NAIST-IS-DT9961024

博士論文

陰関数曲面とパーティクルシステムを用いた仮想粘土細工による自由形状モデリングに関する研究

松宮 雅俊

2002年3月22日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

松宮 雅俊

審査委員：　横矢 直和 教授
千原 國宏 教授
湊 小太郎 教授
竹村 治雄 助教授

陰関数曲面とパーティクルシステムを用いた仮想粘土細工による自由形状モデリングに関する研究*

松宮 雅俊

内容梗概

本論文では，粘土細工を行うような感覚で自由形状のモデリングが行える仮想粘土細工システムについて述べる．まず 1 章では，本研究の背景と目的について述べる．2 章では，仮想現実感技術と陰関数曲面を用いることにより，手を用いて直接的に仮想粘土へ凹凸を加えるという形状の変形を実現する手法について述べる．形状変形に際しては，陰関数曲面表現の特徴を利用して，形状の衝突判定や変形処理に複雑な処理を必要とせず，その定義から自然に変形を表現している．次に，3 章では，パーティクルシステムと陰関数曲面を用いた仮想粘土モデルに対し，手を用いて変形を加える手法について述べる．現実の粘土の変形挙動を少ないパーティクルで表現するとともに，粘土の表面形状を陰関数曲面を用いて表現する．4 章では，上記の 2 つの手法を組み合わせることで，粘土へらと手を用いた，より表現力の高い粘土細工を行える仮想粘土細工システムについて述べる．対話的に変形操作を行うためには，一連の処理を高速に行う必要がある．そこで，上記のそれぞれの手法に対して，陰関数曲面を効率良くポリゴン表現に変換する手法および，対称型マルチプロセッサシステムを用いた並列処理について述べる．また，それぞれの手法を実装した試作システムによる形状変形例を示し，対話的な仮想粘土細工が行えることを実験的に示す．最後に 5 章で本研究を総括する．

キーワード

仮想粘土，陰関数曲面，仮想現実感，パーティクルシステム

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DT9961024, 2002 年 3 月 22 日.

Interactive Virtual Clay Modeling Using Implicit Surfaces and Particle Systems*

Masatoshi Matsumiya

Abstract

This paper describes a new interactive free-form modeling scheme based on the metaphor of clay work. First, the paper discusses design issues and immersive modeling systems which enable a user to design 3D solid objects with curved surfaces by using one's hand intuitively and interactively. Shape deformation can be expressed by simple formulas without complex calculations due to the nature of skeletal implicit surfaces employed to represent smooth free-form surfaces. The paper then presents a virtual clay model using particle systems and implicit surfaces. Particle systems enable the virtual clay model to behave like real clay preserving the volume. Implicit surfaces enable the virtual clay model to have smooth surfaces surrounding particles. Finally, a modeling system based on the above two methods is described. As for an interactive deformation of virtual clay model, a polygonization algorithm has been developed for fast rendering of virtual clay model and multi-threaded processing has been employed for implementing the entire algorithm. Experiments have shown the feasibility of the proposed methods.

Keywords:

Virtual Clay, Implicit Surfaces, Virtual Reality, Particle Systems

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9961024, March 22, 2002.

目 次

図 目 次

表 目 次

1. 序論

1.1 本研究の背景と目的

1.1.1 3 次元自由形状モデリングの難しさ

3 次元コンピュータグラフィクス (CG) を効果的に使用した画像や映像は，プレゼンテーション，映画，コマーシャル，ゲームなどに積極的に活用され，3 次元 CG の必要性は高まる一方である．また，専門家だけではなく，3 次元 CG を作成したいという一般のユーザも増加している．このような 3 次元 CG の作成では，3 次元形状を計算機内に構成するモデリングと呼ばれる作業が必要になる．

3 次元形状のモデリングに一般に用いられている 3 次元形状モデラとしては，AutoCAD[1] などの 3 次元 CAD(Computer Aided Design) ソフトウェアや，Ray Dream Studio[2]，LightWave 3D[3]，Shade[4] などの 3 次元 CG ソフトウェアのモデラ部分がある．これらのソフトウェアでは一般に，図 1 および図 2 に示すように，3 次元形状の透視投影図や 3 面図，形状の組み合わせや親子関係を示した構成図（一般にツリー構造），様々な機能を選択し実行するためのアイコンやメニューなどが 2 次元ディスプレイ上に表示され，それらに対してマウスやタブレットなどの入力装置を用いて 3 次元形状のモデリングを行う．

これらのソフトウェアを用いて 3 次元形状をモデリングする方法は様々であるが，AutoCAD ではメニューやコマンドによって直接座標値を入力したり，4 本の曲線によって閉じた平面図形を描き，その間を曲面として内挿する方法などがある．また，Ray Dream Studio ではポリゴンによって近似された曲面形状を構成し，ポリゴンの頂点をマウスで移動することによって曲面形状を変形させる方法や，平面図形と曲線をマウスで描き，平面図形を曲線に沿って押し出すことで立体化する方法などがある．LightWave 3D では MetaNURBS と呼ばれるパラメトリック曲面の制御点をマウスで直接移動したり，メニューから選択した機能によって移動することで曲面を変形させてモデリングを行う．Shade では図 3 のような曲面形状をモデリングする場合，図 4 の三面図内の制御点やハンドルと呼ばれる曲面形状を決定するものをマウスで移動させることで行う．

以上のように，一般の 3 次元 CG ソフトウェアにおけるモデラでは，複雑な幾

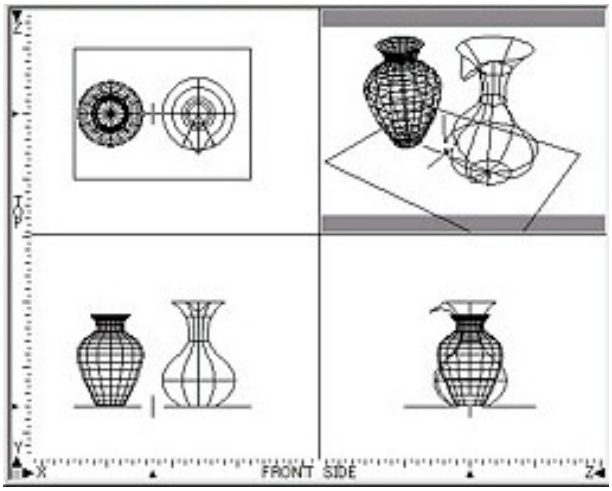


図 1　三面図と透視投影図が表示された市販 3 次元 CG ソフトウェアの画面例（文献 [4] より引用）

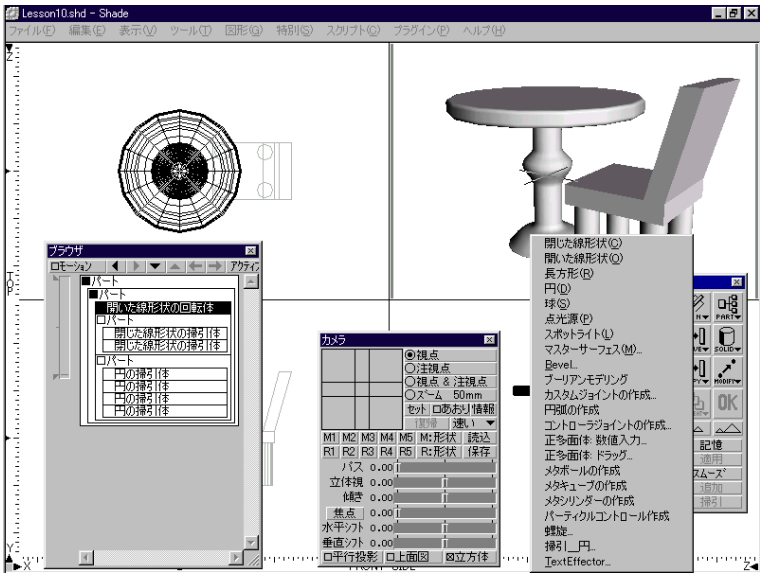


図 2　市販 3 次元 CG ソフトウェアでモデリングに使用するアイコンやメニューの例（文献 [4] より引用）

何操作や位相操作によって 3 次元形状のモデリングを行う．そのため，ユーザには幾何や位相に関する数学的知識が必要なうえ，多くの複雑な操作を覚えなければならない．さらに，3 面図や投影図などの 2 次元表示を用いて 3 次元形状を把握する能力も要求される．すなわち，従来のモデラでは，3 次元形状のモデリングに要するユーザへの負担が非常に大きい．これは，これから 3 次元 CG を始めようとする初心者には敷居の高いものである．また，初心者以外のユーザにとっても，形状をモデリングする際，どのような幾何，位相操作を組み合わせるべきなのかを考えなければならず，形状の設計そのものに集中することができない．

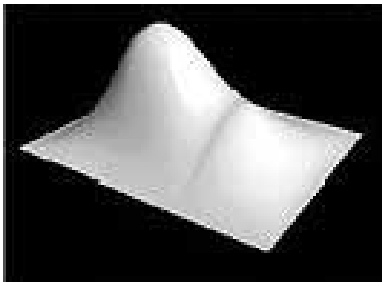


図 3 曲面の CG（文献 [4] より引用）

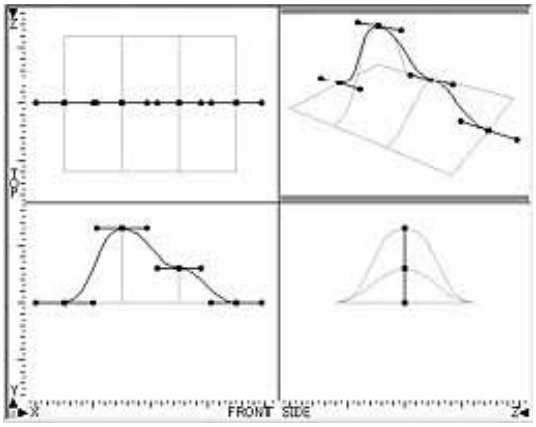


図 4 市販 3 次元 CG ソフトウェアでの曲面のモデリングに使用する制御点とハンドル（文献 [4] より引用）

1.1.2 芸術的造形とバーチャルリアリティ

これまで述べてきた一般の 3 次元 CG ソフトウェアや 3 次元 CAD ソフトウェアで用いられている造形手法に対して，彫塑のような芸術的造形手法がどのように異なるのかを，計算機を用いた造形の観点から整理する [5].

- 対象とする造形物

  3 次元 CAD 工業製品等の機能的な構造物や部品のような，主として数値で明確に規定できる形状の造形を目的としている．また，内部の部品による外装形状の制約等，様々な制約の中で造形される．

  3 次元 CG 3 次元 CAD と芸術的造形の両方の側面を持つが，芸術的造形物に感じられる手作り感のある造形は困難である．例えば,「粘土らしさ」を出すために，図 5 左のような球に対し凹凸を付け，図 5 右のような形状を作成するなどの技術を必要とする [6].

  芸術的造形 形状を作者の頭に描いたイメージに近づけることが目的であり，数値的正確さより人間の感覚的判断による対話的試行錯誤作業が造形の基本である．また，中間段階で生成された形状から次の新しい発想が誘発され，作業の進行に伴ってイメージが変化していくことが多い．このようにして作られる造形物には，手作り感が感じられる．

- 造形インタフェースの機能

  3 次元 CAD・CG 座標値や関数パラメータなどで規定される形状の定義に適する造形インタフェースになっている．

  芸術的造形 頭に描いたイメージを即座に形にできるような直感的造形環境が必要である．そのためには，意図に忠実な造形が簡単な操作で実行できる造形インタフェースが要求される．意図した精度で 3 次元加工が可能な操作インタフェースが必要である．

- 造形インタフェースの構成

3 次元 CAD・CG 3 次元造形を 2 次元の作業に置き換えてマウスを用いるインタフェースが多い．高度な機能を備えるために操作法が複雑になっている．

芸術的造形 造形の自由度を重視する必要がある．その場合，2 次元作業に置き換え難い 3 次元作業が多くなる．そこで，マウスによる 2 次元操作より，直接的 3 次元操作が必要になる．直感的造形作業にも 3 次元の直接操作が適する．機能選択などの操作法を意識することは直感的思考の妨げになるので，単純な機能で意図した造形ができるようなインタフェースの開発が必要である．

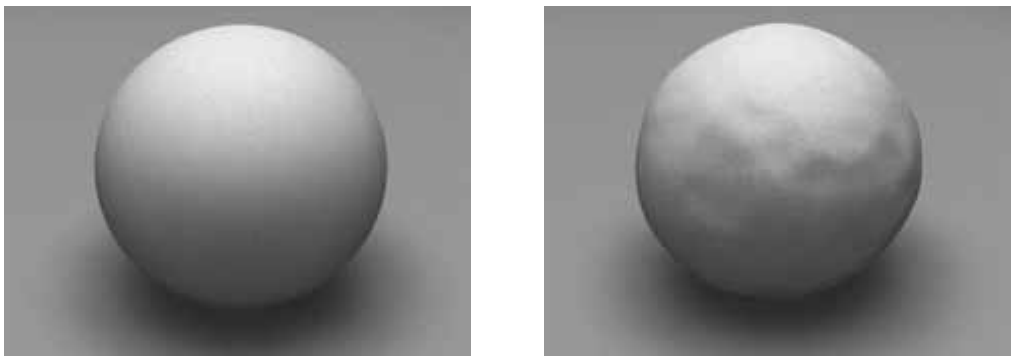


図 5 粘土らしく見せるための凹凸付け（文献 [6] より引用）

一方，バーチャルリアリティ (VR) 技術の発展とともに，現実世界の体験を，VR 技術を応用して計算機上で仮想体験する様々な研究やアミューズメントシステムの開発が行われている．その中でも，仮想空間における対話的な 3 次元形状の作成に関する研究が，VR における重要な課題の一つとして研究されている [7, 8, 9, 10, 11, 12].

没入型モデラは，VR 技術によって計算機の生成する仮想空間に没入して 3 次元形状のモデリングを行うことのできるシステムであり，以下のような利点がある．

直感的な視認

HMD(Head Mounted Display)や液晶シャッター眼鏡などによって，左右の目にそれぞれの視点からの画像を提示して両眼立体視することにより，対象物を立体的に見ることができる．また，磁気トラッカなどの3次元位置センサによって，頭部の位置，姿勢を計測し，その視点からの画像を実時間提示する．このような，両眼立体視と視点追従画像提示の併用によって，3次元形状やそれらの位置関係を容易に把握することが可能となる．

直感的な操作

3次元位置センサや，それを取り付けた手形状入力装置などの装置を用いることで，2次元の入力装置では煩雑になりがちな3次元形状の操作を直接的に行うことができる．

対話的モデリング

仮想環境内では，3次元形状に対する操作は直ちに提示画像に反映され，操作結果が即座に視認できる．この形状操作と視認の同時性により，効率的なモデリング環境を構築することができる．

このように，前述した芸術的造形に関する諸点に対処するためには，VR技術を用いれば良いと考えられる．

1.1.3 本研究の目的

VR技術を用いて，粘土細工や彫刻などの誰もが経験のある形状作成手法によって3次元形状をモデリングできれば，前述した3次元形状モデリングの問題を解決することができる．粘土細工や彫刻などの造形手法では，素材を手で持って移動，回転をさせることや，頭を動かすことによって，好きな方向から眺めながら，手や道具を使って直接的に素材に変形を加えて造形していく．これをコンピュー

タ上の仮想空間で，現実空間と同様に行なうことができれば，特別な知識や難解な操作を新たに覚えなくても，なじみのある方法で直感的に形状をモデリングすることができる．

そこで，本研究では，なじみのある造形手法として粘土細工を取り上げ，コンピュータ上で現実空間と同様の粘土細工が行える自由形状モデリング環境の構築を目指す．さらに，従来のモデリングでは困難と考えられる，手作り感のある形状が作成できる自由形状モデリング環境の構築を目指す．

1.2 従来研究と本研究の方針

粘土細工や彫刻などのような誰もが経験のある造形手法を取り入れることで，直感的な形状モデリングを目指す研究が多数行われている [13, 14, 15, 16, 17, 18, 19, 20]．

まず，彫刻を模倣したモデリングに関する研究として，形状をボクセルデータで表現し，3 次元マウスを用いて操作される切削ツールの動きに従ってボクセル値を更新することで，仮想空間で対話的な形状モデリングを実現しているもの [21] がある．しかし，生成される形状は現実の彫刻のような細密なものではなく，形状操作においても立方体形状をした 3 次元マウスを操作するという現実には無い独特なものである．一方，楕円体で表現される彫刻刀と，平面または 2 次曲面で表現される彫刻素材との論理演算によって，現実の彫刻に似た形状を生成する研究 [22] がある．しかし，ユーザの視点移動に対する画像の描画に時間がかかる描画手法であるため，現実空間での造形作業のように形状を自由に眺めながら対話的にモデリングを行なうことができない．

また，粘土細工を模倣したモデリングに関する研究としては，ボクセル空間に配置された 3 次元セル構造オートマトンを応用したもの [23] や，粒子ベースモデルによって粘土の形状変形を実現しているもの [24, 25] がある．しかし，これらは対話的な形状変形や可視化を考慮していない．他にも，双三次ベジエ曲面で表現された形状を直観的に変形させる方法 [26, 27] があるが，変形手段や変形モデルの提案であり，自由形状のモデリングを行うには到っていない．メタボールを対話的に操作することによってモデリングを行う研究 [28] では，メタボールを移

動させることによる形状の変化が粘土に似ている．しかし，操作する対象はあくまでメタボールであり，メタボールを意識しながらのモデリングは，粘土細工の造形作業とは異なる．

また，粘土という具体的なものではないが，柔物体の変形に関する研究も行われている [29, 30]．しかし，これらは対話的な形状変形や可視化を考慮していない．特殊な装置によって柔物体の変形を反力を通して感じることが可能な方法 [31] は，物理モデルを用いたものではなく，変形挙動は現実の物体と異なる．境界要素法を基にした線形弾性体の高速な計算に関する研究 [32] や，物理モデルとサブディビジョン・サーフェスを用いた対話的な形状変形に関する研究 [33] があるが，これらは形状の分断や融合を表現できない．

その他の造形手法を模倣したモデリングに関する研究として，仮想ろくろを用いた回転体形状モデラ [34] がある．これは回転体の輪郭線を体積を一定に保ちつつ変形することで，ろくろを用いて作成したような形状を生成している．しかし，モデリング可能な形状は回転体に限られ，任意の自由形状のモデリングができない．

以上のように，様々な形で 3 次元形状のモデリングの問題点を解決する試みが行なわれているが，より現実空間での作業に近づけたモデリング環境が必要であると考える．

そこで，本研究では，まず，没入型モデラにおいて手や指を用いて 3 次元形状を直接変形させるという操作方法を採用する．制御点やパーティクルなどを意識して形状を変形させるような間接的な方法を用いずに，それらを意識せずに直感的，直接的に変形できる方法とする．このような，自由形状の対話型モデリングを実現するためには，以下のような課題を解決しなければならない．

1. 現実の粘土のような変形挙動を示す仮想粘土の実現
2. 仮想粘土と手形状とのインタラクションの実現
3. 対話的な形状変形が可能な時間での計算と描画処理の実現

これらの課題を解決するために，本研究では，パーティクルシステムと陰関数曲面による形状表現を採用し，形状変形と描画を高速に計算する手法を開発する．

1.3　本論文の構成

本論文の構成は以下の通りである．2 章では，仮想現実感技術と形状の陰関数曲面表現を用いて，仮想粘土を手で直接「凹ませる」ことと,「つまみ出す」ことによって，粘土細工を行うような感覚で自由形状のモデリングが可能な仮想粘土細工システムについて述べる．ここでは，粘土らしい変形挙動の表現については考慮していない．そこで，3 章では，パーティクルシステムと陰関数曲面を用いた仮想粘土モデルに対し，手を用いて変形を加えることのできる仮想粘土細工システムについて述べる．現実の粘土の変形挙動を少ないパーティクルで表現するとともに，粘土の滑らかな表面形状を陰関数曲面を用いて表現する．4 章では，2 章と 3 章で用いた手法を組み合わせた，手と粘土へらを用いた仮想粘土細工システムについて述べる．最後に 5 章では，本研究を総括する．

2. 陰関数曲面による手を用いた形状変形

2.1 緒言

本章では，仮想現実感技術と陰関数曲面を用いて，仮想粘土を手で直接「凹ませる」ことと,「つまみ出す」ことによって，粘土細工を行うような感覚で自由形状のモデリングを行うための手法について述べる [35, 36, 37, 38, 39, 40].

ここでは，ユーザ・インタフェースに仮想現実感技術 [7, 8] を用いて，計算機の生成する仮想空間に没入して 3 次元形状のモデリングを行うことを想定する．HMD（Head Mounted Display）を装着し仮想空間に没入したユーザが，手形状入力装置を装着した手で直接的に仮想粘土に変形を加えることによってモデリングを行う．また，3 次元位置センサを用いて，頭部と手の位置および姿勢を計測する．

このようなシステムでは，仮想の粘土と手とのインタラクションを表現しなければならない．現実空間での粘土細工では，粘土に指を押し込んでいくと，粘土が指の形状に沿って凹んでいく．また，指でつまみ出すと，つまんでいる指と指の間に粘土がつまみ出される．さらに，このような変形を加えても，粘土形状は滑らかである．このような仮想粘土を表現するために，陰関数曲面（Implicit Surfaces）の一種である，スケルタルサーフェス（Skeletal Implicit Surfaces）[41, 42, 30, 43] を利用する．スケルタルサーフェスは，距離場関数に従って，その骨格となるスケルトンを包み込むように生成される曲面である．このスケルタルサーフェスで仮想の粘土と手を表現し，適切な距離場関数とスケルトンの生成手法を開発することで，先述のような仮想粘土を表現する．

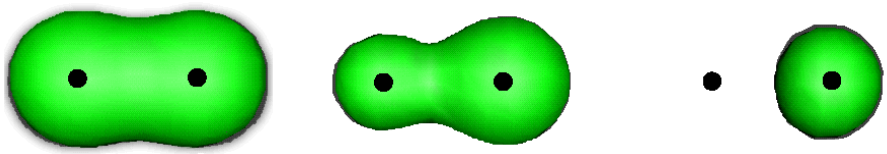
また，仮想粘土の変形やユーザの視点移動による画像を実時間で提示するために，スケルタルサーフェスによる形状を高速に描画する手法についても検討する必要がある．本章では，陰関数曲面を効率良く実時間でポリゴンに変換する手法を応用し，高速な描画を実現する．

2.2 陰関数曲面

本研究では自由形状の表現手法として，陰関数曲面の中でもスケルトン（Skeletal Element）と呼ばれる形状の骨格要素を基にして形状を生成するスケルタルサーフェス（Skeletal Implicit Surfaces）[42, 30] を用いる．スケルタルサーフェスは，一般に知られている，Blobby Molecule[44] やメタボール [45]，Soft Object[46] と類似のものであり，これらの違いは，その定義に用いられている関数の違いである．スケルタルサーフェスでは，この関数の定義が広く，概形が同じである様々な関数が用いられている．そこで，このスケルタルサーフェスで仮想の粘土と手を表現し，適切な距離場関数とスケルトンの生成手法を開発することで，先述のような仮想粘土を表現する．まず，本節ではスケルタルサーフェスについて述べる．

スケルタルサーフェスを用いた陰関数曲面は，自由形状は式（1）の陰関数多項式で表現され，このとき形状はスケルトン S と重み w，距離場関数（Field Function）F によって決定される．$f(P) < c$ となる点 P は形状の外部の点，$f(P) > c$ となる点 P は形状の内部の点，$f(P) = c$ となる点 P は形状の表面上の点であることを表す．ここで，f を陰関数値と呼ぶ．また，スケルタルサーフェスによる形状は常に C^1 連続な曲面となり，式の偏微分により容易に曲面上の任意の点での法線が求まる．

$$
\begin{aligned}
& \{P \in R^3 | f(P) = c\}, \\
where \qquad & f(P) = \sum_{i=1}^{n} w_i F_i(d(P, S_i)).
\end{aligned} \tag{1}
$$



(a)w_1=1.0 w_2=1.0　(b)w_1=0.5 w_2=1.0　(c)w_1=−1.0 w_2=1.0

図 6　重み w によるスケルタルサーフェス形状の変化

スケルトン S は形状の骨格となるもので，点や線分，平面などを用いることができる．重み w はスカラー値であり，正負の値を取ることができる．また，$d(P, S_i)$ は点 P とスケルトン S_i 上の最も近い点との距離を表す．2つの点スケルトンから成る形状の，重み w による変化の例を図6に示す．図6（a）,（b）に示すように，重み w が正の値同士の形状を組み合わせた場合，一方の形状が他方の形状に付加される．また，図6（c）に示すように，重み w が正の値と負の値の形状を組み合わせた場合，正の値の形状が負の値の形状に切削される．距離場関数 F は典型的には図7に示されるような形状となる関数が使われる．制御点CPにおける傾きと影響範囲となる幅を制御することで形状の変形具合いを制御することができる．制御点CPの傾きによる形状の変化を図8に示す．傾きの絶対値が小さい場合，図8（a）のようにつなぎ目が緩やかになり，傾きの絶対値が大きい場合，図8（b）のようにつなぎ目が急になる．

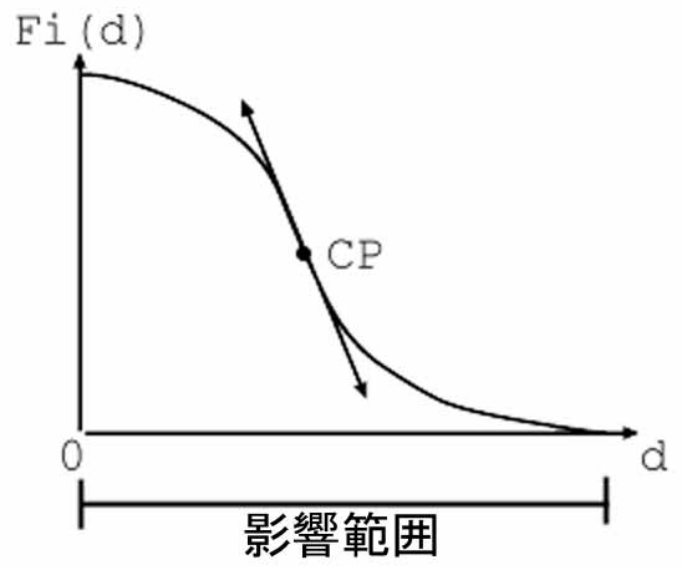
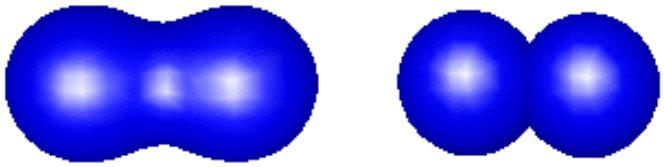


