3D Web-Based Shape Modelling: Data Extraction and Delivery

Ali Abdallah
The National Centre for Computer Animation,
Bournemouth University
Poole, United Kingdom
E-mail: ccengineer@gmail.com

Abstract— Despite the rapid development of hardware including specialized graphical processing units (GPUs) and widening the bandwidth, truly interactive applications allowing for near real-time visualization without loss of a visual quality are still to become a reality. Building up an adaptive 3D Web-based shape modelling environment enables us to design platform independent 3D objects in a collaborative manner, yet delivering compressed meshes, and images to clients with different platforms and devices in an optimized manner is still a difficult job. In this paper, we explore the crucial issues of delivering 3D objects. We focus on data transmission over different transmission bandwidths between servers delivering the service and clients with different platform devices. Our case study relies on a client-server adaptive architecture, supported by different rendering techniques, and is able to deliver compressed meshes and images. We identify the obstacles in delivering compressed data files as well as image streams, and present results based on bandwidth capacity, storage size, extraction and loading times.

Keywords- Adaptive architecture; 3D shape modelling; compression; mesh; image streams.

I. INTRODUCTION

Online applications should be efficient and characterized by real time response. Collecting information about clients is a must to keep any online application updated. This process is done using different available scenarios provided by different online and offline resources such as hardware changes, bandwidth monitoring, server stability and performance, memory consumption and availability, etc. [1].

Three major elements constitute the Web -based collaborative 3D shape modelling environment: Networking, Modelling and Rendering. The proposed environment requires a way of communication between its different parts, which is done by the Network element. 3D shape modelling takes place using the Modelling element which constitutes the core of the proposed environment. Rendering is responsible for generating the geometric model after rendering it at the GPU Level.

Polygonal meshes are considered the most common format to store and represent 3D models. They are embodied in the Boundary Representation (BRep). All modern application programming interfaces (APIs) usually support this format, and allow its implementation to the Web browsers. However, the polygonal mesh, by definition, is an approximation of the mathematically precise model that have well-known issues concerned with loss of the precise shape and visual property definition, limited complexity, large memory consumption, problem with transferring through networks, etc. The inability to access the construction history is also an issue. Transmission of 3D scenes is still a major issue in spite of the extensive research dedicated to resolve this kind of problems [2].

Flash animations are widely used on browsers which cannot support large amount of computational resources, and cannot handle large data files that require high Internet bandwidths, power GPUs and large memories. All are accessed by the browser itself. The problem with such option is that it does not support interactivity over the Web. Another option to be mentioned is generating a stream of images of a 3D model, taken from different angles covering 360 degrees, and displaying them on the browser after being arranged and grouped. The object will be loaded as a 3D model to the browser [3].

In a client-server architecture, 3D data is usually generated at the server side, and presented as polygonal meshes which are composed of vertices and facets and can be transmitted over the network [2][4]. Delivering 3D data and visualizing it using a Web browser over the Web is considerably slow and requires a lot of hardware resources (GPU and Memory) [5][6][7].

In this work, we address the problem of extracting and delivering 3D data contents resulted from different rendering techniques. We introduce data extraction methods to support the data delivery process over the client-server architecture; our aim is to optimize the extraction and the delivery process for better performance.

The paper explores different ways of extracting and delivering 3D models based on the used rendering technique. We focus on extracting and delivering 3D models built using a special adaptive 3D shape modeling environment. The paper structure is as follows: after a survey of related works, we describe the adaptive 3D shape modeling environment and identify the types of models generated based on its three rendering techniques. After that, we discuss the data extraction methods used. The paper discusses three different 3D content extractions and delivery methods based on the applied rendering technique, it compares the methods based on bandwidth, data size, computational time and visualization time. Finally, we discuss the results, as well as some practical recommendations reflecting the advantages and drawbacks of the proposed extraction and delivery techniques.

II. RELATED WORK

Modern computers, and even smaller devices such as tablets and smart phones, are equipped with suitable graphic adapters. Kristian et al. focus on improving the global positioning system (GPS) functionality and trying to decrease the expenses resulted from the browser at the client side. They try to increase performance of the current implemented Web features by designing an extension of the available Web document, in the form of 3D scene description format [14].

