TARDEC

---TECHNICAL REPORT---

No. 14349

By: Wesley Bylsma

CREATION OF VIRTUAL REALITY MODELING LANGUAGE (VRML) GEOMETRY DATA FROM MOVIE.BYU DATA

U.S. Army Research, Development and Engineering Command
U.S. Army Tank-automotive and Armaments Research
Development and Engineering Center
Detroit Arsenal
6501 East 11 Mile Road
Warren, Michigan 48397-5000

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
30 NOV 2004		**-**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
CREATION OF VIRTURAL REALITY MODELING LANGUAGE (VRML) GEOMETRY DATA FROM MOVIE.BYU DATA	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
WESLEY BYLSMA	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
US ARMY TARDEC - NATIONAL AUTOMOTIVE CENTER,ATTN: AMSRD-TAR-N/MS157,6501 EAST 11 MILE RD,WARREN,MI,48397-5000	**14349**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

A process for converting Movie.BYU graphic ".geometry" files using the AWK programming language is presented, with the intent of future use in scene assembly into Virtual Reality Modeling Language (VRML) file.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			
unclassified	**unclassified**	**unclassified**		**9**	

Technical Report 14349
November 2004

CREATION OF VIRTUAL REALITY MODELING LANGUAGE (VRML) GEOMETRY DATA FROM MOVIE.BYU DATA

Wesley Bylsma
U.S. Army Research, Development and Engineering Command (RDECOM)
U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center (TARDEC)
National Automotive Center (NAC)
ATTN: AMSRD-TAR-N/MS157
6501 E 11 Mile Road
Warren, Michigan 48397-5000

1.0 INTRODUCTION

Visualization of complex information is one of the best ways to communicate its meaning. The focus of this effort is on the creation of the geometry portion a Virtual Reality Modeling Language (VRML) file that is used to visualize ground vehicle simulations. As Figure 1 depicts, there are five essential elements that should be included within the composite VRML file for meaningful visualization effects. Only the geometry element is discussed here.

