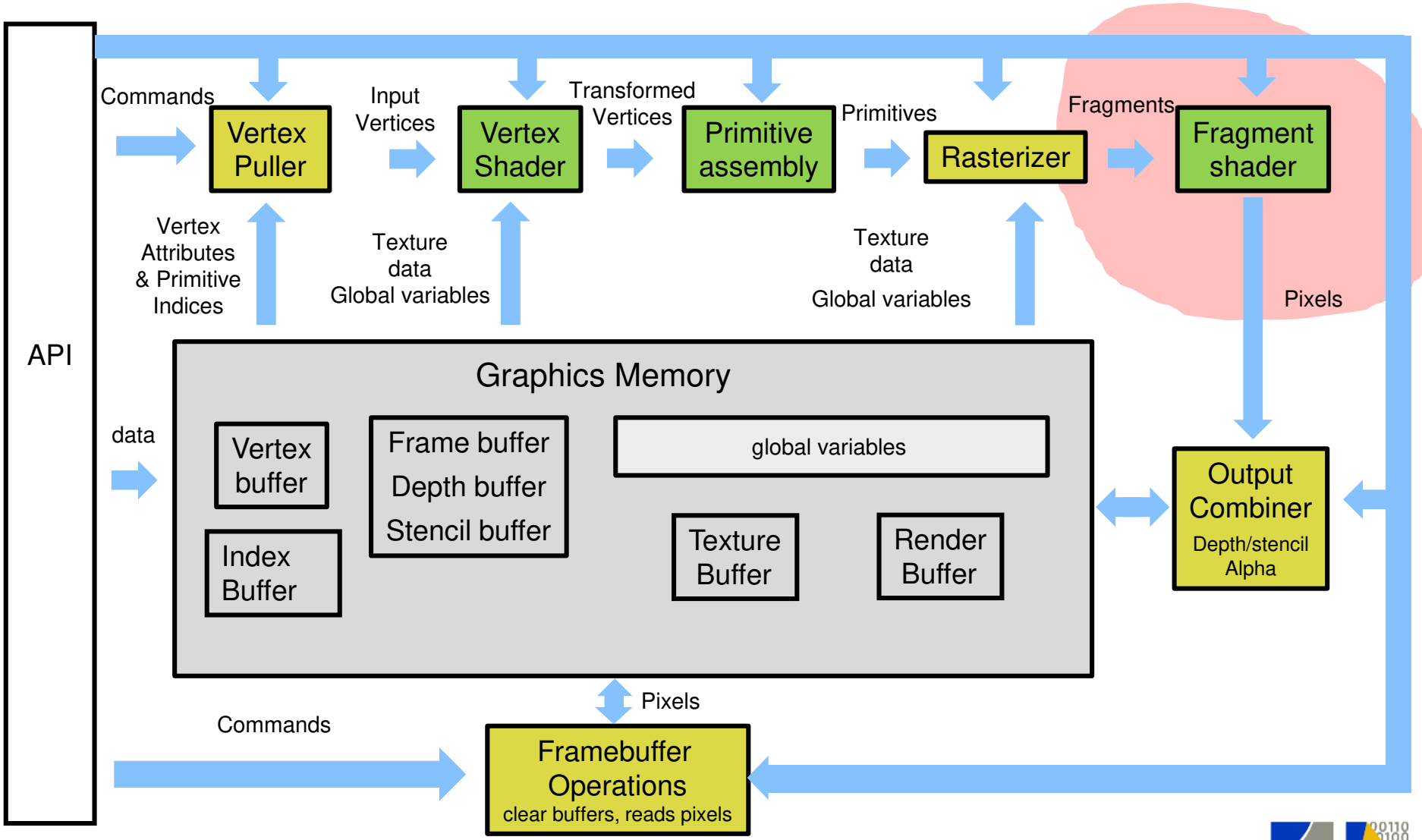
Texturfilterung und Environmentmaps

Prof. Dr. Matthias Hullin

universitätbonn

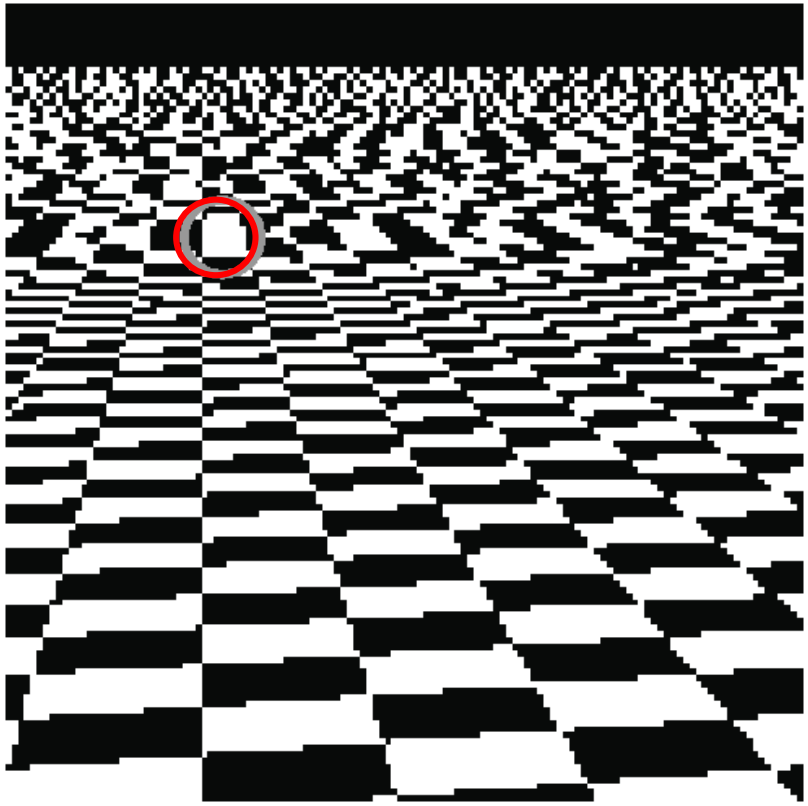
OpenGL-Pipeline



Filterung diskreter 2D-Texturen

Für eine gute Darstellung der Textur reicht es nicht aus, für ein Bildschirmpixel (x', y') die Texturkoordinaten (u, v) zu berechnen. Durch Abtastfehler treten Aliasing-Effekte auf.

Im Bild wurde in jedem Pixel (x', y') das Texel mit den dazugehörigen (auf ganze Zahlen gerundeten) Koordinaten (u, v) dargestellt. In dem mit einem Kreis markierten Gebiet wird in jedem Pixel "zufällig" ein weißes Feld getroffen; die dazwischenliegenden schwarzen Texel werden nicht berücksichtigt. Dadurch ist ein ausgedehntes weißes Gebiet zu sehen, das es in Wirklichkeit aber so gar nicht gibt.



Nyquist-Shannon-Abtasttheorem

- Existiert für eine Funktion $f(x)$ eine endliche Grenzfrequenz $\nu_{\max}$, so dass das Spektrum $F(\nu) = 0$ für $|\nu| > \nu_{\max}$, dann ist die abgetastete Funktion $f(x)$ aus den Abtastwerten $f(m\Delta x)$ fehlerfrei rekonstruierbar, sofern die Abtastfrequenz mindestens doppelt so hoch wie $\nu_{\max}$ ist:

$$\frac{1}{\Delta x} > 2\nu_{\max}$$

Antialiasing

- Das Abtasttheorem liefert eine Anleitung zur Beseitigung des Aliasing (Anti-Anliasing):
- Ist die Abtastrate fest vorgegeben, z.B. durch die Bildschirmauflösung, so muß vor der das darzustellende Signal $f(x)$ mit einem Tiefpassfilter auf

$$\nu_{\max} < \frac{1}{2\Delta x}$$

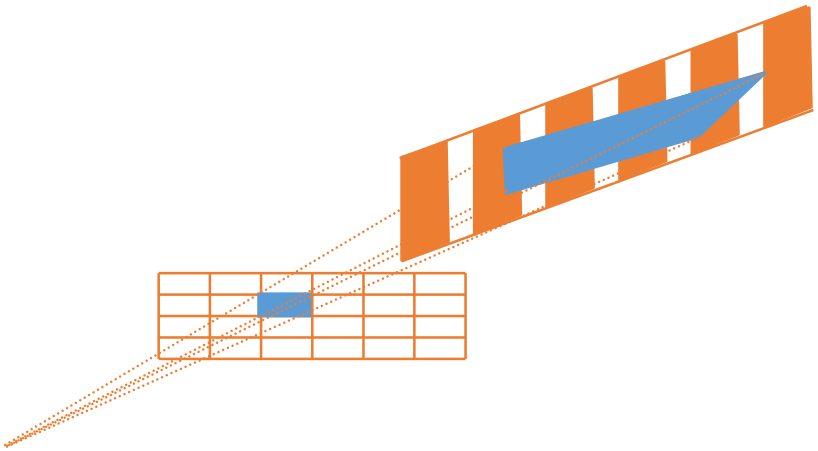
  bandgegrenzt werden.
- Dabei gehen zwar Details verloren, aber genau diese Details sind für Aliasingeffekte verantwortlich, weil sie bei der gegebenen Abtastrate nicht abbildbar sind.

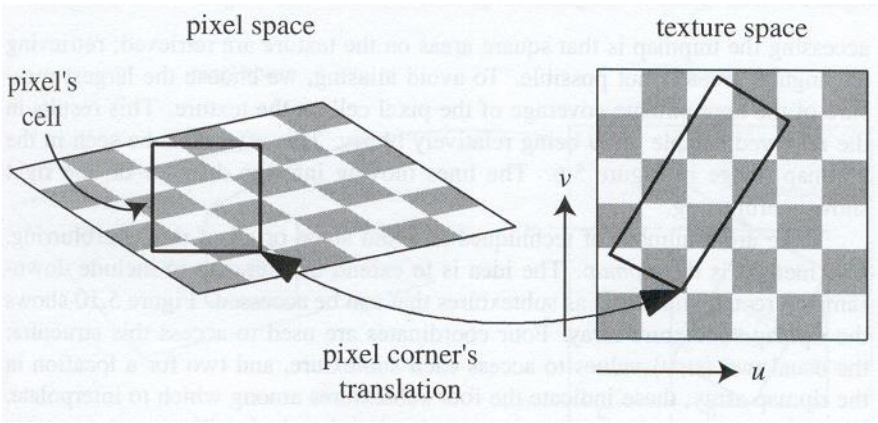
Der Footprint eines Pixels

- Die Projektion eines quadratischen Bildschirmpixels (Kantenlänge 1) auf die Texturebene, wird **Footprint** genannt und ist näherungsweise ein Parallelogramm, das von den Vektoren

$$r_1 = \left(\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}\right)^T, r_2 = \left(\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}\right)^T$$

