Universität Ulm
Fakultät für Informatik
Abteilung Künstliche Intelligenz

Sommersemester 2001

Hauptseminar

„Technik der Spieleprogrammierung“

Simulation von Schwarmverhalten

Ausgeführt von Dominik Maschke

Betreuer Karsten Gugler

Vortrag 20. Juni 2001

Inhaltsverzeichnis

1. Einleitung

Bei der Simulation von Schwarmverhalten will man – wie so oft in der Informatik – ein natürliches Phänomen imitieren. Es tritt genaugenommen immer dann auf, wenn mindestens zwei – oft auch bis zu mehreren hundert – Individuen von gleicher Art zusammentreffen oder sich zusammen aufhalten.

Mögen einige Beispiele aus der Natur den Begriff „Schwarmverhalten“ verdeutlichen: Ein Fischschwarm gleitet durch das Wasser auf der Suche nach Futter, ein Vogelschwarm sammelt sich in der Luft und lässt sich – wie auf ein unsichtbares Zeichen hin – auf einer Stromleitung nieder oder eine Herde Rentiere beweidet eine Wiese und ergreift die Flucht, als sie einen Wolf erblickt. Aber auch Menschen legen in gewissen Situationen ein Schwarmverhalten an den Tag – denkt man nur an die morgendliche Öffnung der Türen eines Kaufhauses zum Sommer-Schluss-Verkauf; das Erdbeben, das eine halbe Stunde später die gleichen Menschen wieder ins Freie lockt oder die alljährliche Pilgerreise deutscher Touristen zu südlichen Zielen.

Da die Individuen eines Schwarmes alle ein sehr ähnliches Verhalten zeigen und – falls erforderlich – dieses auch nahezu gleichzeitig ändern, könnte man auf folgenden Gedanken kommen: Es existiert eine Instanz, die allen Individuen sagt, in welchem Tempo sie sich bewegen sollen, wie viel Platz sie zu den anderen halten sollen, wann sie sich in welche Richtung bewegen oder an welcher Stelle niederlassen sollen.

Gegen diese Annahme spricht allerdings, dass sich die Individuen ja irgendwie einig sein müssten, wer ihr Anführer sei. Zum anderen ist es – wie oben schon erwähnt – einer Gruppe von Individuen beliebiger Anzahl möglich, die für einen Schwarm typischen Manöver auszuüben. Des weiteren kann man beobachten, dass es eine „kritische Anzahl“ an Individuen gibt, ab der ein einheitliches Schwarmverhalten nicht mehr möglich ist und sich die Menge dann aufteilt.

Es gibt mehrere Gründe, weshalb Individuen bestrebt sind, sich im Schwarm zu bewegen. Die Plätze inmitten eines Schwarmes bieten Schutz vor Feinden. Auf der anderen Seite lässt es sich im Rudel leichter und mit einer höheren Erfolgswahrscheinlichkeit jagen. Ein Schwarm nimmt mehr Details seiner Umgebung wahr, als ein einzelnes Individuum. Wenn nur ein Mitglied einer Gruppe einen Punkt oder ein Gebiet aus welchem Grund auch immer für sehr interessant hält, werden durch sein Verhalten in sehr kurzer Zeit zuerst seine unmittelbaren Nachbarn und wie in einer Kettenreaktion bald der ganze Schwarm darauf aufmerksam. All das sind Gründe, weshalb dieses Phänomen auch bei Computerspielen interessant ist.

Ist das Schwarmverhalten etwas, das dem Individuum eigen ist; ist es eine angeborene Fähigkeit? Kann man Schwarmverhalten in Formeln fassen, es also künstlich erzeugen? Zahlreiche mehr oder weniger erfolgreiche Versuche zeigen, dass das Schwarmverhalten auf dem Rechner nachzubilden ist. Auf mögliche Implementierungen soll im folgenden eingegangen werden.

2. Bewegung eines einzelnen Individuums

Bevor wir uns mit der Bewegung einer Gruppe befassen, ist es wichtig, die Bewegung eines einzelnen Individuums zu verstehen.

2.1 Ebenen der Bewegung

An einem Beispiel aus der Natur soll die Bewegung in die drei Ebenen der Kontrolle differenziert werden. Ein Löwe durchstreift ein Gebiet auf der Suche nach etwas Essbarem. Von der Aktionsauswahl über die Steuerung bis hin zur Regelung wollen wir uns nun die

einzelnen Ebenen der Bewegung herausgreifen, die ein Individuum mehr oder weniger bewusst berücksichtigt, um von dem einen Aufenthaltsort zum nächsten zu gelangen.

Angenommen, der Löwe entdeckt ein kleines, leckeres Lebewesen und beschliesst, dieses zum Abendessen nach Hause zu bringen. Die Wahl einer Strategie oder das Fassen eines Zieles wird als Aktionsauswahl bezeichnet.

In der nächsten Ebene, der Steuerung, wird das gewählte Ziel in kleine Zwischenziele zerlegt. Der Löwe entscheidet sich zum Beispiel in dem einen Moment zu beschleunigen, in einem anderen eine Rechtskurve zu laufen.

