International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.9 (2015), pp.189-198
http://dx.doi.org/10.14257/ijmue.2015.10.9.20

Lets3D: A Collaborative 3D Editing Tool Based On Cloud Storage

Yeoun-Ui Ha[1], Jae-Hwan Jin[1] and Myung-Joon Lee[2*]

Department of Electrical/Electronic and Computer Engineering,
University of Ulsan, 93, Daehak-ro, Nam-gu,
Ulsan 680-749, Republic of Korea
[1]{gkdus23, jjhok2000}@gmail.com, [2]mjlee@ulsan.ac.kr*

Abstract

3D Modeling is a process of developing 3D objects and their features, having gained popularity during the last decade. As of now, there are many 3D editors to help 3D modeling such as Google's sketch up, AutoCad, sunglass and others. As these 3D editors have become more useful and 3D-related activities have become rich, collaboration in 3D modeling is getting much attention. In this paper, we present a new 3D editor named Lets3D. In editing and sharing 3D objects, Lets3D enables a group of users to collaborate in real time. To support these collaboration features, Lets3D relies on the XMPP messaging facility and the whiteboard service of C3ware, which is a middleware providing abstract editing operations and synchronization operations on shared objects over cloud storage. To run on Web browsers, Lets3D is developed as an extension of the Three.js editor which uses the Three.js lightweight JavaScript 3D library.

Keywords: *3D Editor, Collaborative 3D editing, C3ware, XMPP, Lets3D, Three.js*

1. Introduction

As of now there are many 3D editors such as Google's sketch up [1], AutoCad [2], sunglass [3] and others. A 3D Editor is a tool which enables users to edit 3D objects in a convenient way during the process of developing 3D objects and their features. Such editing tools internally use a kind of popular graphics APIs such as openGL [4], direct3d [5], and WebGL [6] to handle 3D objects systematically. Recently, to support interactive 3D graphics in Web browsers, xml3d [7] has been proposed as an extension to HTML5, which means 3D contents can be embedded directly into Web sites as HTML components. As 3D-related activity becomes richer and such useful 3D editors are being used more popularly, collaboration in 3D modeling is getting more attention. Unfortunately, the existing 3D collaborative environments for 3D modeling [3,8] are provided only as commercial Web services. Even the famous open source 3D editors like blender [9] and FreeCAD [10] do not support effective collaborative environment.

In this paper, we present a collaborative 3D editor named Lets3D, which is equipped with the functionality of collaborative real-time editing and sharing of 3D objects among a group of users. This paper is the extension of our previous work [11]. The group users share their editing screen through Web browsers, being guided by the knowledge of which objects are being edited by which users. Lets3D is developed as an extension of the Three.js editor [12] running on Web browsers. To render 3D objects, the editor uses the Three.js lightweight cross-browser library for WebGL. To provide the collaborative functionality, Lets3D utilizes the whiteboard service of C3ware [13]. C3ware is a middleware to support various types of collaborative workspaces over cloud storage, and the whiteboard service delivers abstract editing operations and synchronization operations on shared objects in collaborative workspaces. For creating and maintaining user/group

[*] Corresponding Author

information, Lets3D relies on the Openfire XMPP messaging system [14], which is also used to synchronize the editing screens among a group of users via message exchange.

2. Background

2.1. Three.js

Three.js is a lightweight open source JavaScript 3D library, through which animated interactive 3D graphics can be created and displayed on any Web browsers. WebGL is a cross-platform, royalty-free API based on the OpenGL 3D graphics technology across popular browsers. However programming WebGL directly from JavaScript to create and animate 3D scenes is a very complex and error-prone process [6]. So, many things that most WebGL applications need are abstracted by Three.js, helping developers to handle 3D objects more easily. Figure 1 shows some example figures created through Three.js.

