

# 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  
Vendor: ATI Technologies Inc.  
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

```
1 void ElasticPBD::projectStretchingConstraint(float ks)
2 {
3     if (m_iNumEdge <= 1)
4         return;
5
6     for (int i = 0; i < m_iNumEdge; ++i)
7     {
```

```

8      // 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
      // フラグチェック
9      if (m_vInEdge[i] && !m_bUseInEdge)
10         continue;
11
12     // 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
      // 大きくする）
13     const rxEdge &e = m_poly.edges[i];
14     int v1 = e.v[0];
15     int v2 = e.v[1];
16     float m1 = m_vFix[v1] ? 30.0f * m_vMass[v1] : m_vMass[v1];
17     float m2 = m_vFix[v2] ? 30.0f * m_vMass[v2] : m_vMass[v2];
18     if (m1 < glm::epsilon<float>() || m2 < glm::epsilon<float>())
19         continue;
20
21     // 2頂点の位置ベクトル
22     glm::vec3 p1 = m_vNewPos[v1];
23     glm::vec3 p2 = m_vNewPos[v2];
24
25     // 計算点間の元の長さ（制約条件）
26     float d = m_vLengths[i];
27
28     // TODO:重力等を考慮した後の 2頂点座標（スライド中のp'）はp1=m_vNewPos[v1
      // ],p2=m_vNewPos[v2]で得られるので、
29     // 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
      // ],m_vNewPos[v2]に足し合わせる。
30     // ◎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
      // ずに！
31     glm::vec3 dp1, dp2;
32
33     // ----課題ここから----
34     // 重みの計算
35     float w1 = 1.0f / m1;
36     float w2 = 1.0f / m2;
37
38     // p1-p2ベクトルの長さ
39     float vecLength = glm::length(p1 - p2);
40     // 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）
41     if (vecLength < glm::epsilon<float>())
42         continue;
43
44     dp1 = (-w1 / (w1 + w2)) * (vecLength - d) * (p1 - p2) /
      vecLength;
45     dp2 = (w2 / (w1 + w2)) * (vecLength - d) * (p1 - p2) /
      vecLength;
46

```

```

47         // ----課題ここまで----
48
49         // 頂点位置を修正
50         if (!m_vFix[v1])
51             m_vNewPos[v1] += ks * dp1;
52         if (!m_vFix[v2])
53             m_vNewPos[v2] += ks * dp2;
54     }
55 }

```

---

## 2.1.2 Bending Constraint の実装

Code 2 pbd.cpp の projectBendingConstraint 関数

```

1  void ElasticPBD::projectBendingConstraint(float ks)
2  {
3      if (m_iNumTris <= 1 || m_iNumEdge <= 0 || m_vBends.empty())
4          return;
5
6      for (int i = 0; i < m_iNumEdge; i++)
7      {
8          // 2つのポリゴンに挟まれたエッジ情報の取得
9          const rxEdge &e = m_poly.edges[i];
10         if (e.f.size() < 2)
11             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
12
13         // 2つの三角形を構成する4頂点のインデックスを抽出
14         set<int>::iterator itr = e.f.begin();
15         const rxFace &f1 = m_poly.faces[*itr];
16         itr++;
17         const rxFace &f2 = m_poly.faces[*itr];
18         int v1 = e.v[0], v2 = e.v[1], v3, v4;
19         for (int j = 0; j < 3; ++j)
20         {
21             if (f2[j] != v1 && f2[j] != v2)
22                 v4 = f2[j];
23             if (f1[j] != v1 && f1[j] != v2)
24                 v3 = f1[j];
25         }
26         float m1 = m_vFix[v1] ? 30.0f * m_vMass[v1] : m_vMass[v1];
27         float m2 = m_vFix[v2] ? 30.0f * m_vMass[v2] : m_vMass[v2];
28         float m3 = m_vFix[v3] ? 30.0f * m_vMass[v3] : m_vMass[v3];
29         float m4 = m_vFix[v4] ? 30.0f * m_vMass[v4] : m_vMass[v4];
30         if (m1 < glm::epsilon<float>() || m2 < glm::epsilon<float>() ||
31             m3 < glm::epsilon<float>() || m4 < glm::epsilon<float>())
32             continue;

