The PC tachistoscope has 240 pages

BRETT MYORS

Macquarie University, Sydney, New South Wales, Australia

Myors (1998) showed how to combine familiar video page switching with bit-plane layering in video mode 0dh to increase the capacity of the PC tachistoscope from 8 to 32 pages. The present article shows how to combine video page switching with color page switching to implement a 240-page tachistoscope, thus producing an almost eightfold increase in the capacity of the PC tachistoscope. The main limitation of this technique is that the images cannot all be located in the same position on the screen. Complete source code in C is included.

A tachistoscope is an instrument for rapidly displaying visual information and is standard equipment for researchers investigating perception and cognition. Classic mechanical tachistoscopes used half-silvered mirrors and could typically display from two to four images in quick succession. Modern computerized tachistoscopes are based on the idea of *preloading* multiple images into video memory and rapidly switching between them. Computerized tachistoscopes can present many more images than can mechanical ones but are limited by the refresh rate of video monitors. Standard monitors refresh the screen at from 60 to 70 Hz, so a computerized tachistoscope can switch images every 14.3 msec, at best. If this is fast enough, the advantage of the computerized tachistoscope is its greatly increased capacity.

It is well known that many PC video modes support up to eight video pages—that is, the video buffer is large enough to store up to eight separate screen images simultaneously. Segalowitz (1987) was one of the first authors to describe how video page switching could be used for tachistoscopic presentations in text mode, and Haussmann (1992) reviewed the standard video paging techniques for graphics modes. Myors (1998) further increased the capacity of the PC tachistoscope by combining video page switching with bit-plane layering in video mode 0dh to preload up to 32 separate screens. Mode 0dh supports eight video pages. Myors also showed how to increase the low resolution 40-column by 25-line text mode, mode 01h, to hold up to 16 screens. The purpose of this article is to show how the 16-page mode 01h tachistoscope can be increased even further to 240 pages by combining video page switching with color page switching on the VGA. This technique makes use of the 256 VGA DAC registers and, therefore, will not work on earlier hardware; however, the VGA is ubiquitous in the PC environment.

Before describing the technique, mention should be made of super VGA (SVGA) modes. It is unlikely that

SVGA modes will support tachistoscopic applications, because their high resolution and large memory demands require them to utilize bank switching. Given that a high-resolution SVGA image is larger than the video buffer, it must be loaded in several chunks via a single “bank” of video RAM. Bank switching is not conducive to preloading images and rapidly switching between them. Despite its comparatively low resolution, mode 01h is very good for displaying text and can also display simple ASCII drawings. Indeed, it is the low-resolution and low-memory demands of mode 01h that give it its great capacity.

COLOR PAGE SWITCHING

Mode 01h is a standard text mode that can be used to display text from a palette of 16 colors. VGA adaptors, however, support a 256-color palette. The remaining 240 colors need not go to waste in 16 color modes, since the VGA has the capacity for color paging. Color paging is analogous to video paging but uses color registers rather than video RAM. A palette of 16 displayable colors, from the total 256, can be selected via the Color Index Register of the VGA Attribute Controller. A change to this one location immediately selects a new set of 16 colors for the entire screen. Thus, the contents of the screen can be changed very rapidly without writing a single pixel.

The Listing shows how to combine video page switching with color page switching to yield a maximum of 240 preloaded images for tachistoscopic presentation with the C programming language. The program uses BIOS calls and standard Microsoft Graphics Library routines in non-speed-critical areas.

First, the video card is put into mode 01h via a call to `_setvideomode()`. Then the VGA palette RAM is set to a pass through state using BIOS video services (interrupt 10h, function 10h, subfunction 02h). This ensures that the EGA-compatible circuitry of the VGA becomes transparent. Next, interrupt 10h, function 10h, subfunction 13h is used to enable the 16 pages of 16 colors color-paging mode.

The demonstration display is set up so that text from each of the 15 displayable color pages is written on the

Correspondence concerning this article should be addressed to B. Myors, Department of Psychology, Macquarie University, Sydney, New South Wales 2109, Australia (e-mail: bmyors@bunyip.bhs.mq.edu.au).

same line of the screen. Each of the 16 screens within a color page is created via video paging. The function `print_string()` is used to write text to the video pages and to set the color attributes. The 256 color registers (DAC) are set up so that each color page has only one active, non-black color. Thus, although there are 15 lines written on the screen in each color page, only 1 line will be visible, because the remaining 14 are set to the background color (i.e., black). Dixon (1991) showed how changing the colors of pixels present on the screen could be used to change the display so that it looks like a completely new screen, although he concentrated on Macintosh computers and did not describe the color-paging technique reported here. The general approach of using colors rather than pixels to change the appearance of the display is the basis of *color-palette animation*.

Readers may be wondering why only 15 color pages are implemented when 16 are available. Why is it not possible to create 256 pages? The main reason is that 1 of the color pages will control the overscan (boarder) color of the screen and, if this color is not kept black, the overscan region will light up when that page is selected. Such flashing of the overscan area is likely to be distracting.

