Texturen auf virtuellen Wänden Seminararbeit zum Seminar „Effiziente 3D-Renderingalgorithmen“

Universität Paderborn
Fachbereich 17

Anja Austermann

23. Juni 2002

Inhaltsverzeichnis

1 Einsatzgebiete

Grafikanwendungen stellen besondere Anforderungen an die Geschwindigkeit der Hardware. Insbesondere wenn komplexe dreidimensionale Umgebungen in Echtzeit berechnet und durchlaufen werden sollen, ist es notwendig, die Anzahl der zu rendernden Polygone sinnvoll zu reduzieren, ohne daß dabei der optische Eindruck verloren geht. Da der Aufwand und die benötigte Zeit für die Berechnung eines Frames direkt mit der Anzahl der zu rendernden Polygone zusammenhängen, wird versucht, die Polygonanzahl möglichst klein zu halten. Aufgrund der begrenzten Leistungsfähigkeit aktueller Grafikhardware ist es bei sehr großen Polygonanzahlen sonst nicht mehr möglich, genügend hohe Frameraten zu erzielen, die notwendig sind, damit ein Benutzer keine Sprünge zwischen den einzelnen Frames wahrnimmt.
Diese Seminararbeit behandelt das Verwenden von Virtuellen Wänden als eine von vielen Möglichkeiten zur Polygonreduktion, die gerade in sehr polygonreichen Umgebungen verwendet wird, da sie einen großen Performancegewinn bietet.
Eine virtuelle Wand ist ein Viereck, das in eine komplexe Szene eingefügt wird und die hinter ihr liegende Geometrie verdeckt. Auf diese virtuelle Wand wird eine Textur gezeichnet. Diese Textur wird berechnet, indem die sich hinter ihr befindenden Umgebung auf die virtuelle Wand projiziert wird. Durch die Vorabberechnung der Texturen kann bei der Navigation viel Rechenzeit eingespart werden, da alle Polygone, die sich hinter der Wand befinden, nicht berechnet werden müssen.

2 Überblick

Dieser Abschnitt behandelt einige der gängigsten Verfahren zur Polygonreduktion sowie verschiedene texturbasierte Verfahren, die als Grundlage für das Verfahren der virtuellen Wände benutzt wurden.

2.1 Herkömmliche Verfahren zur Beschleunigung des 3D-Renderings

Um die Anzahl der Polygone in einer Szene zu verringern, werden traditionell eine Reihe unterschiedlicher Verfahren eingesetzt, wie beispielsweise Level-of-Detail und Visibility-Culling-Algorithmen.
Level-of-Detail-Algorithmen [2] basieren auf der Grundidee, verschieden detaillierte Versionen von komplexen Objekten zu erzeugen und vereinfachte, also in der Anzahl ihrer Polygone reduzierte Versionen statt der Orignalobjekte zu rendern, wenn die Objekte weit vom Betrachter entfernt sind. Gerade bei einer Navigation durch Szenen, die viele Objekte beinhalten, die aber die meiste Zeit so weit vom Betrachter entfernt sind, daß er nur wenige Details wahrnehmen kann, kann das Rendering der gesamten Szene durch den Einsatz von LoD-Algorithmen deutlich beschleunigt werden.
Das Prinzip des Visibility Cullings [3] hingegen wird verwendet, um das Rendern von Objekten zu unterbinden, die vom Standpunkt des Betrachters nicht sichtbar sind, da sie nicht in dessen Blickfeld liegen oder von anderen Objekten vollständig verdeckt sind.

2.2 Grundideen von texturbasierten Verfahren

In komplexen Szenen reichen diese beiden Verfahren allerdings häufig nicht aus, um die Anzahl der Polygone soweit zu reduzieren, daß eine Navigation in Echtzeit möglich ist.
Aus diesem Grund können texturbasierte Verfahren eingesetzt werden, die den Vorteil haben, daß bestimmte Teile der Szene nicht in jedem Animationsschritt neu gerendert werden müssen, sondern vorab berechnet und während der Animation in die Szene projiziert werden können.
Es gibt verschiedene Ansätze für texturbasierte Renderingverfahren, die sich insbesondere darin unterscheiden, ob jedes zu berechnende Objekt einzeln durch ein einfacheres Objekt mit Textur ersetzt wird, oder ob diese in Gruppen zusammengefasst werden, und wie mit den zwangsläufig entstehenden Projektionsfehlern verfahren wird.
Texturbasierte Verfahren zur Verminderung der Rechenzeit basieren prinzipiell auf der gleichen Grundidee, wie das beispielsweise in OpenGL zur Verfügung stehende Environment-Mapping. Environment-Mapping wird eingesetzt, um das zeitaufwändige, rekursive Berechnen von Spiegelungen durch ein einfacheres Verfahren zu ersetzen. Hierzu wird ein „spiegelndes“ Objekt in einer Szene mit einer Textur versehen, die eine Projektion der Umgebung auf das betreffende Objekt zeigt.

