MeVisLab Release Notes

Table of Contents

Release Notes

This document lists the most relevant changes, additions, and fixes provided by the latest MeVisLab releases.

Version 3.7.2 Stable Release (2023-09)

New Features

- Various small updates for third-party libraries. A notable change is that we now use OpenSSL 3.0 instead of the 1.1.1 series.

Fixes

- Fixed: `DicomImport` accidentally did set interface field values from a non-GUI thread, which isn't allowed in MeVisLab.
- Fixed: `VoxelizeMarkers` crashed if there were no markers to voxelize.
- Fixed: `WEMLoad` crashed if loading a VRML file that was written by `SoSceneWriter` in VRML97 format.
- Fixed a potential crash in `SoCSOSplineEditor`.
- Fixed some memory leaks in `WEMLoad`.
- Improved layout of the letters in the VectorEdit control.
- CMake: Make it possible to use `find_package(MeVisLab)` in parallel sub-directories, without a `find_package(MeVisLab)` in a common parent directory. (The CMAKE_MODULE_PATH variable was previously only extended by the first `find_package(MeVisLab)` call, but this only applies to the current directory and its sub-directories.)
- ADK/ApplicationBuilder:
  - Python site-packages in MeVisLab user packages were not signed for add-on installers.
  - The necessary OpenCV libraries were not added automatically to installer when using the `VideoWriterML` or `VideoWriterInventor` modules.
  - Windows: Fixed calling "Dependencies" tool in the DynamicDependencyAnalyzer.
  - Linux: The OUTPUT_PYTHON_EXE variable wasn't set correctly.
- Small documentation fixes.

Version 3.7.1 Stable Release (2023-07)

New Features

- The `TestWebEngineView` example module now contains a demonstration for how to use the WebEngineView control to show PDF documents.

- Various small updates for third-party libraries.

Fixes

- Fixed: `SoView2DLabel` and `SoView2DRectangle` could display rectangles with a missing corner pixel on certain OpenGL implementations. The rectangles are drawn differently now.
- Fixed: `SoCSOSplineEditor` didn't switch back to edit mode when deleting all currently set seed points.
- Fixed a memory leak in the `ImageResample` module.
- Fixed the example networks for `VideoWriterML` and `VideoWriterInventor`, which were broken after the last update to `RunPythonScript`.
- Fixed: Pylint error checking in MATE could not resolve PythonQt and Inventor imports.
- The MLAB_AUTOMATIC_POSTBUILD_COPY step in CMake was optimized to only update library files if they really did change.
- Fixed: [ADK/ApplicationBuilder] Python wildcard imports, followed by a comment and another wildcard import, would cause an error message when building installers (e.g. for scipy).
- Various small documentation updates.

Notes

- It was discovered that the persistence format for (Vessel)Graph objects — if you switched it to binary by using the `useBinaryPersistence` method on the object wrapper — has changed between MeVisLab versions 3.4.3 and 3.5. Binary files saved by MeVisLab 3.5 and later can't be read with MeVisLab 3.4.3 and earlier.

  This is because the code uses the serialization function of the Boost library, which isn't backwards compatible to older Boost versions. If you need compatibility of persistence files in both directions, you should stick to the non-binary format.

Version 3.7 (2023-05)

Fixes and Enhancements (MeVisLab)

IDE

- The MeVisLab executable got a `-noninteractive` command-line option to print certain error messages to the console instead of opening a dialog.
- Do not reset the author name when changing the target package in a module wizard.
- The profiling view in the IDE marks modules that have been removed in the meantime with a strike-out font now.
- Switched MATE and the TestCenter to use the new `WebEngineView` (instead of the deprecated WebKit-based `WebView`).
- Added high-resolution cursors for the viewers.

- Improve context help function of the scripting console.
- Fixed a few places where modules or MeVisLab itself didn't work with Unicode characters in file names.
- Added a workaround for the Quick Module Search drop-down list not always showing on Linux.
- Fixed a crash in the Inventor Viewer control when referencing a SoNode field without an Inventor scene.
- MATE editor: When adding or removing user packages in MeVisLab, MATE immediately reflects this change without restarting now.
- Fixed an issue that field updates from modules running in a separate process could be received during a long-running calculation. No update notifications were emitted in this case and other problems could be caused. The updates are now properly delayed until after the calculation.
- [Windows only] MeVisLab tries to make sure that module panels are wide enough so that there is a space in the title bar with which they can be dragged around.
- The Project Wizard got an option to create backups of files that would be overwritten.
- The **Convert Pro Files to CMake...** conversion script got a bunch of small improvements.

TestCenter

- The TestCenter got an option to always use the current MeVisLab in secure mode (instead of the configured MeVisLab, which may be any installed - or removed - version).
- Call `(setUp|tearDown)(Class|Module)` when executing Python unittests in the TestCenter. (This uses some protected methods of `unittest.TestSuite`.)
- Added new test functions `hasNoWarning`, `hasWarning`, `hasError`, `hasWarningAndError`, and `hasInfo` to `TestSupport.Base`. They act as context managers that can be used with the "with" statement and are easier to use than the old expectXyz functions which needed to be given a function and arguments to execute.
- Do not crash on encoding error in secure mode.
- Prevent crash when closing the TestCenter while tests are still running.
- Fixed: The **Run All Tests** function on a module would not use the configuration configured for the TestCenter itself.

Scripting

- `PythonPip`
  - Allows to install Python packages to a user's MeVisLab package now (instead of just to the Python directory in MeVis/ThirdParty). A site-packages directory is added to the selected MeVisLab package and added to the Python path in MeVisLab.
  - When installing or removing Python packages with PythonPip, ThirdPartyInformation entries and library `.mli` files (the latter are used for correctly composing application installers) are automatically updated.

- Added an `MLAB.setShouldStop()` method, which is mainly useful for testing.
- Added an optional `pixelScaleFactor` argument to the `updateLayout` method of `SoView2DWrapper`, and removed the `is3D` argument, which wasn't useful.
- Added the column number as argument to the `doubleClickedCommand` of the `ListView` control.
- Fixed some flashing when using the `setContentString` method of the `DynamicFrame` control.
- Worked around an import error with PyTorch.
- Fixed: The `.pth` files from installed Python packages were not evaluated, which could lead to some imports not working as documented, e.g. in the case of PyWin32.

C++

- Removed the interim (build number) member from `ml::Version`. It interfered with our compiler cache.
- Added the static method `View2DFont::createFont`. Made the implementations of `View2DFont` private.
- The `DCMTree::Dict::info` method automatically tries to prefix the tag name with "RETIRED_" if a tag isn't found without it.
- The method `CSOGenerateSeedPoints::redistributeSeedPointsEveryNthPathPoint` got a flag whether to send or not send update notifications.
- Let CMake also install `.pdb/.debug` files if the CMake variable `MLAB_INSTALL_PDBS` is set.
- The CMake function `mlab_install` now supports the `OPTIONAL` argument, so that a missing library doesn't fail the install step.
- Fixed that that `mlMatrix4.h` include would not work without another include.
- The OpenCV library can now be accessed component-wise in CMake.
- `_MLsnprintf` and `_MLsscanf` are deprecated now, please use `snprintf` and `sscanf` instead.

Application Builder

(for those users with an ADK or Application Builder license)

- Applications will now use the application title in certain dialogs which previously always used "MeVisLab".
- Improved the error message when using an invalid license while generating installers.
- The Python import analysis has been refactored to cope with more complex cases of import statements.
- `MeVisLabAppConsole` didn't consider itself to be an application run-time, so it didn't accept valid license files.
- It is now possible to run an application with the developer's license (if the license is bound to a computer).

Windows specific

- DLL loading in MeVisLab doesn't look at the `PATH` environment variable anymore, but instead uses the `AddDllDirectory` system call, which doesn't have a length limitation and should prevent libraries being loaded from "foreign" directories.

  Directories outside the installed or registered packages can be added separately by specifying the variable `ExtraDllLoadPaths` in the MeVisLab preferences file. Hint: You can use environment variables in the value(s).

- The tool check window of the ToolRunner now indicates that either DependencyWalker or Dependencies is sufficient to fulfill the requirements for a DLL dependency checker.

- AddOn installers did not correctly detect a MeVisLab SDK installed in the user path.

Linux specific

- Starting MeVisLab through the starter script will correctly forward SIGTERM and other signals.

- Fixed application wrapper script to work with space characters in the installation path.

- Enabled GStreamer support in OpenCV. This fixes the `VideoWriterML` and `VideoWriterInventor` modules on Linux.

Other

- Added missing `StudyInstanceUID` tag to some DICOM files in the demo data directory.

Third-Party Libraries

A lot of third-party libraries have been updated in this release.

See the list of available third-party libraries here.

Note that this is not the complete list of third-party components used in MeVisLab, for this you should consult the About dialog of MeVisLab, or use the `ThirdPartyInformation` module.

Some notable updates:

- Qt has been updated to version 5.15 (from 5.12).

- The PythonQt scripting wrappers for Qt have been re-created to fix an issue that virtual method QGraphicsItem::itemChange() could not be overridden for all derived classes.

  (Actually this fix has already been in the 3.6.1 release.)

- The Python packages Matplotlib and Sphinx have finally been updated. Matplotlib is still an older version though.

Modules

- `DicomImport`:

  - Optimized reading of image data from large multi-frame files.

  - Correctly handle reversed slice order for RTDose images.
  - Improved error and log handling. The error log model now contains the error message and the name of the tag causing the error (if applicable).
  - Correctly handle 2D+T flag on multi-frame files.
  - Do not create frame-specific tags for single images
  - Fixed incorrect interpretation of MONOCHROME1/2
  - Improve handling of incorrectly specified character encodings and of encoding errors.
  - Ignore DICOMDIR files on import.
  - Handle leading and trailing spaces in DS/IS values.
  - Fixed a crash at program exit after having accessed imported images on Linux.
  - Fixed a possible conflict between imported private DICOM tags and the private tags used internally to hold per-frame tags.
- `CurveCreator`: Fixed a bug that the title of a single curve would not be set if the title separator was not set.
- Added a new macro module `MaskSliceInterpolator` which can interpolate sparse mask slices. It is based on an algorithm that interpolates 2D distance values (inside values being negative) between defined slices.

  Also added an alternative C++ module `SliceDistanceTransform`, which basically does the same.
- `SignedEuclideanDistanceTransform`: Has been re-implemented to be faster.
- `Threshold`: Fixed handling of fractional threshold values on integer images.
- `OrthoSwapFlip`: Avoid to create too many output change notifications.
- `ImageComposer`: Fixed that an invalid time was used when adding images in the t dimension.
- Improved error handling of `LoadBase` and `LocalBase`
- `SetDicomTreeOnImage`: Added a `requireValidDicomTree` option flag, which invalidates the image output if no DicomTree is provided.
- `DrawVoxels3D`:
  - Added support for copying the second input image to a given location
  - Added a `voxelWriteMode` with modes "WriteAlways" and "WriteIfNotZero".
  - Fixed that the `clampedVoxelWriteValue` wasn't always updated anymore.
- The auto-completion in the interactive console of `RunPythonScript` has been fixed.
- `WEMLoad`: Added support for binary `.ply` files.
- `WEMReducePolygons`: Behaves deterministically now.

- `SoWEMRendererEdges`: Implemented support for an per-edge PVL. Added new `overrideSelectedPVL` that can override the PVL that is passed from the renderer. Removed unused `selectedPVL` and `availablePVLS`.
- `SoSceneLoader`: The Collada import has been disabled because of an unfixed security issue.
- `SoView2D`: got an option field `wrapAroundTimePointScrolling` to wrap around when stepping through the time points with the cursor keys.
- `SoView2DLabel`: got a `borderColor` field (default is white). Previously the border color was undetermined and might have been the same color as the text.
- `So3DMarkerEditor`: Append unit (mm) if TEXT_LENGTH is selected as annotation, same as in the `SoView2DMarkerEditor`.
- `SoText2` and `SoText3` would crash when trying to render Unicode characters > 255.
- The `SoShaderParameterPlane` module got a flag `transformToEyeCoordinates`.

Modules and interfaces provided by Fraunhofer MEVIS

Additions

- New module `VoxelizeFibers`.
- New module `FilterMarkersByMask`.
- Added clearance list support to `ThirdPartyLicenseReport`.
- `BoundingBoxListener`: Added configurable epsilon.
- Improved logging support for `CachedPath` module and `cached_path` Python library.
- Added `FieldValues()` context manager to `fmeTestSupport.MlabContext` for temporarily setting field values.
- Made it possible to log warnings on windows, by using `include(FME_HANDLE_ALL_QA_RELEVANT_WARNINGS_AS_ERRORS)`, `include(FME_HANDLE_QA_RELEVANT_WARNINGS_AS_ERRORS)`, or `include(FME_WARN_TO_ERROR_PREPARE_MODE)` in `CMakeLists.txt`.

  Revision of warning and error handling is still in progress.

  Supported only when depending on FME packages.

Functional changes/Bugfixes

- Fixed memory leaks in
  - `CurveProperties`
  - `SoNodeFilter`
  - `SoCSOManualCorrectionEditor`
  - `SoOverrideCursorShape`

Version 3.6.1 Stable Release (2022-12)

New Features

- A lot of third-party libraries have been updated to the latest version for this release.

Fixes

- Fixed: The installer wouldn't install the Visual Studio runtime when required (and possible) on Windows.
- Fixed: MeVisLab linked against the system ICU library on Windows, which wasn't always available. Now it links against the version provided with the SDK, as intended. This only affected the Release mode.
- Fixed: Lots of instances where a Debug mode DLL loaded a Release mode DLL though a Debug mode version of the same DLL was available (and in one case the other way round).
- Fixed: Using the "Find" shortcut in a `WebView` control and typing an existing character would crash MeVisLab in Debug mode. (This actually was an `assert` in the code).
- Fixed: The Linux installer contained copies of files that should have been symlinks. The installer got smaller in size as a result of the fix.
- Fixed: On Linux one couldn't import the `vtk` Python wrappers in Debug mode.
- The tool check dialog of the ToolRunner still listed the old required version for NSIS on Windows. Now it shows the correct required version 3.0.
- Fixed: The **Level planar CSOs** option of the `ExtractRTStruct` module didn't work.
- Fixed: The new `XMarkerListToCSOList` module interpreted the `pointsPerMM` field as "distance between points" instead.

  If you used this module with a value other than 1 you should check your usage.
- The `CLIImporter` module now works without needing to install an additional Python package.

Version 3.6 (2022-10)

Fixes and Enhancements (MeVisLab)

IDE

- The "Python 3 Support" user script has been removed, since MeVisLab uses Python 3 since version 3.1, and everybody should have migrated their scripts by now.
- The module wizards should now work with Unicode characters in the path or values.
- Removed the **Debug Network** button from the TestCaseManager, which wasn't functional anymore. If one wants to see the test network after execution one can add a `EXPECT_TRUE(False)` at the end of the test and activate the **Stop and inspect network on error** option (previously just called **Stop on error**).

- The option **Open C++ Project in IDE** has been removed from the context menu of modules. This didn't really work with the new CMake-based build system, where the location of the build directory isn't as fixed as it was with the old qmake-based system.

- MATE:

  - Fixed a crash in the Python debugger when using an expression in the Evaluate Expression view that was also set in the Watches view.

Scripting

- Fixed a bug with reference counting of Python object arguments of a queued connection between user-defined signals and slots.

  This was done by using type PythonQtSafeObjectPtr instead of PyObject* for the Python arguments. This might affect connections between signal/slots defined in Python code and signal/slots defined in C++ – the C++ signature might need to be adapted in these cases.

- Added the MeVisLabJobScheduler framework to the SDK. This is a framework for convenient execution of batch jobs from MeVisLab.

  Please see MLABJobSchedulerClient for how to use this.

- Modules and controls got a new `droppedFilesCommand` (notice the plural!) alongside the existing `droppedFileCommand`, where you get the dropped files all at once if more than one file was dragged and dropped onto a module/control.

- `TestSupport.ScreenShot.createOffscreenScreenShot` got an extra optional parameter `backgroundColor`.

- Importing of Python modules from a Projects sub-directory (via `mlab_projects`) would fail if the file path to the package itself would also contain a `Projects` directory.

- Added a range check to those methods of `MLABDicomTag` that take an index parameter.

C++

- The type multiplexing of the calculateOutputSubImage method of classic ML modules without image handler can now be done through the C++17 function `std::visit`. The support for this can be found in the include file `mlTSubImageVariant.h` of the ML project. Code examples and ML project wizard have been adapted accordingly.

Application Builder

(for those users with an ADK or Application Builder license)

- Linux Users: The default for the finish page of the installers has changed. The option to run the application after the installer has finished must be enabled explicitly with the variables `$INSTALLER_FINISHPAGE_RUN` and `$INSTALLER_FINISHPAGE_RUN_TEXT`.

- The default encoding of `.mlinstall` and `.mli` files is now UTF-8 on Windows, the same as it was on Linux and macOS already. Also fixed some bugs with Unicode characters in paths.

- The Windows installer generation can alternatively use `Dependencies` instead of `DependencyWalker` for DLL dependency analysis.

- Missing or failing dependency analysis tools will now also lead to a non-zero exit code of the whole installer generation step. (Previously you needed to scan the log for errors.)
- The `--content-list-file`/`--verbose` option of buildInstaller.py didn't collect all files contained in the generated installer in all circumstances (especially on Windows), mostly just the directories.
- Windows: The minimum required version for NSIS is 3.0 now.

Other

- The Mesa OpenGL software driver included with the Windows version of MeVisLab is not a single DLL anymore. If you copied this DLL for your purposes, you now need to copy the whole content of the directory `Package/MeVisLab/IDE/bin/Mesa`.
- The option to immediately run MeVisLab from the SDK installer has been dropped, since this caused some problems.
- The SDK can now be installed without administrator rights (in the users directory) on Windows.
- Documentation:
  - The Scripting Reference in 3.5.0 contained a lot of inaccessible members, this has been fixed.
  - The module help files now contain a reference to appropriate scriptable classes (if available) at input or output fields of type MLBase.
  - Added a new installation guide, which also covers usage in a Docker container.

Third-Party Libraries

A lot of third-party libraries have been updated in this release.

See the complete list of available third-party libraries here.

Note that this is not the complete list of third-party components used in MeVisLab, for this you should consult the About dialog of MeVisLab, or use the `ThirdPartyInformation` module.

Modules

- DicomImport:
  - `DicomImport` and `DicomImportExtraOutput`: Breaking Change: The field `selectedItem` uses now an internal unique Id to select the current volume and not an index anymore. The set of Ids of the recent partition operation can be retrieved either from the `id` attribute in the `outputModel` or via the scripting interface of `outImportResult`.
  - Breaking change: The item tree shown in the panel of the `DicomImport` module now has one level less (the series level), instead the volumes will be displayed with the modality and series description.

    This was necessary because different DICOM series might now be combined to a 4D volume. If the constituting DICOM series had different series descriptions, no series description will be displayed for that volume.
  - One can also provide a list of files instead of just a directory as input for the import process. Files can also be dragged onto the panel.
  - Moved the import process completely into the background. The UI will only block briefly when selecting a volume for the first time now.

  - Enhanced multi-frame files will now be split into the component frames internally and re-composed during the import step to satisfy the given sorting and partitioning rules.
  - The **Stop** button turns into a **Clear** button if the import has already finished.
  - Just requesting the file paths of a volume from the `DicomImportResult` is faster now.
  - Fixed an issue with importing multiple simple multi-frame files. (Some multi-frame files could be skipped.)
  - Added an import option to always split time points.
  - `DicomImport` and `DicomImportExtraOutput` got an extra output `outDicomTree` which will provide the Dicom tree of files that have no image data (marked with **<no image data>** in the list).
  - The import step will also consider the `GridFrameOffsetVector` tab of multi-frame images to determine the correct voxel size in the z dimension.
- Added the modules `ExtractRTStruct` and `ComposeRTStruct` module to convert a RTSTRUCT DicomTree object into a CSOList object and vice versa.
- WEM modules:
  - `WEMClipPlaneToWEM`: The triangulation algorithm for closing a clipped WEM was replaced with a conforming Delaunay triangulation. The resulting WEMs should now always be well-formed.
  - `WEMModify` can center the whole WEM now, not just the individual WEM patches.
  - Fixed: The combination of `WEMGeodesicClip` with `WEMDemergePatches` could cause a crash.
  - Fixed: Getting the nearest point on a WEM surface with the `MLWEMNearestPointWrapper` scripting wrapper could in fact return a point that was not the nearest for WEMs with very different triangle sizes.
  - `WEMDemergePatches` will print a warning now when saving a WEM to STL format when this might break the mesh structure (because the input WEM has multiple nodes at the same position).
  - `SoWEMConvertInventor`: Got an option `useNodeColors` to use the diffuse color of a shape for the WEM nodes.
- `OrthoProjection`: Is now more precise in calculation of the possible value range in `minPos`/`maxPos` mode.
- When expanding an image in the t dimension with `SubImage` the last time point value is replicated now instead of using a default invalid date time value.
- `SoVolumeCutting`: Fixed a crash when interacting with the module while no reference image was connected.
- Fixed: Using a `SoInteractionMapping` below a `SoSeparator` would not work correctly.
- The inspector modules used in the **Output Inspector** have been moved to their own group, which makes them invisible in the search by default.
- Added the modules `CSOListToXMarkerList` and `XMarkerListToCSOList`.
- Improved marker dragging behavior in the `So3DMarkerEditor`.

