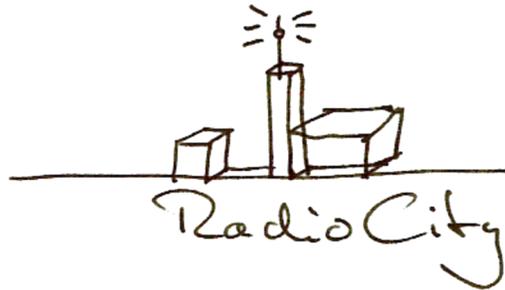


Universität Koblenz-Landau

Institut für Computervisualistik
Arbeitsgruppe Computergraphik



Studienarbeit

Konzeption, Entwicklung und Implementation eines
hierarchischen Radiosity - Systems mit Schwerpunkt auf
photometrischer Konsistenz und Tone Mapping

Vorgelegt von
Florian Koller

Betreuer: Dipl.-Inform. Thorsten Grosch
Prüfer: Prof. Dr.-Ing. Stefan Müller

November 2003

Inhaltsverzeichnis

1	Überblick/Einleitung	1
2	Planung und Realisation	1
2.1	Anforderungen an RADIOCITY	1
2.1.1	System im allgemeinen	2
2.1.2	Oberfläche	3
2.1.3	Tone Mapping Verfahren	4
2.1.4	Photometrische Konsistenz	4
2.2	Integration der parallel ablaufenden Studienarbeiten	5
3	Übersicht und grundlegender Aufbau von RadioCity	5
3.1	Eigenschaften von RADIOCITY	5
3.2	Verwendete Techniken/Systeme und deren Tücken	6
3.2.1	C++	6
3.2.2	OpenSG	6
3.2.3	Qt	7
3.2.4	Doxygen	7
3.3	Architekturaufbau	8
3.4	Laden, Speichern & Struktur erzeugter Daten	8
4	Tone Mapping	10
4.1	Voraussetzungen	11
4.2	Der Bauer: Lineare Skalierung	12
4.3	Der Springer: URQ	12
4.4	Der „gestürzte“ König: NURQ oder, „Fehler im Programm“	13
4.5	Die Dame: NURQV	16
5	Bedienung der Anwendung	16
5.1	Menüeinträge	17
5.2	Vorhandene Dialoge	18
5.3	Hinweis auf Inkonsistenzen	23
6	Photometrische Konsistenz & Farben	24
6.1	Interner Farbraum, Matrixdefinition & Funktionen	26
6.2	Gammakorrektur	27
7	Zusammenfassung & Kritik	28
7.1	Optimierungen für künftige Generationen	28
7.2	Mögliche Erweiterungen und Ausblick	29

8	Anhang/Zusätze	33
8.1	Bildschirmaufnahmen der Menüeinträge	33
8.2	Kleine Installationshilfe für RADIOCITY	35
8.3	Inhalt der beiliegenden CD	37
8.4	Ansichten berechneter Szenen & Bilder der Quantisierungsverfahren	38

1 Überblick/Einleitung

Ziel der Studienarbeit war es ein objektorientiertes Hierarchisches Radiositysystem zu konzipieren und zu entwickeln. Ein derartiges Vorhaben innerhalb des Rahmens einer einzigen Studienarbeit umzusetzen wäre nicht möglich gewesen. Durch die Tatsache, daß sich Dominik Rau und Guido Stegmann parallel um ein Thema bei der Arbeitsgruppe Computergraphik bemühten, ergab sich die Möglichkeit dieses Vorhaben gemeinsam anzugehen. Da eine sinnvolle Trennung der Aufgabenbereiche nötig war wurde dies entsprechend in den Themenstellungen der Studienarbeiten berücksichtigt. Hr. Rau beschäftigte sich in seiner Arbeit mit verschiedenen Meshing-Strategien und den dafür notwendigen Datenstrukturen. Hr. Stegmann lieferte alle notwendigen Komponenten für ein hierarchisches Radiosityverfahren mit Clustering. Diese beiden Teile sollten durch ein drittes Thema ergänzt bzw. zusammengefasst werden. Der Inhalt dieser Studienarbeit sollte zum einen die Realisierung von Anwendungsfunktionalität, ein Konzept zur photometrisch konsistenten Behandlung von Farben und mehrere Quantisierungsverfahren sein.

Am Anfang stand bei dieser Arbeit vor allem die Formulierung des Themas. Diese gestaltete sich schwieriger als erwartet, da obwohl klar zu sein schien was in eine rahmengebende Arbeit gehören sollte, das Festhalten von konkreten Aufgaben besonders schwer fiel. Die erste Hürde waren also „Themaverhandlungen“ die zur enthaltenen Aufgabenstellung führten.

Ein Problem welches sich dann sehr schnell einstellte war das Fehlen von Basisfunktionalität für ein derartig komplexes System. Die Anfangsphase war dadurch gekennzeichnet, daß wir schrittweise die notwendige Grundlage für das System geschaffen haben. So dauerte es auch eine Weile bis sich alle drei Testversionen in einem Modul wiederfanden und zentral zu einer gemeinsamen Anwendung weiterentwickelt wurden.

2 Planung und Realisation

2.1 Anforderungen an RadioCity

Bei der Formulierung der Anforderungen ergab sich folgendes Problem: Einige der aufgeführten Anforderungen konnten nicht im vorhinein formuliert werden bzw. tauchten bei den Überlegungen und ausformulierten Anforderungen überhaupt nicht auf. Dies ist auf die Tatsache zurückzuführen, daß bei Formulierung des Themas der Arbeit noch nicht klar war, welche konkreten Aufgaben tatsächlich Teil der Studienarbeit sein würden. Der Inhalt der Arbeit, der durch die Themenformulierung einen Rahmen erhielt, modifizierte sich dynamisch und auf an mich herangetragene Aufgaben reagierend. Das dabei teilweise andere Anforderungen im Umfang eingeschränkt und

auch teilweise die Erledigung von Aufgabenbereichen verlagert wurde, konnte nicht verhindert werden.

teilweise von meinen Kommilitonen Dominik [Rau] und Guido [Stegmann]¹ übernommen wurden konnte nicht verhindert werden. Dies ergab, soweit ich dies abschätzen kann, aber keine Probleme, da wir in regelmäßigem Kontakt standen und viele Aufgaben gemeinsam konstruktiv angingen und lösten. Im folgenden möchte ich nun einige der Anforderungen an das System und meine eigenen Schwerpunkte nennen und gegebenenfalls erläutern. Dies ist keine vollständige Liste, da im System viel an unsichtbarer Funktionalität vorhanden ist und nicht alles hier aufgeführt und somit dokumentiert werden kann. Es soll ein Einblick in die vielgestaltigen Anforderungen an diese Arbeit vermittelt werden.

2.1.1 System im allgemeinen

- Das Erscheinungsbild des Quelltextes soll möglichst einheitlich sein.
- Als Datentypen sollen soweit sinnvoll möglich OpenSG-Datentypen benutzt werden um die Plattformunabhängigkeit zu garantieren. Zusätzlich müssen sämtliche verwendeten Dateinamen komplett kleingeschrieben werden und als Folge dessen auch alle `#include`-Anweisungen im Quelltext. Dies wird vereinbart um die unterschiede im Dateisystem von Linux und Windows in den Griff zu bekommen.
- Für die Entwicklung wird die OpenSG-Version 1.2.0 (stable) verwendet. Dies geschieht um eine einheitliche Entwicklung innerhalb der Gruppe zu ermöglichen.
- Die Entwicklung soll objektorientiert erfolgen. Dazu soll ein Klassendiagramm angelegt, und schrittweise verfeinert werden.
- Das System soll plattformunabhängig entwickelt werden. Ziel soll es sein, daß die komplette Anwendung auf den Betriebssystemen Linux und Windows lauffähig ist.
- Die Oberfläche soll sich dynamisch an den aktuellen Zustand, in dem sich das Programm befindet, anpassen. Hierzu gehört das Ausblenden und Deaktivieren von, bei schon berechneter Szene, nicht sinnvollen und nicht möglichen Oberflächenfunktionen.
- Die Geometrie des Lambert Emitters soll in die Szene eingeblendet werden können und zusätzlich als schattenwerfende Geometrie berücksichtigt werden. Die Positionsgeometrien der übrigen Lichtquellen (Punktlicht und Spotlichtquelle) sollen über eine Kugel bzw. einen Kegel realisiert werden und ebenfalls einblendbar sein.

¹die Angaben zu den Studienarbeiten befinden sich im Literaturverzeichnis

- Umsetzung einer adäquaten visuellen Rückmeldung über den Fortschritt der Radiositysimulation an den Benutzer um über eventuelle Dauer der Berechnung bzw. das Fortlaufen des Programmes zu informieren
- Auslagerung der eigentlichen Radiosityberechnung in einen eigenen Prozess um ihn bei Bedarf auch abbrechen zu können ohne die Anwendung neu zu starten.
- Erstellung eines eigenen Dateiformates um Szenen und deren Einstellungen abzuspeichern und erneut einladen zu können. Dies wird benötigt um zu einem späteren Zeitpunkt eine erneute Simulation durchführen zu können. Die Clusterstruktur soll beim abspeichern erhalten bleiben.
- Das Einladen und Berechnen von Szenen soll durch die Unterstützung von unterschiedlichen Szenenformaten (z.B. VRML V2.0) möglich sein.
- Das Abspeichern einer berechneten Szene und ein erneutes Laden dieser für Präsentationszwecke soll ermöglicht werden
- Das Abspeichern einer Szenenansicht als digitales Bild soll Bestandteil des Systems werden

2.1.2 Oberfläche

- Es sollen über die Oberfläche Möglichkeiten zum Hinzufügen/Bearbeiten und Entfernen von Lichtquellen angeboten werden.
- Sämtliche für eine Berechnung notwendigen Parameter und Einstellungen sollen über die Benutzerschnittstelle(UI²) eingegeben werden können. Hierzu zählen unter anderem:
 - Einstellungen aller Daten der Radiositysimulation
 - Definition von Monitormatrizen über verschiedene Datensätze (z.B. XYZ, xyL-Monitorwerte)
- Es sollen Bedienungsmöglichkeiten für die bereitgestellte Funktionalität der Anwendung über Dialoge, Menüeinträge und Tastaturkürzel geschaffen werden. Dazu zählen:
 - Eingabe sämtlicher, für die Quantisierungsverfahren notwendigen Parameter in geeigneter Form
 - Abspeichern/Laden der Szeneneinstellungen
 - Ein/Ausschalten von Lichtquellengeometrien
 - Ein/Ausschalten von Gouraud-Shading

²UI = UserInterface, gemeint ist eine Graphische Benutzeroberfläche

- Einblenden von weiteren Informationsobjekten (z.B. Cluster)
- Ausgabe von allgemeinen Szeneninformationen
- Erstellung eines Fensters für statistische Informationen über geladene Szene und Berechnungszeiten

2.1.3 Tone Mapping Verfahren

- Es sollen verschiedene Algorithmen (linear, URQ, NURQ), die visuell verwertbare Skalierungsergebnisse erzeugen, umgesetzt werden.
- Abschließend soll ein qualitativer Vergleich der Verfahren anhand von erzeugten Bildern erfolgen.
- Bei der Implementierung ist auf Performanz und Speicherbedarf zu achten.
- Sollten die implementierten Verfahren, bei Anwendung, signifikante Zeitverzögerungen hervorrufen so soll dies dokumentiert und ausgewertet werden.
- Eine Erstellung eines Dialogs zur automatischen Generierung von Tone Mapping Parametern ist falls möglich zu integrieren.
- Ebenfalls sollen die typischen Fälle der Skalierung einer über- und untersteuerten Szene visuell dokumentiert werden.

2.1.4 Photometrische Konsistenz

- Die verschiedenen Klassen für die photometrische Konsistenz sollen für das System RADIOCITY und auch unabhängig von ihm verwendbar sein. Diese Forderung ist mittels Überladungen von Funktionen zu realisieren.
- Es sollen die Lichtquellen Lambert Emitter, Pointlight und Spotlight in das photometrische Konzept eingebunden werden.
- Eine komplexe Lichtquelle LVK, die komplexe Lichtspezifikationen durch verschiedene Lichtstärkeverteilungskurven realisiert, soll integriert werden.
- Hinzufügen weiterer Lichtquellen (z.B. Sonnenlicht) soll durch die Struktur des Entwurfes ermöglicht werden.
- Ein rudimentäres Farbkonzept soll erstellt werden. Dieses soll die sinnvolle Konvertierung zwischen Farbräumen ermöglichen und stellt zusätzliche Farboperationen zur Verfügung.

- Desweiteren werden alle Funktionen für eine erfolgreiche photometrische Interpretation von Eingabedaten benötigt. Dazu zählen zum Beispiel Funktionen zur Definition von Farbraummatrizen über unterschiedliche Parameter.

Das sich nicht alle Anforderungen zeitgerecht umsetzen lassen war von vorneherein klar und dementsprechend wurden die einzelnen Aufgabenbereiche gegliedert und einer Priorisierung unterzogen. Einige der obigen Anforderungen konnten nicht im Rahmen der Studienarbeit bearbeitet werden, diese weise ich zum Teil explizit im Kapitel 7.1 und zusätzlich im Kapitel 7.2 aus.

2.2 Integration der parallel ablaufenden Studienarbeiten

Mit der Studienarbeit von Dominik [Rau] ergaben sich sehr wenige Abhängigkeiten und die Integration verlief ohne besondere Anstrengungen. Sich überschneidende Bereiche existierten z.B. beim Einlesen der Geometrie. Die Lichtquellenintegration in den hierarchischen Algorithmus erfolgte in Zusammenarbeit und unter Absprache mit Hr. [Stegmann]. Dabei vergingen einige Wochenendstunden und auch die Telefonrechnung stieg in diesem Zeitraum deutlich. Darüber hinaus sind wir alle drei in zahlreiche Treffen mit unserem Betreuer Thorsten Grosch gemeinsam die auftretenden Problemstellungen angegangen. Teilweise kam es vor, daß Abhängigkeiten innerhalb des Projektes zu Wartezeiten führten. So mußte zum Beispiel teilweise auf das Vollenden einer bestimmten Klasse gewartet werden, dies lies sich jedoch immer irgendwie durch das Vorziehen anderer Aufgaben überbrücken. Ein Problem gegen Ende der Studienarbeit war, daß durch die Zusammenarbeit von drei Leuten das System recht groß wurde und es teilweise schwierig wurde zuzuordnen in welchem Teil des Programms korrigiert werden mußte um Fehler zu beheben.

3 Übersicht und grundlegender Aufbau von RadioCity

3.1 Eigenschaften von RadioCity

Zusätzlich zu den, dem Radiosityverfahren zugrundeliegenden Eigenschaften (z.B. Unabhängigkeit des Betrachterstandpunktes) und Einschränkungen (Komplexität der Parameterjustierung) zeichnet sich das System durch folgende Punkte aus:

- Eine objektoriente Entwicklung und Struktur (mit leichten Einschränkungen aufgrund von Geschwindigkeitsüberlegungen)

- Hierarchisches Radiosityverfahren mit Clustering
- Ausgefeilte Meshing Algorithmen (Spezialunterteilung, gebogene Oberflächen) und dafür erforderliche Datenstrukturen
- Tonemapping Verfahren
- Photometrische Konsistenz
- Visualisierung von wichtigen Konzepten (Cluster, verschiedene Linktypen) über einschaltbare Positions- und Informationsgeometrien
- Visualisierung vieler zusätzlicher Informationen
- Plattformunabhängigkeit: Windows (W2K und WXP) und theoretisch jedes Linux/Unix-Derivat³ mit OpenGL und Qt-Unterstützung

3.2 Verwendete Techniken/Systeme und deren Tücken

3.2.1 C++

RADIOCITY wurde komplett in C++ entwickelt. Für mich war es der erste Kontakt mit dieser Programmiersprache. Dementsprechend waren gerade am Anfang einige „sprachliche Hürden“ zu nehmen. Mit Hilfe der im Literaturverzeichnis genannten Bücher war aber die Einarbeitung kein großes Problem. Mit C++ lassen sich einige schöne Dinge umsetzen. Hierzu zählen z.B. die sogenannten „Singletons“. Dies sind Klassenimplementationen von denen zur Laufzeit des Programmes immer nur ein Objekt existiert. Nützlich ist diese Eigenschaft z.B. für Loader-Klassen. Dieses Konzept wurde für die Klassen ColorCalibration und Tonemapper umgesetzt. Eine weitere schöne Möglichkeit ist das Überladen von Operatoren. Dieses kann genutzt werden um eigene Datentypen (z.B. die Farbklassen) so intuitiv wie möglich umzusetzen. Dieses Überladen von Operatoren wurde auch für das Ein- und Auslesen von Daten verwendet.

