

햅틱스(Haptics)

Haptics Rendering



에이치 브이알

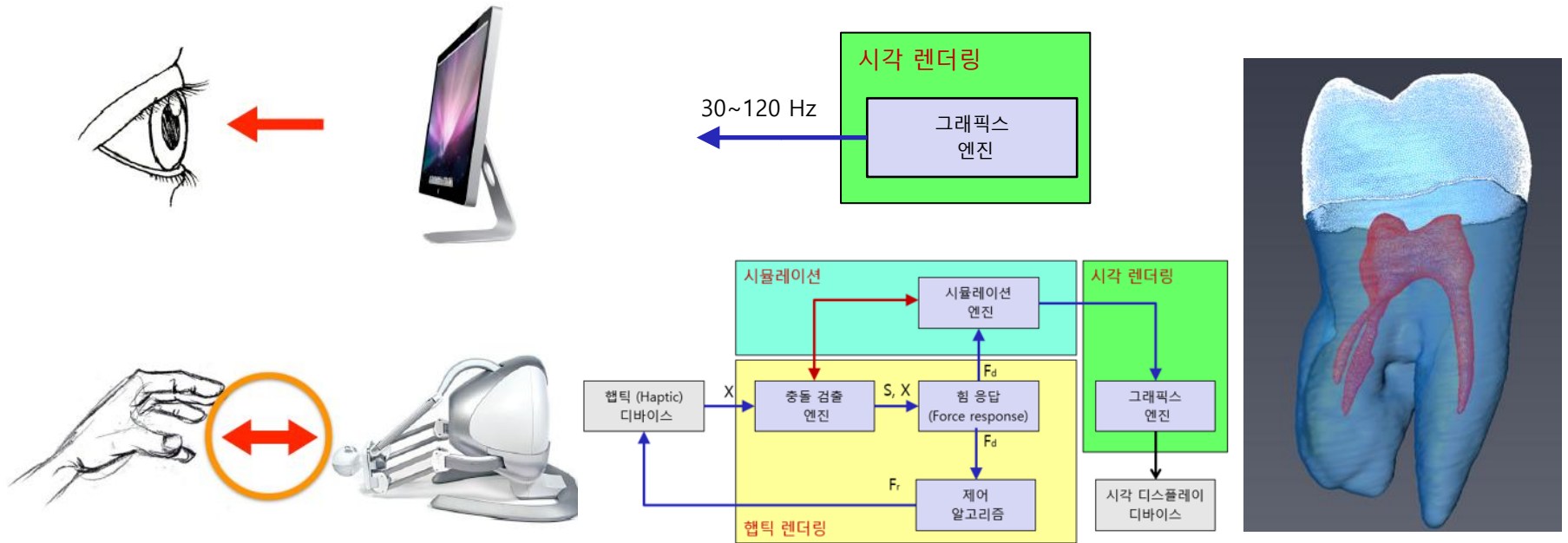
HVR Co., Ltd

www.hvr.co.kr

Haptic@hvr.co.kr

Haptic Rendering

- (점 접촉 모델을 이용한) 사실적인 촉감 생성 및 재현:
 - ✓ 햅틱 기술이 적용된 환경 (실제, 가상 및 원격)에서 가상의 사물이 실제 존재하는 것처럼 느낄 수 있도록, 사람의 촉감 메커니즘과 가상의 사물에 대한 물리적 특성 등을 고려하여, 햅틱 디바이스로 가상의 사물과 상호작용 하는 동안 사실적으로 느낄 수 있는 촉감 발생 과정.
- 햅틱 렌더링은 가상 객체와의 사용자 상호작용에 대한 응답으로 힘을 계산하고 생성하는 프로세스.
- Haptic vs, Visual Rendering:
 - ✓ 양방향 정보 흐름은 햅틱 인터페이스의 가장 두드러진 특징.



Kinesthetic Feedback 햅틱 렌더링을 위한 인터페이스 구조 (1/2)

가상 환경과 물리적인 상호작용을 위해 햅틱 디바이스를 사용:

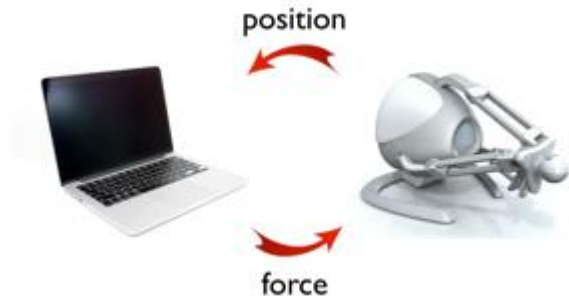
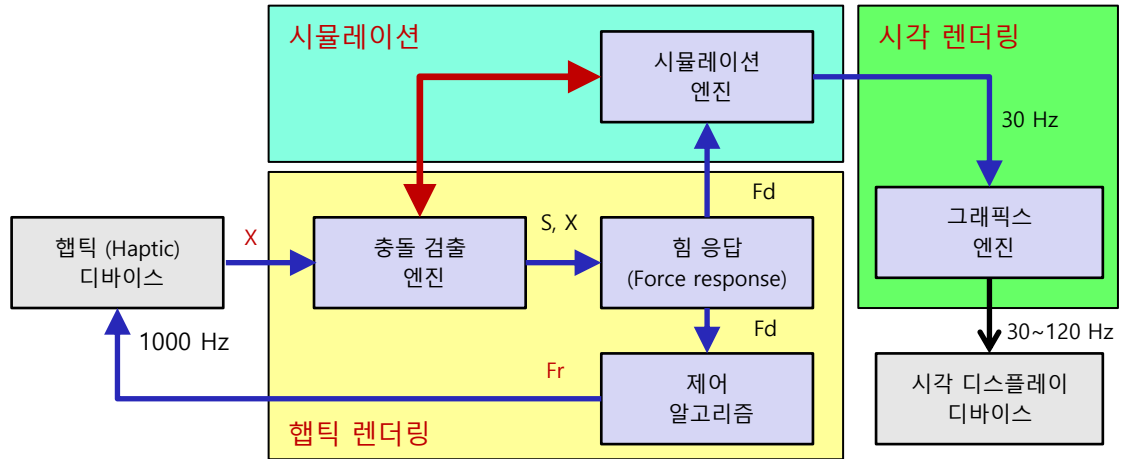
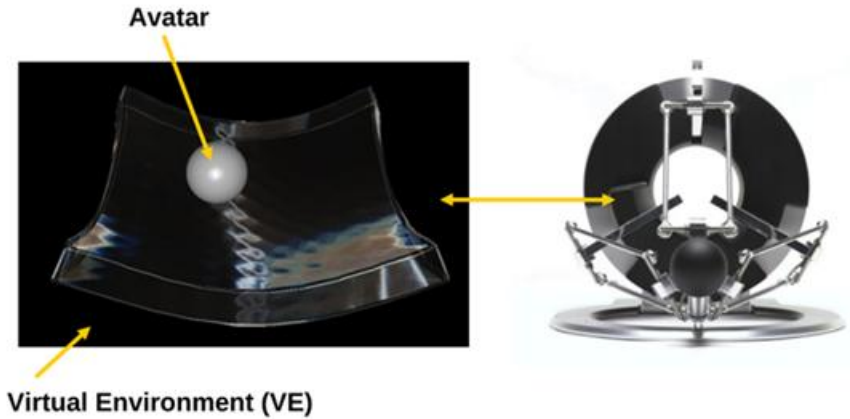
- ✓ End-Effector의 위치를 인코더로 측정.
- ✓ 액추에이터는 사용자에게 힘을 적용.
- ✓ 햅틱 렌더링 알고리즘은 새로운 위치 부여와 같은 힘을 계산.

시뮬레이션 엔진은 가상 환경의 상태를 계산:

- ✓ 가상 객체의 표현(Representations).
- ✓ 물리적 행동의 실 시간 시뮬레이션.
- ✓ 지오메트리 모델링 및 컴퓨터 애니메이션.

그래픽 엔진:

- ✓ 가상 환경(VE)을 화면에 렌더링:
 - ❖ 그래픽에 관한 것만이 아님.
- ✓ OpenHaptics.
- ✓ chai3d:
 - ❖ 햅틱 및 그래픽을 쉬게 처리.
 - ❖ 오픈 소스 멀티 플랫폼의 햅틱 렌더링 프레임워크.



Kinesthetic Feedback 햅틱 렌더링을 위한 인터페이스 구조 (2/2)

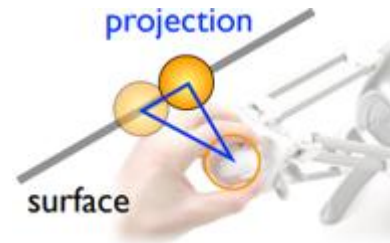
■ 충돌 검출(Collision Detection):

- ✓ 디바이스에 새로운 위치 (x, y, z)가 주어지면 아바타가 충돌되었는지 여부를 모든 객체에 대해 확인.
- ✓ 충돌이 일어난 경우, 어떤 객체의 어느 부분과 충돌되었는지를 검출.



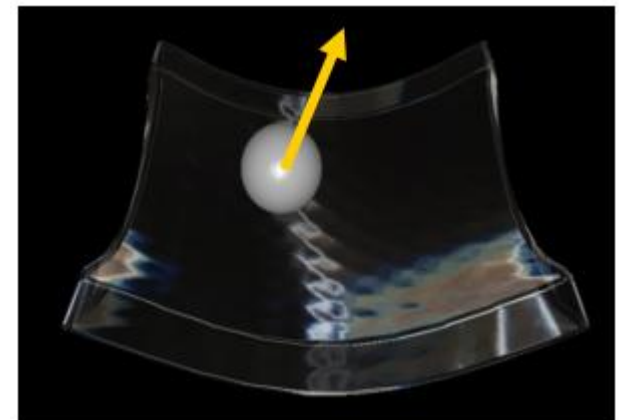
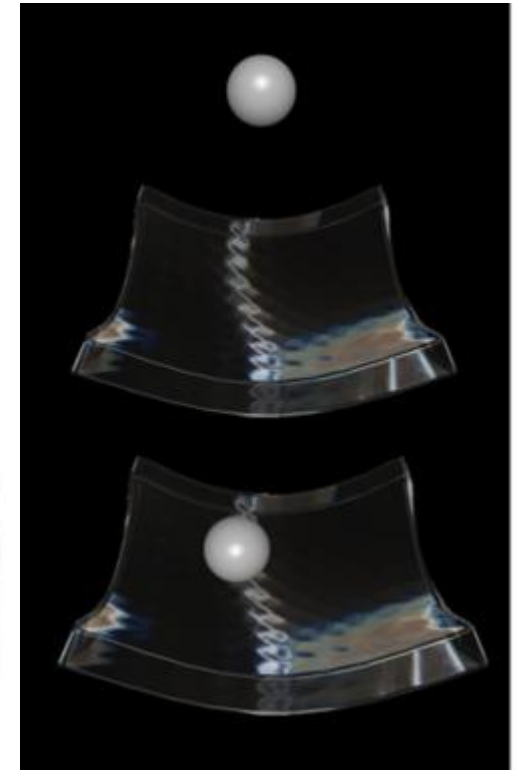
■ 힘 응답(Force Response) 알고리즘:

- ✓ 가상 객체와 충돌 되었을 때, 햅틱 처리기가 가상 객체 내부로 침투하는 것을 방지하기 위해 물리적인 힘 반력을 생성.
- ✓ 가상 객체와 충돌 되었을 때, 다음과 같은 점을 감안하여 힘 반력 (Fx, Fy, Fz) 계산 및 토크 값 계산:
 - ❖ 현실적인 느낌의 접촉.
 - ❖ 현실적으로 보이는 접촉.
- ✓ 반환 값은 일반적으로 디바이스에 적용되는 힘 과 토크 벡터.



■ 제어(Control) 알고리즘:

- ✓ 계산된 힘을 디바이스에 적용:
 - ❖ 디바이스의 안정적인 유지.
 - ❖ 힘은 최대한 계산 결과에 가깝게 적용.
- ✓ 햅틱 렌더링의 이산 시간 (Discrete-time) 특성이 종종 발생.

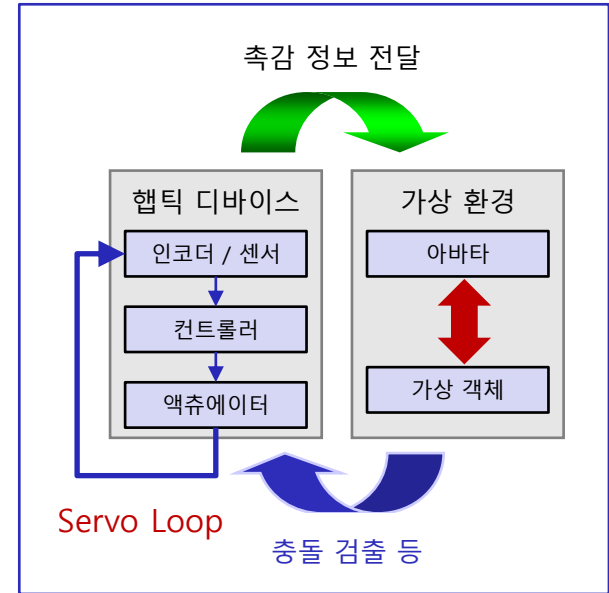


Servo 루프 또는 Servo 쓰레드

- 사람의 사물에 대한 촉감 정보 전달 시간은 보통 0.005초 정도 (200Hz):
 - ✓ 사람의 피부기계수용체(Mechanoreceptors)는 약 400Hz까지의 진동을 감지:
 - ❖ 보다 자연스럽게 정확하게 느끼게 하기 위해 초당 1000번 (1000Hz) 정도의 힘 업데이트 필요.

