A FRAMEWORK FOR DEVELOPING GEOMETRY-GRID TEMPLATES FOR PROPULSION ELEMENTS

Douglas H. Ross, Mark Dillavou, Sankarappan Gopalsamy,
Phillip C. Shum and Alan M. Shih
Department of Mechanical Engineering
University of Alabama at Birmingham (UAB)
Birmingham, AL 35294, U.S.A.
E-mail: {dhross, dillavou, sgopals, shum, ashih}@uab.edu

ABSTRACT

A geometry-grid template consists of a geometric model, a grid over the model and a set of parameters that define the changes to the geometry or grid that the user would desire to modify. This paper presents a framework, called MiniCAD, for interactively developing geometry-grid templates for propulsion elements such as injectors and volutes. The interactive creation of a grid template includes the initial geometry creation or importation and cleanup. The next step is topology and grid creation. Then the interactive extraction of object attributes is accomplished using drag and drop and auto complete to create template parameters. Creation of template parameters includes the writing of equations to form the interrelationships between objects attributes and the parameters. After the template is completed, parameters are modified and the update operation is initiated to execute the equations and the objects parametric relationships to recreate the geometry and grid based on the new parameter values. Grid patterns, which are topological structures that include grid values and can be reused by connection to different geometries, are also introduced. MiniCAD is being developed at Enabling Technology Laboratory (ETLab) at the University of Alabama at Birmingham.

I. INTRODUCTION

A geometry-grid template consists of a geometric model and a grid over the model and a set of parameters that define the changes to the geometry and/or grid that the user would desire to modify. Ideally the parameters can be modified from a graphical user interface. Geometry–grid templates are desirable to create a set of grids for a parametric study of the model with either a varying geometry or grid or both. This paper presents a framework, MiniCAD[1-3], for interactively developing such geometry-grid templates for propulsion elements such as injectors and volutes. The MiniCAD framework is built on the Praxis Environment (Praxis) that provides the base graphical system and can provide a link to other software such as solvers or visualization, and the Geometry-Grid Toolkit[2-6] (GGTK) that provides underlying geometry, topology, and grid generation algorithms. The MiniCAD framework, Praxis and GGTK are being developed at UAB (the University of Alabama at Birmingham).

Geometry–grid template development at UAB has evolved through three methods. The first method was stand-alone programmatic development. The second method was programmatic but based on a library of geometry and grid generation tools with a general purpose GUI. The third and current method is interactive geometry–grid template creation. The evolution was aimed at

reducing the time to develop a template but has also had the affect of reducing the required knowledge of the developer.

There are many undesirable aspects to a stand-alone programmatic template. To develop a programmatic template with no reliance on a library of geometry and grid functions the developer must be knowledgeable enough and take the time to write the underlying routines to create the geometry and grid and create a topological structure to insure that a watertight grid is created. The time may be reduced if the developer already has his own "library" of functions that he has developed for geometry and grid creation, however, the development of the template is still focused on a particular geometry. The development of one template may have little or no benefit for the development of the next template and modifications during development require significant effort.

The programmatic development of a geometry–grid template based on a geometry/ grid library of functions, such as GGTK, addresses some of the problems associated with the stand alone programmatic approach. The developer no longer needs to be proficient in coding geometry and grid generation functions but rather must just be able to understand the libraries interface and application. The development of a library that is assured of seeing use in multiple projects can have time spent on more general modeling operations and full featured robust grid generation. Another possible addition, as seen in GGTK, is an explicit topological structure. The downside is that the template is still developed programmatically which means that it is specific to a particular geometry and the developer must be a programmer conversant with the library.

The MiniCAD interactive geometry – grid template development framework addresses all of the problems associated with the above methods. The template developer is no longer a programmer but instead is a user conversant in geometry and grid generation. The programmer's job shifts to developing tools rather than templates, which enables the development of more general purpose tools for modeling and grid generation. The time to build the template is also drastically reduced as the geometry is either built interactively like in a CAD system or imported from an external CAD file. Through the use of the new grid pattern entities the topology of the grid need not match the topology of the geometric model, which allows the user to create the geometry faster. The grid pattern is created as a separate object from the geometry, which allows it to be reused for different geometric models. This also saves time.

