CGI Studio 3.5

Release Date: January 2018

- General 3.5
- Candera 3.5
  - Candera/FeatStd CMake Build System Changes
  - Candera Base
  - Candera System
  - Candera AssetLoader
  - Textengine
  - Candera Behaviors
  - Candera Scripting
- Courier 3.5
- FeatStd 3.5
  - FeatStd Util
- SceneComposer 3.5

General 3.5

CGI1-21755 - CMake: Generated projects cause too long path names on Windows

Sometimes the path name length of the project files, that are generated by CMake, are too long for the Windows file system, depending on build folder path length, device, OS, CPU, ... Suffixes for the project names have been removed (in FeatStd and Courier CMake files).

Candera 3.5

To keep backwards compatibility for any interface change in the public API the old interface has been kept and marked as deprecated. The deprecated interfaces will be removed in future versions. Compiler warnings will be generated when using a deprecated interface through the usage of Candera Macro CANDERA_DEPRECATED_3_4_0.

Candera/FeatStd CMake Build System Changes

Clipping Enabled by Default

The flag CANDERA_LAYOUT_CLIPPING_ENABLED was disabled by default. From now on the flag CANDERA_LAYOUT_CLIPPING_ENABLED is enabled by default. Reason: for proper layout in terms of HMI controls the clipping of layout areas is mandatory. HMI controls are a major focus of CGI Studio 3.5.

Candera Base

Cached Layout

A new (dynamic) property has been added to the Layouter class: Cached Layout It specifies if the layout of the node should be performed only once until a invalidation inside the subtree occurs. A cached layout will be only laid out again as soon as the layout of an inner node is marked as invalid. See Layouter::SetCached and Layouter::IsCached.

> Widgets or Behaviors shall not invalidate the scene containing a laid out node anymore, but the according node! The default value of the new property is _False_.

Script Parameter Animation Property Setter

A new animation property setter has been added that can animate a numeric variable defined in a script.

Candera System

Property of Type Variant

Support for widget / behavior properties of type FeatStd::Variant has been added. To use datatype serialization of type Variant it is necessary to add the following include statement:

```
#include <Candera/System/Mathematics/MathDataTypes.h>
```

Candera AssetLoader

Property of Type Optional

Support for widget / behavior properties of type FeatStd::Optional has been added. To use datatype serialization of type Optional it is necessary to add the following include statement:

```
#include <CanderaAssetLoader/AssetLoaderBase/OptionalDataType.h>
```

Textengine

Support of Monotype's font engine: iType

Candera supports the font engine 'iType' of 'Monotype' now. In this release a basic integration of iType was done. With the provided iType integration two different shaper are supported: Complex Script and NoShaper. Monotypes own shaper will be part of next releases.

The current integration of Monotype uses its default settings. Textrendering supports several glyph cache/render types: Bitmap, Surface, Glyph, GlyphCache and GlyphAtlas. Two of these types are supported by iType now. These are Bitmap and GlyphAtlas. The others may work but with cache functionality limitations. Therefore, it is not recommended to use one of the other caches. In general: best performances are reached when GlyphAtlas (GPU solution) or Bitmap (CPU solution) is being used.

Due to different algorithms to interpret font data, it is expected to see small differences when comparing monotype and freetype content. E.g.:

- Gray values (transition between glyph and background) differ in its intensity.
- Applied glyph kerning may shift the text by 1 px.

One of the great benefits of iType is its performance when it comes to rasterization of text. This is already visible in the first integration steps and provides a small forecast to the next steps.

Candera Behaviors

Event Handler Behaviors

Classes have been added to support Event Handling. The TriggerBehavior references a ConditionBehavior and an array of ActionBehavior. ConditionBehavior and ActionBehavior are base classes. Any input event is processed by a TriggerBehavior, which evaluates a condition (ConditionBehavior). If the condition evaluates to true, then the TriggerBehavior invokes one or more ActionBehaviors. A filter on event type can be achieved by a specialized ConditionBehavior class for such an event.

Value Processing Behaviors

The ValueBehavior provides a Variant data type which is bindable. A value change results in a processing of that value by sending a ProcessValueEvent to a referenced ProcessValueBehavior.

The ProcessValueBehavior base class has been added. Derived classes can be chained to realize a data flow processing from input to an according (visual) output. Example: from a changed "speed" value to a mapped rotation value and further to an animated rotation of a needle node.

Candera Scripting

Script Events

This versions adds support for script events. Script events can be sent and received by both Lua scripts and native code.

Candera Object References in Scripts

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.

Lua Animation Functions

This versions adds support for animations in Lua scripts.

Courier 3.5

IPC dynamic Component Count and Process Count

Due to a new concept of IPC messages more processes and components are allowed. Actually the size of a message is dynamic by compile time now. Note the following points:

1. Changed ComponentType::Invalid to COURIER_COMPONENT_MAX_COUNT.
2. All applications need the same COURIER_COMPONENT_MAX_COUNT and COURIER_PROCESS_MAX_COUNT to work properly.

FeatStd 3.5

FeatStd Util

Variant Type

A Variant data type has been introduced. The variant can hold a bool, UInt32, Int32 or Float value.

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.
