Appendix B Example Fly File

Overview

This appendix contains an example fly file, "`demo.fly`" (Figure B.1).

Figure B.1 Demo.Fly File

```
;;;Demo World Fly File
;;;National Center for Supercomputing Applications at
;;;University of Illinois in Urbana-Champaign

;;;PURPOSE
;;;     Controls a PolyView "flight" through the demo.hdf sample world. Camera tracks across ;;;
field of tetrahedrons and through an aisle of them, stops at a wall and pans 270
;;;     degrees to its left, then travels through a tunnel of cubes and zooms up to view the ;;;
world from on high. After a few orbits, it then flies back to its original position.

;;;     Initialize Position to edge of array.
               from abs .25 0.0 10
               at abs .25 .25 4.0

;;;     Move the at point to the end of the aisle.
               at abs 1.75 .25 .25

;;;     Move the camera to the other end of the aisle.
               from abs 1.75 0.0 10


               from abs 1.75 .25 .75

;;;     Pan the camera to the left 270 degrees.
               at abs .25 .25 .75
               at abs 1.75 .25 10
               at abs 20 .25 .75

;;;     Move the camera through the tunnel
               from abs 15 .25 .75

;;;     And pull the camera up to view the entire scene.
               from abs 15 2 .75
               at abs .25 .25 4
               from abs 15 8 .75

;;;     Circle the world.
               from abs .75 5 15
               from abs -15 5 .75
               from abs -.75 5 -15
               from abs 15 5 .75
```

Figure B.1 Demo.Fly File (Continued)

```
;;;     Swoop down over the top of the tunnel
                from abs 13 2 .75
                from abs 1.5 1.5 .75

;;;And then underneath the bottom plane.
                from abs -3.5 -4 8
                from abs -.5 -2 12

;;;     Then back to the original position.
                from abs .25 0.0 10

;;;     Track along the perimeter of the world.
                from abs 3 .5 10
                from abs 3 .5 2
                from abs 13 1 2
                from abs 13 1 -.5
                from abs -.5 0.0 -.5
                from abs -.5 1 10

;;;     Back to the original position.
                from abs .25 0.0 10
```

The file code “md.c” in Figure B.2 creates “demo.hdf”.

Figure B.2 File Md.c

```
/*
MD.C
NCSA Software Tools Group
November 29, 1989

DESCRIPTION
        This program creates a sample HDF per-vertex data file useful for testing
        PolyView. The file describes a number of simple three-dimensional objects and
        associated scientific data.


        The created file is named "demo.hdf" and contains the following vgroups and
        vdatasets:


        group1  Vgroup containing all of the following data

        px      X dimension data set
        py      Y dimension data set
        pz      Z dimension data set

        plist3  connectivity list for triangles only
        plist4  connectivity list for triangles and quadrilaterals

        random  random scientific data
        */

        /* INCLUDES */
        #include <stdio.h>
        #include <ctype.h>
        #include "df.h"
```

Figure B.2 File Md.c (Continued)

```
        #include "dfi.h"
```

```
        #include "vg.h"


        /* DEFINES */
        #define POINTS  5000
        #define VERTS   5000

        /* Random number generator information */
        #define MAX_RND       (1<<31)
        #define RND(min,max)  (double)(((((double)lcm_rnd())/(MAX_RND-1))*\
                              ((double) max - (double) min)) + (double) min)

        /* TYPES */


        /* GLOBAL VARIABLES */
               float   px[VERTS];
               float   py[VERTS];
               float   pz[VERTS];

        int    plist3[VERTS];
        int    plist4[VERTS];

        float  random[VERTS];

        int    np = 0;
        int     nv3 = 0;
        int     nv4 = 0;


        /* PROTOTYPES */
               unsigned long lcm_seed;
               void lcm_randomize(unsigned long);
               unsigned long lcm_rnd();

               void init_vars();
               void gen_data();
               void write_to_hdf();


        /* MAIN */
               main(argc, argv)
               int argc;
               char * argv[];
{
        /* Initialize global variables */
               init_vars();

        /* Generate the data for the file */
               gen_data();


        /* Write the data to the file */
               write_to_hdf();
}


        /* FUNCTION DECLARATIONS */

               void
```

