

FishSwarm

Immersive Interaktion mit einem Fischschwarm in einer virtuellen Umgebung

Thomas Reisenweber [MI5053]

André Knörig [MI5261]

MalteBüchmann [MI7922]

Einleitung

Projektidee

Interaktiver Fischschwarm, animiert in einer Unterwasserwelt. Es soll sich nicht um ein Spiel handeln, die Darstellung und einfache Interaktion stehen im Vordergrund. Auch für den unerfahrenen User soll der Umgang schnell und einfach möglich sein.

Der User befindet sich in der CAVE und kann in einer simulierten Unterwasserwelt mit umher schwimmenden Fischschwärmen interagieren.

Bestandteile

Unterwasserwelt

Der Benutzer befindet sich in einer virtuellen Unterwasserwelt. Er wandelt über den Sandboden wie vor Capri und sieht weit und breit nur - Wasser. In der Ferne tummeln sich kleine Fischis, elegant zu einem Schwarm formiert.

Fische

Ein Schwarm von Fischen bewegt sich tollend durch das triefende Nass.

Interaktion

Das Verhalten des Unterwasserwelt-Water wirkt sich auf seine Umwelt aus: Durch Füttern lockt er den Fischschwarm an, durch wildes Gestikulieren in Kombination mit italienischen Ausrufe vertreibt er sie. Verhält er sich ruhig, so kann er genüßlich dem Treiben zuschauen und sich vom Schwarm umschwirren lassen.

Umsetzung

Schwarmlogik

Abbildung 1:
wenn Schwarmteilnehmer kollidieren wird auf den gesamten Bewegungsvektor ein Anteil zugerechnet, der die betreffenden Teilnehmer im nächsten Bewegungsschritt voneinander trennt.

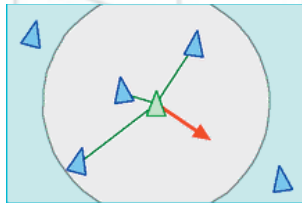


Abbildung 2:
Jede Bewegungsrichtung eines jeden Teilnehmers orientiert sich an der durchschnittlichen Bewegungsrichtung des gesamten Schwarms

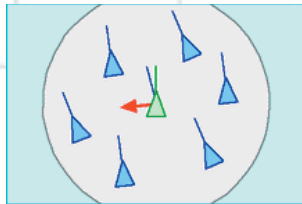
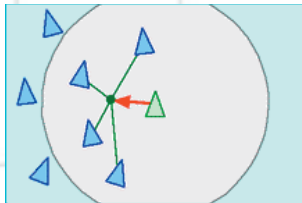


Abbildung 3:
Alle Schwarmteilnehmer streben zu einem bestimmten Anteil auf den Mittelpunkt des Schwarm zu. So wird eine Zusammengehörigkeit der Fische simuliert.



Bei der Schwarmlogik beziehen wir uns auf die von Craig Reynolds (www.red3d.com/cwr/) entwickelten Routinen, die in ihrer Addition ein Schwarmverhalten simulieren. Grundsätzlich werden dabei für jeden Schwarmteilnehmer unterschiedliche Methoden angewandt, die jeweils einen Bewegungsvektor zurückliefern. Die Zusammenführung dieser einzelnen Vektoren liefern insgesamt das Verhalten des Schwarms.

Folgende drei Regeln für die Interaktion der Schwarmteilnehmer untereinander sind besonders wichtig:

räumliche Distanz zwischen den einzelnen Schwarmteilnehmern:

Jeden Berechnungsdurchlauf wird der Abstand zwischen jedem Schwarmteilnehmer mit allen anderen Schwarmmitgliedern berechnet. Wenn eine Kollision zwischen Schwarmteilnehmern erkannt wird, liefert diese Regel für den betreffenden Fisch einen Bewegungsvektor, der die kollidierenden Partner im nächsten Bewegungsprozess wieder voneinander trennt. (vgl. Abbildung 1)

Ausrichtung der Schwarmteilnehmer an der Durchschnittsbewegung des gesamten Schwarms:

