Open CASCADE Technology 7.5.0

Upgrade from older OCCT versions

November 3, 2020

Contents

1 Introduction **5**
1.1 Precautions 5
1.2 Disclaimer 5

2 Upgrade to OCCT 6.5.0 . . . . . **6**

3 Upgrade to OCCT 6.5.1 . . . . . **7**

4 Upgrade to OCCT 6.5.2 . . . . . **8**

5 Upgrade to OCCT 6.5.3 . . . . . **9**

6 Upgrade to OCCT 6.5.4 . . . . . **10**

7 Upgrade to OCCT 6.6.0 . . . . . **11**

8 Upgrade to OCCT 6.7.0 . . . . . **13**
8.1 Object-level clipping and capping algorithm. 13
8.2 Redesign of markers presentation 13
8.3 Default views are not created automatically 13
8.4 Improved dimensions implementation 13
8.5 NCollection_Set replaced by List collection 14

9 Upgrade to OCCT 6.8.0 . . . . . **15**
9.1 Changes in NCollection classes 15
9.2 3D View Camera 15
9.3 Redesign of Connected Interactive Objects 15
9.4 Support of UNICODE Characters 16
9.5 Elimination of Projection Shift Concept 16

10 Upgrade to OCCT 6.9.0 . . . . . **17**
10.1 3D Viewer initialization 17
10.2 Changes in Selection 17
10.3 Changes in Adaptor3d_Curve class 18
10.4 Changes in V3d_View class 18

11 Upgrade to OCCT 7.0.0 . . . . . **19**
11.1 Removal of legacy persistence 19
11.2 Removal of CDL and WOK 20
11.2.1 Automatic upgrade 20
11.2.2 Possible compiler errors 22
11.2.3 Possible runtime problems 24

11.2.4 Option to avoid cast of handle to reference to base type . . . 24
11.2.5 Preserving compatibility with OCCT 6.x . . . 25
11.2.6 Applications based on CDL and WOK . . . 25
11.3 Separation of BSpline cache . . . 26
11.4 Structural result of Boolean operations . . . 26
11.5 BRepExtrema_ExtCC finds one solution only . . . 26
11.6 Removal of SortTools package . . . 26
11.7 On-screen objects and ColorScale . . . 27
11.8 UserDraw and Visual3d . . . 27
11.9 Deprecation of Local Context . . . 29
11.10 Separation of visualization part from TKCAF . . . 29
11.11 Correction of interpretation of Euler angles in gp_Quaternion . . . 29
11.12 Zoom Persistent Selection . . . 30
11.13 Texture mapping of objects . . . 30
11.14 Shape presentation builders . . . 30

12 Upgrade to OCCT 7.1.0 . . . 31
12.1 Presentation attributes . . . 31
12.2 Typedefs . . . 31
12.3 Programmable Pipeline . . . 31
12.4 Transformation persistence . . . 32
12.5 Dynamic highlight and selection properties . . . 32
12.6 Correction in TObj_Model class . . . 32
12.7 Redundant environment variables . . . 32
12.8 Removed features . . . 33
12.9 Other changes . . . 33

13 Upgrade to OCCT 7.2.0 . . . 35
13.1 Removed features . . . 35
13.2 Corrections in BRepOffset API . . . 35
13.3 Corrections in BRepOffset API . . . 36
13.4 Highlight style . . . 36
13.5 Elimination of implicit 3D Viewer updates . . . 37
13.6 Elimination of Quantity_NameOfColor from TKV3d interface classes . . . 37
13.7 Result of Boolean operations on containers . . . 38
13.8 Other changes . . . 38
13.9 BOP - Pairs of interfering indices . . . 38
13.10 Removal of the Draw commands based on old Boolean operations . . . 39
13.11 Change of Face/Face intersection in Boolean operations . . . 39
13.12 Restore OCCT 6.9.1 persistence . . . 39
13.13 Change in BRepLib_MakeFace algorithm . . . 40

13.14 Change in BRepFill_OffsetWire algorithm . . . 40
13.15 Change in Geom(2d)Adaptor_Curve::IsPeriodic . . . 41
13.16 Change in algorithm ShapeUpgrade_UnifySameDomain . . . 41
13.17 Changes in STL Reader / Writer . . . 41
13.18 Refactoring of the Error/Warning reporting system in Boolean Component . . . 41

14 Upgrade to OCCT 7.2.1 . . . 42
14.1 Changes in ShapeUpgrade_UnifySameDomain . . . 42
14.2 Moving BuildPCurveForEdgeOnPlane from BOPTools_AlgoTools2D to BRepLib . . . 42
14.3 Removed features . . . 42

15 Upgrade to OCCT 7.3.0 . . . 44
15.1 Light sources . . . 44
15.2 Shading Models . . . 44
15.3 Custom low-level OpenGL elements . . . 44
15.4 Changes in BOPAlgo_Section . . . 44
15.5 Changes in BRepAdaptor_CompCurve . . . 44
15.6 Removed features . . . 45
15.7 Boolean Operations - Solid Builder algorithm . . . 45
15.8 Boolean Operation classes in BRepAlgo are deprecated . . . 45
15.9 Unification of the Error/Warning reporting system of Application Framework . . . 45

16 Upgrade to OCCT 7.4.0 . . . 46
16.1 Changes in BRepPrimAPI_MakeRevol algorithm . . . 46
16.2 Removed features . . . 46
16.3 Local Context removal . . . 46
16.4 Changes in behavior of Convert algorithms . . . 46
16.5 Changes in selection API and picked point calculation algorithm. . . . 47
16.6 Document format version management improvement. . . . 47
16.7 BRepMesh - revision of the data model . . . 47
16.8 Changes in API of Chamfer algorithms . . . 48
16.9 Aspects unification . . . 48
16.10 Material definition . . . 48
16.11 Changes in Graphic3d_Text and OpenGl_Text API . . . 49
16.12 Presentation invalidation . . . 49
16.13 Interior styles . . . 50
16.14 PrsMgr and SelectMgr hierarchy clean up . . . 50
16.15 Polygon offset defaults . . . 50
16.16 Adding ZLayers in given position . . . 50
16.17 Modified enumerations . . . 50
16.18 Custom defines within env.bat . . . 51

16.19 Switching Boolean Operations algorithm to use BVH tree instead of UB tree . . . 51
16.20 Standard_Stream.hxx no more has "using std::" statements . . . 51

17 Upgrade to OCCT 7.5.0 . . . 52
17.1 RGB color definition . . . 52
17.2 Aspect_Window interface change . . . 52
17.3 Renaming of types . . . 52
17.4 change in construction of offset faces . . . 52
17.5 TKV3d/TKService toolkits changes . . . 52
17.6 Prs3d_Drawer deviation angle . . . 53
17.7 Changes in HLR presentation API . . . 53
17.8 Dimension and Relation presentations moved from AIS to PrsDim . . . 53
17.9 Select3D_SensitiveEntity interface change . . . 53
17.10 Changes in Boolean operations algorithm . . . 53
17.11 Offset direction change . . . 53
17.12 Change of progress indication API . . . 53
17.13 Message_Messenger interface change . . . 55
17.14 Message_Printer interface change . . . 56
17.15 Prs3d_Root deprecation . . . 56
17.16 Support of multiple OCAF application instances . . . 56
17.17 Draw Harness hotkeys . . . 56
17.18 Utf-8 encoding for message files . . . 56

1 Introduction

This document provides technical details on changes made in particular versions of OCCT. It can help to upgrade user applications based on previous versions of OCCT to newer ones.

1.1 Precautions

Back-up your code before the upgrade. We strongly recommend using version control system during the upgrade process and saving one or several commits at each step of upgrade, until the overall result is verified. This will facilitate identification and correction of possible problems that can occur at the intermediate steps of upgrade. It is advisable to document each step carefully to be able to repeat it if necessary.

1.2 Disclaimer

This document describes known issues that have been encountered during porting of OCCT and some applications and approaches that have helped to resolve these issues in known cases. It does not pretend to cover all possible migration issues that can appear in your application. Take this document with discretion; apply your expertise and knowledge of your application to ensure the correct result.

The automatic upgrade tool is provided as is, without warranty of any kind, and we explicitly disclaim any liability for possible errors that may appear due to use of this tool. It is your responsibility to ensure that the changes you made in your code are correct. When you upgrade the code by an automatic script, make sure to carefully review the introduced changes at each step before committing them.

2 Upgrade to OCCT 6.5.0

Porting of user applications from an earlier OCCT version to version 6.5 requires taking into account the following major changes:

- If you are not comfortable with dependence on Intel TBB, FreeImage, or Gl2Ps libraries, you will need to (re)build OCCT with these dependencies disabled.
- The low-level format version of OCAF binary and XML persistence has been incremented. Hence, the files saved by OCCT 6.5 to OCAF binary or XML format will not be readable by previous versions of OCCT.
- The *BRepMesh* triangulation algorithm has been seriously revised and now tries hard to fulfill the requested deflection and angular tolerance parameters. If you experience any problems with performance or triangulation quality (in particular, display of shapes in shading mode), consider revising the values of these parameters used in your application.
- If you were using method *ToPixMap()* of class *V3d_View* to get a buffer for passing to Windows API functions (e.g. *BitBlt*), this will not work anymore. You will need to use method *Image_PixMap::AccessBuffer()* to get the raw buffer data that can be further passed to WinAPI functions.
- As the processing of message gravity parameter in *Message* package has been improved, some application messages (especially the ones generated by IGES or STEP translators) can be suppressed or new messages appear in the application. Use relevant message level parameter to tune this behavior.

3 Upgrade to OCCT 6.5.1

Porting of user applications from an earlier OCCT version to version 6.5.1 requires taking into account the following major changes:

- Method *Graphic3d_Structure::Groups()* now returns *Graphic3d_SequenceOfGroup*. If this method has been used, the application code should be updated to iterate another collection type or, if *Graphic3d_HSetOfGroup* is required, to fill its own collection:

```
const Graphic3d_SequenceOfGroup& aGroupsSeq = theStructure.Groups();
Handle(Graphic3d_HSetOfGroup) aGroupSet = new Graphic3d_HSetOfGroup();
Standard_Integer aLen = aGroupsSeq.Length();
for (Standard_Integer aGr = 1; aGr <= aLen; ++aGr)
{
 aGroupSet->Add (aGroupsSeq.Value (aGr));
}
```

- All occurrences of *Select3D_Projector* in the application code (if any) should be replaced with *Handle(↩ Select3D_Projector)*.
- The code of inheritors of *Select3D_SensitiveEntity* should be updated if they override *Matches()* (this is probable, if clipping planes are used).
- Constructor for *V3d_Plane* has been changed, so the extra argument should be removed if used in the application. It is necessary to add a new plane using method *V3d_Viewer::AddPlane()* if *V3d_Viewer* has been used to manage clipping planes list (this does not affect clipping planes representation). Have a look at the source code for new DRAWEXE *vclipplane* command in *ViewerTest_ObjectsCommands.cxx, VClipPlane* to see how clipping planes can be managed in the application.

4 Upgrade to OCCT 6.5.2

Porting of user applications from an earlier OCCT version to version 6.5.2 requires taking into account the following major changes:

- Any code that has been generated by WOK from CDL generic classes *Tcollection_DataMap* and *Tcollection_IndexedDataMap* needs to be regenerated by WOK to take into account the change in the interface of these classes.
- The enumerations *CDF_StoreStatus* and *CDF_RetrievableStatus* have been replaced by the enumerations *PCDM_StoreStatus* and *PCDM_ReaderStatus*. Correspondingly, the methods *Open, Save* and *SaveAs* of the class *TDocStd_Application* have changed their return value. Any code, which uses these enumerations, needs to be updated.
- *BRepLib_MakeFace* has been modified to receive tolerance value for resolution of degenerated edges. This tolerance parameter has no default value to ensure that the client code takes care of passing a meaningful value, not just *Precision::Confusion*, so some porting overheads are expected.
- If the callback mechanism in call_togl_redraw function was used in the application code, it is necessary to revise it to take into account the new callback execution and provide a check of reason value of Aspect_GraphicCallbackStruct in callback methods to confirm that the callback code is executed at the right moment. Now the callbacks are executed before redrawing the underlayer, before redrawing the overlayer and at the end of redrawing. The information about the moment when the callback is invoked is provided with the reason value in form of an additional bit flag *(OCC_PRE_REDRAW, OCC_PRE_OVERLAY)*. The state of OpenGl changed in callback methods will not be restored automatically, which might lead to unwanted behavior in redrawing procedure.
- The print method used in the application code might need to be revised to take into account the ability to choose between print algorithms: tile and stretch. The stretch algorithm will be selected by default during porting.
- It is recommended to *BRepMesh_DiscretFactory* users, to check *BRepMesh_DiscretFactory::SetDefault()* return value to determine plugin availability / validity. *BRepMesh_DiscretFactory::Discret()* method now returns handle instead of pointer. The code should be updated in the following manner:

```
Handle(BRepMesh_DiscretRoot) aMeshAlgo = BRepMesh_DiscretFactory::Get().Discret (theShape, theDeflection,
       theAngularToler);
 if (!aMeshAlgo.IsNull())  {}
```

- The default state of *BRepMesh* parallelization has been turned off. The user should switch this flag explicitly:
  - by using methods *BRepMesh_IncrementalMesh::SetParallel(Standard_True)* for each *BRepMesh_IncrementalMesh* instance before *Perform()*;
  - by calling *BRepMesh_IncrementalMesh::SetParallelDefault(Standard_True)* when *BRepMesh_DiscretFactory* is used to retrieve the meshing tool (this also affects auto-triangulation in *AIS*).

5 Upgrade to OCCT 6.5.3

Porting of user applications from an earlier OCCT version to version 6.5.3 requires taking into account the following major changes:

- As a result of code clean-up and redesign of *TKOpenGl* driver, some obsolete functions and rendering primitives *(TriangleMesh, TriangleSet, Bezier, Polyline, Polygon, PolygonHoles, QuadrangleMesh* and *QuadrangleSet*) have been removed. Instead, the application developers should use primitive arrays that provide the same functionality but are hardware-accelerated. The details can be found in OCCT Visualization User's Guide, “Primitive Arrays” chapter.
- Applications should not call *AIS_InteractiveObject::SetPolygonOffsets()* method for an instance of *AIS_TexturedShape* class after it has been added to *AIS_InteractiveContext*. More generally, modification of *Graphic3d_AspectFillArea3d* parameters for the computed groups of any *AIS_InteractiveObject* subclass that uses texture mapping should be avoided, because this results in broken texture mapping (see issue 23118). It is still possible to apply non-default polygon offsets to *AIS_TexturedShape* by calling *SetPolygonOffsets()* before displaying the shape.
- The applications that might have used internal functions provided by *TKOpenGl* or removed primitives will need to be updated.
- In connection with the implementation of Z-layers it might be necessary to revise the application code or revise the custom direct descendant classes of *Graphic3d_GraphicDriver* and *Graphic3d_StructureManager* to use the Z-layer feature.
- Global variables *Standard_PI* and *PI* have been eliminated (use macro *M_PI* instead).
- Method *HashCode()* has been removed from class *Standard_Transient*. It is advisable to use global function *HashCode()* for Handle objects instead.
- Declaration of operators new/delete for classes has become consistent and is encapsulated in macros.
- Memory management has been changed to use standard heap *(MMGT_OPT=0)* and reentrant mode *(MMGT_REENTRANT=1)* by default.
- Map classes in *NCollection* package now receive one more argument defining a hash tool.

6 Upgrade to OCCT 6.5.4

Porting of user applications from an earlier OCCT version to version 6.5.4 requires taking into account the following major changes:

- The code using obsolete classes *Aspect_PixMap, Xw_PixMap* and *WNT_PixMap* should be rewritten implementing class *Image_PixMap*, which is now retrieved by *ToPixMap* methods as argument. A sample code using *ToPixMap* is given below:

```
#include <Image_AlienPixMap.hxx>
void dump (Handle(V3d_View)& theView3D)
{
  Standard_Integer aWndSizeX = 0;
  Standard_Integer aWndSizeY = 0;
  theView3D->Window()->Size (aWndSizeX, aWndSizeY);
  Image_AlienPixMap aPixMap;
  theView3D->ToPixMap (aPixMap, aWndSizeX, aWndSizeY);
  aPixMap.Save ("c:\\image.png");
}
```

- Now OpenGL resources related to Interactive Objects are automatically freed when the last view (window) is removed from graphical driver. To avoid presentation data loss, the application should replace an old view with a new one in the proper order: first the new view is created and activated and only then the old one is detached and removed.

- It is recommended to use *NCollection* containers with hasher parameter (introduced in 6.5.3) instead of global definition *IsEqual()/HashCode()* as well as to use explicit namespaces to avoid name collision.