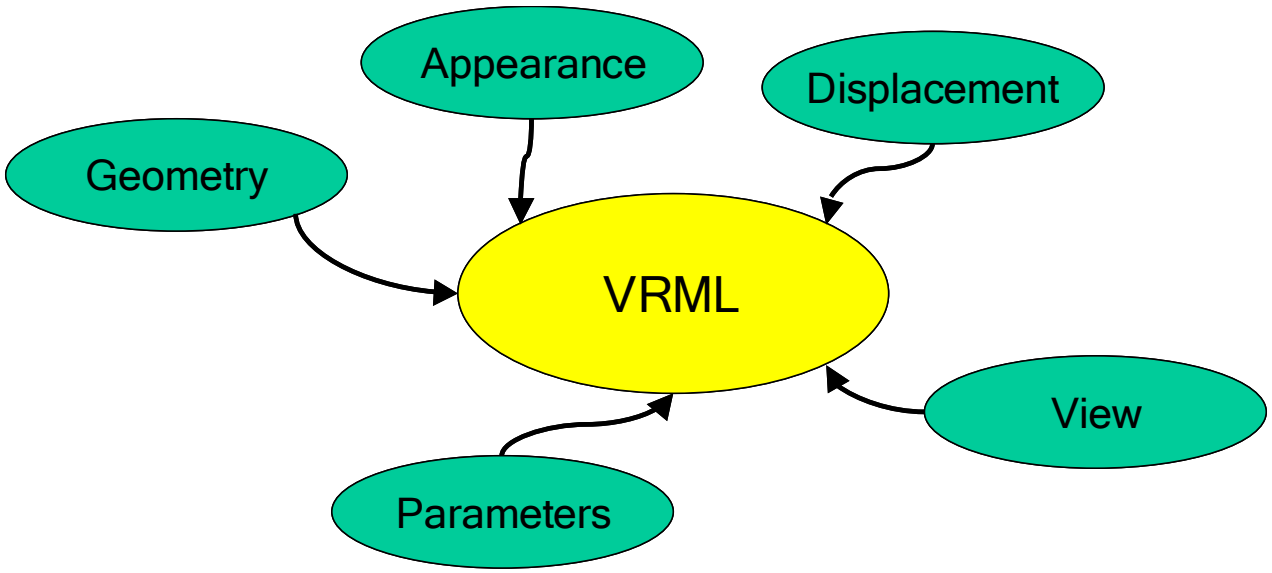

Figure 1 – Essential VRML Visualization Components

The geometry element is a generic file with the ".geometry" extension that contains three dimensional points and connection information that make up polygonal parts, or geometric structures. This report addresses the conversion process between the Movie.BYU format to the generic ".geometry" format to be used with VRML. The conversion process is accomplished with the AWK Programming Language, named after its authors (Alfred V. Aho, Brian W. Kernighan, and Peter J. Weinberger at Bell Labs), which is designed to provide easy data manipulation and extraction of text files. In this case the Free Software Foundation's GNU version, GAWK, is used. More conversion processes may be developed in the future for inclusion of other geometric file formats. The focus of this discussion is restricted to the conversion of the Movie.BYU format. Section 2.0 begins with a discussion of the Movie.BYU format. Section 3.0 discusses the geometry (".geometry") format, section 4.0 discusses the VRML format that will be generated from the ".geometry" format, and section 5.0 outlines the GAWK conversion processes with subsections on specific topics.

2.0 MOVIE.BYU FORMAT

The Movie.BYU format [1] was developed at Brigham Young University and may also be referred to as a ".geo" file. It is a very simple ASCII based format that allows for multiple non-connecting parts. Figure 2 defines the essential parts of the

