2003－CG－112　(18)
2003／8／19

再帰的反覆関数系を用いたハイパーテクスチャ

長峯 望†　望月 茂徳††　蔡 東生†††

現在，コンピュータグラフィックスにおいて，ソリッドノイズは不可欠なものとなっている．ソリッドノイズとは，３次元空間に割り当てられる関数であり，立方体や球などの物体に対して自然な揺らぎを持たせることが可能である．本研究では，3 次元の Random Iteration Algorithm と再帰的反覆関数系(Recurrent Iterated Function System : Recurrent IFS)コードによって生成されるソリッドノイズについて述べ，Recurrent IFS から成るソリッドノイズを生成し、Hypertexture へ応用する．

Hypertexture using Recurrent Iterated Function System

Nozomi Nagamine†　Shigenori Mochizuki††　DongSheng Cai†††

Recently, in computer graphics, a solid noise is an indispensable tool. The solid noise is a function that the random value is assigned to each point in three-dimensional space, and it is possible to give natural fluctuation to the object such as a cube and ball. In this paper, we describe about the solid noise that is generate by using Random Iterated Algorithm and Recurrent Iterated Function System (Recurrent IFS) code in three-dimensional space. Finally, we generate a solid noise or hypertexture using Recurrent IFS.

１．はじめに

現在，コンピュータグラフィックスにおいて，ソリッドノイズは不可欠なものとなっている．ソリッドノイズとは，３次元空間の各点にランダムな値が割り当てられるような関数であり，サーフィス・テクスチャ，確率モデリング，自然現象のアニメーションなどにおいて即時的で強力なツールであることがよく知られている．ノイズは立方体や球などの物体に対して自然な揺らぎを持たせることが可能である．本研究では，3 次元の Random Iteration Algorithm と再帰的反覆関数系(Recurrent Iterated Function System : Recurrent IFS)コードによって生成されるソリッドノイズについて述べる．また，Recurrent IFS から成るソリッドテクスチャ生成をし、Hypertexture を作成する．

†筑波大学 大学院 理工学研究科
††筑波大学 大学院 システム情報工学研究科
†††筑波大学 電子・情報工学系
†Master's Program in Science and Engineering at University of Tsukuba
††Graduate School of Systems and Information Engineering at University of Tsukuba
†††Institute of Information Sciences and Electronics at University of Tsukuba

２．ソリッドノイズ

ソリッドノイズとは，３次元乱数を返す関数 $f : R^3 \rightarrow R$ で定義され，コンピュータグラフィックスに導入されたソリッドテクスチャの概念の一部である．この場合，ノイズとは，統計的な特性を持ったランダム関数である．

ソリッドノイズは３次元物体のテクスチャリングに使われる．ここで３次元物体とは３次元空間上の各点において，ある関数の値を物体の表面上の可視点におけるグレイレベルとして扱い，色を割り当てられたものである．

また，自然とは複雑さと規則性を同時に兼ね備えており，このような自然画像を作り出すことの出来るランダム関数はコンピュータグラフィックスやモデリングにおいて，非常に有効である．しかしながら，多くのプログラム言語における数学ライブラリのランダム関数は，常に適切であるとは限らない．なぜなら，これらの関数が不連続に帰着するからであるが，どんなスケールに対しても定義でき，帯域制限があり，コントロールが可能であることが求められる．

一方、Ken Perlin らによって 1980 年代に hypertexture [1][2]が提唱された．この hypertexture における彼らの

ノイズ設計は経験的なものにならざるを得ない．なぜなら，Ken Perlin らによるノイズコントロール手法は，経験に基づいて行われているからである．つまり，ノイズを経験により見つけることはできるが，任意の複雑なテクスチャノイズを思い通りに生成することは容易ではない．

また，Barnsley らは２次元での形状モデリングや，デジタル画像でのテクスチャレンダリングにカオスダイナミクスを用いた Measure Rendering Method を提案した．

本研究では，2 次元や 3 次元の物体や風景をモデリングできるという利点を持っている Recurrent IFS や L-system（Recurrent IFS へ変換が可能）を適用してノイズ設計を行うことを考える．