Figure B.2 File Md.c (Continued)

```
        init_vars()
/* DESCRIPTION:  Reinitializes all global variables to prepare for generating
        another set of data.
        */
{
        nv3 = 0;
        nv4 = 0;
        np = 0;

}

        void tetra(float, float, float, float);
        void rpp(float, float, float, float, float, float);
        void
        gen_data()
/* DESCRIPTION:  Generates data describing test world that we want to display.
*/
{
        float x, y, z;

/* Create a field of tetrahedrons witha final row of rpps */
        for (x = 0.0; x < 4.0; x += 1.0) {
        for (z = 8.0; z >= 5.0; z -= 1.0) {
        tetra(x, 0.0, z, 0.5);
        }
        rpp(x, 0.0, 4.0,  0.5, 0.5, 0.5);
        }



/* Create an aisle of rpps */
        rpp(1.0, 0.0, 1.0,  0.5, 0.5, 2.5);
        rpp(2.0, 0.0, 1.0,  0.5, 0.5, 2.5);

/* Create an end for the aisle */
        rpp(1.0, 0.0, 0.0,  1.5, 0.5, 0.5);


/* Create a final tunnel of cubes */
        for (x = 3.0; x <= 14.0; x += 1.0) {
        for (z = 0.0; z <= 1.0; z += 1.0) {
        rpp(x, 0.0, z, 0.5, 0.5, 0.5);
        }
        }

        for (x = 2.5; x <= 14.5; x += 1.0) {
        for (y = -0.5; y <= 0.5; y += 1.0) {
        rpp(x, y, 0.5, 0.5, 0.5, 0.5);
        }
        }


        }

        int
        add_pt(x, y, z)
        float x, y, z;

/* DESCRIPTION:  Adds points x, y, and z to the px, py, and pz arrays,
respectively. Also adds random number to the random array. Incrementsn
p.
```

Figure B.2 File Md.c (Continued)

```
*/



{
        px[np] = x;
        py[np] = y;
        pz[np] = z;

        random[np] = RND(0.0, 1.0);

        np++;
        return np-1;
}


        void
        connect3(p0, p1, p2)
        int p0, p1, p2;

/* DESCRIPTION:  Connects points p0, p1, p2 in the plist3 and plist4
arrays. Increments nv3 and nv4. */

{
        plist3[nv3*3+0] = p0+1;
        plist3[nv3*3+1] = p1+1;
        plist3[nv3*3+2] = p2+1;

        plist4[nv4*4+0] = p0+1;
        plist4[nv4*4+1] = p1+1;
        plist4[nv4*4+2] = p2+1;
        plist4[nv4*4+3] = 0;

        nv3++;
        nv4++;
}


        void
        connect4(p0, p1, p2, p3)
        int p0, p1, p2, p3;

/* DESCRIPTION:  Connects points p0, p1, p2, p3 in the plist4 array.
Increments nv4. */
{
        plist4[nv4*4+0] = p0+1;
        plist4[nv4*4+1] = p1+1;
        plist4[nv4*4+2] = p2+1;
        plist4[nv4*4+3] = p3+1;

        nv4++;
}


        void
        tetra(x, y, z, a)
        float x, y, z, a;

/* DESCRIPTION:  Adds the vertices for a tetrahedron with corner (x,y,za
nd edge a to the px, py, and pz arrays, and connects them in plist3
and plist4. The plist4 entry is padded with a zero for the fourth
```

