FPD - Flexible Pixel Distance

FPD means the ability move a single pixel-line of graphics vertically on the screen. FPD can visually be compared to FLD, only that FPD is 8 times more beautiful, but technically it's quite different. You can also simulate FLD with an FPD-effect.

The invention of FPD probably origins from the crap gfx you get when you do a line-cruncher. Linecrunchers are mostly used to move big areas of bitmap-gfx on the screen, where as copying would take far too much time. The crunched lines stack up above the bitmap-area that you actually want to display, but those crunched lines can of course be used for something as well, and FPD is the effect that allows for here. By introducing gaps between the crunched lines (just like in FLD), you can control every individual pixel-line distance from the previous pixel-line.

Here follows a code example, which is based on the linecrunch example. Perhaps you'll better understand all details if you tried that one first. This routine does not really take every pixel-line and do desired amount of FLD on it, but rather it reads one byte per line from a table that tells wether this line should be displayed or not. If the line should not be displayed then a value is stored to $d011 that does not trigger any badline, linecrunch or anything, i call it an FLD-line. If a line should be displayed, then a value that triggers linecrunch is stored to $d011 at the right timing.

Binary: fpd.zip

```
       sei
loop1
        bit $d011 ; Wait for new frame
        bpl *-3
        bit $d011
        bmi *-3

        lda #$3b ; Set y-scroll to normal position
        sta $d011

        jsr SetupFPDTable ; Nice FPD effect

        lda #$59 ; Wait for position where we want FPD 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 #0 ; Clear counter
loop2
```

```
        ldy #5 ; Wait some cycles
        dey
        bne *-1
        nop
        nop
        nop

        lda FPDTable,x ; Load 0 or 1 from table, where 0 will skip a line
with FLD
                       ; and 1 will draw a line using linecrunch
        lsr ; 0 or 1 moves into carry flag
        lda $d012
        adc #7 ; Add 7 or 8 depending on carry
        and #7
        ora #$38
        sta $d011 ; Do one line of FLD or LineCrunch

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

        inx
        cpx #100
        beq loop1 ; Exit if end is reached
        jmp loop2 ; Otherwise loop

SetupFPDTable
        lda #0 ; First clear the table
        sta YPos
        ldy #0
        sta FPDTable,y
        iny
        bne *-4

        lda #0 ; Increase the starting value
        inc *-1
        asl
        sta AddVal

        ldy #0 ; This loop will insert 16 1:s into the table..
               ; At those positions the graphics will be displayed
SFT_1
        lda AddVal
        clc
        adc #10
        sta AddVal
        bpl *+4
        eor #$ff
        lsr
        lsr
        lsr
        lsr
```

```
        sec
        adc YPos
        sta YPos
        tax
        lda #1
        sta FPDTable,x
        iny
        cpy #16
        bcc SFT_1
        rts

YPos    .byte 0
AddVal  .byte 0

        .align $100 ; Align the table to a new page, this way lda FPDTable,x
always takes 4 cycles.
FPDTable
        .dsb 100,0 ; Reserve 100 bytes for the table
```