Advances in Engineering Software **17** (1993) 95–103

Graphics display for graphics data management systems

T.V. Hromadka II

Boyle Engineering Corporation, PO Box 3030, Newport Beach, California 92658-9020, USA

&

M.J. Braksator

Williamson & Schmid, 15101 Red Hill Avenue, Tustin, California 92680, USA

Recent applications of computer software to water resources and environmental master-planning studies include use of graphical displays in order to disseminate information. In this paper, a simple-to-use graphics display program is presented which enables a user to display graphical slides, stored on disk, to the CRT. Such a program enables a user to link graphics slides to text data by use of a read-only text display routine. Application of the provided program in a Graphics Data Base Management System is considered as a case study. Computer code is provided for the graphics slide display.

Key words: GIS, water resources, modeling, graphics, interactive.

GRAPHICS DATA BASE MANAGEMENT SYSTEM: APPLICATION TO MASTER PLANS OF CITY FLOOD CONTROL SYSTEMS

An integrated hydrology/hydraulics/planning/deficiency-analysis Master Plan of Drainage computer model is considered as a case study for applications of a Graphics Data Base Management System (or GDBMS). The computer modeling approach evaluates each link of the Master Plan of Drainage for deficiencies with respect to several defined street flow criteria, and determines mitigation measures of parallel and replacement systems. Because different hydraulic systems have different flow velocity characteristics, hydrology estimates are recomputed as the master plan is developed. In general, a city master plan typically involves about 2000–4000 links and hydrologic subareas, and generates considerable quantities of data that become manageable in a GDBMS environment.

The entire Master Plan is represented by graphics layers in AutoCAD format, which allows for rapid communication of master plan data and estimates in graphical form. Two applications are developed:

Application 1: Graphical representation of data, and access to a data base retrieval system, which is noneditable, and which can be published and distributed to the public.

Application 2: Graphical database storage, and editing via an AutoCAD environment, wherein hydrologic, planning, topographic, and geographic data are accessible for processing in AutoCAD, and thence transferable to the Master Plan of Drainage computer model, with accesss to a data base retrieval system.

In the following, each major element of the GDBMS will be discussed. An application to an example Master Plan of Drainage will be used to demonstrate graphical display opportunites.

COMPUTERIZED MASTER PLAN OF DRAINAGE AND GRAPHICS DATA BASE MANAGEMENT SYSTEM

The total Master Plan of Drainage software package and data base system contains numerous elements and components that span several technical fields, including data base management, geographic information systems (or GIS), hydrologic/hydraulic computer modeling, graphical data base management, flood control engineering and planning, among others. In the following is provided a brief survey of the key elements of the total software package.

Coupled hydrologic modeling technique

Most flood control agencies at city, county, or state level require specific procedures for the calculation of flood flow quantities. Often the procedure may involve the use of two or more estimates, depending upon conditions such as watershed size. In Southern California, several county flood control districts require use of two flood flow estimation techniques dependent upon catchment area, namely, the rational method for areas smaller than about one square mile, and the design storm unit hydrograph method for areas larger than about one square mile. The transition between techniques has been coupled into an integrated computerized Master Plan of Drainage model, enabling the development, for the first time, of an integrated hydrologic computer model with one pass of the analysis, rather than two separate studies. As a result of coupling hydrologic techniques into just one computer model, single-system analysis is available for use in preparing Master Plans of Drainage and upgrading the master plan, thereby greatly reducing the complexity, review process, and cost involved.

The Master Plan of Drainage software contains internal editing and computational elements that involve 152 hydraulic and hydrologic submodels and global modeling commands. The software enables analysis of an integrated open-channel or closed-conduit flood control system on a study-wide basis.