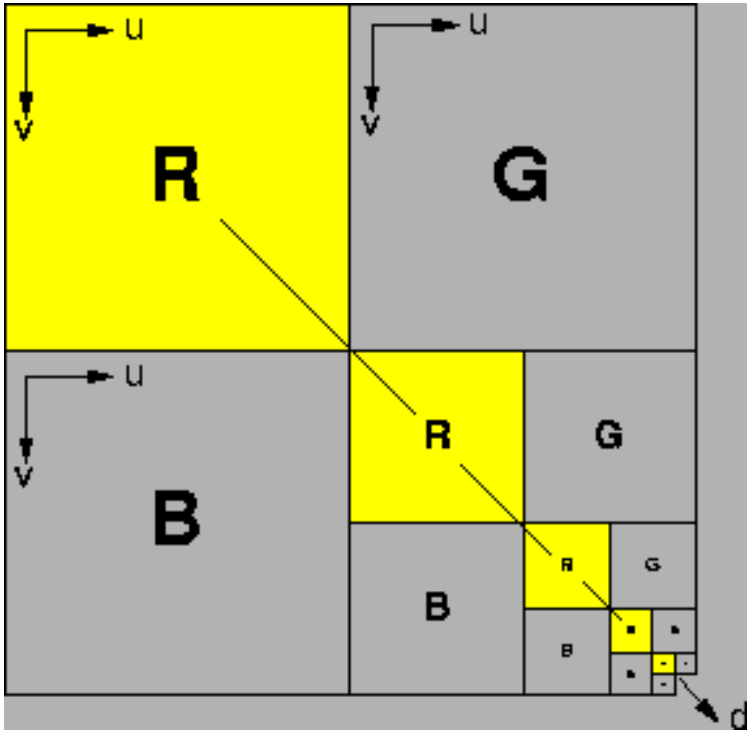
  aufgespannt wird.
- Bei den Echtzeitverfahren wird dieser Footprint nun durch einfachere Flächen ersetzt, für die die Summe oder der Mittelwert der dazugehörenden Texturwerte im voraus berechnet werden kann.





Das Mip-Mapping Verfahren

- Das wichtigste und bekannteste Verfahren für Echtzeitanwendungen ist das **Mip-Mapping-Verfahren**. Eine Mip-Map $C_{\text{mip}}\,[i,j]$ speichert eine quadratische Textur $C[i,j]$ der Größe $n \times n$, wobei $n = 2^k$ eine Zweierpotenz sein muß, in fortlaufend halbierten Auflösungsstufen.



Mip-Mapping

Auf der Stufe $d = 0$ werden die Texturwerte direkt übernommen.

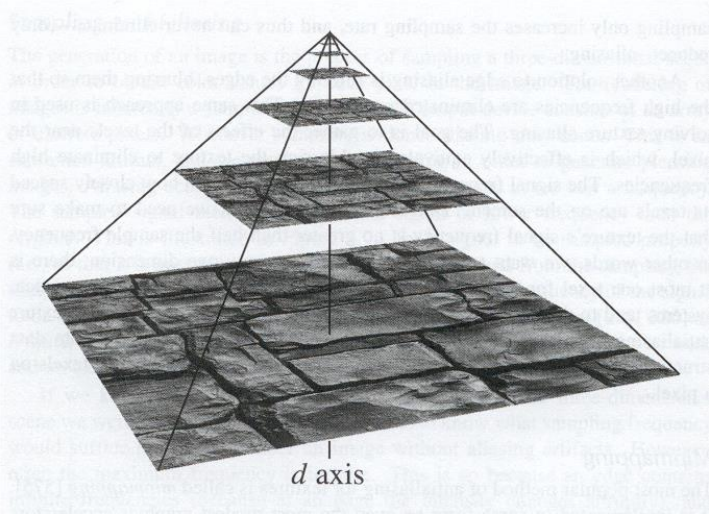
$$C^{0}_{mip}[i, j] = C[i, j],\ 0 \le \mathrm{i,j} < 2^{\mathrm{k}}.$$

Die übrigen Stufen d entstehen durch Filterung (Summation) der jeweils vorhergehenden Stufe.

$$C^{d}_{mip}[i, j] = \frac{1}{4}\left(C^{d-1}_{mip}[2i, 2j] + C^{d-1}_{mip}[2i+1, 2j] + C^{d-1}_{mip}[2i, 2j+1] + C^{d-1}_{mip}[2i+1, 2j+1]\right)$$

$1 \le d < k$ -1 und $0 \le i, j < 2^{k-d}$.

Auf der Stufe d der Texturhierarchie werden also 2^{2d} Texel der Originaltextur als ein einziges Texel dargestellt.



Mip-Mapping

Mit dieser nur einmal vorberechneten Texturpyramide können beliebige Flächenelemente wie folgt texturiert werden:

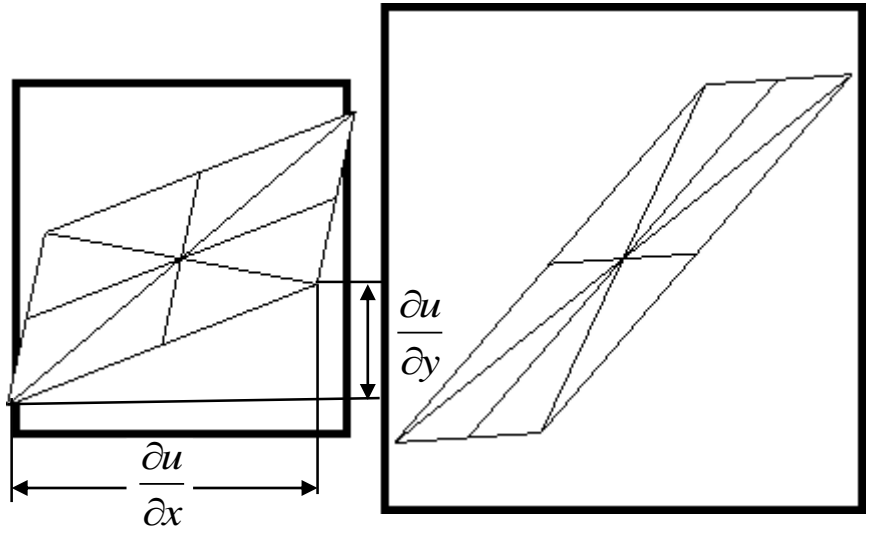
die Texturkoordinaten (u,v) des Pixelmittelpunkts und die Ableitungen nach den Bildschirmkoordinaten

$$\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}, \frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}$$

seien gegeben. Dann hat die Projektion des quadratischen Pixels auf die Texturebene, also der Footprint, die Kantenlängen

$$a = \sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2}$$

$$b = \sqrt{\left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2}.$$



Mip-Mapping

Als Seitenlänge des quadratischen Footprints wählt man oft $l = \max(a, b)$. Das Maximum und nicht etwa der Mittelwert wird deshalb gewählt, weil die entstehende zusätzliche Verschmierung des Ergebnisbildes durch zu große Footprints eher in Kauf genommen werden kann als Aliasing durch zu kleine Footprints. Unter diesen Voraussetzungen kann der Gesamttexturwert auf der Stufe $d = \log_2(l)$ gemäß

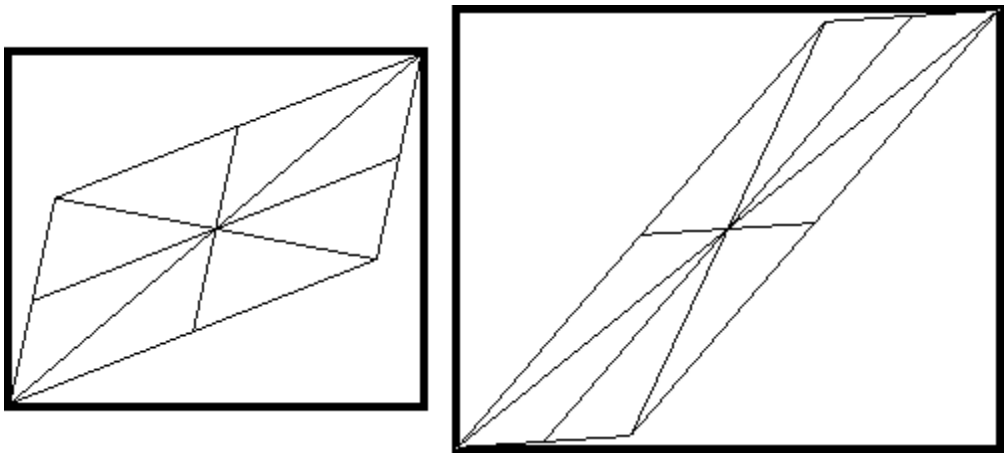
$$C_{tex}(u, v, d) = BiLinInt(C^{d}_{mip}, u, v)$$

berechnet werden. Dies geht natürlich nur, wenn d ein ganzzahliger Wert ist. Ist dies nicht der Fall, dann muß d entweder zum nächstgelegenen ganzzahligen Wert gerundet werden, oder aber man führt eine lineare Interpolation zwischen den beiden benachbarten Stufen $\lfloor d \rfloor$ und $\lfloor d + 1 \rfloor$ durch. Als Ergebnis der sog. *trilinearen* Interpolation erhalten wir dann:

$$C_{tex}(u, v, d) = (d - \lfloor d \rfloor) * BiLinInt(C^{\lfloor d \rfloor + 1}_{mip}, u, v) + (\lfloor d \rfloor + 1 - d) * BiLinInt(C^{\lfloor d \rfloor}_{mip}, u, v).$$

Summed Area Tables (SAT)

Die Richtungsabhängigkeit der Projektion (anisotrop) des Footprints kann durch das Mip-Mapping gar nicht berücksichtigt werden. SAT erlauben anstatt der Approximation durch Quadrate (u,v)-achsenparallele Rechtecke zu verwenden und somit diesem Defizit (zumindest teilweise) Rechnung zu tragen.



