



Real-Time High-Dynamic Range Texture Mapping

Jonathen Cohen, Chris Tchou, Tim Hawkins and Paul Debevec

Präsentiert von Daniel Wickeroth



Einführung – Worum geht's ?

- Darstellung realistischer Szenen
 - Innen - Aussen
- Kein clampen auf 8-Bit-Wertebereich
- Verwenden der Hardwarwe, um Echtzeit zu erreichen



Einführung – Wie funktioniert's ?

- Hinzufügen des „exposure levels“
 - Belichtungszeit
 - Es wird immer nur ein Teil des Wertebereichs dargestellt
- Verwenden von mehreren 8Bit-Texturen
- Zusammenfügen der Texturen durch die Hardware



Einführung – Warum ?

- Texturen enthalten gewöhnlich Radiance Werte, und keine Reflektionskoeffizienten
- HDR-TMs erlauben beliebig grosse Werte
- Darstellung von hellen Reflektionen auf dunklen Flächen



Einführung – Warum ?

- Mit Hilfe der Gamma-Korrektur kann der Wertebereich extrem vergrößert werden.

Beispiel :

Eine 16-Bit-Textur hat den 256-fachen Wertebereich einer 8-Bit-Textur

Mit einem 2,2-gamma-Mapping ergibt sich ein $256^{2,2}$ facher Wertebereich, also ca. Faktor 200.000



Repräsentation und Rendern von HDRTMs Allgemeine Technik

- Die meisten Grafikkarten erlauben nur 8-Bit-Texturen
 - ▶ Deshalb müssen mehrere Texturen zu einer verbunden werden
 - ▶ Das kann die Hardware für uns tun



Repräsentation und Rendern von HDRTMs Allgemeine Technik

$I(v)$: der dargestellte Farbwert

e : exposure level

v : 16-Bit-Textur

Clamp: Funktion, die auf 256 clampt

Bei gegebenem Exposure-Level und
Weißpunkt :

$$I(v) = \text{clamp}(e \cdot v)$$



Repräsentation und Rendern von HDRTMs Allgemeine Technik

Aufspalten der Textur in zwei 8-Bit Texuren v_0 und v_1
wobei v_0 die niedrigen und v_1 die höherwertigen Bits
enthält

$$\begin{aligned} I(v) &= \text{clamp}(e(v_0 + 256 \cdot v_1)) \\ &= \text{clamp}(ev_0 + 256 \cdot e \cdot v_1) \\ &= \text{clamp}(\text{clamp}(e \cdot v_0) + \text{clamp}(256 \cdot e \cdot v_1)) \end{aligned}$$

Mit $\text{clamp}(a+b) = \text{clamp}(\text{clamp}(a)+\text{clamp}(b))$

Repräsentation und Rendern von HDRTMs Allgemeine Technik

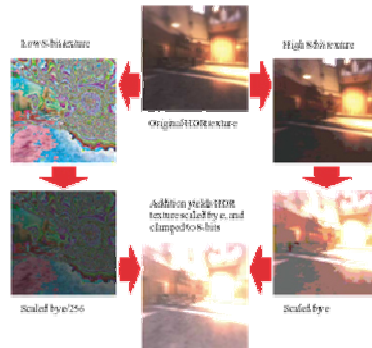


Fig. 1. Overview of HDRTM pipeline.

Repräsentation und Rendern von HDRTMs Allgemeine Technik

Welche Werte sind für e sinnvoll

e	Dargestellter Bereich
$1/512$	0 .. 131072
$1/256$	0 .. 65535
1	0 .. 255
2	0 .. 127,5

→ Einschränkung auf $[1/256 .. 1]$



Repräsentation und Rendern von HDRTMs größere Texturen

- Bei Texturen, die größer sind als 16 Bit, können wir die Gleichung um einen Term erweitern
 - ▶ Das ist nicht gut für die Performanz



Repräsentation und Rendern von HDRTMs größere Texturen

Was tun ?

