Ceetron Cloud Components

Release Notes - New Features

2.0.0

US 668	Added support for highlighting of parts in the geometry component. Added cee.geo.PartSettings.highlight property
US 667	USG: Added a new component for client-side (in-browser) handling of surface FEA models. Suited for streaming/monitoring as well as transient animations with very fast updates. Supports all result mappings (per node, per element, per element node), all result types (scalars, vectors, displacements and rigid body transformations), all draw styles (surface, surface mesh, outline, lines, points), multiple states, adaptive geometries. Supports any number of nodes per element.
US 666	Improved performance of the OpenFOAM file reader.
US 665	Added full support for polyhedron elements in the RemoteModel. Elements with any number of faces and any number of face nodes can be defined. Added polyhedron support to the OpenFOAM and VTU file readers.

1.2.8

US 661	Added support for per part transformations in the Geometry Model. Added cee.geo.Part.transformationMatrix property.
US 660	Added support for drawing thick lines in Geo module. Drawing of thick lines is supported by using the cee.geo.MeshIndexedLines mesh class and then setting the lineWidth in part settings to a value greater than 1.
US 659	Added support for normals in .cgeo files. The cee.geo.CgeoModelGenerator now handles per vertex normals.
US 658	Added Camera.fitViewOrtho() to allow fit view operations when using an orthographic (parallel) projection.
US 657	Added query to get all nodes for a part in a given frame in the RemoteModel. Added cee.ug.QueryPartNodes class.
US 656	Added support for eye lift in the RemoteModel. Eye lift can be used when facing z fighting issues (triangles rendered in the same plane). By applying eye lift to one part, it will be moved slightly towards the eye, resolving the z fighting issues. The current version offers 3 eye lift factors to choose from. See cee.ug.PartSettings.eyeLift property for more information.
US 649	Added CeeCloudServer.isVTFxFile() to the RemoteModel server. Returns true if the file is a valid VTFx file.
US 646	Added texture support to line parts in the geometry model (MeshIndexedLines).
US 645	Added pixelScaleFactor to ViewerOptions, allowing custom factors and also work around embedded corner cases where window is not accessible.

1.2.7

US 643	Added support for eye lift in mrk.PartIndexedTriangles. Useful for e.g. highlighting. Eye lift can be used when facing z fighting issues (triangles rendered in the same plane). By applying eye lift to one part, it will be moved slightly towards the eye, resolving the z fighting issues. The current version offers 3 eye lift factors to choose from.
US 642	Added support for orthographic (parallel) projections. Use Camera.setProjectionAsOrtho() to setup an ortho projection.
US 640	Multiple enhancements to text labels. Labels with an offset direction now work better together with multi line labels. The label attachment lines have been made slightly longer and the actual attachment points are highlighted by drawing a colored point. Added scaling of label font and attachment line length according to pixel scale factor (retina displays)

US 639	Refactored scalar mappers and color legend. Added support for Continuous (ScalarMapperContinuous) and non-uniform filled contours (ScalarMapperFilledContours) scalar mappers. Overlay.addCustomColorLegendForScalarMapper() now supports all 3 scalar mappers. Added support for mapping scalar values to texture coordinates and updating textures, thus making it possible to scalar mapping of results in the client side geometry model.
US 625	Added support for textures in the Geometry Model. Added geo.PartSettings.texture property. Added texture coordinates to MeshIndexedTriangles (via optionalVertexData parameter to the constructor). Added reading of textures from .cgeo and extended the .cgeo format to include texture sampler options.

