

User Interface - Feature #1793

improve color support

10/30/2012 09:49 AM - Greg Shah

Status:	Closed	Start date:	
Priority:	Normal	Due date:	
Assignee:	Marius Gligor	% Done:	100%
Category:		Estimated time:	120.00 hours
Target version:	GUI Support for a Complex ADM2 App	version:	
billable:	No		
vendor_id:	GCD		
Description			
Related issues:			
Related to User Interface - Feature #1801: add some frame options		Closed	02/22/2013 08/16/2013
Related to User Interface - Feature #2378: implements dos-hex-attribute of CO...		New	08/19/2014
Related to User Interface - Feature #2379: implement color support in the Win...		New	
Related to User Interface - Feature #2381: implement PUT-KEY-VALUE COLOR (n ...		New	
Related to User Interface - Feature #2384: implement custom color overrides ...		New	
Related to Base Language - Feature #2385: implement environment access on GUI...		Closed	09/04/2014

History

#1 - 10/30/2012 09:51 AM - Greg Shah

1. COLOR-TABLE
2. support for reading configuration to setup the base colors
3. COLOR stmt forms that don't already have support
4. proper color support at the driver level (ChUI is partial but doesn't support real colors, GUI must support full colors from the beginning)

#2 - 10/31/2012 02:53 PM - Greg Shah

- Target version set to Milestone 12

#3 - 04/29/2014 01:02 PM - Greg Shah

Based on the work in [#1801](#) and [#1811](#), the basic color implementation for ChUI (all 3 drivers) is fully functional. The only thing for ChUI support that is not yet implemented is the ability to implement custom color overrides (encoded escape sequences for the NCURSES terminal and RGB values for Swing/AJAX drivers). This is the equivalent of item 2 in note 1 above.

GUI support is not implemented at all.

Please finish the ChUI support and add GUI support.

#4 - 07/16/2014 03:33 PM - Greg Shah

- File system_colors_article_on_openedge_hive_20140716.pdf added

<http://www.oehive.org/node/413>

#5 - 07/22/2014 11:32 AM - Greg Shah

- Status changed from New to WIP

- Assignee set to Marius Gligor

#6 - 07/28/2014 12:18 PM - Marius Gligor

- File orchard-windows.png added

Today I tried again to use OpenEdge for Windows on customer server.

Unfortunately I cannot use both ChUI and GUI OpenEdge for Windows due to the following error message (see attached picture): "The licence for the client executable has expired. (4399)"

#7 - 07/28/2014 12:46 PM - Marius Gligor

I fixed the issue by using OE 10.2b instead OE 11.3, thanks Constantin.

#8 - 08/14/2014 12:37 PM - Marius Gligor

- File everything.zip added

- File mag_upd20140814a.zip added

Here is a first version of COLOR-TABLE implementation.

The implementation of COLOR-TABLE looks the same as FONT-TABLE implementation in many aspects. Both are system handles but manage different resources.

1. Default colors in COLOR-TABLE are described in OpenEdge manual as a table having 16 RGB colors entries.

I checked on windev01 OpenEdge GUI environment and is true. Initial COLOR-TABLE is 16 entries in size having the RGB values described in manual.

In my implementation the same default colors are loaded from a XML file (default-colors.xml) It is possible to add and delete entries from COLOR-TABLE.

Also it is possible to change the RGB values of a specified color entry only if the color entry is for a dynamic color.

The attribute static/dynamic of a color entry can be changed also using specific COLOR-TABLE methods. All operations for COLOR_TABLE are implemented.

2. System colors on the other hand are not part of the COLOR-TABLE. The values are stored on Windows registry and are used by Windows GUI widgets.

I think that our GUI implementation should be a cross platform solution so I stored the system colors inside a XML file (system-colors.xml)

The values of system colors are taken from my Windows 7 registry. Unlike default colors which are always the same the system colors could have different values.

The article on note 4 describe a way to access system colors within a P4G procedure using procedures contained in archive everything.zip.

I have to study the content of this archive in order to understand how it works.

What I understood so far is that windows.p and windows.i from the archive everything.zip provide access to a set of Windows API functions via a dll library. It seems to work only on Win32 not on win64.

OpenEdge GUI is Windows OS aware. I think that our GUI implementation should be a cross platform solution so I stored the system colors inside a XML file (system-colors.xml)

Currently the system colors are partial implemented, only the loading part and some simple methods to get and set entries in system color table.

The system colors could be used primarily on our GUI interface when drawing GUI widgets.

TBD what other interfaces should be implemented related to system colors.

3. I added also rules for COLOR-TABLE and RGB-VALUE conversion. Using the following test case which cover all COLOR-TABLE methods and attributes in both normal and wrapped mode the conversion is done without compile errors.

```
DEFINE VARIABLE red AS INTEGER NO-UNDO.  
DEFINE VARIABLE blue AS INTEGER NO-UNDO INITIAL 127.  
DEFINE VARIABLE green AS INTEGER NO-UNDO INITIAL 127.  
DEFINE VARIABLE ix AS INTEGER NO-UNDO.  
DEFINE VARIABLE ht AS HANDLE.
```

```
/* Normal */
```

```
ix = COLOR-TABLE:NUM-ENTRIES.  
COLOR-TABLE:NUM-ENTRIES = ix + 1.
```

```
COLOR-TABLE:SET-DYNAMIC(ix, TRUE).
```

```
COLOR-TABLE:SET-RED-VALUE(ix, red).  
COLOR-TABLE:SET-GREEN-VALUE(ix, green).  
COLOR-TABLE:SET-BLUE-VALUE(ix, blue).
```

```

DISPLAY COLOR-TABLE:GET-DYNAMIC (ix) .
DISPLAY COLOR-TABLE:GET-RED-VALUE (ix) .
DISPLAY COLOR-TABLE:GET-GREEN-VALUE (ix) .
DISPLAY COLOR-TABLE:GET-BLUE-VALUE (ix) .

COLOR-TABLE:SET-RGB-VALUE (ix, RGB-VALUE (192, 48, 127)) .
DISPLAY COLOR-TABLE:GET-RGB-VALUE (ix) .

DISPLAY COLOR-TABLE:TYPE .
DISPLAY COLOR-TABLE:INSTANTIATING-PROCEDURE .

/* Wrapped */
ht = COLOR-TABLE:HANDLE .

ix = ht:NUM-ENTRIES .
ht:NUM-ENTRIES = ix + 1 .

ht:SET-DYNAMIC (ix, TRUE) .

ht:SET-RED-VALUE (ix, red) .
ht:SET-GREEN-VALUE (ix, green) .
ht:SET-BLUE-VALUE (ix, blue) .

DISPLAY ht:GET-DYNAMIC (ix) .
DISPLAY ht:GET-RED-VALUE (ix) .
DISPLAY ht:GET-GREEN-VALUE (ix) .
DISPLAY ht:GET-BLUE-VALUE (ix) .

ht:SET-RGB-VALUE (ix, RGB-VALUE (192, 48, 127)) .
DISPLAY ht:GET-RGB-VALUE (ix) .

DISPLAY ht:TYPE .
DISPLAY ht:INSTANTIATING-PROCEDURE .

