International Journal of Geo-Information

MDPI

Article

From IFC to 3D Tiles: An Integrated Open-Source Solution for Visualising BIMs on Cesium

Yiqun Chen *, Erfan Shooraj, Abbas Rajabifard and Soheil Sabri

The Department Infrastructure Engineering, The University of Melbourne, Parkville, Victoria 3010, Australia; erfan.shooraj@unimelb.edu.au (E.S.); abbas.r@unimelb.edu.au (A.R.); soheil.sabri@unimelb.edu.au (S.S.)
* Correspondence: yiqun.c@unimelb.edu.au

Received: 31 August 2018; Accepted: 26 September 2018; Published: 28 September 2018

Abstract: The 3D Tiles specification, created by Cesium, is designed for streaming massive heterogeneous three-dimensional (3D) geospatial datasets online using WebGL technology. The program has prevailed in the WebGIS community due to its ability to visualise, interact, and style 3D objects for various scenarios, such as 3D cities, indoor environments, and point clouds. It offers a new opportunity to integrate Building Information Models (BIM) in the Industry Foundation Classes (IFC) data format with existing geospatial data in a 3D WebGIS platform with open-source implementation. As no open-source solution for converting IFC models into 3D Tiles for online visualization had yet been found, this paper explores feasible approaches and integrates a range of tools and libraries as an open-source solution for the community.

Keywords: BIM; IFC; 3D Tiles; Cesium; BIMServer; glTF; WebGL; 3D WebGIS

1. Introduction

The WebGIS community has been reshaped when Cesium, as a cross-platform virtual globe for dynamic spatial data visualisation, was founded by AGI (Analytical Graphics, Inc., Pennsylvania, United States) in 2011. Since then, Cesium has grown into a leading three-dimensional (3D) WebGIS platform adopting state-of-the-art WebGL technology to serve industries from geospatial and oil and gas to agriculture, real estate, entertainment, and sports [1–3]. It is an open-source JavaScript library for building the world-class 3D globes and maps to visualise static and time-dynamic contents, providing the best possible performance, precision, and visual quality [1]. In 2015, Cesium introduced an open specification named "3D Tiles" for streaming massive heterogeneous 3D geospatial datasets [4,5]. 3D Tiles is mainly used to stream 3D contents, including buildings, trees, point clouds, and vector data. It defines a spatial data structure and a set of tile formats designed for 3D and optimised for streaming and rendering. 3D Tiles models use glTF, the WebGL runtime asset format developed by Khronos [2,4,5]. Besides supporting city-scale 3D buildings, 3D Tiles is also capable of representing complex and detailed interior building environments such as Building Information Models (BIM) in the Industry Foundation Classes (IFC) data format. It significantly extends the Cesium application domains and provides a unique opportunity to seamlessly visualise BIM with traditional spatial datasets on one 3D WebGIS platform.

Recent years have witnessed immense demands for integrating BIM with existing Geographic Information Systems (GIS) [6–11]. BIM provide rich geometric and semantic information through the building lifecycle and use an object-oriented approach to describe the characteristics and behaviors of each building component, as well as its relationships with other components [12]. GIS are platforms for managing and presenting spatial information and cover much broader area such as geospatial visualisation, modelling, analysis and intelligence, and decision-making. The major strength of GIS, as opposed to BIM, is related to its effectiveness in modelling outdoor environments and large-scale

spatial features [6]. They model the spatial information in a complimentary way, and the BIM-GIS integration is widely considered [6,7,10,11] as a path forward for improving the power of using multidimensional data available at multiple granularity levels to support research fields such as energy, environment, navigation, planning, and emergency management. Tremendous research efforts have been devoted to this field in the past decade, from data level (including geometry-level and semantic-level) integration, to process level integration, and to application level integration [6,13,14].

Since visualisation is usually considered the fundamental requirement of BIM-GIS integration, the emphasis of this work was to investigate a wholesome solution for converting BIM, while retaining as much information as possible, into a format that can be rendered and interacted with on a WebGIS platform like Cesium. After reviewing existing solutions and identifying their drawbacks in Section 2, we propose an integrated conversion workflow and implement an IFC-3D Tiles composer based on open-source tools and libraries to facilitate the process. The feasibility and implementation details of the conversation workflow are articulated in Section 3; Section 4 demonstrates the composer performance with real BIM, and Section 5 discusses the known issues and future improvement directions.

