User Interface - Bug #3690

BUTTON widget reaction on SPACEBAR and ENTER keys

08/17/2018 05:04 PM - Eugenie Lyzenko

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Adrian Lungu	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Bug #3889: incorrect initial widget and screen va...	**WIP**
Related to User Interface - Feature #4837: Add javascript keyboard test appli...	**WIP**

History

#2 - 08/17/2018 05:11 PM - Eugenie Lyzenko

The issues to solve here:
1. Investigate and fix how the reaction on ENTER key implemented in 4GL.
2. Find out the alternative way to have key press/release event for SPACEBAR key. The problem is currently we have only key typed event for this key. The Java backend probably even does not emit press/release events for this case.
3. Investigate the button highlight behavior for modern Windows themes (8 and 10) and properly implement in FWD.

#3 - 08/17/2018 07:34 PM - Eugenie Lyzenko

- File button_hovered_color_4gl_Win8.jpg added

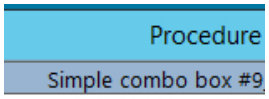
- File button_pressed_color_4gl_Win8.jpg added

Well, just retested with AW VM Windows8 system. Got different button colors for hovered and pressed states:
The hovered Windows8 button color: bee6fd (in Hex)
The pressed Windows8 button color: c4e5f6 (in Hex)

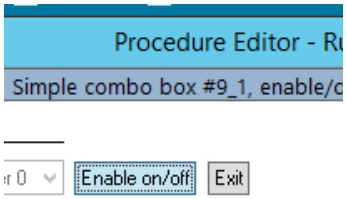
The screens was captured via PrnScr in Windows, paste into Windows Paint and save as 24-bit bmp. Unfortunately colors can be taken via video capture are not correct due to video rendering smoothing.

So the colors are different. The images:

Hovered:



Pressed:



#4 - 08/23/2018 07:48 AM - Hynek Cihlar

Also make sure the following visual behavior works correctly in FWD. When mouse is pressed over a button and mouse is hovered outside the button, the button should change to mouse-over highlighted state. In FWD, all highlights are removed from the button in this case. Also this behavior happens only in Windows 8 and 10 themes.

#5 - 08/23/2018 10:47 PM - Eugenie Lyzenko

Hynek Cihlar wrote:

> Also make sure the following visual behavior works correctly in FWD. When mouse is pressed over a button and mouse is hovered outside the button, the button should change to mouse-over highlighted state. In FWD, all highlights are removed from the button in this case. Also this behavior happens only in Windows 8 and 10 themes.

Please provide an example of what you mean. Is it about dotted rectangle inside button? Something else other?

#6 - 06/11/2019 12:24 PM - Greg Shah

- Start date deleted (08/17/2018)

- Assignee set to Adrian Lungu

#7 - 06/21/2019 07:17 AM - Adrian Lungu

- Status changed from New to WIP

Current behavior:
- If an action is defined "on choose of button" and in the Swing GUI interface we hold down space focusing that button, the action will incorrectly be fired multiple times, instead of only once at the release time. However if the action is defined "on space of button", the action should be fired multiple times.
- There was no Key Release event generated for any key, fact which is crucial for solving this bug

Solving strategy:
- Create a new kind of event, namely KeyRealese.
- To avoid possible new bugs and for performance reasons, restrict the generation of such events only for space-bar keystrokes.
- In the event queue, keep only the key released events related to a ButtonGuiImpl.
- To reproduce the exact behavior of Progress, generate only one "choose event" only when the space-bar is released. (not for every space-bar typed event)
- Avoid triggering action for key released events.
- At a space-bar press and release action, there should be 2 generated user events (space-bar key typed and space-bar key released) and one

artificial event (choose pressed), in this order. As a matter of fact, if space-bar will be hold for a little longer and released, there should be multiple space-bar key typed events generated, but still only one space-bar key released event and only one choose pressed event in the end.

#8 - 06/21/2019 08:15 AM - Greg Shah

Good plan.

#9 - 06/21/2019 08:30 AM - Adrian Lungu

- The bug was fixed following all the steps in the solving strategy. The <<on choose of button>> works as expected for space-bar pressing. However, there is still a problem avoiding the triggering of the event at the space-bar key release which duplicates the <<on " " of button>> action execution.

- To sum the things up, in case the action is defined "on space of a button", than there will be at least 2 space-bar related events (one typed and one release), causing at least 2 action executions.

- What have been already tried:

```
- Consuming the space-bar key released event at this stage to avoid execution. (doesn't seem to have an impact
)
```

```
- Directly emitting a choose event instead of a space-bar key release  (doesn't work as it is not a real key,
and the event is eliminated afterwards, before reaching the triggering)
```

```
- Tried to alter the event's data so it doesn't reassemble a space-bar event at all. (doesn't seem to change t
he behavior, moreover it seems the action is pushed into a queue beforehand)
```

```
- Tried to avoid using the ActionEvent at all, but no remarkable results.
```

#10 - 06/21/2019 08:56 AM - Greg Shah

- Subject changed from BUTTON widget raction on SPACEBAR and ENTER keys to BUTTON widget reaction on SPACEBAR and ENTER keys

Hynek: Please review and comment.

