2002 年度　卒　業　論　文

リアルタイム 3DCG における
個体差を伴う類似形状の
高速な生成および描画に関する研究

指導教員：渡辺　大地　講師

メディア学部 3DCG アプリケーション構築プロジェクト
学籍番号　99p218
首藤　年人

２００３年３月

2002 年度　卒 業 論 文 概 要

論文題目

リアルタイム 3DCG における個体差を伴う類似形状の
高速な生成および描画に関する研究

メディア学部 **学籍番号:** 99p218	**氏名**	首藤　年人	**主査**	渡辺　大地　講師
			副査	和田　篤

キーワード	リアルタイム、個体差、軸変形、3DCG、レンダリング

日常見かける風景を３DCG でリアルタイムに表現しようとする場合に、舞い落ちる木の葉や川辺の石ころなどの少しずつ形の異なった大量の形状を描画する場合がある。こういった個体差を持った形状をリアルタイムで描画するにあたり、個体差を持った形状をすべて読み込み描画する方法がある。しかしこの方法では描画する数の形状データ全てを読み込む必要があるため、データファイルが大きくなる。そこで、本研究では個体差を持った形状を大量に作るときにひとつの形状データと、描画する数の変形パラメータを持たせることで描画に使うデータファイルを少なくする手法を提案する。ファイルが小さくなれば保存媒体でのデータ占有率が少なくすむ、ファイルの入出力が迅速に出来る、ネットワークを介した３DCG の利用に便利になる、などの利点が生まれる。本手法では基となる形状にパラメータを与えるとパラメータにあった変形をした形状を作り出せる。変形された形状はその変形に使ったパラメータとして保存でき、基となる形状データと変形のパラメータがあれば復元ができる。

目次

第1章　はじめに

1.1　研究背景と目的

近年、映画やテレビ、CM、ゲーム、インターネット、医療、建築などの様々な分野で三次元コンピュータグラフィックス（以下３DCG）が利用されており、３DCG の技術は急速な発展を遂げている。かつて３DCG は高価なハードウェアやそれらを扱うための高度な技術が必要であったため、一つの作品を作成するために、たいへん多くの労力・時間・費用を必要としていた。しかし、各種ハードウェア及びソフトウエアの高性能化と低価格化に伴い、廉価なパーソナルコンピュータ上においても見ごたえのある画像を身近な環境でリアルタイムに、そしてインタラクティブに楽しむことができるようになった。例えば最近では家庭におけるコンピュータゲーム上で、３次元空間上にポリゴンで表現されたモデルをプレイヤーがリアルタイムに操作することが可能となり、Web 上の 3 次元空間で気軽に３D モデルを閲覧・操作することが可能になるなど様々な用途に使われている。そして機種に依存しないグラフィックスライブラリも登場し[11]、ネットワークを介した CG の利用も始まっている[12]。

日常見かける風景を３DCG でリアルタイムに表現しようとする場合に、舞い落ちる木の葉や川辺の石ころなどの少しずつ形の異なった大量の形状を描画する場合がある。こういった個体差を持った形状をリアルタイムで描画するにあたって、個体差を持った形状をすべて読み込み描画する方法がある。しかしこの方法では描画する数の形状データ全てを読み込む必要があるため、データファイルが大きくなる。そこで、本研究では個体差を持った形状を大量に作るときにひとつの形状データと、描画する数の変形パラメータを持たせることで描画に使うデータファイルを少なくする手法を提案する。ファイルが小さくなれば保存媒体でのデータ占有率が少なくすむ、ファイルの入出力が迅速に出来る、ネットワークを介した３DCG の利用に便利になる、などの利点が生まれる。本手法では基となる形状にパラメータを与えるとパラメータにあった変形をした形状を作り出せる。変形された形状はその変形に使ったパラメータとして保存でき、基となる形状データと変形のパラメータがあれば復元ができる。

1.2 論文の構成

本論文の構成は次の通りである。まず第 2 章では本研究の手法について論じ、第 3 章にて、第 2 章で述べたプログラムの検討を行い、最後に第 4 章において研究の成果についてのまとめを行い今後の展望に関して触れる。

第2章　個体差を持たせるための変形

本章では、本研究で開発した手法について紹介する。本手法の特徴は個体差を持った大量の物体を一つの形状から行うことでデータファイルを少なくし、リアルタイムで描画できることである。

2.1　本手法の概要

本手法の内容は単一パターンの個体差を持った形状を大量に複製描画するにあたり、ひとつの形状データと描画する数の変形パラメータを持たせることで描画に使うデータファイルを少なくすることを目的とする。これにより、データファイルの読み込み時間を減らすことができ素早く描画を行える。データファイルの読み込み時間を減らすには大量の形状データを使うのではなく一つの形状データから一度にパラメータを与え個体差をもたせて大量に描画することで減らしている。そして、次の節以降で物体に個体差を持たせるための変形手法と、その個体差を持った物体を大量に作り出す方法を説明する。

2.1.1 これまでの似た形状の大量描画

似た形状を大量に描画する方法として一つは物体を変形せずに同じままコピー＆ペーストで流用しまったく同じ形状を大量に描画する方法がある。これはまず、描画したい物体を用意しその物体一つのデータファイルだけを読み込みメモリに保存して描画する数そのままの形で大量に描画する。この方法は一番簡単な方法でひとつの形状ファイルを用意するだけ済み描画にかかる時間も少なくて済む。このときの描画の流れは図１の左（１）で、物体 A そのままで大量に複製し描画、そして物体のデータファイルはひとつの形状のみなのでデータ読み込みにかかる時間は少なく済む。この方法以外に、全てを同じ形状で描画するのではなく個体差を持たせたいときがあるだろう。そのときに使える方法として、予めそれぞれ個体差を持った多数の形状を用意しておき、これらを描画する方法がある。これは物体を描画する前に描画する数だけ変形をしておき、それら変形した物体を全て読み