2. Previous Work

The ability to incorporate and visualise BIM in the Cesium platform has drawn much attention among the Cesium community since 2016, and the Cesium development team released several tools to facilitate this requirement. However, no wholesome open-source solution exists for converting IFC BIM files into 3D Tiles, which can be imported and manipulated into Cesium. Existing BIM to Cesium converters are all commercial applications or services with drawbacks. Here are the two most applicable examples.

2.1. Rendering BIM in Cesium

The first BIM example in Cesium Sandcastle (a folder in the Cesium download package containing the latest features and usage examples) appeared in version 1.35 [15]. This power plant BIM was created by Bentley® (Pennsylvania, United States) using 3D Tiles. Each component of the BIM can be visualised and queried (e.g., height values) in Cesium as shown in Figure 1. The converted 3D Tiles resource (about 114 MB) was encapsulated in a URL and streamed into the Cesium engine. However, the conversion workflow was not documented online since it was a commercial service provided by Bentley®.

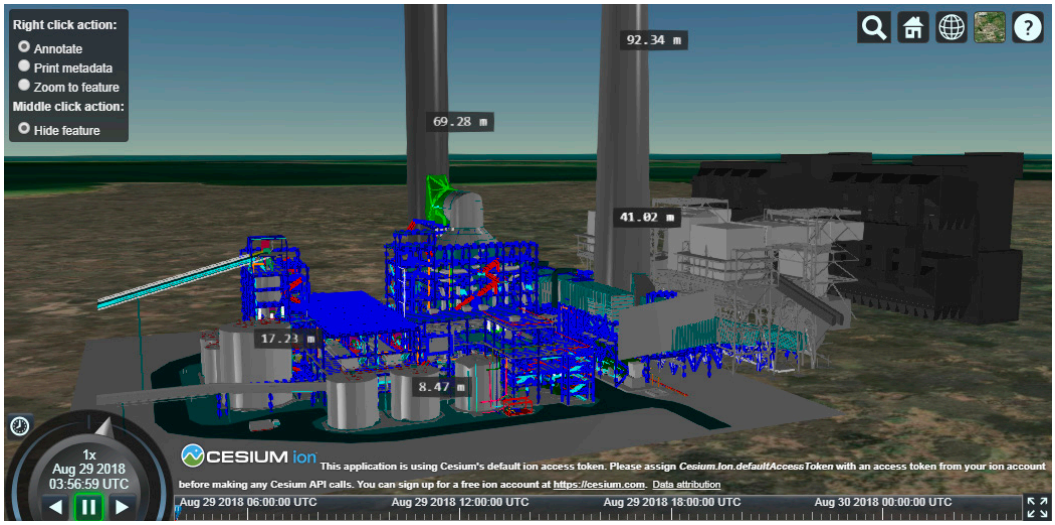

Figure 1. The first Building Information Model (BIM) example shown in Cesium Sandcastle (version 1.35). It is a power plant BIM converted into 3D Tiles by Bentley®. Each component of the BIM is identifiable in Cesium.

Most recently, Cesium offered BIM to the 3D Tiles conversion service for the community. The converted 3D Tiles are hosted on Cesium servers and are exposed as a URL from access. Still, the conversion process is a black box to users, as it does not offer any control options to customise the 3D Tiles generation. More importantly, this is an infeasible solution when the BIM data are sensitive to the restricted public access requirement (which is usually the case), meaning the conversion has to be performed in-house.

2.2. Mago3D (IFC)

Mago3D, created by Gaia3D (Seoul, South Korea), can seamlessly integrate AEC (Architecture, Engineering, and Construction) models and 3D GIS using Cesium [16]. It comes with an open-sourced tool called F4DConverter for converting popular 3D model formats (including .ifc, .3ds, .obj, and .dae) into F4D format, which was devised for Mago3D. This converter splits an original 3D model data into smaller-sized models divided by octrees, applies the Net Surface Mesh (NSM) method on each octree to reduce data size, and creates rougher data [16]. When rendering 3D scenes (Figure 2 shows an example), compared with the 3D Tiles format advocated by Cesium, the F4D format does not yield better performance or visual effect in Mago3D. The LOD (Level of Detail) controlled by NSM generates many surface mapping glitches and conflicts and weigh down the user experience.

Figure 2. IFC model converted into F4D format and rendered in Mago3D using Cesium.

