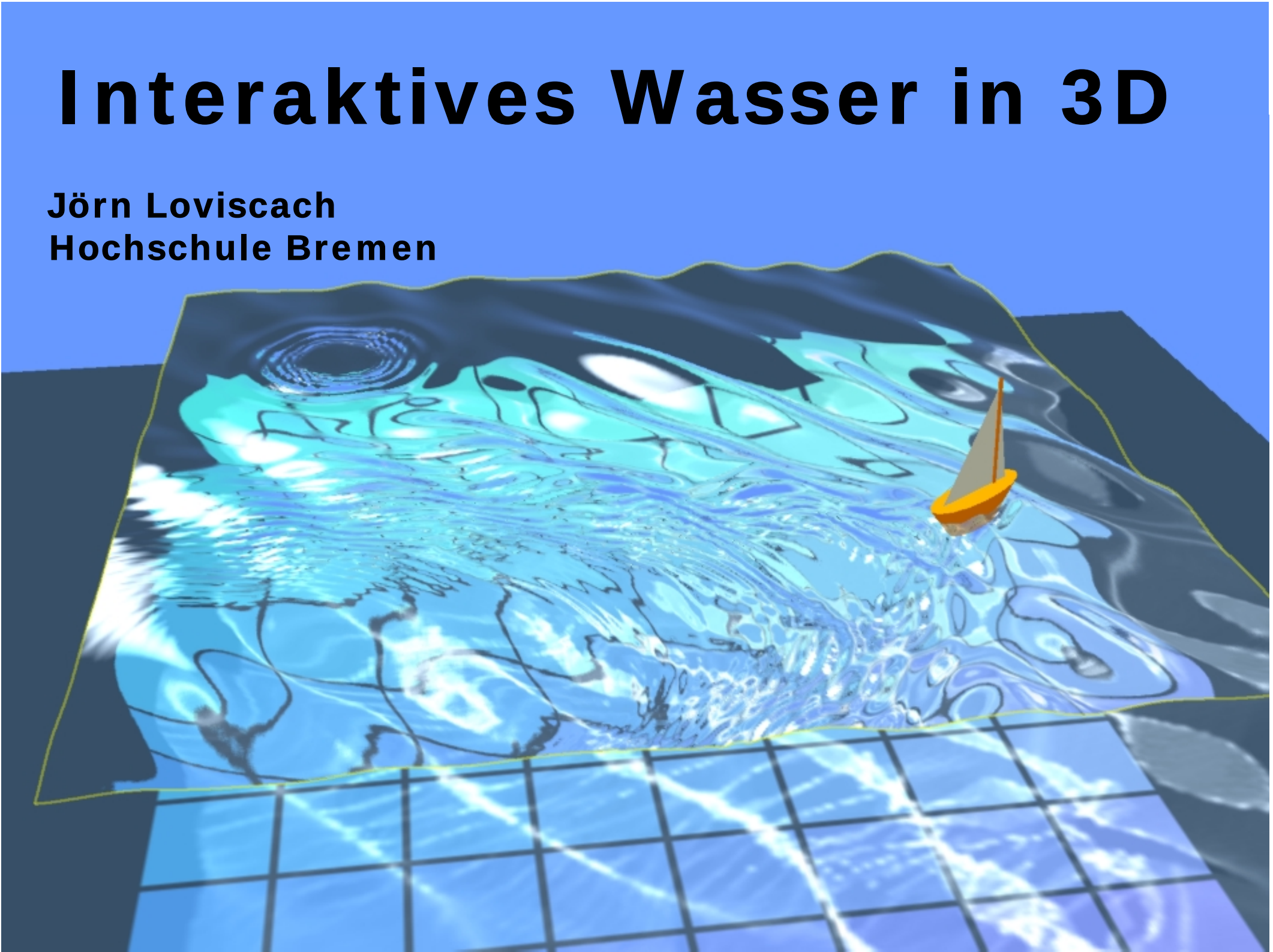
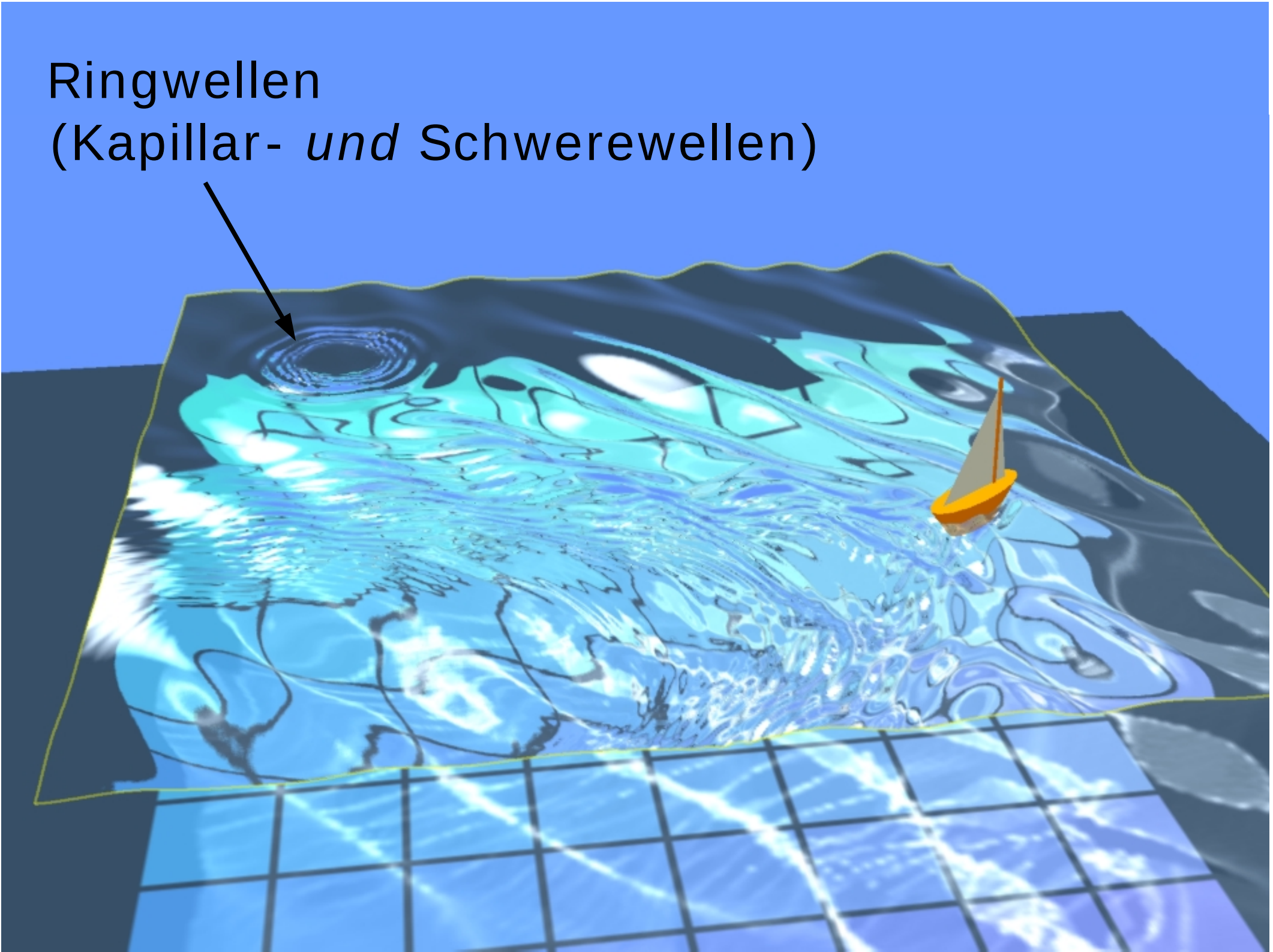
