Multimed Tools Appl (2019) 78:28575–28590
https://doi.org/10.1007/s11042-018-5820-0

Cross-platform online visualization system for open BIM based on WebGL

Xiaoping Zhou[1] · **Jia Wang**[1] · **Ming Guo**[2] · **Zhe Gao**[1]

Received: 11 January 2018 / Revised: 1 February 2018 / Accepted: 19 February 2018 /
Published online: 8 March 2018

Abstract Online BIM visualization system is the first and primary task for building web applications of BIM. Light-weighted, cross-platform and open are three basic rules when building online BIM visualization systems. Currently, some efforts have been made on BIM visualization. However, most of them are designed for local BIM, and the online ones neglect the network load (without light-weighted) or are platform dependent (without cross-platform). This study develops a novel online BIM visualization system based on IFC and WebGL, termed as WebBIM. WebBIM firstly converts the raw IFC geometry data into triangles, light-weights the BIM geometry data by sharing the geometry data among the object instances generated from the same facility component, compresses the geometry, and directly renders the decompressed triangular BIM data in web browsers. Finally, empirical studies from extensive real projects' BIM data show that WebBIM is efficient, capable of visualization of large BIM files and compatible with mainstream devices.

Keywords Cross-platform · Open BIM · Online visualization system · WebGL

1 Introduction

Building information model/modeling (BIM) is a comprehensive representation of buildings. By providing a shared knowledge resource for information about a facility, BIM allows the

✉ Xiaoping Zhou
lukefchou@gmail.com

✉ Jia Wang
wangjia@bucea.edu.cn

✉ Ming Guo
guoming@bucea.edu.cn

[1] Beijing Key Laboratory of Intelligent Processing for Building Big Data, Beijing University of Civil Engineering and Architecture, Beijing 100044, China

[2] Institute of Mapping and Urban Spatial Information, Beijing University of Civil Engineering and Architecture, Beijing 102616, China

data interoperability of all the data and enables the collaboration among the stakeholders during its lifecycle [9]. Due to the significant values of BIM for the Architecture, Engineering and Construction (AEC) industries, an avalanche of studies and applications on BIM emergence in the past decade. Evidently, an increasing number of companies from AEC have embraced BIM as a common paradigm for their projects [2].

Traditional tools on BIM are often released as software, which are installed and used in a local operation system (OS), e.g., Autodesk Revit, Bentley Microstation. In this manner, the BIM data are stored and managed locally and termed as local BIM. Obviously, online BIM systems can be beneficial for all the participants, because the cloud deployment of the BIM system ensures the consistent of the BIM data in real time and facilitates the collaboration among the stakeholders. Undoubtedly, the first and primary task confronted by all these online BIM systems is to visualize the BIM online.

The online BIM visualization system is different from that on local BIM in the following aspects:

1. **Light-weighted**. The network, through which the data interact between server side and client side, is a significant factor for the online BIM visualization system. Thus, the light-weighted BIM data can improve the user experience of the visualization system.
2. **Cross-platform**. The cross-platform requirement should be satisfied in an online BIM visualization system. The prevalence of mobile devices and the advancement of information technology trigger the demand of cross-platform BIM visualization for different terminals. Different users desire to collaborate in different OS with different devices. However, the local BIM visualization can only be applicable for specified OS environment, because different OS provides different programming interfaces.
3. **Open**. To support BIM from different BIM design tools, it is vital that the online BIM visualization system utilizes the third-party open standard.

Apparently, many 3D engines support cross-platform development, e.g., Unity.[1] However, these techniques require adjustments to meet different OS environments. Web Graphic Library (WebGL) is a cross-platform, royalty-free API used to create 3D graphics in a Web browser [16]. Currently, WebGL has been integrated across popular browsers in extensive OSes. Consequently, WebGL is the preferred technique for online BIM visualization system.

The Industry Foundation Classes (IFC) is a platform neutral, open file format specification for BIM [14]. Since IFC is an official International Standard ISO, the online visualization system for open BIM in this paper is built on IFC format to support more BIM from different BIM design tools.

Although some efforts [4–6, 10, 15] have been made on the BIM visualization based on IFC, the factors of the network and the limited resource of the mobile devices have been neglected, hindering their applications in extensive scenarios. In this paper, we developed a cross-platform online visualization system for open BIM using WebGL, termed as WebBIM.

The contributions of this paper include:

1. Propose strategies for light-weighting the open BIM data. The geometry of an IFC object and the geometry references among the IFC objects are fully analyzed. For any IFCObject, the geometry is composed by two parts in description: IFCLocalPlacement and IFCProductDefinitionShape. IFCProductDefinitionShape defines the geometry of the

[1] http://unity3d.com

object, while IFCLocalPlacement defines the location. Based on this analysis, some strategies, e.g., sharing the geometry, compressing the geometry data, are applied to light-weight the BIM.

