User Interface - Feature #2480

Feature # 2252 (Closed): implement GUI client support

Feature # 2446 (Closed): implement BUTTON and IMAGE GUI widgets (runtime and conversion support)

add support for button and image attributes and options

01/07/2015 09:01 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Eugenie Lyzenko	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to User Interface - Feature #3332: Add support for load offset and lo...	**Closed**
Related to User Interface - Bug #3504: sizing and centering issues with butto...	**New**

History

#1 - 01/07/2015 10:30 AM - Greg Shah

- Subject changed from add support for button and image attributes/methods to add support for button and image attributes and options

Some of these attributes are available for both button and image (but no other widgets). Other attributes are available only for button or only for image. We DO NOT want to add any of these to CommonWidget. Instead, we will create 3 new interfaces:

- ButtonInterface - only ButtonWidget implements this and it has all the attributes/methods that are unique to button
- ImageInterface - only ImageWidget implements this and it has all the attributes/methods that are unique to image
- ImageSupport - both ButtonWidget and ImageWidget implement this and it has all the attributes/methods that are only shared by image and button

Each of these will need new unwrap methods in handle.java AND the addition of those interfaces to the list in HandleCommon. You can look at the EditorInterface as an example of how we are doing this.

The reason we are doing this is because for attributes/methods that are very specific to a small widget list (or just one widget), it becomes very messy to add all the support to CommonWidget and GenericWidget, just to "block" access to the methods. It is awkward and make those classes very big, very confusing and it is much more effort to maintain over time. The original implementation was a bad decision that I want to eliminate. We don't have the time to do this all at once, but over time while you are working on a particular widget-specific interface, you can feel free to move things from CommonWidget and clean it up a bit "as we go".

Any attributes/methods that are shared by MOST or ALL widgets, will stay in CommonWidget.

From now on, all attributes and methods that are not in system handles will be defined in the load_descriptors function of methods_attributes.rules. This is a newer, cleaner way to add full support for a method or attribute with just a single line of code. It is thus simpler and less work to encode these.

For some reason, the 4GL sometimes defines widget options that are the negative option to an attribute. For example, you can specify NO-FILL option for a rectangle widget, but the attribute is FILLED. For historical reasons (since we originally implemented widget options long before we made method and attribute support), we originally implemented the negative options as their own configuration value. We no longer want to do that. If a configuration option is the negative of a valid attribute (prove it by making sure that setting the option changes the attribute value), then we should simply use the attribute setter and pass false, instead of creating a separate setter for the negative option. In the NO-FILL example, we would NOT create a setNoFill() and instead we would emit setFilled(false).

Button Widget Options

DEFAULT (needs client-side runtime implementation)
FLAT-BUTTON (when used as an option it can only be specified following NO-FOCUS and in the tree it will be the child of NO-FOCUS)
IMAGE-DOWN
IMAGE-INSENSITIVE
IMAGE-UP
NO-FOCUS

Attributes

AUTO-ENDKEY (button)
AUTO-GO (button)
CONVERT-3D-COLORS (button and image)
DEFAULT (button)
FLAT-BUTTON (button)
IMAGE (button and image)
IMAGE-DOWN (button)
IMAGE-INSENSITIVE (button)
IMAGE-UP (button)
NO-FOCUS (button)
STRETCH-TO-FIT (image)
RETAIN-SHAPE (image)
TRANSPARENT (image)

#2 - 02/25/2015 08:55 AM - Eugenie Lyzenko

The following attributes/options are left to b implemented/verified as working if already implemented:

Button options:

DEFAULT (needs client-side runtime implementation)
FLAT-BUTTON (when used as an option it can only be specified following NO-FOCUS and in the tree it will be the child of NO-FOCUS)
NO-FOCUS

Attributes:

AUTO-ENDKEY (button)
AUTO-GO (button)
DEFAULT (button)
FLAT-BUTTON (button)
NO-FOCUS (button)

#3 - 02/27/2015 05:18 PM - Eugenie Lyzenko

- File evl_upd20150227a.zip added

The update contains conversion support and runtime start point starting point for NO-FOCUS and FLAT-BUTTON options/attributes for button widget. DEFINE BUTTON case and standalone attribute setting including. Also conversion works for dynamically created buttons.

The change to note for this update is renaming ImageButtonSupport.java interface to ButtonInterface.java. This is done because not only image related methods are specific to button widget, there are other attributes that are used only for button widget. So I guess it will be more verbose to have more generic interface class for this case(instead of defining two interfaces, one for image related stuffs, another - for rest).

The runtime implementation for NO-FOCUS/FLAT-BUTTON is still in progress.

#4 - 03/03/2015 09:33 PM - Eugenie Lyzenko

- File evl_upd20150303a.zip added

This update for review contains runtime implementation for NO-FOCUS/FLAT-BUTTON. It is required to do more testing for buttons with images embedded. The other case is working as in 4GL.

Will continue tomorrow.

#5 - 03/06/2015 12:32 PM - Eugenie Lyzenko

- File evl_upd20150306a.zip added

This update for review has completed runtime implementation for NO-FOCUS/FLAT-BUTTON client part plus fixed newly discovered features for usual buttons. During testing I've found the following features affecting button drawing:

```
Flat/not flat
Focused/not focused
Image based/not image based
Having down image/not having down image
```

All these options can be mixed in arbitrary combinations and will produce a bit different drawing layout. That's why the ButtnGuiImpl.java looks a bit overloaded we have to reproduce 4GL behavior.

One consideration about clipping. The feature is now working for button widget but I think it has an issue. If using the clipping rectangle based on origin and size - we will loose right bottom part because the width and height of the clip areas is one pixel smaller that it should be. This may be intentional but may be a buggy. It will be good if someone who implemented clipping gives a clarifications.

Another point not yet implemented - if widget is realized it is not possible to change NO-FOCUS/FLAT-BUTTON attribute - 4GL shows the error message and continue. But I would prefer to test and commit the current stage and then implement this server side feature in addition to other remaining button attributes if there are no objections.

#6 - 03/06/2015 02:04 PM - Greg Shah

Code Review evl_upd20150306a.zip

I'm fine with the changes.

#7 - 03/06/2015 02:05 PM - Greg Shah

> One consideration about clipping. The feature is now working for button widget but I think it has an issue. If using the clipping rectangle based on origin and size - we will loose right bottom part because the width and height of the clip areas is one pixel smaller that it should be. This may be intentional but may be a buggy. It will be good if someone who implemented clipping gives a clarifications.

Hynek/Constantin: thoughts?

> But I would prefer to test and commit the current stage and then implement this server side feature in addition to other remaining button attributes if there are no objections.

I have no objections to this.

#8 - 03/06/2015 02:20 PM - Hynek Cihlar

Greg Shah wrote:

> > One consideration about clipping. The feature is now working for button widget but I think it has an issue. If using the clipping rectangle based on origin and size - we will loose right bottom part because the width and height of the clip areas is one pixel smaller that it should be. This may be intentional but may be a buggy. It will be good if someone who implemented clipping gives a clarifications.
>
> Hynek/Constantin: thoughts?

Eugenie, do you know where the clipping rectangle is first calculated wrong? If not, can you trace it down? Also, is the origin and size of the button correct?

#9 - 03/06/2015 04:01 PM - Eugenie Lyzenko

> Eugenie, do you know where the clipping rectangle is first calculated wrong? If not, can you trace it down? Also, is the origin and size of the button correct?

I'm worry about this:

```
...
   public NativeRectangle(NativePoint topleft, NativeDimension size)
   {
      top = topleft.y;
      left = topleft.x;
      bottom = top + size.height - 1;
      right = left + size.width - 1;
   }
...
```

Correct me if I'm wrong. If we need clipping for area (2,2) to (3,3) what will be the rectangle size in this case, 0?

#10 - 03/06/2015 04:15 PM - Eugenie Lyzenko

> I have no objections to this.

OK. I'm starting the conversion and runtime testing.

#11 - 03/06/2015 04:46 PM - Hynek Cihlar

Eugenie Lyzenko wrote:

> > Eugenie, do you know where the clipping rectangle is first calculated wrong? If not, can you trace it down? Also, is the origin and size of the button correct?
>
> I'm worry about this:

```
...
   public NativeRectangle(NativePoint topleft, NativeDimension size)
   {
      top = topleft.y;
      left = topleft.x;
      bottom = top + size.height - 1;
      right = left + size.width - 1;
```

```
    }
...
```

I believe this is correct. The border line denoted by the coordinate values top, left, bottom, right is included in the rectangle's region. In other words, a rectangle with left equal right is 1 unit wide.

> Correct me if I'm wrong. If we need clipping for area (2,2) to (3,3) what will be the rectangle size in this case, 0?

A rectangle of (top, left) equal (2, 2) and (bottom, right) equal (3, 3) will be 2 units wide and 2 units heigh.

Likely there will be a piece of code wrongly interpreting this. Also I think it would make sense to add the methods width() and height() to encapsulate this in (Native)Rectangle.

#12 - 03/06/2015 08:46 PM - Eugenie Lyzenko

The conversion testing completed. Generated codes are identical, no regressions. Starting runtime testing.