```

However I found some conversion bugs which are not caused by my changes. A simple test case like:

```

DEFINE VARIABLE ht AS HANDLE .
DISPLAY ht .

```

is converted with compile errors. Another simple test like:

```

DISPLAY COLOR-TABLE:HANDLE .

```

is not converted at all!

#9 - 08/18/2014 08:34 PM - Greg Shah

Code Review 0814a

1. Line 463 of language_statements.rules has a reference to KW_FONT.
2. The progress.g change is not correct. As far as I know, RGB-VALUE is a 4GL built-in function. It is not a handle-based method. The function return value is already in the file on line 5714. Don't be fooled that the KW_RGB_VAL is not on that line. The function keyword resolution works different than the method/attribute lookup.
3. Since a ColorTableEntry[] is transferred to the client, it is important to implement Externalizable instead of Serializable. We found that that approach significantly improves performance.
4. Do we need to push each individual call (e.g. SET-RED-VALUE() and SET-BLUE-VALUE()...) down to the client? I would prefer storing the state on the server and then using state sync to push the changes to the client the next time control transfers to the client side. Otherwise it will be a very chatty/slow approach.
5. ClientExports is missing a history entry.

#10 - 08/18/2014 08:37 PM - Greg Shah

The valuse are stored on Windows registry and are used by Windows GUI widgets.

I think that our GUI implementation should be a cross platform solution so I stored the system colors inside a XML file (system-colors.xml)

Exactly right. I don't want to implement any direct integration with the EnvironmentDaemon or calls to native APIs. We will read the values from stored configuration.

My only question is how to handle the "save" functionality.

However I found some conversion bugs which are not caused by my changes. A simple test case like:

...

is converted with compile errors. Another simple test like:

...

is not converted at all!

Yes, we don't support the handle type in frames.

- File color2.png added

- File color1.png added

Here are my conclusion doing some "researches" on color implementation for ChUI and GUI interfaces on Linux OS/ Windows OS.

1. COLOR statement

Indicates the video attribute or color for normal display or for data entry.

```
COLOR [ DISPLAY ] color-phrase [ PROMPT color-phrase ] { field ... } { [ frame-phrase ] }  
COLOR PROMPT color-phrase { field ... } { [ frame-phrase ] }
```

2. COLOR phrase

Specifies a video attribute or color. In Progress Version 7 and later, the COLOR phrase is superseded by the FGCOLOR and BGCOLOR options in graphical user interfaces

and by the PFCOLOR and DCOLOR options in character interfaces. The COLOR phrase is supported only for backward compatibility.

color-phrase Specifies a video attribute or color. Following is the syntax for color-phrase:

```
{ NORMAL  
  | INPUT  
  | MESSAGES  
  | protermcap-attribute  
  | dos-hex-attribute  
  | { [ BLINK- ] [ BRIGHT- ] [ fgnd-color ] [ bgnd-color ] }  
  | { [ BLINK- ] [ RVV- ] [ UNDERLINE- ] [ BRIGHT- ] [ fgnd-color ] }  
  | VALUE ( expression )  
}
```

a.) protermcap-attribute You use the protermcap-attribute option only if you are using UNIX. This is the name assigned to the attribute in the protermcap file (for example, RED, BLINK, etc.).

b.) dos-hex-attribute: A hex string with a value of 00 through FF.

This option should works only on Windows OS. The dos-hex-attribute is the same as COLOR command used on a DOS prompt. For example open a command window and type:

COLOR 24 then press ENTER. The result is a Green/Red background/foreground colors. Then type COLOR command without any parameter and colors will be reset to default values.

Color attributes are specified by TWO hex digits -- the first corresponds to the background; the second the foreground. Each digit can be any of the below values.

0	Black
1	Blue
2	Green
3	Aqua
4	Red
5	Purple
6	Yellow
7	White
8	Gray
9	Light Blue
A	Light Green
B	Light Aqua
C	Light Red
D	Light Purple
E	Light Yellow
F	Bright White

Doing tests on customer's OE for Windows character interface I found that it's not works. I've tried:

```
def var n as int.  
color display "24" n.  
display n.
```

and

```
def var n as int.  
color display "0x24" n.  
display n.
```

unfortunately without success!

c.)

```
{ [ BLINK- ] [ BRIGHT- ] [ fgnd-color ] [ bgnd-color ] }  
  { [ BLINK- ] [ RVV- ] [ UNDERLINE- ] [ BRIGHT- ] [ fgnd-color ] }
```

These options are also available on Windows OS only. Examples of use:

```
BLINK-BRIGHT-RED/GREEN  
RVV-BLUE
```

fgnd-color / bgnd-color are Windows background / foreground colors. According to OE manual we could use the following values for colors. As we can see 17 colors are defined in the following table while the default color table has only 16 entries! According to my tests colors 7 and 8 are basically the same so the following table is basically the same as Windows default colors.