- `DrawVoxels3D`: Improved behavior for min/max image values and undo when the update mode is **Copy Input Image**.
- Added the module `MemoryCacheBuffer` which allows to store a list of images (accessible by setting an index).
- Fixed an issue with the `OffscreenRenderer` and repeatedly applied lights that could occur with some OpenInventor scenes on subsequent renderings.
- Fixed a possible crash in `SoDiagram2D` when input curves and input style palette were changed at the same time.
- Fixed wrong rounding in `DicomRescale` when the input voxel type was a float type and the output an integer type (or vice versa).
- Fixed the update behavior of `ApplyLUT` for images that required internal rescaling (e.g. voxel type int32 or float).
- Fixed a cause for rare crashes in the constructor of the `OrthoReformat3` module.
- Worked around an issue of the Qt based font rendering (enabled by setting the value `View2DEnableQtFontRendering = yes` in the preferences file) being slightly cut off in `SoView2D` and `SoDiagram2D` when used on Windows with a display scaling factor of 150% or greater.

Modules and interfaces provided by Fraunhofer MEVIS

Additions

- Added the `dicomweb.DicomWebAdapter` python class to wrap `dicomweb-client` for convenient DICOM series/study retrieval and storage through DicomWeb/WADO (PACS) APIs.

  The dicomweb-client Python package currently needs to be installed separately through `PythonPip` to use this class.
- New module `AutoSwitchBase`, for automatically selecting the non-null instance of two base inputs.
- New module `LinePlaneIntersection`: Calculate the intersection of a plane with a line.
- New module `PointPlaneDistance`: Calculate the distance of a point to a plane.
- New modules `ComposeImageVector`, `DecomposeImageVector`, `ComposeVector6`, `ComposeVector6FromVector3AndComponents`, and `DecomposeVector6`: Complement the existing Compose/Decompose modules, for use with 6D (image) vectors (e.g. with XMarker modules).
- New module `CurvesFromXMarkerList`: Create curves by sampling XMarker properties (such as position and vector components) along a given dimension (e.g. list order or time)
- New module `MultiFileVolumeListPROutput`: Experimental module to display basic graphical annotations (Presentation States) on DICOM frames.

Functional changes/Bugfixes

- The C++ `ParameterInfo` class no longer derives from `QVariantMap`, but instead contains one. The map can be accessed via the data member now.
- `(BaseWith)ParameterInfoWrappers` now catch exceptions and show the information in an extra 'error' key

- Module `XMarkerListFilter`: Fixed a bug when filtering XMarkers without a name.
- Module `ReadTextFile`: The content of the loaded file is not stored in network files anymore.

Version 3.5 (2022-06)

Note

We have suspended the macOS support of MeVisLab in this release. We are working to bring it back in one of the next releases.

Deprecation Strategy

MeVisLab has grown over the last twenty and more years. The users and we, the manufacturer, have learned over this time that several approaches were replaced by better solutions. But many of the older possibilities were kept in the code to remain backward compatible. This old structures hinder or slow us down to come up with better and more productive ways. We realized this already in the past and marked modules and interfaces as deprecated. During the development of version 3.5 we identified more old and obsolete modules and interfaces and marked them as deprecated as well.

We decided that we will follow a clear deprecation strategy starting from now on. That means that all MeVisLab modules, C++ and Python interfaces, and source code macros that were marked as deprecated are kept throughout the version 3.x for backward compatibility. But they will be removed from MeVisLab with the next major release.

Fixes and Enhancements (MeVisLab)

IDE

- Added a user script to help convert `.pro` files to `CMakeLists.txt`.
- The support for `.pro` files has been removed from the ToolRunner, as we switched to CMake files. We believe that the support for CMake files integrated into most IDE's is sufficient for everyday use.
- If manual saving of internal networks is disabled through `Package.def` also disable auto-save for internal networks.
- Clicking on the background of a network deselects all modules.
- The console log file is always written UTF-8 encoded.
- One could force a reload of a module from MATE while a modal dialog is open, which would crash. This is now prohibited.
- TestCaseManager: Python coverage can now be enabled globally. Test coverage per test case is discarded then, to keep the test report small in size.
- Removed the limit on parallel threads from the Preferences dialog.
- MATE:
  - Python checker: You can switch between mypy and pylint with a keyboard shortcut now (default is **Ctrl**+**F4**).

- The default version of pylint that is installed by default is 2.11.1 now. The default version of rope was updated to 0.20.1.
- When using **Find and Replace in Files** with regular expressions, replace `\1` with captured text, like in **Find and Replace**.
- Allow to skip/abort replacing in files if a file seems to have changed since the search.
- Avoid crash if **F1** is pressed for am MDL tag under the cursor with no validator entry.
- Avoid crash in help editor if the usual first section ("Purpose") was removed manually from the help file.

Scripting

- Make Python wrappers for `QMutexLocker`, `QReadLocker`, and `QWriteLocker` (in `PythonQt.QtCore`) work as context managers.
- `MLABPopupMenuControl::clearItems()` did not destroy sub menus.
- `ctx.showWindow()` in standalone applications could open windows on the wrong screen.
- Fixed a crash in pylint caused by PythonQt when trying to sub-class, e.g., `SoDragger`.
- Change default behavior of `MLABFileDialog.getSaveFileName`. The new default behavior is to ask to confirm overwrites.
- Added `TestSupport.Base.ignoreWarning/ignoreError` methods that don't complain if no warning/error is printed.
- One can add the variable `_mlab_do_not_reload` to a Python module to exempt it from reloading (triggered with **Extras** › **Reload Imported Python Modules**).
- Replaced the scripting methods `MLABModule::killTimer` and `MLABModule::startTimer` and printed a message in the replacement code that the user probably wanted to use another method.
- MeVisLab remembers the last directory one opened files from in a file dialog. These stored directories can now be accessed with `MLABFileDialog.lastFileDialogPath` and `MLABFileDialog.setLastFileDialogPath`.
- Make auto-revert of `TestSupport.ChangeSet` reliable. (Previously, it was only executed when the object was garbage collected.)
- Connecting a callable to the `valueChanged` signal of an `MLABField` wrapping an ML/OpenInventor field would not be triggered if there was no other reason to create a field sensor.
- Use a dedicated enum `CSOVoxelWriteMode` for the `voxelWriteMode` attribute of `MLCSOListWrapper` and `MLCSOWrapper`.
- Added an optional parameter to `TestSupport.Base.expectErrors()` to prevent changing the message severity, so that application code that evaluates `MLAB.priv().hadLogErrors()` can be tested.
- Set a default exception handler when calling `async` methods from MeVisLab, so that exceptions are visible.

- The class `DCMTree::StructuredMF` can also look into `FunctionGroupSequence` tags for frame-specific tags now, and this ability is employed by the `MLABImageField` methods `getFrameSpecificDicomTag` and `getSharedDicomTag` now.
- Added new helper Python class `TestSupport.NotificationCounter.NotificationCounter`.
- Added new helper Python method `TestSupport.DicomTreeCompare.compareDicomTrees`.
- A module test can be disabled completely by adding `testGroups = disabled` to the test definition.
- Some Qt and Inventor classes had no proper != operator in the Python bindings, so that Python fell back to comparing object (pointer) identity. (The == operator on the other hand was used correctly.)
- Not every numpy base scalar type that could be set (as an array) on a `PythonImage` module could be mapped to the corresponding ML datatype.

Controls

- `EventFilter`: Handle the case when the eventFilter method is being called recursively.
- The `Splitter` control got a `childrenCollapsible` attribute.

C++

- The used C++ standard has been raised to C++-17 (from previously C++-11).
- The build system was changed from qmake (`*.pri`, `*.pro`) to CMake. See our CMake documentation. In MeVisLab there is a user script to help converting `.pro` files to `CMakeLists.txt` files.
- More strict compiler warnings are enabled. Many warnings are treated as errors now, for details see `./cmake/module/MeVisLabCompilerSettings.cmake`.
- Processes started with `MLABUtils::launchApplication` will not inherit file handles anymore.
- Refactored the `MLABLogging` mechanism to be thread-safe. On the downside, messages from the IDE can't be captured through the ML logging mechanism anymore.
- Fixed a memory leak in the handling of ML images (when they are split into more than 512K pages according to page size and image size).
- The `ml::DateTime` class in `MLUtilities` will behave slightly different when confronted with invalid dates/times now. (Previously, it silently corrected some invalid date/times.)
- Added the `ScopeGuard` helper class to call code when a scope is left. See `MeVis/Foundation/Sources/MLUtilities/mlScopeGuard.h`.
- `CSOCreationScope` and other scopes now print an error when the end of the scope is reached but the necessary `setId` hasn't been called.
- `MLUtilities` got a header file `mlUnicodeString.h` with conversion functions between UTF-8 encoded std::string and UTF-16 encoded std::wstring.
- Added a - seemingly useless - std::string variant of `StringConversion::writeToStdString` which helps with the use of writeToStdString in template code.
- The `ml::TypedEnumField` can also be used with enum classes now.
- PythonQt and the Qt and Inventor Python bindings have been moved to `MeVis/ThirdParty`.

- Starting to use a more efficient Dicom tree data structure, DCMTree2. So far, it is only used internally. It is planned to replace the DCMTree with the new one in the next major release.
- The Compiler definition `MEVIS_64BIT` was removed. There hasn't been a 32bit version of MeVisLab since some time.
- The include directories of some projects/third-party libraries have changed. Accordingly some includes statements might need to be adapted.
- *Deprecations:*

  We put special care on marking deprecated code parts and pre-processor macros, and enclosed those with the `#if ML_DEPRECATED_SINCE(3, 5, 0) ... #endif` macro function. Per default this code is enabled and does not cause any warnings.

  We recommend the following migration strategy to avoid using of the deprecated code:

  - Define in the CMakeLists.txt file of the project that shall be adapted the pre-processor symbol `ML_DEPRECATED_WARNINGS`, e.g. with the statement `target_compile_definitions(<target> PRIVATE ML_DEPRECATED_WARNINGS)`.
  - Now all code parts trigger a compiler warning that use the old code. The default compiler setting is that warnings are treated as errors. So these warnings have to be fixed before a clean build is possible. Follow here our detailed guidelines either in the corresponding header files or the further description below.
  - Since it is not possible to trigger a deprecation warning on all platforms for all different kind of deprecated code, it is possible to hide the deprecated code by adding a pre-processor symbol like `ML_DISABLE_DEPRECATED_BEFORE=400000` where the number here would correspond to version 4.0.0. So all deprecated code before that version would be hidden. So this could be achived by adding this statement `target_compile_definitions(<target> PRIVATE ML_DISABLE_DEPRECATED_BEFORE=400000)` to the project specific CMakeLists.txt. When there is a successful build, then the migration was successful.

  Some of the things that are newly marked deprecated:

  - The file `./MeVis/Foundation/Sources/ML/mlBasics.h` is marked as deprecated.

    The file `./MeVis/Foundation/Sources/MLUtilities/mlUtils.h` is marked as deprecated.

    The file `./MeVis/Foundation/Sources/MLUtilities/mlSystemIncludes.h` is marked as deprecated.
  - In file `./MeVis/Foundation/Sources/MLUtilities/mlErrorMacros.h` the following macros are marked as deprecated:
    - `ML_CHECK_NEW` should be replaced with `new`.
    - `ML_CHECK_NEW_TH` should be replaced with `new`.
    - `ML_DELETE` should be replaced with `delete`.
    - `ML_DELETE_ARRAY` should be replaced with `delete []`.
  - In file `./MeVis/Foundation/Sources/MLUtilities/mlTypeDefs.h` the following macros are marked as deprecated:
    - `ML_MIN` should be replaced with `std::min()`.

  - `ML_MAX` should be replaced with `std::max()`.
  - `ML_ABS` should be replaced with `std::abs()`.
  - `ML_CLAMP` should be replaced with `std::clamp()`.
  - `ML_IS_64_BIT_SYSTEM`: All places which depend on this definition should be cleared.
- In file `./MeVisLab/IDE/Sources/Libraries/MLABTools/Tools/mlabBasics.h` the following macros are marked as deprecated:
  - `MLABScopedPointer` should be replaced with `std::unique_ptr<T>`.
  - `MLABScopedArray` should be replaced with `std::unique_ptr<T[]>`.
- In file `./MeVisLab/IDE/Sources/Libraries/MLABTools/Tools/mlabMacros.h` the following macros are marked as deprecated:
  - `MLAB_DELETE` should be replaced with `delete`.
  - `MLAB_DISALLOW_COPY_AND_ASSIGN` should be replaced by using C++11 `delete` on copy constructor and assignment operator.
- In file `./MeVisLab/IDE/Sources/Libraries/MLABTools/Tools/mlabLocations.h` the following functions are marked as deprecated:
  - `MLABLocations::executableDirPath`, use `QCoreApplication::applicationDirPath` instead.
  - `MLABLocations::homeDirPath`, use `QStandardPaths::writableLocation(QStandardPaths::HomeLocation)` instead.
  - `MLABLocations::tempDirPath`, use `QStandardPaths::writableLocation(QStandardPaths::TempLocation)` instead.
  - `MLABLocations::userDocumentsPath`, use `QStandardPaths::writableLocation(QStandardPaths::DocumentsLocation)` instead.
- The methods of `ml::TreeNode` that acted on char* values are deprecated (because they are prone for memory leaks); variants of these methods acting on std::string were introduced that should be used instead.
- In the same way, some methods of `ml::Base` using char* values have been deprecated and replacement methods using std::string have been introduced.

Application Builder

(for those users with an ADK or Application Builder license)

- Installer language extensions:
  - Added `IFSET` and `IFNSET` statements, which ignore the value of a variable (unlike `IFDEF/IFNDEF`, which treat 0 or an empty value as undefined).
  - Added `EXISTS/ISDIR/ISFILE` operators, which can be used in `IF` statements.
  - Added the `INCLUDE_IF_EXISTING` command, which does not complain if the given file does not exist.

  - Added the command `INCLUDE_DEPENDENCIES_OF`. This command follows the dependencies declared by `.mldepends` files and includes all `.mli` files in any `<package>/Configuration/Installer/Libraries` directory that have the name of one of these dependencies.

    `.mldepends` files are also created for executables now.

  - When the `EXECUTE` command in an `.mlinstall` file fails with an error code the whole installer generation fails with an error code now. If you want to ignore error codes of a command, use the new `EXECUTE_NO_FAIL`.

- Installer language variables:

  - Make Windows installer execution level configurable: Set variable `INSTALLER_NSIS_EXECUTION_LEVEL` to Standard, Highest, or Admin.

  - Set file version (and other) attributes of a Windows installer if the variable `INSTALLER_FILE_VERSION` is set in the installer definition.

  - Allow to set no license text (by setting an empty `INPUT_LICENSE_TEXT`) for a stand-alone application.

    The license text can also be configured in the installer wizard.

  - The MeVisLabApp executable can be renamed in a standalone installer by setting the variable `STANDALONE_EXECUTABLENAME`.

- A third-party report is automatically generated when an installer is built.

- ModuleDependencyAnalyzer: A Python import from `<package>/Modules/Scripts/python` should not add the entire `Modules/Scripts/python` directory to the installer anymore.

- Overhauled dependency analysis, so that less third-party dependencies should be added unnecessarily.

  Please let us know if something broke for you since a previous version.

- Standalone application installers now include the OpenSSL libraries by default. (This can be suppressed by adding `$WITH_OPENSSL 0` to your `.mlinstall` file.)

- Use the application name in the splash message (if set) instead of the executed module name.

Other

- The **MeVisLabPackageScanner** got an option `-printPackagePaths` that prints the package paths separated by line feeds.

- The FMEwork/ITK and FMEwork/VTK packages have been moved to MeVisLab/ITK and MeVisLab/VTK.

- The QtWebKit support will be removed with MeVisLab 4.0. This library has been superseded by QtWebEngine and isn't maintained anymore.

  If you currently use the `WebView` control, please replace it with the new `WebEngineView`.

  Direct embedding of MeVisLab panels isn't possible with the new control anymore (since the web engine runs in a separate process), but this can be emulated with Remote Rendering. If you use this feature, have a look at the code of the TestWebEngineView module for how this can be done.

Third-Party Libraries

A lot of third-party libraries have been updated in this release, too many to list them all in this place, so only a few will be mentioned.

See the complete list of available third-party libraries here.

Note that this is not the complete list of third-party components used in MeVisLab, for this you should consult the About dialog of MeVisLab, or use the `ThirdPartyInformation` module.

Note

As of 3.5 we will publish our whole third-party repository on GitHub (the actual code for the libraries is downloaded on-the-fly and built with Conan).

- Notable updates:
  - boost has been updated to version 1.78.0
  - dcmtk has been updated to version 3.6.6
  - Mesa (used in the software OpenGL driver on Windows) has been updated to version 20.3.4
  - Python has been updated to version 3.9.11.
  - Qt has been updated to version 5.12.12.
- Removed:
  - Box2d
  - Template Numerical Toolkit

Modules

- Added a new module `DicomImport`, which does a multi-threaded, asynchronous import of DICOM files. It replaces an old, deprecated module with the same name which called an executable for the import, which then wrote DICOM/TIFF pairs to the file system.

  There is a similar module in the Public SDK by our partners at Fraunhofer MEVIS, called `DirectDicomImport`.
- WEM modules:
  - `WEMSave`: Didn't auto-detect the `.geom` file suffix.
  - `WEMLoad`: Also recognizes upper-case file suffixes.
  - `WEMLoad`: Enforce binary STL format if file ends on `.stlb` .
  - `WEMLoad`: Could crash on loading certain `.wrl` (VRML) files.
  - `WEMSurfaceDistance`: `startPos` and `endPos` were swapped.
  - Fixed a thread-safety issue in `WEMIsoSurface` when running in background.
  - `WEMModify`: Rotation was not correctly applied if the scale factor was 1.
  - Fixed a crash in `SoWEMInteract` after reloading the module and then updating a connected input.

- CSO modules:
  - To make the purpose clearer, `CSOConvertToImage` has been renamed to `CSOVoxelizeContours`, and `CSOConvertTo3DMask` to `CSOGenerateSmoothSurfaceFromSparseContours`.
  - The deprecated module `CSOInterpolate` has been removed. It just wrapped the module `CSOShapedBasedInterpolation` by Fraunhofer MEVIS, which still exists.
  - Added a `SoCSOAngleEditor` module which allows to set/edit 3 points to define an angle in an image (the angle value is displayed).
  - Added a `CSOLabelPlacementDistanceLines` module for improved label placement for distance lines.
  - `CSOLabelRenderer`: Had a problem with a CSO (temporarily) consisting of only one seed point.
  - The `CSOFilter` module generated CSOs and CSOGroups with ids that were immediately changed without further notification.
  - Improved rendering of `SoCSODistanceLineEditor` when ghosting.
  - Improved shadow rendering of all `SoCSOEditorExtensions`.
  - All CSO editor modules: Change the maximum value for `defaultSeedPointSize` to 16.
  - `CSOIsoGenerator`: Improved symmetry of generated contours when **Interpolate contours** is on.
  - Fixed that `CSOList::removeAll()` would leak the memory used by the `CSOList` when undo was enabled, e.g., when done from the `CSOManager` with `removeAllCSOsAndGroups`.
- `View2DIsoContourShader`: Changed voxel value scaling slightly, so contours correspond better to those generated with CSOIsoGenerator.
- Made background and margin color of `SoLUTEditor` configurable.
- `SoView2DDrawVoxels3D`: Got an `autoSwitchDrawMode` option that adapts the `Disk*` and `Square*` modes automatically to the current viewer orientation.
- `DrawVoxels3D`: `copyInputImage` trigger field did not work correctly for `Off/Clear` update mode.
- `DicomRescale`: Also modifies the `PixelPaddingValue` and `PixelPaddingRangeLimit` tags now.
- Added `RescaleType` to the info fields of the `DicomRescale` module.
- Allow (temporary) invalid input for `AccumulateImage`.
- `VoxelizeMarkers`: Fixed that `distanceMode WorldDistance`, despite the name, operated in voxel space.
- Added `ThirdPartyInformation` module to list available third-party libraries. (Actually it was already there, but not visible yet.)
- Fixed the `ListModules` module.
- Loading a `CurveList` with a `LoadBase` module will print a warning about the loaded data being a memory leak. This is a conceptual problem that will be fixed in one of the next MeVisLab versions.
- Added new module `CurveCreator` to create a `CurveList` object from text data.

