|  |  |  |  |
| --- | --- | --- | --- |
| MORPHOSIS 개발일지 12차 | | | |
| 기간 | 2019-05-15 ~ 2019-05-29 | 작성자 | 신재욱 |
| 작업 내용 | | | |
| 애니메이션도 중요한데 일단 그거 말고도 해야 되는 것들 다시 한 번 정리해보자.  - FBX SDK에서 UV 좌표 얻어오게 하기.  - Mesh 렌더링 됐던 코드 다시 살리기.  - 3ds Max에서 레벨 만들고 사용한 텍스쳐 종류들 받아오게 하기.(여러 텍스처를 쓰면 UV 좌표를 0~1, 1~2 이런 식으로 할까? 아니면 그냥 텍스처를 다른 걸 쓰는 메쉬끼리는 아예 떼어내서 다른 오브젝트로 만들까?)  - 유니티 엔진에서 예전에 배웠던 버텍스에 맞춰서 정렬하는 기능 써서 충돌 맵 만들고 점령지점이랑 캐릭터 스폰 포인트 만들고 파일 만들어서 추출하기.  - 캐릭터 충돌처리 부분 깔끔하게 고치기.  - 투사체 만드는 부분 살리기.  - 투사체에 컴포넌트 속성 넣기.  - UI 띄우기.  - 디버그를 위해서라도 알파벳과 숫자는 폰트로 띄울 수 있어야 함.  - 전체화면 하는 것도 해야 됨.  지난 이야기:  컨버터와 기타 등등 문제를 해결했고 이제 본 클라에서 값이 잘 들어가는지만 보면 되는 것인데!  이제 파일은 정상적으로 들어온다고 가정하자. 테스트 클라에서 잘 들어갔으니까 똑 같은 코드인데 잘 들어가겠지(안일). 현재 방식은 앞서 설명했던 대로 1. GenerateToWorldMatrix()에서 Lcl행렬을 만들고, 그걸 본 행렬의 Lcl 행렬에 넣은 다음에 그걸 가지고 MakeToWorldMatrix()를 한 뒤, 그 결과물과 Offset 행렬을 곱해서 최종행렬을 만들고 그걸 HLSL에 올려주는 방식이다. 이제는 GlobalTransform이 있으니까 굳이 로컬행렬을 만들고~ to월드행렬 만들고~ 할 필요가 없다. 글로벌 변환 행렬을 그대로 보간해주고 그걸 Offset행렬과 곱해주면 아마…… 기대하는 결과가 나올 것…… 안 나오면…… 큰 일 남…….    함 해보자!    응~ 어떻게 디버깅을 할 지 다시 계획을 짜보자.  1. 저 글로벌 변환 행렬이 내가 생각한 행렬이 맞는가?(Offset과 곱하면 되는 행렬이 맞는가)  2. 곱하는 방식에 문제는 없는가?(예를 들어 보간하는 과정이나 YZ축 문제 등)  3. 추가적으로 변환(예를 들어 전치행렬 등)을 해줘야 하는가?  앗 너무 행복해  아 진짜 FBX SDK 너무 싫다. 의미부터 다시 공부해야 할 듯. 근데 의미를 안 적어놓잖아. 한 번 다시 찾아보자.      LclTranslation.Get()으로 바로 받아올수 있구나 했는데 생각해보니까 언제 그 값이어야 하는지 알아야 해서 안 됨.    Global Transformation이랑 Local Transformaion이랑 뭐가 다른거지?    함수 설명에 Global Transform이 뭔지 설명이 있을 줄 아셨습니까? 유감!    Remarks에 Local Transform Matrix가 무엇인지는 나와있다. 부모의 글로벌 변환에 글로벌 변환을 곱한거라고?    ‘To evaluate these properties separately without taking pre/post rotation, pivots and offsets into consideration, please use GetNodeLocalTranslation(), GetNodeLocalRotation() and GetNodeLocalScaling().’????  피봇이나 오프셋이나 그런거 다 집어치우고 그냥 로컬 회전만 받을 수 있다고?  테스트를 하려면 많은 변화가 필요할 것 같다…  테스트 용도로 쓰고 있던 Project2 프로젝트에 fbxsdk.h를 추가하고 프로젝트 속성에서 참조 디렉토리에 해당 헤더의 위치를 추가해줬다. (이거 하는 법 몰라서 작년엔 샘플 프로젝트를 컨버터로 바꿨어야 했으므로…)    아니 포맷 해줬더니 이렇게?  