가상현실 가시화 시스템을 위한 위젯 라이브러리 설계 및 구현

2014. 11

한국과학기술정보연구원

목 차

1. 서론 ····· **1**

2. 위젯 라이브러리 개요 ····· **2**

가. 위젯 라이브러리의 구조 ····· 2

3. 이벤트 핸들링 ····· **4**

가. VR Juggler 기반 어플리케이션의 실행패턴 ····· 4

나. 이벤트 드리븐 어플리케이션의 실행패턴 ····· 5

다. 입력장치 상태정보의 전달 ····· 6

라. 위젯 라이브러리의 이벤트 ····· 8

마. 이벤트 변환 및 활용 ····· 11

1) 입력장치 정보의 직접 활용 ····· 11

2) 위젯 라이브러리용 이벤트로의 변환 ····· 12

4. 어플리케이션의 인터페이스 구성 ····· **13**

5. 위젯 라이브러리의 내부 구조 ····· **15**

가. 개요 ····· 15

나. Abstract class ····· 16

1) Abstract application ····· 16

2) Abstract button ····· 17

3) Abstract dialog ····· 18

4) Object / Widget ····· 20

다. 메인 어플리케이션 인터페이스 ····· 21

1) Application ····· 21

라. 일반 위젯 ····· 22

1) Check button ····· 22

2) Cubic dialog ····· 22

3) Drop list button ····· 22

4) Drop list ····· 22
5) Drop list item ····· 23
6) Radio button ····· 23
7) Rect button ····· 23
마. 메뉴 위젯 ····· 23
1) Menu item ····· 23
2) Dial button ····· 24
3) Dial menu ····· 24
4) Dialog frame ····· 25
5) Dial panel ····· 25
바. Composite 위젯 ····· 26
1) Legend list button ····· 26
2) Legend list ····· 26
3) Legend list item ····· 26
4) List frame ····· 26
5) List item ····· 27
6) Keypad ····· 27
7) Keypad dialog ····· 27
8) Notification message box ····· 28
9) Select list frame ····· 28
10) Select list item ····· 28
11) Select message box ····· 28

표 차례

[표 3-1] 위젯 라이브러리에 전달되는 입력장치 정보 ····· 7
[표 3-2] 위젯 라이브러리 내에서 만들어지는 이벤트의 종류 ····· 9
[표 3-3] 위젯 라이브러리가 사용하는 액션 ····· 10

[표 5-1] 위젯 라이브러리가 제공하는 위젯 목록 ····· 15
[표 5-2] abstract class의 구성 ····· 16
[표 5-3] vrAbstractButton의 슬롯 ····· 17
[표 5-4] vrAbstractButton의 속성을 지정하는 멤버 함수 ····· 17
[표 5-5] vrAbstractButton에서 만들어내는 이벤트 시그널 ····· 18
[표 5-6] vrWidget 클래스의 이벤트 처리함수 ····· 21
[표 5-7] vrApplication 클래스의 이벤트 ····· 21
[표 5-8] Cubic dialog의 이벤트 핸들러 ····· 22
[표 5-9] Drop list button의 이벤트 핸들러 ····· 22
[표 5-10] Drop list의 이벤트 핸들러 ····· 22
[표 5-11] Drop list의 이벤트 ····· 22
[표 5-12] Drop list item의 이벤트 핸들러 ····· 23
[표 5-13] Rect button의 이벤트 핸들러 ····· 23
[표 5-14] Dial button의 이벤트 핸들러 ····· 24
[표 5-15] Dial menu의 이벤트 핸들러 ····· 24
[표 5-16] Dialog frame의 이벤트 핸들러 ····· 25
[표 5-17] Dialog frame의 이벤트 ····· 25
[표 5-18] Dial panel의 이벤트 핸들러 ····· 25
[표 5-19] Dial panel의 이벤트 ····· 25
[표 5-20] Legend list button의 이벤트 핸들러 ····· 26
[표 5-21] Legend list의 이벤트 핸들러 ····· 26
[표 5-22] Legend list item의 이벤트 핸들러 ····· 26
[표 5-23] List frame의 이벤트 핸들러 ····· 26
[표 5-24] List item의 이벤트 핸들러 ····· 27
[표 5-25] Keypad의 이벤트 핸들러 ····· 27
[표 5-26] Keypad dialog의 이벤트 핸들러 ····· 27
[표 5-27] Notification message box의 이벤트 핸들러 ····· 28

[표 5-28] Select list frame의 이벤트 핸들러 ····· 28
[표 5-29] Select list item의 이벤트 핸들러 ····· 28
[표 5-30] Select message box의 이벤트 핸들러 ····· 28

그림 차례

[그림 1-1] GUI 기반 어플리케이션의 위젯 (왼쪽부터 Qt, wxWidgets, MFC) ····· 1
[그림 1-2] 위젯 라이브러리를 이용한 VR Juggler 기반 어플리케이션 ····· 1

[그림 2-1] 위젯 라이브러리의 계층 구조 ····· 2
[그림 2-2] 위젯 클래스의 hierarchy ····· 3

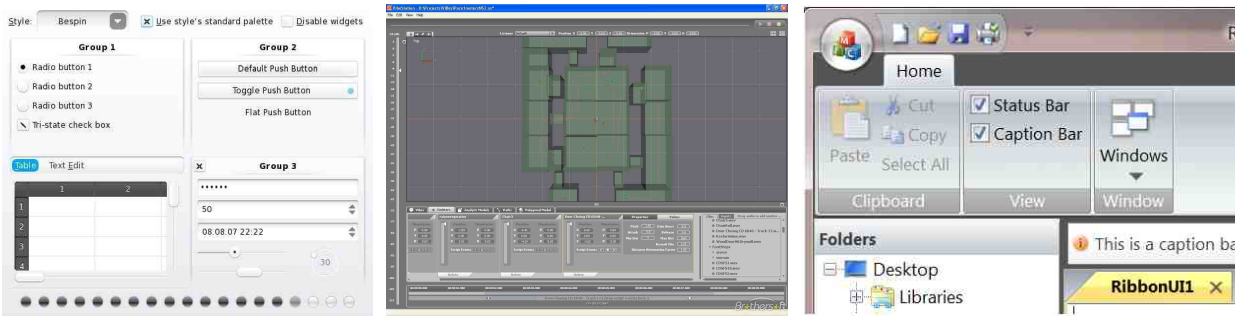
[그림 3-1] VR Juggler 기반 어플리케이션의 실행 흐름 ····· 4
[그림 3-2] VR Juggler의 이벤트 처리 과정 ····· 5
[그림 3-3] 이벤트 드리븐 방식의 이벤트 처리 과정 ····· 6
[그림 3-4] VR Juggler 기반 어플리케이션에서 위젯 라이브러리로의 정보 전달 ····· 7
[그림 3-5] 위젯 라이브러리가 유지하는 입력장치 정보 ····· 8
[그림 3-6] 어플리케이션이 직접 입력장치 정보를 사용하는 방법의 예 ····· 11
[그림 3-7] 위젯 라이브러리의 이벤트 처리 방식 ····· 12

[그림 4-1] 어플리케이션의 기본적인 화면구성 ····· 13
[그림 4-2] 주 다이얼 메뉴의 하위메뉴를 출력한 모습 ····· 13
[그림 4-3] 어플리케이션의 상태에 따라서 바뀌는 보조 다이얼 메뉴 ····· 14

[그림 5-1] 입력장치의 방향에 따라서 이벤트를 발생시키는 루틴 ····· 16
[그림 5-2] 입력장치의 방향과 각도에 따라서 특정 다이얼 메뉴를 활성화시키는 루틴 ····· 17
[그림 5-3] vrAbstractDialog의 독립 쓰레드 ····· 18
[그림 5-4] vrAbstractDialog의 실행 ····· 19
[그림 5-5] vrObject 클래스에 구현되어 있는 이벤트 처리 루틴 ····· 20
[그림 5-6] 메뉴 아이템의 구성 ····· 24

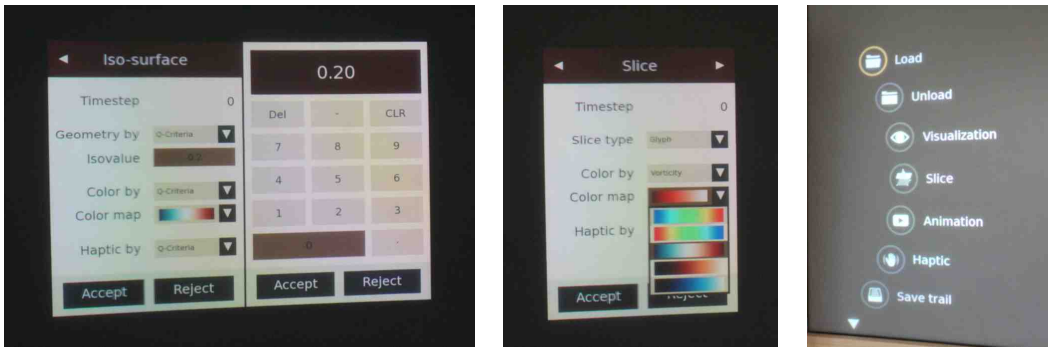
1. 서론

일반적인 GUI 기반 어플리케이션은 Qt, MFC, wxWidgets 등 인터페이스를 쉽게 구현할 수 있도록 도와주는 일련의 위젯 셋을 갖추고 있다[그림 1-1].



[그림 1-1] GUI 기반 어플리케이션의 위젯 (왼쪽부터 Qt, wxWidgets, MFC)

하지만 VR 어플리케이션 개발을 위한 미들웨어로 사용되는 VR Juggler는 기본적인 위젯의 지원이 매우 취약하기 때문에 사용자가 화면 인터페이스의 모든 부분을 일일이 구현해야 하는 단점을 갖고 있다. 뿐만 아니라 앞에서 언급한 위젯 셋은 키보드/마우스 등 주변장치로부터 발생한 이벤트를 일관된 방법으로 관리하고, 어플리케이션은 필요한 이벤트에 대한 핸들러 루틴만 구현하면 되지만 VR Juggler 기반 어플리케이션은 추상화된 이벤트와 그에 대한 처리방법론도 제공하지 않는다.



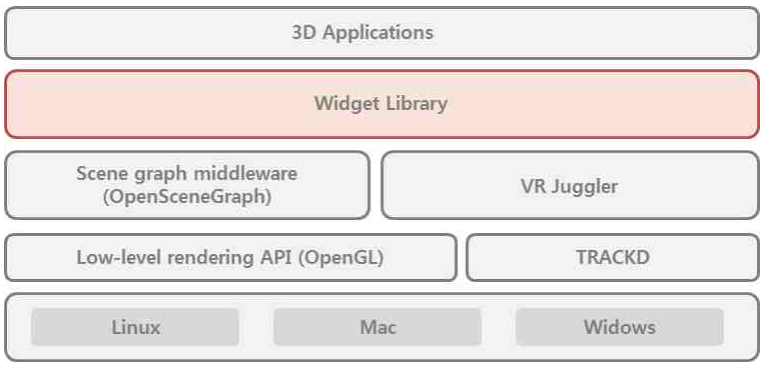
[그림 1-2] 위젯 라이브러리를 이용한 VR Juggler 기반 어플리케이션

이 문서는 VR Juggler의 단점을 보완하고 어플리케이션 인터페이스 개발을 쉽게 하기 위한 위젯 라이브러리의 구성에 대해 설명한다.

