User Interface - Bug #8673

Deduce the logic behind OE focus management and implement in FWD

04/24/2024 08:00 AM - Vladimir Tsichevski

Status:	WIP	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	30%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

Related issues:

Related to User Interface - Bug #8376: Focus transfer problem with frames hav...	**New**	
Related to User Interface - Bug #8095: LEAVE trigger fires in no focus change...	**New**	
Related to User Interface - Bug #7786: FOCUS handle must return only field-le...	**New**	
Related to User Interface - Bug #7685: LEAVE/ENTRY events are raised before M...	**New**	**08/14/2023**
Related to User Interface - Bug #7684: ENTRY event is emitted twice	**New**	**08/14/2023**
Related to User Interface - Support #7339: UI - Focus	**New**	
Related to User Interface - Bug #8711: TAB-STOP attribute wrong implementatio...	**New**	
Related to User Interface - Bug #8770: ENTRY event is not sent for a frame wh...	**New**	

History

#1 - 04/24/2024 08:56 AM - Vladimir Tsichevski

- File 8376-with-focusable.p added

There is a feature related to how OE manages focus I cannot understand.

I wrote the 8376-with-focusable.p example to learn focus-related stuff.

In this example there is a number of frames, most of them nested, which contain a number of buttons.

When I click with mouse in any frame, the first button in this frame is focused.

But there is a single exception: the frame named outerFrame has two enabled buttons, but clicking inside this frame does **not** move focus to one of the frame buttons.

I am stuck, since I cannot find out, why is this frame so special.
Probably, I am missing something obvious?

The logic behind this feature should be discovered in order to be implemented in FWD.

#3 - 04/30/2024 01:49 PM - Vladimir Tsichevski

- Status changed from New to WIP

- % Done changed from 0 to 20

When the target is a frame, and there is no focus hint set in this frame (something roughly similar to the AbstractContainer#current field in FWD), OE searches for the suitable field in its field group. This is known.

The new hypothesis: the search is aborted on the first non-field item (a frame) found in the field group.

In the 8376-with-focusable.p example the outerFrame is enabled with the following statement:

```
ENABLE focusable4 focusable6 WITH FRAME outerFrame.
```

This way results in that the focusable4 and focusable6 buttons move to the end of the field group widgets, and the innerFrame is the first item in the field group.

So, when the user click the outerFrame with mouse the search for the child suitable as focus target, aborts on the innerFrame, and the outerFrame stays as the focus target.

#4 - 04/30/2024 03:33 PM - Vladimir Tsichevski

- Related to Bug #8376: Focus transfer problem with frames having no focusable contents added

#5 - 04/30/2024 03:39 PM - Vladimir Tsichevski

- Related to Bug #8095: LEAVE trigger fires in no focus change situation added

#6 - 04/30/2024 03:47 PM - Vladimir Tsichevski

- Related to Bug #7786: FOCUS handle must return only field-level widgets added

#7 - 04/30/2024 03:49 PM - Vladimir Tsichevski

- Related to Bug #7685: LEAVE/ENTRY events are raised before MOUSE-UP event. added

#8 - 04/30/2024 03:51 PM - Vladimir Tsichevski

- Related to Bug #7684: ENTRY event is emitted twice added

#9 - 05/01/2024 08:16 AM - Vladimir Tsichevski

- Related to Support #7339: UI - Focus added

#10 - 05/01/2024 01:30 PM - Vladimir Tsichevski

- Related to Bug #8711: TAB-STOP attribute wrong implementation for RECTANGLE, IMAGE and TEXT widgets added

#11 - 05/01/2024 05:43 PM - Vladimir Tsichevski

Focus data model

Focus state consists of the following data:

1. Window.focus
   Field-level focused widget, which receives keyboard input, initially null.
   The OE FOCUS handle contains the value of this field for the currently focused window.

1. Window.rawFocus.
   The last widget the user selected by mouse. Initially null. Can be field-level or frame. This field cannot be directly accessed from 4gl.

1. Frame.currentFocus
   The preferred FOCUS candidate field-level.
   This field is either null or points to one of field-level members of the frame field group, which was last selected for Window.focus.

Focus change by mouse events control flow

Focus state can be affected by MOUSE DOWN and MOUSE UP actions.