The huge demands on the transmission of Web -based 3D scenes over the Internet from servers to clients, accompanied by a variety of users with different hardware platforms, devices and GPU powers, prompt researchers focus on mesh compression techniques to ensure the delivery of 3D objects over the Web [2][8][9][12]. Progressive meshes (PM) were presented by Hoppe [10] in 1996, as a cumulative, trimmed and continuous format of polygonal mesh. PM are used in remote visualization, and are capable to display objects' details according to the performance of the requesting client machine [5][10]. In 3D scenes, minimizing the size of the transmitted data by compressing the polygonal mesh is not the only concern. The main objectives of the PM are speeding up the transmission of 3D data, compressing associated attributes like colour and texture, and keeping the quality level of details. Guillaume et al. discuss the above mentioned issues and propose an algorism that allows quick compression for 3D data and generates a binary compressed file [5].

Ramani et al. pointed out that, in order to distribute a model, a special modeller should be installed on each client, so that they can share and modify the same model using the Web browser, in a collaborative manner. They mention that some collaborative modelling systems allow editing some operations such as changing some specific and basic characteristics of the model. One good example of collaboration among users is WebSPIFF [11], which is a tool that allows users to create, modify and delete geometric objects.

Cloud computing empowers different existing technologies such as virtualization and parallel computing by applying some of its characteristics, which include rendering on demand, reliability and efficiency. Cloud computing allows users to benefit from high computing speed over the Web; it also allows storing a large amount of data and real time 3D rendering. Wu et al. described some requirements for their cloud based manufacturing, some of which include the real-time information collection about the clients. Another requirement is 3D models' distribution and sharing by accessing big data files stored on the cloud, as well as processing and managing these data files for modelling and rendering purposes [22].

One of the 3D data compression methods is the curvature prediction method, discussed by Adrien et al. This method is supported with a wavelet formulation method designed to improve rate-distortion (R-D) performance, it is a quantization method to increase the compression rate. These methods are applied to a simple algorithm designed for 2-manifold mesh compression [9].

Xinshu et al. discuss the problem of 3D sensitive information, and suggest an intra-origin data control system (CRYPTON), that allows owners to monitor and direct sensitive 3D data loaded to the browser at the client side [13].

III. ADAPTIVE 3D SHAPE MODELING ARCHITECTURE

The used modelling environment, which is an Adaptive 3D shape modelling architecture (ASMAR), is based on Hybrid representation or HRep, which is the integration and combination of both function and boundary representations. HRep acquires the characteristics of both functional representation (FRep) and boundary representation (BRep), and allows dealing with objects as volumes having an internal structure; it keeps the constructive tree of the modelling process, and allows the representation of the model in the form of a polygonal mesh.

BRep represents 3D objects in the form of a polygonal mesh, which is the approximation of the mathematically precise model. 3-dimensional objects are stored and represented as a set of inter-related facets and vertices, which when rendered, generates an image of a 3D model. This representation can be implemented in Web browsers because it is widely supported by the modern 3D APIs [15]. BRep can be stored in different 3D file formats using the polygonal mesh. Such files can hold all the necessary data needed to render the 3D object at the GPU level and they are easy to be created, stored, and accessed. One of the main concerns of such representation is about the loss of the model precision with the fact that BRep, as mentioned before, is an approximation of the real object. Another concern is about the lack of construction history of the object. Whenever the object is subjected to a change, the original 3D object will be lost and replaced by the latest version of the edited object. Another drawback is that BRep cannot handle objects with internal structures, and it is only concerned with the external surfaces of 3D objects rather than their internal structures. The size of 3D data files can be enormous when dealing with complex objects and facets. Serious problems can arise when transferring the files over the Internet, in addition to large memory consumption.

FRep is based on function representation of 3D models rather than facets and vertices. Such representation reduces the complexity of 3D models and can easily manipulate complex objects without the need of handling large data files and a huge amount of computational resources. HyperFun, which is an open source and high level programming language, uses FRep [16] to implement complex 3D models; HyperFun files are of small size and can handle complex geometric objects [16]. Even though FRep reserves the development track of the model, there are still major weaknesses in using FRep models, such as the fact that they are difficult to be treated and controlled inside a Web browser. Browsers load 3D models using polygonal meshes. This allows users to edit and modify 3D scenes and objects in an interactive manner [18]. One of the serious problems when using FRep, is the need for Java Applets to generate and display HyperFun objects to the Web browser, which could be a very expensive procedure and need vast amount of computational resources.

