User Interface - Feature #1794

Feature # 2252 (Closed): implement GUI client support

implement font support

10/30/2012 09:55 AM - Greg Shah

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Constantin Asofiei	**% Done:**	100%
Category:		**Estimated time:**	80.00 hours
Target version:	GUI Support for a Complex ADM2 App		
billable:	No	**vendor_id:**	GCD

Description

Related issues:

Related to User Interface - Bug #2747: implement font table editing	**New**		
Related to User Interface - Feature #1811: implement the AJAX client driver	**Closed**	**11/01/2013**	**03/06/2014**
Related to User Interface - Bug #2701: default font text sizing issue	**Closed**		
Related to User Interface - Bug #2765: find proper freely licensed replacemen...	**Closed**		
Related to User Interface - Bug #2766: implement a text metrics server process	**New**		

History

#1 - 10/30/2012 09:59 AM - Greg Shah

1. FONT-TABLE
2. support for reading configuration to setup the base fonts
3. proper indexing of fonts
4. font-related attrs/methods for widgets
5. widget options COLUMN-FONT, LABEL-FONT
6. the FontManager implementation, see below for some thoughts about implementation for a Swing GUI (the AJAX implementation is another issue entirely):

-------- Original Message --------
Subject: Re: FontManager discussion
Date: Thu, 01 Dec 2011 23:01:47 +0200
From: Sergiy Yevtushenko <siy@goldencode.com>
Organization: ES@Home
To: ges@goldencode.com
CC: Asofiei, Constantin <ca@goldencode.com>

Gregory,

> A few weeks ago you said that the first pass FontManager would be stubbed out
> and that further development of that class would be a separate task. You also
> said that we needed to discuss that implementation/design.
>
> Please provide more detail on the issues, concerns and functionality of that
> class. We had better discuss it before you leave.

Basically 4GL assumes that exists some predefined list of fonts and then application references these fonts by order number in this list. This list is static and application has only one predefined way to change it via dedicated statement, which, in turn, just invokes system font dialog and replaces definition of specific font. In other words, application is completely isolated from raw font information. This is good for us and allows implement FontManager as simple simple list which is pre-filled during system startup from some configuration storage (directory, I guess). At present I've found very little number of font details which we actually need to know - at present required details are font height and font width. But actual drawing routines may require more details and/or support routines and they are hard to predict right now. I thought that implementing/improving FontManager is better to postpone to time

when actual drawing routines will be written and there will be more knowledge what actually should be in FontManager, how it will be configured and which other services it should provide beside already stubbed ones. Perhaps it worth keep FontManager very simple and pass font number into "drawing context" which is driver-specific and can interpret them in environment-specific way.

Thanks.
*----------

Software Products Team
Golden Code Development
http://www.goldencode.com

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

- Target version set to Milestone 12

#3 - 06/25/2014 02:09 PM - Greg Shah

- Parent task set to #2252

- Estimated time changed from 120.00 to 80.00

- Assignee set to Constantin Asofiei

#4 - 06/25/2014 03:50 PM - Constantin Asofiei

Do we need support for the FONT-TABLE handle? Note that this handle allows to:

- adjust the size of the actual FONT-TABLE
- get metrics in pixels/character units for a given text, using a given font.

#5 - 06/25/2014 03:51 PM - Greg Shah

We won't know if we need FONT-TABLE support (or not) until we see the cut-down customer GUI code. The full customer GUI code did use it, if I recall correctly.

#6 - 06/30/2014 09:40 AM - Constantin Asofiei

- File ca_upd20140630a.zip added

Attached update fixes a problem in the expression compiler: the bytecode builder was using ldc, which can refer 1-byte constant-pool indexes. This works until the statement needs to refer a constant-pool index greater than 255. To solve this, ldc_w needs to be used, which uses two bytes for the constant-pool index.

About the font management infrastracture on server side:

1. new p2j.ui.FontTable class, with server-side implementation of:
   - the FONT-TABLE handle's attributes/methods
   - SYSTEM-DIALOG FONT via FontTable.chooseFont.
   - PUT-KEY-VALUE FONT {#|ALL}, via FontTable.saveFontTable and FontTable.saveFont(font-num)
2. new p2j.ui.FontTableResource interface, defining the FONT-TABLE resource.

