The Engineering Sketch Pad: A Solid-Modeling, Feature-Based, Web-Enabled System for Building Parametric Geometry

Robert Haimes
haimes@mit.edu
Aerospace Computational Design Lab
Massachusetts Institute of Technology

John F. Dannenhoffer, III
jfdannen@syr.edu
Aerospace Computational Methods Lab
Syracuse University

21st AIAA Computational Fluid Dynamics Conference
CFD-38 Grid Generation and Effects of Grid Quality II

Overview

- Background / Objective
- System Architecture
- Software Assemblage & Components
    - Engineering Sketch Pad (ESP)
    - WebViewer
        - Browser JavaScript API
        - Server Requirements
    - OpenCSM (AIAA Paper 2013-0701)
    - EGADS (AIAA Paper 2012-0683)
- Conclusions

Background / Objective

Background

- Need in MDAO community to visualize and interact with 3D configurations
- A geometric tool is needed to provide multi-fidelity (*analysis aware*) configurations that can be used throughout the early phases of design
- Increasing use of browser-based tools in engineering community

Objective

- To build a solid-modeling, feature-based, web-enabled system for constructing & modifying parametric geometry

System Architecture

The Engineering Sketch Pad

Web Server
Acts like an Apache server

OpenCSM
Drives Geometric Builds

EGADS
Geometry Kernel

OpenCASCADE

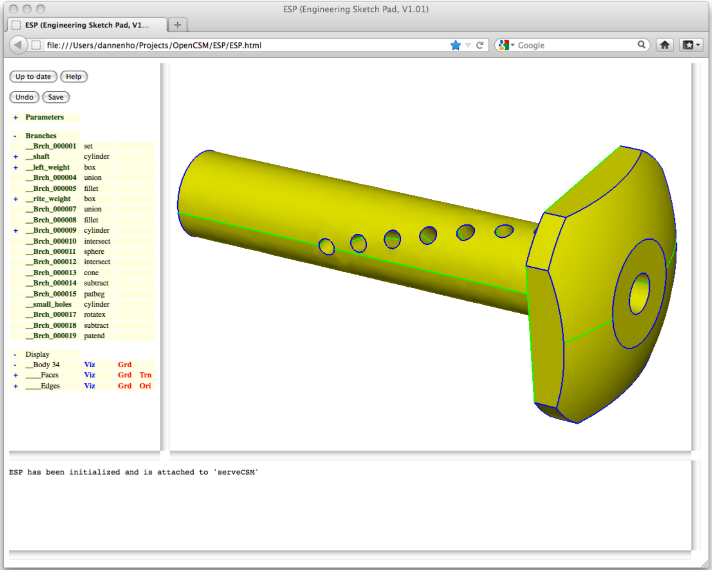
Browser
FireFox, Chrome, Safari

- JavaScript
- WebSockets
- WebGL

WebViewer
3D Viewer JavaScript API

Customized UI
JavaScript, CSS & HTML

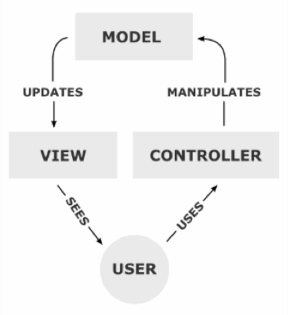
ESP in a Web Browser



Engineering Sketch Pad

- Operates with (almost) any modern browser
- Modification of regeneration values
  - design parameters
  - feature suppression / selection
- Modification of feature tree
  - add/delete/modify branches
- 3-D graphical representation
  - view transformation, picking, locating
- Control of scene properties
  - visibility, mesh, orientation, transparency

MVC



Model is in the server,
View in the Browser,
Controller implemented in both

WebViewer Goals

Potential applications (in an MDAO setting)

- geometry construction
- meshing
- scientific visualization
- multidimensional design space exploration

Uses a web browser, which has the following advantages:

- computer platform independence
- potentially no software installation & no browser plugins needed
- client-server architecture (SOA)
- tablet enabled
- cloud ready
- geographically-dispersed collaboration

HTML5 standard message-passing JavaScript API

- provides full-duplex communications over a single TCP/IP connection
- low packet overhead
- can specify multiple **protocols** – (like MPI communicators)
- messages can be *text* or *binary*
- supports JavaScript *typed* arrays for *binary*

Using WebViewer:

- browser-to-server *text* stream (for **Controller**)
- server-to-browser *text* stream (for **Controller**)
- server-to-browser *binary* stream (for **View**)

OpenGL-like rendering JavaScript API

- based on OpenGL ES 2.0
- small & simple API
- driven by *Vertex Buffer Objects* (VBOs)
  - no individual vertex-based calls
  - layout can be constructed in the server
  - high performance
- no lighting-model or material properties
  - GLSL-like Vertex and Fragment shaders required
  - direct access to GPU but at the cost of programming
- high performance rendering – little JavaScript involvement

