Sample report

Boidによる魚群アタックシミュレーション

1　Boidの概要

Boid とは 1987年に Craig Raynolds の発表した，以下の 3 つのルールを定めるだけで動物の群れのシミュレーションが行えるという理論である [1].

1. Collision Avoidance: 近くの仲間を避ける
2. Velocity Matching: 近くの仲間の速度に合わせる
3. Flock Centering: 近くの仲間の群れに近づく

ルール 1 は速度を調節したり方向転換を行うことで極めて近くにいる仲間や障害物と間隔をあけることを意味しており，衝突を避ける効果がある．ルール 2 は衝突しない程度に近くにいる仲間と同じ方向に進むことを意味し，ルール 1 と合わせることで群れの中に居続ける効果をもたらす．ルール 3 は近くの群れに向かうことを意味し，孤立した個体を群れに引き込む効果がある．ただしこれらは必ずしも同時に成立しないため，実際には自分の近傍の数匹に対してルールを適用したりこれらのルールに重みづけを行い各個体の移動を決める必要がある．

2　シミュレーションの実装

魚は捕食されるリスクを減らすために“魚の学校”を形成する [2] ことの確認を目的とし，本稿では捕食者のいる空間での魚の群れのシミュレーションを行う．教科書 [3] に掲載され，伊庭研究室のホームページ上にコードのある simpleObserverBug を参考に実装を行った．コード内にコメントとして簡単な説明はあるが，本章では主な変更点とパラメータについて述べる．

2.1　simpleObserverFish.java

これは simpleObserverBug.java にわずかな変更を施したものである．プログラム終了時に ObserverSwarm.java の DeathCount() という関数を呼ぶことで，1．全ての魚の捕食された回数の合計，2．各魚の移動回数，3．各魚が 1 歩移動するときに捕食される確率，の 3 つを出力する．

2.2　SeaSpace.java

これは FoodSpace.java からエサの配置を取り除いたものであり，特筆すべき事項はない．

2.3　ObserverSwarm.java

これは ObserverSwarm.java にいくつか変更を施したものである．元のプログラムでは Bug のみを描画するが，これを Fish と Predator を描画するように変更した．また，2.1 節で述べた魚の被捕食回数などを出力する関数が末尾に追加されている．

2.4　Model.java

これは ModelSwarm.java を元に書かれたコードである．元のプログラムでは Bug のみを描画するが，これを Fish と Predator を描画するように変更した．主な変更点は以下である．また，このコード内で指定するパラメータについては表 1 の通りであり，括弧内は提出時点で代入されている値を示す．

- 捕食者（Predator）を配置する．密度を指定するか，初期位置を指定することにより配置する．標準では中心に 1 匹配置する．
- buildActions() 内で，1．捕食者が動く，2．魚が捕食されるかどうか判定する，3．魚が動く，の 3 つの行動を繰り返す．
- 2.1，2.2 節で述べた被捕食回数の計算を行う関数 DeathCount() の追加．

表 1　Model.java におけるパラメータ

パラメータ（設定値）	説明
worldXSize（160）	空間の横幅
worldYSize（160）	空間の縦幅
fishDeinsity（0.008）	最初に各点に魚が置かれる確率，魚の密度
predatorDensity（0.001）	最初に各点に捕食者が置かれる確率，捕食者の密度

2.5　Fish.java

これは Bug.java の特に step() 関数に大幅な変更を加えたコードである．主な変更点は以下であり，コード内で指定するパラメータについては表 2 の通りである．

- 死んだ回数（int die）と移動回数（int loop）を保持する．
- 魚の位置は double 型とした．2 つの魚が整数に直したとき同じ位置にいるとき画面上では点は 1 つしか描画されないが，内部的に個体数は減らない．
- 現在位置（とその近接）に捕食者がいるかどうかを判定し，生死を判定する関数 liveordie() の追加．捕食された場合，ランダムな地点で復活する．
- 2 点を引数としてなす角度を返り値とする関数 double vectorAngle(double x0, double y0, double x1, double y1) の追加．Math.atan2 は-π から π までの角度を返すため，これを 0 から 2π に直すのが主な目的である．
- ドーナッツ型世界（左右，上下が繋がっている世界）における 2 点間の距離を返す関数 double donutDistance(double x0, double x1, double worldsize) の追加．例えば横幅が 10 のとき，0 と 10 が繋がった円状の空間のため，9 から 1 への距離は-8 ではなく 2 である．2 点と世界の幅を引数としてこの計算を行う関数．