7 Upgrade to OCCT 6.6.0

Porting of user applications from an earlier OCCT version to version 6.6.0 requires taking into account the following major changes:

- Due to the changes in the implementation of Boolean Operations, the order of sub-shapes resulting from the same operation performed with OCCT 6.5.x and OCCT 6.6.0 can be different. It is necessary to introduce the corresponding changes in the applications for which the order of sub-shapes resulting from a Boolean operation is important. It is strongly recommended to use identification methods not relying on the order of sub-shapes (e.g. OCAF naming).
- If you need to use OCCT on Mac OS X with X11 (without Cocoa), build OCCT with defined pre-processor macro *CSF_MAC_USE_GLX11*. XLib front-end (previously the only way for unofficial OCCT builds on Mac OS X) is now disabled by default on this platform. If your application has no support for Cocoa framework you may build OCCT with XLib front-end adding *MACOSX_USE_GLX* macro to compiler options (you may check the appropriate option in WOK configuration GUI and in CMake configuration). Notice that XQuartz (XLib implementation for Mac OS X) now is an optional component and does not provide a sufficient level of integrity with native (Cocoa-based) applications in the system. It is not possible to build OCCT with both XLib and Cocoa at the same time due to symbols conflict in OpenGL functions.
- Animation mode and degeneration presentation mode (simplified presentation for animation) and associated methods have been removed from 3D viewer functionality. Correspondingly, the code using methods *SetAnimationModeOn(), SetAnimationModeOff(), AnimationModeIsOn(), AnimationMode(), Tumble(), SetDegenerateModeOn(), SetDegenerateModeOff()* and *DegenerateModeIsOn()* of classes *V3d_View* and *Visual3d_View* will need to be removed or redesigned. Hidden Line Removal presentation was not affected; however, the old code that used methods *V3d_View::SetDegenerateModeOn* or *V3d_View::SetDegenerate↩ ModeOff* to control HLR presentation should be updated to use *V3d_View::SetComputedMode* method instead.
- Calls of *Graphic3d_Group::BeginPrimitives()* and *Graphic3d_Group::EndPrimitives()* should be removed from the application code.
- Application functionality for drawing 2D graphics that was formerly based on *TKV2d* API should be migrated to *TKV3d* API. The following changes are recommended for this migration:
  - A 2D view can be implemented as a *V3d_View* instance belonging to *V3d_Viewer* managed by *AIS↩ _InteractiveContext* instance. To turn *V3d_View* into a 2D view, the necessary view orientation should be set up at the view initialization stage using *V3d_View::SetProj()* method, and view rotation methods simply should not be called.
  - Any 2D graphic entity (formerly represented with *AIS2D_InteractiveObject*) should become a class derived from *AIS_InteractiveObject* base. These entities should be manipulated in a view using *AIS_↩ InteractiveContext* class API.
  - All drawing code should be put into *Compute()* virtual method of a custom interactive object class and use API of *Graphic3d* package. In particular, all geometry should be drawn using class hierarchy derived from *Graphic3d_ArrayOfPrimitives*. Normally, the Z coordinate for 2D geometry should be constant, unless the application implements some advanced 2D drawing techniques like e.g. multiple "Z layers" of drawings.
  - Interactive selection of 2D presentations should be set up inside *ComputeSelection()* virtual method of a custom interactive object class, using standard sensitive entities from *Select3D* package and standard or custom entity owners derived from *SelectMgr_EntityOwner* base. Refer to the Visualization User's Guide for further details concerning OCCT 3D visualization and selection classes. See also *Viewer2D* OCCT sample application, which shows how 2D drawing can be implemented using TKV3d API.
- Run-time graphic driver library loading mechanism based on *CSF_GraphicShr* environment variable usage has been replaced by explicit linking against *TKOpenGl* library. The code sample below shows how the graphic driver should be created and initialized in the application code:

```
// initialize a new viewer with OpenGl graphic driver
Handle(Graphic3d_GraphicDriver) aGraphicDriver =
new OpenGl_GraphicDriver ("TKOpenGl");
```

```
  aGraphicDriver->Begin (new Aspect_DisplayConnection());
  TCollection_ExtendedString aNameOfViewer ("Visu3D");
  Handle(V3d_Viewer) aViewer
= new V3d_Viewer (aGraphicDriver, aNameOfViewer.ToExtString());
  aViewer->Init();

// create a new window or a wrapper over the existing window,
// provided by a 3rd-party framework (Qt, MFC, C# or Cocoa)
#if defined(_WIN32) || defined(__WIN32__)
  Aspect_Handle aWindowHandle = (Aspect_Handle )winId();
  Handle(WNT_Window) aWindow = new WNT_Window (winId());
#elif defined(__APPLE__) && !defined(MACOSX_USE_GLX)
  NSView* aViewHandle = (NSView* )winId();
  Handle(Cocoa_Window) aWindow = new Cocoa_Window (aViewHandle);
#else
 Aspect_Handle aWindowHandle = (Aspect_Handle )winId();
  Handle(Xw_Window) aWindow =
     new Xw_Window (aGraphicDriver->GetDisplayConnection(), aWindowHandle);
#endif // WNT

// setup the window for a new view
  Handle(V3d_View) aView = aViewer->CreateView();
  aView->SetWindow (aWindow);
```

- The following changes should be made in the application-specific implementations of texture aspect:
  - *Graphic3d_TextureRoot* inheritors now should return texture image by overloading of *Graphic3d_↩ TextureRoot::GetImage()* method instead of the old logic.
  - Now you can decide if the application should store the image copy as a field of property or reload it dynamically each time (to optimize the memory usage). The default implementation (which loads the image content from the provided file path) does not hold an extra copy since it will be uploaded to the graphic memory when first used.
  - Notice that the image itself should be created within *Image_PixMap* class from *AlienImage* package, while *Image_Image* class is no more supported and will be removed in the next OCCT release.

8 Upgrade to OCCT 6.7.0

Porting of user applications from an earlier OCCT version to version 6.7.0 requires taking into account the following major changes.

8.1 Object-level clipping and capping algorithm.

- It might be necessary to revise and port code related to management of view-level clipping to use *Graphic3d_ClipPlane* instead of *V3d_Plane* instances. Note that *V3d_Plane* class has been preserved – as previously, it can be used as plane representation. Another approach to represent *Graphic3d_ClipPlane* in a view is to use custom presentable object.
- The list of arguments of *Select3D_SensitiveEntity::Matches()* method for picking detection has changed. Since now, for correct selection clipping, the implementations should perform a depth clipping check and return (as output argument) minimum depth value found at the detected part of sensitive. Refer to CDL / Doxygen documentation to find descriptive hints and snippets.
- *Select3D_SensitiveEntity::ComputeDepth()* abstract method has been removed. Custom implementations should provide depth checks by method *Matches()* instead – all data required for it is available within a scope of single method.
- It might be necessary to revise the code of custom sensitive entities and port *Matches()* and *ComputeDepth()* methods to ensure proper selection clipping. Note that obsolete signature of *Matches* is not used anymore by the selector. If your class inheriting *Select3D_SensitiveEntity* redefines the method with old signature the code should not compile as the return type has been changed. This is done to prevent override of removed methods.

8.2 Redesign of markers presentation

- Due to the redesign of *Graphic3d_AspectMarker3d* class the code of custom markers initialization should be updated. Notice that you can reuse old markers definition code as *TColStd_HArray1OfByte*; however, *Image_PixMap* is now the preferred way (and supports full-color images on modern hardware).
- Logics and arguments of methods *AIS_InteractiveContext::Erase()* and *AIS_InteractiveContext::EraseAll()* have been changed. Now these methods do not remove resources from *Graphic3d_Structure*; they simply change the visibility flag in it. Therefore, the code that deletes and reomputes resources should be revised.
- *Graphic3d_Group::MarkerSet()* has been removed. *Graphic3d_Group::AddPrimitiveArray()* should be used instead to specify marker(s) array.

8.3 Default views are not created automatically

As the obsolete methods *Init(), DefaultOrthographicView()* and *DefaultPerspectiveView()* have been removed from *V3d_Viewer* class, the two default views are no longer created automatically. It is obligatory to create *V3d_View* instances explicitly, either directly by operator new or by calling *V3d_Viewer::CreateView()*.

The call *V3d_Viewer::SetDefaultLights()* should also be done explicitly at the application level, if the application prefers to use the default light source configuration. Otherwise, the application itself should set up the light sources to obtain a correct 3D scene.

8.4 Improved dimensions implementation

- It might be necessary to revise and port code related to management of *AIS_LengthDimension, AIS_AngleDimension* and *AIS_DiameterDimension* presentations. There is no more need to compute value of dimension and pass it as string to constructor argument. The value is computed internally. The custom value can be set with *SetCustomValue()* method.

- The definition of units and general aspect properties is now provided by *Prs3d_DimensionUnits* and *Prs3d_DimensionApsect* classes.
- It might be also necessary to revise code of your application related to usage of *AIS_DimensionDisplayMode enumeration*. If it used for specifying the selection mode, then it should be replaced by a more appropriate enumeration *AIS_DimensionSelectionMode*.

8.5 NCollection_Set replaced by List collection

It might be necessary to revise your application code, which uses non-ordered *Graphic3d_SetOfHClipPlane* collection type and replace its occurrences by ordered *Graphic3d_SequenceOfHClipPlane* collection type.

9 Upgrade to OCCT 6.8.0

Porting of user applications from an earlier OCCT version to version 6.8.0 requires taking into account the following major changes.

9.1 Changes in NCollection classes

Method *Assign()* in *NCollection* classes does not allow any more copying between different collection types. Such copying should be done manually.

List and map classes in *NCollection* package now require that their items be copy-constructible, but do not require items to have default constructor. Thus the code using *NCollection* classes for non-copy-constructible objects needs be updated. One option is to provide copy constructor; another possibility is to use Handle or other smart pointer.

9.2 3D View Camera

If *ViewMapping* and *ViewOrientation* were used directly, this functionality has to be ported to the new camera model. The following methods should be considered as an alternative to the obsolete *Visual3d* services (all points and directions are supposed to be in world coordinates):

- *Graphic3d_Camera::ViewDimensions()* or *V3d_View::Size()/ZSize()* – returns view width, height and depth (or "Z size"). Since the view is symmetric now, you can easily compute top, bottom, left and right limits. *Graphic3d_Camera::ZNear()/ZFar()* can be used to obtain the near and far clipping distances with respect to the eye.
- *Graphic3d_Camera::Up()* or *V3d_View::Up()* – returns Y direction of the view.
- *Graphic3d_Camera::Direction()* returns the reverse view normal directed from the eye, *V3d_View::Proj()* returns the old-style view normal.
- *Graphic3d_Camera::Eye()* or *V3d_View::Eye()* – returns the camera position (same as projection reference point in old implementation).
- *Graphic3d_Camera::Center()* or *V3d_View::At()* – returns the point the camera looks at (or view reference point according to old terminology).

The current perspective model is not fully backward compatible, so the old perspective-related functionality needs to be reviewed.

Revise application-specific custom presentations to provide a proper bounding box, otherwise the object might become erroneously clipped by automatic *ZFit* or frustum culling algorithms enabled by default.

9.3 Redesign of Connected Interactive Objects

The new implementation of connected Interactive Objects makes it necessary to take the following steps if you use connected Interactive Objects in your application.

- Use new *PrsMgr_PresentableObject* transformation API.
- Call *RemoveChild()* from the original object after connect if you need the original object and *AIS_ConnectedInteractive* to move independently.
- Access instances of objects connected to *AIS_MultiplyConnectedInteractive* with *Children()* method.
- For *PrsMgr_PresentableObject* transformation:
  - *SetLocation (TopLoc_Location) -> SetLocalTransformation (gp_Trsf)*
  - *Location -> LocalTransformation*
  - *HasLocation -> HasTransformation*
  - *ResetLocation -> ResetTransformation*

9.4 Support of UNICODE Characters

Support of UNICODE characters introduced in OCCT breaks backward compatibility with applications, which currently use filenames in extended ASCII encoding bound to the current locale. Such applications should be updated to convert such strings to UTF-8 format.

The conversion from UTF-8 to wchar_t is made using little-endian approach. Thus, this code will not work correctly on big-endian platforms. It is needed to complete this in the way similar as it is done for binary persistence (see the macro *DO_INVERSE* in *FSD_FileHeader.hxx).*

9.5 Elimination of Projection Shift Concept

It might be necessary to revise the application code, which deals with *Center()* method of *V3d_View*.

This method was used to pan a *V3d* view by virtually moving the screen center with respect to the projection ray passed through Eye and At points. There is no more need to derive the panning from the Center parameter to get a camera-like eye position and look at the coordinates. *Eye()* and *At()* now return these coordinates directly. When porting code dealing with *Center()*, the parameters *Eye()* and *At()* can be adjusted instead. Also *V3d_View::Set↩ Center(Xpix, Ypix)* method can be used instead of *V3d_View::Center(X, Y)* to center the view at the given point. However, if the center coordinates X and Y come from older OCCT releases, calling *V3d_View::Panning(-X, -Y)* can be recommended to compensate missing projection shift effect.

There are several changes introduced to *Graphic3d_Camera*. The internal data structure of the camera is based on *Standard_Real* data types to avoid redundant application-level conversions and precision errors. The transformation matrices now can be evaluated both for *Standard_Real* and *Standard_ShortReal* value types. *ZNear* and *Z↩ Far* planes can be either negative or positive for orthographic camera projection, providing a trade-off between the camera distance and the range of *ZNear* or *ZFar* to reduce difference of exponents of values composing the orientation matrix - to avoid calculation errors. The negative values can be specified to avoid Z-clipping if the reference system of camera goes inside of the model when decreasing camera distance.

The auto z fit mode, since now, has a parameter defining Z-range margin (the one which is usually passed as argument to *ZFitAll()* method). The methods *SetAutoZFitMode(), AutoZFitScaleFactor()* and *ZFitAll()* from class *V3d_View* deal with the new parameter.

The class *Select3D_Projector* now supports both orientation and projection transformation matrices, which can be naturally set for the projector. The definition of projector was revised in *StdSelect_ViewerSelector3d*: perspective and orthographic projection parameters are handled properly. Orthographic projector is based only on direction of projection - no more *Center* property. This makes it possible to avoid unnecessary re-projection of sensitive while panning, zooming or moving along the projection ray of the view. These operations do not affect the orthographic projection.

10 Upgrade to OCCT 6.9.0

Porting of user applications from an earlier OCCT version to version 6.9.0 requires taking into account the following major changes.

10.1 3D Viewer initialization

3D Viewer now uses GLSL programs for managing frame buffer and stereoscopic output. For proper initialization, application should configure **CSF_ShadersDirectory** environment variable pointing to a folder with GLSL resources - files from folder **CASROOT**/src/Shaders. *Note that **CSF_ShadersDirectory** become optional since OCCT 7.1.0 release*.

10.2 Changes in Selection

Selection mechanism of 3D Viewer has been redesigned to use 3-level BVH tree traverse directly in 3D space instead of projection onto 2D screen space (updated on each rotation). This architectural redesign may require appropriate changes at application level in case if custom Interactive Objects are used.

Standard selection

Usage of standard OCCT selection entities would require only minor updates.

Custom Interactive Objects should implement new virtual method *SelectMgr_SelectableObject::BoundingBox()*.

Now the method *SelectMgr_Selection::Sensitive()* does not return *SelectBasics_SensitiveEntity*. It returns an instance of *SelectMgr_SensitiveEntity*, which belongs to a different class hierarchy (thus *DownCast()* will fail). To access base sensitive it is necessary to use method *SelectMgr_SensitiveEntity::BaseSensitive()*. For example:

```
Handle(SelectMgr_Selection) aSelection = anInteractiveObject->Selection (aMode);
for (aSelection->Init(); aSelection->More(); aSelection->Next())
{
   Handle(SelectBasics_SensitiveEntity) anEntity = aSelection->Sensitive()->BaseSensitive();
}
```

Custom sensitive entities

Custom sensitive entities require more complex changes, since the selection algorithm has been redesigned and requires different output from the entities.

The method *SelectBasics_SensitiveEntity::Matches()* of the base class should be overridden following the new signature:

Standard_Boolean Matches (SelectBasics_SelectingVolumeManager& theMgr, SelectBasics_PickResult& thePickResult), where *theMgr* contains information about the currently selected frustum or set of frustums (see *SelectMgr_RectangularFrustum, SelectMgr_TrangularFrustum, SelectMgr_TriangularFrustumSet)* and *SelectBasics_PickResult* is an output parameter, containing information about the depth of the detected entity and distance to its center of geometry.

In the overridden method it is necessary to implement an algorithm of overlap and inclusion detection (the active mode is returned by *theMgr.IsOverlapAllowed()*) with triangular and rectangular frustums.

The depth and distance to the center of geometry must be calculated for the 3D projection of user-picked screen point in the world space. You may use already implemented overlap and inclusion detection methods for different primitives from *SelectMgr_RectangularFrustum* and *SelectMgr_TriangularFrustum*, including triangle, point, axis-aligned box, line segment and planar polygon.

Here is an example of overlap/inclusion test for a box:

```
if (!theMgr.IsOverlapAllowed()) // check for inclusion
{
  Standard_Boolean isInside = Standard_True;
  return theMgr.Overlaps (myBox.CornerMin(), myBox.CornerMax(), &isInside) && isInside;
}
```

```
Standard_Real aDepth;
if (!theMgr.Overlaps (myBox, aDepth)) // check for overlap
{
  return Standard_False;
}

thePickResult =
SelectBasics_PickResult (aDepth, theMgr.DistToGeometryCenter (myCenter3d));
```

The interface of *SelectBasics_SensitiveEntity* now contains four new pure virtual functions that should be implemented by each custom sensitive:

- *BoundingBox()* – returns a bounding box of the entity;
- *Clear()* – clears up all the resources and memory allocated for complex sensitive entities;
- *BVH()* – builds a BVH tree for complex sensitive entities, if it is needed;
- *NbSubElements()* – returns atomic sub-entities of a complex sensitive entity, which will be used as primitives for BVH building. If the entity is simple and no BVH is required, this method returns 1.

Each sensitive entity now has its own tolerance, which can be overridden by method *SelectBasics_SensitiveEntity::SetSensitivityFactor()* called from constructor.

10.3 Changes in Adaptor3d_Curve class

All classes inheriting *Adaptor3d_Curve* (directly or indirectly) must be updated in application code to use new signature of methods *Intervals()* and *NbIntervals()*. Note that no compiler warning will be generated if this is not done.

10.4 Changes in V3d_View class

The methods *V3d_View::Convert* and *V3d_View::ConvertWithProj()* have ceased to return point on the active grid. It might be necessary to revise the code of your application so that *V3d_View::ConvertToGrid()* was called explicitly for the values returned by *V3d_View::Convert* to get analogous coordinates on the grid. The methods *V3d_View::Convert* and *V3d_View::ConvertWithProj* convert point into reference plane of the view corresponding to the intersection with the projection plane of the eye/view point vector.

