GUI 1 GUI 프로그램의 흐름

GUI 구성, 판(Base)과 버튼(Button)

간단하게 판 위에 버튼하나만 올려 보겠습니다.

```
pro firstgui
tlb=widget_base(title='Wow, My First GUI!')
exitbutton=widget_button(tlb, VALUE='Exit')
end
```

꼭 그래야 하는 것은 아니지만 Top Level Base(최상위 판)이라는 의미에서 tlb 변수를 썼습니다. 최상위 판은 그 위에 아무도 없지만, Button을 보시면 tlb 아래 소속된 요소라고 첫 번째 파라메터로 지정해 놓았습니다. 아직은 실행시켜 봐도 아무것도 보이질 않을 것입니다.

GUI의 일꾼, WIDGET_CONTROL

WIDGET_CONTROL 프로시저는 이름 그대로 모든 GUI 작동을 제어합니다. GUI 프로그래밍 기술의 핵심은 WIDGET_CONTROL에 있습니다. 앞에 만든 GUI를 세상에 보여 주는 것도 WIDGET_CONTROL이 담당합니다.

```
pro firstgui
tlb=widget_base(title='Wow, My First GUI!')
exitbutton=widget_button(tlb, VALUE='Exit')
WIDGET_CONTROL, tlb, /REALIZE
end
```

WIDGET_CONTROL, 제어대상ID [,키워드들]

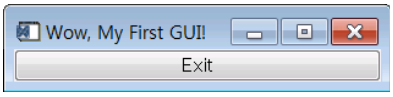
매우 간단한 문법을 가지고 있지만 실제 하고자 하는 일은 다양한 키워드를 조합하여 수행합니다. 여기서는 tlb가 가리키는 최상위 판을 구현하라(REALIZE)는 의미로 사용되었습니다.

아직 GUI의 모양이 마음에 들지는 않을 것입니다. 판이 너무 작다보니 판의 TITLE이 다 보이지도 않습니다.

GUI 요소의 다양한 키워드

WIDGET_BASE에서는 TITLE 이라는 키워드를 사용하였고 어떤 의미를 가지는지는 예상할 수 있을 것입니다 WIDGET_BUTTON은 VALUE라는 키워드를 사용하였고 버튼이므로 버튼 위에 올라갈 문자열이라는 것도 예상할 수 있습니다(그래픽 파일 경로를 지정하면 그림 버튼이 됩니다). 도움말을 펼쳐 보면 꽤 많은 키워드들이 있습니다. 버튼의 가로크기는 xsize로 조정합니다.

```
exitbutton=widget_button(tlb, VALUE='Exit', xsize=300)
```



위젯의 ID

tlb 라든지 exitButton과 같은 위젯(GUI의 요소)의 좌변 변수(ID)에는 어떤 값들이 들어 있을까요?

```
WIDGET_CONTROL, tlb, /REALIZE
print, 'TLB = ', tlb
print, 'exitButton = ', exitButton
end
```

프로그램을 실행시켜 보면 다음과 같은 결과가 나올 것입니다.

```
TLB =            13
exitButton =            14
```

어떤 숫자가 들어가고 있습니다. 이 숫자는 실행시킬 때마다 IDL이 알아서 배정하므로 값이 계속 달라질 수 있습니다. 어쨌든 이 번호만 알면 해당 위젯에 대해 우리는 무슨 일이든 할 수 있습니다. 예를 들어, 프로그램을 종료하고자 할 경우 최상위 판을 파기하면 됩니다. IDL 프롬프트에서 TLB(위 실행 결과의 경우는 13번)를 파기해 봅시다. 열려있던 GUI 창이 닫히는 것을 확인하세요.

```
IDL> widget_control, 13, /DESTROY
```

WIDGET_CONTROL은 이와 같이 위젯의 ID를 이용하여 제어합니다.

위젯의 작동을 관리하는 XMANAGER

지금까지 만든 프로그램은 아무리 Exit 버튼을 눌러도 반응하지 않습니다. 버튼을 눌렀을 때를 포함해서 이 GUI에 대한 자극(이벤트)에 대해 반응할 프로그램(이벤트 처리)를 만들지 않았기 때문입니다. XMANAGER 프로시저는 모든 이벤트를 받아들이고 그 때마다 지정된 이벤트 처리 프로그램을 호출하는 일을 수행합니다. 복잡할 거 같은 일인데, 그냥 다음과 같이 쓰면 됩니다.

xmanager, '위젯을생성한프로그램이름', 최상위판ID

```
WIDGET_CONTROL, tlb, /REALIZE
XMANAGER, 'firstgui', tlb
end
```

프로그램을 실행하고 버튼을 클릭하면 에러가 발생할 것입니다.

```
% XMANAGER: Caught unexpected error from client application.
Message follows...
% Attempt to call undefined procedure/function: 'FIRSTGUI_EVENT'.
```

FIRSTGUI_EVENT 라는 프로시저나 함수가 없다는 불평이군요. 따로 지정하지 않았다면 XMANAGER는 메인루틴의 이름 뒤에 '_event' 가 붙은 루틴을 이벤트 처리 루틴으로 호출합니다.

