アドバンスト CG
第 5 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 5 月 27 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: 4.1 ATI-4.4.17
GLSL version: 4.10
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　Affine Deformations の実装

Code 1　affineDeformation

```
1 glm::vec2 rxMeshDeform2D::affineDeformation(const glm::vec2 &v, const glm::vec2 &pc,
    const glm::vec2 &qc, const double alpha)
2 {
3         // TODO:この部分で 入力頂点座標v と 制御点座標p,q から入力頂点の変形後の座標を計算するコードを書く
4         // - 横ベクト×行列の計算部分は
            ルglm のオペレータ*は使わない方がよい（glm は縦ベクトルを前提としている）
```

```
5       // -
          Aj を計算してから変形後の座標を計算するのでも，スライド p43 の式を直接計算するのでもどちらでも OK

6       // (Aj の前計算は今回は行わないでもよい)
7       //
8       // - 制御点でループして相対座標を計算するまでのコード例:
9
10      // Aj の逆行列部分の計算
11      glm::mat2 WP = glm::mat2(0.f);
12      for (size_t k = 0; k < m_iNfix; k++) // 制御点数 (m_iNfix)でループを回す
13      {
14              int i = m_vFix[k]; // 制御点の頂点インデックス
15
16              // 重み
17              const glm::vec2 &p = m_vP[i];
18              // 固定点と計算点の間の距離に基づく重み
19              double dist = glm::length2(p - v);
20              float w = (dist > 1.0e-6) ? 1.0f / pow(dist, 2.0f * alpha) : 0.0f;
21              // 重心を中心とした制御点の相対座標
22              const glm::vec2 p_hat = p - pc;
23
24              WP[0][0] += p_hat.x * w * p_hat.x;
25              WP[0][1] += p_hat.y * w * p_hat.x;
26              WP[1][0] += p_hat.x * w * p_hat.y;
27              WP[1][1] += p_hat.y * w * p_hat.y;
28      }
29
30      // 行列式
31      double det = glm::determinant(WP);
32      // 正則性のチェック
33      if (det < 1.0e-6)
34      {
35              return v;
36      }
37      // 逆行列の計算
38      glm::mat2 invWP = glm::inverse(WP);
39
40      // fa(v)を計算
41      glm::vec2 fa = glm::vec2(0.f);
42      for (int k = 0; k < m_iNfix; ++k)
43      {           // 制御点数 (m_iNfix)でループを回す
44              int j = m_vFix[k]; // 制御点の頂点インデックス
45
46              // 重心を中心とした制御点の相対座標
47              // 各頂点の座標は配列m_vP と m_vX に格納されている(それぞれ初期形状と変形後の形状)
48              const glm::vec2 p_hat = m_vP[j] - pc; // 初期形状での座標
```

```
49                  const glm::vec2 q_hat = m_vX[j] - qc; // 変形後の座標
50
51                  // 固定点と計算点の間の距離に基づく重み
52                  const glm::vec2 p = m_vP[j];
53                  double dist = glm::length2(p - v);
54                  float w = (dist > 1.0e-6) ? 1.0f / pow(dist, 2.0f * alpha) : 0.0f;
55
56                  // Aj を計算する
57                  glm::vec2 B = v - pc;
58                  float A = w * ((B.x * invWP[0].x + B.y * invWP[0].y) * p_hat.x + (B.x
                        * invWP[1].x + B.y * invWP[1].y) * p_hat.y);
59
60                  fa += A * q_hat;
61          }
62
63          // 変形後の頂点v の座標
64          fa += qc;
65
66          return fa;
67  }
```

