ISCAD User Guide

hk

March 4, 2017

Contents

1 Introduction

Insight CAD is a script-based tool for creating three-dimensional geometry models. All geometric operations are based on the OpenCASCADE geometry kernel. It uses the boundary representation approach (BREP) for treating the geometry. Thus it is possible to import and export the common exchange formats IGES and STEP.

Although the primary intention of Insight CAD is creation of fully parameterized CAD models for systematic numerical simulations, it can also be used for mechanical design purposes. It therefore provides the possibility to export projections and sections of the created models in DXF format for use in drawings. Additionally, there is support for a library of parametric standard parts.

2 Basic Concept

The basic entity in Insight CAD is a “model”. A model is described by a script and stored in an ASCII script file (extension “.iscad”). Inside a model script, symbols are defined, which can represent the following data types:

- Scalars
- Vectors
- Datum objects (axes, planes)
- 3D geometry objects (features)
- Selections of vertices, edges, faces or solids of 3D geometry objects

There is no explicit type declaration: the data type of each symbol is deduced from the defining expression.

Beyond their geometry representation, geometry feature objects are containers for scalar, vector, datum and feature objects. These symbols can be accessed in the model script but are read-only.

It is possible to load another model into the current one by the “loadmodel” command (see section 3.1.4). In this case, the loaded model represents a subassembly, i.e. a compound of features. Since not all defined geometry objects in the submodel may represent assembly components, marking of components is supported by the feature definition syntax and needs to be utilized properly in the definition script of the submodel. Also, when loading models (subassemblies), parameters can be passed to the submodel. These can be scalars, vectors, datums or features.

In a model script, after the definition of the aforementioned symbols, an optional section with postprocessing actions can follow. These can be e.g. file export, drawing export or others.

3 Syntax

The general model script layout begins with a mandatory symbol defining section, followed by an optional postprocessing section, started by “@post”:

```
<identifier> [ = | ?= | : ] <expression>;
...

@post

<postprocessing action>;
...
```

Comments are lines starting with “#” or text regions enclosed by “/*” and “*/” (C-style).

3.1 Symbol Definition

3.1.1 Scalars

Example:

```
D ?= 123.5;
L = 2.*D;
```

The “?=” operator assigns a default value. This needs to be used for symbols which are intended to be used as parameters in the “loadmodel” feature command. Symbols defined by the equal sign operator (“=”) cannot be overridden during the loadmodel command.

Supported operations are listed in table 1.

Insight CAD script	Description
+ - * /	basic algebra
mag(<vector>)	magnitude of vector
sqrt(<scalar>)	square root
sin(<scalar>)	$\sin(x)$
cos(<scalar>)	$\cos(x)$
tan(<scalar>)	$\tan(x)$
asin(<scalar>)	$\arcsin(x)$
acos(<scalar>)	$\arccos(x)$
ceil(<scalar>)	smallest following integer
floor(<scalar>)	largest previous integer
round(<scalar>)	round to next integer
pow(<scalar:a>, <scalar:n>)	a^n
atan2(<scalar:y>, <scalar:x>)	$\arctan\frac{y}{x}$
atan(<scalar>)	$\arctan(x)$
volume(<feature>)	volume of feature
cumedgelen(<feature>)	sum of all edges lengths
<vector>.x	x component of vector
<vector>.y	y component of vector
<vector>.z	z component of vector
<feature> $<identifier>	scalar property of feature
<vector> & <vector>	Scalar product

Table 1: Scalar Operations

There are scalars predefined in each model. They are listed in table 2.

Insight CAD script	Description
M_PI	π
deg	$180/\pi$

Table 2: Predefined Scalars

3.1.2 Vectors

Example:

```
v ?= 5*EX + 3*EY + [0,0,1];
ev = v/mag(v);
```

The "?=" operator assigns a default value. This needs to be used for symbols which are intended to be used as parameters in the "loadmodel" feature command. Symbols defined by the equal sign operator ("=") cannot be overridden during the loadmodel command.

Supported operations are listed in table 3.

There are vectors predefined in each model. They are listed in table 4.

3.1.3 Datums

Some simple examples are given below.

