Architektur und Programmierung von Grafik- und Koprozessoren

Rendering Algorithmen

Stefan Zellmann

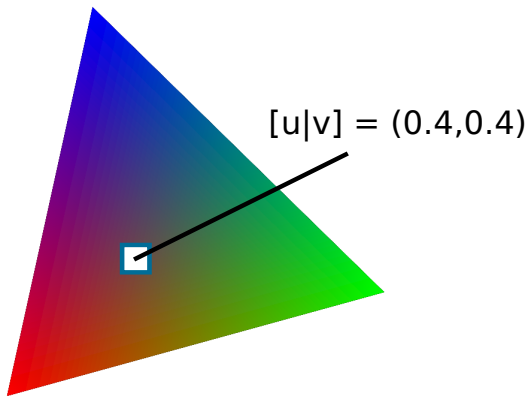
Lehrstuhl für Informatik, Universität zu Köln

SS2019

Texture Mapping

Lineare Interpolation beim Texture Mapping

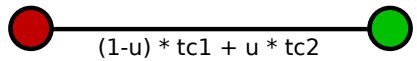
a.)



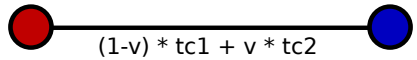
Barycentric Interpolation

b.)

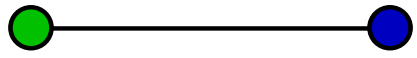
lerp(tc1,tc2, u)



lerp(tc1,tc3, v)

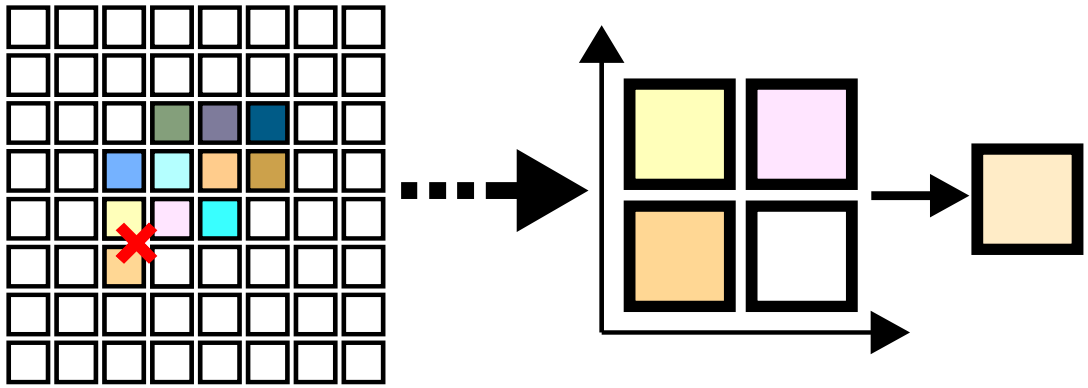


lerp(tc2,tc3, 1-u-v)



Linear Interpolation

c.)



Bilinear Interpolation

Texture Mapping

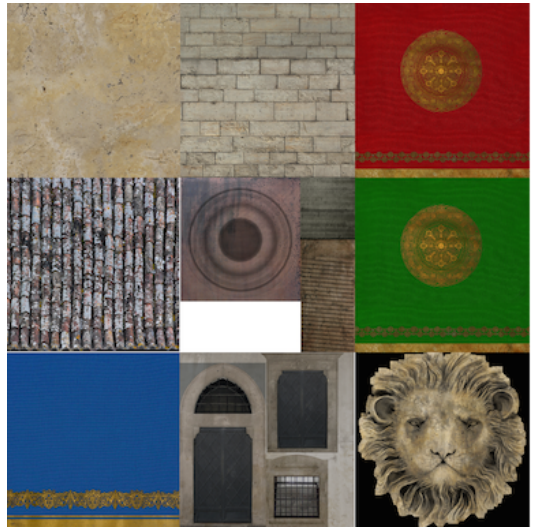
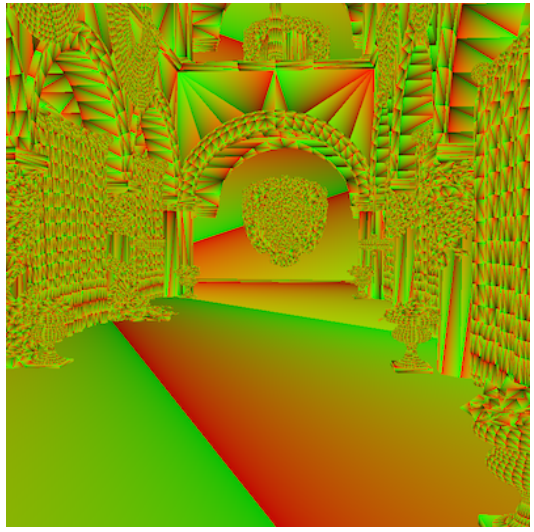
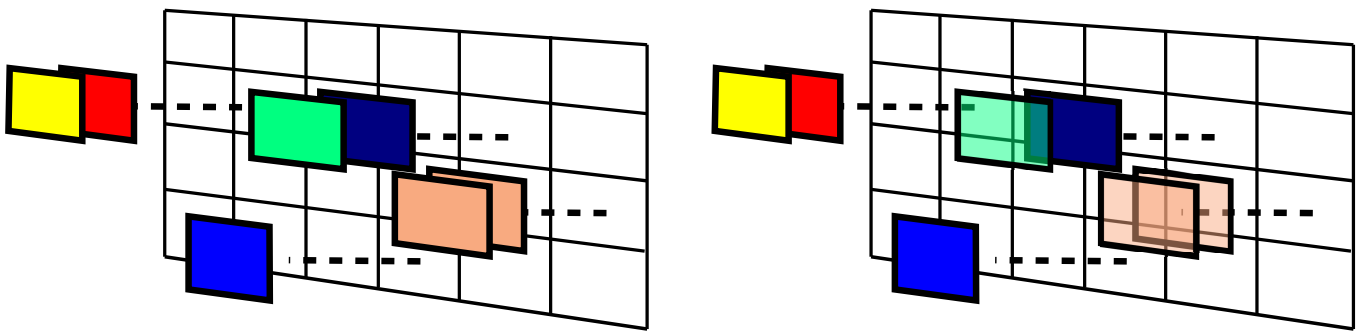





Tabelle: Crytek Sponza 3D Modell: Frank Meinl, (CC-BY 3.0), Rendering: Stefan Zellmann

Tiefentest

Rasterpunkte: *Tiefenpuffer* Datenstruktur speichert *Fragment* mit z-Koordinate am nächsten zum Betrachter.

Alpha Blending: teiltransparente Fragmente determinieren zusammen Farbe an Bildrasterposition (Pixel).



Alpha Blending

Teiltransparente Geometrie

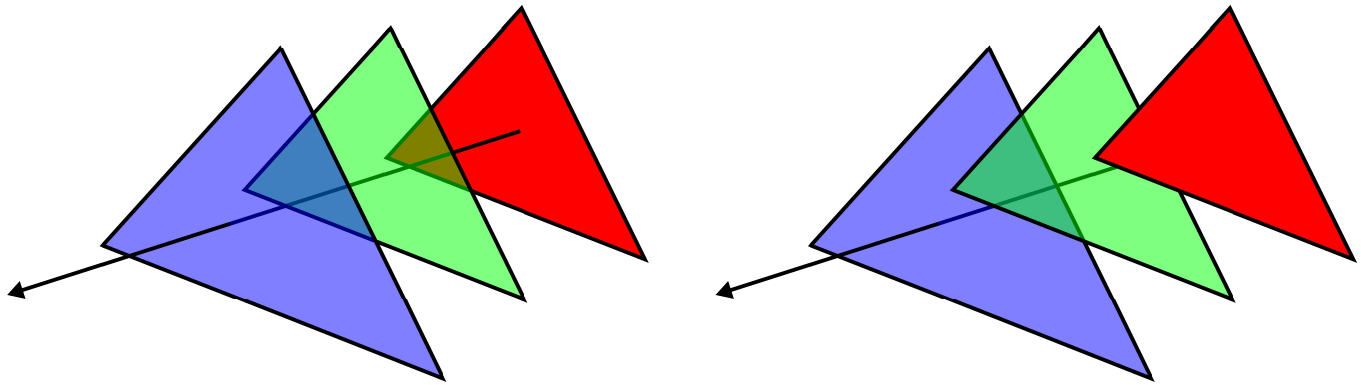


Abbildung: Teiltransparente Geometrie muss tiefenkorrekt gezeichnet werden, da *Alpha Blending* Operation nicht kommutativ.

