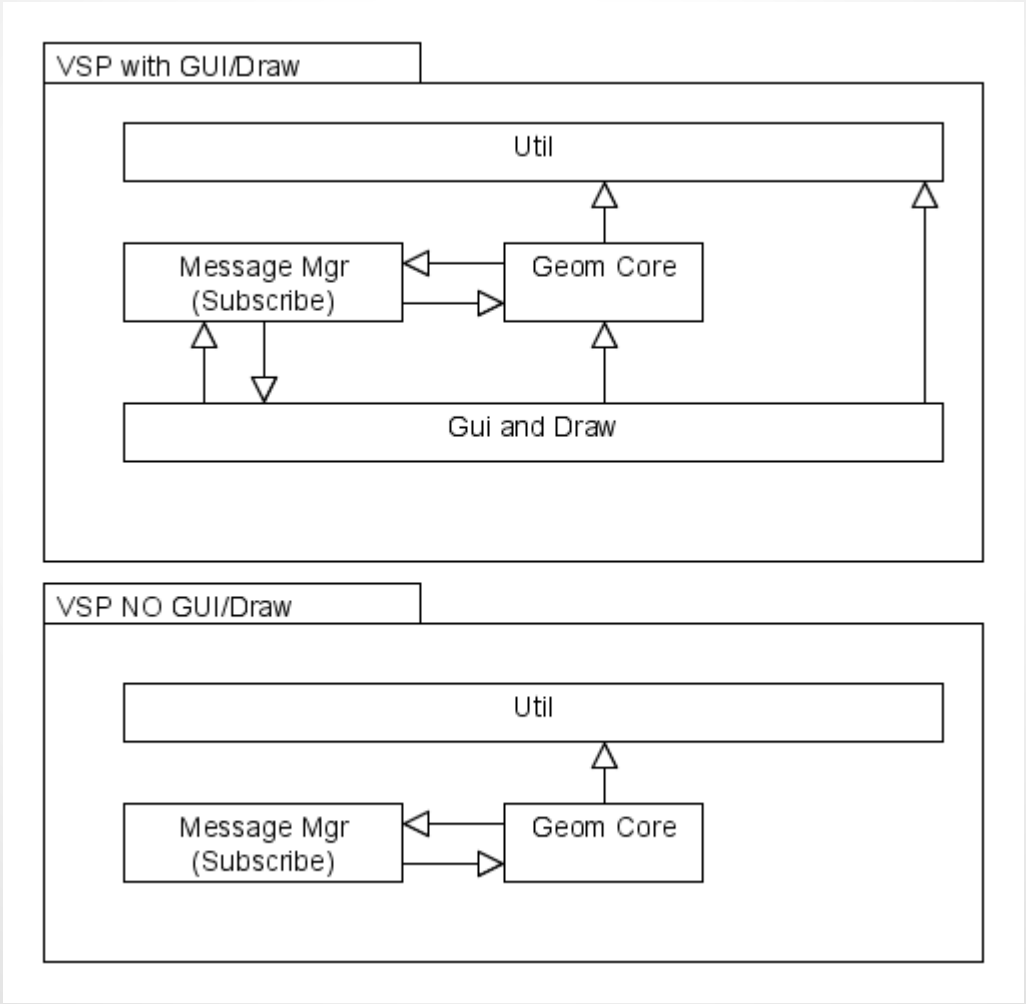
# OpenVSP 3.0 & API

# What and Why

V3.0 is a complete rewrite of VSP

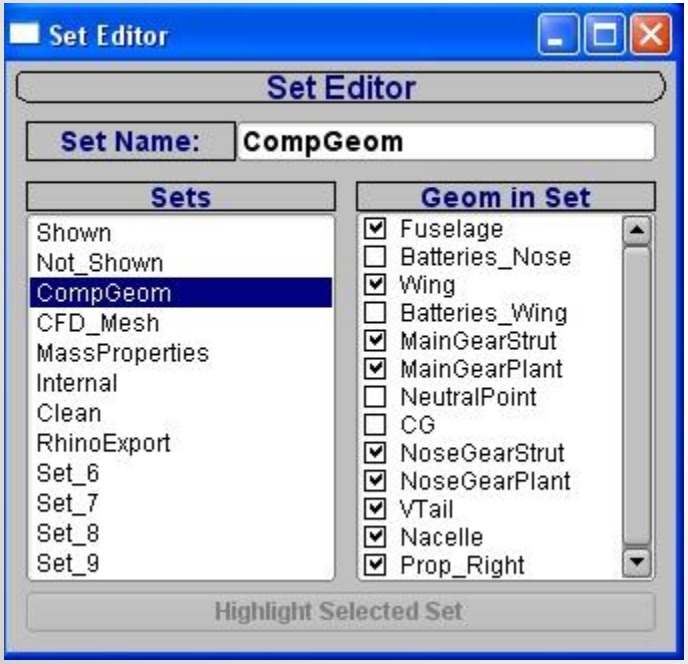
- Code bloat
- Replace old libraries
- Improve curve/surface libraries
- Separate geometry engine from GUI/graphics
- Consistent style
- Easier to extend
- Facilitate user defined geometry

