2003－CG－112　(21)
2003／8／19

並列ビジュアルシミュレーションへの PC 用グラフィックスハードウェア活用に関する検討

村木　茂　　†劉　学振　　†片野　康生　　‡安藤　弥枝

産業技術総合研究所 ボリュームグラフィックス連携研究体
†三菱プレシジョン株式会社
‡科学技術振興事業団

s-muraki@aist.go.jp, liu@mpcnet.co.jp, katano@mpcnet.co.jp,　yae-andou@aist.go.jp

概要

大規模シミュレーションに，コストパフォーマンスの高い PC クラスタを使用し，計算と可視化を同時に実行しようとする試みがある．しかし，このスタイルでは，CPU による計算結果を GPU に転送する処理がボトルネックとなると考えられる．

本稿では，2 次元反応拡散系シミュレーションを例題に，最新 GPU とプログラム言語 Cg による，GPU によるビジュアルシミュレーションの実現法を検討する．

A study on utilizing PC graphics processor for parallel visual simulations

Shigeru Muraki　†Xuezhen Liu　　†Yasuo Katano　　‡Yae Andou

Collaborative Research Team of Volume Graphics, AIST
†Mitsubishi Precision, Co., Ltd.
‡Japan Science and Technology Corporation (JST)

s-muraki@aist.go.jp, liu@mpcnet.co.jp, katano@mpcnet.co.jp,　yae-andou@aist.go.jp

Abstract

Recently, there are some attempts to perform the computation and the visualization of large-scale simulations by employing low-cost PC clusters. However, in this style, the data transfer from the computation task to the visualization task seems to be a bottleneck for the high performance visual simulation.

In this paper, we discuss on how to use the latest GPU and new programming language Cg for visual simulations by using a few 2D reaction-diffusion simulations as the examples.

1．背景と目的

近年，生物物理シミュレーション[1]，医用3次元画像処理[2]，数値流体力学などの多くの分野において，3次元格子上での高速計算と可視化が必要とされている．従来，これらの処理は，計算はスーパーコンピュータ，可視化はグラフィックスワークステーションというスタイルで行われてきたが，最近はコストパフォーマンスの高いPCクラスタで，計算と可視化の両方を行おうとする研究が進められている[3,4]．特に，可視化処理では，PC用グラフィックスプロセッサ（GPU）の3次元ハードウェアテクスチャマッピング機能を利用したボリュームレンダリングと，ソートラスト型並列レンダリングを組み合わせることによって，比較的大きなサイズのボリュームレンダリングを対話的な速さで行えるようになっている[4,5]．しかし，PCクラスタで計算と可視化を同時に行おうとすると，CPUのメインメモリにある計算結果を，GPUのグラフィックスメモリのデータ形式に変換し，転送する必要が生じ，それがボトルネックとなり，対話的なビジュアルシミュレーションの実現を妨げている．

そのような背景から，最近のGPUの高いプログラム性を利用して，シミュレーション計算と可視化処理の両方をGPU内で行うことが検討されている[3, 6]．特に，最新のGPUでは，これまで固定小数点精度の演算に限られていた多くの処理で，浮動小数点数の使用が可能になり，特別な精度調整を行わなくても，シミュレーションプログラムが容易に記述できるようになったことから，今後はこうしたプログラミングスタイルが，シミュレーション分野で市民権を得ると予想される．

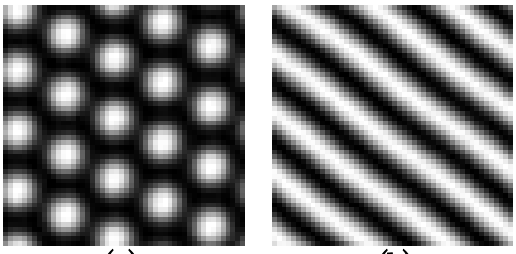
本稿では，医用画像処理や数値流体力学との関連性や，可視化対象としての面白さなどの理由から，2次元反応拡散系のビジュアルシミュレーションを題材とし，そのGPUプログラムの方法と問題点などを考察する．

