CGI Studio 3.0.2

- Candera V3.0.2
  - Candera Engine Base
  - Candera Engine 3D
  - Candera Engine 2D
  - Candera System
  - Candera Device
  - Candera AssetLoader
  - TextEngine
  - Migration Guide
  - CGI-Studio CMake Configuration
- Courier V3.0.2
- FeatStd V3.0.2
  - FeatStd CMake
  - FeatStd Diagnostics
  - FeatStd Util
- SceneComposer V3.0.2
  - Restructuring Sources
  - SceneComposer Usability Improvements

Candera V3.0.2

Release Date: February 19, 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_2_0.

Candera Engine Base

Resource Objects.

Large data buffers that need to be uploaded or that are rarely used are not longer directly stored in DeviceObject objects. Instead, a ResourceDataHandle structure is stored, which can consist of one of :

- pointer to existing data, pointer to disposer function and size (backward compatible)
- information on where data can be found in asset and size
- custom information for custom ResourceProviders.

ResourceObjects are objects that guarantee that during their lifetime, data referenced by a ResourceDataHandle is available.

Bitmap Resource Objects.

The pixel buffer is stored in the new ResourceDataHandle structure. See migration guide for an example on how it can be used.

Introduce C++ keyword override in Candera

In a method declaration, override specifies that the function must be overriding a base class method.

The keyword override is part of C++ 11 standard and supported by MSVC 11 and gcc 4.7. It helps to prevent unintentional newly created functions, e.g. due to typos, that are intended to override virtual base function.

Candera Engine 3D

Shader: Support for precompiled shaders on all enabled platforms.

Uploading shader programs using Shader::Upload can now be done in a more flexible way. Until now it was only possible to upload either shader source or pre-compiled binaries, through a target-specific implementation of RenderDevice::CompileAndLoadShaderPair.

Now it is possible to upload (and compile if necessary) either shader source or shader binary objects. It is even possible to upload complete shader programs in order to avoid shader linking. How shader upload is done depends on the kind of data that is available in the asset and the capabilities of the target platform, however at runtime this is done completely transparent to the user.

Advantage is that the more flexible way of runtime created shader source code (as used in Candera2D implementations over OpenGL ES) is supported as well as using pre-compiled shaders to save execution time for compiling them. While using pre-compiled shaders on enabled platform is recommended due to performance reasons, still backward compatibility to existing solutions using uncompiled shaders is given.

OpenGL ES 2 Reference Shaders relocated

The Candera OpenGL ES 2.0 Shader language reference Shaders have been moved from `<cgi_studio_candera>/src/Candera/Engine3D/ReferenceShaders` to `<cgi_studio_candera>/src/Candera/Engine3D/RefShaders/ESSL1`

OpenGL ES 3 Reference Shaders

Candera now contains a set of reference Shaders for the OpenGL ES 3.0 Shader language. `<cgi_studio_candera>/src/Candera/Engine3D/RefShaders/ESSL3`

Shader Resource Objects.

Fragment and vertex shader data or access information is stored in the new ResourceDataHandle structure. See migration guide for an example on how it can be used.

VertexGeometry Resource Objects.

The vertex, index and element format arrays are stored in the new ResourceDataHandle structure. See migration guide for an example on how it can be used.

VertexBuffer primitive iteration.

Iteration over triangles is replaced by primitive iteration using a forward iterator, which is more performant, allows iteration over point and line primitives and supports multiple parallel iterations. See migration guide for an example on how it can be used.

VertexBuffer vertex access.

Vertex access interface is moved from the VertexBuffer to a separate class for better performance in case of vertex data accessed from assets. See migration guide for an example on how it can be used.

8/16/32 bit indexed array buffers.

VertexGeometry accepts index buffers that consist of 8, 16 or 32 bit indexes. The configuration of index size can be set through the new values of BufferType enumeration: UInt8IndexedArrayBuffer, UInt16IndexedArrayBuffer and UInt32IndexedArrayBuffer.

VertexElementFormat access.

VertexElementFormat access interface is moved from the VertexGeometry to a separate class for better performance in case of vertex data accessed from assets. See migration guide for an

example on how it can be used.

VertexGeometryBuilder support for 8/16/32 bit index buffers.

VertexGeometryBuilder can create index buffers with values on 8, 16 or 32 bit, depending on the number of vetices, or on the configured format if explicitly set.

BlendEquation Min/Max support.

Min/Max options were added to the RenderMode BlendEquation, which specifies how a new pixel is combined with previously rendered pixels in the frame buffer.

Candera Engine 2D

Node2D Listener support.

Node2DListener defines hooks that are called when certain Node functions are triggered, e.g. when a Node's transformation changes.

Auto Arrangement for GridLayouter

Two new methods available for GridLayouter:

- GridLayouter::SetGridAutoArrangement : sets the grid autoArrangement. See method description for possible layouter arrangement settings.
- GridLayouter::GetGridAutoArrangement: retrieves the grid autoArrangement settings of a given node, set by GridLayouter::SetGridAutoArrangement method.

Candera System

Different namespace than 'Candera' for EnumDataTypes

```
#define ENUM_DATA_TYPE_NAMESPACE MyNamespace        // <-- new!

#define ENUM_DATA_TYPE \
    ENUM_DATA_TYPE_BEGIN(SomeEnum) \
        ENUM_DATA_TYPE_ITEM(SomeItem0) \
        ENUM_DATA_TYPE_ITEM_VALUE(SomeItem1, 4) \
        ENUM_DATA_TYPE_ITEM(SomeItem2) \
        ENUM_DATA_TYPE_ITEM_INVISIBLE(InvisibleItem) \
        ENUM_DATA_TYPE_ITEM_VALUE(SomeItem3, 7) \
    ENUM_DATA_TYPE_END(SomeEnum)

#include <Candera/System/MetaInfo/EnumDataType.h>
```

If ENUM_DATA_TYPE_NAMESPACE is defined before including EnumDataType.h, the enum 'SomeEnum' will be placed into MyNamespace. Without that (ENUM_DATA_TYPE_NAMESPACE is not defined), SomeEnum is placed automatically into namespace Candera.

