2007年度　卒　業　論　文

複数枚の折り紙を考慮した折り紙シミュレーション

指導教員：渡辺 大地講師

メディア学部 ゲームサイエンスプロジェクト
学籍番号　M0104287
寺本 彩

2007 年度　　卒　業　論　文　概　要

論文題目

複数枚の折り紙を考慮した
折り紙シミュレーション

メディア学部 **学籍番号：M0104287**	**氏名**	寺本 彩	**指導教員**	渡辺 大地講師

キーワード	折り紙シミュレーション、Half-Edge 構造、3DCG、Z ファイティング

折り紙は日本に古くから伝わる遊戯であり、とても簡単な折り方のものから、複雑な設計のものまで様々な作品が作り出されている。また、近年では数学的な研究のテーマとしても取りあげられるようになり、多くの研究がなされている。折り紙をコンピュータ上で再現する「折り紙シミュレーション」に関連する研究は以前から行われており、ユーザーの入力に基づきリアルタイムな処理を行うものも存在している。

折り紙シミュレーションの実装には、紙の構造をどのように扱うかによって複数のアプローチが考えられる。しかし、現行のリアルタイムな処理を行う折り紙シミュレーションにおいて、履歴の正確な管理、紙同士の重なり順に従った交差のない面の描画、折り紙として矛盾のない形状の維持の 3 点を同時に解決しているものがない。また、折り紙作品には 2 枚以上の折り紙を組み合わせて作成する作品が存在するが、従来の対話的な処理が行える折り紙シミュレーションの場合、使用できる折り紙は現状では 1 枚に限られている。そこで本研究では、コンピュータ上での折り紙シミュレーションをより汎用性の高いものとすることを目的として、従来の折り紙シミュレーションの持つ課題の解決と、複数枚の折り紙を扱えるシミュレーションの作成を行った。

目 次

図 目 次

第 1 章

はじめに

折り紙は日本に古くから伝わる遊戯として浸透している。手先を使う遊びとして広い世代から受け入れられている以外にも、紙の持つ幾何学的な特徴に着目した研究 [1][2] が各分野の研究者により進められ、近年では数学アルゴリズムや工学分野への応用 [3][4] が行われている。

コンピュータ上で折り紙を再現する「折り紙シミュレーション」は今までにも研究 [5][6][7] が行われている。マウスやキーボードなどのデバイスを用いて、リアルタイムに折り紙の形状を操作できるシミュレーションが既に存在しているほか、紙の厚さや重なりを可視化することで構造の把握を容易にするための研究 [8][9] も行われている。折り紙シミュレーションの実装には、紙の構造をどのようにデータとして扱うかによって複数のアプローチが考えられる。

宮崎らが行った研究 [10] では、同一平面上に属する面を面グループとしてまとめて管理した上で、折り操作に伴う面、辺の分割を階層構造で記録するという手法を用いている。この手法では、同一平面上に存在する面の重なり順を正確に管理し、複数の面が 3 次元空間上の同じ位置に存在する場合でも表面に出る面のみを描画することができ、階層構造によるデータ保持で、正確な履歴管理を高速で行うことができる。しかし、この研究では異なる平面上に属する要素同士の位置関係を正確に保つことができない。

古田らが行った研究 [11] では、折り紙の頂点間の距離をバネモデルによって管

理している。各頂点間の距離を一定に保つことで、辺の長さや面の形状も保つことができるが、常に物理演算が行われているため、履歴を設定しづらい。また面を重ね合わせたときに、本来は奥側にあるはずの面が手前に描画されたり、複数の面のテクスチャが混ざって描画され画面がちらつくことがある。この現象は複数の面が同一平面上かそれに近い状態で存在したときに、カメラから見てどの面が一番手前に存在するか (どの面を画面に描画するか) を正確に判断できないために発生する。この現象はＺファイティングやステッチング [12] などの名称で呼ばれる。図 1.1 は 2 つの面が同一平面に位置し、ちらつきが発生しているモデルの例である。

図 1.1: ちらつきが発生しているモデルの例

宮崎らの研究に見られる「正確な履歴管理」と「表面の正確な描画」という利点と、古田らの研究に見られる「折り紙形状の整合性の維持」という利点を合わせ持った折り紙シミュレーションは、現行ではあらかじめ折り線のデータを入力し計算するものにとどまっており、ユーザーの入力に基づき順次折り線を追加できる形にはなっていない。

