《小特集》

複雑に自己組織化されたBoidの研究

蔡　東生*

ABSTRACT Boid algorithm is first proposed by Greg Reynolds in 1989 to express some flock or grouping behavior of animals in computer graphics. The algorithm is simple and it tells only three rules to each boid in the flocks: (1) Avoid colliding each other; (2) Keep the same speed as the nearest neighbor; and (3) Fly towards the center of the flock. Giving these three rules to each boid, all boids organized into one flock and keep flying. Of course it is almost impossible to predetermine the flying paths of all boids, and the boid algorithm gives surprisingly good result with only three simple rules to express flocking behavior of animals. Numerous researches have been performed regarding to the boid and flocking algorithms. In the present report, using the boid algorithm, first, we show the transient avalanching phenomena of flocks caused by the attack of predator using a famous cell automaton named the sandpile model that that is first introduced to explain the self-organized criticality phenomena. Second, using the so-called chaos game algorithm, we show the take-off behavior of flocks from complex structures like a tree.

1. Boid

Boid（bird-oid，鳥もどきの略）は1989年に*Craig Raynolds*によって，考え出された群れの動きを作り出すシミュレーションアルゴリズム[1,2,3]である．
このBoidの群れは，個々の動きは単純なルールに従い，それらの相互作用により全体として複雑な動きをする群れに見えるというものである．このシミュレーションされた鳥のような各オブジェクトのこともBoidと呼ぶ．

1.1 Boid

鳥や魚，陸上動物の群れの集合的な動きは，自然界においてしばしば見ることが出来るとても美しい物である．群れ全体が一つの生き物のように美しく，滑らかに動く．

コンピュータグラフィックスのアニメーションとして群れを表現する時には，人間の手によって大量の各個体についての独立した経路の記述が行われることが多い．大量の経路の記述は大変面倒であるだけでなく，個体同士が衝突をしないなどの制約の保証がない．また群中の一匹の経路だけを変更したい時でも全ての個体の経路を見直さなければならないなどと編集も困難である．そこで，群れの各個体の動きをシミュレーションし，それらが相互作用するBoidという群れアルゴリズム[1]が1989年に*Craig Reynolds*によって考え出された．このアルゴリズムは，現在では，映画やアニメーションなどで群れの動きを表現する時は欠かせない手法となっている．

このBoidのアルゴリズムは，一見複雑に見える群れの動きも，個々の動きは単純なルールに従い，それらの相互作用により全体として複雑な動きをする群れに見えるというものである．

1.2 Boidのルール

Boidは基本的に以下の3つのルールから成り立ち，そのルールを個々の物体に与える事で群れを成す様子をシミュレーションできる．

1. 衝突回避（近くのBoidとの衝突を避ける）
2. 速度調和（近くのBoidとの速度ベクトルを合わせようとする）
3. 群れの集中化（群れの中心に向かおうとする）

この3つのルールはそれぞれ優先順位づけされ，それぞれ強度パラメータを持ち，加速度ベクトルとして計算される．

2. 自己組織化臨界現象と砂山モデル

この章では，外敵に群れが襲われた際に，逃避行動意識もしくは危機意識の伝搬により群れが崩れる様子

Self-Organized Complex Boid. By *DongSheng Cai* (Inst. Inofr. Sci. & Elec., University of Tsukuba).

* 筑波大学電子情報工学系

を表現するため自己組織化臨界現象と砂山モデル[4]について紹介する.

2.1 自己組織化臨界現象（SOC）

自己組織化臨界現象（self-organized criticality, SOC）は, 1987年に*Bak, Tang, Wiesenfield*[4]により提唱された.