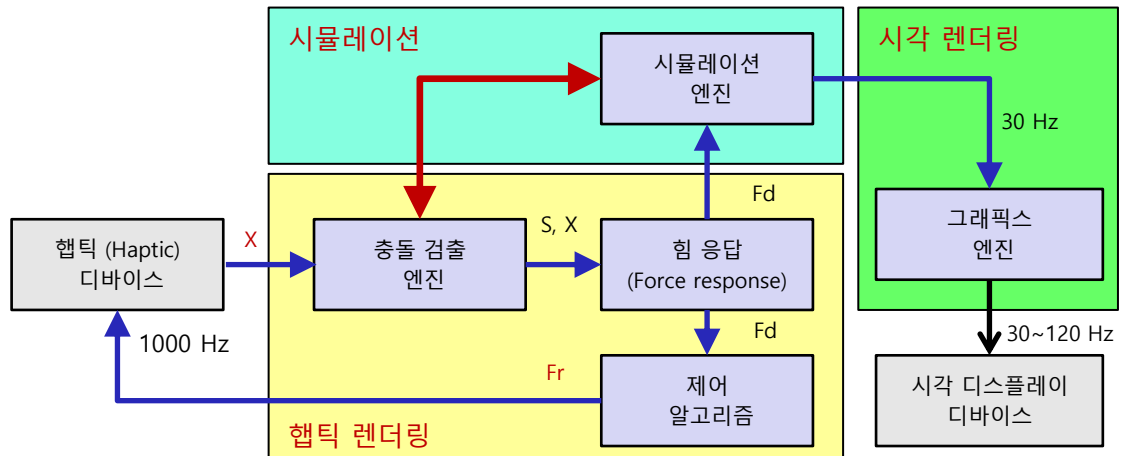
■ Servo 루프 또는 Servo 쓰레드:

- ✓ 햅틱 디바이스로 보낼 힘을 계산하는데 사용하는 높은 주기의 엄격한 제어 루프:
 - ❖ 안정적인 햅틱 피드백을 렌더링 하려면, 이 루프를 일관된 1KHz 속도 이상으로 실행해야 함.
 - ❖ 높은 업데이트 속도를 유지하기 위해
Servo 루프는 일반적으로 별도의 우선 순위가 높은 스레드에서 실행됨.
 - ❖ 멀티 모달 상호작용을 위해 멀티 스레드 사용.



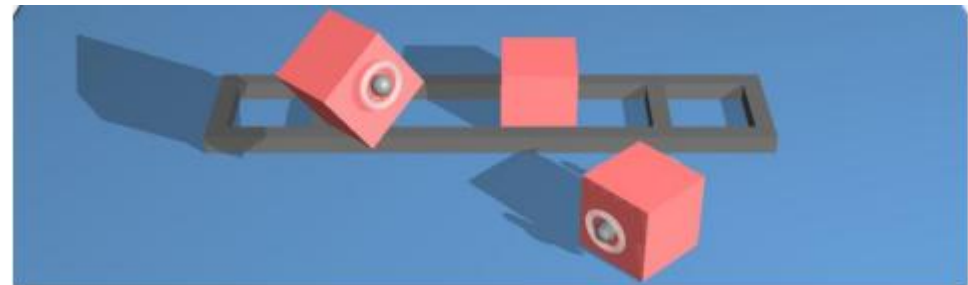
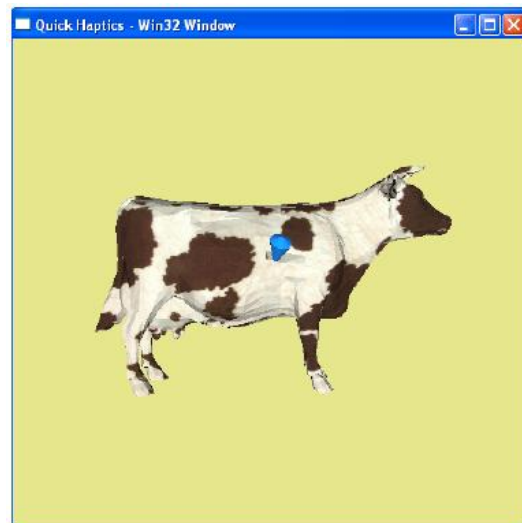
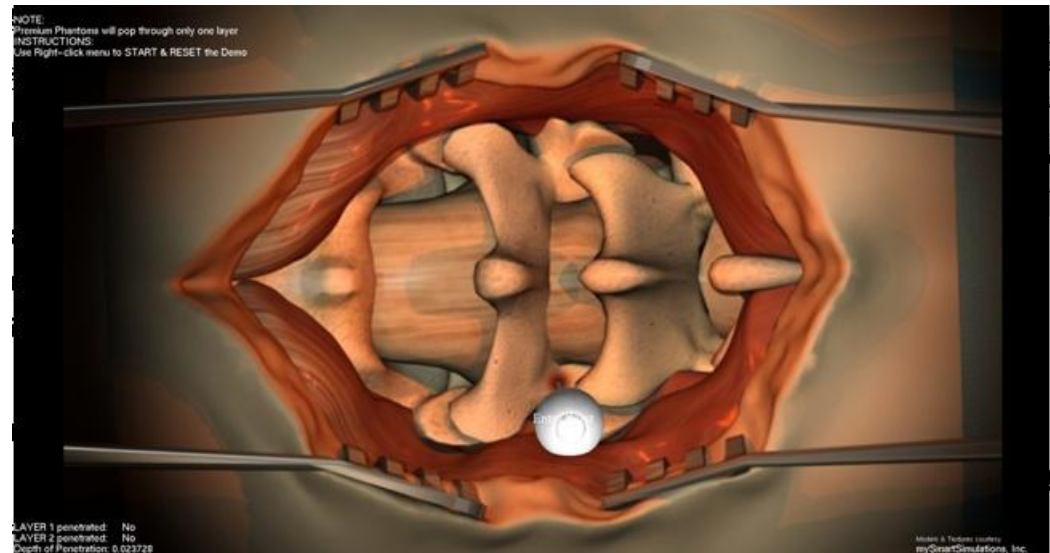
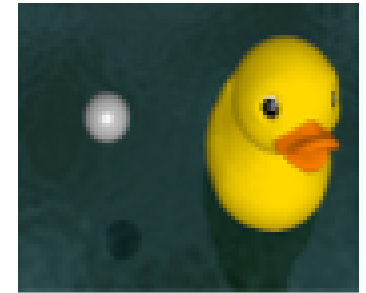
■ 햅틱 렌더링 루프는 보다 빠를수록 좋음:

- ✓ Geomagic 햅틱 장비:
 - ❖ Touch:
 - 500~1600 Hz.
 - ❖ 다른 PHANTOM:
 - 500~1000 또는 2000 Hz.
- ✓ Force Dimension 햅틱 장비:
 - ❖ Omega & Delta:
 - 최대 4 KHz 까지의 재생 빈도 (Refresh rate).
 - ❖ Sigma.7:
 - 최대 4 KHz 까지의 재생 빈도.



충돌 검출 (1/3)

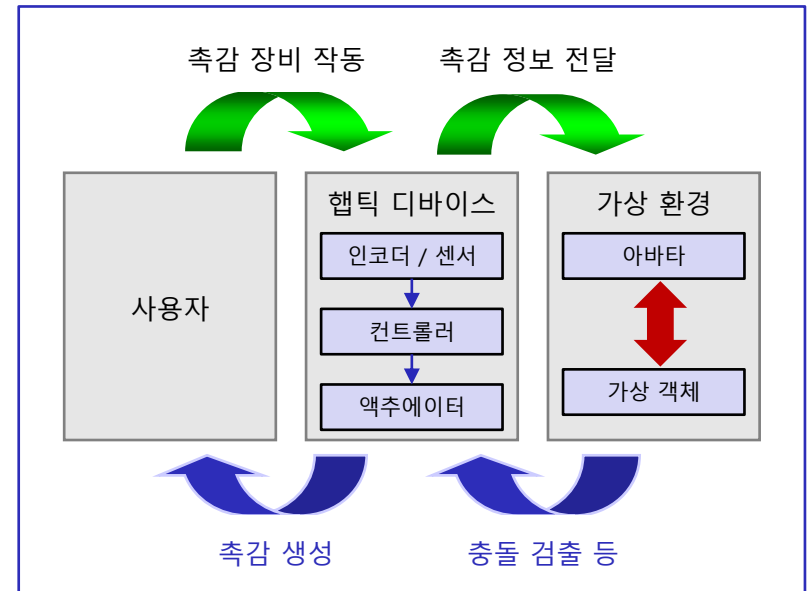
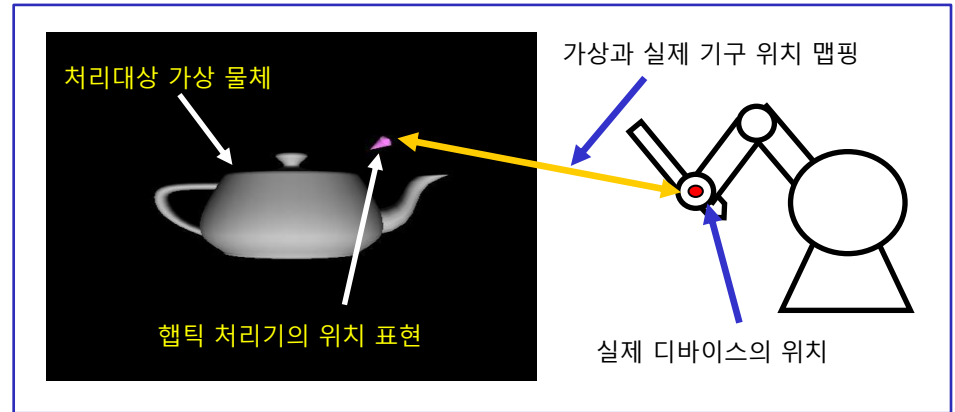
- 충돌 검출 알고리즘은 가상물체의 모양을 표현하는 기하학적 모델에 종속:
 - ✓ 폴리곤(Polygon) 또는 NURBS 기반 모델.
 - ✓ 볼륨(Volume) 데이터 모델.
 - ✓ 유한요소 메소드(Finite Element Method, FEM) 모델.
 - ✓ 유동(Fluid) 모델.
 - ✓ Deformable 모델:
 - ❖ Deform : 촉감을 이용한 연속적인 형태의 변형.
 - ✓ 고급 물리적 모델.
- 햅틱 처리기 모델에 종속:
 - ✓ 단일 포인트(Single Point) 충돌 검출 모델.
 - ✓ 멀티 포인트(Multi Point) 충돌 검출 모델.



각각 메시 객체를 잡는 두 개의 햅틱 포인트로 구성된 두 개의 도구.

충돌 검출 (2/3)

- 아바타(Avatar)는 가상 환경에서 사용자의 물리적 상호작용에 대한 햅틱 인터페이스의 가상 표현:
 - ✓ 그래픽을 사용하는 가상환경에서 가상 처리기(아바타 또는 HIP 또는 SCP 또는 Proxy)의 위치 및 방향 표현을 디바이스의 움직임에 따라 그래픽 좌표 계에 일치시켜 작은 추, 구 또는 도구 모양 등의 형태로 표현하여 사용자의 행동을 대신 처리.
 - ✓ 오퍼레이터는 가상 환경 내에서 아바타의 위치를 제어.
 - ✓ 가상 환경과 아바타 인터페이스 간의 접촉으로 작용과 반작용의 힘을 설정:
 - ❖ 아바타의 지오메트리 및 접촉 유형으로 힘을 조절.
- 시뮬레이션 객체와 물리적 상호작용:
 - ✓ 가상 처리기(아바타)가 가상 물체에 부딪치면 접촉된 방향과 위치, 강도, 속도 등에 따라 햅틱 렌더링을 처리하고, 생성된 촉감 정보를 햅틱 디바이스는 촉감을 느낄 수 있게 사용자에게 전달.



충돌 검출 (3/3)

- 도구 중재 상호작용.



CS277 - Experimental Haptics, Stanford University, Spring 2014

충돌 검출 모델

■ 단일 포인트(Single Point) 충돌 검출 모델:

- ✓ 3 DOF : Position/Translation.
- ✓ Force 렌더링.

■ 멀티 포인트(Multi Point) 충돌 검출 모델:

- ✓ 3DOF + Orientation/Rotation.
- ✓ Force 렌더링 + Torque 렌더링.
- ✓ 일반적으로 6 DOF 디바이스에 유용:
 - ❖ 위치와 방향을 주면 힘과 토크를 출력.
- ✓ 한 개 이상의 충돌 지점을 가지는 가상 물체에 대한 실시간 충돌 검출 시, 물체의 모양 및 모델링의 정밀도 등에 따른 충돌 검출이 각각 다름:
 - ❖ 필요한 계산량은 가상 물체의 모양과 장면(Scene)의 복잡성에 따라서 증가.
 - ❖ 충돌 검출의 효율적이고 최적화된 알고리즘 필요.

