DirectX 기반의 KSL 실행 플랫폼의 개발과 구현

구자효*, 류윤규**

A Study on The Korean Sign Language platform base on DirectX

Ja-Hyo Ku*, Yun Kyoo Ryoo**

요 약

오늘날 디지털 기술과 멀티미디어 영상기법이 발전함에 따라 양질의 영상정보를 획득하기 쉽고 보다 사실적이고 직관적인 정보표현이 가능하여 시각적 욕구를 충족시켜왔다. 대중매체에서 애니메이션 캐릭터를 사용한 영상매체 활용이 지속적으로 늘어나고 있다. 이러한 애니메이션 캐릭터의 표현은 그래픽 기술의 발전으로 입체적이며, 사실적이고 부드러운 연출이 가능해졌다. 일반적으로 다양한 데이터 입력 장치를 이용하여 캐릭터의 섬세한 머리카락의 움직임까지도 표현할 수 있지만, 장애인들과 관련된 멀티미디어의 기술에 대한 연구는 매우 미흡하다. 본 논문에서는 MFC를 이용하여 DirectX 기반의 Korean Sign Language(KSL) 실행 플랫을 연구하였다.

Abstract

The developments of digital and multimedia have been increasing the demand of humans desiring the acquisition of real and intuitive information and diversified expressions and the use of animation characters are continually increasing in mass media. With the development of graphic techniques, these expressions of animation characters have become enabled of real and smooth representations. Although, in general, even fine movements of the hair of characters can be expressed using diverse data input devices, the studies on the multimedia technologies for disabled persons are quite insufficient.

In this paper, Directness it extracts the data which it move sign language and It propose the method which creates a Korea sign language platform.

*Keywords : 한국 수화(Korean Sign Language), DirectX, MFC Platform.

※ 접수일 : 2008.10.20 , 심사완료일 : 2008.12.05

* (주)아이디팩트글로벌 기업부설연구소

** 대구보건대학 행정전산과

※ 본 논문은 중소기업청 기업부설연구소 설치 지원사업에 의한 것임.

Ⅰ. 서론

최근 장애인을 위한 복지 차원의 사회적 관심이 높아지고 있으며 특히 농아 복지에 대한 이해와 의사 소통의 문제가 현실적으로 대두되고 있다. 그 동안 사회의 의사 소통 문제로 한 귀퉁이에서 있던 농아인들을 현실 세계로 이끌어 내고, 사회 일반인으로서의 적응 능력을 키우는 방법들이 많이 연구되고 있다. 최근의 이슈 중 하나는 이러한 농아인들을 위한 수화에 대한 비중이 점점 커지고 있으며 이를 위해 수화 통역사 시험도 시행되고 있다. 가장 가까운 예로 현재 대학의 교양 강좌가 개설되어 학생들이 수강하는 과목 중 하나가 수화이다.

현재 수화 교육을 위해 기초 수화에서 고급 수화 그리고 국제 수화에 이르기까지 다양하고 독특한 교육 프로그램들이 개발되고 있다. 그 동안 일부 사회 복지 기관이나, 특수 직업 종사자, 동아리, 종교 단체 등에서만 제한적으로 사용되던 수화가 일반 사람들이 접하고 배울 수 있다는 것은 농아인들을 위한 사회적 배려와 관심이 성숙한 징조라 보여진다. 그러면 우리 사회의 한 지류인 농아인들이 일반인들과의 문화, 정보등 기타매체에 접근하기 위한 배려는 어느 정도인지 고려해 볼 필요가 있다. 현재까지 농아인들이 일반인들과 접근하기 위한 방법들은 특수전화기, 양방향 문자통신, 자막 수신기등 텍스트 위주의 지식이 대부분이다.

즉, 농아인들을 위해 일반인들의 접근 방법에 대한 전문적인 지식이나 연구를 할 수 있는 실 교육은 찾아보기가 쉽지 않다. 그 이유 중 하나는 농아인들의 언어인 수화에 대하여 일반인들의 이해심 부족과 접근성의 한계를 들 수 있다. 수화를 쉽고 재미있게 접할 수 있는 방법과 이를 활용하여 스스로 수화동작을 검색, 실행하여 학습할 수 있다면 아주 효과적 일 것이다.