Graphical data base

Several data base layers will be required to complete any hydrologic study. These layers will be created individually; however, they may be viewed simultaneously to show any hydrologic information desired. These layers include:

(1) base map consisting of contours and streets right-of-way;
(2) watershed boundary to define study boundaries;
(3) drainage reservations to define alignments;
(4) existing facilities to define alignments;
(5) street flow to determine existing flows;
(6) alignments defined by layers 3, 4, and 5;
(7) subarea boundaries defined by layers 5 and 6;
(8) overall mapping divides for final report;
(9) land use map;
(10) hydrologic soil group map;
(11) rainfall isohyetal map;
(12) hydrologic nodal points defined by layers 6 and 7;
(13) hydrologic element type to define routing parameters.

Some layers, such as the base map, drainage reservations, existing facilities, land use, hydrologic soil group, and rainfall isohyetal maps may be available in digital form. If these layers are not available in digital form then they can either be digitized or scanned. The layers specifically related to the development of a Master Plan of Drainage can either be digitized from a marked-up hard copy or directly using AutoCAD.

Primary hydrologic parameters used in the Master Plan of Drainage computer model are land use, hydrologic soil group, rainfall, and hydrologic subarea topographic data such as area, length of water course, and elevation. In general, a study is discretized into subareas that are 10–20 acres in size. These subareas require definition as to each of the parameters listed above. Additionally, maps are needed in order to communicate these data. By obtaining in digital form or actually digitizing the land use maps, hydrologic soil group maps, rainfall maps, and subarea maps, not only is a digital/graphical representation available for display, but the data can then be processed by a 'polygon processor' in order to partition the subareas into the intersection of all of the graphical layers. Geographic location is provided by use of street layout layers, right-of-way maps and freeway maps. The graphics data base is used to prepare hard-copy maps for reports, as well as graphical layers for display on the computer monitor. Figures 1 and 2 show hard copies of example graphical layers for a Master Plan of Drainage. Figure 1 depicts the land use map and Figure 2 depicts the hydrologic soil group map.

An acceptable base map is chosen and information such as subarea boundaries, alignments, and node numbers is added. This information is entered on a watershed basis. Once all this information is added, any desired layers are overlaid to create a hydrology map. Since these maps are created using AutoCAD, they can be reproduced at any scale.

From each watershed map, the boundary is taken and overlaid with other watershed boundaries to create quadrant maps. These quadrant maps, along with an index map, are the navigational tools by which the user can locate any specific location in the study area.

Polygon processor

The use of geographic information systems (GIS) has become widespread in many facets of engineering and planning, among other fields. A key element of a GIS is the ability to intersect graphical layers so that the several forms of information are resolved into 'cells' wherein all parameters are constant. Figure 3 depicts the resolution of several graphical layers of information into homogeneous cells.

In the Master Plan of Drainage, each subarea requires definition of land use, hydrologic soil group, and rainfall, and the proportions of each within the subarea. The polygon processor performs this important task, and then develops a data base for use in the Master Plan of Drainage computer model. The subarea data are stored in tabled formats, on a subarea basis, indexed according to

