EIN PROTOTYP ZUR IMMERSIVEN BETRACHTUNG UND INTERAKTIVEN MANIPULATION RÄUMLICHER OBJEKTE

Jan Linxweiler, Sebastian Geller, Jan Zimmermann

Institut für Computeranwendungen im Bauingenieurwesen,
Technische Universität Braunschweig

Kurzfassung: *Im Rahmen dieser Arbeit wurde der Prototyp eines immersiven Werkzeugs entwickelt, um dem Ingenieur das Modellieren innerhalb einer Virtual-Reality-Umgebung zu ermöglichen. Diese bietet dem Betrachter neben einer räumlichen Darstellung mit echter Tiefeninformation zusätzlich die Möglichkeit, sich innerhalb der Szene zu bewegen und Objekte zu manipulieren. Dem Ingenieur wird so ein intuitives Arbeiten ermöglicht. Insbesondere werden Aspekte der Implementierung erläutert, die für eine Darstellung und Interaktion innerhalb einer VR-Umgebung von Bedeutung sind.*

1 Einleitung

Bereits 1965 veröffentlichte Sutherland seine Vision von dem „Ultimate Display“ [1] und gab somit den Anstoß für das, was heute als *Virtual Reality (VR)* bezeichnet wird. Diese virtuelle Realität definiert Brooks [5] als wirkungsvolle Erfahrung des Beobachters, in eine reaktive virtuelle Welt einzutauchen. Das „Eintauchen“ wird auch als Immersion bezeichnet.

Der Grad der Immersivität kann dabei durchaus variieren. Für ein hohes Maß an Immersion ist besonders der visuelle Eindruck der virtuellen Welt von Bedeutung [6]. Dazu sollte dem Betrachter eine möglichst realistische dreidimensionale Darstellung geboten werden, in der er sich zusätzlich bewegen kann. Ferner kann der immersive Eindruck erhöht werden, indem der Benutzer die Möglichkeit zur Interaktion mit der virtuellen Welt erhält.

Unter dieser Zielsetzung wurde der Prototyp eines immersiven Werkzeugs zur Gittergenerierung für baumstrukturierte Eulergitter entwickelt. Da die Modelle zur Gittergene-

rierung und Grafik bekannt sind, wurde im Hinblick auf das Eintauchen in das reale Szenario besonderer Wert auf die Umsetzung der perspektivischen Darstellung und des Trackings gelegt. Nach einem kurzen Absatz über die verwendeten Technologien wird dies im Folgenden näher beschrieben.

Als Testumgebung stand das VR-Labor des Instituts für Computeranwendungen im Bauingenieurwesen (CAB) der TU Braunschweig zur Verfügung, wo die Möglichkeit einer stereoskopischen Darstellung mit Hilfe einer Projektionsleinwand besteht. Zusätzlich steht im VR-Labor ein Tracking System [12] zur Verfügung.

2 Aufbau und Umsetzung

Die Entwicklung der Anwendung lässt sich in drei Teilbereiche gliedern. Der erste Teil besteht aus einer grafischen Benutzeroberfläche basierend auf der Qt-Klassenbibliothek [8]. Den zweiten Teilbereich der Anwendung stellt die 3D-Visualisierung des Modells mit VTK [10] dar. In einem weiteren Schritt wurde zur Verwirklichung der Immersivität und Interaktivität eine Anbindung des Tracking-Systems *Flock of Birds* an die Visualisierung vorgenommen.

Qt ist eine logische und gut strukturierte plattformunabhängige Klassenbibliothek der Firma Trolltech [8], die sich unter Linux mittlerweile zu einem De-facto-Standard in der C++ GUI-Programmierung (GUI = **G**raphical **U**ser **I**nterface) entwickelt hat.

Das *Visualization Toolkit* (VTK) ist ein plattformunabhängiges Open-Source-Softwarepaket für 3D-Computergrafik, Visualisierung und Bildverarbeitung. VTK ist in C++ implementiert und bietet selbst keine Userinterface-Komponenten an, kann aber in gängige GUI-Bibliotheken wie beispielsweise Qt oder MFC integriert werden.