以下は step() 関数内の変更点である．

- Boid の 3 つのルールによる動き，前回進んだ方向に次も進もうとする動き，捕食者から逃げようとする動きに重みをつけて次の移動方向を決めるようにした．
- Boid の 3 つのルールの実装．各魚は near 内に仲間を発見したとき，正反対の方向へ進もうとする．near 外で middle 内に仲間を発見したとき，その仲間と同じ方向に進もうとする．middle 外で far 内に仲間を発見したとき，それらの重心方向へ進もうとする．
- 捕食者から逃げる動きの考慮．avoidrange 内に捕食者を発見したとき，捕食者の進行方向から垂直に逃げようとする．このとき図 1 に示すように，2 つの候補のうち捕食者から遠ざかる方向へ移動しようとする．

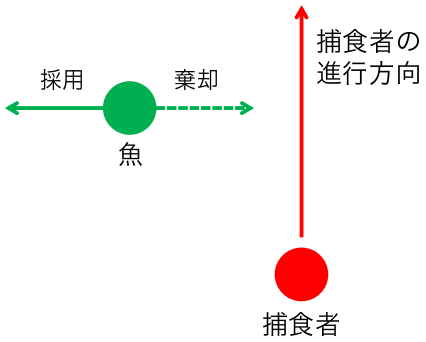

図 1　捕食者を発見時の魚の逃げる方向

- 移動方向が決まったら，最後にわずかに方向をずらす．完全に同じ位置に 2 つの魚がきたとき，以降全く同じ動きをするのを防ぐため．
- Boid のルール 3 に基づき群れの中心に向かって移動するとき，わずかに速度を上げる．孤立した魚が群れを見つけたときや，群れから置いていかれそうなときに群れに合流するため．
- 魚は急な方向転換ができないとする．進みたい方向と前回進んだ方向との間の角度に上限を設ける．

表 2　Fish.java におけるパラメータ

パラメータ（設定値）	説明
v（1）	魚の移動速度
accel（0.01）	魚が群れに合流するとき加速する速度の上限
near（3）	Boid のルール 1 を適用する範囲
middle（15）	Boid のルール 2 を適用する範囲
far（30）	Boid のルール 3 を適用する範囲
straightWeight（1）	前回と同じ方向に進もうとする重み
nearWeight（10）	Boid のルール 1 を適用する重み
middleWeight（1）	Boid のルール 2 を適用する重み
farWeight（0.3）	Boid のルール 3 を適用する重み
randomdir（$\pi/180$）	ランダムに進行方向をずらす際の最大角度
avoidrange（8）	捕食者を発見する範囲
avoidWeight（10000）	捕食者から逃げる重み
curve（$\pi/3$）	魚が曲がれる最大角度

2.6　Predator.java

これは前節の Fish.java をベースに捕食者用に書き換えたものである．この中で用いられるパラメータについては表 3 で，変更点については以下に述べる．

- 捕食者も魚と同様に速度と曲がれる最大角度を持つが，捕食者と被捕食者の体格を考慮し，魚に比べ捕食者の速度は大きく，曲がれる角度は小さくした．
- 捕食者は findrange 内の自分に最も近い魚を狙う．それが追いつける距離，曲がれる角度内にいれば捕食する．また，確実に捕食できるならば捕食する瞬間に少しだけ加速できる．魚が見つからないときは前回の方向から少し逸れた方向へ進む．
- 捕食者は自分と同じ位置にいる魚だけでなく，自分の周りの狭い範囲の魚を移動することなく捕らえられるとする．これは周りの水ごと魚を吸い込む鯨の捕食法や，大きなサイズの捕食者を意識したものである．捕食者が人間であれば手や銛，網などを用いた捕獲にあたる．

表 3　Predator.java におけるパラメータ

パラメータ（設定値）	説明
v（3）	捕食者の移動速度
accel（0.3）	魚を確実に捕食できるとき加算できる速さ
randomdir（$\pi/18$）	ランダムに進行方向をずらす際の最大角度
findrange（30）	魚を発見する範囲
curve（$\pi/12$）	魚が曲がれる最大角度
eatrange（0.3）	自分の周りの魚を移動することなく捕食できる範囲

3　シミュレーションの結果

図 2 に開始からしばらくして群れを形成しはじめたときの様子を示す．緑が魚，赤が捕食者を表す．捕食者の方が探知能力は高く速度も大きいにも関わらず，魚の群れに大きく避けられて捕食し損ねている様子がわかる．また，周りに同調して動くという魚のルールのおかげか，自分の探知範囲内に捕食者が入っていない魚も捕食者から距離をあけるように動いていることがわかる．結果として，捕食者は左下の魚群に対し下から入り何も捕食できずに左へ抜けていった．

