Open CASCADE 6.2 Minor Public Release

Release Notes

Overview

Open CASCADE Technology 6.2 is a minor public release, which includes new features, improvements and bug fixes, over minor release 6.1 and maintenance release 6.1.1.

Version **6.2** is binary incompatible with the previous versions of Open CASCADE Technology, so applications linked against a previous version must be recompiled to run with this Version 6.2.

Note that some changes in Open CASCADE API may require a corresponding update of your applications. Please, consult the **Changes** section for more details.

Table of Contents

- New Features
  - Foundation Classes
  - Visualization
  - Application Framework
  - Modeling Algorithms
  - Products
  - Porting
  - Building Tools
- Improvements
  - Foundation Classes
  - Modeling Algorithms
  - Visualization
  - Data Exchange
  - Draw Test Harness
  - WOK
  - Documentation
  - Products
- Changes
  - Foundation Classes
  - Visualization
  - Data Exchange
  - Products
- Appendix: Bug Fixes

Highlights

- Open CASCADE
  - Implementation of multithread safety in OCCT Kernel
  - Improvement of Exception mechanism on Linux and UNIX
  - Implementation of new visualization approach based on OpenGL arrays for AIS shapes
  - Numerous other improvements in Visualization
  - Porting of OCCT to Microsoft Visual Studio 2005
  - Porting of OCCT to SGI 64 bit
- Products
  - Parasolid reader now supports schema numbers up to 17
  - Implementation of Surface Modification and Surface Curvature Analysis algorithms to SSP product;
  - Numerous improvements in OMF
  - Redesign of Products deliveries including new installation procedures

New Features

Foundation Classes

- Basic support for multithreading has been provided at the level of TKernel services of OCCT:
  - New classes `Standard_Mutex` and `OSD_Thread` provide encapsulation of functions provided by the operating system to manipulate mutexes and manage threads in the uniform way, adapted for usage with Open CASCADE.
  - Open CASCADE optimized memory manager has been protected for safe work within multithread applications where different threads can access it simultaneously. Note that this feature is not activated by default with the purpose of optimal performance. Environment variable `MMGT_REENTRANT` must be set to 1 to activate this feature. Note that setting variable `MMGT_OPT` to 0 (i.e. using CRT memory heap directly) is also thread safe.

    Special note: for applications that heavily use OCCT memory manager from more than one thread, on multiprocessor hardware, execution with option `MMGT_OPT=0` can be faster than with option `MMGT_REENTRANT=1`.
  - OCCT exceptions and signal handling, runtime type identification (RTTI), and some other low-level services are protected to be thread-safe.

  Note that these improvements provide only basic features for using OCCT in multithreaded applications; the application itself is responsible for ensuring safe access to both its own data and not-yet-protected data of OCCT.

- Four new classes have been added to the package `Bnd`, to be used in performance-critical algorithms
  - `Bnd_B2d`: 2-dimensional box using double-precision floating point
  - `Bnd_B3d`: 3-dimensional box using double-precision floating point
  - `Bnd_B2f`: 2-dimensional box using single-precision floating point
  - `Bnd_B3f`: 3-dimensional box using single-precision floating point

  These boxes have the following differences from `Bnd_Box` and `Bnd_Box2d`:
  - New boxes are always limited that improves the performance of some algorithms.
  - All new types consume less memory, particularly when single-precision boxes are used.
  - Line-3dbox intersection (`Bnd_B3x::IsOut(theLine)`) now has the 2nd parameter telling if the line is treated as a ray; this improvement is aimed primarily at aiding 3D visual selection. The new box types are compatible with `NCollection_UBTree`, they can be used to instantiate AABB-trees in the same way as `Bnd_Box` and `Bnd_Box2d`.
  - Class `Bnd_B2x` (hence the classes Bnd_B2f and Bnd_B2d) provide two additional intersection checks: with another box transformed with a given transformation, and with a circle
  - Class `Bnd_B3x` (hence the classes Bnd_B3f and Bnd_3d) provide two additional intersection checks: with another box transformed with a given transformation, and with a sphere.

- New macro definition has been added to `Standard_Version.hxx: OCC_VERSION_HEX`, which defines complete version number of Open CASCADE Technology as a hex number (with two positions per each major, minor, and maintenance version numbers). This facilitates checks for precise version number of OCCT. For example, to check if the current version is OCCT 6.1.1 or above, input: `#if OCC_VERSION_HEX >= 0x060101`

- Class `TColStd_PackedMapOfInteger` now provides Boolean operations between maps (considered as sets of integers): intersection (boolean AND), union (boolean OR), subtraction (or relative complement), and difference (boolean XOR). Each operation is provided in two variants: one is applied to the first operand and stores result in the same map (methods `Intersect()`, `Unite()`, `Subtract()`, and `Differ()`), another stores result in third map (methods `Intersection()`, `Union()`, `Subtraction()`, and `Difference()`) for which the method is called.

  Empty maps can be used anywhere as operands.

  Two other methods added that provide comparison operations on maps: check for equality (method `IsEqual()`), and check for inclusion of all items of the map in another one (method `IsSubset()`).

- New class `TColStd_HPackedMapOfInteger` has been added. This class encapsulates `TColStd_PackedMapOfInteger` class allowing to operate it by Handle. Methods `Map()` and `ChangeMap()` should be used to access encapsulated map object.

- New method `Added()` is provided in class `NCollection_Map`. This method adds a new key to the map (unless the same key is already there) and returns const reference to the corresponding key actually contained in the map. Thus it provides access to the object instances contained in the map; this can be necessary when the map is used as a means to ensure uniqueness (by value) of instances of objects of a particular type.

