3.4.0

US 1159	Added option to enable/disable (auto) back face culling in Simple transparency mode (or when using transparent legend colors). See cee::ug::ModelSettings::setSimpleTransparencyCullMode() for more info.
US 1150	Added export of triangle-based unstructured grid geometries in the current view to Wavefront OBJ (.obj + .mtl) files in the CeeExport component. Note: Experimental - Interface and functionality may change in the future.
US 1157	Added support for coloring vectors by scalar even if the part has fringesVisible set to false.
US 1161	Add support for user controlled ambient intensity in UnstructGridModel. Use ModelSettings.setAmbientIntensity() to control the global ambient light intensity.
US 1162	Added new helper functions in Camera: rotateAboutModelAxis() and rotateAboutScreenAxis()
US 1164	Above, below and undefined opacity have been refactored to be above, below and undefined masking. This fixes several issues with using opacity to mask out above, below and undefined parts of the model and will produce a correct image. Changes to cee.ug.ColorMapper: setAboveRangeOpacity(float opacity) -> enableAboveRangeMasking(bool enableMasking) setBelowRangeOpacity(float opacity) -> enableBelowRangeMasking(bool enableMasking) setUndefinedOpacity(float opacity) -> enableUndefinedMasking(bool enableMasking)

US 1167	Refined how result sections and integration points work in UnstructGridModel.
	
	If elements have more than one section, two groups will be created: "Top" and "Bottom". Defaults will set the Top section group to contains the last section for every category and set the Bottom section group to contains the first section for every category.
	
	Section categories are mainly divided as the following:
	
	SOLIDS: usually identified with the list [Section 1,Section 2,...]
	
	SHELLS : usually identified with the list [Bottom, Middle, Top...]
	
	BEAMS: usually identified with the list of Integration points: [Int. Pnt.1, Int. Pnt.2 ...]
	
	Stress and Strain (for instance) for the top section group would be: "XX Stress[Section 3, Shell Top, Beam Int. Pnt 4]". If there is only one section in a category, the section suffix will be ommited. For instance, if SOLIDS only have Section 1, the name will be: "XX Stress[Shell Top, Beam Int. Pnt 4]".
	
	To get the information of every section category and every section available use from the DataSourceDirectory the methods:
	
	std::vector<SectionInfo> sectionInfos() const;
	
	std::vector<SectionCategoryInfo> sectionCategoryInfos() const;
	
	Default choices for Top and Bottom section groups can be changed in the ReaderSettings using method setTopAndBottomSectionIdForCategory(int categoryId, int sectionIdTop, int sectionIdBottom).
	
	To make effective the changes a call to the method:
	
	DataSourceCae::PollChangeResults DataSourceCae::pollForChanges() must be done to trigger the updates.

US 1156	Added optimized helper class DataElementSetBuilder. Use this instead of calling DataElementSet::addItems() multiple times when creating element sets.
US 1166	Added DataElementSet::elementIndicesForPart(int geometryId, int partId) for optimized retrieval of all element indices for the given geometry and part.

3.3.0

US 1121	cee::geo::Part now has an id that can be set by the user. Default is -1. It's up to the user to decide if this id is unique or not. cee::geo::GeometryModel can now get part by it's id. Will return the first one found if there are more parts with the same id.

