Realtime-Rendering Nicht-Photorealistischer Computergraphiken mit OpenGL

Laborpraktikum,

Niklas Röber
Otto-von-Guericke Universität, Magdeburg
nroeber@cs.uni-magdeburg.de

6. November 2001

Zusammenfassung

Die Nicht-Photorealistische Computergraphik hat sich in den letzten Jahren sehr stark entwickelt. Die Vorteile dieser alternativen Darstellungsmethoden sind vielfältig. Zum einen erkennt der Betrachter ihm gut vertraute Zeichenmethoden, wie z.B. Skizzen oder Ölzeichnungen, zum anderen werden auch nicht mehr alle Details einer Szene gleich stark dargestellt. So ist es möglich, den Betrachter auf bestimmte Bildausschnitte zu fokussieren. Dies kann das Verstehen und Begreifen einer Szene bzw. einer dargestellten Handlung stark vereinfachen. Viele dieser Techniken sind aber nicht Echtzeitfähig, was das "Erleben" einer Szenerie beeinträchtigen kann. Ziel dieses Laborpraktikums war es, einige dieser Techniken in ein Echtzeitsystem zu integrieren, um eine Interaktion mit der Szene zu ermöglichen. Zum Einsatz kam dabei die Fly3D Engine [Fly3D], welche auf der Basis von OpenGL implementiert ist.

1 Einführung

Für Studenten der Otto-von-Guericke Universität Magdeburg in der Fachrichtung Informatik ist im 9. Semester ein Laborpraktikum als Pflichtveranstaltung vorgesehen. Ziel dieses Praktikums ist es, Studenten an wissenschaftliche Arbeiten heranzuführen, und ihnen die Möglichkeit zu geben sich mit dem Thema dieses Praktikums auf die anstehende Diplomarbeit vorzubereiten. In der Diplomarbeit kann so auf das im Laborpraktikum erworbene Wissen aufgebaut werden.
Ziel meines Laborpraktikums war die technische Umsetzung Nicht-Photo-Realistischer Renderingtechniken in eine 3D Computerspiele Engine. Dabei wurde die Fly3D Engine ausgewählt. Diese ist komplett in C++ implementiert und momentan nur für Windows erhältlich. Die für die graphische Darstellung der Szene notwendigen Algorithmen benutzen OpenGL als Interface [Fly3D]. Dadurch ist zugleich eine breite Unterstützung an Gaphikhardware gewährleistet. Eine Anbindung an 3D Studio MAX erlaubt eine einfache Integration vorhandener 3D Szenen. Während des Praktikums wurden mit dem 3D Studio MAX Szenen erstellt, welche dann in die Fly3D Engine importiert wurden. Für die Darstellung der verschiedenen Nicht-Photorealistischen Stile wurden mit Microsoft Visual C++ [VisualC++] mehrere Plugins entwickelt, welche in die Spiele Engine in-

