

## Integration of Java-based BIM with spatial database

S. Malaikrisanachalee<sup>1,\*</sup>, H. Vathananukij<sup>2</sup>

Received: May 2010, Accepted: November 2010

### Abstract

Java is an object-oriented program that has abundant open-source libraries for application development and 3D model rendering. Spatial database is the database that can efficiently store and manage geographic information data through various spatial data management techniques. This paper explores the rationale of coupling java with spatial database to develop an effective platform for future Building Information Modeling (BIM) application. The paper methodically presents the prototype system integration design to demonstrate how the system can be developed. The paper also meticulously presents the logical and physical data models in designing optimum BIM database for a reinforced concrete building. An 8-storey reinforced concrete building was used as an implementation case study to validate the proposed prototype system design and investigates the implementation issues. The outcome shows that not only the proposed prototype system offers technological advantages over the traditional BIM applications, its open-source solution can also overcome the financial constraint that currently inhibits the implementation of BIM especially for medium and small enterprises.

Keywords: BIM, spatial database, project management, construction database, 3D model.

### 1. Potentials of Java-based BIM

Building Information Modeling (BIM) has garnered momentous attention from the architecture, engineering and construction (AEC) industry in recent years since BIM potentially can minimize project time, cost and conflict as well as improve project communication, coordination and simulation [1,2]. BIM lies on the basis of using a three-dimensional (3D) digital model for managing spatial, aspatial and temporal information of building elements to support decision makings in project analysis, design, planning, construction and monitoring throughout the project life cycle. Java provides a competent platform for BIM application development as it comes with abundant open-source libraries and operational graphical user interface classes that offer array of solutions for application development. Java is also equipped with the high-level 3D graphic application programming interface that provides building blocks for composing and animating 3D objects, which can ease the development process for 3D visualization and animation in

BIM application [3,4]. In addition, java generates binary executable that can run on any operating system and hardware platform, which enables the application to run on any computer without restriction.

Since java is an object-oriented program, it supports modeling of real-world building elements where spatial, aspatial and temporal information of building elements can be efficiently encapsulated by the notion of class, object and method. Furthermore, the object-oriented approach potentially provides a means for establishing relationship constraints between building elements. For example, relationship constraints can be established between slabs, beams and columns where slabs must always be connected to their supporting beams and beams must always be connected to their supporting columns. This makes the objected-oriented 3D model logically more realistic as opposed to the traditional non-object 3D model, whose building elements are simply represented by simple geometric objects (e.g. point, line, polygon and box) without associated information of real-world building elements.

Through java applet or servlet technology, java potentially can support the implementation of web-based BIM and allow effective integration between BIM and an enterprise project management system in order to allow project participants to easily access building information and closely monitor project status through the Internet. In addition, with the availability of

\* Corresponding Author: fengshm@ku.ac.th  
1 Assistant Professor, Faculty of Engineering, Kasetsart University, 50 Phaholyothin Rd., Chatuchak, Bangkok 10900  
2 Associate Professor, Faculty of Engineering, Kasetsart University, 50 Phaholyothin Rd., Chatuchak, Bangkok 10900

java-based 3D geospatial viewer technology such as Nasa's World Wind Java, it is possible to integrate java-based BIM with Geographic Information System (GIS) to enhance the application potential to the like of city information modeling [5].

## 2. Prototype system integration design

Spatial database management system can serve as a good data management platform for BIM as it offers an efficient means for managing and analyzing building geometry data where size, shape and position of building elements can be efficiently encoded, processed and retrieved through spatial data management techniques. Figure 1 illustrates an overview of system integration between java-based BIM with spatial database where geometry data of building elements as well as their related attributes are manipulated by PostgreSQL, the open-source object-relational database management system, and PostGIS, the geospatial extension to PostgreSQL that allows spatial data to be stored inside PostgreSQL database.