US 1126	Added support for right aligned legends with tickmarks on left side. Use cee::ug::ScalarSettings::setTickMarksPosition() or cee::vis::CategoryLegend::setTickMarksPosition(). NOTE! Breaking change: cee::vis::CategoryLegend::setLabelsOnLeftSide(true/false) is replaced with cee::vis::CategoryLegend::setTickMarksPosition(LEFT/RIGHT).
US 1132	Ceetron Python Modules (CPM) - CDC targeted towards python users and Visual Workflow Added a separate CDC distribution for Python users. Focus on how to use CPM in automation workflows and as a content produces for Ceetron Cloud [Private]. New features: - Render to image from Python without having a GPU. Using OSMESA to enable shader based rendering in software from a Python script on any VM/Docker container. - Updated documentation on how to use with Python - New examples focusing on automation - Examples for use with Jupyter Notebooks. - Easier dependency management, no need to set path - Support for Python 3.8 (and 3.6, 3.7) - Better error handling if using an incompatible version of Python
US 1137	Added cee::ug::PartSettings::setIntersectable(). Default true, meaning the part is "selectable". If set to false, the part will be excluded when doing rayIntersect(), regionIntersect() and such.
US 1140	Added option to enable/disable (auto) back face culling in Simple transparency mode (or when using transparent legend colors). See cee::ug::ModelSettings::setSimpleTransparencyCullMode() for more info.
US 1118	geo::DataGeometry: Added helper functions createCircle() and createArc()
US 1119	CameraInputHandler: Added functions to set rotate/roll/zoom sensitivity for mouse movement
US 1120	Added CENTER as layout corner option for cee::vis::OverlayItem. The overlay item will be positioned centered in the view, both vertically and horizontally
US 1138	Added several optimizations to how the cee::geo::GeometryModel is updated. Note: This model is best suited for static data. Consider splitting up into smaller models and/or using cee::vis::MarkupPartInstancedGeometry if you have dynamic data that changes often.

3.2.0

US 1070	Data source directories now provide metadata about the simulation run or produce the result database, this includes a title, a description, a simulation and a solution type. See cee::ug::SimulationInfo
US 1099	Added support for showing different scalar results as fringes per part. You can now in the ModelSpec override the global result selection for a given part. Added cee::ug::ModelSpec::setOverridePartFringesResultId()
US 1103	CameraInputHandler now supports "ROLL" navigation type. "ROLL" rotates the model around view direction.
US 1105	Optimized rendering of cee::geo::DataIndexedPolylines.
US 1108	Added support for picking on bounded clipping planes
US 1109	Optimized clipping planes performance and appearance. It is now possible to combine cutting plane clipping and view clipping in on UnstructGridModel. Added ViewClippingPlane VTFx properties.
US 1110	CAE Import: Updated to Abaqus 2020
US 1111	CAE Import: Using Native State Names for Abaqus, Nastran and Hyperworks. Combine Solution Type + Frame Description.

US 1112	Added option to ignore view clipping for specific models. Applies to UnstructGridModel (cee::ug::ModelSettings::setIgnoreViewClipping()) and MarkupModel (cee::vis::MarkupModel::setIgnoreViewClipping()). Query current state using isViewClippingIgnored().
US 1113	Get result min/max on DataSourceInterface without having to load the result. All DataSources deriving from this will support the query: VTFx, VTF, DataSourceCae. Added scalarRange, vectorRange and displacementRange to cee::ug::DataSourceInterface.
US 1107	Added support for showing crinkle cut surfaces. Added cee::ug::CuttingPlane::setCrinkleCutSurface(). If enabled, the cutting plane surface will be the combined surface of all elements intersected by the cutting plane. So the cutting plane will show the tesselation of all intersected elements. This is useful for inspecting and visualizing the mesh of a volume element model.

3.1.0

US 1027	Added support for drawing "old style" simple vectors (line+two triangles, no shaded arrows).
US 1020	Added Blue-White-Red color scheme to UG legends. See cee::ug::ColorMapper::ColorScheme::BLUE_TO_WHITE_TO_RED for more info.
US 1025	Added support for reading paralell VTU files (PVTU)
US 1028	Optimized handling of many (>500) states. CDC is now order of magnitude faster when handing thousands of states. Animation setup is much faster, and changing draw style has been optimized and is a lot faster.
US 1055	Added option to only draw every n'tn vector on the UnstructGridModel. See cee::ug::VectorSettings::setDrawSkipBy()
US 1056	Optimized display model generation with target on many time steps and parts and changing draw style. Also optimized rendering performance.
US 1057	Added the concept of situations (cee::ug::Situation). A situation stores the current view-/model setup (display model), element sets and custom user data. You can apply a situation to the current model and view to recreate the display model from the situation. Situations can be saved to disk and be exported/imported to fit with your current workflow (using cee::ug::SituationIo).
US 1058	Fix memory growth when loading openFoam cases.
US 1059	Part names can contain dots now. No truncation in name if dot found.
US 1060	If a surface cells has a unknown value it will now use the value on its neighbor cell
US 1061	DataSourceVTFx can now be set to read only shells elements. Call method setReadShellsOnly(true) before calling open. Under this configuration Sets are not supported.
US 1063	Added Python end-user methods for result and state filtering in ExportDataSourceVTFx
US 1064	Added an openFOAM reader option to avoid loading processor boundary parts
US 1065	Added relative displacement storage option for VTFx exports. Useful when displacements are very small vs geometry
US 1066	Added support for results mapped on element-surfaces in ImportCae
US 1029	Added DataNodes. values() and DataNodes.ids() in C# (instead of rawNodesPointer() and rawIdsPointer() Added ElementsQuery::allElementCentroids()
US 1046	Added setting for only using first order nodes of higher order elements for visualization. See cee::ug::ModelSettings:setUseFirstOrderElementNodesOnly() for more information.

