```
;Adventure Plus, Packrat Edition
;Steve Engelhardt
;07/30/2005
;
      processor 6502
;Default 2600 Constants set up by dissasembler..
VSYNC   =  $00
VBLANK  =  $01
WSYNC   =  $02
RSYNC   =  $03
NUSIZ0  =  $04
NUSIZ1  =  $05
COLUP0  =  $06
COLUP1  =  $07
COLUPF  =  $08
COLUBK  =  $09
CTRLPF  =  $0A
PF0     =  $0D
PF1     =  $0E
PF2     =  $0F
RESP0   =  $10
AUDC0   =  $15
AUDF0   =  $17
AUDV0   =  $19
AUDV1   =  $1A
GRP0    =  $1B
GRP1    =  $1C
ENAM0   =  $1D
ENAM1   =  $1E
ENABL   =  $1F
HMP0    =  $20
VDEL01  =  $26
HMOVE   =  $2A
HMCLR   =  $2B
CXCLR   =  $2C
CXP0FB  =  $32
CXP1FB  =  $33
CXM0FB  =  $34
CXM1FB  =  $35
CXBLPF  =  $36
CXPPMM  =  $37
INPT4   =  $3C
SWCHA   =  $0280
SWCHB   =  $0282
INTIM   =  $0284
TIM64T  =  $0296

       ORG $F000

START:
       JMP    LF2EF   ;3               ;Jump To Start Game


                                       ;Alternate Start
;LF003: .byte $78,$D8,$4C,$06,$F3       ;Setup for 6507, Start with no Variable Initialisation.

LF003: .byte $78                       ;Four Bytes were removed to make room for the extra
                                       ;anti-flicker code for the title screen (Credit to AA
                                       ;Member 'Channel 2').

;Print Display
LF008: STA    HMCLR   ;3        ;Clear horzontal motion.
       LDA    $86     ;3        ;Position Player00 Sprite To
       LDX    #$00    ;2        ;the X Coordinate of Object1.
       JSR    LF0D2   ;6

       LDA    $88     ;3        ;Position Player01 Sprite to
```

```
        LDX    #$01    ;2        ;the X Coordinate of Object2.
        JSR    LF0D2   ;6

        LDA    $8B     ;3        ;Position Ball Strite to
        LDX    #$04    ;2        ;the X Coordinate of the Man.
        JSR    LF0D2   ;6

        STA    WSYNC   ;3        ;Wait for horizontal Blank.
        STA    HMOVE   ;3        ;Apply Horizontal Motion.
        STA    CXCLR   ;3        ;Clear Collision Latches.

       LDA    $8C     ;3        ;Get the Y Coordinate of the Man.
        SEC            ;2
        SBC    #$04    ;2        ;And Adjust it (By Four Scan Lines)
        STA    $8D     ;3        ;for printing (so Y Coordinate Specifies Middle)

LF02C:  LDA    INTIM   ;4        ;Wait for end of the
        BNE    LF02C   ;2        ;current fame.

        LDA    #$00    ;2
        STA    $90     ;3        ;Set Player00 definition index.
        STA    $91     ;3        ;Set Player01 definition index.
        STA    $8F     ;3        ;Set room definition index.
        STA    GRP1    ;3        ;Clear any graphics for Player01.
        LDA    #$01    ;2
        STA    VDEL01  ;3        ;vertically delay Player 01
        LDA    #$68    ;2
        STA    $8E     ;3        ;Set Scan Lind Count.

;Print top line of Room.
        LDY    $8F     ;3        ;Get room definition index.
        LDA    ($80),Y ;5        ;Get first room definition byte.
        STA    PF0     ;3        ;and display.
        INY            ;2
        LDA    ($80),Y ;5        ;Get Next room definition byte.
        STA    PF1     ;3        ;and display.
        INY            ;2
        LDA    ($80),Y ;5        ;Get Last room defintion byte.
        STA    PF2     ;3        ;and display.
        INY            ;2
        STY    $8F     ;3        ;Save for Next Time.

        STA    WSYNC   ;3        ;Wait for Horizontal Blank.
        LDA    #$00    ;2
        STA    VBLANK  ;3        ;Clear any Vertical Blank.
        JMP    LF072   ;3


;Print Player01 (Object 02)
LF05F:  LDA    $8E     ;3        ;Get Current Scan Line.
        SEC            ;2        ;Have we reached Object2's
        SBC    $89     ;3        ;Y Coordinate?
        STA    WSYNC   ;3        ;Wait for Horzonal Blank.
        BPL    LF072   ;2        ;If Not, Branch.

        LDY    $91     ;3        ;Get the Player01 definition index.
        LDA    ($84),Y ;5        ;Get the Next Player01 Definition byte
        STA    GRP1    ;3        ;and display.
        BEQ    LF072   ;2        ;If Zero then Definition finished.

        INC    $91     ;5        ;Goto next Player01 definition byte.

;Print Player00 (Object01), Ball (Man) and Room.
LF072:  LDX    #$00    ;2
        LDA    $8E     ;3        ;Get the Current Scan Line.
        SEC            ;2        ;Have we reached the Object1's
        SBC    $87     ;3        ;Y coordinate?
        BPL    LF084   ;2        ;If not then Branch.
```

```
       LDY    $90     ;3        ;Get Player00 definition index.
       LDA    ($82),Y ;5        ;Get the Next Player00 definition byte.
       TAX            ;2
       BEQ    LF084   ;2        ;If Zero then Definition finished.

       INC    $90     ;5        ;Go to Next Player00 definition byte.

LF084: LDY    #$00    ;2        ;Disable Ball Graphic.
       LDA    $8E     ;3        ;Get Scan line count.
       SEC            ;2        ;Have we reached the Man's
       SBC    $8D     ;3        ;      Y Coordinate?
       AND    #$FC    ;2        ;Mask value to four either side (getting depth of 8)
       BNE    LF091   ;2        ;If Not, Branch.

       LDY    #$02    ;2        ;Enable Ball Graphic.

LF091: LDA    $8E     ;3        ;Get Scan Line Count.
       AND    #$0F    ;2        ;Have we reached a sixteenth scan line.
       BNE    LF0BD   ;2        ;If not, Branch.

       STA    WSYNC   ;3        ;Wait for Horzontal Blank.
       STY    ENABL   ;3        ;Enable Ball (If Wanted)
       STX    GRP0    ;3        ;Display Player 00 definition byte (if wanted)

       LDY    $8F     ;3        ;Get room definition index.
       LDA    ($80),Y ;5        ;Get first room definition byte,
       STA    PF0     ;3        ;and display.
       INY            ;2
       LDA    ($80),Y ;5        ;Get next room definition byte,
       STA    PF1     ;3        ;and display.
       INY            ;2
       LDA    ($80),Y ;5        ;Get next room definition byte,
       STA    PF2     ;3        ;and display.
       INY            ;2
       STY    $8F     ;3        ;Save for Next Time.

LF0B0: DEC    $8E     ;5        ;Goto next scan line.
       LDA    $8E     ;3        ;Get the scan line.
       CMP    #$08    ;2        ;Have we reached to within 8 scanlines of the bottom?
       BPL    LF05F   ;2        ;If not, Branch.

       STA    VBLANK  ;3        ;Turn on VBLANK
       JMP    LF0C6   ;3

;Print Player00 (Object 01) and Ball (Man)
LF0BD: STA    WSYNC   ;3        ;Wait for Horzontal blank.
       STY    ENABL   ;3        ;Enable Ball (If Wanted.)
       STX    GRP0    ;3        ;Display Player00 definition byte (if Wanted).
       JMP    LF0B0   ;3

;Tidy Up
LF0C6: LDA    #$00    ;2
       STA    GRP1    ;3        ;Clear any graphics for Player01
       STA    GRP0    ;3        ;Clear any graphics for Player00
       LDA    #$20    ;2
       STA    TIM64T  ;4        ;Stat Timing this frame using
       RTS            ;6        ;      the 64 bit counter.


;Position Sprite X horizontally.
LF0D2: LDY    #$02    ;2        ;Start with 10 clock cycles (to avoid HBLANK)
       SEC            ;2        ;Divide the Coordinate.
LF0D5: INY            ;2        ;Wanted by Fifteen I.E.
       SBC    #$0F    ;2        ;Get Course Horizontal
       BCS    LF0D5   ;2        ;Value (In Multiples of 5 Clock Cycles
                                ;(Therefore giving 15 Color Cycles)
       EOR    #$FF    ;2        ;Flip remanter to positive value (inverted).
```

```
       SBC    #$06    ;2        ;Convert to left or right of current position.
       ASL            ;2
       ASL            ;2        ;Move to high nybble for TIA
       ASL            ;2        ;horizontal motion.
       ASL            ;2
       STY    WSYNC   ;3        ;Wait for horozontal blank.

LF0E4: DEY            ;2        ;Count down the color
       BPL    LF0E4   ;2        ;cycles (these are 5 machine/15 color cycles).

       STA    RESP0,X ;4        ;Reset the sprite, thus positioning it coursely.
       STA    HMP0,X  ;4        ;Set horizontal (fine) motion of sprite.
       RTS            ;6


;Preform VSYNC
LF0EC: LDA    INTIM   ;4        ;Get Timer Output
       BNE    LF0EC   ;2        ;Wait for Time-Out
       LDA    #$02    ;2        ;
       STA    WSYNC   ;3        ;Wait for horizonal blank.
       STA    VBLANK  ;3        ;Start Vertical Blanking.
       STA    WSYNC   ;3        ;Wait for horizonal blank.
       STA    WSYNC   ;3        ;Wait for horizonal blank.
       STA    WSYNC   ;3        ;Wait for horizonal blank.
       STA    VSYNC   ;3        ;Start verticle sync.
       STA    WSYNC   ;3        ;Wait for horizonal blank.
       STA    WSYNC   ;3        ;Wait for horizonal blank.
       LDA    #$00    ;2        ;
       STA    WSYNC   ;3        ;Wait for horizonal blank.
       STA    VSYNC   ;3        ;End Vertical sync.
       LDA    #$2A    ;2        ;Set clock interval to
       STA    TIM64T  ;4        ;Countdown next frame.
       RTS            ;6

;Setup a room for print.
LF10F: LDA    $8A     ;3        ;Get current room number.
       JSR    LF271   ;6        ;Convert it to an address.
       LDY    #$00    ;2        ;
       LDA    ($93),Y ;5        ;Get low pointer to room
       STA    $80     ;3        ;Graphics
       LDY    #$01    ;2        ;
       LDA    ($93),Y ;5        ;Get high pointer to room
       STA    $81     ;3        ;Graphics

;Check B&W Switch for foom graphics.
       LDA    SWCHB   ;4        ;Get console switches.
       AND    #$08    ;2        ;Check black and white switch
       BEQ    LF133   ;2        ;Branch if B&W.

;Use Color
       LDY    #$02    ;2
       LDA    ($93),Y ;5        ;Get room color
       JSR    LF2D3   ;6        ;Change if necessary
       STA    COLUPF  ;3        ;Put in Playfiled color register.
       JMP    LF13C   ;3

;Use B&W
LF133: LDY    #$03    ;2
       LDA    ($93),Y ;5        ;Get B&W Color
       JSR    LF2D3   ;6        ;Change if necessary
       STA    COLUPF  ;3        ;Put in the Playfield color register.

;Color Background.
LF13C: LDA    #$08    ;2        ;Get light grey background
       JSR    LF2D3   ;6        ;Change if necessary
       STA    COLUBK  ;3        ;Put it in the Background color register.

;Playfield Control.
```

```
       LDY    #$04    ;2
       LDA    ($93),Y ;5        ;Get the playfield control value.
       STA    CTRLPF  ;3        ;And put in the playfield control register.
       AND    #$C0    ;2        ;Get the  flag.
       LSR            ;2
       LSR            ;2
       LSR            ;2        ;Get the first bit into position.
       LSR            ;2
       LSR            ;2
       STA    ENAM1   ;3        ;Enable right hand thin wall. (if wanted - missile01)
       LSR            ;2
       STA    ENAM0   ;3        ;Enable left hand thin wall (if wanted - missle00)

;Get objects to display.
       JSR    LF235   ;6        ;Get next two objects to display.

;Sort out their order.
       LDA    $95     ;3        ;If the object1 is the
       CMP    #$00    ;2        ;Invisible surround
       BEQ    LF168   ;2        ;Then branch to swap (we want it as player01)

       CMP    #$5A    ;2        ;If the first object is the bridge then
       BNE    LF174   ;2        ;Swap the objects (we want it as player01)

       LDA    $96     ;3        ;If the object2 is the
       CMP    #$00    ;2        ;Invisble surround then branch to leave
       BEQ    LF174   ;2        ;it (we want it as player01)

LF168: LDA    $95     ;3
       STA    $D8     ;3
       LDA    $96     ;3
       STA    $95     ;3        ;Swap the objects to print.
       LDA    $D8     ;3
       STA    $96     ;3

;Setup Object1 to print.
LF174: LDX    $95     ;3        ;Get Object1
       LDA    LFF44,X ;4        ;Get low pointer to it's dynamic information.
       STA    $93     ;3
       LDA    LFF45,X ;4        ;Get high pointer to it's dynamic informtion.
       STA    $94     ;3

       LDY    #$01    ;2
       LDA    ($93),Y ;5        ;Get Object1's X coordinate
       STA    $86     ;3        ;and Store for print.
       LDY    #$02    ;2
       LDA    ($93),Y ;5        ;Get Object1's Y coordinate
       STA    $87     ;3        ;and Store for print.

       LDA    LFF46,X ;4        ;Get low pointer to state value.
       STA    $93     ;3
       LDA    LFF47,X ;4        ;Get high pointer to state value.
       STA    $94     ;3
       LDY    #$00    ;2
       LDA    ($93),Y ;5        ;Retrieve Object1's current state.
       STA    $DC     ;3

       LDA    LFF48,X ;4        ;Get low pointer to state information.
       STA    $93     ;3
       LDA    LFF49,X ;4        ;Get high pointer to state information.
       STA    $94     ;3
       JSR    LF2A1   ;6        ;Find current state in the state information.

       INY            ;2        ;Index to the state's corresponding graphic pointer.
       LDA    ($93),Y ;5        ;Get Object1's low graphic address
       STA    $82     ;3        ;and store for print.
       INY            ;2
       LDA    ($93),Y ;5        ;Get Object1's high graphic address
```

```
       STA    $83     ;3        ;and store for print.

;Check B&W for object01
       LDA    SWCHB   ;4        ;Get console switches
       AND    #$08    ;2        ;Check B&W switches.
       BEQ    LF1C5   ;2        ;Branch if B&W.

;Colour
       LDA    LFF4A,X ;4        ;Get Object1's Color.
       JSR    LF2D3   ;6        ;Change if necessary.
