ANÁLISE DE ESTRUTURAS DE APRESENTAÇÃO DE DADOS PARA SISTEMAS COMPUTACIONAIS PARA TRABALHAR COM REDES POLIGONAIS NAS TAREFAS DE PREPARAÇÃO TECNOLÓGICA DE MANUFATURA ADITIVA

POLYGONAL MESHES DATA STRUCTURE ANALYSIS USED FOR COMPUTATION OF THE PARAMETERS DEFINING ADDITIVE PRODUCTION PROCESS FOR DIFFERENT ADDITIVE MANUFACTURING TECHNOLOGIES

АНАЛИЗ СТРУКТУР ПРЕДСТАВЛЕНИЯ ДАННЫХ ДЛЯ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ ПО РАБОТЕ С ПОЛИГОНАЛЬНЫМИ СЕТКАМИ В ЗАДАЧАХ ТЕХНОЛОГИЧЕСКОЙ ПОДГОТОВКИ АДДИТИВНОГО ПРОИЗВОДСТВА

RIPETSKIY, Andrey V.[1*]

[1] Moscow Aviation Institute (National Research University), Department of Engineering Graphics, 4 Volokolamskoe shosse, zip code 125993, Moscow – Russian Federation
(phone: +74991584333)

** Corresponding author*
e-mail: a.ripetskiy@mail.ru

Received 25 June 2018; received in revised form 22 November 2018; accepted 03 December 2018

RESUMO

O artigo discute uma classe de problemas geométricos, que incluem características associadas à representação geométrica do modelo, típica de tecnologias de produção que requerem representação em camadas do modelo eletrônico, independentemente da tecnologia de produção específica. Por exemplo, identificar parâmetros para avaliar a qualidade da geometria de um modelo quanto à conformidade com os requisitos relevantes para tecnologias de manufatura aditiva. As questões tecnológicas que refletem as características de várias tecnologias de manufatura aditiva também são analisadas. São consideradas as características da apresentação de cálculos de dados de desenvolvimento de processos, dependendo da tecnologia de produção. A organização da estrutura de dados interna para otimizar a velocidade do cálculo de representação em camadas com o uso de modelo multitarefa é mostrada.

Palavras-chave: *tecnologias aditivas, síntese em camadas, modelo geométrico, triangulação, representação em camadas.*

ABSTRACT

In this paper, we describe a class of geometric problems, which can be identified on model geometry allows assessing geometry quality to comply with the current requirements of the additive production technologies (for example, identifying parameters for assessing the quality of a model's geometry for compliance with the requirements relevant to additive manufacturing technologies). Technological issues reflecting the features of various technologies of additive manufacturing were analyzed. The features of the presentation of data calculations of technological training depending on the production technology were considered. Influence of the proper organization internal data structure to optimize the calculation the speed of layered representation using a multithreaded model is shown.

Keywords: *layered synthesis, geometric model, triangulation, layered representation.*

DOI: 10.52571/PTQ.SI1.2018.302_PeriodicoES.pdf

АННОТАЦИЯ

В работе рассматривается класс геометрических задач, к которым можно отнести особенности, связанные с геометрическим представлением модели, характерные для технологий производства требующих послойного представления электронной модели, независимо от конкретной технологии производства (например, выявление параметров оценки качества геометрии модели на предмет соответствия требованиям актуальным для технологий аддитивного производства). Так же проанализированы технологические вопросы, отражающие особенности различных технологий аддитивного производства. Рассмотрены особенности представления данных расчетов технологической подготовки в зависимости от технологии производства. Показана организация внутренней структуры данных для оптимизации скорости расчета послойного представления с использованием многопоточной модели.

Ключевые слова: *послойный синтез, геометрическая модель, триангуляция, послойное представление.*

INTRODUCTION

The current standard for defining geometry used in the technological systems for additive production is the polygonal data representation. Such a representation is supported in all additive production technologies software solutions (Uriondo *et al.*, 2015; Gibson *et al.*, 2010; Formalev *et al.*, 2018).

In the polygonal representation, the geometry of the virtual model is represented as a polygonal mesh, which is a collection of edges, vertices, and polygons. Model vertices are joined by edges, and polygons are considered as sequences of edges or vertices. The basic element of the three-dimensional model is a triangle, which is called the "Facet" and is represented in Figure 1.

The polygonal mesh elements are the following:

1. The vertex – position of the point in the given coordinate system (CS). Most often the coordinates are indicated in Cartesian CS.
2. An edge – connection between two vertices.
3. The facet – closed set of edges.

There are several possibilities to describe a polygonal mesh representation, each of which has its own merits and demerits.

The first one is the vertex representation, which describes the object in the form of a set of vertices connected to each other (Figure 2). This method is the simplest and takes up the least amount of space in the computer's memory.

The second one is the "List of faces" (Figure 3). In this case, the object is described set of faces and a set of vertices. A distinctive feature of this method is that information about all faces is already entered in the file. This method simplifies the process of determining the orientation of the faces in a space. The orientation of the facets is determined by analyzing the sequence of the vertex facets orientation or using a predetermined normal vector orientation of the facet. This method increased computer memory usage.

