社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－CG－133 (1)
2008／11／7

GPUによる形状生成に基づく樹木アニメーションの高速化

赤木 康宏 北嶋克寛
東京農工大学 情報工学科

風などの外力の影響を考慮した樹木アニメーションを生成する場合，枝のしなり及び葉の向きの変化をフレーム毎に求めることで，自然な動きの生成が可能となる．その際，自然物のもつ滑らかな形状を保つために，ほとんどの形状（頂点）情報を再計算する必要があり，映像生成時間の増大を招いている．本稿では，GPU上での形状生成に適した樹木形状モデルを提案し，アニメーション生成の高速化を図る．本手法の特徴は，メインメモリ上にはポリゴンデータをもたず，枝及び葉の情報（位置，姿勢，サイズ）をGPUへ転送することで，GPU内でフレーム毎に形状を再生成する点にある．提案手法をGPU上に実装し，各種実験により高速化の効果を検証する．

An acceleration method for generating tree animations by using a geometry generator on GPU.

Yasuhiro AKAGI Katsuhiro KITAJIMA
Tokyo University of Agriculture and Technology

For generating a tree animation that has realistic movement in the wind, it is required to computes deformations of branches and positions of leaves in each frame. To represent realistic shapes of trees ,the computation time of generating a animation is increased by deforming prosses of trees. In this paper, we propose a suitable tree model for geometry generating on GPU. The most feature of this method is that the main memory do not have the polygon data of a tree. By registering the position and the size of branches and leaves on GPU, our method can dynamically generate the polygon data. We implemente this method on GPU and verify a acceleration rate between GPU and CPU.

1. はじめに

自然景観を対象とするCGを制作する場合，樹木はその形状の複雑さに加え，風などの外力によって柔軟に変形するという特徴から，重要なオブジェクトの1つとして扱われている．樹木形状の生成及びアニメーション表示に関する研究はこれまでも数多くの行われており[1]，近年では実写画像との合成を行っても違和感のない程の自然な映像生成が可能となっている[2]．しかし，映画等で用いられるような高品質な映像を制作しようとする場合，樹木形状を表現するためのポリゴンデータが膨大となり，レンダリングに要する時間が増大するという問題が生じる．さらに，制作者が最終的な映像を短時間のうちに確認できないので，思い通りのシーンを制作するまでに，膨大な時間をかけて試行錯誤を繰り返し行う必要も生じ，制作コストの増大にもつながる．

樹木アニメーションの生成処理は，機械及び建築物等の人工物とは異なり，形状そのものに柔軟性があり，風などの外力により複雑な曲がりを表現する必要があるという特徴がある．アニメーションを生成する際には，フレーム毎にほとんどの頂点位置を再計算する必要があり，多くの計算時間がこの処理に費やされる．そこで，本稿では，Graphics Prossesing Unit (GPU)を用いて形状生成の高速化を図る．GPUを用いたアニメーションに関する研究としては，髪の毛の動きを実時間で描画する手法[5][6]，及びポリゴンメッシュの細分割をGPU上で動的に行う研究[7]などがある．GPUは，頂点及び画素等の独立したベクトルデータ等に対

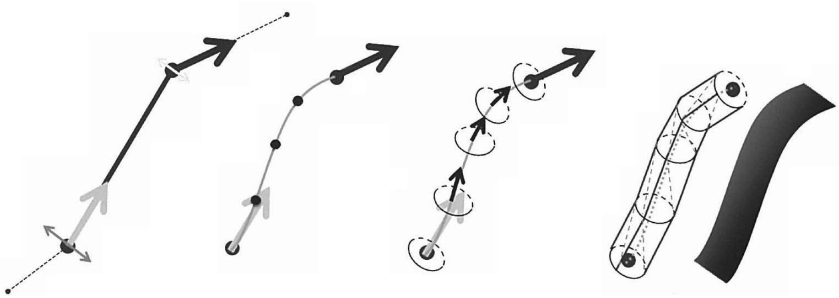

図２枝形状生成の手順

する演算に特化しており，非常に高い並列性を有しているという特徴がある．本稿で提案する手法では，枝及び葉を独立性の高いセグメント単位に分割して扱い，樹木形状の生成に要する最小限の情報を GPU へ送ることで，GPU 上で効率的にポリゴンデータの生成を行う．

