모바일 볼륨 가시화 시스템 개발
(Development of Mobile Volume Visualization System)

박 상 훈 † 김 원 태 †† 임 인 성 †††
(Sanghun Park) (Wontae Kim) (Insung Ihm)

요 약 모델링·시뮬레이션 그리고 센서 장비 기술의 지속적인 발전으로 최근에 매우 높은 해상도를 갖는 방대한 크기의 볼륨 데이타들이 일반화되고 있다. 과학적 가시화 분야에서, 이러한 데이타를 고성능 병렬 컴퓨터를 사용하여 효과적으로 가시화하기 위한 다양한 대화식 실시간 기법들이 제안되어 왔다. 본 논문에서는 모바일 클라이언트, 게이트웨이, 병렬 렌더링 서버로 구성되는 모바일 볼륨 가시화 시스템의 개발에 관해 설명한다. 모바일 클라이언트는 병렬 렌더링 서버에게 전달할 렌더링 / 뷰잉 파라미터를 설정할 수 있는 기능뿐만 아니라, 관심이 있는 특정 영역을 점진적으로 높은 해상도의 영상을 이용해 탐색할 수 있는 기능을 제공한다. 게이트웨이는 안정적인 서비스를 위해 모바일 클라이언트와 병렬 렌더링 서버 사이에서 주고받는 요청과 응답을 관리하는 역할을 한다. 병렬 렌더링 서버는 클라이언트로부터 전달받은 렌더링 컨텍스트를 이용하여 정의된 특정 부분 볼륨을 가시화하고, 고해상도의 최종 영상을 클라이언트에게 되돌려 주는 작업을 수행한다. 제안된 시스템은 PDA를 갖고 있는 여러 사용자가 협력작업(CSCW) 모드를 통해 동시에 볼륨 데이타 가운데 공통으로 관심을 갖는 특정 부분, 렌더링 컨텍스트, 그리고 최종 영상을 공유할 수 있도록 설계되었다.

키워드 : 컴퓨터 그래픽스, 과학적 가시화, 모바일 컴퓨팅, 클러스터 컴퓨팅, 컴퓨터지원 협력작업

Abstract Due to the continuing technical progress in the capabilities of modeling, simulation, and sensor devices, huge volume data with very high resolution are common. In scientific visualization, various interactive real-time techniques on high performance parallel computers to effectively render such large scale volume data sets have been proposed. In this paper, we present a mobile volume visualization system that consists of mobile clients, gateways, and parallel rendering servers. The mobile clients allow to explore the regions of interests adaptively in higher resolution level as well as specify rendering / viewing parameters interactively which are sent to parallel rendering server. The gateways play a role in managing requests / responses between mobile clients and parallel rendering servers for stable services. The parallel rendering servers visualize the specified sub-volume with rendering contexts from clients and then transfer the high quality final images back. This proposed system lets multi-users with PDA simultaneously share commonly interesting parts of huge volume, rendering contexts, and final images through CSCW(Computer Supported Cooperative Work) mode.

Key words : Computer Graphics, Scientific Visualization, Mobile Computing, Cluster Computing, CSCW(Computer Supported Cooperative Work)

1. 서 론

고성능 컴퓨터 하드웨어 기술과 수학·계산이론의 발전을 기반으로 다양한 자연 현상과 물리적인 현상들을 매우 세밀하고 정교하게 모델링·시뮬레이션하고, 이를 3차원 컴퓨터 그래픽스 기법들을 이용하여 가시화하는 과학적 가시화(scientific visualization)연구가 활발하게 이뤄지고 있다[1]. 또한, 단층촬영장비, 3차원 스캐너, 전자현미경, 인공위성용 망원렌즈 등과 같은 첨단 센서 장비의 발전은 다양한 종류와 형태를 가진 세밀하고 방대한 실사 데이타의 획득을 용이하게 하였다. 이렇게 높은

· 이 논문은 2004년도 한국학술진흥재단의 지원에 의하여 연구되었으며 (KRF-2004-041-D00615). 연구의 일부는 2단계 BK21 사업의 지원을 받았음
† 종신회원 : 동국대학교 영상대학원 멀티미디어학과 교수
mshpark@dongguk.edu
(Corresponding author임)
†† 정 회 원 : 케이앤아이테크놀로지 기술연구소
matt99@grmanet.sogang.ac.kr
††† 종신회원 : 서강대학교 컴퓨터학과 교수
ihm@sogang.ac.kr
논문접수 : 2006년 2월 2일
심사완료 : 2006년 5월 23일

해상도의 데이타를 이용하는 주된 목적은 물리적인 자연현상의 원인과 결과를 보다 세밀하고 정확하게 분석·이해하기 위해서이며, 방대한 데이타의 가시화와 분석은 여러 과학·공학 분야에서 그 중요성이 계속해서 강조되고 있다. 컴퓨터 하드웨어 기술의 발달로 프로세서의 계산 능력과 메모리 용량이 빠르게 증대되고 있으나, 고해상도의 방대한 볼륨 데이타를 효과적으로 가시화하는 일반적인 기법은 초고속 네트워크로 연결된 고성능 분산·병렬 시스템을 이용하는 것이다[2,3].

한편 최근 무선 통신 기술의 발전으로 휴대폰과 PDA 중심의 모바일 기기 시장이 급격히 확대되고 있다. 특히 시간과 장소에 구애받지 않고 자유롭게 네트워크에 접속할 수 있는 유비쿼터스(ubiquitous) 컴퓨팅 기술의 발전, 차세대 스마트폰의 개발, DMB(Digital Multimedia Broadcasting)의 상용화 등에 힘입어 모바일 산업의 성장은 더욱 가속화되고 있다. 더욱이 차세대 무선 인터넷 서비스인 와이브로(Wibro : Wireless Broadband)의 상용화는 모바일 산업에 대한 전망을 더욱 밝게 하고 있다. 이러한 모바일 환경의 변화와 표준 그래픽스 라이브러리인 OpenGL ES의 보급 확대를 기반으로 휴대용 모바일 기기를 이용한 PC 수준의 고급 3차원 게임이나 그래픽스 응용 프로그램을 구현하기 위한 노력이 활발히 진행되고 있다. 그러나 이러한 모바일 환경을 기반으로 하는 과학적 가시화 시스템의 개발에 관한 연구는 아직까지 발표된 바가 없다. 이미 널리 알려진 바와 같이 방대한 크기의 데이타로부터 3차원 영상을 생성하기 위한 볼륨 렌더링 알고리즘들은 많은 계산비용과 큰 메모리 공간을 필요로 한다. 따라서 현재의 상용 모바일 기기가 갖는 하드웨어적인 한계로 인해, 고성능 워크스테이션이나 PC상에서 구현된 가시화 알고리즘들[4,5]을 그대로 휴대폰이나 PDA에 이식하는 것은 거의 불가능하다. 빠르게 발전하고 있는 모바일 컴퓨팅 관련 기술의 발전 속도를 고려할 때, 수 년 내에 고성능 그래픽스 가속기를 탑재한 모바일 기기를 이용한 실시간 볼륨 렌더링의 가능성을 기대할 수 있겠지만, 현재의 계산 환경에서 가장 합리적인 접근 방법은 클라이언트-서버 시스템을 구축하고 이를 기반으로 볼륨 가시화 서비스를 제공하고 성능을 최적화 하는 것이다.

본 논문에서는 모바일 볼륨 가시화 시스템의 개발에 대해 설명한다. 구현된 시스템은 PDA 기반의 모바일 클라이언트, 게이트웨이, 병렬 렌더링 서버로 구성된다. 모바일 클라이언트는 병렬 렌더링 서버에게 전달할 렌더링 / 뷰잉(viewing) 파라미터를 설정할 수 있는 기능뿐만 아니라, 관심이 있는 특정 영역을 점진적으로 높은 해상도의 영상으로 탐색할 수 있는 기능을 제공한다. 게이트웨이는 안정적인 서비스를 위해 모바일 클라이언트와 병렬 렌더링 서버 사이에서 주고받는 요구와 응답을 관리하는 역할을 한다. 병렬 렌더링 서버는 클라이언트로부터 전달받은 렌더링 컨텍스트(rendering contexts)를 이용하여 정의된 특정 부분 볼륨을 가시화하고, 고해상도의 최종 영상을 클라이언트에게 되돌려 주는 작업을 수행한다. 본 시스템은 자세한 관찰이 필요한 특정 영역을 점진적으로 높은 해상도로 관찰할 수 있는 기능과 협력작업(CSCW: Computer Supported Cooperative Work)을 통해 네트워크상의 여러 사용자들 사이에 정보 공유와 의견 교환이 가능한 환경을 제공할 수 있다는 특징을 갖고 있다.

논문의 구성은 다음과 같다. 2절에서는 본 시스템과 관련된 기존의 연구들을 비교하고 차이점과 특징을 정리한다. 3절에서는 시스템의 각 부분에 대한 구체적인 설계에 대해 자세히 설명하고 CSCW 환경의 구현과 기능을 소개한다. 4절에서는 실험 결과를 통해 시스템의 성능을 분석한다. 마지막으로 4절에서 결론 및 향후 연구에 대해 논의한다.