In der Ebene der Regelung wird ein Zwischenziel in Befehle für den Bewegungsapparat umgesetzt. Beim betrachteten Löwe sind das im Wesentlichen seine Beine, es könnten aber auch Flügel, Flossen oder ein Motor sein.

Die Aktionsauswahl ist ein Gebiet der KI, die Regelung überlässt man beispielsweise den Elektro-Technikern. Nach der Vorstellung einiger Facetten eines möglichen Bewegungsmodelles für rechnergestützte Individuen soll es in dieser Arbeit um die Ebene der Steuerung gehen. Wir werden sehen, dass das Schwarmverhalten wohl darin festzumachen ist.

2.2 Einfaches Bewegungsmodell

Bei der rechnergestützten Bewegung eines Individuums ist es zweckmäßig, eine Punktmasse als Annäherung zu verwenden. Zum einen sind damit viele Vorgänge und Manöver naturgetreu nachzubilden, und zum anderen ermöglicht es eine einfache und zeitsparende Berechnung der Bewegungen. Beispielsweise hat eine Punktmasse sehr wohl eine Geschwindigkeit und damit einen linearen Impuls; bei einer Drehung – ob im Stand oder während der Bewegung – wird man aber nicht mit dem Drehimpuls konfrontiert.

Ein einfaches Bewegungsmodell eines Individuums (z.B. ein zweidimensionales Fahrzeug) könnte mit den folgenden Variablen implementiert sein:

```
position              Vektor /*hier: 2D-Koordinaten*/
geschwindigkeit       Vektor
kraftrichtung         Vektor
kraft                 Skalar
max_kraft             Skalar
max_geschw            Skalar
beschleunigung        Vektor
```

Damit kann man für jeden Zeitschritt die neue Position folgendermaßen berechnen:

```
ask_individuum_for (kraftrichtung, kraft);
beschleunigung = normalize(kraftrichtung) * truncate(kraft, max_kraft);
geschwindigkeit = truncate(geschwindigkeit+beschleunigung, max_geschw);
position = position + geschwindigkeit;
```

Dabei berechnet `normalize` den Einheitsvektor; `truncate` ist eine Art Minimumsbildung unter Berücksichtigung des Vorzeichens, also

```
truncate (Skalar x, Skalar y) {    /* wobei: y >= 0 */
  if x > 0 then return min(x,  y);
           else return max(x, -y);
}
```

beziehungsweise

```
truncate (Vector x, Skalar y) {    /* wobei: y >= 0 */
   return normalize(x)*truncate(length(x), y);
}
```

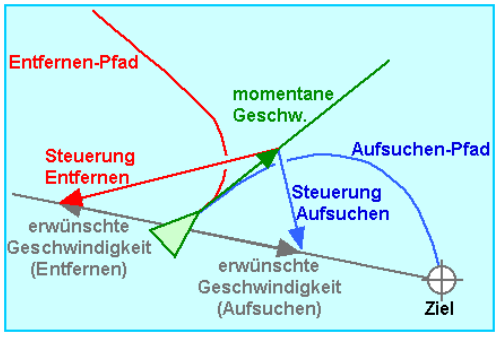
Denkbare Verfeinerungen des Bewegungsmodelles sind die Einführung einer Masse des Individuums oder die Berücksichtigung von Reibung oder Dämpfung der Bewegung. Die

nächsthöhere Ebene – die Steuerung – muss sich also lediglich um die Richtung und den Betrag der Kraft kümmern, mit der im nächsten Zeitschritt beschleunigt werden soll.

2.3 Grundlegende Steuerungsfertigkeiten

Anhand einiger ausgewählter Fähigkeiten und beispielhafter Angabe von Formeln soll die Ebene der Steuerung genauer erläutert werden.

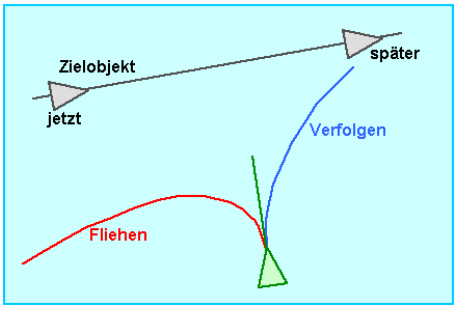
2.3.1 „Aufsuchen“ und „Entfernen“



Beim „Aufsuchen“ soll ein Fixpunkt möglichst schnell erreicht werden. „Entfernen“ ist das Gegenteil von „Aufsuchen“. Hierbei soll sich das Individuum schnellstmöglich von einem bestimmten Punkt entfernen. Die Ziel-Geschwindigkeit hat dabei das entgegengesetzte Vorzeichen.

```
aufsuchen (Vector ziel) {
  ziel_geschw = normalize(ziel-position) * max_geschw;
  kraftrichtung = ziel_geschw – geschwindigkeit;
  kraft = max_kraft;
}
```

