# Engineering Sketch Pad (ESP) Training Session 7: Integration with Other Systems

John F. Dannenhoffer, III

Syracuse University

Bob Haimes

Massachusetts Institute of Technology

Revised for v1.13

# Overview

- OpenCSM data types
- Programming interface (API)
- Sample program
- Hand-on exercises
  - Add an Attribute
  - Find the length of each Edge

# Basic Data Types in OpenCSM

- MODL
- Design parameters (and values)
- Branches (and arguments)
- Boundary representation
- Bodys
- Attributes and csystems

# Basic Data Types in OpenCSM

## MODL

- MODL
  - Container that holds all OpenCSM information
    - Parameters
    - Branches
    - Bodys
  - Session can contain more than one MODL

# Basic Data Types in OpenCSM

## Design parameters (and values)

- Design Parameter
  - name
  - can be single value, 1-D vector, or 2-D matrix of values or a string
  - if numeric, can have lower and upper bounds
- Value
  - current value(s)
  - current velocity(s)

# Basic Data Types in OpenCSM

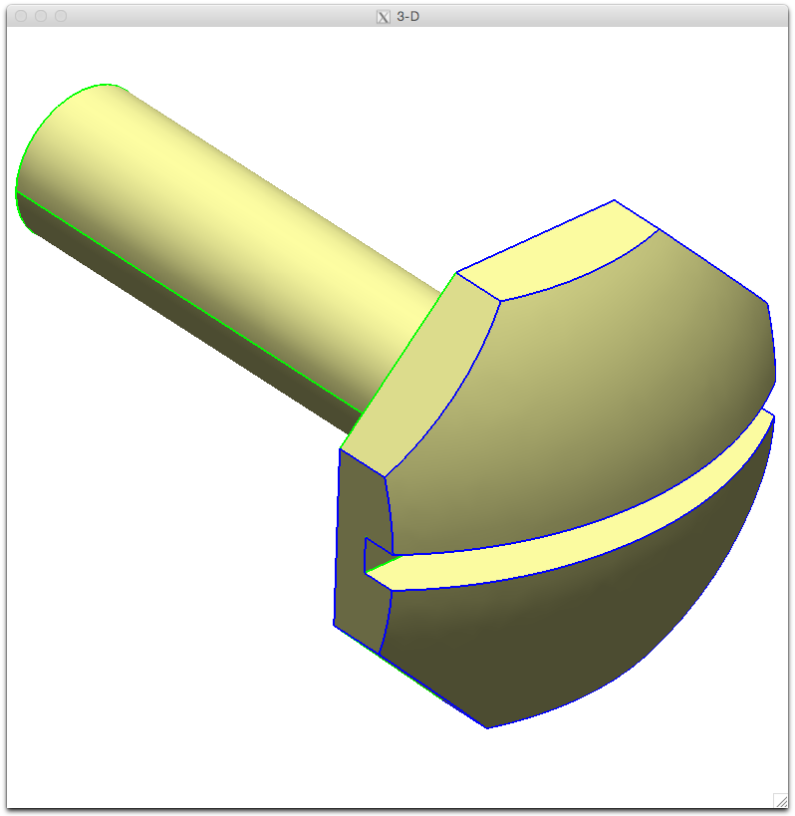
## Branches (and arguments)

- Branch
  - step in the build recipe (feature tree)
  - has a name (by default `Brch_xxxxxx`)
  - has an activity (`active`, `suppressed`, or `inactive`)
  - has zero, one, or two parent Branches
  - has zero or one child Branch
  - can be attributed
- Arguments
  - number of arguments specified by statement type
  - optional arguments are at end of argument list
  - arguments are general expressions

# Basic Data Types in OpenCSM

## Boundary representation

- Node
  - 3-D location in space
- Edge
  - path (curve) in 3-D space
  - bounded by Nodes
- Face
  - surface in 3-D space
  - bounded by Edges



# Basic Data Types in OpenCSM

## Bodys

- SolidBody
  - closed volume in 3-D space
  - can contain holes (voids)
  - composed of Nodes, Edges, and Faces
- SheetBody
  - connected set of Faces
  - composed of Nodes, Edges, and Faces
- WireBody
  - connected set of Edges
  - at most 2 Edges can meet at a Node
  - composed of Nodes and Edges
- NodeBody
  - composed of a single Node

# Basic Data Types in OpenCSM

## Attributes and csystems

- Attributes & Csystem
  - associated with a Body, Face, Edge, or Node
  - has name
  - persists through rebuilds
- Attributes
  - represented by semicolon-separeted list of expressions or a string
- Csystem
  - contains a base point and a direction
  - can be used in APPLYCSYS transformation

