

Game Programming with DirectX

Direct3D Graphics Pipeline
(Animation)
Skeletal Animation

일괄 처리(Batch Processing)

• 일괄 처리의 이해

- 일괄 처리(Batch Processing)
같은 상태(State)의 일(작업)들을 모아서 한꺼번에 처리하는 것
상태의 변화를 최소화함으로써 전체 작업의 효율성을 얻을 수 있음

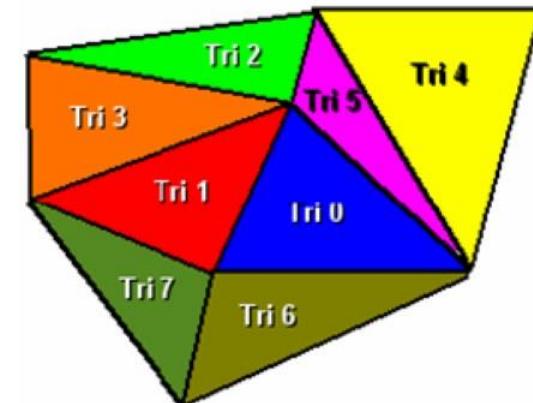
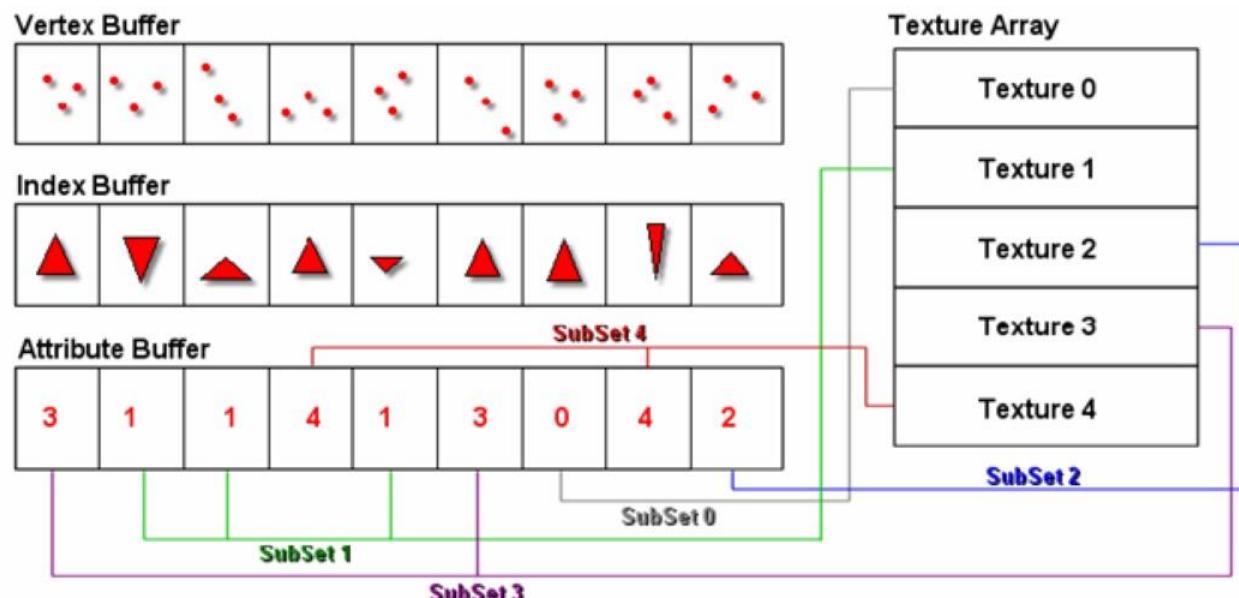
• 렌더링에도 일괄 처리의 개념을 적용

- 배치(Batching) 렌더링
그림 그리기(색칠하기)를 생각하라!
그리는 순서는 중요하지 않다!(왜?)
디바이스의 상태 변화와 데이터의 이동 등을 최소화하자.
- 모델 메쉬의 렌더링
조명(a개), 재질(b개), 텍스쳐(c개)를 사용하는 모델 메쉬의 경우
하나의 다각형을 렌더링하기 위하여 필요한 상태의 경우 = $(a \times b \times c)$ 개
모든 다각형마다 상태가 변한다면 상태를 변경하는 시간이 많아짐
같은 상태의 다각형들을 모아서 한꺼번에(일괄) 렌더링을 한다면?
- 배치 렌더링의 기준을 무엇으로 할 것인가?
일반적으로 재질을 기준으로 설정
재질은 다각형의 표면이 조명에 대한 반응을 어떻게 하는 가를 표현함
텍스쳐도 재질의 일부로 간주할 수도 있음
- 파일에서 모델(메쉬)을 로드하여 렌더링을 할 때도 배치 렌더링을 고려하라.

메쉬(모델)

• 기하학적 메쉬(모델)의 이해

- 배치(Batching) 렌더링
- 파일에서 모델(기하 데이터)을 로딩
- 충돌 검사, 바운딩 박스 계산, 정점 벡터 계산, 복제(Cloning), 최적화, LOD
- 버퍼 관리
 - 정점 버퍼, 인덱스 버퍼, 속성(Attribute) 버퍼, 인접성(Adjacency) 버퍼
- 속성(Attribute) 버퍼
 - 각 면(삼각형)의 속성 값을 저장
- 인접성(Adjacency) 버퍼
 - 각 면의 연결성에 대한 정보를 저장



Tri 0)	1 , 5 , 6
Tri 1)	3 , 0 , 7
Tri 2)	- , 5 , 3
Tri 3)	- , 2 , 1
Tri 4)	- , 5 , -
Tri 5)	2 , 4 , 0
Tri 6)	0 , - , 7
Tri 7)	1 , 6 , -

메쉬(모델)

- **기하학적 메쉬(모델)의 이해**

- 속성(Attribute)

인덱스 버퍼의 각 삼각형에 대한 속성 값(속성 번호)을 저장

어떤 데이터라도 속성으로 사용할 수 있음

일반적으로 속성은 텍스쳐, 재질, 광원 또는 디바이스(렌더) 상태 등을 사용

속성 값이 같은 삼각형들의 집합은 서브셋(Subset)이라고 함

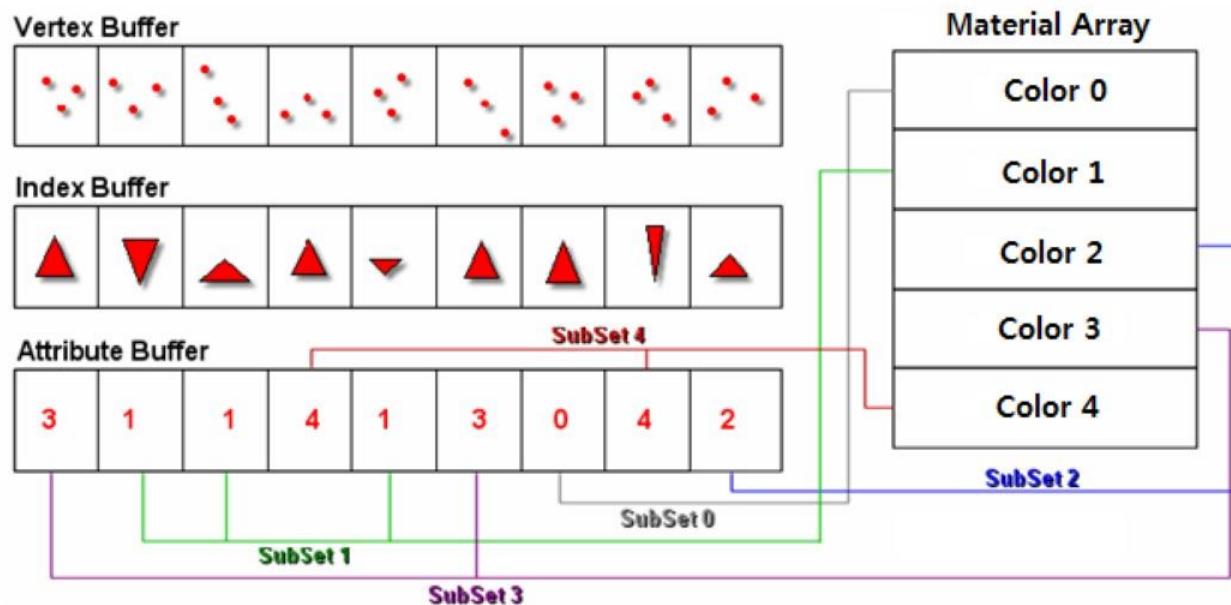
하나의 서브셋에 속한 삼각형들은 같은 디바이스 상태에서 렌더링할 수 있음

- 배치 렌더링(Batch Rendering)

배치 렌더링을 하기 위하여 같은 속성 값을 갖는 삼각형들을 한꺼번에 렌더링

`DrawSubset(UINT nAttributeID);`

- 속성을 사용하여 특정한 서브셋 삼각형들의 렌더링을 조절할 수 있음

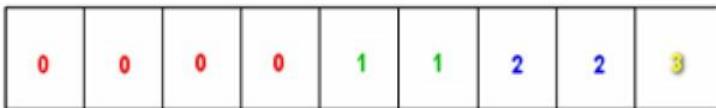
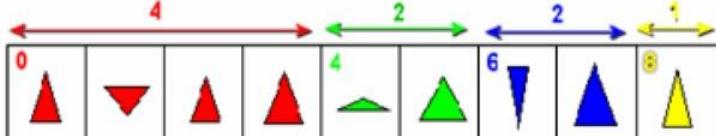
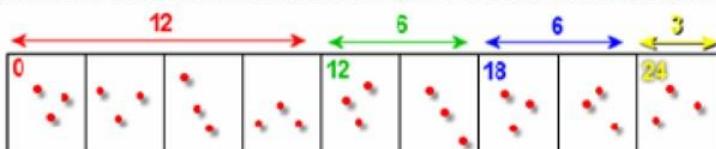
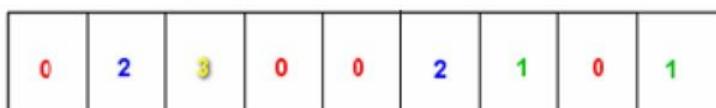
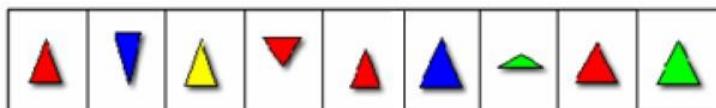
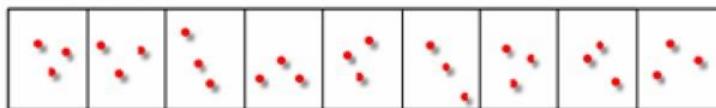


메쉬(모델)

- 기하학적 메쉬(모델)의 이해

- 최적화(Optimization)

- 속성 값을 기준으로 정점 버퍼와 인덱스 버퍼를 조정(순서화)



DrawSubset(0)

1st, 4th, 5th, 8th 삼각형을 렌더링

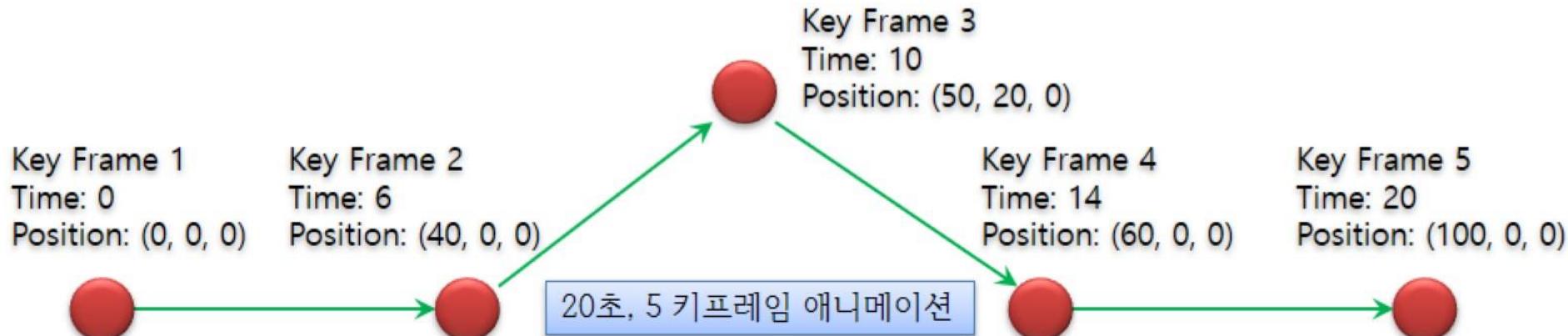
속성 테이블



```
struct ATTRIBUTE_RANGE
{
    UINT m_nAttributelD;
    UINT m_nFaceStart;
    UINT m_nFaceCount;
    UINT m_nVertexStart;
    UINT m_nVertexCount;
};
```

애니메이션(Animation)

- 3D 모델링 도구, 아티스트(Artist), 애니메이터(Animator)
 - 프로그래머는 모델(애니메이션) 데이터를 게임 프레임워크로 통합하는데 집중
 - 데이터(파일: File)의 포맷(Format)을 이해하고 결정하는 것이 중요
 - 데이터 로드(Load), 애니메이션(Animation), 렌더링(Rendering)
- 시간(Time)
 - 애니메이션은 시간에 대한 변환 함수
 - 초(Second), 틱(Tick), ...
 - 애니메이션은 어떤 시각에 대하여 프레임 계층 구조의 프레임을 변환하는 것임
- 키프레임 애니메이션(Keyframe Animation)
 - 키프레임: (키=시간 + 프레임)
특정한 시간에 대한 프레임의 위치, 방향, 크기와 같은 변환 정보를 저장
 - 보간(Interpolation)
임의의 시간에 대한 변환 정보를 동적으로 생성



애니메이션(Animation)

• 모델링 도구

- Autodesk 3DS MAX®

너무 많은 기능과 다양한 용도 때문에 표준적인 파일 포맷을 정하기 어려움
필요에 따라 모델의 데이터를 추출(Export)하기 위한 방법을 제공

- 3DS MAX® SDK

텍스트(Text) 또는 바이너리(Binary) 형식(ASE, XML, ...)

3DS MAX®를 설치할 때 선택하여 설치할 수 있음

- 3DS MAX 스크립트(3DS MAX Script)

데이터의 추출 뿐 아니라 모든 기능을 스크립트로 제어할 수 있음

3DS MAX®를 확장하기 위한 용도로 사용할 수 있음

- 원하는 데이터를 추출하기 위하여 DLL 형태의 익스포터(Exporter)를 작성해야 함

우선적으로 3DS MAX®의 기능을 이해하는 것이 중요

어떤 데이터를 어떻게 사용할 것인가를 결정하는 것이 중요

- 기본적인 익스포터가 제공됨

- C:\Program Files\Autodesk\3ds Max 2017\stdplugs

- 작성한 익스포터는 다음 폴더에 복사하면 됨

- C:\Program Files\Autodesk\3ds Max 2017\plugins
익스포터 DLL 파일의 확장자는 DLE

- 모델을 파일로 익스포트하기

"File → Export → Export"

- 익스포트된 모델 파일을 사용하기

어떻게(?): 그대로 사용, **변환 프로그램**



애니메이션(Animation)

- 애니메이션 데이터(Animation Data)

AnimationSet Walk

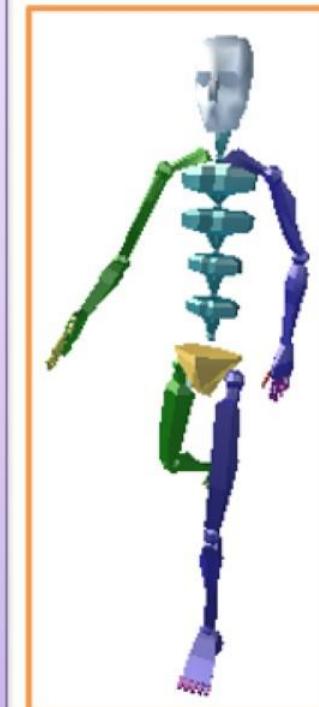
```
{  
    AnimationPeriod  
    Animation  
    {  
        [Pelvis]  
        AnimationKey  
        {  
            { { Key-FrameTime SRT }, ... }  
        }  
    }  
    Animation  
    {  
        [Left Thigh]  
        AnimationKey  
        {  
            { { Key-FrameTime SRT }, ... }  
        }  
    }  
    ...  
}
```

AnimationSet Shoot

```
{  
    AnimationPeriod  
    Animation  
    {  
        [Right Upper Arm]  
        AnimationKey  
        {  
            { { Key-FrameTime SRT }, ... }  
        }  
        Animation  
        {  
            [Right Hand]  
            AnimationKey  
            {  
                { { Key-FrameTime SRT }, ... }  
            }  
            ...  
        }  
    }  
}
```

```
struct FRAME
```

```
{  
    LPSTR m_pstrFrameName;  
    XMFLOAT4X4 m_xmf4x4Transform;  
    CMesh **m_ppMeshes;  
    int m_nMeshes;  
    FRAME *m_pChildFrame;  
    FRAME *m_pSiblingFrame;  
};
```



애니메이션(Animation)

- 애니메이션 템플릿(Animation Templates)

- 애니메이션 = 프레임 계층 구조의 하나의 프레임에 대한 키프레임(변환)들의 집합
 - 애니메이션 = 키프레임들의 배열(집합)
 - 일반적으로 키프레임의 변환 정보는 부모 프레임에 상대적인 SRT 벡터임
- 애니메이션 집합(Set) = 애니메이션들의 집합(예: "Walk", "Shoot", "Run", "Kick", ...)

```
class CAnimationController
{
    DWORD m_dwAnimationTicksPerSecond;
    float m_fTime;
    int m_nAnimationSets;
    CAnimationSet *m_pAnimationSets;
    int m_nAnimationTracks;
    CAnimationTrack *m_pAnimationTracks;
    TRACKEVENT *m_pTrackEvents;
    FRAME *m_pFrameRoot;
}
```

```
class CAnimationTrack
{
    BOOL m_bEnable;
    float m_fSpeed;
    float m_fPosition;
    float m_fWeight;
    CAnimationSet *m_pAnimationSet;
}
```

```
class CAnimationSet
{
    _TCHAR m_strName[256];
    float m_fPosition;
    int m_nType; //Once, Loop, PingPong
    int m_nAnimations;
    CAnimation *m_pAnimations;
}
```

```
class CAnimation
{
    _TCHAR m_strFrameName[256];
    int m_nAnimationKeys;
    CAnimationKey *m_pAnimationKeys;
}
```