수화는 손으로 표현할 수 있는 대부분의 자세를 포함하는 가장 체계적인 언어로서 농아인들에게 가장 비중있는 의사소통의 수단이다. 요즈음 수화에 대한 일반인들의 관심이 높아지고는 있지만, 수화를 습득하기 위해서는 일정한 교육을 받아야 하고 숙달을 위해 많은 시간과 노력이 필요하기 때문에 여전히 불편하다.

수화를 통한 의사소통 과정에서는 손의 움직임뿐만 아니라 얼룩 표정이나 안색 또한 매우 중요한 요소가 된다. 예를 들어, "웃다"라는 수화단어는 손동작이 미소 짓는 얼굴과 함께 표현될 때 "빙그레 웃다"라는 의미가 될 수 있고, 무표정한 얼굴과 함께 행해졌을 때 "비웃다"라는 의미로 받아들여 질 수 있다. 또한, 크게 웃는 얼굴 표정과 함께 "하하 웃다"라는 의미가 될 수도 있다[1]. 따라서 실제 인간이 하는 수화와 최대한 비슷한 효과를 얻기 위해서는 손의 움직임 뿐 아니라 얼굴표정과 안색도 표현 할 수 있도록 구성하여야 한다. 수화동작의 사실적인 표현을 위해 얼굴 요소 및 몸의 움직임을 향상시킬 필요가 있는 것이다.

농아인을 위한 정보통신기술은 특수 전화, 문자통신, 수화통역, 화상통신, 자막방송 등의 영역에서 발전하고 있다. 국내에서도 최근에 다양한 종류의 특수 전화기(골도 전화기, 필담전화기, 음량조절 전화기, 보청기 호환용 전화기 등), 휴대용 문자통신 단말기, 폐쇄자막(Closed Caption) 방송용 TV 등이 개발되었다. 이러한 정보통신 기기들은 관련 서비스가 제공되어야 이용할 수 있는데, 국내에서는 통신 중계서비스, 자막방송 등 농아인의 전화나 TV 이용시 필수적인 서비스가 현재 시범적으로 제공되고 있다. 1997년 9월 무선데이터통신(휴대용 문자통신) 서비스를 개시하였으나 아직 전국 서비스나 전화와의 연계서비스는 제공되지 않고 있다. 미국을 중심으로 한 선진국에서는 과거부터 장애인 복지문제가 매우 큰 이슈로 대두되어 왔기 때문에 장애인 교육과 훈련을 위한 기술개발 및 제품개발에 많은 투자가 이루어지고 있는 상황이다. 장애인용 IT 서비스 개발과 관련된 기술은 4가지 분야로 크게 나눈다.

첫째, 감각기능을 보조하거나 변환하는 기술과, 둘째, 현실감 있는 디지털 정보 재현 기술, 셋째, 인지과학 및 교육심리학의 원리를 적용하는 기술, 마지막으로 통합정보기술이다.

아직까지는 이러한 연구들이 괄목할만한 성과를 거두지 못하고 있으며 무엇보다도 단순 기술개발에만 집중하고 있기 때문에 실제 사용자들이 장애인들에게 큰 관심을 얻지 못하고 있는 실정이다. 특히, 완성도 있고 친밀감 있는 콘텐츠 개발과 응용프로그램 개발이 무엇보다 중요하나 실제 사용 주체가 되어야 하는 장애인들만 사용하는 기술개발로 전락하고 있어 실생활에서의 활용 가능성은 줄어들고 있다.

본 논문에서는 3D 애니메이션으로 구성된 수화동작을 검색하여 실행 할 수 있는 DirectX 기반의 플랫폼을 구현한다.

II. 관련 연구

2.1 KSL(Korean Sign Language)

수화(Sign Language)는 농아인이 사회생활을 하기 위해 사용하는 가장 비중 있는 의사소통 수단으로써, 농아인에 의해 만들어졌고 지속적으로 반복 생성되는 일종의 언어표현의 수단이다. 따라서 각 나라마다 수화의 표현방식이 다르며 한국에서 농아인들이 사용하는 수화를 KSL(Korean Sign Language)라 한다. 이는 일반인들이 발성을 통해서 표현하는 음성체계가 아니라 손을 통해서 표현하는 시각 운동체계이며 단순하게 음성언어의 정확한 손짓을 재현한 것이 아니라, 독특한 표현양식 및 영어의 관용어구와 같은 관용적 수화표현과 문법적 구조를 지닌 하나의 완전한 언어이다.[1]