Candera Device

Uniform Location Caching

Introduced Uniform Caching for Auto uniforms to increase performance. Calls to GetUniformLocation now pass semantic instead of names as parameter, which speeds up comparison within cache.

Maximum Element Index capability

RenderDeviceCapabilities structure contains a new field for the maximum value of vertex index that is supported by the device.

RenderDevice::DrawVertexBuffer

RenderDevice::DrawVertexBuffer handles drawing of a VertexBuffer, regadless of its configuration (primitive types, index formats, upload location).

Added formats on DirectTextureImage

Additional formats have been added to DirectTextureImage::Format, unpacked RGBA and BGRA texture format.

Shader Upload functions now support Upload of precompiled binary shaders.

The function RenderDevice::CompileAndLoadShaderPair only supports the fixed Upload of text or binary shader objects. In order to provide a more flexible handling of shader Upload, this function has been split up into RenderDevice::UploadShaderSource, RenderDevice::UploadShaderBinary and RenderDevice::UploadShaderBinaryProgram. Additionally the function RenderDevice::UploadShader

has been added which calls the appropriate shader upload function transparently to the user. This interface enables the user to choose whether shader shall be available as binary or source shader objects.

Candera AssetLoader

Resource Objects.

The font and raw resources are stored in the new ResourceDataHandle structure. See migration guide for an example on how it can be used.

Dispose after upload.

The dispose after upload feature of DefaultAssetProvider has become deprecated with the introduction of resource objects. The data is no longer attached to the device objects, but rather it is loaded from the asset upon necessity.

TextEngine

Glyph Spacing

Glyph Spacing is available on TextBrush, and LayoutingOptions. This allows a user to correct the spacing between glyphs, by adding a specified number of pixels to the spaces between glyphs.

BitmapFont Engine

BitmapFont Engine is available for usage inside Candera-based applications. A bitmap font is essentially an image file which consists of several characters images and a header that depicts the size and location of each character inside the image. Each bitmap font might contain several fonts - mainly one font face with multiple sizes. One advantage of utilizing bitmap fonts is that the rendering to the screen requires very little resources and since each character is represented as a sub-texture, they can be reused without requiring additional memory on the GPU.

In order to use BitmapFonts, the CMake variable **CANDERA_FONTENGINE** should be changed to *BitmapFont*.

> Harbuzz Text Shaper cannot be used with BitmapFont, either ComplexScriptLib or NoShaping should be chosen.

This font engine uses a proprietary format to describe glyph bitmaps and glyph meta info. In order to use a bitmap font, provide an implementation for Candera::TextRendering::FontStore.

Bitmap fonts require memory mapped assets. So, when the faces are loaded in be sure the storageType is set correctly:

```
TextRendering::FontResourceDescriptor
descriptor.storageType = TextRendering::FontResourceDescriptor::MemoryResource;
```

Generating Bitmap Fonts

The following steps need to be considered:

- Install Bitmap Font Generator from http://www.angelcode.com/products/bmfont/

Open the application. From the Options menu:

- Choose Font Settings. Set as needed the settings for "Font", "Add font file", "Size" etc.
- Configure the Export Options. Set "Font descriptor" to "XML" and Textures to "png - Portable Network Graphics"
- "Save bitmap font as ..."

Convert the resulted XML and PNG files to binary format using the internal tool BmFontCreator using the following command:

- BmFontCreator. exe -o MyFont32.fbm *.fnt

> A bitmap font should be used at its native pixel size. However, they are scalable because of the use of bitmaps, but only scaling down is recommended. In case of up scaling, the visual quality will be reduced. Including separate font sizes for bitmap fonts is possible in order to achieve the best looking results.

Glyph Substitution

A substitution character may be set within the style with the interface Style::SetDefaultCodepoint. This character is used by the text engine in any of the following situations:

- when a glyph is missing from the style. i.e. no matching font can provide that glyph. Exception make control character, like newline or LRE.
- when UTF8 buffers contain malformed characters.

Character Glyph Mapping

GlyphBitmap now contains information about the position within the text from where the glyphs originated, and the location of the glyph within the font.

Glyph Order

The order in which glyphs are generated may be controlled by setting a property within the TextRenderContext. See TextRenderContext::GlyphOrder for details. This is most relevant when doing preprocessing. OriginalGlyphOrder is intended for use when intermediate measurements of the text are required, for instance when truncating texts. OrigitnalChunkOrder is more performant then RenderOrder, and is intende for measurements, where the order of the chunk is not important, for instance instance when computing layout rectangles. RenderOrder shall be used when rendering.

Text Preprocessing

The text engine allows text to be preprocessed by providing utility the class PreprocessingContext. This interface is satisfied by two concrete instances.

- MinimalPreprocessingContext - stores all the information need to be able to render the glyphs.
- MaximalPreprocessingContext - stores all the information available within the text renderer for each glyph. TextBrush allows setting PreprocessedText instead of Text, to speed up text processing within the brush.

Migration Guide

Following an overview about interface changes between Candera V3.0.1 and Candera V3.0.2.