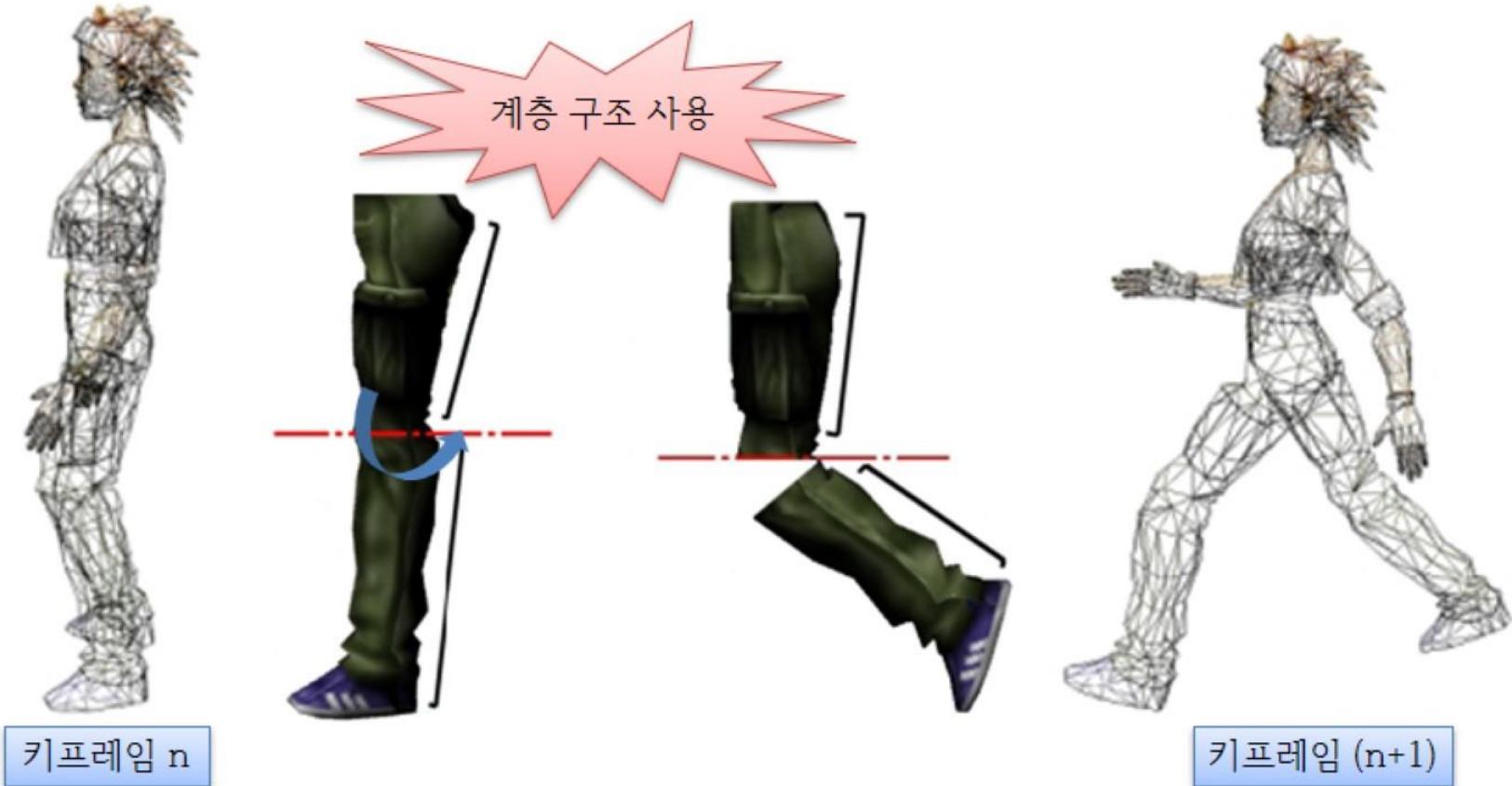
```
class CAnimationKey
{
    int m_nKeys;
    TIMEDFLOATKEYS *m_pKeys;
}
```

```
struct TIMEDFLOATKEYS
{
    float m_fTime;
    int m_nValues;
    float *m_pfValues;
}
```

애니메이션(Animation)

- 캐릭터 애니메이션(Character Animation)

- 분절된 캐릭터 모델(Segmented Character Model)
캐릭터를 분절된 다각형으로 분할하여 계층 구조로 표현
일반적으로 프레임 계층 구조는 캐릭터의 뼈대(Bone)와 연결된 피부를 표현함
각 뼈대들이 움직일 때 연결된 피부 메쉬도 따라서 움직임
- 각 뼈대들에 연결된 피부 메쉬들이 독립적으로 뼈대를 따라 움직이게 됨
관절 부위에서 메쉬들이 연결되지 않고 틈이 생기게 됨(왜? 어떻게 해결?)



애니메이션(Animation)

- 트위닝(Tweening, Morphing)

- 트위닝 게임 캐릭터

- id Software® (Quake II™): md2 파일 형식
 - 관절 부위에서 메쉬의 틈이 생기지 않도록 메쉬를 분할하지 않음
 - 각 키프레임의 위치에서 변환 데이터를 포함하지 않고 메쉬를 따로 생성/저장
 - 임의의 시간에 대한 메쉬는 인접한 두 개의 메쉬의 정점들을 직접 보간하여 구함
 - 프레임 계층 구조를 사용하지 않고 정해진(Canned) 동작들을 메쉬들로 직접 표현



키프레임 n

fTime

키프레임 (n+1)

생성된(Tweened) 정점 데이터

```
n = int(fTime / m_fInterval);
t = (fTime - m_pKeyFrames[n].m_fTime) / m_fInterval;
for (int i = 0; i < m_nVertices; i++)
{
    CVertex v1 = m_pKeyFrames[n].m_pVertices[i];
    CVertex v2 = m_pKeyFrames[n+1].m_pVertices[i];
    pVertices[i] = v1 * (1.0 - t) + v2 * t;
}
```

실시간으로 모델 정점을 보간하여 모델 생성
(보간을 통한 정점을 생성하기 위한 시간이 많이 필요)
(관절 부근에서 다각형의 크기가 변하게 됨)
키프레임의 모든 위치에서 포즈(Pose) 데이터를 가짐
(너무 많은 메모리가 필요)
얼굴(Facial) 애니메이션과 같은 분야에서 많이 사용

애니메이션(Animation)

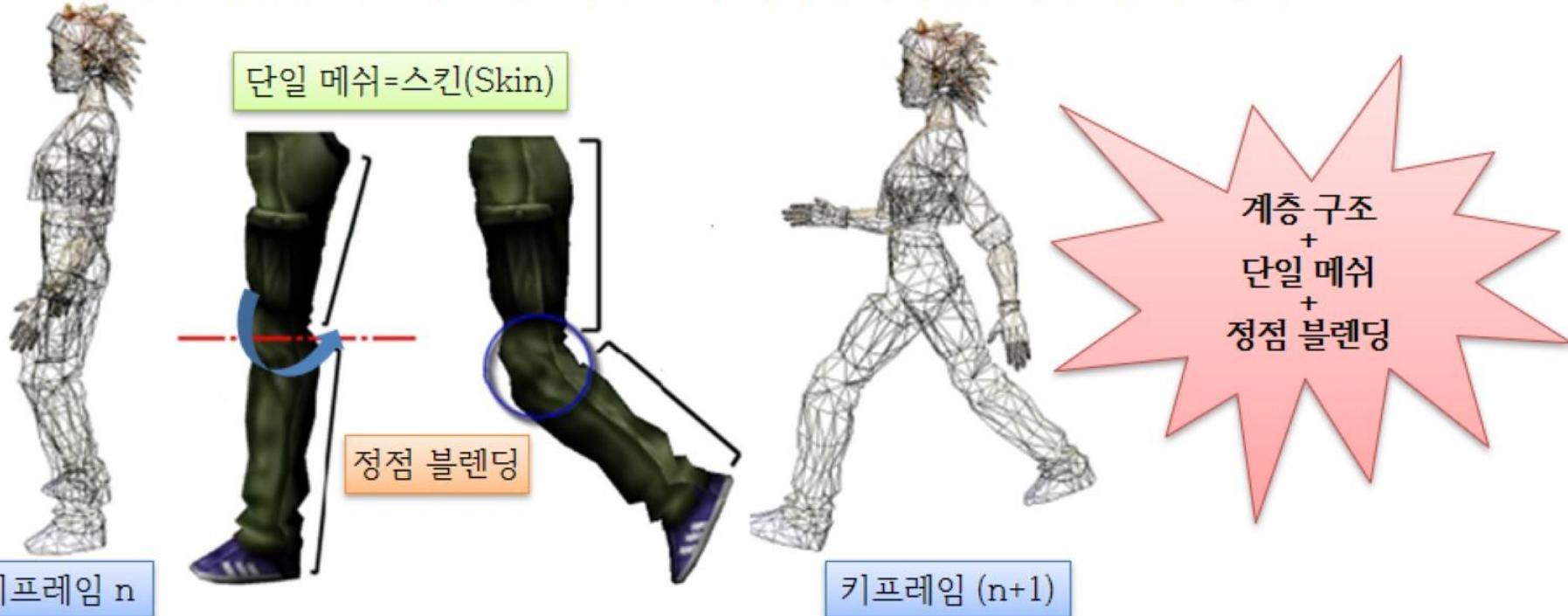
- 캐릭터 애니메이션(Character Animation)

- 정점 블렌딩(Vertex Blending)

- 캐릭터를 계층 구조로 표현하면서 관절 부위에서 틈이 생기지 않도록 하려면?
 - 캐릭터 모델 = 분절된 캐릭터 모델 + 트위닝 캐릭터 모델
 - 트위닝 캐릭터 모델의 장점
정점 블렌딩: 관절 부위에서 정점들이 인접한 키프레임 정점들로 보간
 - 분절된 캐릭터 모델
계층 구조의 장점, 다중 메쉬를 사용하는 단점

- 스키닝(Skinned Character Model)

프레임 계층 구조를 사용하면서 관절 부위에서 다각형들이 신축성을 부여



애니메이션(Animation)

- 스키닝 캐릭터 애니메이션(Skinned Character Animation)

- 스키닝 캐릭터 모델
 - 기본 자세(Reference Pose)에 대한 모델 생성

- 뼈대(Bone, Joint)

계층 구조에서 부모 뼈대에 상대적인 포즈(변환)

(포즈(Pose) = 위치 + 방향 = 행렬)

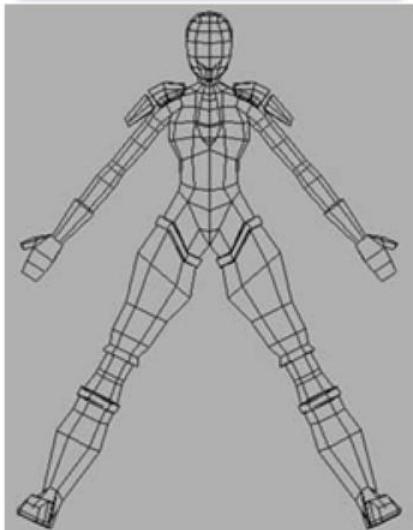
뼈대의 변환 행렬이 아닌 관절의 변환 행렬

스킨 메쉬 + 뼈대(Bone) 구조

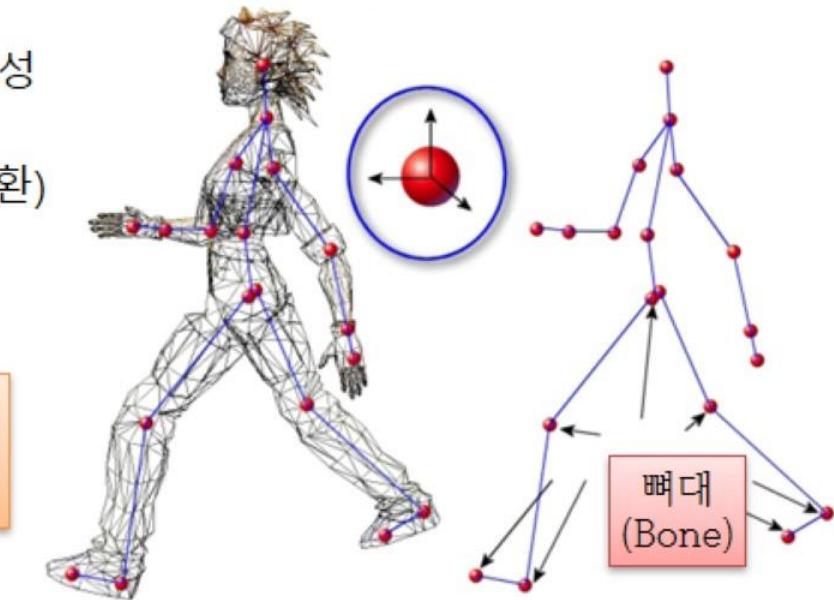
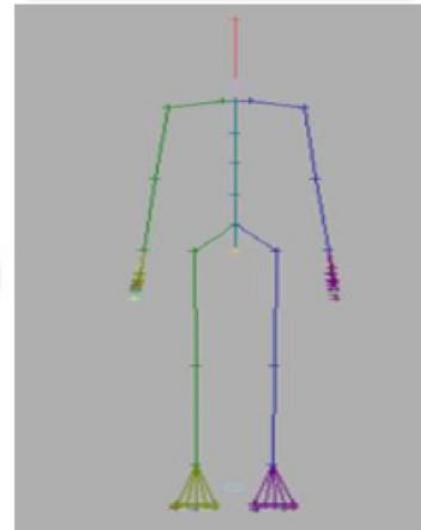
뼈대가 움직일 때 따라 움직이는 정점들과 가중치를 정의

뼈대의 움직임(애니메이션)을 정의

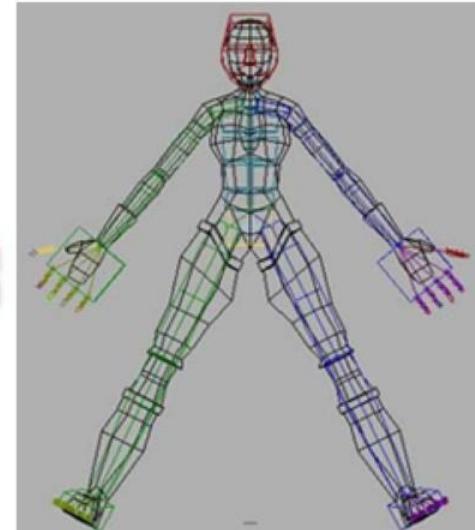
단일 메쉬=스킨(Skin)



뼈대 구조(Bone)



뼈대
(Bone)



애니메이션(Animation)

- 스키닝 캐릭터 애니메이션(Skinned Character Animation)

- 스키н 모델(Skin Model)

스킨 메쉬는 모델 좌표계(바인드 좌표계: Bind Space)에서 기본 자세로 생성
(스킨 메쉬가 프레임 계층 구조의 루트 프레임에 존재할 때 루트 프레임의 좌표계)

- 각 프레임(뼈대)은 스키н 메쉬의 일부분(정점들)의 위치와 방향에 영향을 줌

- 루트-변환 행렬(To-Root Matrix)

- 일반적으로 단일 메쉬인 스키н 모델은 루트 프레임에 저장됨
 - 각 프레임을 루트 프레임 좌표계로 변환하기 위한 행렬이 필요

프레임 F_i 의 좌표계를 부모 프레임 좌표계로 변환하는 행렬(To-Parent Matrix): P_i

프레임 F_i 의 자식 프레임: F_{i+1}

프레임의 루트 프레임 F_0 부터 프레임 F_i 까지 변환 행렬의 곱(To-Root Matrix): R_i

$$R_i = (P_i * P_{i-1} * \dots * P_0)$$

프레임 F_{i+1} 를 루트 프레임 좌표계로 변환하기 위한 행렬

$$R_{i+1} = P_{i+1} * R_i = (P_{i+1} * P_i * \dots * P_0) : \text{Top Down으로 효율적으로 계산}$$

- 뼈대 오프셋 행렬(Bone Offset Matrix, 스키н 오프셋 행렬(Skin Offset Matrix))

- **기본 자세의 계층 구조**에서 프레임 F_i 의 루트 변환 행렬 R_i 의 역행렬($(R_i)^{-1}$)
 - 기본 자세의 계층 구조에서 스키н 메쉬의 정점을 프레임 F_i 의 좌표계로 변환

스킨 메쉬 정점
(루트 프레임의 좌표계)