2 反応拡散系シミュレーション

反応拡散系は，Turing によって提案された，

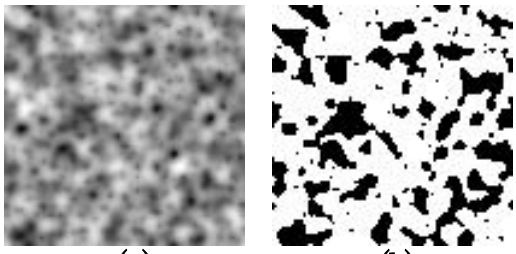
$$\begin{cases} \partial u/\partial t = D_u \nabla^2 u + f(u,v), \\ \partial v/\partial t = D_v \nabla^2 v + g(u,v), \end{cases} \qquad (1)$$

のような偏微分方程式で記述される系で[7]，生物の毛皮模様や形態形成のメカニズムとの関連が研究されている[8]．u, v は，2種類の化学因子の濃度で，それぞれ異なった速度 D_u, D_v で拡散する．$f(u,v)$，$g(u,v)$ は，それぞれの因子の相互作用項である．通常，拡散反応は，系を安定化するように働くと思われがちだが，条件によっては，$f(u,v)=g(u,v)=0$ から導かれる空間一様平衡解 (u_0,v_0) からの僅かな乱れを増幅し，自発的にパターンを発生するようになる（拡散誘導不安定性）[7,8,9]．代表的な反応拡散系として，以下のようなモデルが提案されている．



(a) (b)

図 1．Schnakenberg モデルによるパターン．



(a) (b)

図 2．Thomas モデルによるパターン．

■ ***Schnakenberg* モデル**[8]

$f(u,v)$，$g(u,v)$ が，

$$\begin{cases} f(u,v) = u^2 v + a - u, \\ g(u,v) = b - u^2 v, \end{cases} \qquad (2)$$

のような式で表され，反応が進むにつれて，a, bのわずかな違いにより，水玉模様や縞模様が現れる．空間一様平衡解は $(a+b, b(a+b)^{-2})$ である．図1(a)はa=0.1, b=1.1, D_u=1.0, D_v=20.0の場合，図1(b)は，b=1.41とした場合の収束パターンである．

■ ***Thomas* モデル**[8]

キリンや牛などの動物の毛皮模様のモデルとしてよく使われ，$f(u,v)$，$g(u,v)$ は，

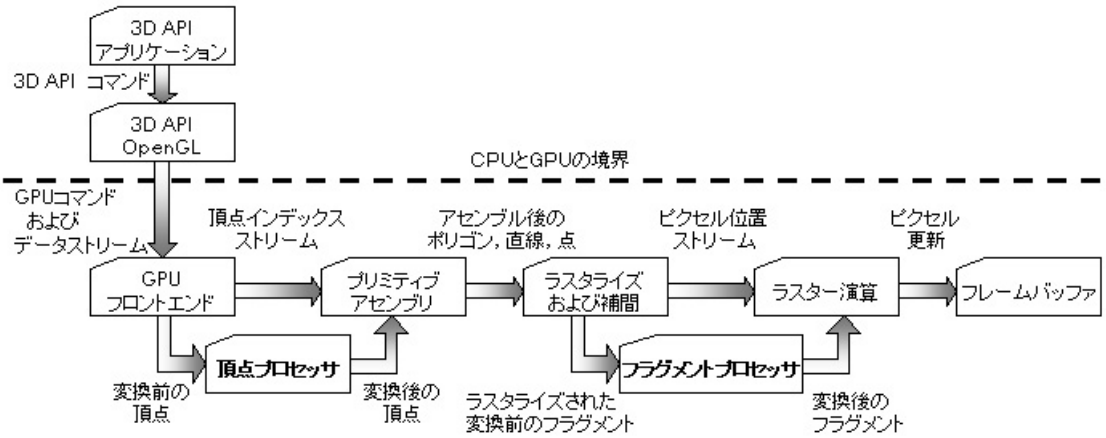


図 3． プログラマブルグラフィックスパイプライン．

$$\begin{cases} f(u,v) = a - u - h(u,v), \\ g(u,v) = \alpha(b - v) - h(u,v), \\ h(u,v) = \dfrac{\rho uv}{1 + u + Ku^2}, \end{cases}$$

