MINICOMPUTER SPEAKEASY IN A DYNAMIC GRAPHICS ENVIRONMENT

M. J. Bailey and D. C. Anderson[1]
Purdue University

ABSTRACT

Speakeasy is an extensible interpretive computer language that enjoys widespread usage on a number of large-scale timesharing systems. An adaptation of Speakeasy implemented on a PDP-11 minicomputer using the UNIX timesharing system is described. The minicomputer version contains many of the basic processor features of the distributed system while maintaining a high degree of language compatibility. The development of the system, however, has stressed adapting and extending the language and processor for research and instruction in a dynamic interactive graphics environment. The description emphasizes basic implementation features and examples of the graphic extensions.

Keywords and Phrases: interactive graphics, interpretive language, minicomputer, Speakeasy

CR Categories: 4.13, 4.20, 8.2

1. INTRODUCTION

The past few years have seen a remarkable infiltration of minicomputers into virtually all fields of industry, research, and instruction. The ready availability of these small-scale processors has created a large demand for quality software of the caliber that is available on the established large-scale computing systems.

One such software package is the Speakeasy language {1}[2]. Speakeasy is an extensible, interpretive, Fortran-based language that enjoys widespread usage on a number of large-scale timesharing systems. The large user population and continued interest in Speakeasy attest to its virtually unlimited extensibility and applicability to a broad spectrum of computer problem-solving needs.

These qualities focused attention on Speakeasy as a powerful addition to the minicomputer graphics software and facilities at Purdue University. This paper describes a minicomputer adaptation of Speakeasy called Speakec, which is designed for use in an engineering research and instruction environment emphasizing interactive computer graphics.

2. ENVIRONMENT

The environment that guided the development of Speakec is the Computer Aided Design and Graphics Laboratory. The laboratory is a research and teaching facility in the School of Mechanical Engineering. Among its computing and display facilities are a Digital Equipment Corporation PDP 11/40 minicomputer with 88K of memory, three Imlac PDS programmable refresh-display minicomputers, a Tektronix 4014 storage tube terminal, a Calcomp plotter, and an Altair 8800 microprocessor.

The majority of the projects in the laboratory involve interactive refresh graphics using the Imlac PDS-1. The Imlac contains 8K of memory and communicates with the PDP-11 over a high-speed parallel interface, forming the dual-processor system shown in Figure 1.

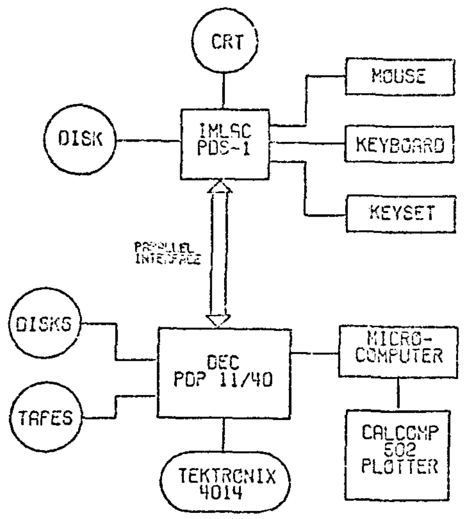


Figure 1 The GRAFIC System Configuration

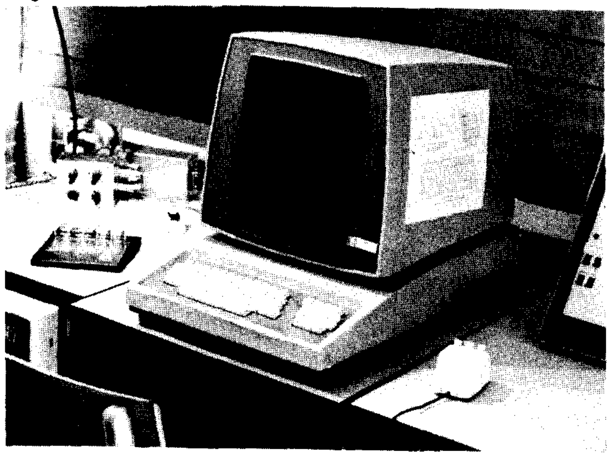
The Speakec refresh graphics commands are based on the GRAFIC system {2,3}, which is a

[1] Authors' Address: Computer Aided Design and Graphics Laboratory, School of Mechanical Engineering, Purdue University, W. Lafayette, IN 47907.

