CGI Studio 3.12

Release Date: July 2023

General 3.12

General Changes

This page describes general updates that were made to CGI Studio for version 3.12.

Documentation Updates

- Dynamic documentation
- New look & feel incl. images and multimedia contents
- Support for multiple language

Sample Solutions New/Updates

- 2D Basic
- 3D Basic
- 3D Getting Started
- MenuScreens
- Industrial

Player New Feature

- Rotation and zoom functions in the Player Panel

Speed up of compile time

Via a project wide clean-up, we could speed up the overall compile time by about one third (33%).

Candera 3.12

VectorialRenderData and respective interfaces were removed

VectorialRenderData and its respective interfaces in RenderDevice2D were removed because they were not used.

GlMaskEffect Scale behavior changed

When the size of the mask image was bigger than the size of the source image, the mask image was scaled down to the source image size. This behavior was not intended and is regarded as a bug that is now fixed. With the GlMaskEffect the same scaling effect can be achieved, just use a dedicated mask node and apply there the appropriate scale to the node transformation.

ProcessValueEvent sender changed from ProcessValueBehavior to Behavior

To enable a more generic processing of ProcessValueEvent, the ProcessValueEvent sender has been changed from ProcessValueBehavior to Behavior. This also allows other behaviors to process values with a ProcessValueBehavior chain. Unfortunately, for the purposes of backward compatibility the interface can't be changed. Furthermore, although the creation of ProcessValueEvent is backward compatible, the ProcessValueEvent::GetSender is not. All usages of that interface have to be changed from ProcessValueBehavior const& to Behavior const&. The ProcessValueBehavior does not expose any additional interfaces. As such, the Dynamic_Cast to a dedicated derived class is the only possible option for the sender. This is still fully backward compatible. Accordingly, the type change is the only required change.

AssetLoaderPostponedTasks argument have been added to the following interfaces:

virtual bool Candera::MetaInfo::PropertyMetaInfo::Set(HostType* object, const Char *value, Candera::AssetLoaderPostponedTasks* assetLoaderPostponedTasks)
AssetProvider::CreateCompositeGroupByAssetId and AssetProvider::CreateCompositeGroup2DByAssetId. When there is an override of the interface, you must also provide the argument and delegate it as necessary. In the case that there is a missing argument when calling those interfaces, you can safely provide a null pointer or provide a local (function stack) instance off AssetLoaderPostponedTasks. The AssetLoaderPostponedTasks instance will collect all initialization steps that have to be postponed and process them in the destructor of AssetLoaderPostponedTasks (scoped mechanism). When disposing the object involved in the call or when disposing the resulting objects, you have to call the Clear method of th

AssetLoaderPostponedTasks instance.

Several clipping related issues have been fixed.

RenderNode::GetComputedBoundingRectangle with ignoreInvisible set to true held the clipping rectangle in the wrong coordinate system (the clipping rectangle must be given in the parent space; the bounding rectangle must be in the local space; the transformation of the clipping rectangle to the local space was missing and has been fixed). Any custom workaround related to this has to be checked and adapted. TextNode2D::IsLayouterAttached considered every layout as a text-related layout (resulting in endless recursion with stack overflow in some use cases). Only layout implementations derived from TextNode2DLayouter are valid text-related layout implementations, and these will correctly set up the expected corresponding values in order to obtain the corresponding bounding rectangle. Any incorrect custom text-related layout implementation has to be adapted. BoundingRectTraverser delivered the bounding rectangle in the wrong space when the processed node (argument of Traverse) and the root node (argument of constructor) were the same. Any custom workaround related to this has to be checked and adapted. Layouter::Arrange provided incorrect arrange/clipping area values for OnArrange/OnClipping. The arrange area contained values from the previous layout run (the positioning is performed by the generic layout code after determining the actual size by the OnArrange call and therefore anyway not allowed to be performed by the derived implementation; values from the previous run caused incorrect behavior in layout implementations that do not consider this fact). The clipping area did not consider any outcome of the generic arrange processing and therefore contained incorrect values (for example, infinite size for automatic size configuration and incorrect alignment results). The arrange/clipping area has been fixed. Any custom workaround related to custom OnClipping implementation has to be checked and adapted. Layouter::OnClipping was setting the local space clipping area instead of the parent space clipping rectangle on render nodes (the clipping expects the clipping rectangle in parent space). This has been fixed and in addition, the layout clipping area is now set for group nodes that have to be clipped when nested clipping is required (e.g. drawer control in list control).