3. Proposed Conversion Approach

This work aimed to explore a feasible and transparent solution for porting the IFC BIM into Cesium. The 3D Tiles format is most attractive for three reasons: (1) it is created and advocated by the Cesium core development team and the Cesium engine has been actively and continuously maintained and optimised to support this format; (2) it is built upon the prevalent glTF™ (GL Transmission Format), which is designed for the efficient transmission and loading of 3D scenes and models by applications. The glTF minimises both the size of 3D assets and the runtime processing needed to unpack and use those assets; (3) currently, it is a proposed OGC (Open Geospatial Consortium) Community Standard. Given these advantages, 3D Tiles was selected as the target format of the BIM conversion.

The proposed conversion approach from IFC BIM to 3D Tiles consists of four key steps: IFC decomposition, IFC to OBJ conversion, OBJ to glTF conversion, and glTF to b3dm (Batched 3D Model, which is a data format defined in 3D Tiles specification) conversion. The first three steps break the BIM into constituent components and convert each component, stored as a small IFC file, to OBJ format and then to glTF format. The last step merges the glTF files to a single constituent or multiple b3dm files containing a batch table hierarchy with a tileset.json file, which collectively defines

3D Tiles. Figure 3 shows the overall conversion workflow. Multiple open-source tools and libraries (highlighted in yellow in Figure 3) were used during the conversion (Appendix A).

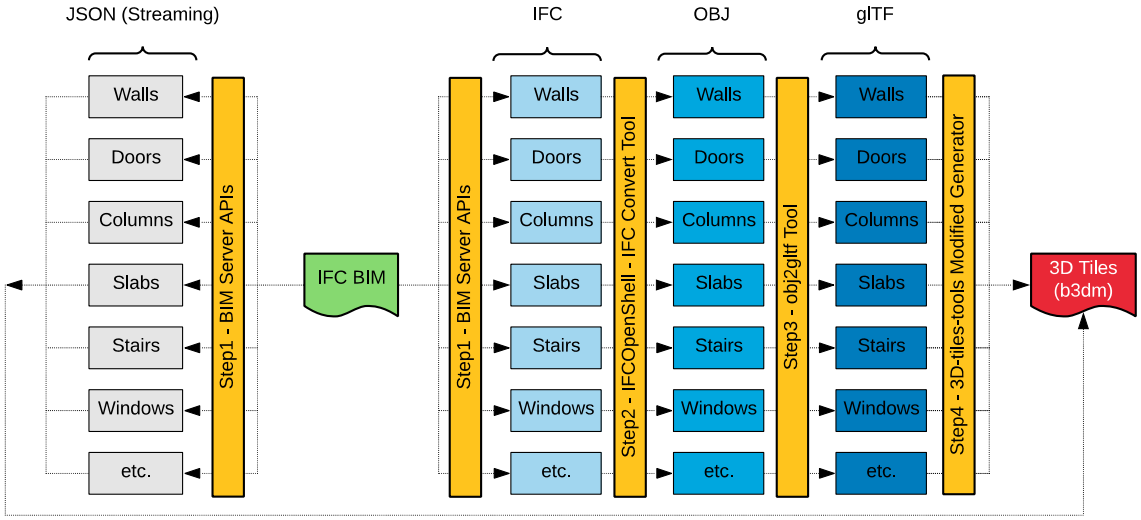

Figure 3. Proposed IFC to 3D Tiles conversion workflow.

3.1. Step 1: IFC Decomposition

The reason for decomposing an IFC file is to retain the attributes of each component after conversion. Many existing tools (such as Open 3D Model Viewer) can export the entire IFC file as a single OBJ model; however, after conversion, the model component cannot be identified separately, and all attributes for each component will be lost. To overcome this issue, the IFC file needs to be uploaded to a BIMServer instance that offers a series of web Application Programming Interfaces (APIs) to manipulate the BIM.

A composer was developed to handle the conversion process, and it is now open-sourced at Github (https://github.com/Erfan-Shooraj/ifc2b3dm). First, a JSON file containing all the IFC types and their respective object identifications (oids) are retrieved from the BIMServer API for the BIM. Then, the composer starts by reading the JSON file, which maintains the oid and type of each component. If the IFC type is one of the desired types with geometry, the composer downloads two files in Ifc2x3tc1 and JSON Streaming formats for that component from the BIMServer using specific queries based on the type. The IFC files are used to retain the geometry of the component, and the JSON Streaming files are used in the final stage to incorporate the attributes of each component in the final 3D Tiles.