図 7　一般的な距離場関数の例



(a) 傾き＝−1.0　　(b) 傾き＝−57.3

図 8　制御点 CP における距離場関数の傾きの影響

2.3　手を用いた形状変形

前節で述べたスケルタルサーフェスの特徴を利用して，仮想粘土の変形を行う．本節では２つのモデリング操作を定義し，それぞれで行われる変形処理について述べる．

2.3.1　２つのモデリング操作：「凹ませる」,「つまみ出す」

本システムでは，形状に対して次の 2 つのモデリング操作を行うことができる．また，初期形状として球が用意される．

（１）形状を凹ませる

まず，手を図 9 の様に人差し指だけを伸ばした形にする．そして，形状に対して人差し指を接触させ押し込んでいくことで，粘土のように，人差し指部分の形状を凹ませていく．この時，接触部分の形状は指形状にあわせて変形する．また，陰関数曲面による形状は常に連続性を保つため，凹みを加えた部分の形状も滑らかである．

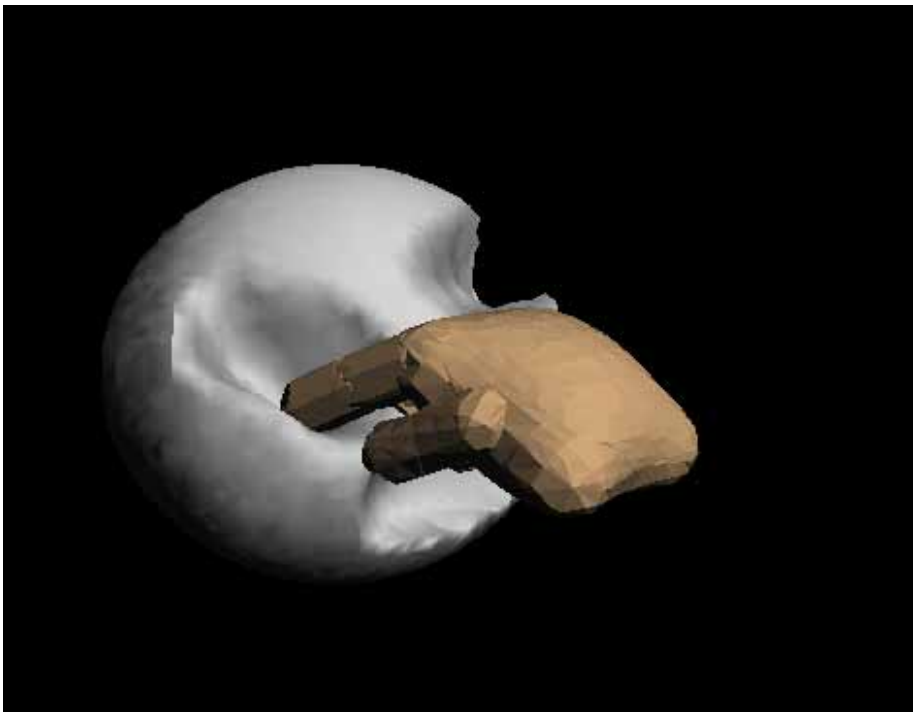


図 9 形状を凹ませた様子

（２）形状をつまみ出す

まず，親指と人差し指，中指を伸ばし，手を図 10 の様に形状をつまむ形にする．そして，手を移動させることで，形状をつまみ出す．実際には，手の動きに沿って 3 本の指の間に円柱状の形状が連続的に生成され，他の形状と接触する場合は滑らかに繋がるが，これをユーザが意識することはない．そのため，手を形状内部から外部へ動かすと次々と形状が生成され，ちょうど手で粘土をつまみ出すように変形させることができる．

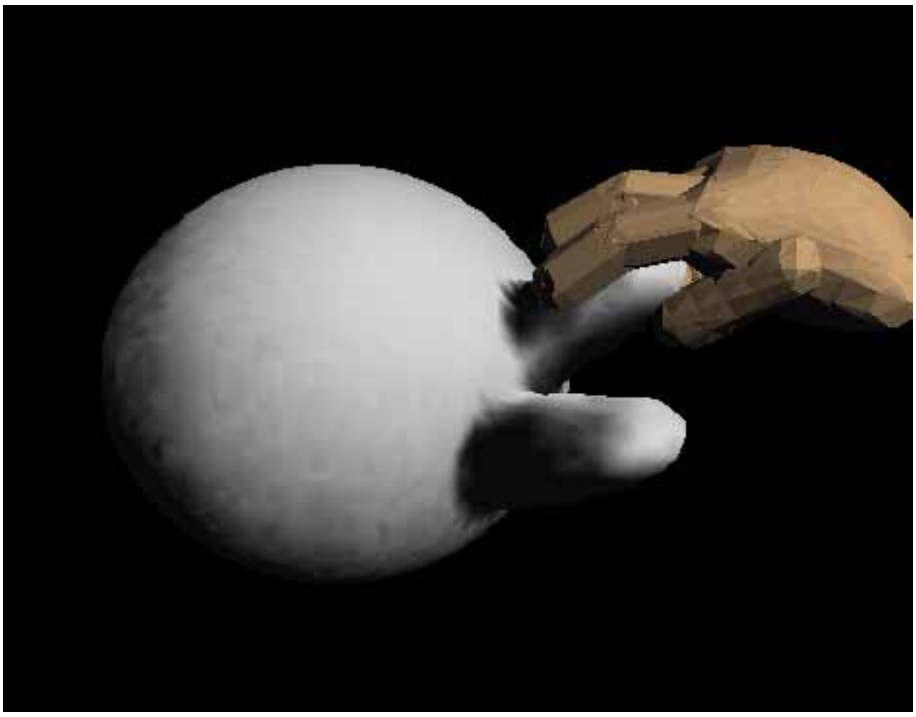


図 10　形状をつまみ出した様子

2.3.2　形状の変形処理

まず，本研究では前節で述べたスケルタルサーフェスの表現式（1）を次の式（2）のように改良して用いる．

$$f(P) = \sum_{i=1}^{n} w_i F_i(\sqrt{d(P, S_i)}). \tag{2}$$

このように式変形するのは，経験的に，表現式（1）のままでは指形状に沿って形状を凹ませることができず，指形状の周囲が広範囲に渡って凹んでしまうためである．形状をつまみ出す際にも，つまみ出した形状が広範囲に広がることを防ぐ．また，同様の理由から，距離場関数 F として式 (3) に示すエルミート (Hermite) 曲線 [47] による関数を用いる．式 (3) において，$H_i^3(d)(i = 0, 1, 2, 3)$ は式 (4) に示す 3 次エルミート関数であり，p_i，$v_i(i = 0, 1, 2)$ は図 11 に，距離 d に対する距離

場関数の変化の様子とともに示したベクトルである.

$$F(d) = \begin{cases} p_0 H_0^3(d) + v_0 H_1^3(d) \\ \qquad +v_1 H_2^3(d) + p_1 H_3^3(d); & (0 \le d < 1.0) \\ p_1 H_0^3(d-1.0) + v_1 H_1^3(d-1.0) \\ \qquad +v_2 H_2^3(d-1.0) + p_2 H_3^3(d-1.0); & (1.0 \le d \le 2.0) \end{cases} . \tag{3}$$

$$\begin{aligned} H_0^3(t) &= (2t+1)(1-t)^2, \\ H_1^3(t) &= t(1-t)^2, \\ H_2^3(t) &= t^2(t-1), \\ H_3^3(t) &= t^2(3-2t). \end{aligned} \tag{4}$$

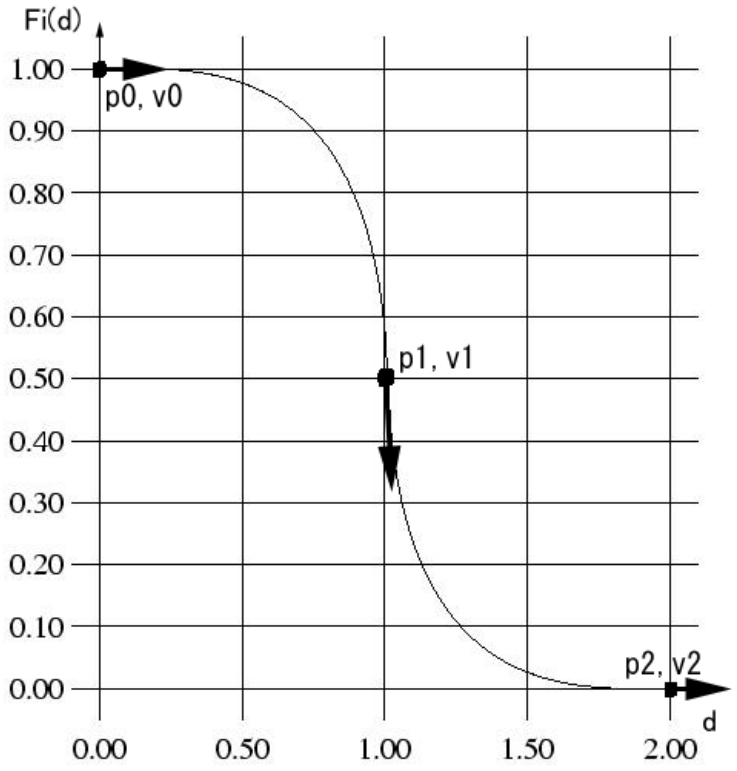


図 11 エルミート曲線による距離場関数

次に，初期形状が 3 つの点スケルトンから成る形状を例に，2.3.1 で述べた 2 つのモデリング操作による形状変形処理について説明する．初期形状 $f_1(P)$ は式（2）から，点スケルトンを $PS_i(i = 1 \ldots 3)$，重みを $w_1 = w_2 = w_3 = 1.0$ として次の式（5）で表される．

$$\begin{aligned} f_1(P) &= F(\sqrt{d(P, PS_1)}) \\ &\quad + F(\sqrt{d(P, PS_2)}) + F(\sqrt{d(P, PS_3)}). \end{aligned} \tag{5}$$

（1）形状を凹ませる

人差し指形状 $f_2(P)$ は線スケルトンで表現し，線スケルトンを LS として，式（6）で表される．

$$f_2(P) = w_1 F(\sqrt{d(P, LS)}). \tag{6}$$

指形状 $f_2(P)$ による初期形状 $f_1(P)$ の凹みは，図 12 に示すように，$f_1(P) - f_2(P)$ という減算によって表現することができ，変形後の形状 $f(P)$ は，指形状の重みを $w_1 = -1.0$ として式（7）のようになる．

$$\begin{aligned} f(P) &= f_1(P) - f_2(P) \\ &= F(\sqrt{d(P, PS_1)}) + F(\sqrt{d(P, PS_2)}) \\ &\quad + F(\sqrt{d(P, PS_3)}) - F(\sqrt{d(P, LS)}). \end{aligned} \tag{7}$$

ここで，手が一定距離移動するごとに，指のスケルトンの形や位置といった幾何情報と，減算という属性をスケルトンデータとして連続的に取得していく．スケルトンのサンプリング間隔は，経験的に，指のスケルトンの長さの 100 分の 1 にした．この間隔であれば，連続的に生成しても，表現式（2）と図 11 に示す距離場関数によって，指形状に沿った「形状を凹ませる」表現ができる．

（2）形状をつまみ出す

つまみ出される（付加）形状 $f_2(P)$ は，凹ませる場合と同様に線スケルトンで表現し，式（6）で表される．2.3.1 節で述べたように，付加形状は 3 本の指の

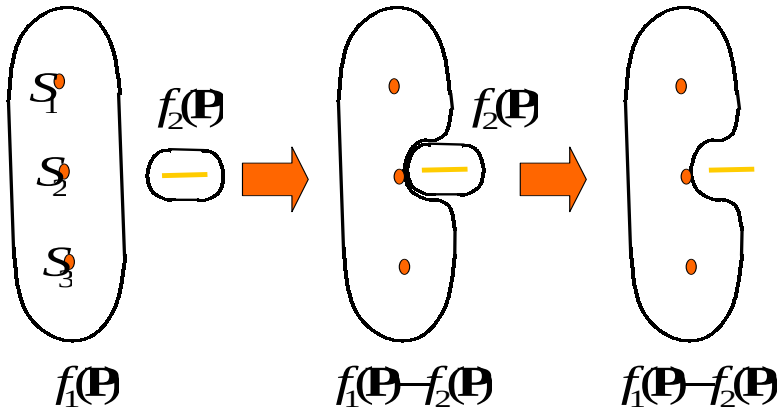


図 12　形状の凹みの表現

間に生成される．初期形状 $f_1(P)$ への形状 $f_2(P)$ の付加は，図 13 に示すように，$f_1(P) + f_2(P)$ という加算によって表現することができ，変形後の形状 $f(P)$ は，付加形状の重みを $w_1 = 1.0$ として式（8）のようになる．

$$\begin{aligned} f(P) &= f_1(P) + f_2(P) \\ &= F(\sqrt{d(P, PS_1)}) + F(\sqrt{d(P, PS_2)}) \\ &\quad + F(\sqrt{d(P, PS_3)}) + F(\sqrt{d(P, LS)}). \end{aligned} \tag{8}$$

ここで，手が一定距離移動するごとに，付加形状のスケルトンの形や位置といった幾何情報と，加算という属性をスケルトンデータとして連続的に取得していく．スケルトンのサンプリング間隔は，2.3.2 と同様，指のスケルトンの長さの 100 分の 1 にした．この間隔であれば，連続的に生成しても，表現式（2）と図 11 に示す距離場関数によって，細く滑らかに繋がった「形状をつまみ出す」表現ができる．

次節では，以上のようにして生成されたスケルトンデータによって表される，スケルタルサーフェスを実時間で描画する手法について述べる．

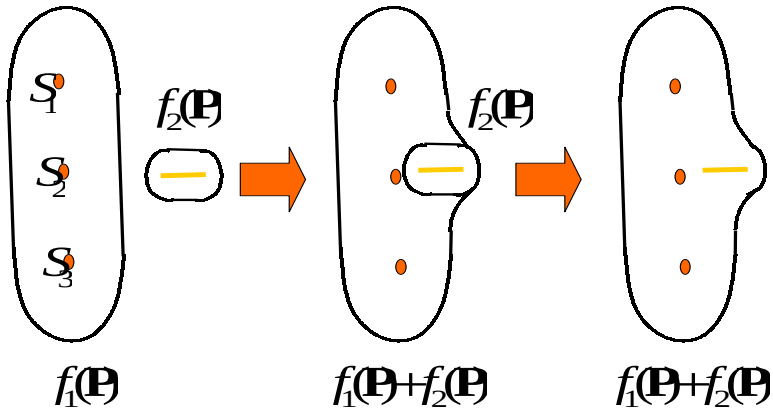


図 13　形状のつまみ出しの表現

2.4　対話操作のための描画手法と並列処理

2.4.1　陰関数曲面の描画手法

対話操作を行うためには，スケルタルサーフェスを用いた形状を実時間で描画してユーザに提示する必要がある．陰関数曲面では曲面上の任意の点での法線が容易に求まることから，その描画手法としてはレイトレーシング法を用いるのが一般的であるが [48, 49]，レイトレーシング法は実時間処理には向かない．そこで，本研究においては，陰関数曲面として表現された形状をポリゴンで近似し，ハードウェアによる高速なポリゴン描画手法を利用する．陰関数曲面を用いた形状をポリゴンに変換する手法としては，スケルトンから多方向に伸ばした直線と曲面との交点を頂点としてポリゴンを生成する Desbrun らの手法 [50] や，関数や密度ボリュームデータの等値面をポリゴン変換する手法 [51, 52] を基にした，ボクセルと曲面との交点を頂点としてポリゴンを生成する Bloomenthal らの手法 [53] などがある．しかし，これらの手法は実時間での対話的なモデリングを考慮したものではない．また，ボクセルを利用して，メタボールの実時間での対話的なモデリングを可能とした手法 [28] もある．この手法では，濃度値（本論文では陰関数値）の計算時に参照するメタボールを限定することによって処理を軽減している．しかし，参照するメタボールを判別する処理はメタボールの総数に依存した

処理量となっており，本システムのようにスケルトン（メタボールに相当）の数が多くなる場合には適用できない．そこで，Bloomenthal らの手法を基に，実時間での対話的なモデリングを可能にするためのポリゴン変換手法を提案する．以下に具体的なアルゴリズムを述べる．

まず，前処理として以下の処理を行い，モデリング作業時のための準備と初期形状の描画を行う．

［前処理］

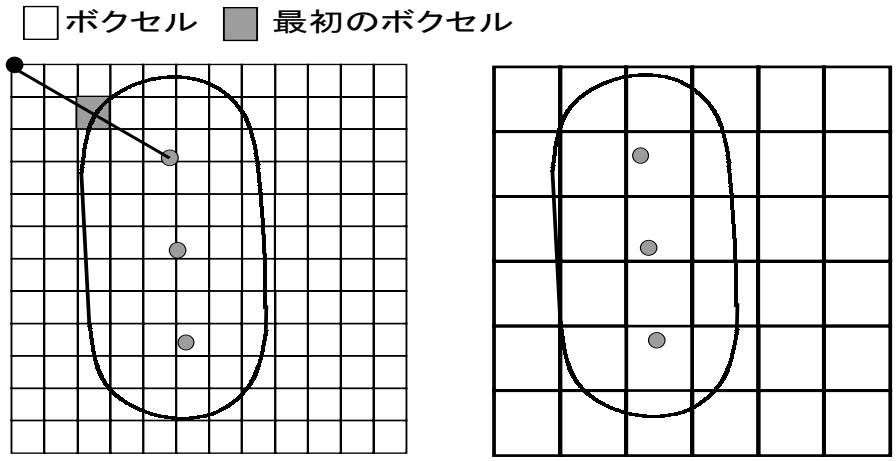
1. 距離場関数の d 軸を均等に 1000 箇所サンプリングし，配列に格納する．配列は

   ```
   double Hermite[1001];
   ```

   のようになる．後で距離場関数を計算する時には，

   ```
   double d; //距離
   double F = Hermite[(int)(1000*d)]; //関数値
   ```

   のように，配列の参照だけで高速に計算することができる．
2. 形状の周囲にポリゴン変換用ボクセル空間とスケルトンデータ用ボクセル空間を設定する．ポリゴン変換用ボクセル空間は曲面とボクセルの稜線との交差点を得るために使用する．また，スケルトンデータ用ボクセル空間はスケルトンデータの保持と陰関数値を求めるために使用する．２つのボクセル空間を簡単のため 2 次元で表したものを図 14 に示す．
3. あらかじめ決められた初期形状のスケルトンデータ群を，それぞれの位置にあるスケルトンデータ用ボクセルに格納する．
4. 以下の手順で初期形状の表面と交差するポリゴン変換用ボクセルを図 15 に示すように探索しながら，それぞれポリゴンへの変換を行う．ボクセルを用いたポリゴン変換処理については後述する．



(a) ポリゴン変換用ボクセル空間　　(b) スケルトンデータ用ボクセル空間

図 14　2 つのボクセル空間

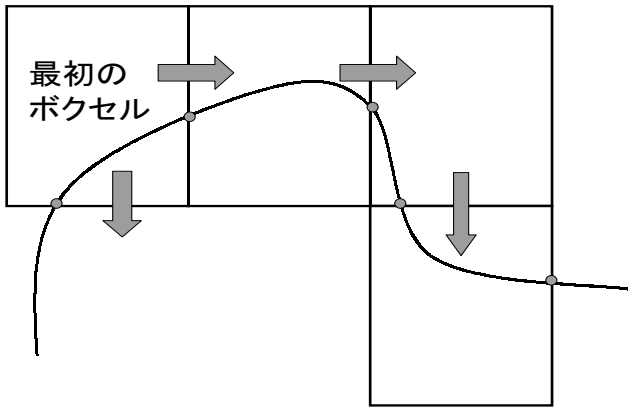


図 15　表面が存在するボクセルの探索

(a) 図 14（a）に示すように，初期形状を構成するスケルトンの内の 1 つの位置と，あらかじめ決められたポリゴン変換用ボクセル空間の端にある点の

位置とを用いた二分法によって，形状表面のあるポリゴン変換用ボクセルの 1 つを求める．

(b) 求めたボクセルに関するポリゴン変換処理を行う．

(c) ポリゴンの頂点が存在するボクセルの稜線を共有するボクセルを求める．初期形状は連続した 1 つの閉じた曲面なので，ポリゴンの頂点が存在するボクセルの稜線を共有するボクセルに曲面が存在する．

(d) (b)，(c) の処理を再帰的に繰り返し，すべての表面形状がポリゴンに変換されれば終了する．

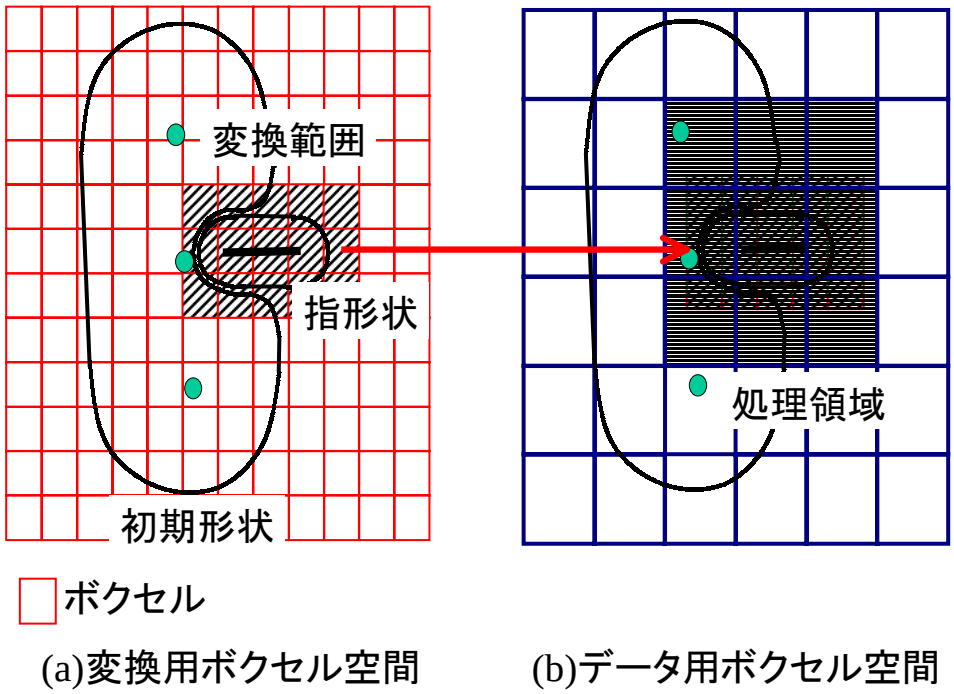


図 16　変換領域と処理領域

前処理の後，対話的に変形を行う時には以下の処理を実時間で行う．

［モデリング時のポリゴン変換処理］

1. 指の動きに従って生成されるスケルトンデータを，その位置にあるスケルトンデータ用ボクセルに格納する．
2. 図 16(a) に示すように，変換が必要になる指形状周囲の変換領域にあるポリゴン変換用ボクセルに対して，以下の処理を繰り返す．
   (a) ボクセルの頂点を 4 点づつ組み合わせて 6 個の 4 面体に分割する．このとき，図 17 に示すように，6 個の 4 面体それぞれが互いに隣接するように分割する．
   (b) 4 面体の各頂点座標における陰関数値を求める．この時，図 16(b) に示すように，変換領域と重なるスケルトンデータ用ボクセルに格納されているスケルトンデータを用いて計算する．
   (c) 図 18 に示すように，求まった陰関数値の符号からその頂点が形状の内部であるか外部であるかを判別し，それを属性として頂点の情報に付加する．

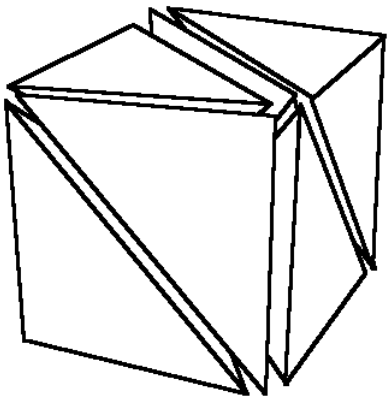


図 17 ボクセルの 6 つの 4 面体への分割

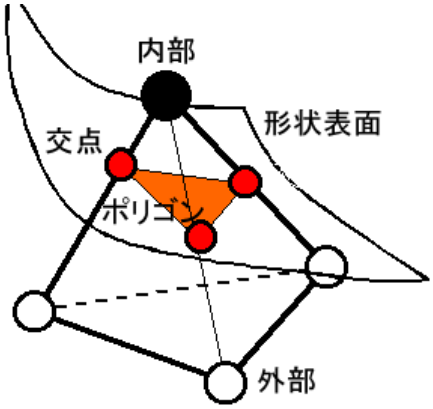


図 18　4 面体と形状表面との交点を用いたポリゴン生成

(d) どの稜線上に交点が存在し，それをどのように接続してポリゴンにするかを，図 19 に示す頂点の内外属性の組み合わせによって判別する．

(e) 稜線と形状との交点を求めるということは，稜線の端点間に含まれた陰関数式の根を求めることであり，これを非線形方程式の根を求める手法である二分法 [54] によって求める．

(f) 図 18 に示すように，求まった交点を基に，三角形ポリゴンを生成する．同時に，交点における法線を計算する．

以上のアルゴリズムに関して説明を加える．本システムでは，変形する領域は付加および切削形状の近傍のみであるので，システムの初期化時に前処理として 1 度全ボクセルをポリゴン変換し，システムの実行時には変換領域のみをポリゴン変換している．これによって，処理を軽減することができる．

また，2.2 で述べたように，スケルタルサーフェスでは，スケルトンごとに距離場関数の影響範囲があるため，変換領域が影響範囲内に入っているスケルトンデータのみを参照すれば良い．そのためにポリゴン変換用のボクセル空間とは別に，スケルトンデータ用のボクセル空間を設定し，スケルトンデータ用ボクセルごとにスケルトンデータを記憶している．ポリゴン変換時には，変換領域と重なるスケルトンデータ用ボクセル内のスケルトンデータのみを参照することによって，処理を軽減するとともに，演算量の増加を抑えている．また，ボクセルを利

