Korean Journal of Remote Sensing, Vol.24, No.3, 2008, pp.245~250

A Prototype Implementation for 3D Feature Visualization on Cell Phone using M3G API

Kiwon Lee† and Woo Cheol Dong

Dept. of Information System Engineering, Hansung University

Abstract : According to public and industrial interests on mobile graphics, a preliminary implementation regarding 3D feature visualization system on cell phone was performed using M3G API, one of the de-facto standards for mobile 3D graphic API. Through this experiment, it is revealed that scene graph structure and 3D mobile file format supported from this API is useful one for 3D geo-modeling and rendering in mobile environment. It is necessary that 3D mobile graphic standards can be considered as one component of current mobile GIS services standards to provide value-added 3D GIS contents.

Key Words : Cell phone, mobile 3D, J2ME, M3G API.

1. Introduction

In these days, mobile 3D graphics is regarded as one of the emerging field in GIS field. However, most mobile services are being provided as 2D or 2.5D (Green, 2005). Of course, mobile 3D is the different history, compared with web 3D graphics in the industrial stage. These two fields link to each other in the up-to-the-date domain of Mobile-3D-Web, but we deal with solely mobile 3D area in this work.

There are several international standards suitable for mobile 3D graphics. Although these are still under development and advancement stage, they provide the application programming interfaces (APIs) for a wide range of mobile 3D applications. Among many freely available 3D graphic APIs, three standard APIs are widely adopted for the purpose of mobile 3D application development: Open GL ES (Embedded System), M3G (Mobile 3D Graphics) for Java ME (Micro Edition), and Direct 3D mobile.

Open GL ES released by Khoros Group is widely used for a range of mobile 3D applications in the entertainment or the cross-platform domain-specific field, and several mobile 3D visualization systems by actual implementation have been reported (Santa *et al.*, 2004; Etz, 2005; Lee and Kim, 2006; Kim and Lee; 2006) and are still under developing stage or industrial uses in the various application areas (Nadalutti *et al.*, 2006; Nurminen; 2006). However, additional graphic functions which are not contained in them need to devise. First, this API does not include 3D data structure. Second, utility toolkit for user interfaces varies on mobile environment, on the

Received June 10, 2008; Revised June 19, 2008; Accepted June 20, 2008.

† Corresponding Author: Kiwon Lee (kilee@hansung.ac.kr)