Windows COLORS fgnd-color / bgnd-color

Index	Color	Abbreviation
0	Black	Bla, Blk
1	Blue	Blu
2	Green	Gre, Grn
3	Cyan	C
4	Red	Red
5	Magenta	Ma
6	Brown	Bro, Brn
7	Gray	Gra, Gry
8	Dark-Gray	D-Gra
9	Light-Blue	Lt-Blu
10	Light-Green	Lt-Gre
11	Light-Cyan	Lt-C
12	Light-Red	Lt-Red
13	Light-Magenta	Lt-Ma
14	Light-Brown	Lt-Bro
15	Yellow	Y
16	White	W

Simple tests like:

```
def var n as int.  
color display "blink-red/green" n.  
display n.
```

and

```
def var n as int.  
color display "red" n.  
display n.
```

works on both ChUI and GUI Windows OS interfaces.

3. DCOLOR, PFCOLOR, FGCOLOR, BGCOLOR attributes.

DCOLOR Attribute (Character Interfaces only)
The color number for the display color of the widget in character mode. This attribute is ignored in graphical interfaces.

Data Type: INTEGER
Access: Readable/Writeable
The color number represents an entry in the color table maintained by the COLOR-TABLE handle.

PFCOLOR Attribute (Character interfaces only)
The color number of the color of a widget that has input focus.
Data Type: INTEGER
Access: Readable/Writeable
The color number represents an entry in the color table maintained by the COLOR-TABLE handle.

FGCOLOR Attribute (Graphical interfaces only)
Data Type: INTEGER
Access: Readable/Writeable
The color number for the foreground color of the widget.
The color number represents an entry in the color table maintained by the COLOR-TABLE handle.

BGCOLOR Attribute (Graphical interfaces only)
Data Type: INTEGER
Access: Readable/Writeable
Specifies the color number for the background color of the widget.
The color number represents an entry in the color table maintained by the COLOR-TABLE handle.

All are integer values which represents an entry in the color table maintained by the COLOR-TABLE handle according to OE manual. However here could be a confusion because the COLOR-TABLE System Handle is available only on GUI interfaces. Only FGCOLOR and BGCOLOR attributes are really indexes on COLOR-TABLE.

COLOR-TABLE System Handle A handle to the current color table.

Interfaces	OS	SpeedScript
Graphical only	Windows only	No

On ChUI interface COLOR-TABLE:NUM-ENTRIES return 0 always and we cannot add entries to COLOR-TABLE. Also DCOLOR and PFCOLOR are indexes on a color table which holds pairs of colors and a list of attributes while an entry in COLOR_TABLE for GUI holds a single color as a RGB value. Here is an interesting example:

```
def var n as int.  
color display "red/green" n.  
display n.  
display "Test" dcolor 5.
```

Running this test both n and "Test" string are displayed on red having a green background.
This means that a new entry in color table is created if "red/green" does not exists. Initially color table contains 5 entries:

```
0 - NORMAL  
1 - INPUT  
2 - MESSAGES  
3 - BOLD  
4 - BLINK
```

color display "red/green" n. will create a new entry at index 5:

```
0 - NORMAL  
1 - INPUT  
2 - MESSAGES  
3 - BOLD  
4 - BLINK  
5 - RED/GREEN
```

If background is missing color display "red" n. the entry will be RED and for foreground NORMAL color is used.
color display "red/green" prompt "white/black" n. will create 2 new entries at index 5 and 6:

```
0 - NORMAL  
1 - INPUT  
2 - MESSAGES
```

- 3 - BOLD
- 4 - BLINK
- 5 - RED/GREEN
- 6 - WHITE/BLACK

4. Looking on Windows registry on the customer environment I found two interesting information (see attached picture). Picture 1 depicts default colors used by OE for GUI Windows.
The second picture depict the color definition of the first 5 color pairs for ChUI interface on Windows.

CONCLUSIONS:

5. The current implementation for ChUI is based mainly on protermcap file definitions which are available only on Linux / Unix OS. For Windows OS the protermcap is substituted by options like:

```
{ [ BLINK- ] [ BRIGHT- ] [ fgnd-color ] [ bgnd-color ] }  
{ [ BLINK- ] [ RVV- ] [ UNDERLINE- ] [ BRIGHT- ] [ fgnd-color ] }
```

Based on these attributes we have to construct and add entries to in-memory color table for ChUI interface which is not the same as COLOR-TABLE used only on GUI. Actually the implementation reside in class ColorSpec Also ColorPalette contains only 8 color and we have to provide a new palette having 16 entries (see paragraph c)
At some point on ChUI color implementation we have to check the OS platform and provide specific implementation.

7. COLOR stmt and COLOR phrase are available on both ChUI and GUI interfaces. The implementation should works on both interfaces.

6. For ChUI interfaces OE does not allow to define custom colors. Only on GUI interface is possible to define new RGB colors and add to COLOR-TABLE. The new colors could be used with FGCOLOR and BGCOLOR options.

8. I could not do tests using dos-hex-attribute on Windows ChUI (see 2-b) However we could implements also this option.

#12 - 08/19/2014 12:53 PM - Greg Shah

These are very good findings.

8. I could not do tests using dos-hex-attribute on Windows ChUI (see 2-b) However we could implements also this option.

I also tried some Windows CHUI tests with these:

- using escaped octal characters ~044 and "~044"
- using non-string inputs 24, x24 and 0x24
- added a FORM statement before the COLOR statement and explicitly using WITH frame to make sure all statements were referencing the same frame

None of these work. I am guessing this was something that only worked on real DOS (and Windows 3.1 but never on Windows 32-bit). Progress did have an MS-DOS client at one time.

Don't worry about implementing this right now, since we can't find a way to make it work on Windows. If we encounter a customer using MS-DOS or can find a way to make it work on Windows, then we will fix it later.

Please document this limitation in the Javadoc for the associated classes, so we don't forget this fact. Also: create a new Redmine task to implement that feature. Have it reference the notes in this task for details.

#13 - 08/20/2014 05:44 PM - Greg Shah

Question from MAG:

ColorPalette should hold multiple entries depending on platform Linux or Windows. ColorPalette is implemented as a Java enumeration which is not a clever idea as we need to have different entries depending on platform. At this moment ColorPalette contains only 8 colors which correspond to the values defined in curses.h. For Windows OS we need 16 colors. Fortunately the existing colors are also needed for Windows.

We have 2 solutions to fix this issue:

1. Add new entries to enum and use the order as color index. The first 8 colors should remain on the same position to preserve the same ordinal number.
2. Redesign the ColorPalette as a Java class instead of an enumeration and do the appropriate changes to related classes.

