Proceedings of the 2014 Winter Simulation Conference
A. Tolk, S. Y. Diallo, I. O. Ryzhov, L. Yilmaz, S. Buckley, and J. A. Miller, eds.

TOWARDS THE IMPLEMENTATION OF A 3D HEAT TRANSFER ANALYSIS IN DYNAMIC-BIM (DYNAMIC BUILDING INFORMATION MODELING) WORKBENCH

Ravi S. Srinivasan

M.E. Rinker,
Sr. School of Construction Management
University of Florida
Gainesville, Florida 32611, USA

Siddharth Thakur
Manoj Parmar
Ishfak Akhmed

Department of Mechanical
And Aerospace Engineering
University of Florida
Gainesville, Florida 32611, USA

ABSTRACT

Energy efficient building design demands a complete understanding of building *envelope* heat transfer along with airflow behavior. Although existing building energy modeling tools provide 2D heat transfer analysis, they fail to execute full-scale 3D heat transfer analysis and lack proper integration with Building Information Modeling BIM tools. This paper addresses these issues first by developing a BIM-integrated plugin tool to extract building geometry and material information from a 3D building model and then demonstrating a complete 3D heat transfer analysis along with grid generation. This paper discusses the preliminary research work in data extraction from Building Information Modeling (BIM) for performing 3D heat transfer in a seamless manner. This approach will help towards the implementation of a 3D heat transfer in Dynamic-BIM Workbench, an integrative, collaborative, and extensible environment. This Workbench enables integration of domain modeling, simulation, and visualization.

1 INTRODUCTION

Integration of Building Information Modeling and building simulation tools is a new field of exploration in building science research domain. Although existing BIM tools allow simple energy, daylighting, and airflow analysis, they are not comparable to the typical, full-scale simulations conducted using standalone analysis tools. Currently, auxiliary programs are used to perform such micro-analysis. Moreover, for complete understanding of indoor airflow behavior in conjunction with envelope heat transfer, a robust tool that is seamlessly integrated with BIM is required. This is particularly important for window-wall interfaces and other intersections of envelope components that develop thermal bridges and double-skin façade assessment. While modeling the real world effects of envelope heat transfer and their effect on indoor airflow behavior is one aspect, co-simulation within BIM is another crucial step as several organizations has directed the use of BIM for new and existing buildings.

Despite several research efforts, current BIM and building performance tools' integration are at a level that are particularly nascent and do not contribute to the larger goal of designing low / net zero energy buildings. Two dimensional heat transfer modeling is possible using software such as THERM (2012), but they lack seamless integration with BIM tools. Although research work is in progress to improve the integration of the simulation software with BIM, existing simulation is limited to two-dimensional models. The recently developed Dynamic-BIM Workbench is a unified, interdependent, and interoperable domain that uses Open Graphics Rendering Engine (OGRE) for domain modeling, simulation, and visualization for energy and environmental assessments of buildings (Srinivasan et al., 2012, 2013), figures 1 and 2.

Using the dynamic linkages available in this Workbench, simulation tools can be linked such that users can perform analyses from within this Workbench. Among others, one of the goals is to develop an integrated tool that can extract all the necessary building information, execute heat transfer analysis, and visualize the results with the provision of flexibility and minimal user effort.

This paper discusses the preliminary research in the implementation of a 3D heat transfer analysis from BIM, which will, then, be extended in the Dynamic-BIM Workbench.

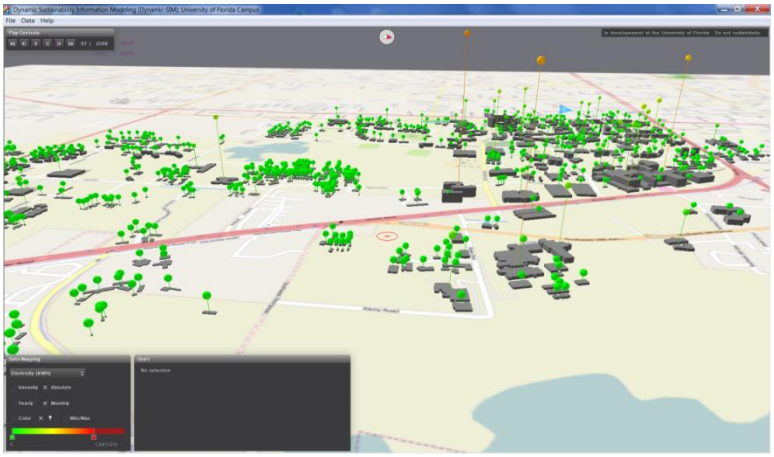

Figure 1: Dynamic-BIM Workbench showing UF Campus.

