Computer Graphics and Animation as a Tool for Biomolecular Visualization

Jody Vykoukal
University Undergraduate Fellow, 1992-1993
Department of Biochemistry and Biophysics
Texas A&M University

Approved:

Fellows Advisor ____________________

Edgar Meyer

Honors Director ____________________

[illegible]

Introduction

The following describes my efforts as an Undergraduate Fellow during the 1992-1993 academic year. I completed my fellowship in the Department of Biochemistry Biographics Laboratory with Professor E.F. Meyer serving as my advisor. Over the course of the past two semesters, I have worked to develop and improve algorithms used in the visualization of biomolecules via x-ray diffraction studies. I would like to thank Dr. Meyer and his colleague, Dr. Stan Swanson, for their assistance and support; through their kindness and guidance, I have learned a great deal.

Background

Solutions to many biochemical questions may be found in the pockets, loops, turns, and clefts of biomolecule tertiary and quaternary structure. X-ray crystallography has been employed over the past three decades as the premier method for mapping proteins and other molecules of biological significance. Fifteen years ago, determining the structure of a single protein appeared to be a lifetime project. However, advances in computer technology have given structural biologists the ability to determine the molecular structure of a protein in a matter of months or even weeks.

Visualization programs such as FRODO perform millions of calculations enabling the crystallographer to transform raw crystallographic data into three dimensional topology maps representing molecular structure (Pines, 1990). Interactive computer graphics work stations allow visualization and manipulation of these molecular structures. Interactive computer graphics are presently used to model the fit of drugs and inhibitors to receptors (Takahashi et at., 1988; Bode et al., 1989; Takahashi et al., 1989). In addition, animated simulations resulting from molecular dynamics calculations performed by a supercomputer are currently utilized to extend the insight provided by molecular modeling. Visualization of a variety of chemically interesting conformational changes including shifts in ring positions and rotations of amino acid side chains in proteins has been made

(Swanson et al., 1989; Geller et al., 1990).

Advances in the field of cybernetics have increased the computational speed and graphics capabilities of much of the equipment available to crystallographer. A definite need exists to develop software which is able to exploit new computer technologies and maximize the capabilities of existing equipment. The Biographics Laboratory at Texas A&M is equipped with an Evans and Sutherland PS330 system linked to a VAX 11/750 as well as an ESV workstation with a self-contained 20 MIPS processor running in a UNIX environment. I have worked with both of these machines, and have become most familiar with the visualization software they utilize.

PRONTO is an improved visualization program developed in the Biographics Laboratory. PRONTO consists of an X-windows implemented slave which acts as an interface between the user and FRODO. While PRONTO offers increased efficiency via the multitasking capabilities of X-windows, the ability to handle the FRODO plot command has never been integrated into the slave. Furthermore, users of the Biographics Laboratory are unable to generate PostScript files for laser printer output; they must use a plotter to generate hard-copies of topology maps for publication. Dr. Meyer and I decided that I could best familiarize myself with the equipment in his laboratory by writing a translator to convert Hewlett Packard Graphics Language (HP-GL) plotter files to PostScript and integrating this algorithm into a PRONTO command, providing a much needed plot option as well as an on screen print previewer.

X-ray Crystallography

Studies of the diffraction of x-rays by crystals were first performed by W.L. Bragg in 1913. He used the crystallographic technique to demonstrate the atomic structure of the NaCl crystal. Fifteen years later, Kathleen Lonsdale used crystallography to show that the benzene ring is a regular hexagon. Crystallographic studies performed by Rosalind Franklin and Maurice Wilkins were used by Watson and Crick to elucidate the structure of DNA.

The primary objective of crystal analysis by x-ray diffraction is to determine the structure of a molecule on atomic level. Once the coordinates of each atom in a molecule are known, information concerning atomic distances, bond angles, and planarity of atom groups can be calculated. Generation of a molecular topology map offers new insight into the relationships between molecule structure and function.

