III. Topics
23. Character Animation

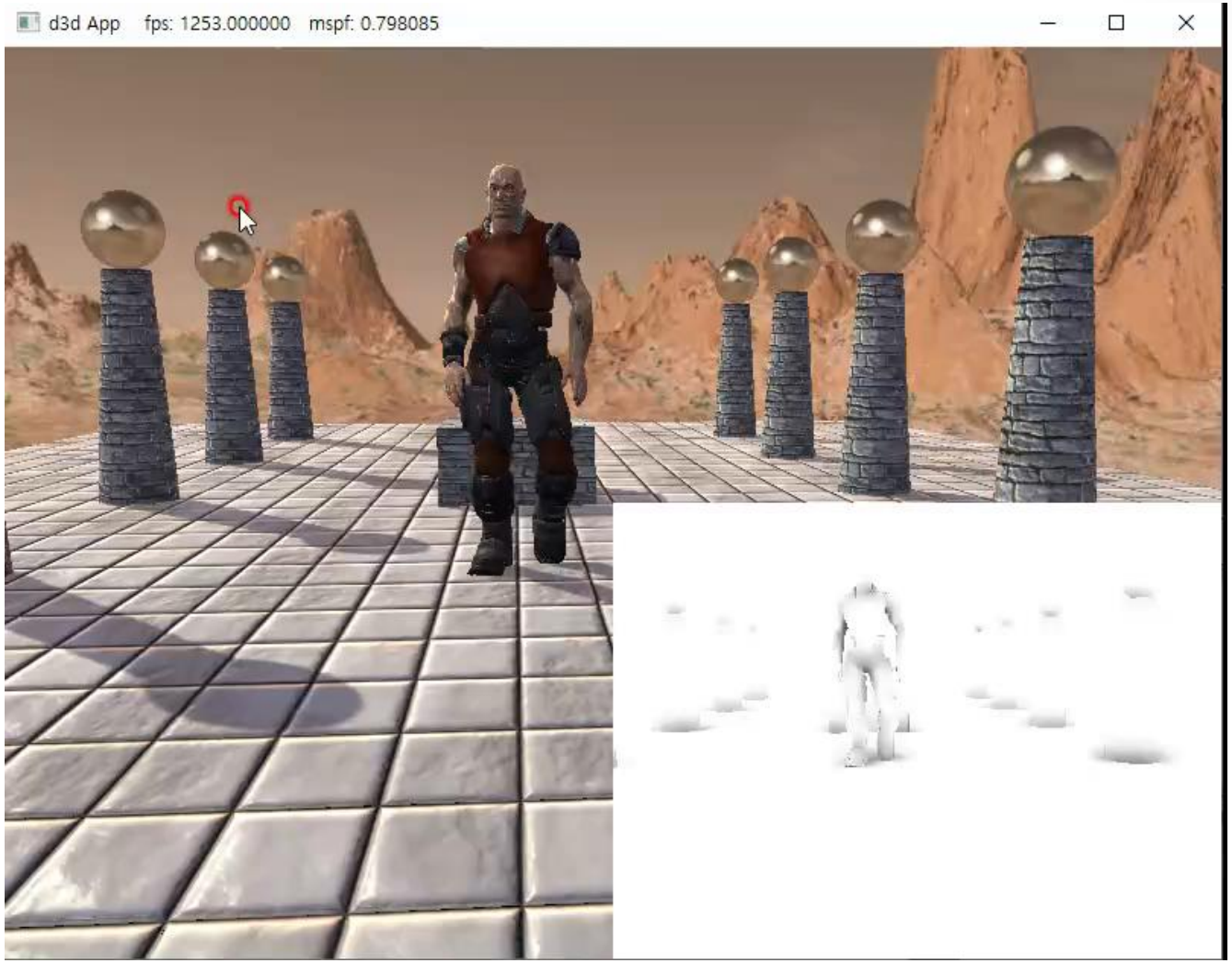
Game Graphic Programming

Kyung Hee University

Software Convergence

Prof. Daeho Lee

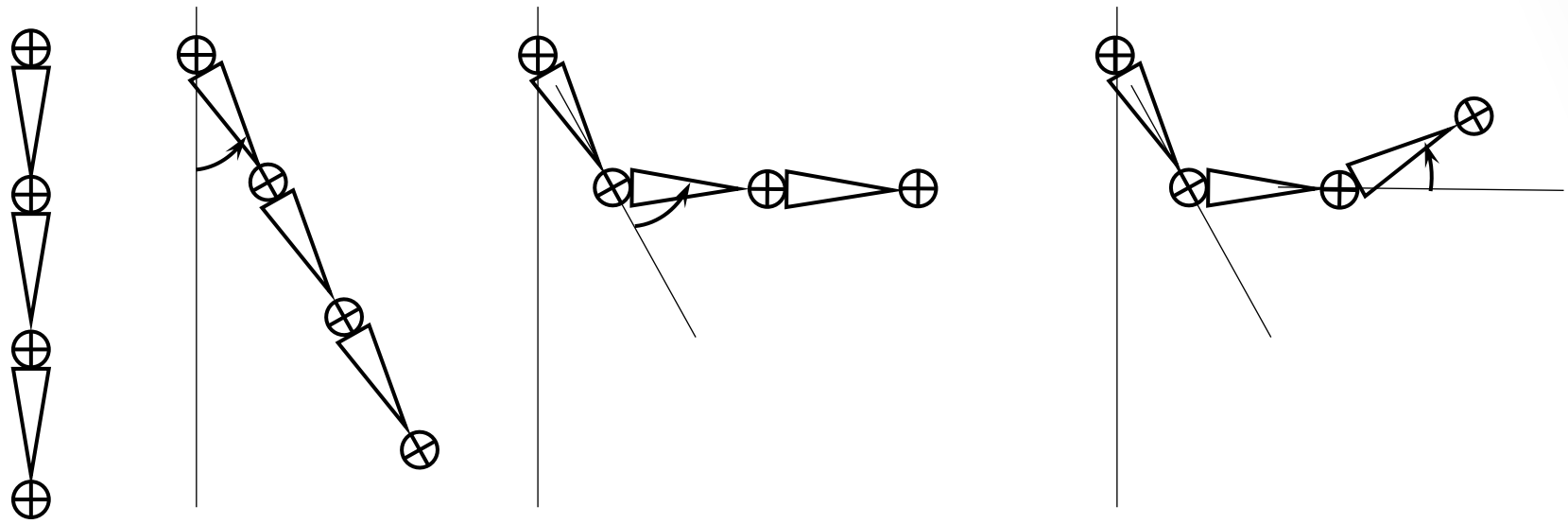
Skinned Mesh



Kyung Hee University
nize@khu.ac.kr

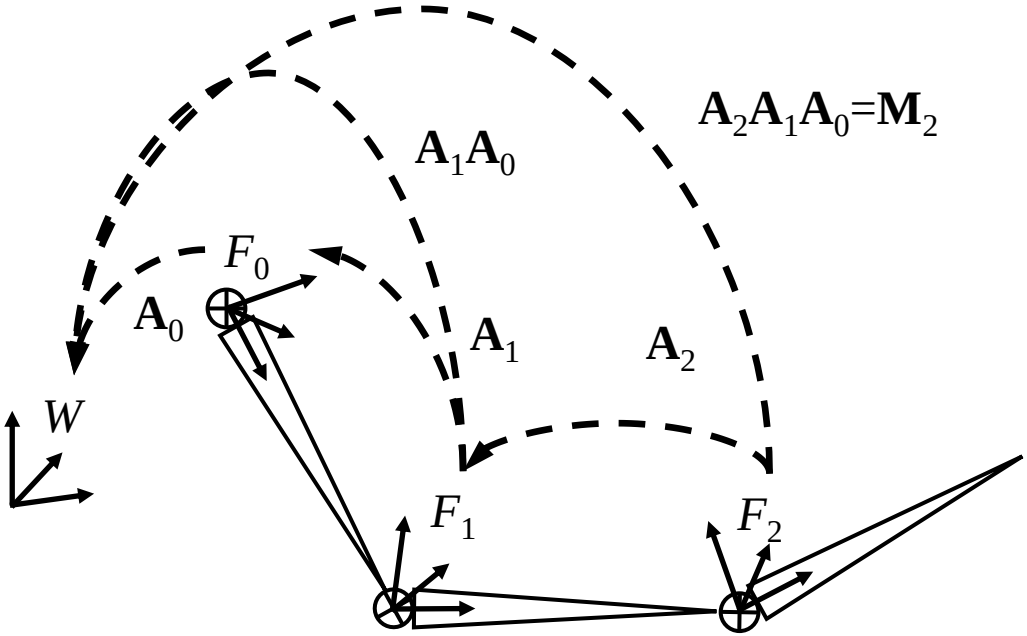
Frame Hierarchies

- Bone hierarchies