Die Funktionalität zur Gittergenerierung ist Bestandteil der vorhandenen Klassenbibliothek des CAB.

3 Visualisierung

Zur Darstellung dreidimensionaler Objekte mit Hilfe zweidimensionaler Anzeigegeräte erfolgt eine Abbildung (Projektion) der räumlichen Objekte auf eine Projektionsebene. Als Projektionsart wird oft die bekannte Zentralperspektive gewählt. In der Computergrafik bedient man sich dafür eines synthetischen Kameramodells, bei dem die Betrachtungsparameter in Analogie zu einer Pseudo-Kamera hergeleitet werden. Die für eine Abbildung notwendigen Parameter sind im Wesentlichen die Position, Lage und Aufnahmerichtung sowie ein Punkt innerhalb der Szene, auf den die Kamera fokussiert ist [7].

Durch diese Parameter wird eine Sichtpyramide *(View Frustum)* definiert, welche in Abb. 1 dargestellt ist. Der typische Anwendungsfall, für den auch das VTK Kameramodell konzipiert wurde, ist die Projektion räumlicher Objekte auf einen Computer-

bildschirm. Dabei wird im Allgemeinen davon ausgegangen, dass der Betrachter auf die Mitte des Darstellungsfensters blickt. Das Zentrum (CW) des *View Windows* und der Blickpunkt (VRP) des Betrachters sind somit identisch. Man spricht in diesem Fall auch von einer symmetrischen Sichtpyramide, da die Halbwinkel des Blickfeldes *(FOV)* jeweils gleich groß sind. Bei der typischen Bildschirmprojektion bleibt diese Symmetrie immer erhalten, da sich die Projektionsfläche stets mit der Kamera mitbewegt und immer dieselbe Form hat. Der Bildschirm stellt also bildlich gesprochen ein bewegliches Fenster auf die Szene dar, das innerhalb des virtuellen Raumes bewegt werden kann.

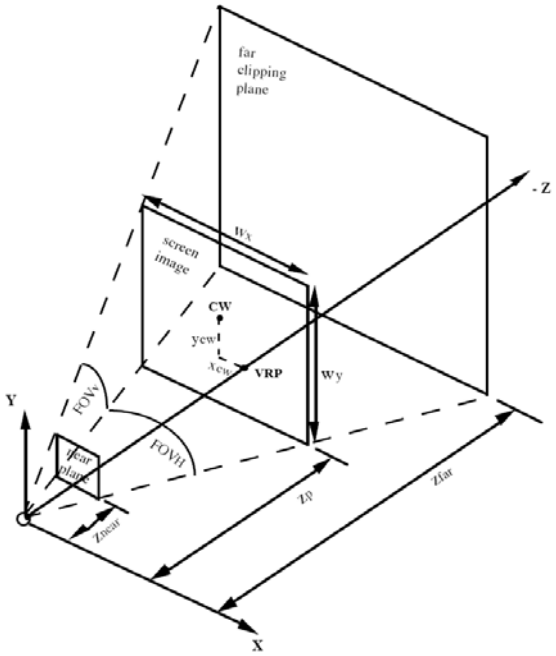

Abb. 1: Sichtpyramide für eine perspektivische Projektion

In einer Virtual-Reality-Umgebung jedoch muss eine andere Form der Projektion gewählt werden. Während sich der Betrachter innerhalb der Szene bewegen kann, ist das Darstellungsfenster in diesem Fall feststehend. Das heißt, dass der Blickpunkt (VRP) der Kamera bzw. des Betrachters nur selten identisch mit dem Zentrum des Darstellungsfensters (CW) ist (Abbildung 2).