また、折り紙作品の中には複数の折り紙をパーツとして組み合わせることで完成する作品が存在する。組み合わせに使用する折り紙の形状により「複合折り紙」

や「ユニット折り紙」と呼ばれるが、現在の折り紙シミュレーションでは「折る」部分と「組み合わせる」部分が別々のシミュレーションとしてしか実装されておらず [13]、複数枚の組み合わせが必要な作品をシミュレーションすることができない。

そこで本研究では折り紙シミュレーションにおいて、正確な履歴管理、表面の正確な描画、折り紙形状の整合性の維持を同時に解決することと、複数のパーツを組み合わせて作られる折り紙作品をシミュレーションすることの 2 点を目的とする。

本研究では、折り紙の各面を同一の平面に存在するものでグループ化して管理する従来手法を用いて、グループごとに紙の重なりを管理し、モデル形状の構築には Half-Edge 構造を用いることで、要素間の位置関係の管理を行った。

本論文では、第 2 章で本研究で用いた折り紙のデータ構造について述べ、第 3 章で折り紙データの参照・更新によって折り操作を行う過程を示す。第 4 章で、折り紙シミュレーションの実装結果を示し、結果の考察、今後の展望を述べる。

第 2 章

折り紙のデータ構造

本研究の実装では、折り紙形状の面、辺、頂点の各要素を Half-Edge 構造により保存し、各面に対し、宮崎らの研究 [10] で述べられている同一平面上の面をグループ化して管理する手法を用いることで、折り紙の形状データを保持している。

本章では、Half-Edge 構造について述べた上で、折り紙モデルの構造、各面での表面の描画、折り操作の履歴管理について述べる。

2.1 Half-Edge 構造を用いたデータ管理

Half-Edge 構造 [14] は、「ループ」「稜線」「半稜線」「頂点」の 4 要素によってモデルの形状を構成するデータ構造である。それぞれの要素は自身に関連する要素のデータを適量保持している。図 2.1 は Half-Edge 構造で表現した図形の例である。図 2.1 中の V_1、V_2、V_3 が頂点、E_1、E_2、E_3 が稜線、H_1、H_2、H_3、H_4、H_5、H_6 が半稜線、L_1 がループである。

以下に各要素の解説を述べる。

頂点 (Vertex)

形状中の 1 点を表す。単独で存在する場合と、稜線の端点となる場合がある。自身の 3 次元空間上の位置、自身を始点とする半稜線のうち 1 つを加えて保持データとしている。図 2.1 の V_1 の場合、自身の位置ベクトル、半稜線 H_2 か H_3 どちらかのデータを保持している。

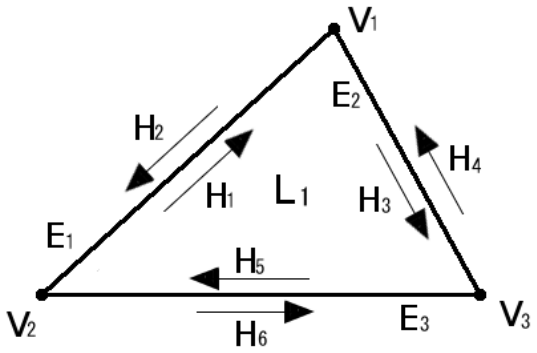

図 2.1: Half-Edge 構造で表現された図形

稜線 (Edge)

形状中の面の境界線を表す。両端点には必ず頂点があり、各点を始点とし逆側の端点への向きを持った半稜線をデータとして保持している。図 2.1 の E_1 の場合、半稜線 H_1と H_2 のデータを保持している。

半稜線 (Half-Edge)

各稜線に 2 個ずつ属している。稜線の端点のうち片方を始点とし、逆側の端点への向きを有している。親稜線、始点の頂点、前後に連結している半稜線、属するループのデータを保持している。半稜線同士の連結関係とループについては、2.1 の中で述べる。図 2.1 の H_1 の場合、親稜線 E_1、始点である頂点 V_2 のデータを保持し、もし H_1 がループ L_1 に属していた場合は、前後の半稜線 H_3と H_5、ループ L_1 のデータも加えて保持している。

ループ (Loop)

形状中の面を表す。ループは複数の半稜線の連結により表現されており、連結している半稜線間には連結の前後関係が存在している。半稜線同士の連結は自身の終点の頂点を始点とする半稜線、始点の頂点を終点とする半稜線との間で行われる。ループは自身を構成する半稜線のうち 1 つをデータとして保持している。図 2.1 の L_1 の場合、L_1 が半稜線 H_1、H_3、H_5 の連結で構成