- Implemented an `isUnderMouse` attribute for `SoView2DRectangle` in the same way as was already done for `SoView2DLabel`.
- Added support for label drawing to `SoView2DRectangle`.
- One can't change the global variables of the `RunPythonScript` module itself in executed code anymore.
- The algorithm of the `Radon` module was re-implemented.
- Added new module `SoCoordinateSystem`, which does smarter rendering than `SoAxis`.
- `SoDiagram2D` uses the same font rendering as `SoView2D` now, which also allows to render any Unicode characters through Qt (with `View2DEnableQtFontRendering = yes` in your preferences file).
- Fixed that `SoView2DOverlayMPR` would print error messages if the overlay got smaller than one pixel.
- `Histogram`: Prints a warning if the mask image is used in label mode and the mask image has a float data type (which suggests it isn't a label image).
- `SoPostEffectMainGeometry` and `SoPostEffectTransparentGeometry` can work on buffers (e.g. created with `SoPostEffectCreateBuffer`) that are smaller than the actual render size to improve performance.
- `DicomDeidentify`: When defining special handling of private tags, private tags in a sequence were not handled correctly if the private creator tag for this tag was (only) in the same sequence (which is proper DICOM).
- `DicomTool`, under the **DICOM Send** section: Fixed that the `dcmSend` tool mode didn't work (anymore) on Windows.
- The `UndoManager` module didn't correctly apply the `maxNumberUndoSteps` when loading from a network.
- The module `ColorModelConverter` got a new mode "Gray to RGB" and overall better handling for when the wrong number of input color channels are present.

  Also renamed mode "RGB to RGB" to "Identity".
- The following modules are marked as deprecated now: `AngleLines2D3D`, `BaseBypassOp`, `ColorTable`, `ConnectedComponents`, `ContourManager`, `CSOBulgeProcessor`, `CSOFreehandProcessor`, `CSOIsoProcessor`, `CSOLiveWireProcessor`, `CSOModifyProcessor`, `CSOPrimitiveProcessor`, `CSOTransformationProcessor`, `DicomBrowser`, `DicomService`, `DistanceLine2D3D`, `DistanceTransform`, `Draw2D`, `EatDicomImport`, `ImportDialog`, `KernelFilter`, `LiveWire`, `LoG`, `MovieCreator`, `Negation`, `ObjectManager`, `OpenImage`, `RegionToContour`, `SaveImage`, `SoAngleLines`, `SoAngleToObjects`, `SoAsciiText`, `SoCake`, `SoDistanceLine`, `SoFixedFunctionShader`, `SoMainAxis`, `SoMinimalDistance`, `SoRuler`, `SoScreenSpaceAmbientOcclusion`, `SoShapeToPointSet`, `SoThresholdToPointSet`, `SoView2DCSOEditor`, `SwapViewer`, `WEMClip`

Modules provided by Fraunhofer MEVIS

Additions

- New module `PDFViewer` for embedding a PDF into an MDL panel.
- New XMarkerList modules:
  - `MarkerDistanceToPoint`

    - `MarkerDistanceToPlane`
    - `ProjectMarkersToPlane`
    - `FilterMarkersByPlane`
    - `FilterMarkersByDistanceToPlane`
    - `MirrorMarkersWithPlane`
    - `XMarkerPathLength`
    - `DuplicateXMarkers`
    - `CreateRandomXMarkers`
    - `AccumulateXMarkerLists`
    - `RigidTransformationFromMarkerLists`
    - `XMarkerListTransformation`
- New module `IntersectCSOWithPlane`.
- New DICOM modules:
    - Added new module `MultiFileVolumeListREGOutput` displaying lots of information of DICOM registration matrices.
    - Added new module `MultiFileVolumeListSMImageOutput` for displaying Whole Slide Microscopy information of DICOM files.
    - Added new module `MultiFileVolumeListDOCOutput` for extracting Encapsulated documents from DICOM files and new module `DicomDOCSave` to support export of Encapsulated documents as DICOM files.
- New `ParameterInfo` data structure for C++ and Python for documenting algorithm parameterization:
    - Basically represents a dictionary of information that can be passed between modules through `ml::Base`/Python objects (implemented via `QVariantMap` in C++ and `dict` in Python)
    - Comes with functionality to collect parameters from a MeVisLab network (see Python function `parameter_info.network_info.createMlabNetworkInfo`) with or without caching, extensible to other sources.
    - New `BaseWithParameterInfo` base class for C++ `ml::Base` objects that provide such info via `getParameterInfo()`.
    - Comes with `ParameterInfoWrappers` for Python access to C++ objects.
    - Comes with the `ParameterInfoInspector` module as output inspector for C++ and Python objects of type `ParameterInfo` or with a `getParameterInfo()` method.
    - See also:
        - Python API documentation
        - C++ API documentation.

- Added Python context managers to `fmeTestSupport.Context` for temporarily adding entries to a dictionary (AdditionalItems), or specifically to the environment (EnvironmentVariables)

Functional changes/Bugfixes

- PCL
  - Many PCL algorithms have changed alpha defaults (in e.g. RGBA values) from 0 to 255 which influence all algorithms and tests which depend on that.
  - `PCLMarchingCubesHoppe`, `PCLMovingLeastSquares`, `PCLConcaveHull`, and `PCLConvexHull` create significantly different results because the algorithms in the third party library PCL have been rewritten.
  - `PCLClusterStatistics` creates different, and better scaled clusterEigenValues now. The new eigen values have the same relation to each other than the old ones.
  - Removed the module `PCLVizualizer`, which is not really needed any more.
- Fixed `Tutorial_MeVisLabAndPython`.
- Updated help overview "DICOM in MeVisLab".
- Speed-up for `ImageCalculationByPlane`.
- `VideoLoad` was reworked.
- `itkImageFileReader` and `itkImageFileWriter`: do not support Philips REC/PAR image files any more.
- `itkImageFileWriter`: Some ITK file-I/O backends or respectively their used thirdparty-libraries do not recognize write failures into non-existing directories or on write-protected files. Some checks have been added for better recognition of such undetected write failures; however, not all can be detected.
- `InterpolateXMarkerList` now allows direct linear interpolation for marker vectors
- `ExtrapolateXMarkersPolynomial`: Fixed sampling of fitted result
- `GVRContourOverlay`: Improved contour rendering to avoid texture sampling artifacts and removed obsolete smoothness parameter from panel
- `OpenIGTLink` integration:
  - `OpenIGTLinkSenderModules` touches field `outMessageCreated` after the corresponding OpenIGTLink message has been asynchronously created
  - `OpenIGTLinkTrajectoryConversion` supports trajectory elements with negative radius
- AsynchronousIO framework: Fixed crash when using `SocketInterfaceWrapper::asyncWrite` and `asyncRead` caused by objects freed outside of the Python interpreter lock.
- DICOM
  - Applied some changes and fixes to support DICOM files > 4GB (as far as supported by the DICOM standard); added some error logs and checks for cases where inappropriate parameters are used.
  - Refactored, tested, cleaned up, and renamed Python wrappers for `MultiFileVolume[List]` and `DirectDicomImport`. Added wrappers for `MultiFileVolumeListOutputs`.
  - `DirectDicomImport`

- Fixed: Valid unicode strings in volume list were displayed as hexadecimal codes instead of correct characters.
- Fixed: **Tag Replacements** functionality would really just replace existing tags, not add new tags.
- Fixed: If no appropriate loader for a DICOM file was found, always a misleading REG load error was logged.
- Fixed: Correctly handles Raw data IOD without error messages now.
- Fixed: Tolerance specifications for `ImageOrientationPatient` tags in DPL configurations were ignored.
- Fixed: Invalid error messages about supposedly wrong dimensions of binary SEG data.
- Fixed: Load failures (or in rare cases incorrect load) of a few binary SEG files.
- Fixed: Import of some Siemens Set'n'Go Series with SetnGo postprocessor. Some modern ones, however, still may not be well supported.
- Fixed: Files with suffixes .0 - .9 are handled preferably as DICOM data before checking other loader formats.
- Fixed: Wrong handling of single frame Enhanced Multi-frame files.
- Fixed: Different behaviour of swap of `t` and `z` extent between `DirectDicomImport` and `itkImageFileReader`.
- Added warnings if volumes are composed which do not have temporally ascending time points.
- Added support for Philips 3D US data.
- Added support for correct display of Whole Slide Microscopy data. See also new related module `MultiFileVolumeListSMImageOutput`.
- Added option to warn of overflows in the internal tag/tree cache.
- Added programming support to deactivate some loader backends during runtime.

- Tag dumps (done by many DICOM related modules):
  - Added a number of private tag descriptions to make their decodings more readable (especially for a number of Siemens devices and from Mayo Clinic).
  - Added a tag decoder for text-like private DICOM tags to make dumps of their contents better readable.
- Added missing DICOM tags in exported DICOM FID files.
- `DicomEnhancedSave`:
  - Added support for many tags. Most (also non-mandatory) tags can at least be copied from incoming data sets.
  - Added some checks in strict mode to detect (and possibly refuse) if multiple series shall be stored.
- `DicomSEGSave`: completed support for saving multiple binary segments.

- `Dicom*Save` modules:
  - Fixed lost patient name if it contained non-ASCII characters.
  - Added SOP Common Module tags to select character sets.
  - Added more/better derivation sequence information by using either better defaults or refusing to save if the necessary information is not provided.
  - Fixed: Corresponding cache entries from the internal DICOM cache are removed on save now, so importing the file again will load the new version.
  - Better support and defaults of whether (Enhanced) DICOM tags are inherited from input data or not before it is used for export.
  - Relaxed too strict laterality checks.
  - Fixed: Unsafe handling of tags with value representation AT.
- `DicomTreeCompare`: Allows automatic cache clear before comparison to prevent comparison with outdated cache contents.
- `DicomConfigurableMessageFilter`: Improved/added configurable messages.

Version 3.4.3 Stable Release (2021-10)

New Features

- Implemented a `isUnderMouse` attribute for `SoView2DRectangle`.
- OpenSSL was updated to version 1.1.1l.

Fixes

- IDE: Referencing field controls in a note item didn't always work.
- IDE: The Output Inspector for vessel graph objects could modify the inspected object.
- IDE: Avoid warning messages when reloading the module cache for write-protected packages on Windows.
- MATE: Coloring of dots displayed for space characters (if enabled) could be displaced for some string constants.
- MATE: Avoid crash in module help editor if starting section does not exist.
- Removed the question mark in the title bar of dialog windows. This was a context help feature of Qt that was on by default, but nobody seems to be using it. If you, against all expectations, *did* use it, you can restore the flag from scripting with

```
from PythonQt import Qt
[...]
myWindowControl.widget().setWindowFlag(Qt.Qt.WindowContextHelpButtonHint, True)
```

- Fix rendering of `SoRenderSurfaceIntersection` if another module enables the z-Buffer in OpenGL.

- Fix some blend mode leakage of View2D text rendering, and some improvements to text rendering in `SoView2DMarkerEditor`.
- `SoView2DOverlayDecoration`: Improve border and tick mark rendering, especially if the underlay is rotated.
- `SoView2DOverlay` could scale float input values improperly when the underlay had a greater value range, leading to weak overlay images, especially if the value range difference was big.
- Avoid the "Matrix is no rotation" warning in `MPRPath` caused by floating number precision problems.
- `LoadBase`: Do not crash when trying to load an invalid vessel graph XML file.
- The "Retrieve" operation of the `DicomQuery` module could fail silently if the target was not available.
- Under certain circumstances scripting method `MLABImageFields::getFrameSpecificDicomTag` could return old values when the image in the field changes.
- Removing a DICOM tag which was previously added to the same overlay DICOM tree, which also had the tag in its parent tree, didn't work.
- Implemented `MLABDicomTag::value` for VR DT (date and time). It returns a `QDateTime` instead of throwing an exception.
- Improved label placing behavior of `CSOLabelRenderer` with option `tryToPlaceLabelsWithoutOverlap` set.
- Improved performance of `CSOIsoGenerator`.
- The OpenInventor modules `SoAsciiText`, `SoText2`, and `SoText3` for text rendering had issues and have been partially reimplemented.
- ADK (Windows): If a symlink can't be copied during installer generation (because of missing access rights), fall back to a full copy of the linked content.

Version 3.4.2 Stable Release (2021-05)

New Features

- WebEngineView:
  - The `WebEngineView` control gained the ability to print the current web page. (Set the `enablePrinting` attribute to get the context menu entry!)
  - Expanded the `TestWebEngineView` example module with a demo for embedding of Inventor viewers. Added a special RemoteRendering scripting wrapper for this.

    This is not as smooth as with the old WebView control though.
  - Also added a `removeObjectFromWebChannel` method to the `MLABWebEngineViewControl` wrapper, so that objects can be deregistered again.
- Python now provides the bz2 and lzma compression modules. For this the following ThirdParty libraries were changed:
  - Updated bzip2 to version 1.0.8.
  - Added XZ Utils (specifically the liblzma part), version 5.2.5.

- OpenSSL was updated to version 1.1.1k.

Fixes

- The module cache was not always updated on updates when located outside the installation directory (the default for macOS).
- Do not create incomplete auto-backup network files in the IDE (could be caused by a timer firing at an inopportune time).
- Fixed a crash in the MATE text editor when F1 was pressed for an MDL file and no validator entry exists for the tag under the cursor.
- A small color fix for the dark mode in MATE.
- Fixed an integer overflow in the MATE Python debugger for large integer values used as dictionary keys.
- MATE now highlights the "nonlocal" keyword in Python files.
- The window icon was not set on Linux.
- Some small rendering fixes for macOS 11 Big Sur.
- Avoid an unnecessary field update from a combo box `Field` control when losing focus.
- Allow a greater timeout value for the `IdleTrigger` module.
- Made `QUrl::FormattingOptions` work as arguments in the PythonQt wrappers.
- Fixed a potential crash in PythonQt when the receiver of a signal was removed inside the signal callback.
- Avoid a Qt assertion in `DicomTagBrowser` when expanding an empty sequence item.
- Handle invalid private tags in the "creator" range in `DicomDeidentify` by always dropping them (instead of throwing an exception).
- Fixed the ID handling of the `addCSOCopy` and `addGroupCopy` scripting methods on the `CSOList` wrapper.
- Fixed a crash in the `WEMThresholdToCSO` module when it was applied to non-triangle WEM patches.
- Fixed an additional offset in the C++ method `CSOGeometry::backprojectTo3D`.
- Fixed a crash that could happen in the `SoCSOLiveWireEditor` module if the user removed all seed points.
- Fixed the structure of closed CSOs generated by the `WEMClipPlaneToCSO`, `WEMThresholdToCSO`, and `WEMClip` modules, which were missing the closing path point list.
- Fixed the mouse over check of `SoView2DVoxelValue` when the `SoView2D` is under a `SoViewportRegion` (and `useManagedInteraction` is off).
- Fixed a buffer overflow of one byte in `WEMLoad` when loading non-WEM formats. This luckily didn't have any ill effects and was only detected through Debug mode on Windows, but still.
- Make the `acceptWheelEvents` attribute also affect the slider in a `Field` control.

- Do not crash when calling `acceptEvent/ignoreEvent` outside the event filter callback of the `EventFilter` control.
- `RTObjectSave` and other DICOM saving modules didn't support Unicode filenames.
- Small documentation improvements.
- For ADK/Application Builder users:
  - Fixed handling of the -runapp argument with the MeVisLab executable for users with the necessary license features.

Version 3.4.1 Stable Release (2020-11)

New Features

- Increased the maximum number of inputs of the `ConcatenateImages` module.
- PythonQt: Make the classes `QMutexLocker`, `QReadLocker`, and `QWriteLocker` work as context manager (in `with` statement) in Python code.
- The Field control gained the `browseModes` `OpenReadOnly` and `DirectoryReadOnly`.
- Removed the warning message when module panels of loaded networks are moved because they were outside the visible desktop area.
- Updated Qt to version 5.12.9.
- Updated freetype to version 2.10.4.
- Updated pydicom to version 2.0.0.
- For ADK/Application Builder users:
  - Can override the execution level for NSIS (Windows) installers by setting `INSTALLER_NSIS_EXECUTION_LEVEL` to the desired value.
  - Can change the name used for registry values, preferences file, etc. without changing the application macro name by setting `STANDALONE_APPNAME` in addition to `STANDALONE_MACRONAME`.

Fixes

- MATE: When using the graphical scrollbar in dark mode, the scroll location was almost invisible.
- Scripting: Fixed `MLABGraphic.createDesktopScreenshot` on Windows (still only returns the primary screen though).
- Scripting: `MLABDicomTree::parentTree()` returns None if no parent tree exists instead of crashing.
- `DirectDicomImport`: Enhanced Multi-Frame DICOM files with only one frame were not correctly loaded.
- The `SoCSOLiveWireEditor` module didn't update its internal state correctly if the input image changed but the visible slice number stayed the same.
- Fixed that the `SoView2DCSOExtensibleEditor` ignored the timepoint settings of extensions.

- The shadow of CSO seed points was not rendered correctly when using the MESA software OpenGL renderer.
- The check if a CSO is parallel to a plane only looked at the seed points of a CSO. But some types of CSOs only have one or two seed points, therefore the path points are now used too.
- Revised the `CSOIsoGenerator` algorithm to produce a more predictable result when applied on pixel patterns.
- The `CSOFilter` module copied CSOs and groups with an initial temporary ID that was changed afterwards without further notification. Now the original ID is used immediately.
- Fixed scripting error when clicking **Create New Package...** in the Preferences dialog.
- Fixed a crash in PyLint when editing a class derived from the OpenInventor `SoDragger` Python wrapper.
- Python: Merged the fixes for bpo-13487 and bpo-35615.
- Minor documentation fixes.
- For ADK/Application Builder users:
  - Fixed ZIP compression failing if files have a modification date before 01-01-1980.

Version 3.4 (2020-09)

Fixes / Enhancements (MeVisLab)

- IDE
  - Documentation: Added a section about recognized values in mevislab.prefs and possible environment variables to the MeVisLab Reference Manual.
  - You can provide a `PreloadModule` tag in your preferences file to preload custom modules in your IDE. This feature is meant to allow customization of the IDE (e.g. to set a variable to a file path that should be determined at runtime).
  - Where available we now use the SPDX license identifiers (from https://spdx.org/licenses) for third-party libraries in the about dialog.
  - Pressing the space bar in the network view of the IDE toggles to/from the special view, where previously you needed to hold the key to get it (which prevented tool tips from appearing).
  - Added the preferences variable `disableImmediateDebugOutputConsoleRefresh` which can be added to the preferences file to disable immediate console outputs, which can be slow and have unwanted side effects.
  - Fixed non-integer GUI scaling factors set with the environment variable `QT_SCALE_FACTOR`.
- TestCaseManager
  - Added a keyboard shortcut **Ctrl**+**F5** for running the selected test in the TestCaseManager (available on Windows and Linux).
  - Clear list of test functions if no test case is selected.

