H T W M
Hochschule für Technik und Wirtschaft Mittweida

Technikumsplatz 17
09648 Mittweida

Fach: Parallelverarbeitung
Prof.: Prof. U. Schneider

Beleg Parallelverarbeitung

Thema: Parallelisierung eines Raytracers mittels PVM

Jens Meichsner, Isabel Drost, Martin Zennig, Gunther Neubauer

Projekt von:
Jens Meichsner,
Martin Zennig,
Gunter Neubauer,
Isabel Drost

Seminargruppe:
if99wp1

Semester: V

1 Allgemeines

1.1 Gründe für die Wahl des Belegthemas

Das Thema für den Beleg in Parallelverarbeitung war freigestellt. Es sollte sich dabei um eine Aufgabe angemessenen Aufwandes handeln. Laut unserer Einschätzung bot ein Raytracer genügend Möglichkeiten zur Parallelisierung und genügend Optionen, die Parallelisierungstiefe zu verändern. Im einfachsten (von uns gewählten) Fall rechnet jeder Client an einer bestimmten Zeile des Zielbildes, deren Nummer er vom Server mitgeteilt bekommt. Aufgrund der Tatsache, dass sich Jens Meichsner bereits im 2. Semester mit den Thema eines Raytracers beschäftigt hatte, lag es nahe, einen solchen nun für einen Linuxcluster unter PVM zu parallelisieren.

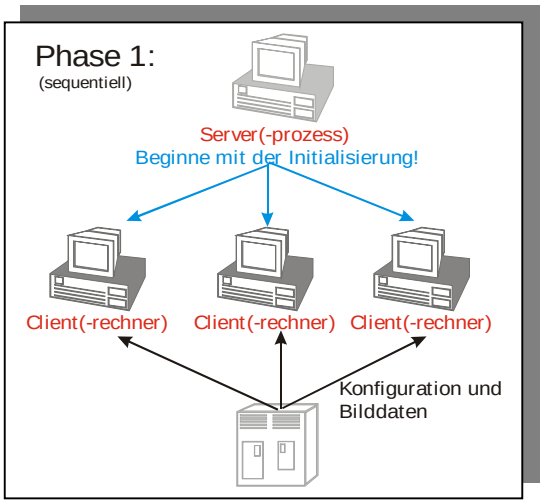
1.2 Notwendige Kriterien

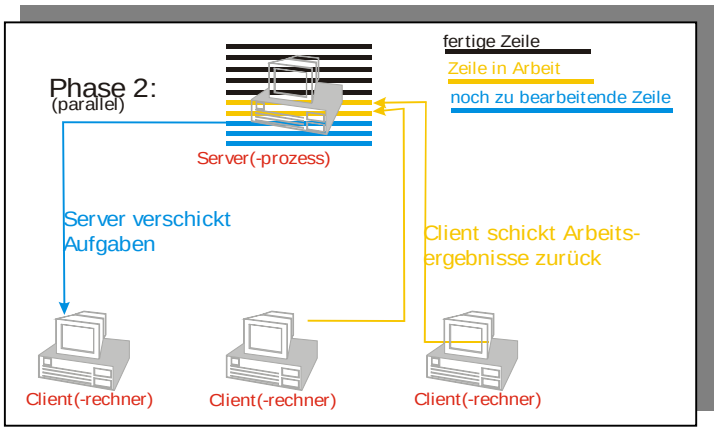
1.2.1 Geschwindigkeit

Aufgrund der Verteilung der Rechenlast auf die Clients soll es möglich werden, die Zeit zum Berechnen eines Bildes möglichst gering zu halten. Bei der Verteilung der Aufgaben ist es notwendig, dass jedem Client die Daten des Bildes bekannt sind (Objekte, Lichteinfall, Reflektionsgrad). Um dies zu garantieren, wurde auf das Netz der Hochschule zurückgegriffen: Die Konfigurationsdateien liegen jeweils im Homebereich dessen, der den Raytracer startet und werden von jedem Client separat eingelesen. Dies verusacht einen verhältnismäßig langen sequentiellen Ablauf in jedem Client. Allerdings sollte mit steigender Komplexität und Größe des Bildes dieser Nachteil kompensiert werden.

1.2.2 Parallelität

Aufgrund der Wahl des Themas fiel es nicht schwer, diverse Möglichkeiten der Parallelisierung zu finden. Als grundlegendes Modell wählten wir eine Client-Server-Architektur. Dabei verwaltet der Server die Resultate der bereits abgearbeiteten Bildzeilen und schickt den Clients neue Aufgaben zu. Wir entschieden uns, jeden Clientprozess eine Bildzeile rechnen zu lassen. Dies verhindert einen zu großen Overhead in der Kommunikation zwischen Server und Client, garantiert aber in unseren Augen dennoch schon bei einer durchschnittlichen Bildgröße von 100x100 Pixel eine genügend große Parallelisierung.





Phase 3 besteht letztendlich aus dem Beenden der Clients und dem Abspeichern des Bildes.

1.2.3 Modularität

Um eine bessere Verwendbarkeit zu garantieren und gleichzeitig die Sourcefiles übersichtlich zu gestalten, soll die Programmierung des Raytracers möglichst modular erfolgen.

1.2.5 Speicherung des Resultatbildes in einem allgemein anerkannten Format

Das durch den Raytracer berechnete Bild soll in einem Bildformat abgespeichert werden, dass von den üblichen Bildbearbeitungsprogrammen angezeigt werden kann (z.B. als einfache Bitmap). Dabei soll keine Komprimierung erfolgen (somit scheiden Formate wie gif oder jpeg aus).