OverlayLayouter::OnArrange provided an actual size that was too small (this mainly has an impact in right-to-left-layout use cases). This has been fixed to allow consideration for the arrange area as well as, when clipping is required, sizes that are too large (which is required for correct alignment in combination with clipping). Any custom workaround related to this has to be checked and adapted. ClippingSetterTraverser was not applying the correct clipping rectangles (due to some of the above issues, the wrong space was used and clipping areas in nested clipping use cases had not been considered). This behavior has been fixed to allow consideration for the provided local clipping area, all nested available local clipping areas, and the correct intersections of all of these into a parent space clipping rectangle for each render node. Any custom workaround related to ClippingSetterTraverser has to be checked and adapted.

Changed order of Courier Component::PostProcess

The execution order of Component::Execute() and Component::PostProcess() has changed.

Interfaces deprecated by CGI 3.7.0 and 3.8.0 have been removed

All interfaces that have been marked with CANDERA_DEPRECATED_3_7_0 or CANDERA_DEPRECATED_3_8_0 have been removed.

Deprecated TextBrush effects removed

The deprecated 2D effects with TextBrush have been removed from builds. A newly introduced flag CANDERA_DEPRECATED_TEXTBRUSH_ENABLED has been turned OFF by default. For text, the TextNode2D node type shall be used. The new flag reduces code size. If backward compatibility is required, then this flag can be set to true (e.g. for porting purposes).

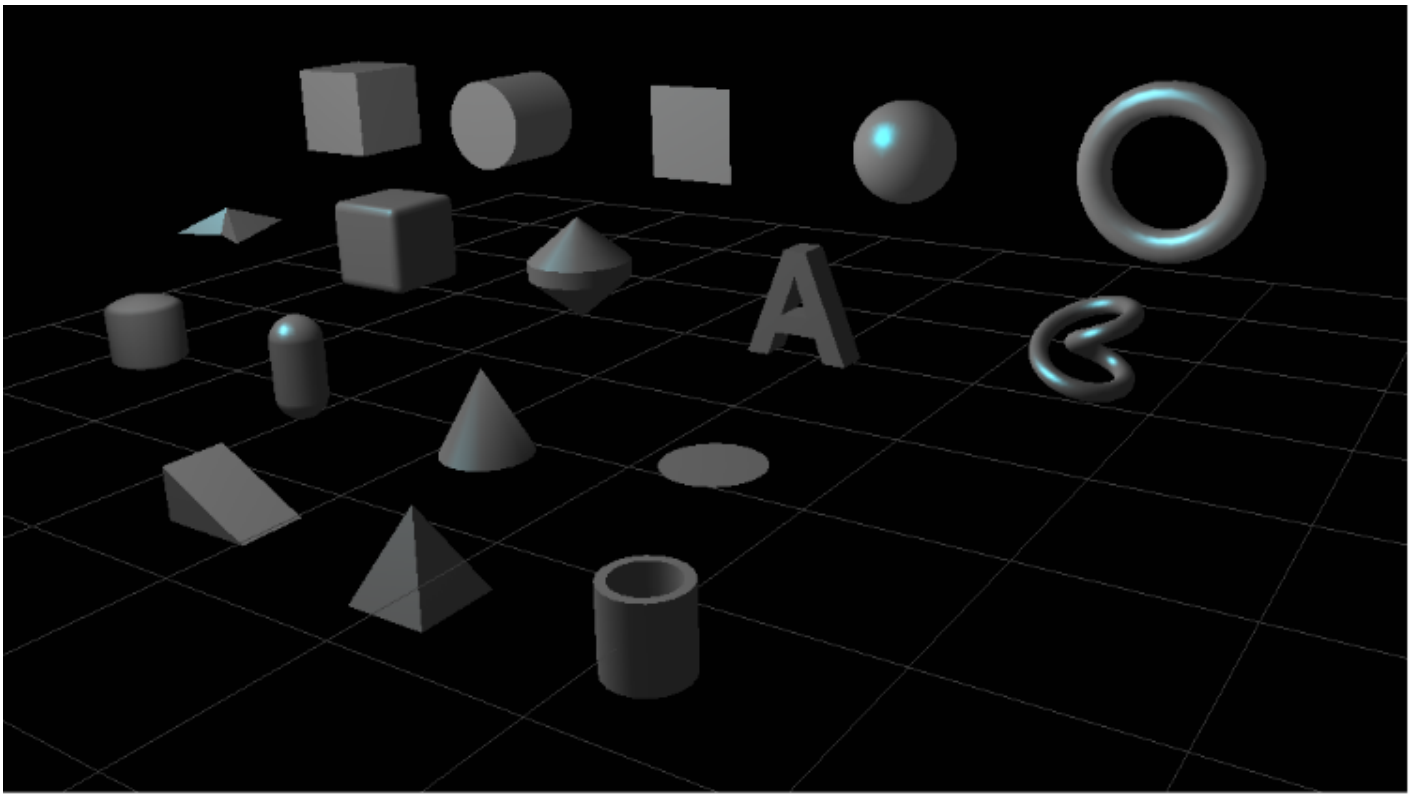
> The flag CANDERA_DEPRECATED_TEXTBRUSH_ENABLED and all related source code (TextBrush, related combined effects, caches, and so on) will be removed in one of the next releases!

