

Engineering Sketch Pad (ESP)



Training Session 3 Solids Fundamentals (2)

John F. Dannenhoffer, III

jfdannen@syr.edu
Syracuse University

Bob Haines

haines@mit.edu
Massachusetts Institute of Technology
updated for v1.18

- Miscellaneous Branches
- Manipulating the Stack
 - GROUP
 - STORE, RESTORE
- Grown Bodys
 - EXTRUDE
 - REVOLVE
 - RULE
 - BLEND
- Creating a Waffle
 - UDPRIM WAFFLE
- Homework Exercises

- SET — set the value of a Local Variable to the given expression
- MARK — push a Mark onto the Stack
- SELECT — select entity for which @-parameters are evaluated
 - see “help” for details
- PROJECT — find the first projection from a given point (in space) in a given direction

- DUMP — write file that contains the Body (not Group) on the top of the Stack
 - if `remove` is not zero, the Body is popped off the Stack
 - if `toMark` is not zero, all Bodys since the Mark are written
- The types of files that can be written by DUMP include:
 - `.brep` or `.BREP` — OpenCASCADE output
 - `.bstl` or `.BSTL` — binary stereolithography output
 - `.egads` or `.EGADS` — EGADS output
 - `.egg` or `.EGG` — EGG restart output
 - `.igs` or `.IGS` — IGES output
 - `.sens` or `.SENS` — sensitivity information
 - `.step` or `.STEP` — STEP output
 - `.stl` or `.STL` — ASCII stereolithography output
 - `.stp` or `.STP` — STEP output
 - `.tess` or `.TESS` — ASCII tessellation output
 - `.ugrid` or `.UGRID` — ASCII AFLR3 output

- During the build process, **OpenCSM** maintains a last-in-first-out (LIFO) “Stack” that can contain Bodys, Marks, and Sketches.
- The `.csm` statements are executed in a stack-like way, taking their inputs from the Stack and depositing their results onto the Stack.
- Bodys can be grouped with the **GROUP** statement
 - all the Bodys back to the Mark (or the beginning of the Stack) are put into a single Group
 - some operations, such as the transformations, **ATTRIBUTE**, **STORE**, and **DUMP** operate on all Bodys in the Group simultaneously
 - Bodys can be ungrouped by giving **GROUP** a negative argument

- The Group on the top of the Stack can be “popped” off the Stack with a `STORE $name index` command
 - if the **name** is alpha-numeric, the Group is stored in a named storage location, with the given **index** (from 0 to 99)
 - if the **name** is a dot (`.`), the Group is not stored (just popped off the Stack)
 - if the **name** is two dots (`..`), all the Groups back to the Mark are popped off the Stack (and not stored)
 - if the **name** is three dots (`...`), everything is popped off the Stack

- Groups can be read from a named storage location and “pushed” onto the Stack with the `RESTORE $name index` command
- The `RESTORE` command is considered a primitive, so its Attributes are put on all the Bodys and all their Faces

- Assume that the Stack contains: 5 7 9 12 (top)
- If one wants to reverse the top two Bodies, use
 - STORE temp 1
 - Stack now contains: 5 7 9
 - storage temp 1 contains 12
 - STORE temp 2
 - Stack now contains: 5 7
 - storage temp 2 contains 9
 - RESTORE temp 1
 - Stack now contains: 5 7 12
 - RESTORE temp 2
 - Stack now contains: 5 7 12 9

- Assume that the Stack contains: 5 7 9 12 (top)
- If one wants to put a mark between the 7 and 9, use
 - `STORE temp 1`
 - Stack now contains: 5 7 9
 - storage `temp 1` contains 12
 - `STORE temp 2`
 - Stack now contains: 5 7
 - storage `temp 2` contains 9
 - `MARK`
 - Stack now contains: 5 7 mark
 - `RESTORE temp 2`
 - Stack now contains: 5 7 mark 9
 - `RESTORE temp 1`
 - Stack now contains: 5 7 mark 9 12

- If you want to duplicate the Group on the top of the Stack, use **STORE** and **RESTORE**
- Depending on the value of **keep** in the **STORE** command, the Group on the top of the Stack is either kept (like a “copy”) or popped off the Stack (like a “cut”)

- not using the **keep** option to duplicate the Body on the top of the Stack

```
STORE    temp
```

```
RESTORE temp
```

```
RESTORE temp
```

- using the **keep** option to duplicate the Body on the top of the Stack

```
STORE    temp 0 1
```

```
RESTORE temp
```

- Use the **DIMENSION** statement to set the size of the array
 - **DIMENSION** creates a Branch, so its arguments can be any expression
- Use the **SET** statement to define the values
 - if name of array is given, set all the values
 - if more values are given than needed, excess are ignored
 - if fewer values are given than needed, last value is repeated

```
CFGPMTR    numRows 3
CONPMTR    numCols 2
DIMENSION  array    numRows numCols
SET        array    "5;2"
```

creates: array = [5, 2, 2, 2, 2]

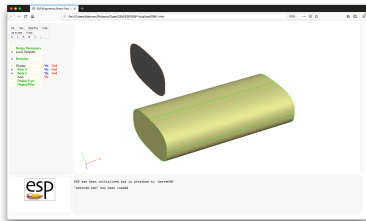
- A single array element can be assigned with

```
SET        array[2,1] 3
```

