Computergrafik II - Prüfungsvorleistung

Teil 2

Visualisierung von Schwarmverhalten am Beispiel von Schnappern, Barrakudas und Haien mit OpenGL und Java

Carl Richter
Tobias Mayerhanser
Silvan Gümüsdere

25. Juni 2017

Inhaltsverzeichnis

1 Abstract

Die vorliegende Arbeit dokumentiert die Visualisierung eines Schwarmverhalten von Raub- und Friedfischen im Meer. Als exemplarische Beispiele hierfür werden Rote Schnapper, Barrakudas und Haie als grafische Primitive dargestellt. Die Überlegungen basieren auf Craig Reynolds drei großen Regeln: Separation, Ausrichtung und Kohäsion. Umgesetzt ist dieses Projekt in Java, OpenGL mit LWJGL2 und enstand im Rahmen der Veranstaltung 'Computergrafik/Visualisierung II' an der HTW Dresden bei Professor Block-Berlitz.

2 Schwarmverhalten

Eine Vielzahl von Lebewesen halten sich in unserer Natur in großen Schwärmen auf. Meist gehören alle Mitglieder einer Aggregation der gleichen Art an. Typische Schwarmtiere sind Ameisen, Fische oder Vögel. Die Schwarmbildung hat für die Tiere große Vorteile: Gemeinsame Nahrungssuche, Schutz vor möglichen Fressfeinden und schnellere Fortbewegung.
1986 veröffentlichte Craig Reynolds, Boids: Das erste, am Computeram Computer modellierte Schwarmverhalten. Reynolds Prinzip basiert auf 3 Regeln:

- **Kohäsion:** Bewege dich in Richtung des Mittelpunkts derer, die du in deinem Umfeld siehst.
- **Separation:** Bewege dich weg, sobald dir jemand zu nahme kommt.
- **Alignment:** Bewege dich in etwa in dieselbe Richtung wie deine Nachbarn.

Dieses Prinzip lässt sich grob auf die meisten Schwärme übertragen.

3 Fiktiver Schwarm

Zur Visualisierung eines Schwarmverhaltens entschieden wir uns für einen Salzwasserfischschwarm, bestehend aus Roten Schnappern, die gemeinsam in Schwärmen durch das Wasser schwimmen und sich von Plankton ernähren, Barrakudas die als Einzelgänger jagt auf kleinere Fische machen (hier Schnapper) und ein Hai, welcher sowohl Schnapper als auch Barrakudas frisst.

4 Eigenschaften und Verhalten der Tiere mit deren Repräsentationen im Programmcode

Jede Tierart hat in der Natur eine Vielzahl verschiedener Verhaltensmuster und Eigenschaften, welche die Lebewesen charakteristisch Formen. Dazu zählen beispielsweise der Körperbau, Höchstgeschwindigkeit, Sichtweite und Intelligenz. In unserer Simulation können wir viele dieser Einflussfaktoren nachbilden. So besitz jeder Fisch eine Höchstgeschwindigkeit in Bewegungsrichtung und eine maximale Drehgeschwindkeit um die eigene Achse. Diese Werte sind abhängig von der Masse in Abhängigkeit zum Bezugssystem.
Zusätzlich hat jeder Fisch eine bestimmte Sehstärke um eigene und fremde Artgenossen für Kollisionsvermeidung, Schwarmbildung und Lebensgefahr zu erkennen.

4.1 Grundüberlegungen

Alle dargestellten Lebewesen erhalten - je nach Tierart und vorhandener Gruppenzugehörigkeit - besondere Eigenschaften. Die komplette Berechnungsgrundlage basiert auf der Vektorklasse aus Teil I dieser Arbeit.

4.1.1 Position

Die Position wird durch einen Ortsvektor $\vec{v} = (x, y)^T$ mit `Vektor2D` abgebildet.

4.1.2 Orientierung

Die Orientierung bzw. Bewegungsrichtung wird als normierter Vektor $\vec{v} = (x, y)^T$ mit `Vektor2D` gespeichert.

4.1.3 Bewegung

Als Bewegung ist $\vec{v} \cdot V_{bew}$ definiert. V_{bew} steht für die Bewegungsgeschwindigkeit.

4.1.4 Drehung

Jedes dargstellte Tier hat eine festgelegte Drehgeschwindigkeit. Das Prinzip ist relativ simpel gehalten: Die kleinen Fische sind verhältnissmäßig schneller in der Drehbewegung, große Exemplare langsamer.
Die Klasse `Vektor2D` und `LineareAlgebra` wurde um die Funktion `rotate` erweitert. Berechnungsgrundlage ist die Drehmatrix. Damit die Drehung der Fische immer harmonisch abläuft, bleibt die Geschwindigkeit konstant.

4.2 Schnapper

Ein Schnapper wird im virtuellen Meer mit folgenden Konstanten instanziiert, falls nicht anders angegeben sind alle Werte Verhältnissangaben.

Eigenschaft	**Wert**
`MIN_SWARMSIZE`	5
`SWARMRADIUS`	150 px
`SEPARATION_STRENGTH`	5
`COHESION_STRENGTH`	5
`COMFORT_RADIUS`	35 px
`COHESION_RADIUS`	105 px
`PANIC_RADIUS`	300 px
`SPEED`	1
`ROTATION_SPEED`	1

4.2.1 Schwarmverhalten

Eine Ansammlung von wenigstens fünf Fischen im 150px Radius gilt in dieser Simulation als Schwarm. Diese Fische erhalten durch ihren Schwarm einen Geschwindigkeitsbonus um die $1\frac{1}{2}$-fache Geschwindigkeit. Kommt ein feindlicher Fisch zu nahe, brechen die Fische den Schwarm auf und versuchen zu fliehen. Ist ein Fisch alleine unterwegs, schwimmt er so lange in die gleiche Richtung, bis er einen Verband von Artgenossen findet und sich diesem anschließt.