込み、メモリに保存し大量に描画する。こちらの問題としてあげられるのが、作りおきした変形後の形状全てのデータファイルを読み込まなければいけないのでデータファイルが大きくなってしまい、ファイル読み込みに時間がかかってしまう。このときの描画の流れは図１の右（２）で、物体 B を描画前に変形をして B１,B２,B３,B４を作っておき、作った B１,B２,B３,B４のデータファイルを読み込み、そして描画時にすでに変形している B１,B２,B３,B４を一つずつ読み込み描画する。メモリに保存されるのは予め作っておいた B１,B２,B３,B４で、描画する数が増えれば増えるほど作りおきしなければならない。

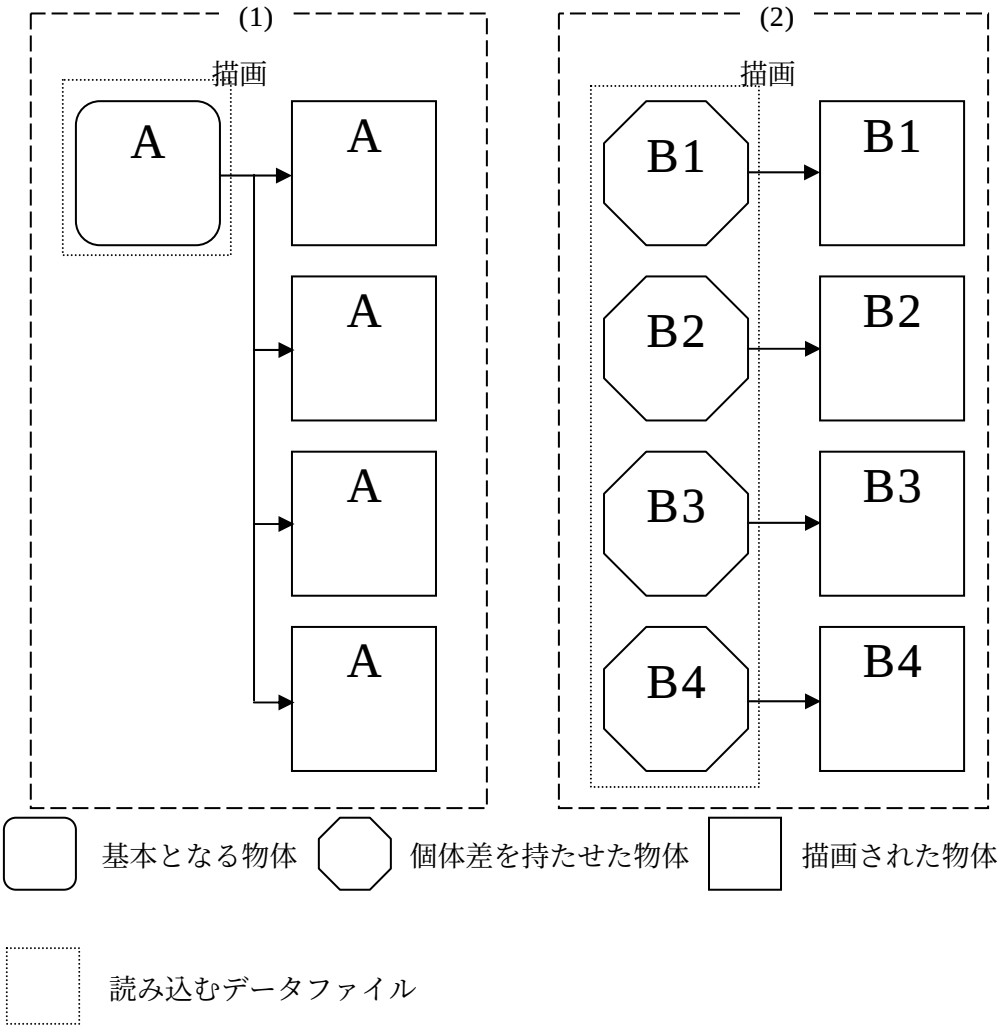

図１　大量な複製描画までの流れの例

2.1.2 本手法の流れ

個体差を持った形状を大量に描画したいときに前述した形状データを描画する数用意する方法では、データファイルが大きくなってしまいデータファイルの読み込みに大変時間がかかってしまうので、描画までにかかる時間を早くするために、データファイルを小さくする手法を提案する。本手法は一つの形状に変形のパラメータを与えるだけで形状が変形され個体差を持った物体を大量に描画することができる。これは、図 2 で、形状のデータファイルの物体 C１つだけを読み込み、物体 C に変形の属性番号を与えると番号に割り当てられた変形方法で変形される。例えば物体 C に属性番号を３と２を与えた場合描画されるのは C３と C２である。本手法の特徴は４つあり、個体差が出せること、同じパラメータを選ぶことにより同じ変形を行うことができること、変形パターンが大量にあること、これらの処理が高速に行えることである。

