

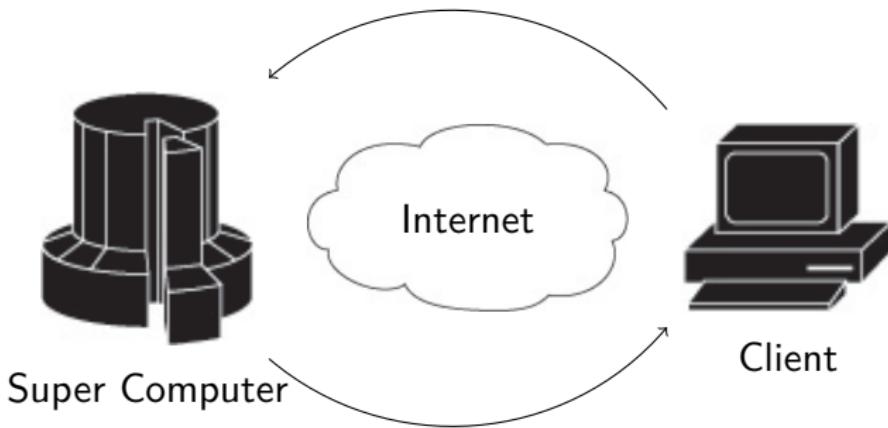
Optimierung und Übertragung von Tiefengeometrie für Remote-Visualisierung

Josef Schulz

Technische Universität Dresden

February 7, 2017

Motivation



Aufgabenstellung

Grundlegendes:

- Entwicklung eines Remote-Visualisierungssystems
 - Client basiert auf JavaScript, WebGL
 - Kommunikation findet über WebSocket-Protokoll statt
-
- Approximation der entstehenden Tiefenbilder durch Dreiecksnetze
 - Auswertung der Resultate in Abhängigkeit des Winkelunterschieds anhand von Ground-Truth-Szenen