11 Upgrade to OCCT 7.0.0

Porting of user applications from an earlier OCCT version to version 7.0.0 requires taking into account the following major changes.

Building OCCT now requires compiler supporting some C++11 features. The supported compilers are:

- MSVC: version 10 (Visual Studio 2010) or later
- GCC: version 4.3 or later
- CLang: version 3.6 or later
- ICC: version XE 2013 SP 1 or later

When compiling code that uses OCCT with GCC and CLang compilers, it is necessary to use compiler option -std=c++0x (or its siblings) to enable C++11 features.

11.1 Removal of legacy persistence

Legacy persistence for shapes and OCAF data based on *Storage_Schema* (toolkits *TKPShape*, *TKPLCAF*, *TKPCAF*, *TKShapeShcema, TLStdLSchema, TKStdSchema*, and *TKXCAFSchema*) has been removed in OCCT 7.0.0. The applications that used these data persistence tools need to be updated to use other persistence mechanisms.

Note

For compatibility with previous versions, the possibility to read standard OCAF data (*TKLCAF* and *TKCAF*) from files stored in the old format is preserved (toolkits *TKStdL* and *TKStd*).

The existing data files in standard formats can be converted using OCCT 6.9.1 or a previous version, as follows.

Note

Reading / writing custom files capability from OCCT 6.9.1 is restored in OCCT 7.2.0. See details in Restore OCCT 6.9.1 persistence section.

CSFDB files

Files in *CSFDB* format (usually with extension .csfdb) contain OCCT shape data that can be converted to BRep format. The easiest way to do that is to use ImportExport sample provided with OCCT 6.9.0 (or earlier):

- Start ImportExport sample;
- Select File / New;
- Select File / Import / CSFDB... and specify the file to be converted;
- Drag the mouse with the right button pressed across the view to select all shapes by the rectangle;
- Select File / Export / BREP... and specify the location and name for the resulting file

OCAF and XCAF documents

Files containing OCAF data saved in the old format usually have extensions *.std, .sgd* or *.dxc* (XDE documents). These files can be converted to XML or binary OCAF formats using DRAW Test Harness commands. Note that if the file contains only attributes defined in *TKLCAF* and *TKCAF*, this action can be performed in OCCT 7.0; otherwise OCCT 6.9.1 or earlier should be used.

For that, start *DRAWEXE* and perform the following commands:

- To convert ∗*.std* and ∗*.sgd* file formats to binary format ∗*.cbf* (The created document should be in *BinOcaf* format instead of *MDTV-Standard*):

```
Draw[]> pload ALL
Draw[]> Open [path to *.std or *.sgd file] Doc
Draw[]> Format Doc BinOcaf
Draw[]> SaveAs Doc [path to the new file]
```

- To convert ∗*.dxc* file format to binary format ∗*.xbf* (The created document should be in *BinXCAF* format instead of *MDTV-XCAF*):

```
Draw[]> pload ALL
Draw[]> XOpen [path to *.dxc file] Doc
Draw[]> Format Doc BinXCAF
Draw[]> XSave Doc [path to the new file]
```

On Windows, it is necessary to replace back slashes in the file path by direct slashes or pairs of back slashes.

Use *XmlOcaf* or *XmlXCAF* instead of *BinOcaf* and *BinXCAF*, respectively, to save in XML format instead of binary one.

11.2 Removal of CDL and WOK

OCCT code has been completely refactored in version 7.0 to get rid of obsolete technologies used since its inception: CDL (Cas.Cade Definition Language) and WOK (Workshop Organization Kit).

C++ code previously generated by WOK from CDL declarations is now included directly in OCCT sources.

This modification did not change names, API, and behavior of existing OCCT classes, thus in general the code based on OCCT 6.x should compile and work fine with OCCT 7.0. However, due to redesign of basic mechanisms (CDL generic classes, Handles and RTTI) using C++ templates, some changes may be necessary in the code when porting to OCCT 7.0, as described below.

WOK is not necessary anymore for building OCCT from sources, though it still can be used in a traditional way – auxiliary files required for that are preserved. The recommended method for building OCCT 7.x is CMake, see build_occt_win_cmake. The alternative solution is to use project files generated by OCCT legacy tool **genproj**, see build_occt_genproj, build_occt_win_codeblocks, and build_occt_macos_xcode.

11.2.1 Automatic upgrade

Most of typical changes required for upgrading code for OCCT 7.0 can be done automatically using the *upgrade* tool included in OCCT 7.0. This tool is a Tcl script, thus Tcl should be available on your workstation to run it.

Example:

```
$ tclsh
% source <path_to_occt>/adm/upgrade.tcl
% upgrade -recurse -all -src=<path_to_your_sources>
```

On Windows, the helper batch script *upgrade.bat* can be used, provided that Tcl is either available in *PATH*, or configured via *custom.bat* script (for instance, if you use OCCT installed from Windows installer package). Start it from the command prompt:

```
cmd> <path_to_occt>\upgrade.bat -recurse -all -inc=<path_to_occt>\inc -src=<path_to_your_sources> [options]
```

Run the upgrade tool without arguments to see the list of available options.

The upgrade tool performs the following changes in the code.

1. Replaces macro *DEFINE_STANDARD_RTTI* by *DEFINE_STANDARD_RTTIEXT*, with second argument indicating base class for the main argument class (if inheritance is recognized by the script):

   ```
   DEFINE_STANDARD_RTTI(Class) -> DEFINE_STANDARD_RTTIEXT(Class, Base)
   ```

Note

If macro *DEFINE_STANDARD_RTTI* with two arguments (used in intermediate development versions of OCCT 7.0) is found, the script will convert it to either *DEFINE_STANDARD_RTTIEXT* or *DEFINE_STANDARD_RTTI_INLINE*. The former case is used if current file is header and source file with the same name is found in the same folder. In this case, macro *IMPLEMENT_STANDARD_RTTI* is injected in the corresponding source file. The latter variant defines all methods for RTTI as inline, and does not require *IMPLEMENT_STANDARD_RTTIEXT* macro.

2. Replaces forward declarations of collection classes previously generated from CDL generics (defined in *TCollection* package) by inclusion of the corresponding header:

```
class TColStd_Array1OfReal; -> #include <TColStd_Array1OfReal.hxx>
```

3. Replaces underscored names of *Handle* classes by usage of a macro:

```
Handle_Class -> Handle(Class)
```

This change is not applied if the source or header file is recognized as containing the definition of Qt class with signals or slots, to avoid possible compilation errors of MOC files caused by inability of MOC to recognize macros (see https://doc.qt.io/qt-4.8/signalsandslots.html). The file is considered as defining a Qt object if it contains strings *Q_OBJECT* and either *slots:* or *signals:*.

4. Removes forward declarations of classes with names *Handle(C)* or *Handle_C*, replacing them either by forward declaration of its argument class, or (for files defining Qt objects) *#include* statement for a header with the name of the argument class and extension .hxx:

```
class Handle(TColStd_HArray1OfReal); -> #include <TColStd_HArray1OfReal.hxx>
```

5. Removes *#includes* of files *Handle_...hxx* that have disappeared in OCCT 7.0:

```
#include <Handle_Geom_Curve.hxx> ->
```

6. Removes *typedef* statements that use *Handle* macro to generate the name:

```
typedef NCollection_Handle<Message_Msg> Handle(Message_Msg); ->
```

7. Converts C-style casts applied to Handles into calls to *DownCast()* method:

```
((Handle(A)&)b)      -> Handle(A)::DownCast(b)
(Handle(A)&)b        -> Handle(A)::DownCast(b)
(*((Handle(A)*)&b)) -> Handle(A)::DownCast(b)
*((Handle(A)*)&b)    -> Handle(A)::DownCast(b)
(*(Handle(A)*)&b)    -> Handle(A)::DownCast(b)
```

8. Moves *Handle()* macro out of namespace scope:

```
Namespace::Handle(Class) -> Handle(Namespace::Class)
```

9. Converts local variables of reference type, which are initialized by a temporary object returned by call to *DownCast()*, to the variables of non-reference type (to avoid using references to destroyed memory):

```
const Handle(A)& a = Handle(B)::DownCast (b); -> Handle(A) a (Handle(B)::DownCast (b));
```

10. Adds *#include* for all classes used as argument to macro *STANDARD_TYPE()*, except for already included ones;

11. Removes uses of obsolete macros *IMPLEMENT_DOWNCAST* and *IMPLEMENT_STANDARD_...*, except *IMPLEMENT_STANDARD_RTTIEXT*.

Note

If you plan to keep compatibility of your code with older versions of OCCT, add option *-compat* to avoid this change. See also Preserving compatibility with OCCT 6.x.

As long as the upgrade routine runs, some information messages are sent to the standard output. In some cases the warnings or errors like the following may appear:

```
Error in {HEADER_FILE}: Macro DEFINE_STANDARD_RTTI used for class {CLASS_NAME} whose declaration is not
       found in this file, cannot fix
```

Be sure to check carefully all reported errors and warnings, as the corresponding code will likely require manual corrections. In some cases these messages may help you to detect errors in your code, for instance, cases where *DEFINE_STANDARD_RTTI* macro is used with incorrect class name as an argument.

11.2.2 Possible compiler errors

Some situations requiring upgrade cannot be detected and / or handled by the automatic procedure. If you get compiler errors or warnings when trying to build the upgraded code, you will need to fix them manually. The following paragraphs list known situations of this kind.

Missing header files

The use of handle objects (construction, comparison using operators == or !=, use of function *STANDRAD_TYPE()* and method *DownCast()*) now requires the type of the object pointed by Handle to be completely known at compile time. Thus it may be necessary to include header of the corresponding class to make the code compilable.

For example, the following lines will fail to compile if *Geom_Line.hxx* is not included:

```
Handle(Geom_Line) aLine = 0;
if (aLine != aCurve) {...}
if (aCurve->IsKind(STANDARD_TYPE(Geom_Line)) {...}
aLine = Handle(Geom_Line)::DownCast (aCurve);
```

Note that it is not necessary to include header of the class to declare Handle to it. However, if you define a class *B* that uses Handle(*A*) in its fields, or contains a method returning Handle(*A*), it is advisable to have header defining *A* included in the header of *B*. This will eliminate the need to include the header *A* in each source file where class *B* is used.

Ambiguity of calls to overloaded functions

This issue appears in the compilers that do not support default arguments in template functions (known cases are Visual C++ 10 and 11): the compiler reports an ambiguity error if a handle is used in the argument of a call to the function that has two or more overloaded versions, receiving handles to different types. The problem is that operator *const handle<T2>&* is defined for any type *T2*, thus the compiler cannot make the right choice.

Example:

```
void func (const Handle(Geom_Curve)&);
void func (const Handle(Geom_Surface)&);

Handle(Geom_TrimmedCurve) aCurve = new Geom_TrimmedCurve (...);
func (aCurve); // ambiguity error in VC++ 10
```

Note that this problem can be avoided in many cases if macro *OCCT_HANDLE_NOCAST* is used, see below.

To resolve this ambiguity, change your code so that argument type should correspond exactly to the function signature. In some cases this can be done by using the relevant type for the corresponding variable, like in the example above:

```
Handle(Geom_Curve) aCurve = new Geom_TrimmedCurve (...);
```

Other variants consist in assigning the argument to a local variable of the correct type and using the direct cast or constructor:

```
const Handle(Geom_Curve)& aGCurve (aTrimmedCurve);
func (aGCurve); // OK - argument has exact type
func (static_cast(aCurve)); // OK - direct cast
func (Handle(Geom_Curve)(aCurve)); // OK - temporary handle is constructed
```

Another possibility consists in defining additional template variant of the overloaded function causing ambiguity, and using *SFINAE* to resolve the ambiguity. This technique can be illustrated by the definition of the template variant of method *IGESData_IGESWriter::Send()*.

Lack of implicit cast to base type

As the cast of a handle to the reference to another handle to the base type has become a user-defined operation, the conversions that require this cast together with another user-defined cast will not be resolved automatically by the compiler.

For example:

```
Handle(Geom_Geometry) aC = GC_MakeLine (p, v); // compiler error
```

The problem is that the class *GC_MakeLine* has a user-defined conversion to *const Handle(Geom_Trimmed↩ Curve)&,* which is not the same as the type of the local variable *aC*.

To resolve this, use method *Value()*:

```
Handle(Geom_Geometry) aC = GC_MakeLine (p, v).Value(); // ok
```

or use variable of the appropriate type:

```
Handle(Geom_TrimmedCurve) aC = GC_MakeLine (p, v); // ok
```

A similar problem appears with GCC compiler, when *const* handle to derived type is used to construct handle to base type via assignment (and in some cases in return statement), for instance:

```
const Handle(Geom_Line) aLine;
Handle(Geom_Curve) c1 = aLine; // GCC error
Handle(Geom_Curve) c2 (aLine); // ok
```

This problem is specific to GCC and it does not appear if macro *OCCT_HANDLE_NOCAST* is used, see below.

Incorrect use of STANDARD_TYPE and Handle macros

You might need to clean your code from incorrect use of macros *STANDARD_TYPE*() and *Handle*().

1. Explicit definitions of static functions with names generated by macro *STANDARD_TYPE()*, which are artifacts of old implementation of RTTI, should be removed.

   Example:

   ```
   const Handle(Standard_Type)& STANDARD_TYPE(math_GlobOptMin)
   {
     static Handle(Standard_Type) _atype = new Standard_Type ("math_GlobOptMin", sizeof (math_GlobOptMin));
     return _atype;
   }
   ```

2. Incorrect location of closing parenthesis of *Handle()* macro that was not detectable in OCCT 6.x will cause a compiler error and must be corrected.

   Example (note misplaced closing parenthesis):

   ```
   aBSpline = Handle( Geom2d_BSplineCurve::DownCast(BS->Copy()) );
   ```

Use of class Standard_AncestorIterator

Class *Standard_AncestorIterator* has been removed; use method *Parent()* of *Standard_Type* class to parse the inheritance chain.

Absence of cast to Standard_Transient*

Handles in OCCT 7.0 do not have the operator of conversion to *Standard_Transient**, which was present in earlier versions. This is done to prevent possible unintended errors like this:

```
Handle(Geom_Line) aLine = ...;
Handle(Geom_Surface) aSurf = ...;
...
if (aLine == aSurf) {...} // will cause a compiler error in OCCT 7.0, but not OCCT 6.x
```

The places where this implicit cast has been used should be corrected manually. The typical situation is when Handle is passed to stream:

```
Handle(Geom_Line) aLine = ...;
os << aLine; // in OCCT 6.9.0, resolves to operator << (void*)
```

Call method *get()* explicitly to output the address of the Handle.

Method DownCast for non-base types

Method *DownCast()* in OCCT 7.0 is made templated; if its argument is not a base class, "deprecated" compiler warning is generated. This is done to prevent possible unintended errors like this:

```
Handle(Geom_Surface) aSurf = ;
Handle(Geom_Line) aLine =
  Handle(Geom_Line)::DownCast (aSurf); // will cause a compiler warning in OCCT 7.0, but not OCCT 6.x
```

The places where this cast has been used should be corrected manually.

If down casting is used in a template context where the argument can have the same or unrelated type so that *DownCast()* may be not available in all cases, use C++ *dynamic_cast*<> instead, e.g.:

```
template <class T>
bool CheckLine (const Handle(T) theArg)
{
  Handle(Geom_Line) aLine = dynamic_cast<Geom_Line> (theArg.get());
  ...
}
```

11.2.3 Possible runtime problems

Here is the list of known possible problems at run time after the upgrade to OCCT 7.0.

References to temporary objects

In previous versions, the compiler was able to detect the situation when a local variable of a "reference to a Handle" type is initialized by temporary object, and ensured that lifetime of that object is longer than that of the variable. In OCCT 7.0 with default options, it will not work if types of the temporary object and variable are different (due to involvement of user-defined type cast), thus such temporary object will be destroyed immediately.

This problem does not appear if macro *OCCT_HANDLE_NOCAST* is used during compilation, see below.

Example:

```
// note that DownCast() returns new temporary object!
const Handle(Geom_BoundedCurve)& aBC =
Handle(Geom_TrimmedCurve)::DownCast(aCurve);
aBC->Transform (T); // access violation in OCCT 7.0
```

11.2.4 Option to avoid cast of handle to reference to base type

In OCCT 6.x and earlier versions the handle classes formed a hierarchy echoing the hierarchy of the corresponding object classes . This automatically enabled the possibility to use the handle to a derived class in all contexts where the handle to a base class was needed, e.g. to pass it in a function by reference without copying:

```
Standard_Boolean GetCurve (Handle(Geom_Curve)& theCurve);
....
Handle(Geom_Line) aLine;
if (GetCurve (aLine)) {
  // use aLine, unsafe
}
```

This feature was used in multiple places in OCCT and dependent projects. However it is potentially unsafe: in the above example no checks are done at compile time or at run time to ensure that the type assigned to the argument handle is compatible with the type of the handle passed as argument. If an object of incompatible type (e.g. Geom_Circle) is assigned to *theCurve*, the behavior will be unpredictable.

For compatibility with the existing code, OCCT 7.0 keeps this possibility by default, providing operators of type cast to the handle to a base type. However, this feature is unsafe and in specific situations it may cause compile-time or run-time errors as described above.

To provide a safer behavior, this feature can be disabled by a compile-time macro *OCCT_HANDLE_NOCAST*. When it is used, constructors and assignment operators are defined (instead of type cast operators) to convert handles to a derived type into handles to a base type. This implies creation of temporary objects and hence may be more expensive at run time in some circumstances, however this way is more standard, safer, and in general recommended.

