**게임알고리즘심화과정**

**과목명 : 게임 일반 프로그래밍**

**능력단위 : 게임 그래픽 프로그래밍**

**제출일자 : 2021년 11월 05일**

**포트폴리오 : Skinning/Bind Pose 애니메이션, 캐릭터의 모션 변경 및 이동 처리 구현**

**작성자 : 최지원**

**(제출내역)**

1. **Skinning/Bind Pose 애니메이션이 구현된 3D 엔진 프로젝트**
2. **기능이 구현된 3D게임 엔진 체계도 및 세부 문서**

**<목차>**

1. **프로젝트의 소개 및 개요**
   1. 프로젝트 소개
   2. Skinning
   3. Bind Pose
2. **프로젝트의 설계 및 세부적 기능 단위로 다이어그램 기술**
   1. UML 다이어그램
      1. 클래스 다이어그램 (첨부)
      2. 시퀀스 다이어그램 (첨부)
3. **프로젝트의 구현된 주요 기술 단위로 분석 및 설계 기술**
   1. 주요 변수 설계
   2. Skinning/Bind Pose 애니메이션
      1. 함수 설계
   3. 캐릭터의 모션 변경 및 이동 처리
      1. 함수 설계
4. **최종 결과 및 추가된 내용**
   1. 결과물 화면 (첨부)
   2. 범용성/ 유연성/ 확장성/ 간결성 고려하여 추가된 내용
5. **프로젝트의 소개 및 개요**
6. **프로젝트 소개**

DirectX 11 윈도우 기반 3D 그래픽 프로그래밍으로 FBX형식의 파일을 로드해서 오브젝트를 렌더링할 수 있다.

데이터가 많지만 계산이 적은 샘플링 방식으로 애니메이션을 렌더링 할 수 있다.

캐릭터의 이동 처리를 구현하여, 이동할 시에 걷기, 달리기 등 원하는 구간의 애니메이션을 출력하거나, 특정 키를 누르면 해당 오브젝트의 모든 애니메이션 구간을 재생하도록 구현하였다.

일반적으로 게임에서는 자체적인 3D 오브젝트 Exporter 플러그인을 사용한다. 개발자가 엔진에 필요한 정보만을 추출해서 원하는 포맷 형식으로 출력하고 빠르게 로드하기 위함이다. FBX파일을 Export, Import하기 위해서는 3DS MAX SDK를 사용해야한다.

본 프로젝트에서는 FBX파일을 직접 Import해서 애니메이션을 구현하기 위해 3DS MAX SDK를 사용했다. 3DS MAX 는 Up Axis가 Z축인 반면, 본 다이렉트X11은 Y축이다. 또한 변환 행렬 계산방식도 다르다. (DirectX는 행기준) 행렬의 저장 방식에 따라 계산 방식이 달라지기 때문에 본 프로젝트에서는 FbxAxisSystem을 MayaZup를 사용해 Up Axis가 다르던 간에 똑같이 출력되는 효과를 얻을 수 있다.

1. **Skinning**

애니메이션은 오브젝트 애니메이션과 캐릭터 애니메이션으로 구분할 수 있다. 오브젝트 애니메이션은 단일 메쉬에 포함된 정점 (버텍스 버퍼)에 동일한 행렬이 적용된다. 캐릭터 애니메이션을 이와 똑같이 적용하면 근육이 떨림이나, 관절의 움직임에 디테일한 애니메이션을 구현하지 못한다.

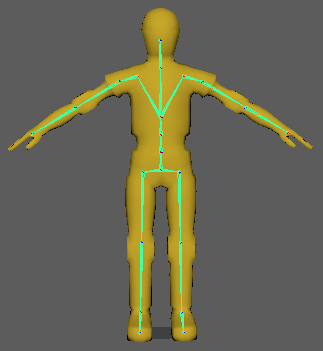
따라서 캐릭터 애니메이션은 가중치 (Weight)를 사용하여 여러 행렬들을 가중치를 따로 주어서 곱해서 합산한 것으로 최종 애니메이션 버텍스 좌표로 쓴다. 즉, 가중치는 정점당 어떤 행렬의 영향을 얼만큼 받느냐를 따져준다. 이와 같은 방식을 **Skinning**이라고 한다.

