CGI Studio 3.2.1

- Candera V3.2.1
  - Candera/FeatStd CMake Build System Changes
  - Candera Engine 3D
  - Candera Engine 2D
  - Candera Scripting
  - Candera System
  - Candera Device
  - Candera Globalization
  - TextEngine
  - Migration Guide
  - 64 Bit Support
- Courier V3.2.1
- FeatStd V3.2.1
  - FeatStd CMake Switches
  - FeatStd Monitor
  - FeatStd Platform
  - FeatStd Util
- Analyzer V3.2.1
  - Scene Analyzer
  - Communication
- SceneComposer V3.2.1
  - Requirements
  - SceneComposer Usability Improvements
  - Photoshop Exporter

Candera V3.2.1

Release Date: June 2016

To keep backwards compatibility for any interface change in the public API the old interface has been kept and marked as deprecated. The deprecated interfaces will be removed in future versions. Compiler warnings will be generated when using a deprecated interface through the usage of Candera Macro CANDERA_DEPRECATED_3_2_1.

Candera/FeatStd CMake Build System Changes

V3.2.0	V3.2.1
CANDERA_MONITOR_CUSTOM_EXPERIMENTS	MONITOR_CANDERA_CUSTOM_EXPERIMENTS
CANDERA_PERFORMANCE_RECORDER_DISABLED logic has been inverted.	MONITOR_CANDERA_PERFORMANCE_RECORDER_ENABLED
MONITOR_OMIT_OPENGL_RECORDING logic has been inverted. Flag were not part of CMake in previous versions.	MONITOR_RECORDING_OPENGL_ENABLED
MONITOR_OMIT_2D_CALLS_RECORDING logic has been inverted. Flag were not part of CMake in previous versions.	MONITOR_RECORDING_2D_CALLS_ENABLED
CANDERA_PERFMON_BUFFERSIZE logic has been inverted. Flag were not part of CMake in previous versions. **See also:** FeatStd CMake Switches	MONITOR_CANDERA_PERF_RECORD_BUFFER_SIZE

Existing monitor settings have been moved from code to CMake settings:

Name	Description	Possible values
MONITOR_RECORDING_OPENGL_ENABLED	Enables or disables OpenGL monitor recordings.	ON/OFF
MONITOR_RECORDING_2D_CALLS_ENABLED	Enables or disables monitoring of 2D driver calls.	ON/OFF

New CMake variable was added for Candera2D layout clipping:

Name	Description	Possible values
CANDERA_LAYOUT_CLIPPING_ENABLED	Defines whether Candera2D layout clipping (setting a clipping rectangle for every render node) is enabled or not.	ON/OFF

Candera Engine 3D

New Functions in Renderer

Renderer has new functions to activate and deactivate a camera (Renderer::ActivateCamera, Renderer::DeactivateCamera), to activate a clear mode (Renderer::ActivateClearMode), and scissor (Renderer::ActivateScissor).

Dirty Area Management

Renderer::RenderCamera now accepts a rectangle as the parameter that defines a dirty area. A dirty area denotes an area on the screen or render target in which the content has changed, thus only this area requires updating by the Renderer. The dirty area rectangle parameter defaults to the full size of the screen or render target. If the user provides his own dirty area rectangle, only this area will be cleared and drawn to. Objects whose bounding rectangle is outside the dirty area will be skipped. Generally speaking, the smaller the dirty area, the smaller the bandwidth required for clearing and drawing, and the more objects can be skipped.

This feature also adds the properties Renderer::Statistics::CulledNodesUsingDirtyAreaScissor and Renderer::Statistics::AverageDirtyAreaFactor. CulledNodesUsingDirtyAreaScissor denotes the number of objects that could be skipped by the Renderer due to their bounding rectangle being outside the dirty area. AverageDirtyAreaFactor denotes the average dirty area divided by the size of the screen or render target. E.g. a factor of 0.8 means that the average dirty area accumulated since the last Renderer::Statistics::Reset is 80% of the size of the screen or render target. The bigger the factor, the less performance improvement can be expected. Under some circumstances, which depends on the actual content being used, performance can even decrease. Therefore always profile this feature to see if your content benefits from it or not.