のように与えられる．図 2(a)は，a=103, b=77, α=1.5, K=0.125, ρ=13 での収束パターン，図 2(b)はその 2 値化画像である．このパラメタにおける空間一様平衡解は(25.0, 25.0)である[1].

■ ***Gray-Scott* モデル[9]**

このモデルは，先の二つのモデルのように静止パターンに収束するのではなく，島状のパターンが細胞分裂のように増殖，崩壊を繰り返す．$f(u,v)$，$g(u,v)$ は，

$$\begin{cases} f(u,v) = -uv^2 + F(1-u), \\ g(u,v) = uv^2 - (F+k)v, \end{cases}$$

で与えられる．

反応拡散シミュレーションの実際の計算法は，(u_0, v_0) を初期値として，式(1)を計算しながら，オイラー積分，

$$\begin{aligned} u_{T+1} &= u_T + \frac{\partial u_T}{\partial t}\Delta T, \\ v_{T+1} &= v_T + \frac{\partial v_T}{\partial t}\Delta T, \end{aligned} \qquad (3)$$

を繰り返すことで実現される．

我々はこれまで，独自に開発した計算可視化システム，VG クラスタ[10]を使って，3 次元反応拡散系シミュレーションとその可視化を対話的に実現する研究を行ってきた[4, 11]．VG クラスタシステムは，画面重畳装置を階層的に接続することにより，原理的には任意サイズのソートラスト型並列ボリュームレンダリングを実時間で実行できる．各 PC は，分割されたサブボリュームの可視化を，GPU（nVIDIA 製 GeForce 4 Ti 4600）の 3 次元テクスチャーマッピング機能を使ったボリュームレンダリングで実行するため，可視化に関してはリアルタイム処理が可能である．しかし，浮動小数点演算の必要な反応拡散系シミュレーション部分は CPU で処理していたため，単位時間ステップ（ΔT）ごとに，式(3)のオイラー積分を実行し，結果をメインメモリ上で 8 ビットに整数化し，グラフィックアクセラレータの 3 次元テクスチャメモリにロードする必要があり，この CPU-GPU 間転送が，効率の良いビジュアルシミュレーション実現のボトルネックとなっていた．

しかし最新の GPU では，テクスチャに浮動小数点数が使用できるようになるなど，プログラム性が大幅に向上している．そのため，反応拡散系シミュレーションを GPU で実現することにより，処理の高速化とボトルネック解消が期待できる．そこで以下では，そのための予備研究として，2 次元の反応拡散系シミュレーションを例に，GPU によるシミュレーション計算の方法について検討する．