스키닝에는 블랜딩, 논블랜딩 방식으로 나뉜다. 논블랜딩은 말그대로 혼합하지 않는다는 뜻으로, 하나의 뼈 안에서 대부분의 스킨 정점들은 논블랜딩이다. 가충치를 1로 둔다. 반면에 블랜딩은 팔을 구부리면 어께 근육도 움직이듯이 각각의 움직임에 따라 영향도가 다른 것이다.

텍스트, 클립아트이(가) 표시된 사진

자동 생성된 설명모델링에서는 본과 연결하는 가중치를 입력하여 본의 움직임에 따라가도록 버텍스를 본에 연결하는 것을 리깅이라고 한다.

source : AnyPortrait

1. **BindPose**

source : Autodesk

BindPose는 T-Pose와 유사한 의미로 초기 뼈가 스킨을 묶을 때 있는 위치이다. 초기 위치값 행렬인 BindPose는 T-Pose로 돌아가는 행렬이다. 본 또한 각각 자신의 로컬 좌표계 기준을 갖고 있기 때문에 본 좌표계로 바꿔줘야한다. 그렇게 바꿔준 뼈의 위치 역행렬과 애니메이션 행렬을 곱하면 초기 애니메이션, Start Pose를 만들 수 있다.

본 프로젝트에서는 하나의 애니메이션으로 프레임 마다 다른 애니메이션이 있다. 반면에 캐릭터 오브젝트 파일 따로 동작별로 애니메이션 파일 따로 Skin 파일을 공유하며 사용하는 경우 BindPose 애니메이션이라고도 한다.

1. **프로젝트의 설계 및 세부적 기능 단위로 다이어그램 기술**
   1. UML 다이어그램
      1. 텍스트이(가) 표시된 사진

         자동 생성된 설명클래스 다이어그램 (첨부)

그림 2 클래스 다이어그램 (Sample.cpp)

|  |  |
| --- | --- |
| 자체 엔진EngineCoreD라이브러리를 사용하는 클래스 | |
| Sample 클래스 | EngineCoreD 라이브러리를 사용해서 wWinMain 함수로 윈도우 창을 생성, 다이렉트X 환경을 구성하여 FBX 로드, 캐릭터의 애니메이션, 이동 등 각종 객체를 관리하는 클래스 |
| KLight 클래스 | 라이트 관련 클래스 |
| KMinimap클래스 | 라이트 디버깅 미니맵 클래스 |

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 3 클래스 다이어그램 (EngineCoreD.Lib)

|  |  |
| --- | --- |
| 구현된 기능의 주요 클래스 EngineCoreD 라이브러리 | |
| KFbxObj 클래스 | FBX 파일 해석 및 로드, 애니메이션 |
| KModel 클래스 | 버퍼 생성, 쉐이더 관리, 오브젝트 렌더 클래스 |
| KMesh 클래스 | 모델을 상속받아 매쉬별로 관리하는 클래스 |
| KWindow클래스 | 윈도우 관련 클래스 |
| KDevice클래스 | DirectX11 관련 클래스 |
| KInput 클래스 | 싱글톤으로 구현된 사용자 Input 입력 클래스 |