Both described methods are commonly used in file formats supporting three-dimensional polygon mesh model representation software for additive production (Savio *et al.*, 2018).

The first file format is STL. This is an open source file format developed by 3D Systems. It is widely used in stereolithography technology known as rapid prototyping were a solid physical model is produced by exposing layer by layer of liquid photopolymer resin hardens by the laser.

The structure of the STL format depends on the type of data representation:

- Binary;
- Text – ASCII (Figure 4).

The geometry of the model stored in the STL format has a "Vertex representation" of the surface mesh, which is approximated by triangular facets. The file first describes the orientation of the normal facet unit – facet normal # # #, then it describes the coordinates of the facets vertices – vertex # # #. Each facet is

described by a triad of vertices, which is set counter-clockwise if the facet is oriented perpendicular to the line of sight. This format of data storage is currently the most widespread for transferring information about the geometry of the model with the aim of converting it into a code for a CNC machine using special software.

The second most common file format is the Wavefront OBJ (Figures 5-7). The structure of the file format is presented in text form. This format uses the representation of the surface grid in the form of a "Facet List". Initially, the vertices are described – "v # # #" using coordinates along the x, y, and z-axes, respectively. Textual coordinates "vt # # #" – one of the distinguishing features of the OBJ format, is given by the coordinates (u, v, [w]). This allows to transfer information about the location of textures, between different graphics editors, and also in some cases to print several types of material, depending on the assigned texture. The coordinates of the textures are specified after the vertex list and are optional. Unlike the STL format, the values of the normal units "vn # # #" are set not before the triad of vertexes defining the facet, but after the list describing the vertices and coordinates of the textures. Also this format allows you to specify the vertex parameters in the "vp # # #" space in the format [u, [v], [w]). Using this operator allows determining how many coordinates are required to describe the geometry of a freeform. For curves, 1D reference points (u) are required. The description of the parameters [u, v] is required to describe the surface. To specify the cutting surfaces, the parameter [w] is additionally indicated. And according to the structural representation "List of facets", the final list is the definition of the sides "f", in which the correspondence of each facet in the format [v1 / vt1 / vn1 v2 / vt2 / vn2 v3 / vt3 / vn3 ...] is made. The value of the parameters v1, v2, v3 is indicated according to the position in the corresponding item in the list. For example, if f 1 // 1 3 // 1 5 // 1 is specified, this means that the vertex coordinates for the face are taken from the vertices that have the sequence number 1, 3, and 5 in the vertex list, respectively. A significant advantage of this format is that it is possible to create n-gonal faces and the definition of vertices is performed not by successive triads, as in the STL format, but by parameters of the "f" list. But, as mentioned in the description of the "List of facets" view, this complicates the structure of the file format, increasing the load on the software that parses values and has a larger file size compared to the STL format.

Ultimately, the format for presenting data in a particular system is chosen based on the dimensions of the tasks being solved, the initial estimate of the dimension of the original data, and the time it is supposed to spend on development.

The efficiency of the use of computing resources by the software module is determined by a variety of factors. One of these is the structure of internal data representations. The amount of memory required depends both on the internal structure and on the types used (Anamova *et al.*, 2016; Ripetskiy *et al.*, 2016; Lurie *et al.*, 2017). For example, a polygon mesh downloaded from an STL file and can be stored in the program memory as is, in the form of a list of triangles, can be converted into a vertex list, and the polygon meshes itself into a list of indexes.

Even more, options can be when organizing data storage for rendering using graphics libraries.

For software in general, the efficiency of data usage becomes really significant when this efficiency becomes the quality of the software product use (Schmid and Levy, 2013). For example, inefficient data storage can cause memory overruns and lead to overflow of the address space allocated to the process (Gibson, 2002). So, a more efficient application from the point of view of resource usage will be able to process more complex geometry or to calculate the model with higher accuracy.

The use of resources by the application can be easily traced back to the standard operating system tools, for example, the Task Manager Tool. We will experiment with the use of resources by various applications for loading models. Let's take the same model and load it with different programs.

To do this, let's consider two applications – the graphical shell for the Slic3r slicer and the ATSS Glicer program. We will load into the programs several instances of the same model and look at the amount of allocated memory.

The graph shows that the amount of allocated memory grows in different ways. This is due to the peculiarity of data organization inside the application. In both cases, the memory volume grows faster than the data representing the polygon mesh itself. This is due to the fact

that in addition to the polygon mesh data, models for rendering are required to store normals as well (Afanasyev *et al.*, 2018).

Nevertheless, we can say that, while other things are equal, the application on Slic3r will quickly go beyond the bounds of the address space of the ATSS Glicer application, which means that it can process a smaller amount of initial data.

The screenshots of the camera layout, with four objects from the test, are shown in Figure 8.

MATERIALS AND METHODS

2.1. Features of data representation of calculations of technological preparation depending on production technology

In the process of technological preparation in the software, the electronic model passes through a series of computational algorithms (Pandey *et al.*, 2007). The results of their work are stored in internal data structures and represent data for calculating output information – a set of control commands.