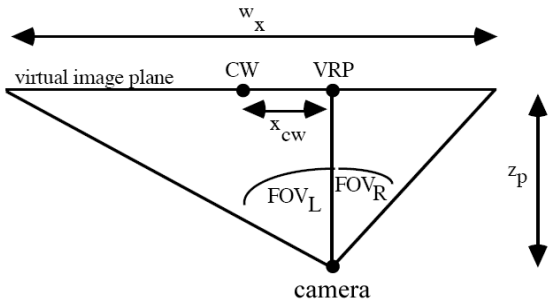

Abb. 2: Off-Center-Projektion

Die Folge ist eine asymmetrische Sichtpyramide, deren Form sich mit der Bewegung des Betrachters ändert. Man spricht in diesem Fall auch von einer Off-Center-Projektion. Zusammen mit der Stereoprojektion kann dem Betrachter auf diese Weise der Eindruck vermittelt werden, sich innerhalb der virtuellen Umgebung zu bewegen, während die Objekte innerhalb der Szene unbeweglich bleiben. Das VTK eigene Kameramodell musste daher erweitert werden, um den Anforderungen gerecht zu werden. Zu diesem Zweck wurde die Klasse `vtkCABFlockCamera` implementiert, die eine perspektivische Darstellung in Abhängigkeit der Position des Betrachters ermöglicht.

4 Tracking

4.1 Realisierung

Neben einer VR-Projektionswand stand das Tracking-System *Flock of Birds* zur Realisierung des Prototyps zur Verfügung. Das System ist in der Lage, Positions- und Orientierungsmessungen mit einer Rate von 144 Messungen pro Sekunde durch-zuführen. Die ermittelten Daten können anschließend sowohl für das *Head-Tracking,* als auch für das *Tool-Tracking* eingesetzt werden. Beim *Head-Tracking* wird mit Hilfe des Sensors die Position des Benutzers bzw. seines Kopfes ermittelt. Entsprechend wird mit Hilfe des *Tool-Trackings* die Position eines Gerätes bestimmt. Die Reichweite des Systems beträgt ca. 3 m. Zudem kann die Rotation um jede der drei Achsen (Eulerwinkel) im Bereich von 0° bis 360° festgestellt werden.

Seitens des Herstellers wird eine Bibliothek [12] zur Verfügung gestellt, die eine Kommunikation mit dem Tracking System *Flock of Birds* ermöglicht. Für die Integration des Tracking Systems in die Applikation wurden verschiedene Klassen entworfen, die die Funktionalität dieser Bibliothek kapseln und im Kontext von VTK nutzbar machen.

- Die Klasse `vtkCABFlockTracker` repräsentiert das eigentliche *Flock of Birds*-Tracking-System. Im Kontext dieser Klasse findet im Wesentlichen die Kommunikation mit dem System statt.
- Die Klasse `vtkCABTrackerTool` steht stellvertretend für einen einzelnen Sensor des Gesamtsystems. Jeder Sensor besitzt ein Attribut vom Typ `vtkTransform`, das die Position und Orientierung des Sensors relativ zum Transmitter angibt.
- Die Klasse `vtkCABFlockCamera` hat die Aufgabe, die Projektion anhand der Sensordaten an die jeweilige Position des Benutzers anzupassen.

- Der Typ `vtkCABFlockDevice` stellt einen generischen Typ zur grafischen Präsentation eines Objekts auf Basis der Sensordaten innerhalb von VTK dar. Der Typ verhält sich dabei wie ein Darstellungsobjekt, dessen Position und Orientierung abhängig von einem Objekt des Typs `vtkCABTrackerTool` ist. Er stellt somit die Basisklasse für Datentypen dar, die beispielsweise eine *Spacemouse* oder einen Datenhandschuh repräsentieren.
- Die Klasse `vtkCABFlockPointer` ist eine konkrete Implementierung des Typs `vtkCABFlockDevice`. Sie stellt eine Möglichkeit zur Interaktion mit der VR-Umgebung mittels *Spacemouse* dar.