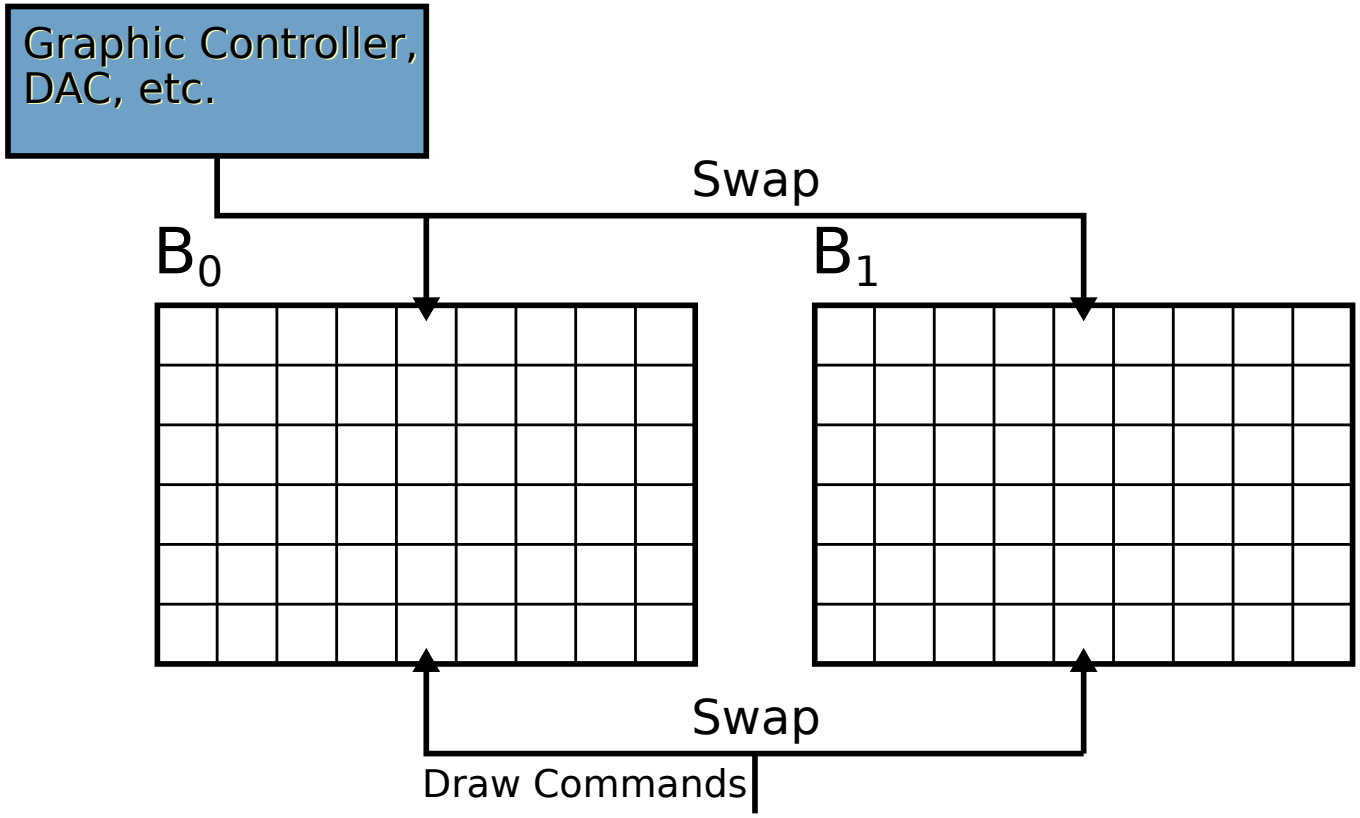
Alpha Blending

- Der einfachste Ansatz ist es, bei jeder Kameraänderung (nur) die teiltransparente Geometrie gemäß der Blickrichtung zu sortieren.
- Fragmente, die sich im weiteren Verlauf ergeben, *blendet* man z. B. durch Vormultiplikation der Eingangsfarbe mit Alpha und der *over* Operation:

$$c_{dst} = c_{src} + (1 - \alpha_{src})c_{dst}, \tag{3}$$

  wobei Subscript *src* und *dst* jeweils Eingangs- und Ausgangsfarbe bezeichnen.
- Weiterführende Methoden für tiefenkorrektes Alpha Blending: Depth Peeling, Linked Lists auf Fragment-Basis.

Double Buffering



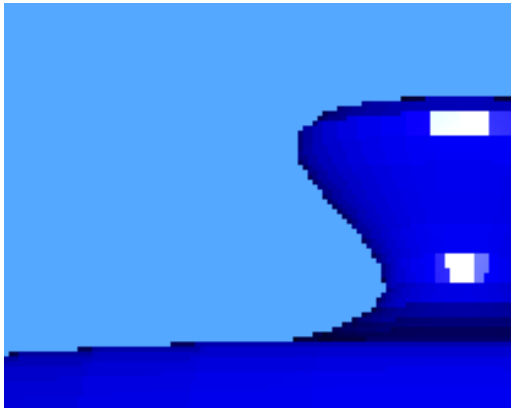
Aliasing und Sampling Theorie

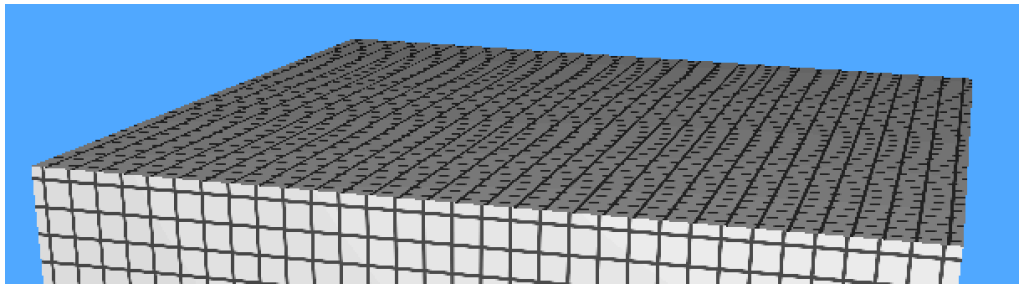
Aliasing und Sampling Theorie[1]

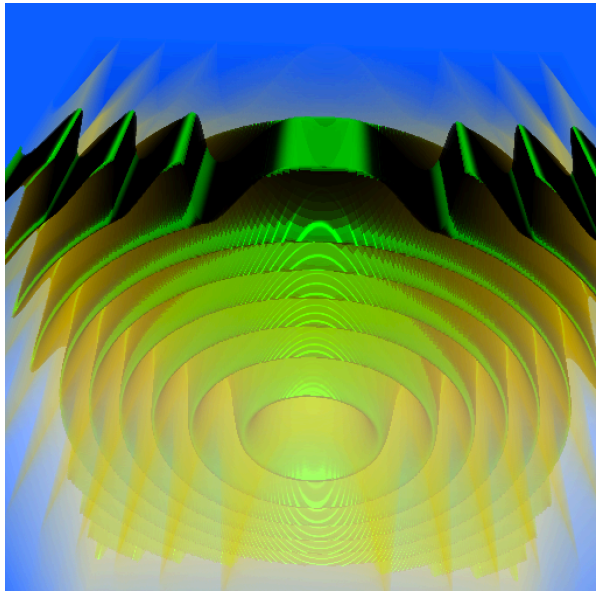
- *Aliasing* durch verschiedenste Quellen spielt in Computergrafik große Rolle.
- *Anti-Aliasing* integraler Teil von Hardware (GPUs).
  - Multi-Sample Anti-Aliasing.
  - Dedizierte Textur-Sampling Einheiten.
  - Spezialisierte Hardware für Mip-Mapping, anisotropisches Filtering etc.
- Daher *kurze* Einführung in Sampling Theorie, um zu verstehen, auf welchen Ebenen Anti-Aliasing nötig.

[1] vgl. z. B. Pharr, Jakob, Humphreys: Physically Based Rendering, 3rd ed. (2017) (`www.pbr-book.org`)

Aliasing Quellen







Aliasing Quellen

- Aliasing in der Bildebene.
- Aliasing im Texturraum.
- Aliasing bei spiegelnden Oberflächen.
- Zeitliches Aliasing.
- Aliasing durch zu niedrige Farbauflösung.
- ...

Dirac Delta Verteilung und Shah Funktion

Dirac Delta Funktion:

$$\int \delta(x)dx = \begin{cases} 1, & \text{if } x = 0 \\ 0 & \text{sonst} \end{cases} \tag{4}$$

Für $f(x)$ stetig differenzierbar folgt $\int f(x)\delta(x)dx = f(0)$.