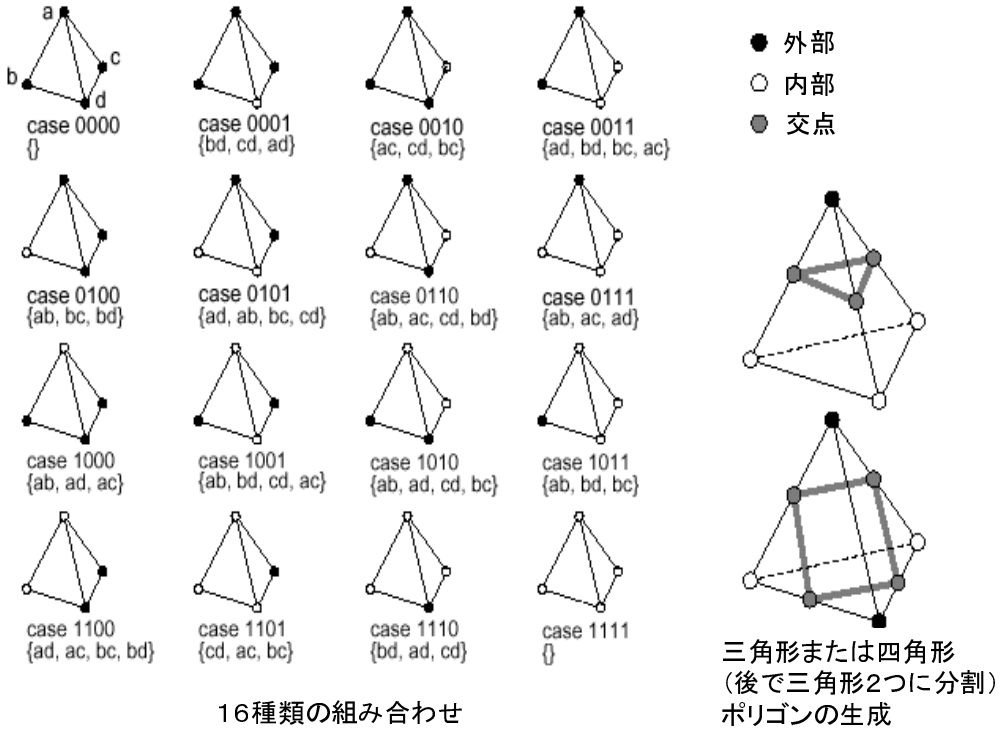


図 19 頂点の内外判定の組み合わせによるパターン分け

用しているため，参照するスケルトンデータを判別する処理が必要ない．

さらに，スケルトンデータの無駄な増加を防ぐために，1つのスケルトンデータ用ボクセル内に記憶できるスケルトンデータの最大数を設ける．最大数以上のスケルトンデータは生成できないため，最大数に達したボクセルの領域には，それ以上の変形を加えることができなくなる．このため，本システムにおいては，1）ポリゴン変換用ボクセルの大きさ，2）スケルトンデータ用ボクセルの大きさ，3）1ボクセルあたりのスケルトンデータの最大記憶数，をパラメータとして，これをユーザの希望するモデリングの複雑さと，計算機の性能に応じて設定することとした．

なお，以上の処理はボクセルごとに独立した処理であるため，これらを複数のCPUで並列に処理することが可能である．

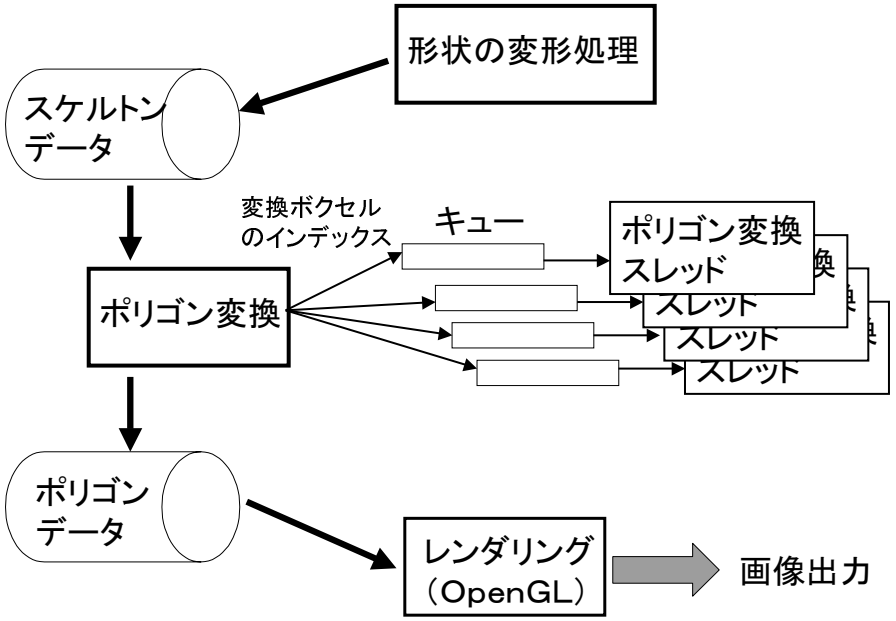


図 20　手を用いた変形処理におけるスレッドの関係

2.4.2　マルチスレッドプログラムの設計

ポリゴン変換処理を実時間で効率的に行うために，SMP(Symmetrical Multi Processor) 形式のマルチプロセッサマシン上でスレッドを用いて並列処理を行う．ポリゴン変換の処理効率を最適化するためには，処理のスレッド分散の仕方を考える必要がある．

ポリゴン変換処理を含めた処理の流れを図 20 に示す．まず，形状の変形処理部で，前述したスケルトンデータの生成を行う．そのスケルトンデータによる陰関数曲面をポリゴン変換部でポリゴン表現に変換する．そして，このポリゴンデータを OpenGL グラフィクスライブラリを用いて，ハードウェアにより高速に描画する．

ポリゴン変換時には，互いに独立したポリゴン変換ボクセルごとの処理をスレッ

ドを用いて並列に処理する．ここでは，マルチスレッドプログラミングにおける，スレッドプールを使ったボス・ワーカモデル [55] を基にしている．形状の変形処理部，ポリゴン変換部，レンダリング部の一連の処理は 1 つのスレッドということになる．ポリゴン変換スレッドは，プロセスの開始直後に生成され，プロセスが消滅するまで存在する．このようにすることで，スレッド生成によるオーバーヘッドを無くすことができる．ここで，ポリゴン変換スレッドの生成数は，ポリゴン変換に用いるボクセルの細かさと，CPU の性能に応じて決定する．

ポリゴン変換部で変換の必要なボクセルの算出を行い，そのボクセルのインデックスを，図 20 に示すように，それぞれのポリゴン変換スレッドが持つキューに登録する．ポリゴン変換スレッドはキューから読み出したインデックスのボクセルに関するポリゴン変換処理を行う．ここで，キューを各ポリゴン変換スレッドごとに存在させることで，ポリゴン変換スレッド間の同期を取る必要がなくなり，同期による待ち時間のオーバーヘッドを低減できる．オーバーヘッドが発生するのは，ポリゴン変換部を含むスレッドと各ポリゴン変換スレッドの間でキューの排他制御による待ちが発生した場合だけになる．

2.5 試作システムによるモデリング実験

提案手法を実装し試作したシステムを用いてモデリング実験を行った．試作システムでは，仮想空間にあらかじめ用意された球の初期形状に対し，HMD（Head Mounted Display）を装着し仮想空間に没入したユーザが，手形状入力装置を装着した手で変形を加えることによってモデリングを行う．HMD と手形状入力装置には 3 次元位置センサが取り付けられており，頭部と手の位置および姿勢を計測する．システムを使用している様子を図 21 に，システム構成を図 22 に示す．なお，計算機として SGI 社 ONYX2（MIPS R10000, 195MHz, 16CPU）を用い，HMD としてオリンパス社 Mediamask を，3 次元位置センサとして Polhemus 社 3SPACE Fastrak を，手形状入力装置として日商エレクトロニクス社 Super Glove をそれぞれ接続して使用した．また，実装は C++言語を用い，マルチスレッドライブラリには Pthreads を使用した．

試作システムによるモデリングの例として，ティーカップ，水槽の中の魚，た

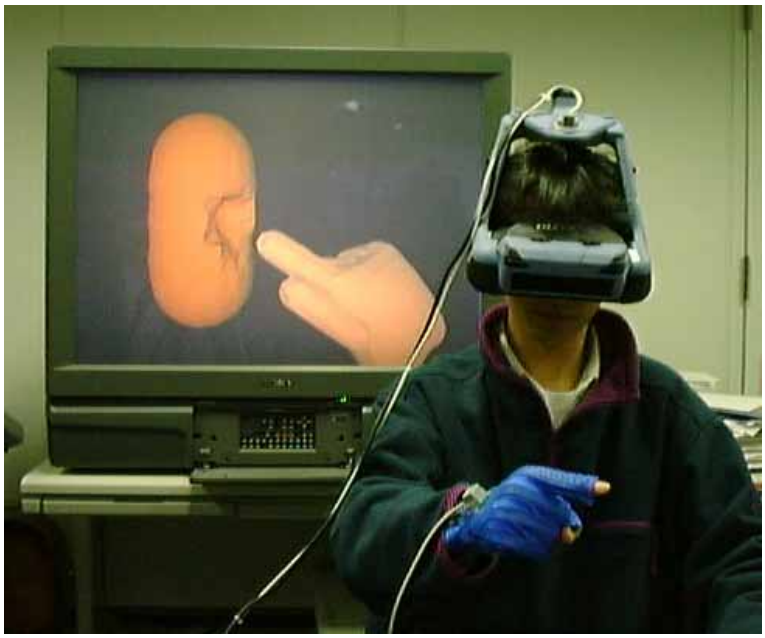


図 21 手を用いた仮想粘土細工の試作システムの様子

こ，鳥の頭を作成したものを図 23，図 24，図 25，図 26 にそれぞれ示す．図 23 のティーカップのモデリング手順について説明する．まず，図 27(a) のように初期形状が用意されている．図 27(b) で示すように,「形状を凹ませる」操作を用いて初期形状の上部を指で凹ませ，さらに飲み口の部分と中に入っている飲み物をモデリングした．その後，図 27(c) で示すように,「形状をつまみ出す」操作で円弧状に手を動かし，取っ手の部分を付け加えた．さらに，細かな調整を加えて，図 23 のようなティーカップが完成した．図 24 の水槽の中の魚は，初期形状につまみ出しで目やひれを付け，凹ませて口を開けた．そして，つまみ出しで海草をモデリングした．他の形状も同様に作成し，いづれの形状も 5 分程度で作成できた．このように，試作システムによって滑らかな曲面を持った自由形状を容易な操作で短時間にモデリングすることができる．

なお，モデリング作業時の画像更新レートは，CPU をポリゴン変換処理に 3 つ，その他の処理に 1 つ用いて，約 10 fps である．この時，ポリゴン変換用ボクセルデータの大きさは $120 \times 120 \times 120$，陰関数データ用ボクセルデータの大きさは

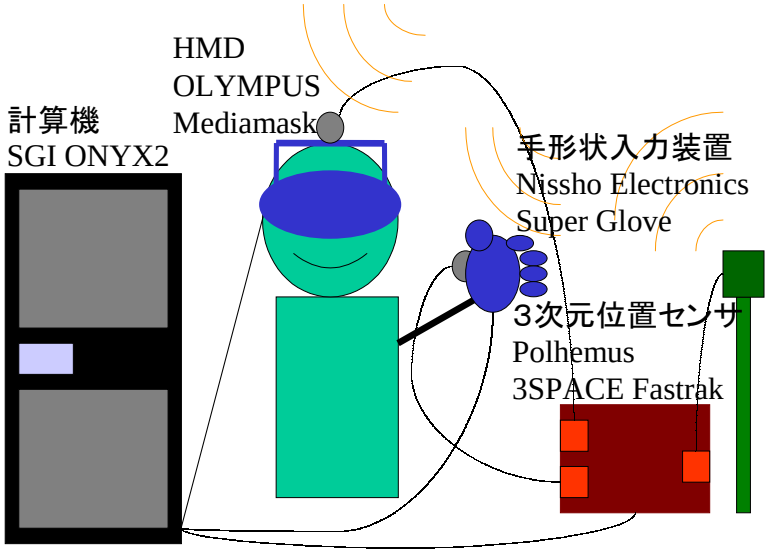


図 22 手を用いた仮想粘土細工の試作システム構成

$48 \times 48 \times 48$，1 ボクセルあたりの陰関数データの最大記憶量は 10 である．

ここで，図 23，24 はシステムの実行時の画面から切り出した画像である．本システムは，陰関数曲面を直接レイトレーシング法によってレンダリングした画像や，DXF[56] や STL[57] といったポリゴンデータ形式で出力できる．これらについては，付録 A および，付録 B に詳述する．図 25，26 はレイトレーシング法による画像である．また，図 28 は，たこモデルを DXF で出力し，市販の 3 次元 CG ソフトウェアに利用した例である．図 29 は STL で出力し，光造形装置（三井造船，COLAMM-300）によって実物体として生成した例である．

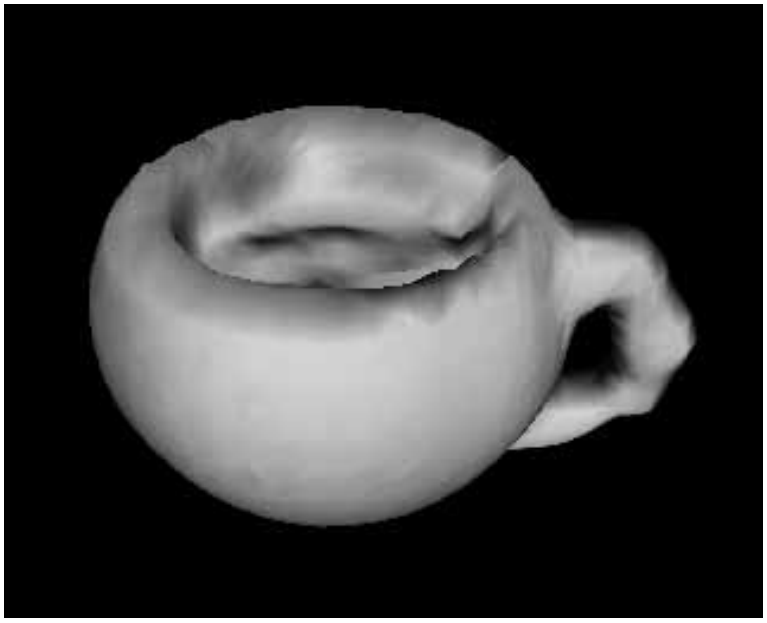


図 23 モデリング例：ティーカップ

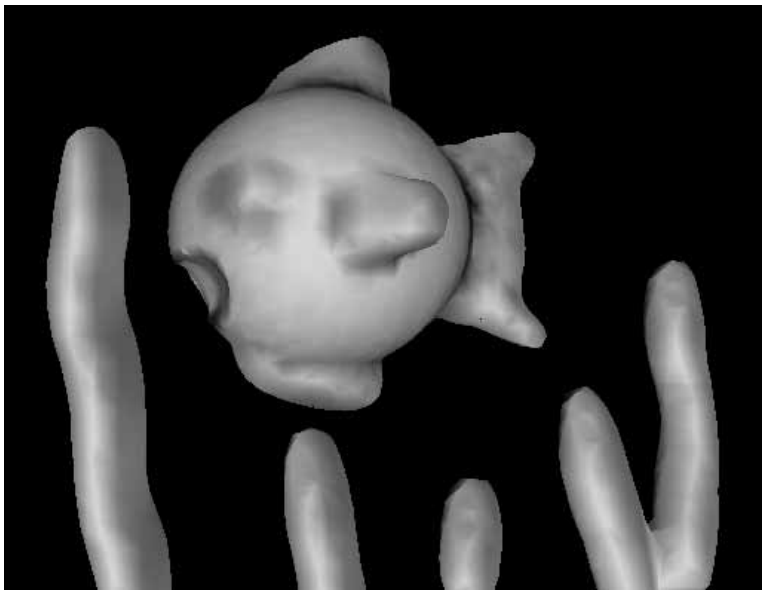


図 24 モデリング例：水槽の中の魚

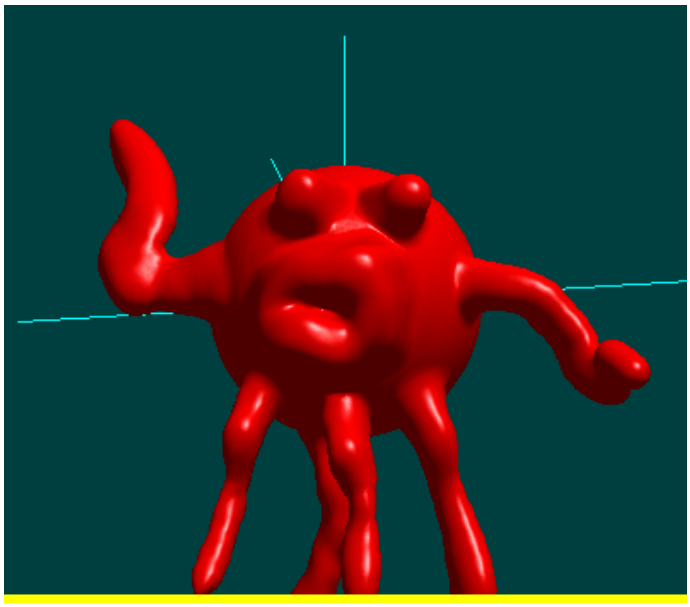


図 25　レイトレーシング法による画像：たこ

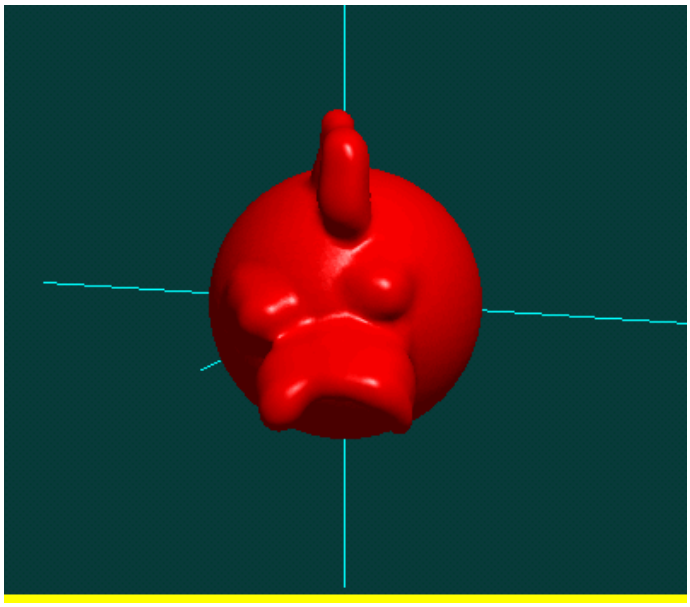
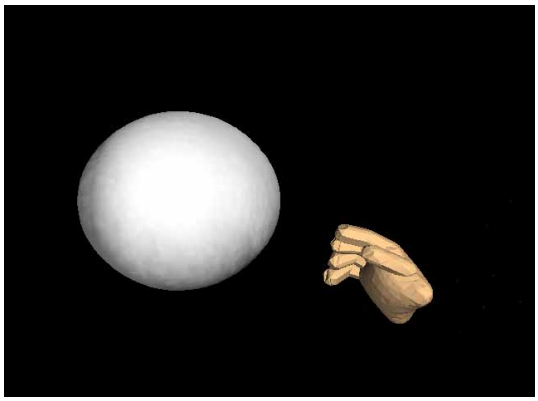
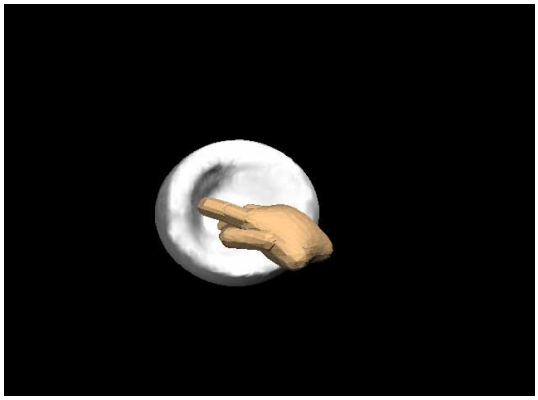


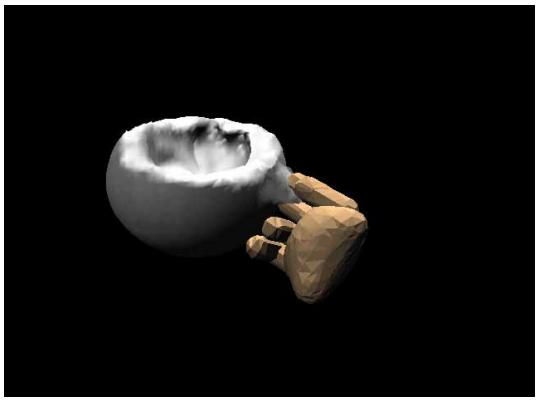
図 26　レイトレーシング法による画像：鳥の頭



(a)



(b)



(c)

図 27 ティーカップのモデリング過程

図 28　たこモデルと市販ソフトウェアを利用して描画した画像

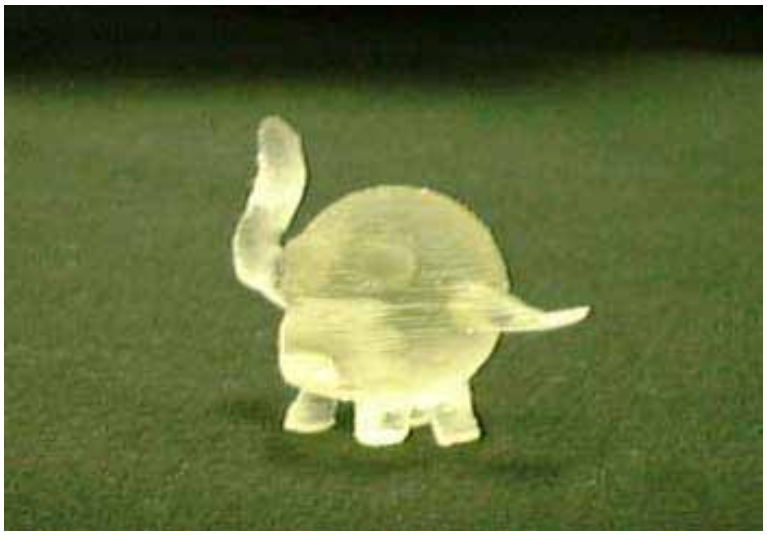


図 29　たこモデルと光造形装置を利用して生成した実物体

2.6 結言

本章では，粘土細工を行うような感覚で手を用いて直感的かつ容易に自由形状のモデリングを行うための手法について述べた．現実空間での粘土細工のような操作と形状の表現を，仮想現実感技術と陰関数曲面による形状の表現を用いることで実現した．また，試作システムによるモデリング例を通して，実際に自由形状を短時間でモデリングすることができることを確認した．試作システムでのモデリングは精密さや正確さに欠けるため，精密な工業製品などを正確にモデリングするような用途には向かないが，設計の初期モデルのモデリングや，2.5 節で示したような正確さが求められないデザイン作成などの用途に有用である．

本手法では，体積保存などのような粘土らしい変形過程に関しては考慮していない．これは粘土細工において重要な要素であるため，次章では，粘土らしい変形過程の表現手法に関して述べる．

3. 仮想粘土モデルによる粘土の変形挙動の表現

3.1 緒言

前章で述べた手法では，粘土らしい変形挙動の表現については考慮していなかった．しかし，現実の粘土細工で行われる，形状を全体的に変形させるなどのような操作を実現するためには，粘土の変形挙動を表現する必要が生じる．本章では，粘土の変形挙動を表現するとともに，それを対話的に行うことが可能な仮想粘土モデルについて述べる [58, 59, 60, 61]．

ここで，粘土というものについて考えてみる．粘土はビンガム塑性流体に分類される物体である．ビンガム塑性流体は以下のような過程で塑性変形する．まず，静止状態では，ある応力（降伏応力）以下では流動に抵抗する 3 次元的な構造を持っている．応力が降伏応力を超えると構造が壊れて流動が起き，変形する．そして，応力が降伏応力以下になると，再び構造が構築される [62]．また，この過程において体積は一定である．本研究では，このような物体を，パーティクルシステムと陰関数曲面を用いて表現する．

パーティクルシステムとは，互いに力を及ぼしあう多数の粒子（パーティクル）を，ある運動方程式に基づいて移動させることによって，全体としてある物理現象のシミュレーションを行うものである [63]．このパーティクルを粘土の粒子とみなして，前述した変形過程に従うように運動させる．ここで，分子動力学のように全ての粘土粒子のシミュレーションを行って現実の挙動を得ることも可能であるが，計算量が多いことから対話的な操作には向かない [64]．本研究では，パーティクルをある程度の粒子の塊と考え，少ないパーティクル数で粘土の変形を表現する．そして，陰関数曲面を用いて，全てのパーティクルを内包するような形状を生成し，粘土の形状を表現する．

3.2 パーティクルシステム

本研究では，パーティクル間が安定距離にあるときに力 $f = 0$，安定距離よりも近づいているときに $f > 0$（斥力），離れているときに $f < 0$（引力）となるよ

うな，パーティクル間の相互作用力を用いる．このような相互作用力 f は，経験的に式（9）および図 30 で示される 3 次多項式で与える．

$$f(w) = \begin{cases} 30.56w^3 - 50.56w^2 + 20.0; & (0.0 \leq w \leq 1.2) \\ 0; & (w > 1.2) \end{cases}. \tag{9}$$

ここで，パラメータ w はパーティクル間距離 r と安定距離 r_0 の比であり，$w = r/r_0$ で与える．

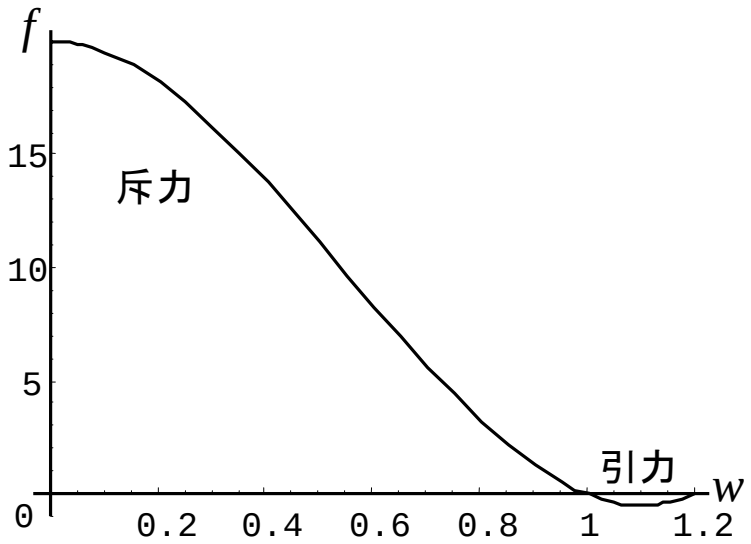


図 30　パーティクル間の相互作用力

そして，パーティクル i は次の運動方程式に従って移動する．

$$\boldsymbol{F}_i = m_i \frac{d^2 \boldsymbol{x}_i}{dt^2} \qquad (\boldsymbol{F}_i > \boldsymbol{F}_{th}). \tag{10}$$

