CGI Studio 3.0.0

Candera V3.0.0

Overview

Release Info

- Release Date: July 10, 2014

Interface Changes

- Candera Engine 3D
- Candera Engine 2D
- Candera Base
- Candera System
- Candera Device
- Candera AssetLoader
- Text Engine

Migration Guide and Backwards Compatibility

In CGI Studio V3.0.0, deprecated interfaces from CGI Studio V2.10.0 can still be used to a large extent, while deprecation functions for interfaces from releases before V2.10.0 have been removed completely.

- When interfaces are used which are marked as **deprecated**, compiler warnings will be generated through the usage of Candera Macro **CANDERA_DEPRECATED_3_0_0**.
- Interface changes in V3.0.0, which are **not backward compatible** to V2.10.0, are explicitely listed in the page below.

Refer to:

- Candera V3.0.0 Migration Guide

Candera Engine 3D

OpenGL ES 3.0 Support

RenderDevice for OpenGL ES 3.0

In order to support OpenGL ES 3.0 a new RenderDevice.cpp has been added, which implements RenderDevice.h with OpenGL ES 3.0 functions. Additionally GlInclude.h, GlTraceMapper and GlTypeMapper have been adapted to support the split of one RenderDevice into several ones for multiple GLES versions. GlTraces are now all displayed under Gles device (instead of Gles20).

For current device packages OpenGL ES 2.0 is still used. If it is supported by a device, usage of OpenGL ES 3.0 can easily be configured in Device.cmake. Simply include Common/OpenGLES30 instead of Common/OpenGLES20 and add the following line to the CMake file.

```
CgiAddDefinitions( -DCGIDEVICE_OPENGLES_30 )
```

Synchronization for OpenGL ES 3.0

The OpenGL ES 3.0 render device uses core OpenGL ES 3.0 fences for synchronization, instead of EGL extension EGL_KHR_fence_sync.

OpenGL ES 3.0 Support for Linux OpenGLAdapter

OpenGLAdapter for Linux hosts now also supports OpenGL ES 3.0 interface (based on GLX and OpenGL 3.0).

OpenGL ES 3.0 ETC2/EAC Texture Compression

OpenGL ES 3.0 introduces ETC2/EAC texture compression as part of the core standard. This enables all OpenGL ES 3.0 capable devices to reduce VRAM and memory bandwidth usage per texture at unnoticeable reduction of image quality, allowing either higher framerates or more and bigger textures used simultaneously.

Candera now supports this feature in the OpenGL ES 3.0 render device. For this purpose Bitmap class has been changed (see Changes in Bitmap Enumerations). The new Bitmap::PixelFormat enumerator has to be used for texture uploading.

See also:

- Import Images for information how to import ETC2/EAC compressed images into SceneComposer.

Cloning Interface

In accordance to basic Unified Cloning Interface, the following changes have been applied to the affected classes in Candera 3D:

- CloneInternal protected interface was removed.
- Clone(Traversing) has become deprecated.
- Copy constructor was implemented or cleaned up.
- Assignment operator was moved to the private section, and its implementation removed.
- Clone() const function was added.

The cloning interface was applied to:

- Node and its descendants: Billboard, Camera, Group, Light, LineList, LodNode, Mesh, MorphingMesh, PlanarShadow, PointSprite, ReflectionCamera, Scene, StereoCamera.
- All other objects that are linked to node and descendants through shared pointers: Appearance (and MultiPassAppearance), Material, RenderMode, Texture, AbstractShaderParamSetter, GenericShaderParamSetter, ShaderParamSetter, Projection, GenericProjection, OrthographicProjection, PerspectiveProjection.
- RenderOrder, as its lifetime is controlled by Scene.

Special considerations:

- General. Clone for shared objects returns a shared pointer to the base shared class.
- General. Descendants of Node implement deprecated "Node* Clone(Traversing)" for backward compatibility.
- Billboard. Bounding box and bounding sphere are copied from the base class, not recomputed.
- Camera. CameraRenderStrategy is not linked to the clone anymore.
- LodNode. Clone contains the same number of levels as the original, but no level information is stored.
- Node. CopyFrom is provided to copy transformations, appearance and other such properties from this node to another one.
- PlanarShadow. Clone loses association to light.

- PlanarShadow. As defined by the Copy constructor (but not the assignment operator, previously used for cloning), the id of the clone is different from the one of the original.
- ReflectionCamera. Clone loses association to source camera.
- Scene. Scene may be cloned. The clone does not receive a render order.
- StereoCamera. ProjectionListener may be copied.
- ShaderParamSetter. Clone does not hold any data since data is handled by the application.
- AssetLoader nodes. They don't support cloning.

Deep Cloning Interface

In accordance to basic Tree Cloning Strategies, the following support is provided for deep cloning in 3D as follows:

- TreeCloner is a specialization of TreeClonerBase.
- DeepNodeCloneStrategy provides a generic node strategy that clones the delegates the cloning of appearance and derived Nodes to specialized strategies.
- DeepAppearanceCloneStrategy provides deep appearance cloning. It clones its constituents by resolving shared instances.
- DeepCompositeGroupCloneStrategy, DeepLodNodeCloneStrategy, DeepStereoCameraCloneStrategy and DeepReflectionCameraCloneStrategy resolve node binding by use of TreeMatch.
- DeepCameraCloneStrategy resolves the sharing of projections.
- DeepSceneCloneStrategy clones the RenderOrder.
- DeepTreeCloner is a wrapper of DeepNodeCloneStrategy that provides an interface similar to TreeCloner.

CompositeGroup

CompositeGroup is a new Node type that is usually created by AssetLoader and which contain:

- a list of AnchorPoints: descendants of the CompositeGroup that can be accessed directly from the CompositeGroup instance.
- a list of Widgets that act on any node within the CompositeGroup subtree.
- a list of AnimationPlayers that act on any node within the CompositeGroup subtree.

Picking Test: Consistent Behavior for deep and flat Picking

In both cases Node::IsPickIntersectingGeometry only returns true for a tested node if it is hit by the intersection ray AND Node::IsEffectiveIntersectionTestEnabled is true.

DeviceObject Listener

DeviceObject now supports adding listeners that are notified before and after the object is uploaded and unloaded and before it is destroyed.

Upload to Multiple Contexts

Device Objects of a Node (and its descendants) can be uploaded to all render target contexts belonging to the cameras the nodes are in scope of.

See also:

> Node::UploadAll(). Furthermore, logical operators for intersection and union of Scope masks have been added to Candera::ScopeMask.

ExternalTextureImage

Added ExternalTextureImage which is a texture image that allows wrapping external created native texture objects.

Car Paint Shaders added

The following reference shaders have been added:

- RefCarbon / RefTransLight1Carbon
  Creates a car paint with typical carbon look.

- RefFlopCarPaint / RefTransLight1FlopCarPaint
  Creates a car paint with shiny, anisotropic two color look.

- RefMetalFlakesCarPaint / RefTransLight1MetalFlakesCarPaint
  Creates a car paint with shiny, sparkling look produced by metal flakes within specular highlight.

- RefFlopMetalFlakesCarPaint / RefTransLight1FlopMetalFlakesCarPaint
  Creates a car paint with shiny, sparkling look produced by metal flakes within specular highlight combined with anisotropic two color look.

Candera Engine 2D

Deterministic Render Order

In the past, RenderNodes with the same depth value have been rendered in an order which was dependent on the point of time the node has been added to the scene. The result was that in SceneComposer the render order could look different as when

loading the scene from the asset on target.

The new implementation ensures that the render order of nodes with the same depth value is defined by the position within the scene tree. Nodes which appear on top of the scene tree (as shown in SceneComposer) are rendered before nodes that are located on the bottom of the scene tree. Therefore the render order is always the same during design on host (SceneComposer) and on target.

Cloning Interface

In accordance to basic Unified Cloning Interface, the following changes have been applied to the affected classes in Candera Engine 2D:

- CloneSelf(CloneOperation) protected interface was removed.
- Clone(TraverseOperation, CloneOperation) has become deprecated.
- Copy constructor lost the optional CloneOperation argument.
- Clone() const function was added.

The cloning interface was applied to:

- Node2D and its descendants: Camera2D, CompositeGroup2D, Group2D, RenderNode, Scene2D.
- Effect2D and Layouter already define the interface appropriately.

Special considerations:

- General. Descendants of Node implement deprecated "Node2D* Clone(Traversing) const" and "Node2D Clone(TraverseOperation, CloneOperation) const" for backward compatibility.

- Camera2D. Camera2DRenderStrategy and RenderTarget2D are not linked to the clone anymore.
- Node2D. Layouter is copied during deep node cloning. Because of this Node2D handles the destruction of Layouter upon disposing the node.
- Scene2D. Scene2D may be cloned. Scene2D render order is not cloned, as it doesn't support different render order bins.

Deep Cloning Interface

In accordance to basic Tree Cloning Strategies, the following support is provided for deep cloning in 2D as follows:

- TreeCloner2D is a specialization for the 2D scene tree of TreeClonerBase.
- DeepNode2DCloneStrategy provides a generic node strategy that clones the delegates the cloning of derived Nodes to specialized strategies.
- DeepCompositeGroup2DCloneStrategy resolves node binding by use of TreeMatch2D.
- DeepRenderNodeCloneStrategy resolves the sharing of effects by use of a map.
- DeepTreeCloner2D is a wrapper of DeepNode2DCloneStrategy that provides an interface similar to TreeCloner2D.

