Wintertraining – Virtueller Bootssimulator für die Schubschiffahrt

Philipp Bolle, Holger Hees, Tobias Maechler, Peter Wasser

1. Zielstellung

In einer ersten Phase wurde in den Jahren 1995/96 ein Simulator für Motorboote unter Federführung der Projektlogistik GmbH Berlin entwickelt. Aufbauend auf den Erkenntnissen aus der Umsetzung dieses Projektes entstand im Jahr 1997 der Gedanke, einen „Virtuellen Bootssimulator für die Schubschiffahrt" zur Ausbildung von Bootsführern zu realisieren. Damit sollte sichergestellt werden, daß eine Ausbildung auch in den Wintermonaten möglich ist. Weitere Aspekte waren die entfallenden Kosten für den Kraftstoff und die Betriebskosten des Schubkahnes, sowie die Möglichkeit, Extremsituationen zu üben ohne Schäden hervorzurufen. Innerhalb ihres Praxissemesters bzw. ihrer Studienausbildung wurden zwei Studenten des Fachgebietes Wirtschaftsinformatik in die Aufgabenstellung eingebunden. Ihre Aufgabe war insbesondere die programmtechnische Umsetzung des Projektes.

2. Anforderungen an die Softwarelösung

Folgende Kriterien mußten bei der Entwicklung der Softwarelösung berücksichtigt werden und wurden auch im wesentlichen bei der bisher vorliegenden programmtechnischen Lösung berücksichtigt.

a) Die Bewegungsabläufe sollten in einer dreidimensionalen Welt erfolgen.
b) Die Simulation von externen Einflüssen wie Wind, Strömung und Gegenverkehr sollte berücksichtigt werden.
c) Eine Integration aller Schiffahrtsregeln, z. B.:
   - Fahren zwischen Bojen
   - Durchfahren von Brücken
   - Vorfahrtsregeln

   mußte enthalten sein.
d) Eine Kollisionsbehandlung, z. B.:
   - zwischen Boot und Brücke
   - zwischen Boot und Hafen
   - zwischen Boot und Untiefen
   - zwischen Boot und Booten

   mußte realisiert werden.

Weiterhin wurden realisierbare aber hohe Ansprüche an die möglichst naturgetreue grafische Umgebung gestellt (Gelände, Himmel, Bäume, Häuser).
Bild 1 zeigt, daß diese Ansprüche annähernd erfüllt werden, weitere Entwicklungsarbeiten aber unbedingt erforderlich sind.

Bild 1: Darstellung der Landschaft mit einer gewissen Detailtreue

3. Hardwareseitige Mindestanforderungen und Entwicklungstools

Als Mindestanforderungen wurden von uns ein PC mit Intel-Pentium 233 MMX Prozessor, sowie eine Open GL Grafikkarte mit 8 MB Speicher angesehen. Der Hauptspeicher des Rechners sollte mindestens 64 MB betragen und bei den Festplatten wäre die Wahl von schnellen SCSI-Festplatten von Vorteil. Als Idealrechner empfehlen wir einen PC mit Intel Pentium II 333MHz MMX, einer ELSA Gloria XL Grafikkarte und 128 MB Arbeitsspeicher. Die notwendigen softwareseitigen Hilfsmittel sind Autodesk 3D-Studio Max, World ToolKit und ein C-Entwicklungswerkzeug.

4. Ziel der Programmentwicklung

Bei einer Animation spricht man von flüssigen Sequenzen, wenn eine Framerate von 12 –15 Frames/s erreicht wird. Das entwickelte Programm sollte eine Framerate von mindestens 5 Frames/s realisieren. Ein Schubschiffsimulator beinhaltet im wesentlichen langsame Bewegungen, so daß die angestrebte Framerate als akzeptabel anzusehen ist. Bei einem Dualprozessor Pentium II-PC wäre eine Steigerung der Framerate auf mindestens das Doppelte möglich. Ein weiteres Ziel der Programmentwicklung war es, auf die Portabilität der Softwaremodule zu achten. Durch die Nutzung der 3D Bibliothek von Word ToolKit wurde von uns sichergestellt, daß das von uns entwickelte Programm auf fast alle Plattformen portierbar ist. Mit der 3D-Schnittstelle Open GL von World ToolKit ergeben sich wenig Probleme bei der Übertragung der Programmierergebnisse auf andere Systeme.

