CGI Studio 3.1.1

Candera V3.1.1

Release Date: November 3, 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_1.

Candera Engine 3D

Shader Uniform Cache Handles

If a shader uses more than a single uniform, a Shader::UniformCacheHandle is the fastest way to access uniform locations. After a uniform cache accessor was registered in a shader with an array of uniforms to be accessed, the resulting uniform cache handle can be used to retrieve uniform locations in constant time. It does not matter if the uniform is an auto or a custom uniform.

Candera uses this mechanism to get rid of uniform location related (string) compares.

Render Order Criterion

The RenderOrderCriterion class extends the OrderCriterion class to enable the user to control preparation and sorting of render order bins. By default it is now used to skip sorting if the order of the bin has not changed since the last pass.

If an OrderCriterion was assigned to the RenderOrderBin, everything is processed as usual. If a RenderOrderCriterion was assigned to the RenderOrderBin, RenderOrderBin::Sort calls RenderOrderCriterion::PrepareRenderOrderCriterionValues, RenderOrderCriterion::HasOrderChanged, and depending on the result of the previous function, RenderOrderCriterion::Sort.
Using the new interface enables the user to have more control over how sorting is performed, and if sorting needs to be performed at all.

Candera's DistanceToCameraOrderCriterion, ReverseDistanceToCameraOrderCriterion, RankOrderCriterion, as well as RenderStateOrderCriterion have all been changed to use the new RenderOrderCriterion interface.

Batch Order Criterion

The BatchOrderCriterion is a RenderOrderCriterion implementation that sorts bins by either VertexBuffer+Shader+RenderMode+Distance to camera, or by VertexBuffer+Appearance+Distance to camera to increase drawcall batching (i.e. implicit geometry instancing) rates. It also takes advantage of the new

RenderOrderCriterion feature to only sort bins when the order has changed.

Extended Geometry Instancing Support

Classes derived from CameraRenderStrategy can now utilize implicit geometry instancing (i.e. drawcall batching) by passing a flag to its base class. The BreakNodeCameraRenderStrategy has been updated to use it by default.

The PointSprite class is now recognized as being batchable, i.e. point sprites will automatically be picked up by drawcall batching (geometry instancing) if possible.

Render Order Usage Change

The render order was previously associated with the Scene. With this version it has been moved to the Camera. Every camera requires its own render order now. This was done to be able to skip sorting of previously sorted render order bins.

The following functions were declared deprecated:

```
void Scene::SetRenderOrder(AbstractRenderOrder* renderOrder);
AbstractRenderOrder* Scene::GetRenderOrder();
const AbstractRenderOrder* Scene::GetRenderOrder() const;
```

PointSprite Interface and Behavior Changes

Point sprites share a single vertex buffer now, unless a custom vertex buffer is set for an instance of a point sprite by the user. Therefore PointSprite::GetVertexBuffer() returns a const pointer in order to avoid alteration of the shared vertex buffer. The color of a point sprite is now determined by the diffuse color of the material to establish consistent usage of the auto uniform that was used by the point sprite shader. GenericShaderParamSetter::SetPointSpriteActivationEnabled() must be set to true, when using it with a PointSprite.

The following functions were deprecated:

```
void PointSprite::SetColor(const Color& color);
const Color& PointSprite::GetColor() const;
```

MorphingMesh Changes

Just like the PointSprite, the MorphingMesh was changed to establish consistent usage of its auto uniform. Therefore GenericShaderParamSetter::SetMorphingMeshActivationEnabled() must be set to true, when using it with a MorphingMesh.

GenericShaderParamSetter Changes

The GenericShaderParamSetter now handles the auto uniforms of the PointSprite and the MorphingMesh.

The following functions were added, and must be used in conjunction with point sprites and morphing meshes:

```
    bool GenericShaderParamSetter::IsMorphingMeshActivationEnabled() const;
    void GenericShaderParamSetter::SetMorphingMeshActivationEnabled
(bool enableMorphingMeshActivation);
    bool GenericShaderParamSetter::IsPointSpriteActivationEnabled() const;
    void GenericShaderParamSetter::SetPointSpriteActivationEnabled
(bool enablePointSpriteActivation);
```

Renderer Statistics Improvements

Renderer::Statistics now also exposes the total number of queries and occlusion query results for the last render pass.

Generation of index buffers update.

Math3D::CreateVertexBufferWithIndexGenerated is now extended with an argument specifying the minimum size of one index (8, 16 or 32 bit).

