# A Hierarchical 3D Data Rendering System Synchronized with HTML

Yousuke Kimura, Tomohiro Mashita, Atsushi Nakazawa, Takashi Machida, Kiyoshi Kiyokawa and Haruo Takemura

*Abstract*— **We propose a new rendering system for large-scale, 3D geometic data that can be used with web-based content man-agement systems (CMS). To achieve this, we employed a geometry hierarchical encoding method "QSplat" and implemented this in a Java and JOGL (Java bindings of OpenGL) environment. Users can view large-scale geometric data using conventional HTML browsers with a non-powerful CPU and low-speed networks. Further, this system is independent of the platforms. We add new functionalities so that users can easily understand the geometric data: Annotations and HTML Synchronization. Users can see the geometric data with the associated annotations that describe the names or the detailed explanations of the particular portions. The HTML Synchronization enables users to smoothly and interactively switch our rendering system and HTML contents. The experimental results show that our system performs an interactive frame rate even for a large-scale data whereas other systems cannot render them.**

***Index Terms*** **—CMS, 3Ddata, Annotation, HTML synchro -nization**

## I. INTRODUCTION

The widely used World Wide Web provides many types of information including texts, sounds and 2D images for various purposes. These sorts of multimedia data are very helpful to enrich contents and enhance users' experiences.

Use of 3D geometric data is an effective method for understanding contents because users can browse them from arbitrary viewpoints. Our goal is to develop a 3D geometric data browser along with user interfaces for the purpose of web-based contents management systems (CMS) into World Wide Web. Our browser can render large-scale geometric data very quickly even on not so powerful user terminals. We designed this system to compensate for three issues.

First, this software should work independently of users' hardware and software environments. Previous solutions have needed additional plug-in software to render 3D geometric data, thus users had to install the software in advance. Additionally, this sorts of plug-in software can not always be ported to all platforms. To resolve this issue, we developed our software as a Java applet so that it would work on any platform including mobile terminals.

The next issue is the rendering performance. Compared to texts or images, rendering 3D geometry requires much computation cost, in particular for rendering large-scale geometric data. Further, our system must work at a frame rate in any environment from desktop PCs to mobile terminals. For this issue, we introduce a hierarchical geometry encoding method "QSplat [1]" as a 3D geometry format. With the QSplat rendering system, we can adjust the image quality of the rendered 3D model and its frame rate according to the terminal's environment and users' interactions.

The third issue is the synchronization between the geometric data and other data in CMS. Because we want to use 3D geometric data as multimedia data in web-based CMS, the objects' locations or objects' portions in the data should be associated with other information, such as texts or sounds in the CMS database. Therefore, we develop a method to draw annotations at the particular positions of the objects, and for HTML synchronization functions. Annotations show names and details onto the geometric data, and HTML synchronization offers user's much interactive browsing experience between the 3D object and the HTML browsers. If users click the links in the HTML browser, the viewing position in the 3D geometric data browser smoothly moves to the particular position. On the other hand, when annotations in the 3D geometric data browser are clicked, the associated HTML or other multimedia data are activated and users can smoothly see and hear a detail description of the portion.

Several solutions, such as VRML, have been proposed to show 3D data in web browsers, called Web3D. But these usually cannot be applied when the objects are complicated, because the size of such data is enlarged by a complicated 3D geometric object and it is hard to render in an interactive frame rate. Several commercial Web3D technologies, such as Cult3D [2], Viewpoint [3], Shockwave 3D [4] solve the data size problem and have various functions for more effective rendering or animation.

Point-based rendering is more useful than polygon-based rendering in cases where the shape of the 3D object is complex [5]. Here, the model is assumed to be of a lot of points of various sizes. These sorts of rendering method can easily introduce multi-resolution rendering, which enables dynamic changes in the resolution of models while the rendering procedure proceeds. QSplat, Surfels[6], and several other systems[7,8,9] are examples of the point-based rendering methods. On our approach, we employed the QSplat method for rendering and data transmission.