３．Recurrent Iterated Function System(Recurrent IFS)

本研究における Recurrent IFS とは

IFS $\{X, \omega_i; \omega_1, \omega_2, ..., \omega_N; i = 1, 2, .., N\}$ と

確率行列 $\{p_{ij} \in [0,1] : i, j = 1,2,..., N\}$ から成り以下の２つの条件を満たすものである

(i) $p_{i1} + p_{i2} + \cdots + p_{iN} = 1 \quad for \;\; i = 1,2,..., N$

(ii)どんな i,j を選んでも次を満たすような有限な整数の数列 $k,l,...,m \in 1,2,...,N$ が存在する

$$p_{ik} p_{kl} \cdots p_{mj} > 0$$

$$\omega_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a_i & b_i & c_i \\ d_i & e_i & f_i \\ g_i & h_i & o_i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} q_i \\ r_i \\ s_i \end{pmatrix}$$

本研究では空間 $X = R^3$ とする．

Recurrent IFS コードにおけるパラメータの例を表１に，マルコフ過程の概略を図１に示す．

図１は Recurrent IFS の３つの変換が作用される状態遷移を表した概略図である．この３つの変換は９つの状態遷移がある．また，１つ前の変換に ω_i が施されたのならシステムは状態 i にいるという[3]．

次に Recurrent IFS を用いてソリッドノイズを生成することを試みる．この際に使用する Borel Measure Rendering Algorithm については以下に示す．

４．Borel Measure

Measure μ はフィールド F 上で定義され，フィールドの各要素に対して正の実数を返す関数 $\mu : F \to [0,\infty) \subset R$ である

また，$A_i \in F \quad for \;\; i = 1,2,...,$ とし

$A_i \cap A_j = 0 \quad for \;\; i \neq j$

$\bigcup_{i=1}^{\infty} A_i \in F$ としたとき，次が成立する

$$\mu\left(\bigcup_{i=1}^{\infty} A_i\right) = \sum_{i=1}^{\infty} \mu(A_i)$$

しかしながら，Borel Measure は実際に測ることが出来ず，Borel Measure Rendering Algorithm としてアルゴリズム化され測られる．

５．Borel Measure Rendering Algorithm

(X, d) をコンパクトな距離空間とし，$\{X; \omega_1, ..., \omega_N; p_1, ..., p_N\}$ を確率付き IFS，A を IFS のアトラクタとする．また，この確率付き IFS に対して Random Iteration Algorithm を施して生成された点列を $\{z_n\}_{n=0}^{\infty}$ とする．このとき，空間 X の Borel 部分集合 B に Borel measure μ が存在し，以下のように近似される[3]．

$$\mu(B) \cong \lim_{n \to \infty} \left\{ \frac{N(B,n)}{(n+1)} \right\}$$

$N(B,n) = \{z_0, z_1, ..., z_n\} \cap B$ に含まれる点の総数

ただし $n = 0,1,2,...$

本研究では Y を R^3 の部分集合とし，Y 上で Random Iteration Algorithm を Recurrent IFS に対して施す．後に空間 Y がグリッドで区切られるように，十分小さな立方の区間 V で Y を分けたとき，各区間 V に存在する点の総数を正規化したものを区間 V の measure として扱う[4]． 具体的には，３次元上の空間に立方格子が張られたものとして考える．

ここで，measure をどのように扱うかという問題がある．例えば，グレイスケールの濃淡値，カラーマップに対応した色，透明度，密度，温度，ノイズ等様々なものに適用することが可能である．本研究では measure をノイズ値として扱う．

Recurrent IFS と Borel Measure Rendering Algorithm によって作られたソリッドノイズを可視化したものを図

２に示す．また，この場合，得られた measure を透明度として可視化を行い，ソリッドノイズを表現している．本研究では，便宜上 Recurrent IFS と Borel Measure Rendering Algorithm によって生成されたソリッドノイズをフラクタルソリッドノイズと呼ぶ．

