Computer Archeology

Contact

HomeCoCoDaggorathCode

Site Page

Code

Good info on the math here: http://archive.li/mJKlz

The game's RAM usage is detailed here: RAM Usage

TODO: what areas are mirrored? how does the flip happen? are graphics/text areas ended differently?

TODO: make a HTML table of links to the SWI routines. Figure out what they all do.

Start

```
C000: CE C0 D1          LDU     #PlayDemo       ; Play demo game code
C003: 20 03             BRA     $C008           ; Do Demo
;
C005: CE C1 24          LDU     #PlayGame       ; Play normal game code
;
C008: 10 CE 10 00       LDS     #$1000          ; Stacks builds to lower from $1000
C00C: 8E FF 00          LDX     #$FF00          ; PIA0
C00F: CC 34 FA          LDD     #$34FA          ; A=34, B=FA
C012: A7 03             STA     3,X             ; CTRL-B = 0_0_110_1_00 : IRQs off ...
C014: A7 01             STA     1,X             ; CTRL_A = 0_0_110_1_00 : ... output register se
C016: 8E FF 20          LDX     #$FF20          ; PIA1
C019: A7 01             STA     1,X             ; CTRL-B = 0_0_110_1_00 : IRQs off
C01B: 6F 03             CLR     3,X             ; DDR selected for B
C01D: E7 02             STB     2,X             ; DDRB = 11111010 (mem and serial inputs)
C01F: 86 3C             LDA     #$3C            ; 0_0_111_1_00
C021: A7 03             STA     3,X             ; Turn off FIRQs
C023: CC 20 46          LDD     #$2046          ; 001000__0001000_110
C026: BD C2 66          JSR     WriteToSAM      ; Graphics=G6R,G6C  Display=8*512 (0x1000)
C029: 86 F8             LDA     #$F8            ; Upper bits ...
```

```
C02B: A7 02             STA     2,X             ; ... of display mode (256x192 color)
C02D: 8E 02 00          LDX     #$0200          ; Clear ...
C030: 6F 80             CLR     ,X+             ; ... all ...
C032: 8C 40 00          CMPX    #$4000          ; ... temporary ...
C035: 25 F9             BCS     $C030           ; ... memory (base page)
C037: EF E3             STU     ,--S            ; Store return to demo or game
C039: 86 02             LDA     #$02            ; DP = ...
C03B: 1F 8B             TFR     A,DP            ; ... 0200
C03D: 10 8E D7 E8       LDY     #$D7E8          ; Table of inits
C041: A6 A0             LDA     ,Y+             ; Get count
C043: 27 41             BEQ     $C086           ; All done ... move on
C045: AE A1             LDX     ,Y++            ; Get destination
C047: 8D 02             BSR     CopyYtoX        ; Do copy
C049: 20 F6             BRA     $C041           ; Do all

CopyYtoX:
; Copy bytes (A is count) from Y to X
C04B: E6 A0             LDB     ,Y+             ; Value from Y
C04D: E7 80             STB     ,X+             ; Store to X
C04F: 4A                DECA                    ; All done?
C050: 26 F9             BNE     CopyYtoX        ; No ... do all
C052: 39                RTS                     ; Out

InitTasks:
C053: 34 77             PSHS    U,Y,X,B,A,CC    ; Save all registers
C055: 1A 10             ORCC    #$10            ; Turn OFF the IRQ interrupt
C057: 8E 02 9F          LDX     #$029F          ;
C05A: 6F 80             CLR     ,X+             ;
C05C: 8C 02 AD          CMPX    #$02AD          ;
C05F: 25 F9             BCS     $C05A           ;
C061: 8E 09 FD          LDX     #$09FD          ; First game-task slot
C064: 9F B9             STX     <nextTask       ; Clear the task list
C066: 6F 80             CLR     ,X+             ; Zero out  ...
C068: 8C 0B 07          CMPX    #$0B07          ; ... game-tasks ... (Not the last task. It is i
C06B: 25 F9             BCS     $C066           ; ... slots
;
C06D: 10 8E D7 DC       LDY     #$D7DC          ; Initial task routines
C071: 0A BB             DEC     <m02BB          ;
C073: CC 00 0C          LDD     #$000C          ;
C076: AE A1             LDX     ,Y++            ; Get next task entry
C078: 27 0A             BEQ     $C084           ; All tasks made ... out
```

```
C07A: BD C2 5C           JSR     ReserveTask      ; Reserve a task slot (pointer in U)
C07D: AF 43              STX     3,U              ; Save code pointer in structure
C07F: BD C2 1D           JSR     $C21D            ; ?? Add task pointer to chain ??
C082: 20 F2              BRA     $C076            ; Do all tasks
C084: 35 F7              PULS    CC,A,B,X,Y,U,PC  ; Done

; Initialization continues from C043
C086: 8D CB              BSR     InitTasks        ; Initialize game tasks
C088: CE DA 91           LDU     #$DA91           ; Object distribution table
C08B: 4F                 CLRA                     ; First object in table is type 0 (SUPREME RING)
C08C: E6 C4              LDB     ,U               ; Get the info
C08E: C4 0F              ANDB    #$0F             ; Hold ...
C090: D7 8C              STB     <numObjs         ; ... number of objects
C092: E6 C0              LDB     ,U+              ; Get the info
C094: 54                 LSRB                     ; Hold ...
C095: 54                 LSRB                     ; ... ...
C096: 54                 LSRB                     ; ... ...
C097: 54                 LSRB                     ; ... ...
C098: D7 8D              STB     <m028D           ; ... first appear on level
C09A: 3F                 SWI                      ; Make an instance of the object
C09B: 17                                          ; SWI_17:Create object structure:
C09C: 6A 05              DEC     5,X              ; Object isn't in any list (room, player, monste
C09E: 5C                 INCB                     ; Next object on next level
C09F: C1 05              CMPB    #$05             ; Past level 4?
C0A1: 2F 02              BLE     $C0A5            ; No ... use next level
C0A3: D6 8D              LDB     <m028D           ; use
C0A5: 0A 8C              DEC     <numObjs         ; Decrement number of objects
C0A7: 26 F1              BNE     $C09A            ; Go back to create more of this type
C0A9: 4C                 INCA                     ; Next object type
C0AA: 11 83 DA A3        CMPU    #$DAA3           ; All object types done?
C0AE: 25 DC              BCS     $C08C            ; Not end of table ... go back
;
C0B0: CE 03 88           LDU     #$0388           ;
C0B3: 0A B7              DEC     <whereToPrint    ; Printing goes to desired descriptor
C0B5: 3F                 SWI
C0B6: 0A                                          ; SWI_A:Clear hand descriptor:
C0B7: 3F                 SWI
C0B8: 02                                          ; SWI_2:Uncompress message m and display:
;
C0B9: F8 DF 0C C9 27 45 00 02 65 C1 03 52 39 3C 00 68 DA CC 63 09 48 ; "COPYRIGHT  DYNA MICRO  MCML
;
```

```
C0CE: 0F B7              CLR     <whereToPrint    ; Printing goes to command area
C0D0: 39                 RTS                      ; Done

PlayDemo:
C0D1: 0A 77              DEC     <gameMode        ; Demo mode flag
C0D3: 8D 3F              BSR     EndOfTapeAccess  ; Configure interrupts
C0D5: 8E DF 10           LDX     #$DF10           ; Wizard picture
C0D8: 0A 9E              DEC     <m029E           ; ??
C0DA: 3F                 SWI                      ; Beam on the wizard
C0DB: 14                                          ; SWI_14:Beam subroutine:
C0DC: 3F                 SWI                      ; Print the first part of the message
C0DD: 02                                          ; SWI_2:Uncompress message m and display:
;
C0DE: 9F D2 02 06 45 06 4A 02 BA 85 97 BD EF 80 ; "_1F_I DARE YE ENTER..._1F_"
;
C0EC: 3F                 SWI                      ; Print the second part of the message
C0ED: 02                                          ; SWI_2:Uncompress message m and display:
;
C0EE: F7 BD EA 20 A0 25 5C 72 BD D3 03 CC 02 04 E7 7C 83 44 6F 7B ; "...THE DUNGEONS OF DAGGORATH!!
;
C102: 3F                 SWI                      ; Wait 1.35 seconds
C103: 10                                          ; SWI_10:Pause for 1.35 seconds:
C104: 3F                 SWI                      ; Another 1.35 seconds
C105: 10                                          ; SWI_10:Pause for 1.35 seconds:
C106: 3F                 SWI                      ; Beam off the wizard
C107: 15                                          ; SWI_15:Beam subroutine:
C108: 3F                 SWI                      ; Wait 1.35 seconds
C109: 09                                          ; SWI_9:Clear secondary screen:
C10A: 0A B4              DEC     <flipScreens     ; ??
C10C: 13                 SYNC                     ; Wait for draw
C10D: 86 02              LDA     #$02             ; Initial light level (means what ??)
C10F: CE D7 D5           LDU     #$D7D5           ; Demo objects
C112: 20 1D              BRA     $C131            ; Play the demo

EndOfTapeAccess:
C114: CC 34 3C           LDD     #$343C           ; 0011_0100 and 0011_1100
C117: B7 FF 21           STA     PIA1_CA          ; Motor off
C11A: F7 FF 23           STB     PIA1_CB          ; 6-bit Sound enabled
C11D: 4C                 INCA                     ; Re-enable ...
C11E: B7 FF 03           STA     PIA0_CB          ; ... 60Hz interrupt
C121: 3C EF              CWAI    $EF              ; Wait for a 60Hz interrupt to happen
```

```
C123: 39                RTS                     ; Done

PlayGame:
C124: 8D EE             BSR     EndOfTapeAccess ; Configure interrupts and sound
C126: CC 10 0B          LDD     #$100B          ; Starting cell (Y=16, X=0B)
C129: DD 13             STD     <playerY        ;
C12B: 0F 17             CLR     <pStrength      ; MSB of strength (start out weak)
C12D: 4F                CLRA                    ; Initial light level (none)
C12E: CE D7 D9          LDU     #GameObjects    ; List of game objects (not demo objects)
;
C131: 3F                SWI                     ; Print "PREPARE!"
C132: 16                                        ; SWI_16:Print PREPARE:
C133: 3F                SWI                     ; Clear light level
C134: 1A                                        ; SWI_1A:Set up level:
C135: 10 8E 02 29       LDY     #$0229          ; Pointer to 1st game object in Y
C139: A6 C0             LDA     ,U+             ; From the list of objects to make
C13B: 2B 12             BMI     $C14F           ; All done ... start the game
C13D: 3F                SWI                     ; Create the game object (type in A)
C13E: 17                                        ; SWI_17:Create object structure:
C13F: 6C 05             INC     5,X             ; Object starts out in pack
C141: 1E 13             EXG     X,U             ; X->U (SWI 18 needs it in U)
C143: 3F                SWI                     ; Initial objects start off revealed
C144: 18                                        ; SWI_18:Change object to proper name and data:
C145: 1E 13             EXG     X,U             ; Restore X and U
C147: 6F 0B             CLR     11,X            ; Already revealed
C149: AF A4             STX     ,Y              ; Link this object to last
C14B: 1F 12             TFR     X,Y             ; This is now last
C14D: 20 EA             BRA     $C139           ; Do all game objects
;
C14F: 0D 77             TST     <gameMode       ;    ?? are we in play-game mode (not demo) ??
C151: 27 13             BEQ     $C166           ; Yes ... don't start with the map
C153: 0A 9B             DEC     <m029B          ;    ??
C155: 8E CD B2          LDX     #ShowMap        ; The routine for drawing ...
C158: 9F B2             STX     <displayFunction ; ... the scroll (seer and vision)
C15A: 0A 94             DEC     <scrollType     ; This is a SEER scroll
C15C: 3F                SWI                     ; Redraw the display
C15D: 0E                                        ; SWI_E:Display playing screen:
C15E: 3F                SWI                     ; Wait for 1.35 seconds
C15F: 10                                        ; SWI_10:Pause for 1.35 seconds:
C160: 3F                SWI                     ; Another 1.35. Total: 1.35*2 = 2.7 seconds
C161: 10                                        ; SWI_10:Pause for 1.35 seconds:
```

```
C162: 0F 9B             CLR     <m029B          ;
C164: 13                SYNC                    ; Wait for two ...
C165: 13                SYNC                    ; ... interrupts (??letting the sound get starte
;
C166: 3F                SWI                     ; Draw the screen
C167: 19                                        ; SWI_19:Bring up normal display:
C168: 3F                SWI                     ; Show the command prompt
C169: 0F                                        ; SWI_F:Ready command prompt:
C16A: 7E C1 F5          JMP     GameLoop        ; Back to top of the game loop

ReadCheckError:
; Read a block from tape to buffer. Restart the computer on an error.
; X points to the buffer to fill. Block type returned in B.
C16D: BF 00 7E          STX     CASSPTR         ; Read buffer
C170: 10 3F             SWI2                    ; Read a block ...
C172: 06                ; A006: BLKIN           ; ... from tape
C173: 4D                TSTA                    ; Error?
C174: 10 26 DE AF       LBNE    $A027           ; Yes ... restart the CoCo
C178: F6 00 7C          LDB     CASSBLKTYPE     ; Return block type
C17B: 39                RTS                     ; Done

StartOfTapeAccess:
C17C: CE FF 00          LDU     #$FF00          ; Base of PIA 0
C17F: CC 34 3C          LDD     #$343C          ; 0011_0100  and 0011_1100
C182: A7 43             STA     3,U             ; Disable 60Hz interrupt
C184: B7 FF 23          STA     PIA1_CB         ; Disable cartridge-detect interrupt and sound o
C187: F7 FF 21          STB     PIA1_CA         ; Disable RS-232 interrupt and Cassette motor ON
C18A: 39                RTS                     ; Done

QuarterSecDelay:
C18B: 9E 00             LDX     <CONST_00       ;
C18D: 30 1F             LEAX    -1,X            ; Long ...
C18F: 26 FC             BNE     $C18D           ; ... delay loop
C191: 39                RTS                     ; Done

SaveToTape:
C192: 8D E8             BSR     StartOfTapeAccess; Tape motor on
C194: 8D F5             BSR     QuarterSecDelay ; Wait
C196: 8D F3             BSR     QuarterSecDelay ; Wait (1/2 second total)
C198: 10 3F             SWI2                    ; Write the tape header
C19A: 0C                ; A00C: WRTLDR
```

```
C19B: 10 3F             SWI2                    ; Write the filename (we setup this block at D7C
C19D: 08                ; A008: BLKOUT
C19E: 8D EB             BSR     QuarterSecDelay ; Wait
C1A0: 10 3F             SWI2                    ; Write the tape header again
C1A2: 0C                ; A00C: WRTLDR
C1A3: 8E 02 00          LDX     #$0200          ; Start of variables (the direct page)
C1A6: CC 01 80          LDD     #$0180          ; Block type = 1 (data) ...
C1A9: FD 00 7C          STD     CASSBLKTYPE     ; ... block size = 128 bytes
C1AC: BF 00 7E          STX     CASSPTR         ; Store for BASIC
C1AF: 10 3F             SWI2                    ; Write the block
C1B1: 08                ; A008: BLKOUT
C1B2: 8C 0F 05          CMPX    #$0F05          ; Written all of our memory?
C1B5: 25 EF             BCS     $C1A6           ; No ... back for more
C1B7: FF 00 7C          STU     CASSBLKTYPE     ; U was last set at C17C. Clever. Block type = FF
C1BA: 10 3F             SWI2                    ; Write an end block
C1BC: 08                ; A008: BLKOUT
C1BD: 8D CC             BSR     QuarterSecDelay ; Delay
C1BF: 20 2B             BRA     $C1EC           ; Turn off motor and fall into game loop

LoadFromTape:
C1C1: 8D B9             BSR     StartOfTapeAccess; Start tape access
C1C3: 10 3F             SWI2                    ; Tape on and start reading
C1C5: 04                ; A004: CRSDON
C1C6: DE 0B             LDU     <backScreen     ; ?? Off screen buffer to use for scratch ??
C1C8: AE C4             LDX     ,U              ;
C1CA: 8D A1             BSR     ReadCheckError  ; Read block
C1CC: 26 F8             BNE     $C1C6           ; Is this a header? No ... keep looking
C1CE: AE C4             LDX     ,U              ; Where we read the header
C1D0: CE 03 13          LDU     #$0313          ; Parsed filename
C1D3: C6 08             LDB     #$08            ; Is this the ...
C1D5: A6 80             LDA     ,X+             ; ... requested ...
C1D7: A1 C0             CMPA    ,U+             ; ... data file?
C1D9: 26 E6             BNE     LoadFromTape    ; No ... find the right header
C1DB: 5A                DECB                    ; Check 8 byte ...
C1DC: 26 F7             BNE     $C1D5           ; ... filename
C1DE: 10 3F             SWI2                    ; Tape on and start reading
C1E0: 04                ; A004: CRSDON
C1E1: 8E 02 00          LDX     #$0200          ; Start of variables to load (direct page)
C1E4: 8D 87             BSR     ReadCheckError  ; Read a block
C1E6: 2A FC             BPL     $C1E4           ; Keep reading if block type was not FF (end of
C1E8: 10 CE 10 00       LDS     #$1000          ; Reset stack
```

```
;
C1EC: BD C1 14          JSR     EndOfTapeAccess  ; Turn off tape and reenable interrupts
C1EF: 0F B8             CLR     <tapeTrigger     ; Tape operation complete
C1F1: 3F                SWI                      ; Draw normal display
C1F2: 19                                         ; SWI_19:Bring up normal display:
C1F3: 3F                SWI                      ; Draw ready prompt
C1F4: 0F                                         ; SWI_F:Ready command prompt:
; Fall into game loop
```

Game Loop

```
GameLoop:
C1F5: CE 02 AB          LDU     #$02AB
C1F8: 0F BB             CLR     <m02BB           ;
C1FA: 1F 32             TFR     U,Y
C1FC: 0D B8             TST     <tapeTrigger     ; ZSAVE or ZLOAD requested?
C1FE: 2E 92             BGT     SaveToTape       ; ZSAVE ... go do it
C200: 2B BF             BMI     LoadFromTape     ; ZLOAD ... go do it
C202: EE C4             LDU     ,U
C204: 27 EF             BEQ     GameLoop         ;
;
C206: 34 60             PSHS    U,Y              ; Hold registers
C208: AD D8 03          JSR     [$03,U]          ; Execute game task
C20B: 35 60             PULS    Y,U              ; Restore
;
C20D: 0D BB             TST     <m02BB           ;
C20F: 26 E4             BNE     GameLoop         ;
C211: C1 0C             CMPB    #$0C
C213: 27 E5             BEQ     $C1FA            ;
C215: 8D 21             BSR     $C238            ;
C217: 8D 04             BSR     $C21D            ;
C219: 1F 23             TFR     Y,U
C21B: 20 DF             BRA     $C1FC            ;

C21D: 34 17             PSHS    X,B,A,CC
C21F: 1A 10             ORCC    #$10             ; Turn OFF the IRQ interrupt
C221: A7 42             STA     2,U
C223: 8E 02 9F          LDX     #$029F
C226: 3A                ABX
```

```
C227: 4F                CLRA
C228: 5F                CLRB
C229: ED C4             STD     ,U
C22B: 10 A3 84          CMPD    ,X
C22E: 27 04             BEQ     $C234           ;
C230: AE 84             LDX     ,X
C232: 20 F7             BRA     $C22B           ;
C234: EF 84             STU     ,X
C236: 35 97             PULS    CC,A,B,X,PC

C238: 34 11             PSHS    X,CC
C23A: 1A 10             ORCC    #$10            ; Turn OFF the IRQ interrupt
C23C: AE C4             LDX     ,U
C23E: AF A4             STX     ,Y
C240: 35 91             PULS    CC,X,PC

C242: 34 74             PSHS    U,Y,X,B
C244: 0D 9B             TST     <m029B          ;
C246: 26 12             BNE     $C25A           ;
C248: 1F 32             TFR     U,Y
C24A: EE C4             LDU     ,U
C24C: 27 0C             BEQ     $C25A           ;
C24E: 6A 42             DEC     2,U
C250: 26 F6             BNE     $C248           ;
C252: 8D E4             BSR     $C238           ;
C254: C6 0C             LDB     #$0C
C256: 8D C5             BSR     $C21D           ;
C258: 20 EE             BRA     $C248           ;
C25A: 35 F4             PULS    B,X,Y,U,PC

ReserveTask:
; Move the next-task-pointer to the next seven-byte slot.
; Return reserved slot pointer in U.
C25C: 34 10             PSHS    X               ; Hold X
C25E: DE B9             LDU     <nextTask       ; Get the slot pointer
C260: 30 47             LEAX    7,U             ; Point to next
C262: 9F B9             STX     <nextTask       ; New slot pointer
C264: 35 90             PULS    X,PC            ; Out

WriteToSAM:
; The SAM chip is mapped only to the address bus. Writing to an even adress
```

```
; clears the target register bit. Writing to the next (odd) address sets the
; bit. This routine copies the 10-bit value in D to the first 10 SAM registers.
;
; The SAM is 16 bits and the value passed is 16 bits, and looks correct for
; the full register set. But this routine only changes the lowest 10 bits.
;
; F6 F5 F4 F3 F2 F1 F0 V2 V1 V0
;
; V is the video mode
; F is the video memory address (F*512 is the address)
;
C266: 34 16             PSHS    X,B,A           ; Preserve the registers
C268: 8E FF C0          LDX     #$FFC0          ; Start of SAM memory
C26B: 44                LSRA                    ; Next bit ...
C26C: 56                RORB                    ; ... into carry
C26D: 25 03             BCS     $C272           ; It is a one ... set the bit
C26F: A7 84             STA     ,X              ; Set the zero (even address)
C271: 8C                ; CMPX  opcode to skip next instruction
C272: A7 01             STA     1,X             ; Set the one (odd address)
C274: 30 02             LEAX    2,X             ; Next register bit
C276: 8C FF D4          CMPX    #$FFD4          ; All done?
C279: 25 F0             BCS     $C26B           ; No ... keep going
C27B: 35 96             PULS    A,B,X,PC        ; Restore and out
```

Interrupt Service

```
InterruptServiceRoutine:
C27D: 8E FF 20          LDX     #$FF20          ; 6-bit sound value
C280: A6 88 E3          LDA     -$1D,X          ; FF03 ... 16.67MS (60Hz) interrupt status
C283: 10 2A 00 99       LBPL    $C320           ; Upper bit 0 ... must have been the horizontal
C287: 86 02             LDA     #$02            ; Set DP to ...
C289: 1F 8B             TFR     A,DP            ; ... base 02xx (in case we are interrupting a B
C28B: 0D B4             TST     <flipScreens    ; Time to flip screens?
C28D: 27 0E             BEQ     $C29D           ; No ... keep what we have
C28F: DC 09             LDD     <activeScreen   ; Get the current visible
C291: DE 0B             LDU     <backScreen     ; Get the current drawing
C293: DD 0B             STD     <backScreen     ; Flip the ...
C295: DF 09             STU     <activeScreen   ; ... visible and drawing screens
C297: EC 44             LDD     4,U             ; Get the SAM settings for the new visible scree
```

```
C299: 8D CB             BSR     WriteToSAM      ; Set the SAM registers to flip the screen
C29B: 0F B4             CLR     <flipScreens    ; Acknowledge the flip
;
C29D: 0D 9C             TST     <beamSound      ; Is the wizard beaming in or out (cut scenes)?
C29F: 27 08             BEQ     $C2A9           ; No ... skip it
C2A1: 03 9D             COM     <beamSoundVal   ; Toggle the wizard sound square wave
C2A3: 96 9D             LDA     <beamSoundVal   ; Get sound value
C2A5: 48                ASLA                    ; 8 bit to ...
C2A6: 48                ASLA                    ; ... 6 bit value (divide by 4)
C2A7: A7 84             STA     ,X              ; Store to FF20 (6 bit sound)
;
C2A9: 0D B1             TST     <hearHeart      ; Are we between rounds?
C2AB: 27 2F             BEQ     $C2DC           ; Yes .. no heart
C2AD: 0A AE             DEC     <heartCounter   ; Time to change heart pattern?
C2AF: 26 2B             BNE     $C2DC           ; No ... skip it
C2B1: 96 AF             LDA     <heartCounterRel ; Reload the ...
C2B3: 97 AE             STA     <heartCounter   ; ... heart counter
C2B5: E6 02             LDB     2,X             ; FF22 ... current single-bit sound
C2B7: C8 02             EORB    #$02            ; Toggle the single-bit ...
C2B9: E7 02             STB     2,X             ; ... sound (makes a pop)
C2BB: 0D AD             TST     <scrollShowing  ; Scroll showing?
C2BD: 27 1D             BEQ     $C2DC           ; Yes ... skip drawing the heart
C2BF: CE 03 88          LDU     #$0388          ; Hand line area ?active or inactive?
C2C2: AE 44             LDX     4,U             ; Hold onto current cursor (we might be printing
C2C4: CC 00 0F          LDD     #$000F          ; New cursor ...
C2C7: ED 44             STD     4,U             ; ... middle of the line
C2C9: 86 20             LDA     #$20            ; 20, 21 ... small-heart characters
C2CB: 03 B0             COM     <heartPicture   ; Toggle heart picture tracking
C2CD: 27 02             BEQ     $C2D1           ; Zero now? Draw the small heart
C2CF: 86 22             LDA     #$22            ; 22, 23 ... large-heart characters
C2D1: BD CA 17          JSR     PrintRegChar    ; Draw the first heart character
C2D4: 6C 45             INC     5,U             ; Bump the cursor for the second picture
C2D6: 4C                INCA                    ; Second heart picture
C2D7: BD CA 17          JSR     PrintRegChar    ; Draw the second heart character
C2DA: AF 44             STX     4,U             ; Restore the text cursor
;
C2DC: CE 02 A1          LDU     #$02A1
C2DF: BD C2 42          JSR     $C242           ;
C2E2: 8E 02 95          LDX     #$0295
C2E5: 10 8E C3 24       LDY     #$C324
C2E9: 6C 84             INC     ,X
```

```
C2EB: 8C 02 9A          CMPX    #$029A
C2EE: 27 0F             BEQ     $C2FF           ;
C2F0: A6 84             LDA     ,X
C2F2: A1 A0             CMPA    ,Y+
C2F4: 2D 09             BLT     $C2FF           ;
C2F6: 6F 80             CLR     ,X+
C2F8: 33 42             LEAU    2,U
C2FA: BD C2 42          JSR     $C242           ;
C2FD: 20 EA             BRA     $C2E9           ;
;
C2FF: 0D 28             TST     <fainting       ; Are we fainting?
C301: 26 1D             BNE     $C320           ; Yes .. ??
C303: 0D 77             TST     <gameMode       ; Are we in a live game?
C305: 27 11             BEQ     $C318           ; Yes ... don't restart the game on a key
C307: 7F FF 02          CLR     PIA0_DB         ; Activate all keyboard columns
C30A: B6 FF 00          LDA     PIA0_DA         ; Check all rows
C30D: 84 7F             ANDA    #$7F            ; Ignore the joystick bit
C30F: 81 7F             CMPA    #$7F            ; Any key pressed?
C311: 27 0D             BEQ     $C320           ; No ... skip processing the key
C313: 8E C0 05          LDX     #$C005          ; Any key was pressed ... start a new game
C316: AF 6A             STX     10,S            ; Return address
C318: 10 3F             SWI2                    ; BASIC function ...
C31A: 00                                        ; ... POLCAT
C31B: 4D                TSTA                    ; Was it a valid key (not, say, SHIFT)
C31C: 27 02             BEQ     $C320           ; No ... don't store it (but we are still starti
C31E: 8D 20             BSR     CharToBuf       ; Store character in ring buffer
C320: B6 FF 02          LDA     PIA0_DB         ; Acknowledge the interrupt so it can fire again
C323: 3B                RTI                     ; Return from interrupt

C324: 06 0A 3C 3C 18 ; ? Counters for task levels ?

CharFromBuf:
; Read character A from input ring buffer and advance the tail. Return 0
; if nothing to read.
C329: 34 15             PSHS    X,B,CC          ; Save
C32B: 1A 10             ORCC    #$10            ; Turn OFF the IRQ interrupt
C32D: 4F                CLRA                    ; Initial return ... no key in buffer
C32E: 8E 02 D1          LDX     #$02D1          ; 32-byte input ring buffer
C331: D6 BC             LDB     <inputHead      ; The ring-buffer head
C333: D1 BD             CMPB    <inputTail      ; Same as the ring-buffer tail?
C335: 27 07             BEQ     $C33E           ; Yes ... return 0 (no input)
```

```
C337: A6 85             LDA     B,X             ; Get the next character from the head
C339: 5C                INCB                    ; Advance the head ...
C33A: C4 1F             ANDB    #$1F            ; ... and wrap ...
C33C: D7 BC             STB     <inputHead      ; ... if needed
C33E: 35 95             PULS    CC,B,X,PC       ; Done

CharToBuf:
; Save character A to input ring buffer and advance tail
C340: 34 15             PSHS    X,B,CC          ; Save all
C342: 1A 10             ORCC    #$10            ; Turn OFF the IRQ interrupt
C344: 8E 02 D1          LDX     #$02D1          ; 32-byte ring buffer
C347: D6 BD             LDB     <inputTail      ; Get tail index
C349: A7 85             STA     B,X             ; Store the character to the tail
C34B: 5C                INCB                    ; Advance the tail ...
C34C: C4 1F             ANDB    #$1F            ; ... and wrap ...
C34E: D7 BD             STB     <inputTail      ; ... if needed
C350: 35 95             PULS    CC,B,X,PC       ; Done
```