For a P2J client, the font-table is initialized from server-side and pushed to the client-side, when the client starts; this includes legacy metrics related to the configured fonts. The actual font table is kept in the existing p2j.ui.client.FontManager class. The metrics for the string literals (which are used to compute widget metrics) can be interrogated from the server-side on first hit and saved in a cache.

On server-side, the font-table is kept in the directory, and can be specified per P2J account or per P2J server. We can choose a combined approach, similar to the appserver's clientConfig node:

- a font table entry defined per account has precedence over a font table entry defined per server
- if a user changes the font table, it will be saved at the account, not the server

This approach will act like if each 4GL client was started with its own progress.ini file (thus the environment is separated). The plus side for this approach is that each user can set his/hers own font/color scheme, especially if at some point we decide to allow the GUI client access to a font/color editor, outside of what the legacy 4GL application allowes: this is an alternative to what the user can do now, i.e. changing the Windows color scheme/fonts can affect the GUI.

Also, we need to decide how we configure the physical fonts:

- an approach is to keep all the physical font files with the P2J server, and the GUI client will dynamically load these fonts. I think we already need this for the bitmap fonts.
- second approach is for the font-table to rely only on the fonts available in the OS.
- third approach is a combination: custom fonts loaded in P2J (which includes bitmap fonts + any application-specific font) plus the fonts available in the OS.
- finally, all of the above are under the constraint that P2J needs to know the legacy metrics of the fonts used in the font-table.

An unknown at this time is if there might be licensing issues if P2J (or the converted application) is shipped with some 3rd-party fonts. If at least the standard fonts are shipped with the application, then the application should look the same on any platform, and solves problems related with font installation.

#7 - 07/01/2014 05:43 AM - Constantin Asofiei

0630a.zip passed conversion testing - no errors/differences after MAJIC conversion.

#8 - 07/02/2014 06:12 AM - Constantin Asofiei

0630a.zip was committed to bzr rev 10558

#9 - 07/02/2014 06:37 AM - Greg Shah

> On server-side, the font-table is kept in the directory, and can be specified per P2J account or per P2J server. We can choose a combined approach, similar to the appserver's clientConfig node:
> - a font table entry defined per account has precedence over a font table entry defined per server
> - if a user changes the font table, it will be saved at the account, not the server
>
> This approach will act like if each 4GL client was started with its own progress.ini file (thus the environment is separated). The plus side for this approach is that each user can set his/hers own font/color scheme, especially if at some point we decide to allow the GUI client access to a font/color editor, outside of what the legacy 4GL application allowes: this is an alternative to what the user can do now, i.e. changing the Windows color scheme/fonts can affect the GUI.

This makes good sense.

> Also, we need to decide how we configure the physical fonts:
>
> - an approach is to keep all the physical font files with the P2J server, and the GUI client will dynamically load these fonts. I think we already need this for the bitmap fonts.
> - second approach is for the font-table to rely only on the fonts available in the OS.
> - third approach is a combination: custom fonts loaded in P2J (which includes bitmap fonts + any application-specific font) plus the fonts available in the OS.

- finally, all of the above are under the constraint that P2J needs to know the legacy metrics of the fonts used in the font-table.

I think the 3rd approach is the correct one. It allows the application to add fonts if needed, but to take advantage of font support that is already there when possible.

> An unknown at this time is if there might be licensing issues if P2J (or the converted application) is shipped with some 3rd-party fonts. If at least the standard fonts are shipped with the application, then the application should look the same on any platform, and solves problems related with font installation.

The converted application will have to license and ship the needed fonts with their code OR they will have to require their users to license and install those fonts.

I don't plan for P2J to include the fonts.

#10 - 07/02/2014 07:01 AM - Constantin Asofiei

- File ca_upd20140702d.zip added

I've cleaned up the code. This update is related to conversion support and stubbed runtime for FONT, LABEL-FONT, COLUMN-FONT, TITLE-FONT, PIXELS-PER-ROW/COLUMN, the FONT-TABLE handle. I'm putting this through testing now and I will release it, if it passes testing.

The only unknown at this time related to conversion is: FONT, LABEL-FONT, COLUMN-FONT, TITLE FONT (plus COLOR counterparts) options can be dynamic. I think we should add these on a as-needed basis.