What's in your opinion the best solution 1 or 2?

#14 - 08/20/2014 05:47 PM - Greg Shah

Enums have the advantage of being very concise in syntax and being easy to read/understand/use. The downside of using an enum is the implicit nature of the indexing and limitations on how values can be structured.

If you were to redesign ColorPalette as a Java class, what approach do you suggest? I don't want to burden the class with lots of getters/setters.

#15 - 08/21/2014 10:58 AM - Marius Gligor

- File mag_upd20140821a.zip added

The attached archive contains my changes which I made to implement COLOR phrase for ChUI interfaces.

The protermcap option for Linux/Unix OS was already implemented so I added only changes for Windows OS options.

1. I implemented COLOR phrase by keeping the ColorPalette as an enumeration. I added new entries in palette and I provided a mechanism for conversion between color indexes and palette indexes. Basically the COLOR phrase is fully implemented except dos-hex-attribute option. Searching on Internet I found that Progress 4 GL is available for multiple platforms including some old ones like MS-DOS, Novell etc. My changes are the implementation for Windows OS. COLOR phrase should be supported on both ChUI and GUI interfaces. The current implementation works only on ChUI.

2. I did tests using the Swing and AJAX clients on Windows OS and works fine. The Windows native client does not implements colors at this moment.

#16 - 08/21/2014 11:17 AM - Greg Shah

I've created [#2379](#) to capture the work item for adding color support to the native Windows console (ChUI) client. We won't work on that as part of this task.

#17 - 08/21/2014 11:42 AM - Greg Shah

Code Review 0821a

I like the result. Some minor feedback:

1. All data members should be at the top of the class, above any constructors or static methods or static initializers. That means the two private members would be located just after the private static data members.

2. The ColorPalette constructor is missing a javadoc entry for its new parameter.

3. As much as possible, we enable the converted application to be executed on any client platform regardless of the original client platform. Part of doing this means that we need to make the P2J runtime client platform act like the original platform acted, so that the application behaves and appears like it always did (from the user's perspective). This means that we should be able to run an app that used to run in Windows console (ChUI) in P2J on Linux and vice versa. The code in ColorSpec uses PlatformHelper to detect the current runtime client. The problem here is that I think we need to detect what the original client was. That is probably a setting we need in the directory. Based on that we would "show" the original system's color map, no matter what the runtime client type is.

#18 - 08/22/2014 07:17 AM - Marius Gligor

- File *mag_upd20140822a.zip* added

Here are my changes for COLOR phrase implementation as a result of your code review.

1. Regarding platform detection, the code inside ColorSpec class is running only on the server side, never on the client side.

As a result the platform detection took place on server side not on the client side.

On other words if server is Running on Windows the Windows color scheme is use otherwise the Linux/Unix color scheme is used.

If we need a more flexible configuration we have to design one based on server registry.

The ColorSpec initialization load color scheme for all supported platforms and the color scheme is selected on ColorSpec#convert() depending on platform or other configurations

which should be designed and implemented. The current implementation is based on server platform detection only.

2. ColorSpec class is never serialized so I eliminated the implementation of Serialization interface and the default constructor.

#19 - 08/22/2014 08:53 AM - Greg Shah

Code Review 0822a

The changes are fine.

the code inside ColorSpec class is running only on the server side, never on the client side.

Good point.

If we need a more flexible configuration we have to design one based on server registry.

Exactly.

Add a static method to util/EnvironmentOps.java called getLegacyPlatform(). The idea of this is different than the purpose of EnvironmentOps.getOperatingSystem(). The getOperatingSystem() is used to back the OPSYS built-in in Progress, which is often used to select different paths at runtime. The most common use is to execute different shell commands based on the actual runtime system type. So this is not something we want to map to the original system's platform type.

But this new getLegacyPlatform() will provide a mechanism to know what behavior should be simulated, when we have the ability to make all client systems act like the legacy platform. The ColorSpec is a perfect example.

This code should lookup the platform as a string in the directory.

```
String platform = Utils.getDirectoryNodeString(null, "legacyPlatform", runtimeClientPlatform);
```

The default value if one is not specified (runtimeClientPlatform in the example above) should be read from the getOperatingSystem(). The calculated platform should be saved in the context-local work area so that subsequent calls the getLegacyPlatform() can use the cached value.

#20 - 08/22/2014 11:03 AM - Marius Gligor

- File color5.png added

- File progress.ini added
- File color4.png added
- File color3.png added

Digging in the customer's OE 10.2b environment I found the configuration file progress.ini which contains among other settings the default colors and font tables. (see progress.ini)

Then I did some tests using PUT-KEY-VALUES COLOR statement. PUT-KEY-VALUES and GET-KEY-VALUES are statement available in Windows OS only and are used to write and read values from system registry or ini files. Ini files are used on earlier Windows version but could be used even on the latest Windows versions.

I used a specific statement PUT-KEY-VALUES COLOR n|ALL which is used to save COLOR-TABLE in OS user environment (registry or ini files).

1. PUT-KEY-VALUES COLOR n save the in memory COLOR-TABLE into registry. All entries are saved regarding the value of n. (see color4.png and color5.png)
2. 1. PUT-KEY-VALUES COLOR ALL save the in memory COLOR-TABLE into registry and three other values for standard colors NORMAL INPUT and MESSAGES.
Standard colors NORMAL INPUT and MESSAGES are stores as color pairs foreground and background. (see color3.png)
3. When OE starts if a COLOR-TABLE is already saved for current user it load the colors from saved table. If not the default 16 colors are loaded into COLOR-TABLE from preogress.ini file
in OE 10.2b or form registry in OE 11.3
4. COLOR-TABLE is saved per user on user environment (HKEY_CURRENT_USER).

In conclusion I think that we have to follow the same behaviour and implements a mechanism to save COLOR-TABLE somewhere on the server side in different files for each P2J user.

#21 - 08/22/2014 12:43 PM - Marius Gligor

- File mag_upd20140822b.zip added

The attached archive 0822b contains implementation of EnvironmentOps.getLegacyPlatform()

#22 - 08/25/2014 10:47 AM - Marius Gligor

- File mag_upd20140825a.zip added

Here are my changes after your code review for 0814a on note 9.