2. Develop a novel cross-platform online visualization system for open BIM, termed as WebBIM. WebBIM contains three parts: Triangle Convertor Model, Instance Light-Weighting Model and WebGL-Render Model.
3. Evaluate WebBIM by real projects' data systematically. We verified that WebBIM is capable of rendering BIM with more than 4 million IFC objects, 37 million triangle faces and 0.1 billion vertices. We also confirmed that WebBIM is compatible with the popular OS, including Windows, iOS, Android.

The remainder of the paper is organized as follows. Section 2 provides the necessary definitions and preliminaries. Section 3 discusses the related works, and Section 4 presents the cross-platform online visualization system of open BIM. Section 5 conducts the empirical studies, and Section 6 concludes the work.

2 Definitions and preliminaries

This section provides relevant notations and definitions and briefly introduces WebGL and Industry Foundation Classes (IFC).

2.1 Notations and definitions

Definition 1(Facility Component) A facility component is a kind of typical component can be reused in design and mass produced in real world. In this paper, C_i represents facility component i, and $C(j)$ i is the j-th instance of C_i in a AEC project.

Figure 1 presents a demonstration of the facility component. Since the four chairs have the same geometry and can be mass produced by a same model in factory, the four chairs are instances of the same chair facility component. Similarly, ground tile is another facility component in Fig. 1.

Definition 2 (BIM) A BIM, denoted as B, is a set of facilities or object instances, which are produced by facility components.

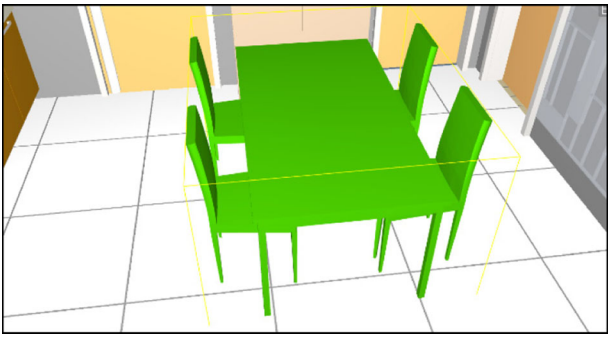


Fig. 1 Example of facility component

In this study, we use IFC as the standard BIM file. Thus, an IFC file is also denoted as B.

Definition 3 (WebBIM) WebBIM is the cross-platform online visualization system for open BIM based on WebGL developed in this study.

WebBIM light-weights and compresses the BIM data to mitigate the network load for online visualization, employs WebGL as the render engine to support different hardware and OS environments from different platforms, and uses IFC as the basic BIM data format to support more BIM design tools.

2.2 WebGL

WebGL is a cross-platform, royalty-free API used to create 3D graphics in a Web browser. Based on OpenGL ES 2.0, WebGL uses the OpenGL shading language, GLSL, and offers the familiarity of the standard OpenGL API. Because it runs in the HTML5 Canvas element, WebGL has full integration with all Document Object Model (DOM) interfaces. Thus, WebGL is a DOM API and can be used from any DOM-compatible language: JavaScript, Java, or—if you embed WebKit into an application on a Mac—Objective C.

Currently, WebGL offers a number of advantages, among them:

- An API that is based on a familiar and widely accepted 3D graphics standard.
- Cross-browser and cross-platform compatibility.
- Tight integration with HTML content, including layered compositing, interaction with other HTML elements, and use of the standard HTML event handling mechanisms.
- Hardware-accelerated 3D graphics for the browser environment.
- A scripting environment that makes it easy to prototype 3D graphics—you don't need to compile and link before you can view and debug the rendered graphics.

Obviously, WebGL naturally meets the cross-platform satisfaction for the online BIM visualization system.

2.3 IFC

The IFC data model intends to describe building and construction industry data. It is an object-based file format with a data model developed by buildingSMART (formerly the International Alliance for Interoperability, IAI) to facilitate interoperability in the AEC industry, and is a commonly used collaboration format in BIM based projects. The IFC model specification is registered by ISO and is an official International Standard ISO 16739:2013.

Because of its focus on ease of interoperability between software platforms, many governments, organizations and companies have made the use of IFC format(s) compulsory for all or partial building projects, e.g., the Danish government, the Finnish state-owned facility management company Senate Properties, the Norwegian Government, Health and Defense client organizations, to name a few.

IFC defines an EXPRESS based entity-relationship model consisting of several hundred entities organized into an object-based inheritance hierarchy. Examples of entities include building elements such as IfcWall, geometry such as IfcExtrudedAreaSolid, and basic constructs such as IfcCartesianPoint. Below is an example of wall object.

