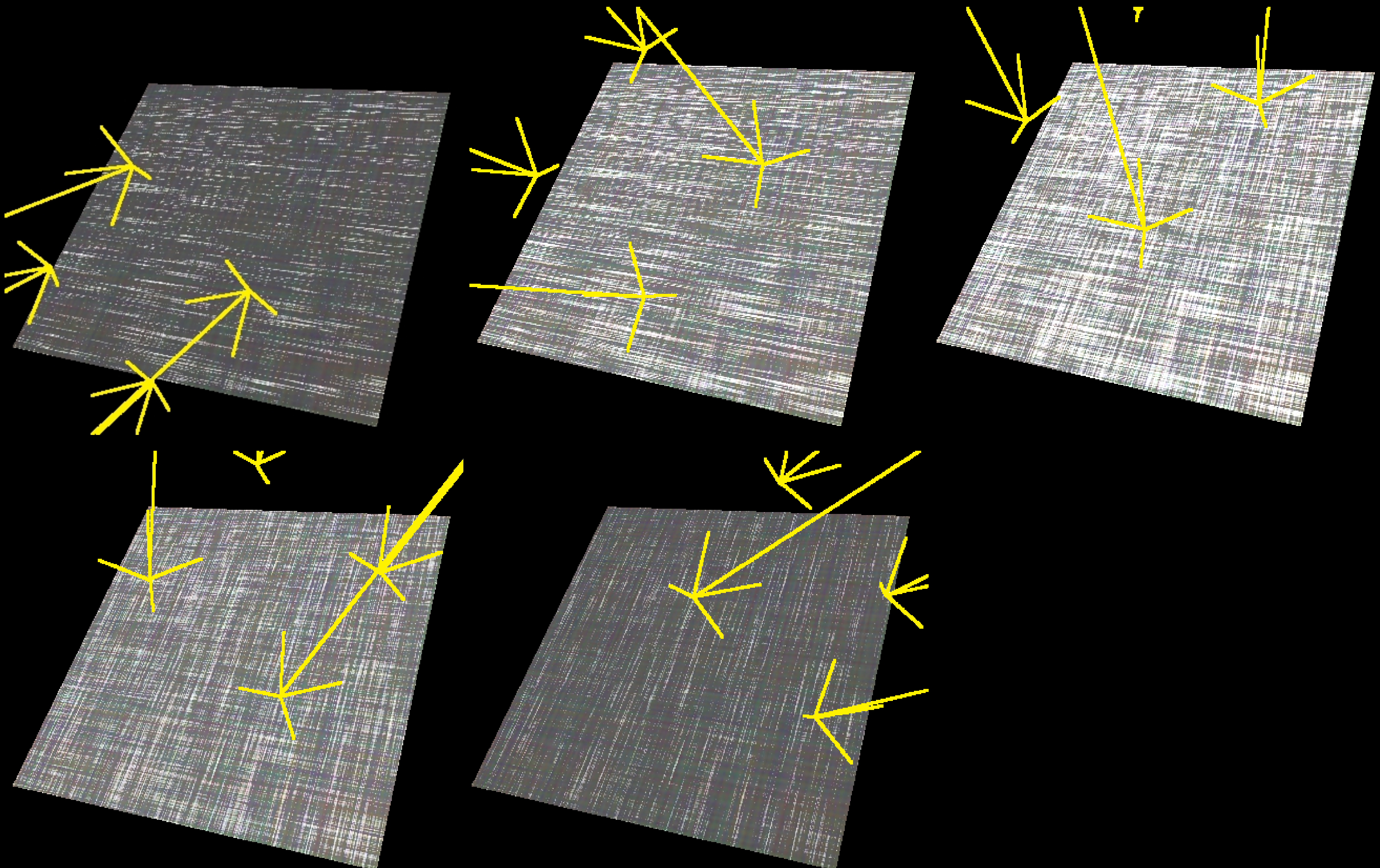
Results



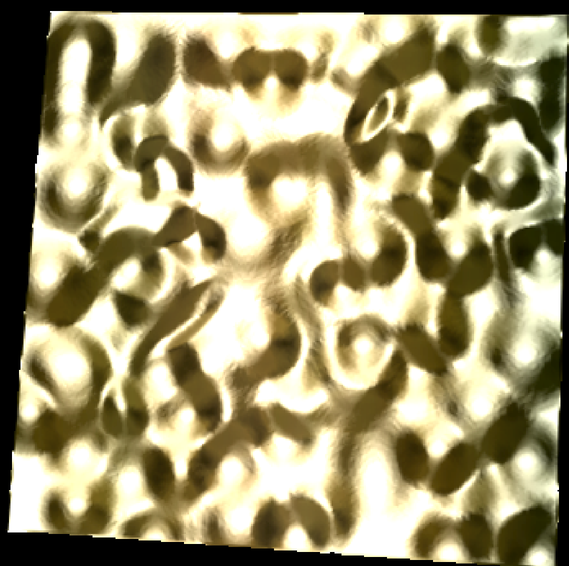
Results



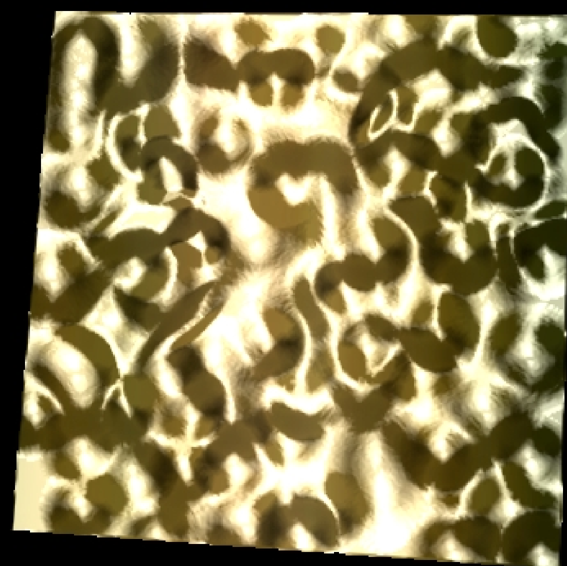
Results



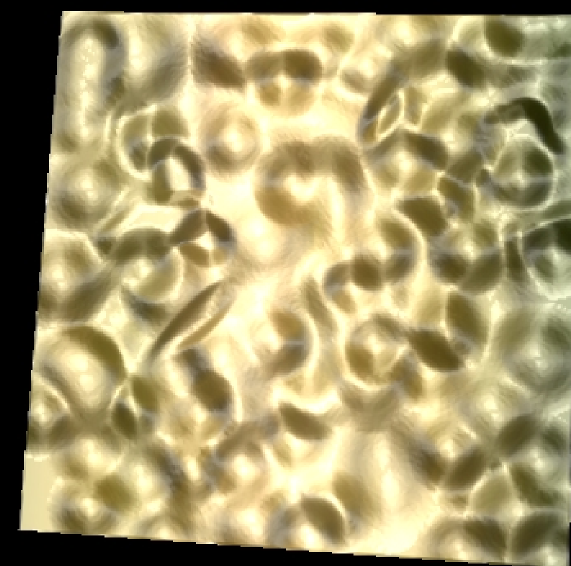
Results



Banks

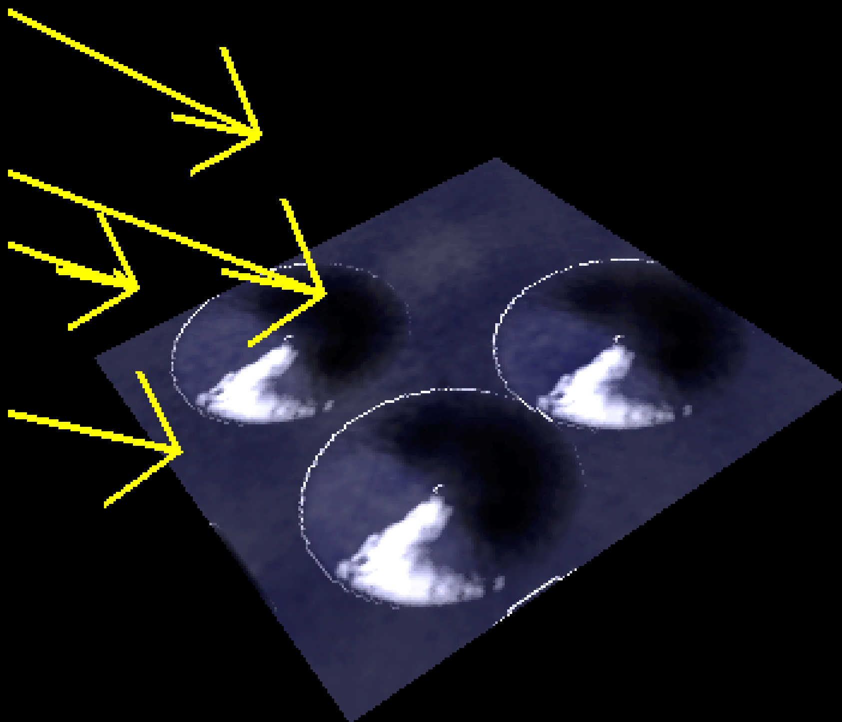


Anisotropic Blinn-Phong

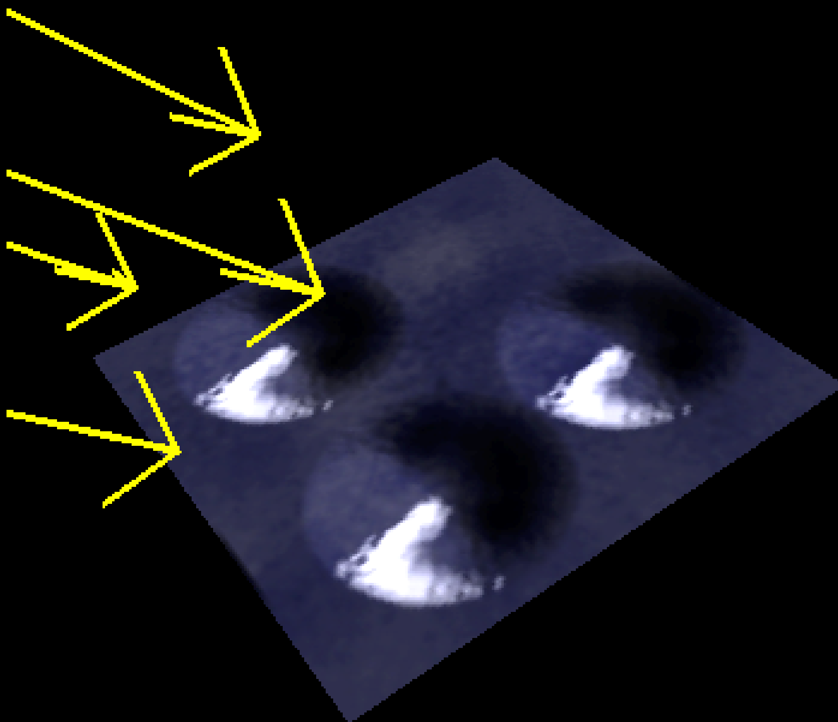


Ward

Issues



NVIDIA register combiner



Software

Hardware Issues

MPI INFORMATIK

Bilinear Filtering of Textures:

- Filtering happens before fed into multi-texturing unit

Dependent Texture Lookup:

- Expensive
- Likely not to happen *within* multi-texturing
- Not widely available (yet)

Discussion

mpi INFORMATIK

Dependent Tex.

- Flexible
- Inconvenient to access
- Expensive to use (depends on data)

Adding New Ops

- Which operations?
- Sqrt, Division, … are *very* expensive
- Always something missing

⇨ **Adding new ops is orthogonal to Dependent Texturing**

Conclusions

INFORMATIK

Technique allows:

- Bump mapping *with*
- Many different
- Shift-Variant BRDFs

Future Work:

- Mip-mapping to avoid aliasing
- Avoid bilinear filtering artefacts

Questions?

INFORMATIK

Thank you!

http://www.mpi-sb.mpg.de