|  |  |  |  |
| --- | --- | --- | --- |
| MORPHOSIS 개발일지 8주차 | | | |
| 기간 | 2018-02-22 ~ 2019-03-20 | 작성자 | 신재욱 |
| 작업 내용 | | | |
| - 컨버터를 FBX SDK의 함수들을 사용하여 작성하였습니다.  이를 위해 정리한 내용은 다음과 같습니다.  Animation Data 부분  필요한 내용은 다음과 같다.     * **Bone**의 LclTranslation, LclRotation, Parent <- **Bone Data** * **Keyframe**의 KeyTime, EffectedBones <- **Animation Data**     **[Bone Data]**  FbxVector4 lTmpVector;  lTmpVector = pNode->GetGeometricTranslation(FbxNode::eSourcePivot);  lTmpVector[0], lTmpVector[1], lTmpVector[2]로 Translation.xyz에 접근가능  LclRotation도 마찬가지로 접근가능.    Parent는 pNode->GetParent()로 ParentNode에 접근할 수 있다.  현재 어떤 방식으로 저장해야 하는지 생각해야 함.    **[Animation Data]**  KeyTime, BoneList를 얻어야 함.  FbxAnimCurve\* lAnimCurve = NULL;  lAnimCurve = pNode->LclTranslation.GetCurve(pAnimLayer, FBX\_CURVENODE\_COMPONNENT\_X);  DisplayCurve(lAnimCurve);    해당 Bone이 가진 LclTranslation.x의 시간별 값(KeyTime, KeyValue)을 알 수 있다.    **<Animation Data 예시>**   * X축에 대해 단일 데이터  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | KeyTime | 0.0s | 0.3s | 0.7s | 1.0s | 1.3s | 1.7s | | Bone001 | xxx | xxx | xxx | xxx | xxx | xxx | | Bone002 | xxx |  | xxx | xxx | xxx |  | | Bone003 | xxx |  | xxx |  |  | xxx | | Bone004 | xxx |  |  | xxx |  | xxx | | Bone005 | xxx |  | xxx |  |  |  |  * 모든 축에 대해 단일 데이터(Rotation 속성)  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | KeyTime | 0.0s | 0.3s | 0.7s | 1.0s | 1.3s | 1.7s | | Bone001 | (x,y,z) | (x,y,z) | (x,y,z) | (x,y,z) | (x,y,z) | (x,y,z) | | Bone002 | (x,y,z) |  | (x,y,z) | (x,y,z) | (x,y,z) |  | | Bone003 | (x,y,z) |  | (x,y,z) |  |  | (x,y,z) | | Bone004 | (x,y,z) |  |  | (x,y,z) |  | (x,y,z) | | Bone005 | (x,y,z) |  | (x,y,z) |  |  |  |   3D Max에서 저장 시에 모든 축에 대하여 키프레임을 저장해야 함.  지금은 그것을 기반으로 작업하고 있음.  이후에 다음과 같은 데이터도 읽을 수 있도록 변경해야 함.   * 모든 축에 대해 단일 데이터(Rotation 속성)  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | KeyTime | 0.0s | 0.3s | 0.7s | 1.0s | 1.3s | 1.7s | | Bone001 | (x,y,z) | (x) | (y) | (y,z) | (x,y) | (y,z) | | Bone002 | (x,y,z) |  | (x) | (y) | (z) |  | | Bone003 | (x,y,z) |  | (x,z) |  |  | (x,y,z) | | Bone004 | (x,y,z) |  |  | (x,y,z) |  | (x,y,z) | | Bone005 | (x,y,z) |  | (x,y,z) |  |  |  |     현재 직접적으로 얻을 수 있는 정보는 어떤 Bone인지, 어떤 속성인지, 어떤 축인지, 어떤 KeyTime인지, KeyValue는 어떤지 등.  따라서 KeyTime과 Bone을 기준으로 데이터를 저장하면 될 것.  일단 작성해보자.    STL set은 iterator를 통해 직접 접근하여 내부 값을 수정할 수가 없었다.(const Template) 따라서 STL Vector로 변경하여 사용함.  (시간의 순서를 고려할 필요가 없었으므로)(아마 나중에 다시 set으로 빼도 될 듯)  C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image001.png  위와 같이 시간별로 Bone마다 Translation과 Rotation을 받는데 성공함.   