```
;=============================================================================
;
CHAR PAD TOOLS
;=============================================================================
;
Peter 'Sig' Hewett 2017
;-----------------------------------------------------------------------------
; Tools for integrating CharPad character sets, tiles, and maps
;=============================================================================
; Map Notes:
;
; CharPad setup : To make a level map, set up for 256 characters
;                 A tile size of 4x4
;                 Set number of tiles to 64
;                 Set map size to 64 x 32
;                 *IMPORTANT* set color to 'per character'
;
; This is unlikely to change, although I want to add an option for 'portrait'
; dimensons of 32 x 64 as a feature once things are working propery.
; This gives me a good sized map with some nice tricks I can pull with the math
; to keep things from bogging down, also makes both charset and map 2048 bytes,
; so copying them around in memory is very easy.
; The per-character color is important as the new collision system will be taking
; advantage of the materials system stored in the upper half of the color byte info.
; This should allow for easier level building.
;
;-----------------------------------------------------------------------------
;
LOAD LEVEL
;-----------------------------------------------------------------------------
; Loading a level and getting it ready for play
;-----------------------------------------------------------------------------
#region "LoadLevel"
LoadLevel
```

```
        ldx CURRENT_LEVEL               ; Load current level
number
        dex                             ; subtract 1 (levels
start at 1 not 0)
        txa
        lsr                             ; multiply by 2 to get a
word offset
        tax                             ; put it in x to lookup
our table
        sta PARAM5                      ; save in scratch param
(5)


        ;-------------------------------------------------------
----------
        ; Here we load all level data into the area below VIC
mem ($8000)
        ; To setup the level

        ;-------------------------------------------------
Loading banked data
@loadBanked

        sei

        lda PROC_PORT
        sta PARAM1

        lda #%00110000                  ; Switch out BASIC,
KERNAL, CHAREN, IO
        sta PROC_PORT

        ldx PARAM5
        lda CHAR_ADDRESS,x
        sta ZEROPAGE_POINTER_1
        lda CHAR_ADDRESS + 1,x
        sta ZEROPAGE_POINTER_1 + 1

        loadPointer ZEROPAGE_POINTER_2, CHAR_MEM

        jsr CopyChars

        ldx PARAM5
        lda MAP_ADDRESS,x
        sta ZEROPAGE_POINTER_1
        lda MAP_ADDRESS + 1,x
        sta ZEROPAGE_POINTER_1 + 1
```

```
        loadPointer ZEROPAGE_POINTER_2, MAP_MEM
        jsr CopyChars

        lda PARAM1
        sta PROC_PORT

        cli


        rts
#endRegion
;-------------------------------------------------------------------
------------------
;
DRAW MAP
;-------------------------------------------------------------------
------------------
; Draw the entire map on the screen. This won't be done that
often as most updates
; to the screen will be scrolling. But for starting a level,
resetting on death,
; or teleporting, we need to build the entire screen.
; This also sets up essential data for using the map.