#13 - 03/07/2015 01:51 PM - Eugenie Lyzenko

- File evl_upd20150307a.zip added

Merge with 10796 code base.

#14 - 03/07/2015 07:40 PM - Eugenie Lyzenko

The testing completed, no regressions. The results: 10794_5c96d7c_20150307_evl.zip.

Looks like I have to re-run testing(at least conversion part) because the level 10796 touches the rule files I've changed.

#15 - 03/09/2015 03:23 PM - Eugenie Lyzenko

The new runtime test results: 10797_5c96d7c_20150309_evl.zip for update based on 10797. No regression found.

Started one more conversion testing to be sure the update is OK for 10798 base.

#16 - 03/09/2015 06:13 PM - Eugenie Lyzenko

Conversion testing completed. No difference in generated code. So the update evl_upd20150307a.zip is ready to be committed and distributed.

#17 - 03/09/2015 06:28 PM - Greg Shah

OK, please commit it.

#18 - 03/09/2015 07:24 PM - Eugenie Lyzenko

evl_upd20150307a.zip has been committed as 10800.

#19 - 03/10/2015 05:05 PM - Eugenie Lyzenko

The investigation shows the default button feature works only in GUI mode. The character mode just ignores this attribute and current P2J implementation for character mode does the same. So looks like the only we need is to add DEFAULT attribute for button and DEFAULT-BUTTON attribute for frame/dialog runtime support for GUI mode.

#20 - 03/10/2015 09:23 PM - Eugenie Lyzenko

Looking for best implementation for button DEFAULT attribute handling I see the following opportunity:
1. Pressing ENTER key causes the window.processEvent() call from ThinClient.processEventsWorker(), where window is WindowGuiIml class instance.
2. On the other hand the event should be routed to the default button of the given frame.
3. We can filter ENTER key event inside WindowGuiIml class(because we need this in GUI mode only).
4. The filter work is:
- Identify ENTER key source,
- Identify the frame for the widget that is the event source
- Detect if the frame has the DEFAULT-BUTTON attribute defined
- Route the ENTER key to default button instead of the initial source
- Consume initial event if required

#21 - 03/11/2015 08:37 AM - Greg Shah

My only concern here is that the logic for this should not be in the window widget, but in the frame widget. Doesn't the window already route the ENTER to the focused frame? There may be more than 1 frame in a window, each with their own default button. It is important to let the natural focus/event processing find the right frame and then let the frame figure out if the event should be routed to the default button.

#22 - 03/11/2015 06:47 PM - Eugenie Lyzenko

- File evl_upd20150311a.zip added

The update for review presents suggested approach in base for implementation of the default button feature. The frame receives the event from any field level widget except button because if button is a source - it handles ENTER key by itself not depending on default button settings. The update also has fast fix for label NPE, just to be able to use fill-in while testing(not supposed to be included in final implementation update).

Continue working for finding best possible solution.

#23 - 03/12/2015 09:22 AM - Greg Shah

Code Review evl_upd20150311a.zip

I'm good with the changes.

#24 - 03/12/2015 08:21 PM - Eugenie Lyzenko

- File evl_upd20150312a.zip added

This updates is merged with the recent code base. Also it adds conversion/runtime support for standalone set/get DEFAULT attribute for a button widget. And a little change to FrameGuiImpl to take into account the ButtonConfig.isDefault attribute in event processing.

Findings in 4GL investigations:
- Frame DEFAULT-BUTTON attribute can not be used with dynamic widgets, only static are possible.
- Dynamic button can be set as default one but before the button is realized. So we need to provide respective logic and message display for server side ButtonWidget.

It is not clear whether is is possible or not in 4GL to assign the new value for any of the frame attributes. For example:

```
...
def frame fr ...

fr:default-button = some_button.
...
```

This code produces compilation error in 4GL. If this assignment is not supported in Progress - we do not need rule change for standalone frame attribute set/get.

Things left to do: To add error messages processing for button attributes that can not be set after realization. And a bit more testing to be sure everything works.

#25 - 03/13/2015 03:22 PM - Eugenie Lyzenko

- File evl_upd20150313a.zip added

This update for review(merged with recent code base) is release candidate for DEFAULT button attribute feature. Added support for error messages for realization(and for previous NO-FOCUS, FLAT-BUTTON too), tested and fixed the behavior when default button has internal images(image is not drawing in this case), fixed incorrect button label dotted line rectangle coordinates and small fix while drawing selected focused button.

The LabelGuiImpl.java was removed from update because this is Constantin's work area. Correct?

If there are no objections I would like to test and commit this update before making any further changes related to remaining button attribute.

#26 - 03/16/2015 01:09 PM - Greg Shah

Code Review evl_upd20150313a.zip

2 questions:

1. Are the changes to ButtonGuiImpl really correct for the full set of your GUI testcases?

2. Do we need any more development (conversion or runtime) to support the frame-phrase default-button as in WITH FRAME my-fr

DEFAULT-BUTTON my-button.?

#27 - 03/16/2015 02:00 PM - Eugenie Lyzenko

1. Are the changes to ButtonGuiImpl really correct for the full set of your GUI testcases?

Yes.

2. Do we need any more development (conversion or runtime) to support the frame-phrase default-button as in WITH FRAME my-fr DEFAULT-BUTTON my-button.?

I guess no. The conversion and runtime support works as expected for both:

```
...
define frame fr ... default-button some_default_button.
...
```

and

```
...
form ... with frame fr default-button some_default_button.
...
```

Converting the same setters for fr.setDefaultButton(some_default_button) in both conversion results.

#28 - 03/16/2015 02:17 PM - Greg Shah

I see two things that are confusing to me:

1. Setting my-button:default-button = true does not set that button's widget ID into the containing frame's config.defaultButtonId.

2. Setting DEFAULT-BUTTON my-button in DEFINE FRAME or a frame phrase does not set the my-button to have its config.isDefault flag as true.

Shouldn't this be made to work a single way for either usage?

#29 - 03/16/2015 02:50 PM - Eugenie Lyzenko

Greg Shah wrote:

> I see two things that are confusing to me:
>
> 1. Setting my-button:default-button = true does not set that button's widget ID into the containing frame's config.defaultButtonId.
>
> 2. Setting DEFAULT-BUTTON my-button in DEFINE FRAME or a frame phrase does not set the my-button to have its config.isDefault flag as true.
>
> Shouldn't this be made to work a single way for either usage?

I think based on what I've read in Progress docs the attributes BUTTON:DEFAULT and FRAME:DEFAULT-BUTTON are independent. It is possible to change one of them and this does not affect other. Only combination of FRAME:DEFAULT-BUTTON != null and BUTTON:DEFAULT true gives the working default button feature. The testing of 4GL code confirms this.

However one issue in ButtonGuiImpl I've just found. When button:defaulttrue and button has images but frame:default-button is NOT assigned the button works as not default and image is drawing. This condition should be rewritten and I'll prepare the change shortly.

#30 - 03/16/2015 03:26 PM - Eugenie Lyzenko

- File evl_upd20150316a.zip added

This update for review fixes the issue with image based button when DEFAULT attribute is set to true but frame's DEFAULT-BUTTON is not set.

#31 - 03/16/2015 03:48 PM - Greg Shah

Code Review evl_upd20150316a.zip

I'm fine with this. Please get it regression tested.

#32 - 03/16/2015 08:05 PM - Eugenie Lyzenko

- File evl_upd20150316b.zip added

The update for review presents conversion support for AUTO-GO and AUTO-ENDKEY attributes. Including dynamic handle based buttons. The word AUTO-END-KEY is also supported as equivalent to AUTO-ENDKEY.

As to runtime support - looks like we already have the proper functionality. The simple tests (pressing button fires the respective event for frame) are working fine. May be we need a bit more complicated test(say getting indirect GO and ENDKEY event from default button which is not really pressed)

to be sure everything is working good.

#33 - 03/16/2015 08:38 PM - Eugenie Lyzenko

Conversion testing for evl_upd20150316a.zip has been completed. Generated codes are identical. Continue with runtime testing.

#34 - 03/16/2015 09:01 PM - Greg Shah

Code Review evl_upd20150316b.zip

I'm good with the changes.

Please do test the additional cases like GO/ENDKEY triggers and make sure that in the case of AUTO-ENDKEY, that the containing block is exited when expected (e.g. ON ENDKEY undo, leave). If everything is working well, you can put this into testing after evl_upd20150316a.zip passes.

#35 - 03/16/2015 09:14 PM - Eugenie Lyzenko

> Please do test the additional cases like GO/ENDKEY triggers and make sure that in the case of AUTO-ENDKEY, that the containing block is exited when expected (e.g. ON ENDKEY undo, leave). If everything is working well, you can put this into testing after evl_upd20150316a.zip passes.

OK. I have the test that monitors ENDKEY event and it works OK. When the ENDKEY capable button is pressed, the test terminates meaning the main loop block is closing. Is this what you mean? Or I need to create one more execution block inside the main loop?

And I have encountered issue with test that uses indirect GO/ENDKEY events firing via default button. The client restarts and I need to investigate whether this is event related issue or something else GUI related.