To create the MiniCAD interactive geometry–grid template framework several external technologies are used which are covered in Section II. The main internal technology used in creating the MiniCAD framework is parametric modeling. The implementation of parametric modeling for geometry created in the MiniCAD framework and the issues and solutions for imported geometry are covered in Section III along with the connection to the topology. The discussion of grid patterns, which is under development, is in Section IV. Template development using object attributes and the user interface is discussed in Section V.

II. EXTERNAL TECHNOLOGY

Following is a description of the technologies that provide some of the resources required to implement the MiniCAD interactive template framework.

PRAXIS ENVIRONMENT

The Praxis Environment provides a consistent set of tools for application development and inter-application connectivity. Based on the Python[7] programming language and providing access to the wxWidgets[8] GUI toolkit, Praxis provides cross-platform tools for graphical user interaction. Praxis also provides "docking station" functionality: MiniCAD's components dock with Praxis, which can then provide access to MiniCAD's functions to any other docked component, and can also provide access to the functions of any other docked component to MiniCAD, making plug-in development very simple. MiniCAD is written in C^{++}, but connects to Praxis through a Python interface, automatically generated using SWIG. Other code and executables can be wrapped similarly, allowing MiniCAD to directly utilize solvers and visualization applications.

The MiniCAD/Praxis system can also act as a straightforward framework for custom applications, using only the desired MiniCAD components and connecting them with new custom code or GUIs through the Praxis Dock.

SUBVERSION VERSION CONTROL SYSTEM

The MiniCAD, GGTK, and Praxis collaborative development is handled by Subversion[9], a successor to the CVS version control system. MiniCAD and GGTK are automatically built for various platforms for each commit, using BuildBot[10]. This helps to ensure that development proceeds in a way that maintains cross-platform compatibility.

GGTK

MiniCAD is supported for geometry creation, model topology creation and grid generation by the Geometry Grid Tool Kit (GGTK). GGTK provides geometric modeling functions using NURBS curves and surfaces and many grid generation methods for both structured and unstructured grids. GGTK also provides a topological model that assures that the grid will be watertight by providing a single geometric value for any point on each topological element. An uncommon structured face grid method that GGTK provides is by re-parameterization[11]. This method is robust for badly formed surfaces with singularities as an alternative to carpeting. This method is also well suited for creating grid over multiple trimmed surfaces, which can result from an application of a grid pattern.

MiniCAD has wrapper classes for the GGTK creation methods. The MiniCAD classes include the added features of displaying the objects and attributes, maintaining a history of creation attributes, transformations and dependency information. When geometry, topology or grid objects are created they are added to a map that connects the objects pointer to a distinct integer value. This map is used to retrieve a reference to objects for display, selection and modification of attributes. MiniCAD's modeling capability includes support for trimmed geometry both creation and editing and reading from IGES files. Both surface modeling and some basic solid modeling are supported.

III. PARAMETRIC MODELING

A key technology required for template development is parametric modeling. Parametric modeling for CAD packages takes two forms, *history based* and the ability to have *inter-relations between parameters*, which can be interpreted as constraints[12]. MiniCAD supports both of these forms. The history based form is divided into two methods. The first method is to remember the creation values for an object. The second method is to remember the creation relationships between objects, which we define as upstream-downstream relationships. Both of these are applicable to objects created inside the MiniCAD environment. The other form is through relationships between object parameters specified by equations associated to object parameters. For the rest of the paper, MiniCAD objects parameters are termed "attributes" and the values that are set in the template are termed "parameters". This method is directly applicable to internally created geometry where values that can be modified are stored in attributes. Investigation is underway for parameter based modification for imported geometry. More details of the various concepts and issues in the two forms of parametric modeling in MiniCAD are given below. Another topic covered is that of imported geometry cleanup since it relates to the creation of a geometry grid template based on imported geometry.

UPSTREAM-DOWNSTREAM CONNECTIONS

Upstream-downstream connections are best described by some examples. A line is generated between two points. The points are upstream of the line and the line is downstream of the points. Whenever the position of the points changes the line is recreated to use the new position of the points. Another simple example would be a cylinder created by extruding a circle. When the radius of the circle is modified the radius of the cylinder is modified. A slightly more complicated example is a trimmed surface created from a set of planar curves. The curves are upstream of the surface that is created to bound the curves and parametric curves are downstream of the surface and the curves. The trimmed surface is then downstream of the surface and parametric curves. A modification of one of the curves rebuilds the surface, parametric curves and trimmed surface although the original information passed to create the trimmed surface was just the 3d planar curves.