ここで，$\boldsymbol{F}_i$ はパーティクル i に働く相互作用力の総和，m_i はパーティクルの質量，$\boldsymbol{x}_i$ はパーティクルの位置である．この運動方程式をオイラー法 [64] による数値計算によって解くことで，パーティクルを移動させる．このとき，パーティクルはある閾値 $\boldsymbol{F}_{th}$ 以上の力が加わらなければ運動しないとする．

3.3 仮想粘土モデル

パーティクルシステムにおいて，初期配置として図 31(a) のように，互いにほぼ安定距離にあるパーティクルを格子状に配置する．変形操作時には図 31(b) のように，パーティクルは互いの距離が安定距離を保つように運動するため，体積をほぼ一定に保った粘土らしい変形を表現できる．また，パーティクルの運動に対する閾値処理は降伏応力に対応している．これによって，より粘土らしい変形の表現ができるとともに，数値計算における収束を早めつつ，発散を抑えることができる．

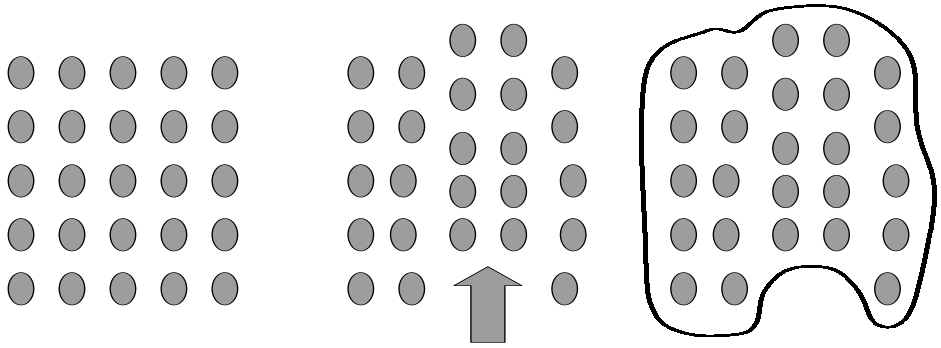


図 31 パーティクルと陰関数曲面による粘土表現

このパーティクルをスケルトンとみなして，前章と同様の手法で陰関数曲面を生成することで，図 31(c) に示すように滑らかな粘土の表面形状が表現できる．仮想粘土モデルでは，式（11）に示すスケルタルサーフェスを用いる．

$$\begin{aligned} &\{P \in R^3 | g(P) = c\} \\ &where \qquad g(P) = \sum_{i=1}^{n} G(d(P, S_i)). \end{aligned} \tag{11}$$

スケルトン S_i は形状の骨格となるもので，点や線分，平面などを用いることができるが，仮想粘土モデルにおいては点のみを用いる．$d(P, S_i)$ は点 P とス

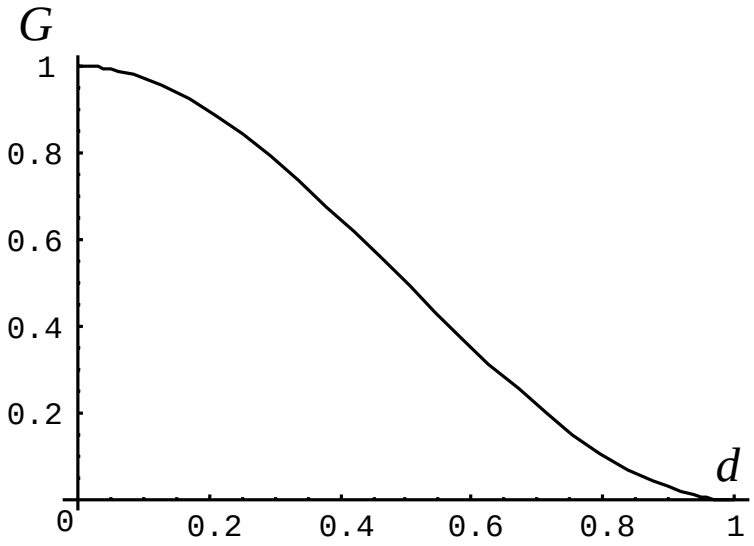


図 32 仮想粘土モデルの距離場関数

ケルトン S_i 上の最も近い点との距離 t と，前節で述べた安定距離 r_0 の比であり，$d = t/r_0$ で与える．また，距離場関数 G として式（12）および図 32 の関数を用いる．

$$G(d) = \begin{cases} 2.0d^3 - 3.0d^2 + 1.0; & (0.0 \leq d \leq 1.0) \\ 0; & (d > 1.0) \end{cases} . \tag{12}$$

次に，仮想粘土モデルに対話的変形を加える手をモデル化する必要がある．図 33 に示すように，指の形状の内部に線分を定義し，この線分を前述のパーティクルシミュレーションに組み込む．この時，各パーティクルと線分との距離に基づいて，相互作用力を式（13）および図 35 の関数を用いて求める．この関数は，指形状に合わせて粘土の形状表面が変形するように，経験的に決定したものである．手の形状には，予めモデリングされたポリゴン表現による 3 次元形状を用いている．

また，手のひら全体を使って形状を押し変形する操作をモデル化する．全ての

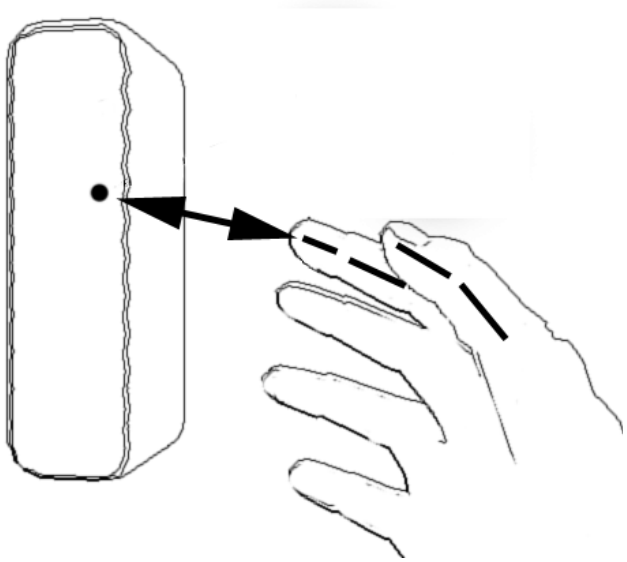


図 33　線分による指のモデル化

指を伸ばした状態の間は，前述した線分を用いずに，図 34 に示すように，手のひらを平面としてパーティクルシミュレーションに組み込む．この時，各パーティクルと平面との距離に基づいて，相互作用力を式（13）および図 35 の関数を用いて求める．

なお，これら 2 つのモデルの切り替えは，親指以外の指の付け根の関節が一つでも 20 度以上曲がっている時は線分モデルを用い，それ以外の時は平面モデルを用いる．

$$h(v) \quad = \quad \begin{cases} 51.5v^3 - 85.25v^2 + 33.75; & (0.0 \leq v \leq 1.2) \\ 0; & (v > 1.2) \end{cases} . \qquad (13)$$

ここで，パラメータ v は線分-パーティクル間距離 s と，指形状と粘土形状を考慮して求めた安定距離 s_0 の比であり，$v = s/s_0$ で与える．

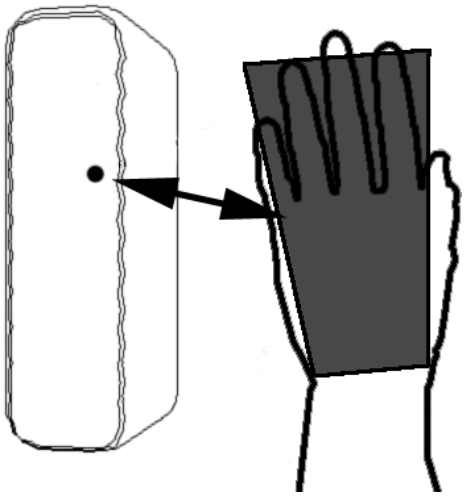


図 34 平面による手のひらのモデル化

3.4 対話操作のための描画手法と並列処理

対話的操作を行うためには，前節で述べた仮想粘土モデルを高速に描画してユーザに提示する必要がある．本章では，2.4 節で述べた高速描画手法を仮想粘土モデルに適用した手法について述べる．また，仮想粘土モデルの変形処理や描画処理などのすべての処理を効率的かつ高速に行うために用いた，対称型マルチプロセッサシステム上でのマルチスレッドプログラミングの設計について述べる．

3.4.1 仮想粘土モデルの描画手法

仮想粘土モデルの高速な描画を行うために，仮想粘土モデルで用いている陰関数曲面をポリゴン表現に変換する．そして，ポリゴンをハードウェアを用いて高速に描画する．変換手法は前章で述べた変換手法と類似のものであるが，以下では，仮想粘土モデルに用いている陰関数曲面をポリゴン表現に変換する手法について述べる．

まず，前処理として以下の処理を行い，モデリング作業時のための準備と初期形状の描画を行う．

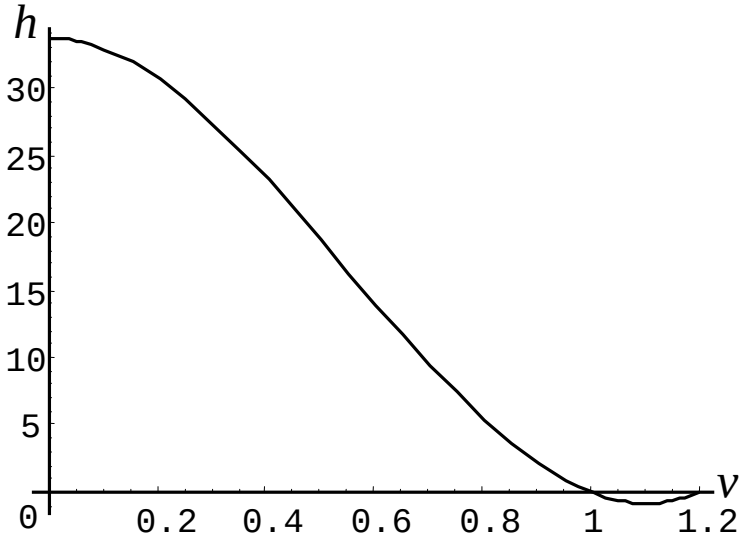


図 35 線分-パーティクル間の相互作用力

［前処理］

1. 陰関数曲面の周囲にボクセル空間を設定する．ボクセル空間は曲面とボクセルの稜線との交差点を得るために使用する．ボクセル空間を簡単のため 2 次元で表したものを図 36 に示す.

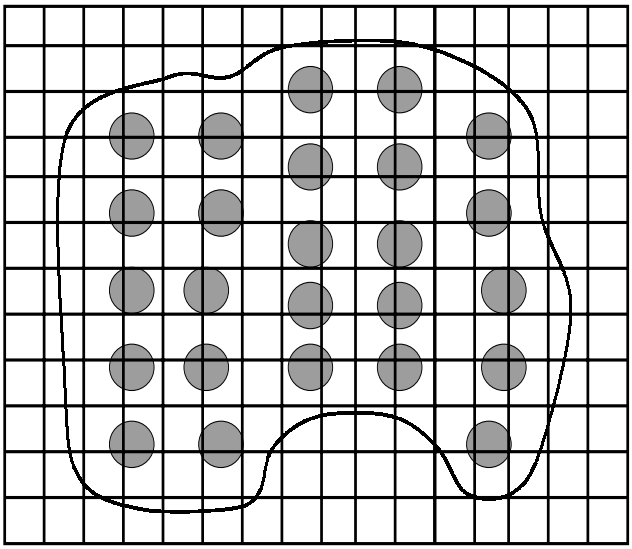


図 36 ボクセル空間

2. すべてのパーティクルの位置に存在するボクセルを中心とした周囲のボクセルを用いてポリゴン表現に変換する．ボクセルを用いたポリゴン変換処理については後述する．

前処理の後，対話的に変形を行う時には以下の処理を実時間で行う．

［モデリング時のポリゴン変換処理］

1. パーティクルの位置情報を調べ，前回の計算時から変更があるパーティクルを求める．
2. 図 37 の網掛け部分に示すように，求めたパーティクルの位置に存在するボクセルを中心とした周囲のボクセルに限り，以下の処理を行う．

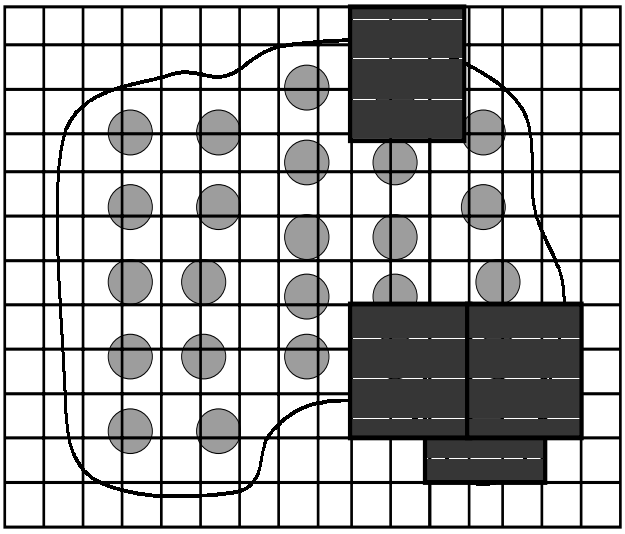


図 37 移動パーティクルの周囲のボクセルのみを変換

(a) ボクセルの頂点を 4 点づつ組み合わせて 6 個の 4 面体に分割する．このとき図 17 に示すように，6 個の 4 面体それぞれが互いに隣接するように分割する．

(b) 4 面体の各頂点座標における陰関数値を求める．

(c) 図 18 に示すように，求められた陰関数値の符号からその頂点が形状の内部であるか外部であるかが判別できるので，それを属性として頂点の情報に付加する．

(d) どの稜線上に交点が存在し，それをどのように接続してポリゴンにするかを，図 19 に示した頂点の内外属性の組み合わせによって判別する．

(e) 交点は陰関数が線形であると仮定し，両端点の陰関数値から式（14）を用いて交点の座標を求める．図 38 で示すように，端点 $\boldsymbol{P}_1$ での陰関数値を f_1，端点 $\boldsymbol{P}_2$ での陰関数値を f_2 としたとき，陰関数曲面とボクセルの稜線との交点 $\boldsymbol{P}$ を求める．

$$\boldsymbol{P} = \boldsymbol{P}_1 + \frac{f_1}{f_1 + f_2}(\boldsymbol{P}_2 - \boldsymbol{P}_1) \tag{14}$$

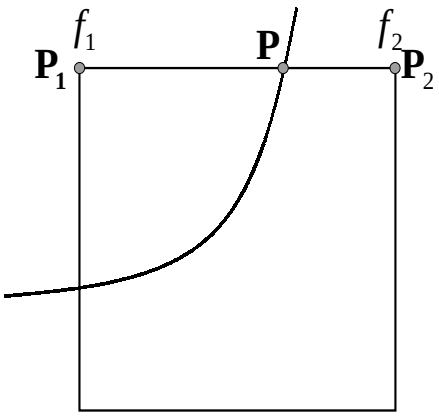


図 38　線形近似で交点座標を求める

(f) 図 18 に示したように，求められた交点を基に，三角形ポリゴンを生成する．同時に，求められた交点における法線を中心差分を用いて計算する．

ここで，ボクセルの細かさは計算機の性能に応じて設定する．また，移動したパーティクルの周囲何個のボクセルを処理するかは，ボクセルの細かさに応じて，形状が正しく表現されるように設定する．また，以上の処理はボクセルごとに独立した処理であるため，これらを複数のスレッドで並列に処理する．

3.4.2　マルチスレッドプログラムの設計

これまでに述べた仮想粘土モデルに関する処理を実時間で効率的に行うために，SMP(Symmetrical Multi Processor) 形式のマルチプロセッサマシン上でスレッドを用いて並列処理を行う．仮想粘土モデルに関する処理は，前章で述べた手法の並列処理に比べ，より多くの処理を並列に行う必要がある．このようなマルチスレッドプログラムでは，以下のようなオーバーヘッドを考慮する必要がある [55, 65, 66, 67]．

- オペレーティングシステムやライブラリが各スレッドを管理するために生じるオーバーヘッド．例えば，スレッドの生成処理などがある．

- 複数のスレッドは，プロセスに存在するすべての資源を共有しているため，複数のスレッドが同一の共有資源にアクセスする場合，その同期をとる必要が生じる．この時，ロックなどの同期機能を用いるが，この同期機能の呼び出しによるオーバーヘッド．

- 同期によって，あるスレッドが別のスレッドを待つ場合，その待ち時間というオーバーヘッド．

そこで，全体の処理効率を最適化するためには，処理のスレッド分散の仕方や，共有資源へのアクセス方法を考え，以上のようなオーバーヘッドを低減する必要がある．

スレッドの関係を図 39 に示す．まず，全体の処理をその役割ごとに大別し，それぞれにスレッドを以下のように割り当てる．1) パーティクルの運動計算を行うパーティクルシステム制御スレッド，2) 陰関数曲面からポリゴンへの変換を行うポリゴン変換制御スレッド，3) ポリゴンデータを OpenGL グラフィクスライブラリとハードウェアによって高速に描画するポリゴン描画スレッド，の 3 つのスレッドで並列に処理を行う．これらのスレッドは，プロセス開始直後に生成され，プロセス終了まで存在する．

これら 3 つのスレッド間の同期について考える．共有データとなるのは，個々のパーティクルの位置を保持するパーティクルデータベースと，ポリゴンの幾何情報を保持するポリゴンデータベースである．それぞれのデータベースへのアクセスは排他制御される．

パーティクルシステム制御スレッドは，非共有メモリにあるパーティクルの位置・速度情報を用いて，オイラー法による数値計算を 1 ステップ分行う．そして，パーティクルデータベースのロックを取得し，非共有メモリにある全てのパーティクルの位置情報をまとめてデータベースへ複製し，ロックを開放する．以上により，同期機能の呼び出しに関するオーバーヘッドと同期の待ちによるオーバーヘッドとのバランスが取れた，効率の良い処理が行える．

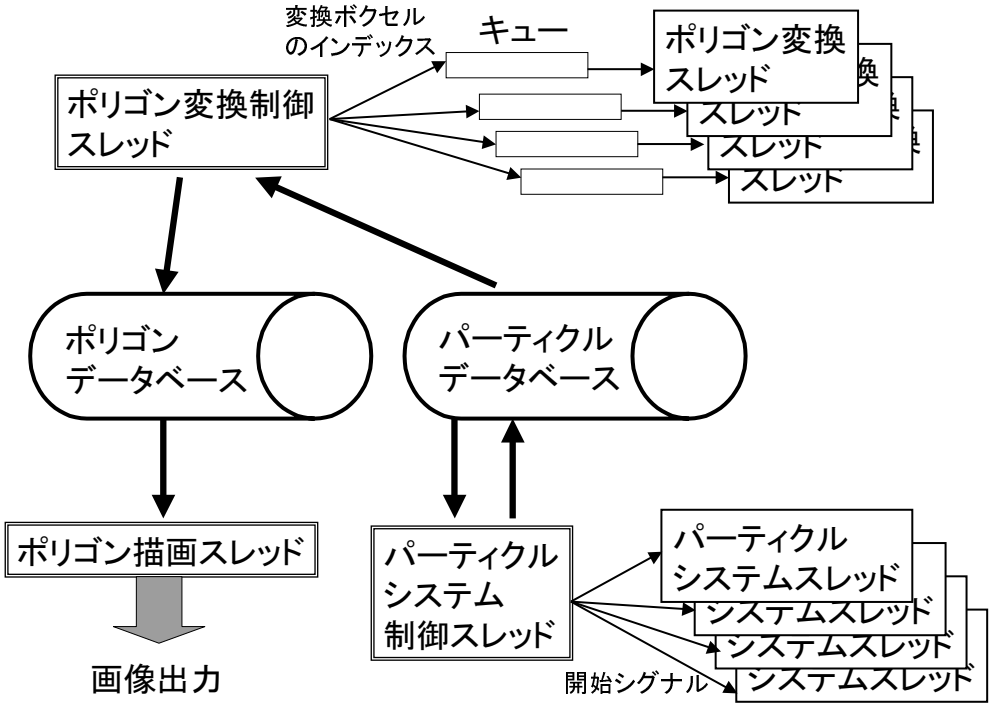


図 39 仮想粘土モデルにおけるスレッドの関係

ポリゴン変換制御スレッドは，まず，パーティクルデータベースから全てのパーティクルの位置を読み出し，非共有メモリに保存する．次に，その位置をスケルトンの位置として生成される陰関数曲面をポリゴンの幾何情報へ変換し，非共有メモリに保存する．最後に，非共有メモリ上のポリゴンの幾何情報をポリゴンデータベースへ登録し，ポリゴンデータベースの更新フラグをセットする．以上により，同期機能の呼び出しに関するオーバーヘッドと同期の待ちによるオーバーヘッドとのバランスが取れた，効率の良い処理が行える．

ポリゴン描画スレッドはポリゴンデータベースからポリゴンの幾何情報を読み出し，描画を行う．この時，まず，データベースの更新フラグを調べ，セットされていれば，ポリゴンの幾何情報を全て読み出し，非共有メモリに保存した後，更新フラグをリセットする．更新フラグがセットされていなければ，読み出しは行

わない．画像の更新レートに比べ，ポリゴン変換制御スレッドによるデータベースの更新レートの方が遅い．そこで，更新のない間は，非共有メモリに保存したポリゴンの幾何情報を用いて描画を行うことで，更新レートのずれを吸収するとともに，同期による待ち時間のオーバーヘッドを減少することができる．

次に，ポリゴン変換スレッドとパーティクルシステムスレッドについて述べる．ここでは，それぞれ，マルチスレッドプログラミングにおける，スレッドプールを使ったボス・ワーカモデルとピアモデル [55] を基にしている．2つのスレッドは，ポリゴン変換制御スレッドとパーティクルシステム制御スレッドの2つの制御スレッドの開始直後に生成され，それぞれの制御スレッドが消滅するまで存在する．このようにすることで，スレッド生成によるオーバーヘッドを無くすことができる．ここで，ポリゴン変換スレッドの生成数は，ポリゴン変換に用いるボクセルの細かさと，CPU の性能に応じて決定する．また，パーティクルシステムスレッドの生成数は，パーティクルや線分の数と，CPU の性能に応じて決定する．

ポリゴン変換制御スレッドは，変換の必要なボクセルの算出を行い，そのボクセルのインデックスを，図39に示すように，それぞれのポリゴン変換スレッドが持つキューに登録する．ポリゴン変換スレッドはキューから読み出したインデックスのボクセルに関するポリゴン変換処理を行う．ここで，キューを各ポリゴン変換スレッドごとに存在させることで，ポリゴン変換スレッド間の同期を取る必要がなくなり，同期による待ち時間のオーバーヘッドを低減できる．オーバーヘッドが発生するのは，ポリゴン変換制御スレッドと各ポリゴン変換スレッドの間でキューの排他制御による待ちが発生した場合だけになる．

パーティクルシステム制御スレッドは，運動計算の1サイクルごとに，各パーティクルシステムスレッドに計算開始を要求するためのシグナル送信を行う．各パーティクルシステムスレッドはシグナルを受け取ると，あらかじめ均等に割り当てられたパーティクルの組み合わせに関する相互作用力を計算し，パーティクルの位置と速度を計算する．パーティクルシステム制御スレッドは，各パーティクルシステムスレッドの計算が終了するのを待ち，終了後データベースの更新を行う．

図 40　仮想粘土モデルの変形挙動の確認実験の様子

3.5　試作システムによる変形挙動の確認

仮想粘土モデルを手を用いて対話的に変形できることを確認するために，前節までに述べた手法を実装し行った，仮想粘土モデルの対話的変形の実験について述べる．

実験は，計算機に SGI 社の ONYX2（MIPS R10000，195MHz，16CPU）を用い，図 40 に示すように，手形状入力装置である Virtual Technologies 社の CyberGlove と，3 次元位置センサである Polhemus 社 3SPACE Fastrak を接続した環境で行った．また，実装は C++言語を用い，マルチスレッドライブラリには Pthreads を使用した．

仮想粘土モデルには図 41 に示すように 216 個のパーティクルを用いた．また，3.2 節で述べた安定距離には $r_0 = 1.0$，閾値には $\boldsymbol{F}_{th} = 0.8$，3.3 節で述べた関数

値には $c = 0.5$，安定距離には $s_0 = 1.0$ を用いた．3.4.1 項で述べたポリゴン変換に用いたボクセルの大きさは一辺が 0.5 であり，移動したパーティクルを中心として周囲 $7 \times 7 \times 7$ 個のボクセルに対し変換処理を行った．

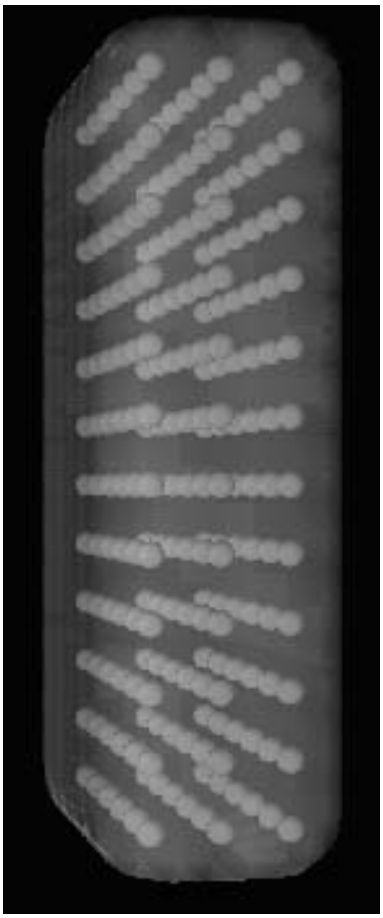


図 41　216 個のパーティクルによる仮想粘土モデル

まず，3.3 節で述べた線分によるインタラクションの様子を確認するために，線分 1 つを内部に持つ円柱形状を作成し，仮想粘土モデルの分断を試みた．円柱形状は一定の速度で平行移動する．変形の様子を図 42 に示す．仮想粘土モデルの変形部分が観察できるように，円柱形状はワイヤーフレームで描画している．図のように，押した部分は凹み，その周囲は体積保存のために盛り上がっていき，最