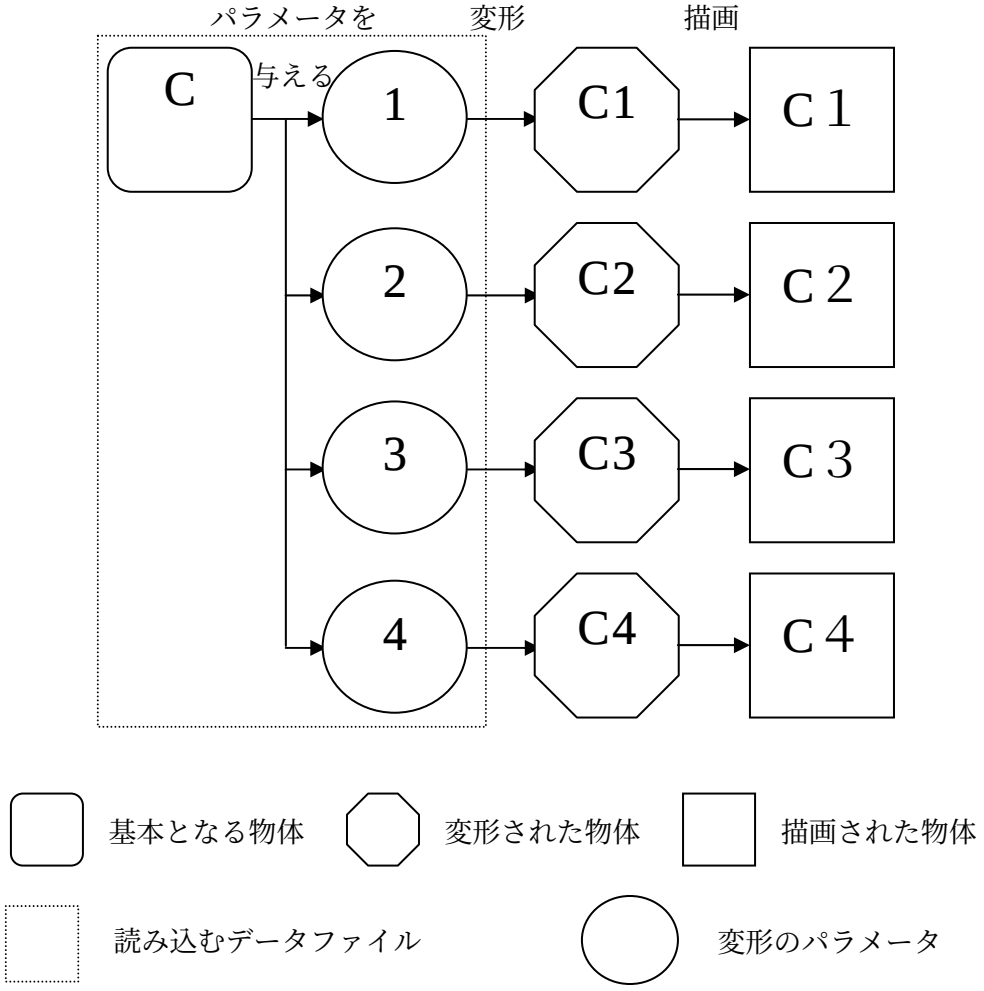

図２　本手法の流れ

2.2 変形手法

本研究で個体差を持たせるために採用した変形手法は自由形状変形（Free-Form Deformation 以下 FFD）をより制御を容易に、かつ計算負荷を軽くした軸変形（Axial Deformations）を用いる。

FFD とは３DCG の物体を変形するための一般的な手法である。変形対象となる物体に対して制御格子と呼ぶパラメータ空間を設定し、制御格子を歪めることで物体を大域的に変形する手法である。FFD をはじめて実用的な形で発表したのは Sederberg と Parry である[1]。その後 FFD の有用性が認められ、CG アニメーションをはじめとする多くの分野で重要な技術として利用された[4][5][13]。Sederberg らによる FFD の手法では、変形対象となる物体に対してそれを囲むような制御点群を格子状に並べて制御立体を生成する。その後制御点の移動によって制御立体を変形させることにより、内部のパラメータ空間を変化させる。これを変形対象の物体に写像することで物体の変形を行う。

Sederberg らは、[1]の中で、FFD の物理的な例を次のように説明している。まず、変形したいと思う何個かの物体が埋め込まれた透明で柔らかい平行六面体の樹脂を考える。この平行六面体を変形すると、物体も柔らかいと考えるので、その周りにある樹脂とともに変形する。Sederberg らによるアルゴリズムは汎用性が高く、ほとんどの形状モデルに対して適用できる。

しかし問題として５つの点が指摘されている。第１に、制御格子として正則な平行六面体しか使えない点である。第２に、Bernstein 多項式を用いているため高次の制御格子を用いると計算負荷が高い点である。第 3 に、１つの制御点の移動が制御格子全体に及ぶため、部分的な変形の際にも全体を計算する必要がある点である。第 4 に、変形後のモデルの形状を予測するのが難しいという点である。第 5 に、制御点が増えると変形のインタフェースが煩雑になるという点である。