- Many objects are composed of parts with a parent-child relationship.
  - (e.g., upper arm, forearm, and hand)
  - Hierarchy transforms; observe that the parent transformation of a bone influences itself and all of its children.

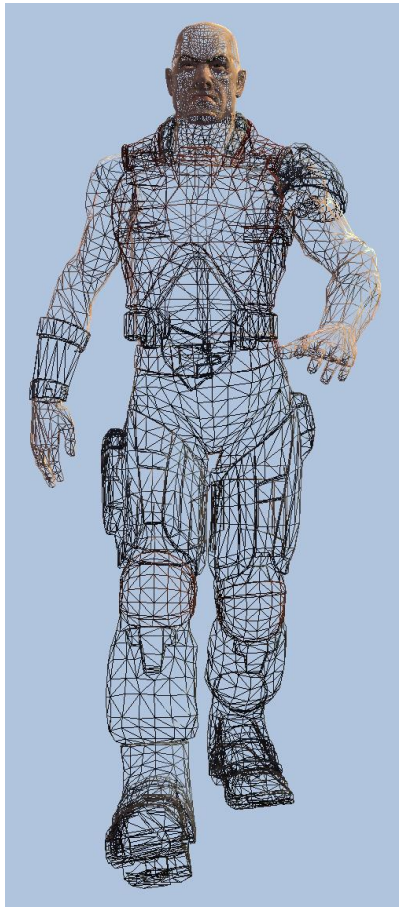


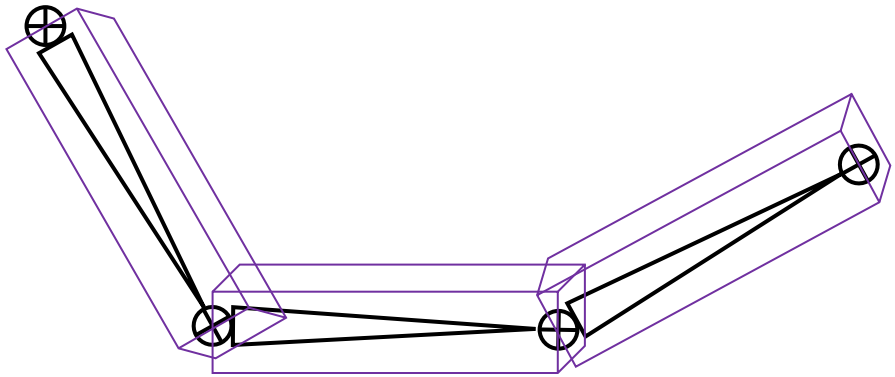
Mathematical Formulation

- Mathematical formulation
  - Each object in the hierarchy is modeled about its own local coordinate system, with its pivot joint at the origin to facilitate rotation.
  - F_i is the ith joint coordinate relative to the world coordinate system.
  - $\mathbf{A}_i$ is a to-parent matrix since it transforms geometry from a child's coordinate system into its parent's coordinate system.