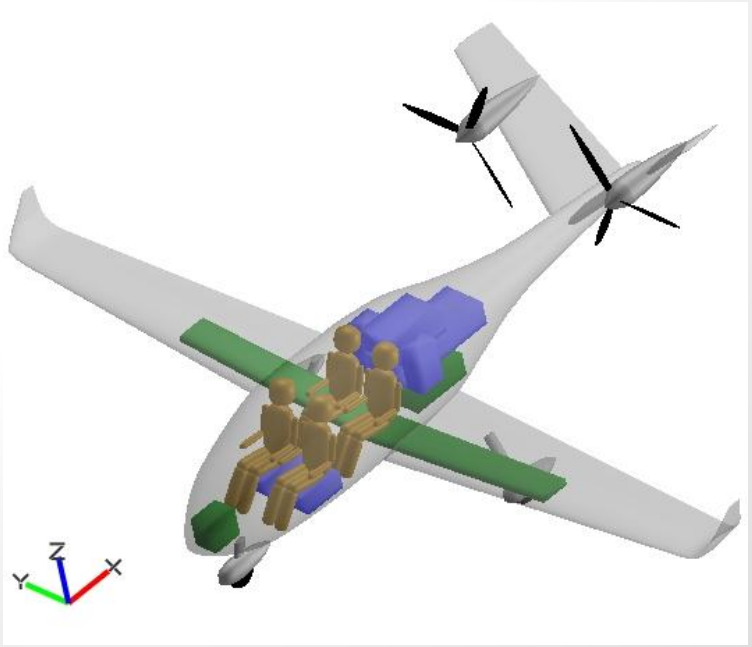
# Separate Core from GUI/Draw



# New Stuff: Sets

- Add geometry to named sets
- Use sets to analyze/export/visualize
- Use single model for multiple purposes