  - Ensure that tests started from a module's context menu and from the TestCaseManager do not interfere with each other.
  - Added tooltip to **Python coverage** check box (if checked) that Python Coverage might list additional Python modules.
- MATE (integrated editor)
  - One can set the expected outer scope for MDL include files (which helps MATE with auto-completion) by adding a comment line `// mdl-validator: ValidatorScope`, for example: `// mdl-validator: Window`.
  - Improved the file search in the project workspace.
  - Some improvements/fixes for the **Find In Files** dialog.
  - Added basic support for QML files, the GUI description language of Qt.
- C++ API changes
  - Removed some global button mask enums from `SoManagedInteraction`.
  - Do not crash when using a diamond shaped network pattern with automatic updates that leads to invalid image page accesses.
- Controls
  - Field control: Make `hintText` attribute work if `comboBox = Yes` and `edit = Yes`.
  - `ListView` control: Do not change column resize mode on hide/show of column.
  - Can select horizontal tabbing when editing cells in the `ListView` and `ItemModelView` controls with the `tabDirection` attribute.
- Scripting
  - If a control implemented with a Python class has the method `controlDestroyed` it will be called when the control's destructor is called.
  - The `QWidget.fontInfo()` and `QWidget.fontMetrics()` methods didn't work in PythonQt.
  - `MLABWebEngineViewControl` now has a method `setWebPage` to set an own `QWebEnginePage` instance.
  - Warn about unknown variables in `MLAB.replaceMDLVariables`.
  - Added a method `qmlModel()` to the `MLAbstractItemModelWrapper` which returns a model suitable for use in the `QuickView` control. Also added usage examples in the `TestQuickView` example module.
- Other fixes and improvements
  - MeVisLab has a changed logo, new icons and splash screens.
- Third-party libraries
  - Updated libjpeg to version 9d (Note: May return slightly different pixel values when loading jpeg files now)

- Updated libjpeg-turbo to version 2.0.4.
- Updated libtiff to version 4.1.0.
- Updated sqlite3 to version 3.32.3.
- Updated CryptoPP to version 8.2.0.
- Updated OpenSSL to version 1.1.1g.
- Updated TBB (Threading Building Blocks) to version 2019 U1.
- Python
  - Added cryptography package pycryptodomex, version 3.9.8.
  - Updated dateutil to version 2.8.1.
  - Updated SQLAlchemy to version 1.3.17.
  - Updated six to version 1.12.0.

Fixes / Enhancements (Modules)

- `SoView2D` and extensions
  - Added new zoom modes `VIEW2D_FIXED_PIXELS_PER_VOXEL_X_SCALED_FOR_HIDPI`, `VIEW2D_FIXED_PIXELS_PER_VOXEL_Y_SCALED_FOR_HIDPI`, and `VIEW2D_FIXED_PIXELS_PER_MM_SCALED_FOR_HIDPI` that scale the zoom value with `GLHiDPIScale`.
  - Fixed `SoView2DVoxelValue` behavior in `SoOrthoView2D` when not using Managed Interaction mode (no value was shown in two of the three views).
  - Fixed a crash in `SoView2D` when building the look-up-table for too large min/max value ranges.
- Other Open Inventor modules
  - One can now use the `#version` directive in `SoGVRShaderFunction`.
  - `SoViewportRegion` got more display options and can use a Python callback for calculation of position and size of the viewport. See the "Details" section of the module's help for how to do this.
- MeVis Path Tracer
  - Removed the broken `AutoUpdateColors` mode from the `SoPathTracerMesh` module.
- CSO
  - `CSOLabelPlacementGlobal`: Added start and end angles for label placement.
  - `CSOLabelPlacementGlobal`: Added a CSOList input that can be used as label placement boundary.
  - `SoCSOLabelRenderer`: Do not move explicitly placed labels when `tryToPlaceLabelsWithoutOverlap` is set.
- WEM

  - `WEMExtrudeCSO`: Fixed error handling.
  - `WEMExtrudeCSO`: Added new extrusion mode `RELATIVE_TIMES_DIRECTION`.
  - `WEMTreshold`: Fix crash on non-triangle-mesh inputs.
- ML modules
  - `EuclideanDistanceTransform`: Added an option to treat the surrounding border as "foreground" value.
  - Switched the `MLVolumetry` project (`Histogram` et al) to use type double instead of float for interface fields and computations.
  - `DicomDeidentify` can retain selected private tags now through an allow-list.
  - `DicomDeidentify`: Fixed some mis-handling of DICOM sequence data.
  - `ImageSave`: Added an option to prevent auto-adding of the file extension.
  - The default TIFF compression of `ImageSave` is now `LZW`.
  - `ImageLoad`: Avoid crash when trying to load unsupported float16 tiff images.
  - Fixed a crash in `DicomFrameSelect` if no frame-specific information was available.
  - `Arithmetic1`: Fixed handling of min/max value for values smaller than zero.
  - Exported helper classes from the `MLOpenCVModules` library for use with own OpenCV modules.
  - `CalculateGradient`: Fixed min/max value range calculation.
  - `Window`: Can handle negative image values now.
  - `RemotePanelRendering`: Can now limit the number of modules that can use the GPU at a time with the environment variable `MLAB_NUMBER_CONCURRENT_PANEL_RENDERING` to avoid a possible bottleneck with graphics drivers.
- Macro modules
  - Tried to make the module panels of `XMarkerListContainer` and `StringListContainer` clearer.
- Other
  - The `OpenCVUtils` Python module has been moved from package FMEwork/ReleaseMeVis to MeVisLab/Standard.

Modules and Libraries Provided by Fraunhofer MEVIS

Modules

- Additions
  - New modules `LoadYAML` and `ReadYAML` for reading YAML files or strings.
  - New module `DirectoryIterator` to conveniently iterate through directories on MeVisLab network level.

- Added `XMarkerListDistance` which computes the distance between two XMarkerLists.
- New module `SeverityChannelLoggingZmqSource` to log received ZeroMQ messages via SeverityChannelLogging (Boost.Log-based).
- Started FME Python Toolbox (Public SDK) documentation with at least few documented packages (hopefully growing).
- New module `ThirdPartyLicenseReport` for comparing licenses of third parties included in MeVisLab against an allow-list.
- New MDL controls
  - `DockWidget` for adding Qt dock widgets (https://doc.qt.io/qt-5/qdockwidget.html) to application windows.
  - `GUIScriptTemplate` for reusing the same MDL GUI with different field names (best used with MDL style prototypes). Search for GUIScriptTemplate in .script files of FMEstable/ReleaseMeVis for examples.

- Extensions and changed behavior
  - `fmeTestSupport.Paths` python module: renamed `IsSubPath()` to `isTrueSubPath()`, improved implementation, and added `isSubPath()` that also returns true for equal paths. New method `normalizedPath()`.
  - The `CachedPath` module for remote directory file/dir caching is no longer experimental, improved the underlying `file_caching` Python package.
  - Improved the error handling of the `ImageFromFile` module.
  - The `HistogramFeatures` module now uses the double datatype.
  - The library `[ML]AsynchronousIO` has been refactored to expose an abstract interface for I/O channels (such as Sockets) in order to transparently support TLS sockets. No changes to networks should be necessary as deprecated names have been introduced.
  - The `[ML]OpenIGTLinkProtocol` libraries have been refactored and separated into an independent C++ as well as two ML dependent libraries `MLOpenIGTLinkConversion` (to/from ML datatypes) and `MLOpenIGTLinkCommunication` (ML modules). The `MLOpenIGTLinkProtocol` library now is only responsible for channel I/O and supports custom message types via factory plugins. There are incompatible changes of module interfaces: The `OpenIGTLinkProtocol` module does not support the device name field anymore as this is message specific and should be set on the message creators (such as the ML modules or own message creating code). The convenience macro modules `OpenIGTLinkServer` and `OpenIGTLinkClient` should be compatible.
  - Multiple improvements and fixes in `MultiFileVolumeListDraftView`.

- DICOM and file IO
  - New experimental modules:
    - `MERACrawler`, `MERAQuery`, and `MERAVolumeInspector` to create and access search indexes to image/DICOM volumes.
    - `DicomEnhancedSave` to export Legacy Converted Enhanced MR DICOM files also supporting compression.

- `DicomSegSave` can also export fractional SEG images.
- Removed `.rek`/`.rec` file support.
- Many DICOM export modules perform stricter checks for the `Laterality` tag to prevent DICOM standard violations.
- `DirectDicomImport` implicitly converts volumes with formerly left-handed world matrices on irregular volumes to right-handed ones.
- Loader backends implemented with modules have been moved into the project `ModuleLoaderBackends`.
- A new loader backend for `DirectDicomImport` automatically supports histology file formats `.btf`, `.czi`, `.dzi`, `.jp2`, `.mi`, `.mrxs`, `.scn`, `.svs`, `.tif`, `.tiff`, `.wsv` on systems where the HistoLoad module is available (it is not included with the Public SDK).

Windows Edition

- Installer: The uninstaller will remove the environment variables MLAB_ROOT and MLAB_COMPILER_VERSION from the environment again if they are still unchanged.

macOS Edition

Note

This release requires at least macOS 10.14 (Mojave) and Xcode 10.3 for development.

Note

This version will probably not work with macOS 10.16/11 (Big Sur). We recommend to refrain from updating to this macOS version if you need to use MeVisLab 3.4.

ADK (also known as Application Builder)

- Can use <, <=, >, and >= lexical string comparison in IF statements in `.mli`/`.mlinstall` files.
- The splash screen of own applications now supports semi-transparent pixels.
- We added the `-fail-silently-if-license-invalid` cmdline argument to MeVisLabApp.
- Automatically set window icon on Windows if a file `<applicationname>Icon.png` exists in `Packages/MeVisLab/IDE/bin`.
- MeVisLabWorkerService: Make sure workers are used in the order they were started.
- Use correct `ReleaseOptimized` variable for standalone-installers, which activates some performance optimizations.
- Ignore already encrypted Python files when pre-compiling Python files for a stand-alone installer.
- Improved Python parsing of the `raise <exc> from <exc>` statement in the ModuleDependencyAnalyzer.

Version 3.3 (2020-03)

Fixes / Enhancements (MeVisLab)

- IDE
  - One can now promote a local macro to a global macro from the context menu of the module.
  - Fields of imported panels in the GUI only allow dragging to/from other fields now if they are exported as `internalName` on the module of the window.
  - Closing the IDE will now also close all module panels (if MeVisLab should be terminated).
  - Prevent recursive reloading of modules while pressing and holding **F5** key.
  - Fixed 'object' not working in console of the PythonObjectInspector - it's '_object' now.
- TestCaseManager
  - Ask the user if the debugger should be disabled if Python coverage is activated (since only one of these can work at a time).
  - Function parameters can now be passed as dicts (keyword arguments instead of position arguments) to iterative tests in a test case.
  - Improved error handling for secure testing mode.
  - The `tearDownTestCase()` function is also called for cancelled tests now.
- MATE (integrated editor)
  - MATE got a dark mode on Windows and Linux (available from the **Preferences** panel).
  - Added support for mypy besides Pylint (not simultaneously though).
  - Update default Pylint version to install to 2.4.1. Also added inference limit config value, which can improve performance when set to a lower value.
  - For modules with several scripting `source` files, Pylint (or mypy) is applied to the combined source to improve the output - note that this is only detected if there is an associated module in a connected MeVisLab instance.
  - The **Watch** area in the Python debugger got a context menu command to remove all watches at once.
  - Improve the behavior of the Python debugger if accessing object attributes to display them in the debugger causes Python exceptions.
- C++ API changes
  - Added method `CSOGeometry::computeClosestPathPoint` which really returns the nearest path point of a CSO.
- Controls
  - Added a `tooltipField` attribute to all controls, which allows to set the tool-tip of a control through a field.

- Fixed initial visibility of buttons using the `visibleOn` attribute in a `ButtonBox` control.
- Fixed sorting in `ListView` control if initial item values are changed later on.

- Scripting
  - Users can now add a custom entry to the context menu of modules. Just add a `.def` file with a `ModuleAction` to your package, which should have a `title`, `condition` (a short Python expression) and a `userScript` attribute. Have a look at the ModuleAction in `Packages/MeVisLab/Standard/Modules/Macros/Wizards/LocalToGlobalUserScript/LocalToGlobal.def` for an example.
  - Added method `enableAlternatingRowColors` to the `MLABListViewControl`.
  - Added method `setAllowRenaming` to the `MLABIconViewControl`.
  - One can get the parent Dicom tree of a `MLABDicomTree` wrapper with `parentTree()`.
  - Added the scripting commands `MLAB.prefsFileNames()` and `MLAB.userPrefsFileName()`.
  - Allow to set additional image dimension information on an `MLWritablePagedImageWrapper` with `setCDimensionInfos()`, `setTDimensionInfos()`, and `setUDimensionInfos()`.
  - Added `deepCopy` method to the `CSOList` scripting wrapper.
  - Fixed MeVisLab hanging when using keyword arguments on a slot that didn't have the `kwargs` parameter.
  - Fixed the `requestStopProcessing` call on remote modules to not apply to anything that is sent to the remote module after this call.
  - Changed output of `MLAB.detailedSystemId()` and `MLAB.systemInfo()` (at least on some platforms). `MLAB.detailedSystemId()` is deprecated now, since it actually isn't very detailed. We recommend to use `PythonQt.QtCore.QSysInfo` instead.
- Other fixes and improvements
  - One can now set the variable `ProfilingTraceFile` in the preferences file to employ the profiler from the start to debug startup problems.
  - Added a `-printJSONInfo` option to the MeVisLabPackageScanner.
- Third-party libraries
  - Updated Qt to version 5.12.6.
  - Updated fontconfig to version 2.13.1.
  - Updated libzeromq to version 4.3.2.
  - Added the dcmmkdir tool from the dcmtk library.
  - Python
    - Updated pyzmq to version 18.1.0.
    - Updated jinja2 to version 2.10.3.

- Updated MarkupSafe to version 1.1.1.

Fixes / Enhancements (Modules)

- `SoView2D` and extensions
  - Improved placement of `SoView2DLabel`.
  - `SoView2DLabel`: Can set margin to background borders.
  - `SoView2DOverlayDecoration` got an option to draw the border outside the image.
  - Added the `updateLayout` method to the `SoView2D` scripting wrapper to update the viewer layout manually.
- Other Open Inventor modules
  - Make `SoMouseGrabber` switchable to Managed Interaction, and use that feature accordingly in `InteractiveRampLUT` and `View2D`.
  - The `SoXxxxMenu` menu widgets got a field to control the factor for darkening/brightening on mouse-over.
- MeVis Path Tracer
  - Added support for IBL_Equirectangular projection to `SoPathTracerBackgroundLight` (automatically detected from input image).
- CSO
  - Added filled rectangle, filled circle, and filled triangle to seed point styles.
  - Seed points now scale with the line thickness.
  - Improved interactive movement of labels.
  - Added new flag `restrictToImageBoundaries` on `SoCSOPointEditor`.
- WEM
  - Improved the WEM output inspector.
  - One can select a simple sub-division mode in `WEMSubdivide` now.
  - Fixed that PVL values were doubled in any `WEMProcessor` module.
- ML modules
  - Added a bunch of graph editing modules:
    - `GraphManager`: Enables graph editing with undo/redo.
    - `LabelTrees`: Sets labels to trees in order of descending size.
    - `LabelSelectedTree`: Sets labels to a selected tree.
    - `LabelFromEdgeToLeaves`: Sets labels from a selected edge down to the leaves.

    - `LabelByThresholdedVolume`: Sets labels for image voxel values under the skeleton positions.
    - `GetTreeLabels`: Retrieves tree labels.
    - `ConnectEdges`: Connects edges in a graph.
    - `DisconnectEdges`: Disconnects or removes edges in a graph.
    - `RemoveTree`: Removes a tree.
    - `SetRoot`: Sets a new root in a tree.
    - `SplitGraph`: Splits a graph and removes trees if label values criterion is not met.
    - `SetLabelToTreeDepth`: Sets labels accordings to tree depth from roots to leaves.
    - `SetImagePropertiesToGraph`: Sets image properties (matrix, image extend) to a graph.
    - `GraphToMarker`: Converts a graph (nodes, edges) to XMarkers (Positions, Vectors).
    - `GraphViewer`: Offers a view of the logical structure of a graph.
  - `Resample3D`: Ensure that the output size is at least 1 in each dimension.
  - `VoxelizeInventorSceneGPU` works on Radeon cards now.
  - `SoImageFileSampler`: Added options to generate seamless and Mip-mapped textures.
  - Added the `DicomDeidentify` module for anonymization of DICOM data.
  - Fixed handling of `WindowCenter` and `WindowWidth` for VOI LUT Function "LINEAR" in `DicomLUT` and `RampLUT` modules - note: this needs to be enabled in existing networks.
  - `LUTCombiner` now allows scaling of the input LUTs.
- Macro modules
  - `DicomTool` does not write multi-frame information to the individual slices it saves anymore.

Modules and Libraries Provided by Fraunhofer MEVIS

Modules

- Additions
  - New module `NormalizeGeometry` for standardization of image geometry (orientation and resolution).
  - New modules for fast and customizable cheapest path search on XMarkerLists (`XMarkerListToCostGraph`, `CostGraphCheapestPath`, `CostGraphGenericCostPlugin`, `CostGraphEdgeLengthCostPlugin`, `CostGraphClosestVertex`).
  - New module `XMarkerListImageData` that reads image data at marker positions and modifies marker properties (vector, type) based on it.
  - New inventor module `SoToggleOnResolution` allows to toggle scene graphs based on the viewer resolution and/or the viewer ratio.

  - New module `CSORemoveSelfIntersection`.
  - New GVR extension module `GVRGammaCorrection`.
  - New module `CompareScreenshotsStable`.
  - New modules `UDPStringSender` and `UDPStringReceiver` for sending and receiving strings over asyncIO UDP sockets.
- Improvements & extensions
  - DICOM and file IO
    - Several internal quality improvements and minor fixes.
  - PCL
    - The output inspector also displays PCL object information now.
  - Improvements to SimulatedHighDPIViewer:
    - added new rendering mode `Offscreen` that fixes problems with some inventor modules, such as `SoViewportRegion`.
    - added event forwarding.

Linux Edition

- Fixed that MeVisLab could not be started directly from the installer.

macOS Edition

Note

This release requires at least macOS 10.13 (High Sierra) and Xcode 9.4 for development. macOS 10.15 (Catalina) is not officially supported yet.

- MeVisLab will provide the root certificates of the macOS system keychain to OpenSSL automatically if none exist at default locations and the environment variable `SSL_CERT_FILE` is unset. Default locations searched in order are:
  1. `~/Library/Application Support/<BundleName>/Certs/cacert.pem`
  2. `~/Library/Caches/<BundleIdentifier>/Certs/cacert.pem` (this is the place, where the auto-generated certificates archive is stored)
  3. `~/Library/MLAB_OpenSSL/cert.pem`
  4. `/Library/MLAB_OpenSSL/cert.pem`
- Allow to setup quick-start permanently via the Preferences panel in MeVisLab.
- PythonPip macro module installs/updates Python modules into the user's Python library (e.g. `~/Library/MLAB_Python/3.6/lib/python/site-packages`) and the integrated Python interpreter loads modules from this location.

- Store Module Caches outside the installed App Bundle (`~/Library/Caches/<BundleIdentifier>/ModuleCache`) and watch version to invalidate cache if necessary.
- Keep dynamic libraries in their original packages when building an App, this will prevent duplicated libraries and reduce App size.

ADK (also known as Application Builder)

- Support +L? and +?L for optional non-recursive collects in .mli/.mlinstall files.
- Applications started with -runappbatch don't show the splash screen now.

Version 3.2 (2019-08)

Qt 5.12 update

We have updated our Qt version to 5.12.3. Between Qt 5.6 and Qt 5.12 some things have changed that might need your attention:

- The (used) license for the Qt libraries is now LGPL 3 instead of LGPL 2.1.
- The license for the Qt helper tools is now GPL 3 instead of LGPL 2.1, which means you can't link against them (which one doesn't do anyway because they are stand-alone tools) without publishing your code under a similar license. Some of these tools you will use when, e.g., compiling your own C++ modules, but none of them will be included in stand-alone applications by our installer wizards.
- The qmake tool needed for translating .pro files into platform-specific project files now expects the compiler executable to be on the command-line on Windows and Linux. We tried to make sure that this happens automatically, but in case this doesn't work for your setup you should call the `vcvarsall.bat` from your Visual Studio installation with argument `amd64` before calling the ToolRunner tool or the batch files created alongside the `.pro` files.
- The `QUrls` returned when clicking a non-standard link in the `WebView` control might have a slightly different string representation than before. If you used this feature you should check if you need to adapt your code to the new format.

Fixes / Enhancements (MeVisLab)

- IDE
  - Single files listed in the `Deployment` section of modules do not cause further dependency analysis.
  - Global modules don't have precedence over local macros anymore. Only if no local macro with the given name exists a global macro with the same name is used (and vice versa).
  - One can disable persistence of module panel positions and panel state by setting the `DisableModuleWindowsPersistence` variable in the preferences file. This helps avoid unnecessary differences in version control systems.
  - Only print package dependency warnings when loading networks in the IDE.
  - Longer enum value lists in the HTML help of a module get a scrollbar.
  - The automatic panel of modules can be opened with **Alt**+**[Shift|Control]**+**double click** and **Alt**+**Enter**.

- Added the facility to display modules with a custom icon (per module type) from a user script that can be selected in the **Preferences** dialog.
- Fixed Python debugger updating local variables too late when function `locals()` (or something similar) is used in code.
- Automatic reload of modules on double-click also considers changed MDL includes now.