6-DOF Interaction



3-DOF
Position/Translation



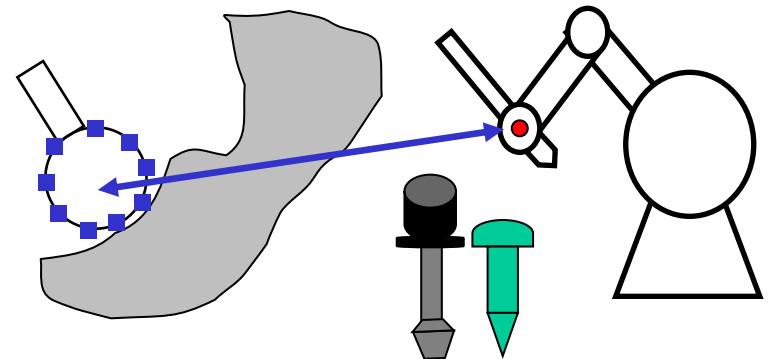
6-DOF
+ Orientation/Rotation



Point-Object



Object-Object

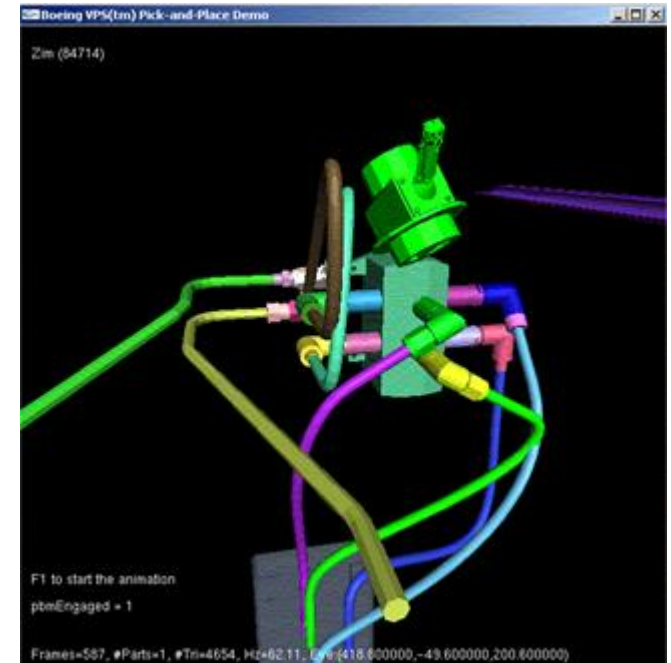


충돌 검출 알고리즘

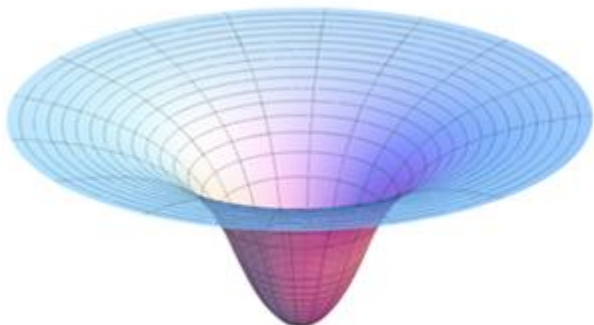
- 충돌 검출 알고리즘의 공통적인 개념:
 - ✓ 효과적인 데이터 구조(바운딩 박스 및 공간 분할).
 - ✓ 전역(Global)과 로컬(Local)간의 효과적인 전이(Transition).
- 근사치(Approximate) 충돌 검출:
 - ✓ 일반적으로 1kHz의 빠른 충돌 검출이 요구되나, 정확한 상호작용 (Interaction) 모델링이 필요 없는 환경에 적용.
 - ✓ 바운딩 박스(Bounding Box) 충돌 검출:
 - ❖ AABB : Axis-Aligned Bounding Box.
 - ❖ OBB : Oriented Bounding Box.
 - ❖ K-Dops : K-Discrete Orientation Polyto-pes



- ✓ H-Collide [Gregory. 1999]:
 - ❖ 햅틱 상호작용을 위해 빠르고 정확한 프레임워크.
 - ❖ 바운딩 박스(Bounding Box) 충돌 검출과 공간 분할이 결합된 알고리즘.
- ✓ 볼륨메트릭(Volumetric) 모델에 Distance Field 적용.
- 정확한 충돌 검출:
 - ✓ 정확한 상호작용(Interaction)과 물리적 모델링이 필요한 환경에 적용하기 위한 알고리즘으로 일반적으로 사용자 정의하여 적용.
 - ✓ SmartCollision™ & 보잉 VPS(Voxmap PointShell Software).



Haptic Rendering

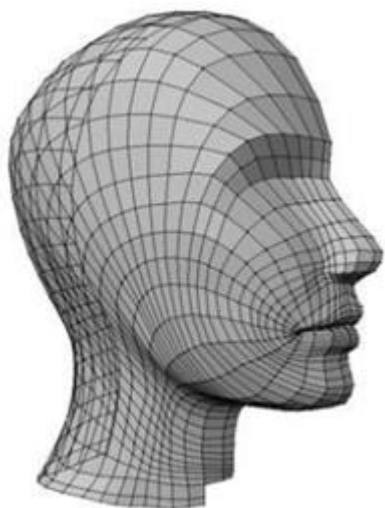


Potential Field

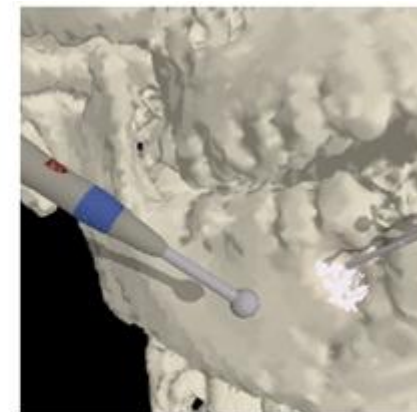
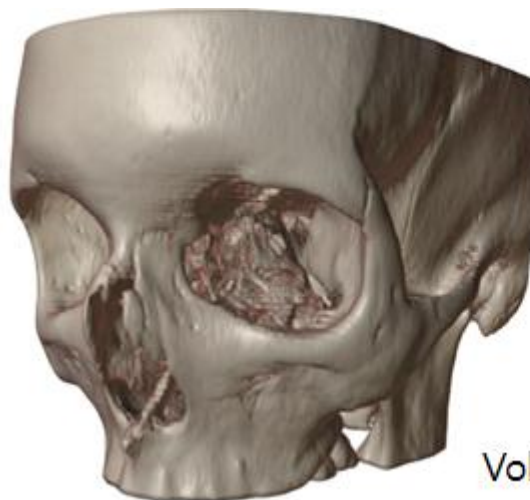


$$S(x,y,z) = (2x^2 + y^2 + z^2 - 1)^3 - (0.1x^2 + y^2)z^3$$

Implicit Surfaces



Polygonal Meshes



Volumetric Data

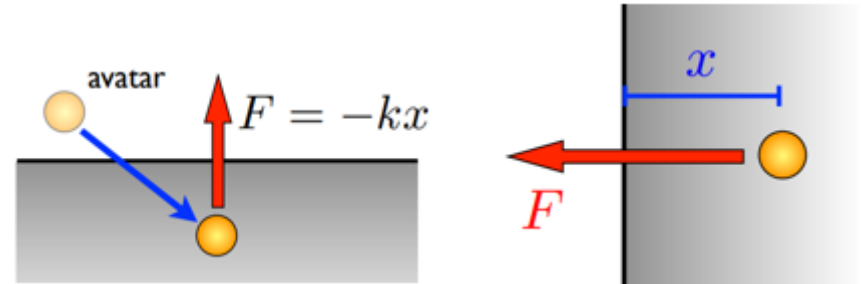
이미지 발취: Prof. Kenneth Salisbury

Potential Fields (1/2)

Virtual Wall:

- ✓ 매우 간단한 가상 환경 (VE) - 3D의 선형 스프링.
- ✓ 안정성 연구에 사용할 수 있고, 보다 복잡한 가상 환경과 상호 작용을 위한 유용한 빌딩 블록.

$$F(x) = \begin{cases} -kx & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}$$

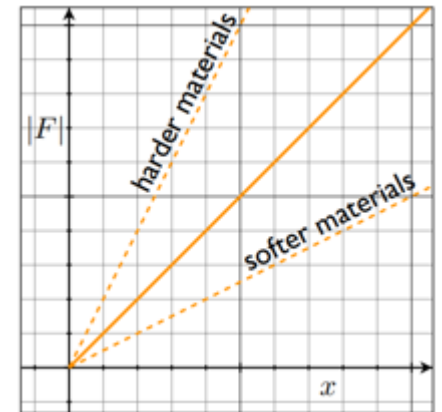
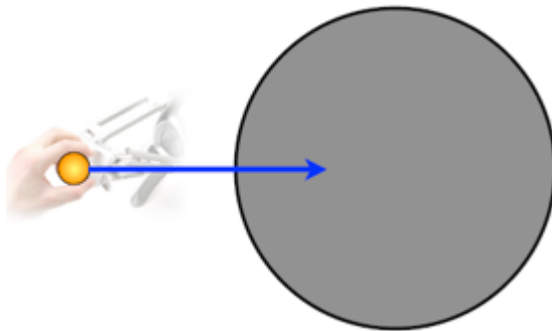


Sphere:

- ✓ 3D로 구체를 렌더링 하는 가장 간단한 방법.

$$F(x, y, z) = -k(x^2 + y^2 + z^2 - r^2)$$

$$F(x, y, z) = \begin{cases} -k(x^2 + y^2 + z^2 - r^2) & \text{if } x^2 + y^2 + z^2 < r^2 \\ 0 & \text{otherwise} \end{cases}$$



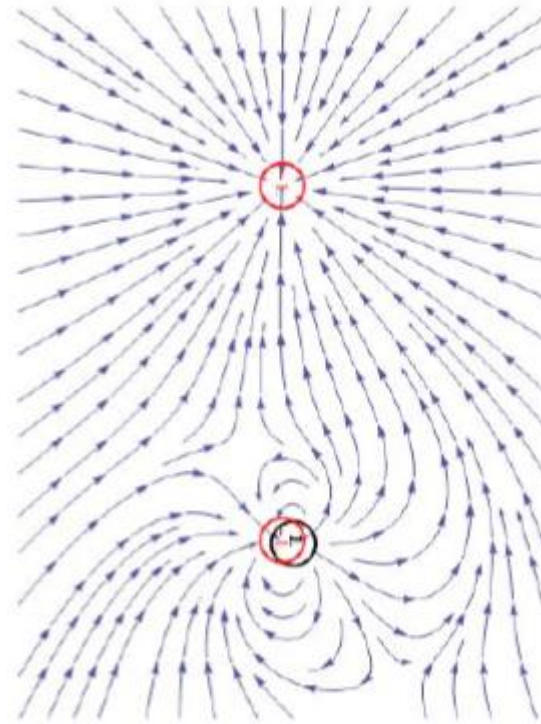
CS277 - Experimental Haptics, Stanford University, Spring 2014

Potential Fields (2/2)

- Potential Field의 용어는 물리학/역학에서 차용.
- 힘은 전위의 그라디언트 벡터 필드:

$$\vec{F} = \nabla U$$

- 직관적인 느낌(3D 스프링).
- 계산하기 쉬움:
 - ✓ 그러나 단순함에는 한계.

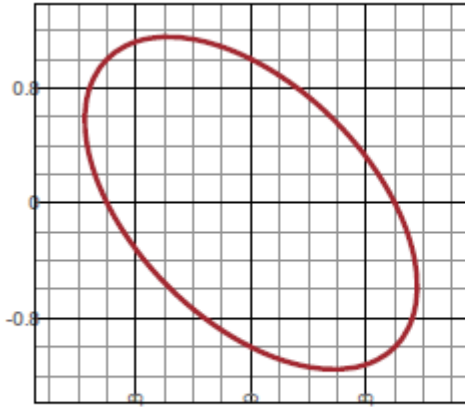


CS277 - Experimental Haptics, Stanford University, Spring 2014

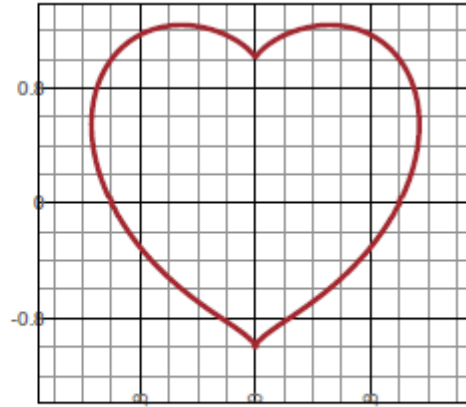
Implicit Surfaces (1/3)

- 수학에서 암시적 표면은 방정식에 의해 정의된 유클리드 공간의 표면:

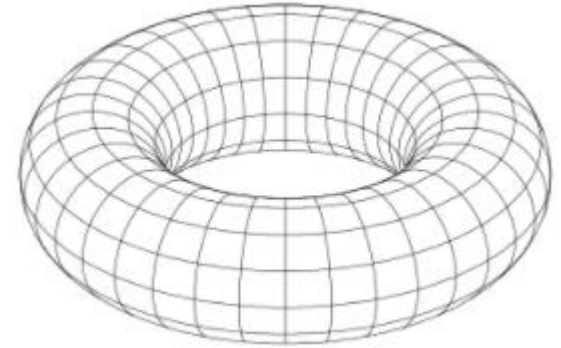
✓ $S(x, y, z) = 0$



$$x^2 + xy + y^2 = 0$$



$$(x^2 + y^2 - 1)^3 - x^2y^3 = 0$$

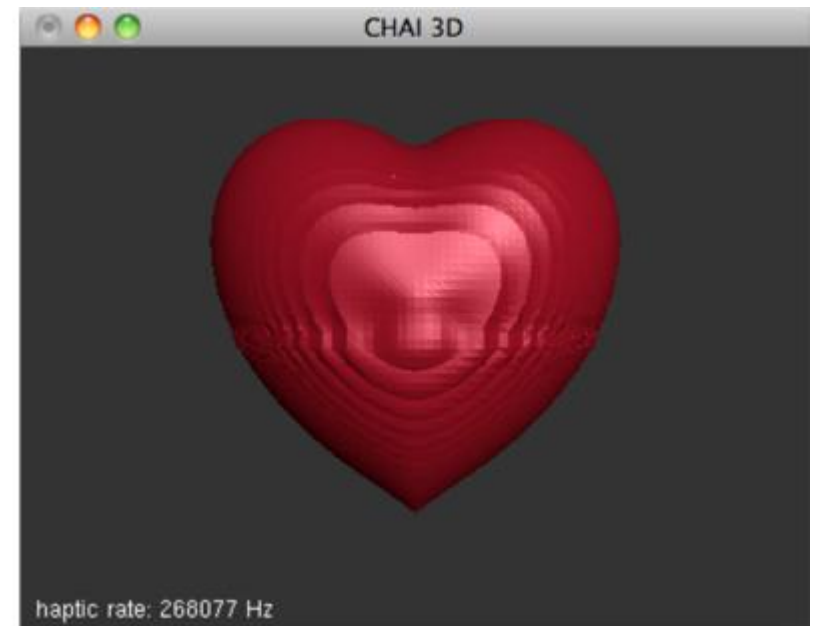
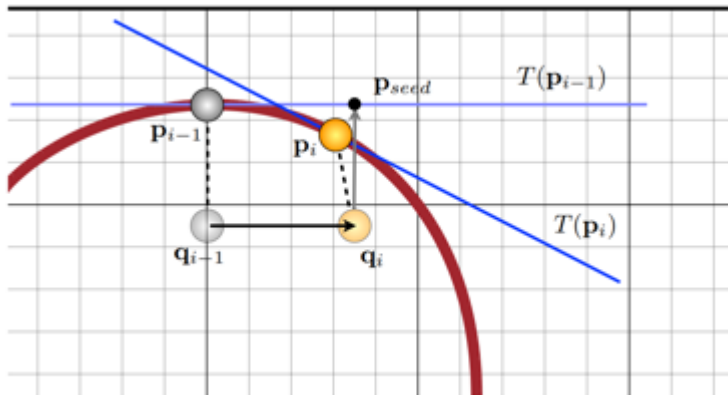
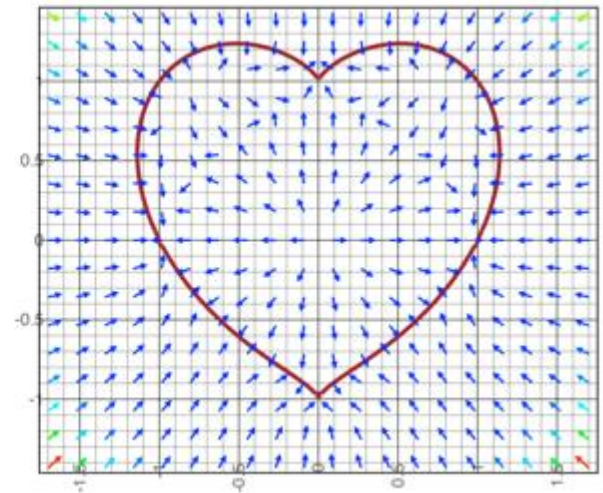
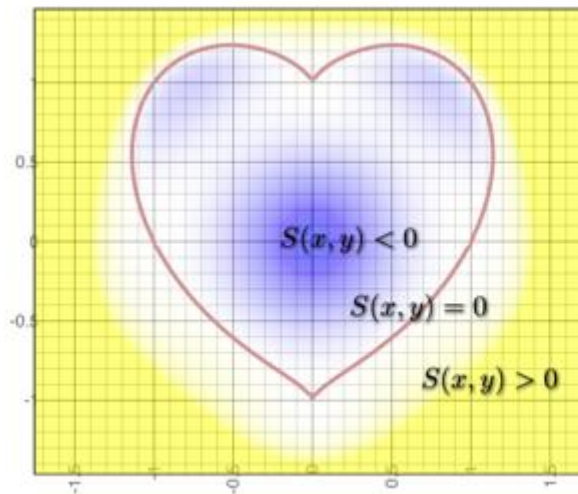