       STA    COLUP0  ;3        ;And set color luminance00.
       JMP    LF1CD   ;3

;B&W
LF1C5: LDA    LFF4B,X ;4        ;Get Object's B&W Color.
       JSR    LF2D3   ;6        ;Change if necessary.
       STA    COLUP0  ;3        ;Set color luminance00.

;Object1 Size
LF1CD: LDA    LFF4C,X ;4        ;Get Object1's Size
       ORA    #$10    ;2        ;And set to larger size if necessary.
       STA    NUSIZ0  ;3        ;(Used by bridge and invisible surround)

;Setup Object 2 to Print.
       LDX    $96     ;3        ;Get Object 2
       LDA    LFF44,X ;4
       STA    $93     ;3        ;Get low pointer to it's dynamic information.
       LDA    LFF45,X ;4
       STA    $94     ;3        ;Get high pointer to it's dynamic information.
       LDY    #$01    ;2
       LDA    ($93),Y ;5        ;Get Object2's X coordinate
       STA    $88     ;3        ;and store for print.
       LDY    #$02    ;2
       LDA    ($93),Y ;5        ;Get Object2's Y coordinate
       STA    $89     ;3        ;and store for print.
       LDA    LFF46,X ;4        ;Get low pointer to state value.
       STA    $93     ;3
       LDA    LFF47,X ;4        ;Get high pointer to state value.
       STA    $94     ;3


       LDY    #$00    ;2
       LDA    ($93),Y ;5        ;Retrieve Object2's current state.
       STA    $DC     ;3
       LDA    LFF48,X ;4        ;Get low pointer to state information.
       STA    $93     ;3
       LDA    LFF49,X ;4        ;Get high pointer to state information.
       STA    $94     ;3
       JSR    LF2A1   ;6        ;Find the current state in the state information.

       INY            ;2        ;Index to the state's corresponding graphic pointer.
       LDA    ($93),Y ;5
       STA    $84     ;3        ;Get Object2's low graphic address.
       INY            ;2
       LDA    ($93),Y ;5        ;Get Object2's high graphic address.
       STA    $85     ;3

;Check B&W for Object2
       LDA    SWCHB   ;4        ;Get Console Switches
       AND    #$08    ;2        ;Check B&W Switch.
       BEQ    LF225   ;2        ;If B&W then Branch.

;Color
       LDA    LFF4A,X ;4        ;Get Object2;s Color
       JSR    LF2D3   ;6        ;Change if Necessary.
       STA    COLUP1  ;3        ;and set color luminance01.
       JMP    LF22D   ;3
```

```
;B&W
LF225: LDA    LFF4B,X ;4        ;Get Object's B&W Color.
       JSR    LF2D3   ;6        ;Change if Necessary.
       STA    COLUP1  ;3        ;and set color luminance01.

;Object2 Size
LF22D: LDA    LFF4C,X ;4        ;Get Object2's Size
       ORA    #$10    ;2        ;And set to larger size if necessary.
       STA    NUSIZ1  ;3        ;(Used by bridge and invisible surround)
       RTS            ;6

;Fill cache with two objects in this room.
LF235: LDY    $9C     ;3        ;Get Last Object
       LDA    #$A2    ;2        ;Set cache to
       STA    $95     ;3        ;no-ojects.
       STA    $96     ;3
LF23D: TYA            ;2
       CLC            ;2        ;Goto the next object to
       ADC    #$09    ;2        ;check (add nine).
       CMP    #$A2    ;2        ;Check if over maximum.
       BCC    LF247   ;2
       LDA    #$00    ;2        ;If so, wrap to zero.
LF247: TAY            ;2
       LDA    LFF44,Y ;4        ;Get low byte of object info.
       STA    $93     ;3
       LDA    LFF45,Y ;4        ;Get high byte of object info.
       STA    $94     ;3
       LDX    #$00    ;2
       LDA    ($93,X) ;6        ;Get objects current room.
       CMP    $8A     ;3        ;Is it in this room?
       BNE    LF26A   ;2        ;If not lets try next object (branch)

       LDA    $95     ;3        ;Check first slot.
       CMP    #$A2    ;2        ;If not default (no-object)
       BNE    LF265   ;2        ;then branch.

       STY    $95     ;3        ;Store this object's number to print
       JMP    LF26A   ;3        ;and try for more.

LF265: STY    $96     ;3        ;Store this object's number to print.
       JMP    LF26E   ;3        ;and then give up - no slots free.

LF26A: CPY    $9C     ;3        ;Have we done all the objets?
       BNE    LF23D   ;2        ;If not, continue.

LF26E: STY    $9C     ;3        ;If so, store current count
       RTS            ;6        ;for next time.

;Convert room number to address.
LF271: STA    $D8     ;3        ;Strore room number wanted.
       STA    $93     ;3
       LDA    #$00    ;2        ;Zero the high byte of the
       STA    $94     ;3        ;offset.
       CLC            ;2
       ROL    $93     ;5
       ROL    $94     ;5        ;Multiply room number by eight.
       ROL    $93     ;5
       ROL    $94     ;5
       ROL    $93     ;5
       ROL    $94     ;5

       LDA    $D8     ;3        ;Get the original room number.
       CLC            ;2
       ADC    $93     ;3
       STA    $93     ;3        ;And add it to the offset.
       LDA    #$00    ;2
       ADC    $94     ;3        ;In effect the room number is
       STA    $94     ;3        ;multiplied by nine.
```

```
       LDA    #$1B    ;2
       CLC            ;2
       ADC    $93     ;3        ;Add the room data base address
       STA    $93     ;3        ;to the offset therefore getting
       LDA    #$FE    ;2        ;the final room data address.
       ADC    $94     ;3
       STA    $94     ;3
       RTS            ;6

;Get pointer to current state.
LF2A1: LDY    #$00    ;2
       LDA    $DC     ;3        ;Get the current object state.
LF2A5: CMP    ($93),Y ;5        ;Have we found it in the list of states.
       BCC    LF2B1   ;2        ;If nearing it then found it and return
       BEQ    LF2B1   ;2        ;If found it then return.

       INY            ;2
       INY            ;2        ;Goto next state in list of states.
       INY            ;2
       JMP    LF2A5   ;3
LF2B1: RTS            ;6

;Check for input.
LF2B2: INC    $E5     ;5        ;Increment low count.
       BNE    LF2BE   ;2

       INC    $E6     ;5        ;Increment hight count if
       BNE    LF2BE   ;2        ;needed.

       LDA    #$80    ;2        ;Wrap the high count (indicating
       STA    $E6     ;3        ;timeout) if needed.

LF2BE: LDA    SWCHA   ;4        ;Get joystick values.
       CMP    #$FF    ;2        ;If any movement then branch.
       BNE    LF2CE   ;2

       LDA    SWCHB   ;4        ;Get the consol switches
       AND    #$03    ;2        ;Mast for the reset/select switchs.
       CMP    #$03    ;2        ;Have either of them been used?
       BEQ    LF2D2   ;2        ;If not branch.

LF2CE: LDA    #$00    ;2        ;Zero the high count of the
       STA    $E6     ;3        ;switches or joystick have been used.
LF2D2: RTS            ;6

;Change color if necessary.
LF2D3: LSR            ;2        ;If bit 0 of the color is set
       BCC    LF2DA   ;2        ;then the room is to flash.

       TAY            ;2        ;Use color as an index (usually E5- the low counter).
       LDA.wy $0080,Y ;4        ;Get flash color (usually the low counter.)

LF2DA: LDY    $E6     ;3        ;Get the input counter.
       BPL    LF2E2   ;2        ;If console/joystick moved reciently then branch.

       EOR    $E6     ;3        ;Merge the high counter with the color wanted.
       AND    #$FB    ;2        ;Keep this color bug merge down the luminance.

LF2E2: ASL            ;2        ;And restore original color if necessary.
       RTS            ;6

;Get the address of the dynamic information for an object.
LF2E4: LDA    LFF44,X ;4
       STA    $93     ;3        ;Get and store the low address.
       LDA    LFF45,X ;4
       STA    $94     ;3        ;Get and store the high address.
       RTS            ;6
```

```
;Game Start
LF2EF: SEI            ;2        ;Set Interupts Off
       CLD            ;2
       LDX    #$28    ;2        ;Clear TIA Registers
       LDA    #$00    ;2        ;&04-&2C i.e. blank
LF2F5: STA    NUSIZ0,X;4        ;Everything And Turn.
       DEX            ;2        ;Everything Off.
       BPL    LF2F5   ;2
       TXS            ;2        ;Reset Stack
LF2FB: STA    VSYNC,X ;4        ;Clear &80 to &FF User Vars.
       DEX            ;2
       BMI    LF2FB   ;2
       LDA    #$14    ;         ;This line and the next added to remove
       STA    $D9     ;         ;The flicker in the title screen (Channel2)
       JSR    LF371   ;6        ;Position the thin walls (missiles)
       JSR    LF3D3   ;6        ;Setup objects rooms and positions.
LF306: JSR    LF384   ;6        ;Check for Game Start
       JSR    LFA23   ;6        ;Make noise if necessary
       JSR    LF2B2   ;6        ;Check for input.
       LDA    $DE     ;3        ;Is The Game Active?
       BNE    LF365   ;2        ;If Not Branch..
       LDA    $B9     ;3        ;Get the room the Chalise is in.
       CMP    #$12    ;2        ;Is it in the yellow castle?
       BNE    LF323   ;2        ;If Not Branch..
       LDA    #$FF    ;2
       STA    $DF     ;3        ;Set the note count to maximum.
       STA    $DE     ;3        ;Set the game to inactive.
       LDA    #$00    ;2        ;Set the noise type to end-noise.
       STA    $E0     ;3
LF323: LDY    #$00    ;2        ;Allow joystick read - all movement.
       JSR    LF4C2   ;6        ;Check ball collisions and move ball.
       JSR    LF5D4   ;6        ;Move the Carried Object
       JSR    LF0EC   ;6        ;Wait for VSYNC
       JSR    LF10F   ;6        ;Setup the room and objects for display.
       JSR    LF008   ;6        ;Display the room and objects.
       JSR    LF556   ;6        ;Deal with object pickup and putdown.
       LDY    #$01    ;2        ;Dissalow joystick read - move vertically only.
       JSR    LF4C2   ;6        ;Check ball collisions and move ball.
       JSR    LF9E7   ;6        ;Deal With Invisible Surround Moving.
       JSR    LF0EC   ;6        ;Wait for VSYNC
       JSR    LF8A5   ;6        ;Move and deal with bat.
       JSR    LF93C   ;6        ;Move and deal with portcullises.
       JSR    LF008   ;6        ;Display the room and objects.
       JSR    LF7CB   ;6        ;Move and deal with the green dragon.
       JSR    LF7B0   ;6        ;Move and deal with the yellow dragon.
       JSR    LF0EC   ;6        ;Wait for VSYNC.
       LDY    #$02    ;2        ;Dissalow joystic read/bridge check - move horrizonally only.
       JSR    LF4C2   ;6        ;Check ball collisions and move ball.
       JSR    LF795   ;6        ;Move and deal with red dragon.
       JSR    LF9B3   ;6        ;Deal with the magnet.
       JSR    LF008   ;6        ;Display the room and objects.
       JMP    LF306   ;3

;Non Active Game Loop.
LF365: JSR    LF0EC   ;6        ;Wait for VSYNC
       JSR    LF008   ;6        ;Display the room and objects.
       JSR    LF10F   ;6        ;Set up room and objects for display.
       JMP    LF306   ;3

;Position missiles to  areas.
LF371: LDA    #$0D    ;2        ;Position missile 00 to
       LDX    #$02    ;2        ;(0D,00) - left thin wall.
       JSR    LF0D2   ;6
       LDA    #$96    ;2        ;Position missile 01 to
       LDX    #$03    ;2        ;(96,00) - right thin wall.
       JSR    LF0D2   ;6
       STA    WSYNC   ;3        ;Wait for horizonal blank.
```

```
        STA    HMOVE   ;3        ;Apply the horizonal move.
        RTS            ;6

LF384:  LDA    SWCHB   ;4        ;Get the console switches.
        EOR    #$FF    ;2        ;Flip (as reset active low).
        AND    $92     ;3        ;Compare with what was before
        AND    #$01    ;2        ;And check only the reset switch
        BEQ    LF3B5   ;2        ;If no reset then branch.
        LDA    $DE     ;3        ;Has the Game Started?
        CMP    #$FF    ;2        ;If not then branch.
        BEQ    LF3D3   ;2
        LDA    #$11   ;2        ;Get the yellow castle room.  ; WAS $11
        STA    $8A     ;3              ;Make it the current room. ; current room start $8A;LOC3
        STA    $E2     ;3              ;Make it the previous room.
        LDA    #$50    ;2        ;Get the X coordinate.
        STA    $8B     ;3              ;Make it the current ball X coordinate.
        STA    $E3     ;3              ;Make it the previous ball X coordinate.
        LDA    #$20    ;2        ;Get the Y coordinate.
        STA    $8C     ;3              Make it the current ball Y coordinate.
        STA    $E4     ;3              Make it the previous ball Y coordinate.
        LDA    #$00    ;2
        STA    $A8     ;3        ;Set the red dragon's state to OK.
        STA    $AD     ;3        ;Set the yellow dragon's state to OK.
        STA    $B2     ;3        ;Set the green dragon's state to OK.
        STA    $DF     ;3        ;Set the note count to zero.. (ops!??)
        LDA    #$A2    ;2
        STA    $9D     ;3        ;Set no object being carried.
LF3B5:  LDA    SWCHB   ;4        ;Get the console switches.
        EOR    #$FF    ;2        ;Flip (as select active low)
        AND    $92     ;3        ;Compare with what was before.
        AND    #$02    ;2        ;And check only the select switch.
        BEQ    LF40C   ;2        ;Branch if select not being used.
        LDA    $8A     ;3        ;Get the Current Room.
        CMP    #$00    ;2        ;Is it the  room?
        BNE    LF3D3   ;2        ;Branch if not.
        LDA    $DD     ;3        ;Increment the level.
        CLC            ;2        ;Number (by two).
        ADC    #$02    ;2
        CMP    #$06    ;2        ;Have we reached the maximum?
        BCC    LF3D1   ;2
        LDA    #$00    ;2        ;If yep then set back to zero.
LF3D1:  STA    $DD     ;3        ;Store the new level number.
LF3D3:  LDA    #$00    ;2        ;Set the current room to the
        STA    $8A     ;3        ; room.
        STA    $E2     ;3        ;And the previous room.
        LDA    #$00    ;2        ;Set the ball's Y coordinate to zero.
        STA    $8C     ;3        ;And the previous Y coordinate.
        STA    $E4     ;3        ;(So can't be seen.)
        LDY    $DD     ;3        ;Get the level number.
        LDA    LF45A,Y ;4        ;Get the low pointer to object locations.