Integrating spatial database with BIM can greatly improve flexibility for BIM data modeling and data query performance. Since PostGIS supports various types of 2D, 3D and 4D spatial objects as summarized in Table 1 [6], it provides superior modeling platform for spatio-temporal data of building elements. Furthermore, PostGIS can store spatial data in binary format, known as Well Known Binary (WKB), which needs less data storage and yields less data processing time than the traditional ASCII-based coordinate strings. In addition, PostGIS includes support for spatial index called Generalized Search Trees (GiST), which can be used to drastically reduce query time for a large BIM database.

PostGIS includes support for a number of spatial queries such

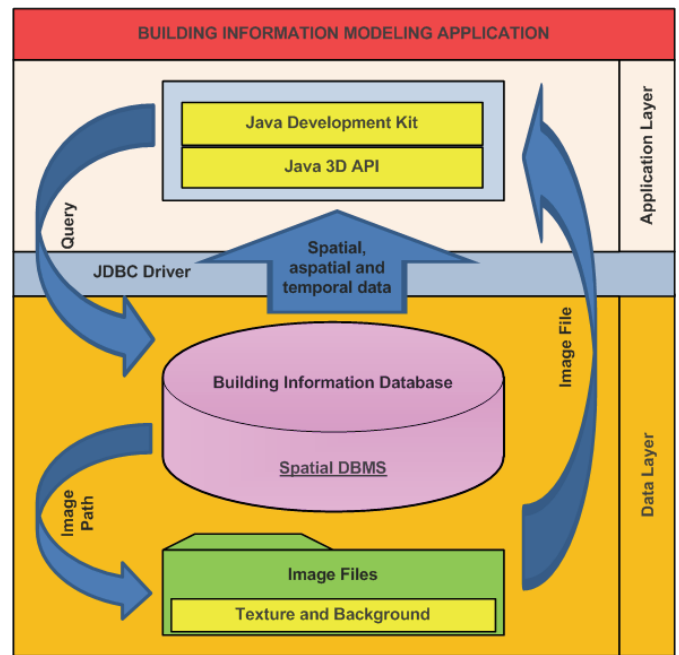


Fig. 1. Prototype System Integration Design

as distance, length, boundary, centroid and area calculations that can serve as useful tools for quantity takeoff in estimating building cost. PostGIS also provides spatial queries for examining spatial relation between building elements including union, intersection, equal, touch, contain, within and buffer operations. These queries can be grateful for verification of building design conflicts as well as evaluation of topological relationship between building elements.

The 3D model application interface includes the standard Java Development Kit (JDK), java 3D API (javax.media.j3d), java 3D utility (com.sun.j3d.utils), java 3D vector math (javax.vecmath) and java abstract window toolkit (AWT) (java.awt) [7]. The java 3D API, widely known as java 3D core, is the minimum requirement for creating, rendering and manipulating 3D objects while the java 3D utility compliments the java 3D core to ease the complication of java 3D programming. The java 3D vector math is needed for supporting multidimensional mathematical objects that are used in java 3D programming while the java abstract window toolkit provides an interface for 3D rendering.

Supplemental image files are used for background and texture setting [7]. Java sets texture of an object by loading an image file from an external folder and draping it over the object to make the object look more realistic. Texture may be predefined based on object type and can be changed on-the-fly per user request. Java supports most types of image files and there is not minimum requirement for image size or resolution. However, an image with higher resolution provides finer texture quality while using more processing time.

Connection between the 3D model application interface and PostgreSQL database can be established through the JDBC driver that provides protocol for java to connect to an external database. It is noted that PostGIS also provides the JDBC extension objects that provide useful data access functions for java to conveniently manage geometry objects in PostgreSQL database [6].

