User Interface - Feature #3332

Add support for load offset and load size parameters of LOAD-IMAGE* methods of BUTTON and IMAGE widgets

09/05/2017 11:15 AM - Hynek Cihlar

Status:	Closed	**Start date:**	
Priority:	Normal	**Due date:**	
Assignee:	Hynek Cihlar	**% Done:**	100%
Category:		**Estimated time:**	0.00 hour
Target version:			
billable:	No	**version:**	
vendor_id:	GCD		

Description

Related issues:

Related to User Interface - Feature #2479: DEFINE IMAGE and image phrase supp...	**Closed**
Related to User Interface - Feature #2480: add support for button and image a...	**Closed**
Related to User Interface - Bug #3504: sizing and centering issues with butto...	**New**

History

#1 - 09/05/2017 12:21 PM - Hynek Cihlar

FWD is missing conversion and runtime support for the following methods IMAGE:LOAD-IMAGE(), BUTTON:LOAD-IMAGE(), BUTTON:LOAD-IMAGE-UP(), BUTTON:LOAD-IMAGE-DOWN() and BUTTON:LOAD-IMAGE-INSENSITIVE() with image load offset and image load size parameters.

The issue was identified in #3330, see #3330-39 and #3330-103.

#3 - 09/05/2017 12:25 PM - Hynek Cihlar

- Related to Feature #2479: DEFINE IMAGE and image phrase support for both button and image added

#4 - 09/05/2017 12:25 PM - Hynek Cihlar

- Related to Feature #2480: add support for button and image attributes and options added

#5 - 09/05/2017 12:30 PM - Hynek Cihlar

3332a revision 11164 implements full conversion support for the missing LOAD-IMAGE* overloads. It also adds initial runtime implementation for both IMAGE and BUTTON widgets focusing on the happy paths.

Still missing are the negative cases - invalid values for image load params, and the widget size effects when load size parameter is set. See uast/image/image_load-image.p and uast/image/button_load-image.p for some of the negative cases.

#6 - 09/05/2017 12:36 PM - Greg Shah

Eugenie: Please review 3330a revision 11189 for the initial runtime stubs that were needed. Also, please review 3332a revision 11164.

#7 - 09/05/2017 08:33 PM - Eugenie Lyzenko

Greg Shah wrote:

> Eugenie: Please review 3330a revision 11189 for the initial runtime stubs that were needed. Also, please review 3332a revision 11164.

I'm OK with stubs implementation in 3332a rev 11164. In 3330a rev 11189 I'm in general OK however I guess it will be better to have single

```
UnimplementedFeature.partial("Offset and size parameters are ignored");
```

per loadImage* family. For the case the some option is missing for example instead of:

```
   public logical loadImage(character name, integer xOffset)
   {
      UnimplementedFeature.partial("Offset and size parameters are ignored");
      return loadImage(name);
   }
```

have:

```
   public logical loadImage(character name, integer xOffset)
   {
      return loadImage(name, xOffset, null, null, null);
   }
```

And placing the error inside
logical loadImage(character name, integer xOffset, integer yOffset, integer width, integer height)

#8 - 09/06/2017 02:36 AM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Greg Shah wrote:
>
> > Eugenie: Please review 3330a revision 11189 for the initial runtime stubs that were needed. Also, please review 3332a revision 11164.

I'm OK with stubs implementation in 3332a rev 11164. In 3330a rev 11189 I'm in general OK however I guess it
will be better to have single
[...]
per loadImage* family. For the case the some option is missing for example instead of:
[...]
have:
[...]

And placing the error inside
logical loadImage(character name, integer xOffset, integer yOffset, integer width, integer height)

I am not against code deduplication, but in this case the warnings nicely and statically localize the places that are unfinished. If we move it in a common place than the unfinished places are not that obvious.