New Renderer::Statistics properties

Renderer::Statistics now also exposes the number of color buffer, depth buffer, and stencil buffer clears, as well as the number of nodes that were culled due to using a dirty area, and the ratio of the average dirty area to its screen/render target area.

CanvasGroup::TranslatePivot renamed

CanvasGroup::TranslatePivot has been renamed to CanvasGroup::TranslatePivot2D.

PlanarShadow::GetId and SetId renamed

PlanarShadow::GetId has been renamed to PlanarShadow::GetStencilBufferId. In the same manner also PlanarShadow::SetStencilBufferId.

Camera Look-At Node

A new feature allows to set a look-at node for a camera. If such a node is set the camera will always look at that node.

See also:

Camera::SetLookAtNode.

Normalize Rectangle from Render Target Space

A new function to normalize a given rectangle from the render target space has been added.

See also:

Math3D::NormalizeRectangle().

Candera Engine 2D

Dirty Area Management

Renderer2D::RenderCamera now accepts a rectangle as the parameter that defines a dirty area. A dirty area denotes an area on the screen or render target in which the content has changed, thus only this area requires updating by the Renderer. The dirty area rectangle parameter defaults to the full size of the screen or render target. If the user provides his own dirty area rectangle, only this area will be cleared and drawn to. Objects whose bounding rectangle is outside the dirty area will be skipped. Generally speaking, the smaller the dirty area, the smaller the bandwidth required for clearing and drawing, and the more objects can be skipped.

This feature also adds the properties Renderer2D::Statistics::BlitsCulled and Renderer2D::Statistics::AverageDirtyAreaFactor. BlitsCulled denotes the number of objects that could be skipped by the Renderer due to their bounding rectangle being outside the dirty area. AverageDirtyAreaFactor denotes the average dirty area divided by the size of the screen or render target. E.g. a factor of 0.8 means that the average dirty area accumulated since the last Renderer2D::Statistics::Reset is 80% of the size of the screen or render target. The bigger the factor, the less performance improvement can be expected. Under some circumstances, which depends on the actual content being used, performance can even decrease. Therefore always profile this feature to see if your content benefits from it or not.

Renderer2D::Statistics

Renderer2D::Statistics provides information on how many blits and clears were submitted to the RenderDevice2D in the process of executing Renderer2D::RenderCamera or Renderer2D::RenderAllCameras. Furthermore the statistics keep track of how many blits could be culled by the Renderer2D, as well as the ratio of the average dirty area to its screen/render target area.

Candera Scripting

New Lua Functions

Two new Lua functions have been added in this version:

- Candera.SetEnabled
- Candera.IsEnabled

New Script Callbacks.

This version adds 5 new Callbacks:

- Awake
- OnDestroy
- OnEnable
- OnDisable
- LateUpdate

Candera System

New EntityComponentSystem Function

Components from the EntityComponentSystem now have a function to see if the component is attached to an entity (Component::IsAttached). Rectangle now has a set function (Rectangle::Set).

New MetaInfo Macro

A new CdaDeprecated macro has been added to make a MetaInfoBase object deprecated. This was further used to deprecate all TextBrush effects.

Candera Device

Performance Recorder for WaitSync added

Monitor instrumentation has been added for WaitSync methods of Render Targets. Timings of WaitSync will be measured for Analyzer usage.

See also:

Candera::Synchronizable::WaitSync

Extended PlatformDescrition structure

A new field of PlatformDescription defines whether the platform uses a Window Manager.

Candera Globalization

Threadsafe Globalization

The CanderaGlobalization classes have been enhanced for thread-safety. Especially Candera::Globalization::CultureChangeListener now handles Candera::Globalization::CultureChangeListener::OnCultureChanged notification in a thread safe way.

See also:

Candera::Globalization::CultureChangeListener::Obtain and Candera::Globalization::CultureChangeListener::Release.

TextEngine

Extended GlyphCacheAccess

TextRendering::GlyphCacheAccess interface was extended with a function that should check if a glyph is valid in current context, even if cached. If the GlyphCacheAccess implementation returns false on this method, the glyph caching system will recall TextRendering::GlyphCacheAccess::Create and will replace the cached item with the one provided by the new call. Create function is extended with a parameter representing the old cached item for reference.