2. 위젯 라이브러리 개요

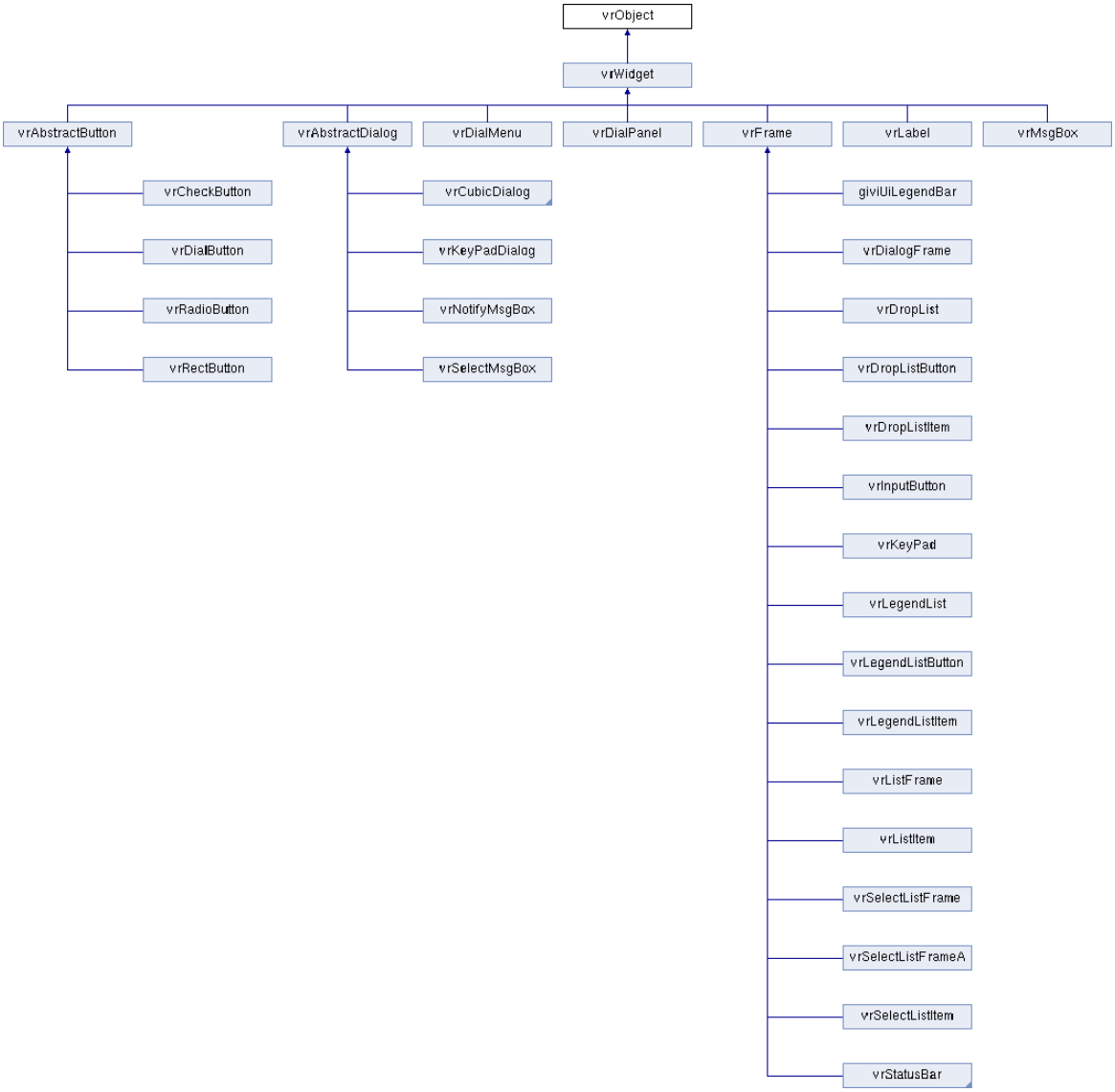
가. 위젯 라이브러리의 구조

가상현실 가시화 어플리케이션을 위한 위젯 라이브러리는 [그림 2-1]과 같은 계층구조를 갖는다. 위젯 라이브러리는 플랫폼에 독립적인 VR Juggler와 OpenSceneGrpah를 이용하기 때문에 다양한 종류의 운영체제를 지원한다는 장점이 있다. 그리고 VR Juggler가 다양한 입/출력 장치를 지원하기 때문에 trackd뿐만 아니라 Flystick, 데이터글로브 등 여러 형태의 장비를 활용할 수 있다는 장점이 있다.



[그림 2-1] 위젯 라이브러리의 계층 구조

하지만 위젯 라이브러리의 한계도 존재하는데, 기존의 MFC나 Qt와 같이 화면 입/출력부터 네트워크, 쓰레드 관리에 이르기까지 어플리케이션이 필요로 하는 모든 기능을 제공하지는 않는다는 점이다. 이 때문에 어플리케이션 개발자는 사용자 인터페이스 처리를 제외한 다른 기능의 구현은 직접 저수준의 API를 사용하거나 별도의 라이브러리를 활용해야 한다.
[그림 2-2]는 위젯 라이브러리를 구성하는 대표적인 위젯 클래스의 구성을 나타낸 것이다. 클래스 hierarchy의 최상단에는 vrObject가 있는데, 이는 위젯 라이브러리에 구현되어 있는 모든 클래스의 base class에 해당한다. vrObject로부터 파생된 vrWidget은 나머지 위젯들의 parent class이며, 이로부터 vrAbstractButton, vrAbstractDialog, vrDialMenu, vrDialPanel, vrFrame, vrLabel, vrMsgBox 등이 만들어진다. 어플리케이션 개발자는 필요에 따라서 해당 클래스를 이용해서 사용자 인터페이스를 구현하도록 구성되어 있다.



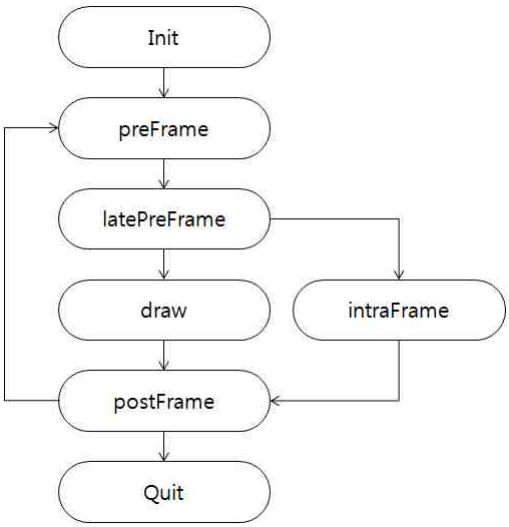
[그림 2-2] 위젯 클래스의 hierarchy

3. 이벤트 핸들링

인터페이스 구성과 사용자의 입/출력만을 지원하기 위한 위젯 라이브러리라면 기본적으로 이벤트 핸들링 방법의 구현이 핵심이 된다. 하지만 VR Juggler는 MFC나 Qt와는 다르게 per-frame operation만을 지원하고, 이벤트 드리븐 어플리케이션의 개발을 지원하기 위한 도구는 전혀 지원하지 않는다. 위젯 라이브러리는 상위 레벨 어플리케이션이 이벤트 드리븐 형식을 갖도록 하기 위해 일련의 이벤트 변환과정을 제공하는데, 이를 이해하려면 우선 VR Juggler와 기본의 이벤트 드리븐 어플리케이션의 작동방식의 차이를 이해해야 한다. 이 장에서는 두 접근방법의 차이를 설명하고, VR Juggler의 이벤트를 이벤트 드리븐 미들웨어가 어떻게 변환해서 사용하는지에 대해 설명한다.

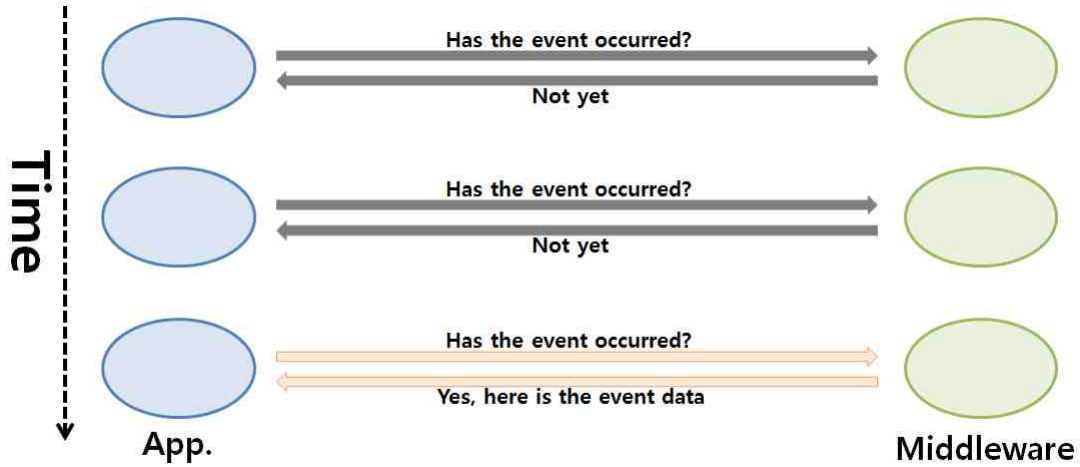
가. VR Juggler 기반 어플리케이션의 실행패턴

모든 VR Juggler 기반 어플리케이션은 기본적으로 프레임(frame) 단위로 실행된다. [그림 3-1]에서 볼 수 있듯이 처음 시작할 때(init)와 끝날 때(quit)를 제외하면 preFrame, latePreFrame, draw(필요한 경우 intraFrame), postFrame을 반복 실행하는데, 어플리케이션이 필요로 하는 기능은 모두 이 함수들 내에서 호출된다. 예를 들어서 OpenSceneGraph를 사용하는 어플리케이션이라면 preFrame이나 latePreFrame에서 scene graph를 수정하고 draw 함수에서 화면에 그림을 그리는 방식으로 실행된다.



[그림 3-1] VR Juggler 기반 어플리케이션의 실행 흐름

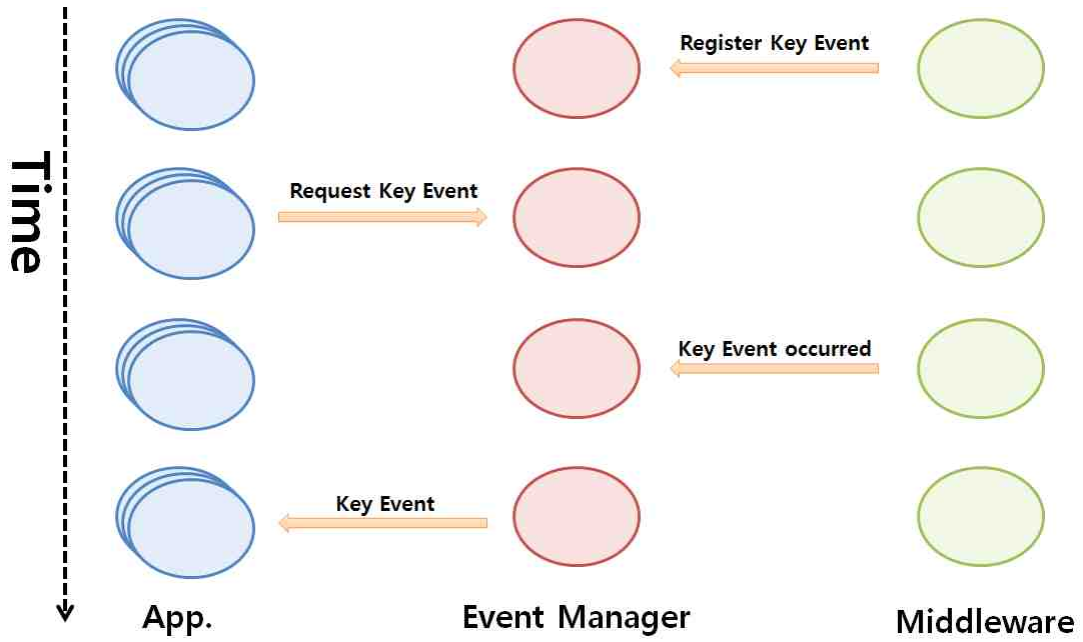
따라서 VR Juggler 기반 어플리케이션의 이벤트 처리는 [그림 3-2]와 같이 시간 순서대로 이루어진다. 즉, 미들웨어나 OS에서 발생하는 이벤트를 어플리케이션이 매번 확인해서 처리하는 형태로 구현해야 한다. 이러한 방식은 어플리케이션이 프레임마다 모든 종류의 이벤트 발생여부를 확인해야 하기 때문에 이벤트 드리븐 어플리케이션에 비해 효율적이지 않다.