Geometry Data 부분 추출해야 하는 내용은 다음과 같다.    Geometry Data <- 기본적인 Mesh Data   * Vertex point * Float2 uv * Float3 normal * Int[] posIdx * Int[] uvIdx     Vertex <- 여기에 Bone의 내용이 들어감   * Float3 pos * Int4 boneIdx * Float4 weight     Animation Data   * 다른 페이지에서 설명하고 있음.     따라서 먼저 Geometry Data를 추출하고, 모든 Bone을 돌면서 영향을 주는 Geometry Data.point[n]에게 해당하는 bone의 idx와 weight를 준다. Bone도 저장해야 함.    pNode에게 어떤 속성이 있는지 확인하고 있으면 읽어오는게 가능했던 거 같은데 어떻게 하더라.    **[Mesh]**    pMesh->GetControlPointsCount();  pMesh->GetElementNormalCount();  pMesh->GetControlPoints();  이걸로 접근가능함.  .fbx 파일의 Vertices가 ControlPoint의 정보를 담고 있음.    Cluster에서 받는 영향을 주는 CP의 정보가 각 Mesh 별로 주어지므로 Cluster가 어떤 Mesh와 연결되어 있는지 알아야 함.    Geometry로도 Controlpoint에 접근할 수 있음을 확인.  UV와 Normal, PolygonVertexIndex는 아직 잘 모르겠다.  UV는 FbxVector2나 FbxVector4로 못 받는 것을 확인.  FbxGeometryElementUV로 받아야 했음. 근데 이거 어떻게 쓰는건지 잘 모르겠다. Ppt를 보자.  GetElementUV(int nIndex = 0);  GetElementUVCount()가 UV 전체 개수가 아니었음.  UV 데이터는 UV랑 uvIdx 이렇게 두 개가 나와야 하는데 어떻게 해야할 지 모르겠다.    C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image001.png  MappingInformationType과 ReferenceInformationType에 따라 UV 받는 방식을 달리해야 함.  C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image002.png    **[Cluster를 통한 BoneIdx, weight]**    C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image003.png    FbxNode\* tmp = lScene->GetRootNode();  FbxGeometry\* geo = tmp->GetGeometry();  int nSkinDeformers = geo->GetDeformerCount(FbxDeformer::eSkin);  for (int i = 0; i < nSkinDeformers; ++i) {  FbxSkin\* skinDeformer = (FbxSkin\*)(geo->GetDeformer(i, FbxDeformer::eSkin));  int nClusters = skinDeformer->GetClusterCount();  for (int j = 0; j < nClusters; ++j) {  FbxCluster\* cluster = skinDeformer->GetCluster(j);  int nIdx = cluster->GetControlPointIndicesCount();  int\* pIdx = cluster->GetControlPointIndices();  double\* pWeights = cluster->GetControlPointWeights();  }  }    pNode->GetNodeAttribute()->GetAttributeType();이 nullptr을 반환하여 FbxNodeAttribute::Etype 과 맞지 않는 문제가 있었다.  이는 GetNodeAttribute()에서 NULL을 반환해서 생긴 문제였음.    pNode->GetName()을 하면 Body, Jumper, Mask 이 이름이 나옴. 기대했던건 Bone의 이름이었는데.  C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image004.png  보니까 Cluster가 Bone이 아니었기 때문에 이런 일이 생기는거 같음  C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image005.png  따라서 pNode->GetLink()함수로 Bone을 얻어왔다.    이로서 GeometryData에서 얻을 수 있는 정보는  어떤 bone인지, 그 bone이 어떤 vertex에게 영향을 주는지, weight는 몇인지 이다.    C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image006.png  어떤 mesh에 해당하는 Cluster인지 알 수 있음. Animation 계산 부분 Bone은 LclTranslation, LclRotation, Parent를 갖는다.  (toParent, toDresspose, toDressposeInv, toWorld      Geometry Data는 Vertex, Index, UV, Normal 등의 Mesh Data와 Bone, Bone이 영향을 주는 Vertex, 해당 Vertex에 대한 weight 등의 Bone Data가 있다.  Animation Data은 시간에 따라 Bone의 위치나 회전각을 맞춰주는 것이 전부.   Exporter 부분 **[내용 정리]**  Cluster Data:   * 어떤 제어점에게 영향을 주는가. * 그 제어점에게 얼마나 영향을 주는가. * 이 클러스터가 어떤 메쉬와 연결되어 있는가.     Geometry Data:   * 그 메쉬에 어떤 제어점이 있는가.     Bone Data:   * 본의 계층구조는 어떻게 되는가. * 본의 위치 정보는 어떻게 되는가.     Animation Data:   * 언제 어떤 본이 얼마만큼 움직이는가.     **[DataManager]**  기존의 GetMeshDataRec()가 얻어내던 것이 ControlPoint가 아니라 PolygonVertexIndex임을 확인.  C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image001.png  아나 진짜.  PolygonVertexIndex는 따로 저장하도록 하자.  C:\Users\신재욱\AppData\Local\Packages\Microsoft.Office.OneNote_8wekyb3d8bbwe\TempState\msohtmlclip\clip_image002.png  기존에 쓰던 코드를 그대로 사용하여 PolygonVertexIndex를 얻어내는데 성공.    **[대략적인 순서]**  위의 내용들을 고려하여 Exporter를 작성해야 함.   1. Geometry Data를 먼저 추출함. 2. Bone Data를 추출. 3. Cluster Data를 따라서 CPs의 ControlPoint들에게 값 추가. 4. Animation Data를 추출. 5. Geometry Data와 Bone Data를 묶어서 저장. 6. Animation Data를 따로 저장.     읽을 때는   1. 묶어서 저장한 Data를 읽어서 Bone과 Mesh Data를 저장함. 2. 따로 Animate Data를 읽어서 저장함. 3. Object는 자기가 사용할 Mesh Data와 Animation Data를 멤버변수로 갖고 렌더링 할 때 사용함. 4. Animate(float time)을 통해서 각 Object별로 변환 행렬을 생성. 5. Update()에서 Animate()를 하고 얻은 결과물(행렬)을 hlsl로 보냄. 6. Hlsl에서 직접 CP들을 행렬변환하여 수정.     **[실제 함수 사용]**  **GetMeshData()**를 하면 해당 메쉬가 가진 모든 CP를 전부 볼 수 있음.  이걸 Vector로 저장.  **GetClusterData()**를 하면 어떤 뼈가 어떤 CP에게 영향을 얼마나 주는지 알 수 있음.  이걸 위에서 저장한 CP에게 추가.  **GetAnimationDataRec()**를 하면 몇 초에 어떤 뼈들이 얼마만큼 변환되어 있어야 하는지 알 수 있음.  이걸 Vector로 저장. **-> 이건 이미 AnimData class 만들어서 끝냈음.**  Bone Data는 뭘로 하지??? 사실 Bone Data는 필요없는 것이 아닐까??    **[파일 저장을 위한 프로토콜]**  파일 사이즈는 어떻게 알지? 맨 앞에 헤더를 달기.  헤더 내용은 다음과 같이.  MeshName(str)  MeshCount(int)  MeshControlPoint \* MeshCount 개수만큼(ControlPoint)    AnimationName(str)  TotalTime(float, millisec)  isLoopAnimation(bool)  NextAnimation(str)  이건 근데 동적이니까 모든 정보가 다 있어야 함.  예를 들어 시간은 몇 개가 있을거다, 그 시간엔 Bone이 몇 개 있을거다.  그럼 맨 처음에 시간 개수랑 각 시간별 Bone 개수를 저장하자.  nKeys, nBones  Bone 배열을 만들어서 idx를 받는것이 필요하지 않을까?    Struct Bone {  Int Idx; //나중에 Vector에서 idx로 순회할 때 넣어주기.  String name;  값  Bone\* parent    ;  } Importer 부분 파일 출력을 해냈으면 읽을 줄도 알아야 한다.  가장 이상적인 방법은 파일을 읽은 뒤, mesh와 animtaionData를 각각 만들고 Object에서 둘 다 관리하는 것.    Importer.Import("fileName", Object& obj)  식으로 만들고 해당 Object는 AnimationData와 MeshData를 둘 다 갖게 만든다.    AnimationMesh를 Object에 넣어서 관리하기 | | | |
| 작업 예정 내용 | | | |
| - 애니메이션 Key값 넣는 부분에서 배열 인덱스 문제가 생겼다.  급하게 이전 코드를 사용하려 해서 생긴 문제. 다시 컨버터에서 작성한 방식으로 적용하면 처리될 것. | | | |
| 기타 | | | |
|  | | | |