Migration Guide

Following an overview about interface changes between Candera V3.2.0 and Candera V3.2.1.

Description	Candera V3.2.0	Candera V3.2.1
Activating ClearMode moved to Renderer	ClearMode::Activate()	Renderer::ActivateClearMode(const ClearMode*)
Activating ClearMode via Camera moved to Renderer	ClearMode::Activate(const Camera&)	Renderer::ActivateClearMode(const Camera&)
Setting a shader uniform moved to Renderer	RenderDevice::SetUniform(Handle, const Char*, const void*, Shader::UniformType, UInt)	Renderer::SetUniform(Handle, const Char*, const void*, Shader::UniformType, UInt)
Submitting a drawcall moved to Renderer	RenderDevice::DrawVertexBufffer(const VertexBuffer&)	Renderer::Draw(const VertexBuffer&)
Submitting an instanced drawcall moved to Renderer	RenderDevice::DrawInstancedVertexBuffer(const VertexBuffer&, UInt32)	Renderer::Draw(const VertexBuffer&, UInt)
Setting the transformation matrix moved to Renderer2D	RenderDevice2D::SetTransformationMatrix(ContextHandle2D, SurfaceType, const Matrix3x2&)	Renderer2D::SetTransformationMatrix(ContextHandle2D, SurfaceType, const Matrix3x2&)
Blit moved to Renderer2D	RenderDevice2D::Blit(ContextHandle2D)	Renderer2D::Blit(ContextHandle2D)
Clear moved to Renderer2D	RenderDevice2D::Clear(ContextHandle2D, Float, Float, Float, Float)	Renderer2D::Clear(ContextHandle2D, Float, Float, Float, Float)
Implementation of GlyphCacheAccess::Create	UInt8* CustomGlyphCache::Create(const GlyphBitmap &bitmap) { Create and return new item based on bitmap }	UInt8* CustomGlyphCache::Create(const GlyphBitmap &bitmap, UInt8* cacheItem) { Create or update the cacheItem and return it. }

64 Bit Support

Support for building with **64 bit** has been **added**. Please note that **32 bit** builds are **no longer supported**. In order to obtain a 64 bit build, one has to configure the build to use a 64 bit compiler. This can be done during CMake configuration. This configuration step can be performed during CMake setup. For users with Visual Studio versions earlier than Visual Studio 2019 (version 16), make sure to select 'x64' as the optional platform for the generator during CMake configuration.

Doing this will automatically enable the CMake variable FEATSTD_64BIT_PLATFORM which is further used in Candera.

Two new types have been introduced for unsigned and signed integers which extend to the correct size when used in a 64 bit system. These are **FeatStd::SizeType** and **FeatStd::OffsetType** (equivalent to std::size_t type and std::ptrdiff_t types).

> The API has been adapted at various locations to use FeatStd::SizeType and FeatStd::OffsetType instead of integer types. Application code using a previous version will be compatible due to implicit integer conversion. Please take care on compiler warnings related to type conversions, alignment casts and signed/unsigned comparisons.

Courier V3.2.1

Foundation Changes

Time stamp and Source ID have been added to the Touch Message and the Touch Info.

See also:

Courier::TouchMsg::SetTimeStamp(), Courier::TouchMsg::SetSourceId() and Courier::TouchInfo.

Visualization Changes

- Renderjob Strategies introduced
- Create View by View ID
- Dirty Area Management

Renderjob Strategies introduced

A new class Courier::RenderJobStrategy has been introduced which is used to sort the RenderJobs. The default sort criterias are:

1. offscreen render jobs before Display render jobs.
2. 2d render jobs before 3d render jobs
3. clear render jobs before normal view render jobs
4. lower camera sequence number before higher camera sequence number For custom sort criteria simply derive from RenderJobStrategy and use the Renderer::SetRenderJobStrategy. The Courier::Renderer will use the Courier::RenderJobStrategy and utilize also the Courier::Gdu::RenderState structure.

Create View by View ID

The class Courier::ViewFactory has been enhanced by a method Courier::ViewFactory::Create which creates a View object identified by its View id. Used by Courier::ViewHandler class.

