User Interface - Bug #2758

Bug # 2677 (New): fix drawing and functional differences between P2J GUI and 4GL GUI

toggle-box selection highlight color depends on the widget's text color and background color

10/14/2015 08:19 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Vadim Gindin	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**case_num:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Feature #2513: implement GUI runtime support for ...	**Closed**	**02/10/2015**

History

#1 - 10/14/2015 08:23 AM - Greg Shah

After 2564b is merged to trunk, this task can be started.

For details, see #2513-141 (and 142), #2513-146 (and 147) and #2513-149 (through 153).

The color does seem to get picked using XOR depending on the widget's background and/or text. 2564b has XOR compositing support, which will work for the background color scenario. But for the color override based on the text color not being black, it might make sense to calculate the color directly and then temporarily just set that color. This is because to use the XOR compositing you would have to draw the selection highlight twice (first in the text color then then again after XOR compositing is turned on). That is inefficient and really doesn't make sense when we can just calculate the right color and draw once.

#2 - 10/14/2015 08:26 AM - Stanislav Lomany

Correction: 2564 A

#3 - 10/20/2015 10:20 AM - Greg Shah

This can now be fixed using the XOR support that exists in the 1811s rev 10995 and later. Changes should go into 1811s.

#4 - 10/22/2015 07:44 AM - Vadim Gindin

XOR (192,192,192) does not suit. It seems XOR bgColor is good. I'm testing

#5 - 10/22/2015 03:02 PM - Vadim Gindin

- File highlight2.png added

- File highlight1.png added

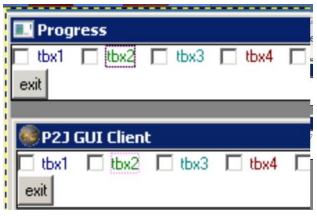
- File highlight_background1.png added

Lets assume we have the following procedure:

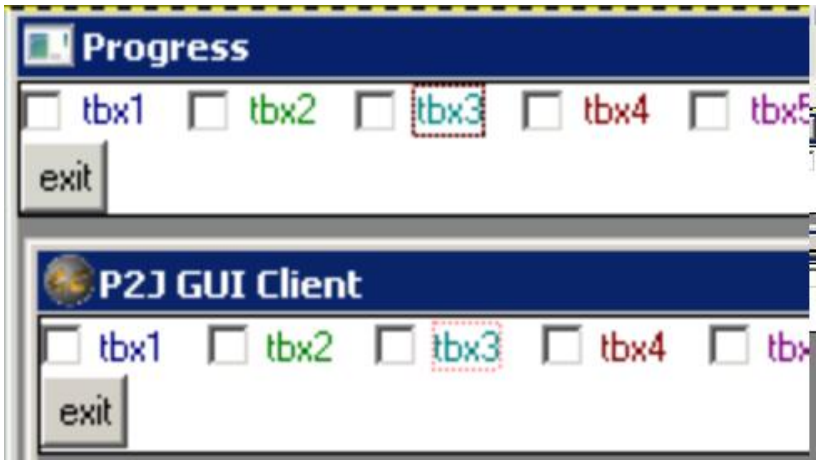
```
def var tbx1 as logical view-as toggle-box .
```

```
def var tbx2 as logical view-as toggle-box size 15 by 1.5.
def var tbx3 as logical view-as toggle-box .
def var tbx4 as logical view-as toggle-box .
def var tbx5 as logical view-as toggle-box .
def var tbx6 as logical view-as toggle-box .
def var tbx7 as logical view-as toggle-box .
def var tbx8 as logical view-as toggle-box .

def button exit_btn label "exit".

def frame f
  tbx1
  tbx2
  tbx3
  tbx4
  tbx5
  tbx6
  tbx7
  tbx8
  exit_btn no-labels.
display tbx1 view-as toggle-box /*size 10 by 1.6*/ with frame f.

/*
tbx1:bgcolor = 1.
tbx2:bgcolor = 2.
tbx3:bgcolor = 3.
tbx4:bgcolor = 4.
tbx5:bgcolor = 5.
tbx6:bgcolor = 6.
tbx7:bgcolor = 7.
*/

tbx1:fgcolor = 1.
tbx2:fgcolor = 2.
tbx3:fgcolor = 3.
tbx4:fgcolor = 4.
tbx5:fgcolor = 5.
tbx6:fgcolor = 6.
tbx7:fgcolor = 7.

enable all with frame f.
wait-for choose of exit_btn.
```

