Simulering av flock- och stimbeteende
TNM085: Modelleringsprojekt

Iris Kotsinas, Jacob Molin
Victor Lindquist, Emma Segolsson

Sammanfattning

Denna rapport redogör för hur flockbeteenden hos djur kan simuleras med hjälp av *Boids algoritm*. Denna algoritm är baserad på tre lokala regler för varje *boid*: separation, riktning och sammanhållning. Tillsammans resulterar dessa regler i att alla *boids* rör sig som en synkroniserad flock. I detta projekt i kursen TNM085 - Modelleringsprojekt har projektgruppen simulerat systemet tvådimensionellt i programvaran *MATLAB* och tredimensionellt som en webbapplikation i *JavaScript*.

Innehållsförteckning

1 Inledning **1**
- 1.1 Bakgrund . . . 1
- 1.2 Syfte och mål . . . 1
- 1.3 Avgränsningar . . . 1

2 Metod **2**
- 2.1 System . . . 2
- 2.2 Modell . . . 2
  - 2.2.1 Boids algoritm . . . 3
  - 2.2.2 Anpassad boids algoritm . . . 6
- 2.3 Numerisk metod . . . 6
- 2.4 Simulering och visualisering . . . 7
  - 2.4.1 Implementation i MATLAB . . . 7
  - 2.4.2 Implementation i JavaScript . . . 9

3 Resultat **10**
- 3.1 MATLAB . . . 10
- 3.2 JavaScript . . . 10

4 Diskussion **11**

5 Slutsatser **12**

Litteraturförteckning **13**

A Plottar från Matlab **14**

B Bilder från visualiseringen i JavaScript **17**

Figurer

2.1 Representation av regeln separation 3
2.2 Representation av regeln separation 4
2.3 Representation av regeln riktning 4
2.4 Representation av regeln sammanhållning 5

A.1 Positioner för tio boids vid iteration 1 14
A.2 Positioner för tio boids vid iteration 40 15
A.3 Positioner för tio boids vid iteration 80 15
A.4 Positioner för tio boids vid iteration 120 16
A.5 Positioner för tio boids vid iteration 160 16

B.1 Överblick på webbläsarfönstret 17
B.2 En grupp av boids i visualiseringen 18
B.3 Reglage för anpassning av simuleringens variabler 18

Kapitel 1

Inledning

Flockbeteende är ett välkänt fenomen hos djur, såsom fåglar och fiskar, och beror på en uppsättning av lokala regler som varje individ följer. Detta modelleringsprojekt undersöker vilka de reglerna är och hur dessa kan implementeras och animeras med hjälp av matematiska modeller.

1.1 Bakgrund

För att få en lättare förståelse kring hur djur beter sig i flock kan det underlätta att simulera beteendet. År 1986 skapade Craig Reynolds en datormodell av djurs koordinerade rörelser i grupp, som fiskstim och fågelflockar [1]. Han kallade datormodellen för *Boids algoritm*. Ordet *boids* är benämningen på en individ i simuleringen av flocken eller stimmet. Boids algoritm använder sig av en uppsättning av tre enkla regler för att bestämma hur de ska röra sig: separation, riktning och sammanhållning. De tre reglerna som är formulerade av Reynolds är fortfarande grunden för modern flockningssimulering och tillämpas i detta projekt.

1.2 Syfte och mål

Syftet med det här projektet är att simulera och visualisera ett rörelsebeteende hos djur med hjälp av matematiska modeller. Rörelsebeteendet beror på en uppsättning av regler och projektets mål är att implementera dessa. Simuleringen ska först implementeras i programvaran *MATLAB* och sedan utvecklas till en webbapplikation med hjälp av JavaScript-biblioteket *Three.js*, som använder JavaScript API:et *WebGL*.

1.3 Avgränsningar

Projektet har avgränsats med avseende på systemets fysikaliska aspekter. Systemet simuleras inte med den påverkan faktorer såsom luftmotstånd och gravitation har på systemet. Ännu en avgränsning är att alla boids har konstant hastighet och därför tas ingen hänsyn till acceleration.

Kapitel 2

Metod

Nedan beskrivs systemet, modellen av systemet samt den numeriska metoden.

2.1 System

Systemet består av en grupp individer som alla rör sig individuellt, men som tillsammans bildar en flock eller ett stim genom att röra sig synkroniserat och beroende på varandra. Varje individ har de fysikaliska egenskaperna position $p(t)$ och hastighet $v(t)$ vid en viss tidpunkt t. Dessa individuella egenskaper påverkas av de andra individerna i flocken. Individerna drar sig mot varandra och rör sig i samma generella riktning, samtidigt som de undviker att kollidera med varandra. De bildar på så sätt en flock eller ett stim.

2.2 Modell

Modellen beskriver systemet genom att beräkna varje individs position $p(t)$ i en flock för att sedan plotta individernas positioner över tid. Individerna ska då förhålla sig till varandra och påverka varandra. Tillsammans bör de då se ut som en flock eller ett stim som rör sig synkroniserat.