６．Recurrent IFS を用いた Hypertexture

フラクタルソリッドノイズによる hypertexture 生成を行うために，［0，1］の 範囲の値を持ったオブジェクトに対する密度関数 $D(x)$を導入する．これはR^3におけるすべての点 x についての密度を表し，三次元上のオブジェクトの形状を決定する．また，オブジェクトは hard region，soft region，outside region の３つの領域を持つ．hard region とは$D(x)=1$となる部分であり，完全な固体を表す，soft region は $0<D(x)<1$ となるような部分である．また，outside region は $D(x)=0$ となる部分であり，オブジェクトが無い部分を表す．

さらに，密度調整関数(Density Modulation Function : DFM) f_i を導入する．これはオブジェクトの密度を調整するために使用される．

これにより，hypertexture は次のように定義される．

$$H(D(x),x) = f_n(\cdots f_2(f_1(D(x))))$$

$f_1, f_2, \cdots, f_n$ はそれぞれ DMF である

また，3 次元での voxel 値は，Measure Rendering Algorithm や Chaos Game Algorithm によって決定される．単純化のために$D(x)=V(x)$とし，voxel 値（measure）はオブジェクトの密度を表すものとする．

次に，本研究で対象とするオブジェクトを示す．対象とするオブジェクトは soft region を持つシンプルな球体とし，中心座標を c，半径を r，soft region を作る強度を s として次式の密度関数で定義する．

$$\begin{aligned}
&D_{[c,r,s]}(x)\\
&r_1^2 := (r - s/2)^2\\
&r_0^2 := (r + s/2)^2\\
&r^2 := (x_x - c_x)^2 + (x_y - c_y)^2 + (x_z - c_z)^2\\
&D := \ if\ \ r_x^2 < r_1^2\ \ then\ 1.0\\
&\qquad if\ \ r_x^2 > r_0^2\ \ then\ \ 0.0\\
&\qquad else\ (r_0^2 - r_x^2)/(r_0^2 - r_1^2)
\end{aligned}$$

また，$sphere(x)$ はこの関数によって作られた球体の密度を返すものとする．

次に，hypertexture を生成するために図２で示されるフラクタルソリッドノイズを DMF として使用する．ここで，フラクタルソリッドノイズを $fractal\ noise(x)$ として表す．

次にこれらをもとに作成された hypertexture と Ken Perlin らによる hypertexture を比較する．図３は Ken Perlin らによって提案された turbulence 関数によるノイズを加えることによって生成されたオブジェクトであり，図４は図２で表されるフラクタルソリッドノイズを加えることによって生成されたオブジェクトである．このとき，それぞれの密度関数は DMF を用いて次のように表される．

$$D(x) = sphere(x) \cdot turbulence(x)$$

$$D(x) = sphere(x) \cdot fractal\ \ noise(x)$$

また，図５，６，７，８はそれぞれ特定の密度以上を固体として扱いレンダリングしたものである．図５，６が Ken Perlin の turbulence 関数を用いたものであり，図７，８がフラクタルソリッドノイズを用いたものである．また，図５，７が密度 0.3 以上を固体として扱い，図６，８が密度 0.5 以上を固体として扱っている．

７．おわりに

本研究では，Recurrent IFS と Borel Measure に基づいてノイズ分布関数を作り，hypertexture におけるノイズとして用いることを提案した．このノイズ分布関数を用いることにより，次のことが考えられる．

１．１通りのノイズの加え方（hypertexture）に対して，RIFS コードの変更により多様なソリッドテクスチャが生成できる．これは従来の hypertexture よりも多様化することを意味し，すなわち，これまで表現が困難であったソリッドテクスチャや不可能であったソリッドテクスチャを生成することが可能であると考えられる．

２．Ken Perlin らの手法では，bias，gain，等の様々な密度調整関数をフィルタのように通すことによって複雑なソリッドテクスチャを実現していたが，これに対して本手法では，ノイズ分布を任意に設計することが

可能であるため単純な操作で同様のソリッドテクスチャを生成することも可能であると考えられる．

3．Recurrent IFS を使用する利点としてマルコフ確率の次数を上げることにより，より情報量の多いノイズ分布関数が生成できる．

4．問題として，任意のノイズ分布が設計可能であるということはコラージュの定理により保証されているが，任意のノイズ分布関数を設計するにはどうすればよいか，という点が挙げられる．これについては3次元のフラクタル画像符号化法の技術が適応可能であると考えられる．