SWI Handler

TODO discussion about this technique

```
SWIHandler:
;
C352: 1C EF             ANDCC   #$EF            ; Re-enable the IRQ
C354: AE 6A             LDX     10,S            ; The Program Counter in the calling frame
C356: A6 80             LDA     ,X+             ; Get the interrupt number immediate
C358: AF 6A             STX     10,S            ; Update the calling frame program counter
C35A: 8E C3 84          LDX     #$C384          ; Offset of first routine
C35D: CE C9 95          LDU     #$C995          ; SWI routine offset bytes
C360: E6 C0             LDB     ,U+             ; Get offset byte to routine
C362: 3A                ABX                     ; Add it to code pointer
C363: 4A                DECA                    ; Found the one we are looking for?
C364: 2A FA             BPL     $C360           ; No ... keep off-setting
C366: AF E3             STX     ,--S            ; Push the address of the routine on the stack
C368: EC 63             LDD     3,S             ; A and B passed from caller
C36A: AE 66             LDX     6,S             ; X from the caller
C36C: EE 6A             LDU     10,S            ; U from the caller
C36E: AD F1             JSR     [,S++]          ; Do the SWI routine
C370: 3B                RTI                     ; Return to caller
```

```
SWI2Handler:
; Execute one of the vectored routines in the BASIC ROM
; 00 = POLCAT
; 02 = CHROUT
; 04 = CSRDON
; 06 = BLKIN
; 08 = BLKOUT
; 0A = JOYIN
; 0C = WRTLDR
;
C371: 5F                CLRB                    ; BASIC functions need ...
C372: 1F 9B             TFR     B,DP            ; ... DP = 0
C374: EE 6A             LDU     10,S            ; The Program Counter in the calling frame
C376: E6 C0             LDB     ,U+             ; Get the interrupt number immediate
C378: EF 6A             STU     10,S            ; Update the calling frame program counter
C37A: CE A0 00          LDU     #$A000          ; BASIC function jump table
C37D: AD D5             JSR     [B,U]           ; Call the BASIC function
C37F: A7 61             STA     1,S             ; Return A value to caller
C381: AF 64             STX     4,S             ; Return Y value to caller
C383: 3B                RTI                     ; Return to caller

SWI_0:
; Light level
C384: 96 6E             LDA     <m026E          ;
C386: 0D 75             TST     <m0275          ;
C388: 27 04             BEQ     $C38E           ;
C38A: 96 6F             LDA     <m026F          ;
C38C: 0F 75             CLR     <m0275          ;
C38E: 5F                CLRB
C38F: 80 07             SUBA    #$07
C391: 90 8B             SUBA    <m028B          ;
C393: 2C 0A             BGE     $C39F           ;
C395: 5A                DECB
C396: 81 F9             CMPA    #$F9
C398: 2F 05             BLE     $C39F           ;
C39A: 8E CB 96          LDX     #$CB96
C39D: E6 86             LDB     A,X
C39F: D7 2D             STB     <dotFrequency   ; New dot frequency
C3A1: 39                RTS
```

```
SWI_1:
; Draw picture X on screen
;  X: points to picture script
;
C3A2: 0F 51              CLR     <m0251          ; Starting new line segment
C3A4: 96 2D              LDA     <dotFrequency   ; Dot Frequency
C3A6: 4C                 INCA                    ; Anything to draw?
C3A7: 27 4D              BEQ     $C3F6           ; No, out
;
C3A9: E6 84              LDB     ,X              ; Else get command
C3AB: C0 FA              SUBB    #$FA            ; Is this a command?
C3AD: 25 20              BCS     $C3CF           ; No, go to standard line
C3AF: 30 01              LEAX    1,X             ; Next in list
C3B1: 10 8E C3 B9        LDY     #$C3B9          ; Graphics Commands
C3B5: E6 A5              LDB     B,Y             ; Get offset
C3B7: 6E A5              JMP     B,Y             ; Go to command
;
; Special graphics commands
;        00-FA: Standard line command
C3B9: 10  ; FA: Return from graphics subroutine
C3BA: 06  ; FB: Jump to subroutine
C3BB: 5E  ; FC: Multiple short segments
C3BC: 0D  ; FD: Jump to xxxx
C3BD: 3D  ; FE: Exit
C3BE: 12  ; FF: Start a new segment
;
; Command FB: Jump to subroutine
C3BF: EC 81              LDD     ,X++            ; Get new address
C3C1: AF E3              STX     ,--S            ; Save return
C3C3: 1F 01              TFR     D,X             ; D->X
; Command FD: Jump to XXXX
C3C5: 8C                 ; CMPX  opcode to skip next instruction
C3C6: AE 84              LDX ,X                  ; Jump address from X
; Command FA: Return from graphics subroutine
C3C8: 8C                 ; CMPX  opcode to skip next instruction
C3C9: AE E1              LDX ,S++                ; Jump address from stack
; Command FF: Start a new segment
C3CB: 0F 51              CLR     <m0251          ; New segment
C3CD: 20 DA              BRA     $C3A9           ; Continue
;
; Regular line command
```

```
C3CF: 0D 51             TST     <m0251          ; Already have start point?
C3D1: 26 06             BNE     $C3D9           ; Yes, skip this
C3D3: 8D 0D             BSR     $C3E2           ; Get coordinates
C3D5: 0A 51             DEC     <m0251          ; Flag now have a start
C3D7: 20 D0             BRA     $C3A9           ; Continue
;
C3D9: 8D 05             BSR     $C3E0           ; Set up new segment
C3DB: BD CA B7          JSR     $CAB7           ; Draw line
C3DE: 20 C9             BRA     $C3A9           ; Back for more
;
C3E0: 8D 15             BSR     $C3F7           ; Move old end to new start
C3E2: E6 80             LDB     ,X+             ; Y coordinate
C3E4: D7 54             STB     <m0254          ; Hold on to it
C3E6: 8D 18             BSR     $C400           ;
C3E8: D3 07             ADDD    <m0207          ; Y center of screen
C3EA: DD 33             STD     <m0233          ; Store new end Y
C3EC: E6 80             LDB     ,X+             ; X coordinate
C3EE: D7 52             STB     <m0252          ; Hold on to it
C3F0: 8D 14             BSR     $C406           ;
C3F2: D3 05             ADDD    <m0205          ; X center of screen
C3F4: DD 35             STD     <m0235          ; Store new end X
; Command FE: Exit
C3F6: 39                RTS                     ; Done
;
C3F7: DC 33             LDD     <m0233          ; Move old Y...
C3F9: DD 2F             STD     <m022F          ; ... to new Y
C3FB: DC 35             LDD     <m0235          ; Move old X
C3FD: DD 31             STD     <m0231          ; ... to new X
C3FF: 39                RTS                     ; Done
;
C400: 96 50             LDA     <m0250          ; Y Scale factor
C402: D0 08             SUBB    <m0207+01       ; Y byte center of screen
C404: 20 04             BRA     $C40A           ; Handle signed multiply
;
C406: 96 4F             LDA     <m024F          ; X scale factor
C408: D0 06             SUBB    <m0205+01       ; X byte center of screen
;
C40A: 25 03             BCS     $C40F           ; Handle signed multiply
C40C: 3D                MUL                     ; Do multiplication
C40D: 20 05             BRA     $C414           ;
C40F: 50                NEGB                    ; Make it positive
```

```
C410: 3D                MUL                     ; Do multiply
C411: BD CA 99          JSR     $CA99           ; Negate D
C414: 7E D3 77          JMP     DRight7         ; Divide D by 128 (fractional math)
;
; Command FC: Multiple short segments
C417: A6 80             LDA     ,X+             ; Next description
C419: 27 B0             BEQ     $C3CB           ; 0 means done
C41B: 8D DA             BSR     $C3F7           ; Move old end to new beginning
C41D: E6 1F             LDB     -1,X            ; Byte
C41F: 57                ASRB                    ; Upper ...
C420: 57                ASRB                    ; ... 4 bits ...
C421: 57                ASRB                    ; ... ...
C422: 57                ASRB                    ; ... ...
C423: 58                ASLB                    ; ... *2
C424: DB 54             ADDB    <m0254          ; Add to old Y
C426: D7 54             STB     <m0254          ; New Y
C428: 8D D6             BSR     $C400           ; Do multiply and prepare
C42A: D3 07             ADDD    <m0207          ; Offset center of screen
C42C: DD 33             STD     <m0233          ; Save new Y
C42E: E6 1F             LDB     -1,X            ; Descriptor
C430: C4 0F             ANDB    #$0F            ; Lower four bits
C432: C5 08             BITB    #$08            ; Is this negative?
C434: 27 02             BEQ     $C438           ; No,
C436: CA F0             ORB     #$F0            ; Else make it negative
C438: 58                ASLB                    ; *2
C439: DB 52             ADDB    <m0252          ; Offset X coordinate
C43B: D7 52             STB     <m0252          ; Store New
C43D: 8D C7             BSR     $C406           ; Scale it
C43F: D3 05             ADDD    <m0205          ; Add offset to center
C441: DD 35             STD     <m0235          ; Absolute coordinate
C443: BD CA B7          JSR     $CAB7           ; Draw line segment
C446: 20 CF             BRA     $C417           ; Continue multiple segments

SWI_2:
; Uncompress message m and display
;  Message bytes are in the code at the call site. Flow continues
;  after the compressed data.
;
C448: AE 6C             LDX     12,S            ; PC from stack
C44A: 3F                SWI                     ; Decompress the message
C44B: 05                                        ; SWI_5:Uncompress message X to buffer:
```

```
C44C: AF 6C              STX     12,S            ; Update the PC to skip message
C44E: 8E 03 35           LDX     #$0335          ; Temporary buffer
C451: 8C                 ; CMPX  opcode to skip next instruction
C452: 3F                 SWI                     ; Print character in A
C453: 04                                         ; SWI_4:Display a single character in A:

SWI_3:
; Display uncompressed message pointed to by X
;  X: points to uncompressed data
C454: A6 80              LDA     ,X+             ; Next character from X
C456: 2A FA              BPL     $C452           ; Printable ... do it
C458: 39                 RTS                     ; End of string

SWI_4:
; Display a single character in A
;  A: the character
;  U: the area descriptor (ignored if <$B7!=0)
C459: 0D B7              TST     <whereToPrint   ; Put text in command window?
C45B: 26 03              BNE     $C460           ; No ... use the requested descriptor
C45D: CE 03 90           LDU     #comStart       ; Yes ... print on the upper half of the screen
C460: AE 44              LDX     4,U             ; Current cursor
C462: BD C9 B2           JSR     PrintCharCRBS   ; Draw the character and advance the cursor
C465: AC 42              CMPX    2,U             ; Filled this area up?
C467: 25 03              BCS     $C46C           ; No ... no scrolling
C469: BD C9 D4           JSR     ScrollTextArea  ; Yes ... scroll the text area
C46C: AF 44              STX     4,U             ; New text cursor offset
C46E: 39                 RTS                     ; Done

SWI_5:
; Uncompress message X to buffer
;  X: the compressed message
C46F: CE 03 35           LDU     #$0335

SWI_6:
; Uncompress message X to given buffer U
;  X: the compressed message
;  U: the buffer
C472: 31 5F              LEAY    -1,U            ; First in buffer holds the algorithm spot
C474: 6F A4              CLR     ,Y              ; Start with 0
C476: 8D 14              BSR     $C48C           ; Get next byte to B
C478: 1F 98              TFR     B,A             ; A holds the length
```

```
C47A: 8D 10             BSR     $C48C           ; Get the byte
C47C: E7 C0             STB     ,U+             ; Store to buffer
C47E: 4A                DECA                    ; All done?
C47F: 2A F9             BPL     $C47A           ; No ... keep going
C481: A7 C4             STA     ,U              ; Store terminator FF
C483: 6D A4             TST     ,Y              ; Did the algorithm finish on a byte boundary?
C485: 27 02             BEQ     $C489           ; Yes ... X is pointing to next correctly
C487: 30 01             LEAX    1,X             ; Advance ...
C489: AF 66             STX     6,S             ; ... X to next message
C48B: 39                RTS                     ; Done

; Uncompress routine
; Y points to algorithm number
; Return byte in B (5 bits)
; Advance compression to next
;
; The algorithm decompresses 8 5-bit values into 5 bytes
; There are 8 algorithm steps to peel out the 5 bit value and advance the pointer
; if the next step needs it advanced.
;
; AAAAABBB BBCCCCCD DDDDEEEE EFFFFFGG GGGHHHHH
;
C48C: 34 42             PSHS    U,A             ; Hold buffer and length
C48E: A6 A4             LDA     ,Y              ; Get algorithm spot
C490: CE C4 A2          LDU     #$C4A2          ; Offset table
C493: A6 C6             LDA     A,U             ; Get offset
C495: AD C6             JSR     A,U             ; Execute next decompress step
C497: A6 A4             LDA     ,Y              ; Bump ...
C499: 4C                INCA                    ; ... to next step
C49A: 84 07             ANDA    #$07            ; Rolls back to 0
C49C: A7 A4             STA     ,Y              ; Next step
C49E: C4 1F             ANDB    #$1F            ; Decompressed is 5 bit
C4A0: 35 C2             PULS    A,U,PC          ; Return
;
C4A2: 08 ; C4A2 + 08 = C4AA  AAAAA
C4A3: 0E ; C4A2 + 0E = C4B0  BBBBB
C4A4: 13 ; C4A2 + 13 = C4B5  CCCCC
C4A5: 17 ; C4A2 + 17 = C4B9  DDDDD
C4A6: 1C ; C4A2 + 1C = C4BE  EEEEE
C4A7: 21 ; C4A2 + 21 = C4C3  FFFFF
C4A8: 25 ; C4A2 + 25 = C4C7  GGGGG
```

```
C4A9: 2A ; C4A2 + 2A = C4CC  HHHHH
;
; Get AAAAA
C4AA: E6 84             LDB     ,X              ; B = X
C4AC: 54                LSRB                    ; B = B >> 3  (0 in bit 7)
C4AD: 54                LSRB                    ; ...
C4AE: 54                LSRB                    ; ...
C4AF: 39                RTS                     ; Done
;
; Get BBBBB
C4B0: EC 80             LDD     ,X+             ; D = X++
C4B2: 7E D3 79          JMP     DRight6         ; D = D>>6
;
; Get CCCCC
C4B5: E6 84             LDB     ,X              ; B = X
C4B7: 20 F5             BRA     $C4AE           ; B = B>>1
;
; Get DDDDD
C4B9: EC 80             LDD     ,X+             ; D = X++
C4BB: 7E D3 7D          JMP     DRight4         ; D = D>>4
;
; Get EEEEE
C4BE: EC 80             LDD     ,X+             ; D = X++
C4C0: 7E D3 77          JMP     DRight7         ; D = D>>7
;
; Get FFFFF
C4C3: E6 84             LDB     ,X              ; B = X
C4C5: 20 E6             BRA     $C4AD           ; B = B>>2
;
; Get GGGGG
C4C7: EC 80             LDD     ,X+             ; D = X++
C4C9: 7E D3 7B          JMP     DRight5         ; D = D>>5
;
; Get HHHHH
C4CC: E6 80             LDB     ,X+             ; B = X++
C4CE: 39                RTS                     ; Done

SWI_7:
; Get random number
;  Return A: random byte
C4CF: 8E 00 08          LDX     #$0008          ; 8 rolls
```

```
;
C4D2: 5F                CLRB                    ; Count of 1's
C4D3: 10 8E 00 08       LDY     #$0008          ; Counting 8 bits in the byte
C4D7: 96 6D             LDA     <rndSeedC       ; Upper most seed
C4D9: 84 E1             ANDA    #$E1            ; 1110_0001
C4DB: 48                ASLA                    ; Count ...
C4DC: 24 01             BCC     $C4DF           ; ... the ...
C4DE: 5C                INCB                    ; ... 1's ...
C4DF: 31 3F             LEAY    -1,Y            ; ... in the ...
C4E1: 26 F8             BNE     $C4DB           ; ... value in A
;
C4E3: 54                LSRB                    ; 1 to carry if number of 1's was odd
C4E4: 09 6B             ROL     <rndSeedA       ; Three ...
C4E6: 09 6C             ROL     <rndSeedB       ; ... byte ...
C4E8: 09 6D             ROL     <rndSeedC       ; ... roll ...
C4EA: 30 1F             LEAX    -1,X            ; ... with B going ...
C4EC: 26 E4             BNE     $C4D2           ; ... far right ...
;
C4EE: 96 6B             LDA     <rndSeedA       ;
C4F0: A7 63             STA     3,S             ; Return the value in A
C4F2: 39                RTS                     ; Done

SWI_8:
; Clear display screen
;
C4F3: DE 09             LDU     <activeScreen   ; U = Visible screen descriptor
C4F5: 8C                ; CMPX  opcode to skip next instruction
;
SWI_9:
; Clear secondary screen
;
C4F6: DE 0B             LDU     <backScreen     ; Drawing screen descriptor
C4F8: D6 2C             LDB     <backgroundColor ; Background color (00 or FF)
C4FA: 8D 1B             BSR     $C517           ; Clear the area
C4FC: EF 6A             STU     10,S            ; Return pointer to the descriptor
C4FE: 39                RTS                     ; Done

SWI_A:
; Clear hand descriptor
;  (both screen buffers)
C4FF: 8E 03 88          LDX     #$0388          ; Hand descriptor space
```

```
C502: CE D8 7C          LDU     #$D87C          ; Clearing Data
C505: 20 06             BRA     $C50D           ; Clear out hand descriptor

SWI_B:
; Clear play field
;  (both screen buffers)
C507: 8E 03 90          LDX     #$0390          ; Playing field
C50A: CE D8 88          LDU     #$D888          ; 1st screen buffer area
;
C50D: 6F 04             CLR     4,X             ; Move cursor...
C50F: 6F 05             CLR     5,X             ; ... to beginning
C511: E6 06             LDB     6,X             ; Color
C513: 8D 02             BSR     $C517           ; Clear space pointed to by U
C515: 33 46             LEAU    6,U             ; Now clear 2nd screen buffer area
;
C517: 34 76             PSHS    U,Y,X,B,A       ; Hold all
C519: 1D                SEX                     ; Expand color mask to word
C51A: 1F 02             TFR     D,Y             ; D->Y for a double word
C51C: 30 C4             LEAX    ,U              ; First address in space
C51E: EE 42             LDU     2,U             ; Last address in space
C520: 36 26             PSHU    Y,B,A           ; Wipe 2 word area, back up two
C522: 11 A3 84          CMPU    ,X              ; Done?
C525: 26 F9             BNE     $C520           ; No, do all
C527: 35 F6             PULS    A,B,X,Y,U,PC    ; Out

SWI_C:
; Update heart rate
C529: 0F C1             CLR     <holdHole       ;
C52B: DC 17             LDD     <pStrength      ; Strength
C52D: DD C2             STD     <m02C2          ;
C52F: 86 06             LDA     #$06
C531: 08 C3             LSL     <m02C3          ;
C533: 09 C2             ROL     <m02C2          ;
C535: 09 C1             ROL     <holdHole       ;
C537: 4A                DECA
C538: 26 F7             BNE     $C531           ;
C53A: 0F C4             CLR     <m02C4          ;
C53C: DC 21             LDD     <m0221          ;
C53E: DD C5             STD     <m02C5          ;
C540: 08 C6             LSL     <m02C6          ;
C542: 09 C5             ROL     <m02C5          ;
```

```
C544: 09 C4             ROL     <m02C4          ;
C546: DC 17             LDD     <pStrength      ;
C548: D3 C5             ADDD    <m02C5          ;
C54A: DD C5             STD     <m02C5          ;
C54C: D6 C4             LDB     <m02C4          ;
C54E: C9 00             ADCB    #$00
C550: D7 C4             STB     <m02C4          ;
C552: 0F C7             CLR     <m02C7          ;
C554: DC C2             LDD     <m02C2          ;
C556: 93 C5             SUBD    <m02C5          ;
C558: DD C2             STD     <m02C2          ;
C55A: 96 C1             LDA     <holdHole       ;
C55C: 92 C4             SBCA    <m02C4          ;
C55E: 97 C1             STA     <holdHole       ;
C560: 0C C7             INC     <m02C7          ;
C562: 24 F0             BCC     $C554           ;
C564: 96 C7             LDA     <m02C7          ;
C566: 80 13             SUBA    #$13
C568: 97 AF             STA     <heartCounterRel ;
C56A: 0D 28             TST     <fainting       ; Are we fainting?
C56C: 26 27             BNE     $C595           ; Yes ... ??
C56E: 81 03             CMPA    #$03
C570: 2E 3C             BGT     $C5AE           ;
C572: 3F                SWI
C573: 0B                                        ; SWI_B:Clear play field:
C574: 96 6E             LDA     <m026E          ;
C576: 97 70             STA     <m0270          ;
C578: 0A 6F             DEC     <m026F          ;
C57A: AD 9F 02 B2       JSR     [displayFunction]; Display playing screen
C57E: 0A B4             DEC     <flipScreens    ;
C580: 13                SYNC                    ; Wait for display
C581: 0A 6E             DEC     <m026E          ; Light level down
C583: 96 6E             LDA     <m026E          ; Down
C585: 81 F8             CMPA    #$F8            ; All fainted out?
C587: 2E EF             BGT     $C578           ; No, keep going
C589: 3F                SWI                     ; Clear screen
C58A: 09                                        ; SWI_9:Clear secondary screen:
C58B: 0A B4             DEC     <flipScreens    ;
C58D: 0A 28             DEC     <fainting       ; Decrement the faint counter
C58F: 0F BC             CLR     <inputHead      ;
C591: 0F BD             CLR     <inputTail      ;
```

```
C593: 20 19              BRA     $C5AE           ;
C595: 81 04              CMPA    #$04
C597: 2F 15              BLE     $C5AE           ;
C599: AD 9F 02 B2        JSR     [displayFunction]; Display playing screen
C59D: 0A B4              DEC     <flipScreens    ;
C59F: 13                 SYNC                    ; Wait a display
C5A0: 0C 6F              INC     <m026F          ;
C5A2: 0C 6E              INC     <m026E          ;
C5A4: 96 6E              LDA     <m026E          ;
C5A6: 91 70              CMPA    <m0270          ;
C5A8: 2F EF              BLE     $C599           ;
C5AA: 0F 28              CLR     <fainting       ; No longer fainting
C5AC: 3F                 SWI                     ; Display playing screen
C5AD: 0F                                         ; SWI_F:Ready command prompt:
C5AE: 9E 17              LDX     <pStrength      ; Strength
C5B0: 9C 21              CMPX    <m0221          ; Heart level
C5B2: 25 01              BCS     $C5B5           ; Can not support it, die
C5B4: 39                 RTS

; Player is dead!

C5B5: 8E DF 10           LDX     #$DF10          ; Beam on Moon Wizard (not Star Wizard)
C5B8: 0A 9E              DEC     <m029E          ;
C5BA: 3F                 SWI
C5BB: 13                                         ; SWI_13:Beam on picture pointed to by X:
C5BC: 3F                 SWI                     ; Print "Yet Another Does Not Return"
C5BD: 02                                         ; SWI_2:Uncompress message m and display:
;
C5BE: FF C1 92 D0 01 73 E8 82 C8 04 79 66 07 3E 80 91 69 59 3B DE F0 ; "_1F_ YET ANOTHER DOES NOT RI
 ;
C5D3: 0F 28              CLR     <fainting       ; No longer fainting
C5D5: 0A 77              DEC     <gameMode       ;
;
C5D7: 20 FE              BRA     $C5D7           ; Endless loop

SWI_D:
; Print contents of hands on status line

C5D9: CE 03 88           LDU     #$0388          ; Hand line descriptor
C5DC: 0A B7              DEC     <whereToPrint   ; Force print to desired descriptor
C5DE: 96 2C              LDA     <backgroundColor ; Base color
```

```
C5E0: 43                COMA                    ; Hands are reverse color
C5E1: A7 46             STA     6,U             ; New color
C5E3: 4F                CLRA                    ; Coordinates...
C5E4: 5F                CLRB                    ; ...far left
C5E5: 8D 22             BSR     $C609           ; Blank left hand slot
C5E7: ED 44             STD     4,U             ; Reposition cursor
C5E9: 9E 1D             LDX     <leftHand       ; Left hand object
C5EB: 8D 2A             BSR     GetObjDscrpt    ; Create string
C5ED: 3F                SWI                     ; And print
C5EE: 03                                        ; SWI_3:Display uncompressed message pointed to
C5EF: CC 00 11          LDD     #$0011          ; Start of right hand space
C5F2: 8D 15             BSR     $C609           ; Blank right hand area
C5F4: 9E 1F             LDX     <rightHand      ; Right hand object
C5F6: 8D 1F             BSR     GetObjDscrpt    ; Decode it
C5F8: 1F 12             TFR     X,Y             ; Over to Y
C5FA: CC 00 21          LDD     #$0021          ; Far right coordinates
C5FD: 5A                DECB                    ; Shift right hand from right
C5FE: 6D A0             TST     ,Y+             ; All accounted for?
C600: 2A FB             BPL     $C5FD           ; No, keep counting from right
C602: ED 44             STD     4,U             ; Coordinates for right hand
C604: 3F                SWI                     ; Print right hand contents
C605: 03                                        ; SWI_3:Display uncompressed message pointed to
C606: 0F B7             CLR     <whereToPrint   ; Printing goes to command line area now
C608: 39                RTS                     ; Done
;
C609: 34 06             PSHS    B,A             ; Hold these
C60B: ED 44             STD     4,U             ; Coordinates
C60D: CC 00 0F          LDD     #$000F          ; 15
C610: 3F                SWI                     ; Print a space
C611: 04                                        ; SWI_4:Display a single character in A:
C612: 5A                DECB                    ; All blanked?
C613: 26 FB             BNE     $C610           ; No, blank all
C615: 35 86             PULS    A,B,PC          ; Done

GetObjDscrpt:
; Unpack the words to build the description for the given object.
; X = pointer to object
; Return X = pointer to buffer
C617: 34 66             PSHS    U,Y,B,A         ; Hold these
C619: 31 84             LEAY    ,X              ; Get pointer to object
C61B: 26 05             BNE     $C622           ; Yes, it is something
```

```
C61D: 8E C6 50           LDX     #$C650          ; EMPTY message
C620: 20 1A              BRA     $C63C           ; Skip all decoding and use EMPTY
C622: CE 03 13           LDU     #$0313          ; Buffer for hand printing
C625: 6D 2B              TST     11,Y            ; Revealed?
C627: 26 09              BNE     $C632           ; No, skip proper name
C629: A6 29              LDA     9,Y             ; Get proper name token
C62B: 8E D8 F4           LDX     #$D8F4          ; Proper name table
C62E: 8D 0E              BSR     $C63E           ; Find proper name
C630: 6F 5F              CLR     -1,U            ; Stick space on end of proper name
C632: A6 2A              LDA     10,Y            ; Object class
C634: 8E D9 6B           LDX     #$D96B          ; Class name table
C637: 8D 05              BSR     $C63E           ; Find class name
C639: 8E 03 13           LDX     #$0313          ; Return pointer to buffer
C63C: 35 E6              PULS    A,B,Y,U,PC      ; Done
;
C63E: 34 12              PSHS    X,A             ; Hold these
C640: 3F                 SWI                     ; Uncompress message
C641: 05                                         ; SWI_5:Uncompress message X to buffer:
C642: 4A                 DECA                    ; Found proper one?
C643: 2A FB              BPL     $C640           ; No, keep going
C645: 8E 03 36           LDX     #$0336          ; Uncompress buffer
C648: A6 80              LDA     ,X+             ; Copy from uncompressed...
C64A: A7 C0              STA     ,U+             ; ... to hand string
C64C: 2A FA              BPL     $C648           ; Copy all, including end marker
C64E: 35 92              PULS    A,X,PC          ; Done

C650: 05 0D 10 14 19     ; 'EMPTY'
C655: FF                 ; END

SWI_E:
; Display playing screen

C656: 0D 28              TST     <fainting       ; Fainting?
C658: 26 05              BNE     $C65F           ; Yes, skip this
C65A: 8D 04              BSR     $C660           ; Refresh display
C65C: 0A B4              DEC     <flipScreens    ;
C65E: 13                 SYNC                    ; Wait on display
C65F: 39                 RTS                     ; Out
;
C660: 34 76              PSHS    U,Y,X,B,A       ; Hold these
C662: DC 26              LDD     <m0226          ; Ambient light level
```

```
C664: DE 24              LDU     <torchPtr        ; Torch pointer
C666: 27 04              BEQ     $C66C            ; No torch lit, go with ambient level
C668: AB 47              ADDA    7,U              ; Add ambient...
C66A: EB 48              ADDB    8,U              ; ... to torch's power
C66C: DD 6E              STD     <m026E           ; Light level to display things with
C66E: AD 9F 02 B2        JSR     [displayFunction]; Refresh screen
C672: 35 F6              PULS    A,B,X,Y,U,PC     ; Done

SWI_F:
; Ready command prompt
C674: 8E C6 7A           LDX     #$C67A           ; Prompt CR and cursor
C677: 3F                 SWI                      ; Print the prompt and back up over cursor
C678: 03                                          ; SWI_3:Display uncompressed message pointed to
C679: 39                 RTS                      ; Done

C67A: 1F 1E 1C 24 FF     ; CR "." "_" BACK END

SWI_10:
; Pause for 1.35 seconds
C67F: C6 51              LDB     #$51             ; 81 / 60 = 1.35 seconds
C681: 13                 SYNC                     ; Wait for 60Hz interrupt
C682: 5A                 DECB                     ; Wait ...
C683: 26 FC              BNE     $C681            ; ... for all interrupts
C685: 39                 RTS                      ; Done

SWI_11:
; Fill X to U with 0s
C686: 4F                 CLRA                     ; Fill with 0's (black background)
C687: 8C                 ; CMPX  opcode to skip next instruction
SWI_12:
; Fill X to U with FFs
C688: 86 FF              LDA     #$FF             ; Fill with FF's (white background)
C68A: A7 80              STA     ,X+              ; Clear the ...
C68C: AC 6A              CMPX    10,S             ; ... entire ... (calling U)
C68E: 26 FA              BNE     $C68A            ; ... buffer
C690: 39                 RTS                      ; Done

; Execute a BASIC ROM routine
; This is never used. Instead the code uses SWI2, which is structured identically but
; designed for an RTI instead of RTS.
C691: 5F                 CLRB                     ; Set DP to ...
```

```
C692: 1F 9B              TFR     B,DP            ; ... zero for BASIC
C694: EE 6C              LDU     12,S            ; Read immediate ...
C696: E6 C0              LDB     ,U+             ; ... byte from code ...
C698: EF 6C              STU     12,S            ; Update the return
C69A: CE A0 00           LDU     #$A000          ; Execute the ...
C69D: AD D5              JSR     [B,U]           ; ... ROM routine
C69F: A7 63              STA     3,S             ; Return A
C6A1: AF 66              STX     6,S             ; Return X
C6A3: 39                 RTS                     ; Done

SWI_13:
; Beam on picture pointed to by X
C6A4: 0F B1              CLR     <hearHeart      ;
C6A6: 3F                 SWI
C6A7: 0A                                         ; SWI_A:Clear hand descriptor:

SWI_14:
; Beam subroutine
C6A8: 3F                 SWI
C6A9: 0B                                         ; SWI_B:Clear play field:
C6AA: CC 80 80           LDD     #$8080
C6AD: DD 4F              STD     <m024F          ;
C6AF: D6 9E              LDB     <m029E          ;
C6B1: 27 04              BEQ     $C6B7           ;
C6B3: C6 20              LDB     #$20
C6B5: 0A 9C              DEC     <beamSound      ;
C6B7: 8D 1E              BSR     $C6D7           ;
C6B9: 5A                 DECB
C6BA: 5A                 DECB
C6BB: 2A FA              BPL     $C6B7           ;
C6BD: 0F 9C              CLR     <beamSound      ;
C6BF: 0F 9E              CLR     <m029E          ;
C6C1: 3F                 SWI
C6C2: 1B                                         ; SWI_1B:Play sound i at full volume:
C6C3: 16                                         ; Sound 16 = Wizard strike
C6C4: 39                 RTS

SWI_15:
; Beam subroutine
C6C5: 3F                 SWI
C6C6: 0B                                         ; SWI_B:Clear play field:
```

```
C6C7: 8D F8              BSR     $C6C1           ;
C6C9: 5F                 CLRB
C6CA: 0A 9C              DEC     <beamSound      ;
C6CC: 8D 09              BSR     $C6D7           ;
C6CE: 5C                 INCB
C6CF: 5C                 INCB
C6D0: C1 20              CMPB    #$20
C6D2: 26 F8              BNE     $C6CC           ;
C6D4: 0F 9C              CLR     <beamSound      ;
C6D6: 39                 RTS
C6D7: 34 50              PSHS    U,X
C6D9: D7 2D              STB     <dotFrequency   ;
C6DB: D7 9D              STB     <beamSoundVal   ;
C6DD: 3F                 SWI
C6DE: 09                                         ; SWI_9:Clear secondary screen:
C6DF: 3F                 SWI
C6E0: 01                                         ; SWI_1:Draw picture X on screen:
C6E1: 0A B4              DEC     <flipScreens    ;
C6E3: 13                 SYNC
C6E4: 35 D0              PULS    X,U,PC

SWI_16:
; Print PREPARE
C6E6: BD D4 89           JSR     SetForExamine   ;
C6E9: CC 01 2C           LDD     #$012C
C6EC: ED 44              STD     4,U
C6EE: 3F                 SWI
C6EF: 02                                         ; SWI_2:Uncompress message m and display:
;
C6F0: 3C 24 58 06 45 D8 ; "PREPARE!"
;
C6F6: 0F B7              CLR     <whereToPrint   ; Printing goes to command area now
C6F8: 0A B4              DEC     <flipScreens    ;
C6FA: 39                 RTS

SWI_17:
; Create object structure
;
; The object structure is filled out with all the class information for
; the object. If the object is a torch, shield, or sword then the base
; class info is used instead (pine, leather, wooden). When the object
```

```
; is revealed then it takes its own properties.
;
; A = type
; B = maze level
; Return X = pointer to object
;
C6FB: DE 0F             LDU     <nextObjSlot    ; Current object pointer
C6FD: EF 66             STU     6,S             ; Return this
C6FF: 30 4E             LEAX    14,U            ; Point to ...
C701: 9F 0F             STX     <nextObjSlot    ; ... next object
C703: A7 49             STA     9,U             ; Object type
C705: E7 44             STB     4,U             ; Maze level
C707: 3F                SWI                     ; Fill out object ...
C708: 18                                        ; SWI_18:Change object to proper name and data:
C709: E6 4A             LDB     10,U            ; Object class
C70B: 8E C7 19          LDX     #$C719          ; Base type table
C70E: A6 85             LDA     B,X             ; Get the basic model for this class
C710: 2B 06             BMI     $C718           ; There is no basic ... skip
C712: E6 4B             LDB     11,U            ; Preserve needed-to-reveal
C714: 3F                SWI                     ; Copy over the basic ...
C715: 18                                        ; SWI_18:Change object to proper name and data:
C716: E7 4B             STB     11,U            ; Preserve the needed-to-reveal
C718: 39                RTS                     ; Done

; Basic type for each class (if applicable). This is used when the object is
; not revealed. Thus unrevealed IRON SWORD acts like the basic WOODEN SWORD.
C719: FF  ; Flask   no basic type
C71A: FF  ; Ring    no basic type
C71B: FF  ; Scroll  no basic type
C71C: 10  ; Shield  (10 = LEATHER SHIELD)
C71D: 11  ; Sword   (11 = WOODEN SWORD)
C71E: 0F  ; Torch   (0F = PINE TORCH)

SWI_18:
; Change object to proper name and data
; Change object to proper name and data
; A = object type
C71F: 48                ASLA                    ; Type times ...
C720: 48                ASLA                    ; ... 4 bytes per entry
C721: 8E DA 00          LDX     #$DA00          ; Object descriptors
C724: 31 86             LEAY    A,X             ; Get the object data
```

```
C726: 30 4A             LEAX    10,U            ; Destination in structure
C728: 86 04             LDA     #$04            ; Four bytes
C72A: BD C0 4B          JSR     CopyYtoX        ; Copy 4 bytes from Y to X
C72D: 8E DA 60          LDX     #$DA60          ; Special object properties (backed up one slot)
C730: 30 04             LEAX    4,X             ; Point to next object
C732: A6 84             LDA     ,X              ; Object's type
C734: 2B 0C             BMI     $C742           ; End of list ... out
C736: A1 63             CMPA    3,S             ; Is this special data for us?
C738: 26 F6             BNE     $C730           ; No ... keep looking
C73A: EC 01             LDD     1,X             ; Copy ...
C73C: ED 46             STD     6,U             ; ... 3 ...
C73E: A6 03             LDA     3,X             ; ... bytes of ...
C740: A7 48             STA     8,U             ; ... special data
C742: 39                RTS                     ; Done

SWI_19:
; Bring up normal display
C743: 3F                SWI
C744: 0A                                        ; SWI_A:Clear hand descriptor:
C745: 3F                SWI
C746: 0B                                        ; SWI_B:Clear play field:
C747: 3F                SWI
C748: 0C                                        ; SWI_C:Update heart rate:
C749: 0C AE             INC     <heartCounter   ;
C74B: 0A AD             DEC     <scrollShowing  ; Scroll is NOT showing
C74D: 0A B1             DEC     <hearHeart      ;
C74F: 3F                SWI
C750: 0D                                        ; SWI_D:Print contents of hands on status line:
; Fall into LOOK
```