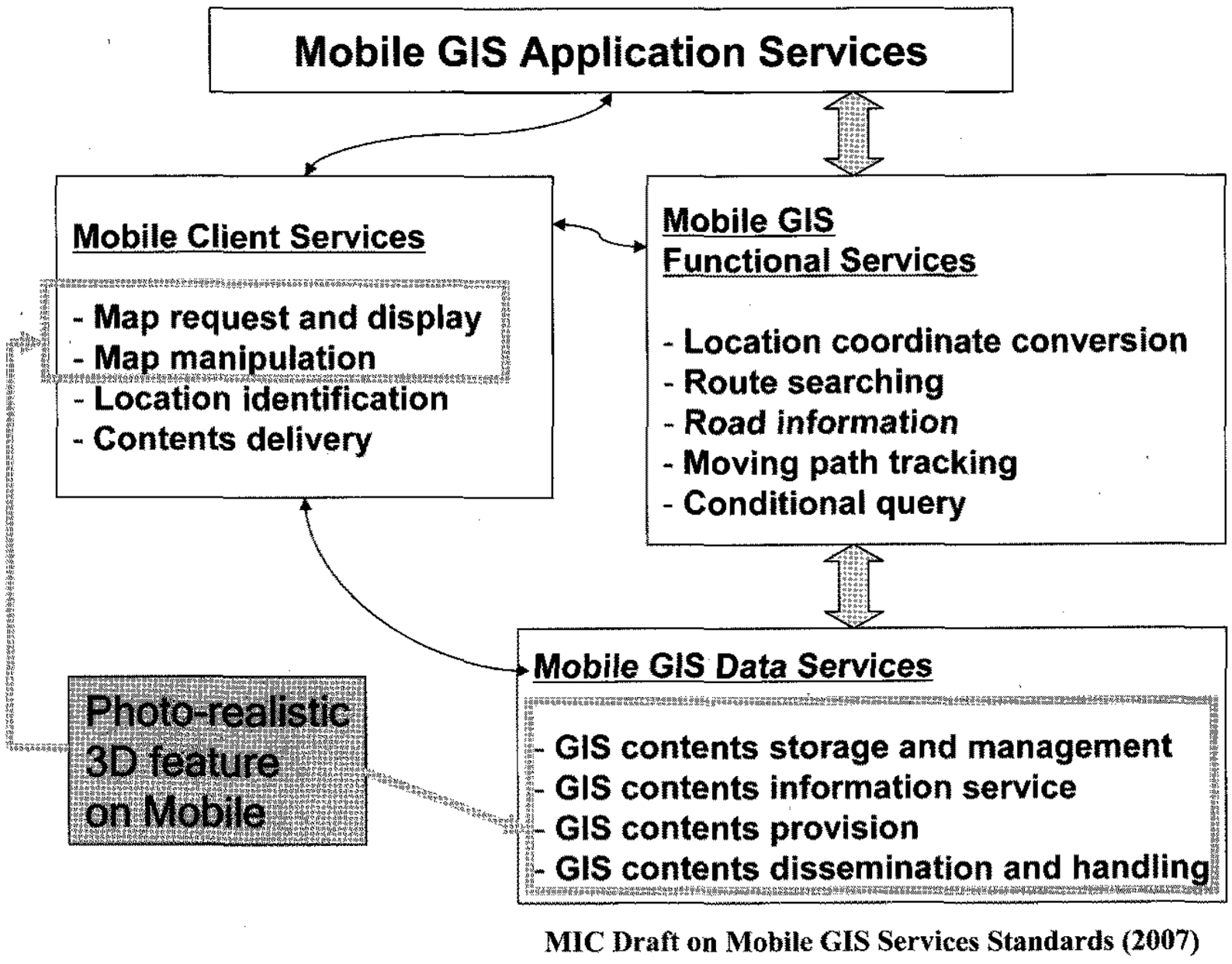


Fig. 1. Functional parts in 3D feature visualization in mobile GIS service standards draft (MIC, 2007).

contrary to standard OpenGL glut functions. Third, this supports the procedural graphic pipeline and pixel pipeline so that it affects run-time performance on function-callback execution. Whereas, Direct 3D mobile shows advantageous aspects on those pitfalls in OpenGL ES, but it is dependent on MS Window mobile OS. Among them, M3G API is used and tested for the further 3D geo-spatial processing on mobile environment, especially cell phone, in this study. Comparison of these APIs is beyond the main scope of this study. In some extents, these can be applied with the consideration to implementation and operation environments to fit a given target system. This does not imply which one is better than others.

As for the other motivation in this study, the nation or international standards of mobile GIS services do not consider the mobile 3D visualization, but 2D map display on mobile environment (Fig. 1). Thus, it is necessary to demonstrate the possibilities of mobile 3D graphics approach.

2. M3G Technical Briefing

M3G API is known to JSR (Java Specification Request) 184. This API provides 3D functionality in a compact package for CLDC/MIDP devices. The API provides two methods for displaying 3D graphics content. The immediate mode API makes it possible for applications to directly create and manipulate 3D elements. Layered on top of this is a scene graph API, also called retained mode, that makes it possible to load and display entire 3D scenes that are designed ahead of time. Applications are free to use whichever approach is most appropriate or to use a combination of the retained mode and immediate mode APIs. The JSR 184 specification also defines a file format for scene graphs (Pulli, 2004).

Unlike to Java 3D graphic API, this is for a scalable, small-footprint, and interactive API for mobile devices such as PDA or cell phone, working with as optional package for J2ME (Java 2 platform,

Micro Edition), CLDC (Connected Limited Device Configuration), and MIDP (Mobile Information Device Profile). CLDC and MIDP are a specification of a framework for J2ME applications running on cell phones and a specification to define the minimum hardware, software, and network requirements for an application to run on an embedded device, respectively. While, M3G as lightweight API is a complement to the OpenGL ES API, so that it is designed to be compatible with OpenGL ES API, low-level standard API. It regarded as an immediate mode. At tutorial in SIGGRAPH 2005, some principles for M3G design was presented as follows: No java code along critical paths, Cater for both software and hardware, Maximize developer productivity, Minimize engine complexity, Minimize fragmentation, and Plan for future expansion. These emphasized on difference with Java 3D, OpenGL, OpenGL ES, or Direct 3D.