#11 - 07/02/2014 09:23 AM - Constantin Asofiei

Some other notes. A new window can use the fonts from a dynamic environment, activated via the LOAD/USE statements. Actually, a new window will always consider the current environment configuration, when reading its font table. Thus:

- if the application reads environments dynamically and relies on them for the font-table, this will mean that the i.e. progress.ini or registry keys (associated with that environment) will need to be available.
- when instantiating the font-table for a specific window, the font-table in the directory will be the "fallback" font-table. Any explicitly loaded environment will have precedence over the directory settings.
- as DefaultFont and DefaultFixedFont are not part of the fonts section, they are static and will not be changed (even if a new environment is loaded).

#12 - 07/02/2014 10:09 AM - Greg Shah

Code Review 0702d

The code looks good.

> The only unknown at this time related to conversion is: FONT, LABEL-FONT, COLUMN-FONT, TITLE FONT (plus COLOR counterparts) options can be dynamic. I think we should add these on a as-needed basis.

Agreed.

#13 - 07/04/2014 03:05 AM - Constantin Asofiei

0702d.zip passed runtime testing. Will merge and release after the release of ges_upd20140627a.zip from #2252.

#14 - 07/11/2014 05:42 AM - Constantin Asofiei

- File ca_upd20140707b.zip added

Replacement for 0702d. Committed to bzr rev 10566.

#15 - 07/11/2014 06:01 PM - Constantin Asofiei

- File ca_upd20140711b.zip added

This version is going through runtime testing now. What I need to finish is some manual testing and the TRPL rules to collect the static strings (which require legacy metrics).

The key changes are:

- the WINDOW's MESSAGE-AREA needs its own widget
- GUI widgets will need to provide a physical dimension (in pixels), similar to physicalLocation.
- the widget's font needs to be provided via a resolveFont API. This is because the FONT attribute is not what it gives the font used in drawing: this is resolved from i.e. the frame, window, etc (in a cascade approach).

#16 - 07/13/2014 02:44 PM - Constantin Asofiei

Runtime testing showed a problem related to the custom auth plugins: in GUI mode, for the window to initialize, it requires knowledge about the font table, fonts, etc. But, ThinClient.initializePrep will activate the default window BEFORE the client authenticates (and has a proper context), thus interrogating the server via the server exports is not possible.

To solve this, I think the custom auth plugin will have to hard-code the font details used to draw the login window, and use its own distinct DEFAULT-WINDOW. After the client has logged in, a new DEFAULT-WINDOW will be created, using the server's font-table.

#17 - 07/13/2014 05:50 PM - Greg Shah

> To solve this, I think the custom auth plugin will have to hard-code the font details used to draw the login window, and use its own distinct DEFAULT-WINDOW. After the client has logged in, a new DEFAULT-WINDOW will be created, using the server's font-table.

This makes sense.

#18 - 07/14/2014 07:13 AM - Constantin Asofiei

- File ca_upd20140714d.zip added

A sample of how the directory configuration will look:

```
        <node class="container" name="font-table">
          <node class="string" name="default-font">
            <node-attribute name="value" value="MS Sans Serif, size 10"/>
          </node>
          <node class="string" name="default-fixed-font">
            <node-attribute name="value" value="Courier New, size 10"/>
          </node>
          <node class="string" name="font0">
            <node-attribute name="value" value="Tahoma, size 10"/>
          </node>
          <node class="string" name="font1">
            <node-attribute name="value" value="System, size 8"/>
          </node>
          <node class="string" name="font2">
            <node-attribute name="value" value="Microsonft Sans Serif, size 10"/>
          </node>
        </node>
        <node class="container" name="system-font-table">
          <node class="string" name="window-title-font">
            <node-attribute name="value" value="Tahoma, size 10"/>
          </node>
        </node>
        <node class="container" name="custom-fonts">
          <node class="container" name="font1">
            <node class="string" name="font">
              <node-attribute name="value" value="MS Sans Serif"/>
            </node>
            <node class="string" name="file">
              <node-attribute name="value" value="DejaVuSans.ttf"/>
            </node>
          </node>
          <node class="container" name="font2">
            <node class="string" name="font">
              <node-attribute name="value" value="Courier New"/>
            </node>
            <node class="string" name="file">
              <node-attribute name="value" value="cr.ttf"/>
            </node>
          </node>
        </node>
```

Also, attached there are samples of the text and font metrics XMLs

#19 - 07/14/2014 07:16 AM - Constantin Asofiei

- File ca_upd20140714c.zip added

This version contains some FontTable fixes and is going through runtime testing again (hopefully the last one).