[그림 3-2] VR Juggler의 이벤트 처리 과정

나. 이벤트 드리븐 어플리케이션의 실행패턴

앞에서도 언급했듯이 MS-Windows뿐만 아니라 Qt, wx 등 많은 위젯은 이벤트 드리븐(event-driven) 방식을 채용하고 있다. 이벤트 드리븐 방식은 다뤄야 할 이벤트에 대한 함수 포인터인 핸들러를 어플리케이션이 등록한 다음, 핸들러의 기반이 되는 미들웨어나 OS에서 이벤트가 발생했을 때 이를 호출하는 형태로 구현된다([그림 3-3] 참조).



[그림 3-3] 이벤트 드리븐 방식의 이벤트 처리 과정

이러한 방식은 어플리케이션이 일일이 특정 이벤트의 발생 여부를 알 필요가 없기 때문에 프로그램 개발이 용이해진다는 장점이 있다.

다. 입력장치 상태정보의 전달

일반적으로 가상현실 어플리케이션의 입력장치는 마우스/키보드보다는 완드, 헤드 트래커 등 특수한 장비를 사용하는 경우가 많다. VR Juggler는 이러한 장비들을 원활하게 사용할 수 있도록 일련의 디바이스 드라이버와 라이브러리를 제공하지만 각 디바이스로부터 전달받는 정보의 폭은 매우 제한되어 있다. 따라서 우리가 일반 위젯에서 볼 수 있는 고수준의 이벤트는(버튼 더블클릭, focus in/out, 스크롤 등) VR Juggler가 제공하는 기본적인 정보를 조합해서 위젯 라이브러리가 새로운 이벤트를 만들어줘야 한다. 이를 위해 VR Juggler를 사용하는 어플리케이션에서는 위젯 라이브러리에 입력장치의 정보를 지속적으로 전달해야 한다.

```
void
vrGIVI::preFrame(void)
{
   1. 완드의 현재 상태 업데이트
   2. 완드의 3차원 공간상의 위치 확인
   3. 완드의 3차원 공간상의 방향 확인
   4. 완드의 버튼 상태 확인
   5. 완드의 스틱 상태 확인

   if (위젯 제어 모드 활성화)
    {
      위젯 이벤트 처리 코어 함수인 vrApplication::updateDeviceStatus 함수를 이용하여
      완드 상태 전달
    }
}
```

[그림 3-4] VR Juggler 기반 어플리케이션에서 위젯 라이브러리로의 정보 전달

[그림 3-4]를 보면 preFrame 내에서 입력장치의 정보를 얻어와서 위젯 라이브러리에 전달하는 것을 확인할 수 있다. 이때 VR Juggler 어플리케이션이 위젯 라이브러리에 전달하는 기본정보는 [표 3-1]과 같이 정리할 수 있다. 여기서 입력장치의 위치는 실제 장비를 운용하는 물리적 공간상의 위치(m 또는 feet 단위로 측정)를 의미한다는 점에 주의해야 한다. 그리고 조이스틱 상태는 X축과 Y축 좌표를 각각 따로 측정하고, 무엇보다도 장비의 종류에 따라서 그 범위가 다르기 때문에 위젯 라이브러리에게 알려주기 전에 정규화(normalize)를 해야 한다.

구분		세부내용
입력장치 (완드 등)	위치	• 3차원 공간상의 좌표 (x, y, z)
	방향	• 입력장치의 방향을 나타내는 3차원 벡터 (x, y, z)
	버튼 상태	• 각 버튼에 대한 ON, OFF, TOGGLE_ON
	조이스틱 상태	• 조이스틱의 X축, Y축 방향 움직임 • 장비에 따라서 [0, 1] 또는 [-1, 1]의 값을 가질 수 있기 때문에 별도의 정규화 과정 필요
입력장치 (헤드 트래커)	추적하지 않음	

[표 3-1] 위젯 라이브러리에 전달되는 입력장치 정보

위젯 라이브러리는 VR Juggler로부터 전달받은 입력장치 정보를 [그림 3-5]의 구조체에 저장하고, 이를 자체 이벤트로 번역한다. GIVI는 어플리케이션 차원에서 단 세 개의 버튼을 사용하도록 구현되었기 때문에 mMenuButton, mSelectButton, mModeButton 정보를 유지하지만 필요한 경우 더 많은 버튼정보를 저장하도록 확장할 수 있다.

```
struct WandState
{
   // wand states
   gmtl::Point3f mPosition;
   gmtl::Vec3f   mDirection;

   // button states
   gadget::Digital::State mMenuButton;
   gadget::Digital::State mSelectButton;
   gadget::Digital::State mModeButton;

   // for stick state
   float mStickX;
   float mStickY;
};
```

[그림 3-5] 위젯 라이브러리가 유지하는 입력장치 정보

라. 위젯 라이브러리의 이벤트

VR Juggler 어플리케이션으로부터 전달받은 입력장치 정보는 vrApplication::translateEvent 함수를 통해 고유 이벤트로의 전환이 이루어진다. 그런데 이 과정에 만들어지는 이벤트는 주로 '입력장치의 이벤트'에 해당된다. 여기에 더해서 위젯 라이브러리가 자체적으로 만드는 이벤트가 별도로 존재하는데 여기에는 위젯의 생성/소멸, 포커스, 활성화, 패널전환, 버튼 선택, 스크롤 등이 포함된다. 입력장치 이벤트와 위젯 라이브러리가 자체적으로 만들어내는 이벤트는 [표 3-2]와 같이 정리할 수 있다. 실제로 구현된 위젯 라이브러리 내에서는 모든 이벤트 이름에 vrEVENT_ 라는 접두어가 붙어서 사용됐다.

구분		내용 및 용도
위젯 라이브러리 자체 발생	NULL	NULL
	CREATE	위젯 생성, 위젯 생성을 위한 메모리 할당 및 초기화
	DESTROY	위젯 소멸, 위젯 제거를 위한 메모리 해제
	PAINT	위젯 렌더링
	SIZE	위젯 크기 변경
	MOVE	위젯 위치 이동
	CLOSE	
	ACTIVATION	위젯 활성화
	VISIBILITY	위젯의 show/hide 변경
	FOCUS	위젯의 포커스 전환
	TIMER	타이머 완드의 중복 입력 처리 및 더블 클릭 이벤트 처리용
	UPDATE	위젯 상태 업데이트
	CUSTOM	사용자 지정 이벤트
	HOLD	포커스를 현재 위젯으로 강제 지정
입력장치 이벤트 번역	WAND	입력장치의 위치/방향
	STICK	입력장치의 조이스틱 입력
	BUTTON	입력장치의 버튼 입력

[표 3-2] 위젯 라이브러리 내에서 만들어지는 이벤트의 종류

[표 3-3]는 액션의 목록을 보여준다. 모든 액션 이름에는 vrACTION이라는 접두어가 붙어서 사용된다.

액션 이름	내용 및 용도
NULL	
ACTIVATE_ON	위젯 활성화를 위한 이벤트 액션,
ACTIVATE_OFF	위젯 비활성화를 위한 이벤트 액션
SHOW	위젯 Show를 위한 이벤트 액션
HIDE	위젯 Hide를 위한 이벤트 액션
FOCUS_IN	위젯 포커스 상태 처리를 위한 이벤트 액션
FOCUS_OUT	위젯 포커스 상태 처리를 위한 이벤트 액션
HOLD_ON	위젯 포커스 강제 지정을 위한 이벤트 액션
HOLD_OFF	위젯 포커스 강제 지정을 위한 이벤트 액션
RANGE	완드의 수직 움직임 이벤트 처리를 위한 액션 vrACTION_RANGE_0 ~ vrACTION_RANGE_8
HRANGE	완드의 수평 움직임 이벤트 처리를 위한 액션 vrACTION_HRANGE_0 ~ vrACTION_HRANGE_8
PANEL_LEFT	다이얼 메뉴 패얼의 포커스 전환을 위한 이벤트 액션
PANEL_BOTTOM	다이얼 메뉴 패얼의 포커스 전환을 위한 이벤트 액션
PANEL_RIGHT	다이얼 메뉴 패얼의 포커스 전환을 위한 이벤트 액션
PANEL_TOP	다이얼 메뉴 패얼의 포커스 전환을 위한 이벤트 액션
STICK_UP	완드의 stick 이벤트를 처리하기 위한 액션
STICK_DOWN	완드의 stick 이벤트를 처리하기 위한 액션
STICK_LEFT	완드의 stick 이벤트를 처리하기 위한 액션
STICK_RIGHT	완드의 stick 이벤트를 처리하기 위한 액션
BUTTON_SELECT_UP	완드의 선택 버트 이벤트를 처리하기 위한 액션
BUTTON_SELECT_DOWN	완드의 버튼 다운 이벤트를 처리하기 위한 액션

[표 3-3] 위젯 라이브러리가 사용하는 액션

마. 이벤트 변환 및 활용

VR Juggler와 위젯 라이브러리를 동시에 사용하는 어플리케이션은 입력장치의 상태정보를 두 가지 방법으로 활용할 수 있다. 하나는 상태정보를 위젯 라이브러리에게 전달하지 않고 직접 활용하는 것이고, 다른 하나는 위젯 라이브러리에게 정보를 보내는 것이다. 하지만 이 두 가지 방법을 동시에 적용하는 경우는 존재하지 않는 것으로 가정한다.

