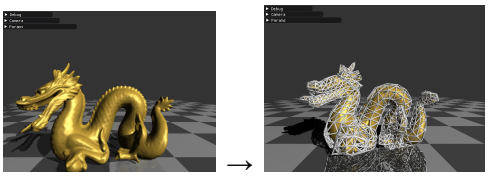
三角形から四面体へのコンバートに関して

2022年08月23日 中川展男



目的

この文章では、論文: A Constraint-based Formulation of Stable Neo-Hookean Materials (2021) の実装に関して必要となるデータ構造とコンバートフローについて解説することを目的とする。

配布物

- tri2tet

windows(x64現状併用する必要がある、fTetWild を併用する必要があります。テスト用に入力用の .obj ファイルも同梱しました。

操作方法

現状 fTetWild と tri2tet の2パス構成です

パス1: fTetWild

```
./FloatTetwild_bin -i surface.obj -o tet.mesh -l 0.5 --coarsen
```

三角形メッシュから、四面体メッシュを作るために fTetWild を呼んでいます。引数の詳細は公式の方を見ていただいたい方がよいですが、上記引数を簡単に説明すると、三角形メッシュ情報(*.obj)を入力して、四面体メッシュ(*.mesh) を生成していて、その際に四面体メッシュが可能な限り荒くなるような指定(-l 0.5 --coarsen)をしています。

四面体メッシュ(tet.mesh)はシミュレーションのみに用いるもので直接描画されるわけではないので必要最小な程度に荒いほうが好ましいです。いわゆるスキニングアニメーションのボーンに相当するような情報になります。

パス2: tri2tet

```
./tri2tet input.{mesh, obj} input.obj output.txt
```

コンソールから、引数3つを必ずこの順で指定して実行する必要があります。引数が足りない場合は usage 表示を行い、再入力を促します。

引数は順に

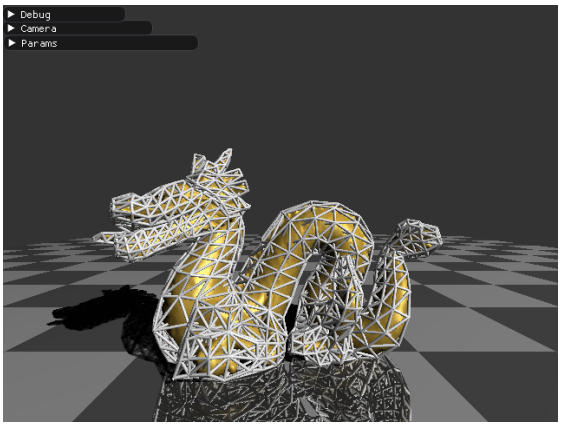
- 入力1: 四面体メッシュ (例: input.mesh)
  - INRIA Medit (*.mesh) 形式
  - Wavefront (*.obj) 形式
- 入力2: 三角形メッシュ (例: input.obj)
  - Wavefront (*.obj) 形式
- 出力: 独自 txt 形式出力 (例: output.txt)

となります。

パス1: fTetWild で生成された .mesh あるいは、Matthias Müller さんの web サイト上で公開されている Blender のアドオンで Blender 上からエクスポートされた四面体の.obj を入力として用いることを想定しています。ただし、.obj で指定する三角形メッシュは、パス1で出力された .obj と同一である必要はありません。しかしながら四面体メッシュの内部に、三角形メッシュを構成する全頂点が含まれる必要があります。

入力データ詳細

四面体メッシュと三角形メッシュは、下図のように三角形メッシュ(ゴールド)を四面体メッシュ(白ワイヤーフレーム)が包むエンベロープのような対応関係になっている必要があります。



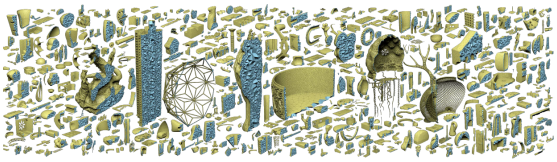
これは、コリジョンを含む物理シミュレーションは四面体メッシュを構成する頂点のみで行っているためです。描画用メッシュが四面体メッシュの外にはみ出てしまうとコリジョンの都合で見た目の不整合が発生してしまいます。例えば、上記例だと床に dragon のゴールド部分が食い込むのを避ける必要があります。

現状床以外のコリジョンは未実装ですし、四面体メッシュのセルフコリジョンをどうロバストかつ高速に実装するのかはまた悩ましい問題となります。

出力される txt ファイルには、下記の情報が含まれます。ウエイト情報を事前計算をすることでランタイム時の計算負荷を削減しています。

- シミュレーション用の四面体メッシュ情報
- 描画用三角形メッシュ情報
- 描画用三角形メッシュを内包するシミュレーションメッシュの四面体と捉えたときのウエイト情報

fTetWild とは何か



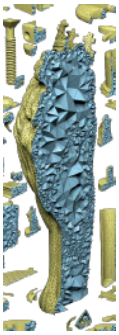
Fast Tetrahedral Meshing in the Wild (2020) で提案された四面体ボリュームメッシュ生成方法です。実装はオープンソースで公開されており、今回三角形メッシュから四面体メッシュのコンバートのために利用しています。

同分野の先行研究として制約付きドロネー分割を用いた TetGen や入力面を完全にリサンプルする CGAL などがありますが、fTetWild は、これらの課題を解決した最新手法で、既存手法よりあらゆる三角形メッシュからのコンバートをロバストに行うことが可能であると主張されています。

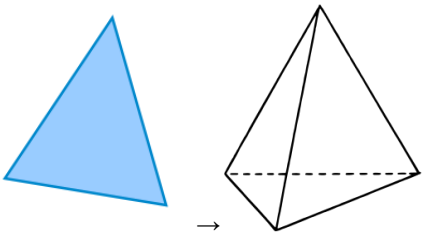
なぜ四面体メッシュが必要なのか

ビデオゲームで一般的に使われているメッシュは美しくかつ高速に描画する必要があります。そのため内部に頂点情報を持たず、表面だけの情報を持つ三角形(surface triangle mesh)で構成される事が多いかと思います。

しかしながら、このような三角形メッシュ情報だけだと今回のように正確な弾性変形を行う際に支障があります。例えば、体積を保存する変形を行う場合は下図のようにできるだけ均一に内部が分割されたメッシュが計算上好ましい事があります。



このような体積(volume)を考慮した形状として、三角形(下図左)の代わりに四面体(下図右)をメッシュ構成要素として用いることで最小の情報(4頂点)で内部解析を行うことが可能となります。



今後の予定

tri2tet から fTetWild を呼び出して、ユーザーからは四面体分割を直接意識させないほうが好ましいかもしれません。(もともとそのような用途(**tet**rahedron to **tri**angle)を考えて tet2tri と命名したので、現状コンバータの名前と内部実装が一致していない悩みがあります)

また、四面体生成用の三角形メッシュは荒くして、描画用の三角形メッシュを高精細にすることで、高速かつ高品質な描画ができる方が好ましいと考えています。

そのためにはコンバートフロー内部で解像度が異なる三角形メッシュを構築できる方が好ましいと考えています。