CompositeGroup2D

CompositeGroup is a new Node2D type that is usually created by AssetLoader and which contain:

- a list of AnchorPoints: descendants of the CompositeGroup2D that can be accessed directly from the CompositeGroup2D instance.
- a list of Widgets that act on any node within the CompositeGroup2D subtree.
- a list of AnimationPlayers that act on any node within the CompositeGroup2D subtree.

SharedPointer: Deprecated '...Ptr' types

All effect Ptr types like e.g. BitmapBrushPtr are marked as deprecated. Please use e.g. BitmapBrush::SharedPointer instead.

Deprecated BaseEffect2DPropertySetter

BaseEffect2DPropertySetter and its derivate classes became deprecated. Effect2D objects, as other objects with attached MetaInfo, can already animate their properties using built-in PropertyMetaInfo AnimationPropertySetters, with an improved performance.

AnimationPropertySetters Namespace Correction

All AnimationPropertySetters residing in `<cgi_studio_candera>/src/Candera/Engine2D/AnimationPropertySetters` have been moved from the *Candera* to the *Candera::Animation* namespace. The following classes are affected by this change:

- AlphaNode2DPropertySetter
- BaseNode2DPropertySetter
- BaseTransformable2DPropertySetter
- BaseTransformable2DRelativePropertySetter
- BaseEffect2DPropertySetter
- Effect2DFloatPropertySetter
- RenderingEnabledNode2DPropertySetter
- Transformable2DRelativeRotatePropertySetter
- Transformable2DRelativeScalePropertySetter
- Transformable2DRelativeScaleXPropertySetter
- Transformable2DRelativeScaleYPropertySetter
- Transformable2DRelativeTranslatePropertySetter
- Transformable2DRelativeTranslateXPropertySetter
- Transformable2DRelativeTranslateYPropertySetter
- Transformable2DRotatePropertySetter
- Transformable2DScalePropertySetter
- Transformable2DScaleXPropertySetter
- Transformable2DScaleYPropertySetter
- Transformable2DTranslatePropertySetter
- Transformable2DTranslateXPropertySetter
- Transformable2DTranslateYPropertySetter

> Moved classes are still accessible via the Candera namespace in CGI-Studio 3.0.0 but these deprecated interfaces will be removed in the next release.

Candera2D Bitmap MipMap Flag

Candera2D internally uses MipMapping for Bitmaps, as long as OpenGL is used to render. Whereas this measure typically improves image quality, MipMaps consume ~1/3 more memory and might bear side effects (interpolation between MipMap levels may produce artifacts). Therefore a MipMapEnabled flag has been added.

See also:

Candera::Bitmap::SetMipMappingEnabled() and Candera::Bitmap::IsMipMappingEnabled().

Camera2D Viewport Transformation functions moved

Viewport transformation functions have been moved from Candera::Camera2D to new class Candera::Math2D. Picking in 2D considers window offset.

Reduced size of members

To reduce heap usage some elements in Candera 2D have been reduced in size:

- The camera sequence number from 32 to 16 bit,
  See also:

  Candera::Camera2D::SetSequenceNumber(Int16 sequenceNumber)
- The bitmap width and height from 32 to 16 bit,
  See also:

  Candera::Bitmap
- The child count of Node2D from 32 to 16 bit,
  See also:

  Candera::Node2D::GetChildCount()

DeviceObject2D listener.

DeviceObject2D now supports adding listeners that are notified before and after the object is uploaded and unloaded and before it is destroyied.

Layout Rectangle

The Layout is no longer done based on bounding rectangle. Instead, Node2D has a new property, layout rectangle. Bouding rectangle is used for picking and dirty area management. Computed layout and bounding rectangle are based on affect layout and bounding rectangle, respectively.

See also:

Node2D::SetLayoutingRectangle() and Candera::Effect2D::GetLayoutingRectangle().

Candera::Node2D uses new Layout Rectangle instead of Bounding Rectangle for layout.

TextBrush and TextBrushCaches compute their Layout Rectangle based on Text Cursor Position. MaximumSize property of TextBrush has been replaced by CacheArea. Bounding Box and Cache Size have been decoupled from Layout. To retrieve the height of a TextBrush with ConstantHeight use GetLayoutingRectangle, and for ActualHeight use GetBoundingRectangle.

Renderer2DListener

A Renderer2DListener allows to listen to Renderer-Events in Candera2D like in Candera 3D with RendererListener.

Candera Base

Camera Groups

CameraGroup is a container that groups cameras from multiple scenes. CameraGroup has an arbitrary number of 2D and/or 3D cameras from different scenes and it is used by applications to easily access and change camera settings uniformly, e.g. in transitions.

See also:

- Texture Render Targets

Unified Cloning Interface

Base classes implement a virtual "Clone() const" method, which is overridden in all children. "Clone() const" builds a new instance of the class using the copy constructor. Copy constructors are protected for use only in derived copy constructors. Copying via assignment operator is prohibited, as it could cause problems, like orphaned nodes, memory leaks or dangling pointers. Assignment operators are thus private.

Cloning is shallow, with only shared pointers and constant character pointers being copied. All other coupling via pointers is broken for the clones.

For more complex cloning see Tree Cloner and Tree Cloning Strategies.

Tree Cloner

TreeClonerBase is provided for cloning trees. The default clone strategy is to call the Clone interface on each node, and build the clone tree with the same node pattern as the original tree.

For deeper cloning, TreeClonerBase supports attaching a TreeCloneStrategy. This

TreeCloneStrategy is typically aware of different types of nodes, and delegates to other strategies.

Tree Cloning Strategies

The deep cloning strategies respect the following rules:

- all shared resources are deep cloned as long as they are not device objects; shared resources in the destination are shared in the same pattern as in the source tree.

- node references are rebuilt as long as source references are found as descendants of the source root; descendants are mapped to the destination tree.
- strategies are designed to support the decorator pattern.
- general strategies delegate to more specific strategies.

Bitmap as SharedPointer

Instances of class Bitmap can only be created by calling static method Bitmap::Create() which returns a SharedPointer.

All interfaces have been changed to pass the shared pointer and the corresponding disposer function has been removed from the interfaces.

Old interface (example):

```
void BitmapImage2D::SetBitmap(Bitmap* bitmap, BitmapDisposerFn disposerFn);
```

New interface (example):

```
void BitmapImage2D::SetBitmap(Bitmap::SharedPointer bitmap);
```

Changes in Bitmap Enumerations

Until now, Bitmap class described the organization of the pixels with the enumerators Bitmap::Format and Bitmap::Type.

With introducing OpenGL ES 3.0 ETC2/EAC Texture Compression, it came up that the bitmap description was not suitable for describing compressed images (which have a format descriptor, but no type descriptor). Another problem was that the Bitmap class allowed invalid combinations of Format and Type, e.g. Bitmap::Format::RgbFormat (3 components) and UnsignedShort4444Type (4 components).

In order to enable compressed bitmaps and to avoid invalid combinations the enumerator Bitmap::PixelFormat has been introduced, here Format and Type have been merged to valid enumeration items, while the ETC2/EAC items have been added.

For this reason Bitmap::Format and Bitmap::Type have become deprecated. Of course it is still possible to use custom formats.

Shared Pointer Concept in Animation Framework

Objects of the following classes can only be created by calling static method Create() which returns a SharedPointer:

- AnimationController
- AnimationTimeDispatcher
- AbstractEasingFunction (and all derived classes)
- InterpolationStrategy (and all derived classes)
- AnimationPropertySetter (and all derived classes)

All interfaces have been changed to pass the shared pointer and the corresponding disposer function has been removed from the interfaces.

Old interface (example):

```
void AnimationPlayer::SetController(AnimationController* controller, DisposerFn disposerFn = 0);
```

New interface (example):

```
void AnimationPlayer::SetController(AnimationController::SharedPointer controller);
```

TreeTraverserBase for 'const' Traversing

Interface of template class TreeTraverserBase has been changed to use only Traverse() and ProcessNode(). TraverseConst() and ProcessConstNode() have been removed. This ensures type safe const correct usage but enforce all const-traverser implementations to be changed. Const traversers have to be defined the following way:

```
class MyTraverser : public TreeTraverserBase<const Node> {
    protected:
        virtual TraverserAction ProcessNode(const Node& node)
        {
        ...
        }
};
```

Candera::Animation Namespace extension

Classes and Types located in `<cgi_studio_candera>/src/Candera/EngineBase/Animation` which formerly belonged to the *Candera* namespace were reassigned to the Candera::Animation namespace.

Affected classes/types:

- AbstractEasingFunction
- AnimationBlendedProperty
- AnimationController
- AnimationGroupPlayer
- AnimationKeyframeSequence
- AnimationPlayer
- AnimationPlayerBase
- AnimationPlayerListener
- AnimationPropertySetter
- AnimationTimeDispatcher
- WorldTimeType
- SequenceTimeType
- BackEaseFunction
- BezierInterpolationStrategy
- BounceEaseFunction
- EaseInterpolationStrategy
- ElasticEaseFunction
- ExponentialEaseFunction
- InterpolationStrategy
- KeyframeSequence
- LinearInterpolationStrategy
- PowerEaseFunction
- SplineInterpolationStrategy
- StepInterpolationStrategy

> Moved classes and types are still accessible via the *Candera* namespace in CGI-Studio 3.0.0 but these deprecated interfaces will be removed in the next release.