#20 - 07/14/2014 01:59 PM - Constantin Asofiei

The failures after testing 0714c.zip look like false negatives, I'm running again to confirm they will pass.

#21 - 07/14/2014 03:07 PM - Constantin Asofiei

- File ca_upd20140714a.zip added

This update changes the frame_generator.xml to collect labels, titles and text widgets, and write them to a ui_strings.txt file.

The tools following tools in testcases/uast/fonts/:

```
uast/fonts/system.metrics/get-font-metrics.p
uast/fonts/system.metrics/get-text-metrics.p
```

where modified to generate proper XMLs and also to allow more granular approach at the font details (i.e. specify a font size and style, to limit the collected metrics only for the needed cases).

#22 - 07/14/2014 04:31 PM - Greg Shah

Code Review 0714c

Some attributes are implemented only with methods that take wrappers but without overloaded versions for primitives. See BrowseColumnWidget.setColumnFont()/setLabelFont() and BaseEntity.setFont().

Otherwise, the posted update looks like a major step forward. I especially like the way you have moved code out of window and into the message/status line widgets.

#23 - 07/14/2014 04:38 PM - Greg Shah

Code Review 0714a

The changes look fine except for the post-rules where strings with the embedded characters (&, | or \n) will be output in both split and unsplit forms. Assuming that is intended, then everything is OK.

#24 - 07/15/2014 03:12 AM - Constantin Asofiei

- File ca_upd20140714g.zip added

Left TODOs related to fonts. This can be worked best when we have the GUI has some minimal widget support

- changing the font-table via the SYSTEM-DIALOG FONT and PUT KEY VALUE FONT (FontTable.chooseFont, saveFont and saveFontTable APIs)
- FontManager.createFont - underline and default size
- support bitmap fonts
- THREE-D support (from default environment and window)

Other TODOs are mainly related to window or widget drawing.

> Some attributes are implemented only with methods that take wrappers but without overloaded versions for primitives. See BrowseColumnWidget.setColumnFont()/setLabelFont() and BaseEntity.setFont().

The primitive overloads are in GenericWidget, which call the wrapper versions - so the widgets need to implement only the wrapper versions.

> 0714a: The changes look fine except for the post-rules where strings with the embedded characters (&, | or \n) will be output in both split and unsplit forms. Assuming that is intended, then everything is OK.

Yes, this was by intent.

Attached update is replacement for 0714c.zip, passed runtime testing, and was committed to bzr rev 10571.

#25 - 07/15/2014 11:29 AM - Constantin Asofiei

- % Done changed from 0 to 70

- Status changed from New to WIP

> Code Review 0714a

Passed conversion testing (both MAJIC and server project). Committed to bzr rev 10574.

#26 - 03/17/2015 10:34 AM - Constantin Asofiei

- File ca_upd20150317b.zip added

Added a hard-coded standard font table plus other misc fixes/enhancements related to font management.

#27 - 03/17/2015 11:15 AM - Greg Shah

Code Review ca_upd20150317b.zip

The update looks very good.

1. As far as I can tell, the use of the 4GL is not really needed for what you are doing in get-[text|font]-metrics.p. In other words, these programs could have been written in C? If my understanding is correct, there would be a nice advantage in both performance and reduction in dependencies. So that we could run this on systems that didn't have the 4GL and that would allow capture and/or comparison of metrics in a more flexible way. I'm not suggesting that you rewrite this now. But I want to know if my assessment is correct so that we might decide to rewrite them later.

2. In FontManager.getMaxFontWidth(), isn't FontDetails fd always going to be the same as FontDetails details?

3. Do we need to check with Guy to confirm that the windev01 font setup is the same as the Progress default font setup?

#28 - 03/17/2015 11:22 AM - Constantin Asofiei

Greg Shah wrote:

> 1. As far as I can tell, the use of the 4GL is not really needed for what you are doing in get-[text|font]-metrics.p. In other words, these programs could have been written in C? If my understanding is correct, there would be a nice advantage in both performance and reduction in dependencies. So that we could run this on systems that didn't have the 4GL and that would allow capture and/or comparison of metrics in a more flexible way. I'm not suggesting that you rewrite this now. But I want to know if my assessment is correct so that we might decide to rewrite them later.