現在，シンプルな Recurrent IFS によるノイズしか扱っていないため，今後，さまざまな Recurrent IFS コードを用いてノイズ分布関数を作成し hypertexture の実装を行う予定である．また，Ken Perlin らによる gain，bias,等を用いた hypertexture を実装し，従来の hypertexture では生成できない，または生成が困難なソリッドテクスチャを見つける予定ある．

表1．RIFS コードのパラメータの例

a	b	c	d	e	f	g	h	o	q	r	s
0.5	0	0	0	0.5	0	0	0	0.5	0	0	0
0.5	0	0	0	0.5	0	0	0	0.5	128	0	0
0.5	0	0	0	0.5	0	0	0	0.5	128	0	128
0.5	0	0	0	0.5	0	0	0	0.5	0	0	128
0.5	0	0	0	0.5	0	0	0	0.5	0	128	0
0.5	0	0	0	0.5	0	0	0	0.5	128	128	0
0.5	0	0	0	0.5	0	0	0	0.5	0	128	128
0.5	0	0	0	0.5	0	0	0	0.5	128	128	128

p_{i1}	p_{i2}	p_{i3}	p_{i4}	p_{i5}	p_{i6}	p_{i7}	p_{i8}
0.13	0.14	0.15	0.09	0.07	0.15	0.15	0.07
0.12	0.07	0.08	0.15	0.17	0.08	0.08	0.21
0.16	0.14	0.08	0.07	0.16	0.14	0.09	0.11
0.13	0.12	0.14	0.15	0.07	0.17	0.15	0.02
0.12	0.08	0.13	0.16	0.10	0.10	0.14	0.13
0.13	0.15	0.13	0.11	0.07	0.08	0.10	0.18
0.09	0.13	0.14	0.16	0.13	0.09	0.14	0.07
0.15	0.15	0.14	0.17	0.07	0.16	0.11	0.01