3.2.2 OpenGL

OpenGL ist ein portables Szenengraphsystem welches nach OpenSource Richtlinien (LGPL⁴) entwickelt wird.

Innerhalb von RADIOCITY wird es vor allem für das Einladen von Szenenformaten und die Darstellung der berechneten Szenen verwendet. OpenGL ist ein recht umfangreiches Projekt und das Hauptproblem in diesem Zusammenhang ist die (fast) fehlende Klassendokumentation. Dies führte zu Problemen, da oft

³entwickelt wurde auf Debian-Sid

⁴Lesser General Public License, siehe <http://www.opensource.org/licenses/lgpl-license.html>

nicht klar war was die vorhandenen Funktionen genau machen. Viel Zeit wurde darauf verwendet unter Windows Qt in Verbindung mit OpenGL zu installieren und benutzbar zu machen. Hier mußten eigene Anpassungen vorgenommen werden und eigends hierfür erstellte (OpenGL-Entwicklungsteam) Klassen eingebunden werden (siehe Quelltext OpenGLWidget). Auch die Installation und Verwendung innerhalb der Entwicklungssoftware Microsoft Visual Studio .NET war relativ zeitaufwändig. Hierfür mußten die Qt-Headerdateien eigenhändig angepasst werden (siehe Abschnitt 8.2 im Anhang).

3.2.3 Qt

Qt ist ein Fenstermanager welcher sich durch Plattformunabhängigkeit und weite Verbreitung auszeichnet. Es besteht die Möglichkeit mittels des sogenannten „Designers“ relativ schnell kleine Dialoge zu erstellen und diese einzubinden. Die Nachteile liegen hierbei allerdings in der Vielzahl an Dateien die für ein einziges Fenster erzeugt werden. Die Funktionalität die das Fenster benutzbar macht muß allerdings komplett selbst implementiert werden. Praktisch ist der Designer auf jeden Fall, da er einem lästige Schreibarbeit abnehmen kann und gerade das Verändern eines Dialoges im Nachhinein sehr leicht fällt. Ich bin einen Mittelweg aus eigener Implementation der Fenster und Benutzung des Designers gegangen und habe so modulare Dialoge entworfen die unseren Anforderungen genügen.

3.2.4 Doxygen

Die beiliegende Quelltextdokumentation wurde komplett mit Doxygen⁵ generiert. Dieses ermöglicht das Extrahieren von Quelltextkommentaren die in einem bestimmten vorgegebenen Format gehalten sind.

⁵<http://www.doxygen.org>

3.3 Architekturaufbau

RADIOCITY ist in sich in mehrere Bereiche gegliedert. Diese ergaben sich zum Teil auch schon aus der Aufgabenverteilung der drei beteiligten Studienarbeiten. So wurden von meiner Seite die Komponenten für das Hauptprogramm, die Dialoge, die Tone Mapper, und die Komponenten für photometrische Konsistenz eingebracht. Von Seiten von Guido Stegmann wurde die Radiositykomponente mitsamt einiger notwendiger Hilfsklassen erstellt. Am Aufbau der Szenenhierarchie durch die Klassenstruktur waren alle drei Studienarbeiten mit unterschiedlichen Schwerpunkten beteiligt. Die Meshing-Komponente wurde mitsamt aller dafür benötigten Daten und Klassen von Dominik Rau implementiert. In der Abbildung 1 wird ein kurzer Überblick über den prinzipiellen Aufbau von RADIOCITY gegeben.

Für den Aufbau der Szenenhierarchie wird eine abstrakte Elementklasse „Ele-

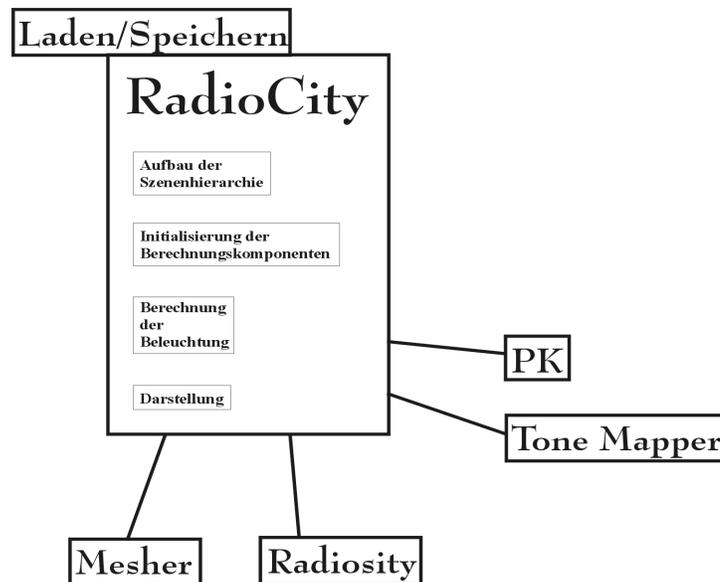


Abbildung 1: Aufbau von RADIOCITY

ment“ benutzt. Von ihr sind sämtliche anderen, für die Radiosityberechnung relevanten, Elemente abgeleitet.

3.4 Laden, Speichern & Struktur erzeugter Daten

Für das Abspeichern einer berechneten Szene, über den Menüeintrag „Save rendered scene“ im Menü „File/Scene“, stehen momentan nur die Formate „.osg“ und „.bin“ zur Verfügung. Bei „.osg“ ist je nach Szene mit langen Wartezeiten zu rechnen, da diese Schreibfunktion recht langsam ist. Das Schreiben von Binärdaten (.bin) geht allerdings recht schnell. Diese so abgespeicherten Dateien können für eine spätere Betrachtung erneut eingeladen werden. Es kann mit

ihnen allerdings keine erneute Berechnung durchgeführt werden. Die Auswahl der Dateiformate wird durch einen Fehler in OpenSG hervorgerufen, da dessen globaler Schreiboperator mit dem System nicht zusammenarbeitet.

Für das Speichern von Szeneinstellungen wird ein eigenes Dateiformat verwendet. Dieses sieht auszugsweise so aus:

```
LAMBERTEMITTER
343, 548.79, 227 %Punkte des Lambert Emitters
343, 548.79, 332
213, 548.79, 332
213, 548.79, 227
25000 %Leuchtdichte des Lambert Emitters
1 1 1 %Farbe
278, 548.79, 279.5 %Mittelpunkt
POINTLIGHT
62.6127, 352.285, 559.2 %Position der Punktlichtquelle
1 1 1 %Farbe
20000 %Lichtstrom
SPOTLIGHT
554.865, 354.222, 239.93 %Position der Spotlichtquelle
-554.865, 29.6317, 89.6812 %Richtung
20000 %Lichtstrom
1 1 1 %Farbe
20 %Exponent n
30 %thetaMax oder Cutoff
PARAMETERS
100 %Amount of visibility rays
0.09 %alpha, beta-Link boundary
15 %deltaTI
0.08 %BFF
110 %BFA
5 %Refine selflinks of cluster boundary
20 %minimal area
0.05 %tmaxPointlight to cluster refine
MONITORMATRIX
    0.589    0.179    0.183    0.000 %Zeilen und Spalten der verwendeten Monitormatrix
    0.290    0.605    0.105    0.000
   -0.000    0.068    1.020    0.000
    0.000    0.000    0.000    1.000
```

Es existiert jeweils eine bezeichnende Zeile welche beim Einlesen der Konfigurationsdatei festlegt, welche Daten als nächstes zu erwarten sind. Die oben mit % eingeleiteten Zeilenenden sind in der Originaldatei nicht vorhanden. Bei manueller Änderung dieser Dateien von Hand sollte vorsichtig zu Werke gegangen werden. Die Funktionalität wurde durch das Überladen der Streamoperatoren für Ein- und Ausgabe (siehe z.B. [DCP]) zur Verfügung gestellt. Um unterschiedliche Konfigurationen einer Szene anzulegen muß diese kopiert und umbenannt werden. Diese unterschiedlichen Dateien können dann geladen werden und jeweils um Lichtquellen und Einstellungen erweitert werden. Nach der Bearbeitung speichert man die szenenspezifische Konfiguration mit der

Funktion „Save scene parameters“ ab. Warum dies momentan so umständlich geschehen muß lege ich im Abschnitt 7.1 dar.

4 Tone Mapping

Generell sollen die in berechneten Bildern vorkommenden hohen Intensitätsunterschiede der darzustellenden Farben auf ein bestimmtes Ausgabemedium (meistens der Monitor) angepasst werden. Da die meisten Ausgabegeräte nur N ganze Werte pro Farbkanal zur Farbeingabe unterstützen, müssen die in der berechneten Szene vorkommenden reellen Werte jeweils dem entsprechenden Wert des Ausgabegerätes zugeordnet werden. Allerdings übersteigt der Wertebereich innerhalb computergenerierter Bilder⁶ fast immer den des Ausgabegerätes. Deshalb muß eine Anpassung vorgenommen werden um die Bilder auf dem Ausgabegerät so originalgetreu wie möglich ausgeben zu können. Dies geschieht mit Hilfe von Quantisierungsverfahren, den hier vorgestellten Tone Mapping Verfahren. Jede Farbe in der berechneten Szene besteht aus drei Komponenten, den jeweiligen Intensitätswerten des zugeordneten Farbkanals. Jeder dieser Werte muß einem der N ganzen Werte $[0, N - 1]$ zugeordnet werden. Dies geschieht auf folgende Weise:

$$Q(v) = \lfloor Nf(v) \rfloor$$

$$\text{mit } f : [0, \text{max}] \rightarrow [0, 1]$$

Hierbei ist f diejenige Funktion die auf eine bestimmte Weise den Farbton in den Zielbereich von 0 bis 1 umrechnet und deswegen im Englischen „tone reproduction function“ genannt wird. Hinweis: Die Notation der hier aufgeführten Formeln habe ich größtenteils übernommen aus: [QTV] Zusätzlich stammt ein Großteil der hier aufgeführten Informationen. Weitere Informationen lieferte das Buch „Radiosity & Global Illumination“ [RGI].

Auf den nachfolgenden Seiten möchte ich die Theorie der implementierten Quantisierungsverfahren vorstellen. Die Verfahren bauen alle aufeinander auf. Dieses läßt auch den direkten Vergleich recht anschaulich werden. Implementiert wurden:

- Lineares Tone Mapping
- URQ Tone Mapping (Uniform Rational Quantization)
- NURQ Tone Mapping (Non Uniform Rational Quantization)

⁶In Fotografien ist dies fast immer der Fall, und damit auch bei visuellen Eindrücken im normalen Leben

- NURQV Tone Mapping (Non Uniform Rational Quantization performed in HSV color space)

Dabei wurde die lineare Skalierung der Farbwerte, obwohl mit den anderen Verfahren durch Parametereinstellungen ebenfalls realisierbar, der Form halber eigenständig implementiert. Die vorgestellten Verfahren wurden ausgesucht, da sie gute Resultate liefern und dennoch ausreichend schnell sind. Sämtliche Verfahren wurden in der Klasse Tonemapper angesiedelt. Sie sind jeweils so implementiert, daß sie für RadioCity direkt verwendet werden können. Darüber hinaus war eine Maßgabe sie auch unabhängig vom hier umgesetzten System benutzen zu können. Dies sollte durch die verwendete Kapselung der Funktionalität gewährleistet sein. Die eigentlichen in RADIOCITY verwendbaren Algorithmen sind wie folgt benannt:

- mapAllLinear(...)
- mapAllURQ(...)
- mapAllNURQ(...)
- mapAllNURQV(...)

Das „All“ deutet darauf hin, daß sämtliche für die Darstellung relevante Elemente berücksichtigt werden. Dies sind die Flächen (Patches) der Szene. Die implementierten Algorithmen können auch selbständig, und damit in anderen Systemen verwendet werden. Dies geschieht über die Verwendung der entsprechenden Funktionen für einzelne Farben. Zum Teil müssen dafür allerdings die unter Voraussetzungen genannten Werte selbst initialisiert werden (siehe Kapitel 4.1 Voraussetzungen). Beim Tone Mapping wird immer der Radiositywert eines Patches in die Darstellungsfarbe kopiert. Diese Trennung von Darstellung und Berechnung ermöglicht es verschiedene Quantisierungsverfahren hintereinander anzuwenden und bei Bedarf auf die ursprünglichen Radiositywerte für die Darstellung zurückzugreifen.

4.1 Voraussetzungen

Bevor eine Skalierung der Farbwerte der Szene durchgeführt werden kann müssen einige wichtige Informationen ausgelesen werden. Dazu gehören:

- Bestimmung des minimalen und maximalen Radiositywert der Patches der Szene, hieraus werden dann die minimalen und maximalen Intensitäten der Szene bestimmt.
- Berechnung des linearen Skalierungsfaktors, dieser wird durch Angabe der oberen Grenze des Zielwertebereichs bestimmt

- Berechnung eines mittleren Wertes aus maximaler und minimaler Intensität, dieser wird für die NURQ und NURQV-Verfahren benötigt

Für die Bestimmung dieser Werte werden Flächen und Lichtquellen traversiert. Als Folge des Berücksichtigen von Lichtquellen ergibt sich der Effekt, daß nach linearer Skalierung mit einem kleinen Parameter in der Regel nur noch die Lichtquellen zu sehen sind. Die Initialisierung der genannten Werte findet durch den Aufruf der Funktion `initialPass()` innerhalb der Klasse `Tonemapper` statt.

4.2 Der Bauer: Lineare Skalierung

Die einfachste Art die Farben der Szene zu skalieren ist die lineare Skalierung. Hierbei wird vor Skalierung ein Maximalwert angegeben auf den dann der in der Szene vorkommende maximale Intensitätswert abgebildet wird. Sämtliche Farbwerte zwischen dem minimalen und maximalen Wert werden linear skaliert.

$$f(val) = val * scalefactor$$

Wobei *val* der alte Intensitätswert des Farbkanals ist und *scalefactor* sich aus dem maximalen Intensitätswert der Szene und dem neuen gewünschten Maximum nach der Skalierung zusammensetzt. Also:

$$scalefactor = \frac{newMaxValue}{maxIntensity}$$

Normalerweise wird für diese Art der Skalierung der neue maximale Wert der Szene auf 1 gesetzt, da dann kein Abschneiden größerer Werte bei der Darstellung auf dem Bildschirm auftritt. Der große Nachteil bei dieser Form der Skalierung läßt sich leicht aus den Ergebnisbildern erkennen. Oft sind nur die hellsten Bereiche des Bildes noch zu erkennen und damit das Verfahren oft nicht ausreichend für die erzeugten Szenen. Dies kann kompensiert werden, indem *newMaxValue* erhöht wird (z.B. auf 20). Dadurch werden auch die bisher zu dunklen Bereiche des Bildes soweit aufgehellt bis Konturen und Einzelheiten erkennbar werden. Allerdings kommen nach einer derartigen Operation Farbtintensitäten größer 1 in der Szene vor. Dies führt oft zu Bereichen in denen Farbverläufe nicht dargestellt werden können, da sämtliche Werte des entsprechenden Bereichs über 1 liegen und vom Ausgabemedium bei einem Wert von 1 abgeschnitten werden. Ein ähnlicher Effekt ist bei der Visualisierung der Radiositywerte zu beobachten.

