Pascal
#15: Apple II Pascal SHORTGRAPHICS Module

Revised by: Cheryl Ewy & Dan Strnad November 1988
Written by: Cheryl Ewy December 1983

This Technical Note describes the Apple II Pascal SHORTGRAPHICS routine, which is available as part of the 48K Run-Time System.

Introduction

Many applications, especially those designed to use the 48K Run-Time System, run out of memory quickly if they use the TURTLEGRAPHICS unit provided with the standard SYSTEM.LIBRARY.

This document describes a library unit called SHORTGRAPHICS which removes the relative polar coordinate features of TURTLEGRAPHICS to save memory.

General Comments

If your application uses (or can be modified to use) only those TURTLEGRAPHICS procedures which refer to absolute screen coordinates, you can use the SHORTGRAPHICS unit. The SHORTGRAPHICS unit has the same segment numbers assigned to it, as does TURTLEGRAPHICS, thus you may not use both in the same program.

Deletions

The following routines are not available in the SHORTGRAPHICS unit:

```
PROCEDURE TURN(ANGLE: INTEGER);
PROCEDURE TURNTO(ANGLE: INTEGER);
PROCEDURE MOVE(DIST: INTEGER);
FUNCTION  TURTLEANG: INTEGER;
```

Additions

The following definitions have been added to the INTERFACE section of SHORTGRAPHICS:

```
TYPE
       FONT=PACKED ARRAY[0..127,0..7] OF 0..255;
VAR
       FONTPTR:^FONT;
```

The variable FONTPTR is a pointer to the memory area used by the WCHAR and WSTRING procedures to display text on the graphics screen.

Thus, if you have a character set named KATAKANA.FONT, you could load it into memory and use it as follows:

```
VAR
       SPECIALFONT:^FONT;          (* where the new font goes *)
       SAVEFONT:^FONT;             (* to save pointer to standard font area *)

       PROCEDURE LOADFONT;
       VAR
              F:FILE;
              NIO:INTEGER;
       BEGIN
              NEW(SPECIALFONT);
              RESET(F,'KATAKANA.FONT');
              NIO:=BLOCKREAD(F,SPECIALFONT^,2,0);
              CLOSE(F)
       END;

       PROCEDURE USESPECIAL;
       BEGIN
              SAVEFONT:=FONTPTR;   (* save standard font pointer *)
              FONTPTR:=SPECIALFONT; (* and point to special font *)
       END;

       PROCEDURE USENORMAL;
       BEGIN
              FONTPTR:=SAVEFONT    (* restore pointer to normal font *)
       END;
```

Memory Considerations

When the system is booted, the heap pointer is normally below the start of high-resolution page one. The TURTLEGRAPHICS unit automatically sets the heap pointer above high-resolution page one. This protects the high-resolution page from being overwritten by your program, but it also prevents you from using the space between the original top of the heap and the start of high-resolution page one for your own variables.

SHORTGRAPHICS does not protect the high-resolution page, thus you may use this extra space for yourself. The following code will check to see if you have n bytes available between the top of the heap and high-resolution page one. If the room is not available, the heap pointer will be jumped to the top of the high-resolution page.

```
PROCEDURE MAKEROOM(N:INTEGER);
CONST
       BOTTOM=8192;

       TOP=16384;
VAR
       CHEAT:RECORD CASE BOOLEAN
                     TRUE:(IPART:INTEGER);
                     FALSE:(PPART:^INTEGER);
              END;
BEGIN
       MARK(CHEAT.PPART);
       IF (CHEAT.IPART+N)>=BOTTOM THEN BEGIN
              CHEAT.IPART:=TOP;
              RELEASE(CHEAT.PPART)
       END
END;
```

Thus, if you wanted to allocate a special font (which requires 1,024 bytes) below the high-resolution page, you could use this code:

```
MAKEROOM(1024);
NEW(SPECIALFONT);
```

If there are at least 1,024 bytes beneath the high-resolution page, the new font will be allocated there. If there is not enough space there, the new font will be allocated above the high-resolution page.

All of these heap allocations should be done as the very first actions of your program. When you finish allocating your variables, you should invoke the following procedure to make sure the heap pointer is above high-resolution page one (thus protecting it).

```
PROCEDURE PROTECT;
CONST
       TOP=16384;
VAR
       CHEAT:RECORD CASE BOOLEAN OF
                           TRUE:(IPART:INTEGER);
                           FALSE:(PPART:^INTEGER);
                     END;
BEGIN
       MARK(CHEAT.PPART);
       IF CHEAT.IPART<TOP THEN BEGIN
              CHEAT.IPART:=TOP;
              RELEASE(CHEAT.PPART);
       END;
END;
```

Warning: Every program written using SHORTGRAPHICS is unprotected from a heap which grows large enough to go into the high-resolution page one area. Therefore, every program using SHORTGRAPHICS should protect

page one by using PROCEDURE PROTECT. You should protect page one even if the program does not use the space below it.

Code Length

When you look at TURTLEGRAPHICS with the LIBRARY program, the code segment has a length of 5,230 bytes and the data segment a length of 386 bytes. SHORTGRAPHICS has a code segment 3,140 bytes in length and a data segment of 18 bytes. Thus, in a test of a null program, you have 2,458 bytes more space available.

Files on the Disk

The following files are on the SHORT GRAPHICS disk:

SHORT.CODE	Contains the SHORTGRAPHICS code. You can use it as a library or use the library program to add it to SYSTEM.LIBRARY in place of TURTLEGRAPHICS.
KATAKANA.FONT	A sample font used to demonstrate the use of alternate fonts.
SYSTEM.CHARSET	The letters in this character set are not as wide as those found in the SYSTEM.CHARSET supplied with the Development System and the Run-Time Systems.
TEST.TEXT, TEST.CODE	A sample program showing some of the concepts discussed in this Technical Note.

Interface Listing of the SHORTGRAPHICS Unit:

The following is the interface section of the SHORTGRAPHICS unit:

```
UNIT SHORTGRAPHICS; INTRINSIC CODE 20  DATA 21;

       INTERFACE
             TYPE

SCREENCOLOR=(none,white,black,reverse,radar,black1,green,violet,white1,
                              black2,orange,blue,white2);

                 FONT=PACKED ARRAY[0..127,0..7] OF 0..255;

             VAR
                 FONTPTR:^FONT;

             PROCEDURE INITTURTLE;
             PROCEDURE MOVETO(X,Y: INTEGER);
             PROCEDURE PENCOLOR(PENMODE: SCREENCOLOR);
             PROCEDURE TEXTMODE;
             PROCEDURE GRAFMODE;
             PROCEDURE FILLSCREEN(FILLCOLOR: SCREENCOLOR);
             PROCEDURE VIEWPORT(LEFT,RIGHT,BOTTOM,TOP: INTEGER);
             FUNCTION  TURTLEX: INTEGER;
             FUNCTION  TURTLEY: INTEGER;
```

```
FUNCTION  SCREENBIT(X,Y: INTEGER): BOOLEAN;
PROCEDURE DRAWBLOCK(VAR SOURCE; ROWSIZE,XSKIP,YSKIP,WIDTH,
                    HEIGHT,XSCREEN,YSCREEN,MODE:
                    INTEGER);
PROCEDURE WCHAR(CH: CHAR);
PROCEDURE WSTRING(S: STRING);
PROCEDURE CHARTYPE(MODE: INTEGER);
```