されているなら、3 つの半稜線うちどれか 1 つのデータを保持しており、この半稜線から前方または後方の半稜線を順次検索していくことで、ループに属する全ての半稜線のデータを取得することができる。H_2、H_4、H_6 の連結で構成されている場合も同様である。本研究では、折り紙の折り線で分割されたそれぞれの面を 1 つのループで扱う。

2.2 複数枚の折り紙を考慮した折り紙モデルの構造

本研究で用いる折り紙モデルは、前述の Half-Edge 構造により構成された 3D モデルを用いて形状を管理している。今回の実装では、表現する折り紙が複数枚の場合でも 1 つの 3D モデルで形状の管理を行っている。これは異なる紙同士の関係性を管理する際に、対象となる紙の形状情報が同一のモデル内に存在する方が紙同士の位置関係の把握を容易に行えるためである。形状の管理に 3D モデルを用いた上で、同一平面上に位置するループをまとめて管理する FaceGroup クラスを用いている。FaceGroup クラスは宮崎らの先行研究 [10] で提案された手法である。それぞれの FaceGroup クラスは同一平面に属する面を、属する面の数に関わらずまとめて管理する。また、FaceGroup では、自身に属する面が FaceGroup 内でどのような順番で重なっているかを記憶している。厚みを考慮せずに折り紙のシミュレーションを行う場合、複数の面が同一の位置に存在するケースが存在し、描画処理や折り操作を正確に行うためには、それぞれの面同士がどのような順番で重なっているかを把握しておく必要があるためである。重なり順の管理法として、宮崎らの手法では、各 FaceGroup が自身に属する面の重なり順をスタック構造により管理している。本研究では、各 FaceGroup が自身に属する面の重なり順を双方向リスト構造により管理している。双方向リスト構造を採用した理由は、新たな面を FaceGroup に追加する際、重なり順の前方・後方両方から検索を行えるほうが検索や追加の効率が良いためである。それぞれの面がどのグループに属するかという情報や、面の重なり順の情報が折り操作によって変化する過程は、3 章で述べる。

2.3 重なり順に基づく表面の描画

本研究では、画面に折り紙を描画する際に、折り操作に基づく形状の変形を行うモデルとは別に画面描画用のモデルを用意している。描画用のモデルは、折り操作を行うモデルに属する各ループと同一位置に 1 枚ずつ分割したループを作成する。これは、1 つの頂点を複数のループで共有し頂点位置の移動により頂点を共有する全てのループの形状に影響が出てしまうという現象を避けるためである。画面に折り紙形状を描画する際は、各 FaceGroup クラスのもつループの重なり順の情報をもとに描画用モデルの各ループの位置をずらし、表面に出るもののみカメラに映るように変更した上で描画用モデルを画面に描画する。この操作により、同一平面上に複数のループが存在するときでも各ループの表面に表れる部分のみ描画することができる。図 2.2 は描画用モデルをカメラ位置から見た画像と、同じモデルを横から見た画像である。

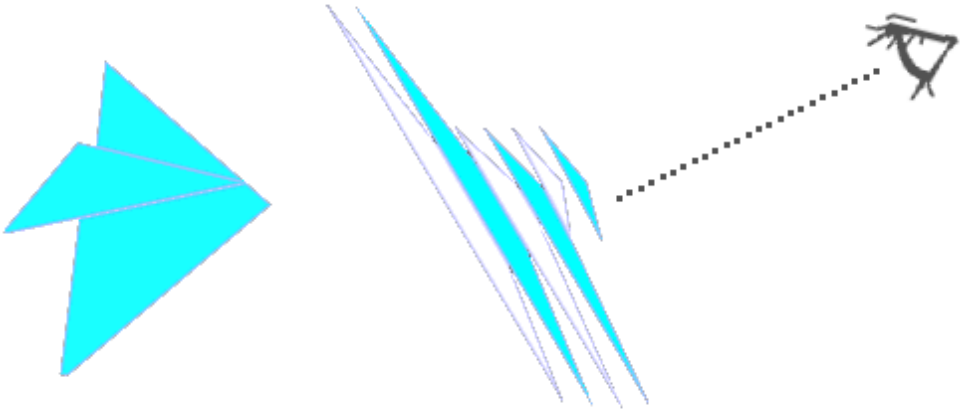

図 2.2: 描画用モデルをカメラ位置から見たものと、横から見たもの

2.4 履歴の設定

折り紙の形状モデルの履歴操作は実装環境のツールキットである FK System [15] の履歴管理機能を利用している。FK System では形状の変形処理をオイラー操作