        STA    $93     ;3
        LDA    LF45B,Y ;4        ;Get the high pointer to object locations.
        STA    $94     ;3
        LDY    #$30    ;2        ;Copy all the objects dynamic information.
LF3ED:  LDA    ($93),Y ;5        ;(the rooms and positions) into
        STA.wy $00A1,Y ;5        ;the working area.
        DEY            ;2
        BPL    LF3ED   ;2
        LDA    $DD     ;3        ;Get the level number.
        CMP    #$04    ;2        ;Branch if level one.
        BCC    LF404   ;2        ;Or two (Where all objects are in defined areas.)
        JSR    LF412   ;6        ;Put some objects in random rooms.
        JSR    LF0EC   ;6        ;Wait for VSYNC
        JSR    LF008   ;6        ;Display rooms and objects.
LF404:  LDA    #$00    ;2        ;Signal that the game has started.
        STA    $DE     ;3
        LDA    #$A2    ;2        ;Set no object being carried.
        STA    $9D     ;3
```

```
LF40C: LDA    SWCHB   ;4        ;Store the current console switches
       STA    $92     ;3
       RTS            ;6

;Put objects in random rooms for level 3.
LF412: LDY    #$1E    ;2        ;For each of the eleven objects..
LF414: LDA    $E5     ;3        ;Get the low input counter as seed.
       LSR            ;2
       LSR            ;2
       LSR            ;2        ;Generate a psudo-random
       LSR            ;2        ;room number.
       LSR            ;2
       SEC            ;2
       ADC    $E5     ;3        ;Store the low input counter.
       STA    $E5     ;3
       AND    #$1F    ;2        ;Trim so represents a room value.
       CMP    LF43A,Y ;4        ;If it is less than the
       BCC    LF414   ;2        ;lower bound for object then get another.
       CMP    LF43B,Y ;4        ;If it equals or is
       BEQ    LF42E   ;2        ;Less than the higher bound for object
       BCS    LF414   ;2        ;Then continue (branch if higher)
LF42E: LDX    LF439,Y ;4        ;Get the dynamic data index for this object
       STA    VSYNC,X ;4        ;Store the new room value.
       DEY            ;2
       DEY            ;2        ;Goto the next object.
       DEY            ;2
       BPL    LF414   ;2        ;Untill all done
       RTS            ;6

;Room Bounds Data.
;Ex. the chalise at location &B9 can only exist in rooms 13-1A for
;     level 3.

LF439: .byte $B9                ;
LF43A: .byte $13                ;Chalice
LF43B: .byte $1A                ;
       .byte $A4,$01,$1D        ;Red Dragon
       .byte $A9,$01,$1D        ;Yellow Dragon
       .byte $AE,$01,$1D        ;Green Dragon
       .byte $B6,$01,$1D        ;Sword
       .byte $BC,$01,$1D        ;Bridge
       .byte $BF,$01,$1D        ;Yellow Key
       .byte $C2,$01,$16        ;White Key
       .byte $C5,$01,$12        ;Black Key
       .byte $CB,$01,$1D        ;Bat
       .byte $B3,$01,$1D        ;Magnet

LF45A: .byte $60                ;Pointer to object locations for game 01.
LF45B: .byte $F4                ;continued
       .byte $91,$F4            ;Pointer to object locations for game 02.
       .byte $91,$F4            ;Pointer to object locations for game 03.

;Object Locations, Game 1 (Room, X, Y, Movement, State)

        .byte $15,$51,$12        ;Black Dot
        .byte $01,$50,$20,$00,$00 ;Red Dragon
        .byte $10,$50,$20,$00,$00 ;Yellow Dragon
        .byte $1D,$50,$20,$00,$00 ;Green Dragon
        .byte $1B,$80,$20        ;Magnet
        .byte $06,$08,$20        ;sword
        .byte $1C,$30,$20        ;Chalice
        .byte $1D,$20,$40        ;Bridge
        .byte $0E,$20,$40        ;Yellow Key
        .byte $09,$20,$40        ;White Key
        .byte $12,$20,$20        ;Black Key
        .byte $1C                ;Portcullis State
        .byte $1C
        .byte $1C
```

```
        .byte $0D,$20,$20,$00,$00 ;Bat
        .byte $78,$00             ;Bat (carrying, Fed-Up)

;Object Locations, Games 2 & 3 (Room, X, Y, Movement, State);LOC2

        .byte $15,$51,$12         ;Black Dot
        .byte $14,$50,$20,$A0,$00 ;Red Dragon
        .byte $19,$50,$20,$A0,$00 ;Yellow Dragon
        .byte $09,$50,$20,$A0,$00 ;Green Dragon
        .byte $0B,$40,$40         ;Magnet
        .byte $06,$10,$20         ;Sword
        .byte $1E,$50,$68         ;Chalice
        .byte $03,$20,$40         ;Bridge; $09 in ADV+
        .byte $1C,$80,$20         ;Yellow Key
        .byte $0E,$8E,$3C         ;White Key
        .byte $1D,$20,$40         ;Black Key
        .byte $1C                 ;Portcullis State
        .byte $1C
        .byte $1C
        .byte $1B,$20,$20,$90,$00 ;Bat
        .byte $78,$00             ;Bat (carrying, Fed-Up)

      ; Room Locations (For Reference)
      ;00; Number Room                          ;0F; White Castle
      ;01; (Top Acess) Reflected/8 Clock Ball   ;10; Black Castle
      ;02; (Top Access) below yellow castle     ;11; Yellow Castle
      ;03; Left of Name                         ;12; Yellow Castle Entry
      ;04; Top of Blue Maze                     ;13; Black Maze #1
      ;05; Blue Maze #1                         ;14; Black Maze #2
      ;06; Bottom of Blue Maze                  ;15; Black Maze #3
      ;07; Center of Blue Maze                  ;16; Black Maze Entry
      ;08; Blue Maze Entry                      ;17; Red Maze #1
      ;09; Maze Middle                          ;18; Top of Red Maze
      ;0A; Maze Entry                           ;19; Bottom of Red Maze
      ;0B; Maze Side                            ;1A; White Castle Entry
      ;0C; (Side Corridor)                      ;1B; Black Castle Entry
      ;0D; (Side Corridor)                      ;1C; Bottom Entry - Other Purple Room
      ;0E; (Top Entry Room)                     ;1D; (Top Entry Room)
                                                ;1E; Name Room

        ;Original Object Locations
        ;Object locations (room and coordinate) for game 01.
        ;
        ;      .byte $15,$51,$12          ;Black dot    (Room, X, Y)
        ;      .byte $0E,$50,$20,$00,$00  ;Red Dragon   (Room, X, Y, Movement, State)
        ;      .byte $01,$51,$20,$00,$00  ;Yellow Dragon(Room, X, Y, Movement, State)
        ;      .byte $1D,$50,$20,$00,$00  ;Green Dragon (Room, X, Y, Movement, State)
        ;      .byte $1B,$85,$39          ;Magnet       (Room,X,Y)
        ;      .byte $06,$0A,$14          ;Sword        (Room,X,Y)
        ;      .byte $1C,$0F,$60          ;Chalice      (Room,X,Y)
        ;      .byte $1D,$26,$3B          ;Bridge       (Room,X,Y)
        ;      .byte $06,$4F,$19          ;Yellow Key   (Room,X,Y)
        ;      .byte $0E,$20,$40          ;White Key    (Room,X,Y)
        ;      .byte $12,$8E,$5F          ;Black Key    (Room,X,Y)
        ;      .byte $1C                  ;Portcullis State
        ;      .byte $1C                  ;Portcullis State
        ;      .byte $1C                  ;Portcullis State
        ;      .byte $1A,$20,$20,$00,$00  ;Bat          (Room, X, Y, Movement, State)
        ;      .byte $78,$00              ;Bat          (Carrying, Fed-Up)
        ;
        ;Object locations (room and coordinate) for Games 02 and 03.
        ;
        ;      .byte $15,$51,$12          ;Black Dot    (Room,X,Y)
        ;      .byte $14,$50,$20,$A0,$00  ;Red Dragon   (Room,X,Y,Movement,State)
        ;      .byte $19,$50,$20,$A0,$00  ;Yellow Dragon(Room,X,Y,Movement,State)
        ;      .byte $04,$50,$20,$A0,$00  ;Green Dragon (Room,X,Y,Movement,State)
        ;      .byte $0E,$80,$20          ;Magnet       (Room,X,Y)
        ;      .byte $11,$20,$20          ;Sword        (Room,X,Y)
```

```
        ;      .byte $14,$30,$20          ;Chalise      (Room,X,Y)
        ;      .byte $0B,$40,$40          ;Bridge       (Room,X,Y)
        ;      .byte $09,$20,$40          ;Yellow Key   (Room,X,Y)
        ;      .byte $06,$20,$40          ;White Key    (Room,X,Y)
        ;      .byte $19,$20,$40          ;Black Key    (Room,X,Y)
        ;      .byte $1C                  ;Portcullis State
        ;      .byte $1C                  ;Portcullis State
        ;      .byte $1C                  ;Portcullis State
        ;      .byte $02,$20,$20,$90,$00  ;Bat          (Room,X,Y,Movement,State)
        ;      .byte $78,$00              ;Bat          (Carrying, Fed-Up)



;Check ball collisions and move ball.
LF4C2: LDA    CXBLPF  ;3
       AND    #$80    ;2        ;Get ball-playfield collision
       BNE    LF4F5   ;2        ;Branch if collision (Player-Wall)

       LDA    CXM0FB  ;3
       AND    #$40    ;2        ;Get ball-missile00 collision.
       BNE    LF4F5   ;2        ;Branch if collision. (Player-Left Thin)

       LDA    CXM1FB  ;3
       AND    #$40    ;2        ;Get ball-missile01 collision.
       BEQ    LF4DA   ;2        ;Branch if no collision.

       LDA    $96     ;3        ;If object2 (to print) is
       CMP    #$87    ;2        ;not the black dot then collide.
       BNE    LF4F5   ;2

LF4DA: LDA    CXP0FB  ;3
       AND    #$40    ;2        ;Get ball-player00 collision.
       BEQ    LF4E6   ;2        ;If no collision then branch.

       LDA    $95     ;3        ;If object1 (to print) is
       CMP    #$00    ;2        ;not the invisible surround then
       BNE    LF4F5   ;2        ;branch (collision)

LF4E6: LDA    CXP1FB  ;3
       AND    #$40    ;2        ;Get ball-player01 collision.
       BEQ    LF51F   ;2        ;If no collision then branch.

       LDA    $96     ;3        ;If player01 to print is
       CMP    #$00    ;2        ;not the invisible surround then
       BNE    LF4F5   ;2        ;branch (collision)

       JMP    LF51F   ;3        ;No collision - branch.

;Player collided (with something)
LF4F5: CPY    #$02    ;2        ;Are we checking for the bridge?
       BNE    LF52F   ;2        ;If not, branch.

       LDA    $9D     ;3        ;Get the object being carried.
       CMP    #$5A    ;2        ;Branch if it is the bridge.
       BEQ    LF52F   ;2

       LDA    $8A     ;3        ;Get the current room.
       CMP    $BC     ;3        ;Is the bridge in this room.
       BNE    LF52F   ;2        ;If not branch.

;Check going through the bridge.
       LDA    $8B     ;3        ;Get the ball's X coordinate.
       SEC            ;2
       SBC    $BD     ;3        ;Subtract the bridge's X coordinate.
       CMP    #$0A    ;2        ;If less than &0A then forget it.
       BCC    LF52F   ;2

       CMP    #$17    ;2        ;If more than &17 then forget it.
```

```
       BCS    LF52F   ;2

       LDA    $BE     ;3        ;Get the bridge's Y coordinate.
       SEC            ;2
       SBC    $8C     ;3        ;Subtract the ball's Y coordinate.
       CMP    #$FC    ;2
       BCS    LF51F   ;2        ;If more than &FC then going through bridge.

       CMP    #$19    ;2        ;If more than &19 then forget it.
       BCS    LF52F   ;2

;No collision (and going through bridge)
LF51F: LDA    #$FF    ;2        ;Reset the joystick input.
       STA    $99     ;3
       LDA    $8A     ;3        ;Get the current room.
       STA    $E2     ;3        ;and store temporarily.
       LDA    $8B     ;3        ;Get the ball's X coordinate.
       STA    $E3     ;3        ;and store temporarily.
       LDA    $8C     ;3        ;Get the ball's Y coordinate.
       STA    $E4     ;3        ;And Store Temporarily.

;C ???
LF52F: CPY    #$00    ;2        ;???Is game in first phase?
       BNE    LF538   ;2        ;If not, don't bother with joystick read.

       LDA    SWCHA   ;4        ;Read joysticks.
       STA    $99     ;3        ;and store value.

LF538: LDA    $E2     ;3        ;Get Temporary room.
       STA    $8A     ;3        ;      and make it the current room.
       LDA    $E3     ;3        ;Get temporary X coordinate
       STA    $8B     ;3        ;      and make it the man's X coordinate.
       LDA    $E4     ;3        ;Get temporary Y coordinate
       STA    $8C     ;3        ;      and make it the man's Y coordinate.

       LDA    $99     ;3        ;Get the Joystick position.
       ORA    LF553,Y ;4        ;Merge out movement not allowed in this phase.
       STA    $9B     ;3        ;And store cooked movement.

       LDY    #$03    ;2        ;Set the delta for the ball.
       LDX    #$8A    ;2        ;Point to ball's coordiates.
       JSR    LF5FF   ;6        ;Move the ball
       RTS            ;6

;Joystick Merge Values
LF553: .byte $00,$C0,$30        ;No change, No horizontal, No vertical.

;Deal with object pickup and putdown.
LF556: ROL    INPT4   ;5        ;Get joystick trigger.
       ROR    $D7     ;5        ;Merget into joystick record.
       LDA    $D7     ;3        ;Get joystick record.
       AND    #$C0    ;2        ;Merget out previous presses.
       CMP    #$40    ;2        ;Was it previously pressed?
       BNE    LF572   ;2        ;If not branch.

       LDA    #$A2    ;2
       CMP    $9D     ;3        ;If nothing is being carried
       BEQ    LF572   ;2        ;then branch.

       STA    $9D     ;3        ;Drop object.
       LDA    #$04    ;2        ;Set noise type to four.
       STA    $E0     ;3
       LDA    #$04    ;2        ;Set noise count to four.
       STA    $DF     ;3

LF572: LDA    #$FF    ;2        ;????
       STA    $98     ;3
```

```
;Check for collision.
       LDA    CXP0FB  ;3
       AND    #$40    ;2        ;Get Ball-Player00 collision.
       BEQ    LF583   ;2        ;If nothing occured then branch.

;With Player00
       LDA    $95     ;3        ;Get type of Player00
       STA    $97     ;3        ;And Store.