The control flow looks as follows:

1. Decide if this mouse type/button combination can cause focus-related actions, return if not.
2. Calculate the effective **target focus** (which can be either a field-level or a frame).
3. If the resulting target focus is the same as the current Window.rawFocus value, then return, do nothing.
4. Fire necessary LEAVE/ENTRY event triggers.
5. If no trigger returned NO_APPLY: # If the target focus is a field-level, assign it to Window.focus and update Frame.currentFocus for the field frame.
6. Save the target focus to Window.rawFocus.

Which mouse event may initiate a focus change

Warning: this section describes the FWD implementation, the information needs probably be checked with original OE.

1. The mouse button is the *LEFT* button for any widget or the mouse button is the *RIGHT* button for any widgets, except the following FWD widget classes:
   1. BorderedPanelGuiImpl
   2. EditorGuiImpl
   3. FillInGuiImpl
   4. ScrollbarGuiButton
   5. ScrollbarGuiImpl
2. event is one of UP, DOWN or CLICK.
3. widget is any, except:
   1. ScrollbarGuiButton
   2. ScrollbarGuiImpl

Effective target focus calculation

1. Initially use the **event source** as the target focus.
2. If the target focus is a widget which cannot be a target focus (either a disabled field-level, or a widget which by its nature does not accept keyboard input), then assign widget's frame to target focus.
3. If target focus is **not a frame**, return the target focus value
4. If the Frame.currentFocus is a **field-level**, then return the Frame.currentFocus value
5. Search the target focus field group **tab ring** for a suitable field: search until either an **enabled field-level** found, and we use it as the new target, or a **frame found**, which aborts the search.

Fire LEAVE/ENTRY event triggers

Given a target focus, fire ENTRY and LEAVE events. Abort the operation if any trigger returns NO-APPLY.

1. If target focus differs from the Window.rawFocus:
   1. If Window.rawFocus is set:
      1. Fire LEAVE trigger for Window.rawFocus
      2. If Window.rawFocus is a **field-level**:
         1. If target focus is a **field-level**:
            1. If the frame of the target focus differs from the frame of Window.rawFocus:
               1. fire LEAVE trigger for the frame of Window.rawFocus
               2. if the Frame.currentFocus field of the frame of the target focus is set or the Frame.currentFocus field of the frame of the target focus differs from the target focus:
                  1. fire ENTRY trigger for the frame of the target focus
         2. Otherwise (target focus is a **frame**):
            1. fire LEAVE trigger for the frame of Window.rawFocus
      3. Otherwise (Window.rawFocus is a frame):
         1. If target focus is **field-level** and the frame of the target focus differs from Window.rawFocus
            and (the frame of the target focus.currentFocus is not set
            or the Frame.currentFocus field of the frame of the target focus differs from the target focus):
            1. fire ENTRY trigger for the **frame of the target focus**
   2. Otherwise (Window.rawFocus is not set):

    1. if target focus is a **field-level** and (Frame.currentFocus field of the frame of the target focus is not set or Frame.currentFocus field of the frame of the target focus differs from target focus):   # fire ENTRY trigger for the frame of the target focus
2. fire ENTRY trigger for the target focus

#12 - 05/03/2024 12:21 PM - Vladimir Tsichevski

Question to gurus:

A Progress trigger can return a value. If the NO-APPLY is returned, the further event processing is aborted.

The question: are values other than the NO-APPLY ever used by Progress?

#13 - 05/03/2024 02:03 PM - Greg Shah

First, I want to record some of our older documentation:

https://proj.goldencode.com/artifacts/javadoc/latest/api/com/goldencode/p2j/ui/chui/package-summary.html#Focus_Management
https://proj.goldencode.com/artifacts/javadoc/latest/api/com/goldencode/p2j/convert/package-summary.html#Transaction_Processing_and_Block

The first reference is not 100% complete. It especially was only looked at in ChUI and the influence of mouse processing, drag and drop, multiple windows and child frames are not considered.

The second reference is based on 9.x and thus is not up to date with the additional block processing of OO, structured error handling or the more recent transaction processing related changes in 10.x and 11.x. It also doesn't consider anything related to appserver/REST/SOAP agent session processing, nor anything about persistent/super procedures.

Even with these caveats, these are useful resources.

> The question: are values other than the NO-APPLY ever used by Progress?

Limiting this question to triggers, the answer is yes and no.

