Research on BIM model lightweighting methods and IoT technology application in the context of WebGL

Yuhe Yuan¹, Jing Yang¹,²*, Ruping Shao¹,², Jiahuan Mao¹ and Chengwei Zhang¹

¹College of Electrical Engineering and Control Science, Nanjing Tech University, 211816 Nanjing, China
²Engineering Research Center of BIM, Nanjing Tech University, 211816 Nanjing, China
muyi_qing@njtech.edu.cn*

Abstract. In view of the fact that most of the existing building information modeling (BIM) platforms are based on the Client/Server (C/S) mode of client-side applications, which brings about problems such as weak data interoperability, poor model semantic integrity and bad user-friendliness, this paper proposed an IFC-JSON and IFC-glTF model format conversion process based on WebGL and an improved Draco algorithm that can further lighten glTF files. The experimental results showed that this option can effectively reduce data redundancy, then combined IoT (Internet of Things) as well as sensor technology in project practice to collect and transmit building-related O&M data, and finally visualised the equipment status of the building on the Web based on WebGL technology, providing a new application way for the combination of BIM technology and IOT technology in the browser/server mode.

1. Introduction
Building information modeling (BIM) technology is widely used for data sharing and information integration management in all phases of projects in the architecture, engineering & construction and facilities management (AEC/FM) industry, with its characteristics of serving the whole life cycle of buildings. The core value of BIM technology is to use the BIM server as the storage carrier for information, so as to interact with multiple clients of the project. With the volume of engineering projects increasing year by year, the difficulty of managing large volume 3D models and data gradually increases, so it is especially important to ensure the complete information display of BIM models and meet the user's convenient access under different operating devices.

Classified according to the different forms of clients, they can be divided into client/server (C/S) architecture and browser/server (B/S) architecture. Most of the existing BIM platforms are based on the client-based C/S architecture, which brings about problems such as weak data interoperability, poor model semantic integrity and bad user-friendliness, limiting the integration of BIM with other technologies. In contrast, the application of BIM technology based on B/S architecture effectively reduces the hardware and software requirements of the web front-end and improves the compatibility with mobile terminals, among which the efficient rendering of BIM models based on WebGL technology is a current research hotspot.

In view of this, this paper proposed a file format conversion process and a 3D model lightening method based on Draco algorithm for large-volume building models for web-based display. The glTF file was selected as the final output format to achieve efficient compression of the model to guarantee
the display effect, while ensuring the retention of building attribute data. In order to broaden the application scenario of the operation and maintenance phase, the IoT (Internet of Things) and sensor technologies were combined to simulate the collection and transmission of building-related operation and maintenance data, combined the dynamic building operation and maintenance data with the lightened BIM model, and finally realised the visualisation of the field equipment status of the building on the web side based on WebGL technology.

2. Research methodology
With WebGL technology as the background, this paper proposed an improved solution for the lightweight presentation of BIM models on the web side in response to the problems of semantic data loss and geometric data redundancy under the current technical approach, and practiced the real-time interaction function of building operation and maintenance data in conjunction with IoT technology.

The solution addressed the problems of lack of semantic integrity and weak information interoperability in the current research, and started with the common IFC file in the field of BIM technology[6], carried out decoupling of semantic and geometric information, stored them as JSON and glTF files respectively, and realised the dual relationship mapping from IFC geometric space and JSON semantic attributes to glTF. To address the problem of stalling in displaying large volume models, a Draco algorithm with improved parameters was proposed to recode the glTF file of the geometry part so as to achieve further lightweighting; to address the problem of displaying static data and dynamic data in the application scenario of operation and maintenance of BIM technology, IoT technology was used to monitor sensor information to achieve 3D model display and O&M function development on the web side. The technical route of the research is shown in Figure 1.

