Fachbereich Mathematik und Informatik
der Philipps-Universität Marburg

GloVEd

Ein 3D-Szenen Editor für GloVe

Fortgeschrittenen Praktikum im WS 2002/2003 von

Markus Mühling und Adam Sosnowski

Leiter: Prof. Dr. Sommer, Axel Schröder

Inhaltsverzeichnis

1. Aufgabenstellung

Das Ziel unseres Projektes war die Implementierung eines 3D-Editors, um die Erstellung von Szenen für das Global Visibility Environment (GloVe) zu erleichtern. GloVe ist eine Testumgebung für globale Sichtbarkeitsalgorithmen, die im Rahmen eines Fortgeschrittenen Praktikums im SS 2001 entwickelt wurde. Szenen für GloVe ließen sich bisher nur mit Hilfe eines Texteditors erstellen. Dabei mußten Eigenschaften wie Typ, Name, Farbe, Position, etc. für jedes Objekt in der Szene im XML-Format in eine Textdatei geschrieben werden. Dies war sehr umständlich und für große Szenen zu aufwendig, wie das folgende Beispiel einer einfachen Szene zeigt:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE Scene SYSTEM "Scene.dtd">
<Scene>
  <Room name="Raum">
    <Pos coo="(0.0,0.0,0.0)"/>
    <Skinlist>
      <Skin dref="(0.5,0.5,0.5)" side="0" emi="(0.0,0.0,0.0)"/>
    </Skinlist>
    <Rot coo="(0,0,0)"/>
    <Scl coo="(40.0, 40.0, 40.0)"/>
  </Room>
  <Pyramid name="Pyramid0">
    <Pos coo="(-0.28, 1.13742854, 0.0)"/>
    <Rot coo="(0.4, 3.039, -1.46)"/>
    <Scl coo="(1.79, 4.19, 1.0)"/>
    <Skinlist>
      <Skin dref="(0.8, 0.8, 0.8)" side="0" emi="(0.0,0.0,0.0)"/>
      <Skin dref="(1.0, 1.0, 0.0)" side="1" emi="(0.0,0.0,0.0)"/>
      <Skin dref="(0.0, 0.0, 1.0)" side="2" emi="(0.0,0.0,0.0)"/>
      <Skin dref="(0.2, 1.0, 0.6)" side="3" emi="(1.0,1.0,1.0)"/>
      <Skin dref="(1.0, 0.276, 1.0)" side="4" emi="(1.0,1.0,1.0)"/>
    </Skinlist>
  </Pyramid>
</Scene>
```

Mit GloVEd sollte das Erstellen von 3D-Szenen per Maus möglich sein.

Folgende Anforderungen an GloVEd wurden gestellt:

- Konstruktion einfacher 3D-Szenen aus elementaren 3D-Objekten
- Schnelle Erzeugung „großer“ Zufallsszenen
- Laden und Speichern von Szenen im XML-Format
- Unterscheidung von „normalen“ 3D-Objekten und Lichtquellen
- Benutzerfreundliche GUI

2. Fachkonzept

Die Benutzeroberfläche von GloVEd ist in mehrere Bereiche unterteilt, siehe Screenshot in Abbildung 1.
Links befinden sich das Tool Panel, das Property Panel und das User View Panel (in der Reihenfolge von oben nach unten). Rechts befinden sich vier 3D-Ansichten der Szene: Ansicht von oben („Top“), Ansicht von Rechts („Right“), Ansicht von vorne („Front“) und benutzerdefinierte Ansicht („User“). Am oberen Rand des Hauptfensters befindet sich das Haupmenü, am unteren Rand des Hauptensters befindet sich die Statuszeile.

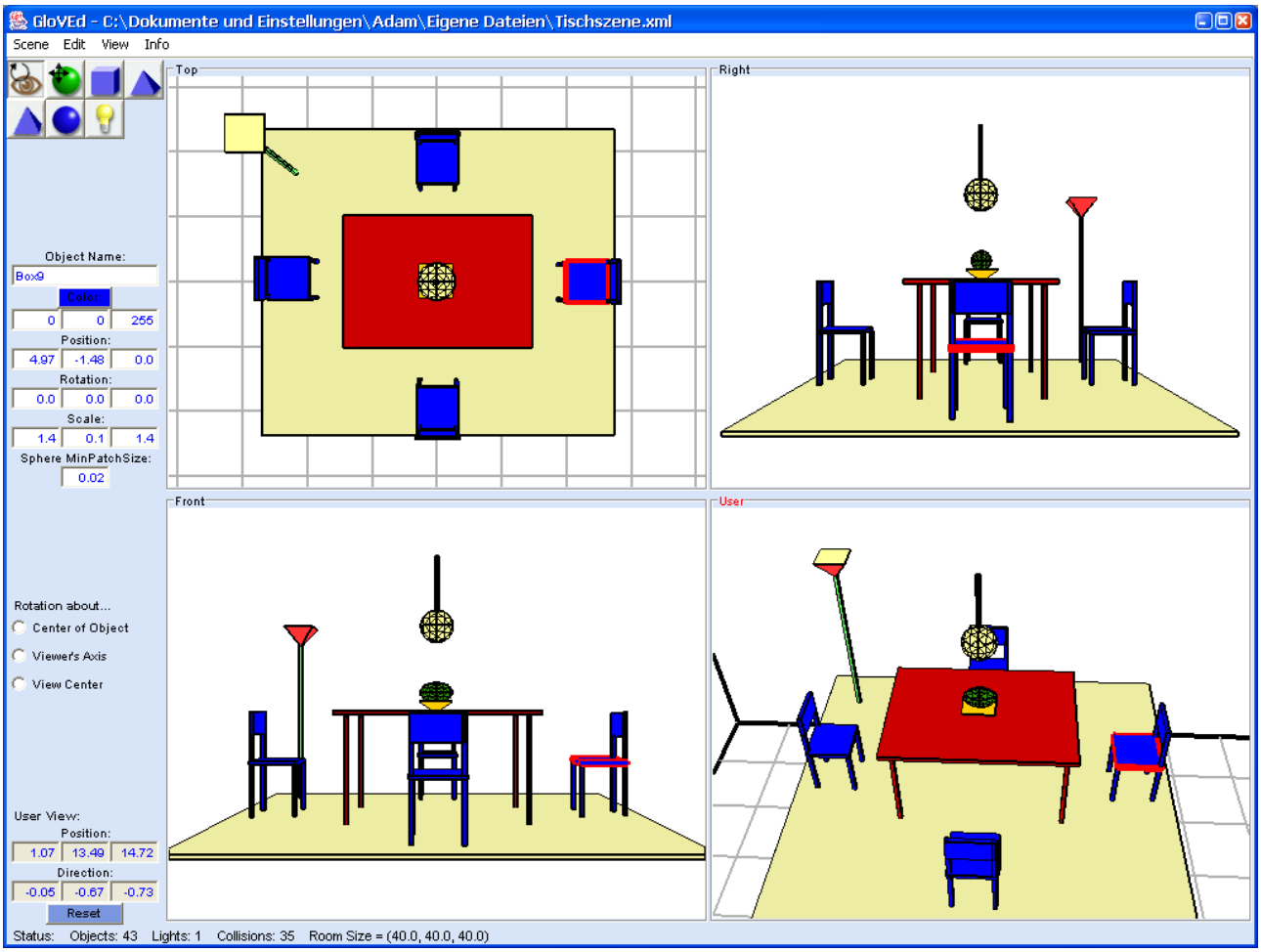

Abbildung 1

2.1. Hauptmenü

Das Hauptmenü von GloVEd gliedert sich in folgende Einträge: „Scene“, „Edit“, „View“ und „Info“.

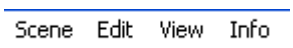

Abbildung 2

Das Menü ist kontextabhängig, d.h. nicht alle Menüeinträge sind immer anwählbar. Es hängt davon

ab, in welchen Modus man sich befindet (Lichtmodus oder Bearbeitungsmodus), in welche Ansicht man geklickt hat, ob ein Objekt selektiert ist und welches Objekt selektiert ist.

