User Interface - Feature #2481

Feature # 2252 (Closed): implement GUI client support

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

enable, test and fix clipping for both rectangle and button GUI widgets

01/07/2015 12:55 PM - Greg Shah

Status:	Closed	**Start date:**	01/07/2015
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

History

#1 - 01/07/2015 12:55 PM - Greg Shah

- Parent task set to #2446

#2 - 04/09/2015 09:58 PM - Eugenie Lyzenko

Some interesting investigation results:

During debugging clip/draw code I've found two points to be considered and at least first one is seems to be a kind of bug.

1. The widget coordinate has one pixel skew to left and up from what we have in 4GL. I know it is always expected to be (0,0) for upper left starting point inside the current drawing area and seems like good approach but gives incorrect placement for widget shape. I have faced this feature in both button and rectangle widgets I worked with. The solution to have the same picture as in 4GL - to make starting point as (1,1).

I suggest this is the expected behavior because we draw one pixel black rectangle for window interior at coord (0,0) so in our implementation the area available to draw is (1,1) while in 4GL it is (0,0), so we need to shift every widget to right/bottom for 1 pixel to exactly match 4GL.

2. The java clipping tool has interesting feature. When we specify clipRect(x,y,width,height) the area available to be changed does not include the right and bottom lines of the rectangle. So the actual area available to draw is (x,y,width-1,height-1). This constraint should be taken into account when clipping rectangle is creating and passing to widget's draw() primitive.

#3 - 04/10/2015 05:45 AM - Greg Shah

> 1. The widget coordinate has one pixel skew to left and up from what we have in 4GL. I know it is always expected to be (0,0) for upper left starting point inside the current drawing area and seems like good approach but gives incorrect placement for widget shape. I have faced this feature in both button and rectangle widgets I worked with. The solution to have the same picture as in 4GL - to make starting point as (1,1).
>
> I suggest this is the expected behavior because we draw one pixel black rectangle for window interior at coord (0,0) so in our implementation the area available to draw is (1,1) while in 4GL it is (0,0), so we need to shift every widget to right/bottom for 1 pixel to exactly match 4GL.

Are we just incorrectly reporting the coordinates of the Window's "client area"? The 1-pixel black rectangle should not be part of the client area, but considered part of the window border I think.

#4 - 04/10/2015 11:25 AM - Eugenie Lyzenko

- File rect_test7_p2j_gui0.jpg added

> Are we just incorrectly reporting the coordinates of the Window's "client area"? The 1-pixel black rectangle should not be part of the client area, but considered part of the window border I think.

It is possible, I need to check. I have put the screen from P2J with rectangle frame marked in yellow color to document the subject of discussion.

#5 - 04/10/2015 02:49 PM - Eugenie Lyzenko

It is not Window issue, my explanation was bad. It is Frame issue. If there is no outer box in a frame our coordinates look good(using NO-BOX in DEFINE FRAME statement). If we use boxed frame - we need one pixel shift. Our coordinate calculation approach does not take into account if the frame is boxed or not, assuming any time there is no box.

#6 - 04/10/2015 02:51 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> It is not Window issue, my explanation was bad. It is Frame issue. If there is no outer box in a frame our coordinates look good(using NO-BOX in DEFINE FRAME statement). If we use boxed frame - we need one pixel shift. Our coordinate calculation approach does not take into account if the frame is boxed or not, assuming any time there is no box.

This is fixed in my #2451 work, so don't worry about it. You can use NO-BOX for now, if the 1pixel problem interferes with your clipping work.

#7 - 04/10/2015 02:54 PM - Greg Shah

Does the solution delegate the calculation of the correct drawing space to the container?

#8 - 04/10/2015 03:38 PM - Constantin Asofiei

Greg Shah wrote:

> Does the solution delegate the calculation of the correct drawing space to the container?

The root cause was the fact that the frame's inner scroll pane was not positioned/sized to take into consideration the frame's box status - it was always on the frame's (0,0) coordinate. As all coordinates are relative to parent, all child widgets were miss-placed by 1 pixel.