In the reminder of this paper, we first give an overview of our systems in Sect.II. Next, we describe QSplat, on which we base our rendering system, in Sect.III. We explain the way of storing and rendering annotations in Sect. IV. In Sect. V, we present an example of synchronization of object rendering and

Manuscript Received on August 20, 2006.
The authors are with Osaka University. Prof Haruo Takemura can be contacted at E-mail: takemura@cmc.osaka-u.ac.jp

HTML browsers. The experimental results are shown in Sect. VI. Finally, we show applications of our method and give conclusions in Sect. VII and Sect. VIII respectively.

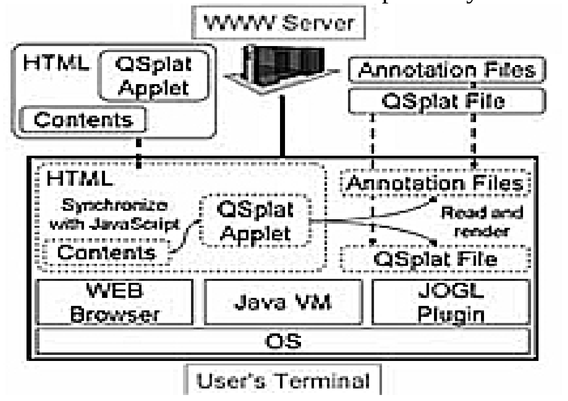


Fig. 1. Architecture of our rendering system. First the user's terminal downloads QSplat Applet with web browser. Then the QSplat applet loads a QSplat File and Annotation Files and renders the 3D model with JOGL.

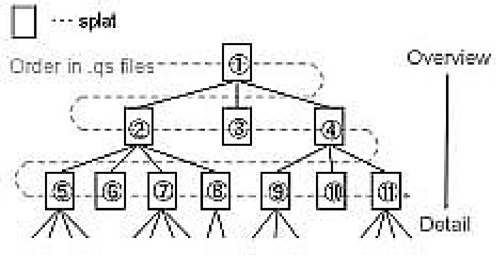


Fig. 2. QSplat data is constructed as a tree structure. Higher level nodes correspond to a rough geometric structure and lower level nodes correspond to the details.

## II. SYSTEM OVERVIEW

The architecture of our system is shown in Fig. 1. This system works on a web browser, Java Virtual Machine (JVM) and JOGL (Java Bindings for OpenGL) [10]. First, the web browser downloads the html file and applet. The applet loads the 3D data (QSplat file) and annotation files from the server, then it renders the 3D model using JOGL.

The annotation files manage information such as annotation IDs and descriptions, or relationships to the 3D model. Anno-tation IDs are used in both HTML (JavaScript) and the Java applet for HTML synchronization. The description can contain various information including words, sentences, hyperlinks or numerical data. The field is used not only for text but also for hyperlinks of sound data or viewing position and directions.

## III. QSPLAT RENDERING

QSplat is a kind of point-based rendering system. The object is expressed as a set of the points (SPLATs) of various sizes.

Each splat is defined by the parameters of; center position, radius, normal, normal cone, and optionally color. Further, the whole object geometry is constructed as a tree structure (Fig. 2). Here, one node corresponds to a splat. One parent node has up to 4 child nodes. Higher level nodes correspond to a rough geometric structure, and lower level nodes correspond to the details. In the QSplat file, each splat's parameters are quantized so that one splat data is encoded to 4 bytes (if color data, 6 bytes).

On the rendering stage, the tree is searched from high-level to low-level nodes. Tree traversal is stopped at the appropriate level and the nodes of that level are rendered. Here, the visibilities of the nodes are also considered. As the target nodes are at a lower level, the result becomes finer and more time is necessary for rendering. Thus, if the processing power is limited, the program draws higher level nodes and keeps interactive frame rates (Fig. 3).

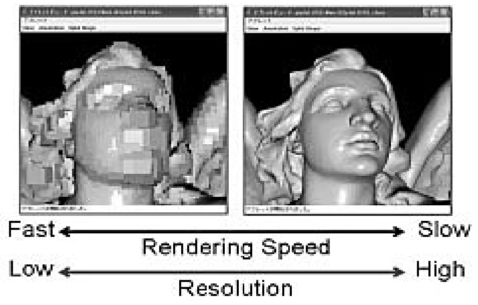


Fig. 3. Trade-off between rendering speed and resolution. As the target nodes are at a lower level in QSplat tree, the result becomes finer and more time is necessary for rendering. If the processing power is limited the system renders higher level nodes in order to keep interactive frame rate.

