FloorspaceJS - A New, Open Source, Web-Based Geometry Editor for Building Energy Modeling (BEM)

Preprint

Dan Macumber, Scott Horowitz,
and Marjorie Schott
National Renewable Energy Laboratory

Katie Nolan and Brian Schiller
Devetry

To be presented at Symposium on Simulation for Architecture and Urban Design (SimAUD) 2018
Delft, Netherlands
June 4-7, 2018

Suggested Citation
Macumber, Dan; Scott Horowitz, Marjorie Schott, Katie Nolan and Brian Schiller. 2018. “FloorspaceJS - A New, Open Source, Web-Based Geometry Editor for Building Energy Modeling (BEM): Preprint.” Golden, CO: National Renewable Energy Laboratory. NREL/CP-5500-70491.
https://www.nrel.gov/docs/fy18osti/70491.pdf

Conference Paper
NREL/CP-5500-70491
March 2018

Contract No. DE-AC36-08GO28308

NOTICE

This report is available at no cost from the National Renewable Energy Laboratory (NREL) at www.nrel.gov/publications.

Available electronically at SciTech Connect http:/www.osti.gov/scitech

Available for a processing fee to U.S. Department of Energy and its contractors, in paper, from:

U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831-0062
OSTI http://www.osti.gov
Phone: 865.576.8401
Fax: 865.576.5728
Email: reports@osti.gov

Available for sale to the public, in paper, from:

U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Road
Alexandria, VA 22312
NTIS http://www.ntis.gov
Phone: 800.553.6847 or 703.605.6000
Fax: 703.605.6900
Email: orders@ntis.gov

Cover Photos by Dennis Schroeder: (left to right) NREL 26173, NREL 18302, NREL 19758, NREL 29642, NREL 19795.

NREL prints on paper that contains recycled content.

FloorspaceJS—A New, Open-Source, Web-Based 2D Geometry Editor for Building Energy Modeling (BEM)

Daniel Macumber[1], Scott Horowitz[1], Marjorie Schott[1], Katie Noland[2], Brian Schiller[2]

[1]National Renewable Energy Laboratory
Golden, CO, USA
daniel.macumber@nrel.gov

[2]Devetry
Denver, CO, USA
info@devetry.com

ABSTRACT

Many industries are rapidly adopting web applications that are inherently cross platform, mobile, and easy to distribute. The Building Energy Modeling (BEM) community is beginning to pick up on this larger trend, with a small but growing number of BEM applications starting on or moving to the web. Currently, there are a limited number of open-source libraries or frameworks specifically tailored for BEM web applications. This paper presents FloorspaceJS, a new, open-source, web-based geometry editor for BEM. FloorspaceJS operates on a custom JSON file format, is written completely in JavaScript, and is designed to be integrated into a variety of applications, both web and desktop applications. FloorspaceJS allows users to define building geometry story-by-story with custom 2D floor plans appropriate for many BEM use cases.

Author Keywords

Building energy modeling; geometry editing; FloorspaceJS

ACM Classification Keywords

Applied computing~Computer-aided design

1 INTRODUCTION

Building geometry is intuitively important in building energy modeling (BEM). All BEM applications require geometry input of some type [1]. Exterior area and orientation of walls, roofs, and slabs are used to calculate heat transfer into and out of the building. Building shape and surrounding context feed shading and incident solar radiation calculations. Internal gains like lighting and equipment use are associated with interior sections of the building. Conditioning equipment serves air volumes within the building. While this list is not exhaustive, it shows many of the calculations for which geometry is used in BEM.

The level of geometric detail available for BEM analyses changes over time as a project moves from concept to construction, as shown in Figure 1. At early stages, little geometric detail about the final building is available so many assumptions must be made. For example, the U.S. Department of Energy commercial reference buildings [2] have fixed geometry for each of the 16 building types. These models are widely used to assess the technical potential of various building efficiency technologies at a national scale where individual building geometry is not known [3]. Shoebox or parameterized geometry generation methods [4] have also been developed, allowing total floor area, number of stories, and general shape parameters to be considered. These methods are useful for early design and stock analysis, where limited information about any one building's geometry is available.