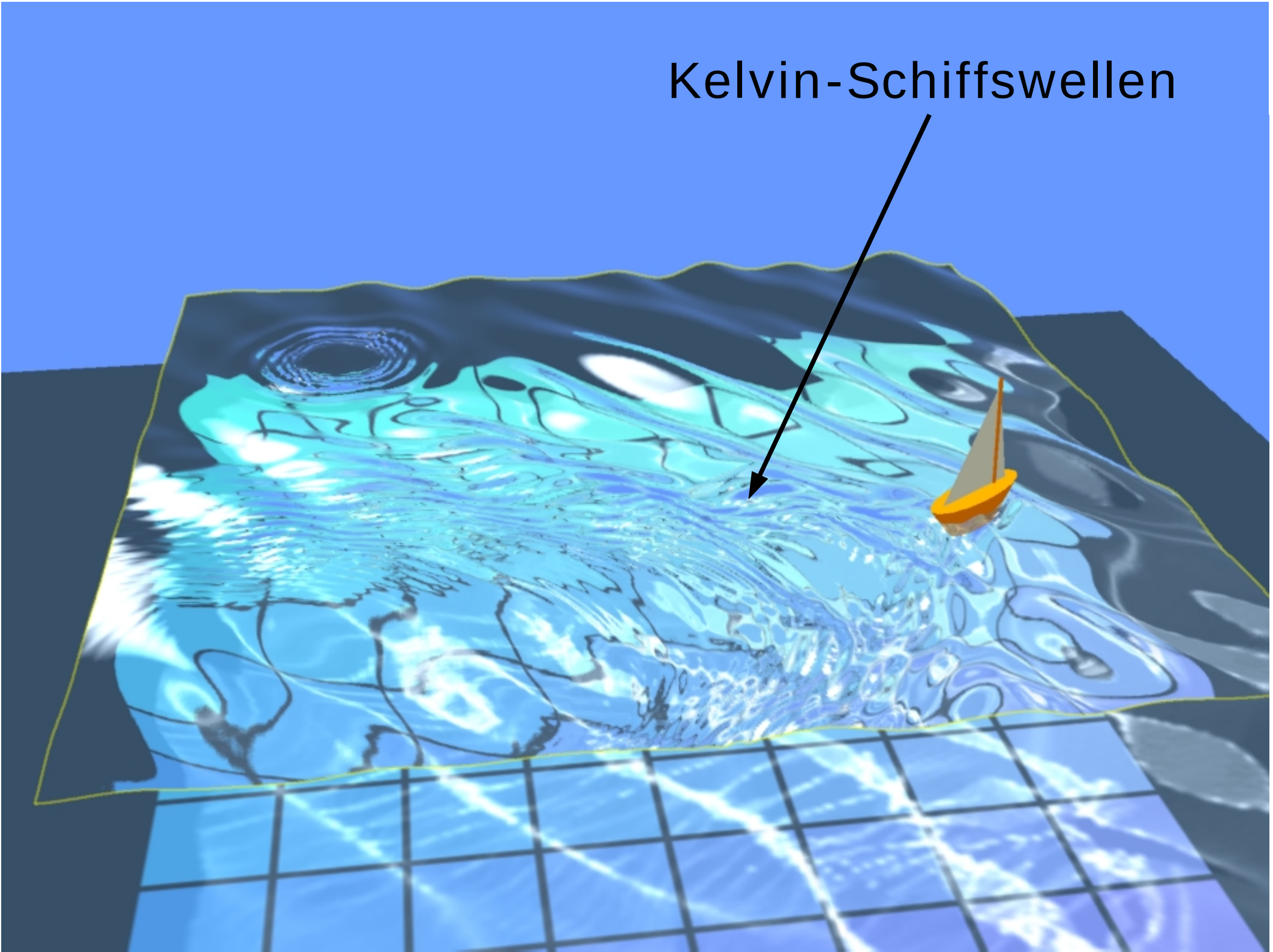
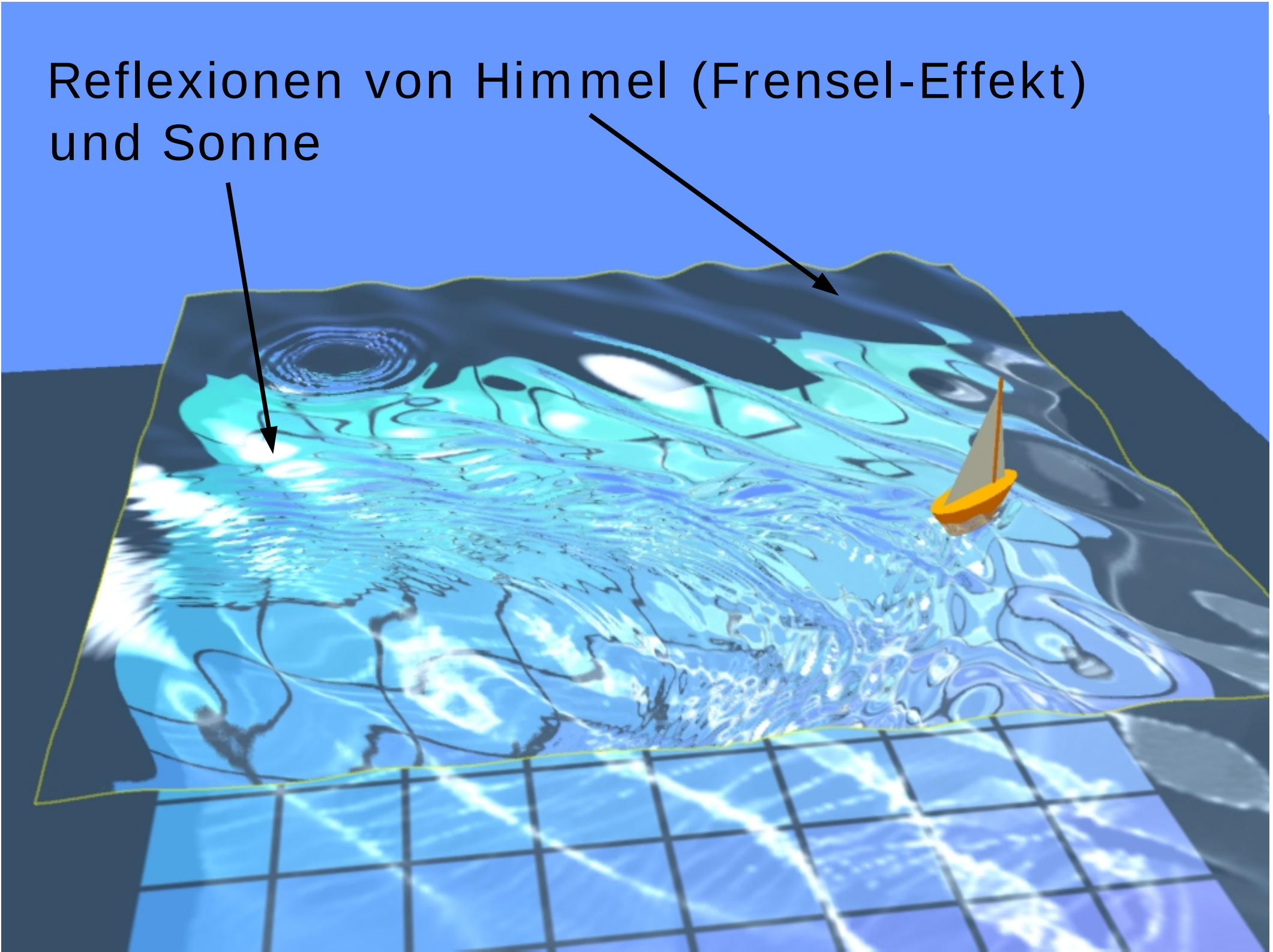
# Interaktives Wasser in 3D