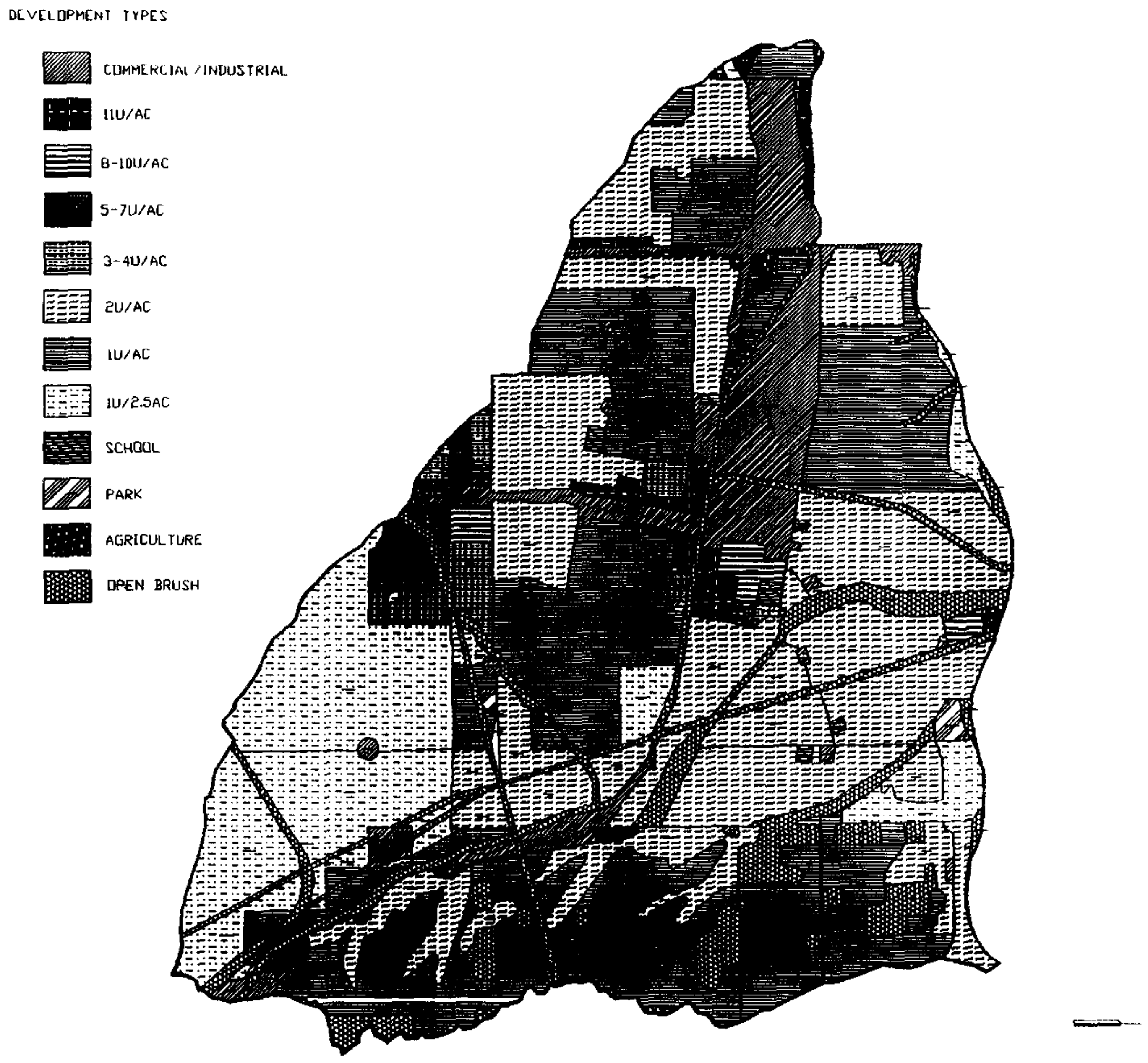

Fig. 1. Example land use map.

subarea number. Thus, the retrieval of a specific subarea number will access these several data, automatically developed by the polygon processor.

Master Plan of Drainage data base

The Master Plan of Drainage may be represented, in a data base form, as a collection of nodes (specific points along the catchment flood control system), and subareas (10–20 acres in size). All information computed by the Master Plan of Drainage, such as deficiency system mitigation needs, flow quantities, hydraulic properties, streetflow characteristics, flood control system characteristics, hydrologic parameters, and costs, among others, are stored in agency-designed tabled form in a data base indexed according to node number, link number, and subarea number. Also stored are data entered directly into the data base such as flood control system history, age, and so forth. Once the data base is assembled, the data base may be linked to the graphical data base which displays the digital graphical layers constructed for the polygon processing (i.e. multiple use of a data base form), while allowing easy access to the Master Plan of Drainage data base.