When a crystal is subjected to an x-ray beam, the electrons of individual atoms diffract the incoming radiation, causing the beam to scatter. The scattered radiation is intercepted by a detection device such a photographic film or photodetector array. A diffraction pattern is formed as a result of this scatter; however, the information concerning the phases of the diffracted waves is lost because only the intensities of the diffracted beams are recorded. Once the relative phases of the diffracted waves are known (derived, deduced, guessed or measured indirectly), it is possible to use fast Fourier transform methods to recombine mathematically the scattered x-rays into a computed three-dimensional electron density map. These maps can be be used to determine the atomic coordinates in three dimensions (Glusker and Trueblood, 1985). Interactive computer graphics systems such as FRODO and PRONTO may be used to further visualize the complex, three dimensional map and manipulate the model to fit the map.

Project Progress

In early September I met with Dr. Meyer and his group to discuss the course my Fellows Project should take. We decided that I should first work on the HP-GL to PostScript translator. Writing the program familiarized me with much of the hardware and software used in the Biographics Laboratory, a step necessary to produce more complex programs, such as a the PRONTO plot command and print previewer.

My first task in writing the translator was to analyze several sample topology map files written in HP-GL and deduce the actions of the various HP-GL commands. Simultaneously, I began to study the command library and syntax of the PostScript language. By mid October I had developed an algorithm to convert HP-GL commands to comparable PostScript commands. Before I could code the algorithm, it was necessary to learn how to use the VAX EDT text editor, C compiler, and linker. This took about two weeks. In early November I began to code the conversion program. As soon as I had a workable prototype of the program, I tested it by trying to convert about twenty-five HP-GL files from the Biographics Laboratory. After a bit of modification the program was able to convert all of the files successfully.

In early December I met with Dr. Meyer and Dr. Swanson to discuss the next stage of the project. We decided that I should try to integrate my conversion program into a plot command executable from PRONTO. I began work on the second phase of the project in late January. I reviewed the FRODO source code in order to find the existing plot command. I discovered that FRODO uses an algorithm to combine various atom coordinate data files into a single generic plot file based upon viewer orientation, scaling, and other parameters. Another algorithm translates the generic file into a HP-GL file. I needed to modify the slave so that these algorithms can be called in order to produce a publication ready image from within PRONTO. It was necessary for me to learn basic UNIX commands in order to review the source code for the PRONTO slave. In February I began to review the PRONTO source code in an effort to understand its mechanism of action. I have determined the general mechanism of the slave and have made changes to the C source code which add a plot option to the PRONTO menu. At this time I am attempting to integrate my PostScript translator and provide the desired PRONTO plot command and print previewer.

Future Goals

I plan to complete the integration of my PostScript translator and plot command into the PRONTO slave. Also, I have recently acquired a copy of Ghostscript from the Free Software Foundation. Ghostscript is a PostScript interpreter which provides preview of PostScript files in the X-windows environment. I would like to utilize this program to provide on screen print preview of topology maps generated through the PRONTO plot command. Such improvements will allow Dr. Meyer to utilize the superior technology present in laser printers as well as cut down on the time necessary to produce publication ready topology maps.

Conclusion

Biomolecular visualization through interactive computer graphics has indeed revolutionized the x-ray crystallographic technique. As a result, an understanding of many important biological functions has been realized. The demand for improved visualization algorithms which utilize the latest computer technologies continues. Over the course of the past year, I have studied and attempted to improve some of the visualization algorithms used to generate biomolecule topology maps. I have learned a great deal about the workings of the crystallographer's tools, and gained a sincere appreciation for the art of crystallography.

