Open CASCADE Technology
Version 7.4.0

Release Notes

October 1, 2019

Overview

Open CASCADE Technology 7.4.0 provides more than 500 improvements and corrections over the previous release 7.3.0.

Open CASCADE Technology

Highlights

Modeling

- Improved robustness, performance and accuracy of BRepMesh algorithm
- Options to control linear and angular deflection for interior part of the faces in BRepMesh
- Improved robustness and stability of Boolean operations and Extrema
- Enabled Boolean Operations on open solids
- Option to suppress history generation to speed up Boolean Operations
- Option to simplify the result of Boolean Operation
- Possibility to calculate surface and volume properties of shapes with triangulation-only geometry
- A new interface for fetching finite part of infinite box in BRepBndLib
- New “constant throat” modes of chamfer creation
- Removal of API for old Boolean Operations

Visualization

- Improved support of embedded Linux platforms
- Selection performance improvement
- Support of clipping planes combinations (clip by box, ¾, etc.)
- New class AIS_ViewController converting user input (mouse, touchscreen) to camera manipulations
- Improved font management
- Improved tools for visualization performance analysis
- Option to display the outline of shaded objects
- Option to exclude seam edges from Wireframe display
- Option to display shrunk mesh presentation
- Possibility to show shapes with dynamic textures (video)
- Support of reading encoded bitmap image from memory buffer
- Removal of the deprecated Local Context functionality from AIS
- Removed dependency from gl2ps (relying on deprecated OpenGL functionality)

Data Exchange

- New tools to import mesh data from glTF 2.0 and OBJ formats
- Support of some non-ASCII encodings in STEP import
- Support of XDE data (assembly structure, colors, names) in export to VRML

Draw Test Harness

- Improved 3D Viewer camera manipulations
- Fixed issues with starting Draw Harness from batch scripts
- Improved support of running Draw Harness in environment without CASROOT

Other

- Improved performance of built-in parallelization routines (OSD_Parallel)
- Tools for convenient and efficient traverse of BVH structures
- Optimization of TPrsStd_AISPresentation attribute
- Sample of 3D Viewer integration in glfw application

Open CASCADE Technology

New Features

Refactoring of BRepMesh

BRepMesh component has been refactored. The new architecture simplifies the process of mesh generation over OCCT's BRep models and enables processing of some corner cases that were hard to handle before.

Improvements

Major improvements of the BRepMesh component have been implemented in the context of the issue #0026106 "BRepMesh - revision of data model" and its descendants:

Internal code structure

- New internal data structures as a backbone;
- Clear separation of data structures, auxiliary tools, and algorithms;

Performance

- Edges tessellation in parallel mode;
- Execution of checks, healing of discrete model, and model pre- and post-processing in partially parallel mode;
- Localization of re-tessellation in case of intersections of discrete representations of edges in the face by the relevant edges only and not the entire face;

Robustness and quality

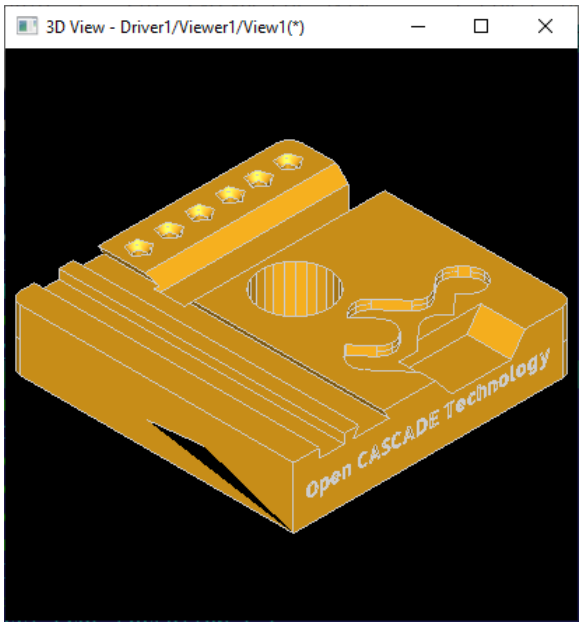
- Pre-processing of a data model to resolve common cases of self-intersections of discrete representation;
- Improved presentation of cones near seam edges;
- Improved presentation of spheres near poles;
- Two new parameters to control linear and angular deflection of the interior part of faces separately from their boundaries;
- Additional improvements for NURBS surfaces to fit specified linear and angular deflection;
- Some improvements to resolve unrestricted consumption of memory and hang-ups on specific cases of NURBS surfaces.