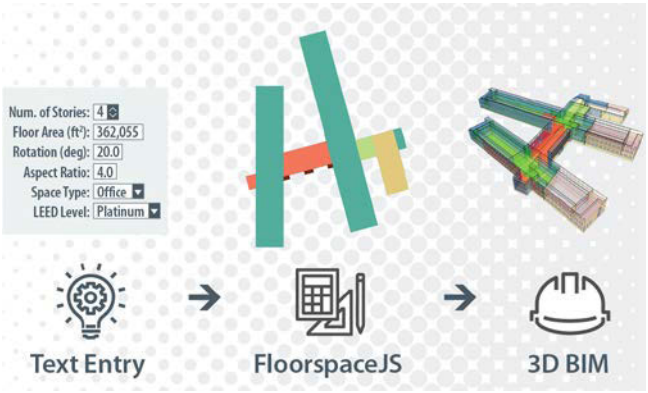

Figure 1: Building energy model Geometry Progression

At the other end of this progression, the architecture-engineering-construction-owner-operator industry is standardizing on building information modeling (BIM) [5], with detailed models available in a rich electronic format as buildings near the construction phase. A variety of algorithms aiming to ease BIM-to-BEM translations [6] [7] [8] automate the process of developing a BEM model during this process. Methods and tools are even being developed to generate energy models from images and sensor scans of existing buildings [9].

However, many BEM use cases such as conceptual design, energy code compliance, and project screening fall into the middle of the progression shown in Figure 1. More geometric information (e.g., footprint, window location, space plan, etc.) may be available than can be incorporated into a parameterized shoebox model. However, a detailed BIM model may not be available. Users may want to incorporate their building's unique geometry into the energy model but may not be ready to invest the time required to develop a full BIM representation. For example, residential building rating tools are applied to existing homes which most often do not have BIM models available. Rather than develop full BIM models, current residential BEM applications often require users to enter areas and orientations for floor, walls, roofs,

and windows as text-based inputs [10], which is both tedious and error prone.

These use cases require an easy-to-use interface to quickly develop geometry for BEM analyses. Several desktop applications have embedded editors that meet these needs [11] [12]. However, to date, none of these editors have been available as open-source widgets based on web technologies.

FloorspaceJS is an open-source widget focused on this section of the BEM geometry progression and designed to enable a range of new energy modeling applications in both the public and private sectors. FloorspaceJS is not intended as a replacement for BIM. As BIM adoption continues to increase, the number of use cases requiring a simplified tool like FloorspaceJS may decrease.

2 DESIGN

FloorspaceJS is a 2D geometry editor. Users can define an explicit floor plan for each story of a building. A story-by-story interface makes it easy to develop space geometry and assign properties. Referencing satellite imagery or floor plan images, when available, speeds up geometry entry. Floor plans can be extruded to create 3D geometry. Separate software to perform this extrusion is discussed in Section 6. Explicit floor plans allow more building-specific information than parameterized shoeboxes but less information than a full 3D BIM model. Split-level buildings, multileveled spaces, and type of roof geometry can also be defined by advanced users.

Sloped walls, complex roofs, detailed shading structures, and other complex 3D structures are out of FloorspaceJS's scope. In general, if users have a 3D BIM model in a tool that can export a useful BEM representation, then it is better to use that export than to recreate a new model using FloorspaceJS.

Reusability and minimal dependencies were key design considerations for software developers. Web technologies can be used in both online and desktop applications. The editor was written in pure JavaScript for maximum portability and reusability. A custom JavaScript Object Notation (JSON) file format was developed to ease integration with other applications. Custom JSON schema design was a key part of FloorspaceJS development.

3 SCHEMA

Extensible Markup Language (XML) schemas are a well-known technology for validating that XML files conform to a certain set of rules and requirements. The Green Building XML schema (gbXML) is a well-known XML schema within the BEM community [13]. JSON schemas [14] provide a method to validate JSON files, equivalent to the use of XML schemas to validate XML files. Specifying the FloorspaceJS file format in JSON schema facilitates JSON file validation as well as communication with third party application developers. In this way, the JSON schema exists as both documentation and as a functional and testable software product. The JSON schema for FloorspaceJS is available online [15].

The root level of the FloorspaceJS schema describes project-wide settings such as the relative angle between the drawing coordinate system and true North. The model unit system is specified at the project level and can be SI or IP. Project-wide non-geometrical objects such as thermal zones, space types, and construction sets are defined at the root level and referenced throughout the model. Similarly, a set of component definitions for objects such as windows are also defined at the root level. These component objects have nominal geometry such as height and width. Instances of these component objects may be placed on geometry throughout the building.

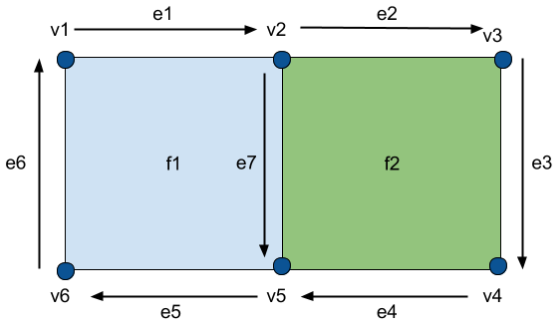

Figure 2. Simple geometry example

All geometry for the model is defined in a list of building story objects at the root level. Each building story includes a geometry object that defines the vertices, edges, and faces associated with that story. Figure 2 shows a simple geometry object with two faces, f1 and f2. Each face is defined by an ordered list of edges, and each edge is defined by two vertices. Two faces may share the same edge but traversed in opposite direction such as edge e7 in Figure 2.

