

GOM 3d file format

Version 1.1a
GOM mbH, November 2002

GOM mbH
Mittelweg 7-8
38106 Braunschweig
Germany

email: support@gom.com
<http://www.gom.com>

1 Introduction

The g3d file format can be used to store several types of 3d data in a file. It is used by the GOM Atos 3d scanner.

The following data types are supported by the format:

- Triangle meshes (also called polygonizations)
- Rastered point clouds
- ISO point clouds
- Unordered point clouds
- Sections or Scanlines
- Feature lines (GOM internal)
- Colored triangle meshes

The actual software (version 5 as of 20.11.2002) uses the triangle meshes and sections only. The other file types will not be used any more.

1.1 Hints for Interpreting the file structure

The g3d files are build of several data blocks.

For each data block, except of the first one, a pointer to its position in the file is stored in a previously read block. It is recommended that this pointer is used to access the block as the position is freely selectable in the file. The block ordering may be completely different from the ordering described in this document.

For each block there is a size argument. When reading a block its size should be compared to known sizes. All data stored in a block is valid up to the block size. If a block size is smaller than in this document the block data can be used completely. Missing data should be interpreted as good as possible. If a block is larger, it is mainly from a newer program version. Then the unknown data should be ignored. New file versions will add data to blocks only. Data blocks will never shrink or change their meaning.

The file version should not be used to classify the contents of the file. Use the block sizes instead. Different data types can be stored in a g3d file. Unknown data should be read over. The first element of each view header is a pointer to the next view header or zero for the last view.

2 File structure

This section describes the global file structure used to store the data.

The data is organized in views. Every view contains a part of the complete data. It is possible to store all views in one file or to use a separate file for each view.

2.1 All views in one file

When all views are stored in one file the file structure is as follows:

Global header
View 1 header
View 1 data
View 2 header
View 2 data
...
View n header
View n data

2.2 Every view in a separate file

If each view is in one file there is a header file containing the view names and several files with one view each.

Header file:

Global header
File list

Views:

Local header
View header
View data

3 The Header

The header is the beginning of every loadable g3d file. The header position is always at the beginning of the file.

3.1 Global header

The global header is at the beginning of every g3d main file.

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
8	string	"%GOM-3DH"	Magic key
2	unsigned short	0x0001: LSB/HSB 0x0100: HSB/LSB	Byte order
2	unsigned short	100	Version number ^a
4	unsigned int	96	Header size
4	unsigned int	0 or 1	Project-flag ^b
4	unsigned int	0 or 1	Flag for all views in one file ^c
4	unsigned int	0...	Number of views
4	unsigned int	...	Pointer to header of first view ^d
64	string	" "	comment

^aDo not use to classify the file version!

^bThe project-flag defines, whether the file belongs to a project (flag = 1). The files should not be loaded individually.

^cFlag = 1 if all views are in one file, flag = 0 if the views are in separate files.

^dOr pointer to file list. Pointer is zero, if there are no views.

3.2 Local header

The local header is at the beginning of a separated view file.

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
8	string	"%GOM-3DV"	Magic key
2	unsigned short	0x0001: LSB/HSB 0x0100: HSB/LSB	Byte order
2	unsigned short	100	Version number
4	unsigned int	24	Header size
4	unsigned int	0 or 1	project-flag ^a
4	unsigned int	...	Pointer to header of view

^aThe project-flag defines, whether the file belongs to a project (flag = 1). The files should not be loaded individually.

4 The File List

The file list is a string list containing the names of the files with the 3d data.

The file names are separated by "\n". All file names must not contain a directory name. The files must be located in the same directory as the header file.

5 Views

Every view contains a set of 3d data. The following view types are available:

<i>Type</i>	<i>Description</i>
0	Triangle mesh
1	Rastered point cloud
2	ISO point cloud
3	Unsorted point cloud
4	Sections or Scanlined point cloud
5	Feature line (internal)
6	Colored triangle mesh

If an unknown view is recognized while reading a file, this view should be ignored.

5.1 Triangle mesh

A triangle mesh contains a set of 3d points and a set of triangles with point numbers.

Header:

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
4	unsigned int	...	Pointer to next view ^a
4	unsigned int	168	Header size
4	unsigned int	...	Id ^b
4	unsigned int	0	Type
64	string	" "	Name
64	string	" "	comment
4	unsigned int	0-...	Number of points
4	unsigned int	...	Pointer to first point ^c
4	unsigned int	28	Size of the point
4	unsigned int	0-...	Number of triangles
4	unsigned int	...	Pointer to first triangle ^d
4	unsigned int	12	Size of a triangle

^aZero, if it is the last view.

