GLOVE 사용자 가이드

2017년

KiSTi www.kisti.re.kr
한국과학기술정보연구원
Korea Institute of Science and Technology Information

제목 차례

표 차례

그림 차례

1. 환경 개요

GLOVE는 고성능 컴퓨팅 환경에서 만들어진 대용량 해석데이터의 가시화/분석을 위한 소프트웨어다. 이 소프트웨어는 대량의 가시화 계산을 위한 전용 클러스터와 VR 입/출력 장치, VR 입/출력 장치를 운영하기 위한 - 비교적 소규모의 - 전용 디스플레이 클러스터 등 서로 다른 특성을 갖는 하드웨어를 동시에 사용하기 때문에 처음 접하는 사용자들에게는 설정이나 운용환경이 다소 복잡하게 느껴질 수 있다. 이 문서는 가상의 기관에서 형태가 조금씩 다른 VR 입/출력 장치와 전용 클러스터를 운용하는 경우를 가정하고, 해당 환경을 위한 GLOVE 설정방법에 대해 설명한다. 가상의 기관이 보유하고 있는 데이터 가시화 시스템의 구성을 그림 1과 같이 가정하자.

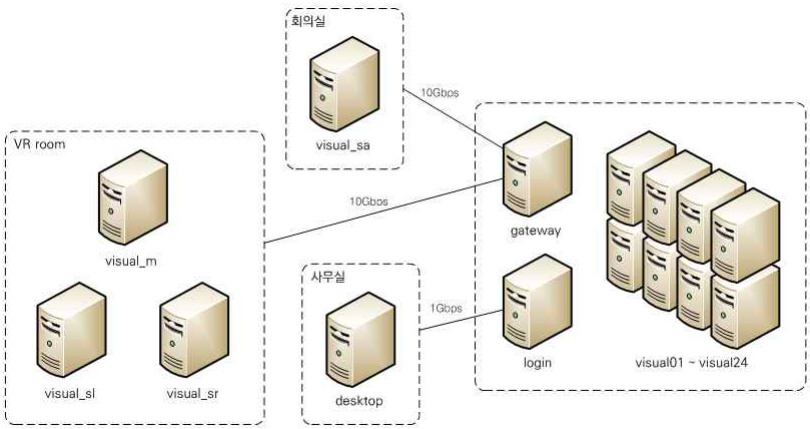

그림 1. 가상기관의 가시화 시스템 구성도

그림에서 VR room은 대형 스크린과 두 대의 고해상도 프로젝터, 6자유도 입력장치가 구비되어 있는 전용 공간을 가정한다. VR room의 콘솔과 6자유도 입력장치, 프로젝터를 운용하는 컴퓨터 역시 visual 클러스터의 일부로 운용되기는 하지만 비디오 케이블의 길이, 콘솔과의 연결거리 등을 감안해서 전산실이 아닌 VR room에 설치되어 있다. 이 노드들은 10Gbps 이더넷을 통해서 visual 클러스터의 게이트웨이 노드에 연결된다.

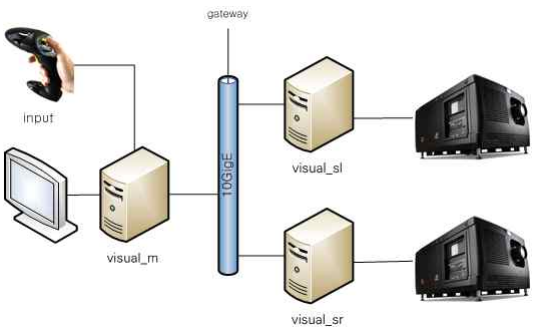

그림 2. VR room의 시스템 구성

회의 도중 필요할 때 자리를 이동하지 않고 소규모 VR 입/출력 시스템을 이용해서 데이터를 분석하기 위한 목적으로 가시화 시스템을 구축할 수도 있다. 그림 1에서 '회의실'이라고 표기한 부분으로 한 대의 그래픽 워크스테이션과 입체영상 출력이 가능한 TV 패널, 6자유도 입력장치 등을 사용할 수 있는 것으로 가정한다.(그림 3)

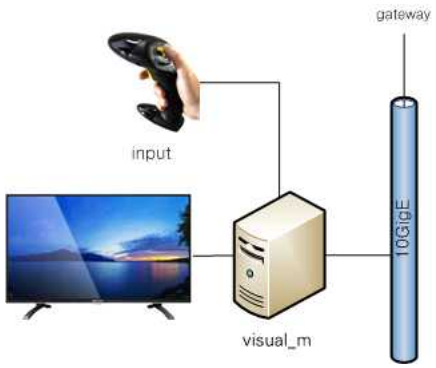

그림 3. 회의실의 가시화 시스템 구성

그림에서 보듯이 회의실의 워크스테이션 역시 10Gbps 이더넷을 통해서 visual 클러스터의 gateway 노드에 접속할 수 있는 것으로 가정한다.

마지막으로 사무실의 개인공간에서 데스크톱 클라이언트나 원격 터미널 프로그램을 이용해서 visual 클러스터에 접속할 수도 있을 것이다. 이때에는 전용 gateway 노드가 아니라 login 노드를 통해서 visual 클러스터에 접속하고, 네트워크도 일반적인 1Gbps 이더넷을 이용하는 것으로 가정한다.

2. GLOVE의 구성

소프트웨어 구성의 관점에서 보면 GLOVE에는 GLORE, 클라이언트, 데이터 변환기, 기타 쉘 스크립트와 리소스 파일(shader, 텍스처 등)이 포함되어 있다.

구분		실행파일 명	특이사항
GLORE		glorem glorew	glorem : GLORE master glorew : GLORE worker
클라이언트	GIVI	givi	GLOVE 가상현실 인터페이스
	데스크톱 클라이언트	-	MS-Windows 용 데스크톱 클라이언트
GIP		giviGipClient giviGipServer	GLORE-GIVI 간 데이터 통신 프로그램
데이터 변환기			
쉘 스크립트		run.sh	GLORE 실행을 위한 쉘 스크립트
		killall.sh	GLORE 실행 종료를 위한 쉘 스크립트
		delipc.sh	IPC shared memory 및 세마포어 정리용 쉘 스크립트

표 1. GLOVE의 실행 가능한 프로그램 구성

① GLORE

GLORE는 데이터 관리와 가시화 연산을 위한 MPI 프로그램이다. GLORE의 첫 번째 역할은 스토리지로부터 사용자가 지정한 해석 데이터를 읽어서 클러스터의 각 노드가 갖고 있는 메모리에 분산 저장하는 것이다. 이후 클라이언트의 요청이 있을 때 필요한 데이터를 추출해서 지오메트리로 변환한 후 클라이언트에게 보내준다. 그림 4에서 보듯이 GLORE는 마스터/슬레이브 구조를 갖고 있으며, 클라이언트와의 통신은 별도의 매니저 프로세스가 담당한다. 그리고 매니저 프로세스와 마스터 프로세스는 클러스터의 동일한 노드에서 실행된다.

