역/촉감 제시 “K-Touch” 햅틱 API 개발

이범찬, 김종필, 류제하
광주과학기술원 인간-기계-컴퓨터 인터페이스 연구실
{bclee, lowtar, ryu}@gist.ac.kr

Development of K-Touch haptic API (Application Programming Interface)

Beom-Chan Lee, Jong-Phil Kim and Jeha Ryu
Human-Machine-Computer Interface Lab
Department of Mechatronics,
Gwangju Institute of Science and Technology

요 약

본 논문은 새로운 햅틱 API 인 “K-Touch”의 개발에 관한 것이다. 그래픽 하드웨어 기반의 핵심 역감 알고리즘을 기반으로 개발된 K-Touch API 는 가상 환경을 구성하는 다양한 데이터 형식(3D polygon model, volume data, 2.5D depth image)에 대한 햅틱 상호작용을 가능하게 하고, 새로운 햅틱 알고리즘 및 장치 개발에 필요한 소프트웨어 확장성을 제공함과 동시에 사용자가 쉽고 빠르게 햅틱 응용분야를 개발할 수 있도록 설계되었다. 아울러 햅틱 감각의 중요 요소인 역감 및 촉감 상호작용을 위해 기존의 햅틱 SDK 및 API 와 달리 역/촉감을 동시에 제시할 수 있는 알고리즘이 개발되었다. 본 논문에서 제안하는 새로운 햅틱 API 의 효용성을 검증하기 위해 다양한 응용분야의 예를 구현하였다. 새로운 햅틱 API 인 K-Touch 는 사용자 및 연구자에게 보다 효율적으로 햅틱 연구를 진행 할 수 있도록 도움을 주는 툴(Tool)로써 중요한 역할을 할 것으로 기대된다.

Keyword: Haptics, Kinesthetic, Tactile, Haptic API.

1. 서 론

햅틱(Haptic)이란 단어는 고대 그리스 어의 “*haptikos/haptesthai*”로부터 유래되었으며, 햅틱 감각은 손을 사용하여 느끼는 모든 지각을 의미하는 용어로 정의되어 사용되었으나, 최근 인체의 모든 촉감 기관을 사용하여 느끼는 지각으로 의미가 넓혀지고 있다. 햅틱 렌더링은 사용자에게 가상 혹은 증강된 환경의 다양한 객체의 정보를 역/촉감을 통해 제공해주는 일련의 계산 과정을 의미한다. 햅틱 인터페이스를 통해 제시되는 역/촉감 정보는 다양한 환경을 실감 있게 인지하도록 도와주며, 사용자의 몰입감을 증가시켜 환경내의 객체들을 효과적으로 느끼고 조작할 수 있도록 도와준다. 햅틱 렌더링은 크게 역감 상호작용과 촉감 제시로 나눌 수 있는데, 전자는 물체의 형상, 굳기, 변형 정보를 역감 제시 인터페이스를 통하여 사용자에게 제시되는 근감각적 정보를 의미하는 반면, 후자는 사용자의 피부에 분포되어 있는 감각 수용기에 촉감 제시 인터페이스 통해 물체의 미세 형상, 거칠기, 마찰, 온도 등을 피부에 직접 정보를 전달하는 것을 의미한다. 따라서 햅틱스에 관한 연구는 역감 또는 촉감 정보를 효율적으로 표현하는 연구를 중심으로 발전해왔으며, 이러한 상호작용은 의료, 군사, 산업, 교육 및 게임 등에 접목되어 발전해왔다.

일반적으로 시각 제시(Graphic rendering)는 30Hz Update 성능으로 가상 환경의 콘텐트를 부드럽게 제시할 수 있지만, 햅틱 제시(Haptic rendering)의 경우 인체의 촉감 인지 체계의 특성상 1KHz 이상

으로 Update 되어야 원하는 역감을 얻으면서 안정된 촉감 상호작용을 제공할 수 있다[1]. 따라서 햅틱 연구분야에서는 빠른 성능으로 가상 환경과의 충돌 검출(Collision detection) 및 반력 계산(Contact force computation)을 수행할 수 있는 다양한 햅틱 렌더링 알고리즘이 연구되어왔다. 그러나 이러한 햅틱 렌더링 알고리즘은 복잡한 구조적 계산 과정을 갖고 있기 때문에 햅틱 연구 분야에 대해 생소한 응용 분야 개발자에게는 추가의 노력을 통한 개발이 불가피했었다. 따라서 전 세계적으로 햅틱 사용을 용이하게 하기 위해 쉽고 빠르게 햅틱 환경을 구축할 수 있는 SDK (Software Development Kits) 또는 API (Application Programming Interface)등이 개발되어 왔다. 이러한 인터페이스 툴의 개발은 시/청각에 촉감을 가미한 여러 응용 분야에 이용이 되어왔다. 그러나 기존의 SDK 및 API 의 경우, 전형적으로 가상 환경을 구성하는 3 차원 Polygon 모델에 대한 상호작용을 제공하였고, 역감을 통한 상호작용만 제공되었다. 또한 소수의 API 를 제외하고는 개발자가 직접 SDK 또는 API 의 구조를 변경하여 새로운 알고리즘 및 장치를 추가할 수 있도록 지원이 되지 않았다.