2. 並列性を考慮した樹木モデル及び揺れの生成手法

本章では，本手法で用いる樹木モデル及びそのモデルに基づく揺れの生成手法について述べる．

2.1 樹木モデル

樹木は１本の幹から複数回の分枝を繰り返すことで，特有の形状を構成している．CG に用いられる樹木モデルでは，表面の幾何情報のみをもつものと，枝を骨格に相当するような，芯及び関節（リンク）をもつ複数の枝セグメントをつなぎ合わせることで表現したもの（図１・左）の２種類に大別できる[3][4]．前者は静的な表示に用いる限りは十分であるが，外力による揺れ，すなわち枝や葉の動的な変形を伴う場合では，変形が困難である．一方，後者は芯と関節を用いて曲げ計算を単純化し，変形計算を容易に行うことができるので，本手法では後者のモデルを用いる．樹木モデルが保持する具体的な情報を次に示す．

・芯の長さ及び太さ（葉であるならば長さ及び幅）
・親セグメントへのリンク（１個）
・子セグメントへのリンク（複数個）
・親セグメントに対する向きの変位

なお，向きの変位については Quaternion を用いて表現する．また，葉は枝セグメントの末端に配置する．

本樹木モデルから形状（ポリゴンデータ）を生成する際には，枝では芯の周囲に太さに応じた円筒形状を配置し，葉であれば向きとサイズに応じた楕円形状を配置する．形状生成の詳細については次節で詳しく述べる．

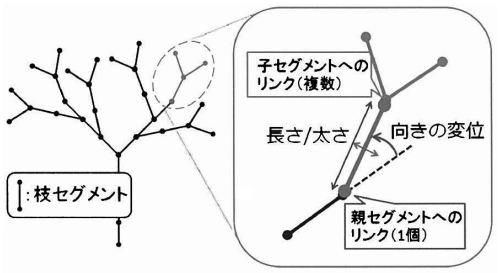

図１ 枝セグメント構造と情報

2.2 樹木モデルからの枝形状生成

自然界の枝形状は曲がりのある円筒状であり，表面に様々な凹凸が見られる．１つの枝セグメントに注目した場合，根元側（始点）と先端側（終点）では太さや向き（断面の法線方向）が異なっているので，生成される形状は滑らかな曲がり及び隣接するセグメントとの連続性をもつことが求められる．そこで，次の手順により形状の生成を行う（図２）．

(1) １つのセグメントの始点と終点の位置，方向及び太さを与える．
(2) 始点と終電を滑らかにつなぐ曲線（中心線）を Cubic Hermite Spline 曲線を用いて表現し，一定間隔ごとに通過点を求める．（図２では３箇所）
(3) 各通過点での断面生成を行う．断面の法線を Quaternion による球面線形補間（slerp），半径を線形補間により求める．

(4) 各断面同士を三角形ポリゴンを用いてつなぎ合わせることで，枝形状ポリゴンを生成する．

(2)で用いる Cubic Hermite Spline 曲線は３次の Spline 曲線の１つであり，始点，終点の位置及び接ベクトルを与えることで曲線が求まる（式１）．

$$p(t) = (1+2t)(1-t)^2 P_1 + t^2(1-t)N_2 + t^2(3-2t)P_2 + t(1-t)^2 N_1 \quad \cdots (1)$$

式１は，４個の３次元ベクトルの線形結合のみで曲線を求めることができるので，処理が単純であり，GPU 上の実装に適している．

(3)で生成する断面の向きに関しては，Quaternion を用いた球面線形補間を用いる（式２）．枝の向きは(2)で求めた中心線を微分することによっても得られるが，

$$\theta = \cos^{-1}(Q_1 \cdot Q_2)$$
$$Q(t) = Q_1 \sin((1-t)\theta) - Q_2 \sin(t\theta) \quad \cdots (2)$$

進行方向のベクトルのみでは，枝のねじれ（進行方向の回転量）を扱うことが困難であるので，Quaternion を用いた．また，断面形状を構成する頂点データは単位円を近似した多角形（本稿では８角形）としてあらかじめ用意することで高速化を図る．断面形状を(2)で求めた位置へ平行移動，及び(3)で求めた姿勢へ回転し，個々の断面を構成する頂点の座標を求める．

2.3 枝形状の高品質化

2.2 節で述べた手法により生成した枝形状は，表面に凹凸のない円筒であり，幹等の太い枝では不自然さがある．そこで，枝形状を構成する三角形のうち，サイズの大きなものを対象にディスプレイスメントマッピングによる凹凸の付加を行う．凹凸の付加はまず，対象となる三角形ポリゴンの各辺の中点に新たな頂点を生成し，ポリゴンの細分割を行う（図３・中央）．次に，頂点をディスプレイスメントマップの情報を基に，面の法線方向へ移動させる（図３・右）．また，レンダリングに時には枝表面の微細な凹凸を表現するために，法線マップを用いる．