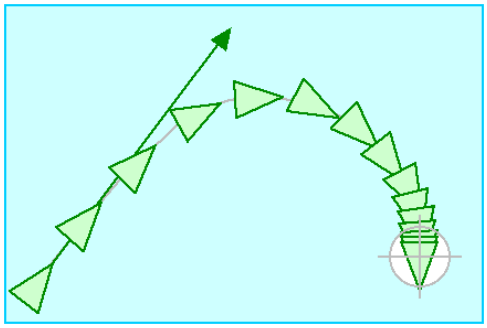
2.3.2 „Verfolgen“ und „Fliehen“



Der Unterschied zwischen „Aufsuchen“ und „Verfolgen“ liegt darin, dass nun ein bewegtes Ziel erreicht werden soll. Dementsprechend verhält es sich auch mit „Entfernen“ und „Fliehen“.

```
verfolgen (Individuum ziel) {
  ziel_vector = ziel.position + ziel.geschwindigkeit;
  aufsuchen (ziel_vector);
}
```

2.3.3 „Punktlanden“ und „Begleiten“



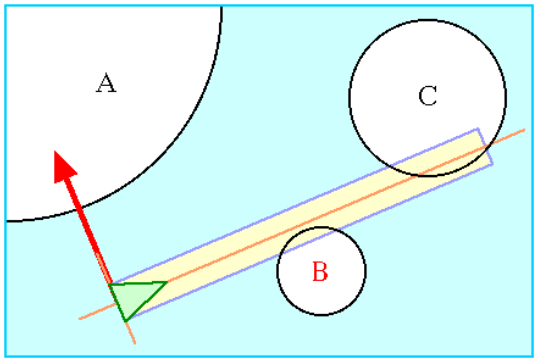
Beim „Aufsuchen“ und „Verfolgen“ wird das Ziel zwar sehr schnell erreicht, das Individuum schießt aber darüber hinaus. Die Fähigkeit „Punktlanden“ ermöglicht es, den Zielpunkt schnell zu erreichen und rechtzeitig (gesteuert durch Brems-Parameter b) zu bremsen, um darauf anzuhalten. Beim „Begleiten“ wird versucht, das Individuum mit der gleichen Richtung und der gleichen Geschwindigkeit wie das Ziel zu bewegen, allerdings mit einem bestimmten Abstand oder Offset dazu.

```
punktlanden (Vector ziel) {
  ziel_geschw = normalize(ziel-position) *
             truncate(length(ziel-position)*b, max_geschw);
  kraftrichtung = ziel_geschw – geschwindigkeit;
  kraft = length(kraftrichtung);
}
begleiten (Individuum ziel, Vector offset) {
  ziel_vector = ziel.position + ziel.geschwindigkeit + offset;
  punktlanden (ziel_vector);
}
```

2.3.4 „Hindernis-Vermeiden“

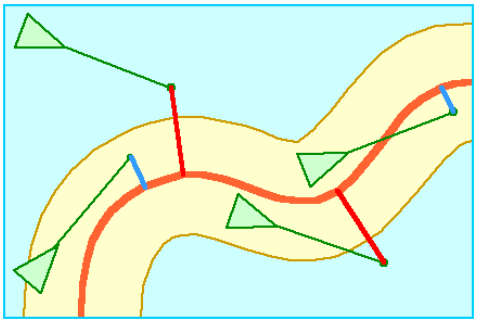
Diese Fähigkeit ermöglicht einem Individuum, in einer Umgebung mit Hindernissen kollisionsfrei zu manövrieren. Handlungsbedarf besteht allerdings nur, falls ein Hindernis sich sowohl in der Nähe als auch im Weg des Individuums befindet. Vereinfacht wird angenommen, dass es sich sowohl beim Individuum als auch beim Hindernis um Kugeln bzw. Kreise handelt.

Ziel von „Hindernis-Vermeiden“ ist es, ein imaginäres Rechteck (Zylinder) in Bewegungsrichtung des Individuums freizuhalten, den Freiraum (vgl. Abbildung). Die Breite ist dabei entsprechend dem Durchmesser des bewegten Objekts zu wählen, die Länge des Freiraumes richtet sich nach der Geschwindigkeit und Agilität des Individuums. Hindernisse, die den Freiraum nicht schneiden, stellen keine unmittelbare Gefahr dar und müssen beim „Hindernis-Vermeiden“ nicht berücksichtigt werden. Befinden sich jedoch die Hüllen mehrerer Hindernisse im Freiraum, so wird nur dasjenige Hindernis berücksichtigt, welches dem Individuum am nächsten liegt. Der Rückgabewert dieser Funktion ist dann ein entsprechendes Steuersignal. Droht keine Kollision, wird ein spezieller Wert (z.B. *nil*) an den Aufrufer zurückgegeben.