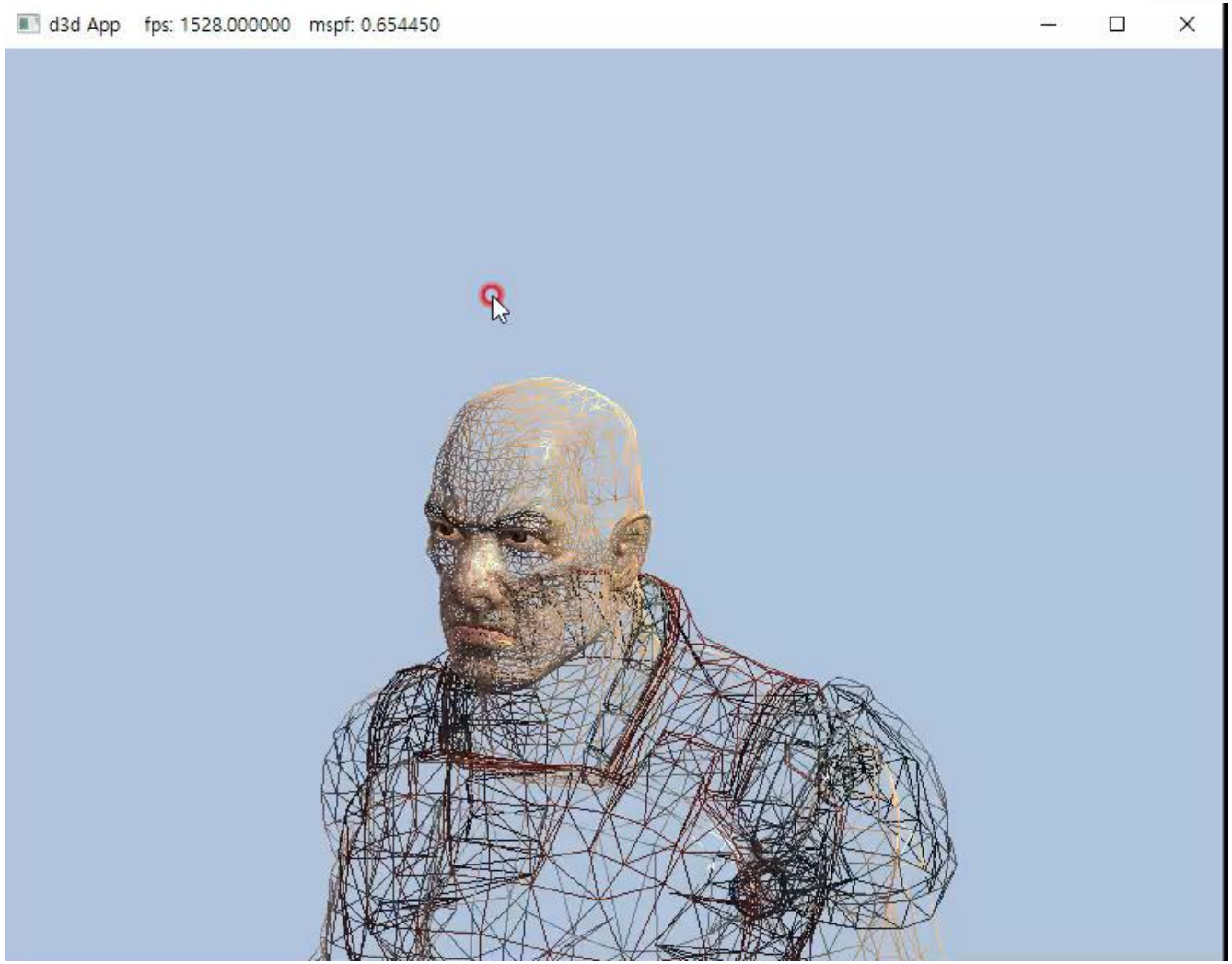


Skinned Meshes (1)





Skinned Meshes (2)



Skinned Meshes (3)

- Reformulating the bones to-root transform
  - $\text{toRoot}_i = \text{toParent}_i \cdot \text{toRoot}_p$
- Offset transform
  - The skin vertices are relative to the bind space, which is the coordinate system the mesh was modeled in.
  - The offset transform can change the vertices from bind space to the space of the bone that influences the vertices.
- Final transform
  - $F_i = \text{offset}_i \cdot \text{toRoot}_i$