MAINTAINING THE TOPOLOGICAL RELATIONSHIP

If the system stopped at the geometry level then the constructor creation information would be sufficient for updates. Because links through the topology to the grid are required it is necessary to store the completed geometry with a more complicated relationship than for a purely geometric modeling system. For example, if only the curves and the extrude operation to create a volume are saved most of the geometry that should maintain a link to the downstream topology and grid would be rebuilt and would loose its connectivity. This does currently limit the framework to only supporting geometric changes that do not alter the topology.
Topology has an upstream-downstream relationship with the geometry. A topological object may have multiple geometric instances. For example, if an edge is the common boundary of two faces, then the edge will have two instances of geometry curves – one with respect to each face. In such cases, a change to either geometric instance will require rechecking of the topology. Boolean operations are not yet supported in the upstream-downstream connections.

The upstream-downstream relationship between topology and grid objects is simple. Updating the topology due to an underlying geometry change (assuming no change in topological structure) will pass the same geometric information to the grid.

ATTRIBUTES

The other feature critical to the development of templates is having attributes that can be interactively selected for use in the template and the assignment and evaluation of equations that define relationships between attributes. The equation that defines a parameters relationship to other parameters is stored as a text string with the attribute. These equations are executed in the python interpreter. The equations currently have no branching or conditional capability. Examples of attributes are a points x, y or z coordinates, a circle radius, the number of grid points or grid packing values.

IMPORTED GEOMETRY PARAMETERS

For external geometry imported through a file format such as IGES there is little or possibly no parameter information such as the case of the MSFC volute file where all of the curves and surfaces are of type trimmed B-Spline. This makes the possibility of parametric modeling of the geometry difficult. We are investigating the use of influence objects to transform the geometry. The difficulties are to create parameters that make sense from an engineering perspective and to properly transform the trimmed geometry such that the topology is maintained.

IMPORTED GEOMETRY CLEANUP

The other aspect of imported geometry that must be addressed when creating a grid is cleanup. The current method of transference of external geometry into MiniCAD is through an IGES file. It is assumed that the imported geometry will be trimmed surfaces. MiniCAD provides visualization for the trimmed surfaces as well as the trim curves both in 3d and 2d parametric space. The reasons for displaying the 2d representation are for a teaching aid, for trouble shooting and for an additional modeling environment.

External geometry that is imported for grid creation is often created with a different purpose in mind; therefore, cleanup may have to be performed on the geometry to make it suitable for grid creation. One of the steps is de-featuring. De-featuring in MiniCAD supports both the deletion of geometry that is not required and the simplification of geometry that remains. Deletion is simply a matter of selecting the trimmed surface to be deleted and pressing the delete key. A somewhat unique feature of MiniCAD is the removal of trimmed surface features by either removing inner trim loops or by modifying the outer trim loop of a trimmed surface. Inner trim loops are deleted through selection and the delete key. Modification of an outer loop currently has two possibilities either removal of a curve or set of curves that forms a 2d concavity in the outer loop, which is replaced by a line segment. The other cleanup is the removal of a curve or set of curves that form a 2d concavity at a corner of the outer loop which is replaced by a corner that uses the u and v parameters of the points where the curve connects to the outer loop.

Another cleanup functionality is the requirement to build additional geometry to close open faces. MiniCAD's geometry construction methods are suitable to the task using complete and trimmed surface generation.

The other method of cleanup is when the topology is formed. By judicious setting of the 2d and 3d tolerance values geometry trim curves and their end points can be merged to form a watertight topological structure. When the vertices are merged it is possible to form edges that only have one vertex. These must be removed but tested that they are not loops, which also only have one vertex. Another supported topology modification is an edge split.

Figure 1 shows the initial imported volute geometry as trimmed surfaces from an IGES file. The geometry is displayed in wireframe to provide a better view of its complexity. Cleanup of the geometry to make a watertight topology involved deletion of trimmed surfaces, deletion of inner loops, deletion of curves from outer loops both along edges and at corners, the use of the topology split edge function and the construction of some additional geometry using extrusion and trimmed surface from planar curves.

