

# Engineering Sketch Pad (ESP) Training

## Session 1: Overview

John F. Dannenhoffer, III

Syracuse University

Bob Haimes

Massachusetts Institute of Technology

Revised for v1.13





# Introductions

- Dr. John Dannenhoffer
  - Syracuse University, `jfdannen@syr.edu`
- Bob Haimes
  - Massachusetts Institute of Technology, `haimes@mit.edu`
- Dr. Dean Bryson
  - Air Force Research Labs, `dean.bryson@us.af.mil`
- Dr. Nitin Bhagat
  - U. of Dayton Research Institute, `nbhagat1@udayton.edu`
- Dr. Ryan Durscher
  - Air Force Research Labs, `ryan.durscher.1@us.af.mil`
- Undergraduate helpers from Syracuse University



# Overview

- Background and objectives
- ESP architecture
- Distinguishing features
- BRep terminology
- Overview of training
- ESP Graphical User Interface (GUI)
- Hands-on exercise:
  - bottle
- Muddy cards



# Background — 1

- Over the past 40 years, there have been an increasingly-complex (complicated) series of “CAD” systems to support the geometry needs of the manufacturers of mechanical devices
  - CAD = “computer aided drafting”
  - CAD = “computer-aided drawing”
  - CAD = “computer-aided design”
  - CAD = “computer-aided development”
- “CAD” has sometimes been erroneously equated with geometry

- These systems are built around the notion that the developer of a geometric model should construct the model to be consistent with the manufacturing process (**mCAD**)
- The analytical designer of a system wants to think about the function and performance of the device being generated, often leading to the generation of a separate **aCAD** model
- The modeling techniques supported by **aCAD** and **mCAD** are often so dissimilar that model transfer between them is done by limited translators or by “starting over”
- This one-way path from **aCAD** to **mCAD** leads to a “broken process”

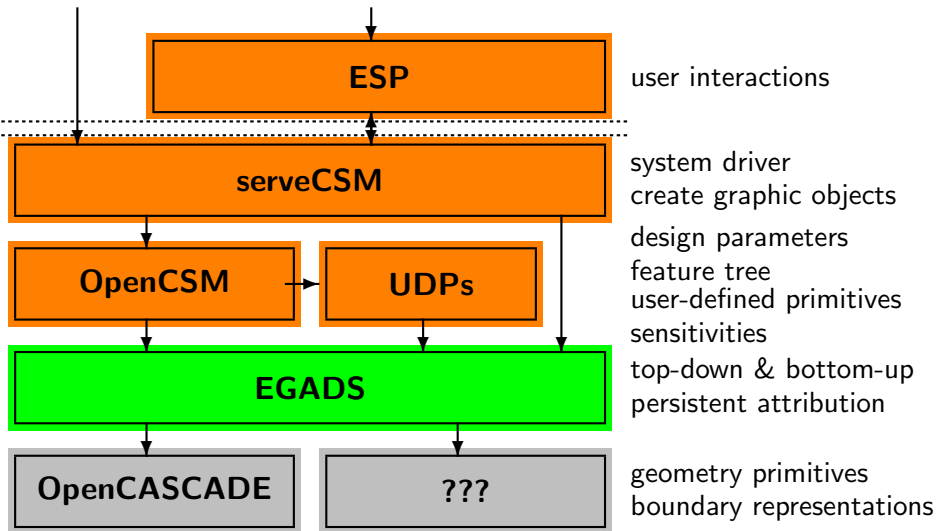


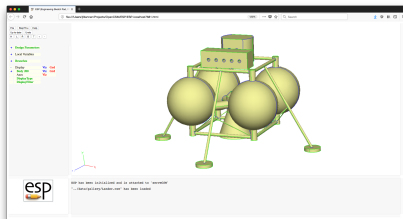
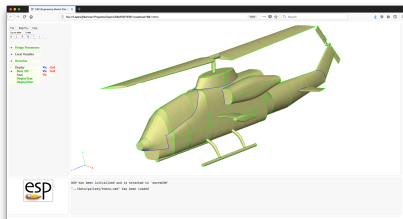
# Objective