Summed Area Tables (SAT)

- Der Berechnungsaufwand der einfachsten Methode, das arithmetische Mittel über alle Texturwerte eines (u, v)-achsenparallelen Rechtecks $((i_0, j_0), (i_1, j_1))$ zu berechnen, ist proportional zur Größe des Rechtecks:

$$C_{avg}(i_0, j_0, i_1, j_1) = \frac{\sum_{i=i_0}^{i_1} \sum_{j=j_0}^{j_1} C[i, j]}{(i_1 - i_0 + 1)(j_1 - j_0 + 1)}$$

Berechnet man dagegen in einem Vorverarbeitungsschritt eine sogenannte Summed-Area-Tabelle

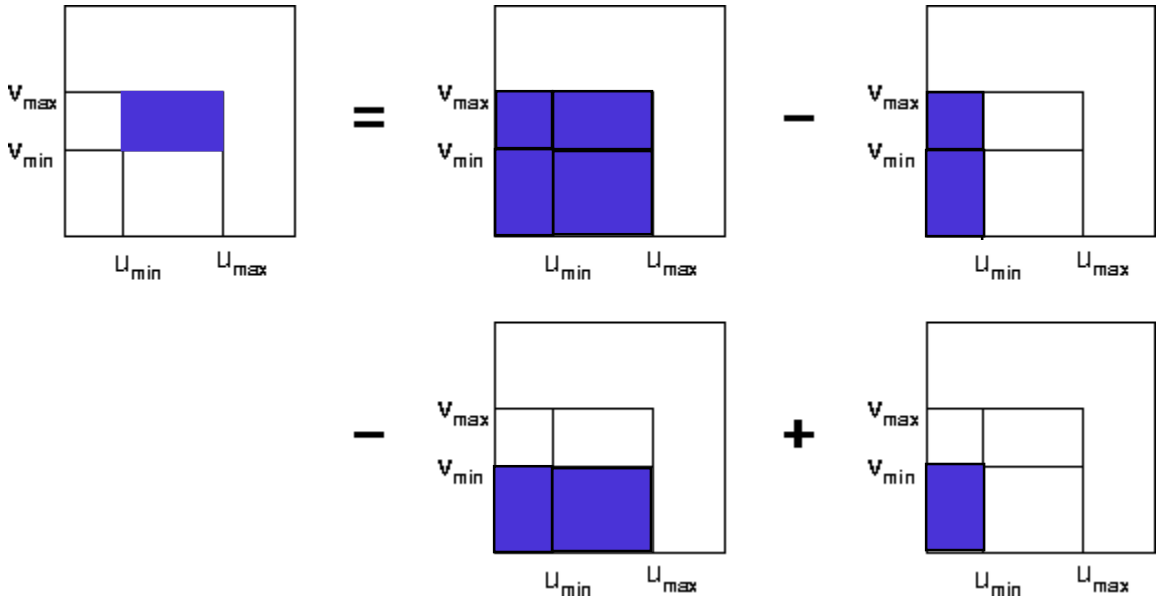
$$C_{sat}(i_s, j_s) = \sum_{i=i_0}^{i_s} \sum_{j=j_0}^{j_s} C[i, j]$$

und setzt

$$C_{sat}[i, -1] = C_{sat}[-1, j] = 0,$$

Summed Area Tables (SAT)

Dann kann C_{avg} alternativ mit konstantem Berechnungsaufwand, unabhängig von der Rechteckgröße berechnet werden:

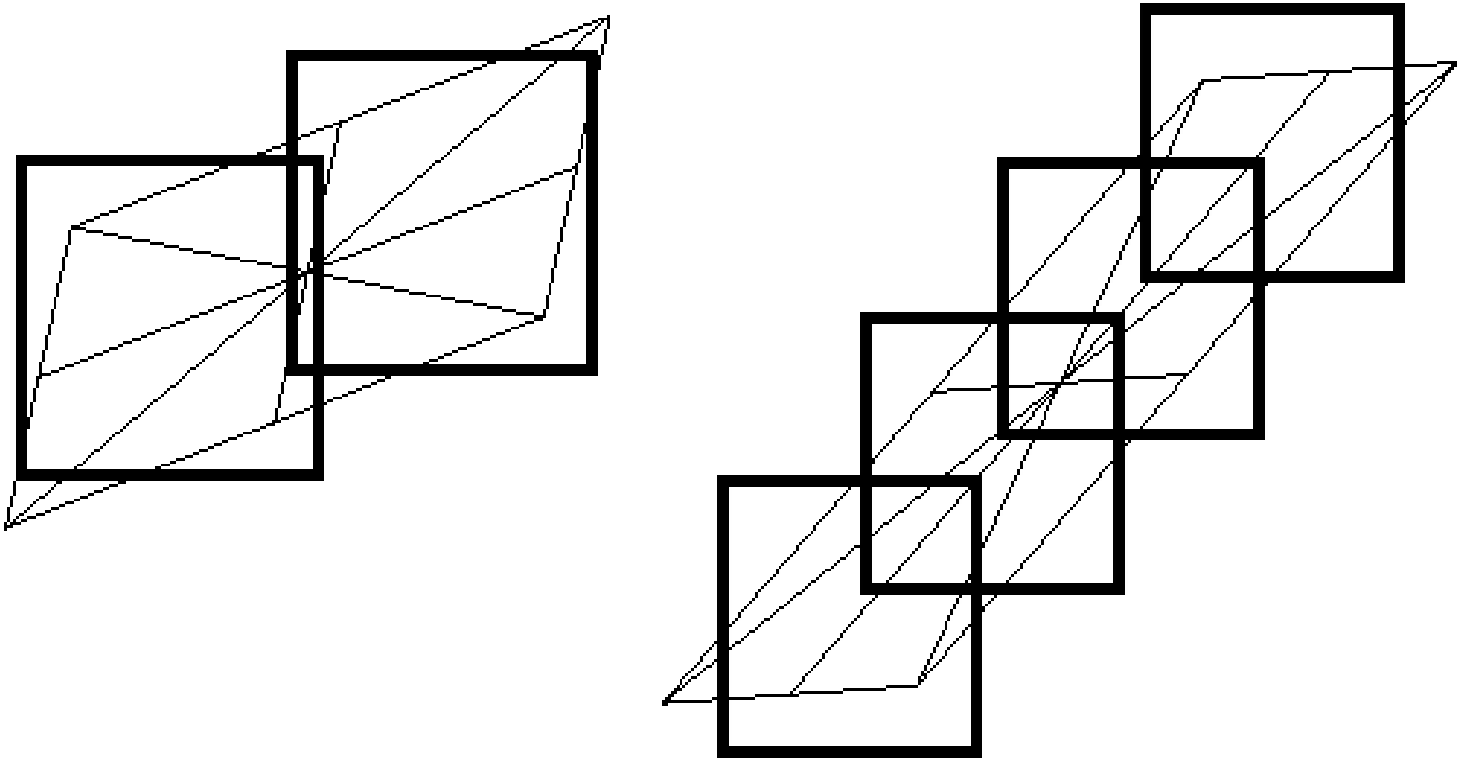


$$C_{avg}(i_0, j_0, i_1, j_1) = \frac{C_{sat}[i_1, j_1] - C_{sat}[i_0 - 1, j_1] - C_{sat}[i_1, j_0 - 1] + C_{sat}[i_0 - 1, j_0 - 1]}{(i_1 - i_0 + 1)(j_1 - j_0 + 1)}.$$

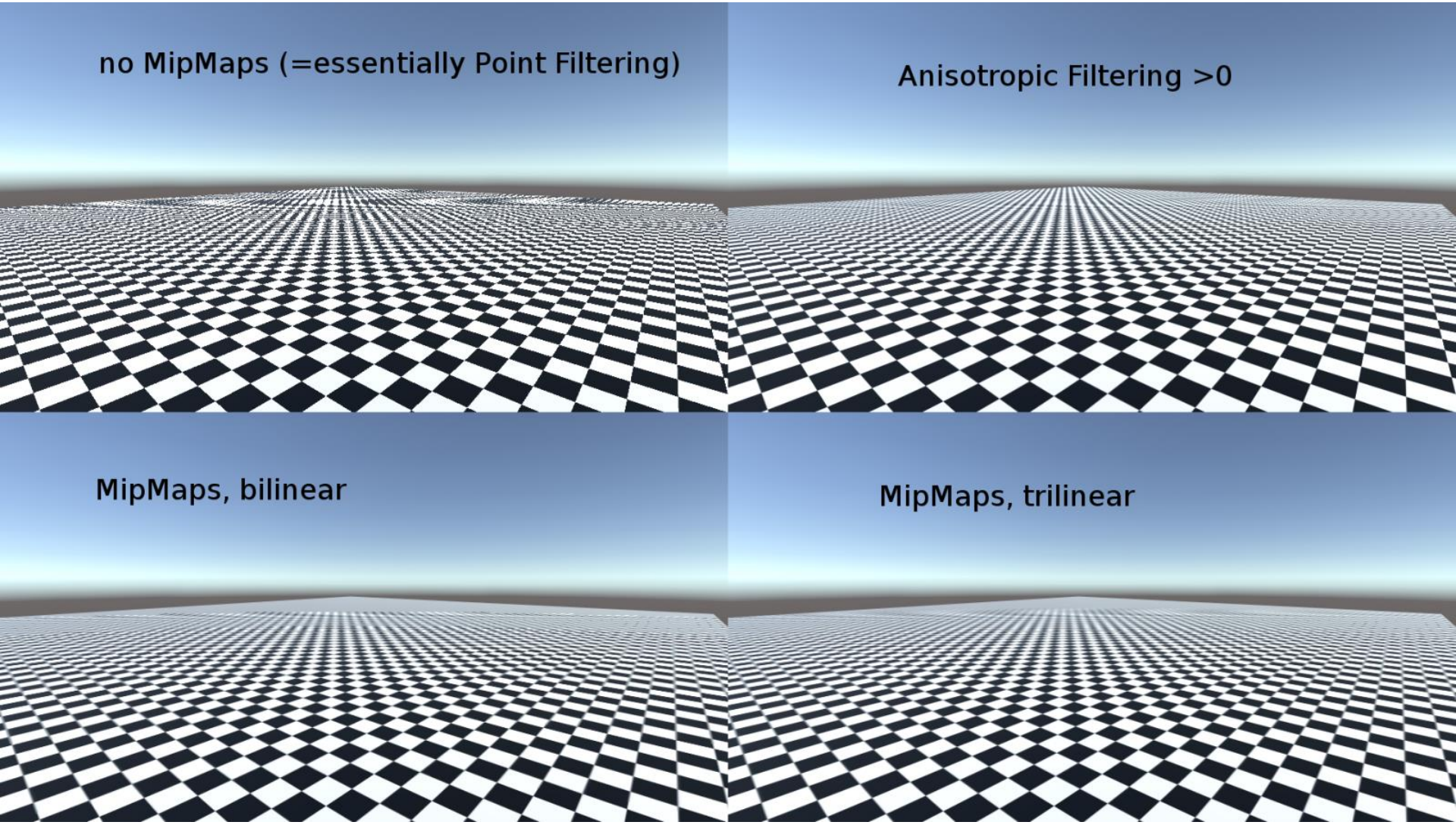
Footprint-Assembly (FPA)

MipMaps und SATs können verwendet warden, komplexere Footprints zu synthetisieren (Schilling und Knittel, 1996). Ziele:

- Maximiere Qualität, Cache-Hitrate und Kohärenz
- Minimiere Rechenaufwand, Texturzugriffe