2.3 Texturierte, virtuelle Wände als Erweiterung der bisherigen Verfahren

Das Verwenden von texturierten virtuellen Wänden [1] ist eine Erweiterung der bisherigen texturbasierten Verfahren, die zwar in Außenszenen, bei denen Darstellungsfehler weniger ins Auge fallen, nützlich sind, sich aber bei Innenszenen als wenig brauchbar erweisen.
Virtuelle Wände unterteilen einen Raum in einzelne Zellen. Alles, was sich zusammen mit dem Betrachter in einer Zelle befindet, wird vom Programm gerendert. Auf die virtuellen Wände, die die Zelle des Betrachters begrenzen, zeichnet der Renderer dem Standort des Betrachters möglichst gut angepasste Texturen. Der Betrachter sieht dadurch eine vollständige Szene, obwohl nur die Objekte tatsächlich während der Navigation gerendert werden, die sich in der gleichen Zelle befinden.

3 Das Verfahren

Die oben beschriebenen Grundideen wurden an der Universität Paderborn in ein praxistaugliches Verfahren umgesetzt, das im folgenden Abschnitt beschrieben und näher erläutert wird.

3.1 Die zwei Phasen des Verfahrens

In der Praxis besteht das Verfahren, das Rendering durch virtuelle Wände zu unterstützen, aus zwei Phasen.

- In der ersten Phase, der Preprocessing-Phase, wird eine Menge virtueller Wände in die Szene eingefügt, die die große Szene in verschiedene Zellen unterteilen. Diese Phase wird nur einmal nach dem Erstellen oder Verändern der Szene durchgeführt.

- In der zweiten Phase, der Durchlaufphase, navigiert ein Betrachter durch die Szene. In dieser Phase werden die jeweils passenden Texturen auf die Virtuellen Wände projiziert und die Szene dargestellt.

3.1.1 Preprocessing-Phase

Die Eingabe für die Preprocessing-Phase ist eine Repräsentation des 3D-Modells einer komplexen Innenraumszene. Innerhalb einer solchen Szene kommen neben den Außendwänden im Regelfall keine realen Wände oder andere Objekte, die große Teile der Szene verdecken, vor.
In das 3D-Modell der Szene werden manuell, also beispielsweise mit Hilfe eines 3D-Modelling-Programms einige virtuelle Wände eingefügt und die Szene somit in Zellen unterteilt.
Die veränderte Szene inklusive der virtuellen Wände wird daraufhin mit Hilfe eines Algorithmus weiterverarbeitet. Dieser Algorithmus unterteilt die Zellen in Zonen, für die jeweils eine bestimmte Textur verwendet werden soll. Hierzu wird eine Projektionsfehlermetrik verwendet, die im Abschnitt 4 genauer erläutert ist.
Zusätzlich berechnet der Algorithmus den Texture-Sampling-Punkt für jede einzelne Zone, also den Punkt, an dem sich das Projektionszentrum bei einer Projektion der hinter der virtuellen Wand liegenden Geometrie auf diese Wand befindet.
Im letzten Schritt werden während der Preprocessing-Phase an allen berechneten Texture Sampling Punkten die Texturen als Off-Axis-Projektionen der hinter der virtuellen Wand liegenden Geometrie auf die Wand berechnet.
Eine Off-Axis-Projektion ist eine Projektion, bei der die Gerade zwischen Mittelpunkt der virtuellen Wand und Projektionszentrum nicht senkrecht auf der virtuellen Wand steht. Dies führt zu einer verzerrten Darstellung, wie in Abbildung 1 zu sehen ist.