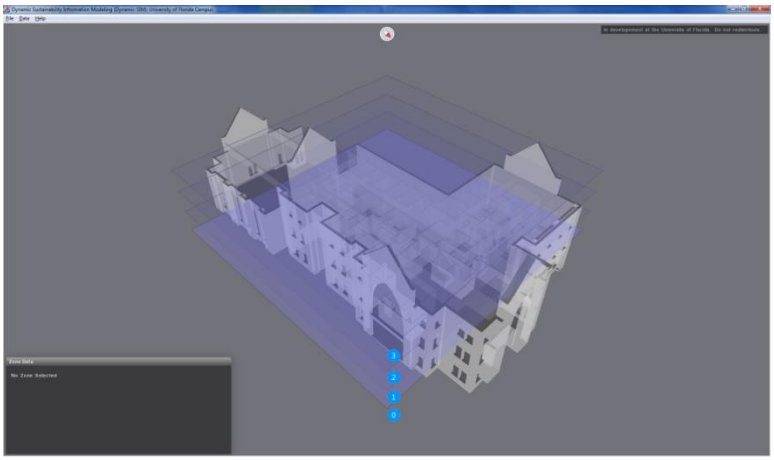

Figure 2: Dynamic-BIM Workbench showing Pugh Hall geometry extracted from Revit.

2 BIM- INTEGRATED 3D HEAT TRANSFER ANALYSIS IMPLEMENTATION

This implementation comprises of three modules namely, (i) data extraction, (ii) automatic 3D mesh generation, and (iii) 3D heat transfer. In order to model complex envelope configuration (i.e., intersection of envelope components that develop thermal bridges at the interfaces of wall with floors, roofs, exterior projections, windows, and other anomalies) and interior spaces, geometry data are captured using a plugin tool. As a preliminary research, this project focuses on extracting geometry information from Autodesk Revit® 2013 software (Revit 2013) because of its widespread use in architectural and engineering applications. The building data of the 3D model is stored in a .rvt file readable by Revit. Geometry data extracted from .rvt file includes all tessellated surface meshes for solid elements which is followed by automatic 3D meshing using an unstructured 3D mesh algorithm. Finite Volume Method (FVM) is used

to solve the heat transfer equations modeling the envelope heat transfer. Simulation is carried out using Loci framework (Luke and George, 2005) that enables efficient parallel processing, thereby increasing simulation speed manifold as compared to conventional algorithms used for building performance applications. Currently, the resultant data is visualized using Tecplot (Tecplot 2013). This will be extended in the Dynamic-BIM Workbench for integrated assessments. Long term expected outcome of the research would have a global outreach for project design teams to conduct early design decision-making, and to operate and maintain low/NZE buildings and their environment. This paper is organized in a manner reflecting the flow of data from the BIM environment to the simulation result visualization stage. It starts with an elaborate sketch of the geometry and material information extraction procedure from Revit, proceeds to mesh generation, explains heat transfer simulation using Loci-STREAM (Thakur and Wright, 2012) and finally concludes by presenting the visualization of the simulation results, figure 3. As a preliminary work, a sample building model with single layered components was used.

Figure 3: Data Flow Overview.

2.1 Extraction of Building Data

The objective of this step is to extract the required data needed for construction of 3D volume mesh and to generate inputs used by heat transfer analysis software. Revit provides a powerful Application Programing Interface (API) for integrating applications. This API is accessible by any .NET compliant programming language. For this project, a plugin was developed using C# programming language for extracting geometry information from Revit.

2.1.1 Revit Building Information Layout

According to Revit API Guide (2012), a single building or drawing component is considered as an Element; which might be a door, window or a wall, etc. In addition, an Element might also be a type of these components. In Revit, Elements are classified in six groups: Model, Sketch, View, Group, Annotation, and Information. Model Elements represent physical items that exist in building project. The geometry information is obtained from these elements. Elements are also classified as Category, Family, Symbol, and Instance. Figure 4 shows further classification of Family Elements.