#9 - 09/06/2017 09:09 AM - Greg Shah

I prefer having a single worker that is unfinished instead of multiple independent locations that are not done. I agree that one must look a little deeper to see the backing code that is not implemented, but the long term result is better. Since we are going to implement this in the short term anyway, it should be fine.

#10 - 10/30/2017 04:43 AM - Hynek Cihlar

3332a rebased against latest trunk.

#11 - 11/02/2017 05:53 PM - Hynek Cihlar

3332a contains the implemented IMAGE and BUTTON load size and offset plus related changes. Also the branch includes the workaround for the limited command line in Windows during build.

The branch is rebased against latest trunk, and is currently undergoing ChUI and GUI regression testing.

Please review.

#12 - 11/03/2017 05:57 AM - Greg Shah

Code Review Task Branch 3332a Revision 11201

1. In the versions of ButtonWidget.loadImage*() which have integer type parameters, the direct use of intValue() will throw an NPE when the instance is unknown. This would be different behavior than the 4GL. It would probably be best to hide this checking inside an internal worker so that the code everywhere doesn't have to be more complicated. Likewise, it is not clear if the automatic conversion of an unknown name to "?" is the correct behavior.

2. Can decimal or int64 values be passed to the load-image() methods?

3. In ButtonWidget.loadImageUp(String name) at line 1653 the code uses loadImageDownInt() instead of loadImageUpInt().

4. We don't support the full range of method signatures for ButtonWidget.loadImage*(). We are currently missing the "mixed" primitive and BDT cases. For example, loadImage*(String, integer, int, integer) and so forth. Do we mark any such mixed cases for wrapping at conversion time (the primitives get wrapped in this case and only in this mixed case)?

5. I don't understand the implementation of ButtonWidget.loadImageInt(). All the offset and width/height data is used to update a local ButtonImageDefinition instance named imgDef. This is only used for calculating if pushScreenDefinition() is called but the imgDef instance itself is discarded. And the offset/size data is discarded and seems to have no effect on the actual load. I don't even understand why the call to pushScreenDefinition() is made since there is no change to the configuration state that is made so there is nothing to be pushed. I must be missing something there.

6. Items 1, 2 and 4 above also apply to ImageWidget.

7. I can't assess the validity of the changes to ButtonGuiImpl.height(), ButtonGuiImpl.width() and IMageGuiImpl.calculateDrawingParameters(). How have you confirmed correctness?

Eugenie: Please review that code.

#13 - 11/03/2017 06:35 AM - Hynek Cihlar

Greg Shah wrote:

> Code Review Task Branch 3332a Revision 11201
>
> 1. In the versions of ButtonWidget.loadImage*() which have integer type parameters, the direct use of intValue() will throw an NPE when the instance is unknown. This would be different behavior than the 4GL. It would probably be best to hide this checking inside an internal worker so that the code everywhere doesn't have to be more complicated. Likewise, it is not clear if the automatic conversion of an unknown name to "?" is the correct behavior.

Will fix.

> 2. Can decimal or int64 values be passed to the load-image() methods?

decimal will cause compile-time error but int64 can be used. I will fix this.

> 3. In ButtonWidget.loadImageUp(String name) at line 1653 the code uses loadImageDownInt() instead of loadImageUpInt().

Fixed.

> 4. We don't support the full range of method signatures for ButtonWidget.loadImage*(). We are currently missing the "mixed" primitive and BDT cases. For example, loadImage*(String, integer, int, integer) and so forth. Do we mark any such mixed cases for wrapping at conversion time (the primitives get wrapped in this case and only in this mixed case)?

Sorry, I'm not aware of any case where the conversion would result in calls with mixed arguments. Can you provide more details on this?

5. I don't understand the implementation of ButtonWidget.loadImageInt(). All the offset and width/height data is used to update a local ButtonImageDefinition instance named imgDef.

The imgDef local variable is assigned to an object reference in the widget config, one of imgDef, imgUpDef, imgDownDef or imgInsDef.