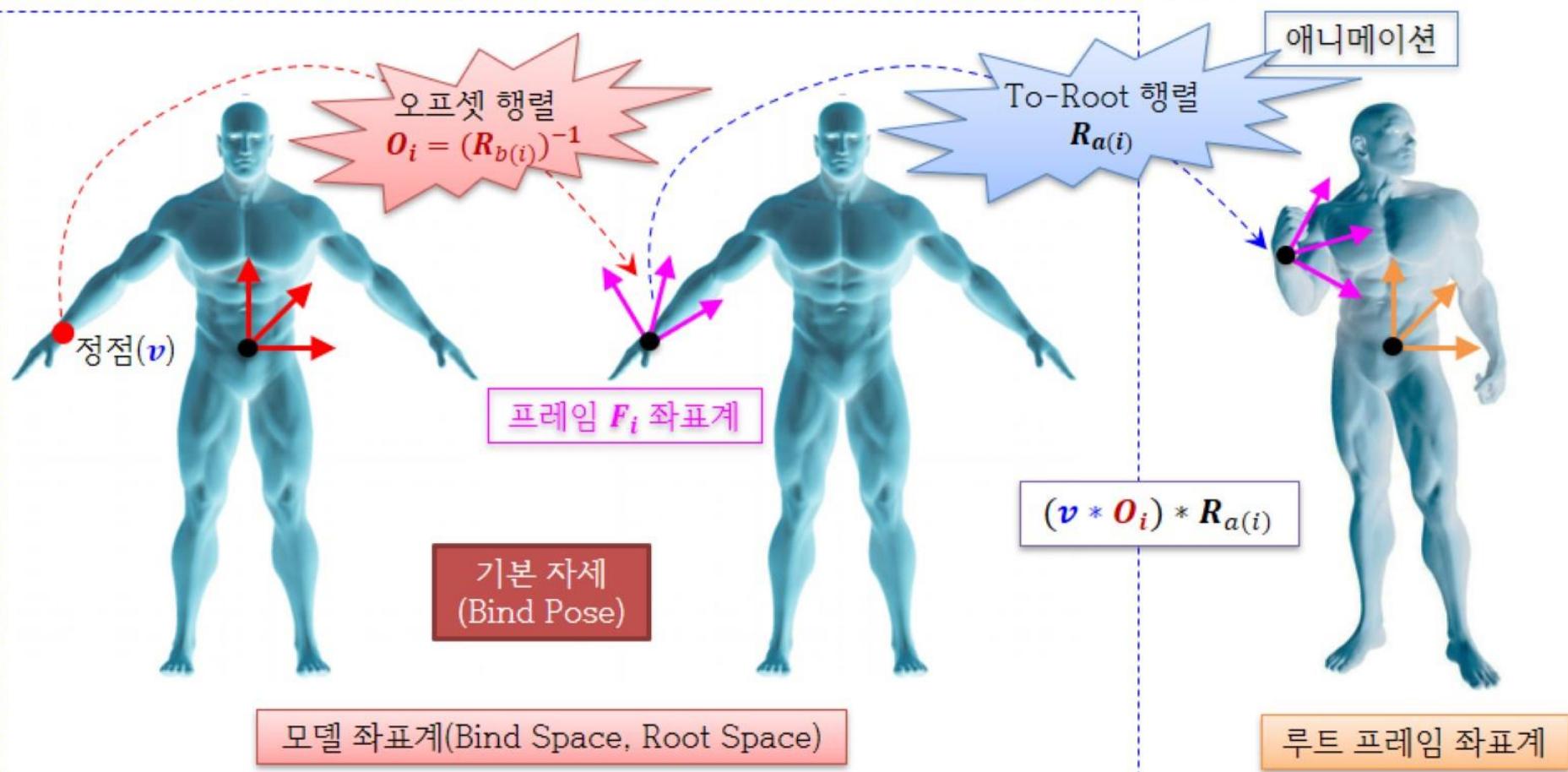
프레임 F_i 의 좌표계 정점

애니메이션(Animation)

- 스키닝 캐릭터 애니메이션(Skinned Character Animation)

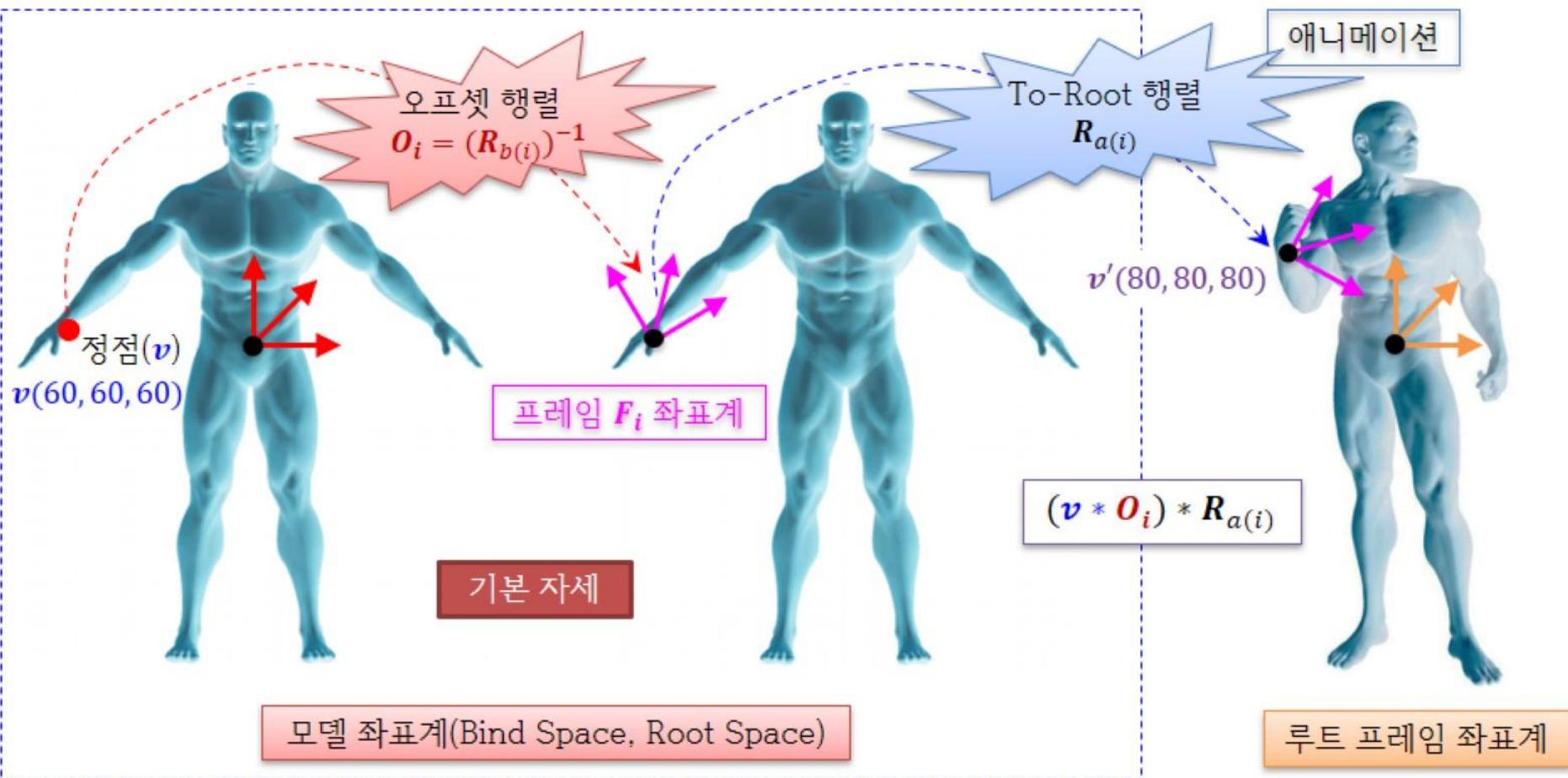
- 뼈대 오프셋 행렬(Bone Offset Matrix)

- 오프셋 변환(Offset Transformation)
 - 스킨 메쉬의 정점을 각 프레임(뼈대)의 오프셋 행렬으로 프레임 좌표계로 변환
기본 자세에서 스킨 메쉬의 정점을 각 프레임의 좌표계로 변환
 - 변환된 정점을 애니메이션 계층 구조의 루트 프레임 좌표계로 변환



애니메이션(Animation)

- 스ки닝 캐릭터 애니메이션(Skinned Character Animation)
 - 뼈대 오프셋 행렬(Bone Offset Matrix)

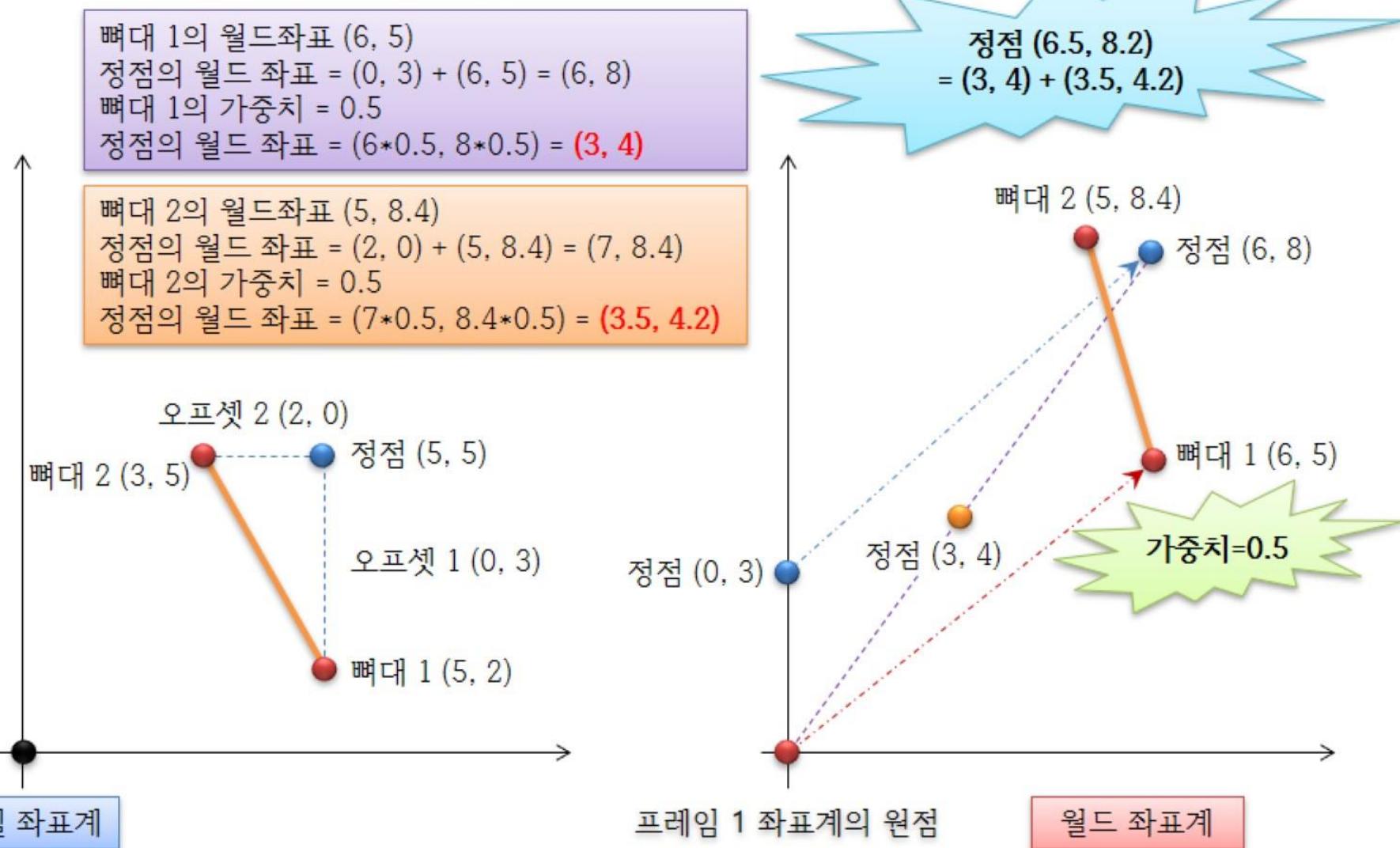


프레임 F_i 오프셋 행렬
 $\mathbf{b}(50, 50, 50)$

프레임 좌표계 정점: $v + (-\mathbf{b}) = (60, 60, 60) + (-50, -50, -50) = (10, 10, 10)$
모델 좌표계 정점: $v + \mathbf{b} = (10, 10, 10) + (70, 70, 70) = (80, 80, 80)$

애니메이션(Animation)

- 스ки닝 캐릭터 애니메이션(Skinned Character Animation)
 - 정점 블렌딩(Vertex Blending)



애니메이션(Animation)

- 스ки닝 캐릭터 애니메이션(Skinned Character Animation)
 - 정점 블렌딩(Vertex Blending)

정점 가중치(Weights)

뼈대 1: 0.25

뼈대 2: 0.5

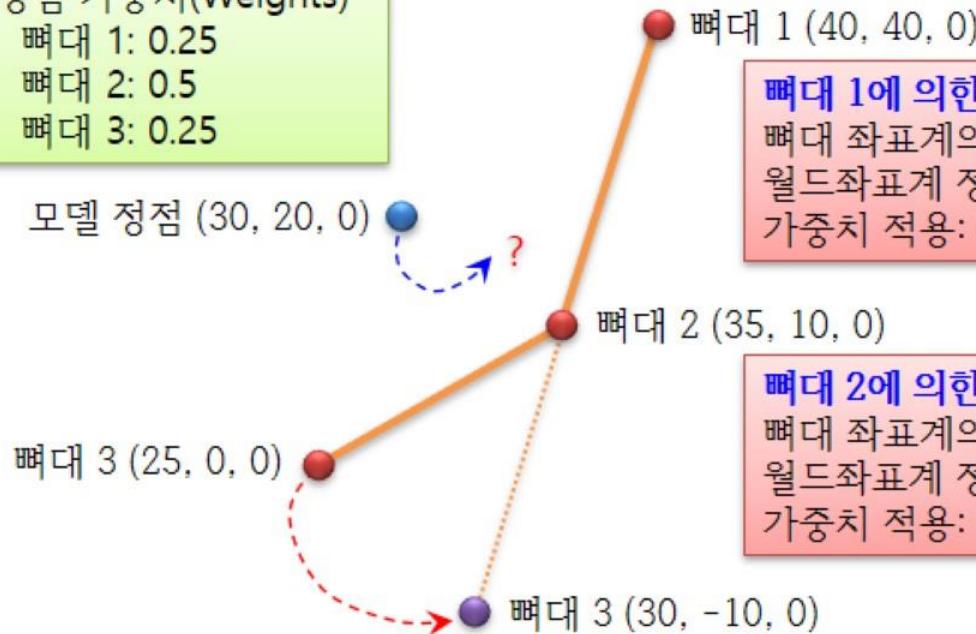
뼈대 3: 0.25

모델 정점 (30, 20, 0)

뼈대 3 (25, 0, 0)

$$v_T = \sum_{i=1}^n (v * O_i * T_i * R_i) * W_i$$

정점의 최종 변환 위치: $(7.5, 5, 0) + (15, 10, 0) + (8.75, 2.5, 0) = (31.25, 17.5, 0)$



뼈대 1에 의한 정점의 변환

뼈대 좌표계의 정점: $(30, 20, 0) + (-40, -40, 0) = (-10, -20, 0)$

월드좌표계 정점: $(-10, -20, 0) + (40, 40, 0) = (30, 20, 0)$

가중치 적용: $(30, 20, 0) * 0.25 = (7.5, 5, 0)$

뼈대 2에 의한 정점의 변환

뼈대 좌표계의 정점: $(30, 20, 0) + (-35, -10, 0) = (-5, 10, 0)$

월드좌표계 정점: $(-5, 10, 0) + (35, 10, 0) = (30, 20, 0)$

가중치 적용: $(30, 20, 0) * 0.5 = (15, 10, 0)$

뼈대 3에 의한 정점의 변환

뼈대 좌표계의 정점: $(30, 20, 0) + (-25, 0, 0) = (5, 20, 0)$

월드좌표계 정점: $(5, 20, 0) + (30, -10, 0) = (35, 10, 0)$

가중치 적용: $(35, 10, 0) * 0.25 = (8.75, 2.5, 0)$

애니메이션(Animation)

- 스킨 애니메이션 모델의 예

- 스킨 애니메이션 모델(Skinned Animation Model)
 - 프레임(Frame) 또는 뼈대(Bone)들의 계층 구조(Frame Hierarchy, Tree)
 - 적어도 하나 이상의 스킨 메쉬(Skinned Mesh)를 가져야 함
 - 스킨 메쉬가 아닌 메쉬(Mesh)를 가질 수 있음(예: 총, 칼, 헬멧, 갑옷 등)
 - 애니메이션 정보(시간에 따른 프레임(뼈대)의 움직임(변환) 정보)
스킨 메쉬의 정점을 블렌딩(Deformation, Animation)하기 위한 정보
- 스킨 메쉬(Skinned Mesh)
바인드 포즈 프레임 정보(Bone Names, Bone Offsets, Bone Indices, Bone Weights)
- 스킨 메쉬가 한 개인 경우
- 스킨 메쉬가 두 개인 경우

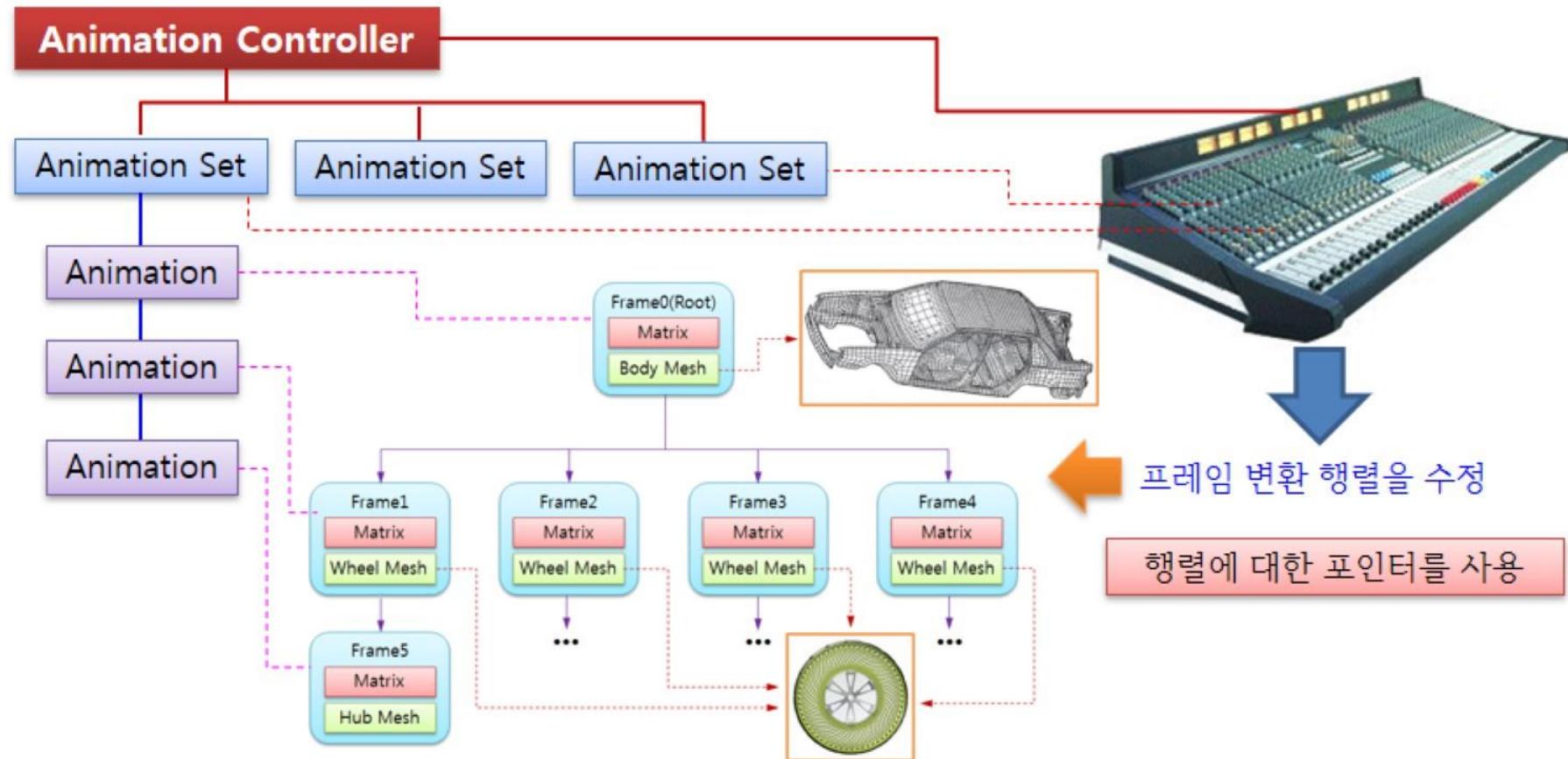
```
armor: L_leg R_leg chest L_elbow R_elbow head R_plate L_plate  
body: hips L_leg ... chest ... head ... L_eye R_eye jaw lip top
```