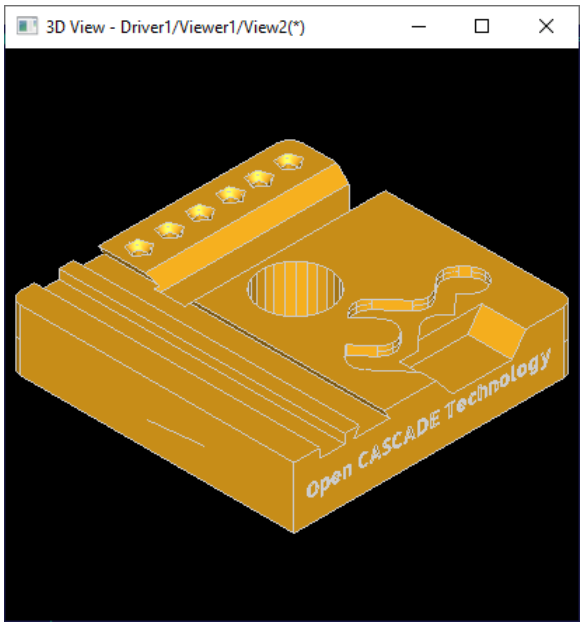
Examples

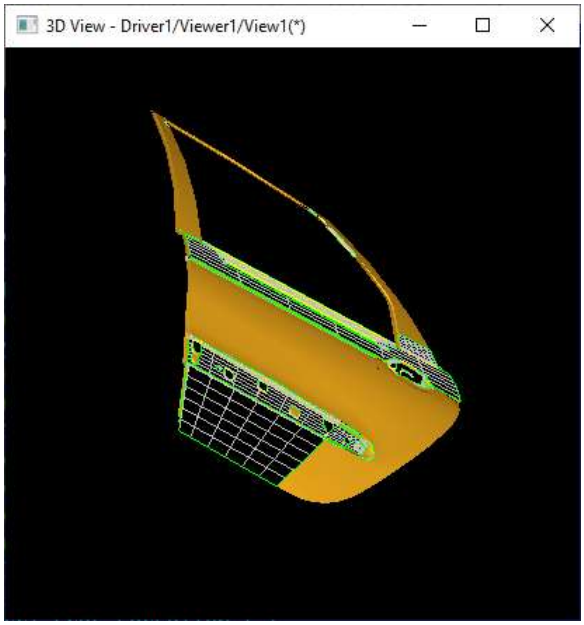
Visual improvements

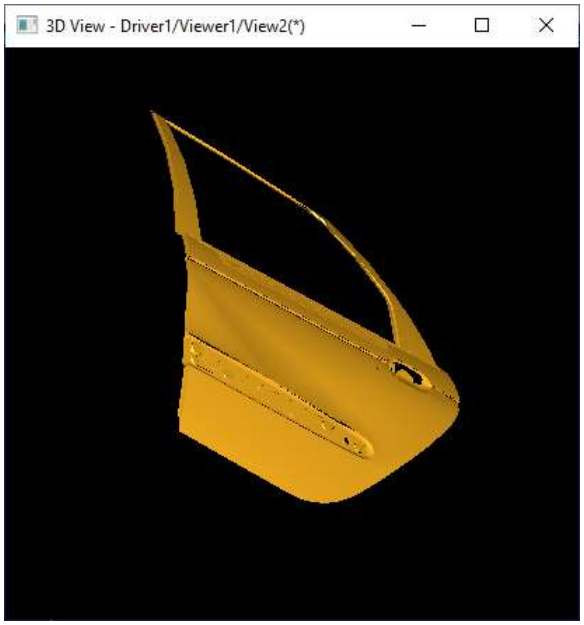
OCCT 7.3.0

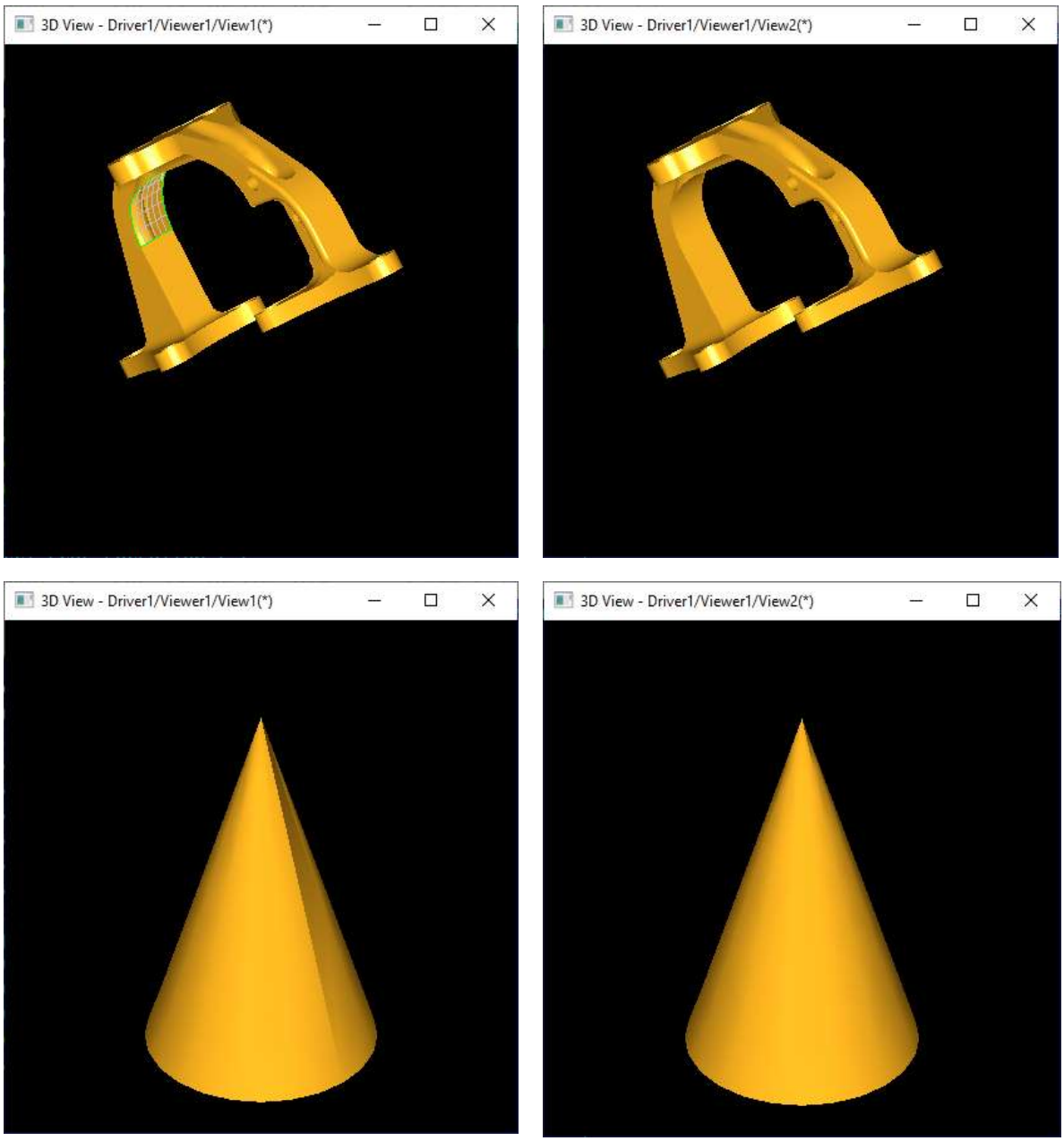
OCCT 7.4.0





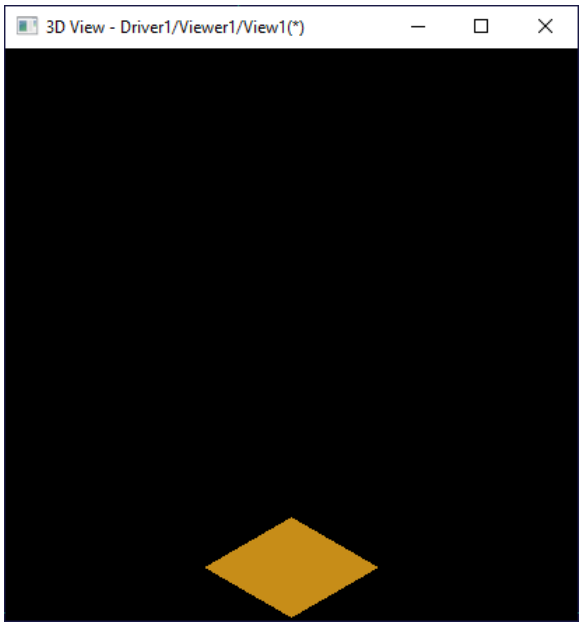






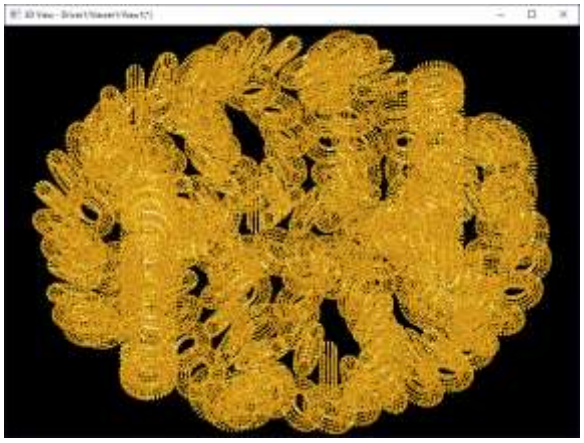
Accuracy improvements

OCCT 7.3.0

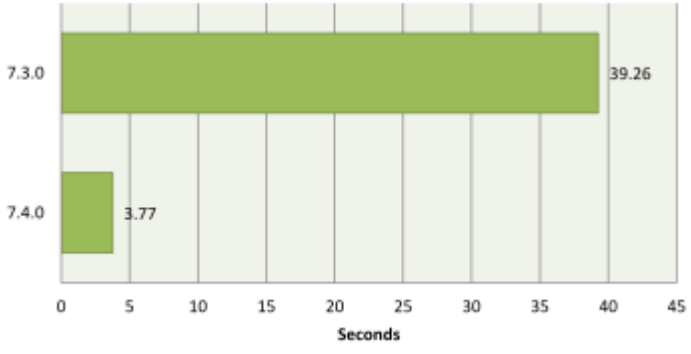


OCCT 7.4.0

Performance improvements



The shape above contains an assembly where a single solid box is replicated ~93000 times. The performance on this case has been improved by more than 10 times comparing to 7.3.0:



Implications on porting

In the new version deflection is controlled more accurately. Due to this improvement it may be necessary to tune parameters of call of the BRepMesh algorithm on the application side to obtain the same visual quality of presentation and/or performance as before.

For the details related to the structure and the usage of updated component see Meshing part of OCCT documentation.