Yes, your assessment is correct, we should rewrite this in C. It was simpler for me to rely on 4GL rather than C, as C is not one of my favourite languages (especially WIN32/GDI APIs). Plus it allowed me to run this on windev01 without any other setup/compilation/etc.

> 2. In FontManager.getMaxFontWidth(), isn't FontDetails fd always going to be the same as FontDetails details?

Yes, you are correct. I'll fix it.

> 3. Do we need to check with Guy to confirm that the windev01 font setup is the same as the Progress default font setup?

Their setup is not the same as the default font setup (at least DEFAULT-FONT and DEFAULT-FIXED-FONT are commented out). We will need an explicit font table for them.

#29 - 03/17/2015 11:25 AM - Greg Shah

> 3. Do we need to check with Guy to confirm that the windev01 font setup is the same as the Progress default font setup?
>
> Their setup is not the same as the default font setup (at least DEFAULT-FONT and DEFAULT-FIXED-FONT are commented out). We will need an explicit font table for them.

OK, so the update has the encoded Progress 4GL default font setup (and NOT the customer setup)?

#30 - 03/17/2015 11:26 AM - Constantin Asofiei

Greg Shah wrote:

> 3. Do we need to check with Guy to confirm that the windev01 font setup is the same as the Progress default font setup?
>
> Their setup is not the same as the default font setup (at least DEFAULT-FONT and DEFAULT-FIXED-FONT are commented out). We will need an explicit font table for them.
>
> OK, so the update has the encoded Progress 4GL default font setup (and NOT the customer setup)?

Correct.

#31 - 03/17/2015 11:28 AM - Greg Shah

Perfect. After you make that one tweek, please do get this regression tested.

#32 - 03/17/2015 06:06 PM - Constantin Asofiei

- File ca_upd20150317c.zip added

Greg Shah wrote:

> Perfect. After you make that one tweek, please do get this regression tested.

Attached version has passed runtime testing.

#33 - 03/17/2015 06:16 PM - Greg Shah

Please check it in.

#34 - 03/17/2015 06:21 PM - Constantin Asofiei

Greg Shah wrote:

> Please check it in.

Committed revision 10814.

#35 - 05/15/2015 03:28 PM - Greg Shah

As we consider font compatibility issues, the following are some useful references:

http://en.wikipedia.org/wiki/Computer_font
http://en.wikipedia.org/wiki/MS_Sans_Serif
http://en.wikipedia.org/wiki/Core_fonts_for_the_web
http://opensource-usability.blogspot.com/2015/05/fonts-for-desktop.html
http://www.ooninja.com/2008/02/metrical-equivalent-fonts-and-font.html
http://corefonts.sourceforge.net/
http://en.wikipedia.org/wiki/Liberation_fonts
http://en.wikipedia.org/wiki/Croscore_fonts
http://en.wikipedia.org/wiki/Ascender_Corporation

#36 - 05/15/2015 04:27 PM - Constantin Asofiei

About using the font/text metrics reported by the driver for widget size formulas. I've tested the Courier New, Microsoft Sans Serif, Segoe UI and Tahoma fonts (all True-type), and captured the AWT font metrics for:

1. scaled via DPI

2. scaled via legacy metrics
3. scaled via DPI + legacy metrics

When compared with the native Windows metrics, the best results were from the DPI scalinng. But even if some font sizes match 100%, for others there are differences. So, there is no way to rely on computing the font metrics using AWT APIs. More, when capturing the metrics from linux (I've defined the fonts via Font.createFont, as these are not in the OS), the metrics don't even match the AWT counterparts captured on Windows.

Next, I've checked how the string metrics vary, for each font scale type (DPI, legacy metrics, DPI + legacy metrics). Again, best solution was using a font scaled via DPI, with a +/- 1 to 3 pixels for height and width. From these, at some points width matches almost fully, while height doesn't.

Finally, I've checked the metrics for a Java logical font, Serif - again, the metrics are not reliable, Windows and Linux produce different results.

The conclusion is that we need the text and font metrics as on the legacy OS, to be able to duplicate the widget sizes; otherwise, the frame layout may be corrupted. After this:

- use the DPI-scaled font for drawing
- when editing/drawing text, we need to rely on the text/font metrics provided by the driver (using the scaled font), so that i.e. word spliting, caret drawing, etc will be done properly.