Table 1. Spatial Data Models Supported by PostGIS

Geometry Type	Geometry Dimension		
	2D (x,y)	3D (x,y,z) or (x,y,m <sup>1</sup> )	4D (x,y,z,m <sup>1</sup> )
Point	SF <sup>2</sup>	ESF <sup>3</sup>	ESF <sup>3</sup>
Line	SF <sup>2</sup>	ESF <sup>3</sup>	ESF <sup>3</sup>
Polygon	SF <sup>2</sup>	ESF <sup>3</sup>	ESF <sup>3</sup>
Multi Point	SF <sup>2</sup>	ESF <sup>3</sup>	ESF <sup>3</sup>
Multi Line	SF <sup>2</sup>	ESF <sup>3</sup>	ESF <sup>3</sup>
Multi Polygon	SF <sup>2</sup>	ESF <sup>3</sup>	ESF <sup>3</sup>
Geometry Collection	SF <sup>2</sup>	ESF <sup>3</sup>	ESF <sup>3</sup>
Curve	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>
Compound Curve	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>
Multi Curve	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>
Curve Polygon	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>
Multi Surface	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>	SQL-MM <sup>4</sup>

<sup>1</sup>Measure value such as linear referencing measurement or Unix time

<sup>2</sup>Simple Features for SQL specification defined by OGC

<sup>3</sup>Extended Simple Features for 3D and 4D spatial data representation

<sup>4</sup>SQL Multimedia specification for circularly interpolated curves

### 3. BIM data modeling for reinforced concrete building

Figure 2 presents the logical data model of a reinforced concrete building in a spatial database. As can be seen, a reinforced concrete building is an aggregation of Entity, which represents an abstract class of all building elements. Entity can be divided into two subclasses: Structure and System. Structure is an abstract class of building's structure elements including footing, beam, floor, column, wall and stair. Structure consists of Geometry, Reinforcing, Material and Time classes.

Geometry, which illustrates location, shape and size of each structure element, is implicitly represented by aggregation of element's cross sections. Rather than explicitly encoding coordinate strings of all geometry vertices, this modeling approach can greatly reduce data structure complexity as well as physical storage space by simply maintaining the geometry data of element cross sections at its head and end. For example, if geometry of beam's head and tail sections are known, other geometrical attributes of the beam including length, volume, surface area, centroid and angular alignment can be implicitly derived using spatial queries provided in spatial database. Nevertheless, if cross section of a structure element is not consistent throughout, an intermediate cross section is needed at the location where the cross section is changed.

Reinforcing is a representation of steel reinforcing details in each structure element. Reinforcing is an aggregation of Steel,

which illustrates detail of each reinforcing steel. Steel can be divided into two subclasses: Main and Stirrup. Main represents major reinforcing steel in a structure element while Stirrup represents shear reinforcing steel. Both Main and Stirrup inherits common properties from Steel, which includes Geometry and Property. Geometry represents locations and alignments of Steel while Property represents physical properties including unit cost of Steel. In order to minimize data storage, rather than storing geometry data of all individual steel bars, Geometry shall store only geometry data of one referenced steel bar through Reference class while geometry data of other steel bars can be implicitly encoded through Offset class.

Material represents the physical attributes of material (e.g. concrete) used for each structure element. Material shall also include unit cost for supporting automatic cost estimation of the element. Time represents the temporal aspect of each structural element. Typically, Time includes construction schedule and actual construction time of the element.

System is a representation of each system component in a building. System can be divided into two subclasses: Mechanical and Electrical. Both Mechanical and Electrical have common properties, which includes Geometry, Material, Cost and Time that inherits from System. Geometry illustrates shape and location of each system component while Material represents physical attributes of the component. Time and Cost represent temporal and financial aspects, respectively, regarding construction, operation and maintenance of each system component.

### 4. Prototype system implementation

An 8-storey reinforced concrete parking building was used as an implementation case study in integrating java-based BIM with spatial database. Figure 3 illustrates the physical database of the building that was derived from the aforementioned logical data model. Several interesting notions in creating and populating the database include:

" Explicit relationship between the Structure and System tables is not available. Rather, the two tables have implicit spatial relationship that can be derived through their geometries.