#36 - 03/16/2015 11:03 PM - Greg Shah

> When the ENDKEY capable button is pressed, the test terminates meaning the main loop block is closing. Is this what you mean? Or I need to create one more execution block inside the main loop?

Yes, it is a good idea to to add an inner loop with a ON ENDKEY undo, leave clause. Make sure that nothing executes after the edit statement, but that the next statement after the inner loop does execute.

> And I have encountered issue with test that uses indirect GO/ENDKEY events firing via default button.

Are you using triggers for these tests (e.g. for the ENDKEY case, does your test have a trigger for the ENDKEY event ON ENDKEY of my-button DO: ... END.)?

#37 - 03/17/2015 09:04 AM - Eugenie Lyzenko

> Are you using triggers for these tests (e.g. for the ENDKEY case, does your test have a trigger for the ENDKEY event ON ENDKEY of my-button DO: ... END.)?

This is not trigger or ENDKEY attribute related issue. This is GUI problem from fill-in widget area, something related to font initialization. Anyway I've found a way to get rid of this, so we can resume ENDKEY related testing. And BTW indirect event firing via default button(pressing ENTER inside fill-in widget) works fine.

#38 - 03/17/2015 10:01 AM - Eugenie Lyzenko

> Yes, it is a good idea to to add an inner loop with a ON ENDKEY undo, leave clause. Make sure that nothing executes after the edit statement, but that the next statement after the inner loop does execute.

The good news is this case is working as expected. So I conclude we already have the runtime support for AUTO-GO and AUTO-ENDKEY attributes.

#39 - 03/17/2015 10:03 AM - Greg Shah

You've also tested both the GO and ENDKEY triggers and they work properly?

Is evl_upd20150316b.zip the final update?

#40 - 03/17/2015 11:29 AM - Eugenie Lyzenko

> You've also tested both the GO and ENDKEY triggers and they work properly?

ENDKEY works OK. But I have made one more test with inner loop and GO event and I have a doubt. Consider the test:

```
...
def button b_sam label "Sample" default auto-go.
def button b_quit label "Quit" auto-endkey.
def var vInt as INTEGER view-as FILL-IN.
def var bVar as logical.

def frame fr
  b_sam vInt b_quit
  with centered size 80 by 20 title "Button Test #20_6, Auto Go button inner loop demo"
  default-button b_sam.
```

```
enable all with frame fr.

do while true on endkey undo, leave:
   message "Before update statement".
   update vInt with frame fr.
   message "After update statement".
   pause.
end.

message "Inner loop finished with ENDKEY".
...
```

Pressing Sample button in 4GL finishes the vInt updating, while in P2J it does nothing. Previous tests shows the GO event is firing for frame(directly and indirectly). So looks like frame does not route the GO event to vInt fill-in in P2J to commit the update.

Do we need to investigate/fix this deviation in this task?

> Is evl_upd20150316b.zip the final update?

I need some debugging for GO event routing to have the final answer.

#41 - 03/17/2015 01:09 PM - Greg Shah

> Do we need to investigate/fix this deviation in this task?

Yes, this should be fixed.

#42 - 03/17/2015 03:33 PM - Eugenie Lyzenko

The runtime testing completed for evl_upd20150316a.zip, CTRL-C 3-way tests was started in separate session. No regressions, the results: 10812_5c96d7c_20150317_evl.zip.

So the update is ready to be committed and distributed.

#43 - 03/17/2015 04:02 PM - Greg Shah

Great! Commit it.

#44 - 03/17/2015 04:29 PM - Eugenie Lyzenko

Greg Shah wrote:

> Great! Commit it.

Committed to rev 10813. Will be distributed shortly.

#45 - 03/18/2015 09:39 PM - Eugenie Lyzenko

Some debugging results:
1. Pressing on button the ENTER key and clicking by mouse gives very different results:
The keyboard button pressing events(the first digit is the action code):

```
32, ENTER. typed
-111, CHOOSE, pressed
-142, GO, pressed
32, ENTER. typed
-111, CHOOSE, pressed
-142, GO, pressed
```

The mouse button pressing(the first digit is the action code):

```
-135, ENTRY, pressed(for WindowsGuiImpl ?)
-151, LEAVE, pressed(for WindowsGuiImpl ?)
-177, RETURN, typed
-111, CHOOSE, pressed
-142, GO, pressed
```

Note for a keyboard source we have 2 set of the same events and it is a bit weird.

The fact is at least one time the frame with fill-in receives the GO event. But if the event initiator is mouse - the fill-in data update is not finishing. Probably something is wrong with routing GO event from the frame to fill-in(or any other data editable containers) and not in button itself. On the other hand it is not clear why the ENTER key can not be fully simulated with:

```
...
EventManager.postEvent(new KeyInput(this, EventType.KEY_TYPED, Key.VK_ENTER));
...
```

Continue debugging.

#46 - 03/19/2015 08:54 AM - Greg Shah

Please post the 4GL sequence of events for (chui, enter key), (gui, enter key), (chui, mouse click), (gui, mouse click). I want to understand what exactly we are trying to duplicate and if there are any differences for chui/gui.

> Note for a keyboard source we have 2 set of the same events and it is a bit weird.

Yes, I'm pretty sure its wrong.

#47 - 03/19/2015 12:13 PM - Eugenie Lyzenko

Looks like the events are:

```
(chui, enter key), (gui, enter key):
low level RETURN key event
CHOOSE in button
GO or ENDKEY for frame owning button if AUTO-GO or AUTO-ENDKEY attributes is true.

(chui, mouse click): (Windows console applications)
mouse click on button - low level SELECT portable mouse (if chui console supports mouse actions)
CHOOSE in button
GO or ENDKEY for frame owning button if AUTO-GO or AUTO-ENDKEY attributes is true.

(gui, mouse click)
mouse click on button - low level SELECT portable mouse
CHOOSE in button
GO or ENDKEY for frame owning button if AUTO-GO or AUTO-ENDKEY attributes is true.
```

#48 - 03/19/2015 12:50 PM - Greg Shah

> Note for a keyboard source we have 2 set of the same events and it is a bit weird.

> Yes, I'm pretty sure its wrong.

Is this code EventManager.postEvent(new KeyInput(this, EventType.KEY_TYPED, Key.VK_ENTER)); responsible for the doubling of events?

#49 - 03/19/2015 01:15 PM - Eugenie Lyzenko

> Is this code EventManager.postEvent(new KeyInput(this, EventType.KEY_TYPED, Key.VK_ENTER)); responsible for the doubling of events?

No, at least not from ButtonGuiIml. Because we have this doubling in CHUI mode too. The second sequence is appearing after GO for first one. Moreover the fill-in commit is happening only after GO from second sequence. I suggest the widget validation in this case is not caused by GO event but some other action(like total Window invalidation from toolkit). Because the regular place when fill-in should be committed is TC.processProgressEvent() line 14613. But the GO event is not explicitly registered for event list and validateWidget() returns false.

#50 - 03/19/2015 01:32 PM - Greg Shah

> Moreover the fill-in commit is happening only after GO from second sequence. I suggest the widget validation in this case is not caused by GO event but some other action(like total Window invalidation from toolkit). Because the regular place when fill-in should be committed is TC.processProgressEvent() line 14613. But the GO event is not explicitly registered for event list and validateWidget() returns false.

I don't understand what you mean by "But the GO event is not explicitly registered for event list". The code in line 14613 is not dependent upon whether there is a GO event registered. It is just checking if there is a validation expression registered for the given widget. If there is a validation expression then it is executed, controlling whether the GO is rejected or not. Does the fill-in in question have a validation expression?

#51 - 03/19/2015 02:01 PM - Eugenie Lyzenko

> I don't understand what you mean by "But the GO event is not explicitly registered for event list".

I mean the trigger on go do something... is not defined for:

```
...
do while true on endkey undo, leave:
   message "Before update statement".
   update vInt with frame fr.
   message "After update statement".
   pause.
end.
...
```

> The code in line 14613 is not dependent upon whether there is a GO event registered. It is just checking if there is a validation expression registered for the given widget. If there is a validation expression then it is executed, controlling whether the GO is rejected or not. Does the fill-in in question have a validation expression?

No, the fill-in does not have such expression and this is not a problematic place.

#52 - 03/19/2015 10:00 PM - Eugenie Lyzenko

Further investigation shows another strange behavior in considered testcases. Pressing AUTO-GO button does not commit fill-in widget but if to return then focus back to fill in and attempt to resume data editing - after first key(digit in our case) the P2J commit the changes and leave update statement.

I've dug into update event processing and found we have separate event loop for this case - TC.waitForEvent() line 11887:

```
...
evt = typeAhead.getKeystroke(seconds * 1000, honorServerEvent);
...
```

Two simultaneous key waiting loop could explain double key generation. So looks like I need to observe full event handling approach to find the solution.

So looks like the issue I'm working on is outside the button widget auto-go feature, at least it's conversion part. And I'm offering to start testing/commit for evl_upd20150316b.zip while I'm working on the issue resolution to not to postpone conversion part too far. What do you think, is it acceptable?