Shah Funktion (a.k.a. "Impulse Train"): unendliche Summe äquidistanter Delta Funktionen:

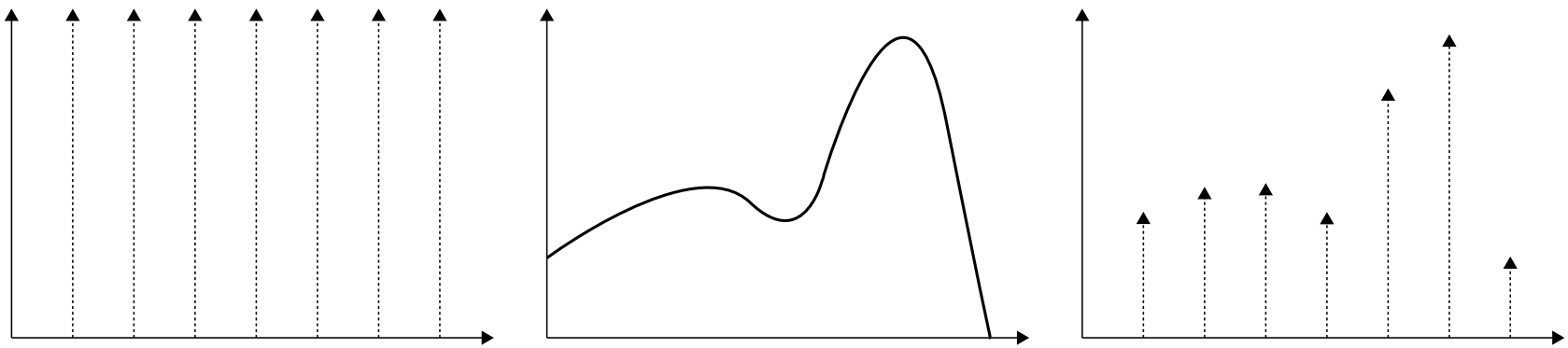
$$\text{Ш}_\Delta(x) = \Delta \sum_{i=-\infty}^{\infty} \delta(x - i\Delta), \tag{5}$$

wobei Δ die *Periode* der Shah Funktion.

Sampling und Rekonstruktion

Durch Multiplikation der stetigen, "zu samplenden" Funktion $f(x)$ mit der Shah Funktion ergibt sich eine unendliche Sequenz von *Sample Punkten*:

$$\text{Ш}_\Delta(x)f(x) = \Delta \sum_{i=-\infty}^{\infty} \delta(x - i\Delta)f(i\Delta). \tag{6}$$



Sampling und Rekonstruktion

Erhalte $\bar{f}(x)$ durch *Faltung* mit *Rekonstruktionsfilterfunktion* $r(x)$:

$$\bar{f}(x) = (\text{Ш}_\Delta(x) f(x)) * r(x), \tag{7}$$

wobei die Faltung zweier Funktionen $f(x)$ und $g(x)$ definiert ist als

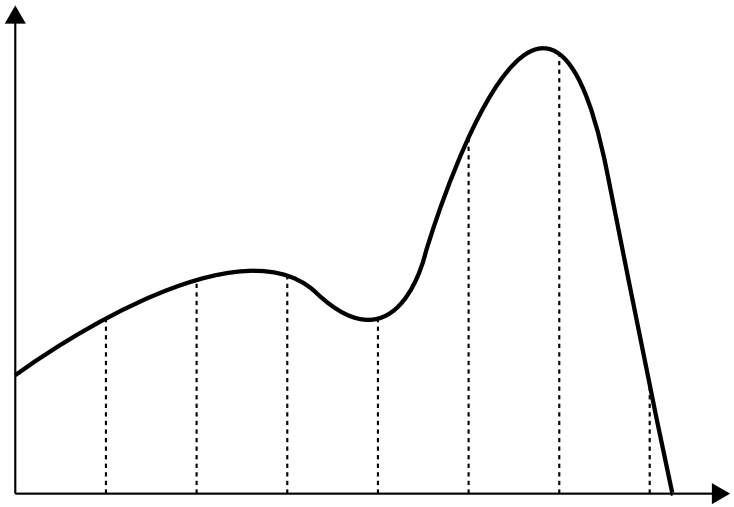
$$f(x) * g(x) = \int_{-\infty}^{\infty} f(\tau) g(x - \tau) d\tau. \tag{8}$$

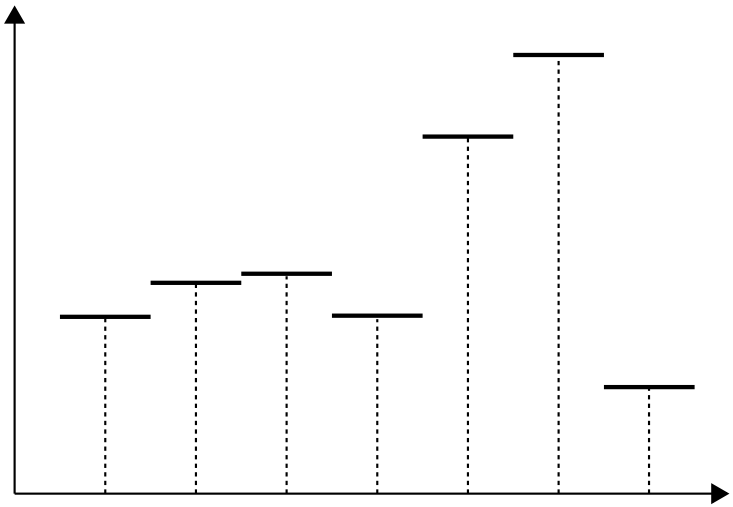
Bei der Faltung mit dem Rekonstruktionsfilter ergibt sich die gewichtete Summe:

$$\bar{f}(x) = \Delta \sum_{i=-\infty}^{\infty} f(i\Delta) r(x - i\Delta). \tag{9}$$

Sampling und Rekonstruktion

Rekonstruktion mit Box Filter

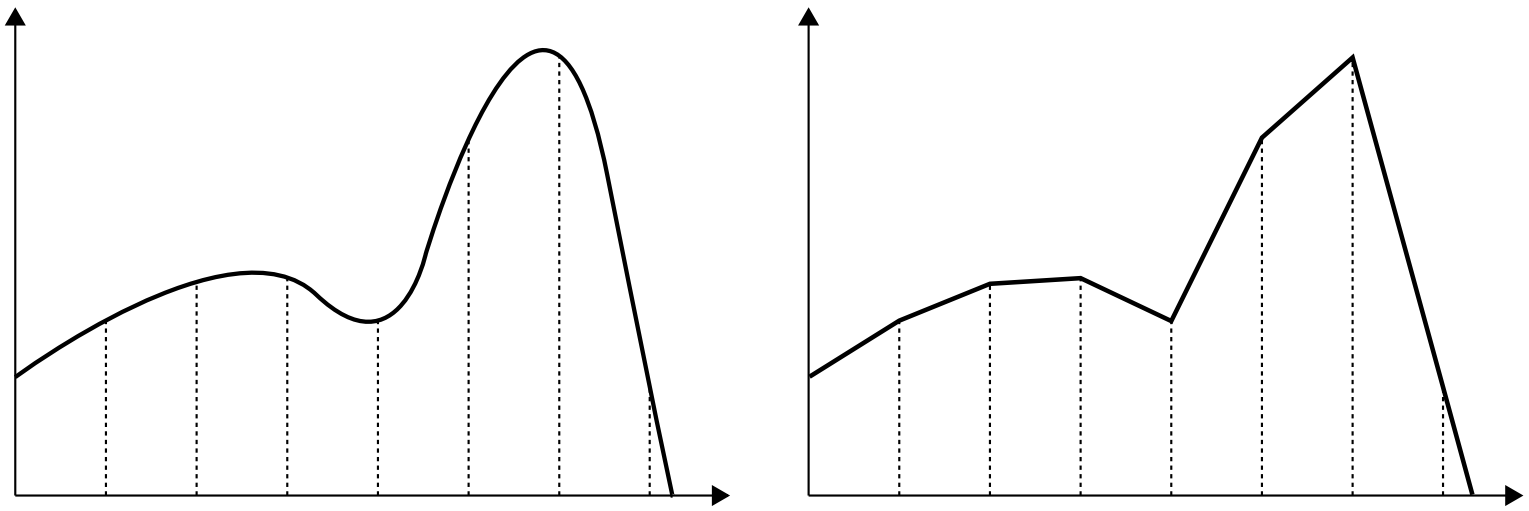




$$r(x) = \begin{cases} 1, & \text{if } |x| < \frac{1}{2} \\ 0 & \text{sonst} \end{cases} . \tag{10}$$

Sampling und Rekonstruktion

Stückweise lineare Rekonstruktion (Tent Filter)



$$r(x) = max(0, 1 - |x|) \tag{11}$$

Rekonstruktion

Exakt rekonstruierbare Signalfunktionen

Es gibt eine Klasse von Funktionen (*bandlimitierte* Funktionen), die, wenn man eine entsprechende Sampling Rate wählt, *exakt* aus der Sequenz von Sample Punkten rekonstruiert werden kann.

