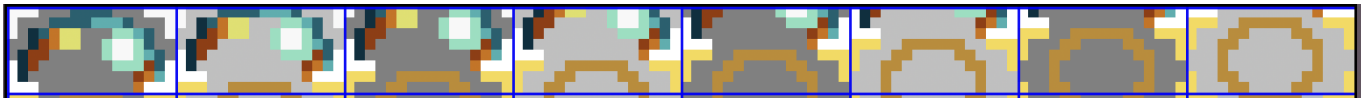
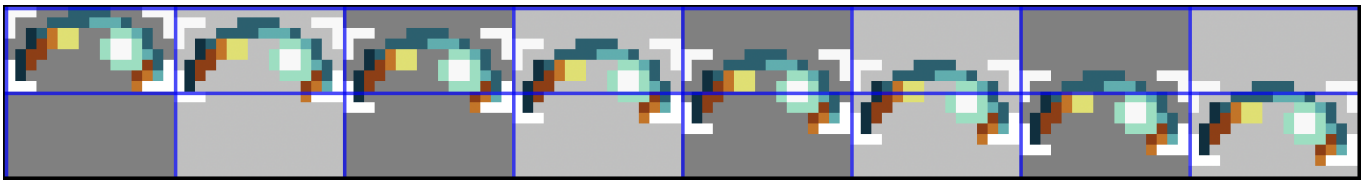
In order to be able to place Objs at arbitrary vertical positions we have to use the FCM character data Y offset (screenRam byte1: bits 5-7) feature of RRB, what this does is to add 8 bytes per unit to the character data being fetched. When drawing, it shifts the character data being fetched up one line per unit, a range of 0 to 7 pixels.



We also need to use the rowmask feature (enabled with colorRam byte0:bit3) and the mask is specified with colorRam byte1:bit0-7.

A ***true bit WILL write*** that line of character data
A ***false bit WILL NOT write*** that line.

When placing Objs vertically we want it to appear shifted down for each sub-row position (ypos & 7) like in the following image, also note that the screen row we want to write into is just (ypos >> 3), being careful to not write out of the bounds of the screen data.



In order to place an Obj vertically like the above picture you can use the following table to see which character indexes, y offsets and rowmasks you should use for the initial row and the following row.

Because we can only offset the character data up, we have to do a bit of hoop jumping when selecting the char index and offsets.

YShift = +ve y offset actions

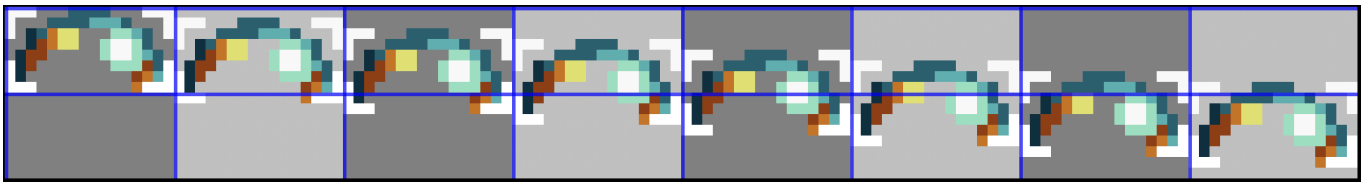
chrPtr = chrPtr + (y offset * 8)

	line 0			**line 1**		
(ypos & 7)	chrPtr	+ve y offset	rowmask	chrPtr	+ve y offset	rowmask
0	chr	0	%11111111	chr+1	0	%00000000
1	chr-1	7	%11111110	chr	7	%00000001
2	chr-1	6	%11111100	chr	6	%00000011
3	chr-1	5	%11111000	chr	5	%00000111

	line 0			line 1		
4	chr-1	4	%11110000	chr	4	%00001111
5	chr-1	3	%11100000	chr	3	%00011111
6	chr-1	2	%11000000	chr	2	%00111111
7	chr-1	1	%10000000	chr	1	%01111111

(Future proposal - allowing -ve FCM character data Y offset)

If another bit was added to GOTOX set features that allowed us to shift character data negatively, you can see from the following table that the chrPtr and shifts are more consistent and would make it easier to implement Objs.



The shifting down of line 1 (chr index + 1) will reveal the remaining part of the Obj from line 0.

YShift = -ve y offset actions

chrPtr = chrPtr - (y offset * 8)

	line 0			line 1		
(ypos & 7)	chrPtr	-ve y offset	rowmask	chrPtr	-ve y offset	rowmask
0	chr	0	%111111111	chr+1	0	%00000000
1	chr	1	%11111110	chr+1	1	%00000001
2	chr	2	%11111100	chr+1	2	%00000011
3	chr	3	%11111000	chr+1	3	%00000111
4	chr	4	%11110000	chr+1	4	%00001111
5	chr	5	%11100000	chr+1	5	%00011111
6	chr	6	%11000000	chr+1	6	%00111111
7	chr	7	%10000000	chr+1	7	%01111111