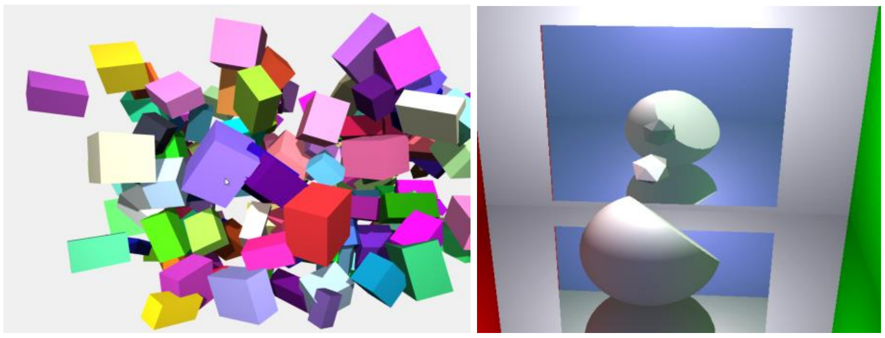

Figure 1. Figures Made by Three.js

Three.js has many features for the effective presentation of 3D scenes in various categories: renderers, scenes, lights, objects, geometry, loaders, and import/export. The library is usually used in web browsers for presenting 3D objects, often being utilized in implementing Web editors and Web games.

2.2. C3ware

C3ware is a collaborative middleware based on cloud storage, which supports various types of workspaces: personal, group, and open workspace. The collaboration patterns based on resource sharing among users are depicted in Figure 2 through C3ware's workspaces.

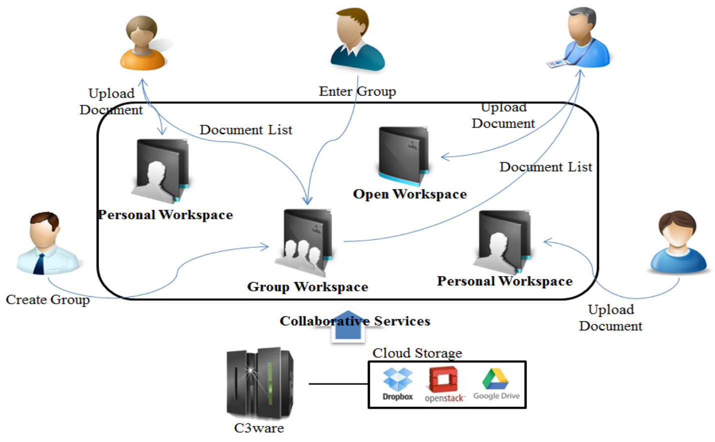

Figure 2. Collaborative Work through C3ware

To support the workspaces effectively, C3ware's service is composed of *user service*, *group service*, and *workspace service*. Besides, as a higher level collaborative service, the *whiteboard* service is provided to support a shared whiteboard among a group of users. As shown in Table 2, *the whiteboard* service manages the data and the operations on whiteboard rooms and graphical objects in the rooms. In addition, the whiteboard service provides a concurrency control mechanism to handle the *modify*/*delete* operations for shared objects with a locking functionality. Before the modify/delete operation for an object in a shared whiteboard is called, the *getPermission* request (to get the permission of editing the object) should be announced.

Table 2. Whiteboard Service in C3ware

Types	Whiteboard services
WhiteboardRoom management	createWhiteboardRoom, deleteWhiteboardRoom, getRoomList.
Object management	createObject, deleteObject, modifyObject, getObjectList, undo, redo.
Concurrency control	getPermission.

2.3. Strophe.js

Strophe.js is an open source JavaScript XMPP client library. It provides APIs for developing XMPP client applications, supporting applications running on Web browsers to use XMPP messaging. There are various types of plugins supported by strophe.js. Some of those plugins are :

- *MUC* plugin (using XEP-0045) supports functions related to multi-chat room like creating chat room, joining and leaving chat room;
- *Register* plugin (using XEP-0077) helps user to register and login into server;
- *Roster* plugin (using XEP-0237) provides methods of getting and managing roster from server ;
- *PubSub* plugin (using XEP-0060) supports functions like publishing/scribing handler.