② GIVI

GIVI는 단일 워크스테이션이나 디스플레이 클러스터에서 실행할 수 있는, 가상현실 입/출력을 지원하는 리눅스 기반 클라이언트다. GLORE의 경우 마스터와 슬레이브에서 실행하는 프로그램이 서로 독립적으로 존재하지만 GIVI는 하나의 실행파일로 마스터/슬레이브/standalone 모드를 모두 지원하는 차이가 있다.

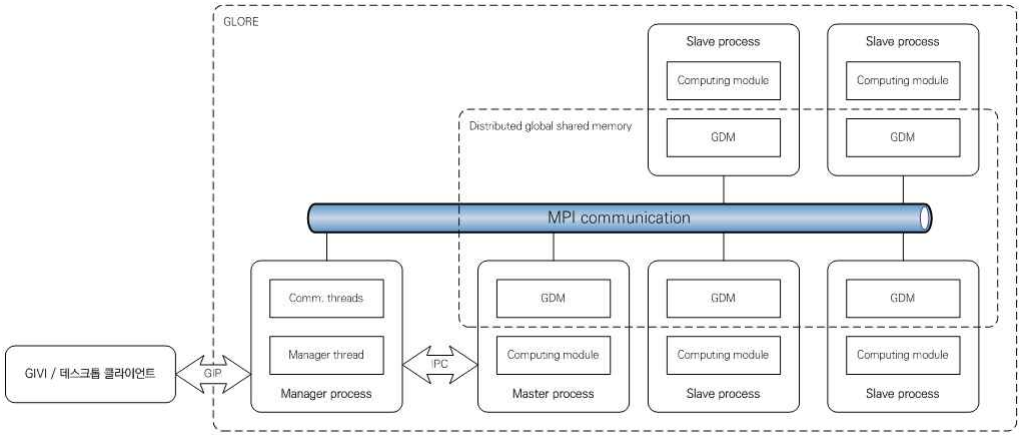

그림 4. GLORE / GIVI / GIP / 데스크톱 클라이언트

③ 데스크톱 클라이언트

데스크톱 클라이언트는 MS-Windows가 설치되어 있는 개인 PC에서 실행할 수 있는 전용 프로그램이다. GIVI와는 다르게 상대적으로 복잡한 작업이 가능하고, 2차변수의 계산 등 GIVI가 지원하지 않는 기능을 수행할 수 있는 강점도 존재한다.

④ GIP

GIP은 GLORE와 클라이언트 사이의 통신을 담당하는 프로그램이다. 이 프로그램을 사용자가 직접 실행하는 경우는 없으며, 문제가 있을 때 특정 노드에서 해당 프로그램이 정상적으로 실행되고 있는지의 여부만 확인할 수 있으면 충분하다.

⑤ 데이터 변환기

Lustre나 GPFS와 같은 병렬 파일시스템은 MPI-IO를 지원하기 때문에 하나의 파일을 읽을 때에도 여러 클라이언트가 파일의 서로 다른 부분을 읽는 등 스토리지 I/O 효율을 향상시킬 수 있는 방법이 존재한다. 하지만 MPI-IO를 지원하지 않는 병렬 파일시스템은 전체 데이터를 단 한 대의 클라이언트가 읽어야 하는 상황이 발생하기 때문에 GLOVE는 데이터를 여러 개의 파일로 분할 저장하는 방식을 채택하고 있다.

GLOVE에 포함되어 있는 데이터 변환기는 상용 어플리케이션의 데이터 포맷을 GLOVE의 독자적인 방식으로 변환하거나, 단일 블록으로 구성되어 있는 데이터를 분할저장하기 위한 프로그램들의 집합이다. 여기에 포함되는 프로그램들은 모두 독립적으로 실행이 가능하며, 빠른 데이터 변환을 위해 병렬처리도 지원한다.

⑥ 쉘 스크립트

쉘 스크립트는 GLOVE의 실행과 종료와 관련이 있는 bash 스크립트의 집합이다.

3. 사전 설정

이 장에서는 그림 1과 같은 환경에서 GLOVE를 실행할 때 필요한 준비사항에 대해 설명한다.

가. 클러스터 노드 구성

그림 1의 환경에서 가장 특이한 사항은 (visual) 클러스터 한 세트라고 해도 일부 노드가 물리적으로 다른 곳에 떨어져 있다는 점이다. 이는 일반적인 계산 시스템에서 보기 힘든 구성이지만 가시화 장비의 규모가 조금만 커지면 장비를 운용하는 노드 일부는 부득이하게 클러스터로부터 멀리 떨어져 있을 수밖에 없다. 그러면서도 클러스터의 홈 디렉터리나 사용자 계정은 동일하게 사용할 수 있어야 하므로 그림 1의 gateway 노드가 반드시 필요하게 된다. 이를 통해 홈 디렉터리 접근, 사용자 계정 정보 접근, NTP 등을 이용한 시간 동기화 등 일원화 된 클러스터 관리체계를 구축하는 것이 바람직하다.

그림 1과 같은 환경에서 주의할 사항은 주력 계산 시스템과 visual 클러스터 사이의 데이터 전송이다. 가장 쉬운 해결방법은 계산 시스템과 visual 클러스터를 동일한 인터커넥트에 연결하고 스토리지를 공유하는 것이지만, 이 접근방법의 여의치 않을 경우에는 두 시스템 사이의 빠른 데이터 전달경로를 구성해야 한다. 이는 단순히 두 시스템의 한 노드를 데이터 교환 전담 노드로 할당하는 수준이 아니라, 각각의 시스템으로부터 여러(많게는 10대 이상) 노드를 데이터 전송용으로 차출해서 수십 GB/s 이상의 데이터를 전송속도를 구현하는 것을 의미한다.

소프트웨어 관점에서 보면 가시화 작업의 특성상 배치스케줄러가 거의 필요하지 않다. 하지만 데이터 변환의 경우에는 사용자 입력을 거의 필요로 하지 않고, 데이터의 크기에 따라서 오랜 시간이 걸리는 작업이 될 수도 있기 때문에 배치작업으로 실행해도 괜찮을 것이다. 따라서 visual 클러스터의 리소스 일부를 배치작업 전용으로 할당해서 전용 배치 스케줄러를 운용하는 방법도 생각해볼 수 있다.

나. 사용자 계정

그림 1에 나타내지는 않았지만 일반적으로 클러스터에는 계정관리를 담당하는 전용 노드가 따로 존재한다. 여기서는 편의상 NIS[1] 서버가 운용되는 것으로 간주하자. 앞에서도 설명했듯이 visual 클러스터와는 완전히 다른 물리적 공간에 설치되어 있는 VR room이나 회의실의 컴퓨터 역시 NIS 서버에 접속할 수 있도록 gateway 노드를 설정하는 것이 바람직하다. 그렇게 하면 VR room과 회의실에서 모두 동일한 계정으로 가상현실 입/출력 장치(프로젝터, TV 패널)와 클러스터를 활용할 수 있다.