The FloorspaceJS geometry schema was designed to promote proper second-level space boundaries [5] between spaces on the same building story. This is important as BEM applications must be able to tell which surfaces are exposed to outdoor conditions and which are interior surfaces shared by two spaces. In the FloorspaceJS representation, an edge that is shared by two faces becomes an interior wall, and an edge referenced by one face becomes an exterior wall. The FloorspaceJS schema does not explicitly address space boundaries between spaces on different stories. These must be established using techniques such as polygon intersection between geometry on different stories. However, the FloorspaceJS editor automatically displays vertices and edges from the previous story for reference when drawing a new story. This helps maintain vertical alignment between spaces of different stories, improving the outcome of polygon intersection algorithms. The geometry object shown in Figure 1 is represented in JSON format as:

```
"geometry": {
  "vertices": [
    {"id": "v1", "x": 0, "y": 10},
```

```
      {“id": "v2", "x": 10, "y": 10},
      {“id": "v3", "x": 20, "y": 10},
      {“id": "v4", "x": 20, "y": 0},
      {“id": "v5", "x": 10, "y": 0},
      {“id": "v6", "x": 0, "y": 0}],
   "edges": [
      {“id": "e1", "vertex_ids": ["v1", "v2"]},
      {“id": "e2", "vertex_ids": ["v2", "v3"]},
      {“id": "e3", "vertex_ids": ["v3", "v4"]},
      {“id": "e4", "vertex_ids": ["v4", "v5"]},
      {“id": "e5", "vertex_ids": ["v5", "v6"]},
      {“id": "e6", "vertex_ids": ["v6", "v1"]},
      {“id": "e7", "vertex_ids": ["v2", "v5"]}],
   "faces": [
      {“id": "f1", "edge_ids": ["e1", "e7", "e5", "e6"]},
      {“id": "f2","edge_ids": ["e2", "e3", "e4", "e7"]}]
 }
```

Each building story defines a list of spaces, each of which references a single face in the building story’s geometry. The face solely represents geometry while the space contains non-geometric BEM information. For example, each space references a thermal zone, space type, construction set, and other top-level objects. The building story also specifies a below floor plenum height, floor to ceiling height, and above ceiling plenum height; these properties can also be overridden at the space level if needed. Each face is extruded by these heights when converting the 2D FloorspaceJS model to a 3D representation, as discussed in Section 6, potentially creating up to three volumes for each face (two plenums and the occupied space). Contextual shading objects such as surrounding buildings are specified by extruding faces assigned to shading objects. Window components are placed on an edge with a parameter named alpha describing how far along the edge the window’s center is. A one-story example with two spaces and the geometry shown in Figure 2 is given below (with some fields omitted for brevity):

```
"stories": [{
   "id": "story1",
   "name": "Story 1",
   "image_visible": false,
   "below_floor_plenum_height": 0,
   "floor_to_ceiling_height": 12,
   "above_ceiling_plenum_height": 0,
   "multiplier": 1,
   "geometry": {...},
   "spaces": [{
      "id": "space1",
      "name": "Space 1",
      "face_id": "f1",
      "thermal_zone_id": "zone1",
      "space_type_id": "spacetype1",
      "construction_set_id": "construction1",
      …
   }, {
      "id": "space2",
      "name": "Space 2",
      "face_id": "f2",
      "thermal_zone_id": " zone1",
      "space_type_id": " spacetype2",
      "construction_set_id": " construction1",
      …
   }],
   "shading": [],
   "windows": [{
      "window_definition_id": "window1",
      "edge_id": "e5",
      "alpha": 0.5,
      …
   }]
```

The one-story example above includes a single window placed halfway along the edge e5. The geometry of the window is defined in a separate window definition that can be reused throughout the model. Examples of three different types of window definitions are given below.

```
"window_definitions": [{
   "id": " window1",
   "name": "Large Window",
   "window_definition_type": "Single Window",
   "wwr": null,
   "sill_height": 3,
   "window_spacing": null,
   "height": 8,
   "width": 12,
   "overhang_projection_factor": null,
   "fin_projection_factor": null
 },{
   "id": " window2",
   "name": "40% WWR",
   "window_definition_type": "Repeating Windows",
   "wwr": null,
   "sill_height": 3,
   "window_spacing": 6,
   "height": 4,
   "width": 4,
   "overhang_projection_factor": 0.5,
   "fin_projection_factor": 0.5
 }, {
   "id": " window3",
   "name": "40% WWR",
   "window_definition_type": "Window to Wall Ratio",
   "wwr": 0.4,
   "sill_height": 3,
   "window_spacing": null,
   "height": null,
   "width": null,
```

```
  "overhang_projection_factor": null,
  "fin_projection_factor": null
}]
```


