User Interface - Bug #3062

COMBO-BOX steals focus when its drop-down active

04/11/2016 04:35 AM - Hynek Cihlar

Status:	Closed	**Start date:**	04/11/2016
Priority:	Normal	**Due date:**	
Assignee:	Hynek Cihlar	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	Cleanup and Stabilization for GUI		
billable:	No	**case_num:**	
vendor_id:	GCD	**version:**	

Description

History

#1 - 04/11/2016 05:04 AM - Hynek Cihlar

Can be reproduced in current trunk in GUI.

Steps:

1. Click COMBO-BOX A drop-down button to activate its drop-down.
2. Click COMBO-BOX B drop-down button to activate its drop-down.
3. Repeat until the problem occurs - the clicked combo will not open its drop-down.

Use combo_box/combo_box21_1.p for testing.

I have identified two places in the code that explicitly place focus in the combo-box, this makes the case above fail. See the TODOs in ComboBox.exitDropDown() and in ThinClient.waitForWorker() just before the call to fixWaitForFocus().

#2 - 05/03/2016 05:34 AM - Hynek Cihlar

- Assignee set to Hynek Cihlar

#3 - 05/03/2016 05:34 AM - Hynek Cihlar

- Status changed from New to WIP

#4 - 05/03/2016 08:47 AM - Hynek Cihlar

One of the causes of this issue is the way combo drop-down waitForWorker() loop is interrupted and subsequently focus handled.

Here's the workflow that produces expected result:

- User clicks in the COMBO-BOX.
- Drop-down is displayed, current TC.waitFor() loop is suspended by executing TC.waitForWorker(), current focused widget is remembered, and the app starts to process the drop-down events.
- User hits ESC key, the key action is eventually translated to an EndConditionException, control returns to the suspended TC.waitFor() and the focus is transferred back to the remembered widget above (COMBO-BOX in this case).

Here's the workflow that does not produce expected result:

- User clicks in the COMBO-BOX.

- Drop-down is displayed, current TC.waitFor() loop is suspended by executing TC.waitForWorker(), current focused widget is remembered, and the app starts to process the drop-down events.
- User clicks into a frame widget, the click is translated into an ESC key action by the combo-box itself.
- The ESC key action is eventually translated to an EndConditionException, control returns to the suspended TC.waitFor() and the focus is transferred back to the remembered widget above (COMBO-BOX in this case) instead of the clicked widget.

To fix this I need to somehow know in the focus-restoration logic how the wait-for was interrupted. If it was interrupted by a click with explicitly choosing the next focus, then the focus-restoration logic should not interfere.

Constantin/Greg, how do you think the drop-down wait-for loop should be interrupted with respect to the focus restoration logic? Should I use another existing condition, come up with a new condition for this specific case or anything else?

Personally I wouldn't translate the click event into the ESC key action, this masks out the true evidence. Maybe I would just throw an end-condition exception directly from one of the drop-down classes.

#5 - 05/03/2016 10:30 AM - Constantin Asofiei

Hynek Cihlar wrote:

> Here's the workflow that does not produce expected result:
>
> - User clicks in the COMBO-BOX.
> - Drop-down is displayed, current TC.waitFor() loop is suspended by executing TC.waitForWorker(), current focused widget is remembered, and the app starts to process the drop-down events.
> - User clicks into a frame widget, the click is translated into an ESC key action by the combo-box itself.
> - The ESC key action is eventually translated to an EndConditionException, control returns to the suspended TC.waitFor() and the focus is transferred back to the remembered widget above (COMBO-BOX in this case) instead of the clicked widget.

Who transfers the focus back to the combo-box? The combo's own 'synthetic' wait-for loop or the outer wait-for loop?

#6 - 05/03/2016 10:40 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > Here's the workflow that does not produce expected result:
> >
> > - User clicks in the COMBO-BOX.
> > - Drop-down is displayed, current TC.waitFor() loop is suspended by executing TC.waitForWorker(), current focused widget is remembered, and the app starts to process the drop-down events.
> > - User clicks into a frame widget, the click is translated into an ESC key action by the combo-box itself.
> > - The ESC key action is eventually translated to an EndConditionException, control returns to the suspended TC.waitFor() and the focus is transferred back to the remembered widget above (COMBO-BOX in this case) instead of the clicked widget.

> Who transfers the focus back to the combo-box? The combo's own 'synthetic' wait-for loop or the outer wait-for loop?

It is the combo's synthetic wait-for loop who transfers the focus, it happens in TC.fixWaitForFocus() after the end condition is raised.

#7 - 05/03/2016 10:59 AM - Constantin Asofiei

Hynek Cihlar wrote:

> > Who transfers the focus back to the combo-box? The combo's own 'synthetic' wait-for loop or the outer wait-for loop?
>
> It is the combo's synthetic wait-for loop who transfers the focus, it happens in TC.fixWaitForFocus() after the end condition is raised.

I think it might work just ignoring the fixWaitForFocus in cases when the wait-for is synthetic: i.e. add a new BlockingOperation.SYNTHETIC_WAIT_FOR and pass it to the waitForWorker call for the drop-down.