       JMP    LF593   ;3        ;Deal with collision.

LF583: LDA    CXP1FB  ;3
       AND    #$40    ;2        ;Get Ball-Player01 collision.
       BEQ    LF590   ;2        ;If nothing has happened, branch.

       LDA    $96     ;3        ;Get type of Player01
       STA    $97     ;3        ;and store.
       JMP    LF593   ;3        ;Deal with collision.

LF590: JMP    LF5D3   ;3        ;Deal with no collision (return).

;Collision occured.
LF593: LDX    $97     ;3        ;Get the object collided with.
       JSR    LF2E4   ;6        ;Get it's dynamic information.
       LDA    $97     ;3        ;Get the object collided with.
       CMP    #$51    ;2        ;Is it carriable?
       BCC    LF5D3   ;2        ;If not, branch.

       LDY    #$00    ;2
       LDA    ($93),Y ;5        ;Get the object's room.
       CMP    $8A     ;3        ;Is it in the current room?
       BNE    LF5D3   ;2        ;If not, branch.

       LDA    $97     ;3        ;Get the object collided with.
       CMP    $9D     ;3        ;Is it the object being carried?
       BEQ    LF5B4   ;2        ;If so, branch (and actually pick it up.)

       LDA    #$05    ;2        ;Set noise type to five.
       STA    $E0     ;3
       LDA    #$04    ;2        ;Set noise type to four.
       STA    $DF     ;3

LF5B4: LDA    $97     ;3        ;Set the object as being
       STA    $9D     ;3        ;      carried.

       LDX    $93     ;3        ;Get the dynamice address low byte.
       LDY    #$06    ;2
       LDA    $99     ;3        ;????
       JSR    LF6AC   ;6        ;????

       LDY    #$01    ;2
       LDA    ($93),Y ;5        ;Get the object's X coordinate.
       SEC            ;2
       SBC    $8B     ;3        ;Subtract the ball's X coordinate.
       STA    $9E     ;3        ;and store the difference.
       LDY    #$02    ;2
       LDA    ($93),Y ;5        ;Get the object's Y coordinate.
       SEC            ;2
       SBC    $8C     ;3        ;Subtract the Ball's Y coordinate.
       STA    $9F     ;3        ;and store the difference.

;No collision
LF5D3: RTS            ;6


;Move the carried object
LF5D4: LDX    $9D     ;3        ;Get the object being carried.
       CPX    #$A2    ;2        ;If nothing then branch (return)
       BEQ    LF5FE   ;2
```

```
       JSR    LF2E4   ;6        ;Get it's dynamic information.
       LDY    #$00    ;2

       LDA    $8A     ;3        ;Get the current room.
       STA    ($93),Y ;6        ;and stroe the object's current room.
       LDY    #$01    ;2

       LDA    $8B     ;3        ;Get the ball's X coordinate.
       CLC            ;2
       ADC    $9E     ;3        ;Add the X difference.
       STA    ($93),Y ;6        ;and store as the object's X coordinate.
       LDY    #$02    ;2
       LDA    $8C     ;3        ;Get the ball's Y coordinate.
       CLC            ;2
       ADC    $9F     ;3        ;Add the Y difference.
       STA    ($93),Y ;6        ;and store as the object's Y coordinate.

       LDY    #$00    ;2        ;Set no delta.
       LDA    #$FF    ;2        ;Set no movement.
       LDX    $93     ;3        ;Get the object's dynamic address.
       JSR    LF5FF   ;6        ;Move the object.
LF5FE: RTS            ;6


;Move the object in a direction by delta.
LF5FF: JSR    LF6AC   ;6        ;Move the object by delta.
       LDY    #$02    ;2        ;Set to do the three
LF604: STY    $9A     ;3        ;portcullises.
       LDA.wy $00C8,Y ;4        ;Get the portal state.
       CMP    #$1C    ;2        ;Is it in a closed state?
       BEQ    LF62F   ;2        ;If not, next portal.

;Deal with object moving out of a castle.
       LDY    $9A     ;3        ;Get port number.
       LDA    VSYNC,X ;4        ;Get object's room number.
       CMP    LF9AD,Y ;4        ;Is it in a castle entry room.
       BNE    LF62F   ;2        ;If not, next portal.

       LDA    WSYNC,X ;4        ;Get the object's Y coordinate.
       CMP    #$0D    ;2        ;Is it above &OD i.e at the bottom.
       BPL    LF62F   ;2        ;If so then branch.

       LDA    LF9B0,Y ;4        ;Get the castle room.
       STA    VSYNC,X ;4        ;And put the object in the castle room.
       LDA    #$50    ;2
       STA    VBLANK,X;4        ;Set the object's new X coordinate.
       LDA    #$2C    ;2
       STA    WSYNC,X ;4        ;Set the new object's Y coordinate.
       LDA    #$01    ;2
       STA.wy $00C8,Y ;5        ;Set the portcullis state to 01.
       RTS            ;6

LF62F: LDY    $9A     ;3        ;Get the portcullis number.
       DEY            ;2        ;goto next,
       BPL    LF604   ;2        ;and continue.

;Check and Deal with Up.
       LDA    WSYNC,X ;4        ;Get the object's Y coordinate.
       CMP    #$6A    ;2        ;Has it reched above the top.
       BMI    LF643   ;2        ;If not, branch.

       LDA    #$0D    ;2        ;Set new Y coordinate to bottom.
       STA    WSYNC,X ;4
       LDY    #$05    ;2        ;Get the direction wanted.
       JMP    LF69F   ;3        ;Go and get new room.

;Check and Deal with Left.
```

```
LF643: LDA    VBLANK,X;4        ;Get the object's X coordinate.
       CMP    #$03    ;2        ;Is it Three or less?
       BCC    LF650   ;2        ;IF so, branch.  (off to left)

       CMP    #$F0    ;2        ;Is it's &F0 or more.
       BCS    LF650   ;2        ;If so, branch.  (off to right)

       JMP    LF662   ;3

LF650: CPX    #$8A    ;2        ;Are we dealling with the ball?
       BEQ    LF659   ;2        ;If so Branch.

       LDA    #$9A    ;2        ;Set new X coordinate for the others.
       JMP    LF65B   ;3

LF659: LDA    #$9E    ;2        ;Set new X coordinate for the ball.

LF65B: STA    VBLANK,X;4        ;Store the next X coordinate.
       LDY    #$08    ;2        ;And get the direction wanted.
       JMP    LF69F   ;3        ;Go and get new room.

;Check and Deal with Down.
LF662: LDA    WSYNC,X ;4        ;Get object's Y coordinate.
       CMP    #$0D    ;2        ;If it's greater than &0D then
       BCS    LF671   ;2        ;Branch.

       LDA    #$69    ;2        ;Set new Y coordinate.
       STA    WSYNC,X ;4
       LDY    #$07    ;2        ;Get the direction wanted.
       JMP    LF69F   ;3        ;Go and get new room.

;Check and Deal with Right.
LF671: LDA    VBLANK,X;4        ;Get the object's X coordinate.
       CPX    #$8A    ;2        ;Are we dealing with the ball.
       BNE    LF692   ;2        ;Branch if not.

       CMP    #$9F    ;2        ;Has the object reached the right?
       BCC    LF6AB   ;2        ;Branch if not.

       LDA    VSYNC,X ;4        ;Get the Ball's Room.
       CMP    #$03    ;2        ;Is it room #3 (Right to secret room)
       BNE    LF696   ;2        ;Branch if not.

       LDA    $A1     ;3        ;Check the room of the black dot.
       CMP    #$15    ;2        ;Is it in the hidden room area?
       BEQ    LF696   ;2        ;If so, Branch.

;Manually change to secret room.
       LDA    #$1E    ;2        ;Set room to secret room.
       STA    VSYNC,X ;4        ;And make it current.
       LDA    #$03    ;2        ;Set the X coordinate.
       STA    VBLANK,X;4
       JMP    LF6AB   ;3        ;And Exit.

LF692: CMP    #$9B    ;2        ;Has the object reached the right of the screen?
       BCC    LF6AB   ;2        ;Branch if not (no room change)

LF696: LDA    #$03    ;2        ;Set the next X coordinate.
       STA    VBLANK,X;4
       LDY    #$06    ;2        ;And get the direction wanted.
       JMP    LF69F   ;3        ;Get the new room.

;Get new room
LF69F: LDA    VSYNC,X ;4        ;Get the object's room.
       JSR    LF271   ;6        ;Convert it to an address.
       LDA    ($93),Y ;5        ;Get the adjacent room.
       JSR    LF6D5   ;6        ;Deal with the level differences.
       STA    VSYNC,X ;4        ;and store as new object's room.
```

```
LF6AB: RTS            ;6

;Move the object in direction by delta.
LF6AC: STA    $9B     ;3        ;Stored direction wanted.
LF6AE: DEY            ;2        ;Count down the delta.
       BMI    LF6D4   ;2

       LDA    $9B     ;3        ;Get direction wanted.
       AND    #$80    ;2        ;Check for right move.
       BNE    LF6B9   ;2        ;If no move right then branch.

       INC    VBLANK,X;6        ;Increment the X coordinate.

LF6B9: LDA    $9B     ;3        ;Get the direction wanted.
       AND    #$40    ;2        ;Check for left move.
       BNE    LF6C1   ;2        ;If no move left then branch.

       DEC    VBLANK,X;6        ;Decrement the X coordinate.

LF6C1: LDA    $9B     ;3        ;Get the direction wanted.
       AND    #$10    ;2        ;Check for move up.
       BNE    LF6C9   ;2        ;If no move up then branch.
       INC    WSYNC,X ;6

LF6C9: LDA    $9B     ;3        ;Get direction wanted.
       AND    #$20    ;2        ;Check for move down.
       BNE    LF6D1   ;2        ;If no move down the branch.

       DEC    WSYNC,X ;6        ;Decrement the Y coordinate.

LF6D1: JMP    LF6AE   ;3        ;Keep going until delta finished.
LF6D4: RTS            ;6


;Adjust room for different levels.
LF6D5: CMP    #$80    ;2        ;Is the room number
       BCC    LF6E8   ;2        ;above &80?

       SEC            ;2
       SBC    #$80    ;2        ;Remove the &80 flag and
       STA    $D8     ;3        ;store the room number.
       LDA    $DD     ;3        ;Get the level number.
       LSR            ;2        ;Devide it by two.
       CLC            ;2
       ADC    $D8     ;3        ;Add to the original room.
       TAY            ;2
       LDA    LFF32,Y ;4        ;Use as an offset to get the next room.

LF6E8: RTS            ;6


;Get player-ball collision.
LF6E9: CMP    $95     ;3        ;Is it the rist object?
       BEQ    LF6F4   ;2        ;YES - Then Branch.

       CMP    $96     ;3        ;Is it the second object?
       BEQ    LF6F9   ;2        ;YES - Then Branch.

       LDA    #$00    ;2        ;Otherewise nothing.
       RTS            ;6

LF6F4: LDA    CXP0FB  ;3        ;Get player00-ball collision.
       AND    #$40    ;2
       RTS            ;6

LF6F9: LDA    CXP1FB  ;3        ;Get player01-ball collision.
       AND    #$40    ;2
```

```
       RTS            ;6


;Find which object has hit object wanted.
LF6FE: LDA    CXPPMM  ;3        ;Get Player00-Player01
       AND    #$80    ;2        ;collision.
       BEQ    LF70C   ;2        ;If nothing, Branch.

       CPX    $95     ;3        ;Is object 1 the one being hit?
       BEQ    LF70F   ;2        ;If so, Branch.

       CPX    $96     ;3        ;Is object 2 the one being hit?
       BEQ    LF712   ;2        ;If so, Branch.

LF70C: LDA    #$A2    ;2        ;Therefore select the other.
       RTS            ;6

LF70F: LDA    $96     ;3        ;Therefore select the other.
       RTS            ;6

LF712: LDA    $95     ;3        ;Therefore select the other.
       RTS            ;6


;Move object.
LF715: JSR    LF728   ;6        ;Get liked object and movement.
       LDX    $D5     ;3        ;Get dynamic data address.
       LDA    $9B     ;3        ;Get Movement.
       BNE    LF720   ;2        ;If movement then branch.

       LDA    RSYNC,X ;4        ;Use old movement.

LF720: STA    RSYNC,X ;4        ;Stoe the new movement.
       LDY    $D4     ;3        ;Get the object's Delta.
       JSR    LF5FF   ;6        ;Move the object.
       RTS            ;6


;Find liked object and get movement.
LF728: LDA    #$00    ;2        ;Set index to zero.
       STA    $E1     ;3

LF72C: LDY    $E1     ;3        ;Get index.
       LDA    ($D2),Y ;5        ;Get first object.
       TAX            ;2
       INY            ;2
       LDA    ($D2),Y ;5        ;Get second object.
       TAY            ;2
       LDA    VSYNC,X ;4        ;Get object1;s room.
       CMP.wy $0000,Y ;4        ;Combare with object2's room.
       BNE    LF748   ;2        ;If not the same room then branch.

       CPY    $D6     ;3        ;Have we matched the second object
       BEQ    LF748   ;2        ;for difficulty (if so, carry on).

       CPX    $D6     ;3        ;Have we matched the first object
       BEQ    LF748   ;2        ;for difficulty (if so, carry on).

       JSR    LF757   ;6        ;Get object's movement.
       RTS            ;6

LF748: INC    $E1     ;5        ;Increment the index.
       INC    $E1     ;5
       LDY    $E1     ;3        ;Get the index number.
       LDA    ($D2),Y ;5        ;Check for end of sequence.
       BNE    LF72C   ;2        ;If not branch.

       LDA    #$00    ;2        ;Set no move if no
```

```
       STA    $9B     ;3        ;liked object is found.
       RTS            ;6


;Work out object's movement.
LF757: LDA    #$FF    ;2        ;Set object movement to none.
       STA    $9B     ;3

       LDA.wy $0000,Y ;4        ;Get oject2's room.
       CMP    VSYNC,X ;4        ;Compare it with object's room.
       BNE    LF792   ;2        ;If not the same, forget it.

       LDA.wy $0001,Y ;4        ;Get Object2's X coordinate.
       CMP    VBLANK,X;4        ;Get Object1;s X coordinate.
       BCC    LF774   ;2        ;If Object2 to left of Object1 then Branch.
       BEQ    LF77A   ;2        ;If Object2 on Object1 then Branch.

       LDA    $9B     ;3        ;Get Object Movement.
       AND    #$7F    ;2        ;Signal a move right.
       STA    $9B     ;3
       JMP    LF77A   ;3        ;Now try Vertical.

LF774: LDA    $9B     ;3        ;Get object movent.
       AND    #$BF    ;2        ;Signal a move left.