2.1.2 Similarity Deformations の実装

Code 2 similarityDeformation

```
1  glm::vec2 rxMeshDeform2D::similarityDeformation(const glm::vec2 &v, const glm::vec2 &pc
     , const glm::vec2 &qc, const double alpha)
2  {
3          // TODO:この部分で 入力頂点座標v と 制御点座標p,q から入力頂点の変形後の座標を計算するコー
             ドを書く
4          // - 横ベクト×行列の計算部分は
             ルglm のオペレータ*は使わない方がよい（glm は縦ベクトルを前提としている）
5          // - μs を先に計算してから変形後の頂点位置を計算
6          // - 行列A_i の前計算は今回は行わないでもよい
7
8          // μs を計算
9          float ms = 0.0f;
10         for (size_t k = 0; k < m_iNfix; k++)
11         {
12                 int i = m_vFix[k];
13                 const glm::vec2 p = m_vP[i];
14                 const glm::vec2 p_hat = p - pc;
15
16                 // 重みw
17                 double dist = glm::length2(p - v);
18                 float w = (dist > 1.0e-6) ? 1.0f / pow(dist, 2.0f * alpha) : 0.0f;
19
```

```
20                  // μsに加算
21                  ms += w * (p_hat.x * p_hat.x + p_hat.y * p_hat.y);
22          }
23
24          // fs(v)を計算
25          glm::vec2 fsv = glm::vec2(0.0f);
26
27          glm::vec2 vecVPc = v - pc;
28          glm::mat2 matVPc;
29          matVPc[0][0] = vecVPc.x;
30          matVPc[0][1] = vecVPc.y;
31          matVPc[1][0] = vecVPc.y;
32          matVPc[1][1] = -vecVPc.x;
33          glm::mat2 trnsMatVPc = glm::transpose(matVPc);
34
35          for (int k = 0; k < m_iNfix; k++)
36          {
37                  int i = m_vFix[k];
38                  const glm::vec2 p = m_vP[i];
39                  const glm::vec2 p_hat = p - pc;
40                  const glm::vec2 q_hat = m_vX[i] - qc;
41
42                  // 重みw
43                  double dist = glm::length2(p - v);
44                  float w = (dist > 1.0e-6) ? 1.0f / pow(dist, 2.0f * alpha) : 0.0f;
45
46                  // 変位行列A
47                  glm::mat2 matP;
48                  matP[0][0] = p_hat.x;
49                  matP[0][1] = p_hat.y;
50                  matP[1][0] = p_hat.y;
51                  matP[1][1] = -p_hat.x;
52
53                  glm::mat2 A = w * matP * trnsMatVPc;
54
55                  // 加算
56                  fsv += glm::transpose((1.0f / ms) * A) * q_hat;
57          }
58          // 変形後の頂点vの座標
59          fsv += qc;
60
61          return fsv;
62  }
```