Vertex Blending

- One or more bones can influence a vertex of the skin.

$$\mathbf{n}' = \text{normalize}(w_0\mathbf{nF}_0 + w_1\mathbf{nF}_1 + w_2\mathbf{nF}_2 + w_3\mathbf{nF}_3)$$
$$\mathbf{t}' = \text{normalize}(w_0\mathbf{tF}_0 + w_1\mathbf{tF}_1 + w_2\mathbf{tF}_2 + w_3\mathbf{tF}_3)$$

```
**********Vertices**********
Position: -0.9983482 67.88861 4.354969
Tangent: -0.257403 0.5351538 -0.8045831 1
Normal: 0.5368453 0.7715151 0.3414111
Tex-Coords: 0.695366 0.9909569
BlendWeights: 0.7470379 0.2529621 0 0  // 0.7470379 + 0.2529621 = 1
BlendIndices: 7 9 0 0                 // bone indices

Position: -0.8930981 67.9022 4.388191
Tangent: 0.6117722 0.7648141 -0.2019754 1
Normal: -0.4419364 0.542224 0.7146226
Tex-Coords: 0.69473 0.983095
BlendWeights: 0.7382242 0.2617758 0 0
BlendIndices: 7 9 0 0
```

M3D Loader (1)

```
// LoadM3d.cpp/h
class M3DLoader {
   void ReadBoneOffsets(std::ifstream& fin, UINT numBones,
      std::vector<DirectX::XMFLOAT4X4>& boneOffsets);
   void ReadBoneHierarchy(std::ifstream& fin, UINT numBones,
      std::vector<int>& boneIndexToParentIndex);
   void ReadAnimationClips(std::ifstream& fin, UINT numBones,
      UINT numAnimationClips,
      std::unordered_map<std::string, AnimationClip>& animations);

//…
};
```

M3D: Model 3D

M3D Loader (2)

```
// soldier.m3d
***************m3d-File-Header***************
#Materials 5
#Vertices 13748
#Triangles 22507
#Bones 58
#AnimationClips 1

***************Materials**********************
Name: soldier_head
Diffuse: 1 1 1
Fresnel0: 0.05 0.05 0.05
Roughness: 0.5
AlphaClip: 0
MaterialTypeName: Skinned
DiffuseMap: head_diff.dds
NormalMap: head_norm.dds

Name: soldier_jacket
// …
```

Textures/jacket_diff.dds

M3D Loader (3)

```
// soldier.m3d
***************Vertices**********************
Position: -0.9983482 67.88861 4.354969
Tangent: -0.257403 0.5351538 -0.8045831 1
Normal: 0.5368453 0.7715151 0.3414111
Tex-Coords: 0.695366 0.9909569
BlendWeights: 0.7470379 0.2529621 0 0  // 0.7470379 + 0.2529621 = 1
BlendIndices: 7 9 0 0

Position: -0.8930981 67.9022 4.388191
Tangent: 0.6117722 0.7648141 -0.2019754 1
Normal: -0.4419364 0.542224 0.7146226
Tex-Coords: 0.69473 0.983095
BlendWeights: 0.7382242 0.2617758 0 0
BlendIndices: 7 9 0 0
// …
```

M3D Loader (4)

```
// soldier.m3d
***************Triangles**********************
0 1 2
3 4 5
6 7 8
9 10 11
12 13 14
15 16 17
18 19 20
19 18 21
21 22 19
23 19 22
24 19 23
19 24 25
21 18 26
26 27 21
// …
```