[2] Numbers in braces ({}) designate references at the end of the paper.

versatile and highly interactive package of Fortran-callable subroutines originally designed to run under the DOS single-user operating system.

Under GRAFIC, individual "segments" of a dynamic display list maintained in the Imlac are manipulated by the user's program executing in the PDP-11. The PDP-11 program may also request data from several graphic input devices interfaced to the Imlac. Two of these are the Stanford Research Institute "mouse" and the five-fingered keyset, shown at the right and left of the Imlac in Figure 2.



Fiqure 2 The Imlac, Keyset and Mouse

The mouse consists of two rotary potentiometers that are perpendicularly oriented to provide continuous X-Y position feedback as it slides along a surface. The keyset is played like chords on a piano and allows the user to have touch control over his program. In addition to these devices, the Imlac keyboard may be used to enter numbers and alphanumeric commands. The mouse and keyset are used simultaneously to interact with the graphics program while the user focuses his attention on the Imlac screen.

3. SYSTEM DESCRIPTION

3.1 The UNIX Environment

Speakec runs on a PDP 11/40 minicomputer under the UNIX operating system {4}. It is written in the C programming language {5}, a high level PL/I-type language that overcomes many of Fortran's shortcomings.

The UNIX and C environment greatly supported and encouraged the Speakec development. Despite initial hesitation to use a language other than Fortran, it soon became apparent that C with UNIX offered for more potential for a demanding system like Speakeasy. UNIX is a versatile timesharing system which is enjoying increased support from a growing users' group of well over 100 members. From the user's viewpoint, UNIX provides excellent programming support which allows the programmer to make optimal use of all the system features from C, thus greatly facilitating device communication and file manipulation. The C language structure makes programming smooth and natural, and its comfortable readability is instantly appreciated. For example, the Speakec processor to date with its 90 linkules contains less than 50 statement labels! Another interesting aspect of the Speakec implementation is the fact that no assembly coding or system modifications were required. The entire processor including the linkule loader executes in under 17K of 16-bit memory and the source is a readable 1500 lines long. From the viewpoint of the computer, the C compiler produces very compact and efficient PDP-11 machine code.

3.2 The Speakec Processor

Speakec is by no means a complete implementation of the distributed version of Speakeasy. A number of language details were bypassed during this initial effort to hasten the completion of a basic processor which was used to experiment with the graphic linkules. Consequently, only "kind=real" variables are allowed at the present time. Other kinds, such as complex, logical, integer, and string will be implemented later as the development continues.

As in the original Speakeasy, the Speakec central program functions as an arithmetic expression processor. In addition to parsing statements and controlling basic communication with the user, the processor also manages a dynamic data area called "named storage" and invokes function calls to "linkules". These two features are the heart of the Speakeasy system and distinguish it from other interpretive systems. The implementation of named storage and linkules illustrates the contributions of the UNIX and C environment to Speakec.

The operation of the execution time linkule loader takes full advantage of the flexible program memory expansion capabilities of UNIX. The loader algorithm is relatively straightforward in C, and requires only 500 words of memory. When a function call is invoked by the processor, control is passed to the C procedure "link". The arguments are the linkule name and the standard linkule calling arguments (described below). Link first examines a threaded list of in-memory routines for the requested linkule. If found, the routine is entered using a standard procedure call. If the linkule is not in memory, link proceeds to search the attached linkule libraries. These libraries consist of C programs that have been previously compiled, loaded, and archived into a UNIX library file. A standard UNIX loader option retains the relocation information on the load image. Speakec currently permits up to three libraries to be specified when the processor is invoked.

