Proposal: Workspace Ornaments for RiverWare 6.5

Phil Weinstein, CADSWES

This document describes the development of a new workspace text and image "ornament" feature in RiverWare. Initially, it will be supported in the Simulation workspace.

Document Status:

- 10-22-2013: Ready for Review

(1) Requirements Overview

Users will be able to add multiple-line text blocks and images to specific locations within the RiverWare Simulation Workspace. The following requirements are for the initial version of this feature; future enhancements are possible.

(1.1) General Ornament Requirements:

1. Each ornament has either **text** or an **image**.
2. Ornaments are *anchored* at a point within the Simulation Workspace. The anchor point is associated with a designated corner, edge, or center of the ornament. That point remains fixed when the size of the ornament changes, e.g. by changing the ornament's font size.
3. Ornaments are automatically scaled with the workspace -- as are, for example, simulation object icons.
4. All ornaments can be shown or hidden with a single toggle setting on the workspace.
5. Creating, editing, and moving ornaments is *disabled*:
    a. when the workspace icon lock is on.
    b. in RiverWare Viewer mode or in a Baseline model (for Scenario users).
6. When enabled, ornaments can be positioned by the user by dragging them with the mouse.
7. Ornaments, if shown, will be included in workspace print outputs and image exports.

(1.2) Text Ornament Requirements:

1. Text Ornaments can be **"minimized".** In the minimized state, only a small "note" icon is shown, and the ornaments text can be momentarily displayed as a tool tip (by hovering the mouse pointer over the note icon). Along with the ability to hide all ornaments (mentioned above), it is also possible to minimize or "restore" all text ornaments.
2. Custom text ornament settings:
    a. **Font:** Each text ornament can be shown with either the workspace's current "canvas font", or a **custom font** configured using the standard Qt Font Dialog. This includes selections of (a) font (face), (b) style (normal / bold / italics), and (c) size.
    b. **Color:** Each text ornament can be shown with either the workspace's current text color, or a custom text color picked with the standard color chooser dialog.

    c. Multiple-line text can be either **left justified, right justified,** or **centered.**
    d. **Drawn Box:** A border frame may optionally be drawn around the text.
3. Rich Text is supported to the extent that that is natively provided by the Qt QTextEdit widget. However, automatic text wrapping is turned off -- the user has direct control of where text lines break.
4. Text ornaments are edited in a separate dialog (not "inline" directly on the workspace), with controls for the options described above. The text ornament editor is shown by double-clicking an ornament. Ornament editors for multiple ornaments can simultaneously be shown.

(1.3) Image Ornament Requirements:

1. An Image Ornament's image is defined as a reference to an external image file of a format directly supported by Qt (e.g. PNG, JPG, GIF). The file paths to these images are saved in the model file *relative to* the model file. So, adding an image ornament to a new model is disabled until the workspace is saved to a model file. The recommended use is locating the Image Ornaments' image files either within the same directory as the model file or in a subdirectory under the model's directory, and these should be moved or copied as a group.

    - **ALTERNATIVE:** In may be reasonable to serialize and save Image Ornament images *within the RiverWare model file.* This could be provided as an option. (Note: a standard base-64 encoding would be used; this is about 75% efficient in general. However, to avoid an unrelated inefficiency related to images having transparency -- which can be detected at run-time -- some special provisions would be needed).

2. The user can **rescale** the image so that its size on the workspace is different from the original image's natural size. In the initial implementation, rescaling is implement with numeric entries (rather than dragging the image rectangle). It is also easy to show the image in its natural size by turning off the rescaling options.

(2) Functional Description

(2.1) Creating a New Text or Image Ornament

In models where model editing is enabled (see Requirements) and when workspace icons are not locked and the Simulation Workspace is shown, controls for adding the two types of workspace ornaments will be available in:

- the Object Palette
- the Simulation Workspace context menu
    - Add Text Ornament ...
    - Add Image Ornament ...