2. 관련 연구

범용 PC 클라이언트에서 고성능 렌더링 서버에 접속하여 방대한 데이타를 효과적으로 가시화하는 다양한 연구 결과들이 발표되었다. 특히 최근에 데이타가 급격하게 방대해짐에 따라서, 원격지에 저장된 데이타 가운데 특정 부분만을 직접 액세스하여 가시화를 수행하거나 서버에서 렌더링 된 영상을 받아 디스플레이만을 수행하는 클라이언트 소프트웨어들이 개발되고 있다. 미켈란젤로 프로젝트를 통해 생성된 방대한 크기의 다각형 데이타를 효과적으로 저장하고 렌더링 하는 ScanView라는 이름의 클라이언트-서버 렌더링 시스템은 대표적인 예라고 할 수 있다[6]. 이 시스템의 클라이언트는 PC상에서 수행되는 간단한 뷰어 인터페이스와 축소된 다각형 모델로 구성되며, 사용자는 인터페이스를 통해 카메라의 위치와 방향을 바꿔가며 축소모델을 실시간 속도로 렌더링하면서 거친 영상들을 생성할 수 있다. 고화질의 영상을 생성하기를 원하는 경우에는 카메라를 적당한 위치에 배치한 후 서버에 접속하여 고화질의 해당 영상을 받을 수 있도록 구현되어 있다.

클라이언트-서버 기반의 볼륨 렌더링 시스템의 예로서 Volume Rover가 있는데[7], 이 시스템은 데이타컷터(Datacutter)[8]와 스토리지 리소스 브로커(Storage Resource Broker)[9]라는 방대한 데이타의 필터링과 관리를 위한 미들웨어를 내부적으로 이용하고 있다. 고성능 그래픽스 하드웨어를 탑재한 PC상에서 구동되는 클라이언트는 텍스쳐 메모리보다 작은 크기의 볼륨 데이타를 실시간에 렌더링 하며, 고해상도의 영상을 원할 경

우 병렬 볼륨 렌더링 서버에게 렌더링을 요청할 수 있도록 설계되었다.

VR2는 볼륨 가시화 소프트웨어에 가상현실을 이용한 사용자 인터페이스를 접목시킨 시스템으로[10], 분리된 고성능의 서버에서 볼륨 렌더링을 수행하고 PC에서는 가상현실 도구를 이용해 볼륨 데이타에 대한 관찰, 조명 설정, 단면 설정 등의 기능을 제공한다. 클라이언트로 사용되는 범용 PC에서 요구되는 가상현실 시스템을 위한 계산비용이 적지 않기 때문에, 볼륨 렌더링은 네트워크로 연결되어 있는 렌더링 서버에서 수행하도록 설계되었다.

기존의 시스템들 각각의 개발 목표나 구현 방법은 조금씩 다르지만, 메모리와 프로세서의 한계로 인해 발생하는 방대한 볼륨 데이타에 대한 가시화 문제를 클라이언트-서버 개념을 이용해 해결하였다는 점에서는 공통점을 갖는다. 본 논문에서 제안하는 시스템도 클라이언트-서버 계산 모델을 기반으로 개발되었지만 다음과 같은 측면에서 기존의 시스템들과는 차별화되는 특징을 갖고 있다. 우선 휴대의 편이성과 이동의 자유로움이라는 장점을 갖고 있는 모바일 기기를 기반으로 클라이언트를 개발하였으며, 또한 동시에 여러 클라이언트가 동시에 시스템에 접속 가능할 뿐만 아니라 CSCW 환경을 지원할 수 있다. 다시 말해, 개별 사용자들이 네트워크에 접속된 고정 PC나 워크스테이션 앞에 앉아서 서버로부터 전송될 볼륨 렌더링의 수행 결과를 기다리기는 상황보다는, 서로 다른 장소에 흩어져있는 사용자들이 각자의 PDA를 통해 게이트웨이에 접속하고 하나의 세션을 만들어, 볼륨 데이타, 렌더링 컨텍스트, 렌더링 결과 영상 등을 공유하면서 협력작업이 가능한 상황을 기본적으로 고려하여 시스템을 개발하였다.

3. 모바일 볼륨 가시화 시스템의 개발

3.1 시스템의 전체적인 구조

본 논문에서 개발된 모바일 볼륨 가시화 시스템은 그림 1과 같이 크게 모바일 클라이언트, 렌더링 서버, 게이트웨이의 세 부분으로 이루어져 있다. PDA에서 실행되는 모바일 클라이언트는 사용자를 위한 볼륨 렌더링 영상을 게이트웨이를 통해 서버에게 요청하고, 렌더링 서버는 방대한 원본 볼륨 데이타를 이용해 요청된 병렬 볼륨 렌더링 작업을 수행한다. 게이트웨이는 모바일 클라이언트들과 세션(session)을 관리하며, 모바일 클라이언트와 렌더링 서버 사이에서 연결을 유지하고 작업을 조율하는 중간 계층의 역할을 담당한다. PDA를 가진 일반 사용자는 무선 네트워크에 접속할 수 있는 환경이면 시간과 장소에 구애받지 않고 간단한 메뉴 조작으로 가시화 서비스를 제공받을 수 있다. 또한 이 시스템을 이용하여 CT(Computed Tomography), MRI(Magnetic Resonance Imaging)와 같은 의료 데이타뿐만 아니라, 공학과 자연과학에서 산출된 다양한 형태의 시뮬레이션 결과들에 대해서 단면 영상, 고화질 렌더링 영상 등에 대한 가시화 서비스를 이용할 수 있다.

3.1.1 모바일 클라이언트

사용자가 클라이언트 프로그램을 실행하여 무선 랜을 통해 게이트웨이에 접속한 후, 데이타 리스트를 요청하면 게이트웨이는 저장하고 있는 볼륨 데이타들의 리스트와 함께 데이타에 대한 간단한 정보를 클라이언트에

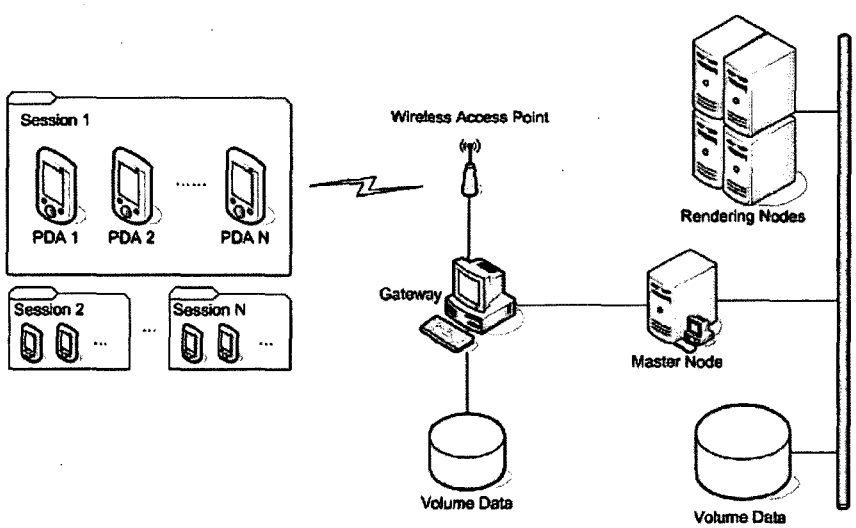


그림 1 시스템의 전체적인 구조

게 넘겨준다. 리스트 가운데 가시화 하고자 하는 볼륨 데이타를 선택하면 게이트웨이에서 원본 볼륨 데이타를 PDA의 메모리 크기에 맞는 작은 크기의 데이타로 필터링하고 이를 클라이언트에게 전송한다. 일단 모바일 클라이언트가 게이트웨이로부터 필터링 된 3차원 볼륨 데이타를 전송받으면, 텍스쳐 매핑(texture mapping) 기법을 이용하여 카메라 방향에 수직인 단면의 영상을 PDA 화면에 디스플레이 하게 된다. 사용자는 GUI상의 메뉴들을 이용하여 카메라의 방향을 자유롭게 변경할 수 있고 단면의 깊이를 바꾸어 가면서 데이타를 관찰할 수 있다. 그림 2는 모바일 클라이언트 프로그램의 각 모듈들의 구조를 보여준다. 네트워크 모듈은 모바일 클라이언트와 게이트웨이와의 통신을 담당하고, 사용자의 입력을 받아들이는 작업은 인터페이스 모듈이 담당한다. 렌더링 모듈은 현재 설정된 파라미터들에 대한 단면 영상을 PDA의 화면에 단면 영상으로 디스플레이 한다. 그리고 이러한 모듈들 전체를 총괄하고 데이타를 교환하며 명령을 내리는 컨트롤링 모듈이 존재한다.

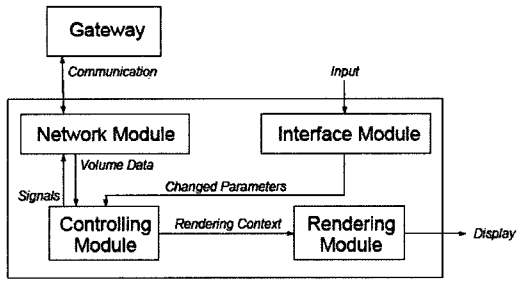


그림 2 모바일 클라이언트의 구조

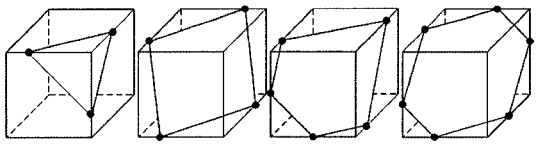


그림 3 육면체에서의 교점

실시간 볼륨 데이타를 관찰을 위해 화면상에 디스플레이 되는 단면 영상은 OpenGL ES를 이용한 2차원 텍스쳐 매핑으로 구현하였다. 사용자가 볼륨 데이타의 회전이나 깊이 평면(depth plane)의 위치를 조절하면 볼륨 데이타의 바운딩 박스(bounding box)와 깊이 평면이 교차하여 생성되는 절단면의 모양이 바뀌게 되는데, 이 때마다 절단면에 포함되는 단면 텍스쳐의 각 텍셀(texel)값을 갱신해야 한다. 이 과정은 다음과 같은 세 단계로 진행된다.