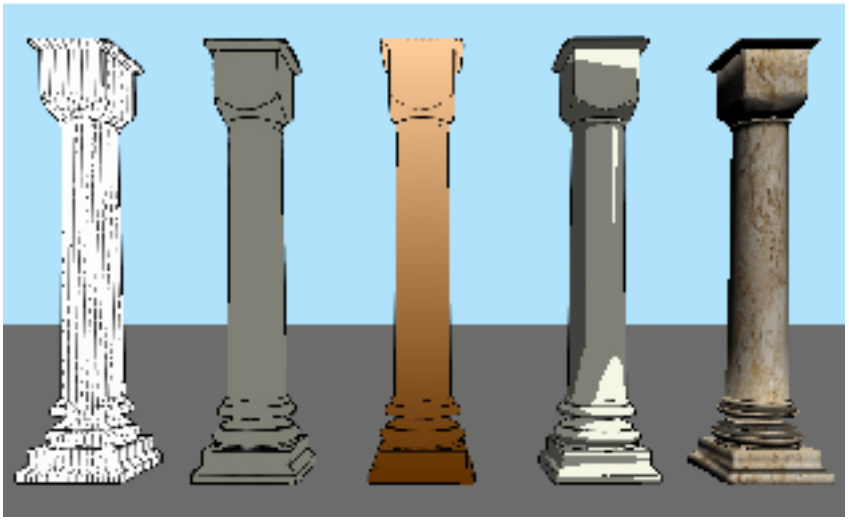

Abbildung 1: Säulen mit verschiedenen Renderingstilen

tegriert wurden. Die Beispielszenen sind aus dem Umfeld der Visualisierungen der Magdeburger Kaiserpfalz entnommen bzw. wurden vor diesem Hintergrund neu modelliert. Mit den Nicht-Photorealistichen Renderingtechniken hat man die Möglichkeit, wichtige Strukturen zu akzentuieren und unwichtige Bildinhalte zu unterdrücken. Dadurch ist es möglich das Auge des Betrachters in bestimmte Richtungen zu lenken [Mas99]. Auch lassen sich so zusätzliche Informationen in die Viualisierungen mit integrieren.
Kapitel 2 beschäftigt sich mit den verwendeten Verfahren und erläutert für die Implementierung wichtige Details. Kapitel 3 zeigt einen kurzen Überblick über die erzielten Ergebnisse und Kapitel 4 gibt eine Zusammenfassung und wagt einen Ausblick für zukünftige Verbesserungen.

Zeit:	Oktober 2000 - Februar 2001
Betreuer:	Bert Freudenberg (ISG) Dr.Maic Masuch (ISG)

2 Methoden

In diesem Abschnitt soll ein kurzer Überblick über die verwendeten Methoden welche zur Darstellung Nicht-Photorealistischer Computergraphiken zum Einsatz kamen gegeben werden. Desweiteren werden Hinweise zur Implementation sowie einige Ergebnisse präsentiert.
Wissenschaftliche Arbeiten zur Generierung Nicht-Photorealistischer Darstellungen mit Hilfe des Computer sind in den letzten Jahren viele veröffentlicht worden. Das Spektrum der verwendeten Techniken ist sehr groß, aber es gibt leider nur wenige Arbeiten zum Thema der Echtzeitgenerierung dieser Nicht-Photorealistischer Computergraphiken.
Zum Einsatz in diesem Praktikum kamen Techniken des Cartoonrenderings, bei denen die Silhouette und die inneren Kanten eines Objektes bestimmt werden mußten. Weiterhin kamen ein Z-Colouring Verfahren zur Anwendung sowie spezielle Inkmaps, welche es dem Betrachter ermöglichen trotz Mipmapping im-

mer einen gleichbleibenden Helligkeitseindruck der Objekte (Textur) zu bekommen.

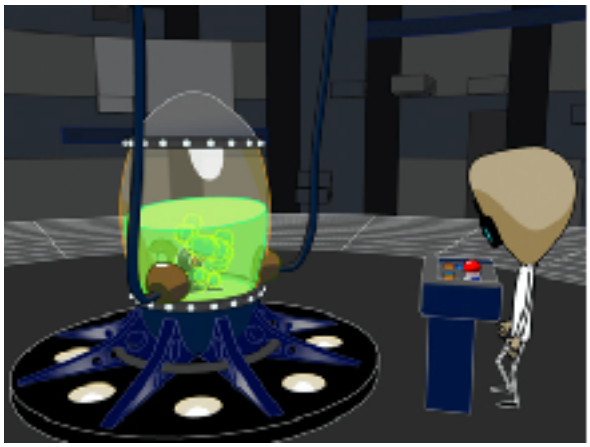

Abbildung 2: Cartoonrendering

Bei der praktischen Umsetzung entschied ich mich für die Fly3D Computer-Spiele Engine. Diese Engine ist frei verfügbar und in C++ implementiert. Zur Darstellung der 3D Graphikobjekte verwendet sie die OpenGL Schnittstelle. Szenen oder andere 3D Elemente lassen sich leicht mit 3D Studio MAX erstellen und in die Fly3D Engine importieren. Für Interaktion mit dem Joystick und für den Sound benutzt die Engine die MICROSOFT DIRECTX API [DirectX].
Für die Fly3D Engine wurden Plugins erstellt in denen die neuen Renderingtechniken implementiert wurden. Diese Plugins sind mit dem Visual C++ Studio von MICROSOFT erstellt worden.
Im Anschluß werden nun die verwendeten Techniken im Detail erläutert. Ergebnisse können in diesem, insbesondere aber auch im nächsten Kapitel betrachtet werden.