The code that relies on the possibility of casting to base should be amended to always use the handle of argument type in function call and to use *DownCast()* to safely convert the result to the desired type. For instance, the code from the example below can be changed as follows:

```
Handle(Geom_Line) aLine;
Handle(Geom_Curve) aCurve;
if (GetCurve (aCure) && !(aLine = Handle(Geom_Line)::DownCast (aCurve)).IsNull()) {
  // use aLine safely
}
```

11.2.5 Preserving compatibility with OCCT 6.x

If you like to preserve the compatibility of your application code with OCCT versions 6.x even after the upgrade to 7.0, consider the following suggestions:

1. If your code used sequences of macros *IMPLEMENT_STANDARD_*... generated by WOK, replace them by single macro *IMPLEMENT_STANDARD_RTTIEXT*
2. When running automatic upgrade tool, add option *-compat*.
3. Define macros *DEFINE_STANDARD_RTTIEXT* and *DEFINE_STANDARD_RTTI_INLINE* when building with previous versions of OCCT, resolving to *DEFINE_STANDARD_RTTI* with single argument

   Example:

   ```
   #if OCC_VERSION_HEX < 0x070000
     #define DEFINE_STANDARD_RTTIEXT(C1,C2) DEFINE_STANDARD_RTTI(C1)
     #define DEFINE_STANDARD_RTTI_INLINE(C1,C2) DEFINE_STANDARD_RTTI(C1)
   #endif
   ```

11.2.6 Applications based on CDL and WOK

If your application is essentially based on CDL, and you need to upgrade it to OCCT 7.0, you will very likely need to convert your application code to non-CDL form. This is a non-trivial effort; the required actions would depend strongly on the structure of the code and used CDL features.

The upgrade script and sources of a specialized WOK version used for OCCT code upgrade can be found in WOK Git repository in branch CR0_700_2.

Contact us if you need more help.

11.3 Separation of BSpline cache

Implementation of NURBS curves and surfaces has been revised: the cache of polynomial coefficients, which is used to accelerate the calculation of values of a B-spline, has been separated from data objects *Geom2d_BSplineCurve, Geom_BSplineCurve* and *Geom_BSplineSurface* into the dedicated classes *BSplCLib_Cache* and *BSplSLib_Cache*.

The benefits of this change are:

- Reduced memory footprint of OCCT shapes (up to 20% on some cases)
- Possibility to evaluate the same B-Spline concurrently in parallel threads without data races and mutex locks

The drawback is that direct evaluation of B-Splines using methods of curves and surfaces becomes slower due to the absence of cache. The slow-down can be avoided by using adaptor classes *Geom2dAdaptor_Curve, GeomAdaptor_Curve* and *GeomAdaptor_Surface*, which now use cache when the curve or surface is a B-spline.

OCCT algorithms have been changed to use adaptors for B-spline calculations instead of direct methods for curves and surfaces. The same changes (use of adaptors instead of direct call to curve and surface methods) should be implemented in relevant places in the applications based on OCCT to get the maximum performance.

11.4 Structural result of Boolean operations

The result of Boolean operations became structured according to the structure of the input shapes. Therefore it may impact old applications that always iterate on direct children of the result compound assuming to obtain solids as iteration items, regardless of the structure of the input shapes. In order to get always solids as iteration items it is recommended to use TopExp_Explorer instead of TopoDS_Iterator.

11.5 BRepExtrema_ExtCC finds one solution only

Extrema computation between non-analytical curves in shape-shape distance calculation algorithm has been changed in order to return only one solution. So, if e.g. two edges are created on parallel b-spline curves the algorithm BRepExtrema_DistShapeShape will return only one solution instead of enormous number of solutions. There is no way to get algorithm working in old manner.

11.6 Removal of SortTools package

Package *SortTools* has been removed. The code that used the tools provided by that package should be corrected manually. The recommended approach is to use sorting algorithms provided by STL.

For instance:

```
#include <SortTools_StraightInsertionSortOfReal.hxx>
#include <SortTools_ShellSortOfReal.hxx>
#include <TCollection_CompareOfReal.hxx>
...
TCollection_Array1OfReal aValues = ...;
...
TCollection_CompareOfReal aCompReal;
SortTools_StraightInsertionSortOfReal::Sort(aValues, aCompReal);
```

can be replaced by:

```
#include <algorithm>
...
TCollection_Array1OfReal aValues = ...;
...
std::stable_sort (aValues.begin(), aValues.end());
```

11.7 On-screen objects and ColorScale

The old mechanism for rendering Underlay and Overlay on-screen 2D objects based on *Visual3d_Layer* and immediate drawing model (uncached and thus slow) has been removed. Classes *Aspect_Clayer2d, OpenGl_GraphicDriver_Layer, Visual3d_Layer, Visual3d_LayerItem, V3d_LayerMgr* and *V3d_LayerMgrPointer* have been deleted. The following auxiliary definition have been removed as well: Aspect_TypeOfPrimitive, Aspect_TypeOfLayer, Aspect_TypeOfEdge, Aspect_TypeOfDrawMode, Aspect_TypeOfConstraint, Aspect_DriverDefinitionError, Aspect_BadAccess.

General AIS interactive objects with transformation persistence flag *Graphic3d_TMF_2d* can be used as a replacement of *Visual3d_LayerItem*. The anchor point specified for transformation persistence defines the window corner of (or center in case of (0, 0) point). To keep on-screen 2D objects on top of the main screen, they can be assigned to the appropriate Z-layer. Predefined Z-layers *Graphic3d_ZLayerId_TopOSD* and *Graphic3d_ZLayerId_BotOSD* are intended to replace Underlay and Overlay layers within the old API.

ColorScale object previously implemented using *Visual3d_LayerItem* has been moved to a new class *AIS_ColorScale*, with width and height specified explicitly. The property of *V3d_View* storing the global *ColorScale* object has been removed with associated methods *V3d_View::ColorScaleDisplay(), V3d_View::ColorScaleErase(), V3d_View::ColorScaleIsDisplayed()* and *V3d_View::ColorScale()* as well as the classes *V3d_ColorScale, V3d_ColorScaleLayerItem* and *Aspect_ColorScale*. Here is an example of creating *ColorScale* using the updated API:

```
Handle(AIS_ColorScale) aCS = new AIS_ColorScale();
// configuring
Standard_Integer aWidth, aHeight;
aView->Window()->Size (aWidth, aHeight);
aCS->SetSize              (aWidth, aHeight);
aCS->SetRange             (0.0, 10.0);
aCS->SetNumberOfIntervals (10);
// displaying
aCS->SetZLayer (Graphic3d_ZLayerId_TopOSD);
aCS->SetTransformPersistence (Graphic3d_TMF_2d, gp_Pnt (-1,-1,0));
aCS->SetToUpdate();
theContextAIS->Display (aCS);
```

To see how 2d objects are implemented in OCCT you can call Draw commands *vcolorscale, vlayerline* or *vdrawtext* (with *-2d* option). Draw command *vcolorscale* now requires the name of *ColorScale* object as argument. To display this object use command *vdisplay*. For example:

```
pload VISUALIZATION
vinit
vcolorscale cs -demo
pload MODELING
box b 100 100 100
vdisplay b
vsetdispmode 1
vfit
vlayerline 0 300 300 300 10
vdrawtext t "2D-TEXT" -2d -pos 0 150 0 -color red
```

Here is a small example in C++ illustrating how to display a custom AIS object in 2d:

```
Handle(AIS_InteractiveContext) aContext = ...;
Handle(AIS_InteractiveObject) anObj =...; // create an AIS object
anObj->SetZLayer(Graphic3d_ZLayerId_TopOSD); // display object in overlay
anObj->SetTransformPersistence (Graphic3d_TMF_2d, gp_Pnt (-1,-1,0)); // set 2d flag, coordinate origin is
       set to down-left corner
aContext->Display (anObj); // display the object
```

11.8 UserDraw and Visual3d

Visual3d package

Package *Visual3d* implementing the intermediate layer between high-level *V3d* classes and low-level OpenGl classes for views and graphic structures management has been dropped.

The *OpenGl_View* inherits from the new class *Graphic3d_CView*. *Graphic3d_CView* is an interface class that declares abstract methods for managing displayed structures, display properties and a base layer code that implements computation and management of HLR (or more broadly speaking view-depended) structures.

In the new implementation it takes place of the eliminated *Visual3d_View*. As before the instance of *Graphic3d_CView* is still completely managed by *V3d_View* classes. It can be accessed through *V3d_View* interface but normally it should not be required as all its methods are completely wrapped.

In more details, a concrete specialization of *Graphic3d_CView* is created and returned by the graphical driver on request. Right after the creation the views are directly used for setting rendering properties and adding graphical structures to be displayed.

The rendering of graphics is possible after mapping a window and activating the view. The direct setting of properties obsoletes the use of intermediate structures with display parameter like *Visual3d_ContextView*, etc. This means that the whole package *Visual3d* becomes redundant.

The functionality previously provided by *Visual3d* package has been redesigned in the following way :

- The management of display of structures has been moved from *Visual3d_ViewManager* into *Graphic3d_StructureManager*.
- The class *Visual3d_View* has been removed. The management of computed structures has been moved into the base layer of *Graphi3d_CView*.
- All intermediate structures for storing view parameters, e.g. *Visual3d_ContextView*, have been removed. The settings are now kept by instances of *Graphic3d_CView*.
- The intermediate class *Visual3d_Light* has been removed. All light properties are stored in *Graphic3d_CLight* structure, which is directly accessed by instances of *V3d_Light* classes.
- All necessary enumerations have been moved into *Graphic3d* package.

Custom OpenGL rendering and UserDraw

Old APIs based on global callback functions for creating *UserDraw* objects and for performing custom OpenGL rendering within the view have been dropped. *UserDraw* callbacks are no more required since *OpenGl_Group* now inherits *Graphic3d_Group* and thus can be accessed directly from *AIS_InteractiveObject*:

```
//! Class implementing custom OpenGL element.
class UserDrawElement : public OpenGl_Element {};

//! Implementation of virtual method AIS_InteractiveObject::Compute().
void UserDrawObject::Compute (const Handle(PrsMgr_PresentationManager3d)& thePrsMgr,
                              const Handle(Prs3d_Presentation)& thePrs,
                              const Standard_Integer theMode)
{
  Graphic3d_Vec4 aBndMin (myCoords[0], myCoords[1], myCoords[2], 1.0f);
  Graphic3d_Vec4 aBndMax (myCoords[3], myCoords[4], myCoords[5], 1.0f);

  // casting to OpenGl_Group should be always true as far as application uses OpenGl_GraphicDriver for
       rendering
  Handle(OpenGl_Group) aGroup = Handle(OpenGl_Group)::DownCast (thePrs->NewGroup());
  aGroup->SetMinMaxValues (aBndMin.x(), aBndMin.y(), aBndMin.z(),
                           aBndMax.x(), aBndMax.y(), aBndMax.z());
  UserDrawElement* anElem = new UserDrawElement (this);
  aGroup->AddElement(anElem);

  // invalidate bounding box of the scene
  thePrsMgr->StructureManager()->Update();
}
```

To perform a custom OpenGL code within the view, it is necessary to inherit from class *OpenGl_View*. See the following code sample:

```
//! Custom view.
class UserView : public OpenGl_View
{
public:
  //! Override rendering into the view.
  virtual void render (Graphic3d_Camera::Projection theProjection,
                       OpenGl_FrameBuffer*          theReadDrawFbo,
                       const Standard_Boolean       theToDrawImmediate)
  {
    OpenGl_View::render (theProjection, theReadDrawFbo, theToDrawImmediate);
    if (theToDrawImmediate)
    {
```

```
      return;
    }

    // perform custom drawing
    const Handle(OpenGl_Context)& aCtx = myWorkspace->GetGlContext();
    GLfloat aVerts[3] = { 0.0f, 0,0f, 0,0f };
    aCtx->core20->glEnableClientState(GL_VERTEX_ARRAY);
    aCtx->core20->glVertexPointer(3, GL_FLOAT, 0, aVerts);
    aCtx->core20->glDrawArrays(GL_POINTS, 0, 1);
    aCtx->core20->glDisableClientState(GL_VERTEX_ARRAY);
  }

};

//! Custom driver for creating UserView.
class UserDriver : public OpenGl_GraphicDriver
{
public:
  //! Create instance of own view.
  virtual Handle(Graphic3d_CView) CreateView (const Handle(Graphic3d_StructureManager)& theMgr)
       Standard_OVERRIDE
  {
    Handle(UserView) aView = new UserView (theMgr, this, myCaps, myDeviceLostFlag, &myStateCounter);
    myMapOfView.Add (aView);
    for (TColStd_SequenceOfInteger::Iterator aLayerIt (myLayerSeq); aLayerIt.More(); aLayerIt.Next())
    {
      const Graphic3d_ZLayerId        aLayerID  = aLayerIt.Value();
      const Graphic3d_ZLayerSettings& aSettings = myMapOfZLayerSettings.Find (aLayerID);
      aView->AddZLayer         (aLayerID);
      aView->SetZLayerSettings (aLayerID, aSettings);
    }
    return aView;
  }
};
```

11.9 Deprecation of Local Context

The conception of Local Context has been deprecated. The related classes, e.g. *AIS_LocalContext*, and methods (*AIS_InteractiveContext::OpenLocalContext()* and others) will be removed in a future OCCT release.

The main functionality provided by Local Context - selection of object subparts - can be now used within Neutral Point without opening any Local Context.

The property *SelectionMode()* has been removed from the class *AIS_InteractiveObject*. This property contradicts to selection logic, since it is allowed to activate several Selection modes at once. Therefore keeping one selection mode as object field makes no sense. Applications that used this method should implement selection mode caching at application level, if it is necessary for some reason.

11.10 Separation of visualization part from TKCAF

Visualization CAF attributes have been moved into a new toolkit *TKVCAF*. If your application uses the classes from *TPrsStd* package then add link to *TKVCAF* library.

Version numbers of *BinOCAF* and *XmlOCAF* formats are incremented; new files cannot be read by earlier versions of OCCT.

Before loading the OCAF files saved by previous versions and containing *TPrsStd_AISPresentation* attribute it is necessary to define the environment variable *CSF_MIGRATION_TYPES*, pointing to file *src/StdResources/MigrationSheet.txt*. When using documents loaded from a file, make sure to call method *TPrsStd_AISViewer::New()* prior to accessing *TPrsStd_AISPresentation* attributes in this document as that method creates them.

11.11 Correction of interpretation of Euler angles in gp_Quaternion

Conversion of *gp_Quaternion* to and from intrinsic Tait-Bryan angles (including *gp_YawPitchRoll*) is fixed.

Before that fix the sequence of rotation axes was opposite to the intended; e.g. *gp_YawPitchRoll* (equivalent to *gp_Intrinsic_ZYX*) actually defined intrinsic rotations around X, then Y, then Z. Now the rotations are made in the correct order.

The applications that use *gp_Quaternion* to convert Yaw-Pitch-Roll angles (or other intrinsic Tait-Bryan sequences) may need to be updated to take this change into account.

11.12 Zoom Persistent Selection

Zoom persistent selection introduces a new structure *Graphic3d_TransformPers* to transform persistence methods and parameters and a new class *Graphic3d_WorldViewProjState* to refer to the camera transformation state. You might need to update your code to deal with the new classes if you were using the related features. Keep in mind the following:

- *Graphic3d_Camera::ModelViewState* has been renamed to *Graphic3d_Camera::WorldViewState*.
- Transformation matrix utilities from *OpenGl_Utils* namespace have been moved to *Graphic3d_TransformUtils* and *Graphic3d_TransformUtils.hxx* header respectively.
- Matrix stack utilities from *OpenGl_Utils* namespace have been moved to *OpenGl_MatrixStack* class and *OpenGl_MatrixStack.hxx* header respectively.
- *OpenGl_View* methods *Begin/EndTransformPersistence* have been removed. Use *Graphic3d_Transform↩ Pers::Apply()* instead to apply persistence to perspective and world-view projection matrices.

11.13 Texture mapping of objects

Textured objects now have the priority over the environment mapping.

Redundant enumerations *V3d_TypeOfSurface* and *Graphic3d_TypeOfSurface*, class *OpenGl_SurfaceDetailState*, the corresponding methods from *Graphic3d_CView, OpenGl_ShaderManager, OpenGl_View, V3d_View* and *V3d↩ _Viewer* have been deleted. Draw command *VSetTextureMode* has been deleted.

11.14 Shape presentation builders

Presentation tools for building Wireframe presentation have been refactored to eliminate duplicated code and interfaces. Therefore, the following classes have been modified:

- *StdPrs_WFDeflectionShape* and *Prs3d_WFShape* have been removed. *StdPrs_WFShape* should be used instead.
- *StdPrs_ToolShadedShape* has been renamed to *StdPrs_ToolTriangulatedShape*.

12 Upgrade to OCCT 7.1.0

12.1 Presentation attributes

This section should be considered if application defines custom presentations, i.e. inherited from *AIS_Interactive↩ Object*. The previous versions of OCCT have three levels for defining presentation properties (e.g. colors, materials, etc.):

1. For the entire structure - *Graphic3d_Structure* / *Prs3d_Presentation*.
2. For a specific group of primitives - *Graphic3d_Group::SetGroupPrimitivesAspect()* overriding structure aspects.
3. For a specific primitive array within the graphic group - *Graphic3d_Group::SetPrimitivesAspect()*.

The structure level has de facto not been used for a long time since OCCT presentations always define aspects at the graphic group level (overriding any structure aspects). Within this OCCT release, structure level of aspects has been completely removed. In most cases the application code should just remove missing methods. In those rare cases, when this functionality was intentionally used, the application should explicitly define aspects to the appropriate graphic groups.

Note that defining several different aspects within the same graphic group should also be avoided in the application code since it is a deprecated functionality which can be removed in further releases. *Graphic3d_Group::SetGroup↩ PrimitivesAspect()* should be the main method defining presentation attributes.