- Pops one or more SheetBodys from the Stack
- Pushes the resultant Body onto the Stack
- Supported grown features include:
 - EXTRUDE — in a given direction for a given distance
 - REVOLVE — around a given axis for a given angular displacement
 - RULE — connect all the SheetBodys/WireBodys back to the Mark by straight lines
 - the first and/or last Xsect can be a NodeBody
 - BLEND — connect all the SheetBodys/WireBodys back to the Mark with smooth curves
 - the first and/or last Xsect can be a NodeBody
 - at the bounding Nodes, the user can specify the radius of curvature in two orthogonal directions
 - SWEEP — a SheetBody/WireBody along a given WireBody
 - this is often problematic in OpenCASCADE
 - LOFT — similar to BLEND, but with less control

- Pops one or more WireBodys from the Stack
- Pushes the resultant Body onto the Stack
- Supported grown features include:
 - **EXTRUDE** — in a given direction for a given distance
 - **REVOLVE** — around a given axis for a given angular displacement
 - **RULE** — connect all the WireBodys back to the Mark by straight lines
 - the first and/or last Xsect can be a NodeBody
 - **BLEND** — connect all the WireBodys back to the Mark with smooth curves
 - the first and/or last Xsect can be a NodeBody
 - at the bounding Nodes, the user can specify the radius of curvature in two orthogonal directions

Note: Original Xsect (SheetBody) and result of EXTRUDE are shown



extrude

```
UDPRIM  supell rx 2 ry_n 1 ry_s 1 n 3  
ROTATEY 90 0 0  
STORE  sections
```

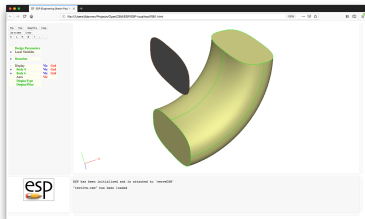
```
RESTORE sections  
TRANSLATE 0 4 0
```

```
RESTORE sections  
EXTRUDE 8 0 0
```

END

- Face-order is: (1) orig Xsect, (2) copy of Xsect, (3) Face from first Xsect Edge, (4) Face from second Xsect Edge, ...

Note: Original Xsect (SheetBody) and result of REVOLVE are shown



```
# revolve
```

```
UDPRIM  supell rx 2 ry_n 1 ry_s 1 n 3  
ROTATEY 90 0 0  
STORE  sections
```

```
RESTORE sections  
TRANSLATE 0 4 0
```

```
RESTORE sections  
REVOLVE 0 4 0 0 0 1 90
```

```
END
```

- Face-order is: (1) orig Xsect, (2) copy of Xsect, (3) Face from first Xsect Edge, (4) Face from second Xsect Edge, ...

- To revolve a Xsect to make a body of revolution:

- do not use:

```
# make whole Body
```

```
REVOLVE  0 0 0    0 1 0    360
```

- use instead:

```
# make half on Body
```

```
REVOLVE  0 0 0    0 1 0    180
```

```
# mirror for second half
```

```
STORE    half    0    1
```

```
RESTORE  half    0
```

```
MIRROR   0    0    1    0
```

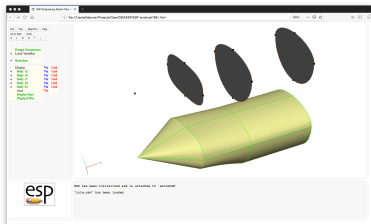
```
# put it all together
```

```
JOIN     0    0
```




Grown Primitive — RULE

Note: Original Xsects (SheetBodys) and result of RULE are shown



```
# rule
```

```
MARK
```

```
POINT 0 0 0
```

```
UDPRIM supell rx 2 ry_n 1 ry_s 1 n 3
```

```
ROTATEY 90 0 0
```

```
TRANSLATE 3 0 0
```

```
UDPRIM supell rx 2 ry_n 1 ry_s 2
```

```
ROTATEY 90 0 0
```

```
TRANSLATE 6 0 0
```

```
UDPRIM supell rx 2 ry_n 1 ry_s 2
```

```
ROTATEY 90 0 0
```

```
TRANSLATE 10 0 0
```

```
GROUP
```

```
STORE sections
```

```
RESTORE sections
```

```
TRANSLATE 0 4 0
```

```
MARK
```

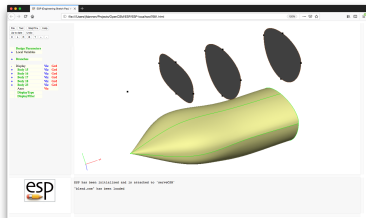
```
RESTORE sections
```

```
RULE
```

```
END
```

● Face-order on later slide

Note: Original Xsects (SheetBodys) and result of BLEND are shown



```
# blend
```

```
MARK
```

```
POINT 0 0 0
```

```
UDPRIM supell rx 2 ry_n 1 ry_s 1 n 3
```

```
ROTATEY 90 0 0
```

```
TRANSLATE 3 0 0
```

```
UDPRIM supell rx 2 ry_n 1 ry_s 2
```

```
ROTATEY 90 0 0
```

```
TRANSLATE 6 0 0
```

```
UDPRIM supell rx 2 ry_n 1 ry_s 2
```

```
ROTATEY 90 0 0
```

```
TRANSLATE 10 0 0
```

```
GROUP
```

```
STORE sections
```

```
RESTORE sections
```

```
TRANSLATE 0 4 0
```

```
MARK
```

```
RESTORE sections
```

```
BLEND
```

```
END
```