By means of a retained mode, M3G supports for a scene graph as a tree structure to represent in a compact and hierarchical one with respect to all the elements of a 3D scene, as the rendered result. The key classes in M3G are World, Graphics3D, and Loader. World is a scene graph root node, and Graphics3D is for 3D graphics rendering context containing global state: frame buffer, depth buffer, viewpoint, hints. Loader can load individual objects and entire scene graphs in a file format with extension of m3g (Pulli *et al.*, 2005, 2008).

This file includes all data related to the given scene graph. Fig. 2 represents an example of scene graph, and each class in this scene graph is explained for the summary (Table 1). This m3g format for 3D graph model with all attributes for rendering can be imported or exported any other commercial 3D graphic software tools. The m3g file format is

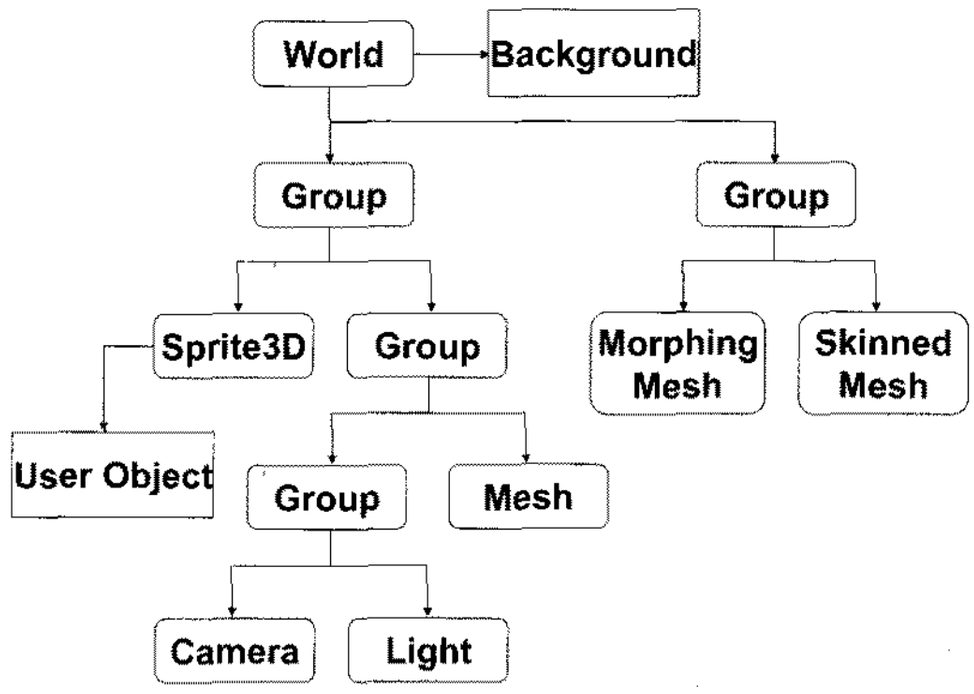


Fig. 2. Basic level of a scene graph in M3G, excerpted from Malizia (2006).

basically a file structure by one-to-one mapping between object type and java classes, so that it is possible that the importing and exporting into other generic 3D graphic tools or browsers with decoding and encoding functions of it.

Fig. 3 shows m3g file structure composing with file identifier of 12 bytes and several sections. Section 0 with header object should be contained, and section 1 is for external reference. Actual attributes and geometric information to be rendered are defined in other sections.

This is one of the main advantageous aspects for mobile applications using M3G API, and regarded as a distinguished feature compared with OpenGL ES implementation. In many cases, there is little else to an application than displaying a scene and playing back some animation created in a 3D modeling tool. Even in more demanding cases, it greatly speeds up development if it is easy to import objects and animations into MIDlets which means MIDP applications. Therefore, the API must provide importer functions for different data types, such as textures, meshes, animations, and scene hierarchies. The data must be encoded in a binary format for compact storage and transmission.