XOR bgColor is not good. I've tried XOR fgColor. It gave good results for cases when bgColor is not set. Have a look:

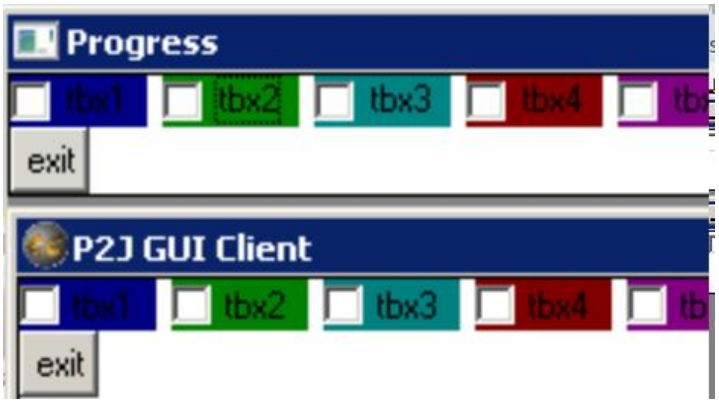


and



It looks like there are to same rectangles: the first is just black and solid and the second - is dotted and colored. Am I right?

XOR fgColor is bad for cases when bgColor is specified instead of fgColor:



Note that highlight is not drawn at all or it is drawn using the same color as background.

Could you help me describe this situation.

#6 - 10/22/2015 04:02 PM - Greg Shah

> It looks like there are to same rectangles: the first is just black and solid and the second - is dotted and colored. Am I right?

I think it is close, but not right.

The rules as I see them:

1. If no explicit BGCOLOR is set.

a) XOR the FGCOLOR and use that to draw a solid highlight rectangle.
b) Change the color to black and draw the dotted highlight over the same rectangle.

2. If BGCOLOR is explicitly set.

a) Change the color to black and draw the dotted highlight over the same rectangle.

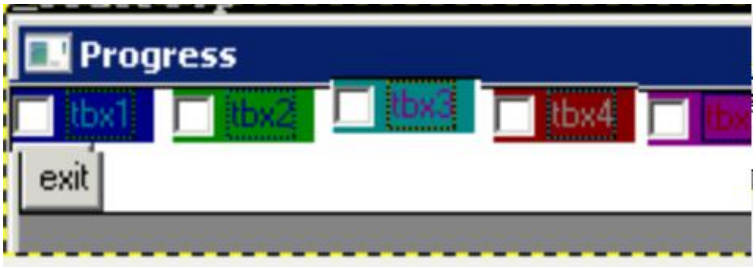
My only question is whether in case 2 (BGCOLOR is set) if the dotted highlight is always drawn in black OR if it draws in the same color as the FGCOLOR. Your testcase doesn't show that.

#7 - 10/22/2015 05:20 PM - Vadim Gindin

- File highlight_both1.png added

- File highlight_both2.png added

I've set fgColor and bgColor in a random order. Here are screen shots:



#8 - 10/22/2015 06:06 PM - Greg Shah

It looks to me that the dotted highlight is always drawn in black, no matter what the FGCOLOR or BGCOLOR.

So the rule seems simple:

1. If no explicit BGCOLOR is set, XOR the FGCOLOR and use that to draw a solid highlight rectangle.
2. No matter what happened in step 1, draw a dotted highlight rectangle in black.

Please implement this and see if it matches.

#9 - 10/23/2015 02:53 PM - Vadim Gindin

It doesn't match. Have a look at tbx3 in previous images for example. It looks like in the case when both background and foreground colours are set - some ternary operation is needed. What do you think?

#10 - 10/24/2015 11:16 AM - Greg Shah

Vadim Gindin wrote:

> It doesn't match. Have a look at tbx3 in previous images for example. It looks like in the case when both background and foreground colours are set - some ternary operation is needed. What do you think?

1. Please show screen capture of the P2J results from your test. I'd like to see how close we are.

2. For each of the failing cases, please post the following:

- FGCOLOR in RGB
- BGCOLOR in RGB
- "solid" portion of the highlight color in RGB

Based on this, hopefully some pattern will emerge.