後には，形状が 3 つに分断される．

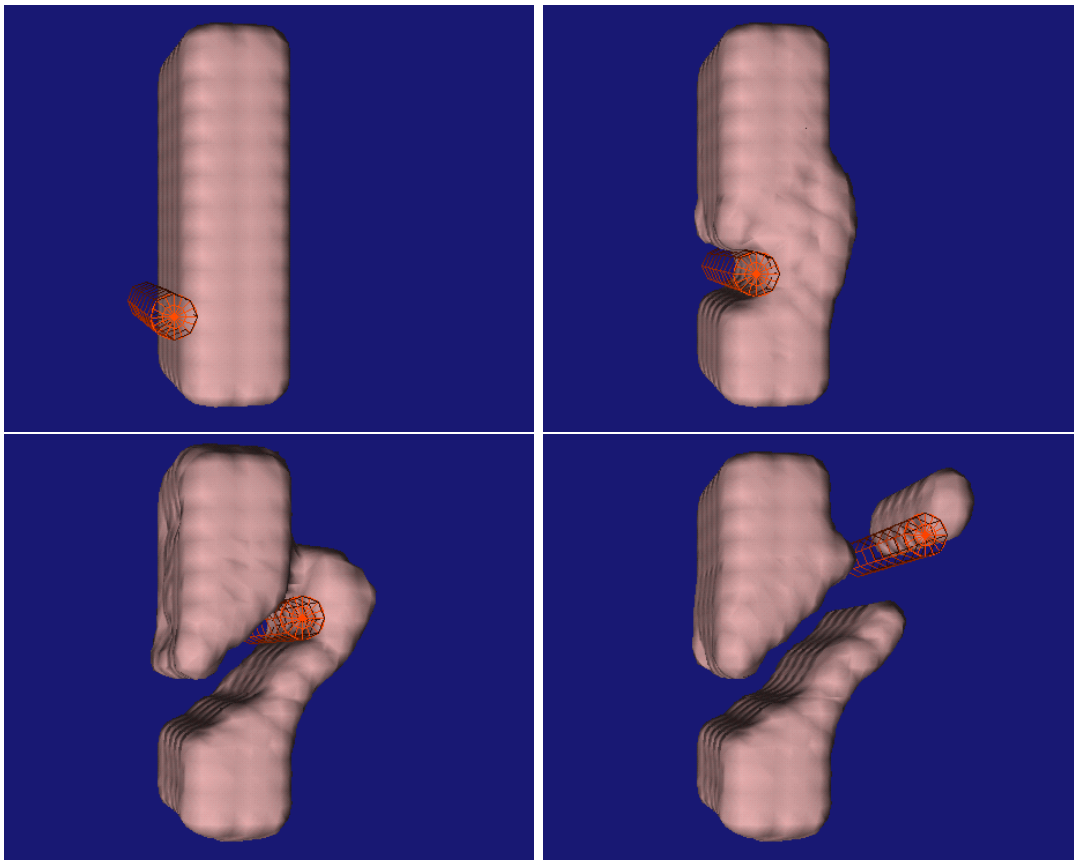


図 42　操作棒で仮想粘土モデルを分断する様子

次に，指を用いて仮想粘土モデルに,「押す」,「穴を開ける」,「摘んで分断する」という操作を加えた様子を図 43，図 44，図 45 および図 46 にそれぞれ示す．図 43 では，人差し指で仮想粘土モデルを押すことで，へこみを加えることができ，その周囲は盛り上がっている様子が確認できる．図 44 では，局所的に押すことで，押した部分の反対側が押し出されていることがわかる．図 45 では，人差し指で仮想粘土モデルに穴を開けることができ，穴の周囲は盛り上がっている．また，図 46 では，人差し指と親指を使って仮想粘土モデルを摘んでいき，分断することができている．図 47 および図 48 には，手のひらで仮想粘土モデルを「押して傾け

る」操作と「一部分を削り取る」操作を行っている様子を示す．図 47 では，手のひらによって形状全体を変形させる様子が確認できる．また，図 48 では，手のひらで削り取られる部分が押しつぶされながら，分断されていく様子が確認できる．以上のように，様々な形状変形を粘土を扱うように行えることが確認できる．

また，3.4.2 項で述べたスレッドそれぞれの 1 サイクルの処理時間を測定した．ポリゴン変換制御スレッドと 12 個のポリゴン変換スレッドに CPU を 12 個，パーティクルシステム制御スレッドと 2 個のパーティクルシステムスレッドに CPU を 2 個，ポリゴン描画スレッドに CPU を 1 個割り当て，測定を行った．ポリゴン変換制御スレッドにおけるポリゴン変換処理では，変形を加える領域に応じて処理時間は増減するが，図 46 のような変形操作において，最大 0.95 秒であった．この時，約 4500 個のボクセルについて変換が必要であった．また，パーティクルシステム制御スレッドにおける，216 個のパーティクルと線分 4 つの運動計算に要する時間は平均 0.044 秒であった．以上のように，それぞれの処理時間だけを見ると対話的操作に十分とはいえない．しかし，それぞれの処理はスレッドにより並列に処理されるため，ポリゴン描画処理による画像の更新レートは CPU を 1 個用いて 30fps となる．このため，ユーザにとっては，手が仮想粘土モデルに接触した後，最大約 1 秒程度遅れて変形するという感覚ではあるが，十分対話的操作が可能であるといえる．

ここで，CPU 数とそれぞれの処理時間について考察する．まず，ポリゴン変換処理について，使用した CPU 数と変換したボクセル数による処理時間の変化を表 1 に示し，それをグラフに表したものを図 49 に示す．図 49 において，CPU 数が 8 個から 10 個までは，CPU 数の増加とともに処理時間が全体的に短くなっているが，11 個では CPU 数が増加したにもかかわらず，処理時間が増加している．また，変換したボクセル数が 1000 個の時は，CPU 数と処理時間の関係が乱れている．これらの現象は，スレッドの増加にともない，3.4.2 節で述べたスレッドの管理や同期によるオーバーヘッドが増加し，各スレッドごとの処理時間とのバランスが取れないことから発生したと考えられる．また，CPU 数が 12 個および 13 個での処理時間が似通っている．これは，先程と同様に，スレッドの増加によるオーバーヘッドの増加のために，これ以上は処理時間が短縮されないと考えられ

る．以上を踏まえ，できるだけ少ない CPU 数で，処理時間を短くできることから，CPU 数は 12 個とした．

次に，パーティクルシステムにおける処理時間について，使用した CPU 数とパーティクル数による処理時間の変化を表 2，および，図 50 に示す．図 50 に見られるように，CPU を増加させることにより処理時間が増加してしまう．これは，各スレッドに割り当てられた処理量が少ないため，複数の CPU を有効に利用できておらず，スレッド分割にともなうオーバーヘッドの増加により生じていると考えられる．例外的に，パーティクル数 216 個の処理時間を見ると，2 個の CPU を用いた方が僅かに処理時間が短い．また，216 個よりも多くのパーティクルを用いた場合，処理に 0.1 秒以上かかり，パーティクルの運動計算が破綻する．以上を踏まえ，できるだけ多くのパーティクル数で，処理時間を短くできることから，パーティクルシステムにおける CPU 数は 2 個とした．また，343 個のパーティクルを用い，スレッドと CPU の 1 対 1 の割り当てを行わずに，さらにスレッドを増加させた場合の処理時間について，表 3，および，図 51 に示す．なお，ポリゴン変換は行わない．図表より，スレッド数を増加させることで処理時間が短縮できることがわかる．しかし，前述のポリゴン変換とともに行う場合は，CPU 数の限界から 0.1 秒以下に短縮することができない．

3.6 結言

本章では，パーティクルシステムと陰関数曲面を用い，少ない計算量で粘土のような変形が可能な仮想粘土モデルについて述べた．そして，パーティクルシステムに線分を導入することで手とのインタラクションを行うことについて述べた．また，仮想粘土モデルの高速な描画手法とマルチスレッドプログラミングを用いた並列処理について述べた．これらの手法により，仮想空間において仮想粘土を手で対話的に変形できることを実験により確認した．

しかし，本手法では，少ないパーティクルで変形挙動を表現するため，計算量が少ないという利点がある反面，細かな変形を加えることができない．次章では，前章と本章の手法を組み合わせた粘土細工モデリングシステムについて述べる．

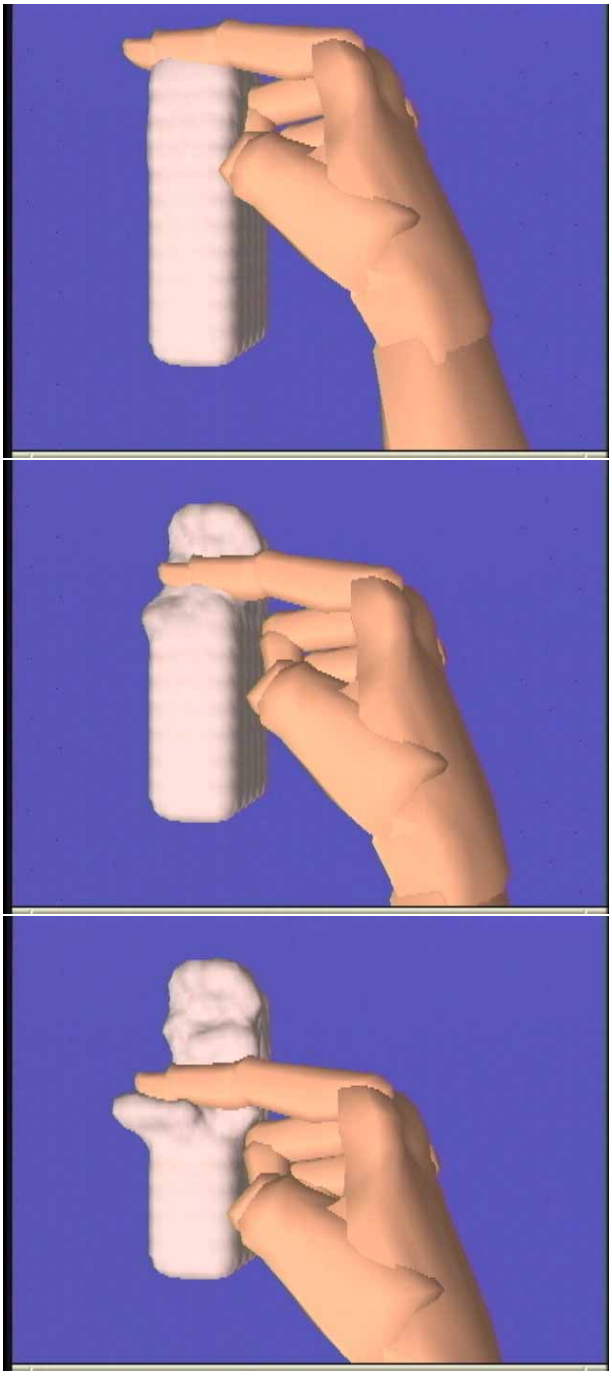


図 43　仮想粘土モデルを人差し指で押す様子 1

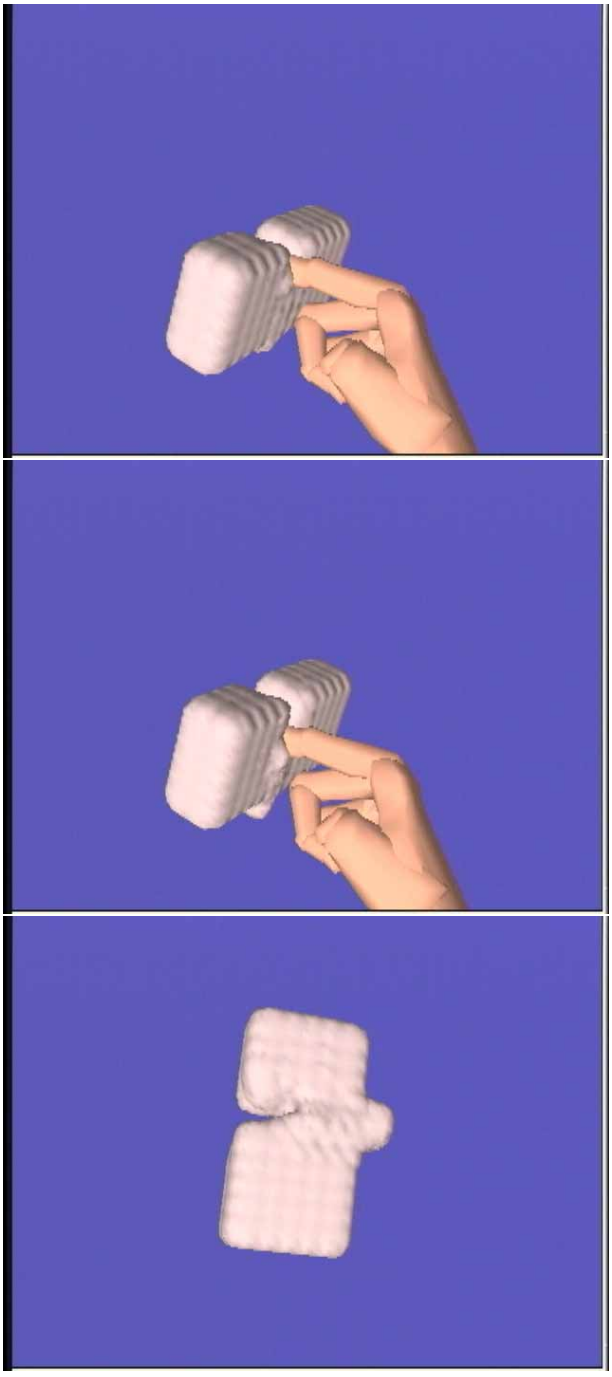


図 44　仮想粘土モデルを人差し指で押す様子 2

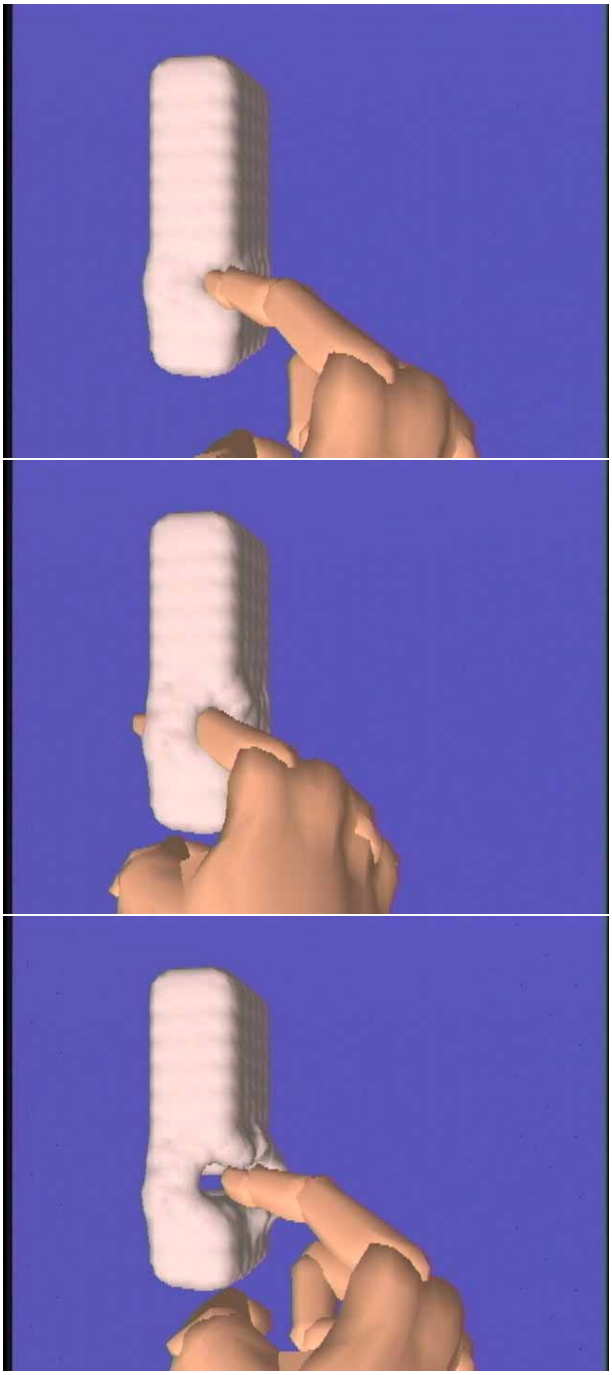


図 45　仮想粘土モデルに人差し指で穴を開ける様子

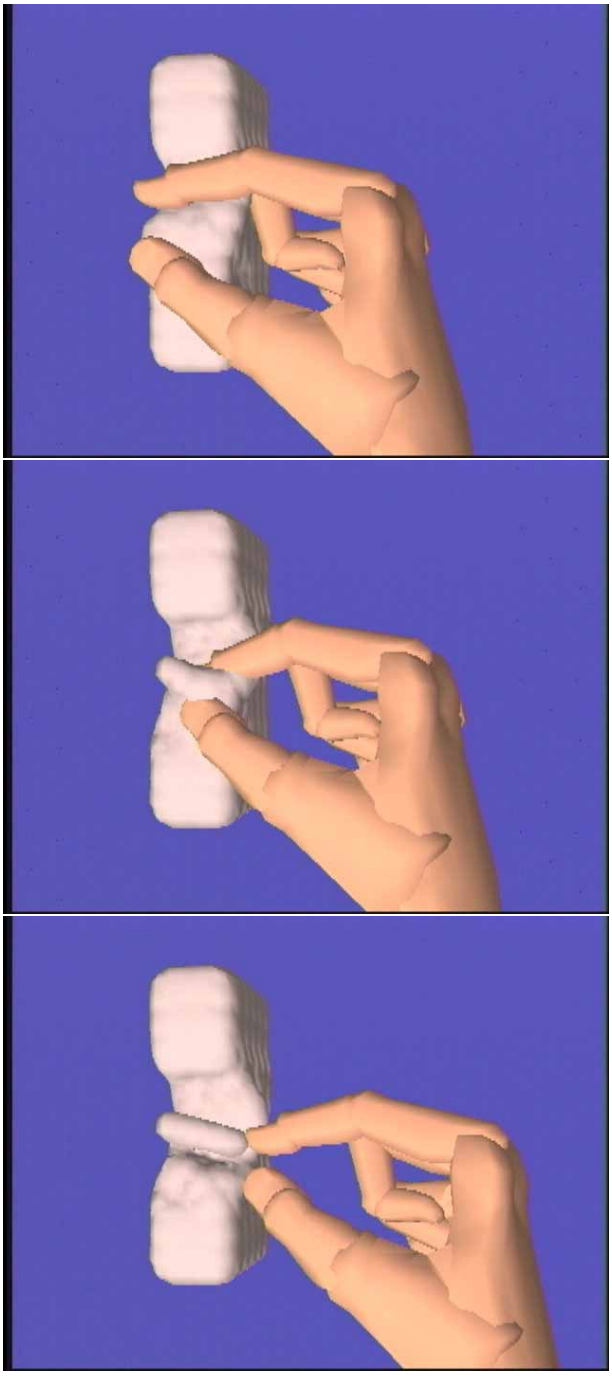


図 46　仮想粘土モデルを人差し指と親指でつまんで分断する様子

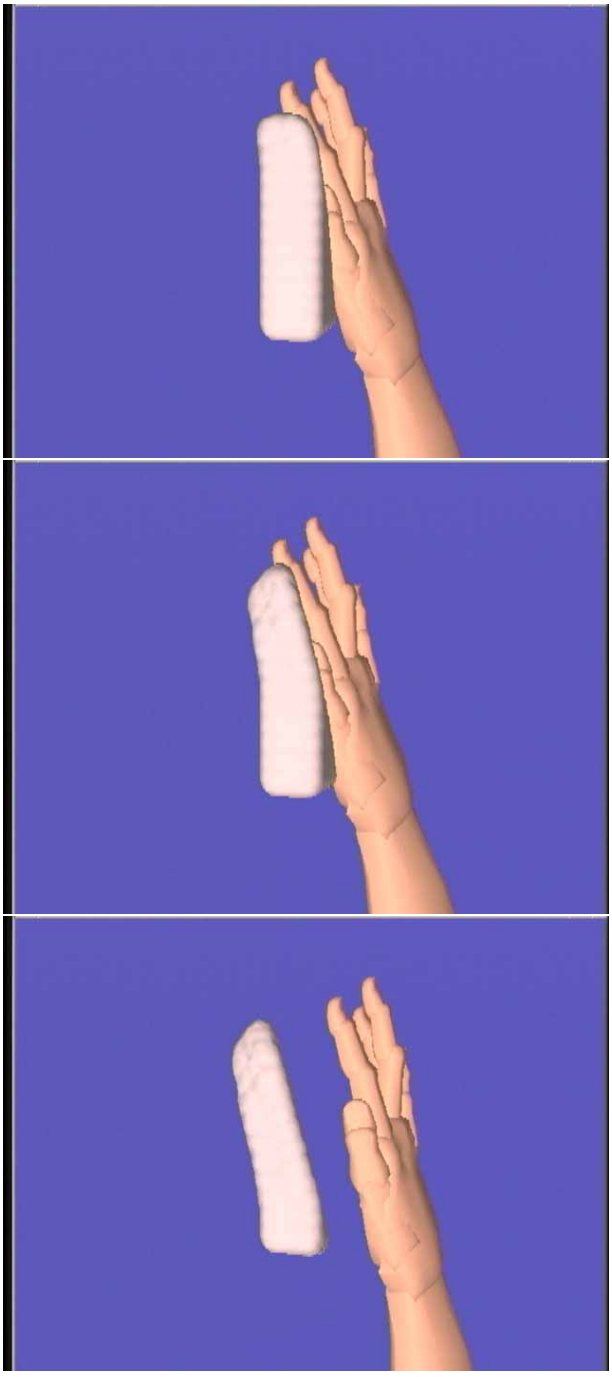


図 47 仮想粘土モデルを手のひらで押して傾ける様子

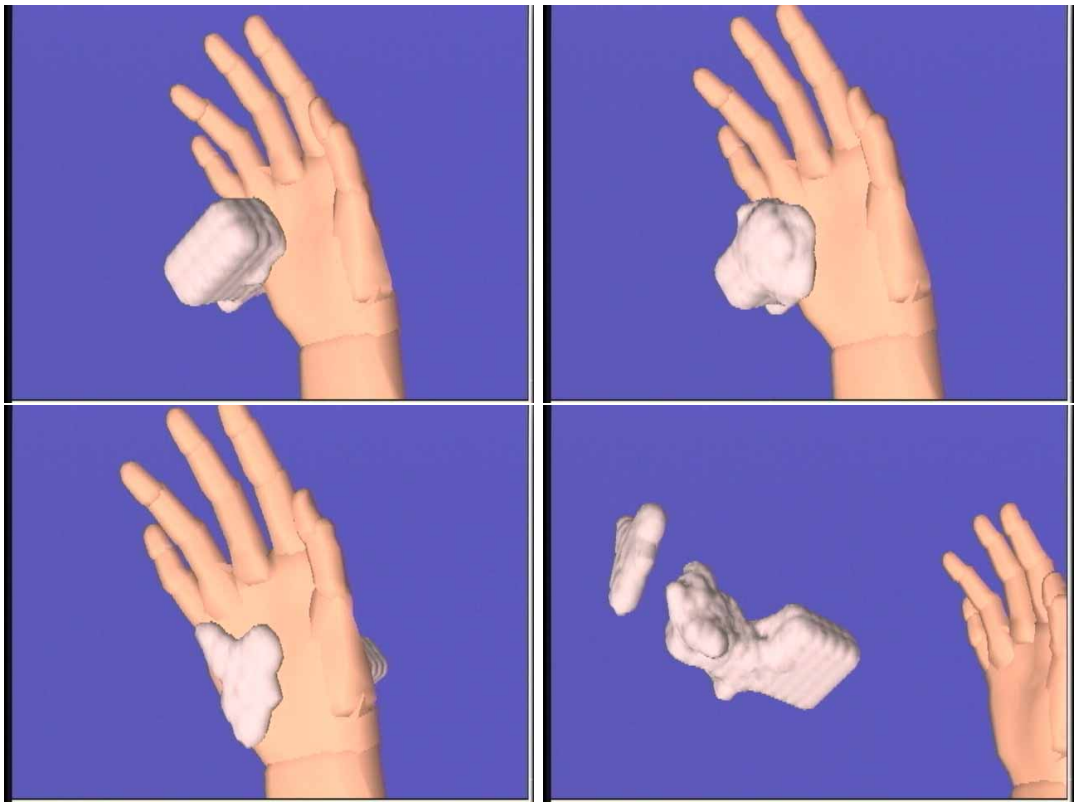


図 48 仮想粘土モデルを手のひらで分断する様子

表 1 CPU 数と変換したボクセル数によるポリゴン変換の処理時間の変化 (秒)

	変換したボクセル数			
	1000	2000	3000	4000
8CPUs	0.32	0.63	1.06	1.25
9CPUs	0.29	0.59	0.94	1.17
10CPUs	0.28	0.52	0.81	1.02
11CPUs	0.35	0.58	0.88	1.08
12CPUs	0.31	0.46	0.79	0.90
13CPUs	0.31	0.47	0.77	0.95

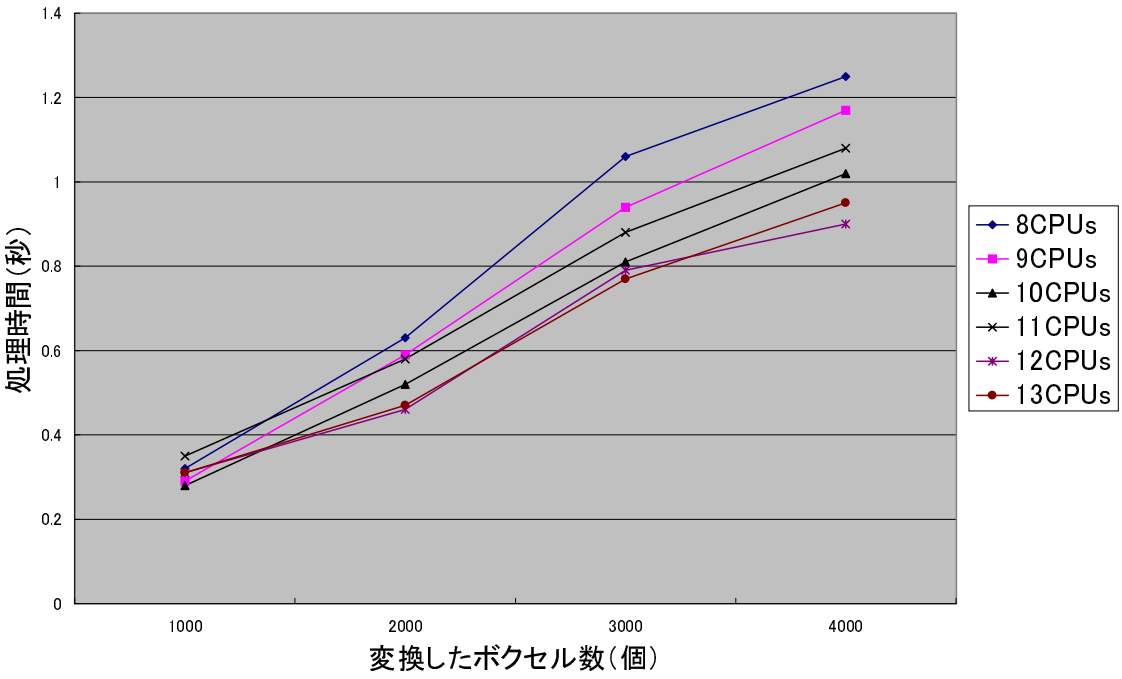


図 49　CPU 数と変換したボクセル数によるポリゴン変換の処理時間の変化

表 2　CPU 数とパーティクル数によるパーティクルシステムの処理時間の変化 (秒)