수화의 기호 구성 방법은 연구자에 따라 다를 수 있지만 본 논문에서는 김승국(1983)의 수화의 기호 구성 방법을 도입하였다.[2]

(1) 지사: 명사 인칭대명사 지시대명사 시간을 가리키는 명사를 지시한다.
예) 아래, 주먹, 코, 머리, 빨강, 과거
(2) 모방: 낱말이 의미하는 동작을 흉내 낸 것이다
예) 보다, 걷어차다, 말하다, 도장, 날개, 눈물
(3) 상형: 낱말이 의미하는 대상의 형체나 그 형체의 변별자질을 손이나 손가락으로 나타내는 것이다.
예) 산, 얼굴, 삼각형, 십자가, 곰, 대궐
(4) 형지: 대상의 형체를 만들어 보이고 그것을 가리키는 동작을 해 보이는 것이다.
예) 비탈, 모퉁이
(5) 형동: 대상의 형체나 그 일부를 만들어 그 대상이 하는 동작을 해 보이는 것이다.
예) 잠자리, 나비, 코끼리
(6) 회의: 둘 이상의 형태소를 합쳐 다른 의미를 가지는 낱말의 기호를 만드는 것이다.
예) 일요일, 소중하다, 장미
(7) 전주: 기존의 기호가 그것을 의미하고 낱말의 동의어와 유사어의 뜻을 나타내는데 사용되는 것이다.
예) 교사, 교원 ,교육자, 사범, 선생, 스승,

2.2 수화학습방법

언어의 습득과 발달은 생득적 기제에 의해서 이루어질 수도 있지만, 사람과 사람간의 사회적 상호작용에 의해서도 이루어질 수 있는데, 일반인과 비교하여 청각장애인의 언어 습득 및 발달에는 사회적 상호작용을 통한 적절하고 풍부한 언어 경험이 무엇보다 중요하다.

이러한 경험은 대화를 기반으로 하는 언어적 경험과 얼굴표정, 손짓, 몸짓을 기반으로 하는 비언어적 경험으로 나뉘는데, 학습자의 전체적인 언어 발달을 위해서는 교육학, 심리학, 의학 등 다양한 측면에서의 분석과 연령, 행동발달 성향을 고려한 복합적인 학습방법으로 진행되어야 할 것이다.

2.3 DirectX

DirectX는 1995년 윈도우 환경에서 멀티미디어와 오락기능을 강화시킨 응용 프로그램 인터페이스로 마이크로소프트사에서 개발하였다. 컴퓨터 게임에 적합한 기능을 고루 갖추고 있어 게임 및 애니매이션의 3D분야에서 다양하게 사용되고 있다. MFC (Microsoft Foundation Class)는 windows기반의 응용 프로그램 개발에 많이 사용되고 있으며 편리한 인터페이스와 다양한 프레임 워크를 제공하여 빠르고 손쉬운 프로그래밍 환경을 제공하는것이 특징이다.

이러한 MFC를 기반으로한 DirectX 프로그래밍은 DirectX 구동시 필요한 프레임워크 개발시간을 단축시키고 편리한 인터페이스 제공 및 기존 응용 프로그램과의 호환성이 뛰어나다.

III. KSL 실행 플랫폼 설계

3.1 DirectX 기반의 KSL 실행 플랫폼 구성

KSL 플랫폼은 3D MAX로 구성된 수화 데이터파일을 X파일로 변환하여 사용자가 단어를 입력 시 해당 하는 내용의 X파일을 불러와 화면상에 렌더링 하여 보여주게 되며 이와 같은 상황이 반복되어 진행 되게 된다. KSL 플랫폼을 구성하기에 앞서 수화 데이터를 X파일로 변환하는 과정이 필요하다. 수화데이터의 기본적인 움직