The implementation of *Graphic3d_Group::SetGroupPrimitivesAspect()* has been changed from copying aspect values to keeping the passed object. Although it was not documented, previously it was possible to modify a single aspect instance, like *Graphic3d_AspectFillArea3d* and set it to multiple groups. Now such code would produce an unexpected result and therefore should be updated to create the dedicated aspect instance.

12.2 Typedefs

The following type definitions in OCCT has been modified to use C++11 types:

- *Standard_Boolean* is now *bool* (previously *unsigned int*).
- *Standard_ExtCharacter* is now *char16_t* (previously *short*).
- *Standard_ExtString;* is now *const char16_t* (previously *const short*).
- *Standard_Utf16Char* is now *char16_t* (previously *uint16_t* for compatibility with old compilers).
- *Standard_Utf32Char* is now *char32_t* (previously *uint32_t* for compatibility with old compilers).

For most applications this change should be transparent on the level of source code. Binary compatibility is not maintained, as *bool* has a different size in comparison with *unsigned int*.

12.3 Programmable Pipeline

Fixed-function pipeline has been already deprecated since OCCT 7.0.0. Release 7.1.0 disables this functionality by default in favor of Programmable Pipeline (based on GLSL programs).

Method *V3d_View::Export()*, based on *gl2ps* library, requires fixed pipeline and will return error if used with default settings. Applications should explicitly enable fixed pipeline by setting *OpenGl_Caps::ffpEnable* flag to TRUE within *OpenGl_GraphicDriver::ChangeOptions()* before creating the viewer to use *V3d_View::Export()*. This method is declared as deprecated and will be removed in one of the the next OCCT releases. The recommended way to generate a vector image of a 3D model or scene is to use an application-level solution independent from OpenGL.

12.4 Transformation persistence

The behavior of transformation persistence flags *Graphic3d_TMF_ZoomPers* and *Graphic3d_TMF_TriedronPers* has been changed for consistency with a textured fixed-size 2D text. An object with these flags is considered as defined in pixel units, and the presentation is no more scaled depending on the view height. The applications that need to scale such objects depending on viewport size should update them manually.

Flags *Graphic3d_TMF_PanPers* and *Graphic3d_TMF_FullPers* have been removed. *Graphic3d_TMF_TriedronPers* or *Graphic3d_TMF_2d* can be used instead depending on the context.

Graphic3d_TransModeFlags is not an integer bitmask anymore - enumeration values should be specified instead. Several transformation persistence methods in *PrsMgr_PresentableObject* have been marked deprecated. Transformation persistence should be defined using *Graphic3d_TransformPers* constructor directly and passed by a handle, not value.

12.5 Dynamic highlight and selection properties

Release 7.1.0 introduces *Graphic3d_HighlightStyle* - an entity that allows flexible customization of highlighting parameters (such as highlighting method, color, and transparency). Therefore, the signatures of the following methods related to highlighting:

- *AIS_InteractiveContext::Hilight()*;
- *AIS_InteractiveContext::HilightWithColor()*;
- *PrsMgr_PresentationManager::Color()*;
- *SelectMgr_EntityOwner::HilightWithColor()*; have been changed to receive *Graphic3d_HighlightStyle* instead of *Quantity_Color*.

Method *AIS_InteractiveContext::Hilight* is now deprecated and highlights the interactive object with selection style.

A group of methods *AIS_InteractiveContext::IsHilighted* has changed its behavior - now they only check highlight flags of the object or the owner in the global status. If the highlight color is required on the application level, it is necessary to use overloaded methods *AIS_InteractiveContext::HighlightStyle* for the owner and the object.

The following methods have been replaced in *AIS_InteractiveContext* class:

- *HilightColor* and *SetHilightColor* by *HighlightStyle* and *SetHighlightStyle*;
- *SelectionColor* setter and getter by *SelectionStyle* and *SetSelectionStyle*.

The API of *Prs3d_Drawer* has been extended to allow setting up styles for both dynamic selection and highlighting. Therefore, it is possible to change the highlight style of a particular object on the application level via *SelectMgr_SelectableObject::HilightAttributes()* and process it in the entity owner.

12.6 Correction in TObj_Model class

Methods *TObj_Model::SaveAs* and *TObj_Model::Load* now receive *TCollection_ExtendedString* filename arguments instead of char*. UTF-16 encoding can be used to pass file names containing Unicode symbols.

12.7 Redundant environment variables

The following environment variables have become redundant:

- *CSF_UnitsLexicon* and *CSF_UnitsDefinition* are no more used. Units definition (*UnitsAPI/Lexi_Expr.dat* and *UnitsAPI/Units.dat*) is now embedded into source code.

- *CSF_XSMessage* and *CSF_XHMessage* are now optional. English messages (XSMessage/∗XSTEP.us∗ and SHMessage/∗SHAPE.us∗) are now embedded into source code and automatically loaded when environment variables are not set.
- *CSF_ShadersDirectory* is not required any more, though it still can be used to load custom shaders. Mandatory GLSL resources are now embedded into source code.
- *CSF_PluginDefaults* and other variables pointing to OCAF plugin resources (*CSF_StandardDefaults*, *CSF_XCAFDefaults*, *CSF_StandardLiteDefaults* and *CSF_XmlOcafResource*) are not necessary if method *TDocStd_Application::DefineFormat()* is used to enable persistence of OCAF documents.

Other environment variables still can be used to customize behavior of relevant algorithms but are not necessary any more (all required resources are embedded).

12.8 Removed features

The following obsolete features have been removed:

- Anti-aliasing API *V3d_View::SetAntialiasingOn()*. This method was intended to activate deprecated OpenGL functionality *GL_POLYGON_SMOOTH, GL_LINE_SMOOTH* and *GL_POINT_SMOOTH*. Instead of the old API, the application should request MSAA buffers for anti-aliasing by assigning *Graphic3d_RenderingParams::NbMsaaSamples* property of the structure returned by *V3d_View::ChangeRenderingParams()*.
- *Prs3d_Drawer::ShadingAspectGlobal()* flag has been removed as not used. The corresponding calls can be removed safely from the application code.
- The methods managing ZClipping planes and ZCueing: *V3d_View::SetZClippingType()*, *V3d_View::SetZCueingOn()*, etc. have been removed. ZClipping planes can be replaced by general-purpose clipping planes (the application should update plane definition manually).
- The 3D viewer printing API *V3d_View::Print()* has been removed. This functionality was available on Windows platforms only. The applications should use the general image dump API *V3d_View::ToPixMap()* and manage printing using a platform-specific API at the application level. Text resolution can be managed by rendering parameter *Graphic3d_RenderingParams::Resolution*, returned by *V3d_View::ChangeRenderingParams()*.
- Methods *PrsMgr_PresentationManager::BoundBox*, *PrsMgr_PresentationManager::Hilight* and *SelectMgr_EntityOwner::Hilight* have been removed as not used. The corresponding method in custom implementations of *SelectMgr_EntityOwner* can be removed safely. *PrsMgr_PresentationManager::Color* with the corresponding style must be used instead.
- Class *NCollection_QuickSort* has been removed. The code that used the tools provided by that class should be corrected manually. The recommended approach is to use sorting algorithms provided by STL (std::sort). See also Removal of SortTools package above.
- Package *Dico*. The code that used the tools provided by that package should be corrected manually. The recommended approach is to use *NCollection_DataMap* and *NCollection_IndexedDataMap* classes.

12.9 Other changes

The following classes have been changed:

- *BVH_Sorter* class has become abstract. The list of arguments of both *Perform* methods has been changed and the methods became pure virtual.
- *Extrema_FuncExtPS* has been renamed to *Extrema_FuncPSNorm*.
- The default constructor and the constructor taking a point and a surface have been removed from class *Extrema_GenLocateExtPS*. Now the only constructor takes the surface and optional tolerances in U and V directions. The new method *Perform* takes the point with the start solution and processes it. The class has become not assignable and not copy-constructable.

- Constructors with arguments ∗(const gp_Ax22d& D, const gp_Pnt2d& F)∗ have been removed from *GCE2d_MakeParabola*, *gce_MakeParab2d* and *gp_Parab2d*. The objects created with some constructors of class *gp_Parab2d* may differ from the previous version (see the comments in *gp_Parab2d.hxx*). The result returned by *gp_Parab2d::Directrix()* method has an opposite direction in comparison with the previous OCCT versions.
- *BRepTools_Modifier* class now has two modes of work. They are defined by the boolean parameter *MutableInput*, which is turned off by default. This means that the algorithm always makes a copy of a sub-shape (e.g. vertex) if its tolerance is to be increased in the output shape. The old mode corresponds to *MutableInput* turned on. This change may impact an application if it implements a class derived from *BRepTools_Modifier*.
- The second parameter *theIsOuterWire* of method *ShapeAnalysis_Wire::CheckSmallArea* has been removed.
- In class *GeomPlate_CurveConstraint*, two constructors taking boundary curves of different types have been replaced with one constructor taking the curve of an abstract type.
- The last optional argument *RemoveInvalidFaces* has been removed from the constructor of class *BRepOffset_MakeOffset* and method *Initialize*.
- The public method *BOPDS_DS::VerticesOnIn* has been renamed into *SubShapesOnIn* and the new output parameter *theCommonPB* has been added.

13 Upgrade to OCCT 7.2.0

13.1 Removed features

The following obsolete features have been removed:

- *AIS_InteractiveContext::PreSelectionColor()*, *DefaultColor()*, *WasCurrentTouched()*, *ZDetection()*. These properties were unused, and therefore application should remove occurrences of these methods.
- *AIS_InteractiveObject::SelectionPriority()*. These property was not implemented.
- The class *LocOpe_HBuilder* has been removed as obsolete.
- The package *TestTopOpe* has been removed;
- The package *TestTopOpeDraw* has been removed;
- The package *TestTopOpeTools* has been removed.
- The packages *QANewModTopOpe*, *QANewBRepNaming* and *QANewDBRepNaming* have been removed as containing obsolete features.
- The following methods of the *IntPolyh_Triangle* class have been removed as unused:
  - *CheckCommonEdge*
  - *SetEdgeandOrientation*
  - *MultipleMiddleRefinement2*.
- The method *IntPolyh_Triangle::TriangleDeflection* has been renamed to *IntPolyh_Triangle::Compute↩ Deflection*.
- The following methods of the *IntPolyh_MaillageAffinage* class have been removed as unused:
  - *LinkEdges2Triangles*;
  - *TriangleEdgeContact2*;
  - *StartingPointsResearch2*;
  - *NextStartingPointsResearch2*;
  - *TriangleComparePSP*;
  - *StartPointsCalcul*.
- The method PerformAdvanced of the *ShapeConstruct_ProjectCurveOnSurface* class has been removed as unused.
- The method Perform of the *ShapeConstruct_ProjectCurveOnSurface* class is modified:
  - input arguments *continuity*, *maxdeg*, *nbinterval* have been removed as unused;
  - input arguments *TolFirst*, *TolLast* have been added at the end of arguments' list.
- Typedefs Quantity_Factor, Quantity_Parameter, Quantity_Ratio, Quantity_Coefficient, Quantity_PlaneAngle, Quantity_Length, V3d_Parameter and V3d_Coordinate have been removed; Standard_Real should be used instead.

13.2 Corrections in BRepOffset API

In classes *BRepTools_ReShape* and *ShapeBuild_ReShape*, the possibility to process shapes different only by orientation in different ways has been removed. Thus methods *Remove()* and *Replace()* do not have any more the last argument 'oriented'; they work always as if *Standard_False* was passed before (default behavior). Methods *ModeConsiderLo()* and *Apply()* with three arguments have been removed.

13.3 Corrections in BRepOffset API

Class *BRepOffsetAPI_MakeOffsetShape*:

- *BRepOffsetAPI_MakeOffsetShape::BRepOffsetAPI_MakeOffsetShape()* - constructor with parameters has been deleted.
- *BRepOffsetAPI_MakeOffsetShape::PerformByJoin()* - method has been added. This method is old algorithm behaviour.

The code below shows new calling procedure:

```
BRepOffsetAPI_MakeOffsetShape OffsetMaker;
OffsetMaker.PerformByJoin(Shape, OffsetValue, Tolerance);
NewShape = OffsetMaker.Shape();
```

Class *BRepOffsetAPI_MakeThickSolid*:

- *BRepOffsetAPI_MakeThickSolid::BRepOffsetAPI_MakeThickSolid()* - constructor with parameters has been deleted.
- *BRepOffsetAPI_MakeThickSolid::MakeThickSolidByJoin()* - method has been added. This method is old algorithm behaviour.

The code below shows new calling procedure:

```
BRepOffsetAPI_MakeThickSolid BodyMaker;
BodyMaker.MakeThickSolidByJoin(myBody, facesToRemove, -myThickness / 50, 1.e-3);
myBody = BodyMaker.Shape();
```

13.4 Highlight style

Management of highlight attributes has been revised and might require modifications from application side:

- New class *Graphic3d_PresentationAttributes* defining basic presentation attributes has been introduced. It's definition includes properties previously defined by class Graphic3d_HighlightStyle (*Color*, *Transparency*), and new properties (*Display mode*, *ZLayer*, optional *FillArea aspect*).
- Class *Prs3d_Drawer* now inherits class *Graphic3d_PresentationAttributes*. So that overall presentation attributes are now split into two parts - Basic attributes and Detailed attributes.
- Class *Graphic3d_HighlightStyle* has been dropped. It is now defined as a typedef to *Prs3d_Drawer*. Therefore, highlight style now also includes not only Basic presentation attributes, but also Detailed attributes which can be used by custom presentation builders.
- Highlighting style defined by class *Graphic3d_PresentationAttributes* now provides more options:
  - *Graphic3d_PresentationAttributes::BasicFillAreaAspect()* property providing complete Material definition. This option, when defined, can be used instead of the pair Object Material + Highlight Color.
  - *Graphic3d_PresentationAttributes::ZLayer()* property specifying the Layer where highlighted presentation should be shown. This property can be set to Graphic3d_ZLayerId_UNKNOWN, which means that ZLayer of main presentation should be used instead.
  - *Graphic3d_PresentationAttributes::DisplayMode()* property specifying Display Mode for highlight presentation.
- Since Highlight and Selection styles within *AIS_InteractiveContext* are now defined by *Prs3d_Drawer* inheriting from *Graphic3d_PresentationAttributes*, it is now possible to customize default highlight attributes like *Display Mode* and *ZLayer*, which previously could be defined only on Object level.

- Properties *Prs3d_Drawer::HighlightStyle()* and *Prs3d_Drawer::SelectionStyle()* have been removed. Instead, *AIS_InteractiveObject* now defines *DynamicHilightAttributes()* for dynamic highlighting in addition to *Hilight↩ Attributes()* used for highlighting in selected state. Note that *AIS_InteractiveObject::HilightAttributes()* and *AIS_InteractiveObject::DynamicHilightAttributes()* override highlighting properties for both - entire object and for part coming from decomposition. This includes Z-layer settings, which will be the same when overriding properties through AIS_InteractiveObject, while *AIS_InteractiveContext::HighlightStyle()* allows customizing properties for local and global selection independently (with Graphic3d_ZLayerId_Top used for dynamic highlighting of entire object and Graphic3d_ZLayerId_Topmost for dynamic highlighting of object part by default).
- The following protected fields have been removed from class *AIS_InteractiveObject*:
  - *myOwnColor*, replaced by *myDrawer->Color()*
  - *myTransparency*, replaced by *myDrawer->Transparency()*
  - *myZLayer*, replaced by *myDrawer->ZLayer()*
- The method *PrsMgr_PresentationManager::Unhighlight()* taking Display Mode as an argument has been marked deprecated. Implementation now performs unhighlighting of all highlighted presentation mode.
- The methods taking/returning *Quantity_NameOfColor* (predefined list of colors) and duplicating methods operating with *Quantity_Color* (definition of arbitrary RGB color) in AIS have been removed. *Quantity_Color* should be now used instead.

13.5 Elimination of implicit 3D Viewer updates

Most AIS_InteractiveContext methods are defined with a flag to update viewer immediatly or not. Within previous version of OCCT, this argument had default value TRUE. While immediate viewer updates are useful for beginners (the result is displayed as soon as possible), this approach is inefficent for batch viewer updates, and having default value as TRUE leaded to non-intended accidential updates which are difficult to find.

To avoid such issues, the interface has been modified and default value has been removed. Therefore, old application code should be updated to set the flag theToUpdateViewer explicitly to desired value (TRUE to preserve old previous behavior), if it was not already set.

The following AIS_InteractiveContext methods have been changed: Display, Erase, EraseAll, DisplayAll, EraseSelected, DisplaySelected, ClearPrs, Remove, RemoveAll, Hilight, HilightWithColor, Unhilight, Redisplay, RecomputePrsOnly, Update, SetDisplayMode, UnsetDisplayMode, SetColor, UnsetColor, SetWidth, Unset↩ Width, SetMaterial, UnsetMaterial, SetTransparency, UnsetTransparency, SetLocalAttributes, UnsetLocalAttributes, SetPolygonOffsets, SetTrihedronSize, SetPlaneSize, SetPlaneSize, SetDeviationCoefficient, SetDeviationAngle, SetAngleAndDeviation, SetHLRDeviationCoefficient, SetHLRDeviationAngle, SetHLRAngleAndDeviation, Set↩ SelectedAspect, MoveTo, Select, ShiftSelect, SetSelected, UpdateSelected, AddOrRemoveSelected, Hilight↩ Selected, UnhilightSelected, ClearSelected, ResetOriginalState, SubIntensityOn, SubIntensityOff, FitSelected, EraseGlobal, ClearGlobal, ClearGlobalPrs.

In addition, the API for immediate viewer update has been removed from V3d_View and Graphic3d_Structure↩ Manager classes (enumerations *Aspect_TypeOfUpdate* and *V3d_TypeOfUpdate*): V3d::SetUpdateMode(), V3d::↩ UpdateMode(), Graphic3d_StructureManager::SetUpdateMode(), Graphic3d_StructureManager::UpdateMode().