; Initializes : MAP_POS_X
;               MAP_POS_Y
;               MAP_POS_ADDRESS
;               MAP_X_DELTA
;               MAP_Y_DELTA
;
; X = Start map X coord (top left corner)
; Y = Start map Y coord (top left corner)
;
; Uses ZEROPAGE_POINTER_4 (as TileDraw uses 1,2,3)
;-------------------------------------------------------------------
------------------
#region "DrawMap"
DrawMap

        lda #0
        sta MAP_X_DELTA
        sta MAP_Y_DELTA

        stx MAP_X_POS
        sty MAP_Y_POS
```

```
        ;-------------------------------------------------------
--------
        ; First find the address for the starting map position

        ldx MAP_Y_POS

        lda MAP_LINE_LOOKUP_LO,x        ; fetch the address for
the line (Y pos)
        sta ZEROPAGE_POINTER_4
        lda MAP_LINE_LOOKUP_HI,x
        sta ZEROPAGE_POINTER_4 + 1

        clc
        lda ZEROPAGE_POINTER_4          ; add the x position
        adc MAP_X_POS
        sta ZEROPAGE_POINTER_4
        lda ZEROPAGE_POINTER_4 + 1
        adc #0
        sta ZEROPAGE_POINTER_4 + 1      ; ZEROPAGE_POINTER_1 now
holds the map start address

                                        ; Save this info for map
usage
        copyPointer ZEROPAGE_POINTER_4, MAP_POS_ADDRESS
        ;-------------------------------------------------------
-----------------------------
        ; Fetch map data and draw tile - coords are in 'tiles'
not character positions

        ldy #0                  ; holds X screen coord
        ldx #0                  ; holds Y screen coord

@loop
        lda (ZEROPAGE_POINTER_4),y              ; fetch map data
        jsr DrawTile                            ; draw the tile

        iny                                     ; inc X and
check for end of screen
        cpy #10                                 ; (10 tiles)
        bne @loop
                                                ; go down one
line on the map (64 char)
        addPointer ZEROPAGE_POINTER_4, 64
        ldy #0
        inx
        cpx #5
        bne @loop
```

```
        rts
#endregion

;-------------------------------------------------------------
------------------
;
DRAW TILE
;-------------------------------------------------------------
------------------
; This routine actually won't be called that often. Most updates
are scrolling,
; so there are very few circumstances that need a whole tile
drawn at once.
;-------------------------------------------------------------
------------------
; After 5 rewrites of this routine I'm actually fairly happy
with it.
; It keeps registers intact. For the amount of work it's doing
it doesn't
; overly rely on PARAM variables. If needed I could pull the
variables off the
; stack and eliminate the PARAM saving altogether, at the
expense of returning clean
;
; X = Screen tile Y coord    - 'flipped' so it dovetails into
the MapDraw
; Y = Screen tile X coord       routine without exhanging data
in registers
; A = Tile # to draw
;
; Restores registers off the stack A / X / Y
;-------------------------------------------------------------
------------------

#region "DrawTile"
DrawTile
        sta PARAM1                      ; save tile number
        sty PARAM2                      ; save X pos
        stx PARAM3                      ; save Y pos

        saveRegs                        ; put registers on the
stack to
                                        ; exit cleaner - this
routine will
                                        ; likely be nested
```

```
        ;-------------------------------------------------
        ; First get the destination for the tile

        lda PARAM3                      ; fetch the Y pos (in
tile coords)
        asl
        asl                             ; multiply by 4 (tiles
are 4 x 4 chars)
        tax                             ; screen line in X

        jsr FetchScreenLineAddress      ; fetch line address
based on current displayed screen
                                        ; Y line address is in
ZEROPAGE_POINTER_1


        lda COLOR_LINE_OFFSET_TABLE_LO,x        ; fetch color
ram line address too
        sta ZEROPAGE_POINTER_3
        lda COLOR_LINE_OFFSET_TABLE_HI,x
        sta ZEROPAGE_POINTER_3 + 1

        lda PARAM2                      ; get X coord
        asl                             ; multiply by 4
        asl
        tax                             ; save it in x

        clc                             ; add to Y line address
        adc ZEROPAGE_POINTER_1
        sta ZEROPAGE_POINTER_1
        lda ZEROPAGE_POINTER_1 + 1
        adc #0
        sta ZEROPAGE_POINTER_1 + 1      ; destination base
address is in ZEROPAGE_POINTER_1

        txa
        clc
        adc ZEROPAGE_POINTER_3          ; color ram destination
is in ZEROPAGE_POINTER_3
        sta ZEROPAGE_POINTER_3
        lda ZEROPAGE_POINTER_3 + 1
        adc #0
        sta ZEROPAGE_POINTER_3 + 1

        ;-------------------------------------------------
-----
        ; Fetch the source tile address
```

```
        ldx PARAM1                          ; Fetch the tile number

        lda TILE_NUMBER_LOOKUP_LO,x
        sta ZEROPAGE_POINTER_2
        lda TILE_NUMBER_LOOKUP_HI,x
        sta ZEROPAGE_POINTER_2 + 1

        ;-------------------------------------------------------
------
        ; Loop through and draw the tile

        ldy #0

@drawloop
        lda (ZEROPAGE_POINTER_2),y      ; Get the character code
        sta (ZEROPAGE_POINTER_1),y      ; store it on the screen
        tax                             ; pass to X as an offset
        lda ATTRIBUTE_MEM,x             ; fetch the color/data
attribute
;        lda #COLOR_WHITE
        sta (ZEROPAGE_POINTER_3),y      ; write it to color ram

        cpy #15                         ; drawn the 15th
character? We're finished
        beq @done

        tya                             ; save Y before the
increment for our test
        iny                             ; (saves having to inc
it in 2 diff places)

                                        ; I need to count 0-3 to
draw a row of tiles,
        and #%00000011                  ; but I need to count 0-
15 to fetch the tile data
        cmp #3                          ; both NEED to use
indirect Y addressing, and saving/
        bne @drawloop                   ; fetching Y rapidly
becomes a tangled nightmare.