임의 구성은 골격(Bone) 형식을 기본 구조로 하고 있다. 각 Bone는 계층구조로 이루어져 있으며 부모의 노드가 변할 때 자식의 노드도 함께 따라 변화한다. 이러한 구조의 행렬들이 X파일에 저장 되어 있어야 하며 정점을 변화시키면서 움직임을 구현하게 되어있다. 정점은 Bone에 영향을 받고, Bone이 회전하면 정점도 회전하게 된다. Bone과 Bone 사이의 관절이 하나로 구성되어 있으면 움직임이 발생할시 관절이 깨져보이게 되므로 자연스러운 동작의 구현이 힘들게 된다. 따라서 관절을 여러 정점들로 구성을 하여 움직임이 자연스럽게 동작할 수 있게 만드는 것이 수화 동작을 구현하는 가장 기본적인 원리이다. 이것은 가중치라는 수치를 적용하여 구현한 것으로 예를 들어 팔과 팔뚝이나 팔뚝과 손목, 다리와 발목 등등 관절을 이어주는 부분에 6:4나 7:3의 비율로 가중치를 주어 두 개 이상의 Bone 구조에 영향을 주어 좀 더 부드러운 움직임을 보여 줄 수 있는 것이다. 이러한 가중치는 합이 1이 되어야 하며 Bone의 개수는 각자 정하기 나름이다. 그림 1은 KSL실행 플랫폼의 전체 구성도이다.

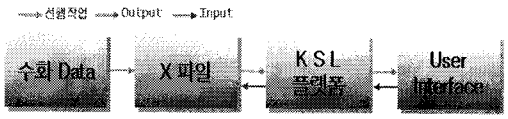

그림 1. KSL 플랫폼 전체 구성도

3.2 KSL 실행 플랫폼의 구성요소

KSL 실행 플랫폼은 그림 2와 같은 구성요소를 가진다. X파일로 변환된 수화데이터 저장을 위한 DB구성 부분과 사용자의 입력에 따른 정보를 디스플레이 하기 위한 Direct 구현부분, 사용자의 입력한 정보를 가지고 하나의 수화 문장을 구성하기 위한 문장구현 부분으로 구성된다.

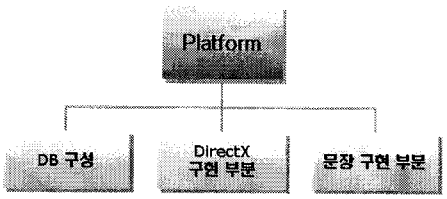

그림 2. KSL 실행 플랫폼 상세도

3.3 DirectX 구현 부분

각 장별로 수화동작을 정리 한 후 본격적인 DirectX 구현 부분이 적용된다. DirectX 구현부분을 살펴보면 Create Mesh Container, Generate Mesh Container, Render 부분으로 나눌 수 있다.

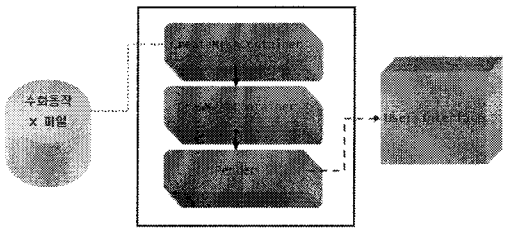

그림 3. DirectX 구현 상세도

Ⅳ. 구현

4.1 데이터베이스 구현

원활한 수화동작의 디스플레이와 실행을 위해서는 최적화된 DB구성이 요구된다. DB는 이식성과 경량화를 고려하여 MS Access기반에서 ODBC를 구성하였다.

일반적으로 네트워크 기반이 아닌 로컬 모드로 구성된 프로그램에서 MS Access는 어떠한 환경에서도 쉽게 구성할 수 있을 뿐 아니라 다른 환경으로의 이식성이 높은 장점이 있다. KSL을 구성하는 요소 중 가장 중요하다고 할 수 있는 수화동작 파일은 위에서 설명한 것과 같이 X파일이라는 포맷을 이용하였으며, 이 X파일은 DirectX를 구현하기 위한 가장 기본적인 파일이라고 할 수 있다. 따라서 DirectX SDK만 있으면 3D애니메이션을 구현 할 수가 있다.

DirectX 프로그래밍에 근본적인 파일포맷이라 여기서 파생된 파일들이 각자의 환경과 요구사항에 맞게 변경되고 재구성 되어 각자의 포맷으로 바뀌어 사용하게 된다. 하지만 현재의 KSL 플랫폼 설계에서는 그렇게 많은 요구사항과 환경적 요건이 필요치 않기 때문에 가장 기본적인 X파일을 사용한다.