3. Design Requirements for 3D Collaborative Editing

In this section, we describe the structure of Three.js 3D editor, and discuss the design requirements for supporting collaborative features over the editor.

3.1. Three.js Editor

The Three.js is an open source 3D editor using the Three.js JavaScript library. The editor runs on Web browsers, implemented with HTML5 and JavaScript. As Figure 3 shows, the screen of the editor is composed of four parts: *Menubar*; *Viewport*; *Sidebar*; *Toolbar*.

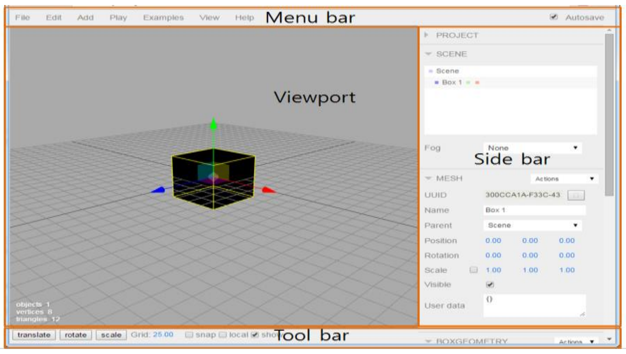

Figure 1. 4 Parts of 3D Editor

In the editor, there are 5 major classes: *Editor*, *Menubar*, *Viewport*, *Sidebar*, and *Toolbar*. The functionality of each class is presented in Table 3.

Table 3. Major Classes of Three.js Editor

Classes	Description
Editor.js	The class manages all of data and processes in Lets3D on 3d objects and edit operations
Menubar.js	The class manages functions for menu bar.
Viewport.js	The class manages 3d objects at viewport, being responsible for displaying 3D objects.
Sidebar.js	This class manages modification of information on 3d objects
Toolbar.js	This class selects type of move operation.

To process operations such as *create object*, *delete object* and *modify object*, The editor uses methods for generating and handling signals. The Editor class is responsible for handling those signals. The other classes generate signals for the designated operations, calling the Editor's methods. In this process, to define and manage signals, the editor uses the signal.js JavaScript library which creates the structure for handling the signals. For example, when an object is created through the menu bar, the *objectadded* signal is generated. If this signal is generated, the handler function registered in the signal is executed. Figure 4 shows the activities for the generation of a signal and the creation of an object.

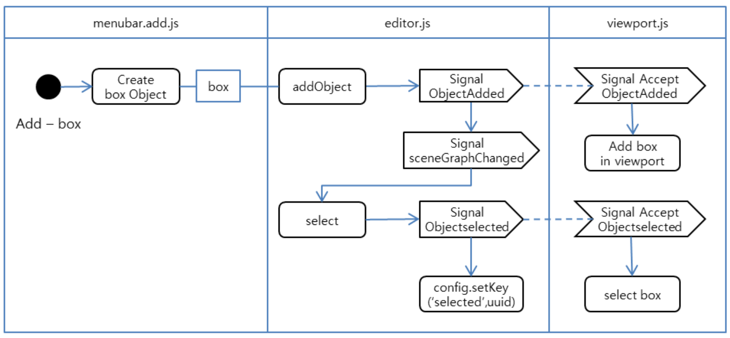

Figure 4. Class Diagram when an Object Added

A single operation can cause several signals, which activate the registered signal handlers. As shown in Figure 4, when an object is created, the *objectadded* signal, the *scenegraphchanged* signal (signal to add the object information in the sidebar) and the *objectselected* signal (signal for selecting the added object) are generated.

3.2. Requirement

In addition to the core editing functionality of usual 3D editors, to enable a group of users to edit shared 3D objects in real time, collaborative 3D editing introduces the following requirements:

1) Managing operation histories on shared objects

Since a group of users asynchronously generate operations on objects from different places, the collaborative environment should provide the mechanism to manage operation histories on shared objects, supporting redo/undo operation both in a local standpoint and in a global standpoint.