1) 입력장치 정보의 직접 활용

어플리케이션이 입력장치 정보를 직접 활용하는 경우는 매 프레임 마다 하드웨어의 상태를 확인해야 한다. 이는 일반적인 VR Juggler 기반 어플리케이션이 취하는 방법과 동일하고, preFrame 함수 내에서 입력장치의 위치, 방향, 버튼과 스틱의 상태 등을 확인한 후 그에 따라서 어플리케이션의 상태를 변경한다.

```
vrGIVI::vrGIVI(vrj::Kernel *kernel, int &argc, char **argv) : vrj::osg::App(kernel)
{
    mWand.init(std::string("VJWand"));
    mHead.init(std::string("VJHead"));
    mStickX.init(std::string("VJXStick"));
    mStickY.init(std::string("VJYStick"));
    mButton0.init(std::string("VJButton0"));
    mButton1.init(std::string("VJButton1"));
}

void
vrGIVI::preFrame(void)
{
   // button event
   gadget::Digital::State mMenuButton   = mButton0->getData();
   gadget::Digital::State mSelectButton = mButton1->getData();
   if (mMenuButton == gadget::Digital::TOGGLE_OFF) {
       // proc. button event
    }

    // 입력장치의 위치와 방향 확인
gmtl::Matrix44f wandMatrix = mWand->getData();
gmtl::Vec3f wandDir;

    mPosition = gmtl::makeTrans<gmtl::Point3f>(wandMatrix); ///< wand position
    gmtl::xform(wandDir, wandMatrix, unitLaser);
    mDirection = gmtl::normalize(wandDir;                   ///< wand direction

    if (mStickX->getData() > 0.5 ) {
       // 조이스틱의 x 방향 움직임에 대한 반응 처리
    }
}
```

[그림 3-6] 어플리케이션이 직접 입력장치 정보를 사용하는 방법의 예

2) 위젯 라이브러리용 이벤트로의 변환

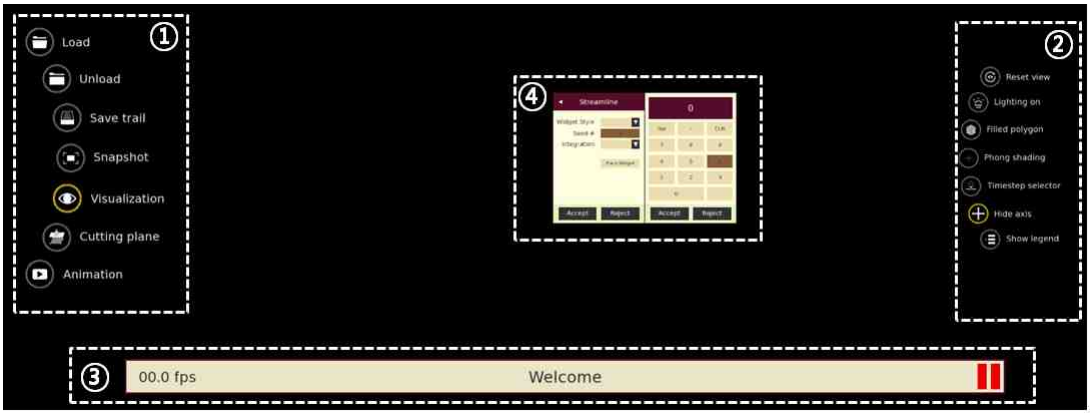
어플리케이션이 소진하지 않은 입력장치 정보는 vrApplication의 translateEvent함수로 전달돼서 위젯 라이브러리용 이벤트로 변환된다. translateEvent 함수에서는 타이머, 메뉴 조작, 위젯 조작 등의 이벤트로 변환한 다음 각 위젯으로 해당 이벤트를 전달한다.

```
void
vrApplication::translateEvent(WandState &state)
{
    gmtl::normalize(state.mDirection);

    this->eventUiTimerAction();          // 타이머 이벤트 처리
    this->eventUiMenuAction(state);      // 메뉴 액션 처리
    this->eventUiWidgetAction(state);    // 위젯 액션 처리
}

void
vrApplication::eventUiWidgetAction(WandState &state)
{
    // get a top widget pointer from the stack list
    vrWidget *topWidget = this->getTopWidget();
    if (topWidget == NULL) return

    // invoke a stick event
    eventWidgetStickAction(topWidget, state.mStickX, state.mStickY);

    // invoke a button click event
    eventWidgetButtonAction(topWidget, state);

    // invoke a range action event
    eventWidgetRangeAction(topWidget, state.mDirection);

    // invoke a wand action event
    eventWidgetWandAction(topWidget, state.mPosition, state.mDirection);
}

void
vrWidget::setAllowedEvent(bool button, bool stick, bool wand)
{
    mWidgetStatus.isButtonEventEnabled = button;
    mWidgetStatus.isStickEventEnabled = stick;
    mWidgetStatus.isWandEventEnabled = wand;
}
```

[그림 3-7] 위젯 라이브러리의 이벤트 처리 방식

4. 어플리케이션의 인터페이스 구성

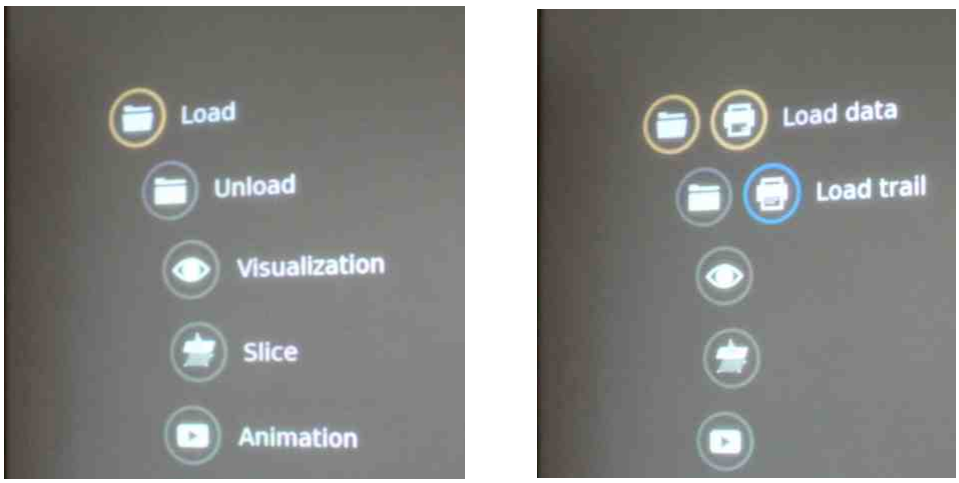
VR Juggler와 OpenSceneGraph, 그리고 위젯 라이브러리를 사용하는 어플리케이션은 [그림 4-1]과 같은 화면 구성을 갖는다.



[그림 4-1] 어플리케이션의 기본적인 화면구성

① 주 다이얼 메뉴

일반적인 GUI 어플리케이션의 주 메뉴와 동일한 역할을 담당한다. 각 메뉴항목에 대응하는 하위메뉴의 출력은 [그림 4-2]와 같이 메뉴항목의 텍스트가 사라진 상태에서 바로 옆에 하위메뉴를 보여주는 방식으로 구현했다.

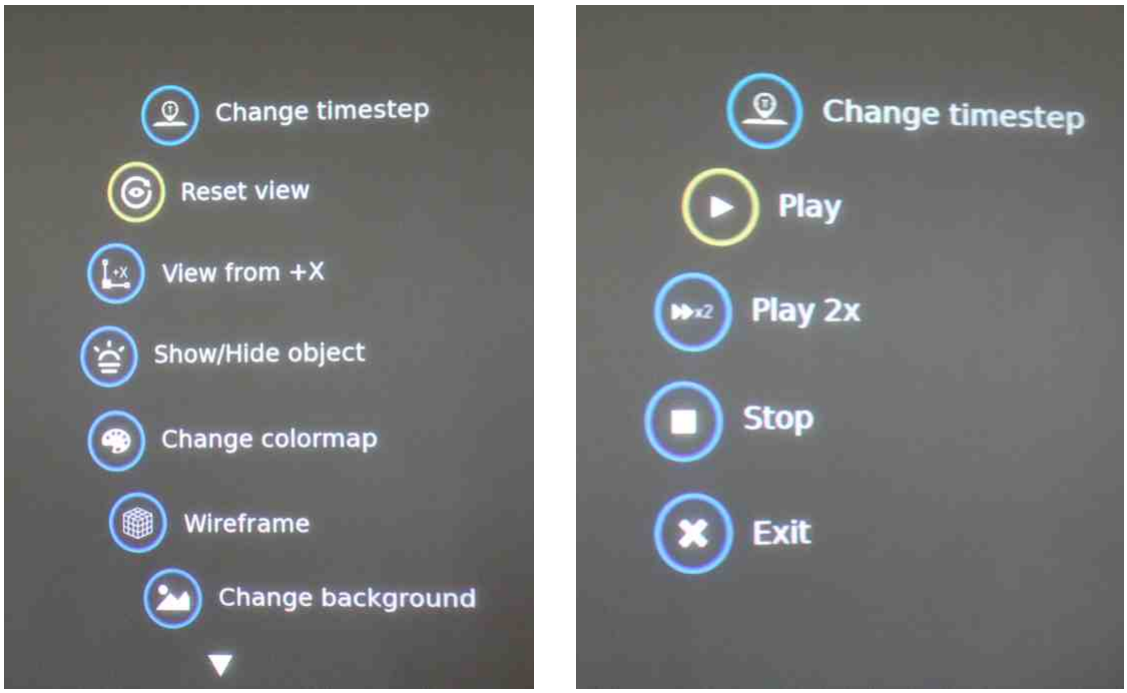


[그림 4-2] 주 다이얼 메뉴의 하위메뉴를 출력한 모습

주 다이얼 메뉴를 구현하는 데에는 Label, Dial button, Dial panel, Dial menu 등의 위젯을 사용했다.

② 보조 다이얼 메뉴

어플리케이션의 보조기능을 실행하기 위한 메뉴로, 위에서 설명한 주 다이얼 메뉴와 형태는 유사하지만 크기를 약간 작게 함으로써 주 메뉴보다는 시각적으로 덜 부각되도록 조절했다. 주 다이얼 메뉴와는 달리 보조 다이얼 메뉴는 하위메뉴가 존재하지 않으며, 그 대신 전체 보조 다이얼 메뉴의 내용이 바뀌도록 했다. 보조 다이얼 메뉴 역시 주 다이얼 메뉴와 마찬가지로 Label, Dial button, Dial panel, Dial menu 등의 위젯으로 구성되어 있다.



[그림 4-3] 어플리케이션의 상태에 따라서 바뀌는 보조 다이얼 메뉴

③ 상태표시 창

어플리케이션의 실행상태를 실시간으로 보여주기 위해 사용한다. 여기에는 어플리케이션이 출력하는 메시지나 fps 등 별도의 위젯을 사용하기에는 그 양이 적지만 사용자에게 반드시 알려야 하는 정보가 담긴다.

④ 다이얼로그

일반적인 GUI 기반 어플리케이션이 사용하는 다이얼로그와 동일하지만 VR 어플리케이션의 특성상 GUI 만큼 다양한 종류의 위젯을 갖추기 쉽지 않고, 사용자의 입력 역시 상대적으로 제약이 높은 만큼 다이얼로그에 의존해서 사용자의 입력을 처리하는 비중이 비교적 높다. 다이얼로그는 버튼, 레이블 등의 기본적인 위젯을 조합해서 새로운 형태의 위젯을 만들거나 keypad와 같이 이미 만들어진 위젯을 사용한다.

5. 위젯 라이브러리의 내부 구조

가. 개요

위젯 라이브러리가 사용하는 위젯 목록은 [표 5-1]과 같이 정리할 수 있다. 우선 위젯은 기본 위젯과 메뉴 위젯, composite 위젯으로 분리할 수 있다. 이 중 기본 위젯은 여러 개를 조합해서 다이얼로그 박스를 만들거나 다른 위젯의 구성요소로 사용된다. 메뉴 위젯은 메인 메뉴나 보조 메뉴를 만들기 위해서만 사용하며, 운영/구현상의 편의를 위해 따로 분리했다. 마지막으로 composit 위젯은 기본 위젯을 조합해서 이미 완성된 형태로 동작하는 위젯으로, 일반적인 다이얼로그 박스는 내용 구성과 위젯 배치가 자유롭지만 composite는 이미 모든 배치는 고정되어 있다는 차이가 있다.