#53 - 03/20/2015 08:34 AM - Greg Shah

> Further investigation shows another strange behavior in considered testcases. Pressing AUTO-GO button does not commit fill-in widget but if to return then focus back to fill in and attempt to resume data editing - after first key(digit in our case) the P2J commit the changes and leave update statement.

I wonder if isPendingGo > 0 but was somehow ignored (until too late) in your code path.

> I've dug into update event processing and found we have separate event loop for this case - TC.waitForEvent() line 11887:
> [...]
> Two simultaneous key waiting loop could explain double key generation.

I don't understand what you mean here. There is no 2nd loop. TC.waitForEvent() can only be called by TC.readKey(), TC.pause() or TC.waitForWorker(). TC.waitForWorker() is used in the UPDATE case. I think the UPDATE somehow ignores the isPendingGo and then continues

editing but then "sees" the GO later.

> So looks like the issue I'm working on is outside the button widget auto-go feature, at least it's conversion part.

Not this time. It is important to clean this up now.

#54 - 03/20/2015 12:33 PM - Eugenie Lyzenko

> I don't understand what you mean here. There is no 2nd loop. TC.waitForEvent() can only be called by TC.readKey(), TC.pause() or TC.waitForWorker(). TC.waitForWorker() is used in the UPDATE case.

I thought during the update statement we have TC.waitForEvent() and processEventsWorker() both active for incoming events. Is this not correct?

> Not this time. It is important to clean this up now.

OK. Understood.

#55 - 03/20/2015 01:15 PM - Greg Shah

> I thought during the update statement we have TC.waitForEvent() and processEventsWorker() both active for incoming events. Is this not correct?

They are both used from waitForWorker(), but they do very different things.

TC.waitForEvent() - this is used to read the next key, it does NOT do any processing for the event that is read.

After an event is read, it is posted into the event processing queue by TC.applyWorker(). applyWorker() uses an eventBracket() and inside that it posts the event, which will cause processing of that queue by TC.pop() which will call TC.processEventsWorker() in which the event is actually processed.

#56 - 03/20/2015 02:59 PM - Eugenie Lyzenko

- File evl_upd20150320a.zip added

This update for review contains the fix for GO event issue. Finally I have found the solution. The idea is any GO tat came from mouse controlled widget stops the update key reader loop. Previously it happens the same way but incoming from keyboard. Certain mouse action result should do the same.

#57 - 03/20/2015 05:26 PM - Eugenie Lyzenko

- File evl_upd20150320b.zip added

Merge with recent code base.

#58 - 03/23/2015 12:21 PM - Greg Shah

Code Review evl_upd20150320b.zip

1. getAuto*Key() should be isAuto*Key().

2. In ThinClient.waitForEvent(), there this code:

```
            // this event came from not updateable widget (button is an example)
            if (osEvent instanceof KeyInput && ((KeyInput)osEvent).keyCode() == Key.VK_ENTER &&
                (currentBlockingOp == BlockingOperation.WAIT_FOR ||
                 currentBlockingOp == BlockingOperation.PROMPT_FOR))
            {
```

I don't understand how that condition can only occur for a "not updateable" widget. This idea assumes too much that is implicit in your implementation. Other usage of VK_ENTER in an OSEvent will break this code. I would like to see this be much more explicit and thus safer.

3. With these changes, is the event processing for BOTH AUTO-GO and AUTO-ENDKEY exactly the same as in note 47 above (with no doubling, no order differences, all 3 events are generated with the right source...). At a minimum I don't see any specific fixes for the AUTO-ENDKEY case.

#59 - 03/23/2015 03:33 PM - Eugenie Lyzenko

Greg Shah wrote:

> Code Review evl_upd20150320b.zip
>
> 1. getAuto*Key() should be isAuto*Key().

OK. Done.

> 2. In ThinClient.waitForEvent(), there this code:

> [...]
>
> I don't understand how that condition can only occur for a "not updateable" widget. This idea assumes too much that is implicit in your implementation. Other usage of VK_ENTER in an OSEvent will break this code. I would like to see this be much more explicit and thus safer.

I'll find another solution.

> 3. With these changes, is the event processing for BOTH AUTO-GO and AUTO-ENDKEY exactly the same as in note 47 above (with no doubling, no order differences, all 3 events are generated with the right source...). At a minimum I don't see any specific fixes for the AUTO-ENDKEY case.

Key doubling is still here(if source is keyboard, not mouse). This needs additional investigation because it is outside GO event processing(independent issue).

AUTO-ENDKEY case does not need special fixes because it woks fine generating ENDKEY event and properly handling this event with the current code.

#60 - 03/23/2015 03:37 PM - Greg Shah

> Key doubling is still here(if source is keyboard, not mouse). This needs additional investigation because it is outside GO event processing(independent issue).

OK, please fix this too.

#61 - 03/23/2015 11:18 PM - Eugenie Lyzenko

I've tested to use another event to unlock the update statement(CHOOSE from GUI button). It works but causes another artifact and does not change the status of the key doubling issue.

So taking into account the key doubled case handles properly GO event on the second pass I suggest if we fix this probably we will have the GO event issue in the keyboard case too. So my plan is:
- Generate CHOOSE event on mouse click. This will not unlock the update key reader cycle.
- Move unlock call from ButtonGuiImpl to another place - assume TC.processEventsWorker()
- Track and fix to place where key doubling is happening. This can be indirect call TC.processEventsWorker() from TC.processEventsWorker() as part

of TC.applyWorker() and TC.pop().

Continue debugging.

#62 - 03/24/2015 09:34 PM - Eugenie Lyzenko

- File evl_upd20150324a.zip added

This update for review has different approach for GO issue fix. Now mouse click causes CHOOSE event to be generated. And this is natural to 4GL. Then CHOOSE event causes special GO for GUI mode to be posted as OSEvent. This event is captured by TC.waitForEvent() and "release" update statement.

This fix solves the GO event issue with update statement and key doubling issue in event loop. Note, the update contains some debugging calls in TC to be removed later because I suggest this is still not final version.

On the other hand I've found another problem we need to fix. If we do not define the button as AUTO-ENDKEY featured and just wait for this button choose in:

```
...
wait-for window-close of current-window or choose of b_quit in frame fr
....
```

pressing b_quit does not release wait-for as expected. The loop is releasing only after pressing any key on keyboard. And this is something unexpected and need to be fixed too, correct?

And the other feature I'm going to investigate and test is the case when both AUTO-END and AUTO-GO options are true for same button. We need to know what will be the key sequence in 4GL and to reproduce the same in P2J.

#63 - 03/25/2015 08:35 AM - Greg Shah

Code Review evl_upd20150324a.zip

I'm OK with this approach.

> The loop is releasing only after pressing any key on keyboard. And this is something unexpected and need to be fixed too, correct?

Yes.

> And the other feature I'm going to investigate and test is the case when both AUTO-END and AUTO-GO options are true for same button. We need to know what will be the key sequence in 4GL and to reproduce the same in P2J.

Yes. And after you determine and if needed fix the order, please leave behind a code comment explaining what the 4GL order of precedence is.

#64 - 03/25/2015 02:42 PM - Eugenie Lyzenko

Investigation results for case when both AUTO-ENDKEY and AUTO-GO options are true for same button:

1. define button does not support this option, AUTO-ENDKEY setting is just ignored, not depending on order of the AUTO-ENDKEY and AUTO-GO options in DEFINE BUTTON statement.

2. However it is possible to change these options after the button has been defined. In this case we can have button with both AUTO-ENDKEY and AUTO-GO attributes with true values. In this case the button works as having only AUTO-ENDKEY. Ability to generate the GO event is ignoring.

Looking for a best approach to implement this.

#65 - 03/25/2015 04:34 PM - Eugenie Lyzenko

- File evl_upd20150325a.zip added

This update for review has fix for wait-for loop "exit" button without endkey option issue. The idea is to propagate CHOOSE as OSEvent for this case in GUI mode. Then this event is captured by TC.waitForEvent(), return event to main loop processor where the event source ID is comparing with known list of "exit" widgets. If we pressed "exit" button, the event is consumed and wait-for loop is stopped.

Also the update contains support for simultaneous usage of the AUTO-ENDKEY and AUTO-GO option. DEFINE BUTTON statement is suggested to use new option setters instead of regular attribute setters(which also are used after DEFINE BUTTON and in dynamic button cases). The new options setters check other option and: 1. ignore AUTO-ENDKEY setting when AUTO-GO is true or 2. disable AUTO-ENDKEY if AUTO-GO is going to be true.

So this is candidate for regression testing.

#66 - 03/25/2015 05:03 PM - Greg Shah

Code Review evl_upd20150325a.zip

1. This code needs to be merged with the latest bzr level.

2. Why did you add the condition around the applyWorker() in waitForWorker()? This could have many unintended consequences. It seems very risky.

3. Adding CHOOSE event processing to the end of waitForWorker() is not correct. Event-specific logic should be in the event processing methods and the isGoPending counter should be incremented from there to control the wait-for exit.

Constantin: please review.

#67 - 03/25/2015 05:40 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: please review.

Eugenie, beside what Greg noted, I don't like using the EventManager$WorkArea.osEvent queue with other non-mouse or non-window events. This queue was intended for events which are raised at the OS level and have priority over any other kind of events.