#11 - 10/24/2015 04:00 PM - Vadim Gindin

Greg, I can find out FGCOLOR and BGCOLOR using PROGRESS functions GET-RED (GREEN | BLUE)-VALUE (index), where index=1..7 from my test procedure, but how would you propose to calculate solid highlight rectangle color. Do you mean to use for that the same technique, used to find component colors, i.e. using "special" program calling MSDN functions?

#12 - 10/25/2015 08:00 AM - Greg Shah

> Do you mean to use for that the same technique, used to find component colors, i.e. using "special" program calling MSDN functions?

No.

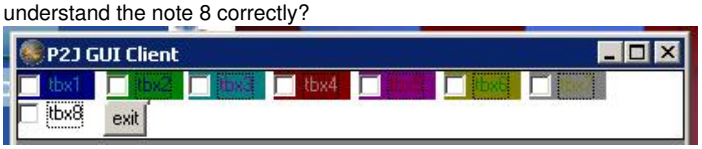
Are the colors you are capturing in your screen-shots accurate? If so, you can use gimp (or any other image editor). Such tools have a "color picker" tool to use the mouse to select a pixel and it will report the exact RGB values.

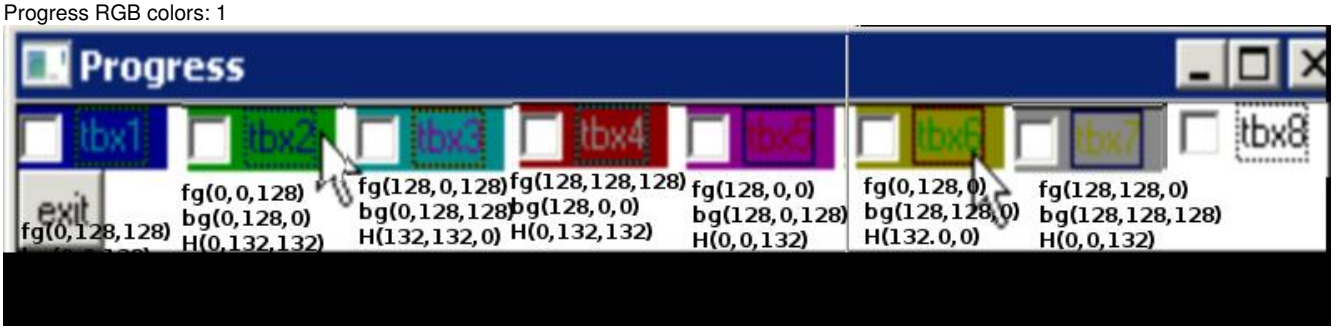
The important thing is that the image it is working from must have the colors accurately captured. In other words, the pixels should not be interpolated or transformed in any way. The best way to ensure this is to use a Windows tool on windev01 for the screen capture. Then transfer it locally and look at the colors.

#13 - 10/26/2015 03:29 AM - Vadim Gindin

- File p2j_highlight.png added

- File rgb_hihglight.png added

1. P2J result
   Solid highlight colors always matches to bgColor because in case when bgColor is specified we do not calculate and draw solid rectangle. Did I understand the note 8 correctly?
   

2. Progress RGB colors: 1
   

   fg - foreground color
   bg - background color
   H - solid part of Highlighted rectangle.

#14 - 10/27/2015 09:26 AM - Greg Shah

> Solid highlight colors always matches to bgColor because in case when bgColor is specified we do not calculate and draw solid rectangle. Did I understand the note 8 correctly?

No, you are correct. The algorithm as described in note 8 is too simple.

> ...
> fg - foreground color
> bg - background color
> H - solid part of Highlighted rectangle.

A pattern is starting to emerge but your data inputs are too regular for us to get the answer.

They are comparing the each separate color element (r, g and b) from fg and bg. If the red value from fg is the same as the red value from bg, then the result is 0. If they are different, then they will transform the existing red values in some way. They do the same thing for green (fg vs bg) and blue (fg vs bg) values.

From one of your examples:

Color	Red	Green	Blue
fg	0	128	0
bg	128	128	0
H	132	0	0

The red values are different so the result is transformed. The green and blue values are the same and so the result is 0.

The problem here is that you have only used 128 and 0 for your colors. We can't tell how the transformation works because the data is too simple. Please use a wider set of variations (0, 31, 32, 63, 64, 95, 96, 127, 128, 159, 160, 191, 192, 223, 224, 255). Make sure that the values that are different are not always 0 and a positive number. Each of the above numbers should be checked against all the others (0 against 31, 32, 63... + 31 against 32, 63, 64, ... + 32 against 63, 64, 96... and so on).