2) Concurrency control

Asynchronous modifications from several users on the shared 3D objects are frequently generated in a 3D collaborative editing environment. So, to ensure the consistency of the objects, an appropriate concurrency control mechanism on the objects should be systematically provided.

3) Reliable storage resources

Resources created in a collaborative 3D editing environment such as 3D objects, operation history, and information on users participating in a collaborative work should be stored in a reliable storage and should be reliably accessed from geographically dispersed users.

4. Lets3D

In this section, we describe the system structure of Lets3D and the techniques used for implementing Lets3D. Basically, Lets3D is extended from the Three.js editor with the functionalities required for collaborative 3D editing environment as discussed in the previous section.

4.1. System Structure

Since Lets3D is an extension of Three.js editor, it is written in JavaScript and runs on Web browsers. To provide collaboration features for shared objects, Lets3D utilizes various services of the C3ware collaborative middleware, while adopting the Openfire XMPP messaging server to synchronize the editing screens among a group of users via message exchange. The user and group information is created and maintained through Openfire, and being shared with C3ware. Figure 5 shows the system structure of Lets3D.

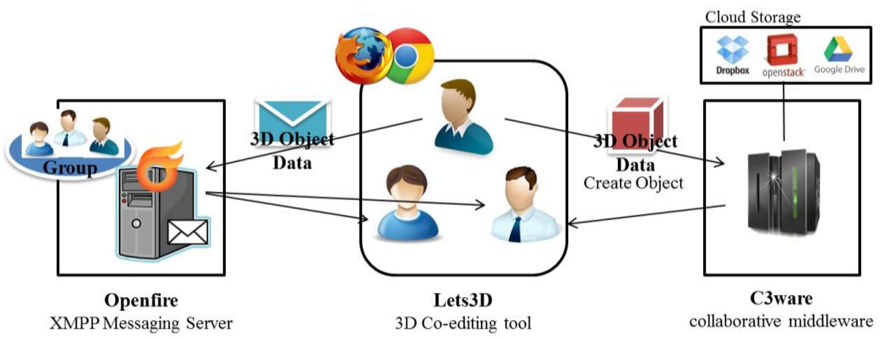

Figure 5. System Structure of Lets3D

4.2. Sharing 3D Objects and Concurrency Control

To support sharing 3D objects and real-time collaborative editing, Lets3D performs the following steps for an editing operation.

1) A member in the group requests an operation on an object.
2) The operation's result is reflected in the screen of the member.
3) The information on the result is sent to the C3ware for update to the collaborative workspace provided by C3ware in association with the group.
4) If the service of C3ware succeeds, the member disseminates the related group message to the other members to synchronize the editing screen of each member in real time.

In this way, the collaborative workspace of C3ware associated with a collaborative editing group stores the data shared among the editing group of Lets3D. The following Table 4 shows the relation of the user operations provided by Lets3D and the corresponding Web services of C3ware.

Table 4. Web Service to be used in User Operation

User operation	Web service
Create Object	CreateObject
Delete Object	DeleteObject
Modify Object	GetPermission
	ModifyObject
Initial Object Screen	GetObjectList
Cancel operation	Undo
Revert Operation Canceled	Redo

As described earlier, asynchronous modifications from a group of users on the shared 3D objects can be requested in Let3D. So, to ensure the consistency of the objects, the modify operation of Lets3D requests the *getPermission* service on the related object to C3ware before requesting the modify operation to C3ware. C3ware uses a locking mechanism to control the synchronization of these asynchronous requests on shared objects. The *getPermission* Web service tries to obtain the lock on the related object. If the object is previously locked by a request from another user, the request is denied. Otherwise, the user obtains the lock on the object and performs the modify operation. After the modify operation is completed, the lock on the related object is released by C3ware. Figure 6 shows this procedure in a graphical way.