4 EDITOR FEATURES

The latest version of the FloorspaceJS editor is available to try online [16]. Currently, FloorspaceJS works best when using the Chrome browser; other browsers are not fully tested. When starting a new project, users are prompted to either create a new floor plan, create a new floor plan with a map background, or open an existing floor plan. If creating a new floor plan with a map, users select the location and orient the drawing grid to align with the building's axes as shown in Figure 3. The example model developed in this section, shown in Figures 4-8, is available as a digital file accompanying this paper and may be opened as an existing floor plan for inspection.

After orienting the drawing area to the map, users press "Done" to begin drawing. If desired, users can import a story-specific floor plan image by selecting the correct story and choosing "Image" in the navigator. Users can move and resize the image using the background map or drawing grid as a reference as shown in Figure 4. Currently, only image formats can be imported; importing other formats such as Adobe PDF is a possible feature for future development.

Once background information is in place, users begin to add spaces to the selected story. Users create and select spaces in the left-hand navigator, then use the drawing tools to add rectangular or polygon geometry to each space. Newly drawn geometry is added to any existing geometry in the space, the operation is cancelled if the resulting geometry is invalid (e.g., the result has multiple non-adjacent polygons or interior holes). The fill tool adds any geometry clicked to the currently selected space, which is particularly useful for quickly replicating similar floor plans across stories. The eraser tool removes any overlapping geometry in any space.

More detailed space properties, including space name, are edited by expanding the space grid view using the green arrow keys next to the space dropdown.

After space geometry is defined, users move to the assignments tab to create building units, thermal zones, space types, construction sets, and pitched roofs. These objects are assigned to spaces by clicking on the appropriate spaces for a selected object (or via the detailed space properties).

Finally, users create windows or other component definitions (e.g., daylighting controls, doors, etc.) via the components tab. Window component definitions have geometric properties such as height, width, and sill height. Once component definitions are complete, users can place instances of these components throughout the building. Window definitions can represent individual windows, banded windows defined by window-to-wall ratio, and repeating windows that repeat at regular intervals. The last two types of window component definitions are applied to an entire edge rather than a single point.

5 WEB TECHNOLOGIES

FloorspaceJS is written entirely in JavaScript with a minimal set of dependencies. The code, available online [17], is distributed under a 3-clause BSD software license [18] that allows it to be reused in a wide range of applications. FloorspaceJS transpiles modern ES2017 syntax using Babel.js [19] and Webpack [20] to produce code that runs in today's browsers. It uses the Vue.js framework [21] with Vuex [22] for state management, OpenLayers [23] for map integration, and d3 [24] for drawing.

To use the widget in a third-party application, a developer downloads an embeddable version of the widget from the release pages [25] and includes this code in their application as an iframe. An initialization Application Programming Interface (API) is available to allow the developer to customize the widget for their application (e.g., SI vs IP units). Finally, the developer calls the import and export APIs to pass data back and forth in the FloorspaceJS JSON format. An example integration of the widget into a web application is available online [26].

6 TRANSLATION TO THREE DIMENSIONS

The FloorspaceJS JSON format describes 2D geometry only. Many BEM tools, including EnergyPlus [27], require 3D geometry for their calculations. Therefore, a translation from the 2D FloorspaceJS JSON to a 3D format is required. In theory, each software application utilizing the FloorspaceJS widget could implement its own translation functionality. However, this would place a higher burden on software applications leveraging the FloorspaceJS widget and would also potentially result in inconsistent translations. For this reason, a reference 2D FloorspaceJS JSON to 3D translation capability has been added to the OpenStudio software development kit (SDK) [28].

Several 3D BEM formats were considered for translation output including gbXML, EnergyPlus Input Data Format (IDF), and OpenStudio Model (OSM) formats. In the end, the ThreeJS JSON model format 3 [29] was selected as the translation output format as it can be readily previewed using web technologies and can support additional application specific properties. Translation from 3D ThreeJS JSON to gbXML, IDF, and OSM is relatively straightforward. Translation from 3D ThreeJS JSON to OSM format has been implemented in the OpenStudio SDK, which itself has existing OSM to gbXML and IDF translators.