Thread pool for multithreaded tasks

Built-in multi-threading support (when OCCT is used without TBB) has been improved with thread pool support. Algorithms relying on **OSD_Parallel** for multithreaded execution will benefit from lower overhead due to reusable thread resources and from better balancing within nested multithreaded algorithms chain (previously each nesting level created independent thread loop, which resulted in threads number multiplication).

The size of global **OSD_ThreadPool** instance can be adjusted for better control of CPU resources usage by OCCT algorithms on application level. Algorithms using **OSD_ThreadPool::Launcher** directly (in contrast to OSD_Parallel) can define working tasks receiving index of a thread within the pool; this can be used to preallocate thread-local storage (like file handles) to optimize work with data.

If OCCT is built with TBB support, it is possible to dynamically switch OSD_Parallel::For() between using either TBB or built-in parallelization implementation via **OSD_Parallel::SetUseOcctThreads()** global flag. Within the Draw Test Harness the parallelization parameters can be adjusted via new command **dparallel** like this:

```
> pload MODELING
> dparallel -occt 1 -nbThreads 10 -nbDefThreads 10
> incmesh shape 0.1 -parallel
```

Tools for convenient traverse of BVH structures

New **BVH_Traverse** and **BVH_PairTraverse** classes have been introduced for convenient traverse of BVH tree and a pair of BVH trees, respectively. These classes implement the BVH tree descend taking into account different scenarios and optimizations such as:

- Descent by the best branch. Each node has a metric, and the node with the best metric is always processed first, which allows finding the necessary result faster.
- Full inclusion test. If at some point the node of the tree is fully accepted (i.e. its box fully matches particular criteria), its children are not going to be checked and will be accepted automatically.