These actions will create a "placeholder" ornament on the workspace. In the Object Palette, the two new controls can be dragged to the workspace. When the context menu is used, the new ornament is placed at the clicked position. The placeholder ornament can be moved by dragging with the mouse pointer. Double clicking the placeholder ornament opens up the editor dialog for the ornament type (text or image).

Tentative: When creating a new ornament, its editor dialog is automatically shown. Canceling that dialog without first applying any changes will abort the ornament creation (so will delete the placeholder ornament).

(2.2) Operations on Selected Text and Image Ornaments

Ornaments can be individually selected by clicking them, their selection state can be toggled by control-clicking them, and they can be "rubber-band" selected with a rectangle drag. Basically, the currently implemented simulation object selection set will be augmented to include ornaments.

A selected ornament will have an altered "selection" appearance.

Operations on selected ornaments include:

- movement, by dragging.
- context menu: deletion
- context menu: "minimize" and "restore"

Deletions will be confirmed with a confirmation dialog box.

(2.3) Editing Text Ornaments

A text ornament can be edited by double-clicking. This brings up a "Text Ornament Editor" dialog box. This dialog box includes:

- A multiple-line text editor (a QTextEdit widget).
- The following property editors, generally arranged as individual rows:
    - Checkbox: Font (override) / button showing the Qt Font Dialog
    - Checkbox: Background Color (override) / button showing the color chooser dialog
    - Checkbox: Text Color (override) / button showing the color chooser dialog
    - Checkbox: Box Frame
    - ComboBox: Alignment, nine options, with these icons and text:
        -  top left
        -  top center
        -  top right
        -  left
        -  center
        -  right

        -  bottom left
        -  bottom center
        -  bottom right
- Sample text property display, demonstrating the active font, background and text colors. Note that the font and color choices are optional overrides of the workspace's canvas configuration; overrides are specified by checking ON the checkbox.
- Bottom buttons: OK, Apply and Close/Cancel. (The latter button shows "Cancel" only if un-applied changes have been made).

(2.4) Editing Image Ornaments

An image ornament can be edited by double-clicking. This brings up an "Image Ornament Editor" dialog box. The first time a new Image Ornament is edited (so doesn't yet have an associated image file), IF the model hasn't yet been saved (i.e. isn't yet associated with a model file), the following message is displayed in the dialog box, and the property controls (widgets) in the dialog box are disabled.

- *The model file must be saved before an image file can be associated with this image ornament.*

The image controls consist of:

- An image file path entry widget and a "[...]" button which brings up an image file chooser. Note that the path is displayed as relative to the model file's current path, but the "absolute" version of that path is used to initialize the file chooser. Environment variables in pathnames are _not_ supported here.
- An image thumbnail display, with image width and height indications, in pixels. (The aspect ratio of the referenced image will not be correctly depicted; the thumbnail display will be a square box).
- Checkbox: [X] Rescale:
    - Checkbox: [X] Width / integer entry widget
    - Checkbox: [X] Height / integer entry widget
      ... Note: if only one of these two checkbox's are checked, the image's original aspect ratio is retained. Both may be checked, but checking off either one forces the other width-or-height checkbox on.
- ComboBox: Alignment, nine options. See the prior section.
- Bottom buttons: OK, Apply and Close/Cancel. (The latter button shows "Cancel" only if un-applied changes have been made).

(3) Development Tasks

1. Proposal and Development Planning (this document).
2. Data Model, with persistence. (Supporting only image file references; not serialized images).
3. Ornament Creation:
    a. Object Palette enhancements

    b. Workspace context menu
4. Text Ornament with Normal Rendering
5. New Dialog: Text Ornament Editor
6. Image Ornament with Normal Rendering
7. New Dialog: Image Ornament Editor
8. Selection Operations and QGraphicsItem Rendering Variations
9. Workspace Operations on Image Ornaments
10. Print and Image Export Testing
11. Correctness, Completeness and Usability Testing
12. Developed Feature Documentation
13. Post-Development Review Changes

(3.1) Proposal and Development Planning (this document).