Insight CAD script	Description
+ -	basic algebra
[<scalar:x>, <scalar:y>, <scalar:z>]	vector from components
<feature> @<identifier>	vector property of feature
<scalar> *<vector>	scaled vector
<vector>/<scalar>	scaled vector
<vector:a> ^<vector:b>	$\vec{a} \times \vec{b}$
bbmin(<feature>)	minimum corner of feature bounding box
bbmax(<feature>)	maximum corner of feature bounding box
cog(<feature>)	center of gravity coordinates of feature
refpt(<datum>)	reference point of datum (base point of axis or plane)
refdir(<datum>)	reference direction of datum (direction of axis or normal of plane)

Table 3: Vector Operations

Insight CAD script	Description
EX	$\vec{e}_x = (1\ 0\ 0)^T$
EY	$\vec{e}_y = (0\ 1\ 0)^T$
EZ	$\vec{e}_z = (0\ 0\ 1)^T$
O	$\vec{O} = (0\ 0\ 0)^T$

Table 4: Predefined Vectors

```
myaxis   ?= RefAxis(O, EX+EY); # diagonal axis
myplane  = Plane(5*EX, EY);  # offset plane
myplane2 = XZ << 5*EX; # same offset plane
axis2    = xsec_plpl(XY, myplane); # axis at intersection of XY-Plane and offset plane
```

The "?=" operator assigns a default value. This needs to be used for symbols which are intended to be used as parameters in the "loadmodel" feature command. Symbols defined by the equal sign operator ("=") cannot be overridden during the loadmodel command.

Supported operations are listed in table 5.

Insight CAD script	Description
<feature> %<identifier>	Access datum inside another feature.
<datum> << <vector:$\vec{\Delta}$>	Copy of datum, translated by $\vec{\Delta}$
Plane(<vector:$\vec{p}_0$>, <vector:$\vec{n}$>)	Datum plane with origin $\vec{p}_0$ and normal $\vec{n}$.
SPlane(<vector:$\vec{p}_0$>, <vector:$\vec{n}$>, <vector:$\vec{e}_{up}$>)	Datum plane with origin $\vec{p}_0$ and normal $\vec{n}$. Additionally, the y-direction of the plane CS is aligned with $\vec{e}_{up}$.
RefAxis(<vector:$\vec{p}_0$>, <vector:$\vec{e}_x$>)	Axis with origin $\vec{p}_0$ and direction $\vec{e}_x$.
xsec_axpl(<datum:ax>, <datum:pl>)	Datum point at intersection between axis and plane
xsec_plpl(<datum:p_1>, <datum:p_2>)	Datum axis at intersection between plane p_1 and p_2
xsec_ppp(<datum:p_1>, <datum:p_2>, <datum:p_3>)	Datum point at intersection between three planes

Table 5: Datum Operations

There are datums predefined in each model. They are listed in table 6.

Insight CAD script	Description
XY	X-Y-Plane
XZ	X-Z-Plane
YZ	Y-Z-Plane

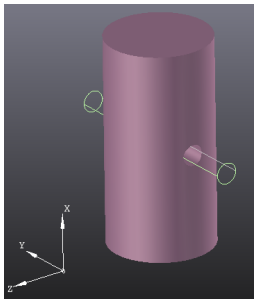
Table 6: Predefined Datums

3.1.4 Features

A very simple example is given below. It consists of two primitive features (cylinder) and a boolean operation (subtraction).

```
tool = Cylinder(-10*EY, 10*EY, 2);

pierced_cylinder:
 Cylinder(0, 20*EX, 10, centered)
 -
 tool
;
```



Geometry symbols can be defined by a "=" or a ":" operator. The difference comes from a possible use of the model as a subassembly later on. Since usually not all defined features in a model are assembly components but some are only intermediate modeling steps, there are these two syntaxes for defining a feature with a subtle difference: "<identifier> = <expression>;" defines an intermediate feature while "<identifier>: <expression>;" does the same geometry operation but marks the result as being an assembly component. In the above example, only the feature "pierced_cylinder" is marked as a component. Thus if the above example would be loaded as a subassembly, only the "pierced_cylinder" will be shown and included in e.g. mass calculations.