$$S(x,y,z) = (2x^2 + y^2 + z^2 - 1)^3 - (0.1x^2 + y^2)z^3$$

Implicit Surfaces

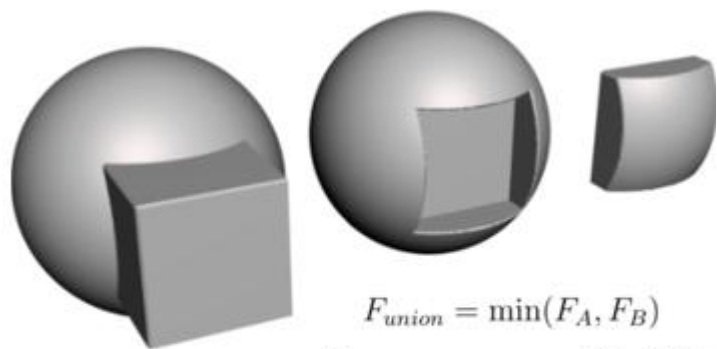
Implicit Surfaces (2/3)

- Proxy 기반 알고리즘으로 렌더링 가능 (1/2).



Implicit Surfaces (3/3)

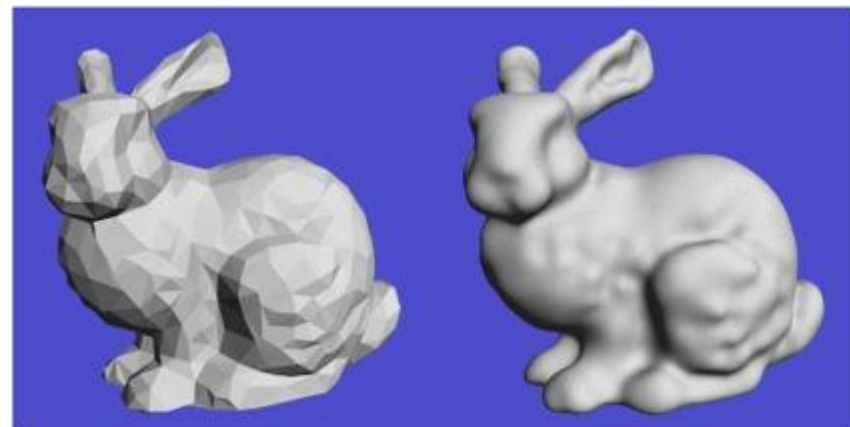
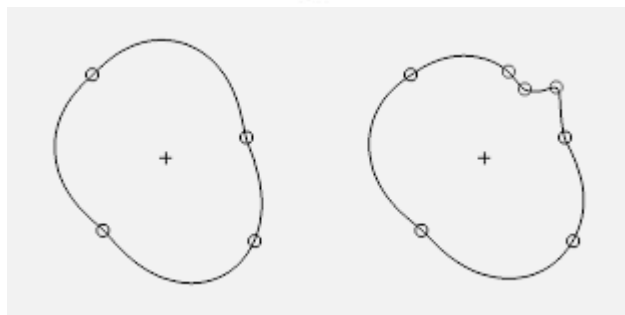
- Proxy 기반 알고리즘으로 렌더링 가능 (2/2).



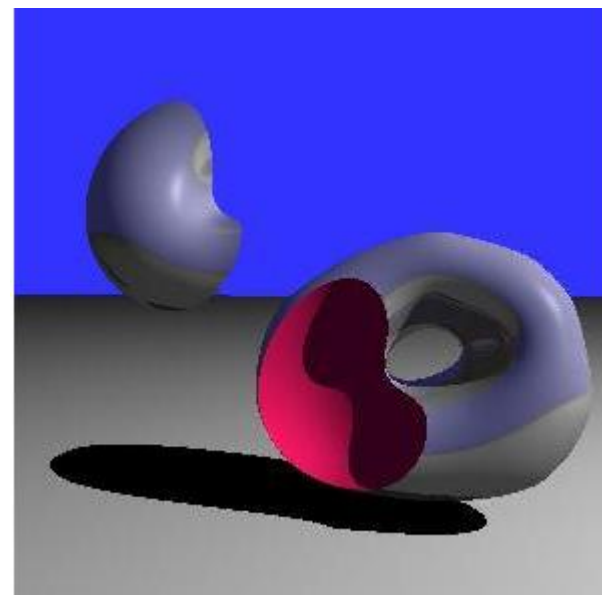
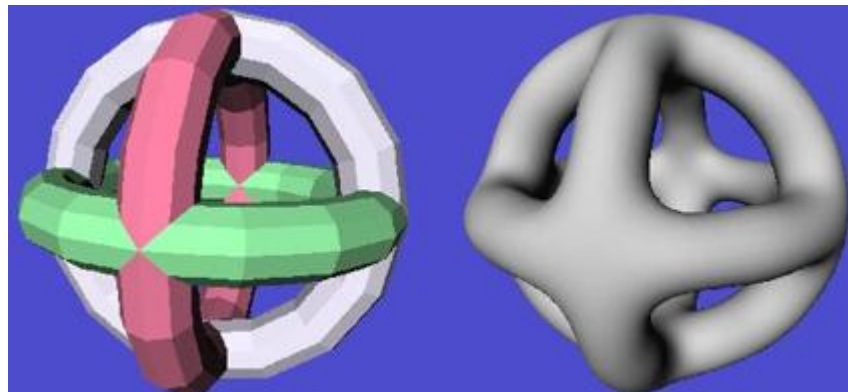
$$F_{union} = \min(F_A, F_B)$$

$$F_{intersection} = \max(F_A, F_B)$$

$$F_{difference} = \max(F_A, -F_B)$$



Polygonal mesh (left) and resulting implicit bunny (right).



Force Model

Force Model의 타입:

- ✓ 포인트 기반 : 단일 포인트의 충돌, 힘.
- ✓ Ray 기반(Line Segment) : 단일 포인트의 충돌, 힘 + 토크.
- ✓ 3D 객체 기반 : 멀티 포인트의 충돌, 힘 + 토크.

Stiffness 렌더링:

- ✓ Potential Fields.
- ✓ Implicit Surfaces.
- ✓ 기하학적(Geometric) 가상 물체:
 - ❖ 제약(Constraint) 기반 법:
 - Spring-Damper 모델, Proxy Contact Point.
 - ❖ Penalty 기반 법:
 - Penetration Depth.
- ✓ 볼륨메트릭(Volumetric) 가상 물체:
 - ❖ 힘의 방향(Direction) & 힘의 양(Amount).

Friction 렌더링:

- ✓ 도구의 움직임에 반대로 저항하는 힘에 측면 힘 추가.

Haptic Texture 렌더링:

- ✓ 저항하는 힘에 작은 교란(Perturbation) 추가.
- ✓ 텍스처 모델 : 위치 모델 Vs. 힘 모델.



Point - Object Interaction
(3DOF Output)



Object - Object Interaction
(6DOF Output)



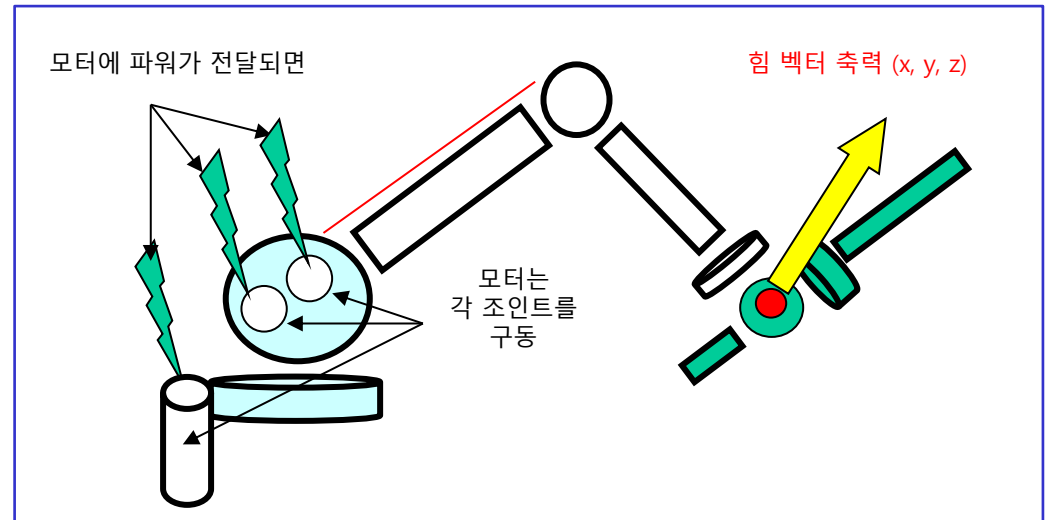
햅틱 디바이스의 힘(Force) 렌더링

- 일반적으로 힘(Force Feedback)은 움직임(Motion)에 저항 하거나 보조(Assist motion) 하는데 사용됨.
- 햅틱 디바이스가 표시하는 힘을 계산하는 방법에는 여러가지가 있음:
 - ✓ 가장 흥미로운 힘에 대한 상호작용 중 일부는 디바이스 End-Effector(사용자가 손에 가지고있는 장치의 기구학적 체인의 끝)의 위치와 가상 환경에서 객체와의 관계를 고려하여 햅틱 디바이스에 의해 표현되는 힘 계산을 수행:
 - ❖ 0의 힘이 렌더링 될 때:
 - 디바이스의 End-Effector의 움직임은 상대적으로 자유롭고 무게가 없어야 함.
 - ❖ 사용자가 디바이스의 End-Effector를 가상 환경에서 움직일 때 햅틱 렌더링 루프 명령은 End-Effect가 표면을 침투하는 저항하는 매우 높은 속도(일반적으로 1000Hz)로 힘을 생성:
 - 이를 통해 사용자는 가상환경에서 객체의 형상을 효율적으로 느낄 수 있음.
- 힘이 계산되는 방식은 다양한 효과를 생성하기 위해 달라질 수 있음:
 - ✓ 예를 들어 힘은 물체 표면을 딱딱하거나 부드럽게, 거칠거나 매끄럽게, 또는 끈적끈적한 느낌 등을 적용할 수 있음.
 - ✓ 또한 햅틱 렌더링에 의해 생성된 힘은 주변 효과를 생성하는데 사용될 수 있음:
 - ❖ 예를 들어, 관성 및 점도는 환경에서 사용자의 자유 공간 움직임을 수정하는 일반적인 방법.
 - ✓ 가상 환경에서 힘의 또 다른 일반적인 용도는 사용자가 객체를 선택하거나 조작을 수행하는 동안 사용자의 움직임을 제한하여 안내(Guidance)를 제공하는 것.



Force Rendering

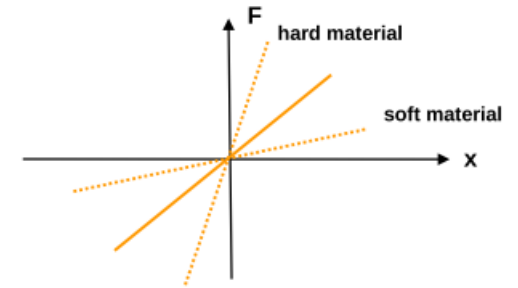
- 일반적으로 힘(Force Feedback)은 움직임(Motion)에 저항(Resist) 하거나 보조(Assist motion) 하는데 사용.
- 힘 벡터(Force vector)는 햅틱 디바이스의 출력 단위.
- 다양한 감각을 생성하기 위해 힘을 계산하는 방법에는 여러 가지가 있음:
 - ✓ 시뮬레이션 할 수 있는 세 가지 주요 힘 클래스는 동적 종속, 시간 종속 또는 둘의 조합.
- 동적 종속(Motion Dependent):
 - ✓ 동적 종속 힘은 햅틱 디바이스의 움직임을 기반으로 계산됨:
 - ❖ 스프링 : Spring.
 - ❖ 댐퍼 : Damper.
 - ❖ 마찰 : Friction.
 - ❖ 관성 : Inertia.
- 시간 종속(Time Dependency):
 - ✓ 시간 함수를 기반으로 계산:
 - ❖ 상수 : Constant.
 - ❖ 주기 : Periodic.
 - ❖ 충격 : Impulses.
- 동적 종속과 시간 종속의 조합.