- The new method `RotationPart()` has been added to the class `gp_Trsf2d` to obtain a rotation angle corresponding to rotational component of the transformation.

Visualization

- Interface of `MeshVS_NodalColorPrsBuilder` has been extended with new methods for building presentations using color scale interpolation by each element in accordance with arbitrary color scale (in addition to existing RGB interpolation). It is built by means of texture mapping. It is necessary to perform the following steps to specify parameters of interpolation:
  - Set the required type of interpolation with `UseTexture` method call (RGB interpolation is used by default for compatibility with the previous version)
  - Specify colors for texture generation with `SetColorMap` method call
  - Specify correspondence between node identifiers and corresponding texture coordinates (range [0, 1]) with `SetTextureCoords` method call

- The possibility to activate dynamic highlighting of already selected objects during mouse movement has been implemented. A new method `SetToHilightSelected()` has been added for this purpose in the class `AIS_InteractiveContext`. `SetToHilightSelected (Standard_True)` method call activates dynamic highlighting of selected objects. By default, selected objects are not highlighted during mouse movements.

- Two new parameters of mesh visualization in shading mode have been added in `MeshVS`, allowing to display mesh in either smooth or flat color mode and manipulate treatment of reflections by OpenGl:

  To enable the default shading mode you should set *MeshVS_DA_Reflection* flag of *MeshVS_Drawer* to `Standard_True` and *MeshVS_DA_SmoothShading* to `Standard_False`.

  To enable smooth shading mode you should set *MeshVS_DA_Reflection* flag of *MeshVS_Drawer* to `Standard_True` and *MeshVS_DA_SmoothShading* to `Standard_True.`

To enable flat shading mode you should set *MeshVS_DA_Reflection* flag of *MeshVS_Drawer* to `Standard_False` and *MeshVS_DA_SmoothShading* to `Standard_False`.

- New drawer attribute `MeshVS_DA_ColorReflection` has been added. This Boolean attribute, when set to true, turns on material reflectance on colored 3D representation of data associated with mesh entities, similarly to shaded mesh representation, to make the presentation look more realistic.

  The attribute value is taken into account by `MeshVS_ElementColorPrsBuilder` and `MeshVS_NodalColorPrsBuilder` classes. These classes computes normal vectors with help of `MeshVS_DataSource::GetNormalsByElement()` new method to produce a correct lighting effect. In general, colored 3D data visualization works slower when material reflectance is on.

- New drawing attribute '`MeshVS_DA_SupressBackFaces`' has been added into MeshVS presentation that allows to manage drawing of back faces of the displayed mesh. Its value by default is false, i.e. both front and back faces are displayed.

Application Framework

- Virtual API has been added in the `TDocStd_Application` class for transaction events. This improvement allows the application to receive signals from OCAF document about each of the events: Open, Commit, and Abort Transaction.

  Every time a document transaction event occurs, the application (class `TDocStd_Application`) that owns the OCAF document calls the corresponding virtual method:

  ```
  OnOpenTransaction (aDocument)
  OnCommitTransaction (aDocument)
  OnAbortTransaction (aDocument)
  ```

  where "aDocument" is a reference to the instance of concerned OCAF document. By default these methods do nothing, but the developer can redefine them in his/her Application subclass, attaching the relevant GUI actions or other events.

  If this API is inconvenient (e.g., for OCAF documents that are not owned by Application), the developer can redefine for the similar purposes the corresponding protected methods of the class `TDocStd_Document`:

  ```
  OpenTransaction()
  CommitTransaction()
  AbortTransaction()
  ```

Modeling Algorithms

- Two new Shape Healing operators have been implemented:

  1. Tool for the division of faces in the shape by the maximum allowed area criterion - `ShapeUpgrade_ShapeDivideArea`

     After applying this operator all faces with the area of more than the specified maximum area will be split.

  2. Tool for removing from faces inner holes having the area less than the specified minimum area - `ShapeUpgreade_RemoveInternalWires`

     This tool considers each face from the whole shape and removes all internal wires having the contour area less than minimum allowed area. Then if the value of `RemoveFaceMode` flag is set to TRUE then separate faces or a group of faces having outer wires consisting only of edges belonging to the removed internal wires (seam edges are not taken into account) are removed.

See Shape Healing User Guide for more details.

Products

DXF

- New global Boolean parameter `read.dxf.anonymous.blocks` has been added in DXF reader. This parameter specifies whether anonymous blocks in a DXF file should be translated or not. Anonymous blocks contain geometrical representation of `DIMENSION`, `HATCH`, and other application-defined constructs not belonging to the geometry of the drawing itself, so their translation may be undesirable. By default, this parameter is enabled, i.e. such blocks are translated.

OMF

- Smooth shading visualization mode has been added in OMF. In this mode, the normal vector is taken at each node of each mesh element and passed to OpenGL to achieve a realistic image of the presentable mesh object. Old flat color mode is also available.

- Possibility to select interactively only directly visible (i.e. not covered by other mesh elements on the current view) mesh nodes or elements has been added. It is implemented through the standard mechanism of selection filters of OCCT visualization. For that purpose new filter classes `SMDSSelect_VisibleFilter and SMDSMeshVS_ VisibleFilter` have been developed as direct descendants of `SelectMgr_Filter.` Support of this feature has been added in OMF Sample and can be evaluated there.