Description	Candera V3.0.1	Candera V3.0.2
Shader Resource Object	const void* fragmentShader = shader->GetFragmentShader(); const void* vertexShader = shader->GetVertexShader();	Shader::ShaderResource fragmentShaderResource(shader->GetFragmentShaderResourceHandle()); const void* fragmentShader = fragmentShaderResource.GetData(); Shader::ShaderResource vertexShaderResource(shader->GetVertexShaderResourceHandle()); const void* vertexShader = vertexShaderResource.GetData(); NOTE: data is guaranteed only until ShaderResource objects are destroyed by going out of scope.
Vertex Geometry Resource Object	bool isGeometryMutable = vg->IsMutable(); const void* vertexArray = vg->GetVertexArray(); void* vertexArray = vg->GetVertexArray(); const UINT16* indexArray = vg->GetIndexBuffer(); UINT16* indexArray = vg->GetIndexBuffer(); const VertexGeometry::VertexElementFormat* formatArray = vg->GetVertexElementFormat(); VertexGeometry::VertexElementFormat* formatArray = vg->GetVertexElementFormat();	bool isVertexArrayMutable = vg->GetVertexArrayResourceHandle().m_isMutable; VertexGeometry::VertexArrayResource vertexArrayResource(vg->GetVertexArrayResourceHandle()); const void* vertexArray = vertexArrayResource.GetData(); void* vertexArray = vertexArrayResource.GetMutableData(); bool isIndexArrayMutable = vg->GetIndexArrayResourceHandle().m_isMutable; VertexGeometry::IndexArrayResource indexArrayResource(vg->GetIndexArrayResourceHandle()); const void* indexArray = indexArrayResource.GetData(); void* indexArray = indexArrayResource.GetMutableData(); bool isFormatArrayMutable = vg->GetFormatArrayResourceHandle().m_isMutable; VertexGeometry::FormatArrayResource formatArrayResource(vg->GetFormatArrayResourceHandle()); const VertexGeometry::VertexElementFormat* formatArray = formatArrayResource.GetData(); VertexGeometry::VertexElementFormat* formatArray = formatArrayResource.GetMutableData(); NOTE: data is guaranteed only until resource objects are destroyed by going out of scope.
Bitmap Resource Object	bool isMutable = bitmap->IsMutable(); const void* pixels = bitmap->GetPixels(); void* pixels = bitmap->GetPixels();	bool isMutable = bitmap->GetPixelsResourceHandle().m_isMutable; Bitmap::PixelsResource pixlesResource(bitmap->GetPixelsResourceHandle()); const void* pixels = pixlesResource.GetData(); void* pixels = pixlesResource.GetMutableData(); NOTE: data is guaranteed only until resource objects are destroyed by going out of scope.

Font/Raw Resource Object	ResourceHandle fontHandle = provider->OpenFontResourceById(fontId); if (fontHandle != 0) { UInt32 fontSize = provider->GetResourceSize(fontHandle); const void* fontAdress = provider->GetResourceAddress(fontHandle); provider->ReadResource(fontHandle, fontData, 0, fontSize); provider->CloseResource(fontHandle); }	ResourceDataHandle fontHandle = provider->GetFontResourceById(fontId); UInt32 fontSize = fontHandle.m_size; const void* fontAddress = ResourceData::GetAddress(fontHandle); ResourceData::CopyData(fontData, fontHandle, 0, fontSize); NOTE: data is guaranteed only until resource objects are destroyed by going out of scope.
VertexBuffer Primitive Iteration	VertexBuffer* vb; const VertexBuffer::Triangle* triangle = vb->GetFirstTriangle(); while (triangle != 0) { UInt16 x = triangle->index[0]; UInt16 y = triangle->index[1]; UInt16 z = triangle->index[2]; triangle = vb->GetNextTriangle(); }	VertexBuffer* vb; for (VertexBuffer::PrimitiveIterator it = vb->GetPrimitiveIterator(); it.IsValid(); ++it) { const VertexBuffer::Primitive& primitive = *it; UInt16 x = primitive.m_index[0]; UInt16 y = primitive.m_index[1]; UInt16 z = primitive.m_index[2]; }
VertexBuffer Vertex access	VertexBuffer* vb; const void* vertex = vb->GetVertex(0); void* mutableVertex = vb->GetMutableVertex(0);	VertexAccessor accessor(VertexAccessor(*vb->GetVertexGeometry())); const void* vertex = accessor.GetVertex(0); void* mutableVertex = accessor.GetMutableVertex(0);
VertexElementFormat access	VertexGeometry* vg; bool isPosition = vg->DoesVertexUsageExistAtUsageIndex(VertexGeometry::Position, 0); const VertexGeometry::VertexElementFormat* positionElementFormat = vg->GetVertexElementFormatAtUsageAndUsageIndex(VertexGeometry::Position, 0);	VertexGeometry* vg; ElementFormatAccessor accessor = ElementFormatAccessor(*vg); bool isPosition = (accessor.GetElementFormatByUsageAndIndex(VertexGeometry::Position, 0) != 0); const VertexGeometry::VertexElementFormat* positionElementFormat = accessor.GetElementFormatByUsageAndIndex(VertexGeometry::Position, 0);

Shader Upload	const void* fragmentShader = shader->GetFragmentShader(); const void* vertexShader = shader->GetVertexShader(); Handle fragmentShaderHandle; Handle vertexShaderHandle; CompileAndLoadShaderPair(vertexShader, vertexShaderHandle, fragmentShader, fragmentShaderHandle);	const void* fragmentShader = shader->GetFragmentShader(); const void* vertexShader = shader->GetVertexShader(); Handle fragmentShaderHandle; Handle vertexShaderHandle; Shader::BuildType buildType; GetShaderBuildType(vertexShader, buildType); if (buildType == Shader::ShaderSource) { UploadShaderSource(Shader::VertexShader, vertexShader, vertexShaderHandle); } else { ... } GetShaderBuildType(fragmentShader, buildType); if (buildType == Shader::ShaderSource) { UploadShaderSource(Shader::FragmentShader, fragmentShader, fragmentShaderHandle); } else { ... } Note: This is normally done transparently by Candera.