Geometry information is extracted from these two family elements. As a first step, the elements are retrieved from Revit. This can be done in a number of ways: by Element ID, Element Filtering and Iterating, Element Selection or by accessing a specific element. Among these options, Element Filtering and Element Selection methods are suitable for geometry information extraction because of versatility and flexibility in operation. This project followed Element Selection method by using the UIDocument.Selection.Elements property to select elements from the current active document. The selected objects are in an ElementSet in Revit. From ElementSet, all selected elements were retrieved. The selection object can also be used to change the current selection programmatically. Figure 5 shows element selection in Revit user interface.

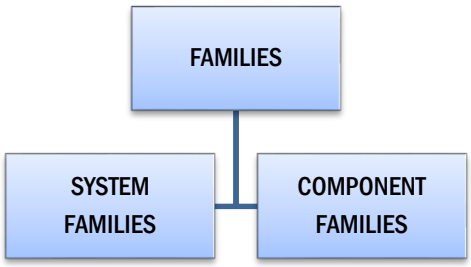

Figure 4: Family Element Classification.

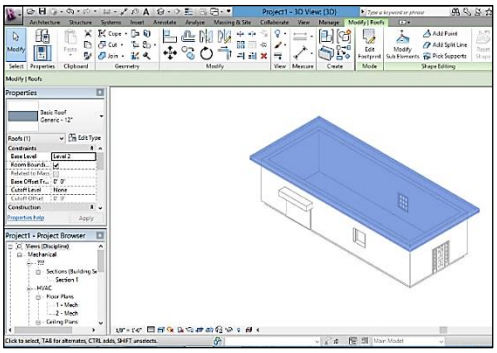

Figure 5: A Selected Roof.

Selected elements were placed in an ElementSet Array Class. Operation on each element was done by iterating ElementSet class. After elements are selected and each element is accessed for manipulation; next step is to convert the element to GeometryElement. Autodesk.Revit.DB namespace contains many classes related to geometry and graphic-related types used to describe the graphical representation in the API. The geometry-related classes include GeometryObject Classes, Geometry Helper Classes, Geometry Utility Classes and Collection Classes. GeometryElement Class is an API Geometry Class, which is a geometric representation of the element. Geometric conversion was done by using the Element.Get_Geometry() method; which essentially pulls any geometry information from any 3D model element. This applies both to system family instances such as walls, floors and roofs, and also to family instances of many categories, e.g. doors, windows, furniture, or masses. The extracted geometry is returned as GeometryElement; which can also be considered as a collection of geometry objects which describes the geometry of the element. Typically, the objects returned at the top level of the extracted geometry will be one of solids, curves, meshes, points, polylines or geometry instances. Figure 6 illustrates the hierarchy of objects found by geometry extraction.

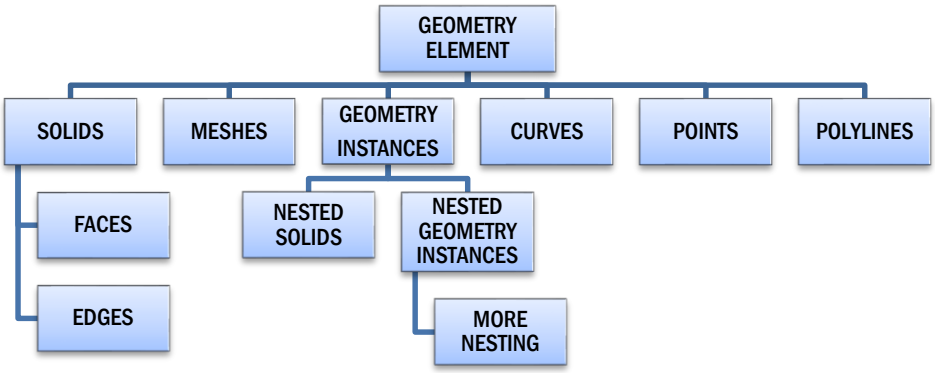

Figure 6: Hierarchy of Objects in GeometryElement.

2.1.2 System Family Objects

System Family Elements are those whose properties and geometric representations are predefined in Revit, e.g. walls, roofs, floors, etc. Their properties and related data are not available for loading and all the information are stored in Revit project file. The geometry element corresponding to a system family consists of one or more items at the top level depicted in figure 6, except geometry instance. In most common cases, this geometry element is a collection of solids only. A solid is a 3D geometry object bounded by a number of surfaces, i.e., Faces in Revit API. A Face in Revit is defined by a number of edge loops. Revit uses a variety of curve types to represent face geometry in a document, table 1.

Table 1. Face Representation in Revit (API Guide, 2012).