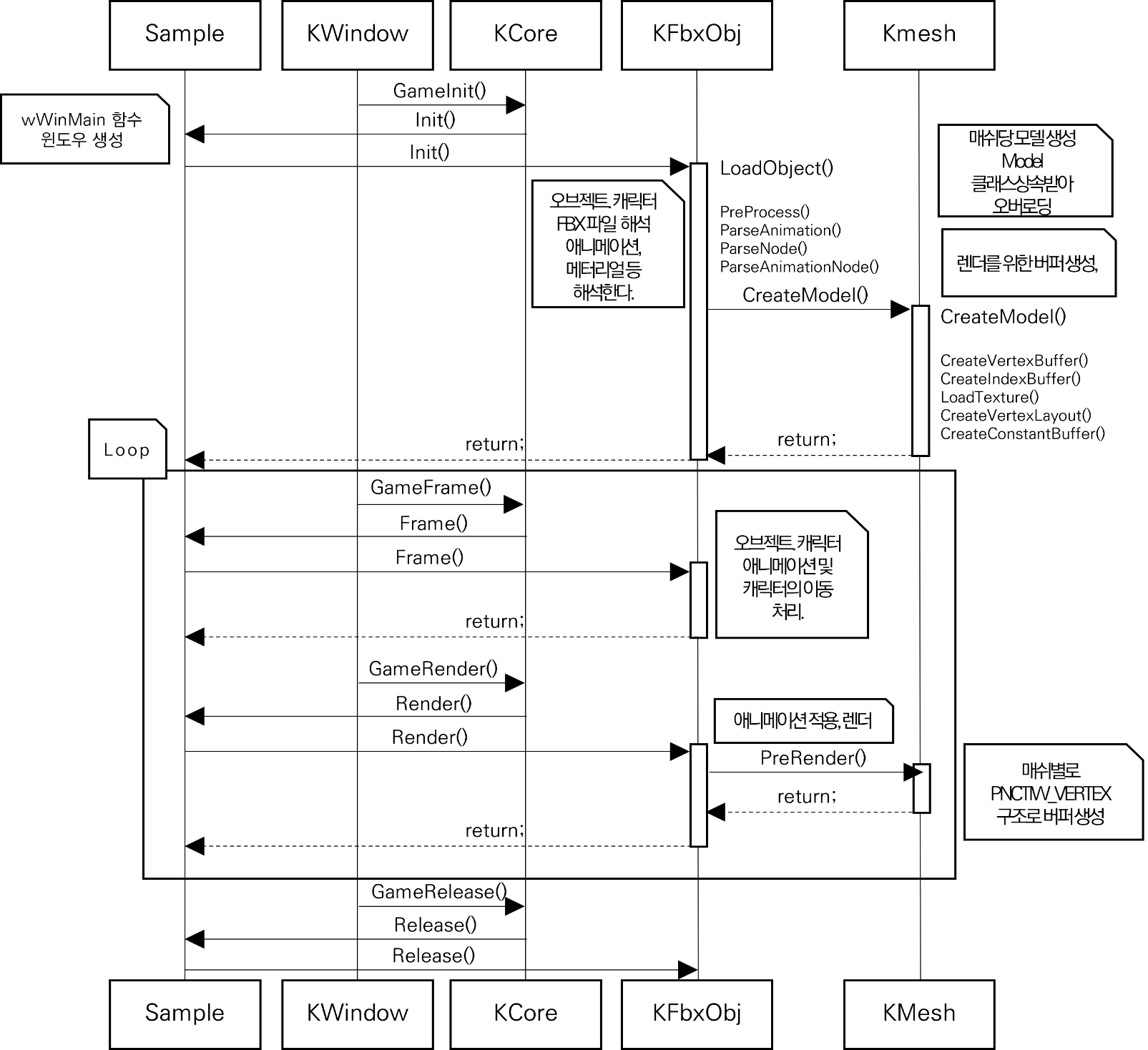
* + 1. 시퀀스 다이어그램 (첨부)

그림 4 시퀀스 다이어그램

|  |  |
| --- | --- |
| 1 | 기본적인 윈도우 생성 카메라, 맵 생성 및 초기화 |
| 2 | 오브젝트와 캐릭터 FBX 파일(버텍스, 애니메이션 정보) 노드 해석, 버퍼생성 |
| 3 | 키입력에 따라 LookVector 사용해 캐릭터 이동 처리 및 상황에 맞는 애니메이션 |
| 4 | 행렬 정보 갱신, 쉐이더에 전달해 스키닝된 캐릭터 렌더링 |