Once the linkule has been located in a library a first-fit search is performed on a threaded list of unused memory slots. If this search fails, the necessary additional program memory is requested from UNIX using a simple system call. Failure at this stage forces link to sequentially free in-memory linkules until sufficient unused memory is available.

The overall process begins by initializing named storage to a user-specified size, and then setting the dynamic linkule pointers to the resulting address limit. As linkules are executed, the program size grows, possibly reaching the 28K-word PDP 11/40 addressing limit.

The Speakec calling sequence passes roughly the same information to the linkule as does Speakeasy, but takes advantage of certain C features. The arguments and their declarations appear in the linkule as shown in Figure 3. While it is not expected that the reader understand C, certain C constructs stand out intuitively as being particularly useful. The variable "varstack" is an array containing information pertaining to each of the arguments passed to the linkule. Each element of the "varstack" array is a structure, that is, varstack [1] actually consists of a ten-byte variable ("arg"), four integer variables ("fkind, fklass, fnrows, and fncols"), and a pointer to another structure ("point"). Thus if the linkule writer wishes to reference the "klass" of the third argument of the linkule, he would reference:

```
varstack[3].fklass
```

The asterisk (*) before the variable "point" indicates that it is a pointer to a structure, not a structure itself. The structure to which it points is named "element" and is defined in Figure 3 just before "varstack".

```
(ans,noargs,icol,varstack,lolimit,variable,delete,
iquery)
       char ans[10];
       int *noargs;
       int icol[10];
       char *lolimit;
       struct element {
              char name[10];
              int kind;
              int klass;
              int nrows;
              int ncols;
              int offset;
              float data[1];
       };
       struct funcstk {
              char arg[10];
              int fkind;
              int fklass;
              int fnrows;
              int fncols;
              struct element *point;
       } varstack[40];
       struct element *(*variable)();
       int (*delete)();
       struct element *(*iquery)();
```

Figure 3   The Linkule Argument List

The "element" structure clearly shows how each Speakec working variable is stored in the named storage area. Each variable consists of a ten-byte name, an integer kind, klass, nrows, and ncols, and a floating point array of values. The "offset" is the length of the storage area for the working variable and is used to locate the next working variable in memory. If one wants to reference the sixth data value of the variable which was the first argument to the linkule, he would reference:

```
varstack[1].point->data[6]
```

The compiler also provides capabilities that ease linkule writing and help maintain compatibility through system changes. Additional files can be inserted in a file at compile time by the statement:

```
#include "file"
```

The arguments and declarations shown in Figure 3 are kept in a file named SUBARGS.H which the linkule writer may simply "#include" in his linkule without typing the lengthy list. C also allows a limited macro expansion facility. For example, the statement,

```
#define SCALAR 0
```

permits a variable's "klass" to be tested against the symbol SCALAR instead of the value 0. This makes the linkule very readable, easy to understand, and less dependent on specific system conventions. A list of appropriate "#define"s is contained in a file named DEFFIL.H which the linkule writer should "#include" in his linkule.

With all of this in mind, Figure 4 shows a typical linkule written in C. This linkule computes the average of whatever argument is passed. The only "klass" checking insures that the argument is not a scalar.

```
#include "DEFFIL.H"
struct element *average
#include "SUBARGS.H"
/**********
**  This linkule computes the average of whatever
**   is passed to it.  The only klass checking is
**   to reject a scalar argument
***********/
{
       struct element *p,*p1;
       register j,nwords;
       float sum;
/*setup returned variable and get pointer to it*/
       p=(*variable)(ans,REAL,SCALAR,1,1)
       if( (j=p) == 0) {
              icol[0]=NOCORE;
              return(0);}
/*make sure was not passed a scalar */
       if(varstack[1].fklass == SCALAR) {
              icol[0]=KLASSERR;
              return(0);}
/*get pointer to passed variable and determine
length*/
       p1=varstack[1].point;
       nwords=p1->nrows * p1->ncols;
/*determine sum*/
       sum=0.;
       for(j=1,j<=nwords;j++)
              sum=+ p1->data[j];
/*store average*/
       p->data[1]=sum/nwords;
/*return pointer to average variable*/
       return(p);
}
```