- Now OMF Sample uses `MeshVS` presentation instead of `SMDS_Inter. SMDSMeshVS` has been modified to support 3D elements. The order of nodes should be correct. There is still a possibility to use `SMDS_Inter` by commenting `"#define USE_MESHVS"` in `"StdAfx.h"` file.

Surfaces from Scattered Points

- New Surface Modification algorithm defines surface point displacement set by the user with the help of the appropriate GUI. On the algorithmic level the requested displacement is defined by U,V coordinates on the initial surface S0 and the final point P that must have the same U,V coordinates on the modified surface S.

  Two methods of surface modification have been implemented.
  - The first method uses only one point displacement directly requested by the user and some additional constraints to fix boundaries, if necessary. Any surface boundary is fixed, if the U,V space distance between the modified point and the boundary exceeds the given value (algorithm parameter).
  - The second method uses the number of displacements (algorithm parameter) to calculate surface point displacement basing on the main displacement with the help of the user defined function.
- New Surface Curvature Analysis algorithm facilitates the analysis of the curvature of a surface at its boundaries (edges) and interior.

  At input the algorithm takes a `TopoDS_Shape` (single face, shell, solid or compound of faces). At output it computes curvature values (Gaussian, Mean, Max or Min), and a normal curvature vector at the set of points on the relevant faces, corresponding to the vertices of triangulation on these faces.

Basing on that algorithm, the SSP sample application provides visual means for surface curvature analysis. The interior curvature of the surface is visualized by mapping curvature values to surface color according to Color Scale. The curvature at boundaries is displayed by straight segments directed at the center of the curvature at each point.

Porting

- Open CASCADE Technology has been ported to SGI 64 bit. Open CASCADE Technology can now be complied on this platform. Please, see the "Building Modules" section of OCCT Introduction to Reference Documentation (HTML format) for details.
- A contribution from Prof. Christophe Geuzaine (University of Liege, Electrical Engineering and Computer Science http://www.montefiore.ulg.ac.be/~geuzaine): a patch has been added in order to provide OCCT compilation on MAC OS and gcc compiler on Windows platform.

Building Tools

- A new preprocessor macro `_OCC64` has been added that allows to know, at compile time, whether the code is compiled for 64- or 32-bit platform. This knowledge is necessary for platform-specific code such as selection of value of UndefinedHandleAddress, memory block allocation granularity, etc. The `–D_OCC64` key must be added to CFLAGS and CXXFLAGS when compiling on a 64-bit platform.
- In order to support multithreading in OCCT and applications based thereon it/they must be linked with the `thread` library. To do so `–lthread` string must be added to LDFLAGS.

Improvements

Foundation Classes

- Declaration of basic OCCT classes such as `Standard_Transient` and WOK templates for generation of Handle classes have been improved to optimize the creation and destruction of Handle objects and other typical low-level operations with classes manipulated by such Handles as `IsKind()` and `DownCast()`.

  Note that method `Standard_Transient::IsKind()` is not virtual anymore, and it is not redefined by every descendant class.

- Implementation of Open CASCADE optimized memory manager has been revised to reduce overhead for memory block header (now 4 bytes on 32-bit platforms) and granularity of memory allocation (now 8 bytes; was 16).

  `Method Standard::Purge()` now returns integer: non-zero in case if some memory has been actually released, or 0 otherwise. This allows using that method in C++ new handler to try releasing memory recycled by OCCT memory manager in out-of-memory situations.

- The size of bucket arrays in the implementation of hash maps in both `TCollection` and `NCollection` packages has been optimized to increase performance, especially for maps containing big numbers of items (10^5 - 10^6 and more).

- Macros `DEFINE_STANDARD_HANDLE`, `IMPLEMENT_STANDARD_HANDLE`, `DEFINE_STANDARD_RTTI`, and `IMPLEMENT_STANDARD_RTTI*` (see Foundation Classes User's Guide) have been improved. Now it is possible to declare transient classes inside namespaces, for instance:

```
========== in *.H ====================
#include <Standard_DefineHandle.hxx>

namespace Foo
{

class Bar : public Standard_Transient
{
   ...
public:
  DEFINE_STANDARD_RTTI (Bar)
}; // end of class Bar

// Definition of HANDLE object using Standard_DefineHandle.hxx
DEFINE_STANDARD_HANDLE (Bar, Standard_Transient )
} // end of namespace Foo
========== in *.CPP ====================
namespace Foo {
IMPLEMENT_STANDARD_HANDLE (Bar, Standard_Transient)
IMPLEMENT_STANDARD_RTTIEXT(Bar, Standard_Transient)
}
...
Foo::Handle_Bar anObject = new Foo::Bar();
...
using namespace Foo;
Handle(Bar) aCopyObject = anObject;
```

- It is now possible to change the priority of a thread before or during its execution. The method `SetPriority` of the class `OSD_Thread` has integer argument that can be negative, zero or positive. By default, the value 0 is assumed. When this method is called, the priority of the thread is set relative to the one of the process: negative argument defines lower and positive -- higher priority.

- A memory leak has been eliminated concerning the use of class `TColStd_PackedMapOfInteger`. Proper destructor has been added to this class.

- The percent sign (%) has been added to the list of measurement units supported by UnitsAPI package.
- Implementation of methods `gp_Vec::Mirror(gp_Vec)` and `gp_Vec::Mirror(gp_Ax2)` has been corrected.
- Method `gp_GTrsf::SetAffinity(const gp_Ax2&, const double)` has been corrected to create a transformation that now produces a correct result, according to the description in gp_GTrsf.cdl: "Transformation of any point P into a point P' such that if H is the orthogonal projection of P on the plane A2, the vectors HP and HP' satisfy: HP' = Ratio * HP".

