Appendix

CMake Flags

The following CMake flags can be seen when building an application using the CMake-GUI. Be aware that some defines require other defines to be set first in order to be displayed in the CMake-GUI.

Candera flags

- **CANDERA_3D_CANVAS_ENABLED** defines whether Candera3D Canvas is enabled or not.
- **CANDERA_ASSETLOADER_MEMORYPOOL_ENABLED** defines whether asset loader objects are allocated from a dedicated memory pool or not.
- **CANDERA_ASSETLOADER_WORKER_THREAD_ENABLED** uses an own worker thread for asynchronous asset loader calls.
- **CANDERA_ASSET_CACHE_SIZE** defines the buffer size for asset caching. A memory chunk of this size will be allocated for caching asset chunks for fast access.
- **CANDERA_ASSET_DECOMPRESSION_CHUNK_SIZE** defines the chunk buffer size for asset decompression. This value is only relevant for File Asset Repositories.
- **CANDERA_BIDIRECTIONAL_TEXT_ENABLED** defines whether bidirectional text parsing is enabled or not.
- **CANDERA_CUSTOM_LOCALIZER_ENABLED** defines whether or not a custom Localizer must be used instead of the DefaultLocalizer.
- **CANDERA_EXTERNAL_SHADER_BUILDER_HEADER_INCLUDE** includes path to external CPP header file, referenced by CANDERA_EXTERNAL_SHADER_BUILDER_HEADER_PATH.
- **CANDERA_EXTERNAL_SHADER_BUILDER_HEADER_PATH** defines path to external CPP header file with BUILD_SHADER macro definition, relative to Common/Internal/RenderDevice2DOver3D/.
- **CANDERA_EXTERNAL_SHADER_BUILDER_SOURCE_PATH** defines path to external CPP source file with binary shaders data, relative to Common/Internal/RenderDevice2DOver3D/.
- **CANDERA_FONTENGINE** defines which font engine to use.
- **CANDERA_GLOBALIZATION_ENABLED** defines whether globalization support is enabled or not.
- **CANDERA_LAYOUT_CLIPPING_ENABLED** defines whether Candera2D layout clipping (setting a clipping rectangle for every render node) is enabled or not.
- **CANDERA_LAYOUT_ENABLED** defines whether Candera dynamic layout is enabled or not.
- **CANDERA_LEGACY_BEHAVIORS_ENABLED** builds with legacy behaviors.
- **CANDERA_MULTILINE_TEXT_ENABLED** defines whether multiline texts are enabled or not.
- **CANDERA_SCRIPTING_ENABLED** defines whether CanderaScripting sources shall be included or not.
- **CANDERA_TEXTENGINE_DLIG_FEATURE_ENABLED** defines if Discretionary Ligatures (dlig) are enabled or not (support only in Harfbuzz for now).
- **CANDERA_TEXTENGINE_MEMORYPOOL_ENABLED** defines whether text engine objects are allocated from a dedicated memory pool or not.

- **CANDERA_TEXTENGINE_NATIVE_LINEHEIGHT_ENABLED** defines whether or not the native line height metrics of the font engine is used. If disabled then 120% of EM size is applied as line height which is compatible with Adobe.
- **CANDERA_TEXTENGINE_WORKER_THREAD_ENABLED** uses an own worker thread for asynchronous textrender calls. It uses the update routine (single threaded) when this flag is disabled.
- **CANDERA_TEXTSHAPER** selects the text shaper.
- **CANDERA_TRANSITIONS_ENABLED** defines whether CanderaTransitions sources shall be included or not.
- **CANDERA_VIDEO_MEMORY_STATISTIC_ENABLED** activates support for video memory allocation statistics. ATTENTION: Activated memory statistics may decrease performance!
- **CANDERA_ASSET_ADDRESS** is the address for eMMC / QSPI flash access
- **CANDERA_ASSET_ADDRESS_USE_DEFAULT** defines the default address for flash access
    - 0x00000000 for eMMC
    - 0x2000000 for QSPI
- **CANDERA_ASSET_LOCATION** is the asset location chosen by the user at CMake configure time
- **CANDERA_BUILD_SCHOST** defines whether SCHost library and DLL are built or not
- **CGIDEVICE_BSP_LIBRARY_PATH** defines the path to directory including libfbdev.a (for RCarD3, RCarH3 and RCarM3 devices)
- **CGIDEVICE_EGL_INCLUDE_PATH** defines path to EGL folder
- **CGIDEVICE_GFX_LIBRARY_PATH** defines the path to graphics libraries (for RCarD3, RCarH3 and RCarM3 devices)
- **CGIDEVICE_OPENGL_EGL_CONFIG_PATH** defines the path to OpenGL and EGL configuration files (for RCarD3 and RCarH3 devices)
- **CGIDEVICE_OPENGL_ES_30_INCLUDE_PATH** defines the path to GLES3 folder
- **CANDERA_EXTERNAL_SHADER_BUILDER_USAGE_DEFAULT** defines whether the default paths for external shader builder for 2DOver3D shaders are used or not."