2.1 Cartoon Rendering

Das Cartoonrendering versucht eine Darstellung der 3D-Szene die an die Darstellung von gezeichneten Figuren in Comicheften oder Fernseh-Trickfilmen angelehnt ist, siehe Abbildung 2. Dabei werden die Objekte durch eine Silhouette von anderen Objekten abgegrenzt. Objekte, welche sich im Vordergrund befinden, haben eine geschlossene Silhouette. Werden die Objekte aber von anderen Objekten verdeckt, so ist die Silhouette an dieser Stelle manchmal unterbrochen. Silhouettenkanten sind oft sehr dick gezeichnet. Beispiele zeigen die Abbildungen 2 und 3.

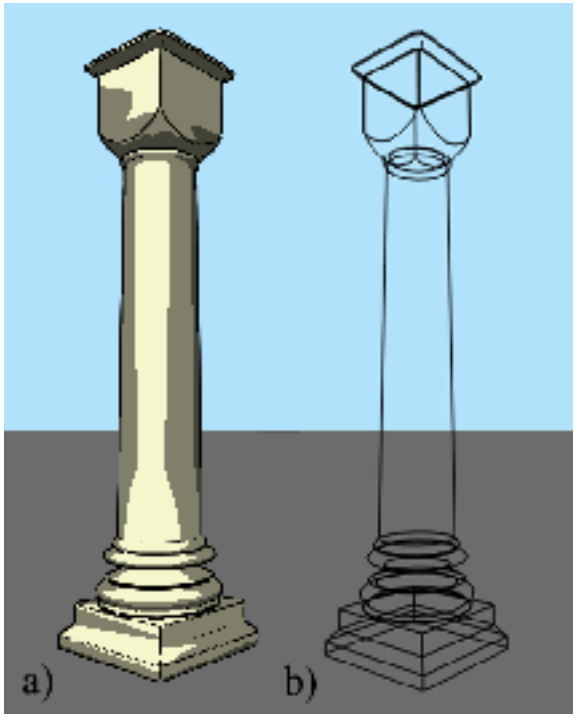

Abbildung 3: a) Cartoonrendering, b) Silhoutte und innere Kanten

Wie in Abbildung 2 zu sehen ist, sind bei einigen Objekten auch die inneren Kanten gezeichnet. Diese helfen, die Form des dargestellten Objektes besser zu erkennen. Üblicherweise sind diese Linien dünner gezeichnet als die Silhouettenkanten selbst, siehe dazu auch Abbildung 3.
Weiterhin zeichnen sich Cartoons durch eine

einfache Färbung aus. Auf Texturen wird meistens verzichtet. Die Objekte sind in einfachen Farben gehalten, deren Intensität aber variiert werden kann, um verschiedene Beleuchtungssituationen zu simulieren. In Abbildung 3a) befindet sich die Lichtquelle schräg über der Säule hinter der Kamera. Am Farbverlauf läßt sich sehr gut erkennen, wo sich die Lichtquelle befindet. Bei direkter Beleuchtung wird ein hellerer Farbton verwendet, bei Schatten ein entsprechend dunklerer.

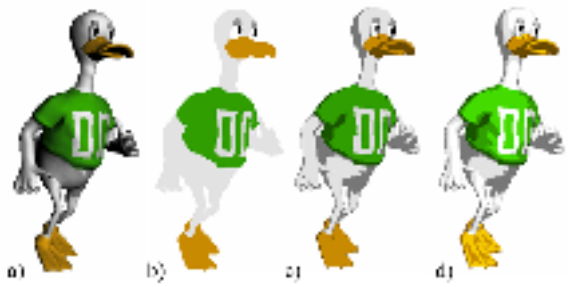

Abbildung 4: Beleuchtung beim Cartoonrendering, a) Gouraud schattiert, b) Flach schattiert, c) mit Schatten, d) mit Schatten und Spitzlicht