Modeling Algorithms

- The memory leak in the class `TopExp_Explorer` has been eliminated. Code removing previously created objects of type `TopoDS_Iterator` has been added in the methods `Init()` and `Clear()`.
- The algorithm for Extrema calculation has been improved to be more precise if one of the arguments is an infinite face or an infinite edge.
- The method `BRepExtrema_DistShapeShape` has been improved to correctly process edges based on curves with C0 continuity (non-continuous first derivative).

  The algorithm of 2d point on face classification used in Boolean operations has been improved to ensure correct processing of the cases when the test ray passes through vertex on the face boundary
- The sewing algorithm has been improved to provide correct Same Parameter flag on seam edges for cylinders.
- A lot of uninitialized fields have been cleaned in packages Extrema, math, ProjLib, Blend, etc.
- Earlier the realization of `SameParameter` raised memory exception on BSplines with the number of knots exceeding 1024. Now a protection has been added, and BSplines with more than 1024 knots are simply ignored; the function returns failed status.
- Shape Healing classes have been modified to avoid exceptions on internal vertices and edges that may be present in the shape (non-manifold case) and to keep them in the resulting shape after Shape Healing operations.
- The bug in `ShapeFix_IntersectingTool` class has been fixed. Earlier the contour was considered self-intersecting if areas of parts of the contour were approximately equal to each other.
- Boolean operations have been improved to treat more robustly some particular situations (boundary conditions for the faces based on periodic surfaces, coincidence of a vertex and an intersection curve, intersection between edges, coincidence between a vertex and a point, intersection between a line and a circle).
- Methods `Get()` of `Bnd_Box` and `Bnd_Box2d` now raise an exception if the box is void.

  Method `RepTools::AddUVBounds()` adds natural bounds of the surface if face has no edges or no pcurves
- The bug with boolean operations (section, fuse, common, cut) reproduced on particular shapes has been fixed.
- The curvature (LProp_CLProps.gxx file) was calculated incorrectly in Open CASCADE, compiled in optimized mode (with compilator option -DNo_Exception). It was reproduced in 50% cases (depending on the current memory state), due to the usage of uninitialized field. Now this problem is fixed.

Visualization

- New visualization approach based on OpenGL arrays for AIS shapes has been implemented both for Wireframe and Shading display modes. This approach gives an advantage in performance for visualization of AIS shapes.

  Note that this improvement necessitated addition of new arguments to methods Add() of classes creating presentations of shapes (`Prs3d_WFRestrictedFace Prs3d_WFDeflectionRestrictedFace, StdPrs_Curve, StdPrs_DeflectionCurve, StdPrs_WFDeflectionRestrictedFace`).

- WNT package compatibility with ANSI C++ has been improved to avoid problems with compilation under Visual Studio 8.0.

- Bug causing exception in `MeshVS` component (when displaying mesh with nodes having big values of ID) has been fixed

- The bug in `BRepMesh_FastDiscret` class has been fixed. Earlier faces in shading mode disappeared when an end vertex of some edge erroneously had a point on the curve representation.

- The bug in wireframe presentation of a shape has been fixed. Earlier isolines were built incorrectly on large shapes (size ~10^6 and larger): sometimes they could be drawn at a distance from the source faces.

- Previously, it was impossible to call the method `Compute()` of `AIS_PlaneTrihedron` because it was private. It was also impossible to overwrite this method because it was static (CDL declaration). Now it is possible to call it as well as redefine it.

- Optimization of packages related to interactive selection in 3D view (`SelectBasics, Select3D, SelectMgr, StdSelect`) has been accomplished. The two main aspects of this optimization are:
  1. More efficient memory usage - double values (8 bytes) have been replaced with float (4 bytes) where applicable, unused data fields removed, misplaced data fields moved from one abstraction level to another one (example: `TopoDS_Shape` object has been moved from a generic owner class - `SelectMgr_EntityOwner` - to concrete entity owner implementation - `StdSelect_BRepOwner`). These changes resulted in ~30% less memory occupied by data structures used for interactive selection on meshes (`MeshVS_Mesh` presentation).
  2. General code revision - unused methods have been removed, class hierarchy has been improved (example: Select3D package)

- OCCT selection mechanism (method `SelectMgr_Selection::Add()`) is now protected against setting a null selection entity: in optimized mode, it silently ignores that; in Debug mode an exception is raised.

  Method `MeshVS_Mesh::ComputeSelection()` is protected against using the null selection entity that can be returned by the virtual function `MeshVS_PrsBuilder::CustomSensitiveEntity()`; such values are ignored and not added to selection.

- The following problems have been corrected in the framework of industrialization of texture mapping (in particular, environment mapping) in OCCT 3D viewer:
  - disappearance of environment texture after opening a new document;
  - exception raised at application closing;
  - incorrect display of texture-mapped fonts;

  - ineffective memory usage (static variables defined in `AIS_InteractiveContext_1.cxx`);
  - reflection of the environment texture from polylines with environment mapping turned on.

- The algorithm of the method `Prs3d_WFShape::Add` has been modified to avoid putting unnecessary line contexts in the group by calling `SetPrimitivesAspect()` when no primitives are actually added to the group. These redundant line attributes might affect the performance of rendering.
- The type of `myPrimitives` field of Graphic2d object has been changed form `Graphic2d_SequenceOfPrimitives` to `TColStd_IndexedMapOfTransient` in order to improve the performance of `SetIndex` method. The performance of Pick operation has been improved.
- `Graphic3d_AspectText3d::Values()` method has been improved to set <ATextureMappedFont> output argument correctly.
- Nodal color presentation builder now draws edges for volume elements as well.
- Volume elements can now be displayed with transparency.

