# CGI Studio 3.9

Release Date: June 2020

# General 3.9

# Changes in MixedReality sample

MixedReality sample has been modified and uses the new Video Stream (Video Stream) functionalities. Furthermore, the intro video is played in a infinite loop instead of once, which is realized with the new condition behavior (Check Video State) and action behavior (Configure and Play Stream).

General 3.9

# Removal of Candera Safety sample

Candera Safety sample, which was added in version 3.4.1, has now been removed in version 3.9. Support for Candera Safety will be dropped until further notice.

# Candera 3.9

- **Animation framework immediately applies the value of the first keyframe after the animation is started**:
  CGI1-38759
  First key frame value is not applied immediately anymore but at the time the keyframe is defined.
- **Truncation wrong left bearing has been considered**:
  CGI1-41803
  Text does not move anymore in x direction when it’s truncated.
- **Performance issue in Node2D::GetWorldTransform()**:
  CGI1-40883
  Performance Improvement for GetWorldTransform() function for 2D nodes.
- **ASTC Compression support for OpenGL ES 3.2**:
  CGI1-42340
  Support of ASTC compressed textures.

## Text Engine Changes

General improvements of the truncation algorithm have been added.

- The default texture size of GlyphAtlas textures has been reduced to 512x512. (CGI1-42474 - 10475)
- Information about the fill status of the glyph atlas and its textures can be retrieved now. (CGI1-42576 - 10477)
- Korean line break has changed to western style. (CGI1-42979 - 10500)
- The ignore list for special control character has been reworked. (CGI1-42963 - 10499)
- The CMake flag which enables the locl feature: CANDERA_TEXTENGINE_WORLDTYPE_LOCL_FEATURE_ENABLED changed its default value to false. If this feature is required, an additional feature from Monotype is required to make it functional. (CGI1-41591 - 10408)
- The datatype PixelPosition has been changed from Int16 to Int32.
- The truncation algorithm has been stabilized regarding out of memory behavior. This has impact on the return type of the functions
  PreprocessedGlyphData*

  Candera::TextRendering::GlyphDataContainer::operator[](TextPosition const textPos);

  bool Candera::TextRendering::GlyphDataContainer::AddNew();

  bool Candera::TextRendering::GlyphDataContainer::AppendInvalidGlyphs(TextPosition const invalidCount); (CGI1-42726 - 10483)
- The Monotype WT-Shaper has been upgraded to version 4.4.0. (CGI1-45481)

## Asset loading improvements

Asset loading has been optimized. This includes changes in asset partitioning. (10430, 10443, 10442, 10417)

## Detailed asset error information and asset initialization

It is possible to receive more detailed information about asset loading failurs now. (10338)

For this reason the Candera::AssetConfig class has been extended by the interface

Candera::AssetConfig::AssetErrorsInfo * Candera::AssetConfig::GetAssetErrorsInfoPtr();
const Candera::AssetConfig::AssetErrorsInfo & Candera::AssetConfig::GetAssetErrorsInfo() const;

This information contains all errors thrown while loading an asset. The different error types are defined in the enumeration Candera::AssetConfig::AssetErrorsInfo::ErrorInfo::Enum.

Additionally, the asset provider can be now tested on whether it is initialized to prevent corrupted access on Candera::AssetProvider (10468).

Candera::AssetProvider::IsInitialized();

## RTTI support

Different classes have received Candera built-in RTTI system support:

- Candera::GraphicDeviceUnit and its derivates.
- Candera::Layouter and its derivates.

## Asynchronous canvas text processing

The 3D text type canvas text supports asynchronous processing now.
It is coupled to the feature of asynchronous text processing of 2D text type TextNode2D. Therefore, the approach is equal. A new property on canvas text enables the asynchronous text processing. This asynchronous text processing can be done asynchronously inside an own worker thread or single threaded but asynchronously scheduled over multiple frames.

The property can be changed in SceneComposer on the canvas text node itself or by code:

Candera::CanvasText::SetAsyncPreRenderEnabled(bool val);

Candera::CanvasText::IsAsyncPreRenderEnabled();

Asynchronous text rendering has been made more robust the API provides now a callback function which will be called in the render thread as soon as the text processing has finished:

void Candera::TextRendering::ITextValidationUser::ResyncNodeRenderFinished()

The TextRenderDispatcher will call this function accordingly.