#4=IFCWALLSTANDARDCASE('3vB2YO$MX4xv5uCqZZG05x', #2, 'Wall xyz', 'Description of Wall', $, #46, #51, $);

"#4"is the line number of the IfcWallStandardCase. The internal member is initialized by the brackets of eight parameters separated by commas. The "3vB2YO$MX4xv5uCqZZG05x" is GUID of the IfcWallStandardCase instance. The second parameter "#2" refers to an IfcOwnerHistory. "Wall xyz" is the name of the wall object, followed by its description. "#46" refers to an IfcLocalPlacement object which defines the object coordinate system of the wall. "#51" is an IfcProductDefinitionShape object defining the shape of the wall object.

Obviously, IFC files contains both the regular entities and the relations between them. For example, an IFC file is usually structured from an IfcProject root instance, which is then decomposed into many products, such as IfcWall, IfcBeam, etc., by several reference layers.

The geometry data of an IFC file is also organized by several IFC entities and relationships among them. The IFC has the ability to represent a wide range of geometry. Table 1 lists the supported geometry types supported in IFC. Curve2D, GeometricSet, GeometricCurveSet are the description models of point, line and surface, respectively. SurfaceModel represents the surface model, while SolidModel describes the solid model. In details, SolidModel can be divided into SweptSolid, Brep, CSG, Clipping, AdvancedSweptSolid. Figure 2 shows an example of a wall object. SweptSolid, where the cross section (IFCRectangleProfileDef), stretching direction (IFCDirection) and stretching distance (IFCCartesianPoint) are defined, is employed to represent the wall object here. Undoubtedly, these models provide the powerful tools to represent different objects. However, to render the objects described in these models, further computation is required.

Due to the limitation computation resources in mobile devices and the resource constraint in web browsers, it is evidently impossible to render the raw geometry model in IFC in a cross-platform online BIM visisualization system. Triangles are simple geometry type which can be renderred directly in most software (and especially 3D viewers). Thus, we use triangles as intermediary geometry type in WebBIM.

3 Related works

This section presents a short review on BIM visualization tools.

Online BIM visualization system is a fundamental task of building web applications. Currently, some efforts have been made in both application and theory. Application

Table 1 The geometry models in IFC format

Geometry Type	Short Description
Curve2D	Two-dimensional curve
GeometricSet	Set of points, curves and faces
GeometricCurveSet	Set of points and curves
SurfaceModel	Surface model
Solid Model	Solid model
SweptSolid	Swept solid generated by stretch and rotation
Brep	Boundary representation
CSG	Constructive solid geometry
Clipping	Computational binary solid geometry
AdvancedSweptSolid	Advanced SweptSolid

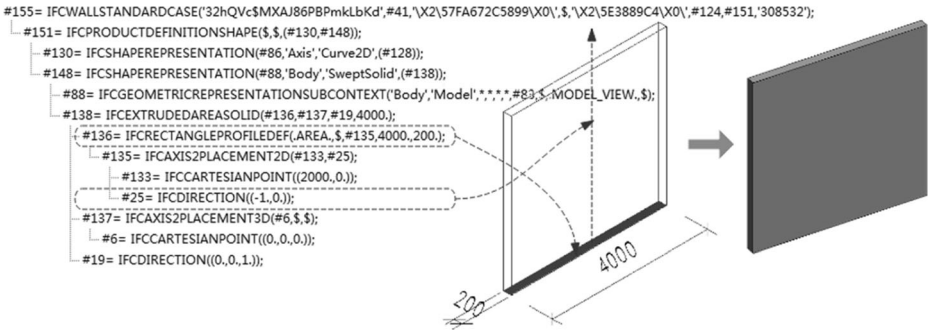


Fig. 2 Geometry representation in IFC file: an example of IFCWallStandardCase

commercial software tools have been widely used on market for several years, e.g., Autodesk Revit, Bentley Architecture, Graphisoft ArchiCAD, and Nemetschek ALLPLAN. With these tools, operations such as BIM creation, editing, and visualization can be effectively performed. However, these tools are all built for local BIM. Moreover, they occupy very large computation resource during their operations, including CPU, memory and GPU.

Currently, there are also some simple commercial IFC viewers addressing the IFC viewing issue [6, 10] and WebGL-based tools [4, 5]. The IFC viewers include the stand-alone tools FZK Viewer [10], BIM Vision [6], to name a few. The tools in this catagory are also built for local IFC files. BIM Surfer [4], BIMviews [5], both of which are built for BIMServer project [3], are two popular WebGL-based BIM visualization tools. However, they both neglect the network load, and thus are inefficient in the Internet environment. Currently, Liu et al. also proposed a Web3D tool for BIM. However, their tool is built on a Flash3D engine (named Away3D), and can only be applied in specific environment. That is, their solution is not cross-platform.

4 Cross-platform visualization of online open BIM

This section presents the details of our cross-platform online visualization system of open BIM, WebBIM.