구분	이름	개요
기본 위젯	Label	• 텍스트 출력
	Panel	• 서로 다른 위젯을 모아서 배치할 수 있는 바탕
	Push button	• 일반적으로 생각할 수 있는 버튼
	Toggle button	• 선택/해제를 교대로 지정할 수 있는 버튼
	Check button	• 선택/해제를 교대로 지정할 수 있는 버튼
	Dropdown button	• 아이템 리스트 중 하나를 선택 가능한 버튼
메뉴 위젯	Dial panel	• 다이얼 메뉴 그룹의 생성 및 관리을 생성하고 관리
	Menu item	• 메뉴 내에서 선택할 수 있는 최소단위
Composite	Keypad	• 숫자를 입력하기 위한 계산기 모양의 인터페이스
	Status bar	• 위젯의 상태나 시스템 메시지 출력
	Legend bar	• 색상 맵 정보 및 범위를 출력하기 위한 위젯
	Message box	• 어플리케이션 실행 도중 특정한 상황이 발생했을 때 이를 사용자에게 알려주기 위해 사용
	Item selector	• 여러 개의 선택대상이 있고, 그 중에서 일부항목을 선택하기 위한 인터페이스

[표 5-1] 위젯 라이브러리가 제공하는 위젯 목록

그리고 위의 위젯을 총괄관리하기 위해 vrAbstractApplication이라는 클래스가 존재한다. 이 클래스는 위젯 라이브러리를 사용하는 모든 어플리케이션의 parent class 역할을 담당하며,

입력장치의 상태정보를 번역하는 루틴을 포함하고 있다. 이 외에도 버튼, 다이얼로그 박스, 기본 위젯을 위한 abtract class가 각각 존재한다.

클래스 이름	내용
vrAbstractApplication	위젯 라이브러리를 사용하는 어플리케이션을 위한 base class
vrAbstractButton	위젯 라이브러리가 제공하는 모든 버튼 위젯의 base class
vrAbstractDialog	위젯 라이브러리가 제공하는 모든 다이얼로그 박스의 base class
vrObject	위젯 라이브러리가 제공하는 모든 객체의 base class
vrWidget	위젯 라이브러리가 제공하는 모든 위젯의 parent class vrObject를 상속한다.

[표 5-2] abstract class의 구성

나. Abstract class

1) Abstract application

```
VR_WAND_RANGE
vrAbstractApplication::angleToRangeH (float degree)
{
   // make sure 0 <= degree <= 180
   degree = osg::minimum(180.0f, osg::maximum(0.0f, degree));

   if (degree < 20.0f)       return VR_WAND_RANGE_0;
   else if (degree < 40.0f)  return VR_WAND_RANGE_1;
   else if (degree < 60.0f)  return VR_WAND_RANGE_2;
   else if (degree < 80.0f)  return VR_WAND_RANGE_3;
   else if (degree < 100.0f) return VR_WAND_RANGE_4;
   else if (degree < 120.0f) return VR_WAND_RANGE_5;
   else if (degree < 140.0f) return VR_WAND_RANGE_6;
   else if (degree < 160.0f) return VR_WAND_RANGE_7;

   return VR_WAND_RANGE_8;
}
```

[그림 5-1] 입력장치의 방향에 따라서 이벤트를 발생시키는 루틴

angleToRange 함수는 수평 각도와 수직 각도 각각에 대해 별도로 구현되어 있지만 유사한 형태로 구현되어 있기 때문에 여기서는 수평 각도를 이벤트로 변환하는 루틴만 제시한다. 그리고 다이얼 메뉴가 출력된 상태에서 사용자가 입력 장치를 왼쪽이나 오른쪽으로 향할

때 어떤 메뉴를 활성화 시킬지를 결정한다.

```
VR_PANEL_ID
vrAbstractApplication::angleToPanelID (float degree)
{
   // make sure 0 <= degree <= 180
   degree = osg::minimum(180.0f, osg::maximum(0.0f, degree));

   if (degree < 90.0) return VR_PANEL_ID_2;
   return VR_PANEL_ID_0;
}
```

[그림 5-2] 입력장치의 방향과 각도에 따라서 특정 다이얼 메뉴를 활성화시키는 루틴

2) Abstract button

vrAbstractButton은 Dial button, Push button 등 위젯 라이브러리가 제공하는 모든 버튼 위젯의 베이스 클래스다. 이 클래스에서는 버튼의 종류를 설정하기 위한 멤버 함수를 제공하며, 입력장치의 상태정보를 이용해서 버튼 클릭, 토글 등의 이벤트를 만들기도 한다.

슬롯 이름	내용
slotButtonStatusChanged	완드의 토글 버튼 및 버튼 이벤트 활성화 여부에 따라 버튼 이벤트 시그널 발생

[표 5-3] vrAbstractButton의 슬롯

멤버 함수 이름	내용
setCheckable	체크 버튼으로 사용할지 여부 설정
setTogglable	토글 버튼으로 사용할지 여부 설정
setEnable	버튼 이벤트를 활성화할 것인지 여부 설정

[표 5-4] vrAbstractButton의 속성을 지정하는 멤버 함수

시그널 이름	내용
clicked	버튼 클릭 상태 전달을 위한 이벤트 시그널 발생 함수
checked	버튼 체크 상태 전달을 위한 이벤트 시그널 발생 함수
toggled	버튼 토글 상태 전달을 위한 이벤트 시그널 발생 함수

[표 5-5] vrAbstractButton에서 만들어내는 이벤트 시그널

3) Abstract dialog

vrAbstractDialog는 어플리케이션이 자체적으로 사용하는 다이얼로그 박스나 일부 composite 위젯의 베이스 클래스다. 이 클래스의 특징은 어플리케이션과는 독립된 쓰레드를 사용하기 때문에 빠른 응답속도를 구현한다는 점이다. 따라서 non-preemptive event driven application과 같이 특정 이벤트 처리를 위해 전체 어플리케이션의 인터페이스 업데이트가 진행되지 않는 현상은 존재하지 않는다.

[그림 5-3]은 vrAbstractDialog 클래스의 독립 쓰레드를 보여주고 있다. 그 구조는 매우 간단해서 다이얼로그 박스에 대해 이벤트가 발생했는지의 여부를 확인하고, 이벤트가 발생했을 때 해당 이벤트를 처리하는 핸들러를 호출하는 방식으로 작동한다. 이 쓰레드는 해당 다이얼로그 박스가 제거될 때까지 계속 실행된다.

```
void
vrAbstractDialog::eventLoop (void)
{
bool ret = false;
vrEvent e;

   while (mIsEventThreadValid)
    {
       usleep(1000);

       // find button event
       ret = this->popEvent(e);

       // checking status & invoke an event
       if (ret) procEvent(e);
    }
}
```

[그림 5-3] vrAbstractDialog의 독립 쓰레드

앞에서 설명했듯이 모든 종류의 다이얼로그 박스는 vrAbstractDialog를 상속받는다. 새로운 다이얼로그 박스가 상속받는 함수 중에는 exec()도 있는데, 이 함수가 다이얼로그 박스를 실행하는 역할을 맡는다. 실제로 어플리케이션이 다이얼로그 박스를 구성해서 화면에 출력하고 실행하려면 vrAbstractDialog 클래스의 exec 함수를 호출해야 한다. [그림 5-4]

```
void
vrAbstractDialog::exec()
{
   if (!mIsModal) return
   else mIsEventThreadValid = true

   // remove dialog
   if (mParentApplication != NULL)
    {
       // register to list
       mParentApplication->insertDialog(this);
       mParentApplication->registerWidget(this);
    }

   mDialogEventThread->setFunctor(boost::bind(&vrAbstractDialog::eventLoop,
this));
   mDialogEventThread->start();
   mDialogEventThread->join();

   if (mParentApplication != NULL)
    {
       mParentApplication->unregisterWidget();
       mParentApplication->deleteDialog(this);
    }
}

void
foo ()
{
   중략

   dialog->exec();   // 어플리케이션의 다이얼로그 박스 실행 방법

   중략
}
```

[그림 5-4] vrAbstractDialog의 실행

4) Object / Widget

vrObject는 모든 위젯의 기본이 되는 클래스로 위젯의 생성과 소멸, 크기변경 등의 이벤트 처리와 이벤트 시그널을 관리한다.

```
bool
vrObject::event(vrEvent e)
{
   switch (e.type()) {
      case vrEVENT_CREATE:
         this->signalCreated();
         break

      case vrEVENT_DESTROY:
         this->signalDestroyed();
         break

      case vrEVENT_PAINT:
         this->signalPainted();
         break

      case vrEVENT_SIZE:
         this->signalSizeChnaged();
         break

      case vrEVENT_MOVE:
         this->signalMoved();
         break

      case vrEVENT_CLOSE:
         this->signalClosed();
         break
   }
   return true
}
```

[그림 5-5] vrObject 클래스에 구현되어 있는 이벤트 처리 루틴

위젯 라이브러리의 모든 위젯들은 vrWidget 클래스를 기본으로 상속받게 되어 있는데, vrWidget 클래스는 다시 vrObject 클래스를 상속받은 클래스이다. vrWidget은 위젯의 상태 및 버튼 등의 이벤트 처리를 담당한다. 메뉴, 패널, 버튼, 다이얼로그, 메시지 박스 등 대부분의 일반 위젯에서 이벤트 처리를 위해 vrWidget 클래스를 상속받도록 설계되어 있다.

이벤트 함수 이름	내용
eventSetActivate	위젯의 활성/비활성 상태 처리
eventSetEventHold	다이얼로그 등의 위젯에서 트리구조의 이벤트 처리
eventSetFocus	위젯의 포커스 상태를 처리
eventSetVisibility	위젯의 Show/Hide 상태 처리
eventUpdateButtonStatus	버튼의 상태 처리
eventUpdateStickStatus	스틱의 상태 처리
eventUpdateWandStatus	완드의 위치 변화 처리
eventUpdateWandDirection	완드의 방향 변화 처리

[표 5-6] vrWidget 클래스의 이벤트 처리함수

다. 메인 어플리케이션 인터페이스

1) Application

이벤트	내용
eventUiMenuAction	메인 메뉴와 보조 메뉴의 동작 처리
eventUiTimerAction	위젯의 시간 이벤트 처리
eventUiWidgetAction	위젯의 상태 및 동작 처리
eventUiPanelSelection	메인 메뉴와 보조 메뉴간의 포커스 전환
eventUiPanelFold	다단계 형태의 메뉴 패널을 제어

[표 5-7] vrApplication 클래스의 이벤트

라. 일반 위젯

1) Check button

- Derived from : vrAbstractButton

2) Cubic dialog

이벤트 핸들러	내용
slotUpdateButtonStatus	다이얼로그의 버튼 입력 등의 이벤트 처리
slotUpdateWandStatus	다이얼로그의 완드 방향 등의 이벤트 처리

[표 5-8] Cubic dialog의 이벤트 핸들러

3) Drop list button

이벤트 핸들러	내용
slotUpdateButtonStatus	완드의 버튼 이벤트 처리
slotUpdateFocusStatus	drop list 버튼의 포커스 상태 처리
slotUpdateActivationStatus	drop list 버튼의 활성화/비활성화 상태 처리
slotSelectedListIdx	drop list 버튼의 선택된 리스트 아이템의 인덱스 전달
slotSelectedListName	drop list 버튼의 선택된 리스트 아이템의 이름을 전달