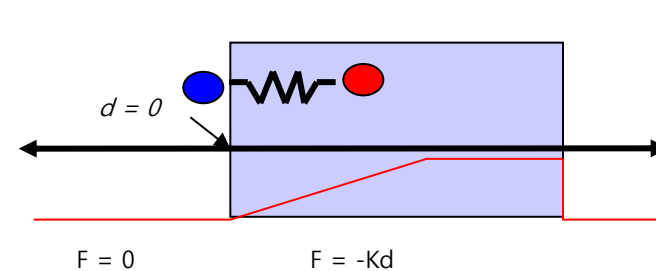
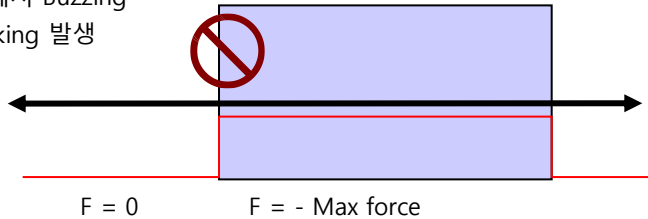
Motion Dependent (1/6)

■ 스프링 모델 (Springs/Stiffness):

- ✓ 매우 다양하고 사용하기 간편하기 때문에 햅틱 렌더링에 사용되는 가장 일반적인 힘 계산.
- ✓ Hooke의 법칙을 적용하여 선형 스프링 힘 계산:
 - ❖ 스프링은 고정 앵커 위치 P_0 와 디바이스 위치 P_1 사이에 부착.
 - ❖ 고정 앵커 위치는 일반적으로 사용자가 터치하는 물체의 표면에 배치됨.
 - ❖ 변위 벡터는 스프링 힘이 항상 고정된 앵커의 위치를 향하도록 함.
- ✓ 강성 상수(Stiffness, k)는 스프링이 자신을 정지(Rest) 길이로 회복하려고 하는 정도를 나타냄:
 - ❖ 낮은 강성 상수는 느슨함을 느끼는 반면, 높은 강성 상수는 딱딱함을 느낌.



경계 면에서 Buzzing
혹은 Kicking 발생



사용자가 힘을 가하면

P_0 , 이상적인 가상 디바이스의 위치(완전히 딱딱한 물체의 표면인 경우).

스프링 변위 거리 $d = P_0 - P_1$.

P_1 , 현재(Current) 가상 디바이스의 위치.

F

스프링 모델의 시뮬레이션 결과:

$F = -Kd$ (Hooke의 법칙).

스프링 또는 Stiffness 상수 K 는 표면 재질 함수.

d 는 변위(Displacement) 벡터.

Motion Dependent (2/6)

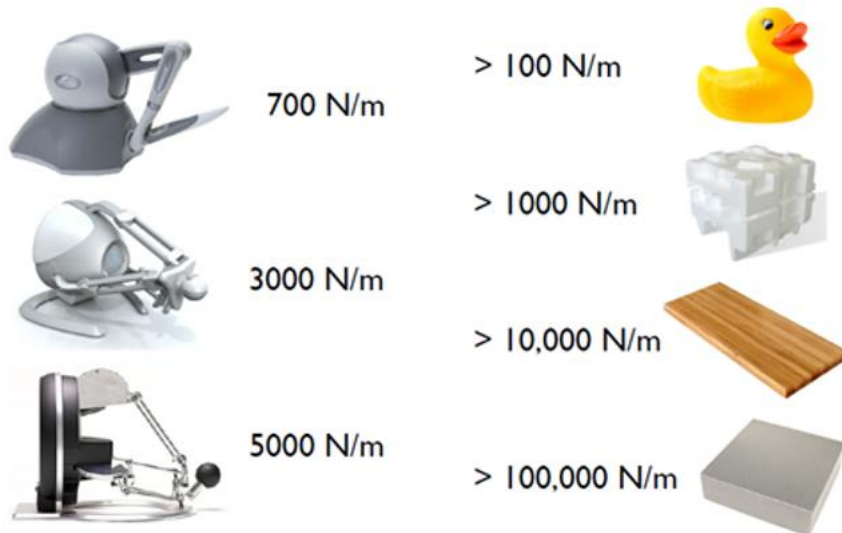
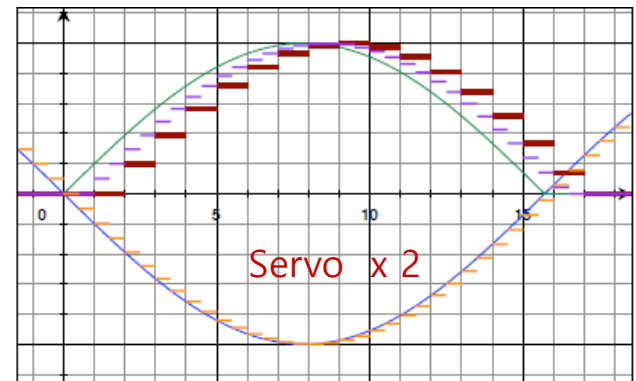
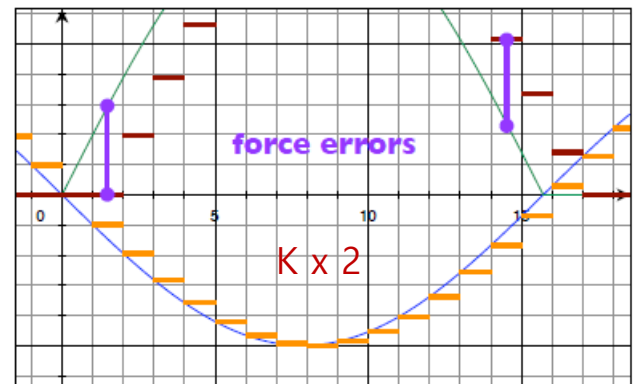
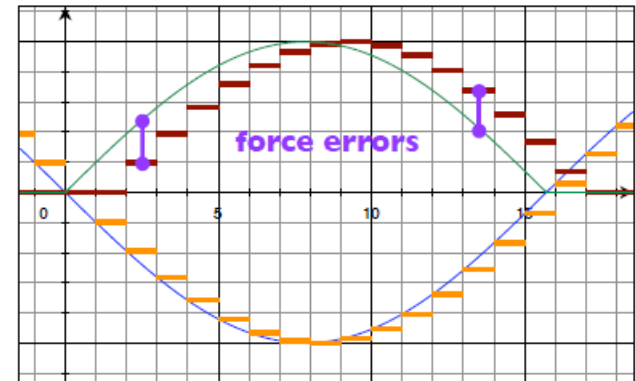
■ 스프링 모델의 이슈:

- ✓ 샘플링 데이터 시스템:
 - ❖ 이산(Discrete) 힘 신호를 발생.
- ✓ Stiffness 효과:
 - ❖ 강성이 증가함에 따라 오류가 증가.
- ✓ Servo Rate 효과:
 - ❖ 서보 루프 속도가 증가하면 오류가 감소.

■ 몇 가지의 쟁점:

- ✓ 높은 강성과 낮은 서보 속도로 인해 과도한 에너지가 생성.
- ✓ 기계 장치를 통해 에너지를 방출.
- ✓ 고급 제어 알고리즘, 딱딱함 시뮬레이션, 사람의 감각 대역폭 등

x position
 F ideal force
 \underline{x} sampled position
 \underline{F} commanded force



CS277 - Experimental Haptics, Stanford University, Spring 2014

Motion Dependent (3/6)

■ 스프링 모델의 불안정:

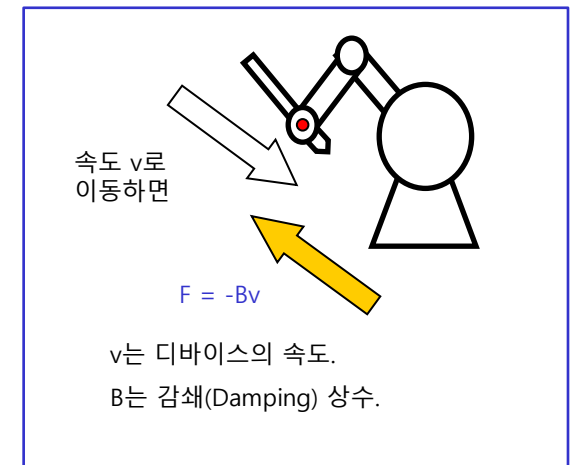
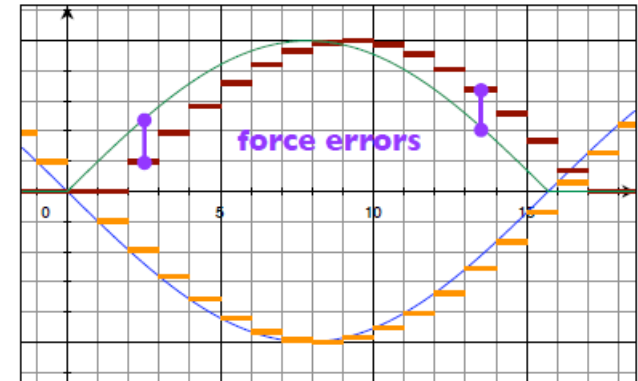
- ✓ 사용자는 연속적인 시간(Continuous time, Analog)으로 힘을 적용하나, 가상 스프링은 이산 시간(Discrete time, Digital)으로 힘을 계산:
 - ❖ 현재의 샘플링 된 힘 계산 지점은 실제 디바이스의 위치와 시간적 차이 발생:
 - 가상 스프링에 의해 여분(Extra) 힘 발생.

■ 댐핑/점성(Damping/Viscosity):

- ✓ 댐핑은 생성된 힘(에너지)이 시간 또는 거리에 따라 점점 감소하는 것:
 - ❖ 일반적으로 댐퍼의 강도는 End-Effector 속도에 비례.
- ✓ 힘은 항상 움직임의 반대 방향으로 생성:
 - ❖ End-Effector를 속도 v 로 이동 하면
힘은 속도 v 의 이동 방향에 반대 방향으로 생성.
 - ❖ 가상 스프링에 의한 여분의 힘 제거에 사용.
- ✓ 점성은 형태가 변화할 때 나타나는 유체의 저항, 서로 붙어 있는 부분이 떨어지지 않으려는 성질.

■ 관성(Inertia):

- ✓ 관성(Inertia)은 질량 이동과 관련된 힘:
 - ❖ 예를 들어, 주어진 궤적을 알고 있다면 (예를 들어, 운동 방정식의 솔루션), 뉴턴의 법칙 ($F=ma$)을 사용하여 해당 운동 중에 느끼는 힘을 쉽게 계산할 수 있음.



Motion Dependent (4/6)

- 스프링-댐퍼(Spring-damper) 모델:

스프링 댐퍼 모델의 시뮬레이션 결과:

$$F = - (Kd - Bv)$$

K : Stiffness, B : 댐핑/점성 (Viscosity/Damping), v : 속도.
d : 변위(Displacement) 벡터.

Damper(가상으로 연결-진동 제거)

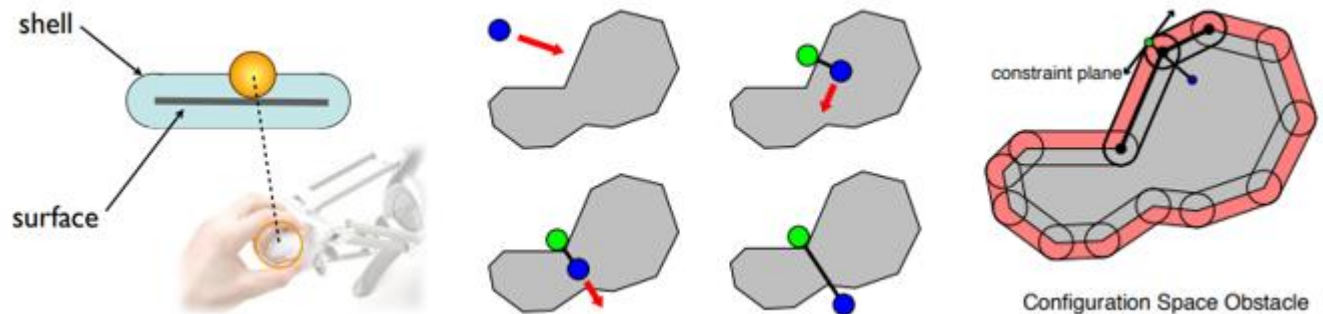
- Virtual Proxy 모델:

가상 Proxy의 시뮬레이션 결과:

$$F = - [Kd(t) - Bd'(t)] n(t)$$

- * K : 딱딱함(Stiffness).
- * B : 댐핑/점성 (Viscosity/Damping).
- * d(t) : 침투(Penetration) 깊이.
- * n(t) : Surface normal

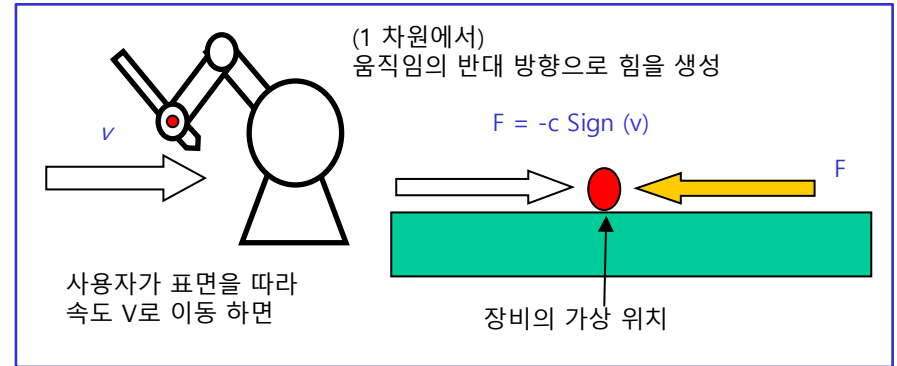
- Finger-Proxy 알고리즘:



Motion Dependent (5/6)

■ 쿨롱 마찰(Coulombic Friction):

- ✓ 가장 기본적인 형태는 일정한 크기의 마찰 상수 c 에 대한 마찰력으로 운동의 방향에 반대하는 쿨롱 마찰.
- ✓ 1 차원에서 쿨롱 마찰력은 방정식 $F = -c \operatorname{sgn}(v)$:
 - ❖ v 는 엔드이펙터의 속도이고 c 는 마찰 상수.
- ✓ 일반적으로 큰 댐핑 상수와 일정한 힘 클램프(범위+스케일)가 작은 댐핑 식을 사용하여 구현됨.
- ✓ 쿨롱 마찰은 느린 움직임의 경우 마찰이 속도에 비례하기 때문에, 방향을 바꿀 때 부드러운 전환을 만드는데 도움이 됨.



