CAD Xpansion SDK

Processing of 3D Models to the CAD Data Exchange Formats

Guide & Reference

Version: 4.0.3

Date: 26.08.2017

Contents

1. INTRODUCTION. 3D DATA EXCHANGE FORMATS IN CAD

There are a lot of universal formats for 3D data exchange in CAD software. The most used are the following:

- STEP - international standard for product data exchange (ISO_10303) - "Standard for the Exchange of Product model data".
- IGES - "Initial Graphics Exchange Specification“, is a file format which defines a vendor neutral data format that allows the digital exchange of information among Computer-aided design (CAD) systems.
- STL – “STereoLithography”, is a format which facet-based representation that approximates surface and solid entities only. It is a list of the triangular surfaces that describe a computer generated solid model. This is the standard input for most rapid prototyping machines.
- Parasolid - geometric modeling kernel originally developed by ShapeData, now owned by Siemens.
- JT – 3D data format developed by Siemens and is used for product visualization, collaboration and data exchange.
- VRML – Virtual Reality Modelling Language, standard file format for representing 3D interactive vector graphics.
- DXF 3D – Drawing eXchange Format developed by AutoCAD.
- ACIS – data format for Geometric modeling kernel developed by Spatial Corporation.
- VDA-FS - (Verband Der Automobilindustrie - FlächenSchnittstelle) - CAD data exchange format for the transfer of surface models between CAD systems. Standard was specified by the German organization VDA.
- and more.

CAD Xpansion SDK uses **STEP, IGES** and **STL** formats in the actual version.

Important property of CAD Xpansion SDK is that the library contains the large set of structured objects from the practice (geometry, mechanical design, building construction, etc.) as well as general geometry possibilities to build arbitrary objects. The 3D models are prepared and parameterized for using in typical practical assemblies and construction tasks. See Appendix 1 for reference.

CAD Xpansion SDK has been designed for programmers. To use the library the appropriate programming knowledge, incl. 3D graphics programming required.

CAD Xpansion SDK based on the Open Cascade Technology.

2. LIBRARY GUIDE

2.1 Review of Possibilities

CAD Xpansion SDK provides the possibilities to describe 3D models for objects from different application areas, there combinations (assemblies), including arbitrary 3D descriptions and export these models in universal CAD data formats: STEP, IGES, STL.

Using the library allows to provide you:

1) Export to the CAD formats of different objects from the following application areas:

- Mechanical design
- Building construction
- etc.

The object lists can be found in the Appendix 1.

2) Import of any model in the STEP, IGES, STL formats, and recognition of shaft parameters and its elements being imported from the STEP **format.**

2.2 Developer's Side

2.2.1 System Requirements and Platforms

The library has been implemented on the following development platform: C++. Please refer to manufacturer for using C or Delphi platform.

Look here for the actual system requirements: http://soft-xpansion.eu/support/sys-lic/#cad

Windows x86 and x64 platforms are supported.

2.2.2 Delivery

Delivery archive file **sx-CAD-library.zip** consists of three folders:

1) **DOC** – documentation and licensing files
    - **CAD Xpansion Guide & Reference.pdf** – this document
    - **GNU Lesser General Public License.txt** – license for the base library Open Cascade
2) **DevRes** – resources you need for development
    - Include: contains the header files for C++ development desktop or server applications using CAD Xpansion SDK. You should include these files to your project before you start to use the library.
    - Samples: this folder contains sample projects. Sample projects can be compiled using Microsoft Visual Studio 2013.
3) **Redist** – redistributable files to be included in the end user application and need to be redistributed along with the client application that uses CAD Xpansion SDK
    - **cad-xpansion-32.dll** for 32 bit applications
    - **cad-xpansion-64.dll** for 64 bit applications
    - **GNU Lesser General Public License.txt**

2.2.3 Authorization

Each copy of CAD Xpansion SDK has its own authorization. It can be demo / trial license or individual license. If you have a trial license first, after that purchase the product you should exchange your license file with the purchased one.

Note! Along with license file you should replace the trial license key in you sources because every license have its own license key. You can't authorize your license with trial key vice-versa.

2.3 End User's Side

2.3.1 System Requirements

The system requirements include the requirements to the application developer (programmer) using CAD Xpansion SDK (platform, technology …) and the requirements to the end user of this application (system requirements, what files and where must be located).

Operating systems: Windows XP (SP3), Vista, 7, 8, 10, Server 2003, 2008, 2012, all SPs.

2.3.2 Redistribution

Files that need to be redistributed along with the client application that uses CAD Xpansion SDK are placed in the "**Redist**" folder. At least one dll (32 bit or 64 bit) and all other files have to be delivered. You should copy redistributable files to one directory – application folder or subfolder. After you purchase SDK license you get your corporate license file "**cad-xpansion.license**" and you will need to overwrite trial license file in order to replace trial license by your own. If you have tried the SDK in several projects, you must overwrite the license file in all these projects.

2.4 Getting Started

In order to use the 3D models export/recognition functionality following should be done in your application:

1) Define your license. Trial or real license should be available and used properly. If you already purchased a real license, please replace the trial license with your own, otherwise you can test CAD Xpansion SDK with the trial license. The trial license hasn't any functional restrictions, but applies DEMO watermark to the objects in an exported file.
2) Compile one or more of delivered samples or try to integrate CAD Xpansion SDK in your project. For this integration you need to add the Library references in the project.
3) Reference library in your projects. Folder "DevRes\include" contains the header files for C++ development desktop or server applications using CAD Xpansion SDK. You should include these files to your project before you start to use the Library API.
4) Load and initialize the library. Before you can use the library in your application, you must load it from "cad-xpansion-32.dll" or "cad-xpansion-64.dll" module using LoadLibrary or LoadLibraryEx function. After successful loading you can use module handle to get entry point of the library anytime. See one of the samples as an example.
5) By constructing objects with a lot of the vertices (e.g., for gears with free-prifile), in 32-bit applications one can meet with a lack of memory. In order to increase it you must set the option "Enable Large Addresses" in the "Project Properties" to YES.

3. PROGRAMMING REFERENCE

3.1 Interfaces

CAD Library provides a set of interfaces to export 3D graphics in STEP, IGES and STL format.

3.1.1 IModel

Interface provides methods and properties to access the model for export.

Root

The property provides a tree root of 3D objects.

Type

ICollection

Access

R

Transformation

The property provides tools for transformation 3D objects.

Type

ITransformation

Access

R

ModelGears

The property provides tools for create gears.

Type

IModelGears

Access

R

ModelBearings

The property provides tools for create bearings.

Type

IModelBearings

Access

R

ModelBeams

The property provides tools for create beams.

Type

IModelBeams

Access

R

ModelShafts

The property provides tools for create shafts.

Type

IModelShafts

Access

R

ModelBoltedJoins

The property provides tools for create bolted joins.

Type

IModelBoltedJoins

Access

R

RecognitionShafts

The property provides tools for shaft recognition.

Type

IRecognitionShafts

Access

R

Header

The property provides header.

Type

IHeader

Access

R

ExportSTEP

Method exports the model in STEP file format.

Parameters

sx_astr Path to the exported file

Returns

sx_bool

ExportIGES

Method exports the model in IGES file format.

Parameters

sx_astr Path to the exported file

Returns

sx_bool

ExportSTL

Method exports the model in STL file format.

Parameters

sx_astr Path to the exported file

Returns

sx_bool

ImportSTEP

Method imports the model from STEP file format.

Parameters

sx_astr Path to the imported file

Returns

sx_bool

ImportIGES

Method imports the model from IGES file format.

Parameters

sx_astr	Path to the imported file

Returns

sx_bool

ImportSTL

Method imports the model from STL file format.

Parameters

sx_astr	Path to the imported file

Returns

sx_bool

CreateBox

Method creates a box.

Parameters

sx_double	Width
sx_double	Height
sx_double	Length

Returns

IShape

Errors

IException

CreateSphere

Method creates a sphere.

Parameters

sx_double	Radius

Returns

IShape

Errors

IException

CreateCylinder

Method creates a cylinder.

Parameters

sx_double	Radius
sx_double	Length

Returns

IShape

Errors

IException

CreateCone

Method creates a cone.

Parameters

sx_double	First radius
sx_double	Second radius
sx_double	Length

Returns

IShape

Errors

IException

CreateTorus

Method creates a torus.

Parameters

sx_double	First radius
sx_double	Second radius

Returns

IShape

Errors

IException

CreateWedge

Method creates a wedge.

Parameters

sx_double	Bottom Width
sx_double	Height
sx_double	Length
sx_double	Top width

Returns

IShape

Errors

IException

CreateHollowCylinder

Method creates a hollow cylinder.

Parameters

sx_double	Outer diameter
sx_double	Inner diameter
sx_double	Length

Returns

IShape

Errors

IException

CreateHollowCone

Method creates a hollow cone.

Parameters

sx_double	First outer diameter
sx_double	Second outer diameter
sx_double	First inner diameter
sx_double	Second inner diameter
sx_double	Length

Returns

IShape

Errors

IException

3.1.2 IModelBearings

Interface provides methods for create bearings.

CreateRollerBearing

Method creates a roller bearing (simplified presentation).

Parameters

sx_double	Outer diameter
sx_double	Inner diameter
sx_double	Width

Returns

IShape

Errors

IException

CreateSlidingBearing

Method creates a sliding bearing.

Parameters

sx_double	Outer diameter
sx_double	Inner diameter
sx_double	Width

Returns

IShape

CreateBearing

Method creates a bearing. See roller bearing parameters in the objects table, Appendix 1.

Parameters

sx_double	Outer diameter
sx_double	Inner diameter
sx_double	Width
BearingType	Type of bearing

Returns

IShape

3.1.3 IModelGears

Interface provides methods for create gears.

CreateExternalGear

Method creates an external gear. Middle of the first tooth located on the Y-axis.

Parameters

sx_gear	Gear structure

or

Method creates an external gear (free profile). Angle = 0 in tooth profile means that the point will be located on the Y-axis.

Parameters

IToothProfile	Tooth profile
sx_gear_profile	Gear with free profile structure

Returns

IShape

Errors

IException

CreateInternalGear

Method creates an internal gear. Middle of the first tooth located on the Y-axis.

Parameters

sx_gear	Gear structure

or

Method creates an internal gear (free profile). Angle = 0 in tooth profile means that the point will be located on the Y-axis.

Parameters

IToothProfile	Tooth profile
sx_gear_profile	Gear with free profile structure

Returns

IShape

Errors

IException

CreateDoubleExternalGear

Method creates a double external gear. Middle of the first tooth located on the Y-axis.

Parameters

sx_gear_double	Double gear structure

or

Method creates a double external gear (free profile). Angle = 0 in tooth profile means that the point will be located on the Y-axis.

Parameters

IToothProfile	Tooth profile
sx_gear_double_profile	Double gear with free profile structure

Returns

IShape

Errors

IException

CreateDoubleInternalGear

Method creates a double internal gear. Middle of the first tooth located on the Y-axis.

Parameters

sx_gear_double	Double gear structure

or

Method creates a double internal gear (free profile). Angle = 0 in tooth profile means that the point will be located on the Y-axis.

Parameters

IToothProfile	Tooth profile
sx_gear_double_profile	Double gear with free profile structure

Returns

IShape

Errors

IException

CreateRackGear

Method creates a rack.

Parameters

sx_double	Tooth module
sx_double	Width
sx_double	Height
sx_double	Length

Returns

IShape

Errors

IException

3.1.4 IModelShafts

Interface provides methods for create shafts.

CreateShaft

Method creates a shaft.

Returns

IShaft

Errors

IException

CreatePlanetCarrier

Method creates a planet carrier (simplified presentation)

Parameters

sx_uint	Number of branches
sx_double	Radius of branches
sx_double	Length

Returns

IShape

Errors

IException

3.1.5 IModelBeams

Interface provides methods for create beams.

CreateBeamRect

Method creates a beam with rectangular profile.

Parameters

sx_double	Height
sx_double	Width
sx_double	Length
sx_double	Depth

Returns

IShape

Errors

IException

CreateBeamCircle

Method creates a beam with circle profile.

Parameters

sx_double	Diameter
sx_double	Length
sx_double	Depth

Returns

IShape

Errors

IException

CreateBeamEllipse

Method creates a beam with ellipse profile.

Parameters

sx_double	Diameter x
sx_double	Diameter y
sx_double	Length
sx_double	Depth

Returns

IShape

Errors

IException

CreateBeamL

Method creates a beam with L-profile.

Parameters

sx_double	Height
sx_double	Width
sx_double	Depth
sx_double	Radius of rounding
sx_double	Length

Returns

IShape

Errors

IException

CreateBeamZ

Method creates a beam with Z-profile.

Parameters

sx_double	Height
sx_double	Width
sx_double	Depth (t)
sx_double	Depth (s)
sx_double	Radius of rounding 1
sx_double	Radius of rounding 2
sx_double	Length

Returns

IShape

Errors

IException

CreateBeamI

Method creates a beam with I-profile.

Parameters

sx_double	Height
sx_double	Width
sx_double	Depth (t)
sx_double	Depth (s)
sx_double	Radius of rounding 1
sx_double	Radius of rounding 2
sx_double	Length
sx_bool	Is need taper

Returns

IShape

Errors

IException

CreateBeamU

Method creates a beam with U-profile.

Parameters

sx_double	Height
sx_double	Width
sx_double	Depth (t)
sx_double	Depth (s)
sx_double	Radius of rounding 1
sx_double	Radius of rounding 2
sx_double	Length
sx_bool	Is need taper

Returns

IShape

Errors

IException

CreateBeamT

Method creates a beam with T-profile.

Parameters

sx_double	Height
sx_double	Width
sx_double	Depth (t)
sx_double	Depth (s)
sx_double	Radius of rounding 1
sx_double	Radius of rounding 2
sx_double	Radius of rounding 3
sx_double	Length
sx_bool	Is need taper

Returns

IShape

Errors

IException

3.1.6 IModelBoltedJoins

Interface provides methods for create bolted joins (bolts, screws, studs etc.).

CreateBoltHex

Method creates a hex bolt.

Parameters

sx_bolt_hex	Hexagon bolt structure

Returns

IShape

Errors

IException

CreateBoltHexSocket

Method creates a bolt with hex socket head.

Parameters

sx_bolt_hexsocket	Hex socket head bolt structure

Returns

IShape

Errors

IException

CreateBoltHexSocketCountersunk

Method creates a bolt with hex socket head and countersunk.

Parameters

sx_bolt_hexsocket_countersunk	Hex socket head with countersunk bolt structure

Returns

IShape

Errors

IException

CreateBoltHexFlange

Method creates a bolt with hex socket head and flange.

Parameters

sx_bolt_hexflange	Hex socket head with flange bolt structure

Returns

IShape

Errors

IException

CreateBoltCarriage

Method creates a bolt with round head and square neck

Parameters

sx_bolt_carriage	Round head and square neck bolt structure

Returns

IShape

Errors

IException

CreateBoltHexSetDogPoint

Method creates a bolt with hexagon head and full dog point

Parameters

sx_bolt_hexset_dogpoint	Hexset and full dog point bolt structure

Returns

IShape

Errors

IException

CreateBoltHexSetCone

Method creates a bolt with hexagon head and flat cone point.

Parameters

sx_bolt_hexset_cone	Hexset and flat cone point bolt structure

Returns

IShape

Errors

IException

CreateBoltT_Head

Method creates a bolt with T-Head (Hammerhead).

Parameters

sx_bolt_t_head	T-Head bolt structure

Returns

IShape

Errors

IException

CreateBoltT_HeadSquareNeck

Method creates a bolt with T-Head (Hammerhead) and square neck.

Parameters

sx_bolt_t_head	T-Head bolt structure

Returns

IShape

Errors

IException

CreateBoltEye

Method creates a bolt with eye (Swing bolt).

Parameters

sx_bolt_eye	Eyebolt structure

Returns

IShape

Errors

IException

CreateBoltU

Method creates a U-bolt (Steel strap).

Parameters

sx_bolt_u	U-bolt structure

Returns

IShape

Errors

IException

CreateBoltRing

Method creates a ring bolt.

Parameters

sx_bolt_ring	Ring-bolt structure

Returns

IShape

Errors

IException

CreateBoltSpigot

Method creates a bolt with spigot and stud end.

Parameters

sx_bolt_spigot	Bolt with spigot and stud end structure

Returns

IShape

Errors

IException

CreateScrewCheeseSlotted

Method creates a slotted cheese head screw.

Parameters

sx_screw_cheeseslotted	Slotted cheese head screw structure

Returns

IShape

Errors

IException

CreateScrewPanSlotted

Method creates a slotted pan head screw.

Parameters

sx_screw_panslotted	Slotted pan head screw structure

Returns

IShape

Errors

IException

CreateScrewCountersunkSlotted

Method creates a slotted countersunk head screw.

Parameters

sx_screw_countersunk_slotted	Slotted countersunk head screw structure

Returns

IShape

Errors

IException

CreateScrewCountersunkRaisedSlotted

Method creates a slotted raised countersunk head screw.

Parameters

sx_screw_countersunkraised_slotted	Slotted raised countersunk head screw structure

Returns

IShape

Errors

IException

CreateScrewFillisterCrossRecessed

Method creates a cross recessed fillister head screw.

Parameters

sx_screw_fillister_crossrecessed	Cross recessed fillister head screw structure

Returns

IShape

Errors

IException

CreateScrewCountersunkCrossRecessed

Method creates a cross recessed countersunk head screw.

Parameters

sx_screw_countersunk_crossrecessed	Cross recessed countersunk head screw structure

Returns

IShape

Errors

IException

CreateScrewCountersunkRaisedCrossRecessed

Method creates a cross recessed raised countersunk head screw.

Parameters

sx_screw_countersunkraised_crossrecessed	Cross recessed raised countersunk head screw structure

Returns

IShape

Errors

IException

CreateScrewHexSocketButton

Method creates a hexagon socket button head screw.

Parameters

sx_screw_hexsocket_button	Hexagon socket button head screw structure

Returns

IShape

Errors

IException

CreateScrewCountersunkSquareNeck

Method creates a countersunk head square neck bolt.

Parameters

sx_screw_countersunk_squareneck	Countersunk head square neck bolt structure

Returns

IShape

Errors

IException

CreateScrewSquareCollar

Method creates a square head bolt with collar.

Parameters

sx_screw_squarecollar	Square head bolt with collar structure

Returns

IShape

Errors

IException

CreateScrewThumb

Method creates a thumb screw.

Parameters

sx_screw_screwthumb	Thumb screw structure

Returns

IShape

Errors

IException

CreateScrewSetSlottedDogPoint

Method creates a slotted set screw with dog point.

Parameters

sx_screw_set_slotted_dogpoint	Slotted set screw with dog point structure

Returns

IShape

Errors

IException

CreateScrewSetSlottedConePoint

Method creates a slotted set screw with cone point.

Parameters

sx_screw_set_slotted_conepoint	Slotted set screw with cone point structure

Returns

IShape

Errors

IException

CreateScrewHexSocketDogPoint

Method creates a hex socket set screw with dog point.

Parameters

sx_screw_set_hexsocket_dogpoint	Hex socket set screw with dog point structure

Returns

IShape

Errors

IException

CreateScrewHexSocketConePoint

Method creates a hex socket set screw with cone point.

Parameters

sx_screw_set_hexsocket_conepoint	Hex socket set screw with cone point structure

Returns

IShape

Errors

IException

StudDoubleEnd

Method creates a double end stud.

Parameters

sx_stud_doubleend	Double end stud structure

Returns

IShape

Errors

IException

3.1.7 IShaftSection

Interface provides methods for create shaft section with grooves, notches, slots.

AddGrooveRectangular

Method adds a rectangular groove.

Parameters

sx_double	Position
sx_double	Width
sx_double	Depth
sx_double	Radius

Errors

IException

AddNotchChevron

Method adds a chevron notch.

Parameters

sx_double	Position
sx_double	Depth

Errors

IException

AddGrooveCircular

Method adds a circular groove.

Parameters

sx_double	Position
sx_double	Depth
sx_double	Radius

Errors

IException

AddCrossHole

Method adds a cross hole.

Parameters

sx_double	Position
sx_double	Depth
sx_double	Radius
sx_double	Angle of rotation in radians. Angles are measured clockwise when looking along the Z-axis toward the origin. If angle = 0 then cross hole located on the X-axis

Errors

IException

AddSlots

Method adds slots.

Parameters

sx_double	Position
sx_double	Length
sx_uint	Number of slots
sx_double	Diameter
sx_double	Inner width slot
sx_double	Outer width slot

Errors

IException

AddGrooveParallelKey

Method adds a parallel key groove.

Parameters

sx_double	Position
sx_double	Length
sx_double	Depth
sx_double	Radius
sx_double	Angle of rotation in radians. Angles are measured clockwise when looking along the Z-axis toward the origin. If angle = 0 then parallel key groove located on the X-axis

Errors

IException

3.1.8 IShaft

Interface provides methods for create shaft from sections.

Count

The property provides number of sections.

Type

sx_uint

Access

R

AddSection

Method adds a section.

Parameters

sx_double	Position
sx_double	First outer diameter
sx_double	Second outer diameter
sx_double	First inner diameter
sx_double	Second inner diameter
sx_double	Length
sx_double	First chamfer (<0) or rounding (> 0)
sx_double	Second chamfer (<0) or rounding (>0)

Returns

IShaftSection

Errors

IException

Item

Method retrieves the item at a given index.

Parameters

sx_uint	Serial number of section in the shaft

Returns

IShaftSection

Errors

IException

Merge

Method concatenates sections into a single object.

Returns

IShape

Errors

IException

AddToCollection

Method adds the shaft sections to the collection.

Parameters

ICollection	Serial number of section in the shaft

3.1.9 ITransformation

Interface provides methods for geometrical transforms of 3D objects.

Transform

The property provides transformation matrix.

Type

sx_matrix

Access

R

Translate

Method adds an offset to the transformation matrix.

Parameters

sx_matrix	Transformation matrix
sx_double	x-coordinate offset
sx_double	y-coordinate offset
sx_double	z-coordinate offset

Rotate

Method adds rotates around an arbitrary axis to the transformation matrix.

Parameters

sx_matrix	Transformation matrix
sx_point	The arbitrary axis
sx_double	Angle of rotation in radians. Angles are measured clockwise when looking along the rotation axis toward the origin

Scale

Method adds a factor scale to the transformation matrix.

Parameters

sx_matrix	Transformation matrix
sx_double	Factor scale

3.1.10 IHeader

FileDescription

The property provides file description.