To support asynchronous text processing in CanvasText, some classes have been extended:

void Candera::TextRendering::TextRenderArgs::SetAdditionalShaperOptions(Candera::Globalization::Culture::SharedPointer& culture, Candera::TextRendering::PixelSize outlineWidth);
Candera::Globalization::Culture::SharedPointer GetCulture() const;
Candera::TextRendering::PixelSize Candera::TextRendering::TextRenderArgs::GetOutlineWidth() const;
Candera::TextRendering::TextSize Candera::TextRendering::TextRenderArgs::GetInnerSizeRestriction() const;
void Candera::TextRendering::TextRenderArgs::SetInnerSizeRestriction(Candera::TextRendering::TextSize const& val);
void Candera::TextRendering::TextRenderArgs::SetValidationUser(Candera::TextRendering::ITextValidationUser * validationUser);
void Candera::TextRendering::TextRenderArgs::SetAsyncEnabled(bool asyncEnabled);

Additionally, it is possible to process the text during general asset scene loading upload time. With this, it is possible to load a 3D scene in background. When the scene has finished loading, the text is already prepared for rendering. This process run does not consider layout. Therefore, another process run during layout might be required. At least, the glyph caches should have its required glyph entries at this point. Rasterization can be skipped then.

The property to enable this feature can be set in SceneComposer on the canvas text node itself or by code:

Candera::CanvasText::SetPreprocessOnUploadEnabled (bool val);
Candera::CanvasText::IsPreprocessOnUploadEnabled ();

## Canvas text layout invalidation

CanvasText supports the possibility to disable and reenable its internal invalidation mechanism. It is possible to toggle the internal layout invalidation which happens inside, too. This is useful for custom layouter implementation as setting a property inside of the layouter code may not trigger layout in the next frame again.

Candera::CanvasText:: SetLayoutInvalidationEnabled (bool val);

Candera::CanvasText:: IsLayoutInvalidationEnabled ();

## TextNode2D invalidation

Candera::TextNode2D supports disabling of the invalidation mechanism now.

Candera::TextNode2D::IsInvalidationEnabled();

Candera::TextNode2D:SetInvalidationEnabled(bool val);

## Animation speedup

It is now possible to increase the speed of an animation for a defined set of milliseconds. bool

Candera::Animation::AnimationPlayer::Speedup(Float speedFactor, SequenceTimeType durationMs)
;

This speedup can be triggered by an AnimationReqMsg, too.

## Glyph atlas statistics

It is possible to retrieve statistics of the glyph atlas now (10477). The statistics include how many glyphs are inside the glyph atlas and how much space is occupied inside the texture(s).

The information can be retrieved on a specific glyph atlas texture index or all used textures. The functions can be called on the specific instances of a glyph atlas:

SizeType GetGlyphCount() const;
SizeType GetGlyphCount(SizeType atlasIndex) const;
Float GetOccupiedSpace() const;
Float GetOccupiedSpace(SizeType atlasIndex) const;

With the introduction of this feature, BitmapAtlas has been extended by a function to extract the occupied space inside:

UInt GetOccupiedSpaceInTexels() const;

## Dedicated text properties for text flow direction

CanvasText and TextNode2D support setting the text reading direction explicitly now. This feature can only be set by code for now.

API on CanvasText:

Candera::TextRendering::BidiBaseLevel::Enum Candera::CanvasText::GetBidiBaseLevel() const;

void Candera::CanvasText::SetBidiBaseLevel(Candera::TextRendering::BidiBaseLevel::Enum val);

API on intermediate text arguments:

Candera::TextRendering::BidiBaseLevel::Enum
Candera::TextRendering::TextRenderArgs::GetBidiBaseLevel() const;
void
Candera::TextRendering::TextRenderArgs::SetBidiBaseLevel(Candera::TextRendering::BidiBaseLevel::Enum val);

API on TextNode2D:

static Candera::TextRendering::BidiBaseLevel::Enum Candera::TextNode2DLayouter::GetBidiBaseLevel(const TextNode2D& node); static void Candera::TextNode2DLayouter::SetBidiBaseLevel(TextNode2D& node, Candera::TextRendering::BidiBaseLevel::Enum val);