Insight CAD script	Description
<feature:a> - <feature:b>	Boolean subtract of feature b from feature a
<feature:a> | <feature:b>	Boolean unite of feature a and feature b
<feature:a> & <feature:b>	Boolean intersection of feature a and feature b
<feature> << <vector:$\vec{\Delta}$>	Copy of feature, translated by $\vec{\Delta}$
<feature> * <scalar:s>	Copy of feature, scaled by s (relative to global origin $\vec{O}$)
<feature>.<identifier:subfeatname>	Access of subfeature

Table 7: 3D Geometry Feature Operations

4 Feature Commands

In this section, an incomplete subset of the avilable feature commands is described in detail. For a complete list, please refer to the online documentation in the iscad editor (press Ctrl+F).

4.1 Transformation

Transform(<feature:f>, <vector:$\vec{\Delta}$>, <vector:$\vec{\Phi}$>) Transformation of feature f. Translation by $\vec{\Delta}$ and rotation around axis $\vec{\Phi}$ (magnitude of $\vec{\Phi}$ represents rotation angle).

Place(<**feature:f**>**,** <**vector:**$\vec{p}_0$>**,** <**vector:**$\vec{e}_x$>**,** <**vector:**$\vec{e}_z$>**)** Places the feature f in a new coordinate system. The new origin is at point $\vec{p}_0$, the new x-axis along vector $\vec{e}_x$ and the new z-direction is $\vec{e}_z$.

4.2 Import

import(<**path**>**)** Imports solid geometry from a file. The format is recognized from the filename extension. Supported formats are IGS, STP, BREP.

Sketch(<**datum:pl**>**,** <**path:file**>**,** <**string:name**>**[,** <**identifier**> **=**<**scalar**>**])** Reads a sketch (i.e. a closed contur) from a file. The geometry in the sketch is expected to be in the X-Y-Plane. It is placed on the given plane "pl". Sketch file format is recognized from the file name extension. Supported are ".dxf" and ".fcstd" (FreeCAD). The name is interpreted as layer name in DXF and sketch name in FreeCAD files.

For FreeCAD sketches, a list of parameter values can optionally be supplied. Upon loading, the sketch will be regenerated through FreeCAD with these values.

loadmodel(<**identifier:modelname**> **[,** <**identifier**> **=** <**feature**> |<**datum**> |<**vector**> |<**scalar**> **])** Parses another Insight CAD model (submodel) and inserts a compound of all features into the current model, which were marked as components in the submodel (i.e. which were defined with the colon ":" operator instead of the equal sign "=" in the submodel).

The model filename has to be "<modelname>.iscad". It is searched for in the following directories:

1. the directories listed in the environment variable "ISCAD_MODEL_PATH" (separated by ":")
2. the subdirectory "iscad-library" in InsightCAEs shared file directory
3. in the current directory

Optionally, a list of symbols is inserted into the namespace of the submodel (parameters).

4.3 Geometry Construction

Primitives There are commands for creation of several different geometrical primitives, e.g.

- 1D: Arc, Line, SplineCurve
- 2D: Quad, Tri (triangle), RegPoly (regular polygon), Circle, SplineSurface
- 3D: Sphere, Cylinder, Box, Bar, Cone, Pyramid, Torus

Extrusion(<**feature:f**>**,** <**vector:**$\vec{L}$> **[, centered])** Extrude the feature f with direction and length $\vec{L}$. When the keyword "centered" is given, the extrusion is centered around f.

Revolution(<**feature:f**>**,** <**vector:**$\vec{p}_0$>**,** <**vector:axis**>**,** <**scalar:**φ> **[, centered])** Creates a revolution of the planar feature f. The rotation axis is specified by origin point $\vec{p}_0$ and the direction vector axis. Revolution angle is specified by φ. By giving the keyword "centered", the revolution is created symmetrically around the base feature.

4.4 Other Feature Commands

There are more features available. A comprehensive list is obtained by pressing Ctrl+F in the iscad editor.