[표 5-9] Drop list button의 이벤트 핸들러

4) Drop list

이벤트 핸들러	내용
slotUpdateWandStatus	dropdown 버튼의 리스트 아이템을 탐색하기 위한 완드 방향 처리

[표 5-10] Drop list의 이벤트 핸들러

이벤트	내용
eventListScrollUpSafety	drop list 아이템 탐색을 위한 스크롤 이벤트 처리
eventListScrollDownSafety	drop list 아이템 탐색을 위한 스크롤 이벤트 처리
eventListFocus	drop list 아이템의 포커스 변경을 위한 이벤트 처리

[표 5-11] Drop list의 이벤트

5) Drop list item

이벤트 핸들러	내용
slotUpdateFocusStatus	drop list 아이템의 포커스 상태 업데이트

[표 5-12] Drop list item의 이벤트 핸들러

6) Radio button

- erived from : vrAbstractButton

7) Rect button

이벤트 핸들러	내용
slotButtonClick	버튼의 클릭 이벤트 처리
slotButtonToggle	버튼의 토글 이벤트 처리
slotButtonCheck	버튼의 체크 이벤트 처리
slotUpdateActiveStatus	버튼의 활성/비활성 상태 업데이트
slotUpdateFocusStattus	버튼의 포서스 상태 업데이트 함수

[표 5-13] Rect button의 이벤트 핸들러

마. 메뉴 위젯

1) Menu item

다이얼 메뉴를 구성하는 메뉴 아이템의 속성 등은 구조체로 정의되어 있으며, vrMenuItem 클래스에 저장된 vrItemElement 들이 메인 및 서브 메뉴를 구성하는데 사용된다.

```
struct vrItemElement
{
   int  mSwitchingIdx;
   std::vector<std::string> mLabels;    // 아이템의 레이블
   std::vector<std::string> mIcons;     // 아이콘 이름 (파일명)

   std::string mParentKey;              // 아이템의 부모 키 값

   bool mElementEnabled;                // 아이템 활성화 여부

   vrElementSlotFun mElementSlot;       // 아이템 선택 시 실행할 함수 포인터
   vrApplication *mElementSlotParent;   // 아이템 선택 시 실행할 함수의 부모 클래스
};
```

[그림 5-6] 메뉴 아이템의 구성

2) Dial button

이벤트 핸들러	내용
slotUpdate	다이얼 버튼의 상태를 업데이트
slotShowLabel	다이얼 버튼의 레이블을 숨기거나 표시
slotButtonClick	다이얼 버튼의 클릭 이벤트 처리
slotUpdateActivationStatus	다이얼 버튼의 활성화/비활성화 상태 처리
slotUpdateFocusStatus	다이얼 버튼의 포커스 상태 처리

[표 5-14] Dial button의 이벤트 핸들러

3) Dial menu

이벤트 핸들러	내용
slotUpdateSubPanelContents	메인 메뉴의 기능 선택에 따른 보조 메뉴의 기능 전환
slotUpdateMainPanelContents	메인 메뉴의 다단계 메뉴 활성화를 위한 업데이트
slotUpdateMenuActivation	메인 메뉴 및 보조 메뉴의 활성화 여부 처리
slotUpdatePanelFocus	메인 메뉴 및 보조 메뉴의 포커스 여부 처리
slotUpdatePanelFold	다단계 형태의 메뉴를 펼치거나 접기 위한 함수
slotUpdateButtonStatus	완드의 버튼 상태를 이용하여 메뉴의 기능 선택
slotUpdateWandStatus	메뉴 탐색 기능을 위한 완드 상태 처리

[표 5-15] Dial menu의 이벤트 핸들러

4) Dialog frame

이벤트 핸들러	내용
slotUpdateWandStatus	다이얼로그 프레임 속 위젯 아이템의 탐색
slotUpdateButtonStatus	다이얼로그 프레임 속 위젯 아이템의 선택 이벤트 처리
slotUpdateFocusStatus	다이얼로그 프레임의 포커스 상태 처리
slotUpdateEventHoldStatus	다이얼로그 프레임 속 위젯들의 이벤트 분배

[표 5-16] Dialog frame의 이벤트 핸들러

이벤트	내용
eventFrameItemFocus	다이얼로그 프레임 속 위젯 아이템의 포커스 상태 변경

[표 5-17] Dialog frame의 이벤트

5) Dial panel

이벤트 핸들러	내용
slotUpdateContents	다이얼 메뉴를 구성하는 패널의 아이템 변경 및 업데이트
slotPanelFold	다이얼 패널을 펼치거나 접기 위한 이벤트 처리
slotUpdatePanelActivation	다이얼 패널의 활성화 상태 처리
slotUpdatePanelVisibility	다이얼 패널의 Show/Hide 상태 처리
slotUpdateButtonStatus	다이얼 패널의 버튼 상태 처리
slotUpdateWandStatus	다이얼 패널의 완드 상태 처리

[표 5-18] Dial panel의 이벤트 핸들러

이벤트	내용
eventScrollFocus	다이얼 패널을 구성하는 다이얼 아이템의 포커스 변경
eventScrollUp	다이얼 아이템을 탐색하기 위한 이벤트 처리
eventScrollDown	다이얼 아이템을 탐색하기 위한 이벤트 처리

[표 5-19] Dial panel의 이벤트

바. Composite 위젯

1) Legend list button

이벤트 핸들러	내용
slotUpdateActivationStatus	Legend list 버튼의 활성/비활성 상태 업데이트
slotUpdateButtonStatus	Legend list 버튼의 버튼 이벤트 처리
slotUpdateFocusStatus	Legend list 버튼의 포커스 상태 업데이트
slotSelecteedListIdx	Legend list 버튼의 선택된 리스트 아이템의 인덱스 반환

[표 5-20] Legend list button의 이벤트 핸들러

2) Legend list

이벤트 핸들러	내용
slotUpdateWandStatus	Legend list의 아이템 탐색을 위한 완드 방향 업데이트

[표 5-21] Legend list의 이벤트 핸들러

3) Legend list item

이벤트 핸들러	내용
slotUpdateFocusStatus	Legend list 아이템의 포커스 상태 업데이트

[표 5-22] Legend list item의 이벤트 핸들러

4) List frame

이벤트 핸들러	내용
slotUpdateButtonStatus	List frame의 버튼 상태 업데이트 및 처리
slotUpdateFocusStatus	List frame의 포커스 상태 업데이트
slotUpdateWandStatus	List frame의 완드 상태 업데이트

[표 5-23] List frame의 이벤트 핸들러

5) List item

이벤트 핸들러	내용
slotUpdate	List item 업데이트
slotUpdateButtonStatus	List item의 버튼 상태 업데이트
slotUpdateFocusStatus	List item의 포커스 상태 업데이트

[표 5-24] List item의 이벤트 핸들러

6) Keypad

이벤트 핸들러	내용
slotUpdateVisibilityChanged	키패드 위젯의 show/hide 상태 업데이트
slotUpdateButtonStatus	키패드 위젯의 버튼 입력 상태 처리
slotUpdateWandStatus	키패드 위젯의 내부 아이템(숫자 버튼 등) 탐색하기 위한 완드 상태 처리
slotUpdateWandDirection	키패드 위젯의 내부 아이템(숫자 버튼 등)을 탐색하기 위한 완드 방향 상태 처리
slotAcceptButtonClick	키패드를 통해 입력 받은 결과 승낙
slotRejectButtonClick	키패드를 통해 입력 받은 결과 거절
slotButton*_Clicked	키패드를 구성하는 각 버튼의 클릭 이벤트 처리
slotButtonClearClicked	현재 입력된 숫자를 전부 제거
slotButtonDeleteClicked	현재 입력된 숫자의 마지막 자리를 제거
slotButtonMinusClicked	음수를 표현하기 위한 이벤트 처리
slotButtonDotClicked	소수점을 표현하기 위한 이벤트 처리

[표 5-25] Keypad의 이벤트 핸들러

7) Keypad dialog

이벤트 핸들러	내용
slotUpdateKeypadValue	키패드를 통해 입력 받은 결과를 실수형으로 저장
slotUpdateKeypadValueInt	키패드를 통해 입력 받은 결과를 정수형으로 저장

[표 5-26] Keypad dialog의 이벤트 핸들러

8) Notification message box

이벤트 핸들러	내용
slotUpdateButtonStatus	메시지 박스의 버튼 상태 업데이트 함수

[표 5-27] Notification message box의 이벤트 핸들러

9) Select list frame

이벤트 핸들러	내용
slotUpdateButtonStatus	완드 버튼의 상태 업데이트 함수
slotUpdateFocusStatus	Select list frame의 포커스 업데이트 함수
slotUpdateWandStatus	완드의 방향 상태 업데이트 함수

[표 5-28] Select list frame의 이벤트 핸들러

10) Select list item

이벤트 핸들러	내용
slotUpdateButtonStatus	완드 버튼의 상태 업데이트 함수
slotUpdateFocusStatus	Select list item의 포커스 상태 업데이트 함수

[표 5-29] Select list item의 이벤트 핸들러

11) Select message box

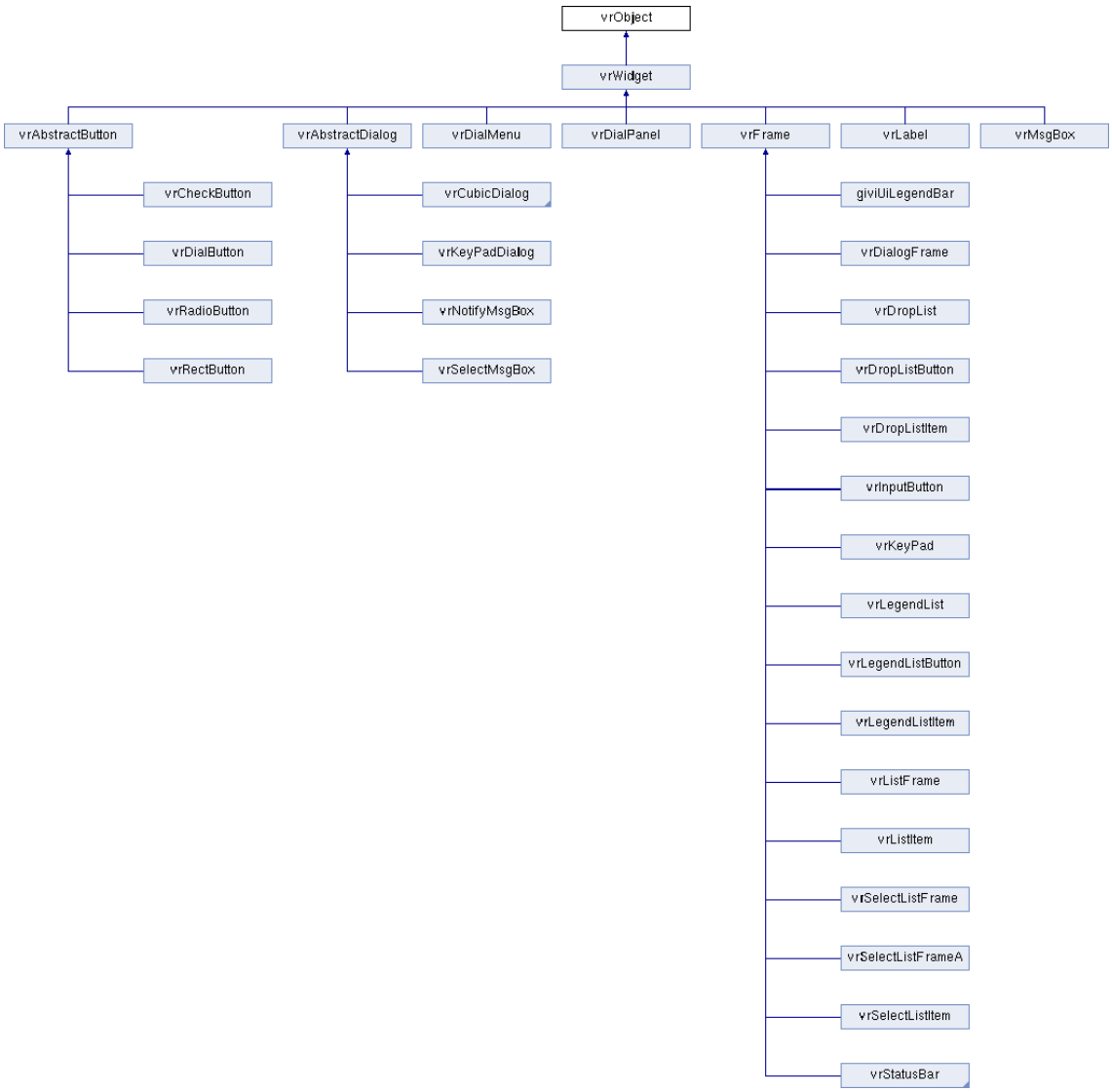
이벤트 핸들러	내용
slotUpdateButtonStatus	완드 버튼의 상태 업데이트 함수
slotUpdateWandStatus	완드의 방향 상태 업데이트 함수