Face Type	Revit API Class
Plane	PlanarFace
Cone	ConicalFace
Cylinder	CylindricalFace
Revolved Surface	RevolvedFace
Ruled Surface	RuledFace

To get access to the Face information of a solid; first the geometry element is iterated to get all the geometry objects in it. Each of the geometry objects are analyzed and if it is a solid, then appended to a solid array. The solid array is then iterated to get each solid. Once a solid has been retrieved, it is then iterated to get all the faces bounding this solid and inserted into a face array. Finally, this face array is iterated to get access to each face object in it. Now, the bottom level geometry information from a face is obtained either by edge tessellation or surface triangulation. In Revit, a face is defined by its closed edge loops. The Face.EdgeLoops property returns all the bounding loops of a face in EdgeArrayArray class. Edge loops are always counter-clockwise. Each of the edge loops can be accessed by iterating this array. Further iteration will give each of the edges. The Edge.Tessellate() method gives a polyline approximation of each edges with an accuracy of about one by sixteenth inch. Once the polyline approximation is obtained, geometry extraction is complete. Figure 7 shows the overall geometry extraction from a solid in edge tessellation approach. However, this approach is only suitable for planar surfaces having no holes in it. Since, only corner point information is not sufficient for curved surfaces; surface triangulation method was followed in this project for geometry extraction. It is possible to triangulate a curved surface or a surface having more than one edge loops using surface triangulation method. Revit platform API provides surface triangulation feature through Face.Triangulate() method. This method has an argument of type double to set the mesh sizing. Its value ranges from 0 for a coarse mesh to 1 for a fine mesh. This method returns an array of meshes in the class Mesh. A mesh is a collection of triangular boundaries; which collectively forms a 3D shape. Meshes are typically encountered inside Revit element geometry if those elements were created from certain import operations and are also used in some native Revit elements such as TopographySurface. Each triangle in Mesh is a MeshTriangle object represented by its vertices. Mesh is iterated to get all the triangular facets of a face and each mesh triangle is further iterated to get the vertices of each triangle. Flow chart in figure 8 represents the data extraction flow in face triangulation method. Figure 9 shows a curved wall in TetView (TetView, 2011) window after surface triangulation. TetView is used to visualize the data readable by TetGen.

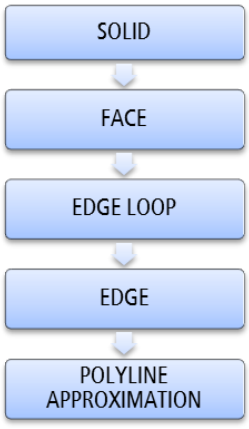

Figure 7: Edge Tessellation Method.

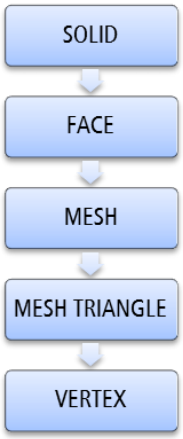

Figure 8: Surface Triangulation.

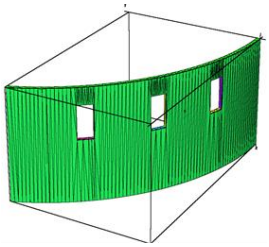

Figure 9: Triangulated Wall Surface in TetView.

2.1.3 Component Family Objects

Component Families represent classes of Family Elements, which are available for loading in project and the properties and geometric representation of which are not predefined in Revit. In the Revit Platform API, both the Family class and FamilyInstance belong to the Component Family. Families in the Revit Platform API are represented by three objects: Family, FamilySymbol and FamilyInstance. Families can be considered as templates of various objects in the design. For example, window is a family. There might be various design templates of windows. Each of the templates is FamilySymbol with specific sets of design settings. When a family symbol is loaded, there are options to choose from various sizes of it. Each size represents an instance of that symbol, called Family Instance. While extracting the geometry information, only information from the family instance is required.

Geometry extraction procedure for a family instance is similar to the procedure followed for a system family object. First, it was converted to a geometry object and then first tyre geometric objects were pulled out as depicted in figure 10. However, for a family instance, geometry element might have a number of nested geometric instances in it. A GeometryInstance represents a set of geometry objects stored by Revit in a default configuration, and then transformed into the proper location because of the properties of the element. It might contain a number of geometry objects including solids, curves, geometry instances etc. If it contains another geometry instance; then it needs to be further iterated to obtain all the geometry objects in it until core level geometry objects are retrieved. These geometry components represent all the subcomponents of the parent family instance. This is to mention that, unlike the system family objects; the coordinate system of a geometry instance is different from the project coordinate system. Therefore, coordinate system of the geometry instance was transformed to the project file's coordinate system go get the actual coordinates of the family instance. This was done by using GeometryInstance.GetInstanceGeometry() method which returns the transformed geometry instance. After that, similar procedure was followed as explained earlier for system family objects to get the geometry information from solids and faces. Figure 8 shows the geometric representation of a window through the plugin. A detailed view of a window-wall intersection is shown in figure 11.