Data Exchange

- The management of warning and fail messages raised during parsing of STEP files has been improved. Earlier non critical fails or warnings were registered once per application life time, after this fix, such messages are registered at each file reading.
- Protection and correct handling of import of toruses with negative radii from STEP have been implemented. Such toruses can be transferred from some systems; however, negative radiuses are prohibited by STEP standard.
- The problem with cutting strings with the length exceeding 72 symbols has been corrected in STEP file writing.
- Fixed bug of disappearing vertex during collecting shells for the case when a compound consists of faces having shared edges and isolated vertex.

Draw Test Harness

- DRAW Tcl interpreter has been fixed to correctly process strings containing symbols in extended part of ASCII symbol table.
- The sweeping algorithm has been fixed to avoid creation of invalid sweep for circular paths.

WOK

- Automatic generation of methods `new()` and `delete()` in the classes that inherit from others is now avoided.

Documentation

- Foundation Classes User’s Guide has been updated.
- Shape Healing User’s Guide has been updated
- STEP Import/Export User’s Guide has been updated

Products

OMF

- A lot of classes and methods have been introduced and modified in OMF.

  Package SMDS:

  - Class `SMDS_Direction`, implements an equivalent of `gp_Dir,` but twice smaller in size. The components of direction are stored as integers.
  - Class `SMDS_MeshElement` now stores normals as `SMDS_Direction` values; `gp_Dir` type is used only in the interface (for compatibility). This allows defining normals for some element nodes only, not all at once. Two new methods have appeared:
    - `ClearNormal (Standard_Integer)`
    - `GetNDirection (Standard_Integer) -` takes `SMDS_Direction` without conversion of integers to doubles.
  - `SMDS_MeshNode::SetPnt` receives an additional Boolean parameter informing if the normals relevant to that node should be nullified.
  - `SMDS_Mesh::RemoveSubMesh()` nullifies the Parent field of the removed submesh.
  - `SMDS_Mesh` uses `TColStd_PackedMapOfInteger` for the map of nodes (faster than the previous MapOfInteger class).
  - `SMDS_Mesh::Clear(),` is now called from the destructor. It has an optional parameter which removes all nodes that become free after the removal of mesh elements.
  - `SMDS_Mesh::GetParent()` returns the parent mesh or a null handle if none.
  - `SMDS_Mesh::HasInverseConnections` finds out if inverse connections exist, without rebuilding them.
  - `SMDSMesh::GetElementFromTree()` replaces `FindElemnt` finding a node or an element in the mesh+children+parents; it is faster than `FindElement().`

  Package `SMDSControl`:

  - `SMDSControl_BoundaryEdges` can create a free boundary in a separate mesh, no more requirements to use the same mesh or a submesh for the result.

  Package `SMDSAlgo`:

  - `SMDSAlgo_BoxBndTreeSelector and SMDSAlgo_IntersectOnTreen unify the architecture of intersection algorithms (Mesh + some entity), using AABB-tree (NCollection_UBTree) and the pre-selection of intersection zone (as a bounding box defined by the caller).`
  - `SMDSAlgo_LineIntersect - New Mesh-Line/Ray intersection class`
  - Distances in `SMDSAlgo::ProjectPointOnMesh()` are now calculated correctly.

  Package `SMDSEdit`:

  - Creation of non-convex quadrangles is now avoided.

  Package `SMDSMeshVS`:

  - `SMDSMeshVS_VisibleFilter`, implements `SelectMgr_Filter` interface for selection of visible mesh entities.
  - `SMDSMeshVS_DataSource` supports Smooth and Flat shading modes. The smooth shading is maintained by the array of internal normal directions (class

SMDSMeshVS_DirectionsBuffer). The predefined normals on mesh faces are passed to the smooth shading drawer; otherwise (if indefinite) these normals are locally calculated and stored in the DataSource.

- SMDSMeshVS_Mesh adapts to the mesh hierarchical structure: method GetActiveMeshId permits to determine if the current mesh (0) or one of its submeshes (>0) or the current mesh with all its submeshes (-1) are displayed by this interactive object.

- It has become possible to create custom element types without subclass SMDS_Mesh. For example, now we can create a custom Edge element:

```
   class MyEdge : public SMDS_MeshEdge { ... /* must redefine Copy() */
... }
   Handle(MyEdge) newEdge = new MyEdge (idNode1, idNode2);
   myMesh->AddElement (newEdge);
```

  Earlier such added custom element would have been incomplete, having a void internal back reference to myMesh.

- OMF NASTRAN parser has been improved to recognize various formats of continuation entries.
- Class SMDSMeshVS_DataSource has been corrected so as to give the nodes of volumic elements in a correct order. Hereby the problem of twisted mesh volumes presentation has been fixed.
- The Mesh ID Factory of OMF allocated a large amount of memory if an element or a node has been added to the mesh with an explicit big ID number. Now this problem has been fixed.

Parasolid

- Handling of Parasolid XT blended edges has been improved. The resulting algorithm performs computation of blended edges close to Parasolid original.
- Reading of colors from Parasolid XT format has been fixed (it failed to work when *color* was not the first attribute attached to an entity).
- Handling of Parasolid XT schema 17 has been completed. Reading of ESC sequences and entity 99 have been implemented.