Figure 4   The Linkule AVERAGE

When a Speakec linkule returns to the processor, it returns a pointer to the data area it just created. If the returned pointer is zero, an appropriate error message is printed by the main program depending on the returned value of icol[0]. These errors are identified by symbol definitions that are also contained in the file DEFFIL.H and are "#included" in each linkule.

A more detailed description of the internal implementation details and operating procedure is given in the Speakec User's Guide {6}.

4. GRAFIC INTERACTION WITH SPEAKEC

The potential of the dynamic graphic capabilities in Speakec will be demonstrated using interactive object definition and visualization.

The linkule SKETCH is invoked by the Speakec statement:

```
object=sketch(maxpts[,z])
```

SKETCH allows the user to define an object in an X-Y plane with all points having a constant z coordinate, specified as the optional second argument (default: z=0). The argument "maxpts" is the maximum number of points the user may enter and is used to allocate temporary storage for the entered coordinates.

When SKETCH begins, a full screen crosshair appears on the Imlac. To enter a line, the user moves the mouse to position the crosshair and types a 2, 3, 4, or 5 on the Imlac keyboard, specifying a solid, blank, dotted, or point line, respectively to be drawn to the current point from the previous point. A keyset strike or reaching the maximum number of points causes the linkule to return an nX4 matrix, where n is the number of points entered. The four columns represent x, y, z, and pencode, respectively. An example of a SKETCH sequence is shown in Figure 5.

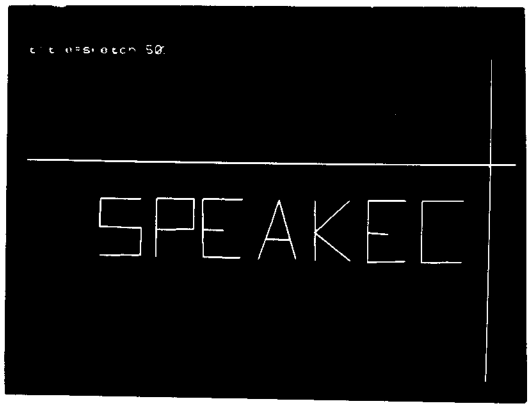


Figure 5 An Example of the SKETCH Linkule

Three dimensional (3D) perspective interaction is made possible with the SKETCH3D linkule:

```
object=sketch3d(maxpts)
```

This linkule is based on the 3D interaction work of J. A. Brewer {7}. SKETCH3D is similar to SKETCH except that the cursor is a 3D perspective view of the principle axes. The cursor is constrained to move in one dimension only. The movement direction is changed by striking the x, y, or z keys on the Imlac keyboard. The returned matrix is of the same format as in SKETCH, and the same conditions terminate the linkule.

The matrices returned by SKETCH and SKETCH3D may be transformed to Speakeasy's "vertex to connectivity" form by:

```
connect(object,v,c)
```

where "v" is a three column vertex matrix and "c" is a two column connectivity matrix.

The matrices produced by CONNECT can be used with the linkules DRAW3D and REVOLVE. DRAW3D displays the 2D projection of the given 3D object specified in the arguments from the current eye position. DRAW3D is capable of producing a picture on the Imlac, the Tektronix, or the Calcomp.

Dynamic views of the object can be produced by:

```
revolve(v,c,axis,angi,angf,dang)
```

Revolve rotates the eye position about the specified axis, redrawing the object at each step. "angi", "angf", and "dang" are the initial angle, final angle, and angle increment, respectively. Of course only the Imlac can be used for this dynamic motion.