This may help (it is the javadoc from BlockManager.returnWorker()):

> Return from the nearest enclosing top-level block. This will process differently based on the type of the top-level block (internal/external procedures, triggers or user-defined functions), the type of return processing that is requested (see BlockManager.ReturnType) and the presence or absence of a value to return.
>
> The following behavior is provided:

Codes	Meaning
C	Invoke LogicalTerminal.consume()
E	Invoke TransactionManager.triggerErrorInCaller()
V	Invoke ControlFlowOps.setReturnValue(val)
N	Invoke ControlFlowOps.setReturnValue("")
F	Invoke setFuncReturn(val)
U	Invoke setFuncReturn(null) (set return value as unknown)
I	Invalid (compile error in Progress)

Return Value	Return Type	Internal Proc	External Proc	Trigger	Function
null	NORMAL	N	N	N	U
non-null	NORMAL	V	V	V	F
null	CONSUME	N	N	NC	U
non-null	CONSUME	V	V	VC	F
null	ERROR	NE	NE	I	U
non-null	ERROR	VE	VE	I	U

In all cases, a ReturnUnwindException will be thrown and this will cause the stack to unwind to the nearest enclosing top-level block. That block will then handle the return in a block specific manner. For example, in a user-defined function, the given value or by default the unknown value will be returned to the caller. Procedures and triggers have void return values.

The part about the ReturnUnwindException is no longer 100% correct but otherwise the details are quite accurate and match the 4GL behavior. The way to interpret (for triggers) is as follows:

- A RETURN. ("NORMAL" return without a value) in a trigger sets the RETURN-VALUE to "" and then returns, allowing default processing to occur.
- A RETURN <expression>. ("NORMAL" return with a value) in a trigger sets the RETURN-VALUE to the evaluated value of the <expression> and then returns, allowing default processing to occur.
- A RETURN NO-APPLY. ("CONSUME" return without a value) in a trigger sets the RETURN-VALUE to "" and then returns, suppressing the default processing.
- A RETURN NO-APPLY <expression>. ("CONSUME" return with a value) in a trigger sets the RETURN-VALUE to the evaluated value of the <expression> and then returns, suppressing the default processing.
- A RETURN ERROR. ("ERROR" return without a value) in a trigger causes a compilation error.
- A RETURN ERROR <expression>. ("ERROR" return with a value) in a trigger causes a compilation error.

So these other constructions can affect the behavior but in no case does the value actually get returned as in a function call. In the "NORMAL" return with a value and "CONSUME" return with a value, the value can only be found in RETURN-VALUE. Relying upon this is tricky at best, because of the very idea of a trigger. It is happening in response to an event occurring, so how do you check the RETURN-VALUE just after a trigger executes BUT BEFORE any other procedure or trigger processing overwrites the RETURN-VALUE. Since there is just one global RETURN-VALUE, it is a pretty poor idea to use it from a trigger, in my opinion.

I hope this helps.

Marian: If I've gotten anything wrong here, please do let us know.

#14 - 05/03/2024 04:00 PM - Vladimir Tsichevski

Thanks, Greg.

One question:

The following two RETURN <expression>. statements look the same:

- A RETURN <expression>. ("NORMAL" return with a value) in a trigger sets the RETURN-VALUE to the evaluated value of the <expression> and then returns, allowing default processing to occur.
- A RETURN <expression>. ("CONSUME" return with a value) in a trigger sets the RETURN-VALUE to the evaluated value of the <expression> and then returns, suppressing the default processing.

How can the "CONSUME" return with a value be expressed in 4gl?

#15 - 05/03/2024 05:34 PM - Greg Shah

I edited it. It should have been RETURN NO-APPLY <expression>..

#16 - 05/13/2024 01:44 PM - Vladimir Tsichevski

- File 8376-with-focusable.p added

- File functions.i added

The #8673-11 now contains the focus data model and the focus change by mouse flow. Please, comment.

The demo program used to test the model is 8376-with-focusable.p (with the include file functions.i).

#17 - 05/13/2024 01:44 PM - Vladimir Tsichevski

- % Done changed from 20 to 30

#18 - 05/14/2024 02:47 PM - Vladimir Tsichevski

- File printEntryLeave.i added

The missing printEntryLeave.i required to run the 8376-with-focusable.p uploaded.

#19 - 05/14/2024 03:53 PM - Vladimir Tsichevski

- Related to Bug #8770: ENTRY event is not sent for a frame when window receives focus added

Files

8376-with-focusable.p	5.53 KB	04/24/2024	Vladimir Tsichevski
functions.i	1.71 KB	05/13/2024	Vladimir Tsichevski
8376-with-focusable.p	3.87 KB	05/13/2024	Vladimir Tsichevski
printEntryLeave.i	133 Bytes	05/14/2024	Vladimir Tsichevski