3ZB-04

リアルタイムでの局所的メッシュ簡略化とその応用について[1]

小田内敦史†
東京電機大学大学院理工学研究科†

築地立家‡
東京電機大学大学院理工学研究科‡

1. はじめに

昨今，一層膨大なデータ量を処理するコンピュータグラフィクス分野において，オリジナルメッシュの特徴を残しつつモデル情報を簡略化することは重要な課題である．これに対して，現在はテセレーションなどGPUを用いて，近似をとるといった手法などが取られている．

本研究は，精度の高いメッシュを簡略化し，かつ注目したい点の解像度を高い状態にし，それをリアルタイムレンダリングする一連の動作を一つのメッシュの内部で行うことを目的としている．指向性を持つ代表的なものとして本研究では視界を挙げている．

同一モデルにおいてオリジナルメッシュを完全に復元する領域と視界に影響の少ない点は解像度を低下させた領域を混在させるために，Quadric Error Metric法（以下QEM法）[1][2]を用いてモデルの局所的な解像度の変更を行った．また，その応用として，CPUに確保されている頂点を用いて衝突判定を行った．

リアルタイムメッシュ簡略化の類似研究には，レイトレーシングを用いたリアルタイムメッシュ簡略化[3]などが挙げられる．

2. 研究の目的

現在用いられるメッシュデータは作成する側の能力向上により非常に高精細になってきており，メッシュの効率的な描画はより重要な課題となってきている．

よって，研究の目的は，元のメッシュ本体を残しつつ，メッシュの簡略化手法をユーザー定義の範囲内に適用することで，簡略化メッシュとオリジナルメッシュを混在させることである．また，これによる描画をはじめとしたメッシュを用いる演算コストを軽減しつつ，限定的なオリジナルメッシュ使用を可能とすることである．また，描画機能以外の計算でも処理速度を確認するために衝突判定プログラムを作成した．

3. 使用機器・環境

以下に使用ハードウェアと使用ソフトウェアの構成を表1と表2として記す．

表1　使用ハードウェア構成

製品名	メーカー
15X7200-i7-RSM-DG7P	iiyama
intel Core i7-4720HQ	intel
SDY1600L-4G	IODATA
GeForce GTX 960M	NVIDIA

表2　使用ソフトウェア環境

製品名	メーカー	バージョン
Windows8.1 Embedded	Microsoft	6.3.9600
Microsoft VisualStudio2013	Microsoft	Version 12.0.31101.00 Update4
Microsoft DirectX SDK	Microsoft	June 2010

4. 研究の方法

本研究では，簡単なレンダリングプログラムを作成し，読み込んだメッシュとそのメッシュに解像度変更プログラムを使って作られた簡略化メッシュをレンダリングプログラムに通した結果，どの程度速度が向上したかを比較した．

作成した環境はDirectX9.0cを用いていたので，その標準的な画像出力関数であるDIRECT3DEVICE9::DrawPrimitiveUP()関数を用いて，線分や三角形，頂点などを描画した．この描画関数をどちらのメッシュの描画にも用いることによって，メッシュは同一の描画ステップを踏んだ．

4.1. メッシュ簡略化プログラム

本研究ではメッシュの簡略化にQEM法を用いた．この手法は面と頂点の関係性からその頂点の重要性を数値化し，線分の縮退をすることでポリゴンを線分へ，線分を頂点へと縮退させる技法である．復元は元のメッシュを限定的に呼び出すことで行った．

QEM法では，頂点がそれぞれエラー値とエラークアドリック行列という固有の値を持つ． エラー値の内容は(1)の式によって求められる．Δ(v)をエラー値とする．

$$\Delta(\mathrm{v}) = \mathrm{v}^{\mathrm{T}}\mathrm{Q}\mathrm{v} \quad \cdots(1)$$

エラー値の計算上におけるvは，線分を縮退したときにエラー値を最小化する頂点の座標である．

[1] Real-time local mesh simplification and its application
Atsushi Odauchi†Graduate School of Tokyo Denki University Graduate School of Science and Engineering†
Tatsuie Tsukiji‡Graduate School of Tokyo Denki University Graduate School of Science and Engineering‡