Courier flags

- **COURIER_DATA_LOCK_ENABLED** enables the data lock.
- **COURIER_ENHANCED_ENABLED** enables data binding and visualization, which enables Candera graphics engine support.
- **COURIER_ENHANCE-SERIALIZATION_ENABLED** activates message serialization.
- **COURIER_MESSAGING_MONITOR_ENABLED** enables messaging monitor.
- **COURIER_PLATFORM** defines the platform Courier is built for.
- **COURIER_PLATFORM_CONFIG** defines the platform configuration Courier is built for.
- **COURIER_PREBUILT_LIB_DIR** sets the location of pre-built Courier library root folder. Compiler and build configuration are added.
- **COURIER_RENDERING_MONITOR_ENABLED** enables rendering monitor.
- **COURIER_USE_PREBUILT_LIBS** uses pre-built Courier libraries in solution instead of Courier sources.
- **COURIER_CMAKE_LOGGING_LEVEL** defines the verbosity level of Courier CMake system
- **COURIER_MESSAGE_SERIALIZATION_ENABLED** activates message serialization
- **COURIER_CUSTOM_INJECT_WIDGET_BASE_ENABLED** enables the use of customized includes.
- **COURIER_PLATFORMDEFS_DIR** sets customer specific search path for Courier platform definitions.
- **COURIER_SCRIPTING_ENABLED** enables Courier scripting system support.

FeatStd flags

- **FEATSTD_32BIT_PLATFORM** a 32 bit compiler is used.
- **FEATSTD_64BIT_PLATFORM** a 64 bit compiler is used.
- **FEATSTD_CUSTOM_CONFIG** adds custom config header file.
- **FEATSTD_GLERRORS_AND_EGLERRORS_ENABLED** enables glGetError and eglGetError calls.
- **FEATSTD_IPC_ENABLED** enables IPC functionality, like IPC channels, etc.
- **FEATSTD_LOGGER_MAX_BUFFER_SIZE** defines the maximum length of a single log message.
- **FEATSTD_LOG_ENABLED** enables logging functionality. If disabled all logging defines are expanded empty and all logging classes (e.g. Appender) will not be available. This means the application which uses e.g. Appender has to scope this usage with FEATSTD_LOG_ENABLED.
- **FEATSTD_MEMORYPOOL_ENABLED** enables optional component MemoryPool. MemoryPool is an optional component of FeatStd.
- **FEATSTD_MEMORYPOOL_FEATURES_DEBUG** defines MemoryPool features for debug build mode.
- **FEATSTD_MEMORYPOOL_FEATURES_MINSIZEREL** defines MemoryPool features for minimal size build mode.
- **FEATSTD_MEMORYPOOL_FEATURES_RELEASE** defines MemoryPool features for debug release mode.
- **FEATSTD_MEMORYPOOL_FEATURES_RELWITHDEBINFO** defines MemoryPool features for release with debugging info build mode.
- **FEATSTD_MONITOR_ENABLED** enables monitor module.
- **FEATSTD_STRINGBUFFER_APPENDER_CANDERATYPESIZE** defines the maximum size of one Candera type used with string buffer appender.
- **FEATSTD_STRINGBUFFER_APPENDER_ENABLED** enables the additional StringBuffer AppendObject method and the StringBufferAppender template. StringBufferAppender implementations for specific types like enum, struct or class types have to be provided by the more specific library/application code (manually or generated).
- **FEATSTD_THREADSAFETY_ENABLED** activates thread safety for various FeatStd modules like logging, etc.
- **FEATSTD_UNITTEST_FRAMEWORK_ENABLED** activates unit test framework support.
- **FEATSTD_FILE_** usually should be '__FILE__' but for some special gcc build it could be set to '__BASE_FILE__'.
- **CGIAPP_ENABLE_ADDITIONAL_BEHAVIORS** defines whether additional behaviors are enabled or not
- **FEATSTD_CMAKE_LOGGING_LEVEL** set verbosity level of FeatStd CMake system

Monitor flags

- **MONITOR_BLOCK_UNTIL_CONNECT** enables waiting until host connects to target.
- **MONITOR_CANDERA_CUSTOM_EXPERIMENTS** enables custom global experiments.
- **MONITOR_CANDERA_PERFORMANCE_RECORDER_ENABLED** defines whether or not to expand performance recocrder macros.
- **MONITOR_CANDERA_PERFORMANCE_RECORDER_GENERATION_DIR** sets the location of the Candera performance recorder generation.
- **MONITOR_CANDERA_PERFORMANCE_RECORDER_TEMPLATE_DIR** sets the location of the Candera performance recorder template.
- **MONITOR_COM_TYPE** sets the type of monitoring connection.
- **MONITOR_FRAME_DEBUGGER_ENABLED** enables basic frame debugger (frame timeline capture, single step mode, no OpenGL specific features).
- **MONITOR_FRAME_DEBUGGER_OPENGL_CAPTURE_ENABLED** enables OpenGL specific Frame Debugger features.
- **MONITOR_MEMORYPOOL_BINARY_LOGGING_ENABLED** activates the Memory Pool activity logging.
- **MONITOR_MEMORYPOOL_BINARY_LOGGING_FILENAME** location where the log-file will be generated. The extension has to be *.membin.
- **MONITOR_RECORDING_2D_CALLS_ENABLED** enables all 2D driver calls.
- **MONITOR_RECORDING_OPENGL_ENABLED** enables all OpenGL Recorder (e.g. Timing Recorder of OpenGL).
- **MONITOR_SINGLE_STEP_INTERVAL** sets the time in milliseconds that is used when incrementing the animation time in single-step mode.
- **MONITOR_TCPIP_ADDRESS** sets the adress of target device which is a server. Therefore this is the listening ip 0.0.0.0 and enables listening to any address.
- **MONITOR_TCPIP_PORT** sets the TcpIp port number.