2.4 樹木モデルからの葉形状生成

樹木の葉形状は，葉脈を軸とする平面的な広がりをもつ面及び，その縁に微細な凹凸や裂け目をもつもの

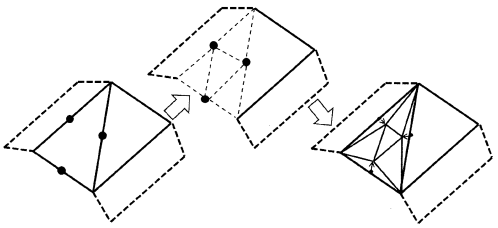

図３ 枝ポリゴンの細分割

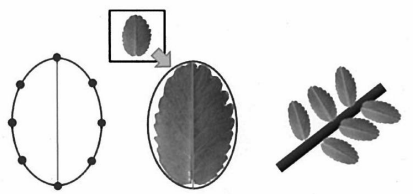

図４ 葉の構造とテクスチャ

が多く，細部の形状は複雑である．さらに，１本の樹木には数千枚以上の葉が存在しており，CG アニメーションを生成する際に，個々の葉を緻密に表現することは困難である．そこで，葉形状を２つの半円を組み合わせた形状で近似し，詳細形状についてはテクスチャマッピングにより補う（図４）ことで，質感の維持しつつポリゴン数を抑える方法を用いる．

樹木全体を描画する際には，葉形状をあらかじめ１つ用意しておき，葉を描画すべき位置（末端の枝側面）へ平行移動及び回転により配置する．

2.5 揺れの生成

本節では，2.1 節で述べた樹木形状モデルに対して，外力による揺れを生成する手法についてその手順を次に示す．

（１） 風のシミュレーション等によって，各セグメントの受ける外力を求める．

（２） 個々のセグメントについて，子セグメントの受ける外力を合計（力を伝搬）し，枝の向きに対して垂直な成分を枝の回転力とする（図５・左）．

（３） セグメントの根元（親セグメントとの接合部）を中心とし，枝の剛性を回転に対する抵抗力として与えることで，セグメントの回転量を計算する（図５・右）．

（４） 全セグメントの回転量が求まった後に，根元からセグメントの回転（形状の変形）を行い，形状を更新する．

上記の処理を繰り返し行うことで，アニメーションを生成する．

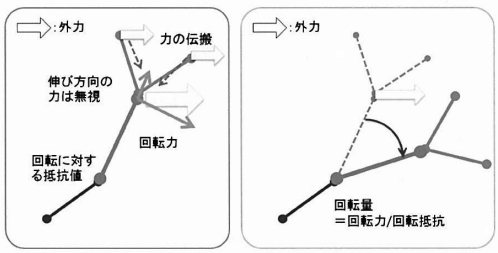

図5 外力による枝の曲げ

上記の処理は，力の伝搬及び幹を基準としたセグメントの回転を含み，枝の再帰性（親セグメントへの従属性）に基づくという特徴がある．また，ポリゴンデータの生成処理に比べ，各セグメントの回転個所は限られており計算量は少ないという特徴もある．よって，GPU は用いず，CPU による再帰処理により計算を行う．

3. GPU への実装

本章では第２章で述べた樹木形状モデルに基づく形状生成を GPU 上で実行するための手法について述べる．

3.1 GPU を用いたポリゴン生成の概要

本節では，読者の GPU 上での処理に関する理解を深めるために，GPU の構造及び GPU を用いたポリゴン生成の概要について述べる．GPU では，レンダリング処理を処理の特徴に応じて複数のステージに分解し，パイプライン化することで並列化の効果を高めている．ステージの中にはアプリケーション側から実行時に処理内容を書き換えることのできる，Programmable Shader と呼ばれるステージが複数ある．本稿では，Shader Model 4.0 における 3 種類の Shader ステージの１つである，Geometry Shader（以下 GS）を主に利用し GPU によるポリゴン生成を行う．GS はプリミティブ形状（点・線分・三角形等）を受け取り，これらの情報を利用し新たな頂点生成を行う処理を担当する．例えば，与えられた三角形を細分割し滑らかな曲面を生成する処理の高速化に用いられる．さらに，１点から数十個の三角形を生成するなど，柔軟な形状生成も可能である．GS によって生成されたプリミティブ形状はそのまま描画することも可能であるが，

