MSH_MAKER v2.0

© Radu Poenaru 2004

Use MSH_MAKER to convert .3ds format file into Orbiter .msh format
(as for now compatible with version 031217)

---

1. HOW TO USE IT?

1. Copy your .3ds file into the same directory as the Msh_Maker, then rename it "input.3ds"

2. Export a second 3ds file named "shadow.3ds", containing ONLY the objects that you wish to cast shadows. You can have the two files (input.3ds and shadow.3ds) identical, in which case all of your mesh will produce a shadow.

3. Run Msh_Maker once; it will create a readable, text version of the .3ds in 'debug.txt'

4. Edit the debug.txt file as needed : change material properties, etc; (see section 2. Debug.txt)

NOTES: The program first looks for a "debug.txt" file in the current directory. If such file exists, it will create a ".msh" file from it (as in step 4). If you rather desire to perform step 2 instead, delete "debug.txt"

2. Debug.txt

The debug.txt serves as an intermediary file between the 3ds and msh formats. Why edit a intermediary file and not directly on the msh file? Because, as you will see , a number of parameters in the debug.txt file will determine how the .msh file will finally look like, saving tons of editing work that would otherwise had to be performed on the .msh file.

[SHADOW_OBJECTS] / [END_SHADOW]

The names of the objects contained in this list will display shadows in Orbiter. Add or remove objects as you see fit. Notice that shadows can

be computationally expensive, so it's a good ideea to keep only the general outline of the hull as being part of your shadow and save the "details" from being rendered as shadows.

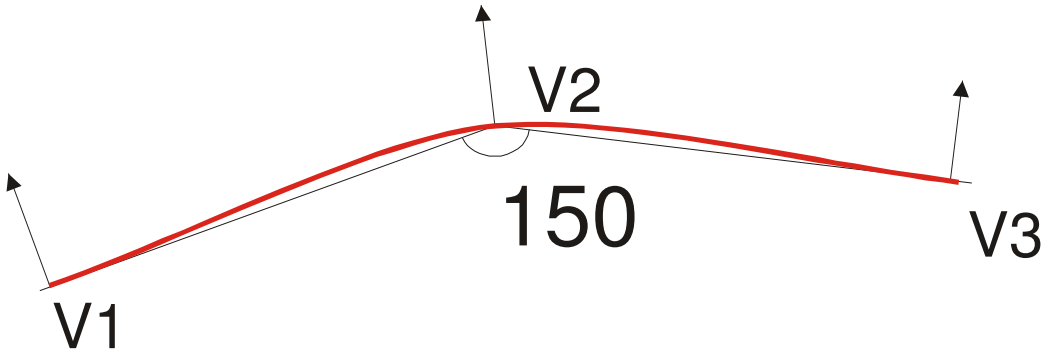
MAT_NAME01 - "name" (string)

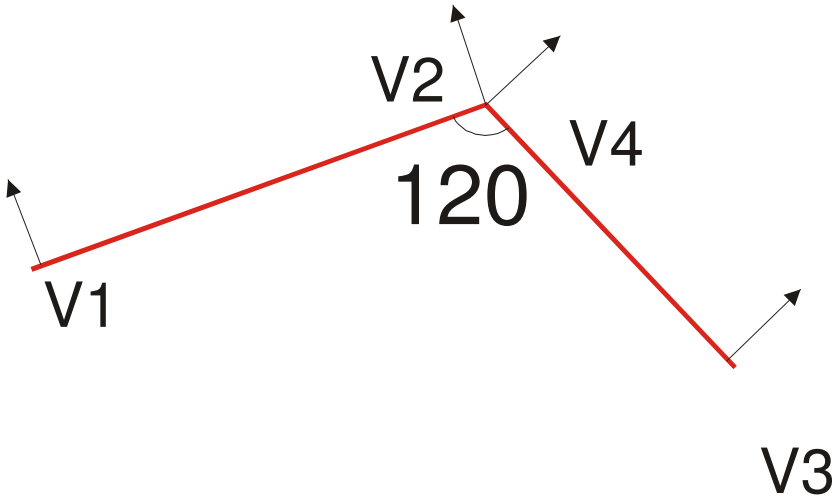
The name of the material being described. You can have any number of materials in your mesh, but remember that multi/sub materials are not supported. You need to create a different material for each and every type of materials being used.