4.1 Overall framework of WebBIM

The WebBIM consists of three key models: Triangle Convertor Model (TriCon), Instance Light-Weighting Model (InsMod) and WebGL Render Model (WebGL-RM). Figure 3 gives out the overall framework of the WebBIM.

TriCon: TriCon converts the raw geometry in IFC into the triangles, which can be rendered directly in WebGL.

InsMod: InsMod light-weights the triangles by separating the facility components from their instances. All the instances generated from the same facility component shares the same geometry description. Moreover, InsMod utilizes the compressed algorithm to reduce the data amount transferring from the servers in the cloud to the users' devices.

WebGL-RM: WebGL-RM renders the light-weighted triangles directly in a web browser.

WebBIM firstly converts the original raw geometry in IFC into triangles in the TriCon, then light-weights the triangles by reusing the geometry in the InsMod. Finally, WebBIM directly renders the BIM using the light-weighted geometry data in WebGL-RM.

4.2 Triangle convertor model (TriCon)

As described in Section 2.3, the IFC specification defines seven common geometry models. The IFC version 2 × 3 mainly uses the CSG, SweptSolid and Brep. CSG limits to the type of cutting. That is, the IFC specification firstly sweeps the physical structure, and then Boolean crop operation between plane and the physical structure is executed. The Brep is usually used to express the geometric shape. Brep takes the vertex and the curve as the basic geometry shapes, because the vertices and the curves can be converted into the triangular mesh surfaces effortlessly. Thus, the first step in TriCon is to convert the geometry data expressed in CSG and SweptSolid into Brep. The conversion process mainly includes three processes: vertex calculation, boundary reorganization and topology reconstruction.

The second step of TriCon is the model boundary segmentation. This study uses the Delaunay triangulation algorithm [17] to delineate the triangulation of the boundary of all the Brep models. In detail, the mesh partitioning process uses the Bowyer-Watson algorithm [18], which utilizes the property of Delaunay cavity. At the beginning, an initial triangle is set up. Only one new vertex is added each time. When the new vertex is added to the Delaunay grid, some triangles no longer meet the property of Delaunay cavity. In this scenario, their outer ball contains the new vertices, and these triangles are removed, which produces new Delaunay cavities. The newly added vertex connects to all the vertices that make up the Delaunay cavity to produce a new edge. It is easy to prove that the triangles made by these newly added edges conform to the property of Delaunay cavity. In this manner, a new vertex is added to the original triangles. The triangular meshes are formulated by literately adding all the vertices of the boundary into the original triangle. This process is easy to implement and free of the dimensions of the space.

After the above steps, the geometric data defined by the IFC standard in Fig. 3 is converted into the triangular mesh data shown in Fig. 4. The triangular mesh data generated by the object is stored as a JSON object, which has been widely used in network applications currently [1]. This storage method is rewarding to both the network transmission and database storage.

The format of the JSON in WebBIM is described Table 2. The parameter “materials” describes the material information of the object instance, while the parameter “uvs” records all the uv data. “vertices” is the vertices in the mesh of the object, and “normals” is the normals of each vertex. The parameter “colors” stores all the colors used in the object instance. The

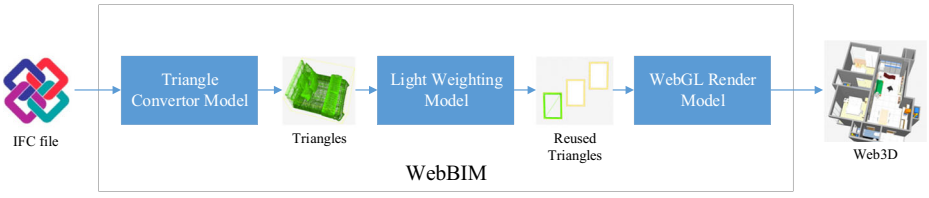


Fig. 3 Overall framework of WebBIM

```
{
 "positions": [
4000,100,0,4000,100,4000,0,100,0,0,100,4000,0,100,0,0,100,40000,-100,0,0,-100,4000,0,-100,0,0,-100,
        4000,4000,-100,0,4000,-100,4000,4000,-100,0,4000,-100,4000,4000,100,0,4000,100,4000,4000,100,0,0,
        100,0,0,-100,0,4000,-100,0,4000,100,4000,0,100,4000,0,-100,4000,4000,-100,4000]
   "indices": [ 1,0,3,3,0,2,5,4,6,7,5,6,9,8,10,11,9,10,13,12,14,
        15,13,14,17,19,18,17,16,19,22,23,21,23,20,21],
  "normals": [ 0,1,0,0,1,0,0,1,0,0,1,0,-1,0,0,-1,0,0,-1,0,0,-1,0,0,
        0,-1,0,0,-1,0,0,-1,0,0,-1,0,1,0,0,1,0,0,1,0,0,1,0,0,
        0,0,-1,0,0,-1, 0,0,-1,0,0,-1,0,0,1,0,0,1,0,0,1,0,0,1],
    "matrix": [ 1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1],
    "colors": [0.5019608,0.5019608,0.5019608,1],
 "materials": [ 0,0,0,0,0,0,0,0,0,0,0,0 ],
      "type": "IfcWallStandardCase"
      "guid": "32hQVc$MXAJ86PBPmkLbKd",
}
```