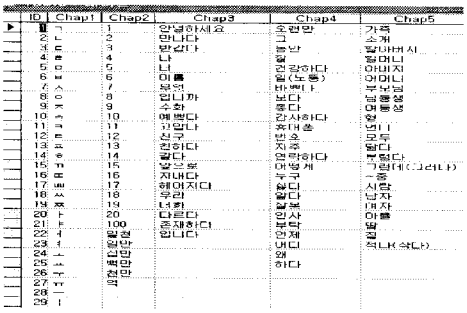

그림 4. 수화 데이터베이스

4.2 X 파일 생성 및 변환

수화 데이터의 가장 기본적인 요소인 Bone 구조를 구현하기 위해 먼저 Bone의 정보를 파악하고 기본적인 Mesh 데이터를 생성하여 최종적으로 X파일로 구성한다. 여기서 Bone 정보만을 가지고는 X파일을 구성하고 렌더링 할 수 없기에 Mesh를 생성한다.

수화데이터는 각각 약 2만여 개의 Mesh로 구성되어 있는데, 이 같은 Mesh 데이터뿐 만 아니라 X파일을 구성하는 요소인 텍스처의 경로, 캐릭터 애니메이션에 필요한 좌표 행렬 및 얼굴표정 등이 포함 되면 기본적인 X파일을 구성할 수 있다. 이렇게 X파일을 Text 파일로 구성하게 되면 각 수화단어의 애니메이션 용량은 약 3MB로 파일 렌더링을 위해 해당 파일을 메모리에 적재하는데, 문장 구현 시에는 단어의 개수만큼이나 메모리를 사용하기 때문에 전체적인 시스템 성능이 저하되는 문제점이 있다.

```
SBoneInfo *FindBone(INode *pBoneNode)
{
    SBoneInfo *pbi = NULL;
    DWORD iBone;

    for (iBone = 0; iBone < m_cbiBones; iBone++)
    {
        if (pBoneNode == m_rgbiBones[iBone].m_pBoneNode)
        {
            pbi = &m_rgbiBones[iBone];
            break;
        }
    }

    return pbi;
}
```

그림 5. Bone정보 검색 알고리즘

이러한 문제를 해결하기 위해 Text파일로 구성된 X파일을 Binary코드를 사용하여 X파일의 용량을 700~800KB로 압축시켜 메모리 적재 시 과부하 문제 해결 및 시스템 성능을 향상시켰다.

그림 5는 Bone정보를 검색하기 위한 알고리즘이고 그림 6은 Mesh정보를 변환하기 위한 알고리즘이다. 그림 7은 X파일 변환 알고리즘을 각각 정의한 함수를 나타낸다..

```
BOOL IsExportableMesh(INode* pNode, Object** pObj)
{
    ULONG superClassID;
        Object *pObjBase;

    assert(pNode);

        pObjBase = pNode->GetObjectRef();
    if (pObjBase == NULL)
        return FALSE;

        if(pObj)*pObj= pObjBase;

    superClassID = pObjBase->SuperClassID();


    if( !pNode->Renderable() || pNode->IsNodeHidden())
    {
        return FALSE;
    }
```

그림 6. Mesh정보 변환 알고리즘

```
HRESULT ExportXFile
    (
    const TCHAR *szFilename,
    ExpInterface *pExportInterface,
    Interface *pInterface,
    BOOL bSuppressPrompts,
    BOOL bSaveSelection,
    HWND hwndParent
    )
{
    LPDIRECTXFILE pxofapi = NULL;
    LPDIRECTXFILEDATA pRootData = NULL;
    LPDIRECTXFILESAVEOBJECT pxofsave = NULL;
    HRESULT hr = S_OK;
    INode *pRootNode;
    SSaveContext sc;
    SPreprocessContext pc;
    SDialogOptions DlgOptions;

    assert(szFilename && pExportInterface && pInterface);
```

그림 7. X파일 변환 알고리즘

4.3 DirectX 함수를 이용한 구현

4.3.1 Create Mesh Container

Mesh 컨테이너 안에서 메테리얼의 개수와 텍스처, 기타 Mesh에 필요한 정보들을 메모리에 확보 해놓는다. 메테리얼이 존재한다면 인자로 넘어온 메테리얼 구조체를 그대로 복사한다. 다음으로 텍스처 이름을 검사하여 텍스처를 생성하고 이름이 있던 자리에는 NULL값을

넣는다. 만약 메테리얼이 없다면 메테리얼이 가지고 있는 기본값을 넣어준다.