HRep, as mentioned before, is the combination and integration of both FRep and BRep modeling techniques. It combines the two and synchronizes them, trying to make use of their characteristics in an efficient way, by eliminating the drawbacks of each one.

Since HRep is composed of two major components, the conversion to these two components is a must. The conversion of FRep to BRep can be done through a process called Polygonization, which is a way of converting FRep object surface into a polygonal mesh by creating flat polygons to approximate shapes. Another way of conversion FRep to BRep is voxelization, which is a procedure responsible for the extraction of voxel representation from a continuous geometric representation. The extraction process can be very expensive in terms of time and resources [19]. The bi-directional conversion process, in Fig. 1, represents the conversion of BRep to FRep, using signed distance fields. Starting from a given polygonal mesh, using the distance-fields technique, we are able to obtain the exact function representation for that mesh, both the interior and the exterior of the object can be represented [20].

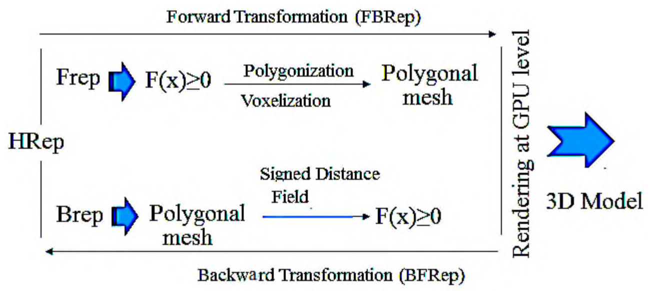


Figure 1. HRep bi-directional conversion showing a two way transformation between FRep and BRep (forward-backward).

IV. 3D DATA EXTRACTION AND DELIVERY

A. 3D Model Data File

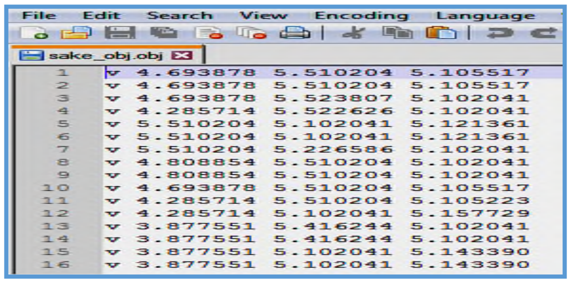
3D models are often stored as a set of 3D points called vertices. Indexing these vertices, allows to construct polygons, which constitute the surface of the 3D model [21].

Object (Obj) file is based on text file with ".obj" extension, it is an extensible 3D graphics (X3D) text based. It allows user authorship, and it is considered as an environment for 3D scenes construction using XML or classic VRML encoding [21]. Open file format was developed by Wavefront [15]. 3D Obj file was adopted by different 3D graphic vendors because it was easy to be imported and exported. Table I shows the structure of the file. The indicator is a single command line followed by a series of values representing the indicator. This digital data is loaded into the Obj file. Our 3D shape modeling environment is supported by special tools that allow to access the GPU buffer, and to extract the rendered 3D object data as vertices and facets and to save them to the Obj file [15].

TABLE I. OBJ FILE INDICATORS AND VALUES DESCRIPTION

Indicator	Description
#	Comment
v	Vertex
vn	Normal
vt	Texture Coordinate
f	Face
l	Line
G	Group

Extracting the 3D model from the GPU buffer takes place using the Marching Cubes Algorithm [15], which is one of the most applied rendering techniques used in 3D modeling. The extraction process allows to access the GPU memory and to extract both the vertices and the facets .The extracted vertices and facets are re-assembled where suitable obj indicators are added before being saved into the text based Obj file.



```
v 4.693878 5.510204 5.105517
v 4.693878 5.510204 5.105517
v 4.693878 5.523807 5.102041
v 4.285714 5.522626 5.102041
v 5.510204 5.102041 5.121361
v 5.510204 5.102041 5.121361
v 5.510204 5.226586 5.102041
v 4.808854 5.510204 5.102041
v 4.808854 5.510204 5.102041
v 4.693878 5.510204 5.105517
v 4.285714 5.510204 5.105223
v 4.285714 5.102041 5.157729
v 3.877551 5.416244 5.102041
v 3.877551 5.416244 5.102041
v 3.877551 5.102041 5.143390
v 3.877551 5.102041 5.143390
```

Figure 2. Obj File showing indicator v as vertex and three different extracted points next to each vertex as vertex values.

