Candera V3.3.0

Release Date: November 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_3_0.

- Candera/FeatStd CMake Build System Changes
- Candera Engine 3D
- Candera Engine 2D
- Candera Base
- Candera Scripting
- Candera System
- Candera Device
- Candera Transitions
- TextEngine
- Candera Behavior
- Migration Guide

Candera/FeatStd CMake Build System Changes

Candera Engine 3D

Changed Light Specular Color to White

Changed default specular light color to White due to feedback from designers. Material default specular color is black anyway, so no specular highlight will occur by default. But if user (designer, technical artist) changes material specular color, an immediate change of appearance will be visible.

Reduction of glGetUniform calls on Shader Upload

By improving handling of Shader Uniforms, the number of calls to glGetUniform during shader uploading was significantly reduced.

Introduction of LoadingHint for DeviceObject.

Version 3.3.0 introduces a LoadingHint for the DeviceObject. Loading hints can e.g. trigger warnings when uploading while rendering, or force unloading from all contexts.

New methods for NodeListener

New methods have been added to NodeListener class. These are:

- Candera::NodeListener::OnNodeAdded: this is invoked whenever a parent is set for the node;
- Candera::NodeListener::OnChildAdded: this is invoked whenever a child node is added to the node;
- Candera::NodeListener::OnChildRemoved: this is invoked whenever a child of the node is removed.

Candera Engine 2D

Introduction of LoadingHint for DeviceObject2D.

Version 3.3.0 introduces a LoadingHint for the DeviceObject2D. Loading hints can e.g. trigger warnings when uploading while rendering, or force unloading from a context.

New methods for Node2DListener

New methods have been added to Node2DListener class. These are:

- Candera::Node2DListener::OnNodeAdded: this is invoked whenever a parent is set for the node;
- Candera::Node2DListener::OnChildAdded: this is invoked whenever a child node is added to the node;
- Candera::Node2DListener::OnChildRemoved: this is invoked whenever a child of the node is removed.

TextNode2D cloning strategy

TextNode2D adopts the unified cloning mechanism, with the introduction of DeepTextNode2DCloneStrategy.

For more information see Unified Cloning Interface, Tree Cloning Strategies and Tree Cloner

---

Improved Pixel Accuracy

Scissoring is now utilized on platforms that support it. This improves pixel accuracy in 2D rendering when a dirty area or a camera scissor is used.

Candera Base

Renderer2D3D

A new renderer has been added. It renders the cameras interleaved in order of their sequence-number. This means that 3D-cameras can be rendered between two 2D-cameras and vice versa. The interleaved renderer however, has the requirement that buffer swapping and clearing needs to be done by the according Cameras directly. Therefore auto clear and auto swapping becomes deprecated. Please see the Migration Guide.

DXT image formats

New compressed formats support in Bitmap is avaiable on selected platforms: Dxt1CompressedRgbFormat, Dxt1CompressedRgbaFormat, Dxt3CompressedRgbaFormat, Dxt5CompressedRgbaFormat.

Candera Scripting

Lua Update.

Lua has been updated to version 5.3.3.

New Script Callbacks.

This version adds a new callback:

- FixedUpdate

Time interface.

This version adds a new interface to get time information in Lua.

Added uniform functions.

This version adds new functions to Lua to access uniforms through Candera's shader parameter setter.

Candera System

Update System

This release adds an update system where entities can be registered to receive any of a series of update calls.

Controller System

This release adds a controller system which offers an abstract interface that the user can implement and attach to entities to receive update calls.

Candera Device

Candera Transitions

New Default Transition Types

The transition framework now supports Cut, Fade, Slide and Scale transitions.

Transition Request Handles

Transition requests can now be assigned a handle allowing creation of multiple requests at once. This is a requirement for courier

---

TextEngine

TTC font iterator.

A function to iterate over the style names of the font faces contained by a TTC font was introduced in the Font.h interface. Additional functions to retrieve the font style name and the font family name of the current font set up are available.

Candera Behavior

Added support for Behaviors.

Behaviors have been added to support application development with considering the hierarchy of the scene graphs. Behaviors are small application building blocks that allow the user to create their own application handling. Being similar to Widgets Behaviors have the advantage that they can rely on hierarchical information, as every Behavior has to be attached to a node in the scene graph. By design they can be implemented in such a way that one behavior can even be reused for both 2D and 3D, if no special functionality of the Node below has to be used. Behaviors also support event handling through mechanisms for event routing (to other behaviors) and having a base interface (Behavior::OnEvent) to handle events.

