Candera V3.1.0

Release Date: July 23, 2015

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_1_0.

- Candera Engine 3D
- Candera Engine 2D
- Candera System
- Candera Device
- Candera AssetLoader
- Candera Globalization
- TextEngine
- SCHost.dll
- Migration Guide

Candera Engine 3D

Occlusion Culling

This new feature allows to disable rendering of the objects when they are not currently seen by the camera because they are obscured by other objects. Occlusion culling becomes enabled when the render strategy for camera is set to OcclusionCullingCameraRenderStrategy. During the rendering, all nodes initialized by the strategy will issue occlusion queries. If query results are available from the previous frame, they are evaluated and all nodes deemed occluded will have their bounding box rendered invisibly instead of the node itself. Visible nodes, or nodes not having a QueryProperty attached, will be rendered as usual. A new query to determine visibility for the next frame will be issued regardless the result of the previous query.

Drawcall Batching by implicit Geometry Instancing

On OpenGL ES3.0 platforms, Candera's renderer automatically batches draw calls of nodes where possible, and issues a single instanced draw call per batch (Geometry Instancing) instead of submitting an individual draw call per node.

To trigger draw call batching, the following prerequisites must be met:

- an instance-able shader must be used, and the corresponding Candera::Shader object must have its maximum instance count set accordingly.
- nodes must share the vertex buffer, shader, textures, and render mode
- nodes must not use multi-pass appearances
- nodes must not use object-space lighting

If prerequisites are not met, nodes will be rendered individually.

In order to achieve good batching rates, nodes should be sorted by shader, or grouped together using a separate camera to guarantee sequenced submission to the renderer.

Renderer Statistics

Renderer::GetStatistics() provides information on how many draw calls, instanced draw calls, instances, vertices, indices, uniforms and SetUniform() calls were submitted to the RenderDevice in the process of executing Renderer::RenderCamera() or Renderer::RenderAllCameras().

CameraRenderStrategy enhanced

CameraRenderStrategy has been enhanced with a new choice "SkipNode" for CameraRenderStrategy::RenderPassAction. The interface has been enhanced with CameraRenderStrategy::SetRenderPassCompleted() and CameraRenderStrategy::Render().

Candera Engine 2D

Animation Property Setter for SnapToDevicePixel

A new animation property setter has been added to animate the "SnapToDevicePixel" property.

See also:

Animation::SnapToDevicePixelEnabled2DPropertySetter

Candera System

Localization

Interface for Internal::LocalizableStringData and Internal::ParameterizedLocalizableStringData has been changed.

Overloaded placement operators *new* and *delete* have been replaced by Internal::LocalizableStringData::Create and Internal::ParameterizedLocalizableStringData::Create methods.

These were introduced to better handle the allocation and creation of new instances. As the default placement *new* operator is used within the implementation of Create methods, the matching operator *delete(void* pMem, void*)* had to be added also.

Candera Device

DirectTextureImage External Buffers and New Formats

DirectTextureImage now supports application defined buffers to be provided for the texture data, as well as additionally supporting the 32 bit unpacked formats RGBA and BGRA.

This functionality can be used by setting the logical and/or physical address properties of DirectTextureImage using DirectTextureImage::SetLogicalAddress and DirectTextureImage::SetPhysicalAddress. The default behaviour of DirectTextureImage, where the buffers are allocated by the driver, is used if these properties are set to the default values. The addresses should be provided as a pointer array of four pointers in the case of logical addresses. Depending on the format, one or more pointers to provided buffers should be set and unused pointers in the array set to 0. In the case of physical addresses, these should be provided as an array of unsigned integers, where unused addresses are set to ~0U.

An additional function DirectTextureImage::Invalidate() has been added to the DirectTextureImage which should be called if the data in the provided buffers was modified.

OpenGlEs20 Device Package

Candera OpenGlEs20 Device Package has been added.

This is a generic device which supports 2Dover3D and takes device specific / native configuration from a Device Configuration Factory. Main purpose is quick setup of new OpenGL ES based platforms for evaluation or benchmarking.

Render API String in 3D Capabilities

DevicePackageDescriptor::RenderDevice3DCapabilities has been enhanced by a string indicating the render API, e.g. "OpenGL ES 3.0".

Candera AssetLoader

Maximum Bitmap Glyph Size

AssetDescriptor provides a new interface for obtaining the maximum size of the glyphs of all the fonts in the asset. The maximum size can be used to optimally initialize the decompression buffer of the TextEngine.

Initial Value of BitmapImage2D MemoryPool

AssetLoader automatically assigns the value BitmapImage2D::MemoryPool::FlashMemory during reading BitmapImage2D instances from asset if the asset repository is directly addressable (e.g. in Flash).

See also:

BitmapImage2D::SetMemoryPool()

Candera Globalization

Custom Localizer Support

CMake feature switch CANDERA_CUSTOM_LOCALIZER_ENABLED has been introduced to enable the use of a custom localizer. In case this flag is enabled (along with CANDERA_GLOBALIZATION_ENABLED), the DefaultLocalizer which loads LanguagePacks from assets will be disabled.

Furthermore the interface to set or get a Localizer in GlobalizationCultureManager class has been changed to SharedPointer instead of pointer.

Additional information on how to set up a custom localizer can be found here: Custom Localizer Support

TextEngine

Bitmap Font Sizes

Bitmap fonts have now an interface to acquire information on available sizes.

See also:

TextRendering::BitmapFont::BmFont::GetEffectiveFontSize()
TextRendering::BitmapFont::BmFont::GetFontSizeCount()
TextRendering::BitmapFont::BmFont::GetPossibleFontSize()

SCHost.dll

Source File Relocation

SCHost.dll source files **SCHost.h** and **dllmain.cpp** are no longer provided separately in *cgi_studio_courier* and *cgi_studio_dev*.
Instead, these files are now located in *cgi_studio_schost/src/SCHost*.

Migration Guide

Following an overview about interface changes between Candera V3.0.4 and Candera V3.1.0.

Description	Candera V3.0.4	Candera V3.1.0
Deprecated DynamicPropertyHost::BrowseableForDescendants, replaced with BrowsableForDescendants	`DynamicPropertyHost:`	`DynamicPropertyHost::Browsa`
Getter for Localizer changed to SharedPointer	`Globalization::GlobalizationCult` `().SetLocalizer(&Globalization::`	`Globalization::GlobalizationCultureMana` `().SetLocalizer(Globalization::Localize`

Setter for Localizer changed to SharedPointer	`Localizer* localizer = Globaliza`
	`if (localizer != 0) {`
	`...`
		`Localizer::SharedPointer localizer = Gl`
		`if (!localizer.PointsToNull()) {`
		`...`