## Customized font metrics

It is possible to set custom metrics on fonts and styles by code. These metrics replace the automatically calculated ones (CGI1-41735 - 10410). The font metrics used by default are aligned to the bounding boxes of the glyphs inside the font/style. With this feature it is possible to define a specific metric value and keep the exact position of text even if the font or style has changed inside or the position is imported from external tools. It is possible to switch back to automatically calculated metrics.

void Candera::TextRendering::Style::SetMetrics(Metrics const& newMetrics); void Candera::TextRendering::Style::SetDefaultMetrics();

void Candera::TextRendering::Font::SetMetrics(Metrics const& newMetrics); void Candera::TextRendering::Font::SetDefaultMetrics();

# Text glyph iterator extension

The glyph iterator interface supports the retrieval of horizontal X bearing now. The derivates have to implement this function to fully support it.

Candera::TextRendering::PreprocessingContext::GlyphData::GetHorizontalBearingX();

## RenderTarget DestroyEvent

The typedef Candera::RenderTarget::DestroyEvent has been introduced for better readability and access to destroy events.

## Usage of Monotype OT caching

In some cases Monotype opentype cache might have used cached data from previous fonts which then leads to wrong opentype table data in use. This has been fixed (CGI1-43094).

## Specular highlights and spot lights

Reference shaders were calculating half vectors in point and spot light without normalizing them. As a result, the distance from the light to a vertex influenced the result, producing incorrect shading. Reference shaders were updated to correctly normalize half vectors in vertex shader for point and spot lights (CGI1-19781).

## Layout monitor

The layout monitor communication protocol has been updated. The IDs referencing to nodes are 64-Bit now to fully support 64-Bit systems (CGI1-39844).

## View invalidation on text invalidation

In case of a text gets invalidated, it will invalidate the courier view in addition to the layout now.

This has been introduced into Candera::TextNode2D and Candera::CanvasText (8719).

## Screenshot tool

The snapshot tool did not fully support features like super sampling when OpenGLES2.0 was used. This has been fixed (CGI1-38463).

## Text direction when using no shaper

The text direction when using NoShaper sometimes selected the wrong text flow direction for specific glyphs. This has been fixed (CGI1-26302).

## Analyzer – Frame Debugger

The algorithm to compress a buffer inside of frame debugger monitor has been replaced with a different implementation (CGI1-45320).

## CanvasSprite Upload strategy

CanvasSprite's default vertex buffer can potentially get uploaded more often than unloaded, causing a crash post-main. This has been fixed (CGI1-33876).

## CanvasText line intersection

An issue that prevented intersecting lines with valid CanvasTexts was fixed (CGI1-26345).

## Release method for SharedPointer and EventListener

Design refactoring: SharedPointer and EventListener have both a Release method (CGI1-21916).

## Redundant mipmap auto generation

Redundant mipmap auto generation in RenderDevice OpenGL ES 1.1 when texture with predefined mipmaps is uploaded. (CGI1-35758)

## Endianess function template LittleToBigEndian is wrong

LittleToBigEndian conversion is wrong (CGI1-44380).

## Dirty Area statistics is wrong for scissor operations

Dirty Area statistics broken for platform that support Scissor operations (CGI1-24992)

## Uniform buffer objects without padding do not work

MaterialBlock and LightBlock shader parsing does not handle padding at struct end correctly in case that there is no padding. Std140 layout says that padding may or may not exist.

## Data binding has dependency to view availability

Data Binding works only if the model updates when the view is already initialized. If the model updates data binding while the view is not initialized yet, after the view is initialized it cannot access the current data binding message. (CGI1-44499)

## CanvasSprite is not unloaded when changed to 9-patch

CGI1-33876 Default shared static vertex buffer of CanvasSprite is not unloaded when being changed to 9-patch, thus triggering unloading post-main causing a crash.