Graphics Data Base Management System

The graphical data base and retrieval software and the

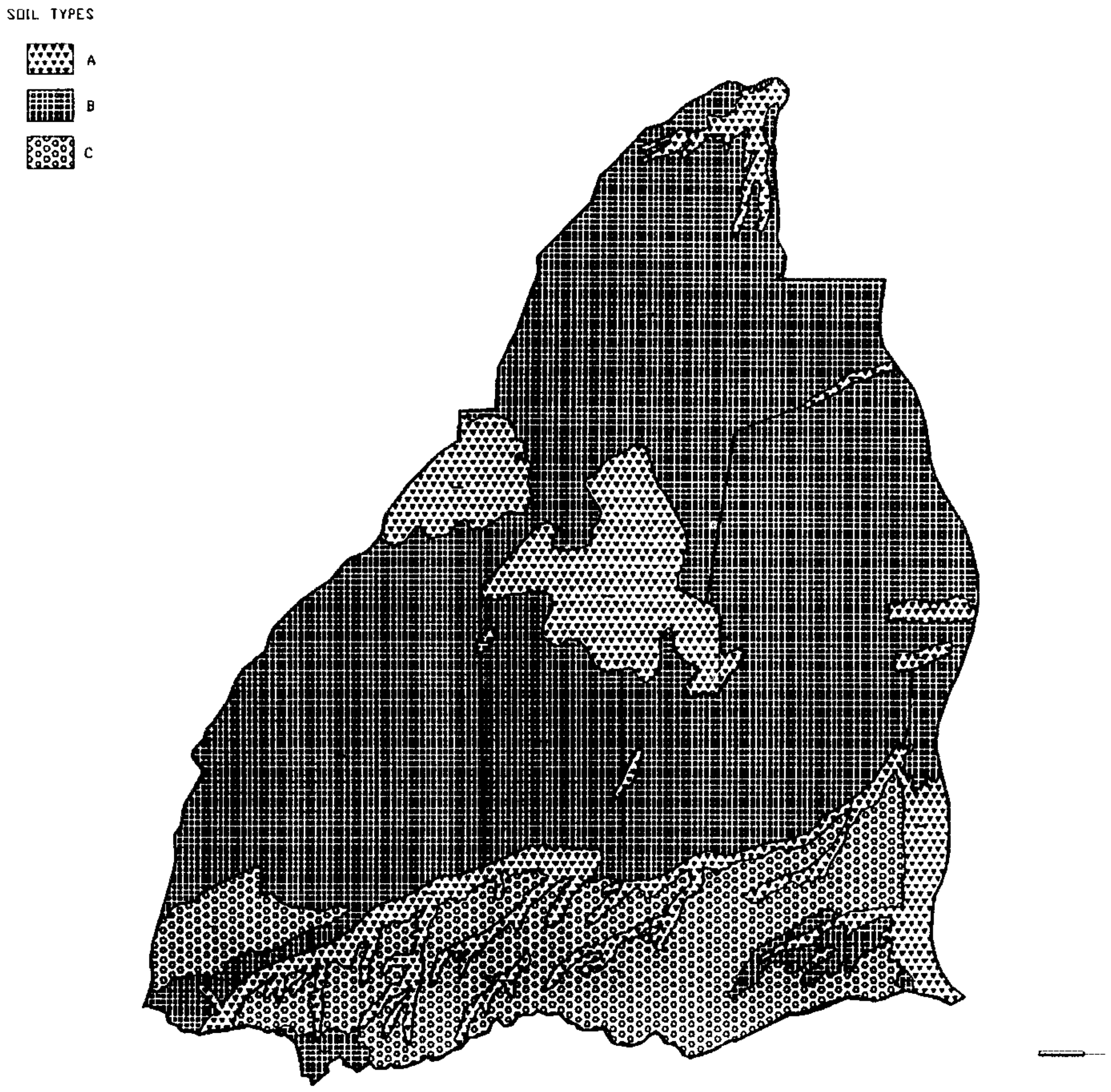

Fig. 2. Example hydrologic soil group map.

Master Plan of Drainage hydrologic/hydraulic computer software are coupled together to form the Graphics Data Base Management System. Each of the above software packages are developed specifically for this application, and do not require the use of other software packages.