Also, please post your data formatted as a table here in Redmine instead of hard coded into an image. Text in a table can be searched and read easily but the image requires someone to put their eyes on it.

#15 - 10/27/2015 03:07 PM - Vadim Gindin

Greg Shah wrote:

> ..
> The problem here is that you have only used 128 and 0 for your colors. We can't tell how the transformation works because the data is too simple. Please use a wider set of variations (0, 31, 32, 63, 64, 95, 96, 127, 128, 159, 160, 191, 192, 223, 224, 255). Make sure that the values that are different are not always 0 and a positive number. Each of the above numbers should be checked against all the others (0 against 31, 32, 63... + 31 against 32, 63, 64, ... + 32 against 63, 64, 96... and so on).

Could you clarify couple of things:

1. not always 0 and a positive number - do you mean that color component (RED for example) can have a negative value?
2. Each of the above numbers should be checked against all the others (0 against 31, 32, 63... + 31 against 32, 63, 64, ... + 32 against 63, 64, 96... and so on) - do you mean, that in the formula i against j i < j always or any i and j from specified set?
3. @ There are a lot of variants: (3 * 16) ^ 2 = 2304 (any i and j) and I will need in every of them check the result manually using GIMP. Is there any other automated way to check resulting color?

#16 - 10/27/2015 03:21 PM - Vadim Gindin

Vadim Gindin wrote:

> ..
>
> 1. @ There are a lot of variants: (3 * 16) ^ 2 = 2304 (any i and j) and I will need in every of them check the result manually using GIMP. Is there any other automated way to check resulting color?

Sorry, there are even more variants: 16 ^ 6 = 16777216

#17 - 10/27/2015 04:34 PM - Greg Shah

- File combos.java added

> not always 0 and a positive number - do you mean that color component (RED for example) can have a negative value?

No. I mean that for most cases, both numbers (e.g. fg:red and bg:red) should be non-zero.

> Each of the above numbers should be checked against all the others (0 against 31, 32, 63... + 31 against 32, 63, 64, ... + 32 against 63, 64, 96... and so on) - do you mean, that in the formula i against j i < j always or any i and j from specified set?
> There are a lot of variants: (3 * 16) ^ 2 = 2304