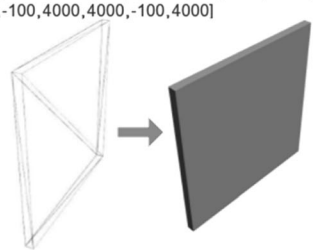



Fig. 4 Triangle representation of the wall object presented in Fig. 2

parameter "indices" defines all the triangle faces by referring the index of vertices defined in the parameter "vertices". The "matrix" parameter is a 4 × 4 matrix, which presents the rotation and translation of the indices. Figure 4 presents an example of the JSON object of the object instance in Fig. 2.

4.3 Light-weighting model (InsMod)

WebBIM aims to provide powerful tool for online visualization of BIM. We collected 20 real-world projects' BIM data, and found that more than 90% BIM model is larger than 100 Mbytes. This huge data proposed challenges for the online visualization for BIM, since pretty a long time is required for transferring 100 Mbytes data from the servers in the cloud to the users' visualization devices in the Internet. Thus, the first and primary task of WebBIM is to light-weight and compress the BIM data.

InsMod addresses this challenge by reusing the geometry data of the object instance generated from the same facility components and compressing the geometry data using gzip [8].

We firstly present an example of sharing the geometry among the object instances generated from the same facility components. Figure 5 is the BIM model of Research Building of the Beijing University of Civil Engineering and Architecture. Table 3 lists the details of some instances. It is noticing that some facility components have more than 100 instances. In this scenario, the geometry can be reduced to be less than 1% by sharing the geometry data. For example, $C_{\text{Window 1000x1800mm}}$ has 104 instances. Suppose the geometry data size of $C_{\text{Window 1000x1800mm}}$ is s Kbytes. The total amount data size of the instances of $C_{\text{Window 1000x1800mm}}$ costs 104 s Kbytes, which imposes the network load for WebBIM.

Table 2 Description of object instance's geometry in JSON

```
{
"materials": [{}],
"vertices": [],
"normals": [],
"colors": [],
"uvs": [],
"matrix": [],
"indices": []
}
```

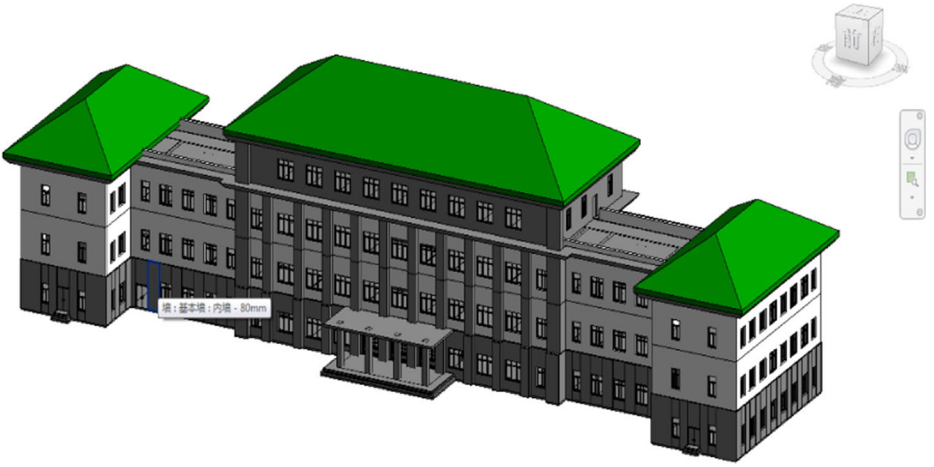


Fig. 5 BIM of Research Building in BUCEA

Obviously, all the instances from the same facility component have the same geometry shape, except the location and the rotation. For any vertex (x, y, z) in a three-dimensional space, it can be translated or rotated to some other point by a 4×4 matrix, termed as model matrix. For example, to translate (x, y, z) to point $(x + T_x, y + T_y, z + T_z)$, the model matrix can be

$$\begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x + T_x \\ y + T_y \\ z + T_z \\ 1 \end{pmatrix}.$$

To rotate the vertex by x-axis with angle θ, the model matrix is

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x \\ \cos\theta \cdot y - \sin\theta \cdot z \\ \sin\theta \cdot y + \cos\theta \cdot z \\ 1 \end{pmatrix}.$$

Obviously, C(j) i and C(k) i have the same materials, vertices, normal, colors, uvs and indices as C_i. The only difference among the instances of C_i is the 4×4 model matrix. Thus, it is possible to share the geometry data among the instances of the same facility component.