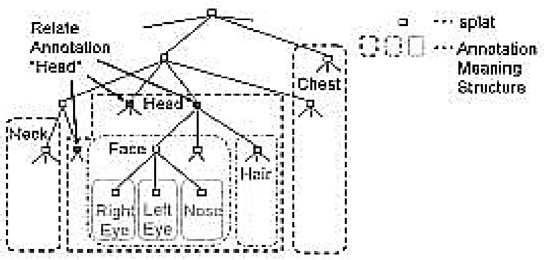


Fig. 4. Relations between annotations and QSplat nodes. One annotation may correspond to multiple nodes in a QSplat tree.

The system can draw the data while the data file is being loaded from server, because the QSplat data are stored from high-level to low-level data in the file. Thus, the user doesn't have to wait until the system loads all the QSplat data from the server. This is a good feature for low bandwidth environments and reduces user irritation. For the same reason, our system draws low resolution (high level) data while a user is changing viewpoints in the browser. During this operation, a quick and interactive response is achieved for user operations.

## IV. ANNOTATION

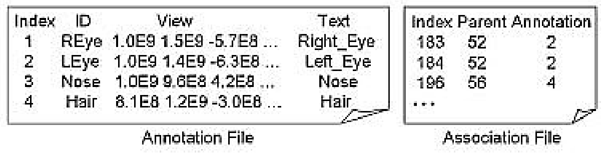
Annotations show names or explanations of 3D models or portions of 3D models, and are rendered with the 3D model. Each annotation has a unique ID (annotation ID). Annotation IDs are commonly used in rendering systems and HTML. How to use annotation ID is described in detail in Sect. V.

### *A. Annotation Files*

Annotations are related to the nodes or leaves in a QSplat tree. If the associated QSplat nodes are visible and rendered, the annotation is also rendered near the splats.

In most cases, one annotation corresponds to multiple nodes. In such cases, we associate the annotation and the root-nodes of the highest subtrees. Fig. 4 shows the case that the annotation "Head" is related to the 3 splats indicated by arrows.

Because the annotation and the splat nodes are not in a one to one correspondence, we used two files to describe the annotation: annotation and association files. An annotation file describes unique annotation IDs, view information (position,



Annotation File

| Index | ID | View | Text |
|---|---|---|---|
| 1 | REye | 1.0E9 1.5E9 -5.7E8 ... | Right_Eye |
| 2 | LEye | 1.0E9 1.4E9 -6.3E8 ... | Left_Eye |
| 3 | Nose | 1.0E9 9.6E8 4.2E8 ... | Nose |
| 4 | Hair | 8.1E8 1.2E9 -3.0E8 ... | Hair |

Association File

| Index | Parent | Annotation |
|---|---|---|
| 183 | 52 | 2 |
| 184 | 52 | 2 |
| 196 | 56 | 4 |
| ... | | |

Fig. 5. Annotation and Association File. The Annotation File keeps annotation's ID, annotation text, view points, and the Association File keeps relationships between annotation Indexes and splat Indexes.

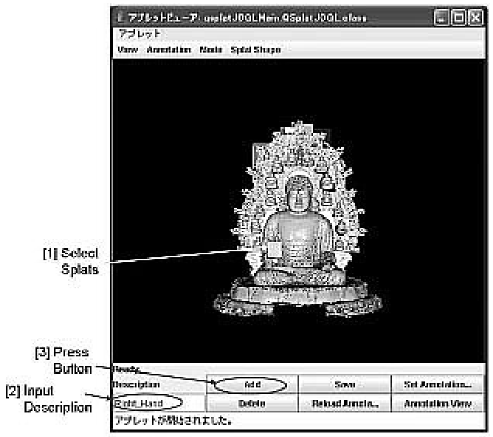


Fig. 6. Annotation editing tool. User can interactively set annotations onto splats by just clicking the portion of the model.

rotation, rotation axis, field of view) and annotation texts. The view information is used for the HTML synchronization function. When a user wants to see a particular portion of the 3D data that accords with the annotation name, this view information is used to set the correct viewpoint.