5 Lower Dimensional Shape Selection

ISCAD supports rule based selection of lower dimensional features. The selection is generated by a selection command, a question mark, followed by the type of shape to query. The result is a selection object:

<feature expression|feature selection>?<vertices|edges|faces|solids> ('<selection command string>'[, parameter o[, ..., parameter n]])

An example: the following expression selects the circumferential face of the cylinder c (all faces, which are not plane) and stores the selection in shell_faces:

```
c = Cylinder(0, 5*EZ);
shell_face = c ? faces('!isPlane');
min_end_face = c ? faces('isPlane && minimal(CoG.z)');
```

The selection command string contains rules for the selection. Finally, the command string is evaluated as a boolean expression comprising comparison operators, boolean operators and query functions. Within these boolean expressions, quantity functions can be used. The available set of query functions and quantity functions depends on the type of shape which shall be queried.

General functions available for all kinds of lower dimensional shapes:

Command	Returns[1]	Description
==, <, >, >=, <=	B	value comparison
!	B	not
&&	B	and
||	B	or
<value 1> ~<value 2> <tolerance>	B	approximate equality
angleMag(<vec 1>, <vec 2>)	Q	angle between vec 1 and vec 2
angle(<vec 1>, <vec 2>)	Q	angle between vec 1 and vec 2
%d<index>	Q	parameter <index> as scalar
%m<index>	Q	parameter <index> as vector
%<index>	Q	parameter <index> as selection set
in(<selection set>)	B	true if shape is in other selection
maximal(<quantity>)	B	true for the shape with maximum quantity
minimal(<quantity>)	B	true for the shape with minimum quantity

Specialized functions:

- Vertices

Command	Returns	Description
loc	Q	location of the vertex

- Edges

Command	Returns	Description
isLine	B	true, if edge is straight
isCircle	B	true, if edge is circular
isEllipse	B	true, if edge is elliptical
isHyperbola	B	true, if edge is on a hyperbola
isParabola	B	true, if edge is on a parabola
isBezierCurve	B	true, if edge is a bezier curve
isBSplineCurve	B	true, if edge is a BSpline curve
isOtherCurve	B	true, if edge is none of the above

[1] Q: returns quantity, B: returns boolean

Command	Returns	Description
isFaceBoundary	B	true, if edge is boundary of some face
boundaryOfFace(<sel. set>)	B	if edge is boundary of one of the faces in set
isPartOfSolid(<set>)	B	if edge is part of one of the solids in set
isCoincident(<set>)	B	if edge is coincident with one of the edges in set
isIdentical(<set>)	B	if edge is identical with one of the edges in set
projectionIsCoincident(<set>,<vec:po>,<vec:n>,<vec:up>,<scalar:tol>)	B	if projection of edge is coincident with some edge in set
len	Q	length of edge
radialLen(<vec:ax>,stvec:po)	Q	radial distance between ends with respect to axis (po,ax)
CoG	Q	center of gravity of edge
start	Q	start point coordinates
end	Q	end point coordinates

- Faces

Command	Returns	Description
area	Q	area of the face
CoG	Q	center of gravity of the face
cylRadius	Q	radius of a cylindrical face
cylAxis	Q	axis direction of a cylindrical face
isPlane	B	true, if is a plane
isCylinder	B	true, if is a cylindrical surface
isCone	B	true, if is a conical surface
isSphere	B	true, if is a spherical surface
isTorus	B	true, if is a toroidal surface
isBezierSurface	B	true, if is a bezier surface
isBSplineSurface	B	true, if is a BSpline surface
isSurfaceOfRevolution	B	true, if is a surface of revolution
isSurfaceOfExtrusion	B	true, if is a surface of extrusion
isOffsetSurface	B	true, if is a offset surface
isOtherSurface	B	true, if is some other kind of surface
isPartOfSolid(<selection set>)	B	
isCoincident(<selection set>)	B	
isIdentical(<selection set>)	B	
adjacentToEdges(<selection set>)	B	
adjacentToFaces(<selection set>)	B	

- Solids

Command	Returns	Description
CoG	Q	center of gravity
volume	Q	volume of the solid

It is possible to supply additional arguments to the selection expression, like scalars, vectors or features.

6 Postprocessing Actions

DXF(<path:outputfile>) << <**feature:f**> <**view_definition**> **[,** <**view_definition**>**, ...]** The DXF post-processing action creates a DXF file for further use in drawings. Several views are derived from the feature f. A <view_definition> takes the following form:

<identifier:viewname> (
<vector:$\vec{p}_0$>, <vector:$\vec{n}$>, up <vector:$\vec{e}_{up}$>
[, section]
[, poly]
[, skiphl]
[, add [l] [r] [t] [b] [k]]
)
It defines a view on the point $\vec{p}_0$ with normal direction $\vec{n}$ of the view plane. The upward direction (Y-direction) is aligned with vector $\vec{e}_{up}$.