7. I can't assess the validity of the changes to ButtonGuiImpl.height(), ButtonGuiImpl.width() and IMageGuiImpl.calculateDrawingParameters(). How have you confirmed correctness?

The tests I have run in uast so far confirm it is correct. However I haven't run all of them, this is WIP.

#14 - 11/03/2017 09:07 AM - Greg Shah

4. We don't support the full range of method signatures for ButtonWidget.loadImage*(). We are currently missing the "mixed" primitive and BDT cases. For example, loadImage*(String, integer, int, integer) and so forth. Do we mark any such mixed cases for wrapping at conversion time (the primitives get wrapped in this case and only in this mixed case)?

Sorry, I'm not aware of any case where the conversion would result in calls with mixed arguments. Can you provide more details on this?

```
def var txt as char init "some_file.jpg".
def var num as int  init 14.

...

button.load-image(txt, 300, num).
button2.load-image("a_different_file.png", num, 64).
```

The caller can be using literals that will normally emit as Java primitives. When mixing those together with BDT vars/fields/expressions will result in converted code that won't compile.

#15 - 11/03/2017 09:10 AM - Greg Shah

> 5. I don't understand the implementation of ButtonWidget.loadImageInt(). All the offset and width/height data is used to update a local ButtonImageDefinition instance named imgDef.

> The imgDef local variable is assigned to an object reference in the widget config, one of imgDef, imgUpDef, imgDownDef or imgInsDef.

On which line of code does that occur? I must be having a mental block, I don't see it.

#16 - 11/03/2017 09:14 AM - Hynek Cihlar

Greg Shah wrote:

> The caller can be using literals that will normally emit as Java primitives. When mixing those together with BDT vars/fields/expressions will result in converted code that won't compile.

Ok, I see it now. I assume the correct solution is to have the primitive types wrapped in BDTs during conversion. I will fix this.

#17 - 11/03/2017 09:15 AM - Hynek Cihlar

Greg Shah wrote:

> On which line of code does that occur? I must be having a mental block, I don't see it.

See the switch statement on line 2712.

#18 - 11/03/2017 10:52 AM - Eugenie Lyzenko

Greg Shah wrote:

> Code Review Task Branch 3332a Revision 11201

> Eugenie: Please review that code.

1. loadImageUp(String) uses loadImageDownInt(...) and this is incorrect.
2. There are the big changes in ButtonGuiImpl.(height()|width()), ImageGuiImpl.calculateDrawingParameters() and related calls. This is very dangerous area and can produce regressions if not to test. I can not tell for 100% sure if the changes are valid or not. The only way to find out it - to run all button and image related testcases and compare pixel by pixel the output. Note you will have to test ALL button types(regular, flat) with ALL image options(only UP; UP DOWN and INSENSITIVE). Then run all testcases for images. And at least compare with current trunk.

Actually I do not understand why to change so lot. The image with offset is the regular image but changed only in size. Why not to construct new image first and then use current code to process new image as having 0 offset? You will need to add only image re-generation code. All size calculation stuff will remain unchanged. Considering the a lot of manual testing this can make sense.

#19 - 11/03/2017 11:53 AM - Hynek Cihlar

Eugenie Lyzenko wrote:

> Actually I do not understand why to change so lot. The image with offset is the regular image but changed only in size. Why not to construct new image first and then use current code to process new image as having 0 offset?

Unfortunately there are several non-trivial cases, that require these changes. See the test cases in image/button_load-image.p image/image_load-image.p.

For illustration

- the BUTTON widget size is changed by the loaded image when the widget size was not explicitly assigned
- the BUTTON widget size is changed by the loaded image only when the loaded image is the UP image
- certain load size values affect the load offset and the effective load size, for example when load width is 0 or height is 0 (but not both)
- BUTTON widget size is changed only by the first loaded image
- the max load offset seems to be image size minus button insets, beyond that an error is shown and the image is not loaded, BUT the image size is set according to the actual image size if load size not given
- and there is more

#20 - 11/08/2017 09:45 AM - Hynek Cihlar

3332a revision 11206 resolves the review points except one exception mentioned below, fixes several widget sizing cases existing in trunk, added conversion and runtime support for DEFINE IMAGE ... IMAGE-SIZE ... . The revision passed GUI regression tests, ChUI testing is in progress.