Express Mesh

- The effect of AutoSize option has been extended to faces (see QMShape_Tessellator class). Now the algorithm always reproduces small features of the shape in the output mesh to the extent defined by the option AutoSize. Earlier, the form of a cylinder was lost degenerating to plane if the radius of the cylinder was less than the *MinSize* parameter.
- Generation of numerous excess triangles along the borders between surfaces of high and low curvature is now avoided.

Changes

Foundation Classes

- Implementation of exceptions and signals handling on Linux and UNIX platforms has been changed. The existing code using try {} statements should be updated (see below).

  Now exceptions are raised and handled as normal C++ exceptions, not emulated by C longjumps as before. This provides the following advantages over previous versions:

  - Usage of OCC library in the program that uses C++ exceptions does not require special workarounds to provide consistent exception handling
  - Generic `catch(...)` with period can be used without conflicting with OCC code (as it was with some implementations of STL)
  - Exception raised after `catch() {}` in the same code block will not be erroneously caught by this catch. Two consecutive `try{} catch{}` blocks are possible within the same code block.
  - C++ stack unwinds with appropriate destruction of objects allocated in the stack.

  However, handling of signals is still performed with longjump functions (on UNIX and Linux systems). The macro `OCC_CATCH_SIGNALS` should be inserted in the code to be able to catch a signal as an exception. This macro instantiates a signal handling object; in case of a software signal the execution will return to this point and an appropriate exception will be raised from this macro. It is recommended to insert `OCC_CATCH_SIGNALS` macro as the first statement in every try {} block containing code which may generate signal (for instance, generic try {} blocks catching `Standard_Failure`).

  See *Foundation Classes User's Guide* for more details.

  Note that this change is controlled by the new compiler option `OCC_CONVERT_SIGNALS`. The old option `NO_CXX_EXCEPTIONS` still remains for compatibility; it can be used to provide the same behavior as in previous versions of OCCT.

- Static global map of types supported previously by `Standard_Type` class is eliminated. In applications that load and unload some OCCT libraries dynamically, this allows to prevent crashes caused by type objects created in these libraries and remaining in the static map even after unloading of the relevant library.

  As a consequence of this change, static methods of that class providing access to that map (`Find()`, `Exists()`, `Add()`, `NumberOfKnownTypes()`, `GiveKnownTypeNumber()`, `Display_Types()`) have been removed.

  To allow checking of object type by its name rather than by `Handle(Standard_Type)` object, new methods `Standard_Type::SubType()` and `Standard_Transient::IsKind()` have been provided. These methods replicate the existing methods accepting `Handle(Standard_Type)` but accept the type name as `Standard_CString`.

- Method `Key()` of `NCollection_DataMap` iterator have become a `const` now.

- OCCT memory manager (both optimized and raw) now performs check of success of memory allocation operations and raises exception `Standard_OutOfMemory` in case if allocation failed. Previously such failure to allocate memory resulted either in Access Violation situation, or in returning a zero pointer (which in most cases also causes Access Violation).

- A new parameter has been added to the constructor of the `TCollection_AsciiString` class from `TCollection_ExtendedString`: the character to be used instead of any non-ascii symbol encountered in the original string. If this argument is null (default), an exception will be raised, as previously.

- The behavior of Handle() classes has been made more strict: now the Handle clears its pointer when being destroyed, thus an attempt to access the Handle() object after its destruction, which could work previously, will now most likely lead to Access violation.

  The example of the erroneous code that will now lead to exception is:

  ```
  const Handle(...Item) & anItem = aList.First();
  aList.Clear();
  aList.Append ( anItem );
  ```

Visualization

- Interface and implementation of MeshVS classes has been revised in order to optimize the processing of mesh data:
  1. Methods `Get/SetHiddenIds()`, `Get/SetNonSelectableIds()` of class `MeshVS_Mesh`, previously accepting boolean flag "nodes or elements", are replaced by separate methods for nodes and elements, accepting and returning map of IDs by handle (as `Handle(TColStd_HPackedMapOfInteger)`) rather than by value:
     - `GetHiddenIds() -> GetHiddenNodes(), GetHiddenElems()`
     - `SetHiddenIds() -> SetHiddenNodes(), SetHiddenElems()`
     - `GetNonSelectableIds() -> GetSelectableNodes()`
     - `SetNonSelectableIds() -> SetSelectableNodes()`
  2. Methods `IsHidden(), IsSelectable()` of `MeshVS_Mesh` class have been replaced by separate methods for nodes and elements:
     - `IsHidden() -> IsHiddenNode(), IsHiddenElem()`
     - `IsSelectable() -> IsSelectableNode(), IsSelectableElem()`
  3. By default, i.e. if no hidden nodes or elements are given by `SetHidden*()`, the `MeshVS_Mesh` class will show all elements but no free nodes; all nodes that are either not hidden or belong to any non-hidden element will be selectable
  4. Method `GetElementsByNode()` has been removed from `MeshVS_DataSource` class since it is not required any more. This allows to avoid additional analysis of the mesh for backward connections in implementations of `DataSource` classes.
  5. Interface of method `GetNodesByElement()` of the class `MeshVS_DataSource` has been revised: instead of filling the sequence passed as an argument, now it must return HArray1 object (or null if something is wrong). This is a more optimal way of passing a set of integers (less memory allocations are involved and data are more compact).

Data Exchange

- STEP translator now reads additional shapes attached to the main shape of the PRODUCT with `SHAPE_REPRESENTATION_RELATIONSHIP` entities regardless of the contents of the main shape (previously they were read only when the main shape was empty).