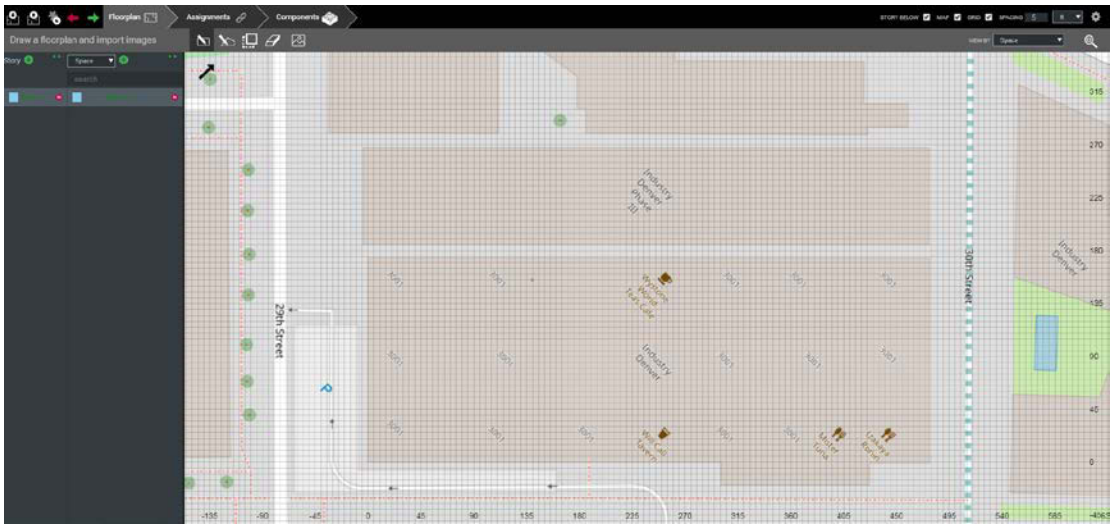

Figure 3. Initial map view

Figure 4. Image overlaid on background map

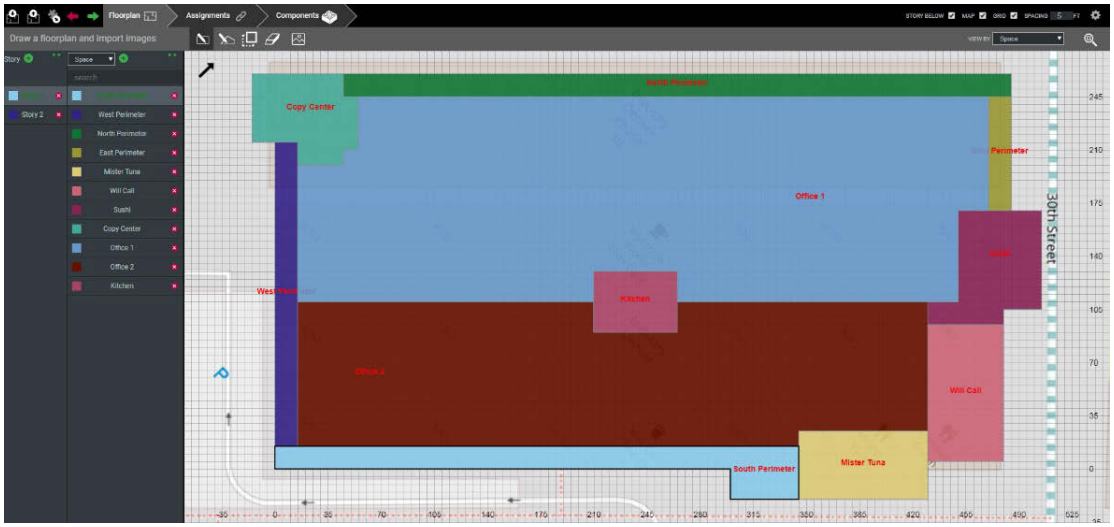

Figure 5. Space geometry

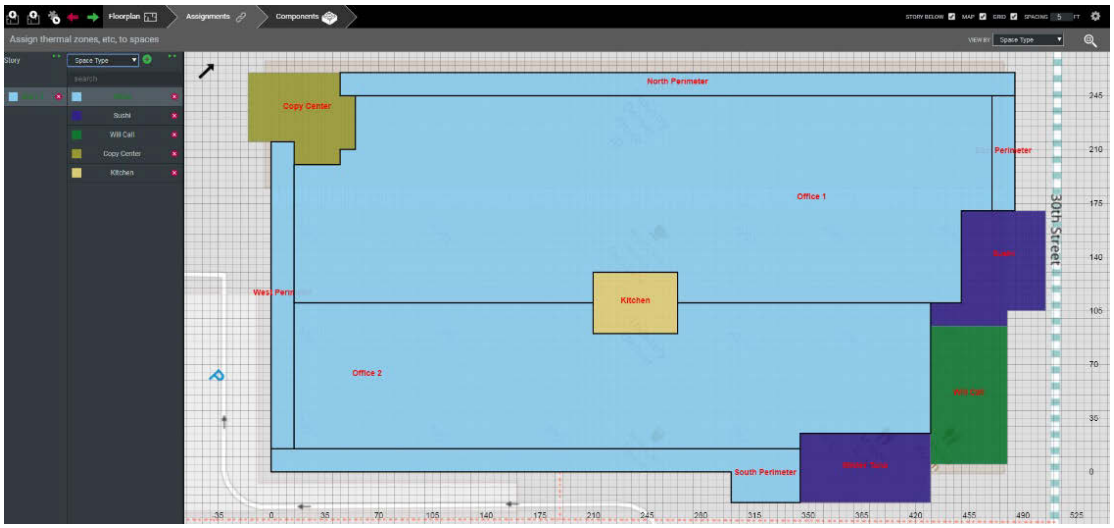

Figure 6. Space property grid view

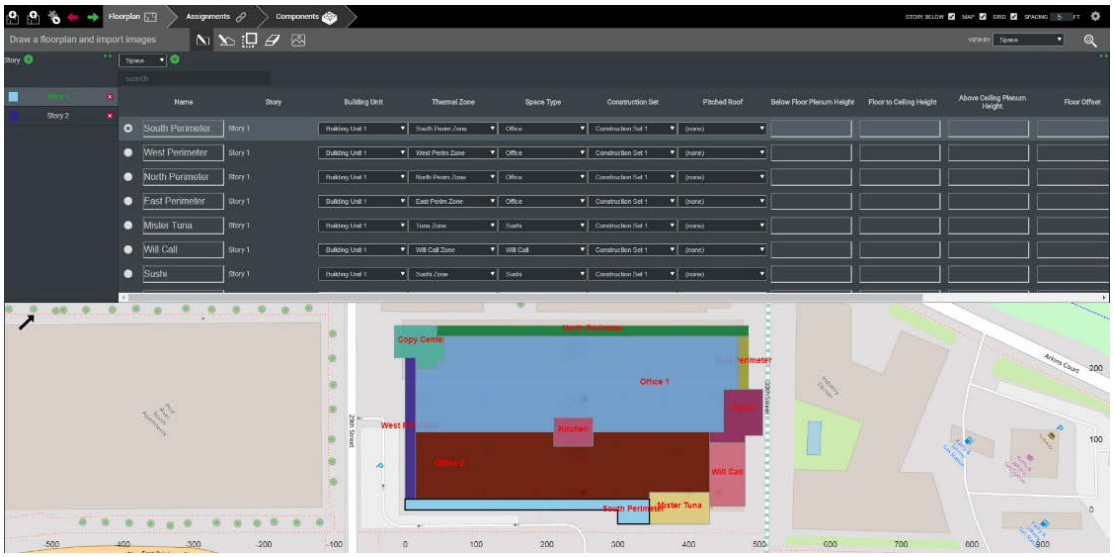

Figure 7. Thermal zoning

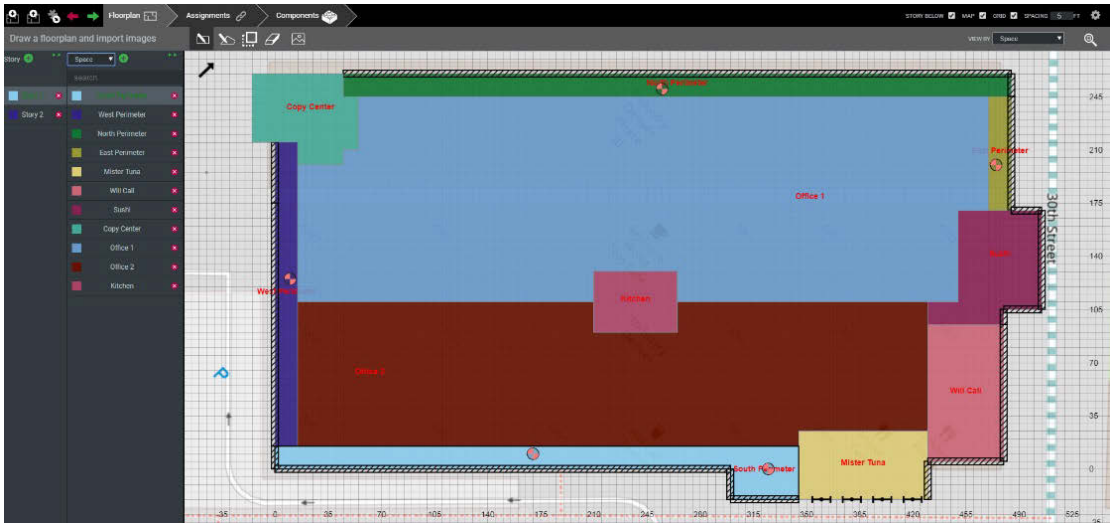

Figure 8. Window and daylight sensor placement

By carefully restricting the software dependencies used in the FloorspaceJS JSON to ThreeJS JSON conversion, the authors have extracted all the code necessary for this translation out of the OpenStudio SDK and into a smaller standalone C++ library. A proof of concept was completed to compile this library to pure JavaScript using Emscripten [30]. The resulting JavaScript library can then be used to translate FloorspaceJS JSON to ThreeJS JSON entirely in JavaScript using the client browser. Figure 9 shows a small web application which has translated the example FloorspaceJS JSON developed in Section 4 and displays the ThreeJS JSON output in a web browser.

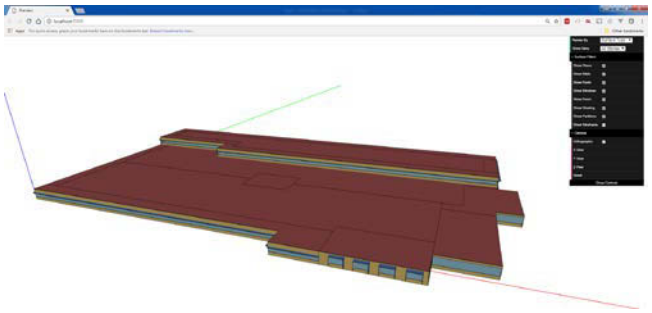

Figure 9. Web Based 3D Preview

This web-based translation proof of concept has not yet been released or incorporated into the FloorspaceJS widget. At present, web-based translation can only support translation to ThreeJS JSON, it cannot produce OSM, IDF, or gbXML as the additional dependencies of those conversions have not been compiled using Emscripten. However, all translation functionality is available in the OpenStudio SDK for the C++, C#, and Ruby languages. An OpenStudio Measure [31] for converting FloorspaceJS JSON to OSM is also available.

7 FUTURE WORK