All diese Effekte lassen sich auch relativ leicht mit einem geeigneten Computerprogramm umsetzen. Die Silhouettenkanten werden bestimmt indem alle Kanten beim Zeichnen getestet werden, ob sie sich zwischen einer Fläche die nach vorne und einer die nach hinten zeigt befinden. Bei diesem speziellen Fall befindet sich die Kante an der Seite des Objektes. Sie gehört somit zur gesuchten Silhouette. Die inneren harten Kanten werden über einen Schwellwert bestimmt. In einem Vorverarbeitungsschritt wurden alle Kanten auf den von ihnen eingeschlossenen Winkel untersucht. Überschreitet dieser Winkel einen gegebenen Schwellwert gehören sie zu den harten Kanten und werden zusätzlich zur Silhouette gezeichnet. Die Stärke dieser Linien ist zumeist geringer als die der Silhouettenkanten, siehe Abbildung 3b). Die Beleuchtung der Objekte erfolgt analog der normalen Beleuchtung. Hier werden aber über einen Gradienten Schwellwerte vorgegeben die entscheiden mit welcher Intensitätsstufe ein Objekt bei einer bestimmten Lichtstärke koloriert wird. Abbildung 4 zeigt verschieden Beleuchtungsmodelle.

2.2 Line-Dithering

Dieses Rendering Verfahren stellt eine Erweiterung des Cartoonrendering dar. Anstatt nur einmal wird eine Kante mehrfach leicht versetzt voneinander gezeichnet. Die Start- und Endpunkte der Linien werden dabei jedesmal wenn ein neues Bild berechnet wird leicht verschoben. Dadurch ergibt sich ein unruhiges Bild und die Objekte sehen durch die vielen ungenau gezeichneten Linien mehr skizzenhaft aus. Aus der Sicht des Betrachters wirkt dieses Bild unvollkommen und nicht fertig. Er ist so eher bereit über den Inhalt der dargestellten Szene zu diskutieren und ihn nicht als festgeschrieben zu betrachten.
Für das Beispiel der Magdeburger Kaiserpfalz ist eine solche Darstellungsweise vorteilhafter als eine photorealistische Darstellung. Der Standort und das Aussehen der Pfalz ist bis Heute nicht eindeutig geklärt. Eine Darstellungsweise welche eher unvollständig wirkt und das Gebäude nur mit minimalen Details wiedergibt lädt eher zum Diskutieren und Überlegen ein, als eine photorealistische Darstellung, welche das Gebäude wie auf einem Photo abbildet. Abbildung 5 zeigt ein Beispiel des Line Dithering.
In einem Vorverarbeitungsschritt werden die zusätzlichen inneren Kanten über einen Schwellwert bestimmt. Art und Stärke lassen sich dabei über verschiedene Parameter einstellen. Koloriert werden die Objekte

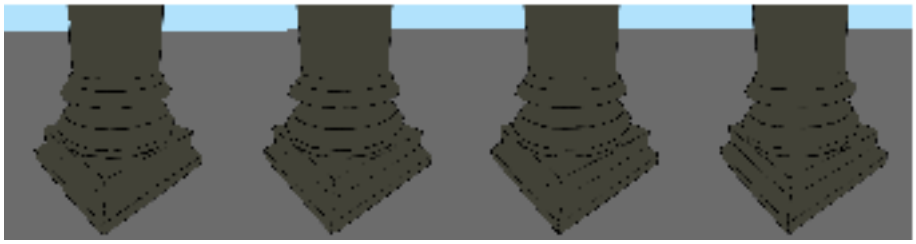

Abbildung 5: Säule mit Line-Dithering - 4 Frames

analog zum Cartoonrendering oder mit dem Z-Colouring Verfahren.

2.3 Z-Colouring

Auch das Z-Colouring ist eine Erweiterung des Standard Cartoonrenderings. Auch hier werden in einem Vorverarbeitungsschritt für alle Objekte die lokalen Minima und Maxima in den Z-Koordinaten der einzelnen Knoten bestimmt. Zusätzlich wird für jedes Objekt eine untere und obere Objektfarbe bestimmt. Die Farbe des Objektes ergibt sich aus einer linearen Interpolation zwischen diesen beiden Farbwerten. Dadurch entsteht ein kontinuierlicher Farbverlauf von unten nach oben. Siehe Abbildung 6 für ein Beispiel.