       STA    $9B     ;3

LF77A: LDA.wy $0002,Y ;4        ;Get Object2's Y Coordinate.
       CMP    WSYNC,X ;4        ;Get Object1's X Coordinate.
       BCC    LF78C   ;2        ;If Object2 Below Object1 Then Branch.
       BEQ    LF792   ;2        ;If Object2 on Object1 Then Branch.

       LDA    $9B     ;3        ;Get Object Movement.
       AND    #$EF    ;2        ;Signal a move up.
       STA    $9B     ;3
       JMP    LF792   ;3        ;Jump to Finish.

LF78C: LDA    $9B     ;3        ;Get Object Movement.
       AND    #$DF    ;2        ;Signal a move down.
       STA    $9B     ;3

LF792: LDA    $9B     ;3        ;Get the Move.
       RTS            ;6


;Move the Red Dragon
LF795: LDA    #$A7    ;2
       STA    $D2     ;3        ;Set the Low address of Object Store.
       LDA    #$F7    ;2
       STA    $D3     ;3        ;Set the High address of Object Store.
       LDA    #$03    ;2
       STA    $D4     ;3        ;Set the Dragon's Delta
       LDX    #$36    ;2        ;Select Dragon #1 : Red
       JSR    LF7EA   ;6
       RTS            ;6

;Red Dragon Object Matrix
LF7A7: .byte $B6,$A4            ;Sword, Red Dragon
       .byte $A4,$8A            ;Red Dragon, Ball
       .byte $A4,$B9            ;Red Dragon, Chalise
       .byte $A4,$C2            ;Red Dragon, White Key
       .byte $00

;Move the Yellow Dragon.
LF7B0: LDA    #$C2    ;2
       STA    $D2     ;3        ;Set the Low Address of Object Store.
       LDA    #$F7    ;2
       STA    $D3     ;3        ;Set the High Address of Object Store.
       LDA    #$02    ;2
```

```
       STA    $D4     ;3        ;Set the Yellow Dragon's Delta.
       LDX    #$3F    ;2
       JSR    LF7EA   ;6        ;Select Dragon #2 : Yellow.
       RTS            ;6

;Yellow Dragon's Object Matrix

LF7C2: .byte $B6,$A9            ;Sword, Yellow Dragon
       .byte $BF,$A9            ;Yellow Key, Yellow Dragon
       .byte $A9,$8A            ;Yellow Dragon, Ball
       .byte $A9,$B9            ;Yellow Dragon, Chalise
       .byte $00


;Move the Green Dragon

LF7CB: LDA    #$DD    ;2
       STA    $D2     ;3        ;Set Low Address of Object Store.
       LDA    #$F7    ;2
       STA    $D3     ;3        ;Set High Address of Object Store.
       LDA    #$02    ;2
       STA    $D4     ;3        ;Set the Green Dragon's Delta.
       LDX    #$48    ;2        ;Select Dragon #3 : Green
       JSR    LF7EA   ;6
       RTS            ;6

;Green Dragon's Object Matrix

LF7DD: .byte $B6,$AE            ;Sword, Green Dragon
       .byte $AE,$8A            ;Green Dragon, Ball
       .byte $AE,$B9            ;Green Dragon Chalise
       .byte $AE,$BC            ;Green Dragon, Bridge
       .byte $AE,$B3            ;Green Dragon, Magnet
       .byte $AE,$C5            ;Green Dragon, Black Key
       .byte $00


;Move Object
LF7EA: STX    $A0     ;3        ;Save Object were dealing with.
       LDA    LFF44,X ;4        ;Get the Object's Dynamic Data.
       TAX            ;2
       LDA    NUSIZ0,X;4        ;Get the Object's State.
       CMP    #$00    ;2        ;Is it in State 00 (Normal #1)
       BNE    LF84E   ;2        ;Branch if not.

;Dragon Normal (State 1)
       LDA    SWCHB   ;4        ;Read console switches.
       AND    #$80    ;2        ;Check for P1 difficulty.
       BEQ    LF802   ;2        ;If Amateur Branch.

       LDA    #$00    ;2        ;Set Hard - Ignore Nothing
       JMP    LF804   ;3

LF802: LDA    #$B6    ;2        ;Set Easy - Ignore Sword.

LF804: STA    $D6     ;3        ;Store Difficulty
       STX    $D5     ;3        ;Store Dynamic Data Address.
       JSR    LF715   ;6

       LDA    $A0     ;3        ;Get Object
       JSR    LF6E9   ;6        ;And get the Player-Ball collision.
       BEQ    LF832   ;2        ;If None Then Branch.

       LDA    SWCHB   ;4        ;Get Console Switched.
       ROL            ;2        ;Move P0 difficulty to
       ROL            ;2        ;bit 01 position.
       ROL            ;2
       AND    #$01    ;2        ;Mask it out.
```

```
       ORA    $DD     ;3        ;Merget in the Level Number.
       TAY            ;2        ;Create Lookup.
       LDA    LF89F,Y ;4        ;Get New State.
       STA    NUSIZ0,X;4        ;Store as Dragon's State (Open Mouthed).
       LDA    $E3     ;3
       STA    VBLANK,X;4        ;Get Temp Ball X Coord and Store as Dragon's.
       LDA    $E4     ;3
       STA    WSYNC,X ;4        ;Get Temp Ball Y Coord and Store as Dragon's
       LDA    #$01    ;2
       STA    $E0     ;3        ;Set Noise Type to 01
       LDA    #$10    ;2
       STA    $DF     ;3        ;Set Noise Count to 10 i.e. make roar noise.

LF832: STX    $9A     ;3        ;Store Object's Dynamic Data Address.
       LDX    $A0     ;3        ;Get the Object Number.
       JSR    LF6FE   ;6        ;See if anoher object has hit the dragon.
       LDX    $9A     ;3        ;Get the Object Address.
       CMP    #$51    ;2        ;Has the Sword hit the Dragon?
       BNE    LF84B   ;2        ;If Not, Branch.

       LDA    #$01    ;2        ;Set the State to 01 (Dead)
       STA    NUSIZ0,X;4
       LDA    #$03    ;2        ;Set Sound Three.
       STA    $E0     ;3
       LDA    #$10    ;2        ;Set a Noise count of &10.
       STA    $DF     ;3

LF84B: JMP    LF89E   ;3        ;Jump to Finish.

LF84E: CMP    #$01    ;2        ;Is it in State 01 (Dead)
       BEQ    LF89E   ;2        ;Branch if So (Return)

       CMP    #$02    ;2        ;Is it int State 02 (Normal #2)
       BNE    LF871   ;2        ;Branch if Not.

;Normal Dragon State 2 (Eaten Ball)
       LDA    VSYNC,X ;4        ;Get the Dragon's Current Room.
       STA    $8A     ;3        ;Store as the Ball's Current Room
       STA    $E2     ;3        ;and Previous Room.
       LDA    VBLANK,X;4        ;Get the Dragon's X Coordinate.
       CLC            ;2
       ADC    #$03    ;2        ;Adjust
       STA    $8B     ;3        ;and store as the ball's X coordinate.
       STA    $E3     ;3        ;and previous X coordinate.
       LDA    WSYNC,X ;4        ;Get the Dragon's Y coordinate.
       SEC            ;2
       SBC    #$0A    ;2        ;Adjust
       STA    $8C     ;3        ;and store as the ball's Y coordinate.
       STA    $E4     ;3        ;and the previous Y coordinate.
       JMP    LF89E   ;3


;Dragon Roaring.
LF871: INC    NUSIZ0,X;6        ;Increment the Dragon's State.
       LDA    NUSIZ0,X;4        ;Get it's State.
       CMP    #$FC    ;2        ;Is it near the end?
       BCC    LF89E   ;2        ;If Not, Branch.

       LDA    $A0     ;3        ;Get the Dragon's Number.
       JSR    LF6E9   ;6        ;Check if the Ball is colliding.
       BEQ    LF89E   ;2        ;If not, Branch.

       LDA    #$02    ;2        ;Set the State to State 02 : Eaten
       STA    NUSIZ0,X;4
       LDA    #$02    ;2        ;Set noise two.
       STA    $E0     ;3
       LDA    #$10    ;2        ;Set the Count of Noise to &10.
       STA    $DF     ;3
```

```
       LDA    #$9B    ;2        ;Get the Maximum X Coordinate.
       CMP    VBLANK,X;4        ;Compare with the Dragon's X Coordinate.
       BEQ    LF896   ;2
       BCS    LF896   ;2

       STA    VBLANK,X;4        ;If too large then Use It.

LF896: LDA    #$17    ;2        ;Set Minimum Y Coordinate.
       CMP    WSYNC,X ;4        ;Compare with the Dragon's Y Coordinate.
       BCC    LF89E   ;2

       STA    WSYNC,X ;4        ;If Too Small, set as Dragon's Y coordinate.

LF89E: RTS            ;6

;Dragon Difficulty
LF89F: .byte $D0,$E8            ;Level 1 : Am, Pro
       .byte $F0,$F6            ;Level 2 : Am, Pro
       .byte $F0,$F6            ;Level 3 : Am, Pro


;Move Bat
LF8A5: INC    $CF     ;5        ;Put Bat in the Next State.
       LDA    $CF     ;3        ;Get the Bat State.
       CMP    #$08    ;2        ;Has it Reached the Maximum?
       BNE    LF8B1   ;2

       LDA    #$00    ;2        ;If So, Reset the Bat State.
       STA    $CF     ;3

LF8B1: LDA    $D1     ;3        ;Get the Bat Fed-Up Value.
       BEQ    LF8C3   ;2        ;If Bat Fed-Up then Branch.

       INC    $D1     ;5        ;Increment its value for next time.
       LDA    $CE     ;3        ;Get the Bat's Movement.
       LDX    #$CB    ;2        ;Position to Bat.
       LDY    #$03    ;2        ;Get the Bat's Deltas.
       JSR    LF5FF   ;6        ;Move the Bat.
       JMP    LF908   ;3        ;Update the Bat's Object.

;Bat Fed-Up
LF8C3: LDA    #$CB    ;2        ;Store the Bat's Dynamic Data Address
       STA    $D5     ;3
       LDA    #$03    ;2        ;Set the Bat's Delta.
       STA    $D4     ;3
       LDA    #$27    ;2        ;Set the Low Address of Object Store.
       STA    $D2     ;3
       LDA    #$F9    ;2        ;Set the High Address of Object Store.
       STA    $D3     ;3
       LDA    $D0     ;3        ;Get Object being Carried by Bat,
       STA    $D6     ;3        ;And Copy.
       JSR    LF715   ;6        ;Move the Bat.

       LDY    $E1     ;3        ;Get Object Liked Index.
       LDA    ($D2),Y ;5        ;Look up the Object Found in the Table.
       BEQ    LF908   ;2        ;If nothing found then Forget it.

       INY            ;2
       LDA    ($D2),Y ;5        ;Get the Object Wanted.
       TAX            ;2
       LDA    VSYNC,X ;4        ;Get the Object's Room.
       CMP    $CB     ;3        ;Is it the Same as the Bats?
       BNE    LF908   ;2        ;If not Forget it.

;See if Bat Can pick up Object.
       LDA    VBLANK,X;4        ;Get the Object's X Coordinate.
       SEC            ;2
       SBC    $CC     ;3        ;Find the differenct with the Bat's
```

```
       CLC             ;2        ;X coordinate.
       ADC    #$04     ;2        ;Adjust so Bat in middle of object.
       AND    #$F8     ;2        ;Is Bat within Seven Pixels?
       BNE    LF908    ;2        ;If not, no pickup possible.

       LDA    WSYNC,X  ;4        ;Get the Object's Y Coordinate.
       SEC             ;2
       SBC    $CD      ;3        ;Find the Difference with the Bat's
       CLC             ;2        ;      Y Coordinate.
       ADC    #$04     ;2        ;Adjust.
       AND    #$F8     ;2        ;Is the Bat within Seven Pixels?
       BNE    LF908    ;2        ;If not, No Pickup Possible.

;Get Object
       STX    $D0      ;3        ;Store Object as Being Carried.
       LDA    #$10     ;2        ;Reset the Bat Fed Up Time.
       STA    $D1      ;3

;Move Object Being Carried by Bat.
LF908: LDX    $D0      ;3        ;Get Object Being Carried by Bat.
       LDA    $CB      ;3        ;Get the Bat's Room.
       STA    VSYNC,X  ;4        ;Store this as the Object's Room.
       LDA    $CC      ;3        ;Get the Bat's X coordinate.
       CLC             ;2
       ADC    #$08     ;2        ;Adjust to the Right.
       STA    VBLANK,X ;4        ;Make it the Object's X coordinate.
       LDA    $CD      ;3        ;Get the Bat's Y Coordinate.
       STA    WSYNC,X  ;4        ;Store it as the Object's Y Coordinate.
       LDA    $D0      ;3        ;Get the Object Being Carried by the Bat.
       LDY    $9D      ;3        ;Get the Object Being Carried by the Ball.
       CMP    LFF44,Y  ;4        ;Are the the Same?
       BNE    LF926    ;2        ;If not Branch.

       LDA    #$A2     ;2        ;Set Nothing Being
       STA    $9D      ;3        ;Carried.

LF926: RTS             ;6


;Bat Object Matrix.
LF927: .byte $CB,$B9              ;Bat,Chalise
       .byte $CB,$B6              ;Bat,Sword
       .byte $CB,$BC              ;Bat,Bridge
       .byte $CB,$BF              ;Bat,Yellow Key
       .byte $CB,$C2              ;Bat,White Key
       .byte $CB,$C5              ;Bat,Black Key
       .byte $CB,$A4              ;Bat,Red Dragon
       .byte $CB,$A9              ;Bat,Yellow Dragon
       .byte $CB,$AE              ;Bat,Green Dragon
       .byte $CB,$B3              ;Bat,Magnet
       .byte $00


;Deal with Portcullis and Collisions.
LF93C: LDY    #$02     ;2        ;For Each Portcullis.
LF93E: LDX    LF9A7,Y  ;4        ;Get the Portcullises offset number.
       JSR    LF6FE    ;6        ;See if an Object Collided with it.
       STA    $97      ;3        ;Store that Object.
       CMP    LF9AA,Y  ;4        ;Is it the Associated Key?
       BNE    LF94F    ;2        ;If not then Branch.

       TYA             ;2        ;Get the Portcullis Number
       TAX             ;2
       INC    $C8,X    ;6        ;Change it's state to open it.

LF94F: TYA             ;2        ;Get the Porcullis number.
       TAX             ;2
       LDA    $C8,X    ;4        ;Get the State.
```

```
       CMP    #$1C    ;2        ;Is it Closed?
       BEQ    LF988   ;2        ;Yes - then Branch.

       LDA    LF9A7,Y ;4        ;Get Portcullis number.