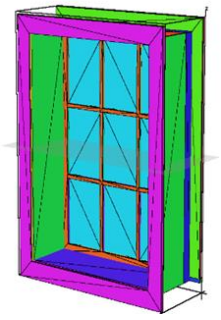

Figure 10: Window Geometry in TetView.

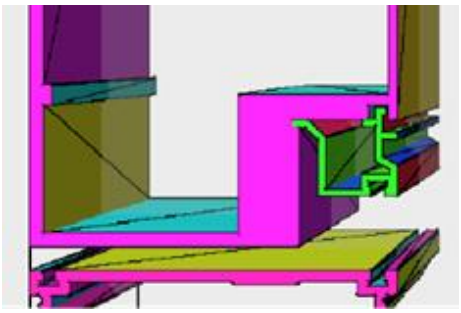

Figure 11: Window-Wall Interface in TetView.

2.1.4 Resolving Boundary Conditions

Although mesh triangles can fully represent a surface geometry, they cannot distinguish surface regions, which are in contact with different solid boundaries. For example, Figure 12 shows two solid objects adjacent to each other.

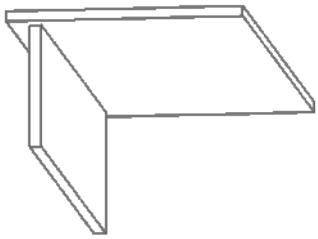

Figure 12: A Wall-Roof Intersection.

Built-in Revit feature cannot identify the region in the bottom surface of the roof, which is in touch with the top surface of the wall. To carry out heat transfer analysis, it is necessary to assign separate boundary conditions for these two regions of the bottom face of the roof. Again, the region of this surface adjacent to the wall must have the same boundary condition as the top surface of the wall has. To resolve this issue, it is needed to refine the mesh triangles of each face and identify which sub triangle is adjacent to which surface; and finally assign proper boundary marker on it. Therefore, instead of assigning boundary condition on Face level it is required to assign boundary condition on refined triangle level. This was done by recursively breaking up each mesh triangle into smaller ones until the area of the minute triangles reach a set value. The set value plays an important role in resolving the surface adjacency issue. Figures 13 and 14 display a progressive improvement on the edge regions of two meshing surfaces by changing the set value of Area Tolerance from 0.5 to 0.25.

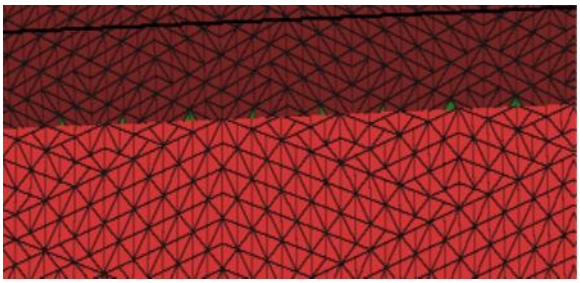

Figure 13: Using Area Tolerance 0.50.

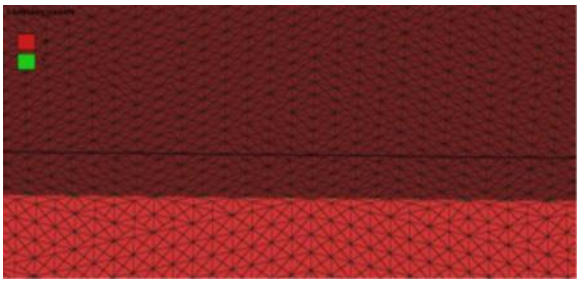

Figure 14: Using Area Tolerance 0.25.