FloorspaceJS makes two major contributions to the BEM community. First, an open-source, web-enabled geometry editor widget lowers the barrier to developing BEM desktop or web applications. Second, the FloorspaceJS JSON schema enables developers to integrate the editor with their own applications or leverage its output in custom workflows.

Future work may include continued development of web-based 2D to 3D translation functionality. Currently, certain functionalities, such as adjacent space detection, are not yet available in the proof of concept web-based translation. Once these features are stable, the web-based translator could be easily integrated with the FloorspaceJS widget to provide a pure JavaScript widget which outputs 3D geometry in ThreeJS JSON format.

The FloorspaceJS developers hope that this contribution can be leveraged by the BEM community in new and creative ways. One possibility is the large body of work being developed around extracting building footprint data from geographic information system (GIS) information [32] [33]. Extruding the exterior building footprint alone generally does not provide sufficient geometric detail for BEM as it does not include separate perimeter and core heating, ventilating, and air-conditioning zones designed for most large buildings. There are methods [34] [35] [36] [37] to apply automatic thermal zoning algorithms to arbitrary building footprints, which will be useful for converting building footprints to usable energy models at city scale. It may be possible for these automatic zoning algorithms to provide output in FloorspaceJS format. In this way, the energy model could be refined and updated as needed in further manual processes.