It is a good question. I don't think that there is any color-specific logic going on here. We can use all 3 values of each color as a "slot". I think 256 combinations will create 86 variants:

```
Total = 86
fg (0, 0, 0) + bg (0, 31, 32)
fg (0, 0, 0) + bg (63, 64, 95)
fg (0, 0, 0) + bg (96, 127, 128)
fg (0, 0, 0) + bg (159, 160, 191)
fg (0, 0, 0) + bg (192, 223, 224)
fg (0, 31, 31) + bg (255, 0, 31)
fg (31, 31, 31) + bg (32, 63, 64)
fg (31, 31, 31) + bg (95, 96, 127)
fg (31, 31, 31) + bg (128, 159, 160)
fg (31, 31, 31) + bg (191, 192, 223)
fg (31, 31, 32) + bg (224, 255, 0)
fg (32, 32, 32) + bg (31, 32, 63)
fg (32, 32, 32) + bg (64, 95, 96)
fg (32, 32, 32) + bg (127, 128, 159)
fg (32, 32, 32) + bg (160, 191, 192)
fg (32, 32, 32) + bg (223, 224, 255)
fg (63, 63, 63) + bg (0, 31, 32)
fg (63, 63, 63) + bg (63, 64, 95)
fg (63, 63, 63) + bg (96, 127, 128)
fg (63, 63, 63) + bg (159, 160, 191)
fg (63, 63, 63) + bg (192, 223, 224)
fg (63, 64, 64) + bg (255, 0, 31)
fg (64, 64, 64) + bg (32, 63, 64)
fg (64, 64, 64) + bg (95, 96, 127)
fg (64, 64, 64) + bg (128, 159, 160)
fg (64, 64, 64) + bg (191, 192, 223)
```

```
fg (64, 64, 95) + bg (224, 255, 0)
fg (95, 95, 95) + bg (31, 32, 63)
fg (95, 95, 95) + bg (64, 95, 96)
fg (95, 95, 95) + bg (127, 128, 159)
fg (95, 95, 95) + bg (160, 191, 192)
fg (95, 95, 95) + bg (223, 224, 255)
fg (96, 96, 96) + bg (0, 31, 32)
fg (96, 96, 96) + bg (63, 64, 95)
fg (96, 96, 96) + bg (96, 127, 128)
fg (96, 96, 96) + bg (159, 160, 191)
fg (96, 96, 96) + bg (192, 223, 224)
fg (96, 127, 127) + bg (255, 0, 31)
fg (127, 127, 127) + bg (32, 63, 64)
fg (127, 127, 127) + bg (95, 96, 127)
fg (127, 127, 127) + bg (128, 159, 160)
fg (127, 127, 127) + bg (191, 192, 223)
fg (127, 127, 128) + bg (224, 255, 0)
fg (128, 128, 128) + bg (31, 32, 63)
fg (128, 128, 128) + bg (64, 95, 96)
fg (128, 128, 128) + bg (127, 128, 159)
fg (128, 128, 128) + bg (160, 191, 192)
fg (128, 128, 128) + bg (223, 224, 255)
fg (159, 159, 159) + bg (0, 31, 32)
fg (159, 159, 159) + bg (63, 64, 95)
fg (159, 159, 159) + bg (96, 127, 128)
fg (159, 159, 159) + bg (159, 160, 191)
fg (159, 159, 159) + bg (192, 223, 224)
fg (159, 160, 160) + bg (255, 0, 31)
fg (160, 160, 160) + bg (32, 63, 64)
fg (160, 160, 160) + bg (95, 96, 127)
fg (160, 160, 160) + bg (128, 159, 160)
fg (160, 160, 160) + bg (191, 192, 223)
fg (160, 160, 191) + bg (224, 255, 0)
fg (191, 191, 191) + bg (31, 32, 63)
fg (191, 191, 191) + bg (64, 95, 96)
fg (191, 191, 191) + bg (127, 128, 159)
fg (191, 191, 191) + bg (160, 191, 192)
fg (191, 191, 191) + bg (223, 224, 255)
fg (192, 192, 192) + bg (0, 31, 32)
fg (192, 192, 192) + bg (63, 64, 95)
fg (192, 192, 192) + bg (96, 127, 128)
fg (192, 192, 192) + bg (159, 160, 191)
fg (192, 192, 192) + bg (192, 223, 224)
fg (192, 223, 223) + bg (255, 0, 31)
fg (223, 223, 223) + bg (32, 63, 64)
fg (223, 223, 223) + bg (95, 96, 127)
fg (223, 223, 223) + bg (128, 159, 160)
fg (223, 223, 223) + bg (191, 192, 223)
fg (223, 223, 224) + bg (224, 255, 0)
fg (224, 224, 224) + bg (31, 32, 63)
fg (224, 224, 224) + bg (64, 95, 96)
fg (224, 224, 224) + bg (127, 128, 159)
fg (224, 224, 224) + bg (160, 191, 192)
fg (224, 224, 224) + bg (223, 224, 255)
fg (255, 255, 255) + bg (0, 31, 32)
fg (255, 255, 255) + bg (63, 64, 95)
fg (255, 255, 255) + bg (96, 127, 128)
fg (255, 255, 255) + bg (159, 160, 191)
fg (255, 255, 255) + bg (192, 223, 224)
fg (255, 0, 0) + bg (255, 0, 0)
```

You can tweak my combos.java to use it to generate 4GL code...

#18 - 10/30/2015 04:33 AM - Vadim Gindin

- File highlight_colors_test.png added

- File rgb.p added

Greg, I've wrote the highlights colors test using your class. Please have a look at it: uast/toggle_box/gui/rgb.p and here is how it looks running: 

.
Is it correct?

P.S. Sorry for stupid question, but why did you selected exactly that colors components set: (0, 31, 32, 63, 64, 95, 96, 127, 128, 159, 160, 191, 192, 223, 224, 255) and why did you selected only combinations specified in the note 17?

#19 - 10/30/2015 08:01 AM - Greg Shah

I didn't check all numbers, but generally the code looks right.

> but why did you selected exactly that colors components set: (0, 31, 32, 63, 64, 95, 96, 127, 128, 159, 160, 191, 192, 223, 224, 255)

I was trying to get a good range of bit patterns, without generating all possible bit patterns.