1. **프로젝트의 구현된 주요 기술 단위로 분석 및 설계 기술**
   1. 주요 변수 설계

|  |  |  |  |
| --- | --- | --- | --- |
| **클래스**  **/구조체** | **타입** | **인스턴스**  **/변수명** | **내용** |
| Sample | KFbxObj | m\_FbxObjA | 캐릭터 타입 FBX 파일의 객체 |
| m\_FbxObjB | 오브젝트 타입 FBX 파일의 객체 |
| KFbxObj | FbxManager\* | m\_pFbxManager | 3DS SDK를 사용해 FBX Importer설정 및 환경 구축 |
| FbxImporter\* | m\_pFbxImporter |
| FbxScene\* | m\_pFbxScene |
| bool | m\_bAnimPlay | 애니메이션 진행 상태 bool 값 |
| float | m\_fElpaseTime | 애니메이션 경과 시간 |
| int | m\_iAnimIndex | 프레임 계산 변수 |
| KVector3 | m\_vLook | 이동처리에서 쓰일 방향 벡터 |
| vector<FbxNode\*> | m\_pFbxNodeList | 카메라 라이트 제외한  모든 Fbx 정보 노드 |
| vector<KMesh\*> | m\_pMeshList | 뼈, 지오메트릭 등 매쉬 단위로 저장한 배열 |
| CB\_DATA | m\_cbData | 쉐이더에 넘길 행렬 상수버퍼구조체 |
| KMesh | enum | OBJECTCLASSTYPE | 매쉬 타입 열거형 상수 |
| vector<FbxNode\*> | m\_pFbxNodeList | 매쉬 노드 리스트 배열 |
| vector<KMatrix> | m\_matBindPoseList | 뼈의 위치 역행렬을 담는 배열 |
| vector<KMesh\*> | m\_pMeshLinkList | 매쉬 연결 배열 |
| KAnimMatrix | m\_matAnimMatrix | 애니메이션 행렬을 담은 구조체 변수 |
| ID3D11Buffer\* | m\_pAnimCB | 애니메이션 상수 버퍼 |
| ID3D11Buffer\* | m\_pIWVertrexBuffer | PNCT에서 IW까지 포함한 버텍스 버퍼 |
| vector<PNCTIW\_VERTEX> | m\_iwList | 정렬된 인덱스 4개씩 쉐이더에 넘기는 배열 |
| FbxNode\* | m\_pFbxNode | 매쉬의 Fbx노드 |
| KMatrix | m\_matWorld | 매쉬의 월드행렬 |
| vector<KMatrix> | m\_AnimationTrack | GetAnimationEvaluator 로 뽑아낸 애니메이션 트랙 행렬 배열 |
| vector<KMesh\*> | m\_pSubMesh | 매쉬의 하위 매쉬 배열 |
| KWeight | vector<int> | m\_IndexList; | 가중치 구조체로 인덱스와 가중치를 저장한다. |
| vector<float> | m\_WeightList; |
| KSkinData | vector<FbxNode\*> | m\_MatrixList | 스킨 데이터 구조체 배열 정렬해서 저장 |
| vector<KWeight> | m\_VertexList |
| PNCTIW\_VERTEX | float | index[4] | PNCT구조에서 인덱스 |
| float | weight[4] | PNCT 구조에서 가중치 |
| KAnimMatrix | KMatrix | matAnimation[255] | 애니메이션 행열 구조체 |