본 논문에서는 기존의 SDK 및 API 에 나타난 제한 점을 극복하고 보완하는 새로운 햅틱 API 인 K-Touch 의 개발을 제안한다. K-Touch 의 핵심 알고리즘은 본 연구팀에서 제안한 그래픽 하드웨어 기반의 알고리즘으로써 가상 환경 내에 존재하는 다양한 데이터 표현(3D polygon, volume data, 2.5 D depth image)에 대해 역감 상호작용이 가능하다 [2]. 또한 본 API 는 햅틱 감각의 중요 요소인 촉감 렌더링 알고리즘을 개발하여 역/촉감 상호작용을 동시에 가능하도록 설계 되었다. 아울러 체계적이고 모듈화 된 소프트웨어 구조를 설계하여 개발자에게 새로운 알고리즘 및 장비를 확장 할 수 있도록 설계 되었으며, 햅틱 연구 분야에 익숙하지 않는 응용 프로그램 개발자에게 보다 쉽고 빠르게 햅틱 환경을 구축할 수 있도록 고안되었다. 따라서 본 K-Touch 는 다양한 햅틱 연구 및 응용 분야에 유용한 툴로써 중요한 역할을 기대할 수 있으며, 역/촉감 상호작용을 동시에 할 수 있으므로 가상 환경과의 상호작용 시 사용자에게 다양한 환경을 더 자세히 인지하도록 도와주며, 사용자의 몰입감을 증대시켜 효과적으로 가상 환경 내 객체들을 느끼고 조작할 수 있도록 도와 줄 것이라 기대한다.

본 논문은 다음과 같이 구성되어있다. 다양한 햅틱 SDK 및 API 에 대한 관련 연구 동향 및 K-Touch API 의 개발 필요성을 2 장에 설명하고, K-Touch 에 구현된 핵심 역/촉감 렌더링 알고리즘 및 K-Touch 소프트웨어 구조 및 설명을 3 장에서 언급하고, 4 장에서는 본 API 의 최적화에 관하여 설명한다. K-Touch API 를 이용한 응용 분야의 구현을 통해 본 API 의 효용성을 5 장에서 검증하고, 마지막으로 본 논문의 결론 및 향후 연구 계획을 6 장에 기술하였다.

2. 관련 연구 동향 및 연구 동기

햅틱 연구분야에서 가장 널리 알려진 SDK 로는 Sensable 사의 GHOST SDK 이다[3]. GHOST SDK 는 최초의 햅틱 SDK 로써 PHANTOM 햅틱 인터페이스를 이용한 햅틱 상호작용 및 응용 분야를 개발할 수 있도록 개발되어 햅틱 연구 분야에 널리 사용되어왔다. 그러나 GHOST 의 경우 surface-based 역감 알고리즘[4,5]이 구현되어 volume 데이터 또는 2.5D depth image 에 대한 역감 상호작용을 제공할 수 없으며, 대부분의 핵심 소프트웨어를 접근하기 어렵기 때문에 새로운 알고리즘 및 장치 지원에 대해 제한적이다. 따라서 보다 유연한 햅틱 툴을 제공하기 위해 2004 년 OpenHaptics Toolkit 이 같은 회사에서 개발되었다[3]. OpenHaptics Toolkit 은 HDAPI, HLAPI 로 나뉘어져 있으며, HDAPI 는 개발자에게 PHANTOM 장비를 이용하여 알고리즘을 개발할 수 있도록 지원하는 디바이스 API 이고, HLAPI 는 그래픽스 프로그래머들에게 쉽게 햅틱 환경을 구축할 수 있도록 OpenGL 과 비슷한 형식을 갖는 High level API 라고 할 수 있다. 그러나 OpenHaptics 의 경우도 surface-based 역감 제시 알고리즘[4,5]이 구현되어 있기 때문에 다양한 데이터 지원이 불가능하며, 핵심 알고리즘을 수정하거나 새로운 알고리즘으로 대체하는데 어려운 구조

를 갖고 있다. 또한 GHOST 및 OpenHaptics 는 보다 사실적인 상호작용을 제공하는 그래픽/햅틱 co-location 을 제공하지 못하는 단점이 있다. 앞서 언급된 햅틱 SDK 와 달리 그래픽/햅틱 co-location 을 제공하는 API 로는 Novint 사의 e-Touch API [6], RechIn 사의 ReachIn API [7] 및 Sensgraphics 사의 H3DTH[8] 가 개발되었다. e-Touch API, ReachIn API 는 VRML 파일 포멧을 지원하며 다양한 응용분야 개발에 사용될 수 있도록 상용화 된 제품이고, H3DTH 는 3 차원 그래픽스를 위한 ISO Standard 인 X3D 기반으로 구현되어 무료로 제공된다. 그러나 세가지 API 의 핵심 역감 제시 알고리즘은 Sensable 사의 GHOST 및 OpenHaptics 의 surface-based 알고리즘을 사용하기 때문에 다양한 데이터 지원에 대해 제한적이다. 또한 새로운 햅틱 렌더링 알고리즘 및 장치를 연결할 수 있는 확장성이 제한적이다. 보다 확장성 있는 햅틱 API 를 제공하기 위해 최근 CHAI 가 Stanford Univ.에서 개발되었다[9]. CHAI 는 햅틱 기술에 익숙하거나 또는 햅틱 응용분야에 관심이 있는 연구자를 위해 개발된 C++기반의 햅틱 Library 라 할 수 있으며, 새로운 알고리즘 및 장치를 추가 할 수 있도록 체계적으로 소프트웨어가 구성이 되어있다. 또한 PHANTOM[3], DELTA[10], OMEGA[10], MPB 6S[11] 등 다양한 햅틱 장비를 지원한다. 그러나 CHAI Library 역시 surface-based[12] 의 역감 제시 알고리즘이 구현되어 있기 때문에 다양한 데이터에 대한 역감 상호작용을 지원하지 못하는 단점이 있다.

