— COMPUTER TECHNOLOGY —

Tachistoscopic software for Macintosh computers

THOMAS A. BUSEY

University of Washington, Seattle, Washington

The requirements for accurate presentation of short-duration stimuli on raster display monitors are summarized, and existing packages for instantiating such displays are evaluated. A new software package that provides tachistoscopic functions for Macintosh computers is described. This software, called MacTach, performs video-buffer switching, allowing tachistoscopic presentation of large or multibit depth images that is not available with alternative methods. Compatibility routines and verification procedures allow compatibility with all Macintosh computers. Applications for animation, motion, and dichoptic presentations are discussed.

With appropriate software, personal computers can provide rapid stimulus presentation that extends the capabilities of a standard tachistoscope. Done improperly, however, such presentation can produce unpredictable results. Short-duration images (16–70 msec) may appear darker in some areas, may occasionally flicker, or may not even appear at all. The solution to all of these problems can be achieved by the use of screen timing and presentation routines that precisely regulate the display and removal of images. For brevity, I will refer to the use of such routines as *screen control*, and I will define the success of screen control as the placement of an image at a precise and predefined moment.

Raster computer displays scan down a special area of memory called a *video buffer* and translate this information into the visible image with the scanning electron beam, a process called *refreshing the screen*. Screen-control success depends on software functions that first indicate when the video monitor has begun to refresh itself, and then transfer the contents of some hidden image to the video buffer for subsequent display. This transfer must occur ahead of the advancing scanning electron beam, and it must be completed in less than the time taken by the video card to refresh the screen (known as the *refresh rate*). Failure to transfer quickly enough results in partially drawn images during the first refresh, which causes the unpredictable results described above.

Traditional drawing techniques change the visual image by altering the contents of the video-buffer memory store, which results in visible changes to the screen. The time required for this change depends on the size of the image. Larger images take more time to be copied to the video buffer. If the image is too large, the transfer time exceeds the refresh rate, resulting in screen-control failure. To avoid this problem, the present package makes use of a hardware feature known as *video-buffer switching*. Instead of changing the contents of the video buffer, video-buffer switching involves switching to an entirely different video buffer. The software alters a memory location, activating a different video buffer containing a previously defined image. Since this procedure avoids the copying of memory from one buffer to another, the new image appears on the next screen refresh and the entire process takes less than 1 msec. The extra time granted by the use of video-buffer switching can be used to prepare the next stimulus for display.

These video-buffer switches must be synchronized to the monitor's refresh cycle to avoid screen-control failure. The present routines accomplish this with timing routines that are triggered by the start of the monitor's vertical retrace interval. During this interval, the scanning electron beam responsible for tracing the image onto the back of the phosphor screen returns to the top of the screen and prepares to refresh the fading image. Switching to a new video buffer during this interval ensures that the new image appears completely during the subsequent refresh. The timing routines wait for this interval to begin and then allow execution of video-buffer switching routines, resulting in a clean image exchange.

Existing Software

The concept of video-buffer switching is not recent. Yet although the history of video-buffer switching can be traced back at least to the Apple II computer (see Cavanagh & Anstis, 1980), most recent papers have addressed IBM

The author would like to thank Geoffrey Loftus, John Palmer, and Elizabeth Busey for their comments on previous drafts of this article and help in development of this package. Special thanks to Cynthia Ames for her help with timing routines, as well as to David Temkin and Juri Munkki for help with the video-buffer switching technique and the Alt Screen INIT. The comments of two anonymous reviewers were invaluable as well. This work was supported in part by an NIMH grant to Geoffrey R. Loftus. MacTach is available from the author. Write to T. A. Busey, University of Washington, NI-25, Seattle, WA 98195; please include a Macintosh disk. Alternatively, the author will e-mail a binhex/stuffit version of MacTach to electronic mail users (busey@u.washington.edu).

and compatible computers. Finley (1989) describes a similar technique of video-buffer switching on the Hercules display controller, for display presentations of text. Heathcote (1988) solves the problem of slow drawing by turning off the video signal until the stimulus is completely drawn—an option not available to Macintosh users. Watson et al. (1986) have written a general discussion of video display buffers, in which they describe not only how a video frame buffer operates, but also techniques of video-buffer switching, luminance calibration, and gamma correction. Since the Macintosh and other computers involve the use of similar technology, this is an excellent reference.

For the Macintosh computer, Rensink (1990) has prepared an excellent description of timing and display routines available through the Macintosh Toolbox. However, all of his fast display routines copy memory in some form or another. Since this copying takes much of the time available between refresh intervals, complex displays must be prepared in advance. Because this technique requires a great deal of memory for each stimulus, the number of stimuli is limited by the amount of memory available on the host Macintosh and can be awkward to program. In addition, color stimuli require more memory, and large color areas cannot be transferred in less than one screen refresh, causing screen-control failure. This is a serious problem, since in short-duration experiments contrast is often varied as an independent variable and color capabilities are required for the alteration of luminance.