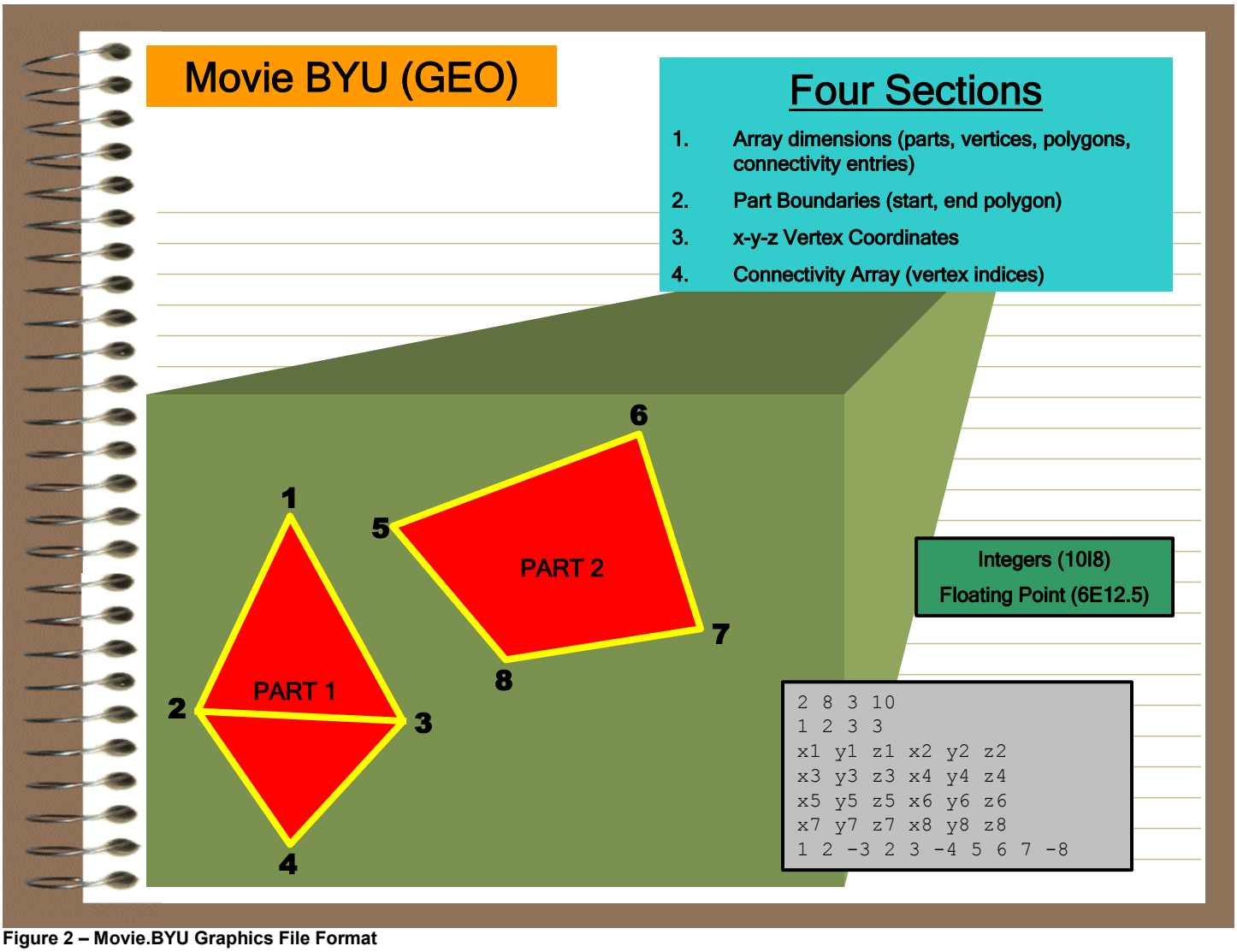

Figure 2 – Movie.BYU Graphics File Format

Movie.BYU file. It contains four sections. Section one defines the number of parts, vertices, polygons, and connectivity. Section two defines starting and ending polygons for each part. Section three contains the three dimensional (x,y,z) data for each vertex. Section four defines the vertices making up each polygon. Sections one, two, and four use integer values and each line has the FORTRAN format of "10I8". In section four, the end of a polygon vertex is signaled by a minus integer. Section three contains floating point numbers and each line has the FORTRAN format of "6E12.5". The gray box in Figure 2 shows an example file layout.

3.0 GEOMETRY FORMAT

The geometry file is also a simple ASCII file like the Movie.BYU, only with a different format. Its format is laid out into sections. Two sections are possible: Points and Part.
Multiple Part sections are possible.

The Point section begins with "#POINTS:" and ends with "#END". Following the colon in the "#POINTS:" header the number of points (or lines) should be given. Each line between the header and trailer contains the x, y, and z component of a three dimensional point (vertex) in space. Below is an example ".geometry" file:

```
#POINTS: 31596
4.634800 1.441500 1.896100
4.634800 1.840200 1.896100
...
```

```
2.323900 1.884100 5.651400
4.532100 1.909800 4.565300
#END
#PART: 1
0 2 1  -1
3 5 4  -1
...
27066 27068 27067  -1
27069 27071 27070  -1
#END
#PART: 2
27072 27074 27073  -1
27075 27077 27076  -1
...
31590 31592 31591  -1
31593 31595 31594  -1
#END
...
```

The Part section begins with "#PART:" and ends with "#END". Following the colon in the "#PARTS:" header the number of the part should be given. Each line between the header and trailer contains the index number into the point array of each point (zero index based) to be connected together to form a polygonal face. This sequence is ended with "-1". For a triangular surface each line will only contain four numbers (three vertices and the value "-1"). The length of each line is not specified since it is just copied into the VRML file (see 4.0 VRML FORMAT). See the example ".geometry" file above.

4.0 VRML FORMAT

The geometric data used in the VRML file [2] is included using the Coordinate and IndexedFaceSet nodes as defined in ISO 14772-1:1997. The creation of these nodes is done during final scene assembly with another program. A description is included here to help understand where the geometry data is included into the final VRML scene file.

The Coordinate Node template is

DEF Cx Coordinate { point [x1 x2 x3 ... xn] }.

This node defines all the points in the geometry file first. By including a name to the node definition, it can be referenced later in other nodes and save space by not requiring a redefinition of all the coordinates.