LOOK command

```
CmdLOOK:
C751: 8E CE 66          LDX     #NormalDisplay  ; The routine for drawing ...
C754: 9F B2             STX     <displayFunction ; ... then normal game screen
C756: 3F                SWI                     ; Redraw the screen
C757: 0E                                        ; SWI_E:Display playing screen:
C758: 39                RTS                     ; Done
```

```
SWI_1A:
; Set up level
;
C759: 97 81             STA     <currentLevel    ; Current level
C75B: C6 0C             LDB     #$0C             ; 12 bytes each (one byte to count each type of
C75D: 3D                MUL                      ; Pointer to ...
C75E: C3 03 98          ADDD    #$0398           ; ... creature count on level
C761: DD 82             STD     <m0282           ; Hold pointer to creature count
;
C763: D6 81             LDB     <currentLevel    ; Current level
C765: 8E CF FD          LDX     #HolesAndLadders ; Table of holes and ladders
C768: 9F 86             STX     <currentHoles    ;
C76A: A6 80             LDA     ,X+
C76C: 2A FC             BPL     $C76A            ;
C76E: 5A                DECB
C76F: 2A F7             BPL     $C768            ;
C771: 8E 03 D4          LDX     #$03D4
C774: CE 05 F4          LDU     #$05F4
C777: 3F                SWI
C778: 11                                         ; SWI_11:Fill X to U with 0s:
C779: BD C0 53          JSR     InitTasks        ;
C77C: BD CC 9C          JSR     MakeMazeLevel    ;
;
C77F: DE 82             LDU     <m0282           ; Pointer to creature counts
C781: 86 0B             LDA     #$0B             ; Start with most powerful
C783: E6 C6             LDB     A,U              ; Get count of creature type in A
C785: 27 06             BEQ     $C78D            ; None to make ... skip
C787: BD CF A5          JSR     CreateCreature   ; Make a creature of type in A
C78A: 5A                DECB                     ; Make all ...
C78B: 26 FA             BNE     $C787            ; ... of that creature type
C78D: 4A                DECA                     ; Next creature type
C78E: 2A F3             BPL     $C783            ; Do all creature types
;
C790: CE 03 C3          LDU     #$03C3           ; (03D4 - 11) Start of monsters on this level
C793: 0F 91             CLR     <restartFind     ; Scan from start of objects
C795: BD CF 63          JSR     GetNextOject     ; Find next object on this level
C798: 27 1C             BEQ     $C7B6            ; No objects ... done
C79A: 6D 05             TST     5,X              ; Somebody already holding this object?
C79C: 2A F7             BPL     $C795            ; Yes ... leave it alone
C79E: 33 C8 11          LEAU    $11,U            ; Point to next monster on this level
C7A1: 11 83 05 F4       CMPU    #$05F4           ; At end of list?
```

```
C7A5: 25 03              BCS     $C7AA           ; No ... leave pointer
C7A7: CE 03 D4           LDU     #$03D4          ; Yes ... start back with 1st monster
C7AA: 6D 4C              TST     12,U            ; Is this monster alive?
C7AC: 27 F0              BEQ     $C79E           ; No ... go to next monster (?? there better be a
C7AE: EC 48              LDD     8,U             ; Chain ....
C7B0: AF 48              STX     8,U             ; ... object ...
C7B2: ED 84              STD     ,X              ; ... to monster
C7B4: 20 DF              BRA     $C795           ; Keep going
;
; Set the colors of the screen areas
C7B6: 96 81              LDA     <currentLevel   ;
C7B8: 84 01              ANDA    #$01            ; Just the lower bit
C7BA: 40                 NEGA                    ; 0->00000000, 1->11111111
C7BB: 97 2C              STA     <backgroundColor ; ? color
C7BD: B7 03 96           STA     comColor        ;
C7C0: B7 03 86           STA     examineColor    ;
C7C3: 43                 COMA                    ; Toggle color for hands
C7C4: B7 03 8E           STA     hndColor        ; Color of hands area
C7C7: 39                 RTS                     ; Done

SWI_1B:
; Play sound i at full volume
;
C7C8: AE 6C              LDX     12,S            ; Caller's PC
C7CA: A6 80              LDA     ,X+             ; Get the effect number
C7CC: AF 6C              STX     12,S            ; Restore the caller's PC
C7CE: C6 FF              LDB     #$FF            ; Full volume
;
SWI_1C:
; Play sound A at volume B
;
C7D0: D7 61              STB     <m0261          ; Store the volume
C7D2: 8E C7 DC           LDX     #$C7DC          ; Effect table
C7D5: 48                 ASLA                    ; Sound number to offset
C7D6: AD 96              JSR     [A,X]           ; Execute the sound routine
C7D8: 7F FF 20           CLR     PIA1_DA         ; All sound off
C7DB: 39                 RTS                     ; Done

SoundEffectsRoutines:
; Sound effects routine entry points
;
```

```
C7DC: C8 2B ; 00 Spider
C7DE: C8 50 ; 01 Snake
C7E0: C9 51 ; 02 Giant
C7E2: C8 3C ; 03 Blob
C7E4: C8 E2 ; 04 Knight
C7E6: C9 55 ; 05 Hatchet Giant
C7E8: C8 4A ; 06 Scorpion
C7EA: C8 DE ; 07 Shielded Knight
C7EC: C8 4D ; 08 Wraith
C7EE: C9 59 ; 09 Galdrog
C7F0: C8 77 ; 0A Demon
C7F2: C8 77 ; 0B Wizard
;
C7F4: C8 0A ; 0C Flask
C7F6: C8 11 ; 0D Ring
C7F8: C8 27 ; 0E Scroll
C7FA: C8 DA ; 0F Shield
C7FC: C8 A6 ; 10 Sword
C7FE: C8 B2 ; 11 Torch
;
C800: C9 3F ; 12 Player hit
C802: C8 E6 ; 13 Wizard beam
C804: C8 72 ; 14 Wall hit
C806: C8 6D ; 15 Creature dying
C808: C8 8A ; 16 Wizard strike

SoundFlask:
C80A: CE C8 23           LDU     #$C823
C80D: 86 04              LDA     #$04
C80F: 20 05              BRA     $C816           ;

SoundRing:
C811: CE C8 1F           LDU     #$C81F
C814: 86 0A              LDA     #$0A
C816: 97 5F              STA     <m025F          ;
C818: AD C4              JSR     ,U              ;
C81A: 0A 5F              DEC     <m025F          ;
C81C: 26 FA              BNE     $C818           ;
C81E: 39                 RTS

C81F: 8E 00 40           LDX     #$0040
```

```
C822: 10                ;CMPY opcode to skip next instruction
C823: 8E 00 80          LDX     #$0080
C826: 10                ;CMPY opcode to skip next instruction

SoundScroll:
C827: 8E 01 00          LDX     #$0100
C82A: 10                ;CMPY opcode to skip next instruction

SoundSpider:
C82B: 8E 00 20          LDX     #$0020
C82E: 8D 05             BSR     $C835           ;
C830: 30 1F             LEAX    -1,X
C832: 26 FA             BNE     $C82E           ;
C834: 39                RTS

C835: 86 FF             LDA     #$FF
C837: 8D 30             BSR     $C869           ;
C839: 4F                CLRA
C83A: 20 2D             BRA     $C869           ;

SoundBlob:
C83C: 8E 05 00          LDX     #$0500
C83F: 8D F4             BSR     $C835           ;
C841: 30 88 30          LEAX    $30,X
C844: 8C 08 00          CMPX    #$0800
C847: 25 F6             BCS     $C83F           ;
C849: 39                RTS

SoundScorpion:
C84A: 86 02             LDA     #$02
C84C: 8C                ; CMPX  opcode to skip next instruction

SoundWraith:
C84D: 86 01             LDA     #$01
C84F: 8C                ; CMPX  opcode to skip next instruction

SoundSnake:
C850: 86 0A             LDA     #$0A
C852: 97 62             STA     <m0262          ;
C854: 10 8E 00 C0       LDY     #$00C0
C858: 8D 74             BSR     $C8CE           ;
```

```
C85A: 8D 69             BSR     $C8C5           ;
C85C: 31 3F             LEAY    -1,Y
C85E: 26 F8             BNE     $C858           ;
C860: 8D 58             BSR     $C8BA           ;
C862: 0A 62             DEC     <m0262          ;
C864: 26 EE             BNE     $C854           ;
C866: 39                RTS
C867: 8D 65             BSR     $C8CE           ;
C869: 8D 5A             BSR     $C8C5           ;
C86B: 20 50             BRA     $C8BD           ;

SoundMonsDeath:
C86D: CE DB DA          LDU     #$DBDA
C870: 20 21             BRA     $C893           ;

SoundWallHit:
C872: CE DB D2          LDU     #$DBD2
C875: 20 1C             BRA     $C893           ;

SoundWizard:
SoundDemon:
C877: 86 08             LDA     #$08
C879: 97 5F             STA     <m025F          ;
C87B: 8D 51             BSR     $C8CE           ;
C87D: 4F                CLRA
C87E: 54                LSRB
C87F: 26 01             BNE     $C882           ;
C881: 5C                INCB
C882: 1F 01             TFR     D,X
C884: 8D A8             BSR     $C82E           ;
C886: 0A 5F             DEC     <m025F          ;
C888: 26 F1             BNE     $C87B           ;

SoundWizStrike:
C88A: CE DB D2          LDU     #$DBD2
C88D: 8D 04             BSR     $C893           ;
C88F: 8D 29             BSR     $C8BA           ;
C891: 33 44             LEAU    4,U
C893: AE C4             LDX     ,U
C895: 10 AE 42          LDY     2,U
C898: 8D CD             BSR     $C867           ;
```

```
C89A: 31 3F             LEAY    -1,Y
C89C: 26 FA             BNE     $C898           ;
C89E: 30 02             LEAX    2,X
C8A0: 8C 01 50          CMPX    #$0150
C8A3: 26 F0             BNE     $C895           ;
C8A5: 39                RTS

SoundSword:
C8A6: BD C9 31          JSR     $C931           ;
C8A9: 80 ; Consumed by routine
C8AA: 8D 76             BSR     $C922
C8AC: 25 04             BCS     SoundTorch      ;
C8AE: 8D 15             BSR     $C8C5           ;
C8B0: 20 F8             BRA     $C8AA           ;

SoundTorch:
C8B2: BD C9 2E          JSR     $C92E           ;
C8B5: A0 ; Consumed by routine
C8B6: 8D 6E             BSR     $C926
C8B8: 20 FC             BRA     $C8B6
;
C8BA: 8E 10 00          LDX     #$1000
C8BD: 34 10             PSHS    X
C8BF: 30 1F             LEAX    -1,X
C8C1: 26 FC             BNE     $C8BF
C8C3: 35 90             PULS    X,PC

C8C5: D6 61             LDB     <m0261          ;
C8C7: 3D                MUL
C8C8: 84 FC             ANDA    #$FC
C8CA: B7 FF 20          STA     PIA1_DA         ;
C8CD: 39                RTS

C8CE: DC 56             LDD     <m0256          ;
C8D0: 58                ASLB
C8D1: 49                ROLA
C8D2: 58                ASLB
C8D3: 49                ROLA
C8D4: D3 56             ADDD    <m0256          ;
C8D6: 5C                INCB
C8D7: DD 56             STD     <m0256          ;
```

```
C8D9: 39                RTS

SoundShield:
C8DA: 8D 39             BSR     $C915           ;
C8DC: 64 24  ; Consumed by routine. Returns to the one who called this.

SoundShldKnight:
C8DE: 8D 35             BSR     $C915           ;
C8E0: 32 12  ; Consumed by routine. Returns to the one who called this.

SoundKnight:
C8E2: 8D 31             BSR     $C915           ;
C8E4: AF 36  ; Consumed by routine. Returns to the one who called this.

SoundWizBeam:
C8E6: 8D 2D             BSR     $C915           ;
C8E8: 19 09 ; Consumed by routine. Returns to the one who called this.

C8EA: 8D 42             BSR     $C92E
C8EC: 60    ; Consumed by routine
C8ED: 9E 63             LDX     <m0263
C8EF: 10 9E 65          LDY     <m0265
C8F2: 4F                CLRA
C8F3: 30 1F             LEAX    -1,X
C8F5: 26 06             BNE     $C8FD           ;
C8F7: 9E 63             LDX     <m0263          ;
C8F9: 88 7F             EORA    #$7F
C8FB: 8D 0D             BSR     $C90A           ;
C8FD: 31 3F             LEAY    -1,Y
C8FF: 26 F2             BNE     $C8F3           ;
C901: 10 9E 65          LDY     <m0265          ;
C904: 88 80             EORA    #$80
C906: 8D 02             BSR     $C90A           ;
C908: 20 E9             BRA     $C8F3           ;
C90A: 97 59             STA     <m0259          ;
C90C: 8D 70             BSR     $C97E           ;
C90E: 23 B3             BLS     $C8C3           ;
C910: 8D B3             BSR     $C8C5           ;
C912: 96 59             LDA     <m0259          ;
C914: 39                RTS
```

```
C915: AE E1             LDX     ,S++            ; Pull return from the stack (returning up a fra
C917: E6 80             LDB     ,X+             ; Get the immediate byte
C919: 4F                CLRA
C91A: DD 63             STD     <m0263          ;
C91C: E6 80             LDB     ,X+             ; Get the next byte
C91E: DD 65             STD     <m0265          ;
C920: 20 C8             BRA     $C8EA           ;

C922: 8D AA             BSR     $C8CE           ;
C924: 20 67             BRA     $C98D           ;
C926: 8D A6             BSR     $C8CE           ;
C928: 8D 54             BSR     $C97E           ;
C92A: 23 97             BLS     $C8C3           ;
C92C: 20 97             BRA     $C8C5           ;

C92E: 9E 03             LDX     <CONST_FF       ;
C930: 10                ;LDY opcode to skip next instruction
C931: 9E 00             LDX     <CONST_00       ;
C933: 9F 5B             STX     <m025B          ;
C935: AE E4             LDX     ,S              ; Return location
C937: E6 80             LDB     ,X+             ; Get the immediate byte
C939: 4F                CLRA
C93A: DD 5D             STD     <m025D          ;
C93C: AF E4             STX     ,S              ; Corrected return address
C93E: 39                RTS                     ; Done

SoundPlayerHit:
C93F: 8D ED             BSR     $C92E           ;
C941: 60 ; Consumed by routine
C942: BD C8 CE          JSR     $C8CE
C945: 44                LSRA
C946: 8D E0             BSR     $C928           ;
C948: BD C8 CE          JSR     $C8CE           ;
C94B: 8A 80             ORA     #$80
C94D: 8D D9             BSR     $C928           ;
C94F: 20 F1             BRA     $C942           ;

SoundGiant:
C951: 8E 03 00          LDX     #$0300
C954: 10                ;CMPY skip next instruction
```

```
SoundHchGiant:
C955: 8E 02 00          LDX     #$0200
C958: 10                ;CMPY skip next instruction

SoundGaldrog:
C959: 8E 01 00          LDX     #$0100
C95C: 9F 5D             STX     <m025D          ;
C95E: 4F                CLRA
C95F: 5F                CLRB
C960: DD 5B             STD     <m025B          ;
C962: 8D BE             BSR     $C922           ;
C964: 25 0B             BCS     $C971           ;
C966: BD C8 C5          JSR     $C8C5           ;
C969: 8E 00 F0          LDX     #$00F0
C96C: BD C8 BD          JSR     $C8BD           ;
C96F: 20 F1             BRA     $C962           ;
C971: 8D BB             BSR     $C92E           ;
C973: 40                NEGA
C974: 8D B0             BSR     $C926           ;
C976: 8E 00 60          LDX     #$0060
C979: BD C8 BD          JSR     $C8BD           ;
C97C: 20 F6             BRA     $C974           ;
C97E: 34 02             PSHS    A
C980: DC 5B             LDD     <m025B          ;
C982: 93 5D             SUBD    <m025D          ;
C984: 34 01             PSHS    CC
C986: DD 5B             STD     <m025B          ;
C988: E6 61             LDB     1,S
C98A: 3D                MUL
C98B: 35 85             PULS    CC,B,PC
C98D: 34 02             PSHS    A
C98F: DC 5B             LDD     <m025B          ;
C991: D3 5D             ADDD    <m025D          ;
C993: 20 EF             BRA     $C984           ;
```