![Figure 1. Research technology route.](image)

3. BIM model processing

3.1. IFC-JSON and IFC-glTF file format conversion
IFC files are a common data file format used by the AEC/FM industry for data interaction and sharing, but they are not compatible with 3D formats used for web display, so they need to be converted to a document supported by WebGL for BIM model visualisation. The proposed approach took the common IFC file as input and the JSON and glTF files supported by the Three.js platform as output, splitting the IFC file into semantic and geometric files for processing, which will ultimately retain the complete component information and enable preliminary file classification and lightweighting. To ensure the
completeness of the BIM application, two common GIS data: CityGML files, GeoJSON files were also given for processing, the specific conversion process and the open source tools used are shown in Figure 2.

![Conversion process of BIM model file](image)

**Figure 2.** Conversion process of BIM model file (contains geographical information).

As shown in Figure 2, for semantic information extraction, the content under the IFC semantic parameters can be extracted in a targeted manner using Xbim[7] and stored in the newly generated JSON file. For geometric information, the original IFC file as input was decomposed into several small IFC files by component type using BIM Server APIS[8] before format conversion, in order to solve the problem of not being able to identify the model components individually after model conversion in the previous study[9]. The next step was to use Ifc Convert[10] to extract and store the geometry information in the IFC file as an OBJ file and a separate MTL file to store the material information. The OBJ format was chosen as the first step in the conversion because there are many tools built around it for geometry format conversion; finally, the open source tool obj2glTF[11] was used to obtain the glTF format file, which is a graphics language interchange format designed for real-time rendering applications, combining scene description information such as node hierarchy and model geometry. The glTF file stores data information such as node hierarchy and model geometry as well as animation images in chunks, while the model is usually loaded with a glTF file in text format only, and other block files can be introduced externally and passed directly to the graphics API without additional parsing and conversion. As a result, the glTF format allows for the maximum compression of the original IFC file while ensuring the semantic integrity of the model and data interaction, enabling the web visualisation of lightweight BIM models.

3.2. *Improved Draco algorithm for lightweighting*

Previous model file lightweighting has mostly stopped at the change in data structure due to the conversion of file types. The improved Draco lightweighting algorithm was used, specifically by adding the control file compression ratio parameter draco.compressionLevel to the build type, to achieve further lightweighting of the final glTF file generated by the above format conversion process.

The Draco algorithm is a technical tool for compressing and decompressing geometric mesh data and point cloud data, encoding the connectivity and geometric information of the mesh separately and storing the code, supporting the compression of points, connectivity information, texture coordinates, colour information, normals and other geometry-related attributes. The design features are well suited
for processing glTF geometry files with point and surface structures, and can improve the storage and transmission of 3D images, allowing for a significant reduction in the number of applications using 3D graphics while maintaining visual fidelity.

The core idea of the algorithm is the classical 3D mesh compression algorithm Edgebreaker, and in particular the Corner Table structure of the algorithm\[12\], which essentially encodes and compresses the mesh graph by traversing it in a breadth-first principle, compressing it to reduce duplicate vertex information and replace it with lighter index information. By describing the partition path of the mesh through operators, the 3D scene can then be created from the transformed encoding, thus achieving compression with much less visual bias using the Draco compression method as opposed to direct surface reduction.

As an example of re-encoding glTF files, the encoding process is shown in Figure 3.

![Diagram of Draco encoding process]

- **Input**: glTF geometry file
- **Read mesh from file**
- **Choose decoder**: glTFDecoder based on file type
- **Decode from file**
- **Create**: MeshEncoder
  - **MeshEncoder**: Encode
  - **Encode header**: Encode encoder data
  - **Encode connectivity**: Compression of connection information, topology of points forming faces, using Edgebreaker
- **Encode point attributes**: Compression of vertex property information, position, normals, texture coordinates, etc.
- **Encode geometry to buffer**: Encoding of geometry is completed with the encoder data.
- **Output**: recoded glTF geometry file

For the application of the Draco algorithm, the open source tool glTF Pipeline is available as support\[13\]. In order to further strengthen the compression capability of the Draco algorithm, this study set the parameter draco.compressionLevel that controls the compression ratio of the file on the basis of recoding, and set separate compression parameters for different types of component glTF files for different application scenarios, and compressed the grid data index on the basis of retaining the original grid order, the specific steps of the application are as follows.