自己組織化とは, 外部からコントロールを受けることなく自発的に自然とシステムがある構造を形成し, 無秩序な状態から秩序だった状態に時間発展していくと現象をさす. 他方, 臨界現象とは2次相転移が起こる相転移点（臨界点）において見られる現象である. ゆらぎ, 相関長の発散, スケール不変性, 冪乗則, フラクタル構造と言った現象がみられる. 温度, 圧力あるいは磁場などといった外部パラメータを調節してちょうどそれらの大きさがある臨界値にあったときだけに見られる, 特異な現象である. この一見反する2つの用語をつなげた自己組織化臨界現象は, システムが自分でパラメータを調節することにより自己組織化へすすむ臨界状態に発展し, その状態を維持する際にみられる現象を意味する. 一度, 系がこの状態（minimally stable state）になると, 系の小さな変化でも全てのサイズと存続時間が, 系の多くの成分に影響を与え, 激変に導く連鎖反応を起こす. 自己組織化臨界現象の挙動は地学（地震, 火山爆発, そして眺望の形成）, 天文学（パルサーのグリッチや太陽のフレア）, 経済学（株式相場の揺らぎ）, 生態系（進化）, 流体力学（乱流）などの多くの系で起こる変化の説明に応用されている[4,5].

自己組織化臨界現象にある系の発生条件をまとめると, 以下のようになる.

- agentの数が過大, 過小とならない.
- agentは局地的な情報に基づき相互作用する.
- ノイズの影響が全体に伝搬する事が出来る.

ここでagentは構成要素の最小単位を表す.

2.2 砂山モデル

Bak, Tang, Wiesenfeld（BTW）が1987年に提案した確率的セルオートマトンは, 砂山モデル（Sandpile model）と呼ばれ, 自己組織化臨界現象を示す代表的なモデルである[4,5]. 砂山の傾斜の角度には安定な角度というものがある. その角度よりも傾斜がなだらかな間は, 砂の堆積によって傾斜が大きくなるが, それをこえて急になるとなだれ（土砂崩れ）が起こってしまい, 自然と安定な角度に落ち着く. この角度を安息角という. 角度とは物のスケールによらない量であり, 小さな砂山も大きな砂山も安定な状態は相似形になる. また, スケール不変性は1つの砂山の斜面に起こる様々なななだれの分布にも現れる. 安息角を保つために, ある場合にはとても小さななだれで済むこともあるが, 時には少量の砂を加えたことが引き金となって, 麓のほうにまで及ぶ大規模ななだれに発展することもある. このようにさまざまなスケールのなだれが共存することになり, 全体としてはスケール不変な分布（冪分布）が実現される[4,5]. 2次元の砂山モデルの実現方法を以下で説明する.

一辺の長さがの2次元有限格子$(L\times L)$を考える. $L\times L$の格子上の各サイト(x,y)にランダムに整数値確率変数$z(x,y)(0,1,2,3)$をふる. $z(x,y)$は(x,y)サイトでのブロック変数（砂山の勾配）を表すものと考える. ある分量の砂を一単位として考え, それが位置する場所とそれを単位として測った砂山の高さの勾配を, 共に離散変数とみなす. 以下のルールに従って時間発展させる.

2次元有限格子$(L\times L)$からランダムにサイト$z(x,y)$を選び, そこサイトのブロック変数を1増やす.

$$z(x,y)\to z(x,y)+1 \qquad (2.1)$$

これは, 砂の堆積によりサイトの勾配が増していく様子を表す.

安息角に相当する閾値をz_c=3とする. あるサイト(x,y)で$z(x,y)>z_c$以上になったときには砂山の崩壊がおこり, そのサイトの勾配は無くなり, 周囲のサイトの勾配が増す.

$$\begin{aligned} z(x,y)>z_c \ :\ & z(x,y)\to z(x,y)-4 \\ & z(x,y)\to z(x,y\pm1)+1 \\ & z(x,y)\to z(x\pm1,y)+1 \end{aligned} \qquad (2.2)$$

あるサイトでの勾配$z(x,y)$が減る代わりにその周囲の勾配が増すことになる. 場合によってはその周囲の勾配が閾値z_cを超えることになり, 2次的ななだれが起こる. このようになだれが連鎖的に広がることとなる.

例外として閾値を超えた勾配を持つサイトが有限格子の境界に位置していたときには, その外側には伝わらないものとする. なだれが格子領域外にまで及んだときは, 外側に流出した土砂は散逸に対応する. (2)のプロセスは有限格子$(L\times L)$上すべてのサイトの変数$z(x,y)$が閾値z_c以下の値に落ち着くまで繰り返される.

3. 砂山モデルの群れへの適用

本研究では, 外敵に襲われた場合を想定した群れの振る舞いの作成を自己組織化臨界現象の代表的なモデルである砂山モデルを用いて試みた. その方法と実行結果をこの章で示す.