[1] 空間一様平衡解を求めるには 3 次方程式を解く必要があり，かなり面倒である．

```
struct RDv2f : vertex2fragment
{
  float4 hPosition  : HPOS;
  float4 texCoord   : TEX0;
};

fragout main(RDv2f IN,
             uniform samplerRECT Chemical,       // 化学因子場 (u,v)
             uniform float2      D,              // 拡散係数 (Du, Dv)
             uniform float2      params,         // パラメタ (a, b)
             uniform float2      TextureSize,
             uniform float       TimeStep )
{
  fragout OUT;

  float2 uv = IN.texCoord.xy;
  float2 sample = f2texRECT(Chemical, uv);

  // 拡散項（周期境界条件）
  uv.x = (IN.texCoord.x > TextureSize.x - 1.0f) ? 0.5f  : IN.texCoord.x + 1.0f;
  float2 diffusion = f2texRECT(Chemical, uv);
  uv.x = (IN.texCoord.x <= 1.0f) ? TextureSize.x - 0.5f : IN.texCoord.x - 1.0f;
  diffusion += f2texRECT(Chemical, uv);
  uv = IN.texCoord.xy;
  uv.y = (IN.texCoord.y <= 1.0f) ? TextureSize.y - 0.5f : IN.texCoord.y - 1.0f;
  diffusion += f2texRECT(Chemical, uv);
  uv.y = (IN.texCoord.y > TextureSize.y - 1.0f) ? 0.5f  : IN.texCoord.y + 1.0f;
  diffusion += f2texRECT(Chemical, uv);
  diffusion -= 4.0f * sample;
  diffusion *= D.xy;

  // 相互作用項
  float2 reaction = sample.xx * sample.xx * sample.yy;
  reaction.y *= -1.0f;
  reaction += params;
  reaction.x -= sample.x;

  // オイラー積分
  sample += (reaction + diffusion) * TimeStep;
  OUT.col.xy = sample;
  return OUT;
}
```

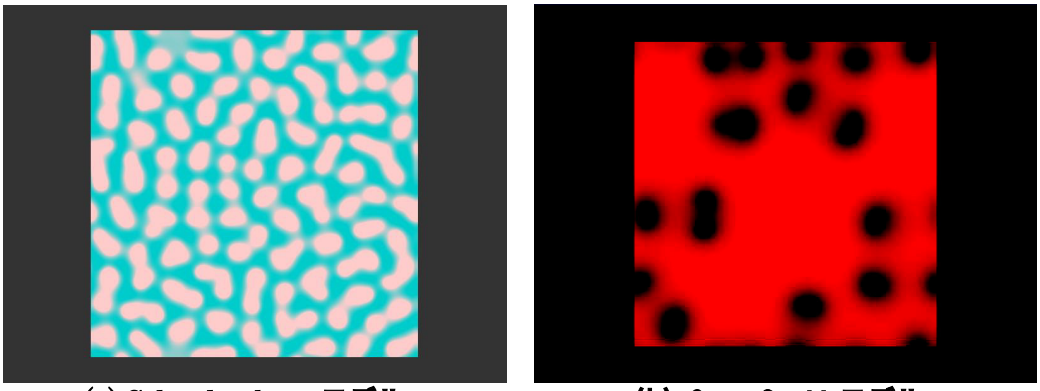
図 3 Cg による 2 次元反応拡散系シミュレーションのフラグメントプログラム

3 GPU による 2 次元反応拡散計算

3.1 ＧＰＵプログラミング

我々の VG クラスタシステムでは, OpenGL の拡張命令を使って nVIDIA 製 GeForce 4 Ti 4600 のレジスタコンビナをプログラムし，テクスチャベースのボリュームレンダリングを行っていたが[4]，反応拡散系シミュレーションのような複雑な偏微分方程式をこの方法で解くことは困難であり，また，浮動小数点数を使うことができず計算精度にも問題があった．

nVIDIA 社の最新 GPU, GeForce FX では，浮動小数点演算機能のサポートを含む大幅な機能拡張がなされた．nVIDIA 社では，この新しい GPU のプログラミングのために，Cg という，C に似た言語を用意している[12]．図 3 は，プログラマブル GPU のパイプライン処理を示している．これらの処理のうち，頂点プロセッサとフラグメントプロセッサのプログラムに Cg が使われ，それぞれ頂点プログラム (*vertex program*)，フラグメントプログラム (*fragment program*) と呼ばれる．



(a) Schnakenberg モデル　　(b) Gray-Scott モデル

図 4 GPU プログラミングによりポリゴン上に発生させたパターン.