Eugenie: this is fixed in FrameGuiImpl.resizeScrollPane. Here is the code which places the scroll-pane in the correct location:

```
   private void resizeScrollPane(int captionHeight)
   {
      boolean box = config().box;

      // location
      NativePoint np = new NativePoint(0, captionHeight);
      if (box)
      {
         np.translate(1, 1);
      }

      fs.setPhysicalLocation(np.x, np.y);
      fs.setLocation(cc.pointFromNative(np));
      // size
      NativeDimension nd = physicalDimension();
      nd.width  = nd.width - (box ? 2 : 0);
      nd.height = nd.height - captionHeight - (box ? 2 : 0);
      fs.setSize(cc.dimensionFromNative(nd));
   }
```

#9 - 04/10/2015 04:58 PM - Eugenie Lyzenko

> Eugenie: this is fixed in FrameGuiImpl.resizeScrollPane. Here is the code which places the scroll-pane in the correct location:

Thank you. This gives expected look for boxed frames.

The fixed FrameGuiImpl.resizeScrollPane will be included in your update, so I do not need to include it in mine, correct?

#10 - 04/10/2015 08:36 PM - Eugenie Lyzenko

- File rect_test4_issue_p2j_gui.jpg added

- File evl_upd20150410a.zip added

This update includes rectangle related clipping functionality plus minor clean up. The GuiSimulator also has minor fix in dimension computing. To get the real size(inclusive) we have make right - left + 1, for example if left = 2, right = 3, the dimension should be 2, not 1.

I've found one screen distortion while rectangle drawing in certain conditions. The example:

```
...
define rectangle r size 4 by 4.
define frame f r with size 60 by 12 title "Rectangle Test #4, FG BG COLOR test".

/* Yellow on blue */
r:fgcolor = 13.
r:bgcolor = 1.
/* Color pairs */
r:dcolor = 2.
r:pfcolor = 3.

view frame f.

message "FG, BG, D, PF COLORS: " r:fgcolor r:bgcolor r:dcolor r:pfcolor "Press a key to disable filled.".
pause.

hide all.

r:filled = false.

view frame f.
...
```

The second view displays one phantom rectangle(see the picture attached). For now I can not say if this is rectangle specific or general issue.

The button part test/fix is still in progress.

#11 - 04/11/2015 08:32 AM - Greg Shah

Code Review evl_upd20150410a.zip

I'm OK with the changes.

#12 - 04/11/2015 07:13 PM - Eugenie Lyzenko

- File evl_upd20150411a.zip added

The update for review adds fix for UI screen phantom rectangle issue. The TC.viewWorker() needs to use indirect repaint() instead of draw() for widget to be refreshed.

#13 - 04/13/2015 06:54 PM - Eugenie Lyzenko

- File evl_upd20150413a.zip added

The update for review adds clipping feature tested and multiple minor fixes taking into account changed widget coordinates base (0,0). Confirmed to work with button tests including dynamic buttons and image based buttons.

Also TC.drawAffectedWidgets() method changes the widget drawing by replacing draw() with repaint(). This fixes one more phantom painting like for rectangle widget. This is happening when button widget become enabled/disabled. Does someone else note the similar artifacts in other places? For example in fill-in drawing. May be we ca not safely call draw() method from TC code.

So this is candidate for testing if there are no notes/objections.

#14 - 04/14/2015 04:50 AM - Constantin Asofiei

Eugenie, from the update, the only changes I'm not OK with are the TC changes:

1. drawAffectedWidgets - this is not called in a TC.eventDrawingBracket, it's called in a TC.eventBracket(true,...) - so the repaint()'s PaintEvent will not be picked up and processed by the event manager loop, it will be discarded.
2. viewWorker - this is not even called in a TC.event, so is not even possible to capture this event. But the original draw() call is bracketed by widgetDrawingOn and widgetDrawingOff calls, which forces output sync, to let any drawing done inside these brackets end up on screen.

So: TC does rely on explicitly called draw(), at least when used in conjunction with widgetDrawingOn and widgetDrawingOff. And we can't easily replace these with repaint() calls, without a more in depth analysis of what the widgetDrawingOn and widgetDrawingOff are supposed to fix.