Fourier Transformation

$f(x)$ hat Repräsentation im *Ortsraum* sowie im *Frequenzraum*.

Fourier Transformation: Ortsraum $\Rightarrow$ Frequenzraum:

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\omega x} dx, \tag{12}$$

inverse Fourier Transformation: Frequenzraum $\Rightarrow$ Ortsraum:

$$f(x) = \int_{-\infty}^{\infty} F(\omega) e^{i2\pi\omega x} d\omega. \tag{13}$$

(Dabei sind $e^{ix} = cos(x) + isin(x)$ und $i = \sqrt{-1}$.)
Idee: *Signale* lassen sich im Limit als gewichtete Summe von Sinusoiden repräsentieren.

Fourier Transformation

Orts- und Frequenzrepräsentation wichtiger Funktionen

Ortsraum	**Frequenzraum**
Konstante: $f(x) = 1$	Dirac Delta: $F(\omega) = \delta(\omega)$
Box: $f(x) = 1 \begin{cases} \text{if} & \lvert x \rvert < \frac{1}{2}, \\ 0 & \text{sonst} \end{cases}$	Sinc: $F(\omega) = \frac{sin(\pi\omega)}{\pi\omega}$
Shah: $f(x) = \Delta \sum_{i=-\infty}^{\infty} \delta(x - \Delta i)$	Shah: $F(\omega) = \frac{1}{\Delta} \sum_{i=-\infty}^{\infty} \delta(\omega - \frac{1}{\Delta})$

Tabelle: vgl. Pharr, Jakob, Humphreys: Physically Based Rendering, 3rd ed. (2017), p. 405.

Fourier Transformation

Wichtige Eigenschaft der Fourier Transformation:
Faltung in Ortsraum entspricht Multiplikation in Frequenzraum, und umgekehrt:

$$f(x)g(x) = F(\omega) * G(\omega), \tag{14}$$

sowie

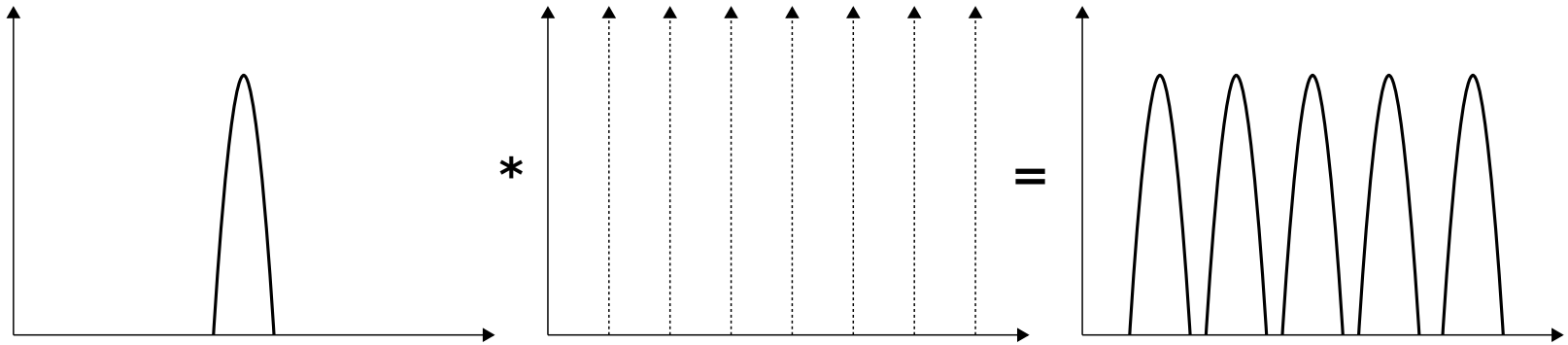
$$F(x)G(x) = f(\omega) * g(\omega). \tag{15}$$

Das Produkt von weiter oben: $\mathrm{III}_\Delta(x)f(x)$ entspricht also einer *Faltung* mit Periode $\frac{1}{\Delta}$ im Frequenzraum:

$$\mathrm{III}_\Delta(x)f(x) = \mathrm{III}_{\frac{1}{\Delta}}(\omega) * F(\omega). \tag{16}$$

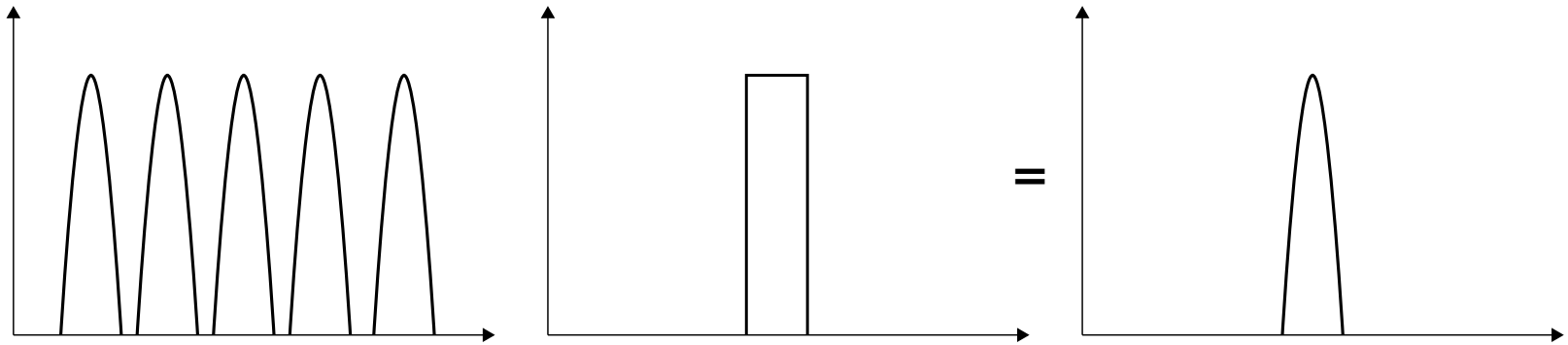
Bandlimitierte Signale

Faltung mit Shah Funktion in Frequenzraum: Signal wird unendlich oft *repliziert*, mit Periode der Shah Funktion.



Bandlimitierte Signale

Multiplikation mit Box Filter in Frequenzraum, um alle außer einer Kopie zu verwerfen.



Box mit Breite proportional zur Sampling Rate Δ:

$$r(x) = \begin{cases} \frac{1}{2\Delta} & |x| < \Delta \\ 0 & \text{sonst} \end{cases}. \tag{17}$$

Bandlimitierte Signale

Im Frequenzraum entspricht dieser Vorgang wegen o. g. Eigenschaften der Multiplikation mit der Shah Funktion $\mathrm{III}_\Delta(x)$ sowie einer Faltung mit dem *sinc* Rekonstruktionsfilter.

