partikelintelligenz

für C.

...partikelintelligenz...

Computergenerierte Strukturen/Konstruktionen durch die Anwendung eines intelligenten Partikelsystems.

Masterthesis MAS CAAD ETH Zürich 2005-2006

Frank Theßeling

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

ETH ZÜRICH
Chair of CAAD
Prof. Ludger Hovestadt
HIL E 15.1
ETH Hönggerberg
8093 Zürich
Switzerland

www.caad.arch.ethz.ch

Text , Zeichnungen | Frank Theßeling

Zürich, Februar 2007

Acknowledgements

Danksagung

Professor of CAAD	Ludger Hovestadt
From CAAD Chair	Sibylla Spycher Markus Braach Katharina Bosch Karsten Droste If Ebenöther Pia Fricker Andrea Gleiniger Steffen Lemmerzahl Kai Rüdenauer Susanne Schumacher Georg Vrachliotis Christof Wartmann Oskar Zieta
MAS Courses Coordinator	Philipp Schaerer
ETHZ Department of Architecture Research / Postgraduate Studies	Bruno Dobler
MAS Team 2005_2006	Benjamin Dillenburger TobiasWendt Matthias Zäh Martin tann Meindert Versteeg Toni Kotnik Monika Annen DongYoun Shin Yael Ifrah or Girot Markovic Sladjana Claudia Fuhr Frank Theßeling

Abstract

Computergenerierte Strukturen/Konstruktionen durch die Anwendung eines intelligenten Partikelsystems.

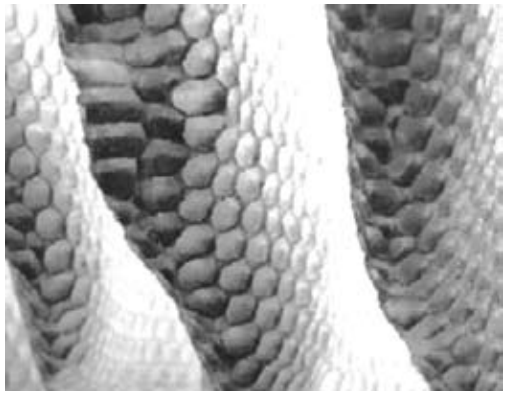
Mit der Anwendung objektorientierter Programmiersprachen ist es möglich ein System aufzubauen welches selbst-organisierende Phenomene abbildet. Ein solches System kombiniert die Zugänglichkeit, die Kohärenz- und die Lenkbarkeitsvorteile der zentralisierten Systeme mit dem Teilen, dem Wachstum und den Autonomiervorteilen der vernetzten Systeme.

Eine im Vorfeld zu dieser Arbeit entwickeltes Programm dient als Grundlage der Untersuchung selbst-organisierender Phenomene.

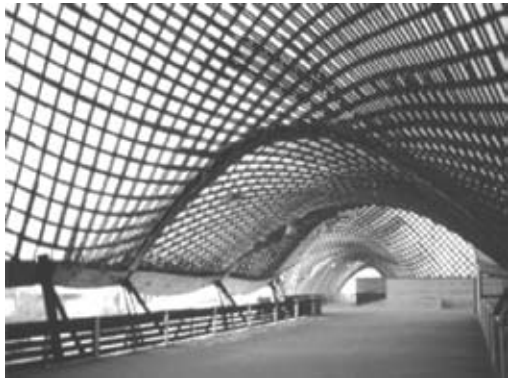
Das Programm bildet ein solches System, ein Schwarmverhalten, ab und bietet die Möglichkeit die generierten Formen zu bauen. Durch die Andwendung des am CAAD Lehrstuhl der ETH-Zürich entwickelten Prozesses der “Digitalen Kette” war es möglich die komplexen Resultate des Programms in gebauter Struktur zu überprüfen.

Das Ziel der Thesis ist die Untersuchung wie selbst-organisierende Systeme im Fomfindungs- Formgenerierungsprozess eingesetzt werden können und die Darstellung eines Produktionsprozesses. Gleichzeitig wird das Schwarmverhalten und dessen Anwendbarkeit auf die Struktur / Konstruktion untersucht.

Inhalt



|1| Bienenwaben



|2| Holzdachkonstruktion

Frei Otto, Mannheim 1975

1. Grundlage

Emergenz durch distributive Intelligenz ist ein wichtiges neues Konzept das in den vergangenen Jahren immer mehr Einfluss auf die Künstliche Intelligenz, Informationstheorie, digitale Technologie, Ökonomie, Wettervorhersagen, Materialwissenschaften und Biometrisches Engineering genommen hat. Im Architektonischen Kontext impliziert dies, das evolutionäre Prozesse nicht nur für die Formfindung angewendet werden sondern auch für die Komposition neuer Materialien und für das Strukturelle Design neuer Architekturen.

Im Folgenden werden die Begrifflichkeiten Strukturemergenz und distributive Intelligenz differenziert. Das gezeigte Beispiel „Partikelintelligenz“ beschäftigt sich mit diesen Phänomenen.

1.2 Distributive Intelligenz.

Distributive Intelligenz ist ein Forschungsfeld der Künstlichen Intelligenz (KI), das auf Agententechnologie basiert, es heißt auch Verteilte Künstliche Intelligenz (VKI). Es wird untersucht wie sich komplexe vernetzte Strukturen nach dem Vorbild Staatenbildender Insekten wie Ameisen, Bienen und Termiten, sowie teilweise auch Vogelschwärme generieren. |fig. 1, 2|

| 1| deterministisches Modell

Ein Beispiel für formgebende, emergente Algorithmen.

Die Abbildung zeigt computergesteuerte Partikel die durch einen deterministischen Algorithmus einen dreidimensionlen Kopf generieren. Dieses Modell beruht im wesentlichen auf dem Teilchenmodell von William T. Reeves.

|2| “Didabots” (Maris & te Boekhorst 1996, nach Pfeifer 2002)

Computergesteuerte Ameisen die durch einen Einfachen Algorithmus die weissen Kuben (Zuckerwürfel) gruppieren.

Folgende “einfache” Intelligenz bentuzt das Programm:

1. suche einen einzelnen Kubus
2. lege ihn zu einem Anderen

G. Beni und J. Wang hatten den Begriff „swarm intelligence“ 1989 im Kontext der Robotikforschung geprägt.

Die Individuen Staatenbildender Insekten agieren mit eingeschränkter Unabhängigkeit, sind in der Erfüllung ihrer Aufgaben jedoch sehr zielgerichtet. Die Gesamtheit solcher Insektengesellschaften ist überaus leistungsfähig, was Forscher auf eine hochgradig entwickelte Form der Selbstorganisation zurückführen. Zur Kommunikation untereinander nutzen Ameisen beispielsweise Pheromone; Bienen den Schwänzeltanz.

Ohne zentralisierte Form der Oberaufsicht ist das Ganze also mehr als die Summe der Teile.

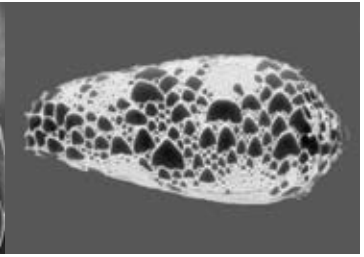
Die VKI - Forschung geht davon aus, dass die Kooperation künstlicher Agenten höhere kognitive Leistungen simulieren kann; Marvin Minsky bezeichnet dies als Society of Mind. |fig. 1,2|

Ein Einsatzbeispiel für diese so genannten Ameisenalgorithmen stellten Sunil Nakrani von der Universität Oxford und Craig Tovey vom Georgia Institute of Technology 2004 auf einer Konferenz über mathematische Modelle sozialer Insekten vor; sie modellierten die Berechnung der optimalen Lastverteilung bei einem Cluster von Internet-Servern nach dem Verhalten der Bienen beim Nektarsammeln.

Programmierte distributive Intelligenz zur Formfindung ermöglicht es komplexe Strukturen zu bilden und Varianten zu generieren. Jede erzeugte Lösung ist eine mögliche Lösung der komplexen Probleme, da der Algorithmus die Probleme und das Verhalten implementiert hat.

|2| Romanesu
Fraktales Wachstum

|2| Nautilus
Wachstum nach Fibunacci



|3| Kegelschnecke
fraktales Muster

|4| Emergenz im Fischschwarm

Die einfachste Form der Entstehung künstlichen Verhaltens kann definiert werden durch:

- Anwendung fester, lokaler Regeln auf die einzelnen Agenten

- Gruppen-Verhalten als emergentes Ergebnis

1.2 Emergenz

Emergenz ausschließlich Abstrakt oder ausschließlich Produktiv zu betrachten macht wenig Sinn. In der Emergenz ist dies untrennbar verbunden. Die Wissenschaft benutzt diese Bezeichnung sowohl für komplexe Strukturen |fig. 1, 2,3 | als auch für die Beschreibung von Verhalten komplexer Systeme in der Naturahlen Welt. |fig. 4|

Ein wichtiger Kern dieses Begriffes wird auch zunehmend in der Entwicklungsbiologie in der Physikalischen Chemie und in der Mathematik verwendet. Die Technik und die Entwicklung von Emergenz ist äußerst mathematisch und strahlt in andere Bereiche aus in denen die Analyse und die Produktion von komplexen Verhalten wichtig sind.

Diese Arbeit beschäftigt sich unter anderem auch damit, wie sich ein entwickeltes mathematisches Modell (Schwarmverhalten) benutzt werden kann um Design, Architektur, Formen und Strukturen zu generieren.

|1| Gaudi “Casa Milà”
Belastungsmodell einer Turmspitze

In diesem Beispiel sieht man, wie Gaui physikalische Modelle aufbaut und Form und Belastung untersucht um die für Ihn beste Lösung zu finden

Artifizielle Evolution (siehe 1.2 S. 19) ist nicht wirklich evolutionär ohne dass sie mit physikalischen Modellen arbeitet. Noch können wir Systeme produzieren die Emergenz benutzen ohne dass sie auch die Selbstorganisation materieller Effekte des Formfindungsprozesses und der Industriellen Produktion beinhalten.

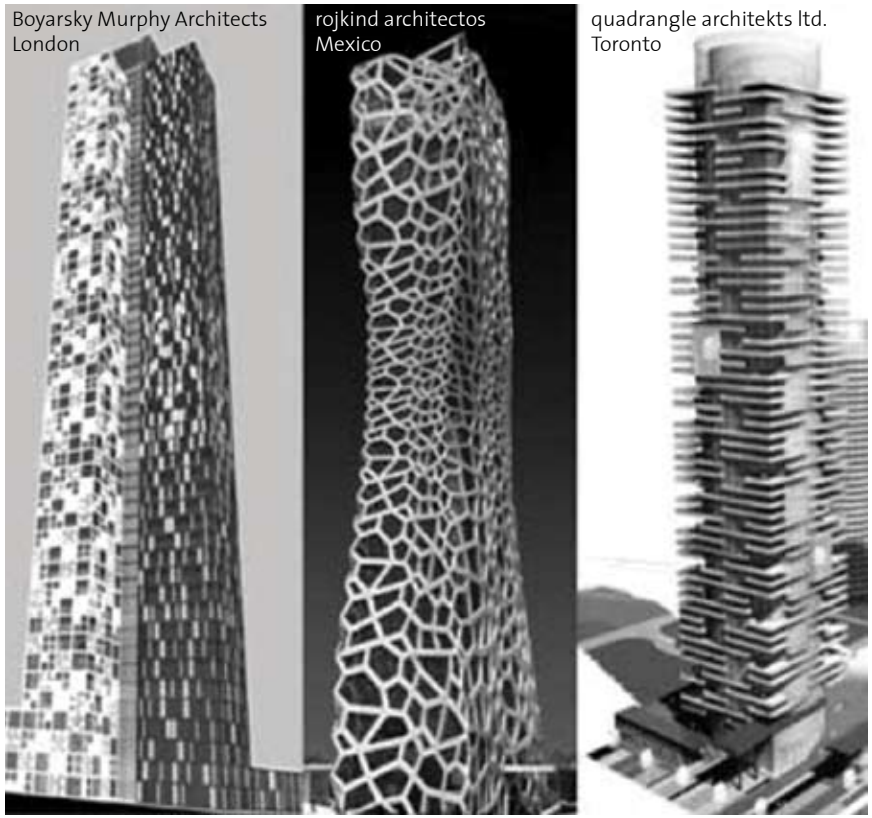
Emergenz fordert die Wiederernennung von Gebäuden nicht als einzelne fixe Körper sondern als komplexe dynamische Gebäudemodelle im Kontext zu anderen Gebäuden und als Wiederholung einer langen Serie die sich vorsetzt zu einer evolutionären Entwicklung bis hin zu einem intelligenten Ökosystem.

Schon Gaudi hat sich mit Physikalischen Formfindungsprozessen beschäftigt. | fig. 1|

Frei Ottos Pionierarbeit in der Formfindung ist gut dokumentiert und in „ Frei Otto in Konversation and DesignGroup“ diskutiert er seine Entwicklung von gedachten Formfindungstechniken in Bezug auf sein Interesse an natürliche Systeme, an dass Verhältnis von Experimentellen Modellen zur Geometrie und an Iterative Mathematik. |fig. 2 S. 18|

Neue Formfindungsmethoden werden für adaptive Strukturen gebraucht und mit Hilfe der Emergenz wird dies erst möglich.

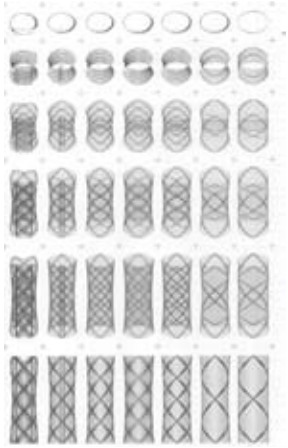
Zusammenfassend ist zu sagen das die so strukturierten Systeme auf der Makroebene Eigenschaften haben, die auf der einfacheren Organisationsebene, der Mikroebene, nicht vorhanden sind. Sie entstehen durch synergetische Wechselwirkungen zwischen den Elementen auf der Mikroebene.