^bAll ids must be different.

^cZero, if there are no points. All points are stored continuously.

^dZero, if there are no triangles. All triangles are stored continuously.

Point:

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
8	double	...	X-coordinate
8	double	...	Y-coordinate
8	double	...	Z-coordinate
4	float	...	Quality value ^a

^aHigher values mean better quality.

Triangle:

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
4	unsigned int	0-...	First point number
4	unsigned int	0-...	Second point number
4	unsigned int	0-...	Third point numbers ^a

^aThe point numbers start with zero. The numbers are defined by the order of the points in the file. The normal vector of the triangle must be calculated by $(p_2 - p_1) \times (p_3 - p_1)$.

5.2 Rastered point clouds

A rastered point cloud contains 3d points ordered in a regular (u,v) grid.

Header:

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
4	unsigned int	...	Pointer to next view ^a
4	unsigned int	212	Header sizes
4	unsigned int	...	Id ^b
4	unsigned int	1	Type
64	string	" "	Name
64	string	" "	Comment
4	unsigned int	0-... Number of points	
4	unsigned int	...	Pointer to first point ^c
4	unsigned int	28	Size of the point
4	unsigned int	1-...	u-scan-raster
4	unsigned int	1-...	v-scan-raster ^d
8	double	...	First orientation vector, X-coordinate
8	double	...	Y-coordinate
8	double	...	Z-coordinate ^e
8	double	...	Second orientation vector, X-coordinate
8	double	...	Y-coordinate
8	double	...	Z-coordinate ^f

^aZero, if it is the last view.

^bAll ids must be different.

^cZero, if there are no points.

^dStepsize of the raster, 0 if there is no raster.

^eThe first orientation vector defines the view direction of the camera. The second orientation vector is for a second camera. If these values are unknown, a zero vector can be defined.

^fThe first orientation vector defines the view direction of the camera. The second orientation vector is for a second camera. If these values are unknown, a zero vector can be defined.

Point:

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
8	double	...	X-coordinate
8	double	...	Y-coordinate
8	double	...	Z-coordinate
4	unsigned int	0-...	u-raster position
4	unsigned int	0-...	v-raster position ^a
4	float	...	Quality value ^b

^aWith the raster a 2d matrix can be defined. The minimal distance of two raster points is defined in the header. Different points are defined by different raster positions. The raster positions of the points must increase in u- and v-direction.

^bHigher values mean better quality.

5.3 ISO point clouds

ISO point clouds are almost identical to rastered point clouds. The differences are:

- The view type is 2
- Single scanlines are signed by different v-raster positions
- The points must be described by the rule "Z above XY"

5.4 Unsorted point clouds

Unsorted point clouds are almost identical to rastered point clouds. Here are the differences:

- The view type is 3
- The u- and v-raster step size is zero
- The u- and v-raster position of every point is zero

5.5 Sections or Scanlines

Sections or scanlined point clouds are almost identical to rastered point clouds. Here are the differences:

- The view type is 4
- Single sections / scanlines are defined by different v-raster positions

5.6 Colored triangle meshes

Colored triangle meshes are like triangle meshes with additional colors for each point.

Header:

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
4	unsigned int	...	Pointer to next view ^a
4	unsigned int	168	Header size
4	unsigned int	...	Id ^b
4	unsigned int	6	Type
64	string	" "	Name
64	string	" "	comment
4	unsigned int	0-...	Number of points
4	unsigned int	...	Pointer to first point ^c
4	unsigned int	32	Size of a point
4	unsigned int	0-...	Number of triangles
4	unsigned int	...	Pointer to first triangle ^d
4	unsigned int	12	Size of a triangle

^aZero, if it is the last view.

^bAll ids must be different.

^cZero, if there are no points.

^dZero, if there are no triangles.

Point:

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
8	double	...	X-coordinate
8	double	...	Y-coordinate
8	double	...	Z-coordinate
4	float	...	Quality value ^a
1	byte	...	Red
1	byte	...	Green
1	byte	...	Blue
1	byte	...	Alpha (unused)

^aHigher values mean better quality.

Triangle:

<i>Length</i>	<i>Type</i>	<i>Default value</i>	<i>Description</i>
4	unsigned int	0-...	First point number
4	unsigned int	0-...	Second point number
4	unsigned int	0-...	Third point number ^a

^aThe point numbers start with zero. The numbers are defined by the order of the points in the file. The normal vector of the triangle must be calculated by $(p_2 - p_1) \times (p_3 - p_1)$.