4.3 Der Springer: URQ

Uniform Rational Quantization, kurz URQ, führt zu wesentlich besseren Ergebnissen als die lineare Skalierung. Allerdings sind die Ergebnisse fast identisch

mit denen einer logarithmischen bzw. exponentielle Skalierung. Diese hier nicht beschriebenen Verfahren sind allerdings zeitaufwändiger von der Berechnung. Die Skalierungsfunktion ist:

$$f(val) = \frac{p * val}{p * val - val + maxIntensity} \quad \text{mit } p \in [1, \infty] \quad (1)$$

Hierbei ist anzumerken, daß mit einem Wert von 1 für den Parameter p , die URQ der linearen Skalierung entspricht. Es ergibt sich somit:

$$f(val) = \frac{val}{maxIntensity}$$

Für das Problem der Generierung eines Wertes für p gibt es unterschiedliche Lösungsansätze (siehe z.B.: [TR]. Ursprünglich wollte ich das in [QTV] vorgeschlagene Verfahren mit einer verbesserten Dialogführung ins System integrieren. Dieses Vorhaben musste ich leider aus Zeitgründen aufgeben. Sämtliche Verfahren skalieren immer nur einen Wert auf einmal. Daß heißt um sie auf Farbbilder anwenden zu können, müssen immer alle Farbkanäle bearbeitet werden. Instinktiv würde man für jeden der Farbkanäle einen Parameter berechnen und verwendet dann für jeden der drei Farbkanäle eine eigene Funktion. Dieser einfache Ansatz führt allerdings zu fehlerhaften Bildern, da sich die Wertebereiche der Farbkanäle stark unterscheiden können und damit auch die daraus resultierenden Funktionen unterschiedliche sind. Damit ergeben sich dann Farbverschiebungen im Ergebnisbild die deutlich zu sehen sind. Als akzeptable Alternative wird in [QTV] vorgeschlagen die URQ auf Farbbilder auszuweiten (siehe Abschnitt 4.4). Damit ergeben sich allerdings andere Probleme, die bei der Implementation auftauchten. Um das URQ-Verfahren auch für Farbbilder verwenden zu können, wird hier einfach für alle drei Farbkanäle derselbe Parameter p benutzt. Damit ergeben sich recht gute Ergebnisse. Der einzige Nachteil ist das manchmal bemerkbare Abnehmen von Farbinformationen. Das setzen des Parameters p erfolgt über die graphische Benutzeroberfläche, die im Kapitel 5 beschrieben wird.

4.4 Der „gestürzte“ König: NURQ oder, „Fehler im Programm“

Das im vorangegangenen Kapitel erklärte Verfahren kann als einheitlich oder gleichförmig in der Hinsicht bezeichnet werden, daß es die Helligkeitsunterschiede zwischen den Pixeln erhält. Tatsächlich stimmt dieses Verhalten nicht immer mit der visuellen Wahrnehmung des Menschen überein. Im Grunde ist die subjektiv empfundene Helligkeit eines Objektes immer stark abhängig von der Helligkeit der Umgebung in der es sich befindet. Dieser Effekt sollte innerhalb eines Quantisierungsverfahrens berücksichtigt werden. Deshalb wird das vorangegangene Verfahren durch einen Zusatz um genau diesen Effekt der

subjektiven Wahrnehmung erweitert. Für jedes Pixel im Bild heißt dies: Berechne die mittlere Helligkeit der Nachbarschaftspixel des Pixels und modifiziere den eigentlichen Helligkeitswert des Pixels mit dem erhaltenen Wert auf geeignete Weise (siehe Formel 2). Im allgemeinen ist es so, daß ein Pixel heller wirkt wenn die Umgebungspixel im Verhältnis zu ihm dunkler sind und umgekehrt. Allerdings wurde in [QTV] festgestellt, daß sich bei Reduktion der Größe der Pixelnachbarschaft schrittweise die Qualität der Ergebnisbilder verbessert. Schlußendlich: Die besten Bilder erzeugt eine Pixelnachbarschaft, die nur noch aus dem Pixel selbst besteht, also im Grunde keine Nachbarschaft mehr ist. Überraschenderweise liefert dieses Verfahren wesentlich bessere Bilder als z.B. das URQ-Verfahren. Dieser Effekt macht es allerdings überhaupt erst möglich, das Verfahren im Rahmen des Systems zu verwenden, da hier nicht auf 2D-Bildern, sondern in 3D-Szenen gearbeitet wird. Innerhalb einer Szene lassen sich keine Pixelnachbarschaften bestimmen. Das komplette Verfahren müßte dann immer auf dem gerenderten Bild durchgeführt werden. Das wiederum hätte zur Folge, daß die Szenenbetrachtung erheblich durch Wartezeiten beeinträchtigt würde⁷.

Für das NURQV-Verfahren wird der Wertebereich in zwei Teile geteilt. Die Aufteilung wird durch die Bildung eines Mittelwertes *meanVal* erreicht.

$$meanVal = \sqrt{minIntensity * maxIntensity}$$

Mit Hilfe des Mittelwertes kann dann die Unterteilung in zwei gleich große Bereiche erfolgen.

$$\frac{maxIntensity}{meanValue} = \frac{meanValue}{minIntensity} = \sqrt{\frac{maxIntensity}{minIntensity}}$$

Zusätzlich wird noch das Verhältnis aus dem Intensitätswert der Pixelnachbarschaft (*zoneValue*)⁸ und dem Mittelwert (*meanValue*) benötigt. Dieses Verhältnis gibt die Helligkeit der „benachbarten Zone“ des Pixel an. Ist der Wert kleiner als 1, ist sie dunkel und bei Werten größer als 1 wird angenommen sie sei hell. Diese Verhältnis wird nun benutzt um den Parameter *p* aus der vorangegangenen Skalierungsmethode wie folgt zu modifizieren:

$$p' = p * (1 - k + k * \frac{pixelIntensity}{meanValue}) \quad \text{mit } k \in [0, 1] \quad (2)$$

Die von mir implementierte Technik entspricht dem in [QTV] unter „4. Non Uniform Rational Quantization“ als „Micro-Zones“ vorgestellten Verfahren. Das eigentliche Problem welches es zu lösen gilt, ist die Erweiterung auf Farbbilder. Ich habe mich an den Vorschlag aus [QTV] gehalten. Dieser sieht vor die

⁷außer natürlich die Implementation ist schnell genug

⁸der hier allerdings dem eigentlichen Intensitätswert *v* des Pixels entspricht

Farbe in den CIE-XYZ-Farbraum umzurechnen bzw. die Y-Komponente aus der gegebenen Farbe zu berechnen. Da dies sowieso Bestandteil des photometrischen Rahmens (siehe Kapitel 6) des Systems ist war die hierfür benötigte Funktionalität schon vorhanden und erforderte keinerlei Erweiterungen. Das Verfahren funktioniert auf folgende Weise:

- Umrechnung der RGB-Farbwerte, einer in der Szene vorkommenden Farbe, in den XYZ-Farbraum
- Skalierung der Y-Komponente mit URQ-Methode mit nach Formel 2 modifizierten Parameter p
- Neuberechnung der RGB-Farbkomponenten unter Verwendung der neuen Y-Komponente des Pixels

Das Verfahren beruht auf der Annahme, daß sich nur minimale Farbverschiebungen ergeben wenn bei der Skalierung die Verhältnisse der Farben erhalten bleiben. Ist z.B. $\max(val_{red}, val_{green}, val_{blue}) = val_{red}$ und $\exists(u, v) \in [0, 1]^2 \mid val_{green} = u * val_{red}$ und $val_{blue} = v * val_{red}$, so ergeben sich die neuen Farbkomponenten durch folgende Formeln:

$$val'_{red} = \frac{val'}{(M[0][1]) + (M[1][1] * u) + (M[2][1] * v)} \quad (3)$$

$$val'_{green} = u * val'_{red} \quad val'_{blue} = v * val'_{red}$$

Wobei val' der neue Intensitätswert bzw. Y-Kanal der Farbe ist und z.B. $M[2][1]$ den Wert in der zweiten Spalte und ersten Zeile der internen Skalierungsmatrix bezeichnet. (Die Werte der Skalierungsmatrix finden sich in Kapitel 6.1) „Interner Farbraum & Matrixdefinition“. Nach Implementation dieses Verfahrens und einigen Testläufen auf Bildern ergab sich allerdings folgendes Problem: Auch nach Anwendung des Verfahrens waren in der Szene immer noch Patches mit Farbwerten über 1 vorhanden. Diese drückten sich durch größere Flächen desselben Farbwerts an Stellen, an denen an sich ein Farbverlauf zu erkennen sein sollte. Dieser Effekt ergibt sich aus Formel 3. Bei den benutzten Szenen und der verwendeten Farbmatrix⁹ kommt es vor das die Summe im Nenner des Bruches kleiner ist als der Zähler. Die Folge sind Werte größer als 1. Dieser Effekt tritt nicht bei allen Szenen auf und somit kann die Technik oft erfolgreich eingesetzt werden. Allerdings muß der Benutzer die Qualität der Skalierungsoperation in jedem Falle selbst überprüfen und gegebenenfalls versuchen, diesen Effekt über eine Anpassung der beiden Parameter p und k zu entfernen. Soweit ich dies absehen kann, ist dies eine Schwäche des Algorithmus. Allerdings kann es natürlich auch sein, daß mir ein Fehler bei der Programmierung unterlaufen ist den ich beim Testen und Überprüfen der Ergebnisse nicht

⁹Die in [QTV] angegebenen Werte wurden ebenfalls getestet, da sie sich von den hier verwendeten unterscheiden.

gefunden habe. Die beschriebenen Effekte lassen sich in den Abbildungen 15, 16 und 17 nachvollziehen. Diese hier angedeutete Einschränkung wurde durch die Bereitstellung eines weiteren auf NURQ aufbauenden Verfahren gelöst.

4.5 Die Dame: NURQV

Auf diese Möglichkeit hat mich mein Betreuer aufmerksam gemacht und auf seine Anregung hin habe ich auch diese Variante noch umgesetzt. Die einfache aber elegante Idee:

- Rechne die gegebene Farbe in den HSV-Farbraum um
- Skaliere den V-Wert (entspricht der Helligkeit des Pixels) der Farbe nach der URQ-Methode mit einem durch Formel 2 modifizierten Parameter p
- Setze den neuen v-Wert der Farbe, lasse H und S unverändert
- Rechne die Farbe zurück in den RGB-Farbraum

Hierbei werden nicht die Formeln der eigentlichen NURQ-Methode benutzt, sondern eben die der fehlerfreie „normalen“ Variante des URQ-Verfahrens. Der V-Wert wird also mit den Formeln 1 und 2 skaliert bevor die Rückrechnung in den RGB-Farbraum erfolgt. Dies führt zu sehr guten Ergebnissen in denen vor allem die Flächeneffekte des NURQ-Verfahrens nicht auftauchen. Zusätzlich zeichnet er sich durch hervorragende Farberhaltung aus. Dieses läßt sich am deutlichsten im Vergleich mit dem URQ-Verfahren feststellen.

Insgesamt sind die implementierten Verfahren allesamt ordentlich schnell, und zwar so schnell das eine tatsächliche Auswertung der Laufzeiten nicht notwendig ist. Die bemerkbare Verzögerung beim anwenden der Verfahren rührt nicht von den verwendeten Algorithmen her sondern kommt durch das Neuzeichnen des Inhaltes des Darstellungsfensters zustande. Diese nimmt vor allem bei hoher Patchanzahl und bei eingeschaltetem Gouraud-Shading stark zu.

5 Bedienung der Anwendung

Um die implementierte Funktionalität auch bedienen zu können wurde eine Benutzerschnittstelle entworfen und umgesetzt. Die wesentlichen Bestandteile möchte ich nun vorstellen und soweit möglich Erläuterungen und Bedienungshinweise geben. Da in RADIOCITY OpenSG für die Darstellung der Szenen benutzt wird und auch die OpenGL-Schnittstelle zu Qt innerhalb von OpenSG realisiert ist, bot es sich an die vorhandenen Navigationsmöglichkeiten von OpenSG zu benutzen. Diese sind Bestandteil der SimpleSceneManager-Klasse. Generell ist zu sagen, daß eingeschränkte Szenenbearbeitungsmöglichkeiten

über manuelle Unterteilung von Flächen und das Hinzufügen von Lichtquellen möglich sind. Dies war nicht explizit Aufgabenstellung und eine starke Ausweitung dieses Aspektes (z.B. Verändern von Materialeigenschaften) hätte zu sehr in Richtung eines Szeneneditors geführt als zu dem geplanten System. Wichtiger waren hier die Möglichkeiten szenenabhängige und vom Benutzer bearbeitbare Berechnungsparameter und weitere Einstellungen abzuspeichern und verwerten zu können. Im Bild 2 ist das Hauptfenster der Applikation zu sehen.

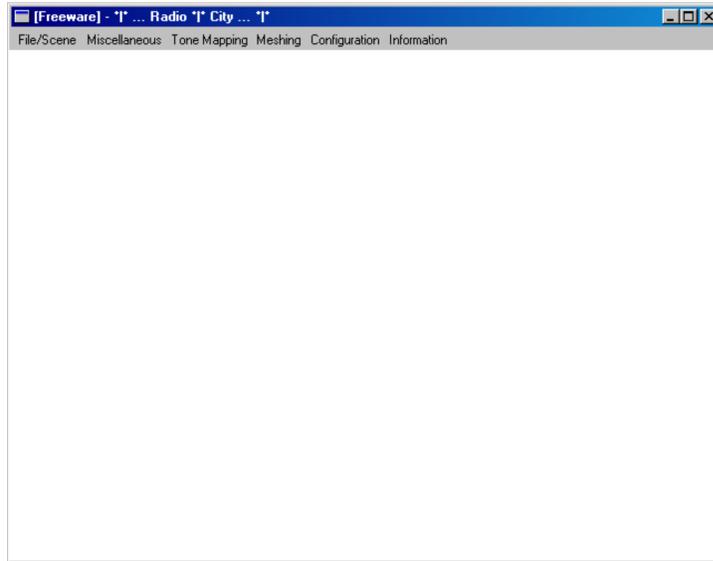


Abbildung 2: Hauptfenster von RADIOCITY

5.1 Menüeinträge

Zuerst ein paar Worte zu den vorhandenen Menüeinträgen, die Bildschirmaufnahmen hierzu finden sich im Abschnitt 8.1:

- File/Scene: Stellt alle notwendigen Dinge zur Verfügung um eine Szene einladen und abspeichern zu können. Desweiteren können hier Monitor-matrizen gespeichert werden und auch die beim Start automatisch erzeugte Standardszene kann bei Bedarf nachgeladen werden. (siehe Abbildung 10
 - Im Prinzip können alle Szenenformaten geladen werden die auch von OpenGL unterstützt werden, da die notwendigen Informationen aus dem OpenGL-Szenengraphen extrahiert werden. Diesen legt OpenGL jeweils aus dem angegebenen Dateiformat an. Am wichtigsten hierbei ist hierbei VRML V2.0 welches sich bequem aus vielen Szeneneditoren exportieren läßt.

- Über „Save scene parameters“ können alle relevanten Daten einer angepassten Szene abgespeichert werden. Diese werden in einem Textfile abgelegt welches automatisch den Namen der ursprünglichen Szenendatei ergänzt um einen kleinen Zusatz („_rcity.txt“) erhält. Diese wird beim späteren Laden der entsprechenden Szene ebenfalls eingeladen und die Berechnung kann wiederholt werden.
 - Die eingestellte Monitormatrix und der Gammawert des Monitors können über den Eintrag „Monitor related ...“ gespeichert und auch geladen werden. Dies erleichtert das Testen der photometrischen Konsistenz erheblich.
- **Miscellaneous:** Hier können das Gouraud-Shading und die Lambert emitter-Geometrien an bzw. ausgeschaltet werden. Zusätzlich läßt sich die Hintergrundfarbe des Darstellungsfensters zwischen schwarz und weiß umschalten. Am wichtigsten ist in diesem Menü aber der Eintrag „Hierarchical Step“ welcher einen Berechnungsschritt der Radiositysimulation ausführt. (siehe Abbildung 11) Dieser kann auch über die Taste H ausgelöst werden.
 - **Tone Mapper:** Hier können die Tone Mapping Verfahren aufgerufen werden. Beim Aufruf werden die im Konfigurationsdialog (siehe Abbildung 7) eingestellten Werte benutzt. Hier ist es auch möglich die Radiositywerte der Flächen als Farbe darzustellen. Die Verfahren können in beliebiger Reihenfolge angewandt werden, da die errechneten Radiositywerte erhalten bleiben und über „Set Radiosity as color“ wiederhergestellt werden können. Zusätzlich kann hier ein Herausrechnen des Monitorgammas aus den Farbwerten der Szene geschehen. Dies ist für photometrische Messungen notwendig.
 - **Meshing:** Hier kann eingestellt werden ob bei der Radiosityberechnung Flächen nur regulär unterteilt werden oder aber die Spezialunterteilung von Dominik [Rau] verwendet werden soll.
 - **Configuration:** Hier können Lichtquellen hinzugefügt und die Einstellungsdialoge für Radiosityverfahren, Monitor und Tone Mapping Verfahren aufgerufen werden.
 - **Information:** Anzeigen von Programminformationen

5.2 Vorhandene Dialoge

Leider ist es mir nicht möglich die Radiosityparameter und deren Einstellung ausführlich zu behandeln. Diese finden in der Studienarbeit von Guido [Stegmann] genauere Beachtung. Hier wird aus Gründen der Vollständigkeit nur der Dialog abgebildet.