표 2에서 볼 수 있듯이 GLOVE를 실행하기 위한 사용자 계정은 디스플레이 전용 계정과 일반 사용자 계정의 두 가지로 구분한다.

1) Network Information Service

구분	내용
디스플레이 전용 계정	프로젝터 출력 노드 전용
일반 사용자 계정	GLOVE 사용자

표 2. GLOVE를 위한 사용자 계정 분류

일반 사용자 계정은 글자그대로 사용자들이 visual 클러스터에 접속해서 GLOVE를 실행할 때 이용한다. 따라서 모든 GLOVE 관련 명령은 일반 사용자 계정으로 실행하면 된다. 반면 디스플레이 전용 계정은 VR room에서 프로젝터 출력을 담당하는 노드에서만 사용한다. 구체적으로 설명하면 그림 1에서 VR room에 있는 visual_sl과 visual_sr이 각각 프로젝터 출력을 담당하는데, 이 두 노드는 디스플레이 전용 계정으로 상시 로그인이 되어있다. 또 임의의 사용자가 X-Windows 전용 프로그램을 자유롭게 실행할 수 있도록 디스플레이 출력에 대한 보안 수준을 낮추고, 화면보호기의 실행도 제한한 상태가 유지된다.

다. 네트워크 포트

GLORE는 네트워크를 통해서 GIVI나 데스크톱 클라이언트와 명령/데이터를 주고받는다. 그림 1과 같은 환경이라면 VR room이나 회의실에서 GLOVE를 실행하는 사용자는 모두 gateway 노드에서 GLORE를 실행해야 한다. 만약 visual 클러스터의 다른 노드도 외부와의 통신이 자유롭다면 visual01~visual24 중 아무 노드에서나 GLORE를 실행할 수 있겠지만 이는 보안문제를 고려해보면 썩 좋은 접근방법이 아니다. 따라서 여기서는 gateway 노드에서만 GLORE를 실행하는 것으로 가정한다. 이 때 서로 다른 사용자가 실행한 glorem이 동일한 포트를 사용할 가능성이 있으므로, 이를 방지하기 위해 사전에 각각의 사용자가 사용하는 네트워크 포트를 할당해 놓는 것이 유리하다. 네트워크 포트 설정은 뒤에서 설명할 glove.conf에서 변경할 수 있다.

4. GLOVE 설정

가. 일반사항

표 3은 GLOVE를 구성하는 각 프로그램이 사용하는 설정파일의 목록을 보여준다.

프로그램	설정파일	세부내용
glorem glorew	hosts.conf	• GLORE가 실행되는 호스트 설정 (MPI 프로그램)
	glove.conf	• glroem이 클라이언트와의 통신을 위해 사용할 네트워크 포트 정보 확인
	log4cxx.conf	• ${HOME}/.glove/log에 프로그램 실행 관련 로그를 남길 때의 설정 • 대부분의 경우 관리자만 사용
giviGipServer	glove.conf	• giviGipClient와의 통신을 위한 네트워크 포트
	log4cxx.conf	• ${HOME}/.glove/log에 프로그램 실행 관련 로그를 남길 때의 설정 • 대부분의 경우 관리자만 사용
giviGipclient	glove.conf	• giviGipClient를 실행하는 호스트 정보 • giviGipServer와의 통신을 위한 네트워크 포트
	log4cxx.conf	• ${HOME}/.glove/log에 프로그램 실행 관련 로그를 남길 때의 설정 • 대부분의 경우 관리자만 사용
givi	glove.conf	• GIVI가 실행되는 노드 구성, IPC 공유 메모리, GLORE 호스트 정보 등 GIVI의 실행에 필요한 전반적인 설정
	widget.conf	• GIVI의 인터페이스 구성 방식 설정
	log4cxx.conf	• ${HOME}/.glove/log에 프로그램 실행 관련 로그를 남길 때의 설정 • 대부분의 경우 관리자만 사용

표 3. GLOVE의 세부 설정파일 목록과 각 설정파일을 사용하는 프로그램

나. hosts.conf

hosts.conf는 glorew가 어떤 호스트에서 실행되는지를 지정하기 위해 사용한다. 이 파일은 MPI 프로그램을 실행할 때 사용하는 machine file (또는 hostfile)과 거의 유사한 형태를 갖고 있는데, 가장 첫 줄에 호스트 수가 나온다는 점에서 차이가 있다.

```
# cat ~/.glove/config/hosts.conf
24
visual01
visual02
중략
visual24
```

그림 5. hosts.conf

다. glove.conf

glove.conf는 GLOVE의 실행에 필요한 설정을 담고 있다. 이 파일은 GLORE, GIVI뿐만 아니라 giviGipserver, giviGipClient 등 GLOVE를 구성하는 세부 프로그램들이 모두 참조하며, 전체 프로그램의 성능이나 처리 가능한 데이터 량 등을 결정할 수 있기 때문에 세부적인 설정 값을 변경할 때 주의를 기울여야 한다.

1) GLORE 관련 설정

glove.conf는 크게 GLORE 설정과 GIVI 설정의 두 부분으로 나눌 수 있다. 그 중 GLORE 관련 설정은 표 4에 나타냈다.

XML 태그	세부내역
master	GLORE master(glorem)가 실행되는 호스트의 IP 주소 또는 호스트명
sync	데스크톱 클라이언트가 glorem에 접속할 때 사용하는 호스트 주소 및 네트워크 포트
async	VR 클라이언트가 glorem에 접속할 때 사용하는 호스트 주소 및 네트워크 포트
ga-max-memory	GLORE가 실행될 때 클러스터의 각 노드에서 할당받을 분산공유메모리의 양(GB)
directSendFlag	

표 4. glove.conf의 구성 중 GLORE 관련 설정항목

① master : GLORE 마스터(glorem)가 실행되는 노드의 호스트명(hostname)이나 IP 주소를 적는다.

② sync : MS-Windows 용 데스크톱 클라이언트를 사용할 때 필요한 설정이다.

	세부내용
ip	MS-Windows 클라이언트가 glorem이 실행되는 노드에 접속할 때 사용할 IP 주소
port	MS-Windows 클라이언트가 glorem이 실행되는 노드에 접속할 때 사용할 네트워크 포트
max-connection	GLORE에 동시 접속할 수 있는 데스크톱 클라이언트의 수

표 5. MS-Windows 용 데스크톱 클라이언트 관련 설정

③ async : VR 클라이언트(GIVI)를 사용할 때 필요한 설정이다.