[16][17] によって行っている。オイラー操作とは、図形に属する頂点、稜線、面の数に成り立つ関係を定めたオイラーの多面体定理に基づいて行う形状の変形操作である。オイラー操作ではそれぞれの操作に 1 対 1 で対応する逆操作が存在し、オイラー操作による形状変形後に行った操作の逆操作を実装することで、形状を確実に元の状態に戻すことができる。

第 3 章

折り操作に伴うデータの参照・更新

折り紙モデルにおける「折り目をつける」、「折る (面を回転させる)」という動作は、本研究の形状モデルに対して行う変形操作を基準に考えると、それぞれ「ループを分割する」、「ループに属する頂点を移動する」と言い換えることができる。本研究の実装プログラムにおいて、ユーザーの折り操作の指定方法は以下の 2 通りである。

1. 移動する頂点、頂点の移動後の位置、折りたたみの角度を指定する。

2. 回転軸となる 1 辺、移動する頂点、折りたたみの角度を指定する。

どちらの指定方法による折り操作も回転軸を中心とした頂点の移動であることには違いはなく、上記 1 の方法を用いた場合は、3.2 で述べる手法で自動的に回転軸を求める。指定された頂点を移動する過程で折り紙の形状として矛盾が生まれないよう、各頂点の移動、ループの分割といった処理を行う。本章では、折り操作によって移動するループの探索や、回転軸の求め方、頂点の移動後の位置の計算、重なり順の更新について述べる。

3.1 各ループが変形する条件

折り操作の際に、各ループが分割や移動を行うかどうかの判断方法は、行う折り操作が 1 つの平面上で完結するという条件の下で、宮崎らの研究 [10] の中で述

べられている。本研究では先行研究と行う折り操作が違うことを考慮したうえで、折り操作により変形が生じるループを以下に示す。なお、図3.1、3.2はそれぞれ、頂点V_1、V_2を選択して 点線の位置を回転軸にした折り操作を行う場合の折り紙モデルである。

- ユーザーが移動元として選択した頂点が属しているループ
  (図3.1のL_2、図3.2のL_1、L_2)
- 既知の変形ループの移動部分と辺を共有しているループ
  (図3.1のL_1)
- 既知の変形ループの移動部分と重なり、ループが折られる方向から見たときに変形するループより重なり順が上になるループ
  (図3.2のL_3)

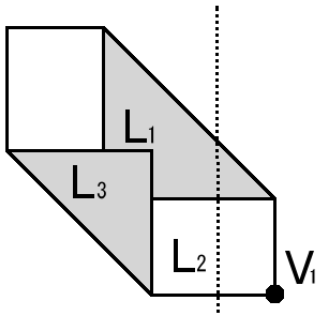

図3.1: 移動する頂点を含むループ

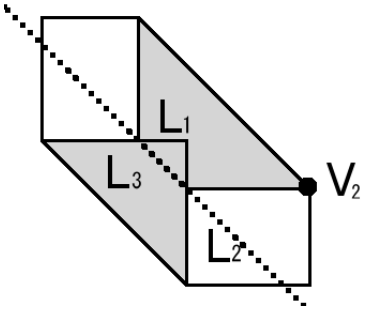

図3.2: 変形するループと辺を共有するループ

複数の平面を考えた場合、これに加えて「折り操作による、ループの移動の軌跡と交わるループ」も移動の影響を受けることになるが、本研究では折り操作途中に接触するループに関しては考慮しなかった。

3.2 回転軸の位置計算

折り操作の際の回転軸や頂点の移動方向は、ユーザーの入力した情報から即座に求めることができる。ユーザーが同一平面上の2点と折り角度を指定する方法

で折り操作を行う場合は直接回転軸の情報は得られないが、以下の計算により回転軸を求めることができる。回転軸を定義するためには、回転軸上の 1 点の位置と回転軸と平行なベクトルという 2 つの情報が必要になる。下の 2 式は、(3.1) 式で回転軸上の 1 点の位置 $\mathbf{P}_a$ を求め、(3.2) 式で回転軸と平行なベクトル $\mathbf{A}$ を求めている。ただし、(3.2) 式で使用している「·」は、ベクトルの内積を表す。式中の $\mathbf{P}_1$、$\mathbf{P}_2$ はそれぞれユーザーが 1 つ目に指定した点の位置と 2 つ目に指定した点の位置を表し、$\mathbf{P}_3$ は $\mathbf{P}_1$、$\mathbf{P}_2$ と同一平面上の $\mathbf{P}_1$、$\mathbf{P}_2$ とは異なる 1 点の位置を表している。$\mathbf{P}_3$ の情報は FaceGroup クラスの持つデータから容易に取得できる。

$$\mathbf{P}_a = \frac{\mathbf{P}_1 + \mathbf{P}_2}{2} \tag{3.1}$$

$$\mathbf{A} = \frac{(\mathbf{P}_a - \mathbf{P}_1) \cdot (\mathbf{P}_3 - \mathbf{P}_1)}{(\mathbf{P}_a - \mathbf{P}_1) \cdot (\mathbf{P}_a - \mathbf{P}_1)}(\mathbf{P}_a - \mathbf{P}_1) \tag{3.2}$$