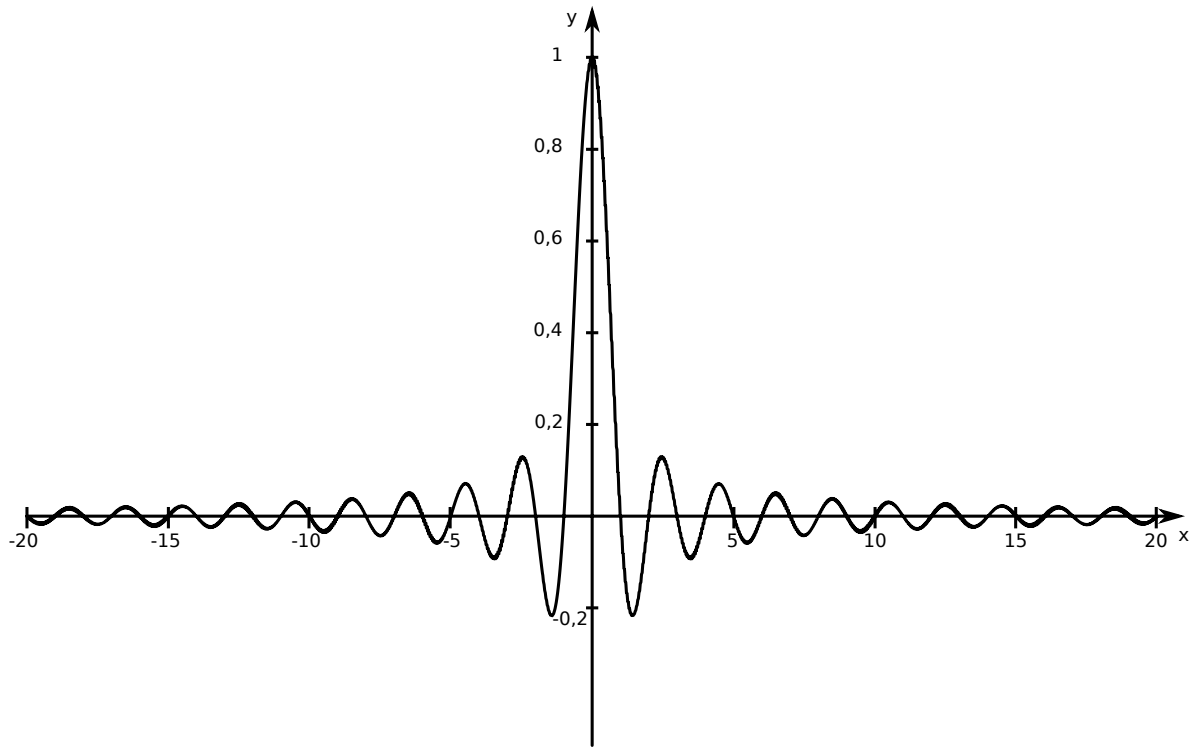


Abbildung: sinc Filter im Interval $[-20..20]$, vgl. Yvonne Percan, Untersuchung und Klassifikation des Fehlerverhaltens bei Direktem Volume Rendering, Diplomarbeit (2014)

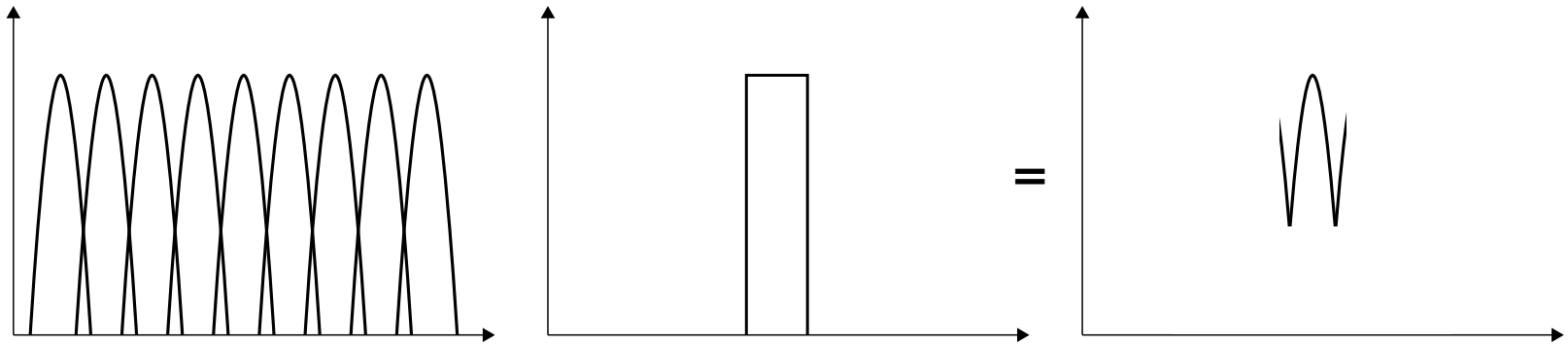
Bandlimitierte Signale

Problem bei dieser Betrachtung:

- *sinc* Filter hat unendliche Ausdehnung.
- Signal muss bandlimitiert sein.

Bandlimitierte Signale

Nicht-bandlimitierte Signale: Replikationen “überlappen”, es wird ein Teil einer anderen Frequenz mit rekonstruiert.



Frequenz *gibt sich als andere Frequenz aus* $\Rightarrow$ *Aliasing*.

Anti-Aliasing im Bildraum

- Nicht-bandlimitierte Signale $\Rightarrow$ niemals exakt rekonstruierbar. Aliasing, egal wie hoch die Sampling Rate.
- Sehr viele Samples: erhöhte Rechenzeit, Floating-Point Rundungsungenauigkeiten etc.
- Platziere so viele Samples, sodass optimal bzgl. *Konvergenz*.

Anti-Aliasing im Bildraum

Menschliches Auge weniger empfindlich ggü. Rauschen. Weiche Schatten: links: 16 uniforme Samples, rechts: 16 zufällige Samples.

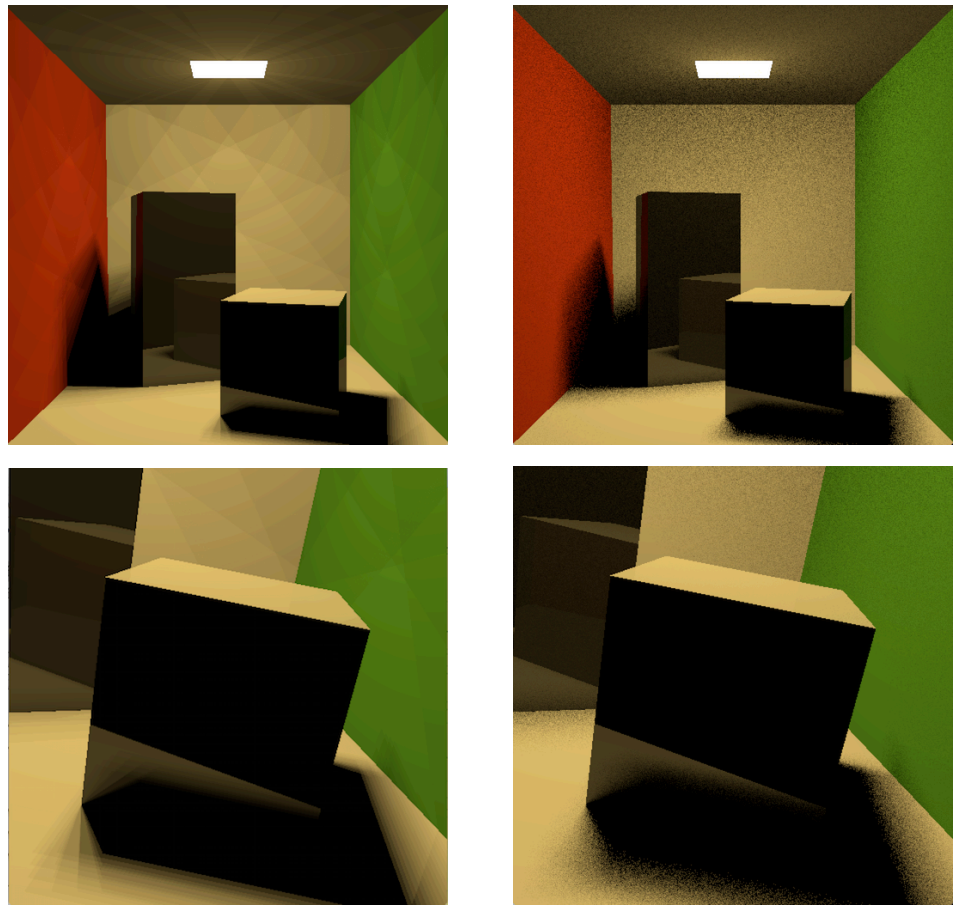


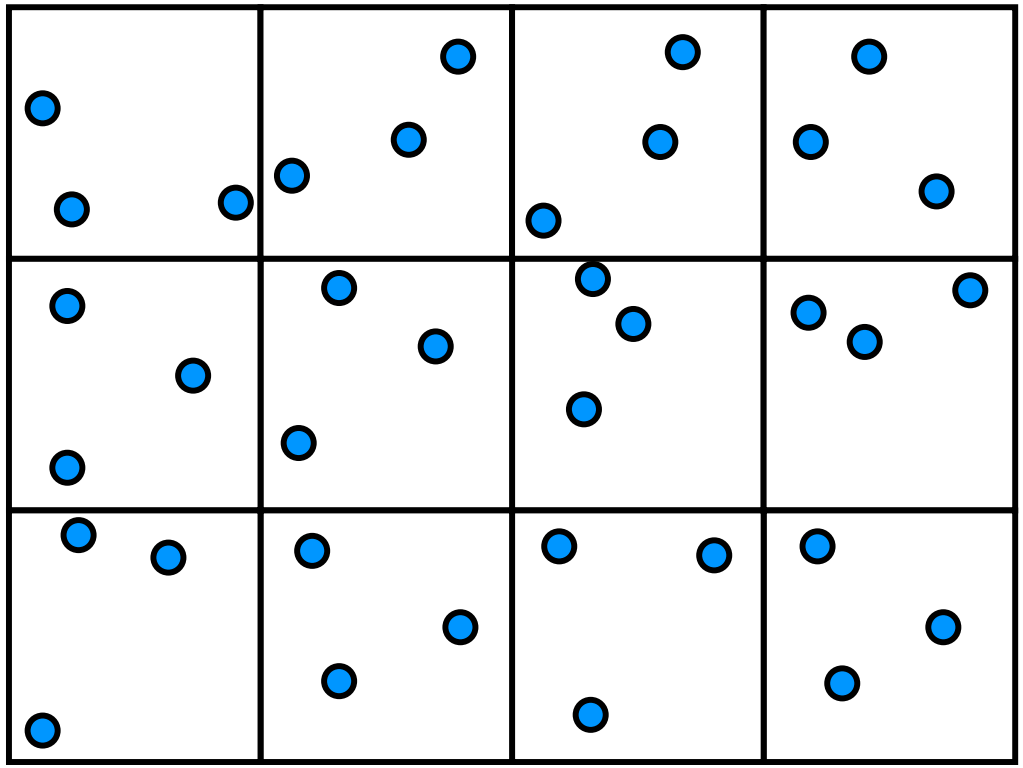
Tabelle: Cornell Box (https://www.graphics.cornell.edu/online/box/), Rendering: Stefan Zellmann