Table 1. The main classes in M3G, related to Fig. 2. Partly cited from Höfele (2007)

Class	Description
Background	Defines a background image or color for the renderer
Camera	A node that defines a camera. It is preprogrammed with two projection matrices: parallel and perspective.
Graphics3D	The main render class for M3G. Capable of retained or immediate rendering.
Group	A scene graph node capable of storing other nodes.
Light	A scene graph node that can define ambient, directional, omni-directional, and spotlights.
Loader	Imports a .m3g file.
Material	Subcomponent for Appearance that defines the material for an object.
Mesh	Defines a polygonal mesh.
Morphing Mesh	Manipulates meshes that have morph targets.
Node	An abstract class shared by all nodes.
Object3D	An abstract class from which all nodes capable of existing in the 3D world are derived.
Skinned Mesh	A scene graph node that represents a mesh with skeletal animation.
Sprite3D	A scene graph node used to define 2D billboards.
Texture2D	A subcomponent of Texture2D used to map Image2D objects to Sprite3D and Mesh3D.
Transform	A generic class for manipulating transformations.
Transformable	An abstract class that gives Sprite3D and Node the capability to store and manipulate transformations.
World	A top-level node for scene graphs.

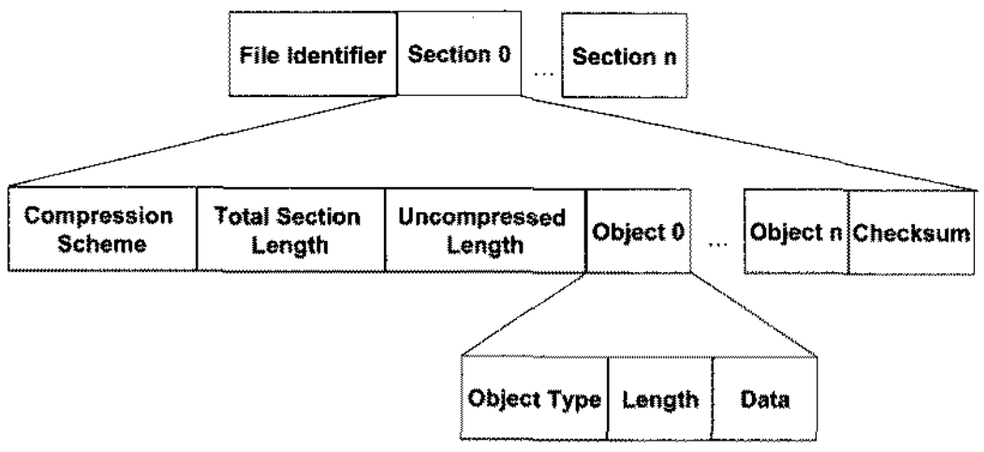


Fig. 3. M3G file structure defined in JSR 184.

3. Implementation Result

In this study, JDK 1.5.0_15, Netbean IDE 6.01 mobility pack, and WTK 2.52 (CLDC1.1/MIDP2.0) were applied for the implementation processes on Windows XP. As for 3D model in testing stage, a simple 3D feature model (Fig. 4) was used in the style of 3D objects, similar to Pipho and LaMonthe (2003). This model is for a single 3D feature with ID in header, and each face in x-y-z modeling coordinate system possesses texture image registered in u-v coordinate system. This model can be replicated with other coordinate values and texture image through scene graph scheme. Besides 3D features, background and base images were also processed as texture images, and these can be changed according to users' selection. While, the relationship and association among node, object3D, and transformable class, as class instances used in this implementation process, is shown in Fig. 5. Object3D class is a base class in 3D scene construction related to scene graph manipulation. Transformable class and node class are used for positioning, rotating, and scaling of 3D objects rendered.

Fig. 6 represents some rendered scenes in the emulator environment. While, the camera object in the control box composed of 4 direction arrows helps for 3D model transformation such as translation and rotation.