Two applications are developed. The first application, or Application 1, enables a publication of the Master Plan data base for distribution to the public. Using 'slides' (i.e. monitor images stored in the graphics data base), the entire study can be resolved into graphics slides of about one-half a square mile in size, showing hydrologic master plan nodes, subareas, links, streets, land use, and hydrologic soil group, among other designed data. Each slide is indexed to successively larger maps so that by selecting quadrants from the monitor, one is able to navigate through the city to a selected point. Additionally, each slide is cross-referenced to a Master Plan of Drainage data base map that stores all the data associated with the slide appearing on the monitor. Figure 4 shows a slide of a data base map which appears on the monitor. Data base operating commands are displayed on the monitor screen, enabling the user to access the slide images. Software, in AutoLISP, for slide displays is contained in the Appendix.

This first application provides significant communica-

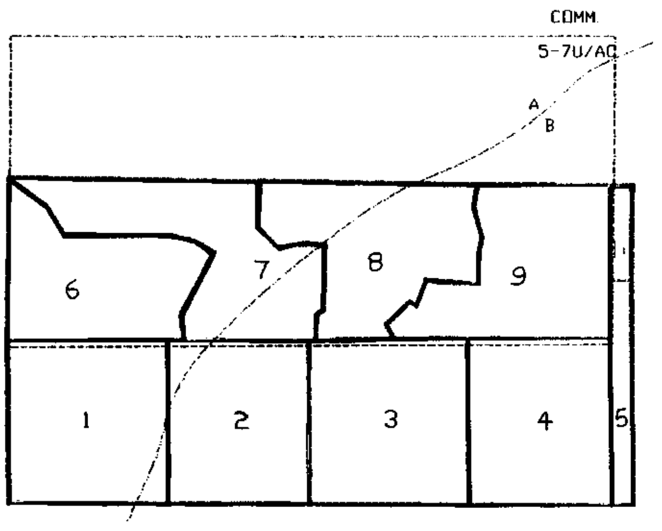

Fig. 3. Overlay of land use, hydrologic soil group and subareas for polygon processing.

tion opportunites for the agency to both the public and the technical sectors. The engineering and planning communities can access the data base for other technical needs, and also inspect the Master Plan of Drainage without reviewing the usual report documents (which typically run to several volumes). The public can inspect the Master Plan, and access information that would otherwise be unavailable.

Application 2 is the actual Management System which includes all the features of Application 1, plus the ability to upgrade the Master Plan of Drainage due to changes in system requirements, land use, and hydrologic parameters, among other factors. Because the agency can perform the upgrade, the master plan can be kept current, enabling up-to-date drainage fee assessment to be developed on an on-going basis.

AutoCAD slide display software

An element of the GDBMS is a slide display program which reads stored AutoCAD slide data and displays these data onto a CRT. The Appendix provides a slide display AutoLISP program. The program documentation is contained throughout the source code, and is self-explanatory.

CONCLUSIONS

A graphics data base management system for computerized Master Plans of Drainage is developed. Two applications are prepared which enable the agency to upgrade the Master Plan in the future, and to publish