Bit Pattern	Decimal Value
00000000	0
00011111	31
00100000	32
00111111	63
01000000	64
00101111	95
01100000	96
01111111	127
10000000	128
10011111	159
10100000	160
10111111	191
11000000	192
11011111	223
11100000	224
11111111	255

There is a good argument that we should add some values in the low end (1, 2, 3, 4, 15, 16), but I hope the already chosen values will do the job.

The reason for thinking about this in bit patterns is that there is likely to be some kind of bitwise operation that is applied consistently to the FB/BG colors. In browse, they XOR 192 to get the result. Here it is something different, but probably not too different.

Find out the resulting highlight colors and record the table here (it should include the FG:RGB, BG:RGB and HIGHLIGHT:RGB). Thenb analyze the bit patterns to see if it is clear what is happening. Try to find a single value that when applied with a bitwise operator like XOR, can duplicate all the results.

#20 - 10/30/2015 01:00 PM - Vadim Gindin

Here is the table of highlight rectangle colors depending on foreground and background toggle-box colors.

Foreground	Background	Highlight
(0, 0, 0)	(0, 31, 32)	(0, 31, 32)
(0, 0, 0)	(63, 64, 95)	(63, 64, 95)
(0, 0, 0)	(96, 127, 128)	(96, 127, 128)
(0, 0, 0)	(159, 160, 191)	(159, 160, 191)
(0, 0, 0)	(192, 223, 224)	(192, 223, 224)
(0, 31, 31)	(255, 0, 31)	(255, 0, 31)
(31, 31, 31)	(32, 63, 64)	(63, 32, 95)
(31, 31, 31)	(95, 96, 127)	(64, 127, 96)
(31, 31, 31)	(128, 159, 160)	(159, 128, 191)
(31, 31, 31)	(191, 192, 223)	(160, 223, 192)
(31, 31, 32)	(224, 255, 0)	(255, 224, 32)
(32, 32, 32)	(31, 32, 63)	(63, 0, 31)
(32, 32, 32)	(64, 95, 96)	(96, 127, 64)
(32, 32, 32)	(127, 128, 159)	(95, 160, 191)
(32, 32, 32)	(160, 191, 192)	(128, 159, 224)
(32, 32, 32)	(223, 224, 255)	(255, 192, 223)
(63, 63, 63)	(0, 31, 32)	(63, 32, 31)
(63, 63, 63)	(63, 64, 95)	(0, 127, 96)
(63, 63, 63)	(96, 127, 128)	(95, 64, 191)
(63, 63, 63)	(159, 160, 191)	(160, 159, 128)
(63, 63, 63)	(192, 223, 224)	(255, 224, 223)
(63, 64, 64)	(255, 0, 31)	(192, 64, 95)
(64, 64, 64)	(32, 63, 64)	(96, 127, 0)
(64, 64, 64)	(95, 96, 127)	(31, 32, 63)
(64, 64, 64)	(128, 159, 160)	(192, 223, 224)
(64, 64, 64)	(191, 192, 223)	(255, 128, 159)
(64, 64, 95)	(224, 255, 0)	(160, 191, 95)
(95, 95, 95)	(31, 32, 63)	(64, 127, 96)
(95, 95, 95)	(64, 95, 96)	(31, 0, 63)
(95, 95, 95)	(127, 128, 159)	(32, 223, 192)
(95, 95, 95)	(160, 191, 192)	(255, 224, 159)
(95, 95, 95)	(223, 224, 255)	(128, 191, 160)
(96, 96, 96)	(0, 31, 32)	(96, 127, 64)
(96, 96, 96)	(63, 64, 95)	(95, 32, 63)
(96, 96, 96)	(96, 127, 128)	(0, 31, 224)
(96, 96, 96)	(159, 160, 191)	(255, 192, 223)
(96, 96, 96)	(192, 223, 224)	(160, 191, 128)
(96, 127, 127)	(255, 0, 31)	(159, 127, 96)
(127, 127, 127)	(32, 63, 64)	(95, 64, 63)
(127, 127, 127)	(95, 96, 127)	(32, 31, 0)
(127, 127, 127)	(128, 159, 160)	(255, 224, 223)
(127, 127, 127)	(191, 192, 223)	(192, 191, 160)
(127, 127, 128)	(224, 255, 0)	(159, 128, 128)
(128, 128, 128)	(31, 32, 63)	(159, 160, 191)
(128, 128, 128)	(64, 95, 96)	(192, 223, 224)