SWI Function Table

SWI	Address	Function
00	C384	Light level
01	C3A2	Draw picture X on screen

02	C448	Uncompress message m and display
03	C454	Display uncompressed message pointed to by X
04	C459	Display a single character in A
05	C46F	Uncompress message X to buffer
06	C472	Uncompress message X to given buffer U
07	C4CF	Get random number
08	C4F3	Clear display screen
09	C4F6	Clear secondary screen
0A	C4FF	Clear hand descriptor
0B	C507	Clear play field
0C	C529	Update heart rate
0D	C5D9	Print contents of hands on status line
0E	C656	Display playing screen
0F	C674	Ready command prompt
10	C67F	Pause for 1.35 seconds
11	C686	Fill X to U with 0s
12	C688	Fill X to U with FFs
13	C6A4	Beam on picture pointed to by X
14	C6A8	Beam subroutine
15	C6C5	Beam subroutine
16	C6E6	Print PREPARE
17	C6FB	Create object structure
18	C71F	Change object to proper name and data
19	C743	Bring up normal display
1A	C759	Set up level
1B	C7C8	Play sound i at full volume
1C	C7D0	Play sound A at volume B

```
SWIOffsetTable:
;
C995: 00 ;  0: C384 SWI_0:Light level:
C996: 1E ;  1: C3A2 SWI_1:Draw picture X on screen:
C997: A6 ;  2: C448 SWI_2:Uncompress message m and display:
C998: 0C ;  3: C454 SWI_3:Display uncompressed message pointed to by X:
C999: 05 ;  4: C459 SWI_4:Display a single character in A:
C99A: 16 ;  5: C46F SWI_5:Uncompress message X to buffer:
C99B: 03 ;  6: C472 SWI_6:Uncompress message X to given buffer U:
C99C: 5D ;  7: C4CF SWI_7:Get random number:
C99D: 24 ;  8: C4F3 SWI_8:Clear display screen:
C99E: 03 ;  9: C4F6 SWI_9:Clear secondary screen:
C99F: 09 ;  A: C4FF SWI_A:Clear hand descriptor:
```

```
C9A0: 08 ;  B: C507 SWI_B:Clear play field:
C9A1: 22 ;  C: C529 SWI_C:Update heart rate:
C9A2: B0 ;  D: C5D9 SWI_D:Print contents of hands on status line:
C9A3: 7D ;  E: C656 SWI_E:Display playing screen:
C9A4: 1E ;  F: C674 SWI_F:Ready command prompt:
C9A5: 0B ; 10: C67F SWI_10:Pause for 1.35 seconds:
C9A6: 07 ; 11: C686 SWI_11:Fill X to U with 0s:
C9A7: 02 ; 12: C688 SWI_12:Fill X to U with FFs:
C9A8: 1C ; 13: C6A4 SWI_13:Beam on picture pointed to by X:
C9A9: 04 ; 14: C6A8 SWI_14:Beam subroutine:
C9AA: 1D ; 15: C6C5 SWI_15:Beam subroutine:
C9AB: 21 ; 16: C6E6 SWI_16:Print PREPARE:
C9AC: 15 ; 17: C6FB SWI_17:Create object structure:
C9AD: 24 ; 18: C71F SWI_18:Change object to proper name and data:
C9AE: 24 ; 19: C743 SWI_19:Bring up normal display:
C9AF: 16 ; 1A: C759 SWI_1A:Set up level:
C9B0: 6F ; 1B: C7C8 SWI_1B:Play sound i at full volume:
C9B1: 08 ; 1C: C7D0 SWI_1C:Play sound A at volume B:

PrintCharCRBS:
; Print character ... handle backspace and carriage return
; 0 - 1E = " ABCDEFGHIJKLMNOPQRSTUVWXYZ!_?."
; 1F = CR
; 20 = small heart (left)
; 21 = small heart (right)
; 22 = large heart (left)
; 23 = large heart (right)
; 24 = backspace
;
C9B2: 81 24             CMPA    #$24            ; Is it backspace?
C9B4: 27 09             BEQ     $C9BF           ; Yes ... go do a backspace
C9B6: 81 1F             CMPA    #$1F            ; Is it CR character?
C9B8: 27 10             BEQ     $C9CA           ; Yes ... go do a CR
C9BA: 8D 5B             BSR     PrintRegChar    ; Print the character
C9BC: 30 01             LEAX    1,X             ; Advance the cursor
C9BE: 39                RTS                     ; Done
;
; Backspace (wrap to end of area)
C9BF: 30 1F             LEAX    -1,X            ; Back cursor up one space
C9C1: 9C 03             CMPX    <CONST_FF       ; Did we underflow?
C9C3: 26 04             BNE     $C9C9           ; No ... keep it
```

```
C9C5: AE 42              LDX     2,U             ; Yes ... wrap ...
C9C7: 30 1F              LEAX    -1,X            ; ... to end of area
C9C9: 39                 RTS                     ; Done
;
; Carriage return
C9CA: 30 88 20           LEAX    $20,X           ; Drop to next row
C9CD: 1E 01              EXG     D,X             ; Mask row offset ...
C9CF: C4 E0              ANDB    #$E0            ; ... back to beginning ...
C9D1: 1E 01              EXG     D,X             ; ... of row
C9D3: 39                 RTS                     ; Done

ScrollTextArea:
; Scroll the text area pointed to by U. Return the new cursor offset.
C9D4: 34 36              PSHS    Y,X,B,A         ; Hold all
C9D6: AE C4              LDX     ,U              ; Start of area
C9D8: EC 42              LDD     2,U             ; Number of characters in the area
C9DA: 83 00 20           SUBD    #$0020          ; Back up 32 characters (one row)
C9DD: ED 62              STD     2,S             ; Return the new cursor offset
C9DF: 8D 2F              BSR     Dleft3          ; D = D * 8 (8 bytes per character)
C9E1: 1F 02              TFR     D,Y             ; Number of bytes to move in Y
C9E3: EC 89 01 00        LDD     $0100,X         ; Data from next text row (32 columns * 8 rows)
C9E7: 6D 47              TST     7,U             ; Mirroring into both screen buffers?
C9E9: 26 04              BNE     $C9EF           ; No ... skip the 2nd screen
C9EB: ED 89 18 00        STD     $1800,X         ; Yes ... store it to 2nd screen
C9EF: ED 81              STD     ,X++            ; Store to first screen
C9F1: 31 3E              LEAY    -2,Y            ; All pixels moved up?
C9F3: 26 EE              BNE     $C9E3           ; No ... keep scrolling
; Blank the new bottom line
C9F5: E6 46              LDB     6,U             ; Get color of area
C9F7: 1D                 SEX                     ; Make it a double
C9F8: 10 8E 01 00        LDY     #$0100          ; 32 columns * 8 rows
C9FC: 6D 47              TST     7,U             ; Mirroring to second screen?
C9FE: 26 04              BNE     $CA04           ; No ... only one screen
CA00: ED 89 18 00        STD     $1800,X         ; Mirror data to second screen
CA04: ED 81              STD     ,X++            ; Blank the area
CA06: 31 3E              LEAY    -2,Y            ; Entire row blanked?
CA08: 26 F2              BNE     $C9FC           ; No ... keep blanking
CA0A: 35 B6              PULS    A,B,X,Y,PC      ; Done

; Shift-D-to-the-left entry points
Dleft5:
```

```
CA0C: 58                ASLB                    ; D << 5 starts here
CA0D: 49                ROLA                    ;
Dleft4:
CA0E: 58                ASLB                    ; D << 4 starts here
CA0F: 49                ROLA                    ;
Dleft3:
CA10: 58                ASLB                    ; D << 3 starts here
CA11: 49                ROLA                    ;
Dleft2:
CA12: 58                ASLB                    ; D << 2 starts here
CA13: 49                ROLA                    ;
Dleft1:
CA14: 58                ASLB                    ; D << 1 starts here
CA15: 49                ROLA                    ;
;
CA16: 39                RTS                     ; Done

PrintRegChar:
; Print character image or heart image
CA17: 34 76             PSHS    U,Y,X,B,A       ; Save everything
CA19: 81 20             CMPA    #$20            ; Letter?
CA1B: 25 0C             BCS     $CA29           ; Yes ... go handle
CA1D: 80 20             SUBA    #$20            ; Must be heart picture
CA1F: C6 07             LDB     #$07            ; 7 bytes ...
CA21: 3D                MUL                     ; ... per heart picture
CA22: C3 DB B6          ADDD    #$DBB6          ; Offset ...
CA25: 1F 01             TFR     D,X             ; ... to bit mask
CA27: 20 1B             BRA     $CA44           ; Already have the pattern -- use it
; Uncompress the character pattern
CA29: C6 05             LDB     #$05            ; Five bytes in each character image
CA2B: 3D                MUL                     ; Pointer now in D
CA2C: C3 DB 1B          ADDD    #$DB1B          ; Offset into character image table
CA2F: 1F 01             TFR     D,X             ; To X
CA31: CE 03 57          LDU     #$0357          ; Expansion buffer
CA34: 3F                SWI                     ; Decompress the pattern
CA35: 06                                        ; SWI_6:Uncompress message X to given buffer U:
CA36: 8E 03 5E          LDX     #$035E          ; Shift ...
CA39: 68 82             ASL     ,-X             ; ... image two times to ...
CA3B: 68 84             ASL     ,X              ; ... middle of ...
CA3D: 8C 03 57          CMPX    #$0357          ; ... raster ...
CA40: 22 F7             BHI     $CA39           ; ... buffer
```

```
CA42: EE 66              LDU     6,S             ; Pointer to screen area
; Draw the raster image
CA44: EC 44              LDD     4,U             ; Current cursor
CA46: 8D C8              BSR     Dleft3          ; Offset = (cur/32)*256 + (cur%32). From yyyxxxx
CA48: 54                 LSRB                    ; ... y is number of whole rows (32*8 = 256 byte
CA49: 54                 LSRB                    ; ... then x is the ...
CA4A: 54                 LSRB                    ; ... offset on the final row. Clever.
CA4B: E3 C4              ADDD    ,U              ; Offset from beginning of area
CA4D: 1F 02              TFR     D,Y             ; To Y for indexing
CA4F: C6 07              LDB     #$07            ; 7 rows in an image
CA51: A6 80              LDA     ,X+             ; Get next row pattern
CA53: A8 46              EORA    6,U             ; Use the color from the area
CA55: A7 A4              STA     ,Y              ; Store the pattern on the screen
CA57: 6D 47              TST     7,U             ; Are we mirroring?
CA59: 26 04              BNE     $CA5F           ; No ... skip 2nd screen
CA5B: A7 A9 18 00        STA     $1800,Y         ; Pattern to 2nd screen
CA5F: 31 A8 20           LEAY    $20,Y           ; Next row
CA62: 5A                 DECB                    ; All rows done?
CA63: 26 EC              BNE     $CA51           ; No ... go do them all
CA65: 35 F6              PULS    A,B,X,Y,U,PC    ; Done

CA67: 34 16              PSHS    X,B,A
CA69: 6F E4              CLR     ,S
CA6B: 6F 61              CLR     1,S
CA6D: 0F C1              CLR     <holdHole       ;
CA6F: DD C2              STD     <m02C2          ;
CA71: 27 24              BEQ     $CA97           ;
CA73: 10 A3 62           CMPD    2,S
CA76: 26 04              BNE     $CA7C           ;
CA78: 6C E4              INC     ,S
CA7A: 20 1B              BRA     $CA97           ;
CA7C: 8E 00 10           LDX     #$0010
CA7F: 08 C3              LSL     <m02C3          ;
CA81: 09 C2              ROL     <m02C2          ;
CA83: 09 C1              ROL     <holdHole       ;
CA85: 68 61              ASL     1,S
CA87: 69 E4              ROL     ,S
CA89: DC C1              LDD     <holdHole       ;
CA8B: A3 62              SUBD    2,S
CA8D: 25 04              BCS     $CA93           ;
CA8F: DD C1              STD     <holdHole       ;
```

```
CA91: 6C 61             INC     1,S
CA93: 30 1F             LEAX    -1,X
CA95: 26 E8             BNE     $CA7F           ;
CA97: 35 96             PULS    A,B,X,PC

CA99: 43                COMA
CA9A: 53                COMB
CA9B: C3 00 01          ADDD    #$0001
CA9E: 39                RTS

CA9F: 34 16             PSHS    X,B,A
CAA1: 9E 43             LDX     <m0243          ;
CAA3: EC E4             LDD     ,S              ;
CAA5: 2A 07             BPL     $CAAE           ;
CAA7: 8D F0             BSR     $CA99           ;
CAA9: 8D BC             BSR     $CA67           ;
CAAB: 8D EC             BSR     $CA99           ;
CAAD: 8C                ; CMPX  opcode to skip next instruction
CAAE: 8D B7             BSR     $CA67
CAB0: ED E4             STD     ,S
CAB2: 35 96             PULS    A,B,X,PC

CAB4: 7E CB 8A          JMP     $CB8A           ;

CAB7: 34 76             PSHS    U,Y,X,B,A
CAB9: 0C 2D             INC     <dotFrequency   ;
CABB: 27 F7             BEQ     $CAB4           ;
CABD: 96 2D             LDA     <dotFrequency   ;
CABF: 97 2E             STA     <m022E          ;
CAC1: DC 35             LDD     <m0235          ;
CAC3: 93 31             SUBD    <m0231          ;
CAC5: DD 3E             STD     <m023E          ;
CAC7: 2A 02             BPL     $CACB           ;
CAC9: 8D CE             BSR     $CA99           ;
CACB: DD 43             STD     <m0243          ;
CACD: DC 33             LDD     <m0233          ;
CACF: 93 2F             SUBD    <m022F          ;
CAD1: DD 41             STD     <m0241          ;
CAD3: 2A 02             BPL     $CAD7           ;
CAD5: 8D C2             BSR     $CA99           ;
CAD7: 10 93 43          CMPD    <m0243          ;
```

```
CADA: 2D 04             BLT     $CAE0           ;
CADC: DD 43             STD     <m0243          ;
CADE: 27 D4             BEQ     $CAB4           ;
CAE0: DC 3E             LDD     <m023E          ;
CAE2: 8D BB             BSR     $CA9F           ;
CAE4: DD 3E             STD     <m023E          ;
CAE6: 1F 89             TFR     A,B
CAE8: 1D                SEX
CAE9: C6 01             LDB     #$01
CAEB: 97 3D             STA     <m023D          ;
CAED: 2A 01             BPL     $CAF0           ;
CAEF: 50                NEGB
CAF0: D7 45             STB     <m0245          ;
CAF2: DC 41             LDD     <m0241          ;
CAF4: 8D A9             BSR     $CA9F           ;
CAF6: DD 41             STD     <m0241          ;
CAF8: 1F 89             TFR     A,B
CAFA: 1D                SEX
CAFB: C6 20             LDB     #$20
CAFD: 97 40             STA     <m0240          ;
CAFF: 2A 01             BPL     $CB02           ;
CB01: 50                NEGB
CB02: D7 46             STB     <m0246          ;
CB04: DC 31             LDD     <m0231          ;
CB06: DD 37             STD     <m0237          ;
CB08: DC 2F             LDD     <m022F          ;
CB0A: DD 3A             STD     <m023A          ;
CB0C: 86 80             LDA     #$80
CB0E: 97 39             STA     <m0239          ;
CB10: 97 3C             STA     <m023C          ;
CB12: AE 42             LDX     2,U
CB14: 9F 49             STX     <m0249          ;
CB16: AE C4             LDX     ,U
CB18: 9F 47             STX     <m0247          ;
CB1A: DC 3A             LDD     <m023A          ;
CB1C: BD CA 0C          JSR     Dleft5          ;
CB1F: 30 8B             LEAX    D,X
CB21: DC 37             LDD     <m0237          ;
CB23: BD D3 7F          JSR     DRight3         ;
CB26: 30 8B             LEAX    D,X
CB28: CE CB 8E          LDU     #$CB8E          ; Bit table (80,40,20,10,08,04,02,01)
```

```
CB2B: 10 9E 43           LDY     <m0243          ;
CB2E: 0A 2E              DEC     <m022E          ;
CB30: 26 22              BNE     $CB54           ;
CB32: 96 2D              LDA     <dotFrequency   ;
CB34: 97 2E              STA     <m022E          ;
CB36: 0D 37              TST     <m0237          ;
CB38: 26 1A              BNE     $CB54           ;
CB3A: 9C 47              CMPX    <m0247          ;
CB3C: 25 16              BCS     $CB54           ;
CB3E: 9C 49              CMPX    <m0249          ;
CB40: 24 12              BCC     $CB54           ;
CB42: D6 38              LDB     <m0238          ;
CB44: C4 07              ANDB    #$07
CB46: A6 C5              LDA     B,U             ; A = 2^b
CB48: 0D 2C              TST     <backgroundColor ;
CB4A: 27 04              BEQ     $CB50           ;
CB4C: 43                 COMA
CB4D: A4 84              ANDA    ,X
CB4F: 8C                 ; CMPX  opcode to skip next instruction
CB50: AA 84              ORA     ,X
CB52: A7 84              STA     ,X
CB54: 96 38              LDA     <m0238          ;
CB56: 84 F8              ANDA    #$F8
CB58: 97 C1              STA     <holdHole       ;
CB5A: DC 38              LDD     <m0238          ;
CB5C: D3 3E              ADDD    <m023E          ;
CB5E: DD 38              STD     <m0238          ;
CB60: D6 37              LDB     <m0237          ;
CB62: D9 3D              ADCB    <m023D          ;
CB64: D7 37              STB     <m0237          ;
CB66: 84 F8              ANDA    #$F8
CB68: 91 C1              CMPA    <holdHole       ;
CB6A: 27 04              BEQ     $CB70           ;
CB6C: D6 45              LDB     <m0245          ;
CB6E: 30 85              LEAX    B,X
CB70: DC 3B              LDD     <m023B          ;
CB72: 97 C1              STA     <holdHole       ;
CB74: D3 41              ADDD    <m0241          ;
CB76: DD 3B              STD     <m023B          ;
CB78: D6 3A              LDB     <m023A          ;
CB7A: D9 40              ADCB    <m0240          ;
```

```
CB7C: D7 3A              STB     <m023A          ;
CB7E: 91 C1              CMPA    <holdHole       ;
CB80: 27 04              BEQ     $CB86           ;
CB82: D6 46              LDB     <m0246          ;
CB84: 30 85              LEAX    B,X
CB86: 31 3F              LEAY    -1,Y
CB88: 26 A4              BNE     $CB2E           ;
CB8A: 0A 2D              DEC     <dotFrequency   ;
CB8C: 35 F6              PULS    A,B,X,Y,U,PC

BitNumbers:
; Left to right
CB8E: 80 40 20 10 08 04 02 01

GetNextWord:
CB96: 34 52              PSHS    U,X,A           ; Hold these
CB98: 9E 11              LDX     <m0211          ; Unparsed user input
CB9A: CE 03 13           LDU     #$0313          ; Decode buffer
CB9D: A6 80              LDA     ,X+             ; Next character
CB9F: 27 FC              BEQ     $CB9D           ; Skip to ...
CBA1: 20 02              BRA     $CBA5           ; ... first non-space
CBA3: A6 80              LDA     ,X+
CBA5: 2F 08              BLE     $CBAF           ; End of the input line
CBA7: A7 C0              STA     ,U+             ; Store the non-space character
CBA9: 11 83 03 33        CMPU    #$0333          ; Only 32 characters input ...
CBAD: 25 F4              BCS     $CBA3           ; ... allowed
CBAF: 86 FF              LDA     #$FF            ; Mark the end ...
CBB1: A7 C0              STA     ,U+             ; ... of the input
CBB3: 9F 11              STX     <m0211          ;
CBB5: 7D 03 13           TST     tmpBuffer1      ; Is there anything?
CBB8: 35 D2              PULS    A,X,U,PC        ; Done

CBBA: 0F 90              CLR     <m0290          ;
CBBC: 8E D9 6A           LDX     #ClassNames     ; Class names
CBBF: 8D 2B              BSR     DecodeInput     ;
CBC1: 2B 05              BMI     $CBC8           ;
CBC3: 27 1A              BEQ     $CBDF           ;
CBC5: DD 8E              STD     <holdIncantWord ;
CBC7: 39                 RTS
;
CBC8: 0A 90              DEC     <m0290          ;
```

```
CBCA: 8E D8 F3           LDX     #ProperNames     ; Proper names
CBCD: 8D 18              BSR     $CBE7            ;
CBCF: 2F 0E              BLE     $CBDF            ;
CBD1: DD 8E              STD     <holdIncantWord  ;
CBD3: 8E D9 6A           LDX     #ClassNames      ; Class names
CBD6: 8D 14              BSR     DecodeInput      ;
CBD8: 2F 05              BLE     $CBDF            ;
CBDA: D1 8F              CMPB    <holdIncantLen   ;
CBDC: 26 01              BNE     $CBDF            ;
CBDE: 39                 RTS
CBDF: 32 62              LEAS    2,S              ; Skip a stack frame to return error
CBE1: 3F                 SWI                      ; Print "???"
CBE2: 02                                          ; SWI_2:Uncompress message m and display:
CBE3: 17 7B D0 ; "???"
CBE6: 39                 RTS                      ; Done

CBE7: 34 76              PSHS    U,Y,X,B,A
CBE9: 4F                 CLRA
CBEA: 20 08              BRA     $CBF4            ;

DecodeInput:
; Find the next input word in the given table of words.
; X=pointer to word table
; $313 is the input word
; Return Z=1 if no input
; Return Z=0 if input but no match
; ?? Need to look at the flags more ... callers do BLE
; Return A=word number, FF=no match (or multiple matches)
; Return B=length of command word
; Return $7B=FF if exact match, 0 if not (important to incantation)
CBEC: 34 76              PSHS    U,Y,X,B,A        ; Hold these
CBEE: 4F                 CLRA                     ; Word number to return
CBEF: 5F                 CLRB                     ; Command word length
CBF0: 8D A4              BSR     GetNextWord      ; Get the next user input word
CBF2: 2B 39              BMI     $CC2D            ; Nothing on the input. Return nothing.
CBF4: 0F 78              CLR     <foundMatch      ; Found-a-match flag
CBF6: 0F 7B              CLR     <perfectMatch    ; Perfect input match
CBF8: E6 80              LDB     ,X+              ; Get the number ...
CBFA: D7 79              STB     <numWords        ; ... of words in table
CBFC: CE 03 13           LDU     #$0313           ; Start of typed word
CBFF: 3F                 SWI                      ; Uncompress the command word
```

```
CC00: 05                                        ; SWI_5:Uncompress message X to buffer:
CC01: 10 8E 03 36       LDY     #$0336          ; 335 is the length, 336 starts the text
CC05: E6 C0             LDB     ,U+             ; Char from uncompressed
CC07: 2B 0E             BMI     $CC17           ; We reached the end of the word ... a match
CC09: E1 A0             CMPB    ,Y+             ; User input match the command word?
CC0B: 26 15             BNE     $CC22           ; No ...
CC0D: 6D A4             TST     ,Y              ; More in the user buffer?
CC0F: 2A F4             BPL     $CC05           ; Yes ... keep checking against the command word
CC11: 6D C4             TST     ,U              ; No more. Did we check all of the command word?
CC13: 2A 0D             BPL     $CC22           ; Yes ... we have a match
CC15: 0A 7B             DEC     <perfectMatch   ; FF means there was an exact match
CC17: 0D 78             TST     <foundMatch     ; Do we already have a match?
CC19: 26 10             BNE     $CC2B           ; Error ... this could match multiple words
CC1B: 0C 78             INC     <foundMatch     ; Now we have a match
CC1D: F6 03 35          LDB     tmpBuffer2      ; Length of command word
CC20: ED E4             STD     ,S              ; Store potential match on the stack to return
CC22: 4C                INCA                    ; Word number for next test word
CC23: 0A 79             DEC     <numWords       ; Tried all words?
CC25: 26 D5             BNE     $CBFC           ; No ... keep looking
CC27: 0D 78             TST     <foundMatch     ; Did we find a match?
CC29: 26 04             BNE     $CC2F           ; Yes ... leave it as it on the stack
CC2B: DC 03             LDD     <CONST_FF       ;
CC2D: ED E4             STD     ,S              ; No matches ... store error to return in D
CC2F: 35 F6             PULS    A,B,X,Y,U,PC    ; Done

GetUserHand:
; The second word is LEFT or RIGHT. Return the pointer to the pointer in U and the
; pointer to the object in X.
CC31: 8E D8 D9          LDX     #SecondWords    ; Second words
CC34: 8D B6             BSR     DecodeInput     ; Decode the input word
CC36: 2F A7             BLE     $CBDF           ; Didn't get a match ... print "???" and abort
CC38: CE 02 1F          LDU     #rightHand      ; Pointer to right hand
CC3B: 81 01             CMPA    #$01            ; Word was "RIGHT" ?
CC3D: 27 07             BEQ     $CC46           ; Yes. Return slot and object
CC3F: CE 02 1D          LDU     #leftHand       ; Pointer to left hand
CC42: 81 00             CMPA    #$00            ; Word was "LEFT" ?
CC44: 26 99             BNE     $CBDF           ; No ... error and abort
CC46: AE C4             LDX     ,U              ; Yes. Return slot and object
CC48: 39                RTS                     ; Done

GetNeighborCells:
```

```
; This function gets the cell values for the 8 cells surrounding a given cell (and the
; value of the center square as well). If a cell is on the edge of the map then the
; invalid neighbor value is FF ... just as if it were solid.
;
; Param A,B: The Y,X coordinate of the center cell
; Param U: The 9-byte buffer to store results in
;
; The cell values are stored in the U buffer in this order from the maze map.
; 0 1 2
; 3 4 5
; 6 7 8
;
CC49: 34 56             PSHS    U,X,B,A         ; Preserve registers
CC4B: 4A                DECA                    ; Y-1
CC4C: 8D 08             BSR     $CC56           ; Check row (Y-1,*)
CC4E: 4C                INCA                    ; Back to center
CC4F: 8D 05             BSR     $CC56           ; Check row (Y,*)
CC51: 4C                INCA                    ; Y+1
CC52: 8D 02             BSR     $CC56           ; Check row (Y+1,*)
CC54: 35 D6             PULS    A,B,X,U,PC      ; Restore and out
;
CC56: 34 06             PSHS    B,A             ; Hold Y,X coordinates
CC58: 5A                DECB                    ; X-1
CC59: 8D 05             BSR     $CC60           ; Check Y,X-1
CC5B: 5C                INCB                    ; Back to center
CC5C: 8D 02             BSR     $CC60           ; Check Y,X
CC5E: 5C                INCB                    ; X+1
CC5F: 8C                ; CMPX  opcode to skip next instruction
CC60: 34 06             PSHS    B,A
CC62: 8D 2A             BSR     IsValidCell     ; Is the cell valid?
CC64: 26 05             BNE     $CC6B           ; No ... Store FF to cell buffer
CC66: 8D 13             BSR     GetCellPointer  ; Get the pointer to the cell
CC68: A6 84             LDA     ,X              ; Get the value from the maze
CC6A: 8C                ; CMPX  opcode to skip next instruction
CC6B: 86 FF             LDA     #$FF
CC6D: A7 C0             STA     ,U+             ; Store the result in the buffer
CC6F: 35 86             PULS    A,B,PC          ; Done

GetRandomCell:
; This function returns a random X,Y coordinate in the maze
; and the pointer to the cell memory in X.
```

```
;
; Return A: 0-31 (Y coordinate)
; Return B: 0-31 (X coordinate)
; Return X: Pointer to cell in memory
;
CC71: 3F                SWI                     ; Get random number in A
CC72: 07                                        ; SWI_7:Get random number:
CC73: 84 1F             ANDA    #$1F            ; 0-31 ... X coordinate
CC75: 1F 89             TFR     A,B             ; Coordinate to B
CC77: 3F                SWI                     ; Get random number in A
CC78: 07                                        ; SWI_7:Get random number:
CC79: 84 1F             ANDA    #$1F            ; 0-31 ... Y coordinate
;
GetCellPointer:
; Return the pointer to cell memory in X for the Y,X coordinate in A,B
;
CC7B: 34 06             PSHS    B,A             ; Hold these
CC7D: 84 1F             ANDA    #$1F            ; Limit coordiantes ...
CC7F: C4 1F             ANDB    #$1F            ; ... (another entry point)
CC81: 1F 01             TFR     D,X             ; Hold in X
CC83: C6 20             LDB     #$20            ; Multiply A (Y coordinate) ...
CC85: 3D                MUL                     ; ... by 32 (one row)
CC86: C3 05 F4          ADDD    #$05F4          ; Offset to maze
CC89: 1E 01             EXG     D,X             ; X coordinate back to B
CC8B: 3A                ABX                     ; Add in the X coordinate
CC8C: 35 86             PULS    A,B,PC          ; Return coordinate and pointer

IsValidCell:
; This function tests a given X,Y coordinate and returns
; Z=1 if within (31,31) or Z=0 if the coordinate is out
; of bounds.
;
; Param A: Y coordinate
; Param B: X coordinate
;
; Return Z: 1 if OK or 0 if out-of-bounds
;
CC8E: 34 06             PSHS    B,A             ; Push the coordinates on the stack
CC90: 84 1F             ANDA    #$1F            ; Mask the coordinate to within range
CC92: A1 E4             CMPA    ,S              ; Is the coordinate within range?
CC94: 26 04             BNE     $CC9A           ; No ... out
```

```
CC96: C4 1F             ANDB    #$1F            ; Mask the coordinate to within range
CC98: E1 61             CMPB    1,S             ; Is the coordinate within range? (return Z=1 if
CC9A: 35 86             PULS    A,B,PC          ; Return with Z=1 if OK or Z=0 if out of range

MakeMazeLevel:
; The maze is a 32x32 cell (one byte per cell) table at $05F4. Each cell has 4 2-bit fields that
; describe the wall in a given direction. A value of 00 means the wall in that direction is open.
; A value of 01 is a normal door in that direction. A value of 10 is a magic door in that direction
; A value of 11 is a solid wall. The 4 fields are stored in the byte as: LL_DD_RR_UU with UU being
; the least significant 2 bits.
;
; The maze is generated by carving out a series of random "runs". The code picks a random starting
; cell and a random direction. It picks a random "number of crossings" for the run from 1 to 8.
; Then it starts opening cells in that direction one by one until one of the following occurs:
; - The run crosses the randomly chosen number of other runs
; - The run reaches the edge of the map
; - The cell would create a block of 4 adjacent open cells
;
; The 3D display during game play can only draw hallways. Four adjacent open cells would create an
; open space that the display can't handle. Thus the check in the run algorithm.
;
; The algorithm keeps count of each new open cell created. Runs are generated until exactly 500 cel
; have been opened. Each level has exactly 500 open cells in it.
;
; Once the 500 cells are open the code adds exactly 70 regular doors and 45 magic doors between adj
; cells. Both cells get a copy of the door in opposite directions.
;
; It is possible (though unlikely) to create a run that does not overlap another. This would be an
; unreachable area that would trap the player or required monsters. Each level is drawn with a pre-
; random number seed. Thus the level is always the same, and the designers chose seeds that produce
; mazes.
;
; Holes and ladders are manually defined for each level and are kept in a separate table.
;
; Maze value: LL_DD_RR_UU
; 00 = open
; 01 = normal door
; 10 = magic door
; 11 = blocked
;
CC9C: 8E 05 F4          LDX     #$05F4          ; Start of level
```

```
CC9F: CE 09 F4           LDU     #$09F4          ; One past end of level (32*32=1024 byte)
CCA2: 3F                 SWI                     ; Fill the buffer with FFs
CCA3: 12                                         ; SWI_12:Fill X to U with FFs:
CCA4: 8E CD 9F           LDX     #$CD9F          ; Random number seeds
CCA7: D6 81              LDB     <currentLevel   ; Offset into seeds ...
CCA9: 3A                 ABX                     ; ... for this level
CCAA: EC 81              LDD     ,X++            ; Copy the ...
CCAC: DD 6B              STD     <rndSeedA       ; ... 3 byte ...
CCAE: A6 84              LDA     ,X              ; ... seed to ...
CCB0: 97 6D              STA     <rndSeedC       ; ... current seed
CCB2: 10 8E 01 F4        LDY     #$01F4          ; Make 500 cells in the "run" process
CCB6: BD CC 71           JSR     GetRandomCell   ; Get a random coordinate
CCB9: DD 7C              STD     <drwMazeY       ; Hold the starting point
;
; Start a new maze "run" of cells
CCBB: 3F                 SWI                     ; Get a random number
CCBC: 07                                         ; SWI_7:Get random number:
CCBD: 84 03              ANDA    #$03            ; Now a random direction (0-3)
CCBF: 97 8A              STA     <drwMazeDir     ; Hold current direction
CCC1: 3F                 SWI                     ; Get a random number
CCC2: 07                                         ; SWI_7:Get random number:
CCC3: 84 07              ANDA    #$07            ; Random 0..7
CCC5: 4C                 INCA                    ; Random 1..8
CCC6: 97 7E              STA     <drwMazeCross   ; Store number of crossings
CCC8: 20 08              BRA     $CCD2           ; Start this run with a step

CCCA: DC 88              LDD     <drwMazeTmp     ; Get the potential new coordinate
CCCC: DD 7C              STD     <drwMazeY       ; Make it the new cell
CCCE: 0A 7E              DEC     <drwMazeCross   ; All crossings in this run placed?
CCD0: 27 E9              BEQ     $CCBB           ; Yes ... start a new run (done with this one)
;
CCD2: DC 7C              LDD     <drwMazeY       ; Get the current cell pointer
CCD4: BD D1 1B           JSR     StepInDirection ; Move in the random direction
CCD7: 8D B5              BSR     IsValidCell     ; Is this cell out of bounds?
CCD9: 26 E0              BNE     $CCBB           ; Yes ... start a new run
CCDB: DD 88              STD     <drwMazeTmp     ; Hold the new coordinates
CCDD: 6D 84              TST     ,X              ; Already an open cell there?
CCDF: 27 E9              BEQ     $CCCA           ; Yes ... count it and keep going (no need to ch
CCE1: CE 09 F4           LDU     #$09F4          ; Buffer to hold cell values
;
; These checks prevent opening a cell if it would make a block-of-4-opens. An open block can't be
```

```
; drawn during game play.
CCE4: BD CC 49           JSR     GetNeighborCells ; Get the neighbor cell values
CCE7: A6 43              LDA     3,U              ; Cell to the left
CCE9: AB C4              ADDA    ,U               ; Cell to the upper left
CCEB: AB 41              ADDA    1,U              ; Cell above
CCED: 27 CC              BEQ     $CCBB            ; This would make the upper left corner 4-opens
;
CCEF: A6 41              LDA     1,U              ; Cell above
CCF1: AB 42              ADDA    2,U              ; Cell to the upper right
CCF3: AB 45              ADDA    5,U              ; Cell to the right
CCF5: 27 C4              BEQ     $CCBB            ; This would make the upper right corner 4-opens
;
CCF7: A6 45              LDA     5,U              ; Cell to the right
CCF9: AB 48              ADDA    8,U              ; Cell to the lower right
CCFB: AB 47              ADDA    7,U              ; Cell below
CCFD: 27 BC              BEQ     $CCBB            ; This would make the lower right corner 4-opens
;
CCFF: A6 47              LDA     7,U              ; Cell below
CD01: AB 46              ADDA    6,U              ; Cell to the lower left
CD03: AB 43              ADDA    3,U              ; Cell to the left
CD05: 27 B4              BEQ     $CCBB            ; This would make the lower left corner 4-opens
;
CD07: 6F 84              CLR     ,X               ; Open this cell up
CD09: 31 3F              LEAY    -1,Y             ; All 500 cells done?
CD0B: 26 BD              BNE     $CCCA            ; No ... use this and keep going
;
; This loops over the cells and sets the "solid wall" bits for directions that are blocked.
CD0D: 0F 7C              CLR     <drwMazeY        ; Start with ...
CD0F: 0F 7D              CLR     <drwMazeX        ; ... Y,X = 0,0
CD11: DC 7C              LDD     <drwMazeY        ; Get the current coordinate
CD13: BD CC 7B           JSR     GetCellPointer   ; Get the cell pointer
CD16: A6 84              LDA     ,X               ; Get the cell value
CD18: 4C                 INCA                     ; Is this a solid?
CD19: 27 26              BEQ     $CD41            ; Yes ... skip it
CD1B: DC 7C              LDD     <drwMazeY        ; Coordinates again
CD1D: CE 09 F4           LDU     #$09F4           ; Status buffer
CD20: BD CC 49           JSR     GetNeighborCells ; Get the status of the neighbors
CD23: A6 84              LDA     ,X               ; Get the value of the cell
CD25: C6 FF              LDB     #$FF             ; Value FF (solid) for compares
CD27: E1 41              CMPB    1,U              ; Cell above us open?
CD29: 26 02              BNE     $CD2D            ; No ... leave the bits open
```

```
CD2B: 8A 03              ORA     #$03            ; Set the "up" bits to solid wall
CD2D: E1 43              CMPB    3,U             ; Cell to the left open?
CD2F: 26 02              BNE     $CD33           ; No ... leave the bits open
CD31: 8A C0              ORA     #$C0            ; Set the "left" bits to solid wall
CD33: E1 45              CMPB    5,U             ; Cell to the right open?
CD35: 26 02              BNE     $CD39           ; No ... leave the bits open
CD37: 8A 0C              ORA     #$0C            ; Set the "right" bits to solid wall
CD39: E1 47              CMPB    7,U             ; Cell to the bottom open?
CD3B: 26 02              BNE     $CD3F           ; No ... leave the bits open
CD3D: 8A 30              ORA     #$30            ; Set the "down" bits to solid wall
CD3F: A7 84              STA     ,X              ; Set solid walls on the edge cells
;
; Add walls next to solid cells
CD41: C6 20              LDB     #$20            ; 32 for compare
CD43: 0C 7D              INC     <drwMazeX       ; Bump the X coordinate
CD45: D1 7D              CMPB    <drwMazeX       ; Reached end of row?
CD47: 26 C8              BNE     $CD11           ; No ... keep going
CD49: 0F 7D              CLR     <drwMazeX       ; Back to X=0
CD4B: 0C 7C              INC     <drwMazeY       ; Bump the Y coordinate
CD4D: D1 7C              CMPB    <drwMazeY       ; End of maze?
CD4F: 26 C0              BNE     $CD11           ; No ... keep going
;
; Add 70 regular doors
CD51: C6 46              LDB     #$46            ; 70 regular doors to make
CD53: CE CD AA           LDU     #$CDAA          ; Table of bit patterns for regular doors
CD56: 8D 15              BSR     $CD6D           ; Make a random regular door
CD58: 5A                 DECB                    ; All done?
CD59: 26 FB              BNE     $CD56           ; No ... do all
;
; Add 45 magic doors
CD5B: C6 2D              LDB     #$2D            ; 45 magic doors to make
CD5D: CE CD AE           LDU     #$CDAE          ; Table of bit patterns for magic doors
CD60: 8D 0B              BSR     $CD6D           ; Make a random magic door
CD62: 5A                 DECB                    ; All done?
CD63: 26 FB              BNE     $CD60           ; No ... do all
;
CD65: D6 97              LDB     <m0297          ; ?? A randomizer count? 0?
CD67: 3F                 SWI                     ; Get next ...
CD68: 07                                         ; SWI_7:Get random number:
CD69: 5A                 DECB                    ; Randomized the full count?
CD6A: 26 FB              BNE     $CD67           ; No ... do all
```

```
CD6C: 39                RTS                     ; Out
;
; Make a door (regular or magic) between two adjacent cells
CD6D: 34 76             PSHS    U,Y,X,B,A       ; Preserve registers
CD6F: 10 8E CD A6       LDY     #$CDA6          ; Bit patterns for solid walls in each direction
CD73: BD CC 71          JSR     GetRandomCell   ; Get random cell
CD76: DD 88             STD     <drwMazeTmp     ; Hold coordinates
CD78: E6 84             LDB     ,X              ; Get cell value
CD7A: C1 FF             CMPB    #$FF            ; Is it solid?
CD7C: 27 F5             BEQ     $CD73           ; Yes ... find an open cell
CD7E: 3F                SWI                     ; Get a random number
CD7F: 07                                        ; SWI_7:Get random number:
CD80: 84 03             ANDA    #$03            ; Make it a direction
CD82: 97 8A             STA     <drwMazeDir     ; Store the direction
CD84: E5 A6             BITB    A,Y             ; Is that direction open (no solid and no existi
CD86: 26 EB             BNE     $CD73           ; No ... find another
CD88: EA C6             ORB     A,U             ; Or in the pattern for the door (magic or regul
CD8A: E7 84             STB     ,X              ; Set the new pattern
CD8C: DC 88             LDD     <drwMazeTmp     ; Get coordinates
CD8E: BD D1 1B          JSR     StepInDirection ; Step in that direction
CD91: D6 8A             LDB     <drwMazeDir     ; Get the direction we came in
CD93: CB 02             ADDB    #$02            ; Flip it ...
CD95: C4 03             ANDB    #$03            ; ... around
CD97: A6 84             LDA     ,X              ; Get value
CD99: AA C5             ORA     B,U             ; Make same door in ...
CD9B: A7 84             STA     ,X              ; ... on both sides
CD9D: 35 F6             PULS    A,B,X,Y,U,PC    ; Restore and out

RandomSeeds:
; These seeds control the shape of the dungeon and placement
; of creatures. Three bytes instead of over 1K of data -- good
; idea. The levels overlap seeds as shown below.
;     0-------
;        1-------
;           2-------
;              3-------
;                 4-------
CD9F: 73 C7 5D 97 F3 13 87

; Bit positions for walls in a given direction (0-3)
;     0U 1R 2D 3L
```

```
CDA6: 03 0C 30 C0

; Bit positions to add a regular door in a given direction
CDAA: 01 04 10 40

; Bit positions to add a magic door in a give direction
CDAE: 02 08 20 80

ShowMap:
; This is the draw-screen function for the scroll
CDB2: DE 0B              LDU     <backScreen      ; Drawing screen descriptor
CDB4: CC 1F 1F           LDD     #$1F1F           ; 32x32
CDB7: DD 7C              STD     <drwMazeY        ; Store the count
;
; First draw the open/closed states of all cells in the maze;
CDB9: DC 7C              LDD     <drwMazeY        ; Get the current map coordinate
CDBB: 8D 54              BSR     GetMapCellMem    ; Get a screen pointer to the cell in Y
CDBD: BD CC 7B           JSR     GetCellPointer   ; Get a pointer to the maze memory
CDC0: 5F                 CLRB                     ; Initial cell state (open)
CDC1: A6 84              LDA     ,X               ; Get the cell wall state
CDC3: 4C                 INCA                     ; Is it FF (solid)?
CDC4: 26 01              BNE     $CDC7            ; No ... draw it open
CDC6: 5A                 DECB                     ; Cell state is solid
CDC7: 86 06              LDA     #$06             ; Six rows per cell
CDC9: E7 A4              STB     ,Y               ; Draw ...
CDCB: 31 A8 20           LEAY    $20,Y            ; ... six ...
CDCE: 4A                 DECA                     ; ... row ...
CDCF: 26 F8              BNE     $CDC9            ; ... block
CDD1: 0A 7D              DEC     <drwMazeX        ; Move left one cell
CDD3: 2A E4              BPL     $CDB9            ; Do all of the row
CDD5: 86 1F              LDA     #$1F             ; Restart row at ...
CDD7: 97 7D              STA     <drwMazeX        ; ... far right
CDD9: 0A 7C              DEC     <drwMazeY        ; Move up a row
CDDB: 2A DC              BPL     $CDB9            ; Do all rows
;
CDDD: 0D 94              TST     <scrollType      ; Is this a "seer" scroll?
CDDF: 27 4A              BEQ     $CE2B            ; No ... skip drawing monsters and objects
;
; Show objects on floor (Seer Scroll)
CDE1: 0F 91              CLR     <restartFind     ; Start at top of list
CDE3: BD CF 63           JSR     GetNextOject     ; Get next object on floor
```

```
CDE6: 27 0F             BEQ     $CDF7           ; All done ... do monsters
CDE8: 6D 05             TST     5,X             ; Is this on the floor?
CDEA: 26 F7             BNE     $CDE3           ; No ... don't show it
CDEC: EC 02             LDD     2,X             ; Get the Y,X coordinate
CDEE: 8D 21             BSR     GetMapCellMem   ; Get map screen pointer for coordinate
CDF0: CC 00 08          LDD     #$0008          ; 4 byte graphics pattern (small dot) for object
CDF3: 8D 28             BSR     DrawMapSymbol   ; Draw an object on the map
CDF5: 20 EC             BRA     $CDE3           ; Draw next object on floor
;
; Show monsters (Seer Scroll)
CDF7: 8E 03 C3          LDX     #$03C3          ; First monster (actually one slot before)
CDFA: 30 88 11          LEAX    $11,X           ; Next monster
CDFD: 8C 05 F4          CMPX    #$05F4          ; All done?
CE00: 27 29             BEQ     $CE2B           ; Yes ... draw holes and ladders
CE02: 6D 0C             TST     12,X            ; Is this creature active?
CE04: 27 F4             BEQ     $CDFA           ; No ... skip it
CE06: EC 0F             LDD     15,X            ; Get the creature coordinates
CE08: 8D 07             BSR     GetMapCellMem   ; Get a pointer to the map screen
CE0A: CC 10 54          LDD     #$1054          ; 4 byte graphics pattern (large dot) for monste
CE0D: 8D 0E             BSR     DrawMapSymbol   ; Draw a creature on the map
CE0F: 20 E9             BRA     $CDFA           ; Do all creatures

GetMapCellMem:
; This functions returns a pointer to the screen for a given cell (Y,X coordinate).
; On the screen, cells are 6-rows high and 8-pixels (one byte) wide.
;
; Param A,B: The Y,X cell coordinate
; Param U:   Start of screen memory
; Return Y:  Pointer to the screen memory for the cell
;
CE11: 1F 02             TFR     D,Y             ; Hold the coordinate
CE13: C6 C0             LDB     #$C0            ; Multiply Y time ...
CE15: 3D                MUL                     ; ... 32*6 (6 rows per cell)
CE16: E3 C4             ADDD    ,U              ; Offset to screen row
CE18: 1E 02             EXG     D,Y             ; X to B (pointer to Y)
CE1A: 31 A5             LEAY    B,Y             ; Add in the column offset
CE1C: 39                RTS                     ; Done

DrawMapSymbol:
; Map symbols are 4 bytes on the screen. They are symmetrical in that the
; top and bottom row are the same and the middle two rows are the same.
```

```
;
; Param A: top and bottom pixel pattern
; Param B: middle 2 rows pixel pattern
;
CE1D: A7 A8 20           STA     $20,Y           ; Top row pattern (A)
CE20: E7 A8 40           STB     $40,Y           ; Middle row pattern (B)
CE23: E7 A8 60           STB     $60,Y           ; Middle row pattern (B)
CE26: A7 A9 00 80        STA     $0080,Y         ; Bottom row pattern (A)
CE2A: 39                 RTS                     ; Done

; Draws holes and player on map (both scroll types)
CE2B: DC 13              LDD     <playerY        ; Player Y,X coordinate
CE2D: 8D E2              BSR     GetMapCellMem   ; Player's cell on map screen
CE2F: CC 24 18           LDD     #$2418          ; 4 byte pattern (X) for player
CE32: 8D E9              BSR     DrawMapSymbol   ; Draw player on the map
;
; Two passes here. 1st draw the holes in the current ceiling. Then draw the holes
; in the current floor.
CE34: 9E 86              LDX     <currentHoles   ; Pointer to ceiling holes/ladders for current l
CE36: 8D 00              BSR     $CE38           ; Draw the holes in this floor's ceiling (fall t
CE38: A6 80              LDA     ,X+             ; Get hole type
CE3A: 2B EE              BMI     $CE2A           ; End of list? Yes ... out
CE3C: EC 81              LDD     ,X++            ; Get the hole's Y,X coordinate
CE3E: 8D D1              BSR     GetMapCellMem   ; Convert to screen pointer
CE40: CC 3C 24           LDD     #$3C24          ; 4 byte pattern (open circle) for hole
CE43: 8D D8              BSR     DrawMapSymbol   ; Draw the hole
CE45: 20 F1              BRA     $CE38           ; Keep going

CE47: 34 12              PSHS    X,A
CE49: 8E CF 48           LDX     #$CF48
CE4C: 0D 73              TST     <m0273          ;
CE4E: 26 0C              BNE     $CE5C           ;
CE50: 30 89 00 01        LEAX    $0001,X
CE54: 0D 74              TST     <m0274          ;
CE56: 26 04              BNE     $CE5C           ;
CE58: 30 89 FF F5        LEAX    -$000B,X
CE5C: 96 8B              LDA     <m028B          ;
CE5E: A6 86              LDA     A,X
CE60: 97 4F              STA     <m024F          ;
CE62: 97 50              STA     <m0250          ;
CE64: 35 92              PULS    A,X,PC
```

```
NormalDisplay:
; This is the routine called to draw the normal game display
CE66: 3F                SWI
CE67: 09                                        ; SWI_9:Clear secondary screen:
CE68: 0F 8B             CLR     <m028B          ;
CE6A: DC 13             LDD     <playerY        ;
CE6C: DD 7C             STD     <drwMazeY       ;
CE6E: 8D D7             BSR     $CE47           ;
CE70: DC 7C             LDD     <drwMazeY       ;
CE72: BD CC 7B          JSR     GetCellPointer  ;
CE75: A6 84             LDA     ,X
CE77: CE 09 F4          LDU     #$09F4
CE7A: 8E 00 04          LDX     #$0004
CE7D: 1F 89             TFR     A,B
CE7F: C4 03             ANDB    #$03
CE81: E7 44             STB     4,U
CE83: E7 C0             STB     ,U+
CE85: 44                LSRA
CE86: 44                LSRA
CE87: 30 1F             LEAX    -1,X
CE89: 26 F2             BNE     $CE7D           ;
CE8B: D6 23             LDB     <playerDir      ;
CE8D: CE 09 F4          LDU     #$09F4
CE90: 33 C5             LEAU    B,U
CE92: 10 8E DB DE       LDY     #$DBDE          ; Wall pictures
CE96: A6 A0             LDA     ,Y+
CE98: 2B 3E             BMI     $CED8           ;
CE9A: E6 C6             LDB     A,U
CE9C: 58                ASLB
CE9D: C1 04             CMPB    #$04
CE9F: 26 08             BNE     $CEA9           ;
CEA1: AE A5             LDX     B,Y
CEA3: 0A 75             DEC     <m0275          ;
CEA5: 8D 27             BSR     $CECE           ;
CEA7: C6 06             LDB     #$06
CEA9: AE A5             LDX     B,Y
CEAB: 8D 21             BSR     $CECE           ;
CEAD: 31 28             LEAY    8,Y
CEAF: 20 E5             BRA     $CE96           ;
CEB1: 39                RTS
```

```
CEB2: 1F 12             TFR     X,Y
CEB4: 6D C5             TST     B,U
CEB6: 26 F9             BNE     $CEB1           ;
CEB8: DB 23             ADDB    <playerDir      ;
CEBA: D7 8A             STB     <drwMazeDir     ;
CEBC: DC 7C             LDD     <drwMazeY       ;
CEBE: BD D1 1B          JSR     StepInDirection ;
CEC1: BD CF 82          JSR     GetCreatureAt   ;
CEC4: 27 EB             BEQ     $CEB1           ;
CEC6: 1E 12             EXG     X,Y
CEC8: 6D 22             TST     2,Y
CECA: 27 02             BEQ     $CECE           ;
CECC: 0A 75             DEC     <m0275          ;
CECE: 34 40             PSHS    U
CED0: 3F                SWI
CED1: 00                                        ; SWI_0:Light level:
CED2: DE 0B             LDU     <backScreen     ;
CED4: 3F                SWI
CED5: 01                                        ; SWI_1:Draw picture X on screen:
CED6: 35 C0             PULS    U,PC

CED8: DC 7C             LDD     <drwMazeY       ;
CEDA: BD CF 82          JSR     GetCreatureAt   ;
CEDD: 27 0C             BEQ     $CEEB           ;
CEDF: 1F 12             TFR     X,Y
CEE1: E6 2D             LDB     13,Y
CEE3: 58                ASLB
CEE4: 8E DA A3          LDX     #CreaturePictures; Get the picture ...
CEE7: AE 85             LDX     B,X             ; ... of the creature
CEE9: 8D DD             BSR     $CEC8           ;
CEEB: C6 03             LDB     #$03
CEED: 8E DC B0          LDX     #$DCB0          ; Draw creature coming ...
CEF0: 8D C0             BSR     $CEB2           ; ... from left
CEF2: C6 01             LDB     #$01
CEF4: 8E DC B9          LDX     #$DCB9          ; Draw creature coming ...
CEF7: 8D B9             BSR     $CEB2           ; ... from right
CEF9: 8E DD 3C          LDX     #$DD3C          ; ?? Part of the hole-in-floor
CEFC: DC 7C             LDD     <drwMazeY       ;
CEFE: BD CF E1          JSR     ScanForHole     ;
CF01: 2B 06             BMI     $CF09           ;
CF03: 8E DC C2          LDX     #HoleList       ; Holes and ladders pictures
```

```
CF06: 48                ASLA
CF07: AE 86             LDX     A,X
CF09: 8D C3             BSR     $CECE           ;
CF0B: 0F 91             CLR     <restartFind    ;
CF0D: DC 7C             LDD     <drwMazeY       ;
CF0F: BD CF 53          JSR     GetObjectAtCoor ;
CF12: 27 10             BEQ     $CF24           ;
CF14: A6 0A             LDA     10,X
CF16: 48                ASLA
CF17: 8E D9 EE          LDX     #ClassPictures  ; Object pictures (by class)
CF1A: AE 86             LDX     A,X
CF1C: 0A 75             DEC     <m0275          ;
CF1E: 8D AE             BSR     $CECE           ;
CF20: 8D AC             BSR     $CECE           ;
CF22: 20 E9             BRA     $CF0D           ;
CF24: 6D C4             TST     ,U
CF26: 26 15             BNE     $CF3D           ;
CF28: 96 23             LDA     <playerDir      ;
CF2A: 97 8A             STA     <drwMazeDir     ;
CF2C: DC 7C             LDD     <drwMazeY       ;
CF2E: BD D1 1B          JSR     StepInDirection ;
CF31: DD 7C             STD     <drwMazeY       ;
CF33: 0C 8B             INC     <m028B          ;
CF35: 96 8B             LDA     <m028B          ;
CF37: 81 09             CMPA    #$09
CF39: 10 2F FF 31       LBLE    $CE6E           ;
CF3D: 39                RTS

; This table is referenced from code at CE49. ?? It might have to do with starting points for cells
CF3E: C8 80
CF40: 50 32
CF42: 1F 14
CF44: 0C 08
CF46: 04 02
CF48: FF
CF49: 9C 64
CF4B: 41 28
CF4D: 1A 10
CF4F: 0A 06
CF51: 03 01
```

```
GetObjectAtCoor:
; Get the next object on this level at the given coordinates
CF53: 8D 0E             BSR     GetNextOject     ; Get the next object
CF55: 27 0B             BEQ     $CF62            ; End of list ... out
CF57: 10 A3 02          CMPD    2,X              ; Do the coordinates match?
CF5A: 26 F7             BNE     GetObjectAtCoor  ; No ... keep looking
CF5C: 6D 05             TST     5,X              ; Is the object on the floor?
CF5E: 26 F3             BNE     GetObjectAtCoor  ; No ... keep looking
CF60: 1C FB             ANDCC   #$FB             ; Clear the zero flag (object found)
CF62: 39                RTS                      ; Done

GetNextOject:
; Get the next (or first) object on this level. Start over at top of list
; if requested or continue from last iteration.
;
; This walks the objects in memory without looking at their chain pointers.
;
; Param >$91 0 to start at top of list, 1 to continue from last
; Return object descriptor in X (if found)
; Return Z=1 if no more, Z=0 if next is in X
;
CF63: 34 02             PSHS    A                ; Preserve A
CF65: 96 81             LDA     <currentLevel    ; Get current level number
CF67: 9E 92             LDX     <objIterator     ; Get current object pointer
CF69: 0D 91             TST     <restartFind     ; Start at top of list?
CF6B: 26 05             BNE     $CF72            ; No ... continue from last time
CF6D: 8E 0B 07          LDX     #$0B07           ; Start of list (-14 ... one slot before)
CF70: 0A 91             DEC     <restartFind     ; Next time through we won't restart
CF72: 30 0E             LEAX    14,X             ; Get pointer to next object
CF74: 9F 92             STX     <objIterator     ; Remember it
CF76: 9C 0F             CMPX    <nextObjSlot     ; Have we reached the end of the list?
CF78: 27 06             BEQ     $CF80            ; Yes ... out with nothing found
CF7A: A1 04             CMPA    4,X              ; Level the same as what we want?
CF7C: 26 F4             BNE     $CF72            ; No ... next object
CF7E: 1C FB             ANDCC   #$FB             ; Clear the Z flag meaning there is a next objec
CF80: 35 82             PULS    A,PC             ; Restore A and out

GetCreatureAt:
; Find the creature (if any) at the given Y,X coordinate.
; Param A,Y: The Y,X coordinate
; Return X: Pointer to creature if found
```

```
; Return NZ if found or Z if not found
;
CF82: 8E 03 C3          LDX     #$03C3          ; Start of creatures (minus pre-decrement)
CF85: 30 88 11          LEAX    $11,X           ; Point to next creature
CF88: 8C 05 F4          CMPX    #$05F4          ; Reached the end of the list?
CF8B: 27 09             BEQ     $CF96           ; Yes ... return Z set (not found)
CF8D: 10 A3 0F          CMPD    15,X            ; Right coordinates?
CF90: 26 F3             BNE     $CF85           ; No ... keep looking
CF92: 6D 0C             TST     12,X            ; Creature is active?
CF94: 27 EF             BEQ     $CF85           ; No ... keep looking (yes, Z=0)
CF96: 39                RTS                     ; Return

GetRandCell:
; Get a random open cell in the current maze.
; Return A,B: The Y,X coordinate of the random open cell
;
CF97: 34 16             PSHS    X,B,A           ; Preserve registers
CF99: BD CC 71          JSR     GetRandomCell   ; Get a random cell
CF9C: ED E4             STD     ,S              ; Put it in return in case it is good
CF9E: A6 84             LDA     ,X              ; Get the cell value
CFA0: 4C                INCA                    ; FF means all walls (solid)
CFA1: 27 F6             BEQ     $CF99           ; This is not a valid cell ... keep looking
CFA3: 35 96             PULS    A,B,X,PC        ; Return the random cell in A,B

CreateCreature:
; Create a new creature of the given type at the next available monster slot.
; The new creature is given a random valid coordinate.
; Param A: The monster type
;
CFA5: 34 76             PSHS    U,Y,X,B,A       ; Preserve registers
CFA7: CE 03 C3          LDU     #$03C3          ; Start of creatures (03D4 minus 11 pre-incremen
CFAA: 33 C8 11          LEAU    $11,U           ; Point to next creature slot
CFAD: 6D 4C             TST     12,U            ; Is this a living creature?
CFAF: 26 F9             BNE     $CFAA           ; Yes ... find an empty slot
CFB1: 6A 4C             DEC     12,U            ; Mark this creature living
CFB3: A7 4D             STA     13,U            ; Set the type
CFB5: C6 08             LDB     #$08            ; 8 bytes of init data
CFB7: 3D                MUL                     ; Type * 8
CFB8: C3 DA BB          ADDD    #$DABB          ; Add to creature-class data table
CFBB: 1F 02             TFR     D,Y             ; Source to Y
CFBD: 1F 31             TFR     U,X             ; Destination to X
```

```
CFBF: 86 08             LDA     #$08            ; Bytes to copy = 8
CFC1: BD C0 4B          JSR     CopyYtoX        ; Copy the 8 bytes of initial data
CFC4: 8D D1             BSR     GetRandCell     ; Get a random cell
CFC6: 8D BA             BSR     GetCreatureAt   ; Is there already a creature there?
CFC8: 26 FA             BNE     $CFC4           ; Yes ... keep looking
CFCA: ED 4F             STD     15,U            ; Put the creature in the random cell
CFCC: 1F 31             TFR     U,X
CFCE: BD C2 5C          JSR     ReserveTask     ;
CFD1: AF 45             STX     5,U
CFD3: CC D0 41          LDD     #$D041          ; task pointer
CFD6: ED 43             STD     3,U
CFD8: A6 06             LDA     6,X
CFDA: C6 04             LDB     #$04
CFDC: BD C2 1D          JSR     $C21D           ;
CFDF: 35 F6             PULS    A,B,X,Y,U,PC

ScanForHole:
; There can only be one hole in the current cell. This scans for a hole in the current
; cell and returns the type in A (or A is negative for none).
;
; Param A,B: Y,X cell coordinate to check
; Return A: Hole type (if found: 00=hole in ceiling, 01=ladder in ceiling, 10=hole in floor, 11=lad
; Return Negative if not found, positive if found
;
CFE1: 34 56             PSHS    U,X,B,A         ; Preserve registers
CFE3: DE 86             LDU     <currentHoles   ; Holes in ceiling of current level
CFE5: 8D 0B             BSR     $CFF2           ; Check for a hole in the ceiling
CFE7: 4D                TSTA                    ; Is there one?
CFE8: 2A 04             BPL     $CFEE           ; Yes ... keep the data and skip the floor
CFEA: 8D 06             BSR     $CFF2           ; Check for a hole in the floor
CFEC: 8B 02             ADDA    #$02            ; Flag that it is in the floor
CFEE: A7 E4             STA     ,S              ; Save the hole result (returns in A)
CFF0: 35 D6             PULS    A,B,X,U,PC      ; Out
;
; Run a list of holes and return the type of the hole
; that has the check coordiantes (or bit 7 set if not found)
CFF2: A6 C0             LDA     ,U+             ; Get hole type
CFF4: 2B 06             BMI     $CFFC           ; End of list ... out
CFF6: AE C1             LDX     ,U++            ; Get hole coordinate
CFF8: AC 62             CMPX    2,S             ; Matches the test coordinate?
CFFA: 26 F6             BNE     $CFF2           ; No ... keep looking
```

```
CFFC: 39                RTS                     ; Return A is hole type or bit 7 set if not found

HolesAndLadders:
; The game maintains a pointer-to-holes-in-current-ceiling in $286. The next list
; after is the list of holes-in-current-floor.
;
; Holes between surface and level 0 (none ... no surface)
CFFD: 80
;
; Holes between levels 0 and 1
CFFE: 01 00 17 ; Ladder Y=00, X=17
D001: 00 0F 04 ; Hole   Y=0F, X=04
D004: 00 14 11 ; Hole   Y=14, X=11
D007: 01 1C 1E ; Ladder Y=1C, X=1E
D00A: 80
;
; Holes between levels 1 and 2
D00B: 01 02 03 ; Ladder Y=02, X=03
D00E: 00 03 1F ; Hole   Y=03, X=1F
D011: 00 13 14 ; Hole   Y=13, X=14
D014: 00 1F 00 ; Hole   Y=1F, X=00
D017: 80
;
; Holes between levels 3 and 4 (none)
D018: 80
;
; Holes between levels 4 and 5
D019: 00 00 1F ; Hole   Y=00, X=1F
D01C: 00 05 00 ; Hole   Y=05, X=00
D01F: 00 16 1C ; Hole   Y=16, X=1C
D022: 00 1F 10 ; Hole   Y=1F, X=10
D025: 80
;
; Holes between levels 5 and 6 (none ... no level 6)
D026: 80

; ?? Task 4 ??
D027: 9E 82             LDX     <m0282          ;
D029: C6 0B             LDB     #$0B
D02B: 4F                CLRA
D02C: AB 85             ADDA    B,X
```

```
D02E: 5A              DECB
D02F: 2A FB           BPL     $D02C           ;
D031: 81 20           CMPA    #$20
D033: 24 08           BCC     $D03D           ;
D035: 3F              SWI
D036: 07                                      ; SWI_7:Get random number:
D037: 84 07           ANDA    #$07
D039: 8B 02           ADDA    #$02
D03B: 6C 86           INC     A,X
D03D: CC 05 08        LDD     #$0508          ; ?? reloads
D040: 39              RTS

; ?? Task ?
D041: 10 AE 45        LDY     5,U             ; Get pointer to creature
D044: 0D 2B           TST     <wizardDead     ; Is the wizard dead?
D046: 26 22           BNE     $D06A           ; Yes ... skip all actions
D048: E6 2C           LDB     12,Y            ; Is this creature alive?
D04A: 26 01           BNE     $D04D           ; Yes ... let it move
D04C: 39              RTS                     ; No ... done with this one
;
D04D: A6 2D           LDA     13,Y            ; Creature type
D04F: 81 06           CMPA    #$06            ; SCORPION?
D051: 27 1A           BEQ     $D06D           ; Yes ... they don't pick up things
D053: 81 0A           CMPA    #$0A            ; DEMON or WIZARD?
D055: 2C 16           BGE     $D06D           ; Yes ... they don't pick up things
D057: EC 2F           LDD     15,Y            ; Monster's coordinates
D059: 0F 91           CLR     <restartFind    ; Reset find cursor to 1st object
D05B: BD CF 53        JSR     GetObjectAtCoor ; Get an object on the floor here
D05E: 27 0D           BEQ     $D06D           ; Nothing to pick up
D060: EC 28           LDD     8,Y             ; This monster's list of held items
D062: AF 28           STX     8,Y             ; Push this object to ...
D064: ED 84           STD     ,X              ; ... the top of the list
D066: 6A 05           DEC     5,X             ; ??
D068: 3F              SWI                     ; Update the screen (stuff was picked up)
D069: 0E                                      ; SWI_E:Display playing screen:
;
D06A: 7E D1 03        JMP     $D103           ; Skip all action
;
D06D: EC 2F           LDD     15,Y            ; Monster's coordinates
D06F: 10 93 13        CMPD    <playerY        ; Same as the players?
D072: 26 3E           BNE     $D0B2           ; No ... no attack
```

```
D074: A6 2D             LDA     13,Y            ; Play creature ...
D076: C6 FF             LDB     #$FF            ; ... sound ...
D078: 3F                SWI                     ; ... at full volume
D079: 1C                                        ; SWI_1C:Play sound A at volume B:
D07A: CC 80 80          LDD     #$8080
D07D: 9E 1D             LDX     <leftHand       ;
D07F: 8D 1D             BSR     $D09E           ;
D081: 9E 1F             LDX     <rightHand      ;
D083: 8D 19             BSR     $D09E           ;
D085: 97 1A             STA     <m021A          ;
D087: D7 1C             STB     <m021C          ;
D089: 1F 21             TFR     Y,X
D08B: CE 02 17          LDU     #$0217
D08E: BD D3 D7          JSR     $D3D7           ;
D091: 2B 06             BMI     $D099           ;
D093: 3F                SWI                     ; Play player hit sound at full volume
D094: 1B                                        ; SWI_1B:Play sound i at full volume:
D095: 13                                        ; 13 = Player hit
D096: BD D4 0C          JSR     $D40C           ; ??
D099: 3F                SWI                     ; Update the heart rate
D09A: 0C                                        ; SWI_C:Update heart rate:
D09B: 7E D1 0F          JMP     $D10F           ;
;
D09E: 34 16             PSHS    X,B,A
D0A0: 27 0E             BEQ     $D0B0           ;
D0A2: A6 0A             LDA     10,X
D0A4: 81 03             CMPA    #$03
D0A6: 26 08             BNE     $D0B0           ;
D0A8: AE 06             LDX     6,X
D0AA: AC E4             CMPX    ,S
D0AC: 24 02             BCC     $D0B0           ;
D0AE: AF E4             STX     ,S
D0B0: 35 96             PULS    A,B,X,PC

; We can see the player
D0B2: 91 13             CMPA    <playerY        ;
D0B4: 26 0D             BNE     $D0C3           ;
D0B6: A6 A8 10          LDA     $10,Y
D0B9: C6 01             LDB     #$01
D0BB: 90 14             SUBA    <playerX        ;
D0BD: 2B 11             BMI     $D0D0           ;
```

```
D0BF: C6 03             LDB     #$03
D0C1: 20 0D             BRA     $D0D0           ;
D0C3: EC 2F             LDD     15,Y
D0C5: D1 14             CMPB    <playerX        ;
D0C7: 26 1B             BNE     $D0E4           ;
D0C9: C6 02             LDB     #$02
D0CB: 90 13             SUBA    <playerY        ;
D0CD: 2B 01             BMI     $D0D0           ;
D0CF: 5F                CLRB
D0D0: D7 8A             STB     <drwMazeDir     ;
D0D2: EC 2F             LDD     15,Y
D0D4: 8D 60             BSR     $D136           ;
D0D6: 26 0C             BNE     $D0E4           ;
D0D8: 10 93 13          CMPD    <playerY        ;
D0DB: 26 F7             BNE     $D0D4           ;
D0DD: D6 8A             LDB     <drwMazeDir     ;
D0DF: E7 2E             STB     14,Y
D0E1: 5F                CLRB
D0E2: 20 1D             BRA     $D101           ;
D0E4: 8E D1 14          LDX     #$D114
D0E7: 3F                SWI
D0E8: 07                                        ; SWI_7:Get random number:
D0E9: 4D                TSTA
D0EA: 2B 02             BMI     $D0EE           ;
D0EC: 30 03             LEAX    3,X
D0EE: 84 03             ANDA    #$03
D0F0: 26 02             BNE     $D0F4           ;
D0F2: 30 01             LEAX    1,X
D0F4: 86 03             LDA     #$03
D0F6: E6 80             LDB     ,X+
D0F8: 8D 55             BSR     $D14F           ;
D0FA: 27 07             BEQ     $D103           ;
D0FC: 4A                DECA
D0FD: 26 F7             BNE     $D0F6           ;

; ?? different rates if in room vs not

D0FF: C6 02             LDB     #$02            ; ?? returned if not in player's room
D101: 8D 4C             BSR     $D14F           ;
;
D103: A6 26             LDA     6,Y             ; Normal task speed reload
```

```
D105: AE 2F             LDX     15,Y            ; Is the creature with ...
D107: 9C 13             CMPX    <playerY        ; ... the player?
D109: 26 06             BNE     $D111           ; No ... use the normal rate
D10B: 3F                SWI                     ; Update the screen
D10C: 0E                                        ; SWI_E:Display playing screen:
D10D: 0F B5             CLR     <m02B5          ;
D10F: A6 27             LDA     7,Y
D111: C6 04             LDB     #$04
D113: 39                RTS

D114: 00 03 01 00 01 03 00

StepInDirection:
; This function moves the Y,X coordinates in A,B in the direction in $8A.
;
; Param A,B: The Y,X coordinates
;
; Return A,B: The new Y,X coordinates
; Return X: Pointer to the new cell
;
D11B: 34 06             PSHS    B,A             ; Hold coordinates
D11D: D6 8A             LDB     <drwMazeDir     ; Get direction
D11F: C4 03             ANDB    #$03            ; Only four
D121: 58                ASLB                    ; 2 bytes for each table entry
D122: 8E D1 2E          LDX     #$D12E          ; Table of X,Y offsets for direction
D125: EC 85             LDD     B,X             ; Get the X,Y offsets
D127: AB E0             ADDA    ,S+             ; Offset the Y
D129: EB E0             ADDB    ,S+             ; Offset the X
D12B: 7E CC 7B          JMP     GetCellPointer  ; Get cell pointer and return
;
; Y,X offsets for each direction
;
D12E: FF 00 ; 00 Up    (Y-1, X)
D130: 00 01 ; 01 Right (Y,   X+1)
D132: 01 00 ; 02 Down  (y+1, X)
D134: 00 FF ; 03 Left  (Y,   X-1)

D136: 34 76             PSHS    U,Y,X,B,A
D138: 8D E1             BSR     StepInDirection ;
D13A: BD CC 8E          JSR     IsValidCell     ;
D13D: 26 0E             BNE     $D14D           ;
```

```
D13F: 1F 03             TFR     D,U
D141: A6 84             LDA     ,X
D143: 4C                INCA
D144: 27 06             BEQ     $D14C           ;
D146: EF E4             STU     ,S
D148: AF 62             STX     2,S
D14A: 86 01             LDA     #$01
D14C: 4A                DECA
D14D: 35 F6             PULS    A,B,X,Y,U,PC

D14F: 34 16             PSHS    X,B,A
D151: EB 2E             ADDB    14,Y
D153: C4 03             ANDB    #$03
D155: D7 8A             STB     <drwMazeDir     ;
D157: EC 2F             LDD     15,Y
D159: 8D DB             BSR     $D136           ;
D15B: 26 3C             BNE     $D199           ;
D15D: BD CF 82          JSR     GetCreatureAt   ;
D160: 26 37             BNE     $D199           ;
D162: ED 2F             STD     15,Y
D164: D6 8A             LDB     <drwMazeDir     ;
D166: E7 2E             STB     14,Y
D168: EC 2F             LDD     15,Y
D16A: 90 13             SUBA    <playerY        ;
D16C: 2A 01             BPL     $D16F           ;
D16E: 40                NEGA
D16F: D0 14             SUBB    <playerX        ;
D171: 2A 01             BPL     $D174           ;
D173: 50                NEGB
D174: D7 C1             STB     <holdHole       ;
D176: 91 C1             CMPA    <holdHole       ;
D178: 2C 02             BGE     $D17C           ;
D17A: 1E 89             EXG     A,B
D17C: 97 C1             STA     <holdHole       ;
D17E: 81 08             CMPA    #$08
D180: 2E 16             BGT     $D198           ;
D182: C1 02             CMPB    #$02
D184: 2E 12             BGT     $D198           ;
D186: 3F                SWI
D187: 07                                        ; SWI_7:Get random number:
D188: 85 01             BITA    #$01
```

```
D18A: 27 0A             BEQ     $D196           ;
D18C: 96 C1             LDA     <holdHole       ;
D18E: C6 1F             LDB     #$1F
D190: 3D                MUL
D191: 53                COMB
D192: A6 2D             LDA     13,Y
D194: 3F                SWI
D195: 1C                                        ; SWI_1C:Play sound A at volume B:
D196: 0A B5             DEC     <m02B5          ;
D198: 4F                CLRA
D199: 35 96             PULS    A,B,X,PC

; ?? Task 3 ??
D19B: DE 24             LDU     <torchPtr       ;
D19D: 27 1D             BEQ     $D1BC           ;
D19F: A6 46             LDA     6,U
D1A1: 27 19             BEQ     $D1BC           ;
D1A3: 4A                DECA
D1A4: A7 46             STA     6,U
D1A6: 81 05             CMPA    #$05
D1A8: 2E 06             BGT     $D1B0           ;
D1AA: C6 18             LDB     #$18
D1AC: E7 49             STB     9,U
D1AE: 6F 4B             CLR     11,U
D1B0: A1 47             CMPA    7,U
D1B2: 2C 02             BGE     $D1B6           ;
D1B4: A7 47             STA     7,U
D1B6: A1 48             CMPA    8,U
D1B8: 2C 02             BGE     $D1BC           ;
D1BA: A7 48             STA     8,U
D1BC: 0A B5             DEC     <m02B5          ;
D1BE: CC 01 08          LDD     #$0108
D1C1: 39                RTS

; ?? Task 1 ??
D1C2: 0D B5             TST     <m02B5          ;
D1C4: 26 07             BNE     $D1CD           ;
D1C6: 8E CD B2          LDX     #ShowMap        ; Are we showing the ...
D1C9: 9C B2             CMPX    <displayFunction ; ... map display?
D1CB: 26 04             BNE     $D1D1           ;
D1CD: 0F B5             CLR     <m02B5          ;
```

```
D1CF: 3F                SWI
D1D0: 0E                                        ; SWI_E:Display playing screen:
D1D1: CC 03 04          LDD     #$0304
D1D4: 39                RTS

; ?? Task 2 ??
D1D5: 4F                CLRA
D1D6: 5F                CLRB
D1D7: 93 21             SUBD    <m0221          ;
D1D9: BD D3 79          JSR     DRight6         ;
D1DC: D3 21             ADDD    <m0221          ;
D1DE: 2E 02             BGT     $D1E2           ;
D1E0: 4F                CLRA
D1E1: 5F                CLRB
D1E2: DD 21             STD     <m0221          ;
D1E4: 3F                SWI
D1E5: 0C                                        ; SWI_C:Update heart rate:
D1E6: 96 AF             LDA     <heartCounterRel ;
D1E8: C6 02             LDB     #$02
D1EA: 39                RTS

; ?? Task 0 ??
D1EB: 0D 77             TST     <gameMode       ; Are we in a demo?
D1ED: 26 2C             BNE     $D21B           ; Yes ... handle demo commands
;
D1EF: BD C3 29          JSR     CharFromBuf     ;
D1F2: 4D                TSTA
D1F3: 27 53             BEQ     $D248           ;
D1F5: 0D 28             TST     <fainting       ; Fainting?
D1F7: 26 F6             BNE     $D1EF           ; Yes ... drop all characters from the buffer
D1F9: 81 20             CMPA    #$20
D1FB: 27 18             BEQ     $D215           ;
D1FD: C6 1F             LDB     #$1F
D1FF: 81 0D             CMPA    #$0D
D201: 27 0F             BEQ     $D212           ;
D203: C6 24             LDB     #$24
D205: 81 08             CMPA    #$08
D207: 27 09             BEQ     $D212           ;
D209: 5F                CLRB
D20A: 81 41             CMPA    #$41
D20C: 25 04             BCS     $D212           ;
```

```
D20E: 81 5A             CMPA    #$5A
D210: 23 03             BLS     $D215           ;
D212: 1F 98             TFR     B,A
D214: 8C                ; CMPX  opcode to skip next instruction
D215: 84 1F             ANDA    #$1F
D217: 8D 33             BSR     $D24C           ;
D219: 20 D4             BRA     $D1EF           ;

D21B: 10 9E 0D          LDY     <nextDemoCommand ;
D21E: E6 A0             LDB     ,Y+
D220: 2A 07             BPL     $D229           ;
D222: 3F                SWI                     ; Wait for 1.35 seconds
D223: 10                                        ; SWI_10:Pause for 1.35 seconds:
D224: 3F                SWI                     ; Another 1.35 seconds (total 2.7 seconds)
D225: 10                                        ; SWI_10:Pause for 1.35 seconds:
D226: 7E C0 00          JMP     $C000           ; Restart

D229: AE A1             LDX     ,Y++
D22B: CE 03 61          LDU     #$0361
D22E: 3F                SWI
D22F: 06                                        ; SWI_6:Uncompress message X to given buffer U:
D230: 33 41             LEAU    1,U
D232: 3F                SWI                     ; Wait for 1.35 seconds
D233: 10                                        ; SWI_10:Pause for 1.35 seconds:
D234: 8C                ; CMPX  opcode to skip next instruction
D235: 8D 15             BSR     ???
D237: A6 C0             LDA     ,U+
D239: 2A FA             BPL     $D235           ;
D23B: 4F                CLRA
D23C: 8D 0E             BSR     $D24C           ;
D23E: 5A                DECB
D23F: 26 E8             BNE     $D229           ;
D241: 86 1F             LDA     #$1F
D243: 8D 07             BSR     $D24C           ;
D245: 10 9F 0D          STY     <nextDemoCommand ;
D248: CC 01 02          LDD     #$0102
D24B: 39                RTS

D24C: 34 76             PSHS    U,Y,X,B,A       ; Hold these
D24E: 0D AD             TST     <scrollShowing  ; Is a scroll showing?
D250: 26 04             BNE     $D256           ; No ... skip closing the scroll
```

```
D252: 3F                SWI                     ; Bring up normal display
D253: 19                                        ; SWI_19:Bring up normal display:
D254: 3F                SWI                     ; Show the command prompt
D255: 0F                                        ; SWI_F:Ready command prompt:
D256: DE 11             LDU     <m0211          ;
D258: 81 1F             CMPA    #$1F            ; User press ENTER?
D25A: 27 13             BEQ     $D26F           ; Yes ... go process the input
D25C: 81 24             CMPA    #$24            ; A backspace?
D25E: 27 1D             BEQ     $D27D           ; Yes ... handle it
D260: 3F                SWI
D261: 04                                        ; SWI_4:Display a single character in A:
D262: A7 C0             STA     ,U+             ; Put this character in the buffer
D264: 8E C6 7C          LDX     #$C67C          ; Cursor data
D267: 3F                SWI                     ; Print cursor
D268: 03                                        ; SWI_3:Display uncompressed message pointed to
D269: 11 83 03 11       CMPU    #$0311          ; Have we reached the 32 char limit?
D26D: 26 45             BNE     $D2B4           ; No ... keep waiting for ENTER
D26F: 4F                CLRA                    ; Print a ...
D270: 3F                SWI                     ; ... space on the end of the line
D271: 04                                        ; SWI_4:Display a single character in A:
D272: DC 03             LDD     <CONST_FF       ;
D274: ED C1             STD     ,U++            ; End mark on the end of the buffer
D276: CE 02 F1          LDU     #$02F1          ; Reset input parse ...
D279: DF 11             STU     <m0211          ; ... pointer to the beginning of the input
D27B: 20 15             BRA     $D292           ; Execute the command
;
; Backspace
D27D: 11 83 02 F1       CMPU    #$02F1          ; Already at the beginning?
D281: 27 31             BEQ     $D2B4           ; Yes ... ignore it
D283: 33 5F             LEAU    -1,U            ; No ... back up one
D285: 8E D2 8C          LDX     #$D28C          ; Back cursor ...
D288: 3F                SWI                     ; ... up one spot
D289: 03                                        ; SWI_3:Display uncompressed message pointed to
D28A: 20 28             BRA     $D2B4           ; Out

D28C: 00 24 24 1C 24 FF   ; BACK BACK "_" BACK END

; Execute the command
D292: 8E D8 94          LDX     #$D894          ; Command first words
D295: BD CB EC          JSR     DecodeInput     ; Check for word match
D298: 27 0D             BEQ     $D2A7           ; Blank input ... skip execution
```

```
D29A: 2A 05             BPL     $D2A1           ; Good command word ... execute the command
D29C: BD CB E1          JSR     $CBE1           ; Print three question marks (error)
D29F: 20 06             BRA     $D2A7           ; Continue ... skip execution
;
D2A1: 48                ASLA                    ; 2 bytes per pointer
D2A2: 8E D9 D0          LDX     #$D9D0          ; Pointer to command functions
D2A5: AD 96             JSR     [A,X]           ; Execute the command
D2A7: CE 02 F1          LDU     #$02F1          ; Start of input buffer
D2AA: 0D AD             TST     <scrollShowing  ; Is a scroll showing?
D2AC: 27 06             BEQ     $D2B4           ; Yes ... skip the command prompt
D2AE: 0D 28             TST     <fainting       ; Are we fainting?
D2B0: 26 02             BNE     $D2B4           ; Yes ... skip the prompt
D2B2: 3F                SWI                     ; Print the command prompt
D2B3: 0F                                        ; SWI_F:Ready command prompt:
D2B4: DF 11             STU     <m0211          ; New start of command
D2B6: 35 F6             PULS    A,B,X,Y,U,PC    ; Done
```