Also, I'm wondering if this is not related to the multi-window focus management in GUI: if there is a wait-for inside a trigger, and there are multiple windows, what happens with the focus at the end of the inner wait-for, if it's not in the same window as the trigger's window? Does it get moved back?

#8 - 05/03/2016 11:10 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > > Who transfers the focus back to the combo-box? The combo's own 'synthetic' wait-for loop or the outer wait-for loop?
> >
> > It is the combo's synthetic wait-for loop who transfers the focus, it happens in TC.fixWaitForFocus() after the end condition is raised.
>
> I think it might work just ignoring the fixWaitForFocus in cases when the wait-for is synthetic: i.e. add a new BlockingOperation.SYNTHETIC_WAIT_FOR and pass it to the waitForWorker call for the drop-down.

I think I need fixWaitForFocus() in case real ESC key is pressed, at least in ChUI where the drop-down is not a separate window and focus needs to be transferred back to the combo. But I could use BlockingOperation.SYNTHETIC_WAIT_FOR in GUI I guess.

> Also, I'm wondering if this is not related to the multi-window focus management in GUI: if there is a wait-for inside a trigger, and there are multiple windows, what happens with the focus at the end of the inner wait-for, if it's not in the same window as the trigger's window? Does it get moved back?

Good question, will try.

#9 - 05/03/2016 12:25 PM - Hynek Cihlar

Hynek Cihlar wrote:

> Constantin Asofiei wrote:
>
> > Hynek Cihlar wrote:
> > Also, I'm wondering if this is not related to the multi-window focus management in GUI: if there is a wait-for inside a trigger, and there are multiple windows, what happens with the focus at the end of the inner wait-for, if it's not in the same window as the trigger's window? Does it get moved back?
>
> Good question, will try.

Yes, native 4GL transfers focus back to the window/widget of the wait-for inside the trigger. fixWaitForFocus() properly replicates this behavior in P2J. This holds true even when the event being waited for is a mouse click, see below.

```
DEF VAR h1 AS HANDLE.
DEF VAR in1 AS CHAR.
DEF VAR in2 AS CHAR.

ENABLE in1 WITH FRAME f1.

CREATE WINDOW h1.
h1:VISIBLE = TRUE.
ENABLE in2 WITH FRAME f2 IN WINDOW h1.

ON 'i':U ANYWHERE
DO:
   MESSAGE "i trigger".
   WAIT-FOR MOUSE-SELECT-CLICK OF in2 IN FRAME f2.
   /* focus is moved to in1 in frame f1 */
   RETURN.
END.

WAIT-FOR CLOSE OF DEFAULT-WINDOW.
```

#10 - 05/05/2016 12:08 PM - Hynek Cihlar

3062a rebased against trunk 11023, now at 11025.

#11 - 05/06/2016 05:52 PM - Hynek Cihlar

3062a revision 11028 resolves all the issues of this issue plus #3071 except one:

Combo drop-down would not open when combo-box button clicked while another drop-down active. The cause would be best explained on the UI workflow:
1. Combo A button is clicked.
2. Mouse pressed event is dispatched, the dispatcher causes the drop-down A to be displayed and the synthetic wait-for event loop to be started.
3. Combo B button is clicked.
4. Mouse pressed event for combo B button is dispatched, the dispatcher causes the drop-down B to be displayed and the synthetic wait-for event loop to be started (while the A's event loop still on call stack).
5. Window deactivation message for drop-down A is dispatched, the dispatcher causes the most recent synthetic wait-for event loop to be exited (combo B).

Clearly point 4 above is where things get wrong, another wait-for event loop is started while the previous one is still running.

I see three solutions: (1) make sure the driver sends the events in correct and guaranteed order, that is window deactivated and THEN mouse pressed. (2) Track all opened overlay windows and add additional logic to close the active overlay when mouse pressed. (3) Get rid of the synthetic wait-for event loop.

I am inclining towards solution (3) as it seems to be the cleanest and simplest one, I don't see why the synthetic wait-for event loop is needed in the first place.

#12 - 05/07/2016 03:22 AM - Constantin Asofiei

Hynek Cihlar wrote:

> I see three solutions: (1) make sure the driver sends the events in correct and guaranteed order, that is window deactivated and THEN mouse pressed. (2) Track all opened overlay windows and add additional logic to close the active overlay when mouse pressed. (3) Get rid of the synthetic wait-for event loop.
>
> I am inclining towards solution (3) as it seems to be the cleanest and simplest one, I don't see why the synthetic wait-for event loop is needed in the first place.

The main reason why a synthetic wait-for is used to open the drop-down is that all triggers are 'hidden' while the drop-down is active. Also, I don't think you need to track all opened overlay windows: for combo's drop-down, only one can be open at a time; the same for popup-menus. So, maybe is possible to track only the current opened drop-down/popup-menu and close that if i.e. mouse is pressed outside of it?