- ESP is:
  - a geometry creation and manipulation system designed specifically to support the analysis and design of aerospace vehicles
  - can be run stand-alone for the development of models
  - can be embedded into other analysis and design systems to support their geometry needs
- ESP is not:
  - a full-featured computer-aided design (CAD) system designed specifically to support the mechanical design and manufacturing of any complex system
  - a system to be used for creating “drawings”



# ESP Architecture







# Distinguishing Features — Solid Modeller

- Construction process guarantees that models are realizable solids
  - watertight representation needed for grid generators
  - sheets and wires are supported when needed
- Parametric models are defined in terms of:
  - Feature Tree
    - “recipe” for how to construct the configuration
  - Design Parameters
    - “values” that describe any particular instance of the configuration



# Distinguishing Features — Feature-based

- Configurations start with the generation of primitives
  - standard primitives: box, sphere, cone, cylinder, torus
  - grown primitives (from sketches): extrude, rule, blend, revolve, sweep, loft
  - user-defined primitives (UDPs)
- Bodys can be modified
  - transformations: translate, rotate, scale, mirror
  - applications: fillet, chamfer, hollow
- Bodys can be combined
  - Booleans: intersect, subtract, union
  - other: join, connect, extract, combine



# Construction Process (1)

```
# bolt example

# design parameters
1:  DESPMTR   Thead    1.00  # thickness of head
2:  DESPMTR   Whead    3.00  # width   of head
3:  DESPMTR   Fhead    0.50  # fraction of head that is flat

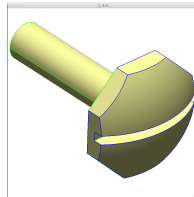
4:  DESPMTR   Dslot    0.75  # depth of slot
5:  DESPMTR   Wslot    0.25  # width of slot

6:  DESPMTR   Lshaft   4.00  # length  of shaft
7:  DESPMTR   Dshaft   1.00  # diameter of shaft

8:  DESPMTR   sfact    0.50  # overall scale factor

# head
9:  BOX       0        -Whead/2 -Whead/2  Thead   Whead   Whead
10: ROTATEX   90  0  0
11: BOX       0        -Whead/2 -Whead/2  Thead   Whead   Whead
12: ROTATEX   45  0  0
13: INTERSECT

...
```



```

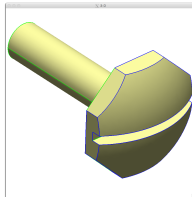
...
14:  SET      Rhead  (Whead^2/4+(1-Fhead)^2*Thead^2)/(2*Thead*(1-Fhead))
15:  SPHERE    0      0  0  Rhead
16:  TRANSLATE Thead-Rhead  0  0
17:  INTERSECT

# slot
18:  BOX      Thead-Dslot  -Wslot/2  -Whead  2*Thead  Wslot  2*Whead
19:  SUBTRACT

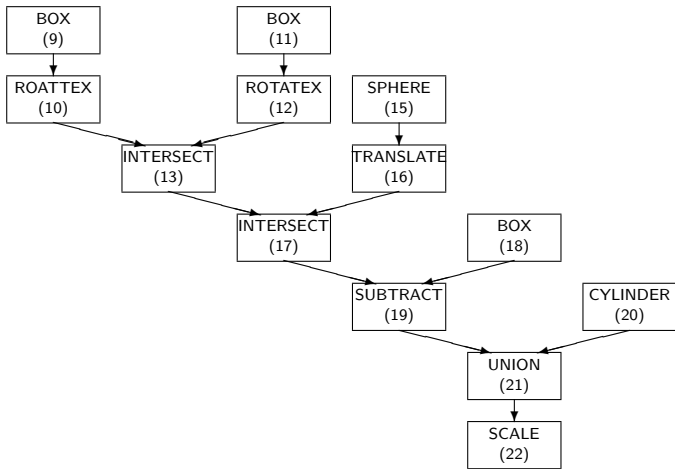
# shaft
20:  CYLINDER -Lshaft  0  0  0  0  0  Dshaft/2
21:  UNION

22:  SCALE    sfact
23:  END