CGIApp flags

- **CGIAPP_ADDITIONAL_BEHAVIOR_PATH** sets the file containing additional behaviors paths.
- **CGIAPP_ADDITIONAL_WIDGET_PATH** sets the path to a txt file.
- **CGIAPP_ENABLE_ADDITIONAL_WIDGETS** points to a custom provided additional widgets file.
- **CGIAPP_WIDGETSET_VERSION** sets the version of the widget.
- **CGIAPP_ENABLE_TUTORIAL** defines whether CGI-Application tutorial is enabled or not (CGI Studio Documentation->Application Development->Candera->Tutorials)
- **CGIAPP_ENABLE_ADDITIONAL_BEHAVIORS** defines whether Behaviors feature is enabled or not
- **CGIAPP_BUILD_TYPE** defines the binary type of the build application. It can be set to NativeDll or NativeExe.
- **CGIAPP_PLAYER_DEBUG_PATH** defines the path to the CGI Player executable for debug builds. On simulation, for example, the executable will be stored in the device directory present next to the CGIPanel.exe.
- **CGIAPP_PLAYER_RELEASE_PATH** is the path to the CGI Player executable for release builds. On simulation, for example, the executable will be stored in the device directory present next to the CGIPanel.exe.
- **CGIAPP_SAMPLE_CLOCK_ENABLED** enables the clock sample feature. If enabled the application will be built with the Clock sample component allowing to use the AnalogWatch control when its Time property is bound to the /Time/CurrentTime data item.
- **CGIAPP_SAMPLE_DYNAMICLIST_ENABLED** enables the dynamic list sample feature.
- **CGIAPP_SCENE_COMPOSER_PATH** defines the path to Scene Composer.
- **CGISTUDIO_FORCE_RESET_USER_SETTINGS** forces the generation of MSVC user settings even if they exist.
- **SCHOST_DLL_NAME_CUSTOM** defines a custorm name for SCHost DLL
- **SCHOST_DLL_NAME_PLATFORM_SPECIFIC** defines whether or not to use platform specific name for SCHost.dll
- **SCHOST_DLL_OUTPUT_PATH** defines the output path for SCHost.dll. Typically is the path to the Scene Composer.

Control Behaviors Flags

- **CONTROLS_BREADCRUMB_BEHAVIORS_ENABLED** - Build with BreadCrumb Behaviors (BreadcrumbActionBehavior, BreadcrumbBehavior)
- **CONTROLS_DEMO_BEHAVIORS_ENABLED** - Build with Behaviors intended for demonstration purpose (ValueGeneratorBehavior)
- **CONTROLS_EDITTEXT_BEHAVIORS_ENABLED** - Build with Edit Text Behaviors (CaretBehavior, EditTextBehavior, EnterExitConditionBehavior)
- **CONTROLS_EXTERNALRESOURCE_BEHAVIORS_ENABLED** - Build with External Resource Behaviors (ExternalBitmapProviderBehavior, ExternalResourceChangeConditionBehavior, SetExternalBitmapActionBehavior)
- **CONTROLS_GLOBALIZATION_BEHAVIORS_ENABLED** - Build with Globalization Behaviors (CultureChangeConditionBehavior, SetCultureActionBehavior)
- **CONTROLS_GRAPH_BEHAVIORS_ENABLED** - Build with Graph Behaviors (BarGraphBehavior, LineGraphBehavior)
- **CONTROLS_HOVER_EVENT_ENABLED** - Activate hover events from Control Touch Session
- **CONTROLS_KEYBOARD_BEHAVIORS_ENABLED** - Build with Keyboard Behaviors (KeyboardActionBehavior, KeyboardBehavior)
- **CONTROLS_MAP_BEHAVIORS_ENABLED** - Build with Map Behaviors (GeocodingBehavior, MapActionBehavior, MapBridgeBehavior, MapLocationActionBehavior, MapNavigationActionBehavior, NavigationInstructionViewBehavior)
- **CONTROLS_MENU_BEHAVIORS_ENABLED** - Build with Menu Behaviors (MenuItemBehavior, MenuScreenActionBehavior, MenuScreenBehavior, PageIndicatorActionBehavior, PageIndicatorBehavior)
- **CONTROLS_MESH_BEHAVIORS_ENABLED** - Build with Mesh2D Behaviors (TransformMesh2DBehavior)
- **CONTROLS_MIXEDREALITY_BEHAVIORS_ENABLED** - Build with Mixed Reality Behaviors (NavigationPathBehavior, PoiBehavior, PoiConfigBehavior, PoiManagerBehavior, PoseTrackingBehavior)
- **CONTROLS_PARTICLE_EMITTER_BEHAVIORS_ENABLED** - Build with Particle Emitter Behaviors (ParticleEmitterBehavior)
- **CONTROLS_PLUGIN_CONFIGURATION_ENABLED** - Use the output from the "Used Behaviors Plugin". See Used Behaviors Plugin.

- **CONTROLS_ROLL_BEHAVIORS_ENABLED** - Build with Roll Behaviors (RollBehavior)
- **CONTROLS_SCRIPTING_BEHAVIORS_ENABLED** - Build with Scripting Behaviors (EnableScriptingSystemBehavior)
- **CONTROLS_SCROLL_LIST_BEHAVIORS_ENABLED** - Build with Scroll & List Behaviors (ScrollViewBehavior, ListBehavior, SnapListItemConfigurationBehavior, SnapListItemExtensionBehavior, SubListBehavior, ScrollableEventActionBehavior, ScrollbarBehavior)
- **CONTROLS_SELECTION_GROUP_BEHAVIORS_ENABLED** - Build with Selection Group Behaviors (ChangeSelectionActionBehavior, SelectionConditionBehavior, SelectionGroupBehavior)
- **CONTROLS_SLIDER_BEHAVIORS_ENABLED** - Build with Slider Behaviors (Slider, CircularSlider, Dial) (SliderBehavior, CircularSliderBehavior, DialBehavior)
- **CONTROLS_STREAMING_BEHAVIORS_ENABLED** - Build with Streaming Behaviors (VideoStateConditionBehavior, VideoStreamActionBehavior, VideoStreamBehavior)
- **CONTROLS_USED_BEHAVIORS_CONFIG_PATH** - Path to the output file of the "Used Behaviors Plugin". Flag is only available if flag *CONTROLS_PLUGIN_CONFIGURATION_ENABLED* is active. See Used Behaviors Plugin.