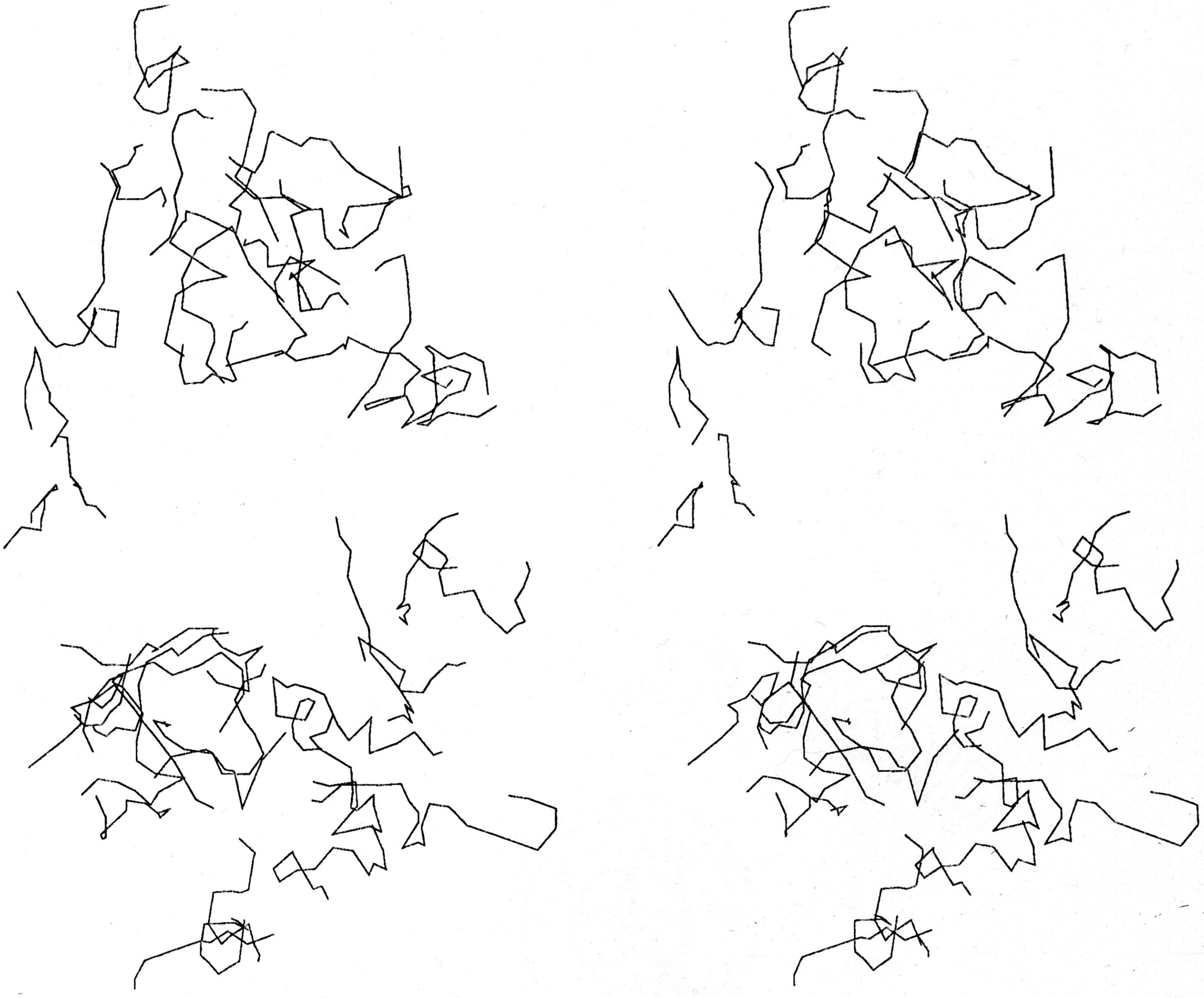

Figure 4. S.M. Swanson LONG 10 IN CUT15, G=3 (CUTHEL.PLT)

Figure 1. Stereo Topology Map printed on a LaserWriter II from a PostScript file generated through my HP-GL translation program.

