Linecrunch

Linecrunch is a VIC-tweak that allows you to move the entire screen upwards, by (as the name suggests) “crunching” 8-pixel char-lines down to a single pixel-line. The biggest advantage of this trick is the ability to move bitmap graphics, since is contains ~9 times the data of a char screen. Linecrunching requires fairly exact timing since the tweak has to be triggered during a short period of the raster line. For crunching more lines, the tweak has to be repeated at the same position each raster line.

Here follows a small code example. Note that the $d021-indicator, that shows approx where the tweak is triggered, is not stable. It doesn't need to be 100% cycle exact, just fair enough. Also note the 7 pixel area with $3fff-pattern just below the linecrunch area. This happens because y-scroll was set to trigger the linecrunch, and when left untouched there are 7 lines until badline is triggered again. Set $d011 to something else just when exiting the loop and the $3fff can be avoided, or smooth scrolling can be obtained instead.. Note also what happens on the linecrunch area: the last line of graphics is repeated. There are no badlines, so no new char info is being fetched. If we switch to bitmap mode (yes, you can do this yourself), you will see that the graphics is actually being crunched, but the color info (which is represented by the chars) is being corrupt.

Binary: linecrunch.zip

```
        sei
        lda #$aa ; Just to make 3fff visible
        sta $3fff
        lda #$17 ; Make letters like 'y','g','p' visible in the linecrunch
area
        sta $d018
loop1
        bit $d011 ; Wait for new frame
        bpl *-3
        bit $d011
        bmi *-3

        lda #$1b ; Set y-scroll to normal position (because we fcuk it up
later on..)
        sta $d011

        jsr CalcNumLines ; Call sinus substitute routine

        lda #$51 ; Wait for position where we want LineCrunch to start
        cmp $d012
        bne *-3

        ldy #10 ; Wait one more line..
        dey
        bne *-1
        nop
        nop
        cmp $d012 ; ..and make a bit more stabel raster
```

```
        bne *+5
        bit 0
        nop

        ldx NumCrunchLines
        beq loop1 ; Skip if we want 0 crunched lines
loop2
        ldy #4 ; Wait some cycles
        dey
        bne *-1
        ldy 0

        lda $d012 ; Do one line of LineCrunch
        and #7
        ora #$18
        inc $d021 ; d021-indicator
        dec $d021
        sta $d011

        nop ; Wait some more cycles so that the whole loop ends up on 63
cycles (= one PAL raster line)
        nop
        nop
        nop

        dex ; Decrease counter
        beq loop1 ; Exit if we reached 0
        jmp loop2 ; Otherwise loop

CalcNumLines
        lda #0
        bpl *+4
        eor #$ff
        lsr
        lsr
        lsr
        sta NumCrunchLines
        inc CalcNumLines+1
        rts

NumCrunchLines
        .byte 0
```

From:
https://codebase64.org/ - **Codebase 64 wiki**

Permanent link:
https://codebase64.org/doku.php?id=base:linecrunch

Last update: **2015-06-27 16:46**