" Since geometry of each system component can come in various shape and dimension, the geometry field in the System table shall use the Geometry Collection data type provided by PostGIS, which can support modeling of point, line and polygon altogether in one tuple to attain data modeling flexibility.

" While the reference\_geometry field in the Reinforce table is meant to store geometry data of the referenced reinforcing steel bar, the x\_offset, y\_offset and z\_offset fields in the Offset table is used to store the offset information between the referenced and its successive reinforcing steel bars. The number field in the Offset table indicates the number of reinforcing steel bars.

" Since PostGIS provides flexibility to accurately encode complete reinforcing details such as bar hooks, splices and spiral stirrup, the estimated volume and weight of reinforcing steel can be calculated more accurately.

" The geometry field in the Section table shall use

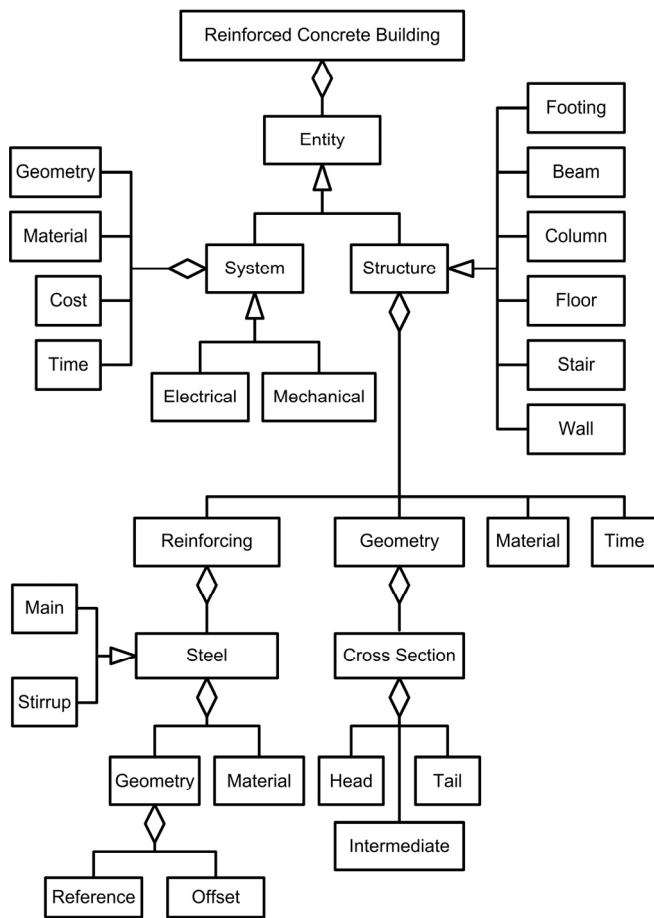


Fig. 2. Logical Data Model of Reinforced Concrete Building

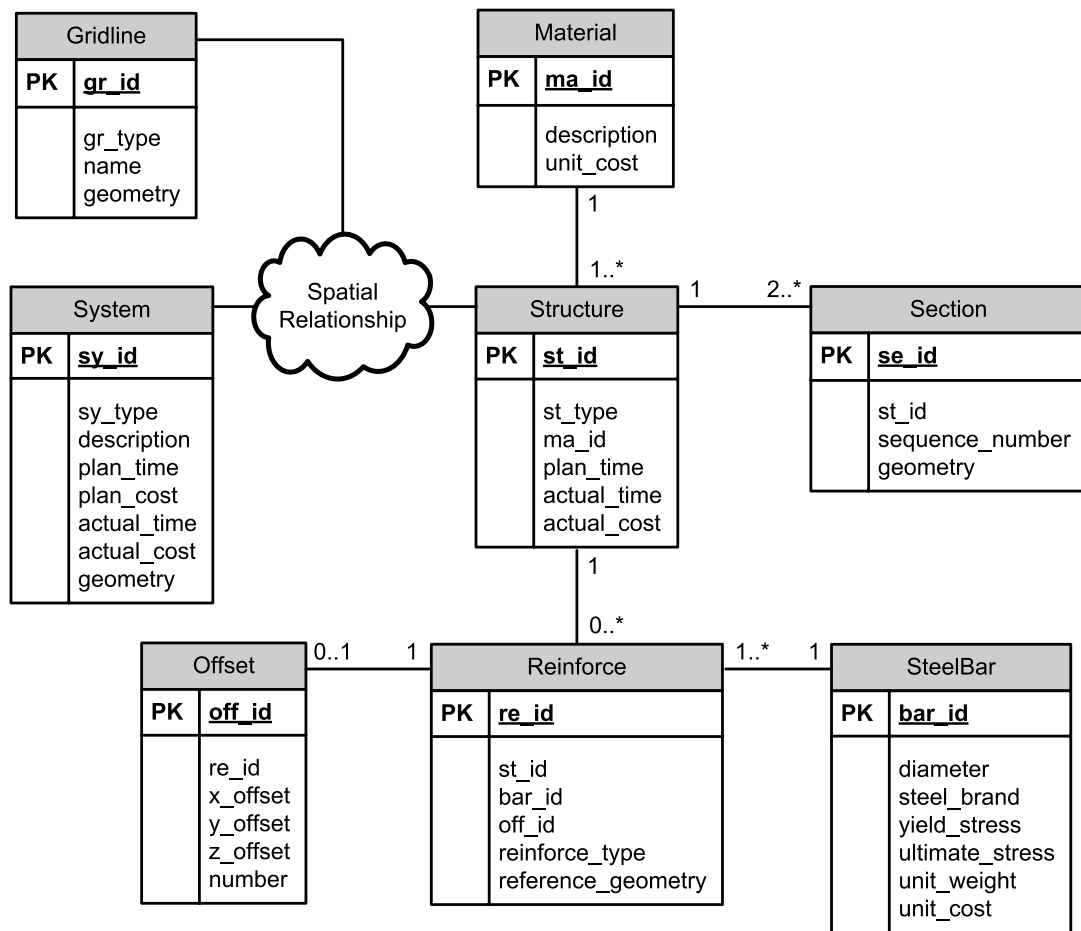


Fig. 3. BIM Database for Reinforced Concrete Building

MultiPolygon data type provided by PostGIS to achieve flexibility to model the structure element with hollow or irregular shape.

" The sequence\_number field in the Section table indicates the position (e.g. head, intermediate, tail) of that section on the structure element.

" The budget cost of each structure element can be automatically calculated from the estimated material and steel reinforcing amounts time their unit costs. However, the actual cost for the entire structure element shall be directly entered into the actual\_cost field in the Structure table to simplify the cost tracking process.