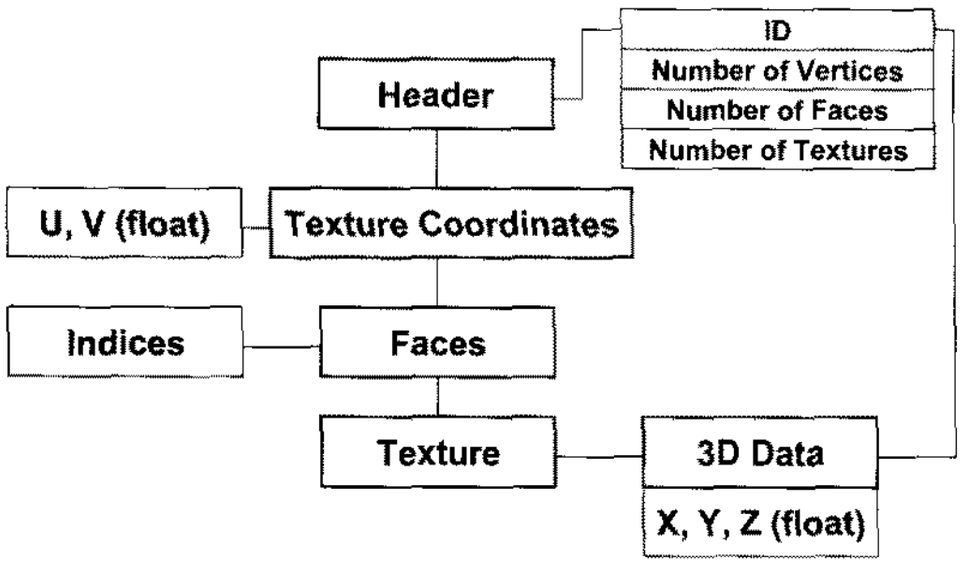
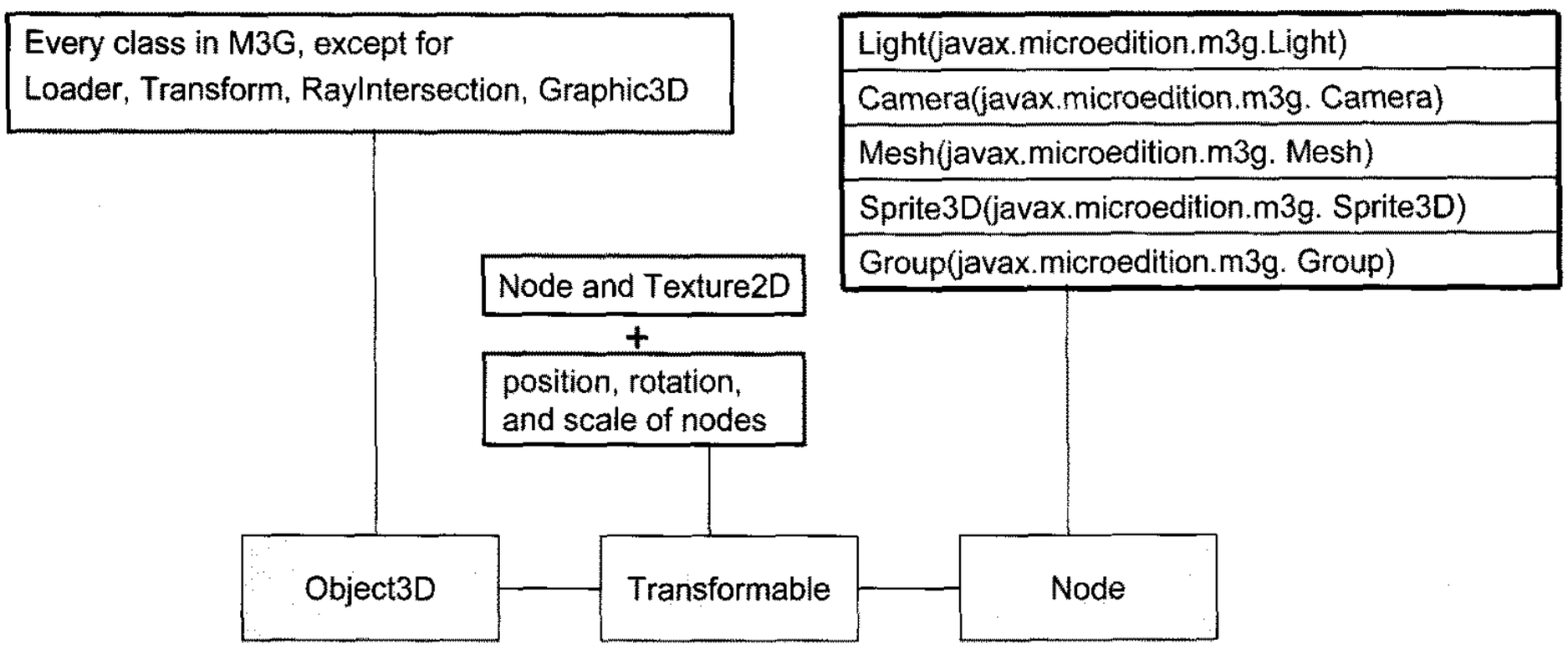