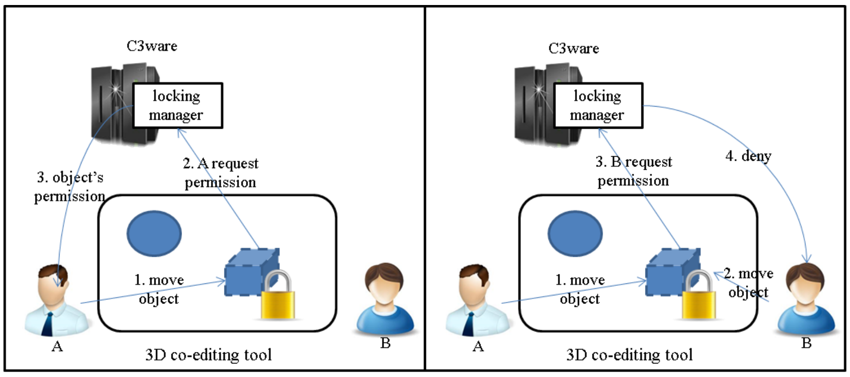

Figure 6. Synchronization through *getPermission*

4.3. Synchronization of Lets3D Editing Screens

To immediately synchronize the editing screens of a group of users, Lets3D relies on the Openfire XMPP server. The Openfire server provides the group management facility for each collaborative editing group of Lets3D. Lets3D opens a group messaging service by creating a chat room for a collaborative editing group. The activities of each Lets3D's user are transmitted to the group members via a group message. Each member who receives the group message synchronizes the editing by reflecting the operation contents held in the message as shown in Figure 7. To handle these XMPP messages in Web browsers, Lets3D uses the Strophe.js Library that can be plugged into Web Browsers for developing XMPP client applications.

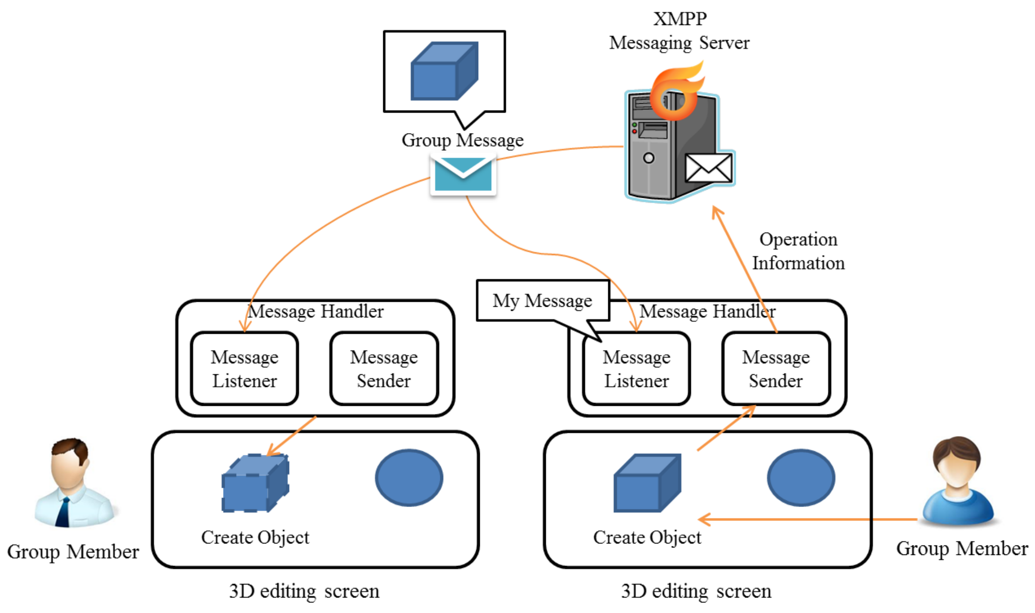

Figure 7. Synchronization between Group Users

4.4. 3D Collaborative Editing Feature

As is the case with the Three.js editor, the screen of Lets3D is composed of four parts: Menubar; Viewport; Sidebar; Toolbar. Unlike the Three.js editor, Lets3D has the *Group* menu on the Menubar. The Group menu provides the login screen and the group management screen. A user can be authenticated by entering his or her user name and password in the login screen. Finishing the authentication, the user can select a group in their group list to perform 3D collaborative editing, or create a new group. Also, the user who creates a group can invite other users into the group by using the invite function. The figure 10 shows the login screen and group management screen of Lets3D.