표 1 기존 햅틱 SDK 및 API 특징

	GHOST	Open Haptics	ReachIn	e-Touch	H3D	CHAI
가상환경	VRML	VRML	VRML	VRML	VRML /XML	Obj, 3ds
역감 알고리즘	Surface based	Surface based	Surface based	Surface based	Surface based	Surface based
촉감 알고리즘	×	×	×	×	×	×
알고리즘 및 장비 확장성	×	×	×	×	×	o
지원 장비	Phantom	Phan-tom	Phantom Delta	Phantom Delta	Phantom	Phantom Delta
Haptic /grahic colloca-tion	×	×	o	o	o	×
데이터 표현 방법	Polygon model	Polygon model	Polygon model	Polygon model	Polygon model	Polygon model
햅틱 환경 구조	Scene graph	Like OpenGL	VRML Scene graph	Modular structure	Unified scene graph	Scene graph

표 1 은 기존의 햅틱 SDK 및 API 의 특징을 나타내고 있다. 앞서 언급한 각 햅틱 SDK 및 API 의 단점을 극복하여 다양한 데이터(3D polygon, volume data, 2.5D depth image)에 대한 햅틱 상호작용을 지원하며, 새로운 알고리즘 및 장치개발을 위한 확장성을 제공함과 동시에 역/촉감 상호작용을 동시에 제공할 수 있는 새로운 햅틱 API 개발의 필요성으로 본 논문에서는 그래픽 하드웨어 기반의 역감 제시 알고리즘을 핵심 기반으로 한 새로운 K-Touch 햅틱 API 개발을 언급하도록 하겠다.

3. K-Touch Haptic API

본 장은 K-Touch 의 핵심 역/촉감 렌더링 알고리즘을 설명하고 소프트웨어 구조, 구현된 class hierarchy 및 각 class 의 역할에 대해 상세히 기술한다.

3-1. 역감 렌더링 알고리즘

K-Touch 의 역감 렌더링 알고리즘은 3 자유도 역감 렌더링에 필요한 IHIP (ideal haptic interaction point) 주변의 기하학적 정보를 각 픽셀에 할당된 깊이 버퍼 값을 참조하여 얻어낸다. 가상 물체의 깊이 정보를 얻기 위해 햅틱 워크스페이스의 6 면 (top,bottom,front,back,left,right) 끝에 위치한 가상 카메라를 이용하여 IHIP 주변의 객체를 렌더링 한다. 이때 형상 정보와 관련 없는 color, texture, light 효과는 제거된다. 여기서 IHIP 는 이상적인 HIP 를 의미하며, 충돌 검출 이전에는 HIP 를 정확히 따르다가 충돌 검출 이후 HIP 에서 분리되어 객체의 surface 에 머무른다. 이렇게 국부적으로 렌더링 된 그래픽 컨택스트의 깊이 정보는 가상카메라의 파라미터를 통해 객체의 6 면에 대한 기하학적 정보로 변환된다. 이 정보를 이용하여 HIP 가 객체 내부에 있는지를 검사하며 이것은 바로 충돌 검출로 이어진다. 또한 이러한 깊이 정보는 LOMI(Local Occupancy Map Instance)를 생성하는데 사용된다. LOMI 는 국부적이고 실시간으로 갱신되는 occupancy map 으로 육면체의 3 차원 격자의 셀들로 구성되어있다. 각 셀들은 IHIP 주변 객체의 일부분에

대한 내부(interior), 외부(free space), 표면(surface)의 정보를 포함한다. LOMI 는 충돌검출 후 반력 계산을 위해 사용되며, Voxmap[13]과 비슷한 데이터형식을 갖는다. 그러나 LOMI 는 IHIP 주변의 국부적 기하형상만을 표현하고, 사이즈가 매우 작아 실시간 연산이 가능하다는 점에서 차이가 있다. 이 LOMI 를 이용하여 충돌검출 이후 IHIP 를 HIP 에 가장 가까운 surface voxel 에 위치시킬 수 있으며 이를 통해 반력의 크기와 방향을 결정해 줄 수 있다. 그림 1 은 가상 물체를 IHIP 위치에 따라 6 개의 각 가상카메라로 본 것을 나타낸다.

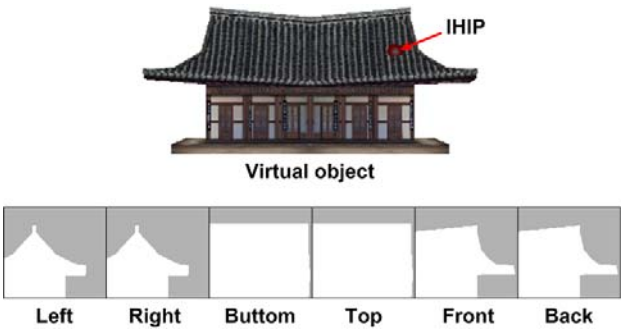

그림 1. IHIP 위치에 상응하는 가상 카메라 렌더링

역감 렌더링의 중요 요소인 충돌 검출(Collision detection)을 위해 6 개의 가상 카메라로부터 획득되는 물체의 깊이 정보를 이용하여 HIP 가 가상 물체의 내부에 위치하는 지를 판단하여 쉽게 충돌을 검사할 수 있다. 그림 2 는 z 축에 위치한 2 개의 가상 카메라로부터 얻어지는 깊이 비교를 통한 충돌 검지의 예를 보여주고 있으며, 3 차원 충돌 검지는 남은 4 개의 가상카메라로부터 얻어지는 깊이 정보를 함께 이용하여 수행된다.