                                        ; by masking out the
last 2 bits in A, we get a number
                                        ; that counts 0-3 over
and over without stopping the
                                        ; data fetch count 0-15.
It's also faster than my other
```

```
                                        ; options by quite a
bit.


        clc                             ; add new line
        lda ZEROPAGE_POINTER_1          ; increment destination
and color ram by 1 line - 4 chars
        adc #40 - 4
        sta ZEROPAGE_POINTER_1          ; by 'backsetting' our
pointers, we don't need to change Y
        lda ZEROPAGE_POINTER_1 + 1      ; when drawing 0-3
characters, and can leave the 0-15
        adc #0                          ; count intact.
        sta ZEROPAGE_POINTER_1 + 1      ; We have to increase
them to the next line anyways, so this
                                        ; only saves time.
        clc
        lda ZEROPAGE_POINTER_3
        adc #40 - 4
        sta ZEROPAGE_POINTER_3
        lda ZEROPAGE_POINTER_3 + 1
        adc #0
        sta ZEROPAGE_POINTER_3 + 1

        jmp @drawloop
@done
        restoreRegs             ; pull registers back off the
stack

        rts

#endRegion

;-------------------------------------------------------------------
-----------------------------------
;
GET MAP LINE ADDRESS
;-------------------------------------------------------------------
-----------------------------------
; A = line you want to fetch
;
; Result : Address passed back in WPARAM1
;-------------------------------------------------------------------
-----------------------------------
#region "MapLineAddress"
MapLineAddress
        tax
```

```
        lda MAP_LINE_LOOKUP_LO,x
        sta WPARAM1
        lda MAP_LINE_LOOKUP_HI,x
        sta WPARAM1
        rts
#endRegion


;=====================================================================
====================================
;
LEVEL DATA AND TABLES
;=====================================================================
====================================
;---------------------------------------------------------------------
----------------------------------
;
LEVEL ADDRESSES AND DATA
;---------------------------------------------------------------------
----------------------------------
; Dummy 1st level values for now, to make it easier to
incorporate more levels and loading later
;---------------------------------------------------------------------
----------------------------------

CURRENT_LEVEL
        byte 0

CHAR_ADDRESS
        word LEVEL_1_CHARS

ATTRIB_ADDRESS
        word ATTRIBUTE_MEM

TILE_ADDRESS
        word TILE_MEM

MAP_ADDRESS
        word LEVEL_1_MAP
;---------------------------------------------------------------------
----------------------------------
;
MAP DATA LOOKUP TABLE 1
;---------------------------------------------------------------------
----------------------------------
; Lookup table to return an address to a map line (Y coord).
