D3DXLoadMeshHierarchyFromX()

Output🡪D3DXFRAME

Output🡪ID3DXAnimationController

**SKINNED MESH**

1. Define mesh: implement ID3DXMesh by calling D3DXCreateMeshFVF()
   1. Build vertex buffer
   2. Build index buffer
   3. Build attribute(subset) buffer
   4. Optimize by using adjacency buffer (adjacency buffer can be gained by calling ::GenerateAdjacency())
2. Define bones(with no hierarchy): implement ID3DXSkinInfo by calling D3DXCreateSkinInfo()
   1. Build bones (bone name, offset matrix)
   2. Build bone influence (influencing vertex indices and respective weights)
   3. Define skinned mesh: Implement (skinned) ID3DXMesh by calling ::ConvertToIndexedBlendedMesh(), which integrates step 1’s mesh data and step 2’s bones into skinned mesh.
3. Setup bone hierarchy using D3DXFRAME derived structure

**ANIMATION**

1. Define animation set: implement ID3DXKeyframedAnimationSet by calling D3DXCreateKeyframedAnimationSet()
   1. Build SRT transformation key array corresponding to *each bone*
   2. Register keys by calling ::RegisterAnimationSRTKeys() corresponding to *each bone*
2. Define animation controller: implement ID3DXKeyframedAnimationController by calling D3DXCreateAnimationController()
   1. Register animation set by calling ::RegisterAnimationSet()
   2. Register animation output by calling ::RegisterAnimationOutput() corresponding to *each bone* (D3DXFRAME derived structure)
   3. Setup animation track to enable the registered animation set by calling ::SetTrack\*() methods

**VERTEX SHADER WITH VERTEX BLENDING FEATURE**

1. Implement vertex shader using ID3DXEffect

**RENDER**

1. Call ID3DXAnimationController::AdvanceTime() to update SRT transformation which corresponds to each bone
2. Calculate final transform of each bone(final transform array); multiply following matrices
   1. Offset matrix of corresponding bone
   2. D3DXFRAME derived structure’s TransformationMatrix member variable, recursively
3. Send final transform array to vertex shader
4. Render