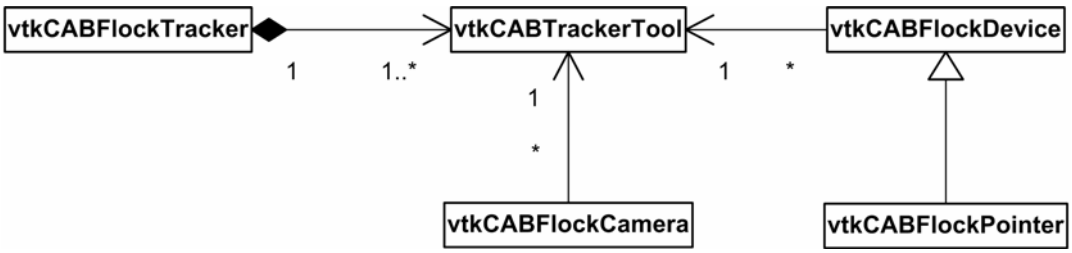

Abb. 3: UML-Darstellung der Trackingklassen

Die Klasse `vtkCABFlockTracker` stellt das eigentliche Tracking-System dar und nimmt somit eine zentrale Rolle ein. Um eine Kommunikation mit dem *Flock of Birds* zu ermöglichen, bietet die Klasse mehrere Parameter, die in erster Linie der Konfiguration dienen. Notwendige Einstellungen sind beispielsweise die Angabe der verwendeten Schnittstelle und die Datenübertragungsrate. Anschließend kann das System durch den Aufruf der Methode `StartTracking()` initialisiert werden. Mit dieser Initialisierung wird das System überprüft und die angeschlossenen Sensoren identifiziert. Stellvertretend für jeden Sensor hält die Systemklasse nach erfolgreicher Initialisierung ein Objekt vom Typ `vtkCABTrackerTool` vor. Anschließend beginnt der *Flock of Birds* mit einer kontinuierlichen Messung. Von der Klasse `vtkCABFlockTracker` wird dazu ein neuer Thread gestartet. Innerhalb dieses Threads wird kontinuierlich eine Schleife zum Einlesen und Verarbeiten der Sensordaten durchlaufen.

Für jeden Sensor wird dabei die Position und Orientierung relativ zum Transmitter bestimmt. Diese müssen zunächst in das Koordinatensystem der virtuellen Umgebung übertragen werden. Anschließend wird aus den Koordinaten und der Orientierung eine Transformationsmatrix erstellt, die an das jeweilige Objekt vom Typ `vtkCABTracker-Tool` übergeben wird. Diese Objekte stellen die Transformationsmatrix daraufhin als ein Objekt vom Typ `vtkTransform` zur Verfügung, so dass sie innerhalb von VTK verwendet werden kann. Der Tracking-Vorgang ist in Abbildung 4 in Form eines Interaktionsdiagramms verdeutlicht.