2.1.1. Menü Scene

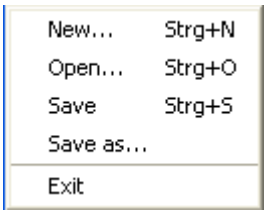

Abbildung 3

2.1.1.1. Menüpunkt Scene | New...

Mit diesem Befehl kann eine neue Szene erstellt werden. Ein Dialog zur Erstellung eines neuen Raumes wird angezeigt (siehe Abbildung 4). In diesem Dialog kann man die Größe des neuen Raumes einstellen. Die Default-Größe beträgt (40.0, 40.0, 40.0).
Die aktuelle Szene wird erst gelöscht, wenn der Dialog mit OK bestätigt wird. Falls die aktuelle Szene nicht leer ist, wird eine Bestätigung zum Löschen der Szene gefordert.

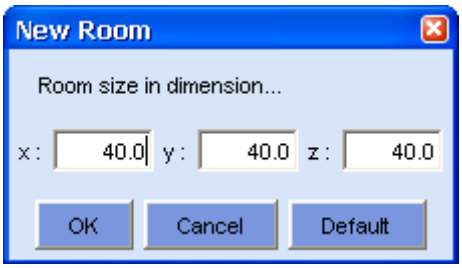

Abbildung 4

2.1.1.2. Menüpunkt Scene | Open...

Mit diesem Befehl kann eine Szene aus einer Datei geladen werden. Die aktuelle Szene wird erst gelöscht, wenn die Szene aus der ausgewählten Datei erfolgreich eingelesen werden konnte. Falls die aktuelle Szene nicht leer ist, wird eine Bestätigung zum Löschen der Szene gefordert. Nach erfolgreichem Öffnen einer Szene wird der Name der Szene in der Titelzeile des Fensters angezeigt.

2.1.1.3. Menüpunkt Scene | Save

Die aktuelle Szene wird unter dem aktuellen Dateinamen im XML-Format abgespeichert. Wenn die Szene noch keinen Namen hat, öffnet sich ein Datei-Speichern Dialog, um einen Dateinamen zu bestimmen. Nach erfolgreichem Speichern wird der Name der Szene in der Titelzeile des Fensters angezeigt.

2.1.1.4. Menüpunkt Scene | Save as...

Es öffnet sich ein Datei-Speichern Dialog, um die aktuelle Szene als XML-Datei abzuspeichern. Im Dialog kann einer neuer Name für die Szene eingegeben werden. Der neue Name der Szene wird in der Titelzeile des Fensters angezeigt.

2.1.1.5. Menüpunkt Scene | Exit

Das Programm wird beendet. Falls die aktuelle Szene nicht leer ist, wird eine Bestätigung zum Löschen der Szene gefordert.

2.1.2. Menü Edit

Abbildung 5

2.1.2.1. Menüpunkt Edit | Undo

Mit diesem Befehl kann der Benutzer die letzte unmittelbare Änderung an einem Objekt rückgängig machen. Zu den Änderungen, die rückgängig gemacht werden können, gehören nur Verschiebung, Rotation und Skalierung eines Objekts. Die Änderung kann nur rückgängig gemacht werden, solange kein anderes Objekt mit der Maus angeklick wird. Im Lichtmodus ist die Undo-Funktion ausgeschaltet.

2.1.2.2. Menüpunkt Edit | Redo

Mit Redo kann der letzte Undo-Befehl rückgängig gemacht werden. Der Undo-Befehl kann nur rückgängig gemacht werden, solange kein anderes Objekt mit der Maus angeklick wird. Im Lichtmodus ist die Redo-Funktion ausgeschaltet.

2.1.2.3. Menüpunkt Edit | Cut

Das gerade selektierte Objekt wird aus der Szene entfernt und im Clipboard gespeichert. Es kann nur ein ganzes Objekt ausgeschnitten werden.

2.1.2.4. Menüpunkt Edit | Copy

Das gerade selektierte Objekt wird im Clipboard gespeichert. Es kann nur ein ganzes Objekt kopiert werden.

2.1.2.5. Untermenü Edit | Insert

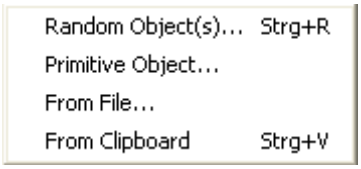

Abbildung 6

Im Untermenü „Insert“ werden vier verschiedenene Möglichkeiten angeboten, Objekte einzufügen. Im Lichtmodus ist dieses Untermenü deaktiviert.

2.1.2.5.1. Menüpunkt Edit | Insert | Random Object(s)...

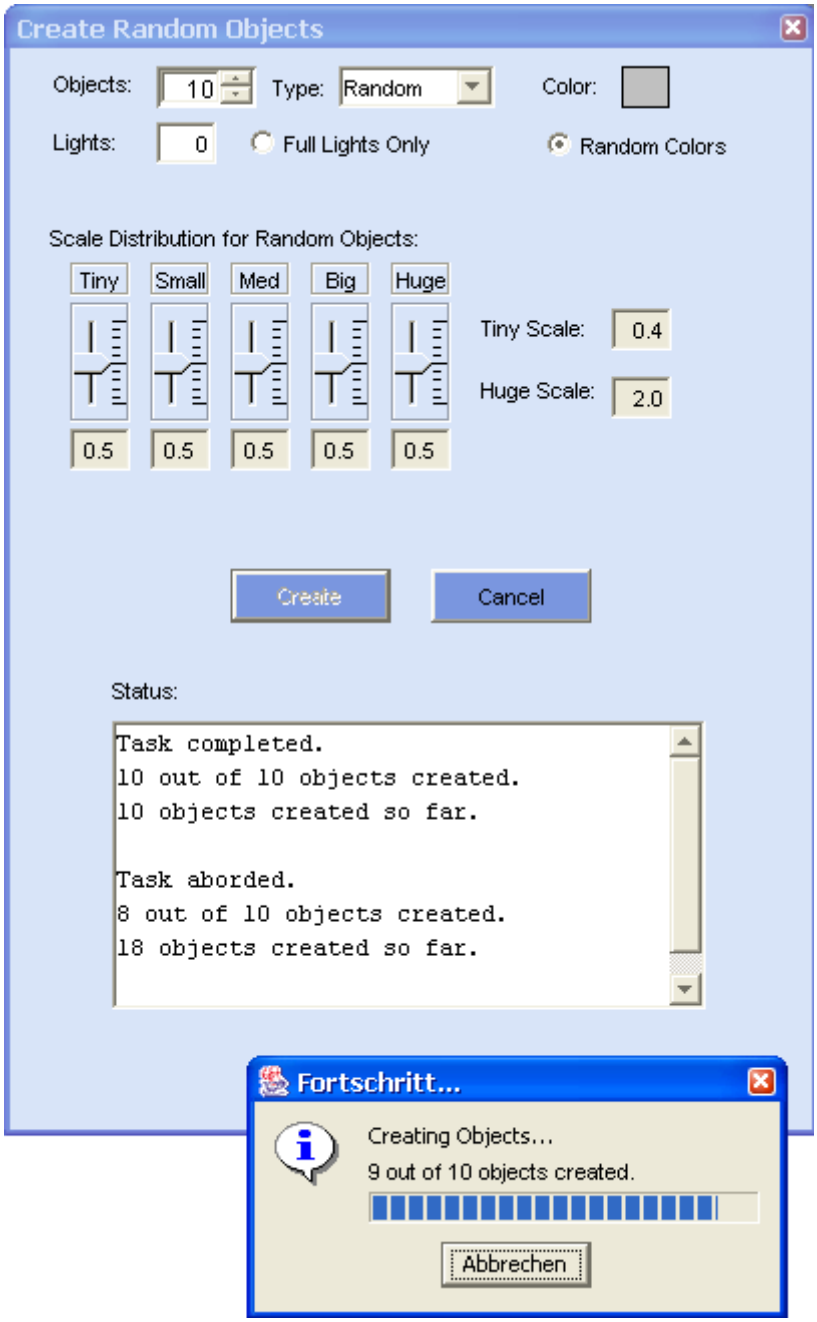

Abbildung 7

Mit diesem Befehl können Zufallsobjekte in die Szene eingefügt werden. Dazu muß vorher eine FillBox definiert worden sein (siehe Abschnitt 2.3.3). Wenn dies nicht der Fall sein sollte, wird der Benutzer darauf hingewiesen und der Dialog zum Erzeugen der Objekte wird nicht geöffnet.