1. 볼륨 데이타의 바운딩 박스와 깊이 평면의 교점 계산
2. 교점의 정렬
3. 단면 텍스쳐 제작

직육면체 형태의 바운딩 박스와 깊이 평면이 교차할 때 각 모서리에서 생성되는 교점의 개수는 접하는 경우를 제외하면 3개에서 6개사이의 값이 된다(그림 3). 이 교점들은 3차원 공간에서 정의되어 있지만 z좌표는 사용자가 조정하는 단면의 깊이 값 $depth$와 같으므로 사실상 x축과 y 축에 대한 좌표만 계산하면 된다. 바운딩 박스의 한 모서리가 깊이 평면과 교차한다면, 그 모서리의 양 꼭지점 사이에 교점이 존재한다. 두 꼭지점의 좌표를 (x_1, y_1, z_1), (x_2, y_2, z_2)라 하고 우리가 구하고자 하는 교점의 좌표를 $(x, y, depth)$라고 놓으면, $x \in [x_1, x_2]$와 $y \in [y_1, y_2]$의 좌표는 z축에 대한 좌표의 변화에 비례하여 다음과 같이 계산된다.

$$x = x_1 + (x_2 - x_1)\frac{|depth - z_1|}{|z_2 - z_1|},$$
$$y = y_1 + (y_2 - y_1)\frac{|depth - z_1|}{|z_2 - z_1|}$$

이제 교점들을 y좌표가 가장 낮은 점부터 시계 반대방향으로 정렬해서 저장한다. y좌표가 가장 낮은 점부터 정렬하는 것은 단면 텍스쳐를 계산할 때를 대비한 것이고 시계 반대방향으로 정렬하는 이유는 OpenGL ES의 명령어를 이용해 렌더링 할 경우 꼭지점의 좌표가 시계 반대방향으로 되어있어야 원하는 모양을 화면에 나타낼 수 있기 때문이다.

단면 텍스쳐를 제작하기 위해서는 원점이 중심인 3차원 공간에서 정의된 교점들을 2차원 텍스쳐 공간으로 변환하고 이 점들을 이용해 실제 볼륨 데이타 공간의 좌표를 구해 알맞은 값을 가져와야 한다. 그림 4는 이 세 공간 사이의 대응 관계를 간략하게 보여준다. 3차원 공간에서 볼륨 데이타 공간으로의 좌표계 변환은 사용자가 볼륨 데이타를 관찰하기 위해 회전을 시키는 과정에서 계산된 회전 행렬(rotate matrix)의 역행렬(inverse matrix)을 이용한 역변환과 원점의 이동만으로 가능하다. 텍스쳐에 값을 넣는 과정은 가장 낮은 y좌표의 교점에서 가장 높은 y좌표의 교점까지 매 y좌표마다 하나씩 존재하는 스캔라인(scanline)을 렌더링함으로써 이루어진다. 이를 위해서 각 스캔라인의 시작점과 끝점의 좌표를 계산해야 하는데, 본 논문에서는 증분을 이용하여 시작점과 끝점을 계산하였다. 그림 5는 실제로 스캔라인 렌더링이 이루어지는 과정에서 시작점과 끝점을 계산하기 위해 sw, sh, ew, eh의 값을 이용하는 모습을 보여주고 있다. sw와 sh는 시작점 좌표의 가로와 세로의 증가량이고, ew와 eh는 끝점 좌표의 가로와 세로의 증가량이다. 이 증가량들은 스캔라인의 렌더링이 이루어지다가 시작점과 끝점이 각각 새로운 점을 만났을 때 새로

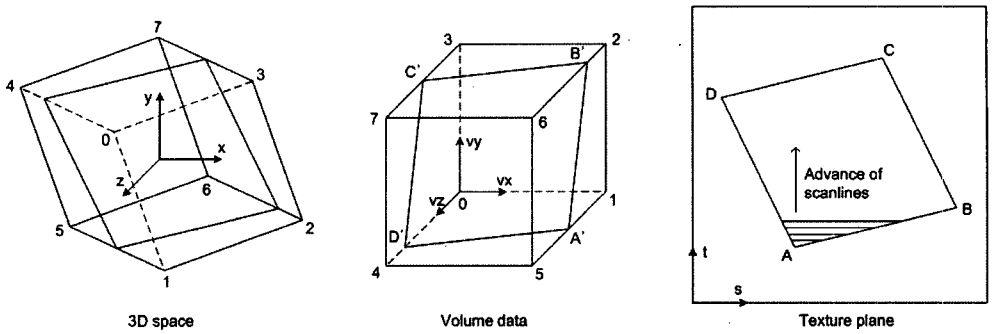


그림 4 3차원 공간, 볼륨 데이타 공간, 2차원 텍스쳐 공간 사이의 대응 관계

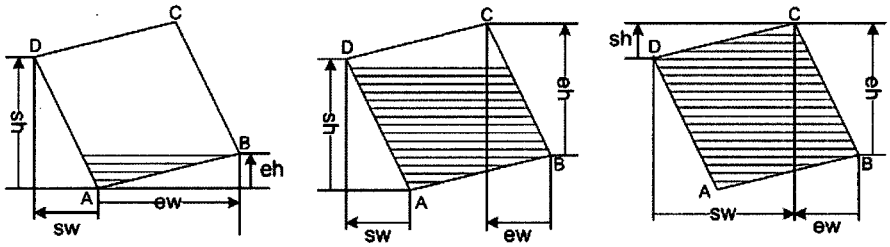


그림 5 교점이 4개인 단면 렌더링 때의 스캔라인의 변화

운 값으로 계산되고, 방향성을 가지고 있기 때문에 그림에는 화살표로 표현되어 있다. 매 스캔라인마다 시작점과 끝점에 각각의 증분을 더해주게 되는데, 시작점의 증분은 (sw/sh), 끝점의 증분은 (ew/eh)가 된다. 실제로 텍스쳐 맵을 구성하는 텍셀값을 결정하는 과정에서도 위에서 설명한 것과 동일한 증분을 이용한 계산 방법을 사용하였다.

현재 PDA 화면상에 디스플레이 중인 데이타 가운데 더욱 자세한 관찰을 원하는 부분이 있는 경우, 그 영역을 스타일러스 펜(stylus pen)을 이용하여 선택하면 해당 영역에 대한 좌표 정보가 게이트웨이에게 전송되고 게이트웨이는 고해상도의 원본 데이타로부터 선택된 부분에 대한 볼륨 데이타만을 다시 필터링하여 클라이언트에게 전송한다. 원근감 표현을 통한 자연스러운 디스플레이를 하기 위해서 OpenGL ES의 glPerspective() 함수를 이용하기 때문에, 디스플레이 화면상의 한 픽셀을 선택했을 때 해당 2차원 좌표 (x_p, y_p)를 볼륨 데이타 공간의 3차원 좌표 (x, y, z)로 매핑하는 계산하는 과정이 고려되어야 한다. 원근 투영시에 정의되는 뷰 프러스텀(view frustum)이 그림 6과 같고 PDA 화면에 대응되는 2차원 디스플레이 영역이 가까운 평면(near plane)에 대응된다고 가정할 때, z 값은 사용자가 조정하는 깊이 평면의 위치 좌표인 $depth$이므로 이미 알고 있는 값이다. 사영(projection)된 영상이 디스플레이 되는 가까운 평면(near plane)의 z좌표를 z_p라고 하면, 실제 3차

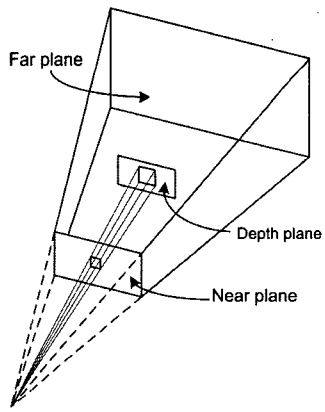


그림 6 원근 투영

원 공간에서의 좌표 (x, y, z)는 다음과 같은 비례식으로 계산 가능하다.

$$z_p : depth = x_p : x, \qquad z_p : depth = y_p : y.$$

위 식을 이용해 사용자가 선택한 영역에 대한 볼륨 데이타의 실제 좌표를 구하면 다음과 같다.

$$(x, y, z) = (\frac{x_p}{z_p} depth,\ \frac{y_p}{z_p} depth,\ depth)$$

사용자는 x, y, z 세 개의 축에 수직인 방향에서 바라본 영상을 이용하여 3차원 공간상의 광원과 카메라에 대한 위치와 벡터에 대한 렌더링 컨텍스트를 조절할 수 있으며, 설정 후 볼륨 렌더링 요청 버튼을 클릭하면 렌

더링 서버는 사용자가 설정한 볼륨 렌더링과 관련된 파라미터들을 모두 전달받아 고화질의 볼륨 렌더링 영상을 계산하고 최종 영상을 다시 클라이언트에게 전송하게 된다.