Three types of queries are constructed and sent to the BIMServer APIs based on the IFC types; the query syntax is shown in Appendix B:

(1) General IfcProduct Query: This is used for the simplest IfcTypes to generate the following IfcTypes from the model: IfcWindow, IfcDoor, IfcColumn, IfcBuildingElementProxy, IfcBeam, IfcCovering, IfcRailing, IfcFlowTerminal, IfcFurnishingElement.

(2) Queries for IfcTypes with openings: This is used to generate separate IFC and JSON (Streaming) files for components with openings such as IfcSlab and IfcWall.

(3) Queries for objects with aggregate components: Some objects contain other components such as aggregates. For example, curtainwalls contain plates and members; stairs contain railings, stair flights and slabs; and ramps contain slabs and ramps. To convert and visualise these components properly, their respective aggregate relationships should be retained during the process, and this query is used to extract the required formats.

After this step, the IFC file stored in the BIMServer is decomposed into a collection of small IFC files and JSON files stored on the local hard drive, ready for the next conversion step.

3.2. Step 2: IFC to OBJ

In this step, the IfcConvert tool from IfcOpenShell library (IfcOpenShell 2018) is applied to convert each component IFC file into a separate OBJ file. It also generates a MTL file for the material of each component.

3.3. Step 3: OBJ to glTF

Using the obj2gltf tool, the composer analyses the OBJ files created in the previous step and converts them into glTF files with the flag "–materialsCommon" for compatibility with Cesium.

3.4. Step 4: glTF to b3dm

In the last step of conversion, the glTF and JSON (Streaming) files are grouped into b3dm files with a batch table hierarchy, using a modified version of "3D-tiles-tool Sample Generator" originally released by Cesium. A number of files within the sample generator library were modified to enable BIM and keep the 3D models intact. The source code is available on Github for more details.

The 3D Tiles generator begins by creating an instance for each glTF file provided and then includes these instances into the header of a b3dm file in the batch table hierarchy. Currently, our approach follows a flat hierarchy for every component with a single parent "building".

As the generator starts creating instances, it analyses each glTF file and finds its corresponding JSON file. The instance will have a class name equal to the file name without the extension, and its properties are extracted from the JSON file. The first JSON object in the file for every component contains information such as "Object Name", "Ifc Type", "GUID", and "BATID". The remaining properties are extracted by searching each file for IfcPropertySets with names "Dimensions" and "Constraints". Each component may have a various number of properties. Finally, the instances are turned into a hierarchy and written in the header of the b3dm file.

There are various means to encapsulate individual components into a b3dm file. For example, components on the same floor, in the same zone, or of the same IFC type can be wrapped up together. More sophisticatedly, quadtree or octree methods can be used to generate an optimised hierarchy of b3dm files based on the spatial information of each component. The method of constructing b3dm files may vary for different applications, and it largely determines how the 3D models will be rendered in Cesium and impacts the visualisation performance. Once all b3dm files are created, they are referenced in tileset JSON files (AGI 2018). A main tileset JSON file is the entry point for Cesium to load the entire 3D scene.

4. Performance

To gauge the performance of the conversion process, two IFC models with various complexities (a simple IFC file of 3.03 MB and a complex IFC file of 214 MB) were tested. The computer used for the conversion has the following specifications: Processor: Intel® Core™ i7-7700K CPU @4.20 GHz with 32.0 GB RAM. The overall processing time and storage space are listed in Table 1. The third step "OBJ to glTF" required the most processing time for both files.

Table 1. IFC to 3D Tiles conversion processing time and intermediate disk usage for a simple (3.03 MB) and a complex (214 MB) BIM.

	Simple IFC		**Complex IFC**	
	Time	**Space**	**Time**	**Space**
Decomposition (IFC and JSON)	3.3 s	11.26 MB	13.4 min	8.67 GB
Decomposition (IFC only)	0.86 s	2.6 MB	1.25 min	347 MB
IFC to OBJ	18 s	1.62 MB	3.5 min	741 MB
OBJ to glTF	36 s	914 KB	22.2 min	268 MB
glTF to b3dm	0.36 s	1.70 MB	9.5 min	568 MB

Since the complex IFC file contains more comprehensive and typical IFC types, the performance test results in the following tables are provided for this complex file. Table 2 shows the intermediate disk usage for each IFC type during the first IFC decomposition step. When extracting attributes from BIM into JSON format for each component, the file size increased dramatically.