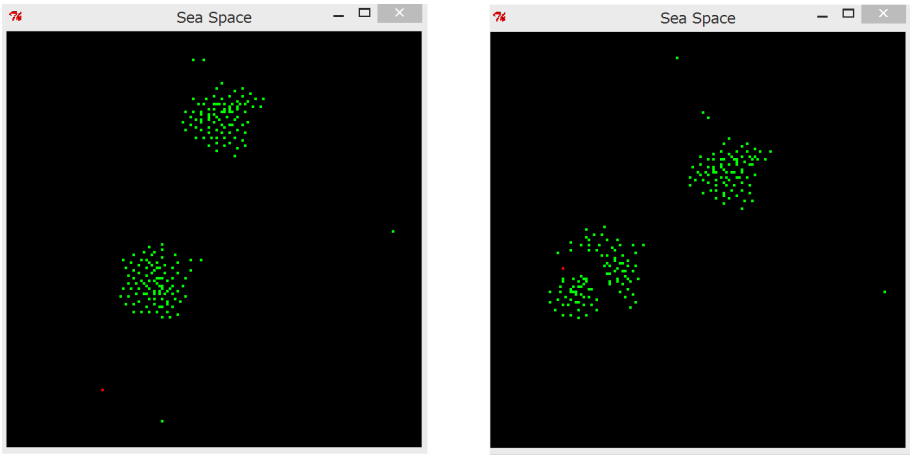

図 2　2 つの魚群を形成した状態（左）と捕食者を避ける群れの様子（右）

図 3 にさらにしばらくして 1 つの群れとなったときの様子を示す．先ほどと同様に捕食者をうまくかわしていることがわかる．また，捕食者によって分断された群れが右下へ移動する間にはやくも再び 1 つの群れを形成している様子がわかる．

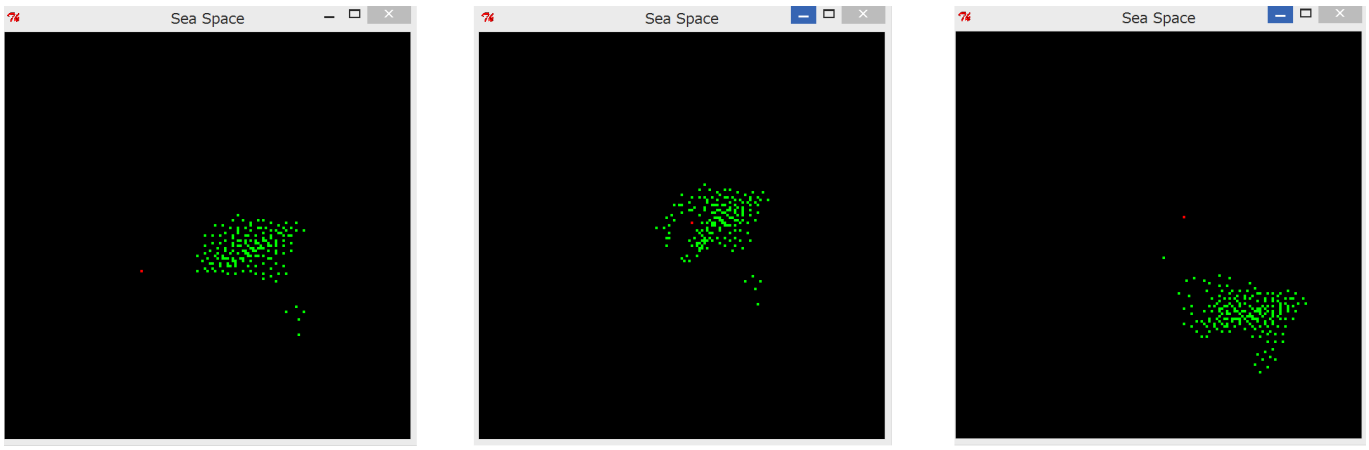

図 3　1 つの魚群を形成した状態（左），捕食者が魚群へ襲い掛かる様子（中央），再び魚群を形成する様子（右）

図 4 は群れから孤立した魚が捕食される様子を示したものである．中央やや下の孤立した魚は捕食者により群れが分断された際に群れへの合流が遅れたものだが，この直後に捕食されてしまう．群れに突っ込まれた場合に犠牲が大きくなるように思われるが，実際は一匹も捕食されないことも多い．その一方で孤立した場合はあっさりと捕食される様子が度々見られる．