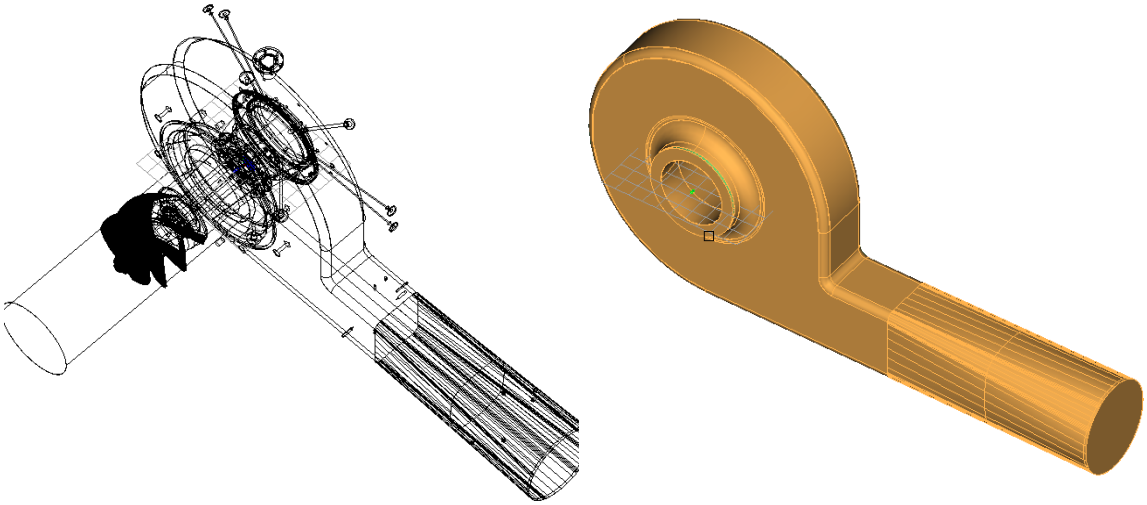

Figure 1: Cleanup of the volute geometry imported from an IGES file

IV. GRID PATTERNS

A grid Pattern is a reusable topological structure of volume, face, edge and vertex entities with grid specification attributes as appropriate for each type. A grid pattern is connected to a geometry model to create a grid. Grid patterns are being developed in the belief that the same topological grid pattern is reusable for many similar geometries. The grid pattern is created separately from the geometry that it will be connected to and has its own geometry only for visualization. To use the grid pattern on a geometry, the grid entities are connected to the geometry through the topological structure using geometry parametric values. Using parametric values allows the geometry to change in 3d yet have the grid still positioned proportionally on the geometry. Connecting to the geometry through the model topological structure guarantees a watertight grid without the grid topology having to be concerned with multiple geometric representations. Grid patterns are developed through composition and decomposition of basic building blocks. In the case of a 2d grid the basic building blocks are faces which can be

composed (add additional face blocks) or decomposed (split edges). For a 3d grid the basic building block is a volume block. Composition consists of adding additional volume blocks and decomposition consists of splitting faces. Figure 2 shows an H-type volume grid pattern block topology displayed as a cube above the volute. The grid pattern block has had the vertices and edges connected to the volute model topology. The four vertices to the left have been connected to model topology edges and the four vertices to the right have been connected to model topology faces. The grid edges on the right face of the cube are attached to portions of a model edge or span two model edges. The rest of the grid edges span multiple model topology faces. The model faces are outlined with solid lines and the grid edge points are shown as dots. A uniform distribution for the grid points was applied.

To support the grid patterns new grid entities are also being developed that can span portions of or multiple topological entities. The new grid entities address a limitation noted in a review of GGTK by the Geometry and Grid Modeling for Numerical Simulation (GGMNS) project that the geometry and grid topology were one to one[13]. The new grid entities at the MiniCAD level have the topological information, the grid attributes and the connection information. The grid edges use a compound curve that can span multiple edges or faces. The grid face can span multiple topological faces. The grid face can be formed either by a process of creating a TFI grid from the grid edges and then projecting onto the surfaces or use the grid edge information to create trimmed surfaces that can be triangulated and combined to use the re-parameterization method.