1.2.6

US 638	GeoServer: Added support for preserving hard edges on models with shared nodes when adding the model to the server with the /api/v1/model/:modelKey/proto_addCgeoToDatabase REST API call. Added query parameter preserveSharpEdges=1. So to add a model and preserve edges, use: /api/v1/model/:modelKey/proto_addCgeoToDatabase/:cgeoFilename?preserveSharpEdges=1 Note that this will increase the data volume needed to be stored and transmitted to the client.
US 637	Added support in the geometry model to preserve hard edges in a geometry with shared nodes. Added MeshIndexedTriangles.constructPreserveSharpEdges() to create a triangle mesh for a geo.Part where the nodes on the edges are duplicated if the two triangles sharing this edge has a face normal above the given crease angle. Added GeometryModelRemoteLoader.configurePreserveSharpEdges() to be able to instruct the remote loader to split sharp edges when streaming parts from a remote server.
US 636	Added support for eye lift and polygon offset in the GeometryModel (cee.geo.PartSettings.eyeLift/polygonOffset). Eye lift can be used when facing z fighting issues (triangles rendered in the same plane). By applying eye lift to one part, it will be moved slightly towards the eye, resolving the z fighting issues. The current version offers 3 eye lift factors to choose from. Polygon offset can be used when drawing lines on top of triangles (e.g. to indicate a mesh or other marker lines). By applying polygon offset to the triangles parts, they will be pushed slightly away from the eye, ensuring that the lines will be drawn without interference from the triangles.
US 580	Extended text labeling support through the mrk.PartLabels class. The class now supports offsetting of labels using a 3D offset direction. The new PartLabels.addWithOffsetDirection() function allows adding labels with a position and an offset direction. The offset direction will be used to offset the labels slightly in the specified direction. The PartLabels.addWithOffsetDirectionFlipTowardEye() lets you specify a similar offset direction, but will flip the direction vector during drawing so that it always points towards the viewpoint.

1.2.3

US 626	Added Send-To-Cloud feature to the ug.RemoteModel. Using the cee.ug.RemoteModel.sendToCloud() method you can share the current model with all settings to anyone with a browser using Ceetron Cloud or your own sharing portal. Right from your own application.
US 577	Added support for geometric clipping in Views. Up to 6 user-defined clipping planes can be specified, and "bounded clipping planes" are support by allowing the user to control how many planes that must clip the geometry before it is considered clipped and not rendered. See the Clipping class for more info (View.clipping property). Added two new classes: cee.Clipping and cee.Plane.

1.2.2

US 623	Added experimental support for a single step mode in CeeCloudServer. By specifying the "singleStepMode" configuration flag to RemoteModel.openModel(), the server will only load a single step at a time, thus greatly reducing the memory resource usage of animations on the server. Note that this might slow down the server.
US 622	Added support for reading OpenFOAM (*.foam) files. Note that polyhedra elements is still not supported.

US 621	Added support for copy on open in CeeCloudServer (using the "copyOnOpen" config flag passed to RemoteModel.openModel()).
US 620	Added support for reloading an analysis in the RemoteModel. RemoteModel.reload() will reload the analysis files on the server and update the client with the changed data. The ModelDirectory will be updated with any new states and result variables.
US 619	The CeeCloudServer is now more proactive in closing any open VTFX files and releasing native session memory whenever a client explicitly closes a RemoteModel or is disconnected from the server. Previous versions relied on Node.js' garbage collection in order to close files and release native memory.
US 617	Added server side cleaning of part piece cache. Clear out data payload of normal part pieces after they have been sent (but retain the records to keep CRC working). Completely prune volatile part pieces from the server cache once they have been sent.
US 609	Now allowing non-power-of-two (NPOT) textures if the texture options comply with WebGL requirements: No mipmaps or tiling.
US 579	Added region selection of parts in the Geometry Model. See geo.GeometryModel.regionIntersect() for more information.

1.2.1

US 608	Added helper class for producing color tables (cee.ColorTableFactory). Useful for scalar mapping in the geometry model, and to produce part colors (as in ug and C3DC).
US 607	Added PartPoints to the MarkupModel. Use MarkupModel.addPointsPart to create a new instance of this class.
US 606	Added a technology preview version of a new geometry server (CeeCloudGeoServer). The GeometryServer is a high performance stateless server implemented in TypeScript. It has a very low resource overhead on the server and support fast loading of remote geometry models. Note: This is a tech. preview version and is not production ready.
US 605	Big performance increase when partially updating the geometry model (cee.geo.GeometryModel). Updates when only some of the parts are changed is now a lot faster.
US 604	Added support for showing node averaged scalar results. New property: ScalarSettings.nodeAveragedValues.
US 603	Draw divider lines between viewports when showing multiple views in a viewer.
US 602	Added GeometryModel.deletePartsAt()