Thin and long triangles were avoided by bisecting the largest side of the parent triangle to generate two new triangles and the same was carried out recursively. Once the sub triangles are obtained, they were checked for adjacency to each face. This was done by first calculating the centroid of each triangle and then creating a projection on each surface in the project file to measure the distance of that point from every face. If the distance was found to be zero then that triangle was considered to be in contact with the face concerned. Face.project() method allows us to make the projection; the return of which is a class called IntersectionResult. The return is null if the point of projection lies outside the surface or the projection fails. For a successful projection with the point of projection lying on the surface concerned, the distance property of the intersection result class returns the distance of the original point i.e. the centroid from the point of projection. If this value is zero then the centroid essentially lies on that face.

Otherwise, the triangle concerned is either a part of an interior surface of the room or an outside one. For the latter two cases, it is need to find out which one is true for the triangular surface concerned.

FindReferenceWithContextByDirection(); a built-in method in API provides a way to throw an imaginary ray in some predefined direction and obtain reference of the object that it intersects on its path from the ray's origin to the end point. To determine if a point on the surface lies on the inside surface of a room or outside surface; it can be considered to be lying at the centre of an imaginary cube. Then using the above feature it is possible to throw rays towards each corner point and face centres of the cube and collect the references to the objects of the intersection. If any of the results returned is null; it indicates that there is no object to intersect the ray along that direction and the triangle containing the point lies on an outside surface of the building. Otherwise, that surface belongs to an interior surface of a room. Now that it is known whether a triangle is located on an interior or an exterior surface of the room; suitable marker can be assigned to distinguish that surface. The same procedure was iterated for all other triangles.

2.1.5 Material Data

Computational heat transfer analysis on the model requires thermal properties (i.e. density, thermal conductivity, and specific heat capacity) of the element material along with geometry information. In Revit model, one element can have several elements and components. For example, FamilyInstance has SubComponents and Wall has CompoundStructure which may contain several CompoundStructureLayers.

Revit Platform API, provides following guidelines for retrieving element material information:

- If the element contains multiple elements, materials are obtained separately
- If the element contains components, materials for each component are extracted from the parameters or in a specific way
- If the component's material returns null, material is acquired from the corresponding Element.Category sub Category

According to the above guidelines: an element with multiple layers is split into multiple solid objects until each solid is consists of a single material. Then each single solid is accessed to get material information. This project considered only single layered elements for simplicity. Future work may involve multilayer elements and development of a material library for seamless computation.

2.2 Automated 3D Volume Mesh Generation

The geometry data of the three-dimensional model thus obtained is now passed to the volume mesh generator. In this project, TetGen (TetGen, 2011) was used for generating volume mesh. TetGen is a program to generate tetrahedral meshes of any 3D polyhedral domains. TetGen generates exact constrained Delaunay tetrahedralizations, boundary conforming Delaunay meshes, and Voronoi partitions. It provides various features to generate good quality and adaptive tetrahedral meshes suitable for numerical methods, such as finite element or finite volume methods. TetGen supports various input file format. For this project .poly file format was used. A poly file represents a piecewise linear complex (PLC) as well as some additional information. A PLC is a set of vertices, segments and facets as shown in figure 15.

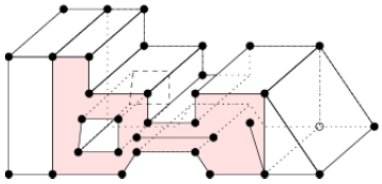

Figure 15. Piecewise Linear Complex or PLC (TetGen, 2011).

```
# Part 1 - node list
#node count, 3D, no attribute, no boundary
marker
  8  3  0  0
#node index, node coordinates
  1  0.0 0.0 0.0
  2  1.0 0.0 0.0
  3  1.0 1.0 0.0
  4  0.0 1.0 0.0
  5  0.0 0.0 1.0
  6  1.0 0.0 1.0
  7  1.0 1.0 1.0
  8  0.0 1.0 1.0
# Part 2 - facet list
# facet count, no boundary marker
  6  0
# facets
  1           # 1 polygon, no hole, no
boundary marker
  4  1 2 3 4   # front
  1
  4  5 6 7 8   # back
  1
  4  1 2 6 5   # bottom
  1
  4  2 3 7 6   # right
  1
  4  3 4 8 7   # top
```

Figure 16. Poly File Format (TetGen, 2011).

Each facet is a polygonal region; it may have any number of sides and may be non-convex, possibly with holes, segments and vertices in it. A facet can represent any planar straight-line graph (PSLG), which is a popular input model used by many two-dimensional mesh algorithms. Poly file consists of four parts: a list of points, a list of facets, a list of (volume) points and a list of region attributes. The first three parts are mandatory, but the fourth part is optional. Figure 16 shows a sample poly file format for volume mesh generation. The nodes are the vertices of the sub triangles created during geometry extraction and the facets are the sub triangles. Taking the poly file as input, TetGen refines the surface mesh and generates the volume mesh, which is used later for computational heat transfer analysis. Figure 17 shows the geometric representation of a model exported to TetGen in poly file format and viewed in TetView.