2.3.5 „Pfad-Verfolgen“

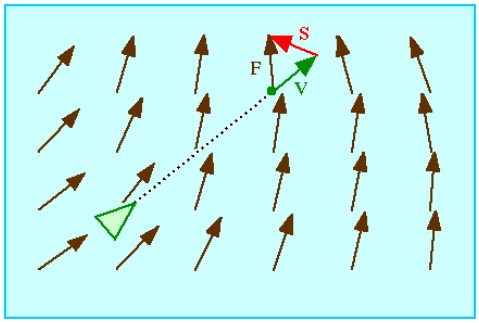
Mit dieser Fertigkeit kann ein Individuum einem vorgegebenen Pfad folgen, der bspw. durch eine parametrisierte Kurve festgelegt ist. Um die Steuerung dafür zu berechnen, wird zunächst eine Vorhersage über dessen zukünftige Position getroffen, vergleichbar mit der aus „Hindernis-Vermeiden“. Ist diese Position ausserhalb eines gewissen Korridors um dem Pfad, wird mit den Steuerwerten der Fähigkeit „Aufsuchen“ korrigiert. Als Ziel dafür wird der Punkt auf dem Pfad verwendet, welcher der vorher berechneten zukünftigen Position am nächsten liegt. Weitere Parallele zum „Hindernis-Vermeiden“: Befindet sich die zukünftige Position – genauer genommen auch alle Punkte bis dorthin – innerhalb des Korridors um den Pfad, wird dies durch einen besonderen Wert dem Aufrufer mitgeteilt.



Eine Variante besteht darin, stets ein Steuersignal zu liefern. Dieses wird stärker – Gewichtung direkt proportional zum Abstand – je weiter die zukünftige Position vom Pfad entfernt ist.

2.3.6 „Flussfeld-Verfolgen“

Für manche Anwendungen ist es nützlich, dem Individuum in Abhängigkeit von dessen Position die Richtung und Geschwindigkeit seiner Bewegung zuzuweisen. Die Implementierung ist denkbar einfach: Für das gesamte Gebiet, in dem sich die Individuen möglicherweise aufhalten könnten, werden Richtungen und Geschwindigkeiten – ein sogenanntes Flussfeld – als Vektoren mit definierter Richtung und Länge vorgegeben. An der zukünftigen Position des bewegten Objekts wird der dort gültige Vektor des Flussfeldes ermittelt. Dies geschieht in der Regel durch Interpolation zwischen benachbarten Werten, falls die Werte (Stärke und

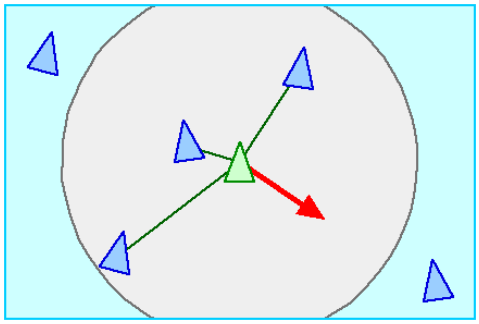


Richtung des Feldes) nur an bestimmten Punkten in einer Tabelle gespeichert sind. Rückgabewert und damit das Steuersignal ist die Differenz zur aktuellen Geschwindigkeit.

```
flussfeld_verfolgen {
  kraftrichtung = get_force(position + geschwindigkeit);
  kraft = length(kraftrichtung);
}
```

2.3.7 „Separieren“

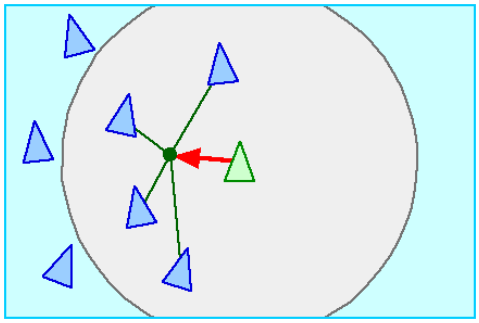
Die Fertigkeit „Separieren“ ermöglicht dem Individuum, eine bestimmte Mindest-Entfernung zu anderen Individuen zu halten. So kann ein zu enger Aufenthalt nebeneinander und damit der Verlust der Manövrierfähigkeit vermieden werden. Das Steuerungssignal wird berechnet, indem zunächst die Individuen bestimmt werden, die sich innerhalb eines kritischen Bereiches aufhalten. Diese erzeugen jeweils eine abstoßende Kraft. In Versuchen [4] hat es sich als praktikabel erwiesen, deren Größe umgekehrt proportional zum Abstand der beiden Individuen zu wählen. Die resultierende Steuerung ergibt sich aus der Vektoraddition aller abstoßenden Kräfte.



```
separieren (Skalar radius) {
  kraftrichtung = sum_up_repulsive_forces_of_teammates_nearer_than(radius);
  kraft = length(kraftrichtung);
}
```