When the screens are set up, the DAC is loaded, and the user is able to tab through the 240 different screens. The down-arrow key moves forward, and the up-arrow key moves backward through the screens. To help navigation and show the operation of color paging, each screen provides three pieces of information: the tachistoscopic page number, between 1 and 240; the video page number, between 0 and 15; and the color page number, between 0 and 14. The Listing shows how the video pages and color pages are combined to give a tachistoscopic page. Pressing escape exits the program. The active colors set up in the Listing change smoothly from cyan to magenta, to emphasize a new color page, but this is not essential for the implementation of color paging. All active (non-black) DAC colors could be set to the *same* color, if desired.

Before switching pages with the `set_page()` function, the program calls the `sync()` function to lock on to the vertical retrace. Locking onto the vertical retrace ensures that screen changes only occur while the electron guns in the monitor are moving from the bottom right-hand corner of the screen to the top left-hand corner and is essential for tachistoscopic applications and smooth screen transitions. Ideally, the code for `sync()` would be incorporated into `set_page()`, to avoid an unnecessary function call, but it was kept separate so that `set_page()` could be accurately timed, as will be described next.

Prior to exiting, the program performs a test of page-switching performance. Times are measured by the standard `clock()` function, which utilizes the on-board time-of-day clock. Although `clock()` has a period of only 54.9 msec, 1,000,000 page switches are performed, to gain adequate accuracy. Results of the timing were periods of 3.46 μ sec for a single page switch on a Pentium 100-MHz computer using an STB Powergraph 64 video ac-

celerator card. This is an order of magnitude less than the retrace interval of 60 μ sec (Sargent & Shoemaker, 1995) and indicates that this computer is quite suitable for tachistoscopic operations, using the methods described here. On a slow 16-MHz 386 PC with a standard Western Digital VGA card, `set_page()` was found to take 49.7 μ sec to execute, just fast enough to support tachistoscopic applications.

Page switching is performed by the `set_page()` routine, which shows how to change the CRT Address Registers of the CRT Controller to implement fast video paging and how to change the Color Index Register of the Attribute Controller to implement fast color paging.

CONCLUSION

The present article shows how to load 240 separate images simultaneously into video memory and rapidly present them on request within a single video refresh. This permits tachistoscopic presentation of stimuli on the PC. The method is particularly suited to presenting large amounts of text or graphics tachistoscopically, as in studies of speed reading (Juola, Ward, & McNamara, 1982), and is based on combining the well-known technique of video page switching with less well-known color page switching in video mode 01h. Color paging can also be implemented in other video modes, such as the low-resolution 0dh graphics mode (which supports 8 video pages) to implement a 120-page tachistoscope.

An alternative approach to achieving multiple color pages could be to repeatedly reprogram the first 16 DAC registers. This could effectively provide an infinite number of color pages and vastly increase the capacity of the PC tachistoscope. However, color paging via the single Color Index Register is easier and much faster than repeatedly reloading the DAC, and speed is critical in tachistoscopic applications.

The main drawback of color paging is that not all of the screens can overlap. Indeed, overlapping images are only possible via conventional video paging. Color page switching relies on having multiple images present on the screen, with only one image set to a nonbackground color. Images set to the background color will be invisible and, hence, effectively absent from the display. If users require a large number of images occupying the same place on the screen, they are advised to consider Myors's (1998) 32-page tachistoscope. It uses the VGA color circuitry to implement bit-plane layering rather than color paging and can hold up to 32 simultaneous images, all in the one location. Nevertheless, if even more images are required and they do not have to be in the same place on the screen, the techniques presented here may be of use.

AVAILABILITY

Copies of the program can be obtained by e-mail to the author (bmyors@bunyip.bhs.mq.edu.au).

REFERENCES

- DIXON, P. (1991). Tachistoscopic display using color-lookup-table display devices. *Behavior Research Methods, Instruments, & Computers*, **23**, 380-386.
- HAUSSMANN, R. E. (1992). Tachistoscopic presentation and millisecond timing on the IBM PC/XT/AT and PS/2: A Turbo Pascal unit to provide general-purpose routines for CGA, Hercules, EGA, and VGA monitors. *Behavior Research Methods, Instruments, & Computers*, **24**, 303-310.
- JUOLA, J. F., WARD, N. J., & McNAMARA, T. (1982). Visual search and reading of rapid serial presentations of letter strings, words and text. *Journal of Experimental Psychology: General*, **111**, 208-227.
- MYORS, B. (1998). The PC tachistoscope has 32 pages. *Behavior Research Methods, Instruments, & Computers*, **30**, 457-461.
- SARGENT, M., & SHOEMAKER, R. L. (1995). *The personal computer from the inside out* (3rd ed.). New York: Addison-Wesley.
- SEGALOWITZ, S. J. (1987). IBM PC tachistoscope: Text stimuli. *Behavior Research Methods, Instruments, & Computers*, **19**, 383-388.

