# アドバンスト CG 第6回レポート

学籍番号:201811411

所属:情報学群情報メディア創成学類

氏名:加藤虎之介

2021年6月6日

## 1 実行環境

## 1.1 実行に用いた OS

macOS Big Sur ver11.3.1

## 1.2 プログラム起動時に表示される情報

OpenGL version: 2.1 ATI-4.4.17

GLSL version: 1.20

 $\begin{tabular}{ll} Vendor: ATI Technologies Inc. \end{tabular}$ 

Renderer: AMD Radeon Pro 5300M OpenGL Engine

# 2 課題 A

## 2.1 修正したソースコード

#### 2.1.1 Stretching Constraint の実装

Code 1 pbd.cpp の projectStretchingConstraint 関数

```
void ElasticPBD::projectStretchingConstraint(float ks)

file (m_iNumEdge <= 1)

return;

for (int i = 0; i < m_iNumEdge; ++i)

for (int i = 0; i < m_iNumEdge; ++i)</pre>
```

```
// 四面体を使うときの内部エッジかどうかの判定&内部エッジを使うかどうかの
8
                          フラグチェック
                        if (m_vInEdge[i] && !m_bUseInEdge)
9
                               continue;
10
11
                        // エッジ情報の取得とエッジ両端の頂点番号および質量の取得 (固定点の質量は
12
                         大きくする)
                        const rxEdge &e = m_poly.edges[i];
                        int v1 = e.v[0];
14
                        int v2 = e.v[1];
15
                        float m1 = m_vFix[v1] ? 30.0f * m_vMass[v1] : m_vMass[v1];
16
                        float m2 = m_vFix[v2] ? 30.0f * m_vMass[v2] : m_vMass[v2];
17
                        if (m1 < glm::epsilon<float>() || m2 < glm::epsilon<float>())
18
                               continue;
19
20
                        // 2頂点の位置ベクトル
21
                        glm::vec3 p1 = m_vNewPos[v1];
22
                        glm::vec3 p2 = m_vNewPos[v2];
23
24
                        // 計算点間の元の長さ (制約条件)
25
26
                        float d = m_vLengths[i];
27
                        // TODO: 重力等を考慮した後の 2頂点座標 (スライド中のp<sup>1</sup>)はp1=m_vNewPos[v1
28
                          ],p2=m_vNewPos[v2]で得られるので,
                        // これらから制約を満たすような位置修正量dp1,dp2を求めて、m_vNewPos[v1]
29
                          ], m_vNewPos[v2]に足し合わせる.
                        // ◎エッジの長さによってはゼロ割が発生することがある.エラーチェックを忘れ
30
                         ずに!
                        glm::vec3 dp1, dp2;
31
32
                        // ----課題ここから----
33
                        // 重みの計算
34
                        float w1 = 1.0f / m1;
35
                        float w2 = 1.0f / m2;
36
37
                        // p1-p2ベクトルの長さ
38
                        float vecLength = glm::length(p1 - p2);
39
                        // エッジの長さが閾値よりも小さい場合はスキップする。(ゼロ割り対策)
40
                        if (vecLength < glm::epsilon<float>())
41
                               continue;
42
43
                        dp1 = (-w1 / (w1 + w2)) * (vecLength - d) * (p1 - p2) /
44
                         vecLength;
                        dp2 = (w2 / (w1 + w2)) * (vecLength - d) * (p1 - p2) /
45
                         vecLength;
```