원래 FBX SDK를 설치할 때 모든 파일을 다 받았었는데 컴퓨터를 포맷하고 샘플 프로젝트들만 깃으로 푸시해놨었기 때문에 생긴 문제인 것 같다.  그래서~ FBX SDK를 새로 받았는데~~ 얘네가 버전을 업데이트 하면서 프로젝트 파일을 없에고 CMake로 만들게 바꿔두었다~~    ;;;  분명 이전 버전을 남겨두지 않았을까???    감사합니다.    잘 생각해보자. 지금 필요한 것들.  Bone들(Offset 행렬 있어야 함), key들(시간, 관련 있는 Bone)(그래야 그 Bone이랑 시간 가지고 그 시간에 그 Bone이 어떻게 움직이고 있는지 알 수 있음, 아마 그걸로 LocalTransform이든 GlobalTransform이든 나올 것)  저게 있으면 애니메이션 데이터는 끝이다. 시간에 따라서 행렬 저장해주면 될 듯.  파일의 양식은 아마 Bone의 기본 Offset 행렬들과 키들의 정보(키의 시간, 본의 idx와 각 본의 그 시간대별 변환 행렬들).    행렬 얻어오는 함수들은 FbxNode에 있음.  FbxNode가 Bone인지는 Node의 속성을 확인하면 됨.    아마 이런거 써서 하면 될 것 같음.    아 뭐야  분명 디렉토리까지 설정해줬는데 왜 정의를 못 찾는 것일까요?    와!  <https://blessingdev.wordpress.com/2017/09/26/visual-studio%EC%97%90-%EC%99%B8%EB%B6%80-%EB%9D%BC%EC%9D%B4%EB%B8%8C%EB%9F%AC%EB%A6%AC-%EC%B6%94%EA%B0%80%ED%95%98%EA%B8%B0/>  의 글을 보고 참고하여 외부 라이브러리 추가를 해보았습니다.    암 생각 없이 .lib 확장자 파일을 전부 추가해줬더니    룰루~  -md, -mt를 빼고 기본만 넣어주었습니다.    그나마 정상이 되었군요.    저 주소로 가서 Common.cxx를 프로젝트에 추가해주니까 해결되긴 했는데 저걸 그냥 추가 안 하고 쓸 수는 없나… 일단 그게 중요한 내용이 아니므로 패스.    아.  프로젝트를 새로 만들고~  VC++ 디렉터리-포함 디렉터리, 라이브러리 디렉터리  C/C++ 추가 포함 디렉터리  링커-입력-추가 종속성 .lib  빌드 되서 나오는 폴더에 .dll 추가  해서 문제 해결.    드디어 여기까지 옴.    아아 정말 완벽해. OK 계획대로 되고 있어.  Bone들을 순회하면서 글로벌 행렬 얻어보기    아니 생각보다 훨씬 편하고 좋은데?  개발일지 10차 문서에서 내가 했던 방식을 가져와보자.  역회전 = LclRotation, 역이동 = LclTranslation  < 순 서 도 >  Lcl Lcl Lcl  이동 회전 이동 회전 이동 회전 역회전 역이동 역회전 역이동 역회전 역이동  (m\_toParent)^-1 m\_toParent  Parent’s DressposeInv Parent’s toWorld  My DressposeInv My toWorld    **끝나지 않는 삽질.**  Lcl Rotation과 Lcl Translation은 부모 좌표계에서 내 좌표계로 바꾸는 방법.  T를 먼저 적용하고, R을 적용하면 내 좌표축이 됨. (저기 나오는 모든 값은 Y축이랑 Z축을 바꿔줘야 함. 저건 맥스 좌표계니까)  부모 변환(ToParent)은 내 좌표계에 있는 점을 부모의 좌표계로 옮기는 변환.  반복하면 뼈 공간의 점이 월드 공간으로 변환됨.  필요한 변환들과 의도를 적어보자.  부모변환행렬: 뼈 공간의 점을 월드 공간으로 변환.  로컬변환행렬: 시간에 따라 뼈가 회전하고 이동한 값만큼 변환.  