- TestCaseManager
  - Allow to run GoogleTest tests from a FunctionalTestCase. Example (to be put into a TestCase's python file:

    ```
    from TestSupport import CodeTest
    CodeTest.inject("$(MLAB_MeVisLab_Standard)/CodeTests/bin/MLBaseTest", ctx)
    ```

  - TestCases can decide to display the used sorting literals in the TestCaseManager (by setting `showTestFunctionSortingLiterals` in the `.def` file).
  - When typing in the filter line of the TestCaseManager, up and down keys change the selected visible test.
  - You can tell a test to use software rendering with the new keyword `preferredRenderer=software` (in the `.def` file of the test) when using secure testing (only on Windows, and on Linux when using the Mesa graphics driver).
  - Tests are by default always run with **Secure testing** if the environment variable `MLAB_USE_SECURE_TESTING_BY_DEFAULT` is set.
  - Module help screenshots made on HiDPI screens are now scaled down automatically.

- MATE (integrated editor)
  - Make dot color (when visible whitespace option is active) configurable.
  - Made the Pylint timeout configurable.
  - Added new keyboard shortcut **Ctrl**+**F3** to find the selected text.
  - Closed documents can be removed from the document session now.
  - Python syntax highlighting in MATE is now more exact with string and number literals. (Did you know that 1.3j is an imaginary number in Python?)
  - One can define and run custom commands from the project workspace context menu now.
  - Fixed and extended **Go Back**/**Go Forward** functionality. It now more closely resembles the functionality you may know from, e.g., Visual Studio.
  - Toggling between `.script` and `.py` file (default binding is **Alt**+**Down**) usually also works now if not connected to a module instance in MeVisLab.

- C++ API changes
  - The `SoPointingAction` class can now declare it doesn't want dragMoveTo calls for mouse clicks, even if the mouse was slightly moved during the click.
  - Added a convenience macro `SO_NODE_ADD_ENUM_FIELD` to the Inventor headers.

- SoDiagram2DExtensions now have more options in which layer to draw.
- The AdjustDimInfo<> template was moved from the code of the SubImage module to the MLTools library.
- Introduced a new ml::OpenGL::getVideoMemorySizeInMB() method which is smarter about returning the texture/video memory actually available in OpenGL.
- Implemented the Base::deepCopy method on BaseList (and XMarkerList).
- View2DSliceList::getSliceNormal() always returns the real slice normal (and not just the z vector) now.
- The SoFieldData::getRemovedFieldNames() method of Inventor nodes is evaluated in MeVisLab now.

- Controls
  - Added the WebEngineView control, which is similar to the WebView control, but uses the QtWebEngine widget based on Chromium instead of the outdated QtWebKit widget. The new control doesn't allow access to modules, scripting, or module panels of MeVisLab, since the browser engine runs in a separate process, but since the old widget isn't supported anymore we will need to remove the old one in one of the next versions and therefore we urge you to switch if you currently use WebView.
  - The expandY attribute of most controls defaults to auto instead of no now, so you now should rarely need to set this yourself.
  - Added a validator tag to the ComboBox control, which takes a regular expression against which to check the input of an editable ComboBox.
  - Added a insertSeparator method to the ComboBox script wrapper.
  - Added methods setModulePanel and setModuleWindow to the DynamicFrame control.
  - Implemented auto-indentation for the PythonTextView control.
  - Added support for direct texture transfer between processes in RemoteRendering (only on Windows, and on the same computer). Set the useDX11SharedTexture attribute on the RemoteRendering control to activate.
  - Added popupMode = [Instant|Delayed|MenuButton] support to the ToolButton control.
  - Added titleField support to the ToolButton control.
  - Added the acceptWheelEvents tag to some controls — default is true.
  - Added the tag comboEditable to the Field control.
  - Added a componentTitles attribute for Field and VectorEdit controls, and setComponentTitles and setComponentTitleAt scripting methods, which allow to override the default component titles of a vector edit control.
  - Fixed initial visibleOn state of panels included with the Panel control.
  - Added visibleOn expressions for MenuItem entries.

  - `visibleOn` expressions on controls are evaluated delayed to fix panel layouts growing in size in some situations. This could happen when an intermediate state was used for size calculation where widgets were temporarily visible together that were actually never shown together in practice.
  - Fixed layout flickering on `Label` controls with a frame and trimming enabled.
  - Added a `windowActivatedCommand` to the `Window` control.
  - The `ListBox` scripting wrapper got a `sort` method.
- Scripting
  - When calling unknown methods on objects derived from `QWidget` a hint is printed that the user perhaps needs to do `from PythonQt import QtGui` first.
  - We enabled multi-threading in Python. In case of problems you can disable it with an entry `PythonMultiThreading = NO` in your preferences file.
  - Added `MLABModule::callOnGUIThread` scripting method to call a scripting function from any thread.
  - The Qt scripting bindings were updated for version 5.12.
  - Added an `MLABSoPathField` class which allows to access the Inventor node path result of (currently only) module `SoPicking` from scripting.
  - Removed the method `MLAB.macRunPrivilegedCommand`.
  - Removed the method `MLAB.gpuTextureMemoryInMB`, rather use `MLAB.gpuVideoMemoryInMB`.
  - `MLABImageField.mapVoxelToWorld` and `.mapWorldToVoxel` scripting methods also work on the Inventor `SbVec3d` class now.
  - Added scripting methods for adding and removing vessel voxels to the `ml::Graph` scripting wrapper.
  - Moved method `MLAB.priv().aboutThirdPartyHTML()` to `MLAB.aboutThirdPartyHTML()`.
  - The arguments to `OpenVDBGridWrapper::toWEM` were switched around. If you used this, please adapt your code.
  - Added a new scripting wrapper that allows to find the real nearest point of a WEM triangle mesh (and not just the nearest node of the mesh). Obtain this wrapper with `wem.createNearestPointOnSurfaceObject()`.
  - Fixed that `WEMWrapper::removeWEMPatchAt` took the argument as id of the WEM instead of as index. Also added `WEMWrapper::removeWEMPatchById`.
  - Added support for async Python functions in MeVisLab in general and in field listeners and test functions in particular.
  - Added a script wrapper for the `MLABRecentFilesHandler`, accessible with `MLAB.createRecentFilesHandler`.
  - Added the `MLABTreeWrapper::deepCopy()` scripting method.

- Added scripting method `MLABPackageManager.getMeVisPythonExecutable` which returns the location of our Python interpreter executable.
- Added the `getTransform` method to the scripting wrappers of `SbMatrix` and `SbMatrixd`.
- MeVisLab now changes the `PATH` on Windows in a way that this change is also visible from Python in `os.environ`. This fixes that augmenting the `PATH` from Python would overwrite the changes from MeVisLab, which prevented loading of modules, e.g., after installing and using an externally installed `NumPy`.

- Other fixes and improvements
  - We changed our policy on unlicensed use: An unlicensed MeVisLab SDK treats modules signed by non-MeVis owners as unlicensed now.
  - Fixed an issue with too long filenames generated in the Python coverage detection of the TestCenter.
  - Fixed that links to overview documents could not have a different title in mhelp files.
  - DICOM image loading duplicates the value of the `PixelSpacing` or `ImagePixelSpacing` tag when this tag only contains one value (thus fixing invalid DICOM files).
  - Project wizards can have a `condition` (a Python expression) which defines if the wizard should be shown.
  - Make module dependency analyzer detect DLLs from scripting wrappers.
  - Added some `DicomTree` error logging.
- Third-party libraries
  - Updated Qt to version 5.12.3.
  - Updated QtWebKit to version 5.212.
  - dcmtk
    - Updated dcmtk to version 3.6.4.
    - Added tools dcmrecv and dcm2pnm to SDK.
  - Updated OpenCV to version 4.0.0.
  - Support for utilizing ffmpeg DLLs has been added to OpenCV on Windows.
  - Python
    - Updated NumPy to version 1.16.4.
    - setuptools has been updated to 40.8.0, pip to 9.0.3.
    - Building the boost Python binding now.
  - Upgraded the Mesa fallback software OpenGL driver for Windows to version 18.1.5.
  - googletest has been updated to version 1.8.1 and googlemock has been newly added.

- Updated libxml2 to version 2.9.9.
- Updated libpng to version 1.6.37.
- Updated OpenSSL to version 1.0.2s.

Fixes / Enhancements (Modules)

- `SoView2D` and extensions
  - Qt font rendering is now the default in `SoView2D`.
  - Added `textShadowColor` field to `SoView2DAnnotation`.
  - Added line stipple pattern to `SoView2DPosition`.
  - Fixed clipping planes in `SoView2DScene` when coordinate system is left-handed.
  - Added scripting wrappers for coordinate transformations to `SoView2D` and `SoOrthoView2D`.
  - Added alpha value parameter to `DrawVoxels3DPreview`.
  - Improved the undo behavior/interface of the `DrawVoxels3D` module.
  - Added an option to restrict drawing of the background color in `SoView2D` to the image area (useful when using a white background color and a blend mode like `Minimum`).
  - `SoView2DRigidRegistrationEditor`: Added an alpha field and a trigger to reset the rotation center, also made rendering of shadow optional.
  - Fix arrow lines not showing in `SoView2DMarkerEditor` in certain corner cases.
- Other Open Inventor modules
  - Multiple `SoRenderSurfaceIntersection` modules share the same buffers now and the buffers are only resized if necessary.
  - Added the option **Only render opaque objects** to the `SoPostEffectMainGeometry` module.
  - The `View3D` module can be switched to `RayCaster` mode now.
  - Improved multi-core and hyper-threading usage of GVR framework, switched to using OpenMP.
  - GVR can render transformed volumes outside of the main volume now.
  - Switched GVR to use geometry shaders for the slab rendering by default.
  - Extended `SoMarkerListPointSet` and `So3DMarkerEditor` to allow filtering XMarkers by timepoints.
  - Added a new `SoCameraWidget` module for moving the Inventor camera from a widget.
  - Added new `SoSlideWidget` module similar to `SoCameraWidget`, intended for touch navigation in viewers.
  - `SoPicking` got a `highlighted` output flag and can provide two extra pointing actions (because sharing the same sub-scene between different SoPicking modules does not work).

  - `SoPicking` allows to get the picked color now.
  - Implemented an update mode for the `SoSceneLoader` module.
  - The `AnimatationRecorder` has been improved.
  - Removed the `font` field from `SoExtText2`, added a `fontSize` field instead, did the same for `So3DXMarker` and `So3DMarkerEditor`.
  - Add relative text placement in `SoExtText2`.
  - Allow GPU filling in `SoRenderSurfaceIntersection` when using the CPU intersection routine.
  - Added `LoadInventorFile` module to load `.iv` scene files.
  - `SoGVRAmbientOcclusion` supports other OpenGL buffer formats now and allows to specify a name for the sampler.
  - `SoVascularSystem`: Added smart snap mode to `highlightMode` which highlights the nearest node towards the root within a given range.
  - `SoInput` works with network paths on Windows now.
  - `SoViewportRegion` can now scale pixel values automatically for HiDPI screens.
  - `SoFramebufferSampler2D/3D` and `SoCreateCubeMapSampler` now ignore the lights in the scene above.
- MeVis Path Tracer
  - Added volume ray to slice projection.
  - Support transformations and texturing.
  - Added bloom effect to `SoPathTracer`.
  - Added silhouette effect to `SoPathTracerMaterial`.
  - Added support for 8bit, RGBA, and gradient volumes.
  - Force full quality updates in `SoPathTracer` when rendering to `OffscreenRenderer`.
- CSO
  - Added a `WEMExtrudeCSO` module for extruding CSOs to a WEM surface.
  - `SoCSOLiveWireEditor` only requests the input image if really required.
  - `SoView2DCSOExtensibleEditor` now sets the `maskValid` flag when interacting.
  - A new MLCSO API `CSOGeometry::computeIsInPlane` is used in `CSOTrailRenderer`.
  - Added a wrapper for the method `CSOGeneratePathPoints::fillPathPointEllipse` (and added an optional `pointDistanceInMM` parameter to all `recompute...` methods).
  - Added a `close` method to the CSO's Python wrapper to close an open CSO.
  - Seed points can also be rendered as triangles now (see module `SoCSOVisualizationSettings`).

    - SoCSOIsoEditor also works correctly on other timepoints than the first now if an image has more than one timepoint.

- WEM
    - WEMModify got an option to merge all patches into one. This functionality is also available from the WEM scripting wrapper.
    - WEMSelectPatches can now filter by any PVL.
    - Added a scripting wrapper method to remove a PVL from a WEM.
    - WEMInfo now shows more information about PVLs.
    - Added option to make SoWEMRendererExtensions unpickable.
    - SoWEMRenderer now offers information whether mouse is over surface in performPreHitTest mode.
    - SoWEMBulgeEditor
        - Got an option to switch between local bulging and dragging the whole WEM patch.
        - Improved drag behavior.
        - Made amount of changing of the influence radius with mouse wheel configurable.
        - Various other improvements.
    - VoxelizeWEM works for multiple patches now.
    - Fixed finding of a starting position for polygonization in WEMVascularSystem.
- ML modules
    - Added support for FloatPixelData DICOM tag to ImageSave and ImageLoad.
    - ImageSave can't save RGBA images in DICOM mode anymore, this wasn't done correctly anyway.
    - Added option to only render vessel voxels to the output image of the GraphToVolume module.
    - RemotePanelRendering can render modules in parent networks now (by specifying the "parent:" prefix), not just sibling modules.
    - Fixed some mouse coordinate offset problems with RemotePanelRendering.
    - Added new error handling mode ExactlyOneOpenInputIsError to the ImageCompare module.
    - One can now specify the page extent for the ImageCompose module (and for the VirtualVolume class).
    - DicomFrameSelect also gives a valid output if there are no frame specific tags at all.
    - Renamed/added some fields on module DistanceFromXMarkerList, and changed the behavior if the input marker list is empty.
    - The Sobel3D module now has an Estimation mode better suited for usage with GVR/PathTracer.

  - The heuristic for determining the 'best' keyframe for mapping markers in the `MPRPath` module when the option **Markers correspond to frames** is off has been changed (and hopefully improved).

- Macro modules
  - Some old modules were deprecated: `SwapViewer`, `Negation`, `DicomBrowser`, `DicomImport`, `DicomService`, `OpenImage`, `SaveImage`, `ImportDialog`.
  - Added support for faster `dcmsend` application to the `DICOMTool` module.
  - `ImageLoadMulti` can also work with 0-based indexing now.
  - Added `dimension` parameter to `SignedEuclideanDistanceTransform`.
  - Module `DicomTool` can store/send image blocks with a gantry-tilt now.

Modules and Libraries Provided by Fraunhofer MEVIS

Modules

- Additions
  - New module `ShortestPath` for Dijkstra calculation of the shortest path in an image.
  - New python package `file_caching` that helps when working with slow, possibly remote and unreliable data sources through semi-transparent on-demand caches configured on a per-system level.
  - New (experimental) module `CachedPath` that wraps some of the `file_caching` functionality into a macro module.
  - New helper modules `Switch2` (switches between exactly two image inputs) and `AutoSwitch` (auto-switches between one default and an optional 'override' image input).
  - New module `CSVReader` for reading comma-separated value files.
  - Added module `ModuleHtmlPostProcessor` which can transform module help files into something suitable for application documentation.
  - New documentation: **FME Python Toolbox (Public SDK)** is a reference of the Python packages in the Fraunhofer MEVIS part of the SDK.
- Improvements & extensions
  - DICOM and file IO
    - Extended and revised the C++ interface of the DICOM libraries with many improved checks.
    - `DICOMTreeItemModel` now also takes an MLBase input which may be a `DicomTree` or a `MultiFileVolumeListRefCounted` (from `DirectDicomImport` and related modules).
    - Added private creator "Fraunhofer MEVIS" with private tags `SourceIndexOfDecomposedFrame` and `SourceSOPInstanceUIDOfDecomposedFrame`.
    - Improved handling of SOPInstanceUIDs from enhanced multi-frame files.

- Added "**Frame Tags View**" tab in `MultiFileVolumeListDiagnosis` to get easy overviews about same tag in multiple frames.
- Added further suppression to `DicomConfigurableMessageFilter` modules.
- Added sorting for tag "`DimensionIndexValues`" in default configuration of `DirectDicomImport` to improve imports of Slide Microscopy (SM) files.
- Added option in module `MultiFileVolumeListFilterOutputs` to search for an arbitrary tag value and to select the used boolean operations.
- Added support in `DirectDicomImport` for color enhanced multi-frame MR files.
- Added support in `DirectDicomImport` to sort according to binary tag contents (e.g. Palette LUTs).
- Fixed unnecessary rare warnings in DICOM overlay calculations (e.g. in `MultiFileVolumeListImageOutput` and `DirectDicomImport`).
- Fixed incorrect shifts in DICOM overlay calculations.
- Fixed incorrect buffer sizes written when binary images are written as DICOM SEG with `DicomSEGSave`.
- Fixed crash of default `MultiFileVolumeListFilterOutputs` after simple add/remove.

- PCL
  - Added module `PCLIntensityRankFilter` which performs a rank filtering on intensity replacements of each point of a point cloud.
  - Added module `PCLR2SonicWCLoader` which can load certain sonar data files.
  - Added module `PCLIterativeClosestPoint` that finds the closest match between points of two point clouds.
- Other
  - Added `SoText2D` node support in `CompareInventorSceneSampling`.

Windows Edition

- Long file path support has been established/restored for our MeVisPython interpreter.
- `ImageLoad` and `ImageSave` (or rather the library `MLImageIO`) can handle paths longer than `MAX_PATH` on Windows now. Note that a special registry setting is still required to make this work on the system (HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem\LongPathsEnabled).
- Command line arguments were not parsed with the correct character encoding.
- The `Dependencies` tool is supported as a replacement for `DependencyWalker` when checking DLL dependencies from a modules's context menu.
- Added the option `FullscreenFlickerFix`, which can be set in the preferences file, to work around a temporary black screen when showing fullscreen panels.

- The Visual Studio 2017 version of MeVisLab can also be used with Visual Studio 2019. If you change the environment variable `MLAB_COMPILER_VERSION` to VS2019-64 you also don't get annoying project conversion dialogs for your C++ module projects.

Linux Edition

- MeVisLab will use the correct available RAM in docker containers when deciding on default memory consumption.

macOS Edition

Note

This release requires at least macOS 10.13 (High Sierra) and Xcode 9.4 for development.

- macOS 10.14 (Mojave) is officially supported now (macOS >= 10.14.4 and Xcode 10.2 recommended)
- MeVisLab is now signed with an Apple Developer Certificate
- Added configurable & extensible Touch Bar support for MacBook Pro in the MeVisLab IDE (see MeVisLab 3.2 macOS Edition Guide) & MDL (see MeVisLab Definition Language Reference)
- Movie recording updates:
  - support `.mp4` in addition to `.mov` file type
  - added more user friendly video compression codec selection for the VideoWriter* modules
  - support HEVC (H265) codec
  - support professional codec types (Apple ProRes)
  - recognize codec type names „H264“ & „H265“ and use the corresponding values on macOS
  - use HQ settings for H264, HEVC & JPEG codecs
- Added search for all Network documents containing a specific Module from within MeVisLab itself (Activate the context menu item **Related Files** › **Show Network Files Using This Module** of a module in a Network Window or in the **Module Search View**) (see also MeVisLab 3.2 macOS Edition Guide)
- Previews of MeVisLab network files are now stored directly as extended attributes of the file. Previously, previews used the deprecated resource forks API and will no longer be displayed using QuickLook. Resave these files to update the preview.
- ProjectGenerator helper now generates Xcode projects using the legacy build system due to a Qt incompatibility
- ProjectGenerator helper now adds a Profiling setup for Apple Instruments to the default scheme for release builds of a module library (use #I to start profiling a module in MeVisLab from Xcode)
- **Supportive Programs** in **MeVisLab** › **Preferences** may be set empty to use the system default application for the file extension or URL scheme
- Fixed that the context help would not jump to the corresponding article (e.g. MDL controls help)
- MeVisLab Workers (aka Background instances) can be hidden from the Dock by enabling the option **Hide Worker instances from Dock** in the **General** section of MeVisLab Preferences Panel

- Setting the installer variable `INSTALLER_MACX_BUILDZIP` to 1 will generate a distribution PKZip file similar to a Finder archive
- Added Swift programming language support for Xcode & Makefiles in `.pro` files
- Added the `MetalKernel1` module that allows to script and apply a Metal kernel to an ML image

ADK (also known as Application Builder)

- The ApplicationBuilder add-on is called ADK add-on again.
- One can use the variables `UNIX`, `WIN32`, `MACOS`, `LINUX` in templates used for installers.