Herausforderungen

- Anwendung der Approximation auf Partikelbasierte Datensätze
-

Gliederung

Motivation

Verwandte Arbeiten

Grundlagen

Szenen

Rückprojektion

Methoden

Ergebnisse

Verwandte Arbeiten

- THIN Systeme
- cluster
- 3D gaming —

TestSpheres

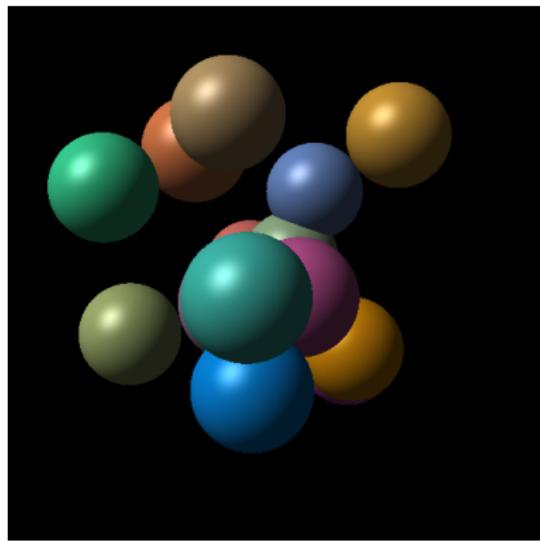


Abb. : Farbbild

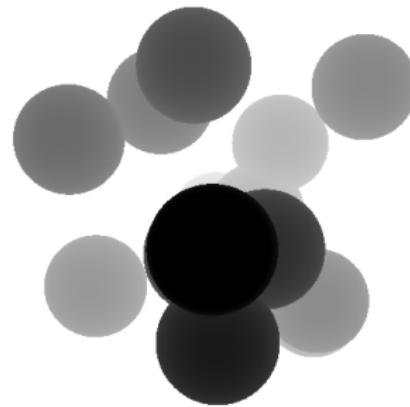


Abb. : Tiefenbild

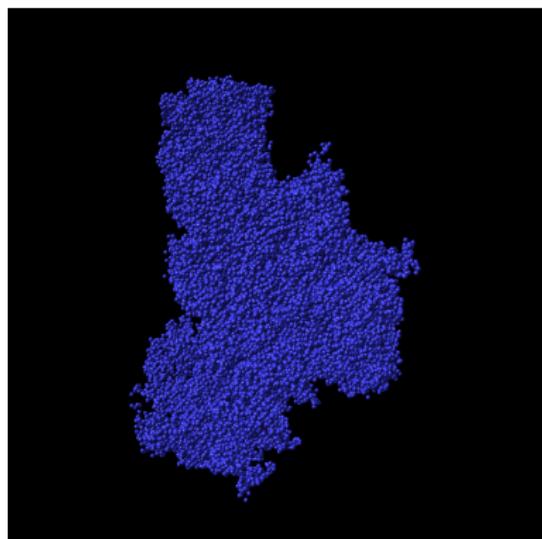


Abb. : Farbbild

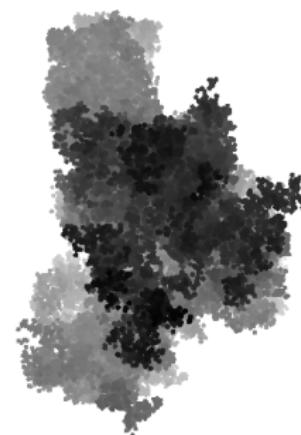
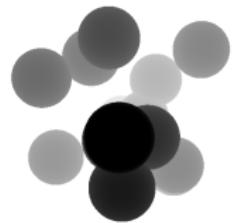


Abb. : Tiefenbild

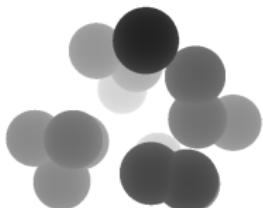
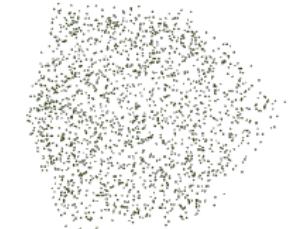
Aufteilung

#	Anzahl Bilder	min Winkel	max Winkel	Winkel Schritt
1	241	0	5	0.25
2	60	6	10	1
3	192	15	90	5

Rückprojektion einer Punktwolke



$$v' = (P \cdot V \cdot M)^{-1} \cdot v$$



$$v'' = (P_i \cdot V_i \cdot M_i) \cdot v'$$

Gradientenbilder

Die Gradientenbilder, wurden mit dem Sobel-Operator erzeugt:

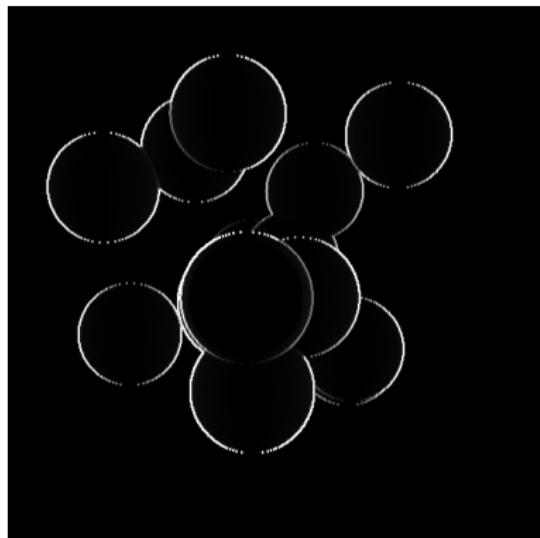


Abb. : $|\nabla_x|$

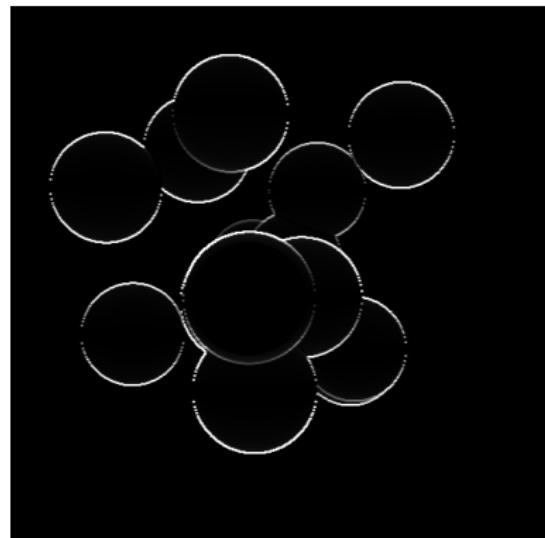
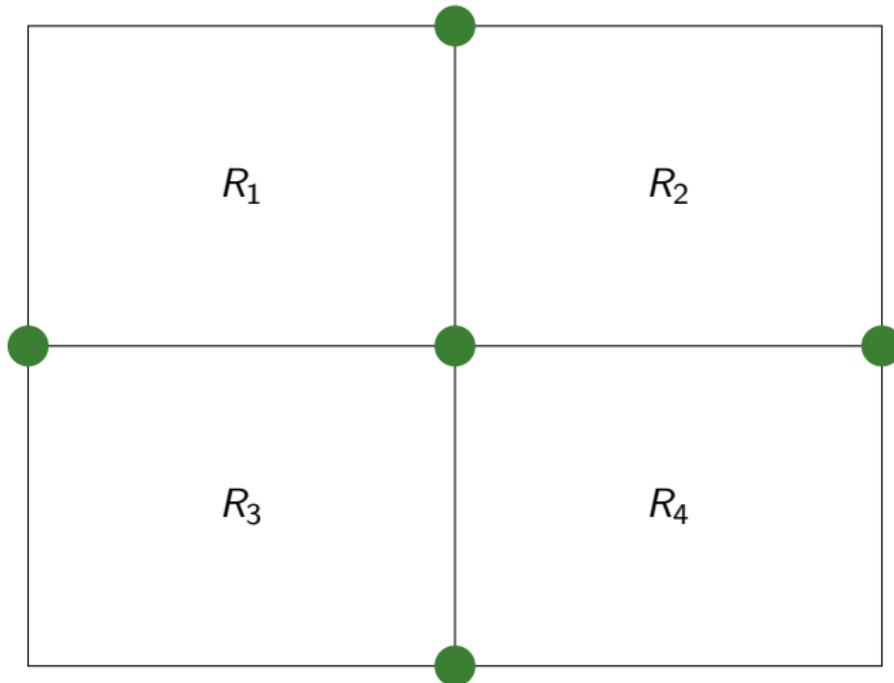


Abb. : $|\nabla_y|$

Quadtree-Ansatz

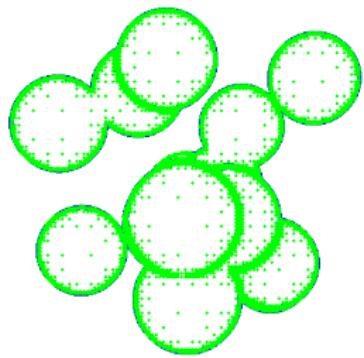


Quadtree: Finden der Seed-Points

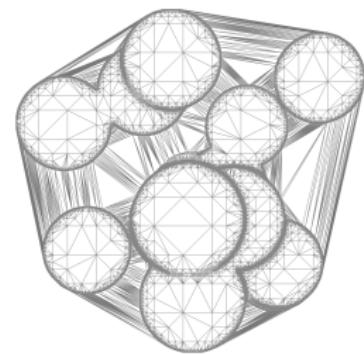
$$c_x = \max_R |\nabla_x| - \min_R |\nabla_x| \quad (1)$$

$$c_y = \max_R |\nabla_y| - \min_R |\nabla_y| \quad (2)$$

Punkte die zum Hintergrund gehören werden verworfen.



delaunay

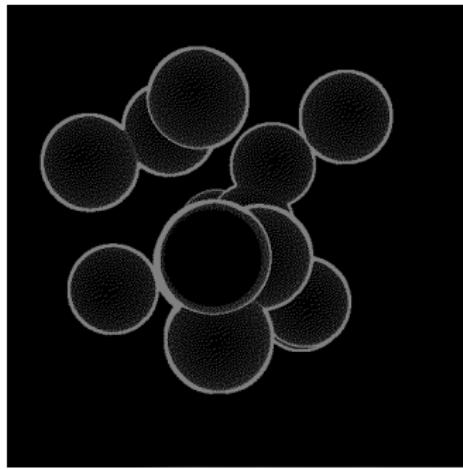


Erzeugung einer Feature-Map aus den Gradientenbildern:

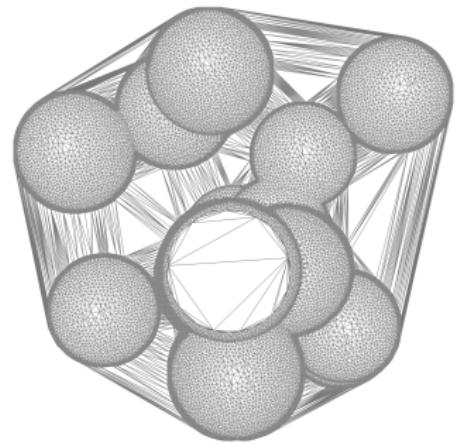
$$\sigma_p = \left(\frac{\|\nabla p\|}{A} \right)^\gamma. \quad (3)$$

```
1  for each y
2    for each x
3      oldpixel      := pixel[x][y]
4      newpixel       := (pixel[x][y] > δ) ? δ : 0
5      pixel[x][y]   := newpixel
6      quant_error   := oldpixel - newpixel
7      pixel[x+1][y ] := pixel[x+1][y ] + quant_error * 7 / 16
8      pixel[x-1][y+1] := pixel[x-1][y+1] + quant_error * 3 / 16
9      pixel[x ][y+1] := pixel[x ][y+1] + quant_error * 5 / 16
10     pixel[x+1][y+1] := pixel[x+1][y+1] + quant_error * 1 / 16
```

Listing 1: Floyd-Steinberg-Algorithmus



delaunay



- Wann ist eine Kante nicht valid
- Wann muss ein Dreieck nachbearbeitet werden.

Valide Kanten

p_1, p_2 - Eckpunkte der Kante

p_m - Mittelpunkt auf der Kante

Ist eine der folgenden Bedingungen nicht erfüllt wird die Kante als nicht valid bezeichnet:

$$\frac{d_{p_1} + d_{p_2}}{2} - d_{p_m} < \varepsilon \quad (4)$$

$$\frac{|d_{p_1} - d_{p_2}|}{\|p_1 - p_2\| (d_{p_1} + d_{p_2})} < \alpha \quad (5)$$

Enthält ein Dreieck zwei nicht valide Kanten, ist das Dreieck nicht valide und wird nachbearbeitet:

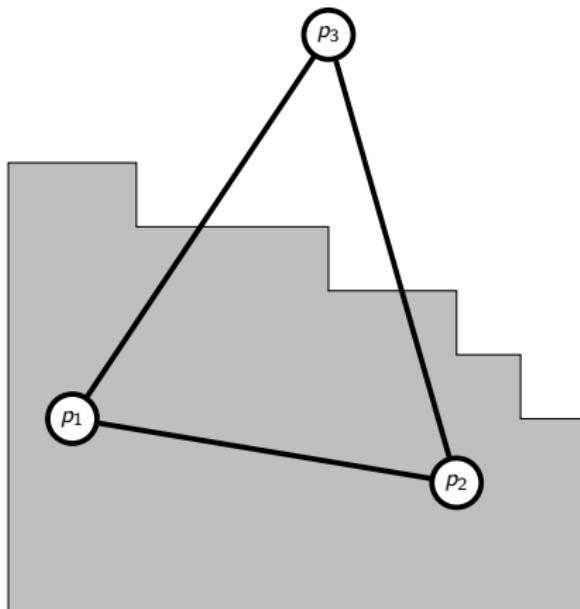


Abb. : 2 Kanten sind nicht valide

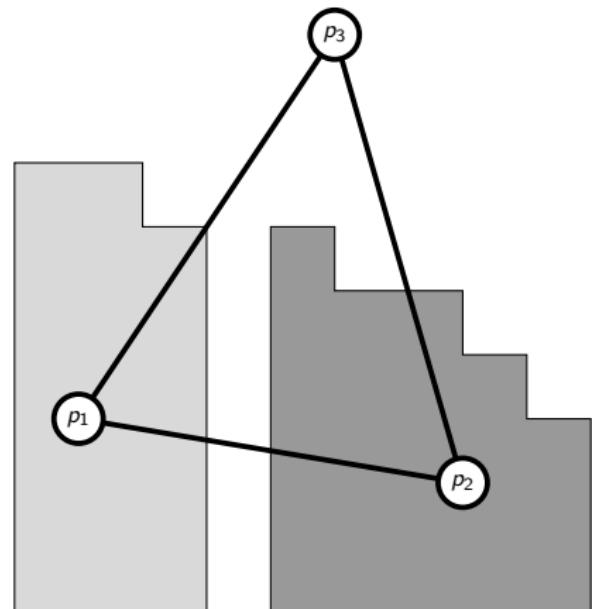
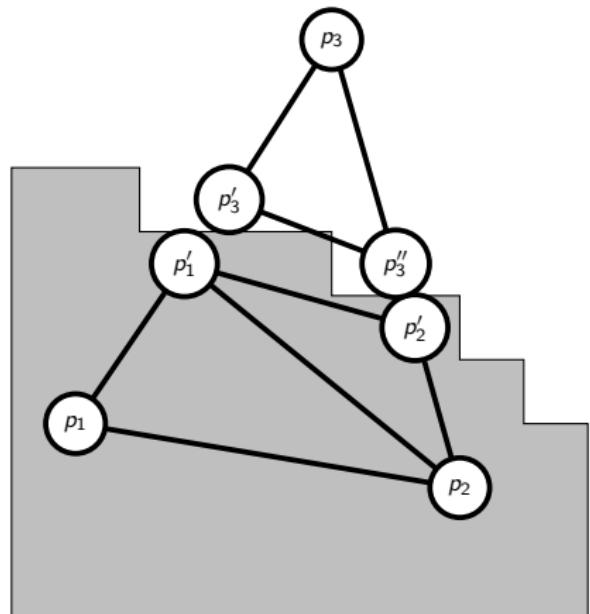
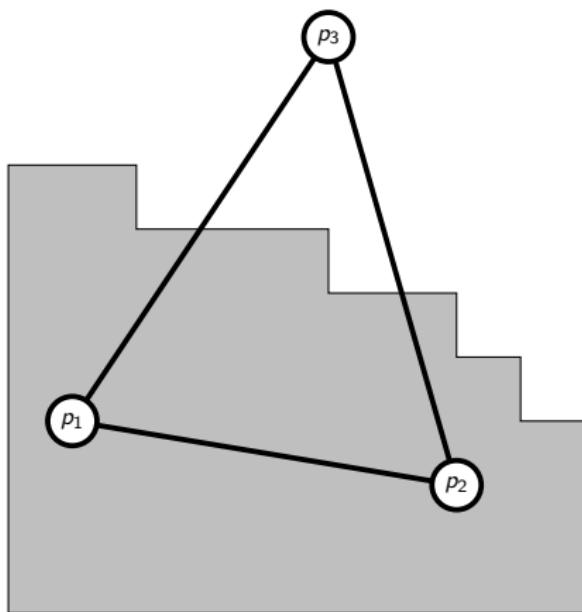
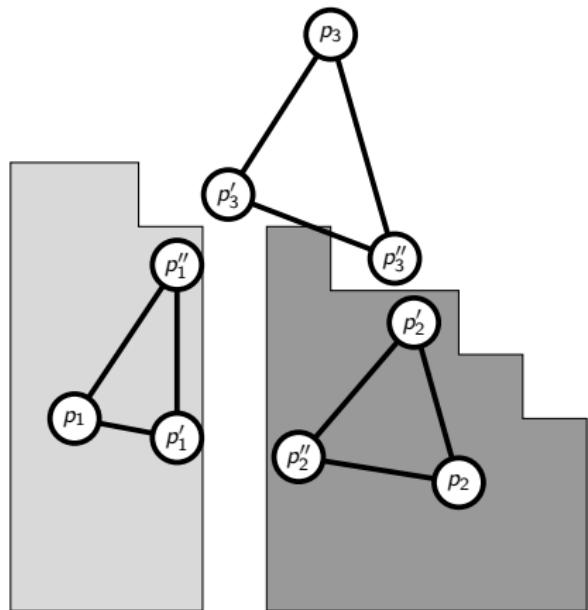
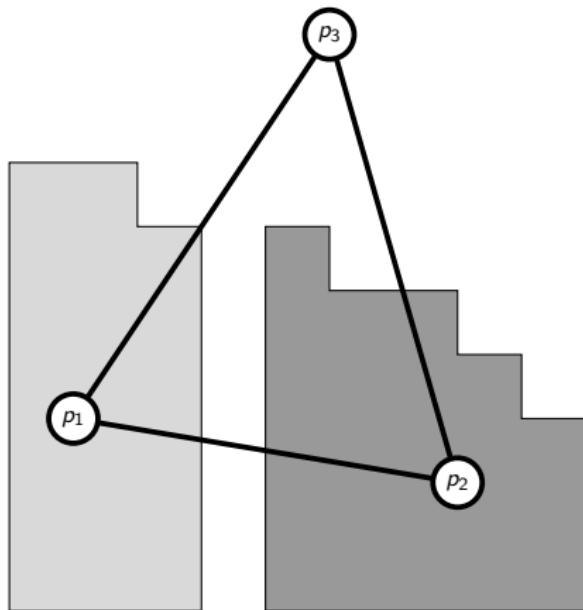