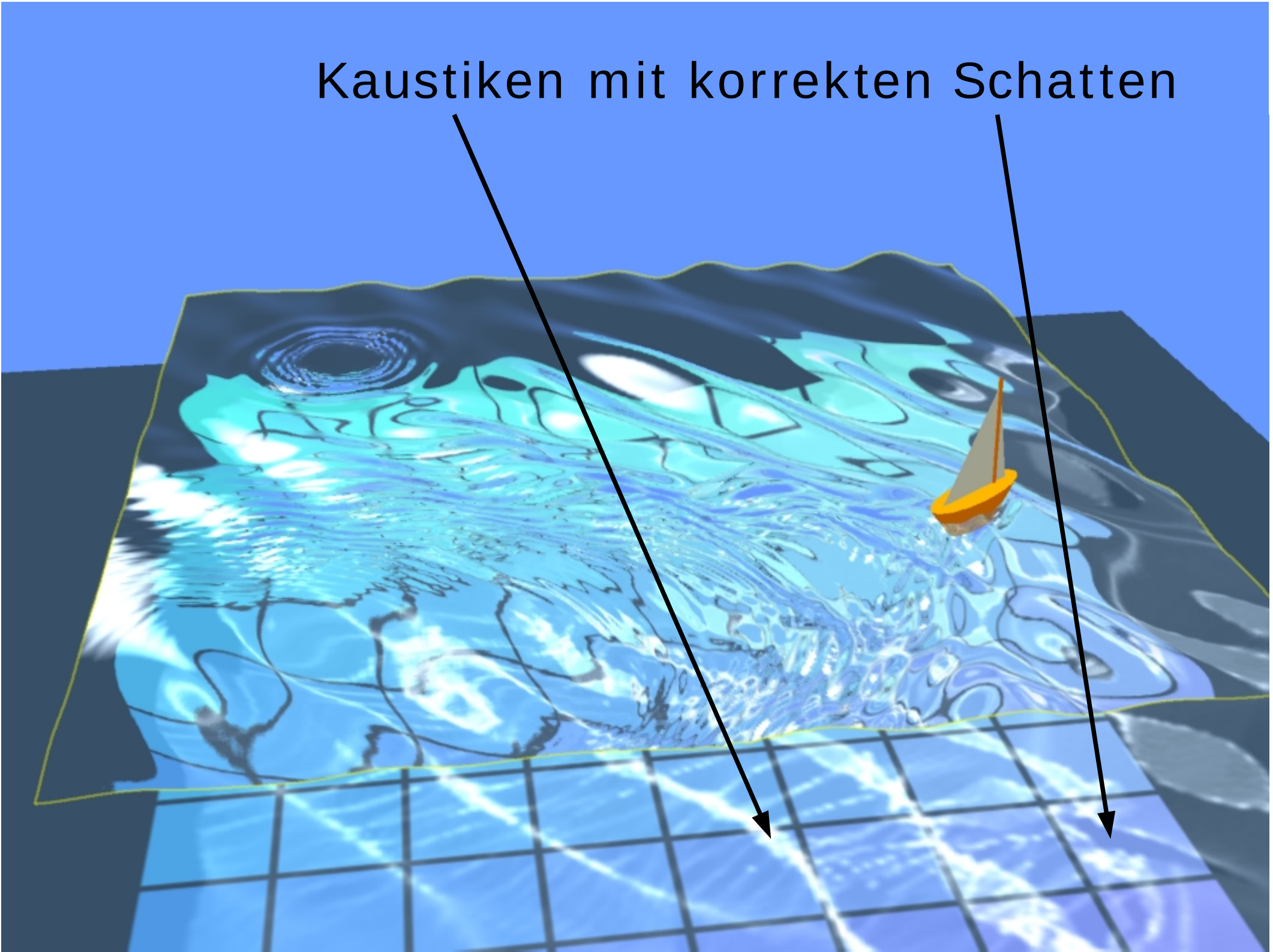
**Jörn Loviscach**
**Hochschule Bremen**