Hochhäuser (competition “city of mississauga”, toronto)

1.3 Architektur

Die Architektur mit ihrer Vielzahl an Problemstellungen bietet ein Feld an dem man das Verhalten verschiedener emergenter Algorithmen experimentell untersuchen kann. Seit jeher ist der Architekt bemüht allen Anforderungen des Bauens zu genügen. Diese Masse an Anforderungen programmiert, ermöglicht dem Architekten Varianten zu generieren diese zu beurteilen und schlussendlich die für ihn gestalterisch beste Lösung zu verwenden. |fig. 1|



|1| “Fit Fabric”

evolutionörer Prozess einer Tragstruktur für ein Hochhaus.
(Emergence and Design Group)

Dies bedeutet, dass man ein höheres Maß an Komplexität erhält. - Erreicht wird ein weiterer Schritt zu einem nahezu perfekten System...

2. Idee

„Schönheit ist das Resultat einer geistigen, auf wissenschaftlichen Erkenntnissen basierenden Ordnung“. (Max Bill)

2.1 Formfindung durch intelligente Partikel

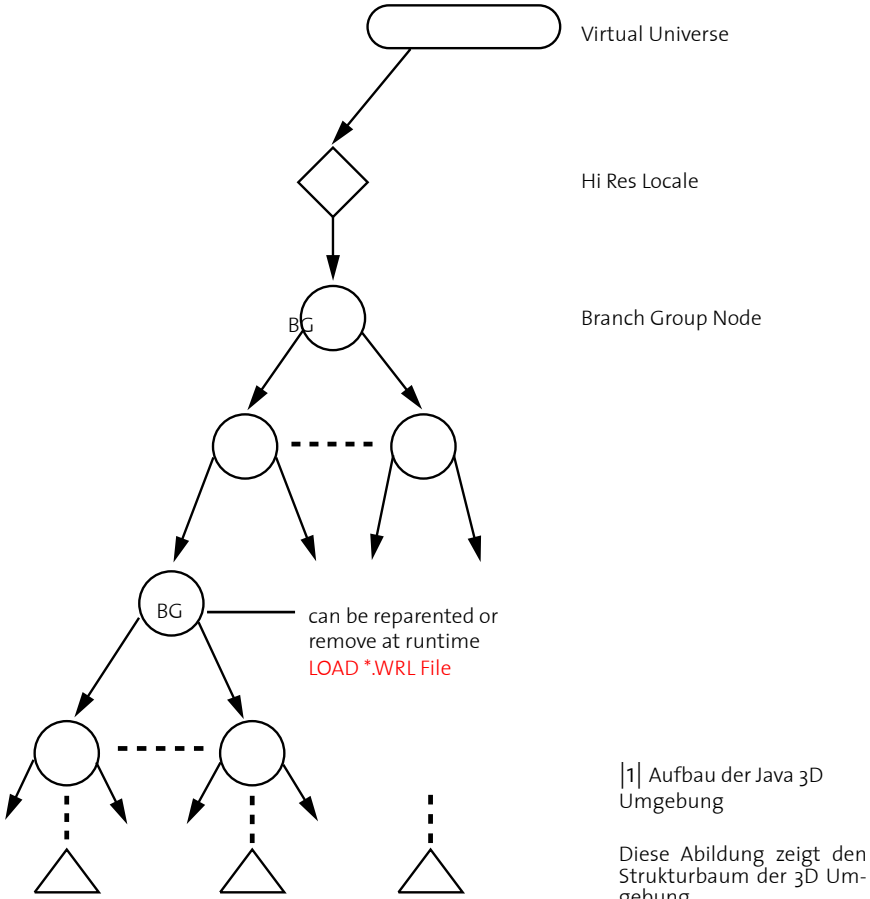
In der Natur findet man viele unsichtbare Kräfte z. B. Magnetfelder, Über- und Unterdruck und Strömungen. Architekten waren schon immer auf der Suche nach Allegorien. Oft beschränkte sich der architektonische Entwurf aber auf die Interpretation gegenständlicher Allegorien „...sieht aus wie ein Vogel...“. Mein Bestreben war es eine „unsichtbare“ Kraft sichtbar zu machen.

Eine Allegorie aus dem Unsichtbaren.

2.2 Schwarm

Vogelschwärme in Ihrer ganz eigenen Ästhetik boten mir die Ideale Voraussetzung diese unsichtbaren Kräfte der Natur sichtbar zu machen. So wollte ich den Weg der Vögel nachzeichnen (Langzeitbelichten). Die schwungvolle Bewegung der Vögel sollte Skulptur, sollte Raum bilden und werden.

Das Schwarmverhalten ist im Kapitel 3.2.3 näher beschrieben.



|1| Aufbau der Java 3D Umgebung

Diese Abildung zeigt den Strukturbaum der 3D Umgebung

3. Programmierung

3.1 Umgebung definieren

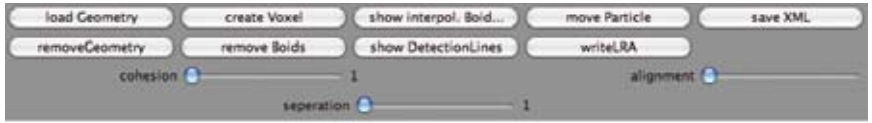
3.1.1 Geometrielader

Als Voraussetzung für den kontrollierten Partikelflug ist es notwendig die Partikel in einem festen vorgegebenen Volumen “fliegen” zu lassen. Hierzu kann das Programm ein beliebiges Volumen einladen. Die Volumen können in jeder 3D Umgebung entwickelt werden (z. B. Cinema 4d, Maya) und zum Einlesen in die Java-3D-Umgebung wird das Volumen im WRL-File Format abgespeichert. |fig. 1,2|

Funktionen für “load Geometry”:

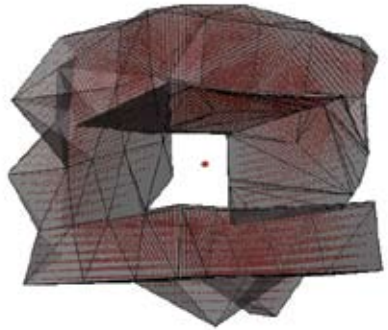
private BranchGroup **loadVrmlFile**(String location){ . . . }
public void **loadObject()** { . . . }

|2| Userinterface



|1| Screenshot

Das Bild zeigt eine freie Geometrie und die roten Punkte zeigen das Voxelgrid innerhalb des Volumens.



Exkurs: Voxelgrid

Der Begriff Voxel stammt aus der 3D-Computergrafik und setzt sich aus den Wörtern volumetric und pixel zusammen. Er bezeichnet zwei Dinge:

1| Datensatz

Bei einem räumlichen Datensatz, der in diskretisierter Form in kartesischen Koordinaten vorliegt, bezeichnet Voxel den diskreten Wert an einer XYZ-Koordinate des Datensatzes. Bei dieser Definition handelt es sich um das dreidimensionale Äquivalent eines Pixels, somit hat ein Voxel keine bestimmte Form. Man spricht hier auch vom isotropen Voxel oder „Volumenpixel“.

2| quaderförmige Zelle

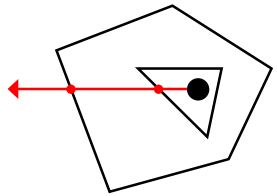
In anderen Fällen bezeichnet ein Voxel eine quaderförmige Zelle innerhalb eines regelmäßig aufgeteilten Quaders oder unbegrenzten Raumes. Am häufigsten wird diese Bedeutung bei bestimmten Techniken zur Beschleunigung von Raytracing oder bei bestimmten Verfahren der numerischen Simulation verwendet. (wikepedia)

3.1.2 Algorithmus zur Volumenfindung

Um die Partikel in einem bestimmten Volumen „fliegen” zu lassen muss auch der Rand und das innere Volumen vom Programm erkannt werden. Hierzu wird ein „Voxelgrid” angelegt. Nachdem die Geometrie aus dem *.WRL-File eingeladen wurde werden die einzelnen Voxel auf ihre Lage im 3D Körper überprüft. |fig 1,2|

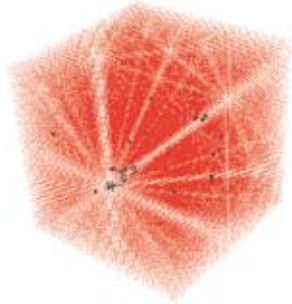
Dazu muss erkannt werden ob der Voxel innerhalb oder außerhalb liegt. Folgende Logik erkennt die Lage im Volumen:

„Liegt ein Punkt Innerhalb, schneidet ein Strahl der vom Voxel bis zum Rand des Voxelfeldes gezogen wird, die Randlinien der Polygone in einer ungeraden Anzahl“.|fig 3|



|3| Position des Voxelpunktes innerhalb der Geometrie

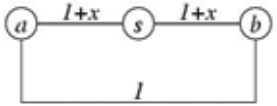
Der Strahl schneidet das Volumen 2mal -der Punkt liegt also außerhalb.



|2| Screenshot

Das Bild zeigt den gesamten Voxelraum des JAVA-Programms. Innerhalb dieses Rasters wird das eingeladene Volumen berechnet.

Edsger Wybe Dijkstra
Rotterdam, May 11, 1930
– Nuenen, August 6, 2002



Pseudocode

```
algorithm Dijkstra(G, s) { berechne alle kürzesten Wege von s zu allen
anderen Knoten in G }

ROT := {s}; GRÜN := {}; dist(s) := 0;
while ROT <> {} do
    wähle v aus ROT, so dass dist(v) minimal von allen Knoten aus ROT;
    färbe v grün;
    for each Nachfolger w von v do
      if w nicht in ROT oder GRÜN
      { w wurde also noch nie besucht }
      then färbe (v, w) grün;
         färbe w rot;
         dist(w) := dist(v) + c(v, w);
      elseif w in ROT
      { w wurde bereits besucht, aber noch nicht abschließend }
      then if dist(w) > dist(v) + c(v, w)
         then färbe (v, w) grün;
            färbe die bisherige grüne Kante zu w rot;
            dist(w) := dist(v) + c(v, w);
         else färbe (v, w) rot;
         fi
      else
      { w ist also in GRÜN und schon abschließend betrachtet }
         färbe (v, w) rot;
      fi
    od
od.
```

Exkurs: “Dijkstra-Algorithmus”

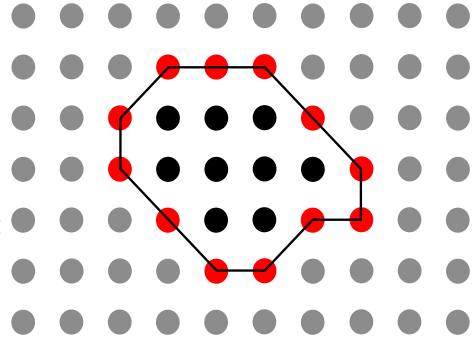
Auch Link-State-Algorithmus. Der Algorithmus von Dijkstra (nach seinem Erfinder Edsger W. Dijkstra) dient der Berechnung eines kürzesten Pfades zwischen einem Startknoten und einem beliebigen Knoten in einem kantengewichteten Graphen. Die Gewichte dürfen dabei nicht negativ sein.

3.1.3 Algorithmus zur Randdefinition

Nachdem die innerhalb des Volumens liegenden Voxel erkannt und markiert wurden muss, damit die Partikel nicht das Volumen verlassen der Rand definiert werden. Ein beschränkter Dijkstra-Algorithmus definiert die Randzone indem er die Anzahl der Nachbarn die im Volumen liegen zählt. Ist die Zahl kleiner als 6 muss der Voxel am Rand liegen. Der Graph hat den kürzesten Spannbaum. |fig. 1|

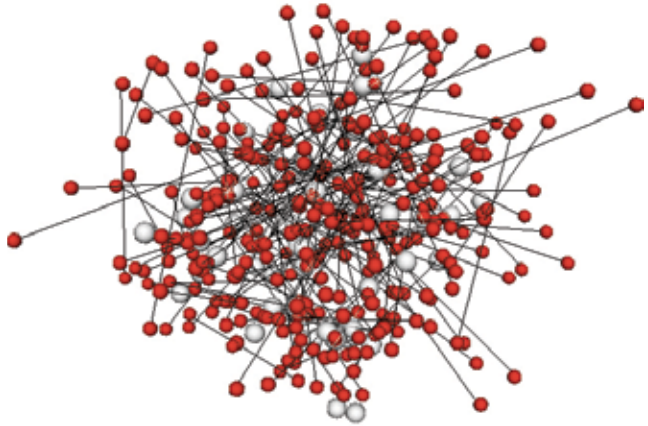
Programmausschnitt:

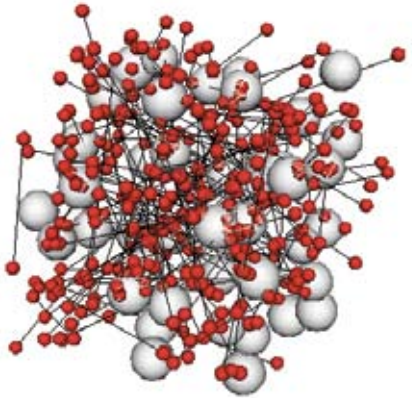
```
if (anzNachbar < 6) {
            voxel[x][y][z].setVoxelColor(2);
            randVoxel.add(voxel[x][y][z].getVoxelPoint());
            // randVoxel.add(voxel[x][y][z]);
}
```



|1| Voxelraster mit Randmarkierung

die roten Punkte zeigen den Rand da sie wenig als 4 Nachbarn haben (2D)



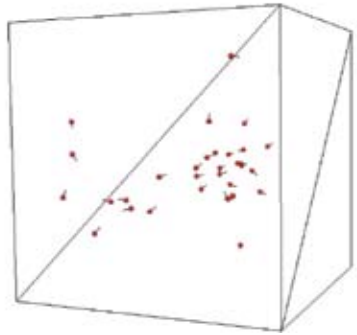


|2| Screenshots

erste Tests mit Bewegungsvectoren.