애니메이션(Animation)

- 애니메이션 제어기(Animation Controller)

- 포함된 애니메이션 집합들에 대한 시간을 관리하기 위한 전역 시간을 관리
게임 타이머 → 애니메이션 제어기 타이머 → 애니메이션 집합 타이머
- 애니메이션 트랙(Track)을 관리
- 애니메이션 믹서(Mixer)
여러 개의 애니메이션을 혼합(Mixing)하여 새로운 애니메이션을 출력하는 기능



애니메이션(Animation)

- 애니메이션 제어기

```
CAnimationController::AdvanceTime(float fElapsedTime)
{
    XMFLOAT3 Scale, Translate;
    XMFLOAT4 Rotate;
    XMFLOAT4X4 xmf4x4Frame;
    FRAME *pAnimationFrame;
    _TCHAR pstrAnimationName[256];
    m_fTime += fElapsedTime;
    for (int i = 0; i < m_nAnimationTracks; i++)
    {
        if (m_pAnimationTracks[i]->m_bEnable)
        {
            m_pAnimationTracks[i].m_fPosition += (fElapsedTime * m_pAnimationTracks[i].m_fSpeed);
            CAnimationSet *pAnimationSet = m_pAnimationTracks[i].m_pAnimationSet;
            float fPosition = pAnimationSet->GetPosition(m_pAnimationTracks[i].m_fPosition);
            for (int j = 0; j < pAnimationSet->m_nAnimations; j++)
            {
                pAnimationSet->GetSRT(fPosition, j, &Scale, &Rotate, &Translate);
                XMStoreFloat4X4(&xmf4x4Frame, XMMatrixTransformation(..., &Scale, NULL, &Rotate, &Translate));
                pAnimationSet->GetAnimationName(j, pstrAnimationName);
                pAnimationFrame = FindFrameByName(m_pFrameRoot, pstrAnimationName);
                pAnimationFrame->m_xmf4x4Transformation = xmf4x4Frame;
            }
        }
    }
}
```

```
class CAnimationController {
    float m_fTime;
    int m_nAnimationTracks;
    CAnimationTrack *m_pAnimationTracks;
    FRAME *m_pFrameRoot;
}
```

```
class CAnimationTrack {
    bool m_bEnable;
    float m_fSpeed;
    float m_fPosition;
    CAnimationSet *m_pAnimationSet;
}
```

애니메이션(Animation)

• XMMatrixTransformation 함수

XMMATRIX XMMatrixTransformation(

XMVECTOR ScalingOrigin,
XMVECTOR ScalingOrientation, //Quaternion
XMVECTOR Scaling,
XMVECTOR RotationOrigin,
XMVECTOR Rotation, //Quaternion
XMVECTOR Translation
);

ScalingOrigin	크기 변환의 중심
ScalingOrientation	크기 변환 회전
Scaling	크기 변환
RotationOrigin	회전 변환의 중심
Rotation	회전 변환
Translation	평행 이동 변환

$$M_{out} = (M_{sc})^{-1} * (M_{sr})^{-1} * M_s * M_{sr} * M_{sc} * (M_{rc})^{-1} * M_r * M_{rc} * M_t$$

XMMATRIX XMMatrixAffineTransformation(

XMVECTOR Scaling,
XMVECTOR RotationOrigin,
XMVECTOR Rotation, //Quaternion
XMVECTOR Translation
);

M_{sc}	크기 변환 중심 행렬
M_{sr}	크기 변환 회전 행렬
M_s	크기 변환 행렬
M_{rc}	회전 변환의 중심 행렬
M_r	회전 변환 행렬
M_t	평행 이동 변환 행렬

$$M_{out} = M_s * (M_{rc})^{-1} * M_r * M_{rc} * M_t$$

Scaling	크기 변환 값
RotationOrigin	회전 변환의 중심
Rotation	회전 변환
Translation	평행 이동 변환

애니메이션(Animation)

- 애니메이션 제어기

```
class CHierarchicalMesh {  
    CAnimationController *m_pAnimationController;  
    FRAME *m_pFrameRoot;  
  
    void Update(FRAME *pFrame, XMFLOAT4X4 *pxmf4x4Parent);  
    void Render(ID3D12GraphicsCommandList *pd3dCommandList, FRAME *pFrame);  
    CAnimationController *CloneAnimationController();  
}
```

```
class CGameObject {  
    XMFLOAT4X4 m_mtxWorld;  
    CHierarchicalMesh *m_pMesh;  
    CAnimationController *m_pAnimationController;  
  
    virtual void Animate(float fTimeElapsed, CCamera *pCamera=NULL);  
    virtual void Render(ID3D12GraphicsCommandList *pd3dCommandList, CCamera *pCamera=NULL);  
}
```

```
void CGameObject::Animate(float fTimeElapsed, CCamera *pCamera) {  
    m_pAnimationController->AdvanceTime(fElapsedTime, NULL);  
}
```

```
void CGameObject::SetMesh(ID3D12Device *pd3dDevice, ID3D12GraphicsCommandList *pd3dCommandList,  
    CHierarchicalMesh *pMesh) {  
    m_pMesh = pMesh;  
    if (pMesh->m_pAnimationController) m_pAnimationController = pMesh->CloneAnimationController();  
}
```

애니메이션(Animation)

- 애니메이션 제어기

- 애니메이션 믹서 트랙(Mixer Tracks)

각 트랙의 속성을 설정, 즉 트랙에 연결될 애니메이션 집합의 속성을 설정

- 트랙의 활성화/비활성화(Enable/Disable)
 - 가중치(Weight)
 - 속도(Speed)
 - 위치(Position)

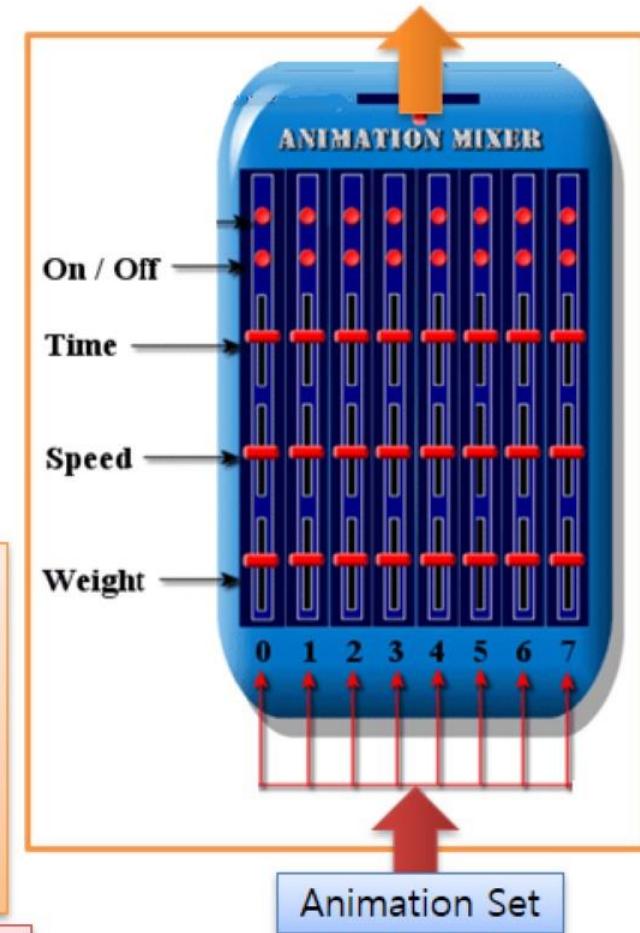
- 애니메이션 시퀀서(Sequencer)

특정 시간에 트랙의 속성을 변경하는 기능

```
struct TRACKEVENT
{
    float m_fStartTime;
    float m_fDuration;
    int m_nTransitionType;
    int m_nType;
    union {
        float m_fSpeed;
        float m_fWeight;
        float m_fPosition;
        bool m_bEnable;
    };
};
```

```
#define TRACK_SPEED 0
#define TRACK_WEIGHT 1
#define TRACK_POSITION 2
#define TRACK_ENABLE 3

class CAnimationTrack
{
    bool m_bEnable;
    float m_fSpeed;
    float m_fPosition;
    float m_fWeight;
    CAnimationSet *m_pAnimationSet;
}
```



$$\text{Linear Interpolation: } f(t) = (1 - t)x + ty$$

$$\text{Cubic Spline Interpolation: } f(t) = 2(x - y)t^3 + 3(y - x)t^2 + x$$

애니메이션(Animation)

- 애니메이션 제어기

- 애니메이션 콜백(Callback)

- 특정 시간에 함수를 호출할 수 있는 기능(예: 사운드 동기화, 이펙트 동기화)

```
void CAnimationController::AdvanceTime(float fTimeElapsed, CAnimationCallbackHandler *pCallbackHandler)
{
    m_fTime += fElapsed;
    for (int i = 0; i < m_nAnimationTracks; i++)
    {
        if (m_pAnimationTracks[i]->m_bEnable)
        {
            ...
            m_pAnimationTracks[i].m_fPosition += (fElapsed * m_pAnimationTracks[i].m_fSpeed);
            CAnimationSet *pAnimationSet = m_pAnimationTracks[i].m_pAnimationSet;
            if (pCallbackHandler)
            {
                void *pCallbackData = pAnimationSet->GetCallback(m_pAnimationTracks[i].m_fPosition);
                if (pCallbackData) pCallbackHandler->HandleCallback(i, pCallbackData);
            }
            ...
        }
    }
}

class CAnimationCallbackHandler
{
    virtual void HandleCallback(int nTrack, void *pCallbackData);
}

class CAnimationSet {
    _TCHAR m_strName[256];
    float m_fPosition;
    int m_nType;
    int m_nAnimations;
    CAnimation *m_pAnimations;
    int m_nCallbackKeys;
    CALLBACKKEY *m_pCallbackKeys;
}

struct CALLBACKKEY
{
    float m_fTime;
    void *m_pCallbackData;
}
```

애니메이션(Animation)

- 애니메이션 제어기
 - 애니메이션 콜백(Callback)

```
class CSoundCallbackHandler : CAnimationCallbackHandler
{
    virtual void HandleCallback(int nTrack, void *pCallbackData);
}

void CSoundCallbackHandler::HandleCallback(int nTrack, void *pCallbackData)
{
    _TCHAR *pWavName = (_TCHAR *)pCallbackData;
    PlaySound(pWavName, NULL, SND_FILENAME);
}

class CGameObjectWithSound::CAnimatedGameObject
{
    CSoundCallbackHandler *m_pCallbackHandler;
    virtual void SetCallbackHandler();
    virtual void Animate(float fTimeElapsed, CCamera *pCamera=NULL);
    virtual void Render(ID3D12GraphicsCommandList *pd3dCommandList, CCamera *pCamera=NULL);
}

CGameObjectWithSound::SetCallbackHandler()
{
    m_pCallbackHandler = new CSoundCallbackHandler();
    m_pAnimationController->SetCallbackKey(0, 0.5f, _T("Explosion.wav"));
}

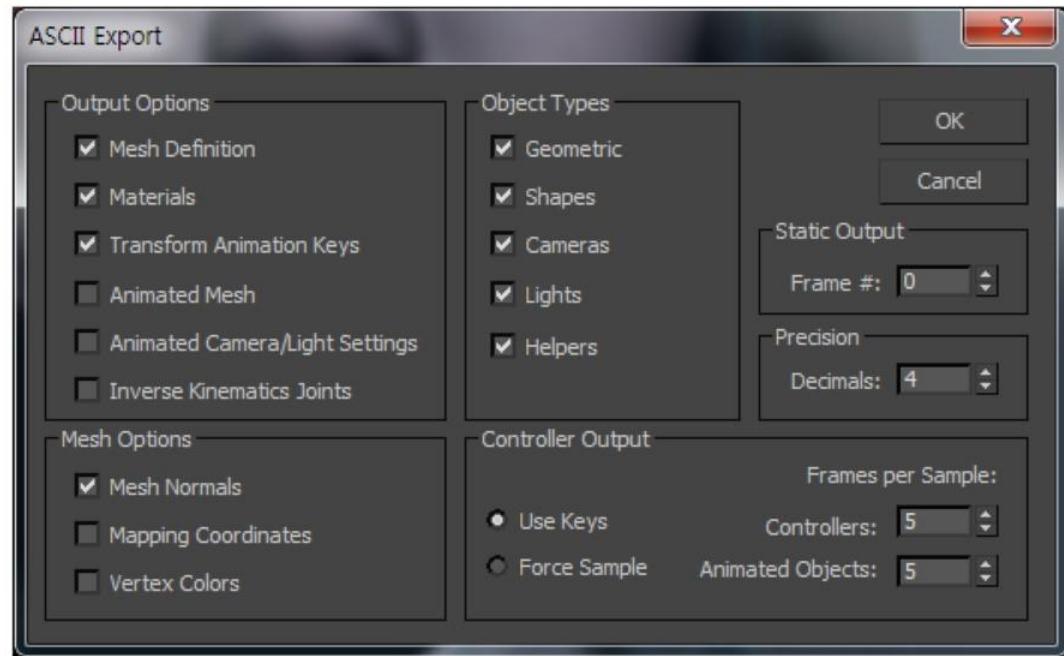
void CGameObject::Animate(float fTimeElapsed, CCamera *pCamera) {
    m_pAnimationController->AdvanceTime(fElapsed, m_pCallbackHandler);
}
```

```
struct CALLBACKKEY
{
    float m_fTime;
    void *m_pCallbackData;
}
```

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 3D Studio MAX®가 제공
- 텍스트 파일
 - 헤더(Header)
 - 씬(Scene) 정보
 - 재질 정보(Material List)
 - 객체 정보(Object List)
 - GeoObject
 - ShapeObject
 - LightObject
 - CameraObject
 - HelpObject
- ASE 파서(Parser)가 필요



C:\Program Files (x86)\Autodesk\3ds Max 2022 SDK\maxsdk\samples\import_export\asciexp

```
*SCENE {  
*SCENE_FILENAME "Alicia.max"  
*SCENE_FIRSTFRAME 0  
*SCENE_LASTFRAME 100  
*SCENE_FRAMESPEED 30  
*SCENE_TICKSPERFRAME 160  
*SCENE_BACKGROUND_STATIC 0.0000 0.0000 0.0000  
*SCENE_AMBIENT_STATIC 0.5255 0.5255 0.5255  
}
```

```
*3DSMAX_ASCIIEXPORT 200  
*COMMENT "AsciiExport Version 2.00 - Tue Dec 04 14:46:43 2012"
```

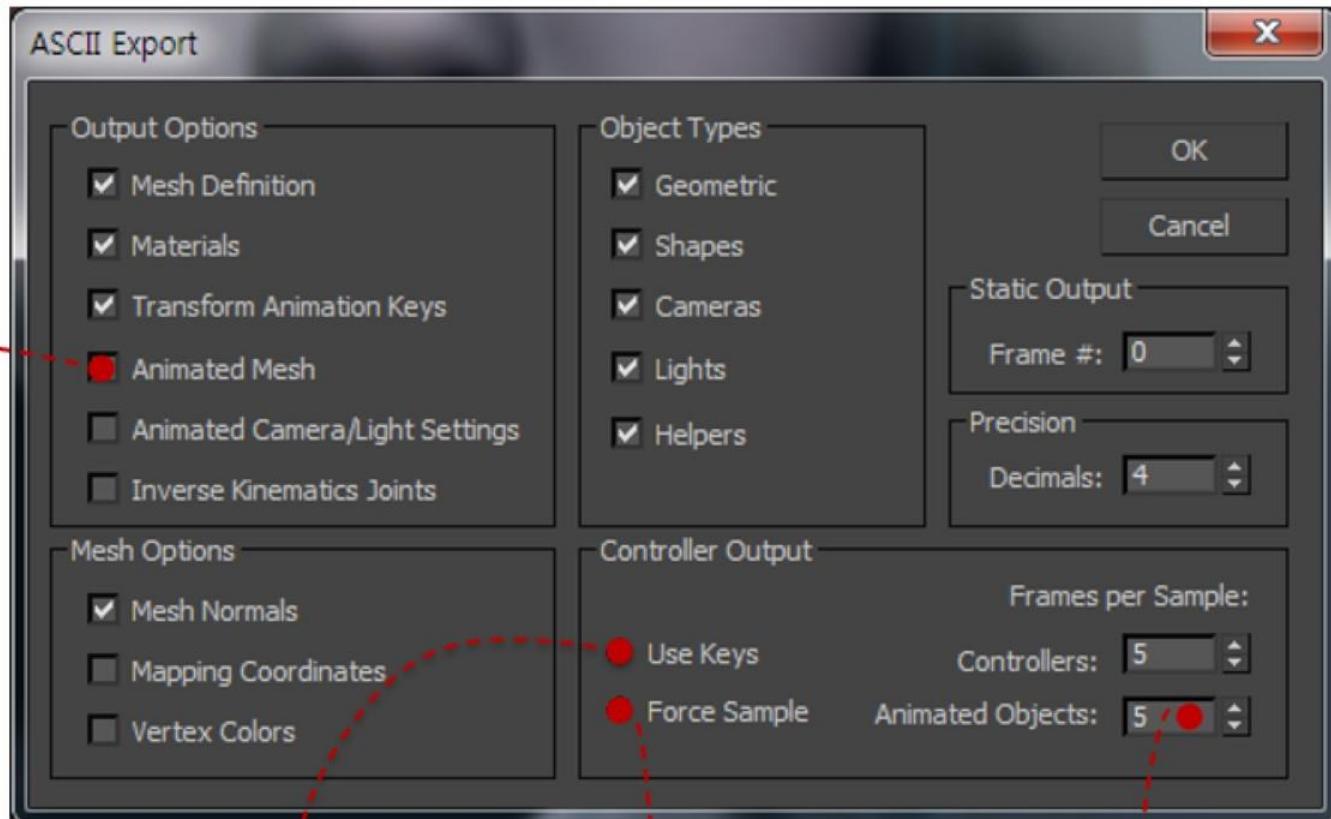
//30 프레임/초
//160 Ticks/프레임

4800 Ticks/초

*MATERIAL_LIST
*GEOMOBJECT
*HELPEROBJECT

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)