● Face-order on later slide

- If the first and last Xsects are both WireBodys
 - a SheetBody is produced that is open on both ends
- If the first or last Xsect is a WireBody
 - a SheetBody is produced that is open on one end and closed on the other
- Otherwise
 - a SolidBody is produced

- (1) first Xsect (or empty if POINT)
- (2) last Xsect (or empty if POINT)
- (3) Face from first Xsect Edge between first and second Xsects
- (4) Face from first Xsect Edge between second and third Xsectss
- ...
- (n) Face from second Xsect Edge between first and second Xsects
- ...

- RULE and BLEND require that all Xsects have the same number of Segments, ordered in the same way
 - new Faces are made by combining all the first Segments, ...
- BLEND allows user-selectable continuity in blend direction
 - C2 - curvature continuity (the default)
 - C1 - slope continuity (obtained with Face repeated once)
 - C0 - value continuity (obtained with Face repeated twice)
- Xsects can be automatically reordered to help eliminate twist by setting **reorder** to a non-zero value
 - positive to start from first Xsect
 - negative to start from last Xsect
- Users can manually reorder Xsects with the **REORDER** command (applied to a Xsect)
 - Reordering only changes the order of Segments, not their shapes



BLEND Continuity (1)

```
# blendCOC1C2
```

```
# original Xsects (top left)
```

```
MARK
```

```
POINT -2 0 0
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
TRANSLATE +2 0 0
```

```
GROUP
```

```
TRANSLATE -3 +1 0
```

```
# Body with C0 at second Xsect (top rite)
```

```
MARK
```

```
POINT -2 0 0
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
TRANSLATE +2 0 0
```

```
BLEND
```

```
TRANSLATE +3 +1 0
```

```
# Body with C1 at second Xsect (bottom left)
```

```
MARK
```

```
POINT -2 0 0
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
TRANSLATE +2 0 0
```

```
BLEND
```

```
TRANSLATE -3 -1 0
```

```
# Body with C2 at second Xsect (bottom rite)
```

```
MARK
```

```
POINT -2 0 0
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
TRANSLATE +2 0 0
```

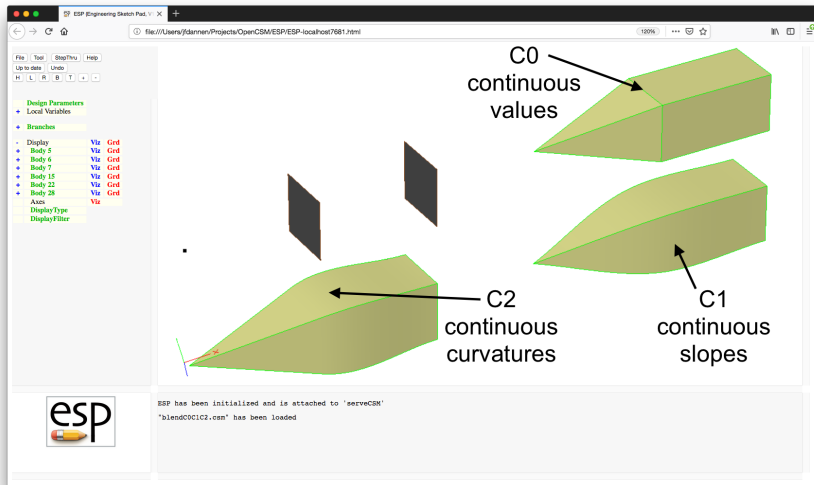
```
BLEND
```

```
TRANSLATE +3 -1 0
```

```
END
```



BLEND Continuity (2)





BLEND Nose/Tail Treatment (1)

```
# blendCOC1C2
```

```
# original Xsects (top left)
```

```
MARK
```

```
POINT -2 0 0
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
TRANSLATE +2 0 0
```

```
GROUP
```

```
TRANSLATE -3 +1 0
```

```
# Body with pointed nose (top rite)
```

```
MARK
```

```
POINT -2 0 0
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
TRANSLATE +2 0 0
```

```
BLEND
```

```
TRANSLATE +3 +1 0
```

```
# Body with slightly rounded nose (bottom left)
```

```
MARK
```

```
POINT -2 0 0
```

```
UDPRIM box dy 1 dz 1
```

```
UDPRIM box dy 1 dz 1
```

```
TRANSLATE +2 0 0
```

```
BLEND "0.1; 0;1;0; 0.1; 0;0;1"
```

```
TRANSLATE -3 -1 0
```

```
# Body with rounded nose (bottom rite)
```

```
MARK
```

```
POINT -2 0 0
```

```
UDPRIM box dy 1 dz 1
```