Table 3 Number of instances of some facility components

Facility component	# of instances
Window 1000 × 1800 mm	104
Window 1500 × 1800 mm	98
Window 1750 × 1800 mm	24
Window 2000 × 1800 mm	33
Column 475 × 610 mm	16
Door 600 × 1800 mm	59

Taken $C_{\text{Window 1000x1800mm}}$ as an example, the data amount of all the instances of $C_{\text{Window 1000x1800mm}}$ is reduced to be $s + (104 - 1) \times 16 \times 8 / 1024$ Kbytes (suppose each number in the model matrix occupies 8 bytes) = $(s + 12.875)$ Kbytes. This process greatly reduce the data amount.

We take window to show the main process of InsMod. The parameters and shapes of a window object is defined by IFCWindowStyle in IFC file. IFCWindow defines a window object instance including its location in world coordinates. The location parameters of the window instance can be obtained from the IFCWindowLiningProperties referred by the IFCWindow. Figure 6 presents an example of instancing IFCWindowStyle to IFCWindow. In other words, all the window instances of the same facility component share the same IFCWindowStyle. With this analysis, we developed our geometry-sharing algorithm of InsMod, the whole process is summarized in Algorithm 1.

Algorithm 1 : Geometry Sharing of InsMod

Input : $\mathcal{B}$

C = [], R = dict()

For each $C_x^{\mathcal{V}}$ in $\mathcal{B}$:

  If C_x not in C:

    Append C_x to C

  End If

  ind = key of C_x in C

  $R[C_x^{\mathcal{V}}]$ = ind

End For

Save C and R to database

To support the geometry sharing process, the data storage of an object instance is, thus, separated into two tables: instances and components. The “instances” table records the basic

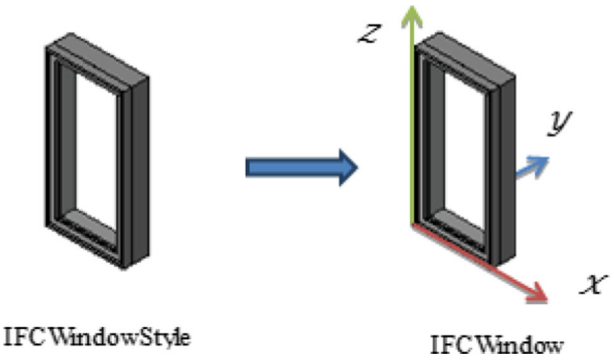


Fig. 6 Instancing IFCWindowStyle to IFCWindow

information, the model matrix and its corresponding facility component index for each object instance, while the "components" table records the triangles of all the facility components.

Finally, InsMod also compressed the geometry data before transferring them from the servers to the clients. Apparently, a bundle of data compress algorithms are candidates. InsMod uses gzip as the data compress algorithm, since gzip is simple and directly supported by the popular web browsers.

4.4 WebGL-RM

WebGL-RM renders the BIM using WebGL. WebGL firstly decompress the received geometry data. Then, the WebGL engine directly renders the triangles formatted in JSON.

We also developed tools for common operations, including rotation, translation, zoom, isolation and so on.

5 Empirical studies

This section presents the efficiency of WebBIM and verifies that WebBIM is capable of loading large BIM file and compatible with different devices.