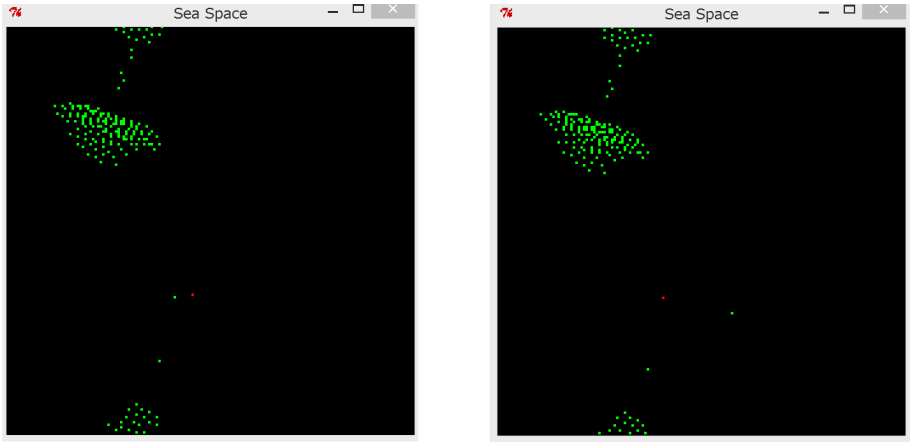

図 4　群れから孤立した魚が狙われる様子（左），捕食されランダムな場所で復活した様子（右）

4　シミュレーションを用いた考察

4.1　群れをなすことによる死亡率への影響

魚群に属することで捕食者に捕らえられる確率が減るのかを確認するため，以下の 3 つの条件下でシミュレーションを行った．ただし avoidrange=8 では魚は十分に遠くから捕食者を発見し，捕食される頻度が極端に少ないため，捕食者の速度と等しく avoidrange=3 とした．移動回数は 3,000 回程度で終了とした．捕食者は 1 匹，魚の密度は 0.01，空間の広さは 120 × 120 である．

1. 3 章で述べた設定値のパラメータを用いた場合．
2. 群れを作らず捕食者から逃げるという動きのみを行う場合．すなわち nearWeight=middleWeight=farWeight=0 とした場合．
3. 群れをなす重みを極端に大きくした場合．3 つの Weight パラメータを設定値の 100 倍にした場合．捕食者をある程度避けるが，それより群れをなすことを優先する．

結果を図 5 に示す．条件 3 では倍以上も被捕食率が高いことがわかる．条件 1 と 2 ではほとんど差がないが，捕食者が魚を食べるのにかかる時間や同時に食べられる匹数に限界があることを考えると，群れで行動する際の被捕食率は更に下がると考えられる．また，群れをなすことには中心にいる魚への水流の影響を軽減するなどの利点があるため，例え被捕食率が個別で行動したときより大きく下がらなくとも，上がらないのであれば群れをなすべきだとも言える．