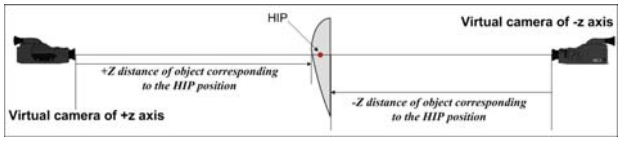

그림 2. z 축 방향에서의 HIP 와 물체의 충돌 검출

앞서 언급한 간단한 충돌 검지는 얇은 물체나 빠르게 움직이는 HIP 와 물체의 충돌을 정확히 검출하지 못하는 단점을 갖고 있다. 두 단점을 보완하기 위해 현재 HIP 와 이전의 IHIP 사이를 보간 방법을 사용하여 정확한 충돌을 검출 한다. 그림 3(a)는 현 HIP 와 이전 IHIP 의 연결 선을 보여주고 있으며 이 연결선은 LOMI 가 갖고 있는 volume 크기로 나뉘어서 물체와의 충돌을 검출하게 된다. 그림 3(b)에서 보듯이 검출된 점이 임시 IHIP 가 된다.

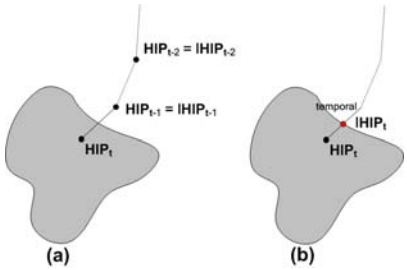

그림 3. 보간 방법에 의한 충돌 검출

반력 계산에는 LOMI 가 사용되는데, LOMI 는 IHIP 를 중심으로 하는 6 개의 가상카메라로부터 획득되는 물체의 국부 지역의 깊이 정보를 표현한다. 시간 t 에서 충돌이 검출되면 HIP_t 는 가상 물체의 내부로 들어가는 반면 $IHIP_t$ 는 이전 t-1 의 $IHIP_{t-1}$ 에 의해 생성된 $LOMI_{t-1}$ 를 참조하여 $LOMI_{t-1}$ 의 셀 중에서 HIP_t 와 가장 가까운 거리에 위치한 surface voxel 에 남게 된다. 다음 t+1 에서 HIP 가 새로운 위치 HIP_{t+1} 로 이동하게 되면 $IHIP_{t+1}$ 위치는 이전 시간에 생성된 $LOMI_t$ 의 가상 물체의 표면을 나타내는 셀들에 의해 결정되고 시간 t+1 에서의 반력의 크기와 방향은 3 차원 벡터 HIP_{t+1} 와 $IHIP_{t+1}$ 에 의해 계산된다. 그림 4 은 반력 계산 및 LOMI 에 의해 IHIP 위치 결정 과정을 보여준다.

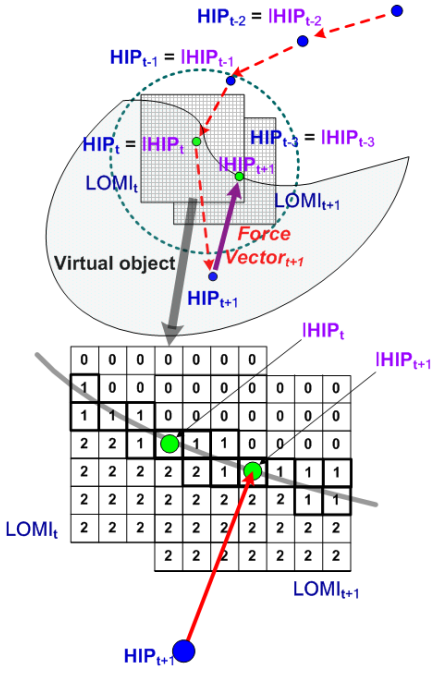

그림 4. LOMI 를 이용한 IHIP 결정 및 반력 계산

3-2. 촉감 렌더링 알고리즘

일반적으로 촉감 제시에 관한 연구는 어떤 정보를 어디에 무엇을 이용하여 제시할 것인지에 대한 연구가 주로 이루어졌다. 촉감 제시는 주로 electromagnetic technologies, pneumatic, shape memory alloys, piezoelectric, electro tactile 시스템 등을 이용하여 손가락, 등, 발, 팔뚝 등에 가상 물체의 성질, 경보, 방향 정보 또는 의사소통의 수단으로 연구가 진행되었다[14]. 그림 5 는 일반적 촉감 제시를 나타낸다.

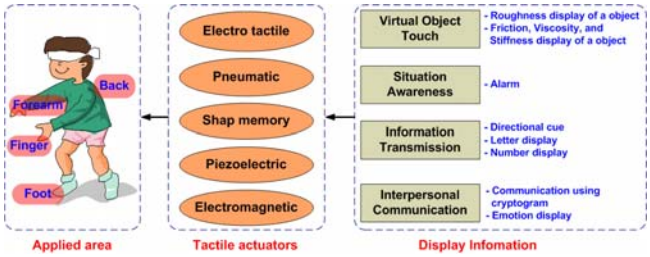

그림 5. 촉감 제시

비록 인체의 다양한 부분에 적용하는 촉감 제시 시스템이 개발되었지만 주로 감각 수용기가 민감하게 분포되어 있는 손가락에 적용되어 가상 물체의 거칠기, 마찰 및 미세 형상을 표현에 관한 연구가 진행되었다. 연구에 의하면 인체의 손가락에 분포된 감각 수용기는 평균적으로 약 250Hz 의 주파수대의 진동을 느끼며[15], 200-700 μm 의 높이 차이를 인식 하는 것으로 조사되었다[16]. 따라서 다양한 촉감 제시 장치를 이용하여 정보를 표현하지만 촉감 Actuator 의 주파수 및 진폭을 조절하며 특정 정보를 표현하는 방법은 같다. 따라서 본 논문에서는 일반적인 촉감 제시 장치를 위한 High 및 Low level 촉감 알고리즘을 개발하고, 검증하기 위해 촉감 제시 시스템을 구축하였다. 그림 6 은 구현된 촉감 제시 구성 및 촉감 정보 형식을 보여준다.

