Design and Implementation of an IFC Data Model to Unity Data Model Transformation Mechanism

Zhong-Qi Wu¹ and I-Chen Wu²

1) Graduate Student, Department of Civil Engineering, National Kaohsiung University of Applied Sciences, zhongqi1105@gmail.com
2) Associate Professor, Department of Civil Engineering, National Kaohsiung University of Applied Sciences, kwu@kuas.edu.tw

Abstract:

The construction industry is moving towards the application of Building Information Modeling (BIM) for more efficient management and integration of engineering information. Today, engineering consultant companies, construction companies, and architectural companies are investing much research into the development of BIM to integrate and visualize engineering information for greater productivity. However, a lack of interactivity is a shortcoming of BIM applications. The core functionality typically provided by a game engine includes a rendering engine, a physics engine, animation, and so on, which can give a user the experience of an immersive virtual reality. By sharing BIM data to a game engine, a user can gain more operation and interaction capabilities. Moreover, it can increase the range of possible applications. BuildingSMART has developed Industry Foundation Classes (IFC) that make it possible to hold and exchange relevant data between different BIM applications during a building’s life cycle. This research designs and implements a transformation mechanism for transforming an IFC Data Model to a Unity Data Model, so data can be imported into Unity to facilitate BIM visualization and manipulation. Moreover, a game engine can support cross-platform programming to enhance portability of BIM data for advancing the usage and application of BIM models.

Keywords: BIM, IFC, Unity, Game Engine

1. INTRODUCTION

In recent years, construction management and other concepts have received increasing attention. Traditional two-dimensional drawings are no longer able to meet the needs of today's construction projects. In order to effectively integrate and manage large project-related information, the current trend is to import Building Information Modeling (BIM) (Eastman, 2008). The BIM concept is mainly based on data integration and visualization. It is hoped that all the project data required during an entire project life cycle, including project planning, construction and building, operation and maintenance, or even demolition, can be integrated into a single model. This method can effectively help different parties from various professions in terms of project management and communication. Owners without any civil engineering background can also have a better understanding of the progress and development of the project.

In terms of the development of BIM technology, there are currently many software manufacturers that are developing BIM-related software, but these manufacturers have different software architectures and different software, and they focus on different areas of expertise. As a result, it is impossible to have complete integration or communication between BIM software. This is against the original intention of the BIM concept. In order to promote the concept of Open BIM, so that the interaction between BIM-related industries will no longer be limited by software, the buildingSMART (2015) organization developed the Industry Foundation Classes (IFC) file format. It is hoped that it can be a master format for various types of BIM software, in order to reduce the gaps between software from different manufacturers, and thereby enhance the ability for each party involved in a project to communicate (Liu, 2010; Fu, 2006; Chen, 2005).

However, currently, BIM generally interacts poorly with engineering applications. Meanwhile, a game engine has a large amount of multimedia elements, and it can provide realistic rendering technology, lighting and visual effects, physical behaviors, etc. Moreover, it can fully simulate the real-time status of a project with high authenticity. In addition, the game engine itself also has a comprehensive development environment, including open API, hardware integration, and cross-platform output functions. It can provide good interactions between users. The above-mentioned advantages can provide users of a BIM model with a more realistic and intimate experience. As a result, if the BIM model can be used in conjunction with the game engine, then we shall have more possibilities in terms of development and application.

For these reasons, this study uses IFC as a source of information for BIM, and develops a model for data
exchange. We attempt to import the BIM model into the Unity3D game engine (Unity Scripting Reference, 2015), and then export a package file exclusively for Unity3D. The aim is to facilitate BIM staff in developing various man–machine interfaces and management functions in the Unity3D platform, so that we can improve the interaction and development of a BIM model.

2. IFC DATA MODEL

The IFC format is mainly used to describe professional construction information related to buildings and structures. It supports the exchange and integration of information during each stage of the construction life cycle. The IFC is constantly updated, in order to meet the demand for information by today’s construction industry. The latest version is IFC 2x4. The version that is commonly supported by BIM-related software is IFC 2x3. In this study, we use IFC 2x3 to perform format conversion for the information of the model.

2.1 IFC Database

The IFC format has many rigorous standards for data access. Not only can it cover engineering- and management-related information for an entire project life cycle, it also has a considerable flexibility for expansion so that it can provide users with different needs with the ability to add new information structures. It is suitable for use as a master standard for various types of BIM-related software. However, since the structure is considerably large, and maintenance and access of the file is very difficult, retrieving a set of data is very time-consuming, especially when the model is large. In order to improve the usability of the IFC file, it is necessary to have the file repository be a database. Through the use of a database, one can more easily view the required data in the IFC file. It is also easier to maintain data within the IFC (Huang, 2015; Chang, 2015).