그림 8은 Create Mesh Container를 정의한 함수이다.

```
HRESULT CXViewer::CAllocateHierarchy::CreateMeshContainer(
    LPCSTR Name,
    CONST D3DXMESHDATA *pMeshData,
    CONST D3DXMATERIAL *pMaterials,
    CONST D3DXEFFECTINSTANCE *pEffectInstances,
    DWORD NumMaterials,
    CONST DWORD *pAdjacency,
    LPD3DXSKININFO pSkinInfo,
    LPD3DXMESHCONTAINER *ppNewMeshContainer)
{
    HRESULT hr;
    D3DXMESHCONTAINER_DERIVED *pMeshContainer = NULL; // 임시로 사용할 메시 컨테이너의 포인터
    UINT NumFaces; // 삼각형 면의 개수
    UINT iMaterial; // 메테리얼 개수
    UINT iBone, cBones; // 본의 개수, 본 카운트 임시 변수
    LPDIRECT3DDEVICE9 p_device = NULL; // 임시로 사용할 메시

    LPD3DXMESH pMesh = NULL;

    *ppNewMeshContainer = NULL;
```

그림 8. Create Mesh Container 함수

4.3.2 Draw Mesh Skinned Mesh

스킨정보의 포인터를 복사해 주면서 AddRef()로 레퍼런스 카운트를 늘려준다. 메시 컨테이너의 오리지널 Mesh에 pMesh를 복사해 주면서 레퍼런스 카운트를 늘려준다. 그 후 Mesh 컨테이너 안에 본 오프셋 매트릭스를 세팅해주기 위해 스킨정보 인터페이스 안에 저장되어있는 Bone의 개수를 얻어온다. 그림 9와 10은 Draw Mesh Container와 Draw Frame을 정의한 함수이다.

```
void CXViewer::DrawMeshContainer(XFRAME* xFrame, LPD3DXMESHCONTAINER pMeshContainerBase, LPD3DXFRAME pFrameBase )
{
    HRESULT hr;
    D3DXMESHCONTAINER_DERIVED *pMeshContainer = (D3DXMESHCONTAINER_DERIVED*)pMeshContainerBase;
    D3DXFRAME_DERIVED *pFrame = (D3DXFRAME_DERIVED*)pFrameBase;
    UINT iMaterial;
    UINT NumBlend;
    UINT iAttrib;
    DWORD AttribIdPrev;
    LPD3DXBONECOMBINATION pBoneComb;

    UINT iMatrixIndex;
    UINT iPaletteEntry;
    D3DXMATRIXA16 matTemp;
    D3DCAPS9 d3dCaps;
    mp_device->GetDeviceCaps( &d3dCaps );
```

그림 9. Draw Mesh Container 함수

```
void CXViewer::DrawFrame(XFRAME* xFrame, LPD3DXFRAME pFrame)
{
    LPD3DXMESHCONTAINER pMeshContainer;

    pMeshContainer = pFrame->pMeshContainer;
    while (pMeshContainer != NULL)
    {
        DrawMeshContainer(xFrame, pMeshContainer, pFrame );

        pMeshContainer = pMeshContainer->pNextMeshContainer;
    }

    if (pFrame->pFrameSibling != NULL)
    {
        DrawFrame(xFrame, pFrame->pFrameSibling);
    }

    if (pFrame->pFrameFirstChild != NULL)
    {
        DrawFrame(xFrame, pFrame->pFrameFirstChild );
    }
}
```

그림 10. Draw Frame 함수

AttribIdPrev변수는 이전 루프의 메시의 Attribute 번호를 저장하는 변수이다. 메테리얼이 여러 개일 때 본 콤비네이션 안에 있는 이 정보로 드로우를 하다가 메테리얼을 교체 해줘야하는 시기를 판단하기 위해 사용된다. 애니메이션의 Bone는 트리구조이기 때문에 현재 있는 Mesh 컨테이너를 모두 그려준 후 다른 형제 프레임이 있는지 검사해서 자신과 같은 깊이의 Mesh 컨테이너가 그려지게 된다. 이후 자기보다 한 단계 깊은 Mesh 컨테이너를 그려주게 되면 결국 애니메이션 Mesh 전체가 그려지게 되는 것이다.

