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.