Candera Engine 2D

TextNode2D and TextNode2DLayouter

TextNode2D is a RenderNode specialized for text rendering. A TextNode2DLayouter needs to be attached to the node for text layout. The DefaultTextNode2DLayouter implements horizontal or vertical truncation for correct text layout inside given or computed area. TextNode2D requires a BitmapBrush effect for text rendering and one TextNodeRenderer object for generating correct images to be rendered. Four TextNodeRenderer implementations are available in Candera, each corresponding to one CacheType option in the existing TextBrush rendering concept.

SkipNode functionality to Camera2DRenderStrategy

SkipNode action was added to Camera2DRenderStrategy to tell the Renderer2D to proceed with rendering, but skip current node.

Candera Base

TextureImage to Image2D Adaptor

An adaptor class TextureImageToImage2DAdaptor was added to enable using a TextureImage (3D only) as an Image2D. The TextureImageToImage2DAdaptor class implements Image2D interfaces and can therefore be used directly with a corresponding effect.

This can for instance be used to make the DirectTextureImage functionality available in 2D scenes.

Example Usage:

```
TextureImageToImage2DAdaptor::SharedPointer image = TextureImageToImage2DAdaptor::Create();
image->SetTextureImage(texture, pixelFormat);
image->Upload();
```

Default layout alignment change

The default value for horizontal and vertical layout alignment was modified from HCenter and VCenter to HStretch and VStretch.

Candera Device

Shared Sampler Objects

On OpenGL ES 3.0 Platforms texture sampling parameters is decoupled from texture information using sampler objects. The benefit is that multiple textures may share the same sampling parameters and therefore a sampler object. A sampler object can be bound to a texture unit in a single draw call, eliminating the need to set all sampling parameters such as filter mode, wrap mode or anisotropy separately per frame.

In Candera the usage of sampler objects is implicit and hidden from the user. The RenderDevice will determine which textures can share a sampler object and will bind them accordingly. Sampler objects can be shared if the MinLod and MaxLod parameters are left at their default values, to utilize the full potential of sampler objects it is beneficial if the other sampler parameters MaxAnisotropy, WrapMode, MipMapFilter, MagnificationFilter and MinificationFilter are the same for as many textures as possible. The LodBias parameter has no effect on sampler objects as it is a texture parameter only.

A new value recorder id was added for CGI Studio Analyzer to track the number of sampler objects that are created: ValueRecId::SamplerObjectCount. The value represents the number of sampler objects created across all contexts.

DirectTextureImage Alpha Format

The newest iMX6 drivers support setting an 8 bit alpha format on DirectTextureImage. The format FormatAlpha has been added.

Added check for custom Bitmap Format

A check for custom bitmap format has been added to Bitmap class. The new function is used by Candera RenderDevice to check the bitmap format before Upload is attempted.

See also:

- Bitmap::IsCustomPixelFormat

Bitmap size computation

As different platforms support different bitmap formats, the size of the pixel array for a bitmap, in VRAM, is platform dependent. To provide a more accurate value for this size the following modifications were made:

- new RenderDevice::GetSize and RenderDevice2D::GetSize methods were added. This method receives a Bitmap parameter and returns the size of its pixel array in VRAM.
- new BitmapImage2D::GetSize method was added. This method returns the estimated value of the video memory used by the bitmap.
- the possibility to explicitly specify a size for a Bitmap within the constructor was preserved. However, the automatic computation of the size, if default value 0 is used with the constructor, was removed. Now, if default value 0 is used for the *size* parameter in constructor, Bitmap::GetSize method returns 0. If a value is set for *size* parameter, the Bitmap::GetSize method returns this value.

New draw methods for RenderTarget class

New RenderTarget::SafeBeginDraw, RenderTarget::SafeEndDraw and RenderTarget::GetActiveRenderTarget methods have been added.

SafeBeginDraw and SafeEndDraw are used to store the active render target during RenderTarget::BeginDraw / RenderTarget::EndDraw sequence. A pointer to the active RenderTarget is set on a SafeBeginDraw call and reset to 0 on a SafeEndDraw call.

The stored render target can be further retrieve using RenderTarget::GetActiveRenderTarget method.

As the current active render target is further used in the internal implementation of Candera, it is recommended to change calls of BeginDraw and EndDraw to calls of SafeBeginDraw and SafeEndDraw.

Migration Guide

Following an overview about interface changes between Candera V3.1.0 and Candera V3.1.1.