Dirty Area Management

Dirty Area Management in Widgets

The easiest way for a widget developer is to simply call the Invalidate method of the Courier::FrameworkWidget whenever the scene tree has been changed (structure and setting changes). By default this will lead to an invalidation of the associated view with a dirty area that marks the complete area as dirty.

The Courier dirty area management enables widget developers to overwrite that Invalidate method. To provide an out-of-the-box dirty area based implementation a base widget is recommended that takes the bounding rectangle (2D) resp. bounding box (3D) before the change and uses it as source for the dirty area of the current frame.

The bounding rectangle/box has first to be transformed into the view space of the camera.

Scene Dependencies

Invalidation dependencies have been extended by an optional dirty area. This enables the application developer to limit invalidation dependencies to dirty areas. Dependent on the size of the dirty area invalidation of a view in a small area no longer leads to an invalidation of the complete chain of invalidation dependencies.

The only change required on application side to limit invalidation dependencies is to provide a dirty area as small as possible when calling Invalidate. If this is already handled by a common widget base class, see next section, no additional changes are required.

Widget Changes

A common base class for 2D and 3D is already provided which overwrites the following method:

See also:

Courier::FrameworkWidget::Invalidate

If the optional dirtyArea parameter is not set, the bounding rectangle (in 2D) resp. the bounding box (in 3D) of the attached node is taken as dirty area. This base class is called BaseSampleWidget, which can be found in "<cgi_studio_root>/cgi_studio_cit/src/CitApps/CitDemoApp/Widget".

For default dirty area management all widgets need to be derived from this widget class.

In case that the bounding rectangle/box is not sufficient for correct invalidation, the widget itself has to provide a corresponding dirty area via the FrameworkWidget::Invalidate method. In this case it is also up to the widget to take care of correct invalidation of the old dirty area.

Camera View Space Transformation

In case an own dirty area is provided, the widget first has to transform its specific dirty area into the camera view space before calling Invalidate. This is done by multiplying the world transformation matrix of the attached node with the view matrix of the camera, as done in the following code snippet:

```
Candera::Matrix3x2 worldTransformation = node->GetWorldTransform();
cameraDirtyArea.Transform(worldTransformation * camera->GetViewMatrix()); // transform dirty are
```

The camera can either be added to the widget via an own property or be retrieved from the parent view in the following way:

```
Courier::ViewScene2D::CameraPtrVector& cameras = GetParentView()->ToViewScene2D()->GetCameraPtrV
```

In case that a scene contains more than one camera the transformation of the dirty area has to be done for each camera.

See also:

BaseSampleWidget::Invalidate implementation as a reference.

FeatStd V3.2.1

FeatStd CMake Switches

V3.2.0	V3.2.1
FEATSTD_MONITOR_TYPE	MONITOR_COM_TYPE
FEATSTD_MONITOR_BLOCK_UNTIL_CONNECT	MONITOR_BLOCK_UNTIL_CONNECT
FEATSTD_MONITOR_TCPIP_PORT	MONITOR_TCPIP_PORT
FEATSTD_MONITOR_TCPIP_ADDRESS	MONITOR_TCPIP_ADDRESS
FEATSTD_MONITOR_TCPIP_STACK_ENABLED	MONITOR_TCPIP_STACK_ENABLED
FEATSTD_MONITOR_SERIALPORT_BAUDRATE	MONITOR_SERIALPORT_BAUDRATE
FEATSTD_MONITOR_SERIALPORT_ADDRESS	MONITOR_SERIALPORT_ADDRESS
PERFORMANCE_RECORDER_GENERATION_DIR	MONITOR_CANDERA_PERFORMANCE_RECORDER_GENERATION_DIR
PERFORMANCE_RECORDER_TEMPLATE_DIR	MONITOR_CANDERA_PERFORMANCE_RECORDER_TEMPLATE_DIR

Existing monitor settings have been moved from code to CMake settings:

> These flags are marked as advanced and should only be changed if the current settings have side effects.