       JSR    LF6E9   ;6        ;Get the Player-Ball Collision.
       BEQ    LF968   ;2        ;If Nont Then Branch.

       LDA    #$01    ;2        ;Set the Portcullis to Closed.
       STA    $C8,X   ;4
       LDX    #$8A    ;2        ;Set to the Castle.
       JMP    LF97F   ;3        ;Put the Ball in the Castle.

LF968: LDA    $97     ;3        ;Get the Object that hit the Portcullis.
       CMP    #$A2    ;2        ;Is it nothing?
       BEQ    LF97C   ;2        ;If so, Branch.

       LDX    $97     ;3        ;Get Object.
       STY    $9A     ;3        ;Save Y
       JSR    LF2E4   ;6        ;And find it's Dynamic Address.
       LDY    $9A     ;3        ;Retrieve Y
       LDX    $93     ;3        ;Get Object's Address.
       JMP    LF97F   ;3        ;Put Object In the Castle.

LF97C: JMP    LF988   ;3

LF97F: LDA    LF9AD,Y ;4     ;Look up Castle endry room for this port.
       STA    VSYNC,X ;4     ;Make it the object's Room.
       LDA    #$10    ;2     ;Give the Object a new Y coordinate.
       STA    WSYNC,X ;4

LF988: TYA             ;2     ;Get the Portcullis number.
       TAX             ;2
       LDA    $C8,X   ;4     ;Get its State.
       CMP    #$01    ;2     ;Is it Open?
       BEQ    LF9A0   ;2     ;Yes - Then Branch.

       CMP    #$1C    ;2     ;Is it Closed?
       BEQ    LF9A0   ;2     ;Yes - Then Branch.

       INC    $C8,X   ;6     ;Increment it's State.
       LDA    $C8,X   ;4     ;Get the State.
       CMP    #$38    ;2     ;Has it reached the maximum state.
       BNE    LF9A0   ;2     ;If not, Branch.

       LDA    #$01    ;2     ;Set to Closed
       STA    $C8,X   ;4     ;   State.

LF9A0: DEY             ;2     ;Goto the next portcullis.
       BMI    LF9A6   ;2     ;Branch if Finished.
       JMP    LF93E   ;3     ;Do next Protcullis.
LF9A6: RTS             ;6

;Portcullis #1, #2, #3
LF9A7: .byte $09,$12,$1B

;Keys #1, #2, #3  (Yellow, White, Black)
LF9AA: .byte $63,$6C,$75

;Castle Entry Rooms (Yellow, White, Black)
LF9AD: .byte $12,$1A,$1B

;Castle Rooms (Yellow, White, Black)
LF9B0: .byte $11,$0F,$10


;Deal With Magnet.
LF9B3: LDA    $B5     ;3        ;Get Magnet's Y Coordinate.
       SEC             ;2
```

```
       SBC    #$08    ;2        ;Adjust to it's .
       STA    $B5     ;3
       LDA    #$00    ;2        ;Con Difficulty!
       STA    $D6     ;3
       LDA    #$DA    ;2        ;Set Low Address of Object Store.
       STA    $D2     ;3
       LDA    #$F9    ;2        ;Set High Address of Object Store.
       STA    $D3     ;3
       JSR    LF728   ;6        ;Get Liked Object and Set Movement.
       LDA    $9B     ;3        ;Get Movement.
       BEQ    LF9D2   ;2        ;If None, then Forget It.

       LDY    #$01    ;2        ;Set Delta to One.
       JSR    LF5FF   ;6        ;Move Object.

LF9D2: LDA    $B5     ;3        ;Reset the Magnet's
       CLC            ;2        ;Y Coordinate.
       ADC    #$08    ;2
       STA    $B5     ;3
       RTS            ;6

;Magnet Object Matrix.

LF9DA: .byte $BF,$B3            ;Yellow Key, Magnet
       .byte $C2,$B3            ;White Key, Magnet
       .byte $C5,$B3            ;Black Key, Magnet
       .byte $B6,$B3            ;Sword, Magnet
       .byte $BC,$B3            ;Bridge, Magnet
       .byte $B9,$B3            ;Chalise, Magnet
       .byte $00



;Deal with Invisible Surround Moving.
LF9E7: LDA    $8A     ;3        ;Get the Current Room.
       JSR    LF271   ;6        ;Convert it to an Address.
       LDY    #$02    ;2
       LDA    ($93),Y ;5        ;Get the Room's Color.
       CMP    #$08    ;2        ;Is it Invisible?
       BEQ    LF9FB   ;2        ;If So Branch.

       LDA    #$00    ;2        ;If not, signal the
       STA    $DB     ;3        ;Invisible surround not
       JMP    LFA22   ;3        ;Wanted.

LF9FB: LDA    $8A     ;3        ;Get the Current Room.
       STA    $D9     ;3        ;And store as the Invisible Surrounds.
       LDA    $8B     ;3        ;Get the Ball's X Coordinate.
       SEC            ;2
       SBC    #$0E    ;2        ;Adjust for Surround,
       STA    $DA     ;3        ;and store as surround's X coordinate.
       LDA    $8C     ;3        ;Get the Ball's Y Coordinate.
       CLC            ;2
       ADC    #$0E    ;2        ;Adjust for Surround.
       STA    $DB     ;3        ;and store as surround's Y coordinate.
       LDA    $DA     ;3        ;Get the Surround's X cordinate.
       CMP    #$F0    ;2        ;Is it close to the right edge?
       BCC    LFA1A   ;2        ;Branch if not.

       LDA    #$01    ;2        ;Flick surround to the
       STA    $DA     ;3        ;otherside of the screen.
       JMP    LFA22   ;3

LFA1A: CMP    #$82    ;2        ;???
       BCC    LFA22   ;2        ;???

       LDA    #$81    ;2        ;???
       STA    $DA     ;3        ;???
```

```
LFA22: RTS            ;6


;Make A Noise.
LFA23: LDA    $DF     ;3        ;Check Not Count.
       BNE    LFA2C   ;2        ;Branch if Noise to be made.

       STA    AUDV0   ;3        ;Turn off the Volume.
       STA    AUDV1   ;3
       RTS            ;6

LFA2C: DEC    $DF     ;5        ;Goto the Next Note.
       LDA    $E0     ;3        ;Get the Noise Type.
       BEQ    LFA47   ;2        ;Game Over

       CMP    #$01    ;2        ;Roar
       BEQ    LFA55   ;2

       CMP    #$02    ;2        ;Man Eaten.
       BEQ    LFA6C   ;2

       CMP    #$03    ;2        ;Dying Dragon.
       BEQ    LFA7F   ;2

       CMP    #$04    ;2        ;Dropping Object.
       BEQ    LFA8C   ;2

       CMP    #$05    ;2        ;Picking up Object.
       BEQ    LFA9B   ;2

       RTS            ;6


;Noise 0 : Game Over
LFA47: LDA    $DF     ;3
       STA    COLUPF  ;3        ;Color-Luminance Playfield.
       STA    AUDC0   ;3        ;Audio-Control 00
       LSR            ;2
       STA    AUDV0   ;3        ;Audio-Volume 00
       LSR            ;2
       LSR            ;2
       STA    AUDF0   ;3        ;Audio-Frequency 00
       RTS            ;6

;Noise 1 : Roar
LFA55: LDA    $DF     ;3        ;Get noise count.
       LSR            ;2
       LDA    #$03    ;2        ;If it was even then
       BCS    LFA5E   ;2        ;Branch.

       LDA    #$08    ;2        ;Get a differnt audio control value.

LFA5E: STA    AUDC0   ;3        ;Set Audio Control 00.
       LDA    $DF     ;3        ;Set the Volume to the Noise Count.
       STA    AUDV0   ;3
       LSR            ;2        ;Divide by Four.
       LSR            ;2
       CLC            ;2
       ADC    #$1C    ;2        ;Set the Frequency.
       STA    AUDF0   ;3
       RTS            ;6


;Noise 2 : Man Eaten
LFA6C: LDA    #$06    ;2
       STA    AUDC0   ;3        ;Audio-Control 00
       LDA    $DF     ;3
       EOR    #$0F    ;2
```

```
       STA    AUDF0   ;3        ;Audio-Frequency 00
       LDA    $DF     ;3
       LSR            ;2
       CLC            ;2
       ADC    #$08    ;2
       STA    AUDV0   ;3        ;Audio-Volume 00
       RTS            ;6


;Noise 3 : Dyning Dragon
LFA7F: LDA    #$04    ;2        ;Set the Audio Control
       STA    AUDC0   ;3
       LDA    $DF     ;3        ;Put the Note Count In
       STA    AUDV0   ;3        ;      the Volume.
       EOR    #$1F    ;2
       STA    AUDF0   ;3        ;Flip the Count as store
       RTS            ;6        ;      as the frequency.


;Noise 4 : Dropping Object.
LFA8C: LDA    $DF     ;3        ;Get Not Count
       EOR    #$03    ;2        ;Reverse it as noise does up.
LFA90: STA    AUDF0   ;3        ;Store in Frequency for Channel 00.
       LDA    #$05    ;2
       STA    AUDV0   ;3        ;Set Volume on Channel 00.
       LDA    #$06    ;2
       STA    AUDC0   ;3        ;Set a Noise on Channel 00.
       RTS            ;6


;Noise 5 : Picking up an Object.
LFA9B: LDA    $DF     ;3        ;Get Not Count.
       JMP    LFA90   ;3        ;and Make Same noise as Drop.


;Left of Easter Egg Room
LFAA0:.byte $F0,$FF,$FF
      .byte $C0,$0C,$30
      .byte $C0,$CC,$F3
      .byte $00,$0C,$03
      .byte $C0,$FC,$1F
      .byte $00,$00,$18
      .byte $F0,$FF,$1F

;Below Yellow Castle
      ;Shares $F0, $FF, $07 From Left of Easter Egg Room ----^
      .byte $00,$00,$18
      .byte $C0,$F1,$1F
      .byte $C0,$00,$00
      .byte $C0,$F1,$1F
      .byte $00,$00,$18
      .byte $F0,$FF,$FF


;Side Corridor
      .byte $F0,$FF,$1F
      .byte $30,$00,$18
      .byte $30,$C1,$79
      .byte $30,$FF,$01
      .byte $30,$C1,$79
      .byte $00,$00,$18
      .byte $F0,$FF,$FF


;Number Room Definition (Level Select Room)
      .byte $F0,$FF,$FF
      .byte $30,$03,$00
      .byte $30,$63,$18
```

```
      .byte $30,$63,$1F
      .byte $F0,$E0,$18
      .byte $30,$03,$00
      .byte $F0,$FF,$1F


;Object #1 States (Portcullis)
      .byte $04,$24,$FB         ;State 04 at FB24 -Open
      .byte $08,$22,$FB         ;State 08 at FB22
      .byte $0C,$20,$FB         ;State 0C at FB20
      .byte $10,$1E,$FB         ;State 10 at FB1E
      .byte $14,$1C,$FB         ;State 14 at FB1C
      .byte $18,$1A,$FB         ;State 18 at FB1A
      .byte $1C,$18,$FB         ;State 1C at FB18
      .byte $20,$1A,$FB         ;State 20 at FB1A
      .byte $24,$1C,$FB         ;State 24 at FB1C
      .byte $28,$1E,$FB         ;State 28 at FB1E
      .byte $2C,$20,$FB         ;State 2C at FB20
      .byte $30,$22,$FB         ;State 30 at FB22
      .byte $FF,$24,$FB         ;State FF at FB24 -Open


;Object #1 States 940FF Castle Gate

        .byte $FE
        .byte $AA
        .byte $FE
        .byte $AA
        .byte $FE
        .byte $AA
        .byte $EE
        .byte $82
        .byte $EE
        .byte $AA
        .byte $FE
        .byte $AA
        .byte $FE
        .byte $AA
        .byte $FE
        .byte $AA
        .byte $00

;Two Exit Room
      .byte $F0,$FF,$1F
      .byte $30,$00,$18
      .byte $30,$F9,$7F
      .byte $30,$C0,$00
      .byte $30,$FF,$FF
      .byte $30,$00,$00
      .byte $F0,$FF,$1F


;Top of Blue Maze
      .byte $F0,$FF,$1F
      .byte $00,$00,$00
      .byte $F0,$FF,$FF
      .byte $30,$00,$00
      .byte $30,$FF,$FF
      .byte $00,$C0,$00
      .byte $F0,$CF,$1F

;Blue Maze #1
      .byte $F0,$FF,$FF
      .byte $00,$00,$80
      .byte $F0,$FF,$9F
      .byte $30,$00,$80
      .byte $30,$FE,$FE
      .byte $00,$C0,$00
```

```
      .byte $F0,$FF,$FF

;Bottom of Blue Maze
      .byte $F0,$FF,$07
      .byte $00,$00,$06
      .byte $30,$FF,$1F
      .byte $30,$00,$00
      .byte $F0,$FF,$F9
      .byte $00,$30,$18
      .byte $F0,$FF,$1F

;Center of Blue Maze
      .byte $F0,$CF,$1F
      .byte $00,$C0,$18
      .byte $F0,$FF,$19
      .byte $00,$00,$18
      .byte $F0,$9F,$1F
      .byte $80,$80,$00
      .byte $F0,$FF,$07

;Blue Maze Entry
      .byte $F0,$FF,$FF
      .byte $00,$00,$00
      .byte $70,$7F,$19
      .byte $00,$60,$18
      .byte $F0,$E7,$1F
      .byte $00,$00,$18
      .byte $F0,$FF,$1F

;Invisible Maze Middle
      .byte $F0,$FF,$CC
      .byte $00,$00,$CC
      .byte $F0,$FF,$CF
      .byte $00,$03,$00
      .byte $F0,$F3,$FC
      .byte $00,$33,$0C
      .byte $F0,$33,$CC

;Invisible Maze Bottom
      ; shared from bottom of maze middle
      .byte $F0,$30,$CC
      .byte $30,$3F,$CF
      .byte $30,$00,$C0
      .byte $F0,$3F,$CF
      .byte $00,$03,$C0
      .byte $F0,$FF,$FF

;Invisible Maze Entry
      .byte $F0,$FF,$0F
      .byte $00,$00,$00
      .byte $F0,$FF,$FF
      .byte $00,$30,$C0
      .byte $F0,$33,$CC
      .byte $00,$33,$CC
      .byte $F0,$FF,$CC

;Castle Definition
      .byte $F0,$FA,$15
      .byte $10,$07,$1F
      .byte $10,$07,$FF
      .byte $10,$00,$FF
      .byte $10,$00,$3F
      .byte $10,$00,$00
      .byte $F0,$FF,$1F

;Object Data
;Offset 0 : Room number of object.
;Offset 1 : X Coordinate of object.