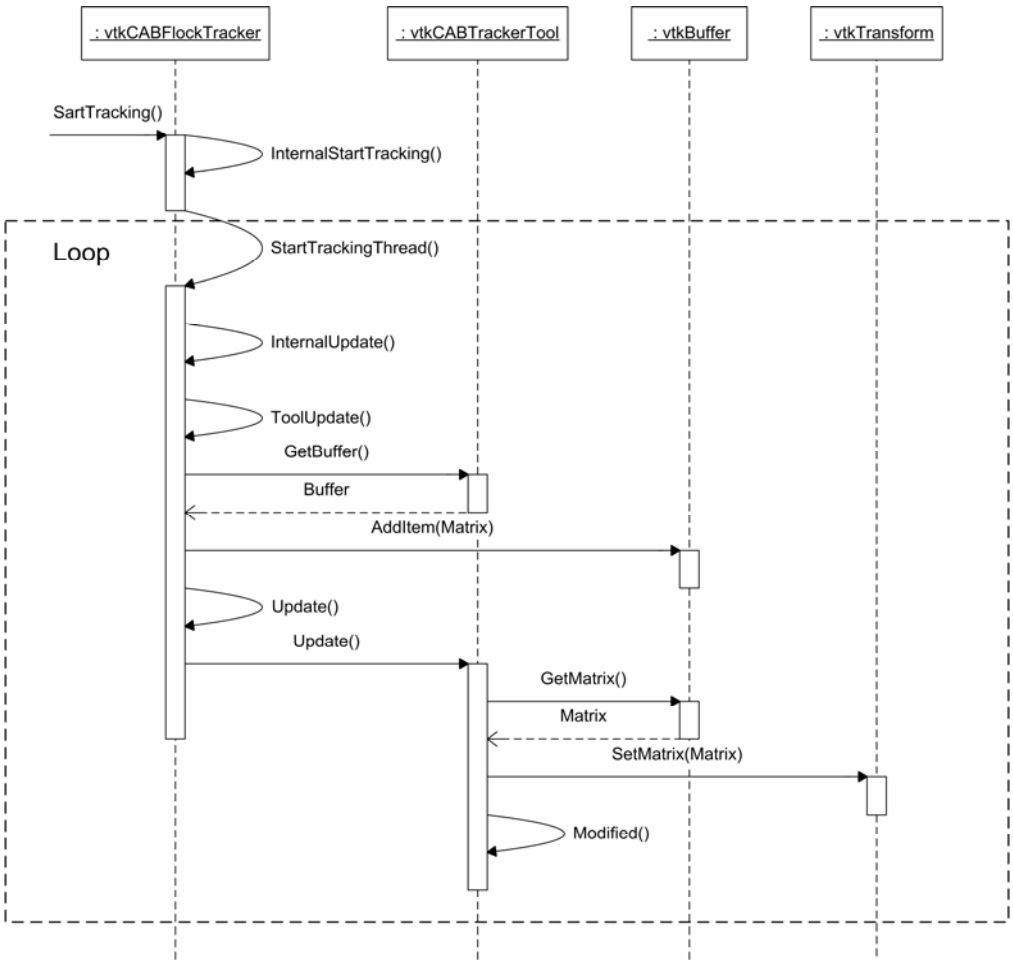

Abb. 4: Interaktionsdiagramm des Trackingprozesses

4.2 Kalibrierung

Elektromagnetische Tracker wie der *Flock of Birds* reagieren sensibel auf metallische Gegenstände und elektronische Geräte. Der sich daraus ergebende Fehler setzt sich zusammen aus einem statischen und einem dynamischen Anteil. Nach der Definition von Azuma [3] treten statische Fehler auch dann auf, wenn sich der Sensor und die Objekte in der Umgebung fix an einer Stelle befinden. Dynamische Fehler hingegen treten erst dann in Erscheinung, wenn der Sensor in Bewegung ist. Es existieren allerdings Ansätze, die es ermöglichen, den statischen Fehler zu korrigieren. Bislang gibt es jedoch noch keine Möglichkeit zur Korrektur des dynamischen Fehlers.

Im Rahmen dieser Arbeit wurde zur Korrektur des statischen Fehlers ein an der Universität von Illinios in Urbana (UIUC) entwickeltes Verfahren eingesetzt, dass auf einem Ansatz von Raab et al. [11] basiert. Dieser Ansatz geht davon aus, dass die Fehler, die aus den Verzerrungen des magnetischen Feldes resultieren, mit Hilfe eines höherwertigen Polynoms angenähert werden können. Die UIUC stellt eine Open-Source-Bibliothek zur Verfügung, mit deren Hilfe eine Kalibrierung des Tracking Systems vorgenommen wurde. In Abbildung 5 ist das verzerrte Gitter aus den eingemessenen Referenzpunkten dargestellt.