일정한 프레임마다 모든 정점을 출력

애니메이션 키를 출력

샘플링 간격(프레임 수)

일정한 간격으로 애니메이션 데이터를 샘플링하여 출력

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 기하 객체(GeoObject)

```
*GEOMOBJECT {  
*NODE_NAME "Bip01"  
*NODE_TM { ... }  
*MESH { ... }  
*PROP_MOTIONBLUR 0  
*PROP_CASTSHADOW 1  
*PROP_RECVSHADOW 1  
*WIREFRAME_COLOR 1.0000 0.0000 0.0000}
```

```
*MESH {  
*TIMEVALUE 0  
*MESH_NUMVERTEX 6  
*MESH_NUMFACES 8  
*MESH_VERTEX_LIST { ... }  
*MESH_FACE_LIST { ... }  
*MESH_NORMALS { ... }  
}
```

```
*MESH_VERTEX_LIST {  
*MESH_VERTEX 0 0.5078 20.4323 665.4061  
*MESH_VERTEX 1 0.5079 -4.8771 640.0967  
...  
*MESH_VERTEX  
*MESH_VERTEX}
```

```
*MESH_FACE_LIST { //반시계 방향(CCW)  
*MESH_FACE 0: A: 0 B: 1 C: 2 AB: 1 BC: 1 CA: 1 *MESH_SMOOTHING 1 *MESH_MTLID 0  
... //와이어 프레임일 때 그리는가?  
*MESH_FACE 7: A: 5 B: 1 C: 4 AB: 1 BC: 1 CA: 1 *MESH_SMOOTHING 1 *MESH_MTLID 0  
}
```

*MESH_NORMALS {

```
*MESH_FACENORMAL 0 0.5774 0.5774 0.5774  
*MESH_VERTEXNORMAL 0 0.0000 -0.0000 1.0000  
*MESH_VERTEXNORMAL 1 1.0000 0.0000 0.0000  
*MESH_VERTEXNORMAL 2 -0.0000 1.0000 0.0000
```

...

```
*MESH_FACENORMAL  
*MESH_VERT  
*MESH_VERT  
*MESH_VERT
```

}

*NODE_TM {

```
*NODE_NAME "Bip01"  
*INHERIT_POS 0 0 0  
*INHERIT_ROT 0 0 0  
*INHERIT_SCL 1 1 1  
*TM_ROW0 0.0000 -1.0000 0.0000  
*TM_ROW1 1.0000 0.0000 0.0000  
*TM_ROW2 0.0000 -0.0000 1.0000  
*TM_ROW3 0.5078 20.4323 640.0967  
*TM_POS 0.5078 20.4323 640.0967  
*TM_ROTAXIS -0.0000 -0.0000 1.0000  
*TM_ROTANGLE 1.5708  
*TM_SCALE 1.0000 1.0000 1.0000  
*TM_SCALEAXIS 0.0000 0.0000 0.0000  
*TM_SCALEAXISANG 0.0000  
}
```

쿼터니언

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 기하 객체(Model)

```
*GEOMOBJECT {  
    *NODE_NAME "Bip01 Pelvis"  
    *NODE_PARENT "Bip01"  
    *NODE_TM { ... }  
    *MESH { ... }  
    *PROP_MOTIONBLUR 0  
    *PROP_CASTSHADOW 1  
    *PROP_RECVSHADOW 1  
    *WIREFRAME_COLOR 0.6510 0.7922 0.9412  
}
```

```
*MESH {  
    *TIMEVALUE 0  
    *MESH_NUMVERTEX 9  
    *MESH_NUMFACES 8  
    *MESH_VERTEX_LIST { ... }  
    *MESH_FACE_LIST { ... }  
    *MESH_NORMALS { ... }  
}
```

```
*MESH_VERTEX_LIST {  
    *MESH_VERTEX 0 0.5078 20.4323 48.2923  
    *MESH_VERTEX 1 10.1000 0.0000 10.0000  
}
```

```
*MESH_FACE_LIST {  
    ...  
    *MESH_FACE 0: A: 0 B: 1 C: 2 AB: 0 BC: 1 CA: 0 *MESH_SMOOTHING 2 *MESH_MTLID 0  
    ...  
    *MESH_FACE 7: A: 0 B: 8 C: 1 AB: 0 BC: 1 CA: 0 *MESH_SMOOTHING 2 *MESH_MTLID 0  
}
```

```
*MESH_NORMALS {
```

```
    *MESH_FACENORMAL 0 0.0000 0.0000 1.0000  
    *MESH_VERTEXNORMAL 0 0.0000 0.0000 1.0000  
    *MESH_VERTEXNORMAL 1 0.0000 0.0000 1.0000  
    *MESH_VERTEXNORMAL 2 0.0000 0.0000 1.0000  
    ...
```

```
    *MESH_FACENORMAL  
    *MESH_VERT  
    *MESH_VERT  
    *MESH_VERT
```

```
*NODE_TM {
```

```
    *NODE_NAME "Bip01 Footsteps"  
    *INHERIT_POS 1 1 1  
    *INHERIT_ROT 1 1 1  
    *INHERIT_SCL 1 1 1  
    *TM_ROW0 1.0000 0.0000 0.0000  
    *TM_ROW1 0.0000 1.0000 0.0000  
    *TM_ROW2 0.0000 0.0000 1.0000  
    *TM_ROW3 0.5078 20.4323 48.2923  
    *TM_POS 0.5078 20.4323 48.2923  
    *TM_ROTAXIS 0.0000 0.0000 0.0000  
    *TM_ROTANGLE 0.0000  
    *TM_SCALE 1.0000 1.0000 1.0000  
    *TM_SCALEAXIS 0.0000 0.0000 0.0000  
    *TM_SCALEAXISANG 0.0000  
}
```

*MESH_MTLID 0 //재질

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 기하 객체(Model)

```
*GEOMOBJECT {  
  *NODE_NAME "Bip01 Spine"  
  *NODE_PARENT "Bip01 Pelvis"  
  *NODE_TM { ... }  
  *MESH { ... }  
  *PROP_MOTIONBLUR 0  
  *PROP_CASTSHADOW 1  
  *PROP_RECVSHADOW 1  
  *WIREFRAME_COLOR 0.6510 0.7922 0.9412  
  *MATERIAL_REF 0 //재질 (Default)  
}
```

```
*MESH_NUMCVERTEX 4  
*MESH_CVERTLIST {  
  *MESH_VERTCOL 0 1.0000 0.0000 0.0000  
  ...  
  *MESH_VERTCOL 3 1.0000 1.0000 1.0000  
}
```

색상(Color)

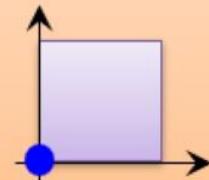
```
*MESH_NUMCVFACES 2  
*MESH_NUMCFACELIST {  
  *MESH_CFACE 0 3 0 2  
  *MESH_CFACE 1 1 2 0  
}
```

```
*MESH {  
  *TIMEVALUE 0  
  *MESH_NUMVERTEX 142  
  *MESH_NUMFACES 254  
  *MESH_VERTEX_LIST { ... }  
  *MESH_FACE_LIST { ... }  
  *MESH_NUMTVERTEX 158  
  *MESH_TVERTLIST { ... }  
  *MESH_NUMTVFACES 254  
  *MESH_TFACELIST { ... }  
  *MESH_NUMCVERTEX 4  
  *MESH_CVERTLIST { ... }  
  *MESH_NORMALS { ... }  
}
```

텍스쳐 v-좌표
($v = 1.0 - v$)로 사용

```
*MESH_TVERTLIST {  
  *MESH_TVERT 0 0.1381 0.4058 0.0000  
  *MESH_TVERT 1 0.0696 0.3917 0.0000  
  ...  
  *MESH_TVERT 157 0.1884 0.4874 0.0000  
}
```

```
*MESH_TFACELIST {  
  *MESH_TFACE 0 0 1 2  
  *MESH_TFACE 1 3 2 1  
  ...  
  *MESH_TFACE 253 154 157 152  
}
```



애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 기하 객체(Model)

```
*GEOMOBJECT {  
  *NODE_NAME "Bip01 Spine"  
  *NODE_PARENT "Bip01 Pelvis"  
  *NODE_TM { ... }  
  *MESH { ... }  
  *PROP_MOTI...  
  *PROP_CAST...  
  *PROP_RECV...  
  *WIREFRAME...  
}
```

```
*MESH {  
  *TIMEVALUE 0  
  *MESH_NUMVERTEX 129  
  *MESH_NUMFACES 126  
  *MESH_VERTEX_LIST { ... }  
  *MESH_FACE_LIST { ... }  
  *MESH_NUMTVERTEX 129  
  *MESH_TVERTLIST { ... }  
  *MESH_NUMTVFACES 126  
  *MESH_TFACELIST { ... }  
  *MESH_NORMALS { ... }  
}
```

```
*MESH_VERTEX_LIST {  
  *MESH_VERTEX 0 0.5078 20.4323 48.2923  
  *MESH_VERTEX 1 19.4899 20.4323 48.2923  
  ...  
  *MESH_VERTEX 127 0.5078 1.4503 48.2923  
  *MESH_VERTEX 128 13.9302 7.0100 48.2923  
}  
  
*MESH_FACE_LIST {  
  *MESH_FACE 0: A: 0 B: 1 C: 2 AB: 0 BC: 1 CA: 0 *MESH_SMOOTHING 2 *MESH_MTLID 0  
  ...  
  *MESH_FACE 125: A: 0 B: 8 C: 1 AB: 0 BC: 1 CA: 0 *MESH_SMOOTHING 2 *MESH_MTLID 0  
}
```

메쉬의 각 면에 대한 법선 벡터
면(삼각형)을 구성하는 각 정점의 법선 벡터

```
*MESH_NORMALS {  
  *MESH_FACENORMAL 0 -0.8373 -0.2615 -0.4801  
  *MESH_VERTEXNORMAL 0 -0.8447 -0.2852 -0.4530  
  *MESH_VERTEXNORMAL 1 -0.8373 -0.2615 -0.4801  
  *MESH_VERTEXNORMAL 11 -0.8594 -0.3507 -0.3720  
  ...  
  *MESH_FACENORMAL 125 -0.2083 0.7926 -0.5731  
  *MESH_VERTEXNORMAL 3 -0.2214 0.7598 -0.6112  
  *MESH_VERTEXNORMAL 46 -0.2795 0.6999 -0.6573  
  *MESH_VERTEXNORMAL 4 -0.2105 0.8042 -0.5559  
}
```

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 재질(Material)

```
*MATERIAL_LIST {  
*MATERIAL_COUNT 6  
*MATERIAL 0 { ... }  
*MATERIAL 1 { ... }  
*MATERIAL 2 { ... }  
*MATERIAL 3 { ... }  
*MATERIAL 4 { ... }  
*MATERIAL 5 { ... }  
}
```

Blinn, Phong, Wire, ...

```
*MATERIAL 0 {  
*MATERIAL_NAME "HAIR"  
*MATERIAL_CLASS "Standard"  
*MATERIAL_AMBIENT 0.3882 0.3098 0.2745  
*MATERIAL_DIFFUSE 0.3882 0.3098 0.2745  
*MATERIAL_SPECULAR 0.4627 0.2196 0.1725  
*MATERIAL_SHINE 0.0000  
*MATERIAL_SHINESTRENGTH 0.5300  
*MATERIAL_TRANSPARENCY 0.0000  
*MATERIAL_WIRESIZE 1.0000  
*MATERIAL_SHADING Blinn  
*MATERIAL_XP_FALLOFF 0.0000  
*MATERIAL_SELFILLUM 0.0000  
*MATERIAL_TWOSIDED 0  
*MATERIAL_FALLOFF In  
*MATERIAL_XP_TYPE Filter  
*MAP_DIFFUSE { ... }  
*MAP_SPECULAR { ... }  
*MAP_SELFILLUM { ... }  
*MAP_REFLECT { ... }  
*MAP_AMBIENT { ... }  
*MAP_OPACITY { ... }  
*MAP_BUMP { ... }  
}
```

```
*MAP_DIFFUSE {  
*MAP_NAME "Map #0"  
*MAP_CLASS "Bitmap"  
*MAP_SUBNO 1  
*MAP_AMOUNT 1.0000  
*BITMAP "Hair_d.jpg"  
*MAP_TYPE Screen  
*UVW_U_OFFSET 0.0000  
*UVW_V_OFFSET 0.0000  
*UVW_U_TILING 1.0000  
*UVW_V_TILING 1.0000  
*UVW_ANGLE 0.0000  
*UVW_BLUR 1.0000  
*UVW_BLUR_OFFSET 0.0000  
*UVW_NOISE_AMT 1.0000  
*UVW_NOISE_SIZE 1.0000  
*UVW_NOISE_LEVEL 1  
*UVW_NOISE_PHASE 0.0000  
*BITMAP_FILTER Pyramidal  
}
```

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 재질(Material)

```
*MATERIAL 1 {
```

```
  *MATERIAL_NAME "Eye"
```

```
  *MATERIAL_CLASS "Multi/Sub-Object"
```

```
  *MATERIAL_AMBIENT 0.5882 0.5882 0.5882
```

```
  *MATERIAL_DIFFUSE 0.5882 0.5882 0.5882
```

```
  *MATERIAL_SPECULAR 0.9000 0.9000 0.9000
```

```
  *MATERIAL_SHINE 0.7600
```

```
  *MATERIAL_SHINESTRENGTH 0.3900
```

```
  *MATERIAL_TRANSPARENCY 0.0000
```

```
  *MATERIAL_WIRESIZE 1.0000
```

```
*NUMSUBMTLS 4
```

```
*SUBMATERIAL 0 { ... }
```

```
*SUBMATERIAL 1 { ... }
```

```
*SUBMATERIAL 2 { ... }
```

```
*SUBMATERIAL 3 { ... }
```

```
}
```

```
*SUBMATERIAL 0 {
```

```
  *MATERIAL_NAME "Eyeliris"
```

```
  *MATERIAL_CLASS "Standard"
```

```
  *MATERIAL_AMBIENT 0.5882 0.5882 0.5882
```

```
  *MATERIAL_DIFFUSE 0.5882 0.5882 0.5882
```

```
  *MATERIAL_SPECULAR 0.9000 0.9000 0.9000
```

```
  *MATERIAL_SHINE 0.7600
```

```
  *MATERIAL_SHINESTRENGTH 0.3900
```

```
  *MATERIAL_TRANSPARENCY 0.0000
```

```
  *MATERIAL_WIRESIZE 1.0000
```

```
  *MATERIAL_SHADING Blinn
```

```
  *MATERIAL_XP_FALLOFF 0.0000
```

```
  *MATERIAL_SELFILLUM 0.0000
```

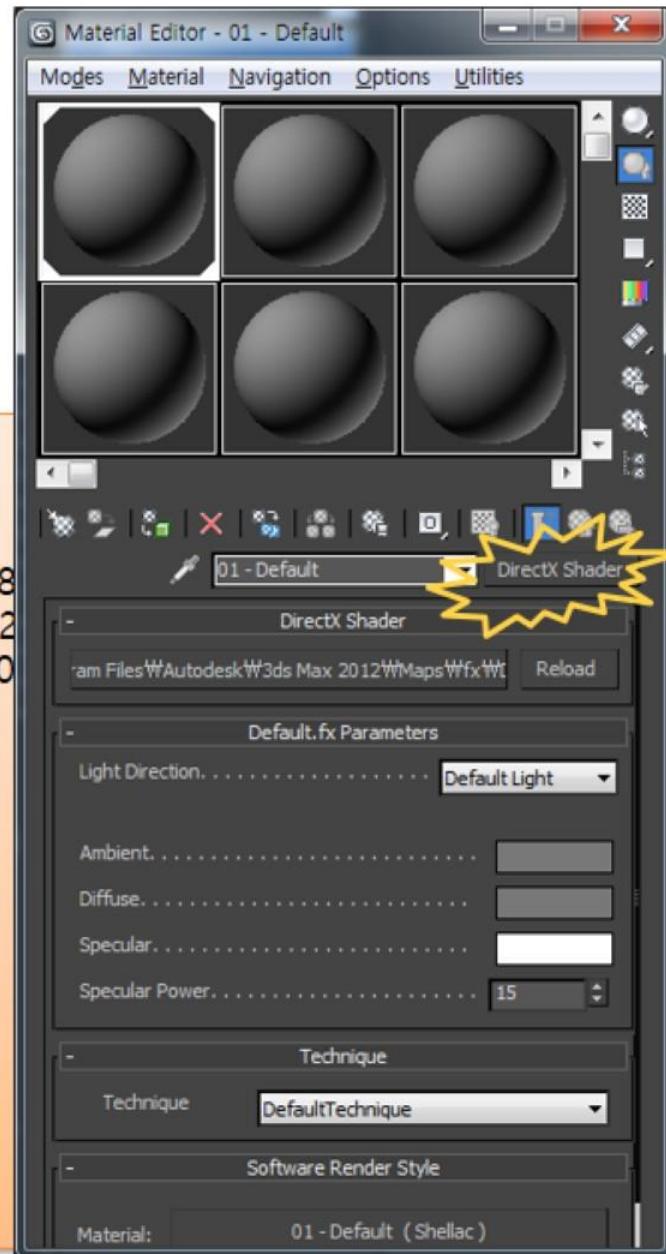
```
  *MATERIAL_FALLOFF In
```

```
  *MATERIAL_XP_TYPE Filter
```

```
  *MAP_DIFFUSE { ... }
```

```
  *MAP_OPACITY { ... }
```

```
}
```