3.3 折り線の作成

本実装のプログラムにおいて、折り線の作成位置 (ループの分割位置) は対象となるループの属する FaceGroup クラス上の回転軸の位置と重なる。変形を行うループに属する辺を順に探索し、辺が先に求めておいた回転軸と交点を持つ場合は、同一のループ上の他の交点と結ぶことでループを分割し、折り線を作成する。この際に折り線と辺の交点が辺の端点以外の位置だった場合、辺の分割も行う。

3.4 頂点の移動と変形するループの検索

変形するループに属し、回転軸よりも選択された頂点側に近い位置に存在する頂点は、折り操作によって位置が移動する。頂点の移動後の位置は、回転軸を中心に頂点の位置を回転することで求める。頂点の移動を行うことで新たに 3.1 の条件に当てはまるループが判明するので、条件に当てはまった場合は、変形操作を行うループとして追加する。頂点が移動する場合は、頂点が属する辺の位置も変化するので、頂点が属する辺を子として持つ全てのループは 3.1 で述べた条件に

当てはまり、変形するループだとわかる。ループごとに折り線と頂点の移動方向が決まれば、移動部分と重なり、重なり順で上方向になるループも判明するので、3.1 の条件に当てはまるループは変形するループだと分かる。

以降、1 つのループの探索が終わるごとに変形操作を行うループが残っていないか調べ、存在する場合は、折り線の作成と頂点の移動を繰り返す。また、今回の実装ではこの操作中に発見されたループが、3 次元空間上で全て同じ平面上に位置している場合のみ、折り操作を行う。

3.5 移動の確定と重なり順の更新

ループ上の頂点を移動しても新たに変形を行うループが発見されなかった場合は、矛盾のない位置に全ての頂点が移動したとして、折り紙モデルの変形を確定する。頂点の移動があったループは、属する FaceGroup と重なり順の更新を行う。ループの分割により新たに生成されたループの場合は属する FaceGroup を検索し新たな要素として追加し、既存のループの場合は、ループが変形後に位置する 3 次元平面を管理する FaceGroup クラスへ、自身が属する FaceGroup クラスを変更する。各ループはシミュレーション上で各 FaceGroup が固有に持つ番号によって、自身がどの FaceGroup に属しているかを管理しているため、ループが属する FaceGroup の変更は、自身の属する FaceGroup 番号を変更することで行う。2.2 で述べたように、本研究では各 FaceGroup 内で重なり順をリスト構造により管理している。重なり順の更新は、移動するループが前操作まで属していた FaceGroup の持つリストから、移動するループを削除し、新たに属する FaceGroup クラスの持つリストの、最初か最後の要素として移動するループを追加することで行う。新たに追加されるループの変形操作による移動方向を FaceGroup クラスの持つ法線ベクトルと比べて、重なり順の最初と最後、どちらに追加されるかを判断する。複数枚のループが同時に移動する場合は、移動前に属する FaceGroup での重なり順を元に追加する順序を決めることで重なり順に矛盾が生じないようにする。前方から複数枚のループを移動する場合は先頭のループから順に移動先の FaceGroup

に追加し、後方から複数枚のループを移動する場合は末尾のループから順に追加を行う。

同一平面の FaceGroup クラスが定義されていなかった場合は、新規に FaceGroup クラスを作成し、作成した FaceGroup クラスに属するループとしてそのループを追加する。この時点では FaceGroup クラスに属するループは追加したループ 1 つだけなので、このループを重なり順最上位とし、作成した FaceGroup クラスの法線ベクトルは、このループの法線ベクトルから取得して設定する。

第 4 章

結果と考察

本章では、第 1 章で掲げた本研究の目標が実装においてかなえられているかを検証した。目標としていたのは従来手法での課題であった「正確な履歴管理」「表面の正確な描画」「折り紙形状の整合性の維持」を同時に解決することと、複数枚の折り紙モデルを扱える折り紙シミュレーションの実装の 2 点である。本研究の実装は 3D グラフィックツールキットである FK System[15] を用いて行った。