4.3 Barrakuda

Barrakudas sind exzellente Jäger. Sie treten bei der Jagt als Einzelgänger auf. Das bedeutet für die Simulation, dass auf Alignment und Kohäsion verzichtet werden kann. Es wird nur auf Separation zwischen den Barrakudas geachtet, damit die Fische kein fremdes Jagtrevier eines Artgenossen betreten.

Eigenschaft	Wert
`SEEK_RADIUS`	200 px
`COMFORT_RADIUS`	50 px
`PANIC_RADIUS`	200 px
`SPEED`	2
`ROTATION_SPEED`	1

4.4 Hai

Der Hai ist in den Weltmeeren an der Spitze der Nahrungskette. Auch in dieser Simulation kann der Hai als einziges fiktives Lebewesen sowohl Schnapper als auch Barrakuda fressen. Der Hai ist durch seine riesige Schwanzflosse viel schneller in der Vorwärtsbewegung als alle anderen Fische. Durch seine Größe ist er aber in seiner Rotationsgeschwindigkeit nicht überlegen. Der Knorpelfisch ist in der Simulation ein einsamer Einzelgänger ohne Artgenossen. Er lässt sich über die Pfeiltasten auf der Tastatur steuern.

Eigenschaft	Wert
`SPEED`	3
`ROTATION_SPEED`	1

5 Oberflächenlogik

5.1 Skalierung

So gut wie alle Parameter der einzelnen Fischarten, wie beispielsweise `COMFORT_RADIUS` oder `PANIC_RADIUS` werden entsprechend der Pixeldichte des Displays skaliert. Die oben erwähnten Eigenschaften von Schnapper, Barrakuda und Hai sind somit die Grundwerte für die Skalierung.

5.2 Koordinatentransformation

(vgl. `transformCoordinates()`-Funktion in der Klasse `BaseObject`)

Die Positionen der einzelnen Individuen werden, sobald sie den rechten bzw. unteren Bildschirmrand erreichen für die linke bzw. obere Seite neu berechnet. Somit bleiben die Individuen kontinuierlich im sichtbaren Bereich des Bildschirms.

Damit die Fische nicht in den Bildschirm "herein-" bzw. "herausploppen" werden die Fische auch `BOXING = 15` Pixel auf jeder Seite außerhalb des sichtbaren Bereichs gerendert.

5.3 Berechnung der Punkte

Um die individuellen Fische mittels OpenGL darstellen zu können, werden drei Punkte benötigt, die dann ein Dreieck bilden, in dem Farbe gezeichnet wird. Diese Dreiecke werden anhand des Ortsvektors zur aktuellen Position des Individuums und des Vektors, der der Orientierung entspricht, d.h. in welche Richtung der Fisch zeigt, berechnet.

Dazu wird zu aller erst zum Ortsvektor der Orientierungsvektor* addiert um die Spitze (Punkt 1) zu berechnen. Für die beiden hinteren Punkte wird der Orientierungsvektor vom Ortsvektor

subtrahiert und dann entweder der Normalenvektor* des Orientierungsvektors addiert (Punkt 2) oder subtrahiert (Punkt 3).

** Diese Vektoren werden entsprechend der Skalierung und der Größe des Individuums noch mit einem Skalar multipliziert um die Größe auf dem Bildschirm anzupassen*

6 Handhabung

6.1 Starten der Simulation über den Launcher

Der Launcher kann entweder über die JAR-Datei oder sofern der Quellcode selbst kompiliert wurde, über die Klasse `FishSimulator` gestartet werden.

Im Launcher der Simulation lassen sich die Anzahl der Individuen (Schnapper, Barrakudas) als auch der Hai an- und ausschalten, die horizontale sowie vertikale Auflösung der Anwendung und der Fenstermodus (Fenster, Maximiert oder Vollbild) einstellen.

Über den großen Button "Launch Simulation" kann die Simulation schließlich gestartet werden.

6.2 Starten der Simulation über die Konsole

Sofern der Quellcode selbst kompiliert wurde, kann die Simulation auch direkt über die Konsole gestartet werden, und zwar über die Klasse `SwarmingBehavior`.

Folgende Übergabeparameter sind dabei zu verwenden:

Parameter	**Erklärung**
`width`	die gewünschte Breite des OpenGL Fensters in Pixeln
`height`	die gewünschte Höhe des OpenGL Fensters in Pixeln
`windowMode`	0 = Fenstermodus, 1 = Maximiert, 2 = Vollbild
`snapperCount`	Anzahl der Schnapper
`barracudaCount`	Anzahl der Barrakudas
`sharkCount`	Anzahl der Haie

Es können entweder alle oder keine Parameter verwendet werden. Sofern keine Parameter angegeben werden, verwendet die Anwendung Standardwerte für alle Parameter.

6.3 Steuerung

6.3.1 Hai

Der bzw. die Haie können mit den Pfeiltasten nach links und nach rechts gesteuert werden.

6.3.2 Kamera

Die Kameraansicht kann mithilfe der gedrückten linken Maustaste verschoben werden.