These internal data structures may differ depending on the printing technology.

2.1. Features of data representation for selective sintering and surfacing technology in the part of forming a layer representation

For most 3D printing technologies, the internal representation of data in the form of many contours separated by layers (Gibson, 2002) is used. Such a representation does not take up a lot of memory resources and is suitable for further calculations of control commands using closed-loop hatching algorithms and equidistant offsets. Calculation methods are algorithms for the cross-section of triangles by a plane. The selection of the algorithm of hatching and their patterns, in combination with technological modes, can be an effective tool for improving the quality of both the synthesis process and the quality of the part (Ripetskiy *et al.*, 2018a; Ripetskiy *et al.*, 2018b). To analyze the structures of these additive production processes for selective laser sintering of polymeric materials, industrial installations Russian SLS and EOS were used.

2.3. Features of the technology of jet printing of products from composite ceramics in the part of the formation of a layered representation

In 3DP (BinderJet) jet printing technologies, the filling of the zone is structurally realized by parallel movements of the printing element, so all the filling segments must be parallel. This feature of jet printing imposes its limitations in the formation of control commands – there are no bypasses of perimeter zones and their equidistant offsets, and filling of the zone occurs by unidirectional hatching.

Therefore, to construct a layered representation and its subsequent filling, it is not necessary to construct closed contours. Calculation of the segments of displacement and filling of the working element can be constructed by the method of tracing based on the solution of the problem of finding the point of intersection of the straight line with the plane given by the three vertices of the triangle and determining the point belonging to this triangle.

Thus, in order to calculate control commands for jet printing technology, there is no need for the requirements for the closeness of the contours obtained by the section of the geometric model by the plane. Instead, it is necessary that the geometric representation contain the correct order of the front facets and back facets in all directions of the tracing (Rabinskiy *et al.*, 2017a; Rabinskiy *et al.*, 2017b).

RESULTS AND DISCUSSION:

3.1. Algorithms for calculating the internal layer representation

At the heart of any additive production process (Gibson *et al.*, 2015), for example, laser sintering or fusing is a file with control commands to be executed on the printer. The command file is compiled according to the geometric representation of the model at the stage of technological preparation of additive production and contains instructions for moving the movable part of the device – the head with the nozzle in the fusing devices or laser beam in sintering devices. The location of the command files during the manufacturing process is shown in Figure 9.

Most printing machines use the G-Code format as the standard for the command file format. The GCode command file is a text file in a single-byte encoding and consists of tuning commands and a set of consecutive commands of idle (G0) and working (G1) pass. Example of the command of the working stroke: G1 X48.700

Y1.049 E2.31026 F1350.000. The sequence of commands of the idle and working stroke executed by the printing device, allow growing in the end the detail of the required geometric shape (Denlinger *et al*., 2014; King *et al*., 2015). The geometric model (a), the layered representation (b), and the visual representation of the command file (c) on the basis of the working pass commands are shown in Figure 10.

An example of listing a command file in G-Code format is shown in Figure 11.

The initial data for building command files is the layered representation of the model, obtained as a result of the slice of the geometric representation of the model. The layer can be represented both in a parametric form as a set of contours, approximated by broken lines and in a raster representation in the form of flat files in BMP format. Based on the technological peculiarities of any additive production (Pandey *et al*., 2007b), the command representation is divided into sections perimeter and filling (infill). This need is determined by the requirements for the smoothness of the edges of the final part and the options for filling internal volumes (Rabinskiy *et al*., 2017c). Algorithms compose control commands differ for different sections of the executable file. So, for the perimeter section of the command file, the task of continuous traversal of closure contours is solved. For the infill section, recursive algorithms for filling closed areas are used.

3.2. Organization of internal data structure to optimize the speed of layered representation calculation using a multithreaded model

Composing a layer-by-layer representation of a geometric model is a very resource-intensive task that requires sufficient hardware computational resources. The resource requirement increases with the accuracy of the layered representation, which depends on the number of layers, which leads to an increase in processor time.

Because the majority of user computing systems in the industry is built on the basis of OCMicrosoftWINDOWS, then all subsequent issues of parallel computing technology will be considered on this platform. We will consider performance issues using the example of parallelization of layer-by-layer calculations based on the initial STL representation of the geometric model. The input data is a surface triangular grid, in which the output requires an ordered set of layers in the form of flat raster files.

The goal is to design an internal application architecture that can parallelize the calculations within the single address space of one process and several calculation flows. The following tasks should be solved:

- launching and maintaining the required number of calculation threads to efficiently load the processor;

- access to model data from calculation flows;

- recording the layer data to the file in sequence for each layer;

- scaling the system by the number of computing resources and cores available at the moment in the system

The tasks are solved by the architecture presented in Figure 12.

3.3. Main features and practical implementation of parallelization of constructing a layer-by-layer representation of the model in many nuclear systems using the example of the CCI PC

The thread manager works from the main thread of the process and can run as many threads to process as there are cores physically contained in the processor. The model in STL lies in the address space of the process and is available to all threads without separate access rights. The layer as a result of the calculation is taken from the calculation flow by the thread manager and is written to the file.