Table 2. Intermediate disk usage for each IFC types during the IFC decomposition step. The original IFC file size was 214 MB and the JSON file size was 179 MB.

	File Number	IFC Size	JSON Size
Original IFC Model	**1**	**214 MB**	**179 MB**
Decomposed IFC Model	**5281**	**347 MB**	**8.32 GB**
- IfcWalls	853	10.3 MB	1.15 GB
- IfcWindows	181	2.35 MB	247 MB
- IfcDoors	398	119 MB	945 MB
- IfcColumns	1698	19.0 MB	2.16 GB
- IfcSlabs	145	1.33 MB	188 MB
- IfcStairs	58	12.0 MB	118 MB
- IfcBuildingElementProxies	372	92.7 MB	783 MB
- IfcBeams	896	12.2 MB	1.25 GB
- IfcCoverings	71	645 KB	91.6 MB
- IfcCurtainWalls	331	50.5 MB	0.99 GB
- IfcRailings	184	9.49 MB	264 MB
- IfcRamps	25	433 KB	33.6 MB
- IfcFlowTerminals	69	17.4 MB	153 MB

Table 3 shows the intermediate disk usage for each IFC type during the second IFC to OBJ step. The "Lost Objects" column indicates the number of IFC files that could not be successfully converted into OBJ format. This failure is mainly caused by geometry errors in the original IFC files which cannot be processed by the IfcConvert tool.

Table 3. Intermediate disk usage for each IFC type during the second IFC to OBJ step.

	OBJ File Number	Lost Objects	OBJ Size	MTL Size
Decomposed IFC Model	**5222**	**59**	**740 MB**	**1.00 MB**
- IfcWalls	841	12	9.22 MB	133 KB
- IfcWindows	181	0	2.77 MB	46.9 KB
- IfcDoors	388	10	311 MB	156 KB
- IfcColumns	1696	2	9.82 MB	285 KB
- IfcSlabs	125	20	1.10 MB	21.0 KB
- IfcStairs	58	0	28.5 MB	15.5 KB
- IfcBuildingElementProxies	357	15	222 MB	59.5 KB
- IfcBeams	896	0	3.83 MB	152 KB
- IfcCoverings	71	0	467 KB	13.6 KB
- IfcCurtainWalls	331	0	87.9 MB	89.8 KB
- IfcRailings	184	0	20.3 MB	34.9 KB
- IfcRamps	25	0	690 KB	3.21 KB
- IfcFlowTerminals	69	0	42.4 MB	15.3 KB

Table 4 summaries the disk usage and processing time for each IFC type for the last two steps. The total b3dm file size was 568 MB, which was 2.65 times of the size of the original IFC file. The entire conversion process required 48.6 min to complete.

Table 4. Intermediate disk usage and processing time for each IFC types during the OBJ to glTF and glTF to b3dm steps.

	File Number	glTF Size	b3dm Size	b3dm Time
Decomposed IFC Model	**5223**	**268 MB**	**568 MB**	**9.5 min**
- IfcWalls	841	5.51 MB	4.30 MB	1062 ms
- IfcWindows	181	1.66 MB	2.10 MB	602 ms
- IfcDoors	388	90.3 MB	350 MB	124,049 ms
- IfcColumns	1696	8.95 MB	6.91 MB	5636 ms
- IfcSlabs	125	792 KB	581 KB	122 ms
- IfcStairs	58	9.04 MB	10.1 MB	1585 ms
- IfcBuildingElementProxies	357	68.1 MB	111 MB	34067 ms
- IfcBeams	896	4.13 MB	2.59 MB	1050 ms
- IfcCoverings	71	401 KB	274 KB	89 ms
- IfcCurtainWalls	331	59.7 MB	46.3 MB	395,362 ms
- IfcRailings	184	7.13 MB	9.5 MB	2610 ms
- IfcRamps	25	291 KB	1.1 MB	445 ms
- IfcFlowTerminals	69	12.2 MB	21.2 MB	3057 ms