Version 3.1.1 Stable Release (2018-12)

New Features

- C++: Updated the `Eigen` library to version 3.3.5. This was necessary because newer updates of Visual Studio 2017 generate a compile error when including certain header files of this library.
- C++: Support CUDA 10 in project generation.
- C++: Added new template class `TypedBaseField` to ML library, which allows for convenient typed access of the contained Base object.
- C++: Added method `clearTriggers` to the `SoPointingAction` class.
- C++: Make libraries `MLGraphUtilities` and `SoCoordUtils1` usable from own projects.
- C++: Added macro `SO_NODE_ADD_ENUM_FIELD` as a shortcut for `SO_NODE_SET_SF_ENUM_TYPE` and `SO_NODE_ADD_FIELD` to Inventor.
- The `Field` control got a `browsingGroup` attribute, which allows to store last used directories for different purposes separately. There is also a new optional parameter of the same name for the methods of the `MLABFileDialog` scripting class.
- Added a scripting wrapper for some of the `OpenVDB` functionality. Using the wrapper can be more efficient than using the `OpenVDB` modules, since several level-set operations can be chained. See class `MLOpenVDBToolsWrapper` in the Scripting Reference for details.
- Added function `addQuadPatchFromNumPyArray` to WEM scripting wrapper, and fixed `addTrianglePatchFromNumPyArray`, which didn't associate the nodes to the faces.
- Also added function `addPatchCopy` to WEM scripting wrapper to copy a patch from another WEM.
- Module `DicomReceiver` got a new field `additionalStoreSCPOptions` to specify additional command line options.
- The minimum and maximum values of the `BoundingBox` module can be activated separately with checkboxes now.
- Added `isMouseOver` field to module `SoMenuItem`.
- Added `relativeAnchorPoint` field to module `SoFixedMenu` (used when not in pop-up mode to position menu items at a relative position of the viewing area).
- Added `performPreHitTest` field to module `So3DMarkerEditor`.

- The `rotation` field of module `WEMModify` is a real `MLABRotationField` now instead of just a `MLABVector4Field`, which means that you now can get/put an Inventor `SbRotationd` out of/into it in Python scripting.
- Improvements for module `CreateRTStruct`:
  - Added field `treatCollinearAsPlanar` which allows to change how collinear CSOs are handled.
  - Input image modality is not restricted to CT or MR any more.
  - An error is reported if the input CSOList contains ungrouped CSOs.
  - Enhanced printed information about problems with referencing slices.
  - Make sure that the `NumberOfContourPoints` is consistent with the number of values in `ContourData` when writing CSOs.
- Windows: The Mesa OpenGL fallback is always added to stand-alone applications if you don't set `WITH_MESA` to 0 in your `.mlinstall` file.

Fixes

- C++: Fixed new ML list field method `updateValueAt` not updating the value.
- C++: Fixed that some memory allocation errors might fail silently.
- C++: Backported a Qt patch for `QProcess::start`: Avoid a name collision in named pipes for processes started in parallel.
- `ComputedAttributes` in an `ItemModelView` control didn't update when one of the attributes in the expression changed.
- The `ColorEdit` control does not set the color to black anymore if the color dialog is cancelled.
- Removed the zoom restriction from the `Resample3D` module (factors above 1000 were set to 1).
- Modules based on `OpenVDB` crashed if used on WEMs with more than one patch.
- Fix handling of keyboard shortcut events in `RemotePanelRendering`.
- Fixed that `SoDiagram2D` used the color of the first style to draw the last curve when `stylePaletteMode` was `STYLE_PALETTE_NAME`.
- Fix crash under certain circumstances when removing the `SoView2DCSOEditor` module from a network.
- Fixed that the output of module `SoPostEffectBackground` in `OffscreenRenderer` turned all black when opening an internal network.
- Fixed menu item placement in `SoFixedMenu` and `SoBorderMenu`: Items attached to lower/left border jumped around when resizing.
- Improved blending of `SoView2D` font rendering with background.
- Fixed an infinite Inventor loop in module `View2DTouch`. Also fixed usage of unknown field `hiResRendering`.
- Ported some Python tutorial modules to Python3: `Tutorial_BasicPython`, `Tutorial_MeVisLabAndPython`, `Tutorial_AdvancedPython`.

- Ported module `CLIImporter` to Python 3.
- The `ChangeSet` Python class would cause an exception when the MeVisLab module on which the class was applied was already deleted when the destructor was called.
- Fixed some bad handling of OpenGL clip planes (which probably nobody really noticed).
- Number inputs in module GUIs didn't allow to type in (temporarily invalid) incomplete floating point numbers.
- MATE: Do not place cursor on previous line when de-indenting.
- MATE: Python refactoring with rope updated to new version 0.10.7 to fix that extracting a function failed.
- MATE: Disabled pylint checks were not persistent.
- MATE: Fixed that the positions of results of the **Find in Files** dialog were wrong if a result line was shown truncated.
- Tools like MATE don't write to the same log file as MeVisLab anymore.
- macOS: QuickLook and Spotlight plugins work again.
- macOS: Hide "Show Tab Bar" menu items added for 10.12 and up.
- macOS: Movie generation improvements (vertical flip of movies generated from image files, presentation time calculation for movies generated from memory images).
- macOS: Fix crash in IDE if size of recent files list is set to zero on system..
- Linux: Improved CUDA detection.
- Linux: Translation tool could not be opened directly from a multi-language module.
- Documentation fixes.

Version 3.1 (2018-06)

Python 3 Upgrade

We have upgraded our Python interpreter from Python 2.7 to Python 3.6 in this release. This requires our users to port their Python 2 scripts to Python 3 (if not already done).

To facilitate the porting, we provide the Python futurize library. (Note: This was removed again in MeVisLab 3.5).

On Windows, the Python 3 interpreter is now compatible with the official Python 3.6 interpreter. This allows to install binary packages from PyPi (the Python package index). A new `PythonPip` tool module facilitates the installation of Python packages from PyPi in MeVisLab.

Support for 10bit displays

MeVisLab now supports rendering with 10bit precision (using OpenGL). This requires a graphics card and monitor that support 10bit precision. The support has been tested on AMD/Barco graphics cards and on NVidia quadro graphics cards. We try to auto-detect the support for 10bit, but it can be enabled/disabled by the environment variable `MLAB_OPENGL_10BIT` as well. If enabled, MeVisLab automatically uses a 10bit capable OpenGL framebuffer (typically a RGB10A2 or RGBA16F buffer). By default, the `SoView2D` viewer will use a high-precision LUT if 10bit support is enabled. You can see if 10bit support

is enabled in the first line of the MeVisLab log (after the graphics card details). It can also be checked in Python using `MLABSystemInfo.glSupports10Bit()`.

Physically-based Path Tracing (CUDA only)

The MeVis Path Tracer offers a state-of-the-art Monte Carlo Path Tracing framework running on NVidia GPUs (CUDA). It supports high-quality physically based rendering of volumes, meshes, thick lines and other geometry. For details, have a look at the MeVis Path Tracer Overview.

OpenVDB integration (sparse grid support)

The OpenVDB library (www.openvdb.org) has been integrated into MeVisLab. OpenVDB supports sparse voxel grids and has a unique level set representation for meshes. It has been used to implement several new MeVisLab modules:

- `CSOToSurface` creates a WEM surface from a list of CSOs.
- `MarkersToSurface` creates a WEM surface from a marker list by rasterizing the markers as spheres.
- `OpenVDBLoad` loads a sparse grid as an ML image volume.
- `OpenVDBSave` saves an ML image volume as an OpenVDB sparse grid.
- `VoxelizeCSO`, `VoxelizeMarkers`, and `VoxelizeWEM` voxelize the respective structures into a reference image coordinate system.
- `WEMLevelSetBoolean` allows to perform boolean operations on closed WEM meshes using a level set representation.
- `WEMLevelSetFilter` allows to apply a kernel filter on a level set generated from a closed WEM mesh and remesh the result.
- `WEMLevelSetOffset` allows to enlarge or shrink a closed WEM using level sets.
- `WEMLevelSetRemesh` remeshes a closed WEM using a level set representation.

Fixes / Enhancements (MeVisLab)

- IDE
  - Added a context menu that allows editing of user scripts from the user script menu item itself.
  - Write the MeVisLab version to saved network files.
  - Support external links in note items.
  - 3D inspectors can zoom with the mouse wheel.
  - Prepend "./" to the displayed type of local macro modules.
  - The **Run In Separate Process** context menu entry shows a hint why it is disabled when it is disabled.
  - Connecting fields by dragging fields to the automatic panel was not undoable and also didn't mark the network as modified.
  - The MeVisLab Output Inspector now supports specialized inspectors for Python object sub-types.
  - Print an error message if a FieldListener has no field or no command given.

  - Fixed **CTRL**+**Tab** switching between networks in IDE and files in MATE.
  - Fixed inheriting module documentation if the module name is not the class name.
- TestCaseManager
  - Can avoid per-function result directory by setting `perFunctionResultDirectory` to `False` in the .def file of the test.
  - Added option to display test report viewer inside the main `TestCaseManager` panel.
- MATE (integrated editor)
  - Newly opened MATE documents are inserted after the tab from which they were opened.
  - MATE now downloads and installs Pylint version 1.8.4 by default.
  - Added option to run Pylint manually in MATE, and added a separate error check view.
  - One can specify a pylintrc file for the Pylint invocations in the preferences dialog.
  - Pylint check and Jedi auto-completion now regard the module's `importPath`.
  - Only highlight text or numbers on double-click.
  - Added (hidden) option to remove trailing spaces - this can be bound to a key shortcut in the shortcut manager.
  - Improved MATE startup performance by doing delayed package loading.
  - The replace-in-files dialog always writes system-specific line endings now.
- C++ API changes
  - Added new ML list fields: `ml::IntListField`, `ml::DoubleListField`, `ml::Vector2ListField`, `ml::Vector3ListField`, and `ml::Vector4ListField` (similar to the Inventor multi-fields).
  - The `mlError`, `mlWarning`, `mlInfo` macros do not need an error type argument anymore. If omitted, `ML_BAD_PARAMETER` is used.
  - `ml::Fields` don't have a copy constructor anymore.
  - Added new `getInterpolatedValue()` method on `ml::PagedImage` which allows to fetch a tri-linear interpolated value at an arbitrary location.
  - `ml::Rotation` now creates a 180° rotation for exactly opposite vectors (previously it did nothing).
  - The Inventor class `SoEvent` now has a `mouseButtonMask` property that contains the currently pressed mouse buttons.
  - Allow to override existing cursor shapes in Inventor viewers with `SoViewerProxy::defineCursor`.
  - Added a `BLANK_CURSOR` value for the cursor shape enum field in Inventor modules.
  - Renamed internal Inventor class `SoLightPath` to `SoLightweightPath` because the name was confusing.
  - The destructor of Inventor class `SbProjector` was made virtual.

  - Added a ML_EXPLICIT_FALLTHROUGH define to all platforms, which expands to `[[fallthrough]]` (C++17) on platforms where an implicit fallthrough would result in a warning at compile time.

  - Added new method ml::OpenGL::isMesaSoftwareRenderer() to check if MeVisLab runs with an OpenGL software renderer.

- Controls

  - Added a `PythonTextView` control for editing Python code in modules. It supports auto-completion of the underlying module context, context sensitive help and an extended context menu. Many modules that offer Python scripting now make use of this control (RunPythonScript, PythonArithmetic, ...).

  - `ItemModelView` can show a color and a checkbox in the same column.

  - Implemented an `alternatingRowColors` attribute on `ItemModelView`.

  - Added `alternateBase` (needed for `alternatingRowColors`), `toolTipBase`, and `toolTipText` color roles to MDL `style`.

  - Added `clickedColumnField` attribute to `ItemModelView` control, which writes the column index onto which the user (double-)clicked to the given field.

  - Introduced `ComputedAttribute` to `ItemModelView` control. This allows to compute a virtual attribute from other attributes using the same expression parser as, e.g., the `dependsOn` mechanism.

  - Implemented `dependsOn`/`visibleOn` on the `Row` element of the `Table` control.

  - Added new `listview` mode on `TabView` control. This facilitates the creation of preferences-style panel with a listview instead of a tabbar to select tabs. The listview can be made hierarchical by using the `tabHierarchy` tag on `TabViewItems`.

  - Fixed initial enabled state of `TabView` with only one child.

  - Fixed that `dependsOn` of `ButtonGroupControl` items always affected the first item.

  - When a parent checkbox in an `ItemModelView` control switches several child checkboxes, these are collected into one ItemModel update event.

  - Removed `slicerScale` attribute on `Slider` control (it didn't work anyway).

  - The **Space** key is not titled "Any" anymore in the `EventFilter` control. Also added the numerical "keyCode" info.

  - Call `shouldCloseCommand` if a modal dialog panel is closed with the **Escape** key.

  - Use system proxy settings by default in `WebView` control if no proxy is specified in the Preferences dialog of MeVisLab.

  - The `Field` control now recognizes the `comboItems` attribute.

  - Fixed wrong auto-alignment groups on the `Grid` control. Previously all Grid children got the same width, regardless of their column.

  - The **More** button on a `Field` control now shows a non-editable dialog if the control is non-editable.

  - Fixed problem with auto-repeat in step-buttons of number widget if a dialog is opened on value change.

- Scripting

  - New field wrapper classes for the ML list fields and some Inventor multi-fields: `MLABIntegerListField`, `MLABDoubleListField`, `MLABVector2ListField`, `MLABVector3ListField`, and `MLABVector4ListField`.

    Since previously these fields were wrapped with `MLABStringField` you will now get an info message if you use the old `value` property which has a string type. You should rather use the new `listValue` property or `stringValue()`/`setStringValue()`.

  - Added `mlab_projects` Python meta package for importing Python modules from MeVisLab module projects. (Not supported in auto-completion!)

    Use

    ```
    from mlab_projects.MySuperDuperProject import Utilities
    ```

    in Python code to import the Python module `Utilities` from file `$(MyPackage)/Projects/MySuperDuperProject/Modules/Scripts/python/Utilities.py`.

  - Added `MLAB.cryptStringUTF8()`/`MLAB.decryptStringUTF8()` scripting methods with UTF-8 support.

  - Added `setVr` method to `MLABMutableDicomTag`.

  - Python object `__object__` in the module context can now be any Python object to be visible through `ctx.object()`. Previously the auto-conversion of Python sequences or maps caused a conversion to a different object type.

  - The field associated with a control can now be accessed with the `field()` method.

  - `MLABProcess.runCommandInConsole` now has an option to wait until finished.

  - Added new scripting methods on macro modules to determine the sub-type (local, ad-hoc).

  - Added scripting method `ctx.shouldAvoidSideEffects()`, which returns true if a module is created just for editor auto-completion.

  - `MLABModule::addModule` does not print package dependency warnings anymore.

  - Print an error message if a call on a module could not be performed because there was no script context.

  - Using the `MLABHttpDownload` scripting class failed for large downloads in the GB range.

  - Added new scripting method `MLABFileManager.setApplicationSupportUmbrellaDirectoryName()` so applications can select another name than 'MeVis'.

- Other fixes and improvements

  - Improved CUDA building support, support CUDA 9.

  - Connecting a bool field to float/double fields correctly sets 0 or 1 (only worked for Inventor fields before).

    - Do not ignore legacy values in lazy loading modules.
    - Save persistent flag of fields if required in lazy loading modules.
    - Worker processes (of remote modules) ignore the `Logfile` variable now.
- Third-party libraries
    - Python
        - MeVisLab's Python distribution was updated to Python 3.6.4.
        - Added pyodbc Python ODBC bridge, version 4.0.21 (Windows only, typically used with sqlalchemy).
        - setuptools was updated to version 38.5.2.
        - NumPy was updated to version 1.14.2.
        - pip was updated to version 9.0.1.
        - coverage was updated to version 4.5.1.
    - Updated Qt to version 5.6.3.
    - Removed ospray library from ThirdParty, but re-added current embree library, version 2.17.0.
    - boost.random can be used in own projects now.
    - Use other method to determine MAC address for UUID generation in dcmtk to avoid certain crashes.
    - The storescu tool supplied by dcmtk library now supports on-the-fly compression.
    - The OpenSSL library was updated to version 1.0.2o.
    - Added OpenMP runtime for MacOS, version 6.0.

Fixes / Enhancements (Modules)

- SoView2D and extensions
    - Added module `SoView2DCine` that allows more control over `SoView2D`'s cine mode.
    - The new `SoView2DCurrentState` module returns the visible portion of an image for use with `SoView2DRectangle`.
    - More control over ruler color in `SoView2DRuler` and `SoView2DAnnotation`.
    - Added a `stepSize` field to `SoView2DSlicer`.
    - `SoView2DOverlayMPR` supports left-handed coordinate systems now.
    - `SoView2DDrawVoxels3D` can draw a brush preview now.
    - The `SoView2DMarkerEditor` module was moved to its own library.
    - `SoView2D` key commands are always available when the mouse is over the viewing area, not just over the image area.

  - `SoView2DExtensions` also show the "Alt" modifier option in their panel.

- Other Open Inventor modules
  - Added new modules for menus in OpenInventor scenes: `SoFixedMenu`, `SoBorderMenu`, and `SoPopupMenu`, with `SoMenuItem`. In contrast to `SoView2DMenu` these support Managed Interaction and are more flexible.
  - Added new modules to query the OpenGL state: `SoGLClearError`, `SoGLColorDepthInfo`, `SoGLGet`, `SoGLStateInfo`.
  - Introduction of the new Managed Interaction Inventor draggers. Have a look at the `MIPlaneDragger` macro module for how to use them.
  - Added `SoMetaInformation1d` and `SoMetaInformation1i` modules.
  - Added a `selectOnlyOnClick` option to `SoSelection` and `SoSelection2`, which doesn't perform the selection if the mouse was moved while holding the mouse button.
  - Added a `keepViewportWhileDragging` option to `SoViewportRegion`.
  - Improved compatibility of Managed Interaction modules with classic event handling modules in Inventor scenes.
  - Added support for sRGB interpolation in `SoLUTEditor` and LUT base classes.
  - Allow to add control points by dragging from the borders in the `SoLUTEditor`.
  - Allow to choose slab start/center/end for the intersection (and an optional offset) in `SoRenderSurfaceIntersection`.
  - `SoCrosshair` can render thin cylinders instead of lines now.
  - Added `ADD_BEFORE_INORDER` modification type to `SoShaderPipelineFunction`.
  - Various changes to `SoVascularSystem`, e.g.:
    - Can work on a copy of the input graph.
    - Enhanced highlighting capabilities.
    - Can choose between integer and float LUT evaluation.
  - Support local transformations in `SoDrawInstanced`.
- CSO
  - Added functions to CSO scripting wrappers for seed point selection.
  - `CSOConvertToImage` got an `outputTypeMode` field to grant more control over the voxel type of the output image.
  - `SoCSO3DRenderer`, `So3DMarkerRenderer`: Added functionality to only render CSOs/markers of selected timepoint.
- WEM
  - Fixed an endless loop in `WEMSubdivide`.

  - `SoWEMRenderer` can cache WEM patches now (e.g. to quickly switch between time points).
  - Added module `WEMRayIntersect` and scripting wrapper `WEMBoundingVolumeHierarchy` for a fast ray intersection with WEM surfaces (based on the embree library).
  - `WEMIsoSurface` will cope with images with left-handed coordinate system now.
  - The `WEMBulgeEditor` uses different values (signs) in the PVL for preview and interaction.
  - Added global normal output to `MarkerListToWEMPlane` module.
- ML modules
  - Added modules `QPainterImage` and `QImageToML` for ML image generation through Qt classes.
  - Added the `ChromaKey` module (for green screen handling of video images).
  - Replaced `Type(De)Composer8|16|32|64` with `Type(De)ComposerN`, which have a dynamic number of visible inputs/outputs.
  - Added `fillValue` field to `ConcatImages`.
  - The `BaseClear` module got `inputValid`/`outputValid` fields.
  - Added an option `useKeyFrameDistanceForSliceThickness` to the `MPRPath` module.
  - Fixed automatic voxel size and bounding box calculation of the `MPR` modules.
  - Added `Minimum` and `Replace` modes to the `AccumulateImage` module.
  - The new `GraphToVolume` module allows to rasterize a `ml::Graph` to a volume.
  - `ImageSave` can now save RGB floating-point TIFF images with a linear (gamma=1) sRGB ICC profile.
  - `ImageCompare` now prints an `ML_CALCULATION_ERROR` instead of an `ML_PROGRAMMING_ERROR` if images differ.
  - We did a clean-up of `MLVesselGraph` library, some old modules were removed.
  - Avoid crashes in GVR modules when the image min/max values are not correct, i.e., there are voxel values outside the stated range.
  - Make `CurveToHistogram` always correctly update its output.
  - Fixed auto-updating of maximum volume limit in `RegionGrowing` module.
  - Fixed property extension loading in `MLImageFormat`.
- Macro modules
  - Created new utility module `ExportImagesAsSlices`.
  - A new `PythonPip` tool module allows to install Python packages from PyPi.
  - Lots of fixes and improvements to the `AnimationRecorder`.
  - Added undo/redo to `DrawVoxels3D`.