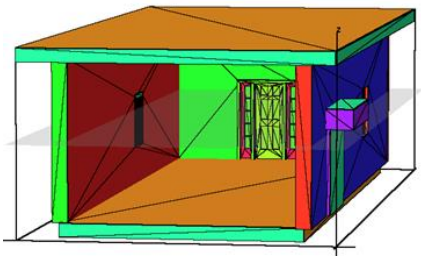

Figure 17. Whole Room Geometry in TetView.

2.2.1 3D Heat Transfer Simulation

Volume mesh data thus generated by TetGen is saved as a volume grid file; which is then converted into a format readable by heat transfer analysis tool. Next step is to run a solver on the data to carry out simulation to obtain final results. This work employs Loci-STREAM (Thakur and Wright, 2012), an evolving Computational Fluid Dynamics (CFD) tool software based on Loci framework. Loci-STREAM utilizes finite volume method to carry out 3D heat transfer analysis, etc.

Loci is a framework for application development, which is designed to reduce the complexity of assembling large-scale finite-volume applications as well as the integration of multiple applications in a multidisciplinary environment. Unlike traditional procedural programming systems (C, FORTRAN) in which one writes code with subroutines, or object-oriented systems (C++, Java), Loci uses a rule-based framework for application design. Users of Loci write applications using a collection of rules and provide an implementation for each of the rules in the form of a C++ class. In addition, the user must create a database of facts representing particular known information of the problem, such as boundary conditions. Once the rules and facts are provided, a query is made to have the system construct a solution.

One of the most interesting features of Loci is its ability to automatically determine the scheduling of events of the program to produce the answer to the desired query, as well as to test the consistency of the input to determine whether a solution is possible given the specified information. The other major advantage of Loci to the application developer is its automatic handling of domain decomposition and distribution of the problem to multiple processors. Once the results from TetGen were processed, an input was created containing details of material properties, boundary condition and solver related parameters in the simulation package environment. At this stage, the solver was built and it was finally applied on the data.

The simulation was carried out in finite volume method and three dimensional heat transfer was considered without including airflow analysis. Simulation result was exported in a portable file format readable by a suitable visualization software. In this project, the data set was visualized by using Tecplot. Figures 18 and 19 show partial view of the temperature distribution of a room boundary at inside surface and outside surface level respectively in Tecplot window.

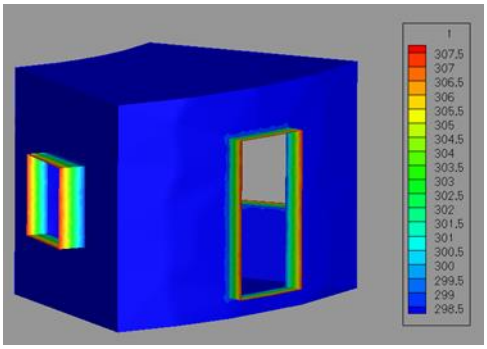

Figure 18. Partial View of Temperature Distribution of a Room Boundary (inside surface).

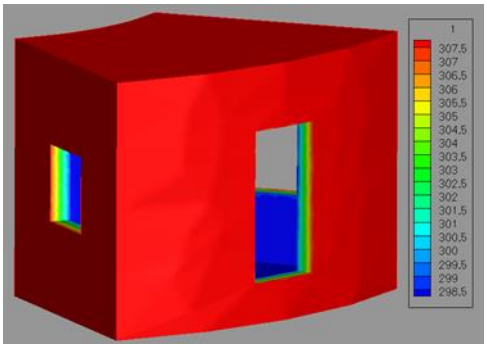

Figure 19. Partial View of Temperature Distribution of a Room Boundary (outside surface).

3 CONCLUSION AND FUTURE WORK

This paper discussed the preliminary research in Dynamic-BIM integrated 3D heat transfer analysis on building models. It introduced a new way to extract building information from a BIM tool for indepth and seamless analysis purpose. Immediate extension of this project involves including multilayer building elements into consideration, developing an efficient code and material property library for extracting material information from the model. Future work may include conjugate 3D heat transfer analysis with moisture and air transfer and above all, seamless integration of all the steps and visualizing the results in OGRE platform for a Dynamic-BIM experience.