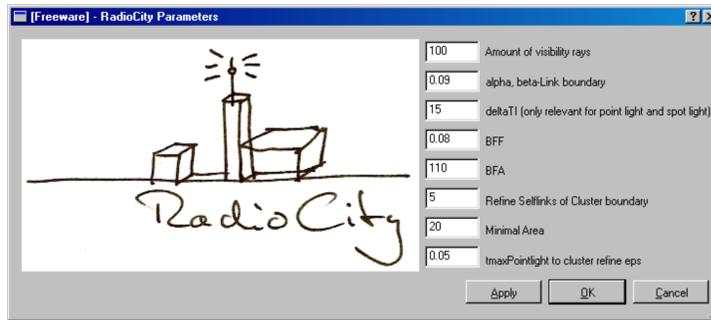


Abbildung 3: Dialog zur Einstellung der Radiosityparameter

Einstellungsmöglichkeiten der Lichtquellen

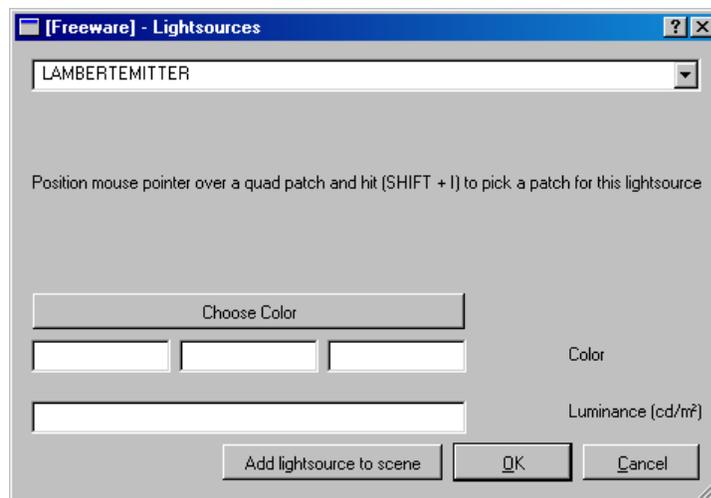


Abbildung 4: Dialog zur Einstellung der Werte des Lambert Emitters

Es können hier die Farbe und eine Leuchtdichte (cd/m^2) angegeben werden. Zusätzlich muß ein Patch aus der Szene ausgewählt werden. Über diesem wird dann in geringem Abstand der Lambert Emitter plaziert. Dies geschieht durch Navigation in der Szene und auswählen des gewünschten Patches durch Drücken der Tastenkombination (SHIFT + i). Hat man erfolgreich eine Fläche ausgewählt wird dies im Dialog angezeigt. Diese Lösung wurde gegenüber eines Tastendrucks auf der Maus bevorzugt, da andernfalls Schwierigkeiten mit der Navigation in der Szene entstehen.

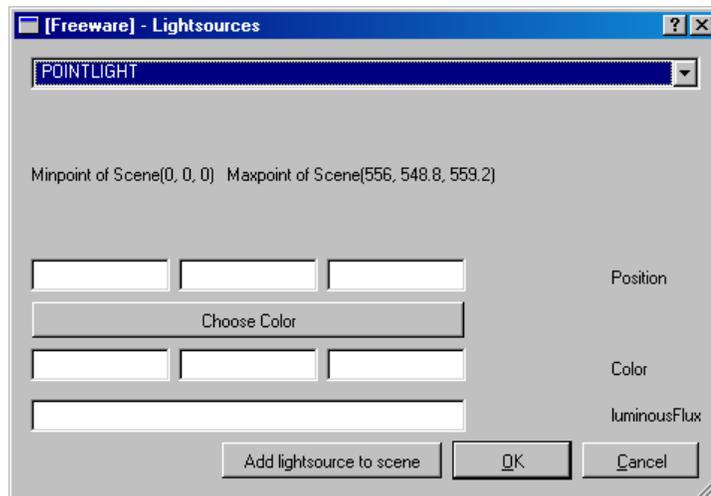


Abbildung 5: Dialog zur Einstellung der Werte der Punktlichtquelle

Bei der Punktlichtquelle wird ebenfalls eine Farbe und der Lichtstrom angegeben. Es muß hier zusätzlich die Position angegeben werden. Dieses kann manuell geschehen, oder aber durch das Auswählen einer Position innerhalb der Szene geschehen. Dafür muß der Mauszeiger an die entsprechende Stelle in der Szene gebracht werden und die Tastenkombination (SHIFT + *) gedrückt werden. Hat man alles richtig gemacht, wird die neue Position innerhalb des Dialogs eingetragen.

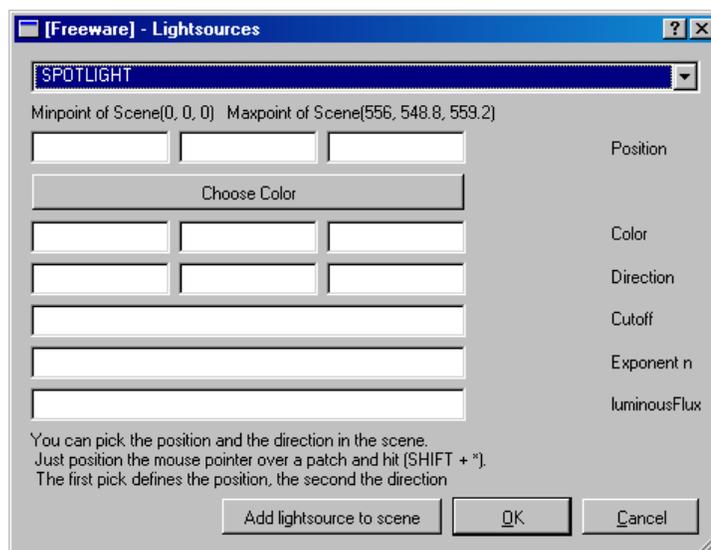


Abbildung 6: Dialog zur Einstellung der Werte der Spotlichtquelle

Hier muß eine Farbe, der Lichtstrom, der Cutoff(thetaMax), der Exponent

N, die Position sowie die Richtung angegeben werden. Die Position und die Richtung lassen sich wie im Dialog beschrieben auswählen.

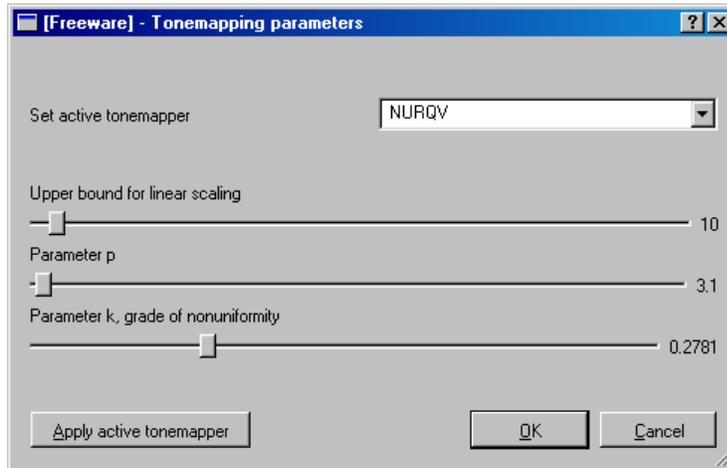


Abbildung 7: Dialog zur Einstellung der Tone Mapping Parameter

Über die Liste kann der aktive Tonemapper ausgewählt und über die Schaltfläche „Apply active tonemapper“ auf die Szene angewandt werden. Es existieren drei Parameter die reguliert werden können. Der erste ist nur für das lineare Verfahren von Bedeutung. Er gibt den Wert an auf den die Helligkeit der Lichtquellen skaliert werden soll. Der Parameter p ist für die Verfahren URQ, NURQ und NURQV relevant. Der Parameter k wird nur von NURQ und NURQV verwendet und gibt den Grad der Ungleichmäßigkeit (non uniformity) an. Die Parameter werden ausführlicher im Abschnitt 4 erläutert. Bei einigen Szenen und deren Intensitätsverteilung kann es vorkommen das der festgestellte Mittelwert der Intensitäten extrem klein bzw. fast 0 wird. In diesem Falle ergeben sich durch die interne Modifikation des Parameters p sehr hohe Werte für den bei den Quantisierungsverfahren NURQ und NURQV zur Anwendung kommenden Parameter p' . Das Ergebnis sind dann extrem hohe Farbwerte in der Szene die nicht mehr den Erwartungen entsprechen. Um dies zu vermeiden kann der Parameter k dann einfach auf 0 gesetzt werden und die Effekte verschwinden. Der Mittelwert wird intern zwar bei zu geringer Größe abgeschnitten, allerdings kann es sein, daß über die Oberfläche der Parameter k nicht klein genug gestellt werden kann um den Effekt zu vermeiden. Bei den meisten getesteten Szenen ließ sich der Parameter aber gut justieren.

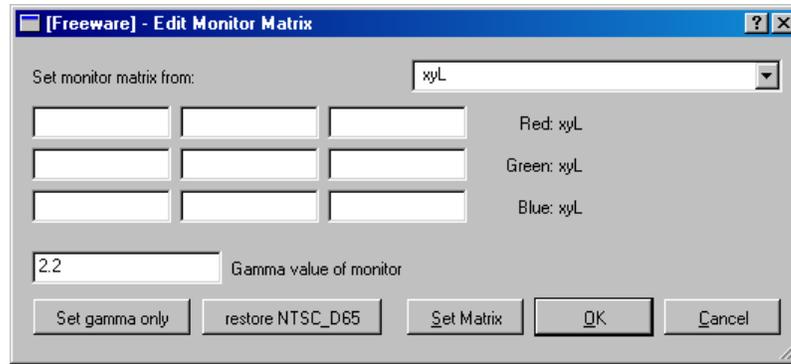


Abbildung 8: Dialog für die Konfiguration der Monitorparameter

Hier können entweder die Primärvalenzen des Monitors im XYZ-Format angegeben werden oder aber über ein Photometer die xyL-Werte für die drei Primärfarben und Weiß gemessen und dann eingetragen werden. Daraus wird dann die Monitormatrix erstellt und steht dann für Farbraumkonvertierungen zur Verfügung. Zusätzlich kann hier der Gammawert des Monitors angegeben werden. Dieser wird für die Funktion „Set gamma corrected values“ des „Tone Mapping“-Menüeintrags benötigt. Falls Fehler bei der Eingabe passieren kann die interne Matrix durch „restore NTSC_D65“ wiederhergestellt werden. Die Matrix und der Gammawert lassen sich einzeln setzen. Bei Betätigung der Schaltfläche „Ok“ werden beide Komponenten gesetzt.

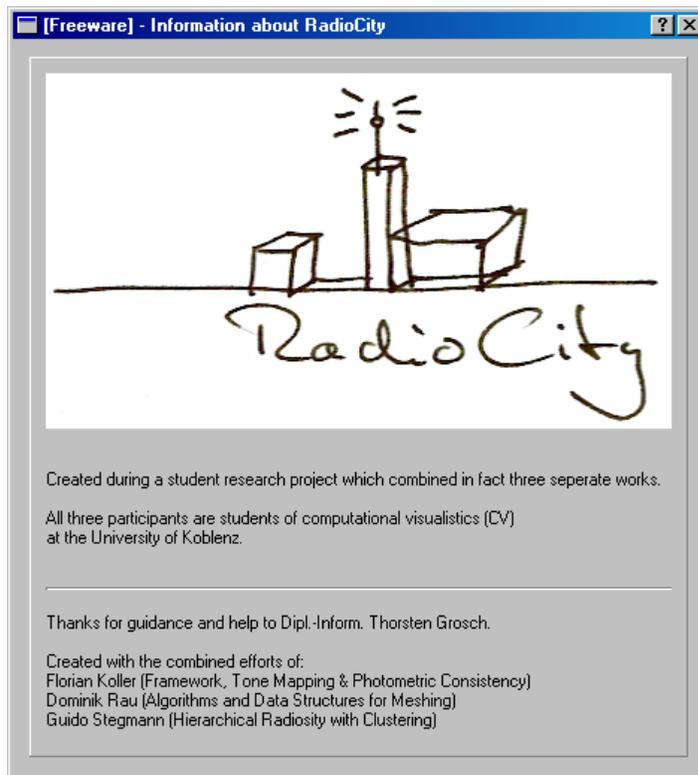


Abbildung 9: Informationsdialog mit den Autoren des Programms

5.3 Hinweis auf Inkonsistenzen

Es kann oder wird höchstwahrscheinlich der Fall sein, daß die hier aufgeführten Informationen und vor allem das Erscheinungsbild und der Aufbau der gezeigten Dialoge sich verändert hat, da man sich selbst von einem derartigen Projekt nicht, oder nur zögernd verabschiedet und es auch nach Abgabe dieser Studienarbeit noch zu Erweiterungen kommen wird. Dies kann sich in einigen Inkonsistenzen zwischen dem gerade aktuell verfügbaren Programm und dieser Ausarbeitung niederschlagen. Im Zweifelsfall sollte dann auf die neueren Quellen zurückgegriffen werden (Quelltext und Dokumentation bzw. eventuelle Begleitliteratur).

6 Photometrische Konsistenz & Farben

Für ein derartiges Rahmensystem, müssen zuerst grundlegende Funktionalitäten wie Farbklassen und Farbkonvertierungsfunktionen erstellt werden. Im Entwurf des Systems ist dementsprechend auch ein Farbkonzept mit angedacht und soweit nötig auch umgesetzt worden. Allerdings ist eine vollständige Integration und eine wirklich umfassende Funktionalität leider nicht mehr realisierbar gewesen (siehe Klassendiagramm, Dokumentation und Quelltext auf der CD (8.3)). Allerdings genügen die implementierten Klassen den Anforderungen des Rahmensystems und es kann mit ihnen gearbeitet werden. Zusätzlich sollte durch die sorgfältige Kapselung auch eine einfache Erweiterung um zusätzliche Konzepte leicht möglich sein. Zum Funktionsumfang der Farbklassen zählen auch einige Operatorüberladungen (siehe Abschnitt 3.2.1) welche das Arbeiten im Quelltext wesentlich vereinfachen.

In einem Rahmensystem für photometrische Konsistenz kommt es vor allem darauf an, die eingelesen und einzustellenden Daten innerhalb des Systems konsistent zu halten um dann bei der Ausgabe eine erfolgreiche Umrechnung in den Zielfarbraum des Ausgabegerätes vornehmen zu können. Sinn und Zweck des ganzen Aufwandes ist es, eine konsistente Interpretation von Farben, Leuchtdichten und Farbkonvertierungen zu ermöglichen. Hierbei müssen eine Vielzahl von kleinen Aufgaben gelöst werden.

Die Eingabedaten der Lichtquellen sind der zentrale Ansatzpunkt für ein solches System. Diese werden dem System anhand von photometrischen Daten vermittelt¹⁰ und unterscheiden sich dadurch von den klassischen Angaben einer Lichtquelle in anderen Systemen. Der Vorteil hierbei ist, daß bei Berücksichtigung der Spezifikation des Ausgabegerätes sämtliche photometrischen Daten auch korrekt simuliert und vor allem dargestellt werden können.