```

```
;Offset 2 : Y Coordinate of object.

;Object #1 : Portcullis
      .byte $11,$4D,$31         ;Room 11, (4D, 31)
;Object #2 : Portcullis
      .byte $0F,$4D,$31         ;Room 0F, (4D, 31)
;Object #3 : Portcullis
      .byte $10,$4D,$31         ;Room 10, (4D, 31

;Object #0 : State
      .byte $00

;Object #1 : State List
      .byte $FF,$05,$FC         ;State FF as FC05

;Object #1 : Graphic
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $FF            ;XXXXXXXX
      .byte $00

;Red Maze Top Left
      .byte $F0,$FF,$FF
      .byte $00,$00,$00
      .byte $F0,$FF,$FF
      .byte $00,$00,$C0
      .byte $F0,$FF,$CC
      .byte $30,$03,$CC
      .byte $30,$F3,$CF

;Bottom Left of Red Maze
 ;shares bottom line of Red maze top left
      .byte $30,$30,$C0
      .byte $F0,$FC,$FF
      .byte $00,$0C,$03
      .byte $70,$7C,$1F
      .byte $00,$60,$18
      .byte $F0,$FF,$1F
```

```
;Top Right of Red Maze
      .byte $F0,$FF,$FF
      .byte $00,$00,$C0
      .byte $F0,$FF,$CF
      .byte $00,$00,$C0
      .byte $30,$FF,$FF
      .byte $30,$00,$00
      .byte $30,$FF,$1F

;White Castle Entry (Red Maze Bottom Right)
      ;shared from bottom line of Red Maze Top Right
      .byte $30,$00,$18
      .byte $F0,$FF,$F9
      .byte $00,$00,$18
      .byte $F0,$FF,$1F
      .byte $00,$00,$00
      .byte $F0,$FF,$1F

;Top Entry Room
      .byte $F0,$FF,$1F
      .byte $10,$00,$18
      .byte $90,$FF,$19
      .byte $10,$18,$18
      .byte $F0,$9F,$7F
      .byte $10,$00,$00
      .byte $F0,$FF,$FF

;Black Maze Top Left
      .byte $F0,$C3,$FF
      .byte $00,$03,$00
      .byte $F0,$FF,$7F
      .byte $00,$00,$00
      .byte $30,$FF,$FF
      .byte $00,$C0,$00
      .byte $F0,$CF,$FF

;Black Maze Bottom Left
      ;Shares $F0,$F0,$FF from Black Maze Top Left-----^ (Mirrored instead of Reversed)
      .byte $10,$00,$80
      .byte $10,$FF,$FF
      .byte $00,$C0,$00
      .byte $F0,$C3,$FF
      .byte $10,$03,$80
      .byte $F0,$C3,$FF

;Black Maze Top Right
      .byte $F0,$FF,$FF
      .byte $00,$00,$C0
      .byte $F0,$FF,$CF
      .byte $00,$00,$0C
      .byte $F0,$CF,$FF
      .byte $00,$0F,$C0
      .byte $30,$CF,$CC

;Black Maze Entry
      ;Shares $30,$CF,$CC From Black Maze Top Right ----^ (Reversed instead of mirrored. )
      .byte $00,$00,$98
      .byte $F0,$FF,$9F
      .byte $00,$00,$00
      .byte $F0,$FF,$FF
      .byte $00,$00,$00
      .byte $F0,$FF,$0F

;Objtect #0A : State
      .byte $00

;Object #0A : List of States
      .byte $FF,$DB,$FC         ;State FF at &FCDB
```

```
;Object #0A : State FF : Graphic
      .byte $c3
      .byte $C3
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $81
      .byte $C3
      .byte $00

;Object #5 State #1 Graphic :'1'
      .byte $08
      .byte $18
      .byte $08
      .byte $08
      .byte $08
      .byte $08
      .byte $1C
      .byte $00

;Object #0B : State
      .byte $00

;Object #0B : List of States
      .byte $FF,$00,$FD

;Object #0B : State FF : Graphic - Key
      .byte $E0
      .byte $BF
      .byte $E5
      .byte $00

;Object #5 State #2 Grphic : '2'
      .byte $3E
      .byte $22
      .byte $02
      .byte $3E
      .byte $20
      .byte $22
      .byte $3E
      .byte $00

;Object #5 State #3 Graphic :'3'
      .byte $3E
      .byte $22
      .byte $02
      .byte $0E
      .byte $02
      .byte $22
      .byte $3E
```

```
      .byte $00


;Object #0E : List of States
      .byte $03,$1A,$FD         ;State 03 at &FD1A
      .byte $FF,$22,$FD         ;State FF as &FD22

;Object #0E : State 03 : Bat - Top
      .byte $81
      .byte $81
      .byte $C3
      .byte $C3
      .byte $FF
      .byte $5A
      .byte $66
      .byte $00

;Object #0E : State FF : Graphic
      .byte $01
      .byte $80
      .byte $01
      .byte $80
      .byte $3C
      .byte $5A
      .byte $66
      .byte $C3
      .byte $81
      .byte $81
      .byte $81
      .byte $00

;Object #6 : States
      .byte $00,$3A,$FD         ;State 00 at &FD3A
      .byte $01,$66,$FD         ;State 01 at &FD66
      .byte $02,$3A,$FD         ;State 02 at &FD3A
      .byte $FF,$4F,$FD         ;State FF at &FD4F

;Object #6 : State #00 : Dragon - Normal State
      .byte $18
      .byte $3C
      .byte $EE
      .byte $FE
      .byte $3D
      .byte $1D
      .byte $1B
      .byte $37
      .byte $7F
      .byte $FC
      .byte $FE
      .byte $FF
      .byte $FF
      .byte $7F
      .byte $3F
      .byte $27
      .byte $4B
      .byte $93
      .byte $26
      .byte $5C
      .byte $00

;Object 6 : State FF : Dragon - Attacking
      .byte $80
      .byte $40
      .byte $2C
      .byte $3E
      .byte $17
      .byte $3E
      .byte $2D
```

```
      .byte $5D
      .byte $9B
      .byte $37
      .byte $7F
      .byte $FC
      .byte $FE
      .byte $FF
      .byte $FF
      .byte $7F
      .byte $3F
      .byte $27
      .byte $4B
      .byte $93
      .byte $26
      .byte $5C
      .byte $00

;Object 6 : State 02 : Dragon - Dead State
      .byte $18
      .byte $3C
      .byte $FE
      .byte $FC
      .byte $3E
      .byte $1D
      .byte $1B
      .byte $3F
      .byte $7E
      .byte $D7
      .byte $EE
      .byte $D4
      .byte $7F
      .byte $3F
      .byte $2B
      .byte $53
      .byte $AE
      .byte $00

;Object #9 : State
      .byte $00

;Object #9 : List of States
      .byte $FF,$7C,$FD         ;State FF at &FD7C

;Object #9 : State FF : Graphics
      .byte $20
      .byte $42
      .byte $FF
      .byte $42
      .byte $20
      .byte $00

;Object #0F : State
      .byte $00


;Object #0F : List of States
      .byte $FF,$86,$FD         ;State FF at FD86

;Object #0F : State FF : Graphic
      .byte $80             ;X
      .byte $00

;Object #4 : State FF : Graphic
      .byte $18         ;---XX---
      .byte $3C         ;--XXXX--
      .byte $66         ;-XX--XX-
      .byte $C3         ;XX----XX
      .byte $FF         ;XXXXXXXX
```

```
.byte $FF         ;XXXXXXXX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $01         ;-------X
.byte $FC         ;XXXXXX--
.byte $FE         ;XXXXXXX-
.byte $C7         ;XX---XXX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $C7         ;XX---XXX
.byte $FE         ;XXXXXXX-
.byte $FD         ;XXXXXX-X
.byte $03         ;------XX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $E7         ;XXX--XXX
.byte $7E         ;-XXXXXX-
.byte $3C         ;--XXXX--
.byte $18         ;---XX---
.byte $7E         ;-XXXXXX-
.byte $FF         ;XXXXXXXX
.byte $C0         ;XX------
.byte $FC         ;XXXXXX--
.byte $FC         ;XXXXXX--
.byte $C0         ;XX------
.byte $FF         ;XXXXXXXX
.byte $7E         ;-XXXXXX-
.byte $01         ;-------X
.byte $C3         ;XX----XX
.byte $E3         ;XXX---XX
.byte $F3         ;XXXX--XX
.byte $FB         ;XXXXX-XX
.byte $DF         ;XX-XXXXX
.byte $CF         ;XX--XXXX
.byte $C7         ;XX---XXX
.byte $80         ;X-------
.byte $7E         ;-XXXXXX-
.byte $FF         ;XXXXXXXX
.byte $18         ;---XX---
.byte $18         ;---XX---
.byte $18         ;---XX---
.byte $18         ;---XX---
.byte $18         ;---XX---
.byte $DB         ;XX-XX-XX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $C3         ;XX----XX
.byte $FF         ;XXXXXXXX
.byte $7E         ;-XXXXXX-
.byte $80         ;X-------
.byte $FE         ;XXXXXXX-
.byte $FF         ;XXXXXXXX
.byte $C3         ;XX----XX
.byte $FE         ;XXXXXXX-
.byte $FC         ;XXXXXX--
.byte $CE         ;XX--XXX-
.byte $C7         ;XX---XXX
.byte $80         ;X-------
.byte $7E         ;-XXXXXX-
.byte $FF         ;XXXXXXXX
.byte $C0         ;XX------
.byte $FC         ;XXXXXX--
.byte $FC         ;XXXXXX--
.byte $C0         ;XX------
```

```
      .byte $FF         ;XXXXXXXX
      .byte $7E         ;-XXXXXX-
      .byte $18         ;---XX---
      .byte $18         ;---XX---
      .byte $7E         ;-XXXXXX-
      .byte $7E         ;-XXXXXX-
      .byte $18         ;---XX---
      .byte $18         ;---XX---
      .byte $01         ;-------X
      .byte $F0         ;XXXX---- ; PE (Packrat Edition 2005)
      .byte $90         ;X--X----
      .byte $F0         ;XXXX----
      .byte $8F         ;X---XXXX
      .byte $88         ;X---X---
      .byte $8E         ;X---XXX-
      .byte $08         ;----X---
      .byte $0F         ;----XXXX
      .byte $80         ;X-------
      .byte $77         ;-XXX-XXX
      .byte $54         ;-X-X-X--
      .byte $57         ;-X-X-XXX
      .byte $51         ;-X-X---X
      .byte $77         ;-XXX-XXX

      ; 'CGE 05'
      ;.byte $E0        ;XXX-----
      ;.byte $9C        ;X--XXX--
      ;.byte $90        ;X--X----
      ;.byte $F4        ;XXXX-X--
      ;.byte $1F        ;---XXXXX
      ;.byte $04        ;-----X--
      ;.byte $06        ;-----XX-
      ;.byte $04        ;-----X--
      ;.byte $07        ;-----XXX
      ;.byte $80        ;X-------
      ;.byte $77        ;-XXX-XXX
      ;.byte $54        ;-X-X-X--
      ;.byte $57        ;-X-X-XXX
      ;.byte $51        ;-X-X---X
      ;.byte $77        ;-XXX-XXX

      ;Original '+' from Adventure Plus Title
      ;.byte $C3        ;XX----XX
      ;.byte $99        ;X--XX--X
      ;.byte $18        ;---XX---
      ;.byte $18        ;---XX---
      ;.byte $FF        ;XXXXXXXX
      ;.byte $FF        ;XXXXXXXX
      ;.byte $18        ;---XX---
      ;.byte $18        ;---XX---
      ;.byte $99        ;X--XX--X
      ;.byte $C3        ;XX----XX


;Object $4 : Adventure Plus Title
      .byte $00,$49,$69         ;Room 1E, (50, 69)

;Object #4 : Current State
      .byte $00

;Object #4 : States
      .byte $FF,$88,$FD         ;State FF at &FD88

;Object #10 : State
      .byte $00

;Object #10 : List of States
       .byte $FF,$F3,$FD         ;State FF at &FDF3
```

```
;Object #10 : State FF : Graphic  - Chalice
      .byte $FB
      .byte $FB
      .byte $FB
      .byte $FB
      .byte $76
      .byte $3C
      .byte $18
      .byte $18
      .byte $7E
      .byte $00

;Object #12 : State
      .byte $00

;Object #12 : List of States
      .byte $FF,$01,$FE

;Object #12
      .byte $00

;Object #5 Number.
      .byte $00,$50,$40          ;#5 Number: Room 00, (50,40)

;Object #5 States.
      .byte $01,$F4,$FC         ;State 1 as FCF4
      .byte $03,$04,$FD         ;State 3 as FD04
      .byte $FF,$0C,$FD         ;State FF as FD0C

;Object #11 : State
      .byte $00

;Object #11 : List of States
      .byte $FF,$12,$FE         ;State FF at FE12

;Object #11 : State FF : Graphic  - Magnet
      .byte $3C
      .byte $7E
      .byte $FF
      .byte $E7
      .byte $C3
      .byte $C3
      .byte $C3
      .byte $C3
      .byte $00

;Room Data

        ;Offset 0 : Low byte foom graphics data.
        ;Offset 1 : High byte room graphics data
        ;Offset 2 : Color
        ;Offset 3 : B&W Color
        ;Offset 4 : Bits 5-0 : Playfield Control
        ;        Bit 6 : True if right thin wall wanted.
        ;        Bit 7 : True if left thin wall wanted.