	パーティクル数			
	125	216	343	512
1CPU	0.016	0.046	0.105	0.236
2CPUs	0.015	0.044	0.146	0.342
3CPUs	0.027	0.062	0.158	0.286
4CPUs	0.028	0.064	0.151	0.263

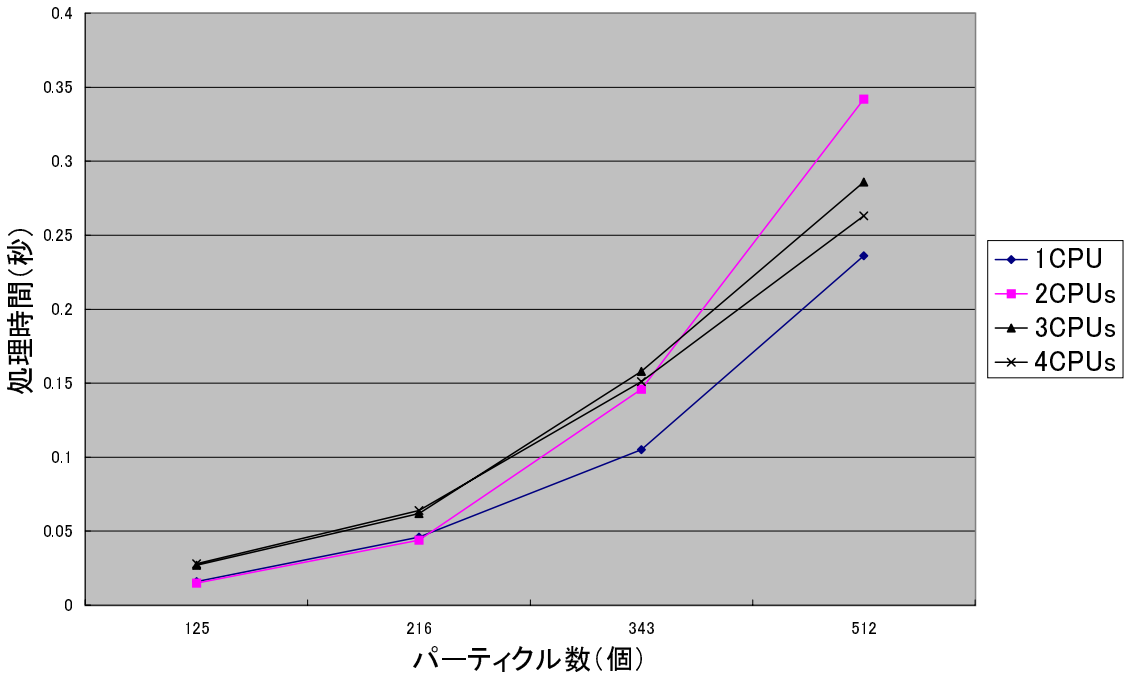


図 50　CPU 数とパーティクル数によるパーティクルシステムの処理時間の変化

表 3　スレッド数による 343 個のパーティクルの処理時間の変化

CPU 数	処理時間 (秒)
10	0.121
15	0.101
20	0.090
25	0.086
30	0.081
35	0.078

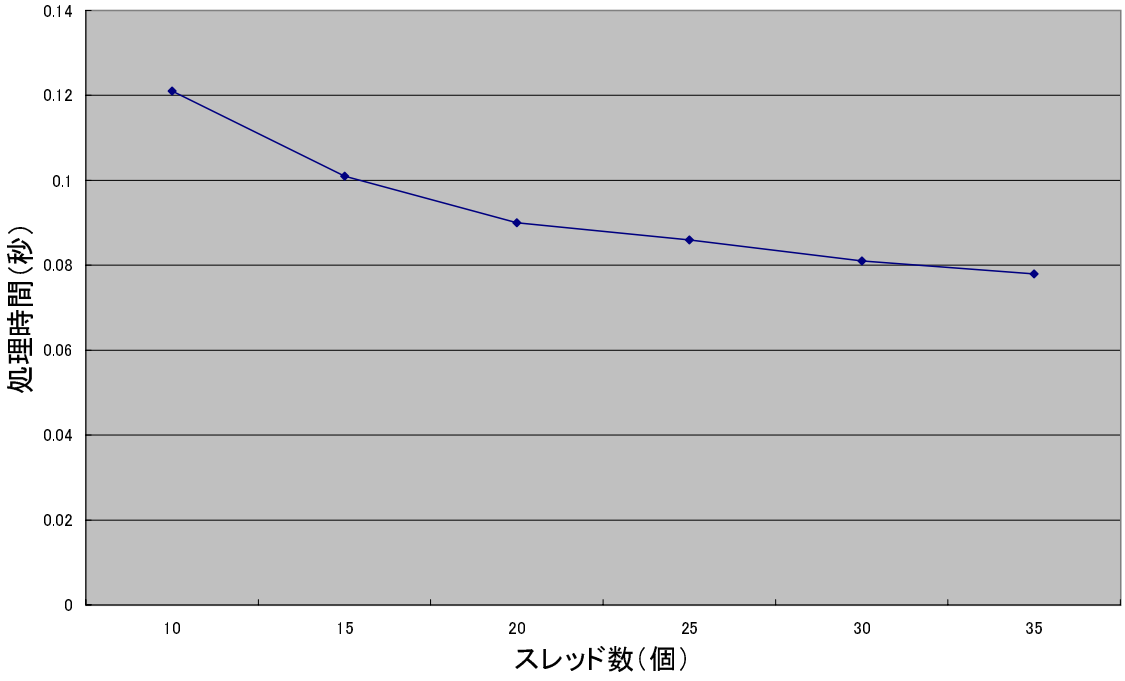


図 51　スレッド数による 343 個のパーティクルの処理時間の変化

4. 手と粘土へらを用いた仮想粘土細工システム

4.1 緒言

2 章および 3 章では，手を用いた粘土細工的な操作による自由形状モデリングについて述べたが，任意形状のモデリングという観点からみると，より細かな変形を加えることができる必要がある．本章では，粘土へらを導入し，手と粘土へらを用いてモデリングを行う仮想粘土細工システムについて述べる．本システムでは，まず，手を用いて仮想粘土を全体的に変形させることで，おおまかで全体的なモデリングを行う．次に，粘土へらを用いてより細かな形状の切削と付加を行う．この操作の時，細かな作業が容易に行えるように，仮想粘土形状の一部分が拡大される．

仮想粘土を全体的に変形させるためには，3 章で述べた仮想粘土モデルによる変形挙動の表現が必要である．一方，粘土へらによる細かな変形を表現するためには，局所的にしか変形が生じないことから物理モデルは必要としない．そこで，2 章で述べた手法における指形状を粘土へらとして用いる．このように，本システムは，2 章で述べた手法と 3 章で述べた手法を組み合わせたシステムである．

4.2 手と粘土へらを用いた仮想粘土細工システムの概要

手と粘土へらを用いた仮想粘土細工システムを使用している様子とシステム構成を図 52，図 53 に示す．ユーザは頭部に HMD を装着し，左手に 3 次元マウス，手形状入力装置を装着した右手に粘土へらとなるペン型入力装置を持つ．なお，計算機として SGI 社 ONYX2（MIPS R10000, 195MHz, 16CPU）を用い，HMD としてオリンパス社 Mediamask を，3 次元位置センサとして Polhemus 社 3SPACE Fastrak を，手形状入力装置として Virtual Technologies 社 CyberGlove をそれぞれ接続して使用した．3 次元マウスには，3 次元位置センサと 2 次元マウスを組み合わせたものを使用し，ペン型入力装置には，棒に 3 次元位置センサを取り付けたものを作成し，使用した．

図 52　手と粘土へらを用いた仮想粘土細工システムを使用している様子

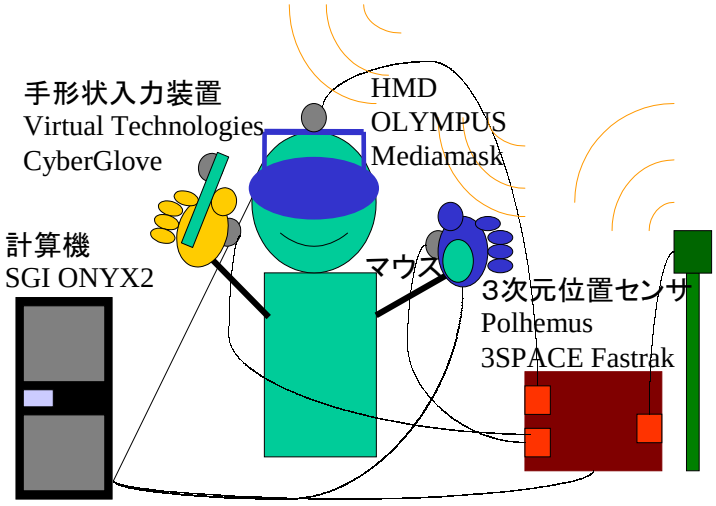


図 53　手と粘土へらを用いた仮想粘土細工システムの構成

ユーザは HMD を用いた立体視によって仮想空間へ没入する．仮想空間内の仮想粘土の移動と回転には 3 次元マウスを用いる．仮想粘土の変形操作には，1) 手を用いた全体変形段階，2) 粘土へらを用いた局所変形段階，の 2 つの段階がある．ユーザはこの 2 つの段階を順に経てモデリングを行う．

手を用いた全体変形段階

システム起動時には，あらかじめペン型入力装置を所定の位置へ置いておく．この間，手を用いた全体変形段階になる．

3 章で述べたものと同様の方法で，手を用いて仮想粘土に変形を加える．

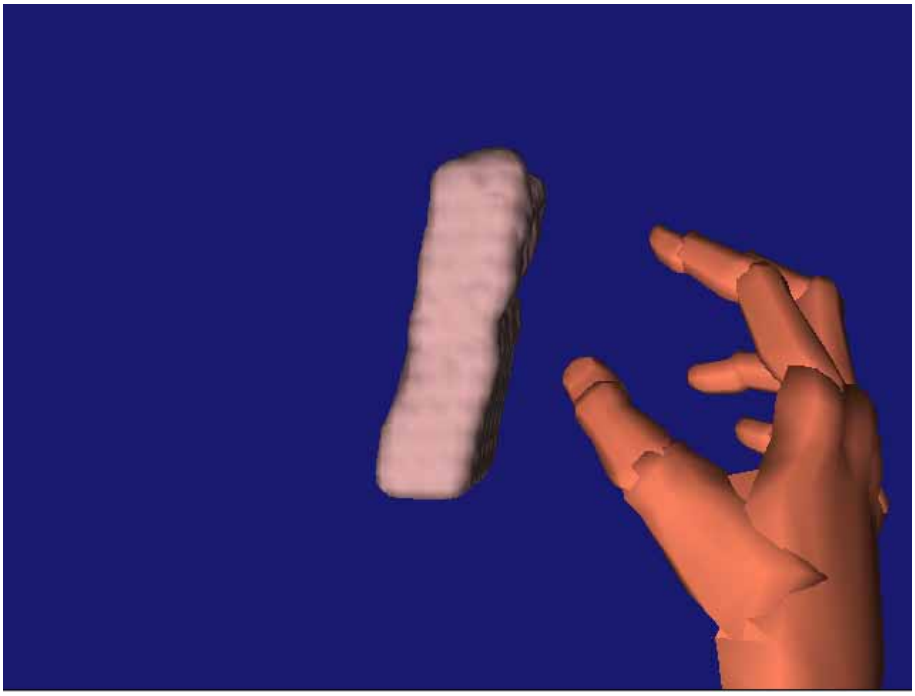


図 54 手を用いた全体変形段階

粘土へらを用いた局所変形段階

手を用いた全体的な変形によるモデリングが終了したら，所定の位置に置かれていたペン型入力装置を右手で持ちあげる．これにより，手を用いた全体変形段階が終了し，粘土へらを用いた局所変形操作へ移行する．

形状の一部分の拡大を行うには，拡大したい部分の中心をペン型入力装置で指定して，3 次元マウスのボタンを押す．形状変形のためのツールはサイズ変更が可能な球で，通過した部分に形状を付加する盛り付けと，通過した部分の形状を削除する削り取りの機能がある．サイズや機能の変更を 3 次元マウスのボタンを押すことで行いながら，ペン型入力装置でツールを操作する．

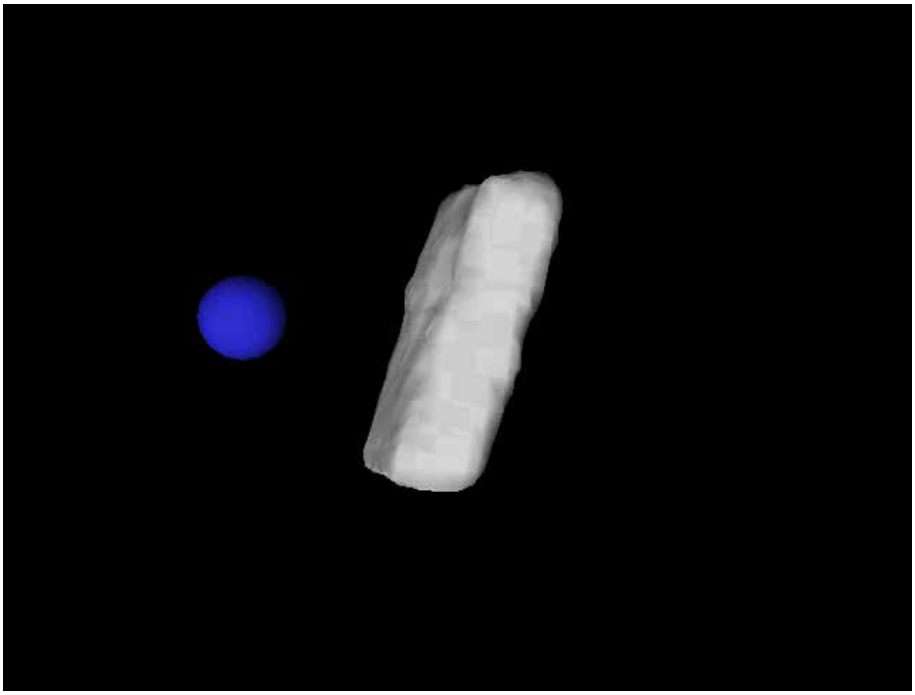


図 55 粘土へらを用いた局所変形段階

4.3 局所的細密表現による2つの変形段階の実現

前節で述べた2つの変形段階を実現するために用いる，全体的な変形と局所的な変形とを表現できる局所的細密表現について述べる．まず，全体変形段階では，図56(a)に示すように，形状を3章で述べた仮想粘土モデルを用いて表現する．図中の黒丸で示すものが仮想粘土モデルにおけるパーティクルである．また，点線で示す格子は，ポリゴン変換ボクセルを示す．

局所変形段階に移行し，細かな形状の付加・切削を行う時には，形状の一部分を拡大表示する．図56(a)に実線で示した格子の部分が拡大され，図56(b)のようになる．図56(a)に実線で示した格子の範囲だけに，点線で示した格子と同じ数のボクセルを再設定し，図56(b)のように，相対的に細かなポリゴン変換を行う．形状表面には陰関数曲面を用いているため，このようにポリゴン変換用ボクセルの大きさを変更して再変換することで，より精度の高いポリゴンの生成が行える．

図56(b)のような拡大表示の状態で，白丸に示す点スケルトンをペン型ツールの軌跡上に配置する．この点スケルトンと，生成される陰関数曲面が小さな球形状となるような距離場関数を用いて，2章で述べた形状変形手法を適用することで，細かな形状の付加・切削を行う．前述のように，ポリゴン生成の精度が高くなっているため，このようなより細かな形状を描画することができる．

ある部分のモデリングが終了したら，図56(c)に示すような，全体表示に切り替える．この時，形状全体を元のポリゴン変換用ボクセルを用いて再変換する．この時，ポリゴン変換用ボクセルが大きいため，先ほど付加・切削した細かな形状は正しくポリゴンに変換されず，描画される形状も滑らかさを欠いた不正確なものになる．しかし，形状データであるスケルトンは保持されているため，作品制作の最終段階でレイトレーシング法によって画像生成を行うような場合には，細かな形状も含めた正しい描画を行うことができる．

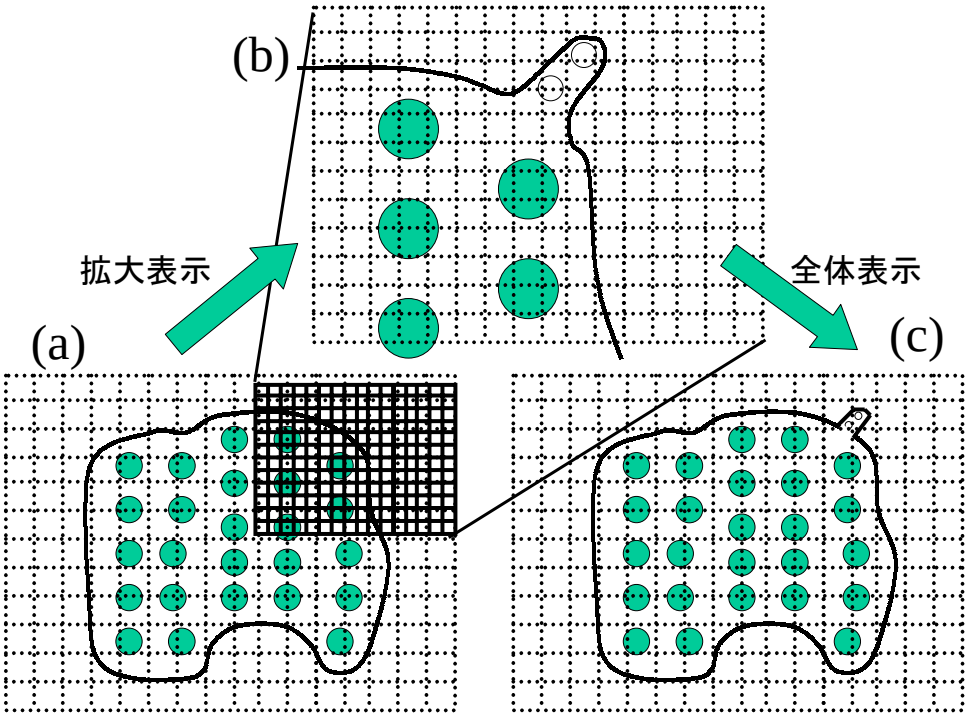


図 56　形状データの局所的細密表現

4.4　モデリング実験によるシステムの有効性の検証

本システムの有効性を検証するために行った，モデリング実験について述べる．仮想粘土モデルには 3 章と同様のパラメータを用い，蟹とモアイの形状のモデリングを行う．

まず，蟹の形状のモデリング過程について説明する．手を用いた全体変形段階において，図 57 に示すように，手を用いて全体のおおまかな形状と口の部分を作成した．次に，右手にペン型入力装置を持ち，手を用いた全体変形段階から粘土へらを用いた局所変形段階へ移行する．先ほど，手を用いた全体変形段階で作成した形状に対し，粘土へらツールを用いて細かな変形を加えていく．その様子

を図 58，図 59，図 60 および図 61 に示す．まず，細かな形状変形を加える部分を図 58 のようにペン型入力装置で指定し，3 次元マウスのボタンを押す．すると，図 59 のように拡大表示に切り替わる．前節で述べたように，指定した部分のみが陰関数曲面からポリゴン表現に，細かなポリゴン変換用ボクセルを用いて再変換され，拡大部分だけが滑らかに描画されていることが確認できる．この状態で，粘土へらを用いて細かな形状の付加と切削を行う．3 次元マウスのボタンを押し，全体表示に戻すと図 60 のようになる．先ほど形状変形を加えた部分が正しく描画されていないことが確認できるが，形状データは保持されているため問題はない．同様の操作を繰り返し，最終的に図 61 のような形状を作成した．

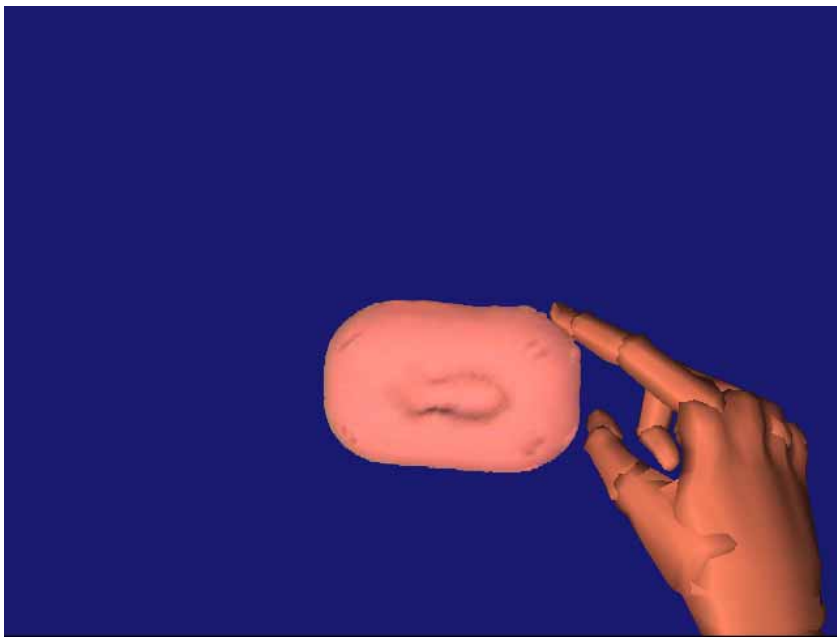


図 57 蟹の作成過程：全体変形段階での手を用いたモデリング

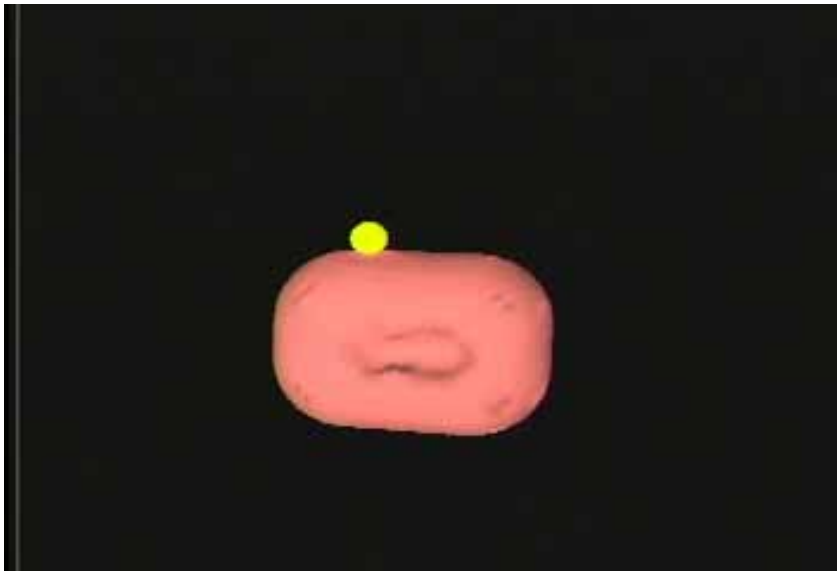


図 58　蟹の作成過程：局所変形段階での拡大部位の指定

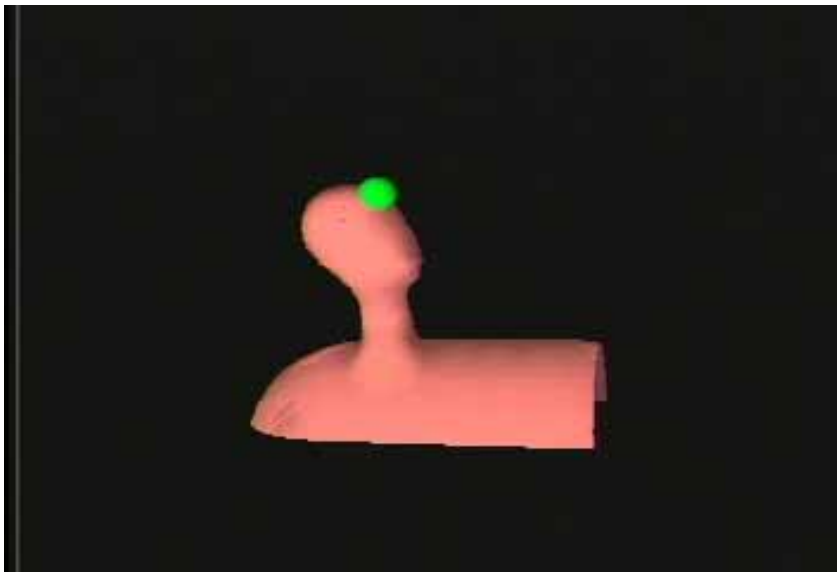


図 59　蟹の作成過程：局所変形段階での拡大部分に対する形状変形

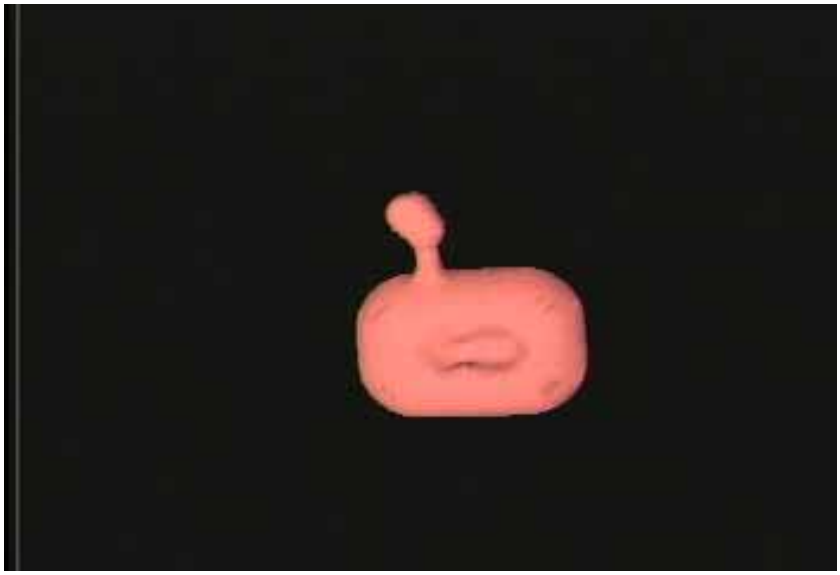


図 60　蟹の作成過程：局所変形段階での全体表示

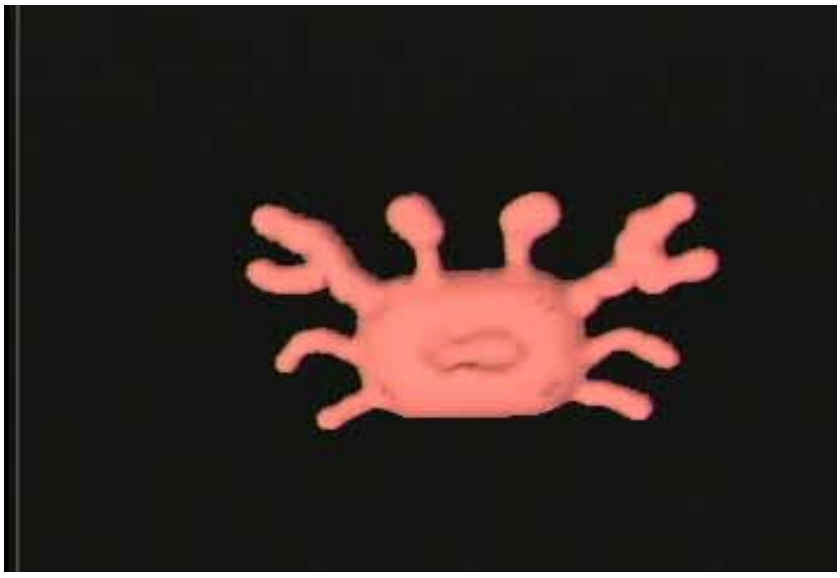


図 61　蟹の作成過程：完成した作品例

モアイの形状についても，図 62，図 63，図 64，図 65，図 66，図 67，図 68 および図 69 に示すような，蟹と同様の過程を通してモデリングを行った．まず，図 62 のように手で形状全体を傾け，図 63 のように全体の形状を整えた．局所変形段階へ移行し，図 64 のように鼻をモデリングした．次に，目のくぼみをモデリングするために図 65 のように拡大部位を指定し，図 66 のように拡大した．拡大部分に対し，図 67 のように目のくぼみをモデリングし，再び図 68 のような全体表示に切り替えた．最終的に図 69 のような形状を作成した．また，作成したモアイ形状を付録 B に述べる sim2dxf を用いて dxf 形式で保存し，市販の 3 次元 CG ソフトウェアで読み込み，レンダリングした作品例を図 70 に示す．