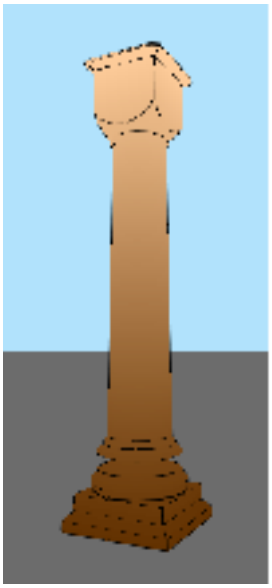

Abbildung 6: Z-Colouring

Diese Darstellungsweise ist sinnvoll um lineare Parameterveränderungen zu visualisiern wie z.B die Höhe eines Objektes. Auf das Beispiel der Kaiserpfalz angewandt zeigt sich so die nach oben immer größer werdende Unsicherheit im Aussehen des Gebäudes. Bei den Ausgrabungen hat man den Grundriß gefunden, welcher so als gegeben angenommen werden kann. Über das eigentliche Aussehen des Gebäudes, speziell der höheren Etagen, kann aber nur spekuliert werden. Wird das Z-Colouring in alle drei Richtungen angewandt, ist diese Verfahren auch dazu geeignet, Nebel zu simulieren.

2.4 Pen und Ink Styles

In der Computergraphik werden Texturen benutzt, um den dargestellten Objekten mehr Details zu verleihen und um ihre Form deutlicher hervortreten zu lassen. Bei sehr feinen Texturen, wie Hatchmaps, können aber unschöne Moirè Muster auftreten, Abbildung 7a).

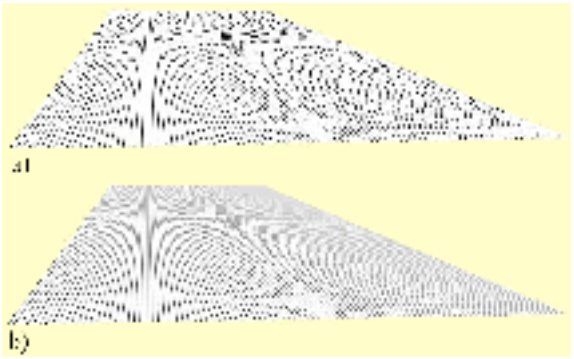

Abbildung 7: a) Ungefiltert, b) Mipmapping

Zur Lösung dieses Problems kann Mipmapping herangezogen werden, welches ein und dieselbe Textur in unterschiedlichen Auflösungen be-

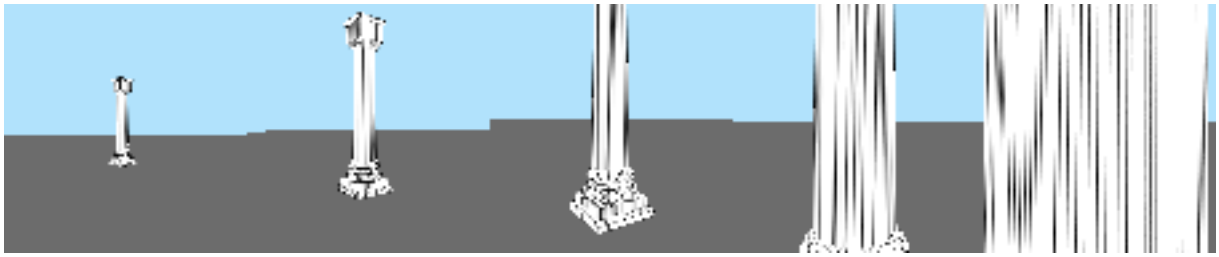

Abbildung 8: Anwendungsbeispiel für Inkmaps

reitstellt und je nach Entfernung zum Betrachter die Richtige auswählt [OpenGL]. Ein unschöner Nebeneffekt hierbei ist das die Texturen in weiter Entfernung nur noch eine graue Masse sind und sich keine Strukturen mehr erkennen lassen, Abbildung 7b).

