

アドバンスト CG

第7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 線形ブレンドスキーニング (LBS) の実装

Code 1 characteranimation.cpp の skinningLBS 関数

```
1 int CharacterAnimation::skinningLBS(vector<glm::vec3> &vrts, const vector<map<int,
2                                     double>> &weights)
3 {
4     // 頂点毎に変換行列を重みをかけながら適用
5     int nv = (int)(vrts.size());
6     for (int i = 0; i < nv; ++i)
7     {
```

```

7   const int n_joints = static_cast<int>(weights[i].size()); // 頂点
8   v 对応するジョイント数
9
10  glm::vec4 v(vrts[i][0], vrts[i][1], vrts[i][2], 1.0); // 更新前(オリジナル)のス
11  キンメッシュ頂点位置
12
13  // TODO:この部分にLBSによる頂点位置の計算を書く
14  // ・変数v_newにスキンメッシュ頂点vのLBSによる更新後の位置を格納する
15  // ・元の頂点座標は3次元ベクトル(glm::vec3)として格納されているが,
16  // 4x4行列との演算のために4次元ベクトル(glm::vec4)に変換していることに注意
17  // ・頂点vに対応するジョイント番号とその重みは以下のようにして取得できる
18  // map<int, double>::const_iterator itr = weights[i].begin();
19  // for(; itr != weights[i].end(); ++itr){
20  //   int j = itr->first; // ジョイント番号
21  //   float wij = static_cast<float>(itr->second); // 重み
22  //   // ここにジョイントjに関する処理を書く
23  //   //
24  // }
25  // ・ジョイントjのrest(bind)poseでのワールド変換行列(スライドp28のBj)は
26  // m_joints[j].B
27  // に格納されており,回転を含むワールド変換行列(スライドp28のWj)は
28  // m_joints[j].W
29  // に格納されている(どちらもglm::mat4型)
30  // [glmでのベクトル・行列演算について]
31  // ・glm::mat4 M(0.0f)と定義時に引数に0を指定すると0で初期化,
32  // glm::mat4 M(1.0f)と指定すると単位行列で初期化される(LBSでは行列WB^-1を足していくので
33  // 単位行列ではなく...)
34  // ・glmでの行列Mとベクトルvの掛け算は単純にM*vでよい(結果のベクトルが返ってくる)
35  // ・glmでの逆行列計算はglm::inverse()を使うと良い
36
37  glm::vec4 v_new = v; // 更新後のスキンメッシュ頂点位置
38
39  // ----課題ここから----
40  // 頂点vに対応する全ジョイントにおけるループ
41  glm::mat4 R(0.0f);
42  map<int, double>::const_iterator itr = weights[i].begin();
43  for (; itr != weights[i].end(); ++itr)
44  {
45      int j = itr->first; // ジョイント番号
46      float wij = static_cast<float>(itr->second); // 重み
47
48      // 正則判定
49      // if (glm::determinant(m_joints[j].B) == 0.0)
50      if (glm::determinant(m_joints[j].B) < glm::epsilon<float>())
51          continue;
52
53      R += wij * m_joints[j].W * glm::inverse(m_joints[j].B);

```

```

50     }
51
52     // 更新後の頂点座標
53     v_new = R * v;
54
55     // ----課題ここまで----
56
57     vrts[i] = glm::vec3(v_new[0], v_new[1], v_new[2]);
58 }
59
60 return 1;
61 }
```