In dem Dialog kann der Benutzer die Anzahl der zufällig zu erzeugenden Objekte eingeben (siehe Abbildung 7). Man kann direkt den Typ der Objekte wählen oder angeben, daß für jedes Objekt der Typ zufällig gewählt werden soll („Type Random“). Genauso kann man die Farbe der Objekte mit Hilfe eines Farbauswahl Dialogs angeben oder die Farbe für jedes Objekt zufällig erzeugen lassen („Random Colors“). Man kann angeben, wie viele der zufällig erzeugten Objekte als Lichtquelle

definiert werden sollen. Dabei kann man wählen, ob immer alle Seiten der Objekte als Lichtquelle definiert werden sollen („Full Lights Only“) oder jeweils eine zufällige Seite.

Mit Hilfe der fünf Schieberegler kann man die Größenverteilung für die zufällig erzeugten Objekte einstellen. Es handelt sich dabei um relative Wahrscheinlichkeiten, daß ein Objekt in einer der fünf Größen „Tiny“, „Small“, „Medium“, „Big“ und „Huge“ erzeugt wird. Die Größen werden automatisch anhand der Größe der zuvor definierten FillBox bestimmt. „Tiny“ beträgt ein zehntel und „Huge“ beträgt ein fünftel der kleinsten Seitenlänge der FillBox. Die Größen dazwischen werden in gleichmäßgen Abständen gebildet. Wenn der errechnete Wert für „Tiny“ kleiner als 0.1 betragen sollte, wird dem Benutzer in einem Dialog vorgeschlagen, als „Tiny“-Größe 0.1 zu wählen.

Mit den Button „Create“ wird die Generierung der Zufallsobjekte gestartet. Daraufhin wird versucht, die angegebene Anzahl Objekte in die FillBox überschneidungsfrei einzufügen. Jedem Objekt wird automatisch ein Name der Form „Objekt-Typ“ + „Zahl“ vergeben, wobei „Zahl“ die Anzahl der gerade in der Szene befindlichen Objekte des betreffenden Typs ist (z.B. Box25).

Sollte der Generierungsvorgang länger dauern, wird eine Fortschrittsanzeige eingeblendet, welche die Möglichkeit zum vorzeitigen Abbruch der Generierung ermöglicht.

Im Statusfenster wird die Anzahl der im letzten Generierungsvorgang erzeugten Objekte und die Anzahl der mit dem aktuellen Dialog bisher erzeugten Objekte ausgegeben. Zwischen den Generierungsvorgängen können die Parameter verändert werden.

Beim Schliessen des Dialogs wird die FillBox aus der Szene entfernt.

2.1.2.5.2. Menüpunkt Edit | Insert | Primitive Object...

Es wird ein Dialog geöffnet, in dem der Benutzer nach Festlegung von Typ, Farbe und Position ein Objekt in die Szene einfügen kann (siehe Abbildung 8).

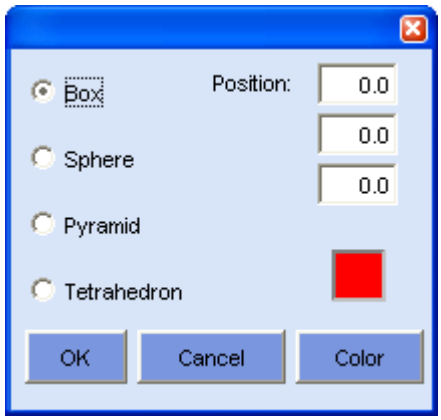

Abbildung 8

2.1.2.5.3. Menüpunkt Edit | Insert | From File...

Der Benutzer kann eine gespeicherte Szene aus einer Datei laden und in die aktuelle Szene einfügen. Wenn der Raum der eingefügten Szene größer ist als der Raum der aktuellen Szene, dann wird der aktuelle Raum in den entsprechenden Dimensionen angepasst. Neu eingefügte Objekte werden umbenannt, wenn bereits Objekte mit dem gleichen Namen in der Szene existieren.

2.1.2.5.4. Menüpunkt Edit | Insert | From Clipboard

Das im Clipboard gespeicherte Objekt wird im Koordinatenursprung, also in der Mitte ders Raumes in der Szene eingefügt und selektiert.

2.1.2.6. Menüpunkt Edit | Delete

Das gerade selektierte Objekt wird aus der Szene entfernt, ohne es im Clipboard zu speichern.

2.1.3. Menü View

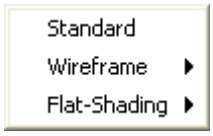

Abbildung 9

2.1.3.1. Menüpunkt View | Standard

Die Zoom-Einstellungen der Ansichten „Top“, „Right“, „Front“ und „User“ sowie die Perspektive der Ansicht „User“ werden auf die Ausgangseinstellungen zurückgesetzt.

2.1.3.2. Untermenü View | Wireframe

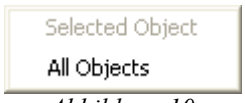

Abbildung 10

2.1.3.2.1. Menüpunkt View | Wireframe| Selected Object

Mit diesem Befehl wird das gerade selektierte Objekt in Drahtgitterdarstellung angezeigt. Dieser Befehl ist im Lichtmodus deaktiviert.

2.1.3.2.2. Menüpunkt View | Wireframe| All Objects

Mit diesem Befehl werden alle Objekte in der Szene in Drahtgitterdarstellung angezeigt. Dieser Befehl ist im Lichtmodus deaktiviert.

2.1.3.3. Untermenü View | Flat-Shading

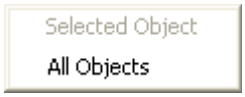

Abbildung 11

2.1.3.3.1. Menüpunkt View | Flat-Shading | Selected Object

Mit diesem Befehl wird das gerade selektierte Objekt in Flat-Shading angezeigt. Dieser Befehl ist im Lichtmodus deaktiviert.

2.1.3.3.2. Menüpunkt View | Flat-Shading | All Objects

Mit diesem Befehl werden alle Objekte in der Szene in Flat-Shading angezeigt. Dieser Befehl ist im

Lichtmodus deaktiviert.

2.1.4. Menü Info

Abb. 12

2.1.4.1. Menüpunkt Info | About...

Der Benutzer erhält allgemeine Informationen über das Programm.

2.2. 3D-Ansichten

Es gibt vier Ansichten der Szene: Ansicht von oben („Top“), Ansicht von Rechts („Right“), Ansicht von vorne („Front“) und benutzerdefinierte Ansicht („User“). Während die ersten drei Ansichten nur gezoomt werden können, in der Perspektive jedoch fest sind, kann in der benutzerdefinierten Ansicht auch die Perspektive verändert werden. Um die Perspektive zu verändern, muß man im Tool Panel den „View Button“ aktivieren (siehe Abschnitt 2.4.1) und dann mit der linken Maustaste in das View-Fenster klicken. Durch Ziehen bei gedrückter linker Maustaste kann man die Position des Augpunktes rotieren. Ebenso kann man durch Ziehen bei gedrückter rechter bzw. mittlerer Maustaste die Position des Augpunktes parallel bzw. senkrecht zur Sichtebene verschieben.
In den drei festen Ansichten kann man die Position des Augunktes ausschließlich senkrecht zur Sichtebene verschieben („Zoomen“).

Um die Ausgangseinstellungen für alle Ansichten wiederherzustellen, muß man den Menüpunkt „View | Standard“ betätigen (siehe Abschnitt 2.1.3.1). Wenn man nur die Perspektive für die User-View zurückstellen möchte, muß man den Button „Reset“ im User View Panel betätigen (siehe Abschnitt 2.6.6).

Im Menü View (siehe Abschnitt 2.1.3) und im Popup Menü (siehe Abschnitt 2.3) befinden sich Befehle, mit denen man die Darstellung der Objekte zwischen Drahtgitterdarstellung und Flat-Shading wechseln kann.