Controls 3.12

New Controls:

3D Shapes Controls

- Arrow
- Capsule
- Cone
- Disc
- OilTank
- Pyramid
- RoundedCube
- Spindle
- Text3D
- Tube
- TwistRing
- Wedge



Behaviors 3.12

New Behaviors:

- Set External Bitmap: Retrieves a bitmap and sets the bitmap of the associated Render Node when its action is triggered.
- Check External Resource Change: Reacts on ExternalResourceChangedEvents, reloading the resource and invoking an update if ExternalResourceMangerId and ExternalResourceId match.
- External Bitmap Provider: Loads a bitmap at scene start from an External Resource Manager using ExternalResourceManagerId and ExternalResourceId.

Updating Behaviors:

- In the CgiStudioControl::InterpolateProcessValueBehavior the default value of property UnitsPerSecond is now 1.0 (was 0.0).
- In the CgiStudioControl::ArithmeticProcessValueBehavior there is now a new option "Absolute".
- The behavior CgiStudioControl::SizeProcessValueBehavior has been renamed to CgiStudioControl::GroupResizeProcessValueBehavior. The behavior CgiStudioControl::NodeSizeProcessValueBehavior has been added.
- In CgiStudioControl::InterpolateProcessValueBehavior the implementation of the "SpeedBased" mode has been fixed and for that a new property "InterpolationTime" has been introduced.
- In CgiStudioControl::ScrollableEventActionBehavior a property "TargetNode" has been added to be consistent with other behaviors.
- In CgiStudioControl::TrackPositionProcessValueBehavior a property "TargetNode" has been added. Use this node to control the scrollable/trackable area of the knob.
- In CgiStudioControl::ListBehavior a property "WrapAround" has been added enabling list wrapping.
- CgiStudioControl::IntervalSwitchBehavior has been changed to have a defined behavior in edge cases (Ontime is 0 and/or OffTime is 0).

FeatStd 3.12

String functions without bounds deprecated

All FeatStd::Internal::String related functions without bounds have been deprecated. Candera::StringPlatform is an alias for FeatStd::Internal::String.

Affected functions are:

- FeatStd::Internal::String::CompareStrings, use function with same name and additional parameter n instead.
- FeatStd::Internal::String::Copy, use function with same name and additional parameter n instead.
- FeatStd::Internal::String::Length, use function FeatStd::Internal::String::NLength instead.
- FeatStd::Internal::AsciiEncoding::CodePointCount, use function with same name and additional parameter n instead.
- FeatStd::Internal::AsciiEncoding::CharCount, use function FeatStd::Internal::AsciiEncoding::NCharCount instead.
- FeatStd::Internal::AsciiEncoding::Compare, use function FeatStd::Internal::AsciiEncoding::NCompare instead.
- FeatStd::Internal::AsciiEncoding::Match, use function with same name and additional parameter codePointCount instead.
- FeatStd::Internal::Utf8Encoding::CodePointCount, use function with same name and additional parameter n instead.
- FeatStd::Internal::Utf8Encoding::CharCount, use function FeatStd::Internal::Utf8Encoding::NCharCount instead.
- FeatStd::Internal::Utf8Encoding::Compare, use function FeatStd::Internal::Utf8Encoding::NCompare instead.
- FeatStd::Internal::Utf8Encoding::Match, use function with same name and additional parameter codePointCount instead.