FFD の問題を克服するために、FFD を改良した変形手法が数多く発表された[2][3][6][7][8][9][14][15][16]。例えば、Griessmair と Purgathofer は、Sederberg らが 3 変数の B´ezier 立体で表現していた制御格子を 3 変数 B-Spline 立体に拡張し、B-Spline 基底関数の特徴を生かした局所的な変形を可能とした[14]。また、Coquillart は、chunk と呼ぶ低次の制御格子を組み合わせた EFFD (Extended Free-Form Deformation) と呼ぶ手法を提案した[15]。この手法は小さい制御格子を組み合わせることにより、制御格子の精度を上げて局所的な変形も行うことができる。さらに Lamousin と Waggenspack は、NFFD(NURBS-based FFDs) と呼ぶ手法を提案し、変形形状の自由度を向上させた[16]。MacCracken と Joy は、任意の位相の制御格子を定義し、曲面分割による空間分割で立体頂点のパラメータを計算し、滑らかな変形を実現する手法を提案した[3]。最近では、山野上と若山は制御格子に弾性モデルを適用することにより、モデルのしなやかさの表現を可能にした[8]。このような改良と、コンピュータ自体の処理能力の向上に伴い、FFD はインタラクティブな変形手法として注目を浴びた。現在では *SOFTIMAGE*™[17] や *AnimationMaster*™[18] などのアニメーション作成ソフトウエアでも使用されている。そして軸変形も FFD を改良した変形手法の中の一つである。

軸変形とは、FFD と別に独自で考案されたものも多いが、基本的な考え方は FFD と同じで、FFD のサブセットともいえる。軸と呼ぶ線を変形媒体として変形を行う変形手法である。軸を変形の媒体とする利点は次の３点である。第１に、制御格子に比べて少ない制御点で操作できるため、変形が容易である。第 2 に、制御点が少ないため計算負荷が軽くすむ。第 3 に、制御軸の形状が物体の形状をある程度近似するため、物体の変形が把握しやすい。

本研究では軸変形を変形手法として採用した。軸変形の基本的な処理の流れは次の通りである。

(1) 変形対象の物体に対して変形媒体となる軸 R を定義する。この軸の位置は物体の内側であっても外側であっても構わない。定義した後、モデルの形状に適合するように、軸の位置や姿勢を修正する。軸とは直線もしくは曲線を指す。

(2) モデルの点 P を軸 R に投影する。ここで、投影とは点 P から一番近い軸 R 上の点 M を得ることである。

(3) 軸 R の形状を変形し、変形した軸をもとにモデルの形状を変形する。

2.2.1 制御軸による変形アルゴリズム

本システムでは、オブジェクトの変形に制御軸と呼ぶ媒体を用いる。制御軸は制御点と制御曲線から構成される。制御曲線は、制御点から導き出される Bezier 曲線[10]である。制御曲線に Bezier 曲線を用いた理由は、制御点の位置から一意に曲線形状が決まるため制御点の座標から直感的に曲線を予想でき、また曲線に沿った滑らかな変形を実現できるためである。

（1） オブジェクトに制御軸を作成するために以下 A）~D）のステップを実行する。

A） オブジェクトの全ての頂点座標を取得し、頂点に ID をつける。

B） 全ての頂点の x 座標を大きさの順に並べ一番大きいｘの頂点を X 一番小さいｘの頂点を X'とし、x 軸上の線分 XX'を作る。この作業を y 座標と、ｚ座標も行い線分 YY'と、線分 ZZ'を作る。

C） B）で作った線分 XX'線分 YY'線分 ZZ'の長さを測り、一番長い線分を軸の基とする。

D） C）で作った線分を 4 等分して求められる５つの点を $P_1P_2P_3P_4P_5$ とする。この５個の点から 4 次 Bezier 曲線を作り、これを制御軸とする。4 次 Bezier 曲線の式を下に紹介する。

$$c(t) = (1-t)^4 P_1 + 4t(1-t)^3 P_2 + 6t^2(1-t)^2 P_3 + 4t^3(1-t)P_4 + t^4 P_5 \qquad (0 \le t \le 1)$$

ただし、制御軸を構成する 5 つの制御点は C）で作成した軸の基の垂直方向にだけ移動できるようにする。

（２） 軸の基が線分 YY’ならば、図３のようにオブジェクトの頂点 V のｙ座標を持つ y 軸上の点を、線分 XX’ならばｘ座標を持つｘ軸上の点を、線分 ZZ’ならばｚ座標を持つｚ軸上の点を点 Q とする。軸の基が線分 YY’ならば端点からの割合 $\frac{YQ}{YY'}$ を、線分 XX’ならば端点からの割合 $\frac{XQ}{XX'}$ を線分 ZZ’ならば端点からの割合 $\frac{ZQ}{ZZ'}$ をとり、上記の Bezier 曲線の式の t に代入して出た点を Q’とする。端点からの割合を t に代入することで、Bezier 曲線を形成する線の同じ高さの点を表せる。点 Q’から $\overrightarrow{QV}$ の分、動かした点を制御軸変形後のオブジェクトの V’＝Q’+ $\overrightarrow{QV}$ となる座標 V’に V を移動する。この作業を全頂点に行う。