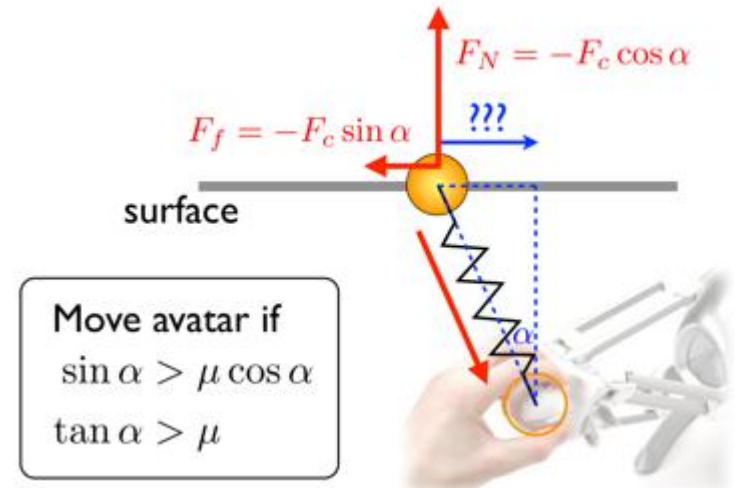
■ 점성 마찰(Viscous Friction):

- ✓ 마찰이 댐핑 식과 클램프를 사용하여 계산된다는 점에서 쿨롱 마찰과 유사:
 - ❖ (차이점은) 댐핑 상수가 낮고 클램프 값이 높은 경향이 있다는 것.

◆ 쿨롱 마찰(Coulomb Friction) (1/2):

- 수직 력 (Normal Force)에 비례하는 마찰 력 (Friction Force).

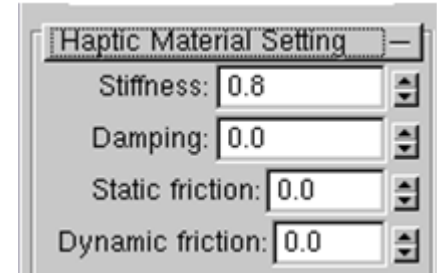
$$F_f = \mu F_N$$



Motion Dependent (6/6)

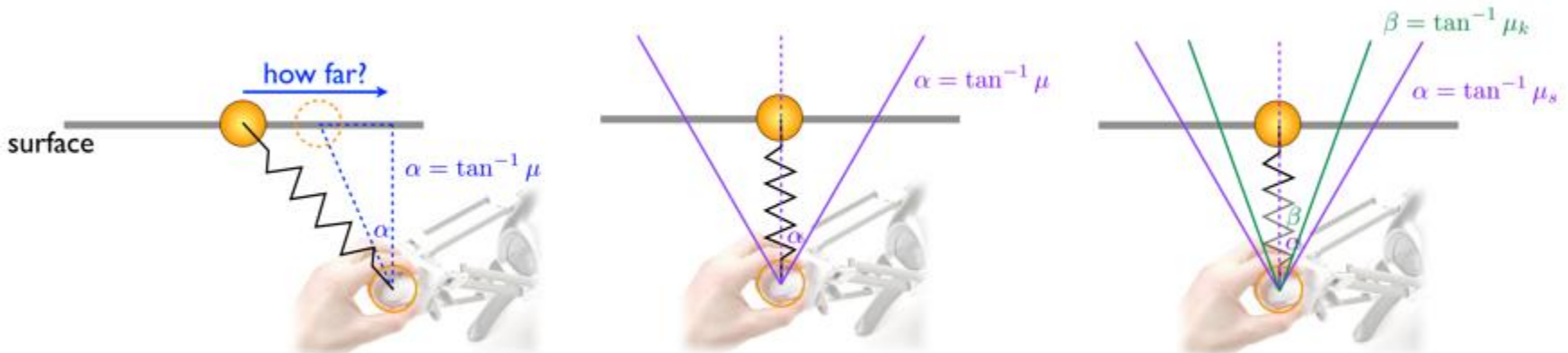
■ 정적/동적 마찰(Static/Dynamic Friction):

- ✓ 정적 마찰은 정지상태에서 표면을 따라 처음 움직임이 시작할 때 발생하는 마찰.
- ✓ 동적 마찰은 표면을 따라 이동 할 때 발생하는 마찰.
- ✓ 일반적으로 Stick-slip 마찰로 언급:
 - ❖ 두 개의 객체가 서로 붙어있는 상태에서 미끄러질 때의 급격한 동작의 변화를 구현.
 - ❖ 마찰 모델은 상대의 움직임이 없을 때와 상대 움직임에 저항할 때 사이를 전환.
- ✓ 마찰력은 항상 표면을 따라 측면 움직임과 반대되며, 마찰력의 크기는 항상 수직(Normal) 접촉력에 비례함.



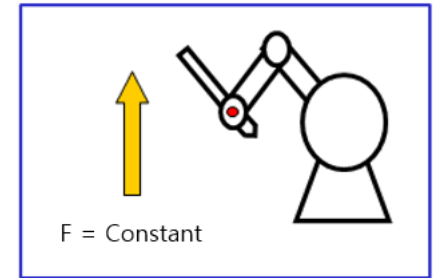
◆ 쿨롱 마찰(Coulomb Friction) (2/2):

- 정적(Sticking) 마찰. $F_s \leq \mu_s F_N$
- 운동(Sliding) 마찰. $F_f = \mu F_N$



Time Dependency

- 시간 종속적인 힘은 시간의 함수로 계산됨을 의미.
- 일정한 힘(Constant Forces):
 - ✓ 일정한 힘은 크기와 방향이 고정된 힘.
 - ✓ 일반적으로 스타일러스(End-Effector)를 무중력 상태에서 움직이기는 것처럼 느끼게 하는 것과 같은 중력 보상에 사용됨:
 - ❖ 반대로, 스타일러스(End-Effector)가 정상보다 무겁게 느껴지도록 하는데도 사용 할 수 있음.
- 주기적인 힘(Periodic Forces):
 - ✓ 주기적인 힘은 시간이 지남에 따라 반복되는 패턴을 적용 할 때 발생 함:
 - ❖ 패턴에는 톱니 파, 사각 파 또는 정현 파 등
 - ✓ 주기적인 힘은 패턴 주기의 주기를 제어하는 시간 상수와 주기의 정점에서 힘이 얼마나 강한 지 결정하는 진폭으로 설명됨:
 - ❖ 또한, 주기적인 힘에는 방향이 필요함.
 - ❖ 진폭은 디바이스가 출력할 수 있는 최대 힘을 초과하지 않아야 됨:
 - 그렇지 않으면 파형의 모양이 잘림.
 - ❖ 또한, 파형의 주파수는 디바이스의 Servo 루프 속도에 의해 제한됨:
 - 예를 들면, 진동 주파수의 이론상 최대는 디바이스의 Servo 루프 속도의 절반.
- 임펄스(Impulses):
 - ✓ 임펄스는 즉시 적용되는 힘 벡터:
 - ❖ 실제로 햅틱 장치를 사용한 임펄스는 짧은 시간 동안 적용하는 것이 가장 좋음.
 - ❖ 또한 충격 반동에 사용되는 것과 같은 믿을 수 있는 충격을 얻으려면 일시적인 힘이 가능한 한 날카로워 야함.
 - ✓ 인간의 감각은 정상 상태의 힘 (큰 진폭의 작은 힘) 보다 힘의 불 연속적인 힘 (작은 진폭의 큰 힘)에 보다 민감 함.
 - ❖ 너무 높은 힘을 렌더링 하려고 하면 디바이스의 물리적 한계로 인해 효과가 없을 수 있음.



OpenHaptics Overview

■ HDAPI(Haptic Device API):

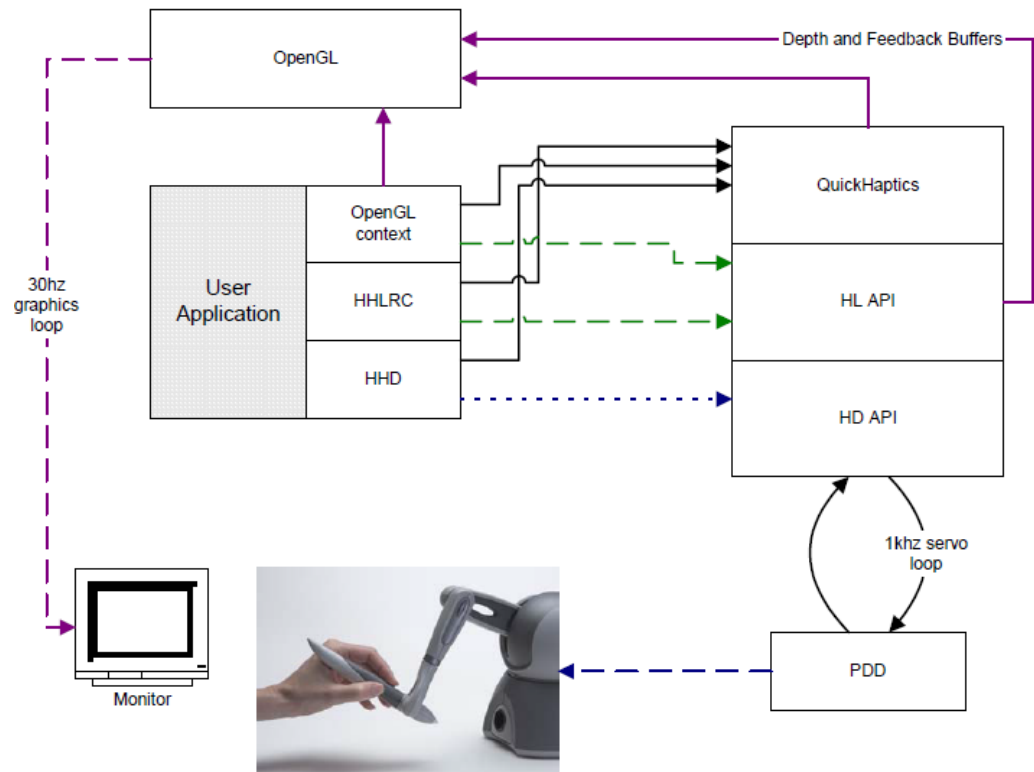
- ✓ 햅틱 디바이스에 대한 낮은 수준의 액세스를 제공.
- ✓ 햅틱 프로그래머가 힘(Force)을 직접 햅틱 렌더링(Direct Haptic Rendering) 할 수 있음.
- ✓ 디바이스의 런타임 동작 구성에 대한 제어를 제공하고, 편리한 유틸리티 기능과 디버깅 지원을 제공.

■ HLAPI(Haptic Library API):

- ✓ HDAPI 보다 높은 수준의 햅틱 렌더링을 제공.
- ✓ OpenGL API 프로그래머에게 친숙한 구조로 설계.
- ✓ 기존 OpenGL 프로그램을 재 사용 할 수 있으며 햅틱 및 그래픽 스레드 간의 동기화를 크게 단순화 함.



■ QuickHaptics microAPI:



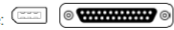
- ✓ C++를 사용하는 API로 OpenHaptics 3.0에서 추가.
- ✓ 햅틱에 비 전문가 일지라도 극적으로 빠르고 쉽게 새로운 햅틱 응용프로그램을 작성하거나 기존 응용 프로그램에 햅틱을 추가 할 수 있는 microAPI.
- ✓ 전형적인 햅틱 응용 프로그램에 대한 논리적(Logical) 단계를 캡슐화(Encapsulate) 하는 그래픽 및 햅틱 함수 호출이 포함되어 있어, 신속한 프로그램 설계 및 배포가 가능 함.
- ✓ 내장된 지오메트리 파서(Parser) 및 지능형 기본 매개변수를 사용하면 최소한의 코드로 햅틱/그래픽 장면(Scene)을 설정할 수 있음.