2.1.2 Dual Quaternion Skinning(DQS) の実装

Code 2 characteranimation.cpp の skinningDQS 関数

```

1 int CharacterAnimation::skinningDQS(vector<glm::vec3> &vrts, const vector<map<int,
2                                     double>> &weights)
3 {
4     // 頂点毎に変換DQ を重みをかけながら適用
5     int nv = (int)(vrts.size());
6     for (int i = 0; i < nv; ++i)
7     {
8         const int n_joints = static_cast<int>(weights[i].size()); // 頂点
9         // v に対応するジョイント数
10        glm::vec3 v = vrts[i]; // 更新前(オリジナル)のスキンメッシュ頂点位置
11
12        // TODO:この部分にDQS による頂点位置の計算を書く
13        // ・変数v_new にスキンメッシュ頂点 v の DQS による更新後の位置を格納する
14        // ・LBS と違って 4x4 行列との掛け算はない(全て四元数との演算)ため,
15        // こちらではglm::vec3として頂点座標を取り出していることに注意
16        // ・頂点v に対応するジョイント番号とその重みは以下のようにして取得できる
17        // map<int, double>::const_iterator itr = weights[i].begin();
18        // for(; itr != weights[i].end(); ++itr){
19        //     int j = itr->first; // ジョイント番号
20        //     float wij = static_cast<float>(itr->second); // 重み
21        //     // ここにジョイントjに関する処理を書く
22        // }
23        // ・ジョイントj の rest(bind) pose でのワールド変換行列(スライドp28のBj)は
24        // m_joints[j].B
25        // に格納されており、回転を含むワールド変換行列(スライドp28のWj)は
26        // m_joints[j].W
27        // に格納されている(どちらもglm::mat4型)
28        // ・四元数や行列については授業スライドの最後の方の説明を参照すること
```

```

28 // •Dual Quaternion を扱うための, DualQuaternion型を定義してある(dualquaternion.h)ので自
29 // 由に使ってください。
30 // 基本的な演算は定義していますが,頂点vの Dual
31 // Quaternion による変換は定義されていないので自分でここに実装してください。
32
33 // ----課題ここから----
34 // 頂点vに対応する全ジョイントにおけるループ
35 DualQuaternion dqi;
36 // 初期化
37 dqi.m_real = glm::quat(0, 0, 0, 0);
38 dqi.m_dual = glm::quat(0, 0, 0, 0);
39
40 map<int, double>::const_iterator itr = weights[i].begin();
41 for (; itr != weights[i].end(); ++itr)
42 {
43     int j = itr->first;           // ジョイント番号
44     float wij = static_cast<float>(itr->second); // 重み
45
46     // 正則判定
47     if (glm::determinant(m_joints[j].B) < glm::epsilon<float>())
48         continue;
49     glm::mat4 M = m_joints[j].W * glm::inverse(m_joints[j].B); // Wj*Bj^-1
50
51     // 回転成分を表す四元数qjを取り出す。
52     glm::quat qj(M);
53     glm::normalize(qj); // 単位四元数を得るため
54
55     // 平行移動成分tjを取り出す。
56     glm::vec3 tj(M[3]); // これBから取り出すかもしれない
57
58     // qj, tjから Dual Quaternion を計算
59     DualQuaternion dqj(qj, tj);
60
61     // 重みwijで頂点 v に対応する全ジョイントの dqj を線形合成 dqi を計算。
62     dqi += wij * dqj;
63 }
64 dqi.normalize(); // 正規化
65
66 // real, dual part の取り出し
67 glm::quat qr = dqi.m_real;
68 glm::quat qd = dqi.m_dual;
69
70 glm::quat q = qr * glm::quat(0.0f, v) * glm::conjugate(qr) + 2.0f * qd * glm::
    conjugate(qr);

```

```

71     // 四元数からベクトル部を取り出す
72     v_new = glm::vec3(q.x, q.y, q.z);
73     // ----課題ここまで----
74
75     vrts[i] = glm::vec3(v_new[0], v_new[1], v_new[2]);
76 }
77
78 return 1;
79 }
```

2.2 実行結果

LBS,DQS で各スキンモデルの変形を行った様子を示す。画像の番号は時系列順になっている。(数字の小さい方から大きい方へ時間が流れている。)