```



# esp Review of Construction Process (3)



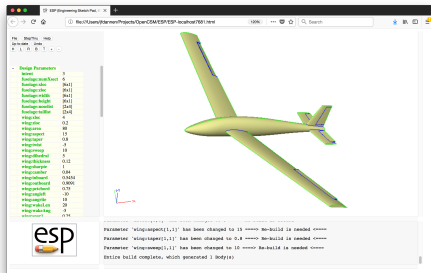


## Distinguishing Features — Parametric

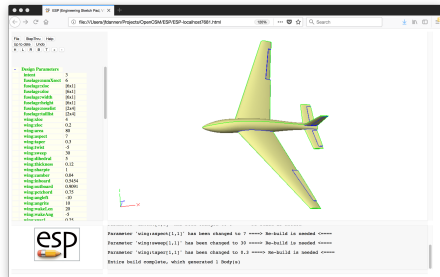
- ESP models typically contain one or more Design Parameters
- Design Parameters can be single-valued, 1D vectors, or 2D arrays of numbers
- Each Design Parameter has a current value, upper- and lower-bounds, and a current “velocity” (which is used to define sensitivities)
- Design Parameters can be “set” and “get”
  - through ESP’s tree window
  - externally via calls to the Application Programming Interface (API)
- Arguments of all operations can be written as “expressions” that reference Design Parameters



# Parameter Changes for Glider



aspect = 15  
sweep = 10  
taper = 0.8



aspect = 7  
sweep = 30  
taper = 0.3

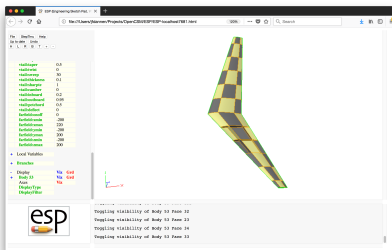
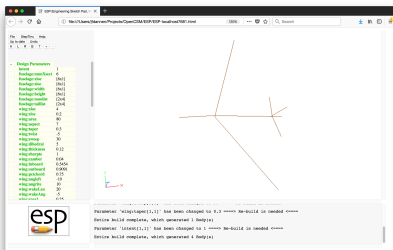
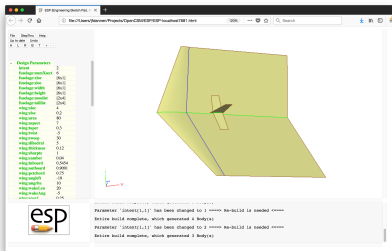
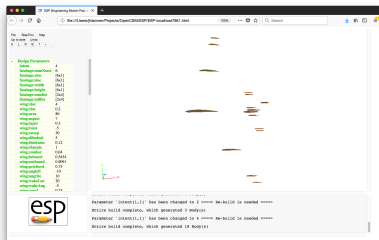


## Distinguishing Features — Associative

- ESP maintains a set of global and local attributes on a configuration that are persistent through rebuilds
- Supports the generation of multi-fidelity models
  - attributes can be used to associate conceptually-similar parts in the various models
- Supports the generation of multi-disciplinary models
  - attributes can be used to associate surface groups which share common loads and displacements
- Supports the “marking” of Faces and Edges with attributes such as nominal grid spacings, material properties, ...