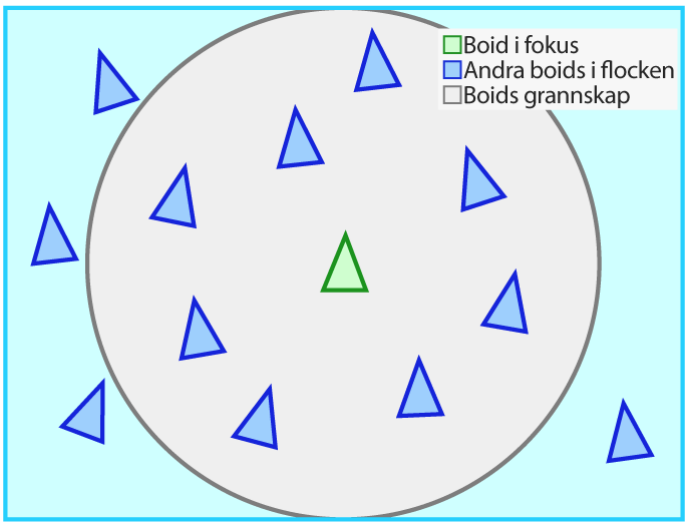
Det går att beskriva en individs position $p(t)$ i systemet vid en viss tidpunkt t. Det kan göras matematiskt med hjälp av (2.1), där $v(t)$ kan beräknas och ersättas med hjälp av (2.2), som beräknas med hjälp av Boids algoritm [2]. Hastigheterna $v_{sep}(t)$, $v_{ali}(t)$ och $v_{coh}(t)$ representerar hastigheter som beräknas baserat på de tre reglerna *separation*, *riktning* respektive *sammanhållning* som beskrivs i Boids algoritm [2]. Boids algoritm beskrivs närmare i sektion 2.2.1.

$$p(t) = \int v(t)dt \tag{2.1}$$

$$v_{tot}(t) = v_{sep}(t) + v_{ali}(t) + v_{coh}(t) \tag{2.2}$$

2.2.1 Boids algoritm

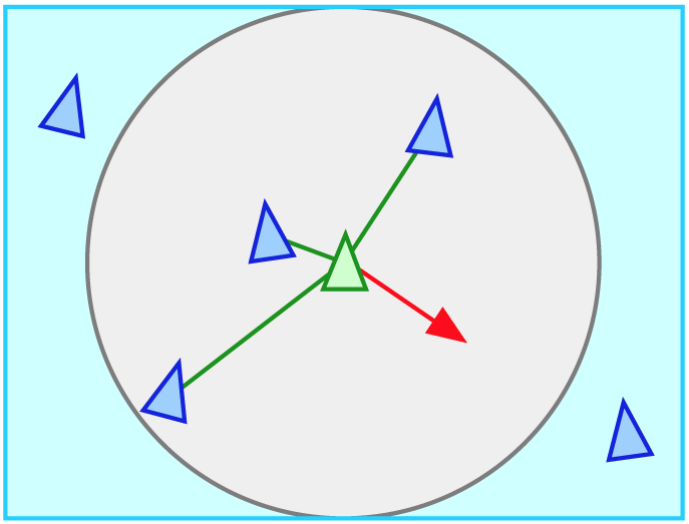
Boids algoritm består i grunden av tre förhållningsregler som varje individ, eller boid, ska förhålla sig till. Nedan förklaras de tre reglerna. Under varje regel finns en figur som visar hur en boid ska agera för att följa regeln. Den gröna triangeln är boiden i fokus, medan de blå trianglarna är andra boids. De blå boids som ligger innanför den grå cirkeln kommer påverka den gröna boiden och är den gröna boidens grannskap. De blå boids som ligger utanför den grå cirkeln kommer inte påverka den gröna boidens nästkommande position. En beskrivande illustration kan ses i Figur 2.1



Figur 2.1: Representation av regeln separation

Separation

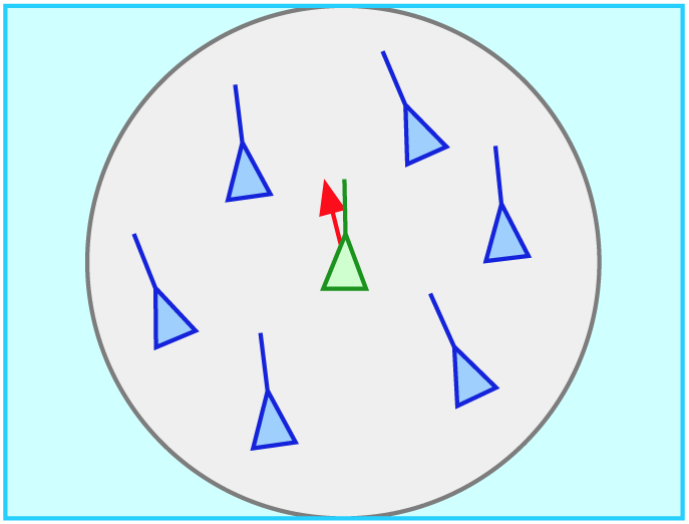
Den första regeln är separation. Det är också den regel som väger tyngst. Denna regel ser till att boids i en flock aldrig kolliderar. Varje boid ska vid ett visst avstånd från dess närliggande boids styra undan från dem för att undvika kollision. I Figur 2.2 visas hur en boid ska agera för att följa regeln. Den gröna boiden ska flytta sig från de blå boids som tillhör den gröna boidens grannskap, alltså de blå boids som befinner sig inom den grå cirkeln. Den röda pilen visar den resulterande riktiningen som den gröna boiden ska röra sig i för att följa regeln och undvika kollision.