Vergleich

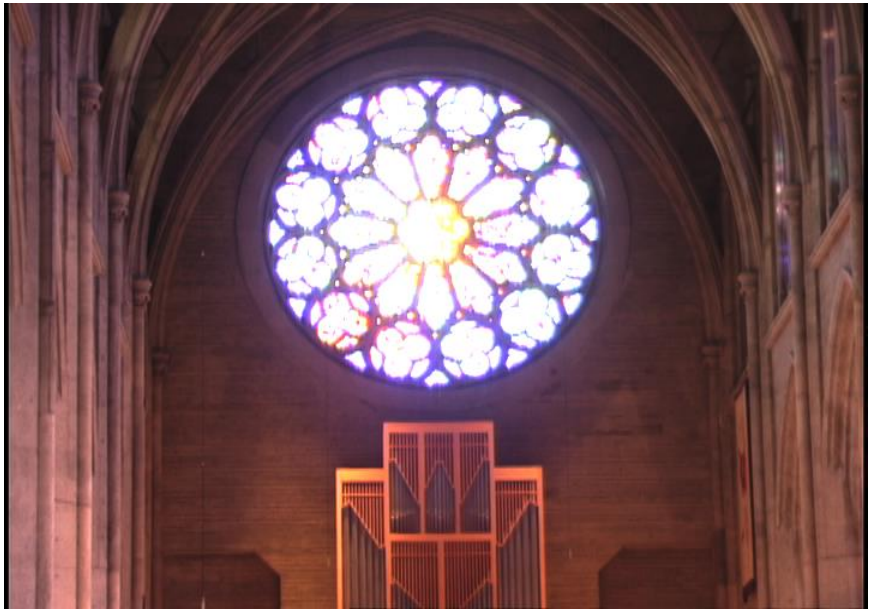

no MipMaps (=essentially Point Filtering)
Anisotropic Filtering >0
MipMaps, bilinear
MipMaps, trilinear

Environment Mapping

Environment Mapping, auch Reflection Mapping genannt, ist zunächst eine einfache Möglichkeit, Reflexionen zumindest approximativ mit Hilfe von Textur-Hardware zu berechnen. Die Grundidee des Environment Mapping ist einfach: Ist ein Objekt verglichen mit dem Abstand zu umgebenden Objekten klein, so hängt die einfallende Beleuchtsstärke nur von der Richtung, nicht von der Position eines Punktes auf dem Objekt ab. Daher kann die einfallende Beleuchtung für ein Objekt vorberechnet und in einer 2D-Textur, der Environment Map gespeichert werden.

High Dynamic Range: HDR

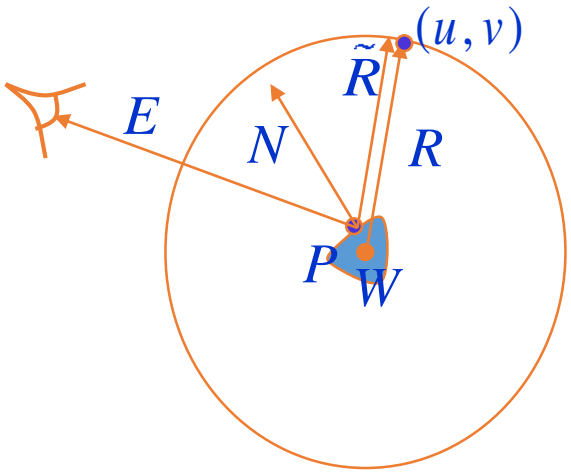
- „Dynamischer Bereich“ bezeichnet das Verhältnis zwischen der maximalen und der minimalen Intensität innerhalb einer Szene.
- Im Falle von „High Dynamic Range“ besagt es, dass die Leuchtdichte ein sehr großes Spektrum von Hell bis Dunkel abdeckt.
- Bildformat: OpenEXR (Opensource HDR-Grafikdateiformat)
    - pro Farbkanal eine 16-Bit-Gleitkommazahl



Environment Mapping

Anschaulich wird das reflektierende Objekt von einer virtuellen Kugel umgeben, auf deren Innenseite die Szenenumgebung als zweidimensionale Textur, die Environment Map aufgetragen ist. Einem Punkt P auf der Objektoberfläche werden dann Texturkoordinaten (u, v) zugeordnet. Die Richtung R und damit die Texturkoordinaten (u, v) können einfach aus der Richtung E zur Kamera und der Normale N in Punkt P berechnet werden:

$$R = \text{reflect}(E, N) = E - 2(E \cdot N)N$$



Environment Mapping

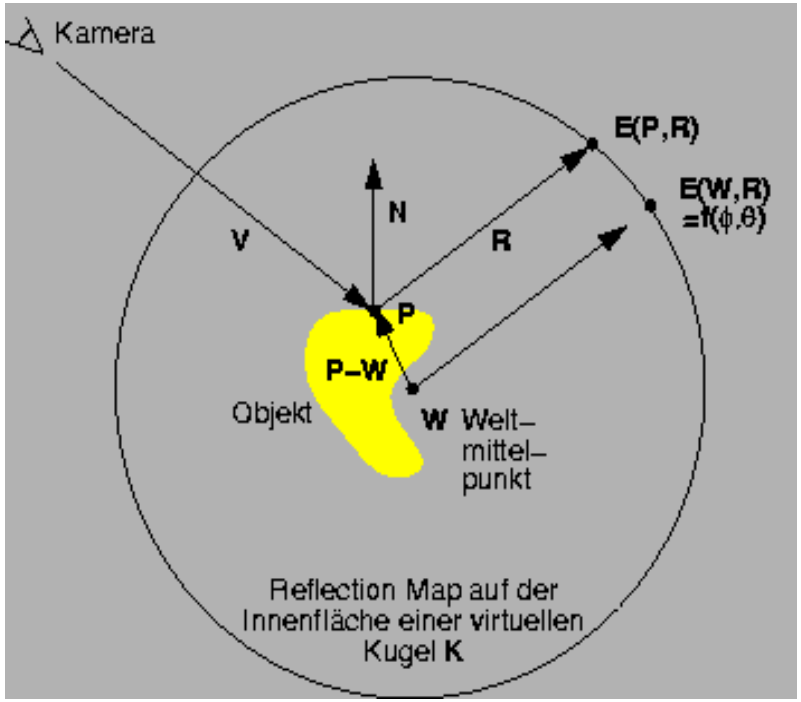
Für die Beleuchtungsrechnung wird ein um spiegelnde Reflexionen erweitertes Phong-Beleuchtungsmodell angewendet:

$$L_{out} = L_{amb} + L_{diff} + L_{spec} + rL_{reflmap}(\phi, \theta)$$

Vorteile des Environment Mapping
+ schnell und einfach zu berechnen
+ liefert gute Visualisierungsergebnisse, wenn die Textur z.B. den Himmel oder einen weit entfernten Horizont repräsentiert.
+ kann verwendet werden, um große ausgedehnte Lichtquellen als Textur darzustellen.

Environment Mapping

Nachteile
- Wie wird die Environment Map generiert und parametrisiert?
- Die Reflektionsberechnung ist nur dann korrekt, wenn der Objektpunkt P sich im Weltmittelpunkt W befindet. Mit zunehmendem Abstand zwischen P und W treten verstärkt Verzerrungen auf.

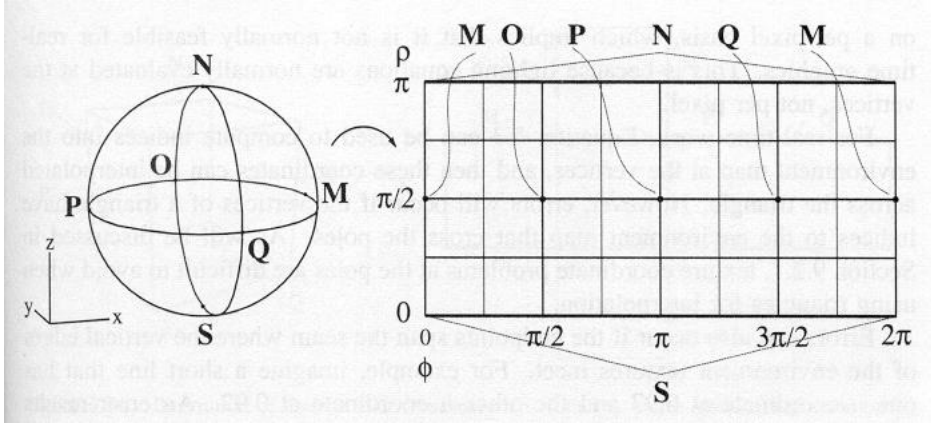


Environment Mapping

- Ist die Environment Map schlecht parametrisiert können erhebliche Aliasing-Probleme auftreten.
- Es wird keine Verdeckungsrechnung durchgeführt. Das Problem, daß der reflektierte Strahl R auf ein blockierendes Szenenobjekt treffen kann, wird ignoriert.
- Szenenobjekte können sich nicht gegenseitig widerspiegeln. Bei der Reflektionsberechnung wird nur die a priori berechnete Environment Map berücksichtigt.

Parametrisierung von Environment Maps

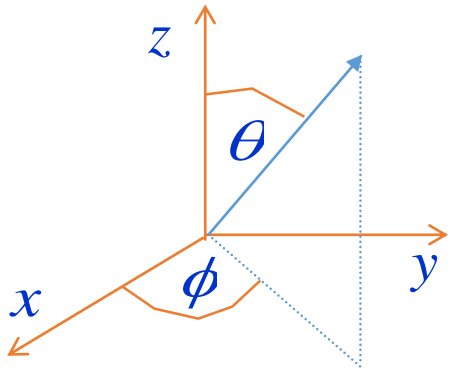
Blinn, Newell 1976 verwenden Kugelkoordinaten zur Parametrisierung der Environment Map. Diese werden dann als Texturkoordinaten auf dem Objekt verwendet.



Ist der reflektierte Strahl $R = (R_x, R_y, R_z)$ gegeben, so berechnen sich (ϕ, θ) analog wie beim Kugel-Mapping gemäß

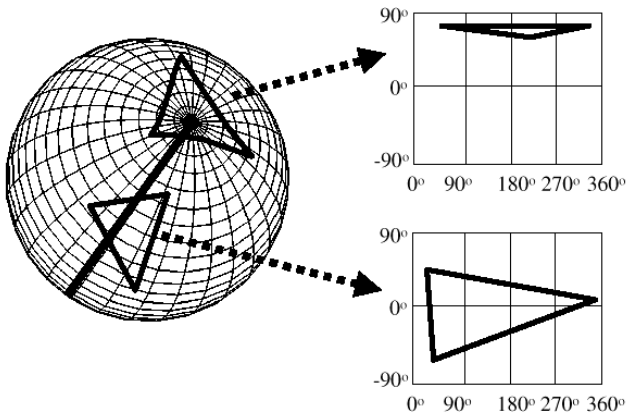
$$\theta = \arccos(R_z)$$

$$\phi = \begin{cases} \arccos(R_x / \sin\theta), & \text{falls } R_y \geq 0 \\ 2\pi - \arccos(R_x / \sin\theta), & \text{sonst} \end{cases}$$