Drawing directly to the screen by using QuickDraw calls is the simplest alternative to the fast-display routines. This technique suffers from two problems, however. First, because drawing commands are sent to the active screen buffer, drawing must be completed before the electron beam scan reaches the active drawing area. This decreases the amount of time available for drawing considerably. The top-left corner of the screen has less than 2 msec from the start of the vertical retrace interval before the electron beam scans it; any changes must be made in less than that time. Subsequent changes must also occur ahead of the electron beam, or the image will remain invisible during the first refresh, resulting in screen-control failure. The second problem is one of verification. There are no software-based timing techniques that indicate whether drawing has taken place ahead of the advancing electron beam. Even if the entire drawing process has taken less than one screen refresh, it is still possible that some or all of the drawing has taken place after the electron beam scan has already updated that section of the screen.

An alternative to these techniques is to use the monitor's color lookup table (CLUT) to hide images until they are fully drawn. Images are placed directly on the screen while the foreground and background colors are kept the same, hiding the drawing process. At the appropriate moment, two CLUTs are exchanged, changing the foreground to a different color and making the entire stimulus visible on the next refresh. Dixon (1991) provides an excellent description of this solution, as well as of the problem of overlapping displays. The intersection of two drawn but still hidden images must be treated separately,

a problem that becomes compounded with the use of more images with overlapping sections. Dixon also points out the limitations of this technique when it is used with many images of multiple depths, such as scanned pictures. In addition, this technique is not available on Macintosh models that lack Color QuickDraw.

The MacTach Software

To overcome the limitations of these methods, the video-buffer switching techniques are included in a library of procedures, called MacTach, that are designed to run visual display experiments. A display sequence done with video-buffer switching works as follows. The software displays a blank screen while the first stimulus is drawn to the hidden video buffer. Then, at the moment determined by the experimental design, the two video buffers are switched, displaying the first image. During the display duration of the first image, the second image is prepared on the hidden video buffer. The two video buffers are once again switched at the end of the first stimulus's display interval. This process continues until all stimuli have been displayed.

This technique is limited by drawing speed. Because only two video buffers are supported, experimental stimuli must be drawn "on the fly."¹ The software must completely draw the second image during the display interval of the first image, which becomes increasingly difficult with shorter durations. To identify when this problem occurs, the present routines include verification procedures that time all drawing steps and report an error when the drawing time exceeds the duration of the previous image. In addition to assisting the programming process, the verification techniques allow any computer to run the same experiment regardless of processing power, with the knowledge that MacTach will identify and report any problems.

Other routines supplied by MacTach adapt the millisecond timing routines described by Rensink (1990) to time events and record reaction times.² Miscellaneous routines determine the characteristics of the host computer, allocate the second video buffer if it is available, hide the menu bar, and restore graphical order after a series of trials. A sample experiment is included with MacTach; it provides a number of auxiliary routines to handle all aspects of setting up and running an experiment.

System Requirements

The MacTach routines are designed to run on the entire line of Macintosh computers. The software uses the video-buffer switching technique on the Macintosh Plus, SE, SE/30, and Classic.³ A similar technique is used on the Macintosh II and Macintosh IIfx in 16-color mode or less.⁴ Also, because the second video buffer is a property of the video card and not the computer, users of the Macintosh IIfx and Macintosh IIfx need only add a Macintosh II video card and monitor to their present systems to gain the fast video-buffer switching technique. When video-buffer switching is not available, MacTach simulates this technique by using off-screen bitmaps and standard Copybits routines for image transfer, which may or

may not transfer quickly enough. The speed of transfer depends on image size and depth, and success is verified by the timing procedures.

The MacTach routines are compatible with System 7 and Multifinder, and the video-buffer switching can be disabled by the user. It will support multiple monitors, and it takes advantage of Apple Toolbox Pascal routines to maintain compatibility with future hardware releases. MacTach is written in Symantec's Think Pascal, Version 4.0.

Applications

Video-buffer switching has advantages over other techniques for three general categories of presentation. The first category consists of short-duration displays of multi-bit images, such as pictures scanned by the Apple Scanner. It takes several refreshes to draw these images under normal conditions, but MacTach can draw and then switch the images quickly in order to maintain screen control. Large, colored displays such as visual search tasks and complex pattern masks also fall into this category.