볼륨 데이타에 대한 동영상 관찰을 위해서 사용자는 모바일 클라이언트 상에서 연속적인 각 프레임에 대한 시점 변화를 지정할 수 있다. 각 시점은 4개에서 8개까지의 조정점(control point)들을 이용해 정의되는 3차원 NURBS 곡선[11]으로 지정되고(그림 7), 시점이 바라보는 방향은 원점, 즉 볼륨 데이타의 중심으로 지정하였다. 각 시점마다 시점의 업-벡터(up-vector)를 정의해 주어야 하는데, 이는 벡터의 외적을 이용해 구할 수 있다. 현재 프레임 번호를 n이라 정의할 때 n번째 프레임에서의 시점의 좌표를 A라 놓고 $n-1$번째 프레임에서의 시점의 좌표를 B라 놓으면 A에서 B로의 벡터가 정의되는데 이 벡터를 v_1이라고 한다. 그리고 A와 볼륨 데이타의 중심을 이용해 정의되는 벡터를 v_2라고 놓는다. 그러면 v_1과 v_2를 외적해서 이 두 벡터에 대해 수직으로 정의되는 벡터를 구할 수 있으며 이 벡터를 n번째 프레임에서의 업-벡터로 정의하였다. 첫 번째 프레임에서의 업-벡터는 두 번째 프레임의 업-벡터를 그대로 적용하였다. 전체 프레임에 대해 이러한 방식으로 카메라의 위치와 업-벡터를 계산하여 게이트웨이에 전송하면 이를 바탕으로 MPEG 동영상을 제작하게 된다. 동영상을 활용한 데이타 가시화는 최근 과학적 가시화 분야에서 중요한 문제로 대두되고 있는 시간가변(time-varying) 볼륨 데이타의 분석을 위해 필수적인 핵심 기능이다.

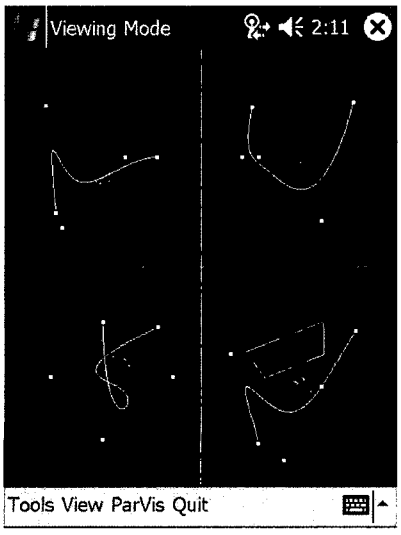


그림 7 NURBS를 이용한 카메라 이동 경로 설정

3.1.2 게이트웨이

모바일 클라이언트들과 렌더링 서버간의 직접적인 통신은 이루어지지 않는다. 모바일 클라이언트와 렌더링 서버의 중간 계층에 게이트웨이가 있으며, 모바일 클라이언트가 고해상도의 볼륨 렌더링 영상을 서버에게 요청하려면 반드시 게이트웨이에 접속하고, 게이트웨이를 통해 이런 요청을 전달해야 한다. 게이트웨이는 모바일 볼륨 가시화 시스템에서 핵심적인 역할을 담당하며, 플랫폼에 관계없이 서버와 클라이언트가 이식 가능한 환경을 제공할 수 있도록 설계되었다. 그림 8은 게이트웨이의 각 모듈들과 기능을 보여준다. 네트워크 모듈은 렌더링 서버와 모바일 클라이언트를 연결해 주는 역할을 담당한다. 볼륨 데이타 관리 모듈은 볼륨 데이타의 특정 부분을 선택하거나 레벨 단위로 필터링 하는 작업을 관리하고, 클라이언트 관리 모듈은 모바일 클라이언트들 사이의 관계를 설정하는 역할을 담당하며, 세션 관리 모듈은 CSCW를 위한 세션 유지를 책임진다. 마지막으로 위의 세 가지 관리모듈을 제어하는 제어 모듈이 존재한다.

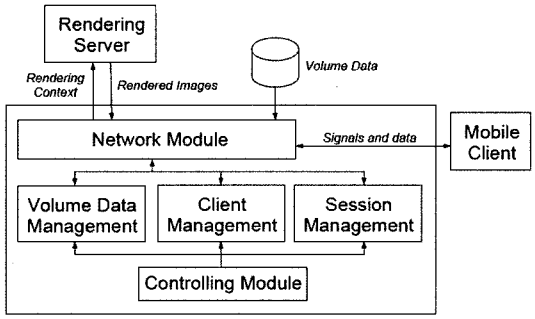


그림 8 게이트웨이의 구조

모바일 클라이언트에서 볼륨 데이타를 요청하면 게이트웨이에서 볼륨 데이타를 전송해 주어야 하는데, 이 때 게이트웨이는 PDA에 내장된 메모리 크기를 고려하여, 필터링 결과로 생성되어야 하는 볼륨 데이타의 해상도를 동적으로 계산한다. 이를 의사코드(pseudo code)로 표현하면 그림 9와 같다. PDA_MAX_MEM는 PDA의 메모리 사양에 따라 사용자가 설정하는 상수 값이다. Windows CE에서는 하나의 응용 프로그램에 대해서 최대 32 메가바이트의 메모리만을 허용하기 때문에, 본 시스템에서는 최대 16 메가바이트의 볼륨 데이타를 PDA가 저장할 수 있도록 설정하여 실험하였다. 이것은 256×256×256 해상도를 갖고 각 복셀(voxel)이 1 바이트로 표현되는 볼륨 데이타의 전체 크기에 해당한다. 그리고 나머지 16 메가바이트의 메모리는 본 프로그램의 안정적인 수행에 이용되도록 하였다. 수 백 메가바이트에서 수 십 기가바이트 단위의 볼륨 데이타들이 일반적

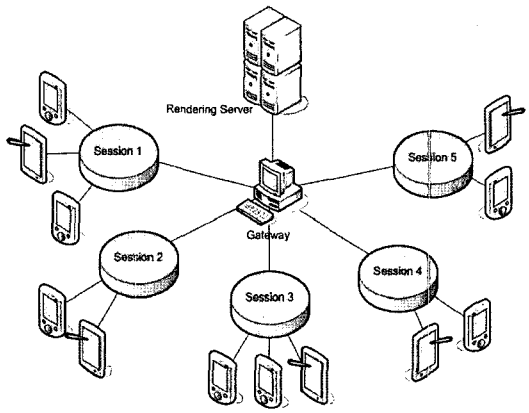
```
if(PDA_MAX_MEM < REQUESTED_VOLUME_DATASIZE)
    SetLevel(&level);
if(level != 0)
    VolumeDataFiltering(level);
send(volume_data);
```

그림 9 볼륨 데이타 필터링의 의사코드

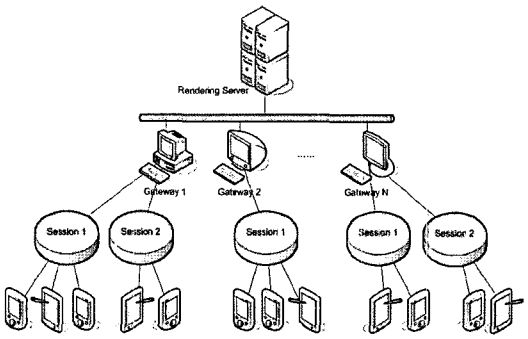
으로 사용되는 현 상황에서 16 메가바이트는 상당히 작은 메모리 단위이고, 그로 인해 PDA에서 확인할 수 있는 볼륨 데이타의 해상도는 상대적으로 상당히 낮아질 수밖에 없다. 따라서 본 시스템에서는, 현재 PDA를 통해 디스플레이 되고 있는 볼륨 데이타 가운데 더욱 자세한 관찰을 원하는 영역을 선택하면, 게이트웨이에 대응되는 부분의 볼륨 데이타에 대해서 그림 9와 같은 필터링 절차를 거쳐 PDA에 보내는 방법을 통해 점진적으로 높은 해상도의 볼륨 데이타를 이용한 관찰이 가능하도록 설계하였다.

자세함의 단계(LOD: Level Of Detail) 개념[12]의 구현을 위해 게이트웨이에서 수행되는 필터링 과정은 다음과 같다. 먼저 요청한 볼륨 데이타가 어떤 단계(level)로 필터링 되어야 하는지를 계산한다(SetLevel() 함수). 여기서 한 단계라는 것은 볼륨 데이타의 가로, 세로, 깊이가 각각 0.5배로 줄어드는 것을 의미한다. 그러므로 PDA_MAX_MEM의 값이 설정되고 선택된 볼륨 데이타의 해상도를 알고 있으면 단계의 값을 간단히 구할 수 있다. 예를 들어서, PDA_MAX_MEM의 값이 256×256×256 크기의 해상도에 대응되고, 요청한 볼륨 데이타가 1024×1024×1024의 해상도를 갖는다면 이 볼륨 데이타의 필터링에는 2단계가 필요한 것으로 계산된다. 그러면 현재의 볼륨 데이타에서 $(2^{level})^3$개 복셀들을 대상으로 필터링을 수행하여 이를 하나의 복셀로 만들고(VolumeDataFiltering() 함수), 필터링된 볼륨 데이타를 모바일 클라이언트에게 보낸다(send() 함수). 이후에 사용자가 더 높은 해상도의 부분 볼륨 데이타를 요청할 때마다 필터링 작업은 동일한 방식으로 반복 수행된다.