Alle Schwarmteilnehmer sind durch die durchschnittliche Fortbewegung des gesamten Schwarms geprägt. Pro Bewegungsschritt wird also die durchschnittliche Bewegungsrichtung aller Schwarmteilnehmer berechnet, an der sich jeder einzelne Fisch orientieren muß. (vgl. Abbildung 2)

Alle Teilnehmer eines zusammengehörigen Schwarms stre-

ben auf den Mittelpunkt des Schwarms zu:

Jeder Schwarmteilnehmer ist durch seine aktuelle Position und seinen aktuellen Bewegungsvektor definiert. Aus dem Durchschnitt der Positionen aller Schwarmteilnehmer ergibt sich der Schwerpunkt des kompletten Schwarms. Um einen Zusammenhalt der Fische untereinander zu gewährleisten, strebt jeder einzelne Fisch zu einem bestimmten Anteil auf diesen berechneten Mittelpunkt zu. (vgl. Abbildung 3)

Mit den drei oben beschriebenen Regeln lässt sich schon ein ziemlich gutes Ergebnis hinsichtlich der Schwarmzusammengehörigkeit und des Verhaltens eines jeden einzelnen Teilnehmer simulieren. Allerdings reicht dies noch nicht aus, um eine befriedigende Simulation eines Fischschwarms zu erreichen, die Simulation erinnert so eher an einen Bienenschwarm. Deshalb sind in unserer Implementation noch weitere Verhaltensregeln festgelegt worden:

der hungrige Fischschwarm verfolgt immer ein Ziel:

Um eine gleichmäßiges Schwimmverhalten zu erreichen, verfolgt jeder Schwarmteilnehmer ein Ziel, das entweder aus dem lockenden Futter besteht, oder einem per Zufall berechneten Ziel, das den Schwarm immer in eine stetige Fortbewegung versetzt. Nur so ist zu erreichen, dass eine flüssige Bewegung jedes einzelnen Teilnehmers im Einklang mit allen anderen Fischen erzielt werden kann.

der Fischschwarm darf sich nur in einem gewissen örtlichen Rahmen bewegen:

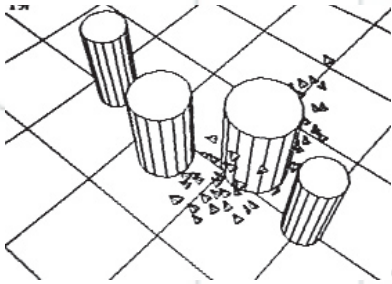


Abbildung 4:
Um die Ziele, mit denen nicht kollidiert werden darf, wird eine Zylindermaske gelegt. Wenn Schwarmteilnehmer in diese Maske dringen, wird ein bestimmter Anteil auf den gesamten Bewegungsvektor zugerechnet, der den kollidierenden Fisch von dem Kollisionszeit „wegtreibt“.

da in der virtuellen Umgebung die räumlichen Ausmaße beschränkt sind, werden den Fischen örtliche Grenzen zugeteilt. Sobald ein Fisch diese Grenze überschreitet, wird sein Bewegungsvektor in die andere Richtung umgelenkt, um so zu sichern, dass der interagierende User den Schwarm auch dann beobachten kann, wenn er ihn nicht durch Futter anlockt, da er sich auch dann immer in seinem Sichtfeld befindet.

der Schwarm darf nicht mit dem interagierenden User kollidieren:

Für den Schwarm können Ziele definiert werden, mit denen die Teilnehmer nicht kollidieren dürfen. Das sind

wie schon erwähnt die einzelnen Teilnehmer untereinander, aber auch zum Beispiel der fütternde User. Dabei wird um das Ziel, mit dem nicht kollidiert werden darf eine zylindrische Maske gelegt. Sobald diese Maske durchschritten ist, wird der Vektor umgelenkt. (vgl. Abbildung 4)

der Schwarm kann erschreckt werden:

Wenn der User den Schwarm mit einer schnellen Handbewegung erschreckt, wird auf die Bewegungsvektoren der Fische ein Bewegungsvektor addiert, der in entgegengesetzter Richtung zu der „erschreckenden“ Quelle ausgerichtet ist.