8 ACKNOWLEDGEMENTS

The authors would like to thank the U.S. Department of Energy's Building Technologies Office and the Bonneville Power Administration for funding the initial development of FloorspaceJS.

REFERENCES

1. D. B. Crawley, J. W. Hand, M. Kummert, and B. T. Griffith, "Contrasting the capabilities of building energy performance simulation programs," *Building and environment,* vol. 43, no. 4, 2008.
2. M. Deru, K. Field, D. Studer, K. Benne, B. Griffith, P. Torcellini, B. Liu, M. Halverson, D. Winiarski, M Rosenberg, M. Yazdanian, J. Huang, and D. Crawley, "US Department of Energy commercial reference building models of the national building stock," 2011.
3. B. Griffith, N. Long, P. Torcellini, R. Judkoff, D. Crawley, and J. Ryan, "Assessment of the technical potential for achieving net zero-energy buildings in the commercial sector," National Renewable Energy Laboratory (NREL), Golden, CO., 2007.
4. M. Heidarinejad, N. Mattise, M. Dahlhausen, K. Sharma, K. Benne, D. Macumber, L. Brackney, and J. Srebric, "Demonstration of reduced-order urban scale building energy models," *Energy and Buildings,* vol. 156, 2017.
5. V. Bazjanac, "Space boundary requirements for modeling of building geometry for energy and other performance simulation," in *CIB W78: 27th Int. Conf.*, 2010.
6. C. M. Rose and V. Bazjanac, "An algorithm to generate space boundaries for building energy simulation," *Engineering with Computers,* vol. 31, no. 2, 2015.
7. W. Yan, M. Clayton, J. Haberl, J. WoonSeong, J. Bun Kim, K. Sandeep, J. Bermudez, and M. Dixit, "Interfacing BIM with building thermal and daylighting modeling," in *13th Int. Conf. of the Int. Building Performance Simulation Assoc.*, 2013.