3.1.4 Positionierung der Partikel

Um die Schwarmpartikel im Volumen zu positionieren werden per Zufallszahlgenerator Werte für X, Y, Z Koordinaten generiert. Liegt dieser Wert innerhalb des Volumens und hat der Punkt einen gewissen Abstand zum Rand wird der Partikel an dieser Position gesetzt und die Postition wird im Objekt gespeichert. Das Voxelgrid ist hier auch die Grundlage für die Positionierung |fig. 1|

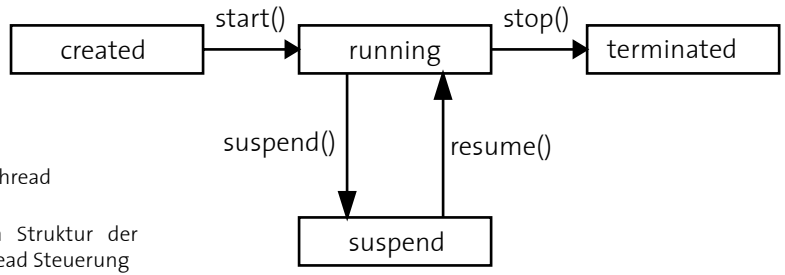


|1| Screenshot

erste Postitionierung der Partikel im Volumen

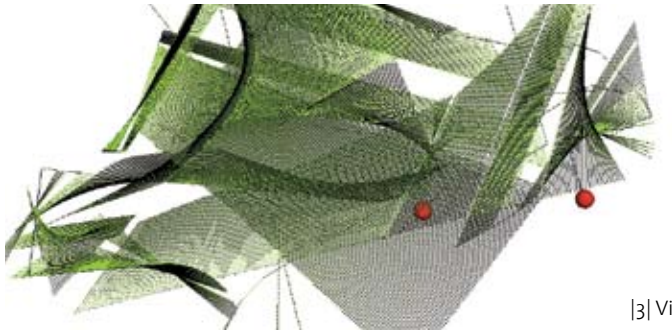
3.1.5 Bewegung der Partikel

Jedem Partikel wird mit der Positionierung ein Richtungsvector zugeordnet. Die Richtung wird durch einen Zufallszahlgenerator ermittelt und in den Partikeln gespeichert. Die erste Bewegung wird so anhand des gespeicherten Richtungsvektors ausgeführt. |fig. 2|



|1| Thread

Java Struktur der Thread Steuerung



|3| Visualisierung

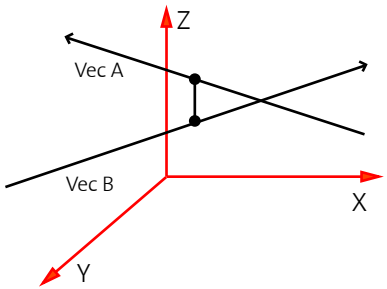
Die Abbildung zeigt das Linienarray zur Kollisionsüberprüfung

3.2 Runtime

Nach der Initialisierung des Volumens und der Positionierung der Partikel im Volumen beginnt das Programm, das intelligente Verhalten der Paritikel abzubilden. JAVA bietet die Möglichkeit zur Laufzeit des Hauptprogramms ein Unterprogramm auszuführen. In diesem Thread wird die Bewegung der Partikel nach bestimmten Prioritäten ausgeführt. |fig. 1|

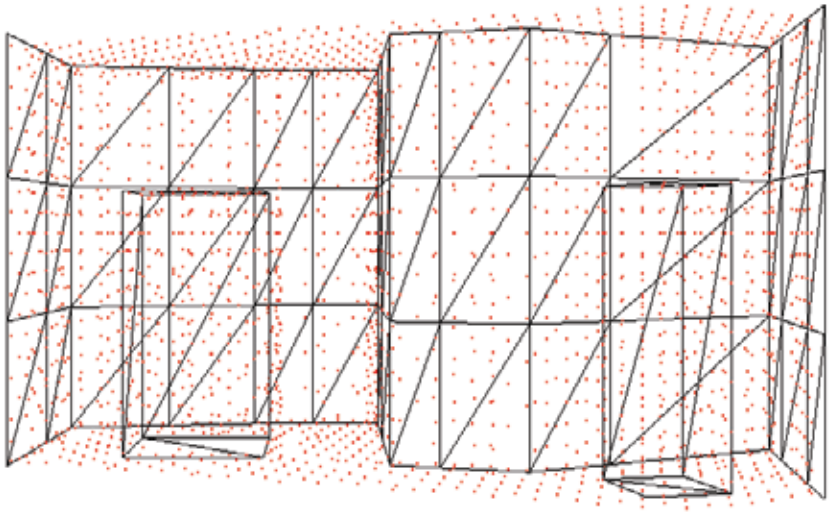
3.2.1 erste Priorität - Kollisionscheck

Damit die verfolgten Linen sich nicht schneiden, da für die Produktion und Fertigung keine Überschneidungen gewünscht sind muss, bevor die Bewegung im Thread ausgeführt wird, die neue Position auf Kollision überprüft werden.



|2| CPA Raumskizze der Berechnung

Ein Formel wie sie auch im Flugverkehr Anwendung findet wurde also im Bewegungs ablauf integriert. - Closest Point of Approach CPA |fig. 2,3|



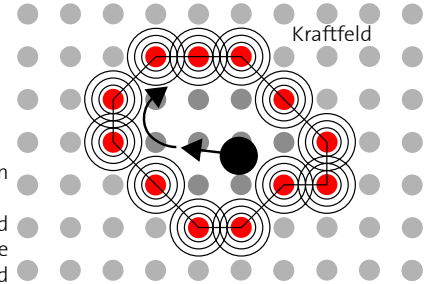
|2| Screenshot Geometrie mit Voxelfeldbegrenzung

Das Bild zeigt eine freie Geomtrie mit dem Kraftfeld (rote Punkte)

3.2.2 zweite Priorität - Randkollision - Kraftfeld

Tritt bei der ersten Priorität keine Kollision auf wird überprüft ob sich der Partikel im Randbereich befindet. Ein Kraftfeld, welches durch die Randvoxel aufgebaut wird lenkt das intelligente Partikel sucsessiv wieder in das “freie” Volumen. |fig. 1,2|

Im Entwicklungsprozess des Programms und durch den Abgleich der Maschinendaten hatte sich gezeigt, dass eine Ablenkung die in etwa der Ablenkung einer Billiardkugel von der Bande entspricht nicht praktikabel war. Einfallswinkel = Ausfallswinkel. d. h. sehr spitze Einfallswinkel ergeben einen sehr Spitzen Ausfallswinkel.

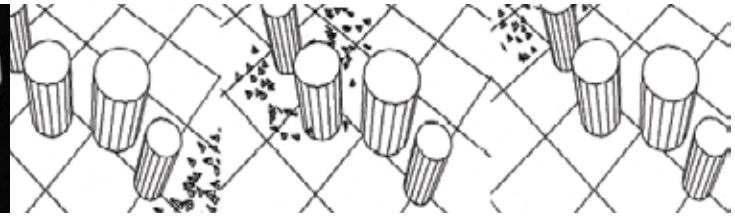


|1| Kraftfeldsystem

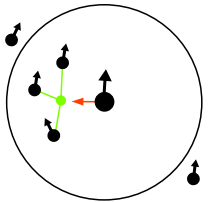
Die roten Punkte zeigen den Rand des Volumens. Die Ringe um diese Punkte zeigen das Kraftfeld

Die Rohrbiegemaschine gibt jedoch einen bestimmen Winkelradius vor. So wurde ein “Kraftfeld” aufgebaut im dem sich die Partikel bewegen können und die Ablenkung vom Rand ist durch die Kraftfeldgröße steuerbar. Die Partikel bewegen sich gleichmäßig vom Rand ins Volumen und der Ablenkungswinkel ist von der Rohrbiegemaschine produzierbar.

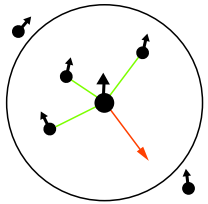
|1| Craig W. Reynolds



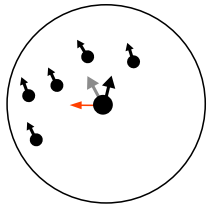
|2|Screenshot von Reynolds programmiertes Schwarmverhalten



Alignment
Anschluss



Seperation
Vereinzelung



Cohesion
Bindekraft

Exkurs: “Boids”

Die gezeigten Diagramme erläutern die drei Verhaltgensweisen zur Steuerung eines Schwarms
Ein Boid (“bird-oids“ = animierte Vögel) ist ein simuliertes Objekt mit vogelartigen Verhalten.

3.2.3 dritte Priorität - Schwarmverhalten

Wenn keine der vorangegangenen Kollisionen stattgefunden hat wird das eigentliche Schwarmverhalten ausgeführt.

Craig Reynolds entwickelte im Jahr 1986 einen Algorithmus der das Verhalten eines Schwarms abbildet. Er benutzte 3 Steuerungsmechanismen die je nach Gewichtung das Verhalten der Partikel (Boids) beeinflussen. | fig. 1|

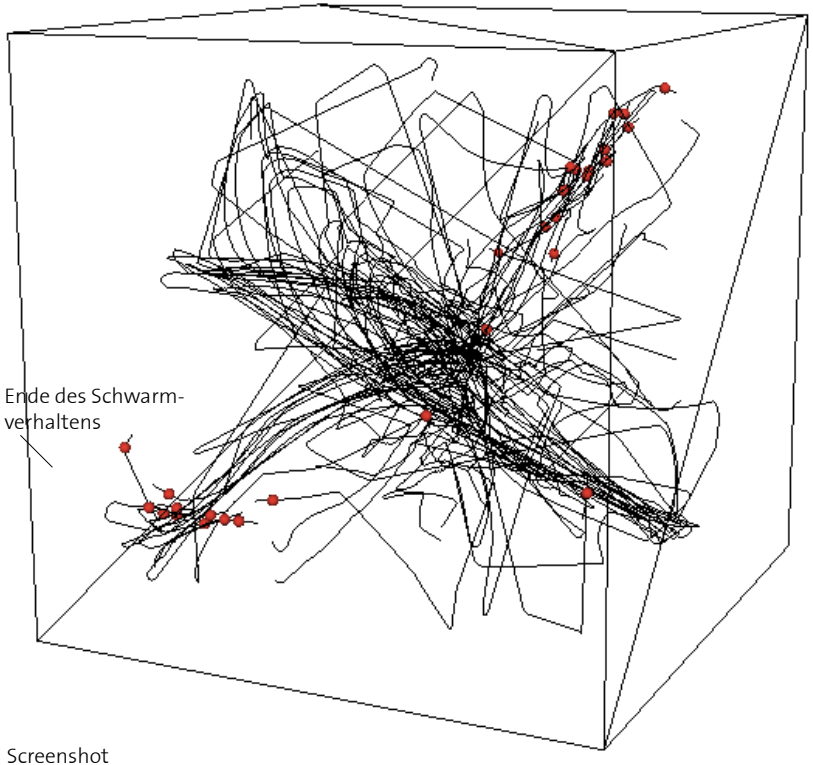
a. Alignment : Bewege dich in Richtung des Mittelpunkts derer, die du in deinem Umfeld siehst.

b. Seperation: Bewege dich weg, sobald dir jemand zu nahe kommt.

c. Cohesion: Bewege dich in etwa in dieselbe Richtung wie deine Nachbarn.

Diese drei einfachen Regeln, deren Zusammenspiel eine Schwarmstruktur begünstigt, ergeben ein ganz natürliches und von außen betrachtet komplexes Verhalten. Das Verhalten setzt sich aus all den lokalen Aktivitäten zusammen. Aus den lokalen Interaktionen entsteht das Gesamtverhalten und der Gesamtzustand beeinflusst wiederum die lokalen Bedingungen. |fig. 2|

Die Entwicklung eines solchen Systems ist komplex. Doch das Gesamtbild ergibt eine Struktur, so dass man von Emergenz sprechen kann.

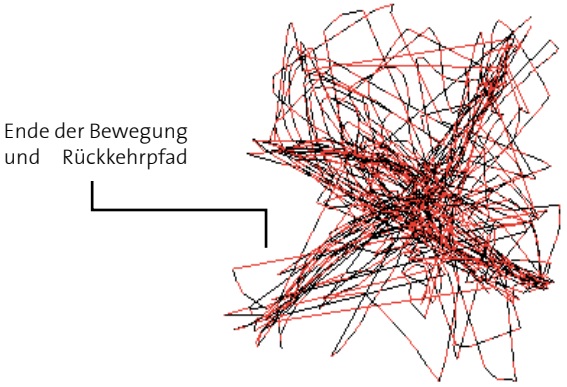


|1| Screenshot

das Bild zeigt den Zustand nachdem der Zeitfaktor erreicht ist. Die Partikel kehren an dieser Stelle zu Ihrem Startpunkt zurück

3.2.4 Vierte Priorität: Rückkehr zum Startpunkt

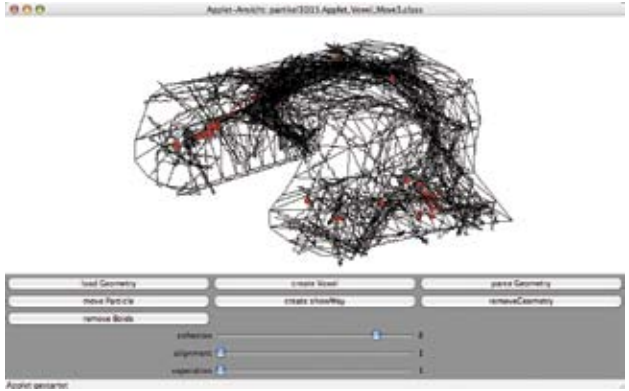
Parametrisch kann die Zeit festgelegt werden, wie lange das Schwarmsystem im Volumen fliegt. Ist ein bestimmter Zeitfaktor überschritten, verhalten sich die Partikel nicht mehr wie ein Schwarm, sondern sie kehren unter Berücksichtigung des Kraftfeldes und der Kollisionsüberprüfung mit den gespeicherten Wegen zum Startpunkt zurück. |fig. 1|



|2| Screenshot
Interpolierte Wege

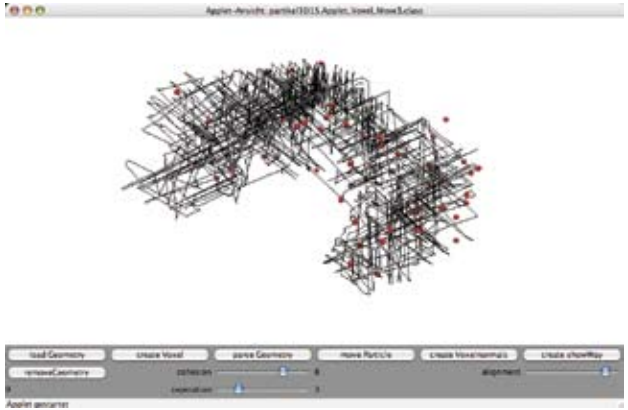
das Bild zeigt den Weg nach der Rückkehr zum Startpunkt. Den Weg sieht man im Vergleich zum Screenshot 1