The IndexedFaceSet Node template is

DEF Cx_Px IndexedFaceSet { coord USE Cx coordIndex [d1 d2 d3 ...-1] }

This node defines all the indexes into a given coordinate set that composes each polygonal face of each geometric structure. By referencing a previously defined Coordinate node, all vertex values can be stored in one node. Each part will have an IndexedFaceSet node and each Coordinate node may contain vertices for multiple parts. An example section of a VRML (.wrl) file is included below:

```
DEF C9 Coordinate { point [ #./testd/vicplsROTb.geometry
-6.166104 0.570916 -0.116510
-6.166104 0.456971 -0.116510
...
-6.300978 0.050792 -0.569112
-6.300978 -0.050808 -0.569112
-6.046978 -0.050808 -0.569112
] } #./testd/vicplsROTb.geometry
DEF C9_P1 IndexedFaceSet { #./testd/vicplsROTb.geometry
  coord USE C9
  coordIndex [
0 1 2 3 -1
7 6 5 4 -1
...
2915 2914 2887 2886 -1
2916 2917 2918 2919 2920 2921 2922 2923 2924 2925 2926 2927 2928 2929 -1
2933 2932 2931 2930 -1
2934 2935 2936 2937 -1
] }
DEF C9_P2 IndexedFaceSet { #./testd/vicplsROTb.geometry
  coord USE C9
  coordIndex [
2938 2939 2940 2941 2942 2943 2944 2945 2946 2947 2948 2949 2950 2951 2952 2953 2954 2955 2956 2957 -1
```

```
2958 2959 2960 2961 2962 2963 2964 2965 2966 2967 2968 2969 2970 2971 2972 2973 2974 2975 2976 2977 -1
] }
DEF C9_P3 IndexedFaceSet { #./testd/vicplsROTb.geometry
  coord USE C9
  coordIndex [
2978 2979 2980 2981 2982 2983 2984 2985 2986 2987 2988 2989 2990 2991 2992 2993 2994 2995 2996 2997 -1
3017 3016 3015 3014 3013 3012 3011 3010 3009 3008 3007 3006 3005 3004 3003 3002 3001 3000 2999 2998 -1
2978 2998 2999 2979 -1
...
2996 3016 3017 2997 -1
2997 3017 2998 2978 -1
] }
DEF C9_P4 IndexedFaceSet { #./testd/vicplsROTb.geometry
  coord USE C9
  coordIndex [
3018 3019 3020 3021 3022 3023 3024 3025 3026 3027 3028 3029 3030 3031 3032 -1
3047 3046 3045 3044 3043 3042 3041 3040 3039 3038 3037 3036 3035 3034 3033 -1
3018 3033 3034 3019 -1
...
3052 3053 3049 3048 -1
3049 3053 3054 3050 -1
] }
```

Note that the Coordinate node contains vertices for four parts.

5.0 GAWK CONVERSION

The creation of the geometry portion of the VRML file is done with GAWK. This is a very useful scripting language that is available for UNIX and Windows operating systems from The Free Software Foundation ("www.gnu.org/software/gawk/gawk.html") or directly from Bell Labs ("cm.bell-labs.com/cm/cs/awkbook/"). Figure 3 outlines the AWK process.

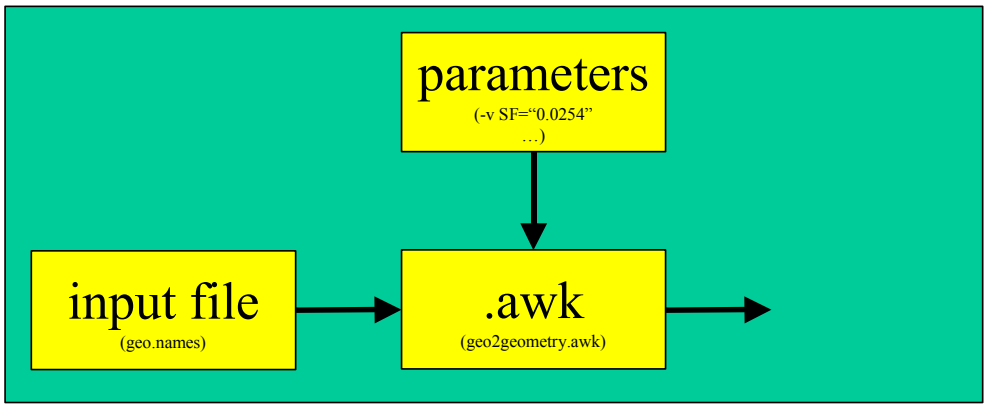

Figure 3 – GAWK process

Parameters are passed to the AWK script before it begins processing the input file. Output can be sent to the standard output or a specified file. An example of the calling structure with the script name "geo2geometry.awk" is given below

```
gawk -v SF="0.0254" -v SRC="./geo/" -v SUF=".geo" -v DES="./testd/" -f ../src/geo2geometry.awk geo.names
```