Description	Candera V3.1.0	Candera V3.1.1
Render Device deprecated texture parameter functions are replaced with a single function.	RenderDevice::SetTextureLodMode RenderDevice::SetTextureFilter RenderDevice::SetTextureWrapMode	RenderDevice::ActivateTexture
Scene deprecated funtions are replaced with Camera functions (Note: Every Camera requires its own RenderOrder!)	Scene::SetRenderOrder Scene::GetRenderOrder	Camera::SetRenderOrder Camera::GetRenderOrder
PointSprite deprecated functions are replaced by using the corresponding functions of the PointSprite's Material	PointSprite::SetColor PointSprite::GetColor	Material::SetDiffuse Material::GetDiffuse
PointSprites share a single VertexBuffer by default to facilitate drawcall batching. Custom VertexBuffers can still be set, but getting the VertexBuffer returns a const pointer now.	VertexBuffer* PointSprite::GetVertexBuffer	const VertexBuffer* PointSprite::GetVertexBuffer

Courier V3.1.1

3.1.1-1 New interface for handling the scope mask of Courier::Messages

A fix was made for the scope mask feature of Courier::Messages to distinguish between the original scope mask of the view and the effective mask of the view, which takes into account the scope mask of its children.

With the fix, the interface of Courier::View class has been modified in the following way:

- Courier::View::OnChangeScopeMask was made deprecated;
- New method Courier::View::SetScopeMask was added; this can be used to set the actual scope mask of the view;
- New method Courier::View::GetEffectiveScopeMask was added; this can be used to retrieve the effective scope mask of the view. The effective scope mask of the view is composed internally, from the actual scope mask and the scope mask of the children of the view;
- New method Courier::View::OnScopeMaskChanged was added. This is called every time the effective scope mask of the view or its actual scope mask gets changed.

Interface of Courier::FrameworkWidget has also been changed:

- new method Courier::FrameworkWidget::OnScopeMaskChanged was added also. This must be called every time the scope mask of the widgets changes (i.e. on a CustomWidget::SetScopeMask method).

FeatStd V3.1.1

FeatStd CMake Switches

New CMake switch to enable / disable glGetError and eglGetError calls

A new CMake switch, FEATSTD_GLERRORS_AND_EGLERRORS_ENABLED, has been added. This enables calls to glGetError and eglGetError functions in Candera. By default this is set to OFF for Release builds and to ON for other builds.

SceneComposer V3.1.1

Candera Related Features

Display Renderer Statistics

The Renderer Statistics structure keeps track of per pass data that is relevant to the User. This includes statistics about the metrics of:

- Draw Calls
- Uniform Calls
- Vertices
- Indices
- Batched Draw Calls
- Occlusion Queries (available on OpenGL ES 3.0 API only)
- Visible Draw Calls (available on OpenGL ES 3.0 API only)
- Invisible Draw Calls (available on OpenGL ES 3.0 API only)

This information is presented to the user in a dedicated panel which can be made visible through the "Render Statistic" menu item from the "View" menu. The panel will be displayed by default in the top right region of the SceneComposer user interface. This panel can be moved in any other area, also (Mantis 6037).

See also:

- Display Renderer Statistics in SceneComposer User Manual
- Candera::Renderer

Candera Feature Geometry Instancing

With Geometry Instancing Meshes having the same appearance can be rendered with one draw call.

> This feature has no visual effect, but it can be observed through Render Statistics.

See also:

- Display Renderer Statistics in SceneComposer User Manual
- Drawcall Batching by implicit Geometry Instancing in SceneComposer User Manual

---

Support for BatchOrderCriterion

A new sorting criterion was introduced for render order bins: BatchOrderCriterion (Mantis 6030). BatchOrderCriterion produces a render order containing series of nodes that can then be batched by the Renderer to maximize usage of Geometry Instancing. BatchOrderCriterion has 2 modes of operation:

- SortbyShaderandRenderMode
- SortbyAppearance

See also:

- Configure a New Render Order Bin in SceneComposer User Manual

---

Occlusion Culling

A new camera render strategy was introduced: OcclusionCulling. When OcclusionCulling render strategy is selected, a new property becomes available in property grid: QueryType.

Default value of QueryType (in case of occlusion culling render strategy) was changed to AnySamplePassedConservative. "Occlusion Culling" Render Strategy option is no longer available on platforms with OpenGL ES API different than 3.0 (Mantis 6059).

See also:

- Occlusion Culling in SceneComposer User Manual
- Candera Engine 3D in SceneComposer User Manual
- Candera::OcclusionCullingCameraRenderStrategy

Integrate Candera TextNode

For this version we provide a basic implementation: TextNode will be integrated as a new node type, and not as a replacement (full migrations - solution conversions - will not be possible).