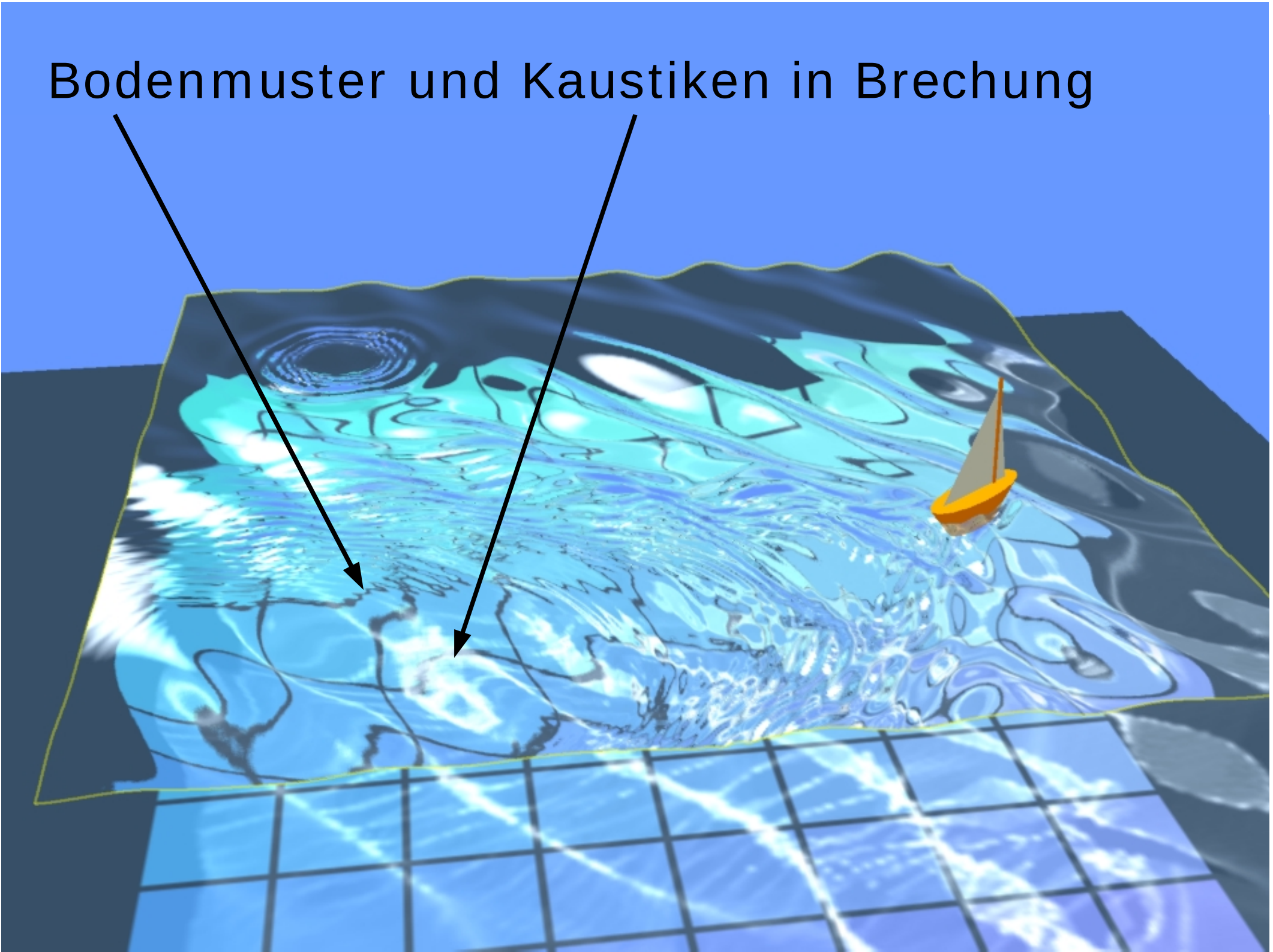



Ringwellen
(Kapillar- *und* Schwerewellen)