촉감 정보는 가상 물체와 상호작용 시 촉감 제시 장치를 이용하여 전달 할 수 있는 물체의 거칠기, 마찰 및 미세 형상을 주파수 및 진폭을 조절할 수 있는 알고리즘을 개발하였다. 일반적인 촉감 제시 시스템은 DSP(Digital Signal Processing) 칩 또는 Micro-controller 를 사용하는데 본 논문에서는 촉감 actuator 를 제어할 수 있는 ATmega 128 Micro-controller 를 이용하였다. 촉감 actuator 는 voice coil 형식의 actuator 를 이용하였으며 2×2 배열을 구성하였다.

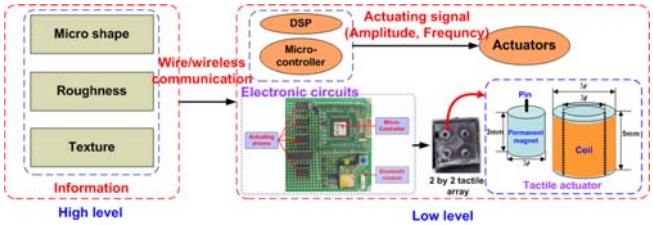

그림 6. 촉감 제시 시스템 구성도

3-3. K-Touch Haptic API 구조

K-Touch API 는 C++기반의 햅틱/그래픽 class 들로 구성되었으며, 햅틱 연구자 및 햅틱을 이용한 응용 분야를 개발하는 개발자를 대상으로 개발된 햅틱 API 이다. K-Touch API 의 소프트웨어 구조는 사용의 편의성, 효율성, 확장성 및 역/촉감 동시 상호작용 지원이 고려되어 설계되었다. 첫째로 편의성 측면에서, 본 API 를 통해 사용자는 햅틱 상호작용의 절차에 대해 깊은 이해 없이도 빠르고 쉽게 햅틱 환경을 구축할 수 있도록 구성되었다. 따라서 사용자는 몇 줄의 C++ Code 를 이용하여 원하는 목적의 햅틱 응용 프로그램을 작성할 수 있다. K-Touch API 의 효율성을 높이기 위해 Scene graph 구조를 갖는 Architecture 를 구성하여 다양한 객체에 대해 체계적이고 효과적으로 햅틱 상호작용이 가능하게 하였으며, 그래픽/햅틱 최적화를 통해 복잡한 가상 환경도 햅틱 상호작용이 가능하도록 구성 하였다. 또한 새로운 알고리즘 및 장치를

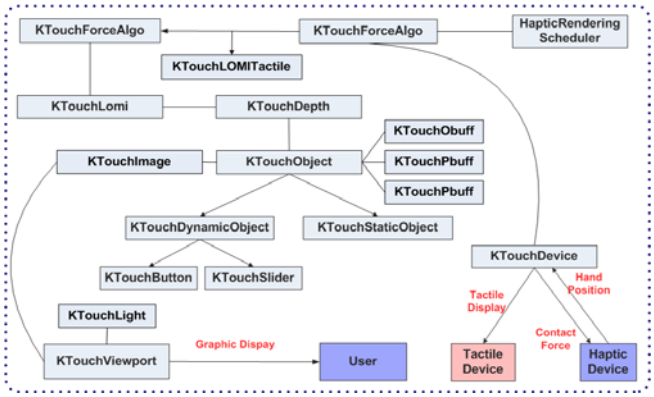

그림 7 은 K-Touch API 의 동작 구조

추가할 수 있도록 각 class 를 체계적으로 모듈화 하려는 노력이 진행되었을 뿐 아니라 역/촉감을 동시에 제시할 수 있는 구조를 설계하여 가상 환경을 사실적으로 인식할 수 있도록 고안하였다.

그림 7 은 K-Touch API 의 동작 구조를 나타내고 있다. KTouchObject 는 가상 환경 내 존재하는 물체의 형상 정보를 나타내며, 정적 또는 동적 물체로 구분되어 표현된다. KTouchImage 는 2.5D depth image 에 대한 햅틱 상호작용을 위한 것으로, KTouchImage 및 KTouchObject 에 정의된 가상 객체는 OpenGL 기반의 그래픽 렌더링을 담당하는 KTouchViewport class 에 사용자에게 제시된다. 그래픽과 더불어 햅틱 효과를 위해 6 개의 가상카메라로부터 획득되는 물체의 깊이 정보는 KTouchDepth class 에 저장 된다. 가상 물체의 깊이 정보를 획득하는 방법에는 크게 3 가지가 있는데, 그래픽 하드웨어의 Frame 버퍼 또는 Pixel 버퍼를 이용하여 깊이 정보를 획득하는 방법이 있으며, voxel 데이터 구조로 정의된 물체의 경우 Object 버퍼에 저장이 되어 충돌 검출 및 반력 연산에 이용된다. 가상 물체와 HIP 와의 충돌에 상응하는 반력을 연산하기 위해 KTouchLomi class 는 LOMI 를 생성하며 KTouchLOMIForceAlgo class 에 의해 충돌 검출 및 반력 연산이 수행된다. KTouchForceAlgo class 는 역/촉감 class 를 관장하며 높은 우선순위를 갖는 HapticRendering Scheduler 에 의해 1KHz 의 Update 되며 장비를 관장하는 KTouchDevice class 를 통해 역/촉감 정보를 제시하게 된다.

3-4. K-Touch Haptic API class hierarchy

K-Touch API 는 효과적으로 가상 객체 및 햅틱 상호작용을 가능하게 하기 위해 2 개의 Hierarchical classes 가 구현되었다. 그림 8 은 KTouchScene 및 KTouchForceAlgo class hierarchy 를 보여준다.