Dieses Verhalten erzeugt einen Ring, so dass in der Realisation der Struktur keine Anfangs und Endstücke ausgebildet werden müssen. Das ermöglicht es, dass die gesamte Stuktur mit nur einem Verbindungsdetail aufgebaut werden kann |fig. 2|



|2| Screenshot 1

In diesem Beispiel sieht man durch die starke Gewichtung des Sliders “Cohesion = 8”, dass sich der Partikelflug verdichtet.



|3| Screenshot 2

Die Sliderwerte stehen hier auf:

Cohesion:	8
Alignment:	9
Seperation:	3

Die Struktur ist offener

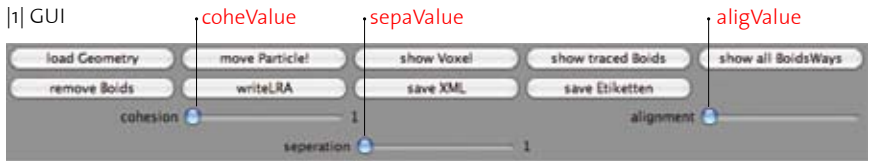
3.2.5 parametrische Änderungen in Echtzeit durch Nutzer

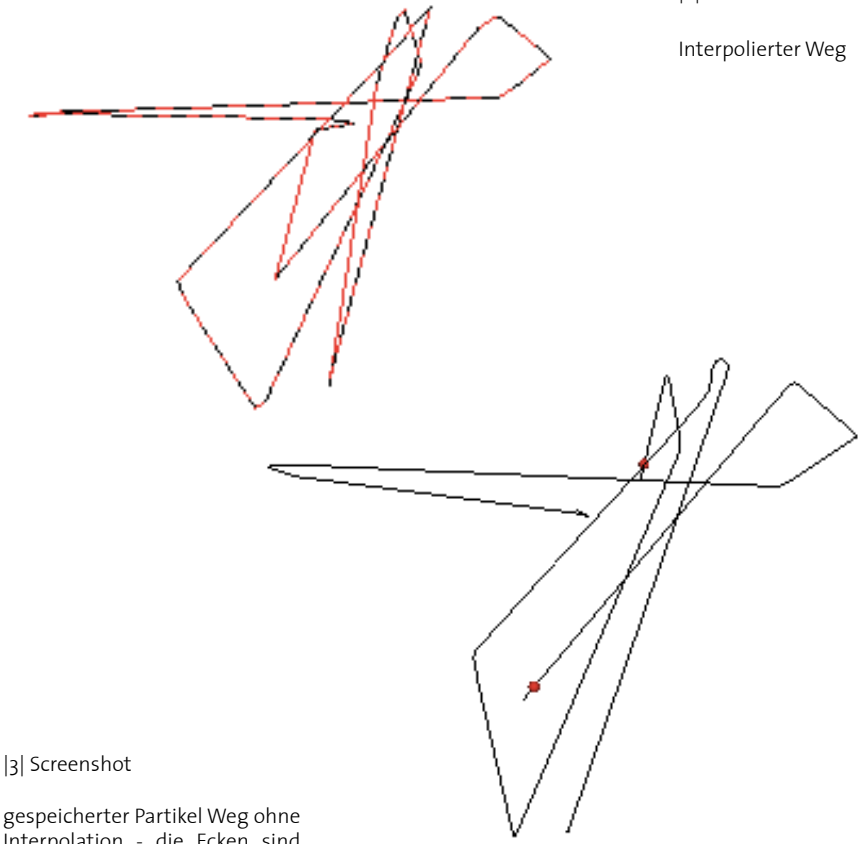
Während des Partikelflugs kann der Nutzer das Verhalten der Partikel ändern. Durch drei Slider |fig. 1| in der JAVA-Benutzeroberfläche kann die Gewichtung von Alignment, Cohesion, und Seperation geändert werden. Es ist so möglich den Schwarmflug aktiv zu beeinflussen und somit auch die spätere Geometrie. |fig. 2,3|

Java-Funktion des Schwarmverhaltens:

```
public void flocking() {
        double coheValue = (Applet_Voxel_Move12.cohesionValue * 2.0d);
        double aligValue = (Applet_Voxel_Move12.alignmentValue * 0.5d);
        double sepaValue = (Applet_Voxel_Move12.seperationValue * 0.1d);
        Vector3f fFlock = new Vector3f();
        double fflockX = ((fc.x * coheValue) + (fa.x * aligValue) + ((fs.x * sepaValue) * 0.33));
        double fflockY = ((fc.y * coheValue) + (fa.y * aligValue) + ((fs.y * sepaValue) * 0.33));
        double fflockZ = ((fc.z * coheValue) + (fa.z * aligValue) + ((fs.z * sepaValue) * 0.33));
        fFlock = new Vector3f((float) fflockX, (float) fflockY, (float) fflockZ);
        fFlock.normalize();
        fFlock.scale(0.01f);
        mytmpBoid.setMyRichuntungsVecBoid(fFlock);
}
```

|1| GUI





|2| Screenshot

Interpolierter Weg

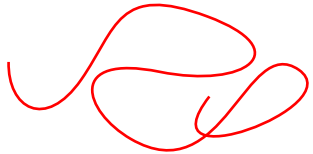
|3| Screenshot

gespeicherter Partikel Weg ohne Interpolation - die Ecken sind “runder”

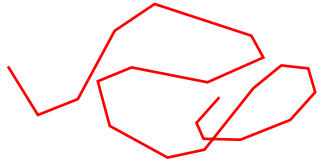
3.3. Vorbereiten der Daten für die Produktion

Nach Ablauf des Bewegunsalgorithmuses müssen die erhaltenen Daten noch für die Produktion aufbereitet werden. Dies geschieht in den folgenden Schritten.

ohne Interpolation:



mit Interpolation:



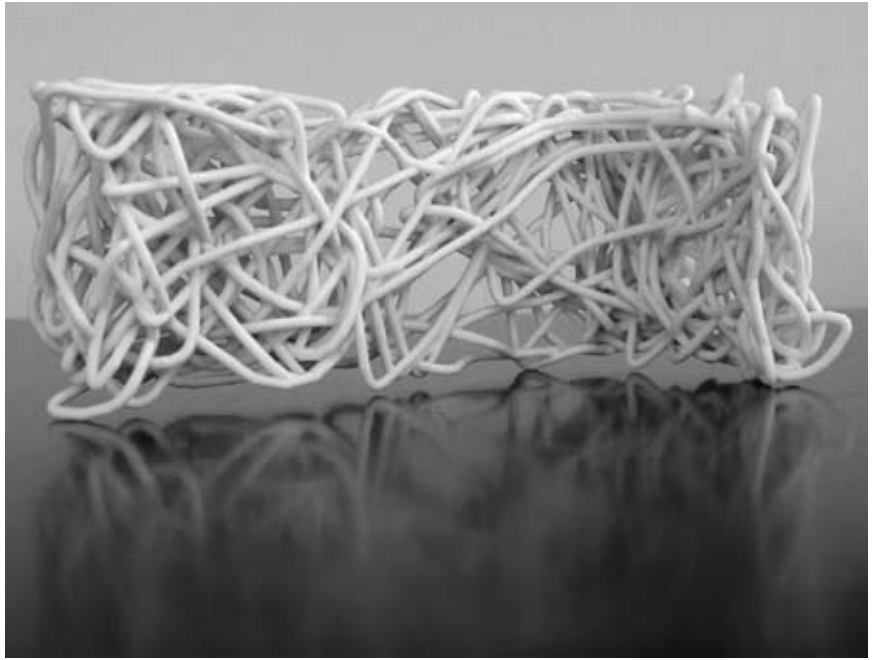
|1| Skizzen interplierte Bewegung

die zwei Linien zeigen den Partikelflug einmal vor und nach der Interpolation

3.3.1 Interpolation der zurückgelegten Bewegungen

Die Granularität der Daten in der gespeicherten Bewegung ist, um Sie als Maschinendaten auslesen zu können zu hoch. Daher wird nach Abschluss der Bewegung der gespeicherte Weg jedes Partikel interpoliert. |fig. 1,2,3|

Die Ergebnisse dieser Interpolation sind - entsprechend der Anzahl der Partikel- unterschiedliche Polygone. Das Programm ermöglicht die Abspeicherung der Polygondaten im XML-Format, und damit die erste Möglichkeit die entstandene Struktur auf einem 3D Drucker auszudrucken. |S. 50, 51|



Gipsprint

Nachdem nun die 3D - Daten des Schwarmverhaltens vorliegen konnten erste Test auf einem 3D Drucker gemacht werden. Dieses Bild zeigt schon den, für die fertige Struktur produzierten Ausdruck.

“JavaClass” zur Speicherung der XML Daten

XML Daten:

```
<partikel3D21.FrankTubes>
        <myTubes>
                <javax.media.j3d.LineStripArray>
                <forceDuplicate>false</forceDuplicate>
                        <retained class=”javax.media.j3d.LineStripArrayRetained”>
                                <stripVertexCounts>
                                        <int>79</int>
                                </stripVertexCounts>
                                <stripStartVertexIndices>
                                        <int>0</int>
                                </stripStartVertexIndices>
                                <stripStartOffsetIndices>
                                        <int>0</int>
                                </stripStartOffsetIndices>
                                <vertexFormat>5</vertexFormat>
                                <c4fAllocated>0</c4fAllocated>
                <vertexCount>79</vertexCount>
                <validVertexCount>79</validVertexCount>
                                <vertexData>
                                        <float>0.44679493</float>
                                        <float>0.39926696</float>
                                        <float>0.3473254</float>
                                        <float>1.0</float>
                                                .
                                                .
                                                .

                                        <float>0.47472036</float>
                                        <float>-0.4209083</float>
                                        <float>-0.247938</float>
```

```
[PARA]
DRAWNUMBER=
DESCRIPTON=
TUBEDIAMETER1= 33.7
TUBEDIAMETER2= 0
WALLTHIKNESS= 2
EXECUTEMODE= 2
ENGUNIT= 0
LIMITSTOPMODE= 1
MEASURELENGTH= 0
LIMITSTOPLENGTH= 0
SUPPORTMODE= 1
LOADPOSX= 400
LOADPOSY= 0
LOADPOSZ= 0
SUBPROGFORLOAD= 0

[INFOTEXT]

[BEND1]
TOOLNR= 1
L=319.0081892944537
LV= 50
R=0.0
RV= 50
A=82.0
AV= 50

[BEND2]
TOOLNR= 1
L=447.48247404691745
LV= 50
R=0.0
RV= 50
A=0.0

[TOOL1]
TOOLNAME= D 33.7 RM85 ETH
TOOLID= 999
RM= 85
RME= 85
RD= 33.7
WS= 2
RMAT= ST.37
WKZH= 100
K= 80
GS= 190
DDM= 29.6
KWS3= 110
KWI3= 106.5
KWS4= 180
KWI4= 176

[XYZOFFSET]
X= 0
Y= 0
Z= 0
```

|1| Maschinenprogramm *.PRG

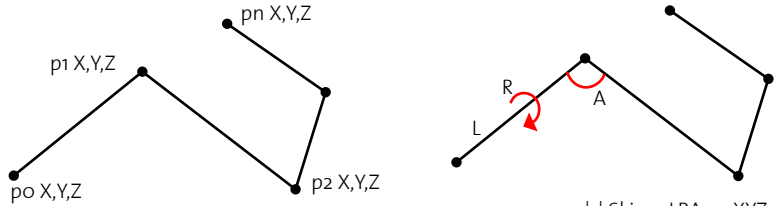
Dies ist der Maschinencode der Rohrbiegemaschine. Die rot markierten Zeilen sind von dem JAVA-Programm geschrieben worden. Die Maschine arbeitet die [Bends] sucsessiv ab. Unter der Routine [PARA] sind Angaben über Rohrdurchmesser, Material etc. abgespeichert.

3.3.2 Umwandlung der Raumkoordinaten in LRA Daten

Die Polygone der interpolierten Wege liefern Raumkoordinaten, da aber das Maschinenprogramm der Rohrbiegemaschine diese Daten nicht verarbeiten kann müssen diese in LRA-Daten umgewandelt werden. |fig. 1|

LRA Daten sind Informationen über die Länge, die Rotation und den Winkel von einem Raumpunkt zu einem anderen. Diese Umrechung liefert eine vom MAS-Team entwickelte JAVA-Class.

