OpenGLによる布アニメーションの試み（第Ⅱ報）

坂本　眞人[1]・長友　世依良[2]・谷上　亜由美[3]

Attempt of Cloth Animation by OpenGL Ⅱ

Makoto SAKAMOTO [1], Seira NAGATOMO [2], and A yumi TANIUE [3]

Abstract

Cloth motion has recently become a topic of large investigation. For example, in fact, virtual garments or virtual cloth draping effects are recurrent visualization elements necessary for CG animation. Moreover, in the apparel and home fabric furniture production, modeling tools are widely demanded to assist the process of cloth design and make it faster [1]. However, we come across many dynamic problems when we study the cloth motion. Above all, the collision detection and response of the cloth motion is one of the difficult studies. Collisions occur, for instance, when parts of flexible objects such as the garments hit some rigid objects such as the mannequins or penetrate towards each other (selfcollisions). We have continued to study the cloth animation by CG. In this paper, we mainly show the results concerning the collision detection and response. The algorithm is implemented in the OpenGL and the C++ on a personal computer.

Key Words:
Cloth Animation, Collision Detection, Computer Graphics, Mass-Spring Model, OpenGL

1 はじめに

近年，コンピュータ技術の進歩やハードウェア性能の向上により 3 次元コンピュータグラフィックス（3DCG）技術が大きく発展してきている。3DCG は映画やデザインツール，ゲームといった様々な分野に応用され，また個人のコンピュータでも製作されるようになってきた。

過去の研究[4]で，3DCG アニメーションへ応用される布の動的な表現を，物理モデルを用いて行った。これにより，布の形状や風により布がなびく表現のアニメーション化を実現した。

しかし，まだ自己衝突や曲げ特性の表現，布の厚みの表現などの未実装といった多くの問題点が残されている。本研究ではその中でも衝突判定と衝突応答の未実装の解決に取り組み，この取り組みを反映させたシミュレーションを行う。

2 シミュレーション手法

布を表現するためのモデリング手法は大まかに幾何学モデルと物理モデルの 2 種類に分けられる。

動的なシミュレーションを行うことを考えると，ある程度簡略化したモデルとして計算できる物理

1） 情報システム工学科准教授
2） 情報システム工学科学部生
3） 情報システム工学専攻大学院生

モデルが今回の研究テーマに適しているだろう。

2.1 マススプリングモデル

物理モデルとして一般に多く用いられるのは**マススプリングモデル**である。マススプリングモデルでは布の持つ異方性を表現するため，布の縦糸と横糸に沿った方向を軸とし，正方形の格子状に配置した質点と，近接する質点を互いにつなぐバネで，布の形状を構成する（図 2.1 参照）[5]。

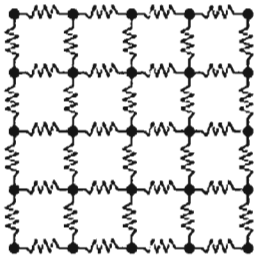

図 2.1：マススプリングモデル[5].

2.2 質点にかかる力

時刻tにおける質点iにかかる力の合力$F_i(t)$は，風力f_{wind}や重力$f_{gravity}$などの外力と，バネの振動に対する減衰力f_{damper}やバネによる力$f_{stretch}$といった布の内部構造より生じる力から以下のように表せる。

$$F_i(t) = f_{gravity} + f_{dampar} + f_{stretch} + f_{wind}$$

2.2.1 バネによる弾性力

質点i，質点jの座標を$P_i(x_i, y_i, z_i)$，$P_j(x_j, y_j, z_j)$とする。ここで，質点jが質点iに加える(及ぼす)力をf_{ij}とすると，f_{ij}はフックの法則により以下の式で表すことができる。

$$f_{ij} = k\left(\left|P_{ij}\right| - L\right)\frac{P_{ij}}{\left|P_{ij}\right|}$$

$$= \frac{k\left(\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2} - L\right)}{\sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}} \begin{bmatrix} x_j - x_i \\ y_j - y_i \\ z_j - z_i \end{bmatrix}$$

ただし，$P_{ij} = P_j - P_i$

（P_i，P_j：質点i，jの位置ベクトル，

k：バネ定数，L：バネの自然長）

質点iに加わる(最大 12 本の)バネによる力の合力を$f_{stretch} = f_i$として表す。

2.2.2 重力

モデルを構成する質点に加わる重力は次式で与えられる。

$$f_{gravity} = m_i g$$

（m_i：質点の質量，g：重力加速度）

2.2.3 ダンパー

このままだと，バネが永遠に振動してしまうため，バネの振動を減衰させるための力を加える。質点iの速度v_iの大きさに比例し，逆向きに働く力とする。

$$f_{damper} = r_i = -\rho v_i$$

（ρ：抵抗係数）

2.2.4 風力

シミュレーションをするにあたり，布がなびいているように振舞わせるために，風力f_{wind}を加える。風力は 3 次元仮想空間上の z 軸方向の力として質点に加えられるものとする。