#15 - 04/14/2015 08:46 AM - Eugenie Lyzenko

- File button19_1_p2j_gui_issue.jpg added

Constantin Asofiei wrote:

> Eugenie, from the update, the only changes I'm not OK with are the TC changes:
>
> 1. drawAffectedWidgets - this is not called in a TC.eventDrawingBracket, it's called in a TC.eventBracket(true,...) - so the repaint()'s PaintEvent will not be picked up and processed by the event manager loop, it will be discarded.
> 2. viewWorker - this is not even called in a TC.event, so is not even possible to capture this event. But the original draw() call is bracketed by widgetDrawingOn and widgetDrawingOff calls, which forces output sync, to let any drawing done inside these brackets end up on screen.
>
> So: TC does rely on explicitly called draw(), at least when used in conjunction with widgetDrawingOn and widgetDrawingOff. And we can't easily replace these with repaint() calls, without a more in depth analysis of what the widgetDrawingOn and widgetDrawingOff are supposed to fix.

The fact is: using draw() call in both cases causes the screen distortions(at least in GUI mode). And I think in other cases too. During fill-in implementation didn't you see the same issue as on the picture attached?

#16 - 04/14/2015 08:52 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Constantin Asofiei wrote:
>
> > Eugenie, from the update, the only changes I'm not OK with are the TC changes:
> >
> > 1. drawAffectedWidgets - this is not called in a TC.eventDrawingBracket, it's called in a TC.eventBracket(true,...) - so the repaint()'s PaintEvent will not be picked up and processed by the event manager loop, it will be discarded.
> > 2. viewWorker - this is not even called in a TC.event, so is not even possible to capture this event. But the original draw() call is bracketed by widgetDrawingOn and widgetDrawingOff calls, which forces output sync, to let any drawing done inside these brackets end up on screen.
> >
> > So: TC does rely on explicitly called draw(), at least when used in conjunction with widgetDrawingOn and widgetDrawingOff. And we can't easily replace these with repaint() calls, without a more in depth analysis of what the widgetDrawingOn and widgetDrawingOff are supposed to fix.
>
> The fact is: using draw() call in both cases causes the screen distortions(at least in GUI mode). And I think in other cases too. During fill-in implementation didn't you see the same issue as on the picture attached?

No, I haven't encountered this issue. Can you tell me the test and steps to recreate this? Although the drawing in GUI is done using relative coordinates, the GuiDriver.draw ensures the coordinates are translated properly, so the drawn/clipping region matches the coordinate/size of the widget being drawn.

#17 - 04/14/2015 09:04 AM - Eugenie Lyzenko

> No, I haven't encountered this issue. Can you tell me the test and steps to recreate this?

This is the uast/button/gui_btn_test19_2.p. Just put focus inside fill-in and type something:

```
...
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.
...
```

> Although the drawing in GUI is done using relative coordinates, the GuiDriver.draw ensures the coordinates are translated properly, so the drawn/clipping region matches the coordinate/size of the widget being drawn.

Yes, I know, this should go this way. But sometimes coordinate translation become inconsistent.

#18 - 04/14/2015 09:19 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Yes, I know, this should go this way. But sometimes coordinate translation become inconsistent.

Do you see this with other widgets beside FILL-IN? Because looks like with my frame layout changes in #2451 (pending work) what you noticed is no longer an issue.

#19 - 04/14/2015 09:24 AM - Eugenie Lyzenko

> Do you see this with other widgets beside FILL-IN? Because looks like with my frame layout changes in #2451 (pending work) what you noticed is no longer an issue.

I see this in rectangle, button and fill-in. The key point is: you need the frame containing the widgets to be titled and boxed.

#20 - 04/14/2015 09:36 AM - Constantin Asofiei

Eugenie Lyzenko wrote:

> > Do you see this with other widgets beside FILL-IN? Because looks like with my frame layout changes in #2451 (pending work) what you noticed is no longer an issue.
>
> I see this in rectangle, button and fill-in. The key point is: you need the frame containing the widgets to be titled and boxed.