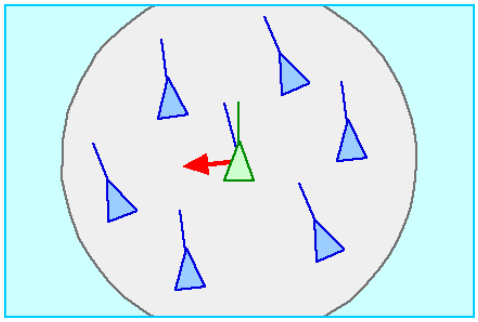
2.3.8 „Zusammenbleiben“

Soll ein Individuum mit anderen, nicht allzu weit entfernten Individuen, eine Gruppe bilden, so ist der Einsatz von „Zusammenblieben“ am sinnvollsten. Ähnlich wie beim „Separieren“ werden alle Individuen näher als ein bestimmter, vorher festgelegter, Abstand gesucht. Der Schwerpunkt von diesen wird als Ziel für die Fertigkeit „Aufsuchen“ verwendet.



```
zusammenbleiben (Skalar radius) {
  ziel_vector = average_pos_of_teammates_nearer_than(radius);
  aufsuchen(ziel_vector);
}
```

2.3.9 „Ausrichten“



„Ausrichten“ ist eine weitere Fertigkeit, die schon einen Ausblick auf die Bewegung mehrerer Individuen bietet. Die Bewegung des betrachteten Individuums soll an die seiner Nachbarn angepasst werden. Dazu wird von allen Individuen innerhalb eines bestimmten Radius der gewichtete (vgl. „Separieren“) Mittelwert von deren Bewegungs-Vektoren errechnet. Dieser ist zugleich die gewünschte Geschwindigkeit, die Differenz zur eigenen Geschwindigkeit ist das Steuersignal.

```
ausrichten (Skalar radius) {
  ziel_geschw = average_vel_of_teammates_nearer_than(radius);
  kraftrichtung = ziel_geschw - geschwindigkeit;
  kraft = length(kraftrichtung);
}
```

3. Bewegung mehrerer Individuen

Haben wir uns bisher mit der Bewegung eines einzelnen Individuums befasst, so soll nun die Bewegung mehrerer Individuen betrachtet werden. Dabei gelingt eine scharfe Trennung der Funktionalitäten – sind sie Eigenschaft der Bewegung eines oder mehrerer Individuen – nicht immer.

Die letzten der vorgestellten Steuerungsfertigkeiten lassen in Ansätzen bereits erkennen, worauf es bei der Bewegung einer Gruppe ankommt. Beim Schwarmverhalten ist es zum einen sehr wichtig, dass die Individuen nicht miteinander kollidieren. Zum anderen aber – und gerade das macht ja das Schwarmverhalten aus – sollen die Individuen zusammenfinden und danach zusammenbleiben bzw. eine Gruppe formen. Auch eine dritte Eigenschaft des Schwarmverhaltens, die Angleichung der Bewegung des Individuums an die seiner unmittelbaren Nachbarschaft, ist nicht von geringer Bedeutung.

Scheinbar müssen für bestimmte Effekte mehrere der bisher vorgestellten Fähigkeiten und möglicherweise noch andere miteinander kombiniert werden. Im Folgenden soll genauer auf Arten der Kombination eingegangen werden.

3.1 Kombination von Steuerungsfertigkeiten

Es gibt zwei grundsätzliche Möglichkeiten, mehrere Strategien zu verbinden. Da man trotz einer Reihe von Vorschlägen für Steuerungen nur einen Befehl an den Bewegungsapparat abgeben kann, ist man gezwungen, gezielt einen Vorschlag herauszugreifen, oder aus allen vorgeschlagenen ein neues Steuersignal zu generieren.

Zum einen kann eine Art Summe oder (evtl. gewichteter) Mittelwert der zu verfolgenden Ziele das eine Steuersignal ergeben. Zum anderen kann das Individuum nacheinander von jedem der möglicherweise widersprüchlichen Pläne kleine Schritte verwirklichen. Die Anzahl der Schritte bzw. die Zeitdauer, für die eine bestimmte Fähigkeit das für das ganze Individuum resultierende Steuersignal generiert, kann – je nach Anwendung – für jede Strategie gleich sein oder wird durch eine Gewichtung vorgegeben.

Eine weitere Möglichkeit, nur eine Fähigkeit auszuwählen, die alleine über das Verhalten des Individuums bestimmen kann, ist die Vergabe von Prioritäten. Im Wesentlichen wird stets das Steuersignal derjenigen Strategie mit der höchsten Priorität ausgewählt. Meldet diese allerdings keinen Handlungsbedarf an, dann erhält die nächstniedrigere Fähigkeit solange die Steuerung über das Individuum.

Eine interessante Mischtechnik: Die Strategie mit der höchsten Priorität wird nur mit einer bestimmten Wahrscheinlichkeit evaluiert. Besteht Handlungsbedarf, so wird das Steuersignal auch verwendet. In den anderen Fällen (kein Handlungsbedarf oder zufällig nicht ausgewählt) wird die nächstpriorisierte Fähigkeit (möglicherweise) berücksichtigt.

