PHILIPS

PHILIPS INTERACTIVE MEDIA of America

Technical Note #78

A Technique for Menu Highlighting

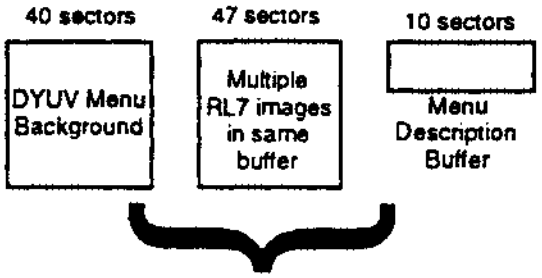
Blake Senftner, Sr. Engineer
PIMA Art Space

July 21, 1992

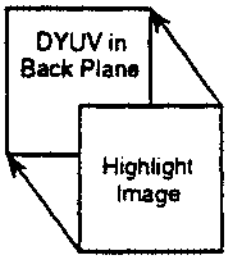
This note reviews a simple method for menu highlighting that is also very flexible. It involves use of a DYUV background image with a series of run-length 7 images. It can be used to highlight standard menu hotspots and can be adapted for use with slide bars and other graphical controls.

A Technique for Menu Highlighting

ASLAN, the PIMA ArtSpace software, uses an extremely simple method for highlighting. Nonetheless, this method is surprisingly flexible. The method involves a DYUV background image with a series of run-length 7 images (RL7). The RL7 images are displayed over the DYUV background. Color key transparency is used to allow the run-length images to be full screen in size; yet they contain the imagery for only a single hotspot's highlight.



Single RTR



RL7 in Front Plane
Color Key Transparency

The data required to load an ASLAN menu consists of:

- The DYUV background
- A buffer that contains the RL7 images (packed together) for the highlights
- A buffer that contains control information

This data is played into memory as a very short, generic ASLAN slide show. Once in memory, the slide show terminates and the menu software takes over. It takes 97 sectors of time to load such a menu after a seek to the location of the data. Much of this time period is masked to the user, because the dissolve to the menu background typical lasts three quarters of the time it takes for loading. Also, the ASLAN preprocessor locates the menu data near the slide shows that access it, thus keeping seek time to a minimum.

The menu control information contains a series of rectangle coordinates. When the cursor is inside the *nth* rectangle, the *nth* run-length highlight image is displayed. This is a very simple technique that responds to user interaction instantly. All that is required to change a highlight state is a scan-synchronized CLUT write and a single DADR instruction in the LCT.

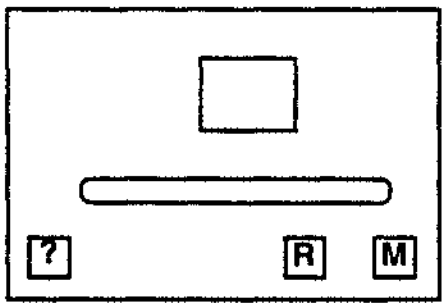
One of the reasons this technique was chosen was to allow the instant display of a new highlight image. It also overcomes some of the limitations of blitting into the menu background. Some of the problems it over comes follow below:

- Because a blit takes time, the user can see medium or large blits taking place
- The memory required to double buffer the display (to hide the blits taking place) and storage
  for the blits is much larger than that required for the run-length technique.

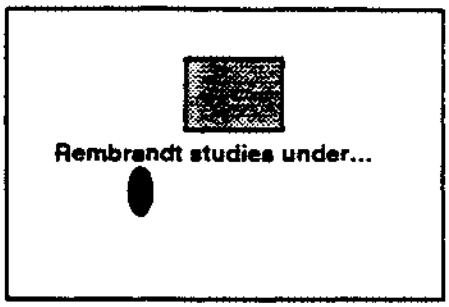
- Because it takes time to perform the blits, the user can "get ahead" of the software's ability to track the correct highlight state

When the user gets ahead of the software's tracking, the cursor and highlight state can be out of synchronization. The run-length/DYUV method is sufficiently fast that the display does not have to be double buffered. It also "unblits" the previous highlight state when changing to a new highlight image.

In addition, notice that because the run length images are full screen, the images displayed do not have to be contained within the hotspot rectangle's coordinates. In the *Dutch Masters* disc, ArtSpace introduced a menu that contains a slide bar control. The slide bar is actually a series of hotspots placed along the slide bar, and the run-length images contain both the slide bar knob and the result, or feedback, of the sliding knob being placed in that location. This simple twist in the use of the menu software required no change in the menu software.



DYUV Menu: Empty Slide Bar Control



RL7 Highlight: Slide Bar Feedback

It is very simple to imagine a rotating know, or a bouncing needle type of graphic control. The only drawback is that when the user exits the series of hotspots that operate a graphic control, the control does not retain its last highlighted state. Using this technique, only one button or control may be highlighted at a time. From the perspective of keeping the cursor and highlight state synchronous, this is good. But, from the perspective of wanting a graphic control to retain its final highlighted state, this is bad.

Soon, we will use a routine that writes a run-length image into a DYUV image. This will allow a graphic control to retain its highlighted state while the user operates another graphic control. The routine needs only a single line buffer to avoid streaking during conversion, and, therefore, still avoids the double-buffering overhead. The blit need only take place when the user has finished operating a graphic control; just changing the run-length image would be sufficient while the user operates the control. Only when the user exits the entire set of hotspots for the graphic control would the run-length to DYUV blit need to take place.

If you try this technique, I'd be interested in any suggestions for improvement.