2.2.1 LBS

- arm

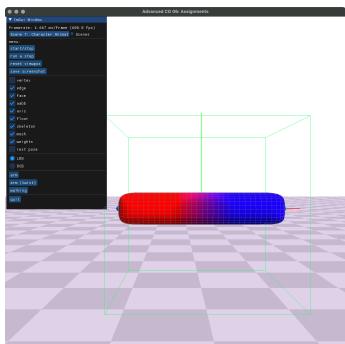


図 1 lbs_arm_00.png

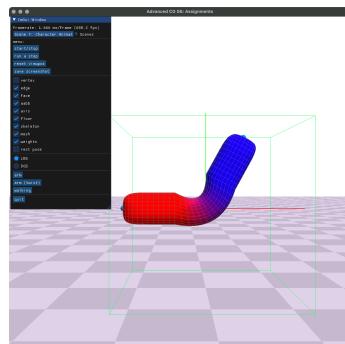


図 2 lbs_arm_01.png

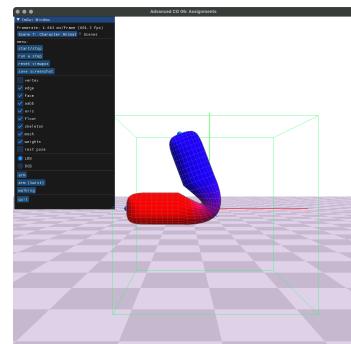


図 3 lbs_arm_02.png

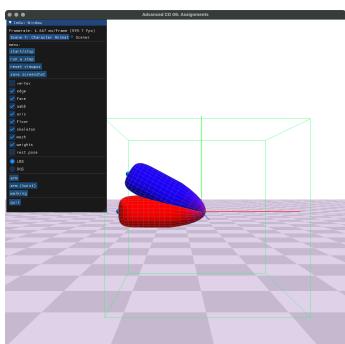


図 4 lbs_arm_03.png

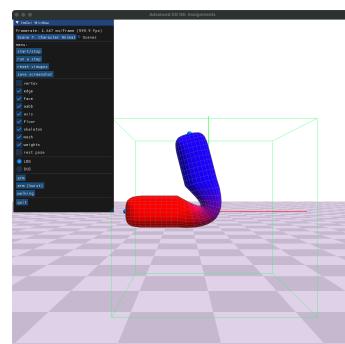


図 5 lbs_arm_04.png

- arm(twist)