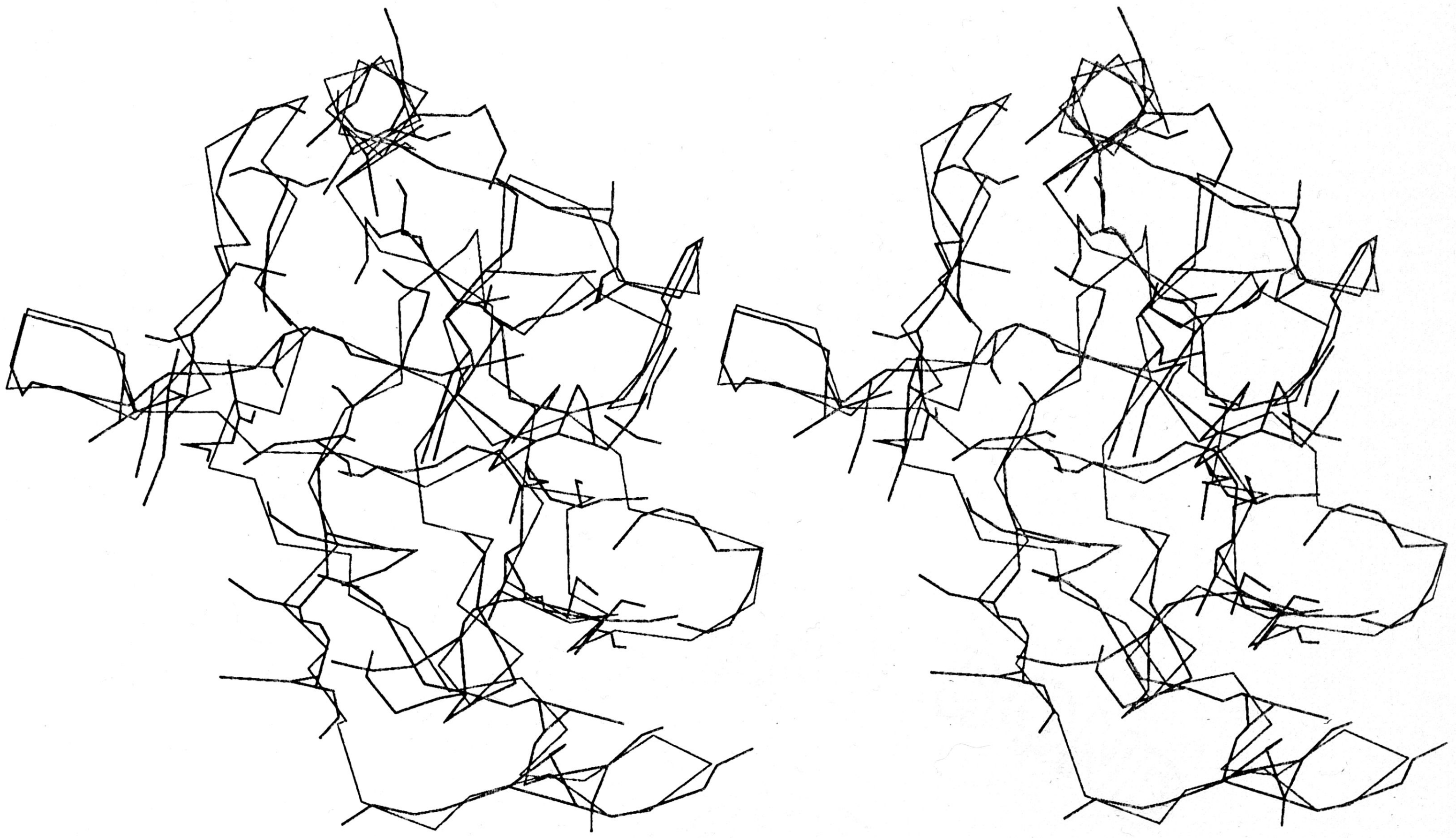

Figure 2. Stereo Topology Map printed on a LaserWriter II from a PostScript file generated through my HP-GL translation program.

REFERENCES

Adobe Systems Incorporated. (1990) *PostScript Language Reference Manual.* Addison Wesley.

Adobe Systems Incorporated. (1990) *PostScript Language Tutorial and Cookbook.* Addison Wesley.

Bode, W., Meyer, E.,Jr., & Powers, J. C. (1989) *Biochemistry 28*, 1951-1963.

Deisenhofer, J., Remington, S. J., & Steigmann, W. (1985) *Methods Enzymol. 115B* , 303-323.

Geller, M., Swanson, S. M., & Meyer, E. F.,Jr. (1990) *J. Am. Chem. Soc. 112*, 8925-8931.

Glusker, J.P., & Trueblood, K.N. (1985) *Crystal Structure Analysis, A Primer.* Oxford University Press.

Groff, J.R., & Weinberg, P.N. (1983) *Understanding Unix.* Que Corporation.

Jack, A., & Levitt, M. (1978) *Acta. Crystallogr. 34A*, 931-935.

Jones, T. A. (1978) *J. Appl. Cryst. 11*, 268-272.