The converted 3D Tiles is published in a web server, and its URL can be directly consumed by the 3D Tiles sample code in Cesium Sandcastle. Each component is interactive in the scene and can be highlighted in yellow, as shown in Figure 4a with a mouseover. Clicking on each component shows all the properties that are stored in the header of the b3dm file in Step 4 (Figure 4b). The entire building model is streamed to the scene piece by piece. How the b3dm files are constructed and how the hierarchy of the tileset JSON is defined determine the rendering behavior and performance. For example, at the bottom-right corner in Figure 4a, there is a row of slabs (IfcSlabs) that belongs to the overpass (IfcWalls), as shown at the bottom-left corner in Figure 4b. They are not rendered at the same time since they are encapsulated in two different b3dm files that are referenced in various hierarchies in the tileset JSON file. Figure 5 is a blended image, the top-half shows a real-world photo of the building and the bottom-half shows the converted 3D Tiles model.

(**a**) (**b**)

Figure 4. (**a**) Highlighted individual IFC component in the converted 3D Tiles; (**b**) query properties for each component from 3D Tiles.

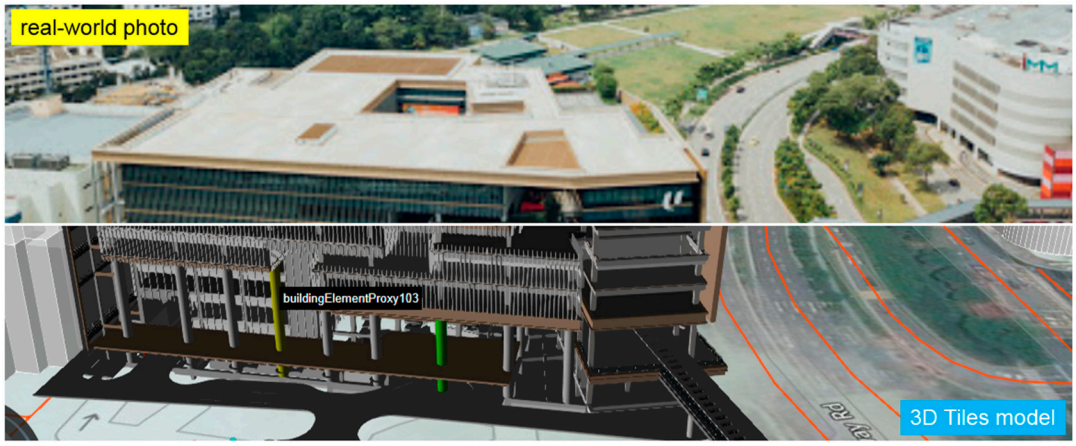

Figure 5. The converted 3D Tile model blends with its real-world photo from the same viewpoint.

5. Discussion and Conclusions

The IFC to 3D Tiles conversion is a time-consuming and resource-intensive process. Despite all the efforts devoted to automating the conversion, some manual adjustments are still required to properly render the created 3D Tiles rendered in Cesium. Since the model conversion is usually considered as one-shot process, it is worthy of the effort.

During the final glTF to b3dm conversion step, the way to construct the b3dm files from the individual component largely depends on the application requirements. Various strategies can be applied to organise the models in a specific hierarchy that best suits the application. For simplicity of implementation, the current composer wraps all components of the same IFC type into a b3dm file. This is not a good strategy, and the drawbacks are clear. First, it does not balance the size of the generated b3dm files and some b3dm files are much bigger than the others. For example, in Table 4, the b3dm file for IfcDoors is 166 times larger than that of IfcWindows, and it requires considerably more time to load the doors into the scene. Second, this strategy does not consider Hierarchical Level of Detail (HLOD), which is critical to improving the rendering performance.

3D Tiles uses geometric error to define the error in meters when a tile is rendered. Tiles are structured into a tree incorporating HLOD to determine if a tile is sufficiently detailed for rendering and if the content of tiles should be successively refined by children tiles of higher resolution. For the most simplified implementation, a root tile has the greatest geometric error, and each successive level of children has a lower geometric error than its parent. Leaf tiles typically have a geometric error of 0. To follow this principle, a systematic method needs to recognise the spatial relationships of the objects (such as external walls, windows, internal stairs, slabs, and floors) to group objects with better hierarchy. However, to achieve this, either an automated or manual process would require considerable implementation time.