3.1 群れの崩れと砂山モデル

自己組織化臨界現象は系の小さな変化でも全てのサイズと存続時間が，系の多くの成分に影響を与え，激変に導く連鎖反応を起こす．この小さな変化によって激変に導く連鎖反応を起こす様子が，群れが崩れる様子に当てはまると考え，自己組織化臨界現象の代表的なモデルである砂山モデルを群れが崩れる様子に適用した．

本研究では，外敵に襲われた場合を想定した群れの振る舞いの作成を砂山モデルを用いて試みた．この場合は群れを保とうとしている群の各個体の心理と，危機感が伝わり，こらえきれずに大規模な崩壊を起こす場合が考えられる．砂山モデルが小規模ななだれを起こしている場合は，微少の崩れを見せる．この場合は自分でパラメータを調節することにより自発的に臨界状態に発展し，その状態を維持しようとしている状態であると考えられる．砂山モデルが大規模な（格子状のほとんどのサイトにわたるような）なだれを起こした時には，群れが大きく崩壊する．小さな変化でも全てのサイズと存続時間が，系の多くの成分に影響を与え，群全体の危機感が蓄積し，激変に導く連鎖反応を起こす状態であると考えられる．

また，スケール不変な冪乗則によって砂山モデルのなだれの大きさ頻度は保証されている．小規模ななだれは頻繁に起こるが，大規模ななだれは起こりにくい．冪乗則に従う事により，群れの崩れるサイズへのある程度の制約が得られ，自然な表現が出来ると考えられる．

3.2 Prey と Predator

本研究では，捕食者・被捕食者系（Predator-Prey 系）[6]においての動作の作成を試みた．

被捕食者は，種の保存，外敵からの防衛などの理由から群れをなすことが多い．そのため，群れを成している Prey が Predator に追われているという動きを想定した．

Predator が Prey を追い，Prey が Predator から逃げるという加速度ベクトルの計算を Predator-Prey の異種間相互作用として，式（3.1），（3.2）を用いる．式（3.1）は Prey を，Predator をとした時，Prey が Predator から受ける作用，式（3.2）は Predator を，Prey をとした時に Prey が Predator から受ける作用とする．

$$\text{Prey}: \quad c_p\left(-\frac{\vec{r}_{ij}}{r_{ij}^2}\right)\exp\left(-\frac{1}{r_{ij}}\right) \tag{3.1}$$

$$\text{Predator}: \quad c_a\sum_{j\in\text{prey}}\cos(\theta_j-\theta_i)\left(\frac{\vec{r}_{ij}}{r_{ij}^2}\right)\exp\left(-\frac{1}{r_{ij}}\right) \tag{3.2}$$

ここで，$\vec{r}_{ij}$ は i の位置から j の位置へのベクトル，r_{ij} はその大きさ，θ_i,θ_j は i,j それぞれの頭軸方向のベクトル，c_p,c_a は強度パラメータとする．この式によって，Pedator が Prey の群れを追い，Prey が Predator から逃げまわることになる．

3.3 3 次元砂山モデルの群れへの適用

外敵に襲われた場合を想定した群れの振る舞いの生成を Predator-Prey 系において，自己組織化臨界現象の代表的なモデルである砂山モデルを用いて試みた．その方法を以下で説明する．

まず，3 次元砂山モデルの各サイトにそれぞれ Boid を割り当てる．3 次元砂山モデルには初期値としてランダムに変数を割り当てる．そして，Predator-Prey 系において，Prey の群れに Predator がある程度接近してきたら，Predator に一番近い Boid に割り当てられている砂山モデルのサイトに 1 ずつ変数を加えていく．時間経過のもとで崩壊が起きたサイトとその崩壊で式（3.4）により変数の変動があったサイトにあたる Boid は数フレーム間，群れの引力から解放され，Predator から逃げる式である式（3.2）が大きく判定されることになる．砂粒を加えるということで Predator の接近を表し，それに対応する砂山モデルでなだれが起き，群れが崩れることになる．