- `GVROrthoOverlay` has a `blendMode` field now.
- Speed-up loading of file list in `ImageLoadMulti`.
- Renamed `CommandNotifyer` to `CommandNotifier` and improved panel.
- Removed the outdated `ImageViewer` application from the SDK.

Modules and Libraries Provided by Fraunhofer MEVIS

Modules

- Additions
  - New module `WalkSiblingDirectories` that can be used to compare the contents of two folders file by file.
  - New module `SoMIMouseOver` allows to detect mouse over and click on child geometry using Managed Interaction.
  - New module `SoMIRotateCameraPlaneDragger` allows for rotation in the camera plane using Managed Interaction.
  - New module `SoMITranslateCameraPlaneDragger` allows for 2D translation in the camera plane using Managed Interaction.
  - New module `SoView2DInfo` that displays SoView2DExtension infos.
  - New module `DynamicLayout` that creates a layout using an MDL file and embeds viewers found in the connected Inventor graph.
  - New module `SoActionFilter` that allows to select which actions are applied to the child graph.
  - New module `InventorGuard` that combines `SoIdentifier` and `SoActionFilter` such that the child graph can not be added directly to the scene.
  - New module `SoNodeFilter` that filters the Inventor scene graph based on node names or types.
  - New module `ComposeBoundingBox` that calculates a composed bounding box of two aligned images.
  - New module `ApplyOrthoOrientation`, extending `NormalizeOrthoOrientation` with an additional "FromReference" mode, which applies the same ortho-orientation (transversal, sagittal, or coronal) as a given reference image.
  - New module `OrthoCombination`, does a combination of three images in different orthogonal orientations into a single one with a defined orientation, which puts the images from the different inputs in separate sub images in the c, t, or u dimension. This allows for convenient combination using modules requiring a single input image (e.g. `OrthoProjection` or kernel filtering).
  - New module `ImageFromFile` that loads a single image from a directory or file of (almost) any image file type supported by MeVisLab (basically an extension to `LoadAny`, internally making use of it if everything else fails).
  - New module `ApplyGlobalModalityLUT`, allowing for normalizing DICOM data containing either a single frame-unspecific (i.e. "shared" or "global") or multiple frame-specific RescaleIntercept or

RescaleSlope values to a single, frame-unspecific, user-defined intercept, slope, and data type. (combining ApplyDicomPixelModifiers and DicomRescale).

- New module `BoundingBoxInReferenceSystem` that computes the smallest SubImage of input1 that would contain all (reformated) voxels of image0..
- New module `LUTColorAtIndex` that gets the color at the given LUT index.
- New module `LUTColorAtValue` that gets the color at the given LUT value.
- New module `SoView2DSetPosition` allows to set the slice index of connected SoView2Ds such that a given position is matched best.
- New module `BoundingBoxListener` notifies changes of the Inventor scene boundingbox. Optionally an object selection is used.
- New module `RotateAtTarget` allows to compute the orientation for a given position, target point and up-vector.
- New module `ImageAlignedBoundingBox` calculates the image-aligned bounding box of the input image in world coordinates.
- New module `WorldAlignedBoundingBox` calculates the world-aligned bounding box of the input image in world coordinates.
- New module `ApplyTransformationMatrix` allows to apply a transformation given as matrix on the input image.
- New module `GVRContourOverlay` that draws a contour of a mask image using a GVR secondary volume.
- New modules `PointSender`/`PointReceiver` that allow to send point lists over a communication channel.
- New module `FilterCurveList` to filter a CurveList by curve properties such as title or style.
- New module `SetMarkerCTUCoordinates` to set non-spatial coordinates for a whole XMarkerList.
- New module `TranslateXMarkers` to translate all positions in an XMarkerList.
- New module `ExtrapolateXMarkersPolynomial` to extrapolate an XMarkerList based on a polynomial fit.

- Improvements & extensions
  - Added simpler, bulge-like node movement to `WEMExpandToMarkers`.
  - The macro module `CSOImageStatisticsOverTime` now allows more than one input CSO.
  - Renamed `NormalizeOrientation` to `NormalizeOrthoOrientation`.
  - `ImageCache` macro and CacheManager python module now support progress output.

Updated third-party libraries in `FMEWork/ThirdParty`

- Updated CTK-CLI python package to version 1.4.
- Updated pydicom python package to version 1.0.1.

- Removed mock python package.

Windows Edition

- The Mesa OpenGL fallback driver has been updated to version 17.5.3.
- HiDPI support has been improved, especially for setups with multiple screens of different resolutions.
- Fixed a small color conversion bug in the video generation with OpenCV under Windows - videos were slightly too dark.

macOS Edition

Note

This release requires at least OS X 10.11.5 (El Capitan) and Xcode 8.2 for development.

- Full Support of macOS 10.13 (High Sierra)
- Worker and Background Processes started from the IDE will show up in the Dock and allow network inspection via Context Menu Entry 'Show IDE'
- Support of OpenMP development with Xcode 9.0 and newer (OpenMP runtime is included with MeVisLab)
- Symbol visibility is restricted to explicitly exported symbols
- ADK
  - Support of INSTALLER_MACX_PREINSTALL_COMMAND & INSTALLER_MACX_POSTINSTALL_COMMAND for executing shell commands during the installation of pkg installers (aka guided installers)
  - Support of MACX_SIGN_INSTALLER, MACX_SIGN_CHECK_IDENTITY_BEFORE_USE, MACX_INSTALLER_SIGN_IDENTITY_NAME to sign pkg installers

Application Builder (previously known as ADK)

- Not showing diagnostic info messages for MeVisLab console applications.
- Do not display certain dialogs if MeVisLab is run in batch mode.
- When setting the preferences variable `ShowFieldHelpInApplications` the field help is shown in applications.
- Can specify installer icon under Windows with `INSTALLER_WINDOWS_ICON` (and added a new default installer icon).
- Removed support for putting .py and MDL files into a zip file in standalone installers.
- Overhaul of the multi-language support in MeVisLab:
  - Calling the language tools from the public SDK works now.
  - Can open Qt Linguist from the context menu.
  - Can set the language from the module's context menu.

  - Better detection of translatable strings.
  - Translations will be applied to web panels.
- Make it possible to use the Unicode plugin in the NSIS installer with `INSTALLER_CUSTOM_NSIS_TOP_INCLUDE`.

Version 3.0.2 Stable Release (2018-01)

New Features

- `MLABModule::addRemoteModule`: Connection can be SSL encrypted.
- Added `isMouseOver` and `isDragging` fields to `SoView2DSlider`.
- Set `isFilePath` attribute on icon path fields of `SoView2DMenuItem`.
- Can link against the SoRenderers and SoWEMModules libraries.
- Use the http proxy configured on the system by default.
- Automatically convert Python unicode strings to QByteArray in object wrappers using UTF-8 encoding.
- `SoCSO3DRenderer` can apply a LUT to the CSO rendering now if a CSO module sets the extra value per path point.

Fixes

- `SoOrthoView2DBase` did not update its fields `sliceOrigin2` and `sliceOrigin3` correctly.
- Setting list values from NumPy lists in Python wrappers could crash under certain circumstances.
- Fixed MDL tag value auto completion in MATE.
- Fixed crash when using `titleField` of box control.
- Fixed incorrect parsing of `DirectDicomImport` config on MacOS.
- Fixed an endless recursion in Python profiling when recursive object references were encountered.
- Make `MPR` module work with left-handed coordinate systems.
- Fixed a crash in the `DynaCurve` module when the `timeData` enum is changed.
- Do never replace content of Debug Output when clicking links, always open in separate browser.
- Support http and other Sphinx default roles in text of note items.
- The editing dialog for note items didn't always show.
- Fixed a restart issue with the MacOS Dock after adding items.
- Fixed undoing of CSO changes not triggering the needed notifications.
- Fix a bug in the item parser of `SoView2DLegend`.
- Fixed light state leakage in `SoPostEffectRenderer`.

- Fixed `SoDepthPeelRenderer` rendering on Intel graphics by adding a depth bias.
- Fixed crash in `CSOConvertToImage` when changing number of time points.
- `CSOIsoGenerator` clears the CSOList if the `updateMode` is set to `AUTO_CLEAR`.
- Fixed `SoView2DOverlayMPR` showing the wrong overlay when zooming in.
- Module `SoShadowMapping` didn't work under `SoGroup` nodes.
- Correctly resize PVLs of WEMs in `WEMThreshold`.
- Changes to the `upVector` of `MPR` module didn't trigger recomputation.
- Fixed endoscopic rendering in GVR raycaster not clipping octree nodes correctly.
- Fixed crash on Intel graphics when using shaders without own uniforms.
- Avoid crash in VTKCowHairExample from welcome screen (only crashed on Nvidia graphics cards).
- Fixed wrong default memory cache size in IDE when main memory size is 16GB or more (on 64 bit systems).
- Fixes to Python futurize.
- Fixed crash when listening to Shortcut events (which isn't a key event) in the EventFilter control. Also added some more key code names for key events.
- Fixed cursor shapes not updating for Inventor viewers when rendered with a `RemotePanelRenderer`.
- ADK: Also sign files in TestCases directory of projects in the Projects directory of a package.

Version 3.0.1 Stable Release (2017-08)

New Features

- Added support for Visual Studio 2017. For this compiler we are abandoning our previous naming scheme (Visual Studio 2015 had the id VC14), and name it VS2017 instead.
- Added `KeepDimension` option to slab render modes of modules `MPR` and `MPRLight` to allow separate depth or time projection.
- `WEMSubdivide` got support for localized subdivision using a PVL list.
- The module `SoClearBuffers` got a dedicated `depth` value to set the depth buffer to an explicit value.
- The module `SoView2DAnnotation` got margin fields.
- Restored original behavior of `MLABMacroModule::addNote()` and `MLABNoteItem::setText()` to add plain text notes (which is faster) and added new methods `MLABMacroModule::addReSTNote()` and `MLABNoteItem::setReSTText()` to programmatically create and edit ReST notes.
- `MLABPackage::isUnderVersionControl()` also detects git repositories now.

Fixes

- Fixed a crash in `SoWEMRenderer` on deconstruction (when an input was still connected).
- Fixed a crash in `CSOVoxelCrop` when no input was connected.

- Avoid crash in `WEMImageData` when filling a custom PVL but not specifying a name.
- Fixed value "bleeding" error in `FloydSteinberg` module.
- Fixed `dependsOn` mechanism on items of the `ToolButtonGroup` control.
- Fixed `SoView2DOverlayMPR` sometimes rendering the wrong overlay when using a lot of overlays.
- Fixed `WEMSubdivide` sometimes not terminating when applied on non-closed WEMs.
- Fixed `MoviePlayer` control not working on Linux because of a missing library.
- Fixed `MLWEMWrapper::getWEMPatchIndex(int id)` which returned the total number of patches instead of the index of the patch.
- Can disable Pylint checks also for the Py3k pass in MATE.
- Fixed Python auto-completion of Qt properties in MATE.
- Fixed jumping to error messages in the ToolRunner.
- Fixed some wrong cursor position problems on the network area in HiDPI mode.
- Fixed the "Save to Image" button of the `MatplotlibCanvas` control.
- Make the `ML_ENUM_VALUE` macro also work with C++-11 strongly typed enums.
- Speed-up of add-on installation on Windows.
- Some small module help updates.

Version 3.0 (2017-06)

The main reason for the change in the major version number is that MeVisLab is now based on Qt5.6. This change also caused some other changes, so you should read the release notes carefully.

Qt5 Porting Issues

- The `clickedCommand` and `mouseButtonClickedCommand` of the ListView control only react on left-click now. If you want to react on right-clicks, use the `pressedCommand` or an EventFilter.
- We switched to a new OpenGL widget in Qt. In case of problems you can get back the old widget by setting the environment variable `MLAB_QT_OPENGL_WIDGET` to `old`.
- The (legacy) support for JavaScript scripting in modules has been removed entirely.
- ActiveX support has finally been dropped entirely.

Python 3 Switch Preparations

We are planning to switch from Python 2 to Python 3 with one of the next releases. To ease the switch we made it possible to make the Python scripting code compatible with both versions.

For this, we introduced a **Python 3 Compatibility Warnings** option in the **Preferences...** dialog under the **Scripting** section. This will emit warnings about non-compatible code when the code is executed. This will also enable warnings about non-compatible code in MATE if Pylint is installed.

There is also a User Script under **Scripting** → **Python 3 Support** → **Futurize Python Scripts** that will automatically convert your code to be compatible with Python 3. It is advisable to check the changes

though, since they sometimes tend to be in the category 'works, but is not pretty', and can be done better by a knowledgeable user.

PythonQt

- The PythonQt wrappers for Qt have been upgraded to Qt5. PythonQt for Qt5 still follows the package naming of Qt4, so, e.g., `QWidget` is still in `QtGui` (instead of `QtWidgets`) and `QWebView` is still in `QtWebKit` (instead of `QtWebKitWidgets`) and so on.
- Changed methods (**old** → **new**):

```
QtCore.QDir.convertSeparators -> QtCore.QDir.toNativeSeparators
QtCore.QRegExp.numCaptures -> QtCore.QRegExp.captureCount
QtCore.Qt.escape -> MLAB.escapeString
QtCore.QTextStream.writeBoolean() -> removed
```

  `QtCore.QUrl` has been redesigned in Qt5 and the API has changed substantially. The new class `QUrlQuery` offers some methods that have been removed from `QUrl`.

- PythonQt has the following new modules: `QtMultimedia`, `QtQuick`, and `QtQml`.
- PythonQt now supports `QtCore.Slot`, `QtCore.Signal`, `QtCore.Property` (compatible to PySide/PyQt), and the creation of dynamic meta objects.
- `QSound` has been moved to `QtMultimedia`.
- Improved `__nonzero__` check in some PythonQt wrappers.
- Added support for Python compare functions (`__eq__`, etc.) in QObject wrappers.

Fixes / Enhancements (MeVisLab)

- IDE:
  - The context menus of fields and modules have been re-worked to reduce the number of items. It may take some time to get used to the new arrangement.
  - Added a `projectsSearchDepth` tag to the Package.def file of each package to set the maximum search depth for looking for projects in the Projects directory.
  - The number of warnings and errors in the console is now displayed in the status bar of the IDE.
  - MeVisLab now automatically handles UTF-8 files with BOM correctly.
  - Added NetworkPanel step buttons to modules that switch their inputs (previously we only allowed one button per module, now we can have a forward and a back button).
  - MeVisLab now checks if an MDL tag may be specified multiple times.
  - Unknown field types in the Interface section of a module give a warning now.
  - Cannot specify Image, MLBase, or SoNode field types in the Parameters section of a module anymore.
  - The image cache can now be cleared from a context menu on the cache size display.
  - Removed option to enable the classic ML host from preferences.
  - Improved network notes, they now use Sphinx markup and can link to objects in the network.

  - Newly created note items are also placed at the mouse cursor if this option is enabled for modules.
  - Added a `isFilePath` attribute for string fields in the Interface/Description section of modules. This activates a special handling in Field controls and in the automatic panel, and tries to convert the path value with `MLAB.unexpandFilename` when saved in a network file.
- TestCaseManager:
  - Code tests can now be run (and compiled!) from the TestCaseManager.
  - The test case methods `Base.verifyTrue` and `Base.verifyFalse` require the argument to be really `True` or `False` now.
  - The TestCaseManager now shows the comments defined on a test case as tool-tip.
  - Show micro second precision in logging time stamps.
  - Support multi-selection of test cases.
  - Test cases can now be organized in "sections" (per package).
  - One can now select which packages should be visible in the TestCaseManager.
  - Expected errors and warnings in a test are not marked as error in the MeVisLab console anymore.
  - `expectError`/`expectInfo`/`expectWarning` should behave correctly now when used recursively.
- MATE (integrated editor)
  - Added highlighting and auto-completion for .mli/.mlinstall files.
  - The integrated Python debugger can now break on Python warnings. This can be used to prepare the switch to Python 3 (when Python 3 compatibility warnings are enabled).
  - Python auto-completion runs in a separate process now which makes typing a much smoother experience.
  - MATE will save files automatically as UTF-8 (marked with a BOM at the start) if required by their content.
  - Added **Extract Method** and **Rename** refactoring functions for Python code.
  - Can change the font size in MATE with **Ctrl**+mouse wheel.
  - Pylint is installed with a fixed version to hopefully fix installation problems in the future.
- C++ API changes
  - Managed Interaction now allows to dynamically remove (or add) interactions from/to a node.
  - Added a `SO_NODE_ADD_FIELD_CALLBACK` macro to Inventor for easily calling a class method when an Inventor field changes.
  - Enabled C++-11 support.
  - `ml::SubImage::setOrigin` keeps the original extent now.
  - Added templated `getTypedValue()` method to classes `ml::BaseField` and `SoSFMLBase`.

- Added a `addBase` overload to `ml::FieldContainer` that takes a `RefCountedBasePtr` as argument. Also added `addBaseWithAllowedType` methods.
- Added support for replacing the page on the `outputSubImage` in `calculateOutputSubImage`.
- `getTile` calls can now take an optional progress update callback as argument.

- Controls
  - The EventFilter control gives more complete information about ContextMenu events.
  - Added a `contextMenuRequestedCommand` to the ListView control.
  - The MoviePlayer control has been re-implemented to use the multimedia framework of Qt.
  - Added scripting method `setItemToolTip` to IconView control.
  - Added `automaticParentCheckboxState` feature to the ItemModelView control which sets the checkbox state of parent items according to the state of their children.
  - Added `autoExpandToDepth` attribute to the ItemModelView control.
  - Added a new `GUIExample` module, removed the old `ExampleGUIScripting`.
  - One has better control over icon sizes of button group controls with the new attributes `iconWidth`, `iconHeight`, and `useOriginalIconSizes`.

- DICOM support
  - We can now decode RLE-lossless and JPEG-lossless image data in DICOM files.
  - Improved performance when reading DICOM files.
  - Dicom UID generation now uses a new prefix.
  - Added support for OL and UC VRs in DcmTree library.
  - Added new `ml::DicomTree` and `ml::MutableDicomTree` Base objects to MLDicomTreeImagePropertyExtension library, also added and improved wrappers for these objects.
  - Added modules `CloneDicomTree`, `DicomTreeToDummyImage`, `GetDicomTreeFromImage`, `LoadDicomTree`, `SaveDicomTree`, and `SetDicomTreeOnImage` which use the new DICOM Base objects.
  - Added a `DicomTagNames` Python module with all known DICOM tag names as members.

- Scripting
  - `MLAB.terminate` now has an optional exit code argument.
  - The preferences dialog gained an option to enable Python 3 compatibility warnings (under Scripting).
  - Added the new `MLABDicom` global scripting object and moved DICOM related methods to it, keeping the old methods in `MLAB` as forwards, added new `tagInfo()` method.
  - Added a warning if a control with the same name was already registered in an MDL panel.

- Added a no-op scripting method `MLAB.moduleLiteral()` to mark module names for dependency analysis.
- Offering Principal Component Analysis via scripting on CSO, WEM, and XMarkers.
- Added `MLAB.macOpenFileInApplication()` scripting function for macOS.
- Handle `SystemExit` Python exception correctly.
- Can copy&paste multi-line Python code into the scripting console and execute it normally now.
- Many basic script wrappers can now return (or take) a NumPy array to improve performance.
- Added scripting methods `desaturateImage`, `grayScaleImage`, and `blendToColor` to `MLABGraphic`.
- Added a method to the IconView control to change the `QImage` of an item.
- Added Inventor scripting wrapper methods to get a full path from a `SoPath`, and a quaternion from an `SbRotation`.

