May 14, 2019

A METHOD TO REDUCE THE NUMBER OF TRIANGLES OF A MESH TO ALLOW ITS FLUENT VISUALIZATION IN A 3D PRINTER CONTROL PANEL

HP INC

A method to reduce the number of triangles of a mesh to allow its fluent visualization in a 3D Printer control panel

This invention disclosure proposes a method for generating a reduced size mesh from a print-resolution mesh received in a 3D printer. The 3D printer receives the content to be printed by means of triangle meshes. To be able to provide the required printing resolution, the number of triangles conforming the 3D part can be quite high, leading to high memory and processing requirements when aiming to manipulate the model. Previewing the 3D model of the part on the printer control panel can lead to a non-responsive User interface and even impact the performance of the printer firmware because there is no limit specified on the number of triangles that the original mesh can have. This invention disclosure presents a method to generate a visualization mesh that has a lower resolution than the original mesh received by the printer. This is accomplished by reducing the number of triangles in the meshes which allows the control panel of the printer to smoothly display the original shape and appearance of the 3D parts. In addition to that, these generated meshes will also be used to create lower size files that can be transferred between units (e.g. Printing Unit, Build Unit and Processing Station Unit).

A 3D printing job may be received in 3MF format [1]. A 3MF file consists of a collection of 3D geometries referenced by one or more affine 3D transform matrices to position the content in the printer's build bed. In the general case, the 3D geometries are represented by means of triangle meshes. The user has total freedom to design the content as complex as he wants to be capable to benefit from the design freedom allowed by 3D printers. The more complex the geometry is, the higher number of triangles are required to represent it. To be able to print, the meshes must be rendered to some voxelization so that the information is ready to be consumed in a predictable amount of time at printing time.

Certain user interactions with the control panel can be enriched by allowing the user to visualize a 3D preview of the parts. For example, if the printer allows to remove parts from a job, or to duplicate some of the parts from the job, using the control panel, the printer has to provide the user with some visualization mechanism to allow part identification. Another example could be to show a preview from the parts on the Processing Station Unit while a job is unpacked, and parts retrieved.
The main problem of directly using the original mesh as it came in the 3MF is that the amount of triangles which represent the triangle mesh can be relatively high. And rendering a mesh conformed of an unbounded amount of triangles can cause the printer control panel to suffer from performance issues that drastically impact the user experience.

Low resolution mesh generation

When the 3MF job is received by the printer, first of all the content is parsed and the triangle meshes are extracted to intermediate files (see *Figure 1*). Then, the build hierarchy is generated with a collection of parts as defined in the 3MF, where every part points to a triangle mesh (stored in an intermediate file) and an affine 3D transform matrix which positions the part into the bed. Then, the step to generate the low-resolution mesh is started. Here, for every triangle mesh, some mesh decimation algorithm is executed as described in the next subsection, and a mesh which contains a bounded amount of triangles is generated which approximates the full resolution mesh. The mesh decimation algorithm defines some maximum number of triangles (e.g. 10000) and iterates decimation till a number of triangles below or equal to the target is achieved. Then, the mesh is stored in another intermediate file and is related to the part/s it belongs to. After that, the job continues its voxelization process to get the job processed and ready to be printed. After that, the low resolution meshes are available to be displayed to the user via the Control Panel.

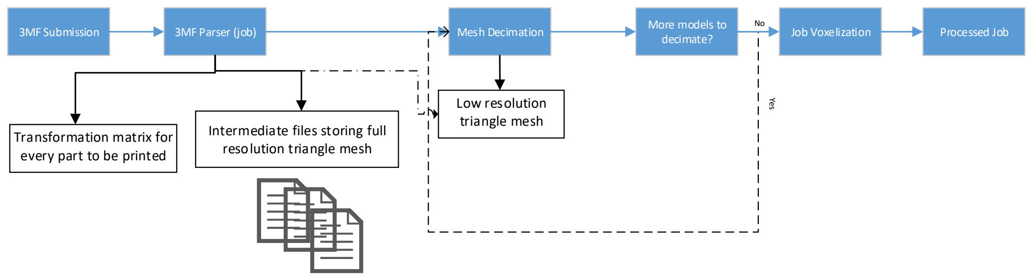

3MF Submission
3MF Parser (job)
Mesh Decimation
More models to decimate?
No
Job Voxelization
Processed Job
Yes
Low resolution triangle mesh
Transformation matrix for every part to be printed
Intermediate files storing full resolution triangle mesh