Do you have a simple test which shows what you are trying to fix?

#68 - 03/25/2015 05:45 PM - Eugenie Lyzenko

1. This code needs to be merged with the latest bzr level.

It was synced yesterday with 10821.

2. Why did you add the condition around the applyWorker() in waitForWorker()?

This is the additional protection from KeyInput consumed event not from buttons other that is used as "exit" button. Otherwise CHOOSE event will be generated twice.

3. Adding CHOOSE event processing to the end of waitForWorker() is not correct. Event-specific logic should be in the event processing methods and the isGoPending counter should be incremented from there to control the wait-for exit.

The reason is to properly process this case we need to know the list of the widgets declared as waiting for choose to exit the waitForWorker() event loop.

#69 - 03/25/2015 05:56 PM - Eugenie Lyzenko

> ...I don't like using the EventManager$WorkArea.osEvent queue with other non-mouse or non-window events. This queue was intended for events which are raised at the OS level and have priority over any other kind of events.

What do you mean by OS level? Moving the mouse over Java application?

> Do you have a simple test which shows what you are trying to fix?

Try to exit application pressing "Quit" button in GUI. Do not touch the keyboard.

```
...
message "Hit a key to start".
pause.

def button b_sam label "Sample" auto-go default.
def button b_quit label "Quit".
def var vInt as INTEGER view-as FILL-IN.
def var bVar as logical.

def frame fr
  b_sam vInt b_quit
  with centered size 80 by 20 title "Button Test #19_2, Auto Go default button attribute demo"
  default-button b_sam.

on choose of b_sam in frame fr
do:
  message "Sample button pressed".
end.

on go of frame fr
do:
  message "GO event has been fired".
end.

enable all with frame fr.

wait-for window-close of current-window or choose of b_quit in frame fr.
...
```

#70 - 03/26/2015 09:06 AM - Eugenie Lyzenko

- File evl_upd20150326a.zip added

This update for review fixes the previous notes.
- applyWorker() in waitForWorker() is now free from condition.
- The event processing code has been moved from waitForWorker() to event processing method.
- The code is checked to be synced with the latest bzr version.

#71 - 03/26/2015 09:17 AM - Greg Shah

Code Review evl_upd20150326a.zip

This is significantly better than before. But it still uses the OSEvent mechanism for something that it wasn't originally designed to do.

Constantin: what do you think?

#72 - 03/26/2015 09:42 AM - Eugenie Lyzenko

> This is significantly better than before. But it still uses the OSEvent mechanism for something that it wasn't originally designed to do.

We need the way to deliver some events to waitForEvent(). Currently it is possible either via real keyboard or OSEvent. We can either use existed tools or create new one especially for the cases we need to handle(GO events in PROMPT_FOR and "exit" button choose in WAIT_FOR). And currently usage of the OSEvent in waitForEvent() just takes a lot of mouse moving events and most of time does nothing, just take them from the queue. Comparing to mouse events two additional "special cases" are not heavy load.

Ideally I think the GUI mode requires it's own waitForEvent() version because we have not only keyboard as event source on the client side. The significant event we need to react can came from widget currently not having the input focus, that's the difference from CHUI. If we use keyboard based handler - we need to make artificial events for other sources simulating the certain keyboard actions.

#73 - 03/26/2015 09:51 AM - Greg Shah

These are some good points.

> The significant event we need to react can came from widget currently not having the input focus, that's the difference from CHUI.

Perhaps this is really the core issue. Shouldn't clicking on a button cause that button to receive focus? That focus change should probably occur **before** any other event processing occurs. If that happened, would the subsequent event processing (e.g. CHOOSE) work more naturally?

Some questions:

1. Does clicking on a mouse button in the 4GL shift focus to that button?
2. Does that focus shift cause LEAVE and ENTRY events to occur?
3. If so, do the LEAVE and ENTRY events occur before any other event processing (e.g. CHOOSE)?

4. Can the focus shift be cancelled like in ChUI (if either the LEAVE or the ENTRY event has a trigger that does a RETURN NO-APPLY, then the focus change is stopped)?

#74 - 03/26/2015 10:21 AM - Eugenie Lyzenko

> Shouldn't clicking on a button cause that button to receive focus? That focus change should probably occur before any other event processing occurs.

Correct and this is how the current button implementation works. The button requests the focus on pressing event. The mouse click is later:

```
...
press(focus request)
release
click(choose)
...
```

This is how it currently works.

> If that happened, would the subsequent event processing (e.g. CHOOSE) work more naturally?

In ideal world yes, but sometimes our focus processing sends LEAVE to widget after widget gets focus. If this happens before CHOOSE generation - the problems can occur.

#75 - 03/26/2015 10:36 AM - Eugenie Lyzenko

1. Does clicking on a mouse button in the 4GL shift focus to that button?

Yes if button's NO-FOCUS is false.

2. Does that focus shift cause LEAVE and ENTRY events to occur?
3. If so, do the LEAVE and ENTRY events occur before any other event processing (e.g. CHOOSE)?
4. Can the focus shift be cancelled like in ChUI (if either the LEAVE or the ENTRY event has a trigger that does a RETURN NO-APPLY, then the

focus change is stopped)?

Need to write testcase to discover this. Will report results when I get.

#76 - 03/26/2015 12:28 PM - Constantin Asofiei

Eugenie, about the EventManager.postOSEvent and when mouse is used to click the button. From what I can tell, past the initial MouseEvent which acts as the root cause for the button's event chain, there should be nothing different from the normal case (when the keyboard is used to generated these events).

When processing the mouse event, P2J gets stuck in the waitForEvent, processing OS events:

```
         if (evt == null && EventManager.hasOSEvents())
         {
            // OS events will be treated first
            applyWorker(EventManager.getOSEvent());

            // recompute the focus (might have changed)
            comp = computeFocus(frameAsTarget);

            // force waiting for a key

            continue;
         }
```

This will not let the loop exit back to the caller (the waitForWorker) until a key was received, even if applyWorker raises some other non-OS event (as generated by the i.e. mouse click).

What I think we need is a way to capture this additional event (the CHOOSE event for BUTTON, correct?), and let it propagate to the waitForWorker loop, which should do all the work without any special handling. The complex part is that when the ButtonGuiImpl.mouseClicked is called, it will post the event to the event loop associated with the applyWorker call from the waitForWorker, which will want to process it. So, what about this: we have an applyWorker mode in which it can process only OS events and "saves" the additional event in some other queue. Once the applyWorker is finished, this event is pushed to the caller. The restriction will be that only one event can be "saved", which will be the root cause for some other events (GO, etc).

#77 - 03/26/2015 12:55 PM - Eugenie Lyzenko

> 2. Does that focus shift cause LEAVE and ENTRY events to occur?

Yes, LEAVE for button loosing focus and ENTRY for button gaining it.

> 3. If so, do the LEAVE and ENTRY events occur before any other event processing (e.g. CHOOSE)?

Yes, first ENTRY then CHOOSE.

> 4. Can the focus shift be cancelled like in ChUI (if either the LEAVE or the ENTRY event has a trigger that does a RETURN NO-APPLY, then the focus change is stopped)?

Yes the focus shifting can be cancelled by RETURN NO-APPLY. The button become focused on the time the mouse is pressed. When mouse releasing the focus returns to the previously focused widget. No CHOOSE is generating in this case. And no LEAVE is generating too.

#78 - 03/26/2015 01:22 PM - Eugenie Lyzenko

> Eugenie, about the EventManager.postOSEvent and when mouse is used to click the button. From what I can tell, past the initial MouseEvent which acts as the root cause for the button's event chain, there should be nothing different from the normal case (when the keyboard is used to generated these events).

But there is. We have a difference.

> When processing the mouse event, P2J gets stuck in the waitForEvent, processing OS events:

I see. And there is no chances to get for waitForEvent caller any high-level events(GO, CHOOSE...) generated by mouse clicking the button.

#79 - 03/26/2015 01:56 PM - Eugenie Lyzenko

> What I think we need is a way to capture this additional event (the CHOOSE event for BUTTON, correct?), and let it propagate to the waitForWorker loop, which should do all the work without any special handling. The complex part is that when the ButtonGuiImpl.mouseClicked is called, it will post the event to the event loop associated with the applyWorker call from the waitForWorker, which will want to process it. So, what about this: we have an applyWorker mode in which it can process only OS events and "saves" the additional event in some other queue. Once the applyWorker is finished, this event is pushed to the caller. The restriction will be that only one event can be "saved", which will be the root cause for some other events (GO, etc).

We need to:
1. Preserve natural 4GL event sequence. We can not postpone some events because it can change the result and is very difficult to debug.
2. Not have any difference for high-level events, like GO and CHOOSE no matter from where they are coming from, keyboard or mouse.
3. Avoid any additional complication than we have to had. Especially for focus related stuffs.

If you worry about usage "OS Event" queue in this case - OK, I can implement one more "special" queue object to handle "special" cases. But I still do not understand what is the "OS Event facility" purpose. Do you think in Windows 4GL Progress code intercepts all mouse events? Do you need this to implement drag-drop?