B. Skinning/Bind Pose 애니메이션

* + 1. 함수 설계

|  |  |  |
| --- | --- | --- |
| **클래스**  **/구조체** | **함수** | **내용** |
| FbxManager | Create() | Static 함수, SDK 관리자 객체를 생성해 이를 사용해서 사용하여 씬 객체 만들기 그러한 씬은 FbxScene 객체는 mesh, light, animation 등과 같은 다양한 요소를 포함할 수 있다. MayaZUp 축으로 설정했다. Up Axis가 다르던 간에 똑같이 출력되는 효과를 얻을 수 있다. |
| KFbxObj | LoadObject() | 3DS MAX SDK를 사용해 FBX를 Import할 수 있는 환경을 만들고 FBX 노드를 해석해 파일이름, 쉐이더를 넘겨받아 모델을 생성한다. |
| PreProcess() | 재귀 함수로써 전체 FBX 노드트리 순환하면서 노드를 멤버변수 m\_pFbxNodeList에 저장한다. |
| ParseAnimation() | FillAnimStackNameArray()를 사용해서 FBX가 갖고있는 애니메이션 정보 만큼 반복해서 애니메이션을 해석한다. |
| ParseAnimStack() | FBX 씬에서 애니메이션 정보를 불러와 읽어진 info가 있을 경우에 로컬타임으로 시작시간 끝나는 시간으로 환산해서 시간으로 분리하게 한다. |
| ParseAnimationNode() | 애니메이션에 대한 데이터를 뽑아주는 GetAnimationEvaluator()를 사용하여 추출한다. 그리고 매쉬별로 시간 마다의 행렬을 저장해 애니메이션 트랙을 만든다. |
| ParseNode() | 재귀함수로 카메라와 라이트를 제외하고 노드 해석 오브젝트 상속구조를 파악해 전체 다 매쉬로 MeshList에 넣어준다. |
| ParseTransform() | 지오매트리 행렬 제외하고 로컬행렬 \* 부모행렬을 곱해서 matWorld행렬이란 이름으로 리턴한다. 애니메이션을 고려해서 지오매트릭 행렬은 나중에 곱한다. |
| ParseMesh() | ParseMeshSkinning() 함수로스키닝 오브젝트 여부를 따져서 얻어낸 가중치 리스트와 인덱스리스트를 정렬한다. make\_pair 함수를 써서 짝을 맞춰서 list에 넣는다. 정렬해서 영향이 적은 인덱스를 제외하고 인덱스 가중치를 4개만 iwVertex에 넣는다. 비 스키닝 오브젝트도 스키닝화한다. 0번의 인덱스는 자기 자신의 월드행렬이 곱해서 오브젝트 애니메이션을 가능하게 한다. 기하행렬을 얻어서 곱해 리스트에 추가한다. |
| Compare() | 짝을 맞춘 리스트를 정렬하는 함수이다. 정렬하는 이유는 가장 영향이 적은 가중치를 제외하고 4개만을 저장하기 위해서다. |
| ParseMeshSkinning() | 불형 함수로 스키닝 오브젝트 여부를 따져준다. 모든 매쉬가 등록된 후에 스킨타입인 것 만을 받아와 디포머를 스킨으로 캐스팅한다. 드레스포즈(TPose)로 돌아가는 초기 위치값 행렬을 만들어 역행렬을 matBindPoseList에 저장한다. 또한 정점리스트와 가중치리스트도 저장한다. |
| GetFindIndex() | 값에 맞는 인덱스로 바꿔줘서 m\_pMeshLinkList에 저장한다. |
| ConvertMatrix() | 일반 행렬으로 복사 |
| ConvertAMatrix() | 아핀 변환으로 복사 |
| DxConvertMatrix | 열기준 행렬을 행기준 행열로 바꿔줌, 다이렉트 기준에 맞는 행렬로 변환한다. |
| Render() | 렌더링하는 함수이다. 서브 메터리얼 존재 유무를 따진다. 그리고 본타입과 같은 렌더 대상이 아니면 제외한다. matPose 뼈의 위치의 역행렬과 애니메이션 행열을 곱해 초기 애니메이션 스타트 포즈를 만든다. 최종 월드행렬과 뷰행렬 투영행렬을 설정한다. |
| KMesh | CreateVertexLayout() | PNCT, IW 멀티스트림 방식으로 버텍스레이아웃를 생성한다. |
| CreateConstantBuffer() | 애니메이션 행렬을 쉐이더에 넘기기 위해 상수버퍼를 생성한다.. |
| CreateVertexBuffer() | PNCT를 만들고 IW 정보가 있는 m\_iwList를 사용해서 버텍스 버퍼를 생성한다. |
| PreRender() | KModel 상속받은 KMesh는PreRender를 오버라이딩한다. 생성한 버퍼들을 넘겨준다. UpdateSubresource로 쉐이더, 데이터와 리소스 버퍼 저장을 한다. 멀티스트림 방식으로 인덱스와 버텍스 버퍼를 넘겨준다. |
| 쉐이더 | Character  Shader.hlsl | 렌더에서 받은 버퍼 값을 쉐이더에서 사용한다. PNCTIW로 VS에서는 로컬좌표에 뼈 행렬을 곱하고 가중치를 곱해 애니메이션 좌표를 만든다. 그 좌표를 사용해 월드 행렬, 뷰행렬, 투영행렬을 만든다. PS에서는 샘플러로 텍스쳐 적용한다. |