To use these classes user needs to implement the methods defining the rules for the node rejection and leaf acceptance:

- **RejectNode** - operates with the bounding box of the node and should provide comparison of the box with some criteria and define if the node should be rejected.
- **Accept** - performs processing of the elements of the tree.

The methods for optimizations of the tree traverse are optional:

- **RejectMetric** - compares the metric of the node with the global one and determines whether the node should be rejected.
- **IsMetricBetter** - compares the two metrics and defines the direction of the tree traverse.
- **Stop** - stops the traverse when the result is achieved.

New classes **BVH_Distance** and **BVH_PairDistance** implement tools to find the min/max distance between some object and BVH tree or between two BVH trees. All optimizations are already implemented, user just needs to define:

- In case of **BVH_Distance:** how the distances from the object to the box and to the element of BVH tree are computed.
- In case of **BVH_PairDistance:** how the distance between the elements of the trees is computed.

The new **BVH_BoxSet** class implements simple and convenient interface to add elements into the tree.

Tools for reading glTF and OBJ files

New classes **RWGltf_CafReader** and **RWObj_CafReader** with TKRWMesh toolkit have been introduced to read glTF and OBJ files. Readers consist of two parts: low-level reader relying on internal structures and a translator into an XDE document. Low-level reader can be used for translating data into non-XDE structures, including file format features, currently unsupported by XDE document.

glTF reader supports 1.0 and 2.0 versions of format specifications and handles common (glTF + bin), embedded (base64-encoded mesh data) and binary glTF (glb) files. Reader preserves scene structure, triangulation, names and colors. The reader relies on RapidJSON library, which should be enabled (HAVE_RAPIDJSON) while building OCCT from source code.

OBJ reader supports reading polygonal information, names (groups) and colors (from MTL file). Polygons are automatically split into triangles using BRepMesh tools.

glTF (and most OBJ) files use Y-up coordinate system which can be unexpected by CAD systems working with Z-up coordinate system. To handle this, glTF and OBJ readers allow defining file and system coordinate systems as input parameters (see **RWMesh_CoordinateSystemConverter** tool).

New readers can be accessed from Draw Harness via commands **ReadGltf** and **ReadObj** like this:


```
> pload XDE VISUALIZATION
> ReadGltf D Buggy.gltf
> vinit
> XDisplay -dispMode 1 D -explore
> vfit
```

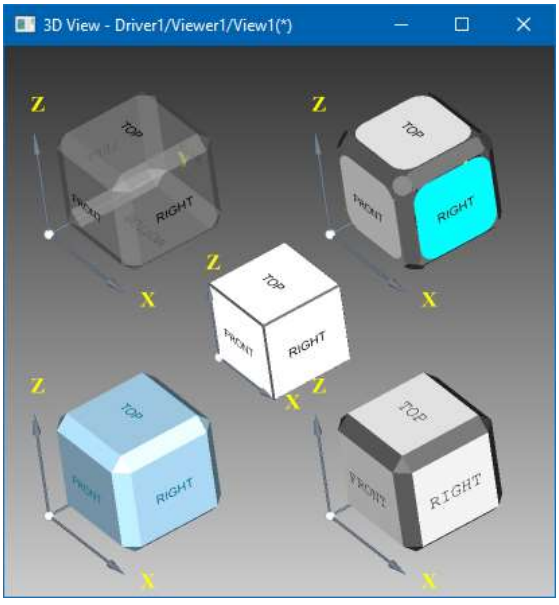
Interactive view cube

A new class **AIS_ViewCube** has been introduced to provide an interactive alternative to **AIS_Trihedron** - a resident auxiliary object displayed in a view corner indicating axes of global coordinate system.

AIS_ViewCube represents an interactive cube with pickable cube sides, edges and vertices, which allow user to open one of standard views. Transition is done with smooth animation. The look-n-feel of a new object can be adjusted to fit color scheme used by the application.