The keyword “section” toggles whether only the outline is projected or if the view plane creates a section through the geometry.

If keyword “poly” is given, the DXF geometry will be discretized. This is more robust but creates much larger DXF files.

Keyword “skiphl” toggles whether hidden lines are output.

The keyword “add” followed by the key letters l, r, t, b and/or k enables creation of additional projections from the left, right, top, bottom and/or back, respectively.

An example is given below:

```
c: Cylinder(0, 100*EX, 20);

@post

DXF("c.dxf") << c
 top ( 0, EX, up EY )
 front ( 0, EZ, up EY )
;
```

gmsh(<path:outputfile>) << <**feature:f**> **as** <**identifier:l**> <**mesh_parameters**> Generates a (triangular or tetrahedral) mesh for an FEA analysis of the feature f. Gmsh is used as a meshing backend. The mesh format is determined by the outputfile extension (.med = MED format). The label of the mesh is set to l.

The syntax of the <mesh_parameters> are as follows:

L = (<scalar:Lmax> <scalar:Lmin>)
[linear]
vertexGroups([<identifier:group name> = <vertex set> [@ <scalar:size>], ...])
edgeGroups([<identifier:group name> = <edge set> [@ <scalar:size>], ...])
faceGroups([<identifier:group name> = <face set> [@ <scalar:size>], ...])
[vertices ([<identifier:vertex name> = <vector:location>)])]

The general mesh size is set by Lmax and Lmin. The optional keyword “linear” switch from quadratic to linear elements. Named groups of vertices, edges and faces can be created using the keywords “vertexGroups”, “edgeGroups” and “faceGroups” repectively. Each group definition takes the form “groupname” = “selection set” (see section 5 for definition of selection sets). Optionally, a mesh size can be assigned to each defined group by appending an @ sign followed by a scalar value.

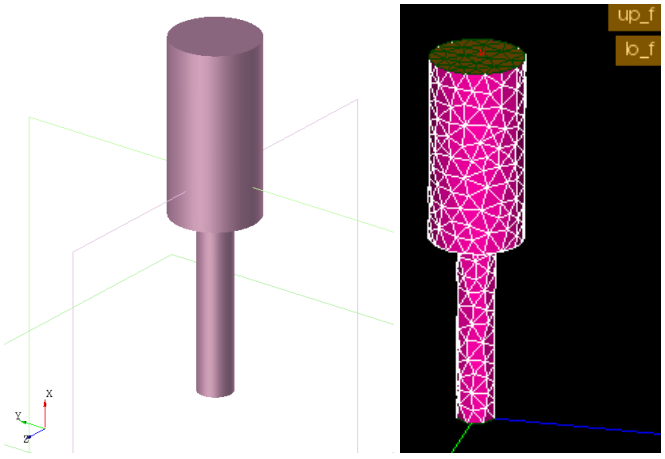
An example is given below:

```
c:
Cylinder(0, 100*EX, 20)
|
Cylinder(100*EX, ax 100*EX, 50)
;

@post
```

```
8
9  gmsh("c.med") << c as cyl
10  L = (10 0.1)
11  linear
12  vertexGroups()
13  edgeGroups()
14  faceGroups(
15   lo_f = c?faces('isPlane&&minimal(CoG.x)')
16   up_f = c?faces('isPlane&&maximal(CoG.x)')
17  )
18 ;
```

Result: ISCAD model (left) and resulting mesh (right):



7 Graphical Editor ISCAD

Iscad is a graphical editor for Insight CAD scripts. It consists of a text editor for editing the script contents and a 3D view and some other elements for inspecting the resulting model. A screenshot is shown in figure 1.

The model script is entered into the text editor widget right of the 3D display. Once a script shall be evaluated, it can be parsed by clicking in the button “Rebuild” or pressing Ctrl+Return.