- Other fixes and improvements
  - We added support for a "CUDA" CONFIG option in .pro files (you still need to install CUDA, though).
- Third-party libraries
  - Updated Qt to version 5.6.2.
  - Updated dcmtk to version dcmtk-3.6.1_20160630.
  - Updated boost to version 1.62.
  - Added the boost_timer library to the possible CONFIG entries.
  - Updated Threading Building Blocks to version 2017 Update 3.
  - Updated OpenSSL to version 1.0.2k.
  - Updated zlib to version 1.2.11.
  - Updated libpng to version 1.4.20.
  - Updated libjpeg-turbo to version 1.4.90.
  - Added the icu unicode library.
  - Updated Crypto++ to version 5.6.5 and moved the include files to a sub-directory "crypto" (so you need to adapt your includes if you used this library).
  - Python
    - Updated NumPy to version 1.9.2.
    - Updated Jinja to version 2.7.3.
    - Updated Pygments to version 2.0.2.
    - Updated docutils to version 0.12.

- Updated Sphinx to version 1.3.1.
- Updated Mako to version 1.0.1.
- Updated pip to version 8.1.2.
- Added pyzmq.

Fixes / Enhancements (Modules)

- SoView2D and extensions
  - `SoView2D` got support for Inventor picking when rendered in 3D.
  - Added a `forceImageUpdate` trigger field to `SoView2D` which can be used to force an immediate update to fields depending on the input image.
  - Added the module `SoView2DArrowHeadSettings`, used in `SoView2DMarkerEditor` and `SoCSOArrowEditor`.
  - Added new module `SoView2DButton`.
  - Added depth visibility feature to `SoView2DLabel`.
  - Added new module `SoView2DGrid` which draws a grid on the view.
  - Added new module `SoView2DCurrentState` to get at pixel spacing and viewing area of a `SoView2D`.
  - Added new module `View2DIsoContourShader`.
  - Added option `useInputResolutionIfAxisAligned` to the `SoView2DOverlayMPR` module.
- Other Open Inventor modules
  - Added the `SoClearBuffers` module.
  - Created a new module `InteractiveRampLUT` that is used in `View2DExtensions`, `View3D`, and `MLImageInspector`.
  - Switched the zoom action of `SoCameraInteraction` to be a managed offset action.
  - Added `SoUndoActions` module which provides a Managed Interaction frontend for the `UndoManager` object.
  - Can zoom (and rotate) with mouse wheel in `SoExaminerViewer` if activated.
  - Changed headlight handling in `SoExaminerViewer` and `SoCameraInteraction`.
  - Added SeekToPoint pointing action to `SoCameraInteraction` module (must be bound to mouse button with `SoInteractionMapping`).
  - The `SoPlaneDragger` module got new fields `enableMouseOverHighlighting` and `enableMouseOverCursor`.
  - Fixed `SoVolumeCutting` on Linux.
- CSO

- The `SoCSOEllipseEditor` can create circles on single click now and resize circles/ellipses around their center.
- New a facility in CSO editor modules to control path point density.
- Added group merge mode to `CSOMerge`.
- Added new module `SoCSODrawOnSurface`.
- Make `SoCSOImageStatistics` honor the time point of the CSO.
- Optimized `CSOLiveWireGraph`.
- Added `pointsPerMM` argument to CSOList scripting wrappers.

- WEM
  - Added a Octasphere shape (sphere constructed from octants) and a Dodecahedron mode to `WEMInitialize`.
  - The first input of `WEMMerge` is now called `inWEM0`.
  - Added `normalMode` field to `WEMSaveAsU3D` to support per-node and per-face normals.
  - Added an option to `WEMModify` to sort the faces of all WEMPatches in the z-direction.
  - `WEMReducePolygons` can now also stop on a maximum error value for edges.
  - Prevent smoothing and polygon reduction of the result of `WEMVascularSystem` at the leaf points by new option.
  - Added `WEMBulgeEditor` modules for 2D and 3D editing.
  - `WEMThreshold` will now intersect faces cleanly.
  - Added a `WEMThresholdToCSO` module which allows to generate CSOs at the interval borders of a PVL value set on the input WEM.
  - `WEMClipPlaneToWEM` can return the clipped remainder and clipping cover in separate patches.
- ML modules
  - The `VideoWriter` modules can now also write separate image files.
  - The `RemotePanelRendering` module can now handle nested GraphicsViews (this needs to be activated with a flag).
  - Added an option to `ComputeConnectedComponents` to set image value or rank as default user data.
  - Switched the `StopWatch` module from using MemCache to ProcessAllPages.
  - Added a `SetFileName` module to override the filename associated with an image.
  - `RemoteRendering` supports HiDPI screens now.
  - Added support for tag name completion to `DicomTagModify`.
  - Added C, T, and U ramp to `TestPattern` module.

  - Added field allowInvalidInputs to `MergeRegions`.
  - Added `MarkerPCA` module which does a principal component analysis on an XMarkerList.
  - Speed up compare of DICOM trees in `ImageCompare module`.
  - Added new module `ComputeMajorAxis` to compute extent of mask items in 2D.
  - Added new module `TransformWorldMatrix`.
  - Added a recursive filter mode to the `ItemModelItemFilter`.
  - `SubImageStatistics` can now compute statistics for multiple mask regions.
  - Improved and renamed `EuclideanDTF` to `EuclideanDistanceTransform`.
  - Added new module `SignedEuclideanDistanceTransform`, which does a distance transformation to the inside and the outside of a mask image.
  - Added module `ChangePageExtent` which allows to change the page extent of an image per dimension.
  - Added `ignoreOrigin` option to `PythonImage` module and `PySubImage` class.
  - Fixed an offset miscalculation in `OrthoReformat3` leading to crashes for large images.
- Macro modules
  - The `DicomTagBrowser` module can now sort by column.
  - Added a 'finalize' section to the `RunPythonScript` module.
  - Added a `onlySetIfChanged` flag to the `SettingsManager` module.

Modules and Libraries Provided by Fraunhofer MEVIS

Miscellaneous

- Additions
  - New module `ImageCache` that wraps different kinds of caches and provides additional auto update modes (e.g., 'auto-update only if valid and different') that can be helpful during prototyping, and a `CacheManager` Python module that can be used to dynamically add caches to a network.
  - New module `ImageCacheForNetworkLoop` that supports iterative updates in cyclic subnetworks.
  - New modules `CSOImageStatistics` and `CSOImageStatisticsOverTime` that calculate statistics of CSOs with respect to a given image.
  - New module `CSOVoxelCrop` and `CSOWorldCrop` that take a CSOList and a bounding box (in voxel or world coordinates, respectively) and removes all points from the CSO that are outside the box.
  - New module `CSOOffset` that can shrink or extend CSOs anisotropically.
  - New module `CSOResample` to resample seed points in a CSO to a given distance.
  - New module `CSODifference` that computes the difference between two CSOs as open CSOs.

- New module `ModifyCSOVisualizationSettings` that modifies certain properties of a CSOVisualizationSettings object.
- New module `CSOOrthoEditor` for to allow rendering and editing CSOs on an `OrthoView2D`.
- New module `WEMClipImagePlanesToCSOList` that converts a WEM to a CSOList with CSOs on the planes of an image.
- New module `SetMarkerProperties` to set common properties for all markers in a marker list.
- New module `ExtrapolateXMarkerLine` that extends a dotted line of XMarkers linearly in both directions.
- New functionality and modules in and general overhaul of the test hierarchy analysis for complex dependencies, i.e., the `applicationTesting` Python package and the `TestHierarchy` project.
- New modules for ListBase-derived list (e.g., XMarkerLists) handling:
  - `SplitList`: Splits a list in two.
  - `Sublist`: Extracts a part of a list .
  - `ListActionInspector`: Shows info about the current action on the list.
- Other new modules:
  - `DetermineTestsOfModule`
  - `CreateModuleList`
  - `CreateTestCenterTestCaseList`
  - `FilterModuleList`
  - `ApplySetOperationToModuleList`
  - `LoadStructuredListFromJSON`
  - `SaveStructuredListToJSON`
  - `FindPythonCallsInDependencies`
- New Python classes `TestCase`, `ModuleTestCase`, and `AlgorithmModuleTestCase` supporting TestCenter tests that come in class form and support cleaner test setups (in fmeTestSupport).
- New user script `ShowExampleNetworkForSelectedModules`.
- Added Python wrappers for most standard `*Load`, `*Save` and `*Compare` modules.

- Improvements & extensions
  - Several improvements to the `AlgorithmModule` and `AlgorithmMacroModule` base classes and the corresponding wizards.
  - The behavior of `CSOListToStylePalette` needed to be changed in order to fix several bugs in palette generation. The output StylePalette is now resized correctly so it can be empty or it can have less than 12 styles. In addition, palette colors and names are now cleared correctly before

each update which may result in different outcome in existing networks. CSO labels are now also set as style names in "CSO Id" and "Group Id" modes.

- `CSOShapeBasedInterpolation` can now compute the distance transform based on the CSOs and thus achieve sub-voxel accuracy.
- `CSOShapeBasedInterpolation` and `ImageShapeBasedInterpolation` now have an additional distance map output and a "use distance to contour" flag.
- `CSOBoolOp` improvements.
- `CSOBoostGeometry` can now optionally convert polygons to CSOs with a single seed point.
- `CSOConvertToXMarkerList` now has an additional mode where the relative position of the marker point in the CSO (start, inner, or end point) is stored as type.
- `XMarkerListSort` sorting is now stable.
- `XMarkerListCompare` can now ignore marker type .
- `XMarkerSamplePattern` can now optionally process the whole input list.
- Extended `CurveGenerator`.
- Extended `ApplicationVersioningSupport` Python module by a new `VCSQuery` module obtaining SVN revision infos for paths.
- Fixed `Bootstrap` update of trace images which failed if having averaged DTI data before.
- `FibersToGraph` now outputs graph usable by WEMVascularSystem.
- Several improvements to the `ModuleSuggest` user script.
- `SoCSOManualCorrectionEditor`: General overhaul and improvements.
- Performance optimization of `QuadratureFilter` .
- Update of `OpenCL` stubs to current version (2.1) due to deprecated OpenCL 1.0, 1.1, and 1.2 methods.
- Many network panels added or improved.

- Bug fixes
  - In `SoRampLUT`, the range and width parameters were swapped in the mapping to the internal `RampLUT` module.
  - `CSOVoxelSetListInfo` now allows to select voxel set at index 0.
  - `XMarkerListFilter` now sets the ListBase:ActNew action when updating its output.
  - `XMarkerListTransformation` now copies the input marker IDs correctly to the output list.
  - `XMarkerSamplePattern` now updates normal when changing `inputXMarkerIndex`.

DICOM related changes

- `ReleaseToolsDICOM` from `ReleaseToolsDICOM.h` and its corresponding namespace have been moved into the project `DICOMTagTools` and its corresponding namespace in file `mlDICOMTagTools.h`.

Renaming and rebuilding projects should solve all compile and linkage problems; the signature of one tool function has changed:

- `DCMTreeInfo determineDCMTreeInfo(DCMTree::Const_TreePtr dcmTree);`

  has been changed to

  `void determineDCMTreeInfo(DCMTree::Const_TreePtr dcmTree, DCMTreeInfo &dcmTreeInfo);`

  It may be necessary to include the header file `mlDICOMTreeInfo.h` to compile this.

- Tools for conversion between ML and DICOM located in `MLMultiFileVolume` have been moved into new projects `MLMLToDicomTools` and `MLDicomToMLTools`. If you only make use of such tool functionality then you may reduce your project dependencies only to one or both of these projects and remove `MLMultiFileVolume`.

- Many functions require a `DicomMessageCollector` (from project `DICOMCachedIO`) now which collects possible issues and other messages. The former way to return a descriptive string has been removed from a large number of functions.

- Changes on DICOM related modules:

  - New module `DicomSEGSave` for exporting DICOM SEG files, supporting the DICOM Segmentation IOD.

  - New module `MultiFileVolumeListSEGOutput` for providing details on SEG files.

  - New module `MultiFileVolumeListDiagnosisOutput`: A diagnosis and analysis module for MultFileVolumeList, providing for example some basic statistic and search functionality as well as validity tests.

  - New module `DicomConfigurableMessageFilter`: for configuring messaging and logging of modules using this.

  - `DicomModifyMultiFileVolumeExport` provides an option to load MultiFileVolumeCache-files now.

  - `DicomTreeCompare` can detect differences in value multiplicity now.

  - `DicomModifyImageTagsPlugin` created exception errors on files where NumberOfTags tags does not have a value.

  - `DicomTreeInfo` and other modules posted unnecessary errors on empty file name selections and disabled update modes when files still do not exist.

  - `DicomModifyTagsPlugin` module does not accept private element tag IDs with lower 8 bits < 0x10.

  - `DicomMultiFileVolumeListExport` did not export pixel data any more if no ImageTagsPlugin loaded them explicitly.

  - `MultiFileVolumeList*Output` modules had an unbounded volume index and/or displayed volumes out of range.

  - `Dicom*Save` modules have an 'updateDone' trigger output now.

  - `Dicom*Save` (except of `DicomSRSave`) modules have an input connector for `DicomTagModify` modules now.

  - `DicomTreeValidate` does not post errors on default instantiation any more.

- `MultiFileVolumeListOutput*` modules provide the currently used `MultiFileVolumeList` at a Base output for convenience now.
- `MultiFileVolumeList*Output` modules have the functionality to display references to original files/volumes now.
- `FileListFilterPlugin` has support for tag search in MF volumes as well as multi-value tag search support now.
- `DicomSCSave` doesn't save some tags anymore that are disallowed by the standard.
- `DirectDicomImport`:
  - Added some documentation about supported DICOM import functionality.
  - Improved performance of loading .nii files.
  - Fixed that Deformable Registration Object cannot be loaded.
  - Fixed problems with non-bitmask / fractional SEG files.
  - Fixed that same sized high dimensional (>4) NM DICOMs were sorted together to a broken image .
  - Fixed problems with separation of multi-segment SEG files.
  - Fixed that unknown standard tags were shown as unknown private tags.
  - Many optimizations in the import process have been implemented, especially on enhanced multi-frame files and large series. 2D pixel data is mostly loaded on demand now, which can speed up import times.

Caution

Errors and messages related to pixel data accesses in DICOM data do not appear during import time any more but much later when data is accessed.

  - When importing data via `ImageLoad`, DCMTK issues were not caught in the `DirectDicomImport` log stream but logged in the console.
  - The panel has been modernized, many elements have been moved into specific dialogs which can be opened with buttons or with double click on a volume in the volume list, all elements and fields still exist; some buttons have been realized as more compact enumerator fields.
  - Fixed that "Position Reference Indicator" tag was removed when importing Enhanced MF DICOM files.
  - `DirectDicomImport` and related `MultiFileVolumeList*Output` modules produced too many/much error output for a single error message.
  - `DirectDicomImport` and `itkImageFileReader` handle handle Sign Extensions of pixel data as well as shifting now.
  - Can read Parametric Maps now.
  - Fixed that decomposing of enhanced multi-frame files did not maintain the structure of Functional Group sequences.

- Fixed that writing relative paths did not work for enhanced DICOM images.
- `DirectDicomImport` and other modules with bit-overlay outputs used SET instead of OR for overlay merges.
- RWV (Real World Mapping Storage) data should not be handled as image data any more.
- Fixed error on valid import of Enhanced DICOM files with private tags in Functional Group macro sequences.
- Fixed crashes while importing corrupted XA data.

- `MultiFileVolumeListSROutput`:
  - Handles retired SR data more strictly now. The module has been revised to provide permission flags which still allow retired structures.

Caution

The module is not backward compatible any more, check functionality and reconfigure modules if they are stored in networks.

- `DicomREGSave`:
  - Handling mismatch of MeVisLab Worldmatrix and DICOM world tags (ImagePositionPatient, etc.). Source can be selected now.
  - `DicomREGSave` and `DicomSEGSave` add tags of Common Instance Reference Module now.
- `MultiFileVolumeListDraftView`:
  - Also supports other frame types than DICOM as well as enhanced multi-frame frames.
  - The robustness against frames without Image information was improved.

- Other DICOM related fixes and changes:
  - Some functionality has been moved to new or other existing libraries.
  - The new Value Representation UR is handled in many places now.
  - Many input fields (for example in `DicomREGSave`) check values with VR CS for standard compliance now, invalid strings will not be saved any more.

Caution

If your application stored invalid strings before, this will not work any more. This changes the behavior especially of `Dicom*Save` modules.

- Known issues:
  - Currently `DirectDicomImport` cannot distinguish between DICOM images with RGBA value representation - as for example stored by `ImageSave` - (which do not obey the DICOM standard) and valid ones only having RGB photometric representation; therefore such images are shown incorrectly.
  - `DirectDicomImport` is missing support for "Philips3D" files.

ITK

- Fixed a few misbehaving module operations.
- Minor changes and cleanups, scripts ported to Python.

VTK

- Changes:
  - Updated VTK to version 7.1, of 2016-05-03.
  - Renamed conversion module module `MLBaseToVTKPoints` to the more correct name `MLBaseToVTKPolyData`.
  - Dropped deprecation support for versions before MeVisLab 2.3.
  - Script components in generated module wrappers use Python instead of JavaScript now.
  - New `isFilePath` MDL feature is used for file names and paths. This may require a confirmation of values in file and path fields when updating networks with stored paths.
  - Removed a number of wrappers for deprecated VTK classes and of many others which do not make sense as wrapped modules. Also some deprecated and useless modules and fields have been removed
- Caveats:
  - In some situations MeVisLab redraw events get lost when rendering objects in VTK renderers. This can lead to missing redraws of the MeVisLab GUI. Dragging at application borders can resize and redraw MeVisLab in such cases.
  - Some classes wrapped as VTK modules are still unstable or have missing example networks.
  - Many connectors and fields have changed. Usually they will automatically be reconnected to new connectors, however this is not always possible. Thus stored networks may be affected or become incompatible.
    - If stored field names could not be restored, in most cases (re-)saving the network removes the warnings.
    - If connections cannot be restored, carefully check which one is missing. In most cases they can be re-established manually.

Point Cloud Library

- New module `PCLStatisticalOutlierRemoval` which uses point neighborhood statistics to filter outlier data.
- New module `PCLRadiusOutlierRemoval` which filters points in a cloud based on the number of neighbors they have.
- New module `PCL83xLoader`: Experimental loader for IMAGINEX Azimuth DELTA Sonar .837/.83B file pairs.
- `PCLBaseObjectWrapper` has a another `setPointMemberXYZIntensityReplacement` function which allows to set a larger point set at once from NumPy arrays.

New third-party libraries in `FMEWork/ThirdParty`

- polar_decomp: ANSI C code from the article 'Polar Matrix Decomposition' by Ken Shoemake, `<shoemake@graphics.cis.upenn.edu>` in 'Graphics Gems IV', Academic Press, 1994.
- httmock: A Python library for mocking requests.
- request: "HTTP for humans" Python module.

Windows Edition

- The Mesa OpenGL fallback driver has been updated to version 12.0.0.
- HiDPI screens are supported on Windows now. Some small bugs may still exist though, especially for mixed display setups.

macOS Edition

Note

This release requires at least OS X 10.11.5 (El Capitan) and Xcode 8.1 for development.

- Full support for macOS 10.12 (Sierra).
- Updated the MeVisLab macOS Edition guide.
- Updated the OsiriX MeVisLab Bridge guide.
- The ToolRunner application may be used to setup Xcode workspaces for dependent or related projects. See *Creating Multi-Project Xcode Workspaces using ToolRunner* in the MeVisLab 3.0 macOS Edition guide.
- The OsiriXBridge modules, the plugin, as well as the documentation have again been included into MeVisLab. Support has been extended to Horos. The source codes are being published to the GitHub community repository.
- For module development, the created Xcode projects will now build against the newest SDK shipped with Xcode and set the deployment target to the OS release against which MeVisLab has been build.

Linux Edition

- One can create NetBeans C++ project files from .pro files now.
- Added command line option `-no-opengl` for running batch jobs on Linux.

ADK

- Make installation of applications possible without admin rights.
- One can create TestCenter addon installers for applications now.
- Improvements to .mlinstall/.mli files:
  - Added a `PREPROCESS_MKDIR` command that creates a directory immediately when encountered.

  - Added a `PREPROCESS_CHECK_EXECUTE` command that executes a command immediately when encountered and sets the `LAST_EXIT_CODE` variable.
  - Only specific environment variables are automatically imported into .mlinstall/.mli files as is (mostly those starting with MLAB_). All other environment variables need to be prefixed with ENV_.
- Created a MeVisLabAppConsole executable that writes output to the console on Windows.
- The ModuleDependencyAnalyzer collects *.ts (translation) files now.
- JavaScript files are not signed in installers anymore.

Releases before 3.0

For the release notes of older MeVisLab releases see this document.