上記の方法により，砂山モデルで小規模のなだれのあった時には小規模な群れの崩れがおこり，大規模ななだれがあった時には，Boid の群れは大きく崩壊する様子が表現できる．群れの大きな崩壊により，系の多くの成分に影響を与え，激変に導く連鎖反応を起こすという自己組織化臨界現象の特徴を群れの動作において表現できることになる．

3.4 実行結果

3.3 節で説明したような方法で，外敵に襲われた場合を想定した群れの振る舞いの生成を Predator-Prey 系において，3 次元砂山モデルを Prey の群れに適用した．3 次元砂山モデル 5 × 5 × 5 の 125 サイトにおいて各サイトに Boid を一匹ずつ割り当てて実行した．また，比較対象として 3 次元砂山モデルの代わりに完全乱数を用いたものを実行した．

図 1，2，3，4 において複数の赤い個体が Prey で，青い個体を Predator とする．図 1 は Predator が Prey に接近してくるところを示したものである．ここから 3 次元砂山モデルによる群れの崩れが始まる．図 2 は砂山

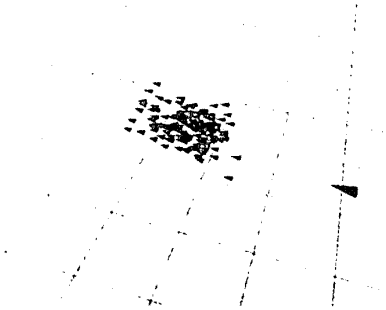

図1　Predatorの接近

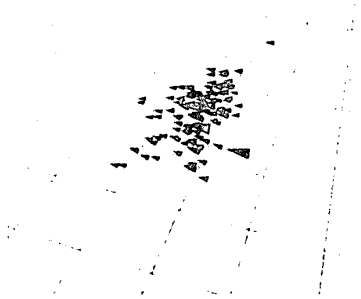

図2　小さな崩壊

図3　大きな崩壊

図4　完全乱数によるもの

モデルが小さな崩壊をおこし，群れが少し崩れているところである．このような小さな崩れは頻繁に起こりながら進行する．図3は砂山モデルが大きな崩壊をおこし，群れが大きく崩れている．このような大きな崩壊は冪乗則に従い，まれにしか起こらない．

また，図4は，砂山モデルのかわりに完全乱数を用いた様子である．完全乱数を用いた場合，乱数を大きくとった時には，図4のようにすぐに大きく群れが崩れすぎてしまい，そのまま群れが離散してしまう．また，小さく取った時はほとんど崩れず，定常の群れとあまり変わらないなど，完全乱数には砂山モデルと違い時間的制約が無いので，群れの崩れを自然に見せるための変数の調節が困難である．また，群れのサイズが変わった時砂山モデルはスケール不変な冪分布を示しているので，砂山モデルのサイズを変えれば同じような群れの崩れを表現できるが，完全乱数は全く違った結果となってしまう．

3次元砂山モデルを用いる事により，様々な規模の崩壊による自然な崩れが表現できると考えられる．

また，**図5**に鳥の形をモデリングし，それをBoidに割り当て，地面にテクスチャマッピングを施し，よりリアルに敵に群れが追われている様子を示す．

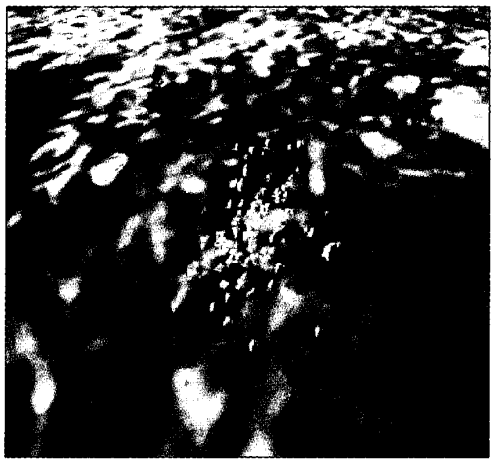

図5　鳥のモデリングとテクスチャマッピングを施した例（1）

4.　Chaos Game Algorithmを用いた群れの飛び立ち

本研究では，フラクタルモデルであるIFS（反覆関数系:Iterated Function System）[7,8]によって作成された木のモデルに，レンダリングアルゴリズムの一つChaos Game Algorithm[7,8]により得られた点をBoidの初期配置として利用し，そこから群れの飛び立つ様子を

