Approach to cross-platform 3d-models binary format development

M.V. Alpatova¹, D.G. Demidov², A.I. Vinokur³ and V.L. Krupenin⁴

¹ Postgraduate Student, Assistant of the IIT Department, Moscow Polytechnic University, Moscow, Russia
² CES, Associate Professor of the IIT Department, Moscow Polytechnic University, Moscow, Russia
³ D.Eng.Sc., Director of the Institute of Applied Information Technologies, Moscow Polytechnic University, Moscow, Russia
⁴ D.Eng.Sc., Chief Research Scientist, Institute of Problems of Mechanical Engineering, Russian Academy of Sciences, Moscow, Russia

vrisk.serket@gmail.com

Abstract. The article presents the process of analysis, processing and compression of 3D objects for their further transfer to the end user over the network. The necessity to optimize the disk storage occupied by 3D textured models and to convert data into a format compatible with any robotic systems is a subject of the discussion. Comparison of a developed binary format with counterparts is resulted.

Keywords: Artificial vision, robotics, 3D content, compression, augmented reality

1. Introduction
In the age of the development of computer vision and machine learning technologies, the problem of transmitting information in the form of 3D content is becoming increasingly important. In 2012, the very notion of a Google Glass device demonstrated the urgent need to develop innovative ways of the media content streaming [1]. AR technology is not the only one where new content format requirements need to be implemented.

While considering the robotics field it should be mentioned that the application of machine vision in manufacturing significantly contributes to output quality control. By 3D-scanning of the manufactured samples and their subsequent comparison with the reference model, it is possible to identify defected goods as well as to show employees problem areas requiring correction by using of augmented reality glasses [2].

2. Research objective
The developer, when creating a ML model to detect objects by the computer, may need to use a special set of data and parameters that describe the properties of the researchable object. Such other information cannot be included in third-party formats that were not developed therefor.

The way out is to create a native binary format of 3D models having a data stored thereon, both about the geometric features of the object and all texture maps, as well as the necessary additional information about the object. The use of lossless compression methods will allow to pack available geometry data together with images with the lowest cost on the occupied disk space by the output file.
3. Stages of a binary model preparing
At the first stage, the developer of 3D objects prepares a 3D object in DAE format [3] and uploads it to the server along with an archive of texture maps. The automation process, that is a processing of model information and converting it to the best format, begins from then onward.

The second step is to create an uncompressed binary file containing geometry and texture maps data. The final step is to compress the model. To reduce the dimensions of the transmitted object, various compression methods can be used, however, to guarantee a successful file unpacking on the end device, applying wide used compression algorithms shall be necessary. DEFLATE, that is a combination of the Huffman Coding and LZ77 algorithms, is one of these algorithms also being used in the prevalence archive format - GZIP [4].

4. Testing the effectiveness of the binary format of 3D models
By conducting a comparative analysis of the original and output file size we will check the practicability of applying the described 3D packing method. To compare the quality of the resulting format (named BFM - Binary Format Model in the tables and graphs below), similar binary formats used by Apple (USDZ) [5] and Google (GLB) [6] in browser applications with augmented reality have been considered.

First of all, let's compare the compression ratio of the model geometry without textures. 12 DAE format models with a total number of triangle meshes between 500 and 300,000 units have been taken to perform such an experiment. Figure 1 shows three dependency graphs of the output file size on the source model’s geometry complexity.

Figure. 1 Graph of the destination file size dependence on the model’s geometry complexity

The exponential dependence of the destination file size on the data amount on the 3D geometric features, common for all three studied formats, is evident from the graph above.