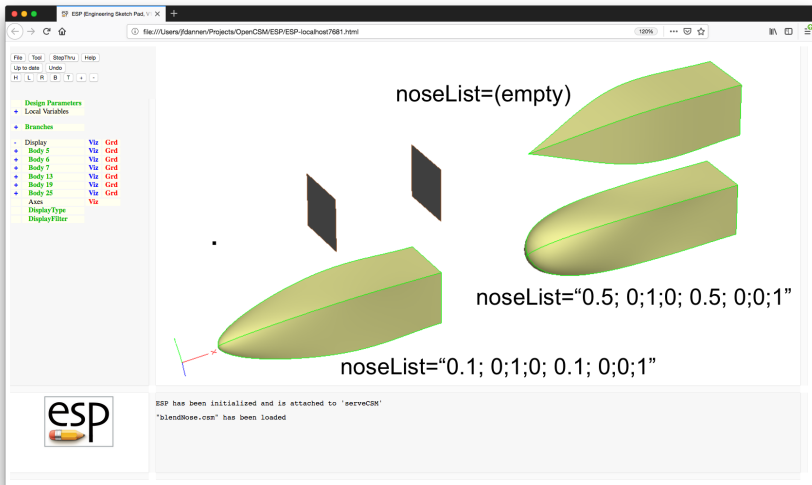
```
UDPRIM box dy 1 dz 1
```

```
TRANSLATE +2 0 0
```

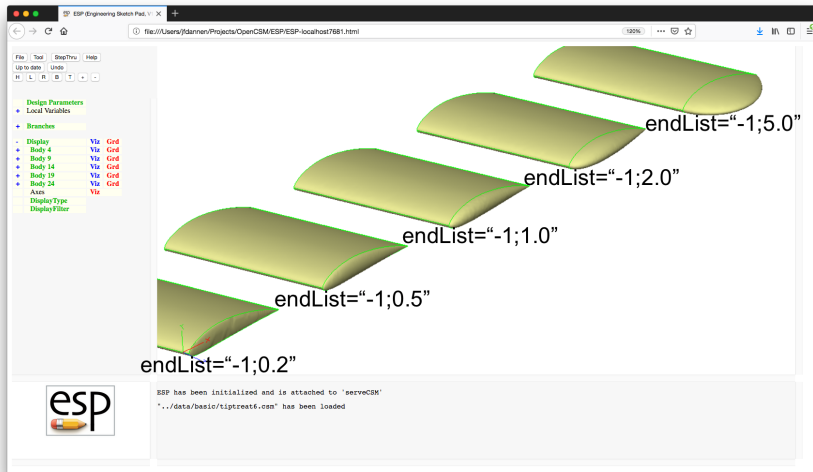
```
BLEND "0.5; 0;1;0; 0.5; 0;0;1"
```

```
TRANSLATE +3 -1 0
```

```
END
```

- If the first Xsect is a SheetBody with 2 or 3 Edges and the `begList` contains 2 entries:
 - `begList[1] = -1`
 - `begList[2] =` the aspect ratio of an approximate ellipse that spans between the first and second Xsect Edge
- The same applies to the last Xsect and `endList`



- Called with `.csm` statement:
`UDPRIM waffle depth <number> filename <name_of_file>`
- Valid statements in file are:
 - `CPOINT` — create a construction point (not in final waffle)
 - `CLINE` — create a construction line (not in final waffle)
 - `POINT` — create a waffle point
 - `LINE` — create one or more waffle segments
 - `PATBEG/PATEND` — create a pattern (loop)
- Keywords can be in lowercase or UPPERCASE
- Coordinates of existing point `<pname>` are given by
 - `x@<pname>` and `y@<pname>`

- Variants of CPOINT and POINT
 - POINT <pname> AT <xloc> <yloc>
 - create point at <xloc,yloc>
 - POINT <pname> ON <lname> FRAC <fracDist>
 - creates point on <lname> at given fractional distance
 - POINT <pname> ON <lname> XLOC <x>
 - creates point on <lname> at given <x>
 - POINT <pname> ON <lname> YLOC <y>
 - creates point on <lname> at given <y>
 - POINT <pname> ON <lname> PERP <pname2>
 - creates point on <lname> that is closest to <pname2>
 - POINT <pname> ON <lname> XSECT <lname2>
 - creates point at intersection of <lname> and <lname2>
 - POINT <pname> OFF <lname> <dist> <pname2>
 - creates point <dist> to the left of <lname> at <pname2>

- Variants of CLINE and LINE

- `LINE . <pname1> <pname2> <attrName1=attrValue1>...`
 - creates unnamed line between <pname1> and <pname2> with given attribute(s) (if any)



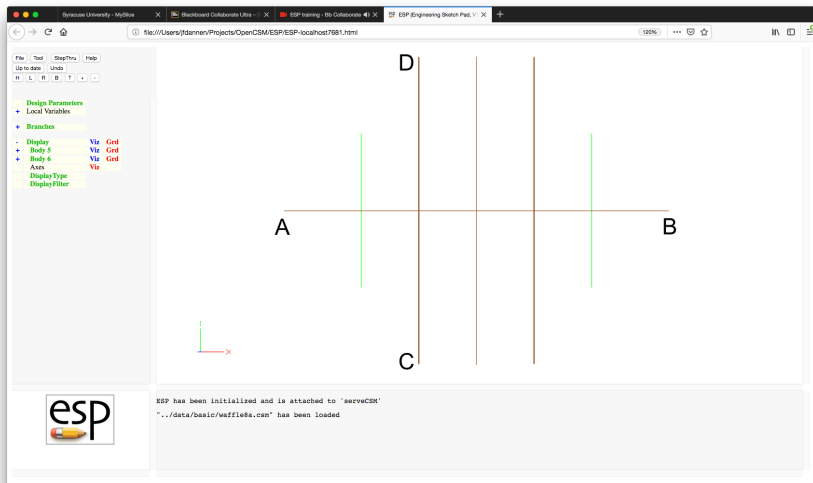
`LINE <lname> <pname1> <pname2> <attrName1=attrValue1>`