2.2 B-rep and Surface Model

The IFC uses B-rep to describe irregular and complex models (as shown in Figure 1, below). Through the use of a matrix description by IFCPOLYLOOP, we use three or more points to form a plane, and use multiple IFCPOLYLOOPs to form a solid body. We can then use multiple solid bodies to form a complete object. Using this method, we can describe various complex models. B-rep has the advantage of offering a simple description, as well as offering descriptions with multiple changes. However, owing to the large number of points that need to be stored and accessed, a disadvantage is that a considerable amount of resources is consumed. Thus, this technique is often used to describe irregular exteriors, or objects that cannot be standardized.

![Figure 1. A B-rep Model: (a) the BIM Model and (b) the B-rep Mesh Model.](image)

2.3 Swept Solid

The IFC’s description of a sweeping object includes sweeping in a straight line, sweeping in a curved line, and sweeping in circles. The basic description of sweeping requires one to first describe a cross section, and then describe the sweeping axis and direction of the cross section, in order to perform sweeping, and then form the shape. To describe the cross section, one can have a rectangular or a polygonal description. A rectangular description can describe the length, width, center, plane displacement, etc. of the object. A polygonal description uses IFCPOLYLINE, and describes the contour of the cross section in sequence using a matrix. Sweeping is then performed on the cross section. The coordinates of the swept objects are transformed, and the object is completed. The changes in Swept Solid are not as diverse as in B-rep. One only needs the parameters of the cross section, the direction of the axis, the sweeping distance, and the sweeping type to perform the calculation.
(as shown in Figure 2). It is beneficial in reducing the amount of data required for geometric description and reducing the burden for data processing.

![Figure 2. Swept Solid: (a) the BIM Model and (b) the Swept Solid Model.](image)

3. **UNITY DATA MODEL**

Unity3D uses GameObject as the unit for each individual object in the game scene. Each GameObject has its individual components, including Script, Rendering, and UI. These components can have hierarchy relationships, such as a parent–child relationship (as shown in Figure 3, below). They can influence each other in terms of location, orientation, and size. MeshFilter stores the mesh information of the GameObject. It helps MeshRenderer to present and render the established mesh objects in the scene. Collider controls the collision mechanism for the objects. For example, MeshCollider will follow the Mesh data in the MeshFilter, and form the corresponding mesh for collision. When a Rigidbody object is being touched, the physical engine in Unity3D will be triggered, and the corresponding event will be executed.

![Figure 3. The Structure of GameObject.](image)

4. **TRANSFORMATION MECHANISM**

A flowchart for the IFC Data Model to Unity Data Model Transformation developed by this study is shown in Figure 4. Users can input requirements for exporting Database, Collections, Data Model, and the Object Type of the IFC model, and the model will collect the requests and submit them to the server. Once the server receives the request, the server will perform an inquiry to the IFC database, and will transform the corresponding object information to the Json format, and return the information back to the user end. Once the model receives the returned IFC data, the data model module will be used to transform the data into a Mesh object model, and the transformed models will be assigned to the GameObject in Unity3D.

The final task is communication between the projects and data transfer. We store the results of the transformation as Unity Package. The file unitypackage saves the objects, components, scripts, and even classes selected under the project. Under the Unity interface, to export the package, we only need to right-click on the
project folder and select Export Package. For this model, we can export files for the solid body using the automatically generated object and Mesh components. For the GameObject established by the Data Model Transformation Module, its components are only stored in the established scenes, and they are not stored in the project folders. If we save the GameObject to the project using the .prefab format, then we will lose the Mesh components in GameObject. As a result, we must save the Mesh components in GameObject to the project in the .asset format and then export the GameObject. This way, the links between GameObject and Mesh components will be saved. Finally, we export the package using “Export Package” function.

Figure 4. A flowchart for the IFC Data Model to Unity Data Model Transformation Mechanism.

The functions of the conversion module include Coordinate Transformation, Mesh Generation, and UV Mapping, as described below.

4.1 Coordinate Transformation

When we want to change the coordinate system from IFC to Unity3D, we only need to set the X-, Y-, and Z-axis vectors and the displacement vector to (1, 0, 0), (0, 0, 1), (0, 1, 0), and (0, 0, 0), respectively, for every coordinate point.