4.1 結果

図 4.1: 180 度の折り操作を 1 度行った状態

図 4.1 は本研究の実装プログラムで、正方形の折り紙モデルに対し折り操作を 1 度行った状態である。折り角度 180 度の折り操作を行ったため、同一平面上に 2

つの面が属しているが、面の交差は生じておらず、表面が正確に描画できていることが確認できた。

図 4.2: 角度をつけた折り操作を行った状態

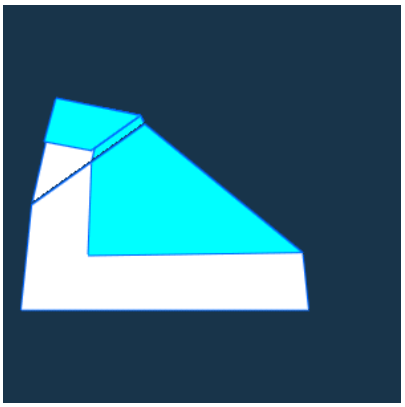

図 4.3: 角度がついた面に対して 180 度の折り操作を行った状態

図 4.2、4.3 は、図 4.1 の状態のモデルに対し、角度をつけた折り操作を行った画面と、その状態のモデルに対し更に角度 180 度の折り操作を行った画面である。１つの折り紙モデル上に複数の平面が存在する場合でも表面の描画が正確に行われることと、重なり順に基づき、移動面の検索が正確に行われていることが確認できた。また、折り操作を行った状態の折り紙モデルに対し履歴の巻き戻し操作を行うことで、モデルの状態を 1 ステップずつ元に戻すことができ、新たに折り操作を行うまでは巻き戻し処理の取り消しも行うことができた。

図 4.4: 面の交差が生じている例

図 4.4 は、図 4.2 の状態のモデルの、画面右下の頂点が画面左奥の角度がついたループより奥側に移動するように折り操作を行った画面である。ループが隣接していて、実際の紙ならば隙間がない部分をループが貫通してしまっている。

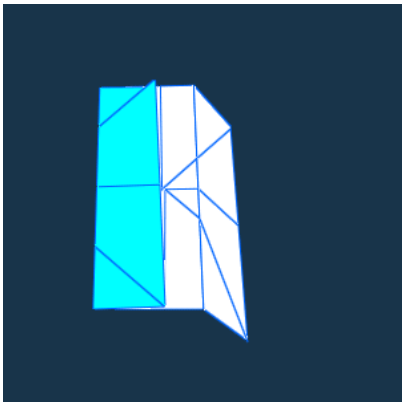

図 4.5: 画面奥側に向かって角度をつけた状態

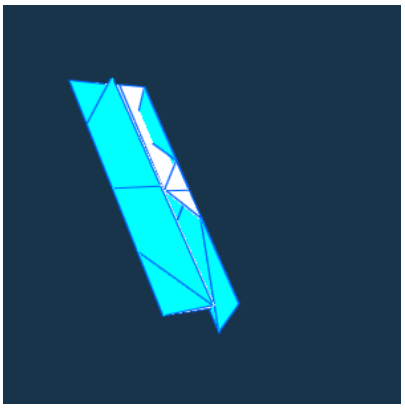

図 4.6: モデルを横側から見た状態

図 4.5 は何回か折り操作を繰り返して FaceGroup に属するループが多くなった状態のモデルに対して、画面右側の頂点を、カメラ位置から見て奥側に角度を付けて折り操作を行った画面で、図 4.6 は同じ状態の折り紙を違うカメラ位置から見た画面である。図 4.5 の位置からの描画には問題がないが、図 4.6 のようにカメラ位置を変えると、2 つの平面が交わる部分でループが他のループに食い込んで描画されている。これは第 2 章 2.2 節の操作によって重なり順に基づく描画を行う際に、各ループで隣接するループの位置状態を考慮していないために、別の FaceGroup に属するループと描画位置が交差してしまうために生じる現象である。

図 4.7 は 2 枚の折り紙モデルが重なった状態のモデルである。2 枚の折り紙モデルを同一平面上に配置しているが、重なり順に基づいて描画を行っているので、ちらつきなどの不具合が発生していないことが確認できる。図 4.8、4.9、4.10 はともに、図 4.7 のモデルに対し、折り操作を行った状態の画面である。2 枚の折り紙モデルにおいても、重なり順が考慮された折り操作が行われていることがわかる。図 4.11 はカメラから見て重なり順が上位の折り紙モデルに対して角度をつけた折り操作を行った後、画面奥側の折り紙モデルに対して手前側のモデルと交差する