The considered architecture of data organization and parallelization of calculations in the formation of a layered representation is realized practically in software components of technological preparation of production (PC CCI). Below is an example of calculating a layer-by-layer representation of a model with an overall size of 15 mm in three axes with an accuracy of 100 μm. The model and the sampling layer are shown in Figure 13.

The calculation was carried out on a dual-core AMD Athlon 2X Dual processor. The dependence of the calculation time on the size of the thread pool and the CPU core load diagram is given in Table 1.

Based on the dependence of the calculation time on the number of calculated flows and the CPU load diagrams, it can be

concluded that the proposed architecture allows parallelizing the construction of the layered representation and effectively loading the computing capacity of the computer. The number of calculation threads must equal the number of processor cores. Further increase in the number of flows is impractical; because gives a slight increase in kernel loads while reducing efficiency by increasing the tasks of managing the flows by the operating system.

Dynamic allocation of pool resources allows scaling the solution based on the number of available processor cores currently in the system.

CONCLUSIONS:

The additive production process consists a lot of computational sub-processes and depends on a large number of parameters, such as quality of the initial data, required technological modes, accuracy, computational resources used to calculate, and quality of the calculation results. Errors that can occur due to technologically incorrect prepared modes and incorrect initial data can lead to defects in the final products.

The work outlines the main problems related to the tasks of preparation technologically correctly prepared modes for additive production taking into account control of the initial data.

The most significant results of this study are understanding the requirements for computing resources and the data structure defining the production constraints for additive technologies. The dimensions of the volume of the chamber for production are increasing, the power consumption is also getting higher – these factors need to be taken into account during stages of the technological preparation entire additive production process.

The data analysis described in this article can be used for algorithms development that automatized technological preparation of an entire additive production process and testing on various computer systems supporting additive production hardware.

ACKNOWLEDGEMENTS:

The work was carried out with the financial support of the state project of the Ministry of Education and Science (project code 2.9219.2017/8.9).

REFERENCES:

1. Afanasyev, S.A., Afanasyev, M.S., Feshchenko, V.S., Lvov, S.A., Zhukov, A.O. *Periódico Tchê Química*, **2018**, *15(30)*, 687-696.
2. Anamova, R.R., Zelenov, S.V., Kuprikov, M.U., Ripetskiy, A.V. *IOP Conference Series: Materials Science and Engineering*, **2016**, *140(1)*, 012003
3. Denlinger, E.R., Irwin, J., Michaleris, P. *Journal of Manufacturing Science and Engineering*, **2014**, *136(6)*, 61007.
4. Formalev, V.F., Kolesnik, S.A., Kuznetsova, E.L. *High Temperature*, **2018**, *56(3)*, 393-397.
5. Gibson, I. *Software solutions for rapid prototyping*, London: Professional Engineering Publishing Limited, **2002**.
6. Gibson, I., Rosen, D., Stucker, B. *Additive manufacturing technologies – 3D printing, rapid prototyping, and direct digital manufacturing*, New York: Springer New York, **2015**.
7. Gibson, I., Rosen, D.W., Stucker, B. Development of additive manufacturing technologies, *Additive Manufacturing Technologies: Rapid Prototyping to Direct Digital Manufacturing,* Berlin/Heidelrberg: Springer Science+ Business Media, **2010**.
8. King, W.E., Anderson, A.T., Ferencz, R.M., Hodge, N.E., Kamath, C., Khairallah, S.A., Rubenchik, A.M. *Applied Physics Reviews*, **2015**, *2(4)*, 41304.
9. Lurie, S.A., Solyaev, Y.O., Lizunova, D.V., Bouznik, V.M., Menshykov, O. *International Journal of Heat and Mass Transfer*, **2017**, *109*, 511-519
10. Pandey, P.M., Venkata Reddy, N., Dhande, S.G. *ICAMT 2004 (Malaysia) & CCAMT 2004 (India) Special Issue*, **2007**, *185(1-3)*, 125-131.
11. Pandey, P.M., Venkata Reddy, N., Dhande, S.G. *Processing Technology*, **2007**, *185(1-3)*, 125-131.
12. Rabinskiy, L.N., Ripetskiy, A.V., Zelenov, S.V., Kuznetsova, E.L. *International Journal of Pure and Applied Mathematics,* **2017a**, *116(3)*, 789-797.

13. Rabinskiy, L.N., Ripetskiy, A.V., Zelenov, S.V., Kuznetsova, E.L. *Journal of Industrial Pollution Control,* **2017b**, *33(1),* 1178-1183

14. Rabinskiy, L.N., Sitnikov, S.A., Pogodin, V.A., Ripetskiy, A.A., Solyaev, Y.O. *Solid State Phenomena*, **2017c**, *269*, 37-50.