The association file contains the relationship between the annotation and the splat indexes. Here, one annotation may correspond to multiple Splats. Fig. 5 shows an example.

To make annotation and association files, we developed the editing tool shown in Fig. 6. Designers can easily create and edit annotations by clicking the splats of 3D data and writing texts or other descriptions.

### *B. Drawing Annotations with a 3D Model*

Annotations must be drawn at the "free space" of the 3D model or other annotations; arrows indicate correspondences between annotations and corresponding splats (Fig.7). Here, two issues need considering; 1. how to find the "free space," and 2. how to avoid "crossing" of the arrows. We describe the algorithm below.

First, we find the rectangular area where the 3D model is rendered. We call this area the "model-rendered area"(Fig. 7(a)). The model-rendered area is split into 4 sub-areas (Fig. 7(a)). Then, 4 areas for locating annotations are defined around the model-rendered area, which we call "annotation-area." Each annotation-area corresponds to each sub-area. The annotations are located by which sub-area the annotated splats are rendered in (Fig. 7(b)). For example, if the annotated splats are rendered in sub-area I, the annotation is located in the annotation-area 1. If multiple sets of annotated splats are located in the same area, the vertical order within the area is used for the arrangement of the annotations. Namely, if three sets of annotated splats are rendered in an area in a particular order, the three annotations are drawn in the free space of the

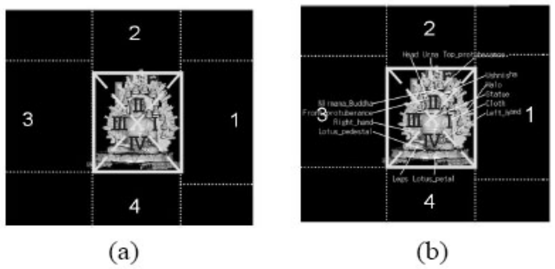


Fig. 7. Deciding annotation arrangement. (a)First, acquire "model-rendered area" and divide the area into 4 triangular sub-areas (I-IV). Then, define 4 "annotation-areas" for locating annotations(1-4), which correspond to sub-areas respectively. (b)Then, acquire the sub-areas where the annotated splats are in, and render the annotations onto the corresponding annotation-areas.

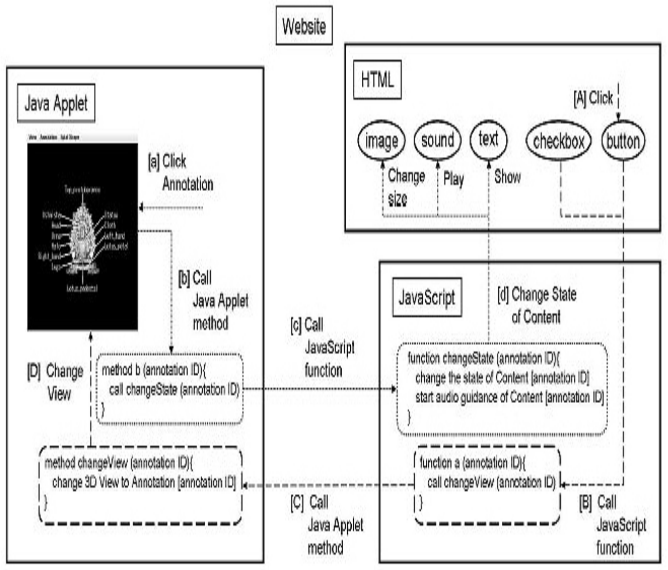
area with the same vertical order of the splats. Finally, arrows are drawn to connect the annotations and the splats.

## V. SYNCHRONIZATION WITH HTML

Our applet can interact with other HTML contents thorough JavaScript, which can call methods of the Java applet, and the Java applet can call functions of JavaScript. We utilize these features. In our applet, some methods are defined for interacting with JavaScript. The HTML synchronization is realized by JavaScript functions calling these methods and controlling the HTML contents.