Figur 2.2: Representation av regeln separation

Riktning

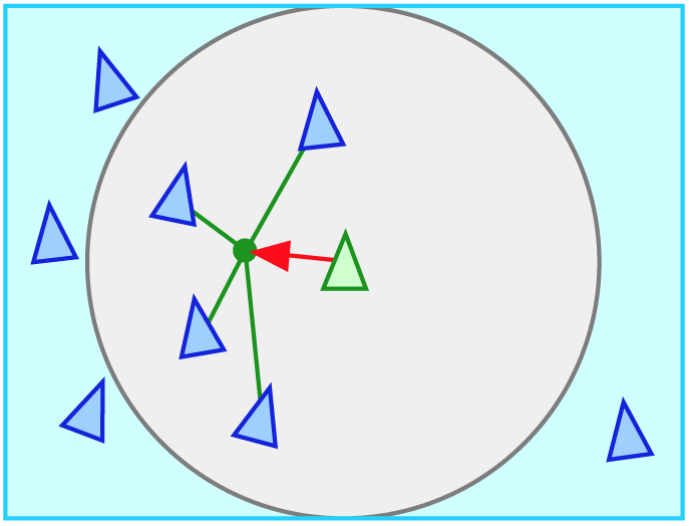
Riktning är en regel som går ut på att varje boid ska styra mot medelriktiningen av boidsen i dess grannskap. Denna regel ser till att flocken rör sig i samma riktning. I Figur 2.3 visas hur en boid ska agera för att följa regeln. Den gröna boiden behöver lägga om sin kurs för att färdas i samma riktning som de blå boidsen i grannskapet. Den röda pilen visar medelriktningen för grannskapet. Den gröna boiden ska lägga om sin kurs enligt den röda pilen.



Figur 2.3: Representation av regeln riktning

Sammanhållning

Sammanhållning betyder att varje boid ska styra mot medelpositionen av dess närliggande boids. I Figur 2.4 visas hur en boid ska agera för att följa regeln. Den gröna boiden ska följa den röda pilen och flytta sig mot den gröna punkten, vilket är medelpositionen för boidsen i den gröna boidens grannskap. Regeln ser till att boidsen håller ihop som en grupp.



Figur 2.4: Representation av regeln sammanhållning

2.2.2 Anpassad boids algoritm

Utöver de tre reglerna i Boids algoritm anpassades modellen med en till regel om att undvika kollision med väggar. Den regeln ser till att flocken kan verka inom fyra väggar och på så sätt kan flocken studeras över en längre tid. Utan väggar kan flocken röra sig utanför den yta som plottas. Baserat på de tre reglerna i Boids algoritm samt regeln om att inte kollidera med väggar skulle boidsen nu kunna röra sig synkroniserat likt ett fiskstim eller en flock av djur.

2.3 Numerisk metod

För att kunna implementera modellen i programkod behövde (2.1) approximeras till en differensekvation. Genom att först skriva om (2.1) till (2.3) och sedan använda derivatans definition [4] i (2.4) för $\dot{p}(t)$ kunde (2.5) härledas genom att lägga ihop (2.3) och (2.4).

$$v(t) = \dot{p}(t) \tag{2.3}$$

$$\dot{p}(t) = \lim_{h \to 0} \frac{p(t+h) - p(t)}{h} \tag{2.4}$$

$$v(t) = \lim_{h \to 0} \frac{p(t+h) - p(t)}{h} \tag{2.5}$$

Ekvation (2.5) kunde sedan förflyttas från kontinuerlig till diskret tidsdomän genom att approximera $\dot{p}(t)$ med hjälp av Eulers metod [3] i (2.6) och låta h vara ett litet tal. Resultatet är (2.7). I (2.7) motsvaras $f(t[n], x[n])$ av $v[n]$ och $x[n]$ av $p[n]$, där $n = 1, 2, 3, \ldots$.

$$\begin{aligned} x[n+1] &= x[n] + hf(t[n], x[n]) \\ \Leftrightarrow \quad f(t[n], x[n]) &= \frac{x[n+1] - x[n]}{h} \end{aligned} \tag{2.6}$$

$$\begin{aligned} v[n] &= \frac{p[n+1] - p[n]}{h} \\ \Rightarrow \quad p[n+1] &= p[n] + hv[n] \end{aligned} \tag{2.7}$$

Genom att sätta in (2.8), den tidsdiskreta motsvarigheten till (2.2), i (2.7) kunde (2.9) tas fram. Ekvation (2.9) användes för att beräkna en individs nästkommande position.

$$v_{tot}[n] = v_{sep}[n] + v_{ali}[n] + v_{coh}[n] \tag{2.8}$$

$$p[n+1] = p[n] + hv_{tot}[n] \tag{2.9}$$

2.4 Simulering och visualisering

Systemet simulerades och visualiserades i både MATLAB och Javascript. Simuleringen implementerades först över ett tvådimensionellt plan i MATLAB för att kunna studera de numeriska värdena samt öka förståelsen för systemet. Därefter implementerades simuleringen i ett tredimensionellt rum i JavaScript. Sektionerna nedan beskriver hur simuleringen implementerades.

2.4.1 Implementation i MATLAB

Initialt simulerades systemet i programvaran MATLAB. De tre tidigare nämnda reglerna i sektion 2.2.1 och den numeriska metoden i sektion 2.3 implementerades och utvecklades i kod. Koden delades upp i filerna *plot_boids.m*, *Flock.m* samt *Boid.m*. I filen *plot_boids.m* skapades en flock med boids som visualiserades via filen *Flock.m*. För att alla boids skulle rotera korrekt beräknades varje boids vinkel i filen *plot_boids.m*. De tre algoritmerna för separation, riktning och sammanhållning implementerades som varsin funktion i MATLAB-filen *Boid.m*.