Standardmäßig sind alle Ansichten auf Perspektivprojektion eingestellt. Man kann aber in allen Ansichten zwischen paralleler Projektion und Perspektivprojektion wechseln. Dazu muß man mit der rechten Maustaste in die betreffende Ansicht klicken und den Menüpunkt „Parallel Pojection“ im Popup Menü benutzen (siehe Abschnitt 2.3.9).

Im Popup Menü befindet sich außerdem ein Eintrag, mit dem man die User-View auf Fullscreen schalten kann (siehe Abschnitt 2.3.8).

2.3. Popup Menü

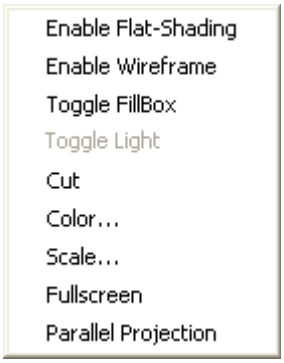

Abbildung 13

Wenn man mit der rechten Maustaste in eine der vier Ansichten klickt, öffnet sich das Popup Menü. Das Menü ist kontextabhängig, d.h. nicht alle Menüeinträge sind immer anwählbar. Es hängt davon ab, in welchen Modus man sich befindet (Lichtmodus oder Bearbeitungsmodus), in welche Ansicht man geklickt hat, ob ein Objekt selektiert ist und welches Objekt selektiert ist.

2.3.1. Menüpunkt Enable Flat-Shading

Mit diesem Befehl wird das gerade selektierte Objekt in Flat-Shading angezeigt. Dieser Befehl ist im Lichtmodus deaktiviert.

2.3.2. Menüpunkt Enable Wireframe

Mit diesem Befehl wird das gerade selektierte Objekt in Drahtgitterdarstellung angezeigt. Dieser Befehl ist im Lichtmodus deaktiviert.

2.3.3. Menüpunkt Toggle FillBox

Mit diesem Befehl wird ein selektiertes Objekt als FillBox definiert bzw. eine FillBox wird wieder zu einem normalen Objekt. Dieser Befehl läßt sich nur anwenden, wenn das selektierte Objekt vom Typ „Box“ ist und das Objekt noch nicht rotiert ist (der Quader ist dann achsenparallel). Wird ein Objekt als FillBox definiert, wird es in Drahtgitterdarstellung angezeigt und die Kanten des Objekts werden dick hellgrün gezeichnet. In einer Szene kann gleichzeitig immer nur eine FillBox existieren. Um ein anderes Objekt als FillBox zu definieren, muß daher zuerst die alte FillBox zu einem normalen Objekt gemacht werden.
Eine FillBox kann nicht rotiert werden. Dieser Befehl ist im Lichtmodus deaktiviert.

2.3.4. Menüpunkt Toggle Light

Das selektierte Objekt bzw. die selektierte Seite eines Objekts wird als Lichtquelle definert und umgekehrt. Dieser Befehl steht nur im Lichtmodus zur Verfügung.

2.3.5. Menüpunkt Cut

Das gerade selektierte Objekt wird aus der Szene entfernt und im Clipboard gespeichert. Es kann nur ein ganzes Objekt ausgeschnitten werden.

2.3.6. Menüpunkt Color...

Mit diesem Befehl kann die Farbe für das gerade selektierte Objekt bzw. für die gerade selektierte Seite eines Objekts geändert werden. Zu diesem Zweck öffnet sich ein Farbauswahl Dialog.

2.3.7. Menüpunkt Scale...

Mit diesem Befehl kann das selektierte Objekt skaliert werden. Dies geschieht mit Hilfe eines speziellen Dialogs (siehe Abbildung 14).

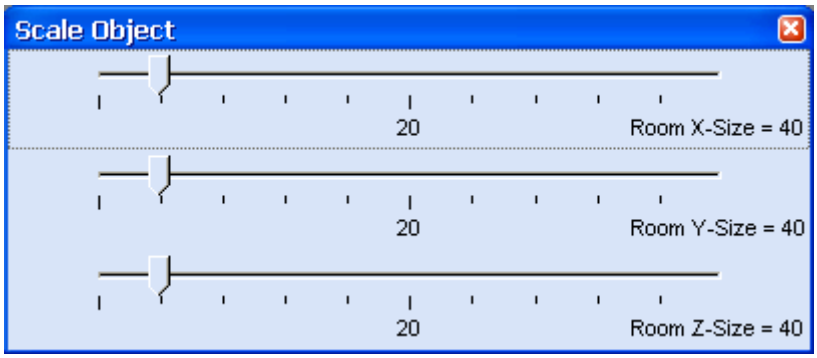

Abbildung 14

Das Objekt kann über Schieberegler in jeder Dimension skaliert werden. Die maximale Skalierung wird in jeder Dimension durch die Ausmaße des Raumes begrenzt. Die Skalierung erfolgt in Schritten von 0.1.

2.3.8. Menüpunkt Fullscreen

Dieser Befehl steht nur für die Ansicht „User“ zur Verfügung. Damit läßt sich die User-View zwischen Fullscreen und normaler Größe wechseln. In Fullscreen nimmt die User-View die Fläche aller vier Ansichten an, die anderen Ansichten sind nicht mehr sichtbar.

2.3.9. Menüpunkt Parallel Projection

Mit diesem Befehl wird in der aktuellen Ansicht zwischen paralleler Projektion und Perspektivprojektion gewechselt. Es sollte hier erwähnt werden, daß in paralleler Projektion nicht gezoomt werden kann.

2.4. Tool Panel

Abbildung 15

2.4.1. View Button

Wenn man den View Button aktiviert (obere Reihe, erstes Icon von links; siehe Abbildung 15), kann man durch Klicken und Ziehen mit der Maus die Ansichten verändern. Für eine genauere Beschreibung siehe Abschnitt 2.2.

2.4.2. Object Button

Wird der Object Button aktiviert (obere Reihe, zweites Icon von links), kann der Benutzer Objekte selektieren und ihre Position und Rotation verändern. Das Selektieren eines Objekts erfolgt in einer beliebigen Ansicht durch einmaliges Klicken mit der linken Maustaste über dem Objekt. Das Selektieren einer Seite eines Objekts erfordert zweimaliges Klicken mit der linken Maustaste über der entsprechenden Objekt-Seite. Bei einem selektierten Objekt bzw. einer selektierten Objekt-Seite werden die Kanten hellblau gezeichnet. Wenn sich das selektierte Objekt jedoch mit anderen Objekten schneidet, werden die Kanten des Objekts dick rot gezeichnet. Bei unselektierten Objekten werden die Kanten schwarz gezeichnet. Es kann gleichzeitig immer nur ein Objekt bzw. eine Objekt Seite selektiert sein.

Wenn ein Objekt selektiert ist, kann es bei gedrückter linker Maustaste rotiert werden, indem die Maus auf und ab bewegt wird. Das Objekt wird dabei standardmäßig um die Achse rotiert, die dem Benutzer aus einer Ansicht entgegenragt. D.h. wird z.B. ein Objekt in der Top-View mit der linken Maustaste angeklickt und die Maustaste gedrückt gehalten, dann wird das Objekt um die y-Achse rotiert. Die Rotationsachse kann während des Rotierens gewechselt, indem man die Taste „x“, „y“ oder „z“ gedrückt hält. Dann wird das Objekt um die entsprechende Achse rotiert. In der User-View können Objekte nicht rotiert werden.

Ein selektiertes Objekt kann verschoben werden, wenn es mit einer der anderen beiden Maustasten angeklickt wird und bei gedrückter Taste die Maus auf und ab bewegt wird. Mit der rechten Maustaste wird das Objekt parallel zur Sichtebene einer Ansicht verschoben, mit der mittleren Maustaste senkrecht zur Sichtebene (also vom Benutzer weg bzw. zum Benutzer hin).

2.4.3. Box Button

Ist der Box Button angewählt (obere Reihe, zweites Icon von rechts), dann wird jedesmal ein Objekt vom Typ „Box“ in die Szene eingefügt, wenn man mit der linken Maustaste in eine Ansicht klickt. Das Objekt wird an die Position eingefügt, über der sich die Maus gerade in einer Ansicht befindet. Dem Objekt wird automatisch ein Name der Form „Box“ + „Zahl“ vergeben, wobei „Zahl“ die Anzahl der gerade in der Szene befindlichen Objekte vom Typ „Box“ ist.