Anti-Aliasing im Bildraum

- Punkt-Samples an *diskreten* Stellen auf der Bildebene.
  - Pixel haben keine Fläche!
- Idee: Mehrere Sample “um Pixel herum”, gewichte jedes Sample mit Rekonstruktionsfilter (z. B. Tent oder Funktion höherer Ordung wie B-Spline).

Anti-Aliasing im Bildraum

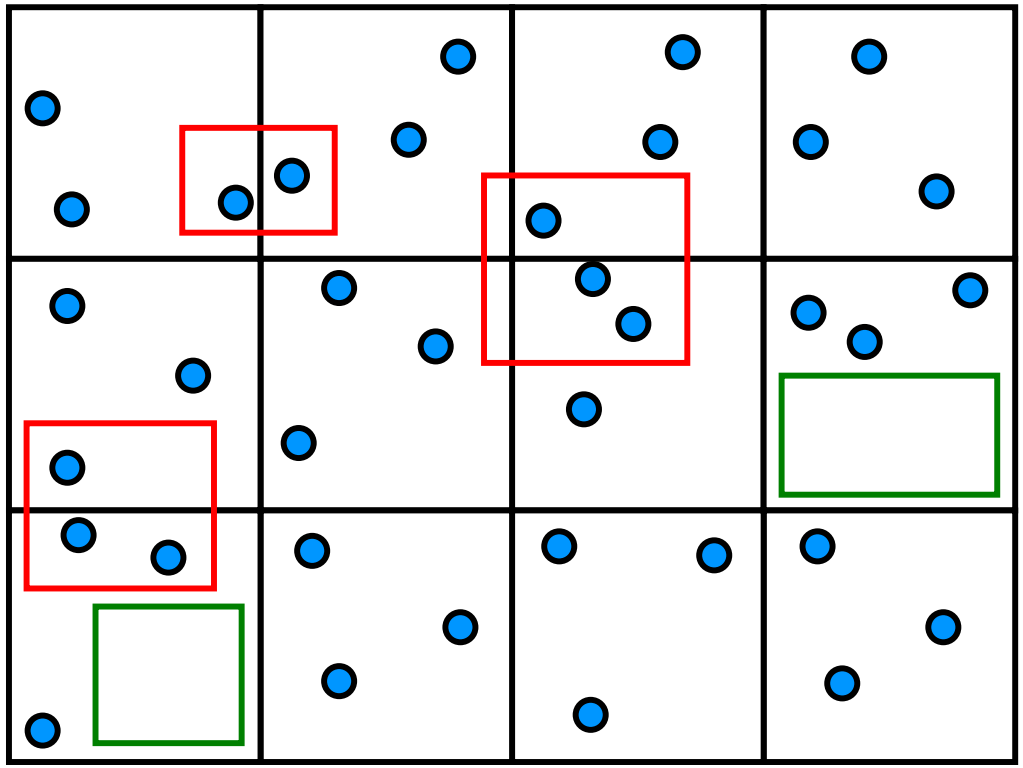
Jittered Sampling



Sample um Pixelposition herum, verschiebe um zufällig uniform verteiltes $\delta \in [0..\frac{1}{2})$.

Anti-Aliasing im Bildraum

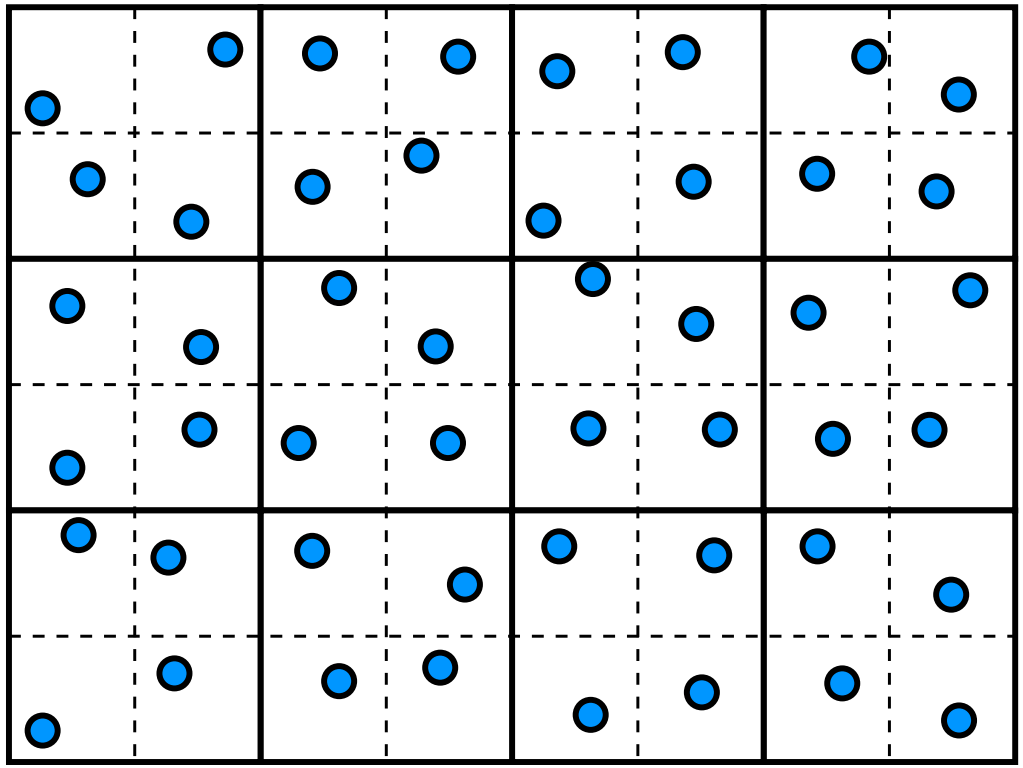
Jittered Sampling



Problem: Cluster-Bildung und unterrepräsentierte Bildregionen.

Anti-Aliasing im Bildraum

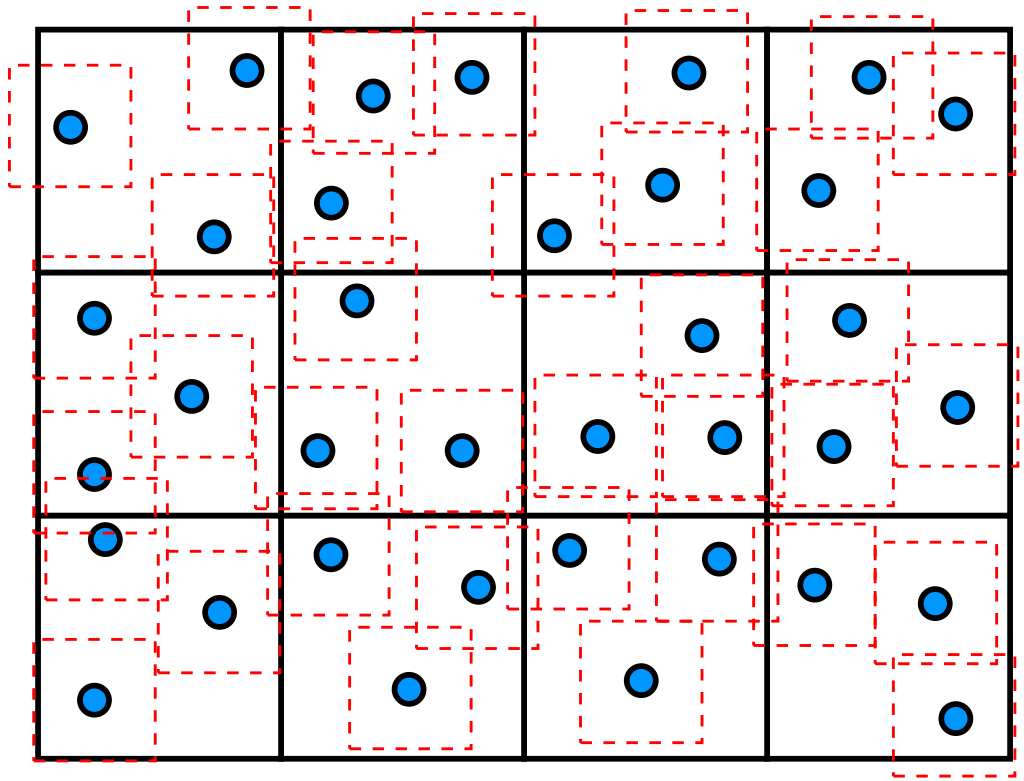
Stratified Sampling



Eine Lösungsstrategie: teile Pixel in *Strata* ein, sample innerhalb der Strata. Bessere Verteilung, jedoch andere Probleme (z. B. fixe Dimensionalität der Strata).