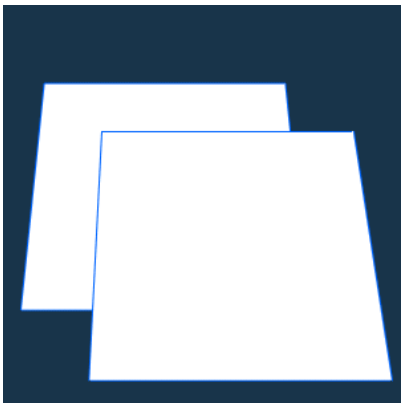

図 4.7: 2 枚の折り紙モデルを表示した状態

図 4.8: 上側の折り紙に対して折り操作を行った状態

図 4.9: 下側の折り紙に対して折り操作を行った状態

ような角度をつけた折り操作を行った状態の画面である。異なる平面同士での重なり順は判断されず、ループの交差が生じてしまっていることがわかる。

4.2 考察

本研究では、リアルタイムな折り紙シミュレーションにおいて、「正確な履歴管理」、「表面の正確な描画」、「折り紙形状の整合性の維持」という点を同時にかなえることと、複数枚の折り紙に対するシミュレーションが行える折り紙シミュレーションを作成することで、折り紙シミュレーションにおける再現の幅を広げることを目的としていた。

図 4.10: 2 枚のモデルが存在する画面で角度をつけた折り操作を行った状態

図 4.11: 2 枚の折り紙モデル間で、ループの交差が生じている状態

まずは従来手法の持つ課題の解決であるが、今回作成したプログラムでは、オイラー操作を用いて「正確な履歴管理」を実現することができた。また、Half-Edge 構造によりモデルの形状を管理し、角度を付けた折り操作を行った場合でもループ同士の隣接関係を正確に管理することができた。これにより、1 回の折り操作で変形するループが複数の平面に及ぶ場合は、折り操作を実行しないという処理を行うことで、折り操作によって、頂点間の距離に矛盾が生じることを防ぐことができた。しかし、ループ同士の衝突判定は行っていないため、ループ同士の交差が生じてしまうことがあり、「折り紙形状の整合性の維持」という課題は解決できていない。FaceGroup によりループを管理し、重なり順の情報を基に、描画するループの、カメラからの距離とサイズを変更する処理を行った結果、同一の FaceGroup に属するループに関して、重なり順に基づく描画を行うことができた。しかし、ループ同士の隣接関係などを考慮していないために、違う FaceGroup に属するループとの間でループの交差が発生することがあり、「表面の正確な描画」は行えていない。よって、先に挙げた従来の折り紙シミュレーションの持つ 3 点の課題を全て解決するという目標は達成できていない。

複数枚の折り紙モデルを扱える折り紙シミュレーションの実装としては、複数枚の折り紙モデルを描画し、同一平面上に属するループに対し重なり順を考慮した描画を行うことができた。しかし、異なる平面に属するループ同士での衝突判

定が行われないので、ループの交差が生じてしまう。この点は 1 枚の折り紙のシミュレーションにおいても発生する課題であり、解決には 1 枚の紙を扱う折り紙シミュレーションの課題の解決が重要であると考える。複数枚の折り紙に対するシミュレーションを行う折り紙シミュレーションは、不完全な状態ではあるが、実装することができた。

4.3 今後の展望

今回の実装では、折り操作を行った結果ループ同士が交差してしまい、モデルの形状が紙として矛盾のあるものになってしまうことがあった。この現象を解決するためには折り操作の際にループ同士で衝突判定を行う必要がある。また、今回は 1 回の折り操作で変形するループが、全て同一の平面上に存在する場合のみ折り操作を行うように設定したが、複数の平面上のループを 1 回の折り操作で変形することができるようになればより多くの状況に対応することができるだろう。重なり順に基づく正確な描画という課題の解決については、本研究で行ったように、モデルの描画位置を変更することで実現しようとした場合、他のループによって描画を妨げられないよう、他の FaceGroup に属するループの位置関係も考慮した上で描画位置を決定する必要がある。今回の実装では、重なり合った折り紙の間に折り紙を「差し込む」という操作は実装できていないが、折り紙作品の中には作成の際に差し込む操作を必須とするものが存在している。図 4.12 は、折り紙で作ったパーツと、パーツを複数組み合わせて作成した折り紙作品の例である。図 4.12 の作品も、折り紙を折って重ねる操作だけではなく、既に同一平面上に存在する紙と紙の間に「差し込む」という動作を行う必要があり、本研究の実装では作成することができない。このような作品を折ることを可能にするためには、重なり順の途中にループを挿入することが可能なプログラムの実装を行う必要がある。その際には、隣接しているループとの間に挿入し、折り紙形状としての矛盾が生じることにならないよう、ループの隣接関係に基づいて、ループを挿入できるかどうかを判断することになると考える。以上が今回本研究で作成したプログ