46

#### 2.1.2 Bending Constraint の実装

Code 2 pbd.cpp の projectBendingConstraint 関数

```
void ElasticPBD::projectBendingConstraint(float ks)
          {
2
                  if (m_iNumTris <= 1 || m_iNumEdge <= 0 || m_vBends.empty())</pre>
                          return;
                  for (int i = 0; i < m_iNumEdge; i++)</pre>
6
                          // 2つのポリゴンに挟まれたエッジ情報の取得
                          const rxEdge &e = m_poly.edges[i];
                          if (e.f.size() < 2)</pre>
10
                                  continue; // このエッジを含むポリゴン数が 1なら処理をスキップ
11
12
                          // 2つの三角形を構成する 4項点のインデックスを抽出
13
                          set<int>::iterator itr = e.f.begin();
14
                          const rxFace &f1 = m_poly.faces[*itr];
15
                          itr++;
16
                          const rxFace &f2 = m_poly.faces[*itr];
17
                          int v1 = e.v[0], v2 = e.v[1], v3, v4;
18
                          for (int j = 0; j < 3; ++j)
19
20
                                  if (f2[j] != v1 && f2[j] != v2)
^{21}
                                          v4 = f2[j];
22
                                  if (f1[j] != v1 && f1[j] != v2)
23
                                          v3 = f1[j];
24
25
                          float m1 = m_vFix[v1] ? 30.0f * m_vMass[v1] : m_vMass[v1];
26
                          float m2 = m_vFix[v2] ? 30.0f * m_vMass[v2] : m_vMass[v2];
27
                          float m3 = m_vFix[v3] ? 30.0f * m_vMass[v3] : m_vMass[v3];
28
                          float m4 = m_vFix[v4] ? 30.0f * m_vMass[v4] : m_vMass[v4];
29
                          if (m1 < glm::epsilon<float>() || m2 < glm::epsilon<float>() ||
30
                             m3 < glm::epsilon<float>() || m4 < glm::epsilon<float>())
                                  continue;
31
```

```
32
                       // 4頂点の位置ベクトル (p2-p4 は p1 に対する相対位置ベクトル) -> スライド
33
                         p36の^(ハット)付きのp2-p4の方
                       glm::vec3 p1 = m_vNewPos[v1];
34
                       glm::vec3 p2 = m_vNewPos[v2] - p1;
35
                       glm::vec3 p3 = m_vNewPos[v3] - p1;
36
                       glm::vec3 p4 = m_vNewPos[v4] - p1;
37
                       // 2面間の初期角度
                       float phi0 = m_vBends[i];
40
41
                       // TODO:エッジを挟んだ 4頂点座標p1,p2,p3,p4 から bending
42
                         constraint を満たす位置修正量 dp1~dp4 を求め,
                       // m_vNewPos[v1]~m_vNewPos[v4]に足し合わせる.
43
                       // ・\uparrowで定義しているp1\sim p4で、p2\sim p4は p1 に対する相対座標 (スライド
44
                         p36 の ^(ハット)付きのp2-p4 の方)にしてあるので注意
                       // ・三角形ポリゴン間の角度の初期値φ Oはm_vBends[i]で得られる (↑で変数
45
                         phi0 に代入済み)
                       // ◎ベクトルの大きさで割るという式が多いが,メッシュの変形によってはゼロ割
                         が発生することがある.エラーチェックを忘れずに!
                       // \bigcircスライドp36の dを計算するときに、-1\sim1の範囲にあるかをちゃんとチェ
47
                         ックして,範囲外ならクランプするように!
                       // - 授業スライドに合わせるためにp1 \sim p4 など配列を使わずに書いている.
48
                       // 配列を使って書き換えても構わないが添え字の違い (配列は ∂から始まる)に
49
                         注意.
                       glm::vec3 dp1(0.0f), dp2(0.0f), dp3(0.0f), dp4(0.0f);