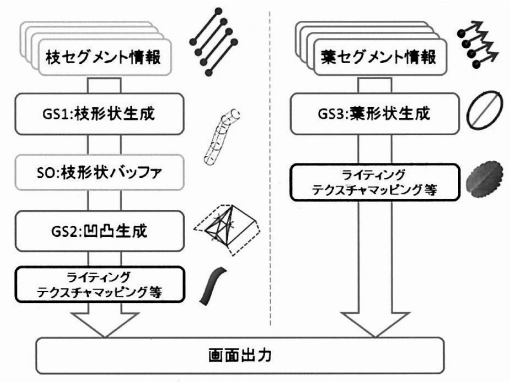

図6 GPU を用いた樹木形状の生成処理

Stream Output（SO）機構を通してバッファ（GPU 上のメモリ）へ格納し，他の描画処理へ利用することもできる． 本稿において実装した GPU 上での樹木描画処理の流れを図６に示す．図６では 3 種類の GS（GS1~3）を用いているが，それぞれの具体的な実装方法については次節以降で述べる．

3.2 GS を用いた枝形状生成処理の実装

2.2 節で述べた枝形状の生成手法に基づいた処理を，GS 上で実装する方法について述べる．枝形状を生成するためには，各枝セグメントの端点における４つの情報（位置，方向，太さ，長さ）が必要である．そこで，CPU 側からは端点を頂点，枝セグメントを線分プリミティブ（両端点の番号をもつ）として必要な情報を GPU へ転送する．データフォーマットを次に示す．

端点 { 位置（4 次元同次ベクトル）
　　向き（Quaternion）
　　太さ（浮動小数値），長さ（浮動小数値）}
枝セグメント { 始点番号（整数値）
　　終点番号（整数値）}

上記のフォーマットに対応した頂点バッファ及びインデックスバッファを生成し，各フレーム毎に端点情報の更新を行う．

枝形状生成用の GS（GS1）では両端点の情報から 2.2 節及び図２で示した手順によりポリゴンを生成する． その際に必要となる Quaternion 及び Cubic Hermite Spline に関する処理はすべて，GPU 上において実装を行った．

GS1 において枝形状を生成する際には，高速化のために，あらかじめ単位円（8 角形）のデータを GPU

上の定数バッファに格納しておき，これを中心線に沿って平行移動及び Quaternion による回転を行うことで，断面を構成する頂点の最終的な座標を計算する．このとき，テクスチャマッピング用の座標も合わせて計算を行う．求めた頂点同士をつなぐ三角形ポリゴンを構成し，次節で述べる表面の凹凸生成処理のために SO を通して GPU 上の枝形状バッファへ格納する．

3.3 枝形状への凹凸の付加

2.3 節で述べた手法に基づき，枝形状を構成するポリゴンに対する細分割及びディスプレイスメントマッピングによる凹凸生成を行う．まず，3.1 節で述べた処理により生成した枝形状バッファに格納されている三角形情報を，3.2 節とは異なる凹凸生成用の GS（GS2）へ送る．GS2 では，あらかじめ定められた閾値を用いて，サイズの大きなものを対象に細分割及び凹凸の付加を行い，新たな三角形ポリゴンを生成する．閾値以下のサイズの三角形は細分割せず，そのまま出力を行う．GS2 で生成したポリゴンデータは Pixel Shader においてライティング及びテクスチャマッピングを行った後に，画面へ出力される．

3.4 葉形状の生成

2.4 節で述べた描画手法に用いる葉のデータは４つの情報（位置，方向，幅，長さ）をもつ１つの点として表現できる．そこで，GPU へ転送するデータフォーマットは次のように定義した．

葉｛　位置（4 次元同次ベクトル）
　　向き（Quaternion）
　　幅（浮動小数値），長さ（浮動小数値）｝

個々の葉は独立であるので，この情報を点群プリミティブとして，葉形状生成用の GS（GS3）へ入力する．GS3 では，枝の断面形状生成と同様に，あらかじめ単位円（8 角形）のデータを GPU 上の定数バッファに格納しておき，これを平行移動及び回転，拡大縮小し葉形状のポリゴンデータを生成する．その後，枝形状と同様に Pixel Shader での処理を経て，画面に出力される．

4.実験

本章では，本稿で提案する高速化の効果を検証するために， CPU 及び GPU の双方に同様な樹木形状生成処理を実装し，アニメーション生成速度の比較を行う．実行環境を表 1 に示す．

表１ 実験環境

CPU	Core2 Extreme QX9650
RAM	8GByte
GPU	GeForce GTX280
OS	Windows Vista(x64)