頂点プログラムは頂点ごとに実行され, 主に座標変換を行う. 頂点プログラムの入力は, 通常, モデル空間位置座標, モデル空間法線ベクトル, テクスチャ座標などだが, Cg を使えば, ユーザーが屈折率などの値を頂点に提供するアプリケーションを容易に記述することができる.

頂点プログラムの出力は, ラスタライザーを通過して, フラグメントプログラムの入力となる. フラグメントプロセッサは, 頂点プロセッサと同様の処理の他, テクスチャ演算が可能である. フラグメントプログラムは, テクスチャ座標の組を使ってテクスチャ画像にアクセスし, フィルター処理を加えたテクスチャ画像を返すことができる. したがって, 2 次元反応拡散系シミュレーションを GPU で実行するには, 浮動小数点数テクスチャ画像の 2 チャンネルに 2 種類の化学因子濃度を格納し, テクスチャの画素値から ΔT 後の濃度値を計算するフラグメントプログラムを用意し, テクスチャ画像をフレームバッファに描画し, それを OpenGL の **glCopyTexSubImage2D()**命令により, 再度テクスチャ画像に書き戻す処理を繰り返せばよい[3].

3.2 反応拡散系のフラグメントプログラム

図 3 は, Cg で記述した Schnakenberg モデルの 2 次元反応拡散系シミュレーションを行うフラグメントプログラムの例である.

Chemical は, 2 次元格子上に化学因子の濃度を格納する浮動小数点数テクスチャ画像で, プログラムは現在のピクセルを 2 次元ベクトル sample に読み出し, さらに 4 近傍の値を使った差分近似で式(1)の右辺第 1 項を計算し, 2 次元ベクトル diffuse に格納する. 次に式(1)の右辺の残りの項が 2 次元ベクトル reaction に計算され, オイラー積分により ΔT 後の濃度値を出力する. 図 3 のプログラムは読みやすさを優先して, 式(1)のラプラシアンをそのまま計算しているが, 式(1)を全体的に変形することで, さらにプログラムのステップ数を減らすことも可能である[3][2].

4 実験

図 3 の Schnakenberg モデルの他, Thomas モデル, Gray-Scott モデルを加えた 3 つのプログラムを作り動作を調べた.

使用した GPU は, nVIDIA 製 GeForce FX 5900 Ultra で, CPU が Intel 製 Pentium 4 (2.4GHz)で, 8 倍速の AGP バスを有する PC に挿入された. ソフトウェアは, nVIDIA 社のホームページからダウンロードした Cg Toolkit (Release 1.1)を使用して, Microsoft 製 Windows XP 上で, Microsoft 製 Visual C++ 6.0 を使って作成した.

図 4 は, 本プログラムにより発生した, Schnakenberg モデルのパターンである. ここで式 (2), (3) のパラメタを, a=0.1, b=1.16, D_u=1.0, D_v=20.0, ΔT=0.01 とした. 化学因子濃度のテクスチャサイズは 128×128 とし, 30 回の更新ごとに 640×480 画素のウィンド上にパターンを表示させた. パターンはプログラム機動とともに一瞬で現れ, 10 秒以内で視覚上完全に収束した. 更新 1 回あたりの計算時間は, 約 1.0 ミリ秒であった. 比較のため, 同じ計算

[2] 図 3 のプログラムの作成には, nVIDIA 社の Greg Janes, ノースカロライナ大学チャペルヒル校の Mark Harris らによるデモプログラム, "Disease" (http://www.cs.unc.edu/~harrism/gdc2003/) を参考にした.

機上で, 完全にソフトウェアで計算する反応拡散シミュレーションプログラムを使用した場合は, 更新 1 回あたり 1.1 ミリ秒で, GPU を使った場合とほぼ同じであった．しかし，GPU を使用した場合, パラメタの範囲によって正常に動作しなくなる問題がみられた. Thomas モデルも，今のところ GPU では正常に動作していない．この原因は，今現在わかっていない．

図 4(b)は GPU で計算を行った Gray-Scott モデルの発生パターンである．更新 20 回あたりに 1 回画面表示を行うと, Schnakenberg モデルと異なり, 実時間で島状のパターンが発生, 消滅を繰り返すダイナミックな映像が，約 30 フレーム/秒で表示できた．しかし，Schnakenberg モデルと同様で, パラメタ変更時の動作は不安定であった.