Name	Description	Possible values
MONITOR_CANDERA_PERF_RECORD_BUFFER_SIZE	Defines the buffer size on target which is used for performance log tracing. When only perf-log streaming is used, it can use a smaller buffer. 4000 as size can be enough then. The effective minimum requirement depends on the amount of recorders.	positive integer
MONITOR_CANDERA_PERF_RECORD_STRID_BUFFER_SIZE	Defines the maximum of stored strings that can be sent (performance logs) with an id to reduce payload.	positive integer
MONITOR_CANDERA_PERF_RECORD_STRING_SIZE	Defines the maximum length of a single string that can be sent (performance logs).	positive integer

FeatStd Monitor

Non-blocking connection setup

The monitor supports non-blocking connection setup now. CGIAnalyzer and target can be connected any time now.

Performance logs: Payload reduction

The payload of performance logs are reduced to approx. 36 percentage of previous version. The buffer for streaming can be smaller now due to restructuring the sent data. Using the recording only functionality can store more recordings within the same buffer. The amount of recordings traced is defined by the size of buffer. Changes to the buffer size can be done by changing the CMake flag MONITOR_CANDERA_PERF_RECORD_BUFFER_SIZE.

See also:

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

FeatStd Platform

64 Bit Support

Support for building with 64 bit has been added. In order to obtain a 64 bit build, one has to configure the build to use a 64 bit compiler. Doing this will automatically enable the CMake variable FEATSTD_64BIT_PLATFORM which is further used in FeatStd. If a 32 bit compiler is used, the CMake variable FEATSTD_32BIT_PLATFORM will be enabled automatically.

Two new types have been introduced for unsigned and signed integers which extend to the correct size when used in a 64 bit system or a 32 bit system. These are **FeatStd::SizeType** and **FeatStd::OffsetType** (equivalent to std::size_t type and std::ptrdiff_t types).

> The API has been adapted at various locations to use FeatStd::SizeType and FeatStd::OffsetType instead of integer types. Application code using a previous version will be compatible due to implicit integer conversion. Please take care on compiler warnings related to type conversions, alignment casts and signed/unsigned comparisons.

FeatStd Util

Thread safe String

The FeatStd::String class has been enhanced by a method String::GetCriticalSection which can be used to protect the internal c-string from modification by other threads.

> The feature is only available when CMake flag FEATSTD_THREADSAFETY_ENABLED is enabled.

Analyzer V3.2.1

Analyzer V3.2.1

Scene Analyzer

Loading Performance Logs

A streamed performance log can be loaded after pressing pause now.
Opening performance logs of older versions is supported now.

Dashboard functionality

The dashboard can load more than the default value recorders (CpuLoad, GpuLoad, MemoryUsage) now. Drag'n'Drop a graph to show multiple value-recorders in one graph is possible now.
Graphs with no content are no longer shown.

Communication

Stabilized the connection and solved crashes (hang up) when pressing stop button.
Requests for communication port uses own scheduler now. This results in better reactivity and a more stabilized connection.
Adaptions made to support non-blocking connection setup.

See also:

- FeatStd Monitor

SceneComposer V3.2.1

Requirements

Requirements

Starting with this version, the .NET 4.5.2 is needed in order to run SceneComposer (Mantis 6824).

SceneComposer Usability Improvements

Solution Conversion from TextEffects to TextNode2D

A new option "Replace Old Text Nodes" was added in the context menu for all scenes, composites and folders containing scenes or composites. This will enable the user to replace the TextNode-TextBrushEffect pairs with the new TextNode2D-BitmapBrushBlendEffect pairs (Mantis 6173 & 6820).

See also:

- Solution Conversion from TextEffects to TextNode2D in SceneComposer User Manual

---

TextNode2D Layout Rework

New properties were added for the TextNode: TextAlignment, LineSpacing and Trimming. A new category called "Text Layout" was added. This includes the properties noted before and also MultiLine and WordWrap. All the properties from this category are visible only if Enable Text Layout is checked. If it is unchecked, the default values of the properties will determine in which way the text is rendered (Mantis 6682 & 6820).

Photoshop exporter now exports the new layout properties. The Y position of the imported TextNode2D is now calculated for each font to achieve <1 px accuracy.