The architecture of synchronization is shown in Fig. 8. If the user clicks a content (Fig. 8 [A]), the associated JavaScript function is called with an annotation ID as an argument (Fig. 8 [B]), where the annotation IDs are predefined to identify annotations and contents associated with a particular annotation. This function calls the applet method with the same annotation ID (Fig. 8 [C]), and the view of QSplat is changed to a predefined one according to the annotation ID (Fig. 8 [D]). On the other hand, if the user clicks an annotation in the applet (Fig. 8 [a]), the predefined method is called with an annotation ID (Fig. 8 [b]), Then the predefined method calls the JavaScript function with an annotation ID (Fig. 8 [c]), and these functions control particular contents corresponding with the annotation ID(Fig. 8 [d]).

We have made a sample website that expounds on the Great Buddha, as shown in Fig. 9. In this site, our applet is located on the left, and the descriptions of each part of the Great Buddha are located on the right. These descriptions correspond to each annotation in the applet. If headings of the descriptions are clicked, JavaScript functions are called to change the view


Website
Java Applet
HTML
[A] Click
image
sound
text
checkbox
button
Change size
Play
Show
[a] Click Annotation
[b] Call Java Applet method
[c] Call JavaScript function
[d] Change State of Content
JavaScript
[D] Change View
[C] Call Java Applet method
[B] Call JavaScript function

```
method b (annotation ID){
  call changeState (annotation ID)
}

function changeState (annotation ID){
  change the state of Content [annotation ID]
  start audio guidance of Content [annotation ID]
}

method changeView (annotation ID){
  change 3D View to Annotation [annotation ID]
}

function a (annotation ID){
  call changeView (annotation ID)
}
```

Fig. 8. Flow of HTML synchronization of our applet. If user clicks a HTML link related to the annotation of 3D data, a function in Java applet is called through JavaScript and changes the view point of the 3D viewer. Inversely, if a annotation in the applet is clicked, a JavaScript function is called, which shows descriptive texts, plays sounds, or performs the other useful operations.

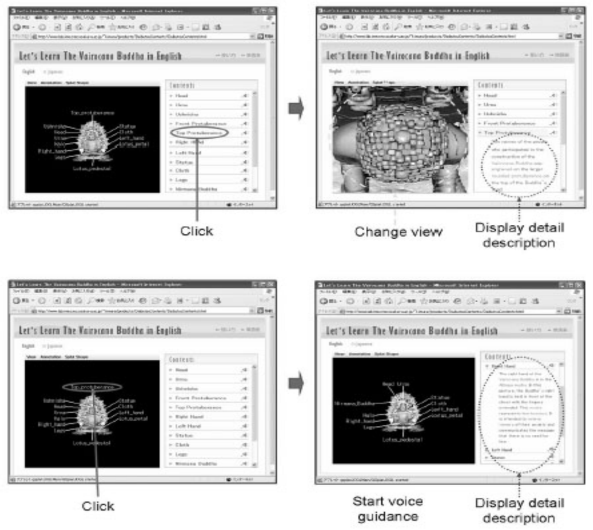

Let's Learn The Vairocana Buddha in English
Contents
Click
Change view
Display detail description
Click
Start voice guidance
Display detail description

Fig. 9. A web page using ours system, which expounds on the Great Buddha. Our applets are located on left, and the descriptions of the Great Buddha are located on the right. Each description corresponds to an annotation in our system. If user clicks a heading of the description, the applet's view point is changed to watch corresponding part of the model. Inversely, if an annotation in the applet is clicked, the detail description corresponding to the annotation is shown.

of QSplat and to show the detailed description. If the user clicks an annotation in the applet, a predefined JavaScript is called, and a detailed description is shown on the right side of web page and the voice guidance is played.

## VI. Experiments

Two experiments were conducted to verify the usefulness of our rendering system in terms of rendering performance.