今回は以下の式のように sin 関数を組み合わせたものを使用した。

$$f_{wind} = \alpha \sin(\beta)\sin(\gamma)$$

（α：任意定数，$\beta = 100t/85.0$，$\gamma = 100t/65.0$）

従って，合力F_iは以下の式のようになる。

$$F_i(t) = f_{gravity} + f_{dampar} + f_{stretch} + f_{wind}$$
$$= m_i g + r_i(t) + f_i(t) + \alpha \sin(\beta)\sin(\gamma)$$

2.3 運動方程式

運動方程式に基づき加速度 $a_i(t)$ は次の式で表される。

$$a_i(t) = \frac{1}{m_i} F_i(t)$$

2.4 オイラー陽解法

2.2 節で計算した質点に働く力を用いて，速度 v_i，座標 x_i は次の式で求められる。

$$v_i(t+\Delta t) = v_i(t) + a_i(t)\Delta t$$
$$x_i(t+\Delta t) = x_i(i) + v_i(t)\Delta t$$

（Δt ：時間ステップ）

2.5 自己衝突

今回，クロスシミュレーションを行う上で**自己衝突**を考慮する。布の自己衝突は布を構成する三角ポリゴンと質点との衝突と辺同士の衝突の 2 種類に分類されるが，本研究では質点とポリゴンの衝突のみを考えることにする。

2.5.1 衝突検出

2.5.1.1 階層的バウンディングボックス

ある物体を覆う形状を**バウンディングボリューム**といい，導入する利点はバウンディングボリューム間での重なり判定を容易に行えることである。まずこれを用いて簡易的に衝突判定を行い，距離の離れたポリゴン同士の無駄な計算を減らす。

バウンディングボリュームはいろいろな種類のものがあるが，本研究では，重なり判定やバウンディングボリュームの更新の計算コストを抑えるため，単純な形状の**軸平行バウンディングボックス**を用いる。軸平行バウンディングボックスとは直方体を構成する辺の 3 方向が空間座標系の XYZ 軸に平行なもののことであり，以下これを**バウンディングボックス**と呼ぶ。

バウンディングボックスをその包含関係で階層化し（図 2.2 参照），各葉ノードではバウンディングボックス 1 つにつきポリゴン 1 つを覆うものとする。ある階層のある 2 つのバウンディングボックス同士が重ならなければ，その子ノードのいずれのバウンディングボックス同士も重なりあわないので，効率化を図れる。質点同様に，バウンディングボックスも時間ステップごとに更新する。

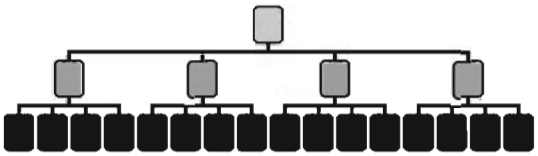

図 2.2：子ノード 4 個の階層的バウンディングボックス.

また，あるバウンディングボックス内において，表面の法線のうち 2 つのなす角度で一番大きいものが π 未満のときは，そのバウンディングボックスの自己衝突は起こりえないため，このノード以降，階層構造を降りていく必要はない。

2.5.1.2 質点とポリゴンの衝突検出

バウンディングボックスを用いて葉ノード同士の衝突が検出されたら，それらに含まれる質点とポリゴンの衝突検出を行う。

衝突している条件は，質点からポリゴンが存在する平面に下ろした垂線の足がポリゴン上に存在することと，質点からポリゴンまでの距離が布の厚さ ε 以下であることである。

ここで，ポリゴンを構成する 3 頂点の座標を x_0, x_1, x_2，質点の座標を x_3 として，質点から平面に降ろした垂線の足の座標を $x_4 = \varpi_0 x_0 + \varpi_1 x_1 + \varpi_2 x_2$ とする。また，これらの係数間には $\varpi_0 + \varpi_1 + \varpi_2 = 1$ の条件が成り立つ。以下，$x_{ij} = x_j - x_i$ とする。

まず x_{24} はポリゴンが存在する平面上にあるので x_{20} と x_{21} を用いて $x_{24} = \varpi_0 x_{20} + \varpi_1 x_{21}$ と表せる。ここで $x_{24} = x_{23} + x_{34}$ であり，$x_{34} \perp x_{20}, x_{34} \perp x_{21}$，つまりこれらの内積は 0

なので，先の式に x_{20} と x_{21} をかけることで以下の2式が導かれる。

$$x_{20}\cdot x_{23} = \varpi_0 x_{20}\cdot x_{20} + \varpi_1 x_{20}\cdot x_{21}$$
$$x_{21}\cdot x_{23} = \varpi_0 x_{21}\cdot x_{20} + \varpi_1 x_{21}\cdot x_{21}$$