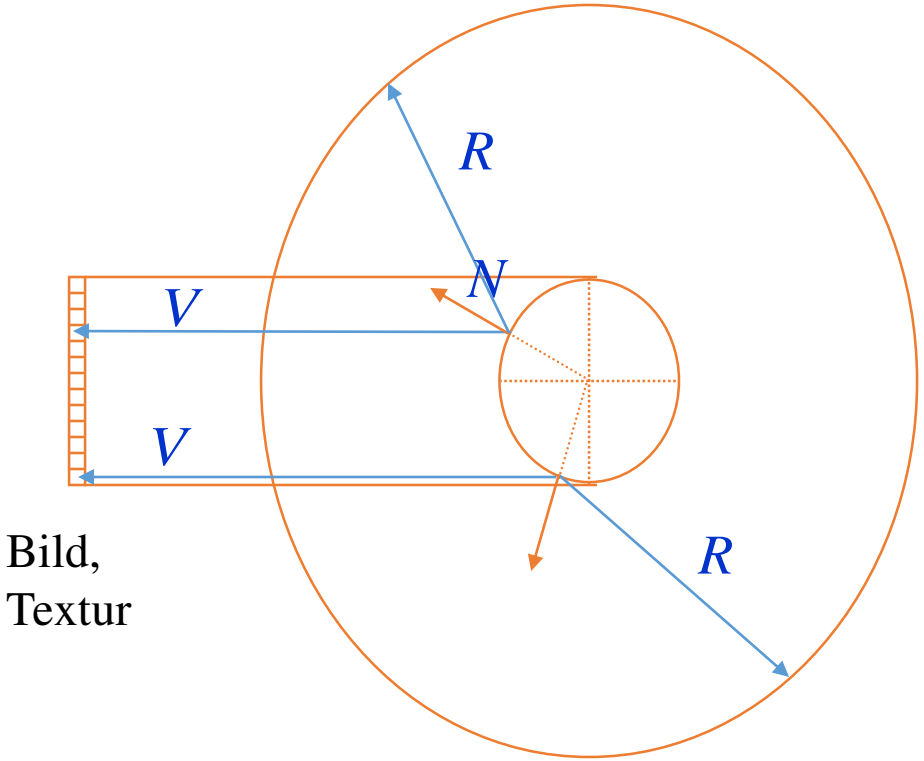
Parametrisierung von Blinn und Newell

+ Einfach verständlich, älteste Parametrisierung.
Die Erzeugung der Environment Map ist kompliziert. In der Praxis wird meist eine Environment Map, die auf den Innenseiten eines Würfels aufgebracht ist, entsprechend umgerechnet.
- Die Parametrisierung mittels Kugelkoordinaten ist schlecht. Um die Pole liegen wesentlich mehr Texel/Fläche als am Äquator. Das führt zu einer schlechten Abtastung.
- Wird gegenwärtig nicht von Hardware unterstützt.
- Ein Dreieck, das den Pol enthält, enthält den Pol nicht mehr, wenn linear in Polarkoordinaten interpoliert wird.
- Schneidet ein Dreieck die Linie mit $\phi = 0$, so schneidet ein linear interpoliertes Dreieck diese nicht mehr.



Sphere Mapping

OpenGL unterstützt Environment Mapping mit der sogenannten Sphere Map. Dabei wird die gesamte umhüllende Kugel des Objekts, auf deren Innenseite wie beim Standard Environment Mapping die Textur der Umgebung aufgebracht ist, auf einen Kreis abgebildet:



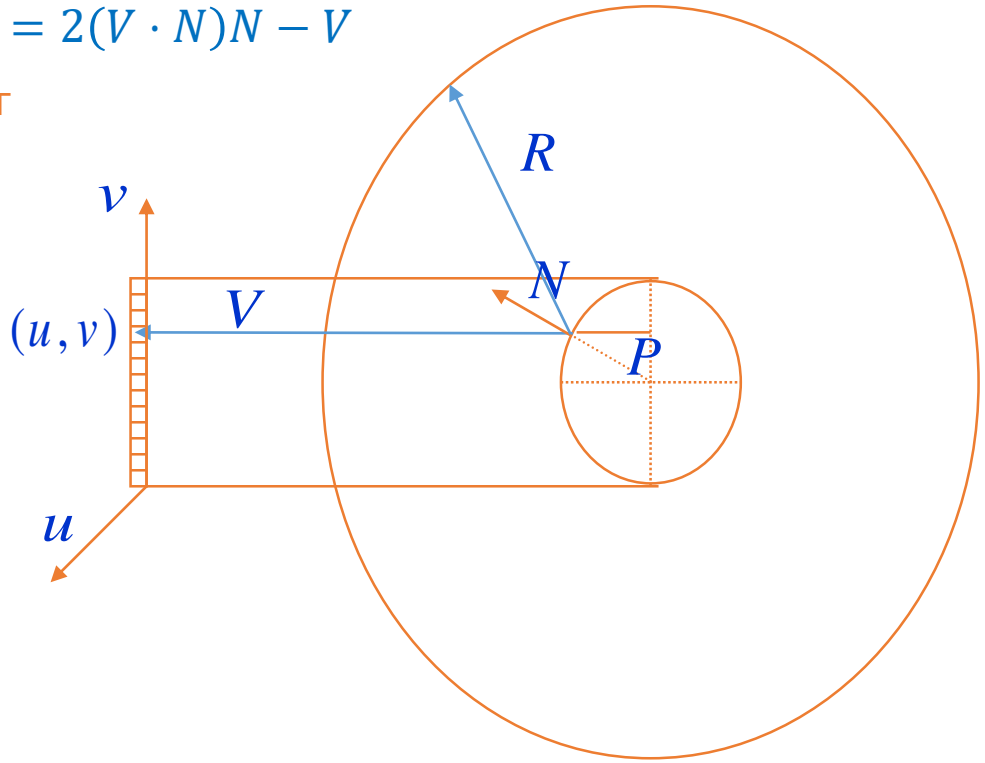
Parametrisierung der Sphere Map

- Berechnung von Texturkoordinaten in einem Punkt P der inneren Kugel:
- Sei $V = (0,0,1)^\top$ die Beobachtungsrichtung und $R = (R_x, R_y, R_z)^\top$ die Richtung des reflektierten Strahls. Dann werden die Texturkoordinaten $(u, v)^\top$ der Spheremap wie folgt berechnet:
- Für den reflektierten Strahl gilt: $R = 2(V \cdot N)N - V$
- Mit $N = (N_x, N_y, N_z)^\top = (u, v, N_z)^\top$ und $V = (0,0,1)^\top$ folgt daraus

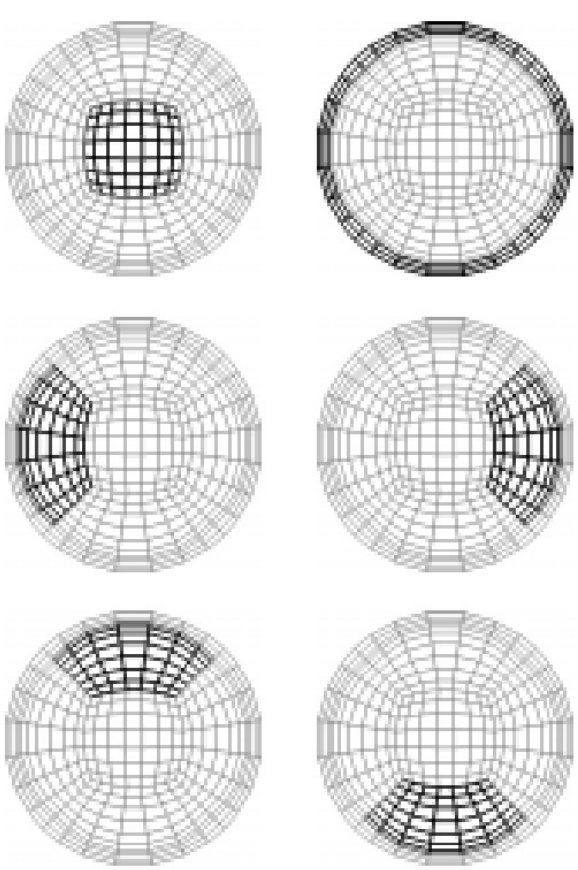
$$\begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = 2\left(\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} u \\ v \\ N_z \end{pmatrix}\right)\begin{pmatrix} u \\ v \\ N_z \end{pmatrix} - \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 2N_z u \\ 2N_z v \\ 2N_z^2 - 1 \end{pmatrix}$$

Auflösen nach u und v liefert

$$u = \frac{R_x}{\sqrt{2(R_z + 1)}}, \quad v = \frac{R_y}{\sqrt{2(R_z + 1)}}$$



Sphere Mapping



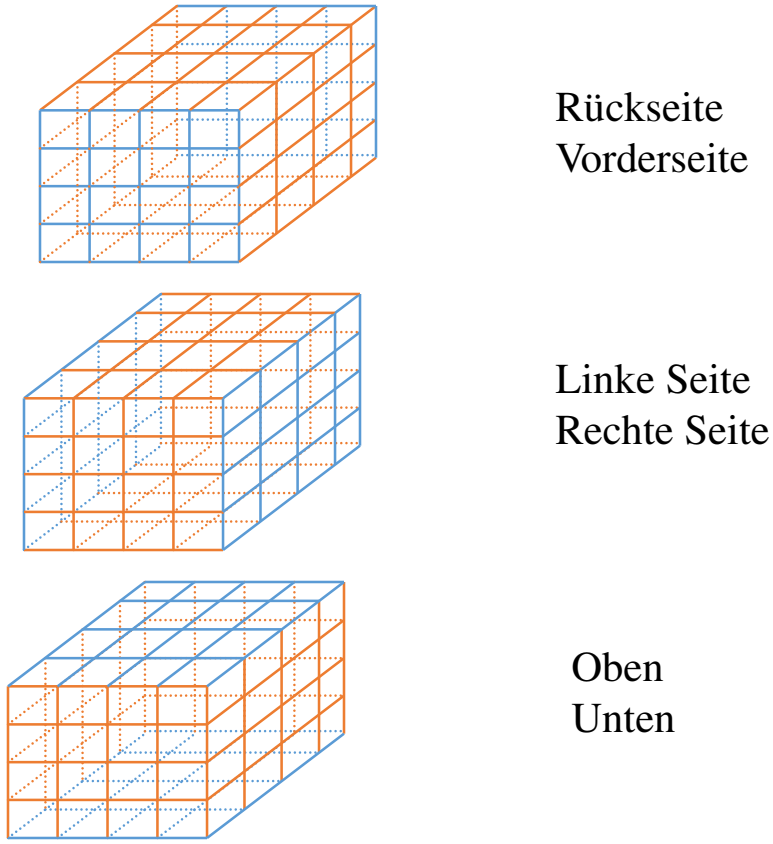
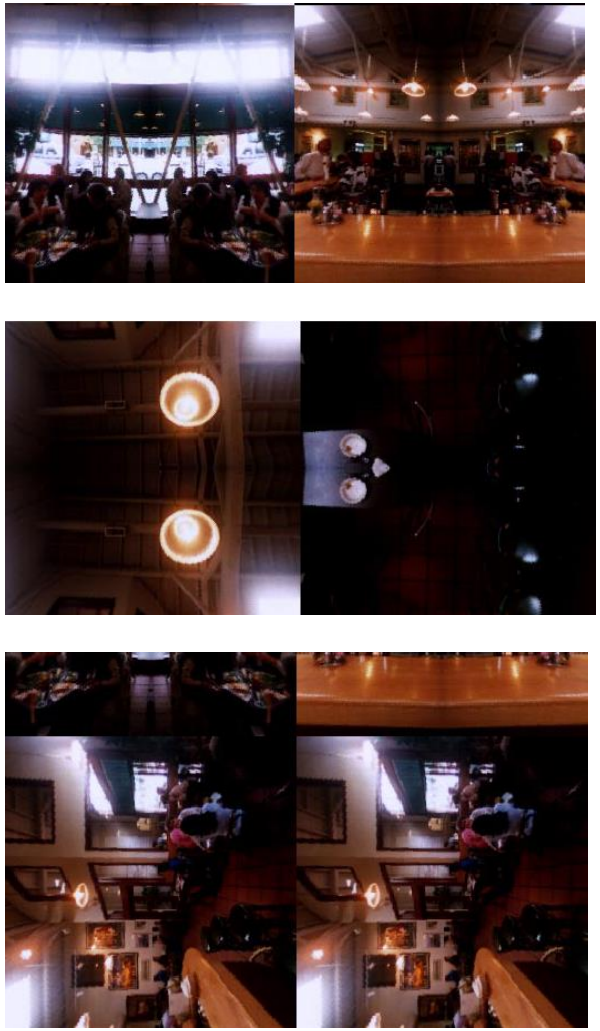