Ins System werden Daten eingelesen und Lichtquellen einer Szene hinzugefügt. Die Farben werden in einem internen Farbraum (siehe Abschnitt 6.1) interpretiert und die Lichtquellen über die vorhandene Funktionalität aus der Klasse `ColorCalibration` photometrisch konsistent gesetzt. Dies heißt, daß zum Beispiel die Leuchtdichte, die für eine Lampe angegeben wurde, im System erhalten bleibt und auch wieder extrahiert werden kann. Es wird also das „zu verteilende Licht“ tatsächlich mit der gewünschten Intensität verwendet. Kennt man sein Ausgabegerät und gibt Leuchtdichten an die im Darstellungsbereich des Zielgerätes liegen, so werden tatsächlich später auch diese Werte¹¹ dargestellt, d.h. sie sind direkt am Gerät meßbar.

Intern wird immer mit Radiositywerten B gerechnet, auch wenn dies nicht zwingend vorgeben ist. Das gesamte Konzept basiert auf einer Konvertierung der photometrischen Größen (Farbe, Leuchtdichte) in eine photometrisch konsistente Farbe L_r , L_r , L_b . Für jede der Lichtquellen existiert mindestens ei-

¹⁰z.B. Leuchtdichte, Lichtstrom und Lichtstärke

¹¹Abweichungen von bis zu 10 % sind allerdings aufgrund von Meßungenauigkeiten möglich

ne `set(...)`-Funktion mit der die photometrischen Eingangsdaten in eine „Farbe“¹² umgewandelt werden. Bei Punkt- und Spotlichtquelle wird über die `set()`-Methode zuerst die Lichtstärke I bestimmt welche dann beim Aufruf der `send()`-Methode und Einbeziehung in die Radiositygleichungen den Wert für B liefert. Aus den so in die Radiositysimulation eingebrachten Werten können nach dem Ende der Berechnung die photometrischen Informationen rekonstruiert werden. Dieser Vorgang sieht z.B. für den Lambert Emitter so aus:

- Benutzer gibt die Farbe (R, G, B) und die Leuchtdichte L in das System
- Intern wird mit den Formeln 4 und 5 die photometrisch konsistente Farbe gesetzt.

$$(L_r, L_g, L_b) = scale * (R, G, B) \quad (4)$$

$$scale = \frac{L}{K_m * (M [0] [1] * r + M [1] [1] * g + M [2] [1] * b)} \quad \text{mit } K_m = 683 \quad (5)$$

Über die Formel 6 kann z.B. dann für ein Patch die Leuchtdichte L_{patch} berechnet werden.

$$L_{patch} = K_m * (M [0] [1] * r + M [1] [1] * g + M [2] [1] * b) \quad (6)$$

Sämtliche dem System zugeführten Farben (seien es die der Lichtquellen oder Materialfarben) werden im internen Farbraum interpretiert. Dieses ist hier der Standard-NTSC-Farbraum mit dem Weißpunkt D65. Das System ließe sich allerdings auch auf die explizite Angabe einer Konvertierungsmatrix pro Farbe erweitern. In diesem Fall müßte eine Konvertierung vom definierten Farbraum in den internen Farbraum erfolgen¹³. Der Benutzer kann seinen Monitor ausmessen und definiert damit die zur Ausgabe verwendete Monitormatrix. Falls diese nicht definiert wird, werden die Farben im internen Farbraum dargestellt. Die Darstellung ist völlig ausreichend für die normale Benutzung des Systems. Falls allerdings photometrische Konsistenz gewährleistet werden soll, so muß eine Monitormatrix angegeben werden. Bei Angabe einer solchen werden automatisch die notwendigen Konvertierungen vor der Darstellung auf dem Zielmonitor vorgenommen. Die Monitormatrix wird ebenfalls innerhalb der generierten Konfigurationsdatei gespeichert. Dies gilt es bei der Verwendung einer vorkonfigurierten Szene auf einem anderen Rechner bzw. Ausgabegerät zu beachten. Im Falle, daß auf einem anderen Rechner mit denselben Einstellungen gearbeitet wird kommt es zu Farbverschiebungen und Fehlern. Dies kann durch

¹²Leuchtdichtewerte L_r, L_g, L_b oder Lichtstärke/Strahlstärkewerten I_r, I_g, I_b

¹³siehe Quelltext: `ColorCalibration::rgb1_to_rgb2()`

ein Rücksetzen der Monitormatrix auf die interne Skalierungsmatrix behoben werden (siehe Monitormatrix-Dialog im Kapitel 5).

Für eine Einführung in die Grundlagen der Photometrie im Zusammenhang mit Radiosity empfehle ich [PaR] und auch [RPP]. Das im Rahmen der Studienarbeit umgesetzte Konzept stützt sich hauptsächlich auf [Arcade] und den von Hr. Prof. Müller in der Vorlesung PCG (Photorealistische Computergraphik) im Wintersemester 2002/2003 vorgestellten Ansatz.

6.1 Interner Farbraum, Matrixdefinition & Funktionen

Die intern verwendete Skalierungsmatrix wird durch folgende Werte generiert:

- xy-Rot: (0.670, 0.330)
- xy-Grün: (0.210, 0.710)
- xy-Blau: (0.140, 0.080)
- xy-Weißpunkt: (0.313, 0.329)

x und y sind dabei die vereinfachten Farbkomponenten XYZ. Sie werden erzeugt indem die jeweilige Komponente durch die Summe der drei Komponenten geteilt wird. Die oben genannten Werte werden dann in die Funktion `ColorCalibration::getMatrix_xy(xy-Rot, xy-Grün, xy-Blau, xy-Weißpunkt)` gegeben. Diese generiert die interne Skalierungsmatrix M

0.589	0.179	0.183
0.290	0.605	0.105
0.000	0.068	1.020

Vergleiche hierzu: ([IaC], [AIM], [ColorM], [Arcade])

Bei der Verwendung und dem Test eines Konsistenzmodells kommt es bedingt durch die Notwendigkeit Messungen durchzuführen zu einigen Problemen. Eines ist davon z.B. die Tatsache, daß Arbeitsplätze von mehreren Leuten benutzt werden und man deswegen nie sicher sein kann ob die Monitoreinstellungen noch mit denen zum Zeitpunkt der Messung übereinstimmen. Daraus ergibt sich das Problem, daß grundlegende Monitorausmessungen oft wiederholt werden müssen. Dementsprechend wären Erleichterungen für dieses ständig notwendige Ausmessen von Vorteil für die Handhabung des Systems. Einige der für die photometrische Konsistenz notwendigen Funktionen sind z.B.:

- `OSG::Matrix4f getMatrix_xy (...) const`
- `OSG::Matrix4f getMatrix_XYZ (...) const`

- `OSG::Matrix4f getMatrix_xyL (...) const`
- `OSG::Matrix4f getMatrix_NTSC_D65 (...) const`
- `OSG::Matrix4f getMatrix_SMPTE_C (...) const`
- `OSG::Matrix4f getMatrix_rgbXYZ (...) const`
- `OSG::Vec3f XYZ_to_xy (...) const`
- `OSG::Vec3f rgb_to_XYZ (OSG::Vec3f const &rgb, OSG::Matrix4f const &M) const`
- `OSG::Vec3f rgb_to_XYZ (OSG::Vec3f const &rgb) const`
- `OSG::Vec3f XYZ_to_rgb (OSG::Vec3f const &XYZ, OSG::Matrix4f const &M) const`
- `OSG::Vec3f XYZ_to_rgb (OSG::Vec3f const &XYZ) const`
- `OSG::Vec3f xyL_to_XYZ (...) const`
- `OSG::Vec3f rgb1_to_rgb2 (...) const`
- `OSG::Real32 getLuminousFlux (...) const`
- `OSG::Real32 getLuminance (...) const`
- `OSG::Real32 getRadiosity (...) const`

Bei sämtlichen hier aufgeführte Funktionen wurden die Funktionsparameter aus Platzgründen weggelassen. Eine Ausnahme wurde nur dort gemacht wo Überladungen der Funktionen existieren. Die vollständige Klassendokumentation enthält alle fehlenden Parameter und liegt der CD im HTML-Format bei.

6.2 Gammakorrektur

Für die Messung der Leuchtdichte an einem Monitor gilt es die Gammakorrektur des Monitors und die von weiteren zwischengeschalteten Komponenten zu berücksichtigen. Diese verändert die dargestellte Helligkeit der Farbwerte, weißt ihnen also aufgrund des voreingestellten Gammawertes eine neue Helligkeit zu. Dies gilt es zu korrigieren, da sonst die intern berechneten Leuchtdichtewerte nicht mit den extern gemessenen übereinstimmen. Im System ist zu diesem Zweck eine Gammakorrektur eingebaut.

Die Bestimmung des Gammawertes eines Monitors/Systems läuft auf folgende Weise ab:

- Bestimme die maximale Leuchtdichte des Monitors L_{max}
- Zeige auf dem Monitor Grauwertbilder C an für die gilt: $0 \leq C \leq 1$, wobei 1 dem höchsten Intensitätswert des Monitors entspricht.
- Messe zu den Grauwertbildern die Leuchtdichte L_C

- Berechne γ und C^γ mit den Formeln 7 und 8
- Wiederhole diese Schritte bis ausreichend Daten vorhanden sind
- Errechne den Mittelwert aus den berechneten Gammawerten
- Führe die Gammakorrektur mit dem ermittelten Gammawert des Systems durch

Nach Durchführung der Gammakorrektur können die intern errechneten photometrischen Informationen, z.B. die Leuchtdichte einer Fläche, direkt abgemessen werden und die Ergebnisse sollten, innerhalb der Meßgenauigkeit, mit den errechneten Werten übereinstimmen.

$$C^\gamma = \frac{L_C}{L_{max}} \quad (7)$$

$$\gamma = \frac{\log\left(\frac{L_C}{L_{max}}\right)}{\log(C)} \quad (8)$$

7 Zusammenfassung & Kritik

Im Anhang finden sich zwar einige Bilder, allerdings blieb gegen Ende leider keine Zeit mehr komplexere Szenen zu modellieren und zu berechnen.

Zusätzlich läßt der Komfort der Oberfläche zum Teil auch noch Wünsche offen, so fehlt teilweise auch noch das Ausblenden/Deaktivieren bestimmter Oberflächeneigenschaften bei schon berechneter Szene. Diese Erweiterung sollte allerdings schnell umsetzbar sein, da die Struktur des Programms daraufhin ausgelegt ist und nur an zentraler Stelle, in der Klasse RadioCityUI, Änderungen vorgenommen werden müssen. Zusätzlich ließe sich bestimmt noch ein bisschen Schädlingsbekämpfung im Quelltext erledigen.

Im Tagesgeschäft blieb leider zumindest ein Strukturkonzept auf der Strecke. Hierbei benötigte Hr. [Stegmann] eine Möglichkeit die einzelnen Elemente der Radiositysimulation unterscheiden zu können. Dabei kamen wir zu keinem anderen Ergebnis als dies teilweise über virtuelle Methoden und leider auch über sogenannte Typenbezeichnungen zu lösen. Dieses endete dann in der Verwendung von sogenannten „Downcasts“ innerhalb der Klassenhierarchie. Dadurch wird die Erweiterbarkeit eingeschränkt bzw. erschwert.

7.1 Optimierungen für künftige Generationen

oder: *Wir mußten draußen bleiben*

- Relativ spät kam der Bedarf nach sogenannte traverse-Funktionen auf mit denen ähnlich zu den traverse-Funktionen von [OpenSG] die Szenenelemente durchlaufen werden können und jeweils eine bestimmte Funktion

für das aktuelle angewendet wird. Diese hätten bei der Implementierung der Tone Mapper und beim durchlaufen der Szenelemente für das Sammeln, Verschießen und generellen Traversierung unserer Hierarchie gute Dienste geleistet. Da allerdings gegen Ende andere Dinge Vorrang hatten existiert als Folge dessen häufig identischer Quelltext zum Abläufen der Blätter der Hierarchie. Also hat sich hier die typische „Copy and Paste“-Problematik eingeschlichen

- Normieren der Szene um dadurch besser kontrollierbare Radiosityparameter zu erhalten
- Die Rückmeldung über den Stand der Radiosityberechnung über die Qt-Klasse ProgressBar fehlt noch. Dies verursacht des öfteren Probleme, da bei größeren Szenen nicht ganz klar ist wie lange der Algorithmus noch läuft.
- Momentan hat nur die Surface der die Patches angehören einen Rhowert und damit kann es passieren, daß falls die Eingabedaten nicht für unser System vorbereitet wurden, beim Laden noch alle Patches mit eigener Farbe dargestellt werden können, nach der Berechnung allerdings alle denselben Farbwert haben. Zur Lösung dieses Problems könnte eine Erweiterung der Patches um eigene Rhowerte erfolgen, dies würde aber den Speicherbedarf der Szene wesentlich erhöhen.
- Ein schöne Eigenschaft des System kam kurz vor Abgabe der Arbeit abhanden. Durch eine Änderung an den Normalen der Flächen ging die Möglichkeit eine intern, über manuelle Patchberstellung, erzeugte Szene mit verschiedenen Lichtquellenkonfigurationen zu speichern verloren. Hiermit hätten die intern erstellten Szenen auch in Dateien gespeichert werden können. Diese Möglichkeit läßt sich im Prinzip wiederherstellen, ist aber nach Einschätzung von Hr. [Rau] zeitaufwändiger und mußte dementsprechend unbearbeitet bleiben.

7.2 Mögliche Erweiterungen und Ausblick

Mögliche Erweiterungen des Systems umfassen z.B. die Integration von LVK und Tageslicht (mit der LVK könnten dann real existierende aus Herstellerangaben stammende Abstrahlverhalten von Lampen simuliert werden). Zusätzlich wäre eine komfortable Testumgebung für die Photometrische Konsistenz denkbar, mit der dann schnell und einfach der Gammawert des Monitors automatisch mit Hilfe der Meßwerte ausgerechnet wird. Dort könnten auch die intern berechneten Größen in aufbereiteter Form dargestellt werden und eine einfache Analyse der photometrischen Szeneneigenschaften (z.B. „dynamic range“ der Szene) wäre so möglich. Zusätzlich sind Möglichkeiten der Erleichterung der Ausmessung des Monitors denkbar, da dies ein relativ zeitraubendes

Verfahren darstellt. In einer Erweiterungsphase könnte RADIOCITY auch um Texturen erweitert werden, und das Zeichnen der Informationsgeometrien der Lichtquellen Pointlight und Spotlight integriert werden.

Es existiert nun die Möglichkeit, aufbauend auf unserer Funktionalität auch komplexere Szenen für RADIOCITY zu erstellen und mit hoher Wahrscheinlichkeit eine erfolgreiche Berechnung durchführen zu können. Eine Möglichkeit wäre es, die beiliegende Museumsszene ([Rau]) um noch komplexere Exponate zu erweitern. Dies stellt wahrscheinlich gerade für den Einarbeitungsprozeß in RADIOCITY eine akzeptable Lösung dar, da hierfür die Parameter voreingestellt verfügbar sind. Das Finden der richtigen Parameter dürfte, neben der externen Szeneerstellung, auch insgesamt betrachtet das zeitaufwändigste Problem darstellen.

Ich hoffe das sich einige interessierte Leser mit der beigefügten CD beschäftigen und vor allem einen Blick auf RADIOCITY werfen. Für dieses liegen berechenbare Szenen bei und der Umgang mit der Anwendung sollte mit den gegebenen Erläuterungen bewältigbar sein. Besonders freuen würde ich mich über konstruktive Kritik und kritische Nachfragen zum Aufbau des Systems. Gerne auch tiefergehende Betrachtungen bis auf Quelltextebene können Bestandteil der Fragen sein. Falls Ihnen diese Arbeit oder ich persönlich bei Ihren Fragen nicht helfen konnte werden sie vielleicht in den Studienarbeiten von Guido Stegmann und Dominik Rau fündig.