CGI-Studio CMake Configuration

- The CMake macro `CgiCreateWidgetSet` now creates an empty WidgetSet if no additional widgets have been added.
- `-m32` has been added as default compiler flag for gcc builds to ensure compilation even on x64 gcc.
- With update of Attribute Caching the user variable CANDERA_MAX_SHADER_ATTRIBUTE_COUNT became obsolete and has been deleted.

Courier V3.0.2

3.0.0-1 Message Processing

1. New Courier::Component::PostProcess method has been added. This method is called after a component has been processed. A custom component is able to react on each process of a component.

2. New Courier::MessageReceiver::Deactivate method has been added. This method deactivates the message receiver in the system. After calling this method the message router stops serving this receiver with appropriate messages (means the method Receive() is called). If a custom message receiver is used this method should be called before destroying the custom message receiver instance.

3.0.0-2 Input Handling

New methods where added for setting custom names for GenericConsoleListener and PosixTerminalListener. These are:

- Courier::Platform::ExtComm::Generic::GenericConsoleInputHandler::SetConsoleListenerName
- Courier::Platform::ExtComm::Posix::SetTerminalListenerName

Corresponding getter methods were also provided.

3.0.0-3 Camera Groups

Activation and enable /disable rendering for views have been reworked to fix different aspects related to handling views together with camera groups.

For this the following modifications have been made:

- New Courier::CameraGroupHandler::OnViewInitContent method added for correctly initializing the cameras of a camera group, when their associated views are recreated.This method retrieves a camera if it has been lost, adds it to the camera group and makes all necessary settings for it to correctly be handled;
- New Courier::ViewScene::ActivateForCamera method has been added. This method activates / deactivates the view for message receiving, regardless of its rendering state and manages a reference count for activation requests for a specific camera of the view;

- New Courier::ViewScene::EnableRenderingForCamera method has been added. This method enables / disables rendering for the current ViewScene and a specific camera and manages a reference count for rendering requests for a specific camera of the view;
- Courier::ViewScene::IsRenderingEnabled method has been moved from Courier::ViewScene2D and Courier::ViewScene3D to Courier::ViewScene. This method returns true if the view is enabled for rendering;
- New Courier::ViewScene::OnActivate protected method has been added. This method delegates Courier::ParentViewActivateEvent event to controller and widgets;
- New protected methods Courier::ViewScene2D::ActivateImpl and Courier::ViewScene3D::ActivateImpl were added. These are actual implementations for Courier::ViewScene2D::Activate and Courier::ViewScene3D::Activate methods;
- New protected methods Courier::ViewScene2D::EnableRenderingImpl and Courier::ViewScene3D::EnableRenderingImpl were added. These are actual implementations for Courier::ViewScene2D::EnableRendering and Courier::ViewScene3D::EnableRendering methods;
- Courier::ViewScene2D::EnableRenderingScene and Courier::ViewScene3D::EnableRenderingScene have been made protected;
- New protected and virtual method Courier::ViewScene::UpdateCameraRenderingState has been added. This is overridden by Courier::ViewScene2D::UpdateCameraRenderingState and Courier::ViewScene3D::UpdateCameraRenderingState methods. These methods set the rendering state for the actual camera from Candera.

3.0.0-4 Render Configuration

A new option Courier::RenderConfiguration::ClearRenderTargetOnUpload was added for configurating the rendering.

This option enables / disables the automatic clearing of the render targets on the next upload. By default, to keep backwards compatibility, this option is set to false.

The interface of Courier::Gdu class has also been modified to support the above change. Therefore Courier::Gdu::Upload and Courier::Gdu::Unload now have a *renderer* parameter which is a pointer to Courier::Renderer.

Also Courier::Renderer::DeleteRenderJobs method has been made public, so it can be used by Courier::Gdu class.

3.0.0-5 Visualization

1. A new Courier::ViewAction::ClearCameraViewports is added to the Courier::ViewReqMsg. This new Courier::ViewAction::ClearCameraViewports will only clear the areas that are rendered by the cameras of the view.

For this, the following changes have been made:

- a new enumeration value has been added to Courier::ViewAction;
- a new *onlyCameraViewPorts* parameter has been added to Courier::View::Clear, Courier::ViewContainer::Clear, Courier::ViewScene2D::Clear and Courier::ViewScene3D::Clear methods.

2. New Courier::Renderer::SetIs2DRequest method has been added to Courier::Renderer, which sets a flag for determining if load/unload on a certain layer is requested by a 2D View. Corresponding getter method was also provided.

FeatStd V3.0.2

FeatStd CMake

Location of generated files

Automatically generated configuration headers are generated into `CMAKE_CURRENT_BINARY_DIR/gensrc/FeatStd` (i.e in most cases `CMAKE_BINARY_DIR/FeatStd/gensrc/FeatStd`) instead of `CMAKE_BINARY_DIR/gensrc/FeatStd`. Automatic CGI-Studio includes paths have been adapted to accommodate this change.

Add custom config header file

New build setting FEATSTD_CUSTOM_CONFIG to specifiy a custom config header file that gets included on top of generated file `CMAKE_CURRENT_BINARY_DIR/gensrc/FeatStd/Config.h`. This allows to add or override global CGI system settings (e.g. macros).