The point 4 from note 12 mentions that there should be support for mixed Java primitive types and BDTs in the introduced loadImage methods. I didn't find any existing cases and the related implementation in the rules files for this, instead I found similar cases that always wrap the method arguments. 3332a revision 11206 always wraps the loadImage arguments, even when all of them are primitive types. If this is not desired please point to a test case or place in rules files that performs this mixed wrapping, if this doesn't exist yet I will implement it.

During testing I found the following issues that exist in trunk. I didn't attempt to fix these as these were out of scope of this issue.

- buttons do not layout correctly when LOAD-SIZE method is called with an image of different size from the default button size
- images loaded in a button widget are always centered in the button, I didn't find any case in native 4GL that would center the image.

#21 - 11/08/2017 09:51 AM - Hynek Cihlar

3332a revision 11206 is rebased againt trunk 11196. Please review.

#22 - 11/08/2017 10:14 AM - Greg Shah

Code Review Task Branch 3332a Revision 11206

1. I have no problem with forced wrapping for load-image*(). Don't worry about the mixed mode use case.

2. I would prefer to leave the primitive versions available in the runtime (e.g. ButtonWidget.loadImage(String)...) in case we implement some conversion improvements later. Can you put those back in?

3. What about int64 support?

#23 - 11/08/2017 10:33 AM - Hynek Cihlar

Greg Shah wrote:

> Code Review Task Branch 3332a Revision 11206
>
> 2. I would prefer to leave the primitive versions available in the runtime (e.g. ButtonWidget.loadImage(String)...) in case we implement some conversion improvements later. Can you put those back in?

Yes, no problem.

> 3. What about int64 support?

This should be covered. Although the native 4GL allows passing int64 values, anything above the max int value will cause a runtime error. Negative values always cause a runtime error.

#24 - 11/08/2017 11:14 AM - Greg Shah

> This should be covered. Although the native 4GL allows passing int64 values, anything above the max int value will cause a runtime error. Negative values always cause a runtime error.

My concern is that this converted code won't compile because integer is more specific than int64. Shouldn't the external API be defined as int64 instead of integer?

#25 - 11/08/2017 11:21 AM - Hynek Cihlar

Greg Shah wrote:

> My concern is that this converted code won't compile because integer is more specific than int64. Shouldn't the external API be defined as int64 instead of integer?

You are right, I forgot to change the parameter types. Will fix.

#26 - 11/08/2017 02:57 PM - Hynek Cihlar

3332a 11207 makes the loadImage methods accept int64 and added the overloads with Java primitive types. The revision passed GUI regression tests, ChUI is in progress.

No more changes are expected to go into the branch, assuming the review and tests pass. Please review.

#27 - 11/08/2017 03:27 PM - Greg Shah

Code Review Task Branch 3332a Revision 11207

I'm good with the code.

You will need to rebase (Eric is about to merge something to trunk). But you don't need to retest. If testing passes, you can rebase and merge to trunk.

#28 - 11/12/2017 07:11 AM - Hynek Cihlar

3332a rebased against latest trunk.

#29 - 11/13/2017 12:31 PM - Hynek Cihlar

3332a passed regression tests and have been merged to trunk as revision 11200. The task branch was archived.

#30 - 11/13/2017 12:34 PM - Hynek Cihlar

- % Done changed from 0 to 100

#31 - 11/13/2017 12:40 PM - Greg Shah

- Status changed from New to Closed

- Assignee set to Hynek Cihlar

#32 - 03/09/2018 10:47 AM - Greg Shah

- Project changed from Base Language to User Interface

#33 - 03/09/2018 10:49 AM - Greg Shah

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