	세부내용
ip	GIVI가 glorem이 실행되는 노드에 접속할 때 사용할 IP 주소
port	GIVI가 glorem이 실행되는 노드에 접속할 때 사용할 네트워크 포트
max-connection	GLORE에 동시 접속할 수 있는 GIVI의 수

표 6. VR 클라이언트 관련 설정

④ ga-max-memory : glorew가 실행될 때 각 노드에서 몇 GB의 메모리를 Global Arrays에게 할당할 것인지를 결정하기 위해 사용한다.

⑤ directSendFlag : GLORE 내에서 응답시간을 줄이기 위해 별도로 구현한 네트워크 전송기법을 사용할지의 여부를 결정한다. 아주 특별한 경우가 아니면 사용자가 이 값을 수정할 필요는 없다.

2) GIVI 관련 설정

XML 태그		세부내용
mode		GIVI가 단일 노드에서 실행될 때 : single GIVI가 여러 노드에서 실행될 때 : multi
vrj		VR Juggler 설정파일(절대경로), 현재는 사용하지 않음
gip-server	external	giviGipClient가 실행되는 호스트 정보
	internal / master	
	internal / port	giviGipClient와 giviGipServer 사이의 통신을 위한 네트워크 포트
	internal / slave-network	인피니밴드 : ib, 이더넷 : ethernet
	internal / slave-count	디스플레이 클러스터를 사용할 때 디스플레이 출력을 담당하는 노드의 수
	internal / slaves	giviGipServer가 실행되는 호스트 정보
glore		glorem이 실행 중인 호스트에 접속할 때 필요한 사용자 계정과 포트 번호
mdb		현재 실행 중인 명령, GLORE로부터 전송받은 geometry를 저장하기 위한 메모리 데이터베이스
msgq		
wand	joystick	6자유도 입력장치의 조이스틱이 갖는 좌표의 범위 (X 축, Y 축)
	button	VR Juggler에서 정의한 입력장치 버튼 중 어떤 버튼을 GIVI의 기능에 대응할지를 결정하기 위해 사용
colormap		사용자 정의 컬러 맵

표 7. glove.conf의 구성 중 GIVI 관련 설정항목

① mode : GIVI가 한 대의 컴퓨터에서 실행될 때에는 'single', 여러 대의 컴퓨터에서 마스터/슬레이브 형태로 실행될 때에는 'multi'로 지정한다.

② vrj : VR Juggler 설정파일의 위치로 현재는 사용하지 않는다.

③ gip-server : giviGipServer가 실행되는 노드(= GIVI 마스터와 GIVI 슬레이브가 실행되는 모든 노드)

	세부내용
host-name	
bind-ip	
port	네트워크 포트

표 8. givi/gip-server/external

master	host-name		호스트 이름
	bind-ip		네트워크 접속에 사용할 IP 주소. 여러 개의 네트워크 카드를 갖고 있는 경우를 대비한 설정
	multicast		사용하지 않음
port			giviGipServer와 giviGipClient 사이의 통신포트
slave-network			giviGipServer와 giviGipClient 사이의 통신채널 인피니밴드 : ib, 이더넷 : ethernet
slave-count			슬레이브 노드의 수
slaves	slave	host-name	호스트 이름
		bind-ip	네트워크 접속에 사용할 IP 주소. 여러 개의 네트워크 카드를 갖고 있는 경우를 대비한 설정
		multicast	사용하지 않음

표 9. givi/gip-server/internal

④ glore : GLORE가 읽는 데이터는 모두 GLORE가 실행되는 클러스터의 스토리지에 저장되어 있는 것으로 가정한다. 에게 데이터를 읽도록 명령을 내릴 때 GIVI가 GLORE에 접속할 때 사용할 사용자 계정과 포트 번호. 데이터 파일 검색을 위해 필요하다.

	세부내용
username	glorem에 접속할 때 사용할 사용자 계정
port	glorem에 접속할 때 사용할 네트워크 포트

표 10. GIVI가 데이터 파일 검색을 위해 GLORE에 접속할 때 사용할 사용자 계정과 포트

⑤ mdb : GIVI를 실행하는 각 노드에서 transaction DB와 response DB에 할당하기 위한 IPC shared memory에 대한 설정을 기술한다. Transaction DB는 '현재 진행 중인 가시화 작업'에 대한 정보를 담고 있으며 용량이 작은 공유메모리 슬롯을 지정한 개수만큼 할당한다. 그리고 Response DB는 GLORE로부터 전송받은 geometry를 저장하기 때문에 용량이 비교적 큰 공유메모리 슬롯을 지정한 개수만큼 할당한다. Transaction DB와 response DB 모두 동일한 설정항목을 갖고 있다.

	세부내용
slot-size	메모리 슬롯의 크기 (bytes)
slot-cnt	메모리 슬롯의 개수
key	IPC shared memory key

표 11. Transaction DB와 response DB의 설정항목

⑥ msgq : 시스템 내부에서 사용하는 값이며, 타 어플리케이션과 충돌이 발생하기 전까지는 사용자가 굳이 수정할 필요가 없다.

⑦ wand : 6자유도 입력장치에 대한 설정을 위해 사용한다. GIVI는 가상현실 입력장치를 조이스틱과 버튼의 조합으로 처리하는데, 이 중 조이스틱의 X-Y 좌표의 범위가 디바이스마다 다를 수 있기 때문에 이에 대한 정보를 joystick 태그를 통해서 GIVI에게 알려준다.

	세부내용
xmin / xmax	6자유도 입력장치의 조이스틱을 움직일 때 X, Y축 방향 좌표 값의 범위
ymin / ymax	

표 12. Wand joystick의 설정항목

button은 GIVI의 각 기능을 실행하는 데에 필요한 버튼을 정의한다. 버튼의 명칭은 VR Juggler 설정 파일에서 정의한 것을 그대로 사용한다.

	세부내용
menu	GIVI에서 command mode / visualization mode 사이의 전환을 위해 사용하는 버튼
mode	Visualization mode에서 rotation / translation 전환을 위해 사용하는 버튼
select	메뉴항목 선택, visualization object 선택 등
keyframe	현재는 사용하지 않음

표 13. Wand joystick의 설정항목

⑧ colormap : 사용자가 직접 정의한 컬러 맵을 기술한다.

```
〈colormap〉
    〈name〉Fuselage vorticity slice〈/name〉
    〈position〉0.0, 0.005, 0.01, 0.015, 0.02, 1.00〈/position〉
    〈red〉      0.00, 0.00, 0.00, 1.00, 1.00, 1.00〈/red〉
    〈green〉    0.00, 1.00, 1.00, 1.00, 0.00, 0.00〈/green〉
    〈blue〉     1.00, 1.00, 0.00, 0.00, 0.00, 0.00〈/blue〉
    〈alpha〉    1.00, 1.00, 1.00, 1.00, 1.00, 1.00〈/alpha〉
〈/colormap〉
```