See also:

- 2D Nodes in SceneComposer User Manual

Logging Assebly & Plugin Versions

SceneComposer now writes in the log file the version of .dlls and .exe files located in SceneComposer's directory. For .dlls and .exes that .NET assemblies that can be loaded, the

assembly version will be written. or the other, the file version (Mantis 4761).

See also:

- Automatic Imports in SceneComposer User Manual

Enable Automatic Import

A new option was added in Preferences and in Automatic Imports configuration to enable/disable the feature (Mantis 6443).

See also:

- Automatic Imports in SceneComposer User Manual

Scripts Creation & Execution

When a new script is created, the script is automatically loaded in the "Script Editor" if visible (Mantis 6854).

The execution of any available script is possible by using the control interface (Play, Pause, Resume, Stop) (Mantis 6529).

See also:

- Scripting in SceneComposer User Manual

Sorting "Solution Explorer" Folders & Items

In the Solution Explorer panel all the folders and items will be sorted only for presentation. They will be sorted alphabetically, folders first, items after. New data will always be appended to internal list. Sorting for SCML file will not be made anymore In this way, the modification of the SCML file will be avoided (Mantis 4761).

"About" Dialogue

Many times it is necessary to copy the version and the revision number of SceneComposer. A new feature allows the user to copy this piece of information directly from the "About" dialogue (Mantis 6513).

Other Changes

- Converted solutions to 3.1.1 do not contain the new specialized uniform setters. When converting a solution from an older version, a check will be made whether the Uniform Setter Templates are up to date. If the values for the shader param uniforms were updated in the latest version of the solution template, the uniforms in the converted solution will be updated. If new uniforms were added, they will be added also in the converted solution (Mantis 6131).

- Point Sprite objects from previous versions are not visible after conversion to the 3.2.1 version. The default shader config will be copied from the latest base solution template (Mantis 6936).

Minor Improvements

- **Alignement** SCHost was aligned with candera type changes for 64bit support (Mantis 6822).
- **3D Sources** SCHost solution does no longer include sources for 3D if CANDERA_3D_ENABLED is not activated. The same is available for 2D.
- **Renamed Button** The "Cancel" button from "Rules Editor" dialog was renamed into "Close" (Mantis 6536).
- **Removed Context Menu Option** The "Rename" context menu option of a transition rule is not necessary. The option was removed (Mantis 6537).
- **Shows More Items** SceneComposer shows more than 5 items as "Recent Solutions" (Mantis 6684).
- **Display Window** Double click on a display in the "Render Targets" panel opens the display window (Mantis 6634).
- **Categories Ordered by Name** In ToolBox, all categories are ordered by name and placed before the elements without a category (Mantis 6638).
- **Log File** The application (SceneComposer) logs the version of all dependent assemblies AND plugins in the log file (Mantis 6823).
- **Import Fonts** The possibility to import fonts was added in the "Text Style Palette" panel, in order to improve the usability (Mantis 5974).
- **Fonts Name** Fonts are now imported using the font name instead of the file name (Mantis 5817).
- **Piece of Information** The user is informed that there are elements which are based on templates and the references will be lost after converting (Mantis 6820).

- **Camera** Scene 2D Editor camera is now reset to fit the objects in the scene instead of a preset default position and zoom (Mantis 6825).

Photoshop Exporter

Photoshop Exporter

- Due to the recommendation to create documents with a 72 DPI resolution, the conversion will be done implicitly without any confirmation from the user. In this way, unexpected behavior which may be caused by internally conversion to 72 DPI will be avoided. The exporter scripts requires the resolution of the document to be 72 DPI in order to correctly compute and report the values of some properties of the layers. However, the working document is not affected, because the export (and the conversion) is performed on a copy of it (Mantis 6647 & Mantis 6795).
- Trimming is no longer done on layers with transparent areas. The whole layer bounds are taken from Photoshop.
- The position of the fonts is now calculated on export/import with little differences from the original (Mantis 6802 & 6646).
- Photoshop exporter now exports text with the appropriate effect for Amber platform (Mantis 6821).

Known Issues

- Photoshop character tracking is not supported, warning has been added in the export log.
- The Photoshop Stroke layer style is not supported in SceneComposer.