- creates line named <lname> between <pname1> and <pname2> with given attribute(s) (if any)



Waffle Example (1)

SolidBody in green; Waffle in brown



```
# SolidBody
CYLINDER 0 0 0 3 0 0 1
STORE    SolidBody

# get bounding box of SolidBody
RESTORE  SolidBody
SET      xmin  @xmin
SET      xmax  @xmax
SET      ymin  @ymin
SET      ymax  @ymax
SET      zmin  @zmin
SET      zmax  @zmax
STORE    .
```



```
# Waffle (centered on SolidBody)
UDPRIM    waffle    filename <<    depth zmax-zmin+2
  POINT   A AT    xmin-1  (ymin+ymax)/2
  POINT   B AT    xmax+1  (ymin+ymax)/2
  LINE     AB  A   B    type=symmetry

  PATBEG   i   3
    POINT   C AT    xmin+i/4*(xmax-xmin) ymin-1
    POINT   D AT    xmin+i/4*(xmax-xmin) ymax+1
    LINE     .   C   D    type=!$bulkhead_+i
  PATEND

>>
TRANSLATE 0 0 zmin-1
STORE      Waffle
```

```
# score the SolidBody by the Waffle and extract Faces
RESTORE    SolidBody
RESTORE    Waffle
SUBTRACT
EXTRACT    0

# generate the internal structure
RESTORE    SolidBody
RESTORE    Waffle
INTERSECT

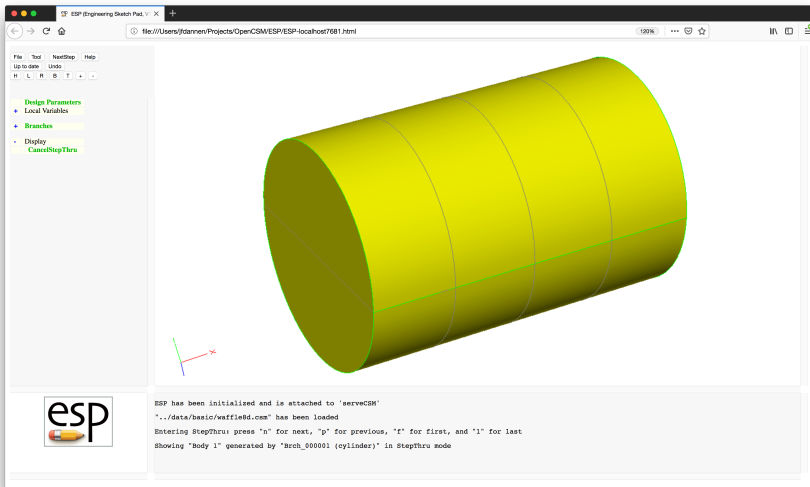
# put them together
UNION

END
```



Waffle Example (5)

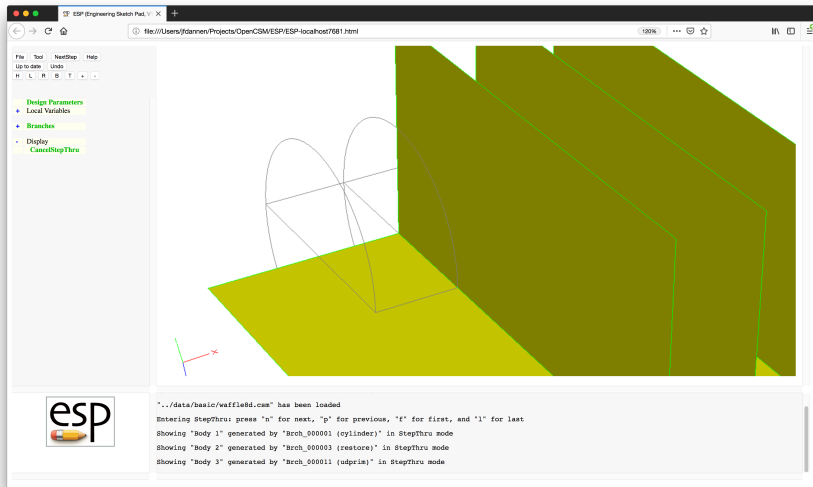
Original SolidBody





Waffle Example (6)