The second category consists of 1-bit (black-and-white) displays intended to run on any Macintosh computer. While the CLUT switching technique is only available on Macintosh models with color QuickDraw, MacTach's initialization routines will suggest video-buffer switching if it is available, and if not, they will emulate video-buffer switching with standard drawing procedures. These alternative drawing techniques and initialization routines make MacTach compatible with all Macintosh models. Verification procedures also indicate when a computer's processing speed is not adequate for the demands of an experiment.

Finally, video-buffer switching is advantageous for running rapidly changing experiments, such as those involving animation and motion displays. These experiments require separate computations for each frame of the display to be defined, and many displays cannot be prepared ahead of time, because of limitations on memory and programming. Doing computations "on the fly" is an alternative. However, the computer typically does not have enough time before the start of the next refresh both to complete the computations for the next frame of animation and to copy the resulting image to the video buffer by using standard fast-display techniques. Using video-buffer switching frees up almost all of the time between refreshes for image computations, allowing successful "on the fly" presentation sequences.

A derivative of this third category involves the use of shuttered LCD glasses as described by Milgram (1987). Such glasses make one lense transparent while the other is opaque. Through alternating between eyes while simultaneously alternating between two images on the computer screen, dichoptic presentations, stereoscopic displays, and simulated depth perception are all possible. Since the two images must

be exchanged at a rate of 60-70 switches per second, screen switching techniques provide the most promise for presentation of large or color stimuli in this context.

Example Program

The listing in the Appendix demonstrates the use of the MacTach routines to present a series of images. Two multibit pictures are displayed for short durations and a zero interstimulus interval. These pictures appear within one refresh when MacTach uses video-buffer switching. The images are created by a scanner or a drawing program and stored in the program's resource fork. The experimental design indicates which picture identification numbers are used on each trial.

REFERENCES

- CAVANAGH, P., & ANSTIS, S. M. (1980). Visual psychophysics on the APPLE II: Getting started. *Behavior Research Methods, Instruments, & Computers*, **12**, 614-626.
- DIXON, P. (1991). Tachistoscopic display using color-look-up table display devices. *Behavior Research Methods, Instruments, & Computers*, **23**, 380-386.
- FINLEY, G. P. (1989). Tachistoscopic software for the Hercules display controller. *Behavior Research Methods, Instruments, & Computers*, **21**, 387-390.
- HEATHCOTE, A. (1988). Screen control and timing routines for the IBM microcomputer family using a high-level language. *Behavior Research Methods, Instruments, & Computers*, **20**, 289-297.
- MILGRAM, P. (1987). A spectacle-mounted liquid-crystal tachistoscope. *Behavior Research Methods, Instruments, & Computers*, **19**, 449-456.
- RENSINK, R. A. (1990). Toolbox-based routines for Macintosh timing and display. *Behavior Research Methods, Instruments, & Computers*, **22**, 105-117.
- WATSON, A. B., NIELSEN, K. R. K., POIRSON, A., FITZHUGH, A., BILSON, A., NGUYEN, K., & AHUMADA, A. J., JR. (1986). Use of a raster framebuffer in vision research. *Behavior Research Methods, Instruments, & Computers*, **18**, 587-594.

NOTES

1. The number of video buffers available depends on the video card and the current bit depth on Macintosh II computers. Some video cards support up to five buffers in the 1-bit-per-pixel mode. For the sake of simplicity, the MacTach routines support only two video buffers—the number available on small-screen Macintosh models and Macintosh II models in 16-color mode.
2. Because of hardware limitations, Macintosh models with Apple Desktop Bus connections to the mouse and keyboard are only accurate to the nearest 1/60 sec, which is not quite millisecond accuracy.
3. In order for the Macintosh Plus, SE, or Classic computers to use video-buffer switching, the user must first install the Alt Screen INIT (which is supplied with MacTach) in order to reserve the second video buffer.
4. The Macintosh IIx video card supports video-buffer switching and even provides more than one video buffer in the 256-color mode. This card appears to work differently than other video cards, however, and currently MacTach does not support video-buffer switching on the IIx. There is some hope that the routines may be adapted, but currently the author does not have the resources to run compatibility tests.