ATTACK command

```
CmdATTACK:
D2B8: BD CC 31          JSR     GetUserHand     ;
D2BB: EE C4             LDU     ,U
D2BD: 26 03             BNE     $D2C2           ;
D2BF: CE 0B 07          LDU     #$0B07
D2C2: 1F 32             TFR     U,Y
D2C4: A6 4C             LDA     12,U
D2C6: 97 19             STA     <m0219          ;
D2C8: A6 4D             LDA     13,U
D2CA: 97 1B             STA     <m021B          ;
D2CC: 9B 19             ADDA    <m0219          ;
D2CE: 46                RORA
D2CF: 44                LSRA
D2D0: 44                LSRA
D2D1: 9E 17             LDX     <pStrength      ; Strength
D2D3: BD D4 36          JSR     $D436           ;
D2D6: D3 21             ADDD    <m0221          ;
D2D8: DD 21             STD     <m0221          ;
D2DA: A6 4A             LDA     10,U            ; Object class
D2DC: 8B 0C             ADDA    #$0C            ; Sound table offset
```

```
D2DE: C6 FF             LDB     #$FF            ; Full volume
D2E0: 3F                SWI                     ; Play sound of object
D2E1: 1C                                        ; SWI_1C:Play sound A at volume B:
D2E2: A6 49             LDA     9,U             ; Proper name
D2E4: 81 13             CMPA    #$13            ; Ring range?
D2E6: 2D 0F             BLT     $D2F7           ; Too low
D2E8: 81 15             CMPA    #$15            ; Ring Range?
D2EA: 2E 0B             BGT     $D2F7           ; Too high
D2EC: 6A 46             DEC     6,U             ; Subtract one from ring counter
D2EE: 26 07             BNE     $D2F7           ; Still good, go on
D2F0: 86 16             LDA     #$16            ; GOLD token
D2F2: A7 49             STA     9,U             ; Now a gold ring
D2F4: BD D6 38          JSR     $D638           ; Change object
D2F7: DC 13             LDD     <playerY        ; Our coordinates
D2F9: BD CF 82          JSR     GetCreatureAt   ; Find creature
D2FC: 27 77             BEQ     $D375           ; None found ignore attack
D2FE: CE 02 17          LDU     #$0217
D301: 1E 13             EXG     X,U
D303: A6 2A             LDA     10,Y            ; Class
D305: 81 01             CMPA    #$01            ; Is it a ring?
D307: 27 16             BEQ     $D31F           ; Yes-- can't miss
D309: BD D3 D7          JSR     $D3D7           ; Otherwise get calculate hit chance
D30C: 2B 67             BMI     $D375           ; Oops, missed
D30E: 10 9E 24          LDY     <torchPtr       ; Torch pointer
D311: 27 06             BEQ     $D319           ; Oh, no, no torch...
D313: A6 29             LDA     9,Y             ; Proper name of torch
D315: 81 18             CMPA    #$18            ; Well, it is dead!
D317: 26 06             BNE     $D31F           ; No, go on
D319: 3F                SWI                     ; Random number
D31A: 07                                        ; SWI_7:Get random number:
D31B: 84 03             ANDA    #$03            ; Between 0 and 3
D31D: 26 56             BNE     $D375           ; Only one in three will hit
D31F: 3F                SWI                     ; Sound a hit
D320: 1B                                        ; SWI_1B:Play sound i at full volume:
D321: 12                                        ; 12 = Player hit
D322: 3F                SWI                     ; Print !!!
D323: 02                                        ; SWI_2:Uncompress message m and display:
;
D324: 16 F7 B0 ; "!!!"
;
D327: BD D4 0C          JSR     $D40C           ;
```

```
D32A: 22 49              BHI     $D375           ;
D32C: 30 48              LEAX    8,U             ; First in object link for monster
D32E: AE 84              LDX     ,X              ; Get pointer to next object in list
D330: 27 08              BEQ     $D33A           ; That's all
D332: 6F 05              CLR     5,X             ; Drop on floor
D334: EC 4F              LDD     15,U            ; Coordinate from monster
D336: ED 02              STD     2,X             ; Now to object
D338: 20 F4              BRA     $D32E           ; Keep going for all
D33A: 9E 82              LDX     <m0282          ;
D33C: E6 4D              LDB     13,U
D33E: 6A 85              DEC     B,X
D340: 6F 4C              CLR     12,U            ; Monster dead
D342: 3F                 SWI
D343: 0E                                         ; SWI_E:Display playing screen:
D344: 3F                 SWI                     ; Sound monster dead
D345: 1B                                         ; SWI_1B:Play sound i at full volume:
D346: 15                                         ; 15 = Creature dying
D347: EC C4              LDD     ,U              ; Monster strength ...
D349: 8D 34              BSR     DRight3         ; ... divided by 8
D34B: D3 17              ADDD    <pStrength      ; Add to our strength
D34D: 2A 02              BPL     $D351           ; No overflow
D34F: 86 7F              LDA     #$7F            ; Maximum positive value
D351: DD 17              STD     <pStrength      ; New strength
D353: A6 4D              LDA     13,U            ; What did we just kill?
D355: 81 0A              CMPA    #$0A            ; Demon...
D357: 27 2D              BEQ     $D386           ; ...go to level 4
D359: 81 0B              CMPA    #$0B            ; Killed the Wizard?
D35B: 26 18              BNE     $D375           ; No. It was a normal creature. Done.
D35D: 0A 2B              DEC     <wizardDead     ; Wizard is dead (creatures don't move)
D35F: CC 07 13           LDD     #$0713          ; Light level
D362: DD 26              STD     <m0226          ; Ambient light
D364: 8E 0B 23           LDX     #$0B23          ; Drop all but ...
D367: 9F 0F              STX     <nextObjSlot    ; ... first object (the Supreme Ring)
D369: DC 00              LDD     <CONST_00       ; Zero constant
D36B: DD 29              STD     <firstPackObject ; Nothing in pack
D36D: DD 24              STD     <torchPtr       ; No torch
D36F: DD 1F              STD     <rightHand      ; Left hand empty
D371: DD 1D              STD     <leftHand       ; Right hand empty
D373: 3F                 SWI                     ; Draw the display
D374: 19                                         ; SWI_19:Bring up normal display:
D375: 3F                 SWI                     ; Update the heart rate (might have died)
```

```
D376: 0C                                    ; SWI_C:Update heart rate:

; Shift D right routine entries
DRight7:
D377: 47                ASRA                ; 7 (divide by 128)
D378: 56                RORB
DRight6:
D379: 47                ASRA                ; 6 (divide by 64)
D37A: 56                RORB
DRight5:
D37B: 47                ASRA                ; 5 (divide by 32)
D37C: 56                RORB
DRight4:
D37D: 47                ASRA                ; 4 (divide by 16)
D37E: 56                RORB
DRight3:
D37F: 47                ASRA                ; 3 (divide by 8)
D380: 56                RORB
DRight2:
D381: 47                ASRA                ; 2 (divide by 4)
D382: 56                RORB
DRight1:
D383: 47                ASRA                ; 1 (divide by 2)
D384: 56                RORB
D385: 39                RTS                 ; Done

; Demon killed

D386: 8E DF 10          LDX     #$DF10      ; Wizard picture
D389: 3F                SWI                 ; Beam him onto the screen
D38A: 13                                    ; SWI_13:Beam on picture pointed to by X:
D38B: 3F                SWI                 ; Print first part of message
D38C: 02                                    ; SWI_2:Uncompress message m and display:
;
D38D: FF C0 57 3E A7 46 C0 90 51 32 28 1E 60 51 09 98 20 C0 E7 DE F0 ; "_1F_ ENOUGH! I TIRE OF THIS
;
D3A2: 3F                SWI                 ; Print second part of message
D3A3: 02                                    ; SWI_2:Uncompress message m and display:
;
D3A4: E8 00 08 48 B0 0C 8A 0A 3C 0D 29 68 0A 23 20 23 DE DD EF 60 ; "   PREPARE TO MEET THY DOOM!!!'
;
```

```
D3B8: 3F                SWI                     ; Pause for 1.35 seconds
D3B9: 10                                        ; SWI_10:Pause for 1.35 seconds:
D3BA: DE 24             LDU     <torchPtr       ; Keep only the ...
D3BC: DF 29             STU     <firstPackObject ; ... torch in the pack
D3BE: 27 04             BEQ     $D3C4           ; There was no torch ... no need to unlink it
D3C0: 4F                CLRA                    ; Clear any ...
D3C1: 5F                CLRB                    ; ... object chained ...
D3C2: ED C4             STD     ,U              ; ... after torch in pack
D3C4: CC 00 C8          LDD     #$00C8          ; Drop a big ...
D3C7: DD 15             STD     <playerWeight   ; ... strain on the user
D3C9: 86 03             LDA     #$03            ; Level 4
D3CB: 3F                SWI                     ; Prepare level
D3CC: 1A                                        ; SWI_1A:Set up level:
D3CD: BD CF 97          JSR     GetRandCell     ; Get random coordinates
D3D0: DD 13             STD     <playerY        ; New coordinates
D3D2: 3F                SWI                     ; Beam off the wizard
D3D3: 15                                        ; SWI_15:Beam subroutine:
D3D4: 3F                SWI                     ; Beam on the wizard
D3D5: 19                                        ; SWI_19:Bring up normal display:
D3D6: 39                RTS                     ; Done

D3D7: 34 56             PSHS    U,X,B,A
D3D9: 86 0F             LDA     #$0F
D3DB: 97 C1             STA     <holdHole       ;
D3DD: EC C4             LDD     ,U
D3DF: A3 4A             SUBD    10,U
D3E1: BD CA 12          JSR     Dleft2          ;
D3E4: A3 84             SUBD    ,X
D3E6: 25 04             BCS     $D3EC           ;
D3E8: 0A C1             DEC     <holdHole       ;
D3EA: 26 F8             BNE     $D3E4           ;
D3EC: D6 C1             LDB     <holdHole       ;
D3EE: C0 03             SUBB    #$03
D3F0: 2A 09             BPL     $D3FB           ;
D3F2: 50                NEGB
D3F3: 86 19             LDA     #$19
D3F5: 3D                MUL
D3F6: BD CA 99          JSR     $CA99           ;
D3F9: 20 03             BRA     $D3FE           ;
D3FB: 86 0A             LDA     #$0A
D3FD: 3D                MUL
```

```
D3FE: ED E3             STD     ,--S
D400: 3F                SWI
D401: 07                                        ; SWI_7:Get random number:
D402: 1F 89             TFR     A,B
D404: 4F                CLRA
D405: E3 E1             ADDD    ,S++
D407: 83 00 7F          SUBD    #$007F
D40A: 35 D6             PULS    A,B,X,U,PC
D40C: 34 76             PSHS    U,Y,X,B,A
D40E: 1F 12             TFR     X,Y
D410: AE A4             LDX     ,Y
D412: A6 22             LDA     2,Y
D414: 8D 20             BSR     $D436           ;
D416: 1F 01             TFR     D,X
D418: A6 43             LDA     3,U
D41A: 8D 1A             BSR     $D436           ;
D41C: E3 4A             ADDD    10,U
D41E: ED 4A             STD     10,U
D420: AE A4             LDX     ,Y
D422: A6 24             LDA     4,Y
D424: 8D 10             BSR     $D436           ;
D426: 1F 01             TFR     D,X
D428: A6 45             LDA     5,U
D42A: 8D 0A             BSR     $D436           ;
D42C: E3 4A             ADDD    10,U
D42E: ED 4A             STD     10,U
D430: AE C4             LDX     ,U
D432: AC 4A             CMPX    10,U
D434: 35 F6             PULS    A,B,X,Y,U,PC
D436: 34 16             PSHS    X,B,A
D438: 0F C1             CLR     <holdHole       ;
D43A: E6 63             LDB     3,S
D43C: 3D                MUL
D43D: DD C2             STD     <m02C2          ;
D43F: A6 E4             LDA     ,S
D441: E6 62             LDB     2,S
D443: 3D                MUL
D444: D3 C1             ADDD    <holdHole       ;
D446: 08 C3             LSL     <m02C3          ;
D448: 59                ROLB
D449: 49                ROLA
```

```
D44A: ED E4             STD     ,S
D44C: 35 96             PULS    A,B,X,PC
```