When completed, the obtained Obj file is loaded using OBJLoader, a special JavaScript function declared in Three.js JavaScript file engine. The OBJLoader function is responsible for loading 3D models with Wave front file format using Three.js as shown in Fig. 3 [23]. The code below shows X3D file embedded in XML encoding.

```
<X3D id="x3d">
  <Scene id="sc1">
    <Viewpoint position="0.15255 0.10231 0.19884"
```

```
            orientation="-   0.27505   0.95696   0.09264
            0.55979" >
    </Viewpoint>
    <Transform id="Hemi">
      <Inline url="Hemi-spher.x3d" solid="false"></Inline>
     </Transform>
  </Scene>
</X3D>
```

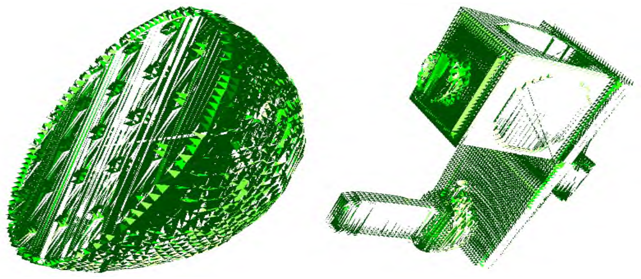


Figure 3. Hemi-sphere and sake-pot complex objects loaded from Obj files.

B. Data Extraction and Delivery

The data extraction and delivery process passes through different phases where different extraction and delivery techniques will be applied according to the technique used in rendering the 3D model. As shown in Fig 4, the Marching Cubes and Hybrid WebGL use the same extraction technique by accessing the GPU memory, extracting all vertices and facets, saving them to a predefined Obj file and loading the Obj file into the browser.

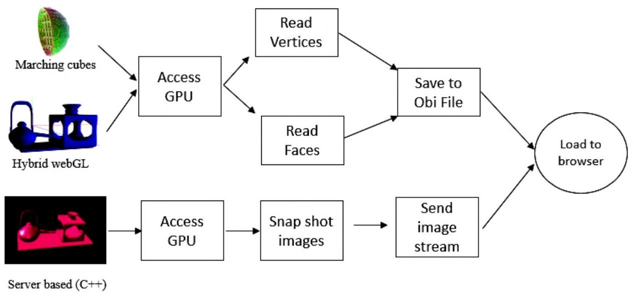


Figure 4. Work flow model showing three different data extraction and delivery processes applied on three different rendered techniques starting from GPU level.

Server based rendering uses a different approach. It starts by accessing the GPU, not to read the points from memory, but to take snapshots for images, from different angels, then it sends the images as an image stream to the client after saving them in an image matrix. The image matrix is then displayed in the browser, as indicated in Fig 4.

The data extraction process is designed to extract and deliver 3D data safely over the Internet. Its role is to access the GPU buffer, extract all the needed data, save them in Wavefront (Obj) file format, then deliver the Obj file over the transmission media. Using (1), we can obtain all points that constitute the 3D model [23].

$$pt = \sum_{z=0}^{s} \sum_{y=0}^{s} \sum_{x=0}^{s} x + s*y + s^2*z. \qquad (1)$$

Where
s: Size of the geometric cube.
pt : 3D point.
vert[]: array of extracted 3D point vertices.
vindex: vertex index.
va[] : array of all vertices.
facet: array of facets .

$$ptx = pt+1 \qquad (2)$$
$$pty = pt + s \qquad (3)$$
$$ptxy = pty + 1 \qquad (4)$$
$$ptz = pt + z^2 \qquad (5)$$
$$ptxz = ptx + s^2 \qquad (6)$$
$$ptyz = pty + s^2 \qquad (7)$$
$$ptxyz = ptxy + s^2 \qquad (8)$$

The process of extracting a vertex is shown in (9)

$$vert[] = \sum_{i=0}^{3} \text{vertices[vindex][i]} \qquad (9)$$

The vertices of the whole object can be obtained using (10)

$$va[] = \sum_{a=0}^{Nb} vert[a] \qquad (10)$$

And the facets can be obtained using (11).

$$face = \sum_{a=0}^{Nb} \sum_{i=0}^{3} face[a][i] \qquad (11)$$

In our study, since we have three types of rendering techniques (Marching cubes, Hybrid WebGL and Server based using C++), it is useful to do experiments on the three rendering types. Our experiments are applied to three different models, one simple model (Android) and two complex models (Hemi-sphere and Sake pot).