" The Gridline table allows transformation between the Cartesian coordinate system and human-understandable grid system and vice versa to improve project communication. While PostGIS can support both local and real-world Cartesian coordinate systems, such coordinates cannot be used to communicate effectively among project participants. For example, construction workers would not understand if they are told to dig a hole at coordinate (5, 5, 0). Instead, they could understand if they are told to dig a hole at certain referenced grid location.

" GiST index, the spatial index provided by PostGIS, shall be generated on all geometry fields in the database to minimize processing times for spatial operations and queries of building geometry data.

" The PostgreSQL database maintains transactional integrity

using multiversion model (Multiversion Concurrency Control) where multiple users can access and edit BIM data at the same time without causing a problem to the database.

Figure 4 illustrates the programming approach in generating 3D objects in java. Initially, the Canvas3D needs to be created. The Canvas3D is an instance of the Canvas class in the AWT package that provides an interface for displaying java 3D objects. The SimpleUniverse from the java utility package is constructed to provide 3D locale for viewing java 3D objects through scene graphs.

The BranchGroup (BG) is the root of a scene graph that is inserted into the locale. One scene graph is constructed for each object type to divide a 3D building model into multiple layers. For example, a 3D building model may include ten different layers: pile, footing, beam, column, slab, wall, roof, door, window, and steel reinforcing. This approach improves the ease of use where a user can choose the layers to be displayed.