게이트웨이에 접속하고 있는 모바일 클라이언트의 수가 많아질수록 게이트웨이가 처리해야 할 데이타의 양도 역시 많아질 수밖에 없다. 게이트웨이가 저장하고 있는 모바일 클라이언트에 대한 정보는 클라이언트 이름, 소켓 정보, 소속된 세션 번호 등인데, 이와 같은 정보만을 유지·관리하는 것은 그리 큰 부담이 되지 않는다. 그러나 다수의 모바일 클라이언트가 동시에 하나의 게이트웨이에 접속한 상황에서, 수 백 메가바이트에서 수십 기가바이트에 이르는 볼륨 데이타를 요청하는 크기에 맞게 필터링하고 결과 데이타를 재전송하는 작업을 반복해야 한다면 게이트웨이에 심각한 병목 현상(bottleneck)이 발생할 우려가 있다(그림 10(a)). 따라서, 본 시



(a) 하나의 게이트웨이(병목현상 발생 가능성이 있는 구조)



(b) 여러 개의 게이트웨이(현재 구현된 시스템의 구조)

그림 10 다중 게이트웨이를 이용한 병목현상의 방지

스템에서는 하나의 게이트웨이에 모든 모바일 클라이언트가 집중해서 접속하는 형식이 아니라, 여러 개의 게이트웨이를 두고 각 게이트웨이가 렌더링 서버와 접속을 유지하고 있는 상태에서 클라이언트의 연결을 받아들여 세션 단위로 전체 작업을 관리 할 수 있는 구조를 설계하였다(그림 10(b)). 일단 모바일 클라이언트들이 특정 게이트웨이에 접속하면, 현재 개설중이 세션에 가입하거나 또는 새로운 세션을 생성할 수 있다. 모든 세션은 고유한 세션 이름을 갖게 되며, 해당 세션에 대한 관리는 세션 이름을 부여한 게이트웨이를 통해 이뤄지게 된다. 이러한 방식을 통해 모바일 클라이언트, 게이트웨이, 렌더링 서버 사이에 계층 관계를 설정함으로써 자연스러운 부하균형(load balancing)이 유지될 수 있다.

3.1.3 렌더링 서버

렌더링 서버는 병렬 볼륨 광선 추적법(parallel volume ray-casting)을 이용한 고화질 영상을 생성하는 것이 주된 역할이다. 서버는 클라이언트로부터 전송한 렌더링 관련 파라미터들(카메라와 조명의 위치와 방향, 가시화할 대상을 정의하는 변환함수(transfer functions), 렌더

링에 참여할 프로세서의 개수 등)을 전송받고, 사용자에 의해 설정된 특정 영역의 볼륨 데이타에 대한 정보를 받아 이를 병렬 볼륨 렌더링 알고리즘을 이용하여 가시화를 수행한다.

그림 11에서 볼 수 있듯이, 병렬 볼륨 렌더링 서버로서 PC 클러스터나 병렬 컴퓨터 등을 사용할 수 있으며, 각 노드들은 원본 볼륨 데이타가 저장된 공유 디스크를 액세스할 수 있다. 마스터 노드(master node)는 게이트웨이에게서 받은 렌더링 컨텍스트를 분석하여 원본 볼륨 데이타 가운데 렌더링 해야 할 영역이 어디인지를 결정하고 작업을 분배한다. 각 노드들은 자신에게 할당된 부분만을 렌더링 하여 영상 조각(tiled image)을 생성하고, 이를 마스터 노드에게 전송하면 마스터 노드는 모든 부분 영상 조각들을 모아 하나의 최종 영상을 만든다. 렌더링 서버의 각 노드들은 MPICH2[13]를 이용해 통신을 하며, 게이트웨이를 통해 볼륨 렌더링 요청 신호를 받으면 MPI_Comm_spawn() 함수를 호출하여 볼륨 렌더링 프로세스를 실행시킨다. 최종 렌더링 결과 영상은 생성 즉시 게이트웨이를 거쳐 해당 클라이언트로 전송되어 PDA 화면상에 디스플레이 된다.

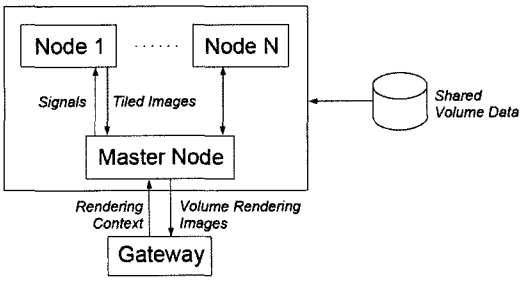


그림 11 렌더링 서버의 구조

3.2 CSCW 환경의 구현

초기의 CSCW는 컴퓨터와 전자 통신 기기를 의사 전달 매체로 사용하는 것[14]이라는 의미로 사용되었다. 그러나 급격한 속도로 발전되어 온 컴퓨터 기술, 인터넷과 인트라넷 등의 네트워크 환경의 발달, 그리고 프로젝트의 거대화 등이 맞물려 CSCW를 단순한 의사 전달의 개념으로 정의할 수는 없게 되었다. CSCW는 이제 그룹웨어(groupware)[15,16]를 통해 컴퓨터를 이용하여 공동의 관심사와 문제들을 협력 작업의 수행을 통해 더 빠르고 정확하게 해결해 나갈 수 있도록 하는 개념으로 인식되고 있다.

관련 연구 결과로서 공개적으로 설치되어 있는 대화식 화면에 사진, 동영상, 웹 페이지 등의 공유, 교환, 전시가 가능한 대화식 소프트웨어[17], 서로 다른 분야의 의사들이 참여하는 원격회의를 통한 건강관리 시스템[18] 등이 있다. 현재 CSCW의 적용 범위도 점차 확대되고 있는 추세이며, 볼륨 가시화 분야에서는 일반적인 PC를 기반으로 한 볼륨 렌더링 소프트웨어에 CSCW 개념을 접목시킨 시스템[19]이 개발되었다.

본 논문에서는 모바일 볼륨 가시화 시스템에 CSCW 개념을 접목시켜 다수의 사용자들이 동시에 같은 장소, 혹은 다른 장소에서 모바일 기기를 이용해 장소에 구애받지 않고 협력 작업을 수행하는 것이 가능하도록 구현하였다. 모든 모바일 클라이언트는 게이트웨이에 접속할 때 현재 게이트웨이 상에 개설된 세션의 리스트를 받게 된다. 리스트를 확인하고 각 세션에서 어떤 작업이 이루어지고 있는지를 확인한 후, 이미 만들어져 있는 세션에 참여하거나 혹은 세션을 새로 만들어 그 세션의 호스트(host) 권한을 가질 수 있다. 즉, 모든 모바일 클라이언트는 어떠한 세션에 속해있고 각 세션마다 호스트 권한은 한 클라이언트에게만 주어진다. 물론 클라이언트들은 현재 속한 세션을 탈퇴한 후 새로운 세션을 개설하는 것도 가능하다.

사용자들이 PDA에서 작동하는 모바일 클라이언트 프로그램을 이용해 각자 자신의 볼륨 렌더링을 위한 렌더링 컨텍스트를 작성하고 변경할 수 있으며, 호스트 클라이언트는 렌더링 컨텍스트를 브로드캐스팅(broadcasting)할 수 있는 권한을 갖는다. 브로드캐스팅을 통해서 동일한 세션에 속한 모든 모바일 클라이언트들은 호스트가 보낸 렌더링 컨텍스트를 확인하고, 그에 대한 각자의 의견을 조율할 수 있다. 브로드캐스팅은 호스트 권한을 가진 클라이언트만이 사용할 수 있기 때문에 호스트가 아닌 모든 클라이언트들은 자신이 속한 세션의 호스트에게 호스트 권한을 요청할 수 있다. 호스트는 이 요청을 승인하거나 거절할 수 있는데, 만약 호스트가 이 요청을 승인하면 호스트가 가지고 있던 권한은 호스트 권한을 요청한 클라이언트에게로 전달된다. 이러한 방식을 통해 한 세션당 한 호스트만이 존재해야 한다는 규칙이 깨지지 않으면서 모든 사용자들이 동등한 권리를 갖는 상황이 유지된다. 또한 호스트가 자신의 렌더링 컨텍스트를 브로드캐스팅할 때, 마지막으로 공유했던 렌더링 컨텍스트에서 바뀐 정보만 브로드캐스팅할 수 있게 하여 네트워크를 오가는 정보의 양을 최소화하였다.

브로드캐스트된 렌더링 컨텍스트는 게이트웨이에서 각 세션마다 따로 저장되어서 만약 새로운 클라이언트가 세션에 참가하였을 때 가장 최근에 공유된 렌더링 컨텍스트를 가지고 협력 작업을 시작할 수 있도록 하였다. 또한 각 클라이언트들은 호스트가 보낸 렌더링 컨텍스트와는 별개로 자신의 렌더링 컨텍스트를 독립적으로 유지할 수 있도록 하였으며, 이것은 게이트웨이가 아닌 자신의 PDA의 로컬 메모리에 저장된다. 이러한 기법을

통해 자신의 렌더링 컨텍스트와 브로드캐스트된 렌더링 컨텍스트를 서로 비교·분석하여 자신의 의견과 공동 작업자들의 의견을 조율할 수 있다.

호스트 권한을 가진 모바일 클라이언트는 자신이 정의한 렌더링 컨텍스트를 바탕으로 얻은 볼륨 렌더링 영상을 자신이 속한 세션의 모든 클라이언트들에게 전송할 수 있다. 모바일 클라이언트들이 볼륨 렌더링 영상을 요청하면 볼륨 렌더링 서버에서 제작된 영상이 게이트웨이를 통해 전송이 되는데, 이때 게이트웨이에서 각 클라이언트들의 권한을 확인하고 만약 호스트에서 요청한 볼륨 렌더링 영상이라면 해당 볼륨 렌더링 영상을 그 호스트가 속한 세션 데이타에 저장해 놓는다. 그러면 추후 호스트에서 볼륨 렌더링 영상을 브로드캐스팅 하겠다고 신호를 보냈을 때 PDA에서 게이트웨이로 볼륨 렌더링 영상이 전송될 필요 없이 게이트웨이에 이미 저장되어 있는 볼륨 렌더링 영상 데이타를 각 클라이언트들에게 보내줄 수 있다. 이러한 방식으로 불필요한 데이타의 통신을 막을 수 있다.