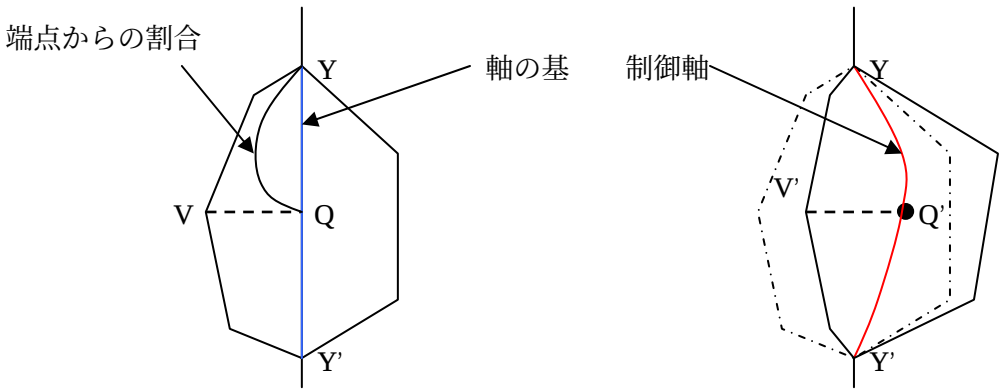

図 3　制御軸の端点からの割合

2.2.2 本手法での変形方法

2.2.1 の軸変形は制御軸を変形すると形状が変形するものなので、制御軸を構成する５つの制御点を動かすことで形状の変形を行う。つまり、この５つの制御点の位置で出来上がる形状が決まる。そこで、再現性を持たせるために、ひとつの形状に対しこの５つの制御点の位置を決める数値１０個と、その形状の描画する座標値を表す数値３個、形状の向きを表す数値３個の合計１６個の数値を一組として描画する形状の数の組分、数値をファイルから入力することで描画する。

第3章　結果

3.1　検証

本節では、一つのファイルだけ読み込んでコピー＆ペーストで描画する方法と、描画する数だけ一つ一つファイル読み込みを行い描画する方法と、本研究で提案する手法を比較することにより、検証を行う。この検証では 58 個の頂点からなる石の形状データを使う。

まず同じ形状に対して、コピー＆ペーストで描画する方法と、一つ一つ形状のデータファイルを読み込み描画する方法と、本手法による個体差を伴う類似形状の描画方法を面有りとワイヤーフレームの２つの方法で、100 個と 1000 個と 10000 個描画してそれぞれの描画にかかる時間、プログラムを実行してから 1 度目の描画が行われるまでの時間、１秒間に描画できる個数をそれぞれ比較する。

本章の表及び図中で、「全て同じ」というのはコピー＆ペーストで描画する方法のことを指しており、「全読み込み」というのは一つ一つデータファイルを読み込み描画する方法のことを指している。「ワイヤー」はワイヤーフレームで描画することで、「面」は面も描画することを指している。

表 1 はそれぞれの方法で描画するに当たり、1 度目の描画までに読み込まれるデータファイルの大きさである。「全て同じ」では 1 つの形状ファイルのみ読み込みなので変わらず一定の大きさで済み、「全読み込み」では描画する数読み込むので、それぞれ 10 倍の大きさのデータ量になる。「本手法」では、１つの形状ファイルと描画する数のパラメータ値なので、パラメータの量と形状ファイル１つ分の大きさになる。

表１　データファイルの大きさ

	10000 個	1000 個	100 個
全て同じ	10.4KB	10.4 KB	10.4 KB
全読み込み	103906 KB	10391 KB	1039 KB
本手法	1450 KB	158 KB	24.8 KB

図５はプログラムを実行してから描画されるまでの時間を測ったグラフである。これは、一度目の描画までの時間をあらわしており、おもにファイルの読み込みや変形をする時間である。それらの作業でかかった時間で比較している。この比較の結果、一つの形状データファイルの読み込む速度が遅いので、ファイルの読み込みをたくさんする「全読み込み」の方法では大量の時間がかかってしまう。「全て同じ」の方法では形状データの読み込みは一つだけで済んでいるので速く描画できる。「本手法」では形状データの読み込み一つとたくさんの形状の変形操作をしているが変形よりもデータファイルの読み込みのほうが時間をとってしまうため本手法と比べて、本手法のほうが速く描画される。