Type

sx_astr

Access

R/W

FileName

The property provides file name.

Type

sx_astr

Access

R/W

Author

The property provides author.

Type

sx_astr

Access

R/W

Organization

The property provides organization.

Type

sx_astr

Access

R/W

Authorisation

The property provides authorisation.

Type

sx_astr

Access

R/W

3.1.11 IShape

Interface provides methods for editing geometry objects.

Subtract

Boolean operation for creating new shape from two shapes. The result contains the points in the current shape **except** of the points in the other shape.

Parameters

IShape	Subtractive shape

Returns

IShape

Errors

IException

Union

Boolean operation for creating new shape from two shapes. The result contains the points in the current shape **or** the points in the other shape.

Parameters

IShape	United shape

Returns

IShape

Errors

IException

Intersect

Boolean operation for creating new shape from two shapes. The result contains the points in the current shape **and** in the other shape.

Parameters

IShape	Intersected shape

Returns

IShape

Errors

IException

Add

Method adds another shape to current shape.

Parameters

IShape	Added shape

Errors

IException

Transform

Method transforms shape.

Parameters

sx_matrix	Transformation matrix

Returns

IShape

Errors

IException

3.1.12 IObject

The object with attributes such as: name, location.

Parent

The property provides parent object.

Type

ICollection

Access

R

Name

The property specifies name object.

Type

IStr

Access

R/W

Transform

The property specifies transformation matrix of object.

Type

sx_matrix

Access

R

IsCollection

The property specifies that object is collection.

Type

sx_bool

Access

R

Transform

Method sets transformation matrix.

Parameters

sx_matrix	Transformation matrix

3.1.13 ICollection

The collection of objects. Interface inherits from IObject.

Count

The property provides a size of the collection.

Type

sx_uint

Access

R

Item

Method retrieves the item at a given index.

Parameters

sx_uint	Serial number of object in the collection

Returns

IObject

Errors

IException

Add

Method adds the collection to the end of the collection.

Returns

ICollection

or

Method adds the shape to the end of the collection.

Parameters

IShape	Shape

Returns

IShapeObject

Remove

Method removes the item from the collection by index.

Parameters

sx_uint	Serial number of object in the collection

Returns

IShapeObject

Errors

IException

3.1.14 IShapeObject

The geometry object with attributes such as: name, color, location. Interface inherits from IObject.

Shape

The property provides geometry object.

Type

IShape

Access

R

Color

The property specifies color of the object.

Type

sx_color_rgb

Access

R/W

3.1.15 IToothProfile

The gear tooth profile.

Spline

The property specifies that tooth profile is spline.

Type

sx_bool

Access

R/W

Add

Method adds point.

Parameters

sx_double	Radius
sx_double	Angle

Errors
IException

3.1.16 IRecognitionShafts

Interface provides methods for recognize shafts.

RecognizeShaft

Method recognize a shaft.

Parameters

IShape	Shape

Returns

IRecognizeShaft

Errors
IException

3.1.17 IRecognizeObject

Interface provides methods for get recognized object data.

Transform

The property provides a transformation matrix of recognized object.

Type

sx_matrix

Access

R

3.1.18 IRecognizeShaft

Interface provides methods for get recognized shaft data. Interface inherits from IRecognizeObject.

SectionsCount

The property provides count of the shaft sections.

Type

sx_uint

Access

R

Section

The property provides a recognized shaft section.

Type

IRecognizeShaftSection

Access

R

3.1.19 IRecognizeShaftSection

Interface provides methods for recognized section data.

Get methods:

GroovesCount

The property provides count of the grooves.

Type

sx_uint

Access

R

Groove

The property provides a groove.

Type

IRecognizeGroove

Access

R

Data

The property provides a recognized shaft section data.

Type

sx_shaft_section

Access

R

Set methods:

Data

The property sets a recognized shaft section data.

Type

void

Parameters

sx_shaft_section

Access

R

3.1.20 IRecognizeGroove

Interface provides methods for get recognized grooves data.

GrooveType

The property provides groove type.

Type

sx_groove_type

Access

R

CrossHole

The property provides a recognized cross hole data.

Type

sx_groove_cross_hole

Access

R

GrooveParallelKey

The property provides a recognized groove parallel key data.

Type

sx_groove_parallel_key

Access

R

NotchChevron

The property provides a recognized notch chevron data.

Type

sx_groove_notch_chevron

Access

R

GrooveRectangular

The property provides a recognized groove rectangular data.

Type

sx_groove_rectangular

Access

R

GrooveSlots

The property provides a recognized groove slots data.

Type

sx_groove_slots

Access

R

3.1.21 IView

Interface provides methods for visualization of objects in a 3D – scene.

Add

Method adds new IShape object to 3D-scene.

Parameters

IShape	new shape

Returns

IShapeVis

Redraw

Method redraws the 3D-scene.

Zoom

Method zooms the 3D-scene.

Parameters

sx_double	zoom factor

StartRotation

Method starts the rotation of the 3D-scene around the screen axis according to the mouse position.

Parameters

sx_int	x-coordinate of the mouse
sx_int	y-coordinate of the mouse

StartPan

Method starts the pan of the 3D-scene around the screen axis according to the mouse position.

Parameters

sx_int	x-coordinate of the mouse
sx_int	y-coordinate of the mouse

Rotation

Method continues the rotation of the 3D-scene.

Parameters

sx_int	x-coordinate of the mouse
sx_int	y-coordinate of the mouse

Pan

Method continues the pan of the 3D-scene.

Parameters

sx_int	x-coordinate of the mouse
sx_int	y-coordinate of the mouse

Resize

Method must be called when the window supporting the 3D-scene changes size.

EnableGrid

Method activates the grid in the 3D-scene if parameter is true, and disactivates the grid if parameter is false.

Parameters

sx_bool	enable

EnableTrihedron

Method displays the trihedron in the 3D-scene if parameter is true, and disactivates the trihedron if parameter is false.

Parameters

sx_bool	enable

Select

Method selects and highlights a shape in the 3D-scene that is located in the mouse position and returns this shape.

Parameters

sx_int	x-coordinate of the mouse
sx_int	y-coordinate of the mouse

Returns

IShapeVis

Select

Method selects and highlights a shape in the 3D-scene that is in the input parameter.

Parameters

IShapeVis	input shape

Erase

Method hides a shape in the 3D-scene that is in the input parameter.

Parameters

IShapeVis	input shape

ClearSelected

Method empties previous selected shapes in the 3D-scene.

UpdateSelected

Method updates the list of selected shapes in the 3D-scene: i.e. highlights the newly selected ones and unhighlights previously selected shapes.

EraseSelected

Method hides selected shapes in the 3D-scene.

Update

Method updates the 3D-scene. The same as Redraw().

FitAll

Method resets the orientation of the 3D-scene and adjusts view parameters to fit the displayed scene, respecting height / width ratio, according to the boundary box of all structures displayed in the 3D-scene. Updates the 3D-scene.

ZFitAll

Method changes Z-min and Z-max planes of projection volume to match the displayed shapes.

RemoveAll

Method removes all the shapes in the 3D-scene.

Reset

Method resets the centering and the orientation of the 3D-scene.

3.1.22 IShapeVis

Interface provides methods for internal geometric objects of the IView interface.

set_Color

Method sets color to current shape.

Parameters

sx_color_rgb	color

set_Shape

Method sets new shape to current shape.

Parameters

IShape	new shape

IsEqual

This method compares the input shape with the current shape and returns true if they are equal, and false if they aren't.

Parameters

IShapeVis	input shape

Returns

sx_bool

3.2 Structures

3.2.1 SX::sx_matrix

A structure is used to specify a transformation matrix. It is a matrix 3x4.

The transformations with the help of transformation matrix can be represented as follow:

V1	V2	V3	T	XYZ	XYZ‘
m11	m12	m13	m14	x	x‘
m21	m22	m23	m24	y	y‘
m31	m33	m33	m34	z	z‘

0	0	0	1	1	1

where {V1, V2, V3} defines the vectorial part of the transformation and T defines the translation part of the transformation.

Members

m11	Type sx_double
m12	Type sx_double
m13	Type sx_double
m14	Type sx_double
m21	Type sx_double
m22	Type sx_double
m23	Type sx_double
m24	Type sx_double
m31	Type sx_double
m32	Type sx_double
m33	Type sx_double
m34	Type sx_double

3.2.2 SX::sx_point

A structure is used to specify a point in 3D.

Members

X	Type sx_double. The x-coordinate
Y	Type sx_double. The y-coordinate
Z	Type sx_double. The z-coordinate

3.2.3 SX::sx_color_rgb

A structure is used to specify a header for output file.

Members

R	Type sx_double. The red component
G	Type sx_double. The green component
B	Type sx_double. The blue component

3.2.4 SX::sx_bolt_hex

A structure is used to specify bolt with hexagon head.

D	Thread diameter
B	Thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
S	Width across flats
K	Height of head
Dw	Diameter of washer face
C	Depth of washer face

3.2.5 SX::sx_bolt_hexscoket

A structure is used to specify bolt with hex socket head.

D	Threaded diameter
B	Thread length

Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
d1	Diameter of head
K	Height of head
S	Key size of socket
T	Depth of socket

3.2.6 SX::sx_bolt_hexscoket_countersunk

A structure is used to specify bolt with hex socket head and countersunk.

D	Threaded diameter
B	Thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
d1	Diameter of head
S	Key size of socket
T	Depth of socket

3.2.7 SX::sx_bolt_hexflange

A structure is used to specify bolt with hex socket head and flange.

D	Threaded diameter
B	Thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
d1	Diameter of head
S	Width across flats
K	Head height

3.2.8 SX::sx_bolt_carriage

A structure is used to specify bolt with round head and square neck.

D	Threaded diameter
B	Thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
d1	Diameter of head
K	Head height
V	Square width across flats
F	Square depth

3.2.9 SX::sx_bolt_hexset_dogpoint

A structure is used to specify bolt with hexset and full dog point.

D	Thread diameter
B	Thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
K	Head height
S	Width across flats

Dp	Dog point diamter
Zp	Dog point lenght

3.2.10 SX::sx_bolt_hexset_cone

A structure is used to specify bolt with hexset and flat cone point.

D	Thread diameter
B	Thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
K	Head height
S	Width across flats
Dp	Dog point diamter
Zp	Dog point lenght

3.2.11 SX::sx_bolt_t_head

A structure is used to specify bolt with T-Head (Hammerhead)

D	Thread diameter
B	Thread length
Ls	Length of unthreaded shank
s1	Width of head
K	Height of head

3.2.12 SX::sx_bolt_eye

A structure is used to specify eye-bolt (Swing bolt)

D	Thread diameter
B	Thread length
Ls	Length of unthreaded shank
d1	Head diameter (Sphere)
d2	Diameter of bore
S	Width of head

3.2.13 SX::sx_bolt_u

A structure is used to specify U-bolt (Steel strap)

D	Thread diameter
B	Thread length
Ls	Length of unthreaded shank
d1	Diameter of tube

3.2.14 SX::sx_bolt_ring

A structure is used to specify ring-bolt

D	Thread diameter
B	Thread length
d1	Diameter of tube
d2	Inner diameter of ring
d3	Diameter of head

K	Height of head

3.2.15 SX::sx_bolt_spigot

A structure is used to specify bolt with spigot and stud end

D	Thread diameter
B	Thread length
Ds	Diameter of unthread shank
Ls	Length of unthread shank
d1	Diameter of head
S	Width across flats
K	Height of head

3.2.16 SX::sx_screw_cheeseslotted

A structure is used to specify slotted cheese head screw

D	Thread diameter
B	Thread length
d1	Diameter of head
K	Height of head
N	Width of slot
T	Depth of slot

3.2.17 SX::sx_screw_panslotted

A structure is used to specify slotted pan head screw

D	Thread diameter
B	Thread length
d1	Diameter of head
K	Height of head
N	Width of slot
T	Depth of slot

3.2.18 SX::sx_screw_countersunk_slotted

A structure is used to specify slotted countersunk head screw

D	Thread diameter
B	Thread length
d1	Diameter of head
N	Width of slot
T	Depth of slot

3.2.19 SX::sx_screw_countersunk_raisedslotted

A structure is used to specify slotted raised countersunk head screw

D	Thread diameter
B	Thread length
d1	Diameter of head
K	Height of head
N	Width of slot

T	Depth of slot

3.2.20 SX::sx_screw_fillister_crossrecessed

A structure is used to specify cross recessed fillister head screw

D	Thread diameter
B	Thread length
d1	Diameter of head
K	Height of head
A	Width of recess
N	Thickness of recess

3.2.21 SX::sx_screw_countersunk_crossrecessed

A structure is used to specify cross recessed countersunk head screw

D	Thread diameter
B	Thread length
d1	Diameter of head
A	Width of recess
N	Thickness of recess

3.2.22 SX::sx_screw_countersunkraised_crossrecessed

A structure is used to specify cross recessed raised countersunk head screw

D	Thread diameter
B	Thread length
d1	Diameter of head
K	Height of head
A	Width of recess
N	Thickness of recess

3.2.23 SX::sx_screw_hexsocket_button

A structure is used to specify hexagon socket button head screw

D	Thread diameter
B	Thread length
d1	Diameter of head
K	Height of head
S	Key size of socket
T	Depth of socket

3.2.24 SX::sx_screw_countersunk_squareneck

A structure is used to specify countersunk head square neck bolt

D	Thread diameter
B	Thread length
d1	Diameter of head
V	Square width across flats
F	Square dept

3.2.25 SX::sx_screw_ squarecollar

A structure is used to specify countersunk head square neck bolt

D	Thread diameter
B	Thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
S	Width across flats
K	Height of head
Dw	Diameter of collar
C	Depth of collar

3.2.26 SX::sx_screw_thumb

A structure is used to specify thumb screw

D	Thread diameter
B	Thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank
d1	Diameter of head
K	Height of head

3.2.27 SX::sx_screw_set_slotted_dogpoint

A structure is used to slotted set screw with dog point

D	Thread diameter
B	Thread length
Dp	Dog point diameter
Zp	Dog point length
N	Width of slot
T	Depth of slot

3.2.28 SX::sx_screw_set_slotted_conepoint

A structure is used to slotted set screw with cone point

D	Thread diameter
B	Thread length
N	Width of slot
T	Depth of slot

3.2.29 SX::sx_screw_set_hexsocket_dogpoint

A structure is used to hex socket set screw with dog point

D	Thread diameter
B	Thread length
Dp	Dog point diameter
Zp	Dog point length
S	Key size of socket
T	Depth of socket

3.2.30 SX::sx_screw_set_ hexsocket_conepoint

A structure is used to hex socket set screw with cone point

D	Thread diameter
B	Thread length
S	Key size of socket
T	Depth of socket

3.2.31 SX::sx_stud_doubleend

A structure is used to specify double end stud

D	Thread diameter
b1	First thread length
b2	Second thread length
Ds	Diameter of unthreaded shank
Ls	Length of unthreaded shank

3.2.32 SX::sx_gear

A structure is used to specify gear

Z	Number of teeth
Mn	Tooth module
B	Gear width
Alpha	Alpha angle
Beta	Angle of the teeth
K	Tip modification factor (k)
X	Nominal addendum modification factor (x)
D	Diameter

3.2.33 SX::sx_gear_profile

A structure is used to specify gear (free profile)

z	Number of teeth
b	Gear width
beta	Angle of the teeth
d	Diameter

3.2.34 SX::sx_gear_double

A structure is used to specify double gear

z	Number of teeth
mn	Tooth module
b	Gear width
bn	Groove width
alpha	Alpha angle
beta	Angle of the teeth
k	Tip modification factor (k)
x	Nominal addendum modification factor (x)
d	Diameter
dn	Groove diameter

3.2.35 SX::sx_gear_double_profile

A structure is used to specify double gear (free profile)

z	Number of teeth
b	Gear width
bn	Groove width
beta	Angle of the teeth
d	Diameter
dn	Groove diameter

3.2.36 SX::sx_shaft_section

A structure is used to specify shaft section

pos	Position
da1	First outer diameter
da2	First inner diameter
di1	Second outer diameter
di2	Second inner diameter
l	Length
lch1	First chamfer (<0) or rounding (> 0)
lch2	Second chamfer (<0) or rounding (> 0)

3.2.37 SX::sx_groove_cross_hole

A structure is used to specify cross hole

x	Position
t	Depth
r	Radius
angle	Angle of rotation in radians. Angles are measured clockwise when looking along the Z-axis toward the origin. If angle = 0 then cross hole located on the X-axis

3.2.38 SX::sx_groove_parallel_key

A structure is used to specify groove parallel key

x	Position
l	Length
t	Depth
r	Radius
angle	Angle of rotation in radians. Angles are measured clockwise when looking along the Z-axis toward the origin. If angle = 0 then cross hole located on the X-axis

3.2.39 SX::sx_groove_notch_chevron

A structure is used to specify notch chevron

x	Position
t	Depth

3.2.40 SX::sx_groove_rectangular

A structure is used to specify groove rectangular

x	Position
t	Depth
r	Radius
m	Width

3.2.41 SX::sx_groove_slots

A structure is used to specify slots

x	Position
l	Length
n	Number of slots
d	Diameter
win	Inner width slot
wout	Outer width slot

3.3 Enumerations

3.3.1 SX:: sx_bearing_type

Specifies the type of a bearing.

Values

sx_bearing_type_SingleRowDeepGrooveBall	Single-row deep-groove ball
sx_bearing_type_DoubleRowDeepGrooveBall	Double-row deep-groove ball
sx_bearing_type_SingleRowAngularContactBall	Single-row angular contact ball
sx_bearing_type_DoubleRowAngularContactBall	Double-row angular contact ball
sx_bearing_type_SelfAligningBall	Self-aligning ball
sx_bearing_type_FourPoint	Four-point
sx_bearing_type_SingleDirectionThrustBall	Single-direction thrust ball
sx_bearing_type_DoubleDirectionThrustBall	Double-direction thrust ball
sx_bearing_type_SingleDirectionAngularContactThrustBall	Single-direction angular contact thrust ball
sx_bearing_type_DoubleDirectionAngularContactThrustBall	Double-direction angular contact thrust ball
sx_bearing_type_SingleRowCylindricalRoller	Single-row cylindrical roller
sx_bearing_type_DoubleRowCylindricalRoller	Double-row cylindrical roller
sx_bearing_type_TaperedRoller	Tapered roller
sx_bearing_type_BarrelRoller	Barrel roller
sx_bearing_type_SphericalRoller	Spherical roller
sx_bearing_type_SingleDirectionCylindricalRollerThrust	Single-direction cylindrical roller thrust
sx_bearing_type_DoubleDirectionCylindricalRollerThrust	Double-direction cylindrical roller thrust
sx_bearing_type_SphericalRollerThrust	Spherical roller thrust
sx_bearing_type_NeedleRoller	Needle roller

3.3.2 SX::sx_groove_type

Specifies the type of a groove.

Values

sx_groove_type_CrossHole	Cross hole
sx_groove_type_GrooveParallelKey	Groove parallel key

sx_groove_type_NotchChevron	Notch chevron
sx_groove_type_GrooveRectangular	Groove rectangular

4. APPLICATION EXAMPLES

Building the CAD Xpansion SDK functionality in your application is quite easy, anyway you can use the samples for a quick start. Example as a ready C++ project you can find in “DevRes\samples” folder.

In "DevRes\samples\export\" folder you can find the examples of shaft models export are given and assembly in the STEP format.

In " DevRes\samples\recognition\" folder you can find the example of shaft model import is given in STEP format, with parameters recognition of its elements. Recognized parameters are displayed in the console.

The project RecognizeShaft (DevRes\samples\recognition \RecognizeShaft\) presents an example of using the library cad-xpansion-32.dll.

One of the important features in this example project is a possibility to find out all components (parts) of an assembly consisting from many differents parts. After importing of a model from STEP-file a recursive function call has been done in order to search for all components of the assembly covering all levels, starting from the main node. This is one of the possible options (the most common option). Alternatively you can also select a node by name or number.

Inside this recursive function:

- For compounded assembly: the recursive call, starting from current node.

- For uncompounded node: you can get the geometry (IShape) and recognize shaft. The shaft recognition function receives geometry (IShape) for the analysis. After recognition process the recognized parameters of all sections and grooves on them will be displayed in the console.

Pay attention to the location of the header files for C++ development desktop or server applications using CAD Xpansion SDK (DevRes\Include\) and the library cad-xpansion-32.dll (Redist\).

5. CAD XPANSION VIEWER

Special test bilingual application CAD Xpansion Viewer.exe allows you to check the work of the export and import functions in the STEP, IGES, STL formats, and also recognition of element parameters by shaft import in the STEP format and store the recognized data in the XML format.

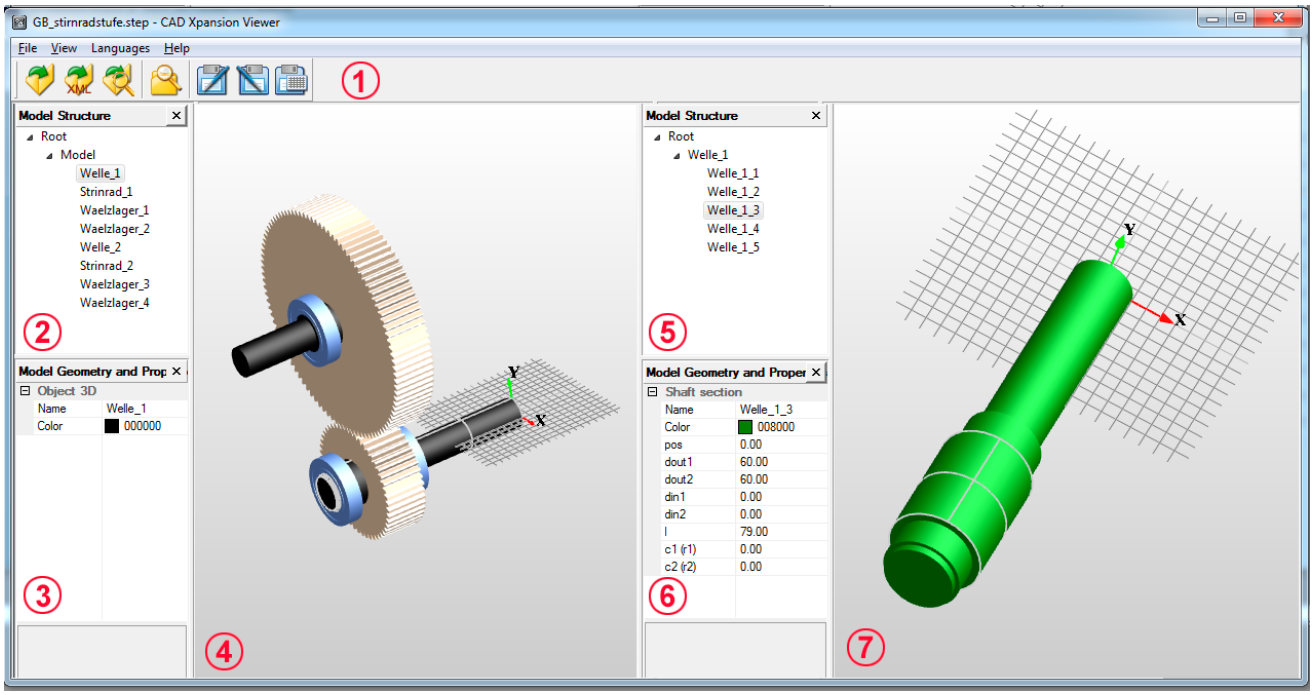
CAD Xpansion Viewer uses the CADViewer.dll library to create a 2-window viewer with a number of built-in capabilities to work with them.