(3.2) Data Model, with persistence. (Supporting only image file references; not serialized images).

An "Ornament" class will be defined in the RiverWare Sim library. It holds all of the persistent data for one text or image ornament. The set of Ornaments defined in the model will be managed by an OrnamentMgr (manager) class, a single member of the SimWorkspace. Ornament supports an XML-based serialization, which will be embedded in the Tcl-encoded model file serialization.

Ornament and OrnamentMgr support callback-based model change notifications. Initially defined callbacks include:

- ORNAMENT_DELETED
- ORNAMENT_CHANGED
- ORNAMENT_MOVED_SIM (repositioning within the Simulation workspace).

(3.3) Ornament Creation:

1. Object Palette enhancements
2. Workspace context menu

See the Functional Description, section (2.1). The Object Palette supports drag and drop operation. (Bonus observation: the Object Palette in one running RiverWare instance actually works for other running instances!).

Development for the creation features will touch on these code areas -- some of which will require at least partial completion of the subsequent "rendering" tasks.

- Object Palette, add buttons
- Object Palette, drag and drop operation
- Workspace context menu, add QActions, condition and handle
- Ornament Manager: Ornament instantiation
- Ornament: initial state

- WorkspaceGfxScene: registration and handling of callbacks from
- WorkspaceGfxScene: minimal instantiation of dummy QGraphicsItems

(3.4) Text Ornament with Normal Rendering

Text Ornaments (and Image Ornaments) are implemented as custom Qt4 QGraphicsItems within a QGraphicsScene subclass (and displayed within a QGraphicsView subclass). These will be implemented in parallel with the QGraphicsItems for simulation objects. The implementation is similar to that of the SimObjGfxItem class (i.e. the part of this class' implementation for the simulation workspace).

Subtasks include:

- Definition of a **OrnamentGfxItem** (a RwGraphicsItem subclass). This class will also implement the Image Ornament, see below.
- Composition of a child QGraphicsTextItem.
- Computation of the OrnamentGfxItem's bounding rectangle and shape (these are similar, but play different roles in the appearance and behavior of the graphics item).
- Application of the configured text properties to this child item. Some of these properties (font, color) may come from the workspace canvas or from the custom configuration of the particular Text Ornament.
- Once basic editing features are implemented in the Text Ornament Editor dialog (next item), Text Ornaments will be made to update their appearance. This will make use of an existing central timer-based mechanism to condense a sequence of property change notifications into a single update operation.

Note: Support for the "minimized" state will be implemented as a variation of the support for Image Ornaments (within the same Ornament instance). See section 3.6.

(3.5) New Dialog: Text Ornament Editor

The Text Ornament Editor dialog edits a single Ornament instance. (The initial implementation will distinguish Text Ornaments from Image Ornaments at this application level. In the data model, both are represented as just an Ornament instance with only "text" properties. In the future, a given Ornament instance may have both text and an image).

See also Section 2.3: Editing Text Ornaments.

The unapplied edit states will be represented within the dialog widgets. "Applying" edits causes those states to be set on the underlying Ornament instance.

Subtasks include:

- Widget layout using Qt Designer. (Text property editor widgets will be defined within a "panel" which can, in the future, be deployed in a single dialog supporting both text and an image).

- Setting the stated of the edit controls (widgets) from an Ornament instance.
- Saving the state of the edit controls (widgets) to an Ornament instance.
- Ornament equality operator for conditioning of the "Apply" and "Cancel/Close" states.
- Sample text property display widget, demonstrating the effects of the current font and color choices.
- Registration for, and handling of the Ornament's ORNAMENT_DELETED callback. (This will just close the dialog box).
- Semi-persistent editor dialog size and location implementation (just within a single RiverWare session).