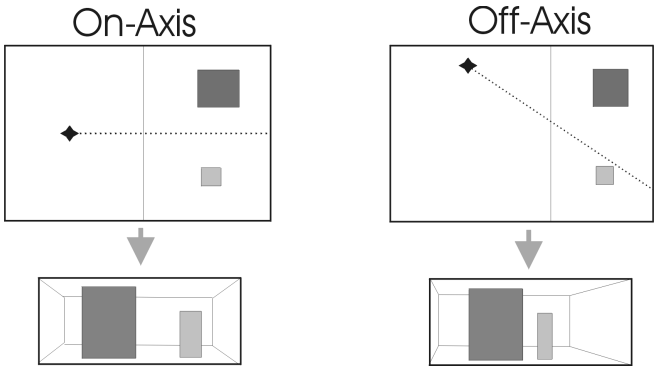

Abbildung 1: On-Axis- und Off-Axis-Projektion

3.1.2 Durchlaufphase

Das Programm für die Navigation durch die Szene bekommt als Eingabe das Ergebnis der Preprocessing-Phase übergeben.

Es berechnet in einem ersten Schritt, welche Virtuellen Wände anzuzeigen sind, welche Texturen auf diese Wände projiziert werden sollen, wo Level-of-Detail-Algorithmen einsetzbar sind und welche Objekte durch virtuelle Wände oder andere Objekte verdeckt sind, so daß Visibility Culling angewandt werden kann. Die Berechnungen erfolgen mit Hilfe eines Scene-Graph-Durchlaufs. Für das Visibility-Culling verhalten sich die im ersten Schritt eingefügten virtuellen Wände genau wie reale, undurchsichtige Wände. Sie verdecken den Teil der Szene, der sich hinter ihnen befindet. Stattdessen wird die Textur angezeigt, die sich auf der Wand befindet.
Texturen passen natürlich nur dann exakt, wenn der Benutzer sich genau an demselben Punkt befindet, von dem aus die Textur berechnet wurde. In allen anderen Fällen stellen sie lediglich eine Annäherung an die reale Szene dar. Eine solche Form der Darstellung nimmt also bewußt in Kauf, daß Darstellungsfehler auftreten, die sich darin äußern, daß die Perspektive der Textur auf der virtuellen Wand und die Perspektive, aus der der Betrachter auf die während der Navigation berechnete Geometrie im Vordergrund sieht, nicht hundertprozentig zusammenpassen. So lange der Projektionsfehler ausreichend klein ist, kann ein menschlicher Betrachter diese Ungenauigkeit allerdings nicht wahrnehmen.
Der Darstellungsfehler lässt sich berechnen. Wenn er zu groß wird, also eine festgelegte Fehlerschranke überschritten wird, ersetzt das Programm die aktuelle Textur je nach Situation durch eine geeignetere Textur oder durch gerenderte Geometrie. Um den Texturwechsel nicht zu auffällig werden zu lassen, erfolgt eine Überblendung von der alten zur neuen Textur. Die alte Textur wird zunehmend transparenter, während die neue Textur immer deutlicher sichtbar wird.
Nachdem die günstigste Textur und die zu rendernden Objekte ausgewählt wurden, rendert das Programm die Geometrie und die mit Texturen versehenen virtuellen Wände innerhalb der aktuellen Zelle und stellt das Ergebnis auf dem Bildschirm dar.
In Abbildung 2 ist eine virtuelle Wand und die Projektion der dahinter liegenden Geometrie auf diese Wand zu sehen.