Kelvin-Schiffswellen


Reflexionen von Himmel (Frensel-Effekt)
und Sonne


Kaustiken mit korrekten Schatten


Bodenmuster und Kaustiken in Brechung

## Bisherige Arbeiten

ein Vierteljahrundert Wassersimulationen
in der Computergrafik

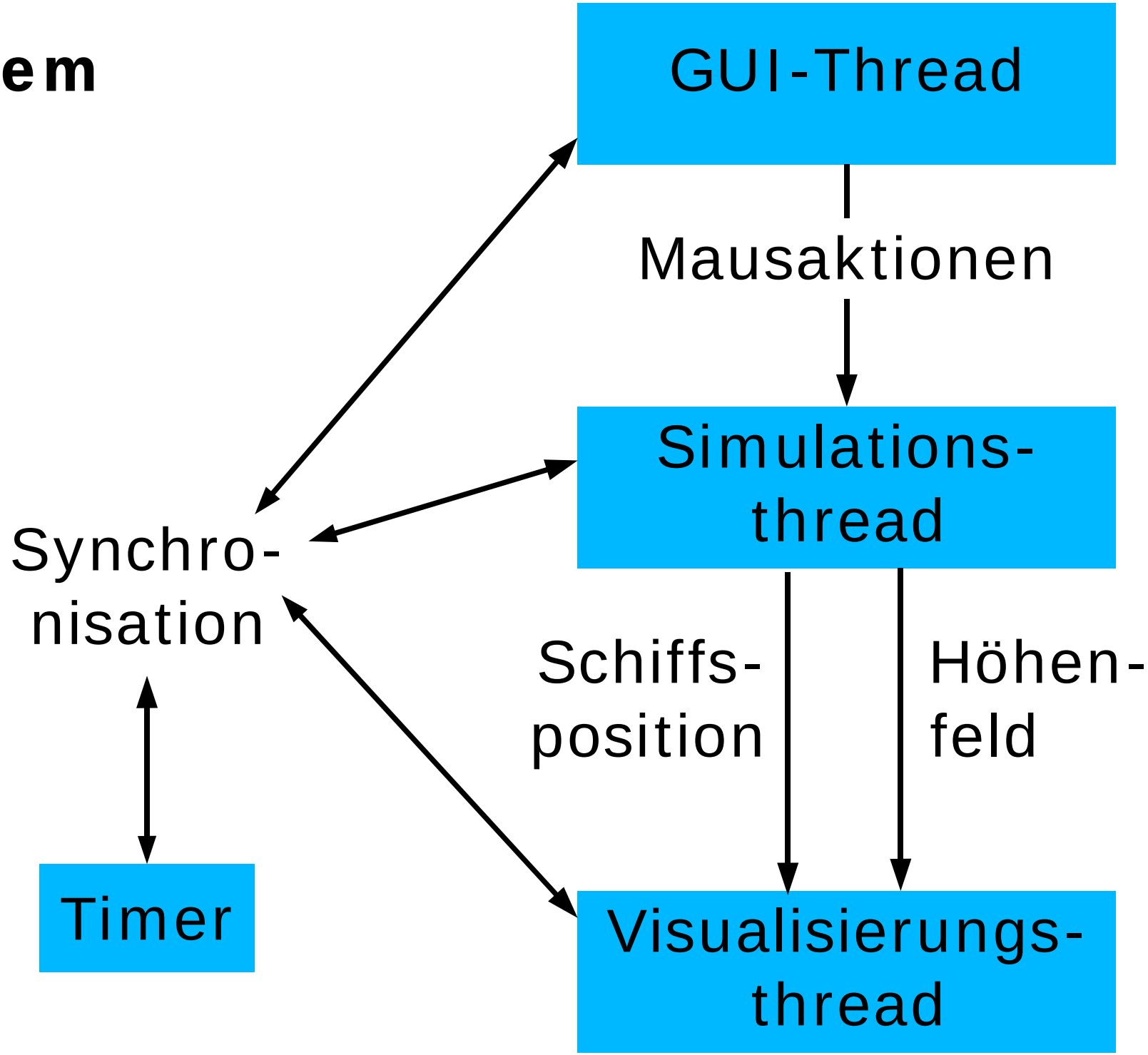
1986: Fournier/Reeves und Peachey

2002: Enright/Marschner/Fedkiw

kommerzielle Produkte:

- Areté Psunami
- Alias|Wavefront Maya 4.5
- ...

**System**


GUI-Thread
Mausaktionen
Simulations-
thread
Synchro-
nisation
Schiffs-
position
Höhen-
feld
Timer
Visualisierungs-
thread

# Implementation

## Intel® SSE

SIMD-Parallelverarbeitung
von vier 32-Bit-Gleitkommazahlen,
hier mit Hilfe der C-Bibliothek IPP 2.0
(Intel® Performance Primitives)

## Pixel-Shader

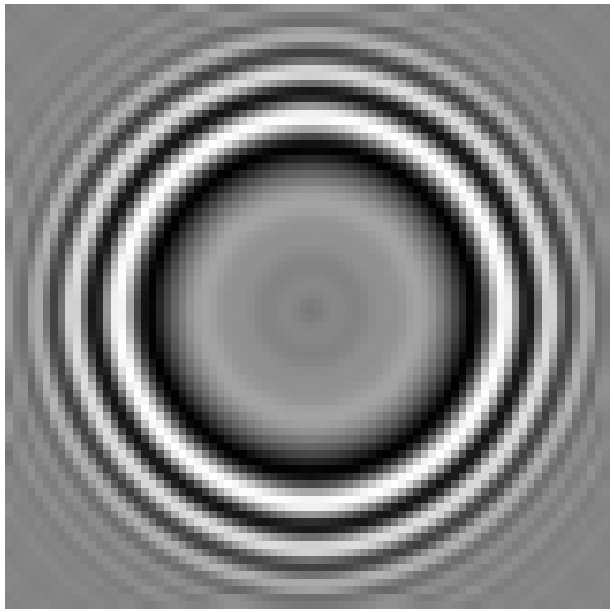
Pro Pixel führt der Grafikchip
konfigurierbare Rechenoperationen aus.

## C++

Geschwindigkeit; Anbindung an Bibliotheken

# Simulation: Wellenausbreitung

Wellenausbreitung: Huygenssches Prinzip
Von jedem Punkt einer Welle
geht eine neue
kreisförmige Welle aus
(lineare Näherung)



Bestimme diese
aus der Dispersionsbeziehung:

$$\omega(k) = \sqrt{gk + \frac{T}{\rho}k^3},$$

$$\Psi(\mathbf{x}) = \int\int_{\mathbb{R}^2} \frac{d^2\mathbf{k}}{4\pi^2} \exp\big(i(\mathbf{k}\cdot\mathbf{x} - \omega(|\mathbf{k}|)\,\Delta t)\big)$$

# Simulation: Wellenausbreitung

Ausbreitung des Wellenfelds: Faltung mit Ringwelle. Numerisch effizienter per FFT.

FFT: periodisch

Simulation eines unendlichen Ozeans durch weiches Abschneiden

# Simulation: Schiffswellen

Aufintegration der Schiffswellen
über einen Zeitschritt
(lineare Interpolation der Bewegung):

$$\iint_{\mathbb{R}^2} d^2\mathbf{y}\,\left(f_t(\mathbf{x}-\mathbf{y})\Phi_1(\mathbf{y}) + f_{t+\Delta t}(\mathbf{x}-\mathbf{y})\Phi_2(\mathbf{y})\right)$$

$$\Phi_1(\mathbf{y}) := \iint_{\mathbb{R}^2} \frac{d^2\mathbf{k}}{4\pi^2} \exp\left(i(\mathbf{k}\cdot\mathbf{y} - \omega(\mathbf{k})\Delta t)\right) \times \left(-1 + i\omega(\mathbf{k})\Delta t \frac{e^{-i\alpha(\mathbf{k})} + i\alpha(\mathbf{k}) - 1}{\alpha(\mathbf{k})^2}\right)$$

$$\Phi_2(\mathbf{y}) := \iint_{\mathbb{R}^2} \frac{d^2\mathbf{k}}{4\pi^2} \exp(i\mathbf{k}\cdot\mathbf{y}) \times \left(1 + i\omega(\mathbf{k})\Delta t \frac{e^{i\alpha(\mathbf{k})} - i\alpha(\mathbf{k}) - 1}{\alpha(\mathbf{k})^2}\right)$$