Separationsalgoritmen tar in alla boids och en detektionsradie (se Algoritm 1). Alla boids testas för om dess position är inom detektionsradien för den specifika boiden. Om så är fallet räknas den som en granne och tas in i beräkningen för den nya riktningen. Denna algoritm ser till att varje boid har ett visst avstånd till dess grannar för att undvika kollision.

```
Data: All boids
Result: velocity
steer = (0,0);
while boid within detection radius of boid b do
    foreach boid do
        steer = steer + close boids position;
    end
end
if steer is changed then
    steer = average of steer;
    velocity = boid position - steer;
    velocity = norm velocity;
else
    velocity not changed;
end
```

Algoritm 1: Separationsalgoritmen i pseudokod

Precis som separationsalgoritmen tar riktningsalgoritmen in alla boids och en detektionsradie (se Algoritm 2). Samtliga boids testas för om dess position är inom detektionsradien för den specifika boiden. Om så är fallet räknas den som en granne och tas in i beräkningen för den nya

riktningen. Denna riktning beror på medelriktningen av samtliga boids i dess grannskap.

```
Data: All boids
Result: velocity
steer = (0,0);
while boid within detection radius of boid b do
    foreach boid do
        steer = steer + close boids velocity;
    end
end
if steer is changed then
    steer = average of steer;
    velocity = norm steer;
else
    velocity not changed;
end
```

Algoritm 2: Riktningsalgoritmen i pseudokod

Sammanhållningsalgoritmen tar in boids inom en given detektionsradie (se Algoritm 3). Samtliga boids testas om de är inom en given sammanhållningsradie innan de skickas in i funktionen. Detta sker eftersom denna radie är störst utav samtliga radier och för alla algoritmer klassas dessa boids som grannar. Funktionen beräknar den nya riktningen mot medelpositionen av dess närliggande boids.

```
Data: All boids within detection radius of the boid b
Result: velocity
steer = (0,0);
while boid within detection radius do
    foreach boid do
        steer = steer + close boids position;
    end
end
if steer is changed then
    steer = average of all boids within detection radius of b;
    velocity = b position;
    velocity = norm velocity;
else
    velocity not changed;
end
```

Algoritm 3: Sammanhållningsalgoritmen i pseudokod

Tillsammans skapar de tre algoritmerna en flock med boids som beror på reglerna för separation, riktning och sammanhållning.

2.4.2 Implementation i JavaScript

En implementation av algoritmerna gjordes i skriptspråket JavaScript för att kunna simulera flockbeteende med en visualisering direkt i en webbläsare. Implementationen gjordes, likt MATLAB-koden, med ett objektorienterat tillvägagångssätt, där varje boid utgör ett eget objekt och flocken innehåller instanser av dessa objekt. Att porta MATLAB-implementationen till JavaScript var inte tidskrävande, eftersom båda programspråken är svagt typade.

För visualiseringen av systemet användes JavaScript-biblioteket *Three.js* [5]. Three.js utnyttjar JavaScript-API:et *WebGL*. Ett API, eller applikationsprogrammeringsgränssnitt, är ett gränsnitt mellan applikationer och en specifik programvara. WebGL är ett API som kan utföra 3D-rendering direkt i webbläsaren, genom att möjliggöra kommunikation mellan JavaScript och användarens grafikkort. Three.js är ett bibliotek som innehåller många användbara datastrukturer och funktioner som gör det enklare för utvecklare att implementera WebGL.

I det här projektet har flertalet datastrukturer och funktioner från Three.js använts. Dessa inkluderar bland annat *Vector3*, vilket är en datastruktur som representerar en vektor eller en punkt i R^3 [6]. *Vector3* har funktioner som möjliggör till exempel vektoraddition och vektormultiplikation, vilket är användbart inom 3D-visualisering, speciellt när rörelser simuleras.

Implementationen i JavaScript innehöll en algoritm som gjorde att alla boids enbart rörde sig innanför en låda. Det gjordes genom att ta fram vektorer från väggarna av lådan till en boid och sedan addera inversen av dessa för att få en ny riktning. Den nya riktningen styr boiden bort från alla väggar, men med en viktning, så den styr bort mest från den närmsta väggen.

I Programkod 2.1 visas hur sammanhållningsalgoritmen såg ut i JavaScript-implementationen.

```
// function cohesion( nearby_boids : Array(Boids) ) : THREE.Vector3
cohesion(nearby_boids) {
  let cohsionVelocity = new THREE.Vector3(); // null vector
  let steer = new THREE.Vector3(); // null vector

  if (nearby_boids.length > 0) {
    nearby_boids.forEach(boid => {
      steer.add(boid.position);
    });

    steer.divideScalar(nearby_boids.length); // average

    cohsionVelocity.subVectors(steer, this.position);
    cohsionVelocity.normalize();
  }

  return cohsionVelocity;
}
```

Programkod 2.1: Sammanhållningsalgoritmen i JavaScript

Kapitel 3

Resultat

I detta kapitel beskrivs resultaten från implementeringen i MATLAB och JavaScript.