4.3.3 Render

Mesh 컨테이너들을 만들고 Drawing 함수 부분이 모두 완성 되었다면 사용자의 요청에 대한 Render결과를 디스플레이 한다. 그림 11은 Rendering을 정의한 함수이다.

```
// Begin the scene
if( SUCCEEDED( mp_device->BeginScene() ) )
{
        mp_device->Clear(0, NULL, D3DCLEAR_ZBUFFER, D3DCOLOR_XRGB(216, 237, 116), 1.0f, 0);
        XFRAMES::iterator p = m_xframes.begin();
        for(; p != m_xframes.end(); p++)
        {
            if(p->second.b_play)
            {
                DrawFrame(&p->second, p->second.p_frame_root);

            }
        }
    // End the scene.
    mp_device->EndScene();

}
```

그림 11. Render 함수

먼저 ZBuffer의 내용을 초기화하고 애니메이션의 각 프레임 마다 Draw Frame 함수를 호출하여 화면에 디스플레이 한다. 추가적으로 디스플레이 전에 카메라의

보는 시점, 위치, 업벡터 등을 환경설정을 하여 사용자가 원하는 환경에서 수화 동작을 볼 수 있다.

4.4 문장 구현

문장 구현에서 가장 중요한 것은 단어와 단어 사이를 이어주는 모션 블랜딩 이라고 할 수 있다.

각 정점들의 좌표를 계산하여 기준이 되는 정점으로 좌표를 변환 시켜 각 동작들 간의 이어짐이 한결 부드럽게 이어지게 만든다. 각 동작들은 각자의 서로 다른 정점 좌표 값을 가지고 있어서 그 사이를 이어주는 캐릭터 애니메이션 보간 방법이 적용되어야 한다. 그러한 보간 방법을 적용하기 전에 최소한의 간격을 만들어주기 위하여 정점들의 좌표를 변환시켜 주는 것이다.

만약 이러한 방법을 취하지 않고 캐릭터 애니메이션 보간 방법을 적용한다면 그 간격이 너무 멀어져 보간 방법을 적용한다 하더라도 캐릭터가 슬로우 모션을 하는듯한 동작을 하기 때문에 정점 좌표를 변환시켜주는 것이 필요하다. 그림 12는 모션 블랜딩을 정의한 함수이다.

```
xframe.notify = m_frame_end_noFification;
xframe.id = notify_id;
xframe.main_wnd = MH_wnd;

XFRAMES:: Pairib r = m_xframes.insert(XFRAMES::value_type(file, xframe));
r.first->second.animation_callback.set_callback_data(&(r.first->second));
D3DXFrameCalculateBoundingSphere( r.first->second.p_frame_root, &r.first->second.bs_center, &r.first->second.bs_r);

float xl, yl, zl, xh, yh, zh;
XFRAMES::iterator p = m_xframes.begin();
float ra = p->second.bs_r;
xl = p->second.bs_center.x - ra;
yl = p->second.bs_center.y - ra;
zl = p->second.bs_center.z - ra;
xh = p->second.bs_center.x + ra;
yh = p->second.bs_center.y + ra;
zh = p->second.bs_center.z + ra;

p++;

D3DXVECTOR3 c;
for(; p != m_xframes.end(); p++)
{
        c = p->second.bs_center;
        ra = p->second.bs_r;

        if(xl > c.x - ra){xl = c.x - ra;}
        if(yl > c.y - ra){yl = c.x - ra;}
        if(zl > c.z - ra){zl = c.x - ra;}
        if(xh < c.x + ra){xh = c.x + ra;}
        if(yh < c.y + ra){yh = c.x + ra;}
        if(zh < c.z + ra){zh = c.x + ra;}
}
m_global_center.x = xl + (xh - xl)/2.0f;
m_global_center.y = yl + (yh - yl)/2.0f;
m_global_center.z = zl + (zh - zl)/2.0f;

D3DXVECTOR3 ru;
ru.x = xh - xl;
ru.y = yh - yl;
ru.z = zh - zl;
ru /= 2.0f;
```

그림 12. 문장구현을 위한 모션 블랜딩

4.5 사용자 인터페이스

3D애니메이션의 실행을 위한 사용자 인터페이스는 그림 13과 같이 구성 하였다. 3D애니메이션 동작화면을 볼 수 있도록 왼쪽 상단에 구성 하였으며, 각 장 별로 구성된 단어와 전체 단어 그리고 검색된 단어를 찾을 수 있는 단어 목록 리스트를 오른쪽에 구성하였다.

