S U R R O U N D

Although I prefer a subroutine structure, Surround was written in a straightforward manner. This was done for two reasons. First, the program isn't very long (only two pages), and it can be fairly easily comprehended. Second, there was a concern for timing problems. Several things happen between each "bip" of the trail, and subroutine calls (and returns) would only add a jerky movement to the snake. (This was the case with an earlier version, which was written with many subroutines.)

As a hint for your programming effort, sometimes it helps to write an entire program out of subroutine modules, which are then ordered by an executive program which calls each section in the proper sequence. The program can be left as it is, or, after debugging is complete, it can be tightened up. In fact, your CHIP-8 Interpreter is structured in this manner, with your program instructions providing the order of each function in the interpreter.

In Surround, memory locations 0200-0208 wait for either a Key 1 or Key 2 press to start the game. Key 1 activates the next section, which draws a border, while Key 2 sends the program directly to 022A. The border is drawn one bit short on two sides so the trail may run into it. Although this section could have been programmed more efficiently, I wanted the graphic effect of two lines drawing "at once" - admittedly a small point.

The next step is a machine language subroutine which copies the display page - with its border or without - into a separate memory page designated "0Z" and located two pages below the highest on-card RAM page. Like the Wizard of Oz, display page "0Z" is a faker, and is used solely to display the score. This was done in order to give the appearance of the score being displayed on the same display page as the game. You