Annual of Konstantin Preslavsky University of Shumen
vol. XXIV C, 2023, pp. 17 – 25, https://doi.org/10.46687/LTZQ6940

STRUCTURAL DESIGN OF 3D MODELS SHARING SYSTEM*

DELYAN H. SARMOV

ABSTRACT: *In the last years a great availability of software and hardware has been noticed for the adaptive manufacturing. The introduction of 3D printing allows to print and offer products that copy successful developments of proven brands. To limit the production of cheap copies of original products, copyright protection technologies are needed. The present material proposes a system for storing and accessing a library of 3D models in which unauthorized distribution of the files is restricted.*

KEYWORDS: *Web-Based, 3D Model, Remote 3D Rendering;*

2020 Math. Subject Classification: *68P20, 68P27*

1 Introduction

Three-dimensional (3D) graphics file formats store information about geometric shapes as text or binary data. Such files are also called vector files. To visualize vector 3D files, calculations are performed in a process called rendering, which can be hardware accelerated. Numerous software products have been developed for processing 3D graphics, such as graphic modeling programs, CAD systems, CNC machine control programs. Last but not least, vector models are also used in 3D printing. In this way, a market is created for ready-made three-dimensional models, which are distributed by various intermediary companies, on specially created platforms, most often Web-based. A few examples are given below:

https://sketchfab.com/3d-models
https://www.turbosquid.com/

*This paper is (partially) supported by Scientific Research Grant РД – 08-125/22.02.2023 of Shumen University

https://free3d.com/
https://www.cgtrader.com/3d-models
https://3dsky.org/3dmodels

Computer graphics is a visual product and therefore it is important that consumers see the product and check its parameters before purchasing it.

Photos and/or video of the model are usually used to allow the user to view it. This way of representation has the disadvantage that a large part of the details and parameters of the model are not represented. There is no way to check the compatibility of multiple objects or how they will look together [1,2].

In the Sketchfab online store example, models are rendered in a Web browser using WebGL [3]. The visualization is realistic and the model can be studied before buying it. Apps have also been developed for the purpose. The main disadvantage is the used models are saved in the cache of the browser or the application before being visualized.

Due to the fact that working with 3D models usually uses libraries that require hardware rendering of the graphics, the files need to physically exist on the device on which they will be used. The file or files containing the model are saved to the potential buyer's device even though he hadn’t decided to pay for the model. This provides opportunities for abuse of authors' rights.

A similar problem exists when distributing images and documents. It is solved using different approaches like adding watermark, rendering in lower resolution, etc [4]. These methods are not suitable for 3D models. Methods have been developed with changing the coordinates, adding or removing points (vertices) in the model. However, when using them, the quality deteriorates and they are not widely used for now.

The problem is even worse with engineering files created using CAD systems. Controlling the distribution of CAD designs is a matter of corporate security and subject to industrial espionage.

2 Commonly used technologies

For Web-based platforms, there are the following approaches for generating visual content.

- Server-side rendering – Template systems are used to generate dynamic content from pre-recorded software on the server. Examples of such are Twig, Smarty, Blade, Django template language.

- Client-side rendering – Software stored on the server (backend) works as a service provider. Examples of such systems are Django, Laravel, Flask Java Spring. The programming languages used are Python, PHP, Java, Perl, Ruby, etc. The interface is generated and visualized in the Web browser by applications written in JavaScript or Typescript called frontend. Examples of systems with which Web frontend is developed are React, Vue, Angular. This effectively uses the browser cache, optimizes network traffic, offloads the server and breaks down the development process. Communication between frontend and backend is most often done using Json or XML standards. It is possible that the backend application is used to serve several frontend programs built for the Web, mobile or desktop platform.

Other, less frequently used approaches are:

- Edge-side rendering, where runtime proxy substitution is done between the server and the client.

- Outside server rendering, where static pages generated on the side system are saved on the server.

- Distributed Rendering – Content is generated on multiple servers at runtime.

The current trends in recent years are towards using client-side rendering.

Server-side rendering in the backend of a 3D model requires hardware support of the 3D rendering libraries and makes communication with the user difficult. Each time the model is operated, a request must be sent to the server and the object must be rendered anew.

When using Client-side rendering, the model is saved on the client machine before it is used. This creates problems with protecting the rights of the creator of the model.