Also I did new tests this time using LOAD USE and UNLOAD statements available only on Windows OS according to OE manual.

1. LOAD statement:

Creates application defaults, involving colors, fonts, environment variables, etc., or loads existing defaults, to a graphical or character application.

2. USE statement:

Specifies environment defaults that apply to subsequent windows that the application creates. The defaults might reside in the registry or in an initialization file.

The defaults can involve colors, fonts, environment variables, etc. You must specify a default in a LOAD statement before you specify it in a USE statement.

The resources of the new current environment are available only for windows created after the new environment is made current.

3. UNLOAD statement:

Unloads a set of environment specifications from the current environment, which might be the registry or an initialization file. Among others two important notes regarding UNLOAD statement:

- An application cannot UNLOAD a set of environment specifications until it terminates all windows that use those specifications.
- If you UNLOAD the current environment, the default environment becomes the current environment. To define a new current environment, use the USE statement.

4. I've created 2 INI files each one containing a COLOR-TABLE having 13 respective 10 entries. Then I used the following test case:

```
DEF VAR ix AS INT.  
LOAD "env1".  
USE "env1".  
ix = COLOR-TABLE:NUM-ENTRIES.  
DISP ix.  
COLOR-TABLE:NUM-ENTRIES= ix + 1.  
PUT-KEY-VALUE COLOR ALL.  
LOAD "env2".  
USE "env2".  
DISP COLOR-TABLE:NUM-ENTRIES.  
USE "env1".  
DISP COLOR-TABLE:NUM-ENTRIES.  
UNLOAD "env1".  
UNLOAD "env2".
```

After executing the test looking inside env1.ini file I found 14 entries in COLOR-TABLE which proves that each environment has its own COLOR-TABLE in other words the COLOR-TABLE is managed per environment. Initially a default environment exists (loaded) having a default COLOR-TABLE loaded from progress.ini or from registry if it was previously saved. PUT-KEY-VALUE COLOR (n | ALL) save the in-memory COLOR-TABLE in current environment. The current environment is initially the default environment but could be switched using the USE environment statement which must be called only after a LOAD environment statement.

Note that LOAD NEW ... statement could create a new INI file if not already exists.

5. In P2J the COLOR-TABLE management is done on each environment on the client side. That's why all server API for COLOR-TABLE are delegated to client.

Code inside ColorTable run on server side but the API calls are delegated to client side on ColorManager class.

Here we have to discuss from where client environments are loaded and to decide where the color tables are managed on server side or on client side.

6. Another issue which I found in P2J relating to environment loading. An instance of EnvironmentDaemon is used to load environments.

EnvironmentDaemon is designed to work on Windows OS only using the Registry class which is a JNI interface for Windows registry. It works also with INI files which could be a cross platform solution.

Even more the initialization of EnvironmentDaemon inside ThinClient.initializePost(...) line 2587 looks like:

```
if (tc.ospd.getOperatingSystem().indexOf("win") != -1)  
{  
    tc.ed = new EnvironmentDaemon(session, single);  
}
```

which means that tc.ed is created only when client runs on Windows OS.

#23 - 08/25/2014 02:18 PM - Greg Shah

In conclusion I think that we have to follow the same behaviour and implements a mechanism to save COLOR-TABLE somewhere on the server side

in different files for each P2J user.

I agree with the general approach, however I think we would not use separate files. We would just store the changes into the server's directory, in some area associated with the user's account.

However, please note that we probably do not need this feature right now. My review of the reports for the current GUI code we are working with is inconclusive (because there are some cases where PUT-KEY-VALUE is being called with runtime values), BUT from looking at the code it does not seem likely that these are COLOR usages. I will check with the customer to confirm. In the meantime, please create a new task for this feature. Make sure to:

- Reference the relevant notes from this task.
- Add this task as a "Related Task" to the new one.

#24 - 08/25/2014 02:47 PM - Marius Gligor

If we decide to load and store custom environments also on server side like default environment we could move the code for COLOR-TABLE management (API) on server side in ColorTable class.

The system colors should be kept on client side in ColorManager class because the system colors are used solely when drawing GUI widgets.

#25 - 08/25/2014 03:05 PM - Greg Shah

Code Review 0822b

1. Minor javadoc changes to getLegacyPlatform():

specify a legacy platform regarding the real platform

should be changed to:

specify the legacy platform on which the original application code ran, which may be different than the converted application's real platform

and this:

```
* The value is read from server directory. If no value is set for legacyPlatform
* The value of legacyPlatform MUST be UNIX or WIN32.
* the returned value is the same as returned by getOperatingSystem call.
```

probably should be this:

```
* The value is read from server directory. If no value is set for legacyPlatform
```

```
* then the returned value is the same as returned by the getOperatingSystem call.  
* If set in the directory, the value of legacyPlatform MUST be UNIX or WIN32.
```

2. The `getOperatingSystem().value` should never be used. Instead, please use `character.toStringMessage()`.

#26 - 08/25/2014 03:16 PM - Greg Shah

Code Review 0825a

I'm fine with the changes.

#27 - 08/25/2014 03:29 PM - Greg Shah

5. In P2J the COLOR-TABLE management is done on each environment on the client side. That's why all server API for COLOR-TABLE are delegated to client.

Code inside `ColorTable` run on server side but the API calls are delegated to client side on `ColorManager` class.

Here we have to discuss from where client environments are loaded and to decide where the color tables are managed on server side or on client side.

Please look at [#1794](#). Constantin had to deal with a very similar set of problems.

We want to store well-known configuration on the server side. In this case, we want the data in the directory, like the per-user fonts can be stored.

I also wonder if his font management code for the directory can be re-purposed for use with colors too? Optimally, we would have a generalized solution (common code for both fonts and color configuration).

6. Another issue which I found in P2J relating to environment loading. An instance of `EnvironmentDaemon` is used to load environments. > `EnvironmentDaemon` is designed to work on Windows OS only using the `Registry` class which is a JNI interface for Windows registry. It works also with INI files which could be a cross platform solution.