Fischmodell

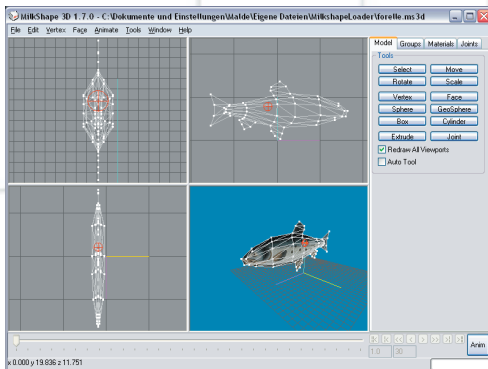


Abbildung 5:
Das Fischmodell im „Milkshape 3D“-Modellierer.



Abbildung 6:
Der fertige Fisch mit Textur.

Import und Darstellung des Fisch-Modells

Um 3D-Modelle einfach in unsere Applikation importieren zu können, wurde das Shareware-Modellierungstool „Milkshape 3D“ gewählt (siehe <http://www.swissquake.ch/chumbalum-soft/>). Mit diesem Tool ist es möglich, 3D-Modelle zu erstellen, zu texturieren und durch ein Skelettsystem zu animieren. Der Aufbau des Dateiformats ist sehr gut dokumentiert und macht es somit leicht, erstellte Figuren zu ändern und in unserer Applikation auszutauschen.

Um eine effiziente Darstellung der Modelle zu gewährleisten, werden diese über Vertex-Arrays gezeichnet. Die aus der Datei importierten Daten liegen jedoch in einem internen Format vor. Beim Einlesen müssen diese zuvor in eine geeignete Struktur überführt werden. Ist dies geschehen, liegen die Daten für die Geometrie, Normalen, Texturkoordinaten und die Zeichenreihenfolge jeweils in einem eigenen Array.

Beim Animieren der Figur müssen nur noch die Geometriedaten entsprechend der Knochenzuordnung transformiert werden.

Probleme

Probleme gab es beim Wechsel zum IRIX-Betriebssystem. Die Milkshape-Dateien wurden nicht mehr richtig eingelesen. Aus diesem Grund wurde nach einigem Probieren darauf verzichtet, die Modelle zur Laufzeit einzulesen. Stattdessen wurde das auf Windows lauffähige Programm angepasst: Es wurden C-Header-Dateien durch das ursprüngliche Programm erstellt, die dann statisch in die FishModel-Klasse eingebunden wurden. Der Nachteil ist natürlich, dass bei Änderungen des Fischmodells die Klasse wieder neu kompiliert werden muss.

Eine Kopie der Milkshape-Spezifikation ist beigefügt (ms3dspec.txt.c).

Umgebung



Abbildung 7

Um den Eindruck einer Unterwassersituation immersiver gestalten zu können, gehen wir von einer Unterwasserwelt in einer Strandungsbucht aus.

Für die gleichmäßige Sandstruktur auf dem Grund bietet sich hierbei eine Bézier-interpolierte Fläche an. Die Grundfläche besteht dabei aus aneinandergereihten Kacheln, die jeweils auf 16 Bézier-interpolierten Punkten basieren. (vgl. Abbildung 7 und Abbildung 8).

Die unendlichen Weiten der Tiefsee vermitteln sich durch eine tiefblaue Hintergrundfarbe und einen diffusen Nebel.

Die Brandungspfähle dienen der besseren Darstellung der räumlichen Tiefe unter Wasser.