Client-side

- completely asynchronous
- scene driven by VBOs received from the server-side
- supplied shaders support:
  - two-sided lighting
  - ambient & diffuse lighting model
  - back-face coloring
  - constant and/or linearly interpolated color-space mapping
  - simple transparency
  - picking
  - bumping of lines forward (in screen Z)
- custom UI via WebViewer specific JavaScript call-backs
- stateless except for view transformation and plotting attributes

Server-side

- handles the **Model** and part of the **Controller** in the MCV paradigm
- maintains state
- constructs VBOs so that the browser does not have to interpret the potentially voluminous data for the **View**
- must communicate via specific WebSocket **protocols**
    - Python has many WebSocket packages
    - *libwebsockets* open source project for C/C++ programming which is used for ESP

OpenCSM

An Open-Source Constructive Solid Modeler

- API accessible
- feature-based parametric solid modeler
  - *feature tree*
  - suite of parameters (driving and driven)
- full suite of standard primitives, boolean operators and transformations
- user-friendly sketcher
- user-defined primitives/features
- configuration files that are readable ASCII text

OpenCSM

csm File Description

- ASCII file that contains build recipe that is executed in a stack-like way
- all arguments are MATLAB-like expressions
- primitives: box, cylinder, cone, sphere, torus
- grown bodies: extrude, loft, revolve, sweep
- user-defined primitives: ellipse, freeform solid, NACA airfoil, …
- applied features: fillet, chamfer, offset, hollow
- boolean operators: union, difference, intersection
- sketches: lines, circular arcs, splines, constraints
- transformations and utilities: translate, rotate, scale, patterns, macros

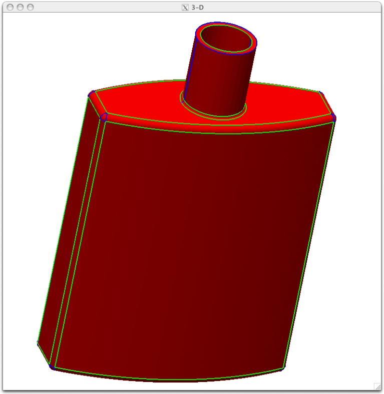
OpenCSM Example

```
# design parameters
despmtr   width        10.00
despmtr   depth         4.00
despmtr   height       15.00
despmtr   neckDiam      2.50
despmtr   neckHeight    3.00
despmtr   wall          0.20
despmtr   filRad1       0.25
despmtr   filRad2       0.10

# basic bottle shape (filleted)
set         baseHt   height-neckHeight
skbeg       -width/2 -depth/4 0
   cirarc   0        -depth/2 0  +width/2 -depth/4  0
   linseg   +width/2 +depth/4 0
   cirarc   0        +depth/2 0  -width/2 +depth/4  0
skend
extrude   0        0        baseHt
fillet    filRad1  0        0

# neck with a hole
set       holeBot height-neckHeight/2
cylinder 0   0   baseHt  0 0 height      neckDiam/2
cylinder 0   0   holeBot 0 0 height+wall neckDiam/2-wall
subtract

# join the neck to the bottle and apply a fillet at the union
union
fillet    filRad2  0        0
```



EGADS

Engineering Geometry Aircraft Design System

- compact object-based API to OpenCASCADE
  - $\sim$ 60 C/C++/FORTRAN functions from about $17,000$ OpenCASCADE C++ methods
- supports the following:
  - manifold and non-manifold Boundary Representations
  - bottom-up construction
    - node, curve, edge, surface, loop, face, shell, body
  - top-down construction
    - primitives, boolean operators, transformations
    - lofts, revolves, fillets, and more
  - attribution that is maintained through regenerations
- Can read and write IGES, STEP, and native file formats

Conclusions

Why ESP and why not CAD?

- *design intent* in a simple, readable ASCII file
- easy to add primitives/features
- can support multi-fidelity geometry, for example:
  - beam, BEM and/or fully expressed solids
  - mid-surface aero and/or OML
- attribution can be used to set material properties/boundary conditions for automation
- HPC friendly (no licensing issues)
- parametric sensitivities
- everything is an API; easy integration into larger processes

Conclusions – Another Assemblage

OpenMDAO Framework

GEM (Geometry Environment for MDAO)

OpenCSM

EGADS

OpenCASCADE

diamond

CAPRI

CAD Systems

Catia, Pro/ENGINEER, UG NX, SolidWorks

Geometry Kernels

quartz

Conclusions

Engineering Sketch Pad:

- an open-source feature-based solid modeler
  - layered upon OpenCASCADE
- is web-enabled
  - uses (almost) any modern Web browser
  - built upon WebSockets and WebGL
- is freely-available for download at:

http://acdl.mit.edu/ESP

Acknowledgements

This work was supported by a NASA Fundamental Aero NRA
Christopher Heath, Technical Monitor