[표 5-30] Select message box의 이벤트 핸들러

Appendix A

1. vrObject 클래스의 구성

가. vrObject의 클래스 다이어그램



나. 함수 목록

- Public member functions

```
                vrObject (vrObject *parent = NULL)
                vrObject (bool widgetType, vrObject *parent = NULL)
          bool  isWindowType () const
   std::string  objectName () const
          void  setObjectName (std::string)
```

- Public attributes

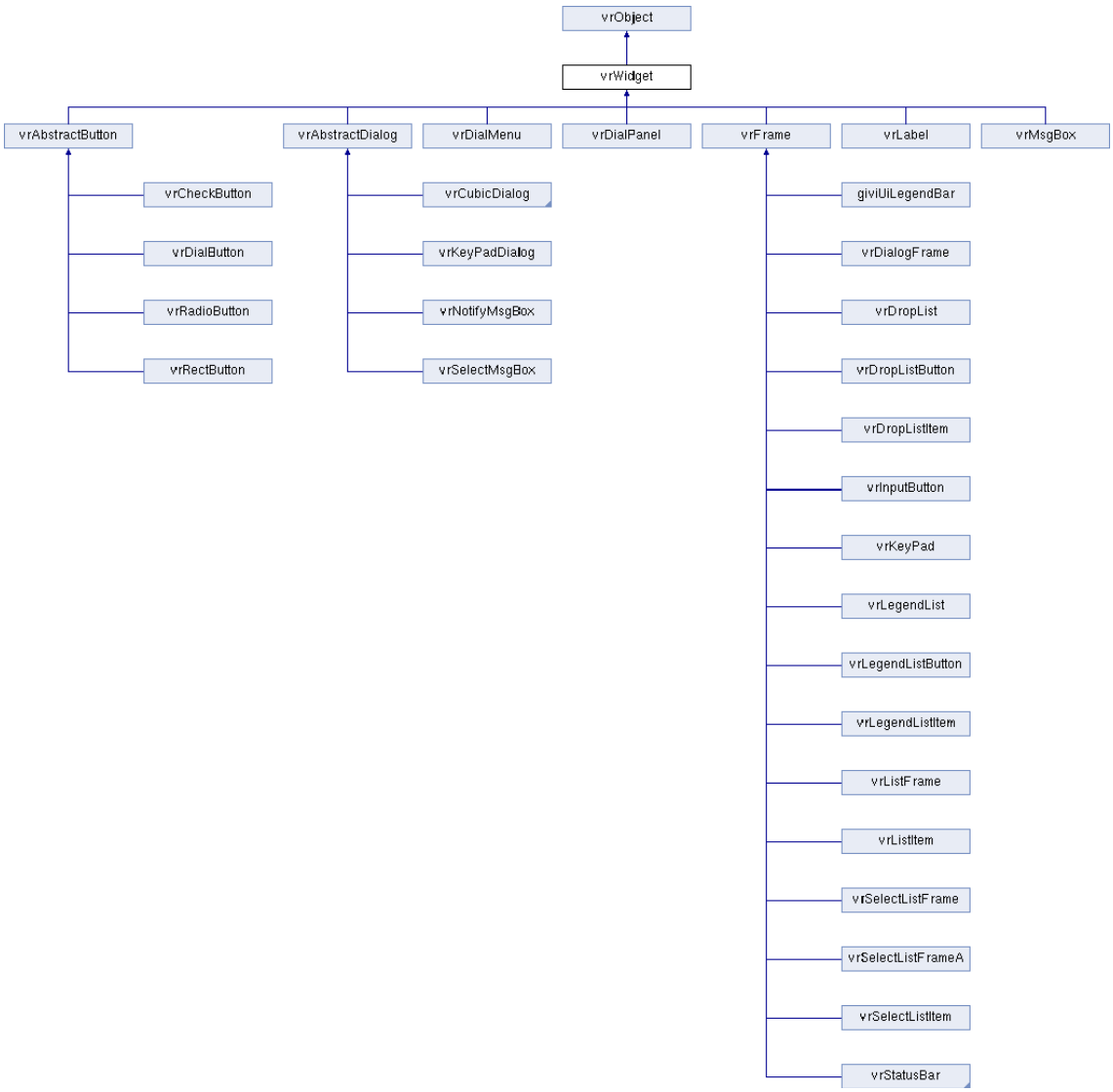
```
              PUBLIC_SIGNAL  signalCreated
             Signal<void()>  signalDestroyed
             Signal<void()>  signalPainted
             Signal<void()>  signalSizeChanged
             Signal<void()>  signalMoved
             Signal<void()>  signalClosed
  Signal<void(std::string)>  signalObjNameChanged
      Signal<void(vrEvent)>  eventSignal
                PUBLIC_SLOT  isWidgetType() const
```

- Protected member functions

```
  virtual bool  event (vrEvent)
  virtual void  customEvent (vrEvent)
    vrObject *  parentObj () const
          void  setParentObj (vrObject *)
```

2. vrWidget 클래스의 구성

가. vrWidget의 클래스 다이어그램



나. 함수 목록

- Public member functions

```
                                vrWidget (vrWidget *parent = NULL)
             virtual void  eventSetEventHold (vrActionType)
             virtual void  eventSetFocus (vrActionType)
             virtual void  evenSetVisibility (vrActionType)
             virtual void  eventUpdateButtonStatus (vrActionType)
             vritual void  eventUpdateStickStatus (vrActionType)
             virtual void  eventUpdateWandStatus (vrActionType)
             virtual void  eventUpdateWandDirection (osg::Vec3)
                     void  setTopWidget (bool)
                     void  move (float, float)
                     bool  isActivate () const
                     bool  isFocus () const
                     bool  isVisible () const
                     bool  isTopWidget () const
                     bool  isHoldEvent () const
                     void  setMatrix (osg::Matrix)
                     void  getMatrix (osg::Matrix &)
                     void  insertChild (vrWidget *)
                     void  insertChild (osg::ref_ptr<osg::Node>)
                     void  deleteChild (std::string)
  osg::ref_ptr<osg::Group>  getSceneGraph () const
                     void  setWidgetName (const std::string)
              std::string  getWidgetName () const
                     void  setWidgetStatus (WidgetStatus)
             WidgetStatus  getWidgetStatus () const
                     void  setParentWidget (vrWidget *parent)
               vrWidget *  getParentWidget ()
                     void  setStyle (vrWidgetStyle)
            vrWidgetStyle  getStyle () const
```

- Public attributes

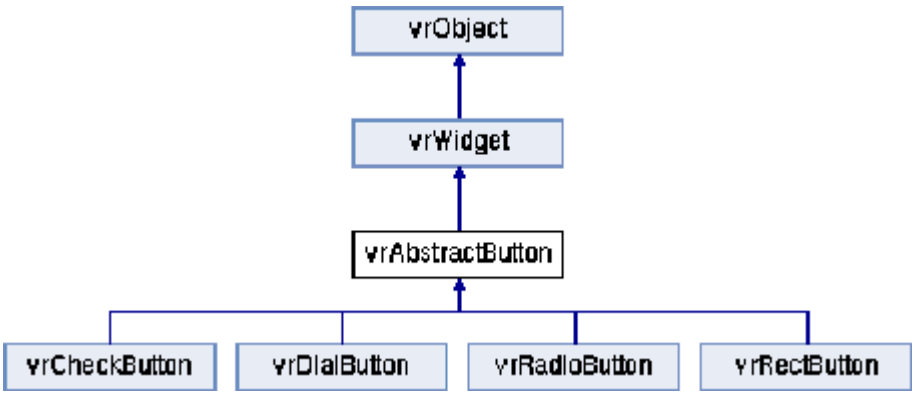
PROTECTED_SIGNAL	signalActivationChanged
Signal<void(vrActionType)>	signalFocusChanged
Signal<void(vrActionType)>	signalVisibilityChanged
Signal<void(vrActionType)>	signalHoldEventChanged
Signal<void(vrActionType)>	signalButtonStatusChanged
Signal<void(vrActionType)>	signalStickStatusChanged
Signal<void(vrActionType)>	signalWandStatusChanged
Signal<void(vrActionType)>	signalWandDirChanged
Signal<void(vrActionType)>	signalWandPosChanged
virtual bool	event (vrEvent)
virtual void	eventSetActivate (vrActionType)

- Protected member functions

virtual void	updateWidget ()
virtual void	customEvent (vrEvent)
void	setAllowedEvent (bool b=false, bool s=false, bool w=false)
virtual float	x () const
virtual float	y () const
virtual float	width () const
virtual float	height () const
virtual void	setGeometryLayout (vrGeometrySize<float>)
virtual void	setGeometryLayout (float x, float y, float width, float height)
vriual vrGeometrySize<float>	getGeometryLayout () const
virtual void	getGeometryLayout (float &x, float &y, float &w, float &h) const
virtual void	setSize (vrSize2D<float>)
virtual void	setSize (float w, float h)

3. vrAbstractButton 클래스의 구성

가. vrAbstractButton의 클래스 구조



나. 함수 목록

- Public member functions

```
                 vrAbstractButton (vrWidget *parent = NULL)
                 vrAbstractButton (ButtonAttribute att, vrWidget *parent =
                 NULL)
            void setCheckable (bool b)
            void setToggleable (bool b)
            void clicked ()
            void checked (bool b)
            void toggled (bool b)
            bool isCheckButton () const
            bool isToggleButton () const
            void setIcons (vrTexList *)
            void setElements (vrItemElement *)
vrItemElement * getElements () const
```

- Public attributes

```
      Signal<void()>  signalClicked
  Signal<void(bool)>  signalToggled
  Signal<void(bool)>  signalChecked
                void  slotButtonStatusChanged (vrActionType action)
```