Third Party Software Flags

- **CGIDEVICE_TARGET_EGL_GLES2_IMPLEMENTATION** select target EGL and GLES2 implementation: libEGL (EGL and GLES2 directly) or GLX (EGL and GLES2 through OpenGLAdapter over GLX)

FreeType flags

- **FREETYPE2_CONFIGURE_CACHE** enables or disables freetype cache configuration.
- **FREETYPE2_FACE_COUNT_MAX** sets the maximum number of faces supported by cache and must be a constant integral C expression.
- **FREETYPE2_FACE_SIZE_COUNT_MAX** sets the maximum number of sizes supported by cache and must be a constant integral C expression.
- **FREETYPE2_GLYPH_CACHE_SIZE** defines the size of the buffer to cache glyph bitmaps and must be a constant integral C expression.
- **FREETYPE2_MODLOAD_DRIVER_BDF** defines whether the BDF (Bitmap Distribution Format) font driver module is loaded or not.
- **FREETYPE2_MODLOAD_DRIVER_CFF** defines whether the CFF font driver module is loaded or not.
- **FREETYPE2_MODLOAD_DRIVER_PCF** defines whether the PCF font driver module is loaded or not.
- **FREETYPE2_MODLOAD_DRIVER_PFR** defines whether the PFR font driver module is loaded or not.
- **FREETYPE2_MODLOAD_DRIVER_T1** defines whether the T1 font driver module is loaded or not.
- **FREETYPE2_MODLOAD_DRIVER_T1CID** defines whether the T1CID font driver module is loaded or not.
- **FREETYPE2_MODLOAD_DRIVER_T42** defines whether the T42 font driver module is loaded or not.
- **FREETYPE2_MODLOAD_DRIVER_TT** defines whether the TT font driver module is loaded or not.
- **FREETYPE2_MODLOAD_DRIVER_WINFNT** defines whether the WINFNT font driver module is loaded or not.
- **FREETYPE2_MODLOAD_MODULE_AUTOFIT** defines whether the autofitter (auto-hinter) module is loaded or not.
- **FREETYPE2_MODLOAD_MODULE_PSAUX** defines whether the helper module psaux is loaded or not.
- **FREETYPE2_MODLOAD_MODULE_PSHINTER** defines whether the module pshinter is loaded or not.
- **FREETYPE2_MODLOAD_MODULE_PSNAMES** defines whether the helper module psnames is loaded or not.
- **FREETYPE2_MODLOAD_MODULE_SFNT** defines whether the helper module sfnt is loaded or not.
- **FREETYPE2_MODLOAD_RENDERER_FT_RASTER1** defines whether the renderer module raster1 is loaded or not.
- **FREETYPE2_MODLOAD_RENDERER_FT_SMOOTH** defines whether the renderer module smooth is loaded or not.

- **FREETYPE2_MODLOAD_RENDERER_FT_SMOOTH_LCD** defines whether the renderer module smooth_lcd is loaded or not.
- **FREETYPE2_MODLOAD_RENDERER_FT_SMOOTH_LCDV** defines whether the renderer module smooth_lcdv is loaded or not.
- **FREETYPE2_OPT_AF_CONFIG_OPTION_CJK** compile autofit module with CJK (Chinese, Japanese, Korean) script support.
- **FREETYPE2_OPT_AF_CONFIG_OPTION_INDIC** compile autofit module with fallback Indic script support.
- **FREETYPE2_OPT_AF_CONFIG_OPTION_USE_WARPER** compile autofit module with warp hinting.
- **FREETYPE2_OPT_CFF_CONFIG_OPTION_DARKENING_PARAMETER_X1** set the default X value for the first control point that define the stem darkening behavior of the CFF engine.
- **FREETYPE2_OPT_CFF_CONFIG_OPTION_DARKENING_PARAMETER_X2** set the default X value for the second control point that define the stem darkening behavior of the CFF engine.
- **FREETYPE2_OPT_CFF_CONFIG_OPTION_DARKENING_PARAMETER_X3** set the default X value for the third control point that define the stem darkening behavior of the CFF engine.
- **FREETYPE2_OPT_CFF_CONFIG_OPTION_DARKENING_PARAMETER_X4** set the default X value for the forth control point that define the stem darkening behavior of the CFF engine.
- **FREETYPE2_OPT_CFF_CONFIG_OPTION_DARKENING_PARAMETER_Y1** set the default Y value for the first control point that define the stem darkening behavior of the CFF engine.
- **FREETYPE2_OPT_CFF_CONFIG_OPTION_DARKENING_PARAMETER_Y2** set the default Y value for the second control point that define the stem darkening behavior of the CFF engine.
- **FREETYPE2_OPT_CFF_CONFIG_OPTION_DARKENING_PARAMETER_Y3** set the default Y value for the third control point that define the stem darkening behavior of the CFF engine.
- **FREETYPE2_OPT_CFF_CONFIG_OPTION_DARKENING_PARAMETER_Y4** set the default Y value for the forth control point that define the stem darkening behavior of the CFF engine.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_ADOBE_GLYPH_LIST** enables the compilation of 'PSNames' with Adobe Glyph List.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_DISABLE_STREAM_SUPPORT** define to disable the use of file stream functions and types, FILE,fopen() etc. Enables the use of smaller system libraries on embedded systems that have multiple system libraries, some with or without file stream support, in the cases where file stream support is not necessary such as memory loading of font files.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_FORCE_INT64** enables the usage of 64-bit integers data types when __STDC__ macro is defined.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_INCREMENTAL** allows the use of FT_Incremental_Interface to load typefaces that contain no glyph data, but supply it via a callback function.