The argument theUpdateMode has been removed from methods Graphic3d_CView::Display(), Erase(), Update(). Method Graphic3d_CView::Update() does not redraw the view and does not re-compute structures anymore.

The following Grid management methods within class V3d_Viewer do not implicitly redraw the viewer: ActivateGrid, DeactivateGrid, SetRectangularGridValues, SetCircularGridValues, RectangularGridGraphicValues, CircularGrid↩ GraphicValues, SetPrivilegedPlane, DisplayPrivilegedPlane.

13.6 Elimination of Quantity_NameOfColor from TKV3d interface classes

The duplicating interface methods accepting *Quantity_NameOfColor* (in addition to methods accepting *Quantity_↩ Color*) of TKV3d toolkit have been removed. In most cases this change should be transparent, however applications implementing such interface methods should also remove them (compiler will automatically highlight this issue for methods marked with Standard_OVERRIDE keyword).

13.7 Result of Boolean operations on containers

- The result of Boolean operations on arguments of collection types (WIRE/SHELL/COMPSOLID) is now filtered from duplicating containers.

13.8 Other changes

- *MMgt_TShared* class definition has been removed - Standard_Transient should be used instead (MMgt_T↩ Shared is marked as deprecated typedef of Standard_Transient for smooth migration).
- Class GeomPlate_BuildPlateSurface accepts base class Adaptor3d_HCurve (instead of inherited Adaptor3d_HCurveOnSurface accepted earlier).
- Types GeomPlate_Array1OfHCurveOnSurface and GeomPlate_HArray1OfHCurveOnSurface have been replaced with GeomPlate_Array1OfHCurve and GeomPlate_HArray1OfHCurve correspondingly (accept base class Adaptor3d_HCurve instead of Adaptor3d_HCurveOnSurface).
- Enumeration *Image_PixMap::ImgFormat*, previously declared as nested enumeration within class *Image_↩ PixMap*, has been moved to global namespace as *Image_Format* following OCCT coding rules. The enumeration values have suffix Image_Format_ and preserve previous name scheme for easy renaming of old values - e.g. Image_PixMap::ImgGray become Image_Format_Gray. Old definitions are preserved as depreacated aliases to the new ones;
- Methods *Image_PixMap::PixelColor()* and *Image_PixMap::SetPixelColor()* now take/return Quantity_Color↩ RGBA instead of Quantity_Color/NCollection_Vec4.
- The method BOPAlgo_Builder::Origins() returns BOPCol_DataMapOfShapeListOfShape instead of BOP↩ Col_DataMapOfShapeShape.
- The methods BOPDS_DS::IsToSort(const Handle(BOPDS_CommonBlock)&, Standard_Integer&) and BO↩ PDS_DS::SortPaveBlocks(const Handle(BOPDS_CommonBlock)&) have been removed. The sorting is now performed during the addition of the Pave Blocks into Common Block.
- The methods BOPAlgo_Tools::MakeBlocks() and BOPAlgo_Tools::MakeBlocksCnx() have been replaced with the single template method BOPAlgo_Tools::MakeBlocks(). The chains of connected elements are now stored into the list of list instead of data map.
- The methods BOPAlgo_Tools::FillMap() have been replaced with the single template method BOPAlgo_↩ Tools::FillMap().
- Package BVH now uses opencascade::handle instead of NCollection_Handle (for classes BVH_Properties, BVH_Builder, BVH_Tree, BVH_Object). Application code using BVH package directly should be updated accordingly.
- AIS_Shape now computes UV texture coordinates for AIS_Shaded presentation in case if texture mapping is enabled within Shaded Attributes. Therefore, redundant class *AIS_TexturedShape is now deprecated* - applications can use *AIS_Shape* directly (texture mapping should be defined through AIS_Shape::Attributes()).
- Methods for managing active texture within OpenGl_Workspace class (ActiveTexture(), DisableTexture(), EnableTexture()) have been moved to *OpenGl_Context::BindTextures()*.

13.9 BOP - Pairs of interfering indices

- The classes *BOPDS_PassKey* and *BOPDS_PassKeyBoolean* are too excessive and not used any more in Boolean Operations. To replace them the new *BOPDS_Pair* class has been implemented. Thus:
  - The method *BOPDS_DS::Interferences()* now returns the *BOPDS_MapOfPair*;
  - The method *BOPDS_Iterator::Value()* takes now only two parameters - the indices of interfering subshapes.

13.10 Removal of the Draw commands based on old Boolean operations

- The commands *fubl* and *cubl* have been removed. The alternative for these commands are the commands *bfuseblend* and *bcutblend* respectively.
- The command *ksection* has been removed. The alternative for this command is the command *bsection*.

13.11 Change of Face/Face intersection in Boolean operations

- Previously, the intersection tolerance for all section curves between pair of faces has been calculated as the maximal tolerance among all curves. Now, each curve has its own valid tolerance calculated as the maximal deviation of the 3D curve from its 2D curves or surfaces in case there are no 2D curves.
- The methods *IntTools_FaceFace::TolReached3d()*, *IntTools_FaceFace::TolReal()* and *IntTools_FaceFace::TolReached2d()* have been removed.
- Intersection tolerances of the curve can be obtained from the curve itself:
  - *IntTools_Curve::Tolerance()* - returns the valid tolerance for the curve;
  - *IntTools_Curve::TangentialTolerance()* - returns the tangential tolerance, which reflects the size of the common between faces.
- 2d tolerance (*IntTools_FaceFace::TolReached2d()*) has been completely removed from the algorithm as unused.

13.12 Restore OCCT 6.9.1 persistence

The capability of reading / writing files in old format using *Storage_ShapeSchema* functionality from OCCT 6.9.1 has been restored in OCCT 7.2.0.

One can use this functionality in two ways:

- invoke DRAW Test Harness commands *fsdread / fsdwrite* for shapes;
- call *StdStorage* class *Read / Write* functions in custom code.

The code example below demonstrates how to read shapes from a storage driver using *StdStorage* class.

```
// aDriver should be created and opened for reading
Handle(StdStorage_Data) aData;

// Read data from the driver
// StdStorage::Read creates aData instance automatically if it is null
Storage_Error anError = StdStorage::Read(*aDriver, aData);
if (anError != Storage_VSOk)
{
  // Error processing
}

// Get root objects
Handle(StdStorage_RootData) aRootData = aData->RootData();
Handle(StdStorage_HSequenceOfRoots) aRoots = aRootData->Roots();
if (!aRoots.IsNull())
{
  // Iterator over the sequence of root objects
  for (StdStorage_HSequenceOfRoots::Iterator anIt(*aRoots); anIt.More(); anIt.Next())
  {
    Handle(StdStorage_Root)& aRoot = anIt.ChangeValue();
    // Get a persistent root's object
    Handle(StdObjMgt_Persistent) aPObject = aRoot->Object();
    if (!aPObject.IsNull())
    {
      Handle(ShapePersistent_TopoDS::HShape) aHShape =
       Handle(ShapePersistent_TopoDS::HShape)::DownCast(aPObject);
      if (aHShape) // Downcast to an expected type to import transient data
      {
        TopoDS_Shape aShape = aHShape->Import();
        shapes.Append(aShape);
      }
```

```
    }
  }
}
```

The following code demonstrates how to write shapes in OCCT 7.2.0 using *StdStorage* class.

```
// Create a file driver
NCollection_Handle<Storage_BaseDriver> aFileDriver(new FSD_File());

// Try to open the file driver for writing
try
{
  OCC_CATCH_SIGNALS
  PCDM_ReadWriter::Open (*aFileDriver, TCollection_ExtendedString(theFilename), Storage_VSWrite);
}
catch (Standard_Failure& e)
{
  // Error processing
}

// Create a storage data instance
Handle(StdStorage_Data) aData = new StdStorage_Data();
// Set an axiliary application name (optional)
aData->HeaderData()->SetApplicationName(TCollection_ExtendedString("Application"));

// Provide a map to track sharing
StdObjMgt_TransientPersistentMap aMap;
// Iterator over a collection of shapes
for (Standard_Integer i = 1; i <= shapes.Length(); ++i)
{
  TopoDS_Shape aShape = shapes.Value(i);
  // Translate a shape to a persistent object
  Handle(ShapePersistent_TopoDS::HShape) aPShape =
    ShapePersistent_TopoDS::Translate(aShape, aMap, ShapePersistent_WithTriangle);
  if (aPShape.IsNull())
  {
    // Error processing
  }

  // Construct a root name
  TCollection_AsciiString aName = TCollection_AsciiString("Shape_") + i;

  // Add a root to storage data
  Handle(StdStorage_Root) aRoot = new StdStorage_Root(aName, aPShape);
  aData->RootData()->AddRoot(aRoot);
}

// Write storage data to the driver
Storage_Error anError = StdStorage::Write(*aFileDriver, aData);
if (anError != Storage_VSOk)
{
  // Error processing
}
```

13.13 Change in BRepLib_MakeFace algorithm

Previously, *BRepLib_MakeFace* algorithm changed orientation of the source wire in order to avoid creation of face as a hole (i.e. it is impossible to create the entire face as a hole; the hole can be created in context of another face only). New algorithm does not reverse the wire if it is open. Material of the face for the open wire will be located on the left side from the source wire.

13.14 Change in BRepFill_OffsetWire algorithm

From now on, the offset will always be directed to the outer region in case of positive offset value and to the inner region in case of negative offset value. Inner/Outer region for an open wire is defined by the following rule: when we go along the wire (taking into account edges orientation) the outer region will be on the right side, the inner region will be on the left side. In case of a closed wire, the inner region will always be inside the wire (at that, the edges orientation is not taken into account).

13.15 Change in Geom(2d)Adaptor_Curve::IsPeriodic

Since 7.2.0 version, method *IsPeriodic()* returns the corresponding status of periodicity of the basis curve regardless of closure status of the adaptor curve (see method *IsClosed()*). Method *IsClosed()* for adaptor can return false even on periodic curve, in the case if its parametric range is not full period, e.g. for adaptor on circle in range [0, π]. In previous versions, *IsPeriodic()* always returned false if *IsClosed()* returned false.

13.16 Change in algorithm ShapeUpgrade_UnifySameDomain

The history of the changing of the initial shape was corrected:

- all shapes created by the algorithm are considered as modified shapes instead of generated ones;
- method Generated was removed and its calls should be replaced by calls of method History()->Modified.

13.17 Changes in STL Reader / Writer

Class RWStl now uses class Poly_Triangulation for storing triangular mesh instead of StlMesh data classes; the latter have been removed.

13.18 Refactoring of the Error/Warning reporting system in Boolean Component

The Error/Warning reporting system of the algorithms in Boolean Component (in all BOPAlgo_$*$ and BRepAlgoA↩ PI_$*$ algorithms) has been refactored. The methods returning the status of errors and warnings of the algorithms (ErrorStatus() and WarningStatus()) have been removed. Instead use methods HasErrors() and HasWarnings() to check for presence of errors and warnings, respectively. The full list of errors and warnings, with associated data such as problematic sub-shapes, can be obtained by method GetReport().

14 Upgrade to OCCT 7.2.1

14.1 Changes in ShapeUpgrade_UnifySameDomain

The following public methods in the class ShapeUpgrade_UnifySameDomain became protected:

- *UnifyFaces*
- *UnifyEdges*

The following public method has been removed:

- *UnifyFacesAndEdges*

14.2 Moving BuildPCurveForEdgeOnPlane from BOPTools_AlgoTools2D to BRepLib

The methods BuildPCurveForEdgeOnPlane and BuildPCurveForEdgesOnPlane have been moved from the class BOPTools_AlgoTools2D to the more lower level class BRepLib.

14.3 Removed features

The following obsolete features have been removed:

- The package BOPCol has been fully removed:
  - *BOPCol_BaseAllocator* is replaced with *Handle(NCollection_BaseAllocator)*;
  - *BOPCol_BoxBndTree* is replaced with *BOPTools_BoxBndTree*;
  - *BOPCol_Box2DBndTree* is removed as unused;
  - *BOPCol_DataMapOfIntegerInteger* is replaced with *TColStd_DataMapOfIntegerInteger*;
  - *BOPCol_DataMapOfIntegerListOfInteger* is replaced with *TColStd_DataMapOfIntegerListOfInteger*;
  - *BOPCol_DataMapOfIntegerListOfShape* is replaced with *TopTools_DataMapOfIntegerListOfShape*;
  - *BOPCol_DataMapOfIntegerMapOfInteger.hxx* is removed as unused;
  - *BOPCol_DataMapOfIntegerReal* is replaced with *TColStd_DataMapOfIntegerReal*;
  - *BOPCol_DataMapOfIntegerShape* is replaced with *TopTools_DataMapOfIntegerShape*;
  - *BOPCol_DataMapOfShapeBox* is replaced with *TopTools_DataMapOfShapeBox*;
  - *BOPCol_DataMapOfShapeInteger* is replaced with *TopTools_DataMapOfShapeInteger*;
  - *BOPCol_DataMapOfShapeListOfShape* is replaced with *TopTools_DataMapOfShapeListOfShape*;
  - *BOPCol_DataMapOfShapeReal* is replaced with *TopTools_DataMapOfShapeReal*;
  - *BOPCol_DataMapOfShapeShape* is replaced with *TopTools_DataMapOfShapeShape*;
  - *BOPCol_DataMapOfTransientAddress* is removed as unused;
  - *BOPCol_IndexedDataMapOfIntegerListOfInteger* is removed as unused;
  - *BOPCol_IndexedDataMapOfShapeBox* is removed as unused;
  - *BOPCol_IndexedDataMapOfShapeInteger* is removed as unused;
  - *BOPCol_IndexedDataMapOfShapeListOfShape* is replaced with *TopTools_IndexedDataMapOfShape↩ ListOfShape*;
  - *BOPCol_IndexedDataMapOfShapeReal* is removed as unused;
  - *BOPCol_IndexedDataMapOfShapeShape* is replaced with *TopTools_IndexedDataMapOfShapeShape*;
  - *BOPCol_IndexedMapOfInteger* is replaced with *TColStd_IndexedMapOfInteger*;
  - *BOPCol_IndexedMapOfOrientedShape* is replaced with *TopTools_IndexedMapOfOrientedShape*;

  - *BOPCol_IndexedMapOfShape* is replaced with *TopTools_IndexedMapOfShape*;
  - *BOPCol_ListOfInteger* is replaced with *TColStd_ListOfInteger*;
  - *BOPCol_ListOfListOfShape* is replaced with *TopTools_ListOfListOfShape*;
  - *BOPCol_ListOfShape* is replaced with *TopTools_ListOfShape*;
  - *BOPCol_MapOfInteger* is replaced with *TColStd_MapOfInteger*;
  - *BOPCol_MapOfOrientedShape* is replaced with *TopTools_MapOfOrientedShape*;
  - *BOPCol_MapOfShape* is replaced with *TopTools_MapOfShape*;
  - *BOPCol_PListOfInteger* is removed as unused;
  - *BOPCol_PInteger* is removed as unused
  - *BOPCol_SequenceOfPnt2d* is replaced with *TColgp_SequenceOfPnt2d*;
  - *BOPCol_SequenceOfReal* is replaced with *TColStd_SequenceOfReal*;
  - *BOPCol_SequenceOfShape* is replaced with *TopTools_SequenceOfShape*;
  - *BOPCol_Parallel* is replaced with *BOPTools_Parallel*;
  - *BOPCol_NCVector* is replaced with *NCollection_Vector*;

- The class *BOPDS_PassKey* and containers for it have been removed as unused.
- The unused containers from *IntTools* package have been removed:
  - *IntTools_DataMapOfShapeAddress* is removed as unused;
  - *IntTools_IndexedDataMapOfTransientAddress* is removed as unused;
- The container *BiTgte_DataMapOfShapeBox* is replaced with *TopTools_DataMapOfShapeBox*;
- The class *BOPTools* has been removed as duplicate of the class *TopExp*;
- The method *BOPAlgo_Builder::Splits()* has been removed as excessive. The method *BOPAlgo_Builder::Images()* can be used instead.
- The method *BOPTools_AlgoTools::CheckSameGeom()* has been removed as excessive. The method *BOPTools_AlgoTools::AreFacesSameDomain()* can be used instead.

15 Upgrade to OCCT 7.3.0

15.1 Light sources

Multiple changes have been applied to lights management within *TKV3d* and *TKOpenGl*:

- *V3d_Light* class is now an alias to *Graphic3d_CLight*. *Graphic3d_CLight* is now a Handle class with refactored methods for managing light source parameters. Most methods of *V3d_Light* sub-classes have been preserved to simplify porting.
- Obsolete debugging functionality for drawing a light source has been removed from *V3d_Light*. Methods and constructors that take parameters for debug display and do not affect the light definition itself have also been removed.
- Light constructors taking *V3d_Viewer* have been marked as deprecated. Use method *AddLight()* of the class *V3d_Viewer* or *V3d_View* to add new light sources to a scene or a single view, respectively.
- The upper limit of 8 light sources has been removed.
- The classes for specific light source types: *V3d_AmbientLight, V3d_DirectionalLight, V3d_PositionalLight* and *V3d_SpotLight* have been preserved, but it is now possible to define the light of any type by creating base class *Graphic3d_CLight* directly. The specific classes only hide unrelated light properties depending on the type of light source.
- It is no more required to call *V3d_Viewer::UpdateLights()* after modifying the properties of light sources (color, position, etc.)

15.2 Shading Models

Graphic3d_AspectFillArea3d has been extended by a new property *ShadingModel()*, which previously has been defined globally for the entire View.

Previously, a triangle array without normal vertex attributes was implicitly considered as unshaded, but now such array will be shaded using *Graphic3d_TOSM_FACET* model (e.g. by computing per-triangle normals). Therefore, *Graphic3d_TOSM_UNLIT* should be explicitly specified to disable shading of triangles array. Alternatively, a material without reflectance properties can be used to disable shading (as before).