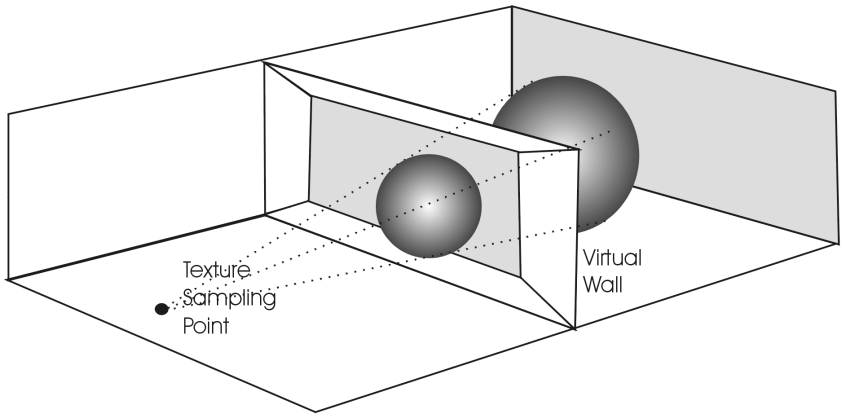

Abbildung 2: Berechnung der Texturen

3.2 Besonderheiten der technischen Realisierung

Das Verfahren der texturierten, virtuellen Wände und insbesondere die Berechnung der Projektionsfehler, die im Abschnitt 4 erklärt ist, sind in dem dieser Arbeit zugrunde liegenden Paper auf einen Spezialfall vereinfacht. Zum einen besitzen die Szenen alle eine rechteckige Grundfläche, was die Berechnung einer sinnvollen Anzahl virtueller Wände und Zonen vereinfacht. Zum anderen kann der Betrachter nur in zwei Dimensionen navigieren. Sein Abstand zum Boden, also die dritte Dimension, ändert sich nicht. Das vereinfacht die Berechnung des Projektionsfehlers. Prinzipiell sind diese Beschränkungen allerdings nicht notwendig. Das Verfahren lässt sich einfach erweitern, um auch anders geformte Grundflächen und variable Höhen der Betrachterposition zu erlauben.
Die Projektion zur Berechnung einer Textur ist im Regelfall eine sogenannte Off-Axis-Projektion. Das Projektionszentrum dieser Projektion wird auch Texture Sampling Punkt genannt. Die Objekte, die letztendlich auf der Textur zu sehen sein sollen, werden vor der Berechnung der Textur durch einen SceneGraph-Durchlauf ermittelt.
Nach Abschluß der Texturberechnungen werden alle in den verschiedenen Texture Sampling Punkten gerenderten Texturen gespeichert. Die Texturen selbst und die Informationen, für welche Positionen des Betrachters sie gültig sind, werden als zusätzliche Informationen in den Scene Graph mit aufgenommen, damit sie während der Echtzeit-Navigation keine Rechenzeit mehr für die Suche der passenden Textur verbraucht wird.
Ein Aspekt, der für die Performance des Programms wichtig ist, ist daß nicht alle Texturen zu jedem Zeitpunkt im Texturspeicher der Grafikhardware gehalten werden müssen. Davon ausgehend, daß der Betrachter sich nur von einer Zone in eine angrenzende Zone bewegen kann, ist es erlaubt, lediglich die Textur der aktuellen Zone und die der angrenzenden Zonen im Speicher zu halten
Einen Spezialfall bei der Berechnung stellen solche Objekte dar, die eine virtuelle Wand schneiden. Von einem solchen Objekt befindet sich ein Teil vor der Wand und ein Teil dahinter. Die Objekte nicht in der Durchlaufphase zu rendern sondern bereits in die Textur der virtuellen Wand mit aufzunehmen, würde dazu führen, daß an dieser Stelle der Projektionsfehler vergrößert wird und nicht mehr so problemlos zu berechnen ist, da das Verfahren zur Bestimmung des Projektionsfehlers davon ausgeht, daß sich alle auf die virtuellen Wand projizierten Objekte vom Betrachter aus gesehen hinter der Wand befinden. Aus diesem Grund ist es zweckmäßig, derartige Objekte zur Laufzeit, also in der Durchlaufphase zu berechnen.
Bei einem Texturwechsel aufgrund eines Überschreitens der Fehlerschranke wären deutliche Sprünge sichtbar, wenn aprupt von einem Frame zum nächsten die Textur geändert würde. Damit der Übergang zwischen zwei Texturen oder auch zwischen Texturen und Geometrie möglichst weich ist, werden Blending-Techniken verwendet. Blending ist im Wesentlichen die Anwendung von Transparenz. Zwischen zwei Zonen gibt es jeweils einen Übergangsbereich, in dem zwei Texturen simultan angezeigt werden. Bei einer Bewegung von Zone A mit Textur A nach Zone B mit Textur B verblasst Textur A zunehmend, während Textur B zunehmend deutlicher sichtbar wird. Genauso funktioniert auch das Blending, wenn der Benutzer sich einer virtuellen Wand nähert und sie durchquert. Ab einem bestimmten Punkt, sind gerenderte Geometrie der Zelle hinter der virtuellen Wand und die Textur der virtuellen Wand simultan sichtbar. Die Textur der virtuellen Wand verblasst beim Übergang in die neue Zelle.