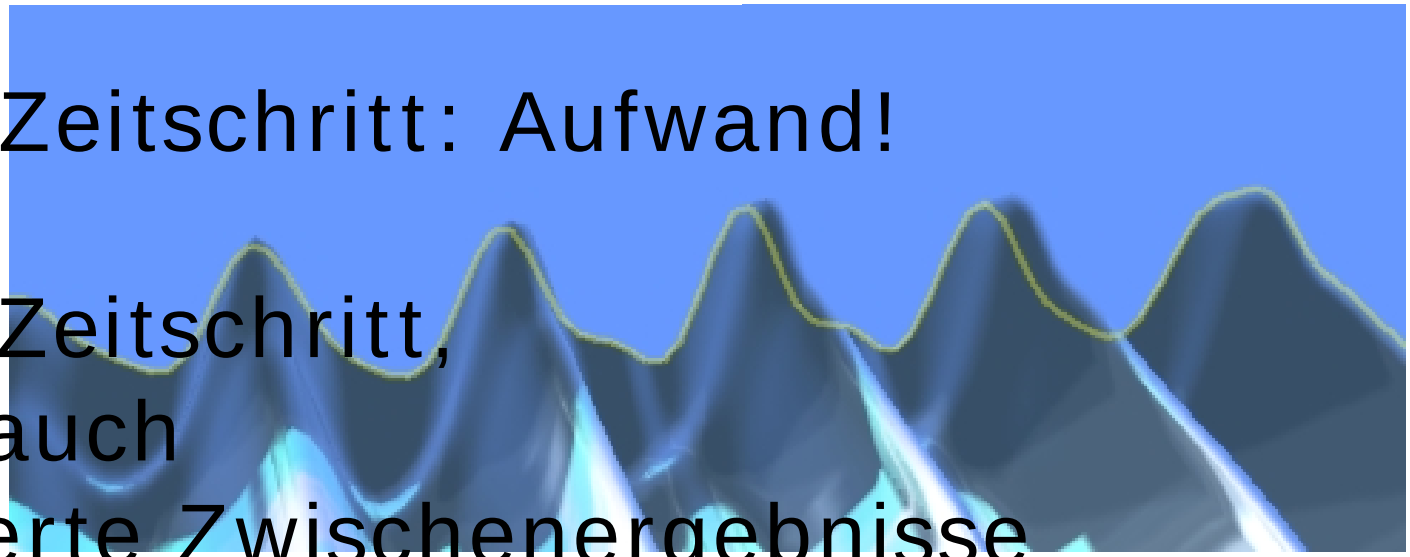
$$\alpha(\mathbf{k}) := (\mathbf{k}\cdot\mathbf{v} - \omega(\mathbf{k}))\Delta t$$

# Simulation: Nichtlinearität

Asymmetrische Wellen:
Modelliere Geschwindigkeitspotential linear,
daraus bestimme Höhenfeld
mit Korrekturen höherer Ordnung

$$\eta(\mathbf{x},t) = ai\frac{|\mathbf{k}|}{\omega}e^{i(\mathbf{k}\cdot\mathbf{x}-\omega t)} - a^2\frac{|\mathbf{k}|^3}{2\omega^2}e^{2i(\mathbf{k}\cdot\mathbf{x}-\omega t)}$$

drei FFTs pro Zeitschritt: Aufwand!



Trick: großer Zeitschritt,
zeige jeweils auch
zwei interpolierte Zwischenergebnisse

# Visualisierung

Effizienz-Prinzip:
wenig Geometrie, viel Texturen

Deshalb:

- Wasseroberfläche als Fläche
  aus 95 x 95 x 2 Dreiecken,

- Texturen (Multitexturing, Pixel Shader)
  mit 384 x 384 Pixel Auflösung

# Visualisierung: Brechung

Textur auf Wasseroberfläche:
lebendige uv-Koordinaten
mit Brechungssimulation

In linearer Näherung berücksichtigt:

- Höhe des Wassers
  (pro Vertex, geringe Auflösung)
- Richtung der Normalen
  (aus Textur, hohe Auflösung)

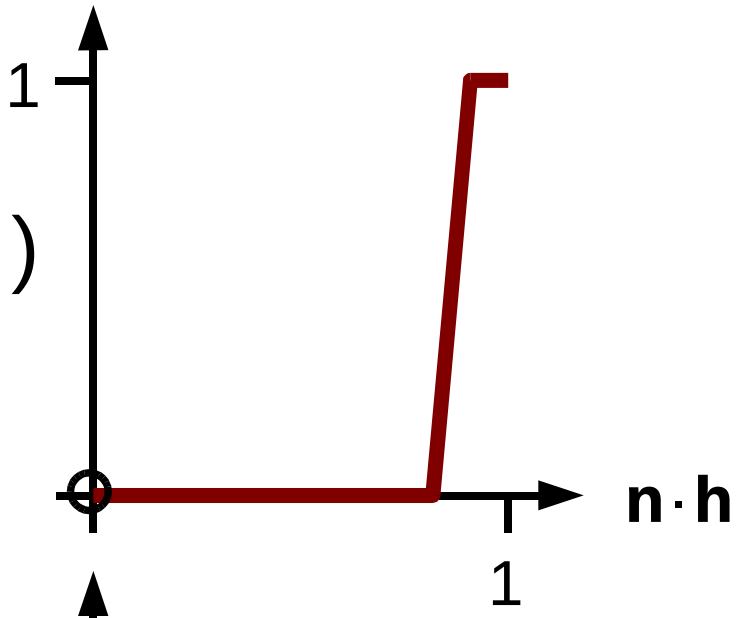
Beide Anteile mit Pixel Shader gemischt.

# Visualisierung: Reflexion

Reflexionen als weitere Texturen,
als stückweise lineare Funktionen modelliert:

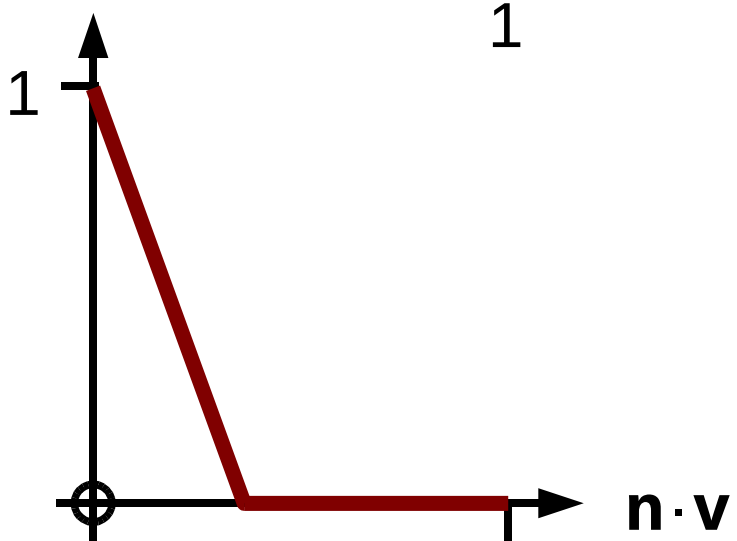
Sonnen-Glanzlichter

clamp(100,0 **n**·**h** – 98,95)