Literatur

- [PaR] Ian Ashdown, P.Eng.: *Photometry and Radiometry, A Tour Guide for Computer Graphics Enthusiasts* Adapted Tutorial from *Radiosity: A Programmer's Perspective* (siehe Referenz [RPP])
- [QTV] Schlick, Christophe: *Quantization Techniques for Visualization of High Dynamic Range Pictures*, Fifth Eurographics Workshop on Rendering (Darmstadt, Germany), p7-18, June 1994
- [RPP] Ian Ashdown: *Radiosity: A Programmer's Perspective*, John Wiley & Sons, Inc., 1994
- [RGI] Francois X. Sillion, Claude Puech: *Radiosity & Global Illumination*, Morgan Kaufmann Publishers, Inc., 1994
- [TR] J. Tumblin, H.Rushmeier: *Tone Reproduction for Realistic Computer Generated Images*, IEEE Computer Graphics & Applications, v13, n6, p42-48, 1993
- [IaC] R. Hall: *Illumination and Color in Computer Generated Imagery*, Springer Verlag, 1989
- [DCP] Bjarne Stroustrup: *Die C++ Programmiersprache*, 4. aktualisierte Auflage, Addison-Wesley, 2000
- [ECP] Scott Meyers: *Effektiv C++ programmieren, 50 Wege zur Verbesserung Ihrer Programme und Entwürfe*, 3. Auflage, Addison-Wesley, 1998
- [WCPP] André Willms: *work shop C++*, Addison Wesley, 2000
- [Arcade] Stefan Müller: *ARCADE: Automatic Radiosity simulation for Complex And Dynamic Environments*, ESPRIT Project Program Reactive LTR Project 24944, April 2000
- [EAR] Wolfram Kresse: Diplomarbeit *Effizientes und anwendbares hierarchisches Radiosity unter besonderer Berücksichtigung der Visibilitätsbetrachtung*, Technische Universität Darmstadt, Fachbereich Informatik, Oktober 1997
- [Rau] Dominik Rau: *Meshing - Datenstrukturen und Algorithmen für das RADIOCITY-System*, Universität Koblenz-Landau: Institut für Computervisualistik Arbeitsgruppe Computergraphik, November 2003

- [Stegmann] Guido Stegmann: *Hierarchisches Radiosity mit Clustering*, Universität Koblenz-Landau: Institut für Computervisualistik Arbeitsgruppe Computergraphik, November 2003
- [AIM] *Accurate Image Manipulation for Desktop Publishing*, http://www.aim-dtp.net/aim/technology/cie_xyz/cie_xyz.htm
- [ColorM] *Color Management part 1: Introduction*, http://www.normankoren.com/color_management.html
- [ToMap] <http://tecfa.unige.ch/perso/staf/bazargan/IN3/Photo/index.html>
- [XYZ] <http://semmix.pl/color/models/emo10.htmw>
- [CGammut] <http://www.wasatchinc.com/digigrafix.gamut.html>
- [CAT] <http://www.ee.washington.edu/conselec/CE/kuhn/ntsc/95x4.htm>
- [GFAQ] www.inforamp.net/~poynton
- [OpenSG] OpenSG: <http://www.opensg.org/>
- [CornellWeb] <http://www.graphics.cornell.edu/online/box/>

8 Anhang/Zusätze

8.1 Bildschirmaufnahmen der Menüeinträge

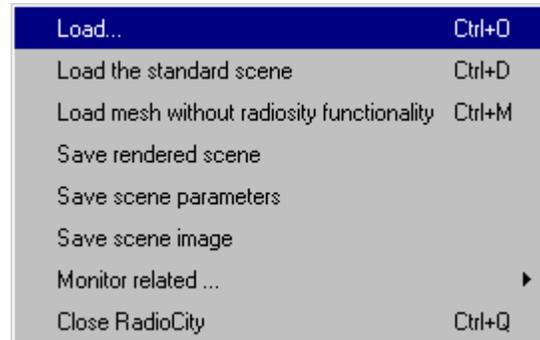


Abbildung 10: Menüeintrag File/Scene

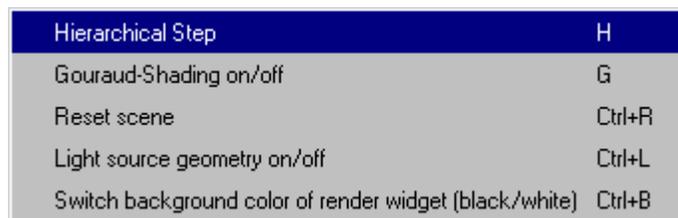


Abbildung 11: Menüeintrag Miscellaneous

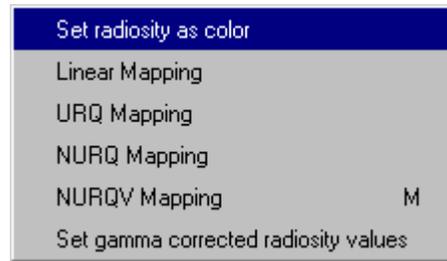


Abbildung 12: Menüeintrag Tone Mapper



Abbildung 13: Menüeintrag Meshing

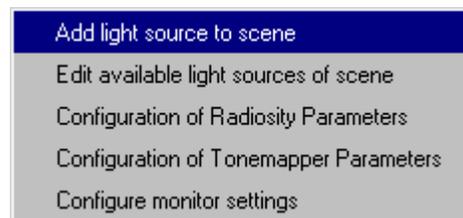


Abbildung 14: Menüeintrag Configuration

8.2 Kleine Installationshilfe für RadioCity

Hier führe ich kurz aus wie bei einer Installation unter Windows vorzugehen ist. Wenn man RADIOCITY nur benutzen möchte und keinerlei Änderungen vornehmen will so sollte sich die Installation unkompliziert gestalten.

- – **Installation der vorkompilierten Version:**
 - Kopieren der Dateien aus dem Ordner RadioCity/msVersion oder RadioCity/stlportVersion auf der beiliegenden CD auf den Rechner auf dem RADIOCITY ausgeführt werden soll.
 - Ausführen von radiocity.exe
 - Falls das Programm nicht laufen sollte kann es eventuell daran liegen das noch weitere .dll-Dateien fehlen ¹⁴. In diesem Falle müssen diese in dasselbe Verzeichnis wie die Datei radiocity.exe gelegt werden
 - Falls weiterhin Probleme existieren empfehle ich die beiliegende Distribution von OpenSG zu installieren und gegebenenfalls auch explizit noch Qt zu installieren.
 - Hinweis: RadioCity wurde unter Microsoft Visual Studio .NET entwickelt, sollte aber keine der .NET-eigenen Dateien verwenden. Verifizieren konnte ich dies allerdings nicht. Es könnte also dennoch sein das .NET installiert sein muß.
 - Falls alles nicht klappen sollte: Linux und OpenSG installieren und erneut versuchen.
- – **Entwicklungsinstallation:**
 - Installation von OpenSG (OpenSG-1.2.0-cl.net-ms.stl.exe (liegt der Arbeit bei)). Falls eine neuere Version benutzt wird, kann es natürlich zu Problemen kommen, da OpenSG ein System ist welches konstant weiterentwickelt wird und vielleicht einige der verwendeten Dinge nicht mehr laufen.
 - Installation von Qt (Qt 2.3 NonCommercial, befindet sich auf der beiliegenden CD)
 - Installation von Microsoft Visual Studio .NET (Dies kann ich allerdings nicht sonderlich empfehlen, da einige der Einstellungen mühsam über die Oberfläche getätigt werden müssen wenn man die Dialoge ändern möchte)
 - Die Quelltextdateien müssen aus dem Repository geholt werden (für die aktuelle Version) und in dasselbe Verzeichnis gelegt werden wie die Projektdatei radiocity.vcproj. Falls kein Zugang zum Repository möglich ist einfach die unter RadioCity/Entwicklungsversion liegenden Dateien verwenden.

¹⁴Vielleicht sind diese hier zu finden <http://www.dll-files.com/>

- Hinzufügen der Installationspfade von Qt und OpenSG dies kann entweder projektübergreifend oder für ein einzelnes Projekt geschehen. Es müssen die Pfade `qtinstalldir/include`¹⁵, `qtinstalldir/lib` und dasselbe nochmal für `opensginstalldir/include` und `opensginstalldir/lib`
 - Ausführen der Datei `generateUI.bat` (dieses ist für die Generierung einiger Dialogdateien erforderlich). Falls bei der Generierung Fehler auftreten ist wahrscheinlich die `PATH`-Variable für den Qt-Installationspfad nicht gesetzt. Diese muß dann von Hand innerhalb von Windows geändert werden. Die notwendige Angabe sieht etwa so aus: `PATH = qtinstalldir/bin/`
 - Ersetzen der „defekten“ Qt-Headerdateien in `qtinstalldir/include/` durch die unter `RadioCity/Entwicklungsversion` liegenden Dateien `qdict.h`, `qintdict.h` und `qlist.h`. Dies ist notwendig, da sonst das Projekt nicht kompiliert werden kann.
 - Kompilieren.
- **Entwicklungsinstallation mit STLPort:**
Um OpenSG und somit auch `RADIOCITY` mit dem STLPort zu verwenden muß statt der oben genannten Version die STLPort-Version (`OpenSG-1.2.0-cl.net-stlport.exe`) für die Installation verwendet werden. Zusätzlich muß noch ein Linker-Einstellung vorgenommen werden (hinzufügen von `stlport_vc7.lib`) und als allererstes Include-Verzeichnis innerhalb von `.NET` muß noch das Verzeichnis `opensginstalldir/include/stlport` dem Projekt hinzugefügt werden. Nur dann wird automatisch auch der STLPort verwendet und nicht die STL-Version von Microsoft (diese hat den Ruf relativ langsam zu sein).
 - Falls Probleme mit der Installationshilfe nicht zu lösen sind liegt im Quelltextverzeichnis (oder Repository) noch die `qtrelated`-Datei in der teilweise ausführlichere Erklärungen zu finden sind. Ein Blick lohnt sich auf jeden Fall. Falls Probleme mit Qt und den für die Dialoge benötigten `moc`-Dateien oder dem `uic`-Kompiler auftreten so empfehle ich das Designer-Tutorial welches Qt 2.3 beiliegt (`qtinstalldir/doc/html/designer/book1.html`). Desweiteren gibt es unterschiedlichen Möglichkeiten die Qt Zusatzaufufe (`uic`, `moc` etc. ;-)) zu lösen (hier über `.bat`-Datei oder `.NET`-Einstellungen gelöst) zu realisieren die dort auch angedeutet werden.

¹⁵mit `qtinstalldir` und `opensginstalldir` sind die lokalen Pfade der Qt- und OpenSG-Installation gemeint

8.3 Inhalt der beiliegenden CD

Auf der CD befinden sich unter anderem:

- Installationsdateien für OpenSG und Qt
- Die Quelltexte sowie deren Dokumentation
- Das vollständige Klassendiagramm in elektronischer Form
- Eine vorkompilierte Version von RadioCity für Windows
- Einfarbige Bilder zur Ausmessung von Monitoren

8.4 Ansichten berechneter Szenen & Bilder der Quantisierungsverfahren

Abbildungsverzeichnis

1	Aufbau von RADIOCITY	8
2	Hauptfenster von RADIOCITY	17
3	Dialog zur Einstellung der Radiosityparameter	19
4	Dialog zur Einstellung der Werte des Lambert Emitters	19
5	Dialog zur Einstellung der Werte der Punktlichtquelle	20
6	Dialog zur Einstellung der Werte der Spotlichtquelle	20
7	Dialog zur Einstellung der Tone Mapping Parameter	21
8	Dialog für die Konfiguration der Monitorparameter	22
9	Informationsdialog mit den Autoren des Programms	23
10	Menüeintrag File/Scene	33
11	Menüeintrag Miscellaneous	33
12	Menüeintrag Tone Mapper	34
13	Menüeintrag Meshing	34
14	Menüeintrag Configuration	34
15	(links) Visualisierung der Radiositywerte der Patches der Szene. Vergleiche dazu die Bilder 16 und 17	40
16	(rechts) NURQ skaliertes Bilde mit $p=300$, $k = 0.1$, Fehlerhafte Bereiche sind zu erkennen. Vergleiche hierzu: Kapitel 4.4	40
17	NURQV: mit $p=300$, $k = 0.1$, Im Vergleich mit Abbildung 16 keine abgeschnittenen Farbwerte mehr zu erkennen	40
18	(links) URQ-skaliertes Bild der nachgebauten Cornellbox	41
19	(rechts) NURQ-skaliertes Bild der nachgebauten Cornellbox	41
20	(links) NURQV-skaliertes Bild der nachgebauten Cornellbox	41
21	(rechts) Bild der Cornellbox (siehe[CornellWeb]; die sichtbaren Unterschiede kommen vor allem durch die komplexeren verwen- deten Materialeigenschaften zustande.)	41
22	Kleiner Raum: 3 Lambert Emitter (Seitenwände (hohe Inten- sität) und Deckenstrahler (schwache Intensität))	42
23	Kleiner Raum: 1 Pointlight	42
24	Kleiner Raum: 1 Spotlight	43
25	Badansicht 1 ohne Farben	43
26	Badansicht 2 mit Farben	44
27	Ansicht 1 der Museumsszene mit Objekten	44
28	Ansicht 2 der Museumsszene mit Objekten	45
29	Gitternetzansicht der beleuchteten Museumsszene	45
30	Nahansicht 1 eines Objektes	46
31	Nahansicht 2 eines Objektes	46

32 Büroszene Ansicht 1: 4 Lambert Emitter (Monitore + Deckenlampen, 2 Spotlichtquellen (Tischleuchten) 47

33 Büroszene Ansicht 2: 4 Lambert emitter (1 sichtbar, 2 Spotlichtquellen) 47

34 Museum mit rotem Spotlight, Schlagschatteneffekte 48

35 von hinten beleuchtetes Turtlemodell (turtle.wrl): 1 Lambert emitter 48

36 Ausschnitt aus berechneter Museumszene ohne Objekte 49

Hinweis: Teilweise bestehen Abweichungen zwischen den digitalen Bildern und den Druckergebnissen. Zusätzlich gehen beim Erzeugen der Bilder teilweise Farbinformationen verloren. Die Originaldaten lassen sich am Besten selber mit dem Programm erzeugen.

Für die Museumsszene möchte ich mich herzlich bei Dominik Rau bedanken. Die benutzte Szene des Badezimmers stammt von Eva Irek (Fraunhofer IGD) und die internen Testszenen wurden zum großen Teil von Guido Stegmann angelegt.