1.2.6 Phong-Shading, Reflexionen und Schattenwurf

Um die Leistungsfähigkeit des Raytracers zu demonstrieren und zu zeigen, das der verwendete Algorithmus die Grundaufgaben jedes modernen Raytracers beherrscht, kann auf Dinge wie korrekte Reflexionen und Schattenwurf nicht verzichtet werden.

1.3 Wunschkriterien

1.3.1 Darstellung des Bildes

Ein Wuschkriterium wäre, dass das mit dem Raytracer errechnete Bild anschließend auf dem Serverrechner dargestellt wird.

1.3.2 Interaktive Konfiguration des Raytracers

Der Raytracer wird im einfachsten Falle durch eine Textdatei konfiguriert. Denkbar wäre, diese Konfiguration interaktiv zu gestalten und vor dem Beginn des Raytracens dem User zu überlassen, welche Features des Programms er nutzen möchte.

1.4 Abgrenzungskriterien

1.4.1 Maximal mögliche Paralellisierung

Aufgrund des Zeitrahmens, der uns vorgegeben wurde, ist es natürlich nicht möglich, eine maximale Parallelisierung zu erreichen. Schon aufgrund der Datenabhängigkeiten der Vorgänge beim Berechnen des Bildes gestaltet sich dieses Problem zu schwierig, um es in einem einfachen Praktikumsbeleg zu lösen. Weiterhin wäre dies in unseren Augen auch nur in begrenztem Maße sinnvoll, da mit steigender Parallelität auch der Kommunikationsaufwand zwischen Client und Server steigt. Weiterhin stehen uns in den Rechnerpools des Haus8 meist maximal 16 Rechner zur Verfügung - somit sollte eine Parallelisierung auf Zeilenebene ausreichend sein.

1.4.2 Interaktive Veränderung der Szenendaten

In Programmen, vergleichbar 3DStudioMax oder Maja ist es möglich, die Bilder in einem sogenannten Builder selbst zusammenzustellen und zu entwickeln. Dies erfordert sehr viel Programmieraufwand und ist innerhalb eines halben Jahres nicht realisierbar. Ein weiterer Grund, die Implementierung eines solchen Editors außen vor zu lassen besteht darin, dass die Erstellung der Bilddaten sich nicht parallelisieren läßt und sich somit auch stark von der eigentlichen Aufgabe, ein parallelisiertes Programm zu entwickeln, entfernt.
Deshalb haben wir uns entschieden, die Exportfunktion von 3DStudioMax zu nutzen: Die Szene wird in diesem Programm zusammengestellt und mithilfe eines Skriptes exportiert. Schließlich kann unser Raytracer die durch das Skript erstellte Datei einlesen und mit dem Berechnen des fertigen Bildes beginnen.

1.5 Copyright

Der Raytracer unterliegt der GNU General Public License[1] Somit ist die Verwendung sowohl für private als auch für kommerzielle Zwecke absolut kostenlos. Der Quellcode darf nach Belieben ergänzt und erweitert und zu Teilen oder komplett in anderen Projekten weiterverwendet werden. Jede Verwendung der Quellen des Raytracers verpflichtet den Programmierer mittels GNU GPL dazu, das Resultat ebenfalls unter diese Lizenz zu stellen.

1.6 Knappe Definition der Aufgabe des Raytracers

Das Programm soll eine in einer Textdatei gespeicherte Szene entsprechend den vom User vorgegebenen Konfigurationseinstellungen tracen und das Resultat in einer eigenen Bilddatei abspeichern.

[1] http://www.fsf.org/licenses/licenses.html#GPL

2 Bedienungsanleitung für den Nutzer

2.1 Der Inhalt der CD

`/`	... Readme, die erste Installationshinweise enthält
`/bin`	... compilierte Versionen des Programmes
`/bin/lin/single`	... Programm für einen einzelnen Linuxrechner
`/bin/lin/parallel`	... Programm für den PVM-Linuxcluster
`/bin/win`	... Programm für einen einzelnen Windowsrechner (SMP-fähig)
`/doc`	... Dokumentation (diese Datei) und Readme, Diplomarbeit zum Thema Oct-Tree
`/src/lin/single`	... Quellcode des Programmes für einen einzelnen Linuxrechner
`/src/lin/parallel`	... Quellcode des Programmes für einen PVM – Cluster
`/src/win`	... Quellcode für Windows
`/pvm`	... bei der Programmentwicklung verwendete PVM-Version
`/helpers`	... Script zum Exportieren der Daten aus 3DStudioMax, GMAX

2.2 Systemvoraussetzungen

Auf der Workstation, auf der der Raytracer arbeiten soll, muß ebenfalls PVM[2] installiert sein. Installationshinweise zu diesem Programm sind der Dokumentation und der Readme, die beim Download diesem Programm beiliegt, zu entnehmen. Der Version des Raytracers liegt die PVMversion, auf der er getestet wurde, bei.