In the CAD Xpansion Viewer, CADViewer.dll is statically built.

The CADViewer.dll library in turn uses the CAD Xpansion SDK library.

Input data for the export demonstration in the program is represented in XML format. Examples of such representation for all library objects can be found in TestApp \ Examples \.

The screenshot below shows the program window:



Specification:

1 – Program toolbar, from which the following actions are available:

 Import 3D model from STEP file and display it

 Open an existing XML document – internal description of 3D model

 Import and Recognise 3D model from STEP file and display it

 Recognition of the selected item or the entire model of the imported STEP-file

 Export model from left pane in STEP, IGES, STL format

 Export model from right pane in STEP, IGES, STL format

 Export shaft geometry data from right pane to internal description of 3D model (XML)

2 - Tree of the model structure (Import)

3 - Window of element parameters selected in the 3D scene (Import)

4 - 3D scene of Viewer (Import)

5 - Tree of the model structure (Recognize)

6 - Window of element parameters selected in the 3D scene (Recognize)

7- 3D scene of Viewer (Recognize)

5.1 Description of CADViewer.dll

The CADViewer.dll library is part of the CAD Xpansion SDK and assembles two viewers for imported and recognized models in one window, arranging them in this order: left viewer for imported model, right viewer for recognized model. Each viewer is a window for visualizing a 3-dimensional model with a window of the model tree and a properties window for editing. In the visualization window, the grid and coordinate axes can be displayed as desired by the user. The window of the model tree and the properties window can also be hidden at the user's request.

To import, export, recognize and visualize the model elements of each of the two viewers, use the cad-xpansion-32.dll library.

Thus, the CADViewer.dll library can be used by an application programmer to create a 2-window viewer that has this list of basic capabilities:

- Import, export, visualization of files in STEP, IGES, STL formats

- Visualization and editing of data presented in XML format

- Recognition of geometry data represented in the STEP format

- Recognition of the geometry of the selected node from the model tree of the imported data of the left viewer, represented in STEP format or in XML format

- Edit the properties of the recognized model

- Saving the data of the recognized model in XML format.

5.2 Example of embedding CADViewer.dll (static embedding of code)

The CADViewer.dll library contains a number of classes that provide various methods for working with each of the two viewers that implement the capabilities of this library.

In "DevRes\samples\CADViewer\" folder you can find the example of a small program that connects the frame from the CADViewer.dll library and creates a menu with a toolbar for using the methods of working with the 2-window viewer. The program is based on a simple MFC application created by the application wizard in Visual Studio 2013. To use the CADViewer.dll functionality, the following steps need to be performed: - The availability of CADViewer.dll sources in the CADViewer directory

- In the project properties Linker-> Input connect CADViewer.lib

- In the header file for the main application window, add the MainFrm.h header file

- Inherit the class of the main application window from the CMainFrame class described in CADViewer.dll

- Add the toolbar and the corresponding menu items

- Build the application.

6. LEGAL INFORMATION

6.1 CAD Xpansion SDK

6.2 OpenCascade

CAD Xpansion SDK based on the Open CASCADE Technology according to the public license in Appendix 2.

7. CONTACT US

For all organisational, technical, and legal questions please call us using the following contact data:

Soft Xpansion GmbH & Co. KG.
Königsallee 45, D-44789 Bochum
Tel.: +49 234 298 4171, Fax.: +49 234 298 4172
Internet: www.soft-xpansion.eu
E-Mail: support@soft-xpansion.com

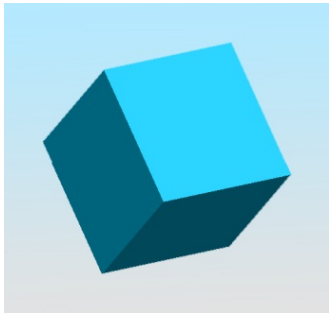
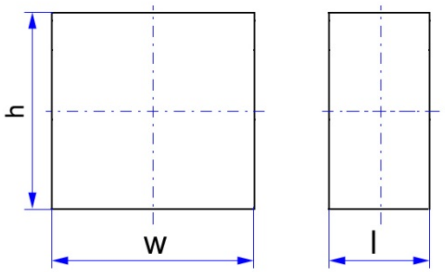
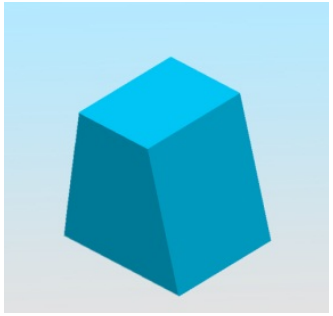
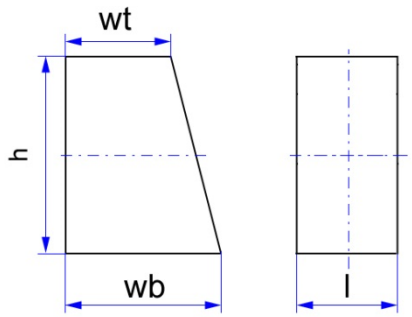
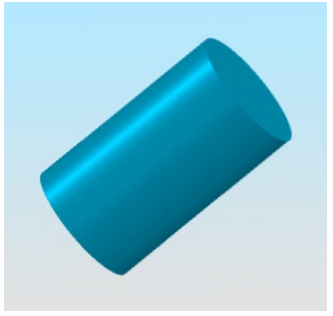
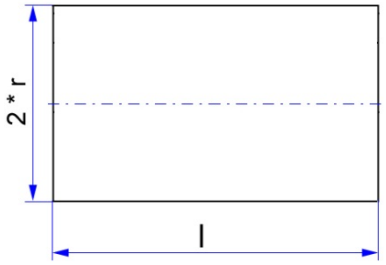
APPENDIX 1. OBJECT LISTS FOR DIFFERENT APPLICATION AREAS

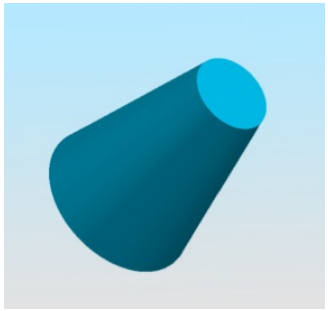
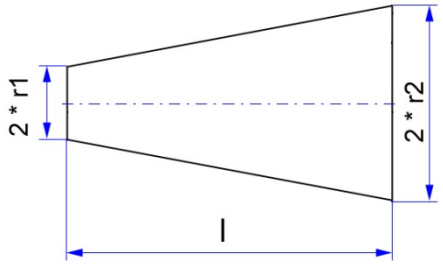
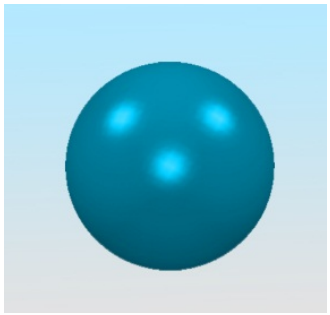
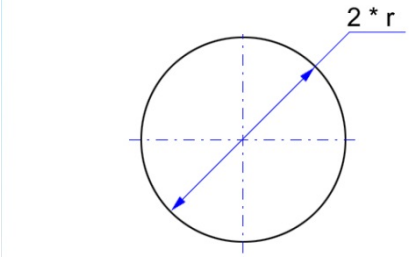
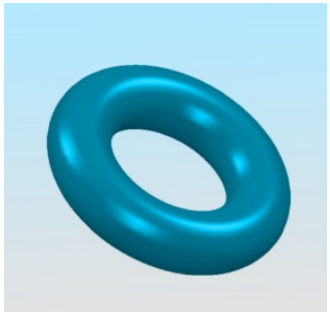
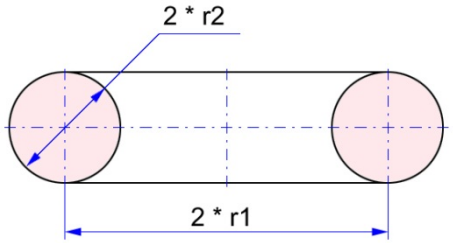
The tables below contain the objects have been implemented in the CAD Xpansion SDK.

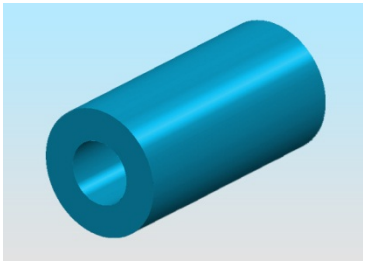
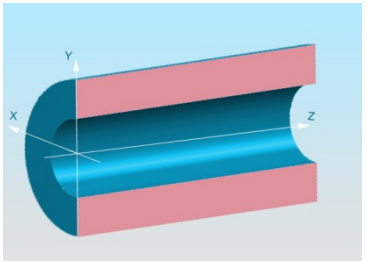
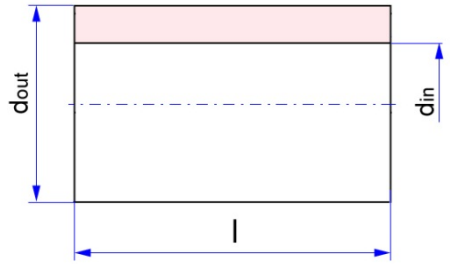
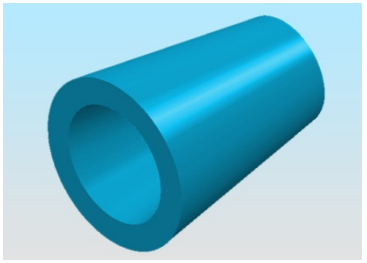
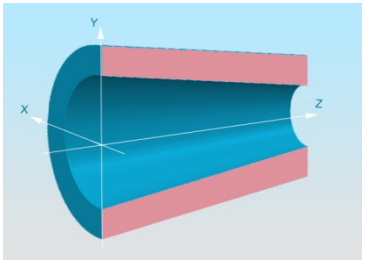
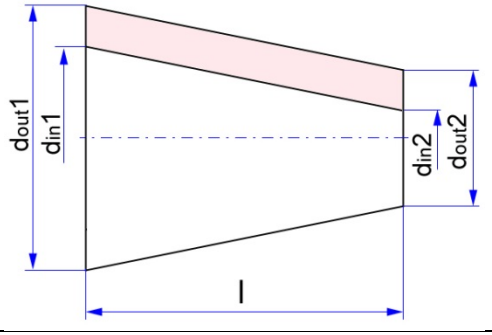
1.1 General Properties of Objects

N	Parameter Name	Parameter Values	Comments
1	Object name		Object name, which will be visible in the hierarchy tree (if such hierarchy is supported in the CAD system)
2	Object color	R, G, B	
3	Coordinates of the object position in the virtual scene	x, y, z	
4	Angle of object rotation around the vector	α x1, y1, z1	The vector coordinates (0,0,0, x1, y1, z1) and the angle of rotation round this vector are specified. The angle is counted clockwise, if you look along the vector direction
5	Scale of object	k	

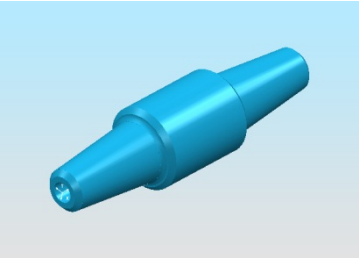
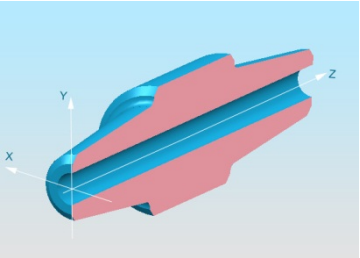
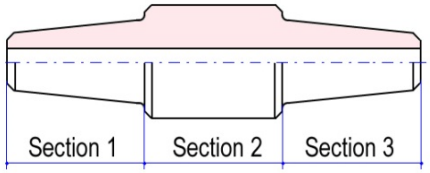
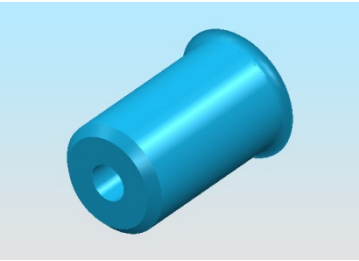
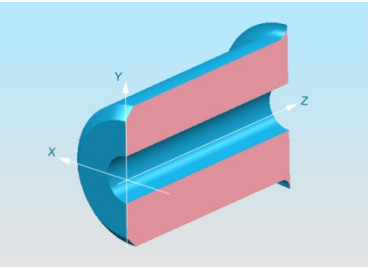
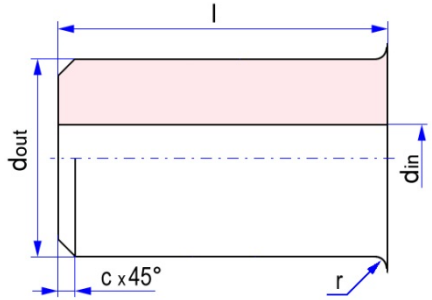
1.2 3D Geometry Objects

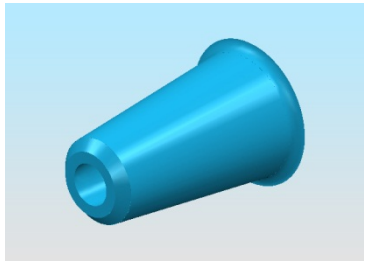
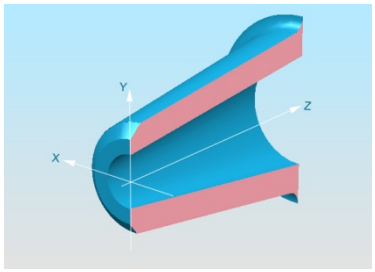
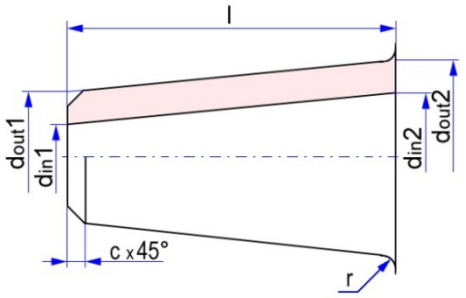
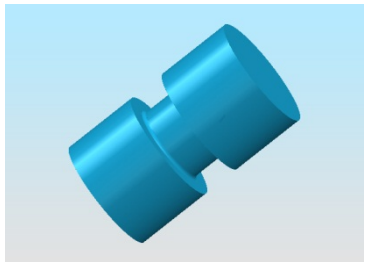
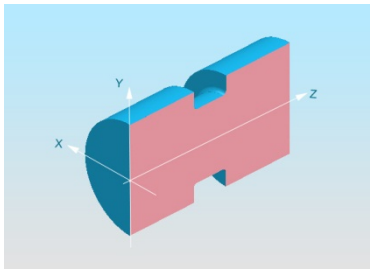
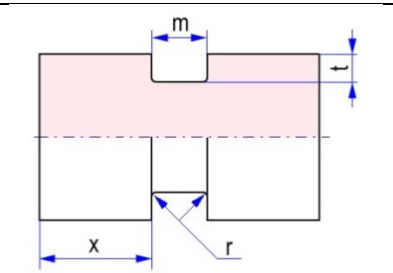
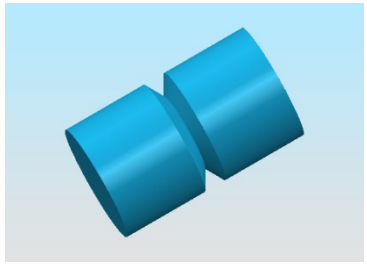
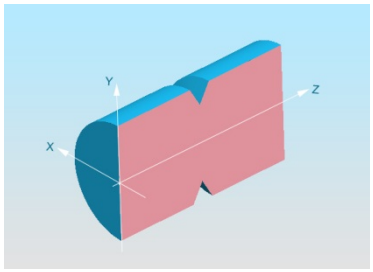
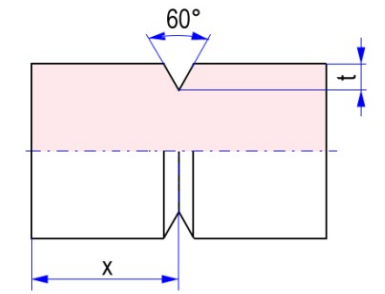
N	Object Name	3D View	2D View	Parameter Values	Comments
1	Elementary Objects				
1.1	Box			w - width h - height l - length	Special case: cube (w = h = l)
1.2	Wedge			wb - bottom width h - height l - length wt - top width.	Special case: 4 faced pyramid (full or truncated, direct or inclined)
1.3	Cylinder			r – radius l – length	Special case: Cylindrical segment

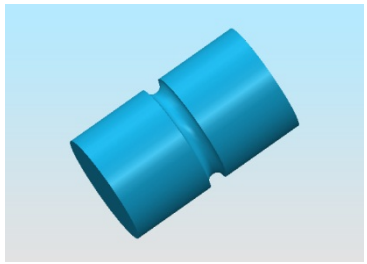
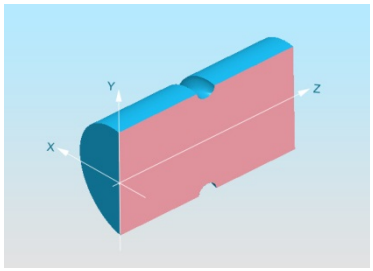
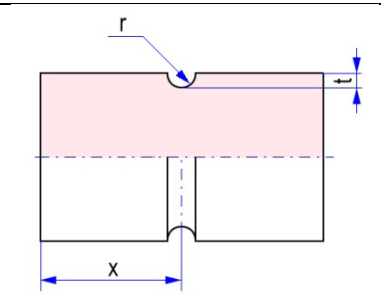
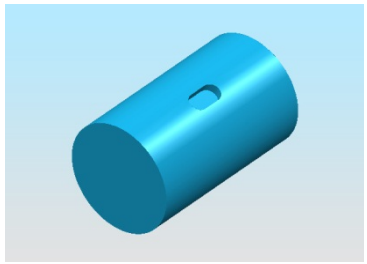
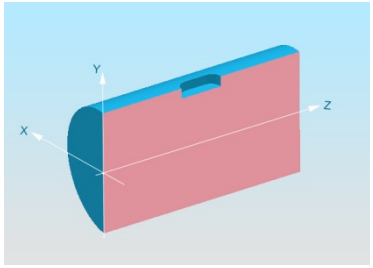
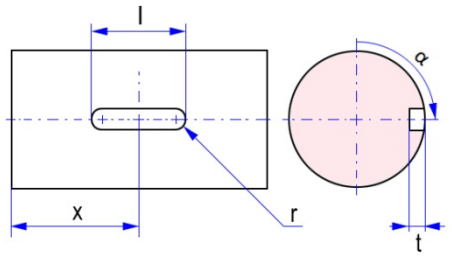
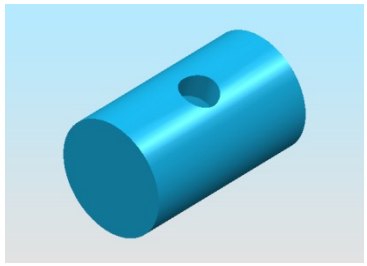
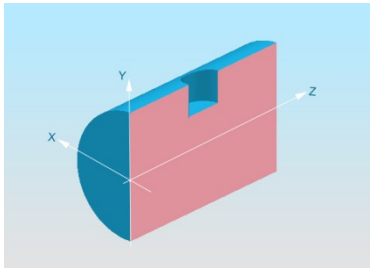
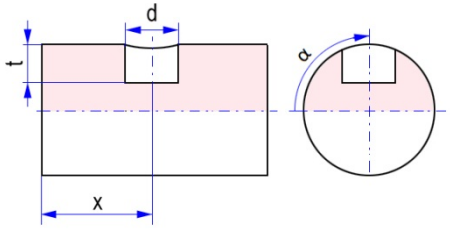
1.4	Cone			r1 - first radius. r2 - second radius. l - length	Special case: complete cone (r1 = 0)
1.5	Sphere			r – radius	Special case: spherical segment
1.6	Torus			r1 - first radius. r2 - second radius.	Special case: Toroidal segment

