|  |  |  |  |
| --- | --- | --- | --- |
| MORPHOSIS 개발일지 11차 | | | |
| 기간 | 2019-05-09 ~ 2019-05-15 | 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.  TestScene을 새로 만들어줬다. 애니메이션 꼴도 보기 싫어서.  루트 시그니처부터 다시 만들자. 했던거니까 복습하는 느낌으로 금방 금방 만듦.  /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  서술자 테이블은 동적으로 관리해야 하는 정보들을 담당할 것.  예를 들면 오브젝트들의 정보나, 쉐이더에서 쓸 텍스처 정보 등.  여기서 헷갈릴 수 있음.  Root Parameter는 세 가지 종류로 구분할 수 있음.  서술자 테이블(Root Descriptor Table), 루트 서술자(Root Descriptor), 루트 상수(Root Constant)  서술자 테이블은 서술자들의 배열이 있는 곳의 정보를 인자로 넘겨준다고 생각하면 편함.  서술자를 통해 버퍼의 데이터를 해석하려면 먼저 인자를 읽고, 인자에 적혀있는 서술자들의 배열 주소로 간 뒤에  거기서 해당하는 서술자를 찾고, 그 서술자로 버퍼의 데이터를 해석하면 됨.  중간에 찾아가는 과정이 길지만 한정된 루트 파라미터 공간을 최대한 많이 쓰려면 이런 방식을 써야 함.  루트 서술자는 서술자를 인자로 넘겨주는 것.  서술자를 통해 버퍼의 데이터를 해석하려면 인자로 받은 서술자를 통해 해석하면 된다.  아래서 쓰게 될 예정.  루트 상수는 그냥 값을 인자로 넘겨주는 것.  그거 쓰면 된다. 진짜 간단하고 속도도 빠르지만 한정된 공간에서 주소값이 아닌 실제 값으로 채워버리면  사용할 수 있는 개수가 엄청나게 줄어들기 때문에 많이 쓸 수는 없다.  지금은 서술자 테이블 두 개와 루트 서술자 두 개를 사용할 예정이고,  서술자 테이블을 사용하려면 D3D12\_ROOT\_DESCRIPTOR\_TABLE 구조체를 만들어서 안에 내용 채워서 줘야 함.  그니까 먼저 그 두 개를 만드는 과정을 아래 보여줄 것.  지금 서술자 테이블로는 오브젝트랑 텍스처 두 개 관리해야 하니까 두 개 만들어주자.  카메라 처리랑 관련된 행렬들 올려줘야 함.      뭐뭐 띄워야 하지?  캐릭터, 무기, 투사체, 레벨, UI, 디버그용 기즈모랑 충돌박스  \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/      아 진짜 뭐가 다른건지 설명도 따로 없고 이게 뭐야. 의미를 어디 적어뒀으면 좋겠는데 잘 모르겠다.  다시 애니메이션으로 돌아옴.  애니메이션 행렬을 어떻게 해야? 러프하게 일단 적어보고 정리해보자.  본의 DressposeInv(앞으로는 이걸 Offset이라고 하자)를 HLSL에 올려주고  오브젝트 단에서 지금 시간가지고 Lcl변환만 올려주면  HLSL에서 DressposeInv \* Lcl \* toWorld 만들어서 곱해주면 되는데  1. 애니메이션 처리를 위한 쉐이더는 버텍스마다 호출되고  2. toWorld를 구하려면 계층구조 정보도 있어야 하고(Lcl로 인해 toWorld를 고쳐야 하니까)  버텍스마다 호출되는거 말고 오브젝트마다 호출될 수 있게만 하면  쉐이더 리소스 뷰를 사용해서 한다면? 엄청 빠르게 구할 수 있을텐데 그것은 나중에 더 강해진 신재욱이 할 것이다.  지금은 어쩔 수 없이 CPU 단에서 Offset이랑 toWorld랑 다 구해야 함. 이거 어쩔 수 없는거 같음.  굳이 그럼 HLSL에 fTime 이런거 안 올려줘도 되겠네. **HLSL에 올리는건 Offset이랑 toWorld랑 구해서 나온 최종행렬 뿐이다.** 흠~ 그렇게 생각하면 마음이 좀 편해지는 것입니다.  씻으면서 애니메이션 생각하다가 아! 하고 뭔가 생각나서 막 해봤지만 이미 해봤었던 방법이었음. 시무룩. 일단은 FbxNode의 EvaluateGlobalTransform(FbxTIme) 함수를 이용해서 글로벌 변환 행렬을 얻어보겠어요.  예상되는 사용법은 저걸로 얻은 특정 시간대 본들의 변환 행렬을 오프셋 행렬(한창 DressposeInv라고 부르던 것)과 곱해서 바로 최종 변환 행렬을 얻어내는 것.  생길 것 같은 문제는 내가 임포트할 때 YZ축을 바꿔주니까 그걸 아마 되돌리고 해야하지 않을까 라는 것.  EvaluateGlobalTransform() 함수는 FbxTime 자료형을 인자로 받는데 그걸 구하려면 FbxAnimCurve 자료형을 구해야 하고~ 그걸 구하려면 FbxNode에서 FbxAnimLayer 자료형을 인자로 줘서 구해야 함.  근데 FbxAnimCurve를 구하려면 GetCurve() 를 해야 하는데 FbxNode에서 바로 그걸 하는 법을 모르겠음. 그래서 기존에 LclTranslation, LclRotation 값을 구할 때 FbxAnimCurve를 구하던게 생각나서 쓰레기같지만 일단 급하니까 그걸 써보기로 했다.    <쓰레기의 본망>  일단 테스트를 위해서 한거니까 어쩔 수 없는 것임.  익스포터에서 저 정보를 쓰게 하자.    과거의 멍청했던 내가 만든 자료형이 이제와서 내 발목을 잡을 줄이야.  하여간 지금은 중요한게 아니니까 XMMATRIX 정보를 넣을 수 있게 하자.  FbxAMatrix 자료형은 XMFLOAT4X4 자료형과 동일할까???  아 테스트할 때는 내 알 바가 아니다. 일단 익스포트 했음.  본 클라이언트 내용들을 전부 바꾸고 있었는데  나는 GenerateToWorldMatrix() 함수에서 GetInterpolatedLocalMatrix()를 호출하여 Local 행렬을 구하고 그걸 n번째 본의 Local 행렬에 넣는다.  그걸가지고 toWorld를 만들고~ 그걸 가지고 Offset이랑 곱하고~  걍 GetInterpolatedGlobalMatrix()로 글로벌 변환 행렬의 보간을 구하고 그걸 Offset이랑 곱하는건?  그대로 본 클라에서 읽어봤는데 이상한 정보가 들어옴  이것도 테스트 해봐야겠다. 일단 강의 들어가기.  왜 이상한 값이 들어오지?????    ???????????????????    아 진짜  저걸 전부 다시 처리해줬더니 잘 들어감. 아….. | | | |
| 작업 예정 내용 | | | |
|  | | | |
| 기타 | | | |
|  | | | |