5. Programmumsetzung

Bei der Realisierung gingen wir in folgenden Schritten vor:

- Erfassung der Funktionalität der C-Bibliothek
- Auswahl eines Programmierstandards
- Benutzung es Mappingverfahrens
- Überlegungen zur Kollisionbehandlung
- Anwendung der Kollisionskonzepte

Die Funktionalität der C-Bibliothek war schnell erschlossen. Um die Performance zu steigern entschlossen wir uns, die Polygonanzahl zu verringern. Dafür fanden wir zwei Lösungen:

1) in 3D-Studio Max gibt es eine Option, die 3D Objekte ohne große Qualitätsverluste herunterrechnet. So gelang es uns, die 3D-Landschaft auf ein Viertel der Polygone zu reduzieren.

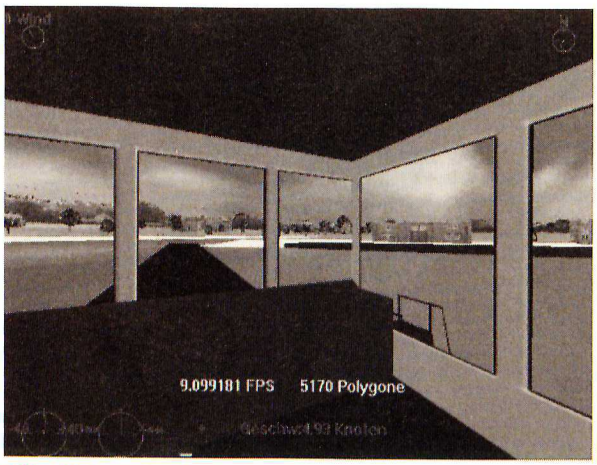

Bild 2: Blick aus dem Schubverband

2) mit Benutzen der Funktion „Level of Detail" wurde es uns möglich, das automatische Austauschen von Objekten zu programmieren. Dies bedeutet, falls ein Schiff oder ein anderes Objekt eine große Entfernung zu betrachten hat, wird es durch ein schnell zu renderndes Objekt ersetzt. So sind unsere Boote bei großer Entfernung einfach Klötze und in der Nähe des Betrachters detaillierte Schiffe.

World ToolKit unterstützt die beiden 3D-Standards Direct 3D und Open GL. Wie bereits erwähnt, haben wir den Standard Open GL genutzt, da somit die Portierbarkeit auf andere Plattformen gegeben ist.
Ein weiters Verfahren zur Verringerung der Polygone (Dreiecke zur Zusammensetzung von Körpern) ist das Mappingverfahren. Dabei werden z. B. aus einem Quader eine Lagerhalle „gebaut" und auf die Seitenflächen dieses Quaders ein Bild der Außenhülle des Lagers, sowie auf das Dach Bilder von Ziegeln projeziert. Eine so erstellte Lagerhalle besteht aus 10 Polygonen (jeweils zwei für die Flächen oben, rechts, links, vorne und hinten). Ohne das Mappingverfahren benötigt man für eine derartige Lagerhalle mit einer Tür und zehn Fenstern (ohne Fensterkreuze) 32 Polygone.

Zur Kollisionsbehandlung stellten wir unterschiedliche Überlegungen an. Um einen kleinen Einblick in die Kollisionsbehandlung zu bekommen, werden wir verschiedene Möglichkeiten vorstellen.

World ToolKit kennt folgende Verfahren:

Kollision zwischen
1. nur Polygonen (ist sehr schnell)
2. Polygon und Objekt (intern prüft der Rechner jedes Polygon des Körpers mit dem einzelnen Polygon)
3. Objekten (sehr langsam, da jedes Polygon mit jedem Polygon des anderen Objektes verglichen wird)
4. Den Boxen, die um ein Objekt gezeichnet werden können, so daß jedes Polygon innerhalb der Box ist (ist sehr schnell, aber es hat den Nachteil, daß keine genaue Kollision stattfinden kann, außer bei Körpern die Quader sind).