애니메이션(Animation)

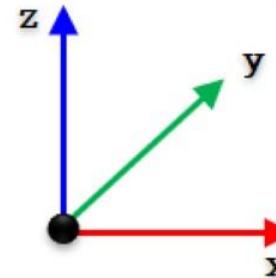
- ASE 파일(ASCII Scene Export File)

- 기하 객체(GeoObject)

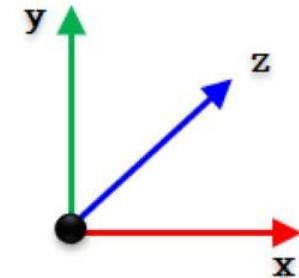
```
*NODE_TM {  
  *NODE_NAME "Bip01"  
  *INHERIT_POS 0 0 0  
  *INHERIT_ROT 0 0 0  
  *INHERIT_SCL 1 1 1  
  *TM_ROW0 0.0000 -1.0000 0.0000  
  *TM_ROW1 1.0000 0.0000 0.0000  
  *TM_ROW2 0.0000 -0.0000 1.0000  
  *TM_ROW3 0.5078 20.4323 640.0967  
  *TM_POS 0.5078 20.4323 640.0967  
  *TM_ROTAXIS -0.0000 -0.0000 1.0000  
  *TM_ROTANGLE 1.5708  
  *TM_SCALE 1.0000 1.0000 1.0000  
  *TM_SCALEAXIS 0.0000 0.0000 0.0000  
  *TM_SCALEAXISANG 0.0000  
}
```

- ① TM_ 행렬에서 y(2열)와 z(3열)를 바꿈
TM_ROW1과 TM_ROW2를 바꿈
- ② 모든 벡터의 y와 z를 바꿈

모든 변환 정보는 3DS Max에서 렌더링하기 위한 정보
회전 각도는 라디안(Radian)으로 저장되어 있음



3DS Max
오른손 좌표계(z-Up)



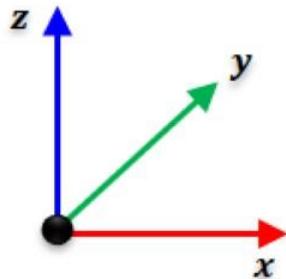
Direct3D
원손 좌표계(y-Up)

Right.x	Right.y	Right.z	0
Up.x	Up.y	Up.z	0
Look.x	Look.y	Look.z	0
Position.x	Position.y	Position.z	1

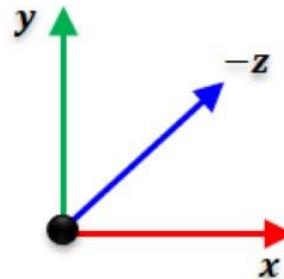
0.0000	0.0000	-1.0000	0
0.0000	1.0000	0.0000	0
1.0000	0.0000	0.0000	0
0.5078	640.0967	20.4323	1

애니메이션(Animation)

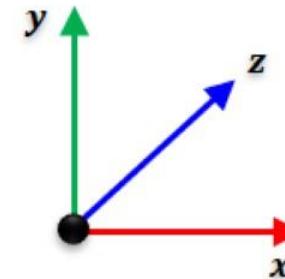
- 좌표계 변환(Coordinates Conversion)



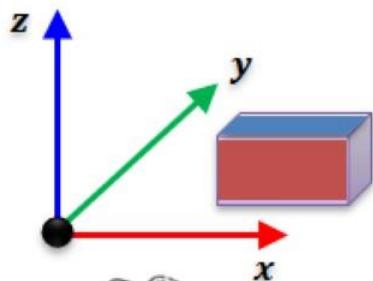
3DS Max: 오른손 좌표계(z-Up)



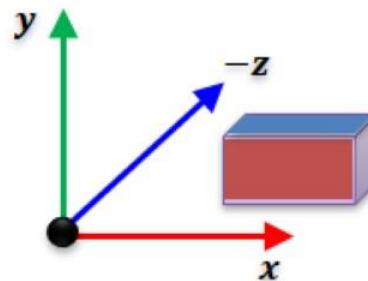
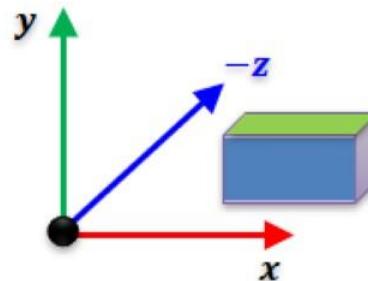
OpenGL: 오른손 좌표계(y-Up)



Direct3D: 원손 좌표계(y-Up)



① $(x, y, z) \rightarrow (x, z, -y)$
3DS Max Exporter: Flip YZ-axis

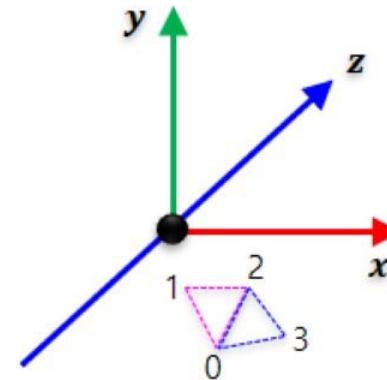
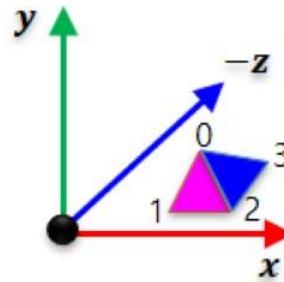
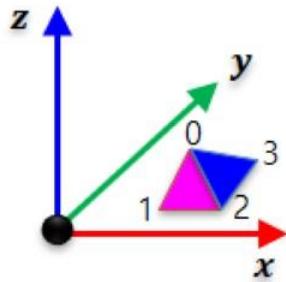


카메라는 좌우가 바뀐 모델을 보게됨
(xy-평면에 대하여 반사된 모델)
xy-평면 반사: ② $(x, y, z) \rightarrow (x, y, -z)$

$$(x, y, z) \rightarrow (x, z, -y) \rightarrow (x, z, y)$$

애니메이션(Animation)

- 좌표계 변환(Coordinates Conversion)



3DS Max: 오른손 좌표계(z-Up)

OpenGL: 오른손 좌표계(y-Up)

Direct3D: 왼손 좌표계(y-Up)

0	(3,2,5)	0
1	(2,1,2)	3
2	(6,1,2)	1
3	(8,4,3)	0
		1
		2
		0
		2
		3

0	(3,5,-2)	0
1	(2,2,-1)	3
2	(6,2,-1)	1
3	(8,3,-4)	0
		1
		2
		0
		2
		3

$$(x, y, z) \rightarrow (x, z, -y)$$

0	(3,5,2)	0
1	(2,2,1)	3
2	(6,2,1)	1
3	(8,3,4)	0
		1
		2
		0
		2
		3

$$(x', y', z') \rightarrow (x', y', -z')$$

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 애니메이션(Animation)

```
*GEOMOBJECT {  
  *NODE_NAME "Bip01 Spine"  
  *NODE_PARENT "Bip01 Pelvis"  
  *NODE_TM { ... }  
  *MESH { ... }  
  *PROP_MOTIONBLUR 0  
  *PROP_CASTSHADOW 1  
  *PROP_RECVSHADOW 1  
  *TM_ANIMATION { ... }  
  *MESH_SOFTSKIN { ... }  
  *MATERIAL_REF 0
```

```
*TM_ANIMATION {  
  *NODE_NAME "Bip01 Spine"  
  *CONTROL_POS_TRACK { ... }  
  *CONTROL_ROT_TRACK { ... }  
  *CONTROL_SCALE_TRACK { ... }  
}
```

```
*CONTROL_POS_BEZIER { ... }  
*CONTROL_ROT_BEZIER { ... }  
*CONTROL_SCALE_BEZIER { ... }  
*CONTROL_COLOR_BEZIER { ... }  
*CONTROL_POS_TCB { ... }  
*CONTROL_ROT_TCB { ... }  
*CONTROL_SCALE_TCB { ... }
```

로컬 좌표계

```
*CONTROL_ROT_TRACK {
```

```
*CONTROL_ROT_SAMPLE 0 -0.5184 -0.8461 0.1236 3.4820  
*CONTROL_ROT_SAMPLE 800 -0.0453 -0.9932 0.1070 0.1659  
*CONTROL_ROT_SAMPLE 2400 0.0063 -0.9998 -0.0213 0.4630  
*CONTROL_ROT_SAMPLE 5600 -0.4231 -0.7316 -0.5345 0.1638  
*CONTROL_ROT_SAMPLE 6400 -0.9887 0.0202 -0.1482 5.2817  
*CONTROL_ROT_SAMPLE 7200 -0.9620 0.1947 0.1912 5.2897  
*CONTROL_ROT_SAMPLE 8000 -0.8769 0.4803 0.0206 5.4694  
*CONTROL_ROT_SAMPLE 8800 0.9918 0.0379 0.1220 0.6817
```

```
*CONTROL_POS_TRACK {
```

```
*CONTROL_POS_SAMPLE 0 62.9285 103.9948 -12.9344  
*CONTROL_POS_SAMPLE 800 61.3391 106.1076 -9.8679  
*CONTROL_POS_SAMPLE 1600 57.3177 109.0960 2.9295  
*CONTROL_POS_SAMPLE 2400 58.2219 109.4368 11.2242  
*CONTROL_POS_SAMPLE 5600 53.7481 158.7482 35.2592  
*CONTROL_POS_SAMPLE 6400 54.2626 145.0902 24.9916  
*CONTROL_POS_SAMPLE 7200 56.2722 100.9487 21.5322  
*CONTROL_POS_SAMPLE 8000 50.3161 54.8600 27.0264
```

Track: 샘플링(Sampling)

```
*CONTROL_POS_LINEAR { ... }  
*CONTROL_ROT_LINEAR { ... }  
*CONTROL_SCALE_LINEAR { ... }
```

회전은 이전 틱과 상대적인 변화량
오일러 회전(회전축, 회전각)

```
*CONTROL_POS_BEZIER {
```

```
*CONTROL_BEZIER_POS_KEY ...
```

```
... *CONTROL_ROT_TCB {
```

```
*CONTROL_TCB_ROT_KEY ...  
...
```

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 애니메이션 보간(TCB Interpolation)

키프레임 데이터: $\{(t_i, P_i, T_i^I, T_i^O)\}$

T_i^I : 입력 접선 벡터(Incoming Tangent Vector)

T_i^O : 출력 접선 벡터(Outgoing Tangent Vector)

Kochanek-Bartels Spline

TCB(Tension, Continuity, Bias)

$(t_i, P_i, T_i^I, T_i^O), (t_{i+1}, P_{i+1}, T_{i+1}^I, T_{i+1}^O)$

$$P_i(t) = A_i + \left(\frac{t - t_i}{\Delta_i}\right) B_i + \left(\frac{t - t_i}{\Delta_i}\right)^2 C_i + \left(\frac{t - t_i}{\Delta_i}\right)^3 D_i$$

$$\Delta_i = t_{i+1} - t_i$$

$$A_i = P_i$$

$$A_i + B_i + C_i + D_i = P_{i+1}$$

$$B_i = \Delta_i T_i^O$$

$$B_i + 2C_i + 3D_i = \Delta_i T_{i+1}^I$$

$$P_i(t_i) = P_i$$

$$P_i(t_{i+1}) = P_{i+1}$$

$$P_i'(t_i) = T_i^O$$

$$P_i'(t_{i+1}) = T_{i+1}^I$$

$$A_i = P_i$$

$$B_i = \Delta_i T_i^O$$

$$C_i = 3(P_{i+1} - P_i) - \Delta_i(2T_i^O + T_{i+1}^I)$$

$$D_i = -2(P_{i+1} - P_i) + \Delta_i(T_i^O + T_{i+1}^I)$$

τ_i : Tension, γ_i : Continuity, β_i : Bias

$$(-1 \leq \tau_i \leq 1) \Rightarrow T_i^I = T_i^O = \frac{(1 - \tau_i)}{2} \{(P_{i+1} - P_i) + (P_i - P_{i-1})\}$$

$$(-1 \leq \gamma_i \leq 1) \Rightarrow T_i^I = \frac{(1 + \gamma_i)}{2} (P_{i+1} - P_i) + \frac{(1 - \gamma_i)}{2} (P_i - P_{i-1}), T_i^O = \frac{(1 - \gamma_i)}{2} (P_{i+1} - P_i) + \frac{(1 + \gamma_i)}{2} (P_i - P_{i-1})$$

$$(-1 \leq \beta_i \leq 1) \Rightarrow T_i^I = T_i^O = \frac{(1 - \beta_i)}{2} (P_{i+1} - P_i) + \frac{(1 + \beta_i)}{2} (P_i - P_{i-1})$$

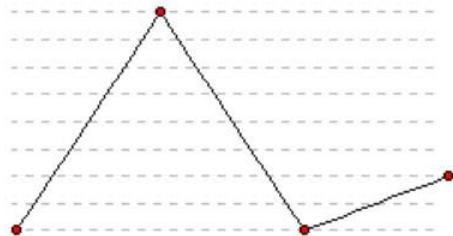
$$T_i^I = \frac{(1 - \tau_i)(1 - \beta_i)(1 + \gamma_i)}{2} (P_{i+1} - P_i) + \frac{(1 - \tau_i)(1 + \beta_i)(1 - \gamma_i)}{2} (P_i - P_{i-1})$$

$$T_i^O = \frac{(1 - \tau_i)(1 - \beta_i)(1 - \gamma_i)}{2} (P_{i+1} - P_i) + \frac{(1 - \tau_i)(1 + \beta_i)(1 + \gamma_i)}{2} (P_i - P_{i-1})$$

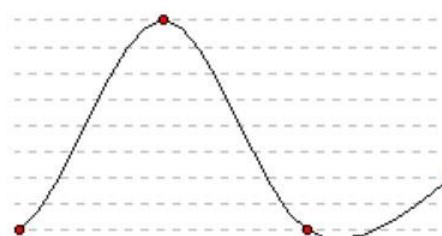
애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 애니메이션 보간(TCB Interpolation)

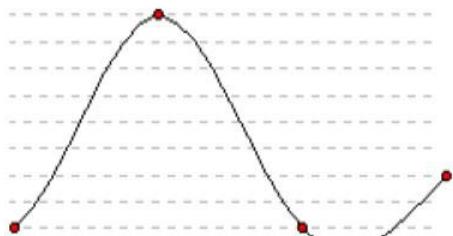


선형 보간(Linear Interpolation)



TCB 보간(TCB interpolation)

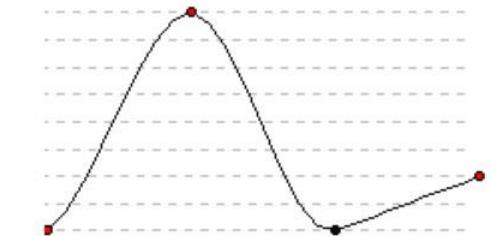
TCB(Tension, Continuity, Bias)



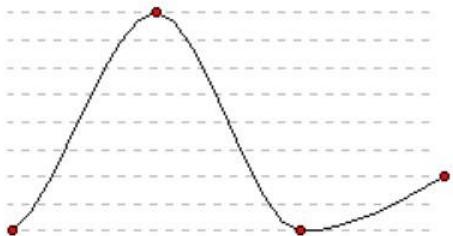
세 번째 키: Tension = 0



세 번째 키: Continuity = 0



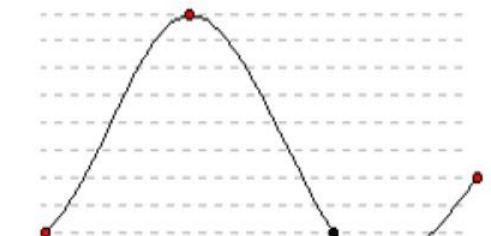
세 번째 키: Bias = 0



세 번째 키: Tension = 1.0



세 번째 키: Continuity = 1.0



세 번째 키: Bias = 1.0

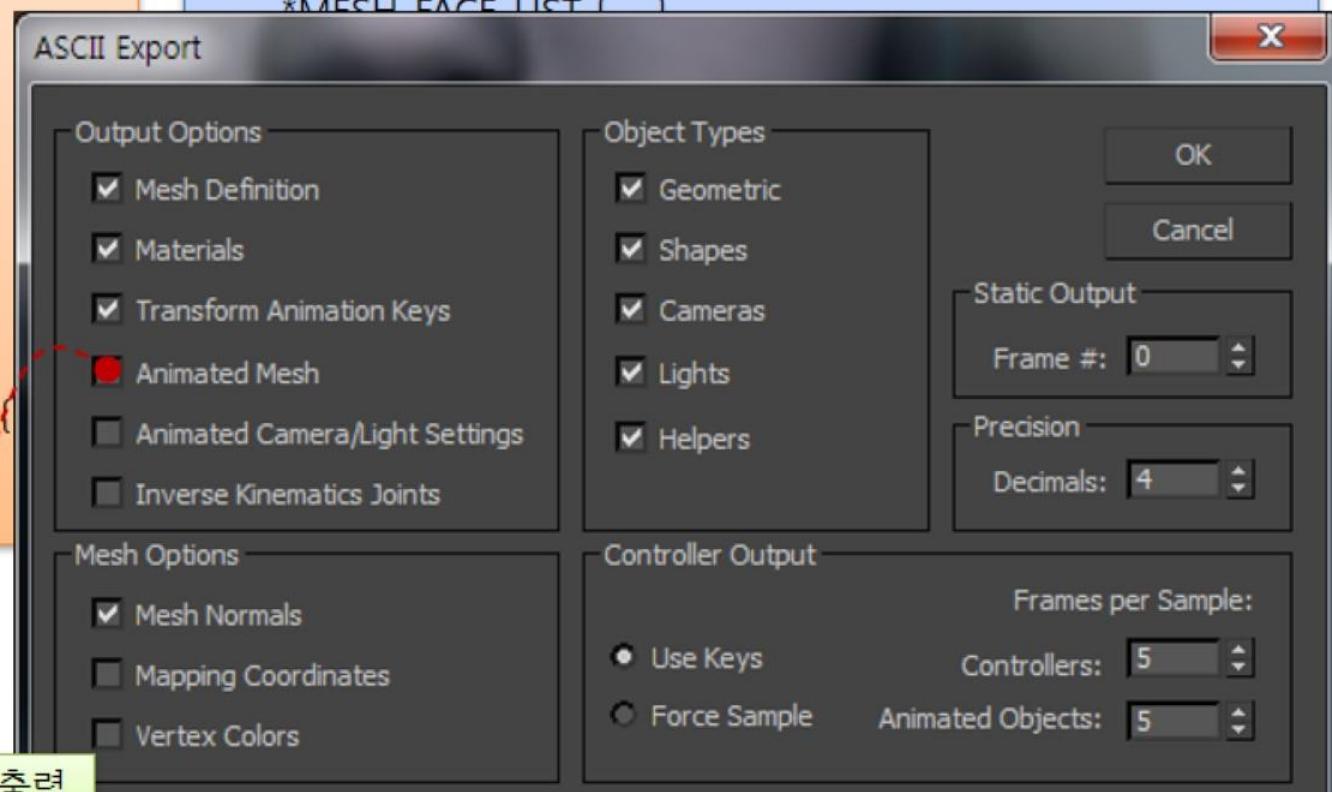
애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 애니메이션 메쉬(Animated Mesh)

```
*MESH {  
*TIMEVALUE 0  
*MESH_NUMVERTEX #  
*MESH_NUMTVERTEX #  
*MESH_NUMCVERTEX #  
*MESH_NUMFACES #  
*MESH_NUMTVFACES #  
*MESH_NUMCVFACES #  
*MESH_VERTEX_LIST { ... }  
*MESH_FACE_LIST { ... }  
*MESH_TVERTLIST { ... }  
*MESH_TFACELIST { ... }  
*MESH_CVERTLIST { ... }  
*MESH_CFACELIST { ... }  
*MESH_NORMALS { ... }  
*MESH_MAPPINGCHANNEL  
*MESH_ANIMATION { ... }  
}
```

