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.