5 まとめと今後の課題

浮動小数点演算をサポートする GPU と，プログラミング言語 Cg を使用して，3 種類の 2 次元反応拡散系のビジュアルシミュレーションを行ってみた．その結果，Schnakenberg モデル，Gray-Scott モデルにおいて，CPU と同程度以上の高速計算と, 実時間の可視化が実現できた．しかし，GPU も Cg も，本来，映像生成のために設計されており, まだ発展途上であるため, 浮動小数点演算がサポートされたとは言え, 反応拡散系のような偏微分方程式系のシミュレーションに使用するには, まだ機能面でも, 安定性でも, 不十分と言わざるをえない. 今回の実験でも，挙動不明な点が多々見られ, 今後の性能向上に期待したい.

我々が比較に用いた GPU を用いない反応拡散系シミュレーションプログラムは計算経過を表示しないので, 結果を可視化するためには CPU から GPU へのデータ転送が必要になる. これを考慮すれば, データ転送を必要としない GPU によるシミュレーション計算は，実時間ビジュアルシミュレーションの実現に非常に有効と考えられる. 今後は，シミュレーションの 3 次元化と， VG クラスタシステムを使った，GPU による大規模並列ビジュアルシミュレーションの実験を行う予定である.

参考文献

[1] 本多久夫, 生物の形づくりの数理と物理, 共立出版，2000 年 4 月.

[2] 安藤祥子，村木 茂，藤代一成，拡散シミュレーションを利用した MR 拡散テンソル場並列可視化法の検討，情報処理学会 第 112 回グラフィックスと CAD 研究会，2003 年 8 月.

[3] M. J. Harris, G. Coombe, T. Scheuermann, A. Lastra, Physically-Based Visual Simulation on Graphics Hardware, *Proc. 2002 SIGGRAPH / Eurographics Workshop on Graphics Hardware* 2002, September 2002.

[4] 村木茂, 高並列ボリューメトリックシミュレーションのための空間分割法に関する一考察, 情報処理学会 第 111 回グラフィックスと CAD 研究会, 2003 年 5 月.

[5] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, Chromium: A Stream-Processing Framework for Interactive Rendering on Clusters, *ACM Trans. Graphics (Proc. SIGGRAPH 2002)*, Vol. 21, No. 3, pp.693-702, July 2002.

[6] W. Li, X. Wei, A. Kaufman, Implementing Lattice Boltzmann Computation on Graphics Hardware. To appear in *The Visual Computer.*

[7] M. Turing, The chemical basis of morphogenesis, *Phil. Trans. Roy. Soc.*, B237, pp. 37-72, 1952.

[8] J. D. Murray *Mathematical Biology II: Spatial Models and Biomedical Applications 3rd Ed.*, Springer, 2003.

[9] 西浦廉政, 自己複製と自己崩壊のパターンダイナミクス，岩波書店，2003 年 2 月.

[10] S. Muraki, M. Ogata, K. Kajihara, K-L. Ma, K. Koshizuka, X. Liu, Y. Nagano, K. Shimokawa, Next-Generation Visual Supercomputing using PC Clusters with Volume Graphics Hardware Devices, *Proc. IEEE SC2001*, October 2001.

[11] S. Muraki, E. B. Lum, K-L. Ma, M. Ogata, X. Liu, A PC Cluster System for Simultaneous Interactive Volumetric Modeling and Visualization, To appear in IEEE Symposium on Parallel and Large-Data Visualization and Graphics (PVG2003), October 2003.

[12] R. Fernando, M. J. Kilgard, *The Cg Tutorial*, Addison Wesley, February 2003.