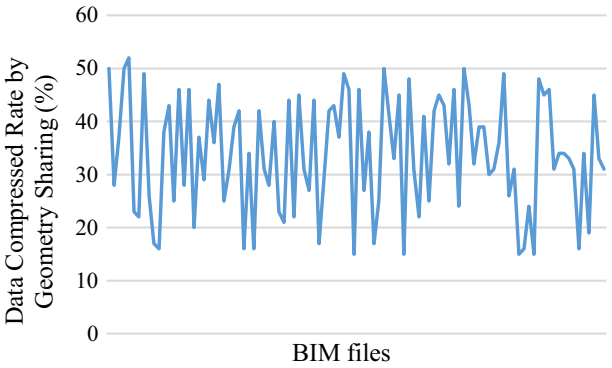


Fig. 7 Data compressed rate by geometry sharing

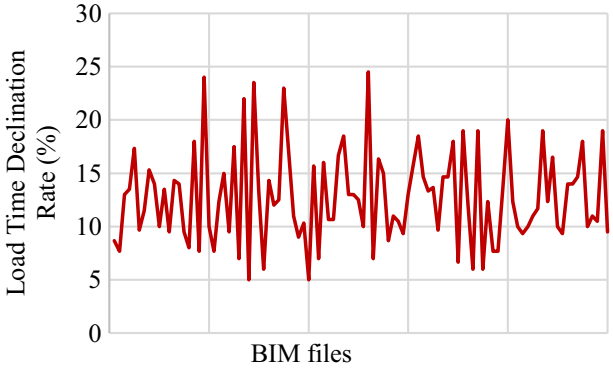


Fig. 8 Data load declination rate by InsMod

5.1 Data light-weighting

We chose 100 BIM files to check how much a BIM file can be compressed through our geometry sharing. The 100 BIM files were obtained from the BluePrint Product of BIM Winner Co. Ltd. [7] and sourced from different disciplines and different projects. Evidently, different BIM file gains different compressed rate from our algorithms. As revealed in Fig. 7, almost all of the BIM files gains less than 50% compressed rate. In some BIM files, the compression rate can be less than 20%. Subsequently, WebBIM is powerful in light-weighting the BIM data.

We also compared the loading time of the 100 BIM files with/without InsMod. Since gzip is utilized in InsMod to mitigate the network load in WebBIM. Generally, gzip compresses the data in 2-4 times. Thus, the load times with InsMod is much more rewarding than those without InsMod. Figure 8 presents the load time declination with InsMod in the 100 BIM files.

5.2 Visualization of large BIM files

To evaluate the performance of WebBIM in large BIM files, we load a BIM file with 50,576 object instance. The BIM file has 4,212,219 IFC objects, 37,142,712 triangle faces and 111,428,136 vertices. The FPS of the WebBIM keeps steadily at around 25

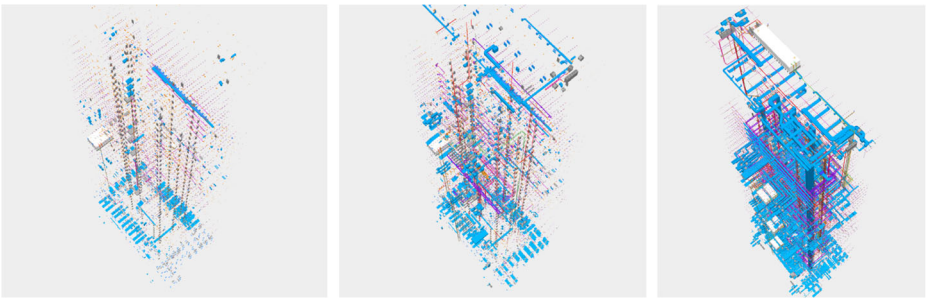
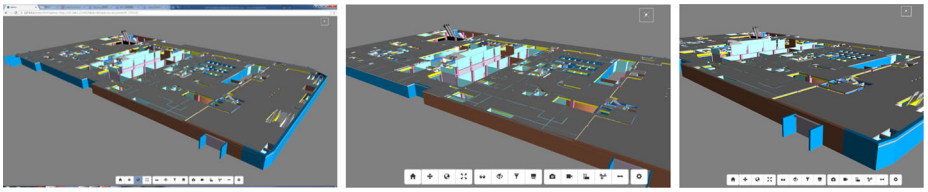


Fig. 9 Loading processes of WebBIM in large BIM file



(a) chrome in personal computer (b) web browser in iphone6 (c) web browser in Android pad

Fig. 10 WebBIM in different devices **a** chrome in personal computer **b** web browser in iphone6 **c** web browser in Android pad

when rotating and translating the model. This suggests that WebBIM is capable of the visualization of large BIM files. Figure 9 presents the loading process of the BIM files.

5.3 Cross-platform visualization of BIM files

To test that WebBIM has the ability to render BIM files on different devices, we loaded BIM files in web browsers from different devices, including personal computer, iPhone and Android pad. Figure 10 shows the visualization of BIM files in these devices. We found that WebBIM has an acceptable performance in all these devices.

6 Conclusions

Online BIM visualization system is the first and primary task for building web applications of BIM. We firstly suggest that an online BIM visualization system has to meet three basic requirements: light-weighted, cross-platform and open. Based on these basic rules, we developed a novel online BIM visualization system based on IFC and WebGL, termed as WebBIM.

WebBIM firstly converts the raw IFC gemotry data into triangles, which can be directly rendered in WebGL. Then, WebBIM light-weights the BIM geometry data by sharing the geometry data among the object instances generated from the same facility component. Moreover, WebBIM compresses the BIM data using gzip before transferring data form the servers in the cloud to the users' devices. Finally, WebBIM directly renders the decompressed BIM data.

At the end, we verified that WebBIM is efficient, capable of visualization of large BIM files and compatible with mainstream devices through extensive real projects' BIM data.