6. Anwendung der Kollisionskonzepte

Kollisionen können von vornherein nur zwischen Booten, bei Landberührung bzw. Untiefen stattfinden. Um nicht die Kollision zwischen dem ganzen Land durchzuführen (Land besteht aus sehr vielen Polygonen), entschlossen wir uns, das Wasser zu strukturieren. Man unterscheidet zwischen dem befahrbaren und dem unbefahrbaren Wasser, wobei als Voraussetzung gilt, daß sich beide Wasserfunktionen optisch für den Betrachter nicht unterschiedlich absetzen dürfen.
Tests zwischen untiefem Wasser (hier als Objekt) und Boot ergaben noch zu große Verzögerungen. Nach einer Überarbeitung des Konzepts kamen wir zu der Lösung, daß eine Kollision nur zwischen einem vorderen und hinteren Polygon des Bootes, sowie mit dem untiefen Wasserobjekt getestet werden muß. Durch diese Variante erhielten wir die erwünschte Geschwindigkeit.

Kollision zwischen Boot und Hafen

Beim Hafen mußten wir ein anderes Konzept anwenden, da ein genaues Anlegen notwendig ist. Aufgrund dessen, daß ein Hafen aus 3 Quadern besteht, entschlossen wir uns für das Konzept Kollision von Boxen. Es befinden sich drei Boxen um den Hafen und eine um den Schiffskörper. Zusätzlich wird diese Abfrage nur durchgeführt, wenn die Koordinaten des Bootes auch wirklich in den Bereich des Hafens kommen. So erreichten wir, daß das Programm zwischen Kollision mit Untiefen oder mit dem Hafen unterscheidet und entsprechend handelt.

Kollision zwischen Boot und Brücke

Bei dieser Variante blieb uns nur die Möglichkeit, die Kollision zwischen jedem Polygon der Brückenpfeiler und des Bootes zu prüfen. Um diesen Vorgang zu optimieren, wird diese Kollisionsroutine erst durchgeführt, wenn sich das Boot in der Nähe der Brücke befindet. Auch wird die Kollision zwischen Brücke und Bootsführerhaus nur über die Höhe des Bootsführerhauses durchgeführt. So umgingen wir das Problem einer zwei-

ten Kollisionsabfrage, indem wir das Abfallen der Framerate verringerten, die beim Durchfahren der Brücke entsteht.

7. Anwendung des Autoboot Konzepts

Das Pflichtenheft forderte, daß sich auf dem See ***unabhängig*** fahrende Boote bewegen. Ziel war es, daß sich diese Boote an die Wasserverkehrsregeln halten und somit jedem Zusammenstoß aus dem Wege gehen können. Um das zu verwirklichen, entschlossen wir uns, die Boote auf festgelegten Strecken fahren und jedes Boot seinen ihm zugewiesenen Fahrweg für andere blockieren zu lassen. Durch dieses Konzept wurde erreicht, daß die Boote die begrenzte Rechenzeit kaum überschritten.

8. Einschätzbarkeit des weiteren Entwicklungsbedarfes und der Erweiterbarkeit

Das Programm ist sehr modular aufgebaut. Das bedeutet, daß durch leichte Veränderungen völlig neue Programme entstehen können. So könnten wir uns z. B. vorstellen, daß man die Steuerungsroutine und das 3D Objekt Boot austauscht und in einen Segelsimulator umwandelt. Obwohl World ToolKit nicht objektorientiert ist, bemühten wir uns das ganze Programm mit globalen Variablen (z. B. Wind) zu versehen. Auf diesem Wege wäre es auch möglich, später eine ganze Klasse in einer virtuellen Wasserlandschaft auszubilden. Erforderlich wäre dann, einen Ausgleich der ***gesamten*** globalen Variablen (wie Strömung, Wind, Lichtverhältnisse) sowie der Position und Orientierung (in welche Richtung das Boot steht) an alle anderen beteiligten Rechner zu schikken. Unsere bisherigen Resultate waren u. a. auf der 3. Deutsch-Polnischen Logistikkonferenz zu sehen.

Verfasser

Philipp Bolle (Student)
Holger Hees (Student)
Dipl.-Ing. Peter Wasser
Technische Fachhochschule Wildau
Fachbereich Ingenieurwesen/Wirtschaftsingenieurwesen
Tel. (0 33 75) 508-925

Dipl.-Ing. (FH) Tobias Maechler
Projektlogistik GmbH
Liebermannstraße 27-37, 13088 Berlin
Tel. (0 30) 960 72 88