1. Gegeben sei eine 24 Bit Textur v
2. Berechne 2 neue Texturen x und y mit
 1. $x = v / 256$
 2. $y = \text{clamp}_{(2^{16}-1)} v$

→ Beide können mit 16 Bit dargestellt werden
3. x und y werden in jeweils zwei 8-Bit-Texturen aufgespalten
4. Bei gegebenem e wird $I(v)$ wie folgt berechnet :
falls e in $[1/256 .. 1]$
$$I(v) = \text{clamp}(\text{clamp}(e \cdot y_0) + \text{clamp}(256 \cdot e \cdot y_1))$$

falls e in $[1/256^2 .. 1/256]$
$$I(v) = \text{clamp}(\text{clamp}(256 \cdot e \cdot x_0) + \text{clamp}(256^2 \cdot e \cdot x_1))$$



Repräsentation und Rendern von HDRTMs größere Texturen

Allgemein gilt :

- Die Anzahl der benötigten Texturen, um eine $8n$ -Textur mit $n \geq 2$ abzuspeichern ist $2n-2$
- Der nutzbare exposure-level Bereich ist $[256^{1-n}..1]$



Repräsentation und Rendern von HDRTMs Gamma-Korrektur

- Die meisten Ausgabegeräte machen eine Gamma-Korrektur
- Um korrekte Werte zu erzeugen, müssen wir das auch machen
- Mit $1/\gamma$ potenzieren
 $(e \cdot v)^{1/\gamma} = e^{1/\gamma} \cdot v^{1/\gamma}$
 $e' = e^{1/\gamma} \quad v' = v^{1/\gamma}$
 $I(v)^{1/\gamma} = \text{clamp}(v'e')$
 $= \text{clamp}(\text{clamp}(e'v0') + \text{clamp}(256e'v1'))$



Repräsentation und Rendern von HDRTMs Implementation auf SGI

- Voraussetzung sind Gamma-Korrigierte 16-Bit Werte
 1. Die höheren Bits in den Framebuffer rendern
 2. Blending ausschalten
 3. Pixeltransferfunktion so setzen, daß alle Werte mit $256e'$ skaliert werden
 4. GLCopyPixel aufrufen → Der Framebuffer wird in sich selbst zurückgelesen
 5. Additive Blending einschalten
 6. Aktuelle Farbe (e', e', e')
 7. Texture Environment so setzen, daß alle Texturwerte mit der aktuellen Farbe moduliert werden.
 8. Die unteren 8 Bits in den Framebuffer rendern



Repräsentation und Rendern von HDRTMs Implementation auf Nvidia GeForce 2

- Problem : keine direkte Unterstützung für Multiplikation mit Werten die größer als 1 sind
- 2 Texture Combiner, die jeweils 4 Werte (A,B,C,D) verarbeiten. Ergebnis : $(A \cdot B) + (C \cdot D)$

Exposure Range	Combiner Name	A	B	C	D	Final Multiplier
$e \in [\frac{1}{2}, 1]$	TC0	x_1	1	x_1	1	x_4
	TC1	TC0	e	x_0	$e/2$	x_2
$e \in [\frac{1}{4}, \frac{1}{2}]$	TC0	x_1	1	x_1	1	x_2
	TC1	TC0	$2e$	x_0	$e/2$	x_2
$e \in [\frac{1}{8}, \frac{1}{4}]$	TC0	x_1	1	x_1	1	x_1
	TC1	TC0	$4e$	x_0	$e/2$	x_2
$e \in [0, \frac{1}{8}]$	TC0	x_1	1	0	0	x_1
	TC1	TC0	$8e$	x_0	$e/2$	x_2



Repräsentation und Rendern von HDRTMs Implementation auf Nvidia GeForce 2

- Die Combiner sind so eingestellt, dass eine zuvor geclampete Zahl nie mit weniger als $\frac{1}{2}$ multipliziert wird
- Nach der abschließenden Multiplikation mit 2 ist der dann auf jeden fall größer als 255
- Problem : Wir können nur mit 16 multiplizieren
 - ▶ Wir benötigen $4n-4$ Texturen
 - ▶ wir können aber immer nur 2 Texturen gleichzeitig verwenden
 - ▶ Wir brauchen einen Trick