CanderaObject becomes DynamicPropertyHost

CanderaObject inherits from DynamicPropertyHost. Objects previously inheriting from DynamicPropertyHost now inherit from CanderaObject. ApplicationData is deprecated. The DynamicPropertyHost allows to attach multiple application data objects.

Candera System

Property Validation

A new macro CdaValidityTest allows the definition of a validation function for Widget, Effect2D and Gdu properties. In case of failed validation the property will be highlighted in SceneComposer property list and an error message will be displayed.

Candera Device

Automatic Context Resource Pools

Context resource pools are created automatically, and assigned automatically to Device objects. If all objects attached to a display share a context resource pool, the display is associated to the context resource pool. Otherwise, the display has a null context resource pool. The interface for replacing context resource pools on the display is deprecated. So is the interface for manually creating context resource pools.

Layout Rectangle of Effects

Effect2D has an interface to compute the layout rectangle. Layout rectangle is the same as bounding rectangle for non-text effects. For text effects the layout rectangle is given by CursorTextMeasureContext, and bounding rectangle is given by GlyphTextMeasureContext.

ActualHight parameter of TextBrush has been removed. Text is now always constant aligned. Applications need to compensate for this parameter by moving the pivot of the node by the offset of the bounding rectangle.

RenderTarget Auto Clearing

Added auto clear flag (RenderTarget::SetAutoClearEnabled and RenderTarget::IsAutoClearEnabled) to Candera::RenderTarget2D / Candera::RenderTarget3D. This flag, unlike the auto swap, does not alter the clear setting of the Camera/Camera2D, but assures that the RenderTarget is cleared automatically before the first camera renders.

Further changes in detail:

- Candera::Camera2DListener has been enhanced by new methods Candera::Camera2DListener::OnRenderTargetChanged() and Candera::Camera2DListener::OnCameraRenderStrategyChanged().
- Candera::CameraListener has been enhanced by new methods Candera::CameraListener::OnRenderTargetChanged() and

Candera::CameraListener::OnCameraRenderStrategyChanged().

- ClearMode is always enabled for Candera::Camera.
- Candera::SharedClearMode and Candera::SharedClearMode2D have been introduced.

See also:

Render Buffer Swapping and Clearing

Warping Orientation

The orientation of the display to use during warping can be set on the Display.

See also:

Display::SetWarpOrientation

Warping Image Bounds

In previous versions the warped image was not restricted to bitmap image bounds. The function Display::SetWarpImageBounds() has to be invoked before function Display::SetWarpMatrix() is called.

GraphicDeviceUnit Name

A name may be attached to a Graphic Device Unit.

See also:

GraphicDeviceUnit::SetName()

Candera AssetLoader

AssetDescriptor

AssetContext class has become deprecated and its functionality is taken by the new AssetDescriptor class.

Old Interface:

```
Candera::DefaultAssetProvider* provider = m_assetProvider;
Candera::AssetContext* assetContext = provider->GetAssetContext();
if (assetContext != 0) {
    for (Int i = 0; i < assetContext->GetAnimationCount(); i++) {
        const Candera::Char* animation = assetContext->GetAnimationName(i);
        //Use animation name
    }
}
```

New Interface in 3.0.0:

```
Candera::DefaultAssetProvider* provider = m_assetProvider;
for (Candera::AssetDescriptor::AssetIdIterator animationIdIterator =  provider->
GetAssetDescriptor().GetAssetIdIterator(AnimationLib
); animationIdIterator.IsValid(); ++animationIdIterator) {
    //use animation AssetId: *animationIdIterator
    //or use animation name: provider->GetNameById(AnimationLib, *animationIdIterator, 0)
}
```

AssetId

AssetId is a unique binary identifier that should be used to retrieve objects from the asset though the AssetProvider interface. AssetIds are supposed to replace the identification of objects by their name/path.

AssetIds should not be hard coded, but referenced through Symbolic Names. For more information see Accessing Items via AssetId. AssetIds can also be retrieved via the AssetDescriptor class iterators as shown in the AssetDescriptor example above.

Old Interface:

```
Candera::DefaultAssetProvider* provider = m_assetProvider;
Bitmap::SharedPointer bitmap = provider->GetBitmap(bitmapName);
```

New Interface in 3.0.0:

```
Candera::DefaultAssetProvider* provider = m_assetProvider;
 //copy AssetId directly from SceneComposer, but better use specified symbolic name or retrieve
AssetId assetId = CDA_LIBRARY_ASSETID(0x0, 0x100);
Bitmap::SharedPointer bitmap = provider->GetBitmapById(assetId);
```

Text Engine

Text Alignment

Enumeration type TextAlignment has been replaced by new enum types HorizontalAlignment and VerticalAlignment.

- HorizontalAlignment: HLeft, HCenter, HRight, HStretch
- VerticalAlignment: VTop, VCenter, VBottom, VStretch

Integrated FreeType 2.5

The following new options are available for configuration from CMake:

- FT_CONFIG_OPTION_NO_ASSEMBLER
- FT_CONFIG_OPTION_SYSTEM_ZLIB
- FT_CONFIG_OPTION_DISABLE_STREAM_SUPPORT
- FT_CONFIG_OPTION_USE_PNG
- FT_CONFIG_OPTION_PIC
- TT_CONFIG_OPTION_SUBPIXEL_HINTING
- TT_CONFIG_OPTION_UNPATENTED_HINTING All these options are disabled by default.

RgbColor deprecated

The class Candera::TextRendering::RgbColor has been deprecated. Candera::Color shall be used instead.

Text Measure Context

TextRenderer::GetBoundingRectangle is deprecated. Instead TextMeasureContext is available to provide similar functionality, in a more configurable manner. GlyphTextMeasureContext is a class provided to emulate the behavior of GetBoundingRectangle for ActualHeight alignment parameter. The rectangle returned by this class may be offset from the text position. To have the same alignment as before when rendering the text, the inverse of this offset needs to be applied. CursorTextMeasureContext is a class provided to emulate the behavior of GetBoundingRectangle

for ConstantHeight alignment parameter. The rectangle returned by this class does not usually have an offset and it contains the final advancement of the cursor. From the two rectangles, all values returned by GetBoundingRectangle may be computed.

Candera V3.0.0 Migration Guide

Following a guide how to uprade application code from V2.10.0 to V3.0.0 Candera Interfaces.

- Migration Guide for Backward Compatible Interface Changes
- Migration Guide for Breaking Interface Changes

Besides interface changes, also the build system has been changed significantly in V3.0.0:

- Candera/FeatStd Build System Changes

Regular Expression Search and Replace Patterns

Migration Guide for Backward Compatible Interface Changes

This page provides an overview about interface changes between Candera V2.10.0 and Candera V3.0.0, which are backward compatible.

Interface Changes with Deprecations

These changes do not require immediate adaptation of existing code. Although deprecated interfaces are still accessible, consider switching to the new interfaces as the deprecated interfaces will be removed with the next release.

Description	Candera V2.10.0	Candera V3.0.0

TextAlignment	TextAlignment
	`Leading`
	`Middle`
	`Trailing`
		HorizontalAlignment
		`HLeft`
		`HCenter`
		`HRight`
		
		VerticalAlignment
		`VTop`
		`VCenter`
		`VBottom`

Effect shared pointer types	[Effect]Ptr e.g. `BitmapBrushPtr`	[Effect]SharedPointer e.g. `BitmapBrush::SharedPointer`

const TreeTraverser	`class MyTraverser : public Tree`
	`protected:`
	`virtual TraverserAction`
	`{`
	`...`
	`}`
	`};`
		`class MyTraverser : public TreeTraverserBas`
		`protected:`
		`virtual TraverserAction ProcessNode`
		`{`
		`...`
		`}`
		`};`

AssetDescriptor	`Candera::DefaultAssetProvider`
	`* provider = m_assetProvider;`
	`Candera::AssetContext`
	`* assetContext = provider->GetA`
	`if (assetContext != 0) {`
	`for (Int i = 0; i < assetCo`
	`const Candera::Char* an`
	`//Use animation name`
	`}`
	`}`
		`Candera::DefaultAssetProvider`
		`* provider = m_assetProvider;`
		`for (`
		`Candera::AssetDescriptor::AssetIdIterator`
		`animationIdIterator =`
		`provider->GetAssetDescriptor().`
		`GetAssetIdIterator(AnimationLib);`
		`animationIdIterator.IsValid(); ++an`
		`//use animation AssetId: *animationIdIt`
		`//or use animation name: provider->GetN`

AssetId	`Bitmap::SharedPointer bitmap= Base::GetAssetProvider()->GetBitmap("MyModule#`	Assign a Symbol Name "myBmp" to the Bitmap Asset and export headers in SceneComposer as "MyAssets.h".
		`#include "MyAssets.h"`
		`Bitmap::SharedPointer bitmap = Base::GetAss`

Animation namespace	`Candera::InterpolationStrategy*`
	`Candera::LinearInterpolationStr`
	`();`
		`Animation::InterpolationStrategy::SharedPoi`
		`Animation::LinearInterpolationStrategy::Crea`
		`();`

New namespace CgiWidgetLibrary	no namespace required	`using namespace CgiWidgetLibrary;`

3D Shallow cloning with flat traversing	`Node* clone = node->Clone();`
	or
	`Node* clone = node->Clone(`
	`Node::Flat);`
		`Node* clone = node->Clone();`

3D Shallow cloning with deep traversing	`Node* clone = node->Clone(Node::Deep);`	`Node* clone = TreeCloner().CreateClone(*node);`