CLIMB command

```
CmdCLIMB:
D44E: DC 13             LDD     <playerY        ; Player's coordinates
D450: BD CF E1          JSR     ScanForHole     ; Is there a hole here?
D453: 2B 1A             BMI     $D46F           ; No ... error
D455: 97 C1             STA     <holdHole       ; Hold this for a bit
D457: 8E D8 D9          LDX     #$D8D9          ; Second words
D45A: BD CB EC          JSR     DecodeInput     ; Are we going up or down?
D45D: 2F 10             BLE     $D46F           ; Syntax error on second word ... error
D45F: D6 C1             LDB     <holdHole       ; The hole in this room
D461: 81 04             CMPA    #$04            ; Word is "UP" ?
D463: 27 0D             BEQ     $D472           ; Yes ... try that
D465: 81 05             CMPA    #$05            ; Word is "DOWN" ?
D467: 26 06             BNE     $D46F           ; No ... error
D469: 86 01             LDA     #$01            ; +1
D46B: C5 02             BITB    #$02            ; Is there a hole or ladder going down?
D46D: 26 09             BNE     $D478           ; Yes ... change levels (no error)
;
D46F: 7E CB E1          JMP     $CBE1           ; Print three "???" and out
;
D472: 86 FF             LDA     #$FF            ; -1
D474: C1 01             CMPB    #$01            ; Is there a ladder going up?
D476: 26 F7             BNE     $D46F           ; No ... error
;
; Change the level
D478: 3F                SWI                     ; Print "PREPARE"
D479: 16                                        ; SWI_16:Print PREPARE:
D47A: 9B 81             ADDA    <currentLevel   ; Change level (up or down)
D47C: 3F                SWI                     ; Setup the level
D47D: 1A                                        ; SWI_1A:Set up level:
D47E: 3F                SWI                     ; Normal display
D47F: 19                                        ; SWI_19:Bring up normal display:
D480: 39                RTS                     ; Back to command processing
```

EXAMINE command

```
CmdEXAMINE:
D481: 8E D4 95           LDX     #DrawInventory   ; Set the display function ...
D484: 9F B2              STX     <displayFunction ; ... to draw the inventory
D486: 3F                 SWI                      ; Redraw the screen
D487: 0E                                          ; SWI_E:Display playing screen:
D488: 39                 RTS                      ; Done

SetForExamine:
; Clear the scratch screen, set the physical screen, set the "print to desired area" flag.
; Return the examine descriptor (#$0380).
D489: 3F                 SWI                      ; Clear the scratch screen, return pointer to de
D48A: 09                                          ; SWI_9:Clear secondary screen:
D48B: AE C4              LDX     ,U               ; This is the physical start of the scratch scre
D48D: CE 03 80           LDU     #examineStart    ; Descriptor for the "EXAMINE" screen area
D490: AF C4              STX     ,U               ; Point to the off-screen area
D492: 0A B7              DEC     <whereToPrint    ; Printing goes to desired descriptor
D494: 39                 RTS

DrawInventory:
; Function for drawing the inventory screen
D495: 8D F2              BSR     SetForExamine    ; Activate the "examine" area
D497: 0F B6              CLR     <tabOrCR         ; We are at the start of a line
D499: CC 00 0A           LDD     #$000A           ; Start "IN THIS ROOM" ...
D49C: ED 44              STD     4,U              ; ... indented 10 spaces
D49E: 3F                 SWI                      ; Print "IN THIS ROOM"
D49F: 02                                          ; SWI_2:Uncompress message m and display:
;
D4A0: 62 5C 0A 21 33 04 9E F6 FC ; "IN THIS ROOM_1F_"
;
D4A9: DC 13              LDD     <playerY         ; Player's coordinates
D4AB: BD CF 82           JSR     GetCreatureAt    ; Is there a creature here?
D4AE: 27 10              BEQ     $D4C0            ; No ... skip indicator
D4B0: AE 44              LDX     4,U              ; Set the cursor to ...
D4B2: 30 0B              LEAX    11,X             ; ... center "!CREATURE!" ...
D4B4: AF 44              STX     4,U              ; ... on the line
D4B6: 3F                 SWI                      ; Print "!CREATURE!"
D4B7: 02                                          ; SWI_2:Uncompress message m and display:
;
```

```
D4B8: 56 C7 22 86 95 91 77 F0 ; "!CREATURE!_1F_"
;
D4C0: 0F 91             CLR     <restartFind    ;
D4C2: DC 13             LDD     <playerY        ; Player's coordinates
D4C4: BD CF 53          JSR     GetObjectAtCoor ; Find object on floor
D4C7: 27 04             BEQ     $D4CD           ; No more on floor ... move to the pack
D4C9: 8D 3A             BSR     $D505           ; Print object description
D4CB: 20 F5             BRA     $D4C2           ; Keep going
;
D4CD: 0D B6             TST     <tabOrCR        ; At the start of the line?
D4CF: 27 02             BEQ     $D4D3           ; Yes ... no need for a CR
D4D1: 8D 2B             BSR     $D4FE           ; No ... print a CR
D4D3: CC 1B 20          LDD     #$1B20          ; A = character "!", B = count 32
D4D6: 3F                SWI                     ; Print "!"
D4D7: 04                                        ; SWI_4:Display a single character in A:
D4D8: 5A                DECB                    ; Print line ...
D4D9: 26 FB             BNE     $D4D6           ; ... of "!"
D4DB: AE 44             LDX     4,U             ; Skip cursor to ...
D4DD: 30 0C             LEAX    12,X            ; ... center ...
D4DF: AF 44             STX     4,U             ; ... "BACKPACK"
D4E1: 3F                SWI                     ; Print "BACKPACK"
D4E2: 02                                        ; SWI_2:Uncompress message m and display:
;
D4E3: 40 82 35 C0 23 5F C0 ; "BACKPACK_1F_"
;
D4EA: 8E 02 29          LDX   #$0229            ; Head of object list
D4ED: AE 84             LDX   ,X                ; Get next object
D4EF: 27 0A             BEQ   $D4FB             ; All done
D4F1: 9C 24             CMPX  <torchPtr         ; Is this the lit torch?
D4F3: 26 02             BNE   $D4F7             ; No ... leave it
D4F5: 63 46             COM   6,U               ; Flip the color to show the torch is lit (we ch
D4F7: 8D 0C             BSR   $D505             ; Print object description
D4F9: 20 F2             BRA   $D4ED             ; Next object
;
D4FB: 0F B7             CLR   <whereToPrint     ; Printing goes to command line area
D4FD: 39                RTS                     ; Done
;
D4FE: 86 1F             LDA   #$1F              ; Print a ...
D500: 3F                SWI                     ; ... line feed
D501: 04                                        ; SWI_4:Display a single character in A:
D502: 0F B6             CLR     <tabOrCR        ; Next print a tab (note this was already 0)
```

```
D504: 39                RTS                     ; Done

D505: 34 16             PSHS    X,B,A           ; Hold these
D507: BD C6 17          JSR     GetObjDscrpt    ; Unpack the text to make this object descriptio
D50A: 3F                SWI                     ; Print the object description
D50B: 03                                        ; SWI_3:Display uncompressed message pointed to
D50C: 96 2C             LDA     <backgroundColor ; Set the background ...
D50E: A7 46             STA     6,U             ; ... color (we may have flipped it printing the
D510: 03 B6             COM     <tabOrCR        ; Whether to print a tab or CR
D512: 27 0A             BEQ     $D51E           ; 0 = Print the CR
D514: EC 44             LDD     4,U             ; Cursor
D516: C3 00 10          ADDD    #$0010          ; FF = Skip to ...
D519: C4 F0             ANDB    #$F0            ; ... second ...
D51B: ED 44             STD     4,U             ; ... column
D51D: 8C                ; CMPX  opcode to skip next instruction
D51E: 8D DE             BSR     $D4FE           ; Print a line feed
D520: 35 96             PULS    A,B,X,PC        ; Done
```

GET command

```
CmdGET:
D522: 8D 52             BSR     $D576           ; Pointer for requested hand
D524: 26 4D             BNE     $D573           ; Hand isn't empty ... error
D526: BD CB BA          JSR     $CBBA           ; ??
D529: 0F 91             CLR     <restartFind    ; Reset the iterator to the first object
;
D52B: DC 13             LDD     <playerY        ; Players coordinates
D52D: BD CF 53          JSR     GetObjectAtCoor ; Get the next object on this level at this coor
D530: 27 41             BEQ     $D573           ; End of list ... no match
D532: 0D 90             TST     <m0290          ;
D534: 26 06             BNE     $D53C           ; Do proper match
D536: A6 0A             LDA     10,X            ; Class match
D538: 91 8F             CMPA    <holdIncantLen  ;
D53A: 20 04             BRA     $D540           ; Skip proper match
D53C: A6 09             LDA     9,X             ; Proper match
D53E: 91 8E             CMPA    <holdIncantWord ;
D540: 26 E9             BNE     $D52B           ; Try next object
D542: AF C4             STX     ,U              ; Object now in hand
D544: 6C 05             INC     5,X             ; 1 means carried
```

```
D546: E6 0A             LDB     10,X            ; Class
D548: 8E D9 FA          LDX     #ClassWeights   ; Weight table
D54B: E6 85             LDB     B,X             ; Get the weight
D54D: 4F                CLRA                    ; Two byte value (positive value)
D54E: 20 1B             BRA     $D56B           ; Update the player's weight and screen
```

DROP command

```
CmdDROP:
D550: 8D 24             BSR     $D576           ; Get left or right hand object pointer
D552: 27 1F             BEQ     $D573           ; Nothing to drop ... syntax error
D554: 4F                CLRA                    ; Hand ...
D555: 5F                CLRB                    ; ... is now ...
D556: ED C4             STD     ,U              ; ... empty
D558: 6F 05             CLR     5,X             ; 0 = on floor
D55A: DC 13             LDD     <playerY        ; Drop at ...
D55C: ED 02             STD     2,X             ; ... player's position ...
D55E: 96 81             LDA     <currentLevel   ; Object on ...
D560: A7 04             STA     4,X             ; ... player's level
D562: E6 0A             LDB     10,X            ; Object class
D564: 8E D9 FA          LDX     #ClassWeights   ; Weight table
D567: E6 85             LDB     B,X             ; Get the weight of the object
D569: 50                NEGB                    ; Negative (removing the weight)
D56A: 1D                SEX                     ; Two byte value
;
D56B: D3 15             ADDD    <playerWeight   ; Update ...
D56D: DD 15             STD     <playerWeight   ; ... carried weight
D56F: 3F                SWI                     ; Update the heart after the change in load
D570: 0C                                        ; SWI_C:Update heart rate:
D571: 20 44             BRA     $D5B7           ; Update hands and screen and done
;
D573: 7E CB E1          JMP     $CBE1           ;  Print "???" error
;
D576: 7E CC 31          JMP     GetUserHand     ;  Get user-requested hand object
```

STOW command

```
CmdSTOW:
D579: 8D FB             BSR     $D576           ; Get the left or right hand object
D57B: 27 F6             BEQ     $D573           ; Nothing in that hand ... error
D57D: DC 29             LDD     <firstPackObject ; Move this ...
D57F: ED 84             STD     ,X              ; ... object to ...
D581: 9F 29             STX     <firstPackObject ; ... beginning of list
D583: 4F                CLRA                    ; Now ...
D584: 5F                CLRB                    ; ... empty ...
D585: ED C4             STD     ,U              ; ... hand
D587: 20 2E             BRA     $D5B7           ; Update hands, draw screen, done
```

PULL command

```
CmdPULL:
D589: 8D EB             BSR     $D576           ; Get the left or right hand object
D58B: 26 E6             BNE     $D573           ; It isn't empty ... error and out
D58D: BD CB BA          JSR     $CBBA           ;
D590: 8E 02 29          LDX     #firstPackObject ; First pack object
D593: 1F 12             TFR     X,Y             ;
D595: AE 84             LDX     ,X              ;
D597: 27 DA             BEQ     $D573           ; End of list ... object not found ... error
D599: 0D 90             TST     <m0290          ; ?? two words given?
D59B: 26 06             BNE     $D5A3           ;
D59D: A6 0A             LDA     10,X            ; Class type
D59F: 91 8F             CMPA    <holdIncantLen  ;
D5A1: 20 04             BRA     $D5A7           ; Only check the class type
;
D5A3: A6 09             LDA     9,X             ; Proper type
D5A5: 91 8E             CMPA    <holdIncantWord ;
D5A7: 26 EA             BNE     $D593           ; Not a match ... keep looking
D5A9: EC 84             LDD     ,X              ; Pointer to next object
D5AB: ED A4             STD     ,Y              ; Pull the current object out of the object chai
D5AD: AF C4             STX     ,U              ; Object now in hand
D5AF: 4F                CLRA                    ; We might use this to ...
D5B0: 5F                CLRB                    ; ... extinguish the torch
D5B1: 9C 24             CMPX    <torchPtr       ; Did we just pull the lit torch?
D5B3: 26 02             BNE     $D5B7           ; No ... move on
D5B5: DD 24             STD     <torchPtr       ; Clear the lit-torch pointer
;
```

```
D5B7: 3F                SWI                     ; Update the hands
D5B8: 0D                                        ; SWI_D:Print contents of hands on status line:
D5B9: 3F                SWI                     ; Redraw the screen
D5BA: 0E                                        ; SWI_E:Display playing screen:
D5BB: 39                RTS                     ; Done
```

INCANT command

```
CmdINCANT:
D5BC: 8E D8 F3          LDX     #ProperNames    ; Proper names
D5BF: BD CB EC          JSR     DecodeInput     ; Decode the input word
D5C2: 2F 2B             BLE     $D5EF           ; Word not found ... fail silently
D5C4: 0D 7B             TST     <perfectMatch   ;
D5C6: 27 27             BEQ     $D5EF           ; Not an exact word match ... fail silently
D5C8: DD 8E             STD     <holdIncantWord ; Hold this for a second
D5CA: DE 1D             LDU     <leftHand       ; Try object ...
D5CC: 8D 02             BSR     $D5D0           ; ... in left hand
D5CE: DE 1F             LDU     <rightHand      ; Now try object in right
;
D5D0: 27 1D             BEQ     $D5EF           ; No object in this hand ... skip
D5D2: A6 4A             LDA     10,U            ; Is this a ...
D5D4: 81 01             CMPA    #$01            ; ... ring?
D5D6: 26 17             BNE     $D5EF           ; No ... skip
D5D8: A6 47             LDA     7,U             ; Already revealed?
D5DA: 27 13             BEQ     $D5EF           ; Yes ... fail silently
D5DC: 91 8E             CMPA    <holdIncantWord ; Input word matches this ring?
D5DE: 26 0F             BNE     $D5EF           ; No ... skip
D5E0: A7 49             STA     9,U             ; Yes ... this is the new proper name
D5E2: 3F                SWI                     ; Change the ring to the powerful one
D5E3: 18                                        ; SWI_18:Change object to proper name and data:
D5E4: 3F                SWI                     ; Play the ring sound
D5E5: 1B                                        ; SWI_1B:Play sound i at full volume:
D5E6: 0D                                        ; 0D = Ring
D5E7: 3F                SWI                     ; Update the hand display
D5E8: 0D                                        ; SWI_D:Print contents of hands on status line:
D5E9: 6F 47             CLR     7,U             ; Mark as revealed
D5EB: 81 12             CMPA    #$12            ; Did we just incant the "FINAL" ring?
D5ED: 27 01             BEQ     PlayerWins      ; Yes ... player wins the game
D5EF: 39                RTS                     ; Done
```

```
PlayerWins:
D5F0: 8E DF 39           LDX     #$DF39          ; Star Wizard picture
D5F3: 0A 9E              DEC     <m029E          ;
D5F5: 3F                 SWI                     ; Beam on the Star Wizard
D5F6: 13                                         ; SWI_13:Beam on picture pointed to by X:
D5F7: 3F                 SWI                     ; Print the final message
D5F8: 02                                         ; SWI_2:Uncompress message m and display:
;
D5F9: FF C4 54 3D 84 D8 08 59 D1 2E C8 03 70 A6 93 05 10 50 20 2E 20 ; "_1F_BEHOLD! DESTINY AWAITS
;
D60E: 3F                 SWI                     ; More final message
D60F: 02                                         ; SWI_2:Uncompress message m and display:
;
D610: C8 00 00 00 00 03 CC 00 81 C5 B8 2E 9D 06 44 F7 BC ; "        OF A NEW WIZARD..."
;
D621: 20 FE              BRA     $D621           ; Infinite loop
```

REVEAL command

```
CmdREVEAL:
D623: BD CC 31           JSR     GetUserHand     ; Get requested left/right hand object
D626: EE C4              LDU     ,U              ; 0 if there is no object (EMPTY)
D628: 27 14              BEQ     $D63E           ; Empty ... nothing to do
D62A: A6 4B              LDA     11,U            ; Already revealed?
D62C: 27 10              BEQ     $D63E           ; Yes, skip it now
D62E: C6 19              LDB     #$19            ; Base multiplier
D630: 3D                 MUL                     ; Multiply
D631: 10 93 17           CMPD    <pStrength      ; Are we strong enough?
D634: 2E 08              BGT     $D63E           ; No, leave it unrevealed
D636: A6 49              LDA     9,U             ; Get proper name
D638: 3F                 SWI                     ; Change object types
D639: 18                                         ; SWI_18:Change object to proper name and data:
D63A: 6F 4B              CLR     11,U            ; Revealed
D63C: 3F                 SWI                     ; Update the hands line
D63D: 0D                                         ; SWI_D:Print contents of hands on status line:
D63E: 39                 RTS                     ; Done
```

TURN command

```
CmdTURN:
D63F: 8E D8 D9           LDX     #$D8D9          ; Second words
D642: BD CB EC           JSR     DecodeInput     ; Decode the user input
D645: 2F 4C              BLE     $D693           ; Not found ... syntax error
D647: D6 23              LDB     <playerDir      ; Current direction
D649: 81 00              CMPA    #$00            ; Turning LEFT?
D64B: 26 07              BNE     $D654           ; No ... try right
D64D: 5A                 DECB                    ; Decrease direction ... turning CCW
D64E: 8D 1D              BSR     $D66D           ; Wrap direction and draw the screen
D650: 8D 22              BSR     $D674           ; Turn left
D652: 20 15              BRA     $D669           ; Continue
;
D654: 81 01              CMPA    #$01            ; Turning RIGHT?
D656: 26 05              BNE     $D65D           ; No ... try around
D658: 5C                 INCB                    ; Increase direction ... turning CW
D659: 8D 12              BSR     $D66D           ; Wrap direction and draw the screen
D65B: 20 0A              BRA     $D667           ; Do the turn
;
D65D: 81 03              CMPA    #$03            ; Turning AROUND?
D65F: 26 32              BNE     $D693           ; No ... syntax error
D661: CB 02              ADDB    #$02            ; Flip direction
D663: 8D 08              BSR     $D66D           ; Wrap direction and draw the screen
D665: 8D 1D              BSR     $D684           ; Turn right
D667: 8D 1B              BSR     $D684           ; Turn right
D669: 0A B4              DEC     <flipScreens    ;
D66B: 13                 SYNC                    ; Wait for the refresh
D66C: 39                 RTS                     ; Done
;
D66D: C4 03              ANDB    #$03            ; Limit direction (wrap around)
D66F: D7 23              STB     <playerDir      ; New direction
D671: 7E C6 60           JMP     $C660           ; Draw the screen
;
; Turning left (line moves right)
D674: 8D 20              BSR     $D696           ; Prepare the display and draw the horizontal li
D676: 26 0B              BNE     $D683           ; Nothing to display ... abort
D678: CC 00 08           LDD     #$0008          ; Y=0, X=8
D67B: 8D 3D              BSR     DrawTurningLine ; Draw the turning line
D67D: C3 00 20           ADDD    #$0020          ; Move X right
```

```
D680: 4D                TSTA                    ; Keep going till ...
D681: 27 F8             BEQ     $D67B           ; ... we flow off the right side
D683: 39                RTS                     ; Done
;
; Turning right (line moves left)
D684: 8D 10             BSR     $D696           ; Prepare the display and draw the horizontal li
D686: 26 0A             BNE     $D692           ; Nothing to display ... abort
D688: CC 00 F8          LDD     #$00F8          ; Y=0, X=F8
D68B: 8D 2D             BSR     DrawTurningLine ; Draw the vertical line
D68D: 83 00 20          SUBD    #$0020          ; Move X left
D690: 2A F9             BPL     $D68B           ; Keep going till we flow off the left side
D692: 39                RTS                     ; Done
;
D693: 7E CB E1          JMP     $CBE1           ; Print "???" syntax error

D696: DE B2             LDU     <displayFunction ; Are we showing ...
D698: 11 83 CE 66       CMPU    #NormalDisplay  ; ... the normal game screen?
D69C: 26 1B             BNE     $D6B9           ; No, then out. Nothing to display in EXAMINE mo
D69E: 8E 80 80          LDX     #$8080          ; ?? Zoom ?
D6A1: 9F 4F             STX     <m024F          ; ?? Zoom ?
D6A3: 0F 8B             CLR     <m028B          ;
D6A5: 3F                SWI                     ; Set the light level to animate turning
D6A6: 00                                        ; SWI_0:Light level:
D6A7: 3F                SWI                     ; Clear the screen
D6A8: 08                                        ; SWI_8:Clear display screen:
D6A9: 8E D6 C6          LDX     #$D6C6          ; Top and bottom lines shown while moving
D6AC: 3F                SWI                     ; Draw the two horizontal lines
D6AD: 01                                        ; SWI_1:Draw picture X on screen:
D6AE: 8E 00 11          LDX     #$0011          ; Top Y coordinate ... 17 (line + 1)
D6B1: 9F 2F             STX     <m022F          ; First Y coordinate
D6B3: 8E 00 87          LDX     #$0087          ; Bottom Y coordinate ... 135 (line -1)
D6B6: 9F 33             STX     <m0233          ; Second Y coordinate
D6B8: 4F                CLRA                    ; Return that we ARE going to draw something
D6B9: 39                RTS                     ; Done

DrawTurningLine:
D6BA: DD 31             STD     <m0231          ; First X coordinate
D6BC: DD 35             STD     <m0235          ; Second X coordinate
D6BE: 8D 00             BSR     $D6C0           ; ?? drawing then erasing? Two different screens
D6C0: BD CA B7          JSR     $CAB7           ; Draw the vertical line
D6C3: 03 2C             COM     <backgroundColor ; ?? two passes ... we set it back
```

```
D6C5: 39                RTS                     ; Done

; Top and bottom horizontal lines while turning
D6C6: 10 00           ; Move to absolute (16,0)
D6C8: 10 FF           ; Line to absolute (255,16)
D6CA: FF              ; New line
D6CB: 88 00           ; Move to absolute (0,136)
D6CD: 88 FF           ; Line to absolute (255,136)
D6CF: FE              ; End
```

MOVE command

```
CmdMOVE:
D6D0: 8E D8 D9          LDX     #SecondWords    ;
D6D3: BD CB EC          JSR     DecodeInput     ; Decode the input word
D6D6: 2D BB             BLT     $D693           ; Bad input ... error
D6D8: 2E 09             BGT     $D6E3           ; Requested direction ... go do it
D6DA: 0A 73             DEC     <m0273          ;  ?? Draw half step magnification?
D6DC: 3F                SWI
D6DD: 0E                                        ; SWI_E:Display playing screen:
D6DE: 5F                CLRB                    ; Moving direction 0 (forward)
D6DF: 0F 73             CLR     <m0273          ; ?? Clear half-step flag
D6E1: 20 0C             BRA     $D6EF           ; Make the move
;
D6E3: 81 02             CMPA    #$02            ; Word is "BACK" ?
D6E5: 26 0C             BNE     $D6F3           ; No ... try others
D6E7: 0A 74             DEC     <m0274          ; ?? back half step mag?
D6E9: 3F                SWI
D6EA: 0E                                        ; SWI_E:Display playing screen:
D6EB: C6 02             LDB     #$02            ; Moving direction 2 (backwards)
D6ED: 0F 74             CLR     <m0274          ; ?? CLear half-step flag?
D6EF: 8D 2F             BSR     $D720           ;
D6F1: 20 1B             BRA     $D70E           ;
;
D6F3: 81 01             CMPA    #$01            ; Word is "RIGHT" ?
D6F5: 26 0A             BNE     $D701           ; No ... try others
D6F7: C6 01             LDB     #$01            ; Moving direction 1 (right)
D6F9: 8D 25             BSR     $D720           ;
D6FB: 26 11             BNE     $D70E           ;
```

```
D6FD: 8D 85             BSR     $D684           ; Line moving left (player moving right)
D6FF: 20 0D             BRA     $D70E           ;

D701: 81 00             CMPA    #$00            ; Word is "LEFT" ?
D703: 26 8E             BNE     $D693           ; No ... error and out
D705: C6 03             LDB     #$03            ; Moving direction 3 (left)
D707: 8D 17             BSR     $D720           ;
D709: 26 03             BNE     $D70E           ;
D70B: BD D6 74          JSR     $D674           ; Line moving right (player moving left)
D70E: DC 15             LDD     <playerWeight   ; Player's weight ...
D710: BD D3 7F          JSR     DRight3         ; ... divided by 8 ...
D713: C3 00 03          ADDD    #$0003          ; ... plus 3
D716: D3 21             ADDD    <m0221          ; Add ...
D718: DD 21             STD     <m0221          ; ... exertion
D71A: 3F                SWI                     ; Update the heart rate
D71B: 0C                                        ; SWI_C:Update heart rate:
D71C: 0A B4             DEC     <flipScreens    ;
D71E: 13                SYNC                    ; Wait for heart rate change
D71F: 39                RTS                     ; Out

D720: 34 06             PSHS    B,A             ; Hold
D722: 6F E2             CLR     ,-S
D724: DB 23             ADDB    <playerDir      ;
D726: C4 03             ANDB    #$03
D728: D7 8A             STB     <drwMazeDir     ;
D72A: DC 13             LDD     <playerY        ;
D72C: BD D1 36          JSR     $D136           ;
D72F: 27 07             BEQ     $D738           ;
D731: 3F                SWI
D732: 1B                                        ; SWI_1B:Play sound i at full volume:
D733: 14                                        ; 14 = Wall hit
D734: 6A E4             DEC     ,S
D736: DC 13             LDD     <playerY        ;
;
D738: DD 13             STD     <playerY        ;
D73A: BD C6 60          JSR     $C660           ;
D73D: 6D E0             TST     ,S+
D73F: 35 86             PULS    A,B,PC
```

USE command

```
CmdUSE:
D741: BD CC 31           JSR     GetUserHand      ; Get requested hand
D744: 27 21              BEQ     $D767            ; Nothing in the hand ... fail silently
D746: EC 09              LDD     9,X              ; Get the object data
D748: C1 05              CMPB    #$05             ; Class is TORCH?
D74A: 26 0B              BNE     $D757            ; No ... try others
D74C: 9F 24              STX     <torchPtr        ; This is our new torch
D74E: BD D5 7D           JSR     $D57D            ; Automatically stow it
D751: 3F                 SWI                      ; Play TORCH sound
D752: 1B                                          ; SWI_1B:Play sound i at full volume:
D753: 11                                          ; 11 = Torch
D754: 3F                 SWI                      ; Update the display
D755: 0E                                          ; SWI_E:Display playing screen:
D756: 39                 RTS                      ; Done
;
D757: 1F 13              TFR     X,U              ; Object pointer now to U
D759: 8E D7 6B           LDX     #UseFunctions    ; USE routines per proper name
D75C: A1 84              CMPA    ,X               ; Is this our object's routine?
D75E: 27 08              BEQ     $D768            ; Yes ... do it
D760: 30 03              LEAX    3,X              ; Next in table
D762: 8C D7 7A           CMPX    #$D77A           ; Tried them all?
D765: 25 F5              BCS     $D75C            ; No ... keep looking
D767: 39                 RTS                      ; No routine ... fail silently
;
D768: 6E 98 01           JMP     [$01,X]          ; Execute the routine

UseFunctions:
D76B: 05          ; THEWS flask
D76C: D7 7A       ; Routine address
;
D76E: 09          ; HALE flask
D76F: D7 83       ; Routine address
;
D771: 08          ; ABYE flask
D772: D7 87       ; Routine address
;
D774: 04          ; SEER scroll
D775: D7 A2       ; Routine address
;
D777: 07          ; VISION scroll
```

```
D778: D7 A0       ; Routine address

UseTHEWS:
D77A: CC 03 E8            LDD     #$03E8          ; Big increase in strength
D77D: D3 17               ADDD    <pStrength      ; Update ...
D77F: DD 17               STD     <pStrength      ; ... player strength
D781: 20 0F               BRA     $D792           ; Make the FLASK sound

UseHALE:
D783: 4F                  CLRA                    ; Clear all ...
D784: 5F                  CLRB                    ; ... exertion
D785: 20 09               BRA     $D790           ; Update and make the FLASK sound

UseABYE:
D787: 9E 17               LDX     <pStrength      ; Strength ...
D789: 86 66               LDA     #$66            ; ... times $66 ...
D78B: BD D4 36            JSR     $D436           ; ... is new ...
D78E: D3 21               ADDD    <m0221          ; ... exertion
D790: DD 21               STD     <m0221          ; Store new exertion
D792: C6 17               LDB     #$17            ; Now an EMPTY ...
D794: E7 49               STB     9,U             ; ... FLASK
D796: 6F 4B               CLR     11,U            ; We revealed it by drinking it
D798: 3F                  SWI                     ; Play the FLASH sound
D799: 1B                                          ; SWI_1B:Play sound i at full volume:
D79A: 0C                                          ; 0C = Flask
D79B: 3F                  SWI                     ; Update the hands
D79C: 0D                                          ; SWI_D:Print contents of hands on status line:
D79D: 3F                  SWI                     ; Update the heart rate
D79E: 0C                                          ; SWI_C:Update heart rate:
D79F: 39                  RTS                     ; Done

UseVISION:
D7A0: 4F                  CLRA                    ; 0 means VISION scroll
D7A1: 8C                  ; CMPX  opcode to skip next instruction
UseSEER:
D7A2: 86 FF               LDA     #$FF            ; FF means SEER scroll
D7A4: 97 94               STA     <scrollType     ; set the scroll type
D7A6: 6D 4B               TST     11,U            ; Has this been revealed?
D7A8: 26 0C               BNE     $D7B6           ; No ... fail silently
D7AA: 3F                  SWI                     ; Play open-scroll sound
D7AB: 1B                                          ; SWI_1B:Play sound i at full volume:
```

```
D7AC: 0E                                ; 0E = Scroll
D7AD: 0F AD             CLR     <scrollShowing  ; 0 means scroll is showing
D7AF: 8E CD B2          LDX     #ShowMap        ; Set display to ...
D7B2: 9F B2             STX     <displayFunction ; ... the map display
D7B4: 3F                SWI                     ; Redraw the screen
D7B5: 0E                                        ; SWI_E:Display playing screen:
D7B6: 39                RTS                     ; Done
```

ZLOAD command

```
CmdZLOAD:
D7B7: 8D 03             BSR     $D7BC           ; Parse the next word
D7B9: 0A B8             DEC     <tapeTrigger    ; Trigger ZLOAD
D7BB: 39                RTS                     ; Done

D7BC: 8E 03 13          LDX     #$0313          ; Start of scratch buffer
D7BF: 33 88 20          LEAU    $20,X           ; End is 16 bytes later
D7C2: 3F                SWI                     ; Fill scratch buffer with FFs
D7C3: 12                                        ; SWI_12:Fill X to U with FFs:
D7C4: 7E CB 96          JMP     GetNextWord     ; Parse the next word and return
```

ZSAVE command

```
CmdZSAVE:
D7C7: 8D F3             BSR     $D7BC           ; Parse the next word into the scratch buffer
D7C9: BF 00 7E          STX     CASSPTR         ; This will be the name of the file
D7CC: CC 00 0F          LDD     #$000F          ; Block type 0 (header) length = 16
D7CF: FD 00 7C          STD     CASSBLKTYPE     ; Prepare for first BLKOUT
D7D2: 0C B8             INC     <tapeTrigger    ; Trigger ZSAVE
D7D4: 39                RTS                     ; Done

; ############################################## Data from here down ##############################

;Initial backpack objects
;
DemoObjects:
D7D5: 0D ; Iron sword     (Elvish = 02)
D7D6: 0F ; Pine torch     (Solar  = 0A)
```

```
D7D7: 10 ; Leather shield (Seer   = 04)
D7D8: FF ; End of list
;
GameObjects:
D7D9: 11 ; Wooden sword
D7DA: 0F ; Pine torch
D7DB: FF ; End of list

; ??Game Task List??
D7DC: D1 EB ;
D7DE: D1 C2 ;
D7E0: D1 D5 ;
D7E2: D1 9B ;
D7E4: D0 27 ;
D7E6: 00 00 ; End of task list

InitCopyTable:
D7E8: 0C 01 03 ; Copy 0C bytes to 103
D7EB: 7E C3 71   ; SWI2 Vector: JMP $C371
D7EE: 7E C3 52   ; SWI  Vector: JMP $C352
D7F1: 7E C2 7D   ; IRQ  Vector: JMP $C27D
D7F4: 7E C2 7D   ; FIRQ Vector: JMP $C27D
;
; Initialize a few local variables
D7F7: 17 02 02 ; Copy 17 bytes to 202
D7FA: 01       ; 202    Constant 1 (two bytes from 201)
D7FB: FF FF    ; 203:204 Constant FFFF
D7FD: 00 80    ; 205:206 ??
D7FF: 00 4C    ; 207:208 ??
D801: D8 70    ; 209:20A Pointer to visible screen descriptor
D803: D8 76    ; 20B:20C Pointer to drawing screen descriptor
D805: D9 88    ; 20D:20E Pointer to demo commands
D807: 0B 15    ; 20F:210 Pointer to next available game object slot
D809: 02 F1    ; 211 Next input to parse
D80B: 0C 16    ; 213:214 Player starting point (for demo)
D80D: 00 23    ; 215:216 Player weight
D80F: 17 A0    ; 217:218 Player strength

; Text descriptors   ??
D811: 54 03 80 ; Copy 54 bytes to 380
D814: 10 00    ; Starts at $1000
```

```
D816: 02 60
D818: 00 00
D81A: 00 FF
;
; $388
D81C: 23 00    ; Starts at $2300
D81E: 00 40
D820: 00 00
D822: FF 00
;
; $390
D824: 24 00    ; Starts at $2400
D826: 00 80
D828: 00 00
D82A: 00 00

CreaturesOnLevels:
; 0398
;1
D82C: 09 09 04 02 ; 9 spiders, 9 snakes, 4 giants, 2 blobs
D830: 00 00 00 00 ; 0 knights, 0 hatchet-giants, 0 scorpions, 0 shield-knights
D834: 00 00 00 00 ; 0 wraiths, 0 galdrogs, 0 demons, 0 wizards
;
;2
D838: 02 04 00 06 ; 2 spiders, 4 snakes, 0 giants, 6 blobs
D83C: 06 06 00 00 ; 6 knights, 6 hatchet-giants
D840: 00 00 00 00 ; 0 wraiths, 0 galdrogs, 0 demons, 0 wizards
;
;3
D844: 00 00 00 04 ; 0 spiders, 0 snakes, 0 giants, 4 blobs
D848: 00 06 08 04 ; 0 knights, 6 hatchet-giants, 8 scorpions, 4 shield-knights
D84C: 00 00 01 00 ; 0 wraiths, 0 galdrogs, 1 demons, 0 wizards
;
;4
D850: 00 00 00 00 ; 0 spiders, 0 snakes, 0 giants, 0 blobs
D854: 00 00 08 06 ; 0 knights, 0 hatchet-giants, 8 scorpions, 6 shield-knights
D858: 06 04 00 00 ; 6 wraiths, 4 galdrogs, 0 demons, 0 wizards
;
;5
D85C: 02 02 02 02 ; 2 spiders, 2 snakes, 2 giants, 2 blobs
D860: 02 02 02 04 ; 2 knights, 2 hatchet-giants, 2 scorpions, 4 shield-knights
```

```
D864: 04 08 00 01 ; 4 wraiths, 8 galdrogs, 0 demons, 1 wizards

D868: 04 0B 11 ; Copy 4 bytes to B11
; ?? Last game task
D86B: 04 00 00 05

D86F: 00                                    ; End of initialization copy list
```

Screen Descriptors

There are two graphics pages. One is displayed while the other is being drawn on. RAM 0x209 points to the visible screen. RAM 0x20B points to the drawing screen. To flip the pages the pointers are swapped.