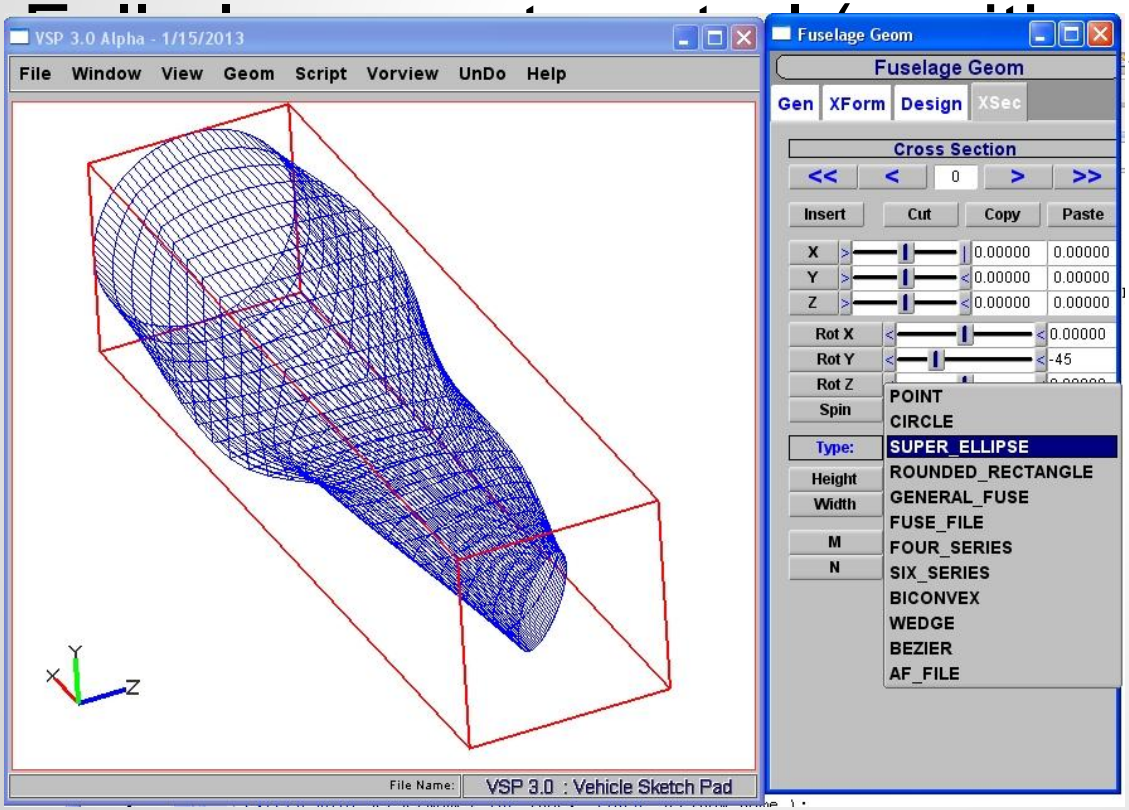
# New Stuff - UnDo

Full parameter undo stack.

# New Stuff - Fuselage XSecs

More cross-section types (airfoils, super ellipse..)

Full i[illegible]/orientation)



# API Design Goals

- Easy to use
- Replicate everything you can do in the GUI
- API calls should not change
- Simple error management
- No memory management
- Easy to make thread safe

# API Design

- Simple c/c++ function calls:

```
string AddGeom( const string & type, const string & parent = string() );
void CutGeomToClipboard(const string & geom_id);
```

- vsp Namespace

- Persistent unique ID strings:

```
string pod_id = vsp::AddGeom( “POD”);
string length_id = vsp::GetParm( pod_id, "Length", "Design" );
```

# API Error Manager

- Single persistent error stack.
- Query error stack at any time (pop, get).
- Error objects contain code and description.
- Optional error check after any line of code:

```
string pod_id = vsp::AddGeom( "POD", fuse_id );
vsp::ErrorMgr.PopErrorAndPrint( stdout );
```

# API Use Cases

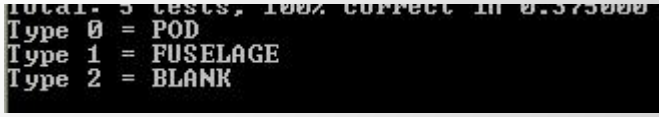
1. Create/Edit geometry
2. Detailed geometry edit
3. File I/O and Computations

# API Use Case - Create/Edit Geometry

```
//==== Init VSP ====//
vsp::VSPInit();
vsp::ErrorMgr.PopErrorAndPrint( stdout );

//==== Get All Available Geom Types =====//
vector<string> types = vsp::GetGeomTypes( );
vsp::ErrorMgr.PopErrorAndPrint( stdout );

//==== Print Out All Available Geom Types ====//
for ( int i = 0 ; i < (int)types.size() ; i++ )
{
	printf( "Type %d = %s \n", i, types[i].c_str() );
}
```

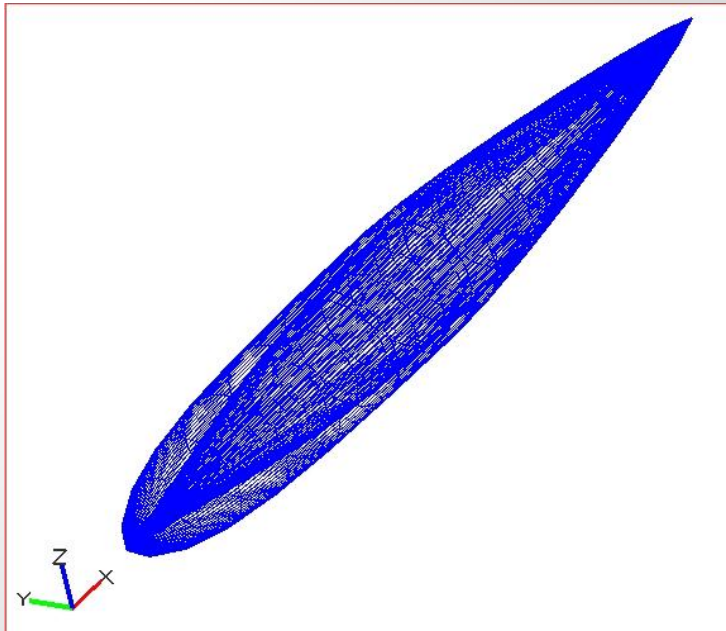


# API Use Case - Create/Edit Geometry

```
//==== Add Fuselage Geom =====//
string fuse_id = vsp::AddGeom( "FUSELAGE" );
vsp::ErrorMgr.PopErrorAndPrint( stdout );

//==== Add Pod Geom =====//
string pod_id = vsp::AddGeom( "POD", fuse_id );

//==== Set Name ====//
vsp::SetGeomName( pod_id, "Pod" );
```