- Protected member functions

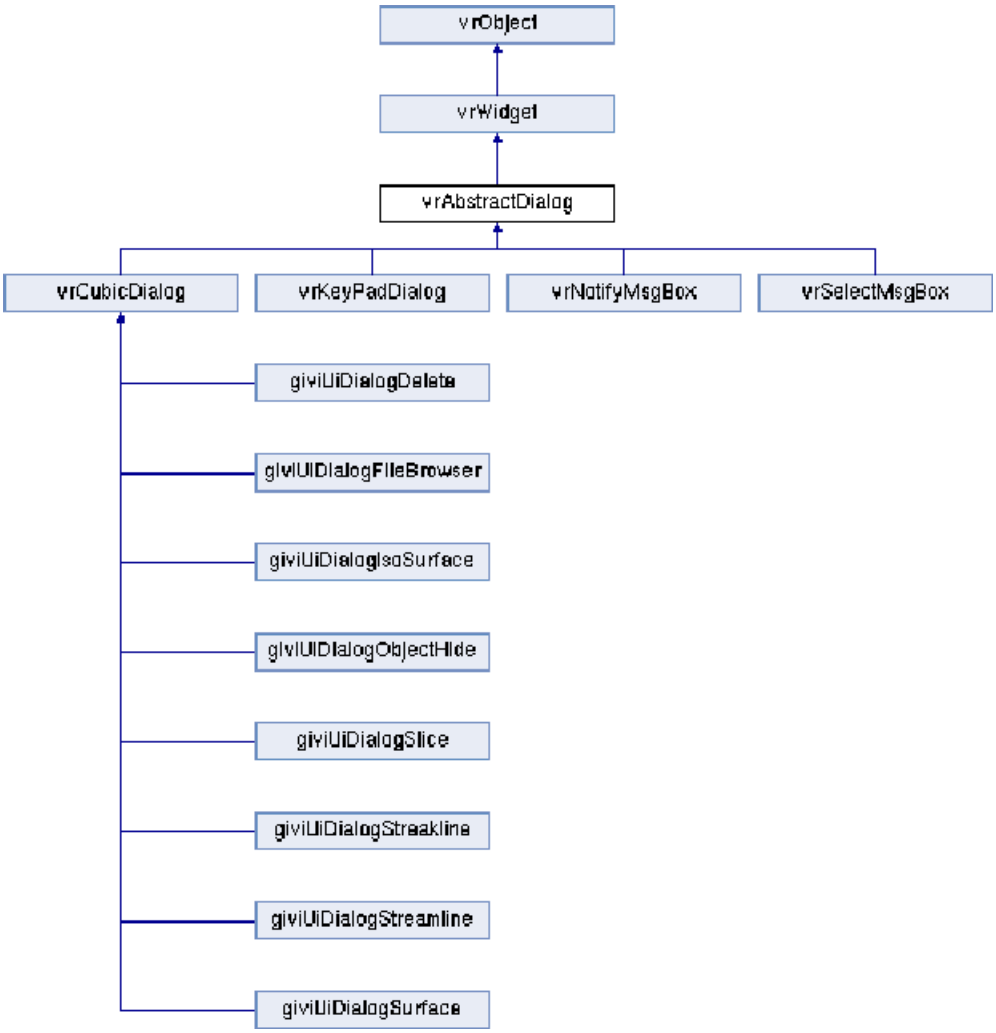
```
 virtual void  makeButton (float size, std::string text)
         void  updateButton
```

- Protected attributes

```
                        osg::Matrix  mShapeMatrix
                        osg::Matrix  mTexMatrix
                    ButtonAttribute  mButtonAttribute
                               bool  mIsButtonChecked
                               bool  mIsButtonToggled
                     vrItemElement *  mElement
                        vrTexList *  mTexList
           osg::ref_ptr<osg::Geode>  mTexGeode
 osg::ref_ptr<osg::MatrixTransform>  mTextTransform
         osg::ref_ptr<vrSafetyText>  mTextDrawable
           osg::ref_ptr<osg::Geode>  mButtonGeode
 osg::ref_ptr<osg::MatrixTransform>  mButtonTransform
```

4. vrAbstractDialog 클래스의 구성

가. vrAbstractDialog의 클래스 구조



나. 함수 목록

- Public member functions

```
                vrAbstractDialog (vrApplication *parent = NULL)
                vrAbstractDialog (bool isModal, vrApplication *parent = NULL)
 virtual void   rejectDialog ()
```

- Public attributes

```
Signal<void()>  signalAccepted
Signal<void()>  signalRejected
  virtual void  acceptDialog ()
```

- Protected member functions

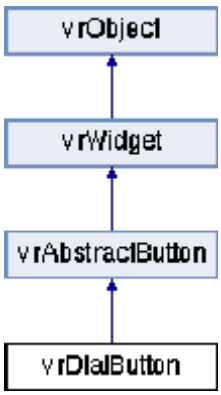
```
          void  exec ()
          void  setDialogStyle (vrDialogStyle)
vrDialogStyle   getDialogStyle () const
          void  setTitle (std::string)
   std::string  getTitle () const
 virtual void   procEvent (vrEvent)
          void  updateAFLayout (float bwRatio, float bhRatio, float sMargin,
                float vMargin, float cMargin)
          void  udpateDialogFrameSize (float thRatio, float chRatio, float
                ahRatio)
          void  updateDialogLayout ()
          bool  isEventThreadValid () const
```

- Protected member functions

bool	mClosingStatus
vrFrame *	mFocusedFrame
vrFrame *	mTitleFrame
vrFrame *	mContentsFrame
vrDialogFrame *	mAcceptBoxFrame
vrLabel *	mTitleLabel
vrRectButton *	mAcceptButton
vrRectButton *	mRejectButton

5. vrDialButton 클래스의 구성

가. vrDialButton의 클래스 구조



나. 함수 목록

- Public member functions

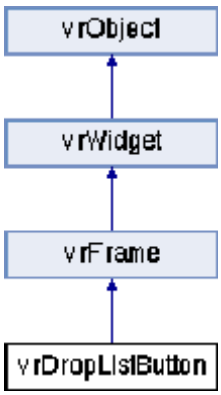
```
                                vrDialButton (vrWidget *parent = NULL)
                                vrDialButton (ButtonAttribute att, vrWidget
                                *parent = NULL)
                          void  slotShowLabel (bool)
                          void  slotUpdateActivationStatus (vrActionType)
                          void  slotUpdateFocusStatus (vrActionType)
                          void  setShaders (osg::ref_ptr<osg::Program>)
  osg::ref_ptr<osg::Program>  getShaders () const
                          void  makeButton (float size, std::string text)
```

- Public attributes

```
void  slotUpdate ()
void  slotButtonClick ()
```

6. vrDropListButton 클래스의 구성

가. vrDropListButton의 클래스 구조



나. 함수 목록

- Public member functions

```
                vrDropListButton (vrWidget *parent)
          void  slotUpdateButtonStatus (vrActionType)
          void  slotUpdateFocusStatus (vrActionType)
          void  slotSelectedListIdx (int)
          void  slotSelectedListName (std::string)
          void  updateLayout ()
          void  insertDropItemList (std::vector<std::string>)
          void  setDefautIdx (int)
          void  setItemColor (osg::Vec4)
          void  setHighlightColor (osg::Vec4)
   std::string  getSelectedText () const
           int  getSelectedIdx () const
```

- Public attributes

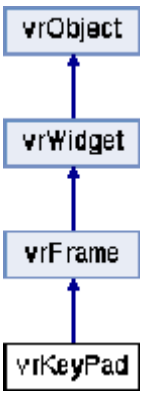
```
     void  slotUpdateActivationStatus (vrActionType)
```

- Protected member functions

  virtual bool `event (vrEvent)`

7. vrKeypad 클래스의 구성

가. vrKeypad의 클래스 구조



나. 함수 목록

- Public member functions

```
        vrKeyPad (bool floatType, vrWidget *parent)
void    slotUpdateVisibilityChanged (vrActionType)
void    slotUpdateWandStatus (vrActionType)
void    slotUpdateWandDirection (osg::Vec3)
void    updateLayout ()
void    setKeyPadStyle (vrDialogStyle, vrComponentStyle)
void    slotAcceptButtonClick ()
void    slotRejectButtonClick ()
void    slotButton0_Clocked ()
void    slotButton1_Clocked ()
void    slotButton2_Clocked ()
void    slotButton3_Clocked ()
void    slotButton4_Clocked ()
void    slotButton5_Clocked ()
void    slotButton6_Clocked ()
void    slotButton7_Clocked ()
void    slotButton8_Clocked ()
```

```
void  slotButton9_Clocked ()
void  slotButtonCliearClicked ()
void  slotButtonDeleteClicked ()
void  slotButtonMinusClicked ()
void  slotButtonDotClicked ()
```

- Public attributes

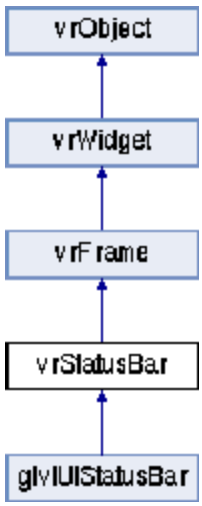
```
        Signal<void(bool)>  signalKeypadUsed
      Signal<void(double)>  signalAcceptValue
         Signal<void(int)>  signalAcceptValueInt
            Signal<void()>  signalAcceptVButtonClicked
            Signal<void()>  signalRejectButtonClicked
  Signal<void(std::string)>  signalValueUpdated
                     void  slotUpdateButtonStatus (vrActionType)
```

- Protected member functions

virtual bool event (vrEvent)

8. vrStatusBar 클래스의 구성

가. vrStatusBar의 클래스 구조



나. 함수 목록

- Public member functions

```
                vrStatusBar (vrApplication *parent)
       virtual  ~virStatusBar ()
  virtual void  setStatusMsg (std::string)
          void  updateLayout ()
```

- Protected attributes

```
  vrFrame *  mContentsBase
  vrLabel *  mContentsLabel
```

9. vrLabel 클래스의 구성

가. vrLabel의 클래스 구조

나. 함수 목록

- Public member functions

	vrLabel (vrWidget *parent = NULL)
	vrLabel (std::string, vrWidget *parent = NULL)
void	setTextStyle (osg::Vec4 tc, std::string fontFile, float ts, osgText::Text::AlignmentType)
void	updateText (std::string)
osg::ref_ptr<vrSafetyText>	getTextDrawable () const
void	setText (std::string)
void	setTextAlign (osgText::Text::AlignmentType)
void	setTextColor (float, float, float, float)
void	setTextColor (osg::Vec4)
void	setTextFont (std::string)
void	setTextSize (float)
void	setTextMaxLimit (float, float)
std::string	getText () const
float	getTextSize () const

- Protected attributes

osg::ref_ptr<vrSafetyText>	mTextDrawable
osg::ref_ptr<osg::Geode>	mTextGeode

10. vrMsgBox 클래스의 구성

가. vrMsgBox의 클래스 구조

나. 함수 목록

- Public member functions

```
        vrMsgBox (std::string msg, vrApplication *parent)
        vrMsgBox (std::string title, std::string msg, vrApplication *parent)
        vrMsgBox (MessageBoxType, std::string msg, vrApplication *parent)
        vrMsgBox (MessageBoxType, std::string title, std::string msg, vrApplication *parent)
virtual ~vrMsgBox ()
   bool exec ()
   void setMsg (std::string)
   void setTitleMsg (std::string)
```

- Protected member functions

```
virtual bool  event (vrEvent)
```