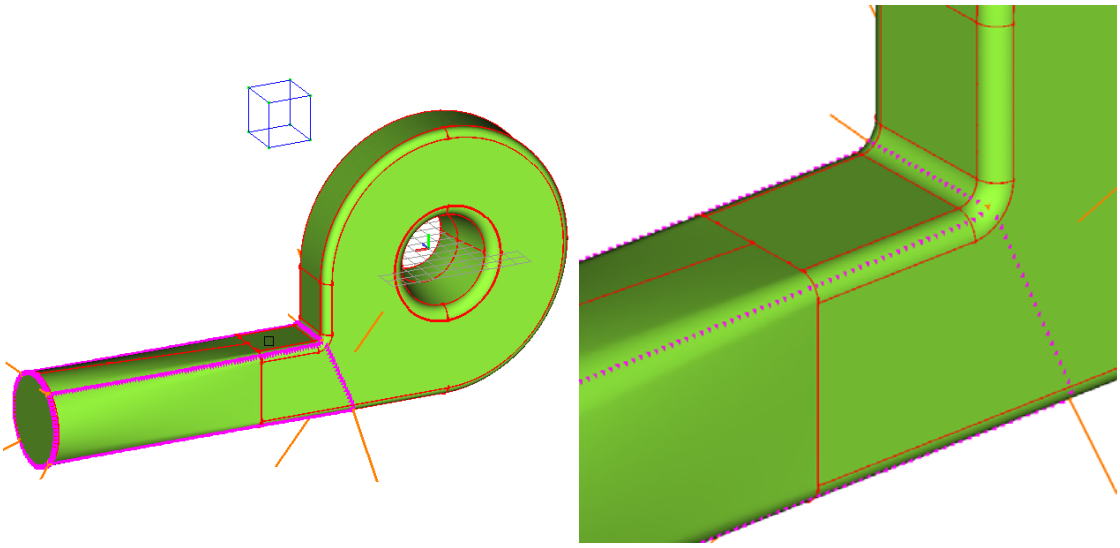

Figure 2: H-type volume grid pattern block on the volute model

V. TEMPLATE DEVELOPMENT

The template framework leverages the power of the Python scripting language which gives the user a lot of flexibility when creating templates. Any user can create templates without any programming knowledge, but to harness the full power of templates, a basic understanding of the Python scripting language is required. Templates are implemented into the graphical interface of MiniCAD which makes generating templates easy. This allows the user to drag objects from the graphics window or tree browser and quickly set objects equations. The input also has text auto-

completion, so after the user drags or types an object's name, a list of all that object's attributes that can be used as template values will be displayed.

Because template creation is interactive, many templates can be created in hours instead of weeks like a programmatic template would take. This also allows users who don't know how to program to develop templates. While the goal of templates is to allow the user to decide which attributes will be changed after the geometry is created, the way the geometry is built will affect the ease with which the template can be built. The user can save the template in a file, and give the file to another user, who can then easily load the file to create a grid with different values.

A typical template is created after the user has generated geometry, topology and grid inside of MiniCAD or imported geometry from another CAD system, cleaned it up and then created topology and grid. Once the user has the desired grid, the user can begin to create a template. A typical template is made up of multiple parameters, and each parameter may have multiple equations. Each parameter represents a different value the user can change, and the equations define how the attribute values of the geometry or grid will change when the parameter's value changes. To create a new parameter, the user will select the "New Parameter" button under the Template interface. The next step is to give the parameter a name, such as "Pipe Radius" and an initial value. The parameter needs to be hooked up to the geometry by specifying equations. To create an equation, the user can select the geometry in the graphics window and drag it to the equation input. A user can model almost any mathematical equation, and sometimes it maybe necessary to have multiple equations for a single parameter to fully define how the geometry and/or grid should change. Currently, each equation is a single line with no branching (if-statements, loops) are allowed in the templates, but may possibly be implemented in the future.

Because MiniCAD uses Python as a scripting language, users can extend MiniCAD templates by writing python scripts. A common example of this might be that a user wants to change a value over a range, and write out a grid in plot3d format at each step. The following simple four line python script will accomplish this by setting the parameter, 14NozzleExitDia, to the values of 2 through 3.8 by a step of 0.2:

```
for i in range(20, 40, 2):
    setGlobalAttributeAndApply("14NozzleExitDia", i/10.0))
    name = "SEIGrid%s.p3d" % i
    a = Plot3dWriter(name, 1)
```

This script can be executed from the MiniCAD command line.

Figure 3 shows the MiniCAD framework with a geometry- 2d grid template developed for a single element injector (SEI). The single element injector in the framework view is displayed with some initial parameter values. The grid for this template did not use grid patterns. Therefore additional geometry had to be created to create the blocking and equations were written to correctly set the number of grid points to maintain a structured grid. Figure 4 shows the single element injector with the outlet diameter increased. The geometry is recreated, the topology updated and the grid updated. In Figure 5 a geometry change is made to increase the throat diameter and again, the geometry is recreated, the topology is updated and the grid is updated.