## CMake Build System Changes

New cmake variables were added:

| Name | Description | Possible values |
|---|---|---|
| CANDERA_GLYPHATLAS_PADDING_SIZE | Padding size around glyphs in the GlyphAtlas. Value depends on how the glyphs are being rendered. E.g. 4 if reference outline shader is used. 1 if bilinear texturing is used. | 4 |
| CANDERA_TEXTENGINE_WORLDTYPE_CLIG_FEATURE_ENABLED | Defines whether or not Contextual Ligatures (clig) are enabled. This flag is for wt-shaper only. | ON/OFF (Default: ON) |

| | | |
|---|---|---|
| CANDERA_TEXTENGINE_WORLDTYPE_DLIG_FEATURE_ENABLED | Defines whether or not Discretionary Ligatures (dlig) are enabled. This flag is for wt-shaper only. | ON/OFF (Default: ON) |
| CANDERA_TEXTENGINE_WORLDTYPE_LIGA_FEATURE_ENABLED | Defines whether or not Standard Ligatures (liga) are enabled. This flag is for wt-shaper only. | ON/OFF (Default: ON) |
| CANDERA_TEXTENGINE_WORLDTYPE_LOCL_FEATURE_ENABLED | Defines Whether or not Localized Forms (locl) are enabled. The feature requires WT-Shaper 4.3.0 including 136475_136476. | ON/OFF (Default: ON) |

# Courier V3.9

## Changes

## Animation speedup

An animation speedup can be triggered by an AnimationReqMsg. See also Animation speedup.

The class Courier::AnimationProperties has been adapted accordingly:

- Float Courier::AnimationProperties::GetSpeedFactor() const;
- Candera::Animation::SequenceTimeType Courier::AnimationProperties::GetSequenceDurationMs() const;
- Courier::AnimationProperties::AnimationProperties( SetMask mask, Float speedFactor, UInt32 repeatCount, Candera::Animation::SequenceTimeType sequenceStartTimeMs, Candera::Animation::SequenceTimeType sequenceDurationMs, Candera::Animation::AnimationPlayer::RepeatMode repeatMode, Candera::Animation::AnimationPlayer::PlayDirection playDirection);
- Courier::AnimationProperties::AnimationProperties(SetMask mask, Candera::Animation::AnimationPlayer::PlayDirection playDirection, Float speedFactor, UInt32 repeatCount, Candera::Animation::AnimationPlayer::RepeatMode repeatMode, Candera::Animation::SequenceTimeType sequenceStartTimeMs, Candera::Animation::SequenceTimeType sequenceDurationMs, UInt32 finishTime);
- Courier::AnimationProperties::AnimationProperties(Float speedFactor, Candera::Animation::SequenceTimeType sequenceDurationMs);
- explicit Courier::AnimationProperties::AnimationProperties(UInt32 finishTime);

---

## Improve animation view dependency handling

If an animation is running and a dependent view is loaded during this animation, the invalidation handling was not done properly. This has been fixed. This fix has also introduced the new interface to force the update on these dependent views:

- void Courier::IViewHandler::UpdateAnimationScenes();

---

## Asset initialization state

The asset accessor supports the access to the initialization state of the asset provider. See also Detailed asset error information and asset initialization.

bool Courier::AssetAccessor::IsAssetProviderInitialized();

---

## Transition interface for Courier::ViewId

Courier provides an interface to control transitions by Courier::ViewId now. In the past it was only possible to do this by using Candera::Transitions::Identifier (9418). To support this, the following interfaces have been added:

- The message type Candera::CanderaSceneTransitionRequestMsg
- bool Courier::IViewHandler::ExecuteCanderaTransitionReqAction(CanderaTransitionRequestAction::Enum, const Courier::ViewId&, const Candera::Transitions::Hint&);

---

## Extended thread-safety in binding sources

The thread safety has been extended in the binding sources of DataBinding when asynchronous binding is used (CGI1-42074 ﷺ 10441).

---

## Courier::Gdu render state

The render state of a Courier::Gdu can be retrieved now:

- Courier::Gdu::RenderState Courier::Gdu::GetRenderState() const;

---

# Controls 3.9

# New Controls

## Android Control

Introduced Android Studio UI element that displays a scene rendered by CGI Studio. It is possible to dispatch events between an android application and CGI Studio application with the ‘Android Event Bridge’ behavior.

# Control Modifications

## General

Fixed issues and improved stability and performance of the following behaviors:

- AnimationActionBehavior
- SwitchRangeProcessValueBehavior
- TextProcessValueBehavior

---

## Roll control

The Roll Control now supports Text Style changes. Properties "UnitStyle", "LensStyle" and "ValueStyle" have been added to Roll.

---

## Button Control

A **Selected** property has been added to the Radio Button Control for databinding use-cases.