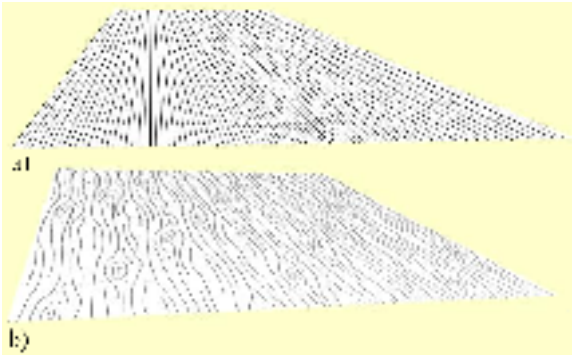

Abbildung 9: Triliniear Filtering a) Hatch Maps, b) Ink Maps

OpenGL erlaubt eine freie Zuweisung der Mipmap Level [OpenGL]. So wurden mit einem Vektorgraphikprogramm verschiedene Strukturen erzeugt, und automatisch Mipmap Level der entsprechenden Größe generiert. Bei aktiviertem Trilinearem Filtering zwischen den einzelnen Mipmap Stufen blenden die Strukturen weich ineinander über und garantieren ein Gleichbleiben der Intensitäten. Auch feine Struktur lassen sich nach wie vor gut erkennen. Abbildung 9 zeigt zwei Beispiele, a) mit einer einfachen Hatchmap und b) mit einer Inkmap mit Holzmaserung.
Abbildung 8 zeigt eine Säule, welche aus unterschiedlichen Entfernungen dargestellt wird. Auch hier ist sehr gut zu erkennen, das man immer einen gleichbleibenden Eindruck von der Struktur des Objektes hat, egal in welcher Entfernung sich der Standpunkt des Betrachters befindet.

3 Ergebnisse

Hier sollen einige Beispielbilder gezeigt werden, welche mit der Fly3D Engine und den in diesem Laborpraktikum entwickelten Plugins erstellt wurden. Die Szenen wurden mit 3D Studio MAX 3.0 modelliert und texturiert und sind zum Teil dem Kaiserpfalz Projekt entnommen.

Abbildung 10: Szene in 3D Studio MAX

Auf der beiliegenden CD (siehe Anhang) befinden sich alle Modelldaten und Skripte, um

diese Bilder zu reproduzieren.
Das erste Beispiel, Abbildung 10, zeigt eine Szene gerendert in 3D Studio MAX 3.0. Dieses Bild dient nur vergleichenden Zwecken. Die Renderzeit betrug 1.43 Minuten.

Abbildung 11: Szene in Fly3D (Photorealistisch)

Das zweite Bild, Abbildung 11, zeigt dieselbe Szene, aber diesmal photorealistisch dargestellt mit Hilfe der Fly3D Engine. Zur Darstellung wurden hierbei keine Nicht-Photorealistischen Renderingtechniken verwendet.

Abbildung 12: Cartoonrendering

Auch diese Bild wird nur zum Vergleich präsentiert. Dieses Bild ist gerendert worden mit einer Framerate von ca. 10-12 fps auf einem Athlon 600 mit *n*vidia TNT2 Graphikkarte.
Abbildung 12 zeigt die gleiche Szenerie, diesmal gerendert mit dem Cartoonrendering Verfahren. Deutlich zu erkennen sind die Silhouetten um jedes Objekt sowie die inneren Kanten der einzelnen Objekte. Für die Simulation der Beleuchtung wurden für jede Farbe drei verschiedene Helligkeiten benutzt. Der Stand der Sonne ist eindeutig und erschließt sich aus dem auf den Objekten abgebildeten Spitzlicht.

Abbildung 13: Line-Dithering

Abbildung 13 präsentiert einen Screenshot aus einer Animation, in welcher die Szene mit der Line-Dithering Methode gezeichnet wurde. Der Effekt ist am Besten in einer Animation zu zeigen. Da dies auf Papier nicht funktioniert, ist hier stellvertretend nur ein Frame der Animation dargestellt. Mit den auf der CD befindlichen Daten kann dies aber leicht reproduziert werden.
Man kann dann deutlich die zitternden Linien erkennen, welche dem Betrachter ein halbfertiges und handgezeichnetes Bild suggerieren.
In Abbildung 14 ist eine dörfliche Szene dargestellt. Sie zeigt den Unterschied zwischen Inkmaps, Häuser auf der linken Seite, und normalem Mipmapping, Häuser auf der rechten Seite.