- **FREETYPE2_OPT_FT_CONFIG_OPTION_INLINE_MULFIX** enables the usage of an inlined assembler version of the `FT_MulFix' function.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_MAC_FONTS** enables support for outline fonts in Mac format (mac dfont, mac resource, macbinary containing a mac resource) on non-Mac platforms.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_NO_ASSEMBLER** prevents the usage of an assembler version of performance-critical functions (e.g. FT_MulFix).
- **FREETYPE2_OPT_FT_CONFIG_OPTION_OLD_INTERNALS** this macro is obsolete. Support has been removed in FreeType version 2.5.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_PIC i**f set FreeType2 will avoid creating constants that require address fixups.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_POSTSCRIPT_NAMES** defines whether or not to load and enumerate the glyph Postscript names in a TrueType or OpenType file.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_SUBPIXEL_RENDERING** activate LCD rendering technology similar to ClearType in the build of the library.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_SYSTEM_ZLIB** allows FreeType's 'ftgzip' component to link to the system's installation of the ZLib library.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_USE_BZIP2** enables FreeType to handle font files that have been compressed with the 'bzip2' program.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_USE_HARFBUZZ** enables FreeType to use the HarfBuzz library to improve auto-hinting of OpenType fonts.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_USE_LZW** enables FreeType to handle font files that have been compressed with the ‘compress’ program.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_USE_MODULE_ERRORS** if enabled change the error codes so the higher byte gives the module in which the error has occurred, while the lower byte is the real error code.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_USE_PNG** enables FreeType to handle the loading of color bitmap glyphs in the PNG format.
- **FREETYPE2_OPT_FT_CONFIG_OPTION_USE_ZLIB** enables FreeType to handle font files that have been compressed with the 'gzip' program.
- **FREETYPE2_OPT_FT_DEBUG_AUTOFIT** if defined, FreeType provides some means to control the autofitter behaviour for debugging purposes with global boolean variables.
- **FREETYPE2_OPT_FT_DEBUG_LEVEL_ERROR** enables the build of FreeType library in debug mode. FREETYPE2_OPT_FT_DEBUG_LEVEL_TRACE enables the build of FreeType library in trace mode.
- **FREETYPE2_OPT_FT_DEBUG_MEMORY** enables the compilation of FreeType library with an integrated memory debugger that is capable of detecting simple errors like memory leaks or double deletes.
- **FREETYPE2_OPT_FT_MAX_MODULES** defines the maximum number of modules that can be registered in a single FreeType library object (32 is the default).
- **FREETYPE2_OPT_FT_RENDER_POOL_SIZE** defines the size in bytes of the render pool used by the scan-line converter to do all of its work.
- **FREETYPE2_OPT_T1_CONFIG_OPTION_NO_AFM** prevents the compilation of ‘t1afm’, which is in charge of reading Type 1 AFM files into an existing face.
- **FREETYPE2_OPT_T1_CONFIG_OPTION_NO_MM_SUPPORT** prevents the compilation of the Multiple Masters font support in the Type 1 driver.

- **FREETYPE2_OPT_T1_MAX_CHARSTRINGS_OPERANDS** defines the charstring stack's capacity (minimum 16 is required).
- **FREETYPE2_OPT_T1_MAX_DICT_DEPTH** defines the maximum depth of nest dictionaries and arrays in the Type 1 stream (minimum 4 is required).
- **FREETYPE2_OPT_T1_MAX_SUBRS_CALLS** defines the maximum number of nested sub-routine calls during glyph loading.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_0** enables support for TrueType CMap table format 0.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_10** enables support for TrueType CMap table format 10.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_12** enables support for TrueType CMap table format 12.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_13** enables support for TrueType CMap table format 13.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_14** enables support for TrueType CMap table format 14.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_2** enables support for TrueType CMap table format 2.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_4** enables support for TrueType CMap table format 4.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_6** enables support for TrueType CMap table format 6.
- **FREETYPE2_OPT_TT_CONFIG_CMAP_FORMAT_8** enables support for TrueType CMap table format 8.
- **FREETYPE2_OPT_TT_CONFIG_OPTION_BDF** defines whether to include or not support for an embedded ‘BDF’ table within SFNT-based bitmap formats.
- **FREETYPE2_OPT_TT_CONFIG_OPTION_BYTECODE_INTERPRETER** defines whether the TrueType driver should be compiled or not with a bytecode interpreter.
- **FREETYPE2_OPT_TT_CONFIG_OPTION_COMPONENT_OFFSET_SCALED** defines whether to use or not the Apple's definition of how to handle component offsets in composite glyphs while compiling the TrueType glyph loader.
- **FREETYPE2_OPT_TT_CONFIG_OPTION_EMBEDDED_BITMAPS** defines whether to support or not embedded bitmaps in all formats using the SFNT module (namely TrueType & OpenType).
- **FREETYPE2_OPT_TT_CONFIG_OPTION_GX_VAR_SUPPORT** defines whether to include or not support for Apple's distortable font technology (fvar, gvar, cvar, and avar tables).
- **FREETYPE2_OPT_TT_CONFIG_OPTION_MAX_RUNNABLE_OPCODES** defines the maximum number of bytecode instructions executed for a single run of the bytecode interpreter, needed to prevent infinite loops.
- **FREETYPE2_OPT_TT_CONFIG_OPTION_POSTSCRIPT_NAMES** defines whether or not to load and enumerate the glyph Postscript names in a TrueType or OpenType file.
- **FREETYPE2_OPT_TT_CONFIG_OPTION_SFNT_NAMES** enables the application to access the internal name table in a SFNT-based format like TrueType or OpenType. FREETYPE2_OPT_TT_CONFIG_OPTION_SUBPIXEL_HINTING defines whether to compile or not subpixel hinting support into the TrueType driver.