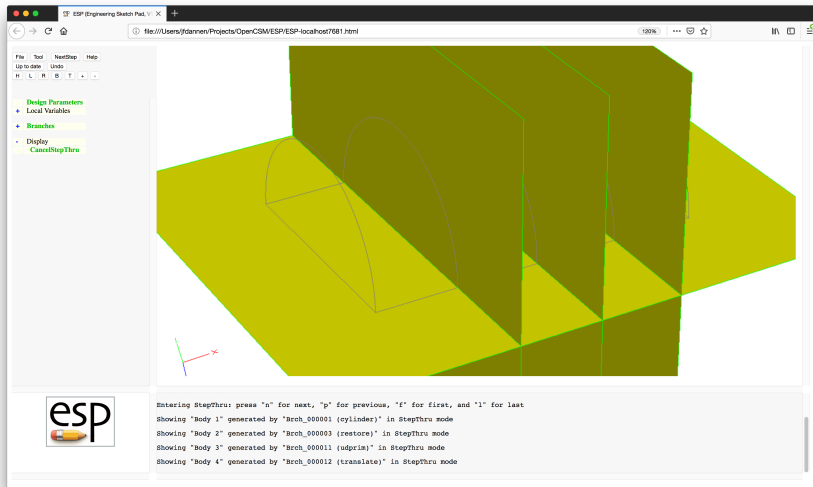
Original Waffle





Waffle Example (7)

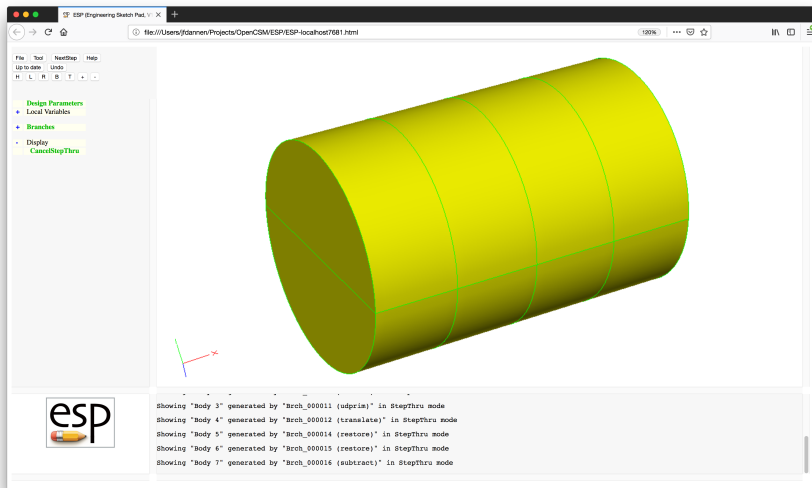
After TRANSLATING the Waffle





Waffle Example (8)

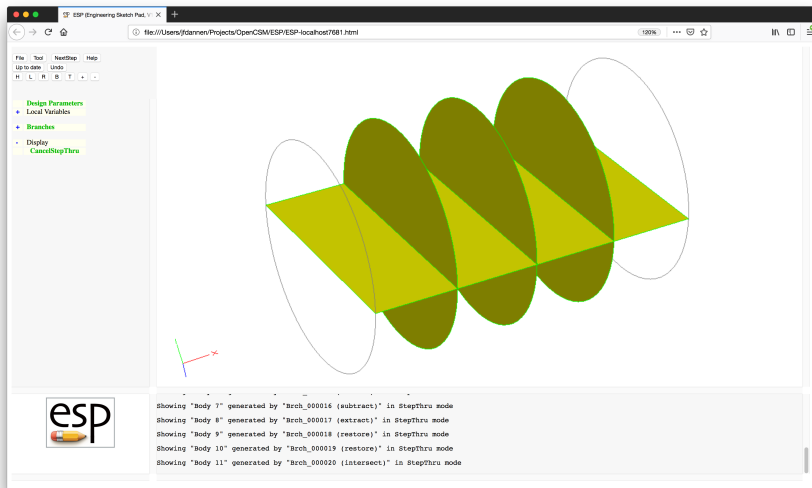
After SUBTRACTION of Waffle from SolidBody





Waffle Example (9)

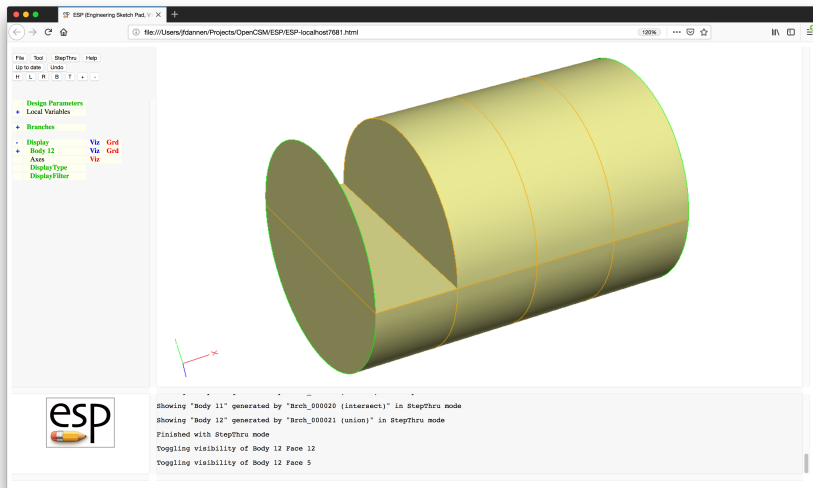
After INTERSECTION of SolidBody and Waffle