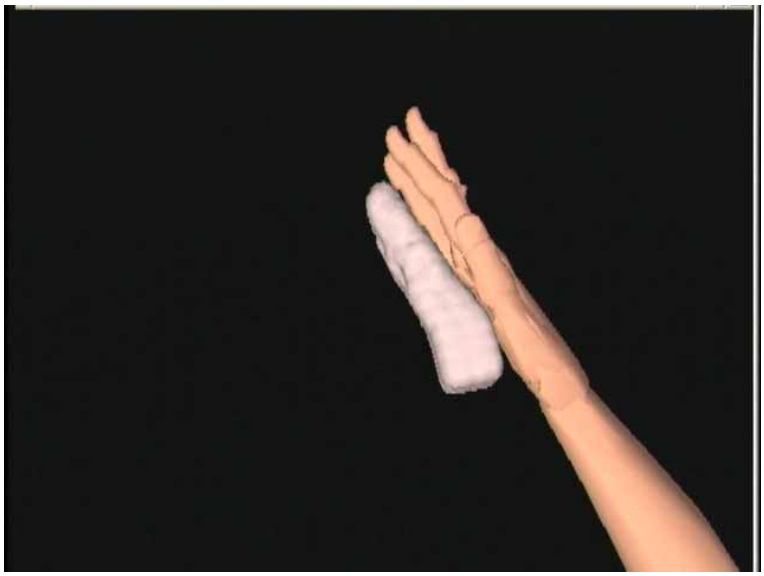


図 62 モアイの作成過程：全体変形段階での手を用いた形状の傾け

図 63　モアイの作成過程：全体変形段階での手を用いたモデリング

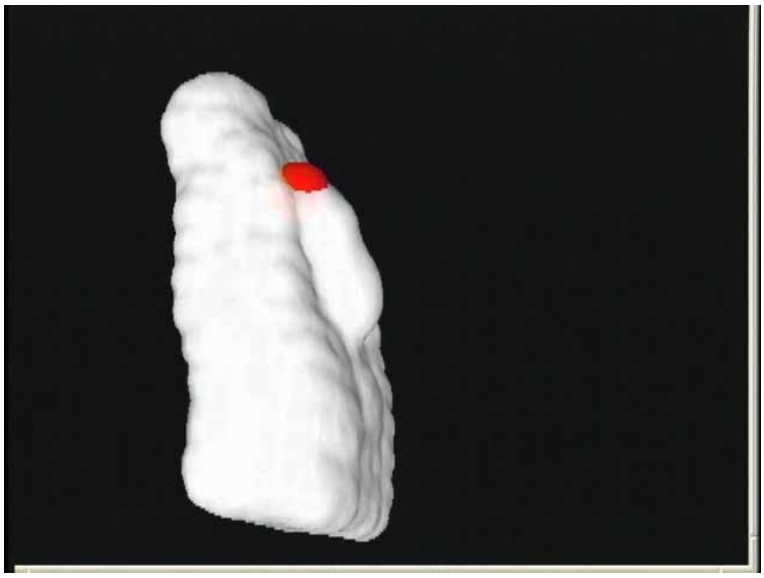


図 64　モアイの作成過程：局所変形段階での形状変形

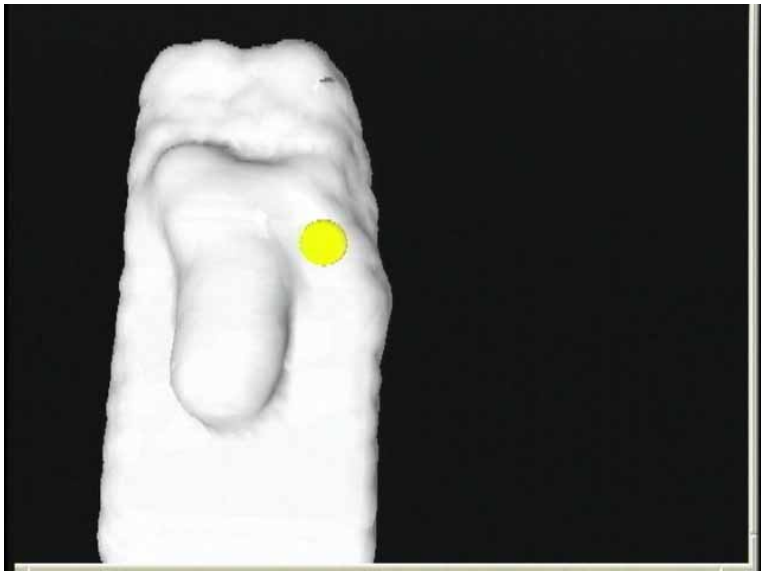


図 65　モアイの作成過程：局所変形段階での拡大部位の指定

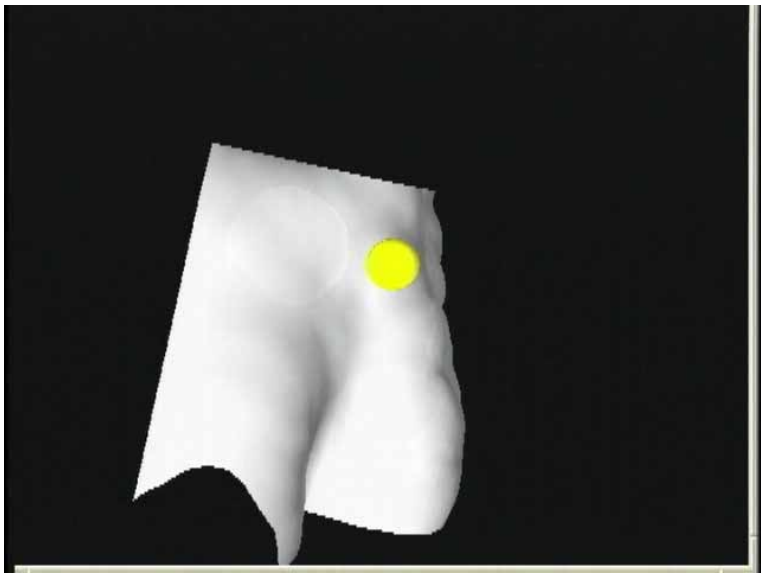


図 66　モアイの作成過程：局所変形段階での拡大表示

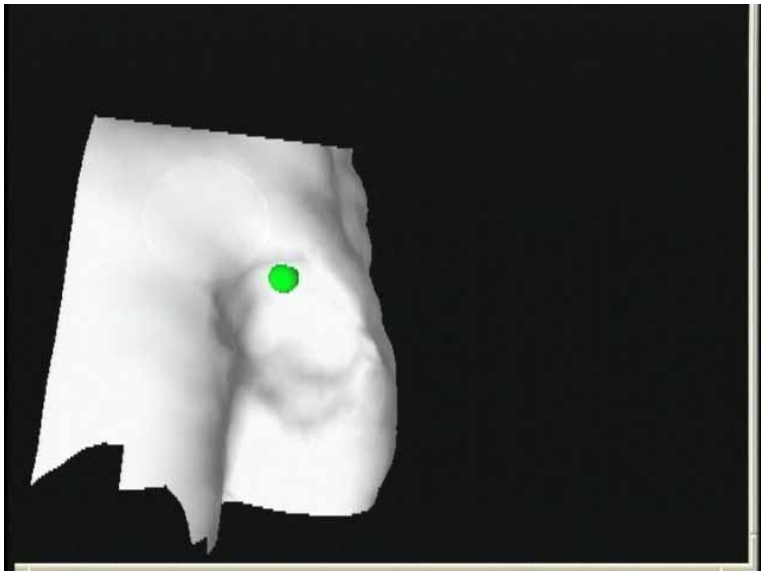


図 67　モアイの作成過程：局所変形段階での拡大部分に対する形状変形

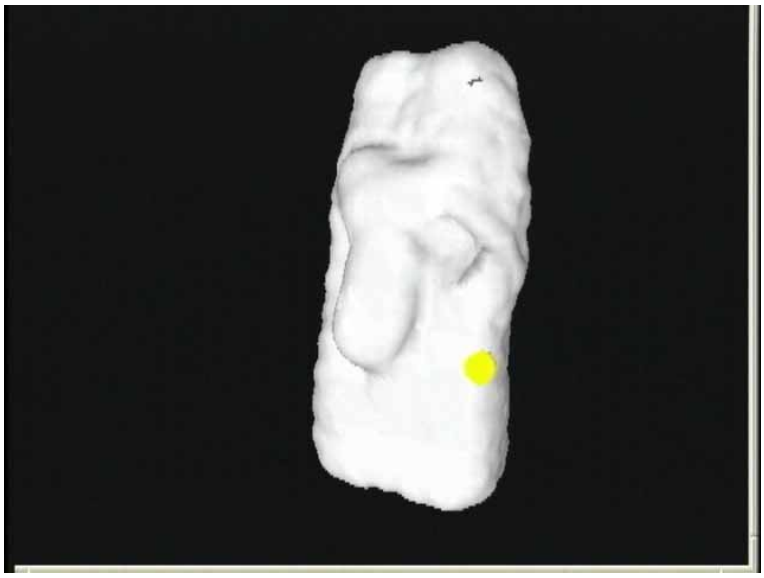


図 68　モアイの作成過程：局所変形段階での全体表示

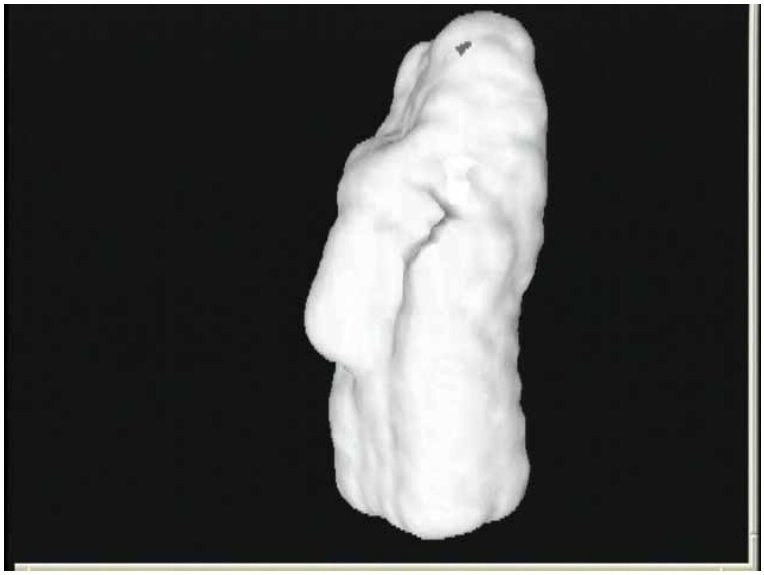


図 69　モアイの作成過程：完成した作品例

図 70　モアイ形状と市販ソフトウェアを利用して描画した画像

4.5 結言

本章では，2 章と 3 章で述べた手法を組み合わせることで，手と粘土へらを用いて形状変形を行える仮想粘土細工システムを構築した．全体的でおおまかな形状変形を，2 章で述べたような形状の切削や付加だけで行うには，何度も切削や付加を繰り返すことになり，手間がかかる．一方，3 章で述べた仮想粘土モデルだけでは，細かな形状変形の表現に限界がある．本章で述べた仮想粘土細工システムでは，2 つの手法の長所を組み合せることで，表現力を高めつつ，より簡単にモデリングが行える．

本章で述べた仮想粘土細工システムでは，手を用いた全体変形段階から粘土へらを用いた局所変形段階へ移行すると，再び手を用いた全体変形段階に戻ることができない．2 つの段階を自由に移行することができれば，よりモデリングの作業効率が向上すると考えられる．そこで，局所的細密表現に拡張を加える必要がある．以下に，そのアイディアを簡単に述べる．

図 71 に示すように，局所変形段階において局所的に付加する形状の点スケルトンを格子状に生成する．また，これらの点スケルトンのみでパーティクルシステムを構成する．このとき，パーティクル間の安定距離を格子の間隔と等しくする．さらに，この独立したパーティクルシステムを，最も近くにある元形状のパーティクルの 1 つと接続関係を持たせ，相対位置が変わらないようにする．

このようにすることで，局所変形段階で形状を付加した後，全体変形段階に移行した場合に，付加した形状もパーティクルシミュレーションによって粘土らしい変形を加えることができる．また，元の形状と付加した形状は，それぞれ独立したパーティクルシステムであり，別々に計算が可能なため，点スケルトンが増加することで計算時間が増加することを抑えることができる．さらに，元の形状を変形させた場合，付加した形状との接続関係により，パーティクルシステムが独立であるため生じる，元形状と付加形状の分離を防ぐことができる．

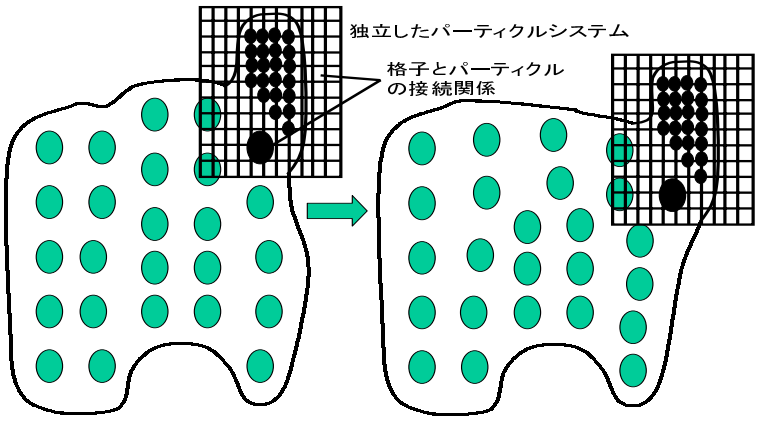


図 71　局所的細密表現の拡張

5. 結論

本論文では，現実の粘土細工による造形と同様の方法を用いて，仮想空間内において自由形状のモデリングを行うことができる仮想粘土細工システムの開発を目的として行った研究について述べた．

まず，2 章において，仮想現実感技術と陰関数曲面を用いることにより，手を用いて直接的に仮想粘土へ凹凸を加えるという形状の変形を実現する手法を提案した．形状変形に際しては，陰関数曲面表現の特徴を利用して，形状の衝突判定や変形処理に複雑な処理を必要とせず，その定義から自然に変形を表現している．指の近傍のボクセルに対してのみ，陰関数曲面をポリゴン表現に変換する手法によって，処理の軽減を図った．また，これらの手法を用いることに加えて，対称型マルチプロセッサシステムにおけるマルチスレッドプログラミングを用いた並列処理によって，実時間の対話性を実現した．提案手法を実際に実装して構築した試作システムによって作成したモデリング例を示し，簡単な操作で直観的に 3 次元形状モデリングが行えることを確認した．

次に，現実の粘土細工の操作を考えると，形状全体を変形させるような操作を行うことがあるが，これを実現するためには，粘土の変形挙動を表現する必要があった．そこで，3 章においては，パーティクルシステムと陰関数曲面を用いた仮想粘土モデルに対し，手を用いて変形を加える手法を提案した．現実の粘土の変形挙動を少ないパーティクルで表現するとともに，粘土の滑らかな表面形状を陰関数曲面を用いて表現した．ここでも，これらの手法を用いることに加えて，閾値以上移動したパーティクルの存在する近傍のボクセルに対してのみ，陰関数曲面をポリゴン表現に変換する手法と，マルチスレッドプログラミングによる並列処理を用いて，対話性を実現した．そして，試作システムによる変形実験によって，対話的に粘土のような変形を加えることができることを確認した．

最後に，4 章において，上記の 2 つの手法の長所を組み合わせることで，粘土へらと手を用いた，より表現力の高い粘土細工を行える仮想粘土細工システムを構築した．このシステムでは，全体的なおおまかな変形を，3 章で述べた仮想粘土モデルによって表現した．そして，細かな変形を加える際には，2 章で述べた陰関数曲面の性質による形状の付加・切削の手法を用いて表現した．また，拡大

表示に切り替えて細かな変形操作を行うことができる．このシステムを用いたモデリング例を示し，粘土細工の要領で 3 次元形状をモデリングすることができることを確認した．

本研究で開発した仮想粘土細工システムで作成できる 3 次元形状は，設計の初期モデルのモデリングや，モデリング例で示したような正確さが求められないデザイン作成などの用途に有用である．現在，一般に用いられている 3 次元 CAD ソフトウェアや 3 次元 CG ソフトウェアによるモデリング方法と比較して，試作システムは特別な前提知識なしにモデリングを行うことができるという特長を持っている．そのため，全くの初心者であっても，容易に 3 次元形状のモデリングを行うことができる．熟練者にとっても，複雑な形状操作に煩わされることなく形状設計に集中でき，対話的試行錯誤による自由な発想で，モデリングを行うことができる．また，一般に用いられている 3 次元 CAD ソフトウェアや 3 次元 CG ソフトウェアによるモデリング方法では困難と考えられる，手作り感のある形状をモデリングすることができる．以上のことから，仮想現実感技術を利用した新たな造形手法を提供できたと考える．現状では，精密さや正確さに欠けるため，精密な工業製品などを正確にモデリングするような用途には向かない．しかし，処理手法の高速化やコンピュータの高性能化，モデリング補助機能の付加などを行うことによって，精密な形状のモデリングも可能である．

本研究では主に，仮想空間における粘土という柔物体の対話的変形手法の開発に重点をおき，一定の成果を確認することができた．しかし，より実用的な仮想空間における粘土細工の実現には，本来，形状操作に対して発生するべき反力や触覚の表現を行う必要がある．VR の研究において，このような反力や触覚についての研究が活発に行われている [68, 18, 69, 70]．図 72 は手と腕に反力を提示することができる装置の例である [71]．これらは，仮想空間における 3 次元形状操作に重要であるため，今後，反力や触覚の高速な計算手法や，それをユーザに与えるインタフェースについて研究を行い，システムに導入する必要がある．

一方，近年，3 次元 CG アニメーションには無い，手作り感や独特でコミカルな動きなどの特徴を持った粘土細工を用いたアニメーション（クレイアニメーション）[72] が，人気を集めている．今後の研究として，このようなクレイアニメー

ションの仮想空間を利用した作成が考えられる.

図 72　CyberForce の外観（文献 [71] より引用）

謝辞

本研究を行う機会を与えて頂き，研究の全過程において直接懇切なる御指導御鞭撻を賜ったソフトウェア基礎講座 横矢 直和 教授に心より感謝申し上げます．

本研究の全過程を通して，終始有益な御助言と励ましの言葉を頂いた像情報処理学講座 千原 國宏 教授 に厚く御礼申し上げます．

本研究の遂行にあたり，終始的確な御助言と励ましの言葉を頂いた計算機アーキテクチャ講座 湊 小太郎 教授 に深く感謝致します．

本研究の全過程を通して，直接懇切なる御指導御鞭撻を賜ったソフトウェア基礎講座 竹村 治雄 助教授（現，大阪大学サイバーメディアセンター教授）に心より感謝申し上げます．

本研究について様々な御助言，御指導を与えて下さった元 ソフトウェア基礎講座 岩佐 英彦 助手（現,（株）ネットシステムズ） およびソフトウェア基礎講座 山澤 一誠 助手 に厚く御礼申しあげます．

海部 陸 君は，本研究の一部を共に行った後輩であり，大きな力となって頂きました．ここに，心より感謝の意を表します．

また，物心両面において常に温かいご支援を頂いたソフトウェア基礎講座の諸氏，ならびに，ソフトウェア基礎講座事務補佐員 北川 知代 女史および，元ソフトウェア基礎講座事務補佐員 福永 博美 女史に感謝します．

私が大阪工業大学 電子工学科在学中，多大なご迷惑をおかけしたにも関わらず，機会ある毎に様々な御助言と励ましの言葉を与えて下さった大阪工業大学 情報科学部 小堀 研一 教授 に深く感謝申し上げます．

大阪樟蔭女子大学 情報処理研究室 飼原 壽夫 助教授および，情報処理研究室の方々には,「教える」ことの楽しさを知る機会を与えて頂くとともに，様々な御助言と励ましの言葉を与えて頂きました．ここに，心より感謝の意を表します．

最後に，両親を含め，私を支援して頂いた多くの方々に，心より御礼申し上げます．本当にありがとうございました．

参考文献

[1] オートデスク株式会社. AutoCAD Release 14 ユーザ ガイド, 1997.

[2] MetaCreations Corporation. Ray Dream Studio 5 ユーザガイド, 1997.

[3] NewTek Incorporated. LightWave 3D version 5.5 日本語ユーザーガイド, 1997.

[4] ExpressionTools, Inc. *Shade User Guide*, 2001.

[5] 廣瀬通孝. バーチャルリアリティの基礎 3　VR 世界の構成手法. 培風館, 2000.

[6] FUJITSU. CELSIUS CG 講座 第 67 回：クレイアニメを作ろう Vol.1. http://primeserver.fujitsu.com/celsius/3dcg/2001_0904_2.html, 2001.

[7] K. Pimentel and K. Teixeira. *Virtual Reality.* Windcrest, 1994.

[8] G. Burdea and P. Coiffet. *Virtual Reality Technology.* John Wiley & Sons, inc., 1994.

[9] J. Butterworth, A. Davidson, S. Hench, and T. M. Olano. 3dm: A three dimensional modeler using a head-mounted display. In *Proc. ACM Interactive 3D Graphics*, pp. 135–139, 1992.

[10] D. A. Bowman and L. F. Hodges. User interface constraints for immersive virtual environment applications. In *Graphics, Visualization and Usability Center Technical Report GIT-GVU-95-26.* Georgia Institute of Technology, 1995.

[11] M. R. Mine. Working in a virtual world: Interaction techniques used in the chapel hill immersive modeling program. In *UNC Chapel Hill Computer Science Technical Report TR96-029*, 1996.

[12] K. Kiyokawa, H. Takemura, Y. Katayama, H. Iwasa, and N. Yokoya. Vlego: A simple two-handed modeling environment based on toy blocks. In *Proc. ACM Virtual Reality Software and Technology (VRST '96)*, pp. 27–34, 1996.

[13] S. W. Wang and A. E. Kaufman. Volume sculpting. In *Proc. ACM Interactive 3D Graphics*, pp. 151–156, 1995.

[14] J. R. Bill and S. K. Lodha. Sculpting polygonal models using virtual tools. In *Proc. CHCCS Graphics Interface '95*, pp. 272–278, 1995.

[15] R. A. Noble and G. J. Clapworthy. Sculpting and animating in a desktop vr environment. In *Proc. IEEE Computer Graphics International*, pp. 187–195, 1998.

[16] S. Coquillart. Extended free-form deformation: A sculpturing tool for 3d geometric modeling. In *Proc. ACM SIGGRAPH '90*, pp. 187–196, 1990.

[17] J. P. Y. Wong, R. W. H. Lau, and L. Ma. Virtual 3d sculpturing with a parametric hand surface. In *Proc. IEEE Computer Graphics International*, pp. 178–186, 1998.

[18] Y. H. Chai, G. R. Luecke, and J. C. Edwards. Virtual clay modeling using the isu exoskeleton. In *Proc. IEEE Virtual Reality Annual International Symposium (VRAIS '98)*, pp. 76–80, 1998.

[19] E. Ferley, M. P. Cani, and J. D. Gascuel. Practical volumetric sculpting. *The Visual Computer*, Vol. 16, No. 8, pp. 469–480, 2000.

[20] L. Markosian, J. M. Cohen, T. Crulli, and J. F. Hughes. Skin: A constructive approach to modeling free-form shapes. In *Proc. ACM SIGGRAPH '99*, pp. 393–400, 1999.

[21] T. A. Galyean and J. F. Hughes. Sculpting: An interactive volumetric modeling technique. In *Proc. ACM SIGGRAPH '91*, pp. 267–274, 1991.

[22] 水野慎士, 岡田稔, 鳥脇純一郎, 横井茂樹. 仮想彫刻—仮想空間における対話型形状生成の一手法. 情報処理学会論文誌, Vol. 38, No. 12, pp. 2509–2516, 1997.