3 Technology selection

This article proposes a method for working with 3D models located on a remote system. Software has been developed and used for other purposes that are useful for the current system. **Figure 1** presents a generalized diagram of a Web application.

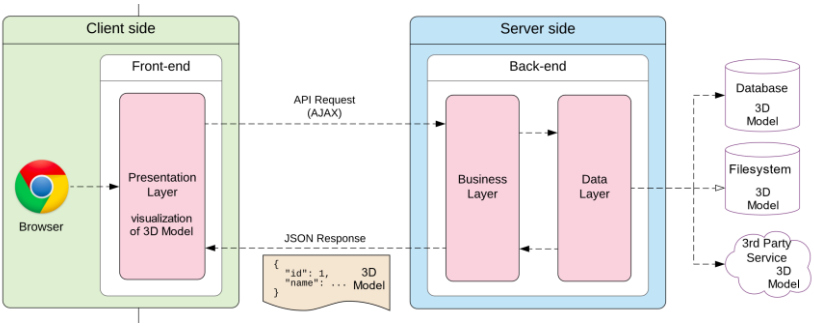

Figure 1. Single page Web Application

As seen in the diagram, models are stored on a server in a database, as separate files from the file system, or can be retrieved from another server using an asynchronous request. They are subsequently transported to the user device using a JSON response. The rendering process is done entirely on the client system.

The following diagram Figure 2 shows separately the rendering process of a standard Linux system during a remote session.

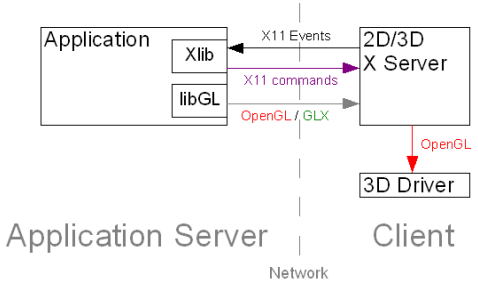

Figure 2. Indirect rendering with X server

In the specified mode of operation, called "indirect rendering", OpenGL commands are encapsulated in the X11 protocol stream and sent for rendering to a remote system. GLX is used to manage the window context. Such a system is useful if the data being rendered is small, static, rendering lists are used, the network has got low latency and high bandwidth. Some disadvantages have been noticed:

- When using dynamically generated lists, the rendering of each vertex in the scene must be transmitted over the network as a packet to the server. Network latency is a limiting factor.

- Textures that are generated every frame also need to be sent to the server, which requires high bandwidth.

- There may be a lack of support or correct configuration of the OpenGL client engine or some of the extensions that are used.

To solve the mentioned problems, the following solutions have been proposed so far.

3.1. A separate application called "screen scraper" is used on the server, which monitors for events in the X11 server that may cause some of the rendered pixels to change. When such an event occurs, data is read from the display buffer and sent to the client. There are products developed using this approach. The disadvantages are:

- Additional software required to monitor for OpenGL scene-changing events.

- Additional software is required when sharing the application.

3.2. Split rendering using a VNC proxy server. OpenGL commands are sent to the proxy server, which splits them and forwards them for rendering in the 3D driver, while input events are forwarded to the X server. The result is a series of two-dimensional images that are sent to connected users. Compression can be applied to optimize traffic [5,6,7].

3.3. Split rendering in process. This method has been adopted and implemented by the VirtualGL team. A library called "GLX interposer" has been developed that can be imported into the model rendering process. The library intercepts GLX commands and establishes the appropriate server contexts. A separate "VGL Transport" library is used to communicate with the user. The downside is that this library does not support multi-user collaboration. The advantage is that it achieves the highest processing speed compared to other methods.

To summarize:

- If it is not necessary to maintain functionality such as "teamwork", the optimal choice is VirtualGL with GLX interposer and VGL Transport.

- If the user has to work in a team, it is necessary to use VirtualGL with a proxy server. There is a "TurboVNC" project that claims to be optimal for working with VirtualGL [8].

4 Structural diagram of system architecture

Figure 3**Error! Reference source not found.** shows a general diagram of an architecture providing flexibility in loading a graphical model.