#11 - 06/21/2019 09:21 AM - Hynek Cihlar

Adrian, please post your changes here.

#12 - 06/21/2019 09:40 AM - Greg Shah

Actually, please create a task branch 3690a and put the changes there.

#13 - 06/21/2019 12:27 PM - Adrian Lungu

Branch 3690a created.
Changes committed to branch.

#14 - 06/21/2019 07:09 PM - Hynek Cihlar

- File dont_trigger_release_key.diff added

Adrian Lungu wrote:

> - The bug was fixed following all the steps in the solving strategy. The <<on choose of button>> works as expected for space-bar pressing. However, there is still a problem avoiding the triggering of the event at the space-bar key release which duplicates the <<on " " of button>> action execution.

The problem is that the triggers are executed before ButtonGuiImpl.handleBasicEvents where you try to stop processing of the release key event.

Since we never want to trigger the release key events, how about to protect the trigger execution with the release key event type? See the attached diff. If you choose this solution, please make sure the case is well documented in the code.

#15 - 06/21/2019 07:12 PM - Hynek Cihlar

There is another related issue. In OpenEdge when both enter and chose triggers for a button are defined, pressing Enter key only triggers enter, while in FWD we trigger both, enter and chose.

#16 - 07/23/2019 10:02 AM - Greg Shah

- Related to Bug #3889: incorrect initial widget and screen values for combo boxes and selection lists added

#17 - 07/23/2019 10:05 AM - Greg Shah

Adrian, do the changes in 3690a revision 11321 represent your fixes for the code review items? If so, is it ready for review again?

#18 - 07/23/2019 10:17 AM - Adrian Lungu

Yes, revision 11321 of 3690a fixes the points mentioned and it is ready for review. Last reviewed revision was 11320.

#19 - 07/23/2019 04:23 PM - Hynek Cihlar

Code review 3690a revisions 11321 and 11322. I'm good with the changes. Just please remove the commented out code in 11322 (unless it's required to stay in for clarity).

#20 - 07/24/2019 08:35 AM - Adrian Lungu

Indeed, revision 11321 was meant to fully fix bug #3690, while revision 11322 is still work in progress. There should be done more tests to ensure well functionality of the screen value of the combo box. (#3889)

#21 - 07/24/2019 10:00 AM - Greg Shah

I think it makes sense to split these changes. Please merge the 11320 and 11321 revisions into 4045a.

Then create a new 3889a branch and move revision 11322 into there. Branch 3690a can be archived as dead at that point.

#22 - 07/31/2019 07:29 AM - Adrian Lungu

- % Done changed from 0 to 100

Moved revision 11322 onto a new branch 3889a. Revisions 11320 and 11321 were merged into 4045a. Branch 3690a is no longer needed at this stage.

#23 - 08/01/2019 02:59 PM - Sergey Ivanovskiy

Adrian Lungu wrote:

> Moved revision 11322 onto a new branch 3889a. Revisions 11320 and 11321 were merged into 4045a. Branch 3690a is no longer needed at this stage.

The rev 11370 branch 4045a has a regression that the dialog message buttons are not sensitive to mouse clicking. Please test simple alert message box.

#24 - 08/02/2019 12:03 PM - Adrian Lungu

I have fixed the lack of response from the buttons inside the alert box. This was due to the fact that no ActionEvent was created on a choose key input event. This kind of event is also generated by the mouse click, which explains why there was no functionality attached to the button.

I have committed the fix in revision 11372 of branch 4045a.

#25 - 08/20/2019 02:13 PM - Sergey Ivanovskiy

There are failed CHUI regression tests with 4045a related to this issue. Pressing Space key is not properly handled. Please run this testcase and press SPACE key

```
MESSAGE "Press Space or Enter or Click Me." VIEW-AS ALERT-BOX.
```

#26 - 08/20/2019 04:05 PM - Sergey Ivanovskiy

Sergey Ivanovskiy wrote:

> There are failed CHUI regression tests with 4045a related to this issue. Pressing Space key is not properly handled. Please run this testcase and press SPACE key
> [...]

It can be reproduced with chui_native client.

#27 - 08/20/2019 04:33 PM - Greg Shah

Sergey: Can you please prepare the fix for this?

#28 - 08/20/2019 06:15 PM - Sergey Ivanovskiy

- File KeyInput.patch added

I can propose this diff for CHUI client.

#29 - 08/21/2019 01:44 AM - Sergey Ivanovskiy

Key released events are not detected by Linux terminal using ncurses and please correct if it is not true. Thus the idea to use key released for this case should be reworked. I just looked at this article https://blog.robertelder.org/detect-keyup-event-linux-terminal/ and googled this technical issue.

#30 - 08/21/2019 08:34 AM - Greg Shah

I don't object to the proposed change. I do have a question. Why is it happening now? Is this the first time we relied upon the key released event for processing?

Adrian: Please review.

#31 - 08/21/2019 09:02 AM - Sergey Ivanovskiy

Greg Shah wrote:

> I don't object to the proposed change. I do have a question. Why is it happening now? Is this the first time we relied upon the key released event for processing?