All experiments run over a 64-bit windows 7 operating system, use Intel core i7-6700 CPU at 3.4 GHz. The available installed RAM is 16 GB, the GPU being used is Intel HD Graphics 530 (GT2) with 1150 MHz clock with ability to access the main memory (2x64 bit DDR3L-1600), we use google chrome browser version 59.0.3071.115 (64 bit).

Since we are using dynamic complex objects that are subjected to different parameter changes, the 3D data extraction process will directly be affected by these changes. One of the parameters is the resolution parameter. In our experiment, we apply the extraction process to the resolution test, in order to monitor the extraction process time and size. The extraction test of the resolution parameter is performed on the three different models we have.

C. Marching Cubes Data Extraction and Delivery

In this study, we use three objects, Hemi-Sphere and Sake Pot (both are complex objects) and Android robot (simple object). The loaded files formats are OBJ and X3D file formats.

TABLE II. RENDERING USING MARCHING CUBES

Rendering using Marching Cubes			
	Hemi-Sphere (complex object)	Sake Pot (complex object)	android (simple object)
Loading time in sec	0.382 sec	0.275 sec	0.104 sec
No. of Vertices	52504	55652	2828
No. of Facets	27119	27814	1352
Total	79623	83466	4180
Loading Obj File in sec	0.559 sec	0.598 sec	0.105 sec
OBJ File Size in KB	1707 KB	1875 KB	86 KB
Loading X3D File	0.001 sec	0.002 sec	0.001 sec
X3D File Size in KB	1976 KB	1902 KB	72 KB

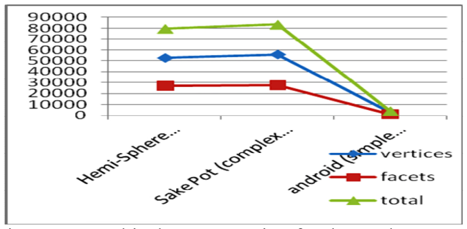


Figure 5. Graphical representation for the total extracted vertices and facets, comparing the 3 different objects.

After the extraction process is performed, we obtain the following results, as shown in table II.

- The loading time difference between the three objects is minor, with no big difference.
- The extracted facets and vertices for the two complex objects are approximately the same, and they are double of those of the simple objects
- The OBJ loading time of all objects is acceptable and is less than 0.6 sec.
- The file size of the complex objects is relatively big (more than 1.5 MByte) while that of the simple objects is relatively small, less than 100 Kbytes.
- The loading time of the X3D file is significantly better than that of the Obj file and is considered relatively short.
- The X3D file size is close to the Obj File size of the three objects.

Table II shows the variation in loading extraction time between 0.104 sec for simple object and 0.382 sec as max for the Hemi-sphere complex object. This variation indicates that the extraction process can be very fast and with no major difference between simple and complex objects when applying Marching Cubes rendering. We can notice that the total number of extractions for Hemi-sphere (79623) is less than that for the sake-pot (83466) by 5% while the extraction time for Hemi-sphere is considerably higher (0.382 sec) than that of the sake-pot (0.275 sec) by almost 37%. The fact is that the extraction time is not directly proportional with the extracted number of facets and vertices, but it is related to the complexity level of the rendered object itself. In our case, the Hemi-sphere model is considered more complex than the sake-pot model and that is the reason why the extraction process taking more time. The number of the extracted facets and vertices in Fig. 5 shows that the more vertices and facets to be extracted, the longer the extraction time.

D. Hybrid Data Extraction and Delivery

This test is applied to two complex objects using the Hybrid rendering technique. Here, the 3D complex objects are subjected to a resolution parameter. The extraction process takes place in six different phases, where the resolution parameter is changing accordingly.

TABLE III. EXPERIMENT APPLIED ON THREE DIFFERENT OBJECTS USING HYBRID RENDERING.