Example:

`FEATSTD_CUSTOM_CONFIG="path/to/MyConfig.h"`

MyConfig.h

```
#define FEATSTD_DEBUG_ASSERT(condition) \
    if (!(condition)) { \
        FeatStd::Internal::Generic::assert((#condition), __FILE__, __LINE__, false); \
    }
```

FeatStd Diagnostics

Debug Macros

The following macros can get overridden for custom needs:

- FEATSTD_DEBUG_BREAK
- FEATSTD_DEBUG_ASSERT
- FEATSTD_DEBUG_FAIL
- FEATSTD_DEBUG_REENTRANCE_GUARD

If these macros are not defined in custom header file the default implementation is taken, e.g. (code as in FeatStd/Diagnostics/Debug.h)

```
#if !defined(FEATSTD_DEBUG_ASSERT)              // <-- allows to override in custom header file
    #if defined(FEATSTD_DEBUG)

        FEATSTD_LINT_MACRO_WHILEFALSE(FEATSTD_DEBUG_ASSERT)
        #define FEATSTD_DEBUG_ASSERT(condition) \
                do { \
                    if (!(condition)) { \
                        if (FeatStd::Diagnostics::DebugControl::Assert((#condition), \
                                                                       __FILE__, __LINE__)) { \
                            FEATSTD_OS_BREAK(); \
                                      } \
                                } \
                            FEATSTD_SUPPRESS_MSC_WARNING_FOR_NEXT_EXPRESSION(4127, while (false)
                        } while (false)
    #else
        #define FEATSTD_DEBUG_ASSERT(condition)
    #endif
#endif
```

See also:

Add custom config header file

FeatStd Util

UTF8 String Constructor

The String class constructor now has an optional parameter, to specify whether the length of a non-null-terminated UTF8 string with multi-byte characters is given in bytes or in code-points.

SceneComposer V3.0.2

Restructuring Sources

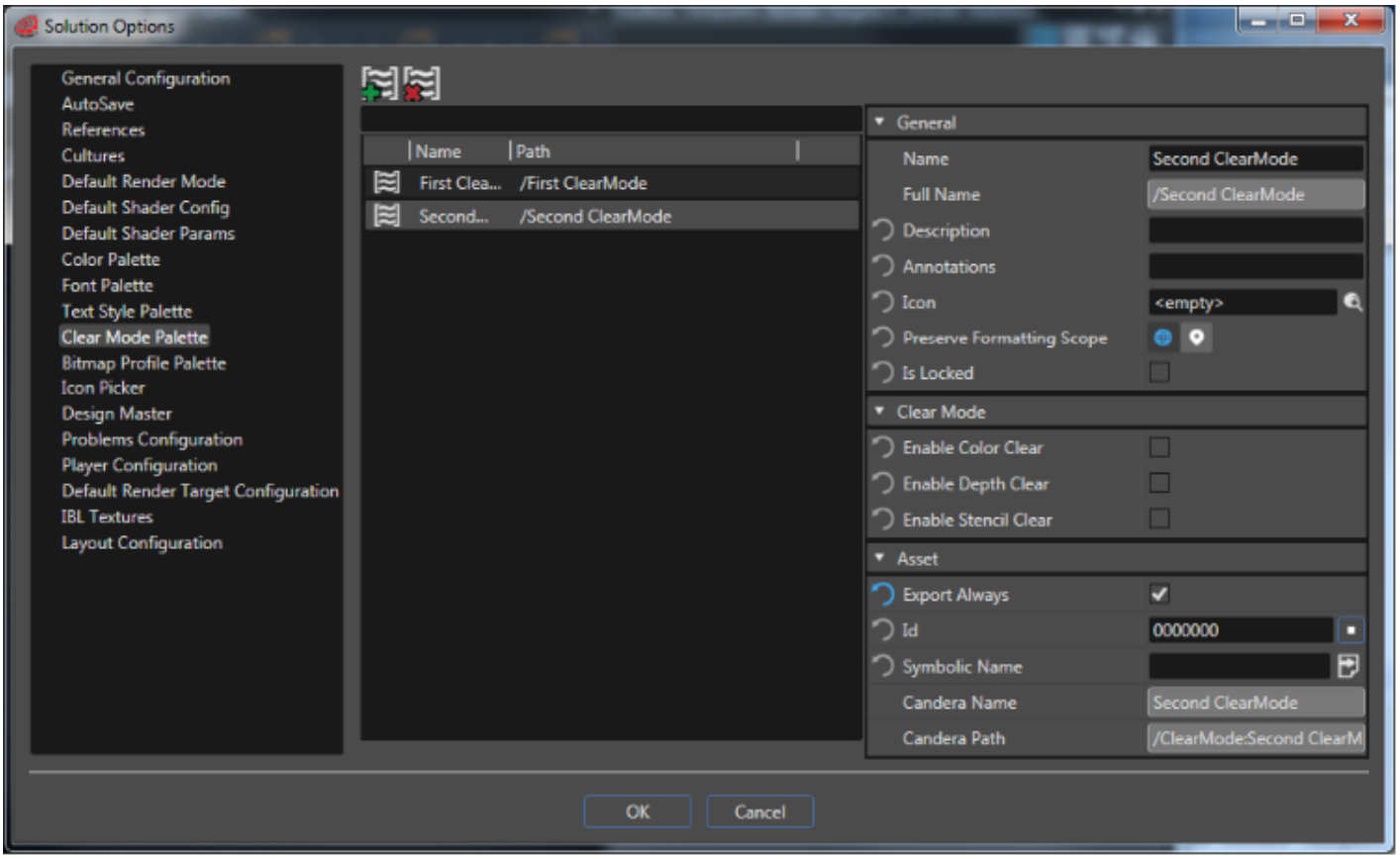
There were 2 major changes in source code organization:

- Decoupling: SCHost projects which produce the SCHost.lib were decoupled from cgi_studio_scenecomposer to a separate repository. Also, some projects were renamed: CommonLib -> Common, SCHostLib -> SCHostApp, SCImporterLib -> MffWriter. The new repository is called cgi_studio_schost and can be configured and build separately to obtain the SCHost.lib required by custom environments that create SCHost.dll files.
- 2D/3D separation: SCHost can be compiled/built now with only CANDERA_2D_ENABLED or CANDERA_3D_ENABLED, as required by the selected platform.