## High-level routines

- `ocsmLoad` — create a MODL by reading a .csm file
- `ocsmCheck` — check that Branches are properly ordered
- `ocsmBuild` — build Bodys by executing the MODL up to a given Branch

- `ocsmSave` — save a MODL to a file
- `ocsmCopy` — copy a MODL
- `ocsmFree` — free up all storage associated with a MODL
- `ocsmInfo` — get info about a MODL

## Branch (and argument) management

- ocsmNewBrch — create a new Branch
- ocsmGetBrch — get info about a Branch
- ocsmSetBrch — set activity for a Branch
- ocsmDelBrch — delete a Branch (or whole Sketch if SKBEG)
- ocsmGetName — get the name of a Branch
- ocsmSetName — set the name for a Branch
- ocsmPrintBrchs — print Branches to file

- ocsmGetArg — get an Argument for a Branch
- ocsmSetArg — set an Argument for a Branch

- `ocsmRetAttr` — return an Attribute for a Branch by index
- `ocsmGetAttr` — get an Attribute for a Branch by name
- `ocsmSetAttr` — set an Attribute for a Branch

- `ocsmRetCsys` — return a Csystem for a Branch by index
- `ocsmGetCsys` — get a Csystem for a Branch by name
- `ocsmSetCsys` — set a Csystem for a Branch

- `ocsmPrintAttrs` — print global Attributes to file

- ocsmNewPmtr — create a new Parameter
- ocsmDelPmtr — delete a Parameter
- ocsmFindPmtr — find (or create) a Parameter
- ocsmGetPmtr — get info about a Parameter
- ocsmGetBnds — get the Bounds of a Parameter
- ocsmPrintPmtrs — print external Parameters to file

- ocsmGetValu — get the Value of a Parameter
- ocsmGetValuS — get the Value of a string Parameter
- ocsmSetValu — set a Value for a Parameter
- ocsmSetValuD — set a (double) Value for a Parameter

## Body management

- ocsmGetUV — get the parametric coordinates on an Edge or Face
- ocsmGetXYZ — get the coordinates on a Node, Edge, or Face
- ocsmGetNorm — get the unit normals for a Face
- ocsmGetVel — get the velocities of coordinates on a Node, Edge, or Face

- ocsmGetBody — get info about a Body
- ocsmPrintBodys — print all Bodys to file
- ocsmPrintBrep — print the BRep associated with a specific Body

- `ocsmSetVel` — set the velocity for a Parameter
- `ocsmSetVelD` — set the (double) velocity for a Parameter
- `ocsmGetTessVel` — get the tessellation velocities on a Node, Edge, or Face

- `ocsmPerturb` — create a perturbed MODL
- `ocsmSetDtime` — set sensitivity FD time step (or select analytic)

- ocsmVersion — return current version
- ocsmSetOutLevel — set output level

- ocsmSetEgg — set up alternative tessellation by an external grid generator
- ocsmEvalExpr — evaluate an expression
- ocsmPrintEgo — print the contents of an EGADS ego
- ocsmGetText — convert an OCSM code to text
- ocsmGetCode — convert text to an OCSM code

- `ocsmGetSketch` — get string data associated with a Sketch
- `ocsmSolveSketch` — solve for new Sketch variables
- `ocsmSaveSketch` — overwrite Branches associated with a Sketch

# Sample Application (1)

```
/*
 ************************************************************************
 *                                                                      *
 * simpleCSM.c -- simple OpenCSM application                            *
 *                                                                      *
 *               Written by John Dannenhoffer @ Syracuse University     *
 *                                                                      *
 ************************************************************************
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <math.h>

#include "egads.h"
#include "OpenCSM.h"
```

# Sample Application (2)

```
/*****************************************************************/
/*                                                               */
/*   main program                                                */
/*                                                               */
/*****************************************************************/

int
main(int       argc,                    /* (in)  number of arguments */
     char      *argv[])                 /* (in)  array  of arguments */
{
    int          status, imajor, iminor, nbrch, npmtr, nbody;
    int          ipmtr, builtTo, type, ichld, ileft, irite;
    int          nnode, nedge, nface, iface, npnt, ntri, ipnt;
    const int    *ptype, *pindx, *tris, *tric;
    double       vals[10], *vel=NULL, vbest, vtest;
    const double *xyz, *uv;
    void         *modl;
    modl_T       *MODL;
```

# Sample Application (2)

```
/* ------------------------------------------------------------ */

    /* report version */
    status = ocsmVersion(&imajor, &iminor);
    printf("ocsmVersion -> status=%d, imajor=%d, iminor=%d\n",
           status, imajor, iminor);

    /* load a .csm file */
    status = ocsmLoad("bolt.csm", &modl);
    printf("ocsmLoad -> status=%d\n",
           status);
    if (status != EGADS_SUCCESS) return(EXIT_FAILURE);

    /* get info about the current MODL */
    status = ocsmInfo(modl, &nbrch, &npmtr, &nbody);
    printf("ocsmInfo -> status=%d, nbrch=%d, npmtr=%d, nbody=%d\n",
           status, nbrch, npmtr, nbody);
```