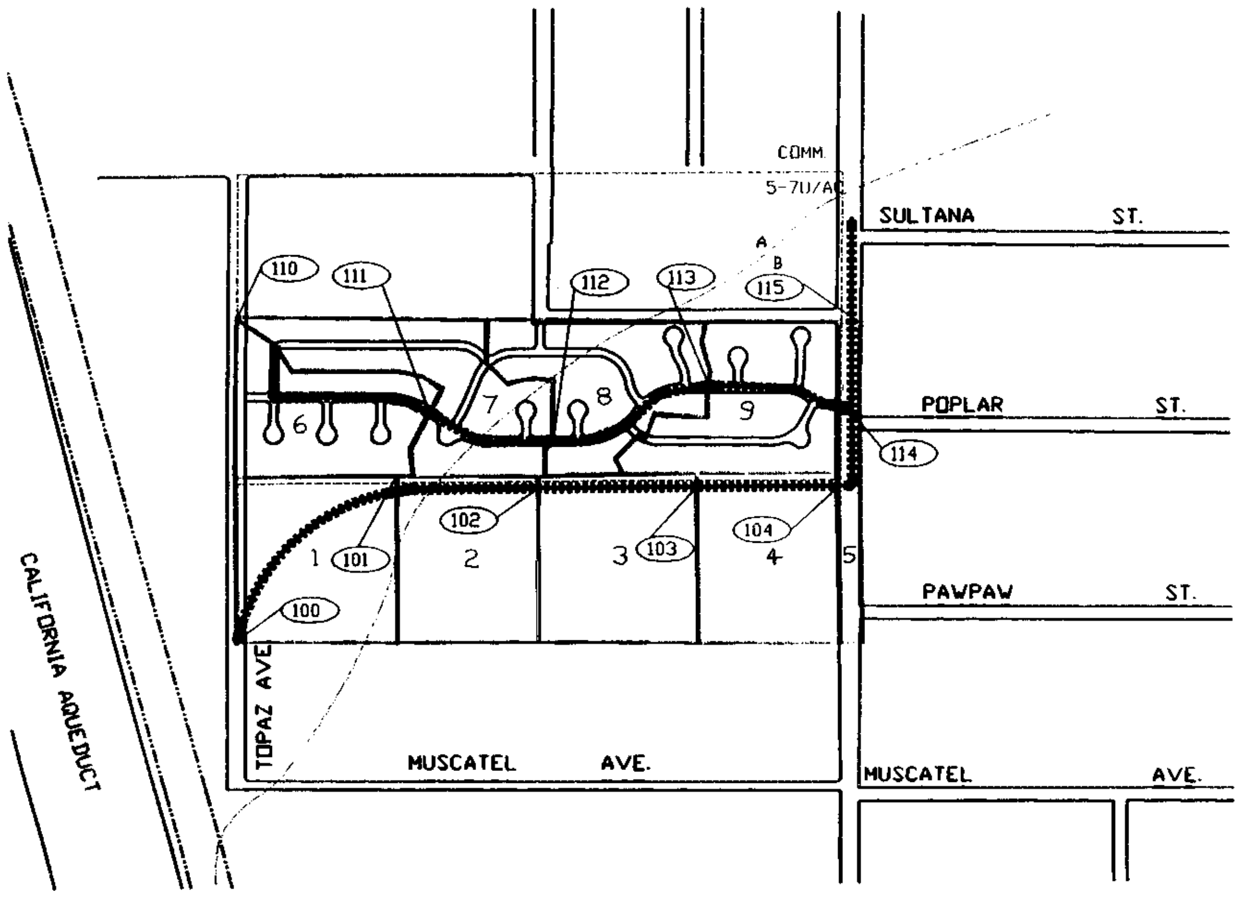

Fig. 4. Data base map display.

the Master Plan in computer graphics form for distribution to the public. Because of the ease of communication opportunities afforded by this approach, the utility in Agency public information programs may be significant.

BIBLIOGRAPHY

1. Hromadka II, T.V. *Computer Methods in Urban Hydrology, Rational Methods and Unit Hydrograph Methods*, Lighthouse Publications, Mission Viejo, CA, USA, 1983.
2. Hromadka II, T.V., Durbin, T.J. and DeVries, J.J. *Computer Methods in Water Resources*, Lighthouse Publications, Mission Viejo, CA, USA, 1985.
3. Hromadka II, T.V., McCuen, R.H. & Yen, C.C. *Computational Hydrology in Flood Control Design and Planning*, Lighthouse Publications, Mission Viejo, CA, USA, 1987.
4. Hromadka II, T.V. *Computational Hydraulics of Irregular Channels*, Lighthouse Publications, Mission Viejo, CA, USA, 1988.
5. Hromadka II, T.V., Durbin, T.J. & DeVries, J.J. *Computer Methods in Water Resources and Environmental Engineering*, Lighthouse Publications, Mission Viejo, CA, USA, (in press).