SceneComposer Usability Improvements

Clear Mode Palette

A ClearMode Palette View was added in Solution Options. By using it ("File" > "Solution Options"), it is possible to create different clear mode items which can be saved and later reused.

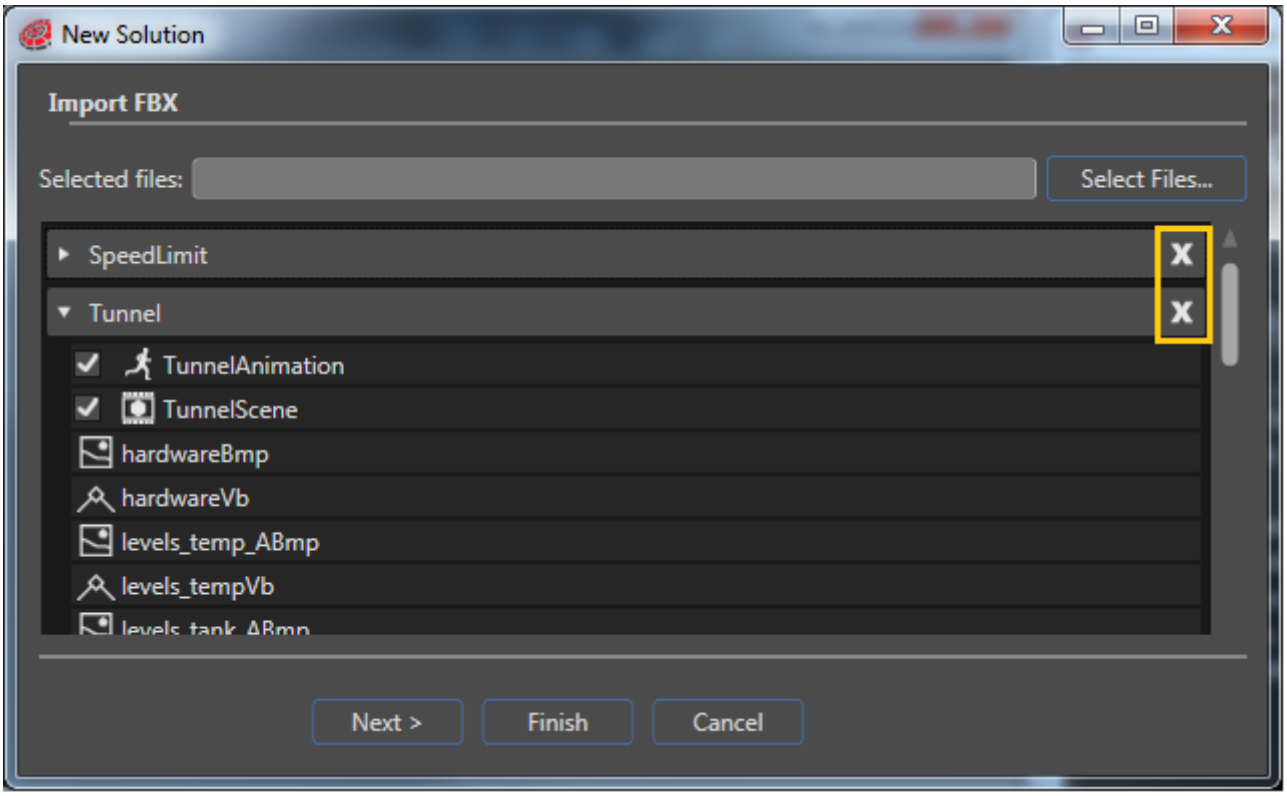


See also:

- Clear Mode in SceneComposer User Manual

Delete Items

In the ImportView of the "New Solution Wizard", it is now possible to delete selected items before they are imported.

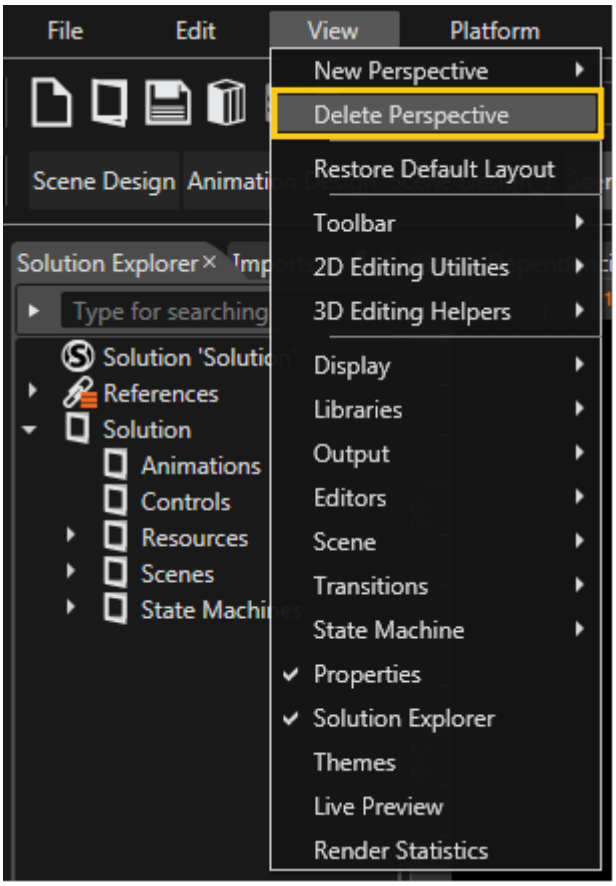


See also:

- New Solution Wizard in SceneComposer User Manual

Delete Perspective

The option "Delete Current Perspective" inside "View->New Perspective" is available (at the same level with "New Perspective").