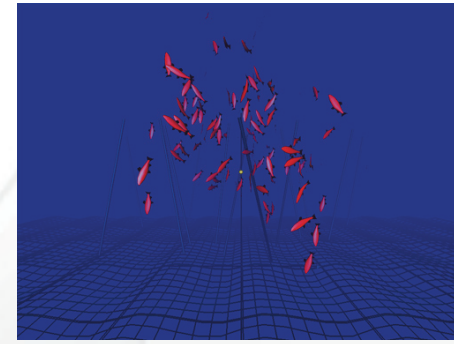


Abbildung 8

Interaktion

Zur Steuerung nimmt der Benutzer eine reflektive Kugel in die Hand, die von einem optischen Tracking-System geortet werden kann. Mit ihr interagiert der Benutzer mit seiner Umwelt:

Streckt er die Hand ruhig von seinem Körper, werden die Fische neugierig und erkennen das Futter in der Hand des Benutzers. Durch schnelle Bewegungen der Hand werden die Fische vertrieben.

Verdeckt der Benutzer das Futter, schwimmen die Unterwasserbewohner unbeeinflusst ihres Weges.

Zusätzlich wird grob der Körper des Benutzer durch die Position der VR-Brille bestimmt, so dass er ein Hindernis für umherschwimmende Fische darstellt.

Software-Architektur

Trennung der VR- und OpenGL-Technik von der Anwendungslogik

Die komplette technische VR-Ansteuerung plus OpenGL-Initialisierung wurde in der DisplayManager-Klasse gekapselt. Ihr wird lediglich bei Programmstart eine Implementierung des VRApplication-Interfaces mitgegeben, an die alle Zeichenbefehle und

Interaktionen über simple, wohldefinierte Methoden überreicht werden. Dadurch lassen sich sowohl die technische Umgebung als auch die eigentliche Anwendung sehr leicht austauschen.

Gegenstände in der virtuellen Welt

Alle Gegenstände in der Welt beerben eine Oberklasse SimObject, in der alle Baseigenschaften verwaltet