15. Ripetskiy, A., Vassilyev, S., Zelenov, S., Kuznetsova, E. *Key Engineering Materials*, **2018a**, *771*, 97-102.

16. Ripetskiy, A., Zelenov, S. Kuznetsova, E., Rabinskiy, L. *Key Engineering Materials*, **2018b**, *771*, 91-96.

17. Ripetskiy, A.V., Zelenov, S.V., Vucinic, D., Rabinskiy, L.N., Kuznetsova, E.L. *International Journal of Pure and Applied Mathematics*, **2016**, *111(2)*, 343-355.

18. Savio, G., Rosso, S., Meneghello, R., Concheri, G. *Appl. Bionics. Biomech*, **2018**, *2018*, 1654782.

19. Schmid, M., Levy, G. Quality management and estimation of quality costs for additive manufacturing with SLS, *Fraunhofer Direct Digital Manufacturing Conference*, Zürich: ETH-Zürich, **2013**.

20. Uriondo, A., Esperon-Miguez, M., Perinpanayagam, S. *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, **2015**, *229(11)*, 2132-2147.

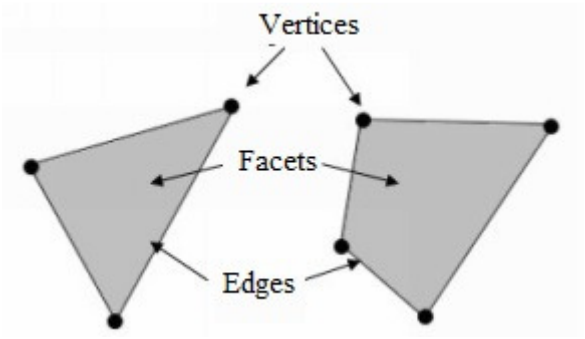


Figure 1. *Element of the polygonal mesh*

List of vertices

v0	0,0,0	v1 v5 v4 v3 v9
v1	1,0,0	v2 v6 v5 v0 v9
v2	1,1,0	v3 v7 v6 v1 v9
v3	0,1,0	v2 v6 v7 v4 v9
v4	0,0,1	v5 v0 v3 v7 v8
v5	1,0,1	v6 v1 v0 v4 v8
v6	1,1,1	v7 v2 v1 v5 v8
v7	0,1,1	v4 v3 v2 v6 v8
v8	.5,.5,0	v5 v6 v7 v8
v9	.5,.5,1	v0 v1 v2 v3

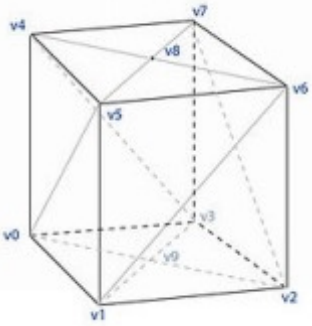


Figure 2. *Vertex grid representation*

List of facets

List of facets

f0	v0 v4 v5
f1	v0 v5 v1
f2	v1 v5 v6
f3	v1 v6 v2
f4	v2 v6 v7
f5	v2 v7 v3
f6	v3 v7 v4
f7	v3 v4 v0
f8	v8 v5 v4
f9	v8 v6 v5
f10	v8 v7 v6
f11	v8 v4 v7
f12	v9 v5 v4
f13	v9 v6 v5
f14	v9 v7 v6
f15	v9 v4 v7

List of vertices

v0	0,0,0	f0 f1 f12 f15 f7
v1	1,0,0	f2 f3 f13 f12 f1
v2	1,1,0	f4 f5 f14 f13 f3
v3	0,1,0	f6 f7 f15 f14 f5
v4	0,0,1	f6 f7 f0 f8 f11
v5	1,0,1	f0 f1 f2 f9 f8
v6	1,1,1	f2 f3 f4 f10 f9
v7	0,1,1	f4 f5 f6 f11 f10
v8	.5,.5,0	f8 f9 f10 f11
v9	.5,.5,1	f12 13 14 15

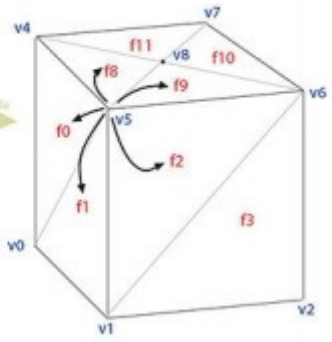
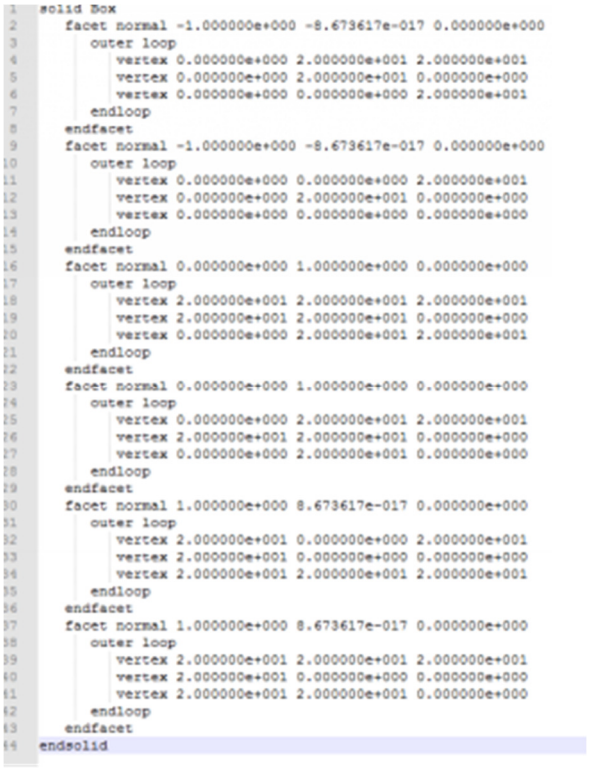