Fig. 4. A 3D feature model applied in this implementation.

4. Concluding Remarks

A preliminary implementation regarding 3D feature visualization on cell phone was performed with the purpose of the mobile application of the further 3D geo-processing techniques using M3G API. Practically, 3D feature models with texture, based on scene graph structure in retained mode, are



Relationship of Node, Object3D, and Transformable classes

Fig. 5. Relationship of node, object3D, and transformable classes, excerpted from Morales and Nelson (2007).

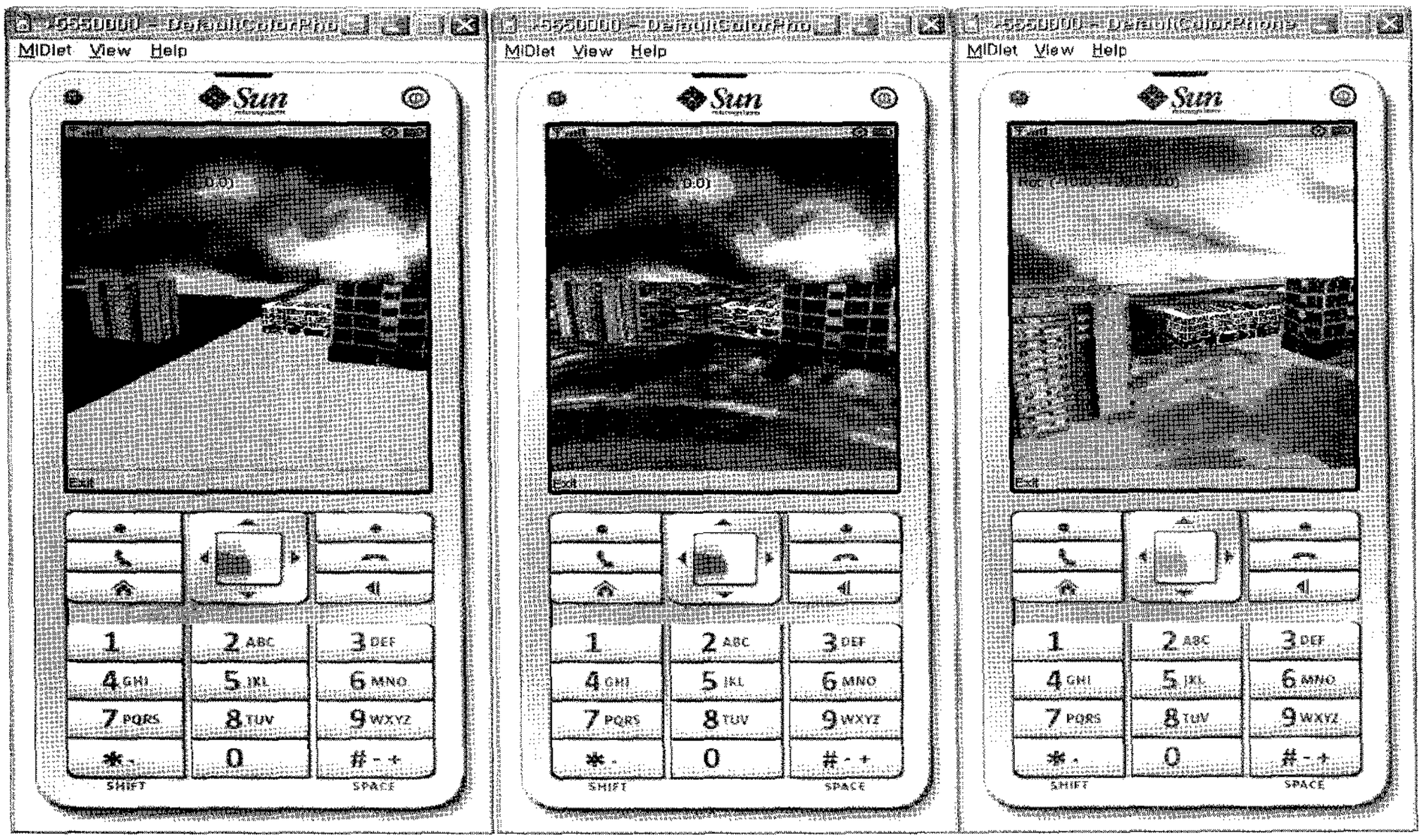


Fig. 6. Implementation results with the background and base texture images, testing in cell phone emulator.

constructed. This implementation scheme using M3G API shows some advantageous aspects. First, the performance for object-based 3D visualization and additional transformation process is on acceptable level to general users' sides. Second, 3D model in scene graph model can be directly imported and exported to other 3D modeling and rendering system in the form of m3g file format, unlike OpenGL ES API. Third, the realistic 3D scene generation in cell phone can flourish value-added contents for mobile GIS services. Fourth, the possibility for 3D processing standards specification for mobile GIS services can be considered in the next stage of standard development.

Acknowledgements

This research was financially supported by Hansung University in the year of 2008.

References

Etz, M. and J. Haist, 2005. Mobile 3D Viewer, *CGTopics*, 4, 28p.