Yes, the key released event is used to solve #3690-7, isn't it?

#32 - 08/21/2019 09:53 AM - Adrian Lungu

Indeed, the key released event was designed especially for this. The changes look good to me as it conducts a valid behavior for the ChUI separately. I didn't had the chance to test enough the changes of #3690 on the ChUI, but the diff seem to fix the problem.

> Key released events are not detected by Linux terminal using ncurses and please correct if it is not true. Thus the idea to use key released for this case should be reworked. I just looked at this article https://blog.robertelder.org/detect-keyup-event-linux-terminal/ and googled this technical issue.

I wasn't aware of such aspect at the moment of implementation, as the tests were only done on the GUI. If this however stands, it shouldn't be a problem as long as the KEY_RELEASE event is actively used only in ButtonGuiImpl.handleBasicEvents and KeyInput.canProduceAction. While ButtonGuiImpl is not used in ChUI and KeyInput is aware of the ChUI (due to the diff in #3690-28), any related issue shouldn't appear anymore.

#33 - 08/21/2019 10:16 AM - Greg Shah

Sergey: Please commit the change to 4045a. It fixes a regression so it is OK to apply to that frozen branch.

#34 - 08/21/2019 10:39 AM - Sergey Ivanovskiy

Committed revision 11388 (4045a). I will start a new round of the chui regression tests for rev 11388 (4045a) if there are no objections.

#35 - 08/21/2019 12:26 PM - Eric Faulhaber

Sergey Ivanovskiy wrote:

> I will start a new round of the chui regression tests for rev 11388 (4045a) if there are no objections.

Yes, please do.

#36 - 08/24/2019 04:23 AM - Sergey Ivanovskiy

- File test-pause-1.p added

I have another regressed simple test case that doesn't work with GUI client.

```
MESSAGE "step 1".

PAUSE.

MESSAGE "step 2".

PAUSE.

MESSAGE "done".
```

If you press SPACE, then step 2 and done messages appear. I guess it is due for SPACE there is a released key event that occurs after the next business method PAUSE. is invoked. Debugging breaks this observation as it is related to the concurrency issue too.

#37 - 08/26/2019 09:24 AM - Sergey Ivanovskiy

Adrian Lungu wrote:

> Current behavior:
> - If an action is defined "on choose of button" and in the Swing GUI interface we hold down space focusing that button, the action will incorrectly be fired multiple times, instead of only once at the release time. However if the action is defined "on space of button", the action should be fired multiple times.
> - There was no Key Release event generated for any key, fact which is crucial for solving this bug

Solving strategy:
- Create a new kind of event, namely KeyRealese.
- To avoid possible new bugs and for performance reasons, restrict the generation of such events only for space-bar keystrokes.
- In the event queue, keep only the key released events related to a ButtonGuiImpl.
- To reproduce the exact behavior of Progress, generate only one "choose event" only when the space-bar is released. (not for every space-bar typed event)
- Avoid triggering action for key released events.
- At a space-bar press and release action, there should be 2 generated user events (space-bar key typed and space-bar key released) and one artificial event (choose pressed), in this order. As a matter of fact, if space-bar will be hold for a little longer and released, there should be multiple space-bar key typed events generated, but still only one space-bar key released event and only one choose pressed event in the end.

Adrian, could you provide references for 4GL tests that proved this behaviour? I need them in order to check them with the PAUSE fixes.

#38 - 08/26/2019 04:45 PM - Sergey Ivanovskiy

Are there tests for READKEY.? Are there evidences that 4GL uses key released events for its run-time implementation?

#39 - 08/26/2019 04:56 PM - Sergey Ivanovskiy

For an example, if we press SPACE for a while, then depending on the OS drivers, the SPACE typed events will be generated as repeated events.

#40 - 08/26/2019 04:58 PM - Sergey Ivanovskiy

Committed revision 11391 (4045a) should fix #3690-36.

#41 - 08/26/2019 05:19 PM - Greg Shah

Sergey: Please explain the idea behind this change. The safety and correctness of the change is not obvious to me.

Adrian: Please respond to the questions above. This regression needs to be resolved with priority.

#42 - 08/27/2019 01:28 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Sergey: Please explain the idea behind this change. The safety and correctness of the change is not obvious to me.

Please look at TypeAHead.this.dequeueEvent() and EventManager.eventFromKey. It follows that TypeAHead.getKeystroke(int msec, boolean honorServerEvents) returns null for key released events when we press and release SPACE key, but
the run-time implementation of ThinClient.waitForEvents depends on key events. Thus these changes fixed this run-time implementation of ThinClient.waitForEvents for the case when SPACE key is pressed and then released.
Committed revision 11392 (4045a) fixed the same issue for READKEY. That can be observed with this simple test

```
MESSAGE "step 1".

PAUSE.

MESSAGE "step 2".

PAUSE.
```

```
MESSAGE "step 3".

READKEY.

MESSAGE "step 4".

READKEY.

MESSAGE "done".
```

I think that the changes, rev. 11391 and rev. 11392, are safe. In any case we should run CHUI regression tests again.