일단 프로그램이 XMANAGER 안에서 멈추어 있으므로 상황을 다시 정리해 원상태로 돌려 놓습니다.

```
IDL> retall
```

이벤트 처리 루틴

이벤트 처리 루틴으로 XMANAGER는 딱 하나의 변수만 전달해 줍니다. 이벤트 구조체라는 변수인데 어떤 값이 전달되는지 한번 봅시다.

xmanager는 무슨 일이 일어나면 일단 firstgui_event를 호출할 것이고 이 때 이벤트 구조체를 하나 넘겨줄 것입니다. 위의 firstgui_event 루틴에서는 ev 변수로 이 이벤트 구조체를 받아냅니다.

```
pro firstgui_event, ev
help, ev, /struct
end

pro firstgui
tlb=widget_base(title='Wow, My First GUI!')
exitbutton=widget_button(tlb, VALUE='Exit', xsize=300)
WIDGET_CONTROL, tlb, /REALIZE
xmanager, 'firstgui', tlb
end
```

이벤트 구조체

프로그램을 실행시키고 버튼을 클릭하면 이벤트 처리 루틴이 호출될 것입니다. 이벤트 구조체에 어떤 값이 전달되는지 확인해 보세요.

```
**Structure WIDGET_BUTTON,4tags,length=16,data length=16:
   ID              LONG                 18
   TOP             LONG                 17
   HANDLER         LONG                 17
   SELECT          LONG                  1
```

모든 이벤트 구조체는 ID, TOP, HANDLER를 가지고 있습니다.

- ID : 이벤트를 발생시킨 위젯의 ID
- TOP : 최상위 판의 ID
- HANDLER : 이벤트 처리와 연계된 위젯 ID

ID와 HANDLER가 같은 것처럼 보이겠지만(사실 대부분 같습니다), 아닐 때도 분명 존재하는데, 예를 들어 라디오버튼 그룹 같은 경우, 각 버튼마다 이벤트 처리를 따로 하지 않고 버튼 그룹에 대한 이벤트로 처리하는 것이 일반적입니다. 이런 경우라면 ID(각각의 라디오 버튼)과 핸들러(라디오버튼 그룹)이 다를 수 있습니다. 이 내용은 앞으로 다른 예제에서 더 자세히 다루어질 기회가 있을 것입니다.

이벤트 구조체는 위 세가지 기본 요소 외에 GUI 마다 특성화된 내용을 가지고 있습니다. 버튼의 경우 SELECT 필드가 있어서 선택이 되었는지(1) 해제 되었는지(0)를 이벤트 구조체에 묶어 보내고 있는 것을 확인할 수 있습니다.

이벤트를 발생한 위젯 구별

이 단순한 예제는 버튼이 하나밖에 없습니다만 여러개의 버튼, 텍스트창, 메뉴막대, 그림창, 스프레드시트 등로 구성된 GUI에서 누가 이벤트를 일으켰는지 어떻게 알 수 있을까요? 이벤트 구조체의 ID 필드가 해당 위젯의 ID를 알려 주긴 하지만, ID는 단순히 숫자일 뿐입니다. 이 숫자만 가지고는 상황을 판단하기 어렵습니다.

각 GUI 요소는 UVALUE 키워드가 있어 어떤 값이든지 하나를 저장할 수 있습니다. 다른 방법도 있긴 하지만 많은 경우 UVALUE를 활용하여 위젯을 식별합니다.

```
pro firstgui_event, ev
widget_control, ev.id, GET_UVALUE=uval
if uval eq 'exitbutton' then $
        widget_control, ev.top, /DESTROY
end

pro firstgui
tlb=widget_base(title='Wow, My First GUI!')
exitbutton=widget_button(tlb, VALUE='Exit', xsize=300, $
  uvalue='exitbutton')
WIDGET_CONTROL, tlb, /REALIZE
xmanager, 'firstgui', tlb
end
```

WIDGET_CONTROL을 이용하여 UVALUE를 가져오는 구문은 GUI 프로그램에서 매우 흔하게 쓰입니다. 이벤트 구조체의 ID와 TOP이 활용되는 것도 살펴 두세요.

GUI 프로그램의 흐름

종료만 할 수 있는 극도로 간단한 예제 프로그램이었지만, 다음 내용은 일반적인 GUI 프로그래밍 기법입니다.

- GUI의 구성은 판(BASE)위에 GUI 요소(Button, Text 등)를 올리는 형태로 만들어 냅니다.
- GUI 관련의 일이라고 할 수 있는 거의 모든 것들은 WIDGET_CONTROL이 수행합니다. GUI 프로그램의 시작(/REALIZE)과 끝(/DESTROY)도 담당합니다.
- XMANAGER가 GUI 프로그램의 흐름을 총괄지휘합니다.
- UVALUE는 사용자가 필요한 의도대로 아무 데이터 타입이든 저장하고 꺼내 쓸 수 있게 만든 보조 저장소입니다. 이벤트를 발생시킨 위젯의 정체를 분석하는 데에도 활용될 수 있습니다.
- 이벤트 처리 루틴으로는 오직 이벤트 구조체 하나만 넘어갑니다. 이 안에서 ID와 TOP을 꺼내고, 이를 시작으로 WIDGET_CONTROL을 이용해 필요한 정보를 조사하거나 위젯에 어떤 조치를 취하게 됩니다.