Das neu eingefügte Objekt wird automatisch selektiert. In der User-View können keine Objekte eingefügt werden. Dieser Button ist im Lichtmodus deaktiviert.

2.4.4. Pyramid Button

Wie in Abschnitt 2.4.3 nur mit Objekten vom Typ „Pyramid“. Der Pyramid Button befindet sich in der oberen Reihe ganz rechts.

2.4.5. Tetrahedron Button

Wie in Abschnitt 2.4.3 nur mit Objekten vom Typ „Tetrahedron“. Der Tetrahedron Button befindet sich in der unteren Reihe ganz links.

2.4.6. Sphere Button

Wie in Abschnitt 2.4.3 nur mit Objekten vom Typ „Sphere“. Der Sphere Button befindet sich in der unteren Reihe als zweiter von links.

2.4.7. Light Button

Mit dem Light Button (untere Reihe, ganz rechts) kann der Benutzer zwischen dem Lichtmodus und dem Bearbeitungsmodus wechseln. Im Lichtmodus werden alle Objekte in Drahtgitterdarstellung angezeigt. Nur die Seiten der Objekte, die als Lichtquelle definiert sind, werden in Flat-Shading in ihrer Farbe angezeigt. Im Lichtmodus können neue Lichter definiert werden und Lichter wieder zu normalen Objekt-Seiten gemacht werden. Dazu muß im Lichtmodus der Object Button aktiviert werden. Nun können Objekte bzw. einzelne Objekt-Seiten selektiert werden (siehe Abschnitt 2.4.2) und dann mit Hilfe des Popup Menüs die Lichteigenschaft verändert werden (siehe Abschnitt 2.3.4).
Im Lichtmodus können keine neuen Objekte in die Szene eingefügt werden.

2.5. Property Panel

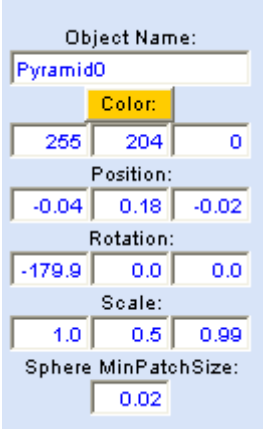

Abbildung 16

2.5.1. Object Name

Dieses Feld zeigt den Namen des gerade selektierten Objekts an. Wenn nur eine Seite eines Objekts selektiert ist, wird angegeben, zu welchem Objekt die Seite gehört. Durch Eingabe in das Textfeld kann der Objektname geändert werden. Der Objektename wird erst akzeptiert, wenn es kein anderes Objekt in der Szene mit dem gleichen Namen gibt. Die Eingabe des neuen Namens muß mit Return bestätigt werden. Dies gilt auch für die Eingaben in alle nachfolgenden Felder.

2.5.2. Color

Der Color Button zeigt die Farbe des gerade selektierten Objekts bzw. der gerade selektierten Objektseite an. Wird der Button betätigt, öffnet sich ein Farbauswahl Dialog und es kann eine neue Farbe ausgewählt werden. Unter dem Color Button befinden sich drei Felder, die die RGB-Werte der aktuellen Farbe anzeigen. Man kann in die Felder direkt die RGB-Werte der Farbe eingeben. Es werden nur ganzahlige Werte zwischen 0 und 255 akzeptiert. Bei Fehleingabe wird dem Benutzer das erlaubte Format angezeigt.

2.5.3. Position

Ausgabe der Position des gerade selektierten Objekts in x-, y-, z-Koordinaten. Die Anzeige wird laufend aktualisiert. Man kann in die Felder neue Koordinaten für das Objekt eingeben. Es sind nur Koordinaten innerhalb des Raums erlaubt. Bei Fehleingabe werden dem Benutzer die Grenzen für

die Koordinaten angezeigt.

2.5.4. Rotation

Ausgabe der Rotation des gerade selektierten Objekts um die x-, y-, und z-Achse. Es handelt sich um Winkel in Grad. Positive Werte stehen für Winkel gegen den Uhrzeigersinn. Die Anzeige wird laufend aktualisiert. Man kann durch Eingabe in die Felder dem Objekt eine neue Rotation zuweisen. Es sind Winkel größer 360 Grad erlaubt.

2.5.5. Scale

Ausgabe der Größe des gerade selektierten Objekts in den Dimensionen x, y und z. Die Anzeige wird laufend aktualisiert. Man kann in die Felder neue Skalierungen für das Objekt eingeben. Es sind nur Skalierungen erlaubt, die nicht größer sind als die Dimension des Raumes. Bei Fehleingabe werden dem Benutzer die Grenzen für die Skalierung angezeigt.

2.5.6. Sphere MinPatchSize

Die MinPatchSize bezeichnet die minimale Fläche für die Dreiecke, aus denen ein Objekt vom Typ „Sphere“ erzeugt werden soll. Je kleiner der Wert für MinPatchSize ist, desto mehr Dreiecke werden für das Erzeugen einer Kugel verwendet (desto detaillierter wird die Kugel gezeichnet). Als sinnvolle untere Grenze für die Komplexität einer Kugel ist der Wert 0.005 festgelegt. Der maximale Wert für MinPatchSize beträgt 0.1. Bei diesem Wert stellt die Kugel ein Oktahedron dar.
Bei Fehleingabe werden dem Benutzer das erlaubte Minimum und Maximum für MinPatchSize angezeigt.

Ist ein Objekt vom Typ „Sphere“ selektiert, zeigt dieses Feld die MinPatchSize des Objekts an. Ansonsten zeigt es die MinPatchSize für darauffolgend eingefügte Objekte vom Typ „Sphere“ an. Dies gilt sowohl für Objekte, die per Sphere Button in die Szene eingefügt werden (siehe Abschnitt 2.4.6), als auch für Objekte, die während der Zufallserzeugung generiert werden (siehe Abschnitt 2.1.2.5.1).

2.6. User View Panel

Abbildung 17

2.6.1. Center of Object

In diesem Modus erfolgt die Rotation des Augpunktes in der User-View um das Zentrum des gerade seletkierten Objekts.

2.6.2. Viewer's Axis

In diesem Modus erfolgt die Rotation der Augpunktes in der User-View um den Augpunkt selbst.

2.6.3. View Center

In diesem Modus erfolgt die Rotation des Augpunktes in der User-View um das Zentrum der Szene.

2.6.4. Position

Ausgabe der aktuellen Position des Augpunktes der User-View in x-, y-, z-Koordinaten. Die Anzeige wird laufend aktualisiert.

2.6.5. Direction

Ausgabe des aktuellen Richtungsvektors des Augpunktes der User-View in x-, y-, z-Koordinaten. Die Anzeige wird laufend aktualisiert.

2.6.6. Reset Button

Stellt die Ausgangseinstellungen für die Perspektive in der User-View wieder her.

2.7. Statuszeile

In der Statuszeile wird die aktuelle Anzahl der in der Szene befindlichen Objekte und Lichter angezeigt. Außerdem wird die Anzahl der Kollisionen zwischen Objekten angezeigt und die Größe des derzeitigen Raumes.

