INTERACTIVE COMPILATION USING GIMMS

Thomas C. Waugh
Department of Geography
University of Edinburgh
Edinburgh, Scotland, U.K.

GIMMS is a general purpose, user oriented geographic processing system which is extensively used for the production of medium- to high-quality thematic maps. It is a large integrated system and typically operates on a 'mainframe' computer such as the IBM 370 series and derivatives.

The system has a wide range of options allowing great flexibility to the user. For example, the *TEXT command has over 35 user settable parameters with at least another dozen accessible from within the text string to be drawn. With such a wide range of options (over 700 in the system), facilities to optimize their use must be provided.

One of the major problems in computer mapping is the design of the maps. With a line printer system it was not really essential to 'design' the maps since the output was generally so crude. However, with line plotter systems, the capability to produce high quality maps exists and the cartographic aspect is increased.

With batch oriented systems there is a significant delay in the return of a plot which can vary from a few minutes to a few days. Inevitably mistakes are made, such as text overlapping other text or map symbolism. Thus it will take several runs to complete a design. In practice maps are generally not designed or are badly designed, and therefore, methods are required to make it easy to design a map. Interactive graphics is the obvious answer.

GIMMS always could operate in batch mode or interactive mode, so theoretically there should be no problem in generating maps interactively and plotting only the final product on a hardcopy device. In practice it wasn't quite like that. Earlier versions of the system ran in what could be called pseudo-batch as far as drawing is concerned. Take, for example, the *TEXT command. If you draw a piece of text and it isn't quite right, then you should be able to change some of the options and try again. However, the text has already been drawn either on the graphic display or the plotter. A system could be developed that only drew the final version, but that would negate the idea of graphic interaction. The alternative is to provide a system that 'remembers' the final version of each command and then redraws all the commands at one time. This latter method may generate very large and complicated sets of commands for a complex map. It is also inefficient for batch processing where the facility is not required. In addition, it is not necessary and may be counterproductive (e.g. expensive) to involve all of a map in the design process. For example, in the basic design of the layout of a map it is not necessary to shade polygons, and that is often the most expensive part of producing a map.

A decision was therefore taken in 1976 to develop an interactive compilation system which became subsequently the *COMPILE module of the GIMMS system. This system is not intended to produce final maps, it is intended to produce map skeletons. A conscious decision was taken to exclude functions which were of limited use in the layout design process. In addition, it was felt that the production of final maps on an interactive graphic display was undesirable due to the low resolution of most devices and the cost of computing when running interactively.

The sub-system was to provide easy to use interactive graphic facilities to design maps using minimal computer resources, and to provide a map skeleton which would be used (and modified) to produce final maps.

Several problems were identified and given consideration in the design. First and foremost was the effect of running GIMMS on a multiuser mainframe. This has many consequences, the most serious of which are the CPU available and the line speed to the users device. Most mainframes which provide timesharing have the characteristic that the CPU time available to an individual is a function of the number of users and may seriously constrict the use of computer graphics which often require a great deal of computer power. Another serious constriction is the line speed available to the user on a large multi-user system. It rarely exceeds 1200 baud and often is as low as 300 baud. This effectively rules out rapid interaction using graphics unless intelligent terminals are being used.

An important consideration in the design is the requirement for graphic display independence since it is not known which devices would be attached at the various installations. Therefore, the graphic interface is via a very basic set of graphic primitives which any vector plotting device can draw and in some instances ignore (e.g. blank a portion of the screen).

The structure identified for the sub-system was a common one, namely that of creating graphic objects, modifying them, and redrawing them as requested. The most useful output of the system is the set of commands necessary to produce the graphic objects. This set of commands is a skeleton of the map being produced and forms the design basis of a set of commands to form a completed map. This can reduce the design cycle from days to minutes, thereby increasing the likelihood of higher cartographic quality.

The major problem encountered was the user interface. The GIMMS language at that time was relatively primitive being a positional parameter system. For example, the command

```
*TEXT 5 9.2 0.5 45 'TEST'
```

would print the text TEST at position 5,9.2 with size 0.5 cm and at 45 degrees. In fact, a size of 0.5 is the default size, but it had to be specified so that the angle (45) could be specified.

This language was not easy to use in an interactive mode since it often meant giving values that were not necessary, and it required a knowledge of all the values prior to one to be changed. A major lack was the capability to support graphic interaction. The system also had no 'help' capability.

A new language system was thus developed. Called the GPIS (General Parameter Input System) it is a keyword oriented system with extensive facilities to read parameters of various types. The basic type of parameters are INTEGER, REAL, LABEL, STRING and the KEYWORD itself. In addition, REAL values may be vectors of values and may be set by a graphic cursor.
The language supports both batch and interactive use as typified by the use of a graphic cursor. For example, the command

```
        *TEXT X=5,Y=9.2,ANGLE=45,TEXT='TEST'
```

could be a command in a batch run or typed in at the terminal. If it was desired to use the graphic cursor, then a colon(:) is inserted into the input stream. For example,

```
                      *TEXT X :
```

would cause the graphic cursor to be displayed. After positioning the cursor to the required position, the user indicates that the point is chosen (usually pressing a button or typing a character). The system then responds with a message of the form

```
        OPTION=X     =5.23
        OPTION=Y     =9.12
```

which indicates the position selected by the user. The user can modify this position by reselecting the cursor or by typing in a new value. For example,

```
        X=5.1
```

would produce the message

```
        OPTION=X     =5.10
```

and the new position chosen would be 5.1,9.12. Thus, the user can explicitly set values by typing them into the system or by selecting the graphic cursor. This allows a common system for batch and interactive work.