Green, D. R., 2005. *Going Mobile: Mobile Technologies and GIS*, URISA, 60p.

Höfele, C. (Ed), 2007. *Mobile 3D Graphics: Learning 3D Graphics with the Java Micro Edition*, 305p, 432p.

Kim, S. Y. and K. Lee, 2006. Development of Mobile 3D Terrain Viewer with Texture Mapping of Satellite Images, *Korean Journal of Remote Sensing*, 22(5): 351-356.

Lee, K. and S.-Y. Kim, 2006. Development of Mobile 3D Urban Landscape Authoring and Rendering System, *Korean Journal of Remote Sensing*, 22(3): 221-228.

Malizia, A., 2006. *Mobile 3D Graphics*, Springer, 155p.

Morales, C. and D. Nelson, 2007. *Mobile 3D Game Development: From Start to Market*, Clarles River Media.

Nadalutti, D., L. Chittaro, and F. Buttussi, 2006. Rendering of X3D Content on Mobile Devices with OpenGL ES, *Proceedings of Web3D, 2006*: 19-26.

Nurminen, A., 2006. m-LOMA - a Mobile 3D City Map, *Proceedings of Web3D 2006*: 7-18.

Pulli, K., 2004. The rise of mobile graphics, *Nokia Information Quarterly*, 3: 14-15.

Pulli, K., T. Aarnio, K. Roimela, and J. Vaarala, 2005. Designing Graphics Programming Interfaces for Mobile Devices, *IEEE Computer Graphics and Applications*, 25(8).

Pulli, K., T. Aarnio, K. Roimela, and J. Vaarala, 2008. *Mobile3D Graphics with OpenGL ES and M3G*, Elsevier, pp. 120-129, 314

Pipho, E and A. LaMonthe, 2003. *Focus on 3D Model*, The Premier Press Game Development Series, 200p.

Sanna, A., C. Zunino, and F. Lamberti, 2004. A distributed architecture for searching, retrieving and visualizing complex 3D models on Personal Digital Assistants, *Int. Jour. Human-Computer Studies*, 60: 701-716.