LISTING

Demonstration of the 240-Page Mode 01h Tachistoscope

```
#include <stdio.h>
#include <conio.h>
#include <dos.h>
#include <time.h>
#include <graph.h>

#define ESCAPE          27
#define UP              72
#define DOWN           80
#define PAGES          240
#define PAGESIZE       2048

// VGA control
#define INPUT_STATUS_REGISTER 0x3da
#define VSYNC_MASK          0x08
#define CRT_CONTROLLER       0x3d4
#define CRT_ADDRESS_HI      0x0c
#define CRT_ADDRESS_LO      0x0d
#define INPUT_STATUS_PORT_1  0x3da
#define ATTRIBUTE_CONTROLLER_INDEX 0x3c0
#define ATTRIBUTE_CONTROLLER_COLOR 0x34
#define ATTRIBUTE_CONTROLLER_DATA 0x3c0

// PROTOTYPES
void print_string (int row, int col, char *str, char _far *vb, int color);
void sync (void);
void set_page (int page);

char DAC[256][3],
    paletteRAM[] = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 0},
    _far *video = (char _far *) 0xb8000000l; // Video buffer

void main (void)
{
    char buf[42];
    int i, page, ch, shade;
    union REGS inregs, outregs;
    struct SREGS sregs;
    long l;
    clock_t tm;
    double dtm;

    if (!_setvideomode(_TEXTC40))
        ;
}
```

LISTING (Continued)

```

printf("Video mode not supported.\n");
return;
}
_displaycursor(_GCURSOROFF);

// Set palette RAM to pass through state.
inregs.h.ah = 0x10;
inregs.h.al = 0x02;
inregs.x.dx = (unsigned int) paletteRAM;
segread(&sregs);
sregs.es = sregs.ds;
int86x(0x10, &inregs, &outregs, &sregs);

// Enable 16 pages of 16 colors mode.
inregs.h.ah = 0x10;
inregs.h.al = 0x13;
inregs.h.bh = 0x01;
inregs.h.bl = 0x00;
int86(0x10, &inregs, &outregs);

// Set up tachistoscopic display
for (i = 0; i < 15; ++i)
{
    shade = (i + 1) * 4;
    DAC[i * 16 + i + 1][0] = shade;
    DAC[i * 16 + i + 1][1] = 64 - shade;
    DAC[i * 16 + i + 1][2] = 63;
    for (page = 0; page < 16; ++page)
    {
        sprintf(buf, "PC Tachistoscope: (%3d, %2d, %2d)",
            i * 16 + page + 1, page, i);
        print_string(i + 4, 4, buf, video + page * PAGESIZE, i + 1);
    }
}

// Set DAC
inregs.h.ah = 0x10;
inregs.h.al = 0x12;
inregs.x.bx = 0x00;
inregs.x.cx = 0xff;
inregs.x.dx = (unsigned int) DAC;
segread(&sregs);
sregs.es = sregs.ds;
int86x(0x10, &inregs, &outregs, &sregs);

// User control
set_page(page = 0);
while ((ch = getch()) != ESCAPE) {
    switch (ch) {
        case UP : page = (page - 1 + PAGES) % PAGES; break;
        case DOWN : page = (page + 1) % PAGES; break;
        default : continue;
    }
    sync();
    set_page(page);
}

```

LISTING (Continued)

```

    }
    // Rapid switching
    tm = clock();
    for (l = 0; l < 1000000; ++l)
        set_page(l);
    tm = clock() - tm;
    dtm = (((double) tm) / 1000000.0) / ((double) CLOCKS_PER_SEC);

    _setvideomode(_TEXT80);

    printf("One million page switches occurred in %.1f seconds.\n"
        "One page switch occurred in %lf milliseconds (%.1lfHz).\n"
        "One vertical retrace takes 0.06 milliseconds.\n",
        ((float) tm) / ((float) CLOCKS_PER_SEC),
        dtm * 1000.0, 1.0 / dtm);
    }

void print_string (int row, int col, char *str, char _far *vb, int color)
{
    vb += (row * 40 + col) * 2;
    while (*str)
    {
        *vb++ = *str++;
        *vb++ = color;
    }
}

void sync (void)
{
    // Wait for vertical retrace
    while ( inp(INPUT_STATUS_REGISTER) & VSYNC_MASK);
    while (!(inp(INPUT_STATUS_REGISTER) & VSYNC_MASK));
}

void set_page (int page)
{
    int start_address;

    start_address = (page & 0x0f) << 10;
    start_address += (int) video;

    _disable();
    // Set video page
    outpw(CRT_CONTROLLER, (start_address << 8) | CRT_ADDRESS_LO);
    outpw(CRT_CONTROLLER, start_address | CRT_ADDRESS_HI);
    // Set color page
    inp(INPUT_STATUS_PORT_1);
    outp(ATTRIBUTE_CONTROLLER_INDEX, ATTRIBUTE_CONTROLLER_COLOR);
    outp(ATTRIBUTE_CONTROLLER_DATA, page >> 4);
    _enable();
}

```