See also:

- Perspective Support in SceneComposer User Manual

Search Option for Hot-Keys Panel

Due to the huge number of available hot-keys (shortcuts), it could be difficult to find a specific hot-key. In order to solve this issue, a search option was made available in the upper side of the Hot Key panel ("Help" > "Hot Keys").

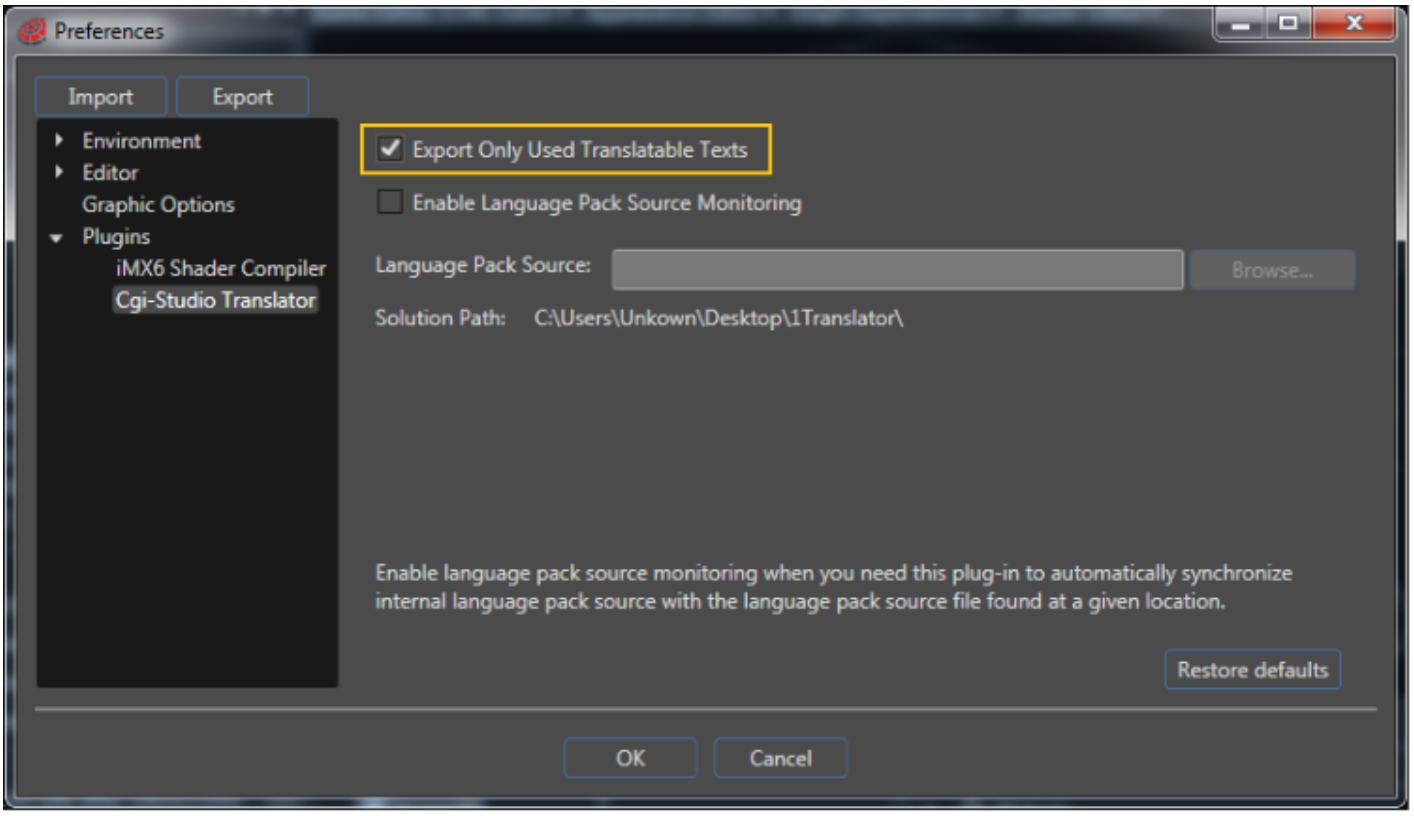


See also:

- Shortcut Customization in SceneComposer User Manual

Export Only Used Translatable Texts

In the previous versions of SC, the unreferenced text ids were not exported into the asset. To allow the export of all the text ids even if they are not used, a special option was added. Entitled "Export Only Used Translatable Texts" this option is available as a check box in the "File" > "Preferences" > "Plugins" > "Cgi-Studio Translator" panel.