3. Technische Umsetzung

Das Projekt wurde mit dem Java SDK 1.4.1 und Java3D 1.3 entwickelt. Es unterteilt sich in 6 Packages:

- *gloved*
- *geometry*
- *orbitbehavior*
- *picking*
- *rtree*
- *io*

Im Package *gloved* befinden sich die GUI-Klassen, das Package *geometry* beinhaltet die Klassen zur Repräsentation der 3D-Objekte und die Packages *orbitbehavior* und *picking* enthalten Klassen, die die Interaktion zwischen der Szene und dem Benutzer ermöglichen. Da in unserem Programm mehrere Sichten auf die Szene ermöglicht werden, wurden die Java3D-Klassen zum Navigieren durch die Szene um die Methode *setCanvas* erweitert, um je nachdem in welchem Fenster (Top, Right, Front, User) man sich befindet, dem OrbitBehavior-Objekt den richtigen Canvas zuzuweisen. Die pickbehavior-Klassen sind eine Erweiterung der entsprechenden Klassen aus der Bibliothek Java3D. Diese wurden so modifiziert, daß beim Verschieben der 3D-Objekte die Blickrichtung des Benutzers berücksichtigt wird. Weiterhin kann mit Hilfe der Klasse *CollisionTest* eine Methode an die PickBehavior übergeben werden, die beim Verändern der 3D-Objekte eine Schnitterkennung anstößt. Die Klasse *OrbitBehaviorCallback* ermöglicht es, eine Methode an die *OrbitBehavior* zu übergeben, um beim Verändern der Sicht die entsprechenden Werte in der GUI zu aktualisieren. Die wichtigste Klasse im Package *gloved* ist die Klasse *MainFrame*. Sie enthält sowohl den Aufbau der GUI als auch den Aufbau des Szenegraphen. Der Schwerpunkt beim Aufbau der GUI lag darin, mit den Möglichkeiten von Swing und Java3D eine möglichst „benutzerfreundliche“ GUI zu entwerfen. Das Package *rtree* beinhaltet die Datenstruktur zur Verwaltung der 3D-Objekte. Wir haben eine bestehenende R-Baum Implementierung von einem Mitarbeiter der Universität von Kalifornien übernommen und an unsere Bedürfnisse angepasst. Diese Datenstruktur ermöglicht uns eine effiziente Schnitterkennung (siehe Kapitel 3.3). Der Datenaustausch zwischen GloVE und unserem Programm geschieht mit Hilfe von XML. Die benötigten Klassen befinden sich im Package *io*.

3.1. Aufbau des Szenegraphen

Zentraler Begriff von Java3D ist der Szenegraph. Dieser Graph besitzt im Allgemeinen eine Baumstruktur. Die Wurzel ist immer ein Knoten vom Typ „Virtual Universe“. In diesem Universum gibt es „lokale Welten“. Wir interessieren uns für die Darstellung genau einer lokalen Welt. Abbildung 18 zeigt den Aufbau der lokalen Welt.

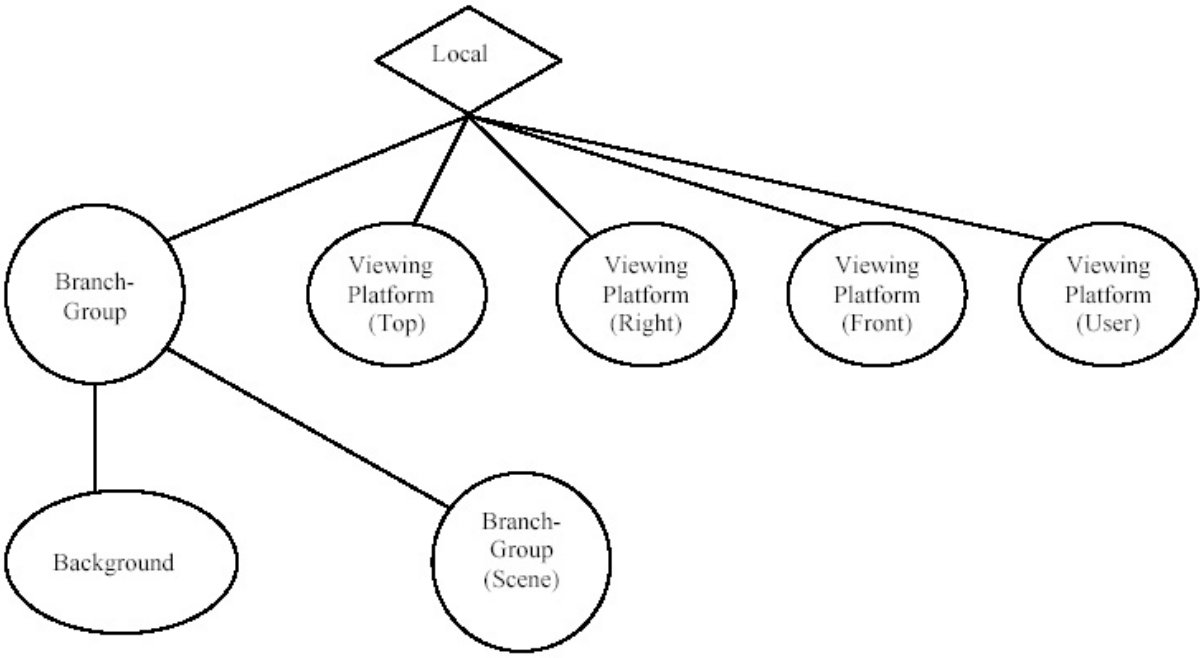

Abbildung 18

Das besondere an unserem Szenegraphen ist, daß vier „ViewingPlatforms“ (für jede Ansicht eine „ViewingPlatform“) an das Local-Objekt angehängt werden. Mit Hilfe der „ViewingPlatform“ werden praktisch alle Aspekte der Darstellung (Art der Perspektive, Blickrichtung,...) der Szene abgehandelt. Zur Darstellung der elementaren 3D-Objekte haben wir die Klasse *Primitive* implementiert. Diese abstrakte Klasse ist eine Erweiterung der Klasse *BranchGroup*. Es existieren folgende elementaren 3D-Objekte, welche die Klasse Primitive implementieren.

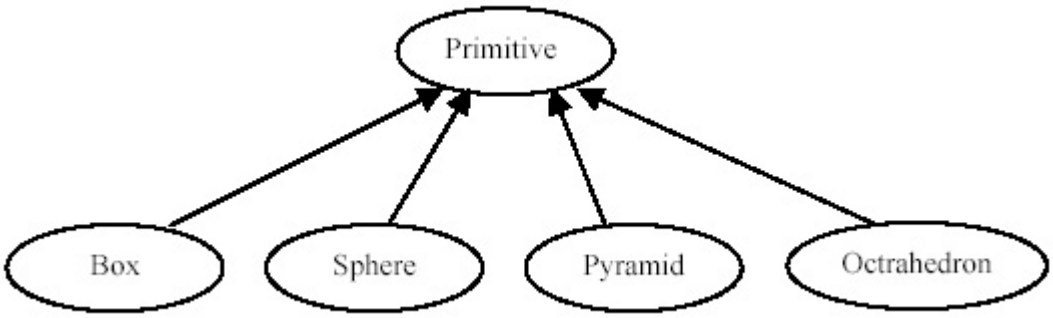

Abbildung 19

An den BranchGroup *Scene* des Szenegraphen werden die unterschiedlichen 3D-Objekte vom Typ Primitive wie folgt angehängt:

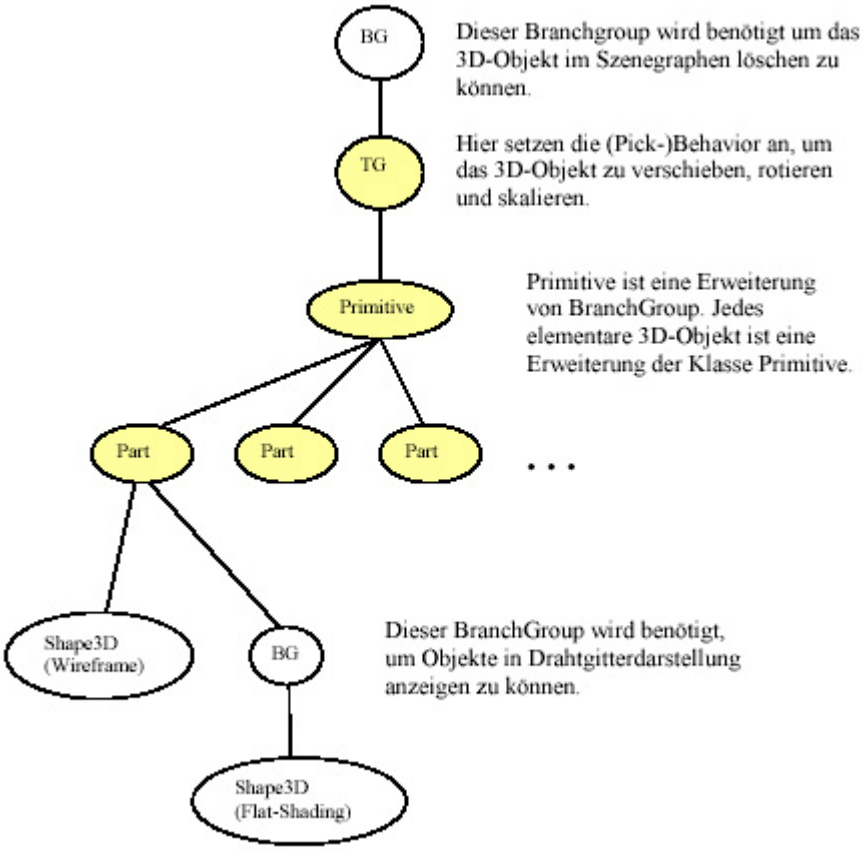

Abbildung 20

Primitive selbst besteht aus mehreren Objekten der Klasse *Part*, je nachdem aus wievielen Flächen das 3D-Objekt besteht. Dies ermöglicht es einzelne Seiten des 3D-Objekts zu selektieren. Bei den gelb markierten Knoten der obigen Grafik ist Pickreporting eingeschaltet. Picking ermöglicht dem Benutzer mit visuellen Objekten der Szene zu interagieren. Ein visuelles Objekt der Szene wird „gepickt“, indem der Mauszeiger über dem entsprechenden Objekt platziert und der linke „Mousebutton“ gedrückt wird. Durch diesen Klick wird ein Behaviorobjekt gestartet. Mi Hilfe der Position des Mauszeigers wird ein Strahl in die Szene berechnet und mit den visuellen Objekten der Szene auf Schnitt getestet. Das 3D-Objekt, welches die geringste Entfernung zum Strahlursprung hat, wird ausgewählt.