Nachdem die LRA-Daten generiert wurden, werden vom Programm die Steuerungsprogramme für die einzelnen Rohre geschrieben und auf der Festplatte abgespeichert.



|1| Skizze LRA zu XYZ

Die Firma MEWAG, Waasen, Schweiz entwickelte ein “lesbares” Steuerungsprgramm, so dass die für die Steuerung der Maschine notwendigen Informationen, wie z. B. der Biegekopf, der Rohdurchmesser etc. in dem von der JAVA-Klasse erzeugte Programmcode eingearbeitet werden konnte. Der so erzeugte Programmcode wird dann in die Software der Maschine geladen. |fig. 2|

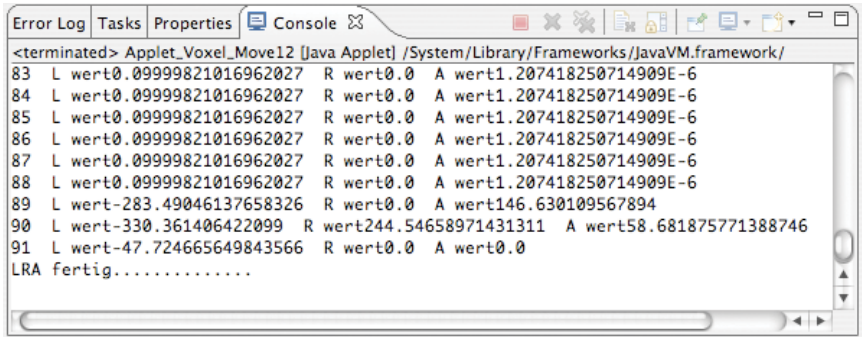
```
gesamt Rohre10
laenge gesamt10460.381755165145  (rohdaten)
-------------------------------
Rohr Nr.1
  L:223.595821699333358 R:0.0  A:70.3489098916333
  L:421.351772128901 R:30.021636031011383  A:74.19729488984935
  L:412.469035677776826 R:0.0  A:0.0
Rohrlaenge: 1271.8552410554046
Gesamtlaenge: 1271.8552410554046
-----------------
Rohr Nr.2
  L:212.469035677776826 R:-67.12968894785595  A:66.474542637230 66
  L:440.38208773550883 R:94.698808561673  A:56.914125158495466
  L:437.82512447477444 R:0.0  A:0.0
Rohrlaenge: 1273.7270214662692
Gesamtlaenge: 2545.582262521674
-----------------
Rohr Nr.3
  L:237.82512447477444 R:-68.79485093775669  A:31.078865843217727
  L:442.626603972629 R:14.785595369476992  A:83.86231057123223
  L:413.45331497533925 R:0.0  A:0.0
Rohrlaenge: 1264.423711259306
Gesamtlaenge: 3810.0059737809797
-----------------
Rohr Nr.4
  L:212.469035677776826 R:-67.12968894785595  A:66.474542637230 66
  L:440.38208773550883 R:94.698808561673  A:56.914125158495466
  L:437.82512447477444 R:0.0  A:0.0
Rohrlaenge: 1273.7270214662692
Gesamtlaenge: 5545.582262521679
-----------------
Rohr Nr.5
  L:223.595821699333358 R:0.0  A:70.3489098916333
  L:421.351772128901 R:30.021636031011383  A:74.19729488984935
  L:412.469035677776826 R:0.0  A:0.0
```

|2| Rohrliste

3.3.3 Rohrlisten erstellen - Länge, Stücknummer, Position

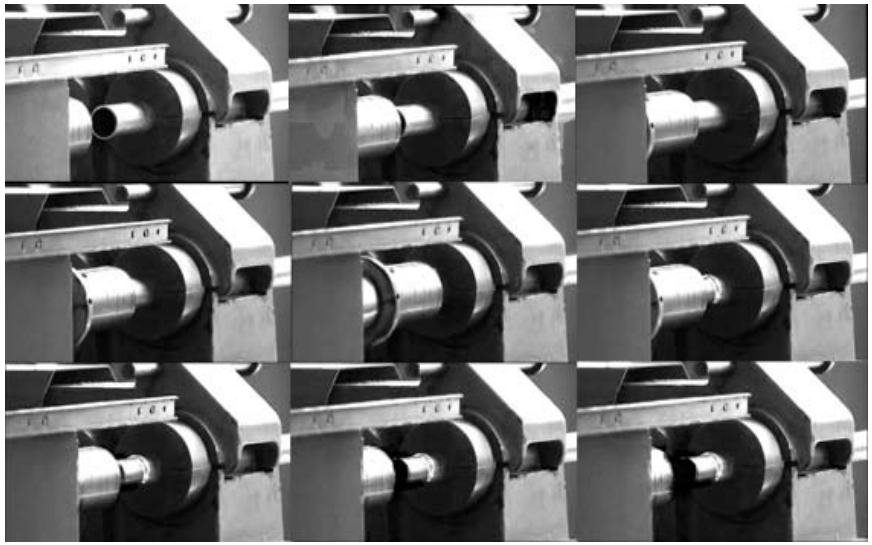
Bekannt sind jetzt alle für die Maschine notwendigen Daten. In der JAVA - Umgebung werden jetzt im Systemfenster die einzelnen Rohrlisten angezeigt so dass man diese ausdrucken und für die weitere Produktion verwenden kann.

Mit der “Copie and Paste”-Funktion des Systems werden die Daten an ein Textprogramm übergeben und die so erhaltenen Dateien werden abgespeichert und für den handwerklichen Prozess verwendet. |fig. 1|



|1| Screenshot Systemfenster

Die Rohrlisten enthaltenen Angaben über die Anzahl der Biegungen die LRA-Daten jeder einzelnen Biegung sowie die abgewickelte Länge der Rohre. |fig. 2|



|2| Umformprozess

Die dargestellten 9 Bilder zeigen den Umformprozess des Umformwerkzeugs Das Rohr wird in zwei Spannbacken eingespannt und durch das Aufschieben eines hohlen Stahlzylinders wird das Rohr auf den Innendurchmesser gestaucht.

|3| Rohrsteckverbindung

3.4. Produktion der Strukturen/Konstruktionen

Nach Erläuterung der entwickelten Software wird im folgenden Teil die Produktion der Rohre und das Aufbauen der Struktur erklärt.

|1| Rohrenden mit Teilemarkierung

3.4.1 Schneiden der Rohre

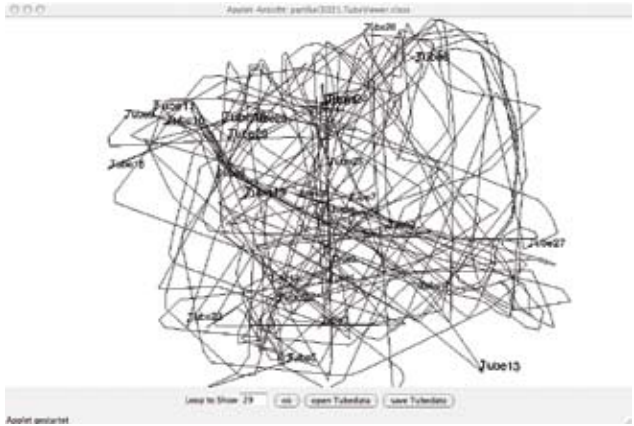
Die Rohre werden anhand der Rohrstückliste (S. 55) auf die abgewickelte Länge geschnitten.

3.4.2 Rohrende Umformen

Um die Rohe später aneinanderfügen zu können wird ein Rohrende mit einem besonderem Werkzeug umgeformt. |fig. 1,2,3|

3.4.3 Daten in die Rohrbiegemaschine laden

Die Softwaresteuerung der Rohrbiegemaschne übernimmt ein Windows - Rechner mit einem von der MEWAG entwickelten Rohrbiegeprogramm. Über einen USB- Anschluss konnten die im Laptop erzeugten *.PRG - Files (S. 52) in den Computer geladen werden. Die einzelnen Programme wurden dann in einer Datenbank abgelegt und sucsessiv für die Produktion aufgerufen.



|2| Screenshot "Tubeviewer"

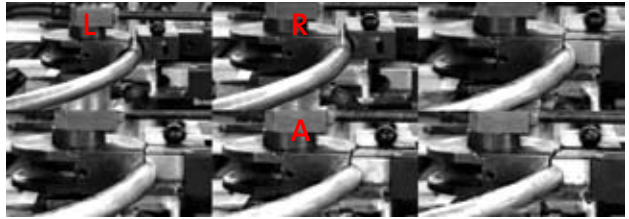
Das Bild zeigt die Gesamtstruktur

|3| Screenshot "Tubeviewer"

Das Bild zeigt denn roten Pfad zur Veranschaulichung des Weges.

3.4.4 Biegen der Rohre

Nachdem die Daten in die Maschnine geladen wurden, kann der Biegeprozess beginnen. Die Rohrbiegemaschine beginnt die LRA Daten auszulesen. Zuerst wird das Rohr vorgeschoben L. Dannach erfolgt die Rotation des Rohres R und im zum Abschluss eines [bends] wird der Winkel A gebogen. |fig. 1|



|1| Biegeprozess

3.4.5 Aufbau der Struktur

Die Produktion der einzelnen Rohre erfolgte bei der Firma MEWAG im Emmental, Schweiz. Alle Rohre wurden dannach an die ETH - Zürich gefahren und dort aufgebaut. Markierungen der einzelen Rohre und die ausgedruckten Stücklisten halfen bei dem Zusammenbau.

Da es sich um eine sehr komplexe Struktur handelt versagen die "normalen" 2D -Aufbaupläne, so dass ein speziell für die Struktur entwickelter 3D Betrachter programmiert wurde. Dieser Betrachter ermöglicht die Verfolgung der einzelnen Rohre und somit ein genaues Aufbauen der Struktur. |fig. 2, 3|

|1| Struktur

von Benjamin Dillenburger

|2| Struktur

"Swarmintelligenz"
von Frank Theßeling

|3| Struktur

von Tobias Wendt

3.5 Präsentation

3.5.1 Vernissage 5. September 2006 um 18:00 Uhr

Schlussendlich wurden die im Rahmen der in Gruppenarbeit entstandenen Pavillions an der ETH-Zürich aufgebaut. Der Produktionsprozess war bei allen gleich, jedoch basieren alle Strukturen auf unterschiedlichen Programmen und Algorithmen. | fig. 1, 2, 3, 4|

|4| Präsentationstisch

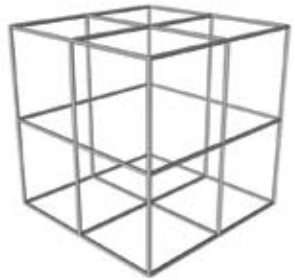
Thomas Feuerstein, „Thomas Hobbes“, Soziale Emergenz II

4. Analysen

4.1 Programmeinstellungen

4.1.1 Schwarmsystemparameter

Durch die Änderung der Schwarmparameter Alignment, Cohesion und Seperation ist es möglich in einem Volumen verschiedenste Varianten zu erzeugen.



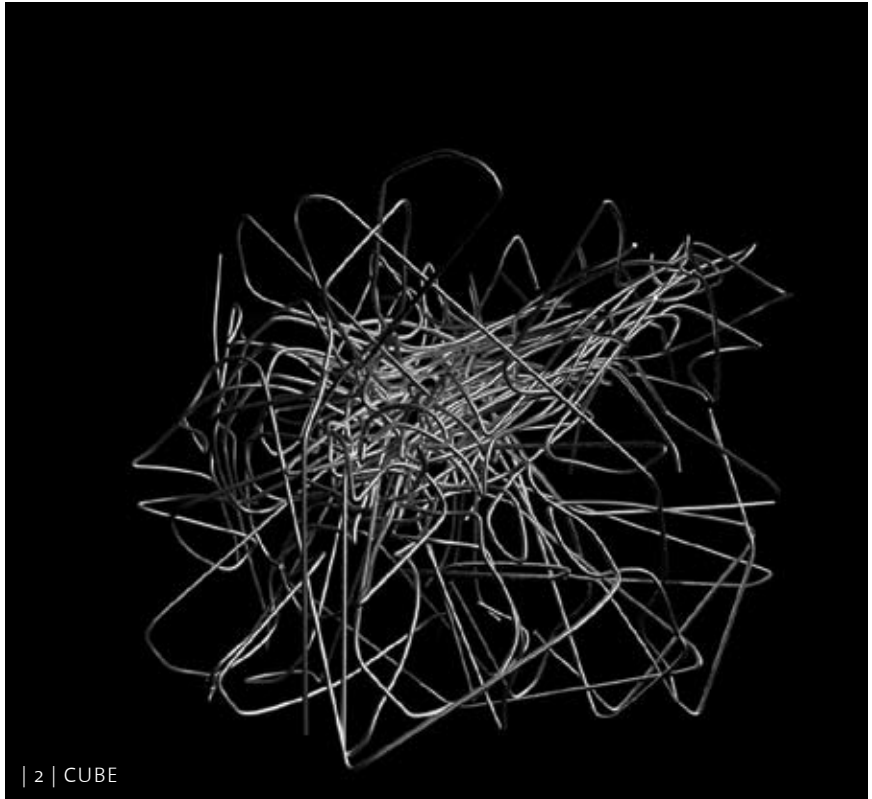
|1| Kubus

Zur Veranschaulichung der Möglichkeiten habe ich einen einfachen Kubus gewählt, da die Verfolgung der Partikel in einer komplexen Geometrie undeutlicher ist. | fig. 1|