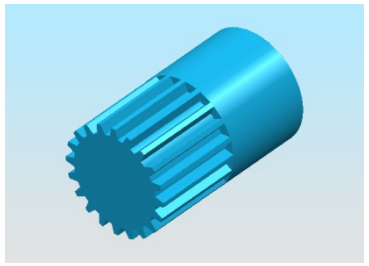
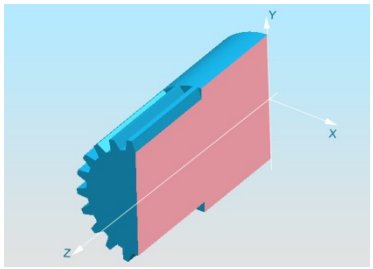
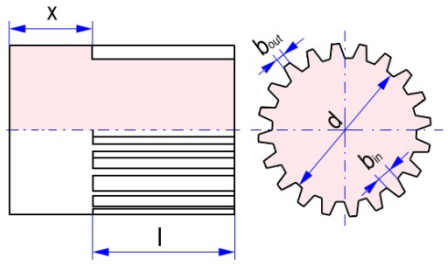
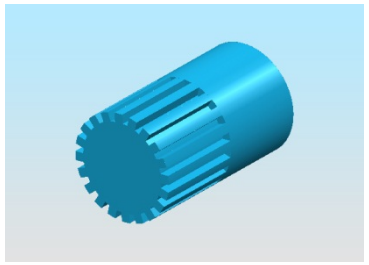
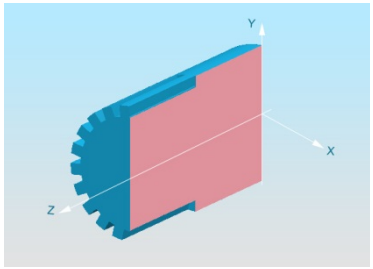
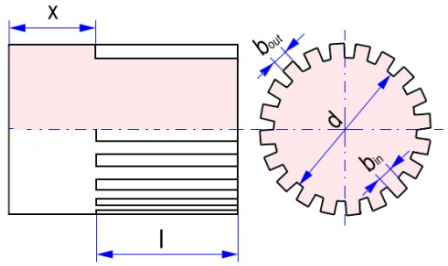
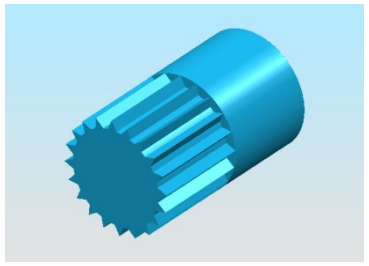
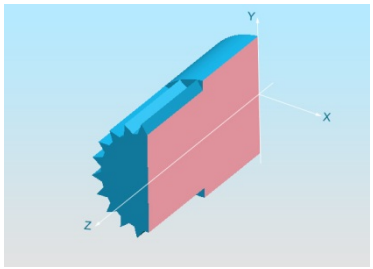
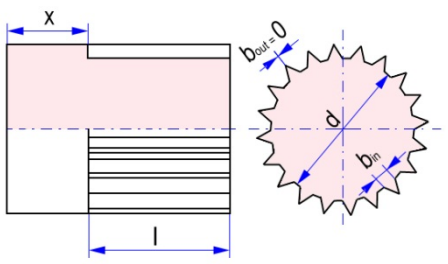
N	Object Name	3D View	3D View (cross-section)	2D View	Parameter Values	Comments
2	Derived Objects					
2.1	Hollow cylinder				d_{out} – outer diameter d_{in} - inner diameter l - length	For d_{in} = 0: solid cylinder
2.2	Hollow cone				$d_{out}1$ - first outer diameter $d_{out}2$ - second outer diameter $d_{in}1$ - first inner diameter $d_{in}2$ - second inner diameter l - length	For $d_{in}1=d_{in}2=0$: solid cone

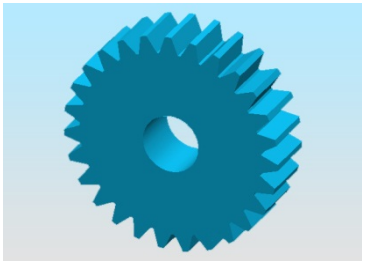
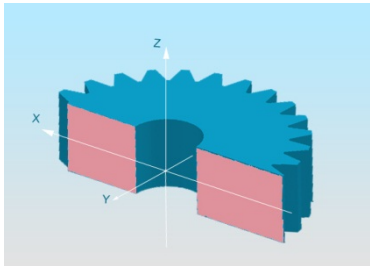
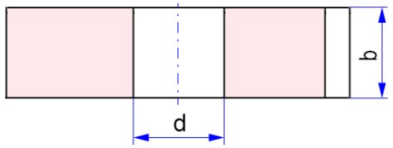
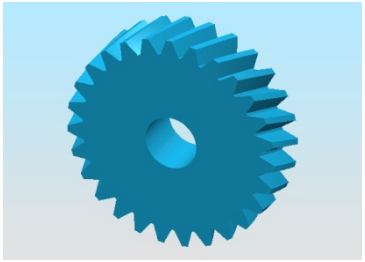
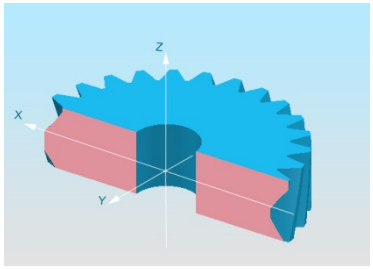
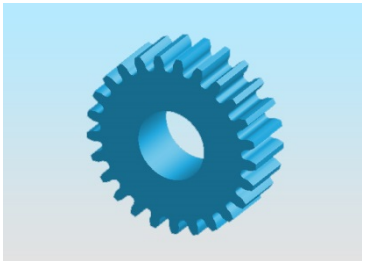
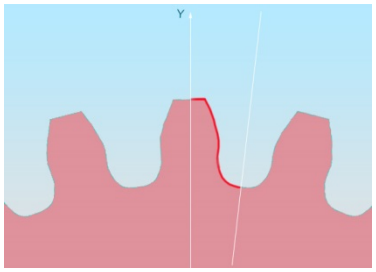
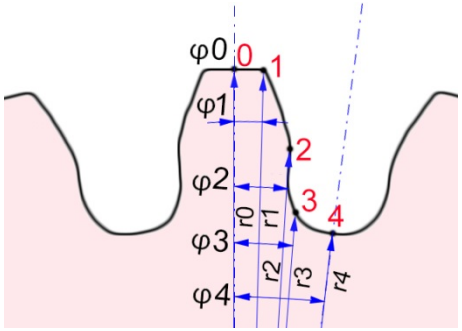
1.3 Mechanical Design

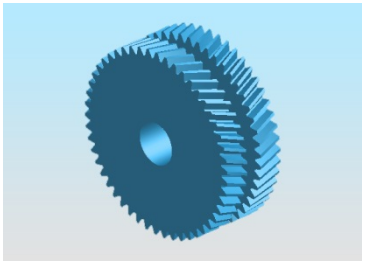
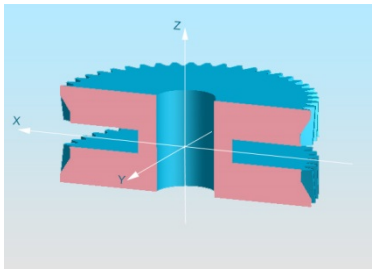
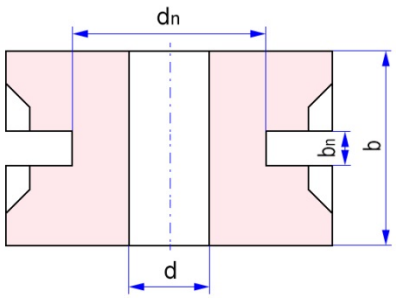
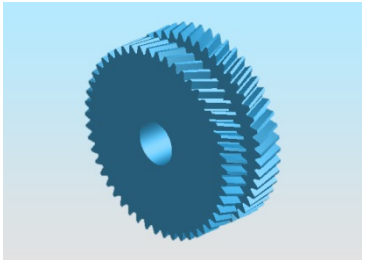
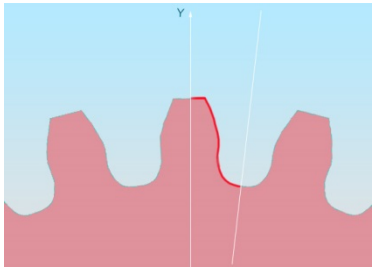
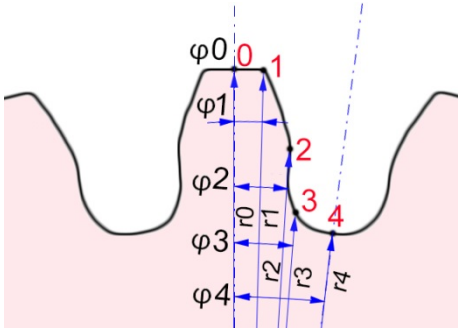
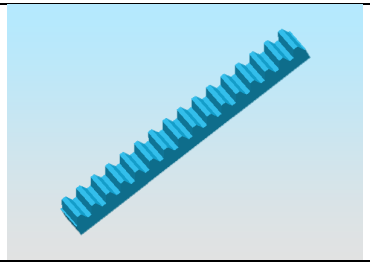
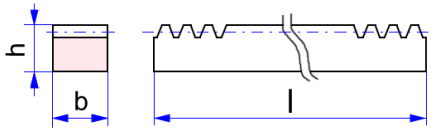
N	Object Name	3D View	3D View (cross-section)	2D View	Parameter Values	Comments
1	Shaft					
1.1	Shaft sections					Multisection shaft, is created from separate sections: Hollow cylinder and Hollow cone
1.2	Chamfer and rounding	Constructive elements available for creation in any section of a shaft. When necessary, a chamfer can be represented in the form of two sections of the shaft - by using of the objects 1.2.1 and 1.2.2. In this case, c = 0 and r - if necessary.				
1.2.1	Hollow cylinder				d_{out} - outer diameter d_{in} - inner diameter l – length c x 45° - chamfer width (< 0) r - rounding radius (> 0)	At one border of shaft section is possible an existence either chamfer or rounding

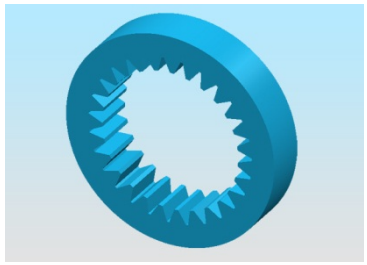
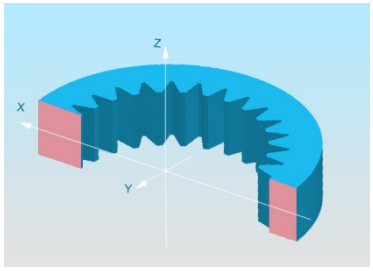
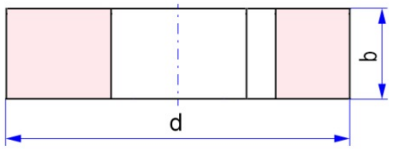
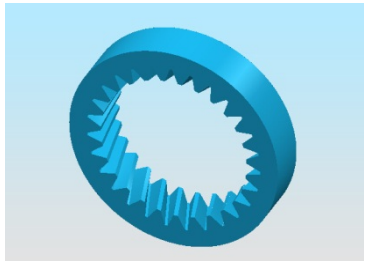
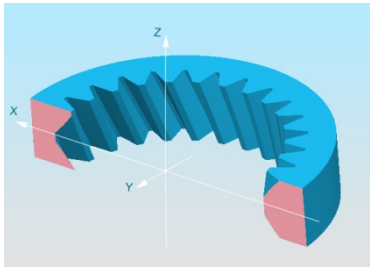
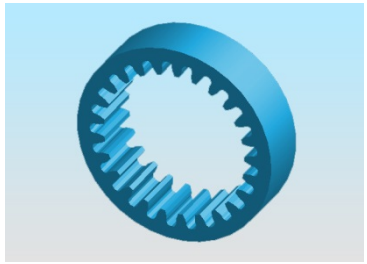
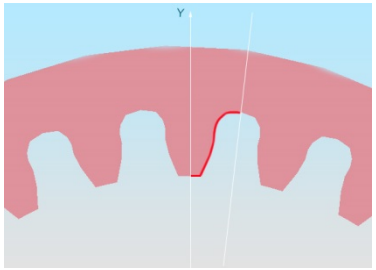
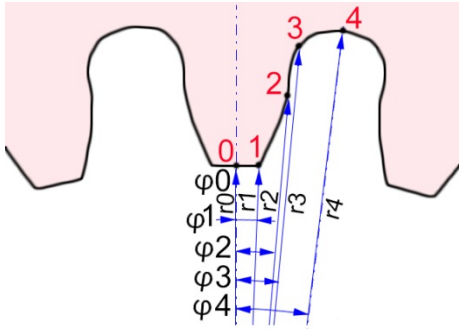
1.2.2	Hollow cone				$d_{out}1$ - initial outer diameter $d_{out}2$ - final outer diameter $d_{in}1$ - initial inner diameter $d_{in}2$ - final inner diameter l - length c x 45° - chamfer width (< 0) r - rounding radius (> 0)	At one border of shaft section is possible an existence either chamfer or rounding
1.3	Groove	Constructive elements available for creation in any section of a shaft The position “x” is specified relative to the shaft section heading. The number of these elements in the section is not limited.				
1.3.1	Rectangu-lar groove				x - position m - width t - depth r – radius	
1.3.2	Chevron notch				x - position t – depth	

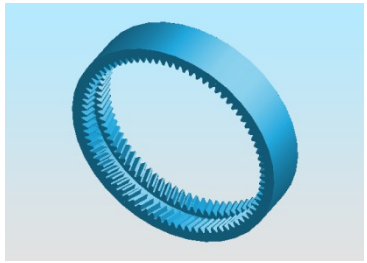
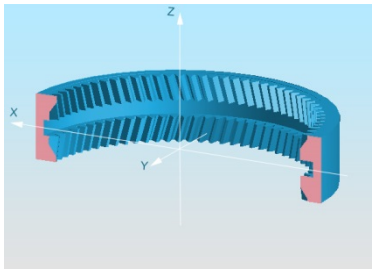
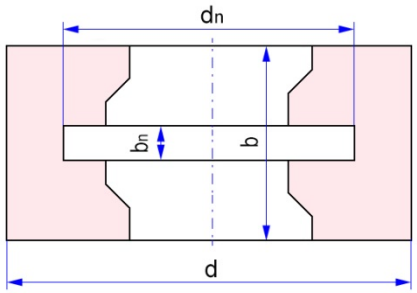
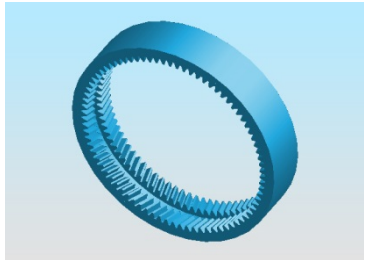
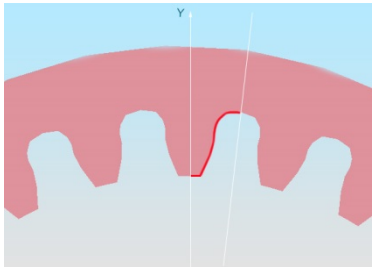
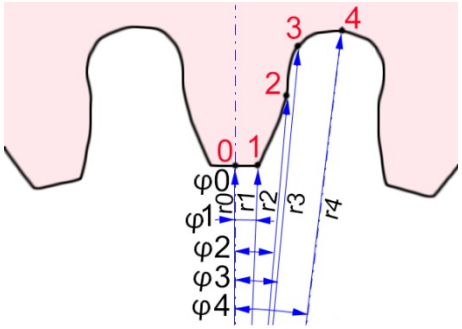
1.3.3	Circular groove				x - position t - depth r – radius	
1.3.4	Parallel key groove				x - position l - length t - depth r – radius α - angle of rotation (in radians)	For through-hole: t >= d_{out} Angles are measured clockwise when looking along the Z-axis toward the origin. If angle = 0 then cross hole located on the X-axis.
1.4	Hole	Constructive elements available for creation in any section of a shaft The position “x” is specified relative to the shaft section heading. The number of these elements in the section is not limited.				
1.4.1	Cross-hole				x – position t – depth d – diameter α - angle of rotation (in radians)	For through-hole: t >= d_{out} Angles are measured clockwise when looking along the Z-axis toward the origin. If angle = 0 then cross hole located on the X-axis.

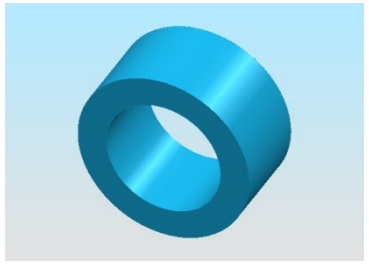
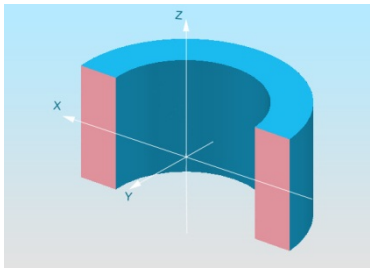
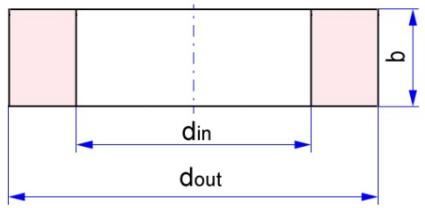
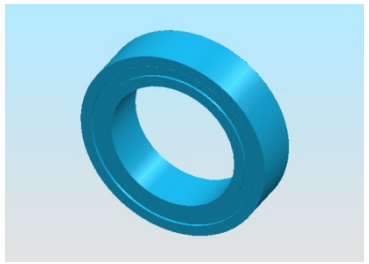
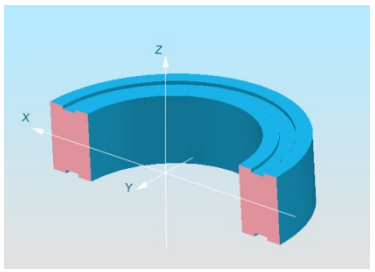
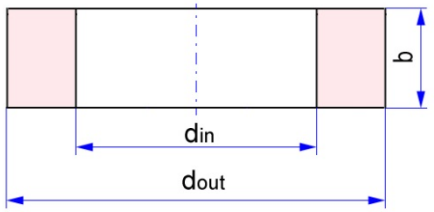
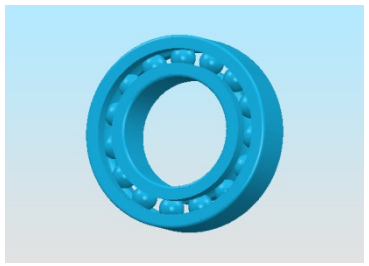
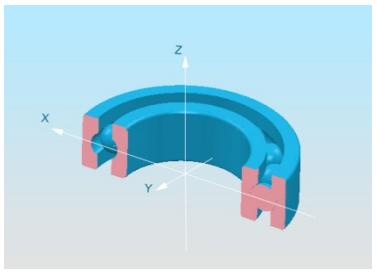
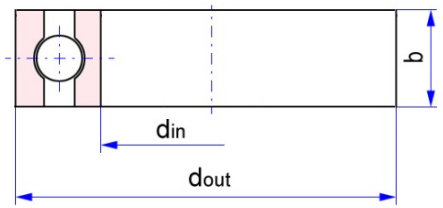
1.5	Slots	Constructive elements available for creation in any section of a shaft The position “x” is specified relative to the shaft section heading. The number of these elements in the section is not limited.				
1.5.1	Wedge slots				x - position l - length n - number of slots d - inner diameter b_{in} - root thickness b_{out} - tip thickness	$b_{in} > b_{out}$
1.5.2	Rectangular slots				x - position l - length n - number of slots. d - inner diameter b_{in} - root thickness b_{out} - tip thickness	$b_{in} = b_{out}$
1.5.3	Triangular slots				x - position l - length n - number of slots. d - inner diameter b_{in} - root thickness b_{out} - tip thickness	$b_{out} = 0$
2	Spur gear					

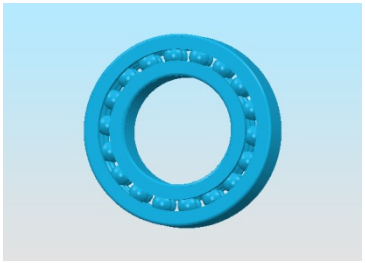
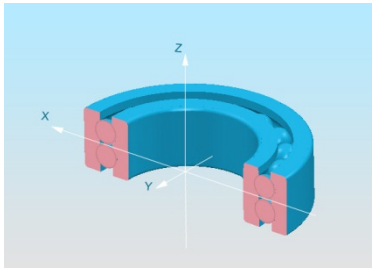
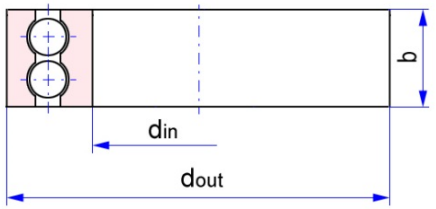
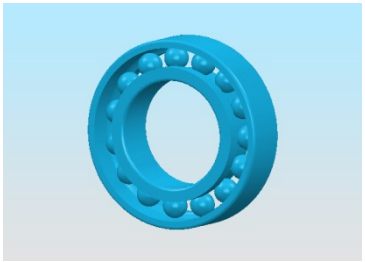
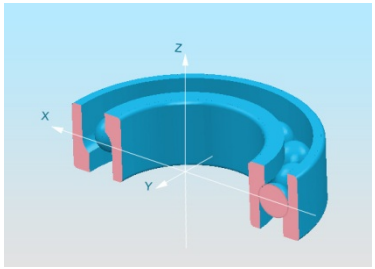
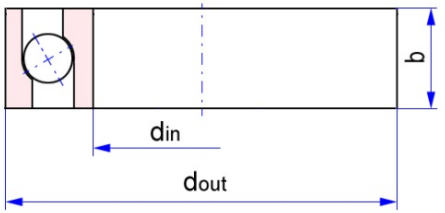
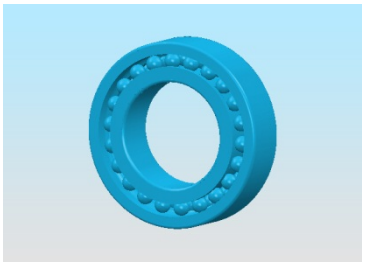
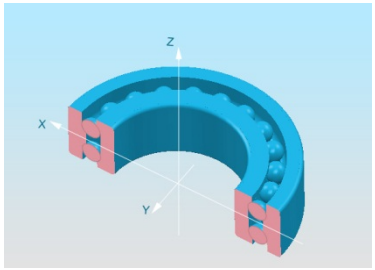
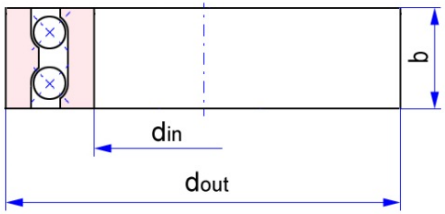
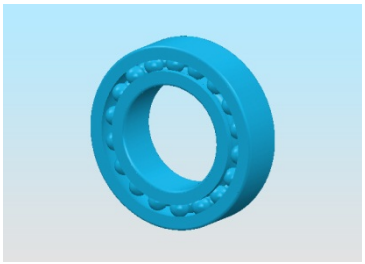
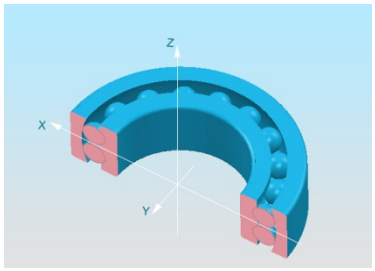
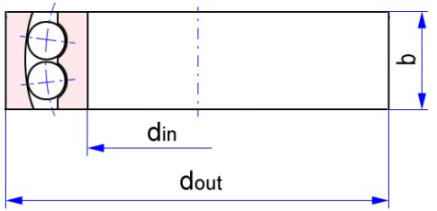
<table>
<tr><td>2.1</td><td>External gear</td><td colspan="4"></td></tr>
<tr><td>2.1.1</td><td>Straight teeth</td><td>  </td><td>  </td><td rowspan="2">  </td><td rowspan="2">z - number of teeth
mn - tooth module
b - gear width
α - alpha angle
β - angle of the teeth
k - tip modification factor
x - nominal addendum modification factor
d - inner diameter</td><td></td></tr>
<tr><td>2.1.2</td><td>Skew teeth</td><td>  </td><td>  </td><td></td></tr>
<tr><td>2.1.3</td><td>Free-profile</td><td>  </td><td>  </td><td>  </td><td>Tooth profile
r1, φ1 (in radians)
r2, φ2
r3, φ3
r4, φ4...
z - number of teeth
b - gear width
β - angle of the teeth
d - inner diameter
spline - approximation method (false/true)</td><td>The number of the profile points is not limited</td></tr>
</table>