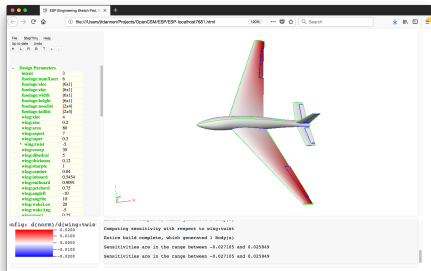
# Multiple Models for Glider



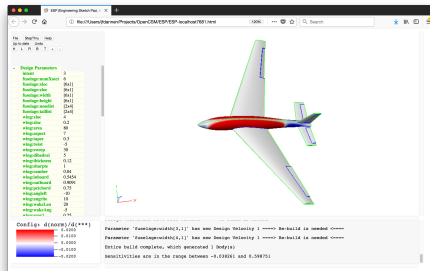


## Distinguishing Features — Differentiated

- ESP allows a user to compute the sensitivity of any part of a configuration with respect to any Design Parameter
- Many of OpenCSM's commands have been analytically “differentiated”
  - efficient, since there is no need to re-generate the configuration
  - accurate, since there is no truncation error associated with “differencing”
- Other commands (currently) require the use of finite-differenced sensitivities
  - robust, due to new mapping technique
  - less efficient, since it requires the generation of a “perturbed” configuration
  - less accurate, since one needs to carefully select a “perturbation step” that is a balance between truncation and round-off errors



twist



fuselage width



# Distinguishing Features — Extensible

- Users can add their own user-defined primitives (UDPs)
  - create a single primitive solid
  - are written in C, C++, or FORTRAN and are compiled
  - can be written either top-down or bottom-up
  - have access to the entire suite of methods provided by EGADS
  - are coupled into ESP dynamically at run time
- Users can add their own user-defined functions (UDFs)
  - consume one or more Bodys from stack
  - are otherwise similar to UDPs
- Users can add their own user-defined components (UDCs)
  - can be thought of as “macros”
  - create zero or more Bodys
  - are written as .csm-type scripts



## Distinguishing Features — Deployable

- ESP's back-end (server) runs on a wide variety of modern compute platforms
  - LINUX
  - OSX
  - Windows
- ESP's user-interface (client) runs in most modern web browsers
  - FireFox
  - Google Chrome
  - Safari
  - Note: Internet Explorer is not supported at this time
- ESP can be distributed anywhere in the computer environment
  - open-source project (using the LGPL 2.1 license) that is distributed as source



# Distinguishing Features — Embeddable

- Models are defined in `.csm` files
  - human readable ASCII
  - stack-like language that is consistent with Feature Tree traversal
  - contains looping via “patterns”
  - contains logical (if/then) constructs
  - contains error recovery via thrown/caught signals
- OpenCSM modeling system is defined by an Application Programming Interface (API) that allows it to be embedded into other applications
  - load a Master Model
  - interrogate and/or edit the Master Model
  - execute the Feature Tree and create BRep(s)
  - interrogate the BRep(s)
  - “set” and “get” sensitivities



# Boundary Representation – BRep

<div>Top Down</div> <div>↓</div> <div>↑</div> <div>Bottom Up</div>	Topological Entity	Geometric Entity	Function
	Model		
	Body	Solid, Sheet, Wire	
	Shell		
	Face	<b>surface</b>	$(x, y, z) = \mathbf{f}(u, v)$
	Loop		
	Edge	<b>curve</b>	$(x, y, z) = \mathbf{g}(t)$
	Node	<b>point</b>	

- *Solids* are open at machine precision – tolerances
  - Node points that bound Edges may not be on the curve
  - Edge curves that bound the Faces (through Loops) may not be on the underlying surface



# Overview of Training

- The training is divided into a series of sessions that cover the best practices for using ESP
- Each session starts with a lecture in which the basic ideas are discussed
- Each session then contains a series of hands-on exercises, designed to reinforce the basic ideas