5.1 INPUT

The input file is named "geo.names". Its contents are just a list of files to convert. Its format can contain up to two columns separated by spaces. The first column is the Movie.BYU (or .geo) file name and the optional second column is "1" if normals are to be reversed on the geometry. An example is included below:

```
vicplsROTb
axle1ROTb
axle2ROTb 1
axle34ROTb
axle5ROTb
wheel1ROTb 1
retdrawbar
trailerbitspreadwtank2
RMS1.50-ROT
```

Notice the "1" after "axle2ROTb" and "wheel1ROTb" to indicate that normals are to be reversed.

5.2 PARAMETERS

Table 1 defines and describes the parameters passed to the GAWK script for processing the Movie.BYU files. The VRML file assumes all values are in SI units--meters for distance. Note that all parameters are passed as strings within double quotation marks.

Table 1 - Parameter Definitions

VARIABLE	DESCRIPTION	EXAMPLE
SF	Scale Factor. Conversion to meters.	"0.0254"
SRC	Source Directory.	"./geo/"
SUF	Movie.BYU File Suffix Extension.	".geo"
DES	Destination Directory.	"./testd/"

It is assumed that all file names listed in the "geo.names" input file have the same extension (in the example ".geo").

5.3 OUTPUT

The converted file will have the same name as in the "geo.names" file, but with a ".geometry" extension. This format is discussed in section 3.0.

While it is not considered directly part of the conversion process, output of a formatted version of the "geo.names" file is required for scene assembly as discussed in Section 1.0 and Figure 1. An example of this simple AWK program is given below

```
gawk '{printf "%3d   %s\n", NR,$1;}' geo.names > G.names
```

The "G.names" file is used for visual inspection of the geometry index number to ensure proper alignment with other VRML properties and is used in final scene assembly. An example of the "G.names" file format is given below.

```
  1   vicplsROTb
  2   axle1ROTb
  3   axle2ROTb
  5   axle34ROTb
  6   axle5ROTb
  7   wheel1ROTb
  8   retdrawbar
  9   trailerbitspreadwtank2
 10   RMS1.50-ROT
```

5.4 CODE SECTIONS

The GAWK script "geo2geometry.awk" is included in Appendix A. It begins with initializing variables. Then each name is checked to see if it is a duplicate. If it is, processing skips to the next file name. For each file name a flag is set to whether normals should be reversed on not. Processing for each file name is continued as follows:

1. Read header of Movie.BYU file
2. Read in parts data with **GeoReadParts(geo_nprt,fi)**
3. Read in points data with **GeoReadPoints(geo_npts,fi)**
4. Read in lines data that defines each polygon with **GeoReadLines(geo_nlin,fi,geo_ply,geo_eprts)**
5. Write out point data with **GeoWritePoints(fo,geo_npts,geo_ptsx,geo_ptsy,geo_ptsz)**
6. Write out part data with **GeoWriteParts(geo_nprt,fo,RNOR,geo_ply,geo_lin,geo_eprts,geo_bprts)**

Table 2 defines some important internal parameter definitions.

Table 2 - Internal Parameter Definitions

VARIABLE	DESCRIPTION	EXAMPLE
RNOR	Reverse Normals (1=yes, other =no).	1
FS	Field Separator.	" "
FIELDWIDTHS	Field Widths. (for fixed column width reading)	"8 8 8 8 8 8"

Reversing normals (RNOR=1) is accomplished by saving the sequence of polynomial vertices into an array and then writing them out in reverse order. (Without specific normals specified, the right-hand-rule is used to define them. Reversing the vertex order accomplishes this.)

6.0 SUMMARY/CONCLUSION

A simple script based conversion process between Movie.BYU and a generic ".geometry" format was described for use in scene assembly of VRML files. It should be noted that the scene assembly portion mentioned in section 1.0 could be done with X3D [3]. Currently, however, many advanced utilities, such as Cortona Movie Maker [4] will only work with VRML and therefore is the focus at this time.