QuickHaptics (1/2)

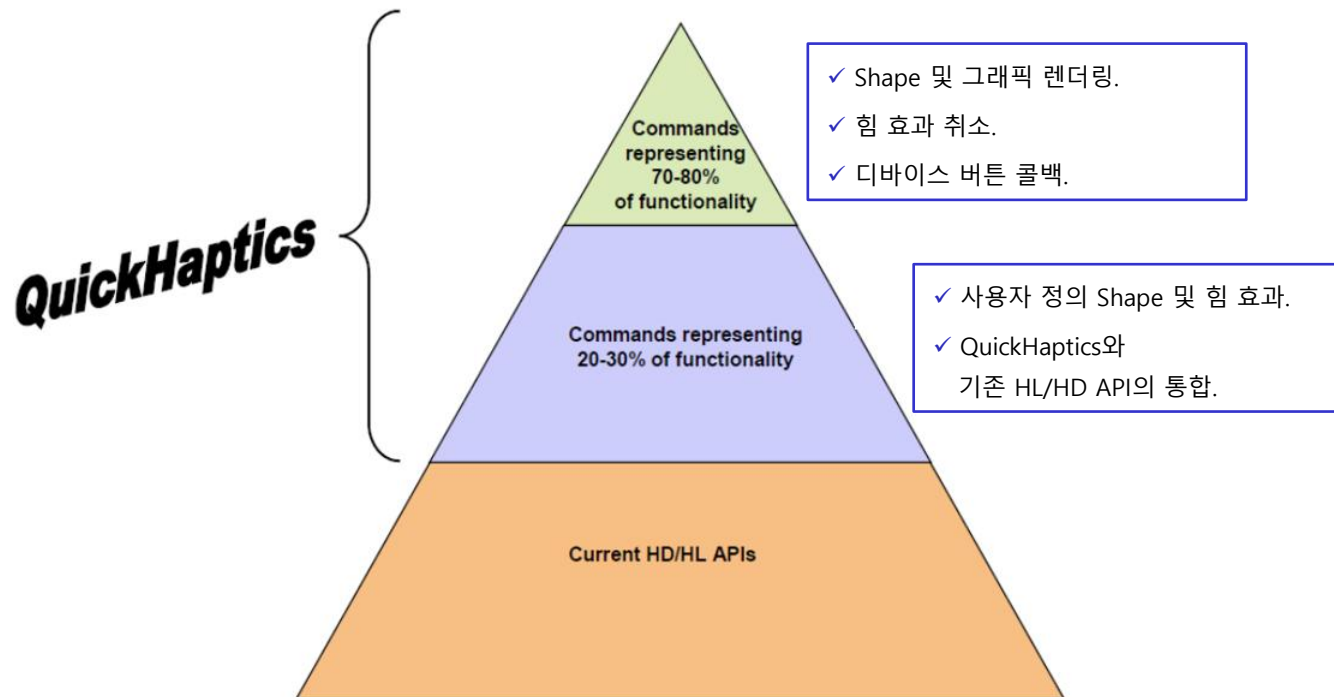
- 모든 햅틱/그래픽 응용 프로그램에 공통적인 기본 단계를 캡슐화(Encapsulating) 하여 프로그래밍을 더 간단하게 만들:
 - ✓ 캡슐화는 QuickHaptics micro API의 C++ 클래스에서 구현됨.
 - ✓ 극적으로 빠르고 쉽게 컴퓨터 햅틱 응용 프로그램에 촉감 부여:
 - ❖ C++에 친숙한 다양한 분야의 전문가가 햅틱 분야가 비 전문 일지라도 빠르고 쉽게 촉감 구현 할 수 있음.
 - ✓ 빠른 프로그램 디자인 및 전개, 기존 프로그램에 촉감 추가, 새로운 아이디어 시도 및 햅틱 예제 생성.
- 일반적인 사용 시나리오를 예상하여, 광범위한 기본 매개변수 설정이 적용되어 사용자가 햅틱 지원 응용 프로그램을 매우 효율적으로 코딩 할 수 있음.
- 햅틱/그래픽 애플리케이션에 필요한 일반적인 단계:
 - ✓ 인기있는 애니메이션 패키지에서 지오메트리 파일 구문 분석.
 - ✓ 그래픽 윈도우 만들기 및 OpenGL 환경 초기화.
 - ✓ 하나 또는 여러 햅틱 장치 초기화.
 - ✓ 장면(Scene) 및 카메라 설정.
 - ✓ 장면 오브젝트에 힘 및 강성 매개변수 맵핑.
 - ✓ 상호작용에 대한 콜백 응답 설정.
- 프로그래머는 햅틱 및 그래픽 매개변수에 대한 기본값 선택을 사용하여, 카메라 위치, 디바이스 공간 매개변수 또는 다양한 모양 속성에 대한 값을 명시적으로 정의 할 필요없이 실행 가능한 장면을 만들 수 있음:
 - ✓ [Default Parameter Values for Shape and Display Windows](#) 참조.

OpenHaptics	v3.5	Touch Device Driver v2023.1.4: <i>for Ethernet Touch or Touch X, USB Touch or Touch X, and HID Touch devices</i> Interface: 	Download	More
		OS: 10/11 (64-bit) Phantom Device Driver v5.1.7: <i>for Firewire or Parallel devices</i> Interface:  OS: 7/8/10 (64 bit)	Download	

OpenHaptics				
v3.5		Touch Device Driver v2022.10.10: <i>for Ethernet Touch or Touch X, USB Touch or Touch X, and HID Touch devices</i> Interface: 	Download	
		OS: 10/11 (64-bit)		
		Touch Device Driver v2018.10.22: <i>for Ethernet Touch or Touch X, and USB Touch or Touch X devices</i> Interface: 	Download	
		OS: 7/8/10 (64-bit)		
		Phantom Device Driver v5.1.7: <i>for Firewire or Parallel devices</i> Interface:  OS: 7/8/10 (64-bit)	Download	

QuickHaptics (2/2)

- QuickHaptics 첫 번째 수준(Level):
 - ✓ 한 페이지 미만의 C++ 코드로 비교적 복잡한 장면을 프로토타입 하는데 사용 할 수 있음.
- QuickHaptics 두 번째 수준(Level):
 - ✓ 사용자 정의 힘 효과, 보다 유연한 모델 상호작용 및 사용자 정의 콜백 함수를 제공하는 함수.
- 세 번째 수준(Level):
 - ✓ 기존 OpenHaptics 2.0의 HLAPI 및 HDAPI 기능이 제공하는 기반 위에 구축.
 - ✓ 대부분의 경우 이전에 OpenHaptics 2.0에서 개발된 응용 프로그램은 OpenHaptics 3.5에서 거의 또는 전혀 수정하지 않고 실행되어야 함.



QuickHaptics Micro API 클래스

QuickHaptics Micro API는 C++로 구현되고 4 가지 기본 기능 클래스를 정의:

✓ DeviceSpace 클래스:

- ❖ 햅틱 디바이스가 이동 할 수 있는 작업 공간(Workspace).

✓ QHRenderer 클래스:

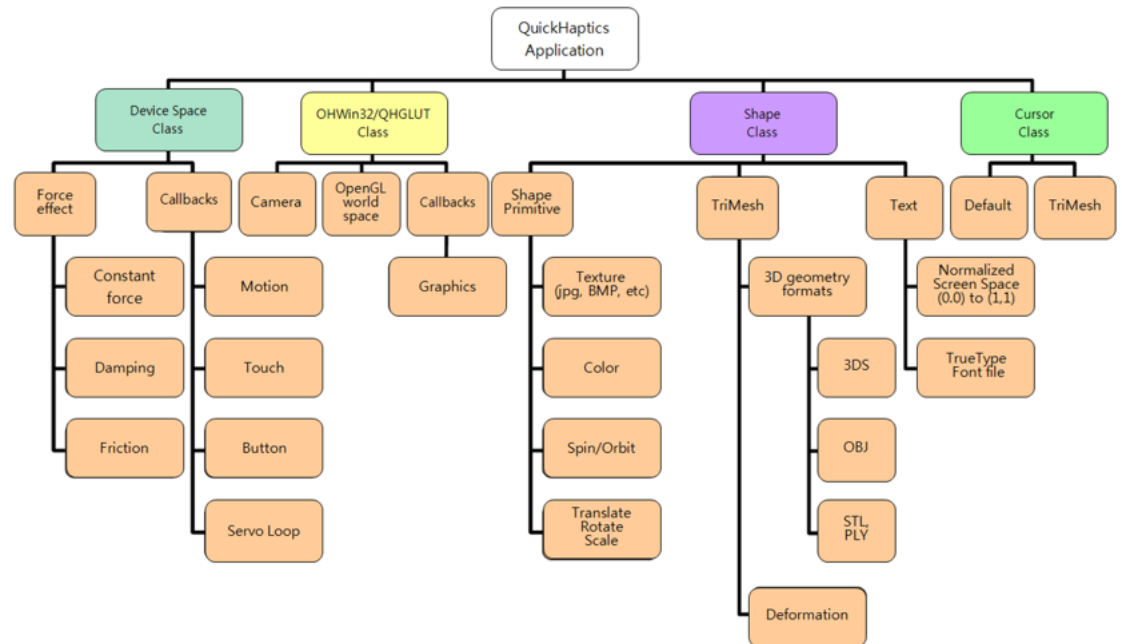
- ❖ QHWin32 및 QHGLUT의 기본 클래스.
- ❖ 카메라 시점(Viewpoint)에서 Shape를 렌더링하고 사용자가 햅틱 디바이스를 사용하여 해당 객체들을 느낄 수 있는 화면 윈도우.

✓ Shape 클래스:

- ❖ 그래픽 및 햅틱으로 렌더링 할 수 있는 하나 이상의 지오메트리 객체에 대한 기본 클래스.

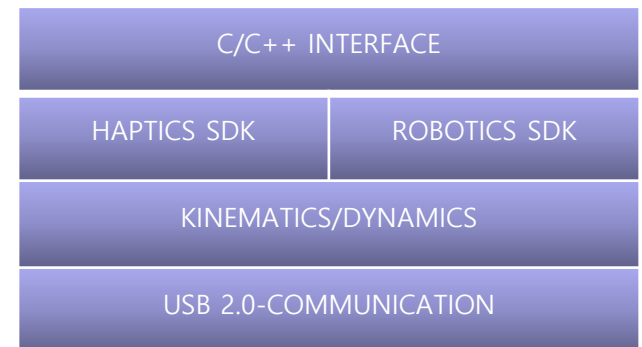
✓ Cursor 클래스:

- ❖ Touch 디바이스에 있는 두 번째 링크의 끝 지점을 그래픽으로 표시.
- ❖ 이 끝 지점은 때때로 햅틱 인터페이스 포인터(HIP)라고도 함:
 - 사진에서 Touch의 HIP 표현.
 - HIP는 다른 Touch 디바이스와 동일한 위치에 있음.



Force Dimension SDK (1/2)

- Force Dimension SDK를 통해 사용자는 햅틱 장치 프로그래밍의 모든 복잡성을 숨겨서 애플리케이션에 햅틱 기능을 쉽게 추가 할 수 있음:
 - ✓ 이 소프트웨어는 Haptic SDK 와 Robotic SDK 의 두 부분으로 구성됨.
 - ✓ Haptic SDK를 사용하는 프로그래머는 코드에 몇 줄의 C / C ++ 만 추가하면 Force Dimension 햅틱 장치의 고 충실도 포스 렌더링을 최대한 활용할 수 있음:
 - ❖ Haptic SDK는 시뮬레이션 응용 프로그램 개발자가 Force Dimension 햅틱 제품을 훨씬 쉽게 사용할 수 있도록 대부분의 기존 햅틱 시각화 패키지와 즉시 호환되는 호환성을 제공.
 - ✓ 고급 제어 작업을 수행해야하는 연구자와 사용자를 위해 Haptic SDK는 Force Dimension 햅틱 장치의 모든 측면에 액세스하고 제어 할 수 있는 다양한 저수준 기능을 제공.
 - ✓ Force Dimension은 햅틱 소프트웨어와 함께 로봇 및 협업 애플리케이션 개발을 목표로 하는 Robotic SDK를 제공:
 - ❖ Robotic SDK를 사용하면 햅틱 장치가 공간에서 안전하고 원활하게 이동할 수 있으므로 사람과 기계가 협업 할 수 있는 새로운 방법을 제공.
- 이식성:
 - ✓ Force Dimension SDK는 모든 주요 플랫폼에서 실행되며, 필요에 따라 Force Dimension 개발 팀에서 보다 전문화된 환경으로 이식 할 수 있음.
- 개방형 제어:
 - ✓ Force Dimension SDK는 고급 제어 작업을 개발하려는 사용자를 위해 Encoder 판독 값, 모터 명령 및 Kinematics 모델과 같은 디바이스의 모든 요소를 완벽하게 제어할 수 있는 기능을 제공.
- 모듈화:
 - ✓ Force Dimension SDK를 사용하면 단일 프로그래밍 인터페이스와 일관된 구문으로 모든 Force Dimension 제품에 대한 응용 프로그램을 개발 할 수 있음.



Force Dimension SDK (2/2)

■ 타사 호환성:

- ✓ 현존하는 대부분의 햅틱 가시화 패키지와 쉽게 사용할 수 있는 호환성 제공:
 - ❖ 햅틱을 사용하는 시뮬레이션 응용 프로그램을 개발자가 쉽게 사용.
 - ❖ SenseGraphics H3D, Matlab, Labview , Simulink 등
- ✓ chai3d SDK:
 - ❖ 햅틱, 시각화 및 대화형 실시간 시뮬레이션을 위한 오픈 소스 소프트웨어 라이브러리:
 - C ++로 작성된 chai3d는 개발자가 3D 모델링과 힘 피드백 렌더링 기능을 결합한 응용 프로그램을 보다 쉽고 직관적으로 만들 수 있도록 설계.
 - 수년 동안 chai3d는 게임, 시뮬레이터, 교육용 소프트웨어, 인터랙티브 아트, 과학적 시각화 및 의료 어플리케이션과 같은 다양한 분야에서 수많은 연구 및 제작 프로젝트에 사용.

■ 멀티-플랫폼 운영 체제:

- ✓ Microsoft : Windows 32-bit.
- ✓ Microsoft : Windows 64-bit.
- ✓ Apple : macOS 14 (Sonoma).
- ✓ Linux : Linux (x86_64).
- ✓ Linux : Linux (armv7l).
- ✓ Linux : Linux (aarch64).
- ✓ BlackBerry (QNX)
- ✓ Wind River : VxWorks.

■ 멀티-컴파일러:

- ✓ Visual C++.
- ✓ Borland C++ Builder,
- ✓ GCC 등



```

60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
// close the connection
dhdClose();
}

if ((t1-t0) > REFRESH_INTERVAL) {
    // update timestamp
    t0 = t1;

    // retrieve position
    if (dhdGetPosition (&px, &py, &pz) < DHD_NO_ERROR) {
        printf ("error: cannot read position (%s)\n", dhdErrorGetLastStr());
        done = 1;
    }

    // retrieve force
    if (dhdGetForce (&fx, &fy, &fz) < DHD_NO_ERROR) {
        printf ("error: cannot read force (%s)\n", dhdErrorGetLastStr());
        done = 1;
    }

    // display status
    printf ("p (%+0.03f %+0.03f %+0.03f) m | f (%+0.01f %+0.01f %+0.01f) N | freq %0.02f kHz\n", px, py, pz, fx, fy, fz, dhdGetComFreq());

    // user input
    if (dhdKbHit() && dhdKbGet() == 'q') done = 1;
}
}

```


CHAI3D (Computer Haptics and Active Interface) (2/3)

■ 햅틱 알고리즘:

- ✓ chai3d는 Finger-proxy 모델, 포텐셜 필드 및 암시적 기반 모델을 포함하여 광범위한 힘 렌더링 알고리즘을 결합하여 프로그래머가 통합된 힘 피드백 기능으로 정교한 시뮬레이션을 쉽게 개발할 수 있도록 함.