2D Shallow cloning with flat traversing	`Node2D* clone = node->Clone();`
	or
	`Node2D* clone = node->Clone(Node2D::Flat);`
	or
	`Node2D* clone = node->Clone(TraverseFlat, CloneShallow);`
		`Node2D* clone = node->Clone();`

Bitmap type/format changes	`UInt8* pixels = GetBitmapDataFr`
	`Bitmap* bitmap = &Bitmap(128, 1`
	`Bitmap::RgbaFormat`
	`, Bitmap::UnsignedByteType,`
	`Bitmap::PackAlignment1, pix`
	`true, true);`
		`UInt8* pixels = GetBitmapDataFromArbitrarySo`
		`Bitmap* bitmap = &Bitmap(128, 128,`
		`Bitmap::RgbaUnsignedBytePixelFormat,`
		`Bitmap::PackAlignment1, pixels, 0,0,`
		`true, true);`

GetBoudingRectangle with ExcludeFinalAdvance, ActualTransversalSize	`bound = textRenderer.GetBouding`
	`Measuri`
		`GlyphTextMeasureContext context;`
		`textRenderer.Render(context, layoutingOptio`
		`bound = context.GetTextRectangle();`

GetBoudingRectangle with IncludeFinalAdvance, ActualTransversalSize	`bound = textRenderer.GetBouding`
	`Measuri`
		`CursorTextMeasureContext cursorContext;`
		`GlyphTextMeasureContext glyphContext;`
		`glyphContext.SetNextContext(&cursorContext)`
		`textRenderer.Render(glyphContext, layouting`
		`glyphBound = glyphContext.GetTextRectangle(`
		`cursorBound = cursorContext.GetTextRectangl`
		`bound = Rectangle(`
		`glyphBound.GetLeft(),`
		`glyphBound.GetTop(),`
		`Math::Maximum(`
		`cursorBound.GetWidth() - glyphBound`
		`glyphBound.GetWidth()),`
		`glyphBound.GetHeight());`

GetBoudingRectangle with ExcludeFinalAdvance, ContantTransversalSize	`bound = textRenderer.GetBouding Measuri`	`CursorTextMeasureContext cursorContext; t GlyphTextMeasureContext glyphContext; e glyphContext.SetNextContext(&cursorContext) textRenderer.Render(glyphContext, layouting glyphBound = glyphContext.GetTextRectangle(cursorBound = cursorContext.GetTextRectangl bound = Rectangle(glyphBound.GetLeft(), cursorBound.GetTop(), glyphBound.GetWidth()), cursorBound.GetHeight());`
GetBoudingRectangle with IncludeFinalAdvance, ContantTransversalSize	`bound = textRenderer.GetBouding Measuri`	`CursorTextMeasureContext cursorContext; t GlyphTextMeasureContext glyphContext; e glyphContext.SetNextContext(&cursorContext) textRenderer.Render(glyphContext, layouting glyphBound = glyphContext.GetTextRectangle(cursorBound = cursorContext.GetTextRectangl bound = Rectangle(glyphBound.GetLeft(), cursorBound.GetTop(), Math::Maximum(cursorBound.GetWidth() - glyphBound glyphBound.GetWidth()), cursorBound.GetHeight());`

Extended Interfaces

These changes extend existing interfaces by adding new functionality and do not require any adaptation of existing code. However, custom implementations may be replaced by these library functions.

Description	Candera V2.10.0	Candera V3.0.0
3D Deep cloning with flat traversing	No previous support.	`Node* clone = node->Clone(); DeepNodeCloneStrategy cloneStr DeepNodeCloneStrategy::Pair pa cloneStrategy.Execute(pair, pa`

3D Deep cloning with deep traversing	No previous support.	`Node* clone = DeepTreeCloner()`
		
		or
		
		`TreeCloner treeCloner;`
		`DeepNodeCloneStrategy cloneStr`
		`treeCloner.SetNodeCloneStrateg`
		`Node* clone = treeCloner.Creat`

Header File Relocations

In CGI Studio 2.x, some functionality had been moved from Candera into FeatStd, keeping redirection macros and header files for backward compatibility. In CGI Studio 3.0.0, redirection code has been removed to a large extent. Following is an overview about macros and header files that will be removed.

Candera/System/Diagnostics Cleanup

> To ensure backward compatibility, macros and files formerly located in Candera/System/Diagnostics can still be accessed in CGI-Studio V3.0.0 . Please be aware that this access will be removed with the next release.

Header files from `Candera/Systems/Diagnostics` which only represented redirections to files located in FeatStd should no longer be used.
Please use respective files in `FeatStd/Diagnostics` and the namespace `Featstd::Diagnostics` instead.

Deprecated	Use instead
Candera/System/Diagnostics/Appender.h	FeatStd/Diagnostics/Appender.h
Candera/System/Diagnostics/CharBuffer.h	FeatStd/Diagnostics/CharBuffer.h
Candera/System/Diagnostics/ConsoleAppender.h	FeatStd/Diagnostics/ConsoleAppender.h
Candera/System/Diagnostics/Debug.h	FeatStd/Diagnostics/Debug.h
Candera/System/Diagnostics/DebuggerOutSystemMemoryStatistic.h	FeatStd/Diagnostics/DebuggerOutSystemMemoryStatistic.h
Candera/System/Diagnostics/ErrorHandling.h	FeatStd/Diagnostics/ErrorHandling.h
Candera/System/Diagnostics/FileAppender.h	FeatStd/Diagnostics/FileAppender.h
Candera/System/Diagnostics/LocationInfo.h	FeatStd/Diagnostics/LocationInfo.h
Candera/System/Diagnostics/LogControl.h	FeatStd/Diagnostics/Log.h
Candera/System/Diagnostics/LogEvent.h	FeatStd/Diagnostics/LogEvent.h

Candera/System/Diagnostics/Logger.h	FeatStd/Diagnostics/Logger.h
Candera/System/Diagnostics/LoggerTyped.h	FeatStd/Diagnostics/LoggerTyped.h
Candera/System/Diagnostics/LogLevel.h	FeatStd/Diagnostics/LogLevel.h
Candera/System/Diagnostics/Measurable.h	FeatStd/Diagnostics/Measurable.h
Candera/System/Diagnostics/SystemMemoryStatistic.h	FeatStd/Diagnostics/SystemMemoryStatistic.h

Similarly, all macros contained in files located in `Candera/System/Diagnostics` which only served as redirection to macros located in FeatStd should not be used any more.

Deprecated	Use instead
CANDERA_DEBUG_BREAK	FEATSTD_DEBUG_BREAK
CANDERA_DEBUG_ASSERT	FEATSTD_DEBUG_ASSERT
CANDERA_DEBUG_FAIL	FEATSTD_DEBUG_FAIL
CANDERA_DEBUG_REENTRANCE_GUARD	FEATSTD_DEBUG_REENTRANCE_GUARD
CANDERA_COMPILETIME_ASSERT	FEATSTD_COMPILETIME_ASSERT
CANDERA_UNUSED_PARAMETER	FEATSTD_UNUSED
CANDERA_PANIC	FEATSTD_PANIC
CANDERA_PANIC_IF	FEATSTD_PANIC_IF
CANDERA_LOG_FUNC	FEATSTD_LOG_FUNC
CANDERA_LOG_LOCATION	FEATSTD_LOG_LOCATION
CANDERA_LOG_SET_REALM	FEATSTD_LOG_SET_REALM
CANDERA_LOG_DEBUG	FEATSTD_LOG_DEBUG
CANDERA_LOG_INFO	FEATSTD_LOG_INFO
CANDERA_LOG_WARN	FEATSTD_LOG_WARN
CANDERA_LOG_ERROR	FEATSTD_LOG_ERROR
CANDERA_LOG_FATAL	FEATSTD_LOG_FATAL
CANDERA_LOG_REALM	FEATSTD_LOG_REALM
CANDERA_LOG	FEATSTD_LOG

Migration Guide for Breaking Interface Changes

These changes are incompatible with earlier versions of Candera and will require immediate adaptation of existing code.

Interface Changes without Deprecations

Description	Candera V2.10.0	Candera V3.0.0

Shared pointers in animation framework	`AnimationController* animController`
	`AnimationController::Create();`
	`AnimationTimeDispatcher* animDispatc`
	`AnimationTimeDispatcher::Create();`
	`InterpolationStrategy* interpolation`
	`SplineInterpolationStrategy::Create();`
	`BackEaseFunction backEase;`
		`AnimationController::SharedPointer animCo`
		`AnimationController::Create();`
		`AnimationTimeDispatcher::SharedPointer ar`
		`AnimationTimeDispatcher::Create();`
		`InterpolationStrategy::SharedPointer inte`
		`SplineInterpolationStrategy::Create();`
		`BackEaseFunction::SharedPointer backEase`
		`BackEaseFunction::Create();`

Shared pointers and Create methods for animation property setters instead of constructors	e.g.
	`Candera::Transformable2DTranslateYPr`
	`Candera::MaterialPropertySetter mate`
		e.g.
		`Candera::Transformable2DTranslateYPropert`
		`Transformable2DTranslateYPropertySetter::`
		`();`
		`Candera::MaterialPropertySetter::SharedPo`
		`MaterialPropertySetter::Create();`

Shared pointer destruction	`Bitmap* bmp = CANDERA_NEW(...);`
	`CANDERA_DELETE(bmp);`
	`bmp = 0;`
		SharedPointers may not be manually disposed, freed or deleted. Additionally SharedPointers cannot be set to NULL in the same way as pointers. According code has to be removed:
		`Bitmap::SharedPointer bmp =`
		`Bitmap::Create(...);`
		`// no manual destruction`
		`// no need to set to zero / NULL`

Bitmap as SharedPointer	`Bitmap bitmap(...);`	`Bitmap::SharedPointer bitmap =` `Bitmap::Create(...);`

BitmapTextureImage, CubeMapTextureImage, BitmapImage2D setters take SharedPointer to Bitmap and no disposer function	`BitmapTextureImage textureImage =`
	`BitmapTextureImage::Create();`
	`Bitmap bitmap = CANDERA_NEW(Bitmap)(`
	`textureImage->SetBitmap(bitmap, 0);`
		`BitmapTextureImage textureImage =`
		`BitmapTextureImage::Create();`
		`Bitmap::SharedPointer bitmap =`
		`Bitmap::Create(...);`
		`textureImage->SetBitmap(bitmap);`

Header File Relocations

In CGI Studio 2.x, some functionality had been moved from Candera into FeatStd, keeping redirection macros and header files for backward compatibility. In CGI Studio 3.0.0, redirection code has been removed to a large extent. Following is an overview about removed macros and header files.