Bitmap fonts. I was hoping to find something like TwelveMonkeys, which can "plug in" support for different image types directly into the JRE, but this isn't the case with the bitmap fonts. The AWT's Font class can't be used with bitmap fonts, even by libgdx. So, libgdx uses its own BitmapFont class, and uses a code like this to define it (although setting the "size" parameter for bitmap fonts doesn't have an effect - only true-type fonts use it):

```
        FreeTypeFontGenerator generator = new FreeTypeFontGenerator(Gdx.files.internal("sserife.fon"));
        FreeTypeFontParameter parameter = new FreeTypeFontParameter();
        parameter.size = size;
        BitmapFont font = generator.generateFont(parameter);
        generator.dispose();
```

Beside this, the complex part is that this requires the entire GDX infrastructure to be initialized, to be able to draw using this font, as the generator.generateFont API can not be used from outside of a libgdx rendering context (i.e. AWT event queue can't use this API, as it requires access to a OpenGL context, which is set only by ligdx).

For Swing, my conclusion is that we either find alternatives for the unsupported bitmap fonts (match metrics and look as good as possible) or add bitmap font rendering of our own. Integrating libgdx into the Swing driver provides more complexity than what we will gain from it.

Another unknown is what the Web driver will be able to do: does this support bitmap fonts? From what I could find, the font-face CSS3 statement doesn't support bitmap fonts. More, when defining custom fonts via the directory, we should provide a URL to the web driver, to download the font.

So, to resume:

- use DPI scaling for the swing driver
- legacy font&text metrics need to be captured for all cases which affect widget size
- drawing should rely on what the driver reports as metrics for the current font/text
- bitmap fonts might be problematic even for the web driver

#37 - 05/15/2015 04:51 PM - Greg Shah

I have an idea what scaled via DPI and scaled via legacy metrics mean. Can you explain what you did for scaled via DPI + legacy metrics?

> The conclusion is that we need the text and font metrics as on the legacy OS, to be able to duplicate the widget sizes; otherwise, the frame layout may be corrupted.

Are you proposing that the legacy text/font metrics are only to be used for layout?

Doesn't layout depend on the default system font (type and size as specified in the Progress config), no matter what actual font is in use? If so, then does that mean we only need to capture legacy metrics for the default system font?

> use the DPI-scaled font for drawing
> when editing/drawing text, we need to rely on the text/font metrics provided by the driver (using the scaled font), so that i.e. word spliting, caret drawing, etc will be done properly.

I think I'm OK with this, but I am worried that the differences in text output will make automated testing impossible.

> For Swing, my conclusion is that we either find alternatives for the unsupported bitmap fonts (match metrics and look as good as possible)

Yes, I think this makes sense.

> Another unknown is what the Web driver will be able to do: does this support bitmap fonts?

As far as I know, browsers don't support bitmap fonts. Bitmap fonts was something used originally for speed. But as vector font processing has become feasible with better CPU implementations, bitmap fonts have fallen out of favor since they don't scale well. On Windows this is a legacy thing, since many old apps and dialogs still rely upon those fonts. Sadly, Progress is one of those. :(The only other type of software that still uses bitmap fonts may be games. Since the web environment has no compatibility requirements, I think there is no reason for them to add bitmap font support now or in the future.

In other words, we really will want to duplicate the look using vector font support, whether in AWT/Swing or the web.

#38 - 05/15/2015 05:02 PM - Constantin Asofiei

Greg Shah wrote:

> I have an idea what scaled via DPI and scaled via legacy metrics mean. Can you explain what you did for scaled via DPI + legacy metrics?

What I tried to do was to bring the metrics as close as possible to the legacy ones. So, if the metrics after DPI scaling don't match the legacy metrics, I tried to bring the font to match, by using a AffineTransform.getScaleInstance(legacyWidth/fontWidth, legacyHeight/fontHeight) transformation (this is what I refer to as legacy metrics scaling). But this gets worse results, unfortunately.

> The conclusion is that we need the text and font metrics as on the legacy OS, to be able to duplicate the widget sizes; otherwise, the frame layout may be corrupted.

> Are you proposing that the legacy text/font metrics are only to be used for layout?

Yes. Although in some cases (like labels or TEXT widgets) we might need to rely on some kind of scaling, if the text doesn't fit the boundaries (especially if it slarger). I already do this for TEXT widget, but I didn't find a case yet for labels.

> Doesn't layout depend on the default system font (type and size as specified in the Progress config), no matter what actual font is in use? If so, then does that mean we only need to capture legacy metrics for the default system font?