(3.6) Image Ornament with Normal Rendering

Image Ornament rendering will be implemented within the OrnamentGfxItem class -- same as for Text Ornaments. This class will conditionally manage a child QGraphicsPixmapItem, in parallel with its child QGraphicsTextItem for managing the Ornament's text. The original, unscaled image will be maintained as a QPixmap within the OrnamentGfxItem. That will either be directly shared with the child QGraphicsPixmapItem -- OR if rescaling is configured, a rescaled version of that QPixmap will be set on that child.

Subtasks include:

- Composition of a child QGraphicsTextItem (mentioned above).
- Enhancement to the computation of the OrnamentGfxItem's bounding rectangle and shape.
- Application of the configured image properties to this child item -- basically preparation of a QPixmap.
- ALSO: "Note Icon" support for the minimized state of **Text Ornaments.**

(3.7) New Dialog: Image Ornament Editor

The Image Ornament Editor dialog edits the Image-related properties of an Ornament instance. It shares many implementation provisions with the Text Ornament Editor dialog (see two-items prior).

See also Section 2.4: Editing Image Ornaments.

Subtasks specific to the Image Ornament Editor dialog include:

- Display of the *relative* image file path in a line editor.
- Use of a file chooser dialog, initialized to the *absolute* image file path.
- Conversion of the file path picked from the file chooser to a path relative to the current model file's path.
- Rescale controls logic.

(3.8) Selection Operations and QGraphicsItem Rendering Variations

The WorkspaceSelection class includes a representation of the selected simulation objects in the workspace. This will be enhanced to also represent the set of selected Ornaments. Some operations on that selected SimObjs will be enhanced to also operate on the selected Ornaments. See Section 2.2: Operations on Selected Text and Image Ornaments.

An alteration in the appearance of OrnamentGfxItems will be implemented for the "selected" state. For Text Ornaments, the foreground and background colors will be reversed. For Image Ornaments (and "minimized" Text Ornaments) a heavy border will be shown.

A change in the selection state of any OrnamentGfxItems will result in the repainting of those items, but only after a brief timer-delay to collapse many selection state change notifications into a single update operation.

(3.9) Workspace Operations on Image Ornaments

The following operations are implemented at the workspace level:

- Toggle: Show/Hide all Ornaments
- Minimize all Text Ornaments
- Restore (un-minimize) all Text Ornaments

These changes are effected by changing properties within the data model (existing Ornament instances). That results in change notifications (ORNAMENT_CHANGED callbacks) to the corresponding OrnamentGfxItems, causing them to update. (This will be mediated by an existing timer-based mechanism to reduce the number of GUI updates).

(3.10) Print and Image Export Testing

Code changes are not anticipated for correct Print and Image Export operations. But this will need to be tested, and any problems found will need to be addressed.

(3.11) Correctness, Completeness and Usability Testing

General functional and "Gestalt" testing.

(3.12) Developed Feature Documentation

Preparation of a feature document with screenshots.

(3.13) Post-Development Review Changes

Review by others may result in requests for changes to this feature. Review notes, responses, change decisions and development status will be informally documented on an internal webpage. The feature document will be updated to reflect any changes made.

(4) Development Estimates

Phil's development estimates -- 10-22-2013

Task	Days	Description
1	1.25	Proposal and Development Planning (this document).
2	1.0	Data Model, with persistence.
3	1.25	Ornament Creation: Palette / Context Menu
4	2.0	Text Ornament with Normal Rendering
5	1.5	New Dialog: Text Ornament Editor
6	1.0	Image Ornament with Normal Rendering
7	1.25	New Dialog: Image Ornament Editor
8	1.0	Selection Operations, Rendering Variations
9	1.0	Workspace Operations on Image Ornaments
10	0.25	Print and Image Export Testing
11	0.75	Correctness, Completeness and Usability Testing
12	0.75	Developed Feature Documentation
13	2.0	Post-Development Review Changes
	15.0	TOTAL estimate (days)

--- (end) ---