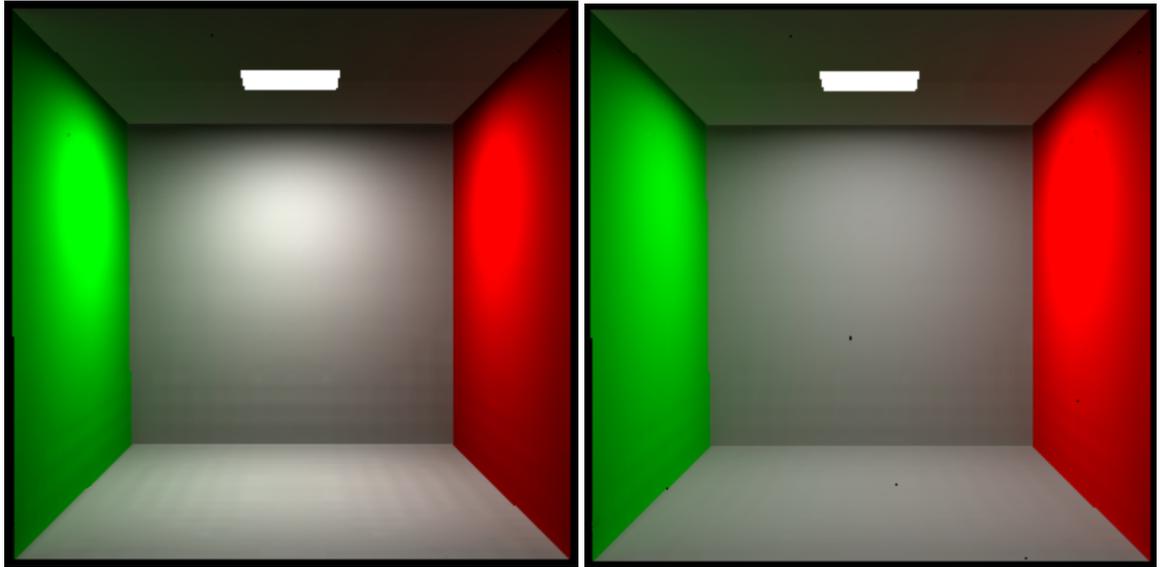


Abbildung 15: (links) Visualisierung der Radiositywerte der Patches der Szene. Vergleiche dazu die Bilder 16 und 17

Abbildung 16: (rechts) NURQ skaliertes Bilde mit $p=300$, $k = 0.1$, Fehlerhafte Bereiche sind zu erkennen. Vergleiche hierzu: Kapitel 4.4

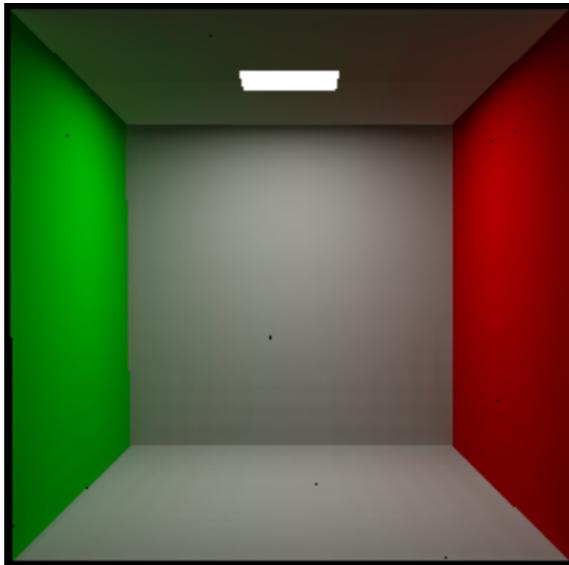


Abbildung 17: NURQV: mit $p=300$, $k = 0.1$, Im Vergleich mit Abbildung 16 keine abgeschnittenen Farbwerte mehr zu erkennen

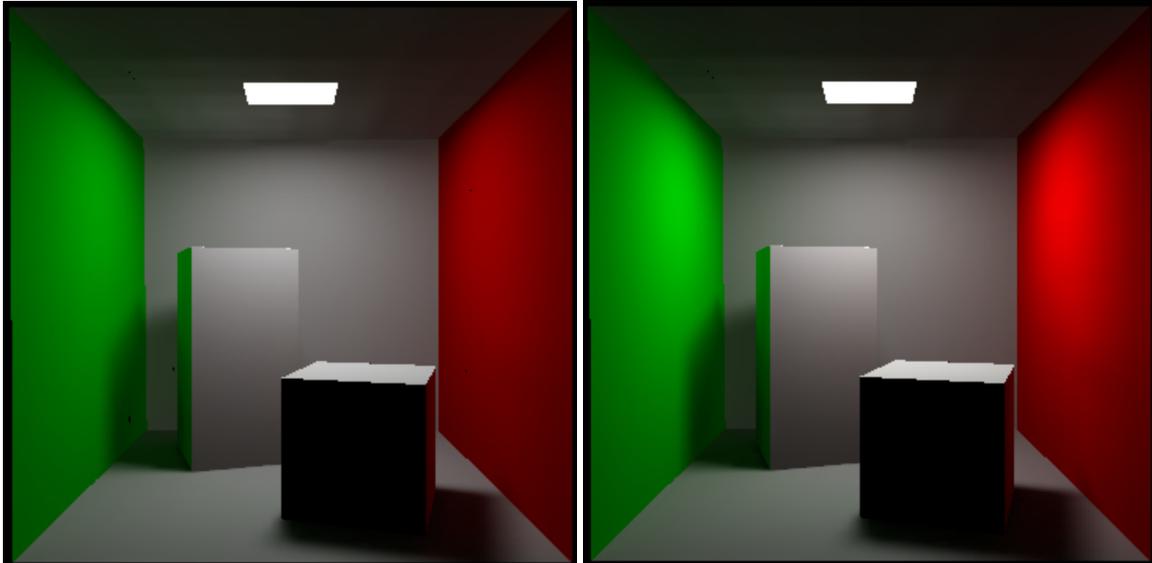


Abbildung 18: (links) URQ-skaliertes Bild der nachgebauten Cornellbox

Abbildung 19: (rechts) NURQ-skaliertes Bild der nachgebauten Cornellbox

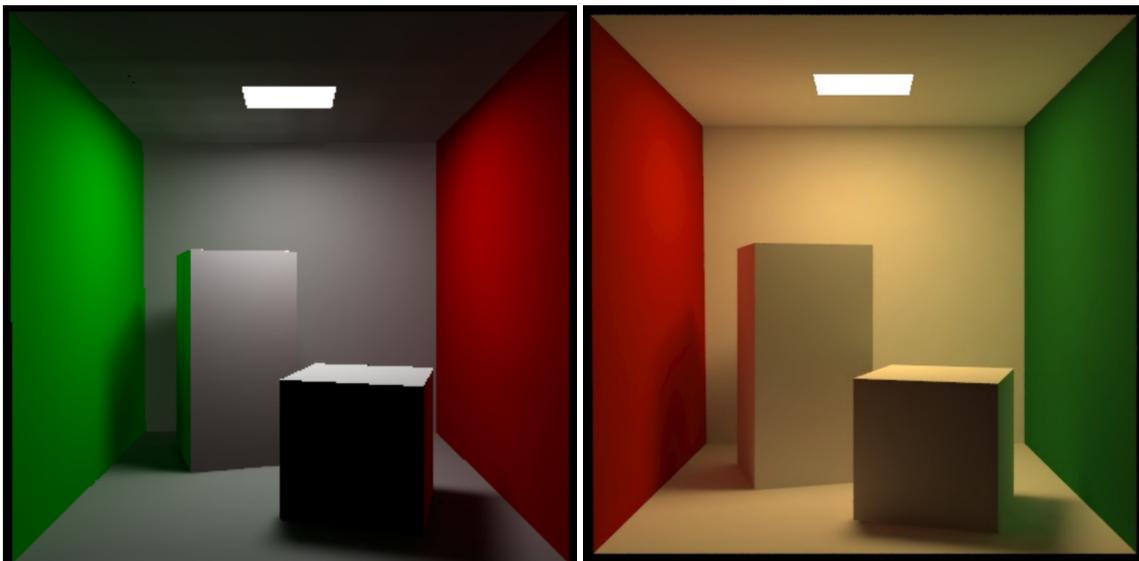


Abbildung 20: (links) NURQV-skaliertes Bild der nachgebauten Cornellbox

Abbildung 21: (rechts) Bild der Cornellbox (siehe[CornellWeb]; die sichtbaren Unterschiede kommen vor allem durch die komplexeren verwendeten Materialeigenschaften zustande.)

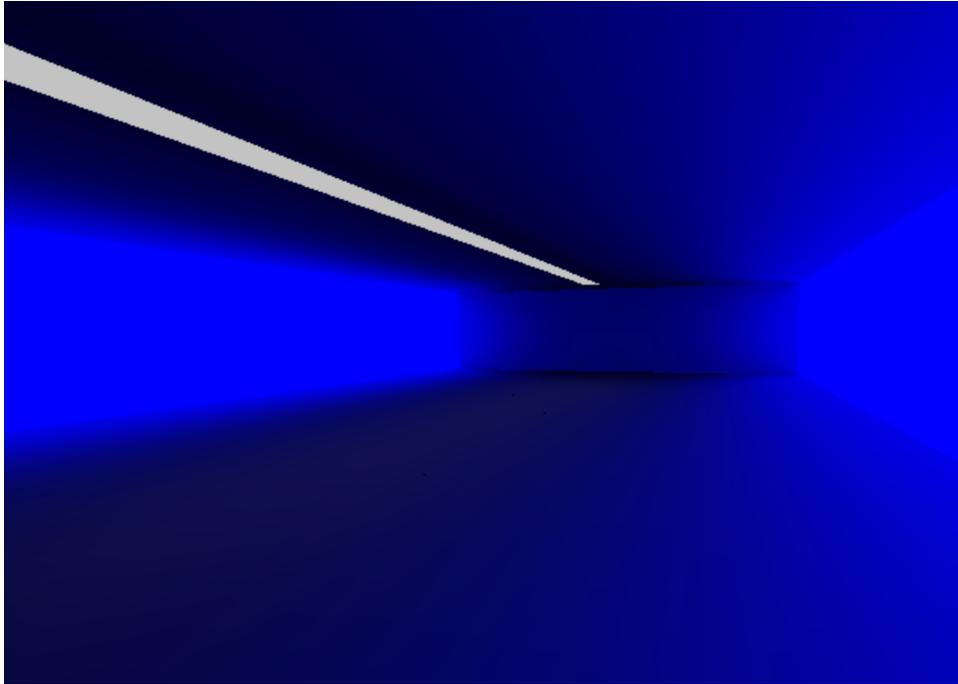


Abbildung 22: Kleiner Raum: 3 Lambert Emitter (Seitenwände (hohe Intensität) und Deckenstrahler (schwache Intensität))