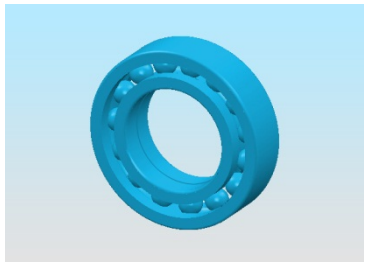
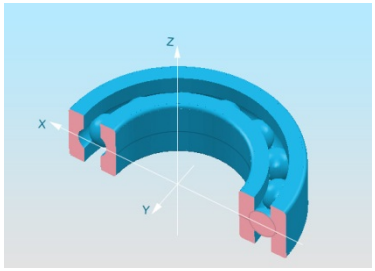
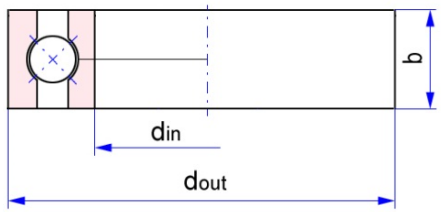
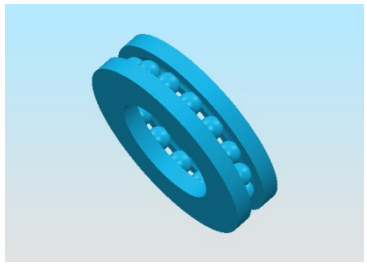
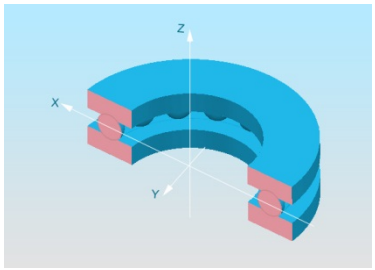
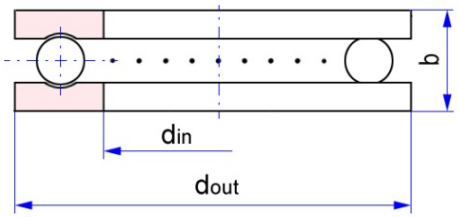
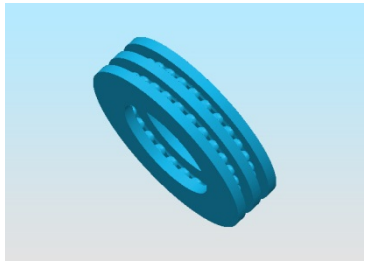
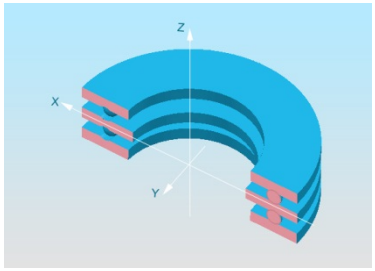
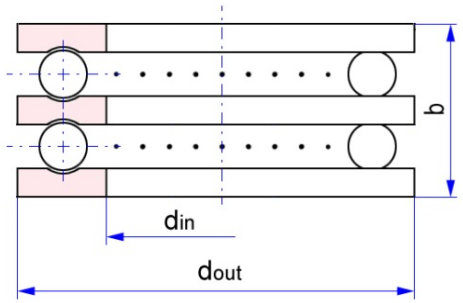
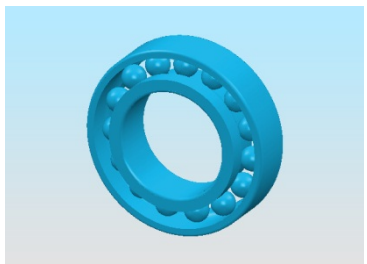
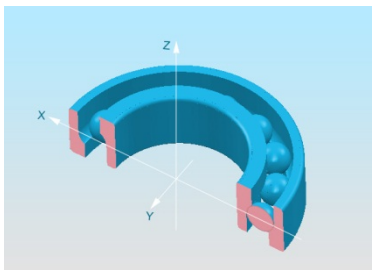
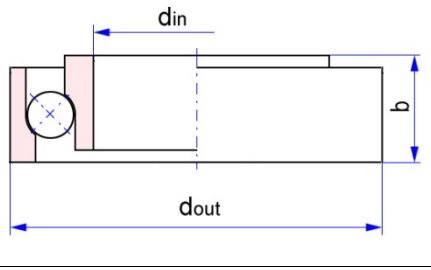
2.1.4	Duoble helical				z - number of teeth mn - tooth module b - gear width b_n- grove width α - alpha angle β - angle of the teeth k - tip modification factor x - nominal addendum modification factor d - inner diameter d_n - grove diameter	
2.1.5	Duoble helical Free-profile				Tooth profile r1, φ1 (in radians) r2, φ2 r3, φ3 r4, φ4... z - number of teeth b - gear width b_n- grove width β - angle of the teeth d - inner diameter d_n - grove diameter spline - approximation method (false/true)	
2.1.6	Gear rack				mn - tooth module b - width h – height l – length	

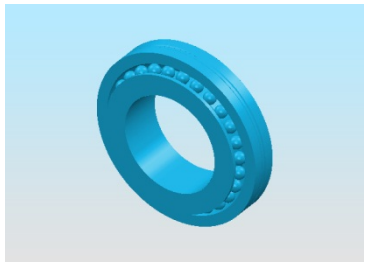
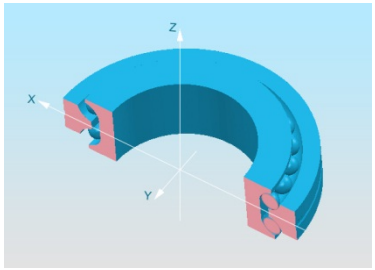
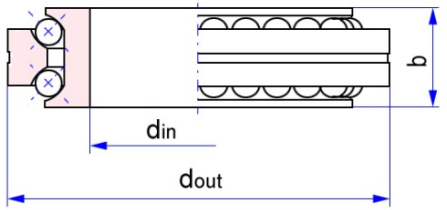
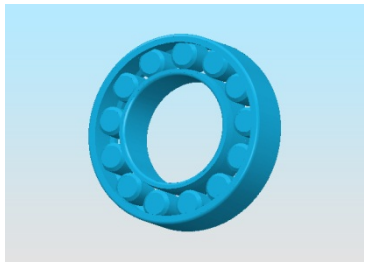
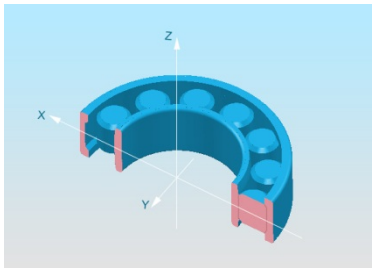
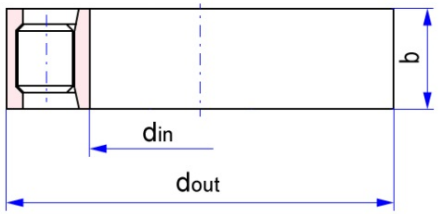
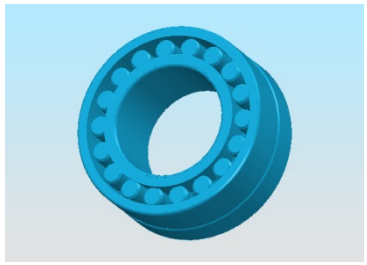
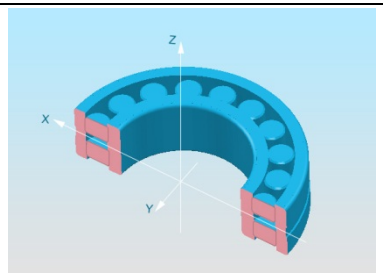
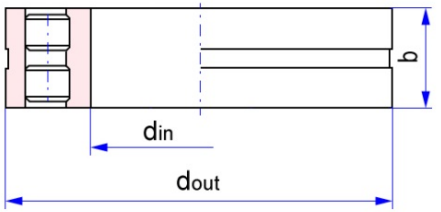
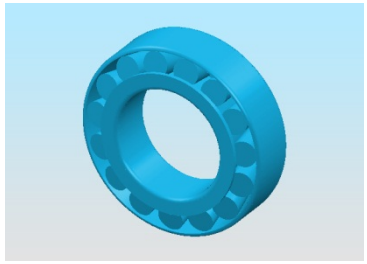
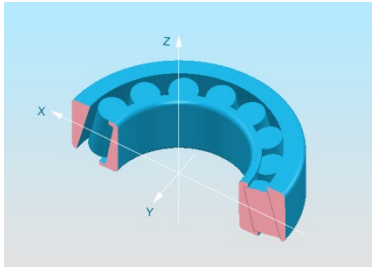
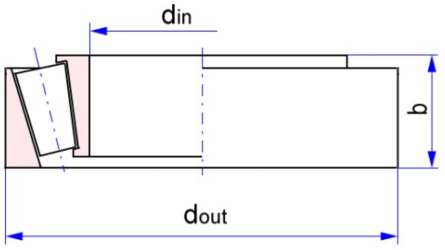
<table>
<tr><td>2.2</td><td>Internal gear</td><td colspan="4"></td></tr>
<tr><td>2.2.1</td><td>Straight teeth</td><td>  </td><td>  </td><td rowspan="2">  </td><td rowspan="2">z - number of teeth
mn - tooth module
b - gear width
α - alpha angle
β - angle of the teeth.
k - tip modification factor
x - nominal addendum modification factor
d - outer diameter</td><td></td></tr>
<tr><td>2.2.2</td><td>Skew teeth</td><td>  </td><td>  </td><td></td></tr>
<tr><td>2.2.3</td><td>Free-profile</td><td>  </td><td>  </td><td>  </td><td>Tooth profile
r1, φ1 (in radians)
r2, φ2
r3, φ3
r4, φ4...
z - number of teeth
b - gear width
β - angle of the teeth
d - outer diameter
spline - approximation method (false/true)</td><td>The number of the profile points is not limited</td></tr>
</table>

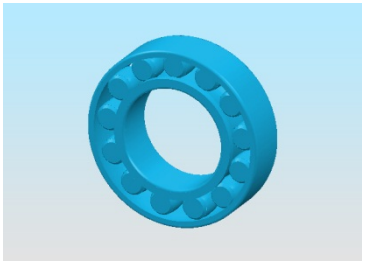
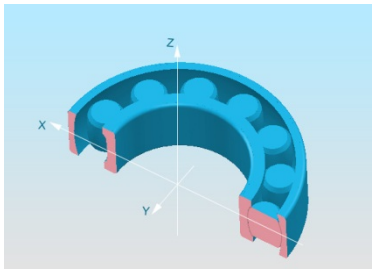
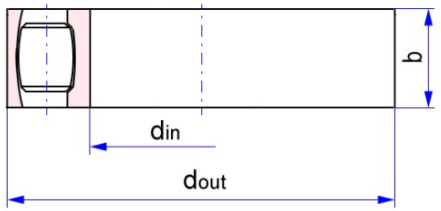
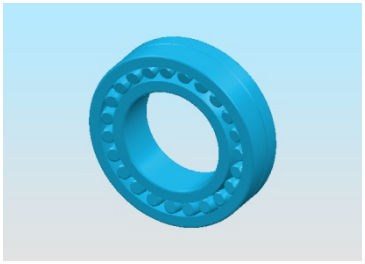
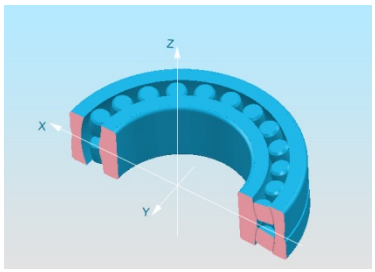
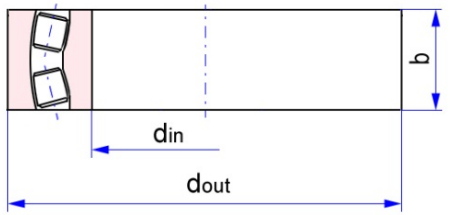
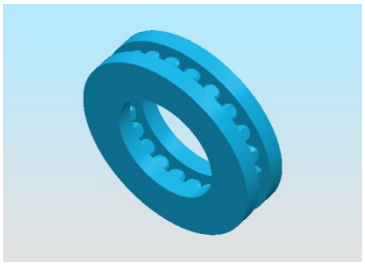
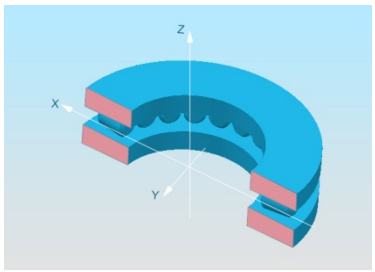
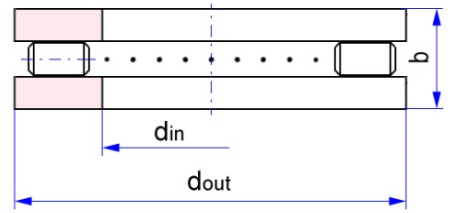
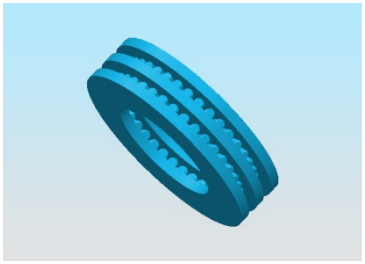
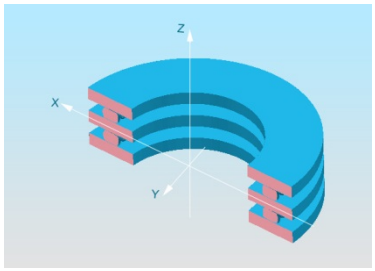
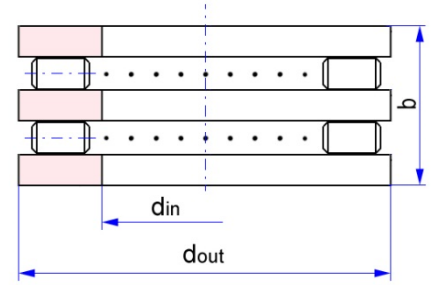
2.2.4	Duoble helical				z - number of teeth mn - tooth module b - gear width b_n- grove width α - alpha angle β - angle of the teeth k - tip modification factor x - nominal addendum modification factor d - outer diameter d_n - grove diameter	
2.2.5	Duoble helical Free-profile				Tooth profile r1, φ1 (in radians) r2, φ2 r3, φ3 r4, φ4... z - number of teeth b - gear width b_n- grove width β - angle of the teeth d - outer diameter d_n - grove diameter spline - approximation method (false/true)	
3	Bearing					
3.1	Sliding bearing					

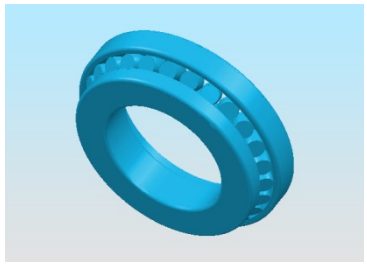
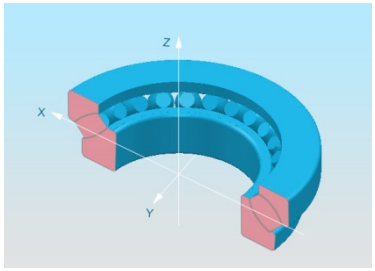
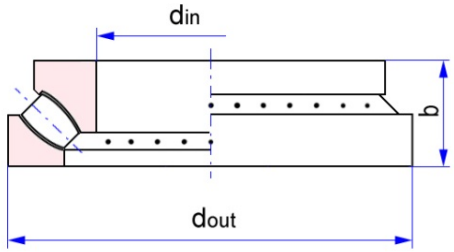
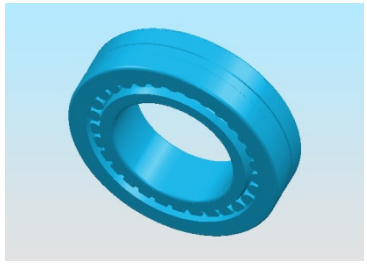
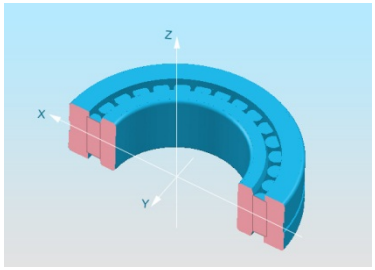
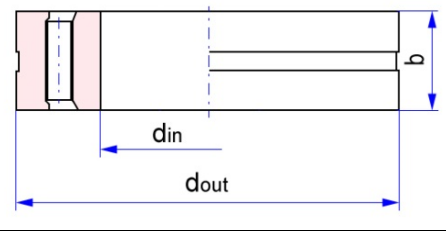
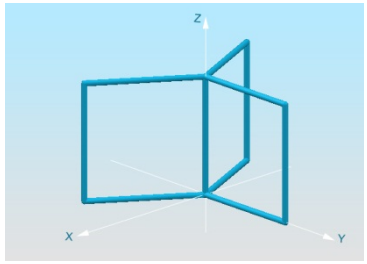
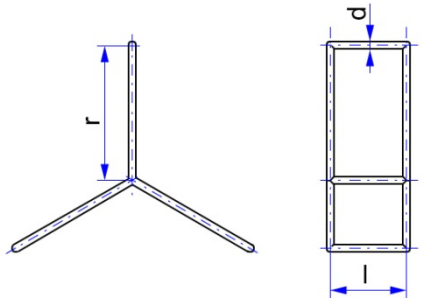
3.1.1	Sliding bearing				d_{out} - outer diameter d_{in} - inner diameter b - width	
3.2	Roller bearing					
3.2.1	Simplified presenta-tion				d_{out} - outer diameter d_{in} - inner diameter b - width	Simplified presentation in the form of 3 cylinders
3.2.2	Single-row deep-groove ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 1 (by MDESIGN)

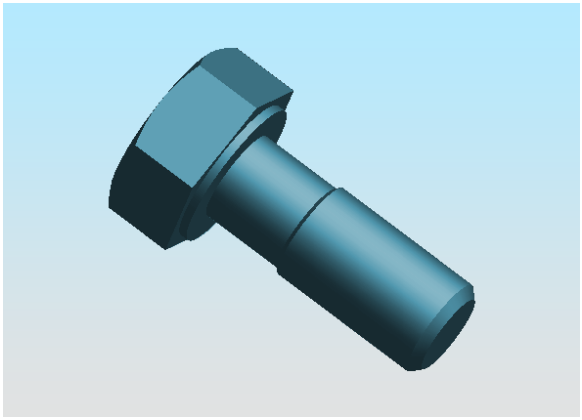
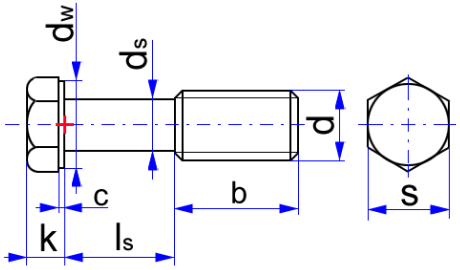
3.2.3	Double-row deep-groove ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 2 (by MDESIGN)
3.2.4	Single-row angular contact ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 3 (by MDESIGN)
3.2.5	Double-row angular contact ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 4 (by MDESIGN)
3.2.6	Self-aligning ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 5 (by MDESIGN)