3.1 MATLAB

Implementationen i MATLAB resulterade i ett tvådimensionellt plan där boids i formen av trianglar rör sig efter de tre reglerna nämnda i sektion 2.2.1. Initialt placeras alla boids ut slumpmässigt på planet med en slumpmässig riktningsvektor. De tre reglerna för separation, sammanhållning och riktning resulterar i att samtliga boids efter en oförutsägbar tid rör sig i ett flockliknande mönster. I bilaga A visas fem plottar över hur tio boids rör sig över det tvådimensionella planet under 160 iterationer. Boidsen representeras av röda trianglar och deras positioner visas med x-koordinater och y-koordinater i det övre, högra hörnet i varje plot.

3.2 JavaScript

Implementationen i JavaScript resulterade i en tredimensionell simulering av flockbeteende i en webbapplikation. Varje boid är en enkel 3D-modell av en fågel och alla boids rör sig inom en låda. På webbapplikationen kan användaren ändra på simuleringens variabler via ett reglage. I bilaga B visas bilder från visualiseringen i JavaScript och på reglagen för anpassning av simuleringens variabler. JavaScript-implementationen hittas på följande länk: `http://www.student.itn.liu.se/~vicli268/tnm085-boids/Threejs/`.

Kapitel 4

Diskussion

Simulering av flockbeteenden hos djur med Boids algoritm är baserad på tre regler. Dessa regler har i detta projekts implementerats i både MATLAB och JavaScript. Systemet simulerades först tvådimensionellt i programvaran MATLAB. Därefter utvecklades systemet till en tredimensionell webbapplikation i JavaScript, där en till regel adderades för att undvika kollision med väggarna.

Både MATLAB och JavaScript är svagt typade programspråk, vilket gjorde det relativt enkelt att porta programkoden. Att ett programspråk är svagt typat innebär att kompilatorn bestämmer alla variablers datatyper, om inte utvecklaren själv specificerar variabelns datatyp. Nackdelen med ett svagt typat programspråk är att felsökning blir svårare, eftersom datatypsomvandling ibland kan vara problematiskt. Datatypsomvandling är ett problem, eftersom variabler byter datatyp vid vissa operationer och felmeddelandena ges inte förrän nästa operation på variabeln utförs.

En avgränsning som gjordes i projektet var att låta alla boids ha en konstant hastighet, det vill säga en acceleration som alltid är noll. En ny ritningsvektor beräknas fram genom att addera de närliggande boids riktningsvektorer. Innan den nya riktningsvektorn ersätter den gamla normaliseras den till längden ett, vilket resulterar i att vi inte tar hänsyn till en acceleration. Om accelerationen hade tagits hänsyn till skulle varje boid ha kunnat matcha sin hastighet med närliggande boids och skapat en variation i rörelse.

Det finns ett antal funktioner som kunde ha implementerats för att vidareutveckla systemet. Till exempel hade diverse hinder kunnat skapats tillsammans med en regel för att undvika dessa. Ytterligare hade ett rovdjur kunnat implementerats som flocken skulle ha flytt ifrån, exempelvis användarens muspekare.

Kapitel 5

Slutsatser

Projektets huvudmål var att simulera ett flockliknande beteende. Projektet utgick ifrån Boids algoritm, en algoritm baserad på tre grundregler kring separation, riktning och sammanhållning. Simuleringen implementerades tvådimensionellt i MATLAB och tredimensionellt som en webbapplikation i JavaScript. I JavaScript adderades ännu en regel för att undvika kollision med väggarna. I webbapplikationen skapades ett reglage för anpassning av simuleringens variabler som användaren kan ändra på. Projektgruppen har med framgång utvecklat en fungerande webbapplikation som simulerar flockbeteenden hos djur med hjälp av numeriska metoder.

Litteraturförteckning

[1] Reynolds, C. Boids, Background and Update.
[2020-03-01] Hämtad från: http://www.red3d.com/cwr/boids/

[2] Reynolds, Craig W. Flocks, Herds, and Schools: A Distributed Behavioral Model. Computer Graphics. 1987;21(4):25-34.
[2020-03-01] Hämtad från: https://dl.acm.org/doi/pdf/10.1145/37402.37406?download=true

[3] Ljung, L. & Glad, T. Modellbygge och simulering, upplaga 2:5.
Lund: Studentlitteratur. 2004

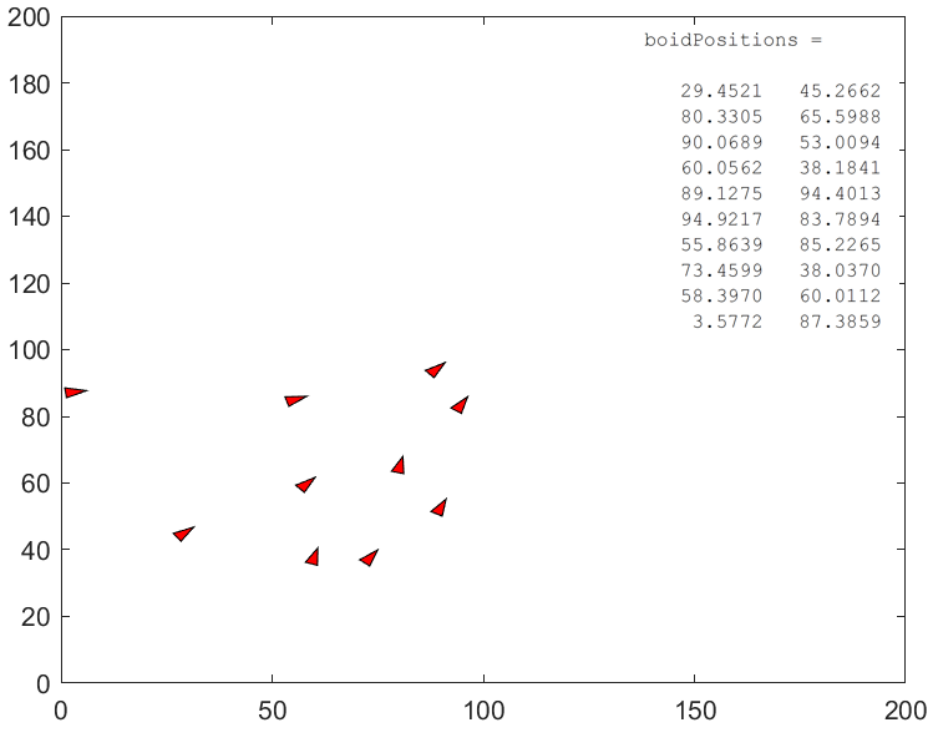
[4] Forsling, G. & Neymark, M. Matematisk analys en variabel, andra upplagan.
Stockholm: Liber. 2011