#80 - 03/26/2015 03:41 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> We need to:
> 1. Preserve natural 4GL event sequence. We can not postpone some events because it can change the result and is very difficult to debug.
> 2. Not have any difference for high-level events, like GO and CHOOSE no matter from where they are coming from, keyboard or mouse.
> 3. Avoid any additional complication than we have to had. Especially for focus related stuffs.

I agree.

> But there is. We have a difference.

Please post a test which shows this difference. And I mean difference in ChUI/GUI behavior in 4GL, not P2J, because I can't find one (I checked your test and some AUTO-GO and UPDATE cases).

> If you worry about usage "OS Event" queue in this case - OK, I can implement one more "special" queue object to handle "special" cases. But I still do not understand what is the "OS Event facility" purpose. Do you think in Windows 4GL Progress code intercepts all mouse events? Do you need this to implement drag-drop?

It's not only about the "OS Event" queue usage. The same event should be processed in the same way, without complicated hacks for GUI.

For the BUTTON case in GUI, the order is now, once a button with AUTO-GO is pressed:

- MOUSE-CLICK: posted to the OS Event queue and managed by TC.waitForEvent "OS event" processing code
- ButtonGuiImpl.mouseClicked posts a CHOOSE event (we are still in the TC.waitForEvent "OS event" processing code)
- CHOOSE is followed by a GO (in case of a AUTO-GO) (we are still in the TC.waitForEvent "OS event" processing code)
- after GO, we escape the TC.waitForEvent "OS event" processing code and return control to TC.waitForWorker

In ChUI, the order is this:

- space bar key press
- TC.waitForEvent receives a KeyInput which is managed by TC.checkForSystemEvent, which in turn treats these:
  - an ItemEvent posted by Button.handleBasicEvents, which is later converted to CHOOSE
  - CHOOSE, which is followed by a GO (in case of a AUTO-GO)
  - after GO, we escape the TC.checkForSystemEvent and return control to TC.waitForWorker

Once the initial mouse event is treated, any 4GL event (like CHOOSE) raised during the mouse event processing needs to "escape" the OS event processing in TC.waitForEvent and be treated by downstream code, which can do its job without knowledge about how that CHOOSE was raised (mouse-click or key press).

#81 - 03/26/2015 04:51 PM - Eugenie Lyzenko

> Please post a test which shows this difference. And I mean difference in ChUI/GUI behavior in 4GL, not P2J, because I can't find one (I checked your test and some AUTO-GO and UPDATE cases).

I have committed my button testcases(uast/button) int bzr 1263. The test gui_btn_test19_2.p you already have. The test to demonstrate GO issue - gui_btn_test20_6.p. The difference is in P2J, right. In 4GL there is one point, after the system has the CHOOSE event - no differences then. But the sources of the CHOOSE are different in 4GL, mouse click for mouse or ENTER/SPACEBAR keys for keyboard.

For P2J implementation:

1. Your assumptions for MOUSE-CLICK processing are not confirmed by gui_btn_test20_6.p. Pressing GO button does not finish update statement how it happens in 4GL.
2. Pressing "exit" button dows not captured by wait-for ... choose of b_quit in frame fr in gui_btn_test19_2.p how it happens in 4GL.

The reason is the same - we are in TC.waitForEvent waiting for the real key pressing from keyboard. These two issues I fixed with my recent changes using OSEvent queue.

#82 - 03/26/2015 05:34 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> The reason is the same - we are in TC.waitForEvent waiting for the real key pressing from keyboard. These two issues I fixed with my recent changes using OSEvent queue.

I understand what you mean; but when we will add full GUI event support for other widgets (like COMBO-BOX or TOGGLE-BOX) we will have to handle each and every event and treat it separately so that is pushed to the OS Event queue, if this event was triggered by a mouse action.

This is another reason why I think is better to try to find a generic solution for this BUTTON/CHOOSE/GO problem now, instead of treating each case separately.

I'll look on the tests you mention tomorrow.

#83 - 03/26/2015 10:46 PM - Eugenie Lyzenko

> This is another reason why I think is better to try to find a generic solution for this BUTTON/CHOOSE/GO problem now, instead of treating each case separately.

I agree, the general solution is the best. Ideally this should work as in 4GL, no matter what is the source for the events like CHOOSE and GO. After the high level events are starting to live - they should be handled the same way in GUI and ChUI, like I suggest it is happening in 4GL. Moreover the mouse can also be used in ChUI mode in 4GL too(on Windows at least).

I think we need to make mouse events(at least press/release) to be returned from TypeAhead.getKeystroke() as non-null event. We can even substitute mouse events with respective key event there. For example mouse click->enter key press. However if 4GL code depends on MOUSE-*-* this can be a bad idea. Anyway to avoid special fixed for every GUI we need to distribute mouse event to low level of the P2J when high level 4GL events are starting to live. It does not have to be the same queue(we can use another queue if it is possible in 4GL to use keyboard and mouse simultaneously) as for key events but as result these events must be returned from TypeAhead.getKeystroke().

#84 - 03/27/2015 09:43 AM - Eugenie Lyzenko

There is one thing I've recently found related to button widget. In Windows 4GL there are 4 prepackaged images to be used in button widget(Page 381 pf the ABL Reference):

```
btn-up-arrow
btn-down-arrow
btn-left-arrow
btn-right-arrow
```

You can see how it looks at the button widget description page in same reference(for up and down case). We need the same functionality, correct?

Looks like we have to implement them the same way as "gclogo.ico" file. Can we just duplicate images from Windows? Or for property reasons we can not use images from commercial 4GL product and have to create our own versions?

#85 - 03/27/2015 09:44 AM - Eugenie Lyzenko

- File button_4gl_laf.jpg added

- File button_4gl.jpg added

> Or for property reasons we can not use images from commercial 4GL product and have to create our own versions?

Yes. They can look exactly the same, but we have to create ours from scratch.

#86 - 03/27/2015 04:55 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> I think we need to make mouse events(at least press/release) to be returned from TypeAhead.getKeystroke() as non-null event.

I suspect my initial implementation is a little (or a little more) incomplete. I think what I was trying was not to make a PAUSE. or READKEY. to terminate on a mouse action - as you can see, TC.waitForEvent is called by these, too. But what I didn't consider was that a mouse trigger can affect a 4GL event processing loop. See this example:

```
DEF VAR i AS INT.
FORM i WITH FRAME f1.
ON 'mouse-select-down':U ANYWHERE
DO:
   MESSAGE "click" i.
   i = i + 1.
   APPLY "go" TO FRAME f1.
   RETURN.
END.

UPDATE i WITH FRAME f1.
```

Also, a mouse event can terminate a wait-for loop, when used as an exit condition:

```
DEF VAR i AS INT.
FORM i WITH FRAME f1.
ON 'mouse-select-down':U OF CURRENT-WINDOW
DO:
   MESSAGE "click" i.
   i = i + 1.
   APPLY "go" TO FRAME f1.
   RETURN NO-APPLY.
END.

ENABLE ALL WITH FRAME f1.

WAIT-FOR "left-mouse-click" OF CURRENT-WINDOW.
```

What about this: let the TC.pause and TC.readkey call TC.waitForEvent with some parameter to wait only for key events, and in other cases, when OS events are treated, return it directly. We may need to either make i.e. MouseEvt extend KeyInput or treat these events explicitly at the TC.waitForWorker.

> We can even substitute mouse events with respective key event there. For example mouse click->enter key press.

We can't do this, as there are 4GL triggers for mouse: see the portable mouse events (MOUSE-SELECT-DOWN for example for a mouse-click).

#87 - 03/27/2015 08:45 PM - Eugenie Lyzenko

> What about this: let the TC.pause and TC.readkey call TC.waitForEvent with some parameter to wait only for key events, and in other cases, when OS events are treated, return it directly. We may need to either make i.e. MouseEvt extend KeyInput or treat these events explicitly at the TC.waitForWorker.

It is a good idea and I think it can be implemented. I've experimented last hours and can say it does not work "out of the box". The issues I fixed recently does not disappear. Looks like we need to solve several points before this can work:

- The first thing I want to do is to separate mouse moving from other mouse events. This does not mean I want to completely ignore moving. This mean to separate moving to another queue to be able to assign different priority to different events. Otherwise moving will trash our queue and potentially can make the application irresponsible to more important but less frequent mouse event.

- I think TC.waitForEvent should receive not direct Java mouse events but 4GL portable mouse events or may be plus "three-button" mouse events, if we are going to support both. However may be we need to transform "three-button" events to "portable" before they appear in OS Event queue for TC.waitForEvent.

- We need to have code that converts 4GL mouse events into 4GL high level events. Correct me if I'm wrong but this is very different comparing to what is currently implemented in P2J. Our widgets handle "native" AWT mouse events transforming to 4GL high level events. If we change AWT mouse with 4GL simulator - the mouse handlers in all widget should be rewritten. Or TC will handle the 4GL mouse events too.

What do you think?