그림 6. 사용자 정의 컬러맵의 형태

사용자가 기술하는 컬러 맵의 특징은 데이터 값의 범위를 [0.0, 1.0]으로 정규화 한다는 점이다. 예를 들어서 어떤 데이터의 값의 범위가 [-121.0, 72.5]이고 데이터 값 17.7을 노란색, -121.0과 72.5는 검정색으로 표현한다고 가정하자. GIVI 컬러맵을 정의할 때에는 [-121.0, 72.5]를 [0, 1]로 맞추기 때문에 17.7은 그림 7와 같이 0.7168에 해당한다.

$$\frac{17.7-(-121.0)}{72.5-(-121.0)} \approx 0.7168$$

그림 7. 17.7의 정규화

이를 반영한 컬러 맵은 그림 8와 같이 정의한다. 주의할 사항은 (데이터 값, R, G, B, A)와 같은 형태로 컬러 맵을 정의하는 것이 아니라 데이터 값의 집합, 각 데이터 값에 대한 R의 집합, 각 데이터 값에 대한 G의 집합 과 같은 방식으로 색상을 기술한다는 점이다. 다시 말해서 텍스트를 컬럼 단위로 읽을 때 비로소 (데이터 값, R, G, B, A)를 알 수 있게 된다.

```
〈colormap〉
    〈name〉User defined colormap〈/name〉
    〈position〉 0.00,   0.7168,  1.00〈/position〉
    〈red〉       0.00,   1.00,    0.00〈/red〉
    〈green〉     0.00,   1.00,    0.00〈/green〉
    〈blue〉      0.00,   0.00,    0.00〈/blue〉
    〈alpha〉     1.00,   1.00,    1.00〈/alpha〉
〈/colormap〉
```

그림 8. 사용자 정의 컬러맵

라. widget.conf

GIVI가 실행되는 환경은 어느 하나로 특정할 수 없다. 대형 스크린과 프로젝터, 디스플레이 클러스터가 사용될 수도 있고, 한 대의 컴퓨터와 TV 패널이 될 수도 있다. 그에 따라서 viewport의 크기나 해상도가 달라지기 때문에 메뉴의 크기나 위치 등 화면에 출력하는 사용자 인터페이스에 대한 세부 설정이 가능해야 한다. GIVI는 이를 위해 widget.conf라는 설정파일을 사용하며, ${HOME}/.glove/config에 위치한다. 그림 9는 실제로 사용하는 widget.conf의 내용을 보여준다.

```
〈?xml version="1.0"?〉

〈widget〉
  〈dialpanel〉
    〈deltaangle〉10.0〈/deltaangle〉
    〈diameter〉0.56〈/diameter〉
    〈quantity〉7〈/quantity〉
    〈depth〉1〈/depth〉
    〈location〉RIGHT〈/location〉
    〈position〉11.24654, 3.0, -10.0〈/position〉
    〈posttranslate〉0.0, 3.0, -10.0〈/posttranslate〉
  〈/dialpanel〉

  〈dialpanel〉
    〈deltaangle〉10.0〈/deltaangle〉
    〈diameter〉0.8〈/diameter〉
    〈quantity〉7〈/quantity〉
    〈depth〉2〈/depth〉
    〈location〉LEFT〈/location〉
    〈position〉-14.63812, 3.0, -10.0〈/position〉
    〈posttranslate〉0.0, 3.0, -10.0〈/posttranslate〉
```

```
  〈/dialpanel〉

  〈highlight〉
    〈color〉1.0, 1.0, 0.0〈/color〉
    〈linewidth〉4.0〈/linewidth〉
  〈/highlight〉
〈/widget〉
```

그림 9. widget.conf의 실제 내용 구성

그림 9의 설정파일에서 사용하는 각각의 XML 태그는 표 14에서 자세히 설명하고 있다.

항목		세부내용
dialpanel	deltaangle	• 다이얼 메뉴 아이템의 상대적인 거리(각도) • 허용 값의 범위 : 1.0 ~ 20.0
	diameter	• 다이얼 메뉴를 구성하는 각 메뉴 아이템(버튼)의 지름 • 허용 값의 범위 : 0.1 ~ 10.0
	quantity	• 다이얼 메뉴가 한 번에 출력할 수 있는 메뉴 아이템의 개수 • 허용 값의 범위 : 3 ~ 7
	depth	• 허용 값의 범위 : 1 ~ 2
	location	• 해당 다이얼 패널의 위치 • 가능한 값 : LEFT(메인 패널) 또는 RIGHT(옵션 메뉴)
	position	• 가상현실 공간에서의 다이얼 패널의 위치 (x, y, z)
	posttranslate	•
highlight	color	• 다이얼 메뉴 아이템이 활성화 됐을 때의 색을 RGB로 정의 • 기본 값 : 노란색 (1.0, 1.0, 0.0)
	linewidth	• 활성화 된 메뉴 아이템의 선 두께 • 기본 값 : 4.0

표 14. widget.conf의 구성

5. GLOVE 실행/종료

이 장에서는 GLORE와 GIVI를 실행하기 위해 준비해야 하는 사항들과 구체적인 실행/종료방법에 대해 설명한다.

가. 환경변수 설정

GLOVE를 사용하기 전에 그림 10의 환경변수를 설정해야 한다. 환경변수 중 GLOVE_HOME과 GLOVE_BIN은 GLORE, GIVI에 공통적으로 해당되며, JDK_HOME부터 VPR_DEBUG_ENABLE에 이르는 8개의 환경변수는 GIVI가 필요로 한다.

```
# export GLOVE_HOME=/usr
# export GLOVE_BIN=${GLOVE_HOME}/bin
# export JDK_HOME=/usr/java/default
# export VJ_BASE_DIR=/usr/
# export VJ_DATA_DIR=/usr/share/vrjuggler-3.1.8/
# export JCCL_CFG_PATH=/usr/share/vrjuggler-3.1.8/data/configFiles
# export JCCL_BASE_DIR=/usr/share/jccl-1.5.1
# export TWEEK_BASE_DIR=/usr/share/tweek-1.5.1
# export VPR_DEBUG_NFY_LEVEL=0
# export VPR_DEBUG_ENABLE=0
```

그림 10. GLOVE 실행을 위한 환경변수 설정

GLORE와 GIVI 모두 클러스터 환경에서 실행되는 것을 전제하기 때문에 - 사용자 홈 디렉터리를 클러스터의 모든 노드가 공유하는 것을 가정하고 - 모든 환경변수를 ${HOME}/.bashrc에 적어두는 것이 편리하다. 만약 ${GLOVE_HOME}이 /usr이 아니라면 ${GLOVE_BIN} 역시 통상적인 PATH가 아닌 디렉터리를 가리킬 수도 있기 때문에 이 경우에는 그림 11과 같이 ${PATH}를 지정하던지, GLORE/GIVI를 실행할 때마다 절대경로를 줘야 한다.

```
# export PATH=$PATH:${GLOVE_BIN}
```

그림 11. GLOVE 실행을 위한 PATH 설정

나. 설정파일 디렉터리

kai.demo 홈 디렉터리에는 서로 다른 환경에서 GLOVE를 실행할 때 필요한 설정파일들을 상황에 맞게 나눠서 저장했다. 현재 visual 클러스터의 /home/kai.demo/.glove/ 디렉터리를 보면 config.vr, config.panel, config.window 디렉터리가 독립적으로 존재하고, config이라는 이름으로 이 세 디렉터리 중 하나를 가리키는 symbolic link가 만들어져 있다.