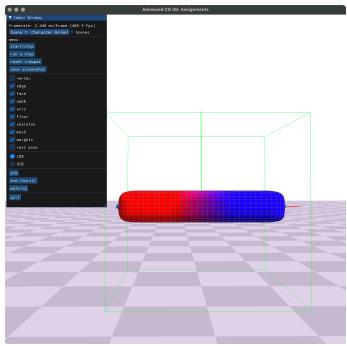


図 6 lbs_twist_00.png

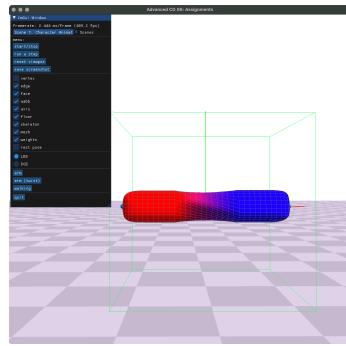


図 7 lbs_twist_01.png

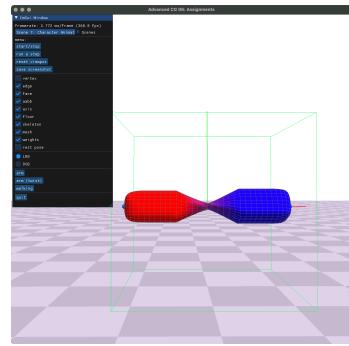


図 8 lbs_twist_02.png

- walking

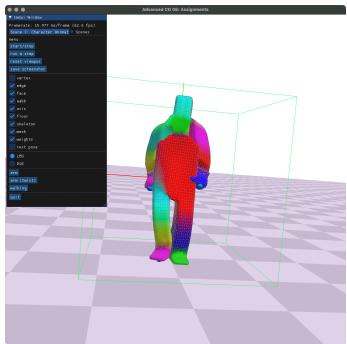


図 9 lbs_walking_00.png

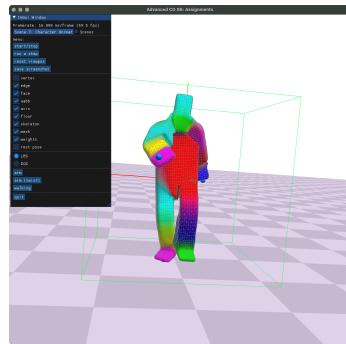


図 10 lbs_walking_01.png

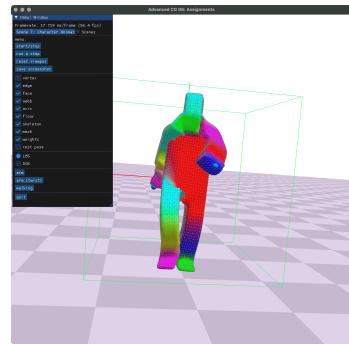


図 11 lbs_walking_02.png

2.2.2 DQS

- arm

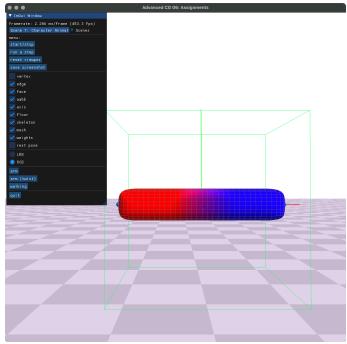


図 12 dqs_arm_00.png

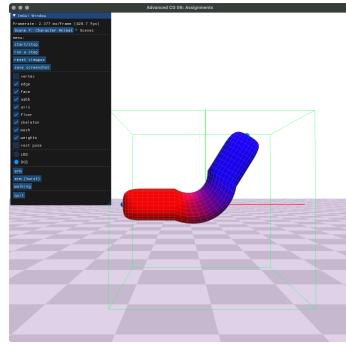


図 13 dqs_arm_01.png

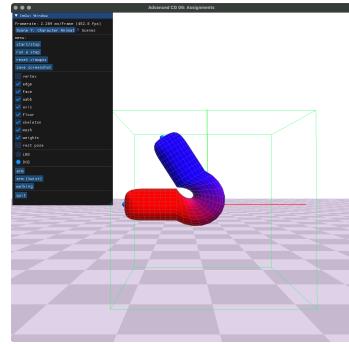


図 14 dqs_arm_02.png

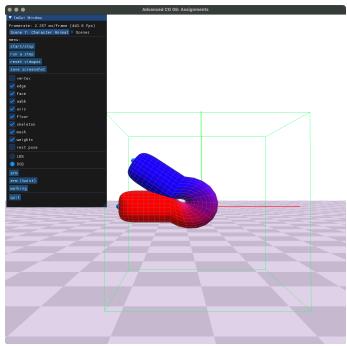


図 15 dqs_arm_03.png

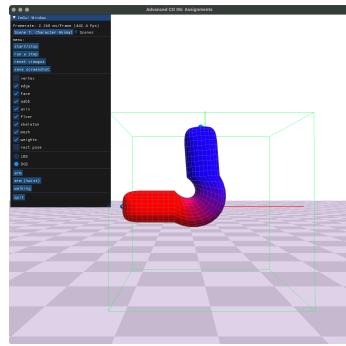


図 16 dqs_arm_04.png

- arm(twist)

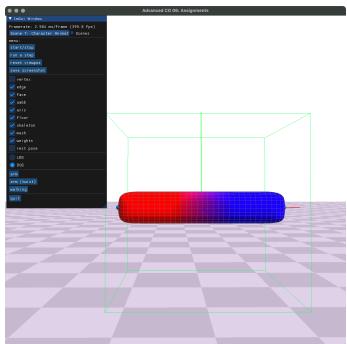


図 17 dqs_twist_00.png

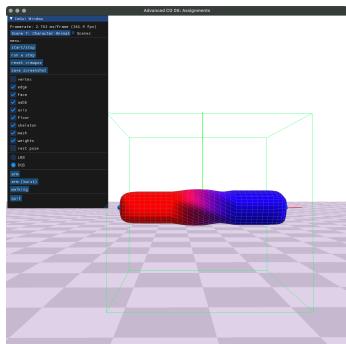


図 18 dqs_twist_01.png

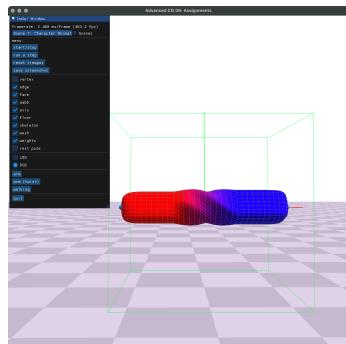


図 19 dqs_twist_02.png

- walking

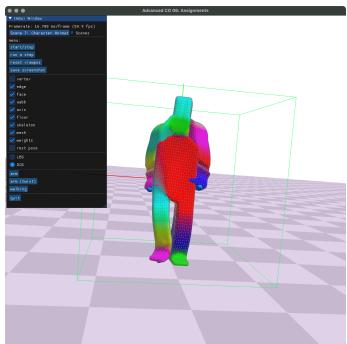


図 20 dqs_walking_00.png

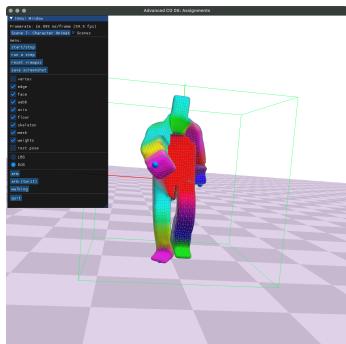


図 21 dqs_walking_01.png

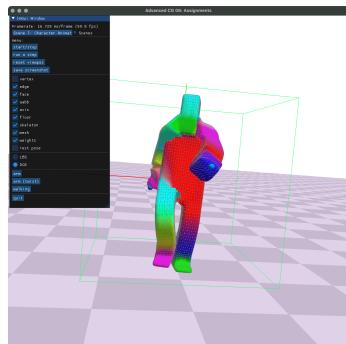


図 22 dqs_walking_02.png