# Sample Application (3)

```
/* change the shaft's diameter (Dhsaft) to 0.50 */
status = ocsmFindPmtr(modl, "Dshaft", OCSM_EXTERNAL, 1, 1, &ipmtr);
printf("ocsmFindPmtr -> status=%d, ipmtr=%d\n",
       status, ipmtr);

status = ocsmSetValuD(modl, ipmtr, 1, 1, 0.5);
printf("ocsmSetValuD -> status=%d\n", status);

/* build the MODL (and do not return Bodys directly) */
nbody = 0;
status = ocsmBuild(modl, 0, &builtTo, &nbody, NULL);
printf("ocsmBuild -> status=%d, builtTo=%d\n",
       status, builtTo);

/* find the sensitivity with respect to the shaft's length (Lshaft) */
status = ocsmFindPmtr(modl, "Lshaft", OCSM_EXTERNAL, 1, 1, &ipmtr);
printf("ocsmFindPmtr -> status=%d, ipmtr=%d\n",
       status, ipmtr);

status = ocsmSetVelD(modl, 0,     0, 0, 0.0);
printf("ocsmSetVelD(0) -> status=%d\n",
       status);
```

# Sample Application (4)

```
status = ocsmSetVelD(modl, ipmtr, 1, 1, 1.0);
printf("ocsmSetVelD(1) -> status=%d\n",
       status);

status = ocsmPrintPmtrs(modl, stdout);
printf("ocsmPrintPmtrs -> status=%d\n", status);

/* rebuild the MODL (needed to propagate velocities to the arguments) */
nbody = 0;
status = ocsmBuild(modl, 0, &builtTo, &nbody, NULL);
printf("ocsmBuild -> status=%d, builtTo=%d\n",
       status, builtTo);

/* get info about the current MODL (which should now contain Bodys) */
status = ocsmInfo(modl, &nbrch, &npmtr, &nbody);
printf("ocsmInfo -> status=%d, nbrch=%d, npmtr=%d, nbody=%d\n",
       status, nbrch, npmtr, nbody);

/* get info about last Body (which is on the stack) */
status = ocsmGetBody(modl, nbody, &type, &ichld, &ileft, &irite, vals, &nnode,
printf("ocsmGetBody -> status=%d, type=%s, ichld=%d, ileft=%d, irite=%d, nnode=
       status, ocsmGetText(type), ichld, ileft, irite, nnode, nedge, nface);
```

# Sample Application (5)

```
/* find the maximum velocity for each Face */
for (iface = 1; iface <= nface; iface++) {
    printf("iface=%d\n", iface);

    /* get tessellation info (poke into MODL structure to get etess) */
    MODL = (modl_T *)modl;
    status = EG_getTessFace(MODL->body[nbody].etess, iface,
                            &npnt,  &xyz, &uv, &ptype, &pindx,
                            &ntri, &tris, &tric);
    printf("    EG_getTessFace -> status=%d, npnt=%d, ntri=%d\n",
           status, npnt, ntri);

    /* storage for velocities */
    vel = (double*) EG_alloc(3*npnt*sizeof(double));
    if (vel == NULL) {
        printf("malloc failed\n");
        return(EXIT_FAILURE);
    }
```

# Sample Application (6)

```
    /* getting velocity automatically computes sensitivities */
    status = ocsmGetVel(modl, nbody, OCSM_FACE, iface, npnt, NULL, vel);
    printf("    ocsmGetVel -> status=%d\n",
           status);

    vbest = 0;
    for (ipnt = 0; ipnt < npnt; ipnt++) {
        vtest = sqrt(vel[3*ipnt]*vel[3*ipnt]+vel[3*ipnt+1]*vel[3*ipnt+1]+vel[3*
        if (vtest > vbest) vbest = vtest;
    }

    printf("        vbest=%12.5e\n", vbest);

    EG_free(vel);
}
```

# Sample Application (7)

```
    /* free up the MODL */
    status = ocsmFree(modl);
    printf("ocsmFree(modl) -> status=%d\n",
           status);

    /* remove tmp files (if they exist) and cleanup udp storage */
    status = ocsmFree(NULL);
    printf("ocsmFree(NULL) -> status=%d\n",
           status);

    return(EXIT_SUCCESS);
}
```

# Hands-on Exercises

- Add an Attribute (`myName=$theShaft`) to the CYLINDER Branch
- Find the length of each Edge