# Training Sessions

- 1 ESP Overview
- 2 Sketching fundamentals
- 3 Solids fundamentals
- 4 CSM scripts
- 5 Using UDPs, UDFs, and UDCs
- 6 Sensitivities
- 7 Integration with other systems
- 8 Putting it all together
- 9 Writing a UDP or UDF (optional)



# Launching ESP

- For interactive use, ESP requires both a server (serveCSM) and a client/browser (ESP.html)
- Technique 1: start browser automatically:

```
setenv ESP_START "open -a /Applications/Firefox.app ../ESP/ESP.html"
```

or

```
export ESP_START="open -a /Applications/Firefox.app ../ESP/ESP.html"
```

or

```
set ESP_START="open -a /Applications/Firefox.app ../ESP/ESP.html"
```

and then

```
serveCSM ../data/tutorial1
```

- Technique 2: start browser separately:

```
serveCSM ../data/tutorial1
```

and then open a browser on ESP.html



# ESP Command Line (1)

- To start serveCSM

```
serveCSM [filename[.csm]] [options...]
```

where [options...] include:

- filename is the name of the .csm file that contains the Model
- -batch runs the case but does not attach to a browser
- -help or -h prints listing of acceptable options
- -jrnl jrnlname can be used to replay a previous session
  - current session is stored in file portXXXX.jrnl
  - file must be renamed to be used for next session
- -outLevel n selects the output level (1 is the default)
- --version or -version or -v to return version information
- ...



## ESP Command Line (2)

- Other [options...] include:
  - `-dict dictname` loads constants defined in dictionary file
  - `-dumpEgads` to dump EGADS file in form `Body_XXXXXX.egads` after each Body is built
  - `-loadEgads` to load EGADS file if it exists in current directory
  - `-port portnum` selects the port for communication with the browser (7681 is the default)
  - `-verify` to execute ASSERT statements that contain `verify=1`



## ESP Command Line (3)

- Other (less frequently used) [options...] include:
  - `-addVerify` to create `.csm_verify` file that contains verification assertions
  - `-egg eggname` uses an external grid generator
  - `-port portnum` selects the port for communication with the browser (7681 is the default)
  - `-plotCP` to plot B-spline control points
  - `-printBboxs` prints bounding box info for each Node, Edge, and Face in each Body on the stack
  - `-printBreps` prints BRep info to server window
  - `-sensTess` to produce configuration sensitivity (instead of tessellation sensitivity) output





# Image Manipulation via the Mouse

- Translation
  - press and drag any mouse button
- Rotation
  - hold down **Ctrl** and drag any mouse button
  - hold down **Alt** and drag any mouse button
- Zoom
  - hold down **Shift** and drag any mouse button
  - scrolling the middle mouse button also scrolls in/out
- Flying mode
  - press **!** in Graphics window to toggle mode
  - image continues moving image until mouse is released



# Image Manipulation via Key Presses (1)

“flying-mode” is off by default

Key-press	“flying-mode” off	“flying-mode” on
←	rotate left 30°	translate left
→	rotate right 30°	translate right
↑	rotate up 30°	translate up
↓	rotate down 30°	translate down
+	zoom in	zoom in
-	zoom out	zoom out
<b>PgUp</b>	zoom in	zoom in
<b>PgDn</b>	zoom out	zoom out
<b>Home</b>	home view	home view

Note: holding **Shift** reduces the increment



## Image Manipulation by Key Presses (2)

Key-press	orientation	note
<b>Ctrl-h</b>	home view	$y$ vs $x$
<b>Ctrl-f</b>	front view	$y$ vs $x$
<b>Ctrl-l</b>	left side view	$y$ vs $z$
<b>Ctrl-r</b>	right side view	$y$ vs $-z$
<b>Ctrl-b</b>	bottom view	$z$ vs $x$
<b>Ctrl-t</b>	top view	$-z$ vs $x$
<b>Ctrl-i</b>	zoom in	
<b>Ctrl-o</b>	zoom out	