2.3 Das Netzwerk, das Clients und Server verbindet

Der Raytracer wurde mittels PVM dafür konzipiert, in einem Cluster von (Linux-) Rechnern zu laufen - erst dann wird sich zeigen, dass Bilder schneller berechnet werden, als auf vergleichbaren herkömmlichen Ein- oder Zweiprozessor-maschinen (siehe dazu auch die Vergleichstests weiter unten in der Dokumentation). Dazu müssen gewisse Voraussetzungen gegeben sein:
Das Prinzip von PVM beruht darauf, bestimmte Rechner die über ein Netzwerk verbunden sind, zu einem Cluster zu verbinden. Dazu wird PVM gestartete und mittels add hosts <hostip> die gewünschten Hosts dem Verbund hinzugefügt. Der Raytracer erwartet nun, auf jedem als Clientrechner agierenden Host seine Daten an den gewohnten Stellen vorzufinden. Das heißt konkret, dass sich sowohl Konfigurationsdatei (raytracer.cfg) als auch die Datei mit den Bilddaten (ROOM.txt) in dem Verzeichnis befinden müssen, von dem aus er gestartet wird (üblicherweise $PVM_ROOT/bin/$ARCH).
Das Netzwerk der Hochschule, an der der Raytracer entwickelt wurde, ist dazu besonders gut geeignet, da hier an jeder Workstation der gleiche Homebereich des jeweiligen Users zur Verfügung steht. Die Nutzer erreichen also die Daten auf den Homebereichen selbst über Netzwerk. Nachteilig dabei ist lediglich, dass bei hoher Netzlast hier der sequentielle Anteil durch das Einlesen der Datei bei jedem Client nochmals erhöht wird. Dem steht allerdings gegenüber, dass bei der Alternativlösung, jedem Client die Daten vom Server zu schicken, die Netzbelastung eine ebensolche Beeinträchtigung hervorrufen würde.

2.4 Die Installation des Raytracers

2.3.1 Reine Installation ohne erneutes Kompilieren (nur bei Standardlinux)

1. Setzen Sie die Variable $PVM_ROOT auf das Wurzelverzeichnis, in dass Sie PVM installiert haben und setzen Sie $ARCH ebenfalls, wie von PVM vorgegeben
2. Starten Sie das Shellscript install im bin-Verzeichnis des Raytracers. Es wird alle relevanten Dateien an die korrekten Stellen kopieren. 2.3.2 Compilieren des Raytracers und anschliesendes Installieren

Starten Sie das Shellscript Compile. Es wird alles Weitere erledigen. Alternativ dazu können Sie die im Verzeichnis /src/lin/parallel vorhandenen Makefiles anpassen und anschliesend ein aimk all durchführen. Die dabei erstellten Objectfiles können Sie mit aimk clean löschen.

[2] http://www.epm.ornl.gov/pvm

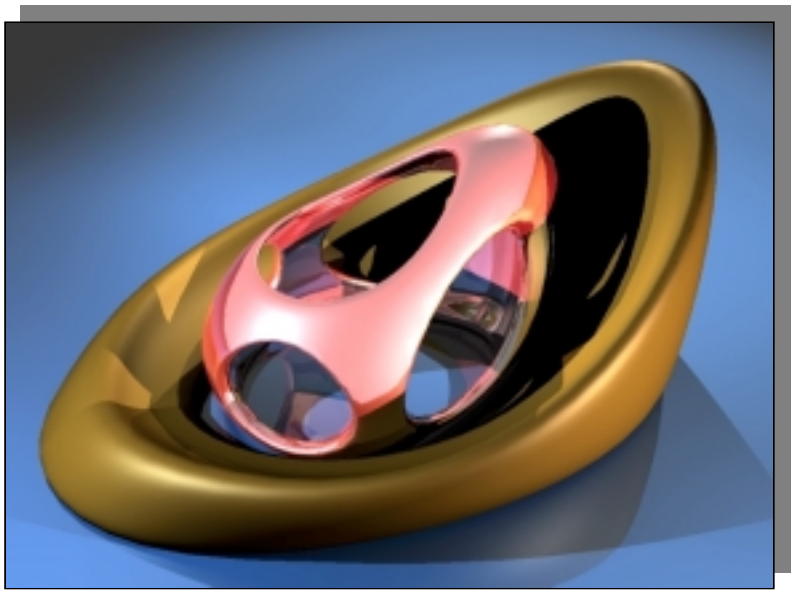
2.4 Start des Raytracers

Insofern alle Dateien an den richtigen Stellen liegen, sollte es jetzt kein Problem sein, den Raytracer ein erstes Mal zu starten. Dazu ist einfach eine Konsole, dann pvm zu starten, die gewünschten Rechner dem Cluster hinzuzufügen. Anschließend kann man mit

```
spawn -> server
```

den Raytracer starten. Der Name der Datei, die letztendlich das berechnete Bild enthalten soll, wird in der Datei raytracer.cfg spezifiziert. Diese muß sich zusammen mit dem Server, dem Client und der ROOM.txt in dem Verzeichnis befinden, in dem PVM nach den auszuführenden Programmen sucht. Die ROOM.txt beinhaltet die Daten, des Bildes, das berechnet werden soll.

Beispiel eines getraceten Bildes:



2.5 Aufbau der Konfigurationsdatei

Die Konfigurationsdatei des Raytracers ist eine einfache Textdatei, welche mit jedem gewöhnlichem Textverarbeitungsprogramm editiert werden kann. Sie steuert die Arbeitsweise des implementierten Raytracers, berücksichtigt aber keine Einstellungen zur Funktionsweise von PVM.

Die Datei enthält folgende globale Einstellungen:

- Schattenwurf an/aus
- Reflexionen an/aus
- Name der Datendatei
- Name des Ausgabebildes
- Höhe und Breite des Ausgabebildes
- Hintergrundfarbe
- Position und Ausrichtung der Kamera

Der Parser, welcher die Datei einliest unterscheidet zwischen zwei Arten von Befehlen:
Befehle zum Starten von Funktionen und Befehlen zum Laden von Variablen

Funktionen und Variablen stehen jeweils am Anfang einer Zeile. Funktionen beginnen mit einem beliebigen Zeichen, während Variablen mit einem ‚$' beginnen.

