

Engineering Sketch Pad (ESP) Training

Session 1: Overview

John F. Dannenhoffer, III

Syracuse University

Bob Haimes

Massachusetts Institute of Technology

Revised for v1.08





Special Notice about Training

- These training slides are an updated version of the slides that were used at the first ESP training class
 - Tuesday–Friday, 21–24 July 2015
 - Russ Engineering Center, Room 141 at Wright State University
 - special thanks to Prof. Ramana Grandhi for the use of the facilities and his generous hospitality
- The updates to the slides were made to include:
 - errors discovered during the training
 - additions to ESP v1.08 that were made as a direct consequence of the Muddy Cards submitted during the course
- This training material is to be used in conjunction with the ESP tutorials
 - tutorials contain step-by-step instructions that are not included in these training slides
 - accessed via **Help** button in ESP



Introductions

- Dr. John Dannenhoffer
 - Syracuse University, `jfdannen@syr.edu`
- Bob Haimes
 - Massachusetts Institute of Technology, `haimes@mit.edu`
- Dr. Edward Alyanak
 - Air Force Research Labs, `edward.alyanak.1@us.af.mil`
- Dr. Nitin Bhagat
 - Wright State University, `nitin.bhagat@wright.edu`
- Dr. Darcy Allison
 - Optimal Flight Sciences, `darcy.allison.ctr@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”

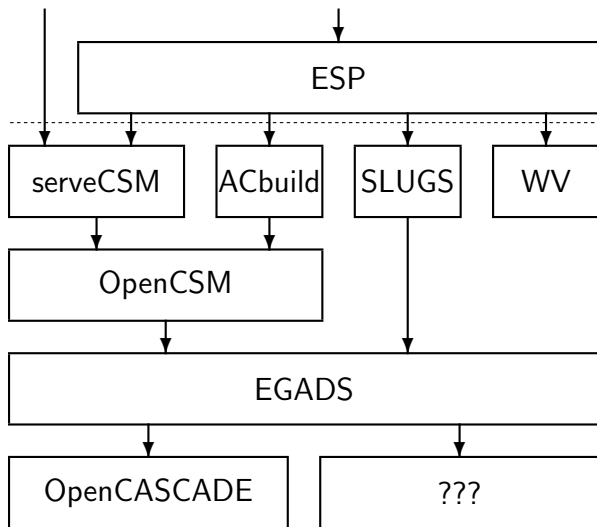


Objective

- 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 build 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



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



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, ...



Distinguishing Features — Differentiated (1)

- 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”



Distinguishing Features — Differentiated (2)

- Other commands (currently) require the use of finite-differenced sensitivities
 - less efficient, since it requires the generation of a “perturbed” configuration
 - robust, since a new “mapping” technique guarantees the correct association of points in the baseline and perturbed geometries
 - less accurate, since one needs to carefully select a “perturbation step” that is a balance between truncation and round-off errors



Distinguishing Features — Extensible

- Users can add their own user-defined primitives (UDPs)
 - UDPs create a single primitive solid
 - UDPs are written in C, C++, or FORTRAN and are compiled
 - UDPs can be written either top-down or bottom-up
 - UDPs have access to the entire suite of methods provided by EGADS
 - UDPs are coupled into ESP dynamically at run time
- Users can add their own user-defined components (UDCs)
 - UDCs can be thought of as “macros”
 - UDCs create zero or more Bodys
 - UDCs 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: IE 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 is consistent with Feature Trees
 - contains looping and logical decisions via “patterns”
- 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 a BRep
 - interrogate the BRep
 - “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	surface	$(x, y, z) = \mathbf{f}(u, v)$
	Loop		
	Edge	curve	$(x, y, z) = \mathbf{g}(t)$
	Node	point	

- *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 Expressions and patterns
- 5 Using UDPs and UDCs
- 6 Sensitivities
- 7 Putting it all together
- 8 Writing a UDP



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



■ To start serveCSM

```
serveCSM filename [-port portnum] [-outLevel n] [-jrnl jrnlname] [-batch]
```

- filename is the name of the .csm file that contains the Model
- -port portnum selects the port for communication with the browser (7681 is the default)
- -outLevel n selects the output level (1 is the default)
- -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
- -batch runs the case but does not attach to a browser



ESP Screen Layout

- Graphics window
 - 3D image
 - 2D sketcher
 - forms
- Tree window
 - Design Parameters
 - Local Variables
 - Branches
 - Display
- Key window
 - color key
- Messages window

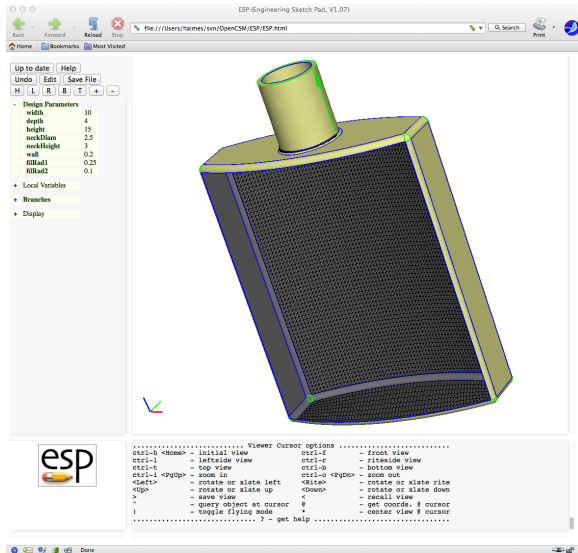




Image Manipulation via the Mouse

- 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
PgUp	zoom in	zoom in
PgDn	zoom out	zoom out
Home	home view	home view

Note: holding **Shift** reduces the increment



Image Manipulation by Key Presses (2)

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



Image Manipulation via Buttons

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

Button are on third row 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”
 - *
- Find the location of the cursor (in 3D space) and report it in the Messages window
 - @
- Identify the object (Edge or Face) and list all its attributes in the Messages window
 - ^
- 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



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

esp bottle After Image Manipulations

ESP (Engineering Sketch Pad, V1.07)

file:///Users/haimes/svn/OpenCSM/ESP/ESP.html

Back Forward Reload Stop

Home Bookmarks Most Visited

Up to date Help

Undo Edit Save File

H L R B T + -

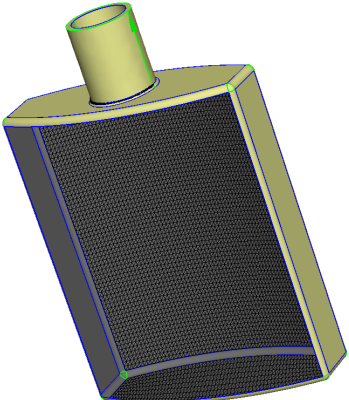
Design Parameters

- width 10
- depth 4
- height 15
- neckDiam 2.5
- neckHeight 3
- wall 0.2
- filRad1 0.25
- filRad2 0.1

Local Variables

Branches

Display



esp

Viewer Cursor options

ctrl-h <Home>	- initial view	ctrl-f	- front view
ctrl-l	- leftside view	ctrl-r	- rite side view
ctrl-t	- top view	ctrl-b	- bottom view
ctrl-i <PgUp>	- zoom in	ctrl-o <PgDn>	- zoom out
<Left>	- rotate or xlate left	<Rite>	- rotate or xlate rite
<Up>	- rotate or xlate up	<Down>	- rotate or xlate down
>	- save view	<	- recall view
?	- query object at cursor	#	- get coords. # cursor
!	- toggle flying mode	*	- center view # cursor
? - get help			



Muddy Cards

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