MATERIAL_TENSION - 1.0 (in radians)

This is one of the parameters that I was talking about above. It controls how smooth, or how sharp your mesh is. This is being done at material level. You can have different tensions for each material…

What does the 1.0 mean .. it is 1.0 radians (~ 57 degrees), and it specifies the angle at which MshMaker will "break" the edge in to a sharp edge.





If you are familiar with 3D modeling then , you might have guessed, this has to do with normals and lighting.
The final outcome being that higher numbers will produce a smoothed object (as you might need for say, a rocket cylinder), while smaller numbers will produce an object with sharper edges.

MATERIAL_AMBIENT r g b [0..255]
MATERIAL_DIFFUSE r g b [0..255]
MATERIAL_SPECULAR r g b [0..255]
These values are directly taken from the 3ds and passed to the msh file.. they're pretty straightforward so I won't take much time writing about them

MATERIAL_SHININESS 10
MATERIAL_SHIN_STRENGTH 0
Again, these two values are directly taken from the material properties of the 3ds files. You should be aware of their meaning as a 3D modeler, but just in case you get confused: one of them means how much the material shines, while the other one sets how diffuse the shining is.

TRANS_PERCENT 100
Percentage for transparency… 0 means the object is invisible, 50 means it is 50% opaque and 100 means the object is totally opaque. You only need to set this parameter (either in the debug.txt, or directly in your modeling program) to enable transparency. MshMaker knows to position transparent objects to the end of the mesh for proper rendering.

SELF_ILLUM 0
Percentage for self-illumination. 0 means the object emits no light of it's own. 100 means the object will shine at the color set by MATERIAL_DIFFUSE. Note that this kind of illumination will obviously not cast any shadows. It will simply make the object "glow"

TWO_SIDED
If this parameters is present at a material property list, then 2 objects will be constructed for each object of this material. The second one will be the "inverted" version of the first one (ie. vertex order in triangles) causing the actual object to be visible on "both sides"(ie. double sided). This can be very useful for thin objects that need to be visible both on the inside and outside(ie. engine bell)

MASTER_SCALE 1
You can use this parameter as a last minute scalling of your mesh. This is a global parameter, so ALL of your mesh will be scalled by the amount entered here.

NAMED_OBJECT name

The name of the object currently being described

TRIANGLE_VERTEXLIST number

The list of vertices making up the current object. “number” is the number of vertices. Note that material tension is not applied yet, so the final number of vertices in the msh might end up being larger.

TRIANGLE_FACELIST number

The list of indices of vertices making up the triangles of your object. ”number” is the number of triangles, and this is the final number that will end up in the msh file aswell

TRIANGLE_MATERIAL name

The name of the material for this object.

3. Keyframe.txt

OBJECT name

The name of the object being described. Note that Orbiter SDK expects object indices not names.You can match names vs. object indices by searching for the object name in the final msh file.. a object index is written in the format :

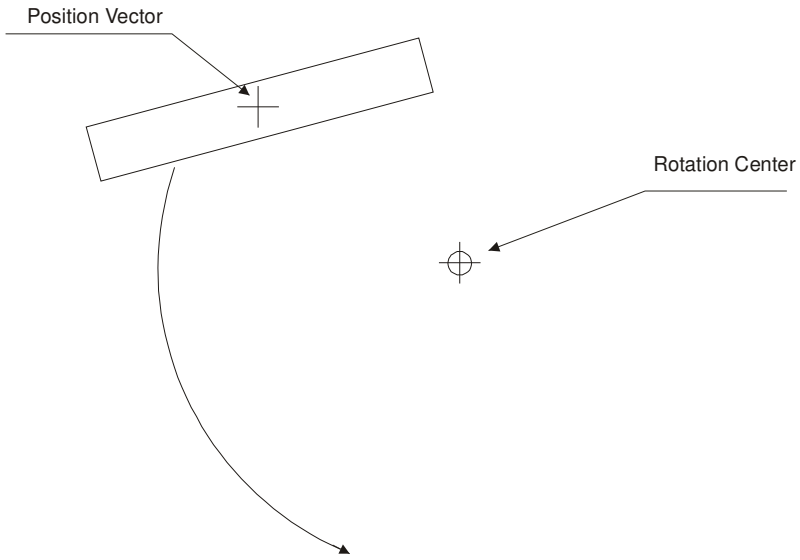
GEOM 127 126 ; <Object_name> / <Object_index>

HEIRARCHY -1

Index of heirarchy for structured objects. The algorithm for the tree is very clever to use, but kinda hard to explain in here. Search for some dedicated 3DS-Heirarchy tutorials for more info on this.