Density	Load time in sec	No. of Vertices	Vertices Extraction Time	No. of Facets	Facets Extraction Time	Total extraction Time	.Obj Size in KB
Hemisphere: Complex Object using Hybrid modeling							
30	0.1 sec	12868	25.7 sec	12868	17.2 sec	42.9 sec	769 KB
40	0.4 sec	12868	60.0 sec	12868	73.3 sec	133.3 sec	1787 KB
50	0.4 sec	38160	419.9 sec	38160	285.7 sec	705.7 sec	2341 KB
60	0.9 sec	55648	1076.8 sec	55648	918.5 sec	1995.4 sec	2751 KB
70	1.5 sec	73568	3584.2 sec	73568	2544.6 sec	2128.8 sec	4666 KB
80	1.7 sec	94584	8544.0 sec	94584	6585 sec	4529.0 sec	5227 KB
Sake-Pot: Complex Object using Hybrid modeling							
30	0.1 sec	2636	5.7 sec	2636	3 sec	8.6 sec	157 KB
40	0.2 sec	7848	39.0 sec	7848	21.5 sec	60.5 sec	322 KB
50	0.2 sec	16460	180.2 sec	16460	90.0 sec	270.2 sec	531 KB
60	0.5 sec	28468	136.8 sec	28468	121.6 sec	258.4 sec	699 KB
70	1.3 sec	37492	124.8 sec	37492	280.2 sec	405.0 sec	1053 KB
80	2.2 sec	40488	1140.2 sec	40488	1140.2 sec	1080.4 sec	1355 KB

From the results shown in Table III, we conclude the following:

- The loading time is increasingly changing with the changing of the resolution of the extracted objects. The higher the resolution, the longer the loading time needed to display the object. A simple comparison shows that loading an object of 30% resolution takes a small fraction of a second (0.179 sec for a Hemi-sphere) while loading the same objects of 80% resolution takes 1.701 seconds, which means 10 times more time is needed to load the same object.

- The number of extracted points (vertices and facets) increases as the resolution of the extracted object increases. Therefore, the file size increases.
- Table III shows that the more complex the 3D object are, the longer extraction time is needed, because more facets and vertices are to be extracted. Therefore, more storage space is needed to save the Obj files.
- Fig. 6 shows a large difference in the extraction times between two different complex objects. The extraction time is directly affected by the resolution of the object, the level of complexity and the number of extracted facets and vertices.
- Fig. 7 compares the Obj file sizes, and shows that the higher the level of complexity, the more storage is needed to save the Obj files. In our experiment, the Hemi-sphere needs more than 5 Mbytes to store the Obj file of 80% resolution while it needs less than 1.5Mbyte to store the Obj file for the sake-pot model at the same level of resolution.

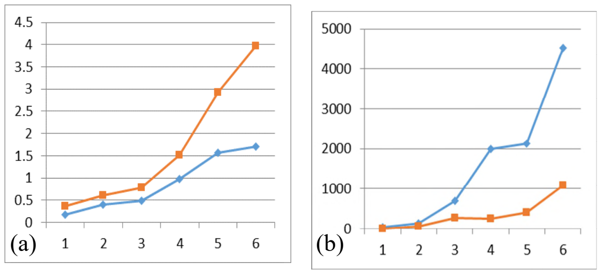


Figure 6. Graphical representation for the time taken (a) and the extracted values (b) for Hemi-sphere (orange line) and Sake Pot (blue line) respectively when applying the resolution parameter.

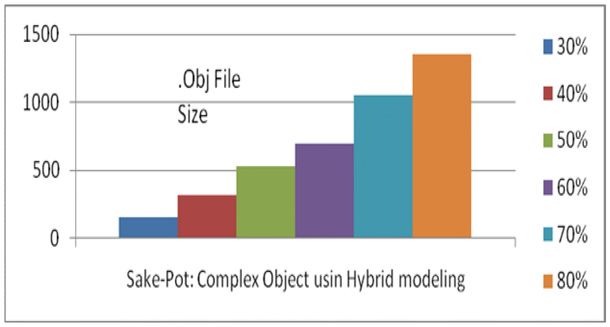


Figure 7. Bar chart representation illustrating the Obj file size with respect to resolution parameter. As the resolution parameters increase, the file size increases.