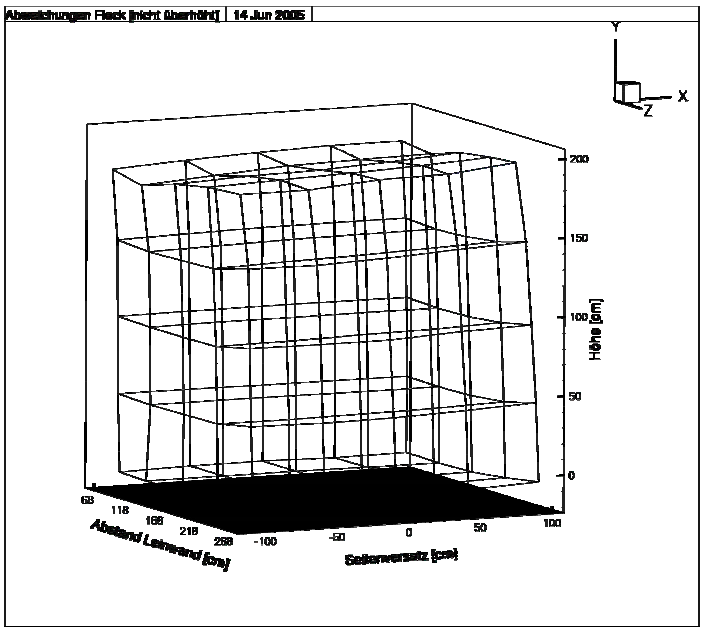

Abb. 5: Darstellung des eingemessenen Referenzgitters

4.3 Synchronisation

Innerhalb der Anwendung existieren der GUI-Thread auf Basis von Qt und der Thread, der für das Auslesen der Sensordaten zuständig ist. Diese Threads müssen synchronisiert werden, da von beiden auf die Transformationsmatrizen der Sensoren zugegriffen wird. Ansonsten kann dies zu Inkonsistenzen führen, deren Folge z.B. unerwünschte Darstellungen oder gar Programmabstürze sind. Ein typischer Ansatz zur Lösung dieses Problems wäre die Verwendung von *Mutexen*. Leider eignet sich diese Technik nicht dazu, einen GUI-Thread mit einem Nicht-GUI-Thread zu synchronisieren, da diese die Ereignisschleife des GUI-Threads und somit die Benutzerschnittstelle „einfrieren“ können. Das Ereignismodell von Qt ermöglicht es daher, eigene Ereignisse zu definieren, und diese an den GUI-Thread zu senden. Die dafür zu verwendende Methode ist dabei Thread-sicher, das heißt, sie ist mit dem GUI-Thread synchronisiert.

Innerhalb der Anwendung wird diese Methode daher von der Klasse `vtkCABTracker-Tool` verwendet, um anderen Objekten, wie z.B. denen vom Typ `vtkCABFlockCame-ra`, neue Sensordaten zu übermitteln.

