CSM Commands

Primitives

POINT	xloc yloc zloc
BOX	xbase ybase zbase dx dy dz
SPHERE	xcent ycent zcent radius
CYLINDER	xbeg ybeg zbeg xend yend zend radius
CONE	xvrtx yvrtx zvrtx xbase ybase zbase radius
TORUS	xcent ycent zcent dxaxis dyaxis dzaxis ...
	... majorRad minorRad
IMPORT	$filename bodynumber=1
UDPRIM	$primtype $argName1 argValue1 ...argValue4
	name → UDP/UDF
	/name → path($pwd)/name.udc
	$/name → path($csm)/name.udc
	$$/name → path($root)/udc/name.udc
RESTORE	$name index=0

Grown

EXTRUDE	dx dy dz
RULE	reorder=0
BLEND	begList=0 endList=0 reorder=0 oneFace=0
REVOLVE	xorig yorig zorig dxaxis dyaxis dzaxis angDeg
SWEEP	
LOFT*	smooth

Applied

FILLET	radius edgeList=0 listStyle=0
CHAMFER	radius edgeList=0 listStyle=0
HOLLOW	thick=0 entList=0 listStyle=0

Booleans

INTERSECT	$order=none index=1 maxtol=0
SUBTRACT	$order=none index=1 maxtol=0
UNION	toMark=0 trimList=0 maxtol=0
JOIN	toler=0 toMark=0
CONNECT	faceList1 faceList2 edgeList1=0 edgeList2=0
EXTRACT	entList
COMBINE	toler=0

Transforms

TRANSLATE	dx dy dz
ROTATEX	angDeg yaxis zaxis
ROTATEY	angDeg zaxis xaxis
ROTATEZ	angDeg xaxis yaxis
SCALE	fact xcent=0 ycent=0 zcent=0
MIRROR	nx ny nz dist=0
APPLYCSYS	$csysName ibody=0
REORDER	ishift iflip=0

Sketch

SKBEG	x y z relative=0
SKVAR	$type valList
SKCON	$type index1 index2=-1 $value=0
LINSEG	x y z
CIRARC	xon yon zon xend yend zend
ARC	xend yend zend dist $plane=xy
SPLINE	x y z
SSLOPE	dx dy dz
BEZIER	x y z
SKEND	wireonly=0

Solver

SOLBEG	$varList
SOLCON	$expr
SOLEND	

Stack

MARK	
STORE	$name index=0 keep=0
GROUP	nbody=0

Logic

IFTHEN	val1 $op1 val2 $op2=and val3 $op3 val4
ELSEIF	val1 $op1 val2 $op2=and val3 $op3 val4
ELSE	
ENDIF	

Looping

PATBEG	$pmtrName ncopy
PATBREAK	expr
PATEND	

Error handling

CATBEG	sigCode
CATEND	
THROW	sigCode

Declarations

DIMENSION	$pmtrName nrow ncol despmtr=0
CFGPMTR	$pmtrName values
DESPMTR	$pmtrName values
CONPMTR	$pmtrName value
OUTPMTR	$pmtrName
LBOUND	$pmtrName bounds
UBOUND	$pmtrName bounds

Attribution

ATTRIBUTE	$attrName attrValue
CSYSTEM	$csysName csysList
GETATTR	$pmtrName attrID global=0

User-defined components

INTERFACE	$argName $argType default=0
END	

Miscellaneous

SET	$pmtrName exprs
UDPARG	$primtype $argName1 argValue1 ...
SELECT	$type arg1 ...
ASSERT	arg1 arg2 toler=0 verify=0
DUMP	$filename remove=0 toMark=0
EVALUATE	$type arg1 ...
NAME	$branchName
PROJECT	x y z dx dy dz useEdges=0
MESSAGE	$text $schar=_

User-defined Primitives/Functions

bezier	$filename debug imax jmax cp[]
biconvex	thick camber
box	dx dy dz rad @area @volume
createBEM	$filename space imin imax nocrod
createPoly	$filename hole[]
csm	$filename $pmtrname pmtrvalue @volume
droop	xle thetale xye thetate
editAttr	$attrname $input $output overwrite
	$filename verbose @nchange
ellipse	rx ry rz nedge thbeg
fitcurve	$filename ncp ordered periodic...
	... xform[] xyz[] @npnt @rms
flend	fraca fracb toler plot
freeform	$filename imax jmax kmax xyz[]
ganged	$op toler
guide	nxsect origin axis
hex	corners[] uknots[] vknots[] wknots[] @area @volume
import	$filename bodynumber @numbodies
kulfan	class[] ztail[] aupper[] alower[]
naca	series thickness camber maxloc offset sharpte
naca456	thkcode toc xmaxt leindex camcode cmax xmaxc cl a