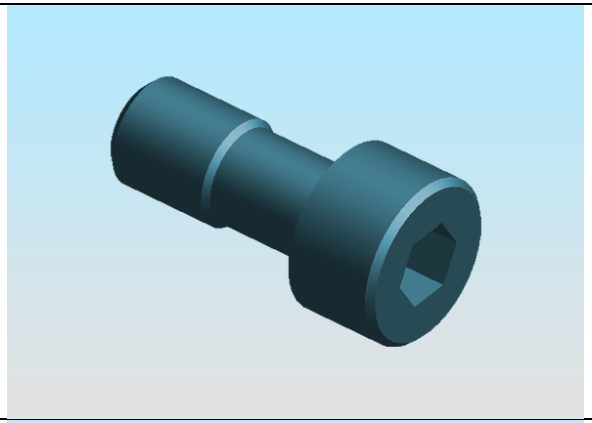
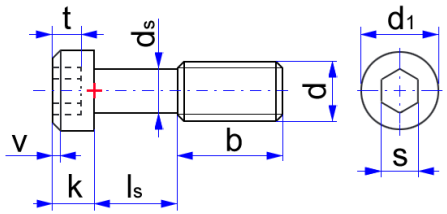
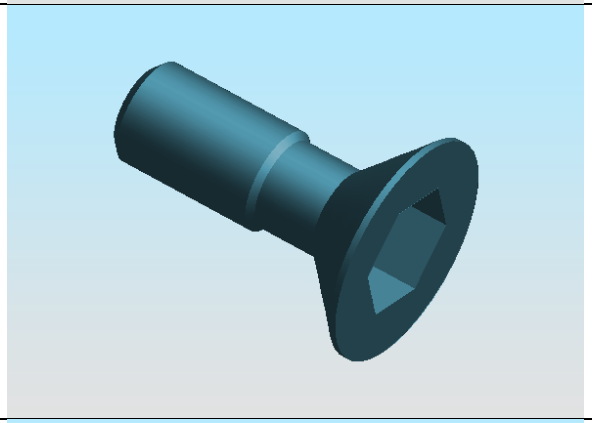
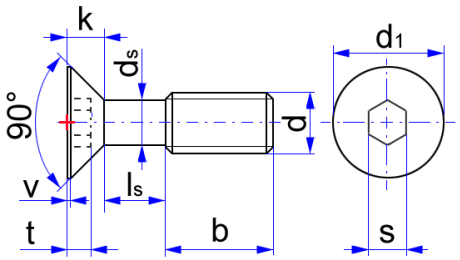
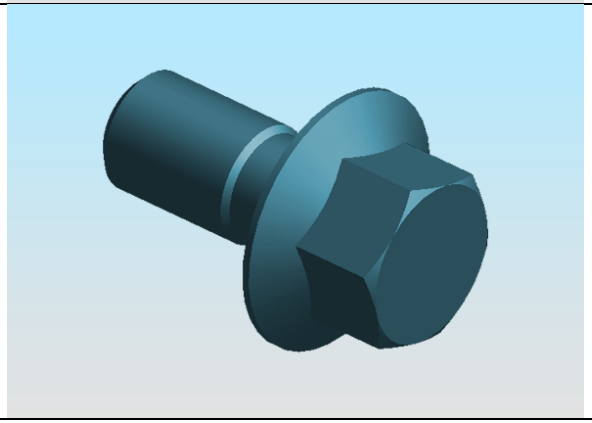
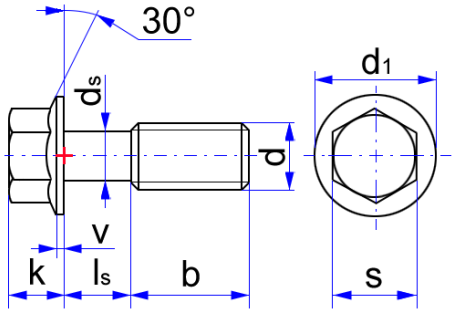
3.2.7	Four-point				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 6 (by MDESIGN)
3.2.8	Single-direction thrust ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 7 (by MDESIGN)
3.2.9	Double-direction thrust ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 8 (by MDESIGN)
3.2.10	Single-direction angular contact thrust ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 9 (by MDESIGN)

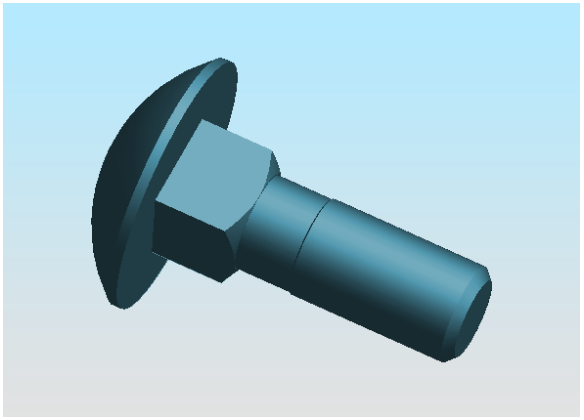
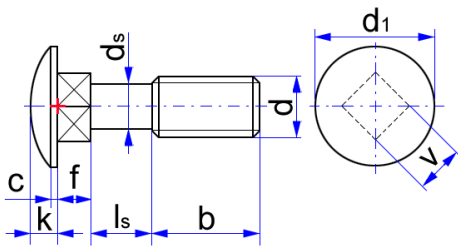
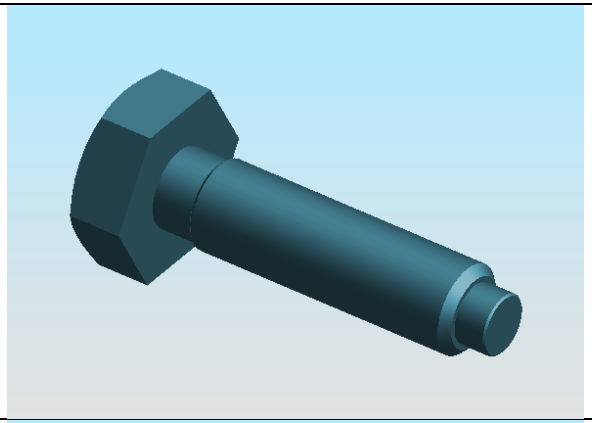
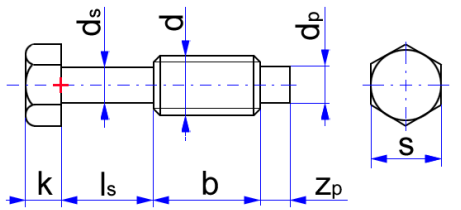
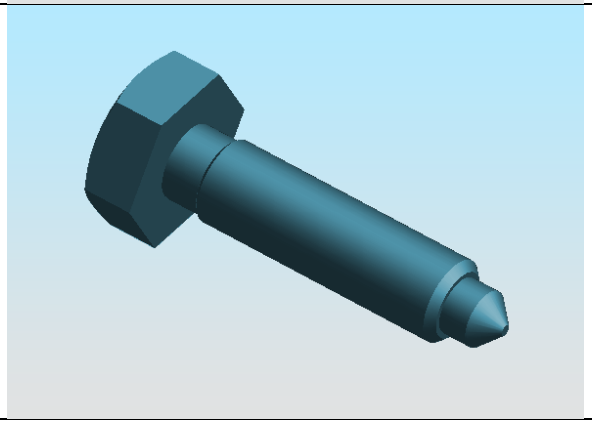
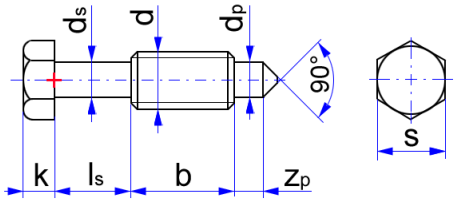
3.2.11	Double-direction angular contact thrust ball				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 10 (by MDESIGN)
3.2.12	Single-row cylindrical roller				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 11 (by MDESIGN)
3.2.13	Double-row cylindrical roller				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 12 (by MDESIGN)
3.2.14	Tapered roller				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 13 (by MDESIGN)

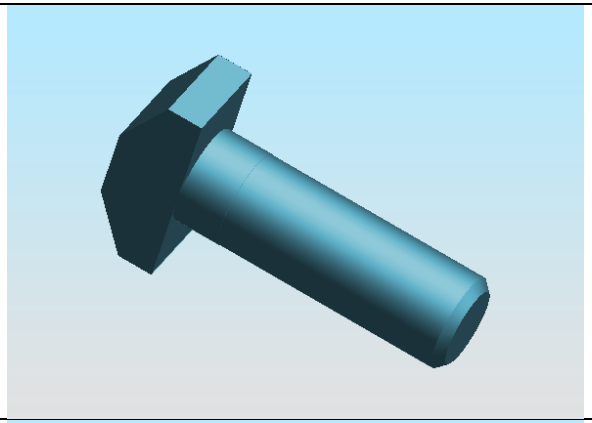
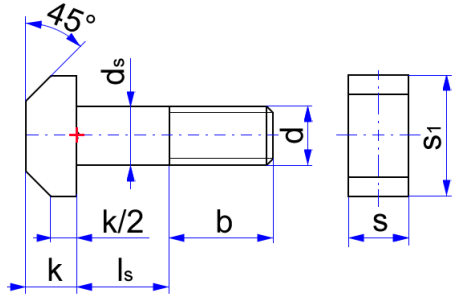
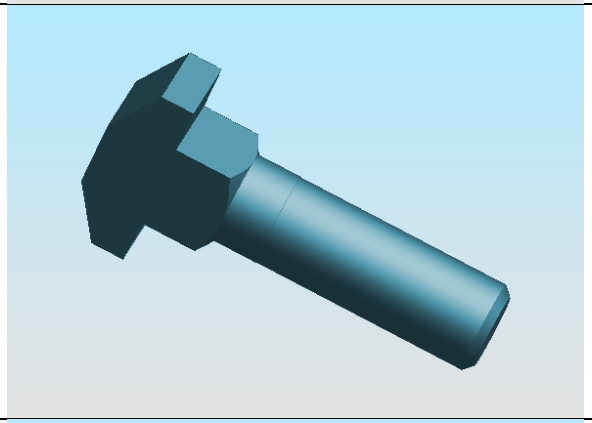
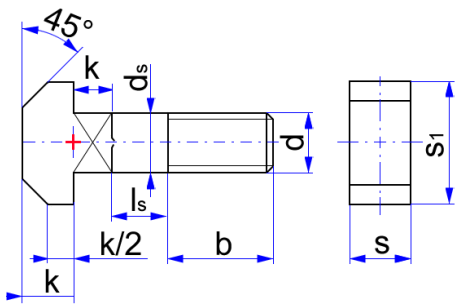
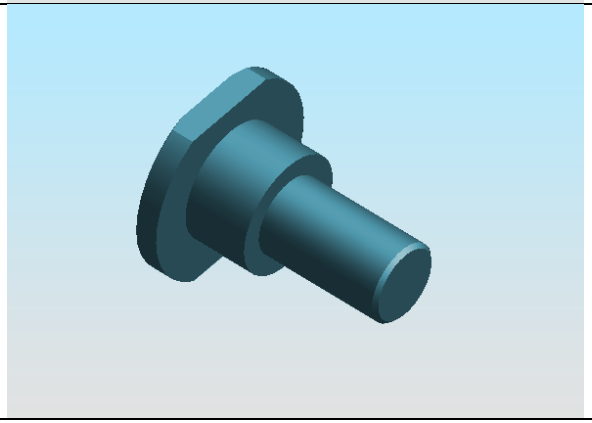
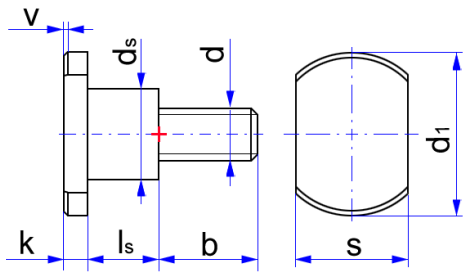
3.2.15	Barrel roller				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 14 (by MDESIGN)
3.2.16	Spherical roller				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 15 (by MDESIGN)
3.2.17	Single-direction cylindrical roller thrust				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 16 (by MDESIGN)
3.2.18	Double-direction cylindrical roller thrust				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 17 (by MDESIGN)

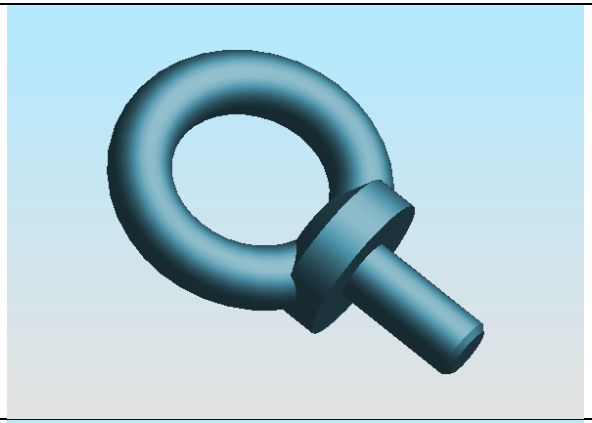
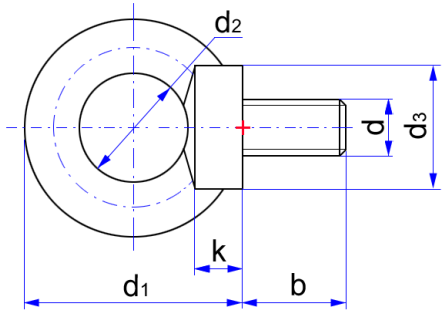
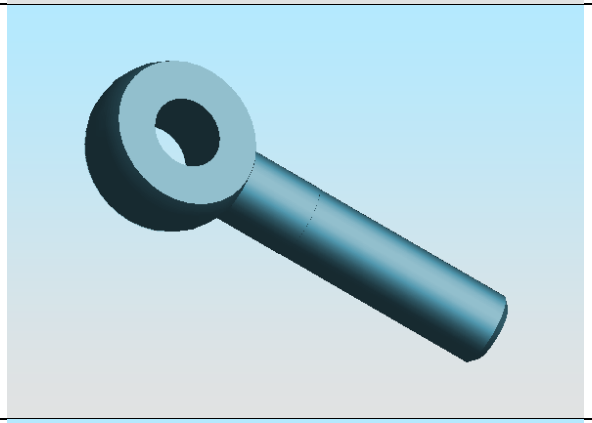
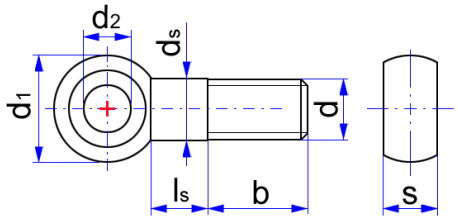
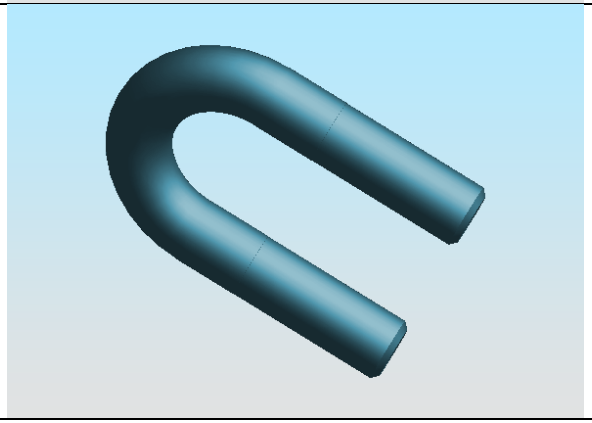
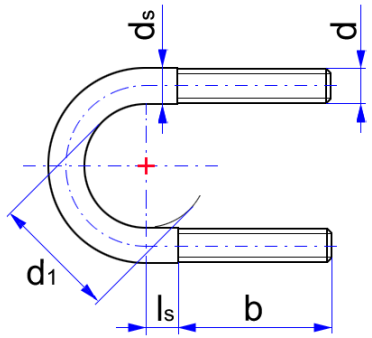
3.2.19	Spherical roller thrust				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 18 (by MDESIGN)
3.2.20	Needle roller				d_{out} - outer diameter d_{in} - inner diameter b - width	Type 19 (by MDESIGN)
4	Planet carrier					
4.1	Simplified presenta-tion				n - number of branches r - radius of branches l - length	d = 0.05 * r n >= 2

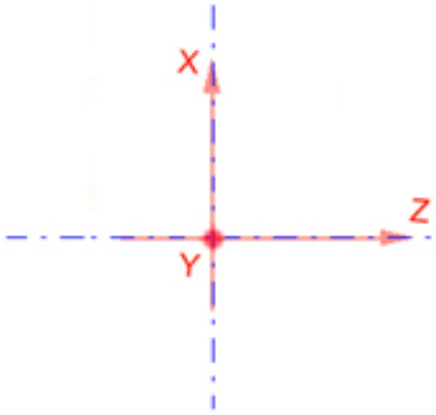
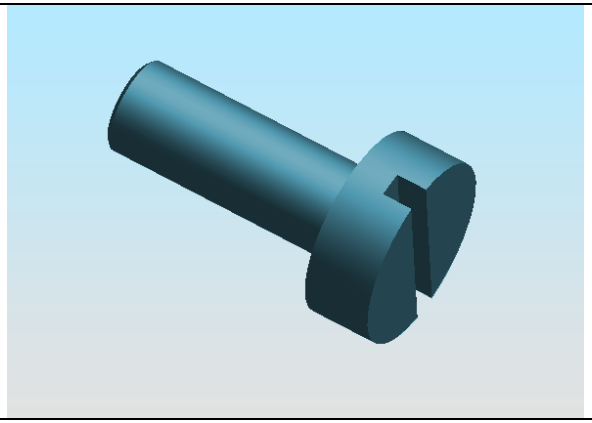
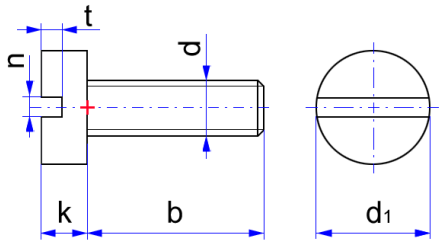
<table>
<tr><td>5</td><td colspan="5">Bolted joins</td></tr>
<tr><td>5.1</td><td>Bolts</td><td colspan="4">The thread in 3D-objects is presented conditionally in the cylinder form.
+ - the location of the local coordinate system of the object.</td></tr>
<tr><td></td><td></td><td></td><td>Coordinate system for bolts
 
</td><td></td><td>Y-axis has been directed „us“</td></tr>
<tr><td>5.1.1</td><td>Hexagon head bolt</td><td>  </td><td> 
</td><td>d - thread diameter
b - thread length
ds - dia. of unthreaded shank
ls – length of unthreaded shank
s - width across flats
k - height of head
dw - dia. of washer face
c - depth of washer face</td><td>k > c >= 0
ls >= 0
ds < s, if ls > 0
d < s, if ls = 0
ds <= dw <= s, if ls > 0
d <= dw <= s, if ls = 0
head fase = 30°x s
thread fase = (d - d*0.8)x 45°</td></tr>
</table>

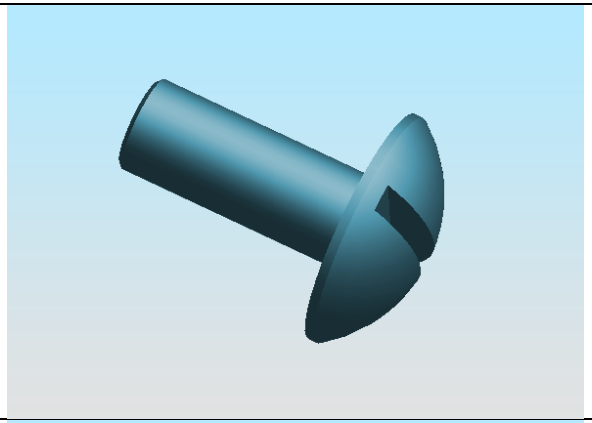
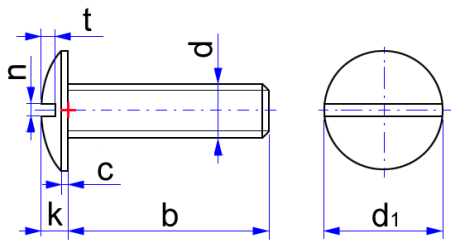
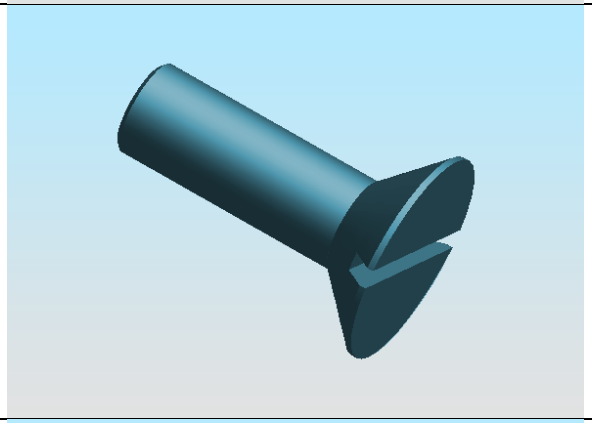
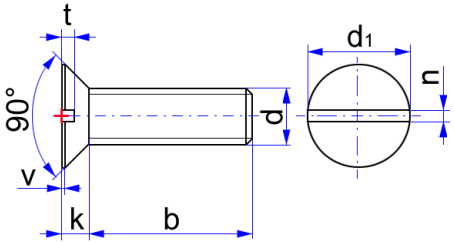
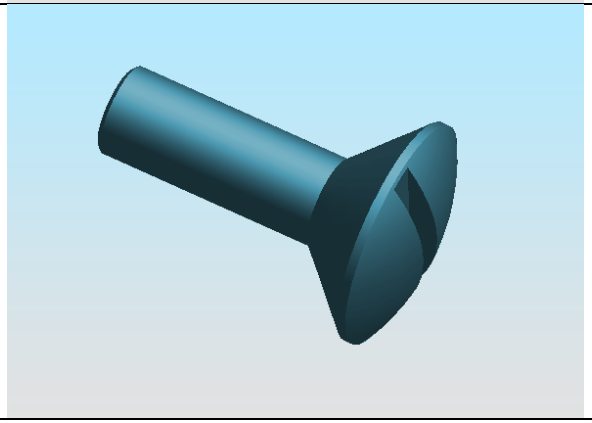
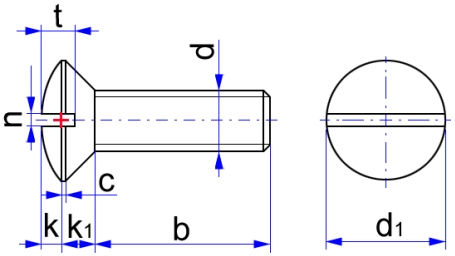
5.1.2	Hexagonal socket head cap screw			d - thread diameter b - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank d1 - diameter of head k - height of head s - key size of socket t – depth of socket	ls >= 0 d1 > ds, if ls > 0 d1 > d, if ls = 0 s < 0.87 * d1 - 0.04 * d1 t < k head fase = v x 45° v = 0.02 * d1
5.1.3	Hexagon socket countersunk head screw			d - thread diameter b - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank d1 - diameter of head s - key size of socket t – depth of socket	ls >= 0 d1 > ds, if ls > 0 d1 > d, if ls = 0 if ls > 0, k = f(d1, ds) if ls = 0, k = f(d1, d) t = f(d1) v = 0.03 * d1 s < 0.87 * d1
5.1.4	Hexagon flange bolt			d - thread diameter b - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank d1 - diameter of head s - width across flats k - height of head	ls >= 0 d1 > ds, if ls > 0 d1 > d, if ls = 0 d1> 1.15 * s d1 < 3.5*k - v v = 0.03 * d1