3.0.0

US 887	Optimization of isovolume, isosurface and cutting plane computation.

US 992	Isosurface/Isovolume: Added support for computing and mapping isos from scalar results of all results mapping types (per node, per element, per element node, per element surface).
US 994	cee::ug::DataElements::addElements(): Significant performance improvement (6x speedup).
US 996	Optimized handling of non-volume element parts and loading of VTFx files containing node blocks with ids.
US 997	UnstructGridModel: Optimized display model generation. Apply results and switching between draw styles is now a lot faster (2x - 5x faster).
US 1003	Changed the license key environment variables from CEETRON_3D_COMPONENTS* to CEETRON_DESKTOP_COMPONENTS*. Use CEETRON_DESKTOP_COMPONENTS_LICENSE_KEY_A and CEETRON_DESKTOP_COMPONENTS_LICENSE_KEY_B.
US 1006	Added setting for controlling if the above/below legend range color markers should be drawn with the color legend. See ScalarSettings::setAboveBelowRangeColorMarkerVisibilityMode() for more info.
US 1007	Added support for reading and writing PNG and JPEG images. See cee::ImageIoJpeg, cee::ImageIoPng for more info.
US 1008	Added support for bulk calculations on parts and cutting planes. The following values are computed on the part/cutting plane surface: min/max/average scalar result, area, centroid, surface/node count and resultant (surface_area*surface_result summed for all surfaces). See cee::ug::BulkCalculation for more info.
US 1010	Added support for contour lines on parts, cutting planes and isosurfaces. Any scalar result can be shown as contour lines, independent of what is shown as filled countours (fringes). See ModelSpec::setContourLinesResultId(), CuttingPlane/Isosurface::setMapScalarResultId() for more info.
US 1011	Added Python automation examples. See the README.txt file in the Examples/Python folder for more information.
US 1012	Added support for Data Provider Plugins in UnstructGridModel. Data Providers allow users to create their own providers (e.g. file readers) that dynamically plugs into the server and behaves like any built-in reader. The Data Provider is developed in C++ and compiled as a .dll/.so/.dylib. To develop Data Providers you can download the Data Provider Framework from ceetron.com. DataProviders are used in the ImportCAE component. To use a data provider in CDC, load it with the cee::imp::cae::DataSourceCae::loadDataProviderPlugin() method. Data providers also support pushing updated data to the server, making it possible to alter the data delivered by the provider during the session. The same data provider can be used in the Ceetron Cloud Components (C3) server.
US 1013	Added support for ResultCalculator plugins. By using CeetronResultCalculatorFramework (separate download, free to use) customers of CDC users can add derived results implemented in C++ via .dll/.so/.dylib loaded at runtime into the CDC based app. A calculator specifies the number of input results required and produces an output result.
US 1014	Starting with version 3.0, only 64 bit versions of CDC are included in the distribution. The WIN32 version is deprecated (Linux has been x64 only for a long time). Minimum requirement for Windows is now Visual Studio 2015, and we distribute a single version for VS2015, VS2017 and VS2019. Minimum requirement for Linix is Ubuntu 14.04 (gcc 4.8.4)
US 1016	Update of 3rd party CAE readers. Added support for ANSYS 2019R2, Hyperworks (H3D) (requires separate license). Several bug fixes and optimizations to all CAE readers. Optimized performance of all CAE readers.
US 1017	Ceetron 3D Components (C3DC) is now named Ceetron Desktop Components (CDC). Documentation and source code is updated according to this.

!Backlog

US 554