# Image Manipulation via Buttons

Button press	orientation	note
<b>H</b>	home view	$y$ vs $x$
<b>L</b>	left side view	$y$ vs $z$
<b>R</b>	right side view	$y$ vs $-z$
<b>B</b>	bottom view	$z$ vs $x$
<b>T</b>	top view	$-z$ vs $x$
<b>+</b>	zoom in	
<b>-</b>	zoom out	

Button are near top of Tree window



## Image Manipulation via the Tree Window

- In the Tree window, **Display** contains an entry for each Body
- If the **Body** is expanded (the + on the left is pressed), then entries appear for **Faces** and **Edges**
- If the **Faces** and/or **Edges** are expanded, the names of all Faces and/or Edges are listed
- **Viz** toggles the visibility of the associated Body(s), Face(s), or Edge(s)
- **Grd** toggles the visibility of the grid of the associated Body(s), Face(s), or Edge(s)
- **Trn** toggles the pseudo-transparency of the the associated Face(s)
- **Ori** toggles the orientation vectors of the associated Edge(s)
- Toggling at a “group” level effects the setting of its children



# Image Inquiry

- Re-center the image at the current location and set a new “rotation center”
  - \* or 8
- Find the location of the cursor (in 3D space) and report it in the Messages window
  - @ or 2
- Identify the object (Edge or Face) and list all its attributes in the Messages window
  - ^ or 6
- List the key-press options in the Messages window
  - ?
- Orientation of image in Graphics window
  - red axis in  $x$ -direction
  - green axis in  $y$ -direction
  - blue axis in  $z$ -direction



# Image Manipulation

- Turn off the visibility of the Node, Edge, or Face at cursor
  - **v**
- Toggle the grid on the Edge or Face at cursor
  - **g**
- Toggle the transparency of the Edge or Face at cursor
  - **t**
- Toggle the orientation of the Edge at cursor
  - **o**



# StepThru Mode

- Show step-by-step build process
  - **StepThru** button (near top of Tree Window)
- Next step in build process
  - **NextStep** button (near top of Tree Window) or **n** key in Graphics Window
- Previous step in build process
  - **p** key in Graphics Window
- Exit StepThru mode
  - **CancelStepThru** at bottom of Display listing in Tree Window

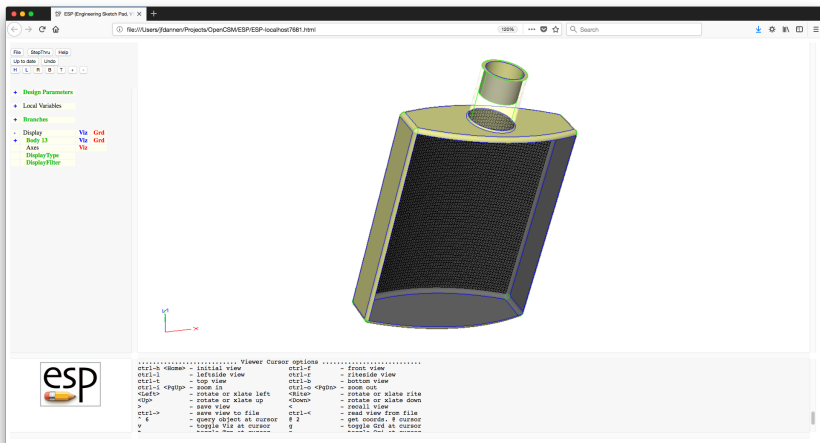


# Hands-on exercise

- 1 Start serveCSM using the file `bottle.csm`
- 2 Explore the various image manipulation tools
- 3 See if you can get the image on the next page
- 4 Use StepThru to see how the bottle was created



# bottle After Image Manipulations





# Muddy Cards

- Opportunity to provide immediate “feedback”
- Any questions about presentation material, critique of sample problems, ...
- Questions will be answered at next session