CONTACT

The author is an engineer with the U.S. Army Research, Development and Engineering Command (RDECOM), located at the U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center (TARDEC). Interested parties can contact the author at the U.S. Army Tank-automotive and Armaments Research, Development and Engineering Center (TARDEC), ATTN: AMSRD-TAR-N/MS157, 6501 E 11 Mile Rd., Warren, Michigan 48397-5000, email: “bylsmaw@tacom.army.mil”.

REFERENCES

[1] Movie.BYU format, "lc.cray.com/doc/movie/".

[2] The Virtual Reality Modeling Language (VRML), ISO/IEC 14772-1:1997 and 14772-2:2002, "www.web3d.org". (The Virtual Reality Modeling Language consists of two parts. Part 1 (ISO/IEC 14772-1) defines the base functionality and text encoding for VRML. Part 2 (ISO/IEC FDIS 14772-2) defines the base functionality and all bindings for the VRML External Authoring Interface).

[3] X3D, ISO/IEC Draft 19776-1:200x, 19776-2:200x, 19777:200x, "www.web3d.org". (X3D encodings — ISO/IEC FDIS (Final Draft International Standard) 19776-1:200x (XML encoding) (.html) (.zip 220KB) 2004-09-26 Specifies the encoding of X3D files using the Extensible Markup Language (XML). X3D encodings — ISO/IEC FDIS (Final Draft International Standard) 19776-2:200x (Classic VRML encoding) (.html) (.zip 90KB) 2004-07-21 Specifies the encoding of the functionality and constructs defined in X3D using Classic VRML encoding. X3D language bindings — ISO/IEC FCD (Final Committee Draft) 19777:200x (.html) (.zip 143KB) 2003-05-15 Specifies the binding of the services in the X3D architecture to the ECMAScript programming language for use in X3D internal representation (Script nodes) and for external application access Specifies the binding of the services in the X3D architecture to the Java programming language for use in X3D internal representation (Script nodes) and for external application access).

[4] Parallelgraphics, Inc. "www.parallelgraphics.com".

DEFINITIONS, ACRONYMS, ABBREVIATIONS

RDECOM – U.S. Army Research, Development and Engineering Center
TACOM - U.S. Army Tank-automotive and Armaments Command
TARDEC - TACOM Research, Development and Engineering Center
NAC - National Automotive Center