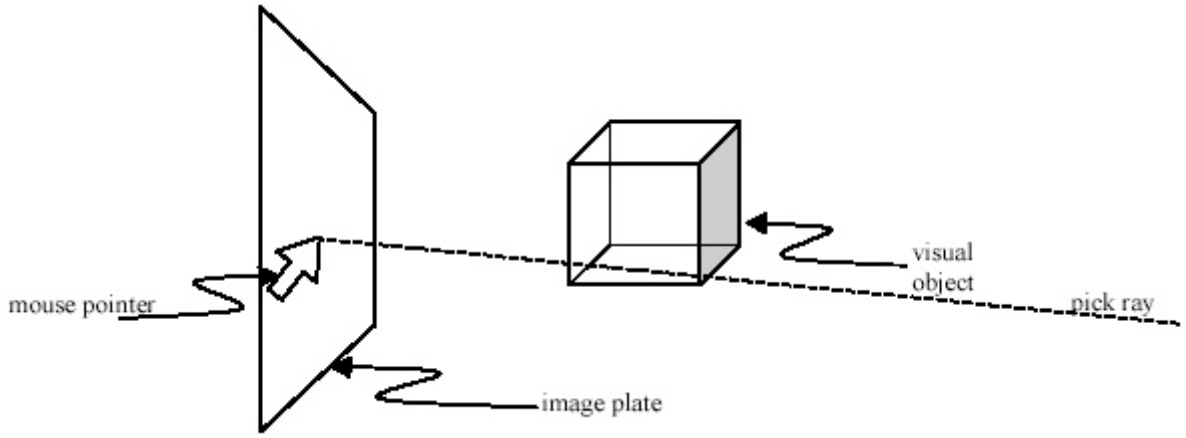

Abbildung 21

Das PickingBehavior liefert den Pfad zum ausgewählten 3D-Objekt innerhalb des Szenegraphen. Mit Hilfe von Pickreporting kann man bestimmen, welche Knoten entlang des Pfades zurückgeliefert werden (siehe Abbildung 21).

3.2. Schnitterkennung

Eine schnelle Schnitterkennung war von besonderer Bedeutung für unser Programm. Sie ermöglicht es zum einen relativ schnell, „große“ Zufallsszenen von sich nicht schneidenden Objekten zu erstellen. Zum anderen ist es mit einer schnellen Schnitterkennung möglich, Kollisionen in Echtzeit anzuzeigen (d.h. noch während man die 3D-Objekte verändert). Als zugrunde liegende Datenstruktur wählten wir hierzu einen R-Baum.
Zur Verwaltung unserer 3D-Objekte im R-Baum haben wir eine Klasse *Object3D* implementiert. Objekte vom Typ Object3D enthalten Referenzen auf das zugehörige Primitive-Objekt sowie dessen TransformGroup-Objekt. Über das Userdata-Feld von Primitive wird auch von einem Primitive-Objekt auf das zugehörige Objekt vom Typ Object3D referenziert. Object3D enthält zusätzlich ein Objekt vom Typ *BoundingBox* (minimale, achsenparallele Box, die das 3D-Objekt einschließt), die sobald das zugehörige 3D-Objekt rotiert, skaliert oder verschoben wird, aktualisiert werden muß. Außerdem wurde in Object3D die Methode zur Schnitterkennung zwischen zwei 3D-Objekten realisiert. Diese Methode wird weiter unten beschrieben.

Ein R-Baum ist eine Zugriffsstruktur für die effiziente Verwaltung von geometrischen Objekten. Er ist eine Verallgemeinerung der Idee des B^+-Baums. Im R-Baum gibt es zwei Typen von Knoten, nämlich die Blattknoten und die inneren Knoten. Die eigentlichen Datensätze bzw. die Verweise werden in den Blattknoten gehalten. In den inneren Knoten liegen nur solche Einträge, die aus einer n-dimensionalen Region (in unserem Fall n=3) und Referenzen auf Nachfolger Knoten bestehen. Die n-dimensionale Region ist die minimale Box, die alle Boxen oder Datensätze der Nachfolgerknoten begrenzt. Wenn man überprüfen möchte, ob ein 3D-Objekt geschnitten wird, muß die Schnitterkennung für konvexe Polyeder also nicht mehr für alle 3D-Objekte der Szene durchgeführt werden. Die Einteilung des R-Baums in Regionen schränkt die Anzahl der möglichen Kandidaten sehr stark ein. Der Test, ob sich zwei achsenparallele Boxen schneiden, funktioniert sehr schnell. Erst wenn sich die „Minimal Bounding Boxen“ zweier 3D-Objekte schneiden, werden die Objekte selbst auf Kollision überprüft. Die Schnitterkennung für zwei konvexe Polyeder läuft in 6 Schritten ab:

Die beiden 3D-Objekte, die auf Kollision überprüft werden sollen, nennen wir A und B. Jedes Objekt besteht aus mehreren Ebenen (E_{Ai} und E_{Bi}) und Kanten (K_{Ai} und K_{bi}).
Jede Seite eines Objekts definiert eine Ebene, die den Raum in zwei Halbräume teilt. Der positive Halbraum ist derjenige, in den der Normalenvektor hineinzeigt.

1)	$\exists E_{Ai}: B \subset E_{ai}^{+}$	==>	kein Schnitt
2)	$\exists E_{Bi}: A \subset E_{Bi}^{+}$	==>	kein Schnitt
3)	$\forall E_{Ai}: B \subset E_{Ai}^{-}$	==>	enthalten
4)	$\forall E_{Bi}: A \subset E_{Bi}^{-}$	==>	enthalten
5)	$\exists K_{Ai}: B \cap K_{Ai} \neq \varnothing$	==>	Schnitt
6)	$\exists K_{Bi}: A \cap K_{Bi} \neq \varnothing$	==>	Schnitt

Im ersten Schritt wird überprüft, ob eine Ebene von Objekt A existiert, so daß alle Punkte von Objekt B im positiven Halbraum dieser Ebene liegen. Falls eine solche Ebene existiert, schneiden sich die beiden Objekte A und B nicht. Test 2 verläuft bis auf Vertauschung der Rollen von Objekt A und B analog.
Die beiden nächsten Tests (3 und 4) überprüfen, ob für alle Ebenen eines Objekts die Punkte des anderen Objekts im negativen Halbraum liegen. D.h., falls zum Beispiel für alle Ebenen von Objekt A die Punkte von Objekt B komplett im negativen Halbraum liegen, so ist Objekt B in Objekt A enthalten.