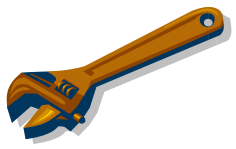
  Note that though it is correct according to AP203 and AP209 (mixed model representation), this could lead to reading unwanted entities (such as auxiliary lines written in this way by some CAD systems). Use new parameter `read.step.shape.relationship` to manage this situation.

Products

OMF

- The classes `MeshTools_NASParser` and `MeshTool_NASBaseReader` have been renamed to `OMFTools_NASParser` and `OMFTools_NASBaseReader correspondingly.`
- Nastran mesh reader adds hexahedrons into SMDS structure with inversion of nodes to have a consistent normal of element faces as a result.
- Re-orientation of nodes of volume elements (tetrahedron, prism, pyramid, hexahedron) has been corrected in order to save the initial element face orientation.

Bug Fixes



- Since last minor release (version 6.1) Open CASCADE 6.2 incorporates **142** modifications (bug fixes, enhancements and other corrections). For details, refer to **Appendix**.

Appendix: Open CASCADE 6.2 Bug Fixes

- Foundation Classes
- Modeling Algorithms
- Visualization
- Application Framework
- Data Exchange
- Draw Test Harness
- WOK
- Documentation
- Samples
- Development Environment

Products

- DXF
- OMF
- Parasolid
- Surfaces from Scattered Points
- Express Mesh

	Foundation Classes, 22 bug fixes
ID	**Short Description**
9755	Incorrect behavior of try - catch macro
10010	Test behavior of Units package and add support of percent (%) as unit
10096	Reduce overhead imposed by OCC memory manager
11664	New optimized bounding box types
12131	Improvement of Exception mechanism on Unix and Linux platforms
12186	Basic multithread safety in OCCT Kernel
13051	Using nonexistent objects in Standard_ForMapOfTypes class
13131	Optimize code of OCC classes generated by WOK
13151	Provide complete version number of OCCT as hex
13189	Optimize OCC maps for work with big amounts of data
13438	Small modifications in the Units package
13596	Using C++ Namespace with Handle types
13604	Setting OSD_Thread priority
13846	Memory leak in TColStd_PackedMapOfInteger
13963	Some improvements in Bnd and gp classes
13964	Boolean opoerations on TColStd_PackedMapOfInteger
14166	NCollection_DataMap::Iteartor should have const method ::Key()
14325	Incorrect implementation of methods Mirror() in gp_Vec
14591	OCCT memory manager should raise appropriate exception if memory allocation fails
14670	Method to access object contained in the map
14672	Provide exception-safe conversion from ExtendedString to AsciiString
14794	Patch from GMSH for integration into OCCT 6.2

	Modeling Algorithms, 51 bug fixes
ID	**Short Description**
10086	Query SAMT-05-002
10392	Extrema_ExtPC class does not initialize its fields in constructor.
11081	BRepExtrema_DistShapeShape misses one of two solutions.
11789	There is a severe memory leak in TopExp_Explorer methods Init and Clear.
12203	Non manifold sewing generates invalid shape.
12257	Fuse operation fails.
12460	During command "thrusection" in DRAW exception is raised on attached shapes.
12487	Boolean operations fail on some compounds.
12522	Extrema problems. Case: one of the Extrema arguments is an infinite face or an infinite edge.
12547	Linking error of pseudo-inline function
12572	Extrema problem on edges based on curves with continuity C0.
12627	Classification of a point comparing to a face is incorrect.
12635	Bugs in projector algorithm
12661	Wrong calculation of bnd box for edge if edge has polygon of triangulation.
12805	It is necessary to add the new method IntTools_FClass2d::IsHole()
12831	Extrema_ExtPC has uninitialized fields.
12832	math_NewtonFunctionRoot has uninitialized fields (X for example)
12833	Uses uninitialized fields in ProjLib_Fuction class
12836	Method to get rotation angle from gp_Trsf2d
12840	Uninitialized fields in IntImp_ZerCSParFunc
12849	Blend_Extremity has uninitialized fields.
12850	class TopOpeBRepDS_Curve has uninitialized fields
12851	class Geom2dHatch_Intersector has uninitialized fields
12884	Wrong result of projection curve on surface
12888	Wrong result of projection curve on surface
12918	Boolean Operations failed
12919	Add new methods BOPTools_SSInterference::SetSharedEdges()/SharedEdges()
13081	It is necessary to have some modifications in IntTools_FClass2d.cxx
13115	Modification of the Shape Healing operations to keep non-manifold topology.
13116	Boolean Operations produce faulty shape.
13140	Exception during creation pipe on from attached spine and profile.
13149	Following of exception improvement
13186	Problem with Boolean operation
13209	Exception is raised while performing Boolean operations
13209	Exception is raised while performing boolean operations
13211	Wrong treatment of conical faces in BOP algorithm
13267	Summary: FlxShape works incorrectly on small contours
13354	Exception in Same Parameter
13395	Pipe is constructed wrongly on a result of revolution
13595	MakePipe raises exception
13657	Incorrect Curvature in optimized mode
13962	Exception when drawing face without pcurves
14076	Wrong hiding of toroidal and cylindrical shapes
14162	Bug initiated by pb in Partition Algorithm of SALOME Project (pb: Sph_03)