Anti-Aliasing im Bildraum

Low-Discrepancy Sampling



Informell: finde Sampling Positionen, die Überlapp der roten Boxen minimieren.

Anti-Aliasing im Bildraum

Diskrepanz

Sei B Familie von Polyedern b_i in $[0,1)^n$ (Linien, Rechtecke, Boxen mit Ursprung bei 0 und Kantenlänge < 1). Sei P eine Menge von N Sample Punkten innerhalb des Polyeders $[0,1)^n$.

Die Diskrepanz von P bzgl. B ist dann:

$$D(P,B) = \sup_{b \in B} \left| \frac{N_b}{N} - V(b) \right|, \tag{18}$$

wobei N_b die Anzahl Sample Punkte in b und $V(b)$ das Volumen von b.

Anti-Aliasing im Bildraum

Diskrepanz

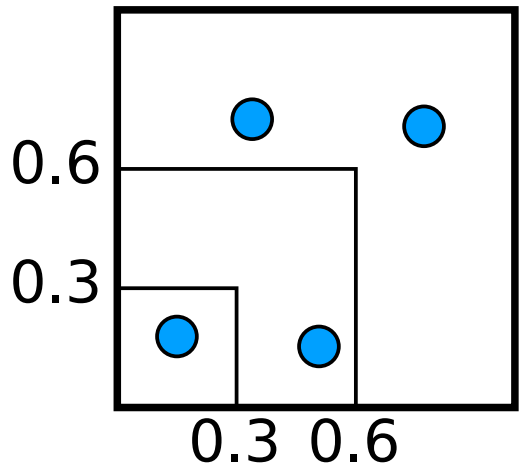


Abbildung: vgl. Pharr, Jakob, Humphreys: Physically Based Rendering, 3rd ed. (2017)

Punkte in $[0..1)^2$. Die Diskrepanz in der kleinen Box ($b_1 = [0, 0.3)$) beträgt $D = 0.25 - 0.09 = 0.16$, die Diskrepanz in der größeren Box ($b_2 = [0, 0.6)$) beträgt $D = 0.5 - 0.36 = 0.14$. Diskrepanz bzgl. P in $B = \{b_1, b_2, ..\}$ ist Maximum über alle D.

Anti-Aliasing im Bildraum

Bemerkung zur Diskrepanz

Offensichtlich haben uniform verteilte Sample Punkt Sequenzen die niedrigste Diskrepanz. Wir suchen Sequenzen mit niedriger Diskrepanz, die keine uniformen Sampling Artefakte mit sich bringen.

Anti-Aliasing im Bildraum

Low-Discrepancy Sequenzen

- “Quasi-Zufallszahlen”, entsprechen regelmäßigem Bildungsgesetz.
- Weiterführend. Beispiele sind:
  - Halton Sequenz.
  - Sobol Sequenz.

Multisample Anti-Aliasing auf GPUs

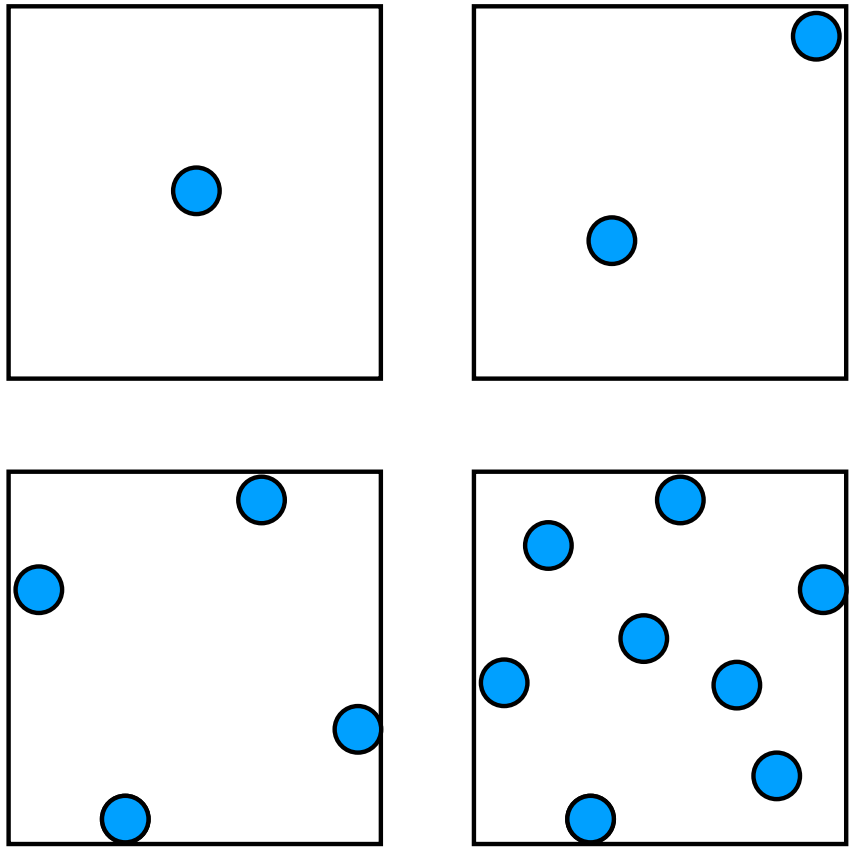


Abbildung: vereinfacht gemäß: EQAA Modes for AMD 6900 Series Graphics Cards, AMD Developer Relations (2011)