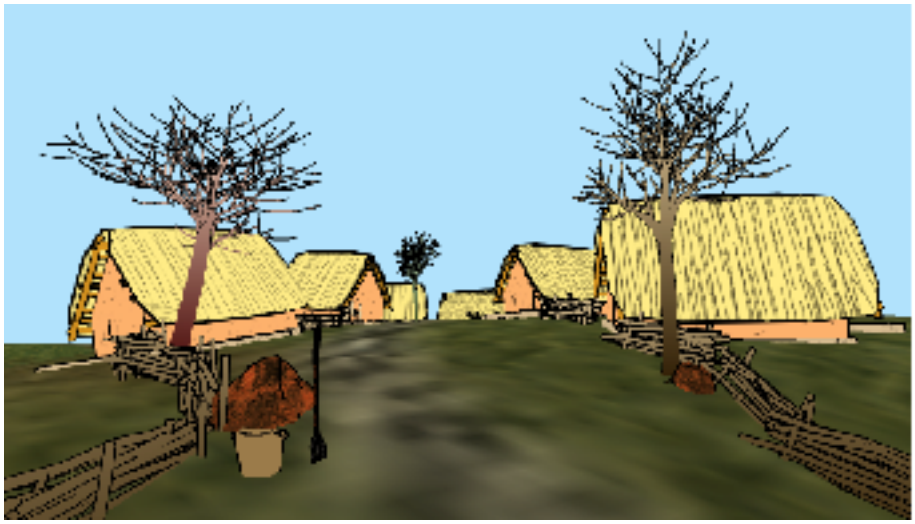

Abbildung 14: Linke Seite Inkmaps, rechte Seite Mipmapping

Während die Textur der Dächer der Häuser auf der rechten Seite durch das Mipmapping mit zunehmender Entfernung unscharf wird, ist sie auf der linken Seite auch beim entfernten Haus noch sehr gut zu erkennen. Zudem ist der Helligkeitseindruck der Texturen auf der linken Seite konstant.

Abbildung 15: Z-Colouring

Das nächste Beispiel, Abbildung 15, stellt die Szene mit dem Z-Colouring Verfahren dar. Hier werden wie bei den anderen Verfahren die Silhouettenkanten und die harten, inneren Kanten gezeichnet. Das Shading erfolgt anhand der Position der einzelnen Knoten. Es wird linear zwischen zwei Farben innerhalb der Objekte interpoliert. So lassen sich zusätzliche Informationen integrieren, wie hier z.B die Höhe eines Objektes.
Das letzte Beispiel zeigt einen Ausschnitt aus der Säulenhalle im Erdgeschoß der Magdeburger Kaiserpfalz. Hier kam eine einfache Hatchtextur zum Einsatz. Man kann sehr gut erkennen, wie durch das trilinieare Filtering die einzelnen Texturen weich ineinander überblenden und so auch hier einen gleichbleibenden Helligkeitseindruck der Textur vermitteln, ohne Details zu verwischen.

4 Zusammenfassung und Ausblick

Während meines Laborpraktikums habe ich mich mit einigen ausgewählten Methoden zur Erzeugung Nicht-Photorealistischer Computergraphiken vertraut gemacht und einige davon in eine Echtzeit Umgebung implementiert. Es wurden für die Fly3D Game Engine zwei Plugins entwickelt, die diese Renderingmetho-