3.2 Simulation von Schwarmverhalten

Wie oben bereits angeklungen, reicht es für den Teilnehmer eines Schwarms aus, die drei Grundfähigkeiten „Separieren“, „Zusammenbleiben“ und „Ausrichten“ sowie deren Kombination zu beherrschen. Mischt man noch weitere Steuerungsfähigkeiten dazu, entstehen einige interessante Variationen eines Schwarms. Auf einige Beispiele soll im Folgenden eingegangen werden.

Dennoch sind an dieser Stelle einige generelle Hinweise vorweg angebracht. Auch wenn in dieser Arbeit nur von Bewegungen die Rede ist, so ist es durchaus denkbar, dass sich die Individuen in ähnlicher Weise über andere Aktionen einigen. Ob ein gemeinsamer Angriff geplant wird oder nur die Verteidigung aufrecht erhalten wird, kann beispielsweise durch eine Art Abstimmung zwischen jeweils benachbarten Individuen geschehen. Alle Berechnungen

sind überdies vom jeweils gezeigten Bildschirmausschnitt unabhängig, sie werden über das gesamte Spielfeld aufsummiert und berechnet.

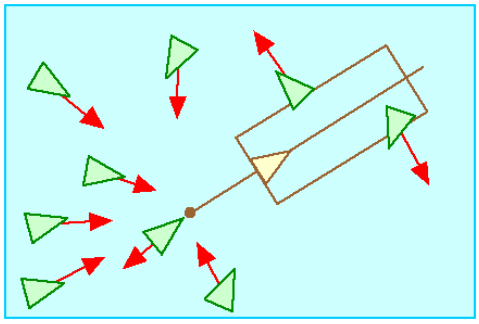
3.2.1 Zielloser und gelenkter Schwarm

Um einen ziellosen Schwarm zu simulieren, erhalten die beteiligten Individuen lediglich die drei Grundfähigkeiten des Schwarmverhaltens. Bei einem gelenkten Schwarm muss noch mindestens eine weitere Fähigkeit zur Auffindung des gewünschten (Zwischen-) Ziels von jedem teilnehmenden Individuum beherrscht werden.

Zum einen kann jedes der Individuen aus dem Schwarm – und damit der ganze Schwarm – mit der Fähigkeit „Aufsuchen“ auf einen festen Punkt zielen. Zum anderen kann der Schwarm – auch hier ist wieder jedes einzelne Individuum gemeint – durch „Pfad-Verfolgen“ oder „Flussfeld-Verfolgen“ gelenkt werden.

Eine weitere Möglichkeit: Ein virtueller Punkt wandert – wie der künstliche Hase beim Hunderennen – den Pfad entlang. Die Individuen zielen dann mittels „Aufsuchen“ auf diesen Punkt. Selbst wenn sich der virtuelle Punkt sehr ruckartig fortbewegt und dessen Weg insbesondere Knicke beschreibt, entsteht im Schwarm ein weiches Reiseverhalten, da die Individuen aufgrund ihrer Trägheit und Schwarmzugehörigkeit nur langsam die Richtung und Geschwindigkeit ändern können. Auch die Fähigkeit „Anführer-Verfolgen“ (vgl. Abbildung) lenkt den Schwarm in eine gewünschte Richtung – vorausgesetzt der Anführer wird durch eine andere der soeben beschriebenen Methoden auf dem richtigen Weg gehalten.

Der Anführer in der Abbildung ist durch das Rechteck vor sich und den Punkt hinter sich gekennzeichnet. Alle anderen Individuen müssen aus dem Rechteck vor dem Anführer fliehen, um diesem freie Bahn zu gewähren. Die zweite Priorität hat das Anvisieren des Punktes hinter dem Anführer. So wird diesem zum einen Bewegungsfreiheit garantiert, zum anderen verfolgen die übrigen Individuen so den Anführer. In der Abbildung sind für alle Individuen die Steuersignale aus der Fähigkeit „Anführer-Verfolgen“ eingezeichnet.



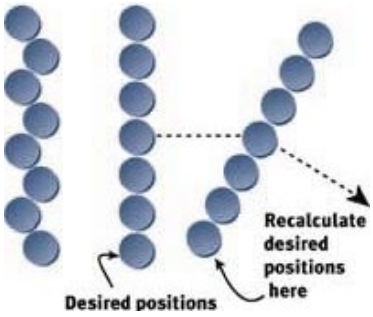
Eine sehr interessante Variante für die Realisierung eines gelenkten Schwarms ist der „Experten-Schwarm“: Nur einige der teilnehmenden Individuen sind mit einer Zielfähigkeit ausgestattet. Da diese in der entsprechenden Situation ihre Richtung oder Geschwindigkeit ändern, gleichen sich alle anderen in ihrem Bestreben, einen Schwarm zu bilden, relativ schnell daran an. Die Angleichung erfolgt dann ohne das Wissen des übergeordneten Ziels, das dem ganzen Schwarm gemeinsam sein soll. Der Unterschied zu „Anführer-Verfolgen“ besteht darin, dass diejenigen Individuen, die das gewünschte Ziel des Schwarmes nicht kennen, auch nicht wissen müssen, wem sie zu folgen haben. Allein das Bestreben, einen Schwarm zu bilden reicht aus, um alle Individuen zum Ziel zu bringen.