2.1.3 Rigid Deformations の実装

Code 3　rigidDeformation

```
glm::vec2 rxMeshDeform2D::rigidDeformation(const glm::vec2 &v, const glm::vec2 &pc,
  const glm::vec2 &qc, const double alpha)
{
        // TODO:この部分で 入力頂点座標v と 制御点座標p,q から入力頂点の変形後の座標を計算するコードを書く
        // - 横ベクト×行列の計算部分は
          ルglm のオペレータ*は使わない方がよい（glm は縦ベクトルを前提としている）
        // - μf を先に計算してから変形後の頂点位置を計算
        // - 行列A_i の前計算は今回は行わないでもよい

        // μf を計算
        // mf = sqrt(A^2 + B^2)とする
        float A = 0.0f;
        float B = 0.0f;
        for (size_t k = 0; k < m_iNfix; k++)
        {
                int i = m_vFix[k];
                const glm::vec2 p = m_vP[i];
                const glm::vec2 p_hat = p - pc;
                const glm::vec2 q_hat = m_vX[i] - qc;

                // 重みw
                double dist = glm::length2(p - v);
                float w = (dist > 1.0e-6) ? 1.0f / pow(dist, 2.0f * alpha) : 0.0f;

                A += w * (q_hat.x * p_hat.x + q_hat.y * p_hat.y);
                B += w * (q_hat.x * -p_hat.y + q_hat.y * p_hat.x);
        }
        const float mf = glm::sqrt(A * A + B * B);

        // fr(v)を計算
        glm::vec2 frv = glm::vec2(0.0f);

        glm::vec2 vecVPc = v - pc;
        glm::mat2 matVPc;
        matVPc[0][0] = vecVPc.x;
        matVPc[0][1] = vecVPc.y;
        matVPc[1][0] = vecVPc.y;
        matVPc[1][1] = -vecVPc.x;
        glm::mat2 trnsMatVPc = glm::transpose(matVPc);

        for (int k = 0; k < m_iNfix; k++)
        {
                int i = m_vFix[k];
                const glm::vec2 p = m_vP[i];
```

```
43                  const glm::vec2 p_hat = p - pc;
44                  const glm::vec2 q_hat = m_vX[i] - qc;
45
46                  // 重みw
47                  double dist = glm::length2(p - v);
48                  float w = (dist > 1.0e-6) ? 1.0f / pow(dist, 2.0f * alpha) : 0.0f;
49
50                  // 変位行列A
51                  glm::mat2 matP;
52                  matP[0][0] = p_hat.x;
53                  matP[0][1] = p_hat.y;
54                  matP[1][0] = p_hat.y;
55                  matP[1][1] = -p_hat.x;
56
57                  glm::mat2 A = w * matP * trnsMatVPc;
58
59                  // 加算
60                  frv += glm::transpose((1.0f / mf) * A) * q_hat;
61          }
62
63          // 変形後の頂点v の座標
64          frv += qc; // ここも書き換えること
65
66          return frv;
67  }
```

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。

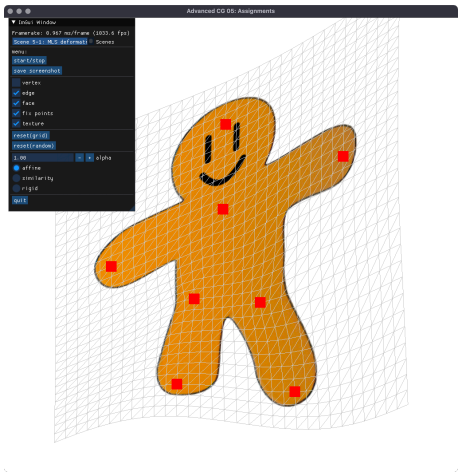

図 1　Affine Deformation の実行結果

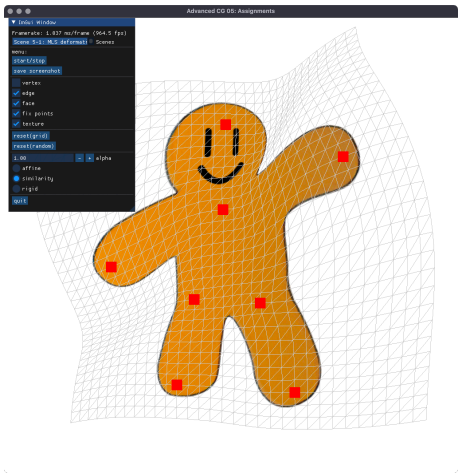

図 2　Similarity Deformation の実行結果

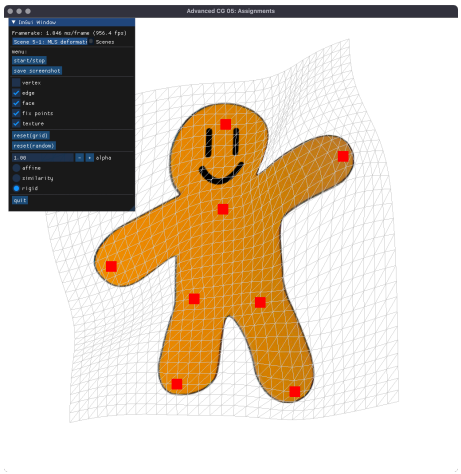

図 3　Rigid Deformation の実行結果