디렉터리	세부내용
config.vr	지하 1층 VR room에서 GLOVE를 실행할 때 필요한 설정파일
config.panel	5층 회의실에서 GLOVE를 실행할 때 필요한 설정파일
config.window	임의의 장소에서 윈도우 클라이언트를 실행할 때 필요한 설정파일

표 15. /home/kai.demo/.glove 디렉토리 내 GLOVE 설정 디렉터리 구성

사용자가 GLOVE를 특정 환경에서 실행하려면 그에 맞춰서 symbolic link를 수정해야 한다. 만약 5층 회의실에서 GLOVE를 실행하려고 하면 실행하는 곳의 위치가 바뀔 때마다 그림 12와 같이 symbolic link를 수정한다.

```
# rm config
# ln  -s  config.vr  config      (지하 1층에서 kai.demo 계정으로 GLOVE를 실행할 때)
# ln  -s  config.panel  config  (5층 회의실에서 kai.demo 계정으로 GLOVE를 실행할 때)
```

그림 12. symbolic link 수정을 통한 config 디렉터리 변경

다른 계정으로 GLOVE를 실행하고, 해당 계정으로 GLOVE를 실행할 때의 클라이언트가 확실히 정해져 있다면 ${HOME}/.glove/config 디렉터리를 만든 후에 kai.demo 계정의 설정파일 중 필요한 내용을 복사해서 사용하면 된다.

다. GLORE 실행

VR room이나 5층 회의실에서 GLOVE를 실행할 때에는 visual 클러스터의 gateway 노드에서 GLORE를 먼저 실행한다.

```
# glorem
  또는
# run.sh
```

그림 13. 슬레이브 노드에서 GIVI의 실행

그림 13의 두 명령 모두 GLORE를 실행한다는 점은 동일하지만 run.sh를 실행할 때에는 기존에 실행하던 GLORE 프로세스를 종료한 후 새로 실행하는 차이가 있다. 일단 GLORE가 실행되면 gateway에서는 glorem, visual의 다른 계산노드에서는 glorew가 각각 실행된다.

라. GIVI 실행

GIVI를 실행하려면 몇 가지 준비 작업이 필요하다. 여기서는 E2나 KVM의 모드 전환보다는 GIVI를 실행할 때 실제로 필요한 작업 위주로 설명한다.

1) 입력장치 준비

ART를 사용하는 VR room과 5층 회의실에서는 입력장치의 calibration이 필요하다. Calibration을 매번 실행할 필요는 없지만 입력장치가 사용자의 의도와 다르게 작동한다고 느껴질 때에는 바로 calibration을 실행해서 최적의 상태를 유지해야 한다. 하지만 구체적인 calibration 방법은 이 문서의 범위를 벗어나기 때문에 따로 설명하지 않겠다.

한 가지 유의해야 할 사항은 VR room에서 MS-Window와 리눅스를 혼용하기 때문에 발생하는 문제가 있는데, GLOVE(리눅스 기반)를 실행할 때에는 설정 프로그램(DTrack)에서 FlyStick과 헤드 트래커를 제외한 모든 마커를 disable 시켜야 VR Juggler가 정상적으로 작동한다.

2) NVIDIA framelock 설정

NVIDIA framelock은 NVIDIA GPU를 장착한 컴퓨터 사이의 프레임 동기화를 담당한다. 현재 VR room에는 두 대의 프로젝터가 독립적으로 입체영상(left, right)을 출력하기 때문에 두 프로젝터의 영상 출력시간을 맞추기 위해 framelock을 이용한다. Framelock 설정을 위해 visual_sl에서 nvidia-settings를 실행하고, 실행화면의 왼쪽 리스트에서 'Frame Lock'을 선택하면 그림 14과 같은 화면을 볼 수 있다. VR room은 visual_sl을 framelock 서버, visual_sr을 framelock 클라이언트로 설정한다.

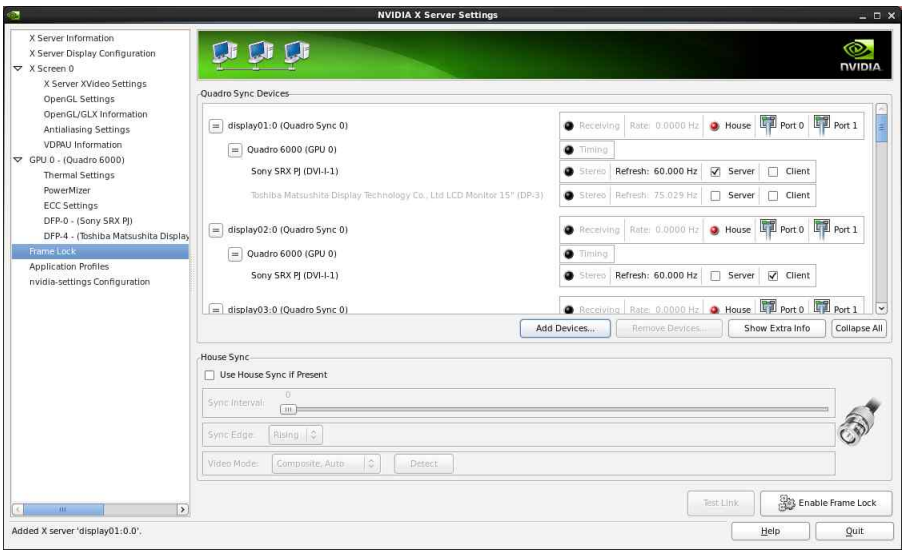

그림 14. nvidia-settings에서 Frame Lock 메뉴를 선택했을 때의 화면

여기서 동기를 맞춰야 하는 호스트와 스크린 번호를 추가한다. ‘Add Devices..’ 버튼을 누르면 그림 15의 다이얼로그 박스가 나오는데, 여기서 각각 visual_sl:0.0과 visual_sr:0.0을 추가한다. 여기서 visual_sl과 visual_sr은 각각 호스트명이고, :0.0은 X-Windows가 부여하는 스크린 번호를 의미한다.

그림 15. X 스크린 추가를 위한 다이얼로그 박스

두 대의 호스트를 모두 추가한 후에는 체크박스에서 visual_sl을 framelock 서버, visual_sr을 framelock 클라이언트로 지정한다. 마지막으로 ‘Enable Frame Lock’ 버튼을 눌러서 framelock을 활성화한다. 이 때 컴퓨터 뒷면을 보면 framelock LED가 점등하는 것을 볼 수 있다.

