UNIT24: SKINNED MESH – BIND POSE (SKELETON)

【学習要項】
☐Skeleton
☐Bone
☐Bind pose
☐Offset transform

【演習手順】
1．前回使用した FBX ファイル（cube.004.fbx）をロードしオフセット行列とダミーのポーズ行列をセットしテストする

2．skinned_mesh.h に skeleton 構造体を定義する

```
 1:  struct skeleton
 2:  {
 3:    struct bone
 4:    {
 5:      uint64_t unique_id{ 0 };
 6:      std::string name;
 7:      // 'parent_index' is index that refers to the parent bone's position in the array that contains itself.
 8:      int64_t parent_index{ -1 }; // -1 : the bone is orphan
 9:      // 'node_index' is an index that refers to the node array of the scene.
10:      int64_t node_index{ 0 };
11:
12:      // 'offset_transform' is used to convert from model(mesh) space to bone(node) scene.
13:      DirectX::XMFLOAT4X4 offset_transform{ 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1 };
14:
15:      bool is_orphan() const { return parent_index < 0; };
16:    };
17:    std::vector<bone> bones;
18:    int64_t indexof(uint64_t unique_id) const
19:    {
20:      int64_t index{ 0 };
21:      for (const bone& bone : bones)
22:      {
23:        if (bone.unique_id == unique_id)
24:        {
25:          return index;
26:        }
27:        ++index;
28:      }
29:      return -1;
30:    }
31:  };
```

3．skinned_mesh::mesh 構造体にメンバ変数(bind_pose)を追加する

```
skeleton bind_pose;
```

4．FBX メッシュからバインドポーズの情報を抽出する fetch_skeleton メンバ関数を skinned_mesh クラスに実装する

```
 1:  void skinned_mesh::fetch_skeleton(FbxMesh* fbx_mesh, skeleton& bind_pose)
 2:  {
 3:    const int deformer_count = fbx_mesh->GetDeformerCount(FbxDeformer::eSkin);
 4:    for (int deformer_index = 0; deformer_index < deformer_count; ++deformer_index)
 5:    {
 6:      FbxSkin* skin = static_cast<FbxSkin*>(fbx_mesh->GetDeformer(deformer_index, FbxDeformer::eSkin));
 7:      const int cluster_count = skin->GetClusterCount();
 8:      bind_pose.bones.resize(cluster_count);
 9:      for (int cluster_index = 0; cluster_index < cluster_count; ++cluster_index)
10:      {
11:        FbxCluster* cluster = skin->GetCluster(cluster_index);
12:
13:        skeleton::bone& bone{ bind_pose.bones.at(cluster_index) };
14:        bone.name = cluster->GetLink()->GetName();
15:        bone.unique_id = cluster->GetLink()->GetUniqueID();
16:        bone.parent_index = bind_pose.indexof(cluster->GetLink()->GetParent()->GetUniqueID());
```

```
17:       bone.node_index = scene_view.indexof(bone.unique_id);
18:
19:       //'reference_global_init_position' is used to convert from local space of model(mesh) to
20:       // global space of scene.
21:       FbxAMatrix reference_global_init_position;
22:       cluster->GetTransformMatrix(reference_global_init_position);
23:
24:       // 'cluster_global_init_position' is used to convert from local space of bone to
25:       // global space of scene.
26:       FbxAMatrix cluster_global_init_position;
27:       cluster->GetTransformLinkMatrix(cluster_global_init_position);
28:
29:       // Matrices are defined using the Column Major scheme. When a FbxAMatrix represents a transformation
30:       // (translation, rotation and scale), the last row of the matrix represents the translation part of
31:       // the transformation.
32:       // Compose 'bone.offset_transform' matrix that trnasforms position from mesh space to bone space.
33:       // This matrix is called the offset matrix.
34:       bone.offset_transform
35:         = to_xmfloat4x4(cluster_global_init_position.Inverse() * reference_global_init_position);
36:     }
37:   }
38: }
```

5．skinned_mesh クラスの fetch_meshes メンバ関数で fetch_skeleton メンバ関数を呼び出す
※fetch_bone_influences メンバ関数呼び出しの直後で呼び出す

```
fetch_skeleton(fbx_mesh, mesh.bind_pose);
```

6．skinned_mesh クラスの render メンバ関数でダミー行列を定数バッファ(data.bone_transforms)にセットする
※動作確認後#if-#endif ディレクティブのコードは無効にすること

```
 1: #if 1
 2:     // Bind pose transform(Offest matrix) : Convert from the model(mesh) space to the bone space
 3:     XMMATRIX B[3];
 4:     B[0] = XMLoadFloat4x4(&mesh.bind_pose.bones.at(0).offset_transform);
 5:     B[1] = XMLoadFloat4x4(&mesh.bind_pose.bones.at(1).offset_transform);
 6:     B[2] = XMLoadFloat4x4(&mesh.bind_pose.bones.at(2).offset_transform);
 7:
 8:     // Animation bone transform : Convert from the bone space to the model(mesh) or the parent bone space
 9:     XMMATRIX A[3];
10:     // from A0 space to model space
11:     A[0] = XMMatrixRotationRollPitchYaw(XMConvertToRadians(90), 0, 0);
12:
13:     // from A1 space to parent bone(A0) space
14:     A[1] = XMMatrixRotationRollPitchYaw(0, 0, XMConvertToRadians(45)) * XMMatrixTranslation(0, 2, 0);
15:
16:     // from A2 space to parent bone(A1) space
17:     A[2] = XMMatrixRotationRollPitchYaw(0, 0, XMConvertToRadians(-45)) * XMMatrixTranslation(0, 2, 0);
18:
19:     XMStoreFloat4x4(&data.bone_transforms[0], B[0] * A[0]);
20:     XMStoreFloat4x4(&data.bone_transforms[1], B[1] * A[1] * A[0]);
21:     XMStoreFloat4x4(&data.bone_transforms[2], B[2] * A[2] * A[1] * A[0]);
22: #endif
```

7．実行し、メッシュの変形を確認する（スクリーンショット画像：cube.004.2.png）

【評価項目】
□バインドポーズ
□オフセット行列