15.3 Custom low-level OpenGL elements

The following API changes should be considered while porting custom *OpenGl_Element* objects:

- *OpenGl_ShaderManager::BindFaceProgram()*, *BindLineProgram()*, *BindMarkerProgram()* now take enumeration arguments instead of Boolean flags.

15.4 Changes in BOPAlgo_Section

The public method *BuildSection()* in the class *BOPAlgo_Section* has become protected. The methods *Perform()* or *PerformWithFiller()* should be called for construction of the result of SECTION operation.

15.5 Changes in BRepAdaptor_CompCurve

The method *BRepAdaptor_CompCurve::SetPeriodic* has been eliminated. Since the new version, the method *BRepAdaptor_CompCurve::IsPeriodic()* will always return FALSE. Earlier, it could return TRUE in case if the wire contained only one edge based on a periodic curve.

15.6 Removed features

- The methods *SetDeflection*, *SetEpsilonT*, *SetDiscretize* of the class *IntTools_EdgeFace* have been removed as redundant.
- Deprecated functionality *V3d_View::Export()*, related enumerations Graphic3d_ExportFormat, Graphic3d_↩ SortType as well as optional dependency from gl2ps library have been removed.

15.7 Boolean Operations - Solid Builder algorithm

Previously, the unclassified faces of *BOPAlgo_BuilderSolid* algorithm (i.e. the faces not used for solids creation and located outside of all created solids) were used to form an additional (not closed) solid with INTERNAL orientation. Since the new version, these unclassified faces are no longer added into the resulting solids. Instead, the warning with a list of these faces appears.

The following public methods of the *BOPAlgo_BuilderSolid* class have been removed as redundant:

- *void SetSolid(const TopoDS_Solid& theSolid);*
- *const TopoDS_Solid& Solid() const;*

15.8 Boolean Operation classes in BRepAlgo are deprecated

The API classes in the package BRepAlgo providing access to old Boolean operations are marked as deprecated:

- BRepAlgo_Fuse
- BRepAlgo_Common
- BRepAlgo_Cut
- BRepAlgo_Section Corresponding classes from the package BRepAlgoAPI should be used instead.

15.9 Unification of the Error/Warning reporting system of Application Framework

Class *CDM_MessageDriver* and its descendants have been removed; class *Message_Messenger* is used instead in all OCAF packages. By default, messenger returned by *Message::DefaultMessenger()* is used, thus all messages generated by OCAF are directed in the common message queue of OCCT.

In classes implementing OCAF persistence for custom attributes (those inheriting from *BinMDF_ADriver*, *XmlM*↩ *DF_ADriver*), uses of method *WriteMessage()* should be replaced by call to method *Send()* of the inherited field *myMessageDriver*. Note that this method takes additional argument indicating the gravity of the message (Trace, Info, Warning, Alarm, or Fail).

Class *Message_PrinterOStream* can be used instead of *CDM_COutMessageDriver* to direct all messages to a stream. If custom driver class is used in the application, that class shall be reimplemented inheriting from *Message*↩ *_Printer* instead of *CDM_MessageDriver*. Method *Send()* should be redefined instead of method *Write()* of *CDM*↩ *_MessageDriver*. To use the custom printer in OCAF, it can be either added to default messenger or set into the custom *Message_Messenger* object created in the method *MessageDriver()* of a class inheriting *CDF_Application*.

16 Upgrade to OCCT 7.4.0

16.1 Changes in BRepPrimAPI_MakeRevol algorithm

Previously the algorithm could create a shape with the same degenerated edge shared between some faces. Now it is prevented. The algorithm creates the different copy of this edge for each face. The method *Generated(...)* has been changed in order to apply restriction to the input shape: input shape can be only of type VERTEX, EDGE, FACE or SOLID. For input shape of another type the method always returns empty list.

16.2 Removed features

- The following methods of the class *BRepAlgoAPI_BooleanOperation* have been removed as obsolete or replaced:
  - *BuilderCanWork* can be replaced with *IsDone* or *HasErrors* method.
  - *FuseEdges* removed as obsolete.
  - *RefineEdges* replaced with new method *SimplifyResult*.
- The method *ImagesResult* of the class *BOPAlgo_BuilderShape* has been removed as unused. The functionality of this method can be completely replaced by the history methods *Modified* and *IsDeleted*.
- The method *TrackHistory* of the classes *BOPAlgo_RemoveFeatures* and *BRepAlgoAPI_Defeaturing* has been renamed to *SetToFillHistory*.
- The method *GetHistory* of the class *BRepAlgoAPI_Defeaturing* has been renamed to *History*.
- The classes *BRepAlgo_BooleanOperations* and *BRepAlgo_DSAccess* have been removed as obsolete. Please use the BRepAlgoAPI_$*$ classes to perform Boolean operations.
- *BRepAlgo_DataMapOfShapeBoolean* has been removed as unused.
- *BRepAlgo_DataMapOfShapeInterference* has been removed as unused.
- *BRepAlgo_EdgeConnector* has been removed as unused.
- *BRepAlgo_SequenceOfSequenceOfInteger* has been removed as unused.

16.3 Local Context removal

Previously deprecated Local Context functionality has been removed from AIS package, so that related methods have been removed from AIS_InteractiveContext interface: *HasOpenedContext()*, *HighestIndex()*, *LocalContext()*, *LocalSelector()*, *OpenLocalContext()*, *CloseLocalContext()*, *IndexOfCurrentLocal()*, *CloseAll↩ Contexts()*, *ResetOriginalState()*, *ClearLocalContext()*, *UseDisplayedObjects()*, *NotUseDisplayedObjects()*, *Set↩ ShapeDecomposition()*, *SetTemporaryAttributes()*, *ActivateStandardMode()*, *DeactivateStandardMode()*, *Keep↩ Temporary()*, *SubIntensityOn()*, *SubIntensityOff()*, *ActivatedStandardModes()*, *IsInLocal()*, *AddOrRemoveSelected()* taking TopoDS_Shape.

A set of deprecated methods previously related to Local Context and now redirecting to other methods has been preserved to simplify porting; they will be removed in next release.

16.4 Changes in behavior of Convert algorithms

Now methods *GeomConvert::ConcatG1*, *GeomConvert::ConcatC1*, *Geom2dConvert::ConcatG1*, *Geom2d↩ Convert::ConcatC1* modify the input argument representing the flag of closedness.

16.5 Changes in selection API and picked point calculation algorithm.

SelectBasics_PickResult structure has been extended, so that it now defines a 3D point on the detected entity in addition to Depth value along picking ray. *SelectMgr_SelectingVolumeManager::Overlap()* methods have been corrected to fill in *SelectBasics_PickResult* structure (depth and 3D point) instead of only depth value, so that custom *Select3D_SensitiveEntity* implementation should be updated accordingly (including *Select3D_SensitiveSet* subclasses).

16.6 Document format version management improvement.

Previously Document format version restored by *DocumentRetrievalDriver* was propagated using static methods of the corresponding units (like *MDataStd* or *MNaming*) to static variables of these units and after that became accessible to Drivers of these units. Now Document format version is available to drivers via *RelocationTable*. The Relocation table now keeps *HeaderData* of the document and a format version can be extracted in the following way: *theRelocTable.GetHeaderData()->StorageVersion()*. Obsolete methods: *static void SetDocumentVersion (const Standard_Integer DocVersion)* and *static Standard_Integer DocumentVersion()* have been removed from *BinMDataStd*, *BinMNaming*, *XmlMDataStd* and *XmlMNaming*.

16.7 BRepMesh - revision of the data model

The entire structure of *BRepMesh* component has been revised and separated into several logically connected classes.

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

BRepMesh_FastDiscret and *BRepMesh_FastDiscretFace* classes have been removed.

The following changes have been introduced in the API of *BRepMesh_IncrementalMesh*, component entry point:

- Due to revised logic, *adaptiveMin* parameter of the constructor has been removed as meaningless;
- *BRepMesh_FastDiscret::Parameters* has been moved to a separate structure called *IMeshTools_Parameters*; the signatures of related methods have been changed correspondingly.
- Interface of *BRepMesh_Delaun* class has been changed.

Example of usage: Case 1 (explicit parameters):

```
#include <IMeshData_Status.hxx>
#include <IMeshTools_Parameters.hxx>
#include <BRepMesh_IncrementalMesh.hxx>

Standard_Boolean meshing_explicit_parameters()
{
  BRepMesh_IncrementalMesh aMesher (aShape, 0.1, Standard_False, 0.5, Standard_True);
  const Standard_Integer aStatus = aMesher.GetStatusFlags();
  return !aStatus;
}

Standard_Boolean meshing_new()
{
  IMeshTools_Parameters aMeshParams;
  aMeshParams.Deflection               = 0.1;
  aMeshParams.Angle                    = 0.5;
  aMeshParams.Relative                 = Standard_False;
  aMeshParams.InParallel               = Standard_True;
  aMeshParams.MinSize                  = Precision::Confusion();
  aMeshParams.InternalVerticesMode     = Standard_True;
  aMeshParams.ControlSurfaceDeflection = Standard_True;

  BRepMesh_IncrementalMesh aMesher (aShape, aMeshParams);
  const Standard_Integer aStatus = aMesher.GetStatusFlags();
  return !aStatus;
}
```

16.8 Changes in API of Chamfer algorithms

Some public methods of the class BRepFilletAPI_MakeChamfer are released from excess arguments:

- method Add for symmetric chamfer now takes only 2 arguments: distance and edge;
- method GetDistAngle now takes only 3 arguments: index of contour, distance and angle.

16.9 Aspects unification

Fill Area, Line and Marker aspects (classes *Graphic3d_AspectFillArea3d*, *Graphic3d_AspectLine3d*, *Graphic3d_AspectMarker3d* and *Graphic3d_AspectText3d*) have been merged into new class *Graphic3d_Aspects* providing a single state for rendering primitives of any type. The old per-primitive type aspect classes have been preserved as sub-classes of *Graphic3d_Aspects* with default values close to the previous behavior. All aspects except Graphic3d_AspectFillArea3d define Graphic3d_TOSM_UNLIT shading model.

The previous approach with dedicated aspects per primitive type was handy in simplified case, but lead to confusion otherwise. In fact, drawing points or lines with lighting applied is a valid use case, but only *Graphic3d_AspectFillArea3d* previously defined necessary material properties.

As aspects for different primitive types have been merged, Graphic3d_Group does no more provide per-type aspect properties. Existing code relying on old behavior and putting interleaved per-type aspects into single Graphic3d_Group should be updated. For example, the following pseudo-code will not work anymore, because all *SetGroupPrimitivesAspect* calls will setup the same property:

```
Handle(Graphic3d_Group) aGroup = thePrs->NewGroup();
aGroup->SetGroupPrimitivesAspect (myDrawer->ShadingAspect()->Aspect());
aGroup->SetGroupPrimitivesAspect (myDrawer->LineAspect()->Aspect());    //!< overrides previous aspect

Handle(Graphic3d_ArrayOfSegments) aLines = new Graphic3d_ArrayOfSegments (2);
Handle(Graphic3d_ArrayOfTriangles) aTris = new Graphic3d_ArrayOfTriangles (3);
aGroup->AddPrimitiveArray (aLines); //!< both arrays will use the same aspect
aGroup->AddPrimitiveArray (aTris);
```

To solve the problem, the code should be modified to either put primitives into dedicated groups (preferred approach), or using *SetPrimitivesAspect* in proper order:

```
Handle(Graphic3d_Group) aGroup = thePrs->NewGroup();

aGroup->SetGroupPrimitivesAspect (myDrawer->ShadingAspect()->Aspect());
Handle(Graphic3d_ArrayOfTriangles) aTris = new Graphic3d_ArrayOfTriangles (3);
aGroup->AddPrimitiveArray (aTris);

Handle(Graphic3d_ArrayOfSegments) aLines = new Graphic3d_ArrayOfSegments (2);
aGroup->SetPrimitivesAspect (myDrawer->LineAspect()->Aspect()); //!< next array will use the new aspect
aGroup->AddPrimitiveArray (aLines);
```

16.10 Material definition

Decomposition of Ambient, Diffuse, Specular and Emissive properties has been eliminated within *Graphic3d_MaterialAspect* definition. As result, the following methods of *Graphic3d_MaterialAspect* class have been removed: SetReflectionMode(), SetReflectionModeOn(), Ambient(), Diffuse(), Emissive(), Specular(), SetAmbient(), SetDiffuse(), SetSpecular(), SetEmissive().

Previously, computation of final value required the following code:

```
Graphic3d_MaterialAspect theMaterial; Quantity_Color theInteriorColor;
Graphic3d_Vec3 anAmbient (0.0f);
if (theMaterial.ReflectionMode (Graphic3d_TOR_AMBIENT))
{
  anAmbient = theMaterial.MaterialType (Graphic3d_MATERIAL_ASPECT)
            ? (Graphic3d_Vec3 )theInteriorColor           * theMaterial.Ambient()
            : (Graphic3d_Vec3 )theMaterial.AmbientColor() * theMaterial.Ambient();
}
```

New code looks like this:

```
Graphic3d_MaterialAspect theMaterial; Quantity_Color theInteriorColor;
Graphic3d_Vec3 anAmbient = theMaterial.AmbientColor();
if (theMaterial.MaterialType (Graphic3d_MATERIAL_ASPECT)) { anAmbient *= (Graphic3d_Vec3 )theInteriorColor;
       }
```

Existing code should be updated to:

- Replace Graphic3d_MaterialAspect::SetReflectionModeOff() with setting black color; SetReflectionModeOn() calls can be simply removed. R.g. theMaterial.SetAmbientColor(Quantity_NOC_BLACK).
- Replace Graphic3d_MaterialAspect::Ambient(), SetAmbient(), Diffuse(), SetDiffuse(), Specular(), SetSpecular(), Emissive(), SetEmissive() with methods working with pre-multiplied color. E.g. theMaterial.SetAmbientColor(Graphic3d_Vec3 (1.0f, 0.0f, 0.0f) $*$ 0.2f).
- Avoid using Graphic3d_MaterialAspect::Color() and SetColor() with non-physical materials (Graphic3d_MATERIAL_ASPECT). These materials do not include color definition, because it is taken from Graphic3d_Aspects::InteriorColor() - this has not been changed. However, previously it was possible storing the color with SetColor() call and then fetching it with Color() by application code (the rendering ignored this value); now SetColor() explicitly ignores call for Graphic3d_MATERIAL_ASPECT materials and Color() returns DiffuseColor() multiplication coefficients.

16.11 Changes in Graphic3d_Text and OpenGl_Text API

Parameters of *Text* in *Graphic3d_Group* are moved into a new *Graphic3d_Text* class. *AddText* of *Graphic3d_Group* should be used instead of the previous *Text*.

The previous code:

```
Standard_Real x, y, z;
theAttachmentPoint.Coord(x,y,z);
theGroup->Text (theText,
                Graphic3d_Vertex(x,y,z),
                theAspect->Height(),
                theAspect->Angle(),
                theAspect->Orientation(),
                theAspect->HorizontalJustification(),
                theAspect->VerticalJustification());
```

should be replaced by the new code:

```
Handle(Graphic3d_Text) aText = new Graphic3d_Text (theAspect->Height());
aText->SetText (theText.ToExtString());
aText->SetPosition (theAttachmentPoint);
aText->SetHorizontalAlignment (theAspect->HorizontalJustification());
aText->SetVerticalAlignment (theAspect->VerticalJustification());
theGroup->AddText (aText);
```

OpenGl_Text contains *Graphic3d_Text* field.

OpenGl_TextParam struct is removed. Constructor and *Init* of *OpenGl_Text* with *OpenGl_TextParam* are also removed. Instead of using them, change *OpenGl_Text*.

Please, note, that after modifying *OpenGl_Text*, *Reset* of *OpenGl_Text* should be called.

FormatParams of *OpenGl_Text* is replaced by *Text*.

16.12 Presentation invalidation

Historically AIS_InteractiveObject provided two independent mechanisms invalidating presentation (asking presentation manager to recompute specific display mode or all modes):

1. *AIS_InteractiveObject::SetToUpdate()*, marking existing presentation for update. This is main invalidation API, which is expected to be followed by *AIS_InteractiveContext::Update()* call.

2. *AIS_InteractiveObject::myToRecomputeModes* + *myRecomputeEveryPrs*. This is auxiliary invalidation API, used internally by AIS_InteractiveContext::SetColor()/UnsetColor() and similar modification methods.

The latter one has been removed to avoid confusion and unexpected behavior. In addition, two methods *AIS_InteractiveObject::Update()* have been deprecated in favor of new *AIS_InteractiveObject::UpdatePresentations()* recomputing only invalidated presentations.

Custom presentations implementing interface methods *AIS_InteractiveObject::SetColor()* and others should be revised to use *AIS_InteractiveObject::SetToUpdate()* or updating presentation without recomputation (see *AIS_InteractiveObject::SynchronizeAspects()* and *AIS_InteractiveObject::replaceAspects()*).

16.13 Interior styles

- *Aspect_IS_HOLLOW* is now an alias to *Aspect_IS_EMPTY* and does not implicitly enables drawing mesh edges anymore. Specify Graphic3d_AspectFillArea3d::SetDrawEdges(true) with Graphic3d_AspectFillArea3d::SetInteriorStyle(Aspect_IS_EMPTY) to get previous behavior of Aspect_IS_HOLLOW style.
- *Aspect_IS_HIDDENLINE* does not implicitly enables drawing mesh edges anymore. Specify Graphic3d_AspectFillArea3d::SetDrawEdges(true) with Graphic3d_AspectFillArea3d::SetInteriorStyle(Aspect_IS_HIDDENLINE) to get previous behavior of Aspect_IS_HIDDENLINE style.

