Sprite Crunching

Sprite crunching is a way to shrink sprites in y-size, but it is rather complicated. Actually it is possible to stretch sprites with the same technique, but that can be done much easier with normal $d017-stretching. This is nothing you do if you don't *really* need it.

Sprite crunching is based on a glitch in the VIC-chip (as usual) that appears on a cycle-exact position every rasterline. You trigger the glitch by setting back $d017 to 0 on this exact position, and all sprites that had their $d017-bit set to 1 will be affected.

Now what will happen depends on where in the sprite you already are. There is an internal sprite-graphics-counter for each sprite that, in the normal case, is increased by 3 each line the sprite is displayed. Each rasterline 3 bytes of the sprite will be displayed, so naturally the counter is increased by 3. This counter can have values from 0 to $3f. Depending on what value it has, different things will happen when triggering the sprite crunching glitch. Sometimes 3 will be added (as normal), sometimes something else. Sometimes a negative value will be added(!).

When the sprite-graphics-counter reaches $3f, the sprite will end. Normally this happens after 21 displayed lines. However if using the glitch results in adding anything not dividable by 3, the counter will not end up on $3f, but $3d or $3e instead. On the next line it will pass $3f and wrap around to $00 or $01, and the sprite will continue to be displayed for some 21 more lines (though the graphics will be mis-aligned of course).

Examples of usage:

In Rutig Banan/FLT sprite crunching is used on the third sprite-line, adding -1 to the counter, which effectively causes the sprite to be displayed almost 3 times. Y-expanded sprites and toggling graphics bank gives you ~120 lines of sprite area to play with. The graphics is then copied to the right position to make the letters move up/down.

Krestage/Crest shows the first example of sprite crunching when used for shrinking sprites. There you can also find exact documentation of what happens on each value of the sprite-graphics-counter when triggering the glitch, hand-haxxored by Crossbow himself. Finding the (probably) shortest path through the table, he managed to shrink the sprites from 21 lines to 17. By using this table, you can sort out what possibilities there are to do something useful with this, depending what you're after.

In Edge of Disgrace/Booze Design sprite crunching is used in the 4×4-plasma as an alternative to a normal $d017-stretcher. Here the glitch is triggered on position $1e, making it add -9. This effectively jumps back 3 sprite-lines, repeating the graphics in a way that fits the 4×4-timing. So, instead of setting $d017 twice each line to repeat the graphics, it is set twice every 4 lines, saving some life supporting cycles in the timing.

To add a bit more content to this page, here is that table. You read it like “AA : BB → CC”, where AA is the current position in the sprite. BB is the value that will be added if triggering the glitch (+3 of no glitch), and CC is the resulting position. So, if you want to shrink your sprites, you should look for BB values > 3, and if you want to enlarge your sprite then look for BB values < 3. Don't forget that the sprite only ends on $3f, so the only ways to end the sprite are from $3b or $3c.

Now play away with it. Can you make a sprite smaller than 17 lines and beat Crossbow? ;D

```
00 : +1 -> 01
01 : +4 -> 05
02 : --------
03 : +4 -> 07
04 : +1 -> 05
05 : 0  -> 05
06 : -1 -> 05
07 : 0  -> 07
08 : +1 -> 09
09 : +4 -> 0d
0a : +3 -> 0d
0b : +4 -> 0f
0c : +1 -> 0d
0d : +8 -> 15
0e : +7 -> 15
0f : +8 -> 17
10 : +1 -> 11
11 : +4 -> 15
12 : +3 -> 15
13 : +4 -> 17
14 : +1 -> 15
15 : 0  -> 15
16 : -1 -> 15
17 : 0  -> 17
18 : +1 -> 19
19 : +4 -> 1d
1a : +3 -> 1d
1b : +4 -> 1f
1c : +1 -> 1d
1d : -8 -> 15
1e : -9 -> 15
1f : -8 -> 17
20 : +1 -> 21
21 : +4 -> 25
22 : +3 -> 25
23 : +4 -> 27
24 : +1 -> 25
25 : 0  -> 25
26 : -1 -> 25
27 : 0  -> 27
28 : +1 -> 29
29 : +4 -> 2d
2a : +3 -> 2d
2b : +4 -> 2f
2c : +1 -> 2d
2d : +8 -> 35
2e : +7 -> 35
2f : +8 -> 37
30 : +1 -> 31
31 : +4 -> 35
32 : +3 -> 35
```

```
33 : +4 -> 37
34 : +1 -> 35
35 : 0  -> 35
36 : -1 -> 35
37 : 0  -> 37
38 : +1 -> 39
39 : +4 -> 3d
3a : +3 -> 3d
3b : +4 -> 3f (end)
3c : +1 -> 3d
3d : -28 -> 15
3e : -29 -> 15
3f : end
```