This table assumes a landscape
```

```
; layout (64 x 32 tiles). A portrait style table will be done in
the future to allow more
; vertical maps
;-------------------------------------------------------------------
-----------------------------------
MAP_LINE_LOOKUP_LO
        byte <MAP_MEM
        byte <MAP_MEM + 64
        byte <MAP_MEM + 128
        byte <MAP_MEM + 192
        byte <MAP_MEM + 256
        byte <MAP_MEM + 320
        byte <MAP_MEM + 384
        byte <MAP_MEM + 448
        byte <MAP_MEM + 512
        byte <MAP_MEM + 576
        byte <MAP_MEM + 640             ; 10
        byte <MAP_MEM + 704
        byte <MAP_MEM + 768
        byte <MAP_MEM + 832
        byte <MAP_MEM + 896
        byte <MAP_MEM + 960
        byte <MAP_MEM + 1024
        byte <MAP_MEM + 1088
        byte <MAP_MEM + 1152
        byte <MAP_MEM + 1216
        byte <MAP_MEM + 1280            ;20
        byte <MAP_MEM + 1344
        byte <MAP_MEM + 1408
        byte <MAP_MEM + 1472
        byte <MAP_MEM + 1536
        byte <MAP_MEM + 1600
        byte <MAP_MEM + 1664
        byte <MAP_MEM + 1728
        byte <MAP_MEM + 1792
        byte <MAP_MEM + 1856
        byte <MAP_MEM + 1920            ;30
        byte <MAP_MEM + 1984
        byte <MAP_MEM + 2048            ;32

MAP_LINE_LOOKUP_HI
        byte >MAP_MEM
        byte >MAP_MEM + 64
        byte >MAP_MEM + 128
        byte >MAP_MEM + 192
        byte >MAP_MEM + 256
        byte >MAP_MEM + 320
```

```
        byte >MAP_MEM + 384
        byte >MAP_MEM + 448
        byte >MAP_MEM + 512
        byte >MAP_MEM + 576
        byte >MAP_MEM + 640             ; 10
        byte >MAP_MEM + 704
        byte >MAP_MEM + 768
        byte >MAP_MEM + 832
        byte >MAP_MEM + 896
        byte >MAP_MEM + 960
        byte >MAP_MEM + 1024
        byte >MAP_MEM + 1088
        byte >MAP_MEM + 1152
        byte >MAP_MEM + 1216
        byte >MAP_MEM + 1280            ;20
        byte >MAP_MEM + 1344
        byte >MAP_MEM + 1408
        byte >MAP_MEM + 1472
        byte >MAP_MEM + 1536
        byte >MAP_MEM + 1600
        byte >MAP_MEM + 1664
        byte >MAP_MEM + 1728
        byte >MAP_MEM + 1792
        byte >MAP_MEM + 1856
        byte >MAP_MEM + 1920            ;30
        byte >MAP_MEM + 1984
        byte >MAP_MEM + 2048            ;32

;-------------------------------------------------------------------
-----------------------------------
;
TILE ADDRESS LOOKUP TABLE
;-------------------------------------------------------------------
-----------------------------------
; Lookup table to find the start address of a tile on the
current map. All current level tiles are
; held in TILE_MEM, there are 64 entries
;-------------------------------------------------------------------
-----------------------------------

TILE_NUMBER_LOOKUP_LO
        byte <TILE_MEM                  ; 0
        byte <TILE_MEM + 16
        byte <TILE_MEM + 32
        byte <TILE_MEM + 48
        byte <TILE_MEM + 64
        byte <TILE_MEM + 80
```

```
byte <TILE_MEM + 96
byte <TILE_MEM + 112
byte <TILE_MEM + 128
byte <TILE_MEM + 144
byte <TILE_MEM + 160            ; 10