Each screen is divided into three sections (3*2 = 6 descriptors below):

- Play field
- Hands/heart
- 4 lines of text

```
ScreenDescriptors:
D870: 10 00 ; Start of upper play-field 0x1000
D872: 23 00 ; End of play-field (152 rows)
D874: 20 46 ; SAM value 0010000__001000_110 (G6R,G6C Display=8*512=0x1000)
;
D876: 28 00 ; Start of upper play-field 0x2800
D878: 3B 00 ; End of play-field (152 rows)
D87A: 20 A6 ; SAM value 0010000__010100_110 (G6R,G6C Display=20*512=0x2800)

D87C: 23 00 ; Start of hand/heart row
D87E: 24 00 ; End of hand/heart row (8 rows)
D880: 00 00
;
D882: 3B 00 ; Start of hand/heart row
D884: 3C 00 ; End of hand/heart row (8 rows)
D886: 00 00

D888: 24 00 ; Start of text lines
D88A: 28 00 ; End of text lines (4*8 rows)
D88C: 00 00
;
D88E: 3C 00 ; Start of text lines
```

```
D890: 40 00 ; End of text lines (4*8 rows)
D892: 00 00
```

First Words

```
; FirstWords:
D894: 0F
D895: 30 03 4A 04 6B    ; 00 "ATTACK"
D89A: 28 06 C4 B4 40    ; 01 "CLIMB"
D89F: 20 09 27 C0       ; 02 "DROP"
D8A3: 38 0B 80 B5 2E 28 ; 03 "EXAMINE"
D8A9: 18 0E 5A 00       ; 04 "GET"
D8AD: 30 12 E1 85 D4    ; 05 "INCANT"
D8B2: 20 18 F7 AC       ; 06 "LOOK"
D8B6: 20 1A FB 14       ; 07 "MOVE"
D8BA: 20 21 56 30       ; 08 "PULL"
D8BE: 30 24 5B 14 2C    ; 09 "REVEAL"
D8C3: 20 27 47 DC       ; 0A "STOW"
D8C7: 20 29 59 38       ; 0B "TURN"
D8CB: 18 2B 32 80       ; 0C "USE"
D8CF: 28 34 C7 84 80    ; 0D "ZLOAD"
D8D4: 28 35 30 D8 A0    ; 0E "ZSAVE"
```

Second Words

```
SecondWords:
D8D9: 06
D8DA: 20 18 53 50      ; 00 "LEFT"
D8DE: 28 24 93 A2 80   ; 01 "RIGHT"
D8E3: 20 04 11 AC      ; 02 "BACK"
D8E7: 30 03 27 D5 C4   ; 03 "AROUND"
D8EC: 10 2B 00         ; 04 "UP"
D8EF: 20 08 FB B8      ; 05 "DOWN"
```

Proper Names

```
ProperNames:
;                           index class text
D8F3: 19
D8F4: 38 67 58 48 AD 28   ;  00:  01   "SUPREME"
D8FA: 28 54 FA B0 A0      ;  01:  01   "JOULE"
D8FF: 31 0A CB 26 68      ;  02:  04   "ELVISH"
D904: 38 DA 9A 22 49 60   ;  03:  03   "MITHRIL"
D90A: 20 A6 52 C8         ;  04:  02   "SEER"
D90E: 28 28 82 DE 60      ;  05:  00   "THEWS"
D913: 20 64 96 94         ;  06:  01   "RIME"
D917: 30 AC 99 A5 EE      ;  07:  02   "VISION"
D91C: 20 02 2C 94         ;  08:  00   "ABYE"
D920: 20 10 16 14         ;  09:  00   "HALE"
D924: 29 66 F6 06 40      ;  0A:  05   "SOLAR"
D929: 30 C5 27 BB 45      ;  0B:  03   "BRONZE"
D92E: 30 6D 56 0C 2E      ;  0C:  01   "VULCAN"
D933: 21 13 27 B8         ;  0D:  04   "IRON"
D937: 29 59 57 06 40      ;  0E:  05   "LUNAR"
D93C: 21 60 97 14         ;  0F:  05   "PINE"
D940: 38 D8 50 D1 05 90   ;  10:  03   "LEATHER"
D946: 31 2E F7 90 AE      ;  11:  04   "WOODEN"
D94B: 28 4C 97 05 80      ;  12:  01   "FINAL"
D950: 30 4A E2 C8 F9      ;  13:  01   "ENERGY"
D955: 18 52 32 80         ;  14:  01   "ICE"
D959: 20 4C 99 14         ;  15:  01   "FIRE"
D95D: 20 4E F6 10         ;  16:  01   "GOLD"
D961: 28 0A D8 53 20      ;  17:  00   "EMPTY"
D966: 21 48 50 90         ;  18:  05   "DEAD"
```

Class Names

```
ClassNames:
D96A: 06
D96B: 28 0C C0 CD 60  ; 00 "FLASK"
D970: 20 64 97 1C     ; 01 "RING"
D974: 30 A6 39 3D 8C  ; 02 "SCROLL"
D979: 30 E6 84 95 84  ; 03 "SHIELD"
D97E: 29 27 77 C8 80  ; 04 "SWORD"
D983: 29 68 F9 0D 00  ; 05 "TORCH"
```

```
DemoCommands:
D988: 01          ; one word
D989: D8 A3       ; EXAMINE
;
D98B: 03          ; three words
D98C: D8 BA       ; PULL
D98E: D8 DE       ; RIGHT
D990: D9 83       ; TORCH
;
D992: 02          ; two words
D993: D8 CB       ; USE
D995: D8 DE       ; RIGHT
;
D997: 01          ; one word
D998: D8 B2       ; LOOK
;
D99A: 01          ; one word
D99B: D8 B6       ; MOVE
;
D99D: 03          ; three words
D99E: D8 BA       ; PULL
D9A0: D8 DA       ; LEFT
D9A2: D9 79       ; SHIELD
;
D9A4: 03          ; three words
D9A5: D8 BA       ; PULL
D9A7: D8 DE       ; RIGHT
D9A9: D9 7E       ; SWORD
;
D9AB: 01          ; one word
D9AC: D8 B6       ; MOVE
;
D9AE: 01          ; one word
D9AF: D8 B6       ; MOVE
;
D9B1: 02          ; two words
D9B2: D8 95       ; ATTACK
D9B4: D8 DE       ; RIGHT
;
D9B6: 02          ; two words
```

```
D9B7: D8 C7        ; TURN
D9B9: D8 DE        ; RIGHT
;
D9BB: 01           ; one word
D9BC: D8 B6        ; MOVE
;
D9BE: 01           ; one word
D9BF: D8 B6        ; MOVE
;
D9C1: 01           ; one word
D9C2: D8 B6        ; MOVE
;
D9C4: 02           ; two words
D9C5: D8 C7        ; TURN
D9C7: D8 DE        ; RIGHT
;
D9C9: 01           ; one word
D9CA: D8 B6        ; MOVE
;
D9CC: 01           ; one word
D9CD: D8 B6        ; MOVE
;
D9CF: FF           ; End of list

; Command function jump table
CommandTable:
D9D0: D2 B8 ; 00 ATTACK
D9D2: D4 4E ; 01 CLIMB
D9D4: D5 50 ; 02 DROP
D9D6: D4 81 ; 03 EXAMINE
D9D8: D5 22 ; 04 GET
D9DA: D5 BC ; 05 INCANT
D9DC: C7 51 ; 06 LOOK
D9DE: D6 D0 ; 07 MOVE
D9E0: D5 89 ; 08 PULL
D9E2: D6 23 ; 09 REVEAL
D9E4: D5 79 ; 0A STOW
D9E6: D6 3F ; 0B TURN
D9E8: D7 41 ; 0C USE
D9EA: D7 B7 ; 0D ZLOAD
D9EC: D7 C7 ; 0E ZSAVE
```

```
; Object pictures by class
ClassPictures:
D9EE: DC 19 ; Flask
D9F0: DC 21 ; Ring
D9F2: DC 2A ; Scroll
D9F4: DB FA ; Shield
D9F6: DC 0F ; Sword
D9F8: DC 07 ; Torch

; Object weights by class
ClassWeights:
D9FA: 05    ; Flask
D9FB: 01    ; Ring
D9FC: 0A    ; Scroll
D9FD: 19    ; Shield
D9FE: 19    ; Sword
D9FF: 0A    ; Torch

ObjectData:
; Object descriptors by object index
;     CC RR MM PP: Class, Reveal, Magic Attack, Physical Attack
;                  nn Class   Proper
DA00: 01 FF 00 05 ; 00 Ring    Supreme
DA04: 01 AA 00 05 ; 01 Ring    Joule
DA08: 04 96 40 40 ; 02 Sword   Elvish
DA0C: 03 8C 0D 1A ; 03 Shield  Mithril
DA10: 02 82 00 05 ; 04 Scroll  Seer
DA14: 00 46 00 05 ; 05 Flask   Thews
DA18: 01 34 00 05 ; 06 Ring    Rime
DA1C: 02 32 00 05 ; 07 Scroll  Vision
DA20: 00 30 00 05 ; 08 Flask   Abye
DA24: 00 28 00 05 ; 09 Flask   Hale
DA28: 05 46 00 05 ; 0A Torch   Solar
DA2C: 03 19 00 1A ; 0B Shield  Bronze
DA30: 01 0D 00 05 ; 0C Ring    Vulcan
DA34: 04 0D 00 28 ; 0D Sword   Iron
DA38: 05 19 00 05 ; 0E Torch   Lunar
DA3C: 05 05 00 05 ; 0F Torch   Pine
DA40: 03 05 00 0A ; 10 Shield  Leather
DA44: 04 05 00 10 ; 11 Sword   Wooden
```

```
DA48: 01 00 00 00 ; 12 Ring    Final
DA4C: 01 00 FF FF ; 13 Ring    Energy
DA50: 01 00 FF FF ; 14 Ring    Ice
DA54: 01 00 FF FF ; 15 Ring    Fire
DA58: 01 00 00 05 ; 16 Ring    Gold
DA5C: 00 00 00 05 ; 17 Flask   Empty
DA60: 05 05 00 05 ; 18 Torch   Dead

ObjectSpecial:
; Special object properties by proper-index
;
; Thanks to Aaron Oliver for pointing out:
; Note that the physical and magic defense of the shields is swapped. This is a well known
; bug in the code. These values are multipliers. 80 means 1. 40 means 0.5.
;
; In this table the leather and bronze shields have a little reduction for magic and no
; reduction for physical. Since these are purely physical shields, the numbers should be
; reversed. The mithril shield has 0.5 for both physical and magic.
;
DA64: 00 03 12 00 ; Supreme Ring    (proper, strikes, ??, ??)
DA68: 01 03 13 00 ; Joule Ring      (proper, strikes, ??, ??)
DA6C: 03 40 40 00 ; Mithril Shield  (proper, magic defense, physical defense, ??)
DA70: 06 03 14 00 ; Rime Ring       (proper, strikes, ??, ??)
DA74: 0A 3C 0D 0B ; Solar Torch     (proper, minutes, physical illumination, magic illumination)
DA78: 0B 60 80 00 ; Bronze Shield   (proper, magic defense, physical defense, ??)
DA7C: 0C 03 15 00 ; Vulcan Ring     (proper, strikes, ??, ??)
DA80: 0E 1E 0A 04 ; Lunar Torch     (proper, minutes, physical illumination, magic illumination)
DA84: 0F 0F 07 00 ; Pine Torch      (proper, minutes, physical illumination, magic illumination)
DA88: 10 6C 80 00 ; Leather Shield  (proper, magic defense, physical defense, ??)
DA8C: 18 00 00 00 ; Dead Torch      (proper, minutes, physical illumination, magic illumination)
DA90: FF          ; End of list

ObjectDist:
; This table defines how objects are distributed to creatures on the various
; levels. There are 18 types of objects from most powerful to least. Each
; type of object gets one byte ... two nibbles. The first nibble indicates
; where the object first appears. The second nibble indicates how many there
; are. For instance, the ABYE FLASK first appears on level 1. There are 6 of them.
; They are assigned to 1, 2, 3, 4, 5, and then wrapping back around to level 1 again
; for the last one. The most powerful creatures on each level are given the most
; important objects. The Demon on level 2 gets a SEER SCROLL, though you can never
```

```
; pick it up.
;
; Note that the code assigns objects through level 5 even though there isn't a
; level 5. Based on the types of objects assigned to level 5, I believe this is
; a bug in the wrapping code and not a level that got left out.
;
;                                Level   0       1       2       3       4      ?5?
DA91: 41 ; 00 Supreme Ring    1 start 4    --      --      --      --      00      --
DA92: 31 ; 01 Joule Ring      1 start 3    --      --      --      01      --      --
DA93: 31 ; 02 Elvish Sword    1 start 3    --      --      --      02      --      --
DA94: 32 ; 03 Mithril Shield  2 start 3    --      --      --      03      04      --
DA95: 23 ; 04 Seer Scroll     3 start 2    --      --      05      06      07      --
DA96: 23 ; 05 Thews Flask     3 start 2    --      --      08      09      0A      --
DA97: 11 ; 06 Rime Ring       1 start 1    --      0B      --      --      --      --
DA98: 13 ; 07 Vision Scroll   3 start 1    --      0C      0D      0E      --      --
DA99: 16 ; 08 Abye Flask      6 start 1    --      0F 14   10      11      12      13
DA9A: 14 ; 09 Hale Flask      4 start 1    --      15      16      17      18      --
DA9B: 14 ; 0A Solar Torch     4 start 1    --      19      1A      1B      1C      --
DA9C: 16 ; 0B Bronze Shield   6 start 1    --      1D 22   1E      1F      20      21
DA9D: 01 ; 0C Vulcan Ring     1 start 0    23      --      --      --      --      --
DA9E: 04 ; 0D Iron Sword      4 start 0    24      25      26      27      --      --
DA9F: 08 ; 0E Lunar Torch     8 start 0    28 2E   29 2F   2A      2B      2C      2D
DAA0: 08 ; 0F Pine Torch      8 start 0    30 36   31 37   32      33      34      35
DAA1: 03 ; 10 Leather Shield  3 start 0    38      39      3A      --      --      --
DAA2: 04 ; 11 Wooden Sword    4 start 0    3B      3C      3D      3E      --      --
;
; These are added to the backpack after all other objects have been created:
;          11 Wooden Sword       3F
;          0F Pine Torch         40
;
; These are the objects created for the demo:
;          0D Iron Sword         3F
;          0F Pine Torch         40
;          10 Leather Shield     41
```

Check List

TODO put these tables side-by-side

Start With

```
WOODEN SWORD
PINE TORCH
```

Level 0

```
BLOB      VULCAN RING
BLOB      IRON SWORD
CLUB GIANTLUNAR TORCH
CLUB GIANTLUNAR TORCH
CLUB GIANTPINE TORCH
CLUB GIANTPINE TORCH
SNAKE     LEATHER SHIELD
SNAKE     WOODEN SWORD
SNAKE
SNAKE
SNAKE
SNAKE
SNAKE
SNAKE
SNAKE
SPIDER
SPIDER
SPIDER
SPIDER
SPIDER
SPIDER
SPIDER
SPIDER
SPIDER
```

Level 1

```
HATCHET GIANTRIME RING
HATCHET GIANTVISION SCROLL
```

```
HATCHET GIANTABYE FLASK
HATCHET GIANTABYE FLASK
HATCHET GIANTHALE FLASK
HATCHET GIANTSOLAR TORCH
PLAIN KNIGHT BRONZE SHIELD
PLAIN KNIGHT BRONZE SHIELD
PLAIN KNIGHT IRON SWORD
PLAIN KNIGHT LUNAR TORCH
PLAIN KNIGHT LUNAR TORCH
PLAIN KNIGHT PINE TORCH
BLOB         PINE TORCH
BLOB         LEATHER SHIELD
BLOB         WOODEN SWORD
BLOB
BLOB
BLOB
SNAKE
SNAKE
SNAKE
SNAKE
SPIDER
SPIDER
```

Level 2

```
DEMON        SEER SCROLL
SHIELD KNIGHTTHEWES FLASK
SHIELD KNIGHTVISION SCROLL
SHIELD KNIGHTABYE FLASK
SHIELD KNIGHTHALE FLASK
SCORPION     SOLAR TORCH
SCORPION     BRONZE SHIELD
SCORPION     IRON SWORD
SCORPION     LUNAR TORCH
SCORPION     PINE TORCH
SCORPION     LEATHER SHIELD
SCORPION     WOODEN SWORD
```

```
SCORPION
HATCHET GIANT
HATCHET GIANT
HATCHET GIANT
HATCHET GIANT
HATCHET GIANT
HATCHET GIANT
BLOB
BLOB
BLOB
BLOB
```

Level 3

```
GALDROG      JOULE RING
GALDROG      ELVISH SWORD
GALDROG      MITHRIL SHIELD
GALDROG      SEER SCROLL
WRAITH       THEWS FLASK
WRAITH       VISION SCROLL
WRAITH       ABYE FLASK
WRAITH       HALE FLASK
WRAITH       SOLAR TORCH
WRAITH       BRONZE SHIELD
SHIELD KNIGHTIRON SWORD
SHIELD KNIGHTLUNAR TORCH
SHIELD KNIGHTPINE TORCH
SHIELD KNIGHTWOODEN SWORD
SHIELD KNIGHT
SHIELD KNIGHT
SCORPION
SCORPION
SCORPION
SCORPION
SCORPION
SCORPION
SCORPION
```

```
SCORPION
```

Level 4

```
WIZARD        SUPREME RING
GALDROG      MITHRIL SHIELD
GALDROG       SEER SCROLL
GALDROG       THEWS FLASK
GALDROG       ABYE FLASK
GALDROG       HALE FLASK
GALDROG       SOLAR TORCH
GALDROG       BRONZE SHIELD
GALDROG       LUNAR TORCH
WRAITH        PINE TORCH
WRAITH
WRAITH
WRAITH
SHIELD KNIGHT
SHIELD KNIGHT
SHIELD KNIGHT
SHIELD KNIGHT
SCORPION
SCORPION
HATCHET GIANT
HATCHET GIANT
PLAIN KNIGHT
PLAIN KNIGHT
BLOB
BLOB
CLUB GIANT
CLUB GIANT
SNAKE
SNAKE
SPIDER
SPIDER
```

```
CreaturePictures:
DAA3: DE 26 ; Spider
DAA5: DF CA ; Snake
DAA7: DD 41 ; Giant
DAA9: DE 59 ; Blob
DAAB: DE 82 ; Knight
DAAD: DD 51 ; Hatchet-giant
DAAF: DE 3F ; Scorpion
DAB1: DE 9D ; Shield-knight
DAB3: DE 07 ; Wraith
DAB5: DD A3 ; Galdrog
DAB7: DF 65 ; Demon
DAB9: DF 10 ; Wizard

MonsterData:
;     To-kill  See  MShield  Damage  PShield  ?task-speed?
DABB: 00 20    00   FF       80      FF       17 0B ; Spider
DAC3: 00 38    00   FF       50      80       0F 07 ; Snake
DACB: 00 C8    00   FF       34      C0       1D 17 ; Giant
DAD3: 01 30    00   FF       60      A7       1F 1F ; Blob
DADB: 01 F8    00   80       60      3C       0D 07 ; Knight
DAE3: 02 C0    00   80       80      30       11 0D ; Hatchet-giant
DAEB: 01 90    FF   80       FF      80       05 04 ; Scorpion
DAF3: 03 20    00   40       FF      08       0D 07 ; Shield-knight
DAFB: 03 20    C0   10       C0      08       03 03 ; Wraith
DB03: 03 E8    FF   05       FF      03       04 03 ; Galdrog
DB0B: 03 E8    FF   06       FF      00       0D 07 ; Demon
DB13: 1F 40    FF   06       FF      00       0D 07 ; Wizard
```

Text Characters

Characters are 5x7 printed on 8x8 boundaries. The "extra" rows/columns allow for spacing between the characters. All 7 rows of a character are 5-bit-packed into 5 bytes and unpacked every single time needed.

```
TextCharacters:
DB1B: 30 00 00 00 00 ; SPACE 00110 > 00000 00000 00000 00000 00000 00000 00000  .....  ..X..  XXXX.
DB20: 31 15 18 FE 31 ; A     00110 > 00100 01010 10001 10001 11111 10001 10001  .....  .X.X.  X...X
DB25: 37 A3 1F 46 3E ; B     00110 > 11110 10001 10001 11110 10001 10001 11110  .....  X...X  X...X
DB2A: 33 A3 08 42 2E ; C     00110 > 01110 10001 10000 10000 10000 10001 01110  .....  X...X  XXXX.
DB2F: 37 A3 18 C6 3E ; D     00110 > 11110 10001 10001 10001 10001 10001 11110  .....  XXXXX  X...X
```

```
DB34: 37 E1 0F 42 1F ; E     00110 > 11111 10000 10000 11110 10000 10000 11111  .....  X...X  X...X
DB39: 37 E1 0F 42 10 ; F     00110 > 11111 10000 10000 11110 10000 10000 10000  .....  X...X  XXXX.
DB3E: 33 E3 08 4E 2F ; G     00110 > 01111 10001 10000 10000 10011 10001 01111
DB43: 34 63 1F C6 31 ; H     00110 > 10001 10001 10001 11111 10001 10001 10001  X...X  .XXX.  ....X
DB48: 33 88 42 10 8E ; I     00110 > 01110 00100 00100 00100 00100 00100 01110  X...X  ..X..  ....X
DB4D: 30 42 10 86 2E ; J     00110 > 00001 00001 00001 00001 00001 10001 01110  X...X  ..X..  ....X
DB52: 34 65 4C 52 51 ; K     00110 > 10001 10010 10100 11000 10100 10010 10001  XXXXX  ..X..  ....X
DB57: 34 21 08 42 1F ; L     00110 > 10000 10000 10000 10000 10000 10000 11111  X...X  ..X..  ....X
DB5C: 34 77 5A D6 31 ; M     00110 > 10001 11011 10101 10101 10101 10001 10001  X...X  ..X..  X...X
DB61: 34 63 9A CE 31 ; N     00110 > 10001 10001 11001 10101 10011 10001 10001  X...X  .XXX.  .XXX.
DB66: 33 A3 18 C6 2E ; O     00110 > 01110 10001 10001 10001 10001 10001 01110
DB6B: 37 A3 1F 42 10 ; P     00110 > 11110 10001 10001 11110 10000 10000 10000  XXXX.  .XXX.  XXXX.
DB70: 33 A3 18 D6 4D ; Q     00110 > 01110 10001 10001 10001 10101 10010 01101  X...X  X...X  X...X
DB75: 37 A3 1F 52 51 ; R     00110 > 11110 10001 10001 11110 10100 10010 10001  X...X  X...X  X...X
DB7A: 33 A3 07 06 2E ; S     00110 > 01110 10001 10000 01110 00001 10001 01110  XXXX.  X...X  XXXX.
DB7F: 37 EA 42 10 84 ; T     00110 > 11111 10101 00100 00100 00100 00100 00100  X....  X.X.X  X.X..
DB84: 34 63 18 C6 2E ; U     00110 > 10001 10001 10001 10001 10001 10001 01110  X....  X..X.  X..X.
DB89: 34 63 15 28 84 ; V     00110 > 10001 10001 10001 01010 01010 00100 00100  X....  .XX.X  X...X
DB8E: 34 63 1A D7 71 ; W     00110 > 10001 10001 10001 10101 10101 11011 10001
DB93: 34 62 A2 2A 31 ; X     00110 > 10001 10001 01010 00100 01010 10001 10001  X...X  X...X  XXXXX
DB98: 34 62 A2 10 84 ; Y     00110 > 10001 10001 01010 00100 00100 00100 00100  X...X  X...X  ....X
DB9D: 37 C2 22 22 1F ; Z     00110 > 11111 00001 00010 00100 01000 10000 11111  .X.X.  .X.X.  ...X.
DBA2: 31 08 42 10 04 ; !     00110 > 00100 00100 00100 00100 00100 00000 00100  ..X..  ..X..  ..X..
DBA7: 30 00 00 00 1F ; _     00110 > 00000 00000 00000 00000 00000 00000 11111  .X.X.  ..X..  .X...
DBAC: 33 A2 13 10 04 ; ?     00110 > 01110 10001 00001 00110 00100 00000 00100  X...X  ..X..  X....
DBB1: 30 00 00 00 04 ; .     00110 > 00000 00000 00000 00000 00000 00000 00100  X...X  ..X..  XXXXX
```

Heart Pictures

```
DBB6: 00 00 01 01 00 00 00
; ........
; ........
; .......X
; .......X
; ........
; ........
; ........

DBBD: 00 A0 F0 F0 E0 40 00
```

```
; ........
; X.X.....
; XXXX....
; XXXX....
; XXX.....
; .X......
; ........

DBC4: 00 01 03 03 01 00 00
; ........
; .......X
; ......XX
; ......XX
; .......X
; ........
; ........

DBCB: 00 B0 F8 F8 F0 E0 40
; ........
; X.XX....
; XXXXX...
; XXXXX...
; XXXX....
; XXX.....
; .X......

DBD2: 00 80   00 01
DBD6: 00 50   00 04
DBDA: 00 50   00 05

WallPictures:
DBDE: 03    ; Left
DBDF: DC 4F ; Left wall open
DBE1: DC 6B ; Left wall with physical door
DBE3: DC 9B ; Left magic door
DBE5: DC 33 ; Left wall solid
DBE7: 00    ; Front
DBE8: DC 6A ; Front wall open (draw nothing)
DBEA: DC 8B ; Front wall with physical door
DBEC: DC A9 ; Front wall magic door
DBEE: DC 45 ; Front wall (lines at top and bottom)
```

```
DBF0: 01    ; Right
DBF1: DC 5D ; Right wall open
DBF3: DC 7B ; Right wall with physical door
DBF5: DC A2 ; Right magic door
DBF7: DC 3C ; Right wall solid
DBF9: FF    ; End
```

Object Pictures

```
ShieldPic:
; Shield
DBFA: 86 AC         ; Move to absolute (172,134)
DBFC: 80 C0         ; Line to absolute (192,128)
DBFE: 7A BA         ; Line to absolute (186,122)
DC00: 80 A8         ; Line to absolute (168,128)
DC02: FC            ; Draw short lines
DC03: 3E            ;     Short line to relative (-4,6)
DC04: 04            ;     Short line to relative (8,0)
DC05: 00            ;     End of short lines
DC06: FE            ; End of image


; Torch
DC07: 76 3C         ; Move to absolute (60,118)
DC09: FC            ; Draw short lines
DC0A: F7            ;     Short line to relative (14,-2)
DC0B: FF            ;     Short line to relative (-2,-2)
DC0C: 2A            ;     Short line to relative (-12,4)
DC0D: 00            ;     End of short lines
DC0E: FE            ; End of image


; Sword
DC0F: 72 50         ; Move to absolute (80,114)
DC11: 7C 64         ; Line to absolute (100,124)
DC13: FF            ; Start new line
DC14: 76 52         ; Move to absolute (82,118)
DC16: 72 56         ; Line to absolute (86,114)
DC18: FE            ; End of image
```

```
; Flask
DC19: 6E A2         ; Move to absolute (162,110)
DC1B: FC            ; Draw short lines
DC1C: 51            ;     Short line to relative (2,10)
DC1D: 0E            ;     Short line to relative (-4,0)
DC1E: B1            ;     Short line to relative (2,-10)
DC1F: 00            ;     End of short lines
DC20: FE            ; End of image


; Ring
DC21: 7A 3C         ; Move to absolute (60,122)
DC23: FC            ; Draw short lines
DC24: 11            ;     Short line to relative (2,2)
DC25: 1F            ;     Short line to relative (-2,2)
DC26: FF            ;     Short line to relative (-2,-2)
DC27: F1            ;     Short line to relative (2,-2)
DC28: 00            ;     End of short lines
DC29: FE            ; End of image


; Scroll
DC2A: 76 C2         ; Move to absolute (194,118)
DC2C: FC            ; Draw short lines
DC2D: 1F            ;     Short line to relative (-2,2)
DC2E: 34            ;     Short line to relative (8,6)
DC2F: F1            ;     Short line to relative (2,-2)
DC30: DC            ;     Short line to relative (-8,-6)
DC31: 00            ;     End of short lines
DC32: FE            ; End of image
```

Walls and Doors

```
; Left wall
DC33: 10 1B         ; Move to absolute (27,16)
DC35: 26 40         ; Line to absolute (64,38)
```

```
DC37: 72 40         ; Line to absolute (64,114)
DC39: 88 1B         ; Line to absolute (27,136)
DC3B: FE            ; End of image

; Right wall
DC3C: 10 E5         ; Move to absolute (229,16)
DC3E: 26 C0         ; Line to absolute (192,38)
DC40: 72 C0         ; Line to absolute (192,114)
DC42: 88 E5         ; Line to absolute (229,136)
DC44: FE            ; End of image

; Front wall (line at top and bottom)
DC45: 26 40         ; Move to absolute (64,38)
DC47: 26 C0         ; Line to absolute (192,38)
DC49: FF            ; Start new line
DC4A: 72 40         ; Move to absolute (64,114)
DC4C: 72 C0         ; Line to absolute (192,114)
DC4E: FE            ; End of image

; Left wall open
DC4F: 26 1D         ; Move to absolute (29,38)
DC51: 26 40         ; Line to absolute (64,38)
DC53: 72 40         ; Line to absolute (64,114)
DC55: 72 1B         ; Line to absolute (27,114)
DC57: FF            ; Start new line
DC58: 10 1B         ; Move to absolute (27,16)
DC5A: 26 40         ; Line to absolute (64,38)
DC5C: FE            ; End of image

; Right wall open
DC5D: 26 E5         ; Move to absolute (229,38)
DC5F: 26 C0         ; Line to absolute (192,38)
DC61: 72 C0         ; Line to absolute (192,114)
DC63: 72 E5         ; Line to absolute (229,114)
DC65: FF            ; Start new line
DC66: 10 E5         ; Move to absolute (229,16)
DC68: 26 C0         ; Line to absolute (192,38)
DC6A: FE            ; End of image

; Left wall physical door
DC6B: 80 28         ; Move to absolute (40,128)
```

```
DC6D: 41 28         ; Line to absolute (40,65)
DC6F: 44 38         ; Line to absolute (56,68)
DC71: 77 38         ; Line to absolute (56,119)
DC73: FF            ; Start new line
DC74: 5C 30         ; Move to absolute (48,92)
DC76: 5D 34         ; Line to absolute (52,93)
DC78: FD DC 33      ; Jump to DC33

; Right wall with physical door
DC7B: 80 D8         ; Line to absolute (216,128)
DC7D: 41 D8         ; Line to absolute (216,65)
DC7F: 44 C8         ; Line to absolute (200,68)
DC81: 77 C8         ; Line to absolute (200,119)
DC83: FF            ; Start new line
DC84: 5C D0         ; Move to absolute (208,92)
DC86: 5D CC         ; Line to absolute (204,93)
DC88: FD DC 3C      ; Jump to DC3C

;Front wall with physical door
DC8B: 72 6C         ; Line to absolute (108,114)
DC8D: 43 6C         ; Line to absolute (108,67)
DC8F: 43 94         ; Line to absolute (148,67)
DC91: 72 94         ; Line to absolute (148,114)
DC93: FF            ; Start new line
DC94: 5E 7E         ; Move to absolute (126,94)
DC96: 5E 82         ; Line to absolute (130,94)
DC98: FD DC 45      ; Jump to DC45

; Left magic door
DC9B: 80 28         ; Line to absolute (40,128)
DC9D: 42 32         ; Line to absolute (50,66)
DC9F: 75 3A         ; Line to absolute (58,117)
DCA1: FE            ; End of image

; Right magic door
DCA2: 80 D8         ; Move to absolute (216,128)
DCA4: 42 CE         ; Line to absolute (206,66)
DCA6: 75 C6         ; Line to absolute (198,117)
DCA8: FE            ; End of image

; Front magic door
```

```
DCA9: 71 6C         ; Move to absolute (108,113)
DCAB: 43 80         ; Line to absolute (128,67)
DCAD: 72 94         ; Line to absolute (148,114)
DCAF: FE            ; End of image

; Creature on left
DCB0: 64 1C         ; Move to absolute (28,100)
DCB2: FC            ; Draw short lines
DCB3: 44            ;     Short line to relative (8,8)
DCB4: 2E            ;     Short line to relative (-4,4)
DCB5: 42            ;     Short line to relative (4,8)
DCB6: 4C            ;     Short line to relative (-8,8)
DCB7: 00            ;     End of short lines
DCB8: FE            ; End of image

; Creature on right
DCB9: 64 E4         ; Move to absolute (228,100)
DCBB: FC            ; Draw short lines
DCBC: 4C            ;     Short line to relative (-8,8)
DCBD: 22            ;     Short line to relative (4,4)
DCBE: 4E            ;     Short line to relative (-4,8)
DCBF: 44            ;     Short line to relative (8,8)
DCC0: 00            ;     End of short lines
DCC1: FE            ; End of image

HoleList:
; Table for drawing holes/ladders
DCC2: DD 0E  ; Hole in ceiling
DCC4: DC CA  ; Ladder through ceiling
DCC6: DD 2A  ; Hole in floor
DCC8: DC D0  ; Ladder through floor
```