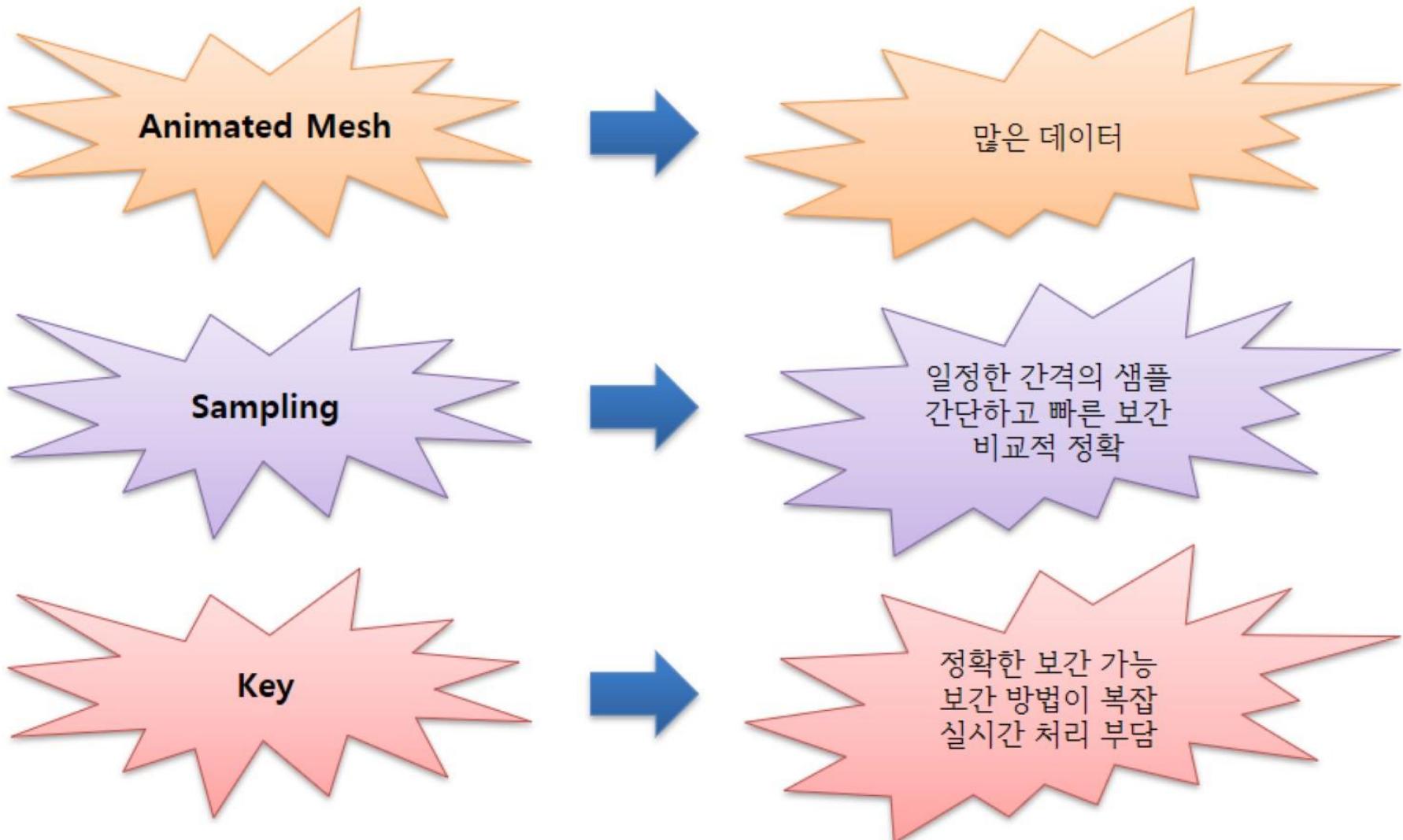
```
*MESH_ANIMATION {  
*MESH {  
*TIMEVALUE 0  
*MESH_NUMVERTEX 388  
*MESH_VERTEX_LIST { ... }  
*MESH_NUMFACES 724  
*MESH_FACE_LIST { ... }  
}
```



애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 애니메이션 보간



애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 애니메이션 샘플링과 출력(Animation Sample Output)

- ① 샘플링 시간 t 에 대한 변환 행렬 M_w 를 샘플링
 - ② 로컬 변환 행렬 $M_i = M_w * (M_p)^{-1}$
 - ③ M_i 를 위치(Position: P_i), 회전(Rotation: R_i), 크기(Scale: S_i) 변환 행렬로 분할
 - ④ 샘플 값이 이전 시간 $(t - 1)$ 과 같으면 샘플 값을 출력하지 않음
 - ⑤ 회전의 경우 이전 시간 $(t - 1)$ 에 **상대적인** 회전의 양을 출력 $R_i = R_i * (R_{i-1})^{-1}$
 - ⑥ 샘플 값이 이전 시간 $(t - 1)$ 의 샘플 값과 같으면 샘플 값을 출력하지 않음
- P_i, R_i, S_i 변환 행렬이 단위 행렬



로컬 좌표계

(회전축, 회전각)

회전 샘플 값을 사용하여 실제 애니메이션을 하려면 각 샘플 시간의 회전 샘플 값은 **누적**하여 저장해야 한다.
회전축과 회전 각도는 쿼터니언으로 변경하여 저장한다.

크기 변환의 회전축과 회전 각도는 쿼터니언으로 변경하여 저장한다.

어떤 샘플 키(틱: Tick) 데이터가 출력되지 않을 수 있음(이전과 같을 경우)

애니메이션 데이터를 로드할 때 이전 샘플 키 데이터를 중복시키거나 다른 방법으로 해결해야 한다.

각 프레임의 애니메이션의 길이가 다를 수 있음

각 프레임의 샘플 키 데이터의 길이가 같도록 해야 함(길이가 짧은 프레임의 마지막 키를 복사해서 반복)

```
*CONTROL_POS_TRACK {
  *CONTROL_POS_SAMPLE 0 62.9285 103.
  *CONTROL_POS_SAMPLE 800 61.3391 103.
  *CONTROL_POS_SAMPLE 1600 57.3177
  ...
  *CONTROL_POS_SAMPLE 7200 56.2722
  *CONTROL_POS_SAMPLE 8000 50.3161
}
```

```
*CONTROL_ROT_TRACK {
  *CONTROL_ROT_SAMPLE 0 -0.5184 -0.8461 0.1236 3.4820
  *CONTROL_ROT_SAMPLE 800 -0.0453 -0.9932 0.1070 0.1659
  *CONTROL_ROT_SAMPLE 2400 0.0063 -0.9998 -0.0213 0.4630
  ...
  *CONTROL_ROT_SAMPLE 8800 0.9918 0.0379 0.1220 0.6817
}
```

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 애니메이션 렌더링

계층 구조에서 프레임 F_i 의 애니메이션 데이터가 없으면 프레임의 변환 행렬 M_i 를 사용하여 렌더링

$$M_w = M_i * M_p$$

계층 구조에서 프레임 F_i 의 애니메이션 데이터가 있으면 프레임의 애니메이션 행렬 A_i 를 생성하여 렌더링

$$M_w = A_i * M_p \quad A_i = S_i(t) * R_i(t) * T_i(t)$$

각 프레임의 샘플 키 데이터가 있으면 보간을 사용하여 해당하는 $S_i(t)$, $R_i(t)$, $T_i(t)$ 를 계산한다.

각 프레임의 샘플 키 데이터가 없으면 프레임의 변환 행렬 M_i 에서 해당하는 $S_i(t)$, $R_i(t)$, $T_i(t)$ 를 생성한다.

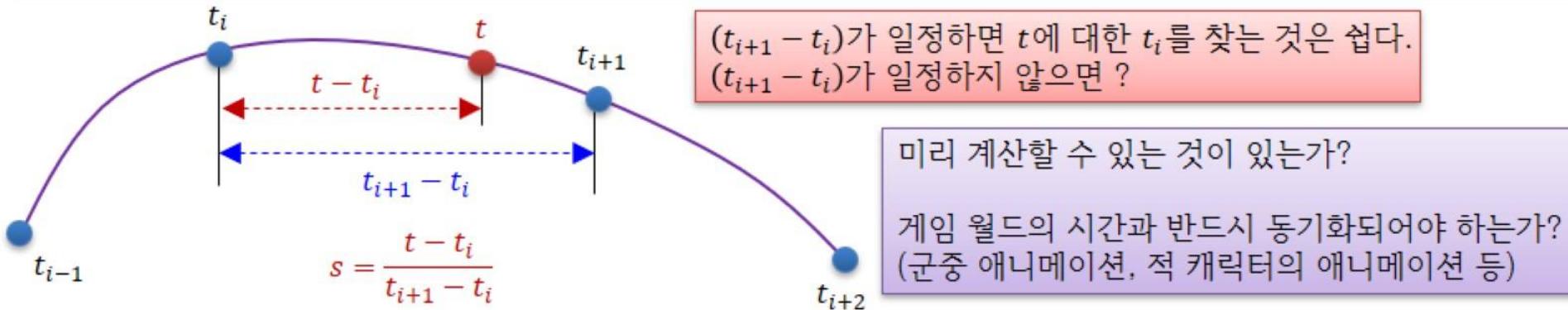
$R_i(t)$: t_i 의 회전 쿼터니언과 t_{i+1} 의 회전 쿼터니언을 구면 보간하고 그 결과를 행렬로 변환하여 생성한다.

$T_i(t)$: t_i 의 위치 벡터와 t_{i+1} 의 위치 벡터를 선형 보간하고 그 결과를 행렬로 변환하여 생성한다.

$S(t)$: t_i 의 스케일 벡터와 t_{i+1} 의 스케일 벡터를 선형 보간하고 그 결과를 행렬로 변환하여 생성한다.

$S_R(t)$: t_i 의 스케일 쿼터니언과 t_{i+1} 의 스케일 쿼터니언을 구면 보간하고 그 결과를 행렬로 변환하여 생성한다.

$$S_i(t) = S_R(t)^{-1} * S(t) * S_R(t)$$



XMVECTOR **XMVectorLerp**(XMVECTOR vV0, XMVECTOR vV1, float t);

XMVECTOR **XMQuaternionSlerp**(XMVECTOR qQ0, XMVECTOR qQ1, float t);

애니메이션(Animation)

- ASE 파일(ASCII Scene Export File)

- 스키니드 메쉬(Skinned Mesh)

```
*MESH {  
*TIMEVALUE 0  
*MESH_NUMVERTEX #  
*MESH_NUMTVERTEX #  
*MESH_NUMCVERTEX #  
*MESH_NUMFACES #  
*MESH_NUMTVFACES #  
*MESH_NUMCVFACES #  
*MESH_VERTEX_LIST { ... }  
*MESH_FACE_LIST { ... }  
*MESH_TVERTEXLIST { ... }  
*MESH_TFACELIST { ... }  
*MESH_CVERTEXLIST { ... }  
*MESH_CFACELIST { ... }  
*MESH_NORMALS { ... }  
*MESH_MAPPINGCHANNEL { ... }  
*MESH_ANIMATION { ... }  
*MESH_WEIGHTS { ... }  
}
```

```
*MESH_WEIGHTS {  
*MESH_NUMVERTEX 3  
*MESH_NUMBONE 4  
*MESH_BONE_LIST  
{  
*MESH_BONE_NAME 0 "Bone01"  
...  
*MESH_BONE_NAME 3 "Bone03"  
}  
*MESH_BONE_VERTEX_LIST  
{  
*MESH_BONE_VERTEX 0 0.2000 1 0.3000 2 0.5000 3 0.0000  
*MESH_BONE_VERTEX 1 0.2000 1 0.3000 2 0.3000 3 0.2000  
*MESH_BONE_VERTEX 2 0.3000 1 0.3000 2 0.1000 3 0.3000  
}  
}
```



기본 ASE Exporter는 스키니드
정보를 출력하지 않음

애니메이션(Animation)

- 3Ds Max IGameExporter

```
<?xml version="1.0" encoding="utf-8" ?>
<IGame xmlns="urn:maxxxml" Version="2.0" Date="Tue Dec 04 15:12:32 2012">
  <SceneInfo FileName="Sman.max" StartFrame="0" EndFrame="53" FrameRate="30" TicksPerFrame="160" CoordinateSystem="directx" RotationFormat="Quaternion" ObjectSpace="false"></SceneInfo>
  <MaterialList Count="3">
    <Material Index="0" Name="Material::lambert23_Texture::file8" Class="Standard">
      <Diffuse Value="0.000000 0.000000 0.000000"></Diffuse>
      <Ambient Value="0.000000 0.000000 0.000000"></Ambient>
      <Specular Value="0.200000 0.200000 0.200000"></Specular>
      <Glossiness Value="0.050000"></Glossiness>
      <Opacity Value="1.000000"></Opacity>
      <SpecularLevel Value="0.700000"></SpecularLevel>
      <TextureMaps Count="1">
        <Texture Index="0" Name="Texture::file8" Slot="Ambient">
          <BitmapTexture Filename="ch_m_03face.BMP">
            <ClipU Value="0.000000"></ClipU>
            <ClipV Value="0.000000"></ClipV>
            <ClipW Value="1.000000"></ClipW>
            <ClipH Value="1.000000"></ClipH>
          </BitmapTexture>
        </Texture>
      </TextureMaps>
    </Material>
    ...
  </MaterialList>
</IGame>
```

애니메이션(Animation)

- 3Ds Max IGameExporter

```
<Node Name="Fbx_Root" NodeID="172" WireframeColor="083d8aff" NodeType="Helper">
  <NodeTM>
    ...
  </NodeTM>
<Node Name="Root" NodeID="1" ParentID="172" WireframeColor="860606ff" NodeType="Helper">
  <NodeTM>
    ...
  </NodeTM>
  <Node Name="Hips" NodeID="2" ParentID="1" WireframeColor="aebacbff" NodeType="Bone">
    <NodeTM>
      ...
    </NodeTM>
    <Mesh>
      ...
    </Mesh>
    <TMController>
      <PRSCController>
        ...
      </PRSCController>
    </TMController>
    <Node Name="Spine" NodeID="3" ParentID="2" WireframeColor="aebacbff" NodeType="Bone">
      ...
    </Node>
  </Node>
</Node>
</Node>
```

애니메이션(Animation)

- 3Ds Max IGameExporter

```
<NodeTM>
  <Translation>
    0.000000 0.000000 0.000000
  </Translation>
  <Rotation>
    0.707107 -0.000000 -0.000000 0.707107
  </Rotation>
  <Scale>
    1.000000 1.000000 1.000000 -0.000000 -0.000000 -0.000000 1.000000
  </Scale>
</NodeTM>
```

```
<FaceVertices>
  4 5 7
  ...
  8 0 3
</FaceVertices>
```

```
<FaceNormals>
  0 2 1
  ...
  29 31 30
</FaceNormals>
```

```
<MaterialIDs>
  0
  ...
  5
</MaterialIDs>
```

```
<SmoothingGroups>
  1
  ...
  256
</SmoothingGroups>
```

```
<Vertices Count="9">
  56.451309 93.642014 11.683585
  ...
  56.635628 93.595306 11.640996
</Vertices>
```

```
<Normals Count="32">
  -0.420455 0.893451 -0.157998
  ...
  -0.276073 -0.948655 -0.154392
</Normals>
```

```
<EdgeVisibility>
  1 0 1
  ...
  1 1 1
</EdgeVisibility>
```

```
<Mesh>
  <Vertices Count="9">
    ...
  </Vertices>
  <Normals Count="32">
    ...
  </Normals>
  <Faces Count="14">
    <FaceVertices>
      ...
    </FaceVertices>
    <FaceNormals>
      ...
    </FaceNormals>
    <MaterialIDs>
      ...
    </MaterialIDs>
    <SmoothingGroups>
      ...
    </SmoothingGroups>
    <EdgeVisibility>
      ...
    </EdgeVisibility>
  </Faces>
</Mesh>
```

애니메이션(Animation)