Even more the initialization of `EnvironmentDaemon` inside `ThinClient.initializePost(...)` line 2587 looks like:

Yes, this is a bug. See note 136 of [#2252](#).

As you note, there is no reason we can't use stanza-based INI files on non-Windows platforms. In fact, our whole approach is to try to make the 4GL applications as portable as possible in such cases. Please check with Constantin to see if he has already done anything on this, if not, then you can fix it.

If we decide to load and store custom environments also on server side like default environment we could move the code for COLOR-TABLE management (API) on server side in `ColorTable` class.

As far as I understand things, the client can't change the COLOR-TABLE, it just uses whatever has been loaded/modified. If so, then we can keep things clean (no need to make many calls to the client) by managing the COLOR-TABLE on the server and then just synching the state to the client the next time control flow shifts.

The system colors should be kept on client side in `ColorManager` class because the system colors are used solely when drawing GUI widgets.

This makes sense.

#28 - 08/27/2014 06:34 AM - Marius Gligor

I found a conversion problem and cannot fix it. Both COLOR-TABLE and FONT-TABLE have a NUM-ENTRIES attribute. When are used as wrapped resource the conversion is done with error. I have the following P4 code:

```
ht = COLOR-TABLE:HANDLE.  
  
ix = ht:NUM-ENTRIES.  
ht:NUM-ENTRIES = ix + 1.  
  
ht:SET-DYNAMIC(ix, TRUE).
```

which is converted in Java as:

```
ht.assign(ColorTable.asHandle());  
ix.assign(ht.unwrapFontTable().getNumEntries());  
ht.unwrapFontTable().setNumEntries(plus(ix, 1));  
ht.unwrapColorTable().setDynamic(ix, true);
```

For getNumEntries and setNumEntries ht.unwrapFontTable() is generated instead ht.unwrapColorTable(). The problem come from conversion rules. In method_attributes.rule we have:

```
<!-- FONT-TABLE -->  
<rule>methodText.length() == 0 and  
      (htype == prog.kw_font_tab or ref.type == prog.var_handle)
```

```
<!-- COLOR-TABLE -->  
<rule>methodText.length() == 0 and  
      (htype == prog.kw_colr_tab or ref.type == prog.var_handle)
```

On wrapped mode methodText.length() 0 and ref.type prog.var_handle and because the rules for FONT-TABLE are defined before the rules for COLOR-TABLE and NUM-ENTRIES are defined on both rules ht.unwrapFontTable() is generated regarding the handle is for COLOR-TABLE. In other words the rules:

```
<rule>methodText.length() == 0 and  
      (htype == prog.kw_font_tab or ref.type == prog.var_handle)
```

```
<rule>methodText.length() == 0 and  
      (htype == prog.kw_colr_tab or ref.type == prog.var_handle)
```

are not enough on this case. We MUST also have a condition which tell if handle is for FONT-TABLE or COLOR-TABLE How we could solve this issue?

#29 - 08/27/2014 07:06 AM - Constantin Asofiei

Marius Gligor wrote:

I found a conversion problem and cannot fix it. Both COLOR-TABLE and FONT-TBALE have a NUM-ENTRIES attribute.

The problem here is that NUM-ENTRIES is common attribute for both the COLOR-TABLE and FONT-TABLE resources. When handle vars are used, there is no way of determining at conversion time the type of resource referenced by that handle. To solve this, we extract the converted API into its own interface, and add unwrap APIs for that interface. See the unwrapInstantiatingProcedure and the INSTANTIATING-PROCEDURE attribute for an example.

#30 - 08/27/2014 09:56 AM - Marius Gligor

Constantin Asofiei wrote:

Marius Gligor wrote:

I found a conversion problem and cannot fix it. Both COLOR-TABLE and FONT-TBALE have a NUM-ENTRIES attribute.

The problem here is that NUM-ENTRIES is common attribute for both the COLOR-TABLE and FONT-TABLE resources. When handle vars are used, there is no way of determining at conversion time the type of resource referenced by that handle. To solve this, we extract the converted API into its own interface, and add unwrap APIs for that interface. See the unwrapInstantiatingProcedure and the INSTANTIATING-PROCEDURE attribute for an example.

I've fixed the conversion issue. I created a new interface NumEntries for NUM-ENTRIES getter and setters.

FontTableResource and ColorTableResource interfaces extends NumEntries.

I added handle.unwrapNumEntries and I did the appropriate rules changes for COLOR-TABLE and FONT-TABLE.

The conversion is now OK and the converted code is compiled without errors. Also runtime tests for conversion procedure works fine.

Thanks.

#31 - 08/28/2014 04:30 AM - Constantin Asofiei

Greg Shah wrote:

As you note, there is no reason we can't use stanza-based INI files on non-Windows platforms. In fact, our whole approach is to try to make the 4GL applications as portable as possible in such cases. Please check with Constantin to see if he has already done anything on this, if not, then you can fix it.

I haven't worked on this. Also, note that this problem I think was encountered by Ovidiu, too: some configuration was being read from the .ini file, which I think was set at the appserver agent configuration. So, we need to ensure that .ini files can be specified for normal users, batch processes and appserver agents.

More, in case of custom auth plugins, I think we need to push a default/per-server .ini file (we might be able to limit this to only default-window/font/color-table config, but I think is best to allow access to the entire .ini file).

#32 - 08/28/2014 09:08 AM - Marius Gligor

- File mag_upd20140828a.zip added

- File color-nodes.xml added

Here are my latest changes for COLOR-TABLE implementation.

1. The management of COLOR-TABLE is done now on server side. However FONT-TABLE management is implemented on client side perhaps due to the following differences extracted from OE manual:

Unlike the COLOR-TABLE system handle, the FONT-TABLE system handle does not allow you to set fonts dynamically. Font entries can only be changed by the user through the font system dialog box. Fonts are always dynamic.

Default colors are loaded from server directory. I attached also a file containing the structure of proposed directory nodes. Because system colors are not part of 4GL environment I think that system color table should be stored on server directory only globally (per server not per user).

Also the table should be read only which means that could not be changed from Java code.

2. Unlike ChUI colors GUI colors are dynamics. FGColor and BGColor are indexes into the current color COLOR-TABLE. Because COLOR-TABLE entries could be changed at runtime from P4 code the color values corresponding to FGColor and BGColor MUST be evaluated every time are used to draw widgets.