Other flags

- **CGICODECHECK_PC_LINT_CONFIG_DIR** sets the full path to the directory containing PC-Lint configuration files.
- **CGICODECHECK_PC_LINT_CUSTOM** enables custom rules for linting.
- **CGICODECHECK_PC_LINT_CUSTOM_CONFIG_PATH** sets the full path to the custom PC-Lint configuration directory.
- **CGICODECHECK_PC_LINT_ENABLED** defines whether or not enabling PC-Lint code checks.
- **CGICODECHECK_PC_LINT_EXECUTABLE_PATH** sets the full path to the PC-Lint executable, NOT to the generated lin.bat.
- **CGIDEVICE_RCARD1_USE_DRW2D** can be used to switch between using the Draw 2D API for DaveHD or using 2D over 2D using OpenGL directly.
- **CGISTUDIO_BUILD_LUA** define whether or not the LUA library is enabled.
- **CMAKE_CONFIGURATION_TYPES** defines a semicolon separated list of supported configuration types: Debug, Release, MinSizeRel and RelWithDebInfo. Anything else will be ignored.
- **CMAKE_INSTALL_PREFIX** sets the install path prefix, prepended onto install directories.
- **CMAKE_VERBOSE_CGISTUDIO_LOG** defines whether or not printing CGI-Studio Cmake debug logs.
- **SCHOST_BASE_PATH** sets the location of cgi_studio_schost.
- **CMAKE_AR** defines path to “ar” (compiler archiver)
- **CMAKE_ASM_FLAGS** defines compilation flags to get support for assembler via compiler
- **CMAKE_BUILD_TYPE** defines the built type (Debug, Release ...)
- **CMAKE_COLOR_MAKEFILE** enables color output when using the Makefile generator
- **CMAKE_LINKER** defines the path for linker
- **CMAKE_MAKE_PROGRAM** defines the path to the native build system
- **CMAKE_NM** defines the path to "nm" (a tool used to examine binary files)
- **CMAKE_OBJCOPY** defines the path to “objcopy" (utility that copies an object file to another)
- **CMAKE_OBJDUMP** defines the path to “objdump” (objdump displays information about one or more object files)
- **CMAKE_PROJECT_NAME** defines the name of the current project
- **CMAKE_SH** defines the path to “sh”
- **CMAKE_SKIP_INSTALL_RPATH** defines whether include or not RPATHs in the install tree
- **CMAKE_SKIP_RPATH** defines whether add or not run time path information
- **CMAKE_STATIC_LINKER_FLAGS** defines the linker flags to be used to create static libraries
- **CMAKE_STATIC_LINKER_FLAGS_DEBUG** defines the flags used by the linker during debug builds

- **CMAKE_STATIC_LINKER_FLAGS_MINSIZEREL** defines the flags used by the linker during release minsize builds
- **CMAKE_STATIC_LINKER_FLAGS_RELEASE** defines the flags used by the linker during release builds
- **CMAKE_STATIC_LINKER_FLAGS_RELWITHDEBINFO** defines the flags used by the linker during Release with Debug Info builds
- **CMAKE_STRIP** defines the path to the “strip” utility (removes inessential information from executable binary programs and object files)
- **CMAKE_TOOLCHAIN_FILE** defines the path to the toolchain file (it contains paths to the utilities to compile, link libraries and create archives, and other tasks to drive the build)
- **CMAKE_VERBOSE_MAKEFILE** defines whether enable or disable verbose output from Makefile builds
- **CMAKE_CXX_FLAGS_DEBUG** defines the CXX debug flags.
- **CMAKE_CXX_FLAGS_MINSIZEREL** defines the CXX Minimum Size Release Flags
- **CMAKE_CXX_FLAGS_RELEASE** defines the CXX release flags.
- **CMAKE_CXX_FLAGS_RELWITHDEBINFO** defines the CXX release with debug info flags.
- **CMAKE_C_FLAGS_DEBUG** defines the C debug flags.
- **CMAKE_C_FLAGS_MINSIZEREL** defines the C minimum size release flags.
- **CMAKE_C_FLAGS_RELEASE** defines the C release flags.
- **CMAKE_C_FLAGS_RELWITHDEBINFO** defines the C release with debug info flags.
- **CMAKE_RC_COMPILER** defines path to the resource compiler.
- **FEATSTD__FILE__** defines the “__FILE__” macro (for some special gcc build it could be set to ‘__BASE_FILE__’)
- **INTEGRITY_OS_PATH** defines the root path to Integrity OS installation.
- **MLC_SIMULATION** defines whether to build or not the project without statemachine and allow direct connection to Matlab.
- **MULTI_TOOLCHAIN_PATH** defines the Root path to MULTI tool chain.
- PKG_CONFIG_EXECUTABLE
- **CMAKE_INT_FLAGS** defines the flags for the Integrate tool.
- **CMAKE_INTEGRATE** defines the path to the Integrate tool (intex.exe).
- **CMAKE_MEM** defines the path to the Gmem tool (gmemfile.exe).

Candera Memory Pools Analysis

Memory Pools Configuration

Introduction

If free memory in backing heap is exhausted, "Out of Memory" occurs. The possible reasons are as the below.