これらと係数の関係式から $\varpi_0, \varpi_1, \varpi_2$ を求める。そしてこれらがすべて $0\sim1$ の範囲に存在するとき，垂線の足がポリゴン上にあることになるが，ポリゴンのエッジに外から近づく質点を検出できるよう閾値 δ を定め $-\delta\sim1+\delta$ の範囲を用いる。

質点からポリゴンまでの距離 d はポリゴンの法線 n を用いて以下のようになる。

$$d = \left|x_{34}\right| = \left|n\cdot x_{23}\right|$$

2.5.2 衝突応答

布を構成する質点とポリゴンの衝突を検出した後は，非弾性衝突をする力と，めり込みを解消する力の 2 つの力による力積を作用させることで衝突応答させる。

ポリゴン上の衝突点の速度 v_4 は，ポリゴンを構成する質点の速度 v_0, v_1, v_2 と先に求めた $\varpi_0, \varpi_1, \varpi_2$ を用いて，次の式で求められる。

$$v_4 = \varpi_0 v_0 + \varpi_1 v_1 + \varpi_2 v_2$$

まず非弾性衝突をする力積について考える。ポリゴン上の衝突点にポリゴン法線方向の力積 I が働くと，すべての質点の質量が同じで m のとき，ポリゴンを構成する質点の速度は次のようになる。

$$v_0' = v_0 + \varpi_0 \frac{I}{m}$$
$$v_1' = v_1 + \varpi_1 \frac{I}{m}$$
$$v_2' = v_2 + \varpi_2 \frac{I}{m}$$

すると衝突後の衝突点の速度は，それらの重み付け和であるので，次のようになる。

$$v_4' = v_4 + \left(\varpi_0^2 + \varpi_1^2 + \varpi_2^2\right)\frac{I}{m}$$

また質点の速度は以下のようになる。

$$v_3' = v_3 - \frac{I}{m}$$

衝突後の質点とポリゴン上の衝突点の法線方向の速度が等しくなるので v_3' と v_4' の式より力積 I は以下の式で求まる。

$$I = m\frac{\left(v_3 - v_4\right)\cdot n}{1+\varpi_0^2+\varpi_1^2+\varpi_2^2}\frac{n}{|n|}$$

めり込みを解消する力は Δt の間に $\varepsilon - d$ だけ n の方向に動くようにする必要があるので，この力を加えると，質点と衝突点の法線方向の絶対速度が $(\varepsilon - d)n/\Delta t$ になる。この条件は

$$\left(v_3' - v_4'\right)\cdot n = \frac{\left(\varepsilon - d\right)}{\Delta t}$$

以上のことを用いて衝突応答の力積 I' は以下の式に表せる。

$$I' = I - \frac{m\left(\varepsilon - d\right)}{\Delta t}\frac{1}{1+\varpi_0^2+\varpi_1^2+\varpi_2^2}\frac{n}{|n|}$$

3 プログラム内容

本プログラムは，C/C++言語を用いたプログラム開発環境において，標準的な CG ライブラリである OpenGL を用いて作成した。今回プログラムの作成には，Microsoft Visual Studio 2008 Professional Edition を使用した。

3.1 OpenGL

OpenGL とは，高品質カラーイメージ，リアルタイム処理，グラフィックスカード対応といった特徴を有するグラフィックスライブラリである。座標系は右手座標系を採用し，点，線，ポリゴンなどの 3D プリミティブの作成，移動，回転，拡大・縮小，ライティングとシェーディング，テクスチャマッピングなどの機能をもつ。

3.2 モデリング

質点座標を頂点とし，三角ポリゴンを生成する。

今回，3 次元仮想空間における X-Y 平面上に初期座標をとる(図 3.1 参照)。

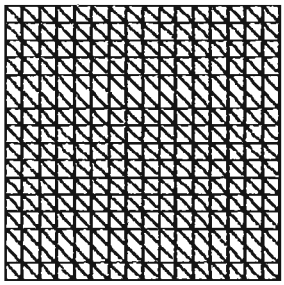

図 3.1：17×17 個の質点による布モデル.