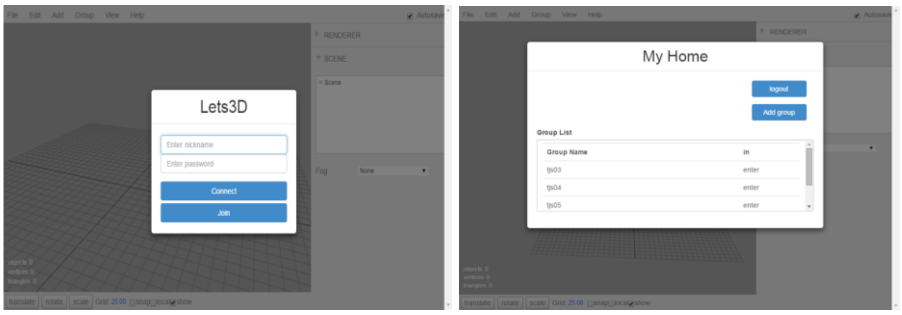

Figure 8. Login and Group Management Screen of Lets3D

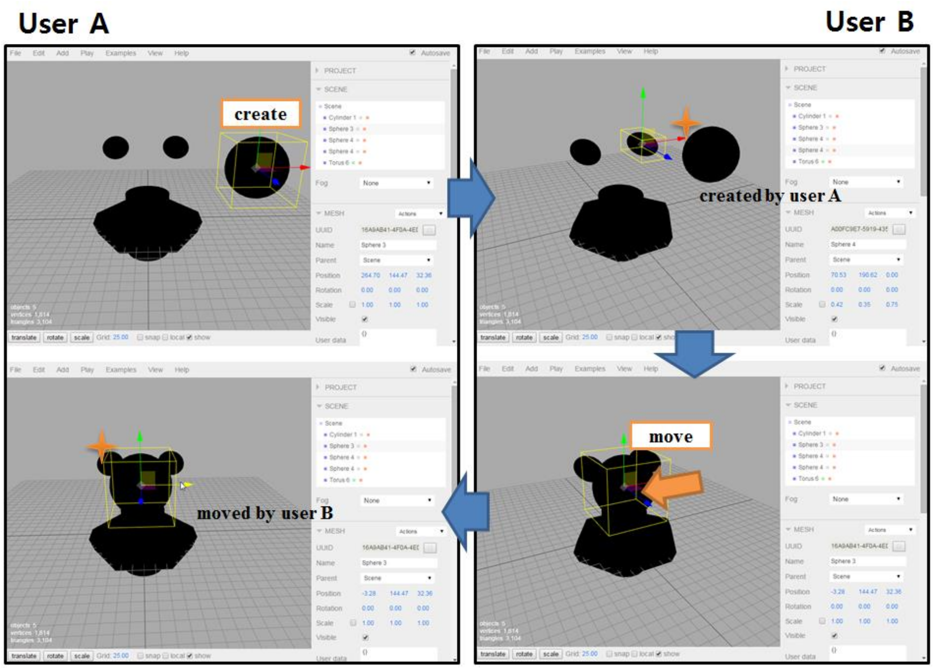

Figure 9. Collaborative Editing through Lets3D

Usually, users perform the following procedures for collaborative 3D editing through Lets3D.

1. A user makes a registration to the Openfire XMPP server through the login and join screen of Lets3D as shown in Figure 8.
2. The user creates a group, inviting other users to the group at the group management screen of Lets3D.
3. As Figure 9 shows, the group members perform editing operations on 3D objects, sharing the same editing screen. With the help of the concurrency control service of C3ware, Lets3D synchronizes the editing screens of the group users through XMPP messaging.
4. Each 3D object created or modified by a group user is stored in the related group workspace over cloud storage.