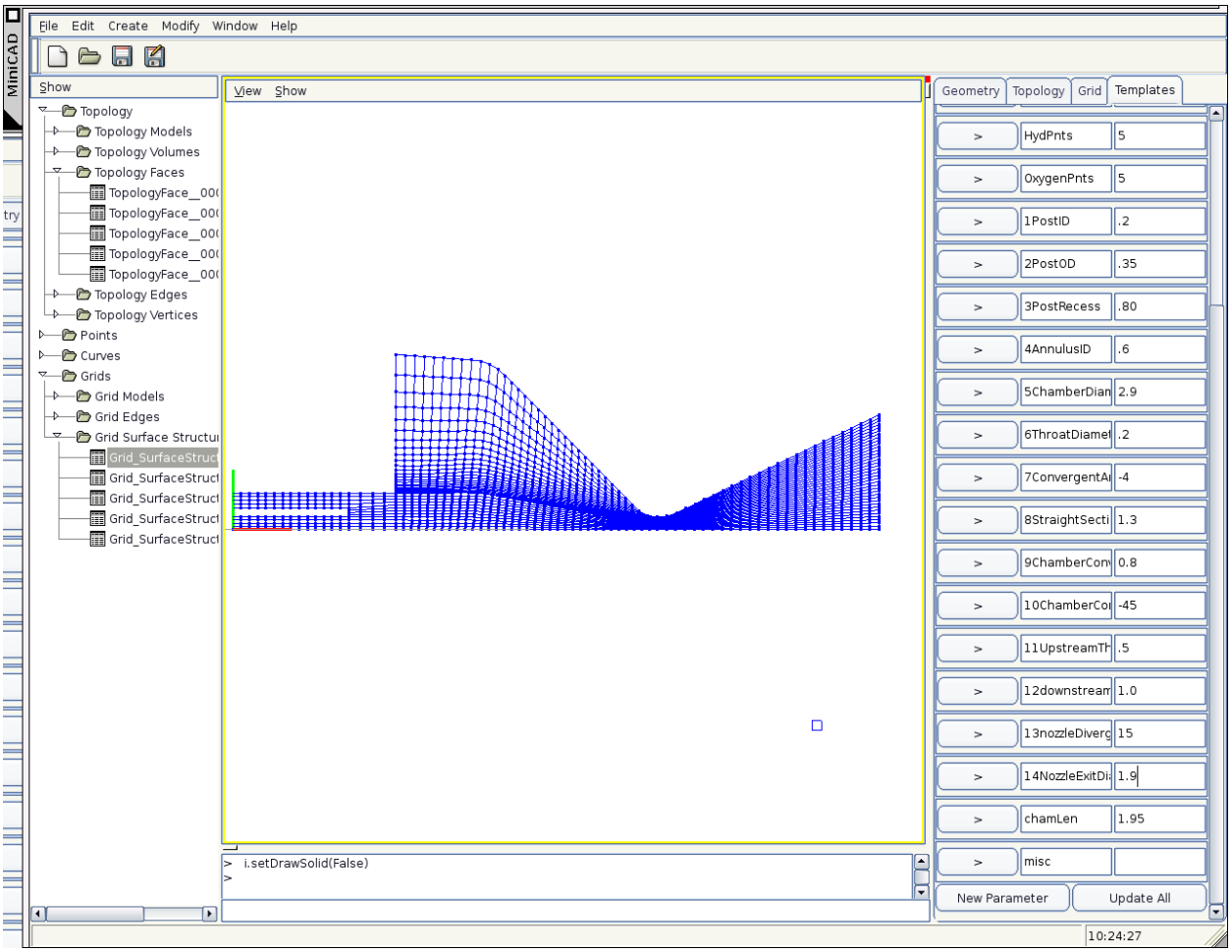

Figure 3: MiniCAD framework with SEI template

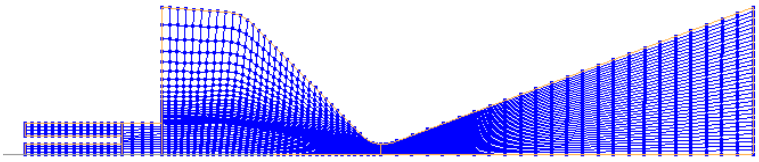

Figure 4: SEI outlet diameter increased

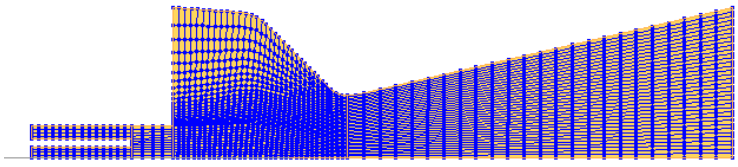

Figure 5: SEI throat diameter increased

VI. CONCLUSIONS

The MiniCAD template framework provides a basis for interactively creating grid templates from interactively created geometry or imported geometry. The time to generate a geometry-grid template is improved over previous methods. By reducing the development time for a geometry-grid template, it will enable greater ability to perform design parametric studies critically needed for improving designs. Its geometry, topology and grid generation engine, GGTK, provides MiniCAD users advanced methods to create sophisticated models and the compatibility to import external geometry models. Currently MiniCAD provides many CAD-like features and basic solid modeling capabilities, with near-term plan to further enhance its modeling capabilities. Grid Pattern Library in MiniCAD will greatly facilitate the mesh generation process, as it will allow the user to map a geometry with a pre-defined grid pattern and provides a shortcut in terms of generating the grid from a geometric model.

Praxis Environment provides MiniCAD a technological foundation for collaboration with other software, as it serves as a "docking station" to allow functionalities from other modules commutable with MiniCAD. With its provided visualization and user interface functionalities, it also makes possible the development of an integrated simulation system that involves all the components, including solvers and post-processing tools.

ACKNOWLEDGEMENTS

This research was supported in part by the NASA CUIP Program, the DOD MSRC PET Program, and the NSF ITR Program. The authors would like to express their gratitude to Mr. Robert Garcia, Mrs. Lisa Griffin and Mr. John Peugeot at NASA MSFC for their valuable suggestions and input to the template framework development.

