|  |  |  |  |
| --- | --- | --- | --- |
| MORPHOSIS 개발일지 9차 | | | |
| 기간 | 2018-03-21 ~ 2019-04-02 | 작성자 | 신재욱 |
| 작업 내용 | | | |
| 요약본 애니메이션 계산 부분 작성. 상세내용 기존의 애니메이션 최종 변환 행렬을 계산할 때, 각 key class의 std::vector<AnimationBone> m\_bones 변수에 인자로 받은 boneIdx를 기준으로 접근하였었으나 **이 경우, 각 key가 가진 m\_bones 내에서 boneIdx로 찾기 때문에 의도한 결과가 나오지 않는 문제**가 있었다.  이를 해결하기 위해 AnimationData class 내부에 **bone들을 총괄하는 std::vector<AnimationBone> m\_bones 를 선언**하고 인자로 받은 boneIdx를 사용하여 **총괄하는 m\_bones에서 Bone을 찾고 그 내용을 가지고 계산하게 변경**하였다.  bones을 사용하여 AnimationData에서 만들 때, **인자로 받은 bones 배열을 수정하고 있던 문제**가 있었다. AnimationData의 멤버변수인 m\_bones를 인자로 받은 bones와 같은 size로 만들고(m\_bones.resize(bones.size());), 다시 순회하면서 parent 주소를 대입해주었다.(if(bones[i].parentIdx != -1) m\_bones[i].parent = &m\_bones[bones[i].parentIdx];)  **keys를 저장하는 부분을 수정하였다.** 인자로 들어오는 keys는 각 key마다 bone이 몇 개 있는지, 그 bone들의 idx는 어떻게 되는지, key들의 keyTime은 어떻게 되는지, 각 bone들의 시간에 따른 translation, rotation 값이 어떻게 되는지 갖고 있다.  1. m\_keys의 size를 keys와 같게 해준다.  2. 각 key마다 keyTime을 저장한다.  3. 각 key마다 갖고 있는 bone의 개수를 저장한다.  4. 각 bone마다 이미 위에서 만들어둔 멤버변수 m\_bones의 내용을 참조한다.  5. boneReferenceInfo에 어떤 key가 그 bone의 정보를 갖고 있는지 insert한다.  AnimationData class의 Init()에선 std::vector<**AnimationBone**> m\_bones와 std::vector<**AnimationKey**> m\_keys를 만들어 주는 것 외에도 std::vector<**BoneReferenceInfo**> m\_boneReferenceInfo 또한 만들어준다.  여기서 **BoneReferenceInfo** struct는 **어떤 Bone에 대한 정보를 어떤 Key들이 갖고 있는지 미리 저장**해두어 **빠르게 time 비교를 할 수 있도록 도와주는 역할**을 한다.  GenerateToWorldMatrix(float time)에서 더 이상 m\_keys.front().m\_pBones로 관리할 필요가 없어졌으므로 해당 함수 내용을 수정하였다.  기존의 GetInterpolatedLocalMatrix()에서 사용하던 방식을 아예 뒤엎어야 했다. **keyIdx는 현재 Key와 다음 Key가 보간을 위해 필요하고 이는 boneReferenceInfo에서 찾아야 한다.** 다음 세 가지 경우가 나올 수 있다.  1. m\_boneReferenceInfo.idxOfKeySet의 원소가 하나밖에 없는 경우  2. 현재 key가 마지막 key일 경우  3. 현재 key와 다음 key를 둘 다 찾은 경우  1번의 경우, 둘 다 반환하고 그대로 계산한다. 그럼 같은 내용으로 보간이 이뤄지므로 사실상 같은 값이 반환된다. 2번의 경우도 마찬가지. 3번의 경우는 기대했던 대로 보간하면 된다. **결과적으로 함수의 내용은 크게 달라지지 않는다.** | | | |
| 작업 예정 내용 | | | |
|  | | | |
| 기타 | | | |
|  | | | |