5. Comparison with other Application

In this section, we compare Lets3D with other popular 3d editing environments such as Blender, Clara and Verold. Blender is a famous free and open source 3D creation environment. It supports the entire 3D pipeline—modeling, simulation, rendering, compositing and motion tracking and so on. Blender is implemented in C/C++ and provides Python API for customizing Blender applications. Unfortunately, Blender supplies no collaborative features yet. Clara and Verold provide collaborative 3D editing environment as Lets3D does, but these environments are commercial services through registration to the related Web sites. As Table 5 summarizes, Lets3D supports real-time collaboration features as Clara and Verold do. But Lets3D is extended from the open source Three.js editor, developed as a free 3D collaborative editing tool. In addition, Lets3D is based on the middleware using cloud storage and support XMPP messaging among users.

Table 5. Comparison Lets3D with Other Application

	blender	**clara**	**verold**	**Lets3D**
Platform	Window & Linux	Web	Web	Web
Development Language	Python/ C/C++	JavaScript	JavaScript	JavaScript
Commercial	-	✓	-	-
Real-time collaborative editing	-	✓	-	✓
Concurrency control	-	✓	-	✓
Open source	✓	-	-	✓
Group management	-	-	-	✓

(- : unsupported, ✓ : supported)

6. Conclusion

In this paper, we described the development of a new collaborative 3D editor runs on Web browsers. To support collaborative features, Let3D utilizes the whiteboard service of C3ware. Through the whiteboard service, Lets3D stores 3D objects into collaborative workspaces provided by C3ware and performs synchronized edit operations on those shared 3D objects. Lets3D inherits the core editing functionality from the open source Three.js editor based on the Three.js WebGL JavaScript library, which ascertains the simple and easy user interface of Lets3D. The ease in editing and the novelty for collaboration of Lets3D would make the tool very useful especially for non-professional users who want to handle 3D objects in their usual life.

Acknowledgements

This research was partially supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (No. 2013R1A1A4A01004459).

References

[1] http://www.sketchup.com
[2] A. Yarwood and B. S. Palm, “Introduction to AutoCAD 2016: 2D and 3D Design”, Routledge, **(2015)**.
[3] http://sunglass.io
[4] https://www.opengl.org/documentation
[5] https://msdn.microsoft.com/en-us/library/windows/desktop/hh309466(v= vs.85).aspx
[6] http://www.webgl-publisher.com/TechInfoEn.html
[7] http://xml3d.org

[8] https://clara.io
[9] https://www.blender.org
[10] http://www.freecadweb.org/wiki/index.php?title=Main_Page
[11] Y. U. Ha, J. H. Jin and M. J. Lee, "Supporting Collaborative 3D Editing over Cloud Storage", In Proceedings of the 7th International Interdisciplinary Workshop Series", Jeju Island, Korea, August 19-22 **(2015)**.
[12] J. Dirksen, "Learning Three. js–the JavaScript 3D Library for WebGL", Packt Publishing Ltd, **(2015)**.
[13] H. C. Lee, J. E. Park and M. J. Lee. "C3ware: A Middleware Supporting Collaborative Services over Cloud Storage", The Computer Journal: bxs168, **(2013)**.
[14] P. Saint-Andre, "Extensible messaging and presence protocol (XMPP): Core", **(2011)**.

Authors

Yeoun-Ui Ha, is an M.S student in the School of Electrical Engineering at the University of Ulsan in Korea. She received her BS in School of Computer Engineering and Information Technology at the University of Ulsan in Korea. Her research interests include Collaborative System and middleware.

Jae-Hwan Jin, is an M.S. student in the School of Electrical Engineering at the University of Ulsan in Korea. He received his B.S. from the University of Ulsan in Korea. His research interests include Cloud Storage Service, Collaborative System, and Messaging System.

Myung-Joon Lee, is a professor in the School of Electrical Engineering at the University of Ulsan in Korea. He received his Ph.D. from the KAIST (Korea Advanced Institute of Science and Technology) in Korea. He has (co-)authored more than 200 research publications including numerous works on collaborative system, distributed system and biological system.