Figure B.2 File Md.c (Continued)

```
entry.
*/
{
        int p0, p1, p2, p3;

        /* Add the four points to the array */
        p0 = add_pt(x,y,z);
        p1 = add_pt(x+a,y,z);
        p2 = add_pt(x,y+a,z);
        p3 = add_pt(x,y,z+a);

        /* Connect the dots */
        connect3(p0, p2, p1);

        connect3(p0, p3, p2);
        connect3(p0, p1, p3);
        connect3(p1, p2, p3);
}
        void
        rpp(x, y, z, a, b, c)
        float x, y, z, a, b, c;

/* DESCRIPTION:  Adds the vertices for a rectangular parallelpiped
with near corner (x, y, z) and far corner (x+a, y+b, z+c) to the
px, py, and pz arrays, and connects them in plist4.
*/
{
        int p[8];

/* Add the points to the array */
        p[0] = add_pt(x, y, z);
p[1] = add_pt(x, y+b, z);
        p[2] = add_pt(x+a, y+b, z);
        p[3] = add_pt(x+a, y, z);
        p[4] = add_pt(x, y, z+c);
        p[5] = add_pt(x, y+b, z+c);
        p[6] = add_pt(x+a, y+b, z+c);
        p[7] = add_pt(x+a, y, z+c);

/* Connect the dots */
        connect4(p[5], p[6], p[2], p[1]);
        connect4(p[0], p[4], p[5], p[1]);
        connect4(p[0], p[3], p[7], p[4]);
        connect4(p[2], p[6], p[7], p[3]);
        connect4(p[7], p[6], p[5], p[4]);
}


        void
        write_to_hdf()
/* DESCRIPTION:  Writes the data sets to the a new hdf file.
*/
{
DF              * f;
VGROUP          * vg;
VDATA   * vs;
int             vgid, vsid;
int             n;

/* Create the HDF file */
        f = DFopen("demo.hdf", DFACC_ALL, 0);
```

Figure B.2 File Md.c (Continued)

```
/* Initialize Vgroup and Vsubgroup ids for creating new items */
        vgid = -1;
        vsid = -1;

/* Create a Vgroup and name it "group1" */
        vg = (VGROUP *) Vattach(f, -1, "w");
        Vsetname(vg, "group1");

/* Create px, py, and pz vdatasets and insert them in group1 */
        vs = (VDATA *) VSattach(f, -1, "w");
        VSsetname(vs, "px");
        VSsetfields(vs, "px");

        VSwrite(vs, px, np, NO_INTERLACE);
        Vinsert(vg, vs);
        VSdetach(vs);
        vs = (VDATA *) VSattach(f, -1, "w");
        VSsetname(vs, "py");
        VSsetfields(vs, "py");
        VSwrite(vs, py, np, NO_INTERLACE);
        Vinsert(vg, vs);
        VSdetach(vs);
        vs = (VDATA *) VSattach(f, -1, "w");
        VSsetname(vs, "pz");
        VSsetfields(vs, "pz");
        VSwrite(vs, pz, np, NO_INTERLACE);
        Vinsert(vg, vs);
        VSdetach(vs);


/* Create random scientific values dataset */
        vs = (VDATA *) VSattach(f, -1, "w");
        VSfdefine(vs, "random", LOCAL_FLOATTYPE, 1);
        VSsetname(vs, "random");
        VSsetfields(vs, "random");
        VSwrite(vs, random, np, NO_INTERLACE);
        Vinsert(vg, vs);
        VSdetach(vs);


/* Create plist3 and plist4 vdatasets and insert them in group1 */
        vs = (VDATA *) VSattach(f, -1, "w");

        VSfdefine(vs, "plist3", LOCAL_INTTYPE, 3);
        VSsetname(vs, "plist3");
        VSsetfields(vs, "plist3");
        VSwrite(vs, plist3, nv3, NO_INTERLACE);
        Vinsert(vg, vs);
        VSdetach(vs);

        vs = (VDATA *) VSattach(f, -1, "w");
        VSfdefine(vs, "plist4", LOCAL_INTTYPE, 4);
        VSsetname(vs, "plist4");
        VSsetfields(vs, "plist4");
        VSwrite(vs, plist4, nv4, NO_INTERLACE);
        Vinsert(vg, vs);
        VSdetach(vs);


/* Close up the file */
```

```
        Vdetach(vg);
        DFclose(f);
```

Figure B.2 File Md.c (Continued)

```
}


/* RANDOM NUMBER GENERATOR FUNCTIONS */

        void lcm_randomize(seed)
        unsigned long seed;
{
        lcm_seed = seed;
}


        unsigned long lcm_rnd()
{
        unsigned long rnd_val;


        rnd_val = lcm_seed;
        lcm_seed = ((314159269 * lcm_seed) + 453806245) % MAX_RND;
        return (rnd_val);
}
```