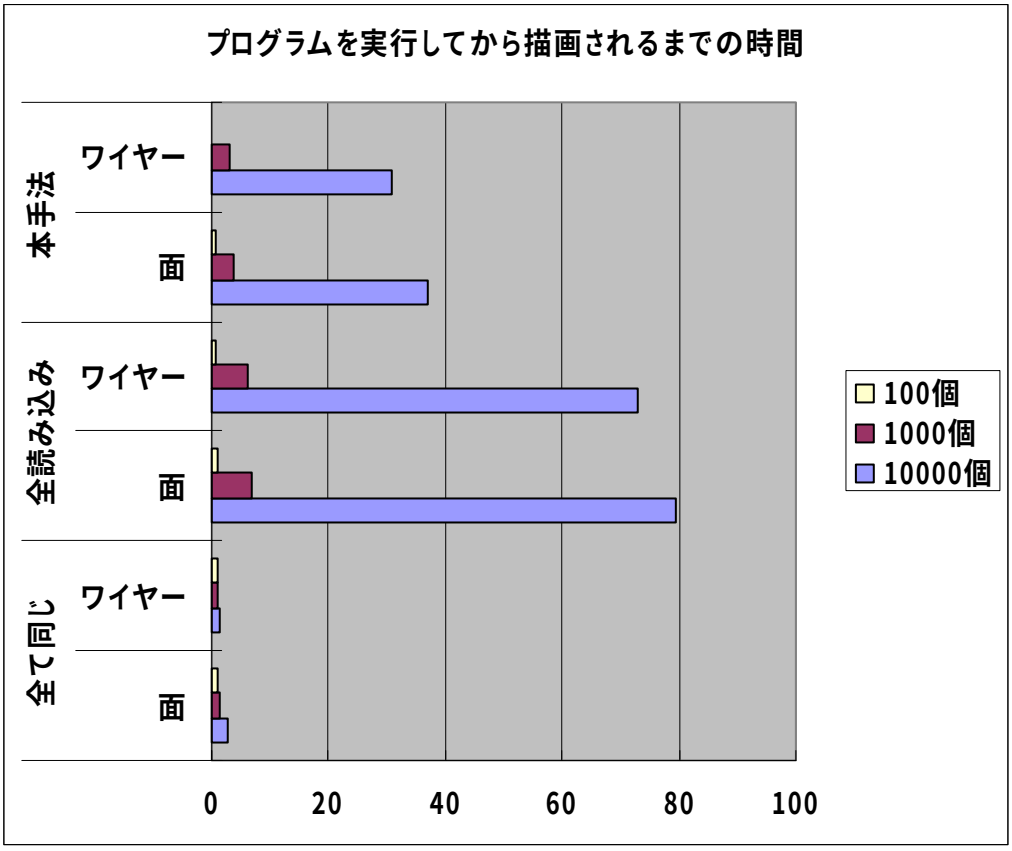

図５　１度目の描画までにかかる時間

図６のデータは二度目以降の描画にかかっている時間である。これは一度目の描画でメモリにデータが入っているので二度目以降の描画は素早く描画でき毎回同じ速度で描画される。その描画時間を比較している。結果は３つの手法共にそれほど差が出なかった。「全て同じ」コピー＆ペーストの方法のデータはどの方法よりも一番早く描画できるので、そのコピー＆ペーストの方法と差がなく描画できるということはリアルタイムに適しているといえる。

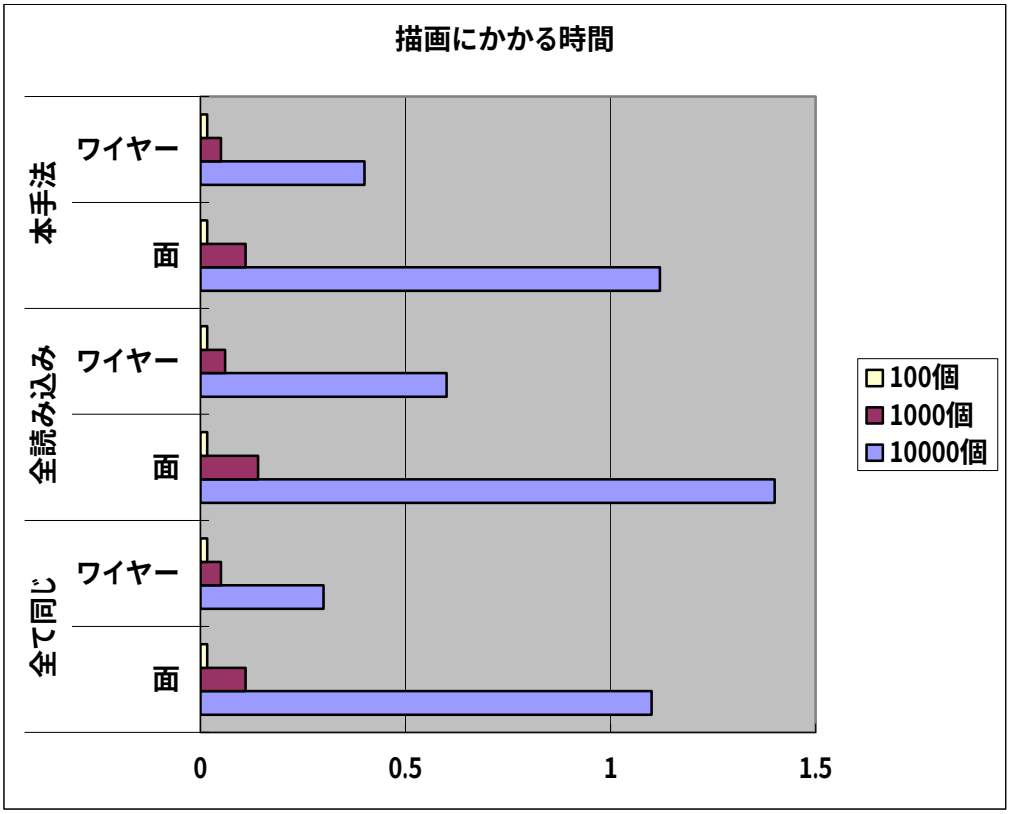

図６　描画にかかる時間

表２はそれぞれの方法を行って１秒間に描画することのできる形状の個数を表している。この１秒間に描画できる形状の個数においてもコピー＆ペーストに近い値が出ているので、描画の速さに問題はない。

表２　１秒間に描画できる形状の個数

	fps
全て同じ	**9100 個**
全読み込み	**7120 個**
本手法	**8900 個**

図７と図８は本手法の面有りとワイヤーフレームとで、同じ変形属性を与えて描画した図である。この結果により、同じ変形属性を与えることで同じ変形の仕方ができることが証明された。