APPENDIX
GENERAL APPROACH SUMMARY
APPLICATIONS 1 & 2

```
 /* vslide - a program to display AutoCAD slide files */


#include <graphics.h>
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>

#define offsetvector 251
#define endoffile 252
#define solidfill 253
#define commonendpoint 254
#define newcolor 255

void getheader();
int verifyheader();
void setaspect();
int getrecord();
void drawvector();
void drawoffset();
void newcolor();
void drawcommon();
void solidfill();
void Initialize(void);
void ScreenViewport(void);

int headerrec[31];
int datarec[8];
int curX=0, curY=0, curColor;
int     GraphDriver;
int     GraphMode;
double  AspectRatio;
int     MaxX, MaxY;      /*highest x and y dots on the screeen*/
int     MaxColors;
int     ErrorCode;
struct  palettetype palette;
struct PTS { int x, y; };
FILE *fp;

int main(int argc, char *argv[])
{
   char thefile[20], tempchar;
   int stat, i;
   if (argc < 2)
   {
      printf ("Usage: vslide <slide filename>\n", argv[0]);
      exit(1);
   }
   Initialize();
   for (i = 1; i < argc; i++)
   {
      strcpy(thefile,argv[i]);
      if (strstr(thefile, ".") == NULL)
         strncat(thefile,".sld",4);
      if ((fp = fopen(thefile, "rb")) == NULL)
      {
         closegraph();
         printf ("Can't open %s\n", thefile);
         exit(1);
      }
      getheader();
      if (verifyheader() != 0)
      {
         cleardevice();
         closegraph();
         printf ("%s is not a slide file\n", thefile);
         exit(1);
      }
      cleardevice();
      setaspect();
      do {stat = getrecord();} while (stat != endoffile);
      fclose(fp);
      do {tempchar = fgetc(stdin);} while (tempchar != '\n');
   }
   closegraph();
}

void Initialize(void)
{
   int xasp, yasp;
   GraphDriver = DETECT;
   initgraph(&GraphDriver, &GraphMode, "");
   ErrorCode = graphresult();
   if (ErrorCode != grOk)
   {
      printf("Graphics System Error: %s\n", grapherrormsg(ErrorCode));
      exit(1);
   }
   getpalette(&palette);
   MaxColors = getmaxcolor() + 1;
   MaxX = getmaxx();
   MaxY = getmaxy();
   getaspectratio(&xasp, &yasp);
   AspectRatio = (double)xasp / (double)yasp;
}

void ScreenViewport(void)
{
   setviewport(0,0,639,497,1);
}

void getheader()                /* first 31 bytes of the .SLD file composes */
{                               /* a header with info about the hardware and*/
   int ch, i;                   /* graphics screen settings.               */
   for (i = 0; i < 31; i++)
   {
      headerrec[i] = fgetc(fp);
   }
}

int verifyheader()
{
 int i;
 char f_header[] = {0x41, 0x75, 0x74, 0x6F, 0x43, 0x41, /*<---  current values */
                    0x44, 0x20, 0x53, 0x6C, 0x69, 0x64, /*for the file header  */
                    0x65, 0x0D, 0x0A, 0x1A, 0x00};      /*in AutoCAD release 11*/
 for (i = 0; i < 17; i++)
 {
    if (headerrec[i] != f_header[i]) /* Bytes in the header should always */
       return (-1);                  /* be the same. verifyeader() checks */
 }                                   /* the validity of the .SLD file.    */
 return(0);
}

void setaspect()                 /* Function sets ratio that will allow */
{                                /* for displaying perfect circles      */
   float ratio, xasp, yasp;      /* without an elliptical effect.       */
   ratio = (((headerrec[23]) +
           (headerrec[24]*(2^8)) +
           (headerrec[25]*(2^16)) +
           (headerrec[26]*(2^24)))/10000000.0);
   xasp = ratio;
   yasp = 1.0;
   setaspectratio(xasp, yasp);
}
```

```
int getrecord()                    /* Function reads the file and evaluates   */