E. Server Based Data Extraction and Delivery

TABLE IV. EXPERIMENT APPLIED ON SAKE-POT OBJECTS USING SERVER BASED RENDERING

Snap-shot rate	Matrix size	No. of images	Time taken	Images/sec	Image size\Kbytes	Total size
2	45*45	2025	198 sec	10 image/sec	100 KB	197.7 KB
3	42*42	1764	224 sec	7.8 image/sec	100 KB	172.2 KB
4	38*38	1444	274 sec	5.2 image/sec	100 KB	141.0 KB
5	35*35	1125	160 sec	7.0 image/sec	100 KB	109.8 KB
6	32*32	1024	166 sec	6.1 image/sec	100 KB	100 KB
7	28*28	784	126 sec	6.2 image/sec	100 KB	76.5 KB
8	25*25	625	112 sec	5.5 image/sec	100 KB	61.0 KB
9	22*22	484	104 sec	4.6 image/sec	100 KB	47.2 KB

Server based rendering generates images with fixed size; the precision of the object depends on the number of generated images. The snap-shot rate determines the rate of captured images per sec. In our test, we apply eight different rates starting from two and increasing up to nine. As the rate increases, the number of generated images decreases.

Table IV shows that at rate two, the image matrix is 45 by 45 or 2025 images, while at rate nine, the image matrix becomes 22 by 22 or 484 images only. As the snap-shot rate increases, the number of captured images and the time taken to generate the images decreases; therefore, the total size of the generated images decreases as well. That means, the higher the snap-shot rate, the more precision we obtain, and more storage size and image generation time are needed.

Table IV reveals that when the image matrix starts to shrink, the number of images starts to decrease, and the time taken to generate the images starts to decrease too, since the image size is fixed, the storage size needed to save all the generated images is directly proportional to the number of generated images. As a result, when the image matrix shrinks, the precision of the 3D model is reduced, and the storage space needed to store the object images is reduced as well.

V. CONCLUSION

In this work, we identify and implement a special extraction and delivery data module to extract the online mesh (vertices and facets) and deliver the raw data after being loaded into different file formats or image streams.

The visual and numerical results have lead us to the following conclusions:

- Extracting raw data from the GPU buffer can be done. The extraction time depends on the level of complexity of the object rather than the number of facets and vertices that exist.

- Raw data extracted from Marching Cubes or hybrid rendering is delivered using Obj and X3D file formats.
- Extracted objects using server based rendering use image stream to deliver the 3D object.
- Loading the extracted file (.obj) which hides the functions and data from users, and displays 3D models on the browser can be done.
- There is a considerable difference when extracting raw data between simple and complex objects, in terms of extraction time, file size and loading time.
- Image streams can be a good solution to clients with low GPU power, but require more storage and bandwidth.

When comparing the three extraction tests, we can conclude the following:

- The loading time for Hemi-sphere with Marching Cubes rendering is almost equivalent to that of Hybrid rendering at 40% resolution.
- The loading time for Hybrid rendering with 80% resolution is 4.4 times higher than that of marching cubes. That is why Hybrid rendering is best used with powerful GPUs.
- The Obj file size loaded with Hybrid rendering at 80% resolution (5227 KB) is considerably smaller compared with that loaded from Marching Cubes rendering (1707 KB). That means we need more extraction time and more GPU power in Hybrid rendering, but lower storage to save Obj files compared to Marching Cubes rendering.

REFERENCES

[1] Y. Jung, J. Behr, T. Drevensek, and S. Wagner, "Declarative 3D Approaches for Distributed Web-Based Scientific Visualization Services," Proceedings of the 1st International Workshop on Declarative 3D for the Web Architecture, 2012.

[2] M. Limper, S. Wanger, C. Stein, Y. Jung, A. Strok, F. IGD, and T. Darmstadt, "Fast Delivery of 3D Web Content: A case Study, Web3D," Proceedings of the 18th International Conference on 3D Web Technology, pp. 11-17, 2013.

[3] M. Rodrigues, M. Kormann, and L. Davison, "A Case Study of 3D Technologies in Higher Education: Scanning the Metalwork Collection of Museums Sheffield and its Implications to Teaching and Learning," International Conference on Information Technology. IEEE, 1-6, 2011.

[4] A. Evans, M. Romeo, A. Bahrehmand, J. Age, and J. Blat, "3D Graphics on the Web: a Survey," Interactive Technology Group, University Pompeu Fabra, Barcelona, Spain. 2014.