#43 - 08/27/2019 03:53 AM - Adrian Lungu

Firstly, related to #3690-37:

> Adrian, could you provide references for 4GL tests that proved this behavior? I need them in order to check them with the PAUSE fixes.

I will attach here an archive with some tests I used. Regarding specifically to the question, choose.p and space.p are the simple tests which reflect the behavior of holding the space-bar down with triggers on choose/space. However I see that #3690-42 is a fix for the issue mentioned, but the archive may still prove of good use for quick testing.

> Are there tests for READKEY.? Are there evidences that 4GL uses key released events for its run-time implementation?

> For an example, if we press SPACE for a while, then depending on the OS drivers, the SPACE typed events will be generated as repeated events.

I am not aware of any testcase including READKEY. Now, it is not clear if the run-time implementation of 4GL does explicitly use the KEY_RELEASE event or it is just an OS behavior. However, Linux drivers does send multiple events of KEY_TYPED/KEY_PRESSED if the key is hold.

It looked like the KEY_RELEASE event was the most natural way to solve the bug. Despite the huge implication of Key Events in the application, I retained the generation of KEY_RELEASE events just for space-bars and triggers are not fired for such events. Basically for now, these are dummies used for generating choose events or repaint. At this stage, any other use of KEY_RELEASE should be either ignored or converted into a more convenient event.

#44 - 08/27/2019 03:54 AM - Adrian Lungu

- File 3690.tar.gz added

#45 - 08/27/2019 07:54 AM - Sergey Ivanovskiy

Adrian, thank you for these tests. If we hold some key down on the keyboard, then auto repeats are generated for this pressed key. This case should be taken into account when we consider 4GL behaviour.

#46 - 08/27/2019 09:21 AM - Greg Shah

Sergey Ivanovskiy wrote:

> Adrian, thank you for these tests. If we hold some key down on the keyboard, then auto repeats are generated for this pressed key. This case should be taken into account when we consider 4GL behaviour.

Are there additional deviations between FWD and the 4GL?

#47 - 08/27/2019 09:36 AM - Greg Shah

> Please look at TypeAHead.this.dequeueEvent() and EventManager.eventFromKey. It follows that TypeAHead.getKeystroke(int msec, boolean honorServerEvents) returns null for key released events when we press and release SPACE key, but the run-time implementation of ThinClient.waitForEvents depends on key events.

I've reviewed this code as well as EventManager.getKeyEvent() and KeyInput. It is not clear to me that a key released event will be returned as null. Can you explain it further?

My other concern: if we ever return null from TypeAhead in other cases besides KEY_RELEASED, then this new code may cause unexpected behavior. It is not obvious that null is only ever a KEY_RELEASED case.

#48 - 08/27/2019 09:41 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Sergey Ivanovskiy wrote:
>
> > Adrian, thank you for these tests. If we hold some key down on the keyboard, then auto repeats are generated for this pressed key. This case should be taken into account when we consider 4GL behaviour.
>
> Are there additional deviations between FWD and the 4GL?

I don't understand now why the action for SPACE is triggered on release key event. According to these tests if we press SPACE on the button (ref. space.p), then the corresponding action is triggered one time.

#49 - 08/27/2019 09:42 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Please look at TypeAHead.this.dequeueEvent() and EventManager.eventFromKey. It follows that TypeAHead.getKeystroke(int msec, boolean honorServerEvents) returns null for key released events when we press and release SPACE key, but the run-time implementation of ThinClient.waitForEvents depends on key events.

I've reviewed this code as well as EventManager.getKeyEvent() and KeyInput. It is not clear to me that a key released event will be returned as null. Can you explain it further?

My other concern: if we ever return null from TypeAhead in other cases besides KEY_RELEASED, then this new code may cause unexpected behavior. It is not obvious that null is only ever a KEY_RELEASED case.

Adrian, please explain this code.

#50 - 08/27/2019 09:50 AM - Sergey Ivanovskiy

Sergey Ivanovskiy wrote:

> Greg Shah wrote:
>
> > Please look at TypeAHead.this.dequeueEvent() and EventManager.eventFromKey. It follows that TypeAHead.getKeystroke(int msec, boolean honorServerEvents) returns null for key released events when we press and release SPACE key, but the run-time implementation of ThinClient.waitForEvents depends on key events.
>
> I've reviewed this code as well as EventManager.getKeyEvent() and KeyInput. It is not clear to me that a key released event will be returned as null. Can you explain it further?
>
> My other concern: if we ever return null from TypeAhead in other cases besides KEY_RELEASED, then this new code may cause unexpected behavior. It is not obvious that null is only ever a KEY_RELEASED case.
>
> Adrian, please explain this code.

Sorry, yes, the null value for READKEY and PAUSE statement seems can be from key released events. I don't know.

#51 - 08/27/2019 10:04 AM - Sergey Ivanovskiy

OK. I understand now that if we press SPACE, then the triggered action is executed at the moment when this key is released. (ref: space.p) That is why the key released is used now.

#52 - 08/27/2019 10:35 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Please look at TypeAHead.this.dequeueEvent() and EventManager.eventFromKey. It follows that TypeAHead.getKeystroke(int msec, boolean honorServerEvents) returns null for key released events when we press and release SPACE key, but the run-time implementation of ThinClient.waitForEvents depends on key events.