(continued on other side)

(UDPs/UDFs — continued from other side)

nurbbody	`$filename`
parsec	`yte poly[] param[] meanline ztail[]`
pod	`length fineness @volume`
poly	`points[]`
printBbox	
printBrep	
printEgo	
radwaf	`ysize zsize nspoke xframe[]`
sew	`$filename toler bodynum`
stag	`rad1 beta1 gama1 rad2 beta2 gama2 ...` `... alfa xfrnt xrear`
stiffener	`beg[] end[] depth angle`
supell	`rx rx_w rx_e ry ry_s ry_n n n_w n_e ...` `... n_s n_n n_sw n_se n_nw n_ne offset nquad`
waffle	`depth segments[] $filename progress`

User-defined Components

$$/applyTparams	
	`factor`
$$/biconvex	`thick`
$$/boxudc	`dx dy dz @volume`
$$/contains	`@contains`
$$/diamond	`thick`
$$/flapz	`xflap[] yflap[] theta gap openEnd`
$$/gen_rot	`xbeg ybeg zbeg xend yend zend...` `... rotang @azimuth @elevation`
$$/overlaps	`@overlaps`
$$/popupz	`xbox[] ybox[] height`
$$/spoilerz	`xbox[] ybox[] depth thick theta overlap extend`
$$/swap	

Built-in Functions

General

```
pi(x)
min(x,y)
max(x,y)
sqrt(x)
abs(x)
int(x)
nint(x)
ceil(x)
floor(x)
mod(a,b)
sign(test)
exp(x)
log(x)
```

Trigonometric

```
log10(x)
sin(x)
sind(x)
asin(x)
asind(x)
cos(x)
cosd(x)
acos(x)
acosd(x)
tan(x)
tand(x)
atan(x)
atand(x)
atan2(y,x)
atan2d(y,x)
hypot(x,y)
hypot3(x,y,z)
```

Sketch utilities

```
incline(xa,ya,dab,xb,yb)
Xcent(xa,ya,dab,xb,yb)
Ycent(xa,ya,dab,xb,yb)
Xmidl(xa,ya,dab,xb,yb)
Ymidl(xa,ya,dab,xb,yb)
seglen(xa,ya,dab,xb,yb)
radius(xa,ya,dab,xb,yb)
sweep(xa,ya,dab,xb,yb)
turnang(xa,ya,dab,xb,yb,dbc,xc,yc)
dip(xa,ya,xb,yb,rad)
smallang(x)
```

Conversions

```
val2str(num,digits)
str2val(string)
findstr(str1,str2)
slice(str,ibeg,iend)
path($pwd) or path($csm) or path($root) or path($file)
```

Logic

```
ifzero(test,ifTrue,ifFalse)
ifpos(test,ifTrue,ifFalse)
ifneg(test,ifTrue,ifFalse)
ifmatch(str,pat,ifTrue,ifFalse)
ifnan(test,ifTrue,ifFalse)
```

Dot-suffixes

`x.nrow`	number of rows in `x` or 0 if a string
`x.ncol`	number of columns in `x` or 0 if a string
`x.size`	number of elements in `x` (=x.nrow*x.ncol) or len of str `x`
`x.sum`	sum of elements in `x`
`x.norm`	L2-norm (RMS) of elements in `x`
`x.min`	minimum value in `x`
`x.max`	maximum value in `x`

Character Set

#	hash	introduces comment
”	quotes	ignore spaces until following ”
\	backslash	ignore this and following characters and concatenate next line
<space>	space	separates arguments in .csm file (except between ” and ”)
0-9		digits used in numbers, names, and strings
A-Z a-z		letters used in names and strings
_ : @		characters used in names and strings
? % =		characters used in strings
.	period	decimal separator (used in numbers), introduces dot-suffixes (in names)
,	comma	separates function arguments and row/column in subscripts
;	semicolon	multi-value item separator
( )	parentheses	groups expressions and function arguments
[ ]	brackets	specifies subscripts in form [row,column] or [index]
{ } < >		characters used in strings
+ - * / ^		arithmetic operators
$	dollar	as first character, introduces a string that is terminated by end-of-line or un-escaped plus, comma, or open-bracket
@	at-sign	as first character, introduces @-parameters
’	apostrophe	used to escape comma, plus, or open-bracket within strings
!	exclamation	if first character of implicit string, ignore $! and treat as an expression
|	bar	cannot be used (reserved for OpenCSM internals)
~	tilde	cannot be used (reserved for OpenCSM internals)
&	ampersand	cannot be used (reserved for OpenCSM internals)