M3D Loader (5)

```
// soldier.m3d
***************BoneOffsets*******************
BoneOffset0 1.331581E-06 1 4.16881E-11 0 -5.551115E-17 -4.16881E-11 1 0 1
-1.331581E-06 8.412415E-23 0 -2.230549 -0.03756041 -37.46099 1
BoneOffset1 4.16881E-11 2.757707E-06 1 0 1 -2.25986E-16 -4.16881E-11 0
2.220447E-16 1 -2.757707E-06 0 -37.46099 -1.274681 -0.03755855 1
BoneOffset2 1.240504E-06 -1.274266E-06 1 0 0.9954989 0.09477358 -
1.114154E-06 0 -0.09477358 0.9954989 1.386098E-06 0 -40.94087 -5.174486 -
// …
BoneOffset54 -0.2109876 -0.1779905 -0.961147 0 -0.975253 0.1047948
0.1946776 0 0.06607245 0.9784362 -0.1956962 0 35.72023 -5.788335 -
10.36661 1
BoneOffset55 -0.1409264 -0.2373523 -0.961147 0 -0.9556651 -0.220918
0.1946776 0 -0.2585419 0.9459698 -0.1956961 0 18.3955 0.2603853 -10.36661
1
BoneOffset56 0.00974141 -0.2615723 -0.9651347 0 -0.9980835 -0.06152691
0.006601164 0 -0.06110846 0.9632207 -0.2616704 0 4.902436 -0.7261673 -
9.58015 1
BoneOffset57 -0.2555867 0.05648369 -0.9651347 0 0.1919549 0.9813814
0.006601127 0 0.9475382 -0.1835751 -0.2616703 0 -6.686508 2.100421 -
9.58015 1
```

M3D Loader (6)

```
// soldier.m3d
***************BoneHierarchy*****************
ParentIndexOfBone0: -1
ParentIndexOfBone1: 0
ParentIndexOfBone2: 1
ParentIndexOfBone3: 2
ParentIndexOfBone4: 3
ParentIndexOfBone5: 4
ParentIndexOfBone6: 5
ParentIndexOfBone7: 6
// …
ParentIndexOfBone49: 48
ParentIndexOfBone50: 2
ParentIndexOfBone51: 50
ParentIndexOfBone52: 51
ParentIndexOfBone53: 52
ParentIndexOfBone54: 2
ParentIndexOfBone55: 54
ParentIndexOfBone56: 55
ParentIndexOfBone57: 56
```

M3D Loader (7)

```
// soldier.m3d
***************AnimationClips****************
AnimationClip Take1
{
  Bone0 #Keyframes: 76
  {
  Time: 0 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996 0.5000004
  Time: 0.0166666 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996  // …
  Time: 0.0333333 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996  // …
  Time: 0.05 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996 0.5000004
  Time: 0.0666666 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996  // …
  Time: 0.0833333 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996  // …
  Time: 0.0999999 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996  // …
  Time: 0.1166666 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996  // …
  // …
  Time: 1.2 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996 0.5000004
  Time: 1.2166666 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996 // …
  Time: 1.2333333 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996 // …
  Time: 1.25 Pos: 0.03756343 37.46099 2.230549 Scale: 1 1 1 Quat: -0.5000003 -0.4999996 -0.4999996 0.5000004
  }

  Bone1 #Keyframes: 76
  {
// …
```

Animation Clip Class

```
// SkinnedData.cpp/h
struct AnimationClip {
   float GetClipStartTime()const;
   float GetClipEndTime()const;

   void Interpolate(float t,
      std::vector<DirectX::XMFLOAT4X4>& boneTransforms)const;

    std::vector<BoneAnimation> BoneAnimations;
};
```

Animation Data Class (1)

```
// SkinnedData.cpp/h
class SkinnedData {
public:
   UINT BoneCount()const;

   float GetClipStartTime(const std::string& clipName)const;
   float GetClipEndTime(const std::string& clipName)const;

   void Set(std::vector<int>& boneHierarchy,
      std::vector<DirectX::XMFLOAT4X4>& boneOffsets,
      std::unordered_map<std::string, AnimationClip>& animations);

  void GetFinalTransforms(const std::string& clipName, float timePos,
       std::vector<DirectX::XMFLOAT4X4>& finalTransforms)const;
private:
   std::vector<int> mBoneHierarchy;
   std::vector<DirectX::XMFLOAT4X4> mBoneOffsets;
   std::unordered_map<std::string, AnimationClip> mAnimations;
};
```