Abbildung von 6 Würfelseiten auf eine Sphere Map

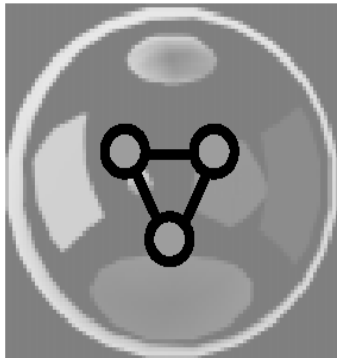
Berechnung einer Sphere Map



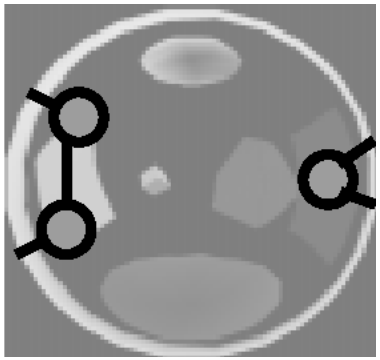
Vor/Nachteile

+ Ein Foto einer spiegelnden Kugel aus hinreichender Entfernung ("Light Probe") kann schon eine überzeugende Sphere Map liefern

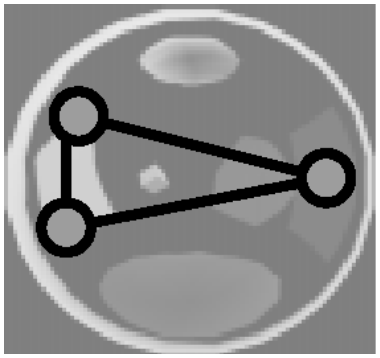
- Interpolation der Texturkoordinaten führt zu Artefakten.
- Unregelmäßige Abtastung: Maximale Abtastraten in Richtungen entgegen der Beobachtungsrichtung, Abtastrate in Richtung der Beobachtungsrichtung geht gegen 0. In Beobachtungsrichtung hat die Parametrisierung eine Singularität.
- Aliasing Probleme vor allem am Rand.



Relativ gute Abtastung.



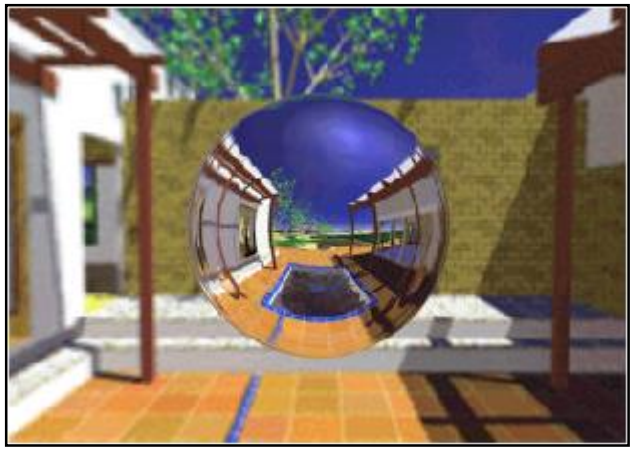
Beabsichtigte Interpolation



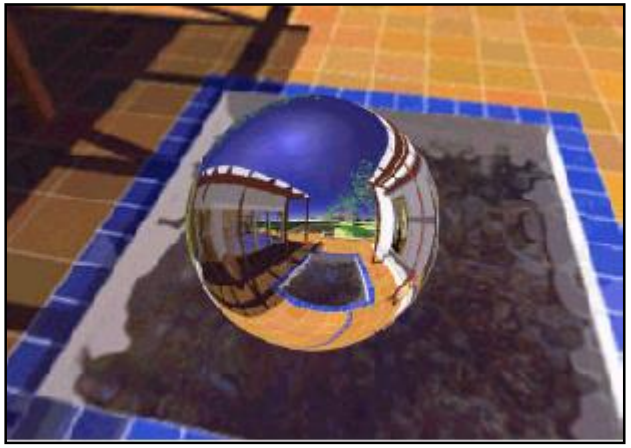
Tatsächliche Interpolation mit Hardware, Wrapping Effekt

Vor/Nachteile

Abhängigkeit vom Beobachtungspunkt



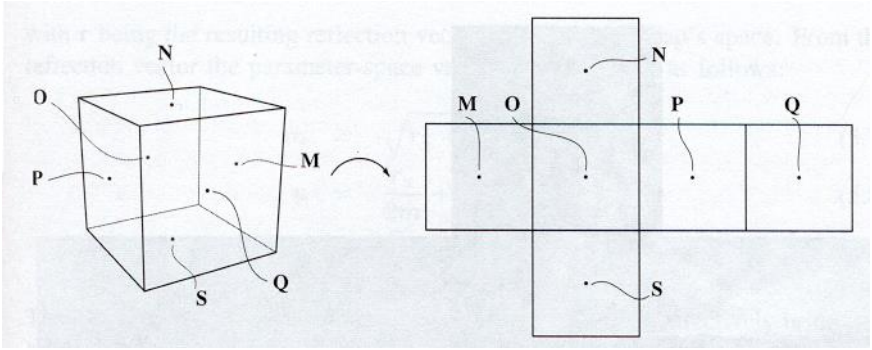
Korrekte Spiegelung.



Falsche Spiegelung: Der Pool sollte in der Spiegelung nicht sichtbar sein.

Parametrisierung von Greene

Greene 1986 verwendet anstatt der Projektion der Umgebung auf eine Kugel eine Projektion auf einen Würfel. Die Environment Map besteht aus 6 ebenen Texturen entsprechend den 6 Würfelseiten. Zur Erzeugung wird eine Kamera in der Mitte des Würfels platziert und 6 Aufnahmen in jede Richtung gemacht. In der Praxis wird die Szene ausgehend vom Mittelpunkt des Objekts 6 mal mit unterschiedlichen Blickrichtungen gerendert. Je nach Richtung des reflektierten Strahls wird eine der 6 Texturen ausgewählt.

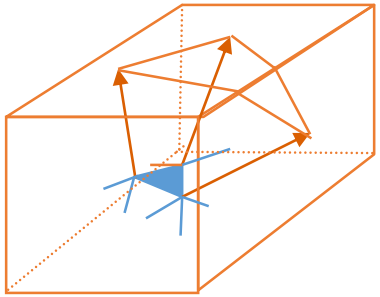


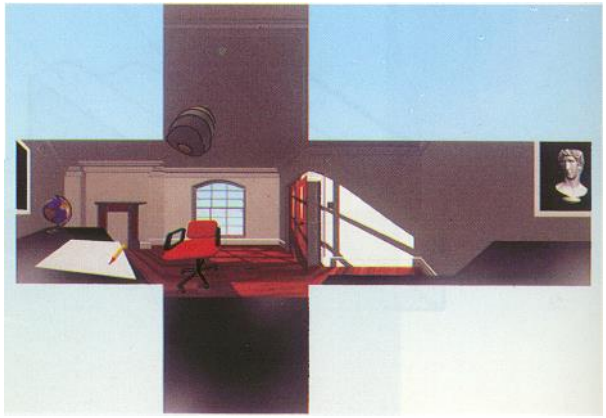
+ Die Parametrisierung ist regelmäßiger eine Kugelkoordinatenparametrisierung.
+ Die Environment Map kann einfach mit Hilfe von Hardware erzeugt werden.
+ Implementierung in Hardware (z.B. GL_TEXTURE_CUBE_MAP)

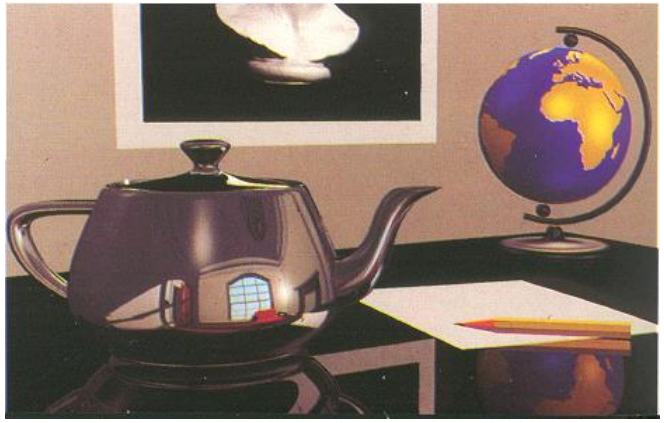
Parametrisierung von Environment Maps

Die Projektion auf einen Würfel hat auch Nachteile:

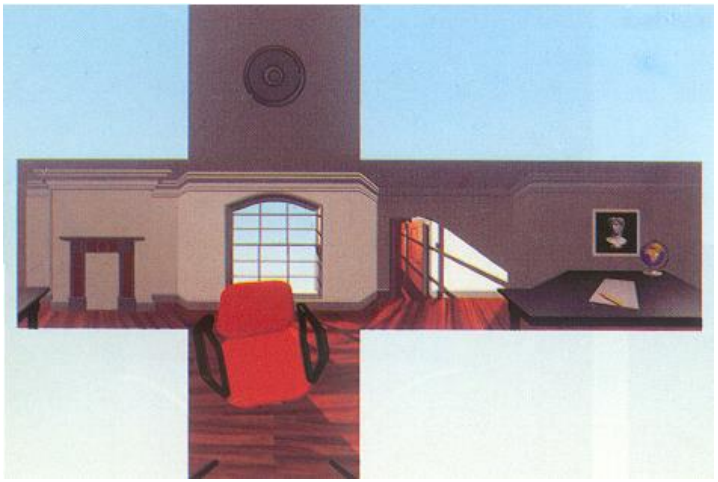
-Liegen die Texturkoordinaten von Eckpunkten von Dreiecken des Objekts in unterschiedlichen Würfelseiten, so ist es schwierig dazwischen zu interpolieren. Eine Methode ist, diese Dreiecke entsprechen zu unterteilen.
- Filterung und bilineare Interpolation entlang Würfelkanten ist schwierig.