■ 그래픽 렌더링:

- ✓ chai3d 프레임 워크는 정적, 동적 및 관절 바디를 포함하는 다단계 장면 그래프 (Multi-level scene graphs)를 생성하는데 필요한 데이터 구조를 제공.
- ✓ 가벼운 OpenGL 기반 그래픽 엔진은 전용 3D 그래픽 가속 하드웨어를 사용하여 가상 환경을 쉽게 렌더링 할 수 있는 기반을 제공.
- ✓ 모든 객체 메시, 암시적 모양, 볼륨, 표면 재질 및 텍스처 속성은 프로그래머가 쉽게 확장하여 고급 또는 응용 프로그램 별 기능을 통합 할 수 있는 잘 구성된 기본 클래스로 표현 됨.

■ 3D 모델:

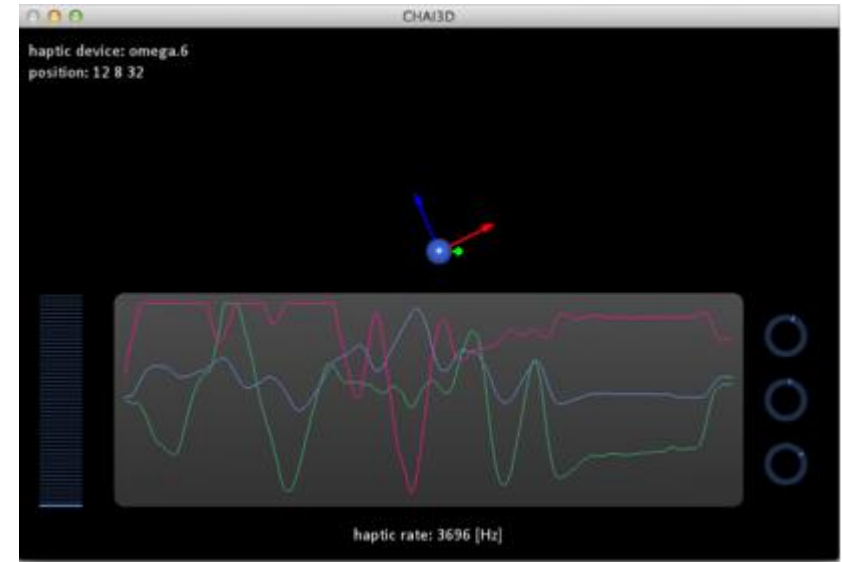
- ✓ chai3d는 Autodesk 3D Studio Max 및 Alias Wavefront와 같은 전문 응용 프로그램에서 3D 파일 가져 오기 및 내보내기를 지원.
- ✓ 글꼴 파일 및 2D 이미지에 대한 지원이 프레임 워크에 포함되어 있음.

■ 확장 모듈:

- ✓ 타사 구성 요소에 대한 지원은 chai3d 핵심 요소의 기능을 독립적으로 보완하는 조직화된 확장 모듈을 통해 이루어 짐:
 - ❖ 확장 모듈에는 현재 강체 및 변형 가능한 물체를 실시간으로 시뮬레이션하기 위한 ODE, BULLET 및 GEL 역학 엔진용 래퍼가 포함되어 있음.
- ✓ 맞춤형 설계된 햅틱 디스플레이 솔루션을 통합하려는 개발자를 위해 잘 문서화된 템플릿이 제공 됨:
 - ❖ 경량의 모듈 식 아키텍처 덕분에 chai3d는 그래픽 또는 역학 엔진을 포함한 타사 라이브러리와 쉽게 결합 할 수 있음.

CHAI3D (Computer Haptics and Active Interface) (3/3)

- 다양한 햅틱 디바이스지원:
 - ✓ Force Dimension 사의 Omega.x, Delta.x, Sigma.7 및 lambda.7 제품 군:
 - ❖ Windows, Linux, Mac OS-X.
 - ✓ 3DSysytems 사의 Phantom, Omni, Touch 제품 군:
 - ❖ Windows.
 - ✓ Novint 사의 Falcon 지원.
- 지원되는 하드웨어:
 - ✓ Leap Motion 사의 Controller:
 - ❖ Windows, Linux, Mac OS-X.
 - ❖ [extras/tdLeap/doc](#) 참조.
 - ✓ Sixense 사의 Razer, Hydra Controller:
 - ❖ Windows, Linux, Mac OS-X.
 - ❖ [external/sixense/doc](#) 참조.
 - ✓ Oculus:
 - ❖ Windows.
 - ❖ [module/OCULUS/doc](#) 참조.
 - ✓ 가상 햅틱 디바이스.
 - ✓ 유니버설 햅틱 디바이스 핸들러.
 - ✓ 사용자 정의 디바이스의 통합을 위한 템플릿.
- 커스텀 디바이스를 위한 I/O 보드:
 - ✓ Servo2Go I/O 보드, Sensoray 626 I/O 보드.
- 컴퓨터 햅틱스, 가시화 및 상호적용적인 실시간 시뮬레이션 프레임워크:
 - ✓ 오픈 소스.
 - ✓ C++ 라이브러리.
 - ✓ 멀티 플랫폼 폼:
 - ❖ Windows, Mac OS-X, Linux.



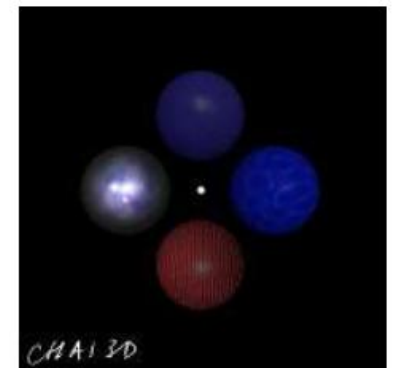
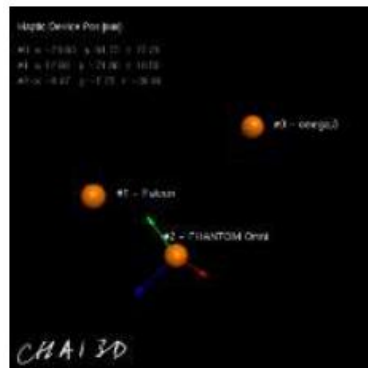
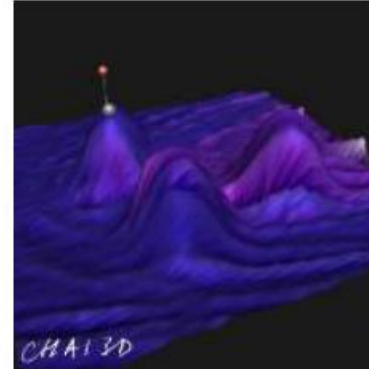
chai3d 기능 (1/2)

■ Core:

- ✓ C++ API다중 플랫폼(Windows, Linux, Mac OS X).
- ✓ 애플리케이션 템플릿(GLUT, .NET, Qt).
- ✓ 스레드(그래픽 및 햅틱).
- ✓ 고정밀 클릭.

■ 그래픽 렌더링:

- ✓ OpenGL 2.1.
- ✓ Scene Graph 구조.
- ✓ 조명 및 그림자 투사.
- ✓ 1D, 2D 및 3D 텍스처 속성.
- ✓ 재질 속성(Material Properties).
- ✓ 포인트 클라우드.
- ✓ 볼륨(Volumes).
- ✓ 라인 세그먼트.
- ✓ 메시 객체(Mesh objects).
- ✓ 형상(Shape) 프리미티브:
 - ❖ Box, Cylinder, Sphere, Torus, Cone, Plane.
- ✓ 2D 위젯.
- ✓ 비트 맵 폰트.
- ✓ 프로그램 셰이더.



■ 오디오:

- ✓ OpenAL.
- ✓ 햅틱 사운드 프리미티브.

chai3d 기능 (2/2)

■ 힘 알고리즘:

- ✓ Finger-Proxy 모델.
- ✓ 볼륨 렌더링.
- ✓ Potential field.
- ✓ Coulomb 마찰(Friction) 모델.
- ✓ Stick-slip, 점성(Viscous).
- ✓ 진동(Vibration) 및 자성(Magnetic) 효과.
- ✓ 멀티 포인트 접촉(Grasping).

■ 충돌 검출:

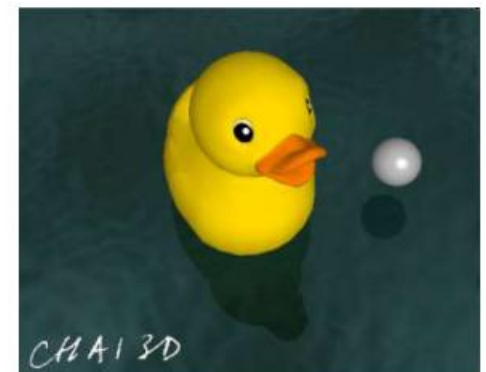
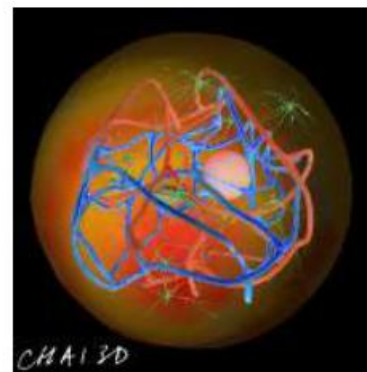
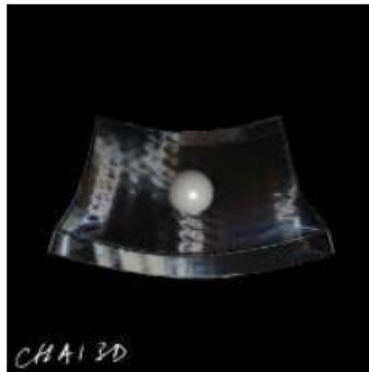
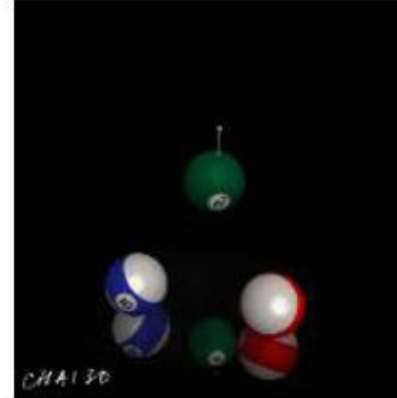
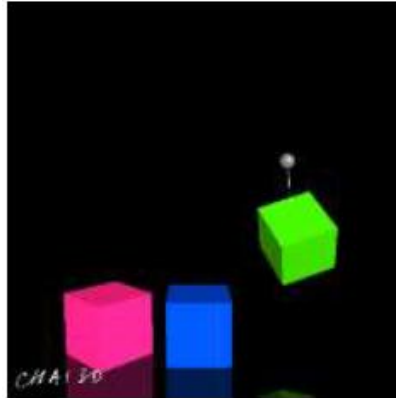
- ✓ AABB(Axis-aligned bounding box) 모델.

■ 모듈:

- ✓ 강체 역학(ODE).
- ✓ 변형(Deformable) 모델(GEL).
- ✓ 로봇틱스 시뮬레이션(V-REP).
- ✓ Oculus Rift HMD(OCULUS).
- ✓ BULLET.

■ 다양한 파일 형식 지원:

- ✓ 이미지 파일 :
 - ❖ .BMP, .GLF, .JPG, .PNG, .PPM, .RAW.
- ✓ 메시 파일:
 - ❖ .3DS, .OBJ, .STL.
- ✓ 오디오 파일:
 - ❖ .WAV.



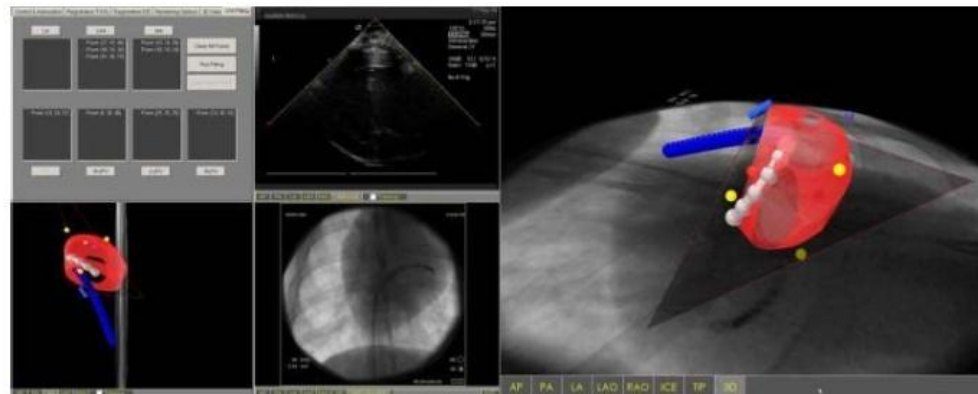
범용 햅틱 디바이스를 위한 소프트웨어 개발 툴킷

■ 사용자 정의 소프트웨어 개발:

- ✓ 오픈 소스 패키지의 일부가 아닌 것:
 - ❖ 비용, 소유권 및 개발 과제의 일환으로 협정된 소스 코드 접근(Access).
- ✓ 고객의 요구에 맞게:
 - ❖ caid3d/Force Dimension SDK 기반 및 사용자 3D 모델 사용.
- ✓ 추가적인 기능 포함:
 - ❖ 로봇 제어 인터페이스, 제어 콘솔 디자인 등

■ 응용 프로그램 개발:

- ✓ 응용 프로그램의 사용자 인터페이스 개발:
 - ❖ GUI 및 HUI, GUI 위젯과 시뮬레이션 하드웨어 구성 요소를 사용.
- ✓ 개념 증명(Proof-of-concept)의 빠른 개발:
 - ❖ 예를 들어, Hansen Medical의 외과 콘솔 시뮬레이터.



■ 사용자 정의 고급 충돌 엔진(Collision Engine):

- ✓ 3D 메시 객체 (CAD 에서) 기반:
 - ❖ 그래픽 및 햅틱 렌더링에 대한 동일 모델.
 - ❖ 지역의 세부 사항 및 전체 크기 사이의 높은 비율을 허용.
 - ❖ 껍질(Shell) 및 얇은 곡선 시트 (Sheets)의 렌더링에 적합.
 - ❖ 객체 사이의 고유한 거리 계산.
 - ❖ 충돌 침입 방지.
- ✓ 고유의 성능:
 - ❖ 높은 재생률 및 높은 해상도.
 - ❖ 고 품질의 햅틱 렌더링 품질.
 - ❖ 하위 구동 인터페이스와의 호환.