Abb. : 2 Kanten sind nicht valide

2 Kanten sind nicht Valide



3 Kanten sind nicht Valide



JPEG PSNR

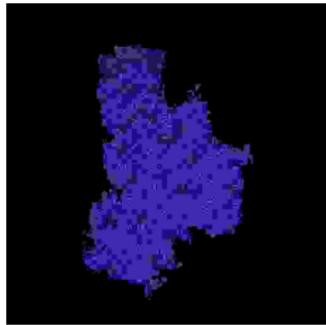


Abb. : 20,09 dB

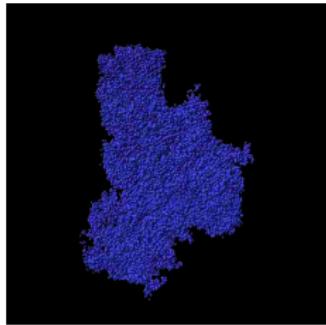


Abb. : 23,42 dB

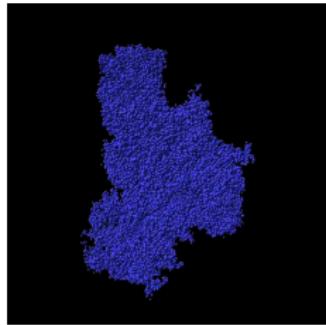
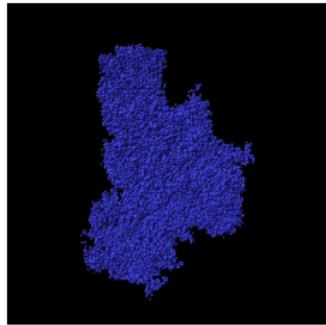
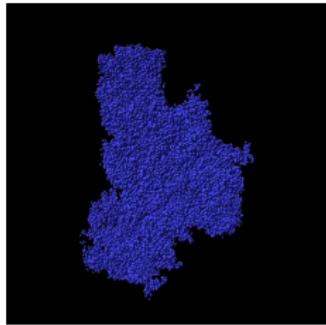
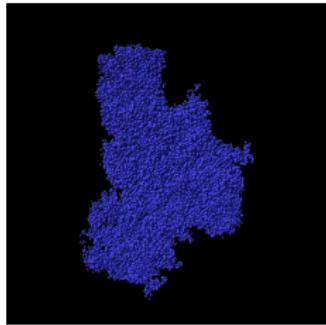


Abb. : 24,32 dB



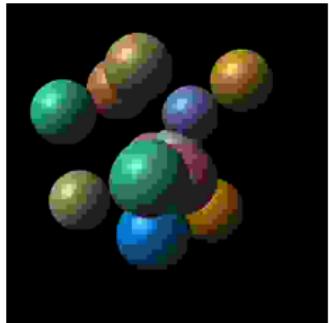


Abb. : 27,25 dB

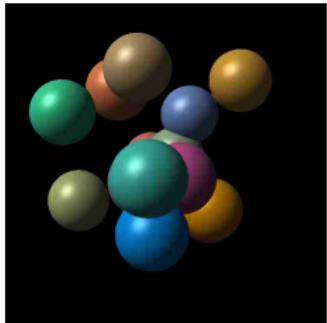


Abb. : 34,88 dB

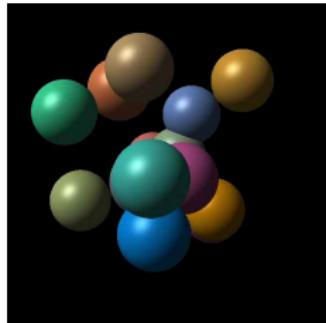


Abb. : 36,63 dB