APPENDIX

Listing

```

procedure DoASampleSession;                                {called from the menu bar}
const
  TotalTrials = 72;                                         { Total number of trials in a session }
  PreTrialDuration = 40;                                   { Pre-trial duration, about 600 msec }
  PostTrialDuration = 10;                                  { Post-trial duration, about 150 to 167 msec }
type
  DataCard = record { Holds all subject information. Most filled when creating a new subject }
    initials, name, phone, age: str255;                   { Subject info }
    subnum, session: integer;                             { Subject info }
    H1Base, H1Inc, ISIBase, ISIInc: integer;              { Experimental parameters- vary duration, ISI }
    Contrast: real;                                        { Subject contrast level }
    dates: array[Nsession] of string;                    { Dates of each session }
    conditions: condorder;                               { conditions for the experiment }
    correct: condorder;                                   { correct responses }
    resps: condorder;                                    { Subject responses }
  end;

var
  FirstPictureDuration, SecondPictureDuration: integer;    { durations for the two pictures }
  FirstID, SecondID: integer;                             { ResEdit resource numbers for the two pictures }
  trial: integer;                                          { Current trial number }
  ErrorFile: Text;                                         { Holds all timing error messages }
  data: datacard;                                          { Holds all subject information }

procedure GetVariables (trial: integer; var FirstID, SecondID, FirstPictureDuration,
  SecondPictureDuration: integer);
var
  cond: integer; { current condition number }
begin
  cond := data.conditions[trial, data.session];
  { Use the condition to choose the durations and pict id numbers for a trial. For sample, just assign. }
  FirstID := 1001;                                         { Resedit resource number for the current first picture }
  SecondID := 1002;                                       { Resedit number for the current second picture }
  FirstPictureDuration := 8;                               { This is 8 screen refreshes, about 120 to 133 msec }
  SecondPictureDuration := 2;                             { This is 2 screen refreshes, about 30 to 34 msec }
end; { GetVariables }

procedure GetResponse;                                     { get the subjects response }
var
  ResponseTime, TempLong: longint;
  TempString: string;
  TempLong: longint;
  TheChar: Char;                                           { Returned by RT }
begin
  ResponseTime := RT(TheChar);                             { Reaction time- not used in this experiment }
  TempString := TheChar;
  StringToNum(TempString, TempLong);
  data.Resps[trial, data.session] := TempLong;            { save the subjects response }
end; { getResponse }

```

APPENDIX (Continued)

```

procedure DoTrial (trial: integer; var FirstID, SecondID, FirstPictureDuration,
  SecondPictureDuration: integer);
var
  FirstPicture, SecondPicture: pichandle;      { Handles to our two pictures }
  success: boolean;                             { Did we draw fast enough? }
procedure FinishDisplay (Duration: integer; errmsg: str255);
begin
  Success := FinishTimingTicks(Duration);      { Wait for Duration screen refreshes }
  if not success then writeln(ErrorFile, ErrorMessage); { Report Errors }
end; {FinishDisplay}
begin {DoTrial}
  StartTimingTicks;                            { Keep a record of drawing time }
  FirstPicture := GetPicture(FirstID);          { Get the first stimulus }
  DrawOnAltScreen;                             { Send drawing to hidden buffer }
  DrawPicture(FirstPicture, FirstPicture^.picFrame); { Draw first stimulus }
  FinishDisplay(PreTrialDuration, 'First Picture took too long to draw'); { Wait until time to display }
  StartTimingTicks;                            { Keep a record of drawing time }
  DisplayAltScreen;                            { Display first picture }
  SecondPicture := GetPicture(SecondID);        { Get the second stimulus }
  DrawOnMainScreen;                           { Send drawing to hidden buffer }
  DrawPicture(SecondPicture, SecondPicture^.picFrame); { Draw second stimulus }
  FinishDisplay(FirstPictureDuration, 'Second Picture took too long to draw'); { Wait until time to display }
  StartTimingTicks;                            { Keep a record of drawing time }
  DisplayMainScreen;                          { Display second picture }
  EraseAltScreen;                             { Erase first picture }
  FinishDisplay(SecondPictureDuration, 'Second Picture took too long to display'); { Wait second dur. }
  StartTimingTicks;                            { Keep a record of drawing time }
  DisplayAltScreen;                            { Display blank screen }
  FinishDisplay(PostTrialDuration, 'Blank screen took too long to display'); { Display post trial screen }
  FillMainScreen(White);                      { Erase second picture }
  DisplayMainScreen;                          { End trial with correct screen on display }
end; {DoTrial}

begin {DoASampleSession}
  rewrite(ErrorFile, 'Timing Error Messages'); { Open the timing error file }
  OpenSubject;                                { Get the subject's data file }
  trial := 1;
  PrepareForDrawing;                          { Erase the menu bar }
  repeat
    { Get the display variables and picture id's for current trial }
    GetVariables(trial, FirstID, SecondID, FirstPictureDuration, SecondPictureDuration);
    { Display the two pictures }
    DoTrial(trial, FirstID, SecondID, FirstPictureDuration, SecondPictureDuration);
    GetResponse;                              { Get the subjects response }
    trial := trial + 1;                        { Go to next trial }
  until Trial > TotalTrials;                   { Repeat until all trials have been run }
  data.session := data.session + 1;           { Run subsequent session next time }
  RecoverFromDrawing;                        { Redraw the menu bar }
  SaveSubject;                               { Save the subject's data file }
  Close(ErrorFile);                          { Save the error file }
end; {DoASampleSession}

```