4 Fehlermaße und Texturwechsel

Der letzte Abschnitt weist bereits darauf hin, daß der Projektionsfehler exakt bestimmt und zum Auslösen eines Texturwechsels verwendet werden kann. In diesem Abschnitt wird das Berechnen des Projektionsfehlers genauer erläutert.
Der Fehler ist gleich null, wenn der Betrachter sich im Texture Sampling Punkt, also dem Punkt befindet, der als Projektionszentrum für das Rendering der Textur verwendet wurde. Bewegt sich der Betrachter parallel zur virtuellen Wand vom Texture Sampling Punkt weg, so vergrößert sich der Projektionsfehler. Er wird ebenfalls größer, wenn sich der Betrachter näher zur Wand hin oder weiter von ihr weg bewegt.
Der Projektionsfehler ist definiert als:

$$f = \Delta b \frac{xP - xT}{xP}$$

f steht hierbei für den Projektionsfehler, also die Distanz zwischen der Abbildung eines Objektpunktes in der Textur und der Abbildung des Punktes bei einer korrekten Projektion, Δb ist die Distanz zwischen dem Texture Sampling Punkt und dem Standpunkt des Betrachtes, die beide auf einer Parallele zur Virtuellen Wand liegen. xP ist die Differenz der X-Werte der Betrachterposition und der Position des darzustellenden Punktes hinter der virtuellen Wand und xT die Differenz der X-Werte von Betrachterposition und virtueller Wand, wie in der Abbildung 3 zu sehen ist.

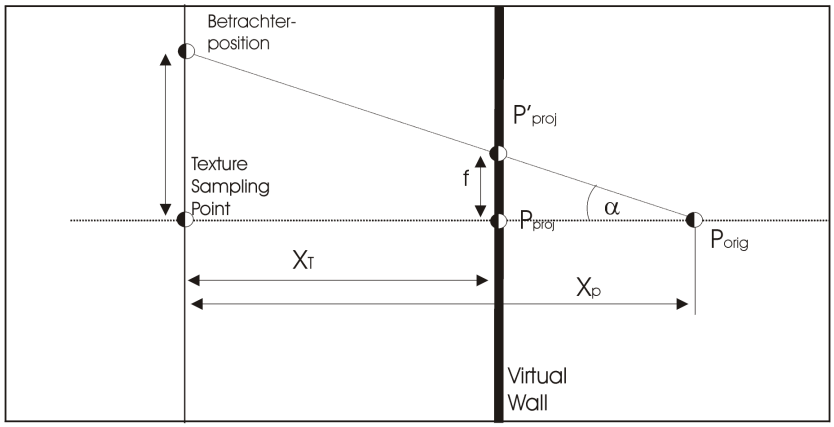

Abbildung 3: Darstellungsfehler

Neben der Fehlerdistanz kann auch der Fehlerwinkel, also der Winkel zwischen den Geraden vom Projektionszentrum zum Punkt P und vom Projektionszentrum zum Punkt P' berechnet werden.

$\alpha = atan(\frac{f}{xP - xT}) = atan(\frac{\Delta b}{xP})$

4.1 Zellen und Zonen

Um den Texturwechsel zu systematisieren, statt zur Laufzeit die Projektionsfehler zu berechnen, wird jede Zelle in mehrere Zonen eingeteilt, innerhalb derer eine bestimmte Textur für die virtuelle Wand gültig ist. Wie weit eine Zone mindestens von der virtuellen Wand entfernt sein muß, hängt davon ab, wie groß der maximale, gerade noch akzeptable Fehler ist.
Der maximale Fehler tritt unter zwei Bedingungen auf:

- Der aktuelle Standpunkt des Benutzers befindet sich in einer der Ecken einer Zone, die sich nah wie möglich an der virtuellen Wand befinden.
- Der auf die Textur zu projizierende Punkt befindet sich ebenfalls so nah wie möglich an der virtuellen Wand.

In Abbildung 4 ist zu sehen, wie eine Szene in Zonen eingeteilt wird. Für jede Zone gibt es für jede begrenzende virtuelle Wand eine Textur. Steht der Benutzer in der Zone D1, so wird die Geometrie in der Zelle D direkt gerendert und nicht für die Berechnung einer Textur verwendet. Als Projektionszentrum für die Texturen der virtuellen Wände wird der Mittelpunkt der Zone D1 verwendet.