16.14 PrsMgr and SelectMgr hierarchy clean up

Proxy classes *Prs3d_Presentation*, *PrsMgr_ModedPresentation* and *PrsMgr_Prs* have been removed. Code iterating through the list of low-level structures AIS_InteractiveObject::Presentations() should be updated to access PrsMgr_Presentation directly. Forward declarations of *Prs3d_Presentation* should be corrected, since it is now a typedef to *Graphic3d_Structure*.

Proxy classes *SelectBasics_SensitiveEntity* and *SelectBasics_EntityOwner* have been removed - *Select3D_SensitiveEntity* and *SelectMgr_EntityOwner* should be now used directly instead.

16.15 Polygon offset defaults

Graphic3d_PolygonOffset default constructor has been corrected to define Units=1 instead of Units=0. Default polygon offset settings Mode=Aspect_POM_Fill + Factor=1 + Units=1 are intended to push triangulation (Shaded presentation) a little bit behind of lines (Wireframe and Face Edges) for reducing z-fighting effect of Shaded+Wireframe combination. The change in defaults (Units changed from 0 to 1) is intended to cover scenario when camera direction is perpendicular to model plane (like 2D view).

Application observing unexpected visual difference on this change should consider customizing this property within AIS_InteractiveContext default attributes or on per-presentation basis via *Graphic3d_Aspects::SetPolygonOffset()* methods.

16.16 Adding ZLayers in given position

Interface of insertion ZLayer in the viewer has been improved with ability to insert new layer before or after existing one. Previously undocumented behavior of *V3d_Viewer::AddZlayer()* method has been corrected to insert new layer before *Graphic3d_ZLayerId_Top*. Applications might need revising their custom layers creation code and specify precisely their order with new methods *V3d_Viewer::InsertLayerBefore()* and *V3d_Viewer::InsertLayerAfter()*.

16.17 Modified enumerations

Applications using integer values of the following enumerations in persistence should be corrected as these enumerations have been modified:

Name
AIS_TypeOfAttribute
Aspect_InteriorStyle
Font_FontAspect

16.18 Custom defines within env.bat

env.bat produced by Visual Studio project generator *genproj.bat* has been modified so that *CSF_DEFINES%* variable is reset to initial state. Custom building environment relying on old behavior and setting extra macros within *CSF_DEFINES%* before env.bat should be updated to either modify custom.bat or setup new variable *CSF_DEF↩ INES_EXTRA%* instead.

16.19 Switching Boolean Operations algorithm to use BVH tree instead of UB tree

Since OCCT 7.4.0 Boolean Operations algorithm uses BVH tree instead of UBTree to find the pairs of entities with interfering bounding boxes. The following API changes have been made:

- BOPTools_BoxBndTree and BOPTools_BoxBndTreeSelector have been removed. Use the BOPTools_Box↩ Tree and BOPTools_BoxTreeSelector instead.
- BOPTools_BoxSelector::SetBox() method now accepts the BVH_Box instead of Bnd_Box.
- Methods BOPTools_BoxSelector::Reject and BOPTools_BoxSelector::Accept have been removed as unused.
- The RunParallel flag has been removed from the list of parameters of BOPAlgo_Tools::IntersectVertices method. Earlier, it performed selection from the UB tree in parallel mode. Now all interfering pairs are found in one pass, using pair traverse of the same BVH tree.

16.20 Standard_Stream.hxx no more has "using std::" statements

Standard_Stream.hxx header, commonly included by other OCCT header files, does no more add entities from *std namespace* related to streams (like *std::cout*, *std::istream* and others) into global namespace. The application code relying on this matter should be updated to either specify std namespace explicitly (like std::cout) or add "using std::" statements locally.

17 Upgrade to OCCT 7.5.0

17.1 RGB color definition

OCCT 3D Viewer has been improved to properly perform lighting using in linear RGB color space and then convert result into non-linear gamma-shifted sRGB color space before displaying on display. This change affects texture mapping, material definition and color definition.

Previously *Quantity_Color* definition was provided with unspecified RGB color space. In practice, mixed color spaces have been actually used, with non-linear sRGB prevailing in general. Since OCCT 7.5.0, *Quantity_Color* now specifies that components are defined in linear RGB color space.

This change affects following parts:

- Standard colors defined by *Quantity_NameOfColor* enumeration have been converted into linear RGB values within Quantity_Color construction.
- Application may use new enumeration value *Quantity_TOC_sRGB* for passing/fetching colors in sRGB color space, which can be useful for interoperation with color picking widgets (returning 8-bit integer values within [0..255] range) or for porting colors constants within old application code without manual conversion.
- *Graphic3d_MaterialAspect* color components are now expected in linear RGB color space, and standard OCCT materials within *Graphic3d_NameOfMaterial* enumeration have been updated accordingly.
- Texture mapping now handles new *Graphic3d_TextureRoot::IsColorMap()* for interpreting content in linear RGB or sRGB color space. It is responsibility of user specifying this flag correctly. The flag value is TRUE by default.
- Method *Image_PixMap::PixelColor()* has been extended with a new Boolean flag for performing linearization of non-linear sRGB. This flag is FALSE by default; application should consider passing TRUE instead for further handling *Quantity_Color* properly as linear RGB values.

17.2 Aspect_Window interface change

Unexpected const-ness of Aspect_Window::DoResize() method has been removed, so that application classes implementing this interface should be updated accordingly.

17.3 Renaming of types

Enumeration BRepOffset_Type is renamed to ChFiDS_TypeOfConcavity.

17.4 change in construction of offset faces

Now by default offset faces of non-planar faces may be planar faces if their originals can be approximated by planes.

17.5 TKV3d/TKService toolkits changes

The following changes could be highlighted while porting:

- *Prs3d::GetDeflection()* has been moved to *StdPrs_ToolTriangulatedShape::GetDeflection()*.
- *Prs3d_ShapeTool* has been moved to *StdPrs_ShapeTool*.
- *StdSelect_ViewerSelector3d* has been moved to *SelectMgr_ViewerSelector3d*.
- *Font_BRepFont* has been moved to *StdPrs_BRepFont*.

- Visualization classes now use *TopLoc_Datum3D* (from *TKMath*) instead of *Geom_Transformation* (from *TKG3d*) as smart pointer to *gp_Trsf*. This is rather an internal change, but some applications might need to be updated.

17.6 Prs3d_Drawer deviation angle

Properties Prs3d_Drawer::HLRAngle() and Prs3d_Drawer::HLRDeviationCoefficient() have been removed from classes *Prs3d_Drawer*, *AIS_Shape* and *AIS_InteractiveContext*. Prs3d_Drawer::DeviationAngle() should be now used instead of Prs3d_Drawer::HLRAngle() and Prs3d_Drawer::DeviationCoefficient() instead of Prs3d_Drawer::HLRDeviationCoefficient(). The default value of Prs3d_Drawer::DeviationAngle() property has been changed from 12 to 20 degrees to match removed Prs3d_Drawer::HLRAngle(), previously used as input for triangulation algorithm.

17.7 Changes in HLR presentation API

Methods computing HLR presentation within *PrsMgr_PresentableObject::Compute()* have been renamed to *PrsMgr_PresentableObject::computeHLR()* and now accept *Graphic3d_Camera* object instead of removed *Prs3d_Projector*.

17.8 Dimension and Relation presentations moved from AIS to PrsDim

Presentation classes displaying Dimensions and Relations have been moved from *AIS* package to *PrsDim*. Corresponding classes should be renamed in application code (like *AIS_LengthDimension -> PrsDim_LengthDimension*).

17.9 Select3D_SensitiveEntity interface change

The method Select3D_SensitiveEntity::NbSubElements() has been changed to be constant. Select3D_SensitiveEntity subclasses at application level should be updated accordingly.

17.10 Changes in Boolean operations algorithm

- TreatCompound method has been moved from *BOPAlgo_Tools* to *BOPTools_AlgoTools*. Additionally, the map parameter became optional:

```
void BOPTools_AlgoTools::TreatCompound (const TopoDS_Shape& theS,
                                        TopTools_ListOfShape& theLS,
                                        TopTools_MapOfShape* theMap = NULL);
```

17.11 Offset direction change

Offset direction, which used in class Adaptor2d_OffsetCurve for evaluating values and derivatives of offset curve is unified for offset direction used in class Geom2d_OffsetCurve: now offset direction points to outer ("right") side of base curve instead of the previously used inner ("left") side. Old usage of class in any application should be changed something like that:

Adaptor2d_OffsetCurve aOC(BaseCurve, Offset) –> Adaptor2d_OffsetCurve aOC(BaseCurve, -Offset)

17.12 Change of progress indication API

The progress indication mechanism has been revised to eliminate its weak points in previous design (leading to implementation mistakes). Redesign also allows using progress indicator in multi-threaded algorithms in more

straight-forward way with minimal overhead. Note however, that multi-threaded algorithm should pre-allocate per-task progress ranges in advance to ensure thread-safety - see examples in documentation of class Message_↩ ProgressScope for details.

Classes Message_ProgressSentry and Message_ProgressScale have been removed. New classes Message_↩ ProgressScope and Message_ProgressRange should be used as main API classes to organize progress indication in the algorithms. Instances of the class Message_ProgressRange are used to pass the progress capability to nested levels of the algorithm, and an instance of the class Message_ProgressScope is to be created (preferably as local variable) to manage progress at each level of the algorithm. The instance of Message_ProgressIndicator is not passed anymore to sub-algorithms. See documentation of the class Message_ProgressScope for more details and examples.

Methods to deal with progress scopes and to advance progress are removed from class Message_Progress↩ Indicator; now it only provides interface to the application-level progress indicator. Virtual method Message↩ _ProgressIndicator::Show() has changed its signature and should be updated accordingly in descendants of Message_ProgressIndicator. The scope passed as argument to this method can be used to obtain information on context of the current process (instead of calling method GetScope() in previous implementation). Methods Show(), UserBreak(), and Reset() are made protected in class Message_ProgressIndicator; methods More() or UserBreak() of classes Message_ProgressScope or Message_ProgressRange should be used to know if the cancel event has come. See documentation of the class Message_ProgressIndicator for more details and implementation of Draw_ProgressIndicator for an example.

Let's take a look onto typical algorithm using an old API:

```
class MyAlgo
{
public:
  //! Algorithm entry point taking an optional Progress Indicator.
  bool Perform (const TCollection_AsciiString& theFileName,
                const Handle(Message_ProgressIndicator)& theProgress = Handle(Message_ProgressIndicator)())
  {
    Message_ProgressSentry aPSentry (theProgress, (TCollection_AsciiString("Processing ") +
       theFileName).ToCString(), 2);
    {
      Message_ProgressSentry aPSentry1 (theProgress, "Stage 1", 0, 153, 1);
      for (int anIter = 0; anIter < 153; ++anIter, aPSentry1.Next())
      {
        if (!aPSentry1.More()) { return false; }
        // do some job here...
      }
    }
    aPSentry.Next();
    {
      perform2 (theProgress);
    }
    aPSentry.Next();
    bool wasAborted = !theProgress.IsNull() && theProgress->UserBreak();
    return !wasAborted;
  }

private:
  //! Nested sub-algorithm taking Progress Indicator.
  bool perform2 (const Handle(Message_ProgressIndicator)& theProgress)
  {
    Message_ProgressSentry aPSentry2 (theProgress, "Stage 2", 0, 100, 1);
    for (int anIter = 0; anIter < 100 && aPSentry2.More(); ++anIter, aPSentry2.Next()) {}
    return !aPSentry2.UserBreak();
  }
};

// application executing an algorithm
Handle(Message_ProgressIndicator) aProgress = new MyProgress();
MyAlgo anAlgo;
anAlgo.Perform ("FileName", aProgress);
```

The following guidance can be used to update such code:

- Replace `const Handle(Message_ProgressIndicator)&` with `const Message_Progress↩ Range&` in arguments of the methods that support progress indication. Message_ProgressIndicator object should be now created only at place where application starts algorithms.
- Replace `Message_ProgressSentry` with `Message_ProgressScope`. Take note that Message_↩ ProgressScope has less arguments (no "minimal value"). In other aspects, Message_ProgressScope mimics an iterator-style interface (with methods More() and Next()) close to the old Message_ProgressSentry

(pay attention to extra functionality of Message_ProgressScope::Next() method below). Note that method Message_ProgressScope::Close() is equivalent of the method Relieve() of Message_ProgressSentry in previous version. Class Message_ProgressSentry is still defined (marked as deprecated) providing API more close to old one, and can be still used to reduce porting efforts.

- Each Message_ProgressScope should take the next Range object to work with. Within old API, Message_ProgressSentry received the root Progress Indicator object which mantained the sequence of ranges internally. Message_ProgressScope in new API takes Message_ProgressRange, which should be returned by Message_ProgressScope::Next() method of the parent scope. Do not use the same Range passed to the algorithm for all sub-Scopes like it was possible in old API; each range object may be used only once.

Take a look onto ported code and compare with code above to see differences:

```
class MyAlgo
{
public:
  //! Algorithm entry point taking an optional Progress Range.
  bool Perform (const TCollection_AsciiString& theFileName,
                const Message_ProgressRange& theProgress = Message_ProgressRange())
  {
    Message_ProgressScope aPSentry (theProgress, TCollection_AsciiString("Processing ") + theFileName, 2);
    {
      Message_ProgressScope aPSentry1 (aPSentry.Next(), "Stage 1", 153);
      for (int anIter = 0; anIter < 153; ++anIter, aPSentry1.Next())
      {
        if (!aPSentry1.More()) { return false; };
        // do some job here...
      }
    }
    {
      perform2 (aPSentry.Next());
    }
    bool wasAborted = aPSentry.UserBreak();
    return !wasAborted;
  }

  //! Nested sub-algorithm taking Progress sub-Range.
  bool perform2 (const Message_ProgressRange& theProgress)
  {
    Message_ProgressScope aPSentry2 (theProgress, "Stage 2", 100);
    for (int anIter = 0; anIter < 100 && aPSentry2.More(); ++anIter, aPSentry2.Next()) {}
    return !aPSentry2.UserBreak();
  }
};

// application executing an algorithm
Handle(Message_ProgressIndicator) aProgress = new MyProgress();
MyAlgo anAlgo;
anAlgo.Perform ("FileName", aProgress->Start());
```

17.13 Message_Messenger interface change

Operators $<<$ with left argument *Handle(Message_Messenger)*, used to output messages with a stream-like interface, have been removed. This functionality is provided now by separate class *Message_Messenger::StreamBuffer*. That class contains a stringstream buffer which can be filled using standard stream operators. The string is sent to a messenger on destruction of the buffer object, call of its method Flush(), or using operator $<<$ with one of ostream manipulators (*std::endl, std::flush, std::ends*). Manipulator *Message_EndLine* has been removed, *std::endl* should be used instead.

New methods *SendFail(), SendAlarm(), SendWarning(), SendInfo()*, and *SendTrace()* are provided in both *Message_Messenger* class and as static functions in *Message* package (short-cuts to default messenger), returning buffer object for the output of corresponding type of the message.

The code that used operator $<<$ for messenger, should be ported as follows.

Before the change:

```
Handle(Message_Messenger) theMessenger = ...;
theMessenger << "Value = " << anInteger << Message_EndLine;
```

After the change, single-line variant:

```
Handle(Message_Messenger) theMessenger = ...;
theMessenger->SendInfo() << "Value = " << anInteger << std::endl;
```

After the change, extended variant:

```
Handle(Message_Messenger) theMessenger = ...;
Message_Messenger::StreamBuffer aSender = theMessenger->SendInfo();
aSender << "Array: [ ";
for (int i = 0; i < aNb; ++i) { aSender << anArray[i] << " "; }
aSender << "]" << std::endl; // aSender can be used further for other messages
```

17.14 Message_Printer interface change

Previously, sub-classes of *Message_Printer* have to provide a triplet of *Message_Printer::Send()* methods accepting different string representations: TCollection_AsciiString, TCollection_ExtendedString and Standard_CString. *Message_Printer* interface has been changed, so that sub-classes now have to implement only single method *Message_Printer::send()* accepting TCollection_AsciiString argument and having no Endl flag, which has been removed. Old three Message_Printer::Send() methods remain defined virtual with unused last argument and redirecting to new send() method by default.

17.15 Prs3d_Root deprecation

Redundant class Prs3d_Root has been marked as deprecated - Prs3d_Presentation::NewGroup() should be called directly.

17.16 Support of multiple OCAF application instances

Class *CDF_Session* has been removed. That class was used to store global instance of OCAF application (object of class *CDM_Application* or descendant, typically *TDataStd_Application*). Global directory of all opened OCAF documents has been removed as well; such directory is maintained now by each instance of the *CDM_Application* class.

This allows creating programs that work with different OCAF documents concurrently in paralel threads, provided that each thread deals with its own instance of *TDataStd_Application* and documents managed by this instance.

Note that neither *TDataStd_Application* nor *TDocStd_Document* is protected from concurrent access from several threads. Such protection, if necessary, shall be implemented on the application level. For an example, access to labels and attributes could be protected by mutex if there is a probability that different threads access the same labels / attributes:

```
{
  Standard_Mutex::Sentry aSentry (myMainLabelAccess);
  TDF_Label aChildLab = aDocument->Main().NewChild();
  TDataStd_Integer::Set(aChildLab, 0);
}
```

17.17 Draw Harness hotkeys

Draw Harness hotkeys **W** (Wireframe) and **S** (Shaded) have been re-mapped to **Ctrl+W** and **Ctrl+S**. Hotkey **A** has been remapped to **Backspace**. Hotkeys WASD and Arrays are now mapped for walk-through navigation in 3D Viewer.

17.18 Utf-8 encoding for message files

Message files (with extension .msg) are now expected to be in UTF-8 encoding (unless they have UTF-16 BOM in which case UTF-16 is expected). This allows using arbitrary Unicode symbols for localization of messages.

Existing message files containing 8-bit characters (previously interpreted as characters from Latin-1 code block) should be converted to UTF-8.