[5] Three.js Fundametals
[2020-03-01], Hämtad från: https://threejsfundamentals.org/threejs/lessons/threejs-fundamentals.html

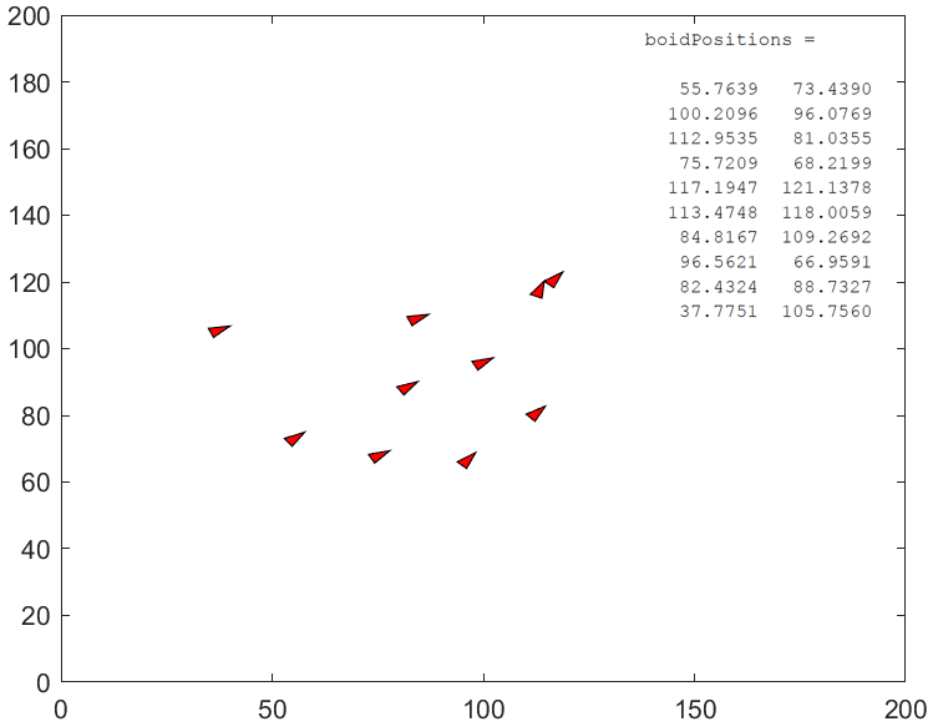
[6] Vector3 - three.js docs
[2020-03-08] Hämtad från: https://threejs.org/docs/#api/en/math/Vector3