CanderaPlatform/OS Cleanup

Removed	Use instead
CanderaPlatform/OS/FixedPoint	FeatStd/Platform/FixedPoint
CanderaPlatform/OS/FractionNumberMath.h	FeatStd/Platform/FractionNumberMath.h
CanderaPlatform/OS/FractionNumber.h	FeatStd/Platform/FractionNumber.h
CanderaPlatform/OS/CodePointIterator.h	FeatStd/Platform/CodePointIterator.h

Candera TimePlatform

Removed	Use instead
Candera/Time.h	CanderaPlatform/OS/TimePlatform.h

Candera/FeatStd Build System Changes

Library Changes

Candera base feature libraries have been joined into the new common library called **Candera.lib**. The following libraries from CGI-Studio V2.x no longer exist separately:

- CanderaEngineBase.lib
- CanderaEngine2D.lib
- CanderaEngine3D.lib
- CanderaGlobalization.lib
- CanderaSystem.lib
- CanderaOsWin32.lib
- CanderaWidget.lib

CMake Switch Name Modification

The names of almost all CMake switches used in CGI-Studio V2.x have been revised to improve effect and feature association. The feature name now reflects, to which component the feature belongs to.

> Please be aware, that even though the CMake Switch names and preprocessor defines from CGI-Studio V2.x still work in CGI-Studio V3.0.0, this backward compatibility is going to be removed with the next release.

Candera CMake Switch Changes

- For detailed information on Candera configuration please see Application Development > Build System Setup > Building Candera.
- For Candera compiler defines refer to Candera Configuration.

V2.10.0	V3.0.0
CGIFEATURE_ENABLE_CANDERA_2D	CANDERA_2D_ENABLED
CGIFEATURE_ENABLE_CANDERA_3D	CANDERA_3D_ENABLED
CGIFEATURE_VRAM_ALLOCATOR_CHECK	CANDERA_VRAM_ALLOCATOR_CHECK_ENABLED
CGIFEATURE_VRAM_ALLOCATOR_STATISTICS	CANDERA_VRAM_ALLOCATOR_STATISTICS_ENABLED
CGIFEATURE_ENABLE_RENDER_STATE_CACHING	CANDERA_RENDER_STATE_CACHING_ENABLED
CGIFEATURE_ENABLE_GLOBALIZATION	CANDERA_GLOBALIZATION_ENABLED
CGIFEATURE_ENABLE_ASSET_DECOMPRESSION	CANDERA_ASSET_DECOMPRESSION_ENABLED
CGIFEATURE_TEXTSHAPER	CANDERA_TEXT_SHAPER
CGIFEATURE_ENABLE_BIDIRECTIONAL_TEXT	CANDERA_BIDIRECTIONAL_TEXT_ENABLED
CGIFEATURE_ENABLE_MULTILINE_TEXT	CANDERA_MULTILINE_TEXT_ENABLED
CGIFEATURE_ENABLE_VIDEO_MEMORY_STATISTIC FEATSTD_ENABLE_VIDEO_MEMORY_STATISTIC	CANDERA_VIDEO_MEMORY_STATISTIC_ENABLED
CGIUNITTEST_ENABLED	CANDERA_UNITTEST_ENABLED
CGIUNITTEST_INCLUDE_ISOLATED	CANDERA_UNITTEST_INCLUDE_ISOLATED
CGIDEVICE_ENABLE_4BIT_GLYPH_CACHE	CANDERA_4BIT_GLYPH_CACHE_ENABLED
CGIFEATURE_ENABLE_LOG	FEATSTD_LOG_ENABLED
CGIFEATURE_ENABLE_MEMORYPOOL	FEATSTD_MEMORYPOOL_ENABLED
CGIFEATURE_ENABLE_TEXTENGINE_MEMORYPOOL	CANDERA_TEXTENGINE_MEMORYPOOL_ENABLED
CGIFEATURE_ENABLE_ASSETLOADER_MEMORYPOOL	CANDERA_ASSETLOADER_MEMORYPOOL_ENABLED
CGIFEATURE_ENABLE_MEMORY_STATISTIC	FEATSTD_SYSTEM_MEMORY_STATISTIC_ENABLED
CGIFEATURE_SYSTEM_MEMORY_STATISTIC_FILE_AND_LINE _TRACKING	FEATSTD_SYSTEM_MEMORY_STATISTIC_FILE _AND_LINE_TRACKING_ENABLED
CGIFEATURE_FRACTIONAL_NUMBER_TYPE	FEATSTD_FRACTIONAL_NUMBER_TYPE
CGIFEATURE_ENABLE_MONITOR	FEATSTD_MONITOR_ENABLED

CGISTUDIO_BUILD_FREETYPE	Has been removed, Freetype will be included automatically.

FeatStd CMake Switch Changes

- Refer to the FeatStd 3.0.0 CMake Changelog to learn about FeatStd V3.0.0 CMake switch modifications.
- For FeatStd compiler defines refer to FeatStd Configuration.

Regular Expression Search and Replace Patterns

The following Regular Expression Search & Replace patterns may be used to quickly apply changes to existing source code. They aim at fixing many (breaking) changes introduced with the new version of Candera. However, as simple text replacement with regular expressions is used, corrections which resulted from applying these patterns may be incorrect in some situations. Therefore, the patterns are just provided as-is in the hope that they may be useful for code migration.

To apply the patterns Notepad++ or any other compatible Regular Expression Search & Replace tool can be used. When using Notepad++ please make sure to select the search method "Regular expression" and check "Match case".

Patterns for Fixing Breaking Changes

Use case	Search pattern	Replace pattern
Declarations of AnimationPlayer pointer	(\\S*)SharedPointer<AnimationPlayer>	AnimationPlayer::SharedPointer
Declarations of AnimationGroupPlayer pointer	(\\S*)SharedPointer<AnimationGroupPlayer>	AnimationGroupPlayer::SharedPointer
Declarations of AnimationTimeDispatcher pointer variables	((\\W)|(const\\s+))(Candera::)?AnimationTimeDispatcher((\\s*)*)(\\s*?[\\w:]+)(\\s*?=\\s*?(NULL|0))?	$2$4AnimationTimeDispatcher::SharedPointer$6$7
Declarations of AnimationController pointer variables	((\\W)|(const\\s+))(Candera::)?AnimationController((\\s*)*)(\\s*?[\\w:]+)(\\s*?=\\s*?(NULL|0))?	$2$4AnimationController::SharedPointer$6$7
Declarations of InterpolationStrategy pointer variables	((\\W)|(const\\s+))(Candera::)?InterpolationStrategy ((\\s*)*)(\\s*?[\\w:]+)(\\s*?=\\s*?(NULL|0))?	$2$4InterpolationStrategy::SharedPointer$6$7

Calls to AnimationKeyframeSequence::SetInterpolationStrategy()	SetInterpolationStrategy\\(\\s*?&?\\s*?([\\w:\\(\\)]+),.*?\\)	SetInterpolationStrategy \\($1\\)
Calls to AnimationPlayer::SetController()	SetController\\(\\s*?&?\\s*?([\\w:\\(\\)]+),.*?\\)	SetController\\($1\\)
Declarations of Animation PropertySetter pointer variables	((\\W)|(const\\s+))(Candera::)?(\\w*)PropertySetter((\\s*)*?)(\\s*?\\w+)(\\s*?=\\s*?(NULL|0))?	$2$4$5PropertySetter::SharedPointer$7$8
Creation of Animation PropertySetter objects	(CANDERA_NEW|FEATSTD_NEW)(\\s*?\\(\\s*?)(Candera::)?([\\w:]*)PropertySetter(\\s*\\))	$3$4PropertySetter::Create\\(\\)
Calls to AnimationBlendedProperty::SetAnimationPropertySetter()	SetAnimationPropertySetter\\(\\s*?&?\\s*?([\\w:\\(\\)]+)\\)	SetAnimationPropertySetter\\($1\\)
Forward declarations of classes for pointer declarations which are now SharedPointer instances	(namespace Candera.*\\{.*)([\\r\\n\\s\\w;]*)(class AnimationController;.*[\\r\\n])	#include <Candera/EngineBase/Animation/AnimationController.h>\\n\\n$1
Forward declarations of classes for pointer declarations which are now SharedPointer instances	(namespace Candera.*\\{.*)([\\r\\n\\s\\w;]*)(class AnimationTimeDispatcher;.*[\\r\\n])	#include <Candera/EngineBase/Animation/AnimationTimeDispatcher.h>\\n\\n$1
Declarations of Bitmap pointer variables	((\\W)|(const\\s+))(Candera::)?Bitmap((\\s*)*)(\\s*?\\w+)(\\s*?=\\s*?(NULL|0))?	$2$4Bitmap::SharedPointer$6$7
Forward declarations of classes for pointer declarations which are now SharedPointer instances.	(namespace Candera.*\\{.*)([\\r\\n\\s\\w;]*)(class Bitmap;.*[\\r\\n])	#include <Candera/EngineBase/Common/Bitmap.h>\\n\\n$1
Calls to e.g. BitmapImage2D::SetBitmap()	SetBitmap\\(\\s*?&?\\s*?([\\w:\\(\\)]+),.*?\\)	SetBitmap\\($1\\)
Calls to CubeMapTextureImage::SetBitmapFace()	SetBitmapFace\\(\\s*?&?\\s*?([\\w:\\(\\)]+),(.*?),.*?\\)	SetBitmapFace\\($1,$2\\)
Use CanderaPlatform/OS/TimePlatform.h instead of Candera/Time.h	(#include <Candera\\/Time\\.h>)	#include <CanderaPlatform\\/OS\\/TimePlatform\\.h>