> I've reviewed this code as well as EventManager.getKeyEvent() and KeyInput. It is not clear to me that a key released event will be returned as null. Can you explain it further?
>
> My other concern: if we ever return null from TypeAhead in other cases besides KEY_RELEASED, then this new code may cause unexpected behavior. It is not obvious that null is only ever a KEY_RELEASED case.

I found another solution that fixed EventManager.eventFromKey for released keys events. Please review the committed rev. 11393. It should overcome this case that you pointed.

#53 - 08/27/2019 10:48 AM - Greg Shah

Yes, this is **much** better. The intention is clear.

Please do the following:

- run ChUI regression testing
- check the standalone testcases related to this issue
- check the big GUI app smoke tests

#54 - 08/28/2019 03:56 AM - Sergey Ivanovskiy

Greg Shah wrote:

- run ChUI regression testing

CHUI main regression tests passed for rev 11393 (4045a) in one batch run and two single runs.

- check the standalone testcases related to this issue

Seems to be OK (passed).

- check the big GUI app smoke tests

The calculated values for the smoke test 1 differs from the one we have on the video. I can't check that all calculated values are OK in these tests.

#55 - 08/29/2019 02:21 PM - Sergey Ivanovskiy

I didn't think about LASTKEY functionality. Does it work properly? Released keys events can affect this value even if the CHUI tests were passed.

#56 - 08/29/2019 02:23 PM - Greg Shah

Please make changes to exclude LASTKEY from effects by key release.

#57 - 08/29/2019 02:59 PM - Sergey Ivanovskiy

Greg Shah wrote:

> Please make changes to exclude LASTKEY from effects by key release.

I am looking into lastevent/ for test cases and lastkey_reset.p seems to be important.

#58 - 03/16/2020 10:21 AM - Adrian Lungu

I recently found a bug related to the issue here - regarding the web interface. The web driver doesn't emit release events, which means the widgets

which use the released keys, won't work properly. By now, the button is the only widget which is sensible to those (that is why typing space on the button will only paint it without firing the action).

At first glance, a change similar to the one done in the swing driver should be adopted (generate release key events for space-bar). However, the web driver doesn't seem to be that flexible, letting the release events propagate to the server. In fact, it looks like a more complex system is built around the onkeyup and onkeydown events.

I logged the type of event which arrives at sendKey(key, evt) - the high end method for sending key events to the server in p2j.keyboard.js.
For a simple space-bar type on a button:

```
keyup
```

For a **single** longer press and release of the space-bar on a button - which seems strange:

```
keydown
keydown
keyup
keydown
keydown
keydown
keyup
keydown
keyup
keydown
keydown
keydown
keydown
keydown
keyup
```

Yet, I am not sure how the key events are handled in the web driver. But from this slim investigation, it seems that major changes should be done in order to:

1. send 2 events when typing a key (press and release / keyup and keydown)
2. understand why "random" events are sent to the server when holding down key (both keyup and keydown)

Even though the KEY_RELEASED event is not a progress event, it should be generated in order to properly reproduce the default behaviors of Windows widgets. Also, I am not sure now if this is the **only** case in which this is required - there may be also other widgets with Windows implementation which makes use of released keys.

Ultimately, I can't say this is a high priority issue yet - but it can be a reference point when dealing with KEY_RELEASED dependent widgets.

#59 - 03/16/2020 10:32 AM - Greg Shah

Whatever solution is created, please minimize the number of events forwarded to the FWD client (Java side). Optimally, we would be able to go into one location in the javascript code to enable key release support for a particular key.

#60 - 03/16/2020 10:39 AM - Sergey Ivanovskiy

Adrian Lungu wrote:

> I recently found a bug related to the issue here - regarding the web interface. The web driver doesn't emit release events, which means the widgets which use the released keys, won't work properly. By now, the button is the only widget which is sensible to those (that is why typing space on the button will only paint it without firing the action).
>
> At first glance, a change similar to the one done in the swing driver should be adopted (generate release key events for space-bar). However, the web driver doesn't seem to be that flexible, letting the release events propagate to the server. In fact, it looks like a more complex system is built around the onkeyup and onkeydown events.
>
> I logged the type of event which arrives at sendKey(key, evt) - the high end method for sending key events to the server in p2j.keyboard.js.
> For a simple space-bar type on a button:
> [...]
> For a **single** longer press and release of the space-bar on a button - which seems strange:
> [...]
>
> Yet, I am not sure how the key events are handled in the web driver. But from this slim investigation, it seems that major changes should be done in order to:
>
> 1. send 2 events when typing a key (press and release / keyup and keydown)
> 2. understand why "random" events are sent to the server when holding down key (both keyup and keydown)
>
> Even though the KEY_RELEASED event is not a progress event, it should be generated in order to properly reproduce the default behaviors of Windows widgets. Also, I am not sure now if this is the **only** case in which this is required - there may be also other widgets with Windows implementation which makes use of released keys.
>
> Ultimately, I can't say this is a high priority issue yet - but it can be a reference point when dealing with KEY_RELEASED dependent widgets.