After parsing the model, the following results are displayed:

- A list of all created feature symbols in the “Model Steps” list box. If the check box is checked, the 3D geometry is displayed in the 3D view.
- The values of all scalars and vectors in the “Variables” list box.

  For each vector variable, a check box is displayed. If it is checked, the vector is interpreted as a point location and the point is shown in the 3D display window.
- All defined datums are listed in the “Datums” list box. Again, the check box controls, whether the datum is displayed in the 3D view.

7.1 3D Graphics Display

View Manipulation

- Dragging: Shift + mouse move
- Scaling: Ctrl + horizontal mouse move

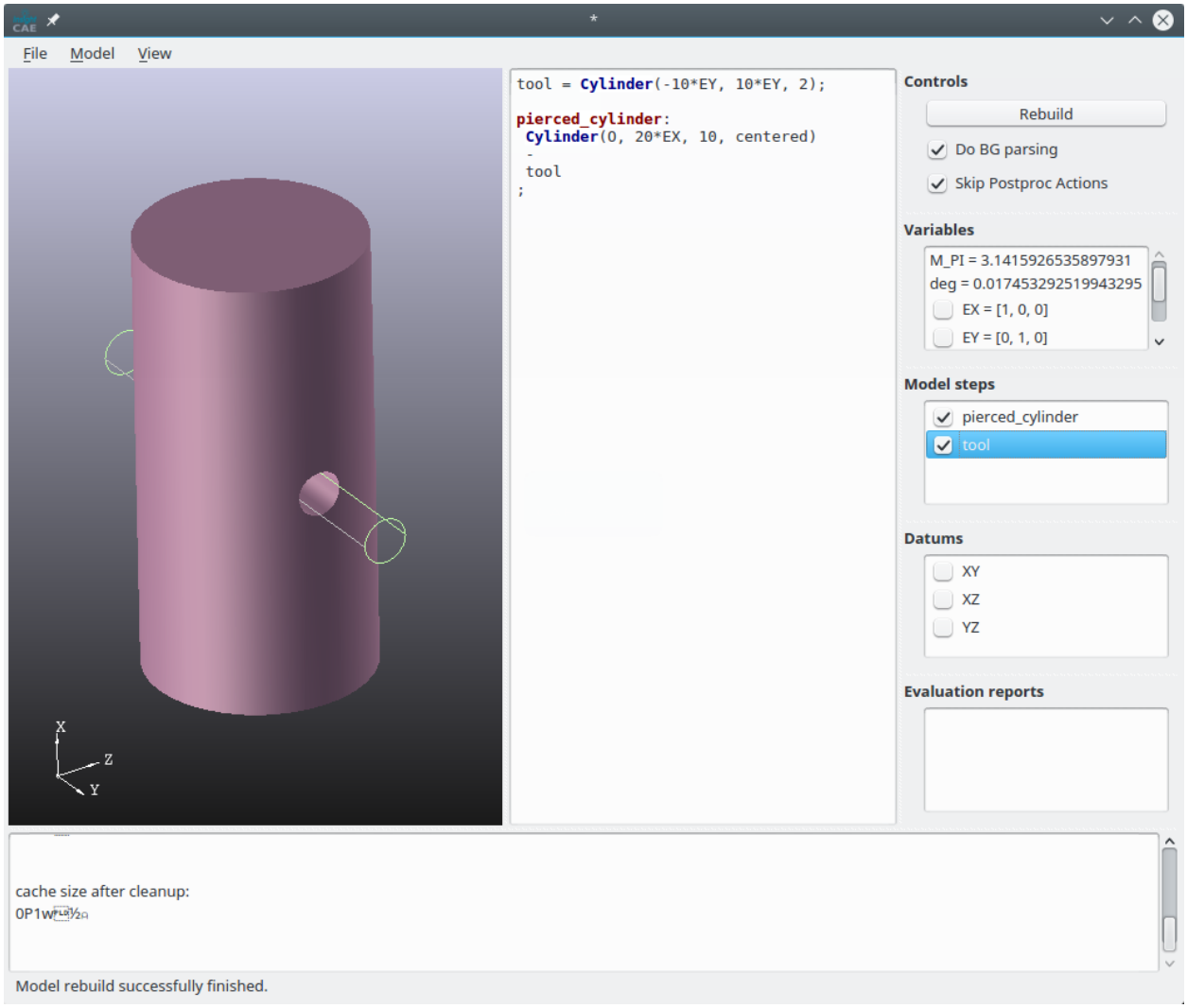


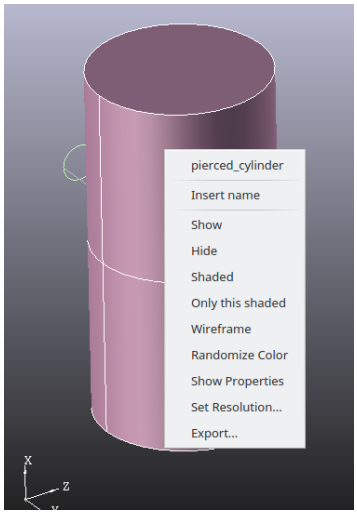
Figure 1: Screenshot of the ISCAD main window

- Rotating: Alt + mouse move

Model Navigation When hoovering the mouse pointer over a displayed feature in the 3D geometry window, it is highlighted. The highlighted feature is the one, which would be selected during subsequent mouse clicks.

When the left mouse button is pressed, the highlighted feature is selected and all its contained reference points are displayed.