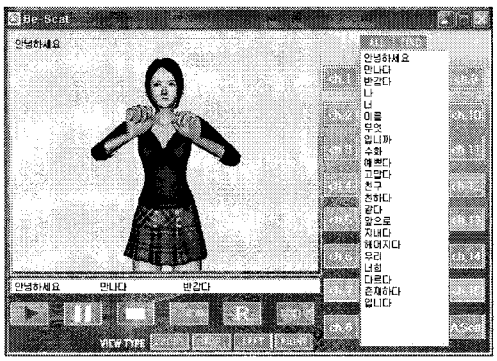

그림 13. 사용자 인터페이스

동작 화면 아래에는 현재 등록된 단어를 보여주는 단어 등록 창이 있으며 오른쪽의 단어 리스트를 더블클릭하거나 단어 등록 창으로 Drag & Drop을 시켜 단어를 추가 할 수 있다. 더블클릭이나 Drag & Drop은 사용자가 쉽게 단어를 추가 할 수 있도록 기존의 마우스에 익숙한 사용자들이 헤매지 않고 등록할 수 있도록 하였다.

단어 등록 창 아래에는 프로그램의 기능 버튼을 추가하였다. 실행, 일시정지, 정지, 단어 반복 설정, 초기화, 종료 등 여러 기능을 넣었다. 하단에는 동작화면을 여러 각도에서 볼 수 있도록 정면, 윗면, 우측면, 좌측면 카메라 설정 버튼으로 구성 되어 있어 동작을 상세하게 볼 수 있도록 하였다.

V. 결론

수화의 학습 기법과 비교하여 일반인과 영유아기 및 장애인의 수화학습에 높은 효율성을 제공하고 장애인의 사회 적응력을 더욱 높이기 위해서 사용자 인터페이스 기능이 향상된 DirectX 기반의 KSL실행 플랫폼을 구현하였다. 이는 일반인들이 수화를 보다 쉽게 접근하여 사용할 수 있도록 3D애니메이션 수화 DB를 설계 하고 구현하였다. 3D 수화 데이터는 모션캡쳐를 통해 획득한 데이터를 3D MAX로 매핑하여 약 500개를 단어별로 생성

하였다. 단어의 유사어, 동음이의어는 배제하였다. 그러나 한글은 다양한 표현과 많은 용어로 구성되어 있기 때문에 한글로 구성되는 모든 단어를 구성하려면 실제로 다양한 경우의 수가 발생한다. 이러한 경우의 수는 실행을 위한 플랫폼의 구성과 3D애니메이션 수화데이터의 파일 사이즈와 많은 관계를 가진다. 현재 구성된 KSL 실행 플랫폼은 3D애니메이션 수화동작을 효과적으로 실행하기 위하여 고안되었다. 향후 수화동작의 보다 빠른 실행을 위해 반복된 테스트를 통해 안정되고 다양한 조건의 실행환경이 기반 되어야 한다.

또한, 향후 유비쿼터스 환경에 적합한 수화 번역 서비스 및 3D애니메이션을 이용한 수화사전을 구현하여 언어적 의미를 확장한다면 일반인들에게도 쉽게 응용할 수 있을 것으로 기대한다.

참 고 문 헌

[1] 김승국, "한국 수화의 심리언학적연구", 성균관대학교 박사학위논문, 1983.

[2] 손천식, "한국수화의 관용표현", 한국수화연구회 제1회 공개 연구발표회-수화농문화를 생각하는 세미나, VOL.1, pp.21-41, 2000.

[3] 김경진, "청각장애 학생의 언어지도와 수화교육의 탐색", 국립특수교육원 교육연구사, 특수교육 교육과정 연구 제2집, Vol.2, pp.247-264, 2001.

[4] Ja-Hyo Ku. Nam-Chul Jo, Senong-Bok Yang " A Study on the method of creates a realty 3D sign language gesture", *ICIC2008*, vol. 1, pp. 185-188, June 2008.

[5] 이 찬수, 김 종성, 박 규태, 장 원, 변 증남, "지문자를 포함한연속된 한글 수화의 실시간 인식 시스템 구현", 전자 공학지 제35권 C편 제 6호, pp. 78-79, 1998.

[6] 최 창석 외 수화 동작의 인식 및 동영상, " 생성의 연구 동향",전자 공학 회지, pp. 671-681, vol. 23, no. 6, 1996.

[7] 김 종성, 이 찬수, 장 원, 변 증남, "한글 수화용 동적 손 제스처의 실시간 인식 시스템의 구현에 관한 연구", 전자공학회지제 34권 C편 제 2호, pp. 61-70, 1997.

[[8] K. H. Quek, "Toward a Vision-Based Hand GestureInterface", *in Proc. of the VRST94 Conf.*, pp. 17-31, 1994.