APPENDIX A – GEO2GEOMETRY.AWK SCRIPT

```
#
# geo2geometry
# -v SF=0.254 (scale factor to meters)
# -v SRC="./geo/" (source directory)
# -v SUF=".geo" (source file extension suffix)
# -v DES="./vrml/" (destination directory)
#
# RNOR (reverse normals, 0 = no, 1 = yes)
#
BEGIN {
#---set fixed widths for Movie .BYU/.geo inputs
  fint="8 8 8 8 8 8 8 8 8";
  ffloat="12 12 12 12 12 12";
   FS = " ";
}
{
   fi = SRC$1SUF;

   if (fi in names)
   {
      print "Duplicate",fi;
   }
   else
   {
      names[fi]=1;
      RNOR = $2;  # set reverse normals flag
      fo = DES$1".geometry";
      print fi,"->",fo;

      FIELDWIDTHS=fint;
      getline < fi;
      geo_nprt = $1;
      geo_npts = $2;
      geo_nply = $3;
      geo_nlin = $4;

#---READ IN PARTS DATA
      GeoReadParts(geo_nprt,fi);

#---READ IN PTS
      GeoReadPoints(geo_npts,fi);

#---READ IN LINES AND CREATE POLY'S INDEX TO THEM
      GeoReadLines(geo_nlin,fi,geo_ply,geo_eprts);

#---Write ".geometry"
      print "Writing ",fo;
      printf "# %s\n",fi > fo;

      GeoWritePoints(fo,geo_npts,geo_ptsx,geo_ptsy,geo_ptsz);
      GeoWriteParts(geo_nprt,fo,RNOR,geo_ply,geo_lin,geo_eprts,geo_bprts);

#---reset for read of *.names file
      FS = " ";
      close(fi);
   }
}
END {
   print "Done.";
}

function GeoReadParts(geo_nprt,fi,   i,j)
{
   print "Reading Parts ",fi;
   FIELDWIDTHS=fint;
   i = 0;
   while (i < geo_nprt)
   {
      err = getline < fi;
      if (err <= 0) {print "---error reading",fi;exit;};
      for (j = 1; j <= NF; j=j+2)
      {
         geo_bprts[i]=$j;
```

```
         geo_eprts[i]=$(j+1);
         i = i + 1;
      }
   }
}

function GeoReadPoints(geo_npts,fi,  i,j)
{
   print "Reading Points ",fi;
   FIELDWIDTHS = ffloat;

   i = 0;
   while (i < geo_npts)
   {
      err = getline < fi;
      if (err <= 0) {print "---error reading",fi;exit;};
      for (j = 1; j <= NF; j=j+3)
      {
         geo_ptsx[i] = $j;
         geo_ptsy[i] = $(j+1);
         geo_ptsz[i] = $(j+2);
         i = i + 1;
      }
   }
}

function GeoReadLines(geo_nlin,fi,geo_ply,geo_eprts,   cntl,cntp,savi,i,j)
{
   print "Reading Lines",fi;
   FIELDWIDTHS = fint;

   cntl = cntp = 0;
   savi = -1;

   i = 0;
   while (i < geo_nlin)
   {
      err=getline < fi;
      if (err <= 0) {print "---error reading",fi;exit;};
      for (j = 1; j <= NF; j++)
      {
         geo_lin[i]=$j;
         if (savi < 0)
         {
            geo_ply[cntp] = i; # /* index zero based here for start of poly in lin[]*/
            savi = i;
         }
         if (geo_lin[i] < 0)
         {
            cntl = cntl + 1;
         }
         if (cntl == geo_eprts[cntp])
         {
            cntp = cntp + 1;  #/* start next poly index to lin[] */
            savi = -1;
         }
      i = i + 1;
      }
   }
}

function GeoWritePoints(fo,geo_npts,geo_ptsx,geo_ptsy,geo_ptsz,  i)
{
   print "Writing Points",fo;
   printf "#POINTS: %d\n",geo_npts >> fo;
   for (i=0; i < geo_npts; i++)
   {
      printf "%f %f %f\n", SF*geo_ptsx[i],SF*geo_ptsy[i],SF*geo_ptsz[i] >> fo;
   }
   printf "#END\n" > fo;
}

function GeoWriteParts(geo_nprt,fo,RNOR,geo_ply,geo_lin,geo_eprts,geo_bprts,    i,cntl,k,cnt,val,savri,savr,z)
{
   print "Writing Parts",fo;
   for (i=0; i < geo_nprt; i++)
   {
```

```
      printf "#PART: %d\n",i+1 >> fo;
      if (RNOR!=1)
      {
         cntl = 0;
         k = geo_ply[i];
         cnt = 0;
         do {
            val = geo_lin[k];
            if (val < 0)
            {
               val = -val;
               cntl = cntl + 1;
               printf "%d -1\n",val-1 >> fo; #/* make zero based */
            }
            else
            {
               printf "%d ",val-1 >> fo; #/* make zero based */
            };
            cnt++;
            k = k + 1;
         } while (cntl < (geo_eprts[i] - geo_bprts[i] + 1));
      }
      else
      {
         #/* reverse normals */
         # printf("---Reversing Normals for BODY:%s\n",par_bodname[ind]);
         savri=0;
         cntl = 0;
         k = geo_ply[i];
         cnt = 0;
         do {
            val = geo_lin[k];
            if (val < 0)
            {
               val = -val;
               cntl = cntl + 1;
               savr[savri]=val-1; /* make zero based */
               printf "%d ",savr[0] >> fo;
               for (z=savri;z>0;z--)
               {
                  printf "%d ",savr[z] >> fo; /* make zero based */
               }
               printf "-1\n" >> fo;
               savri=0;
            }
            else
            {
               savr[savri]=val-1; /* make zero based */
               savri++;
            }
            cnt++;
            k = k + 1;
         } while (cntl < (geo_eprts[i] - geo_bprts[i] + 1));
      }
      printf "#END\n" > fo;
   }
}
```