#13 - 05/07/2016 04:12 AM - Hynek Cihlar

Constantin Asofiei wrote:

> Hynek Cihlar wrote:
>
> > I see three solutions: (1) make sure the driver sends the events in correct and guaranteed order, that is window deactivated and THEN mouse pressed. (2) Track all opened overlay windows and add additional logic to close the active overlay when mouse pressed. (3) Get rid of the synthetic wait-for event loop.
> >
> > I am inclining towards solution (3) as it seems to be the cleanest and simplest one, I don't see why the synthetic wait-for event loop is needed in the first place.
>
> The main reason why a synthetic wait-for is used to open the drop-down is that all triggers are 'hidden'

True, even if I removed the synthetic wait-for loop, I would have to mask out the triggers and than restore them on drop-down exit. This would not solve the problem. I think I will have to either resolve the UI events order or track the currently opened overlay.

#14 - 05/11/2016 08:56 AM - Hynek Cihlar

3062a revision 11032 resolves all the issues. The branch has passed GUI regression testing, ChUI regression testing is in progress. Please review.

#15 - 05/13/2016 02:10 PM - Hynek Cihlar

3062a revision 11038 passed ChUI and GUI regression tests and was rebased against trunk revision 11028. Please review.

#16 - 05/13/2016 02:57 PM - Greg Shah

- Subject changed from COMBO-BOX steels focus when its drop-down active to COMBO-BOX steals focus when its drop-down active

Code Review Task Branch 3062a Revision 11038

The changes look good. I did commit some minor comment cleanup, but otherwise it is in good shape.

Eugenie and Constantin: do either of you have any concerns? I plan for Hynek to merge this to trunk if you don't see anything wrong.

#17 - 05/13/2016 03:13 PM - Constantin Asofiei

Greg Shah wrote:

> Eugenie and Constantin: do either of you have any concerns?

The changes are OK for me.

#18 - 05/13/2016 03:27 PM - Eugenie Lyzenko

Greg Shah wrote:

> Eugenie and Constantin: do either of you have any concerns? I plan for Hynek to merge this to trunk if you don't see anything wrong.

I'm checking out the 3062a to see the difference. It needs some time. So if you have just diffs comparing to trunk please publish here. This can decrease the time for me to review and answer.

#19 - 05/13/2016 04:06 PM - Eugenie Lyzenko

My only concern is related to removed logic for leave-and-repost mouse clicks approach. You need to be sure the updated code has respective functionality implemented instead of removed one. For example when drop-down is opened clicking mouse outside drop-down should generate mouse event **AFTER** drop-down closed. But only for cases when the mouse is not over the part of the drop-down itself.

On the other hand clicking on the drop-down scroll-bar should not cause the drop-down to disappear. Complete testing for these cases must be performed to avoid regressions in this very sensitive area.

#20 - 05/13/2016 05:20 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> My only concern is related to removed logic for leave-and-repost mouse clicks approach. You need to be sure the updated code has respective functionality implemented instead of removed one. For example when drop-down is opened clicking mouse outside drop-down should generate mouse event **AFTER** drop-down closed. But only for cases when the mouse is not over the part of the drop-down itself.

Clicking outside a drop-down causes it to be closed. Is there any other use case you had in mind?

> On the other hand clicking on the drop-down scroll-bar should not cause the drop-down to disappear. Complete testing for these cases must be performed to avoid regressions in this very sensitive area.

I have tested scrolling and other mouse interactions with the combo-box widget.

#21 - 05/13/2016 06:04 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Eugenie Lyzenko wrote:
>
> > My only concern is related to removed logic for leave-and-repost mouse clicks approach. You need to be sure the updated code has respective functionality implemented instead of removed one. For example when drop-down is opened clicking mouse outside drop-down should generate mouse event **AFTER** drop-down closed. But only for cases when the mouse is not over the part of the drop-down itself.
>
> Clicking outside a drop-down causes it to be closed. Is there any other use case you had in mind?

Imagine you have opened drop-down and press other button. The reaction for this single click must be:
- drop-down to be closed
- then the other button receives the mouse click event

I mean the second event should not be ignored.

#22 - 05/13/2016 06:10 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Imagine you have opened drop-down and press other button. The reaction for this single click must be:
> - drop-down to be closed
> - then the other button receives the mouse click event

Yes, this works as expected.

#23 - 05/13/2016 06:24 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Yes, this works as expected.

OK. I have no other concerns for now.

#24 - 05/15/2016 12:48 PM - Hynek Cihlar

3062a rebased against trunk rev 11029 and now at rev 11041. Greg, please let me know when it can be merged to trunk.

#25 - 05/16/2016 09:15 AM - Greg Shah

Please go ahead and merge to trunk.

#26 - 05/16/2016 09:43 AM - Hynek Cihlar

3062a merged to trunk as revision 11030 and archived.

#27 - 05/16/2016 09:43 AM - Hynek Cihlar

- % Done changed from 0 to 100

#28 - 05/16/2016 09:45 AM - Greg Shah

- Target version set to Milestone 16

- Status changed from WIP to Closed

#29 - 11/16/2016 12:22 PM - Greg Shah

- Target version changed from Milestone 16 to Cleanup and Stabilization for GUI