Bilaga A

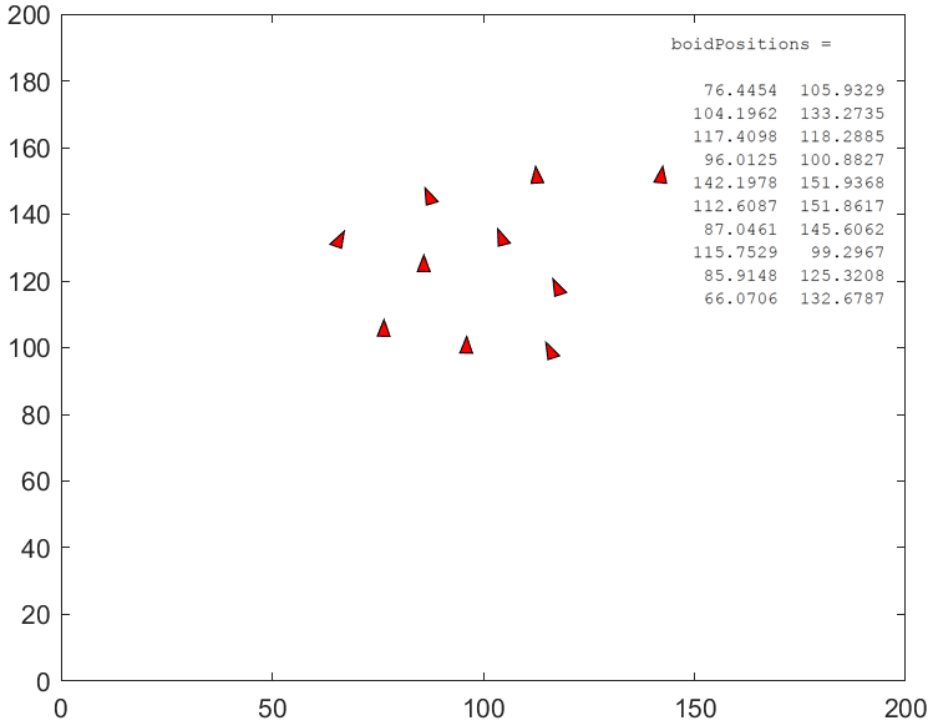
Plottar från Matlab



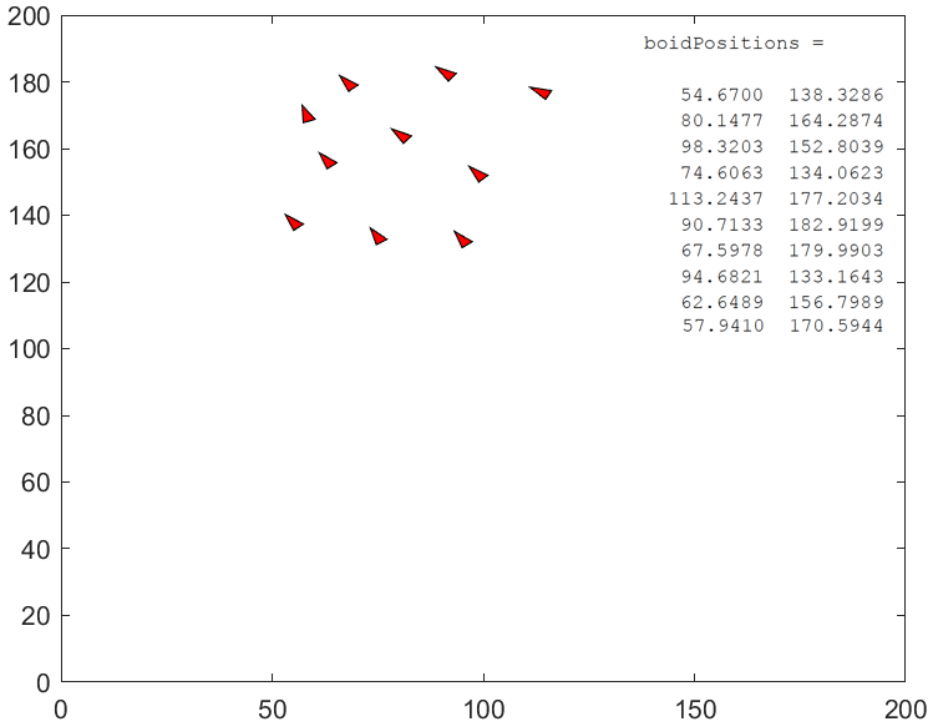
Figur A.1: Positioner för tio boids vid iteration 1



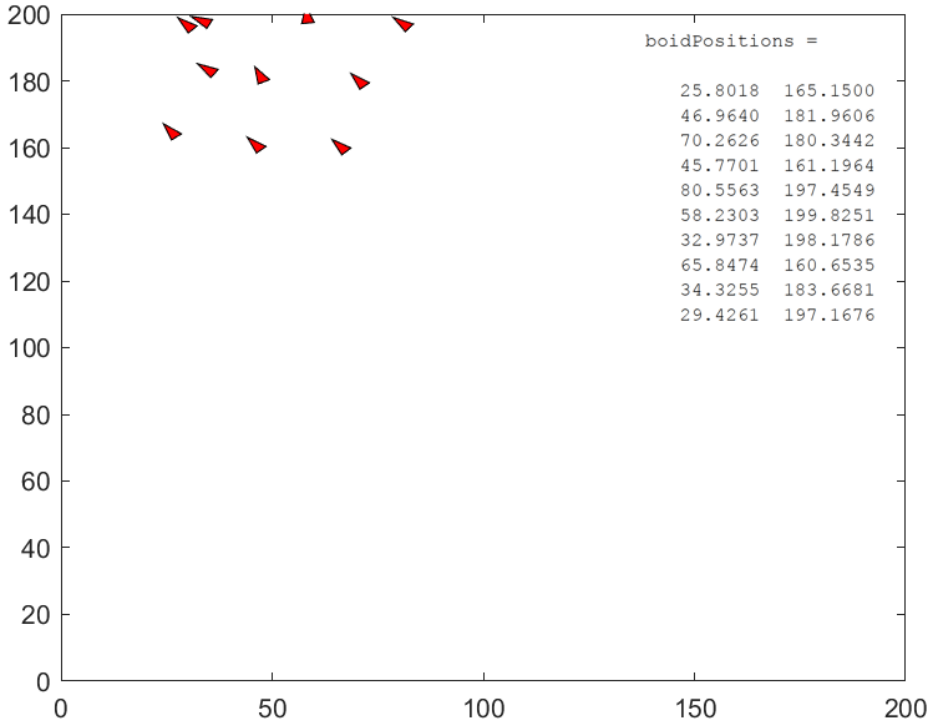
Figur A.2: Positioner för tio boids vid iteration 40



