To: MTB Distribution

From: Paul W. Benjamin

Date: 06/15/81

Subject: Graphics in Compose

INTRODUCTION

It has been proposed that the Multics WORDPRO Text Formatter (compose) supply the user with a means by which permanent graphic segments created by the Multics Graphics System can be inserted in a document (compout segment). Although not an advantage on many device types, this feature would be quite useful on Diablo-type terminals, graphics terminals with hard copy devices, and certain photocomposition equipment. The purpose of this MTB is to determine how this might be accomplished and what resources would be required to do it.

DEVICE SUPPORT

The issue of the various devices supported by compose and/or MGS arises. One question is what to do about device types that compose supports for which there is no graphic definition table or graphic support procedure. These can certainly be created for the Diablo-type devices and for photocomposition, in fact, Jim Falksen has written a GDT and GSP for the DTC300S which exists on System-M but is not installed. The real question is what to do about other devices that compose supports such as ASCII or VIP7801? Clearly, the user must be able to compose his document on any terminal, whether or not that terminal supports graphics. There are 2 possible approaches. One is to create GDTs and GSPs for these as well, using the characters that are available in an attempt to create graphic output. The other approach is to simply display an outline around the area that it will occupy, utilizing perhaps vertical bars and underscores. I have seen attempts to implement the first approach and feel that the quality of the output is, depending on the graphics in question, in the best case barely satisfactory and oftentimes considerably worse. Considering that, and the manpower required to create such support, I would recommend that the latter approach be taken. It turns out that the Mergenthaler, the only currently supported photocomposition device, falls into the category of devices that do lend themselves well to this implementation, it too, would simply display an outline. In the future, however, photocomposition devices that do support graphics well will be supported by compose, and GDT/GSPs would need to be written for

them as well. Conversely, compose support for graphic devices such as the TEK4014 would be desirable. A beneficial byproduct of such support would be the ability to use such devices as 'photo-simulators' for photocomposition output.

A problem associated with the diablo-type terminals is the fact that such terminals either cannot back up (scroll up) or do not do it in an acceptable fashion. To circumvent this, the GSPs must be written in such a manner that the output progresses from top to bottom. Falksen's DTC300S support works in that fashion. However, there are problems associated with displaying graphic structures that are horizontally adjacent to text or other graphic structures. Compose would have to be able to intermix its output. Unfortunately, something of the nature of "shift n positions to the right, display a vector of m positions to the right, shift x positions down, shift y positions to the left, display a vector of z positions to the right, etc.", while optimal for display for the graphics system itself, does not lend itself well to the sort of intermixing that compose must do. What compose needs is a plotline by plotline approach, something like "for this plotline, starting from the left, shift n positions to the right, display a vector of m positions to the right, for the next plotline, starting from the left, shift x positions to the right, display a vector of y positions to the right, etc.". This sort of output can then be readily intermixed by compose with adjacent text. Since this sort of thing is sub-optimal for regular graphics applications, it is apparent that the GSPs and GDTs used by compose should be different from and separate from those currently used by MGS. They could then be contained in, or identified by, the device table for the given device. It is unclear as to whether or not the standard graphic support for traditional graphic devices such as the TEK4014 could be utilized, it appears that much of them could be copied or utilized directly but that certain portions thereof would need to be changed.

IMPLEMENTATION

To implement this, one new compose control would be necessary, .igr (insert graphic). The propose syntax would be:

.igr; <pathname> <structure> <width> <length> {<rotation>}

where pathname is the pathname of the permanent graphic segment, structure is the desired graphic structure within the pgs, width and length are the width and length in 10-pitch characters of the area in the document where the graphic is to be inserted, and rotation is the positive rotation in degrees around the z-axis (assumed to be zero if omitted). A new control argument, -nographics, which would inhibit the display of graphics, would also be useful in situations where the pgs was currently unavailable or the user was loath to spend the additional

cpu/clock time required to generate the graphics. An outline would be displayed, ala the default action for device types that did not support graphics. A similar control argument, -noart, currently exists to inhibit artwork.

Compose then, on encountering an insert_graphic control in the input file (in the absence of the -nographics control argument), would attempt to find the GDT/GSP for the user's device and either do the equivalent of the setup_graphics command (directing output to a temporary file) or determine that no graphic support was available. In the latter case, it would only be necessary to display an outline of the area that the user specified for the graphic. Otherwise, the real work begins. A new module, graphics_size_, would be called to determine the origin and the 'width' and 'length' of the graphic. A calculation would then be made to determine what scaling factor would be necessary to make the 'picture' fit in the 'box'. The appropriate calls are then made to various entrypoints in graphic_manipulator_ and graphic_compiler_, terminating in a call to graphic_compiler_$display. Compose then has something it can deal with, and the battle is won.

MANPOWER REQUIREMENTS

What follows is a list of the tasks necessary for implementation and an estimate of the man-time-units required to accomplish them:

1a) Write a generalized procedure to display the outline (box) to accomodate the -nographics control argument and for devices that do not lend themselves well to graphics. (2 man-weeks)

1b) Write GDT/GSP for DTC300S. This would be the pilot program and would accordingly take longer than other devices. (9 man-months)

1c) Write GDT/GSP for HYTERM. (6 man-months)

2a) Create Compose support for currently supported graphics devices. (6 to 12 man-months per device, assuming the availability of a terminal)

2b) Create Compose-specific GDT/GSPs for the terminals in 2a). (3 to 4 man-months per device, again assuming availability of terminals)

3) Write graphic_size_. This is substantially complete, the time represents debugging. (4 to 6 man-weeks)

4) Change Compose to support and implement .igr and -nographics. (6 man-months)