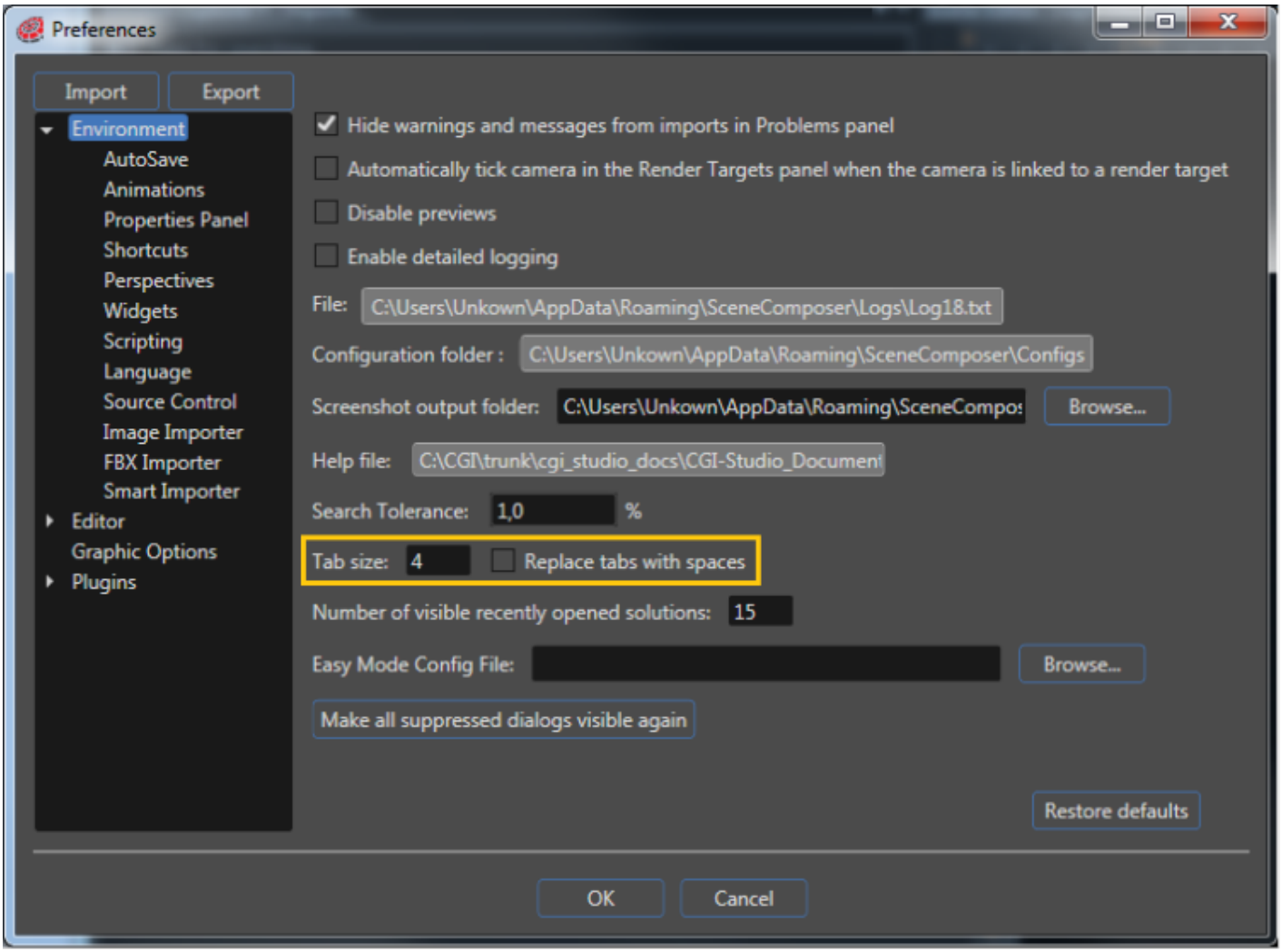


If the option is used there will be exported just the referenced text ids. If the option is left as default, all the text ids will be exported into the asset.

> The plugin does not work in order to receive the notifications if the "Enable Plugin Notification" is not used when the asset is generated.

Tab Size Configuration

Tab size configuration was added for the Shader Editor panel ("View" > "Editors" > "Shader Editor"). The user can change the tab size and decide if tabs should be changed to spaces from "File" > "Preferences" > "Environment". (Mantis issue 3724)



See also:

- Tab Options in SceneComposer User Manual

FTC Command Wildcard Support

If there are many files (i.e. images) which must be imported, an improvement - wildcard support - can help us to import them simultaneously. For instance, if in a folder we have many graphic files named "img1.png", "img2.png", "img3.png" ..., and if it is necessary to import all of them we can do this by specifying through the wildcard (*) that we need to import simultaneously the files which respect the rule "img*.png".

To use the FTCCmd extension for wildcard image import with option to set bitmap attributes, following option is available:

```
FTCCmd ... /BitmapProfile <bitmap_profile_full_name>
```

See also:

- FTCCmd Operation Mode Import in SceneComposer User Manual

Composites with Widgets

Created composites will include widgets with the observation that the widgets will be included only if are binded to a node from the created composite.

Minor Improvements

- **Check/Uncheck Licenses.** The possibility to check/uncheck all licences was added.
- **Buttons Highlighted.** The "New Solution Dialog" wizard and the "Change Platform" wizard have a button highlighted in order to suggest the action performed by pressing the Enter key.
- **New Tooltip.** The tooltip for the "Revert" button in SCML Editor was modified to "Discard changes".
- **Time Displayed.** Now, the "Output" panel also displays the time of any given message.
- **Screen Space Management.** In order to improve the screen space, the "Close" button ("X") has become visible only for the selected panels.
- **Zoom Buttons.** The zoom buttons in "Animation Curve Editor" were replaced for better visibility.
- **New Options in Scene Editor.** The possibility to Delete, Copy and Paste was added in Scene Editor through context menu commands and keyboard.
- **Reorganizing Multiple Selected Elements.** When reorganizing multiple selected elements, the focus will be kept on all elements that were multi-selected.
- **New Shortcuts.** New key shortcuts were added for the tree panels. CTRL + A selects all the visible elements and SHIFT + Home / End selects all the elements which are in visible level from the selected item to the beginning / end.
- **Just One Animation.** The same animated property isn't allowed to be added for the same object in an animation more than one time, on the same channel (Mantis 3722).

- **New Commands Added.** Export All and Clear commands are available in the context menu in CanderaLoggingView (Mantis 4593).
- **Activate Animation.** Double click on an AnimationNode from an AnimationGroup will activate the corresponding animation (Mantis 5049).
- **Rearrange the Perspectives.** The user has the possibility to rearrange the perspectives through drag and drop.
- **Text Search in Shader Editor.** A new option was added in the Text Search panel, for search only in the ShaderEditor (Mantis 4637).
- **Launcher Improvements.** SCLauncher horizontal scroll bar is now hidden. Vertical scrollbar is always visible and active. All checked requirements are now shown and the text has been reformatted (Mantis 2820).