Figur A.3: Positioner för tio boids vid iteration 80



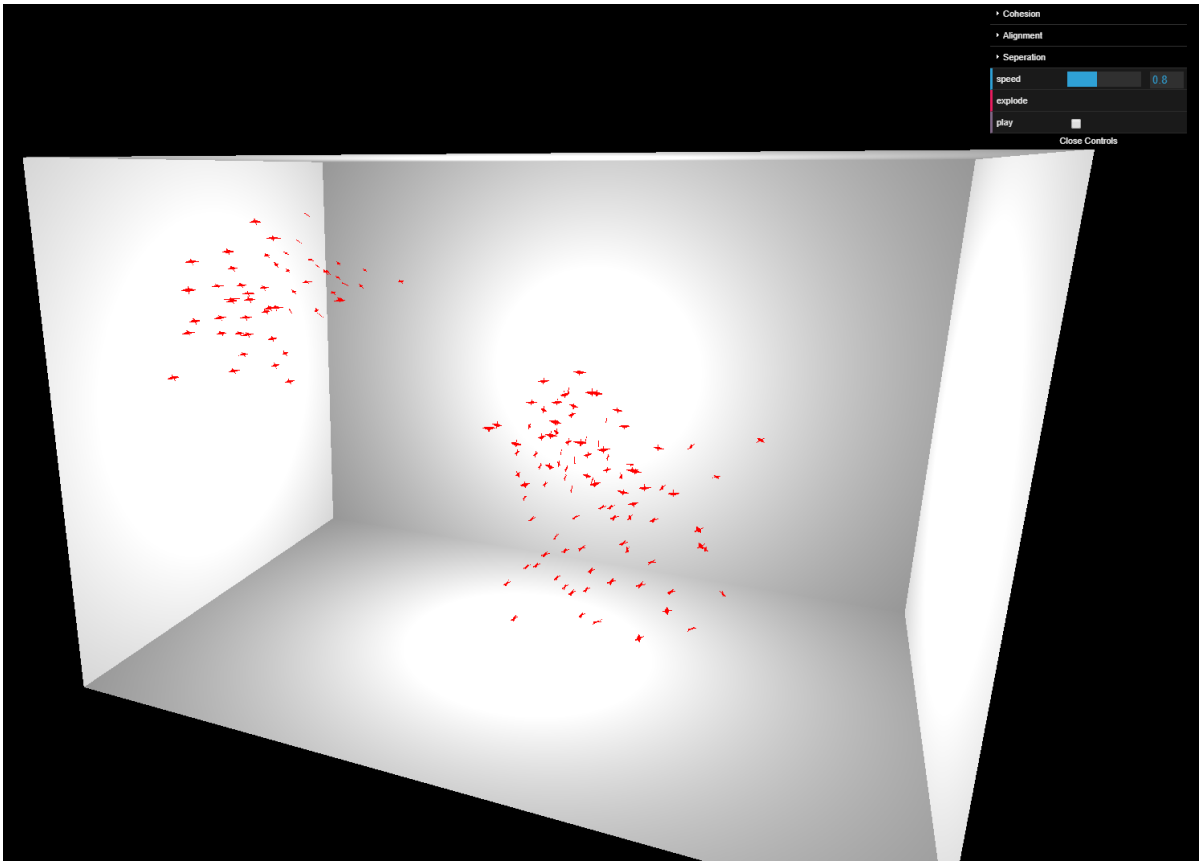
Figur A.4: Positioner för tio boids vid iteration 120



Figur A.5: Positioner för tio boids vid iteration 160

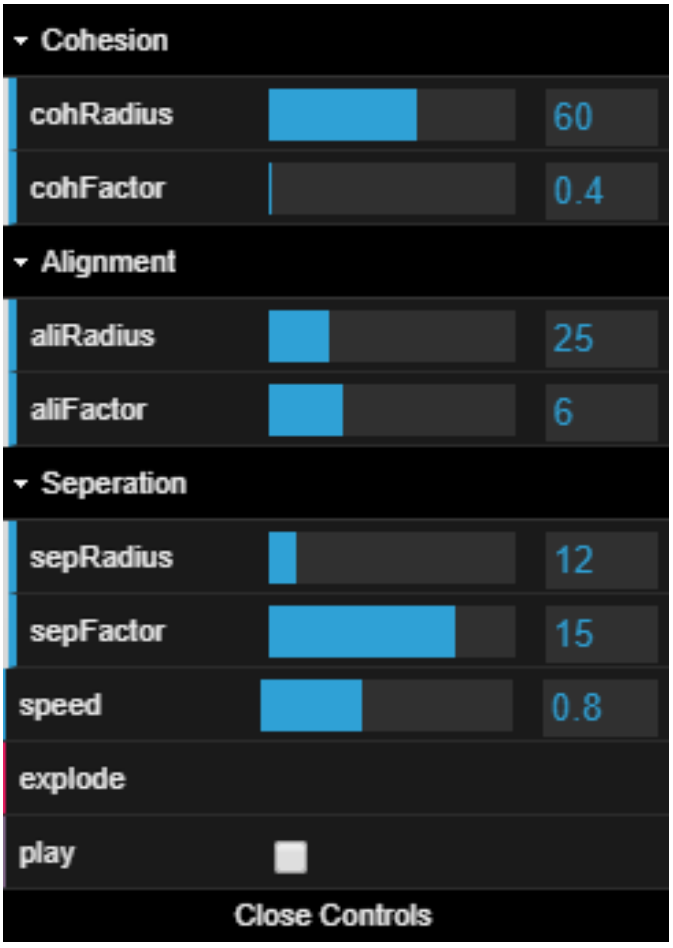
Bilaga B

Bilder från visualiseringen i JavaScript



Figur B.1: Överblick på webbläsarfönstret

Figur B.2: En grupp av boids i visualiseringen



Figur B.3: Reglage för anpassning av simuleringens variabler