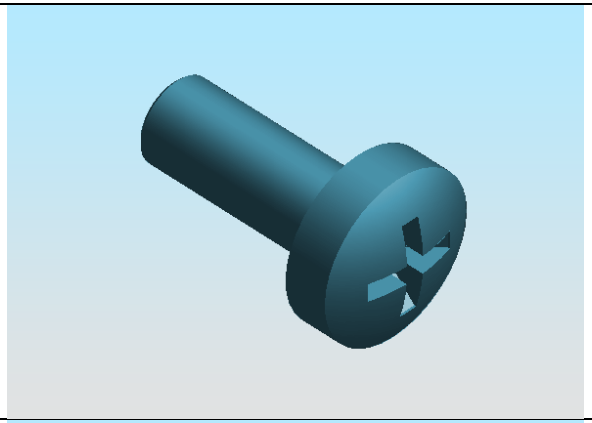
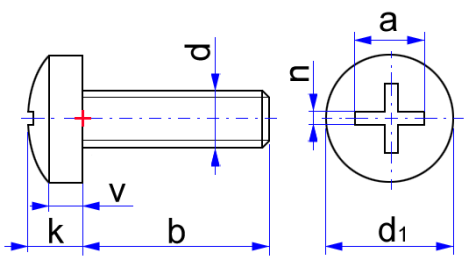
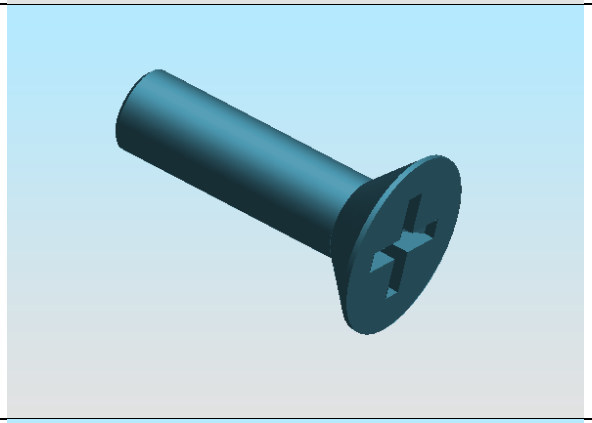
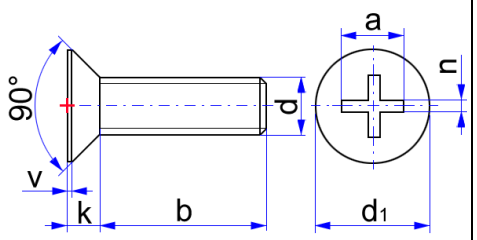
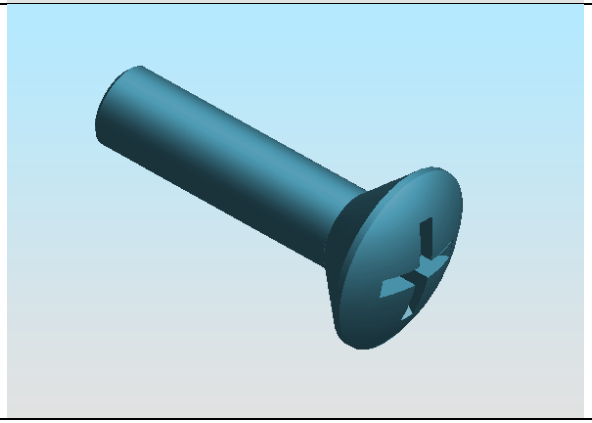
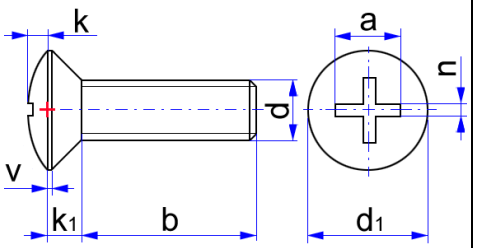
5.1.5	Carriage bolt			d - thread diameter b - thread Length ds - dia. of unthreaded shank ls – length of unthreaded shank d1 - diameter of head k - height of head v - square width across flats f - square dept	d1 > 1.41*v k < 0.5*d1 c = 0.04*d1 ls >= 0 v > ds, if ls > 0 v > d, if ls = 0
5.1.6	Hexagon set bolt with full dog point			d - thread diameter b - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank s - width across flats k - height of head dp - dog point diameter zp - dog point length	ls >= 0 ds < s, if ls > 0 d < s, if ls = 0 head fase = 30°x s thread fase = (d - d*0.8)x 45° dp < 0.7*d
5.1.7	Hexagon set bolt with flat cone point			d - thread diameter b - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank s - width across flats k - height of head dp - dog point diameter zp - dog point length	ls >= 0 ds < s, if ls > 0 d < s, if ls = 0 head fase = 30°x s thread fase = (d - d*0.8)x 45° dp < 0.7*d Height of truncated cone = 0,4*dp

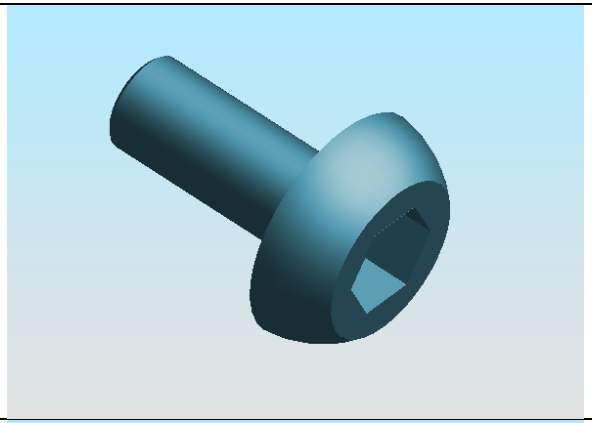
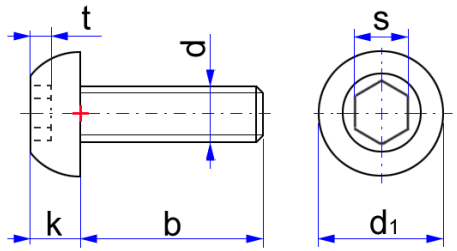
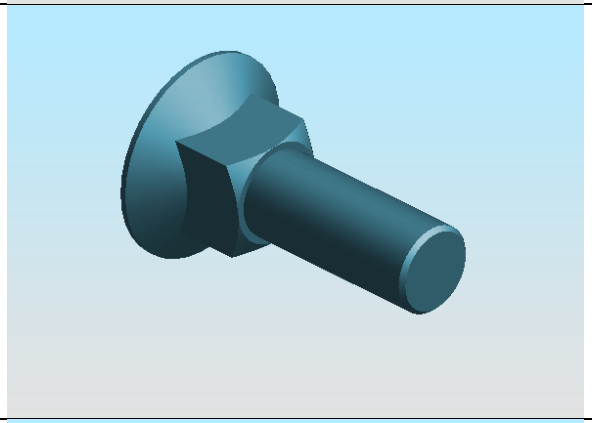
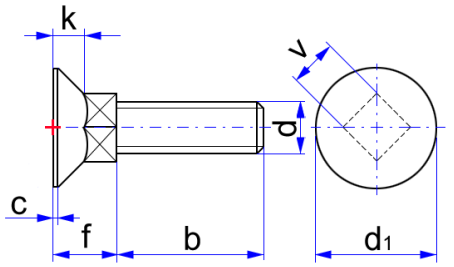
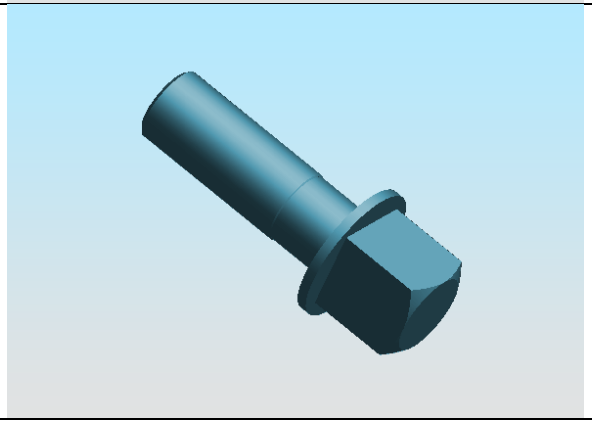
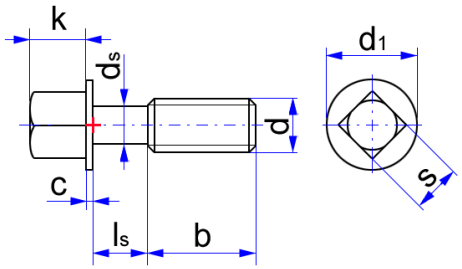
5.1.8	T-head bolt			d - thread diameter b - thread length ls – length of unthreaded shank s1 - width of head k - height of head	ls >= 0 ds = 1.02*d s1 > ds, if ls > 0 s1 > d, if ls = 0 s = 1.01*d k < s1 thread fase = (d - d*0.8)x 45°
5.1.9	T-head bolts with square neck			d - thread diameter b - thread length ls – length of unthreaded shank s1 - width of head k - height of head, height of neck	ls >= 0 ds = 1.02*d s1 > ds, if ls > 0 s1 > d, if ls = 0 s = 1.01*d s = width of neck k < s1 thread fase = (d - d*0.8)x 45°
5.1.10	Bolt with spigot and stud end			d - thread diameter b - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank d1 - diameter of head s - width across flats k - height of head	ls > 0 d1 > ds ds > d s > ds k > 2 * v 1 head fase = v x 30° v = 0.02 * d1

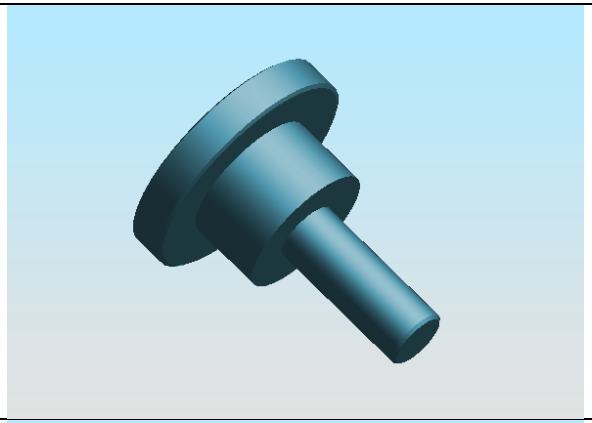
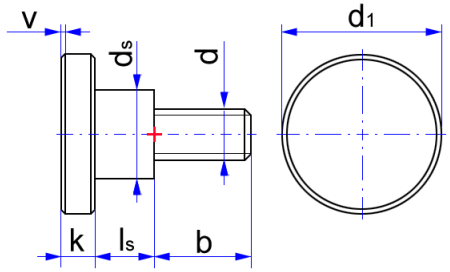
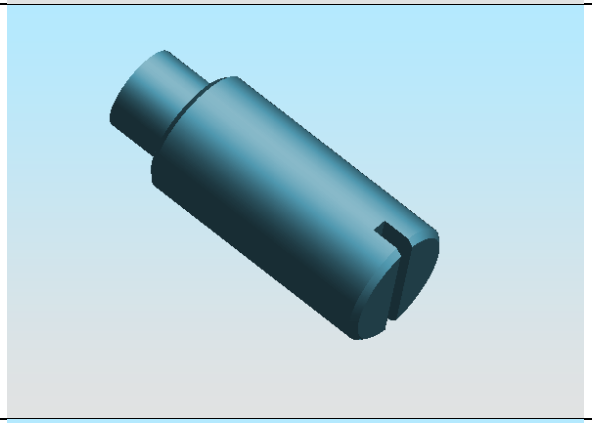
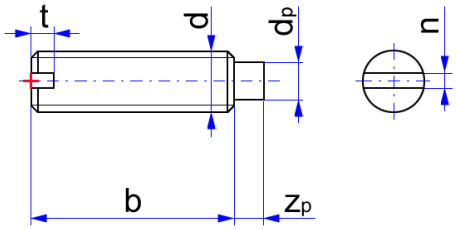
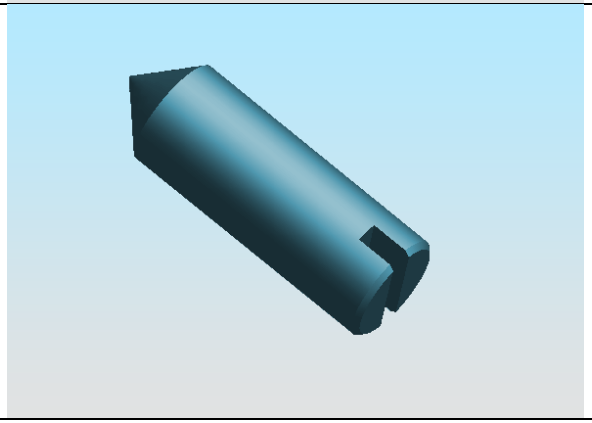
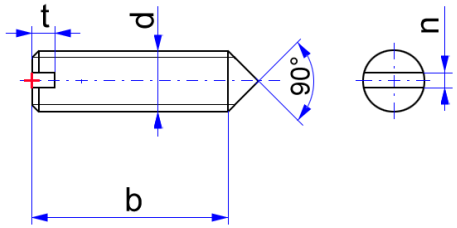
5.1.11	Ringbolt			d - thread diameter b - thread length d1 - outer diameter of ring (torus) d2 - inner diameter of ring (torus) d3 - diameter of head k - height of head	d1 > d2 d2 > d d3 >= d2 k <= (d1 - d2)/2 head cone = 120°
5.1.12	Bolt eye			d – thread diameter b – thread length ls – length of unthreaded shank d1 – diameter of head (sphere) d2 – diameter of bore s – width of head	ls >= 0 ds = 1.02*d d1 > s d1 > 1.5*d d1 > d2 thread fase = (d - d*0.8)x 45°
5.1.13	U-bolt			d - thread diameter b - thread length ls - length of unthreaded shank d1 - diameter of tube	ls >= 0 ds = 1.02*d thread fase = (d - d*0.8)x 45°

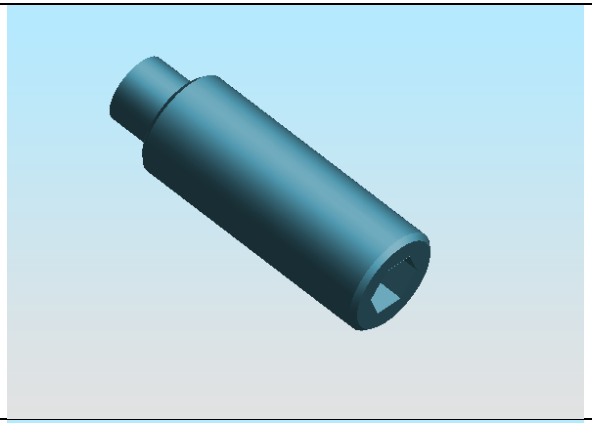
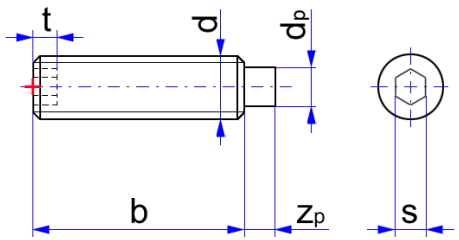
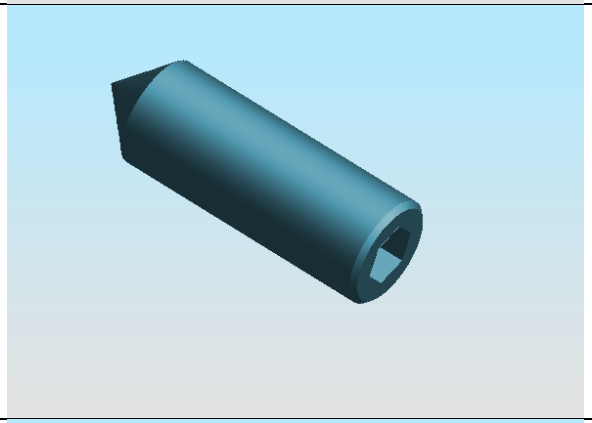
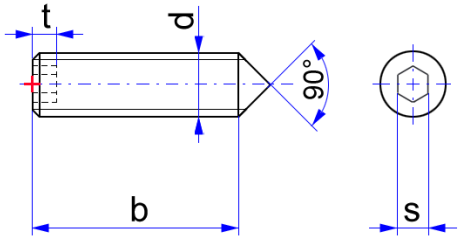
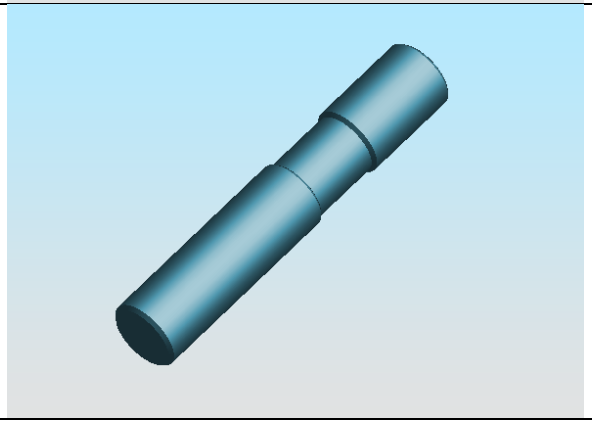
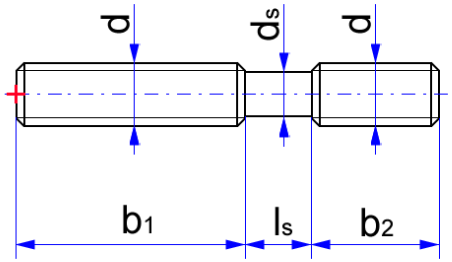
<table>
<tr><td>5.2</td><td>Screws</td><td colspan="4">The thread in 3D-objects is presented conditionally in the cylinder form.
+ - the location of the local coordinate system of the object.</td></tr>
<tr><td></td><td></td><td></td><td>Coordinate system for screws
 
</td><td></td><td>Y-axis has been directed „us“</td></tr>
<tr><td>5.2.1</td><td>Slotted cheese head screw</td><td>  </td><td> 
</td><td>d - thread diameter
b - thread length
d1 - diameter of head
k - height of head
n – width of slot
t - depth of slot</td><td>d1 > d
t < k
n < d</td></tr>
</table>

5.2.2	Slotted pan head screw			d - thread diameter b - thread length d1 - diameter of head k - height of head n – width of slot t - depth of slot	d1 > d k < 0.5*d1 t < k n < d c = 0.04 * d1
5.2.3	Slotted countersunk head screw			d - thread diameter b - thread length d1 - diameter of head n – width of slot t - depth of slot	d1 > d t < k n < d k = f(d1, d) v = 0.03 * d1
5.2.4	Slotted raised countersunk head screw			d - thread diameter b - thread length d1 - diameter of head k - height of head n – width of slot t - depth of slot	d1 > d k < 0.5*d1 t < (k + k1) n < d k1 = f(d1, d) c = 0.04 * d1 angle of head = 90°

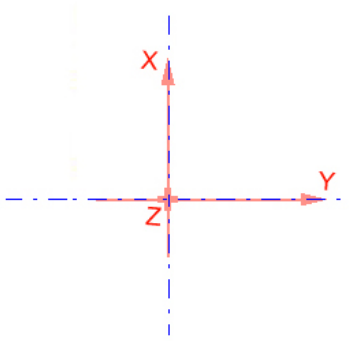
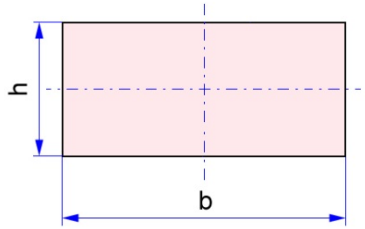
5.2.5	Cross recessed fillister head screw			d - thread diameter b - thread length d1 - diameter of head k - height of head a - width of recess n – thickness of recess	d1 > d k < 0.5*d1 k = 1.5 * v a < 0.6 * d1
5.2.6	Cross recessed countersunk head screw			d - thread diameter b - thread length d1 - diameter of head a - width of recess n – thickness of recess	d1 > d k = f(d1, d) v = 0.03 * d1 a < 0.6 * d1
5.2.7	Cross recessed raised countersunk head screw			d - thread diameter b - thread length d1 - diameter of head k - height of head a - width of recess n – thickness of recess	d1 > d k < 0.5*d1 k1 = f(d1, d) a < 0.6 * d1 angle of head = 90°

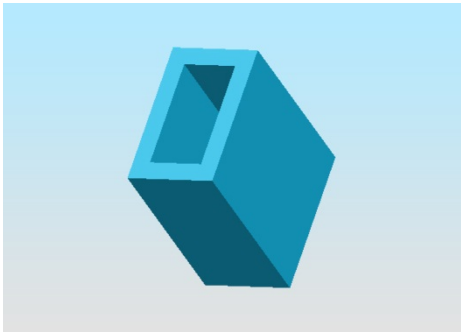
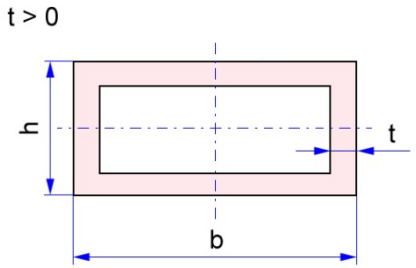
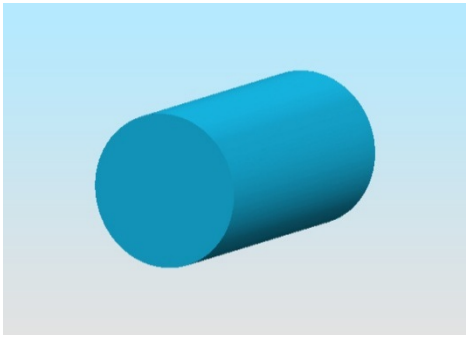
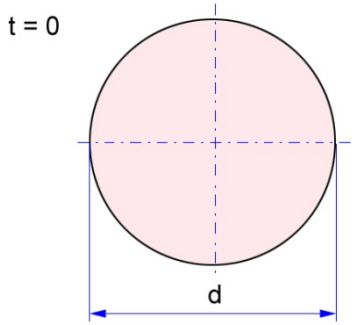
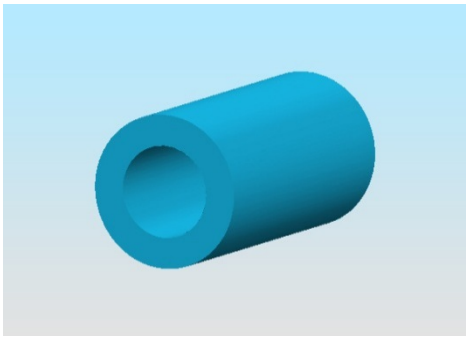
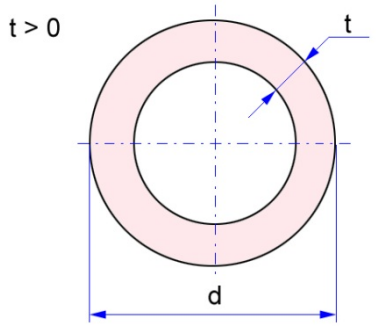
5.2.8	Hexagon socket button head screw			d - thread diameter b - thread length d1 - diameter of head s - key size of socket t – depth of socket	d1 > d k = 0.3*d1 s < 0.87 * dk dk=2* $\sqrt{((d1/2)^2 - k^2)}$ t < k
5.2.9	Countersunk head square neck bolt			d - thread diameter b - thread length d1 - diameter of head v - square width across flats f - head dept	d1 > 1.41*v v > d k = f(d1, v) c = 0.03*d1 angle of head = 120°
5.2.10	Square head bolt with collar			d - thread diameter b - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank s - width across flats k - height of head d1 - dia. of collar c - depth of collar	d1 > 1.41*s k > c >= 0 ls >= 0 head fase = 30°x s