Sollte der Parser einen Befehl oder Variable erkannt haben, nutzt er folgende Wörter/Zahlen als Argumente. Als Trennzeichen dient der Tabulator oder die Leertaste. Der Parser achtet auf die Groß- und Kleinschreibung.

Befehl/ Variable	Wirkung
REM	leitet einen einzeiligen Kommentar ein
$IMAGE_NAME	bestimmt den Namen der Ausgabedatei
$IMAGE_WIDTH, $IMAGE_HEIGHT	bestimmt die Größe der Ausgabedatei
$INPUT_DATA	bestimmt den Namen der Datei, die die Bilddaten enthält
$SHADOWS	mit angefügter Option 1 ... Schattenwurf einschalten 0 ... Schattenwurf ausschalten
$REFLECTIONS	1 ... Reflexionsberechnung einschalten 0 ... Reflexionsberechnung ausschalten
$COLOR_BG_R, $COLOR_BG_G, $COLOR_BG_B	Rot-, Grün- und Blauanteil der Hintergrundfarbe (0.0 – 1.0)
$CAM_POS_X, $CAM_POS_Y, $CAM_POS_Z	Kammeraposition
$CAM_POS_ANGLE_X, $CAM_POS_ANGLE_Z	Drehung der Kammera bezüglich der einzelnen Achsen
$CAM_POS_ZOOM	eingestellter Zoom der Kammera (eingestellte zusätzliche Vergrößerung des Bildes)
QUIT	schließt die Konfiguration ab, muß nicht angegeben werden, ist aber aus Sicherheitsgründen empfehlenswert

2.6 Die Datendatei

Zum Einlesen der Datendatei dient der selbe Parser wie zur Konfigurationsdatei, nur das er diesmal andere Funktionen kennt. Zur Erstellung der Datendatei wird empfohlen das Export-Script für 3D Studio Max zu nutzen.

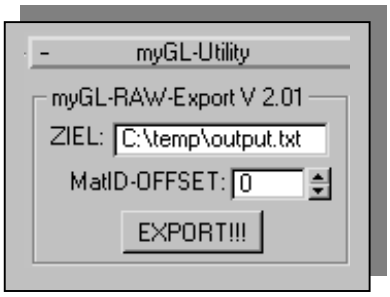
2.6.1 Funktionen:

Funktion	Wirkung
REM	leitet einen einzeiligen Kommentar ein
QUIT	sofortige Beendigung der Bearbeitung der Datei (Nachfolgende Funktionen werden ignoriert, muß nicht angegeben werden, wird aus Sicherheitsgründen allerdings empfohlen.)
_OBJ <Anzahl aller Eckpunkte> <Anzahl aller Dreiecke> <Anzahl aller Shader> <Anzahl aller Lichtquellen>	Anweisung zur Speichervorbereitung. _OBJ ist optional, es darf kein Parameter vergessen werden.
_SHD <Ambient-red> <Ambient-green> <Ambient-blue> <Diffuse-red> <Diffuse-green> <Diffuse-blue> <Specular-red> <Specular-green> <Specular-blue> <Shininess> <Reflection-Amount>	Legt einen Shader an. In der Szene muss sich mindestens ein Shader befinden. Es darf kein Parameter weggelassen werden.
_LIGHT <X_Pos> <Y_Pos> <Z_Pos> <Color_red> <Color_green> <Color_blue> <Amount> <ShadowCasting>	Legt eine Lichtquelle an. In der Szene muss sich mindestens eine Lichtquelle befinden. Die Parameter sind obligat.
_PT <X_Pos> <Y_Pos> <Z_Pos>	Legt einen Eckpunkt an.
_TR <Point_A> <Point_B> <Point_C> <Shader>	Legt ein Dreieck an.

2.6.2 Das Export-Script

Das Exportscript wurde in der 3D Studio Max eigenen Script-Sprache MAXScript geschrieben und braucht zur Ausführung 3D Studio Max (min. Version 2.0) oder GMAX.

Kopieren Sie die Script-Datei das Unterverzeichnis „Scripts“ von 3D Studio Max und starten Sie es über die Werkzeugpalette „Dienstprogramme“.



Funktionsweise:
Markieren Sie in 3D Studio Max alle zu exportierenden Objekte. Wählen Sie danach den Pfad und Name der Ausgabedatei und drücken Sie zum Schluss den Button „EXPORT!!!“.

Da 3D Studio Max intern alle Datenstrukturen beginnend mit „1“ und nicht mit „0“ durchnummeriert, wurde ein weiteres Feld eingeführt. Tragen Sie hier „-1“ ein um alle Indizes um eins zu verringern.

3 Grundprinzipien des Raytracers

3.1 Methoden, dreidimensionale Welten mit dem Computer zu berechnen

Um dreidimensionale Objekte im Computer nachzubilden bedient man sich der Mathematik, genauer gesagt der Geometrie. Hierbei zerlegt man das gewünschte Objekt in eine größere Anzahl von Dreiecken, welche dann vom Computer auf dem Bildschirm dargestellt werden müssen.

In den Jahren der Entwicklung grafischer Systeme und Algorithmen haben sich zwei Verfahren zur Darstellung von Dreiecken herausentwickelt.