表現する．この章ではそのことについて説明し，実行結果を示す．

4.1 複雑な形状への配置

Chaos Game Algorithm[7,8]は，複雑なフラクタル画像を生成するためのアルゴリズムである．実際の鳥や動物などがいる海岸線や山，地形，木などはフラクタルで説明される事が多く，コンピュータグラフィックスで地形や木などを表現する時もフラクタル生成アルゴリズムが用いられる事が多い．木や地形などをIFS[7,8]で生成し，同じ変換を用いたChaos Game Algorithmを用いる事により複雑な木や地形などが簡単なアルゴリズムで生成でき，その生成物の座標を簡単に得る事が出来る．L-システム[9~11]によって作成された木をIFSに変換し[10]，Chaos Game Algorithmによって得られた座標をBoidの初期配置として利用する事が出来るようにした．本稿では，L-Systemにより作成された木のモデルを使用し，そこからのBoidの飛び立ちを表現する．また，Boidの配置が分かりやすいように葉の無いモデルを用いた．その図を以下の**図6**に示す．

4.2 Chaos Game Algorithmにより得られた軌跡を用いた飛び立ち

Chaos Game Algorithmで得られた点は，Eltonの定理により，Chaos Game Algorithmのアトラクタの空間に一様に埋められていくことが数学的に証明されている．すなわち，Chaos Game Algorithmを用いた点の軌跡を追っていくと偏りがなく均一に空間を埋めていくことになる．BoidをChaos Game Algorithmの軌跡の通りに初期配置として配置し，その順序を逆にたどり順番に飛び立たせることによって，空間に配置されたBoidを，偏りなく全体的に均一に飛び立たせることができる．

Boidの初期配置をこれらで得られた点の座標に割り当てることによって，アトラクタ空間内に偏りがなく配置することが出来る．また，座標を割り当てた順に飛び立たせることで，むらのない飛び立ちが表現できると考える．群れが敵に襲われた時などの緊急時を考えた場合において，すばやく，むらのない飛び立ちが表現できるものと考えられる．

4.3 IFSのアドレスを用いた飛び立ち

Boidの初期配置をChaos Game Algorithmにより得られた点の座標に割り当て，最初に飛び立ったBoidと距離関数により得られた距離の小さいものから順に飛び立たせる．このことにより，Chaos Game AlgorithmによってBoidをIFSによって作成された木のモデルに初期配置として配置した場合，実際の距離的に近いBoidではなく，同じ枝や近い枝やに止まっているBoidから順番に飛び立たせる事が出来る．距離的に近いBoidを飛び立たせるよりも，生体の視界などを考えると，アドレスを用いたほうが，より自然な飛び立ちが表現できるものと思われる．

また，本研究で作成したIFSを用いた木のモデルには，それぞれの節にアドレスがつけられているので，そのアドレスに基づきBoidを配置する事が出来，このアドレスを利用する事により，Boidを配置したい場所に配置する事も出来る．

図7に木の節のアドレスの割り当ての例を示す．

4.4 実行結果

4.4.1 初期配置

Chaos Game Algorithmを用いた複雑な地形への配置の実行例として，図6の木のモデルを用いる．また，Boidの配置が分かりやすいように，透明度を上げた木のモデルを用い，上方に視点をおいた図を以下に示す．また，図にはChaos Game Algorithmを用いたものと，その確率を変えたものと，その比較対象として完全乱数を用いたものを示す．それぞれ125匹のBoidを配置させたものである．