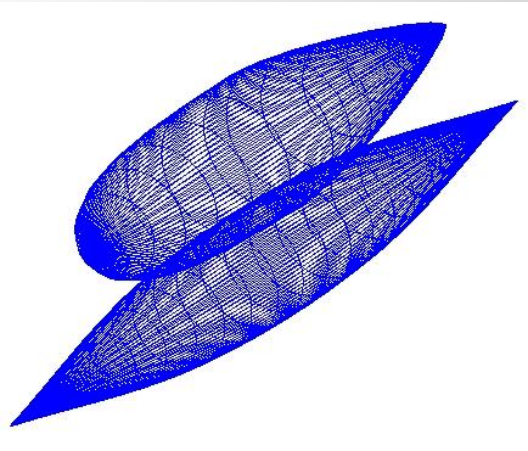
# API Use Case - Create/Edit Geometry

```
//==== Change Length ====//
string len_id = vsp::GetParm( pod_id, "Length", "Design" );
vsp::SetParmVal( len_id, 7.0 );

//==== Change Finess Ratio ====//
vsp::SetParmVal( pod_id, "FineRatio", "Design", 10.0 );

//==== Change Y Location ====//
string y_loc_id = vsp::GetParm( pod_id, "Y_Location", "XForm" );
vsp::SetParmVal( y_loc_id, 1.0 );

//==== Change X Location ====//
vsp::SetParmVal( pod_id, "X_Location", "XForm", 3.0 );
```



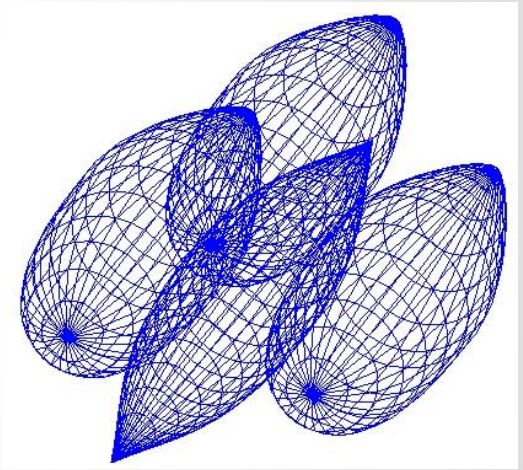
# API Use Case - Create/Edit Geometry

```
//==== Change Symmetry =====//
vsp::SetParmVal( pod_id, "Sym_Planar_Flag", "Sym", vsp::SYM_XZ  );

//==== Copy Pod Geom =====//
vsp::CopyGeomToClipboard( pod_id );
vsp::PasteGeomClipboard( fuse_id );

//==== Set Name ====//
vsp::SetGeomName( pod_id, "Original_Pod" );
string second_pod_id = vsp::FindGeom( "Pod", 0 );

vsp::SetParmVal( second_pod_id, "Sym_Planar_Flag", "Sym", 0 );
vsp::SetParmVal( second_pod_id, "Y_Location", "XForm", 0.0 );
vsp::SetParmVal( second_pod_id, "Z_Location", "XForm", 1.0 );
```



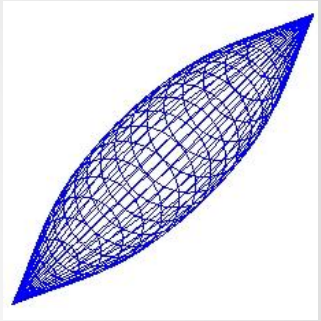
# API Use Case - Edit XSecs

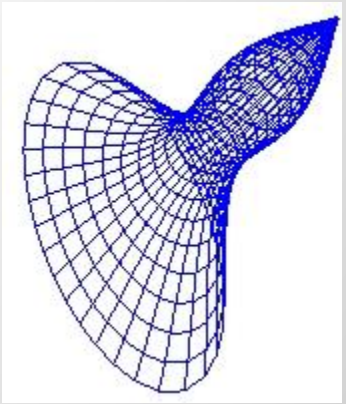
```
string fuse_id = vsp::AddGeom( "FUSELAGE" );

//==== Get XSec Surf ID ====//
string xsurf_id = vsp::GetXSecSurf( fuse_id, 0 );

//==== Change Type of First XSec ====//
vsp::ChangeXSecType( xsurf_id, 0, vsp::SUPER_ELLIPSE );

//==== Change Shape of First XSec ====//
string xsec_id = vsp::GetXSec( xsurf_id, 0 );
string w_id  = vsp::GetXSecParm( xsec_id, "Super_Width");
string h_id = vsp::GetXSecParm( xsec_id, "Super_Height");
vsp::SetParmVal( w_id , 4.0 );
vsp::SetParmVal( h_id, 2.0 );
```