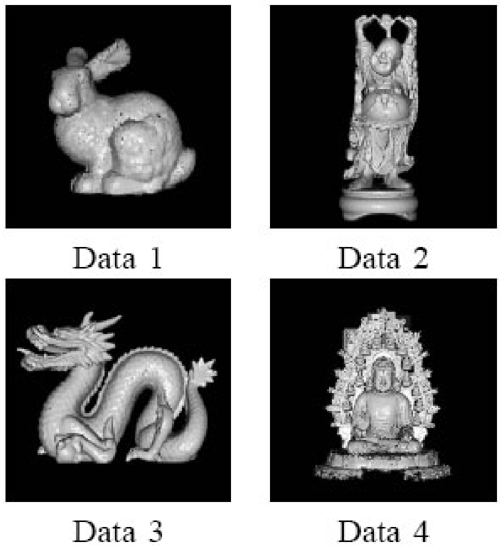


Fig. 10. Experimental 3D Geometry Data (VRML and QSplat) from The Stanford 3D Scanning Repository. Data 1 consist of 69000 polygons, Data 2 include 1087000 polygons, Data 3 consist of 871000 polygons, and Data 4 include 3109000 polygons.

### *A. Experiment 1: Comparing with existing rendering systems*

We compared the rendering speed of our system and that of a VRML viewer, the Cortona VRML Client[11]. VRML viewers are widely-used 3D rendering systems that can be executed on web browsers. In Experiment 1, we prepared 4 kinds of VRML data and corresponding QSplat data (Fig. 10) and located them on a server. Then, we downloaded them with a client PC (Intel Pentium 4 2.5 GHz, 768MB memory, NVIDIA GeForce4 Ti4200, Internet Explorer 6.0, JRE 1.5.0.07) through a wired LAN (about 85 Mbps), rendered them with Cortona or our system, and measured their execution time. We got Data 1, Data 2, and Data 3 from The Stanford 3D Scanning Repository [12].

Table I shows the results of Experiment 1. First, our system's executing time to finish rendering was much shorter. Specifically , the larger the data size, the larger the difference between the execution times. When a viewer renders VRML data, a scene graph is made, thus if the data size is large, execution time increases explosively. On the other hand, with QSplat, a scene graph is not made, so even if the data size is large, execution time increases only linearly. Next, because our system renders from lower to higher-resolution data sequentially, users can reasonably quickly see the first view as the time to first view depends on the lowest data. In VRML, users have to wait until all data are rendered. This waiting time gets longer proportional to the data size. In terms of frame rate, the frame rate of the VRML viewer in changing the view was smaller than that of our system for each data size.

TABLE I: COMPARISON OF OUR RENDERING SYSTEM WITH A VRML VIEWER

| | Data Size [MB] | | Number of Polygons [thousands] | Number of Splats [thousands] | Time to Begin Displaying Image [sec.] | | Time to Finish Rendering [sec.] | | Frame Rate While Changing View [fps] | |
|---|---|---|---|---|---|---|---|---|---|---|
| | VRML | QSplat | VRML | QSplat | VRML | QSplat | VRML | QSplat | VRML | QSplat |
| Data 1 | 4.4 | 0.2 | 69 | 58 | 2 | 1 | 2 | 2 | *a | 8.0 |
| Data 2 | 71.2 | 6.7 | 1087 | 1757 | 71 | 1 | 71 | 4 | 1.2 | 8.0 |
| Data 3 | 56.8 | 8.1 | 871 | 2129 | 51 | 1 | 51 | 5 | 1.2 | 8.0 |
| Data 4 | 142.5 | 9.9 | 3109 | 2591 | *b | 1 | *b | 6 | *b | 8.0 |

[a] enough high speed.
[b] The system can't render the model because the data sizes are too large.

These experimental results show that users can not smoothly browse large size 3D geometric data with a VRML viewer. However, using our system, the resolution of the image is adjusted to maintain the frame rate so that users can browse even large size 3D geometric data.

TABLE II: COMPARING EXECUTION ON A DESKTOP TERMINAL WITH ON A MOBILE TERMINAL, AND COMPARING EXECUTION THROUGH A WIRED LAN WITH THROUGH A WIRELESS LAN.

| | Data Size [MB] | Number of Splats [thousands] | Download + Rendering Time [sec.] | | | Re-rendering Time After Changing View [sec.] | |
|---|---|---|---|---|---|---|---|
| | | | Desktop/ Wired | Mobile/ Wired | Mobile/ Wireless | Desktop | Mobile |
| Data 1 | 0.2 | 58 | 1.4 | 1.6 | 6.1 | - | 0.2 |
| Data 2 | 6.7 | 1757 | 4.2 | 4.8 | 59.8 | 3.0 | 3.6 |
| Data 3 | 8.1 | 2129 | 4.4 | 5.4 | 70.2 | 3.1 | 3.9 |
| Data 4 | 9.9 | 2591 | 5.3 | 6.2 | 86.4 | 4.0 | 4.5 |