Figure 3. *Graphical representation of the "List of faces" form*



```
solid Box
   facet normal -1.000000e+000 -8.673617e-017 0.000000e+000
      outer loop
         vertex 0.000000e+000 2.000000e+001 2.000000e+001
         vertex 0.000000e+000 2.000000e+001 0.000000e+000
         vertex 0.000000e+000 0.000000e+000 2.000000e+001
      endloop
   endfacet
   facet normal -1.000000e+000 -8.673617e-017 0.000000e+000
      outer loop
         vertex 0.000000e+000 0.000000e+000 2.000000e+001
         vertex 0.000000e+000 2.000000e+001 0.000000e+000
         vertex 0.000000e+000 0.000000e+000 0.000000e+000
      endloop
   endfacet
   facet normal 0.000000e+000 1.000000e+000 0.000000e+000
      outer loop
         vertex 2.000000e+001 2.000000e+001 2.000000e+001
         vertex 2.000000e+001 2.000000e+001 0.000000e+000
         vertex 0.000000e+000 2.000000e+001 2.000000e+001
      endloop
   endfacet
   facet normal 0.000000e+000 1.000000e+000 0.000000e+000
      outer loop
         vertex 0.000000e+000 2.000000e+001 2.000000e+001
         vertex 2.000000e+001 2.000000e+001 0.000000e+000
         vertex 0.000000e+000 2.000000e+001 0.000000e+000
      endloop
   endfacet
   facet normal 1.000000e+000 8.673617e-017 0.000000e+000
      outer loop
         vertex 2.000000e+001 0.000000e+000 2.000000e+001
         vertex 2.000000e+001 0.000000e+000 0.000000e+000
         vertex 2.000000e+001 2.000000e+001 2.000000e+001
      endloop
   endfacet
   facet normal 1.000000e+000 8.673617e-017 0.000000e+000
      outer loop
         vertex 2.000000e+001 2.000000e+001 2.000000e+001
         vertex 2.000000e+001 0.000000e+000 0.000000e+000
         vertex 2.000000e+001 2.000000e+001 0.000000e+000
      endloop
   endfacet
endsolid
```

Figure 4. *ASCII structure of the STL format*

```
# Blender v2.77 (sub 0) OBJ File: ''
# www.blender.org
mtllib Cube.mtl
o Cube
v 1.000000 -1.000000 -1.000000
v 1.000000 -1.000000 1.000000
v -1.000000 -1.000000 1.000000
v -1.000000 -1.000000 -1.000000
v 1.000000 1.000000 -0.999999
v 0.999999 1.000000 1.000001
v -1.000000 1.000000 1.000000
v -1.000000 1.000000 -1.000000
vn 0.0000 -1.0000 0.0000
vn 0.0000 1.0000 0.0000
vn 1.0000 -0.0000 0.0000
vn 0.0000 -0.0000 1.0000
vn -1.0000 -0.0000 -0.0000
vn 0.0000 0.0000 -1.0000
usemtl Material
s off
f 2//1 4//1 1//1
f 8//2 6//2 5//2
f 5//3 2//3 1//3
f 6//4 3//4 2//4
f 3//5 8//5 4//5
f 1//6 8//6 5//6
f 2//1 3//1 4//1
f 8//2 7//2 6//2
f 5//3 6//3 2//3
f 6//4 7//4 3//4
f 3//5 7//5 8//5
f 1//6 4//6 8//6
```