[5] H. Lee, G. Lavoue, and F. Dupont, "Rate-distortion Optimization for Progressive Compression of 3D Mesh with Colour Attributes," The Visual Computer, 137-153, 2012.

[6] M. Rezayat, "The Enterprise-Web Portal for Life-cycle Support," J. Computer-aided design, 2000: 32 .85-96.

[7] S. Abrahamson, D. Wallace, N. Senin, and P. Sferro, "Integrated Design in a Servicemarket Place," Computer-Aided Design, 2000.

[8] J. Peng, C. Kim, And C. Jay Kuo, "Technologies for 3D Mesh Compression: A survey," Journal of Visual Communication and Image Representation. Volume 16, issue 6, 2005.

[9] A. Maglo, C. Courbet, C. Alliez., and C. Hudelot, "Prouressive Compression Of Manithld Polyzon Meshes," Computers and Graphics, Shape Modeling International (S1\11) pp. 349-359. 2012.

[10] H. Hoppe, "Progressive Meshes," Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 99-108. 1996.

[11] K. Ramani, A. Agrawat, and M. Babu, "CADDAC: Multi-Client Collaborative Shape Design System with Server-based Geometry Kernel," in Journal of Computing and Information Science in Engineering, volume 3, issue 2, 2003.

[12] H. Grasberger, P. Shirazian, B. Wyvill, and S. Greenberg, "A Dataefficient Collaborative Modelling Method Using Websockets and the Blobtree for Over-the Air Networks," in Proceedings of the 18th International Conference on 3D Web Technology. New York, pp. 29–37, 2013.

[13] X. Bong, Z. Chen, H. Siadati. S. Tople. Saxena, and Z. Liang, "Protecting Sensitive Web Content From client-side vulnerabilities with CRYPTONs," Proceedings of the 2013 ACM S1GSAC Conference on Computer & Communications Security pp. 1311-1324, 2013.

[14] K. Sons, F. Klein, D. Rubinstein, S. Byelozyorov, and P. Slusallek, " XML3D- Interactive 3D graphics for the Web," Proceedings of the 15th International Conference on 3D Technology, pp. 175-184, 2010.

[15] A. Abdallah, O. Fryazinov, V. Adzhiev, and A. Pasko, "3D Web-Based Shape Modelling: Building up an Adaptive Architecture," ACI-11 2014 . The Seventh International Conference on Advances in Computer-Human Interactions. 2014.

[16] R. Cartwright, V. Adzhiev, A. Pasko, Y. Goto, and T. Kunii, "Web-based Shape Modelling with Hyper-Fun," ILIN Computer Graphics and Applications. Vol. 25. pp. 60-69, 2005.

[17] C. Vilbrandt, G. Pasko, A. Pasko, P. Fayolle, T. Vilbrandt, and J. Goodwin, "Cultural Heritage Preservation Using Constructive Shape Modeling," Volume 23, number 1 pp. 25-41, 2004.

[18] O. Fryazinov, A. Pasko, and V. Adzhiev, "An Exact Representation of Polygonal Objects by Differentiable Scalar Fields Based on Binary Space Partitioning," Technical Report "TR-NCCA-2008-03", The National Centre for Computer Animation. Bournemouth University. UK. 2008.

[19] J. Pantaleoni, "VoxelPipe: A Programmable Pipeline for 3D Voxelization," (NVIDIA), in High Performance Graphics, 2011.

[20] M. Sanchez, O. Fryazinov, and A. Pasko, "Efficient Evaluation of Continuous Signed Distance to a Polygonal Mesh," 12 Proceedings of the 28th Spring Conference on Computer Graphics, pp. 101-108, NY, USA, 2012.

[21] K. McHenry, and P. Bajcsy, "An overview of 3D Data Content File Formats and Viewers," National Center for Supercomputing Application. USA. 2008.

[22] D. Wu, D. Rosen, L. Wang, and D. Schaefer, "Cloud-Based Manufacturing: Old Wine in New Bottles? Variety Management in Manufacturing," Proceedings of the 47th CIRP Conference on Manufacturing Systems, 2014.

[23] A. Abdallah, "Securing Online 3D Web-Based Models", ICT and Societal Challenges, LAU Beirut / New York, 2017.