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,

Revision #2
Created 2023-03-02 02:58:42 UTC by Kanai Tomoaki
Updated 2023-06-13 13:41:48 UTC by Kanai Tomoaki