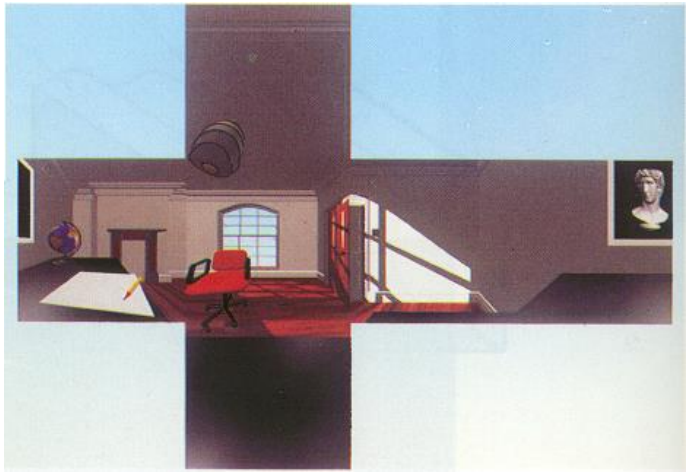






Parametrisierung von Enviroment Maps



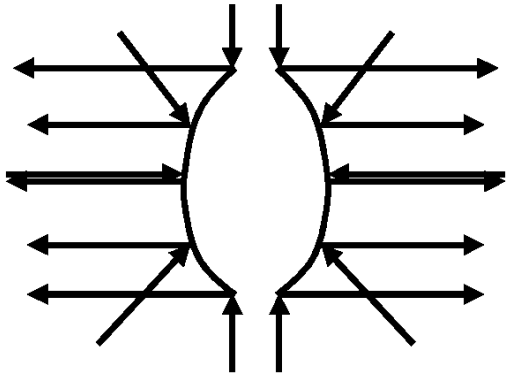


Die Environment Map ist abhängig vom Objektmittelpunkt!

Dual-Parabolisches Mapping

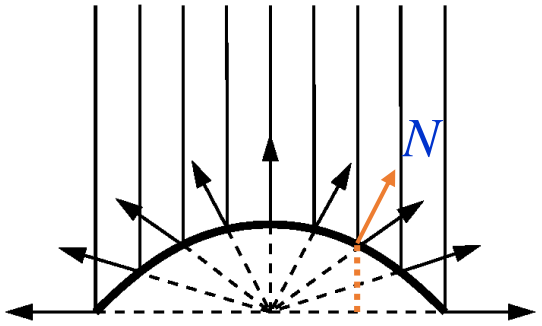
Heidrich und Seidel (1999) verwenden als reflektierendes Objekt anstatt einer Kugel zwei Paraboloide.

$$f(x,y) = \frac{1}{2} - \frac{1}{2}(x^2 + y^2),\ x^2 + y^2 \le 1$$



Im Gegensatz zum Sphere-Mapping, wo die gesamte Umgebung in einer Textur abgelegt wird, werden zwei Texturen verwendet. Eine für die Halbkugel entgegen der Blickrichtung (hinten) und eine in Blickrichtung (vorne). Für die Normale im Punkt *(x,y)* ergibt sich:

$$N = \frac{1}{\sqrt{x^2 + y^2 + 1}} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



Dual-Parabolisches Mapping

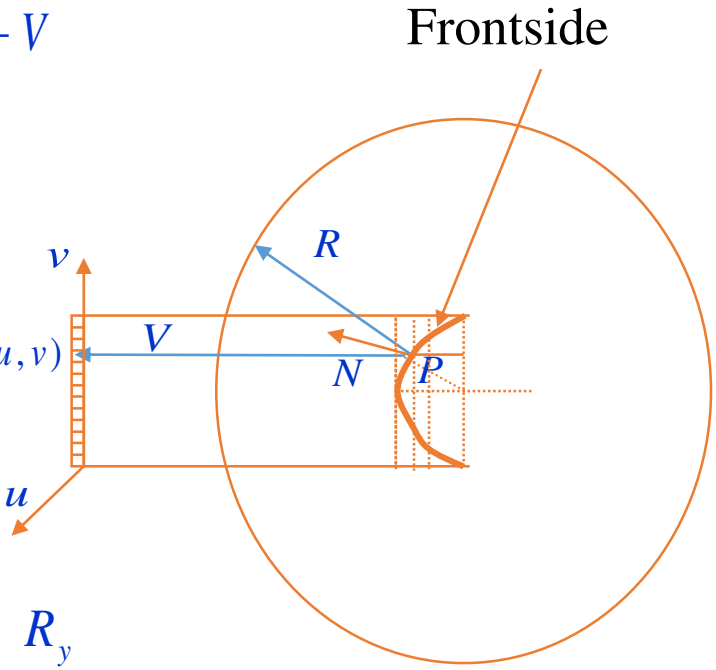
Berechnung von Texturkoordinaten in einem Punkt P des Objekts:
Sei $R = (R_x, R_y, R_z)^t$ die Richtung des reflektierten Strahls (sie ist abhängig von der aktuellen Beobachtungsrichtung!). Dann berechnen sich die Texturkoordinaten $(u,v)^t$ wie folgt.

Für den reflektierten Strahl gilt: $R = 2(V \square N)N - V$

mit $N=(N_x,N_y,N_z)^t = \frac{1}{\sqrt{u^2+v^2+1}}(u,v,1)^t$

und $V=(0,0,-1)$. Daraus folgt

$$\begin{pmatrix} R_x \\ R_y \\ R_z \end{pmatrix} = 2\left(\frac{1}{u^2+v^2+1}\begin{pmatrix}0\\0\\-1\end{pmatrix}\cdot\begin{pmatrix}u\\v\\-1\end{pmatrix}\right)\begin{pmatrix}u\\v\\-1\end{pmatrix} - \begin{pmatrix}0\\0\\-1\end{pmatrix} = \begin{pmatrix} \frac{2}{u^2+v^2+1}u \\ \frac{2}{u^2+v^2+1}v \\ -\frac{2}{u^2+v^2+1}+1 \end{pmatrix}$$



Auflösen nach u und v liefert $u = \frac{R_x}{1-R_z}$, $v = \frac{R_y}{1-R_z}$

Vor-und Nachteile

Dual-parabolic Maps lassen sich mit einfachen Matrizenoperationen für unterschiedliche Blickrichtungen umrechnen. D.h. sie sind blickpunktsabhängig.

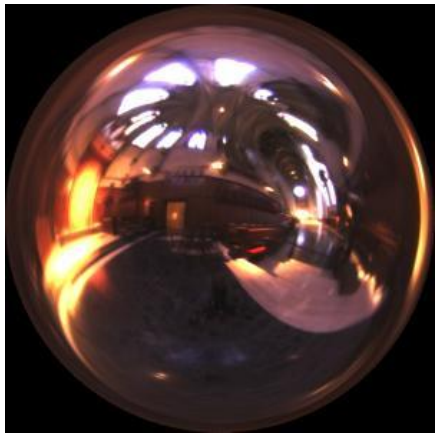
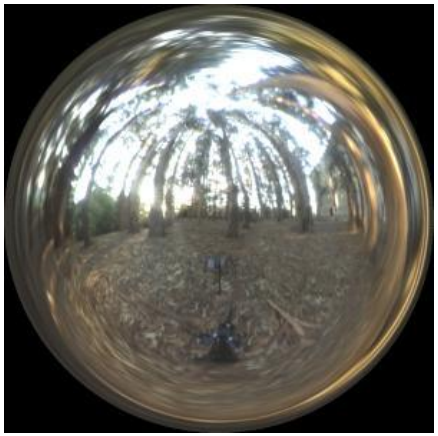
+ Wesentliche gleichmäßigere Abtastung als beim Sphere-Mapping.
+ Weniger Aliasing Probleme.
- Interpolation der Texturkoordinaten führt zu Artefakten.
- Nicht in Hardware verfügbar.

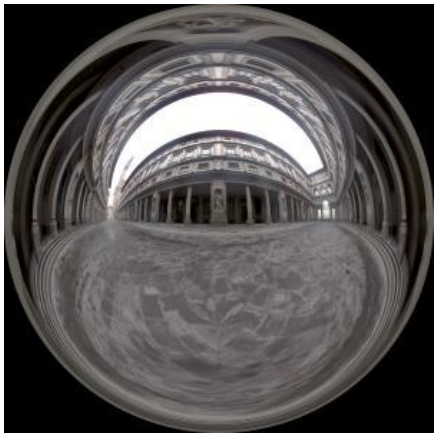
Würfel auf Dual-Parabolische Maps abgebildet.

Angular Maps [Debevec 1998]

https://www.pauldebevec.com/Probes/

"The coordinate mapping of these images is such that the center of the image is straight forward, the circumference of the image is straight backwards, and the horizontal line through the center linearly maps azimuthal angle to pixel coordinate."



Vorgefilterte Environmentmaps

- Vorgefilterte Environmentmaps speichern für eine feste Position x die reflektierte Leuchtdichte in alle Richtungen des Halbraums:

$$L_r(x,\hat{v},\hat{n},\hat{t}) = \int_{\Omega_i} \rho(\hat{\omega}_r(\hat{v},\hat{n},\hat{t}),\omega_i(\hat{l},\hat{n},\hat{t}))L_i(x,\hat{l})\underbrace{(\hat{n}\cdot\hat{l})}_{\cos(\theta_i)}\,d\hat{l},$$

wobei $\hat{v}$ Beobachtungsrichtung

$\{\hat{n},\hat{t},\hat{n}\times\hat{t}\}$ lokales Koordinatensystem in x

$\hat{\omega}_r(\hat{v},\hat{n},\hat{t})$ Blickrichtung und $\hat{\omega}_i(\hat{l},\hat{n},\hat{t})$ Lichtrichtung bzgl. des lokalen Koordinatensystems

ρ BRDF

Vorgefilterte Environmentmaps

- Beleuchtung durch umgebendes Licht, abgespeichert in einer High Dynamic Range (HDR) Environmentmap.
- Beispiel für HDR

Environment map

Rendered result

Vorgefilterte Environmentmaps

- $$L_r(x,\hat{v},\hat{n},\hat{t}) = \int_{\Omega_i} \rho(\hat{\omega}_r(\hat{v},\hat{n},\hat{t}),\omega_i(\hat{l},\hat{n},\hat{t}))L_i(x,\hat{l})\underbrace{(\hat{n}\cdot\hat{l})}_{\cos(\theta_i)}d\hat{l},$$

  Bemerkungen:

  - L_i kann als ungefilterte Environmentmap betrachtet werden.
  - Die reflektierte Leuchtdichte L_r hängt sowohl von der Beobachtungsrichtung als auch von der Orientierung der Oberfläche,d.h. vom lokalen Koordinatensystem ab.
  - Die vorgefilterte Environmentmap ist fünfdimensional:
    - Zwei Winkel für die Beobachtungsrichtung $\hat{v}$
    - Drei Winkel für das lokale Koordinatensystem $\{\hat{n},\hat{t},\hat{n}\times\hat{t}\}$

Diffuse Environmentmaps

- In diesem Fall ist die BRDF konstant:

$$\rho(\hat{v},\hat{n}) = \frac{k_d}{\pi},\ k_d \in [0,1]$$

Damit ergibt sich

$$L_r(x,\hat{v},\hat{n},\hat{t}) = \int_{\Omega_i} \frac{k_d}{\pi} L_i(x,\hat{l})(\hat{n}\cdot\hat{l})d\hat{l}$$

Damit ist die reflektierte Leuchtdichte nur noch von der Normalen abhängig und wir erhalten:

$$L_r(x,\hat{n}) = \int_{\Omega_i} \frac{k_d}{\pi} L_i(x,\hat{l})(\hat{n}\cdot\hat{l})d\hat{l}$$

Diese Environmentmap speichert die diffuse Beleuchtung im Punkt x. Sie ist über die Flächennormale $\hat{n}$ indiziert.