It seems that the repeats keys are generated when you press and hold the key.

#61 - 03/16/2020 12:12 PM - Sergey Ivanovskiy

It seems that keyup browser's event means that a key is released. Is it correct to change this part of the js web client part? This part has special program to test keystokes testcases/uast/keyboards/web.

#62 - 03/16/2020 12:29 PM - Sergey Ivanovskiy

Adrian Lungu wrote:

> Yet, I am not sure how the key events are handled in the web driver. But from this slim investigation, it seems that major changes should be done in order to:
>
> 1. send 2 events when typing a key (press and release / keyup and keydown)
> 2. understand why "random" events are sent to the server when holding down key (both keyup and keydown)

It is not random. Please provide what is the test case that proves that the events are generated incorrectly.

> Even though the KEY_RELEASED event is not a progress event, it should be generated in order to properly reproduce the default behaviors of Windows widgets. Also, I am not sure now if this is the **only** case in which this is required - there may be also other widgets with Windows implementation which makes use of released keys.

Please handle this on the java web client side.

#63 - 03/16/2020 01:15 PM - Greg Shah

> > Even though the KEY_RELEASED event is not a progress event, it should be generated in order to properly reproduce the default behaviors of Windows widgets. Also, I am not sure now if this is the only case in which this is required - there may be also other widgets with Windows implementation which makes use of released keys.
>
> Please handle this on the java web client side.

I don't want to send a large number of additional events up to Java if we only need a very small number of cases. For example, if we only need to send keyUp for space, then we would not send keyUp events for all keys. Otherwise we are creating a new performance issue.

#64 - 03/17/2020 05:10 AM - Adrian Lungu

- File keyupdown.p added

- File keyupdown.png added

- File diff.txt added

Greg Shah wrote:

> I don't want to send a large number of additional events up to Java if we only need a very small number of cases. For example, if we only need to send keyUp for space, then we would not send keyUp events for all keys. Otherwise we are creating a new performance issue.

In the Swing client, this is what is happening (generating KEY_RELEASED only for space-bar) - this will also be done in the web client.

Sergey Ivanovskiy wrote:

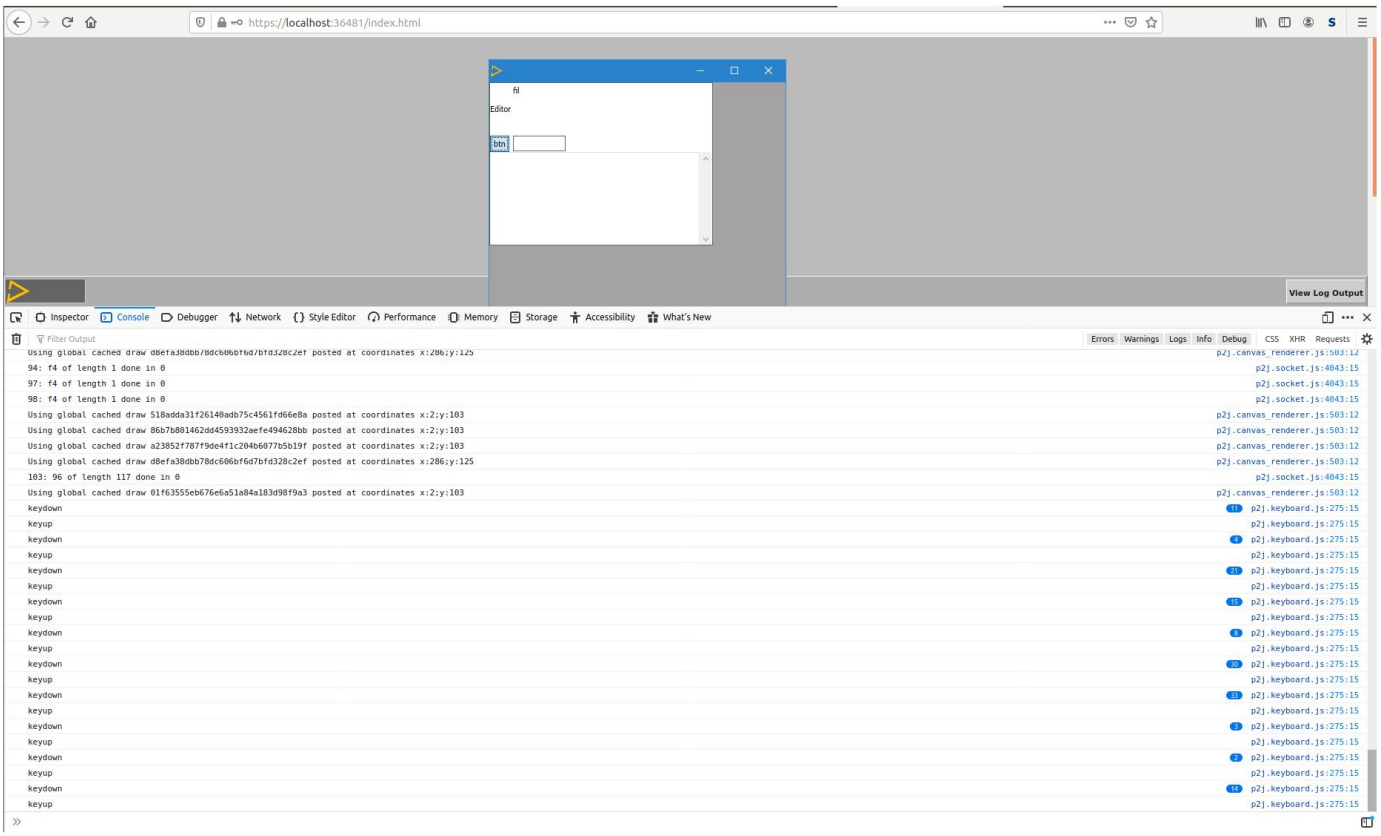
> It seems that keyup browser's event means that a key is released. Is it correct to change this part of the js web client part? This part has special program to test keystokes testcases/uast/keyboards/web.