Jede Variante hat ihren eigenen Charakter. | fig. 1, 2 ,3, 4 S. 64 - 67 |

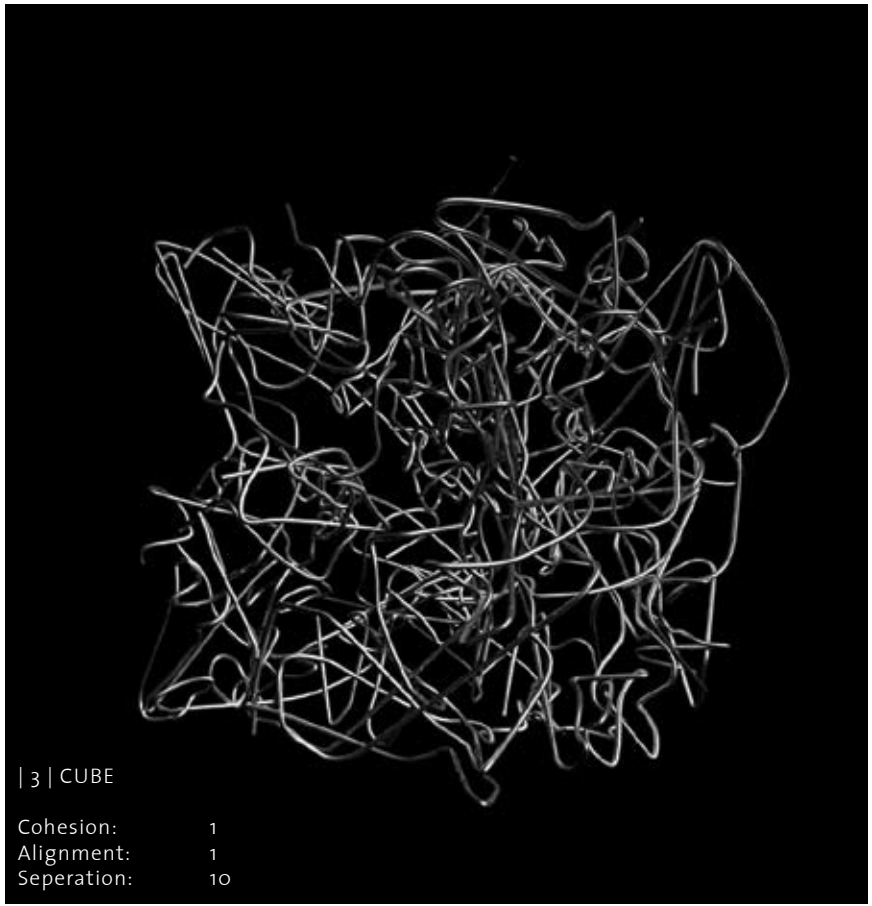
| 1 | CUBE

Cohesion: 1
Alignment: 1
Seperation: 1



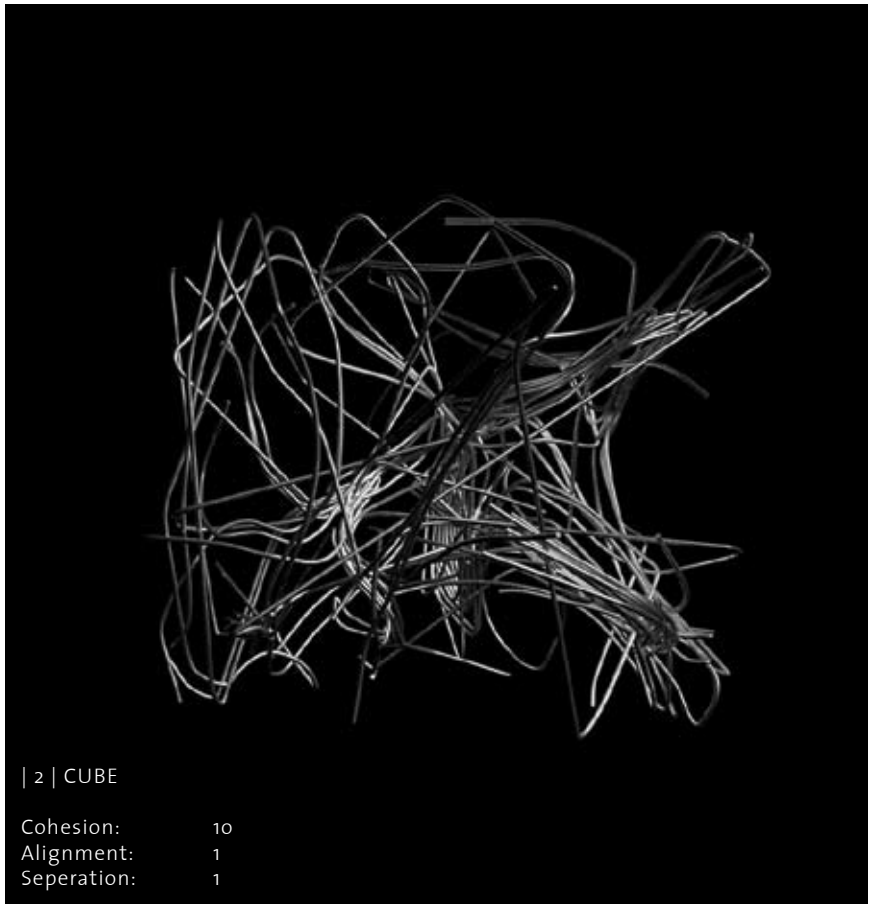
| 2 | CUBE

Cohesion: 1
Alignment: 10
Seperation: 1



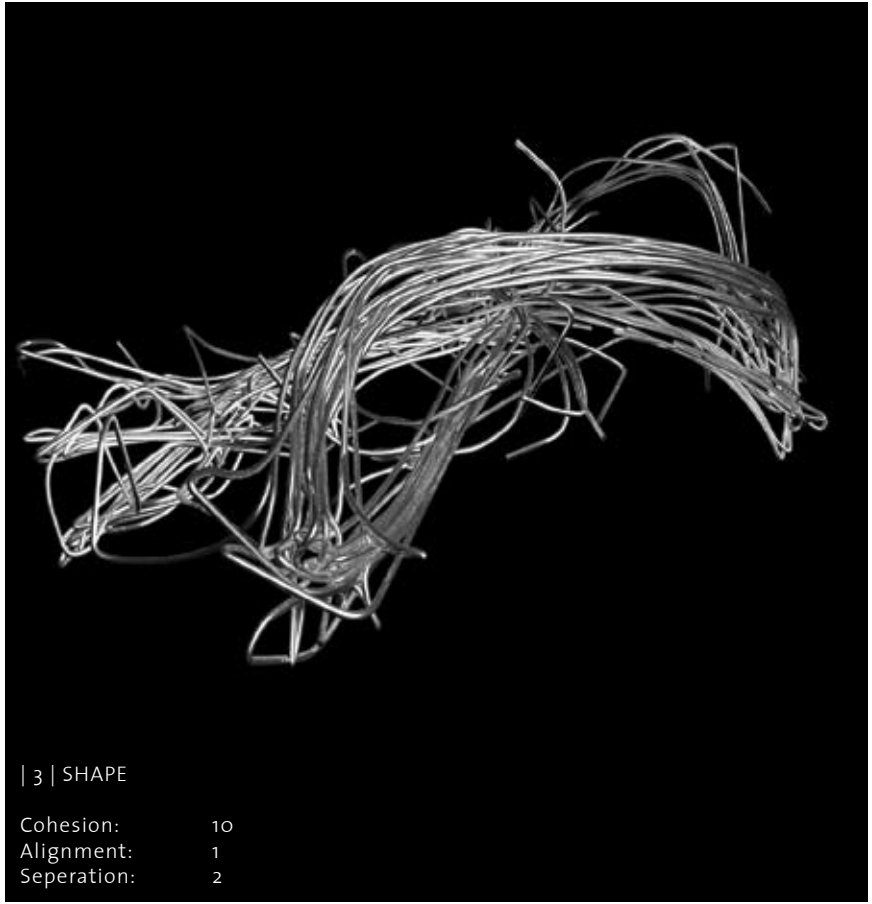
| 3 | CUBE

Cohesion: 1
Alignment: 1
Seperation: 10



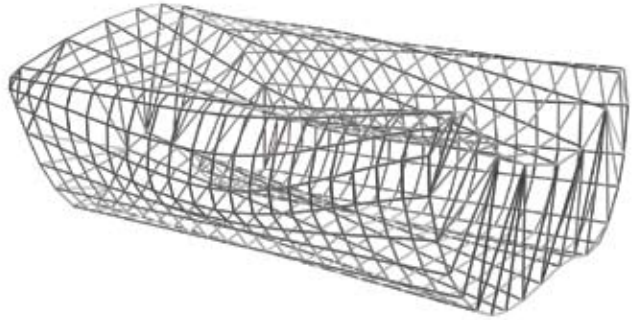
| 2 | CUBE

Cohesion: 10
Alignment: 1
Seperation: 1



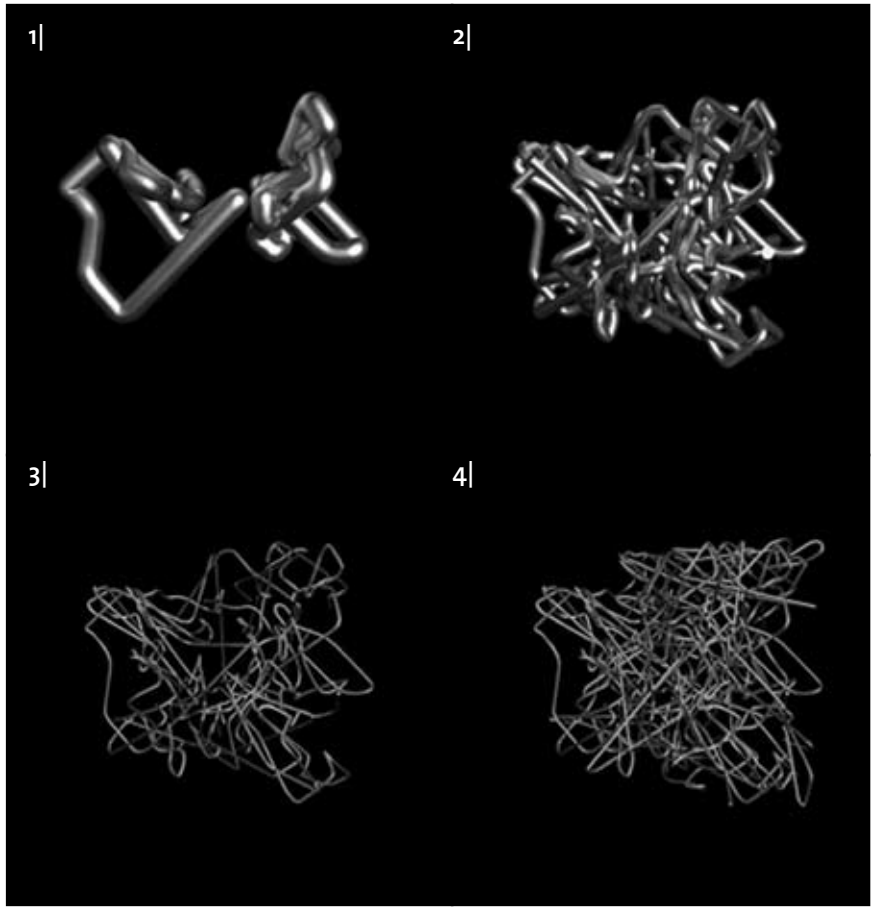
4.1.2 Volumengeometrien

Das Programm ermöglicht die Generierung gleichartiger Strukturen in Verschiedensten Volumengeometrien. Erreicht wird dies durch die Möglichkeit- das man jegliche Form die in einer CAD 3D Umgebung entwickelt wurde einlesen kann.



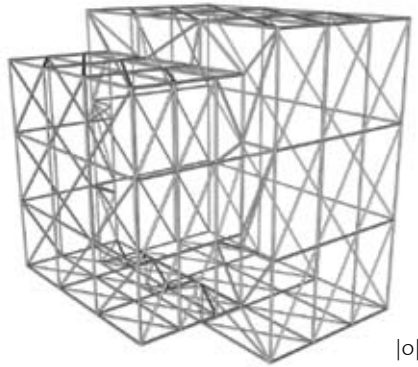
|1| gekurvtes Volumen

Es können sowohl gekurvte Formen verwendet werden wie auch orthogonale Formen. Stellt man sich unsere Umwelt in Volumen vor, so ist es praktisch möglich für jedes gewünschte Volumen eine Tragstruktur aus Partikeln zu formen. |fig. 1|



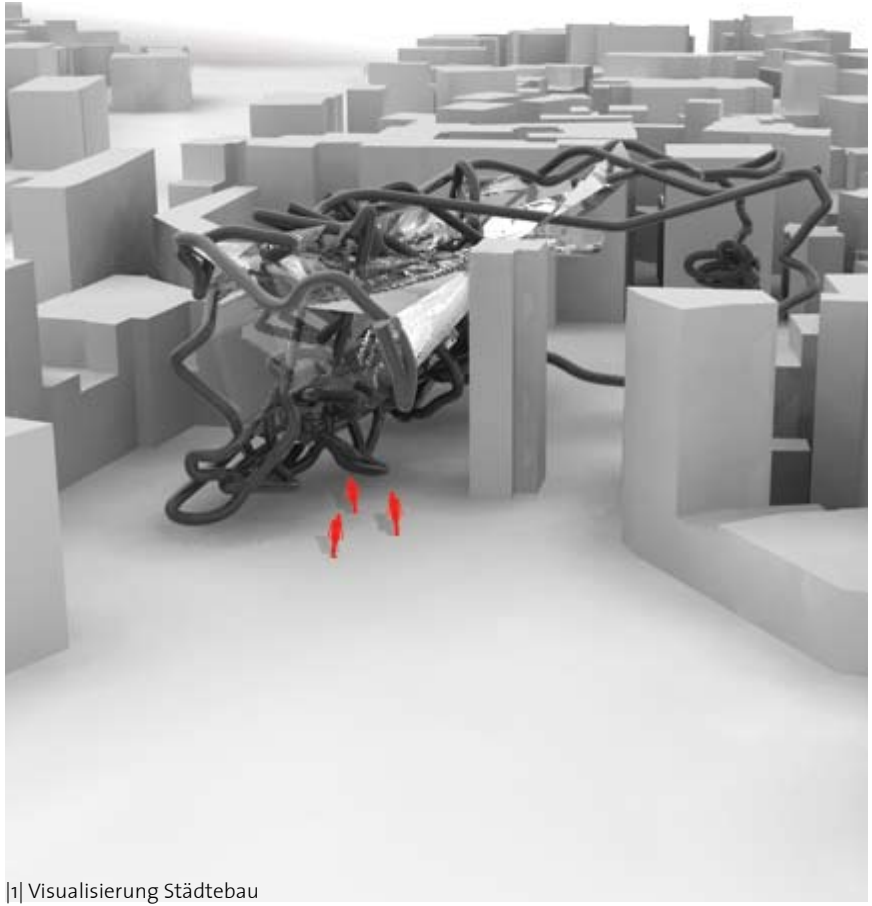
4.1.3 Rohrdurchmesser / Anzahl der Partikel