4. 실험결과와 성능분석

모든 그래픽스 소프트웨어는 빠른 렌더링 속도를 중요시한다. 본 시스템은 제한된 하드웨어 사양을 갖는 PDA를 클라이언트로 사용하여 상대적으로 대용량의 볼륨 데이타를 다루기 때문에 최적화 문제가 더욱 중요하다. 본 절에서는 ARM 프로세서를 사용하는 PDA를 기반으로 실행되는 모바일 클라이언트에서 사용된 최적화 기법들과 최적화로 인해 향상된 성능을 살펴보고, 시스템의 실행 결과에 대해 설명한다.

4.1 사용된 최적화 기법들

대부분의 ARM 프로세서 기반의 모바일 시스템에는 부동 소수점(floating point) 연산 하드웨어가 장착되어 있지 않기 때문에, 부동 소수점 연산 속도는 고정 소수점(fixed point) 연산에 비해 상대적으로 느리다. 부동 소수점 연산 하드웨어를 장착해서 연산 속도를 향상시키는 것이 가능하지만 아직까지 이것은 하드웨어 가격의 증가뿐만 아니라, 모바일 기기에서 중요하게 인식되는 높은 전력 소비의 문제를 발생시킨다. 하지만 고정 소수점 연산만으로 부동 소수점 연산에 근접한 정확도를 가지면서 빠른 연산 속도를 얻을 수 있다. 본 시스템에서는 4바이트 정수 변수를 기반으로 16-비트 고정 소수점 방식을 사용하였으며, 나눗셈과 곱셈 연산의 경우 최대한의 정확도를 유지하기 위해서 임시로 64-비트 정수 변수를 사용하였다.

컴퓨터 그래픽스에서 삼각함수는 매우 자주 쓰이지만 일반적으로 계산 시간이 오래 걸리는 단점이 있다. 따라서 전처리 과정에서 0도에서 90도 사이의 각도에 대해

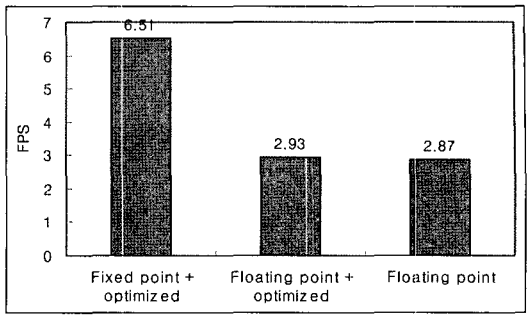


그림 12 최적화 여부에 따른 1초당 프레임 수 비교

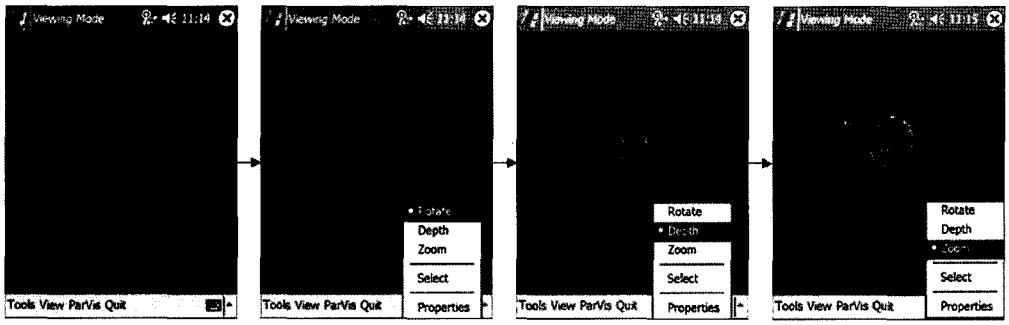
삼각함수 참조 테이블을 미리 만들어 놓고, 필요시에 값을 참조하는 방법을 사용하였다. 1사분면에 해당하는 삼각함수 값만을 저장하여도 나머지 각도에 대한 함수 값을 간단히 구할 수 있다. 그 밖에 반복문의 효율적인 처리, 부호 있는 변수에 대한 효과적인 나눗셈 연산 등도 고려하였다. 이 모든 최적화 부분들은 어셈블리 코드를 ARM 프로세서에 맞게 구현하여 사용하였다.

그림 12는 최적화 기법의 사용 여부에 따른 1초당 프레임 수(FPS: Frames Per Seconds)를 보여준다. 코드의 모든 부동 소수점 연산(Floating point)을 고정 소수점 연산(Fixed point)으로 교체한 것만으로도 큰 성능 향상이 이루어졌다는 것을 볼 수 있다. 하지만 위에서 언급된 그 외의 최적화 기법들(optimized)에 대한 성능 향상은 매우 미미한 편이다.

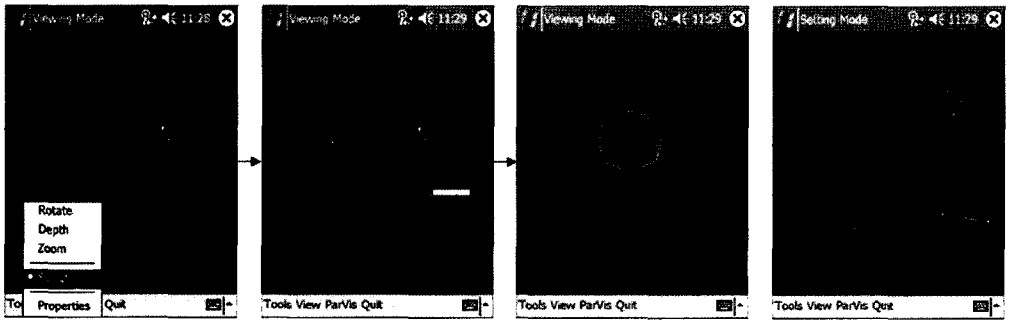
4.2 실험 결과

클라이언트 프로그램 개발을 위해 Intel PXA270 (624MHz), 64MB SDRAM를 내장하고, VGA(480×640) 디스플레이를 지원하는, MS Windows Mobile 2003 기반의 HP iPAQ hx4700 PDA 시스템이 이용되었다. 또한, GUI 구현을 위해 MS Embedded Visual C++ 4.0, Vincent OpenGL ES[20], 그리고 서버와의 통신을 위한 코드 개발을 위해 MS Winsock을 활용하였다. 게이트웨이 프로그램은 펜티엄4 2.0GHz CPU에 1GB 메모리를 내장한 PC에 이식되었다. 그리고 렌더링 서버 프로그램은 SGI Onyx2 병렬 시스템과 PC 클러스터 시스템에 이식되어 테스트되었는데, 이것은 [5]에서 개발된 병렬 볼륨 광선추적 코드를 기반으로 한다. 본 논문에서는 실험을 위해 15개의 노드로 구성된 리눅스 기반 PC 클러스터를 사용하였으며, 각 노드들은 dual Intel Pentium4(3.0Ghz)와 2GB 메모리를 탑재하고 있다. 각 노드들 사이의 통신을 위해 MPICH2가 이용되었다.

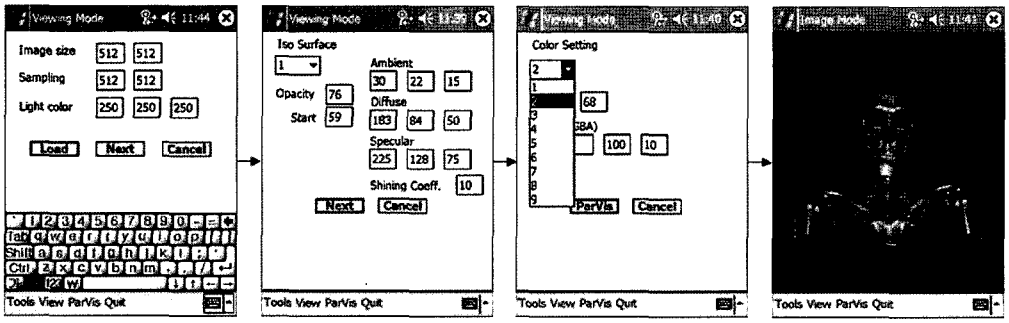
그림 13은 모바일 클라이언트의 실행 화면을 캡춰한 것이다. 사용자는 자세한 관찰을 위해 볼륨 데이타를 회전, 깊이 평면의 변경, 확대 및 축소 작업을 하고(그림 13(a)), 더욱 자세한 관찰을 위해 세부 영역을 선택하며



(a) 볼륨 데이타의 제어



(b) 부분 볼륨 데이타의 요청



(c) 렌더링 컨텍스트 전송 및 볼륨 렌더링 영상 요청

그림 13 모바일 클라이언트의 실행화면

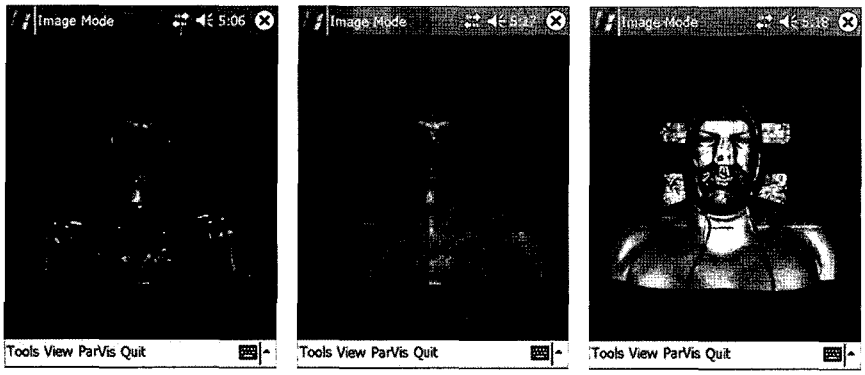


그림 14 볼륨 렌더링 영상들

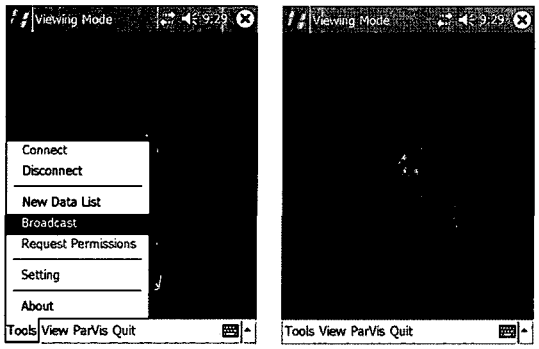
(그림 13(b)), 렌더링 컨텍스트를 조절하여 렌더링 서버에세 볼륨 렌더링을 요청하고 결과 영상을 디스플레이 한다(그림 13(c)).