In case that the upper limit is unknown, the constant FeatStd::Internal::String::BoundedStringLength can be used. This constant has a configured value of 4096.

TChar and Ucs2Char deprecated

The types TChar and Ucs2Char have been deprecated. Use Char instead. CGI Studio since long time only supports UTF-8 text encoding. Thus it was decided to remove the outdated option to also support wide characters.

TextEncoding methods Ucs4 and NUcs4 deprecated

The methods TextEncoding::Ucs4 and TextEncoding::NUcs4 return an UTF-32 encoded codepoint. The term "UCS4" is outdated. Therefore these methods have been deprecated. Use TextEncoding::GetUtf32 instead.

Fixed Point Option removed

There was the option to use fixed point arithmetic instead of floating point. However, this option was not supported anymore since long time. Therefore this option was now removed. The flags FEATSTD_FIXED_POINT_ARITHMETIC and FEATSTD_FIXED_POINT_FRACTIONAL_BITS have been removed.

Extended Stream API in FeatStd

The interface of FeatStd::Internal::IO::Stream has been extended to allow for Seeking (random-access). Multiple stream implementations have been added (FeatStd::Internal::IO::FileStream, FeatStd::Internal::IO::MemoryStream, FeatStd::Internal::IO::ReadOnlyMemoryStream) and several adapters (FeatStd::Internal::IO::TextReader, FeatStd::Internal::IO::TextWriter).

Migration Guide

Description	V3.11.0	V3.12.0
Unbound String function deprecated	Int32 CompareStrings(const Char* lhs, const Char* rhs)	Int32 CompareStrings(const Char* lhs, const Char* rhs, SizeType n)
Unbound String function deprecated	void Copy(Char *to, const Char *from)	void Copy(Char *to, const Char *from, SizeType n)
Unbound String function deprecated	SizeType Length(const Char *str)	SizeType NLength(const Char *str, SizeType size)
Unbound TextEncoding function deprecated	UInt32 CodePointCount(const Char* string)	UInt32 CodePointCount(const Char* string, SizeType n)
Unbound TextEncoding function deprecated	UInt32 CharCount(const Char* string)	UInt32 NCharCount(const Char *string, UInt32 size)
Unbound TextEncoding function deprecated	Int32 Compare(const Char* s1, const Char* s2)	Int32 NCompare(const Char* s1, const Char* s2, SizeType charCount)
Unbound TextEncoding function deprecated	bool Match(const Char* s1, const Char* s2)	bool Match(const Char* s1, const Char* s2, UInt32 codePointCount)
Type change	Data type TChar.	Use Char instead.
Type change	Data type Ucs2Char.	Use Char (UTF-8 encoded) or UInt16 as UTF-16 encoded codepoint or UInt32 as UTF-32 encoded codepoint instead.
TextEncoding change	TextEncoding::Ucs4 and TextEncoding::NUcs4 methods.	Use TextEncoding::GetUtf32 instead.
FixedSizeString change	FixedSizeString::TCharCapacity method.	Use FixedSizeString::CharCapacity instead.

Scene Composer 3.12

New Feature

- Fusion
  - Fusion allows you to structure your scene logic easily in a visual node designer
- Vector Graphics
  - Support for vector graphics is added, which can be created in Scene Composer or imported via SVG files.
- State Machine
  - State machine editor functionality for annotations added. Annotations with text nodes without interaction to state machine and nodes can be colored and annotated with an image.
  - UML pseudostate functionality added
- Smart Importer
  - Support for Figma design files is added
- Requirement tracking
  - For easier requirement tracking an URI, next to annotation and description, can be defined by the user. If the URI is valid a clickable link is available to directly access the resource (e.g. a link to your requirement tool).
- Display rotation support
  - Rotation of Scene can be configured, in case your HMI is physically rotated build into your product.
- Referencing external resource functionality (Android Platform only)

Changes

- Smart Importer
  - AI for control detection is directly integrated in Scene Composer
  - Additional UI/UX improvements
- Automatic import feature improved
  - Automatic import was improved to support new use cases and the handling was improved.
- Improved Properties panel
- Improved HMI Report function
  - HMI report doesn't require Python anymore and is executed faster.
- Display Preview support