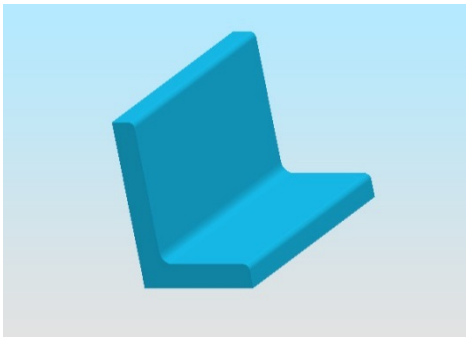
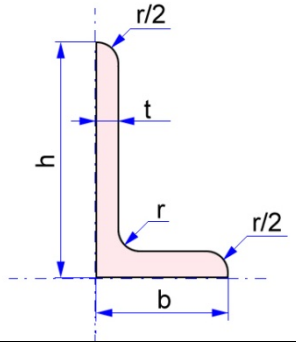
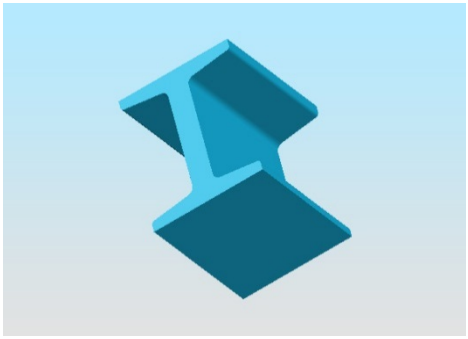
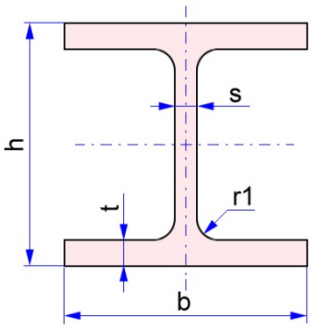
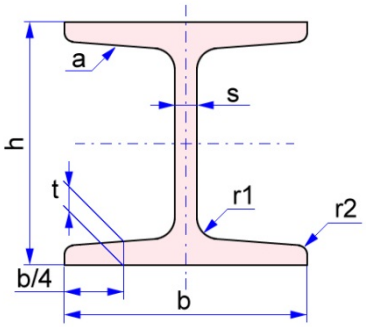
5.2.11	Thumb screw			d - thread diameter b - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank d1 - diameter of head k - height of head	ls > 0 d1 > ds ds > d k > 2 * v 2 head fase = v x 30° v = 0.01 * d1
5.2.12	Slotted set screw with dog point			d - thread diameter b - thread length dp - dog point diameter zp - dog point length n – width of slot t - depth of slot	n < d t < b dp < 0.7*d Zp >= 0
5.2.13	Slotted set screw with cone point			d - thread diameter b - thread length n – width of slot t - depth of slot	n < d t < b

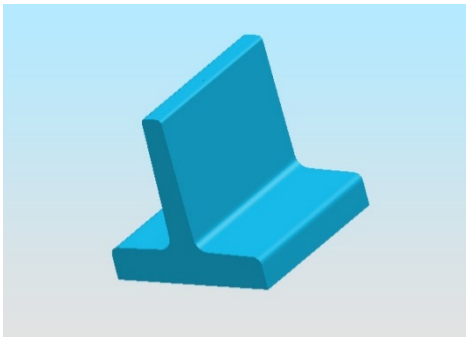
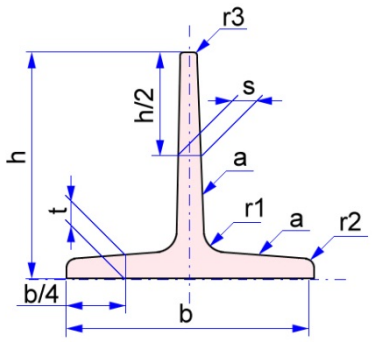
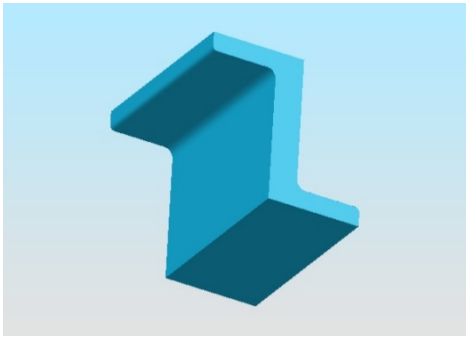
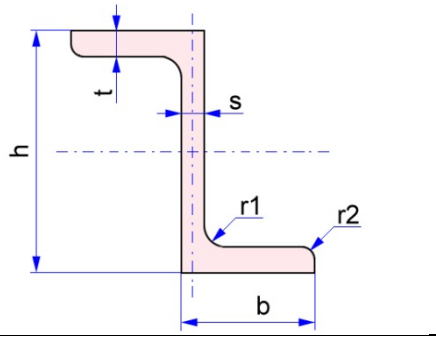
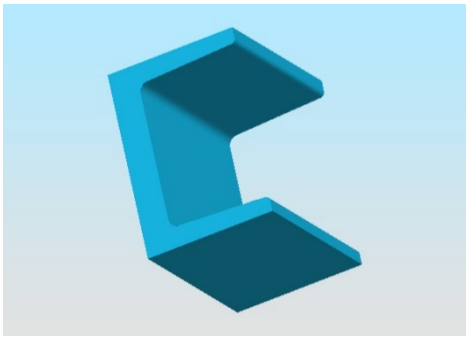
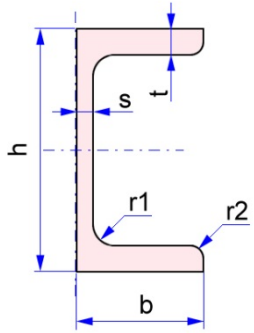
5.2.14	Hex socket set screw with dog point			d - thread diameter b - thread length dp - dog point diameter zp - dog point length s - key size of socket t – depth of socket	b > t dp < 0.7*d Zp >= 0
5.2.15	Hex socket set screw with cone point			d - thread diameter b - thread length s - key size of socket t – depth of socket	b > t
5.2.16	Double end stud			d - thread diameter b1, b2 - thread length ds - dia. of unthreaded shank ls – length of unthreaded shank	ls > 0

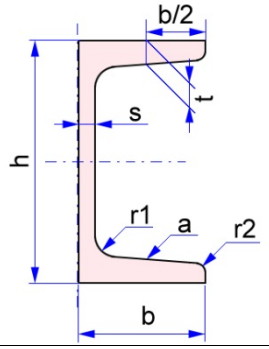
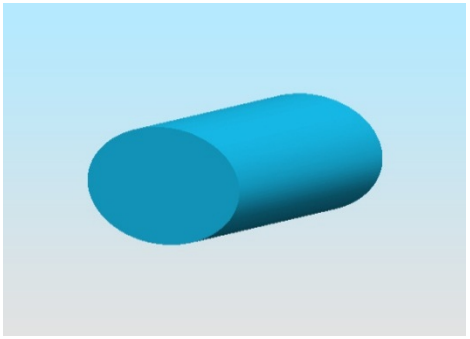
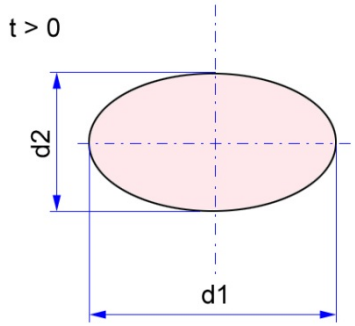
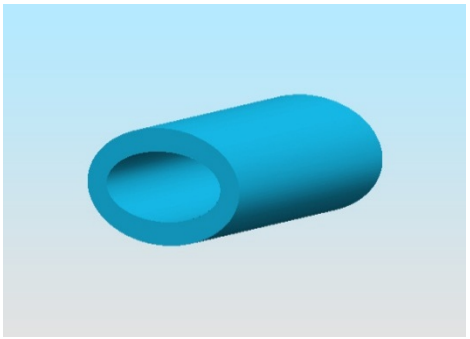
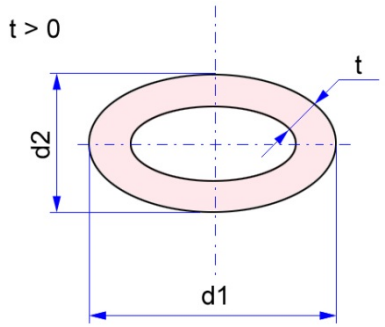
1.4 Building Construction

N	Object Name	3D View	2D View	Parameter Values	Comments
1	Beams		Coordinate system for beams 	l – length of beams (General properties)	Z achsis has been directed „from us“
1.1	Rectangular				
1.1.1	Rectangular solid		t = 0 	h - height b - width t - depth	t = 0

1.1.2	Rectangular hollow			h - height b - width t - depth	t > 0
1.2	Round				
1.2.1	Round solid			d – diameter t - depth	t = 0
1.2.2	Round hollow			d – diameter t - depth	t > 0

<table>
<tr><td>1.3</td><td>L-profile</td><td>  </td><td> 
</td><td>h- height
b - width
t - depth
r - radius of rounding</td><td></td></tr>
<tr><td>1.4</td><td>I-beams</td><td colspan="4"></td></tr>
<tr><td>1.4.1</td><td>Without taper</td><td rowspan="2">  </td><td> 
</td><td rowspan="2">h - height
b - width
t - depth
s - depth
r1 - radius of rounding 1
r2 - radius of rounding 2
a - taper</td><td>Is need taper = FALSE</td></tr>
<tr><td>1.4.2</td><td>With taper</td><td> 
</td><td>Is need taper = TRUE
a = 14%</td></tr>
</table>

1.5	T-steel			h - height b - width t - depth s - depth r1 - radius of rounding 1 r2 - radius of rounding 2 r3 - radius of rounding 3 a – taper	Is need taper = TRUE a = 2%
1.6	Z-steel			h - height b - width t – depth s - depth r1 - radius of rounding 1 r2 - radius of rounding 2	
1.7	U-profile				
1.7.1	Without taper			h - height b - width t - depth s - depth r1 - radius of rounding 1 r2 - radius of rounding 2 a - taper	Is need taper = FALSE

1.7.2	With taper				Is need taper = TRUE a = 8% for h <=300mm a = 5% for h > 300mm
1.8	Ellipse				
1.8.1	Ellipse solid			d1 - diameter x d2 - diameter y t - depth	t = 0
1.8.2	Ellipse hollow			d1 - diameter x d2 - diameter y t - depth	t > 0
2	…….				

APPENDIX 2. GNU LESSER GENERAL PUBLIC LICENSE

Open CASCADE Technology version 6.7.0 and later are governed by GNU Lesser General Public License (LGPL) version 2.1 with additional exception.

Note: A specific license is applied to Open CASCADE Technology version 6.6.0 and earlier.

GNU Lesser General Public License

Version 2.1, February 1999

[This is the first released version of the Lesser GPL. It also counts as the successor of the GNU Library Public License, version 2, hence the version number 2.1.]

Preambule

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public Licenses are intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users.

This license, the Lesser General Public License, applies to some specially designated software packages–typically libraries–of the Free Software Foundation and other authors who decide to use it. You can use it too, but we suggest you first think carefully about whether this license or the ordinary General Public License is the better strategy to use in any particular case, based on the explanations below.

When we speak of free software, we are referring to freedom of use, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish); that you receive source code or can get it if you want it; that you can change the software and use pieces of it in new free programs; and that you are informed that you can do these things.

To protect your rights, we need to make restrictions that forbid distributors to deny you these rights or to ask you to surrender these rights. These restrictions translate to certain responsibilities for you if you distribute copies of the library or if you modify it.

For example, if you distribute copies of the library, whether gratis or for a fee, you must give the recipients all the rights that we gave you. You must make sure that they, too, receive or can get the source code. If you link other code with the library, you must provide complete object files to the recipients, so that they can relink them with the library after making changes to the library and recompiling it. And you must show them these terms so they know their rights.

We protect your rights with a two-step method: (1) we copyright the library, and (2) we offer you this license, which gives you legal permission to copy, distribute and/or modify the library. To protect each distributor, we want to make it very clear that there is no warranty for the free library. Also, if the library is modified by someone else and passed on, the recipients should know that what they have is not the original version, so that the original author's reputation will not be affected by problems that might be introduced by others.

Finally, software patents pose a constant threat to the existence of any free program. We wish to make sure that a company cannot effectively restrict the users of a free program by obtaining a restrictive license from a patent holder. Therefore, we insist that any patent license obtained for a version of the library must be consistent with the full freedom of use specified in this license.

Most GNU software, including some libraries, is covered by the ordinary GNU General Public License. This license, the GNU Lesser General Public License, applies to certain designated libraries, and is quite different from the ordinary General Public License. We use this license for certain libraries in order to permit linking those libraries into non-free programs.

When a program is linked with a library, whether statically or using a shared library, the combination of the two is legally speaking a combined work, a derivative of the original library. The ordinary General Public License therefore permits such linking only if the entire combination fits its criteria of freedom. The Lesser General Public License permits more lax criteria for linking other code with the library.

We call this license the "Lesser" General Public License because it does Less to protect the user's freedom than the ordinary General Public License. It also provides other free software developers Less of an advantage over competing non-free programs. These disadvantages are the reason we use the ordinary General Public License for many libraries. However, the Lesser license provides advantages in certain special circumstances.

For example, on rare occasions, there may be a special need to encourage the widest possible use of a certain library, so that it becomes a de-facto standard. To achieve this, non-free programs must be allowed to use the library. A more frequent case is that a free library does the same job as widely used non-free libraries. In this case, there is little to gain by limiting the free library to free software only, so we use the Lesser General Public License.

In other cases, permission to use a particular library in non-free programs enables a greater number of people to use a large body of free software. For example, permission to use the GNU C Library in non-free programs enables many more people to use the whole GNU operating system, as well as its variant, the GNU/Linux operating system.

Although the Lesser General Public License is Less protective of the users' freedom, it does ensure that the user of a program that is linked with the Library has the freedom and the wherewithal to run that program using a modified version of the Library.

The precise terms and conditions for copying, distribution and modification follow. Pay close attention to the difference between a "work based on the library" and a "work that uses the library". The former contains code derived from the library, whereas the latter must be combined with the library in order to run.

Terms and Conditions for Copying, Distribution and Modification

0. This License Agreement applies to any software library or other program which contains a notice placed by the copyright holder or other authorized party saying it may be distributed under the terms of this Lesser General Public License (also called "this License"). Each licensee is addressed as "you".

A "library" means a collection of software functions and/or data prepared so as to be conveniently linked with application programs (which use some of those functions and data) to form executables.

The "Library", below, refers to any such software library or work which has been distributed under these terms. A "work based on the Library" means either the Library or any derivative work under copyright law: that is to say, a work containing the Library or a portion of it, either verbatim or with modifications and/or translated straightforwardly into another language. (Hereinafter, translation is included without limitation in the term "modification".)

"Source code" for a work means the preferred form of the work for making modifications to it. For a library, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the library.

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running a program using the Library is not restricted, and output from such a program is covered only if its contents constitute a work based on the Library (independent of the use of the Library in a tool for writing it). Whether that is true depends on what the Library does and what the program that uses the Library does.

1. You may copy and distribute verbatim copies of the Library's complete source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and distribute a copy of this License along with the Library.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Library or any portion of it, thus forming a work based on the Library, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) The modified work must itself be a software library.

b) You must cause the files modified to carry prominent notices stating that you changed the files and the date of any change.

c) You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.

d) If a facility in the modified Library refers to a function or a table of data to be supplied by an application program that uses the facility, other than as an argument passed when the facility is invoked, then you must make a good faith effort to ensure that, in the event an application does not supply such function or table, the facility still operates, and performs whatever part of its purpose remains meaningful.

(For example, a function in a library to compute square roots has a purpose that is entirely well-defined independent of the application. Therefore, Subsection 2d requires that any application-supplied function or table used by this function must be optional: if the application does not supply it, the square root function must still compute square roots.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Library, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Library, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Library.

In addition, mere aggregation of another work not based on the Library with the Library (or with a work based on the Library) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may opt to apply the terms of the ordinary GNU General Public License instead of this License to a given copy of the Library. To do this, you must alter all the notices that refer to this License, so that they refer to the ordinary GNU General Public License, version 2, instead of to this License. (If a newer version than version 2 of the ordinary GNU General Public License has appeared, then you can specify that version instead if you wish.) Do not make any other change in these notices.

Once this change is made in a given copy, it is irreversible for that copy, so the ordinary GNU General Public License applies to all subsequent copies and derivative works made from that copy.

This option is useful when you wish to copy part of the code of the Library into a program that is not a library.

4. You may copy and distribute the Library (or a portion or derivative of it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange.

If distribution of object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place satisfies the requirement to distribute the source code, even though third parties are not compelled to copy the source along with the object code.

5. A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it, is called a "work that uses the Library". Such a work, in isolation, is not a derivative work of the Library, and therefore falls outside the scope of this License.

However, linking a "work that uses the Library" with the Library creates an executable that is a derivative of the Library (because it contains portions of the Library), rather than a "work that uses the library". The executable is therefore covered by this License. Section 6 states terms for distribution of such executables.

When a "work that uses the Library" uses material from a header file that is part of the Library, the object code for the work may be a derivative work of the Library even though the source code is not. Whether this is true is especially significant if the work can be linked without the Library, or if the work is itself a library. The threshold for this to be true is not precisely defined by law.

If such an object file uses only numerical parameters, data structure layouts and accessors, and small macros and small inline functions (ten lines or less in length), then the use of the object file is unrestricted, regardless of whether it is legally a derivative work. (Executables containing this object code plus portions of the Library will still fall under Section 6.)

Otherwise, if the work is a derivative of the Library, you may distribute the object code for the work under the terms of Section 6. Any executables containing that work also fall under Section 6, whether or not they are linked directly with the Library itself.

6. As an exception to the Sections above, you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice, provided that the terms permit modification of the work for the customer's own use and reverse engineering for debugging such modifications.

You must give prominent notice with each copy of the work that the Library is used in it and that the Library and its use are covered by this License. You must supply a copy of this License. If the work during execution displays copyright notices, you must include the copyright notice for the Library among them, as well as a reference directing the user to the copy of this License. Also, you must do one of these things:

a) Accompany the work with the complete corresponding machine-readable source code for the Library including whatever changes were used in the work (which must be distributed under Sections 1 and 2 above); and, if the work is an executable linked with the Library, with the complete machine-readable "work that uses the Library", as object code and/or source code, so that the user can modify the Library and then relink to produce a modified executable containing the modified Library. (It is understood that the user who changes the contents of definitions files

in the Library will not necessarily be able to recompile the application to use the modified definitions.)

b) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (1) uses at run time a copy of the library already present on the user's computer system, rather than copying library functions into the executable, and (2) will operate properly with a modified version of the library, if the user installs one, as long as the modified version is interface-compatible with the version that the work was made with.

c) Accompany the work with a written offer, valid for at least three years, to give the same user the materials specified in Subsection 6a, above, for a charge no more than the cost of performing this distribution.

d) If distribution of the work is made by offering access to copy from a designated place, offer equivalent access to copy the above specified materials from the same place.

e) Verify that the user has already received a copy of these materials or that you have already sent this user a copy.

For an executable, the required form of the "work that uses the Library" must include any data and utility programs needed for reproducing the executable from it. However, as a special exception, the materials to be distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

It may happen that this requirement contradicts the license restrictions of other proprietary libraries that do not normally accompany the operating system. Such a contradiction means you cannot use both them and the Library together in an executable that you distribute.

7. You may place library facilities that are a work based on the Library side-by-side in a single library together with other library facilities not covered by this License, and distribute such a combined library, provided that the separate distribution of the work based on the Library and of the other library facilities is otherwise permitted, and provided that you do these two things:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities. This must be distributed under the terms of the Sections above.

b) Give prominent notice with the combined library of the fact that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

8. You may not copy, modify, sublicense, link with, or distribute the Library except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, link with, or distribute the Library is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

9. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Library or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Library (or any work based on the Library), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Library or works based on it.

10. Each time you redistribute the Library (or any work based on the Library), the recipient automatically receives a license from the original licensor to copy, distribute, link with or modify the Library subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties with this License.

11. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Library at all. For example, if a patent license would not permit royalty-free redistribution of the Library by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Library.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply, and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

12. If the distribution and/or use of the Library is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Library under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

13. The Free Software Foundation may publish revised and/or new versions of the Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Library does not specify a license version number, you may choose any version ever published by the Free Software Foundation.

14. If you wish to incorporate parts of the Library into other free programs whose distribution conditions are incompatible with these, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

15. BECAUSE THE LIBRARY IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE LIBRARY, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE LIBRARY "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE LIBRARY IS WITH

YOU. SHOULD THE LIBRARY PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

16. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE LIBRARY AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE LIBRARY (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE LIBRARY TO OPERATE WITH ANY OTHER SOFTWARE), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

Open CASCADE Exception (version 1.0) to GNU LGPL version 2.1.

The object code (i.e. not a source) form of a "work that uses the Library" can incorporate material from a header file that is part of the Library. As a special exception to the GNU Lesser General Public License version 2.1, you may distribute such object code incorporating material from header files provided with the Open CASCADE Technology libraries (including code of CDL generic classes) under terms of your choice, provided that you give prominent notice in supporting documentation to this code that it makes use of or is based on facilities provided by the Open CASCADE Technology software.