Figure 5. *OBJ format structure*

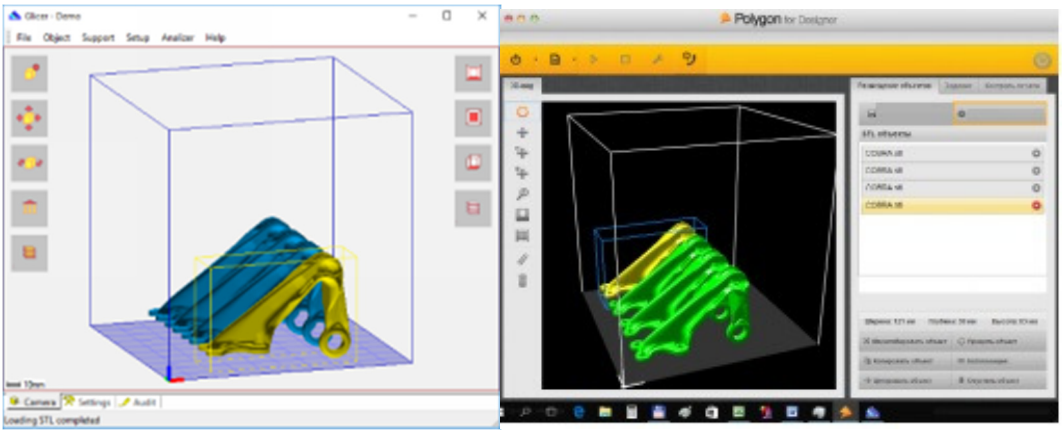


Figure 6. *Examples of camera layout in different software*

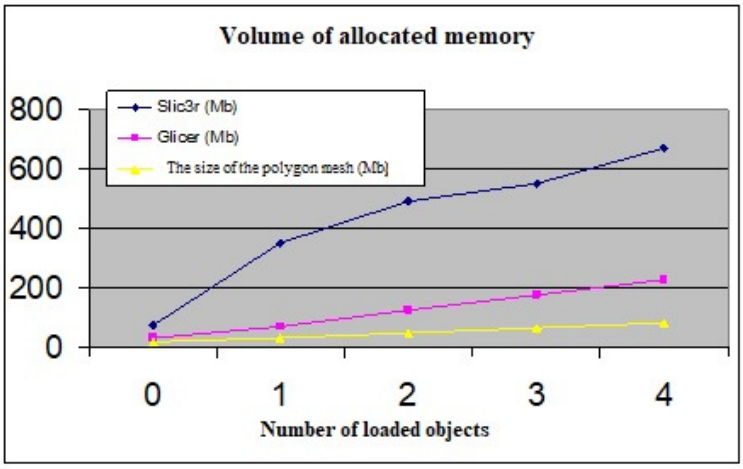


Figure 7. *The dependence of the allocated memory of the application on the number of downloaded objects*

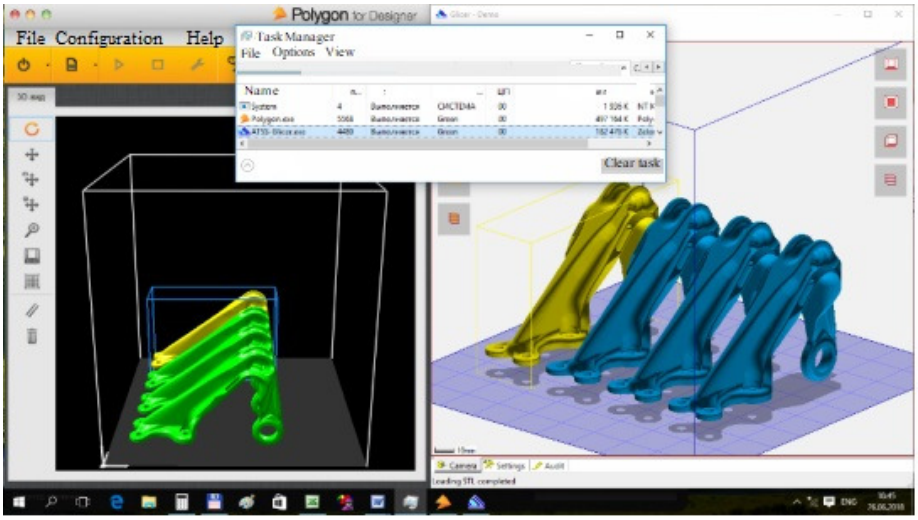


Figure 8. *The values of allocated memory in various applications*

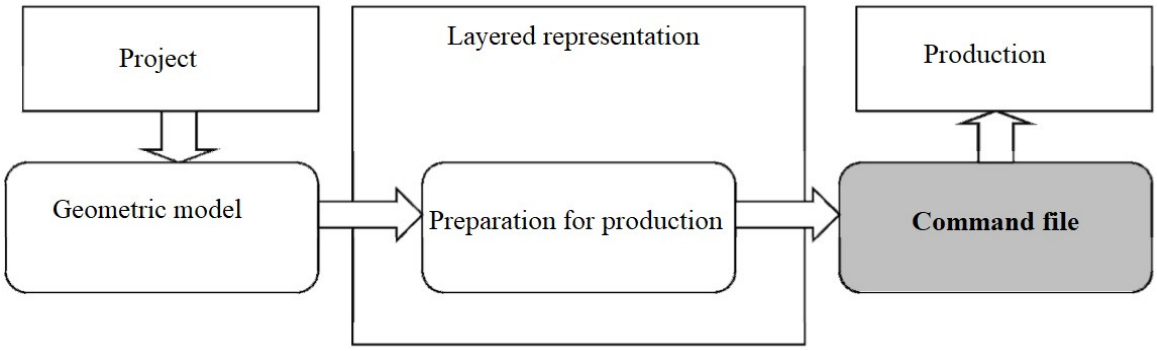
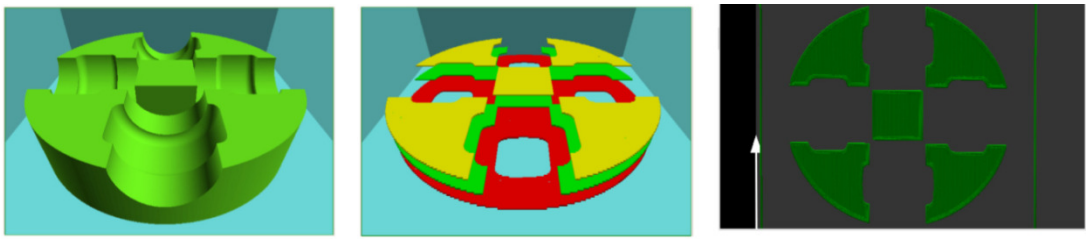