그림 14는 모바일 클라이언트의 요청에 따라 렌더링 서버에서 볼륨 렌더링을 수행하여 가시화된 영상을 PDA를 통해 디스플레이하는 상황을 보여준다. 사용자의 적절한 렌더링 컨텍스트 조절을 통해 다양한 볼륨 데이타 영상을 얻을 수 있음을 확인할 수 있다.

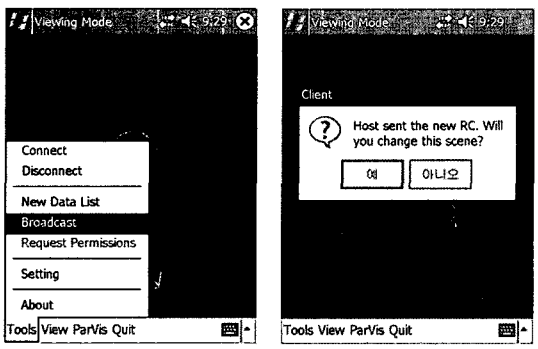
그림 15와 그림 16은 두 대의 PDA를 이용하여 협력 작업을 하는 과정을 설명한다. 각각의 그림들은 위에서 언급했던 렌더링 컨텍스트를 공유하는 과정과 하나의 모바일 클라이언트가 호스트 권한을 취득하는 과정을 단계별로 보여준다.

그림 17을 통해 PDA에서 다루는 볼륨 데이타의 크기가 렌더링 성능에 어떠한 영향을 미치는지 확인할 수 있다. 볼륨 데이타의 크기가 작을수록 프레임 수가 높아지는 것은 당연하지만, 2~16 메가바이트 크기의 데이타 사이에는 그리 큰 차이가 없음을 알 수 있다.

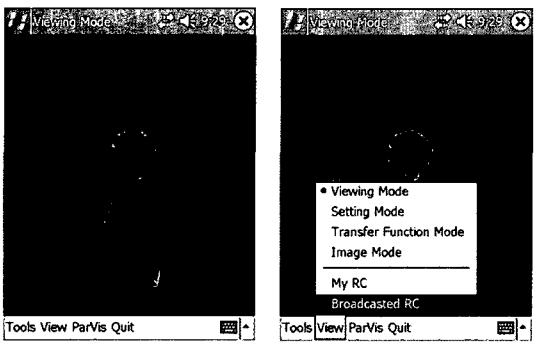
본 시스템과 같이 단면 영상의 렌더링을 통해 볼륨 데이타를 관찰하는 경우, 화면에 디스플레이 되는 픽셀



(a) 렌더링 컨텍스트의 브로드캐스트

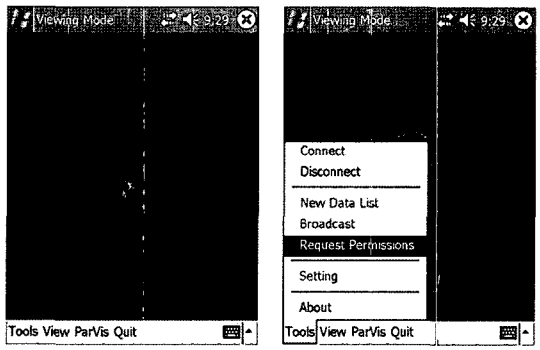


(b) 클라이언트의 승인 시 렌더링 컨텍스트를 저장

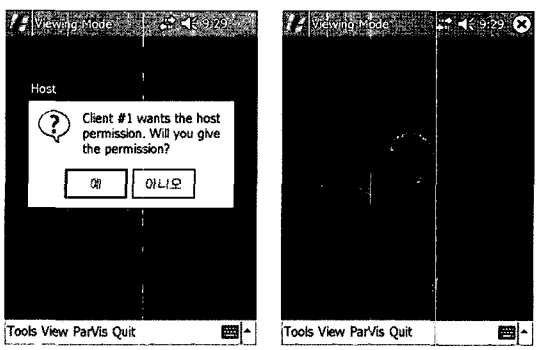


(c) 자신의 렌더링 컨텍스트와는 별개로 저장됨

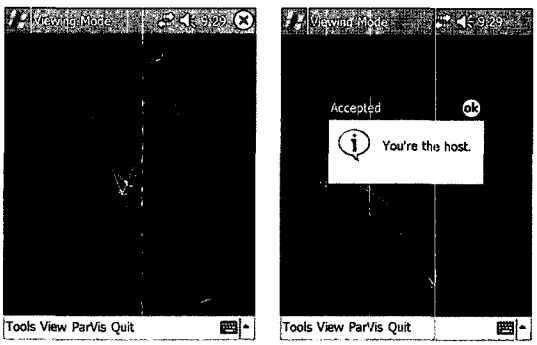
그림 15 렌더링 컨텍스트의 공유



(a) 클라이언트에서 호스트 권한을 요청



(b) 호스트의 승인 시 권한이 이동됨



(c) 호스트와 클라이언트의 권한이 변경됨

그림 16 호스트 권한의 요청

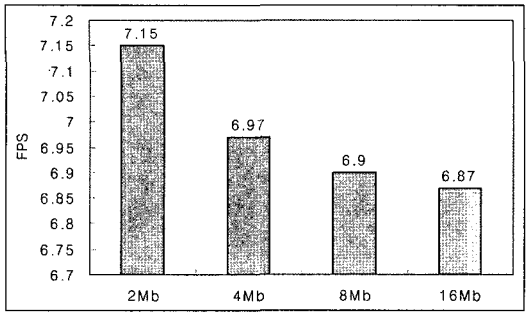


그림 17 볼륨 데이타의 크기에 따른 1초당 프레임 수 비교

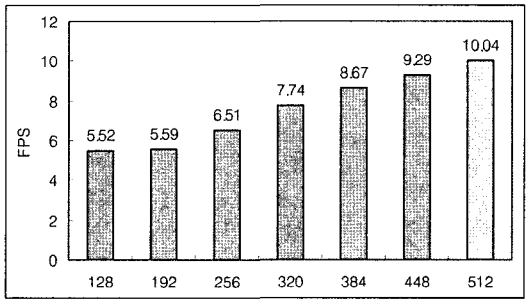


그림 18 단면의 거리에 따른 1초당 프레임 수 비교

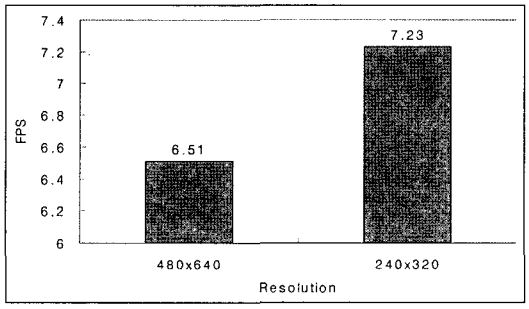
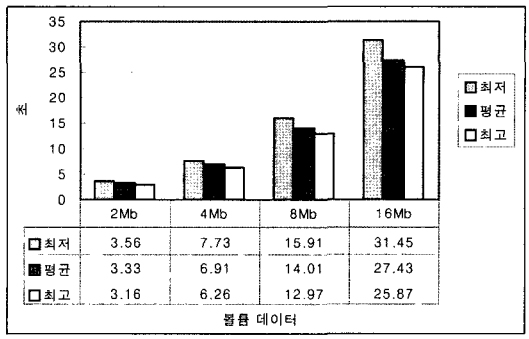
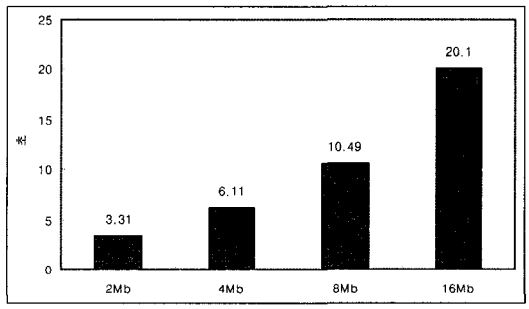


그림 19 디스플레이 창의 크기에 따른 성능 비교

의 개수에 따라 렌더링의 성능 차이가 발생한다. 즉, 그림 18에서 확인할 수 있는 바와 같이 단면 렌더링 영상을 가깝게 설정해서 더 많은 픽셀들이 그려지도록, 혹은 멀리에서 보도록 설정하여 적은 수의 픽셀만이 그려지도록 하는 실험을 통해 성능 차이를 확인할 수 있는데, 렌더링 속도의 변화폭이 비교적 큰 것을 알 수 있다. 만일, 3D 그래픽스 가속칩을 장착한 모바일 기기에서 동일한 실험을 수행한다면 훨씬 향상된 성능을 예상할 수 있다. 그림 19는 동일한 볼륨 데이타와 렌더링 컨텍스트를 이용한 상황에서, 디스플레이 해상도가 각각 480×640, 240×320인 두 PDA를 이용하여 렌더링 성능 분석을 한 결과를 보여준다. 실제로 240×320 디스플레이 해상도를 갖는 PDA가 상대적으로 성능이 낮은 CPU을 장착한



(a) 무선랜을 이용한 전송



(b) 로컬 네트워크를 이용한 전송

그림 20 통신의 종류에 따른 볼륨 데이타 전송 속도

모델임에도 불구하고, 디스플레이 영상의 화질은 떨어지지만 렌더링 속도는 오히려 더 빠름을 확인할 수 있다.