Frei wählbare Rohrdurchmesser für die Konstruktion erzeugen eine unterschiedliche Charakteristik der Struktur. Anzahl und Durchmesser der einzelnen Elemente werden durch den Gestalter festgelegt und ermöglichen so eine differenzierte Analyse der gewünschten Gestalt. |fig. 0,1,2,3,4|



|0| Volumengeometrie

1| Visual		2| Visual		3| Visual		3| Visual	
Partikel:	2	Partikel:	11	Partikel:	11	Partikel:	23
Durchmesser:	4	Durchmesser:	2	Durchmesser:	0.5	Durchmesser:	0.5
Seperation:	10	Seperation:	10	Seperation:	10	Seperation:	10
Alignment:	1	Alignment:	1	Alignment:	1	Alignment:	1
Cohesion:	1	Cohesion:	1	Cohesion:	1	Cohesion:	1



|1| Visualisierung Städtebau

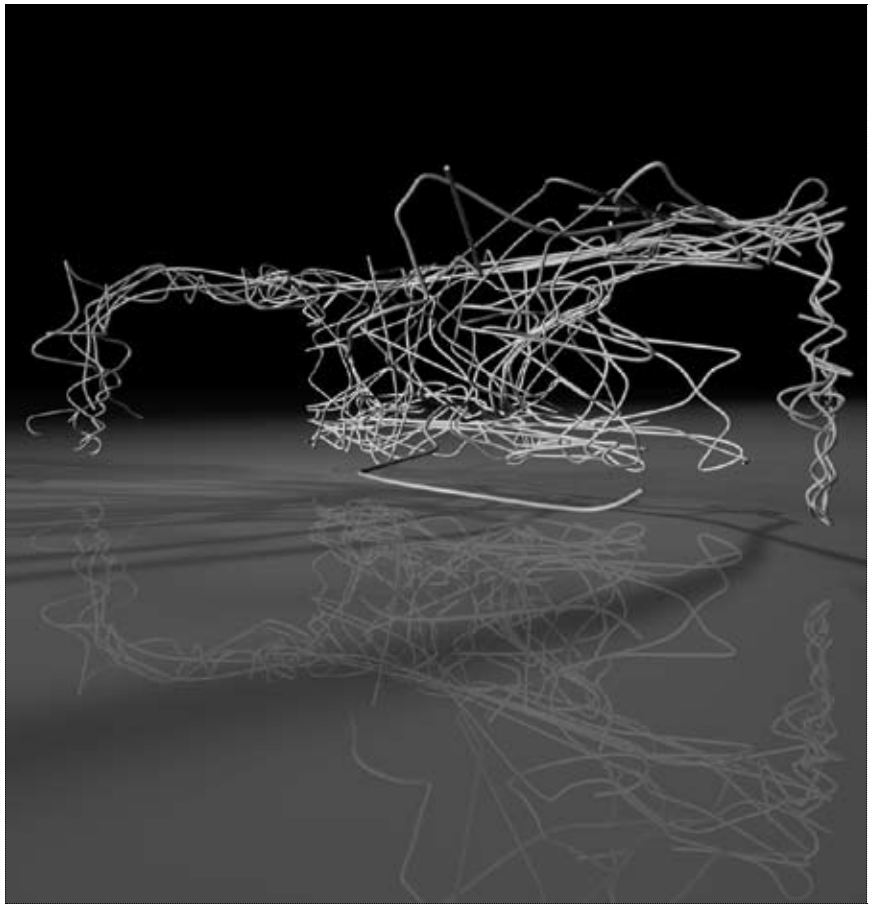
4.2 Varianten

Die Parameter des Programms und die Möglichkeit, das unterschiedliche Volumen eingelesen werden können ermöglichen es dem Entwerfer für viele architektonische Aufgaben eine Struktur zu erzeugen. Im Charakter sind diese zwar öhnlich aber dennoch der Aufgabe entsprechend gestaltet

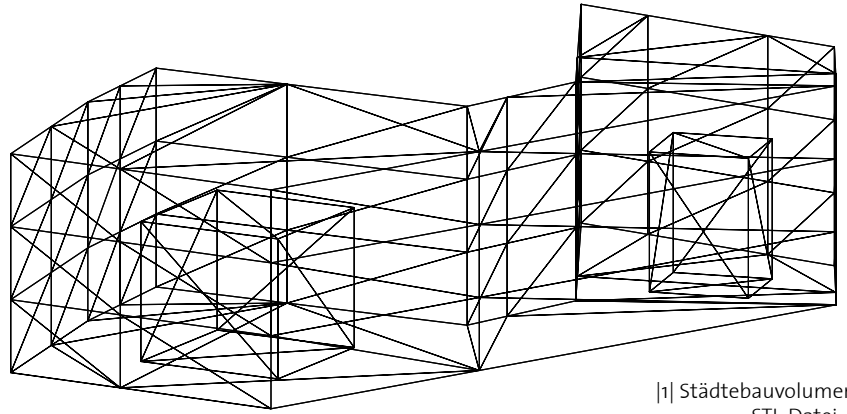
4.2.1 Städtebau

Das folgende Beispiel zeigt die Möglichkeit Strukturen im städtbaulichen Kontext zu entwickeln | fig. 1, 2|

|2| Grundstück und Stadtplan



Gebäudevolumen durch Partikelflug

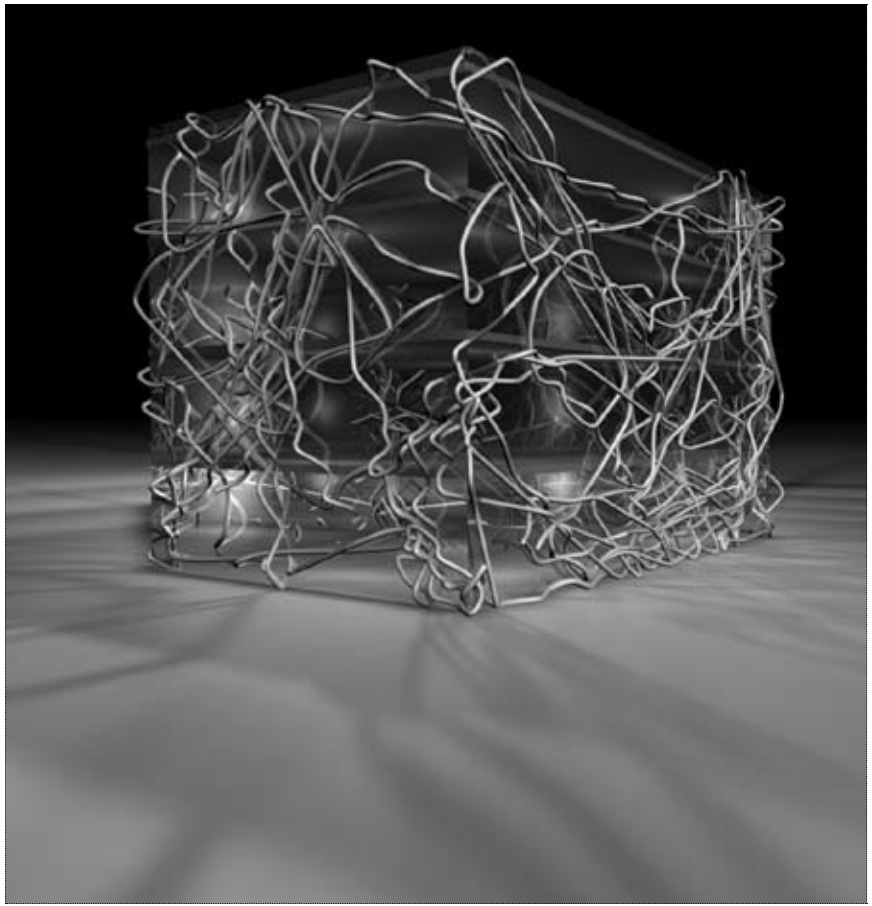


|1| Städtebauvolumen
STL-Datei

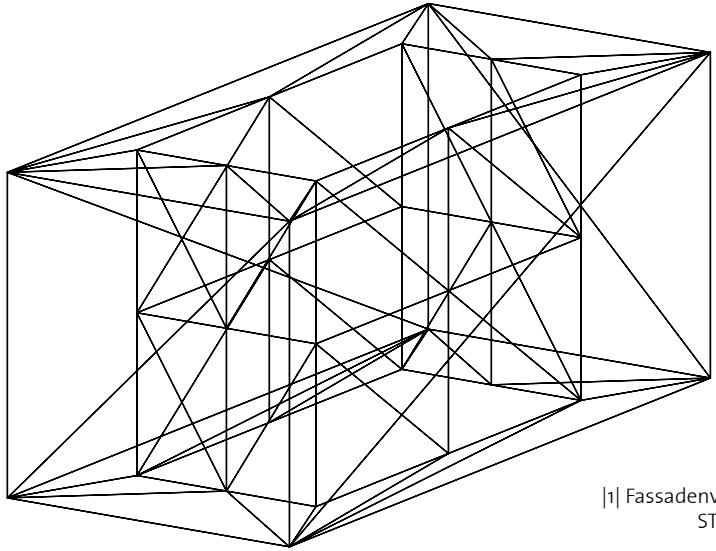
Daten des Volumenkörpers

Das Gebäude definiert sein Volumen durch die Städtebauliche Situation. Freiflächen werden Analysiert und sind die Grundlage für den Partikelflug.

Partikel:	15
Colhesion:	3
Seperation:	2
Alignment:	1



4.2.2 Architektur
Fassadenkonstruktion

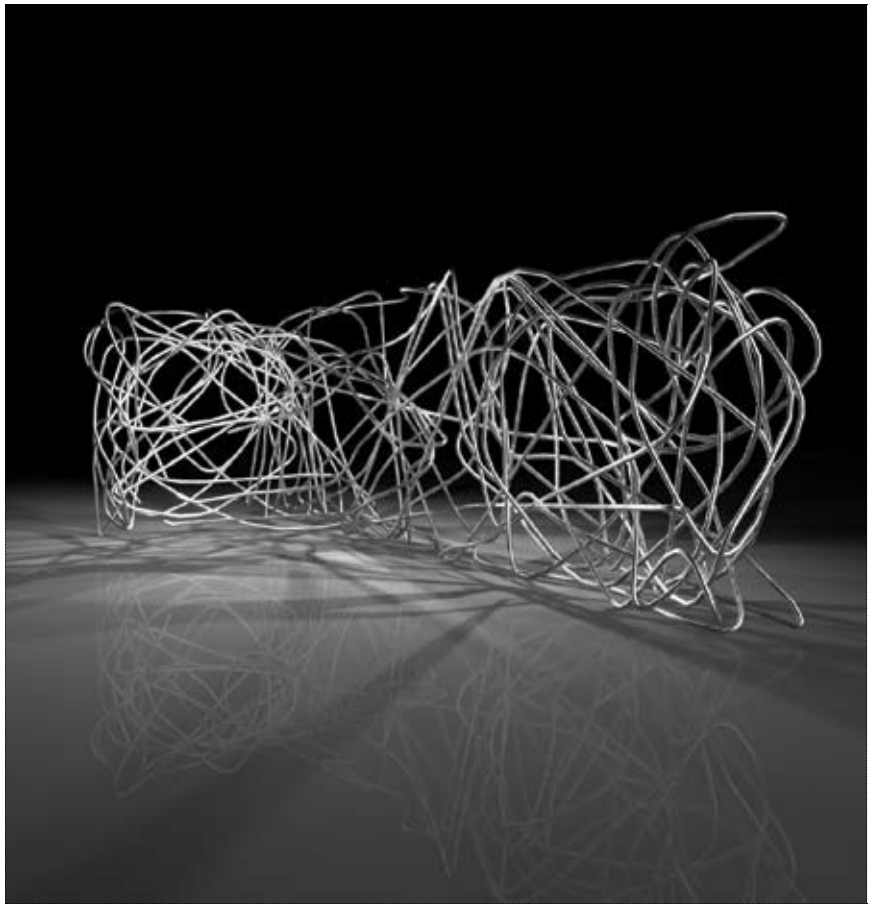


|1| Fassadenvolumen
STL-Datei

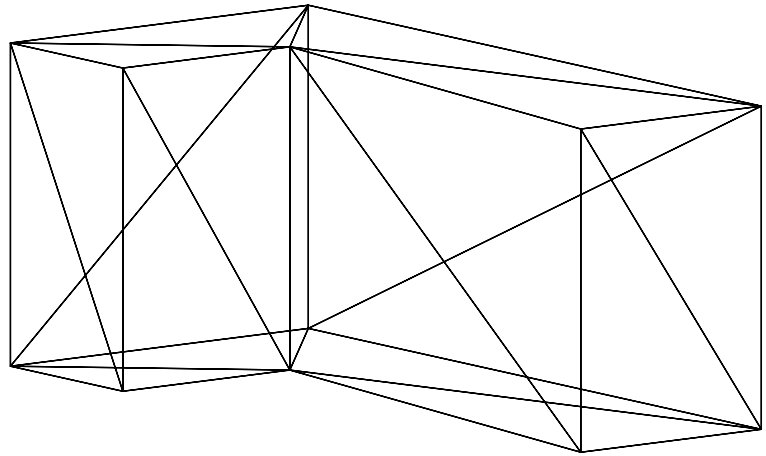
Daten des Volumenkörpers

Das Programm steuert die Partikel im Randbereich zwischen dem inneren- und dem äußeren Volumen.