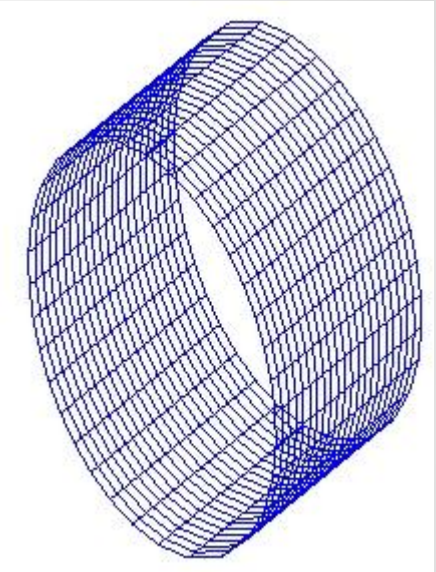




# API Use Case - Edit XSecs

```
//==== Copy Cross-Section to Clipboard ====//
vsp::CopyXSec( xsurf_id, 0 );

//==== Paste Cross-Section ====///
vsp::PasteXSec( xsurf_id, 1 );
vsp::PasteXSec( xsurf_id, 2 );
vsp::PasteXSec( xsurf_id, 3 );
```

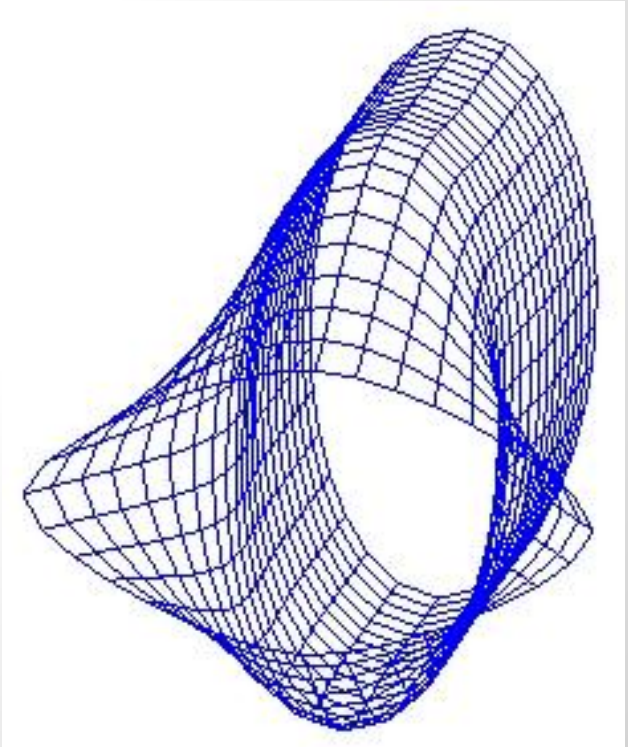


# API Use Case - Edit XSecs

```
//===== Change Type To File XSec ====//
vsp::ChangeXSecType( xsurf_id, 0, vsp::FILE_FUSE );
string file_xsec_id = vsp::GetXSec( xsurf_id, 0 );

//===== Build Point Vec ====//
vector< vec3d > pnt_vec;
pnt_vec.push_back( vec3d( 0.0, 0.0, 2.0 ) );
pnt_vec.push_back( vec3d( 0.0, 1.0, 0.0 ) );
pnt_vec.push_back( vec3d( 0.0, 0.0,-2.0 ) );
pnt_vec.push_back( vec3d( 0.0,-1.0, 0.0 ) );
pnt_vec.push_back( vec3d( 0.0, 0.0, 2.0 ) );

//===== Load Points Into XSec ====//
vsp::SetXSecPnts( file_xsec_id, pnt_vec );
```



# File I/O and Computations*

```
//==== Set Working Directory and Read File ====//
SetWorkingDir( “C:/vsp/api/” );
ReadVSPFile( “Model.vsp” );

//==== Edit Model and Save Model ====//
string fuse_id = FindGeom(“Fuselage”, 0 );
vsp::SetParmVal( fuse_id,  "Length", "Design", 20.0 );
WriteVSPFile(“Model.vsp” );

//==== Compute Mass Properties ====//
int num_slices = 50;
MassOutput results = vsp::MassProperties( num_slices , “MassSet” );
```

*work in progress