Do you have a recreate for rectangle/button, with the same (or similar) mis-placement as the fill-in? I can't find anything problematic with buttons in gui_btn_test19_2.p.

#21 - 04/14/2015 09:56 AM - Eugenie Lyzenko

> Do you have a recreate for rectangle/button, with the same (or similar) mis-placement as the fill-in? I can't find anything problematic with buttons in gui_btn_test19_2.p.

The issue is triggering by certain conditions, different for different widgets.

For rectangle the testcases is: gui_btn_test0.p, press the "Fill on/off" and "Round on/off" buttons to see.

For button itself the good testcases is: gui_btn_test5.p, press "Enable on/off" button to see effect.

I have updated all button/rectangle testcases in bzr test repo 1266, so you can download from there.

If you do not see any UI distortions(meaning fixed with your work) - it is good, I just remove the ThinClient from update and can start regression testing.

#22 - 04/14/2015 12:35 PM - Eugenie Lyzenko

Sorry for rectangle it is better to use rect_test4.p, after filling become off.

#23 - 04/14/2015 12:55 PM - Eugenie Lyzenko

- File evl_upd20150414a.zip added

The update has modified TC to brace widget painting inside TC.eventDrawingBracket() for both cases, viewWorker() and drawAffectedWidgets(). How about this?

#24 - 04/14/2015 01:20 PM - Constantin Asofiei

Eugenie, the tests you mention work fine with my #2451 work, and are incorrect without it: so this problem was related to some layout issue.

So, the plan is: remove the TC changes, add the FrameGuiImpl.resizeScrollPane change I mentioned in note 8 (we can benefit from it now), and if all changes are in the GUI package, you can release it without testing, if Greg is OK with them, too.

#25 - 04/14/2015 01:29 PM - Eugenie Lyzenko

Constantin Asofiei wrote:

> So, the plan is: remove the TC changes, add the FrameGuiImpl.resizeScrollPane change I mentioned in note 8 (we can benefit from it now), and if all changes are in the GUI package, you can release it without testing, if Greg is OK with them, too.

Another word neither FrameGuiImpl(the updates will come from you) not ThinClient are included in my update, correct?

And yes, all my changes are in ui.client.gui package.

#26 - 04/14/2015 01:32 PM - Constantin Asofiei

Eugenie Lyzenko wrote:

> Another word neither FrameGuiImpl(the updates will come from you) not ThinClient are included in my update, correct?

Yes, TC must not be included in your update. But do include in your update the FrameGuiImpl change I mentioned in note 8.

#27 - 04/14/2015 01:47 PM - Eugenie Lyzenko

- File evl_upd20150414b.zip added

The repacked update for review with modified FrameGuiImpl and removed ThinClient.

#28 - 04/14/2015 02:18 PM - Greg Shah

Code Review evl_upd20150414b.zip

I'm good with the changes. Please go ahead and check them in since they only affect GUI and won't be tested by MAJIC regression testing.

#29 - 04/14/2015 02:48 PM - Eugenie Lyzenko

0414b committed in bzr as 10834. Will be distributed shortly.

#30 - 04/14/2015 03:16 PM - Greg Shah

- Status changed from New to Closed

#31 - 11/16/2016 12:13 PM - Greg Shah

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

Files

rect_test7_p2j_gui0.jpg	39.5 KB	04/10/2015	Eugenie Lyzenko
rect_test4_issue_p2j_gui.jpg	37.4 KB	04/11/2015	Eugenie Lyzenko
evl_upd20150410a.zip	14 KB	04/11/2015	Eugenie Lyzenko
evl_upd20150411a.zip	141 KB	04/11/2015	Eugenie Lyzenko
evl_upd20150413a.zip	149 KB	04/13/2015	Eugenie Lyzenko
button19_1_p2j_gui_issue.jpg	31.6 KB	04/14/2015	Eugenie Lyzenko
evl_upd20150414a.zip	149 KB	04/14/2015	Eugenie Lyzenko
evl_upd20150414b.zip	24.9 KB	04/14/2015	Eugenie Lyzenko