(128, 128, 128)	(127, 128, 159)	(255, 0, 31)
(128, 128, 128)	(160, 191, 192)	(32, 63, 64)
(128, 128, 128)	(223, 224, 255)	(95, 96, 127)
(159, 159, 159)	(0, 31, 32)	(159, 128, 191)
(159, 159, 159)	(63, 64, 95)	(160, 223, 192)
(159, 159, 159)	(96, 127, 128)	(255, 224, 31)
(159, 159, 159)	(159, 160, 191)	(0, 63, 32)
(159, 159, 159)	(192, 223, 224)	(95, 64, 127)
(159, 160, 160)	(255, 0, 31)	(96, 160, 191)
(160, 160, 160)	(32, 63, 64)	(128, 159, 224)
(160, 160, 160)	(95, 96, 127)	(255, 192, 223)
(160, 160, 160)	(128, 159, 160)	(32, 63, 0)
(160, 160, 160)	(191, 192, 223)	(31, 96, 127)
(160, 160, 191)	(224, 255, 0)	(64, 95, 191)
(191, 191, 191)	(31, 32, 63)	(160, 159, 128)
(191, 191, 191)	(64, 95, 96)	(255, 224, 223)
(191, 191, 191)	(127, 128, 159)	(192, 63, 32)
(191, 191, 191)	(160, 191, 192)	(31, 0, 127)
(191, 191, 191)	(223, 224, 255)	(96, 95, 64)
(192, 192, 192)	(0, 31, 32)	(192, 223, 224)
(192, 192, 192)	(63, 64, 95)	(255, 128, 159)
(192, 192, 192)	(96, 127, 128)	(160, 191, 64)
(192, 192, 192)	(159, 160, 191)	(95, 96, 127)
(192, 192, 192)	(192, 223, 224)	(0, 31, 32)
(192, 223, 223)	(255, 0, 31)	(63, 223, 192)
(223, 223, 223)	(32, 63, 64)	(255, 224, 159)
(223, 223, 223)	(95, 96, 127)	(128, 191, 160)
(223, 223, 223)	(128, 159, 160)	(95, 64, 127)
(223, 223, 223)	(191, 192, 223)	(96, 31, 0)
(223, 223, 224)	(224, 255, 0)	(63, 32, 224)
(224, 224, 224)	(31, 32, 63)	(255, 192, 223)
(224, 224, 224)	(64, 95, 96)	(160, 191, 128)
(224, 224, 224)	(127, 128, 159)	(159, 96, 127)
(224, 224, 224)	(160, 191, 192)	(64, 95, 32)
(224, 224, 224)	(223, 224, 255)	(63, 0, 31)
(255, 255, 255)	(0, 31, 32)	(255, 224, 223)
(255, 255, 255)	(63, 64, 95)	(192, 191, 160)
(255, 255, 255)	(96, 127, 128)	(159, 128, 127)
(255, 255, 255)	(159, 160, 191)	(96, 95, 64)
(255, 255, 255)	(192, 223, 224)	(63, 32, 31)
(255, 0, 0)	(255, 0, 0)	(0, 0, 0)

#21 - 10/30/2015 03:01 PM - Greg Shah

OK, well the algorithm is seems very simple. It is indeed XOR based. For all of the entries I tested (except for one, see below), the algorithm is this:

h:red = fg:red XOR bg:red
h:green = fg:green XOR bg:green
h:blue = fg:blue XOR bg:blue

The only exception is this entry:

(0, 31, 31)	(255, 0, 31)	(255, 0, 31)

Could the data entry on this one be wrong? It is the only one that I found which doesn't match.

Please check this one.

Also please use this algorithm to calculate the values and compare them with the results to confirm it is the same.

#22 - 11/01/2015 10:07 AM - Vadim Gindin

You're right:

(0, 31, 31)	(255, 0, 31)	(255, 0, 31)

That's incorrect data. Correct result color is (255, 31, 0). I've set your formula and checked result. It works. Thank you! I've committed this change with revno 10997.

#23 - 11/02/2015 06:53 AM - Greg Shah

You have visually tested the results in P2J, comparing side by side with the 4GL?