3.1.1 Scanlining

Der Reihe nach werden alle Dreiecke in einen Puffer gezeichnet, welcher sich nicht nur die Farbinformationen sondern auch die Tiefeninformationen aller Pixel merkt. Das Dreieck wird Linienweise von oben nach unten abgetastet (darum Scanline). Bei jedem zu zeichnenden Pixel wird geprüft, ob sich nicht schon ein Pixel mit geringerer Tiefe an dieser Stelle im Puffer befindet. Sollte dieses zutreffen wird das Pixel einfach nicht gezeichnet.
Das Scanliningverfahren ist sehr schnell und wird von den heutigen Grafikkarten mit 3D-Beschleunigung genutzt. Leider kann man mit ihm Reflexionen und Schattenwurf nur vortäuschen oder nur sehr ungenau durch Annäherung nachbilden lassen.

3.1.2 Raytracing

Dieser Algorithmus verfolgt die Strahlen vom Auge des Betrachters durch die abzubildende virtuelle Welt. Dabei wird jede Fläche geprüft, ob sie mit dem Strahl zusammenstößt. Es wird die dem Auge am nächsten liegende Fläche behalten und von dieser zum Schluss die Farbe bestimmt.
Das Raytracingverfahren ist sehr langsam und daher vom Geschwindigkeitsaspekt (vom jetzigen Standpunkt) nicht mit dem Scanliningverfahren vergleichbar. Dafür sind hier physikalisch korrekte Reflexionen und Schattenwurf sehr einfach realisierbar.

Heutige professionelle Grafikprogramme wie Maya, SoftImage oder 3D Studio Max verknüpfen beide Verfahren miteinander um das Maximum an Geschwindigkeit zu erreichen und dennoch Effekte zu bieten, welche nur durch Raytracing erreichbar sind.

3.2 Der entwickelte Raytracer

Der zum Einsatz kommende Raytracer ist eine komplette Eigenentwicklung. Als Ausgangsdaten nutzt dieser Geometriedaten, welche vorher mit 3D Studio Max erstellt und exportiert wurden.

3.2.1 Eigenschaften des Raytracers

- Reflexionen und Schattenwurf
- Farbbestimmung durch Phong-Shading
- Keine maximale Anzahl von Flächen vorgegeben (nur vom Speicher begrenzt)
- Keine maximale Anzahl von Lichtquellen vorgegeben
- Beschleunigung durch Oct-Tree

3.2.2 Funktionsweise des Raytracers

3.2.2.1 Vorbereitung und Ladevorgang:

Um den Raytracer in den Ausgangszustand zu bekommen, wird eine Konfigurationsdatei und eine Datendatei benötigt. Die Konfigurationsdatei heißt „raytracer.cfg“ und enthält globale Einstellungen (Beispieldatei siehe Anhang). In ihr ist auch der Name der Datendatei enthalten, welche danach automatisch geladen wird. Diese zweite Datei enthält nur Geometriedaten, welche aus 3D Studio Max per Max-Script exportiert wurden.

Nachdem beide Dateien geladen wurden, kommt es zu einer ersten Vorverarbeitung um beim späterem Raytracing Rechenzeit zu sparen:

- Dreiecke werden gebildet (aus drei Eckpunkten [Vertex])
- Jedes Dreieck erhält einen normierter Normalenvektor.
- Shading-Vektoren der Eckpunkte werden gebildet (werden für das Phong-Shading benötigt)
- Aus allen Dreiecken wird der Oct-Tree gebildet.

3.2.2.2 Raytracing:

Da der Raytracer die Formate des Endbildes sowie die Position der virtuellen Kamera (Auge des Betrachters) kennt, muss man ihm nur die Koordinaten des zu berechneten Pixels übergeben.

- Er bestimmt den Sehstrahl (aus den Kameradaten) in den Raum.
- Oct-Tree bestimmt alle Flächen, welche getestet werden müssen.
- Alle errechneten Flächen werden auf Zusammenstoß mit dem Sehstrahl getestet (es wird die mit der geringsten Entfernung zur Kamera behalten).
- Es wird ein Sehstrahl zu jeder Lichtquelle generiert, sollte für diese Lichtquelle Schattenwurf aktiviert sein, werden über den Oct-Tree sämtliche Flächen zur Schattenwurfbestimmung generiert. Auch hier wird jede Fläche getestet, ob sich der Lichtstrahl mit ihr schneidet, die erste, bei der dies der Fall ist, wird für den Schattenwurf verwendet.
- Wurde keine Fläche geschnitten, so wurde durch das Licht auch kein Schatten erzeugt, die Lichtintensität muss für diese Lichtquelle neu berechnet werden.
- Wurde für das Material dieser Fläche Reflexion aktiviert, wird der Reflexionsstrahl bestimmt. Die Berechnung wird beim zweiten Punkt fortgesetzt, wobei jetzt ein neuer Sehstrahl (der dem Reflexionsstrahl entspricht), verwendet wird.
- Lichtanteil und Reflexionsanteil werden zusammengerechnet. Die Farbe wird zurückgegeben...

3.2.3 Besonderheiten und Bemerkungen

3.2.3.1 Rekursionsabbruch bei den Reflexionen

Um zu verhindern, dass ein Strahl bis in die Unendlichkeit reflektiert wird, kommt es bei jeder neuen Reflexion zu einem Test, ob der neue Reflexionsanteil an der Gesamtfarbe des Resultates größer als 10% ist. Sollte dieser Wert unterschritten werden, dann wird abgebrochen.
Dieses Verfahren setzt voraus, das es kein Material gibt, welches sich zu 100% aus der Farbe des reflektierten Objektes zusammensetzt.