Holes and Ladders

```
; Ladder through ceiling
DCCA: FB DC D6 ; Ladder subroutine
DCCD: FD DD 0E ; Hole in ceiling

; Ladder through floor
```

```
DCD0: FB DC D6 ; Ladder subroutine
DCD3: FD DD 2A ; Hole in floor

; Ladder subroutine
DCD6: 18 74         ; Move to absolute (116,24)
DCD8: 80 74         ; Line to absolute (116,128)
DCDA: FF            ; Start new line
DCDB: 18 8C         ; Move to absolute (140,24)
DCDD: 80 8C         ; Line to absolute (140,128)
DCDF: FF            ; Start new line
DCE0: 1C 74         ; Move to absolute (116,28)
DCE2: 1C 8C         ; Line to absolute (140,28)
DCE4: FF            ; Start new line
DCE5: 28 74         ; Move to absolute (116,40)
DCE7: 28 8C         ; Line to absolute (140,40)
DCE9: FF            ; Start new line
DCEA: 34 74         ; Move to absolute (116,52)
DCEC: 34 8C         ; Line to absolute (140,52)
DCEE: FF            ; Start new line
DCEF: 40 74         ; Move to absolute (116,64)
DCF1: 40 8C         ; Line to absolute (140,64)
DCF3: FF            ; Start new line
DCF4: 4C 74         ; Move to absolute (116,76)
DCF6: 4C 8C         ; Line to absolute (140,76)
DCF8: FF            ; Start new line
DCF9: 58 74         ; Move to absolute (116,88)
DCFB: 58 8C         ; Line to absolute (140,88)
DCFD: FF            ; Start new line
DCFE: 64 74         ; Move to absolute (116,100)
DD00: 64 8C         ; Line to absolute (140,100)
DD02: FF            ; Start new line
DD03: 70 74         ; Move to absolute (116,112)
DD05: 70 8C         ; Line to absolute (140,112)
DD07: FF            ; Start new line
DD08: 7B 74         ; Move to absolute (116,123)
DD0A: 7B 8C         ; Line to absolute (140,123)
DD0C: FF            ; Start new line
DD0D: FA            ; Return

; Hole in ceiling
DD0E: 22 64         ; Move to absolute (100,34)
```

```
DD10: 18 5C         ; Line to absolute (92,24)
DD12: 18 A4         ; Line to absolute (164,24)
DD14: 22 9C         ; Line to absolute (156,34)
DD16: 22 64         ; Line to absolute (100,34)
DD18: 18 64         ; Line to absolute (100,24)
DD1A: FF            ; Start new line
DD1B: 22 9C         ; Move to absolute (156,34)
DD1D: 18 9C         ; Line to absolute (156,24)
DD1F: FF            ; Start new line
DD20: 1C 2F         ; Move to absolute (47,28)
DD22: 1C 60         ; Line to absolute (96,28)
DD24: FF            ; Start new line
DD25: 1C A1         ; Move to absolute (161,28)
DD27: 1C D2         ; Line to absolute (210,28)
DD29: FE            ; End of image

; Hole in floor
DD2A: 76 64         ; Move to absolute (100,118)
DD2C: 80 5C         ; Line to absolute (92,128)
DD2E: 80 A4         ; Line to absolute (164,128)
DD30: 76 9C         ; Line to absolute (156,118)
DD32: 76 64         ; Line to absolute (100,118)
DD34: 80 64         ; Line to absolute (100,128)
DD36: FF            ; Start new line
DD37: 76 9C         ; Move to absolute (156,118)
DD39: 80 9C         ; Line to absolute (156,128)
DD3B: FF            ; Start new line
DD3C: 1C 2F         ; Move to absolute (47,28)
DD3E: 1C D2         ; Line to absolute (210,28)
DD40: FE            ; End of image
```

Club Giant Picture

```
ClubGiantPic:
DD41: 68 62         ; Move to absolute (98,104)
DD43: FC            ; Draw short lines
DD44: D7            ;     Short line to relative (14,-6)
DD45: D4            ;     Short line to relative (8,-6)
DD46: 14            ;     Short line to relative (8,2)
```

```
DD47: 12           ;     Short line to relative (4,2)
DD48: 30           ;     Short line to relative (0,6)
DD49: 1D           ;     Short line to relative (-6,2)
DD4A: 0D           ;     Short line to relative (-6,0)
DD4B: FD           ;     Short line to relative (-6,-2)
DD4C: 29           ;     Short line to relative (-14,4)
DD4D: 00           ;     End of short lines
DD4E: FD DD 62     ; Jump to DD62
;
```

Hatchet Giant Picture

```
HatchetGiantPic:
; Giant (with hatchet)
DD51: 68 62        ; Move to absolute (98,104)
DD53: 5E 7C        ; Line to absolute (124,94)
DD55: 60 7E        ; Line to absolute (126,96)
DD57: 6A 64        ; Line to absolute (100,106)
DD59: FF           ; Start new line
DD5A: 66 84        ; Move to absolute (132,102)
DD5C: 5C 72        ; Line to absolute (114,92)
DD5E: 66 76        ; Line to absolute (118,102)
DD60: 6E 72        ; Line to absolute (114,110)
;
; Common Giant
DD62: 66 84        ; Line to absolute (132,102)
DD64: FC           ; Draw short lines
DD65: 02           ;     Short line to relative (4,0)
DD66: 56           ;     Short line to relative (12,10)
DD67: 56           ;     Short line to relative (12,10)
DD68: 17           ;     Short line to relative (14,2)
DD69: EE           ;     Short line to relative (-4,-4)
DD6A: 02           ;     Short line to relative (4,0)
DD6B: EA           ;     Short line to relative (-12,-4)
DD6C: BB           ;     Short line to relative (-10,-10)
DD6D: BB           ;     Short line to relative (-10,-10)
DD6E: EA           ;     Short line to relative (-12,-4)
DD6F: EA           ;     Short line to relative (-12,-4)
DD70: 00           ;     End of short lines
```

```
DD71: 4E 5C         ; Move to absolute (92,78)
DD73: FC            ; Draw short lines
DD74: C2            ;     Short line to relative (4,-8)
DD75: 51            ;     Short line to relative (2,10)
DD76: 3E            ;     Short line to relative (-4,6)
DD77: CF            ;     Short line to relative (-2,-8)
DD78: FC            ;     Short line to relative (-8,-2)
DD79: 42            ;     Short line to relative (4,8)
DD7A: 13            ;     Short line to relative (6,2)
DD7B: 00            ;     End of short lines
DD7C: 6A 5A         ; Move to absolute (90,106)
DD7E: FC            ; Draw short lines
DD7F: 1E            ;     Short line to relative (-4,2)
DD80: 11            ;     Short line to relative (2,2)
DD81: F3            ;     Short line to relative (6,-2)
DD82: 62            ;     Short line to relative (4,12)
DD83: 39            ;     Short line to relative (-14,6)
DD84: E2            ;     Short line to relative (4,-4)
DD85: 0C            ;     Short line to relative (-8,0)
DD86: E4            ;     Short line to relative (8,-4)
DD87: 8A            ;     Short line to relative (-12,-16)
DD88: E2            ;     Short line to relative (4,-4)
DD89: 00            ;     End of short lines
DD8A: 56 54         ; Move to absolute (84,86)
DD8C: FC            ; Draw short lines
DD8D: 54            ;     Short line to relative (8,10)
DD8E: 65            ;     Short line to relative (10,12)
DD8F: 2E            ;     Short line to relative (-4,4)
DD90: CA            ;     Short line to relative (-12,-8)
DD91: BA            ;     Short line to relative (-12,-10)
DD92: A1            ;     Short line to relative (2,-12)
DD93: D4            ;     Short line to relative (8,-6)
DD94: EE            ;     Short line to relative (-4,-4)
DD95: 12            ;     Short line to relative (4,2)
DD96: D2            ;     Short line to relative (4,-6)
DD97: 13            ;     Short line to relative (6,2)
DD98: E1            ;     Short line to relative (2,-4)
DD99: 20            ;     Short line to relative (0,4)
DD9A: F6            ;     Short line to relative (12,-2)
DD9B: 24            ;     Short line to relative (8,4)
DD9C: 72            ;     Short line to relative (4,14)
```

```
DD9D: 58           ;     Short line to relative (-16,10)
DD9E: EE           ;     Short line to relative (-4,-4)
DD9F: C5           ;     Short line to relative (10,-8)
DDA0: BE           ;     Short line to relative (-4,-10)
DDA1: 00           ;     End of short lines
DDA2: FE           ; End of image
```

Galdrog Picture

```
GaldrogPic:
DDA3: 50 7C        ; Move to absolute (124,80)
DDA5: 5E 72        ; Line to absolute (114,94)
DDA7: 6E 78        ; Line to absolute (120,110)
DDA9: 84 70        ; Line to absolute (112,132)
DDAB: 68 4E        ; Line to absolute (78,104)
DDAD: 84 30        ; Line to absolute (48,132)
DDAF: 44 48        ; Line to absolute (72,68)
DDB1: 54 20        ; Line to absolute (32,84)
DDB3: 16 58        ; Line to absolute (88,22)
DDB5: 34 72        ; Line to absolute (114,52)
DDB7: 5C 80        ; Line to absolute (128,92)
DDB9: 34 8E        ; Line to absolute (142,52)
DDBB: 16 A8        ; Line to absolute (168,22)
DDBD: 58 E0        ; Line to absolute (224,88)
DDBF: 44 B8        ; Line to absolute (184,68)
DDC1: 84 D0        ; Line to absolute (208,132)
DDC3: 70 B2        ; Line to absolute (178,112)
DDC5: 84 90        ; Line to absolute (144,132)
DDC7: 6E 88        ; Line to absolute (136,110)
DDC9: 5E 8E        ; Line to absolute (142,94)
DDCB: 50 84        ; Line to absolute (132,80)
DDCD: FF           ; Start new line
DDCE: 84 70        ; Move to absolute (112,132)
DDD0: FC           ; Draw short lines
DDD1: C5           ;     Short line to relative (10,-8)
DDD2: 92           ;     Short line to relative (4,-14)
DDD3: BE           ;     Short line to relative (-4,-10)
DDD4: C3           ;     Short line to relative (6,-8)
DDD5: 43           ;     Short line to relative (6,8)
```

```
DDD6: 5E           ;     Short line to relative (-4,10)
DDD7: 72           ;     Short line to relative (4,14)
DDD8: 45           ;     Short line to relative (10,8)
DDD9: 00           ;     End of short lines
DDDA: 52 7A        ; Move to absolute (122,82)
DDDC: FC           ; Draw short lines
DDDD: 78           ;     Short line to relative (-16,14)
DDDE: E9           ;     Short line to relative (-14,-4)
DDDF: 8D           ;     Short line to relative (-6,-16)
DDE0: EC           ;     Short line to relative (-8,-4)
DDE1: 33           ;     Short line to relative (6,6)
DDE2: 0C           ;     Short line to relative (-8,0)
DDE3: 24           ;     Short line to relative (8,4)
DDE4: 72           ;     Short line to relative (4,14)
DDE5: 47           ;     Short line to relative (14,8)
DDE6: E7           ;     Short line to relative (14,-4)
DDE7: 00           ;     End of short lines
DDE8: 16 A8        ; Move to absolute (168,22)
DDEA: FC           ; Draw short lines
DDEB: 2D           ;     Short line to relative (-6,4)
DDEC: C2           ;     Short line to relative (4,-8)
DDED: 3D           ;     Short line to relative (-6,6)
DDEE: 30           ;     Short line to relative (0,6)
DDEF: 4B           ;     Short line to relative (-10,8)
DDF0: 4B           ;     Short line to relative (-10,8)
DDF1: ED           ;     Short line to relative (-6,-4)
DDF2: B2           ;     Short line to relative (4,-10)
DDF3: 9D           ;     Short line to relative (-6,-14)
DDF4: 71           ;     Short line to relative (2,14)
DDF5: 3D           ;     Short line to relative (-6,6)
DDF6: DD           ;     Short line to relative (-6,-6)
DDF7: 91           ;     Short line to relative (2,-14)
DDF8: 7D           ;     Short line to relative (-6,14)
DDF9: 52           ;     Short line to relative (4,10)
DDFA: 63           ;     Short line to relative (6,12)
DDFB: A3           ;     Short line to relative (6,-12)
DDFC: 2D           ;     Short line to relative (-6,4)
DDFD: ED           ;     Short line to relative (-6,-4)
DDFE: 2D           ;     Short line to relative (-6,4)
DDFF: CB           ;     Short line to relative (-10,-8)
DE00: CB           ;     Short line to relative (-10,-8)
```

```
DE01: D0            ;     Short line to relative (0,-6)
DE02: DD            ;     Short line to relative (-6,-6)
DE03: 42            ;     Short line to relative (4,8)
DE04: ED            ;     Short line to relative (-6,-4)
DE05: 00            ;     End of short lines
DE06: FE            ; End of image
```

Wraith Picture

```
WraithPic:
DE07: 3E 44         ; Move to absolute (68,62)
DE09: 44 58         ; Line to absolute (88,68)
DE0B: 38 64         ; Line to absolute (100,56)
DE0D: FF            ; Start new line
DE0E: 4A 5A         ; Move to absolute (90,74)
DE10: 46 4A         ; Line to absolute (74,70)
DE12: FC            ; Draw short lines
DE13: 33            ;     Short line to relative (6,6)
DE14: F5            ;     Short line to relative (10,-2)
DE15: F5            ;     Short line to relative (10,-2)
DE16: C1            ;     Short line to relative (2,-8)
DE17: 5A            ;     Short line to relative (-12,10)
DE18: 62            ;     Short line to relative (4,12)
DE19: 0E            ;     Short line to relative (-4,0)
DE1A: 00            ;     End of short lines
DE1B: 64 50         ; Move to absolute (80,100)
DE1D: FC            ; Draw short lines
DE1E: B3            ;     Short line to relative (6,-10)
DE1F: 17            ;     Short line to relative (14,2)
DE20: 34            ;     Short line to relative (8,6)
DE21: EB            ;     Short line to relative (-10,-4)
DE22: 0A            ;     Short line to relative (-12,0)
DE23: 3D            ;     Short line to relative (-6,6)
DE24: 00            ;     End of short lines
DE25: FE            ; End of image
```

Spider Picture

```
SpiderPic:
DE26: 7C A0         ; Move to absolute (160,124)
DE28: FC            ; Draw short lines
DE29: C2            ;     Short line to relative (4,-8)
DE2A: 22            ;     Short line to relative (4,4)
DE2B: E4            ;     Short line to relative (8,-4)
DE2C: 24            ;     Short line to relative (8,4)
DE2D: 2C            ;     Short line to relative (-8,4)
DE2E: EC            ;     Short line to relative (-8,-4)
DE2F: 04            ;     Short line to relative (8,0)
DE30: 04            ;     Short line to relative (8,0)
DE31: E2            ;     Short line to relative (4,-4)
DE32: 42            ;     Short line to relative (4,8)
DE33: 00            ;     End of short lines
DE34: 7C A8         ; Move to absolute (168,124)
DE36: FC            ; Draw short lines
DE37: C1            ;     Short line to relative (2,-8)
DE38: 21            ;     Short line to relative (2,4)
DE39: 12            ;     Short line to relative (4,2)
DE3A: F2            ;     Short line to relative (4,-2)
DE3B: E1            ;     Short line to relative (2,-4)
DE3C: 41            ;     Short line to relative (2,8)
DE3D: 00            ;     End of short lines
DE3E: FE            ; End of image
```

Scorpion Picture

```
ScorpionPic:
DE3F: 70 4A         ; Move to absolute (74,112)
DE41: FC            ; Draw short lines
DE42: E0            ;     Short line to relative (0,-4)
DE43: EE            ;     Short line to relative (-4,-4)
DE44: 2C            ;     Short line to relative (-8,4)
DE45: 42            ;     Short line to relative (4,8)
DE46: 14            ;     Short line to relative (8,2)
DE47: 14            ;     Short line to relative (8,2)
DE48: 20            ;     Short line to relative (0,4)
DE49: 0C            ;     Short line to relative (-8,0)
DE4A: CC            ;     Short line to relative (-8,-8)
```

```
DE4B: 22            ;     Short line to relative (4,4)
DE4C: 0C            ;     Short line to relative (-8,0)
DE4D: 22            ;     Short line to relative (4,4)
DE4E: 00            ;     End of short lines
DE4F: 7C 5A         ; Move to absolute (90,124)
DE51: FC            ; Draw short lines
DE52: E0            ;     Short line to relative (0,-4)
DE53: 0C            ;     Short line to relative (-8,0)
DE54: 2C            ;     Short line to relative (-8,4)
DE55: 20            ;     Short line to relative (0,4)
DE56: 04            ;     Short line to relative (8,0)
DE57: 00            ;     End of short lines
DE58: FE            ; End of image
```

Blob Picture

```
BlobPic:
;
; Body outline
DE59: 52 82         ; Move to absolute (130,82)
DE5B: FC            ; Draw short lines
DE5C: 28            ;     Short line to relative (-16,4)
DE5D: 7D            ;     Short line to relative (-6,14)
DE5E: 5F            ;     Short line to relative (-2,10)
DE5F: 50            ;     Short line to relative (0,10)
DE60: 5B            ;     Short line to relative (-10,10)
DE61: F5            ;     Short line to relative (10,-2)
DE62: 2F            ;     Short line to relative (-2,4)
DE63: D5            ;     Short line to relative (10,-6)
DE64: 17            ;     Short line to relative (14,2)
DE65: 17            ;     Short line to relative (14,2)
DE66: F3            ;     Short line to relative (6,-2)
DE67: 22            ;     Short line to relative (4,4)
DE68: E1            ;     Short line to relative (2,-4)
DE69: 14            ;     Short line to relative (8,2)
DE6A: DD            ;     Short line to relative (-6,-6)
DE6B: 8F            ;     Short line to relative (-2,-16)
DE6C: 8D            ;     Short line to relative (-6,-16)
DE6D: DB            ;     Short line to relative (-10,-6)
```

```
DE6E: EC            ;     Short line to relative (-8,-4)
DE6F: 00            ;     End of short lines
;
; Eyes
DE70: 56 82         ; Move to absolute (130,86)
DE72: FC            ; Draw short lines
DE73: 33            ;     Short line to relative (6,6)
DE74: 31            ;     Short line to relative (2,6)
DE75: 1B            ;     Short line to relative (-10,2)
DE76: 91            ;     Short line to relative (2,-14)
DE77: 3B            ;     Short line to relative (-10,6)
DE78: 5F            ;     Short line to relative (-2,10)
DE79: F5            ;     Short line to relative (10,-2)
DE7A: 00            ;     End of short lines
;
; Mouth
DE7B: 6C 74         ; Move to absolute (116,108)
DE7D: 72 76         ; Line to (118,114)
DE7F: 78 90         ; Line to (144,120)
DE81: FE            ; End of image
```

Knight Picture

```
KnightPic:
DE82: 22 7C         ; Move to absolute (124,34)
DE84: FC            ; Draw short lines
DE85: 04            ;     Short line to relative (8,0)
DE86: 1F            ;     Short line to relative (-2,2)
DE87: 0E            ;     Short line to relative (-4,0)
DE88: FF            ;     Short line to relative (-2,-2)
DE89: 00            ;     End of short lines
DE8A: 50 8E         ; Move to absolute (142,80)
DE8C: 40 88         ; Line to absolute (136,64)
DE8E: 2E 92         ; Line to absolute (146,46)
DE90: 40 9C         ; Line to absolute (156,64)
DE92: 52 8C         ; Line to absolute (140,82)
DE94: 4C 88         ; Line to absolute (136,76)
DE96: 40 92         ; Line to absolute (146,64)
DE98: 3A 8C         ; Line to absolute (140,58)
```

```
DE9A: FD DE B3      ; Jump to DEB3
;
```

Shield Knight Picture

```
ShieldKnightPic:
DE9D: 1E 7E         ; Line to absolute (126,30)
DE9F: FC            ; Draw short lines
DEA0: 50            ;     Short line to relative (0,10)
DEA1: 0F            ;     Short line to relative (-2,0)
DEA2: E0            ;     Short line to relative (0,-4)
DEA3: 00            ;     End of short lines
DEA4: 2C 96         ; Move to absolute (150,44)
DEA6: 34 A6         ; Line to absolute (166,52)
DEA8: 4C A4         ; Line to absolute (164,76)
DEAA: 5C 96         ; Line to absolute (150,92)
DEAC: 4C 88         ; Line to absolute (136,76)
DEAE: 34 86         ; Line to absolute (134,52)
DEB0: 2C 96         ; Line to absolute (150,44)
DEB2: FF            ; Start new line
;
; Common knight
DEB3: 50 8C         ; Move to absolute (140,80)
DEB5: 80 98         ; Line to absolute (152,128)
DEB7: 84 A0         ; Line to absolute (160,132)
DEB9: 84 90         ; Line to absolute (144,132)
DEBB: 7E 90         ; Line to absolute (144,126)
DEBD: 54 82         ; Line to absolute (130,84)
DEBF: FF            ; Start new line
DEC0: 54 7E         ; Move to absolute (126,84)
DEC2: 7E 6E         ; Line to absolute (110,126)
DEC4: 84 6E         ; Line to absolute (110,132)
DEC6: 84 5C         ; Line to absolute (92,132)
DEC8: 80 66         ; Line to absolute (102,128)
DECA: 50 74         ; Line to absolute (116,80)
DECC: FF            ; Start new line
DECD: 50 8C         ; Move to absolute (140,80)
DECF: FC            ; Draw short lines
DED0: 3A            ;     Short line to relative (-12,6)
```

```
DED1: D9           ;     Short line to relative (-14,-6)
DED2: 83           ;     Short line to relative (6,-16)
DED3: DE           ;     Short line to relative (-4,-6)
DED4: AD           ;     Short line to relative (-6,-12)
DED5: E6           ;     Short line to relative (12,-4)
DED6: A1           ;     Short line to relative (2,-12)
DED7: E2           ;     Short line to relative (4,-4)
DED8: 22           ;     Short line to relative (4,4)
DED9: 61           ;     Short line to relative (2,12)
DEDA: 26           ;     Short line to relative (12,4)
DEDB: EA           ;     Short line to relative (-12,-4)
DEDC: 20           ;     Short line to relative (0,4)
DEDD: 3D           ;     Short line to relative (-6,6)
DEDE: DD           ;     Short line to relative (-6,-6)
DEDF: E0           ;     Short line to relative (0,-4)
DEE0: 00           ;     End of short lines
DEE1: 34 80        ; Move to absolute (128,52)
DEE3: 14 80        ; Line to absolute (128,20)
DEE5: FC           ; Draw short lines
DEE6: 0E           ;     Short line to relative (-4,0)
DEE7: 21           ;     Short line to relative (2,4)
DEE8: 02           ;     Short line to relative (4,0)
DEE9: E1           ;     Short line to relative (2,-4)
DEEA: 0E           ;     Short line to relative (-4,0)
DEEB: 00           ;     End of short lines
DEEC: 4A 66        ; Move to absolute (102,74)
DEEE: FC           ; Draw short lines
DEEF: E0           ;     Short line to relative (0,-4)
DEF0: 02           ;     Short line to relative (4,0)
DEF1: D0           ;     Short line to relative (0,-6)
DEF2: 08           ;     Short line to relative (-16,0)
DEF3: 30           ;     Short line to relative (0,6)
DEF4: 02           ;     Short line to relative (4,0)
DEF5: 20           ;     Short line to relative (0,4)
DEF6: 01           ;     Short line to relative (2,0)
DEF7: 30           ;     Short line to relative (0,6)
DEF8: 02           ;     Short line to relative (4,0)
DEF9: D0           ;     Short line to relative (0,-6)
DEFA: 01           ;     Short line to relative (2,0)
DEFB: 87           ;     Short line to relative (14,-16)
DEFC: 00           ;     End of short lines
```

```
DEFD: 2E 6E         ; Move to absolute (110,46)
DEFF: 40 66         ; Line to absolute (102,64)
DF01: 40 64         ; Line to absolute (100,64)
DF03: 1E 66         ; Line to absolute (102,30)
DF05: 14 62         ; Line to absolute (98,20)
DF07: 1E 5E         ; Line to absolute (94,30)
DF09: 40 60         ; Line to absolute (96,64)
DF0B: 40 62         ; Line to absolute (98,64)
DF0D: 14 62         ; Line to absolute (98,20)
DF0F: FE            ; End of image
```

Moon Wizard Picture

```
MoonWizardPic:
DF10: 2E 62         ; Move to absolute (98,46)
DF12: FC            ; Draw short lines
DF13: 21            ;     Short line to relative (2,4)
DF14: 2F            ;     Short line to relative (-2,4)
DF15: 2D            ;     Short line to relative (-6,4)
DF16: FD            ;     Short line to relative (-6,-2)
DF17: CE            ;     Short line to relative (-4,-8)
DF18: C2            ;     Short line to relative (4,-8)
DF19: F2            ;     Short line to relative (4,-2)
DF1A: 12            ;     Short line to relative (4,2)
DF1B: 0F            ;     Short line to relative (-2,0)
DF1C: 1E            ;     Short line to relative (-4,2)
DF1D: 3F            ;     Short line to relative (-2,6)
DF1E: 21            ;     Short line to relative (2,4)
DF1F: 12            ;     Short line to relative (4,2)
DF20: E3            ;     Short line to relative (6,-4)
DF21: E0            ;     Short line to relative (0,-4)
DF22: 00            ;     End of short lines
DF23: 68 9A         ; Move to absolute (154,104)
DF25: FC            ; Draw short lines
DF26: 21            ;     Short line to relative (2,4)
DF27: 2F            ;     Short line to relative (-2,4)
DF28: 2D            ;     Short line to relative (-6,4)
DF29: FD            ;     Short line to relative (-6,-2)
DF2A: CE            ;     Short line to relative (-4,-8)
```

```
DF2B: C2           ;     Short line to relative (4,-8)
DF2C: F2           ;     Short line to relative (4,-2)
DF2D: 12           ;     Short line to relative (4,2)
DF2E: 0F           ;     Short line to relative (-2,0)
DF2F: 1E           ;     Short line to relative (-4,2)
DF30: 3F           ;     Short line to relative (-2,6)
DF31: 22           ;     Short line to relative (4,4)
DF32: 12           ;     Short line to relative (4,2)
DF33: E2           ;     Short line to relative (4,-4)
DF34: E0           ;     Short line to relative (0,-4)
DF35: 00           ;     End of short lines
DF36: FD DF 65     ; Jump to DF65
```

Star Wizard Picture

```
StarWizardPic:
DF39: 28 56        ; Move to absolute (86,40)
DF3B: 40 5C        ; Line to absolute (92,64)
DF3D: 2A 64        ; Line to absolute (100,42)
DF3F: 36 52        ; Line to absolute (82,54)
DF41: 38 68        ; Line to absolute (104,56)
DF43: 28 56        ; Line to absolute (86,40)
DF45: FF           ; Start new line
DF46: 42 8C        ; Move to absolute (140,66)
DF48: FC           ; Draw short lines
DF49: 70           ;     Short line to relative (0,14)
DF4A: AD           ;     Short line to relative (-6,-12)
DF4B: 35           ;     Short line to relative (10,6)
DF4C: 1B           ;     Short line to relative (-10,2)
DF4D: B3           ;     Short line to relative (6,-10)
DF4E: 00           ;     End of short lines
DF4F: 60 92        ; Move to absolute (146,96)
DF51: 78 94        ; Line to absolute (148,120)
DF53: 64 88        ; Line to absolute (136,100)
DF55: 6A 9A        ; Line to absolute (154,106)
DF57: 74 8A        ; Line to absolute (138,116)
DF59: 60 92        ; Line to absolute (146,96)
DF5B: FF           ; Start new line
DF5C: 50 74        ; Move to absolute (116,80)
```

```
DF5E: FC            ; Draw short lines
DF5F: 53            ;     Short line to relative (6,10)
DF60: EC            ;     Short line to relative (-8,-4)
DF61: E4            ;     Short line to relative (8,-4)
DF62: 4D            ;     Short line to relative (-6,8)
DF63: B0            ;     Short line to relative (0,-10)
DF64: 00            ;     End of short lines
```

Demon Picture

```
DemonPic:
DF65: 40 7C         ; Move to absolute (124,64)
DF67: FC            ; Draw short lines
DF68: 4E            ;     Short line to relative (-4,8)
DF69: C0            ;     Short line to relative (0,-8)
DF6A: 7B            ;     Short line to relative (-10,14)
DF6B: 9C            ;     Short line to relative (-8,-14)
DF6C: D4            ;     Short line to relative (8,-6)
DF6D: E4            ;     Short line to relative (8,-4)
DF6E: E1            ;     Short line to relative (2,-4)
DF6F: E1            ;     Short line to relative (2,-4)
DF70: DD            ;     Short line to relative (-6,-6)
DF71: 1C            ;     Short line to relative (-8,2)
DF72: 96            ;     Short line to relative (12,-14)
DF73: 03            ;     Short line to relative (6,0)
DF74: 00            ;     End of short lines
DF75: 1C 82         ; Move to absolute (130,28)
DF77: FC            ; Draw short lines
DF78: 03            ;     Short line to relative (6,0)
DF79: 45            ;     Short line to relative (10,8)
DF7A: 71            ;     Short line to relative (2,14)
DF7B: DA            ;     Short line to relative (-12,-6)
DF7C: 1E            ;     Short line to relative (-4,2)
DF7D: 11            ;     Short line to relative (2,2)
DF7E: E1            ;     Short line to relative (2,-4)
DF7F: 00            ;     End of short lines
DF80: 30 86         ; Move to absolute (134,48)
DF82: 36 8E         ; Line to absolute (142,54)
DF84: 74 A4         ; Line to absolute (164,116)
```

```
DF86: 84 84         ; Line to absolute (132,132)
DF88: 82 76         ; Line to absolute (118,130)
DF8A: 78 5E         ; Line to absolute (94,120)
DF8C: 5A 6E         ; Line to absolute (110,90)
DF8E: 84 84         ; Line to absolute (132,132)
DF90: 48 6A         ; Line to absolute (106,72)
DF92: FF            ; Start new line
DF93: 40 66         ; Move to absolute (102,64)
DF95: FC            ; Draw short lines
DF96: 1F            ;     Short line to relative (-2,2)
DF97: BD            ;     Short line to relative (-6,-10)
DF98: F1            ;     Short line to relative (2,-2)
DF99: 53            ;     Short line to relative (6,10)
DF9A: 00            ;     End of short lines
DF9B: 42 66         ; Move to absolute (102,66)
DF9D: FC            ; Draw short lines
DF9E: 1E            ;     Short line to relative (-4,2)
DF9F: 32            ;     Short line to relative (4,6)
DFA0: 11            ;     Short line to relative (2,2)
DFA1: 73            ;     Short line to relative (6,14)
DFA2: 00            ;     End of short lines
DFA3: 58 70         ; Move to absolute (112,88)
DFA5: 48 78         ; Line to absolute (120,72)
DFA7: FF            ; Start new line
DFA8: 3E 84         ; Move to absolute (132,62)
DFAA: 14 80         ; Line to absolute (128,20)
DFAC: 34 7A         ; Line to absolute (122,52)
DFAE: 40 7A         ; Line to absolute (122,64)
DFB0: 3C 7C         ; Line to absolute (124,60)
DFB2: 72 80         ; Line to absolute (128,114)
DFB4: 50 82         ; Line to absolute (130,80)
DFB6: 44 82         ; Line to absolute (130,68)
DFB8: 3E 84         ; Line to absolute (132,62)
DFBA: FF            ; Start new line
DFBB: 28 82         ; Move to absolute (130,40)
DFBD: FC            ; Draw short lines
DFBE: FF            ;     Short line to relative (-2,-2)
DFBF: 1E            ;     Short line to relative (-4,2)
DFC0: 11            ;     Short line to relative (2,2)
DFC1: F2            ;     Short line to relative (4,-2)
DFC2: 3F            ;     Short line to relative (-2,6)
```

```
DFC3: 20            ;     Short line to relative (0,4)
DFC4: 0F            ;     Short line to relative (-2,0)
DFC5: C0            ;     Short line to relative (0,-8)
DFC6: FF            ;     Short line to relative (-2,-2)
DFC7: 31            ;     Short line to relative (2,6)
DFC8: 00            ;     End of short lines
DFC9: FE            ; End of image
```

Snake Picture

```
SnakePic:
DFCA: 84 82         ; Move to absolute (130,132)
DFCC: 70 7A         ; Line to absolute (122,112)
DFCE: 5C 7C         ; Line to absolute (124,92)
DFD0: 5E 7E         ; Line to absolute (126,94)
DFD2: 5E 82         ; Line to absolute (130,94)
DFD4: 5C 84         ; Line to absolute (132,92)
DFD6: 70 82         ; Line to absolute (130,112)
DFD8: 80 8C         ; Line to absolute (140,128)
DFDA: 84 88         ; Line to absolute (136,132)
DFDC: 84 72         ; Line to absolute (114,132)
DFDE: 78 6C         ; Line to absolute (108,120)
DFE0: 6A 76         ; Line to absolute (118,106)
DFE2: 78 70         ; Line to absolute (112,120)
DFE4: 7C 74         ; Line to absolute (116,124)
DFE6: 7C 7E         ; Line to absolute (126,124)
DFE8: FF            ; Start new line
DFE9: 64 78         ; Move to absolute (120,100)
DFEB: FC            ; Draw short lines
DFEC: E0            ;     Short line to relative (0,-4)
DFED: E2            ;     Short line to relative (4,-4)
DFEE: EE            ;     Short line to relative (-4,-4)
DFEF: E0            ;     Short line to relative (0,-4)
DFF0: F1            ;     Short line to relative (2,-2)
DFF1: 22            ;     Short line to relative (4,4)
DFF2: EE            ;     Short line to relative (-4,-4)
DFF3: 06            ;     Short line to relative (12,0)
DFF4: 2E            ;     Short line to relative (-4,4)
DFF5: E2            ;     Short line to relative (4,-4)
```

```
DFF6: 11            ;     Short line to relative (2,2)
DFF7: 20            ;     Short line to relative (0,4)
DFF8: 2E            ;     Short line to relative (-4,4)
DFF9: 22            ;     Short line to relative (4,4)
DFFA: 20            ;     Short line to relative (0,4)
DFFB: 00            ;     End of short lines
DFFC: FE            ; End of image

DFFD: 4B 53 4B ; "KSK" Initials of Keith S. Kiyohara, co-creator of the game
```