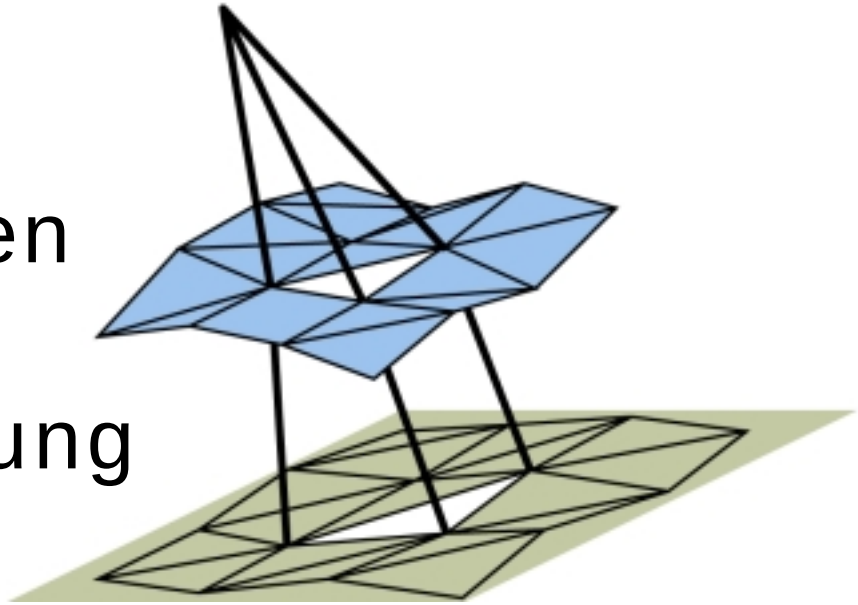

Himmel-Reflexion (Fresnel)

clamp(1,0 – 2,5 **n**·**v**)



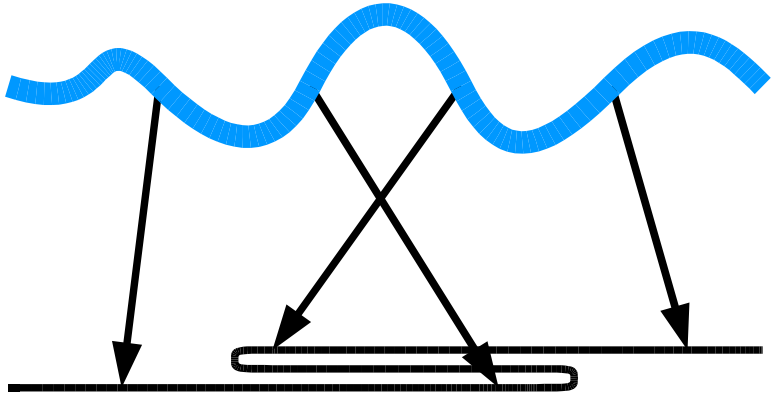
# Visualisierung: Kaustiken 1

Kaustiken auf dem Grund
als Sammlung von Dreiecken

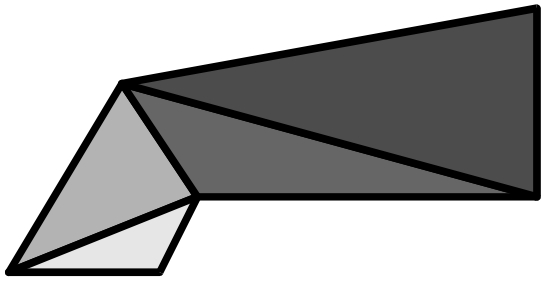


Brechung in linearer Näherung

Überschneidungen:
additives Blending!



Helligkeit umgekehrt
proportional zur Fläche



# Visualisierung: Kaustiken 2

Trick zur Helligkeitsberechnung:

Fläche jedes Dreiecks
nicht explizit
per CPU
berechnen,
sondern MIP-Map
benutzen:
Grafik-Hardware!

# Visualisierung: Kaustiken 3

Probleme:

- Aliasing in Kaustiken
- Dreiecke in Kaustiken einfarbig

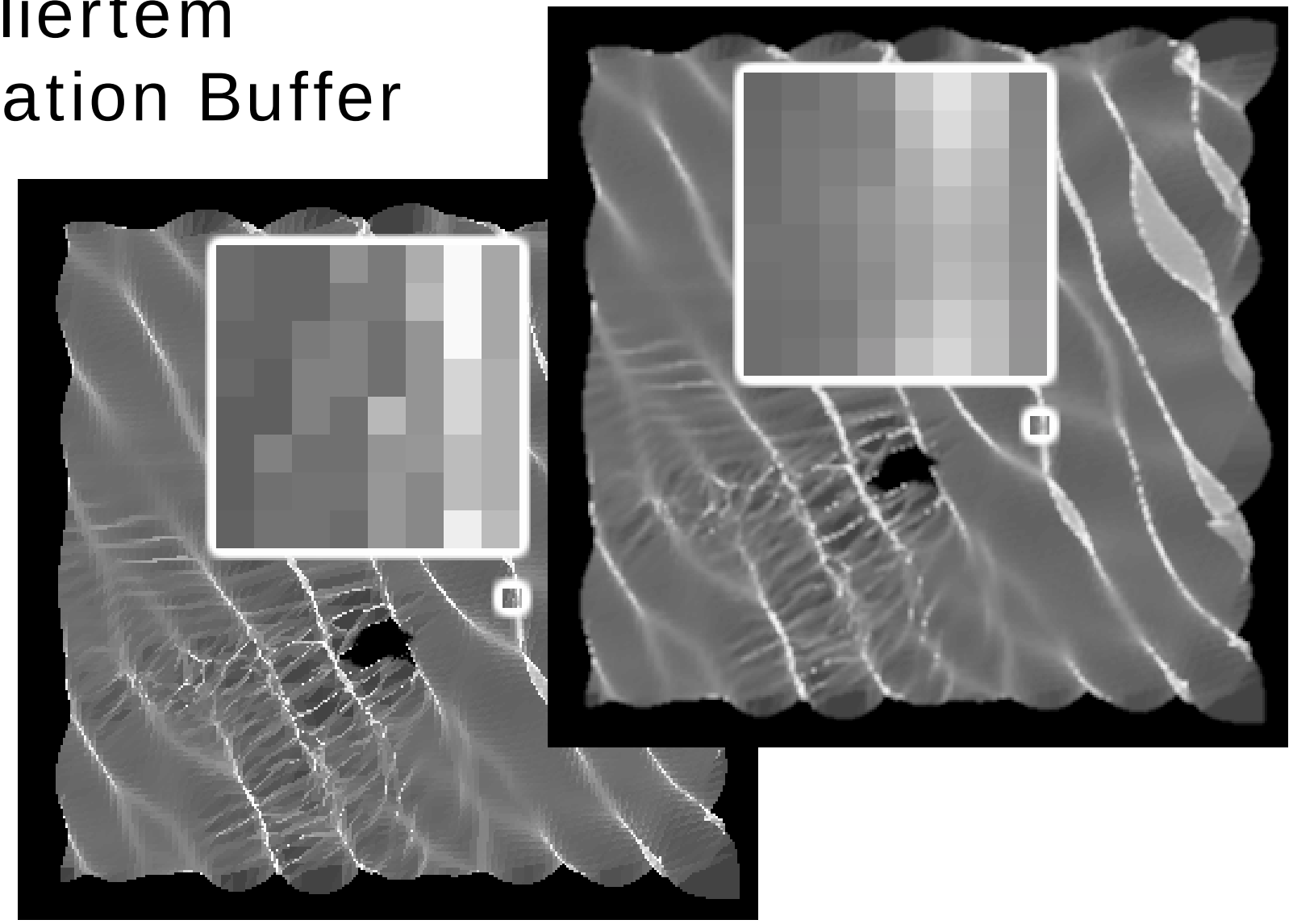
Grafikbuffer auslesen und per CPU glätten?
Dauert zu lange!