3.2.3.2 Beschleunigung durch den Oct-Tree

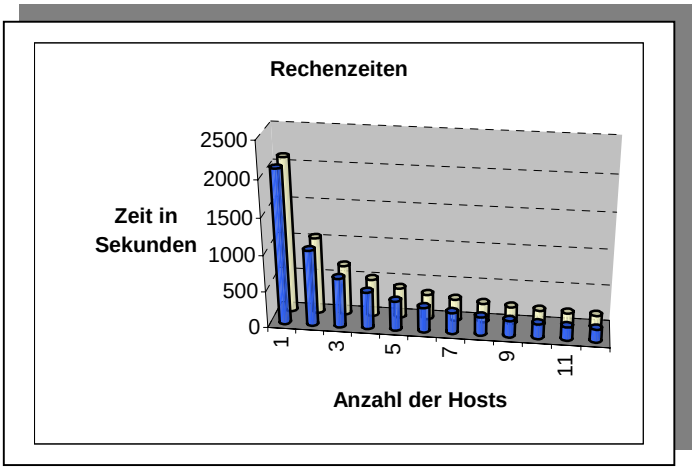
Da das Raytracing sehr langsam ist und schon bei sehr wenigen Flächen alle Rechenzeitrekorde bricht (95% bis 99% der Rechenzeit wird verwendet um die Fläche zu finden, welche den Sehstrahl als erste durchstößt), hat man mehrere Verfahren entwickelt um die Berechnung zu beschleunigen.
Die schnellsten Verfahren zielen darauf ab, die Anzahl der Flächen zu minimieren, welche man testen muss. Dazu unterteilt man die virtuelle Welt und setzt sie in einem Tree zusammen (meist BSP-Trees, QUAD-Trees oder OCT-Trees). Nun testet man am Anfang den Sehstrahl mit dem Tree und dieser generiert nur die Flächen, welche tatsächlich für einen Kontakt mit dem Sehstrahl in Frage kommen.
In diesem Projekt wurde zur Beschleunigung ein Oct-Tree implementiert. Er bietet zum einen eine ausreichend gute Beschleunigung und zum anderen hält sich der Programmieraufwand in Grenzen. Die genaue Funktionsweise eines solchen Baumes wurde der Diplomarbeit von Gordon Mueller (`97) entnommen, die sich als PDF-Datei mit auf der Projekt-CD befindet.

Anmerkung: Mittlerweile lässt man mehrere unterschiedlich aufgelöste oder strukturierte Trees parallel gegeneinander laufen und es werden nur die Flächen des Baumes genutzt, welcher weniger generiert hat. Bsp.: Raytracer in 3D Studio Max nutzt optional ein oder zwei Trees. Auch durch Anwendung aller weiteren Möglichkeiten zur Beschleunigung, kommt man mit dem Raytracer nicht an die Geschwindigkeit des Scanlinings heran.

3.2.3.3 Entwicklung

Der Raytracer wurde in MS Visual C/C++ 6.0 für Windows entwickelt. Die Implementierung erfolgte komplett in C, zum einen da C++ bei der Größe des Projektes einen zu großen Overhead darstellen würde und zu anderen, um einen kleinen Geschwindigkeitsvorteil zu erzielen (nach eigenen Messungen bis zu 15% schneller gegenüber C++).
Nach Fertigstellung erfolgte die Portierung nach Linux, was minimale Änderungen am Quellcode (besonders an Compileranweisungen) mitbrachte. Da der Raytracer vollkommen modular aufgebaut ist, konnte er zum Schluss ohne Änderungen in PVM integriert werden.

3.2.3.4 Laufzeitvergleiche



Zur Ermittlung der Rechenzeiten wurden jeweils ein bis zwölf Rechner mit identischer Ausstattung verwendet (PII–350 mit 128MB RAM). PVM erstellte jeweils 16 Prozesse, welche dann auf alle verfügbaren Rechner verteilt wurden.