### *B. Experiment 2: Executing in several environments*

We executed our systems on a desktop PC (same one used in Experiment 1) thorough a wired LAN (about 85 Mbps), on a mobile PC (Intel Pentium M 1.1GHz, 504 MB memory, Intel 82852/82855 GM/GME Graphics Controller, Internet Explorer 6.0, JRE 1.5.0.07) through a wired LAN, and on a mobile PC through wireless LAN (about 8Mbps), and measured and compared their execution times.

Table II shows the results of Experiment 2. The difference of terminal doesn't materially affect the execution time. It shows that our system has the potential to be used on any terminal. On the other hand, network speed enormously influences execution time. In the case of a wireless LAN, execution time increases in proportion to data size. In actual use, if the data size is too large, our system stops downloading at a proper resolution level and renders the model at that level.

## VII. APPLICATIONS

We designed this rendering system to use with content management systems. For example, e-learning systems such as those involving biology, art or history need this sort of geometric data. Also, applications for digital archive systems need this sort of rendering system. We are now designing a new CMS that can deal with QSplat data and their annotations. Our rendering system is included in this CMS. The architecture of this CMS is shown in Fig. 11

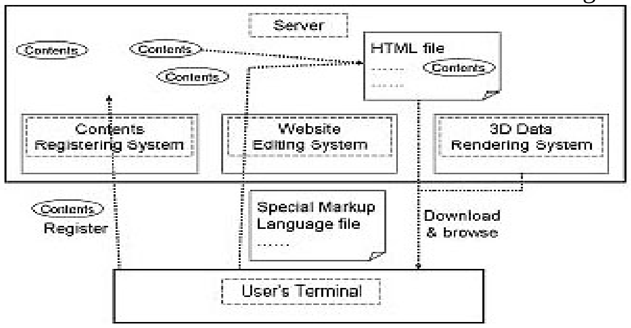


Fig.11. Proposed CMS using Our Applet. Designer uploads 3D data and bodytext with special markup language onto server. The server analyzes the markup language and generates HTML files.

The system consists of contents registering, website editing, and 3D data rendering systems. The contents registering system manages data including texts, images or other multimedia files. The website editing system generates HTML files from source scripts that users edit with a special markup language. The 3D data rendering system is downloaded with HTML files and renders 3D geometric data on a HTML browser. Our applet is used as the 3D data rendering system. Using this CMS, a user can design websites with various contents including 3D geometric and annotation data, even if the user doesn't have detailed knowledge of HTML or the architecture of the CMS.

## VIII. CONCLUSION AND FUTURE WORK

We proposed a new 3D geometric data rendering system that can be used in conventional Web browsers or Content Management Systems. We implemented QSplat as a Java Ap -plet so that it can quickly and precisely render large scale 3D geometric data. Our rendering system can dynamically control the rendering frame rate and the preciseness of the rendering. Thus, it can be used in a variety of hardware/software/network environments such as powerful desktop PCs or mobile wireless terminals. The HTML synchronization technique enables users to smoothly view 3D geometry contents.

We will extend this system for use with the Content Management System. We are now trying to implement a website editing system that allows users to make a website with registered 3D geometric data and synchronizing functions. We are also going to implement a content registering system that users can register 3D geometric data through a web interface. For this, we are presently designing a markup language and implementing functions to convert it into HTML.

## ACKNOWLEDGMENT

This work is supported by Ministry of Education, Culture, Sports, Science and Technology under the "Development of fundamental software technologies for digital archives" project.