[23] 荒田秀樹, 高井昌彰, 高井那美, 山本強. 能動的ボクセル空間における仮想粘土モデリング—モデルの基本理念—. 電子情報通信学会論文誌 (D-II), Vol. J82-D-II, No. 11, pp. 2008–2016, 1999.

[24] 小田泰行, 千葉則茂. 粒子ベースモデルによる粘土のビジュアルシミュレーション. 情報処理学会研究報告, Vol. 97, No. 124, pp. 25–30, 1997.

[25] 小田泰行, 村岡一信, 千葉則茂. 仮想粘土の粒子ベース・ビジュアルシミュレーション. 情報処理学会論文誌, Vol. 42, No. 5, pp. 1142–1150, 2001.

[26] 前野輝, 岡田稔, 鳥脇純一郎. 粘土細工モデリングにおける物体変形操作に関する基礎検討. 情報処理学会研究報告, Vol. 99, No. 19, pp. 7–12, 1999.

[27] 前野輝, 岡田稔, 鳥脇純一郎. 形状関数を用いた双三次ベジエ曲面の直観的変形手法. 日本バーチャルリアリティ学会論文誌, Vol. 5, No. 1, pp. 811–817, 2000.

[28] 梅村隆, 岡田稔. メタボールを用いた会話的モデラのためのパッチの一生成法. 画像電子学会誌, Vol. 26, No. 4, pp. 306–313, 1997.

[29] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: Viscoelasticity, plasticity, fracture. In *Proc. ACM SIGGRAPH '88*, pp. 269–278, 1988.

[30] M. P. Cani-Gascuel and M. Desbrun. Animation of deformable models using implicit surfaces. *IEEE Trans. on Visualization and Computer Graphics*, Vol. 3, No. 1, pp. 39–50, 1997.

[31] K. Kameyama. Virtual clay modeling system. In *Proc. ACM Virtual Reality Software and Technology (VRST '97)*, pp. 197–200, 1997.

[32] D. L. James and D. K. Pai. Artdefo: Accurate real time deformable objects. In *Proc. ACM SIGGRAPH '99*, pp. 65–72, 1999.

[33] K. T. McDonnell, H. Qin, and R. A. Wlodarczyk. Virtual clay: A real-time sculpting system with haptic toolkits. In *Proc. ACM Interactive 3D Graphics '01*, pp. 179–190, 2001.

[34] 亀井克之, 中村泰明, 阿部茂. エネルギー最小化による変形可能仮想ろくろモデラ. 電子情報通信学会論文誌 (D-II), Vol. J76-D-II, No. 8, pp. 1772–1779, 1993.

[35] 松宮雅俊, 清川清, 竹村治雄, 横矢直和. 陰関数表現を用いた仮想空間内曲面モデリングシステム. 電子情報通信学会総合大会講演論文集, No. A-16-35, 1998.

[36] 松宮雅俊, 清川清, 竹村治雄, 横矢直和. 陰関数表現を用いた仮想空間没入型曲面形状モデラ. 日本バーチャルリアリティ学会第 3 回大会論文集, pp. 53–56, 1998.

[37] 松宮雅俊, 清川清, 竹村治雄, 横矢直和. 没入型仮想環境における 3 次元曲面形状モデリング. 電子情報通信学会技術報告, MVE98-86, 1999.

[38] M. Matsumiya, K. Kiyokawa, H. Takemura, and N. Yokoya. An immersive modeler for curved objects using implicit surfaces. In *Proc. SPIE Intelligent Systems in Design and Manufacturing II*, pp. 109–120, 1999.

[39] M. Matsumiya, H. Takemura, and N. Yokoya. An immersive modeling system for 3d free-form design using implicit surfaces. In *Proc. ACM Virtual Reality Software and Technology 2000 (VRST2000)*, pp. 67–74, 2000.

[40] 松宮雅俊, 竹村治雄, 横矢直和. 自由形状モデリングのための陰関数曲面を用いた仮想粘土細工システム. 情報処理学会論文誌, Vol. 42, No. 5, pp. 1151–1160, 2001.

[41] J. Bloomenthal and B. Wyvill. Interactive techniques for implicit modeling. In *Proc. ACM Interactive 3D Graphics*, pp. 109–116, 1990.

[42] B. Wyvill. Animation and special effects. In J. Bloomenthal, editor, *Introduction to Implicit Surfaces*, pp. 242–269. Morgan Kaufmann Publishers, 1997.

[43] E. Galin and S. Akkouche. Incremental techniques for implicit surface modeling. In *Proc. IEEE Computer Graphics International*, pp. 312–321, 1998.

[44] J. Blinn. A generalization of algebraic surface drawing. *ACM Trans. on Graphics*, Vol. 1, No. 3, pp. 135–256, 1982.

[45] 西村仁志, 平井誠, 河合利幸, 河田享, 白川功, 大村皓一. 分布関数による物体モデリングと画像生成の一手法. 電子情報通信学会論文誌 (D), Vol. J68-D, No. 4, pp. 718–725, 1985.

[46] G. Wyvill, C. McPheeters, and B. Wyvill. Data structure for soft objects. *The Visual Computer*, Vol. 2, No. 4, pp. 227–234, 1986.

[47] 技術系 CG 標準テキストブック編集委員会. コンピュータグラフィックス 技術系 CG 標準テキストブック. 財団法人画像情報教育振興協会, 1995.

[48] J. C. Hart. Sphere tracing: A geometric method for the antialiased ray tracing of implicit surfaces. *The Visual Computer*, Vol. 12, No. 10, pp. 527–545, 1996.

[49] G. Wyvill. Ray tracing implicit surfaces. In J. Bloomenthal, editor, *Introduction to Implicit Surfaces*, pp. 166–195. Morgan Kaufmann Publishers, 1997.

[50] M. Desbrun, N. Tsingos, and M. P. Gascuel. Adaptive sampling of implicit surfaces for interactive modeling and animation. In *Proc. Implicit Surfaces '95*, pp. 171–185, 1995.

[51] 土井章男, 小出昭夫. 等関数値曲面生成のための 4 面体格子法. 第 3 回 NICOGRAPH 論文コンテスト論文集, pp. 55–61, 1987.

[52] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*, Vol. 21, No. 4, pp. 163–169, 1987.

[53] J. Bloomenthal. An implicit surface polygonizer. In P. Heckbert, editor, *Graphics Gems IV*, pp. 324–349. Academic Press, 1994.

[54] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing, 2nd edition.* Cambridge University Press, 1993.

[55] B. Nichols, D. Buttlar, and J. P. Farrell. *Pthreads Programming.* O'Reilly & Associates, Inc., 1996.

[56] J. D. Murray and W. vanRyper. *Encyclopedia of Graphics File Formats.* O'Reilly & Associates, Inc., 1994.

[57] 3D Systems, Inc. *Stereolithography Interface Specification*, 1988.

[58] 松宮雅俊, 竹村治雄, 横矢直和. パーティクルシステムと陰関数曲面を用いた仮想粘土モデリング. 日本バーチャルリアリティ学会第 5 回大会論文集, pp. 225–228, 2000.

[59] 松宮雅俊, 竹村治雄, 横矢直和. パーティクルシステムと陰関数曲面による仮想粘土細工. 平成 12 年電気関係学会関西支部連合大会講演論文集, G15-20, 2000.

[60] M. Matsumiya, H. Takemura, and N. Yokoya. Interactive virtual clay using implicit surfaces and particle systems. In *ACM SIGGRAPH2001 Conference Abstracts and Applications*, p. 211, 2001.

[61] 松宮雅俊, 横矢直和, 竹村治雄. パーティクルシステムと陰関数曲面を用いた対話的仮想粘土モデル. 日本バーチャルリアリティ学会第 6 回大会論文集, pp. 517–518, 2001.

[62] 中村喜代次. 非ニュートン流体力学. コロナ社, 1997.

[63] A. Watt. *3D Computer Graphics*. Addison-Wesley, 2000.

[64] 越塚誠一. 数値流体力学. 培風館, 1997.

[65] D. R. Butenhof. *Programming with POSIX Threads*. Addison Wesley Longman, Inc., 1997.

[66] B. Lewis and D. J. Berg. *Multithreaded Programming with Pthreads*. Sun Microsystems, Inc., 1998.

[67] M. Walmsley. *Multi-Threaded Programming in C++*. Springer-Verlag London, Ltd., 2000.

[68] T. V. Thompson II, D. E. Johnson, and E. Cohen. Direct haptic rendering of sculptured models. In *Proc. ACM Interactive 3D Graphics*, pp. 167–176, 1997.

[69] V. Popescu, G. Burdea, and M. Bouzit. Virtual reality simulation modeling for a haptic glove. In *Proc. IEEE Computer Animation '99*, pp. 195–200, 1999.

[70] W. A. McNeely, K. D. Puterbaugh, and J. J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *Proc. ACM SIGGRAPH '99*, pp. 401–408, 1999.

[71] Immersion Corporation. *Immersion Corporation - 3D Technologies CyberForce*. http://www.immersion.com/products/3d/interaction/cyberforce.shtml, 2001.

[72] P. Lord and B. Sibley. *creating 3-D animation*. Harry N. Abrams, Inc., 1998.

研究業績

1．学会誌および学術専門誌 (原著論文)

1-1 松宮 雅俊, 竹村 治雄, 横矢 直和: “自由形状モデリングのための陰関数曲面を用いた仮想粘土細工システム”, 情報処理学会論文誌, Vol.42, No.5, pp.1151-1160, May 2001.（第 2 章に関連）

2．国際会議論文集

2-1 M. Matsumiya, K. Kiyokawa, H. Takemura, and N. Yokoya: “An immersive modeler for curved objects using implicit surfaces”, *Proc. SPIE, Vol.3833, Intelligent Systems in Design and Manufacturing II*, pp.109-120, Boston, Massachusetts, September 1999.（第 2 章に関連）

2-2 H. Yoshimori, M. Matsumiya, H. Takemura, and N. Yokoya: “Combination of two- and three-dimensional space for solid modeling”, *ACM SIGGRAPH2000 Conference Abstracts and Applications*, p.191, New Orleans, Louisiana, July 2000.

2-3 M. Matsumiya, H. Takemura, and N. Yokoya: “An immersive modeling system for 3D free-form design using implicit surfaces”, *Proc. ACM Sympo. on Virtual Reality Software and Technology 2000 (VRST2000)*, pp.67-74, Seoul, Korea, October 2000.（第 2 章に関連）

2-4 H. Takemura, H. Yoshimori, M. Matsumiya, and N. Yokoya: “An immersive modeling workbench using a combination of two- and three-dimensional interface”, *Proc. 10th Int. Conf. on Artificial Reality and Tele-Existence (ICAT2000)*, pp.195-200, Taipei, Taiwan, October 2000.

2-5 M. Matsumiya, H. Takemura, and N. Yokoya: “Interactive virtual clay using implicit surfaces and particle systems”, *ACM SIGGRAPH2001 Conference*

Abstracts and Applications, p.211, Los Angeles, California, August 2001.（第 3 章に関連）

2-6 H. Kawai, M. Matsumiya, H. Takemura, and N. Yokoya: "Elastic object manipulation using coarse-to-fine representation of mass-spring models", *ACM SIGGRAPH2001 Conference Abstracts and Applications*, p.176, Los Angeles, California, August 2001.

3. 研究会およびシンポジウム発表

3-1 松宮 雅俊, 清川 清, 竹村 治雄, 横矢 直和: "没入型仮想環境における 3 次元曲面形状モデリング", 電子情報通信学会 マルチメディア・仮想環境基礎研究会, 信学技報 MVE98-86, February 1999.（第 2 章に関連）

3-2 吉森 勇人, 松宮 雅俊, 岩佐 英彦, 竹村 治雄, 横矢 直和: "2 次元平面と 3 次元空間の組み合わせによるモデリング環境", 電子情報通信学会 マルチメディア・仮想環境基礎研究会, 信学技報 MVE99-65, February 2000.

3-3 河合 裕文, 松宮 雅俊, 竹村 治雄, 横矢 直和: "疎密バネモデルを用いた柔物体の仮想環境下での視覚および力覚提示手法の提案", 電子情報通信学会 画像工学研究会, 信学技報 IE2000-167, January 2001.

3-4 森本 龍太郎, 松宮 雅俊, 竹村 治雄, 横矢 直和: "没入型仮想環境とペン型デバイスを利用した地形モデリング", 電子情報通信学会 マルチメディア・仮想環境基礎研究会, 信学技報 MVE2000-125, March 2001.

3-5 松宮 雅俊, 竹村 治雄, 横矢 直和: "手による対話操作可能なパーティクルシステムと陰関数曲面を用いた仮想粘土モデル", 電子情報通信学会 高度交通システム研究会，画像工学研究会, 信学技報 ITS2001-36，IE2001-175, January 2002.（第 3 章に関連）

3-6 海部 陸, 松宮 雅俊, 横矢 直和: "局所的に細密な陰関数表現による拡大作業機能を持つ没入型仮想彫刻モデラ", 電子情報通信学会 高度交通システム研

究会，画像工学研究会, 信学技報 ITS2001-35，IE2001-174, January 2002.（第 4 章に関連）

4．全国大会等での口頭発表

4-1 松宮 雅俊, 清川 清, 竹村 治雄, 横矢 直和: “陰関数表現を用いた仮想空間内曲面モデリングシステム”, 1998 年電子情報通信学会総合大会講演論文集, No.A-16-35, March 1998.（第 2 章に関連）

4-2 松宮 雅俊, 清川 清, 竹村 治雄, 横矢 直和: “陰関数表現を用いた仮想空間没入型曲面形状モデラ”, 日本バーチャルリアリティ学会第 3 回大会論文集, pp.53-56, August 1998.（第 2 章に関連）

4-3 吉森 勇人, 松宮 雅俊, 竹村 治雄, 横矢 直和: “2 次元/3 次元空間をシームレスに融合したモデリング環境”, 日本バーチャルリアリティ学会第 4 回大会論文集, pp.353-354, September 1999.

4-4 森本 龍太郎, 松宮 雅俊, 竹村 治雄, 横矢 直和: “没入型自然景観モデラ”, 日本バーチャルリアリティ学会第 5 回大会論文集, pp.139-140, September 2000.

4-5 松宮 雅俊, 竹村 治雄, 横矢 直和: “パーティクルシステムと陰関数曲面を用いた仮想粘土モデリング”, 日本バーチャルリアリティ学会第 5 回大会論文集, pp.225-228, September 2000.（第 3 章に関連）

4-6 河合 裕文, 松宮 雅俊, 佐藤 哲, 山澤 一誠, 竹村 治雄, 横矢 直和: “弾性体のバネモデルの疎密表現による計算量削減手法”, 日本バーチャルリアリティ学会第 5 回大会論文集, pp.229-232, September 2000.

4-7 松宮 雅俊, 竹村 治雄, 横矢 直和: “パーティクルシステムと陰関数曲面による仮想粘土細工”, 平成 12 年電気関係学会関西支部連合大会講演論文集, No.G15-20, November 2000.（第 3 章に関連）

4-8 松宮 雅俊, 横矢 直和, 竹村 治雄: “パーティクルシステムと陰関数曲面を用いた対話的仮想粘土モデル”, 日本バーチャルリアリティ学会第 6 回大会論文集, pp.517-518, September 2001.（第 3 章に関連）

4-9 海部 陸, 松宮 雅俊, 横矢 直和: “局所的に細密なデータ構造を持つ陰関数表現を用いた没入型仮想彫刻モデラ”, 日本バーチャルリアリティ学会第 6 回大会論文集, pp.469-470, September 2001.（第 4 章に関連）

5．表彰

5-1 平成 12 年度情報処理学会関西支部学生奨励賞受賞，2001 年 1 月．

5-2 平成 12 年度電気関係学会関西支部連合大会奨励賞受賞，2001 年 4 月．

付録

A. 陰関数曲面のレイトレーシング法

本論文中のモデリング例のレイトレーシング法による画像生成に使用するために開発した，陰関数曲面のレイトレーシング法について述べる．

まず，図 A.1 で示すように，視点 V_0 とスクリーン上の画素（描画点）との位置関係から，視線ベクトル V を求める．そして，視線が形状と交わる場合，交点 P と光源 L_0 との位置関係から光源ベクトル L を求める．視線ベクトル V，光源ベクトル L，交点 P 上の法線ベクトル N から，フォンモデルによる式（A1）を用いて交点 P における色を求める．

$$
\begin{aligned}
C &= (s \cdot k_d \cos\alpha + k_e) \cdot C_s + s \cdot k_s \cos^n\beta \cdot C_w, \\
where & \\
\cos\alpha &= -L \cdot N, \\
\cos\beta &= 2(L \cdot N)(N \cdot V) - L \cdot V.
\end{aligned}
\tag{A1}
$$

ここで，k_d は拡散反射係数，k_s は鏡面反射係数，k_e は環境反射係数を表す．また，s は入射光の強さ，C_s は物体の表面色，C は求める点の色，C_w は白色成分（赤：1.0，緑：1.0，青：1.0）を表す．

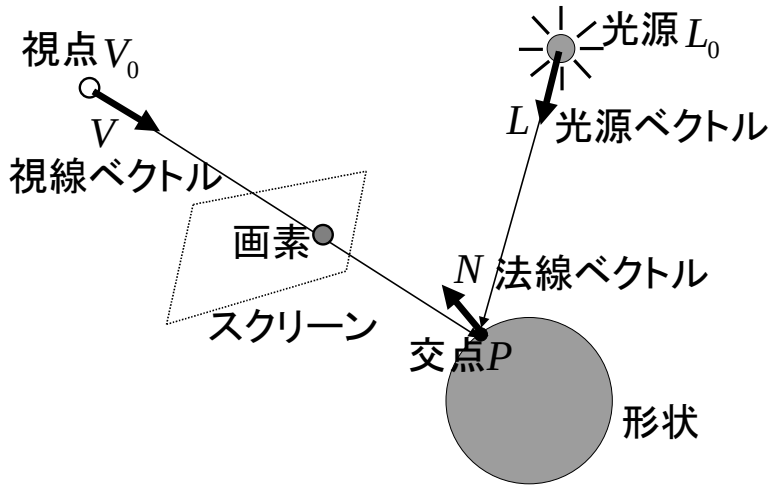


図 A.1 レイトレーシング法

本研究で用いている陰関数曲面（Skeletal Implicit Surfaces）では，視線と形状の交点を解析的に求めることができない．そこで，交点を計算する手法が必要になる．また，計算をできるだけ高速に行える方が望ましい．以下では，本研究で用いている陰関数曲面による形状と視線との交点を求める手法について述べる．

ここで，1つの線スケルトンと1つの点スケルトンからなる陰関数曲面形状と，視線を図 A.2 に示す．

1. すべての点スケルトンから視線に垂線を下ろし，交点を求める．また，すべての線スケルトンの両端点から視線に垂線を下ろし，交点を求める．図 A.2 においては，交点 P_1，P_2，P_3 になる．
2. 視点 V_0 に近いものから順に交点を調べ，以下の処理を繰り返す．
   (a) 交点での陰関数値を求める．
   (b) 交点の内外判定を行い，i) 表面，ii) 外側，iii) 内側で処理を分ける．
      i. 表面の場合には，そのままその交点を視線と形状との交点とする．

ii. 外側の場合には，次の交点に関して処理を行う．すべての交点について外側の判定であれば，視線と形状は交差しないと判断する．

iii. 内側の場合には，その交点と視点 V_0 を用いて形状と視線の交点を挟み撃ち法で求める．図 A.2 においては，視点 V_0 と交点 P_2 を用いて挟み撃ち法により，形状と視線の交点 P を求める．また，その点における法線ベクトル N を中心差分で求める．

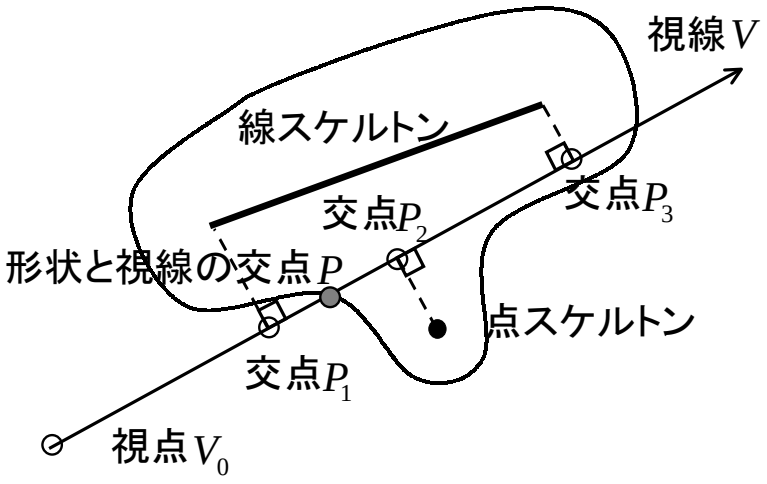


図 A.2 形状と視線との交点の求め方

以上の処理を図 A.1 で示す，スクリーン上の全ての画素について行うことで，レイトレーシング法によるレンダリング画像を得ることができる．すべての処理は画素ごとに独立しているため，画素単位で容易に並列化することができる．

図 A.3 に前述した手法を実装したアプリケーション（Skeletal Implicit Ray Tracer ： sirt）の実行中画面を示す．16 個の CPU を用い，1 つの CPU が縦 1 列の画素列を上から順に処理を行うため，図に示すように描画されていく．

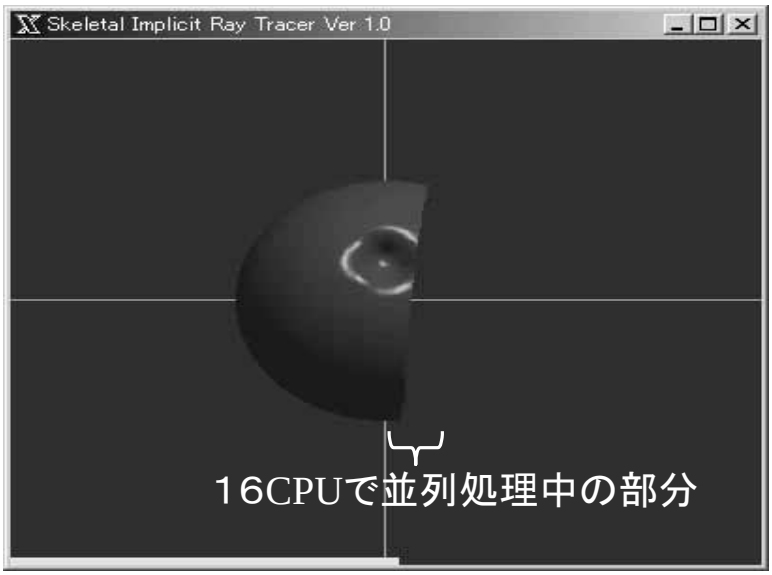


図 A.3 Skeletal Implicit Ray Tracer の実行画面

B. 補助アプリケーション：sim2dxf，sim2stl

本論文で述べたモデリングシステムで作成した 3 次元形状を他のアプリケーションで利用するために，専用の陰関数表現記述フォーマットである SIM（Skeletal Implicit Model）を定義するとともに，一般的なポリゴン表現記述フォーマットである DXF や STL（SLA）への変換ツールである sim2dxf と sim2stl を作成した．sim2dxf と sim2stl では，SIM ファイルを入力とし，引数で与える任意のポリゴン変換ボクセルの大きさで陰関数表現をポリゴン表現に変換し，DXF および STL ファイルとして出力する．ここでは，参考のために，これらのフォーマットと例を掲載する．

SIM フォーマット

1

スケルトンの種類（点，線の別および距離場関数の種類を示す整数）

10

第 1 頂点の X 座標

20

第 1 頂点の Y 座標

30

第 1 頂点の Z 座標

11

第 2 頂点の X 座標（線スケルトンの場合）

21

第 2 頂点の Y 座標（線スケルトンの場合）

31

第 2 頂点の Z 座標（線スケルトンの場合）

100

0

:

:

繰り返し

:

:

例

```
1
1
10
0.373921
20
-2.322046
30
0.020310
11
0.179444
21
-2.524822
31
0.433907
100
0
    :
    :
```

DXF フォーマット

0
SECTION
2
ENTITIES
0
3DFACE
8
SIM
62
2
10
第 1 頂点の X 座標
20
第 1 頂点の Y 座標
30
第 1 頂点の Z 座標
11
第 2 頂点の X 座標
21
第 2 頂点の Y 座標
31
第 2 頂点の Z 座標
12
第 3 頂点の X 座標
22
第 3 頂点の Y 座標
32
第 3 頂点の Z 座標

13
第 4 頂点の X 座標（第 3 頂点と同じ値）
23
第 4 頂点の Y 座標（第 3 頂点と同じ値）
33
第 4 頂点の Z 座標（第 3 頂点と同じ値）
　　:
　　:
　繰り返し
　　:
　　:
0
ENDSEC
0
EOF

例

```
0
SECTION
2
ENTITIES
0
3DFACE
8
SIM
62
2
10
-1.701883
20
-0.100000
30
-0.100000
11
-1.700000
21
-0.155445
31
-0.100000
12
-1.701506
22
-0.101506
32
-0.101506
```

```
13
-1.701506
23
-0.101506
33
-0.101506
    :
    :
0
ENDSEC
0
EOF
```

STL フォーマット

```
solid モデル名
  facet normal (X) 数値データ (Y) 数値データ (Z) 数値データ
    outer loop
      vertex (X) 数値データ (Y) 数値データ (Z) 数値データ
      vertex (X) 数値データ (Y) 数値データ (Z) 数値データ
      vertex (X) 数値データ (Y) 数値データ (Z) 数値データ
    endloop
  endfacet
  facet normal (X) 数値データ (Y) 数値データ (Z) 数値データ
    outer loop
      vertex (X) 数値データ (Y) 数値データ (Z) 数値データ
      vertex (X) 数値データ (Y) 数値データ (Z) 数値データ
      vertex (X) 数値データ (Y) 数値データ (Z) 数値データ
    endloop
  endfacet
                          :
                          :
                       繰り返し
                          :
                          :
endsolid モデル名
```

例

```
solid SIM
  facet normal 9.620935e-01 2.054036e-01 1.794027e-01
    outer loop
      vertex -1.657095e+00 -3.500000e-01 -1.500000e-01
      vertex -1.650000e+00 -3.832306e-01 -1.500000e-01
      vertex -1.655068e+00 -3.550677e-01 -1.550677e-01
    endloop
  endfacet
  facet normal 9.622563e-01 1.923493e-01 1.925221e-01
    outer loop
      vertex -1.657095e+00 -3.500000e-01 -1.500000e-01
      vertex -1.655068e+00 -3.550677e-01 -1.550677e-01
      vertex -1.655912e+00 -3.500000e-01 -1.559118e-01
    endloop
  endfacet
                          :
                          :
endsolid SIM
```