* 1. 캐릭터의 모션 변경 및 이동 처리
     1. 함수 설계

|  |  |  |
| --- | --- | --- |
| **클래스**  **/구조체** | **함수** | **내용** |
| Sample | Frame() | KInput 객체를 사용해 g\_Input 인스턴스로 사용자의 입력을 받아서 애니메이션 트리거 역할을 한다. Up Down키는 전진 후진, Left Right 키는 회전을 주었다. 회전하면 캐릭터의 LookVector, 월드행렬의 31,32,33 성분을 이용해 저장하였다. 그리고 오브젝트 균일한 이동을 위하여 벡터의 정규화를 한다.  캐릭터의 월드행렬을 사용해 이동 처리를 구현했다. 이동처리를 할 때 앞으로 전진하면 뛰는 모션, 뒤로 후진하면 걷는 애니메이션이 된다. |
| KFbxObj | Frame() | Sample에서 F4를 누르면 전체 오브젝트 샘플링 방식의 캐릭터 애니메이션을 실행한다. 만약 전체 애니메이션을 실행하고 있으면 캐릭터 이동을 방지하고, 다시 F4를 누를때까지 반복한다. |

1. **최종 결과 및 추가된 내용**
2. 결과물 화면 (첨부)

텍스트이(가) 표시된 사진

자동 생성된 설명

그림 6 F4 입력 스키닝 캐릭터, 비스키닝 오브젝트 전체 애니메이션

그림 5 방향키 입력 , 캐릭터 이동 애니메이션

1. 범용성/ 유연성/ 확장성/ 간결성 고려하여 추가된 내용

- KFbxObJ 객체에 m\_vLook 방향 벡터를 만들어 회전 시에 방향이 바뀌고 캐릭터 월드행렬 \_31,\_32\_33 요소를 더하고 정규화해서 m\_vLook 방향 벡터를 저장, 캐릭터 이동 구현

- 키보드 입력 시에 해당 상황에 맞는 애니메이션, 시작 시간과 끝타임으로 원하는 애니메이션을 출력하는 함수로 구현하여, 어떤 오브젝트이든 원하는 키프레임 시간만 알면 애니메이션을 구현할 수 있게 범용성과 유연성을 높임

- 사용자 편의를 위해 Write 2D Text로 오른쪽 하단에 조작 설명을 함 F1은 프레임 F4는 전체 애니메이션 재생 방향키로 캐릭터 이동을 알려줌

- 3D 그림자, 프로젝션 쉐도우, 쉐도우 디버깅 미니맵 적용