When the right mouse button is pressed, a context menu for the selected feature is displayed.



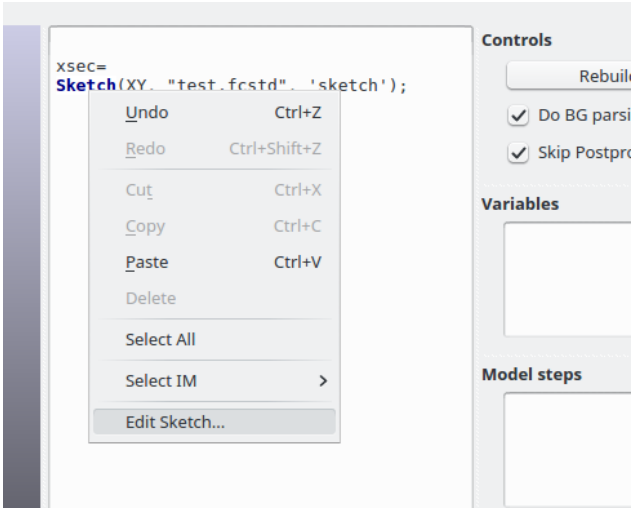
The context menu provides these functions:

- The first entry is the feature name. When it is selected, the definition in the script editor is highlighted and the cursor jumps to it.
- “Insert name”: inserts the name of the feature symbol at the cursor location
- “Export...”: Export the feature geometry to a file (BREP, STP, IGES, STL)

7.2 Text Editor

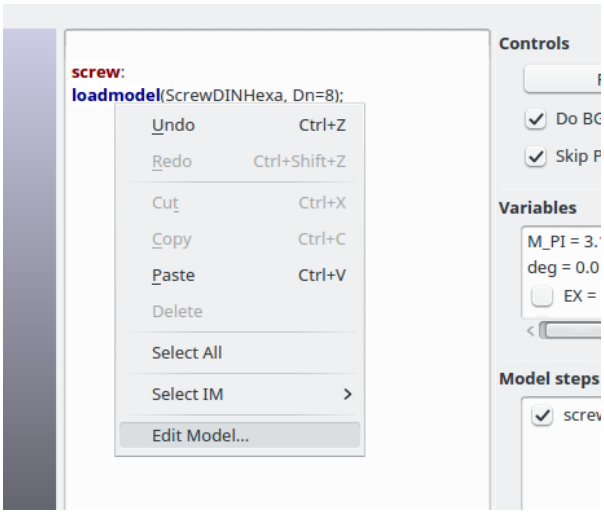
When script code is entered into the text editor window, it is parsed in the background. Once this has been done successfully, some extensions of the context menu is available:

- Context menu on “Sketch” command:



  When selecting the “Edit Sketch...” entry, FreeCAD is launched and the sketch editor opened. If the FreeCAD-file or sketch inside it is not yet existing, they are created.

- Context menu on “loadmodel” command:



When selecting the “Edit Model...” entry, another instance of iscad is launched with the specified model script loaded.