**Step1**: Serialisation of glTF mesh data. Traversing component information, generate binary indexes, insert relevant nodes and generate scene-node-mesh objects.

**Step2**: Mesh model data compression. Read the serialised glTF file as input to the draco_encoder and determines the component type based on the component id, adds a draco.compressionLevel parameter ranging from 0 to 10 and outputs it to the algorithm’s custom .drc format file.

**Step3**: Mesh model data decompression. Read the .drc file as draco_decoder input and re-exports it as a glTF file after decompression.

**Step4**: glTF file reading display. Use GLTFLoader and DracoLoader in three.js as data interfaces for compressed model output display.

### 4. Experimental analysis and web-side development

#### 4.1. Model file conversion comparison

In order to verify the effectiveness of the geometric part IFC-glTF file format conversion strategy and the lightness of the Draco algorithm, this paper used Revit software as the drawing tool for BIM models, and selected six groups of building model files with similar building structures but different scales to compare and analysed the compression rate and web-side rendering effect of the models.

Applying the above proposed process to convert the file format of the geometric profiles and recording the file size data for IFC-OBJ, IFC-glTF and IFC-DracoITF, Figure 4 is used to show the above files used for geometric presentation, which clearly showed the change in file size before and after compression. The final decoupled JSON semantic file and lightweight DracoglTF geometry files sizes obtained under this approach are shown in Table 1.
Figure 4. Geometry file size after conversion.

Table 1. Compression results for model files.

| Serial number | Original model size (MB) | Semantic file size (JSON) (MB) | Geometric file size (DracoglTF) (MB) | Geometric compression ratio |
|---------------|--------------------------|-------------------------------|-------------------------------------|---------------------------|
| 1             | 16.5                     | 0.4                           | 2.9                                 | 82.42%                    |
| 2             | 29.6                     | 4.3                           | 4.9                                 | 83.45%                    |
| 3             | 82.1                     | 10.1                          | 15.4                                | 81.24%                    |
| 4             | 203.5                    | 15.8                          | 20.7                                | 89.83%                    |
| 5             | 507.2                    | 63.1                          | 75.3                                | 85.15%                    |
| 6             | 1551.7                   | 72.5                          | 172.5                               | 88.89%                    |

Analysis of the data in Table 1 showed that decoupling the IFC file into a JSON file for the semantic part and a glTF file for the geometric part, and invoking the Draco algorithm to compress the glTF file, had a significant compression effect, with compression rates above 80% and up to 89.83%. Model 2, a BIM model file of a school building with an original size of 29.6MB, was selected and the converted intermediate file was rendered under the common 3D engine Three.js. The rendering results and frame rates for the three formats are shown in Table 2. The analysis showed that after the conversion of DracoglTF format, only the geometric part of the glTF file was extracted for rendering, which can effectively reduce the file size of the part to be rendered, and the frame rate was greatly improved.

Table 2. Rendering and file information of BIM model.

| File format | OBJ file | glTF file | DracoglTF file |
|-------------|----------|-----------|----------------|
| Web side display | ![OBJ file](image1.png) | ![glTF file](image2.png) | ![DracoglTF file](image3.png) |
| File size/MB | 236.5 | 74.3 | 4.9 |
| Frame rate/ fps | 13 | 35 | 58 |

4.2. Web-side functional development
The Three.js framework allows for the development of 3D interactive functionality, allowing users to query component property information through the browser. Taking a structural column of this building as an example, the properties of the Revit scene and the web page are shown in Figure 5 and Figure 6 respectively.
This study introduced IoT technology into the lightweighted BIM model, aiming at visualisation of the model and real-time display of equipment status data, which was exported via open database connectivity in the modeling software Revit.