#88 - 04/01/2015 06:46 PM - Eugenie Lyzenko

- File evl_upd20150401a.zip added

This update for review present new approach for GO/CHOOSE issue resolution. The idea is to include mouse clicks in key event processing if it is requested by special option in TC.waitForEvent() method. Default value for option is false. The MouseEvt class now extends KeyInput to avoid casting exception. This allows:
- to get rid of any button/event specific code introduced recently
- to avoid double key generation
- process GO/CHOOSE, loop exit triggers with mouse properly.
- leave OSEvent queue untouched

The only thing I did intentionally - mouse moving events are ignored even we request mouse events processing.

#89 - 04/02/2015 06:45 AM - Constantin Asofiei

I'm good with the changes. Are the tests in note 86 now working properly?

#90 - 04/02/2015 08:07 AM - Eugenie Lyzenko

Constantin Asofiei wrote:

> I'm good with the changes. Are the tests in note 86 now working properly?

No, the first test finishes update only on click inside fill-in widget. The second test does not end wait-for on mouse click in any case.

But I think this issue is something outside the button attributes and GO or ENDKEY feature.

#91 - 04/02/2015 09:53 AM - Greg Shah

Code Review evl_upd20150401a.zip

This is a substantial improvement!

Fix the remaining event processing and I think you are ready to go.

#92 - 04/02/2015 09:50 PM - Eugenie Lyzenko

- File evl_upd20150402a.zip added

The update is the next step fixing client side key loop by 4GL mouse event unlocking. Now the mouse press/click triggers properly unlock the key reader. The remaining issue is the registered trigger is not called on the server side(with return value is null). Looks like the server can not correspond the data passed from the client to the trigger registered on server side.

So continue debugging.

#93 - 04/03/2015 08:48 AM - Greg Shah

Code Review evl_upd20150402a.zip

I think you have found some important things here, but the approach is not correct.

The triggers won't fire because EventList.lookup() just tells you if there is a match for the event + widget + scenario (e.g. exit condition or not...) that is registered in the current list. It doesn't fire the trigger (ThinClient.invokeTriggers() does that.

As with the previous solution, I think we want to avoid the extra trigger and go-pending processing being included in this event reading loop. The important idea here is that the portable mouse events need to be processed synchronously at this point, can we post these events and return?

Constantin: what do you think?

#94 - 04/03/2015 11:21 AM - Eugenie Lyzenko

> I think you have found some important things here, but the approach is not correct.
>
> The triggers won't fire because EventList.lookup() just tells you if there is a match for the event + widget + scenario (e.g. exit condition or not...) that is registered in the current list. It doesn't fire the trigger (ThinClient.invokeTriggers() does that.

Yes, you are right, the trigger is not calling in TC.waitForEvent(). It should be fired before the point I'm looking the exit from key reader. I think it happens in TC.applyWorker() that executes TC.processProgressEvent(). The TC.processProgressEvent() already has the code to handle 4GL mouse events - processing PortableMouseEvent that is generated from native mouse event. This is the place where we call the trigger but for some reasons it is not happening correctly.

> As with the previous solution, I think we want to avoid the extra trigger and go-pending processing being included in this event reading loop. The important idea here is that the portable mouse events need to be processed synchronously at this point, can we post these events and return?

I see.

#95 - 04/03/2015 02:52 PM - Eugenie Lyzenko

- File evl_upd20150403a.zip added

This update for review has fix for mouse related triggers/exit events. The idea is:
1. In TC.waitForEvent() we leave the key reading loop if we have either mouse related trigger in update statement or exit event in wait-for case.
2. In TC.processProgressEvent() the native mouse event is converting to the PortableMouseEvent if the computed event source is the real widget. In the case when the topmost widget is virtual(id < 0 and trigger can not be called) but we need to handle ANYWHERE or CURRENT-WINDOW event location - the generation of the PortableMouseEvent is making with natural native mouse event source(current window).

Please comment this because something I miss can be not correct/not acceptable.

#96 - 04/03/2015 04:50 PM - Greg Shah

Code Review evl_upd20150403a.zip

The changes to TC.processProgressEvent() seem reasonable.

I'm still concerned about the TC.waitForEvent() changes. It is good that the isGoPending is not manipulated. But I think the use of currentEventList.lookup() is not correct. These two cases should not be handled there. I think that the real problem is that the PortableMouseEvent instances are not being processed like normal events. If we take this proposed approach, I think we will constantly be patching this TC.waitForEvent() code to add more cases that are already handled by our normal event processing.

Constantin: please comment.

#97 - 04/04/2015 12:12 PM - Constantin Asofiei

Greg Shah wrote:

> Constantin: please comment.

Eugenie, I have some concerns/questions:

1. in TC.waitForEvent, there is a applyWorker(osEvent); at line 11931. If we execute this AND let the osEvent be handled by the caller (i.e. TC.waitForWorker), wouldn't the trigger execute twice? Once here and once by the applyWorker called by TC.waitForWorker.
2. there are also the window events (i.e. WINDOW-RESIZED, MAXIMIZIED etc), which should act the same way as the mouse events. Thus, if the includeMouseEvent parameter for waitForEvent is set to true, why not do something like this:

```
         if (evt == null && EventManager.hasOSEvents())
         {
            if (!includeMouseEvent)
            {
               // this is the case for readkey/pause/etc which can't act on the portable mouse events or o
n the window events

               // get possible OS event
               Event osEvent = EventManager.getOSEvent();

               // OS events will be treated first
               applyWorker(osEvent);

               // recompute the focus (might have changed)
               comp = computeFocus(frameAsTarget);
               continue;
            }
            else
```

```
            {
               // let the event be processed by the caller
               evt = osEvent;
            }
         }
```

#98 - 04/06/2015 07:18 PM - Eugenie Lyzenko

- File evl_upd20150406a.zip added

The code for review is the next iteration of the mouse event processing fix(may be the final). The changes:
1. The TC.waitForEvent() is cleaned from any widget related additional processing. Only mouse click now considering. Press/release are included in click(see MouseEvt.legacyEvents() for details) so no need to explicitly check in TC.waitForEvent().
2. In TC.processProgressEvent() in event replace code for native->4GL mouse event recursive call is replaced with event posting to more clear preserve event sequence and avoid recursion.
3. In TC.processProgressEvent() handling of PortableMouseEvent is changed(the root cause was here in missing functionality). We can have two events embedded in single one. Example - mouse release which has mouse click inside. And because we can have the triggers for both - we need to execute them sequentially. And here we have to check possible exit conditions for both cases too.

#99 - 04/07/2015 07:36 AM - Greg Shah

Code Review evl_upd20150406a.zip

I like the approach, it seems correct now.

Constantin: any objections?

#100 - 04/07/2015 10:41 AM - Constantin Asofiei

Greg Shah wrote:

> Code Review evl_upd20150406a.zip
>
> I like the approach, it seems correct now.
>
> Constantin: any objections?

This bit of code in TC.waitForEvent looks too specific:

```
            if (includeMouseEvent && osEvent instanceof MouseEvt &&
                ((MouseEvt)osEvent).getEvent().getID() == MouseEvt.MOUSE_CLICKED)
```

Eugenie, have you tried doing this instead of just including the mouse click events?

```
            if (includeMouseEvent) /* better rename this to includeOSEvent, if this works */
            {
```

```
            return osEvent;
        }
```

This will cover any kind of OS events (including window events like minimize/maximize) which can be caught by triggers (and these triggers can in turn affect the wait-for loop/go pending/etc).

#101 - 04/07/2015 11:15 AM - Eugenie Lyzenko

> This bit of code in TC.waitForEvent looks too specific:
>
> ```
>             if (includeMouseEvent && osEvent instanceof MouseEvt &&
>                 ((MouseEvt)osEvent).getEvent().getID() == MouseEvt.MOUSE_CLICKED)
> ```
>
> Eugenie, have you tried doing this instead of just including the mouse click events?
>
> ```
>             if (includeMouseEvent) /* better rename this to includeOSEvent, if this works */
>             {
>                return osEvent;
>             }
> ```

This works fine. But:
1. Are you planning to use OSEvent engine for something other than mouse events?
2. I have not wanted to include mouse move events into event processing path(at least for now). Because it is a huge CPU consumer. Yes, it is possible to define mouse movement trigger in 4GL. But now we don't have support for mouse movement because it is required code to transform native movement into "portable mouse" movement or "three-button mouse" movement. Do we really need to handle mouse move?

#102 - 04/07/2015 11:31 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> This bit of code in TC.waitForEvent looks too specific:
>
> [...]
>
> Eugenie, have you tried doing this instead of just including the mouse click events?
> [...]

> This works fine. But:
> 1. Are you planning to use OSEvent engine for something other than mouse events?

Ah, I've misinterpreted something, the OSEvent engine is not used for legacy window events, is used only for some pseudo-events related to window activation. So yes, regarding legacy events, only mouse events are included in the OSEvent, so only these need to escape TC.waitForEvent.

> 2. I have not wanted to include mouse move events into event processing path(at least for now). Because it is a huge CPU consumer. Yes, it is possible to define mouse movement trigger in 4GL. But now we don't have support for mouse movement because it is required code to transform native movement into "portable mouse" movement or "three-button mouse" movement. Do we really need to handle mouse move?