The TransformGroup (TG) controls spatial transformation of an individual 3D object. For example, if a building has twenty beams, all twenty beams are referenced to the same BranchGroup but the location, width, length, height, rotating angles, and texture of each individual beam are configured through the TransformGroup. In generating each 3D building element in java, the geometry data are manipulated and retrieved from the spatial database and used to construct the TransformGroup of that element.

Figure 5 presents different views of the 8-storey reinforced concrete parking building created by java. The java 3D model can automatically generate different views of the building: front, back, side, top, plan and section views. However, due to coding imperfection, the 2D views generated from the java 3D model in Figure 5 are not truly two dimensional as the piles in the back are still can be seen in the front, side and section views. Nevertheless, this is by no mean the limitation of java as more programming improvement will be able to overcome the imperfection.

The java 3D model has the ability display objects in shade, transparent and wireframe modes. Furthermore, volume and surface area of an object can be automatically calculated. Because the java 3D model is object-oriented, distinguishing the difference between the formwork and painting areas of an object is possible. Since cost and activity duration are by-products of quantity, cost estimate and project scheduling can be done automatically by java if all supporting data are available.

It is noted that transformation between the AutoCAD 3D solid model and the java 3D model is possible. One major obstacle, however, is that the two systems have discrepancy in their coordinate orientation. Furthermore, since the AutoCAD 3D solid model is not object-oriented, the real-world object type (e.g. beam, column) of each geometric object (e.g. polygon, box) exported from AutoCAD is unknown. To improve the transformation between the two systems, the real-world object type should be indicated in the AutoCAD 3D solid model by using specific color for each real-world object type to allow java to determine the real-world object type based on its color [8].

## 5. Conclusion

Java-based BIM employs an object-oriented approach, which offers a more logically realistic model than the traditional non-object 3D model represented by simple geometric objects. Java-based BIM provides great tools for 3D rendering and is flexible to work on any operating platform. In addition, java-based BIM offers great potential for future extension by incorporating with GIS or an enterprise project management system. Integrating java-based BIM with spatial database is a promising alternative for future BIM application development since spatial database could provide significant improvement for BIM data modeling and management. Additionally, spatial database is equipped with a number of spatial queries that can be useful tools for quantity takeoff, verification of building design conflicts and evaluation of topological relationship between building elements. Not only the proposed prototype system offers technological advantages over the traditional BIM applications, it is also an open-source solution that is free to acquire, customize and distribute, which can overcome the financial constraint that inhibits the implementation of BIM among small and medium enterprises. Nevertheless, in order to fully exploit the proposed prototype system, multiple implementation issues are still required further research including the programming technique to overcome insurmountable data processing time when the model contains a large number of reinforcing steels.