KTouchScene 은 Scene graph 를 구성하는 기본으로 이 class 로부터 햅틱 장치, 햅틱 환경을 위한 world class 및 가상 객체 클래스가 파생이 된다. KTouchWorld 는 햅틱 Scene graph 를 구성하는 최상위 노드로 사용이 되며 가상 환경 내 존재하는 객체의 그래픽/햅틱 제시를 담당한다. KTouchObject 는 정적 및 동적 특징을 갖는 물체의 class 로 파생이 되며, 정적 물체는 다시 버튼 및 슬라이더 class 로 파생이 되어 가상 환경 내 존재하는 물체를 정의한다. KTouchDevice 는 햅틱 장치의 추상 class 로써 역/촉감 장비 class 에 파생이 된다. 만약 새로운 장비를 추가하고자 한다면 이 class 로부터 상속을 받아 추가할 수 있다.

KTouchForceAlgo 은 역/촉감 렌더링 하위 class 를 포함하는 최상위 추상 class 로써 역/촉감 알고리즘 class 는 이 class 로부터 상속을 받아 생성이 된다. 본 API 에 구현된 역감 상호작용 알고리즘은 3 자유도 기반이기 때문에 LOMI 를 이용한 역감 상호작용 알고리즘은 KTouch3dForceAlgo class 로부터 파생되어 정의 되었다. 촉감 제시 알고리즘은 가상 물체의 거칠기, 마찰 및 미세 형상을 표현할 수 있도록 KTouchTexureAlgo class 에 촉감 장비의 주파수 및 진폭을 조절할 수 있도록 구현되어있다. 예를 들어, 6 자유도 역감 상호작용 알고리즘을 추가하게 되며 KTouchForceAlgo 로부터 파생하면 되고, 새로운 촉감 제시 알고리즘을 추가하고자 한다면 KTouchTactileAlgo 로부터 파생하여

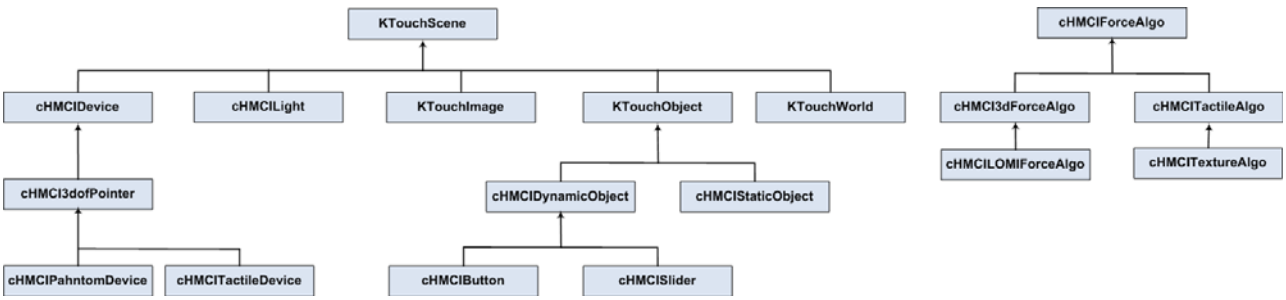

그림 8. KTouchScene 및 KTouchForceAlgo class hierarchy

새로운 class 를 생성하면 된다.

4. K-Touch Haptic API 최적화

K-Touch API 에 구현된 핵심 역감 상호작용 알고리즘은 그래픽 하드웨어를 이용하기 때문에 그래픽 렌더링을 빠르게 수행할 수 있어야 하며, 1KHz 이내로 충돌 검출 및 반력 연산을 수행하여야 한다. 따라서 효율적인 햅틱 상호작용 제시를 위해 본 장에서는 K-Touch API 의 그래픽/햅틱 최적화 문제를 언급한다.

4-1. 그래픽 렌더링 최적화

가상 물체를 OpenGL 을 이용하여 그래픽 렌더링 하는 경우 일반적으로 컴퓨터 그래픽스 분야에서 vertex array, indexed geometry, display list, vertex buffer object 등의 방법을 사용한다.

Vertex array 는 가상 물체를 구성하는 vertices, faces, texture coordinates, surface normal, color 정보를 array 형태로 구성하여 그래픽 렌더링 하는 방법이다. Indexed geometry 의 경우 가상 물체를 이루고 있는 점 데이터를 공유하는 메쉬의 인텍스 정보만을 이용하여 렌더링 하는 것이기 때문에 모든 점 데이터를 이용하여 그래픽 렌더링 하는 방법보다 효율적이라 할 수 있다. Display list 는 물체의 기하정보를 프로그램 캐시 메모리에 올려서 그래픽 렌더링을 하기 때문에 일반적으로 앞의 방법 보다는 효율성이 뛰어나다고 할 수 있다. 마지막으로 Vertex buffer object 는 OpenGL 의 확장 명령어를 사용함으로써 물체의 기하 정보를 그래픽스 하드웨어 메모리에 올리고 그래픽스 하드웨어 자체에서 렌더링 하는 기법을 말한다. Display list 와 Vertex buffer object 의 가장 큰 특징은, 전자는 메모리에 등록된 물체 기하정보를 실시간으로 바꿀 수 없지만 후자는 실시간으로 기하정보를 바꿀 수 있다는 것이다. 일반적인 그래픽 렌더링의 효율성을 알아보기 위해 가상 환경에 존재하는 물체의 메쉬 수를 증가 하면서 렌더링 시간을 측정하였다. 표 2 는 각 방법에 대해 가상 물체를 구성하고 있는 메쉬 수를 증가시키면서 그래픽 렌더링 수행 결과이다. 측정된 시간은 millisecond 단위이며 각 방법에 대해 100 번의 시간 측정 후 평균을 구한 값이다. 시간 측정 결과를 토대로 가상 물체는 Vertex buffer object 와 Indexed geometry 방법을 이용하여 그래픽 렌더링 함으로써 효율을 높였다.