If the current COLOR-TABLE is stored on server side each time we convert a FGColor or BGColor index into a RGB color we have to do a call on server because the draw is done on client side.

You said something related to a synch between server and client. How we can do that? Could you give an example to understand how it works?

3. Regarding INI files. The initial (default) environment should be loaded from server directory. PUT-KEY-VALUE should save default COLOR-TABLE and default FONT-TABLE on server directory per client. On custom environments PUT-KEY-VALUE should save COLOR-TABLE and FONT-TABLE inside the INI file associated with the custom environment.

INI files should be used only for custom environments using statements like LOAD, USE and UNLOAD environment where environment should be the name of an INI file.

One question arise here. From where INI files are loaded from server side or from client side?

I saw that conversion of LOAD, USE and UNLOAD statements is already implemented and generate sequences like:

```
EnvironmentOps.load("env");
EnvironmentOps.use("env");
EnvironmentOps.unload("env");
```

However we have to add some code to integrate COLOR-TABLE and FONT-TABLE management with LOAD, USE and UNLOAD statements. I have to check and understand how EnvironmentOps are implemented.

4. System colors are not part of 4GL environment. The system colors are used by Windows OS when draw graphical objects and in our case are P2J specific.

We could use system colors solely as default colors when draw GUI widgets by replacing the colors defined in GuiOptions with corresponding system

color.
We could define look and feel themes via system colors.

#33 - 08/28/2014 12:55 PM - Greg Shah

we need to ensure that .ini files can be specified for normal users, batch processes and appserver agents

Yes.

I think is best to allow access to the entire .ini file

Agreed.

#34 - 08/28/2014 03:32 PM - Greg Shah

Code Review 0828a

The code is fine. I like the direction it is moving.

#35 - 08/28/2014 03:48 PM - Greg Shah

Because system colors are not part of 4GL environment I think that system color table should be stored on server directory only globally (per server not per user).

Actually, since legacy users will often have had the app installed on their local windows system, they may have had custom system colors assigned. And there may be disabled people for which assessability is important but whose changes would not be shared with everyone. I think it is best to keep the same structure of per-account/group with the defaults being allowed at the server level.

Also the table should be read only which means that could not be changed from Java code.

Yes.

If the current COLOR-TABLE is stored on server side each time we convert a FGColor or BGColor index into a RGB color we have to do a call on server because the draw is done on client side.

We can avoid that by using our state sync approach.

You said something related to a synch between server and client. How we can do that? Could you give an example to understand how it works?

Please see:

ui/ClientState
ui/ServerState
ui/LogicalTerminal (both getChanges() and applyChanges())
ui/chui/ThinClient (both getChanges() and applyChanges())
net/StateSynchronizer
net/Protocol (see registerSynchronizer(), attachChanges() and applyChanges())

The basic idea is to "piggyback" the transfer of state changes on the next message sent to the other side. By timing the apply process on the target side such that it is before any possible use of that state, it appears to the code like the state is always synchronized with the version on the other side.

From where INI files are loaded from server side or from client side?

For standard application INI files, I'd like to do this from the server. This is different from how Progress works, but if we can solve that problem then it is a nice improvement for our customers.

We should be able to read a default version from the application jar file. It will be tricky in the case where a custom version needs to be read. Also, the current conversion/runtime doesn't support this server-side idea.

Constantin: what do you think?

We could use system colors solely as default colors when draw GUI widgets by replacing the colors defined in GuiOptions with corresponding system color.

Yes, this makes sense. We will have to capture a good set of basic (Windows 7?) defaults. These can be hard coded into our Java source and then the customer should be able to override.

#36 - 08/29/2014 12:18 PM - Marius Gligor

1. The P2J implementation for LOAD, USE and UNLOAD statements is conforming with OpenEdge manual.

It works only if client is running on Windows OS and the API calls are delegated to client side (see EnvironmentDaemon line 60).
I think that we have to do some changes here in order to accommodate our needs for GUI drivers perhaps on another Redmine task.

2. Another issue related to environments. Environments are application specific.
All users which are authorised to run an application which use custom environments must have INI files having the name of environments.
The question here is: An application environment (colors and fonts) should be the same for all users?

#37 - 08/29/2014 12:27 PM - Greg Shah

I think that we have to do some changes here in order to accommodate our needs for GUI drivers perhaps on another Redmine task.

I'm fine with that approach so long as we determine the requirements now.

The question here is: An application environment (colors and fonts) should be the same for all users?

I think we will need per-user level. The reason is the same as for the system colors, current installations are often on Windows desktop systems which 1 system per user. Each user can have different app environments today. Even in a Windows Terminal Server environment, I think the registry is split for users so different users have different content.

#38 - 09/02/2014 11:40 AM - Marius Gligor

- *File mag_upd20140902a.zip added*
- *% Done changed from 0 to 80*
- *Status changed from WIP to Review*

Here are my latest changes on color improvements:

- synchronization state between server and client.
- default system colors hard coded.
- added color support to GUI driver.
- other improvements.

Doing tests using color GUI implementation I found a difference between P4GL and Java (Swing Color).

In P4GL an integer RGB value is constructed as: `red | (green << 8) | (blue << 16)`

In Java (Swing Color) the same expression is constructed as `blue | (green << 8) | (red << 16)`

I fixed this issue by providing 2 methods when RGB values are returned in class ColorRgb, one used when converted P4GL code is executed `ColorRgb.getRgb()` and the other `ColorRgb.getGuiRgb()` used when an RGB value is converted to a `java.awt.Color` object.

#39 - 09/03/2014 06:01 AM - Marius Gligor

- File *mag_upd20140903a.zip* added

I did some changes inside ColorTable class to load colors on a per user manner.
I've fixed also some javadoc texts.

#40 - 09/03/2014 10:04 AM - Greg Shah

Code Review 0903a

The changes look good.

1. WindowConfig is missing a history entry.
2. The call from the client to the server's LogicalTerminal.getColor() will be expensive. Should we pull down all colors for the referenced environment in one call instead of potentially making multiple calls to obtain all the colors?