{                                  /* the numbers for their graphical display */
   int i;                          /* significance. Most of the numbers are   */
   datarec[1]=fgetc(fp);           /* composed of more than one byte. All     */
   datarec[0]=fgetc(fp);           /* numbers are within fixed ranges.        */
   switch (datarec[0])             /* For specific info on what the ranges are*/
   {                               /* refer to the AutoCAD Reference Manual.  */
      case offsetvector :          /*<-- this condition evaluates the x,y*/
         for (i = 2; i < 5; i++) /*offsets of the last vector and draws*/
         datarec[i]=fgetc(fp);     /*a new vector using fcn drawoffset().*/
         drawoffset();
         break;
      case endoffile :             /*<-- end file reading*/
         break;
      case solidfill :             /*<-- calls fcn that fills closed     */
      for (i = 2; i < 6; i++)      /*objects in a current drawing color. */
            datarec[i]=fgetc(fp);
         solidfill();
         break;
      case commomendpoint :        /*<-- calls fcn that draws vector from */
         datarec[2]=fgetc(fp);     /*last point used, which is common to  */
         drawcommon();             /*more than one vector.                */
         break;
      case newcolor :              /*<-- call to set a new drawing color. */
         newcolor();
         break;
      default :                    /*<-- call to draw a vector. */
         for (i = 2; i < 8; i++)
            datarec[i]=fgetc(fp);
         drawvector();
         break;
   }
   return datarec[0];
}

void drawvector()
{
 int i, temp[4];
 for (i = 0; i <= 1; i++, i++)
 {
    temp[0] = datarec[i]*256+datarec[i+1];
 }
 for (i = 2; i <= 7; i++, i++)
 {
    temp[i/2] = datarec[i+1]*256+datarec[i];
 }
 line (temp[0], MaxY-temp[1], temp[2], MaxY-temp[3]);
 curX = temp[0]; curY = temp[1];               /*TURBO C's 0,0 origin is in    */
}                                              /*upper left corner of the scrn.*/
                                               /*AutoCAD's 0,0 origin is in    */
                                               /*lower left corner of the scrn.*/
void drawoffset()                              /*MaxY-temp.. fixes this problem*/
{
int temp[4];
   temp[0] = curX + (char)datarec[1];
   temp[1] = curY + (char)datarec[2];
   temp[2] = curX + (char)datarec[3];
   temp[3] = curY + (char)datarec[4];
   line (temp[0], MaxY-temp[1], temp[2], MaxY-temp[3]);
   curX = temp[0]; curY = temp[1];
}

void newcolor()
{
   int acadcolors[] = {BLACK, RED, YELLOW, GREEN, CYAN, BLUE, MAGENTA,
                       WHITE, DARKGRAY, LIGHTBLUE, LIGHTRED, LIGHTMAGENTA,
                       LIGHTCYAN, LIGHTGREEN, LIGHTGRAY, BROWN};
   setcolor(curColor = acadcolors[datarec[1]]);
}

void drawcommon()
{
   int temp[4];
   temp[0] = curX + (char)datarec[1];
   temp[1] = curY + (char)datarec[2];
   temp[2] = curX;
   temp[3] = curY;
   line (temp[0], MaxY-temp[1], temp[2], MaxY-temp[3]);
   curX = temp[0]; curY = temp[1];
}
```

```
void solidfill()
{
   struct PTS outs[10];
   int ptnum, i, numberofpoints;
   setfillstyle( SOLID_FILL, curColor );
   numberofpoints = datarec[2];
   for (ptnum = 0; ptnum < numberofpoints; ptnum++)
   {
      for (i = 0; i < 6; i++)
         datarec[i]=fgetc(fp);
      outs[ptnum].x = datarec[3]*256+datarec[2];
      outs[ptnum].y = MaxY-(datarec[5]*256+datarec[4]);
   }
   fillpoly( numberofpoints, (int far *)outs );
   for (i = 0; i < 6; i++)
      datarec[i]=fgetc(fp);
}
```