New object can be activated in Draw Harness via the **vviewcube** command like this:

```
> pload VISUALIZATION
> vinit
> vviewcube vc
```



View Controller

A new class **AIS_ViewController** implements mapping of user input events from various devices (mouse, touchpad, keyboard) onto camera manipulations in OCCT 3D Viewer. This class supports standard mouse and touchscreen input layouts, cooperates with OCCT 3D Viewer update pipeline (including picking and selection) and supports two-thread approach (with separate threads for GUI and rendering), so it can be used for more rapid and robust integration of OCCT 3D Viewer into the application.

Draw Harness already benefits from migration onto **AIS_ViewController** by providing uniform user experience across platforms. The integration with particular GUI framework is, however, beyond the scope of this class - currently it was successfully used for redirecting user input from native APIs (WinAPI, Xlib, Cocoa) and in QtQuick application.

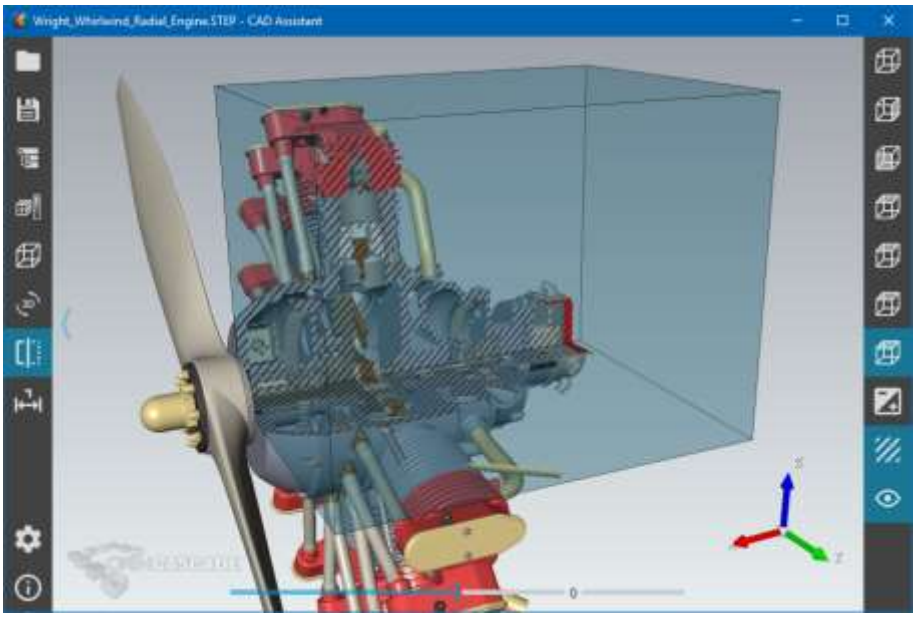
View Controller also utilizes information from picking services to provide a better user experience within perspective camera projection.

Complex clipping in 3D viewer

3D Viewer now supports Clipping Chains which define logical AND (conjunction) operation in addition to Clipping Planes defining logical OR (disjunction) supported previously. This enables providing interactive ¾ section (when ¼ of detail is cut off) and box section presentations. New feature is available through new **Graphic3d_ClipPlane** class properties - see documentation for methods **Graphic3d_ClipPlane::SetChainNextPlane()**.

In Draw Harness, clipping chains can be activated via command **vclipplane** like this:

```
> pload MODELING VISUALIZATION
> box b 20 40 20 100 20 30
> vinit
> vdisplay b -dispMode 1
> vclipplane pln -set -boxint 25 25 25 55 55 55
```