Use CanderaPlatform/OS/TimePlatform.h instead of Candera/Time.h	([\\s=\\n\\(<])Time::	$1TimePlatform::
Use headers in FeatStd, that were previously located in CanderaPlatform/OS	#include <CanderaPlatform\\/OS\\/Diagnostics\\/(FixedPoint(.*)|FractionNumberMath.h|FractionNumber.h|CodePointIterator.h)>	#include <FeatStd\\/Platform\\/$1.h>

Patterns for fixing backward compatible changes

Use case	Search pattern	Replace pattern
Usage of EaseFunction qualifier	([^(\\w|(Animation::)|\\/|(enum)|(class))]|\\(|(Candera::)|\\s)((Back|Bounce|Elastic|Power|Exponential)EaseFunction)	$1Animation::$3
Usage of InterpolationStrategy qualifier	([^(\\w|(Animation::)|\\/|(enum)|(class))]|\\(|(Candera::)|\\s)((Bezier|Ease|Linear|Spline|Step)?InterpolationStrategy)	$1Animation::$3
Usage of Animation core classes qualifier	([^(\\w|(Animation::)|\\/|(enum)|(class))]|\\(|(Candera::)|SharedPointer<|\\s)(Animation(BlendedProperty|Controller|GroupPlayer|KeyframeSequence|Player|PlayerBase|PlayerListener|PropertySetter|TimeDispatcher|TimeType))(\\W)	$1Animation::$3$5
Usage of Animation core classes qualifier	([^(\\w|(Animation::)|\\/|(enum)|(class))]|\\(|(Candera::)|\\s)(AbstractEasingFunction|CameraGroupAnimationPropertySetter|KeyframeSequence|SequenceTimeType)	$1Animation::$3
New namespace CgiWidgetLibrary	public\\s+(Listener)\\s*?<	public CgiWidgetLibrary::$1<
Removed indirection from Candera to FeatStd for Logging	CANDERA_DEBUG_(\\w*)\\(	FEATSTD_DEBUG_$1\\(
Removed indirection from Candera to FeatStd for Logging	CANDERA_LOG(\\w*)\\(	FEATSTD_LOG$1\\(
Removed indirection from Candera to FeatStd for Logging	([^:\\/\\w]|\\s)(Log(Control|Level))	$1FeatStd::Diagnostics::$2
Removed indirection from Candera to FeatStd	CANDERA_PANIC(\\w*)\\(	FEATSTD_PANIC$1\\(
Removed indirection from Candera to FeatStd	CANDERA_COMPILETIME_ASSERT	FEATSTD_COMPILETIME_ASSERT
Removed indirection from Candera to FeatStd	CANDERA_UNUSED_PARAMETER	FEATSTD_UNUSED

Removed indirection from Candera to FeatStd	#include <Candera\\/Systems\\/Diagnostics\\/(Appender|CharBuffer|ConsoleAppender|Debug|DebuggerOutSystemmemoryStatistic|ErrorHandling|FileAppender|LocationInfo|Measurable|SystemMemoryStatistic|Log(ger(Typed)?|Event|Level|))(.h|Control.h)>	#include <FeatStd\\/Diagnostics\\/$1.h>
Removed indirection from Candera to FeatStd	#include <FeatStd\\/Diagnostics\\/LogRealm.h>	#include <FeatStd\\/Diagnostics\\/Log.h>

Courier V3.0.0

3.0.0-1 Courier CMake Switches

Note:

For detailed information on Courier configuration please see Application Development > Courier > Tutorial 5 > Build Process

V2.10.0	V3.0.0
COURIER_ENABLE_ENHANCED	COURIER_ENHANCED_ENABLED
COURIER_ENABLE_IPC	Has been unified with Featstd IPC switch. FEATSTD_IPC_ENABLED
COURIER_ENABLE_MESSAGING_MONITOR	COURIER_MESSAGING_MONITOR_ENABLED
COURIER_ENABLE_RENDERING_MONITOR	COURIER_RENDERING_MONITOR_ENABLED
COURIER_ENABLE_LOG	FEATSTD_LOG_ENABLED

Note:

Refer to the Candera 3.0.0 CMake Changelog to learn about Candera V3.0.0 CMake switch modifications.

Refer to the FeatStd 3.0.0 CMake Changelog to learn about FeatStd V3.0.0 CMake switch modifications.

3.0.0-2 Data Binding Changes

The following changes have been made:

- Bugfix: Bindable properties of widgets were wrong displayed in SceneComposer. The problem has been solved, by replacing the function bound global string variable with a function bound global map that maps the meta info property instance to the corresponding editor string.
- Bugfix: Data binding was no longer working correctly on target builds with O2 optimization enabled. For this, Courier::DataBindingPrivate::DataItemDescriptor structure was modified.

3.0.0-3 Visualization Changes

The following changes have been made:

- Composite Group support was added. For this, the following has been modified:
  - Courier::CompositePath class was added;
  - New *viewId* and *compositePath* parameters added for Courier::IViewHandler::ExecuteAnimationAction, Courier::IViewHandler::FindWidget, Courier::IViewHandler::SetWidgetProperty and Courier::IViewHandler::SetFocus methods;
  - New *compositePath* parameters added for Courier::View::GetFrameworkWidget, Courier::View::GetAnimation, Courier::ViewContainer::GetFrameworkWidget, Courier::ViewContainer::GetAnimation and Courier::ViewScene::GetFrameworkWidget methods;
  - New *viewId* and *compositePath* parameters added for Courier::AnimationReqMsg, Courier::AnimationResMsg and Courier::AnimationIndMsg;
  - New *compositePath* parameters added for Courier::SetPropertyReqMsg, Courier::SetPropertyResMsg, Courier::SetFocusReqMsg, Courier::SetFocusResMsg, Courier::LostFocusIndMsg and Courier::WidgetMsgBase messages.

- Added support for automatic upload of Owner Render Targets. For this, the following has been modified:
  - Courier::FrameworkWidget::WakeUpAllRenderComponents method added;
  - *layerIndex* parameter added to Courier::Gdu constructor;
  - New Courier::Gdu::OwnerState enum added for 3D render targets;
  - Courier::Gdu::GetLayerIndex public method added;
  - Courier::Gdu::GetOwnerState, Courier::Gdu::OnTransformChange and Courier::Gdu::OnNodeRemoved public methods were added for 3D render targets;
  - Courier::Renderer::WakeUpAllRenderComponents method was added.

- Courier adaptation needed for new Candera feature related to FreeType streaming . For this, Courier::ViewHandler::SetFontStoreProviderCallback and Courier::ViewHandler::GetDefaultFontStore were added.

- Bugfix: Fix added for animation invalidate mechanism to correctly invalidate related scenes. For this, the following has been modified:
  - The enum Courier::AnimationInvalidator::Setter has been modified;
  - Scene3DPtrVector and Scene2DPtrVector typedefs were removed.

- Bugfix: interface changed for Courier::ViewHandler::Init() method to accept every kind of Candera::AssetConfig. For this, the following has been modified:
  - *assetConfiguration* parameter type changed from Courier::AssetConfiguration* to Candera::AssetConfig* for Courier::AssetAccessor::Init and IViewHandler::Init method.

- Bugfix: Fixed wrong touch input handling on overlay surfaces. For this, the following has been modified:
  - *pointerId* and *sourceId* parameters have been added to Courier::TouchInfo constructor.

- Bugfix: Fixed issue related to wrong loading of scene's content for Courier::LoadReqMsg. For this, the following has been modified:
  - The Courier::LoadReqMsg will provide the same behavior as before to keep backward compatibility. A new Courier::TryLoadReqMsg and Courier::TryAsyncLoadReqMsg were introduces. Courier::TryLoadReqMsg creates the render target if the layer is not already used. Otherwise nothing is done for the render target. Also, Courier::TryLoadReqMsg tries to upload the scene (this may fail if there is currently no EGL context available - in this case a normal Courier::LoadReqMsg should be sent before activating the view). The new Courier::TryAsyncLoadReqMsg will have the same behavior as Courier::TryLoadReqMsg after the scene context if fully loaded;
  - A new *forceEnable* parameter has been added to Courier::Renderer::EnableLayer method;
  - A new *forceUpload* parameter has been added to Courier::View::LoadContent, Courier::ViewContainer::LoadContent, Courier::ViewHandler::SetCurrentLoadingViewScene, Courier::ViewScene::PartialLoad , Courier::ViewScene::AsyncLoadContent, Courier::ViewScene2D::LoadContent, Courier::ViewScene2D::PartialLoad and Courier::ViewScene2D::AsyncLoadContent, Courier::ViewScene3D::LoadContent, Courier::ViewScene3D::PartialLoad and Courier::ViewScene3D::AsyncLoadContent methods.