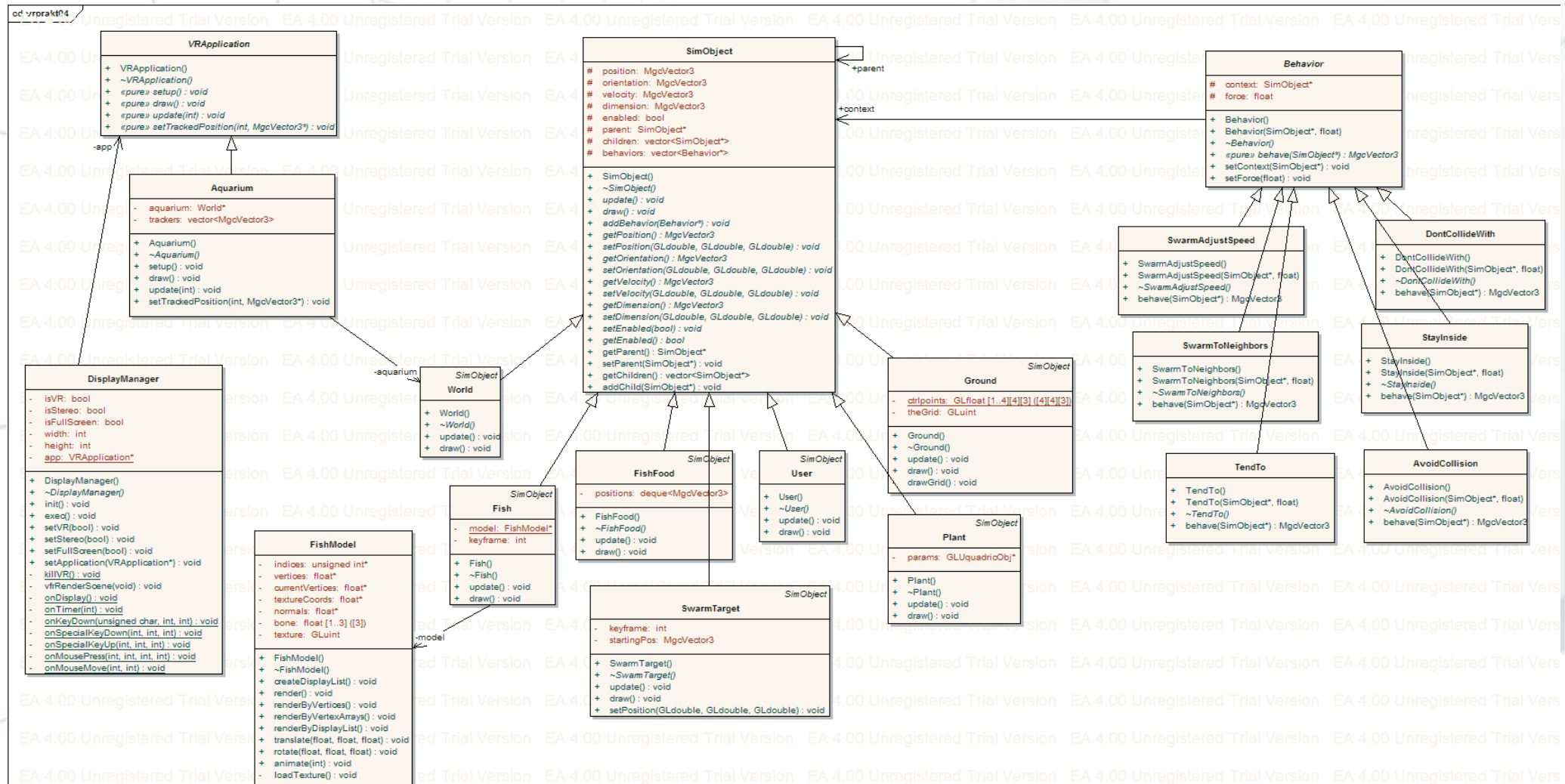


Abbildung 9: UML-Modell der Anwendung

werden. Die rekursive Struktur orientiert sich am *Component-Entwurfsmuster*.

Die Anwendung `Aquarium` baut nun ihre Welt mittels dieser Objekte zusammen, indem sie sie einzeln erzeugt und mit der Container-Klasse `World` hierarchisch strukturiert. Es entsteht eine Art abstrakter Szenengraph, der im Folgenden durch einen einfachen Aufruf bei Gelegenheit (Zeichnen, Aktualisieren) traversiert wird. Wie die Objekte im einzelnen gezeichnet und aktualisiert werden, bestimmen sie selbst.

Während des Ablaufs der Simulation können auf jeder Stufe Veränderungen in der Komposition vorgenommen werden.

Verhalten und Interaktion

Die bisher statischen Simulations-Objekte werden nun mit beliebigen Verhaltensweisen ausgestattet. Einem Objekt können beliebig viele `Behaviors` hinzugefügt werden. Innerhalb eines mitgelieferten Kontext erzeugt eine Verhaltensweise eine Reaktion, die sich in einer Bewegung ausdrückt.

Das Objekt, das mit diesem Verhalten ausgestattet ist, kann die berechneten Reaktionen dann nach Belieben umsetzen.

Fazit

Schwarmlogik

Auch wenn Reynolds Regel-Prinzip zunächst als simpel und komfortabel erschien, erwies sich die Feinabstimmung der einzelnen Regeln als äußerst haarige Angelegenheit. Trotz der von uns zusätzlich implementierten Regeln eignet sich der Algorithmus nur bedingt für die Simulation eines Fischschwarmes, da die Interaktion der Schwarmteilnehmer untereinander sehr hoch betont ist und von außen z.B. durch Zielangaben nur bedingt Einfluß genommen werden kann.

Software-Architektur

Die gute Planung unserer Softwarearchitektur am Anfang des Projektes, hat uns trotz des hohen Aufwandes am Anfang sehr viel Zeit am Ende der Aufgabe erspart. Unsere Struktur ist daraufhin ausgerichtet evtl. in weiteren Projekten weiter ausgebaut werden zu können. So kann man nach Belieben weitere Lebewesen in die Welt aufnehmen, und ihnen eigene Verhaltensweisen hinzufügen, bis ein komplexes Ökosystem entstanden ist. Dies war während der Implementation und Erweiterung unseres Projektes eine große Hilfe und wird hoffentlich noch Nutzen in der Zukunft finden.