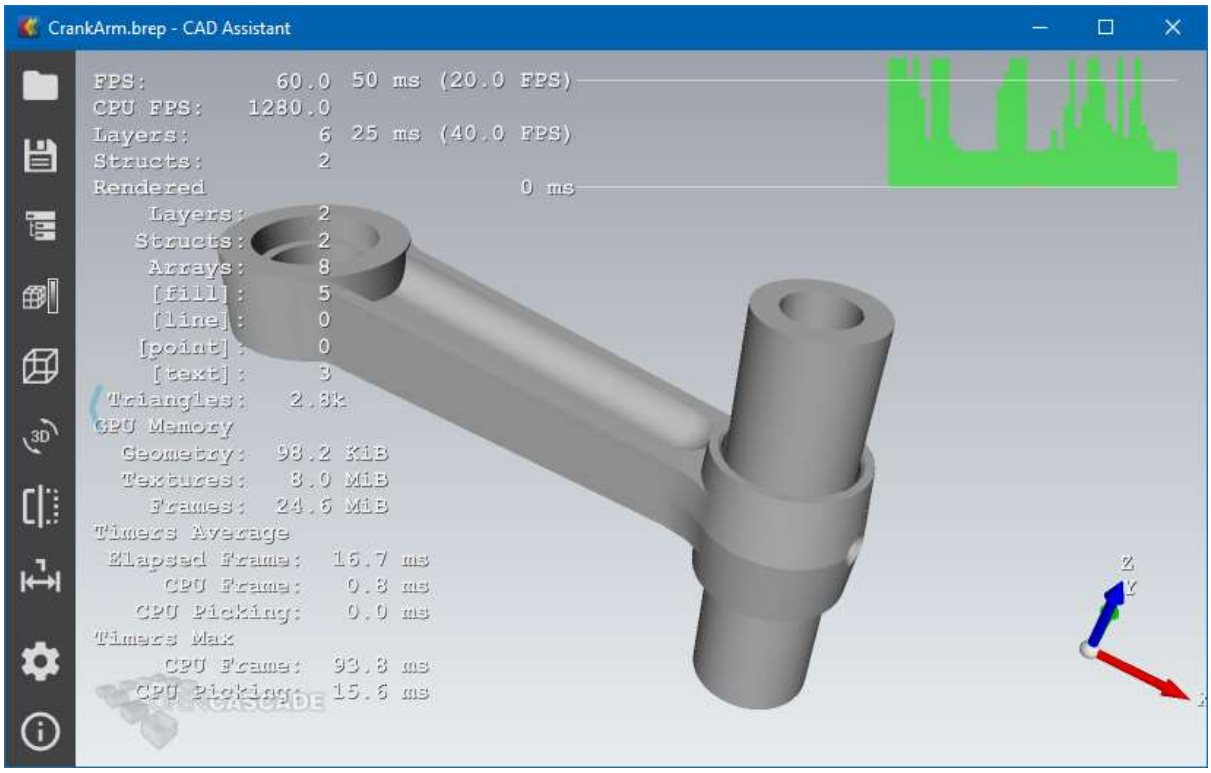
Tools for analysis of 3D viewer performance

Tools for profiling 3D Viewer performance integrated into OCCT have been extended. The tools now provide more counters to analyse and can display a chart with frame timings.

These tools can be helpful for optimization of performance of visualization pipeline in OCCT-based application. As they are integrated into 3D Viewer, they can be easily activated with minimal efforts - see **Graphic3d_RenderingParams::CollectedStats** and Graphic3d_RenderingParams::**ToShowStats**.

The onscreen performance counters can be activated in Draw Harness via command **vrenderparams** like this:

```
> pload VISUALIZATION
> vinit -w 1024 -h 512
> vviewcube vc
> vdisplay vc -trihedron bottomright 100 100
> vrenderparams -perfCounters full -perfChart 100
```



Dynamic display of shape outline

3D Viewer now supports new flag **Graphic3d_Aspects::SetDrawSilhouette()**, which allows including shape outline within Shaded presentation. Combination of this flag with **Aspect_IS_HIDDENLINE** interior style and suppressed seam edges face boundaries (see **Prs3d_Drawer::SetFaceBoundaryUpperContinuity()**) make it possible to achieve HLR-alike effect without expensive modelling algorithms.

New options are available within Draw Harness command **vaspects**, and can be used like this:

```
> pload MODELING VISUALIZATION
> box b 2 0 0 1 2 3
> psphere s 1
> vinit
> vsetcolorbg 255 255 255
```



```
> vdisplay -dispMode 1 b s
> vfit
> vaspects b s -setDrawSilhouette 1 -setEdgeColor BLACK
           -setFaceBoundaryDraw 1 -setMostContinuity c0
           -setFaceBoundaryColor BLACK -setInteriorStyle HIDDENLINE
```