## REFERENCES

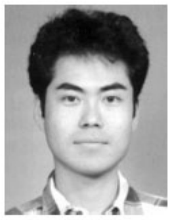
[1] S. Rusinkiewicz, M. Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes, Proc. ACM SIGGRAPH 2000, pp.343-352, 2000.
[2] "Cult3D," http://www.cult3d.com/
[3] "Viewpoint," http://www.viewpoint.com/pub/
[4] "Adobe Shockwave Player,"http://sdc.shockwave.com/products/shock -waveplayer/
[5] T.Fujimoto, K.Konno and N.Chiba. Introduction to Point-based Graphics, The Journal of the Society for Art and Science Vol.3, No.1, pp.8-21, 2004.
[6] H. Pfister, M. Zwicker, J. Baar and M. Gross. Surfels: Surface elements as rendering primitives, Proc. ACM SIGGRAPH, 2000
[7] F. Duguet, G. Drettakis. Flexible Point-based Rendering on Mobile Devices, IEEE Computer Graphics and Applications, Vol.24, No.4, pp.57-63, 2004.
[8] Y. Okamoto, S. Yamazaki and K. Ikeuchi. Efficient Point-based Rendering Method for Huge 3D Models using Sequential Point Clusters, Proc. Meeting on Image Recognition and Understanding 2004 (MIRU 2004), Vol.1, pp.207-212, 2004.
[9] Q.Peng, W.Hua and X.Yang. A new approach of point-based rendering, *Proc. Computer Graphics International 2001*, pp.275-282, 2001.
[10] "Java Bindings for OpenGL," https://jogl.dev.java.net/.
[11] "Cortona VRML Client," http://www.parallelgraphics.com/ products /cortona /.
[12] "Stanford 3D Scanning Repository," http://graphics.stanford.edu/data /3Dscanrep/.

**Yousuke Kimura** received the B.S. in Osaka University in 2005. He is currently pursuing his Master's degree in Information Technology at Osaka University. He is engaged in studies relating computer graphics. He is a student member of IEICE.

**Tomohiro Mashita** graduated from Osaka University in 2001 and completed the M.S. and doctoral programs in 2003 and 2006, respectively. He is a postdoctoral fellow in Osaka University. He is engaged in studies relating to computer vision and human interface. He is a student member of RSJ, HIS, and IEICE.

**Atsushi Nakazawa** received the B.S degree in 1997 and the Ph.D. degree in Engineering Science from Osaka University in 2001. Since 2001, he have been worked for Institute of Industrial Science, University of Tokyo as a postdoctoral researcher. Since 2003, he have been working for Cyber media Center, Osaka University as a lecturer. His research interests are Computer Vision and Robotics, in particular, estimating human posture from images, analysis human motion using motion capture and humanoid robot control. He is a member of IEEE, IPSJ, and RSJ.

**Takashi Machida** received the B.S. in Osaka University in 1998. In 2000, he received the M.S. degrees in Nara Institute of Science and Technology. In 2002, he became a Assistant Professor at the Cyber media Center of Osaka University after he retired from the Graduate School of Nara Institute of Science and Technology. He is engaged in research on image processing, computer vision and computer graphics. He holds a PhD in Engineering. He is a member of IEEE and ACM.



**Kiyoshi Kiyokawa** received his M.S. and Ph.D. in Information Systems from Nara Institute of Science and Technology in 1996 and 1998, respectively. He was a Research Fellow of the Japan Society for the Promotion of Science in 1998. He worked for Communications Research Laboratory from 1999 to 2002. He was a visiting researcher at Human Interface Technology Laboratory of the University of Washington from 2001 to 2002. He is currently an Associate Professor at Cyber media Center, Osaka University. His research interests include Virtual Reality, Augmented Reality, 3D User Interface and CSCW. He is a member of IEICE, IPSJ, VRSJ and ACM.

**Haruo Takemura** received B.E., M.E., and PhD. degree from Osaka University in 1982 1984 and 1987 respectively. In 1987, he joined Advanced Telecommunication Research Institute, International. In 1994, he joined Nara Institute of Science and Technology, Nara as associate professor at Graduate school of Information Science and Technology. From 1998 to 1999, he was a visiting associate professor at University of Toronto, Ontario, Canada. In 2001, he became a full professor at Cyber media Center, Osaka University, Osaka, Japan. Since August 2005, he also serves as a vice-director of Cyber media Center. His research interest includes Interactive Computer Graphics, Human-Computer Interaction and Mixed Reality. He is a member of IEICE, IPSJ, VRSJ, HIS, IEEE and ACM.