3.2.2 Formationen

Formationen können gebildet werden, indem nicht nur für die ganze Gruppe ein virtueller Punkt einem Pfad folgt sondern für jedes Individuum ein solcher Punkt existiert. Realisiert wird dies ähnlich wie bisher durch einen einzigen zentralen Punkt, der eine Art Mittelpunkt des Schwarmes darstellt.

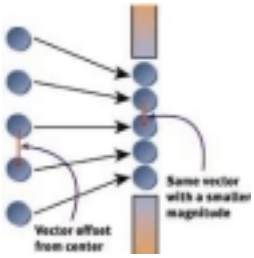
An diesem hängt für jedes teilnehmende Individuum ein Offset-Vektor. Durch Drehen oder Strecken dieser Vektoren um den zentralen Punkt ist es auch möglich, eine Formation zu drehen oder strecken oder auch soweit zu stauchen, bis sich zwei Individuen berühren.

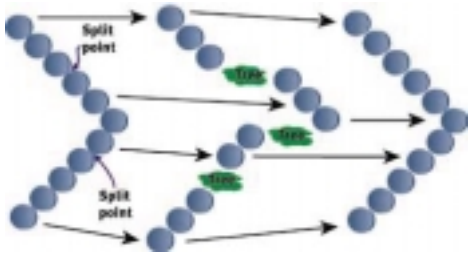
Als Grundfähigkeiten für eine Formation bieten sich von den drei bisherigen Standard-Fertigkeiten lediglich „Separieren“ und „Ausrichten“ an. Die Fähigkeit „Zusammenbleiben“ entspricht nicht den allgemeinen Anforderungen an eine Formation. So kann es durchaus gewollt sein, einzelne Individuen in einer gewissen Entfernung vom Hauptfeld zu bewegen. Die Abbildung zeigt eine Reihe ausbildende Formation, die sich im Laufe der Zeit fortbewegt und dabei dreht.

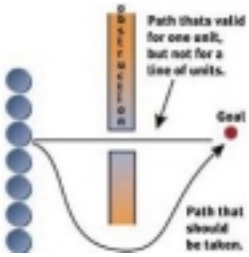


3.2.3 Vermeidung von Hindernissen und Passieren einer Engstelle

Soll sich ein Schwarm durch ein Gebiet bewegen, in dem Hindernisse oder Engstellen zu überwinden sind, so ändert sich an den bisher beschriebenen Methoden nichts. Lediglich die Fähigkeit „Hindernis-Vermeiden“ muss zu den restlichen Fähigkeiten beherrscht werden. In einigen Fällen sollte aber gerade bei Formationen über andere Möglichkeiten der Passierung einer Engstelle nachgedacht werden. Es kann sinnvoll sein, die Formation zu stauchen, sie zu teilen oder gar einen neuen Weg zu planen. (vgl. Abbildungen)







Eine interessante Alternative, die sowohl dem Programmierer als auch dem Prozessor die Arbeit erleichtern dürfte, befähigt Formationen ebenfalls, Hindernisse intelligent zu umgehen: Die oben beschriebenen virtuellen Ziel-Punkte jedes Individuums bewegen sich möglicherweise sogar durch die Hindernisse hindurch. Die Individuen beherrschen sowohl die drei Schwarm-Fähigkeiten als auch „Hindernis-Vermeiden“ und „Aufsuchen“ des virtuellen Zielpunkts. Die Formation zerteilt sich also an einem Hindernis, findet danach aber schnell wieder zusammen.

Im durchreisten Gebiet dürfen sich lediglich nicht allzu viele Hindernisse befinden und die Ziel-Punkte müssen sich in der Nähe des Hindernisses unterhalb der Maximalgeschwindigkeit der einzelnen Individuen fortbewegen. In gewisser Weise gelten diese Einschränkungen aber für jede Art und Weise, eine Formation in einem Korridor mit Hindernissen zu manövrieren.