Migration Guide

Following an overview about interface changes between Candera V3.2.2 and Candera V3.3.0.

Description	Candera V3.2.2	Candera V3.3.0

RenderTarget Changes due to Renderer2D3D.	void RenderTarget::SetAutoSwapEnabled(bool enabled);
	bool RenderTarget::IsAutoSwapEnabled() const;
	void RenderTarget::SetAutoClearEnabled(bool enabled);
	bool RenderTarget::IsAutoClearEnabled() const;
	void RenderTarget2D::SetClearMode(const SharedClearMode2D::SharedPointer& clearMode);
	SharedClearMode2D::SharedPointer RenderTarget2D::GetSharedClearMode() const;
	void SetClearMode(const SharedClearMode::SharedPointer& clearMode);
	SharedClearMode::SharedPointer GetSharedClearMode() const;
		Use corresponding Camera and Camera2D settings.
		
		void Camera2D::SetClearColorEnabled(bool isEnabled);
		bool Camera2D::IsClearColorEnabled();
		void Camera2D::SetSwapEnabled(bool isEnabled);
		bool Camera2D::IsSwapEnabled() const;
		void Camera2D::SetClearColor(const Color& clearColor);
		const Color& Camera2D::GetClearColor();
		void Camera::SetClearMode(const ClearMode& clearMode);
		const ClearMode& Camera::GetClearMode();
		void Camera::SetSwapEnabled(bool enable);
		bool Camera::IsSwapEnabled() const;
		
		\ Use dedicated clear and swap cameras if you don't want to manage clearing and swapping on dynamic camera configurations.

RenderDevice Changes due to LoadingHint	static bool RenderDevice::UploadBitmapTextureImage(BitmapTextureImage& textureImage, UInt unit);
	static bool RenderDevice::UploadCubeMapTextureImage(CubeMapTextureImage& textureImage, UInt unit);
	static bool RenderDevice::UnloadBitmapTextureImage(BitmapTextureImage& textureImage);
	static bool RenderDevice::UnloadCubeMapTextureImage(CubeMapTextureImage& textureImage);
	static bool RenderDevice::UploadVertexBuffer(VertexBuffer& vertexBuffer);
	static bool RenderDevice::UnloadVertexBuffer(VertexBuffer& vertexBuffer);
	static bool RenderDevice::UploadShader(Shader& shader);
	static bool RenderDevice::DeleteShader(Handle vertexShaderHandle, Handle fragmentShaderHandle, Handle programHandle);
	static bool RenderDevice::UploadRenderBuffer(RenderBuffer& renderBuffer);
	static bool RenderDevice::UnloadRenderBuffer(RenderBuffer& renderBuffer);
	static bool RenderDevice::UploadDirectTextureImage(DirectTextureImage& textureImage, UInt unit);
	static bool RenderDevice::UnloadDirectTextureImage(DirectTextureImage& textureImage);
		static bool Renderer::UploadBitmapTextureImage(BitmapTextureImage& textureImage, UInt unit, DeviceObject::LoadingHint loadingHint);
		static bool Renderer::UploadCubeMapTextureImage(CubeMapTextureImage& textureImage, UInt unit, DeviceObject::LoadingHint loadingHint);
		static bool Renderer::UnloadBitmapTextureImage(BitmapTextureImage& textureImage, DeviceObject::LoadingHint loadingHint);
		static bool Renderer::UnloadCubeMapTextureImage(CubeMapTextureImage& textureImage, DeviceObject::LoadingHint loadingHint);
		static bool Renderer::UploadVertexBuffer(VertexBuffer& vertexBuffer, DeviceObject::LoadingHint loadingHint);
		static bool Renderer::UnloadVertexBuffer(VertexBuffer& vertexBuffer, DeviceObject::LoadingHint loadingHint);
		static bool Renderer::UploadShader(Shader& shader, DeviceObject::LoadingHint loadingHint);
		static bool Renderer::DeleteShader(Handle vertexShaderHandle, Handle fragmentShaderHandle, Handle programHandle, const Char* shaderName, DeviceObject::LoadingHint loadingHint);
		static bool Renderer::UploadRenderBuffer(RenderBuffer& renderBuffer,