Spiegelnde Environmentmaps

- Eine rein spiegelnde (spekulare) BRDF ist gegeben durch

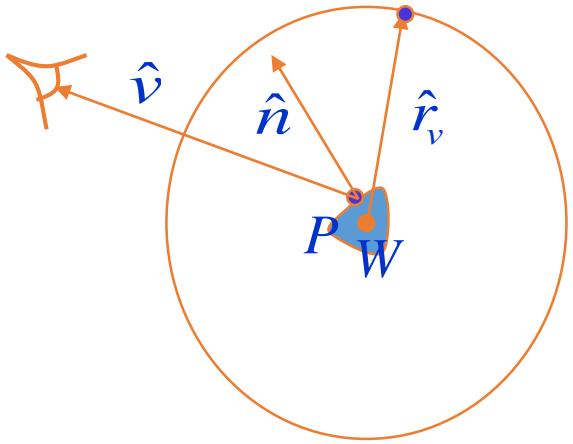
$$\rho\left(\hat{v},\hat{l}\right) = \frac{k_s \delta(\hat{r}_v(\hat{n}) - \hat{l})}{(\hat{n}\cdot\hat{v})},\ k_s \in [0,1]$$

- Damit ergibt sich

$$L_r(x,\hat{v},\hat{n},\hat{t}) = \int_{\Omega_i} \frac{k_s \delta(\hat{r}_v(\hat{n}) - \hat{l})}{(\hat{n}\cdot\hat{v})} L_i(x,\hat{l})(\hat{n}\cdot\hat{l})d\hat{l}$$

$$= \int_{\Omega_i} \frac{k_s \delta(\hat{r}_v(\hat{n}) - \hat{l})}{(\hat{n}\cdot\hat{v})} L_i(x,\hat{l})(\hat{n}\cdot\hat{l})d\hat{l}$$

$$= k_s L_i(x,\hat{r}_v(\hat{n}))$$



- Das entspricht exakt dem von Blinn und Newell vorgeschlagenen Environment-Mapping zur Simulation von Spiegelungen.

Spekulare Environmentmaps

- Die BRDF des Phongmodells ist

$$\rho\left(\hat{v},\hat{l}\right)=\frac{k_s(\hat{r}_v(\hat{n})\cdot\hat{l})^m}{(\hat{n}\cdot\hat{l})},\ k_s\in[0,1],\ m\ >0$$

- Damit ergibt sich

$$L_r(x,\hat{v},\hat{n},\hat{t})=\int_{\Omega_i}\frac{k_s(\hat{r}_v(\hat{n})\cdot\hat{l})^m}{(\hat{n}\cdot\hat{l})}L_i(x,\hat{l})(\hat{n}\cdot\hat{l})d\hat{l}$$

$$=\int_{\Omega_i}k_s(\hat{r}_v(\hat{n})\cdot\hat{l})^m L_i(x,\hat{l})d\hat{l}$$

- Anstatt mit $\hat{n}$ und $\hat{v}$ zu indexieren, kann direkt mit dem Relektionsvektor $\hat{r}_v(\hat{n})$ indexiert werden.

$$L_r(x,\hat{r}_v(\hat{n}))=\int_{\Omega_i}k_s(\hat{r}_v(\hat{n})\cdot\hat{l})^m L_i(x,\hat{l})d\hat{l}$$

- Die Indizierung mit dem Reflektionsvektor $\hat{r}_v(\hat{n})$ wird in OpenGL beispielsweise in Cube-Map Parametrisierung unterstützt.

Environmentmaps mit isotropen BRDFs

Wir approximieren isotrope BRDFs durch BRDFs, die nicht nur isotrop, sondern auch radialsymmetrisch um $\hat{r}_v(\hat{n})$ sind:

$$\rho\left(\hat{v},\hat{l}\right) = p(\hat{n}\cdot\hat{r}_v(\hat{n}),\hat{l}\cdot\hat{r}_v(\hat{n})).$$

Damit ergibt sich

$$L_r(x,\hat{v},\hat{n},\hat{t}) = \int_{\Omega_i} p(\hat{n}\cdot\hat{r}_v(\hat{n}),\hat{l}\cdot\hat{r}_v(\hat{n}))L_i(x,\hat{l})(\hat{n}\cdot\hat{l})d\hat{l}$$

Ist die BRDF hoch spekular, so ist die BRDF fast überall Null, außer wenn

$$\hat{r}_v(\hat{n}) \approx \hat{l} \Rightarrow \hat{n}\cdot\hat{r}_v(\hat{n}) \approx \hat{n}\cdot\hat{l}$$

. Dies liefert eine dreidimensionale Fkt.

$$L_r(x,\hat{r}_v,\hat{n}\cdot\hat{r}_v) = \hat{n}\cdot\hat{r}_v \int_{\Omega_i} p(\hat{n}\cdot\hat{r}_v(\hat{n}),\hat{l}\cdot\hat{r}_v(\hat{n}))L_i(x,\hat{l})d\hat{l}$$

Environmentmaps mit isotropen BRDFs

Approximiert man die BRDF durch ein Tensorprodukt

$$\rho\left(\hat{v},\hat{l}\right) = p(\hat{n}\cdot\hat{r}_v(\hat{n}),\hat{l}\cdot\hat{r}_v(\hat{n})) = F(\hat{n}\cdot\hat{r}_v(\hat{n}))\cdot G(\hat{l}\cdot\hat{r}_v(\hat{n}))$$

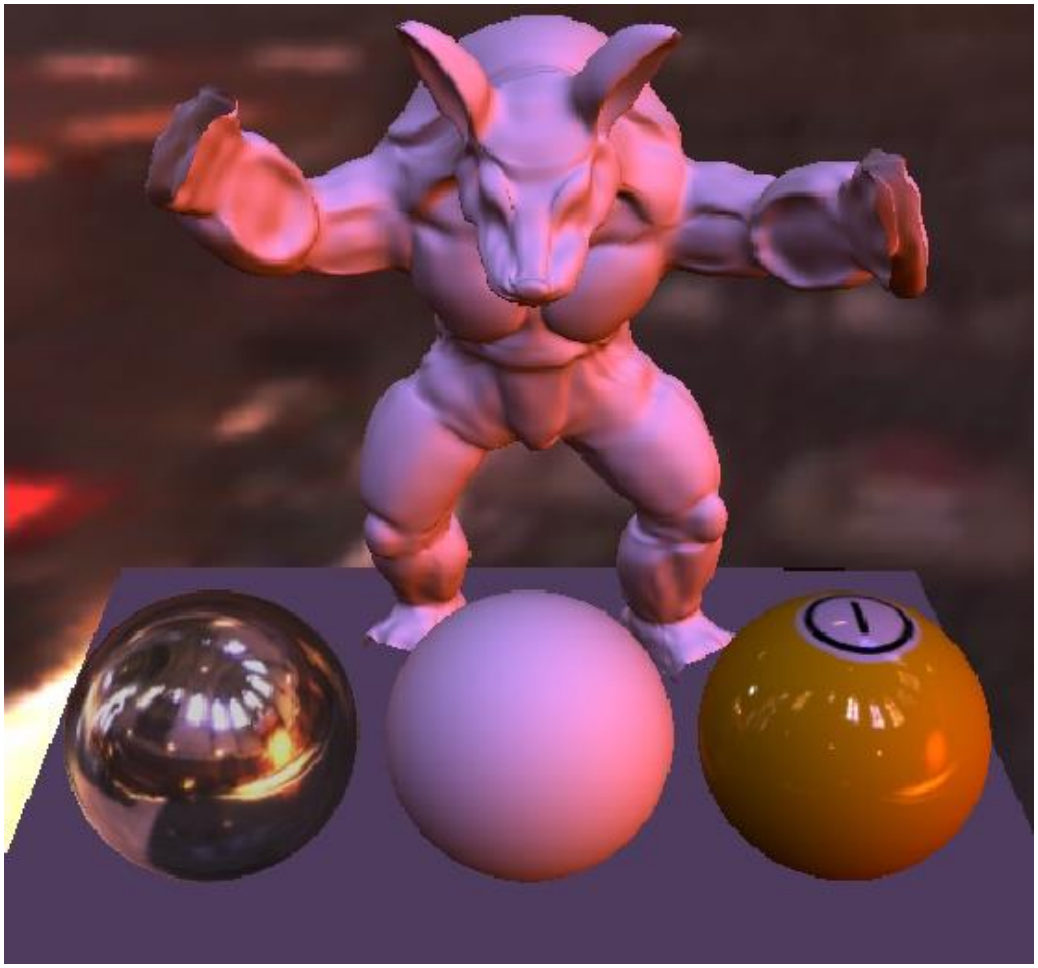
So ergibt sich

$$L_r(x,\hat{r}_v,\hat{n}\cdot\hat{r}_v) = \hat{n}\cdot\hat{r}_v \int_{\Omega_i} p(\hat{n}\cdot\hat{r}_v(\hat{n}),\hat{l}\cdot\hat{r}_v(\hat{n}))L_i(x,\hat{l})d\hat{l}$$

$$\approx (\hat{n}\cdot\hat{r}_v)F(\hat{n}\cdot\hat{r}_v)\int_{\Omega_i} G(\hat{l}\cdot\hat{r}_v(\hat{n}))L_i(x,\hat{l})d\hat{l}$$

Diese Funktion ist nur noch zweidimensional, da die Abhängigkeit von $\hat{n}\cdot\hat{r}_v(\hat{n})$ vor das Integral gezogen werden kann.

Beispiel



Reflection Mapping im Allgemeinen

- Boot auf See (Tafel)