---

## Text and TextButton

Text and Text Button are now compatible with the Set Text behavior.

---

## Arithmetic Operation Behavior

Modulo operation has been added to Arithmetic Operation behavior.

---

## Set Scale Behavior

Fixed an existing issue with z-axis scaling.

---

## Controls Use BitmapBrushColorBlend

All Controls are now using BitmapBrushColorBlend.

---

## Set Layout Size Behavior

- Renamed the Behavior "Set Size" to Set Layout Size
- Added Relative/Absolute mode property (Prior it was ‘Relative’ only)
- Added Y-Axis scaling

---

## VideoStream Control (2D and 3D)

Property "Timestamp" and "Source Type" have been added to VideoStream in 2D and 3D.

---

## VideoStream Control (2D and 3D)

Property "Timestamp" and "Source Type" have been added to VideoStream in 2D and 3D.

---

## Cover Flow Control

Property "Orientation" has been added to CoverFlow. No solution conversion required (only unnecessary properties "BindableProperty" and "BindableProperty_1" have been removed).

---

# Control Behaviors 3.9

## New Behaviors

The following Behaviors have been added in CGI Studio 3.9:

- Action/Streaming/Configure and Play Stream
- Condition/Control/Check Video State
- Control/Android Event Bridge
- Control/Public Property/Set Layout

## Properties

The following Behavior properties have been changed or added:

- CheckValueConditionBehavior:
  - Added Operation Mode and Value Behavior
- AnimationActionBehavior:
  - Added Change Action
- AnimationStateConditionBehavior:
  - Added Change Action detection
- PropertyProviderBehavior:
  - Added Style
  - Added Layout
  - Added Stack Arrangement
- VideoStreamBehavior:
  - Added Timestamp
  - Added Source Type

## Further Changes

# FeatStd 3.9

## Increased stack size for Integrity threads

The stack size has been increased for Integrity threads. Every thread created by FeatStd::Internal::Thread is affected. So every Candera engine thread other than the main thread has the new stack size.

## Assertion function with message

Additional assertion function has been added which prints a defined messsage when hit. (9735)
The macro FEATSTD_DEBUG_ASSERT_MESSAGE(condition, message, ...) can be found in FeatStd/Diagnostics/Debug.h.
This macro is resolved into the public interface:
**bool FeatStd::Diagnostics::DebugControl::Assert(const Char* expression, const Char* filename, Int lineNr, const Char *msgFormat, ...);**

## Option to copy StringData

A CMake option FEATSTD_STRINGDATA_COPY_ENABLED has been added for copying the internal StringData of a FeatStd::String. When the flag is enabled (default: OFF), then instead of sharing and reference counting the internal StringData, it is copied. This might be useful in application environments, where FeatStd::String objects are modified by several threads without controlling access via FeatStd::String::GetCriticalSection().

## Interface Changes

- Shared Pointer

  FeatStd::MemoryManagement::SharedPointer is an intrusive smart pointer that injects 2 functions "Release" and "Retain" into the class scope of the object to be shared. Because "Release" and "Retain" are commonly used names for other functions, this can potentially result in a naming conflict with undefined behavior. Therefore these functions were renamed to "SharedPointerIntrusiveRelease" and "SharedPointerIntrusiveRetain". Any external use of these functions must be adapted.
- Migration Guide

| Description | FeatStd V3.9.0 | Candera V3.9.0 |
|---|---|---|
| SharedPointer changes | Using a SharedPointer within a class introduced "Release" and "Retain" functions. | Using a SharedPointer within a class now introduces "SharedPointerIntrusiveRelease" and "SharedPointerIntrusiveRetain" instead. |

# SceneComposer 3.9

## Functional Changes

- **Welcome Screen**
  CGI1-44422 - Welcome Screen for CGI Studio Scene Composer
  The welcome screen shows highlights of the new release about CGI Studio.
  You can disable this dialog if you select the check box at the bottom of the dialog.
- **Photoshop Importer**
  CGI1-44314 - Smart Photoshop Importer
  The Smart Photoshop Importer allows the user to import Photoshop files (*.psd) with a direct mapping to CGI Studio Controls.
  This process of mapping Photoshop element to existing Controls is assisted by artificial intelligence to allow the user a fast and easy mapping.