- Fragmentation of allocated blocks, not possible to fit in a new block.
- Initial analysis was wrong, configuration does not fit.
- “Unexpected” allocations.
- Executed use case cannot fit into supplied memory.

Please see the following pages:

- Memory Pool Configuration
- “Unexpected” Allocations

Memory Pool Configuration

Fragmentation

- A heap will fragment, when buffers with different lifetimes get allocated
- Fragmentation endangers system stability over run time

Heap Areas

- Permanent Allocation
  - will never return the allocated block to the backing heap
  - will be managed by the associated bin (bin free list)
- Transient Allocation
  - will immediately return the allocated block to the backing heap
- Two distinct memory areas
- The permanent area will never fragment
- The transient area can fragment

Default Memory Pool Example

Let’s have a look on an example:

```
static const MemoryPoolConfig config = FEATSTD_MEMORYPOOL_CONFIG_INITIALIZER(
    config,                          // name of the variable receiving the configuration
    "FeatStdDefaultMemoryPool",      // the name of the MemoryPool
    MemoryPoolFlags::None,           // Flags that control the memory pool behavior
    128,                             // all bins up to 128 bytes are small bins
    cPoolBinSpecs,                   // the bin configurations
    0,                               // preset allocations
    DefaultBackingHeap,              // type name of the backing heap
    &OnOutOfMemoryDef,               // optional the out of memory function
    0                                // optional OnDestroy function
```

```
static const BinSpecification cPoolBinSpecs[] = {
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None,  8), // 732
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 12), // 594
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 16), // 340
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 20), // 143
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 24), // 270
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 28), // 18
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 32), // 156
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 36), // 93
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 40), // 7
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 44), // 128
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 48), // 107
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 52), // 114
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 64), // 30
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 72), // 114
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 76), // 8
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 80), // 9
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 88), // 5
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 92), // 6
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 96), // 8
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 100), // 4
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 108), // 14
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 112), // 5
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 116), // 16
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 120), // 34
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 124), // 54
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 128), // 81
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 136), // 153
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 144), // 145
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 168), // 15
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 180), // 4
...
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 16384), // 1
    // mandatory final entry
    FEATSTD_MEMORYPOOL_BIN_SPECIFICATION(MemoryPoolBinFlags::None, 0)
};
```

Memory Pool Example Outcome

- This configuration leads to fragmentation
- All flags (bin & pool) are set to *None* -> Allocations are always transient
  - Transient allocations can fragment
  - When all allocations are transient, fragmentation is certain
- Small Bin Limit is set to 128
  - All allocations up to 128 take the best fit bin size + 4 Byte overhead
  - All allocations > 128 take exactly the requested size + 8 Byte overhead
  - The configured bins > 128 are not considered at all, they just consume their header memory

Interpret MemoryPoolAnalysis.xlsx

When a permanent block with size 12800 Byte is used for 160 Byte allocation…

12800
C:\CGI-Studio_3.10\cgi_studio_candera\src\Candera\Engine2D\Core\RenderNode.cpp(66)

- The default behavior of Memory Pool for allocations is:

    1. Determine the bin according the requested buffer size
    2. Check the free list of the identified bin
    3. If no block is found in the free list, try to allocate a buffer from the backing heap.
    4. If the allocation from the backing heap fails, try to allocate from the next larger bins (overflow).

- In the example above, step 4 occurred -> Allocation of a new bin failed!

Proper Memory Pool Analysis

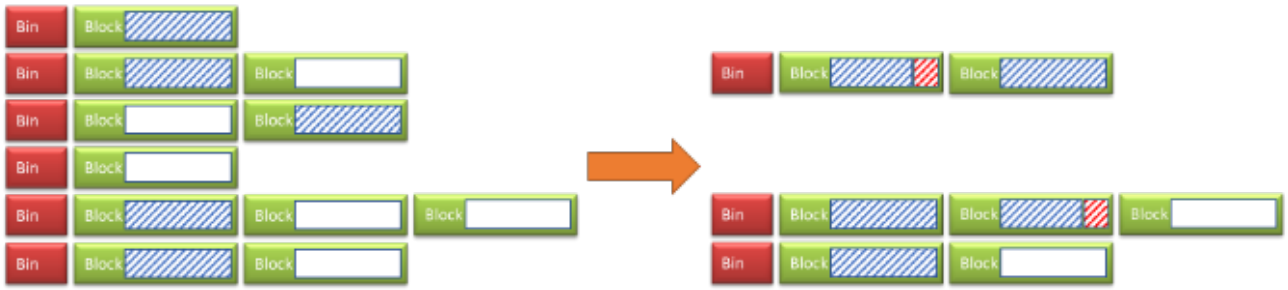
- A proper configuration for initial analysis…
- has a “small bin limit” of 4 Byte
    - All allocations shall take the requested size
- defines *PermanentOnly* flag on the pool
    - Bins shall be reused (freed blocks are returned to bin free list)
- defines additionally *NoOverflow* flag on the pool
    - inhibit overflow allocations
    - If specified, allocation requests from a bin will never overflow to a larger bin
    - Note: a test run with this flag might fail (Out of memory…) due to many unused blocks -> in this case remove the flag

Proper Memory Pool Configuration

- A proper final Memory Pool configuration…
- has a “small bin limit” in the upper area (where allocations are rarely)
    - Keep number of transient allocations to a minimum
- uses Flags on bins instead of the pool
    - defines *PermanentOnly* flag on small bins
    - Defines one bin with None flag, uses transient allocations for large blocks, which are rarely used (no reuse intended)
- uses few bins
    - balance wasted space vs. unused (rarely used) blocks

