SceneComposer 3.5

Property Editing Improvements

In CGI Studio SceneComposer for some properties the user can choose from a defined set of values e.g. the type of layout he wants to use (GridLayout, StackLayout...) or he can define the values for horizontal and vertical alignment. CGI Studio 3.5 offers intuitive icons to the user, so he can easily decide, which option he wants to take. Tooltips when hovering these icons make these icons even more user friendly.

Several items have properties where more than one value can be entered:

- position properties require a value for the x- and a value for the y-coordinate
- size properties require a value for the x- and a value for the y-coordinate
- margin properties even require 4 values (for left, top, right and bottom)

With CGI Studio 3.5 the input fields for these values are positioned next to each other in multiple columns to save space and reduce scrolling effort.
At the same time, the appearance of the properties has been very much improved and looks a lot tighter.

Nine-Patch

When scaling images, the loss of too much quality shall be prevented. To optimize the scaling process, with CGI Studio 3.5 each image allows the configuration as Nine-Patch. By defining four stretchable areas for left, top, right and bottom, the image will be divided into 9 differently scaled patches (thus Nine-Patch) as seen in the image below. The Nine-Patch configuration is considered at any kind of scale operation - no matter if it is a manual scale operation or during layout calculation.

See also:

- 9-patch image in SceneComposer User Manual
- Tutorial 14: 9-patch image in SceneComposer User Manual

Pivot Hint

In addition to optimized resizing of images using the Nine-Patch approach described above, the user can define a Pivot Hint for each image. Additionally the Pivot Hint is used by Behaviours to allow them to set a Pivot Offset to an image that has been set.

State Machine

Since this version on CGI Studio introduces support for a new integrated solution to model behaviour based on the "state machine" concept. By using traditional design means, multiple state machines can be defined. A state machine contains states and transitions and allows defining model changes into the graphical model when certain conditions are met. The state machine links work-flow logic to existing CGI Studio graphical items and building blocks such as behaviours and controls.

See also:

- State Machine in SceneComposer User Manual

Controls & Behaviours

There is a bunch of new predefined Controls in SceneComposer. They cover many elements necessary in a user interface. Among those Controls are a progress-bar, gauge, telltale, list, button, slider and many more. Controls quicken the process of HMI development massively. The HMI elements can interact with each other and because of the underlying Behaviors no single line of code has to be written for many use cases.

To make these Controls work and to be able to define own Controls, the list of Behaviors has increased significantly. There are many different Behaviors reaching from Event Handling to Value Processing.

See also:

- Controls & Behaviours in SceneComposer User Manual
- Getting Started with Controls in SceneComposer User Manual

Building Blocks for Behaviours

Behaviours can be nested and in this way combined to a reusable behaviour building block that is part of the solution and appears in the Solution Explorer. These behaviour blocks offer the possibility to configure, which of the properties of this behaviour block will be published and are

therefore editable at the time the behaviour block is instantiated and used in the SceneComposer scene.

When many behaviors are nested, the hierarchy of the behaviors is shown trough the visualization of the behaviors integral map (= "tree") structure. The tree structure of nested behaviors is now visualized properly in the Scene Tree View. This makes it easier to see, which behaviors belong together.

See also:

- Behavior Building Blocks in SceneComposer User Manual

EventHandler Tab

Every control needs an EventHandler in order to handle user events and take appropriate actions. With CGI Studio 3.5 an EventHandler Tab has been introduced in the Properties View for all Controls. In this EventHandler Tab the user can choose the event he wants to react on from a drop-down menu. Furthermore, conditions and actions can also be configured with drop-down menus in this tab. Having only a selection of suitable and applicable conditions available at configuration time after selecting an event makes this new EventHandler Tab a welcome and user friendly feature.

Drop Shadow Effect

A drop shadow text effect was implemented in order to permit - in 2D scenes - the creation of visual effects similar to those available in other graphic tools. By using any of its properties - like "Distance", "Shadow Scale", "Blur Filter", "Light Angle" - the user has the possibility to set the shadow effect on any text as needed.

Monotype

Font engine iType feature has been integrated. iType is a font engine which main purpose is fast and qualitative rasterization of fonts.

Monotype/iType has a special porting API with which target and OS specifics can be overridden. CGI Studio provides a single solution for all targets. For this, the framework links the API to the device and operating system layers. It is also possible to use an own customized port instead of the CGI Studio solution.

See also:

- Integration of Monotype in SceneComposer User Manual

Candera Scripting

This versions adds support for script events. Script events can be sent and received by both Lua scripts and native code. At the same time, this versions adds support for referencing Candera objects in scripts. These object references are aware about the lifetime of the objects they reference without interfering with them. They can be passed from native code to scripts and vice versa, as well as dereferenced to call object specific methods in Lua.

New Data Types: Optional and Variant

Especially for the implementation of the new Behaviors new data types have been introduced.

- Optional: The Optional data-type allows the use of uninitialized objects. Therefore it provides methods to check whether the object has been initialized or not.
- Variant: The Variant data-type allows to store one value in different types. The type can be an integer, unsigned integer, floating point number or boolean.

Revision #6
Created 2023-03-02 01:02:36 UTC by Kanai Tomoaki
Updated 2025-08-14 00:59:24 UTC by Tsuyoshi.Kato