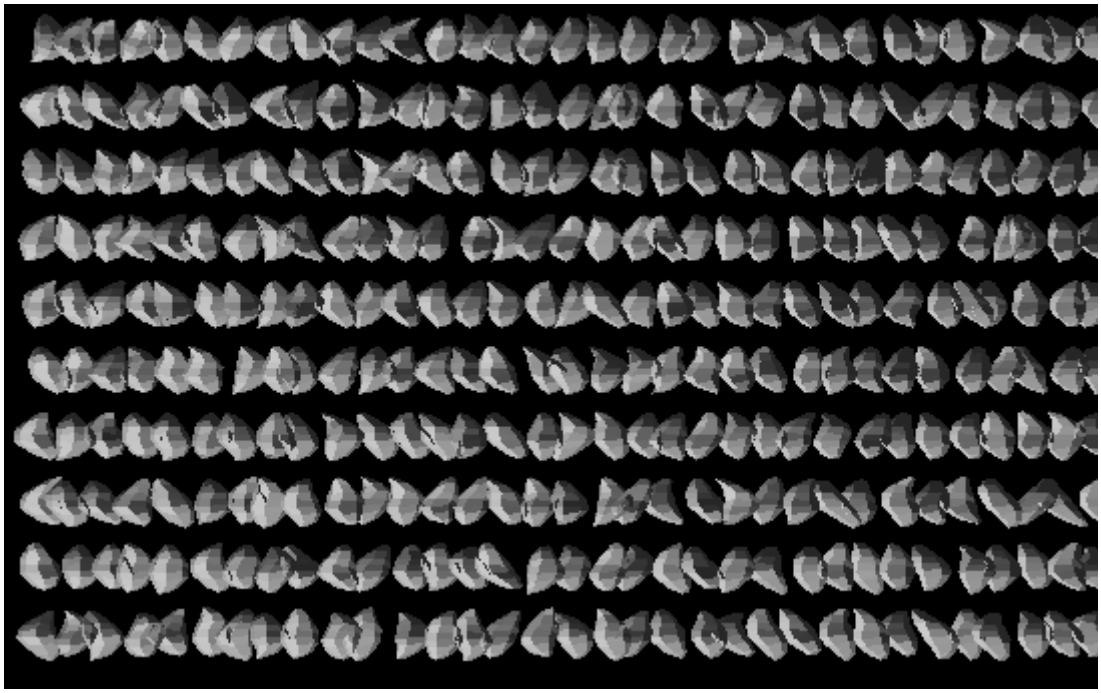

図７　面ありでランダム変形と軸変形を行った図

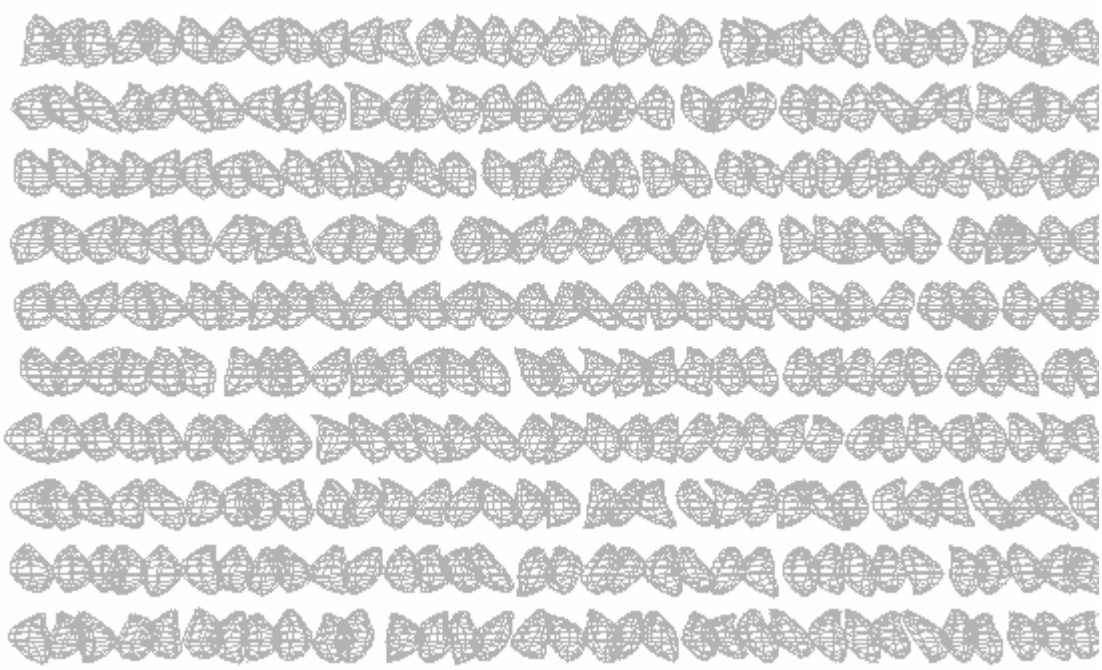

図８　図７と同じ変形を行いワイヤーフレームで描画した図

図９は葉の形状 1 枚を本手法を使って個体差を持たせ、1000 枚ランダムに配置し描画した図で、描画までに 4.65 秒かかった。

図９　　葉の形状への応用

3.2　考察

描画する数が増えれば増えるほど本手法を使ったほうがデータファイルの読み込み時間を抑えて描画される。これは本手法ではデータファイルが形状 1 つと変形パラメータだけ読み込めば個体差を持った物体を大量に作り出すことができるので、描画する数が増えても描画までにかかる時間は変形にかかる時間だけで済む。変形はファイル読み込みよりも素早くおこなえるため、1 度目の描画をされるまでの時間に差が出たのである。読み込むデータファイルが描画する数に比例して増えていくので描画する数が増えれば増えるほど本手法は一つ一つ読み込む方法より速くなる。

そして描画にかかる時間で面有りとワイヤーフレームで差が出たのは、面有りの場合、変形すると面の角度が基データと変わるためレンダリング時に面一つ一つ新たに法線ベクトルを計算しなければならないため、面が増えれば増えるほど計算時間がかかってしまうためである。