- 3Ds Max IGameExporter

```
<TMController>
  <PRSController>
    <Position Value="56.635628 93.595306 11.640996">
      <PositionXYZController>
        ...
        </PositionXYZController>
      </Position>
    <EulerController>
      <EulerX Count="107" Type="Bezier">
        ...
        </EulerX>
      <EulerY Count="107" Type="Bezier">
        ...
        </EulerY>
      <EulerZ Count="107" Type="Bezier">
        ...
        </EulerZ>
      </EulerController>
    </PRSController>
  </TMController>

  <EulerX Count="107" Type="Bezier">
    0 0.391017
    ...
    8480 -1.213951
  </EulerX>

  <PositionXYZController>
    <X Value="56.635628">
      <BezierFloatController count="54">
        <Key time="0" value="62.928478"></Key>
        <Key time="160" value="62.770702"></Key>
        ...
        <Key time="8320" value="44.623100"></Key>
        <Key time="8480" value="43.363914"></Key>
      </BezierFloatController>
    </X>
    <Y Value="93.595306">
      <BezierFloatController count="54">
        <Key time="0" value="103.994789"></Key>
        ...
        <Key time="8480" value="31.223736"></Key>
      </BezierFloatController>
    </Y>
    <Z Value="11.640996">
      <BezierFloatController count="54">
        <Key time="0" value="-12.934444"></Key>
        ...
        <Key time="8480" value="21.554623"></Key>
      </BezierFloatController>
    </Z>
  </PositionXYZController>
```

Bezier Key
Linear Key
Constraint
List
Sample

TCB Key
Linear Key
Bezier Key
Constraint
List
Euler
Sample

TicksPerFrame="160"

애니메이션(Animation)

- 3Ds Max IGameExporter

```
<TMController>
  <PRSController>
    <Position KeyCount="14" Type="FullSampled" SampleRate="4">
      ...
      </Position>
    <Rotation KeyCount="14" Type="FullSampled" SampleRate="4">
      ...
      </Rotation>
  </PRSController>
</TMController>
```

```
<Position KeyCount="14" Type="FullSampled" SampleRate="4">
  <Sample frame="0">
    62.928478 -12.934444 103.994789
  </Sample>
  <Sample frame="640">
    61.888680 -11.028728 105.478966
  </Sample>
  <Sample frame="1280">
    58.780556 -3.626082 108.163116
  </Sample>
  ...
  <Sample frame="7680">
    56.437634 27.03973
  </Sample>
  <Sample frame="8320">
    44.623100 22.49315
  </Sample>
</Position>
```

```
<Rotation KeyCount="14" Type="FullSampled" SampleRate="4">
  <Sample frame="0">
    0.510956 -0.121856 0.833894 0.169400
  </Sample>
  ...
  <Sample frame="8320">
    -0.099564 0.193720 0.721450 0.657320
  </Sample>
</Rotation>
```

TicksPerFrame="160"

애니메이션(Animation)

- 3Ds Max IGameExporter

```
<Node Name="Gum" NodeID="170" ParentID="164" MaterialIndex="2" NodeType="Mesh">
  <NodeTM>
    ...
  </NodeTM>
  <Mesh>
    ...
  </Mesh>
  <Modifiers Count="1">
    <Modifier modName="Skin" IsSkin="true" SkinType="MaxSkin">
      <Skin VertexID="0" Type="Rigid">
        <Bone BonelD="108"></Bone>
      </Skin>
      ...
      <Skin VertexID="39" Type="Blended">
        <Bone BonelD="108" Weight="0.522268"></Bone>
        <Bone BonelD="109" Weight="0.477732"></Bone>
      </Skin>
      ...
      <Skin VertexID="78" Type="Blended">
        <Bone BonelD="108" Weight="0.100000"></Bone>
        <Bone BonelD="107" Weight="0.898112"></Bone>
        <Bone BonelD="109" Weight="0.001888"></Bone>
      </Skin>
    </Modifier>
  </Node>
```

"Physique"
"MaxSkin"

```
<Mesh>
  <Vertices Count="79">
    ...
  </Vertices>
  <Normals Count="79">
    ...
  </Normals>
  <Faces Count="146">
    <FaceVertices>
      ...
    </FaceVertices>
    <FaceNormals>
      ...
    </FaceNormals>
    <MaterialIDs>
      ...
    </MaterialIDs>
    <SmoothingGroups>
      ...
    </SmoothingGroups>
    <EdgeVisibility>
      ...
    </EdgeVisibility>
  </Faces>
</Mesh>
```

애니메이션(Animation)

- 3Ds Max IGameExporter

<NodeTM>

<Translation>
56.635628 93.595306 11.640996

</Translation>

<Rotation>
-0.241516 -0.503487 -0.675817 0.481085

</Rotation>

<Scale>

0.900000 0.900000 0.900000 -0.000000 0.017647 -0.061773 0.997934

</Scale>

</NodeTM>

<Vertices Count="9">
56.451309 93.642014 11.683585
56.633759 93.742844 11.768262
56.725346 93.749626 11.562858
56.542896 93.648788 11.478181
56.527325 93.790947 11.611484
56.545570 93.801033 11.619951
56.554729 93.801712 11.599410
56.536484 93.791626 11.590942
56.635628 93.595306 11.640996
</Vertices>

<Vertices Count="9">
0.125000 0.125000 -0.125000
0.125000 0.125000 0.125000
0.125000 -0.125000 0.125000
0.125000 -0.125000 -0.125000
0.250000 0.012500 -0.012500
0.250000 0.012500 0.012500
0.250000 -0.012500 0.012500
0.250000 -0.012500 -0.012500
0.000000 0.000000 0.000000
</Vertices>

<Normals Count="32">
-0.420455 0.893451 -0.157998
...
-0.276073 -0.948655 -0.154392
</Normals>

<Normals Count="32">
1.000000 0.000000 0.000000
...
-0.707107 0.000000 -0.707107
</Normals>

Object Space

애니메이션(Animation)

- 스ки닝 캐릭터 애니메이션(Skinned Character Animation)
 - 샘플 데이터(애니메이션)

```
AnimationSet Walk {
```

```
    Bone00 #Keyframes: 76 { // (ID, KeyTime, Translation, Scale, Rotation)
```

```
        00 0.0000000 0.037563 37.46099 -2.230549 1 1 1 -0.500000 -0.499999 -0.499999 0.500000
```

```
        ...
```

```
        74 1.2333333 0.037563 37.46099 -2.230549 1 1 1 -0.500000 -0.499999 -0.499999 0.500000
```

```
        75 1.2500000 0.037563 37.46099 -2.230549 1 1 1 -0.500000 -0.499999 -0.499999 0.500000
```

```
}
```

```
...
```

```
    Bone57 #Keyframes: 76 {
```

```
        00 0.0000000 5.250221 6.273849 -3.863562 1 1 1 -3.783569
```

```
        ...
```

```
        74 1.2333333 5.250221 6.273850 -2.324624 1 1 1 -1.384446 -9.204851 -0.791195 0.611563
```

```
        75 1.2500000 5.250221 6.273849 -1.191340 1 1 1 -7.032388 -6.862930 -0.791195 0.611563
```

```
}
```

```
}
```

```
// 오프셋 행렬
```

```
00 1.33158 1 4.1688 0 -5.55111 -4.1688 1 0 1 -1.33158 8.41241 0 -2.23054 -0.0375604 -37.4609 1
```

```
...
```

```
57 4.16881 2.757707 1 0 1 -2.25986 -4.16881 0 2.22044 1 -2.75770 0 -37.4609 -1.27468 -0.03755 1
```

```
// 서브셋(ID, vStart, Vertices, fStart, Faces)
```

```
SubsetID: 0 0 3915 0 7230
```

```
...
```

```
SubsetID: 4 13474 274 22065 442
```

```
// 프레임 계층 구조(부모 인덱스)
```

```
-1 0 1 2 3 4 5 6 7 7 7 7 6 12 ... 2 54 55 56
```

```
// 정점 버퍼(Position[3], Normal[3], Texture[2], Weights[4], Bone-Indices[4])
```

```
00000 -0.99834 67.88861 4.35496 -0.53684 0.77151 0.34141 0.69536 0.99095 0.74703 0.25296 0 0 7 9 0 0
```

```
00001 -0.89309 67.90220 4.38819 -0.44193 0.54222 0.71462 0.69473 0.98309 0.73822 0.26177 0 0 7 9 0 0
```

```
...
```

```
13747 -0.85812 67.91822 4.40049 -0.32190 0.89450 0.31020 0.69758 0.98264 1.00000 0 0 0 57 0 0 0
```

애니메이션(Animation)

- 스ки닝 캐릭터 애니메이션(Skinned Character Animation)

- 예제

```
class CVertex
{
    XMFFLOAT3 m_xmf3Position;
    XMFFLOAT3 m_xmf3Normal;
    XMFFLOAT2 m_xmf3Texture;
    XMFFLOAT3 m_xmf3Weights;
    BYTE m_pbyBoneIndices[4];
};
```

1.0f - (m_xmf3Weights.x + m_xmf3Weights.y + m_xmf3Weights.z)

하나의 정점에 영향을 주는 뼈대의 최대 개수는 4개이면 충분하다고 알려져 있음

0	프레임 변환 행렬 0
1	프레임 변환 행렬 1
2	프레임 변환 행렬 2
...	...
n-1	프레임 변환 행렬 (n-1)

```
//정점 버퍼(Position[3], Normal[3], Texture[2], Weights[4], Bone-Indices[4])
0000 -0.99834 67.88861 4.35496 -0.53684 0.77151 0.34141 0.69536 0.99095 0.74703 0.25296 0 0 7 9 0 0
0001 -0.89309 67.90220 4.38819 -0.44193 0.54222 0.71462 0.69473 0.98309 0.73822 0.26177 0 0 7 9 0 0
...
3915 -0.85812 67.91822 4.40049 -0.32190 0.89450 0.31020 0.69758 0.98264 1.00000 0 0 0 57 0 0 0
```

```
D3D12_INPUT_ELEMENT_DESC d3dInputLayout[] =
{
    { "POSITION", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, 0, D3D12_INPUT_PER_VERTEX_DATA, 0 },
    { "NORMAL", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, 12, D3D12_INPUT_PER_VERTEX_DATA, 0 },
    { "TEXCOORD", 0, DXGI_FORMAT_R32G32_FLOAT, 0, 24, D3D12_INPUT_PER_VERTEX_DATA, 0 },
    { "WEIGHTS", 0, DXGI_FORMAT_R32G32B32_FLOAT, 0, 32, D3D12_INPUT_PER_VERTEX_DATA, 0 },
    { "BONEINDICES", 0, DXGI_FORMAT_R8G8B8A8_UINT, 0, 48, D3D12_INPUT_PER_VERTEX_DATA, 0 }
};
```

애니메이션(Animation)

- 스ки닝 캐릭터 애니메이션(Skinned Character Animation)

```
VS_OUTPUT VS(VS_INPUT input)
{
    VS_OUTPUT output;
    float fWeights[4] = {0.0f, 0.0f, 0.0f, 0.0f};
    fWeights[0] = input.boneWeights.x;
    fWeights[1] = input.boneWeights.y;
    fWeights[2] = input.boneWeights.z;
    fWeights[3] = 1.0f - fWeights[0] - fWeights[1] - fWeights[2];
    float3 position = float3(0.0f, 0.0f, 0.0f);
    float3 normal = float3(0.0f, 0.0f, 0.0f);
    for (int i = 0; i < 4; i++)
    {
        position += fWeights[i] * mul(float4(input.position, 1.0f), gmtxBoneTransforms[input.boneIndices[i]]).xyz;
        normal += fWeights[i] * mul(input.normal, (float3x3)gmtxBoneTransforms[input.boneIndices[i]]);
    }
    output.texcoord0 = input.texcoord0;
    output.positionW = mul(float4(position, 1.0f), gmtxWorld).xyz;
    output.normalW = mul(normal, (float3x3)gmtxInvWorld);
    matrix mtxWorldViewProjection = mul(gmtxWorld, gmtxView);
    mtxWorldViewProjection = mul(mtxWorldViewProjection, gmtxProjection);
    output.position = mul(float4(input.position, 1.0f), mtxWorldViewProjection);

    return(output);
}
```

```
struct VS_INPUT {
    float3 position : POSITION;
    float3 normal : NORMAL;
    float2 texcoord0 : TEXCOORD;
    float3 boneWeights : WEIGHTS;
    uint4 boneIndices : BONEINDICES;
};

struct VS_OUTPUT {
    float4 position : SV_POSITION;
    float3 positionW : POSITION;
    float3 normalW : NORMAL;
    float2 texcoord0 : TEXCOORD0;
};
```

```
cbuffer cbSkinnedInfo
{
    float4x4 gmtxBoneTransforms[128];
};
```

애니메이션(Animation)

- 외부 라이브러리의 사용

- Open Asset Import Library

http://assimp.sourceforge.net/main_downloads.html

Collada (.dae)
Blender 3D (.blend)
3ds Max 3DS (.3ds)
3ds Max ASE (.ase)
Wavefront Object (.obj)
XGL (.xgl, .zgl)
Stanford Polygon Library (.ply)
AutoCAD DXF (.dxg)
LightWave (.lwo)
LightWave Scene (.lws)
modo (.lwo)
Stereolithography (.stl)
DirectX X (.x)
AC3D (.ac)
Milkshape 3D (.ms3d)
TrueSpace

Quake I (.mdl)
Quake II (.md2)
Quake III Mesh (.md3)
Quake III Map/BSP (.pk3)
Doom 3 (.md5*)
Valve Model (.smd, .vta)
Starcraft II M3 (.m3)
Unreal (.3d)

Ogre XML (.xml)
Irrlicht Mesh (.irrmesh)
Irrlicht Scene (.irr)

BlitzBasic 3D (.b3d)
Quick3D (.q3d, .q3s)
Neutral File Format (.nff)
Sense8 WorldToolKit (.nff)
Object File Format (.off)
PovRAY Raw (.raw)
Terragen Terrain (.ter)
3D GameStudio (3DGS) (.mdl)
Izware Nendo (.ndo)

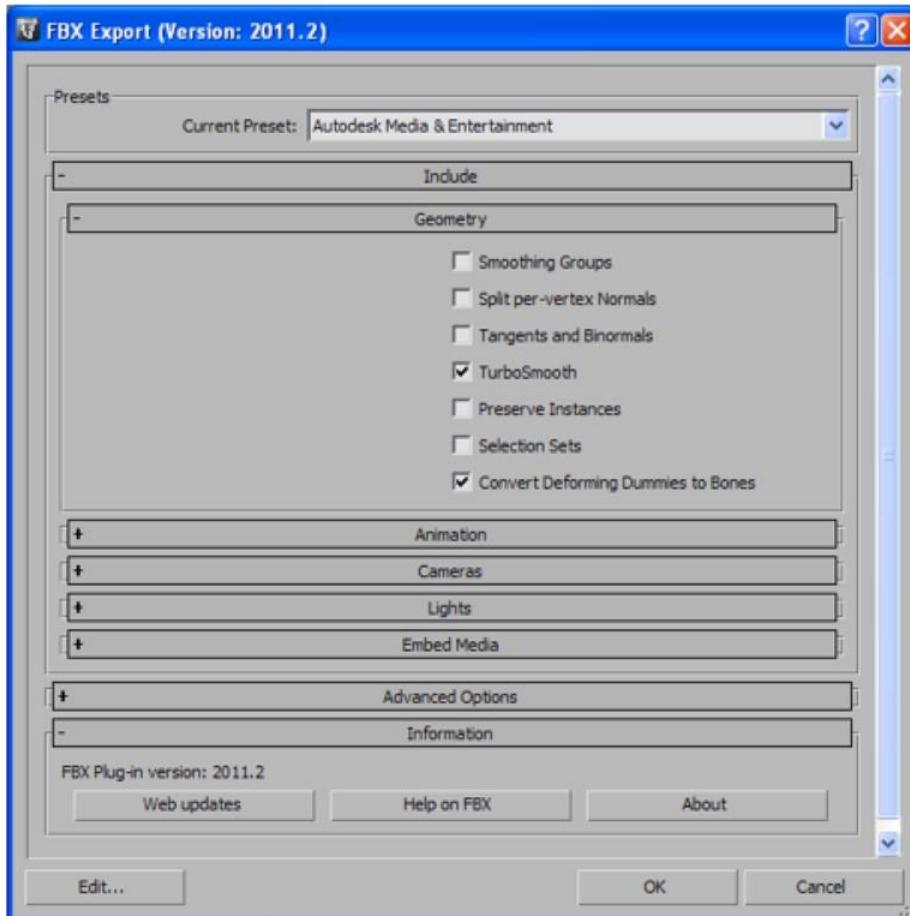


애니메이션(Animation)

- 외부 라이브러리의 사용

- FBX SDK(Software Development Kit)

- FBX 파일 형식에 대한 문서를 제공하지 않음
- FBX 응용 프로그램은 SDK를 사용해야 함
- <http://www.autodesk.com/fbx>



Autodesk®

Autodesk FBX

Autodesk® FBX® asset exchange technology facilitates higher-fidelity data exchange between several Autodesk content creation packages—Autodesk® 3ds Max®, Autodesk® Maya®, Autodesk® MotionBuilder®, Autodesk® Mudbox®, and Autodesk® Softimage® software—and provides support for certain third-party and proprietary applications. Whether you are using FBX within an entertainment pipeline or as part of design production, files are more seamlessly transferred, more data is retained, and workflows are more efficient.

- Single-step interoperability—Maximize creative potential and optimize productivity with single-step interoperability workflows between products in the Autodesk® Entertainment Creation Suites facilitated through Autodesk FBX.
- 2D and 3D tools and support—Efficiently use multiple 3D tools and enjoy support for a wide range of 3D and 2D data.
- Easier data exchange—Exchange digital assets with other studios whose pipelines are standardized on tools different from yours.
- More efficient pipelines—Integrate off-the-shelf and proprietary toolsets and transfer custom data between applications.

[Download Free* FBX Plug-ins, FBX Converter, and FBX QuickTime Viewer](#)

FBX plug-ins for 3ds Max and Maya, FBX importer/exporter to other applications, and the FBX plug-in for Apple® QuickTime® software.

[Download Free* FBX Software Development Kit \(SDK\)](#)

Easy-to-use, C++ development platform and API toolkit for manipulating information stored within FBX.

[Download Free* FBX Extensions SDK](#)

Source code, instructions, and examples for customizing FBX.

[System Requirements \(pdf - 61Kb\)](#)

*Free products are subject to the terms and conditions of the end-user license and services agreement that accompanies the software.