Figures 6 through 9 illustrate a SKETCH3D and REVOLVE sequence.

```
bracket=sketch3d(10)
```

In Figure 6, the 3D cursor is in the X-Y plane with movement restricted to the Y direction. By alternating lines in the Y and X directions a rectangle is formed in Figure 7. Switching to movement in the Z direction, Y direction, and then Z direction again, the 90° angle bracket shown Figure 8 is quickly created.

Figure 9 shows the REVOLVE linkule invoked by:

```
connect(bracket,vbr,cbr)
revolve(vbr,cbr,y,0.,360.,10)
```

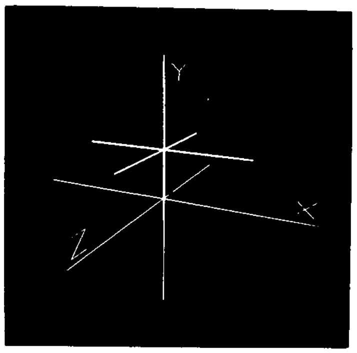


Figure 6 The 3D Cursor on the Y Axis

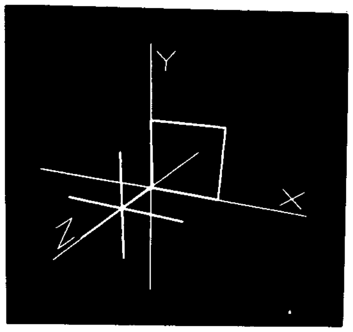


Figure 7 Sketching a 3D Bracket

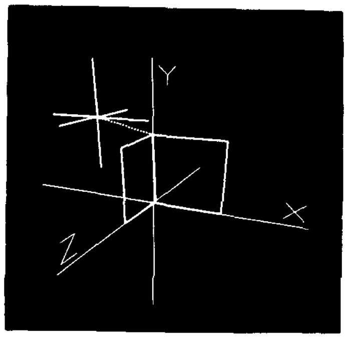


Figure 8 The Completed Bracket

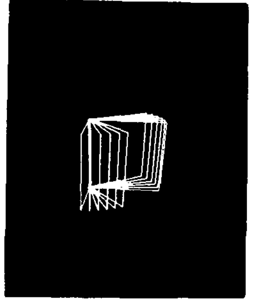


Figure 9 Using the REVOLVE Linkule

To enhance 3D visualization, a linkule was created to perform Warnock hidden line removal {8} on a 3D object. The application of this linkule to the angle bracket is shown in Figure 10.

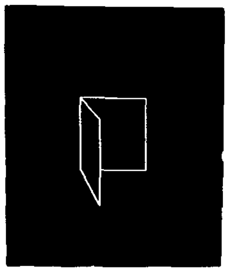


Figure 10 Results of Warnock Linkule

Additional three dimensional data generation is accomplished by defining transformations. Figure 11 shows object #1, a line segment, being defined on the X axis by SKETCH3D. A linkule is then called which produces a 4x4 transformation matrix defining a 30° rotation about the Z axis. Another linkule applies this matrix six times to object #1, automatically producing object #2, a semi-circular arch in Figure 12.

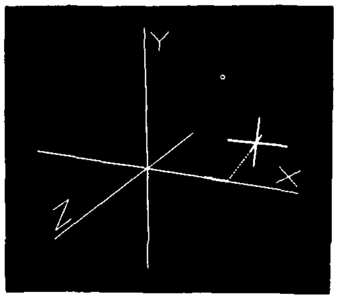


Figure 11 Sketching a Cross Section

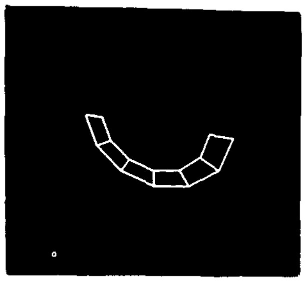


Figure 12 Generated Arch

Another transformation is defined as a 1.5 inch displacement in the Z direction. This transformation is applied to object #2 three times producing the final object, a semi-circular trough in Figure 13.