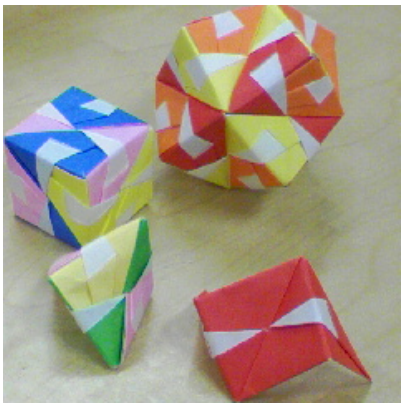

図 4.12: 複数枚の組み合わせによる折り紙作品とパーツ

ラムの今後の課題である。今後はこれらの点を解決し、より実用性の高い折り紙シミュレーションの実装を目指したい。

謝辞

本研究を進めていくにあたりご指導頂いた、渡辺大地講師のはじめとするゲームサイエンスプロジェクトのスタッフの皆様と、日々支えてくださっている研究室メンバーに心より御礼申し上げます。

参考文献

[1] 加藤渾一, ”折り紙と方程式”, http://www.nikonet.or.jp/spring/ori_h/ori_h.htm

[2] 森継修一, ”折り紙による３次元方程式の解法について”, 日本応用数理学会論文誌, Vol.16, No.1 pp. 79-92

[3] 日下貴之, 野島武敏, 勇田篤, ”折り紙モデルに基づく放物曲面シェルのインフレータブル構造化” 材料力学部門講演会公演論文集, Vol.2002 pp.547-548

[4] 齋藤淳, 野島武敏, ”折り畳み/展開可能な構造モデルの検討”, 年次大会講演論文集, Vol.2004, No.6 pp. 75-76

[5] 三谷純, ”折紙の展開図からの折りたたみ形状推定”, 情報処理学会研究報告, Vol.2006, No.18 pp. 1-6, 2006-CG-122-(1)

[6] 内田 忠, 伊藤 英則, ”折り紙過程の知識表現とその処理プログラムの作成”, 情報処理学会論文誌, Vol.32, No.12 pp. 1566-1573

[7] 舘知宏, ”Rigid Origami Simulation”, http://www.tsg.ne.jp/TT/software/index.html

[8] 横山卓弘, 高井昌彰, ”厚さを持った折り紙シミュレーションとその評価”, 情報処理学会研究報告, Vol. 2000, No. 115 pp. 19-24, 2000-CG-101-4

[9] 三谷純, 鈴木宏正, ”折り紙の構造把握のための形状構築と CG 表示”, 情報処理学会論文誌, Vol.46, No.1 pp. 247-254

[10] 宮崎慎也, 安田孝美, 横井茂樹, 鳥脇純一郎 , ”仮想空間における折り紙の対話型操作の実現”, 情報処理学会論文誌, vol.34 (9), pp.1994-2001

[11] 古田陽介, 三谷純, 福井幸男, ”マウスによる仮想折り紙の対話的操作のための計算モデルとインタフェース”, 情報処理学会シンポジウムシリーズ Vol.2007, No.4 pp.137-144, 2007

[12] OpenGL 策定委員会 著, 松田晃一 訳, ”OpenGL プログラミングガイド 原著第 5 版”, ピアソンエデュケーション, 2006

[13] 田中亜紀子, ”3 次元ＣＧによるユニット折り紙シミュレーション”, http://www.slis.tsukuba.ac.jp/ amy/semi/shoroku01/tanaka.html

[14] Weiler, K., ”Edge-Based Data Structures for Solid Modeling in Curved-Surface Environments” IEEE Computer Graphics and Applications, Vol.5, No.1, pp.43-52, January(1985).

[15] 渡辺大地, ”FK Toolkit System & TinyFK Toolkit System”, http://www.teu.ac.jp/aqua/ earth/FK/

[16] B. G. Baumgart, ”A Polyhedron Representation for Computer Vision”, AFIPS Conf. Proc., Vol. 44, 1975 NCC, pp. 589-596.

[17] M. Mantyla, R.Sulonen, ”A Solid Modeler with Euler Operaters”, IEEE Computer Graphics and Applications, Vol.2, No.7, pp.17-31, September(1982).