Any real value parameter may be set to any of 4 cursor modes. They are

- :P position indicated by single cursor position
- :D size (or distance) indicated by giving two cursor positions
- :A angle indicated by giving two cursor positions
- :M relative shift (or movement) indicated by two cursor positions

and this allows the cursor setting of several different types of parameter.

The system recognizes 3 types of terminator to a cursor action. Using the example of the Tektronix 4000 series storage displays they are terminating a cursor action by one of three characters, the 'space', the letter 'C', and the letter 'A'. The 'space' bar is the normal terminator and will set the appropriate parameter. The letter 'C' means reselect the graphic cursor for another point. For example, if a size parameter is to be set by the cursor, two points are required; therefore, the C code is used. For example, SIZE:

```
                 cursor appears,position,type C
                 cursor reappears, position, type space
```

system responds with

```
         OPTION=SIZE   =7.23
```

which is the distance between the two points.

In many cases, a command will have several parameters able to be set by the cursor, For example, giving the command

```
         *NORTHPT    ?
```

will trigger the 'help' information for the *NORTHPT command which will appear in this form:

```
         X          /R,:P/
         Y          /R,:P/
         SIZE       /R,:D/
         ANGLE      /R,:A/
         .
         .
```

which indicates that the parameters

- X and Y are of type real and may be set by cursor positioning
- SIZE is of type real and may be set by the cursor using two points
- ANGLE is of type real and may be set by the cursor using two points.

These parameters may be set explicitly (e.g. X=5.2,Y=7.1 SIZE=1.5,ANGLE=72) or may be set separately by the cursor as in the example above or may be set using the 'A' code. This terminator sets all the possible parameters by invoking the cursor once and pointing at two positions. For example, *NORTHPT :

- cursor appears, position at base of required north arrow, type C
- cursor reappears, position at top of north arrow, type A

system responds with

```
         OPTION=X         =5.14
         OPTION=Y         =7.21
         OPTION=SIZE      =1.43
         OPTION=ANGLE     =69.57
```

indicating that 'A'll the options have been set with the two cursor points.

The system is therefore very flexible in the manner which the user chooses to set parameters and does not require special hardware, only a capability to 'point' to a position and indicate one of three terminators.

The compilation system thus operates by selecting a command, such as *TEXT, giving the parameters, by typing or by cursor, drawing the text, modifying the parameters and redrawing if necessary, and when the required image has been created the option

KEEP

is specified, to which the system responds with a message of the form

OBJECT STORED AS OBJECT 5

where the number 5 means that it is the fifth object stored.

Graphic objects may be drawn (*REDRAW), moved (*MOVE), deleted (*DELETE), stored on a file (*STORE), restored from a file (*RESTORE), and listed (*COMMANDS) on the terminal or to a file. The last command (*COMMANDS) decodes the objects into the commands necessary to generate the objects and which can create a text file of these commands, thus forming an editable skeleton of the map design.

An important consideration of the design of the system was to ensure that it would not be tied to any one display device, or size of device. This is achieved by using a theoretical map space in the compilation system. For example, the command

*NEWMAP MAP SIZE=70,60

will set up a map size of 70 cm by 60 cm. There are few display screens on which this can be shown at full size; therefore, the system will scale the map to the screen available. However, all specification of parameters is in the map image space so that specifying a command of

*TEXT X=54.2,Y=43

would place the text at the correct place in the map image space. Furthermore, all cursor functions return values in the map image space.

Since the map space may be quite large, the ability to zoom into the map space is provided. The user selects a box and the area within the box is expanded to fit the screen. The command

*ZOOM OFF

will go back to the full map and

*ZOOM ON

will return to the last zoom area. Zooming within a zoomed area is allowed. Even within a zoom all parameters are specified in the map image space and are set as such by the cursor.

The system has certain useful characteristics. It does not consume large amounts of CPU and line time by constant redraw (unless requested) and requires only a simple graphic display (with or without a cursor). It has great flexibility in input and provides a 'help' facility to the interactive user. The same system can operate in a batch mode to produce final maps requiring more computer resources.

Although the system requires minimum resources to operate, it has built-in capabilities to make use of more sophisticated hardware and intelligent terminals. For example, if a raster terminal or refresh terminal is available then the system will use selective erasure to delete objects before retrying them with changed parameters. For storage screens this function is ignored. The ability to change pens will be translated on some terminals to be a change of line type or of colour. For example, on the Tektronix 4014 a change of line type will be effected and on the 4027 a change of colour.

In addition, to the use of specialised hardware, the system has the capability to utilise truly intelligent terminals. The @ character is used to pass control to a user specified subroutine which may insert or modify parameter values. For example,

*TEXT X=1,Y=5,SIZE=0.7,TEXT='TEST TEXT' @

would set up the appropriate parameters and then pass control to a user specified subroutine. An example of how this would be used is that the user routine would pass the parameters to an intelligent terminal which would interact with the user under local control and pass back modified parameters to GIMMS. This offloads some of the work to a local intelligent terminal.

Development is currently under way to provide an interface between GIMMS and the APPLE II microcomputer to provide these facilities. The software in the APPLE II will be written in PASCAL.

At this point a short videotape was shown.

It has transpired that the *COMPILE subsytem of GIMMS is a heavily used facility, not only to design maps but also to design diagrams of all sorts. It has met its design requirements of reducing the design bottleneck with software that uses minimal computer resources on a timesharing mainframe at relatively low line speeds. Perhaps the most important step in the development of the subsystem was the development of the GPIS user language which has proved to be an extremely powerful tool in its capability to support batch, interactive, and interactive graphics uses.