As for the rendering material of 3D models, the current glTF files created and embedded in b3dm file follows the KHR_materials_common extension [17]. The PBR_metallic_roughness material extension [18] introduced in glTF2.0 could improve the model quality. More sophisticated b3dm shaders could also increase the overall quality at the expense of rendering time.

BIM-GIS integration is a critical and challenging task and received considerable attention from both BIM and GIS communities. With the emphasis on visualising and interacting with BIM on a 3D WebGIS platform (i.e., Cesium), this paper first examined the current solutions and their defects. We then proposed an integrated IFC-3D Tiles conversion workflow by using a series of open-source tools and libraries. A composer tool (open-sourced on Github) was developed to automate the four conversion steps. The feasibility of the proposed approach was tested with real BIM, and the performance of the conversion tools was gauged and discussed based on the processing time and intermediate disk usage for each process step and each IFC type. Though the current composer is

premature and has room for future improvements, it has the potential to be used and enhanced by both communities.

Author Contributions: Conceptualization, Y.C. and S.S.; Methodology, Y.C.; Software, E.S. and Y.C.; Validation, S.S. and A.R.; Formal Analysis, Y.C. and E.S.; Investigation, Y.C. and E.S.; Resources, A.R.; Data Curation, S.S.; Writing-Original Draft Preparation, Y.C.; Writing-Review & Editing, A.R., S.S. and E.S.; Visualization, Y.C. and E.S.; Supervision, A.R.; Project Administration, Y.C. and S.S.

Funding: This research received no external funding

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Open-Source Tools and Libraries Used in This Work

BimServer API
https://github.com/opensourceBIM/BIMserver

IfcOpenShell IFC2X3 IfcConvert 0.5.0-dev
http://ifcopenshell.org/ifcconvert.html
https://github.com/IfcOpenShell/IfcOpenShell

obj2gltf
https://github.com/AnalyticalGraphicsInc/obj2gltf

3d-tiles-tools sample generator
https://github.com/AnalyticalGraphicsInc/3d-tiles-tools

Appendix B. BIMServer API Queries

General IfcProduct Query:

```
{
  "type": {
   "name": "IfcProduct",
   "includeAllSubTypes": true
 },
  "includes": [
   "validifc:ContainedInStructure",
   "validifc:OwnerHistory",
   "validifc:Representation",
    "validifc:ObjectPlacement",
    "validifc:AllProperties"
 ],
  "oid": "Object's oid"
 }
```

Queries for IfcTypes with openings:

```
{
  "type": {
    "name": "Object type (e.g. IfcWall or IfcSlab)",
    "includeAllSubTypes": true
 },
 "includes": [
    "validifc:ContainedInStructure",
    "validifc:OwnerHistory",
   "validifc:Representation",
    "validifc:ObjectPlacement",
    "validifc:AllProperties"
```

```
],
 "oid": "Object's oid",
  "include": {
    "type": "Object type (e.g. IfcWall or IfcSlab)",
    "field": "HasOpenings",
    "include": {
      "type": "IfcRelVoidsElement",
      "field": "RelatedOpeningElement",
      "include": {
       "type": "IfcOpeningElement",
       "includes": [
         "validifc:ContainedInStructure",
         "validifc:OwnerHistory",
         "validifc:Representation",
         "validifc:ObjectPlacement",
         "validifc:AllProperties"
      ]
  },
     "includes": [
     "validifc:ContainedInStructure",
     "validifc:OwnerHistory",
      "validifc:Representation",
     "validifc:ObjectPlacement",
      "validifc:AllProperties"
     ]
  },
     "includes": [
     "validifc:ContainedInStructure",
     "validifc:OwnerHistory",
      "validifc:Representation",
      "validifc:ObjectPlacement",
      "validifc:AllProperties"
     ]
  }
}
```

Queries for objects with aggregate components:

```
{
 "type": {
   "name": "Object type (e.g. IfcCurtainWall or IfcStair)",
   "includeAllSubTypes": true
  },
 "includes": [
   "validifc:ContainedInStructure",
   "validifc:OwnerHistory",
    "validifc:Representation",
   "validifc:ObjectPlacement",
   "validifc:AllProperties"
  ],
  "oid": "Object's oid",
  "include": {
```

```
"type": "Object type (e.g. IfcCurtainWall or IfcStair)",
"field": "IsDecomposedBy",
"include": {
 "type": "IfcRelAggregates",
 "field": "RelatedObjects",
"includes": [
"validifc:ContainedInStructure",
"validifc:OwnerHistory",
"validifc:Representation",
 "validifc:ObjectPlacement",
 "validifc:AllProperties"
]
},
 "includes": [
"validifc:ContainedInStructure",
"validifc:OwnerHistory",
"validifc:Representation",
 "validifc:ObjectPlacement",
"validifc:AllProperties"
]
}
}
```