ACKNOWLEDGMENTS

This work is supported by the UF Opportunity Seed Fund awarded by the UF Office of Research. The authors would like to thank Patrick Hughes and Mahabir Bhandari at Building Technologies Research & Integration Center (BTRIC), Oak Ridge National Laboratory (ORNL) for their collaborative support.

REFERENCES

Autodesk Revit 2013, 2012. Accessed July 18, 2012. http://usa.autodesk.com/revit

Autodesk Revit API Guide, 2012. Accessed July 19, 2012. http://wikihelp.autodesk.com/Revit/enu/2013/Help/00006-API_Developer%27s_Guide

Luke E., George T., 2005. "Loci: A Rule-Based Framework for Parallel Multidisciplinary Simulation Synthesis," *Journal of Functional Programming*, Volume 15, Issue 03, pp. 477-502, Cambridge University Press.

Bhandari M. and Srinivasan, R.S. 2102. "Window-Wall Interface Correction Factors: Thermal Modeling of Integrated Fenestration and Opaque Envelope Systems for Improved Prediction of Energy Use." In *Proceedings of Fifth National Conference of IBPSA-USA*, August 1-3.

Srinivasan, R.S., Kibert, C.J., Fishwick, P., Ezzell, Z., Thakur, S., Ahmed, I., and Lakshmanan, J. 2012. "Preliminary Researches in Dynamic-BIM (D-BIM) Workbench Development," In *Proceedings of Winter Simulation Conference*, Berlin, Germany, 2012.

Srinivasan, R.S., Kibert, C., Fishwick, P., Thakur, S., Lakshmanan, J., Ezzell, Z., Parmar, M., Ahmed, I. "Dynamic-BIM (D-BIM) Workbench for Integrated Building Performance Assessments," In *Proceedings of the Advances in Building Sciences Conference*, Madras, India, 2013.

Tecplot, 2013. Accessed February 2, 2013. www.tecplot.com

TetGen, 2011. Accessed August 23, 2012. http://tetgen.berlios.de

THERM, 2012. Accessed January 17, 2013. http://windows.lbl.gov/software/therm/therm.html

Thakur S., Wright J., 2012. An All-Speed Solver for Unsteady Reacting and Non-Reacting Flows Using a Rule-Based Framework, In *Proceedings of 42nd AIAA Fluid Dynamics Conference*, #AIAA-2012-3254.

TetView, 2011. Accessed November 15, 2012. http://tetgen.berlios.de/tetview.html

AUTHOR BIOGRAPHIES

RAVI SRINIVASAN, Ph.D., C.E.M. is an Assistant Professor of Low/Net Zero Energy Buildings at the Rinker, Sr. School of Construction Management at the University of Florida. He received his Ph.D. and M.S. in Architecture from the University of Pennsylvania and M.S in Engineering from the University of Florida. His current research focuses on the development of Dynamic Sustainability Information Modeling (D-SIM) with integrative three-dimensional building simulation and visualization platform using extensible virtual environments for net zero campuses and cities. He is currently a member of the National Fenestration Research Council (NFRC) ANS Committee, reviewer for reputed journals such as Energy, Applied Energy, Building & Environment, Energy & Buildings and International Journal of Strategic Property Management. He was a co-organizer of iiSBE Net Zero Built Environment International Conference; co-track coordinator at the 45th Winter Simulation Conference. He has chaired sessions in reputed conferences such as Winter Simulation, Building Simulation (BS 2011 held in Sydney, Australia), and Advanced Building Sciences (IIT-Madras). His e-mail address is sravi@ufl.edu

SIDDHARTH THAKUR, Ph.D., is Adjunct Associate Professor in the Department of Mechanical and Aerospace Engineering at the University of Florida. His research expertise is in the areas of Computational Fluid Dynamics, heat transfer, and combustion modeling. He has published extensively in international journals and conference proceedings. His e-mail address is sst@ufl.edu

MANOJ PARMAR, Ph.D. was a Postdoc at the Department of Mechanical and Aerospace Engineering at the University of Florida while helping with this project. His research expertise is in heat transfer. His e-mail address is ishahav@ufl.edu

ISHFAK AHMED, M.S. was a graduate student at the Department of Mechanical and Aerospace Engineering at the University of Florida while working on this project. His research expertise is in heat transfer. His e-mail address is ishahav@ufl.edu