Multisample Anti-Aliasing auf GPUs

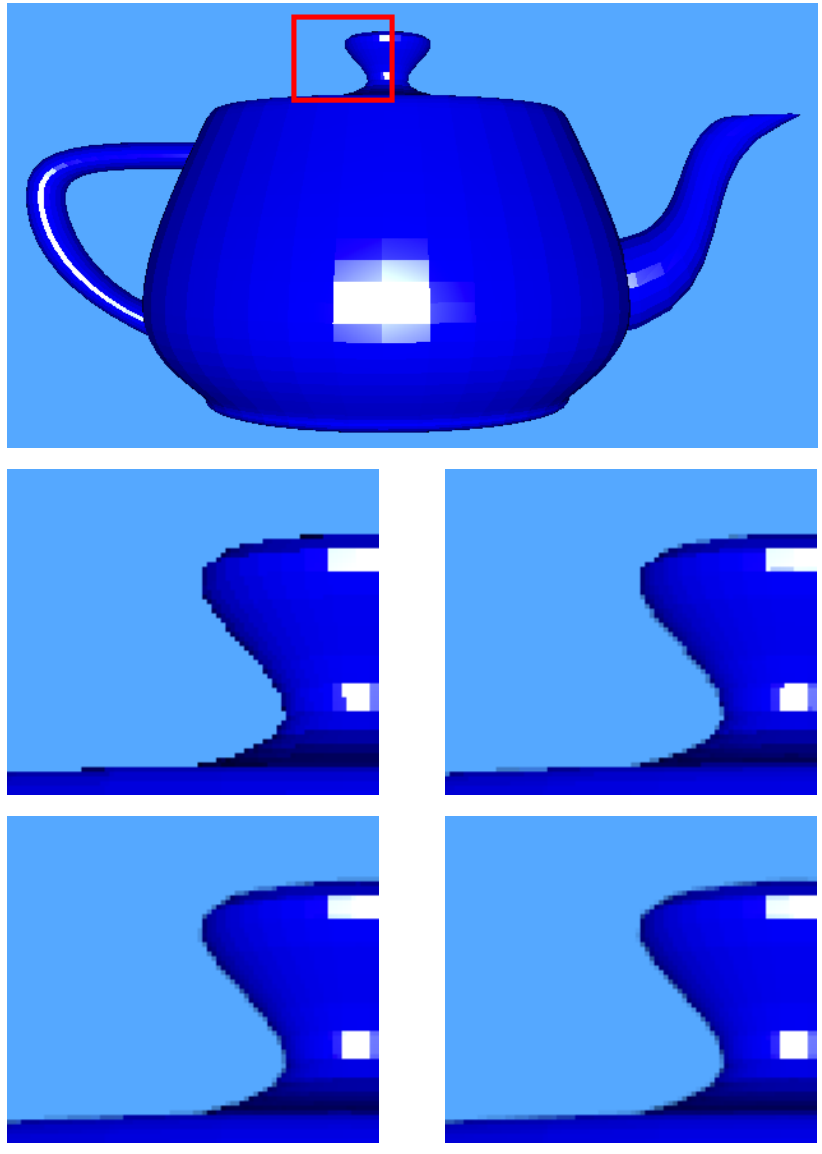
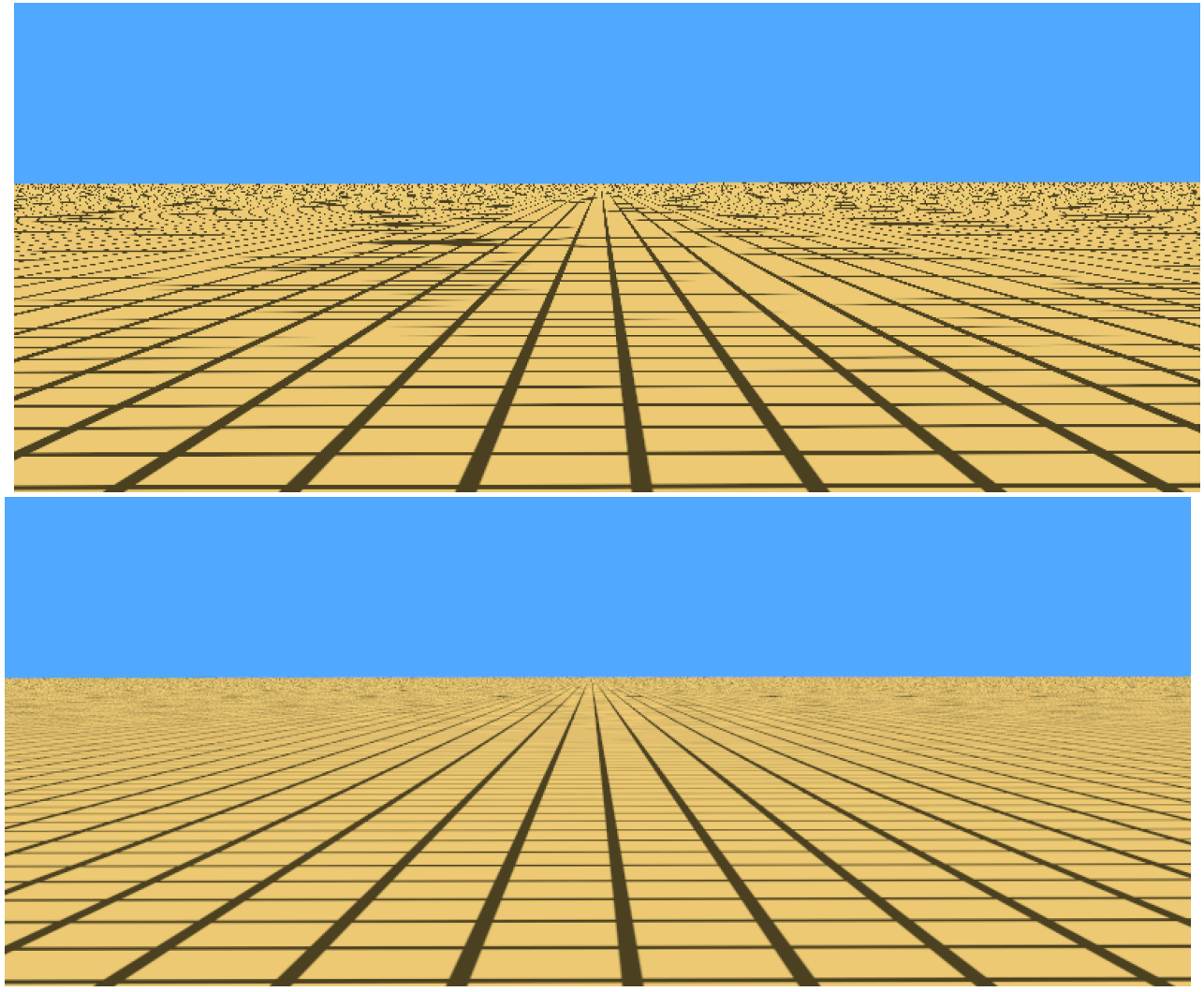


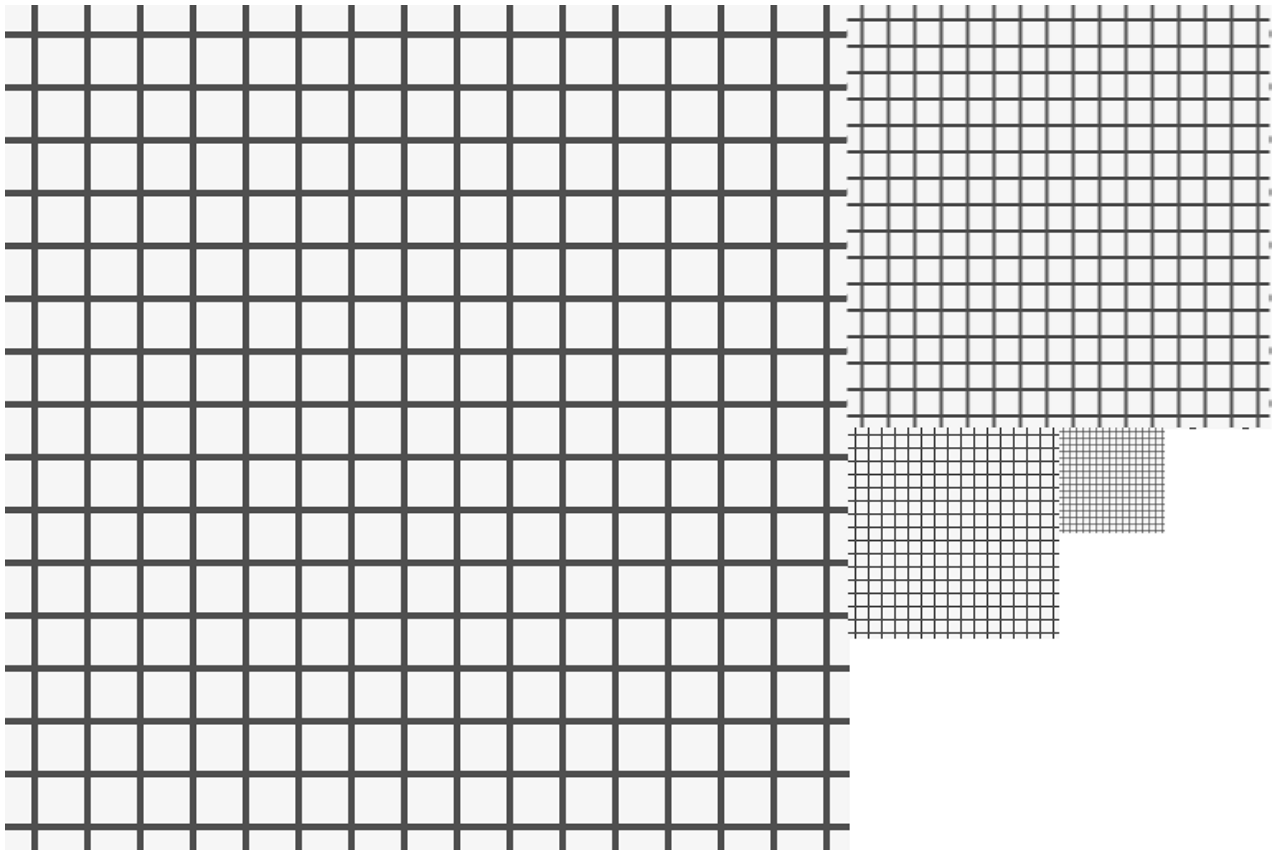
Tabelle: 1x, 2x, 4x und 8x Multisample Antialiasing (MSAA)

Textur Sampling auf GPUs



Textur Sampling auf GPUs

Mip-Mapping



Textur Sampling auf GPUs

Texture Derivatives

GPU *Raster Engines* wählen Mip-Map Level aufgrund der z-Distanz benachbarter Rasterpunkte. 1^{st}-order Derivatives $\Rightarrow$ Raster Engines verarbeiten 2×2 Bildschirmregionen ("Quads").