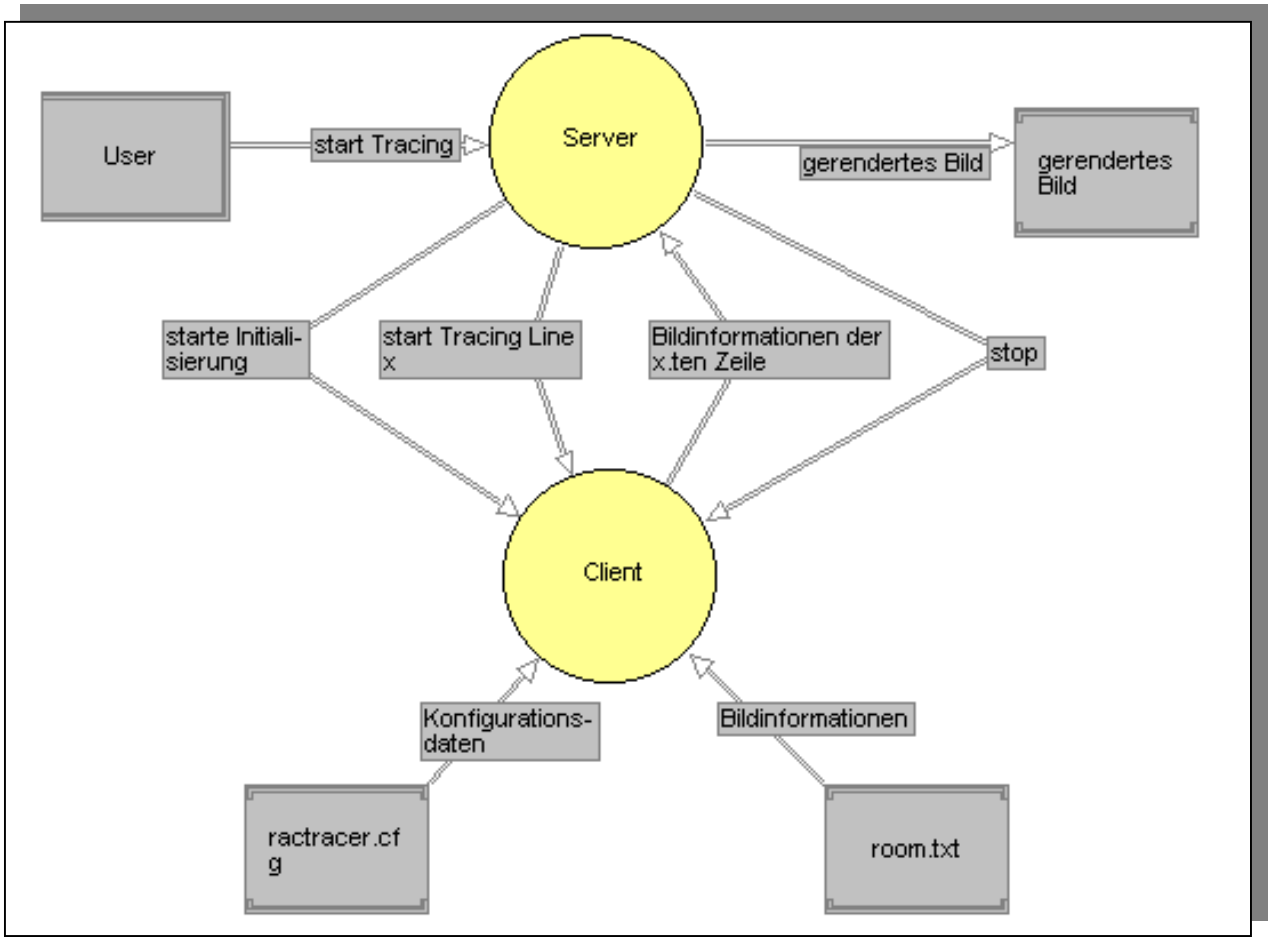
4 Quellcodedokumentation

4.1 Allgemeines

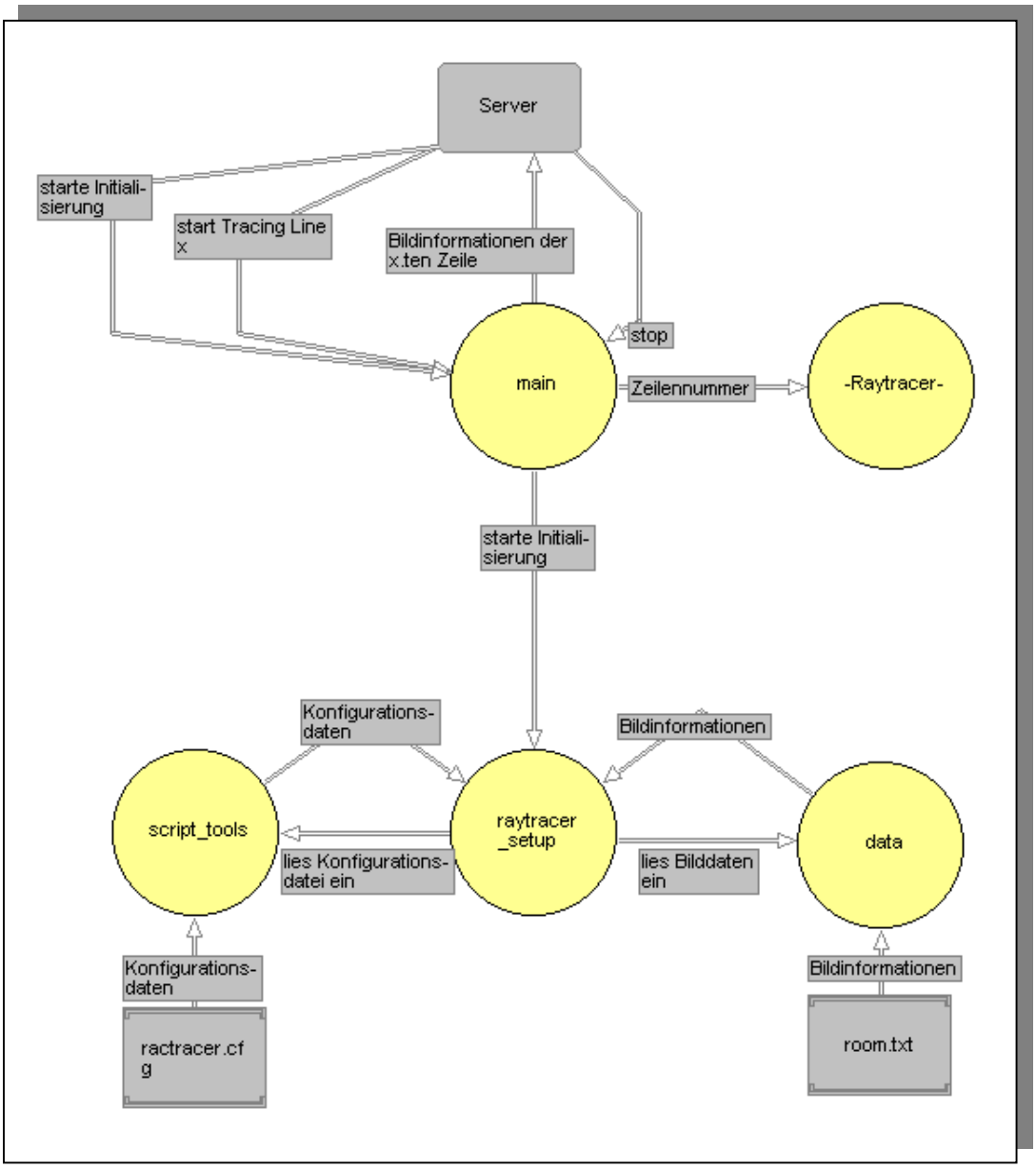
In dieser Quellcodedokumentation soll nur ein Überblick über die Arbeitsweise des Programmes gegeben werden. Die genaue Dokumentation erfolgt durch ausreichende Kommentare im Quelltext selbst. Unterstützt wird die Erklärung durch graphische Darstellungen der Beziehungen der Funktionen bzw. ganzer Module untereinander und ihre Zusammenarbeit. Es wird dabei lediglich auf entscheidende Funktione eingegangen.