50
51
                       // ----課題ここから----
52
                       // n1, n2の正規化前のベクトルを_n1, _n2とする。
53
                       auto _n1 = glm::cross(p2, p3);
                       auto _n2 = glm::cross(p2, p4);
55
                       // _n1, _n2の長さ
                       float _n1Length = glm::length(_n1);
57
                       float _n2Length = glm::length(_n2);
58
                       //_n1, _n2 の大きさによってゼロ割が発生しないか確認。
59
                       if (_n1Length < glm::epsilon<float>() || _n2Length < glm::</pre>
60
                         epsilon<float>())
                              continue;
61
62
                       // n1, n2を計算
                       auto n1 = glm::normalize(_n1);
64
                       auto n2 = glm::normalize(_n2);
65
66
                       // dを計算
67
                       float d = min(max(glm::dot(n1, n2), -1.0f), 1.0f);
68
69
```

```
// q1~q4
70
                           auto q3 = (glm::cross(p2, n2) + glm::cross(n1, p2) * d) /
71
                           auto q4 = (glm::cross(p2, n1) + glm::cross(n2, p2) * d) /
72
                             _n2Length;
                           auto q2 = -(glm::cross(p3, n2) + glm::cross(n1, p3) * d) /
73
                             _n1Length - (glm::cross(p4, n1) + glm::cross(n2, p4) * d) /
                             _n2Length;
                           auto q1 = -q2 - q3 - q4;
74
75
                           // 重み
76
                           float w1 = 1.0f / m1;
77
                           float w2 = 1.0f / m2;
78
                           float w3 = 1.0f / m3;
79
                           float w4 = 1.0f / m4;
80
81
                           // dpi の計算に必要な定数部分の事前計算
82
                           float qLengthSqrSum = pow(glm::length(q1), 2.0) + pow(glm::
83
                             length(q2), 2.0) + pow(glm::length(q3), 2.0) + pow(glm::
                             length(q4), 2.0);
                           // ゼロ割予防
84
                           if (qLengthSqrSum < glm::epsilon<float>() || (w1 + w2 + w3 + w4
85
                             ) < glm::epsilon<float>())
                                   continue;
86
                           auto CP = -4.0f / (w1 + w2 + w3 + w4) * sqrt(1.0f - d * d) *
87
                             (acos(d) - phi0) / qLengthSqrSum;
88
                           // dp1~dp4
89
                           dp1 = w1 * CP * q1;
90
                           dp2 = w2 * CP * q2;
91
92
                           dp3 = w3 * CP * q3;
                           dp4 = w4 * CP * q4;
93
94
                           // ----課題ここまで----
95
96
                           // 頂点位置を移動
97
                           if (!m_vFix[v1])
98
                                   m_vNewPos[v1] += ks * dp1;
99
                           if (!m_vFix[v2])
100
101
                                   m_vNewPos[v2] += ks * dp2;
                           if (!m_vFix[v3])
102
                                   m_vNewPos[v3] += ks * dp3;
103
                           if (!m_vFix[v4])
104
                                   m_vNewPos[v4] += ks * dp4;
105
                   }
106
           }
107
```