Figure 1. The job submission pipeline

Mesh decimation algorithm

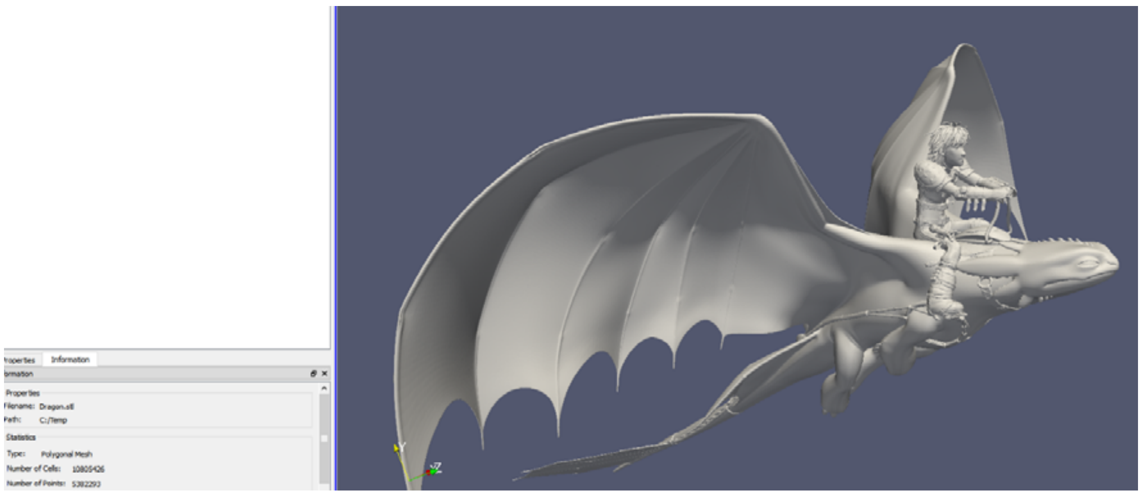
There are many efficient mesh decimation algorithms in the literature. Any of them would fit in our approach as long as they allow to target a maximum number of triangles (bound). The only difference will be on the produced reduced mesh depending on the used algorithm. Mesh decimation algorithms aim to find an approximation mesh to an original triangle mesh which approximates the shape of the original mesh and minimizes the error. For example, we can find several algorithms implemented in the open source VTK [2] library:

- *vtkDecimatePro*: The algorithm is based on [3]. It proceeds as follows: Each vertex in the mesh is classified and inserted into a priority queue. The priority is based on the error to delete the vertex and retriangulate the hole. Vertices that cannot be deleted or triangulated (at this point in the algorithm) are skipped. Then, each vertex in the priority queue is processed (i.e., deleted followed by hole triangulation using edge collapse). This continues until the priority queue is empty. Next, all remaining vertices are processed, and the mesh is split into separate pieces along sharp edges or at non-manifold attachment points and reinserted into the priority queue. Again, the priority queue is processed until empty. If the desired reduction is still not achieved, the remaining vertices are split as necessary (in a recursive fashion) so that it is possible to eliminate every triangle as necessary.
- *vtkQuadraticClustering*: The algorithm based in [4]. The general approach of the algorithm is to cluster vertices in a uniform binning of space, accumulating the quadric of each triangle (pushed out to the triangles vertices) within each bin, and then determining an optimal position for a single vertex in a bin by using the accumulated quadric. In more detail, the algorithm first gets the bounds of the input poly data. It then breaks this bounding volume into a user-specified number of spatial bins. It then reads each triangle from the input and hashes its vertices into these bins. (If this is the first time a bin has been visited, initialize its

quadric to the 0 matrix.) The algorithm computes the error quadric for this triangle and adds it to the existing quadric of the bin in which each vertex is contained. Then, if 2 or more vertices of the triangle fall in the same bin, the triangle is discarded. If the triangle is not discarded, it adds the triangle to the list of output triangles as a list of vertex identifiers. (There is one vertex id per bin.) After all the triangles have been read, the representative vertex for each bin is computed (an optimal location is found) using the quadric for that bin. This determines the spatial location of the vertices of each of the triangles in the output.

- *vtkQuadraticDecimation*: The algorithm is based in [5]. The algorithm is based on repeated edge collapses until the requested mesh reduction is achieved. Edges are placed in a priority queue based on the "cost" to delete the edge. The cost is an approximate measure of error (distance to the original surface)–described by the so-called quadric error measure. The quadric error measure is associated with each vertex of the mesh and represents a matrix of planes incident on that vertex. The distance of the planes to the vertex is the error in the position of the vertex (originally the vertex error iz zero). As edges are deleted, the quadric error measure associated with the two end points of the edge are summed (this combines the plane equations) and an optimal collapse point can be computed. Edges connected to the collapse point are then reinserted into the queue after computing the new cost to delete them. The process continues until the desired reduction level is reached or topological constraints prevent further reduction. Note that this basic algorithm can be extended to higher dimensions by taking into account variation in attributes (i.e., scalars, vectors, and so on).