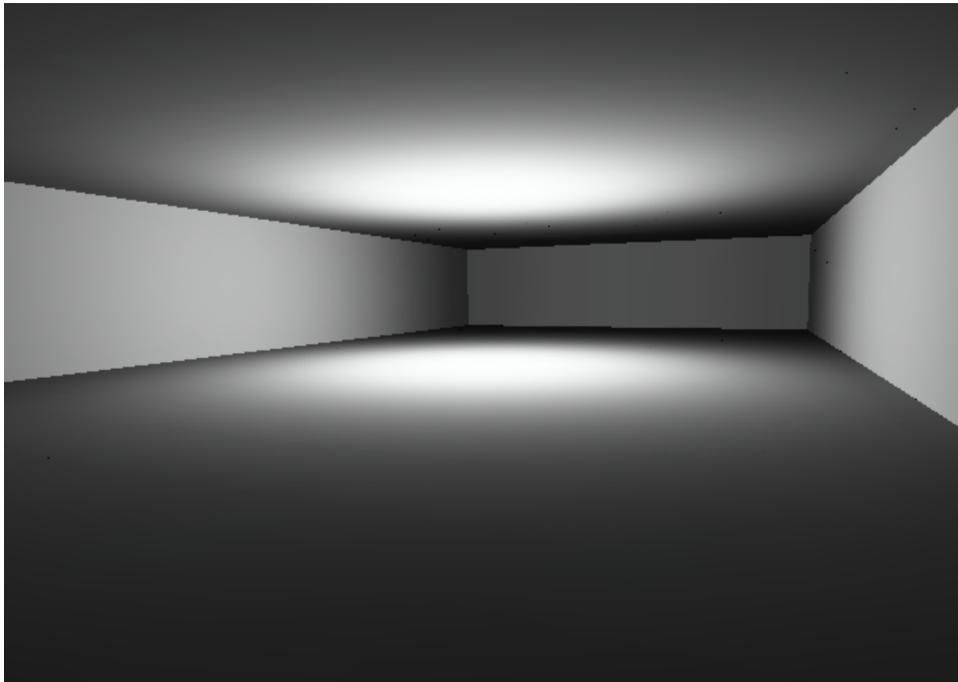


Abbildung 23: Kleiner Raum: 1 Pointlight

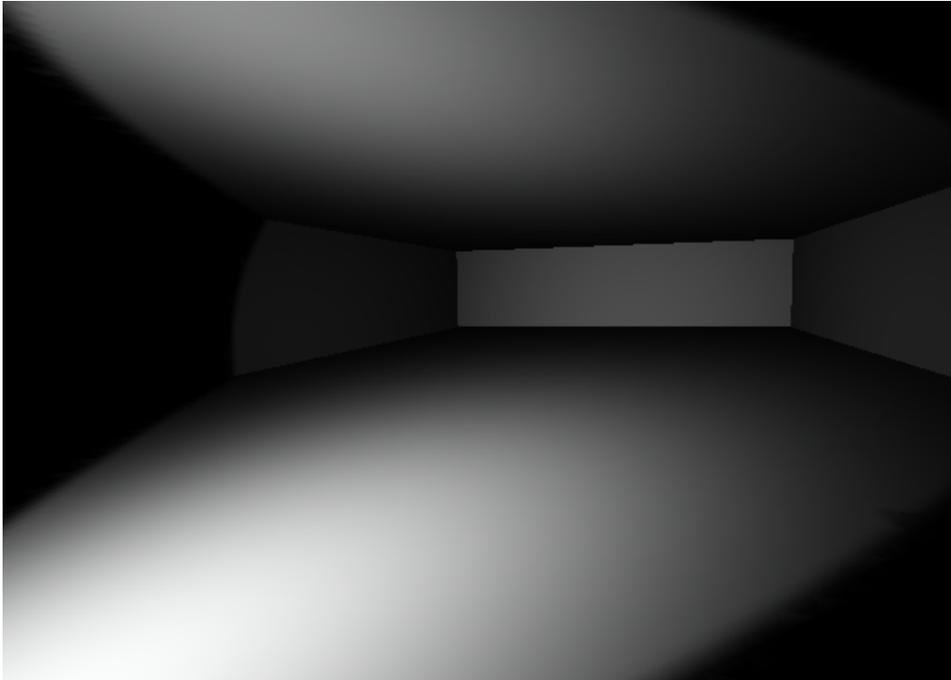


Abbildung 24: Kleiner Raum: 1 Spotlight

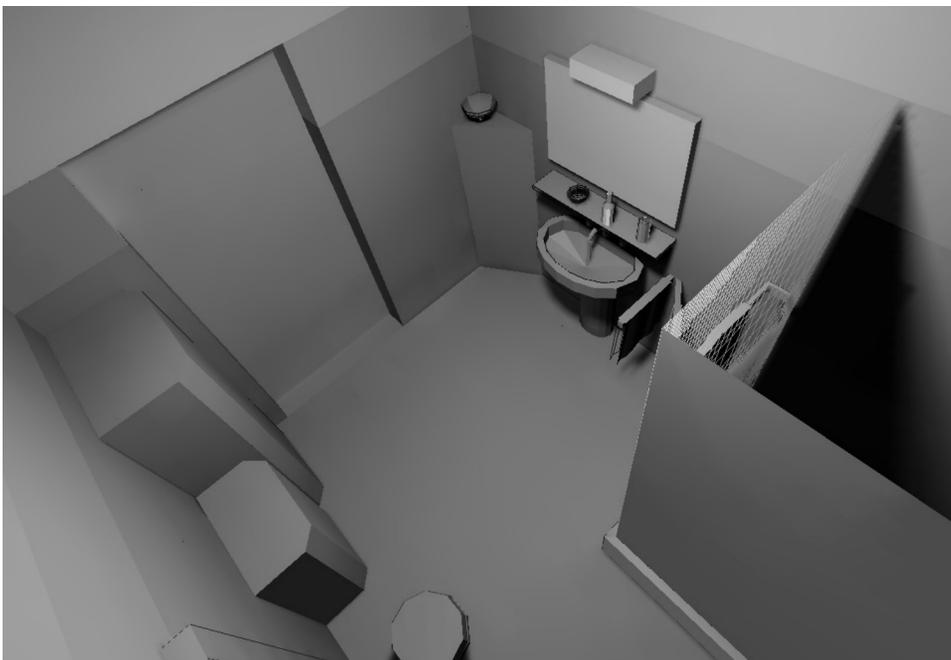


Abbildung 25: Badansicht 1 ohne Farben

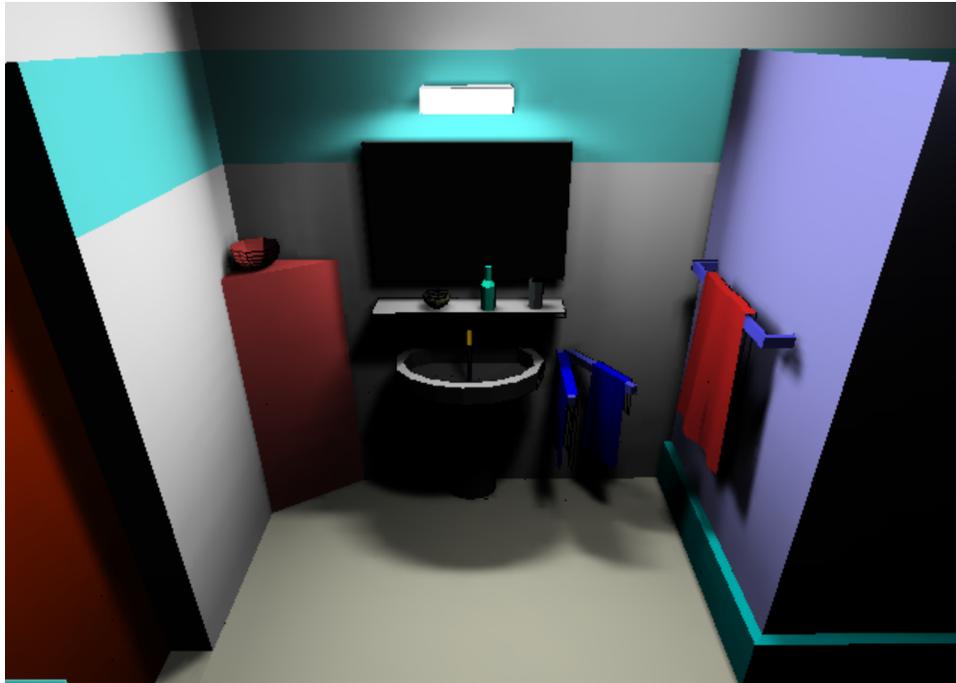


Abbildung 26: Badansicht 2 mit Farben

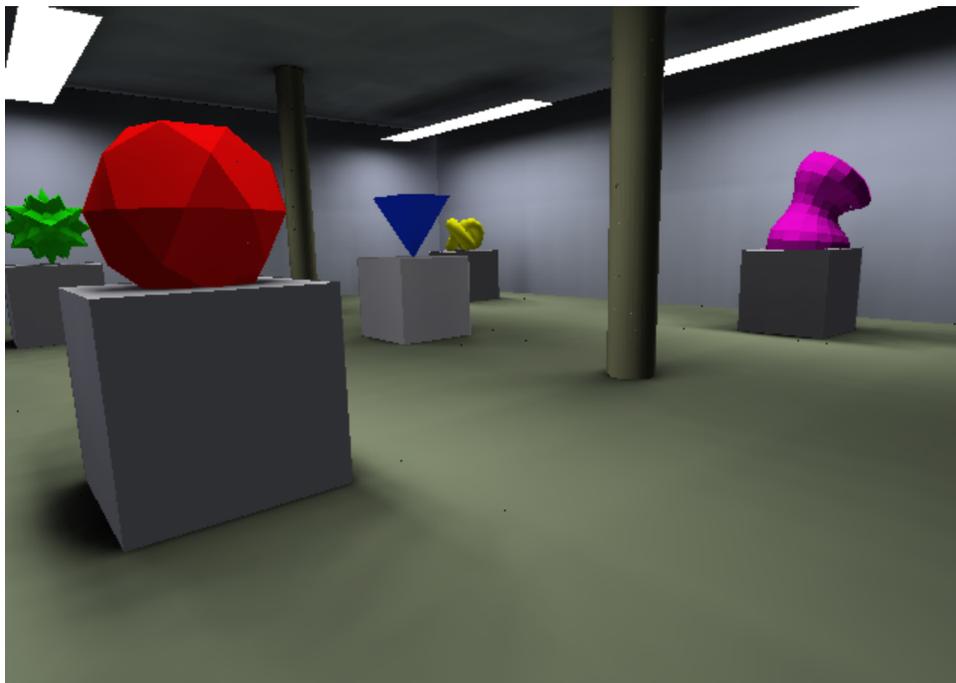


Abbildung 27: Ansicht 1 der Museumsszene mit Objekten

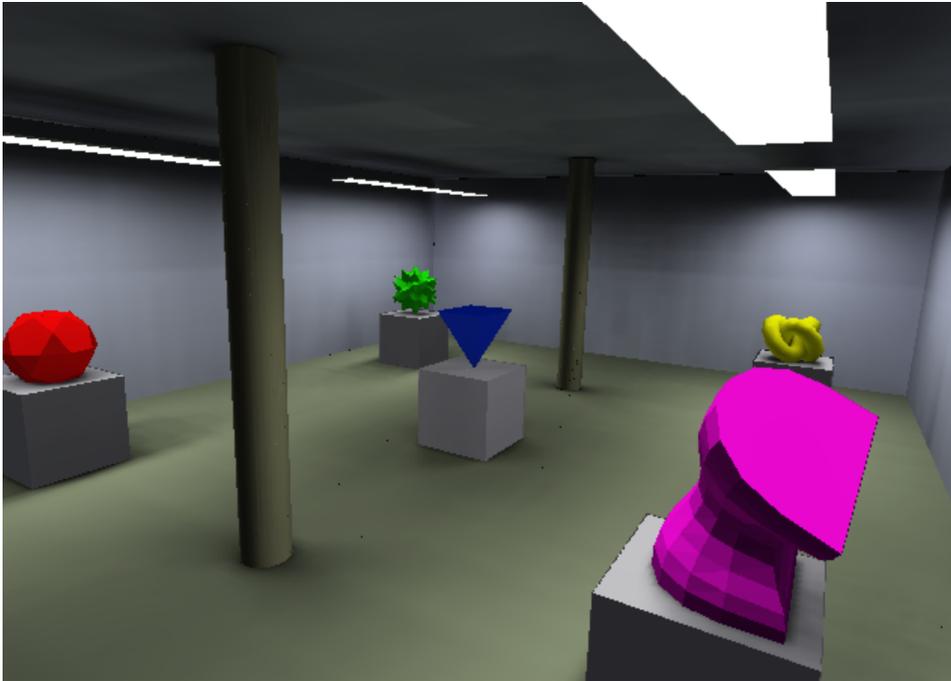


Abbildung 28: Ansicht 2 der Museumsszene mit Objekten

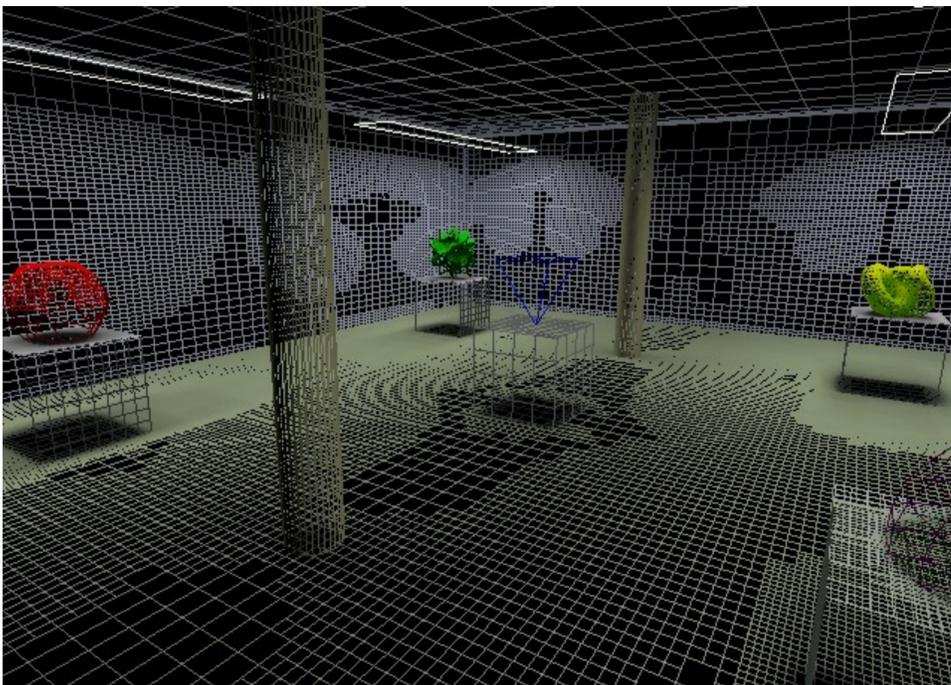


Abbildung 29: Gitternetzansicht der beleuchteten Museumsszene

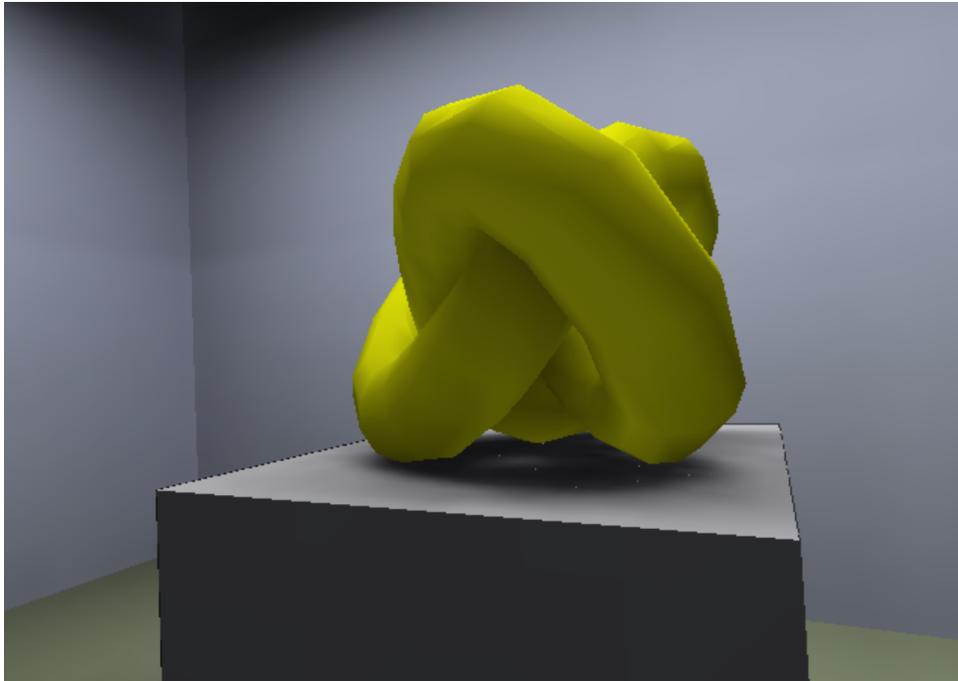


Abbildung 30: Nahansicht 1 eines Objektes

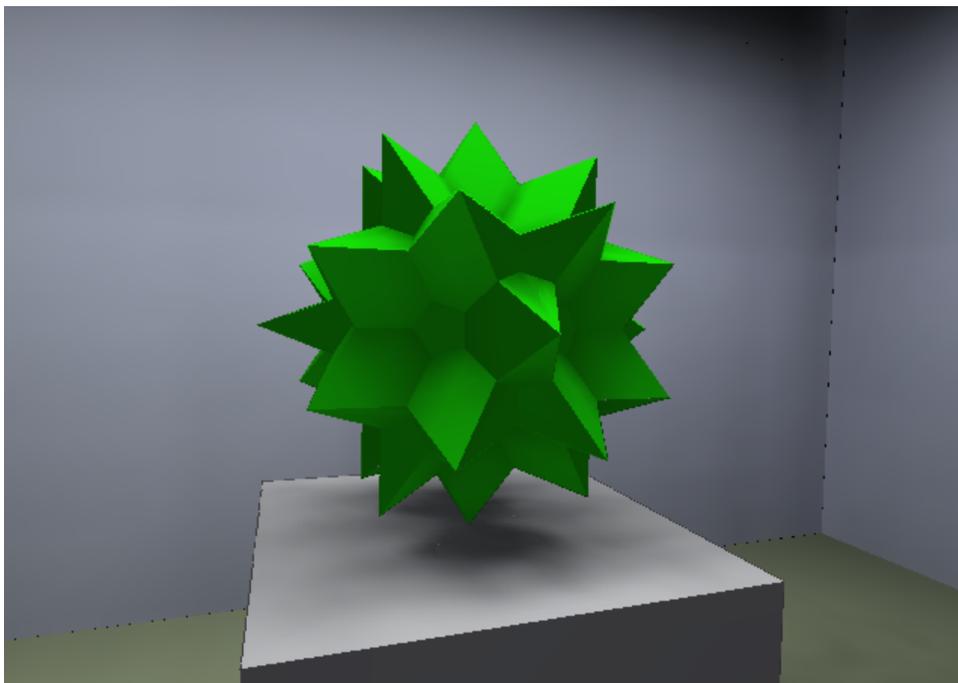


Abbildung 31: Nahansicht 2 eines Objektes



Abbildung 32: Büroszene Ansicht 1: 4 Lambert Emitter (Monitore + Deckenlampen, 2 Spotlichtquellen (Tischleuchten))



Abbildung 33: Büroszene Ansicht 2: 4 Lambert emitter (1 sichtbar, 2 Spotlichtquellen)

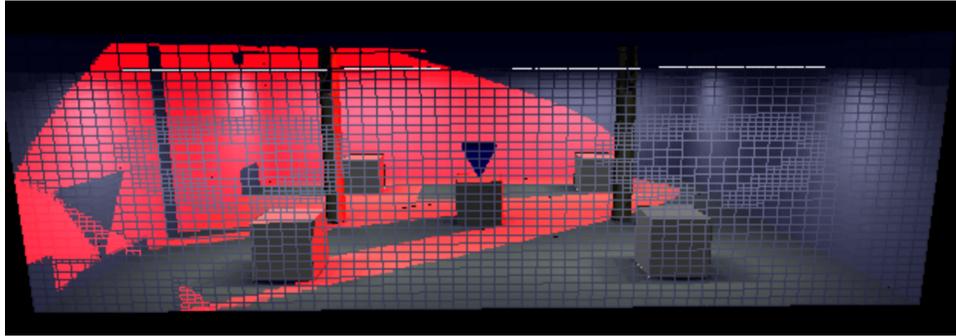


Abbildung 34: Museum mit rotem Spotlight, Schlagschatteneffekte

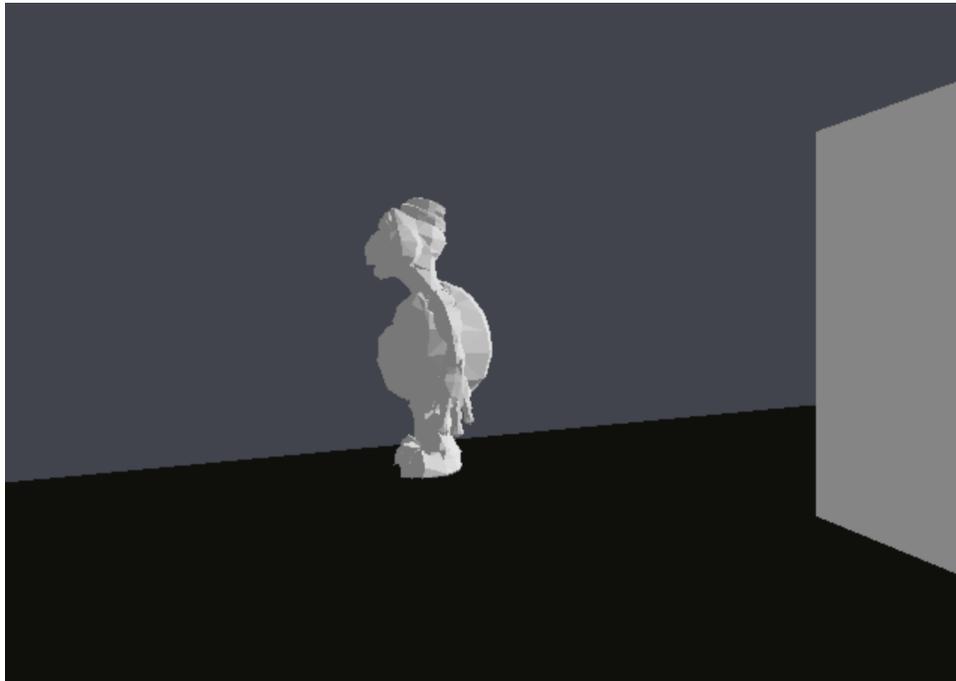


Abbildung 35: von hinten beleuchtetes Turtlemodell (turtle.wrl): 1 Lambert emitter



Abbildung 36: Ausschnitt aus berechneter Museumszene ohne Objekte