Pines, M., *ed.* (1990) *Finding the Critical Shapes*, Report from the Howard Hughes Medical Institute, Delaware.

Steigmann, W. (1974) Ph.D. Thesis, T.U. Munchen.

Swanson, S. M., Wesolowski, T., Geller, M., & Meyer, E. F. (1989) *J. Mol. Graphics 7*, 240-244.

Takahashi, L. H., Radhakrishnan, R., Rosenfield, R. E.,Jr., & Meyer, E.,Jr. (1989) *Biochemistry 28*, 7610-7617.

Takahashi, L. H., Radhakrishnan, R., Rosenfield, R. E.,Jr., & Meyer, E.,Jr., Trainor, D. A., Stein, M. (1988) *J. Mol. Biol 201*, 423-428.

APPENDIX

HP-GL to PostScript Conversion C Source Code

```
/**********************************************************************/
/*                  HPGL to PostScript CONVERT-O-MATIC                */
/*  Biographics Laboratory, Department of Biochemisry and Biophysics  */
/*        Texas A&M University.  November 1992.  Jody Vykoukal.       */
/*                                                                    */
/*   NOTE:  This program only supports the following HPGL commands:   */
/*          IN, LB, LT, SI, SP, PU, PD, VS.                           */
/**********************************************************************/

#include <stdlib.h>
#include <stdio.h>
#include <ctype.h>

void fill_carousel();
void gate_1(int key_1);
void error_exit(int gate, int key_2);
void IN();
void LB();
void LT();
void SI();
void SP();
void PU();
void PD();
void VS();
void landscape();
void build_file_name();
double get_number(FILE *file);
double scale(double number);

FILE *input, *output;              /* Global Varibles */
int statement = 0;
double color[8], size[8];
char in_file[100], out_file[100];

main()
{
   int key_1;

   fill_carousel();

   printf("\nEnter name of file to be converted: ");
   scanf(" %s", in_file);
   build_file_name();

   if((input = fopen(in_file, "r")) != NULL)
   {
      if((output = fopen(out_file, "w")) !=NULL)
      {
         landscape();
         printf("\n\nConverting...");
         while((key_1 = fgetc(input)) != EOF)
         {
            statement++;
            /* Digest non alphabetic characters */
            while(key_1 != EOF && !(isalpha(key_1)))
               key_1 = fgetc(input);

            if (key_1 != EOF)
               gate_1(key_1);
         }
      }
      else(printf("\nError opening %s.", out_file));
   }
   else(printf("\nError opening %s.", in_file));

   printf("\n\n%d statements converted.", statement);
   fprintf(output,"\nstroke\n\nshowpage"); /* Must use stroke to print */
   fclose(input);
}



/* Main Switching Function */
/* Send to specific conversion function. */

void gate_1(int key_1)    /* Get first letter of HPGL command */
{
   int key_2;

   key_2 = toupper(fgetc(input));
   switch (toupper(key_1))
   {
      case 'I':                /* I statements */
         switch(key_2)
         {
            case 'N':
               IN();
               break;
            default:
               error_exit('I', key_2);
               break;
         }
         break;

      case 'L':                /* L statements */
         switch(key_2)
         {
            case 'B':
               LB();
               break;
            case 'T':
               LT();
               break;
            default:
               error_exit('L', key_2);
```

```
                  break;
              }
              break;

          case 'P':              /* P statements */
              switch(key_2)
              {
                  case 'D':
                      PD();
                      break;
                  case 'U':
                      PU();
                      break;
                  default:
                      error_exit('P', key_2);
                      break;
              }
              break;

          case 'S':              /* S statements */
              switch(key_2)
              {
                  case 'I':
                      SI();
                      break;
                  case 'P':
                      SP();
                      break;
                  default:
                      error_exit('S', key_2);
                      break;
              }
              break;

          case 'V':              /* V statements */
              switch(key_2)