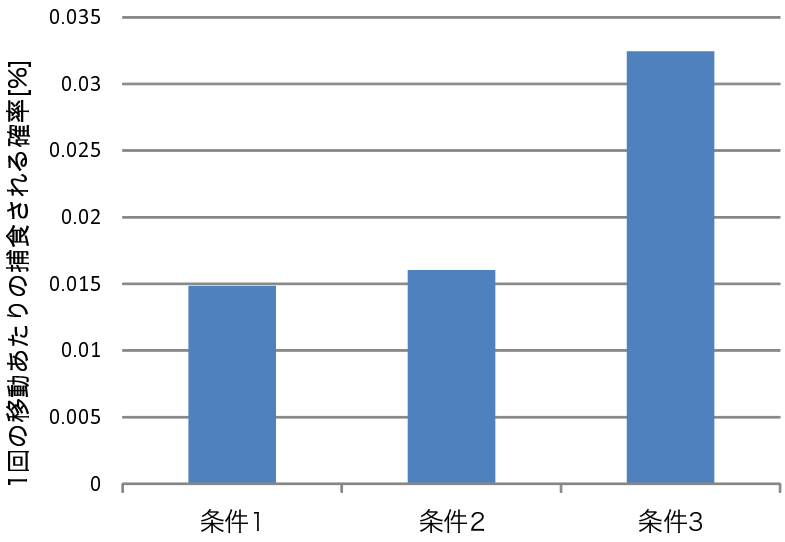

図 5　魚群をなす場合（条件 1），魚群をなさない場合（条件 2），捕食者がいても魚群を優先する場合（条件 3）の被捕食確率

4.2　捕食者の数の影響

捕食者が複数いた場合に捕食者 1 匹あたりが捕食できる魚の数がどう変化するため，捕食者を 1，2，3，4，5，10，20，50，100 匹としてシミュレーションを行った．魚の密度は 0.01，空間の広さは 120 × 120 である．結果は図 6 のように，捕食者が多いほど 1 匹あたりの捕食できる数は増加する傾向が見られた．捕食者が多い場合，群れを分断する回数が増えて魚が群れをなすのが追いつかないのが理由と考えられる．捕食者は魚群を見つけるとそちらへ向かうため，捕食者には群れを構成するような動きを組み込まなくとも，間接的に互いに近づくこととなる．これは捕食者のなす群れとも考えられ，捕食者も群れをなすことで恩恵を得られると考えられる．

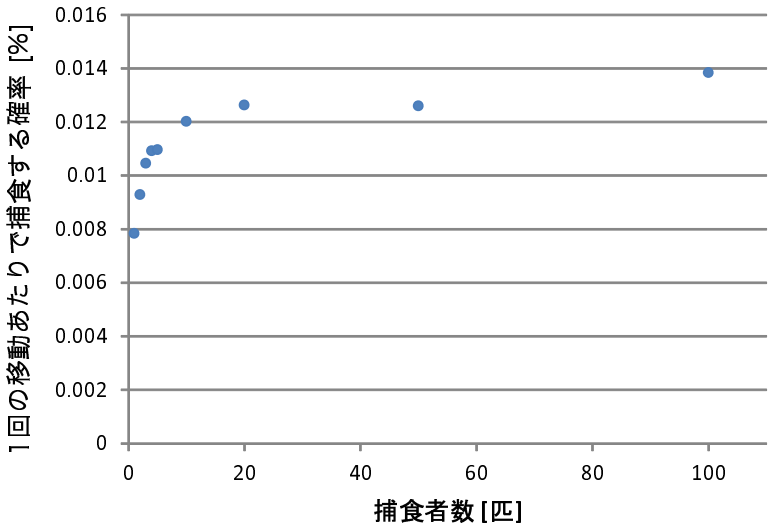

図 6　捕食者が複数いる場合の捕食者 1 匹あたりの捕食数

4.3　捕食者のサイズの影響

鯨は魚群の周りの海水ごと吸い込み，あとから水をろ過することが知られている．こういった捕食者に対しては群れをなすことは不利に働くと考え，eatrange を 1 から 5 まで変化させて，群れをなす場合となさない場合との比較を行った．その他の条件は，avoidrange=8 である点以外は 4.1 節の条件 1（群れあり），条件 2（群れなし）と同じである．結果は図 7 のようになり，やはり捕食者が大型で一度に広範囲の魚を食べられるほど群れをなすことは不利という結果が得られた．

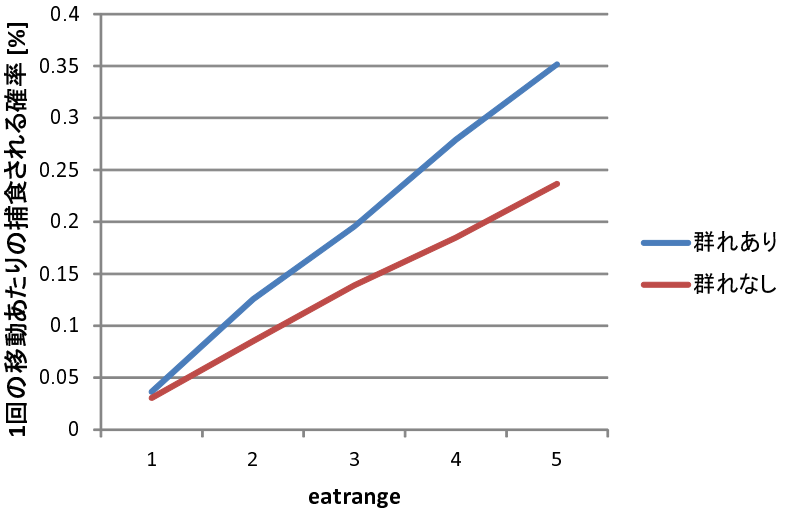

図 7　捕食者のサイズが大きい場合の被捕食確率

参考文献

[1] Craig, R.W. Flocks, herds, and schools: A distributed behavioral model. ACM Siggraph Computer Graphics, Vol. 21, No. 4, pp. 25–34, 1987.

[2] Partridge, B.L. The structure and function of fish schools. Scientific american, pp. 114–123, 1982.

[3] 伊庭斉志 (2007)『複雑系のシミュレーション』 コロナ社.