```

```

32
33 // 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
    p36 の ^ (ハット) 付きの p2-p4 の方
34 glm::vec3 p1 = m_vNewPos[v1];
35 glm::vec3 p2 = m_vNewPos[v2] - p1;
36 glm::vec3 p3 = m_vNewPos[v3] - p1;
37 glm::vec3 p4 = m_vNewPos[v4] - p1;
38
39 // 2面間の初期角度
40 float phi0 = m_vBends[i];
41
42 // TODO: エッジを挟んだ 4頂点座標 p1, p2, p3, p4 から bending
    constraint を満たす位置修正量 dp1~dp4 を求め,
43 // m_vNewPos[v1]~m_vNewPos[v4] に足し合わせる.
44 // ・↑で定義している p1~p4 で, p2~p4 は p1 に対する相対座標 (スライド
    p36 の ^ (ハット) 付きの p2-p4 の方) にしてあるので注意
45 // ・三角形ポリゴン間の角度の初期値  $\phi_0$  は m_vBends[i] で得られる (↑で変数
    phi0 に代入済み)
46 // ◎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!
47 // ◎スライド p36 の d を計算するときに, -1~1 の範囲にあるかをちゃんとチェ
    ックして, 範囲外ならクランプするように!
48 // - 授業スライドに合わせるために p1~p4 など配列を使わずに書いている.
49 // 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 0 から始まる) に
    注意.
50 glm::vec3 dp1(0.0f), dp2(0.0f), dp3(0.0f), dp4(0.0f);
51
52 // ----課題ここから----
53 // n1, n2 の正規化前のベクトルを _n1, _n2 とする。
54 auto _n1 = glm::cross(p2, p3);
55 auto _n2 = glm::cross(p2, p4);
56 // _n1, _n2 の長さ
57 float _n1Length = glm::length(_n1);
58 float _n2Length = glm::length(_n2);
59 // _n1, _n2 の大きさによってゼロ割が発生しないか確認。
60 if (_n1Length < glm::epsilon<float>() || _n2Length < glm::
    epsilon<float>())
61     continue;
62
63 // n1, n2 を計算
64 auto n1 = glm::normalize(_n1);
65 auto n2 = glm::normalize(_n2);
66
67 // d を計算
68 float d = min(max(glm::dot(n1, n2), -1.0f), 1.0f);
69

```

```

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

```

### 2.1.3 Volume Constraint の実装

Code 3 pbd.cpp の projectVolumeConstraint 関数

```
1 void ElasticPBD::projectVolumeConstraint(float ks)
2 {
3     if (m_iNumTets <= 1)
4         return;
5
6     for (int i = 0; i < m_iNumTets; i++)
7     {
8         // 四面体情報 (四面体を構成する 4 頂点インデックス) の取得
9         int v1 = m_vTets[i][0], v2 = m_vTets[i][1], v3 = m_vTets[i]
10            [2], v4 = m_vTets[i][3];
11
12         // 四面体の 4 頂点座標と質量の取り出し
13         glm::vec3 p1 = m_vNewPos[v1];
14         glm::vec3 p2 = m_vNewPos[v2];
15         glm::vec3 p3 = m_vNewPos[v3];
16         glm::vec3 p4 = m_vNewPos[v4];
17         float m1 = m_vFix[v1] ? 30.0f * m_vMass[v1] : m_vMass[v1];
18         float m2 = m_vFix[v2] ? 30.0f * m_vMass[v2] : m_vMass[v2];
19         float m3 = m_vFix[v3] ? 30.0f * m_vMass[v3] : m_vMass[v3];
20         float m4 = m_vFix[v4] ? 30.0f * m_vMass[v4] : m_vMass[v4];
21         if (m1 < glm::epsilon<float>() || m2 < glm::epsilon<float>() ||
22             m3 < glm::epsilon<float>() || m4 < glm::epsilon<float>())
23             continue;
24
25         // 四面体の元の体積
26         float V0 = m_vVolumes[i];
27
28         // TODO: 四面体の 4 頂点座標 p1, p2, p3, p4 から volume
29         // constraint を満たす位置修正量 dp1~dp4 を求め、
30         // m_vNewPos[v1]~m_vNewPos[v4] に足し合わせる。
31         // ◎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
32         // が発生することがある。エラーチェックを忘れずに！
33         // - 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、
34         //   calVolume() 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
35         //   ても良い
36         // - 授業スライドに合わせるために p1~p4 など配列を使わずに書いている。
37         //   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 0 から始まる) に
38         //   注意。
39         glm::vec3 dp1(0.0f), dp2(0.0f), dp3(0.0f), dp4(0.0f);
40
41         // ----課題ここから----
```

```

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