              {
                  case 'S':
                      VS();
                      break;
                  default:
                      error_exit('V', key_2);
                      break;
              }

          default:
              error_exit(key_1, '_');
              break;
      }
  }
  /* End of switching loop */
```

```
/* Statement Conversion Functions */

void IN() /* Initialization of Device */
{
   int text;

   fprintf(output,"\nnewpath");
}

void LB() /* Label */
{
   int text;

   fprintf(output,"\n  (");
   (void)fgetc(input);

   while((text = fgetc(input)) != '\3')
      fprintf(output,"%c", text);

   fprintf(output,") show");
}

void LT() /* Select Line Type */
{
   int pattern_number;

   pattern_number = (int)abs(get_number(input));

   switch(pattern_number)  /* Approximate HPGL patterns */
   {
      case 1:
         fprintf(output,"\n  [1 9] 0 setdash");
         break;
      case 2:
         fprintf(output,"\n  [5 5] 0 setdash");
         break;
      case 3:
         fprintf(output,"\n  [7 3] 0 setdash");
         break;
      case 4:
         fprintf(output,"\n  [8 1 1 1] 0 setdash");
         break;
      case 5:
         fprintf(output,"\n  [6 2 2 2] 0 setdash");
         break;
      case 6:
         fprintf(output,"\n  [4 2 2 2 2] 0 setdash");
         break;
      default:
         break;
   }
}
```

```
void PD() /* Move to x,y with Pen Down */
{
   double x, y;

   x = scale(get_number(input));
   y = scale(get_number(input));

   fprintf(output,"\n  %3.4lf %3.4lf lineto", x, y);
}

void PU() /* Move to x,y with Pen Up */
{
   double x, y;

   x = scale(get_number(input));
   y = scale(get_number(input));

   fprintf(output,"\n  %3.4lf %3.4lf moveto", x, y);
}

void SI() /* Absolute Character Size Instruction */
{
   double width, height;
   int points;

   width = get_number(input);
   height = get_number(input);

   /* Convert height from cm to points */
   points = (int)(28.35 * height);
   if (points < 1) points = 1;

   fprintf(output,"\n\n/Times-Roman findfont");
   fprintf(output,"\n%d scalefont setfont\n", points);
}

void SP() /* Select Pen */
{
   int pen;

   pen = (int)get_number(input) - 1;

   if (pen != -1)
   {
      fprintf(output,"\nstroke\n\nnewpath");
      fprintf(output,"\n  %1.4lf setgray", color[pen]);
      fprintf(output,"\n  %2.4lf setlinewidth", size[pen]);
   }
}

void VS()  /* Not necessary for PostScript.  Just get the number. */
{
   double speed;
   speed = get_number(input);
}

void error_exit(int gate, int key_2)
{
   printf("\nUnknown HP-GL Command: %c%c.", gate, key_2);
   printf("\nStatement No. %d.", statement);
   exit(0);
}

double get_number(FILE *file)
{
   int digit, n = 0;
   char number[50];

   while(((digit = fgetc(file)) != ',') && digit != ';' && digit != '\n')
   {
      number[n] = digit;
      n++;
   }
   number[n] = '\0';

   return(atof(number));
}

double scale(double number)
{
   return(number * 0.070866);   /* Convert HPGL units to Points (72/inch) */
}

void fill_carousel()
{
   FILE *carousel;
   int pen;
   double points;

   if((carousel = fopen("carousel.dat", "r")) != NULL)
   {
      for(pen = 0; pen <= 7; pen++)
      {
         color[pen] = get_number(carousel);

         /* Get pen width in mm and convert to point width */
         points = (2.835 * get_number(carousel));

         size[pen] = points;
      }
      fclose(carousel);
   }
   else
   {
      printf("\nFATAL ERROR.  Could not open carousel.dat.");
      exit(0);
   }
}

void landscape()
{
   char view;

   do
   {
      fflush(stdin);
      printf("\nLandscape or Portrait?  Enter P or L: ");
      view = toupper(getc(stdin));
   }  while (view != 'L' && view != 'P');

   if (view == 'L')
      fprintf(output, "\n  612 0 translate\n  90 rotate");
}

void build_file_name()   /* Build new file name with .ps extension */
{
   int index = 0;
   char character;

   do
   {
      out_file[index] = in_file[index];
      character = in_file[index];
      index++;
   }  while (character != '\0' && character != '.');

   out_file[index] = 'p';
   out_file[index+1] = 's';
   out_file[index+2] = '\0';

   printf("\nTranslation to be stored in %s\n", out_file);
}
```