Animation Data Class (2)

```
// SkinnedData.cpp/h
void SkinnedData::GetFinalTransforms(const std::string& clipName,
    float timePos,  std::vector<XMFLOAT4X4>& finalTransforms)const {

    UINT numBones = mBoneOffsets.size();
    std::vector<XMFLOAT4X4> toParentTransforms(numBones);

    auto clip = mAnimations.find(clipName);
    clip->second.Interpolate(timePos, toParentTransforms);

    std::vector<XMFLOAT4X4> toRootTransforms(numBones);

    toRootTransforms[0] = toParentTransforms[0];
```

Animation Data Class (3)

```
// SkinnedData.cpp/h

   for(UINT i = 1; i < numBones; ++i) {
      XMMATRIX toParent = XMLoadFloat4x4(&toParentTransforms[i]);
      int parentIndex = mBoneHierarchy[i];
      XMMATRIX parentToRoot
         = XMLoadFloat4x4(&toRootTransforms[parentIndex]);
      XMMATRIX toRoot = XMMatrixMultiply(toParent, parentToRoot);
      XMStoreFloat4x4(&toRootTransforms[i], toRoot);
   }
   for(UINT i = 0; i < numBones; ++i) {
      XMMATRIX offset = XMLoadFloat4x4(&mBoneOffsets[i]);
      XMMATRIX toRoot = XMLoadFloat4x4(&toRootTransforms[i]);
      XMMATRIX finalTransform = XMMatrixMultiply(offset, toRoot);
      XMStoreFloat4x4(&finalTransforms[i],
         XMMatrixTranspose(finalTransform));
   }
}
```

Model Instance Class

```
// Application
struct SkinnedModelInstance {
   SkinnedData* SkinnedInfo = nullptr;
   std::vector<DirectX::XMFLOAT4X4> FinalTransforms;
   std::string ClipName;
   float TimePos = 0.0f;

   void UpdateSkinnedAnimation(float dt) {
      TimePos += dt;
      if(TimePos > SkinnedInfo->GetClipEndTime(ClipName))
         TimePos = 0.0f;

      // Compute the final transforms for this time position.
      SkinnedInfo->GetFinalTransforms(ClipName, TimePos,
         FinalTransforms);
    }
};
```

Load Model

```
// Application
class SkinnedMeshApp : public D3DApp {
// …
    std::string mSkinnedModelFilename = "Models\\soldier.m3d";
    std::unique_ptr<SkinnedModelInstance> mSkinnedModelInst;
    SkinnedData mSkinnedInfo;
    std::vector<M3DLoader::Subset> mSkinnedSubsets;
    std::vector<M3DLoader::M3dMaterial> mSkinnedMats;
// …
};
void SkinnedMeshApp::LoadSkinnedModel() {
    std::vector<M3DLoader::SkinnedVertex> vertices;
    std::vector<std::uint16_t> indices;
    M3DLoader m3dLoader;
    m3dLoader.LoadM3d(mSkinnedModelFilename, vertices, indices,
        mSkinnedSubsets, mSkinnedMats, mSkinnedInfo);

    mSkinnedModelInst = std::make_unique<SkinnedModelInstance>();
    mSkinnedModelInst->SkinnedInfo = &mSkinnedInfo;
    mSkinnedModelInst->FinalTransforms.resize(mSkinnedInfo.BoneCount());
// …
}
```

Update Constant Buffers

```
// Application
void SkinnedMeshApp::UpdateSkinnedCBs(const GameTimer& gt) {
   auto currSkinnedCB = mCurrFrameResource->SkinnedCB.get();
   mSkinnedModelInst->UpdateSkinnedAnimation(gt.DeltaTime());
   SkinnedConstants skinnedConstants;

   std::copy(
      std::begin(mSkinnedModelInst->FinalTransforms),
      std::end(mSkinnedModelInst->FinalTransforms),
      &skinnedConstants.BoneTransforms[0]);

   currSkinnedCB->CopyData(0, skinnedConstants);
}
```