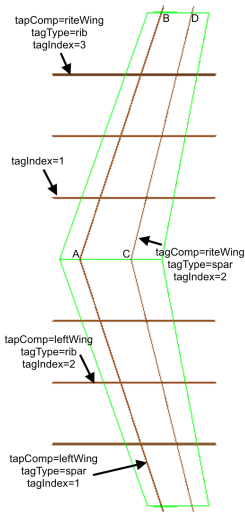




Waffle Example (10)

After UNION of scored SolidBody and interior Waffle





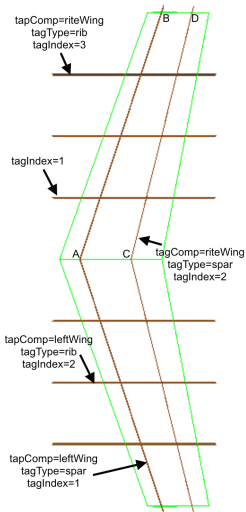
```
SET      xmin      @xmin-0.1
SET      xmax      @xmax+0.1
SET      ymin      0
SET      ymax      @ymax+0.1
SET      zmin      @zmin-0.1
SET      zmax      @zmax+0.1
STORE    .
```

```
UDPARG   waffle     depth wing:nrib      # ensures rebuild
UDPARG   waffle     depth wing:spar1
UDPARG   waffle     depth wing:spar2
UDPRIM   waffle     depth zmax-zmin filename <<
```

construction lines for spars

```
CPOINT A   AT      0+wing:spar1*croot 0
CPOINT B   AT      wing_xtip+wing:spar1*ctip wing_ytip
CPOINT C   AT      0+wing:spar2*croot 0
CPOINT D   AT      wing_xtip+wing:spar2*ctip wing_ytip
```

```
CLINE AB      A B
CLINE CD      C D
```



rite spars

POINT E ON AB YLOC ymin

POINT F ON AB YLOC ymax

LINE EF E F tagComp=riteWing tagType=spar tagIndex=1

POINT G ON CD YLOC ymin

POINT H ON CD YLOC ymax

LINE GH G H tagComp=riteWing tagType=spar tagIndex=2

rite ribs

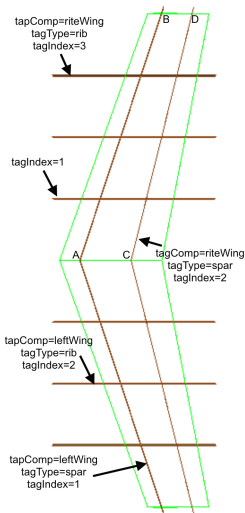
PATBEG irib wing:nrib

CPOINT I AT xmin wing_ytip*irib/(wing:nrib+1)

CPOINT J AT xmax y@I

LINE . I J tagComp=riteWing tagType=rib ...
tagIndex=!val2str(irib,0)

PATEND



```
# left spars
POINT E AT x@E -y@E
POINT F AT x@F -y@F
LINE EF E F tagComp=leftWing tagType=spar tagIndex=1

POINT G AT x@G -y@G
POINT H AT x@H -y@H
LINE GH G H tagComp=leftWing tagType=spar tagIndex=2

# left ribs
PATBEG irib wing:nrib
  CPOINT I AT xmin -wing_ytip*irib/(wing:nrib+1)
  CPOINT J AT xmax y@I
  LINE . I J tagComp=leftWing tagType=rib ...
                                tagIndex=!val2str(irib,0)

PATEND
>>
```

- Simple wing
- Simple fuselage
 - OML (outer mold line)
 - structure
- Starter files are in
`$ESP_ROOT/training/ESP/data/session03`

Generated with UDPRIM naca: thickness camber

```
# naca
```

```
UDPRIM naca thickness 0.00 camber 0.04  
TRANSLATE -2 0 0
```

```
UDPRIM naca thickness 0.12 camber 0.00
```

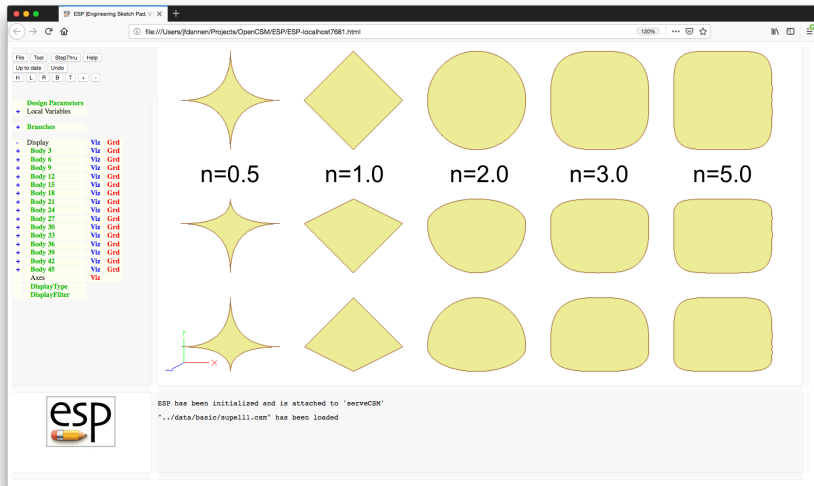
```
UDPRIM naca thickness 0.12 camber 0.04  
TRANSLATE +2 0 0
```