My primary concern with the use of the SET_XOR_COMPOSITE and RESET_COMPOSITE is that these are deeply coded into sensitive code in the drivers. In your case, it seems like we really don't need to do that. In the browse highlight usage, we don't know what we are overdrawing so making the windowing system do compositing is the best solution. But for toggle box, we don't care what has been drawn below the highlight, we can always calculate the right solid rectangle highlight color. In fact, we can calculate it in advance. The only time we have to change it is when the BGCOLOR or FGCOLOR change. Please change the code to:

1. Calculate and save off the highlight color, whenever the BGCOLOR or FGCOLOR change. The calculation is a simple use of the XOR bitwise operator on the RGB values.

2. Simply use the pre-calculated highlight color as the drawing color for the solid highlight rectangle. Remove the XOR compositing operations.

This approach is slightly faster at runtime. More importantly, it is less fragile since it doesn't depend on sensitive code at the driver level.

#24 - 11/02/2015 07:52 AM - Vadim Gindin

Greg Shah wrote:

> You have visually tested the results in P2J, comparing side by side with the 4GL?

Not only visually, I've also picked rgb color values and compared them with the values in the table. They all match.

> My primary concern with the use of the SET_XOR_COMPOSITE and RESET_COMPOSITE is that these are deeply coded into sensitive code in the drivers. In your case, it seems like we really don't need to do that. In the browse highlight usage, we don't know what we are overdrawing so making the windowing system do compositing is the best solution. But for toggle box, we don't care what has been drawn below the highlight, we can always calculate the right solid rectangle highlight color. In fact, we can calculate it in advance. The only time we have to change it is when the BGCOLOR or FGCOLOR change. Please change the code to:
>
> 1. Calculate and save off the highlight color, whenever the BGCOLOR or FGCOLOR change. The calculation is a simple use of the XOR bitwise operator on the RGB values.
>
> 2. Simply use the pre-calculated highlight color as the drawing color for the solid highlight rectangle. Remove the XOR compositing operations.
>
> This approach is slightly faster at runtime. More importantly, it is less fragile since it doesn't depend on sensitive code at the driver level.

I'd seen that the testing window is opened with some delay. I hope it will speed it up.

#25 - 11/02/2015 02:17 PM - Vadim Gindin

Greg, I have some difficulty implementing initialization of highlightColor from afterConfigUpdate(..). I need to recall gc.colors() before highlight color calculation and there is NPE happen. I've thought may be it will be simpler not to store separate field and calculate highlight color each time in drawHighlight(..) just like that:

```
      int red = gc.fgColor.getRed() ^ gc.bgColor.getRed();
      int green = gc.fgColor.getGreen() ^ gc.bgColor.getGreen();
      int blue = gc.fgColor.getBlue() ^ gc.bgColor.getBlue();

      highlightColor = new ColorRgb(red, green, blue);
```

#26 - 11/02/2015 02:27 PM - Greg Shah

OK, I'm fine with that approach.

#27 - 11/02/2015 03:25 PM - Vadim Gindin

committed with revno 11002.

#28 - 11/02/2015 03:36 PM - Greg Shah

Code Review Task Branch 2677a Revision 11002

I'm good with the changes.

Have you tested the result and confirmed that it works like the 4GL?

#29 - 11/02/2015 03:38 PM - Vadim Gindin

Greg Shah wrote:

> Code Review Task Branch 2677a Revision 11002
>
> I'm good with the changes.
>
> Have you tested the result and confirmed that it works like the 4GL?

Yes, I've tested it.

#30 - 11/02/2015 03:56 PM - Greg Shah

- % Done changed from 0 to 100

- Status changed from New to Closed

- Target version set to Milestone 12

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

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

Files

highlight1.png	17.2 KB	10/22/2015	Vadim Gindin
highlight2.png	15.6 KB	10/22/2015	Vadim Gindin
highlight_background1.png	15 KB	10/22/2015	Vadim Gindin
highlight_both1.png	10.1 KB	10/22/2015	Vadim Gindin
highlight_both2.png	7.1 KB	10/22/2015	Vadim Gindin
rgb_hihglight.png	41.1 KB	10/26/2015	Vadim Gindin
p2j_highlight.png	5.22 KB	10/26/2015	Vadim Gindin
combos.java	1.85 KB	10/27/2015	Greg Shah
highlight_colors_test.png	14.9 KB	10/30/2015	Vadim Gindin