Test 5 überprüft, ob eine Kante von Objekt A existiert, welche Objekt B schneidet. Die Überprüfung, ob eine Kante eine Fläche schneidet, haben wir mit Hilfe von Plückerkoordinaten realisiert. Plückerkoordinaten sind ein sehr eleganter Ansatz zur Erkennung von Strahl-Polygon-Schnitten. Eine gerichtete Linie im 3D ist durch zwei Punkte realisiert. Diese Punkte werden nun auf einen 6D-Punkt gemappt (Plücker-Transformation). Eine Plückerkoordinate repräsentiert somit einen Strahl (Linie). Das Skalarprodukt zweier Plückerkoordinaten P und Q gibt Aufschluß darüber, ob Linie Q Linie P schneidet bzw. parallel verläuft oder ob Linie Q sich mit oder gegen den Uhrzeigersinn um Linie P dreht. Unter Verwendung von Plückerkoordinaten und Plücker-Skalarprodukten läßt sich somit ein einfacher Algorithmus zur Erkennung von Strahl-Polygon-Schnitten implementieren.
Test 6 überprüft analog zu Test 5, ob eine Kante von Objekt B existiert, die Objekt A schneidet.
Falls einer der Tests erfolgreich war, bricht die Methode zur Schnitterkennung ab. Daß keiner der Tests true ergibt, ist nicht möglich.

4. Fazit

Im Verlauf unseres Projekts vertieften wir unsere Kenntnisse im Bereich 3D-Programmierung und sammelten Erfahrung mit den Bibliotheken AWT/Swing, Java3D und Xerces. Insbesondere das Zusammenspiel von Java3D und AWT/Swing gestaltete sich nicht immer unproblematisch.

Java3D selbst erwies sich im Umgang manchmal schwerfällig bis fehlerhaft. Nicht immer haben wir die Ergebnisse erhalten, die durch das Studium der API-Docs zu erwarten waren. Als Beispiel hierfür sei an dieser Stelle nur die Schnitterkennung auf Grundlage von BoundingPolytopes genannt. Es stellte sich erst nach vielen erfolglosen Versuchen die Gewissheit heraus, daß die Schnitterkennung von Java3D nicht exakt arbeitet. Diese Erkenntnis wurde durch Diskussionsbeiträge in einem Java3D-Forum auf java.sun.com bestätigt. Aus diesem Grund mußten wir eine eigene Schnitterkennung implementieren.
Die Navigation mit der Maus wird in Java3D durch das Behavior-Konzept ermöglicht. Auch hier dauerte es mitunter sehr lange, bis wir das gewünschte Verhalten erhielten. Nicht selten mußten wir dafür den Quellcode der MouseBehavior-Klassen von Sun anpassen.

Unter dem Strich betrachtet können wir aber trotz dieser und andere Unwegsamkeiten sagen, daß die Java3D-API uns sehr die Arbeit erleichtert hat. Insbesondere die vielen Hilfsklassen zur Lösung geometrischer Probleme haben uns einiges an Arbeit abgenommen, so daß wir uns größtenteils auf das konzentrieren konnten, worauf wir bei dem Projekt das Hauptaugenmerk gelegt haben: Die Benutzerfreundlichkeit. Im Verlauf dieses Projekt mußten wir lernen, daß die Implementierung einer benutzerfreundlichen Bedienoberfläche sich sehr viel schwieriger gestaltet, als man es sich oftmals vorstellt. Wir haben verschiedene Konzepte und Lösungen ausprobiert und wieder verworfen, ehe wir zu der jetzigen Funktionalität gekommen sind. Da aber jede Fehlentscheidung im Bereich der Implementierung der Funktionalität sehr kostspielig ist, haben wir frühzeitig in langen Diskussionen für und gegen verschiedene konkurrierende Konzepte abgewogen.

Eines der größten Probleme bei diesem Projekt waren jedoch die vielen kleinen und mitunter subtilen Bugs, die sich meistens erst nach einiger Zeit offenbart hatten. Zwar können durch klare und strikt modulare Programmierung viele Fehler von vornherein ausgeschlossen werden. Viele Bugs stellen sich aber erst im wirklichen Umgang mit dem Programm heraus und sind Ergebnis von mehreren zusammenspielenden Faktoren, an die man beim Implementieren einzelner Programmteile niemals denken würde. Daher stellte das stetige Ankämpfen gegen die fortwährend wachsende Bug-Liste eine der zeitraubensten Tätigkeiten während des Projektes dar.

Abschließend ein Screenshot des Programms mit einer geladenen Szene in Vollbildansicht, siehe Abbildung 22.

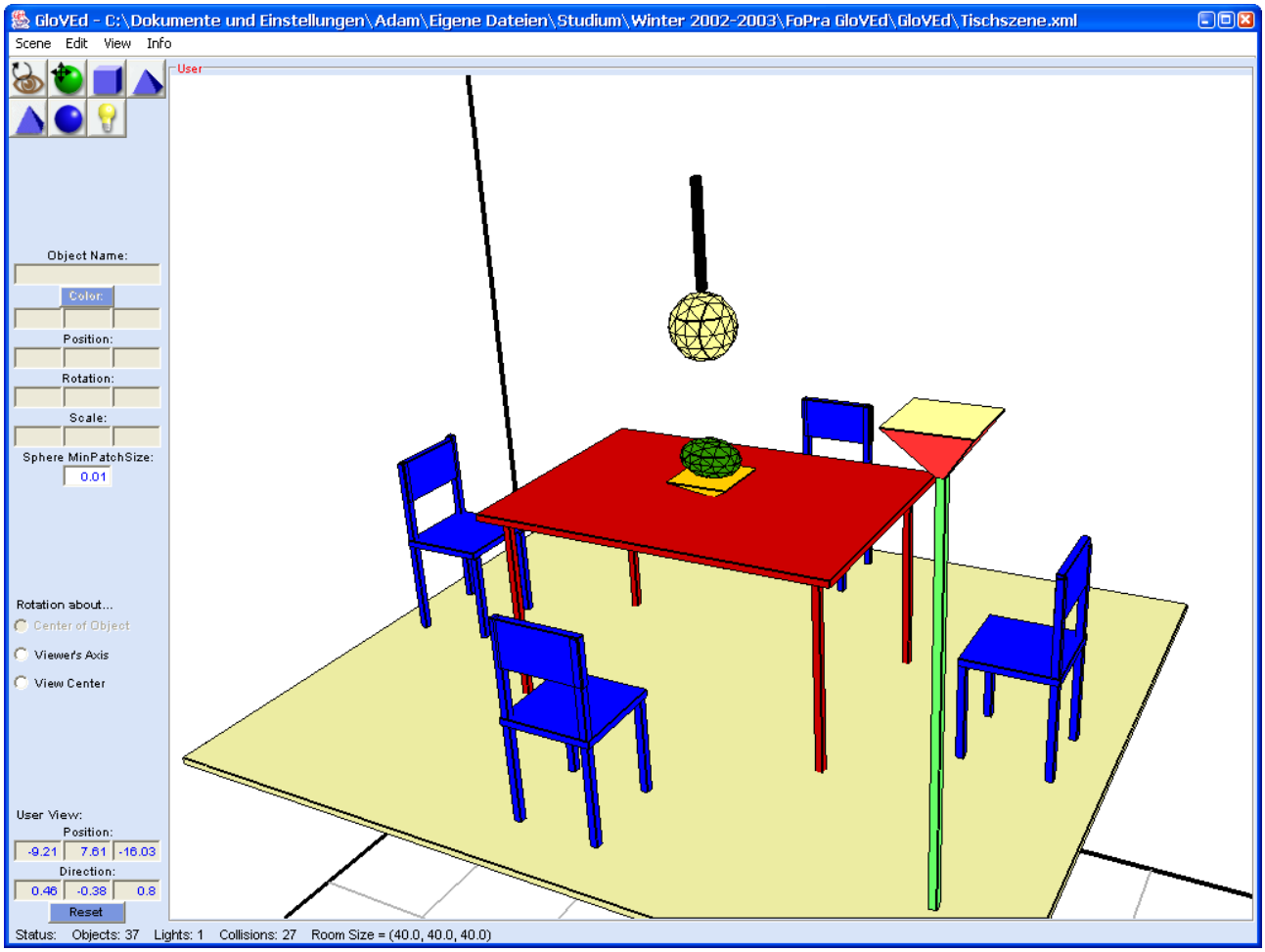

Abbildung 22