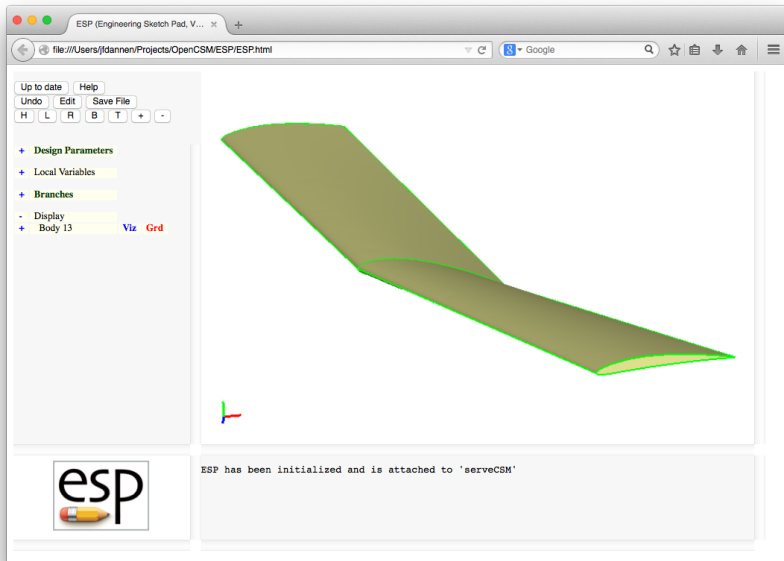
```
END
```



Generated with UDPRIM supell: rx, ry, n

Generated with \$ESP_ROOT/data/basic/supell1.csm





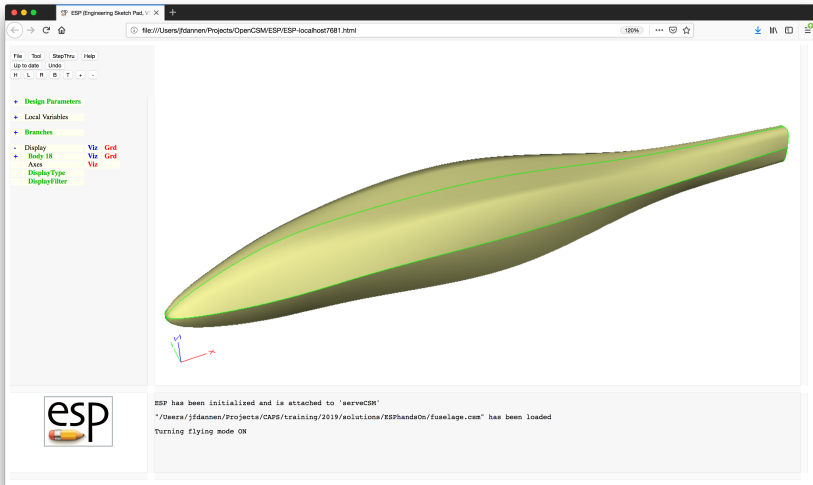
Xroot	X-coordinate of root leading edge	0.00
Yroot	Y-coordinate of root leading edge	0.00
Zroot	Z-coordinate of root leading edge	0.00
croot	chord of root	2.00
troot	thickness/chord of root	0.12
mroot	camber/chord of root	0.04
aroot	angle of attack of root (deg)	7.50
Xtip	X-coordinate of tip leading edge	0.50
Ytip	Y-coordinate of tip leading edge	0.25
Ztip	Z-coordinate of tip leading edge	8.00
ctip	chord of tip	1.75
ttip	thickness/chord of tip	0.08
mtip	camber/chord of tip	0.04
atip	angle of attack of tip (deg)	-5.00

- What happens if you switch from RULE to BLEND?
- What happens if we change the sequence of transformations from SCALE, ROTATEZ, TRANSLATE to ROTATEZ, SCALE, TRANSLATE?
- What happens if we do the TRANSLATE first?
- Could you change the Design Parameters to `area`, `aspectRatio`, `taperRatio`, `sweep`, and `twist`?

$$AR = \frac{b^2}{S} \quad S = b(c_{\text{tip}} + c_{\text{root}})/2 \quad \tau = \frac{c_{\text{tip}}}{c_{\text{root}}}$$

Simple Fuselage (1)

- Fuselage by blending a series of super-ellipses (SUPELLs), where the dimensions of the X-sections are provided in arrays



xloc	width	zcent	height	power
0.0	0.0	0.0	0.0	2
1.0	1.0	0.1	1.0	2
4.0	1.6	0.4	2.0	3
8.0	1.6	0.4	2.0	3
12.0	1.0	0.3	1.2	2
16.0	0.8	0.2	0.4	2

- Can you make the radius at the nose 0.2 in a top view and 0.1 in a side view?
- Can you make the fuselage between the two sections whose power is 3 have a constant cross-section?
- Can you create a SheetBody that has a plane of symmetry and cross-sections at every y , starting at $y = 1/2$ and spaced with $\Delta y = 1$?
- Can you color the odd-numbered bulkheads red and even-numbered bulkheads blue?
- Can you color the Edges at the intersections of the symmetry plane and bulkheads white?