4.1 樹木形状の生成結果

図７に GPU を用いた樹木形状の生成結果，図８に入力データからポリゴンを生成せずに描画した枝セグメントの状態を示す．

図７　GPU を用いた樹木の描画結果

図８　枝セグメントの描画結果

これらの結果から，枝セグメントの情報に応じた樹木形状が GPU を用いて生成されていることが確認できた．

4.2 形状生成速度の比較

GPU を用いたポリゴン生成による高速化の効果を検証するために，同等の処理を CPU 上において実装し，処理時間の比較を行う．CPU 側の処理時間は，外力による変形後の独立性のあるセグメントの情報から①ポリゴンを生成しメモリに格納する処理及び②生成した形状をレンダリングする処理の２つの合計時間と

表２　GPU 及び CPU での樹木形状生成時間の比較

ポリゴン数	CPU による計算時間（秒）			GPU による計算時間（秒）	時間比 CPU/GPU
	①生成	②描画	合計	（生成+描画）	
108, 808	1. 27	0. 004	1. 27	0. 006	200. 5
201, 760	2. 58	0. 008	2. 58	0. 013	201. 7
450, 592	6. 51	0. 016	6. 52	0. 032	205. 9
823, 840	12. 42	0. 025	12. 45	0. 064	193. 9
1, 321, 504	20. 34	0. 039	20. 38	0. 108	189. 3
2, 536, 192	39. 73	0. 056	39. 79	0. 196	202. 8
4, 109, 056	65. 20	0. 080	65. 28	0. 352	185. 2

した（データ転送に要する時間は除外した）．CPU 側で行うポリゴン生成処理は並列化せず，シングルスレッドにより実行した．樹木のポリゴン数を 10 万～400 万の範囲で変化させ，1000 フレーム分の描画にかかる時間を計測し，それぞれの比較を行った結果を表２に，ポリゴン数と時間比のグラフを図９に示す．この結果から，GPU を用いたポリゴン生成を行った場合では，CPU を用いた方法に比べ約 200 倍高速化されていることが分かる．一方，ポリゴン数の増加に伴い若干の効率低下も見られる．本実験環境のハードウェアでは，形状出力用のバッファサイズの上限から，約 500 万ポリゴンの樹木が 1 回の描画パスで処理できる限界であり，上限値付近での処理効率低下を引き起こしていると考える．効率の低下は 1 割以下であるので，ポリゴン数の増加によらず一定の高速化の効果が得られているといえる．

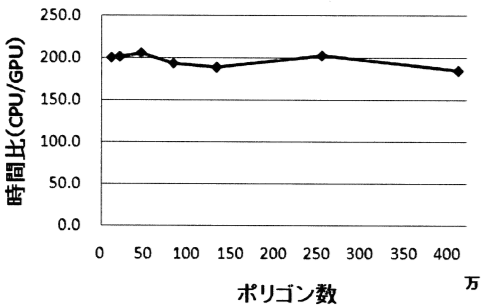

図９　アニメーション生成時間の比率

5.おわりに

本稿では，GPU を用いた樹木ポリゴン生成の高速化手法を提案した．Geometry Shader を用いた形状生成に適した並列性の高い樹木形状モデルを提案することで，効率的な樹木形状生成を実現した．実験により，GPU を用いることでポリゴン生成速度を約 200 倍に高速化できることが確認できた．

参考文献

[1] S. Ota, T. Fujimoto, M. Tamura, K. Muraoka, K. Fujita, and N. Chiba, : 1/fβ Noise-Based Real-Time Animation of Trees Swaying in Wind Fields, CGI2003, pp.52-59, 2003.

[2] J. Beaudoin and J. Keyser, : Simulation levels of detail for plant motion, Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on Computer animation, pp.297-304, 2004.

[3] 赤木康宏，佐波晶，北嶋克寛，物理シミュレーションに基づく風に揺れる樹木のアニメーション，情報処理学会論文誌（D），Vol.46，No.7，pp. 1797-1809，2005.

[4] 神田仁，大谷淳: 効率的かつリアルな 3 次元樹木モデルのアニメーションの検討，電子情報通信学会技術研究報告，Vol.102，No.660，pp.81-86，2003

[5] Masaki Oshita, : Real-time Hair Simulation on GPU with a Dynamic Wisp Model, Computer Animation and Virtual Worlds, Vol. 18, Issue 4, pp.583-593, 2007.

[6] E. Sintorn and U. Assarsson, : Real-time approximate sorting for self shadowing and transparency in hair rendering, In Proceedings of the 2008 Symposium on interactive 3D Graphics and Games (SI3D '08), pp. 157-162, 2008.

[7] Haik Lorenz, Jürgen Döllner : Dynamic Mesh Refinement on GPU using Geometry Shaders, Proceedings of the 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 2008, 2008.