References

1. Cesium. Available online: https://cesiumjs.org/about/ (accessed on 4 February 2018).
2. glTF. Available online: https://www.khronos.org/gltf/ (accessed on 1 March 2018).
3. Murshed, S.M.; Al-Hyari, A.M.; Wendel, J.; Ansart, L. Design and Implementation of a 4D Web Application for Analytical Visualization of Smart City Applications. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 276. [CrossRef]
4. 3D Tiles Specification. Available online: https://github.com/AnalyticalGraphicsInc/3d-tiles/tree/master/specification#tileset-json (accessed on 4 February 2018).
5. 3D Tiles. Available online: https://cesium.com/blog/2015/08/10/introducing-3d-tiles/ (accessed on 4 February 2018).
6. Amirebrahimi, S.; Rajabifard, A.; Mendis, P.; Ngo, T. A BIM-GIS integration method in support of the assessment and 3D visualisation of flood damage to a building. *J. Spat. Sci.* **2016**, *61*, 317–350. [CrossRef]
7. Isikdag, U.; Zlatanova, S. Towards defining a framework for automatic generation of buildings in CityGML using Building Information Models. In *3D Geo-Information Sciences: Lecture Notes in Geoinformation and Cartography*; Lee, J., Zlatanova, S., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 79–96.
8. El-Mekawy, M.; Ostman, A. Semantic mapping: An ontology engineering method for integrating building models in IFC and CityGML. In Proceedings of the 3rd ISDE Digital Earth Summit, Nessebar, Bulgaria, 12–14 June 2010.
9. Karimi, H.A.; Akinci, B. *CAD and GIS Integration*, 1st ed.; Karimi, H.A., Akinci, B., Eds.; CRC Press: Boca Raton, FL, USA, 2010.
10. El Meouche, R.; Rezoug, M.; Hijazi, I. Integrating and managing BIM in GIS, software review. In Proceedings of the ISPRS 8th 3DGeoInfo Conference & WG II/2 Workshop, Istanbul, Turkey, 27–29 November 2013.
11. Arroyo Ohori, K.; Biljecki, F.; Diakite, A.; Krijnen, T.; Ledoux, H.; Stoter, J. Towards an integration of gis and bim data: What are the geometric and topological issues? In Proceedings of the 12th 3D Geoinfo Conference on ISPRS Annals of the Photogrammetry Remote Sensing and Spatial Information Sciences, Melbourne, Australia, 26–27 October 2017; Volume IV-4/W5.
12. Eastman, C.; Teicholz, P.; Sacks, R.; Liston, K. *BIM Handbook: A Guide to Building Information Modeling for Owners, Managers, Designers, Engineers and Contractors*, 2nd ed.; Wiley: Hoboken, NJ, USA, 2011; ISBN 978-0-470-54137-1.

13. Hijazi, I.; Ehlers, M.; Zlatanova, S. BIM for geo-analysis (BIM4GEOA): Set up of 3D information system with open source software and open specifications (OS). In Proceedings of the 5th International 3D Geoinfo Conference, Berlin, Germany, 3–4 November 2010; pp. 45–49.
14. Stouffs, R.; Tauscher, H.; Biljecki, F. Achieving Complete and Near-Lossless Conversion from IFC to CityGML. *ISPRS Int. J. Geo-Inf.* **2018**, *7*, 355. [CrossRef]
15. Cesium v1.35. Available online: https://cesium.com/blog/2017/07/05/cesium-version-1.35-released/ (accessed on 4 February 2018).
16. Mago3D. Available online: http://www.mago3d.com/homepage/spec.do (accessed on 28 July 2018).
17. KHR Materials Common. Available online: https://github.com/KhronosGroup/glTF/tree/master/extensions/1.0/Khronos/KHR_materials_common (accessed on 28 July 2018).
18. PBR Metallic Roughness. Available online: https://github.com/KhronosGroup/glTF-WebGL-PBR (accessed on 28 July 2018).