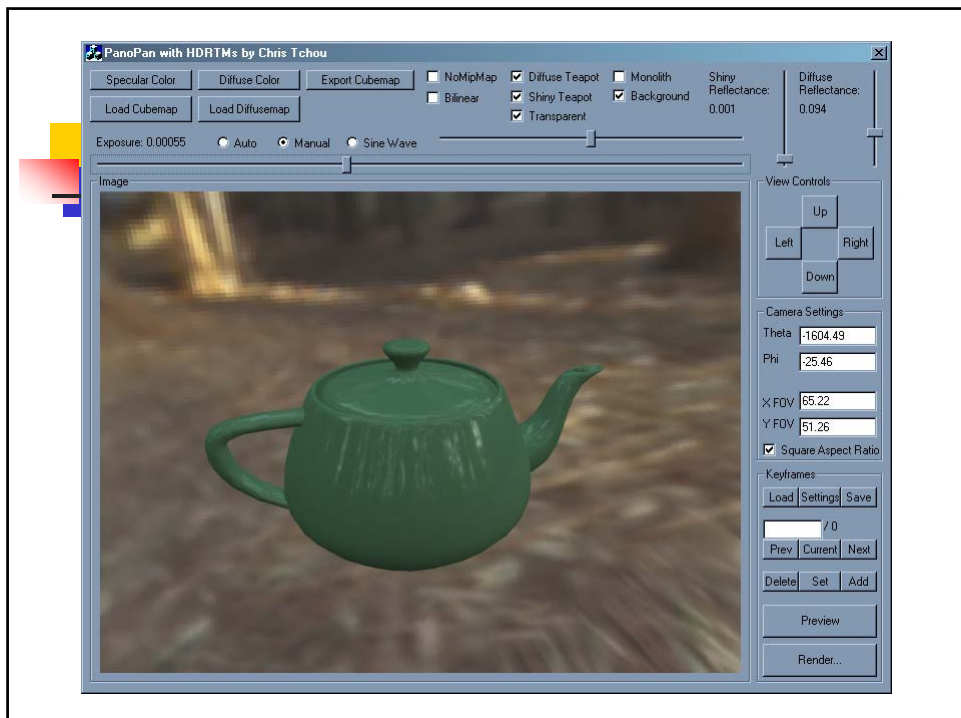
Repräsentation und Rendern von HDRTMs Implementation auf Nvidia GeForce 2 / Trick

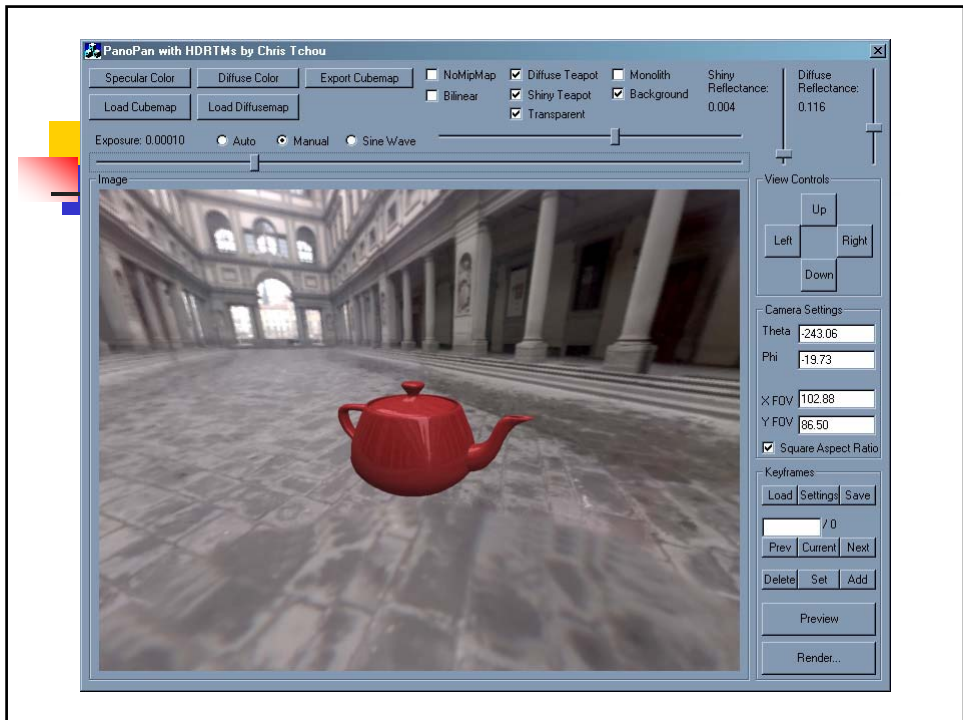
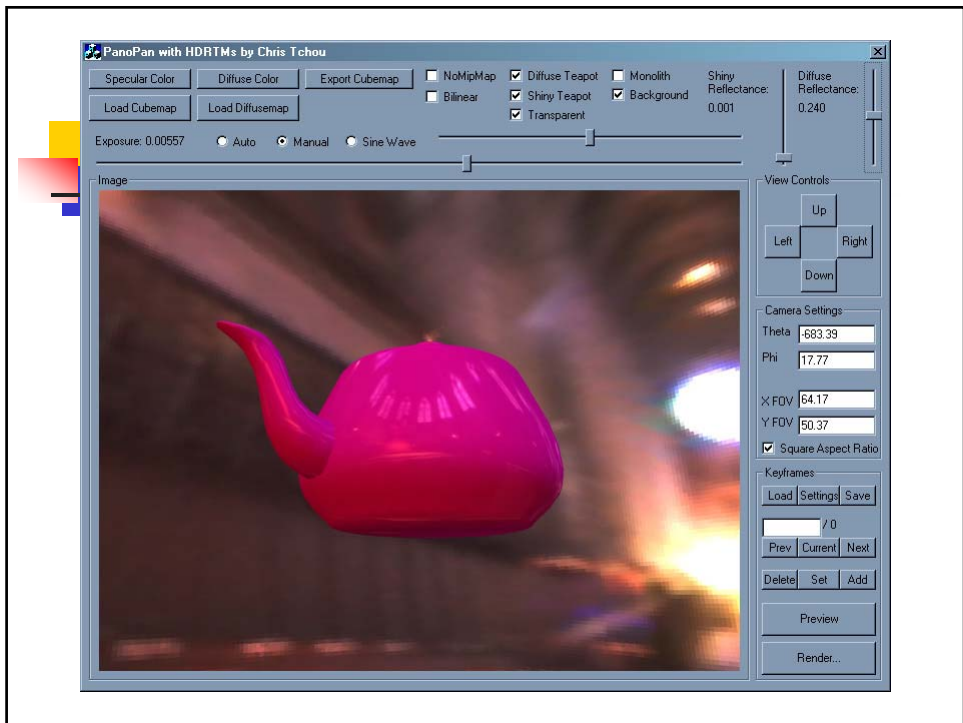
- Wir erstellen zunächst die bisher verwendeten Texturen x_0 und x_1 , wobei x_0 die niedrigen und x_1 die höherwertigen Bits enthält
- Dann erstellen wir eine dritte Textur z mit
$$z = \text{clamp}(16 \cdot \text{clamp}(x_1))$$
- Wir erhalten $\text{clamp}(256x_1)$ durch $\text{clamp}(16z)$
 - ▶ Falls $e > 1/16$ ersetzen wir x_1 durch z im texture combiner setup
 - ▶ Falls $e < 1/16$ multiplizieren wir e mit 16, teilen B und D in der TC1 Stage durch 16 und verwenden x_1 wie in der Tabelle