No. The layout depends on the widget's current font. The fallback goes like this: widget font -> frame font -> default-font. That's why we need metrics for the entire font table.

> use the DPI-scaled font for drawing
> when editing/drawing text, we need to rely on the text/font metrics provided by the driver (using the scaled font), so that i.e. word spliting, caret drawing, etc will be done properly.

> I think I'm OK with this, but I am worried that the differences in text output will make automated testing impossible.

Yes, this will pose problems in automated testing. But at least it doesn't interfere with the layout - which is one of the major components which relies on these metrics, and we need it to be accurate.

#39 - 05/19/2015 09:32 AM - Constantin Asofiei

If the font size is missing, 4GL looks like it defaults to these:

1. courier new: 12
2. segoe ui: 12
3. tahoma: 12
4. microsoft sans serif: 12
5. ms sans serif: 10 (12 does not match)
6. system: 10 (12 matches too)
7. fixedsys: 10 (12 matches too)

I will add these default sizes to the font-table.xml.

#40 - 05/19/2015 09:59 AM - Constantin Asofiei

- File ca_upd20150519b.zip added

This update adds:

- support for custom fonts in the Swing GUI driver
- support for implicit font size used by 4GL
- fixed FONT option at the DEFINE VAR stmt

#41 - 05/19/2015 12:53 PM - Greg Shah

Code Review ca_upd20150519b.zip

The changes look good.

1. In regard to this todo in SwingGuiDriver.createFont():

```
// TODO: if a new font is created from the byte array, then register it with the graphics
// for all windows
```

How important is this or is this just an optimization?

2. Another optimization to consider would be to push the entire FontTable configuration from the server to the client at client startup. This would avoid the up-calls during FontManager processing.

#42 - 05/19/2015 01:13 PM - Constantin Asofiei

Greg Shah wrote:

> Code Review ca_upd20150519b.zip
>
> The changes look good.
>
> 1. In regard to this todo in SwingGuiDriver.createFont():
>
> [...]
>
> How important is this or is this just an optimization?

That comment is obsolete, I forgot to remove it. When drawing text, the font needs to be selected in the Graphics object before the operation... so no registration is required.

> 2. Another optimization to consider would be to push the entire FontTable configuration from the server to the client at client startup. This would avoid the up-calls during FontManager processing.

Yes, this is a good idea. I think we can push the metrics and definitions for the fonts defined in the font table fast. The legacy metrics for the strings it's a little tricky, because currently we don't know with which font a string will be used. We can push the string metrics for all fonts in the font table (so each string will initially have font-table-size metrics) - this will reduce the load, especially if metrics are captured for other fonts/font styles than what the font table uses.

#43 - 05/20/2015 07:41 AM - Constantin Asofiei

- File ca_upd20150520d.zip added

This version is built on top of 0519b.zip and improves the font table and metric loading + some other GUI-related fixes.

#44 - 05/21/2015 07:12 AM - Constantin Asofiei

0520d.zip has passed runtime testing.

#45 - 05/23/2015 10:06 AM - Greg Shah

Code Review ca_upd20150520d.zip

It looks good. This is a nice improvement over my previous approach (yours is much cleaner). Please commit it.

The only question I have is whether FontDetails and FontStyle should be moved to the com.goldencode.p2j.ui package. It seems these are not just client classes anymore. Even if we should do this, there is no reason to hold back the commit. The change can be made in a future update.

#46 - 05/25/2015 03:19 AM - Constantin Asofiei

Greg Shah wrote:

> Code Review ca_upd20150520d.zip
>
> It looks good. This is a nice improvement over my previous approach (yours is much cleaner). Please commit it.

Committed revision 10869.

> The only question I have is whether FontDetails and FontStyle should be moved to the com.goldencode.p2j.ui package. It seems these are not just client classes anymore. Even if we should do this, there is no reason to hold back the commit. The change can be made in a future update.

Yes, I think they should be moved.

#47 - 06/16/2015 10:17 AM - Greg Shah

There is a very useful discussion and tutorial in #2546 notes 92 and 98. This explains some problems with matching font metrics and also describes a process by which the sizing formulas can be determined for a given widget type.

#48 - 08/08/2015 10:52 AM - Constantin Asofiei

Created task branch 1794c from trunk rev 10917.