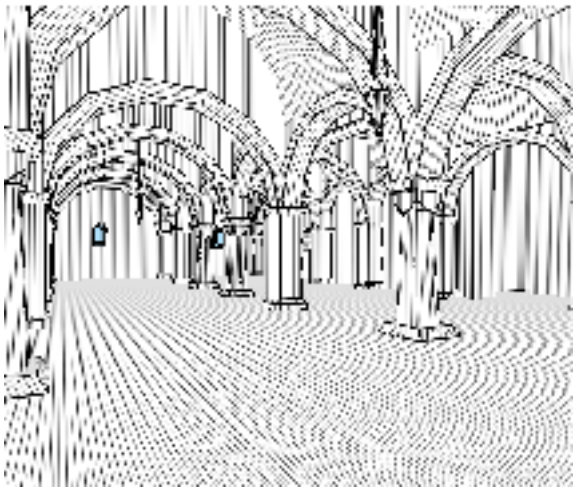

Abbildung 16: Hatchmaps - Gewölbe Kaiserpfalz

den beinhalten. Erzielte Ergebnisse wurden im vorigen Kapitel präsentiert. Alle hier vorgestellten und implementierten Renderingmethoden laufen selbst bei unoptimierten Geometrien interaktiv.
Teile dieser Arbeit wurden für ein Paper verwendet, welches zur Eurographics 2001 eingereicht und angenommen wurde [Freu01].
Schritte zur Weiterentwicklung und Verbesserung dieser Arbeit wären das Hinzufügen weiterer Techniken sowie Kombinationen aus den bisherigen Darstellungen. Als Weiterentwicklung würde z.B. das Überlagern mehrere verschiedener Renderingstile in Frage kommmen, beispielsweise das Blenden von Photorealismus und Nicht-Photorelismus sowie das Hinzufügen von Methoden zur Visualisierung zusätzlicher Informationen oder Parameter.
Zum Zeichnen der Silhouetten oder inneren Kanten könnten verschiedene Linienstile simuliert werden. So könnten für den Betrachter "schönere" Bilder entstehen, die Helfen die dargestellte Szene schneller zu verstehen.
Bei zukünftigen Entwicklungen ist ein weiteres Augenmerk auf die Frame-Kohärenz in den Animationen zu legen. Die Frame-Kohärenz ist dafür verantwortlich das es keine abrupten Änderungen zwischen den einzelnen Frames gibt, welche beim Betrachten störend wirken können.
Eine detailierte Geschwindigkeitsmessung und ein Vergleich mit verschiedenen Systemen würde die Arbeit abrunden.

5 CD-Inhalt

Die beiliegenden CD enthält alle Daten, um die hier abgebildeten Beispiele reproduzieren zu können. Auch der Quellcode der angefertigten Plugins liegt in Form eines Visual C++ Arbeitsbereiches auf der CD. Ist die Fly3D Engine ordnungsgemäß installiert, lassen sich alle Dateien über eine Batchdatei starten.

CD-Inhalt:	Fly3D	- Installation der Fly3D Engine
	Toonz	- Installation der Toonz-Komponenten
	Doku	- Dieses Dokument sowie ein paar Ergebnisse

Literatur

[Fly3D] www.fly3d.com.br

[OpenGL] www.opengl.org

[VisualC++] www.microsoft.com/VisualStudio

[DirectX] www.microsoft.com/DirectX

[Freu01] B. Freudenberg, M. Masuch, T. Strothotte, "Walk-Through Illustrations: Frame-Coherent Pen-and-Ink Style in a Game Engine", Eurographics 2001

[Mas99] M. Masuch, B. Freudenberg, B. Ludowici, S. Kreiker, T. Strothotte, "Virtual Reconstruction of Medival Architecture", Eurographics 1999

[Gooch] A. Gooch, B. Gooch, P. Shirley, E. Cohen, “A Non-Photorealistic Ligthing Model For Automatic Technical Illustration”

[Nor00] J. D. Northrup, L. Markosian, “Articstic Silhouettes: A Hybrid Approach”, NPAR 2000

[Klein] A. W. Klein, W. Li, M. M. Kazhdan, W. T. Corrêa, A. Finkelstein, T. A. Funkhouser, “Non-Photorealistic Virtual Environments”

[Mar] L. Markosian, M. A. Kowalski, S. J. Trychin, L. D. Bourdev, D. Goldstein, J. F. Hughes, “ Real-Time Nonphotorealistic Rendering”

[Ras99] R. Raskar, M. Cohen, “Image Precission Silhouette Edges”, Symposium on Interactive 3D Graphics, Atlanta 1999

[Lake] A. Lake, C. Marshall, M. Harris, M. Blackstein, “Stylized Rendering Techniques For Scalable Real-Time 3D Animation”