Partikel:	22
Colhesion:	10
Seperation:	2
Alignment:	1



4.2.3 Skulptur
L-Strukturvolumen

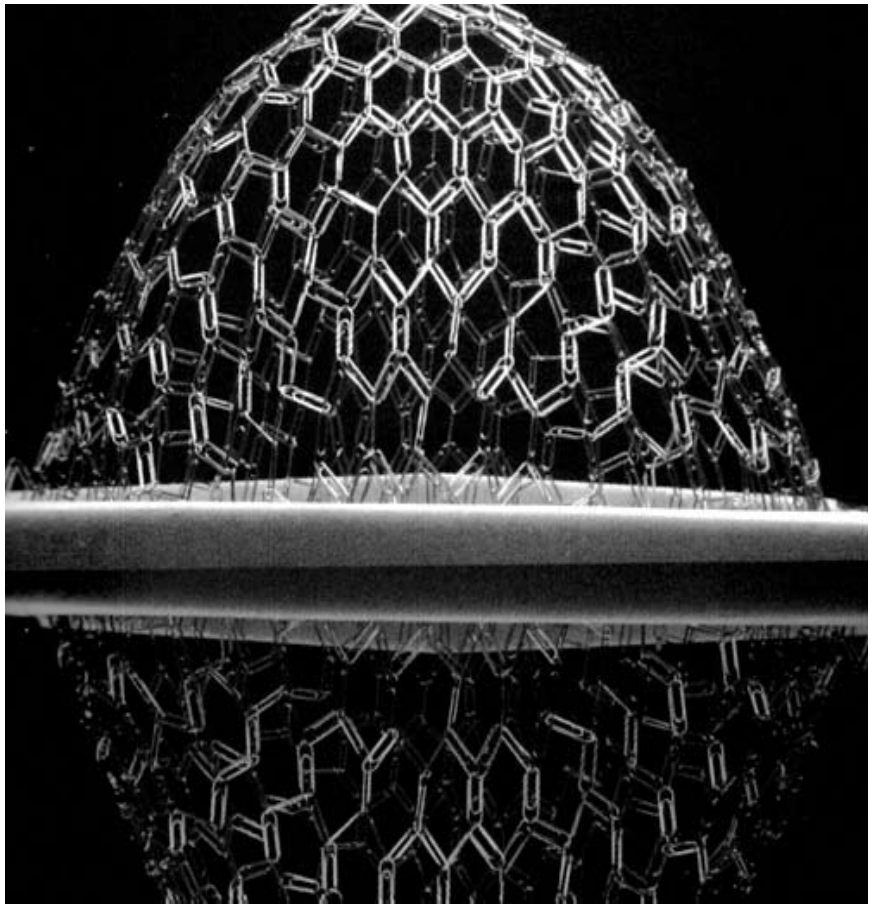


|1| Strukturvolumen
STL-Datei

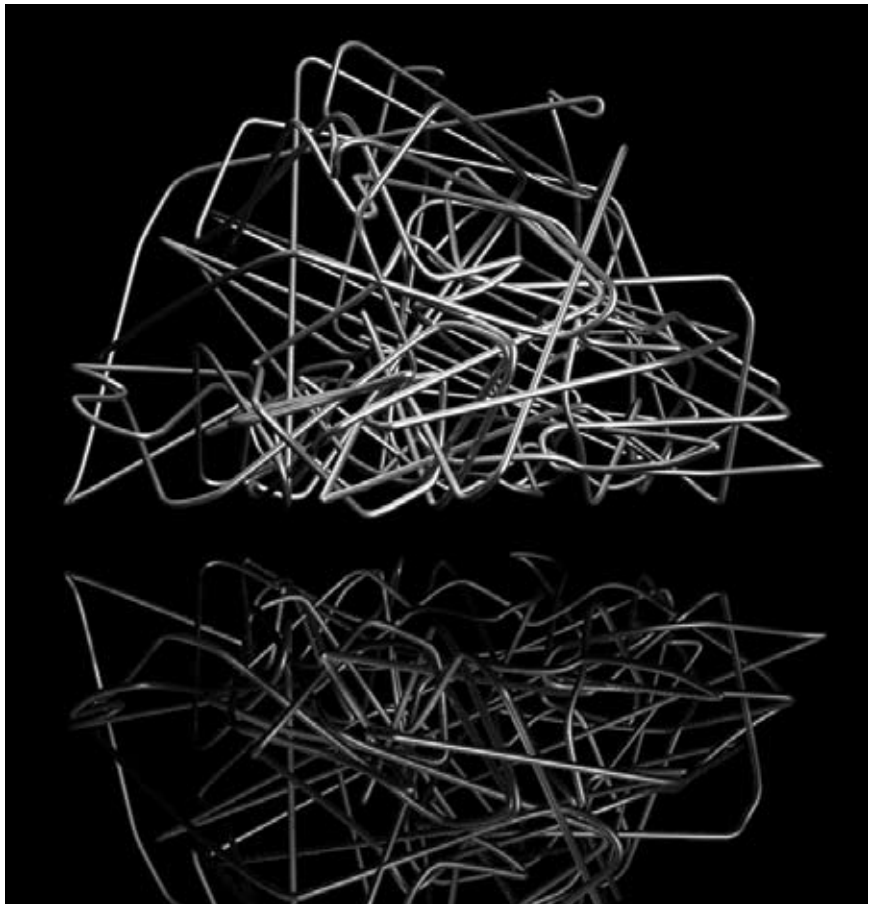
Daten des Volumenkörpers

die Partikel bewegen sich innerhalb des L-Förmigen Volumens (Struktur der MAS Gruppenarbeit)

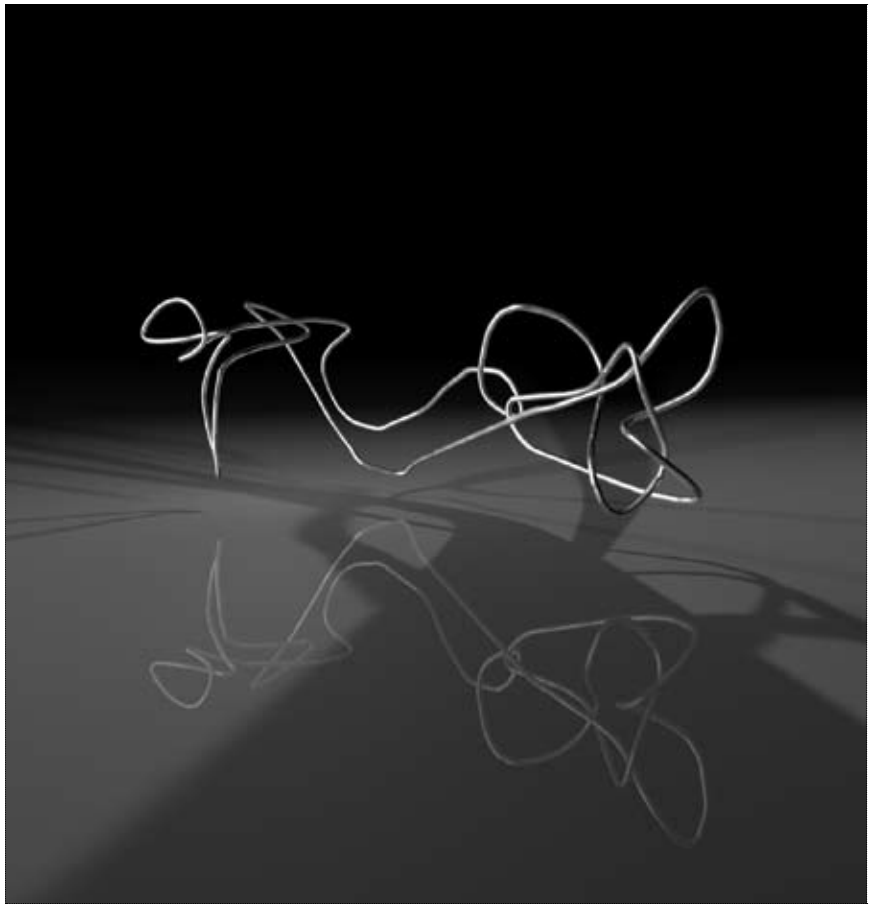
Partikel:	12
Colhesion:	10
Seperation:	2 - 10 (Echtzeitsteuerung)
Alignment:	1



80



81



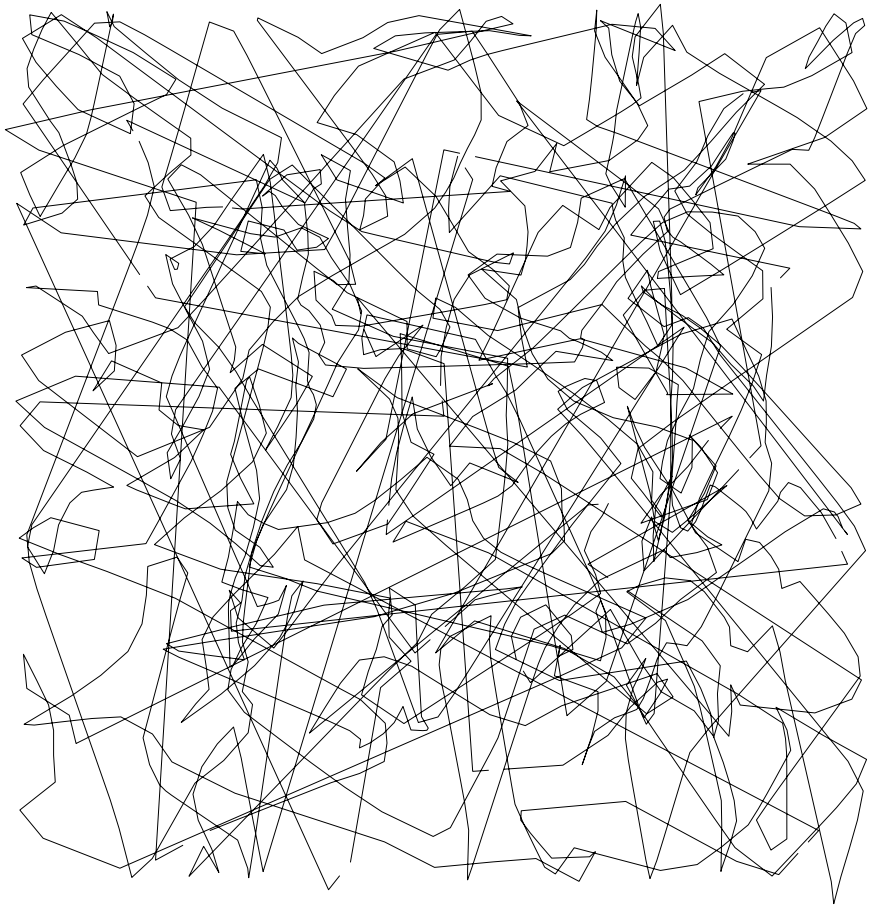
5. Fazit / Ausblick

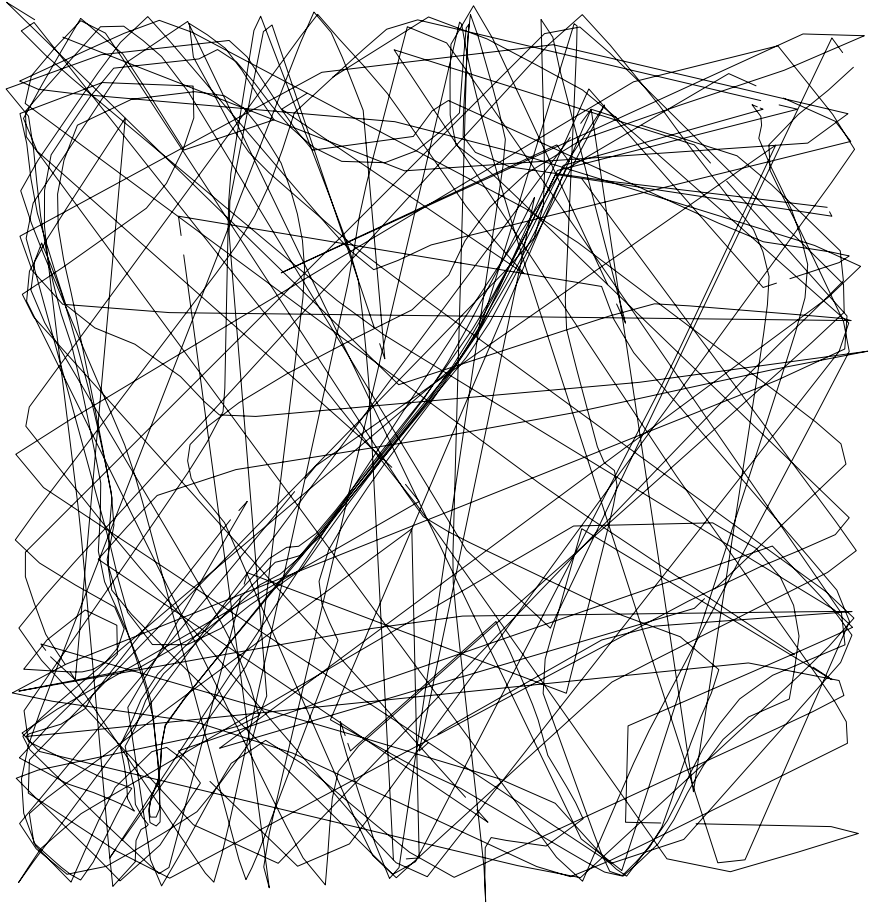
Betrachtet man die vorangegangenen Beispiele wird deutlich, dass die einfache Abbildung natürlicher Phänomene zwar ausreicht um verschiedene Strukturen für eine konkrete Bauaufgabe zu generieren doch benötigt es erheblich mehr an intelligenter Steuerung den gesamten Architektonischen Prozess abzubilden - einen „belastbaren“ Entwurf zu generieren, d. h. die Implementierung von Anforderungen wie sie die Bauaufgabe stellt. (wie z. B. die Statik, die Raumnutzung, die Haustechnische Ausstattung usw.)

Dennoch konnte gezeigt werden, das durch die Anwendung dieser Algorithmen neue Architekturen/Strukturen realisierbar werden. Die „Digitale Kette“ - entwickelt an der ETH Zürich Prof. Hovestadt- ermöglicht durch ihre Integrative Betrachtung des Bauens die Realisierung solcher Strukturen.

Das Sichtbarmachen dieser natürlichen Phänomene erlaubt es dem Architekten neue Formen zu erforschen und entsprechend der Bauaufgabe anzuwenden.

Gelingt es in Zukunft durch die Kombination verschiedener Algorithmen Programmstrukturen zu generieren, die nicht nur die natürlichen Phänomene abbilden, sondern auch die Bauaufgabe als Struktur innehaben, wird eine Architektur möglich, die in Form und Funktion nachhaltig, innovativ und vor allem komplexer ist als die, die durch einen konventionellen Entwurfsprozess entstanden ist.





Literatur

EMERGENCE: MORPHOGENETIC DESIGN STRATEGIES
AD Architectural Design
Michael Hensel, Archim Menges, Michael Weinstock
Wiley-Adademy, London 2004

COMPUTER - AIDED - ARCHITECTURAL DESIGN
William J. Mitchell
Van Nostrand Company, London 1977

ARTIFICIAL LIFE
Stefen Levy
First Vintage Books, New York 1993

ARTIFICIAL LIFE
Prof. Dr. Rolf Pfeifer
ETH Zürich, 2001

ADAPTIVE UND SELBSTORGANISIERENDE SYSTEME IN DER ARCHITEKTUR
Ulrich Königs
GAM.02 - Graz Architecture Maganzine 2005

DIVERCITY©
Ulrich Königs und IFAU - Christoph Heinemann, Christoph Schmidt
erschienen in: IFG Ulm (Hrsg.), Strategischer Raum-Urbanität im 21. Jahrhundert, Frankf. a.M., 2000

ROBOTER - HERDEN UND SIMULIERTES SCHWARMVERHALTEN
Michael Pach
www.schwarmverhalten.de

FLOCKS, HERDS AND SCHOOLS: A DISTRIBUTED BEHAVIORAL MODELL
Craig W. Reynolds
Computer Graphics 21. Juli 1987, pp. 25-34. (SIGGRAPH '87)

STEERING BEHAVIORS
FH Regensburg, Diplomarbeit
Thomas Feilkas Christian Schnellhammer
Regensburg, 2002

IMPROVED COLLISIOM DETECTION AND RESPONSE
Kasper Fauerby
http://www.peroxide.dk 2003

PROBLEMSEMINAR “ARTIFICIAL LIFE”
ANT ALGORITHMEN FÜR KOMBINATORISCHE OPTIMIERUNGSPROBLEME
Heiko Stamer
Universität Leibzig, 2001

Dipl.Ing.FH FrankThesseling

http://frank.thesseling.com

mail@frank.thesseling.com

MAS 2005 06 | CAAD | ARCH | ETHZ | CH