#49 - 08/08/2015 10:57 AM - Constantin Asofiei

Please review branch 1794c rev 10918.

I've added support for a new per-user node, font-aliases. In this container, the keys represent a font-table or system-font-table key (font1, window-title-font, etc) or default-font or default-fixed-font for the default fonts. When such a key exists, the driver will override the font specifications (to create the driver-level font) with the ones specified in the alias. The font's legacy metrics will remain unchanged - only the drawing font is overridden.

These font aliases also support "custom font" definitions - as the custom fonts are mapped via the font's name (and not the internal key), a font alias can have a custom font attached.

#50 - 08/09/2015 01:01 PM - Greg Shah

Code Review Task Branch 1794c Revision 10918

I'm good with the changes, except I think the ds.unbind() in FontTable.readFontAliases() should be in a finally block.

These seem pretty independent of other changes. I'm let you make the call on when they can be merged to trunk. Unless there is a file-level conflict, I think we don't have to make others re-test after rebase.

#51 - 08/09/2015 04:59 PM - Constantin Asofiei

> I'm good with the changes, except I think the ds.unbind() in FontTable.readFontAliases() should be in a finally block.

I've fixed the ds.unbind in both places.

> I'm let you make the call on when they can be merged to trunk.

I will wait until 1790d and 2567a are merged to trunk - this is will save them a rebase.

#52 - 08/11/2015 12:15 PM - Constantin Asofiei

Rebased branch 1794c from trunk rev 10918 (new branch rev 10920).

#53 - 08/12/2015 09:26 AM - Constantin Asofiei

Rebased branch 1794c from trunk rev 10919 (new branch rev 10921).

#54 - 08/17/2015 06:56 AM - Constantin Asofiei

Rebased branch 1794c from trunk rev 10923 (new branch rev 10925).

#55 - 08/18/2015 03:28 PM - Constantin Asofiei

Rebased branch 1794c from trunk rev 10924 (new branch rev 10926).

#56 - 08/19/2015 03:50 AM - Constantin Asofiei

Rev 10927 contains support for underline fonts. No support for drawing text with partially underlined text yet at the driver level - this should be done by drawing the prefix, underlined text, suffix.

Greg: I want to merge 1794c today if that's OK.

#57 - 08/19/2015 08:36 AM - Greg Shah

Code Review Task branch 1794c Revision 10927

I'm fine with the changes, except SwingGuiDriver needs a history entry.

I don't think we need support for partial underlining. It is OK to take your approach.

> I want to merge 1794c today if that's OK.

Go ahead.

#58 - 08/19/2015 08:46 AM - Constantin Asofiei

Merged to trunk rev 10925, notification email sent and branch 1794c archived.

#59 - 10/09/2015 09:18 AM - Greg Shah

- % Done changed from 70 to 100

- Status changed from WIP to Closed

The last of the font work will be completed in #1811 with some discussion taking place in #2322. As such, I'm closing this task.

#60 - 10/14/2015 02:40 PM - Constantin Asofiei

See #2701 note 4 for font replacements via custom-fonts.

#61 - 12/08/2015 08:38 AM - Greg Shah

Useful KB article:

http://knowledgebase.progress.com/articles/Article/P123946

I think this is exactly what we already do, but I just wanted to store the link for future reference.

#62 - 11/16/2016 12:12 PM - Greg Shah

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

Files

ca_upd20140630a.zip	36 KB	06/30/2014	Constantin Asofiei
ca_upd20140702d.zip	566 KB	07/02/2014	Constantin Asofiei
ca_upd20140707b.zip	565 KB	07/11/2014	Constantin Asofiei
ca_upd20140711b.zip	437 KB	07/11/2014	Constantin Asofiei
ca_upd20140714d.zip	1.14 KB	07/14/2014	Constantin Asofiei
ca_upd20140714c.zip	441 KB	07/14/2014	Constantin Asofiei
ca_upd20140714a.zip	33.8 KB	07/14/2014	Constantin Asofiei
ca_upd20140714g.zip	438 KB	07/15/2014	Constantin Asofiei

ca_upd20150317b.zip	186 KB	03/17/2015	Constantin Asofiei
ca_upd20150317c.zip	186 KB	03/17/2015	Constantin Asofiei
ca_upd20150519b.zip	180 KB	05/19/2015	Constantin Asofiei
ca_upd20150520d.zip	356 KB	05/20/2015	Constantin Asofiei