- Bugfix: No proper clean was done when active transitions received Courier::ShutdownMsg message. Courier::TransitionHandler::Finalize method has been added. This method will first finish active transitions and second destroy all created transitions to bring the transition handler into a finalized state. The Courier::TransitionHandler::Finalize method

will finalize the Courier::TransitionHandler (if set), stop all animations and finalize the views. Handling of transition errors has also been added. For this, Courier::Transition::TransitionError enum, Courier::Transition::SetLastTransitionError method and Courier::Transition::GetLastTransitionError method were introduced.

- Bugfix: Wrong dependent views' invalidation. The new interface allows to add dependency views to another view for one camera or for all cameras using Courier::View::AddInvalidationDependency method. To remove the dependency Courier::View::RemoveInvalidationDependency can be used for one camera dependency or for all camera dependencies (it always has to be called in a symmetric way to the Add call). Also, Courier::View::GetFirstInvalidationDependency was introduced.

- Bugfix: no means to retrieve an animation player for a given animation ID. For this Courier::ViewHandler::GetAnimationPlayer was made public.

- Bugfix: Wrong sorting of widgets' vector. The default behavior can be changed by calling the new Courier::ViewScene::SetWidgetSortingStrategy method. Courier::ViewScene::GetWidgetSortingStrategy returns the current set strategy.

- Bugfix: Too long processing time for Courier::ViewReqMsg. For this, *initContent* parameter was added to Courier::ViewReqMsg message for Create and CreateAll actions. This member should be set to false to delay the initialization of the scene from the first loading of the scene.

3.0.0-4 AddOns Changes

The following changes have been made:

- Bugfix: Fixed render block issue due to Wayland input handler. For this, Courier::InputHandling::WaylandContext::GetEventQueue method was added;
- Bugfix: Courier::TouchHandling::TouchSession::GetFrameworkWidgets() not accessible for derived classes. To fix this, Courier::TouchHandling::TouchSession::GetFrameworkWidgets() was moved in the protected area;
- Bugfix: TouchSession is not working. For this, the down state management was replaced with a more dynamic approach that also considers non-linear source Ids. cCOURIER_DEFAULT_MAX_TOUCHPOINTER_COUNT was set to 10 in Courier::TouchHandling class and Courier::TouchHandling::TouchSessionBase::InvalidSourceId const was declared to indicate a invalid surface Id.

FeatStd V3.0.0

FeatStd CMake Switches

Note:

For detailed information on FeatStd configuration please see Application Development > FeatStd > FeatStd Features

V2.10.0	V3.0.0
CGIFEATURE_ENABLE_LOG has been unified with FEATSTD_ENABLE_LOG	FEATSTD_LOG_ENABLED
CGIFEATURE_ENABLE_MEMORYPOOL has been unified with FEATSTD_ENABLE_MEMORYPOOL	FEATSTD_MEMORYPOOL_ENABLED
CGIFEATURE_ENABLE_MEMORY_STATISTIC has been unified with FEATSTD_ENABLE_SYSTEM_MEMORY_STATISTIC	FEATSTD_SYSTEM_MEMORY_STATISTIC_ENABLED
CGIFEATURE_SYSTEM_MEMORY_STATISTIC_FILE_AND_LINE_TRACKING has been unified with FEATSTD_ENABLE_SYSTEM_MEMORY_STATISTIC_FILE_AND_LINE_TRACKING	FEATSTD_SYSTEM_MEMORY_STATISTIC_FILE _AND_LINE_TRACKING_ENABLED
CGIFEATURE_FRACTIONAL_NUMBER_TYPE has been unified with FEATSTD_FRACTIONAL_NUMBER_TYPE	FEATSTD_FRACTIONAL_NUMBER_TYPE
CGIFEATURE_ENABLE_MONITOR has been unified with FEATSTD_ENABLE_MONITOR	FEATSTD_MONITOR_ENABLED
COURIER_ENABLE_IPC has been unified with FEATSTD_ENABLE_IPC	FEATSTD_IPC_ENABLED
FEATSTD_ENABLE_THREADSAFETY	FEATSTD_THREADSAFETY_ENABLED
FEATSTD_ENABLE_UNITTEST_FRAMEWORK	FEATSTD_UNITTEST_FRAMEWORK_ENABLED
FEATSTD_ENABLE_DLT	FEATSTD_DLT_ENABLED

Note:

Refer to the Candera 3.0.0 CMake Changelog to learn about Candera V3.0.0 CMake switch modifications.

FeatStd Diagnostics

MemoryAppender added

MemoryAppender is added which allows logging events to a buffer in memory.

FeatStd Memory Management

Removed Sharing Object

The outdated and orphaned file SharingObject.h has been removed.

Retain Count atomic and uncopyable

The retain count of shared objects - enabled by macro "FEATSTD_SHARED_POINTER_DECLARATION" - has been made atomic to make increments and decrements thread-safe. Additionally the retain count has been made uncopyable to avoid copying the retain count when cloning shared objects.

FeatStd Memory Pool

Enforce Initialization of BackingHeap

Initialization of BackingHeap (MallocBackingHeap) is enforced. BackingHeap initializations are reference counted and missing initialization will assert. This is to detect missing initialization early, even if MallocBackingHeap does not require Init (but switching to other BackingHeap will cause crashes then). A new parameter in BackingHeap::Destroy() / MallocBackingHeap::Destroy() forces the destruction of the heap (even if init counter is larger 1).

FeatStd Monitor

Monitor module migrated from Candera

The Candera module CanderaMonitor has been migrated to FeatStd.

See also:

FeatStd CMake Switches for the changed CMake feature flag related to Monitor functionality.

FeatStd Util

String Ids for translatable Text

A new class TextId has been added for accessing translatable texts from a string.

See also:

String::String(const TextId& id)

SceneComposer V3.0.0

Candera Related Features

Array Editor

Array properties are now supported for dynamic items (widget, effect, render target and layouter). Special editors are available in properties panel for changing the value.

See also:

- Candera::ArrayProperty

Candera Driven Validation

A new feature was added which enables Candera engine to validate the content from SceneComposer. For each dynamic property of widgets, effects, render targets or layouters, Candera engine can generate a problem message (error or warning). This message is displayed in the problems panel and also as a tooltip in the properties panel for the item on which the property is set.

If an item has properties with Error problems, the asset is not generated for it.

See also:

- Candera::DynamicProperties::ValidateValue

Camera Groups

The purpose of this work package is to support the creation of camera groups and to offer extended support for display visualization by enabling/disabling camera groups. In Candera, a CameraGroups is a class containing a list of Camera2D and Camera3D objects.

The camera groups were added and they allow the user to group together cameras from the entire solution. The camera groups are displays in a tab or the render targets explorer panel.

Widgets can be associated to camera groups (widget property of type CameraGroup* is supported). However, the widget should not modify the state of the camera group (shall not add or remove cameras)

See also:

- Camera Group in SceneComposer User Manual
- Candera::CameraGroup

Composite Animations

The Composite Animation was Removed. An animation can now contain animated properties for composites and scenes (they can also be mixed).

See also:

- Add Animated Property in SceneComposer User Manual

Clear Mode

A new item of type "Clear Mode" can be created in SceneComposer and referenced from any render target.

See also:

- Clear Mode Item in SceneComposer User Manual
- Candera::ClearMode

Items AssetId

All items in a solution have a type and a name which identifies them in their parent container and a full name which uniquely identifies them in the entire solution (for example a mesh in a scene could have the full name /Global/Scenes/MyScene/Group:RootNode/Mesh:MyMesh).

Similar to the full name which identifies the items in the solution, for Candera there is a CanderaPath which uniquely identifies an item in an asset file (for example the mesh mentioned above will be referenced in Candera using the path /Scene:Global::Scenes::MyScene/Group:RootNode/Mesh:MyMesh). Such a string will have to be stored in the asset library for animations or widgets which reference scene nodes. Also it will be processed every time the scene is loaded.

> Candera does not have the concept of solution folders, so in order to distinguish between items with the same name but located in different solution folders the CanderaName

> property of the items contain also the name of the folders (for example the scene /Global/Scenes/MyScene will have the CanderaName Global::Scenes::MyScene).

To improve the performance we introduced another way to identify items using numerical Ids instead of strings.

- **Symbolic Names.** This type of names can be assigned to items to make their Id accessible from applications. A C++ header with all the symbolic names from the solution can be generated via File->Export Symbolic Names.

See also:

- Items AssetId in SceneComposer User Manual

EGL Context Sharing

A Context Resource Pool (CRP) is used to share resources among different Context Providers. It allows grouping of render targets and associated device objects (Shaders, Texture Images, Vertex Buffers, etc.) in a common pool.