8. N. Yu, Y. Jiang, L. Luo, S. Lee, A. Jallow, D. Wu, J. I. Messner, R. M. Leicht, and J. Yen, "Integrating BIMserver and OpenStudio for energy efficient building," *Computing in Civil Engineering,* 2013.
9. Y. K. Cho, Y. Ham, and M. Golpavar-Fard, "3D as-is building energy modeling and diagnostics: A review of the state-of-the-art," *Advanced Engineering Informatics,* vol. 29, no. 2, 2015.
10. [Online]. Available: http://www.remrate.com/
11. James J. Hirsch and Associates, "eQuest Introductory Tutorial, version 3.64," 2010. [Online]. Available: http://doe2.com/download/equest/eQ-v3-64_Introductory-Tutorial.pdf.
12. E. Wilson, "Using BEopt to Optimize Home Energy Performance," *Home Energy,* July/August 2015.
13. [Online]. Available: http://gbxml.org/
14. [Online]. Available: http://json-schema.org/
15. [Online]. Available: https://github.com/NREL/floorspace.js/blob/develop/schema/geometry_schema.json
16. [Online]. Available: https://nrel.github.io/floorspace.js/
17. [Online]. Available: https://github.com/NREL/floorspace.js
18. [Online]. Available: https://opensource.org/licenses/BSD-3-clause
19. [Online]. Available: https://babeljs.io/
20. [Online]. Available: https://webpack.js.org/
21. [Online]. Available: https://vuejs.org/
22. [Online]. Available: https://vuex.vuejs.org/
23. [Online]. Available: https://openlayers.org/
24. [Online]. Available: https://d3js.org/
25. [Online]. Available: https://github.com/NREL/floorspace.js/releases
26. [Online]. Available: https://nrel.github.io/floorspace.js/embedded.html
27. D. B. Crawley, L. K. Lawrie, F. C. Winkelmann, W.F. Buhl, Y. J. Huang, C. O. Pedersen, R. K. Strand, R. J. Liesen, D. E. Fisher, M. J. Witte, and J. Glazer, "EnergyPlus: creating a new-generation building energy simulation program," *Energy and Buildings*, vol. 33, no. 4, 2001.
28. R. Guglielmetti, N. Long, and D. Macumber, "OpenStudio: An Open Source Integrated Analysis Platform," *Proc. Building Simulation 2011: 12th Conf. Int. Building Performance Simulation Assoc.*, 2011.
29. [Online]. Available: https://github.com/mrdoob/three.js/wiki/JSON-Model-format-3.
30. [Online]. Available: https://github.com/kripken/emscripten.
31. A. Roth, D. Goldwasser and A. Parker, "There's a measure for that!," *Energy and Buildings,* vol. 117, 2016.
32. M. Brédif, O. Tournaire, B. Vallet, and N. Champion, "Extracting polygonal building footprints from digital surface models: A fully-automatic global optimization framework," *ISPRS journal of photogrammetry and remote sensing,* vol. 77, 2013.
33. O. Tournaire, M. Brédif, D. Boldo, and M. Durupt, "An efficient stochastic approach for building footprint extraction from digital elevation models," *ISPRS Journal of Photogrammetry and Remote Sensing,* vol. 65, no. 4, 2010.
34. T. Dogan and C. Reinhart, "Shoeboxer: An algorithm for abstracted rapid multi-zone urban building energy model generation and simulation," *Energy and Buildings,* vol. 140, 2017.
35. T. Dogan, C. Reinhart, and P. Michalatos, "Autozoner: an algorithm for automatic thermal zoning of buildings with unknown interior space definitions," *Journal of Building Performance Simulation,* vol. 9, no. 2, 2016.
36. E. Rodrigues, A. R. Amaral, A. R. Gaspar, Á. Gomes, M. C. Gameiro da Silva, and C. H. Antunes, "GerAPlanO-A new building design tool: design generation, thermal assessment and performance optimization," *Energy for Sustainability,* 2015.
37. Y. Chen, T. Hong, and M. A. Piette. "Automatic Generation and Simulation of Urban Building Energy Models Based on City Datasets for City-Scale Building Retrofit Analysis," *Applied Energy,* vol. 205, 2017.