표 2. 각 렌더링 방법의 효율

	가상 물체의 메쉬 수				
렌더링 방법	**200K**	**300K**	**400K**	**500K**	**600K**
Vertex Array	17.32	33.31	33.65	49.97	50.14
Indexed Geometry	16.82	33.31	33.31	34.66	49.98
Display List	16.65	16.65	16.65	16.65	16.65
Vertex Buffer Object	16.65	16.65	16.65	16.65	16.65

4-1. 햅틱 렌더링 최적화

K-Touch API 구현된 역감 상호작용 알고리즘은 연산 시간이 가상 환경을 구성하는 객체의 복잡도에 의존되지 않고 LOMI 에 의해 결정된다. 즉, LOMI 를 구성하는 voxel 의 수에 따라 역감 렌더링의 효율이 결정된다. LOMI 는 격자 단위인 voxel 로 이루어진 구조로써 voxel 크기 및 수로 LOMI 가 형성된다. LOMI 는 물체 표면에 위치하는 IHIP 를 경정하는 것으로 IHIP 를 결정하기 위해 매 햅틱 렌더링 시간마다 LOMI 를 탐색하여 IHIP 를 결정하게 된다. 탐색의 효율을 높이기 위해 LOMI 는 각 면에 대해 3 등분 되어 IHIP 가 될 수 있는 후보(Candidate)의 영역이 존재하는 부분만 탐색을 실행하게 된다. 따라서 LOMI 전체를 탐색하는 것보다 효율이 높다. 표 3 은 LOMI 의 voxel 크기와 수를 변화하며 충돌 검지 및 반력 계산에 소요되는 시간을 측정한 결과이다.

표 3 역감 상호작용에 소요되는 연산 시간

	LOMI 의 Voxel 수					
Voxel size[mm]	**11^3**	**21^3**	**41^3**	**61^3**	**101^3**	**201^3**
0.15	0.081	0.081	0.071	0.083	0.091	0.083
0.2	0.089	0.076	0.074	0.072	0.089	0.089
0.25	0.081	0.083	0.083	0.083	0.085	0.087
0.3	0.075	0.078	0.074	0.076	0.075	0.092

0.35	0.082	0.079	0.084	0.095	0.087	0.076
0.4	0.078	0.073	0.076	0.076	0.073	0.093
0.45	0.088	0.082	0.086	0.087	0.085	0.095
0.5	0.08	0.096	0.08	0.085	0.093	0.097

표 3 의 결과에서 나타나듯 많은 수의 voxel 을 이루고 있는 LOMI 가 구성이 되어도 약 0.9 milli-second 의 연산 시간이 소요된다는 것을 알 수 있다. 따라서 1KHz 이상의 역감 렌더링을 수행할 수 있다.

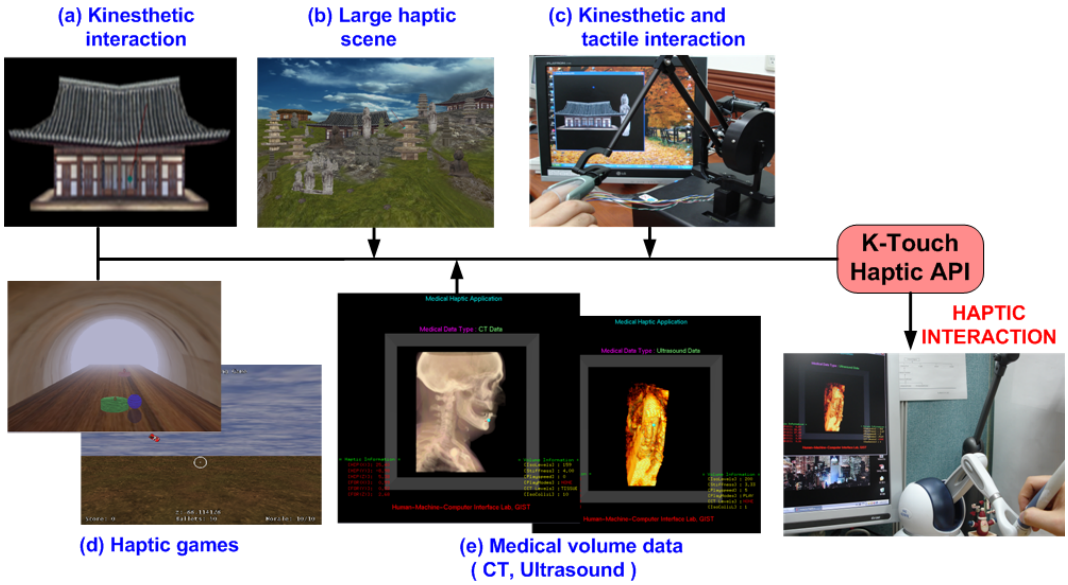

그림 9 K-Touch API 를 이용한 응용 분야의 예

5. 응용 분야 구현

본 논문에서 제안한 K-Touch API 의 효용성을 검증하기 위해 다양한 예를 구성하였다. 구현된 예는 PHANTOM Omni 햅틱 인터페이스[3]와 펜티엄 4 3.2GHz 의 CPU, GeForce 6600 그래픽스 하드웨어 기반에 구현되었다. 그림 9(a)는 일반적인 역감 상호작용을 나타내는 것으로 물체의 형상을 힘을 이용하여 느낄 수 있으며, 그림 9(b)는 복잡한 가상 환경에 대한 역감 상호작용을 나타내고 있다. 가상 환경을 구성하는 물체들의 총 메쉬 수는 약 600K 로 구성되어 있다. 그림 9(c)는 두 물체의 다른 촉감을 느끼며 역/촉감 상호작용을 하는 것을 보인다. 이때 손가락에 2×2 촉감 장비에 의해 물체의 거칠기가 제시 된다. 그림 9(d)는 역감이 가미된 게임 콘텐트를 나타내는 것으로 마치 총을 쏘는 듯한 힘을 제시 받으며 게임을 할 수 있는 Shooting 게임과 퍽과 공이 부딪칠 때 힘이 느껴지는 Haptic Air-hockey 게임이다. 마지막으로 그림 9(e)는 의료 영상에 대한 햅틱 상호작용을 나타내고 있다. 의료 영상은 주로 CT, MRI, 초음파 측정에 의해 획득된 Volume 데이터 이다. 본 API 는 3 차원 Polygon 데이터뿐 아니라 3 차원 Volume 데이터에 대한 역/촉감 상호작용이 가능하다.