3) GIVI slave

GIVI의 실행 모드는 standalone, master, slave의 세 가지로 나눠진다. 이 중 standalone 모드는 5층 회의실에서 한 대의 컴퓨터로 GIVI를 실행할 때 필요하고, VR room에서 실행할 때에는 visual_al과 visual_sr에서는 slave 모드, visual_m에서는 master 모드로 실행한다.

```
# givi  --vrjslave
```

그림 16. 슬레이브 노드에서 GIVI의 실행

만약 슬레이브 노드가 여러 대라면 pdsh같은 명령을 이용해서 동시에 실행할 수도 있다. pdsh를 실행할 때에는 visual_sl과 visual_sr에 일일이 로그인 할 필요가 없으며, visual_m에서 바로 실행하면 된다.

```
# pdsh  -R  ssh  -w  visual_sl,visual_sr  ‘givi  --vrjslave’
```

그림 17. 여러 대의 슬레이브 노드에서 동시에 GIVI를 실행하는 방법

두 노드에서 givi가 제대로 실행이 됐다면, giviGipServer도 같이 실행되고 있는 것을 확인할 수 있다.

```
# ps aux | grep giviGipServer  (단일 슬레이브거나 특정 노드에서만 확인할 경우)
# pdsh -R ssh -w visual_sl,visual_sr 'ps aux | grep giviGipServer'  (여러 노드에 대해 동시 확인)
```

그림 18. givi의 정상적인 실행여부 확인

4) GIVI master

각 슬레이브 노드에서 GIVI 실행이 정상적으로 되는 것을 확인한 후에 visual_m에서 givi를 마스터 모드로 실행한다.

```
# givi  ~/.glove/config/kai.active.jconf  --vrjmaster
```

그림 19. givi의 마스터 모드 실행

GIVI를 마스터 모드로 실행할 때에는 VR Juggler 설정파일(그림 19에서 첫 번째 파라미터)을 추가로 기술해줘야 한다.

5) GIVI standalone

5층 회의실에서 실행할 때에는 GIVI가 단 한 대의 컴퓨터에서 실행되기 때문에 마스터와 슬레이브를 별도로 구분할 필요가 없다. 이 경우에는 그림 20와 같이 실행하면 된다.

```
# givi  ~/.glove/config/kai.active.jconf
```

그림 20. givi의 standalone 모드 실행

마. GIVI 종료

GIVI의 종료방법은 크게 세 가지로 볼 수 있다.

① 가상현실 환경의 다이얼 메뉴에서 'Quit'을 선택해서 종료하는 방법(그림 21)

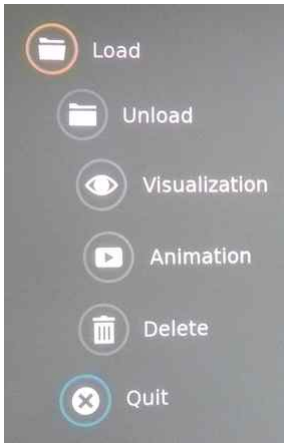

그림 21. Quit 메뉴

② visual_m에 떠있는 콘솔 윈도우에서 ESC를 치는 방법

③ 터미널에서 명령을 직접 실행하는 방법 : 앞에서 설명한 두 가지 방법으로 GIVI를 종료했다고 해도 IPC garbage(공유메모리, 세마포어 등)가 남아있을 수 있기 때문에 그림 22의 명령을 두세 번 반복 실행해주는 것이 좋다.

```
# pdsh -R ssh -w visual_m,visual_sl,visual_sr "killall givi ; killall giviGipServer ; killall giviGipClient ; delipc.sh"
```

그림 22. pdsh를 이용한 GIVI 실행 종료

바. GLORE 종료

GLORE의 종료는 그림 23과 같이 한다.

```
# killall.sh
```

그림 23. GLORE의 실행 종료

6. 유의사항

이 장에서는 외부 사이트에 대한 기술지원 업무를 수행하면서 발생한 기술적 문제와 해결방법에 대해 설명한다.

가. 리눅스, X-Windows, Barco E2

리눅스의 그래픽 카드 드라이버와 X-Windows, Barco E2 장비 사이의 궁합이 맞지 않아서 액티브 스테레오 출력에 한계가 있다.

E2는 기본적으로 '대형 캔버스에 여러 개의 비디오 입력을 자유롭게 배치해서 출력하는' 솔루션이라고 생각하면 크게 틀리지 않다. 다만, 입력 해상도가 최대 2048×2160 수준으로 제한을 받기 때문에 이를 고려해서 X-Windows를 설정해야 한다. 문제는 nvidia-settings[2]와 X-Windows 설정파일[3]에서 해상도나 디스플레이 구성방법을 변경해도 실제로 X-Windows를 실행해보면 사용자가 지정한 해상도가 아닌, E2가 보내주는 EDID[4]에 맞춰서 해상도와 화면구성이 임의로 바뀐다는 점이다. 게다가 KAI에 설치되어 있는 E2와 가장 잘 맞는 2048×2160 해상도는 nvidia-settings로는 지정이 불가능하다는 문제도 있다. 따라서 사용자는 디스플레이를 설정할 때 우선 E2로 해상도를 강제지정하고, 그 내용을 xorg.conf에 저장한 후 추가설정을 해야 혼란을 최소로 줄일 수 있다. 설정과정은 다음과 같다.

제일 먼저 할 일은 E2로 해상도를 2048×2160으로 강제 지정하는 것이다. 이는 E2 엔지니어가 직접 해줄 수 있다. 그렇게 한 후에는 편의를 위해 xorg.conf에 해상도를 2048×2160으로 적어둔다. 그 다음에는 NVIDIA mosaic 설정을 해야 하는데, 이는 xorg.conf의 Screen section에서 nvidia-XineramaInfoOrder를 설정하는 것으로 가능하다. 설정 세부내용은 visual_sl이나 visual_sr의 /etc/X11/xorg.conf에서 직접 확인할 수 있다. nvidiaXineramaInfoOrder를 설정한 후에는 두 개의 비디오 출력을 하나의 데스크톱처럼 사용할 수 있게 된다.

그림 24. 프로젝터 한 대가 출력하는 화면 구성. 두 개의 1920×2160 출력을 하나의 3840×2160 출력으로 조합한다.

2) NVIDIA GPU를 사용하는 X-Windows 용 디스플레이 설정 프로그램
3) /etc/X11/xorg.conf
4) Extended Display Identification Data

하지만 이 상태에서 OpenGL 어플리케이션을 전체화면 모드로 실행하면 프로젝터가 출력하는 화면의 절반만 사용하는 문제가 발생한다(그림 25).