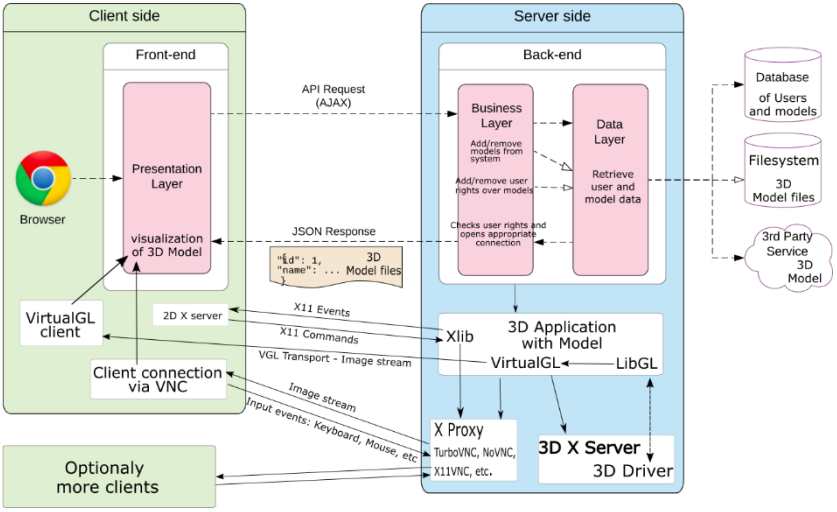

Figure 3. System architecture with 3 ways of processing 3D models

In the specified system, the user logs in and, depending on his rights to the models, 3 ways of processing requests are possible:

4.1. User has rights over the file. The model is retrieved and sent to the frontend for processing.

4.2. User has no rights over the file. The model is retrieved and loaded into an application running in a separate session on the server. Rendering is done with VirtualGL. The rendering results (sequence of images) are sent to the client using VGL transport over a TCP port. In order to maintain communication, it is necessary that the 2D X server infrastructure is installed on the client's computer.

4.3. The user does not have rights to the file, there is no 2D X server infrastructure, or support for multiple users is required. The model is fetched, rendered in a full-screen application in a separate server session. The resulting image sequence is sent via a proxy to the client.

Intermediate compression of information is possible in each of the three variants.

5 Conclusion

5.1. To prevent the uncontrolled distribution of 3D models, it is necessary to develop methods for their visualization without copying the files to the user's device.

5.2. There are proven technologies such as VirtualGL and VPN servers that can be used to create the necessary infrastructure.

5.3. At the moment, there are no complex solutions combining all the components of the proposed system.

5.4. Further studies of the performance of the system when running different GPU optimized virtual machines are needed.

REFERENCES:

[1] A. Evans, M. Romeo, A. Bahrehmand, J. Agenjo, J. Blat, “3D graphics on the web: A survey”, Computers & Graphics, Volume 41, 2014, Pages 43-61, ISSN 0097-8493,

[2] Patel, V., Prabhu, G., Rita, E., & Karani, R. (2015). “3D visualization of products for online shopping”, International Journal of Computer Applications, 132(5), 35-39.

[3] Scopigno, R., Callieri, M., Dellepiane, M., Ponchio, F., & Potenziani, M. (2017). “Delivering and using 3D models on the web: are we ready?”, Virtual Archaeology Review, 8(17), 1–9

[4] Begum M, Uddin MS. “Digital Image Watermarking Techniques: A Review” Information. 2020; 11(2):110.

[5] Stegmaier, Simon, Marcelo Magallón, and Thomas Ertl. "A generic solution for hardware-accelerated remote visualization.", VisSym. 2002.

[6] M. D. Carroll, I. Hadžić and W. A. Katsak, "3D rendering in the cloud,", Bell Labs Technical Journal, vol. 17, no. 2, pp. 55-66, Sept. 2012, doi: 10.1002/bltj.21544.

[7] Hadžic, M. D. Carroll and H. C. Woithe, "Low-Level Frame-Buffer Scraping for GPUs in the Cloud," 2016 IEEE International Symposium on Multimedia (ISM), San Jose, CA, USA, 2016, pp. 529-532, doi: 10.1109/ISM.2016.0116.

[8] "A Study of the Performance of VirtualGL 2.1 and TurboVNC 0.4", Version 1d, 8/20/2008 -- The VirtualGL Project.

Delyan H. Sarmov
Faculty of Mathematics and Informatics
Konstantin Preslavsky University of Shumen
9700 Shumen, Bulgaria
E-mail: d.sarmov@shu.bg