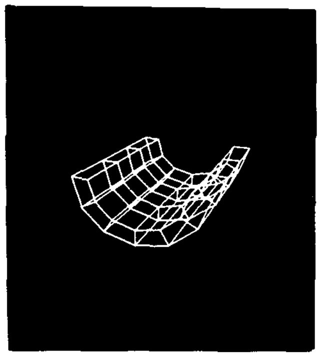


Figure 13 A 3D Generated Trough

Current goals call for application of this data generation technique to supply input to finite element analysis linkules. The output from such linkules would then be returned to the Speakec processor for graphical display of stresses and deformations.

5. CONCLUSIONS

Minicomputer Speakeasy complements program-oriented graphics by fulfilling a need for "non-programming" computer graphics. Often graphics capabilities are desired, but it is inappopriate for the inexperienced user to totally familiarize himself with graphics packages to program the application. Speakec provides immediate and more pleasant access to interactive graphics experience without the intermediate system impositions required to prepare and execute even the simplest of graphic programs.

The refresh graphics capability is an experimental feature that will receive concentrated attention in the future. To those accustomed to the rapid interaction rates of the simpler DOS Fortran-based system, Speakec graphics is extremely slow under moderate or heavier system loads. Future related research efforts will focus on increasing system throughput using distributed processing techniques with additional hardware.

As the response problems are solved, Speakec will become an effective tool for a wide variety of interactive graphic application areas. Current plans call for the addition of linkules for curve and surface manipulation, continuous tone hidden-surface processing, and finite element mesh generation, analysis, and display.

Minicomputer Speakeasy appears to have a promising future, especially for the graphics research and instruction environment at Purdue. As the popularity of minicomputers continues to increase, Speakec should lend itself well to future environments and needs.

ACKNOWLEDGMENTS

The authors extend their sincere thanks to Dr. Stan Cohen for his kind advice and encouragement during the development of Speakec. His enthusiasm and openness are the foundation on which the project was supported. We also thank Dr. John Brewer for his ideas and work in 3D graphical interaction and for introducing us to Speakeasy and Dr. Cohen.

REFERENCES

{1} Cohen, Stan and Pieper, Steven C., The SPEAK-EASY-3 Reference Manual/Level Lambda, Argonne National Laboratory Report ANL-8000 (May 1976).

{2} GRAFIC User's Manual, The Computer Aided Design and Graphics Laboratory, School of Mechanical Engineering, Purdue University, West Lafayette, Indiana (June 1976).

{3} Anderson, D. C. and Belleville, R. L., "Two Approaches to On-line Graphics Systems," to be published in Computers and Graphics, Pergamon Press.

{4} Ritchie, Dennis and Thompson, Ken, "The UNIX Time-Sharing System," Communications of the ACM, 17, 7 (July 1974), 365-375.

{5} Ritchie, Dennis, "C Reference Manual," from Documents for Use with the UNIX Time-Sharing System by Bell Laboratories, 6th edition.

{6} Anderson, D. C. and Bailey, M. J., SPEAKEC User's Guide, The Computer Aided Design and Graphics Laboratory, School of Mechanical Engineering, Purdue University, W. Lafayette, Indiana (August 1977).

{7} Brewer, J. A., "Three Dimensional Design by Graphical Man-Computer Communication," Ph.D. Thesis, The Computer Aided Design and Graphics Laboratory, School of Mechanical Engineering, Purdue University, W. Lafayette, Indiana (May 1977).

{8} Newman, W. M. and Sproull, R. F., Principles of Interactive Computer Graphics, McGraw-Hill, New York, New York, 1973.