#41 - 09/03/2014 11:00 AM - Marius Gligor

- File *mag_upd20140903b.zip* added

I fixed the issues reported on Code Review 0903a (note 39)

#42 - 09/03/2014 11:49 AM - Greg Shah

Code Review 0903b

The changes look good. I like the more general approach to reading color tables from the server.

What is the list of remaining work?

#43 - 09/03/2014 11:58 AM - Marius Gligor

I have no other works to do on this task.
Nevertheless they are other issues to implements like LOAD, USE, UNLOAD, PUT-KEY-VALUE which are common to both color and font tables.
I think that should be covered on other Redmine tasks.

#44 - 09/03/2014 12:12 PM - Greg Shah

I have no other works to do on this task.

OK, please get this regression tested (both conversion and runtime).

Nevertheless they are other issues to implements like LOAD, USE, UNLOAD, PUT-KEY-VALUE which are common to both color and font tables.
I think that should be covered on other Redmine tasks.

Please create a new task (make sure to document the details of the work needed in the task history and not in the description field). Make it related to this task and also to [#1649](#). Assign it to yourself. You can work that next.

#45 - 09/04/2014 07:44 AM - Marius Gligor

- File *mag_upd20140904a.zip* added

The changes in 0903b passed both conversion and regression tests. However I didn't do the commits and distributions yet because I found an interesting issue.

Working on task [#2385](#) related to environment and reading the mail from Ovidiu I found on OE documentation an interesting and important issue which was not implemented.

Let me explain. In OE manual related to the INI file/Registry entries for colors the following format is presented

colorn = { R , G , B , | colorname }

where:

- n color number (max. 255).

- R, G and B are red, green and blue integer values example 127,0,127

- colorname is the name of a system color example COLOR-SCROLLBAR

This means that we could add system colors values not only RGB values. I tested on the customer environment and it's true, it's works.

I created an environment INI file having the color section like:

```
[Colors]
;*****
; THE DEFINITION OF COLOR 0 THROUGH 15 IS PRIVATE TO THE PROGRESS ADE.
; MODIFYING COLORS 0 THROUGH 15 MAY PREVENT THE PROGRESS ADE FROM RUNNING.
; The following color definitions correspond to the ADE standards.
; 0 to 15 - reserved
color0=COLOR-ACTIVECAPTION
color1=0,0,128
color2=0,128,0
color3=0,128,128
color4=128,0,0
color5=128,0,128
color6=128,128,0
color7=128,128,128
color8=192,192,192
color9=0,0,0
; color10=0,255,0
; color11=0,255,255
; color12=255,0,0
; color13=255,0,255
; color14=255,255,0
; color15=255,255,255

NORMAL=0,15
INPUT=15,0
MESSAGES=15,1
```

Loading and using this custom environment the color 0 is mapped to COLOR-ACTIVECAPTION system color. When the mapping is not possible due to a wrong system color name the entry is mapped to color 0,0,0. As a result I did the appropriate changes in order to implement this future.

Please let me know if you agree my changes and if I need to repeat regression tests for this changes.

#46 - 09/04/2014 08:03 AM - Marius Gligor

- File mag_upd20140904b.zip added

A minor change to log a message whenever colors cannot be mapped and are mapped by default to a 0 RGB value.

#47 - 09/04/2014 09:03 AM - Greg Shah

Code Review 0904b

I am OK with the changes and I do think they are safe enough to avoid further testing.

I have a minor concern that we are taking the overhead of String.format() in every iteration of the main for loop in ColorTable.makeColorTable(), even though it is an unlikely case that the log message is needed. The only change I'd like to see is to make a private logging method that hides both the String.format() and the call to LOG.warning(). This can be called from all 3 locations, so we don't duplicate code.

It also may make sense to put a different bit of text in each of the 3 places, to explain the reason for the failure:

- system color name is invalid
- no system color is defined for the given (valid) system color name
- improperly formatted color string was provided on input

#48 - 09/04/2014 09:31 AM - Marius Gligor

- File mag_upd20140904c.zip added

Done.

#49 - 09/04/2014 09:46 AM - Greg Shah

Code Review 0904c

It is good. Go ahead and check it in/distribute it.

#50 - 09/04/2014 10:08 AM - Marius Gligor

- % Done changed from 80 to 100

0904c Passed conversion and regression tests. Committed revision 10605.

#51 - 09/04/2014 10:16 AM - Greg Shah

- Status changed from Review to Closed

#52 - 11/16/2016 12:12 PM - Greg Shah

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

Files

system_colors_article_on_openedge_hive_20140716.pdf	129 KB	07/16/2014	Greg Shah
orchard-windows.png	34.5 KB	07/28/2014	Marius Gligor
mag_upd20140814a.zip	560 KB	08/14/2014	Marius Gligor

everything.zip	95.2 KB	08/14/2014	Marius Gligor
color1.png	58.5 KB	08/19/2014	Marius Gligor
color2.png	57 KB	08/19/2014	Marius Gligor
mag_upd20140821a.zip	18 KB	08/21/2014	Marius Gligor
mag_upd20140822a.zip	18.1 KB	08/22/2014	Marius Gligor
progress.ini	5.62 KB	08/22/2014	Marius Gligor
color3.png	54.2 KB	08/22/2014	Marius Gligor
color4.png	50.8 KB	08/22/2014	Marius Gligor
color5.png	42.2 KB	08/22/2014	Marius Gligor
mag_upd20140822b.zip	29.4 KB	08/22/2014	Marius Gligor
mag_upd20140825a.zip	305 KB	08/25/2014	Marius Gligor
mag_upd20140828a.zip	321 KB	08/28/2014	Marius Gligor
color-nodes.xml	5.03 KB	08/28/2014	Marius Gligor
mag_upd20140902a.zip	356 KB	09/02/2014	Marius Gligor
mag_upd20140903a.zip	356 KB	09/03/2014	Marius Gligor
mag_upd20140903b.zip	356 KB	09/03/2014	Marius Gligor
mag_upd20140904a.zip	357 KB	09/04/2014	Marius Gligor
mag_upd20140904b.zip	357 KB	09/04/2014	Marius Gligor
mag_upd20140904c.zip	357 KB	09/04/2014	Marius Gligor