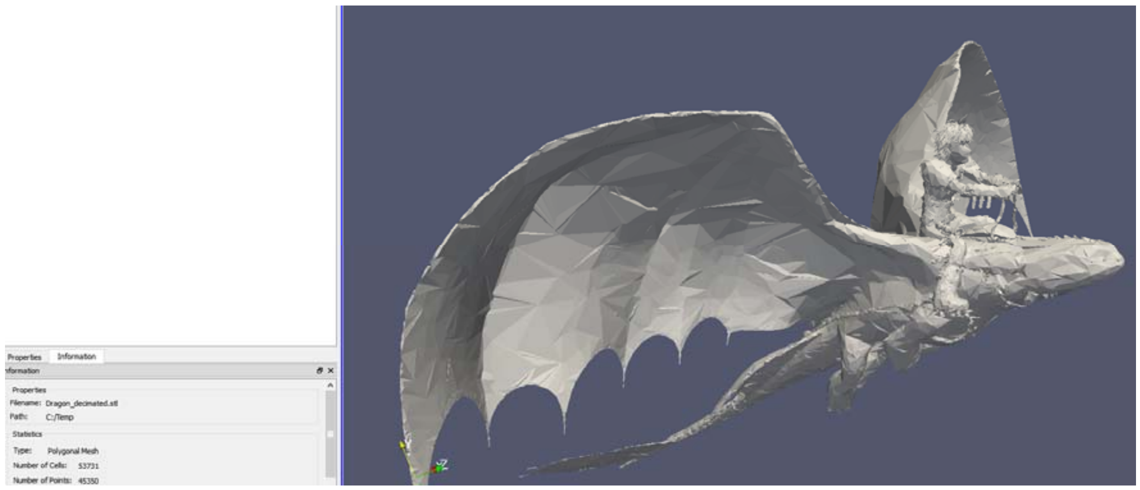
Figure 2 shows and example of a mesh which originally contains more than 10 millions of triangles. In this case, *vtkDecimatePro* was chosen and a limit of 60000 triangles was defined. Then, the algorithm was configured to reduce 99.5% of the triangles and resulting mesh contained 53731 triangles (*Figure 3*). Finally, *Figure 4* shows a couple of captures of our proof of concept of a viewer for the control panel, showing how the decimated mesh from previous example is shown and an example of a mesh with surface color information.



Properties
Filename: Dragon.stl
Path: C:/Temp

Statistics
Type: Polygonal Mesh
Number of Cells: 10805426
Number of Points: 5382293

Figure 2. Example of a print resolution mesh containing more than 10 million triangles.



Filename: Dragon_decimated.stl
Path: C:/Temp
Type: Polygonal Mesh
Number of Cells: 53731
Number of Points: 45350

Figure 3. Mesh from Figure 2 reduced to approximately 53k triangles.

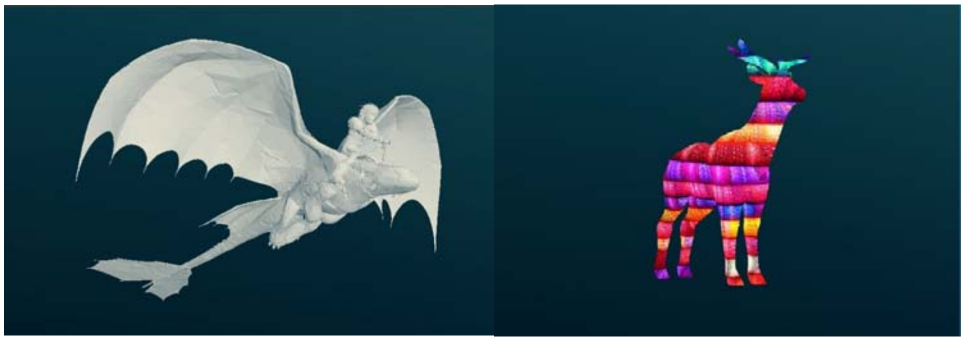


Figure 4. Example of visualization in Control Panel using our proof of concept for the decimated mesh from Figure 3 and a decimated model maintaining color information in remaining vertices.

The proposed method has the following advantages:

- It is able to limit the memory and CPU consumption of showing a 3D job preview by reducing the amount of triangles on the print resolution meshes.
- Color information is obtained from the triangle vertices so, as long as some vertices are maintained from the original mesh, the preview will keep the color information and also an approximation of the full resolution mesh surface color.
- As the produced preview files have a reduced size, they could be transferred to the Build Unit where the available storage for jobs is quite reduced, and then transferred to the Processing Station Unit so that the preview could also be displayed in the job related operations (i.e. unpack or extract).
- Displaying the 3D job preview on the control panel exposes a new variety of workflows that can be more easily and efficiently managed like:
  - Digitally mark (e.g. with specific color or textures) the parts of the job that have errors during printing. This allows user to visually identify which parts are not valid once the job is fully printed.
  - Ability to select a part, view its details, remove it or even add a new copy of it in the existing job (if there is enough space in the Build Unit) via control panel.

  - o Visually identify parts of the job via control panel as they are being unpacked and cleaned in the Processing Station.

- The method is detectable as the generated lower resolution mesh is displayed to the user via the control panel.

1. References

[1] http://3mf.io
[2] http://www.vtk.org
[3] W, Schroeder, J. Zarge, and W, Lorensen, "Decimation of triangle meshes", Computer Graphics, 26(2):65-70, July 1992.
[4] P. Lindstrom, "Out-of-core simplification of Large Polygonal Models". Proc. ACM SIGGRAPH, pp. 259-262, 2000.
[5] M. Garland and P. Heckbert, "Surface simplification using quadric error metrics", Proc. ACM SIGGRAPH'97. pp. 209-216. 1997.

Disclosed by Sergio Gonzalez-Martin, Jordi Torres and Roberto Panchon Carton, HP Inc.