#### 2.1.3 Volume Constraint の実装

Code 3 pbd.cpp の projectVolumeConstraint 関数

```
void ElasticPBD::projectVolumeConstraint(float ks)
         {
                if (m_iNumTets <= 1)</pre>
                        return;
5
                for (int i = 0; i < m_iNumTets; i++)</pre>
                 {
                        // 四面体情報 (四面体を構成する 4項点インデックス)の取得
                        int v1 = m_vTets[i][0], v2 = m_vTets[i][1], v3 = m_vTets[i]
9
                         [2], v4 = m_vTets[i][3];
10
                        // 四面体の 4頂点座標と質量の取り出し
11
                        glm::vec3 p1 = m_vNewPos[v1];
                        glm::vec3 p2 = m_vNewPos[v2];
13
                        glm::vec3 p3 = m_vNewPos[v3];
14
                        glm::vec3 p4 = m_vNewPos[v4];
15
                        float m1 = m_vFix[v1] ? 30.0f * m_vMass[v1] : m_vMass[v1];
16
                        float m2 = m_vFix[v2] ? 30.0f * m_vMass[v2] : m_vMass[v2];
17
                        float m3 = m_vFix[v3] ? 30.0f * m_vMass[v3] : m_vMass[v3];
18
                        float m4 = m_vFix[v4] ? 30.0f * m_vMass[v4] : m_vMass[v4];
19
                        if (m1 < glm::epsilon<float>() || m2 < glm::epsilon<float>() ||
20
                          m3 < glm::epsilon<float>() || m4 < glm::epsilon<float>())
                               continue;
21
22
                        // 四面体の元の体積
23
                        float V0 = m_vVolumes[i];
24
25
                        // TODO:四面体の 4頂点座標p1,p2,p3,p4 から volume
26
                          constraint を満たす位置修正量 dp1~dp4 を求め,
                        // m_vNewPos[v1]~m_vNewPos[v4]に足し合わせる.
27
                        // ◎ベクトルの大きさで割るという式が多いが,メッシュの変形によってはゼロ割
28
                         が発生することがある.エラーチェックを忘れずに!
                        // - 四面体の体積はスライドに書いてある式を書くのでもよいし,
29
                        // cal Volume()という四面体の体積計算用関数も用意してあるのでこれを使っ
                          ても良い
                        // - 授業スライドに合わせるためにp1 \sim p4 など配列を使わずに書いている.
31
                        // 配列を使って書き換えても構わないが添え字の違い (配列は Oから始まる)に
32
                          注意.
                        glm::vec3 dp1(0.0f), dp2(0.0f), dp3(0.0f), dp4(0.0f);
33
34
                        // ----課題ここから----
35
```

```
// 重み
36
                           float w1 = 1.0f / m1;
37
                           float w2 = 1.0f / m2;
38
                           float w3 = 1.0f / m3;
39
                           float w4 = 1.0f / m4;
40
41
                           // q1~q4
42
                           auto q1 = glm::cross(p2 - p3, p4 - p3);
43
                           auto q2 = glm::cross(p3 - p1, p4 - p1);
44
                           auto q3 = glm::cross(p1 - p2, p4 - p2);
45
                           auto q4 = glm::cross(p2 - p1, p3 - p1);
46
47
                           // dpi の計算式の定数部分
48
                           float wsum = w1 + w2 + w3 + w4;
49
                           float qLengthSqrSum = pow(glm::length(q1), 2.0) + pow(glm::
50
                             length(q2), 2.0) + pow(glm::length(q3), 2.0) + pow(glm::
                             length(q4), 2.0);
51
                           if (wsum < glm::epsilon<float>() || qLengthSqrSum < glm::</pre>
52
                             epsilon<float>())
53
                                   continue;
54
                           auto CP = -(calVolume(p1, p2, p3, p4) - V0) / (wsum *)
55
                             qLengthSqrSum);
56
                           // dp1~dp4
57
                           dp1 = w1 * CP * q1;
58
                           dp2 = w2 * CP * q2;
59
                           dp3 = w3 * CP * q3;
60
                           dp4 = w4 * CP * q4;
61
62
                           // ----課題ここまで----
63
64
                           // 頂点位置を移動
65
                           if (!m_vFix[v1])
66
                                   m_vNewPos[v1] += ks * dp1;
67
                           if (!m_vFix[v2])
68
                                   m_vNewPos[v2] += ks * dp2;
69
                           if (!m_vFix[v3])
70
71
                                   m_vNewPos[v3] += ks * dp3;
                           if (!m_vFix[v4])
72
                                   m_vNewPos[v4] += ks * dp4;
73
                   }
74
           }
75
```

# 2.2 実行結果

格子状メッシュかつ、alpha=1.00の条件でプログラムを実行した結果は以下のようになった。

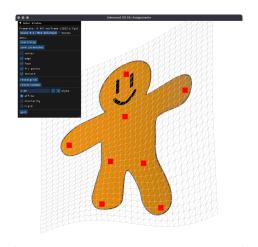


図1 Affine Deformation の実行結果

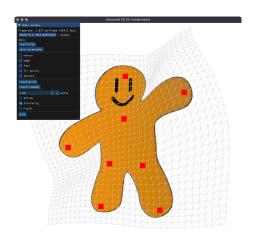


図 2 Similarity Deformation の実行結果

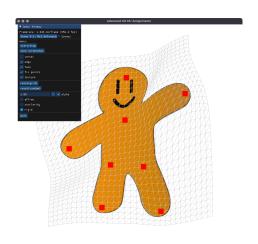


図3 Rigid Deformation の実行結果