***Figure 9**. The place of the executable file in the production process*



a *b* *c*

***Figure 10.** Stages of Model Representation: a – Geometric model; b – Layered representation; c – Visual representation of the command file*

File Edit Format View Help

```
; generated by PK TPP
G21
M107
M85 S600
M104 S200
M190 S110
M109 S250
G28
G90
G1 X20 Y0 F10000
G90
G92 E0
M82
G1 F1200.000 E-1.00000
G92 E0
G1 Z7.100 F9600.000
G1 E1.00000 F1200.000
G1 X1.900 Y1.049 F9600.000
G1 X48.700 Y1.049 E2.31026 F1350.000
G1 X48.700 Y1.049 F9600.000
G1 X48.700 Y47.989 E3.62442 F1350.000
G1 X48.700 Y47.989 F9600.000
G1 X1.900 Y47.989 E4.93468 F1350.000
G1 X1.900 Y47.989 F9600.000
G1 X1.900 Y1.049 E6.24883 F1350.000
G1 F1200.000 E5.24883
G92 E0
;segType:Perimeter
G1 E1.00000 F1200.000
G1 X6.900 Y19.534 F9600.000
G1 X6.900 Y19.532 E1.00006 F1350.000
G1 X6.900 Y19.534 E1.00012 F1350.000
G1 F1200.000 E0.00012
G92 E0
```

***Figure 11.** Example of listing a command file*

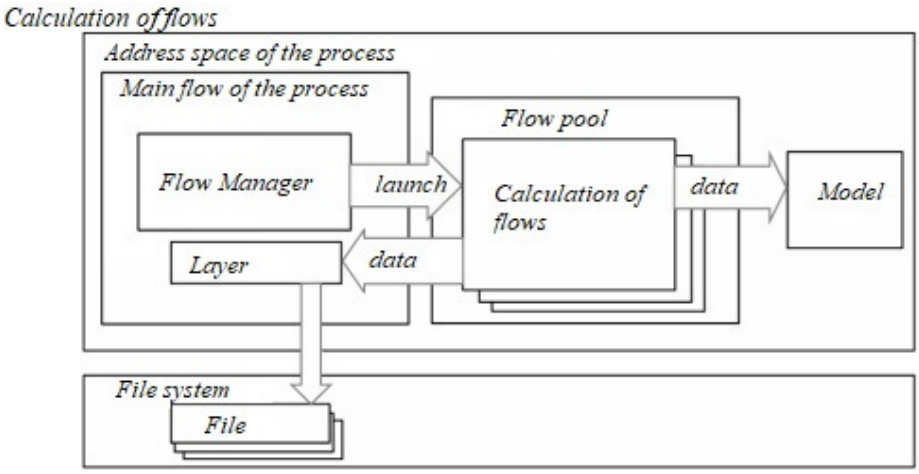


***Figure 12**. Architecture for task solution*

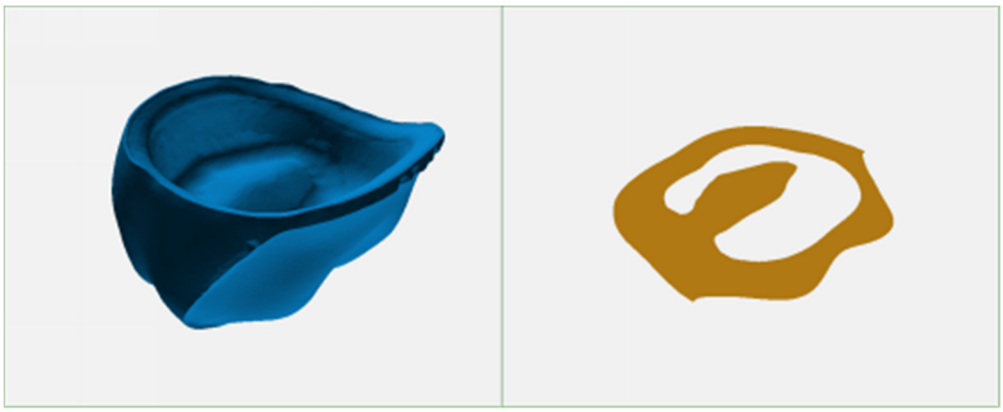


***Figure 13**. The considered model and a random layer*

Table 1. *Dependence of the calculation time on the number of threads*

Number of threads	Time of calculation, sec
1	32.36
2	17.05
4	16.59