Let us consider a texture map compression. For this case 10 archive files of different sizes from 200 to 7000 KB have been taken. There are 4 types of textures in each archive:
- Diffuse map
- Normal map
- Metallic map
- Roughness map

Figure 2 shows the difference between the original archive size and its compressed version. All samples have been sorted in ascending order of their size on the disk (Table 1).
### Table 1. Texture input and output data

| Sample number | Archive size with textures, KB | Compressed Archive Size, KB |
|---------------|--------------------------------|-----------------------------|
| 1             | 200                            | 2793                        |
| 2             | 670                            | 1478                        |
| 3             | 2480                           | 3321                        |
| 4             | 3070                           | 3153                        |
| 5             | 3440                           | 1806                        |
| 6             | 3530                           | 372                         |
| 7             | 6080                           | 1736                        |
| 8             | 6530                           | 1900                        |
| 9             | 6540                           | 1499                        |
| 10            | 6950                           | 1011                        |

Figure. 2 Data on input and output texture size

The data above shows the advantage of compressing texture maps with a size of more than 3 MB, otherwise, the information to be encoded may take up more space than the initial data volume. The archive size has reduced on average by 2 times, and maximum by 7 times compared to the native file size.

Table 2 shows all the parameters of the source files and the total sizes of the models’ binary format, which includes both geometry and texture maps. In the graph below (Fig. 3), it can be noticed that the dependence of the output file size repeats the function diagram of the compressed sample demonstrated on Fig. 2. However, there is a different amplitude of dependencies in the considered three formats. Moreover, in our format this amplitude is the smallest, and the native size is stably lower than the counterparts.
Table 2. Data on the input and output sizes of models with textures

| Sample number | DAE size, KB | Textures size, KB | Total size, KB | USDZ size, KB | GLB size, KB | BFM size, KB |
|---------------|-------------|------------------|---------------|--------------|--------------|--------------|
| 1             | 65          | 6540             | 6605          | 2502         | 3203         | 2793         |
| 2             | 138         | 3530             | 3668          | 1310         | 1568         | 1478         |
| 3             | 737         | 6950             | 7687          | 5309         | 7704         | 3321         |
| 4             | 1642        | 6080             | 7722          | 4521         | 10633        | 3153         |
| 5             | 2100        | 6530             | 8630          | 5107         | 3461         | 1806         |
| 6             | 4125        | 670              | 4795          | 3805         | 2789         | 372          |
| 7             | 7436        | 3440             | 10876         | 6656         | 10280        | 1736         |
| 8             | 8185        | 3070             | 11255         | 5996         | 13936        | 1900         |
| 9             | 9567        | 200              | 9767          | 5795         | 15302        | 1499         |
| 10            | 12298       | 2480             | 14778         | 6844         | 8699         | 1011         |

Figure 3. Graphs of the dependencies of the source archive size with data on the model with the occupied volume of the final binary file

The graph above demonstrates that the compression of the developed BFM format is at average 5 and 3.5 times better than the GLB and USDZ file formats correspondingly.

5. Conclusion

The most hot-spot issue in the age of Internet technologies and online services’ dynamic development is the compression of transmitted content and the security of its transmission. The task to develop a format for storing 3D model data that would meet modern requirements has been assigned.

To solve the problem, an algorithm for processing, compressing and packing data on 3D contents’ geometry and textures has been developed being applied in practice afterwards. The comparative analysis proves the effectiveness of the specified algorithm and its response to the requirements and todays’ expectations.

References

[1] Jeff Tang Beginning Google Glass Development. – 1st edition Apress, 2014. - p. 368
[2] Pietikäinen M.K. Texture Analysis in Machine Vision. - Singapore: World Scientific Publishing Co Pte Ltd, 2000. - p. 280
[3] Remi Arnaud, Mark C. Barnes COLLADA: Sailing the Gulf of 3D Digital Content Creation. 1st edition AK Peters, 2006. - p. 250
[4] Serbin Ya. How GZIP Compression Works // Habr URL: https://habr.com/ru/post/221849 (reference date: 25.09.19)
[5] AR Quick Look // Apple Developer URL: https://developer.apple.com/augmented-reality/quick-look (reference date: 25.09.2019)

[6] Import and preview 3D assets // ARCore Develop URL:
https://developers.google.com/ar/develop/java/sceneform/import-assets (reference date: 25.09.2019)