REFERENCES

1. MiniCAD Documentation http://www.eng.uab.edu/me/ETLab/Software/minicad/

2. Gopalsamy, S., Ross, D.H., Shih, A.M., "API for Grid Generation Over Topological Models," The 13th International Meshing Roundtable, Williamsburg, Virginia, USA, September 19-22, 2004, pp. 221-230.

3. Soni, B., Cheng, G.C., Koomullil, R., Shih, A., Luke, E., and Thompson, D., "Enabling Computational Technologies for Aerothermodynamics Applicable to Next Generation Launch Vehicles," AIAA Paper 2004-3987, 40th AIAA/ASME/SAE/ASEE Joint Propulsion Conference and Exhibit, July 11-14, 2004.

4. GGTK Documentation http://www.eng.uab.edu/me/ETLab/Software/ggtk/

5. Shih, A.M., Yu, T-Y, Gopalsamy, S., Ito, Y., and Soni, B.K., "Geometry and Mesh Generation for High Fidelity Computational Simulations Using Non-Uniform Rational B-Splines," Applied Numerical Mathematics (In Print)

6. Ito, Y., Shih, A. M. and Soni, B. K., "Reliable Isotropic Tetrahedral Mesh Generation Based on an Advancing Front Method," Proceedings of the 13th International Meshing Roundtable, Williamsburg, VA, September 2004, pp. 95-105.

7. Python Programming Language. http://www.python.org/

8. wxWidgets Home. http://www.wxwidgets.org/

9. subversion.tigris.org http://subversion.tigris.org/

10. The BuildBot http://buildbot.sourceforge.net/

11. S. Gopalsamy, Douglas H. Ross, Yasushi Ito and Alan M. Shih, “Structured Grid Generation over NURBS and Facetted Surface Patches by Reparameterization”, Proceedings of the 14th International Meshing Roundtable, San Diego, CA, September 2005.

12. Marr, G., “Parametric and Feature-Based Modeling”, in thesis – Development of Methodology for Creating Families of Parts, Worcester Polytechnic Institute, http://alum.wpi.edu/~gregm/thesis/node11.html, 1996.

13. John Steinbrenner, “Review of Geometry and Grid Tool Kit (GGTK)”, http://www.pointwise.com/ggmns/review/Mesh%20and%20Geometry%20APIs/ggtk_review.html, 4 June 2004.