4.2 Zusammenarbeit von Server und Client

Der Raytracer arbeitet nach einem Client-Server-Prinzip. Die Kommunikation zwischen Server und Clients erfolgt über Messages. Dabei wird auf Messageformen zurückgegriffen, die von PVM zur Verfügung gestellt werden.
Der Server beginnt seine Arbeit, indem er zuerst die gewünschte Anzahl Clients startet. Danach teilt er ihnen mit, wie die Konfigurationsdatei heißt und wo sie sich befindet. Sind alle Clients mit dem Einlesen ihrer Konfiguration fertig, beginnt der Server mit der Verteilung der Aufgaben. Nacheinander bekommen alle Clientprozesse eine Zeilennummer benannt. Die entsprechende Zeile des Bildes wird von den Clients gerendert und die Bilddaten in Form eines PVM-gepackten Bytearrays an den Server zurückgegeben. Dieser kopiert die Zeile an die richtige Stelle im Bild und verschickt an den frei gewordenen Client eine neue Zeile zum Berechnen, insofern das Bild noch nicht fertig berechnet wurde.
Wurden alle Zeilen des Bildes berechnet, schickt der Server den Clients ein Signal, dass sie dazu bringt, sich zu beenden, speichert das Bild ab und beendet schließlich sich selbst.



4.2 Arbeitsweise eines Clients



Der Client wird in seiner grundsätzlichen Arbeitsweise von main gesteuert. Hier wartet er darauf, dass der Server ihm mitteilt, welche Konfiguration er verwenden soll und liest sie über die Funktion raytracer_setup aus dem Modul raytracer ein. Diese nutzt die Module script_tools zum Einlesen der Konfigurationsdaten und data um die Bilddaten einzulesen. Anschließend wartet er solange auf neue zu berechnende Zeilen, bis der Server ihm mitteilt, dass das Bild fertig ist und beendet sich dann selbst.

Das Modul raytracer ist ebenfalls für den gesamten Tracevorgang verantwortlich. Initiiert wird er, indem die Funktion raytrace_linerequest aufgerufen und ihr die zu tracende Zeile übergeben wird.

Die Datei ROOM.txt enthält alle Daten, die das Bild ausmachen. Sie wird über das Modul script_tools eingelesen und verwaltet.

5 Verwendete Arbeitsmaterialien

5.1 Verwendete Literatur

- Diplomarbeit von Gordon Mueller (`97)
- Course Scripts PVM/MPI, A.E. MSc for parallel and scientific Computing, Reading University, GB
  © 2001 by Prof. V. Alexandrov
- Beispielprogramme, die von PVM[3] mitgeliefert wurden

5.2 Verwendete Software

- MS VisualStudio zur Entwicklung des Raytracers
- mc sowie gcc version 2.96 20000731 zur Entwicklung des Servers sowie der Kommunikationsfunktionalität des Clients
- MS Word97, Adobe Acrobat4.0, Case4/0, Adobe Photoshop5.5, CorelDraw8 zur Erstellung der Dokumentation

6 Anhang:

Verteilung der Aufgaben im Team

Team-Leiter und Entwicklung des Raytracers:	Jens Meichsner
Umsetzung und Parallelisierung nach Linux:	Isabel Drost
Zusätzliche Programmierung und Vergleichsmessungen:	Martin Zennig, Gunnar Neubauer

Beispiel einer Konfigurationsdatei:

```
REM   Datei nicht verändern!!!
REM   Jens Meichsner

REM   Bild-Einstellungen
$IMAGE_NAME output.bmp
$IMAGE_WIDTH      640
$IMAGE_HEIGHT     480

REM   Daten-Datei
$INPUT_DATA ROOM.txt

REM   Raytracer-Einstellungen
$SHADOWS    1
$REFLECTIONS      0

REM   Hintergrundfarbe
$COLOR_BG_R 0.0
$COLOR_BG_G 0.0
$COLOR_BG_B 0.0

REM   Kameraeinstellungen
$CAM_POS_X  0.0
$CAM_POS_Y  -100.0
$CAM_POS_Z  25.0
$CAM_POS_ANGLE_Z  0.0
$CAM_POS_ANGLE_X  -15.0
```

[3] http://www.epm.ornl.gov/pvm

```
$CAM_POS_ZOOM     1.5

QUIT
```

Originaltext der GNU-General-Public-Liscence:

GNU GENERAL PUBLIC LICENSE
Version 2, June 1991

Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software--to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things. To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software. Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations. Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all. The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law:that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation inthe term "modification".) Each licensee is addressed as "you".Activities other than copying, distribution and modification are notcovered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program
is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.
b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.
c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users

may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.) These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program,and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it. Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,
c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the

author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.
This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.
8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.
9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.
Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.
10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.
NO WARRANTY
11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.
12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED OF LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.
END OF TERMS AND CONDITIONS