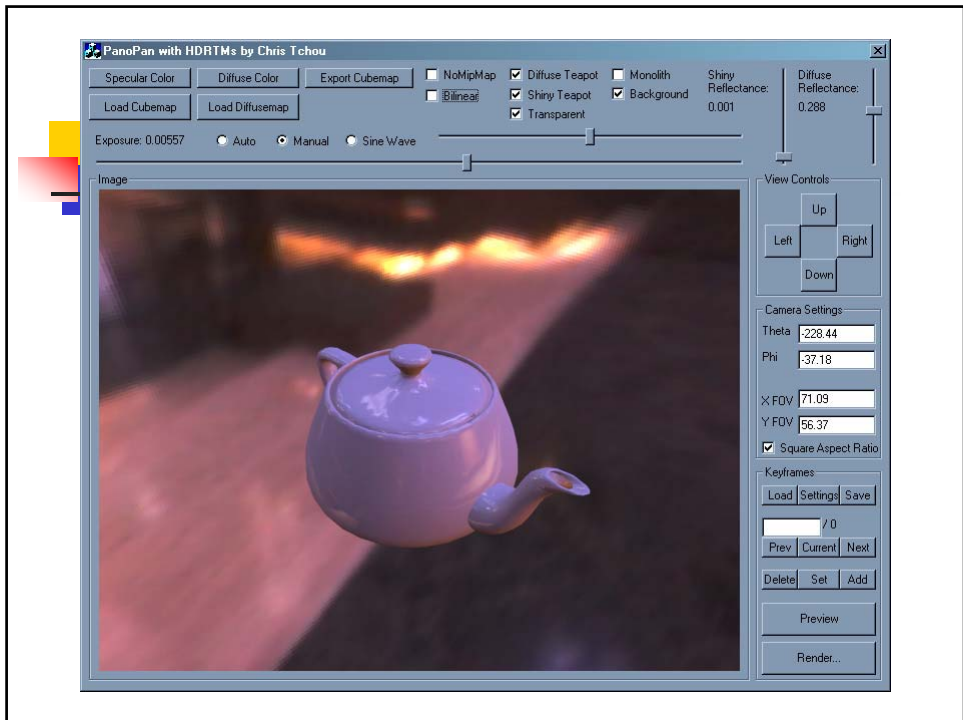
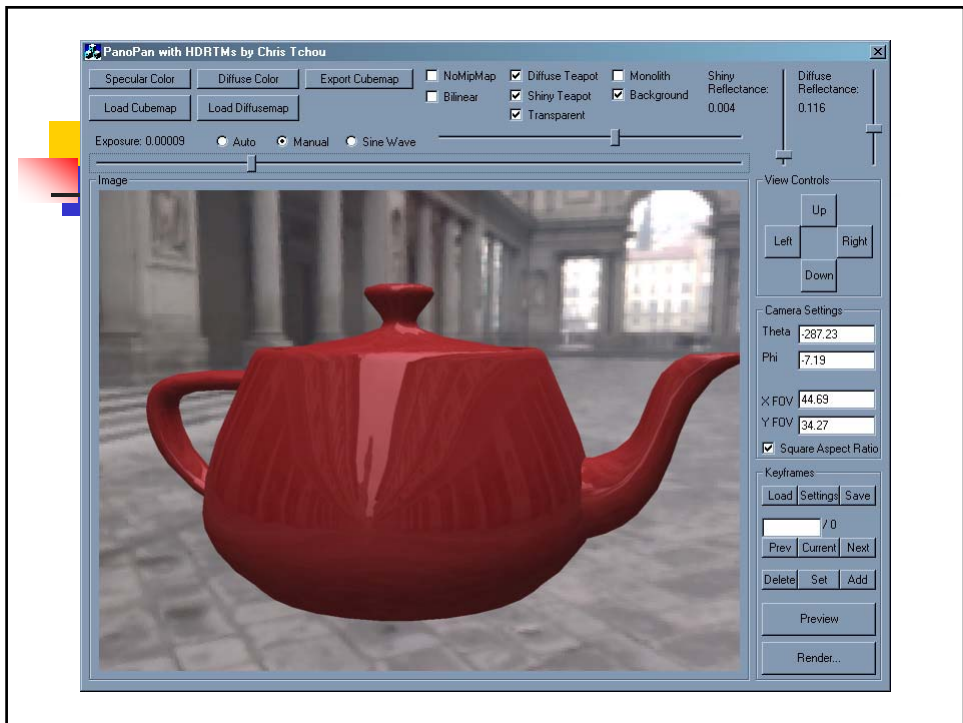
Anwendungen