OK, I understand now. Mouse move doesn't have portable/legacy events, so we don't need to handle them outside TC.waitForEvent, but they are treated by P2J in case hovering or some other action related to mouse move is needed (the applyWorker call in TC.waitForEvent ensure these are processed).

So the only issue remaining to mouse is: allow the MOUSE_PRESSED and MOUSE_RELEASED beside MOUSE_CLICKED to escape TC.waitForEvent (as there are portable mouse events for mouse pressed/released). Otherwise, the changes are fine.

#103 - 04/07/2015 11:59 AM - Eugenie Lyzenko

> So the only issue remaining to mouse is: allow the MOUSE_PRESSED and MOUSE_RELEASED beside MOUSE_CLICKED to escape TC.waitForEvent (as there are portable mouse events for mouse pressed/released). Otherwise, the changes are fine.

It is a good point. Actually I have played with these options yesterday. And there is no difference. The MOUSE_PRESSED and MOUSE_RELEASED

itself do not produce 4GL mouse events. Take a look at MouseEvt.legacyEvents() the standalone mouse press/release are just ignored(because click == 1). The real event generation is happening when P2J get MOUSE_CLICK - the 4GL press and release/click is generating only when click received.

We can include MOUSE_PRESSED and MOUSE_RELEASED into consideration but it is useless I think. Why do we need extra work without any goal?

#104 - 04/07/2015 12:05 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> So the only issue remaining to mouse is: allow the MOUSE_PRESSED and MOUSE_RELEASED beside MOUSE_CLICKED to escape TC.waitForEvent (as there are portable mouse events for mouse pressed/released). Otherwise, the changes are fine.

> It is a good point. Actually I have played with these options yesterday. And there is no difference. The MOUSE_PRESSED and MOUSE_RELEASED itself do not produce 4GL mouse events. Take a look at MouseEvt.legacyEvents() the standalone mouse press/release are just ignored(because click == 1). The real event generation is happening when P2J get MOUSE_CLICK - the 4GL press and release/click is generating only when click received.
>
> We can include MOUSE_PRESSED and MOUSE_RELEASED into consideration but it is useless I think. Why do we need extra work without any goal?

As I recall it's a little tricky to capture a mouse up/down portable event separated from a mouse click/double click. Try this: in 4GL GUI, press the mouse click while over a button, keep it pressed, move it outside of the button and release it. A click should not be generated, only a mouse down. Use this:

```
DEF BUTTON btn LABEL "Aa".

FORM btn WITH FRAME f1.

ON 'mouse-select-down':U OF  btn IN FRAME f1
DO:
   MESSAGE "mouse down!".
   RETURN.
END.

ON 'mouse-select-click':U OF  btn IN FRAME f1
DO:
   MESSAGE "mouse click".
   RETURN.
END.

ENABLE btn WITH FRAME f1.
WAIT-FOR CLOSE OF CURRENT-WINDOW.
```

#105 - 04/07/2015 01:14 PM - Eugenie Lyzenko

- File evl_upd20150407a.zip added

OK. This is the updated change for review has included MOUSE_PRESSED and MOUSE_RELEASED into key reading loop. But to make your sample working we have to do one more change - move mouse down 4GL event generation to the MOUSE_PRESSED handler of the MouseEvt class. Otherwise we will have two mouse down events and one click for single native mouse click.

So Greg, Constantin, let me know please if the change is now OK and I can start testing.

#106 - 04/07/2015 01:21 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> ... But to make your sample working we have to do one more change - move mouse down 4GL event generation to the MOUSE_PRESSED handler of the MouseEvt class. Otherwise we will have two mouse down events and one click for single native mouse click.

Please confirm if this works the same on Windows, too. After this, you can go ahead and test.

#107 - 04/07/2015 01:36 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > ... But to make your sample working we have to do one more change - move mouse down 4GL event generation to the MOUSE_PRESSED handler of the MouseEvt class. Otherwise we will have two mouse down events and one click for single native mouse click.
>
> Please confirm if this works the same on Windows, too. After this, you can go ahead and test.

What do you mean? P2J on Windows?

#108 - 04/07/2015 01:48 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:

> Eugenie Lyzenko wrote:
>
> > ... But to make your sample working we have to do one more change - move mouse down 4GL event generation to the MOUSE_PRESSED handler of the MouseEvt class. Otherwise we will have two mouse down events and one click for single native mouse click.
>
> Please confirm if this works the same on Windows, too. After this, you can go ahead and test.

What do you mean? P2J on Windows?

Yes, P2J on windows.

#109 - 04/07/2015 04:54 PM - Eugenie Lyzenko

> Please confirm if this works the same on Windows, too. After this, you can go ahead and test.

Yes, P2J on Windows works the same.

So I'm starting the regression testing for 0407a update.

#110 - 04/07/2015 08:20 PM - Eugenie Lyzenko

The conversion testing completed. The generated code has a changes but they are expected. It is SetAutoGo() replacing with SetAutoGoOption() and SetAutoEndKey() with SetAutoEndKeyOption(). All the rest is the same.

So I'm starting the runtime testing.

#111 - 04/08/2015 08:39 AM - Eugenie Lyzenko

The main part completed without regressions. Continue with CTRL-C sessions.

#112 - 04/08/2015 09:26 AM - Eugenie Lyzenko

- File evl_upd20150408a.zip added

This is the merge with recent 10831 code base.

Because ThinClient has crossing I'm going to restart runtime testing. The rules was not changed, so conversion testing is not necessary. Is it OK?

#113 - 04/08/2015 10:07 AM - Greg Shah

Code Review evl_upd20150408a.zip

It looks good.

Agreed, no conversion testing is needed.

#114 - 04/08/2015 08:08 PM - Eugenie Lyzenko

The regression testing completed. There are no issues, results - 10831_9aea7ab_20150408_evl.zip.

I'm going to commit and distribute update.

#115 - 04/08/2015 08:19 PM - Eugenie Lyzenko

The update has been committed in bzr as 10832.

#116 - 04/09/2015 07:54 AM - Greg Shah

As far as I understand, all features for this task are finished and working properly. Is that correct?

There are the specially named button images described in notes 84 and 85. I'm going to move that work to another task. We'll close this one if there is nothing left to do here.

#117 - 04/09/2015 08:34 AM - Eugenie Lyzenko

Greg Shah wrote:

> As far as I understand, all features for this task are finished and working properly. Is that correct?

Yes, the only thing left is the BUTTON-*-ARROW predefined images noted in 84 and 85.

> There are the specially named button images described in notes 84 and 85. I'm going to move that work to another task. We'll close this one if there is nothing left to do here.

OK.

#118 - 04/09/2015 09:48 AM - Greg Shah

- Target version set to Milestone 12

- Status changed from New to Closed

The special button arrow images task is #2545.

#119 - 11/16/2016 12:13 PM - Greg Shah

- Target version changed from Milestone 12 to GUI Support for a Complex ADM2 App

#120 - 09/05/2017 12:25 PM - Hynek Cihlar

- Related to Feature #3332: Add support for load offset and load size parameters of LOAD-IMAGE methods of BUTTON and IMAGE widgets added*

#121 - 03/09/2018 10:49 AM - Greg Shah

- Related to Bug #3504: sizing and centering issues with button image loading added

Files

evl_upd20150227a.zip	122 KB	02/27/2015	Eugenie Lyzenko
evl_upd20150303a.zip	128 KB	03/04/2015	Eugenie Lyzenko
evl_upd20150306a.zip	129 KB	03/06/2015	Eugenie Lyzenko
evl_upd20150307a.zip	129 KB	03/07/2015	Eugenie Lyzenko
evl_upd20150311a.zip	55 KB	03/11/2015	Eugenie Lyzenko
evl_upd20150312a.zip	125 KB	03/13/2015	Eugenie Lyzenko
evl_upd20150313a.zip	131 KB	03/13/2015	Eugenie Lyzenko
evl_upd20150316a.zip	131 KB	03/16/2015	Eugenie Lyzenko
evl_upd20150316b.zip	75.2 KB	03/17/2015	Eugenie Lyzenko
evl_upd20150320a.zip	205 KB	03/20/2015	Eugenie Lyzenko
evl_upd20150320b.zip	205 KB	03/20/2015	Eugenie Lyzenko
evl_upd20150324a.zip	205 KB	03/25/2015	Eugenie Lyzenko
evl_upd20150325a.zip	240 KB	03/25/2015	Eugenie Lyzenko
evl_upd20150326a.zip	240 KB	03/26/2015	Eugenie Lyzenko
button_4gl.jpg	167 KB	03/27/2015	Eugenie Lyzenko
button_4gl_laf.jpg	36.4 KB	03/27/2015	Eugenie Lyzenko
evl_upd20150401a.zip	244 KB	04/01/2015	Eugenie Lyzenko
evl_upd20150402a.zip	245 KB	04/03/2015	Eugenie Lyzenko
evl_upd20150403a.zip	245 KB	04/03/2015	Eugenie Lyzenko
evl_upd20150406a.zip	245 KB	04/06/2015	Eugenie Lyzenko
evl_upd20150407a.zip	245 KB	04/07/2015	Eugenie Lyzenko
evl_upd20150408a.zip	245 KB	04/08/2015	Eugenie Lyzenko