**Acknowledgement:** The authors gratefully acknowledge supports from the Thailand Research Fund.

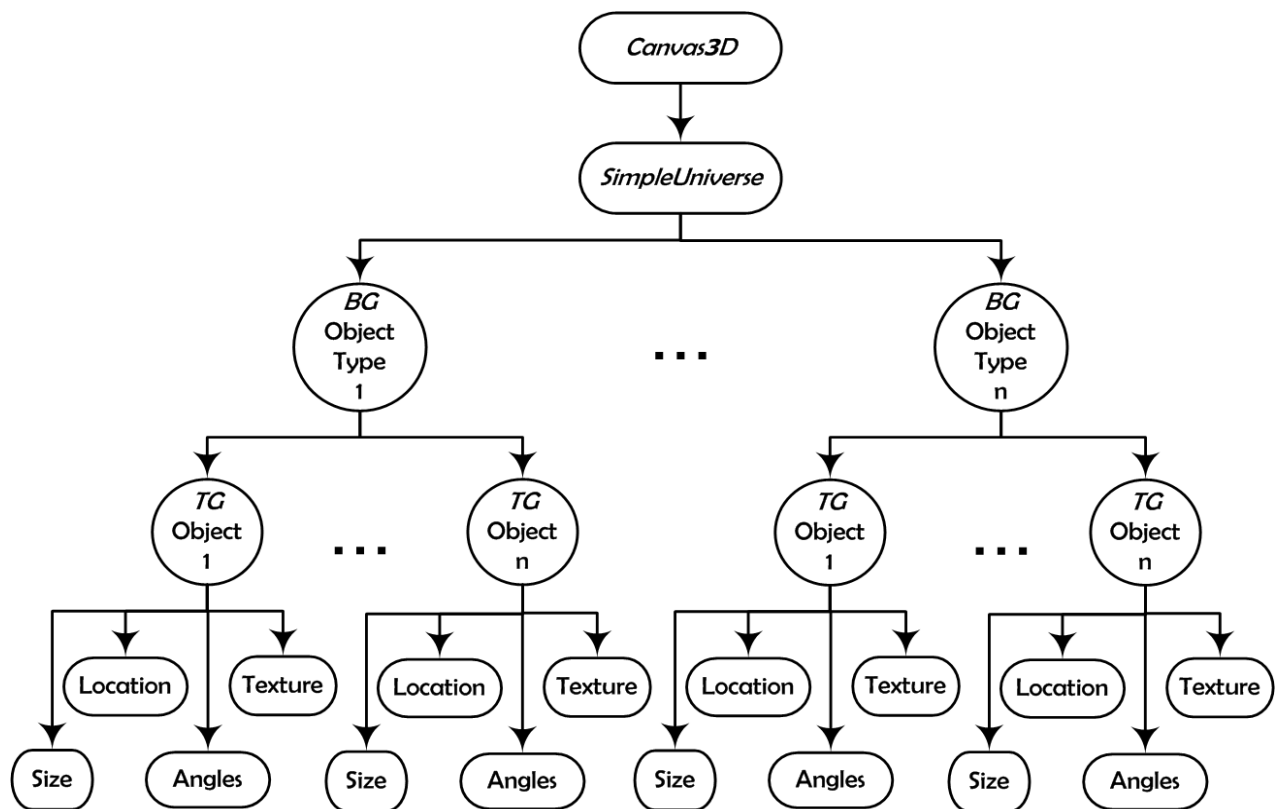


Fig. 4. 3D Model Generation in Java Environment

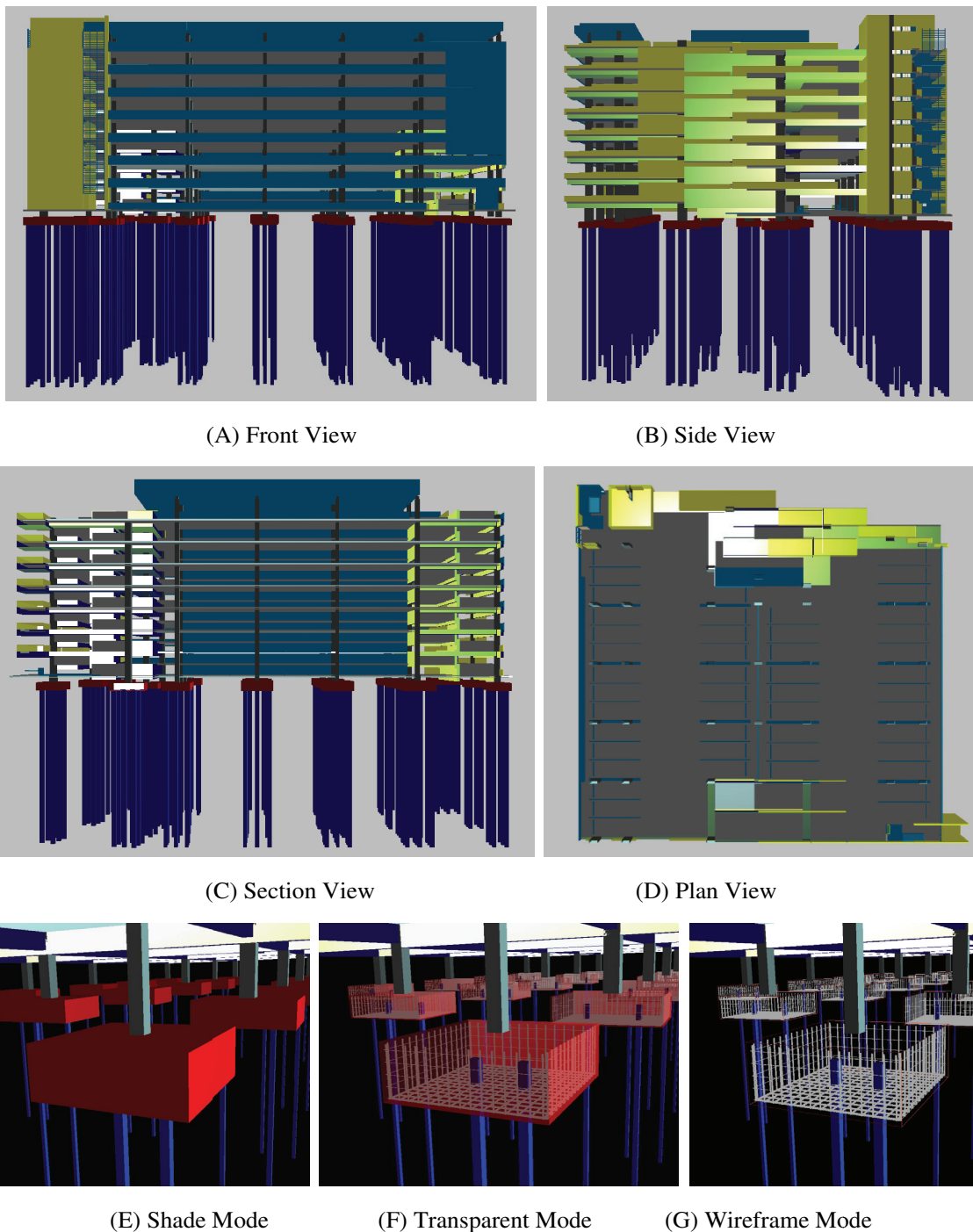


Fig. 4. 3D Model Generation in Java Environment

## References

- [1] Khemlani, L. "Should We BIM? Pushing the State of the Art in AEC", Cadalyst, USA, Volume 6, 2003
- [2] Khemlani, L., "The Eureka Tower: A Case Study of Advanced BIM Implementation", AECbytes, USA, Volume 6, 2004
- [3] Sun Microsystems. "Getting Started with the Java 3D API", Tutorial version 1.6.2, Sun Microsystems Inc., California, USA, 2001
- [4] Twilleager, D.; "Introduction to Graphics Programming with Java 3D", Available source: <http://www.euclideanspace.com>, 2006
- [5] Weber, A., Heinimann, A. and Messerli, P., "Use of NASA World Wind Java SDK for Three-Dimensional Accessibility
- [6] Visualization of Remote Areas in Lao P.D.R.", 6th ICA Mountain Cartography Workshop, Lenk, Switzerland, 2008
- [7] Refractions Research, "PostGIS 1.4.0 Manual", Available source: <http://postgis.refractions.net/>, 2009
- [8] Sangkaew, W., Malaikrisanachalee, S. and Vathananukit, H., "Java-based Three-Dimensional Object-Oriented Model for Building Information Modeling.", In Proceedings, 13th National Convention on Civil Engineering, Choburi, Thailand, 2008
- [9] Srisomboon, W. and Malaikrisanachalee, S. "Four-Dimensional Object-Oriented Construction Project Planning.", In Proceedings, 13th National Convention on Civil Engineering, Choburi, Thailand, 2008