The IFC uses the Cartesian right-hand coordinate system to describe the relationship of the model in a three-dimensional space. However, there is more than one method to define the coordinates. Unity3D uses a left-handed coordinate system where the Y-axis is pointing upward. Provided that the positive direction of the X-axis does not change, the positive directions for the Y- and Z-axes are reversed when compared to the IFC coordinate system. Huang and Wu (2015) studied the issues of using different coordinate systems, and used matrix transformation of the models to transform the directions of the IFC objects to the correct position (Equation 1) through the use of a coordinate transformation equation for homogeneous coordinates.

The method that IFC uses to describe the position of an object is through an object-oriented class. The classes are packaged in the following order: Global → Entity → Item. Each class describes the “corresponding” direction vector and displacement in the X- and Z-axes in the next class. By using the external mechanism, we can obtain the direction vector in the Y-axis. By substituting the values into Equation 1, we can complete the coordinate conversion for each class.

The coordinate transformation formula is:

\[
\begin{bmatrix}
X_x & Y_x & Z_x & P_x \\
X_y & Y_y & Z_y & P_y \\
X_z & Y_z & Z_z & P_z \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
OP_x \\
OP_y \\
OP_z \\
1
\end{bmatrix}
= \begin{bmatrix}
OP'_x \\
OP'_y \\
OP'_z \\
1
\end{bmatrix},
\]

(1)

where:

- \((X_x, Y_x, Z_x)\): X-axis vector
- \((Y_x, Y_y, Y_z)\): Y-axis vector
- \((Z_x, Z_y, Z_z)\): Z-axis vector
- \((P_x, P_y, P_z)\): Displacement vector
- \((OP_x, OP_y, OP_z)\): Object position vector.
4.2 Mesh Generation

In a model, the basic units for the exterior of its components are simple polygons. A simple polygon can be either a convex polygon or a concave polygon. The difference between the two depends on whether it has more than 180° (if it has points where the vector product is negative). Taylor implemented a method using a mesh to represent any arbitrary simple polygon (Polygon Triangulation, 2015). When establishing each triangular mesh, it is necessary to strictly abide by the following conditions. (1) Each observation point can only form a triangular mesh with the points before and after it. (2) Each triangular mesh shall not have any other point. (3) The normal vectors for each triangular mesh must be the same as the normal vectors of the simple polygon (the described direction of the triangular mesh should not be reversed). (4) Once a triangular mesh is completed, the observation point is to be removed from the data structure. Based on the above method, we can produce a mesh for any simple polygon. The workflow is shown in Figure 5.

![Figure 5. A Flowchart for Mesh Generation.](image)

4.3 UV Mapping

In order to have the UV coordinates of the triangular mesh on the same plane, so that they can be connected in series without causing a mapping fault, the UV module will first store the coordinate points before establishing the triangular mesh. When we complete a full polygon, the user or the model can automatically assign the corresponding UV coordinates to each point. Before adding new UV coordinates, the model performs a normalization of all the already stored points, and resets the original point to (0, 0, 0). The model also filters polygons with normal vectors parallel to the X-, Y-, and Z-axes. The polygons are projected in a two-dimensional plane (as shown in Figure 6, below). After image scaling, we can obtain UV coordinates for the polygon mesh.

![Figure 6. UV Mapping.](image)
5. DISCUSSION AND CONCLUSION

During the construction of a project, in order to effectively integrate and manage engineering content, importing building information modeling (BIM) technology is a current trend. However, in general, interactions with BIM and the development of BIM are insufficient. Importing a BIM model into a game engine can dramatically improve BIM allowing a wider range of applications and interactions. The first step for importing a BIM model is to render the geometric information of the solid body. How to render a variety of complex graphics in a way that will comply with the game engine is also an important research topic. In this study, we developed a geometric conversion mechanism for transforming an Industry Foundation Classes (IFC) data model to a Unity3D data model, enabling one to successfully import a BIM model into the Unity platform. It facilitates future users with regard to interactive design and program development. It is hoped that the BIM model can be extended to another platform, and that there will be further technology-related applications for BIM.

This study is currently only for IFC’s B-Rep and Swept Solid geometry models. In the future, it is expected that support for the rest of the CSG model will be included, so that the exterior of the BIM model can be imported into the game engine more completely. In terms of information access, the technique is currently limited to supporting information about the exterior of the model, while the inside of the IFC model still contains an amount of non-graphical information. In the future, we expect to be able to add modules to access engineering information so that the BIM model can perform a more complete conversion, and that the BIM and the game engine can be more completely integrated.

ACKNOWLEDGMENTS

The authors would like to thank the R.O.C. Ministry of Science and Technology for their financial support under Grant No. MOST 103- 2221- E -151-021-MY3.

REFERENCES