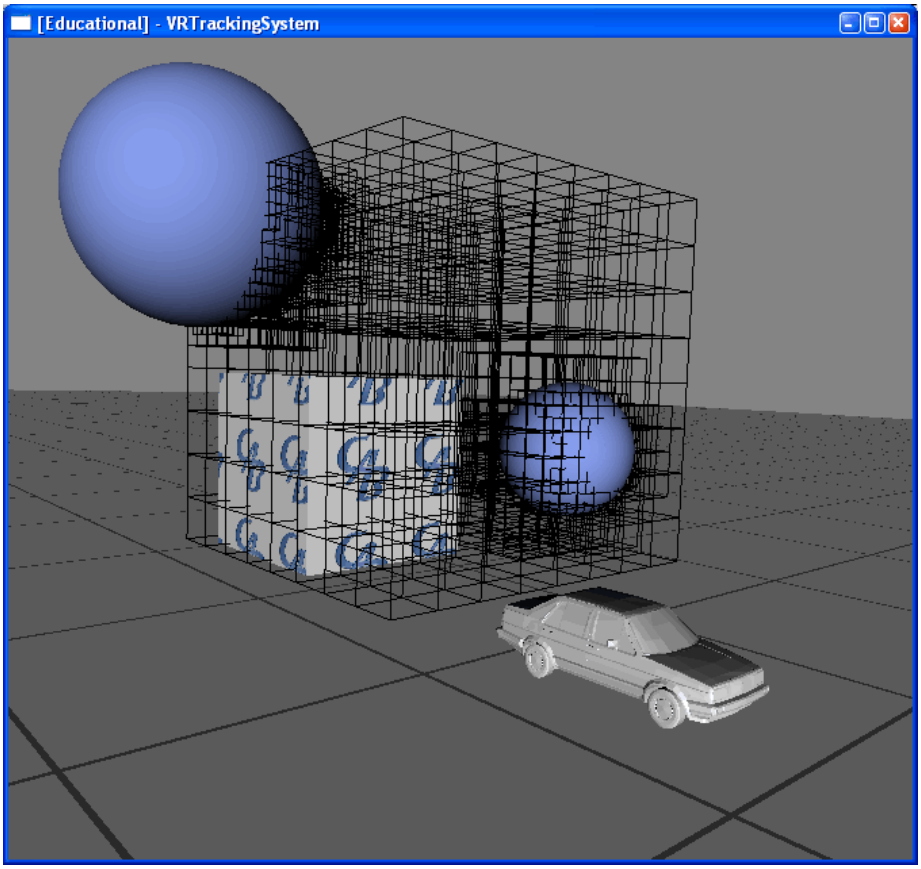

Abb. 6: Darstellung einer VR-Szene im Gittergenerierungsprozess

5 Ergebnis

Im Rahmen dieser Arbeit wurde der Prototyp einer Anwendung zur Integration des Prozesses der Gittergenerierung in eine interaktive virtuelle Umgebung realisiert. Abbildung 6 zeigt eine mögliche Szene, in der der Ingenieur durch interaktive Kommunikation mit der Simulationsumgebung Objekte manipulieren und verschieben kann und somit den Prozess der Gittergenerierung dynamisch beeinflussen kann. Im Vordergrund der Entwicklung standen dabei immersive Darstellungs- und Interaktionstechniken, die den Anwender bei den Modellierungsaufgaben unterstützen. Die Grundlage der Arbeit bildete die CAB-Klassenbibliothek, die bereits die notwendigen Datentypen und Algorithmen für die geometrische Modellierung und Gittergenerierung zur Verfügung stellte sowie die Grafikbibliothek VTK.

Da es sich bei der entwickelten Anwendung bislang lediglich um einen Prototypen handelt, bietet dieser vielfache Ansatzmöglichkeiten für weitere Arbeiten.

Für den Simulationsprozess aber auch für die Visualisierung könnte auf den institutseigenen Cluster zurückgegriffen werden, um den Rendervorgang und die Simulationsberechnungen zu parallelisieren. Derzeit führen Gitter mit einer hohen Knotenanzahl dazu, dass die Darstellung nicht mehr flüssig angezeigt wird. Dieser Umstand ließe sich durch paralleles Rendern beheben. Momentan ist die Anwendung speziell auf das

Virtual Reality-Labor des CAB ausgelegt. Es wäre jedoch auch denkbar, die Applikation dahingehend zu erweitern, dass sie innerhalb einer CAVE oder mit einem *Head Mounted Display* ausführbar wird.

Um die Immersivität der Applikation zu steigern, gibt es viele Möglichkeiten. Nahe liegend ist zunächst neben der Verschiebung auch eine Rotation und Skalierung mit der *Spacemouse* zu ermöglichen. Für diese Funktionen wäre auch der Einsatz eines Datenhandschuhs vorstellbar, da dieser ein intuitiveres Manipulieren ermöglicht. Schließlich ließe sich ein virtuelles Menü einführen, das sich mit Hilfe der *Spacemouse* bedienen lässt und die üblichen Funktionen der Anwendung darbietet.

Literatur

[1] I. E. Sutherland: The Ultimate Display, Proceedings of IFIP Congress, 1965

[2] Ascension Technology Corporation, Flock of Birds – Installation and Operation Guide, Burlington, 2002

[3] R. T. Azuma: A Survey of Augmented Reality, Hughes Research Laboratories, Malibu, 1997

[4] J. Linxweiler: Ein Prototyp zur immersiven Betrachtung und interaktiven Manipulation räumlicher Objekte, Diplomarbeit, Institut für Computeranwendungen im Bauingenieurswesen, TU Braunschweig, 2005

[5] F. P. Brooks: What's Real About Virtual Reality, Department of Computer Science, University of North Carolina at Chapel Hill, 1999

[6] S. R. Ellis: Nature and Origin of Virtual Environments: A Bibliographic Essay, Computing Systems in Engineering, 1991

[7] W.-D. Fellner: Computergrafik, BI-Wissenschaftsverlag, Mannheim, 1992

[8] Trolltech - Qt, http://www.trolltech.com/products/qt/

[9] V. Kindratenko: Calibration of electromagnetic tracking devices, Urbana, 1999

[10] B. Lorensen, K. Martin, W. Schroeder: The Visualization Toolkit – An Object-Oriented Approach to 3D Graphics, Kitware, 2002

[11] F. Raab, E. Blood, T. Steiner, H. Jones: Magnetic Position and Orientation Tracking System, IEEE Trans. Aerospace and Electronic Systems, 1979

[12] Flock of Birds von Ascension
http://www.ascension-tech.com/products/flockofbirds.php