Manipulation & rendering

SceneComposer will provide a new 2D node type: TextNode2D. Properties of the new objects:

- "Text" - specifies the text to be rendered
- "Style" - specifies the style to be used for rendering
- "Renderer" - specifies the renderer used by the text node
- "Text Laying Out Enabled" - enable additional laying out properties

The TextNode2D will be associated with a BitmapBrushColorBlend effect. The TextNode element in the "Toolbox" panel will create a TextNode2D object having a BitmapBrushColorBlend effect when it will be dragged into the "Scene Editor". Relationship with Translator plugin: the Text property can be edited by Translator plugin.

Photoshop export

Photoshop exporter will export the text layers as TextNode2D objects. TextNode2D objects having BitmapBrushColorBlend effects will be exported instead of RenderNode2D having TextBrushBlend effects. Instead of setting BoundingRectangle property, Size (laying out property) shall be set on TextNode.

Point text is converted to paragraph text upon export. This ensures the bounding rectangle is exported properly and the text has the same position as in Photoshop with pixel accuracy.

Alignment support has also been discussed with Candera team, the text will be aligned properly regardless of "Text Laying Out" being enabled.

Solution migration (conversion)

SceneComposer does not offer a conversion from RenderNode2D to TextNode2D.

SceneComposer Usability Improvements

Adapt SceneComposer to Extended GenericShaderParamSetter

IsPointSpriteActivationEnabled and IsMorphingMeshActivationEnabled flags were added in UniformSetter. When a uniform setter is dragged over a pointsprite or morphing mesh, the corresponding flag will be activated accordingly. Also, the new default solutions contain specialized uniform setters for morphing meshes and pointsprites which are assigned by default.

The uniforms setters on which auto activation is not enabled, from older solutions, must be updated by the user.

Color property of PointSprite object was removed. Instead, a material must be assigned to the PointSprite and the diffuse component will specify the desired color (Mantis 5896).

See also:

- Shader Parameters in SceneComposer User Manual
- Generic ShaderParamSetter in SceneComposer User Manual
- Default Shader Configuration in SceneComposer User Manual

Vertex Buffer 24 bit Index Export

VertexBuffers can now be converted to indexed buffers of various index sizes (8bit, 16bit or 32 bit index). BufferType configuration property (which had 2 options: ArrayBuffer and IndexBuffer) was removed. "Is Indexed" property was added. If enabled, the vertex buffer will be converted to an index array buffer, by selecting the optimal index size. Therefore, depending on vertex count, an 8bit, 16bit or 32bit index buffer will be created. BufferType property in AssetData section will specify the effective buffer type (Mantis 5408).

See also:

- Vertex Buffer Editor in SceneComposer User Manual

Context Menu inside "Generate Asset" View

Inside "Generate Asset Library" the options "Select, Expand All and Collapse All" are available for every item right clicked inside "Scene Tree" side of the view (Mantis 5958).

See also:

- Generate Asset Library in SceneComposer User Manual

Browsing through Tabs

A button was added at the end of the tab row. By using it, the user has the possibility to browse through all the tabs from this panel, even if they are not visible (Mantis 5320).

See also:

- SceneComposer GUI in SceneComposer User Manual

Device Capability

"3D Render API" device capability info was added in "SCHost Information" dialog (Mantis 6215).

See also:

- SCHost Information in SceneComposer User Manual

Minor Improvements

- **Adapt Bitmap Profile** The name of RgbFloatingPoint9E5FPixelFormat was changed to RgbFloatingPoint9E5PixelFormat (Mantis 6057).
- **Default Size.** The deafult size of the text in TextNode is 14px instead of 0px as it was until now (Mantis 5961).
- **Drag-drop Multiselected Items.** The user is able to drag-drop multiselected elements from different folders into another one (Mantis 5611).
- **Switched Properties.** From now on the "Style" property is displayed under the "Text" property, in the Properties panel of a "TextNode" (Mantis 6144).
- **Default New Solution.** When a new solution is created on iMX6 platform, the default value for "Solution Template" should be "Base Solution OpenGLES 3.0" (Mantis 6106).

- **Deactivated Button.** The "Generate" button, from "Generate Asset Library" view, will be disabled if the path for the "Asset File" is invalid (Mantis 5904).

SceneComposer Known Issues

- Render Statistics are not reset when opening a new solution if there is no display or if nothing is rendered on it.