I will look into these tests. Eventually, I think this is the correct way to start generating KEY_RELESED, if not the only way - by changing the semantics of onkeyup and onkeydown. However, is it to early to say right now.

Sergey Ivanovskiy wrote:

> It is not random. Please provide what is the test case that proves that the events are generated incorrectly.

I attached the example I am using - in fact you can use any example which has a button. I attached the diff to show where I am logging the events. For a button is easier to debug as there are no other polluting messages. I get the following result (**on a single hard long press of the space-bar on the button**):



#65 - 03/17/2020 05:16 AM - Adrian Lungu

Now I see that this behavior can't be reproduced on Chrome. Maybe it is a bug in Firefox regarding long key presses?

#66 - 03/17/2020 06:11 AM - Adrian Lungu

After checking the GUIKeyboardReader, I see there is the following code which triggers that additional keyups:

```
if (evt.keyCode === keys.IME_PROCESSKEY)
{
    var keyUpEvt = p2j.createKeyEvent("keyup", false, true, null, evt.ctrlKey,
       evt.altKey, evt.shiftKey, evt.metaKey, repeatCode, 0);

    // check if the keyboard event is created successfully
    if (keyUpEvt)
    {
        onkeyup(keyUpEvt);
    }

    p2j.consumeEvent(evt);
    return;
}
```

I couldn't find very strong documentation regarding keys.IME_PROCESSKEY (229 key code), but I guess this was the reason why Firefox and Chrome showed different behaviors. (Firefox fires these events - for me right now seem random, but maybe they are not). With this observation, this means the space-bar alone is working properly: sending continuous keydown signals and one keyup signal at the end.

#67 - 03/17/2020 10:54 AM - Adrian Lungu

Committed fix in 4231b/rev. 11373. The changes were minimal - only the space-bar generates additional released events.

#68 - 03/17/2020 11:11 AM - Greg Shah

Code Review Task Branch 4231b Revision 11372

The changes seem good to me.

Sergey: Any objections?

#69 - 03/17/2020 11:43 AM - Sergey Ivanovskiy

It seems the changes in 4231b Revision 11372 are good.

#70 - 05/26/2020 05:35 AM - Adrian Lungu

- Status changed from WIP to Review

#71 - 05/26/2020 06:44 AM - Greg Shah

- Status changed from Review to Test

#72 - 06/20/2020 12:04 PM - Greg Shah

- Status changed from Test to Closed

Task branch 4231b has been merged to trunk as revision 11347.

#73 - 08/01/2020 03:43 PM - Sergey Ivanovskiy

This is a simple test page for keyboard reader for branch 3821c. Please place them in 3821c project root with these two js: keystrokes.js and keyboard_testcases.js and open this html page in the browser. There are several automatic tests and manual gui test, click on the "GUI Manual Testing" button and click on the input field, then press any combination and look at the Progress 4GL generated code. In this branch pressing SPACE produces two keycodes 32. It is a real regression and it isn't present in the old versions.

#74 - 08/01/2020 03:45 PM - Sergey Ivanovskiy

I didn't observe that it was due to the changes related SPACE processing. The browser generates SPACE key down, SPACE key pressed and SPACE key released. Does it make sense to send SPACE key DOWN when it was down and to send SPACE key RELEASED when it was released?
Finally, ALT+SPACE produces two events. Is it correct?

#75 - 08/01/2020 03:46 PM - Sergey Ivanovskiy

- File test.html added

- File keystrokes.js added

- File keyboard_testcases.js added

#76 - 08/03/2020 08:27 AM - Greg Shah

Sergey: I think we should have this test app built into FWD and expose it for testing purposes. Do we want to expose it on the FWD server or client? The client might be best because we could access it via menu item on virtual desktop and it would never conflict with the server-side customer web apps.

#77 - 08/03/2020 08:48 AM - Adrian Lungu

Sergey Ivanovskiy wrote:

> I didn't observe that it was due to the changes related SPACE processing. The browser generates SPACE key down, SPACE key pressed and SPACE key released. Does it make sense to send SPACE key DOWN when it was down and to send SPACE key RELEASED when it was released?
> Finally, ALT+SPACE produces two events. Is it correct?