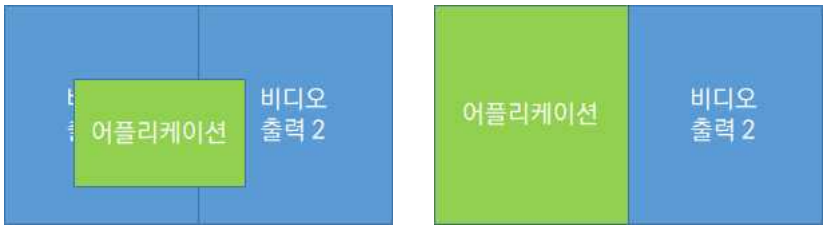

그림 25. nvidiaXineramaInfoOrder만 설정했을 때 fullscreen 모드
(좌: maximize 전, 우: maximize 후)

이를 해결하려면 nvidiaXineramaInfoOrder뿐만 아니라 원래의 X-Windows에서의 Xinerama 설정도 같이 수정해줘야 한다(그림 26).

```
Section  "Screen"
    중략
    Option  "Xinerama" "True"
    중략
EndSection
```

그림 26. xorg.conf에서 Xinerama 설정

nvidiaXineramaInfoOrder와 Xinerama를 모두 지정하면 OpenGL 어플리케이션을 전체화면 모드로 실행할 때 그림 27와 같은 형태의 출력이 가능하게 된다.

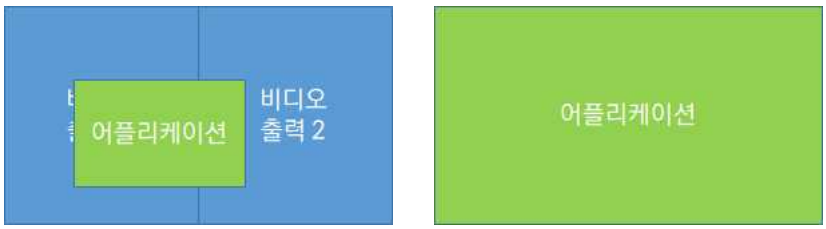

그림 27. Xinerama 설정을 포함시켰을 때 fullscreen 모드
(좌: maximize 전, 우: maximize 후)

이 방법의 단점은 액티브 스테레오 출력을 할 때 화면 출력속도가 24fps로 고정된다는 사실이다. 이에 대한 근본적인 해결방법은 E2를 사용하지 않고 해상도를 3840(또는 4096)×2160으로 고정한 후 패시브 스테레오 출력으로 변경하는 것이지만, 현재 사용하는 MS-Windows 어플리케이션과의 혼용이 불가능하게 된다.

나. X-Windows의 불안정성

앞에서 설명한, NVIDIA 디바이스 드라이버와 X-Windows, Barco E2 사이의 원활하지 않은 연계 때문에 X-Windows 자체가 불안해지는 단점이 존재한다. 심한 경우에는 GLOVE나 다른 OpenGL을 사용하는 어플리케이션을 실행한 후, gnome-terminal을 새로 띄울 때 X-Window가 통째로 죽는 현상을 볼 수 있다. 이 현상은 프로젝터와 직접 연결된 노드(visual_sl, visual_sr)에서 나타날 수 있다. 아직 이 문제에 대한 해결책은 찾지 못한 상태이므로, 문제가 발생하면 X-Windows를 다시 시작하는 수밖에 없다.

X-Windows의 재시작 방법은 우선 Ctrl + Alt + F2/F3/F4 ...를 눌러서 텍스트 콘솔로 화면을 전환한다. 그 상태에서 root 계정으로 로그인 한 후, 그림 28과 같이 연속적으로 runlevel을 변경하면 display 계정으로 자동 로그인이 된 상태의 X-Windows를 볼 수 있다.

```
# init 3        (X-Windows 종료)
# init 5        (X-Windows 시작)
```

그림 28. runlevel 변경

X-Windows가 정상적으로 실행되면 터미널을 띄우고 그림 29과 같이 xhost를 실행해서 임의의 사용자가 자유롭게 해당 호스트에서 X-Windows 프로그램을 실행할 수 있도록 보안수준을 낮춘다.

```
# xhost +
```

그림 29. xhost를 이용한 접근 제어 해제

다. ART FlyStick2의 조작 특징

ART FlyStick2는 6자유도를 갖는 가상현실 입력장치로, 여러 대의 카메라로 위치와 방향을 측정하고, 그 결과를 어플리케이션에게 알려준다. 주위 조명에 따라서 약간의 영향을 받는 경우도 있지만 대부분의 경우에는 충분히 안정적이고 정확한 동작을 보장하기 때문에 많은 사용자층을 확보한 제품이다. 다만 버튼과 조이스틱의 조작감이 최신 입력장치에 비해 많이 떨어지기 때문에 이른바 '또박또박' 눌러준다는 느낌으로 조작을 해야 정확한 입력이 가능하다.

7. 참고자료

```
<?xml version="1.0"?>

<glove>

<glore>
   <!-- hostname (or IP address) of glore master -->
   <master>10.1.12.1</master>

   <!-- for desktop client -->
   <sync>
      <ip>10.1.12.1</ip>
      <port>10007</port>
      <max-connection>100</max-connection>  <!-- max # of clients allowed -->
   </sync>

   <!-- for VR -->
   <async>
      <ip>10.1.12.1</ip>
      <port>10008</port>
      <max-connection>100</max-connection>  <!-- max # of clients allowed -->
   </async>

   <!-- for GDM -->
   <ga-max-memory>28</ga-max-memory>  <!-- max memory size per node (GB) -->

   <directSendFlag>true</directSendFlag>
</glore>

<givi>
   <!-- mode can be either 'single' or 'multi' -->
   <mode>multi</mode>

   <!-- full path to VRJ config -->
   <vrj>/home/gibeom.gu/Build/glove/config/picasso.jconf</vrj>

   <!-- may be omitted if we can get host information from the VRJ config -->
   <gip-server>
      <external>
        <host-name>master01</host-name>
        <!--
        <bind-ip>172.16.12.1</bind-ip>
        <port>10002</port>
        -->
        <bind-ip>10.1.12.1</bind-ip>
        <port>10002</port>
      </external>                          <!-- hostname (or IP address) -->
```

```
    <internal>
        <master>
            <host-name>master01</host-name>
            <bind-ip>10.1.12.1</bind-ip>
            <multicast>172.16.12.1</multicast>
        </master>
        <port>10002</port>
        <slave-network>ib</slave-network>
        <slave-count>16</slave-count>     <!-- hostname (or IP address) -->
        <slaves>
            <slave>
              <host-name>display01</host-name>
              <bind-ip>10.1.11.1</bind-ip>
              <multicast>172.16.11.1</multicast>
            </slave>
            <slave>
              <host-name>display02</host-name>
              <bind-ip>10.1.11.2</bind-ip>
              <multicast>172.16.11.2</multicast>
            </slave>
```

그림 30. glove.conf의 예