Wasted Space vs. Unused blocks

- Many permanent bins leads to many unused blocks
    - free blocks in bin free list
- Few permanent bins leads to memory “waste” in blocks
    - Allocation request smaller than block size
- The memory “waste” of few bins is smaller than free blocks from too many bins.
    - Free bin blocks are reused more often



Process of Configuration

- Take dumps with proper configuration for analysis
  - Play more than one use case to get better data
- If it is not possible to run with flag *NoOverflow* …
  - Remove the flag
  - Analysis is still valid, most overflows come from low size bins which are anyway “dense”
- Based on the results
  - Reduce the number of bins drastically
  - Choose a “small bin limit” where allocations are temporary requests Release of temporary memory vs. fragmentation

Reduce Number of Bins

Reduce the number of bins generously
Too many bins -> many free blocks not used
“Wasted Space” < Not used blocks!

```
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 8), // 732
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 12), // 594
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 16), // 340
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 20), // 143
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 24), // 270
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 28), // 18
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 32), // 156
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 36), // 93
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 40), // 7
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 44), // 128
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 48), // 107
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 52), // 114
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 64), // 30
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 68), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 72), // 114
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 76), // 8
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 80), // 9
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 84), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 88), // 5
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 92), // 6
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 96), // 8
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 100), // 4
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 108), // 14
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 112), // 5
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 116), // 16
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 120), // 34
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 124), // 54
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 128), // 81
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 132), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 136), // 153
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 140), // 2
```

```
..._SPECIFICATION(MemoryPoolBinFlags::<...>,  8),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 12),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 16),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 20),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 24),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 32),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 36),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 64),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 80),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 96),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 128),
..._SPECIFICATION(MemoryPoolBinFlags::<...>, 136),
```

Choose “Small Bin Limit”

- Remember: Avoid non used blocks!
- Summarize bins to small bins as far as possible
- “Small Bin Limit” shall start where allocations are temporary requests and fragmentation is more unlikely

```
...
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 224), // 4
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 256), // 6
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 272), // 9
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 320), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 328), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 336), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 344), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 352), // 9
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 360), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 368), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 380), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 416), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 440), // 3
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 444), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 460), // 4
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 476), // 2
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 504), // 2
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 512), // 7
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 520), // 2
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 532), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 572), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 600), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 616), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 704), // 6
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 800), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 1024), // 6
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 1248), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 1600), // 1
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 2048), // 1
```

```
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 272),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 352),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 476),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 532),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 704),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 1024),

..._BIN_SPECIFICATION(MemoryPoolBinFlags::<...>, 1600),
```

Result

```
static const BinSpecification cPoolBinSpecs[] = {
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly,  8),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 12),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 16),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 20),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 24),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 32),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 36),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 64),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 80),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 96),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 128),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 136),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 144),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 220),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 272),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 352),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 476),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 532),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 704),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::PermanentOnly, 1024),
..._BIN_SPECIFICATION(MemoryPoolBinFlags::None, 1600),


// mandatory final entry
..._BIN_SPECIFICATION(MemoryPoolBinFlags::None, 0)
};
```

```
// define the configuration for DefaultMemoryPool.
static const MemoryPoolConfig config =
FEATSTD_MEMORYPOOL_CONFIG_INITIALIZER(
    config,                         // name of the variable receiving the configuration
    "FeatStdDefaultMemoryPool",     // the name of the MemoryPool
    MemoryPoolFlags::None,          // Flags that control the memory pool behavior
    1024,                           // all bins up to 1024 bytes are small bins
    cPoolBinSpecs,                  // the bin configurations
    0,                              // preset allocations
    DefaultBackingHeap,             // type name of the backing heap
    &OnOutOfMemoryDef,              // optional the out of memory function
    0                               // optional OnDestroy function
```

Advanced Topics

- Pre-Allocation
  - Define how many permanent blocks shall be pre-allocated in each bin
- Combine with OverflowPreferred flag on bins

    - first try to overflow and then allocate from the backing heap
    - Limit overflows by adding flag OverflowBarrier to bin configuration
    - Such a configuration might improve the memory layout
- A sophisticated out of memory function could trigger the release of unneeded memory
    - Return value controls if the allocation request will be repeated or abandoned

“Unexpected” Allocations

VRAM Memory Pool

- [Applicable for platforms Traveo, Traveo II, RH850]
- Why in the VRAM 4 buffers are allocated for render buffers of a single Render Target?
- What happens:
  - During a transition first scene is unloaded, next scene is loaded
  - In case of startup, the first scene is the only one existing
  - Default Behavior of Courier:
    - when the last View on a Render Target is unloaded, the Render Target is also unloaded
  - The VramAllocator does never immediately free the memory, because the GPU might still write to it (Note: GPU runs
  - asynchronously to the CPU)
  - -> New 2 buffers are allocated when the second scene is loaded
- How to fix it?
- Send on startup a ViewPlaceholderReqMsg
  - increases the reference counter on Render target.
  - The message has two parameters
  - ViewId: this is a "reference" view which is used to determine the render target object
  - Load: true means to increase the reference count, false would decrease it (removes this placeholder from the internal list)
- Alternatively, you can also use a dummy scene with just a camera
  - This view would be always active and never unloaded.

Asset Cache

- From the Memory Pool Analysis Sheet:

16384
C:\CGI-Studio_3.10\cgi_studio_candera\src\CanderaAssetLoader\AssetLoaderBase\AssetCache.cpp(38)

- -> This is the cache used by the Asset Loader when copying images from flash to RAM
- Most or all images are directly rendered from flash
- The cache size is best practice for many platforms, but not when flash is directly accessible!
- -> Proposal: If flash is directly accessible, reduce the cache
  - E.g., to 1 kB (you win 15 kB!)
- CMake flag CANDERA_ASSET_CACHE_SIZE