Dynamic data transmission in the IoT is realised by an IoT program consisting of a node driver and a server-side program. The Zigbee technology is suitable for scenarios in buildings with low network bandwidth and unstable networks due to its low cost, low power consumption and no device limitations, as well as the MQTT protocol’s ability to provide an orderly and reliable, bi-directional connection with guaranteed network connectivity that is open, simple and easy to implement. Therefore, this study used Zigbee protocol and MQTT protocol as the core to achieve protocol interfacing and designed a high degree of freedom IoT device control module. After the device was started, the raw data collected by the sensor was switched on and off by the controller, and the raw data was verified, then the verified data was transmitted to the message middleware with the help of the Zigbee network, which sifted and analysed the data and sent it to the MQTT server, completing the interface between the Zigbee and MQTT protocols. Finally, the data was received, parsed and stored by the Web server to the MQTT server in a publish or subscribe communication. The specific transmission process of dynamic IoT data is shown in Figure 7. The sensor application and data presentation is shown in Figure 8.

---

**Figure 5.** Revit model scene.

**Figure 6.** Web-side model scene and property display.

---

**Figure 7.** IoT data transfer flow chart.

**Figure 8.** Sensor display (temperature sensor as an example).
5. Conclusion
This paper proposed a file conversion process for B/S model development and a model lightweighting method to address the problems of weak data interoperability, poor semantic integrity and bad user-friendliness under the existing C/S model and finally realized model information display and the development of interactive functions based on IoT technology. By decoupling the semantic part and geometric part of the IFC format files common to BIM platforms, the JSON file and glTF file supported by the 3D engine were obtained respectively, then by introducing the Draco algorithm and compression parameters, the 3D models were lightened while retaining the information of building components.

The experimental results showed that the method has a significant compression function for models, the compression ratio of the geometric part can be guaranteed to be above 80%, and can complete the interactive development of IoT technology based on Three.js platform while retaining the component information, which not only conforms to the current development process recommended by the latest 3D engine, but also well extends the application scenario, which has positive significance for the promotion of BIM technology.

References
[1] Ozturk G B Interoperability in building information modeling for AECO/FM industry 2020 Automation in Construction 113:103122.
[2] Xu Z, Zhang Y, Xu X 3D visualization for building information models based upon IFC and WebGL integration 2016 Multimedia Tools and Applications 75(24):17421-17441.
[3] Du Z, Wang H J, Tan Z and Huang H D Research on the lightweight application of water conservancy project BIM based on WebGL and IFC 2020 China Rural Water and Hydropower 199-203.
[4] Wang C. P Large architectural models real-time display system based on WebGL 2018 South China University of Technology.
[5] Zhang H Y Research on BIM visualization methods based on WebGL 2017 Xi'an University of Architecture and Technology.
[6] Laako M and Kiviniemi AO The IFC standard: A review of History, development, and standardization 2012 Information Technology.
[7] Černý M, Ward A, Benghi C and Lockley S xBIM project https://github.com/xBimTeam.
[8] Laat R D, Lindeque R and Nijs B D BIMserver https://github.com/opensourceBIM/BIMserver.
[9] Chen Y, Shooraj E, Rajabifard A and Sabri S From IFC to 3D Tiles: An Integrated Open-Source Solution for Visualising BIMs on Cesium 2018 ISPRS International Journal of Geo-Information 7(10):393.
[10] Moult D, Krijnen T, Kämäräinen A and Hahnebach B IfcOpenShell-IfcConvert https://github.com/IfcOpenShell/IfcOpenShell.
[11] Lilley S, Cozzi P and Amato M obj2glTF https://github.com/CesiumGS/obj2gltf.
[12] Rossignac J, Safonova A and Szymczak A 3D Compression Made Simple: Edgebreaker on a Corner-Table 2001 IEEE Shape Modeling International Conference. Genoa: Georgia Institute of Technology 278-283.
[13] Lilley S, Taglang R, Li K N and Cozzi P gltf-pipeline https://github.com/CesiumGS/gltf-pipeline.