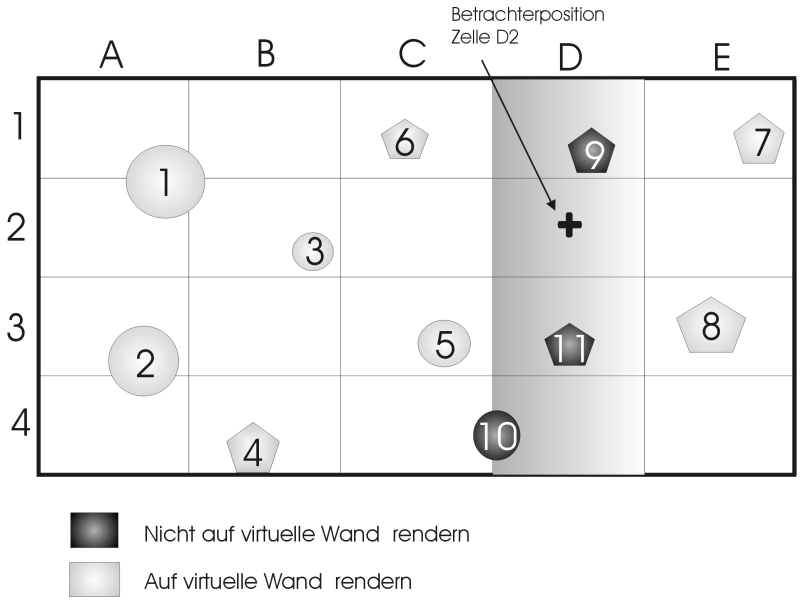

Abbildung 4: Einteilung in Zonen

Legt man den maximalen, gerade noch erlaubten, Fehlerwinkel α fest, so kann man berechnen, wie weit eine Zone mindestens von der virtuellen Wand entfernt sein muß, wenn sich der Betrachter um Δb vom Texture-Sampling-Point entfernt. Als sinnvoll hat sich hier ein maximaler Fehlerwinkel von 15° oder besser 10° erwiesen. Der Abstand lässt sich nach folgender Formel berechnen:

$xTmin = \frac{\Delta b}{tan\alpha}$

Aus diesem Ergebnis lässt sich jetzt die maximale Breite zG einer Zelle ermitteln, die von einer Anzahl nT Zonen also einer Anzahl von nT verschiedenen Texturen abgedeckt werden kann.

$zG = 2\Delta b * nT$

5 Laufzeitbetrachtungen

Virtuelle Wände können zur Verbesserung der Performance in der Echtzeitnavigation durch 3D-Szenen eingesetzt werden. Daher stellt sich die Frage, wie groß der Performancegewinn wirklich ist, also wieviel Rechenzeit tatsächlich eingespart werden kann, wenn neben den gängigen Verfahren wie Visibility Culling und Level-of-Detail-Algorithmen virtuelle Wände zum Einsatz kommen. Generell ist zu sagen, daß es hier einen Tradeoff zwischen Rechenzeit, Speicherplatzverbrauch und Qualität der Szenendarstellung gibt. Je mehr virtuelle Wände man in eine Szene einzieht, desto kleiner werden die einzelnen Zellen und desto weniger Geometrie muß zur Laufzeit gerendered werden, wodurch sich eine Verbesserung des Laufzeitverhaltens ergibt. Allerdings müssen bei einer größer werdenden Anzahl von Zellen mehr Texturen berechnet und gespeichert werden. Die Anzahl der Texturen lässt sich verringern, indem man die Fehlerschwelle heraufsetzt, was aber zu einer Verringerung der Qualität der Szenendarstellung führt.

6 Zukunftsausblick

Weitere Versuche sind insbesondere für die Berechnung eines optimalen Fehlerwinkels notwendig, der sich aus den Eigenschaften der aktuellen Szene ergibt. Außerdem wird derzeit an einer Methode gearbeitet, virtuelle Wände automatisch in die Szene einzufügen, sowie an einer Möglichkeit, den immensen Speicherplatzverbrauch durch die großen Mengen vorberechneter Texturen in einer komplexen Szene zu reduzieren. Hier sind auch Methoden in der Diskussion, die Texturen zur Laufzeit berechnen.

Literatur

[1] Ebbesmeyer P.: Textured Virtual Walls - Archieving Interactive Frame Rates During Walkthroughs of Complex Indoor Environments. IEEE Virtual Reality Annual International Symposium VRAIS 1998, Atlanta 14.3.1998 18.3.1998

[2] Watt A.: 3D-Computer Graphics. Addison-Wesley, 1999

[3] Foley, J. D.; van Dam, A.; Feiner, S. K.; Hughes, J. F.; Phillips, R. L.: Computer Graphics Principles and Practice. Addison-Wesley, 1990.