The provided application proves to be very useful for this kind of matters. The KEY_RELEASED event is generated only by SPACE and used only by the button widget. Other widgets will instantly consume the event and ignore it. Other keys do not generate KEY_RELEASED event.

Related to the DOWN and RELEASED events, these are needed for esthetics at first. The button should be drawn as pressed when DOWN is processed. The button should be redrawn when released and eventually fire a trigger. Also, note that holding the SPACE key will generate several 32 key codes (mostly DOWN events).

Related to ALT+SPACE, I guess the driver should avoid generating release events when the SPACE is part of a key code with modifiers. I can't identify a testcase to reproduce a bug exploiting this. I will fix double events for ALT+SPACE at the driver level.

#78 - 08/03/2020 09:11 AM - Sergey Ivanovskiy

Adrian Lungu wrote:

> Sergey Ivanovskiy wrote:
>
> > I didn't observe that it was due to the changes related SPACE processing. The browser generates SPACE key down, SPACE key pressed and SPACE key released. Does it make sense to send SPACE key DOWN when it was down and to send SPACE key RELEASED when it was released?
> > Finally, ALT+SPACE produces two events. Is it correct?
>
> The provided application proves to be very useful for this kind of matters. The KEY_RELEASED event is generated only by SPACE and used only by the button widget. Other widgets will instantly consume the event and ignore it. Other keys do not generate KEY_RELEASED event.
>
> Related to the DOWN and RELEASED events, these are needed for esthetics at first. The button should be drawn as pressed when DOWN is processed. The button should be redrawn when released and eventually fire a trigger. Also, note that holding the SPACE key will generate several 32 key codes (mostly DOWN events).
>
> Related to ALT+SPACE, I guess the driver should avoid generating release events when the SPACE is part of a key code with modifiers. I can't identify a testcase to reproduce a bug exploiting this. I will fix double events for ALT+SPACE at the driver level.

What is the driver level? Please use for testing the test page attached in this Redmine thread #3690-75, #3690-73.

#79 - 08/03/2020 09:14 AM - Sergey Ivanovskiy

Greg Shah wrote:

> Sergey: I think we should have this test app built into FWD and expose it for testing purposes. Do we want to expose it on the FWD server or client? The client might be best because we could access it via menu item on virtual desktop and it would never conflict with the server-side customer web apps.

I used the standalone test page for testing what key codes are produced by p2j.keyboard.js. Please review the files attached in #3690-75. It seems it is not required to start the server and the web client to test what Progress 4GL keys are generated by js web client.

#80 - 08/03/2020 09:28 AM - Adrian Lungu

Removed key released events for SPACE with modifiers in 3821c/rev. 11430 as mentioned in #3690-77 to avoid ALT+SPACE double event creation.

#81 - 08/03/2020 09:31 AM - Adrian Lungu

Sergey Ivanovskiy wrote:

> What is the driver level? Please use for testing the test page attached in this Redmine thread #3690-75, #3690-73.

I was referring to WinKeyboardReader and p2j.keyboard.js. The test page shows that ALT+SPACE (1056) is now generated only once.

#82 - 08/03/2020 09:32 AM - Greg Shah

Sergey Ivanovskiy wrote:

> Greg Shah wrote:
>
> > Sergey: I think we should have this test app built into FWD and expose it for testing purposes. Do we want to expose it on the FWD server or client? The client might be best because we could access it via menu item on virtual desktop and it would never conflict with the server-side customer web apps.
>
> I used the standalone test page for testing what key codes are produced by p2j.keyboard.js. Please review the files attached in #3690-75. It seems it is not required to start the server and the web client to test what Progress 4GL keys are generated by js web client.

Yes, I understand. My point is that I want this built into FWD so we don't have to spend time trying to recreate this application in order to test. If you want to add this to the FWD server, that is OK but I worry that the URL can conflict with customer web URLs. So perhaps you should expose it with something very unique like "fwd_javascript_keyboard_test_application".

#83 - 08/04/2020 08:59 AM - Sergey Ivanovskiy

- Related to Feature #4837: Add javascript keyboard test application for the virtual desktop added

Files

button_hovered_color_4gl_Win8.jpg	15.5 KB	08/17/2018	Eugenie Lyzenko
button_pressed_color_4gl_Win8.jpg	23.5 KB	08/17/2018	Eugenie Lyzenko

dont_trigger_release_key.diff	845 Bytes	06/21/2019	Hynek Cihlar
KeyInput.patch	1.35 KB	08/20/2019	Sergey Ivanovskiy
test-pause-1.p	87 Bytes	08/24/2019	Sergey Ivanovskiy
3690.tar.gz	573 Bytes	08/27/2019	Adrian Lungu
keyupdown.png	169 KB	03/17/2020	Adrian Lungu
keyupdown.p	360 Bytes	03/17/2020	Adrian Lungu
diff.txt	907 Bytes	03/17/2020	Adrian Lungu
test.html	5.38 KB	08/01/2020	Sergey Ivanovskiy
keystrokes.js	11 KB	08/01/2020	Sergey Ivanovskiy
keyboard_testcases.js	29.7 KB	08/01/2020	Sergey Ivanovskiy