PIVOT_OFFSET_XYZ x y z

Vector (XYZ) from the center of the object to the pivot point (center of rotation). You need to add this offset vector to the object position vector to determine the actual rotational point, if you have a rotational animation. In the graphic below, PIVOT_OFFSET represents the diference between Position Vector and Rotation Center: (note you get the Position Vector from the Translation keyframes)



TRANSLATIONKEYS number

The following list is a “number” number of keys representing translations for the current object:

FRAME_NR: 0 **POS XYZ**: 44.3373 -10.1205 0

- means at frame 0 ,the Position Vector of the object is (44.33,-10.12,0.0)

ROTATIONKEYS number

The following list is a “number” number of keys representing rotations for the current object:

FRAME_NR: 100 **ANG**: 0.829031 **AXIS_V XYZ**: 0 0 –1

- means at frame 100, there is a rotation or 0.829 radians on the axis (0,0,-1) around the point (PositionVector + PivotOffset).

To set up a rotation from the example above use :

```
mt.transform = MESHGROUP_TRANSFORM::ROTATE;
mt.P.rotparam.axis=_V (0,0,-1); // read from the AXIS_V XYZ

mt.P.rotparam.angle=0.829031/100 * delta_frames;
                    //read total angle from from ANG ,
                    //divided by total number of frames in the
                    //interval (from 100 to 0 are 100 frames),
                    // then multiplied by the number of ellapsed
                    //frames  (from the OrbiterAPI)
```

mt.P.rotparam.x=44.3373;
mt.P.rotparam.y=-8.1205;
mt.P.rotparam.z=0;
//the center of rotation is computed by adding
//PIVOT_OFFSET_XYZ (in our ex. 0 2 0),
//to the center of the object (44.3373,-10.1205,0).

NOTE:all the XYZ vectors in the keyframe file are in 3DS units, and are thus not scaled down by MASTER_SCALE.
Rescale PIVOT_OFFSET_XYZ and POS_XYZ to fit your needs. AXIS_V and ANG do not need to be scaled down.

NOTE: as with all the animation keys, the FRAME_NR represents the frame where the animation has ENDED rather than when the animation started.

Improvements for V2.0

- added double-sided materials support. If you object is created from a double sided material, both sides will be visible in Orbiter. This can get usefull for engine nozzles, etc...

- added shadow support. Now MSH_MAKER uses two 3ds files : input.3ds and shadow.3ds

Improvements for V1.21

- keyframe information (animations) are now read from the 3ds and passed into a "keyframe.txt" when performing animations
- program no longer crashes when no materal / texture is present in the 3ds file. Dunno why someone would not use any materials though...

Improvements for V1.2

- MASTER_SCALE now works. Vertex coords are now divided by MASTER_SCALE before they are put in the .msh file.
(ie. a "MASTER_SCALE 10 " will produce a .msh ten time smaller than in the .3ds file)
- all the material parameters supported in Orbiter are now gathered from the .3ds
- changed default MATERIAL_TENSION to 1.0 ; now only sharp edges get new vertex by default.
- objects are ordered in the .msh file so that transparent objects are last in the file; this assures proper rendering.

Improvements for V1.1

- fixed a bug causing a div/0 when normals were perpendicular.
- fixed a bug that crashed the program when no textures were used.
- now the vertex parser scans for unreferenced vertices and deletes them from the stream.
- changed the meaning of MATERIAL_TENSION; MATERIAL_TENSTION no longer refers to the angle between faces; now is the threshold for the angle between facet normal & the average of normals in that point. Not that it would matter much or anything.....

Send any comments or bug reports to
poenaru.radu@rcc.rondo-ganahl.com