14227	Bug initiated by pb in Partition Algorithm of SALOME Project (pb: Use_01)
14409	Bug initiated by pb in Partition Algorithm of SALOME Project (pb: Use_02)
14506	Wrong section of faces with bspline geometry
14536	Boolean Operation Algorithm failed
14777	Boolean Operation Algorithm failed
14780	Boolean operations (section, fuse, common, cut) give incorrect result
14782	Method segment works wrongly for periodical BSpline curve
	Visualization, 28 bug fixes
ID	**Short Description**
9833	Modify MeshVS to support smooth and flat shading
9915	Provide means in MeshVS to interpolate color along element according to color scale.
11770	Selected sub-objects can be highlighted by mouse in 3D Viewer only one time.
11804	Regression concerning to incorrect definition of a bounding box (minimal and maximal values) for markers in 3D view.
11904	Problems with texture mapping.
12121	Optimization of existing selection classes
12146	Package WNT: Compilation problems under Visual Studio 8.0.
12297	Migration to new visualization approach based on OpenGL arrays
12327	Prs3d_WFShape::Add creates unnecessary line contexts in the group
12476	Improving highlighting performance of large scale model
12531	V2d_View::ScreenPostScriptOutput method produce incorrect .ps file.
12844	Regression in Trihedron
12977	Creating a second 3D view after closing the first one crashes an application on ATI Radeon cards
13100	Crash in MeshVS with large meshes
13142	Face is not displayed in Shading mode after gluing Operation
13144	Incorrect isolines building after Cut operation
13439	The text in ColorScale is not displayed on Linux.
13439	The text in ColorScale is not displayed on Linux.
13682	AIS_PlaneTrihedron forbids re-defintion of the method ::Compute()
13801	MeshVS: Incorrect representation of mesh volumes in shading
13852	MeshVS: Sometimes depth filtering is off after interactive highlighting of mesh entities
13960	MeshVS: Possibility to turn on material reflectance on colored data presentation
14258	Volume elements are not displayed in shading mode with transparency
14259	Nodal color presentation for volume elements doesn't draw edges
14384	Exception during iteration on sensitive entities of selection
14385	Avoid copying the map objects in mesh visualisation - operate with maps by handles
14480	Mesh presentation for volume elements doesn't draw edges in shading.
14735	Regression in wireframe visualization
	Application Framework, 1 bug fix
ID	**Short Description**
13605	Transaction signal API in TDocStd_Document
	Data Exchange, 8 bug fixes

ID	Short Description
11856	Error in reading of step file.
11857	Step toruses with negative radiuses did not translated
12995	Avoid using cout in STEP and IGES translators
13105	Revising of parameters of STEP translator
13200	Impossible of correct import/export sequence of operations.
13378	Disappearing vertex after sewing (following bug SAM13352)
13627	Crash during reading a.igs file on Linux (RedHat) platform
13697	New Shape Healing operators implemented by request EADS-CCR
	Draw Test Harness, 2 bug fixes
ID	**Short Description**
63	It's impossible to open files containing localization characters in the name.
12213	Invalid sweep was created by command "buildsweep" in DRAW in case circular path.
	WOK, 2 bug fixes
ID	**Short Description**
10033	WOK: Avoid defining new() and delete() for classes derived from other.
12619	The extraction on Windows fails.
	Documentation, 1 bug fix
ID	**Short Description**
12994	Undocumented parameters in STEP and IGES translators.
	Samples, 1 bug fix
ID	**Short Description**
14869	Problem of reading screw.step in OCC 6.1.1 Import/Export sample.
	Development Environment, 3 bug fixes
ID	**Short Description**
14237	Problem of CASCADE compilation : pthread symbols are not resolved
14788	Preprocessor macro is needed to distinguish 64- vs 32-bit compilation mode at compile time
14898	Implementation of optional compilation of OCC modules by make files

Product Bug Fixes

The following bug fixes have been performed for Open CASCADE specific development customers.

	DXF, 1 bug fix
ID	**Short Description**

12980	Option to avoid reading data from anonymous blocks (HATCH, DIMENSION, and other generated data).
	OMF, 15 bug fixes
ID	**Short Description**
7638	Nastran files are imported incorrectly (see attached picture).
9115	Introduce smooth shading mode.
9665	Add possibility to select only visible mesh entities.
9832	Modify OMF sample so as to use MeshVS_Mesh presentation instead of SMDSInter and modify SMDS and SMDSMeshVS packages.
12251	Various improvements in OMF.
12481	Addition of elements with same ID destroys the integrity of SMDS_Mesh.
13102	SmoothShading mode crashes when the element IDs form a sparse array.
13214	Back pointer to mesh is not set in SMDS_Mesh::AddElementWithID.
13252	Rename package MeshTools to OMFTools.
13536	Nastran reader not always recognizes continued record in NASTRAN file
13729	Problem with reading NASTRAN files
13802	Twisted mesh volume presentation
13825	Too much memory is allocated to store free IDs in MeshIDFactory
14257	Orientation of 3D elements is changed
14263	Wrong orientation of hexes after import from Nastran
	Parasolid, 3 bug fixes
ID	**Short Description**
12389	Problems with translation of BlendedEdge
13042	Colors are not read from XT file
13353	Fails in reading of Parasolid XT files
	Surfaces from Scattered Points, 3 bug fixes
ID	**Short Description**
13129	Modification to keep non-manifold topology.
13440	Iimplementation of Surface Modification and Surface Curvature Analysis algorithms.
13560	TestFEAPI is not loaded in DRAW by means of pload command.
	Express Mesh, 1 bug fix
ID	**Short Description**
13193	Implement AutoSize feature for discretizing interior of face by analogy with edge.