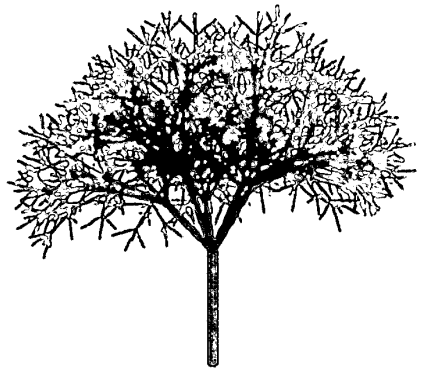

図6　使用した木のモデル

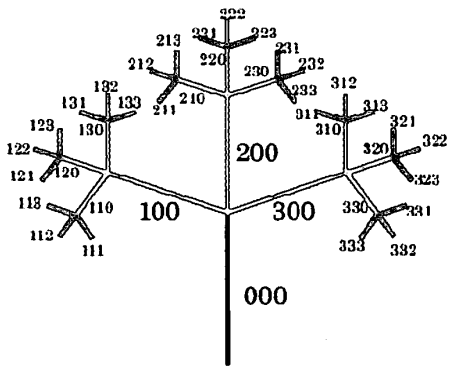

図7　IFSを使用した木へのアドレスの割り当ての例

図8は，Chaos Game Algorithmの確率を測度均一に決定したp_iを用いたものである．IFS, T_1, T_2, T_3, T_4の確率をそれぞれp_1, p_2, p_3, p_4とすると，p_1=0.28，p_2=0.274，p_3=0.184，p_4=0.262となる．図8を見ると，枝の多いところには，Boidが多く配置され，木全体にむらなく配置が行われているのが分かる．

図9は，Chaos Game Algorithmの確率を変え，むらのある配置を行った結果である．図8の時と同様に，p_1, p_2, p_3, p_4を表すとp_1=0.53，p_2=0.19，p_3=0.14，p_4=0.14によってBoidの配置を行ったものである．図6の時と比べてむらがあり，木の中心付近にはほとんど配置されていないのが分かる．

図10は，Chaos Game Algorithmの代わりに完全乱数を用いて木のモデルに配置させたものである．図8と比べると，木全体に配置されているのだが，枝の多い少ないに関係なく配置されているのが分かる．また，前方部分にあまり個体数の多くない偏りのある配置となっている．

乱数を用いた時は，木にむらなく配置させたくても偏りが出てしまう時がある．しかし，Chaos Game Algorithmを用いると，空間内にむらなく配置する事が出来る．また，むらのある配置を行いたい時は，確率を変えればむらのある配置を行う事が出来る．また，その自然なむらのある配置の生成も確率のそれぞれの値を変える事によりコントロールする事が出来るので，Chaos Game Algorithmによる配置は有効であるといえる．

4.4.2 飛び立ち

Chaos Game Algorithmを用いた飛び立ちの実行例を以下の図に示す．以下の**図11**は，Chaos Game Algorithmにより得られた点の軌跡に従い飛び立たせたものである．木のモデルに4.4.1で示したようにChaos Game Algorithmによって125匹のBoidの初期配置を行い，その位置からの飛び立ちから定常の群れに移行する様子を示している．本稿の実行では，使用したコンピュータのメモリの関係で多くのBoidを配置することが出来ず，125匹が最大Boid数となっている．

Chaos Game Algorithmにより木のモデルに配置させたBoidがむらなく飛び立ちを行っている様子である．また，飛び立ちの際にそれぞれのBoidは近くの枝の位置と角度を調べ，その枝にあたらないような方向への飛び立ちを行っている．

5. まとめと今後の展望

非定常な群れの動きとして，Predator-Prey系において，群れが敵に襲われる際の群れが崩れる動きに，自己組織化臨界現象との相似点を見出し，自己組織化臨界現象の代表的なモデルである3次元砂山モデルを群れの崩れる動きに適用し，作成した．

もう一つの群れの非定常な動きとして，群れの飛び立ち動作作成を行った．まず，Iterated Function Systemによって木のモデルを作成し，その木のモデルに同じ変換によるChaos Game Algorithmによって，Boidに初期配置を与えてやり，Chaos Game Algorithmによって与えられた点の軌跡に従って木のモデルからの飛び立ちの動作を作成した．