그림 20에서는 사용된 통신의 차이에 따른 데이타 전송 속도를 나타내고 있다. 결과에서 볼 수 있듯이 무선랜을 이용한 통신이 더 느리고 더 불안정하게 이루어지고 있음을 확인할 수 있다.

5. 결론 및 향후연구

본 논문에서는 모바일 환경에서 수행되는 클라이언트-서버 개념을 이용한 볼륨 렌더링 시스템의 설계 및 구현에 대해 설명하였다. 개발된 시스템은 볼륨 데이타 서버와의 접속, 원하는 볼륨 데이타의 선택, 볼륨 데이타의 미리보기, 특정 부분에 대한 LOD 관찰, 기본적인 볼륨 데이타 조작(회전, 축소, 확대 등), 렌더링 컨텍스트의 설정, 고화질 볼륨 렌더링 요청 등과 같은 데이타 가시화를 위한 다양한 기능을 포함하고 있으며, 이를 용이하게 수행할 수 있는 편리한 GUI 환경을 제공한다. 또한 CSCW 개념을 도입하여 다수의 사용자가 동시에 효율적인 협력 작업을 수행할 수 있는 환경을 구현하였으며, 카메라의 연속적인 이동경로 설정을 통해 동영상을 제작하여 시간 흐름에 따른 데이타의 분석이 가능하도록 설계하였다.

이 시스템은 여러 분야로의 확장 가능성을 고려하여 구현되었기 때문에 특정 분야에서 필요로 하는 기능들을 추가하면 다양한 응용 프로그램으로 발전시키는 것이 가능하다. 예를 들어 본 시스템에 데이타베이스 시스템과의 연동 기능을 추가함으로써, 전문적인 의료 데이타의 관리와 가시화를 지원할 수 있을 뿐만 아니라 시간과 장소의 제약 없이 협력 진료가 가능한 의료 환경을 구축할 수 있을 것이다. 또한 계산 비용의 부담 때문에 모바일 기기에 적용하기 힘든 광선 추적 렌더러(ray tracing renderer)와 같은 컴퓨터 그래픽스 관련 소프트웨어들을, 본 논문에서 설명된 모바일 클라이언트-병렬 렌더링 서버 개념을 이용해 구현한다면 모바일 기기를 이용해 일반적인 PC나 워크스테이션에서와 마찬가지로 고화질의 사진과 같은 영상(photo-realistic image)을 받아 보는 것이 가능할 것이다.

실험 결과 무선 네트워크 속도가 시스템의 전체적인 성능에 영향을 미치는 가장 중요한 요소이다. 현재 일반적으로 사용되는 무선 환경은 유선 네트워크에 비해 속도가 느리고 위치에 따라 속도의 변화가 심하다는 문제점을 갖고 있으나, 최근의 네트워크 기술의 발전 속도를 감안할 때 이러한 기술적인 문제는 조만간 해결될 수 있을 것으로 기대한다.

참 고 문 헌

[1] C. Hansen and C. Johnson (Ed.), *The Visualization Handbook*, Elsevier Academic Press, 2005.

[2] M. Beyon, C. Chang, U. Catalyurek, T. Kurc, A. Sussman, H. Andrade, R. Ferreira, and J. Saltz, "Processing large-scale multidimensional data in parallel and distributed environments," *Parallel Computing*, pp. 827-859, May 2002.

[3] A. Chervenak, I. Forster, C. Kesselman, C. Salisbury and S. Tuecke, "The data grid: towards an architecture for the distributed management and analysis of large scientific datasets," *Journal of Network and Computer Applications*, Vol. 23, pp. 187-200, 2000.

[4] K.-L. Ma and S. Parker, "Massively parallel software rendering for visualizing large-scale data sets," *IEEE Computer Graphics and Applications*, Vol. 21, No. 4, pp. 72-83, 2001.

[5] C. Bajaj, I. Ihm, G. Koo and S. Park, "Parallel ray casting of visible human on distributed memory architectures," In *Proceedings of VisSym '99 (Joint EUROGRAPHICS-IEEE TCCG Symposium on Visualization)*, pp. 269-276, May 1999.

[6] D. Koller, M. Turitzin, M. Levoy, M. Marini, G. Croccia, P. Cignoni and R. Scopigno, "Protected interactive 3D graphics via remote rendering," *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004)*, Vol. 23, No. 3, pp. 695-703, 2004.

[7] Computational Visualization Center, http://www.ices.utexas.edu/cvc, University of Texas at Austin, 2005.

[8] T. Kurc, U. Caralyrec, C. Chang, A. Sussman and J. Saltz, "Visualization of large data sets with the active data repository," *IEEE Computer Graphics and Applications*, Vol. 21, No. 4, pp. 24-33, 2001.

[9] A. Rajasekar, M. Wan and R. Moore, "MtSRB & SRB - components of a data grid," In *Proceedings of Symposium on Distributed Computing*, July 2002.

[10] A. Fuhrmann, B. Ozer, L. Mroz and H. Hauser, "VR2 interactive volume rendering using PC-based virtual reality," Tech. Rep 2002-014. VRVis, 2002.

[11] P. Schneider and D. Eberly, *Geometric Tools for Computer Graphics*, Morgan Kaufmann, 2002.

[12] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson and R. Huebner, *Level of Detail for 3D Graphics*, Morgan Kaufmann, 2002.

[13] MPICH2, http://www-unix.mcs.anl.gov/mpi/mpich2/index.htm, Argonne National Laboratory, 2005.

[14] B. Johnson, G. Weaver, M. Olson and R. Dunham, "Using a computer-based tool to support collaboration: a field experiment," In *Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 343-353, 1986.

[15] C. Bullen and J. Bennet, "Learning from user experiences with groupware," In *Proceedings of the Conference on Computer Supported Cooperative Work*, pp. 291-302, 1990.

[16] C. Ellis, S. Gibbs and G. Rein, "Groupware - some issues and experiences," *Communications of the ACM*, Vol. 34, No. 1, pp. 39-58, 1991.

[17] H. Brignull, S. Izadi, G. Fitzpatrick, Y. Rogers and T. Rodden, "The introduction of a shared interactive surface into a communal space," In *Proceedings of ACM Conference on Computer Supported Cooperative Work*, pp. 49-58, 2004.

[18] B. Kane, S. Luz, G. Menezes and D. Hollywood, "Enabling change in healthcare structures through teleconferencing," In *Proc. of the 18th IEEE Symposium on CBMS*, 2005.

[19] K. Baek and I. Ihm, "SGVR: a collaborative volume visualization system," *Journal of Korea Information Science Society (A)*, Vol. 24, No. 5, pp. 417-428, 1997.

[20] Vincent: A 3-D Rendering Library for Mobile Devices, http://ogl-es.sourceforge.net, 2005.

박 상 훈

1993년 8월 서강대학교 수학과 졸업(이학사). 1995년 8월 서강대학교 컴퓨터학과 졸업(공학석사). 2000년 2월 서강대학교 컴퓨터학과 졸업(공학박사). 2000년 3월~2000년 6월 서강대학교 컴퓨터학과 박사후연구원. 2000년 7월~2002년 8월 University of Texas at Austin의 TICAM (Texas Institute of Computational and Applied Mathematics) 박사후연구원. 2002년 9월~2005년 2월 대구가톨릭대학교 컴퓨터정보통신공학부 조교수. 2005년 3월~현재 동국대학교 영상대학원 멀티미디어학과 조교수. 관심분야는 컴퓨터 그래픽스, 과학적 가시화, 고성능 컴퓨팅, 모바일 컴퓨팅 등

김 원 태

2004년 서강대학교 컴퓨터학과 졸업(공학사). 2006년 서강대학교 대학원 컴퓨터학과 졸업(공학석사). 2006년~현재 케이앤아이테크놀로지 기술연구소 재직중. 관심분야는 Embedded Computer Graphics, Programmable GPU.

임 인 성

1985년 2월 서울대학교 자연과학대학 계산통계학과 졸업(이학사). 1987년 5월 Rutgers~The State University of New Jersey 컴퓨터학과 졸업(이학석사). 1991년 7월 Purdue University 컴퓨터학과 졸업(이학박사). 1999년 7월~2000년 6월 University of Texas at Austin의 TICAM 연구 교수 1993년 3월~현재 서강대학교 컴퓨터학과 교수. 관심분야는 컴퓨터 그래픽스, 과학적 가시화, 고성능 계산 등임