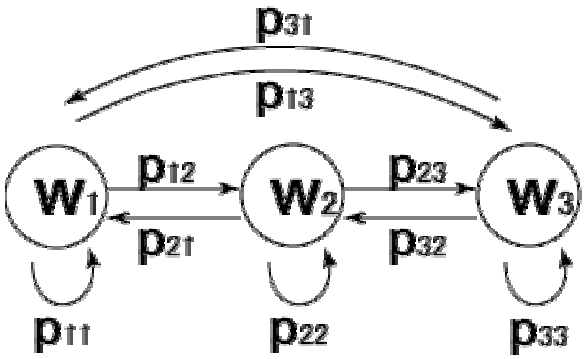

図1．Recurrent IFS におけるマルコフ過程の概略図

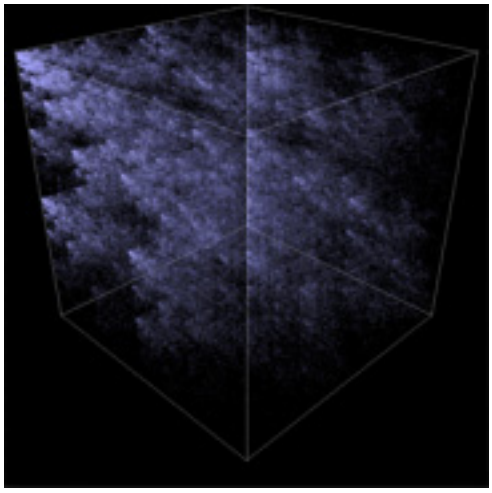
図2．フラクタルソリッドノイズ


図3．Turbulence 関数による hypertexture

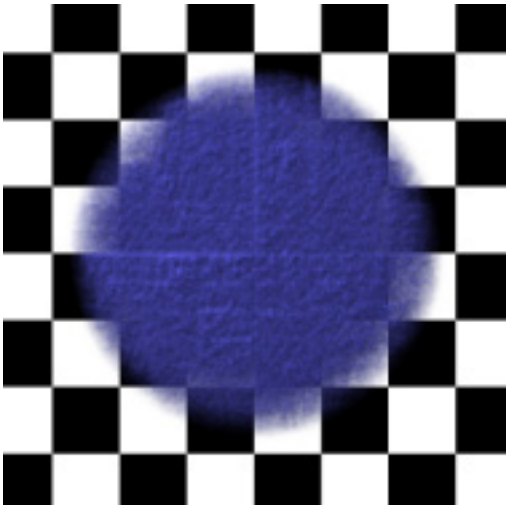

図４．フラクタルソリッドノイズによる hypertexture

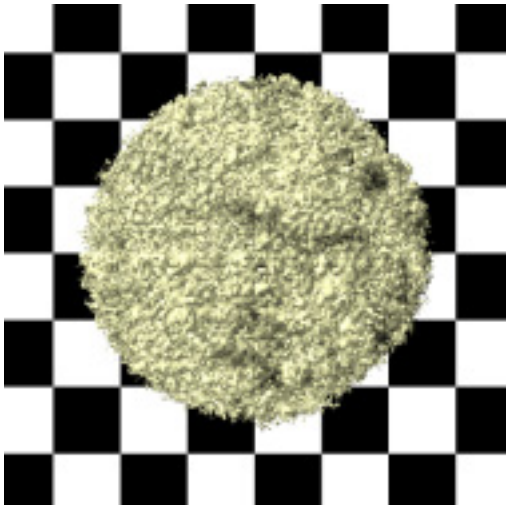

図７．密度 0.3 以上(fractal noise)

図５．密度 0.3 以上(turbulence)

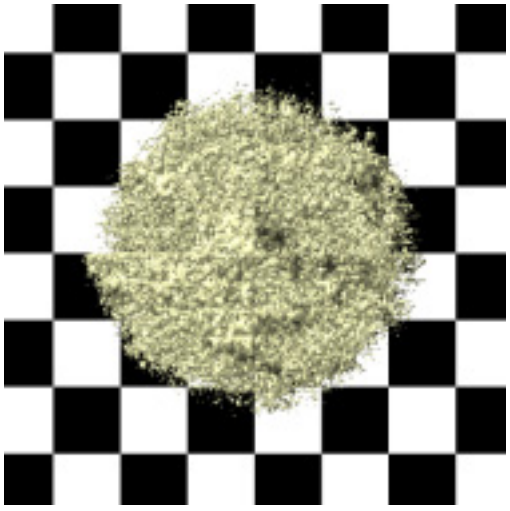

図８．密度 0.5 以上(fractal noise)

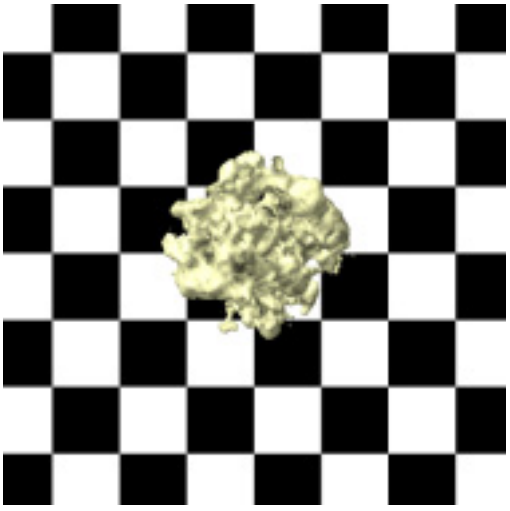

図６．密度 0.5 以上(turbulence)

参考文献

[1] Ken Perlin and Eric M. Hoffert, "Hypertexture," Computer Graphics, Vol. 23, No. 3, July 1989 (SIGGRAPH Conference Proceedings).

[2] Ebert, D., Musgrave, F., Peachey, P., Perlin, K., and Worley, S., "Texturing and Modeling: A Procedural Approach, Third Edition", with contributions from W. Mark and J. Hart, Morgan Kaufman, November 2002

[3] M. F. Barnsley : "Fractals Everywhere", Academic Press, 1988.

[4] Barnsley, M.F., Jacquin, A., Malassenet, F., Reuter, L., Sloan, A.D., "Harnessing Chaos For Image Synthesis", SIGGRAPH(88), pp. 131-140.