Each platform has some rules and restrictions related to CRPs, for example on some platform multiple displays can share the same CRP, while on some other platform a CRP is required for each Window Surface. SceneComposer displays the relationship between CRPs, Displays, Window Surfaces and Frame Buffer Objects in the Render Target Explorer panel (main view and an additional CRP view). Depending on the platform, the texture render targets (frame buffer objects) may have an Owner property of type render target reference or CRP.

If the type of the Owner property is render target reference, the owner must be a display render target (window surface).

If the type of the Owner property is CRP (integer value in SceneComposer), the possible values will be obtained from the available displays.

If the Owner property is available for a render target but it is not set by the user, the render target will be considered orphan.

See also:

- Context Resource Pool in SceneComposer User Manual
- Candera::ContextResourcePool

BitmapFormat and BitmapType

These properties were made obsolete and their value were merged into PixelFormat which contains only the valid combinations of the former properties.

SceneComposer Usability Improvements

Reusable Solution Library

The main goal of a shareable component library (SCL) solution is to make all the resources from a particular SCS solution available to be used in other solutions. SceneComposer provides the possibility to reuse any part from a solution that is exported as a SCL.

See also:

- Reusable Solution Library in SceneComposer User Manual

Appearance Wrapper

In some circumstances it could be necessary to expose for the user just those properties which should be modified to get a desired setting of an object. The Appearance Wrapper feature allows to expose only some properties of an appearance attached to a node or the properties pertaining to its containments (material, render node, shader, uniform setter, texture).

See also:

- Appearance Wrapper in SceneComposer User Manual

Shader Details

Shader requirements are now available in SceneComposer. This includes the number of lights, materials and textures. They are displayed in a new tab "Details" in the Property Grid, when a shader program is selected. The tab also contains the available attributes and uniforms.

See also:

- Shader Editor in SceneComposer User Manual

Problem Viewer Extensions

Due to the huge quantity of informations which can be displayed on the Problems browser, it is possible to suppress warnings or messages solely related to imported content.

To get focused on problems related to SceneComposer solution configuration, many changes were made related to problem management:

- The problems configuration panel is available from the problem browser via "Configure problems" button.
- Now it is possible to ignore Warning/Info problems for an item or for all descendants of a folder (these are user suppression flags stored in user data file outside of the solution).
- Now it is possible to ignore a specific Warning/Info problem for entire solution (these are global suppression flags stored in the solution).
- The items which have Error problems can not be rendered and it is also not possible to generate an asset library for them.

See also:

- Problems Browser in SceneComposer User Manual

Bitmap Usage Validation

Custom formats available for each bitmap converter will have an additional property to specify for which engine are they supported (Candera2D, Candera3D, Mixed). This information is displayed in the properties panel of the BitmapProfile used by the Bitmap (see property CanderaModelType). During validation of the solution, if CanderaModelType of the bitmap profile used by the bitmap does not match the texture, effect, widget, a warning will be displayed (Mantis 4461).

See also:

- Bitmap Profile in SceneComposer User Manual

Activate Referenced Items

The possibility to activate items referenced through paths has been added. The Activate button will load the item in the property grid and in others panels linked to the type of the item (Mantis 4621).

See also:

- Activation of Items Referenced in SceneComposer User Manual

---

Uniform Auto Activation

Uniform setters contain some flags used to enable the activation of various auto uniforms (camera position, lights, material, texture, etc.) which are defined in shader programs.

A new property was added to uniform setters:

- **Enable Auto Activation.** This property becomes available just if the "Enable uniform setter auto activation" flag (Solution Options>General Configuration panel) is properly set. After this operation, a check box with the same name will become available in the Properties panel. This property specifies if the auto activation of the flags should be enabled based on the global values stored in the solution properties or locally in this item (Mantis 4657).

See also:

- Generic ShaderParamSetter in SceneComposer User Manual
- Candera::GenericShaderParamSetter

Multi Materials Import from FBX

This feature enables the user to import FBX objects that have multiple materials (Multi/Sub object) associated to the geometry (For example different polygons are associated different materials). As Candera does not support multi-material objects, an object that has a multi-material is split into separate objects in which the polygons share the same material. Therefore, such an object will be imported as a group containing the sub-meshes that sum up to the FBX object.

See also:

- How to Import Resources in SceneComposer User Manual

Auto-hide in Selected Panels

Auto-hide is offered in selected panels, similar to VisualStudio. Also available are the following options: Float, Dock as Tabbed Document, Hide and Close.

See also:

- SceneComposer GUI in SceneComposer User Manual

Perspective Support

It is now possible to save a new perspective layout, which will be based on the current perspective. Also the possibility to delete, to export and to import a perspective was added.

See also:

- SceneComposer GUI in SceneComposer User Manual

Scene Templates

Scene templates are now supported in SceneComposer. A scene created by the user can be made template by checking its IsTemplate property. The dialogue used to create new scenes was modified to allow the user to select an existing template or import.

See also:

- Scene Templates in SceneComposer User Manual

Import Fonts Button

A new "Import Fonts" button was added in the Text Style Palette section from the Solution Options ("File" > "Solution Options"). If there are no fonts already imported, the user can import any available fonts by using the "Import Fonts" button which is placed on the upper-right corner of the Text Style Palette section.

See also:

- Text Style in SceneComposer User Manual

Support for Display and Device Meta-Info

Display is now a dynamic item and has dynamic properties (like widget, render target, effect and layouter properties). Device Package is a new dynamic item which is displayed as the root element in the Render Target Explorer panel.

See also:

- Add a new Display in SceneComposer User Manual

Render Target Preview

Preview functionality for render targets has been extended: when deactivating/activating a render target - by unchecking/checking the checkbox - in order to disable/enable the preview of it, the render target gets also unloaded/uploaded. Therefore, one can make use of several WindowSurfaceRenderTarget objects having the same layer by activating only one of them - the others will not be uploaded (Mantis 3935).

See also:

- Add Render Target in SceneComposer User Manual

Import KTX Image Format

A new image format - "Khronos Texture" (.ktx) - can be imported and used in SceneComposer.

See also:

- Import .KTX Image Format in SceneComposer User Manual

Global Cameras Rendering Order

The global cameras rendering order changed. The rendering sequence for 2D and 3D cameras is:

- 1) 2D cameras
- 2) 3D cameras

See also:

- Rendering Order in SceneComposer User Manual

Minor Improvements

- **Animation Editing Modes.** In Animation Timeline Editor and Animation Curve Editor, the buttons representing the keyframe edit modes were replaced with one ComboBox containing all the options.
- **Comments on Composite Properties.** Now it is possible to set a description for the composite properties. This description will be shown as tooltip in the properties panel for the associated property of the the composite node.

- **Ghosting.** This functionality pertains to the "keyboard support" feature which is still under development. Ghosting was implemented for the panels with a tree structure. A placeholder will indicate the possible position of a node (on the same level as another, as child of another node or as property for a node).

- **Usages Panel.** If a reference has usages in the solution, when trying to remove it, a panel containing these usages will be displayed.

- **Added TextStyle.** New TextStyle support was added in Theme.

- **Default Solution.** Folders for the new item types like ClearMode, TextStyle and BitmapProfile were added. Also some folders were moved or removed.

- **Optional SVNModule.** Due to performance reasons, the SVNModule is now optional in SceneComposer. If the assembly FTC.SourceControl.SVN.dll does not exist in the same folder as SceneComposer executable no SVN functionality will be available. Additionally it is possible to disable the SVNModule via a checkbox in the Preferences->SourceControl panel (Mantis 4362).

- **Texture Generation.** No longer available for vertex buffer editing (Mantis 4204).

SceneComposer Known Issues

- SCL feature is not working properly on WinXp due to the long path of the temporary directory
  `(c:\Documents And Settings\...).`

- There is no automatic conversion for SCLs. Any SCL solution needs to be converted manually and re-exported.
- Uniform auto activation is not working yet for appearances which have no explicit uniform setter but are using one from the solution config.
- Texts configured by using Master Language in SCL solutions, are not loaded in the solution where the SCL is referenced (the Ids are displayed instead of the text).
- Mipmapping might not work correctly in some situations.
- Photoshop import: position and bounding box of text layers are not imported correctly (texts are shifted).
- Photoshop import: in some situations, text layers are imported as bitmaps and may cause crashes.

Import from Photoshop

Photoshop Import/Export

From Candera 2.10 to 3.0, TextBrush was changed in that:

- Alignment property was removed and TextBrush in 3.0 behaves just like Alignment was "ConstantHeightAlignment"
- LayoutingArea property was introduced

SceneComposer accomodates these changing by:

- Post-processing solutions when converting 2.10 solutions or importing Photoshop solutions (.scsxe). Because in 2.10 text effects were imported using Alignment=ConstantHeightAlignment, during the post-processing, render nodes will be repositioned. During conversion, same processing occurs, so positions might be changed in order for the text to appear to the same position in the view.
- Setting LayoutingArea at Photoshop-export time. LayoutingArea value is set to BoundingRectangle value.

Import Names Validation

This change was done due to misalignments between SceneComposer's name validation rules and name validation rules implemented in importers and PhotoshopExporter. The PhotoshopExporter script's name validation rules were very restrictive, replacing commonly used characters (even spaces) with underscore. The attempt was to get PhotoshopExporter's naming validation as close as possible to SceneComposer's ones.

Photoshop Known Issues

- Some texts might appear slightly offseted after converting older solutions or importing from Photoshop due to Candera computations. Especially visible in text areas with center justification (text appears shifted to right).
- Position and bounding box of text layers are not imported correctly (texts are shifted).
- In some situations, text layers are imported as bitmaps and may cause crashes.