Visualisierung einer HDR-Szene

- Es wurde ein High Dynamic Range Image Panorama Viewer geschrieben (Jonathan Cohen)
- Cube based Environment Mapping
- DOF's : Rotation, Zoom, Exposure Level



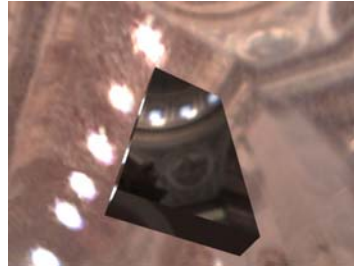




Anwendungen

Lighting

- Mit Hilfe von HDRTM können scharfe spekulare Spiegelungen auf dunklen Objekten gerendert werden.
 - ▶ Teapot, fresnel Reflektion auf dunklem Monolithen



Anwendungen

Lighting

- Berechnung der Position der Spiegelung mit
 - ▶ Environment mapping
 - ▶ Ändern der Kamera Projektionsmatrix in Abhängigkeit der Position des Reflektors
- Exakte Berechnung der Position, aber die Intensität stimmt nicht
 - ▶ Reflektierte Farbe in full dynamic range berechnen
 - ▶ Spekularer Reflektionskoeffizient Φ
 - ▶ $I(v) = \text{clamp}(e\Phi v) \rightarrow e' = e \cdot \Phi$
 - ▶ Gamma-Korrektur $e' = (e \cdot \Phi)^{1/\gamma}$



Anwendungen

Lighting

- Rendern von spiegelnden Materialien (glossy)
 1. Das Objekt mit einer vorverarbeiteten diffusen Textur rendern
 2. Dann noch mal mit einer dunkleren, spekularen environment map rendern
 3. Die Ergebnisse werden im Framebuffer addiert
- Ist zwar physikalisch nicht korrekt, wegen der gamma-Korrektur, sieht aber gut aus.