以下，質点座標は 2.3 節で記述した方法で求められるので，それをもとに繰り返しポリゴンを生成する。

3.3 シェーディング

シェーディングとは光源とオブジェクトの形状などを元に，モデルに陰影をつけることである。今回はスムーズシェーディングを行った。OpenGL では，スムーズシェーディングの方法として，**グーローシェーディング**を採用している。

グーローシェーディングでは，立体を構成する面の明るさ１つ１つをディフューズ（方向性のない拡散反射）で計算し，ポリゴン面を構成する質点１つ１つの明るさを，隣り合う面の明るさの補間によって求める[3]。

プログラムでは，求めた各頂点の法線ベクトルの指定を行えば，OpenGL で提供されているコマンドでグーローシェーディングが実行される。

3.4 テクスチャマッピング

テクスチャとは元来，織物の質感を意味する。CG の分野では図形の表面につけられた模様や，質感を表現するための描き込みを指す。

テクスチャマッピングとは，オブジェクトの表面にテクスチャの画像データを貼り付けることで，物体の質感を向上させることをいう。

今回は，24bit の bmp 形式の画像データ(図 3.2 参照)を用いてテクスチャマッピングを行った。

図 3.2：画像データ.

3.5 アニメーション

OpenGL では，ディスプレイコールバック関数を呼び出すことにより，この関数内で設定されたオブジェクトやシーンが描画される。

アニメーションは，オブジェクトや視点を少しずつ動かす毎に描画を繰り返さなければならないので，このディスプレイコールバック関数を繰り返して呼び出すことになる。この関数を繰り返して呼び出す方法として OpenGL ではアイドル状態(イベントがない場合)での描画と一定時間毎の描画の 2 通りの方法を提供している。

今回は，アイドル状態で描画を繰り返す(描画更新を行う)方法を用いてアニメーションを行う。

4 シミュレーション結果

本研究では，プログラムは C++言語により作成し，Intel Pentium D（OS：Windows XP Professional）上で実行した。

4.1 実行結果

シミュレーション結果を下図(図 4.1 参照)に示す。同図の左のように自己衝突を起こした後，そのまますり抜けずに右のように衝突応答が可能になった。

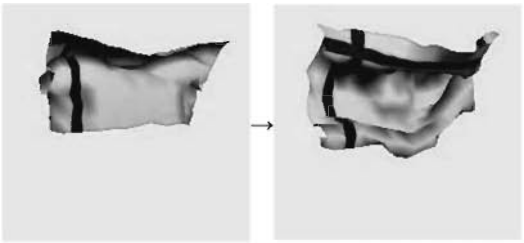

図 4.1：実行結果（左：自己衝突、右：衝突応答）.

4.2 考察

シミュレーションの結果，ゆっくりとした穏やかな衝突に関しての衝突応答はできていたが，風を強くするなどですばやく布が動き衝突が起きたとき，めり込みを解消することはできなかった。衝突検出は質点とポリゴンの位置関係でしか行っていないため，Δt の間に布をすり抜け，検出できなかったものと思われる。これは曲げに抵抗する力を布に与えたり，あるいは時間積分の方法を変更し，動的シミュレーションとして十分な速さで更新できるような Δt に対応させたりすることで解決しうる。

質感の表現に関しては，グーローシェーディングにより布のような柔らかな陰影が表現できた。しかし，モデルには曲げに抵抗する力を含めていないので，とても柔らかい布の表現しかできなかった。また，布特有の表面の光の拡散は今回実装しておらずリアルな表現とはいえない結果になった。

今後の課題としては，上に述べたように，曲げ特性の再現や数値計算法の見直しが挙げられる。

5 おわりに

今回は一般的に多く用いられるマススプリングモデルを使用してクロスシミュレーションを行ったが，実際の織物の糸構造から計算を行っている研究もある[1]。

本研究では自己衝突について考慮し，三角ポリゴンと各質点との衝突検出を行い，それに対する衝突応答を実装した。

実行結果から，比較的ゆっくりとしたシミュレーションにおいては自己衝突の検出ができ，衝突応答を行うことができたと考えられる。しかし，時間ステップが大きいと衝突検出ができず，衝突応答をすべき時に行えないためにすりぬけが起こってしまった。

考察でも述べたが，ポリゴンと辺の衝突を検出できなかったことも，すり抜けの一因と考えられる。また，曲げバネの実装が不十分だったため曲げ特性の再現ができていない。

今後は，以上で述べた問題点の改善を含め，その他の課題への取り組みを行い，よりリアルな布の表現を目指したい。

参考文献

[1] U. Cugini, “Cloth Shape Design”, Special Issue of Japanese Society for the Science of Design, Vol.15-4, No.60, pp.11-24, 2008.

[2] 井上真理, “繊維集合体としての布の特性について”, SEN'I GAKKAISHI(繊維と工業), 2008.

[3] 今間俊博, “CG 基礎セミナー 3DCG・映像メディアに強くなる”, オーム社, 2002.

[4] 坂本眞人, 谷上亜由美, “OpenGL による布アニメーションの試み”, Memoirs of the Faculty of Engineering, University of Miyazaki, No.37, pp.299-304, 2008.

[5] 田川和義, 林宏卓, 木島竜吾, 小鹿丈夫, “真空内での布の挙動の比較による布シミュレーションの精度検証”, 日本バーチャルリアリティ学会第 7 回大会論文集, 2003.