부모변환행렬의 역행렬: 월드 공간의 점을 뼈 공간으로 변환.  먼저 T포즈의 점을 ‘부모변환행렬의 역행렬’을 곱하면 그 점이 속한 뼈 공간으로 변환된다.  그 상태에서 로컬변환행렬을 곱해서 그 점을 변환하면 걔가 뼈가 회전하고 이동한만큼 회전하고 이동함.  그리고 다시 ‘부모변환행렬’을 곱해서 현재 뼈 공간에서 변환된 점을 다시 월드 공간으로 옮겨준다.  쉽게 생각하면 저 멀리 있는 점을 회전시킬 기준점이 원점이 되도록 끌고와서 회전시킨 다음에 다시 보내주는 것인데, 다시 보내줄 때 고려해야 될 사안이 로컬변환행렬의 내용이다.  처음에 T 포즈의 점을 뼈 공간으로 끌고 올 때는 Offset 행렬만 해주면 되니까 전혀 문제 없는데 부모 뼈들이 변환된 값만큼 적용이 되어야 함. 그걸 위해서 처음에 뼈 공간으로 가져올 때는 Offset 변환 행렬을 곱해주고 다시 월드 공간으로 변환될 때는 로컬변환행렬과 부모변환행렬을 곱해주면 됨. 로컬변환행렬이 시간에 따라 변하기 때문에 시간마다 Bone 마다 로컬변환행렬과 부모변환행렬을 곱한  아!!! 어쨌든 실시간으로 계산해야 하는거잖아!!! 뭘 계산해줘야 하죠?  각 Bone마다 ToRoot변환행렬을 구해야 함.  무조건 부모가 자식보다 인덱스가 작을 수밖에 없다. 그니까 인덱스 순서대로 ToRoot를 구하면 전혀 문제가 되지 않는다. ToRoot를 구하려면 부모의 ToRoot도 알아야 하니까. 부모부터 먼저 구해야 한다.  회전 = LclRotation, 이동 = LclTranslation  < 수 정 된 순 서 도 >  Lcl Lcl Lcl  이동 회전 이동 회전 이동 회전 역회전 역이동 역회전 역이동 역회전 역이동  (toParent)^-1 toParent  Parent’s Offset Matrix Parent’s toRoot  Offset Matrix toRoot  아이고. Lcl이 단위행렬이라는 가정 하에 toRoot를 구해보고 GlobalTransform이 toRoot의 역행렬인지 함 보자.    아니 근데 진짜 쉽긴 쉬운데?  함수 포인터를 한 번 써봤는데 넘 편하다.    값이 맞게 들어간다고는 안 했음.  혹시 몰라! 곱하면 단위행렬 될지도 몰라!  Lcl들은 부모에서 내 좌표계로 만드는 값 -> toParent의 반대. 월드에서 뼈 공간으로 보냄.  이동 회전 순서로 진행. 이것의 역행렬이 toParent.  toParent는 뼈에서 월드로 보냄.  지금 알았는데 여러 사람들의 애니메이션 코드를 참고할 때, 다 같은 부분이 보였던건 실제로 다 같은 코드였기 때문이었다… 애니메이션 셋, 애니메이션 컨트롤러 등은 이용희 교수님의 예제 코드가 있었다.  저기서 구하는 GlobalTransform은 아마 맨 앞의 My Dresspose의 역할을 해주지 않을까?  LocalTransform이 Lcl일 것이고, 그럼 시간별로 저 뒤의 toWorld 값을 구해서 그걸 키프레임마다 저장해두면? 그리고 그걸 보간해서 사용한다면? 그렇게 하면 행렬들만 저장해도 되지 않을까?  LclRotation과 LclTranslation이 부모 좌표계로 가는 변환이었으니까 그걸 바탕으로 구상을 짜보자.  Bone마다 일단 자신의 toParent를 만들어두고 부모의 toWorld를 받아서 내 Lcl이랑 어캐 잘 하면 내 toWorld(이용희 교수님은 toRoot 용어를 쓰심)를 구할 수 있을 것. 그걸 시간마다 구해서 행렬로 저장해보자.   1. 모든 Bone을 돌면서 LclRotation과 LclTranslation으로 toParent를 만듦. 2. 모든 Bone을 돌면서 GlobalTransform을 구함. 3. 모든 KeyFrame의 KeyTime에 대해서 그 KeyFrame에서 관리하는 Bone들의 | | | |
| 작업 예정 내용 | | | |
|  | | | |
| 기타 | | | |
|  | | | |