また，(1)の式におけるQがエラークアドリック行列であり，線分を縮退する際には縮退される線分の頂点各々が持つ二つの行列は足される．また，頂点が隣接するうちの一つのポリゴンが持つ平面式を(2)の式としたとき，

$$ax + by + cz + d = 0 \quad \cdots(2)$$

(2)の式の平面式を表す行列をpとすると，

$$p = [a, b, c, d]^T \quad \cdots(3)$$

$$K_p = pp^T \quad \cdots(4)$$

と表すことができ，その頂点が隣接するポリゴンの数をdとし，ポリゴンnの行列式をKp(n)とするとき，エラークアドリック行列Qは(4)の式から

$$Q = \sum_{n=1}^{d} K_p(n) \quad \cdots(5)$$

として求められる．

エラー値の用途は，メッシュをどの程度簡略化するかという閾値として使用できる．Δ(v)=0をメッシュの原型としたときに，ユーザーが設定した閾値が0に近ければ近いほど原型を残し，0から離れれば離れるほどメッシュの構成は簡素になる．

局所化を行うための閾値としては，任意の点Pにおける距離と，任意のベクトルvと各ポリゴンが持つ法線との角度である．任意のベクトルvはカメラの向くベクトルを想定したものである．この2つの要素を使用し，メッシュ復元の閾値とした．

4.2. 衝突判定プログラム

本研究は処理の分岐数などの原因からGPUを使用していないため，高解像度のオリジナルメッシュを部分的にCPU内で使用できることが利点である．よって，研究の応用として衝突判定プログラムを作成し，これを局所簡略化したメッシュに使用することを本研究の応用とした．

衝突判定プログラムは，メッシュのポリゴンを取得し，対象となる平面のポリゴンとの符号付距離を計算し，その結果で衝突判定を行うこととした．

5. 結果

採取するデータに関しては，衝突判定のみ，描画のみ，衝突判定と描画の両方，という状況を用意し，どの程度のフレーム速度が出るかどうかを基準とした．

メッシュの簡略化の描画結果としては図1のようになった．

メッシュの描画速度，衝突判定，描画と衝突判定の結果としては表3にまとめた．

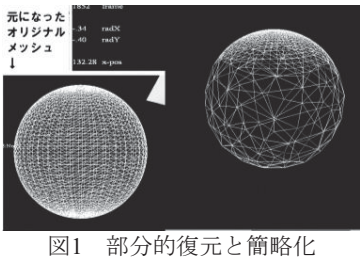

図1　部分的復元と簡略化

表3　メッシュの描画速度

状態	簡略化したメッシュ[fps]	簡略化していないメッシュ[fps]
描画のみ	57.92	52.90
衝突判定のみ	56.30	30.21
描画と衝突判定	49.63	14.63

このデータを取る際の簡略化において，オリジナルポリゴン数が4512枚のメッシュを簡略化し，簡略化後では平均ポリゴン数は445枚となった．

6. 結論と考察

第5節の結果から分かるように，描画という点に関しては，かなりメッシュを削減したにもかかわらず10%程度しか向上がみられない．また，衝突判定に関しては，2倍に近い速度の改善がみられる．そして，双方を同時に使用した場合では3倍程度の速度改善が認められる．これは描画機能においては，グラフィックスカードの性能に大部分の計算処理が依存していることが原因であると考えられる．

今回作成したメッシュの局所的簡略化プログラムには復元部分の探索計算コストが大きな問題となった．よって，復元部分を効率的に探索するために八分木検索などをポリゴン配列などに使い，座標などから分類分けをできるようにすることで，復元をさらに効率化することが課題として上げられる．

参考文献

[1]Micheal Garland and Paul S. Heckbert,”Surface Simplification Using Quadric Error Metrics”, Proc. of ACM SIGGRAPH ’97, pages 209-216, 1997.
[2]Hugues Hoppe,”Progressive Meshes”, Proc.in Siggraph’96, pages 99-108, 1996.
[3]望月直樹,牧野光則.”没入型ディスプレイでの人間の視野を利用した人間の視野を利用したレイトレーシング法の詳細度制御”.大学院研究年報理工学研究科篇第40号/2010, 中央大学大学院研究年報編集委員会, 2010.