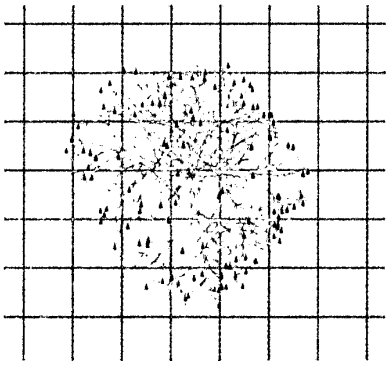

図8　Chaos Game Algorithmによる配置

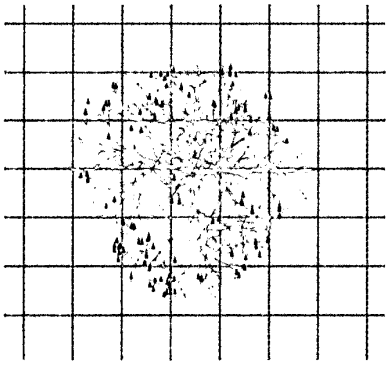

図9　確率を変えた配置

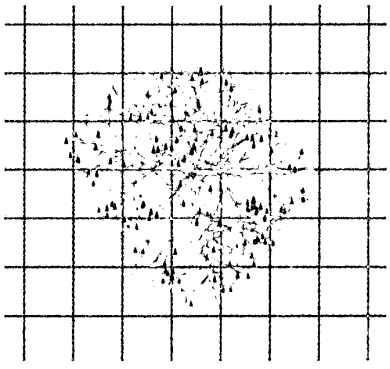

図10　完全乱数による配置

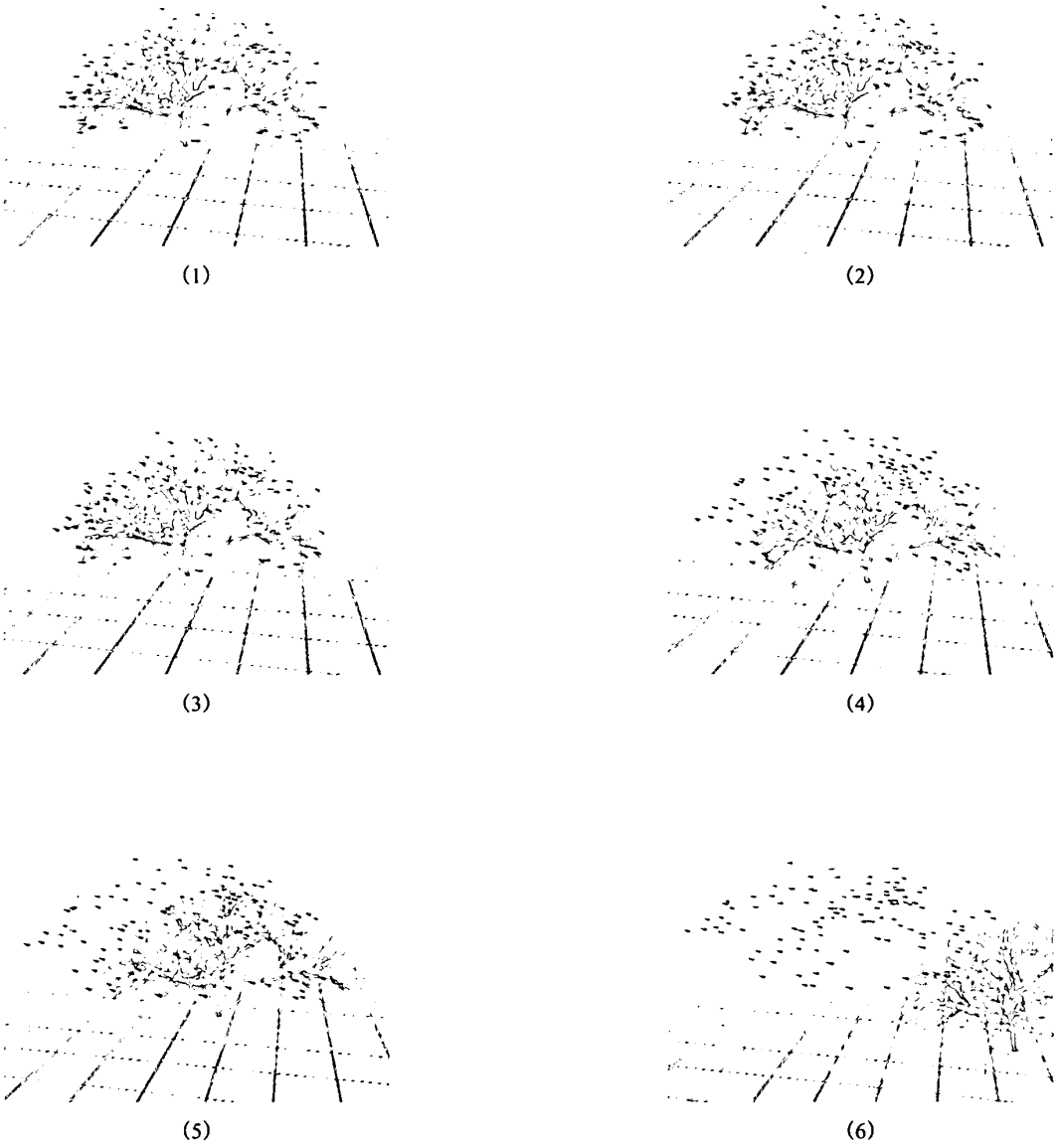

図11　Chaos Game Algorithm による飛び立ち

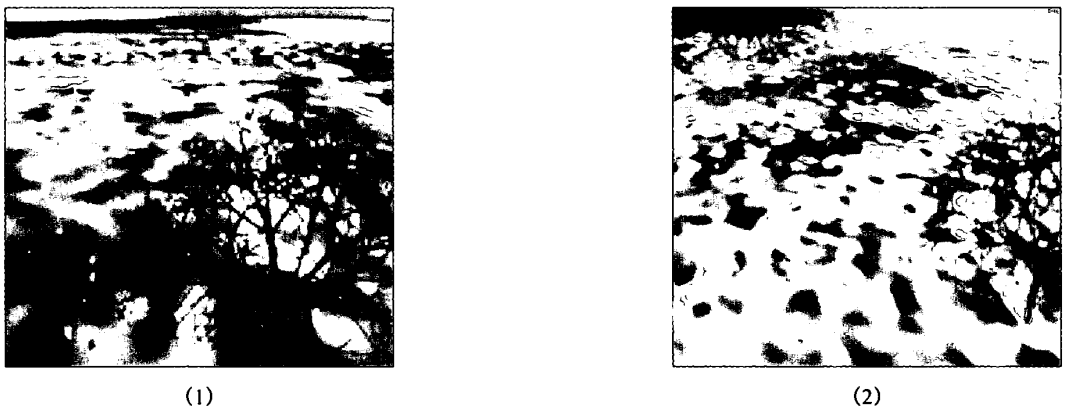

図12　鳥のモデリングとテクスチャマッピングを施した例

また，Chaos Game Algorithmの確率を変える事や，IFSのアドレスを用いる事により，群れの飛び立ちコントロールが出来る事を示した．

参 考 文 献

1) Craig W. Reynolds: Flock, Herds and Schools A Distributed Behavioral Model, Computer Graphics, 21(4), 25/34 (1987)
2) N. Shimoyama et al.: Collective motion in a system of motile elements, Physical Review Letters, 76(20), 3870/3873 (1996)
3) 合原幸一 編: 別冊日経サイエンス 複雑系がひらく世界, 日経サイエンス社 (1997)
4) Per Bak, Chao Tang, and Kurt Wiesenfeld: Self-organized criticality, Phisical Review A, **38**-1, 364/374 (1988)
5) Per Bal: How nature works : the science self-organized criticality, springer Verlag (1996)
6) Shin I. Nishimura and Takashi Ikegami: Emergence of collective strategies in a prey-predator game model, Artificial Life, 3, 243/260 (1998)
7) Michael F.Barnsley: FRACTALS EVERYWHERE, Academic Press, Boston (1988)
8) Michael F.Barnsley: FRACTALS EVERYWHERE SECOND EDITION, Academic Press, Boston (1993)
9) Przemyslaw Prusinkiewicz, Mark Hmmel, Radomir Mch: The Artificial Life of Plants, Artificial Life for Graphics, Animation, Multimedia and Virtual Reality, SIGGRAPH 98 Course 22 Notes Florida, 19-24 July 1998.
10) Przemyslaw Prusinkiewicz, Aristid Lindenmayer, The Algorithmic Beauty Plants, Springer-Verlag, New York (1990). With J.S.Hanan, F.D.Fracchi, D.R.Flowler, M.J.M.deBoer and L.Mercer.
11) Przemyslaw Prusinkiewicz, Mark Hmmel, Jim Hanan, Radomir Mch: Visual models of plant development, From M.T.Michalewicz editor, Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Science, CSIRO Publishing (1996)