byte <TILE_MEM + 176
byte <TILE_MEM + 192
byte <TILE_MEM + 208
byte <TILE_MEM + 224
byte <TILE_MEM + 240
byte <TILE_MEM + 256
byte <TILE_MEM + 272
byte <TILE_MEM + 288
byte <TILE_MEM + 304
byte <TILE_MEM + 320            ; 20
byte <TILE_MEM + 336
byte <TILE_MEM + 352
byte <TILE_MEM + 368
byte <TILE_MEM + 384
byte <TILE_MEM + 400
byte <TILE_MEM + 416
byte <TILE_MEM + 432
byte <TILE_MEM + 448
byte <TILE_MEM + 464
byte <TILE_MEM + 480            ; 30
byte <TILE_MEM + 496
byte <TILE_MEM + 512
byte <TILE_MEM + 528
byte <TILE_MEM + 544
byte <TILE_MEM + 560
byte <TILE_MEM + 576
byte <TILE_MEM + 592
byte <TILE_MEM + 608
byte <TILE_MEM + 624
byte <TILE_MEM + 640            ; 40
byte <TILE_MEM + 656
byte <TILE_MEM + 672
byte <TILE_MEM + 688
byte <TILE_MEM + 704
byte <TILE_MEM + 720
byte <TILE_MEM + 736
byte <TILE_MEM + 752
byte <TILE_MEM + 768
byte <TILE_MEM + 784
byte <TILE_MEM + 800            ;50
byte <TILE_MEM + 816
byte <TILE_MEM + 832
```

```
        byte <TILE_MEM + 848
        byte <TILE_MEM + 864
        byte <TILE_MEM + 880
        byte <TILE_MEM + 896
        byte <TILE_MEM + 912
        byte <TILE_MEM + 928
        byte <TILE_MEM + 944
        byte <TILE_MEM + 960            ; 10
        byte <TILE_MEM + 976
        byte <TILE_MEM + 992
        byte <TILE_MEM + 1008
        byte <TILE_MEM + 1024           ; 64

TILE_NUMBER_LOOKUP_HI
        byte >TILE_MEM                  ; 0
        byte >TILE_MEM + 16
        byte >TILE_MEM + 32
        byte >TILE_MEM + 48
        byte >TILE_MEM + 64
        byte >TILE_MEM + 80
        byte >TILE_MEM + 96
        byte >TILE_MEM + 112
        byte >TILE_MEM + 128
        byte >TILE_MEM + 144
        byte >TILE_MEM + 160            ; 10
        byte >TILE_MEM + 176
        byte >TILE_MEM + 192
        byte >TILE_MEM + 208
        byte >TILE_MEM + 224
        byte >TILE_MEM + 240
        byte >TILE_MEM + 256
        byte >TILE_MEM + 272
        byte >TILE_MEM + 288
        byte >TILE_MEM + 304
        byte >TILE_MEM + 320            ; 20
        byte >TILE_MEM + 336
        byte >TILE_MEM + 352
        byte >TILE_MEM + 368
        byte >TILE_MEM + 384
        byte >TILE_MEM + 400
        byte >TILE_MEM + 416
        byte >TILE_MEM + 432
        byte >TILE_MEM + 448
        byte >TILE_MEM + 464
        byte >TILE_MEM + 480            ; 30
        byte >TILE_MEM + 496
        byte >TILE_MEM + 512
```

```
byte >TILE_MEM + 528
byte >TILE_MEM + 544
byte >TILE_MEM + 560
byte >TILE_MEM + 576
byte >TILE_MEM + 592
byte >TILE_MEM + 608
byte >TILE_MEM + 624
byte >TILE_MEM + 640            ; 40
byte >TILE_MEM + 656
byte >TILE_MEM + 672
byte >TILE_MEM + 688
byte >TILE_MEM + 704
byte >TILE_MEM + 720
byte >TILE_MEM + 736
byte >TILE_MEM + 752
byte >TILE_MEM + 768
byte >TILE_MEM + 784
byte >TILE_MEM + 800            ;50
byte >TILE_MEM + 816
byte >TILE_MEM + 832
byte >TILE_MEM + 848
byte >TILE_MEM + 864
byte >TILE_MEM + 880
byte >TILE_MEM + 896
byte >TILE_MEM + 912
byte >TILE_MEM + 928
byte >TILE_MEM + 944
byte >TILE_MEM + 960            ; 10
byte >TILE_MEM + 976
byte >TILE_MEM + 992
byte >TILE_MEM + 1008
byte >TILE_MEM + 1024           ; 64
```