Different from current studies on visualization of BIM, WebBIM is light-weighted, cross-platform and open. Undoubtedly, WebBIM provides a powerful fundamental tool for the BIM applications. We can anticipate that WebBIM can be rewarding to more applications during the life-cycle of a building, including 4D simulation [12], safety analysis [20], indoor route planning [19], quality assurance [13], life-cycle management [11], to name a few.

Acknowledgements This work was supported by the Beijing Natural Science Foundation under grant no. 4174087, the Scientific Research Project of Beijing Educational Committee under grand no. SQKM201710016002, and the Natural Science Foundation of China under grant nos. 71601013 and 71531012.

References

1. Afsari K, Eastman CM, Castro-Lacouture D (2017) JavaScript Object Notation (JSON) data serialization for IFC schema in web-based BIM data exchange. Autom Constr 77:24–51
2. Azhar S (2011) Building information modeling (bim): trends, benefits, risks, and challenges for the aec industry. Leadersh Manag Eng 11(3):241–252
3. BIM Server, http://www.bimserver.org, [Online; accessed 20-Sep-2017]. 2017
4. BIM Surfer, http://bimsurfer.org, [Online; accessed 20-Sep-2017]. 2017
5. BIM views, http://www.bimview.fr, [Online; accessed 20-Sep-2017]. 2017
6. BIM Vision, http://www.bimvision.eu/home, [Online; accessed 20-Sep-2017]. 2017
7. BluePrint, http://bp.rickricks.com, [Online; accessed 20-Sep-2017]. 2017
8. Deutsch LP. GZIP file format specification version 4.3, 1996
9. Eastman CM, Eastman C, Teicholz P, & Sacks R. BIM handbook: a guide to building information modeling for owners, managers, designers, engineers and contractors. John Wiley & Sons, 2011
10. FZK Viewer, http://www.iai.fzk.de/www-extern/index.php, [Online; accessed 20-Sep-2017]. 2017
11. Hammad A, Zhang C, Hu Y, Mozaffari E (2006) Mobile model-based bridge lifecycle management system. Computer-Aided Civil and Infrastructure Engineering 21(7):530–547
12. Hu Z, Zhang J, Deng Z (2008) Construction process simulation and safety analysis based on building information model and 4D technology. Tsinghua Sci Technol 13:266–272
13. Kim M, Wang Q, Park J, Cheng J, Sohn H, Chang C (2016) Automated dimensional quality assurance of full-scale precast concrete elements using laser scanning and BIM. Autom Constr 72:102–114
14. Liebich T, Adachi Y, Forester J, Hyvarinen J, Karstila K, & Wix J (2006) Industry foundation classes IFC2× 3. International Alliance for Interoperability 467-476
15. Liu X, Xie N, Tang K, Jia J (2016) Lightweighting for web3d visualization of large-scale bim scenes in real-time. Graph Model 88:40–56
16. Parisi T. WebGL: up and running. O'Reilly Media, Inc, 2012
17. Rajan VT (1994) Optimality of the Delaunay triangulation in $\mathbb{R}d$. Discrete Comput Geom 12(2):189–202
18. Rebay S (1993) Efficient unstructured mesh generation by means of Delaunay triangulation and Bowyer-Watson algorithm. J Comput Phys 106(1):125–138
19. Teo T, Cho K (2016) BIM-oriented indoor network model for indoor and outdoor combined route planning. Adv Eng Inform 30(3):268–282
20. Zhang J, Hu Z (2011) BIM-and 4D-based integrated solution of analysis and management for conflicts and structural safety problems during construction: 1. Principles and methodologies. Autom Constr 20(2):155–166

Xiaoping Zhou received the B.E. and M.E. from Beijing Information Science and Technology University in 2006 and 2009, respectively. He is an Associate Professor at the Beijing University of Civil Engineering and Architecture. His research interests include building information model/modelling, big data mining and artificial intelligence. E-mail: lukefchou@gmail.com.

Jia Wang received the Ph.D. degree from the Beijing Jiaotong University, China, in 2014. She is a Professor at the Beijing University of Civil Engineering and Architecture. Her research interests include building information model/modelling, fire information system. E-mail: wangjia@bucea.edu.cn.

Ming Guo is currently an Associate Professor in the Institute of mapping and urban spatial information, Beijing University of Civil Engineering and Architecture. He received his Ph.D. in Photogrammetry and Remote Sensing from Wuhan University in 2011. His current research interests are in the areas of LiDAR measurement technology, mobile mapping system, 3D spatial index, 3D model reconstruction and spatial analysis of point cloud. E-mail: gmrichard@whu.edu.cn.

Zhe Gao received the B.E. in electronic science and technology from Century College of Beijing University of Posts and Telecommunications in 2015. He is currently a postgraduate student in the School of Electrical and Information Engineering at Beijing University of Civil Engineering and Architecture. His research interests include Building Information Modeling and data processing. E-mail: gaozhe@stu.bucea.edu.cn.