3.3 Algorithmische Betrachtung einer Schwarmsimulation

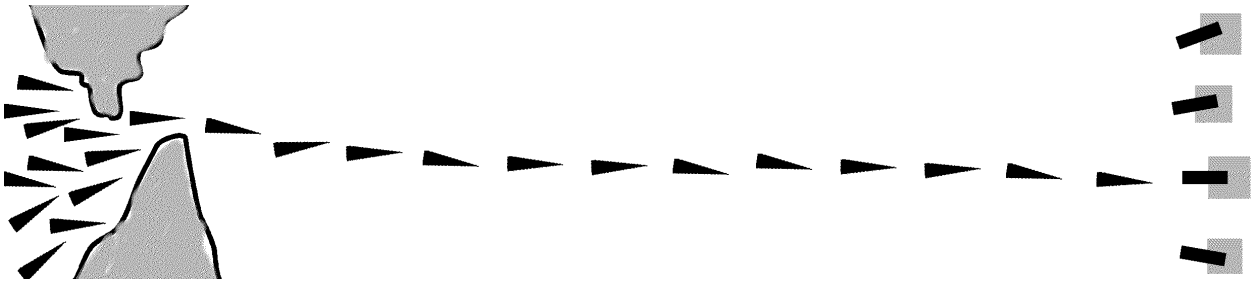
Eine naive Implementierung auf dem Rechner würde die Komplexität $O(n^2)$ (n: Anzahl der am Schwarm teilnehmenden Individuen) erzielen: Jedes Individuum muss über jedes andere nachdenken, und sei es nur, um zu entscheiden, dass es bei den Berechnungen nicht berücksichtigt werden muss. Aufgrund von Beobachtungen aus der Natur kann man aber folgern, dass es einen Algorithmus mit konstanter (bzw. bei der Simulation mehrerer Individuen auf einem Rechner: mit linearer) Laufzeit geben muss. Schliesslich kann in der Natur ein Schwarm beliebig groß werden, ohne dass sich die teilnehmenden Individuen vor lauter Nachdenken kaum mehr bewegen.

Man könnte sich auch mit einem $O(n^2)$-Algorithmus begnügen, wenn man n auf effiziente Weise – also in linearer Laufzeit – ausreichend klein halten könnte. Eine Möglichkeit besteht darin, die Individuen so in Gruppen aufzuteilen, dass sich die Mitglieder in räumlicher Nähe befinden. Ein einzelner müsste dann nur die Mitglieder seiner Gruppe und der unmittelbar benachbarten Gruppen für das Schwarmverhalten berücksichtigen. Lösungen dafür wurden

bereits in diesem Seminar unter dem Stichwort „Bucketsort“ – einem schachbrettartigen Sortier-Algorithmus – vorgestellt.

4. Beispiel: Angriff ohne Schwarmverhalten

Auf ein authentisches Beispiel wird bewusst verzichtet; der geneigte Leser kennt folgende Situation sicherlich aus dem einen oder anderen Computerspiel: Der eine Spieler – im Bild von links kommend durch die dreieckigen Einheiten dargestellt – greift die feste Stellung seines Gegners an – im Bild durch quadratische Türme mit Schussvorrichtung symbolisiert.



Trotz der Überzahl der dreieckigen Einheiten wird der Spieler auf der rechten Seite kaum in Bedrängnis geraten. Die Engstelle verhindert den Vormarsch als Gruppe. Würden sich dagegen die Einheiten nach der Engstelle wieder sammeln, so könnten sie gemeinsam auf die Stellung losschlagen. Zudem würden sich die Schüsse der Verteidiger auf mehrere Angreifer verteilen.

5. Zusammenfassung

Das Bestreben mehrerer Individuen, sich in einer Gruppe zu formieren und nach Aussen hin gleichförmig – quasi als ein großes Ganzes – aufzutreten, wird auch Schwarmverhalten genannt. In dieser Arbeit wurde der Frage nachgegangen, wie man ein solches Verhalten auf dem Rechner simulieren kann.

Nach der Unterteilung der Bewegung in die Ebenen Aktionsauswahl, Steuerung und Regelung wurden einige grundlegende Steuerungs- oder Bewegungsfähigkeiten eines einzelnen Individuums vorgestellt. Verschiedene Techniken der Mischung mehrerer Steuersignale runden die Grundlagen für die Simulation von Schwarmverhalten ab.

Dieses ist lediglich eine Mischung aus den drei elementaren Bewegungsmustern „Separieren“, „Zusammenbleiben“ und „Ausrichten“, die von am Schwarm beteiligten Individuen beherrscht werden müssen. Beherrschen die Schwarmteilnehmer noch bestimmte zusätzliche Steuerungsfähigkeiten, ist sogar ein gerichteter Schwarm oder eine Formation möglich. Beide sind in der Lage, Hindernisse, die sich auf dem Weg zum Ziel befinden, zu umgehen und danach wieder eine Einheit zu bilden.

Die Arbeit endet mit einer algorithmischen Betrachtung der Schwarmsimulation sowie mit einem Beispiel von fehlendem Schwarmverhalten in Computerspielen.

6. Referenzen

[1] C. W. Reynolds; „Flocks, Herds, and Schools: A Distributed Behavioral Model“; SIGGRAPH 87; Juli 1987

[2] Xiaoyuan Tu, D. Terzopoulos; „Artificial Fishes: Physics, Locomotion, Perception, Behavior“; Toronto 1994

[3] Craig W. Reynolds; „Not Bumping Into Things“; SIGGRAPH 88; August 1988

[4] Craig W. Reynolds; „Steering Behaviours For Autonomous Characters“; SIGGRAPH 97; 14. Januar 1997

[5] James Kennedy, Russell Eberhart; „Particle Swarm Optimisation“

[6] Dave C. Pottinger; „Implementing Coordinated Movement“; 29. Januar 1999
www.gamasutra.com/features/19990129/ implementing_01.htm