6. 결론 및 향후 연구 계획

본 논문에서는 새로운 햅틱 API 인 K-Touch 를 제안 하였다. 본 API 는 C++기반의 햅틱/그래픽 class 로 구성되었으며, 햅틱 연구자 및 햅틱을 이용한 응용 분야를 개발하는 개발자를 대상으로 개발된 햅틱 API 이다. K-Touch API 의 소프트웨어 구조 디자인은 사용의 편의성, 효율성, 확장성 및 역/촉감 동시 상호작용 지원이 고려되어 설계되었다. 본 API 의 가장 뚜렷한 특징은 다양한 데이터에 대해 역/촉감 상호작용이 가능하다는 것이다. 따라서 전형적인 3 차원 Polygon 모델을 포함하여 Volume 데이터 및 Z-cam(뎁스 카메라)을 통해 획득된 2.5D depth image 에 대해서도 햅틱 상호작용을 제공한다. 아울러 기존의 SDK 및 API 와 달리 역/촉감을 동시에 제시할 수 있는 알고리즘 및 소프트

웨어 구조를 개발하여 역감 및 촉감 상호작용을 통해 사용자가 보다 효과적으로 환경내의 객체들을 느끼고 조작할 수 있도록 개발되었다. 따라서 본 API 는 햅틱을 이용한 응용 분야 개발자 및 햅틱 기술을 연구하는 연구자에게 효과적인 툴로 이용될 것으로 기대된다.

그러나 제안된 K-Touch API 는 개발 초기 단계기 때문에 보다 완성도 높은 햅틱 API 를 구성하기 위해 향후 수행해야 할 연구 과제가 있다. 우선 보다 몰입감 있는 체험 환경을 제공할 수 있도록 그래픽/햅틱 co-location 시스템을 개발할 것이며, 다양한 햅틱 장비를 지원하기 위해 새로운 장비 class 를 구현할 것이다. 아울러 알고리즘 측면에서는 6 자유도 역감 상호작용이 가능한 알고리즘 개발을 수행할 것이며, 보다 사실적인 물체 표면 정보를 제시 하기 위해 촉감 데이터 획득을 정교하게 수행할 것이다. 마지막으로 사용자가 쉽게 햅틱 환경을 생성하고, 수정하고, 저장할 수 있는 HUI(Haptic User Interface)를 개발하여 보다 많은 사람들이 쉽게 햅틱 환경을 구축할 수 있도록 제공할 것이다.

감사의 글

본 연구는 정보통신부 선도기반기술개발사업(차세대 PC 기술개발), 광주과학기술원 실감방송연구센터(RBRC), 과학기술부 실감모델링 개발사업 지원에 의해 수행되었음.

참고문헌

[1] K. Salisbury, F. Barbagli, and F. Conti, “Haptic Rendering: Introductory Concepts”, *IEEE Computer Graphics and Applications*, vol. 24, no. 2, pp. 24-32, 2004.

[2] Jong-Phil Kim, Beom-Chan Lee, and Jeha Ryu, “Haptic Rendering with Six Virtual Cameras”, *HCI international 2005*, pp. 467, 2005.

[3] SensAble Technologies Inc., www.sensable.com

[4] Salisbury, J., Brock, D., Massie, T., Swarup, N., and Zilles, C, “Haptic rendering: programming touch interaction with virtual objects” *Symposium on Interactive 3D Graphics*, pp. 123-130, 1995.

[5] Zilles, C.B., Salisbury, J.K., “A constraint-based god-object method for haptic display**”,** *Int. Conf. Proc. IEEE/RSJ*, vol. 3, pp. 146-151, 1995.

[6] Novint, www.novint.com

[7] ReachIn , www.reachin.se

[8] SenseGraphics Inc., http://www.sensegraphics.com

[9] CHAI 3D, http://www.chai3d.org

[10] Force dimension, http://www.forcedimenstion.com

[11] MPB Inc., http://www.mpb-technologies.ca

[12] S. Walker and K. Salisbury “Large Haptic Topographic maps: Marsview and the Proxy Graph Algorithm”, *Proc. ACM Symp. Interactive 3D graphics*, pp. 83-92, 2003.

[13] McNeely, W., Puterbaugh, K., and J. Troy. “Six degreeof-freedom haptic rendering using voxel sampling”, *Int. Conf. roc. Proc. ACM SIGGRAPH*, pp. 401–408, 1999.

[14] Benali Khoudja M., Hafez M., Alexandre J.M., and Kheddar A., “Tactile Interfaces - A State of the Art Survey”, *International Symposium on Robotics*, 2004.

[15] Kontarinis, D.A., Howe R.D., “Tactile Display of Contact Shape in Dexterous Telemanipulation”, *ASME Advances in Robotic, Mechatronics and Haptic Interfaces*, vol. 49, pp. 81-88, 1993.

[16] G. Werner and V. B. Mountcastle, “Quantitative Relations Between Mechanical Stimuli to the Skin and Neural Responses Evoked by them”, *in: D. Kenshalo (ed.) The Skin Senses*, pp. 112-138, 1968.