Glättung auf der Grafikkarte?
Selten Hardware-beschleunigter
Accumulation Buffer!

Accum. Buffer emulieren! (Pixel-Shader)

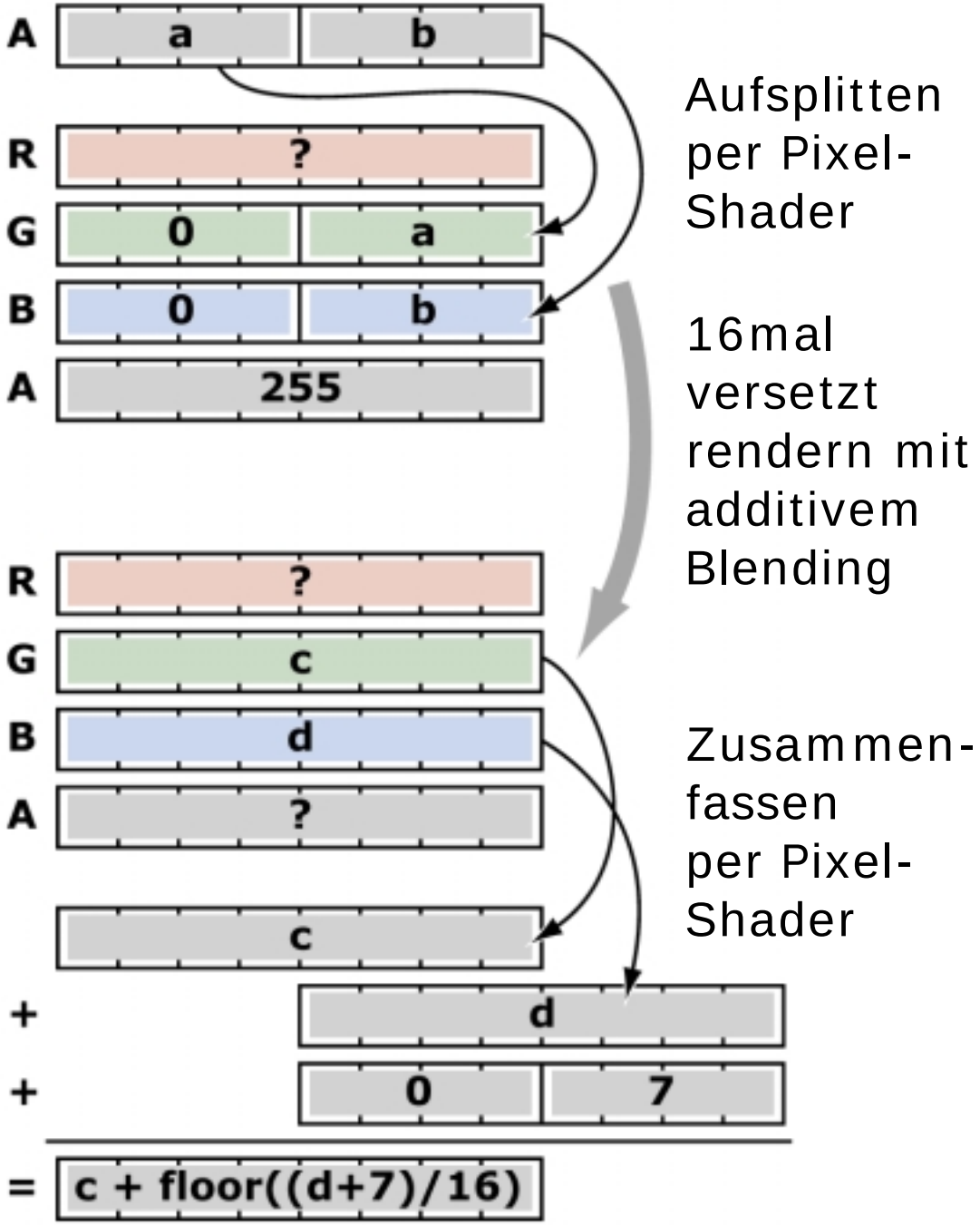
# Visualisierung: Kaustiken 4

Weichzeichnen
per emuliertem
Accumulation Buffer



# Visualisierung: Kaustiken 5

Weichzeichnen
per emuliertem
Accumulation Buffer



# Fazit

Leistung auf 2,5-GHz-Rechner:

| | |
|---|---|
| Wellenausbreitung (1/3 Schritt) | 52 ms |
| Schiffswellen erzeugen | 6 ms |
| Kaustiken erzeugen und glätten | 18 ms |
| Reflexionstexturen erzeugen | 51 ms |
| Wasserfläche rendern | 42 ms |
| **gesamt** | **170 ms = 6 fps** |

Mögliche Weiterentwicklung:

- Reflexionstexturen mit Pixel-Shadern und Texture-Combinern beschleunigen
- weitere nichtlineare Effekte