3.3　本手法の有用性

本手法の解決すべき問題であった個体差、再現性、多様性、高速化が実現されているかで本手法の有用性が決まってくる。まず個体差は無事実現されたことは描画されたものを見ればわかる通り実現され一つ一つに個体差を持たすことができた。再現性についても図７と図８を上下比較して見ればわかる通り変形方法が同じだとわかる。多様性については、ランダムに揺らがすことと、軸変形を用いたことにより、たくさんの変形パターンが出来た。高速化はデータファイルの読み込み時間が少ないため描画する数が増えれば増えるほど本手法を使ったほうが早く描画出来た。

第4章　まとめ

4.1　結論

本研究では、一つの形状から個体差を持った類似形状の高速な生成および描画の手法を提案した。一つの形状と変形パラメータにより類似形状の生成を行ったことにより、データファイルの容量を小さくまとめることができる。物体に個体差を持たせるための変形には軸変形を使い物体を曲げた。この変形にパラメータを使うことによって好きな変形をさせることができる。この手法では大量に描画すればするほど他の方法で描画するよりデータファイルの読み込み時間が少なく済むため描画が早くできる。

本手法を使わない場合はデータファイルが大きいため読み込みに時間を取られ、1 度目の描画までに時間を多く取られていたが、これを本手法ではデータファイルを小さくまとめることで早くできるようになった。

本手法を使うことによりデータファイルが小さく済むため、ネットワークを介した CG に有効であるだろう。

4.2　今後の展望

どんなモデルにも対応できるような変形方法が必要であり、更なる描画速度の向上をめざす必要がある。

そしてこの研究の目標として Image Analogies[19]に書かれているような範囲を設定したところに同じようなものを描くという技術を本手法に適用して、例えば 3 次元空間上に木の幹と葉や石などのオブジェクト一つずつだけで風景を自動で作れるようなものの作成も期待される。

謝辞

本研究を実施するにあたり、日ごろから適切な指導と多くの助言をして頂きました東京工科大学メディア学部　渡辺　大地　講師に感謝いたします。

電気通信大学大学院　武田研究室の和田　篤氏には論文執筆にあたり貴重な助言を頂きました。

そして東京工科大学　メディア学部　メディア学科の諸氏に感謝いたします。

最後に日頃から多くの力添えや助言を頂いた先輩や同輩、そして後輩に深く感謝し、厚く御礼申し上げます。

参考文献

[1] Thomas W.Sederberg and Scott R.Parry, Free-Form Deformation of Solid Geometric Models,Computer Graphics, Vol.20, No.4 ,Aug.1986

[2] Karan Singh and Eugene Fiume, Wires: A Geometric Deformation Techique,Computer Graphics (proceedings of SIGGRAPH'98),1998

[3] Ron MacCracken and Kenneth I.Joy, Free Form Deformation with Lattice of Arbitrary Topolpgy,Computer Graphics (proceedings of SIGGRAPH'96),1996

[4] 柿本正憲,インタラクティブ変形モデラの開発,NICOGRAPH 論文集,1992

[5] 渡辺大地,早野勝之,千代倉孔明,柔軟な形状制御機能を持った Cyber World Toolkit,第１１回 NICOGRAPH 論文コンテスト論文集,1995

[6] F.Lazarus,S.Coquillart and P.Jancene, Axial deformations: an intuitive deformation technique, Computer Aided Design, Vol.26,No8 ,Aug 1994

[7] 山野上寛,村方仁,千代倉孔明,「制御軸を用いた直感的な大域変形操作の実現」,NICOGRAPH/MULTIMEDIA 論文コンテスト論文集,1997

[8] 山野上寛,若山和子,「リアルタイム 3 次元 CG におけるしなやかさを考慮した物体表現の有用性」,3D Modeling, vol 2,千代倉研究室 編, 1996.

[9] 山野上寛,「制御軸を用いた大域的な形状変形手法に関する研究」,慶応義塾大学大学院政策メディア研究科修士論文,1998

[10] 千代倉弘明,ソリットモデリング,工業調査会,1985

[11] Open GL Programming Guide, Addison-Wesley,1992.

[12] M.Pesce, VRML-Brouwsing and Building Cyberspace, New Riders Pudlishing, 1995.

[13] On-Line Computer Graphics Notes http://graphics.cs.ucdavis.edu/GraphicsNotes/Free-Form-Deformations/Free-Form-Deformations.html

[14] J.Griessmair and W.Purgathofer, Deformation of Solids with Trivariate B-Splines, *Proc. Eurographics 89*, Elsevier Science Publishers, North-Holland, 1989.

[15] S.Coquillart, Extended Free-Form Deformation: A Sculpturing Tool for 3D

Geometric Modeling, *SIGGRAPH Proceedings*, 1990.

[16] H.J.Lamousin, W.N.Waggenspack, NURBS-Based Free-Form Deformations, *IEEE Computer Graphics and Applications*, Nov.1994.

[17] SOFTIMAGE_index http://www.sofimage.jp/

[18] Hash, Inc. http://www.hash.com/

[19] Aaron Hertzmann (New York University and Microsoft Research)Charles E. Jacobs, Nuria Oliver (Microsoft Research)Brian Curless (The University of Washington)David H. Salesin (The University of Washington and Microsoft Research)Image Analogies,Images and Textures(proceedings of SIGGRAPH'96)