        ;Offset 5 : Room Above
        ;Offset 6 : Room Left
        ;Offset 7 : Room Down
        ;Offset 8 : Room Right

;Room Data (Original from Adventure Plus 2003)
        .byte $DC,$FA,$03,$0A,$21,$00,$00,$00,$00 ;00; Number Room.                              Purple
        .byte $B2,$FA,$60,$0A,$21,$08,$02,$80,$03 ;01; (Top Access) Reflected/8 Clock Ball
        .byte $B2,$FA,$70,$0A,$21,$10,$03,$83,$01 ;02; (Top Access) below yellow castle         Green
        .byte $A0,$FA,$70,$0A,$21,$06,$01,$86,$02 ;03; Left of Name ;offset 4 was $61
        .byte $3E,$FB,$80,$0A,$21,$11,$05,$07,$06 ;04; Top of Blue Maze                          Blue
        .byte $53,$FB,$80,$0A,$21,$00,$06,$08,$04 ;05; Blue Maze #1                              Blue
```

```
        .byte $68,$FB,$80,$0A,$21,$07,$04,$0E,$05 ;06; Bottom of Blue Maze                     Blue
        .byte $7D,$FB,$80,$0A,$21,$04,$08,$06,$08 ;07; Center of Blue Maze                     Blue
        .byte $92,$FB,$80,$0A,$21,$05,$07,$01,$07 ;08; Blue Maze Entry                         Blue
        .byte $A7,$FB,$08,$08,$25,$0A,$0A,$0B,$0A ;09; Maze Middle                             Invisible
        .byte $CE,$FB,$08,$08,$25,$03,$09,$09,$09 ;0A; Maze Entry                              Invisible
        .byte $B9,$FB,$08,$08,$25,$09,$0C,$1C,$0D ;0B; Maze Side                               Invisible
        .byte $C7,$FA,$98,$0A,$21,$1C,$0D,$1D,$0B ;0C; (Side Corridor) ;
        .byte $C7,$FA,$60,$0A,$21,$0F,$0B,$0E,$0C ;0D; (Side Corridor)
        .byte $74,$FC,$80,$0A,$21,$06,$10,$0F,$10 ;0E; (Top Entry Room)
        .byte $E3,$FB,$0C,$0C,$21,$0E,$0F,$0D,$0F ;0F; White Castle
        .byte $E3,$FB,$00,$02,$21,$01,$1C,$02,$1C ;10; Black Castle
        .byte $E3,$FB,$03,$0A,$21,$06,$03,$04,$01 ;11; Yellow Castle
        .byte $DC,$FA,$03,$0A,$21,$12,$12,$12,$12 ;12; Yellow Castle Entry
        .byte $89,$FC,$08,$08,$25,$15,$14,$15,$16 ;13; Black Maze #1
        .byte $B0,$FC,$08,$08,$24,$16,$15,$16,$13 ;14; Black Maze #2
        .byte $9B,$FC,$08,$08,$24,$13,$16,$13,$14 ;15; Black Maze #3
        .byte $C2,$FC,$08,$08,$25,$14,$13,$1B,$15 ;16; Black Maze Entry
        .byte $26,$FC,$36,$0A,$21,$19,$18,$19,$18 ;17; Red Maze #1
        .byte $4D,$FC,$36,$0A,$21,$1A,$17,$1A,$17 ;18; Top of Red Maze
        .byte $38,$FC,$36,$0A,$21,$17,$1A,$1D,$1A ;19; Bottom of Red Maze
        .byte $5F,$FC,$36,$0A,$21,$18,$19,$18,$19 ;1A; White Castle Entry
        .byte $29,$FB,$36,$0A,$21,$89,$89,$89,$89 ;1B; Black Castle Entry
        .byte $DC,$FA,$66,$0A,$21,$1D,$07,$8C,$08 ;1C; Bottom Entry - Other Purple Room         Pur
        .byte $74,$FC,$36,$0A,$21,$8F,$01,$10,$03 ;1D; (Top Entry Room)                        Red
        .byte $B2,$FA,$66,$0A,$21,$11,$01,$06,$03 ;1E; Name Room                               Purple


      ; Room Data (For Testing - 2005)
      ;.byte $DC,$FA,$22,$0A,$21,$00,$00,$00,$00   ;00; Number Room.                           Purple
      ;.byte $B2,$FA,$22,$0A,$A1,$08,$02,$80,$03   ;01; (Top Acess) Reflected/8 Clock Ball
      ;.byte $B2,$FA,$20,$0A,$21,$11,$03,$83,$01   ;02; (Top Access) below yellow castle       Green
      ;.byte $A0,$FA,$20,$0A,$61,$06,$01,$86,$02   ;03; Left of Name
      ;.byte $3E,$FB,$80,$0A,$21,$10,$05,$07,$06   ;04; Top of Blue Maze                       Blue
      ;.byte $53,$FB,$80,$0A,$21,$1D,$06,$08,$04   ;05; Blue Maze #1                           Blue
      ;.byte $68,$FB,$80,$0A,$21,$07,$04,$03,$05   ;06; Bottom of Blue Maze                    Blue
      ;.byte $7D,$FB,$80,$0A,$21,$04,$08,$06,$08   ;07; Center of Blue Maze                    Blue
      ;.byte $92,$FB,$80,$0A,$21,$05,$07,$01,$07   ;08; Blue Maze Entry                        Blue
      ;.byte $A7,$FB,$08,$08,$25,$0A,$0A,$0B,$0A   ;09; Maze Middle
Invisible/Playfield Priority
      ;.byte $CE,$FB,$08,$08,$25,$03,$09,$09,$09   ;0A; Maze Entry
Invisible/Playfield Priority
      ;.byte $B9,$FB,$08,$08,$25,$09,$0C,$1C,$0D   ;0B; Maze Side
Invisible/Playfield Priority
      ;.byte $C7,$FA,$22,$0A,$21,$1C,$0D,$1D,$0B   ;0C; (Side Corridor)
      ;.byte $C7,$FA,$24,$0A,$21,$0F,$0B,$0E,$0C   ;0D; (Side Corridor)
      ;.byte $74,$FC,$80,$0A,$21,$0D,$10,$0F,$10   ;0E; (Top Entry Room)
      ;.byte $E3,$FB,$0E,$0C,$21,$0E,$0F,$0D,$0F   ;0F; White Castle                          White
      ;.byte $E3,$FB,$00,$02,$21,$01,$1C,$04,$1C   ;10; Black Castle                          Black
      ;.byte $E3,$FB,$18,$0A,$21,$06,$03,$02,$01   ;11; Yellow Castle                         Yellow
      ;.byte $DC,$FA,$12,$0A,$21,$12,$12,$12,$12   ;12; Yellow Castle Entry                   Yellow
      ;.byte $89,$FC,$08,$08,$25,$15,$14,$15,$16   ;13; Black Maze #1
      ;.byte $B0,$FC,$08,$08,$24,$16,$15,$16,$13   ;14; Black Maze #2
      ;.byte $9B,$FC,$08,$08,$24,$13,$16,$13,$14   ;15; Black Maze #3
      ;.byte $C2,$FC,$08,$08,$25,$14,$13,$1B,$15   ;16; Black Maze Entry
      ;.byte $26,$FC,$36,$0A,$21,$19,$18,$19,$18   ;17; Red Maze #1                           Red
      ;.byte $4D,$FC,$36,$0A,$21,$1A,$17,$1A,$17   ;18; Top of Red Maze                       Red
      ;.byte $38,$FC,$36,$0A,$21,$17,$1A,$17,$1A   ;19; Bottom of Red Maze                    Red
      ;.byte $5F,$FC,$36,$0A,$21,$18,$19,$18,$19   ;1A; White Castle Entry                    Red
      ;.byte $29,$FB,$36,$0A,$21,$89,$89,$89,$89   ;1B; Black Castle Entry                    Red
      ;.byte $DC,$FA,$66,$0A,$21,$1D,$07,$8C,$08   ;1C; Bottom Entry - Other Purple Room      Purple
      ;.byte $74,$FC,$36,$0A,$21,$8F,$01,$10,$03   ;1D; (Top Entry Room)                      Red
      ;.byte $B2,$FA,$66,$0A,$21,$06,$01,$06,$03   ;1E; Name Room                             Purple

      ;Original Adventure Values - For Reference
      ;.byte $DC,$FA,$22,$0A,$21,$00,$00,$00,$00   ;00; Number Room.                           Purple
      ;.byte $B2,$FA,$22,$0A,$A1,$08,$02,$80,$03   ;01; (Top Access) Reflected/8 Clock Ball
      ;.byte $B2,$FA,$20,$0A,$21,$11,$03,$83,$01   ;02; (Top Access) below yellow castle       Green
      ;.byte $A0,$FA,$20,$0A,$61,$06,$01,$86,$02   ;03; Left of Name
```

```
      ;.byte $3E,$FB,$80,$0A,$21,$10,$05,$07,$06   ;04; Top of Blue Maze                        Blue
      ;.byte $53,$FB,$80,$0A,$21,$1D,$06,$08,$04   ;05; Blue Maze #1                           Blue
      ;.byte $68,$FB,$80,$0A,$21,$07,$04,$03,$05   ;06; Bottom of Blue Maze                    Blue
      ;.byte $7D,$FB,$80,$0A,$21,$04,$08,$06,$08   ;07; Center of Blue Maze                    Blue
      ;.byte $92,$FB,$80,$0A,$21,$05,$07,$01,$07   ;08; Blue Maze Entry                        Blue
      ;.byte $A7,$FB,$08,$08,$25,$0A,$0A,$0B,$0A   ;09; Maze Middle
Invisible/Playfield Priority
      ;.byte $CE,$FB,$08,$08,$25,$03,$09,$09,$09   ;0A; Maze Entry
Invisible/Playfield Priority
      ;.byte $B9,$FB,$08,$08,$25,$09,$0C,$1C,$0D   ;0B; Maze Side
Invisible/Playfield Priority
      ;.byte $C7,$FA,$22,$0A,$21,$1C,$0D,$1D,$0B   ;0C; (Side Corridor)
      ;.byte $C7,$FA,$24,$0A,$21,$0F,$0B,$0E,$0C   ;0D; (Side Corridor)
      ;.byte $74,$FC,$80,$0A,$21,$0D,$10,$0F,$10   ;0E; (Top Entry Room)
      ;.byte $E3,$FB,$0E,$0C,$21,$0E,$0F,$0D,$0F   ;0F; White Castle                          White
      ;.byte $E3,$FB,$00,$02,$21,$01,$1C,$04,$1C   ;10; Black Castle                          Black
      ;.byte $E3,$FB,$18,$0A,$21,$06,$03,$02,$01   ;11; Yellow Castle                         Yellow
      ;.byte $DC,$FA,$12,$0A,$21,$12,$12,$12,$12   ;12; Yellow Castle Entry                   Yellow
      ;.byte $89,$FC,$08,$08,$25,$15,$14,$15,$16   ;13; Black Maze #1
      ;.byte $B0,$FC,$08,$08,$24,$16,$15,$16,$13   ;14; Black Maze #2
      ;.byte $9B,$FC,$08,$08,$24,$13,$16,$13,$14   ;15; Black Maze #3
      ;.byte $C2,$FC,$08,$08,$25,$14,$13,$1B,$15   ;16; Black Maze Entry
      ;.byte $26,$FC,$36,$0A,$21,$19,$18,$19,$18   ;17; Red Maze #1                           Red
      ;.byte $4D,$FC,$36,$0A,$21,$1A,$17,$1A,$17   ;18; Top of Red Maze                       Red
      ;.byte $38,$FC,$36,$0A,$21,$17,$1A,$17,$1A   ;19; Bottom of Red Maze                    Red
      ;.byte $5F,$FC,$36,$0A,$21,$18,$19,$18,$19   ;1A; White Castle Entry                    Red
      ;.byte $29,$FB,$36,$0A,$21,$89,$89,$89,$89   ;1B; Black Castle Entry                    Red
      ;.byte $DC,$FA,$66,$0A,$21,$1D,$07,$8C,$08   ;1C; Bottom Entry - Other Purple Room     Purple
      ;.byte $74,$FC,$36,$0A,$21,$8F,$01,$10,$03   ;1D; (Top Entry Room)                     Red
      ;.byte $B2,$FA,$66,$0A,$21,$06,$01,$06,$03   ;1E; Name Room                            Purple


;Room differences for different levels (level 1,2,3)

LFF32: .byte $10,$0F,$0F         ;Down from Room 01
       .byte $05,$11,$11         ;Down from Room 02
       .byte $1D,$0A,$0A         ;Down from Room 03
       .byte $1C,$16,$16         ;U/L/R/D from Room 1B (Black Castle Room)
       .byte $1B,$0C,$0C         ;Down from Room 1C
       .byte $03,$19,$19         ;Up from Room 1D (Top Entry Room)

;Objects

        ;Offset 0 : Low byte object information (moveable stuff)
        ;Offset 1 : High byte object information (moveable stuff)
        ;Offset 2 : Low byte to object's current state
        ;Offset 3 : High byte to object's current state
        ;Offset 4 : Low byte list of states
        ;Offset 5 : High byte list of states
        ;Offset 6 : Colour
        ;Offset 7 : Colour in B&W.
        ;Offset 8 : Size of object

LFF44: .byte $D9       ;0 Invisible Surround Offsets..0
LFF45: .byte $00       ;1
LFF46: .byte $01       ;2
LFF47: .byte $FC       ;3
LFF48: .byte $02       ;4
LFF49: .byte $FC       ;5
LFF4A: .byte $A0       ;6
LFF4B: .byte $0C       ;7
LFF4C: .byte $07       ;8

      .byte $F8,$FB,$C8,$00,$F1,$FA,$00,$00,$00   ;#1 Portcullis #1            Black   9
      .byte $FB,$FB,$C9,$00,$F1,$FA,$00,$00,$00   ;#2 Portcullis #2            Black   12
      .byte $FE,$FB,$CA,$00,$F1,$FA,$00,$00,$00   ;#3 Portcullis #3            Black   1B
      .byte $E8,$FD,$EB,$FD,$EC,$FD,$10,$00,$00   ;#4 Name                     Flash   24
      .byte $02,$FE,$DD,$00,$05,$FE,$10,$00,$00   ;#5 Number                   Green   2D
```

```
      .byte $A4,$00,$A8,$00,$2E,$FD,$36,$0E,$00   ;#6 Dragon #1                Red     36
      .byte $A9,$00,$AD,$00,$2E,$FD,$1A,$06,$00   ;#7 Dragon #2                Yellow  3F
      .byte $AE,$00,$B2,$00,$2E,$FD,$C8,$00,$00   ;#8 Dragon #3                Green   48
      .byte $B6,$00,$78,$FD,$79,$FD,$1E,$06,$00   ;#9 Sword                    Yellow  51
      .byte $BC,$00,$D7,$FC,$D8,$FC,$9E,$02,$07   ;#0A Bridge                  Purple  5A
      .byte $BF,$00,$FC,$FC,$FD,$FC,$18,$06,$00   ;#0B Key #01                 Yellow  63
      .byte $C2,$00,$FC,$FC,$FD,$FC,$0E,$0E,$00   ;#0C Key #02                 White   6C
      .byte $C5,$00,$FC,$FC,$FD,$FC,$00,$00,$00   ;#0D Key #03                 Black   75
      .byte $CB,$00,$CF,$00,$14,$FD,$00,$00,$00   ;#0E Bat                     Black   7E
      .byte $A1,$00,$82,$FD,$83,$FD,$1E,$08,$00   ;#0F Black Dot               Lt Gray 87
      .byte $B9,$00,$EF,$FD,$F0,$FD,$CB,$06,$00   ;#10 Chalice                 Flash   90
      .byte $B3,$00,$0E,$FE,$0F,$FE,$00,$06,$00   ;#11 Magnet                  Black   99
      .byte $BC,$00,$FD,$FD,$FE,$FD,$00,$00,$00   ;#12 Null                    Black   A2

;Not Used
      .byte $00,$00,$00,$00,$00,$00,$00,$00,$00,$00,$00
;6502 Vectors (Not Used??)
      .byte $00,$F0
      .byte $00,$F0
      .byte $00,$F0
```