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

- 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
- 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
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
WebSockets

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
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
<table>
<thead>
<tr>
<th>csm File Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASCII file that contains build recipe that is executed in a stack-like way</td>
</tr>
<tr>
<td>all arguments are MATLAB-like expressions</td>
</tr>
<tr>
<td>primitives: box, cylinder, cone, sphere, torus</td>
</tr>
<tr>
<td>grown bodies: extrude, loft, revolve, sweep</td>
</tr>
<tr>
<td>user-defined primitives: ellipse, freeform solid, NACA airfoil, …</td>
</tr>
<tr>
<td>applied features: fillet, chamfer, offset, hollow</td>
</tr>
<tr>
<td>boolean operators: union, difference, intersection</td>
</tr>
<tr>
<td>sketches: lines, circular arcs, splines, constraints</td>
</tr>
<tr>
<td>transformations and utilities: translate, rotate, scale, patterns, macros</td>
</tr>
</tbody>
</table>
# 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
  - ~ 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
<table>
<thead>
<tr>
<th>Why ESP and why not CAD?</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>design intent</em> in a simple, readable ASCII file</td>
</tr>
<tr>
<td>easy to add primitives/features</td>
</tr>
<tr>
<td>can support multi-fidelity geometry, for example:</td>
</tr>
<tr>
<td>beam, BEM and/or fully expressed solids</td>
</tr>
<tr>
<td>mid-surface aero and/or OML</td>
</tr>
<tr>
<td>attribution can be used to set material properties/boundary conditions for automation</td>
</tr>
<tr>
<td>HPC friendly (no licensing issues)</td>
</tr>
<tr>
<td>parametric sensitivities</td>
</tr>
<tr>
<td>everything is an API; easy integration into larger processes</td>
</tr>
</tbody>
</table>
Conclusions – Another Assemblage

**OpenMDAO Framework**

- GEM (Geometry Environment for MDAO)
- OpenCSM
- EGADS
- OpenCASCADE

**CAD Systems**
- Catia, Pro/ENGINEER, UG NX, SolidWorks

**Geometry Kernels**

- Haimes & Dannenhoffer
- Engineering Sketch Pad

**Supporting Tools**

- **diamond**
- **quartz**

**Date**

- 26 June 2013
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